Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Department of Information Engineering

Masters course in
Control systems engineering

Generative algorithms for myocardial scar tissue structures

Supervisor: Prof./ Dr. Gian-Antonio Susto

Graduate student: Victor Hans Bayer Puerta

Correlator: Dr. Jose Maria Pozo

ACADEMIC YEAR 2024/25

Graduation date 27.03.2025

Acknowledgments

This work is supported by ELEM Biotech, a biomedical simulation spin-off from the Barcelona Supercomputing Center. I want to thank my supervisor at ELEM, Jose M. Pozo, for his invaluable guidance during the project. Thank you for letting me experiment with (sometimes misguided) approaches and sharing your expertise with me. I also want to express gratitude to my work colleagues, thanks to whom coming to work was never a chore but a pleasure. I greatly enjoyed the atmosphere and weekly snacks and miss our conversation about engineering topics and problem solving. Thanks for supporting me with your expertise in computer systems. I also want to thank Christopher Morton and Mariano Vázquez, CEO and CTO of ELEM Biotech for giving me the opportunity at their company. Further thanks go to my supervisor Prof. Susto from the university of Padova for his reviews and his easy-going nature in bureaucratic manners. Special thanks go to my family, who always support me in difficult situations.

This work is splitted into two parts. In part one i provide background information on used software, the clinical features a user might care about, machine learning algorithms, potential pitfalls and connections as well as code libraries. Part two contains the concrete contributions done through my work. It starts with a summary of already implemented algorithms which are used for obtaining the mesh data and mapping to a cylindrical representation. These parts have not been implemented by me but serve to provide context, especially to users of the software, since they form part of the "MRI to generation" pipeline. My contributions are mainly in making a motivated choice for a suitable data representation and algorithm, doing experiments with them, solving the arising problems and providing the prototype of an applicable generative process.

Nomenclature

MLE - Maximum likelihood estimation

MAP - Maximum a-posterior

PCA - Principal component analysis

VAE - Variational autoencoder

GAN - Generative adversarial network

D_{KL} - Kullback-Leibler divergence

ELBO - Evidence lower bound

EM - Expectation maximization

MRI - Magnetic resonance imaging

CT - Computed tomography

AHA - American heart association

UVC - Universal ventricular coordinates

LV - Left ventricle

RV - Right ventricle

LCX - Circumflex branch of the left coronary artery

LCA - Left anterior descending coronary artery

RCA - Right coronary artery

IHD - Ischemic heart disease

ACD - Atherosclerotic cardiovascular disease

WHO - World health organization

ATP - Adenosine triphosphate

B-spline - Basis spline

VTK - Visualization toolkit

SDF - Signed distance function

GP - Gaussian Process

Contents

Chapter 1

Motivation

The incorporation of engineering methodologies, such as simulations, machine learning, and software-development, into modern healthcare systems offers potential benefits in several areas. Past engineering innovations have already revolutionized modern healthcare to a nearly magical extent by enabling 3D scans of patient-specific anatomies with tomographic technologies. Already, algorithms aid in diagnosis [78], drug discovery [35], and therapy outcome prediction [63]. Already, simulations, computer vision, and robotics have become vital parts of surgery. The development of computational sciences has initiated in-silico medicine [81], allowing for a powerful analysis of diseases, treatments, drugs, and further via computer simulations.By leveraging medical imaging technology, computational models enable an in-depth analysis of blood flow [40], tissue structures [28] and underlying physical processes [68] [43]. Examples are the examination of electrical patterns in the Atria [73], mechanical and electro-physiological properties related to tachycardia [85], graph neural networks for blood pressure pulses [74] and simulation of cardiac stimulation by pacemakers [19]. It is possible to differentiate between patient-specific [42] and population-based approaches [1]. The former aims to aid diagnosis and treatment of a specific individual. The latter analyzes whole groups. By simulating drugs on computer models of representative human populations, animal testing can be reduced, enabling faster, cheaper, and more ethical pipelines. Furthermore, in silico models enable the representation of otherwise underrepresented patient groups (such as women and/or children with ischemia). Regarding recent technological development, artificial intelligence has become a technology of major interest due to its predictive and generative abilities. While powerful, these models are often black-boxes and hard to interpret, which is a major limiting factor, especially in safety-critical systems. The last few years have shown great promise with regard to generative algorithms [71], a field which gained momentum by the recent success of Large Language Models [75]. Generative applications range from image generation [18] to prompt-conditioned language generation [36], molecule generation [32] and more. In this work we examine generative algorithms for 3D-meshes corresponding to organic geometries. Similarly to algorithms for face generations [72], we aim to create a generalized population of plausible ischemic scar geometries. Heart meshes with respective scars model infarcted hearts and are suited for the computer simulation of physical properties and therapy responses. As such, we enable the representation of a general infarcted heart population which can be used to examine the effects of scar shape, demographic groups and therapy methods. This work is supported by ELEM Biotech, a biomedical simulation spin-off from the Barcelona Super-computing Center. ELEM models virtual cardiovascular and respiratory systems for specific populations and with desired clinical conditions and treatments, based on biophysical simulations. These services include applications in drug delivery via airways [11], aortic valve geometries and their effects [56], examining cardio-toxicity of drugs [1] and more.

1.1 Problem formulation

Aim of this work is to generate myocardial scar-tissue structures as they occur after a heart attack. The generated data can be used to gain insights with in-silico methods. We orient our problem in the area of generative machine learning algorithms for 3D shapes [30]. The developed software can be incorporated into ELEM's simulation software [80] and enables the examination of user-defined ischemic cardio-myopathies. By creating a set of plausible scars one can evaluate the resulting bio-physical properties via simulation. The main questions addressed in this work are:

- What kind of features are most relevant?
- What is an adequate shape representation?
- Which algorithm is best suited for shape generation?

This work is separated in two parts. Part one provides the background information on used software, cardiology, shape representations and machine learning. Part two explains the data-acquisition, processing and generative algorithm. This involves motivating the chosen representation and algorithm as well as outlining encountered difficulties and solutions. We start by explaining the used software in chapter 2. An introduction into cardiology, ischemia and relevant clinical features is provided in chapter 3. Examinations of common shape representations are found in chapter 4. Chapter 5 provides an overview over (generative) machine learning. It ends with the examination of specific algorithms which are potentially suitable for shape generations. These chapters provide the necessary background information. Chapter 7 explains the data-acquisition process from MRI scans to volumetric 3D data. Chapter 8 examines the chosen data representation. Chapter 9 provides a visual analysis of the data and occurring patterns. This chapter ends by providing a set of generations. Chapter 10 summarizes our work and looks at future directions.

Chapter 2

Part 1: Software

2.1 The Visualization Toolkit

The Visualization Toolkit (VTK) is a C++ class library which offers several objects for the manipulation of data. It is the companion software to a textbook called "The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics" [69], which was written in the 20-th century and offers perspective on the field of visualization. As very successful software it is widely used in 3D computer graphics and data visualization. Among its capabilities is efficient handling of scalars, vectors, tensors, volumetric methods as well as implicit fields and unstructured grid meshes. Methods contained in VTK are triangulation, contouring, mesh smoothing and more. Its main features are

- data objects, such as finite element meshes, or unstructured grids for data processing and analysis
- data arrays, which define discrete field values over any dataset (grid, mesh, points set). For example, pressure, temperature, velocity vectors, stress tensors
- filters for data processing and analysis which enable, for instance, smoothing, decimation, surface reconstruction, geometric transformations, substructures selection, implicit modeling
- advanced rendering techniques
- parallelizability of some functionalities for high performance computing
- an API for C++ and Python

Within the context of this work, VTK methods are used extensively to manipulate data and create visualizations within Paraview.

2.2 Paraview

Paraview is an open-source platform for interactive data visualization focused on simulation and high performance computing. It is built on top of the Visualization Toolkit (VTK) library. Among its functionalities is the analysis of extremely large data-sets, rendering and simulation results. Due to its parallel processing capabilities it is used to visualize tera-scale data which is often required for CFD-simulations. Furthermore, it runs on windows, mac-OS as-well as on linux. This makes it a multi-platform visualization application with an extensive range of use-cases. Examples are the analysis of climate-data [50] (see figure 2.3) and automotive aerodynamics [29] (see figure 2.2). The Data used in this work corresponds to a human heart

Figure 2.1: Exemplary Applications of Paraview

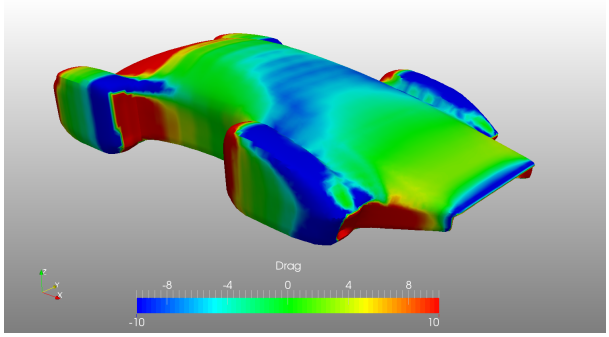


Figure 2.2: Car Drag

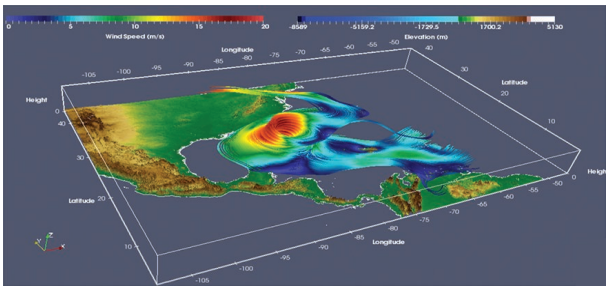
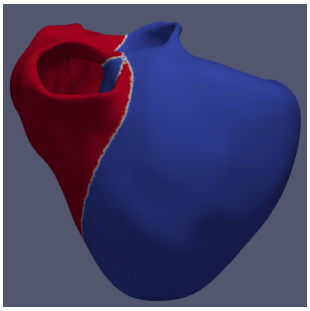


Figure 2.3: Wind Speed

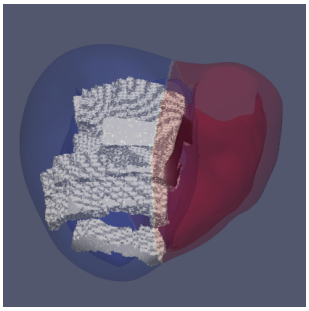
Car Drag visualization from *www.simscale.com/wp-content/uploads/2014/07/futuristic-car_drag.png*
Weather Speed visualization from *www.kitware.com/integration-of-paraview-catalyst-with-regional-earth-system-model/*

geometry and ischemic scar tissue structures. The reference mesh consists of three million cells and 600.000 vertices. As such, Paraviews capability in handling large data-sets effectively is required. Figure 2.4 displays our data in Paraview. We differentiate between the left ventricle (blue) and the right ventricle (red). The shape of ischemic scars is examined only within the left ventricle. It is obtained by processing MRI scans. Figure 2.4a displays a template heart. Figure 2.4b and figure 2.4c display two processed patient scars within the template heart. The hole displayed in figure 2.4c is not physical, instead it is due to the processing, which leaves out the basal region. Equally, the block-like form of the scars is due to the processing. Figure 2.4b display a disjoint scar tissue structure. This can also be attributed to the processing. With significant likelihood the scar tissue it connected. This is explained with more detail in chapter 7.

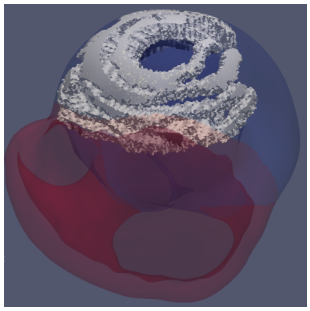
Figure 2.4: Our Data in Paraview



(a) Front View of a Heart



(b) View of a scar from the front



(c) View of a scar from the below

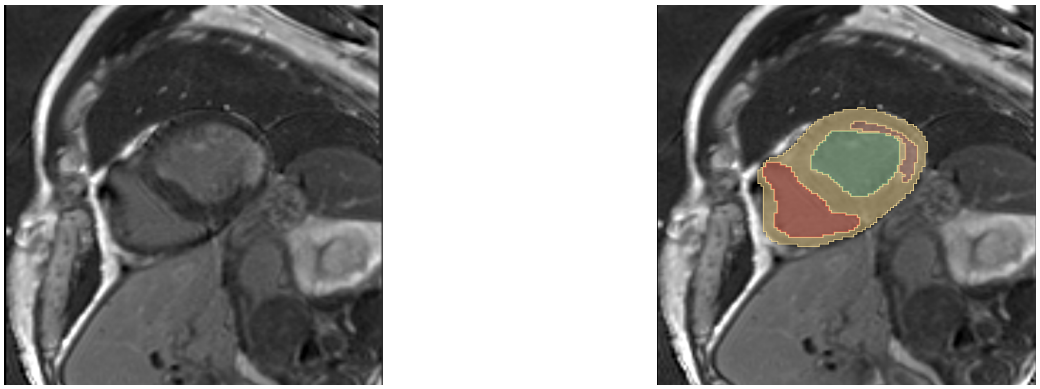
2.3 3D Slicer

3D Slicer is a comprehensive, open-source platform for medical image computing, widely used in clinical and research settings. It supports visualization, processing, segmentation and analysis of various imaging modalities, including MRI, CT, Ultrasound and other tomographic methods. Its flexibility allows users to handle multidimensional datasets with precision, making it a strong research tool for tasks like treatment planning, diagnostic assessments and quantitative analysis. 3D Slicer is a powerful due to its wide-ranging functionality and its strong user and developer community. Ongoing advancements aim to improve segmentation accuracy and enhance automation further reducing the risk of errors and expanding its use in various medical imaging fields.

Key features of 3D Slicer include contrast enhancement and thresholding, which aid in the accurate segmentation of anatomical structures. Segmentation, a critical step in analyzing medical images, involves isolating regions of interest, such as tumors or ischemic scars. While 3D Slicer provides both manual and semi-automated tools, manual segmentation can be prone to user-dependent variability.

Figure 2.5 displays the slice of an MRI scan and the manually obtained segmentation mask of the heart. The brown mask representing an ischemic scar (see fig. 2.5b) does align fully with the endocardium, the innermost layer of the myocardium. This misalignment contradicts the known physical properties of ischemic scars which typically grow from endocardium to epicardium. Such errors, common in manual segmentation, highlight the need for meticulous attention.

Figure 2.5: MRI scan and Segmentation



(a) MRI slice (b) Segmentation

yellow = myocardium, green = left chamber, red = right chamber, brown = ischemic scar

Chapter 3

Part 1: Basics of Cardiology

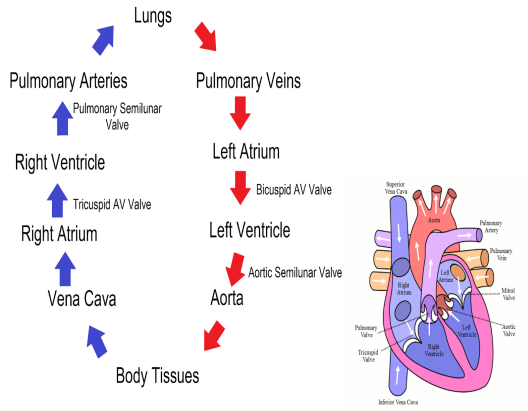
In the study of heart, clinicians often differentiate between the left and right ventricle/atrium [84]. The atrium is responsible for receiving blood, while the ventricles are responsible for the distribution to lungs and body tissue. Size-wise the ventricles are much bigger, the biggest part being the left ventricle. For the purposes of this work the atrium and right ventricle are neglected since ischemic scarring occurs predominantly in the left ventricle only [22].

Left and right ventricle (subsequently referred to as LV and RV) operate with equal quantities of blood and under equilibrium conditions but differ strongly in their size, shape and function. The LV operates against high pressure in the circulatory pathways throughout the body tissue whereas the resistance in the pulmonary circulation is much lower. Therefore, while both sides generate the same flow, the pulmonary flow is created against lower pressure. Accordingly, the left ventricle, presents the more powerful, thicker, myocardial muscle. It receives oxygen-rich blood from the left atrium and pumps it to the aorta. From here the blood is distributed to other parts of the body. The oxygen is used for cellular respiration and enables the synthesis of ATP. After this the de-oxygenated blood is directed to the right atrium via the Vena Cava. Furthermore, the aorta branches into the left and right coronary artery, blood vessels which transport oxygenated blood to the heart muscle itself. Blockage of the coronaries leads to insufficient alimentation of the heart and causes infarcts which manifest as scarring of the myocardium. The right ventricle receives de-oxygenated blood from the right atrium and pumps it to the pulmonary artery. Following this, the blood is re-oxygenated in the lungs and directed to the left atrium through the pulmonary veins. This closes the circulation. Due to the relation to coronary blockages, the spatial distribution of ischemic scar tissue correlates with the trajectory of the coronary arteries (displayed in figure 3.2).

3.1 Left Ventricular Segmentation and Coronary Territories

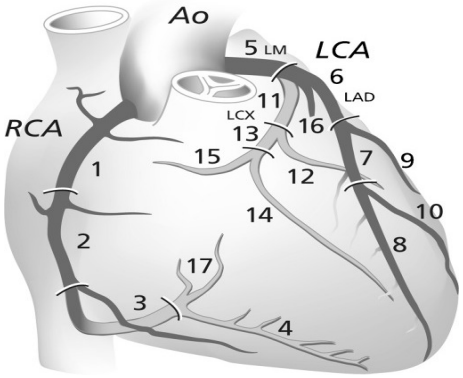
Coronaries can be associated in an approximate manner to 17 standardized heart regions. These regions are used to segment the heart according to the standard myocardial segmentation established by the American Heart Association (AHA), and are generally referred to as *AHA-regions*. It defines several regions in the left ventricle which are used by clinicians to communicate locations across different heart geometries. Figure 3.3 visualizes the AHA-regions of a heart. Figure 3.3 visualizes a left ventricle with colored AHA-regions in a heart-shaped and a disk-shaped representation. The apex is mapped to the disk-center, while the base is mapped to the disk-border. Figure 3.2 visualizes the right coronary artery (RCA) and left coronary artery (LCA) and their traversal through the AHA-regions. Blockages manifest as scarring in associated regions. Figure 3.5 displays such scars in respective regions. Notice that the scars are predominantly in a location corresponding to a specific coronary artery. This underlines the correlation between location and affected coronary/AHA-region.

Figure 3.1: Blood Circulation



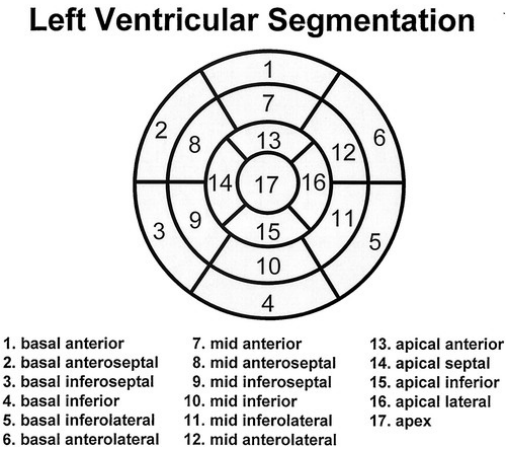
https://www.zenflowchart.com/blog/blood-circulation-in-heart-flowchart

Figure 3.2: Coronaries



https://www.heartfoundation.org.nz/images/heart-healthcare/coronary-artery-disease.png

Figure 3.3: Left Ventricular Segmentation and Coronary Territories



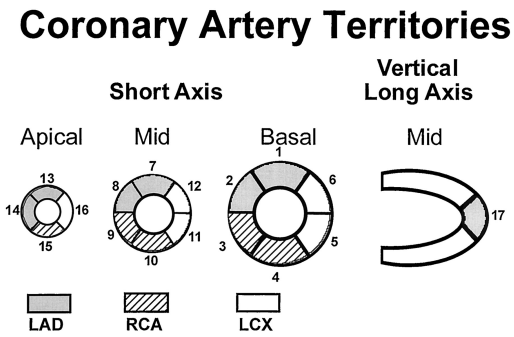


Figure 3.4: Heart with Colored AHA Regions

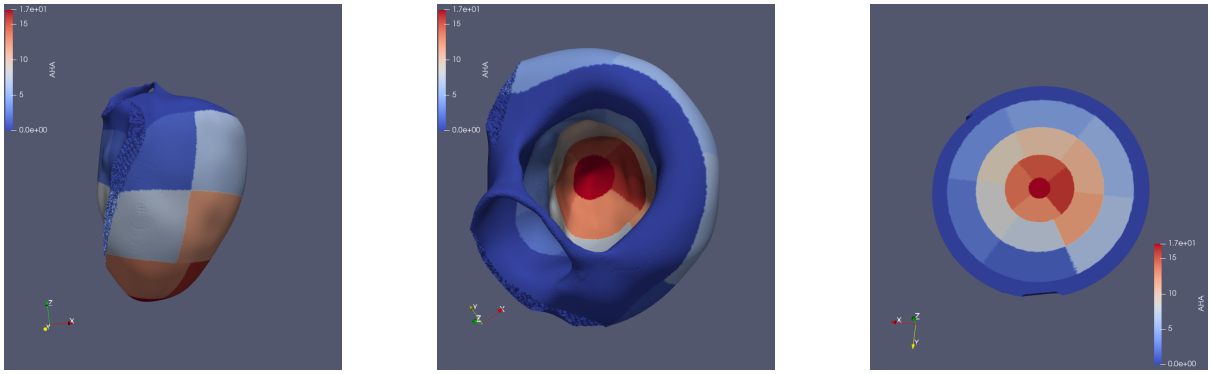
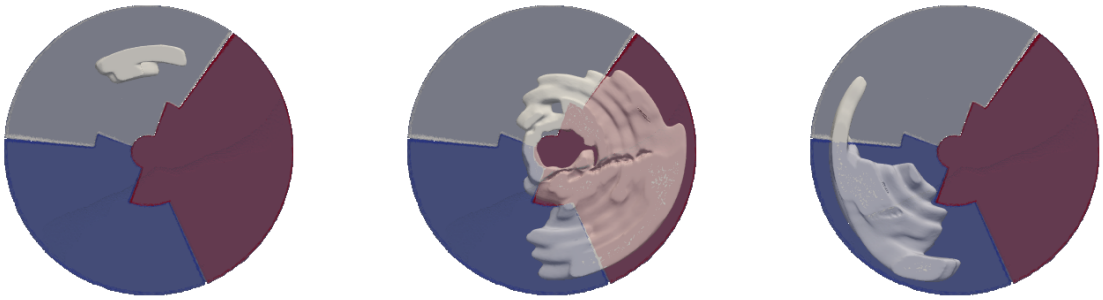


Figure 3.5: Scars in coronal regions (white: LCX, blue: RCA, red: LCA)



(a) Scars in LCX region (b) Scars in LAD region (c) Scars in RCA region

3.2 Ischemic Cardiomyopathy

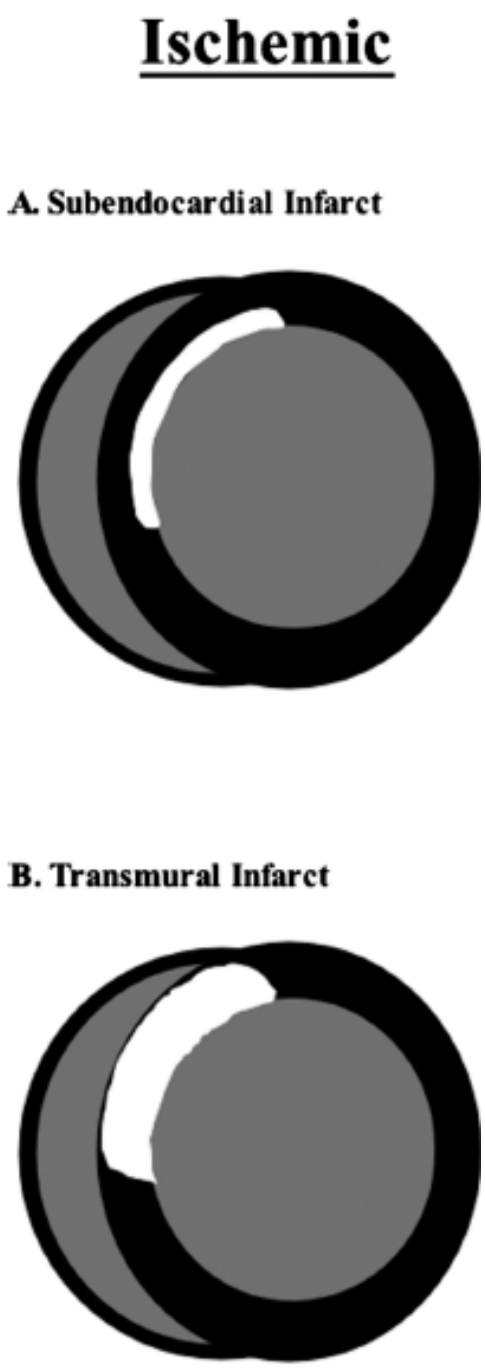


Figure 3.6: Transmurality of Infarction

Cardiomyopathy terms diseases which affect the myocardium leading to abnormal behavior of the heart, for example causing enlargement, thickness or stiffness. This effect can lead to symptoms of heart failure, inefficient blood flow and pathologies in the electrical activation pattern (e.g. arrhythmia). Literature differentiates between *non-ischemic* and *ischemic* heart disease (IHD). *Ischemic* cardiomyopathy is related to coronary artery disease which can manifest as scarring. All other cardio-myopathies are *non-ischemic*. IHD is by far more common and will be the focus of this work. Coronary artery disease reduces blood flow to the heart and the resulting insufficient nutrition causes cell death (necrosis) in affected areas. Insufficient nutrition causes the formation of scar tissue, which has different electro-mechanical properties than the surrounding healthy tissue. A result can be arrhythmia, weakened mechanical contraction and an overall decrease in heart functioning. In response to ischemic scarring, the heart adopts different techniques to mitigate the impaired functioning. The success of these adoptions, called *hyper-enhancements*, depends on the size of the scar. Common adaption strategies are hypertrophy, where the heart-muscle enlarges in an attempt to increase its mechanical pumping ability, and re-vascularization, where the coronaries can grow new branches to re-supply damaged cells with nutrients and prevent total necrosis. The size of the scar can be quantified by its *segmental occupancy* (see figure 3.7) and *transmurality* (see figure 3.6). One can differentiate between *transmural* and *subendocardial* infarctions. Different sizes are correlated to different outcomes of the hyper-enhancements and treatment options. Its quantification can be used as measure for the seriousness of IHD. Contrary to *chronic* ischemic scars, *acute* ischemic scars tissue can recover due to re-vascularization. The extend to which this is possible depends on the *transmurality* (see figure 3.6) of the impaired tissue. It is known, that scars grow from endocardium to epicardium (i.e. from the inner wall to the outer heart wall). A scar with less than 50% transmurality has good chances of recovering, above 70% it is very unlikely [25]. Therefore, it is common to examine the segmental intensity (see figure 3.7), together with the segmental transmurality of scar tissue in order to evaluate the likelihood of recovery, for example AHA-region 1 might be affected to a degree of 30 % with a transmurality of 75% .

3.3 Significance of Contribution

The disease IHD, also referred to as coronary artery disease (CAD) and atherosclerotic cardiovascular disease (ACD), is one of the leading causes of death worldwide. Approximately 1.72%

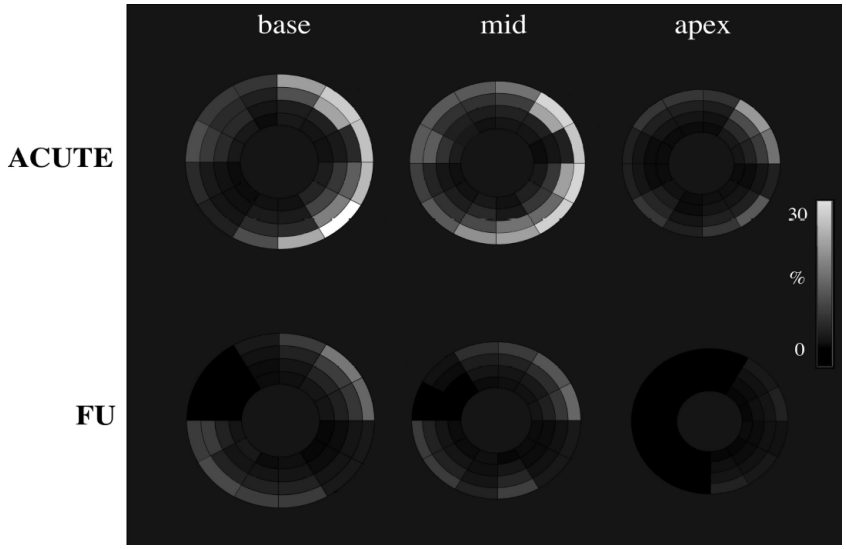
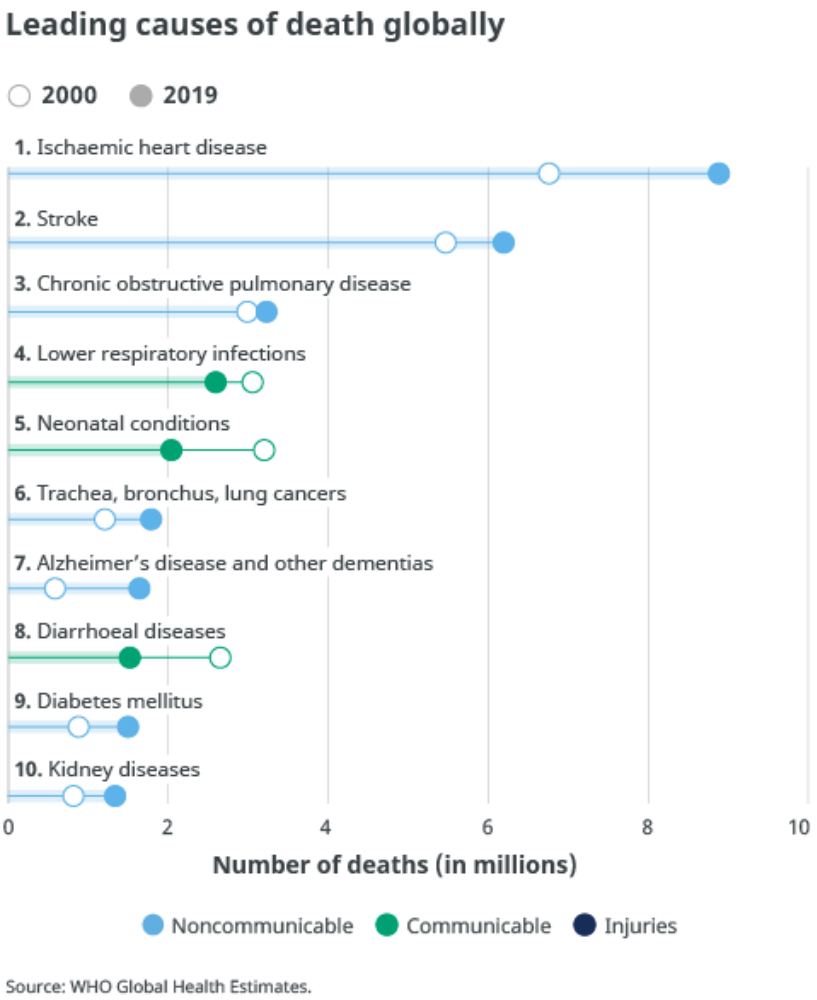


Figure 3.7: Segmental Intensity

of the world's population is affected, resulting in around nine million deaths annually [37]. According to the WHO, 16% of the annual deaths are caused by IHD. In figure 3.8 heart diseases represent the leading cause of death with a significant margin. Considering this it is safe to say that heart disease continues to be among the most important medical issues. Although the heart is one of the most studied organs, many challenges and mysteries remain. The advance of engineering methodologies and computing into the field of human biology can help to reach a deeper understanding of heart function, heart diseases, causes of death, and treatments. Never before has such a low risk/insight ratio been possible. We deliver a framework for the generation of ischemic scar models, which can be leveraged with simulations of the cardiovascular system. The generated scars can be filtered to represent desired segmental intensities and transmuralities. This enables the in-silico study of treatments and biophysical effects. Furthermore, privacy concerns are circumvented by the usage of artificial generation.

Figure 3.8: WHO Estimates



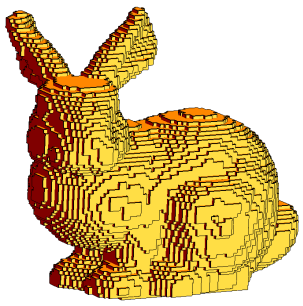
Chapter 4

Part 1: Shape representations

Training machine-learning algorithms requires a suitable data-representation. For example, molecules might be represented as graphs or strings. Depending on the case different representations might be used. For shapes several possible representations exist, each with different advantages and disadvantages. In the following, these representations are presented and specific characteristics, and main fields of application, are outlined. For our algorithm we will choose the signed distance function due to its memory-efficient representation which enables point-to-point correspondence.

4.1 Voxels

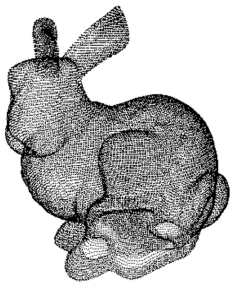
Voxels can be understood as an analog of pixels in 3D-space. They share many properties. For example, a representation of a 3D-shape by voxels corresponds to discretizing the space with a grid and labeling each element with the corresponding voxel-value. Like pixel-values, these can represent a colour, a texture, or any other label. Voxel representations are the natural data-format for medical images e.g. MRI scans. Here a whole 3D domain is voxelized. Fine discretization enables high accuracy. This representation in a fixed domain enables voxel-to-voxel correspondence between different images. However this voxel-to-voxel correspondence is not necessarily meaningful for the encoded object (imagine a rabbit in the upper corner and lower corner of the domain). Another disadvantage is that voxelization of a shape located within a domain is an inefficient way of representing 3D shapes since one also needs to voxelize the space around it. This leads to a high memory requirement. Furthermore, these representations are usually not differentiable.



(a) Voxel representation of a rabbit
https://cse.iitkgp.ac.in/ pb/research/3dpoly/bunny$_2$*.PNG*

4.2 Point Clouds

A common representation of 3D shapes are Point Clouds. Point Clouds are discrete sets of points, where each point is labeled with a position in 3D-space. Points can also be labeled with properties beyond location. This is among the simplest and most memory efficient representations. The point density can be balanced according to the needs. Adaptive density allows for good balancing between memory and detail. Furthermore, point clouds are easily manipulated (e.g. translated, rotated or scaled). However, they are not differentiable and do not provide simple methods for the establishment of point-to-point correspondence. Since no inherent surface information available this representation is unsuited for rendering and simulation. Computing a polygon representation from a point-cloud is computationally expensive. Point-clouds are the raw data-format obtained by scanning physical space e.g. lidar data.



(a) Point cloud representation of a rabbit
https://www.researchgate.net/publication/228945815_Reconstructing_Implicit_Surfaces_with_Boundaries

4.3 Meshes

Meshes, or polygons, correspond to the standard structure required for many algorithms in computer science and engineering. They consist of a set of points in 3D space connected in a graph structure. It is the most information heavy shape representation, meaning it is among the most useful shape representations but potentially with significant memory requirements. It contains surface information which is suitable for rendering and numerical solvers. Through a high (selective) density of polygons high accuracy is possible. On the downside it has high memory requirements since it contains point and connectivity information. Furthermore, point-to-point correspondence between polygons can be established only for equal topologies. A representation as polygons is usually not differentiable. Meshes are a standard in engineering, especially for numerical simulations and computer aided design. It is also used extensively in computer graphics, gaming, animations and engineering. As such it is studied to a high degree making shape manipulations relatively easy.

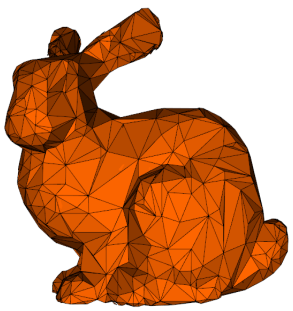


Figure 4.3: Mesh representation of a rabbit
https://i.stack.imgur.com/pFtBn.png

4.4 Implicit representations

Implicit shape representations do not attempt to represent a shape explicitly, e.g. as one would do with a field of occupancy, point clouds or meshes. Instead, the shape information is contained implicitly in some function over 3D space. A very common type of implicit shape representations is through the use of (un-)signed distance functions. Here, the 3D-shape is represented by the level of the distance function. For example, values of zero might correspond to the surface, negative values to the inner volume and positive values to the outer volume. Each value represents the nearest distance to the boundary. In principle it can also correspond to some other measure. Implicit representations can be very memory efficient through storing a function instead of a volume. If this function is continuous it can be differentiable. Surface information is contained as a level-set and the extraction of mesh-representations is possible with efficient algorithms such as marching cubes. Being a function within a domain, point-to-point correspondence is available. Much like for voxels, this correspondence is not necessarily meaningful for the encoded shape. Shape manipulation can be more difficult than for other representations. Furthermore, post-processing is required to obtain explicit representations. These representations are quite popular in computer graphics.

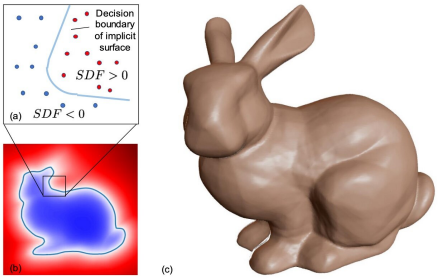


Figure 4.4: implicit representation of a rabbit
https://arxiv.org/pdf/1901.05103.pdf

4.5 Statistical Shape Models

Statistical Shape Models (SSMs) are popular models in computational anatomy. Shapes are represented as vectors of landmarks. Usually, point-to-point correspondence between the landmarks of different shapes is available or can be established (e.g. between tips of a thumbs). The shape distribution can then defined by employing linear methods, such as PCA or nonlinear methods such as Gaussian Processes or variational autoencoders [21]. This enables the sampling of new shapes, detailed analysis and model fitting methodologies. This representation is ubiquitous and well studied, making shape manipulation relatively easy, for example through the usage of provided libraries [47]. PCA and Gaussian processes are among the most easily interpreted machine learning models. Furthermore, they can be very information efficient if the required landmarks are sparse. On the down-side, point-to-point correspondence must be available, restricting the method to equal topologies. The main application domain is in medical engineering.

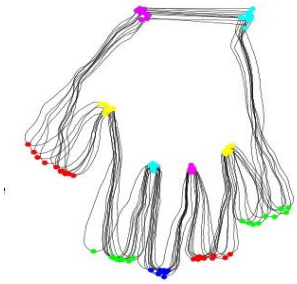


Figure 4.5: statistical shape model of a hand
https://www.slideserve.com/albert/information-theory-and-automatic-model-building

Chapter 5

Part 1: Machine Learning basics

All of machine learning aims to find a set of model parameter which minimize/maximize a loss/objective. Usually, the term objective refers to a quantity that is maximized while the loss is minimized. The objective is designed so that the parameterized model captures desired data properties. Usually it corresponds to a likelihood-measure describing how well a parametrization describes available data and/or a loss-function which measures deviations from desirable properties. The optimal parameters are found at the extremum of this function. Since the set of all possible parametrization is infinite, assumptions must be made to restrict the search-space to a feasible size. One such assumption might be that the data comes from a multi-variate Gaussian distribution, that it follows a linear line (or is separable by such) or that any optimal model is smooth. The class of considered models is called the hypothesis class. It might for example consist of polynomials of 2nd order, Gaussian processes with exponential kernel or three layer deep, eight neurons wide networks with sigmoid activation functions. During algorithm design it is common to distinguish between a Fisherian and a Bayesian approach. The first bases itself on the long-term frequency of events and estimates deterministic parameters, while the latter infers a posterior distribution of probabilistic parameters. In Bayesian models, a known prior distribution over the parameters is updated to form a posterior distribution. Fisherian models use deterministic parameters, imposing no distribution. It can be shown that both approaches are in some sense equivalent and relate to each other via available prior knowledge on the model parameters. In both cases the parameters are updated to (explicitly) optimize an objective and (implicitly) model the available data. If this data is labeled, supervised approaches can be used. Sometimes, the data is unlabeled so that unsupervised learning methods are required. Some algorithms combine both. We provide an introduction into the basics of machine learning. Using this basis we then examine principal component analysis and variational auto-encoders, elucidating their similarities and differences. These algorithms can be used for variational inference, an approach which can be applied for generative purposes by considering an unobserved generating variable.

5.1 supervised/unsupervised/semi-supervised ML

An important distinction that can be made in machine learning is between supervised learning, unsupervised learning and semi-supervised learning. These terms correspond to learning from a labeled dataset $D = \{(x_i, y_i)\}_{i=0}^{N}$, extracting patterns from an unlabeled dataset $D = \{(x_i)\}_{i=0}^{N}$ and combining both methodologies respectively. Supervised learning is more common and better studied. Contrary to unsupervised learning it can be applied directly to make predictions by modeling the conditional probability $p(y|x)$. Furthermore it comes with rigorous mathematical guarantees. On the other hand, unsupervised learning requires no information on the labels y and can not be used (directly) for prediction tasks. Examples of unsupervised learning are clustering and dimensionality reduction. Mathematically rigorous definitions of training-error

and generalization-error are available only for supervised training settings. However, extension of these concepts to unsupervised learning may be possible in some cases. Semi-supervised learning combines both approaches. For example by reducing dimensionality and applying supervised learning on the low dimensional data representations.

5.1.1 supervised machine learning

In supervised machine learning, a data-set

$$D = \{(x_1, y_1), ..., (x_n, y_n)\}, \tag{5.1}$$

with $x \in \mathcal{R}^d$ and $y \in \mathcal{R}^l$, is available. Data samples are drawn from some unknown distribution $(x, y) \sim p(X, Y)$. Goal is finding a function $h_\theta(x)$ so that

$$y = h_\theta(x) \text{ for } (x, y) \sim p(X, Y), \tag{5.2}$$

where θ corresponds to the model parameters. This requires the definition of a hypothesis class $\mathcal{H}$ that contains all the functions of the hypothesis examined $h_\theta \in \mathcal{H}$. Ideally, the hypothesis class $\mathcal{H}$ would contain all possible functions. In practice, assumptions must be made to restrict the search-space to a feasible size.

Given $\mathcal{H}$, one aims to find the optimal model $h_\theta \in \mathcal{H}$. This involves finding the function that minimizes (or maximizes) some performance measure in the training data $D_{train} \subseteq D$. However, the actual goal of machine learning is generalization to unseen data and not an optimal description of the training data. The unseen data can be represented by the test data $D_{\text{test}} \subseteq (D \setminus \{D_{\text{train}}\})$. If D_{train} does not have the same distribution as D_{test}, one speaks of a distribution shift. Performance measures, such as training and test error, are evaluated in the training data D_{train} and the test data D_{test}.

The training error can be defined as

$$L_{training} = \frac{1}{|D_{train}|} \sum_{(x_i, y_i) \in D_{train}} (y_i - h_\theta(x_i))^2. \tag{5.3}$$

The test error can be defined as

$$\epsilon_{test} = \frac{1}{|D_{test}|} \sum_{(x_i, y_i) \in D_{test}} (y_i - h_\theta(x_i))^2. \tag{5.4}$$

The true generalization error ϵ is unknown. However, by the *weak law of large numbers*, the empirically estimated ϵ_{test} converges to ϵ as $|D_{test}| \to \infty$. If $D_{test} \sim p(X, Y)$ is drawn from the same distribution as the data D, then ϵ_{test} is an unbiased estimator of the generalization error, i.e.

$$\mathbb{E}(\epsilon_{test}) = \mathbb{E}_{(x,y)\sim p(X,Y)} \left[y - h_\theta(x) \right] = \epsilon. \tag{5.5}$$

The final objective of supervised machine learning is to find

$$\hat{h}_{\theta_{opt}} = \underset{\theta}{\text{argmin}}\ \mathbb{E}_{(x,y)\sim p(X,Y)} \left[y - h_\theta(x) \right] \approx \underset{\theta}{\text{argmin}} \sum_{(x_i, y_i) \sim p(X,Y)} \left[y_i - h_\theta(x_i) \right], \tag{5.6}$$

where the last step is due to the famous monte-carlo approximation of definite integrals.

Consider the Chernoff bound

$$\Pr_{D\sim p(X,Y)} \left[\left| \frac{1}{n} \sum_{(x,y)\in D} (y - \hat{h}_{opt}(x))^2 - \mathbb{E}_{(x,y)\sim p(X,Y)} \left[(y - \hat{h}_{opt}(x))^2 \right] \right| \geq \epsilon \right] \leq e^{-2n\epsilon^2} \tag{5.7}$$

For simplicity let us write

$$\Pr\left[|err(h) - e\hat{r}r(h)| \geq \epsilon\right] \leq e^{-2n\epsilon^2} \tag{5.8}$$

By union bounds

$$\Pr\left[\exists h \in \mathcal{H} : |err(h) - e\hat{r}r(h)| \geq \epsilon\right] \leq |\mathcal{H}|\, e^{-2n\epsilon^2}. \tag{5.9}$$

Consider an arbitrary value δ so that

$$|\mathcal{H}|e^{-2n\epsilon^2} \leq \delta \tag{5.10}$$

which holds if

$$n \geq \frac{\ln 2|\mathcal{H}| + \frac{1}{\delta}}{2\epsilon^2} \tag{5.11}$$

so that, by conversion of the union bound,

$$h \in \mathcal{H} : \Pr\left[|err(h) - e\hat{r}r(h)| \leq \epsilon\right] \geq 1 - \delta \tag{5.12}$$

where

$$\epsilon = \sqrt{\frac{\ln 2\,|\mathcal{H}| + \frac{1}{\delta}}{2n}}. \tag{5.13}$$

This provides some solid mathematical guarantees on supervised learning. If the training data-size n is large enough and it comes from the same distribution $p(x, y)$ as the test-data, then the generalization error is bounded and tends towards zero with probability 1.

Two common algorithms for finding optimal parameters in a supervised machine learning setting are *maximum likelihood estimation* (Fisherian) and *maximum a-posterior* (Bayesian). It instructive to approach deep learning algorithms with neural nets by considering these two types of machine learning algorithms.

5.1.2 unsupervised machine learning and information theory

The setting of unsupervised machine learning deals with discovering patterns in data. In essence, there is some data

$$D = \{x_1, , ..., x_n\}$$

without any available labels. Any algorithm which extracts patterns from label-less data belongs to the category of unsupervised learning. These algorithms can not be used directly for predictions and classifications. Instead, they aim to cluster the data points into a specified amount of groups (gaussian mixture models, K-nearest-neighbours), reduce dimensionality (principal component analysis, gaussian process latent variable models, variational auto-encoders) and/or discover other meaningful patterns (apriori, frequent pattern, eclat).

Unsupervised learning can be seen as an information representation problem. It involves finding the best representation for the relevant property. The available data are often noisy and contain irrelevant information. The extraction of all relevant information (and/or removal of irrelevant information) for the given task corresponds to finding an optimal representation. A common term for this is feature extraction. This description, via features, can be used to make predictions and/or compress data. It is often connected to dimensional augmentations. Kernel-methods might construct additional dimensions to empower linear methods, or apply dimensionality reductions to remove noise. Finding suitable representations is an important task of machine learning and often half the solution. A good representation contains all the relevant information, but ideally not more, in a computationally accessible way.

Contrary to supervised learning, unsupervised learning can not be defined in any simple way making evaluation difficult. Many unsupervised learning algorithms optimize a proxy objective that does not represent the actual goal. As an example, consider the K-nearest-neighbors algorithm which groups data into k clusters. The actual goal is to discover *meaningful* patterns which reveal something about the data, not clustering into k groups. The clustering discovered is not necessarily useful. Meaningful clustering depends on the right number of groups, but it is impossible to give an axiomatically consistent definition of the 'right' clustering [38].

For the development of some intuition, the problem of unsupervised learning can be related to information theory. Both are about finding patterns in data and representing information in terms of these patterns. Consider the problem of compressing data $X = [X_1, X_2]$ with some compressor $C(X)$, then

$$|C(X)| \leq |C(X_1)| + |C(X_2)|.$$

In essence, the compression of $X = (X_1, X_2)$ is smaller than the compression of X_1 and X_2 summed. The gap between the two corresponds to the mutual information, information on X_2 which can be described by X_1 and vice versa. Consider an optimal (purely theoretical and un-computable) compressor $K(X)$ of minimal length and maximal compression, then

$$|K(X)| \leq |C(X)| + |K(C)|,$$

where $K(C)$ corresponds to the theoretical information magnitude required to define the compressor K itself. The unsupervised learning problem can be viewed as the problem of finding this optimal compressor. Some program space is searched for an optimal representation of the information. Since the space of all possible programs is prohibitively large we have to restrict the search space through our model parametrization. Common models perform clustering and/or dimensionality reduction. Both can be viewed as compressing information into a number of groups.

In information theory important concepts related to information compression are the entropy and the mutual information. The shannon entropy is a measure of the uncertainty or unpredictability of a random variable. For a discrete random variable X with possible outcomes $\{x_1, x_2, \ldots, x_n\}$ and a corresponding probability distribution $P(X = x_i) = p_i$, the Shannon entropy $H(X)$ is defined as:

$$H(p) = -\mathbb{E}\left[p(x) \log p(x)\right] = \sum_{i=1}^{n} p_i \log p_i \tag{5.14}$$

By the source coding theorem, an i.i.d random variable X with density $p(x)$, which occurs N times, can be compressed into $N * H(X)$ bits, where the log is computed with base two. As example, consider long sequences of k symbols, each representing a random variable with a distribution defined by the frequency of occurrence. Then the amount of bits required to encode this sequence is $\sum_k N_k H(X_k)$.

Related to the shannon entropy is the relative entropy. It is also called Kullback-Leibler divergence and an ubiquitous quantity in machine learning. It is defined as

$$D_{KL}(p, q) = -\mathbb{E}\left[p(x) \log \frac{p(x)}{q(x)}\right]. \tag{5.15}$$

The Kullback-leibler divergence has a geometrical interpretation. Although it is asymmetric and thus not a distance, it is intimately related to the Fisher Information metric and geodesics on an information manifold. Informally speaking, it is a quadratic form approximation of the

geodesic distance between two distributions. Many unsupervised learning algorithms minimize the Kullback-Leibler (D_{KL}) divergence with the goal of approximating $p(x)$ with a parametrized $q(x)$.

The cross-entropy unifies the shannon entropy and D_{KL} divergence and allows for another interpretation. It is defined as

$$H(p,q) = -\mathbb{E}\left[p(x)\log q(x)\right] = \sum_i p_i \log q_i \tag{5.16}$$

and measures the information required to describe $q(x)$ with $p(x)$. It can be interpreted as the log-likelihood of the data-distribution $q(x)$ when drawing samples from the model-distribution $p(x)$. The relation with relative entropy and Shannon entropy is given via

$$H(p,q) = H(p) + D_{KL}(p,q) = H(q) + D_{KL}(q,p). \tag{5.17}$$

Informally speaking, the amount of information required to describe p with q is the amount of information required to describe p plus the relative entropy $D_{KL}(p,q)$ i.e. the asymmetric distance (divergence) between p and q.

These information theoretical quantities enable the description of data-distributions $p(x)$ via parameterized model distributions $q(x)$. Through optimization of the Kullback-Leibler divergence a model distribution can be fitted to the data. If $D_{KL}(p,q) = 0 = D_{KL}(q,p)$, then both distributions have equal Shannon-entropy $H(p) = H(q)$ and can be used to encode each other ($H(p,q) = H(q,q) = H(q) = H(p)$). Many algorithms use the Kullback-Leibler divergences as objective function to approximate distributions.

Another important quantity is Mutual Information (MI), which quantifies the amount of information obtained about one random variable through another random variable. It measures the reduction in uncertainty (used here synonymous to entropy) about one variable when knowing the value of the other. For two random variables X and Y, the mutual information $I(X;Y)$ is defined as:

$$I(X;Y) = H(X) - H(X|Y). \tag{5.18}$$

The conditional entropy $H(X|Y)$ can be expressed via

$$H(X|Y) = H(X,Y) - H(Y) \tag{5.19}$$

Alternatively, mutual information can be expressed as

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} P(x,y)\log\left(\frac{P(x,y)}{P(x)P(y)}\right) = D_{KL}(P(x,y)||P(x)P(y)). \tag{5.20}$$

Thus the mutual information measures how independent two random variables are from each other. If

$$P(x)P(y) = P(x,y)$$

both variables are perfectly independent and

$$I(X;Y) = D_{KL}(P(x,y)||P(x)P(y)) = 0.$$

Often in machine learning (for example in gaussian process latent variable models), a complicated data-distribution $p(x,y,z)$ is parameterized as a factorization $p(x)p(y)p(z)$ (implicitly assuming independence). The optimal parameters are determined through minimization of the D_{KL} divergence. Mutual Information is also used to measure statistical similarities between data X and Y, for example between images in image registration [48]. Many works approach unsupervised learning from an information theoretic perspective [61], [33], [8], [27].

5.1.3 semi-supervised machine learning

Semi-supervised machine learning combines supervised and unsupervised methods, for example by reducing the dimensionality or grouping the data before training a classification or regression algorithm. Examples may be clustering unlabeled data to a subset of labeled data, performing dimensionality reductions prior to a supervised learning setup and/or assigning high-confidence model predictions of unlabeled data-points as labels. The variational autoencoder [15] is a semi-supervised learning algorithm. One obtains a latent distribution through nonlinear dimensionality reduction ($z = f(x)$) and unsupervised learning, while the reconstruction ($\hat{x} = g(z)$) is optimized through supervised learning. The D_{KL} divergence is used to align the chosen parametrization of the (unknown) latent distribution $q(z)$ with some assumed prior-distribution $p(z)$. On the other hand, the data-sample x is used as reference label for the reconstruction $\hat{x} = g(z)$ in a supervised learning manner. The objective function sums two respective loss-terms.

5.1.4 Limits: the Bias-Variance Trade-off

The Bias-Variance trade-off is the classical example for explaining an intricate balance between training error and generalization capabilities. Assume a labeled dataset

$$D = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n).\}$$

Let $h_\theta(x)$ be a parametrization of a model, let $y(x)$ be the outcome observations corresponding to the data-sample $x \in D$, i.e. $y_1 = y(x_1)$. Let $\hat{y}(x) = \int_y p(y|x)dy$ be the expected label given some feature vector x. For noiseless data $y(x) = \hat{y}(x)$. Let

$$\hat{h}_\theta(x) = \mathbb{E}[h_\theta(x), D] = \int_D \int_x h_\theta(x) p(x|D_{train}) dx dD_{train}$$

denote the expected model given the data D, with $D_{train} \subset D$. Here, $p(x|D_{train})$ denotes the probability of drawing x given the data D_{train}, while $p(D_{train})$ denotes the probability of drawing the training-data D_{train} from D. Denote by $\mathbb{E}\left[(h_\theta(x) - y)^2\right]$ the expectation of the error given a sample $x \in D$. Let $\hat{y}$ be the true observation and y its measurement. Let $\hat{h}_\theta(x)$

be the true model and $h_\theta(x)$ the parametrized model. Then the generalization error E becomes

$$
\begin{aligned}
\mathrm{E} &= \mathbb{E}_{x,y,D}\left[(h_\theta(x) - y)^2\right] \\
&= \mathbb{E}_{x,y,D}\left[\left(h_\theta(x) - \hat{h}_\theta(x) + \hat{h}_\theta(x) - y\right)^2\right] \\
&= \mathbb{E}_{x,y,D}\left[\left(h_\theta(x) - \hat{h}_\theta(x)\right)^2\right] + \mathbb{E}_{x,y,D}\left[\left(\hat{h}_\theta(x) - y\right)^2\right] + 2\mathbb{E}_{x,y,D}\left[\left(h_\theta(x) - \hat{h}_\theta(x)\right)\left(\hat{h}_\theta(x) - y\right)\right] \\
&= \mathbb{E}_{x,y,D}\left[\left(h_\theta(x)) - \hat{h}_\theta(x)\right)^2\right] + \mathbb{E}_{x,y,D}\left[\left(\hat{h}_\theta(x) - y\right)^2\right] + 2\mathbb{E}_{x,y}\left[\left(\hat{h}_\theta(x) - \mathbb{E}_D\left[h_\theta(x)\right]\right)\left(\hat{h}_\theta(x) - y\right)\right] \\
&= \mathbb{E}_{x,y,D}\left[\left(h_\theta(x)) - \hat{h}_\theta(x)\right)^2\right] + \mathbb{E}_{x,y,D}\left[\left(\hat{h}_\theta(x) - y\right)^2\right] + \underbrace{2\mathbb{E}_{x,y}\left[\left(\hat{h}_\theta(x) - \hat{h}_\theta(x)\right)\left(\hat{h}_\theta(x) - y\right)\right]}_{0} \\
&= \mathbb{E}_{x,y,D}\left[\left(h_\theta(x)) - \hat{h}_\theta(x)\right)^2\right] + \mathbb{E}_{x,y,D}\left[\left(\hat{h}_\theta(x) - y\right)^2\right] \\
&= \mathbb{E}_{x,y,D}\left[\left(h_\theta(x) - \hat{h}_\theta(x)\right)^2\right] + \mathbb{E}_{x,y,D}\left[\left(\hat{h}_\theta(x) - \hat{y} + \hat{y} - y\right)^2\right] \\
&= \mathbb{E}_{x,y,D}\left[\left(h_\theta(x) - \hat{h}_\theta(x)\right)^2\right] + \mathbb{E}_{x,y,D}\left[(\hat{y} - y)^2\right] + \mathbb{E}_{x,y,D}\left[\left(\hat{h}_\theta(x) - \hat{y}\right)^2\right] + \underbrace{2\mathbb{E}_{x,y,D}\left[\left(\hat{h}_\theta(x) - \hat{y}\right)(\hat{y} - y)\right]}_{0} \\
&= \underbrace{\mathbb{E}_{x,y,D}\left[(h_\theta(x)) - \hat{h}_\theta(x))^2\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{x,y,D}\left[\left(\hat{h}_\theta(x) - \hat{y}\right)^2\right]}_{\text{Bias}^2} + \underbrace{\mathbb{E}_{x,y,D}\left[(\hat{y} - y)^2\right]}_{\text{Irreducible Error}}
\end{aligned}
\tag{5.21}
$$

This decomposition of the generalization error into three terms enables a differentiated analysis. The variance captures how much the model would change if it were trained on a different training set. It reflects the degree to which the model is overspecialized. The bias captures the degree to which our model is misaligned from the true data. Plotting both terms over model capacity (see Figure 5.1) highlights that while complex models can achieve better training errors, the simpler models generalizes best (see Figure 5.2). This is in accordance with an information theoretical point of view and Occam's razor [16]. "Entities must not be multiplied beyond necessity".

Figure 5.1: The Bias Variance Tradeoff

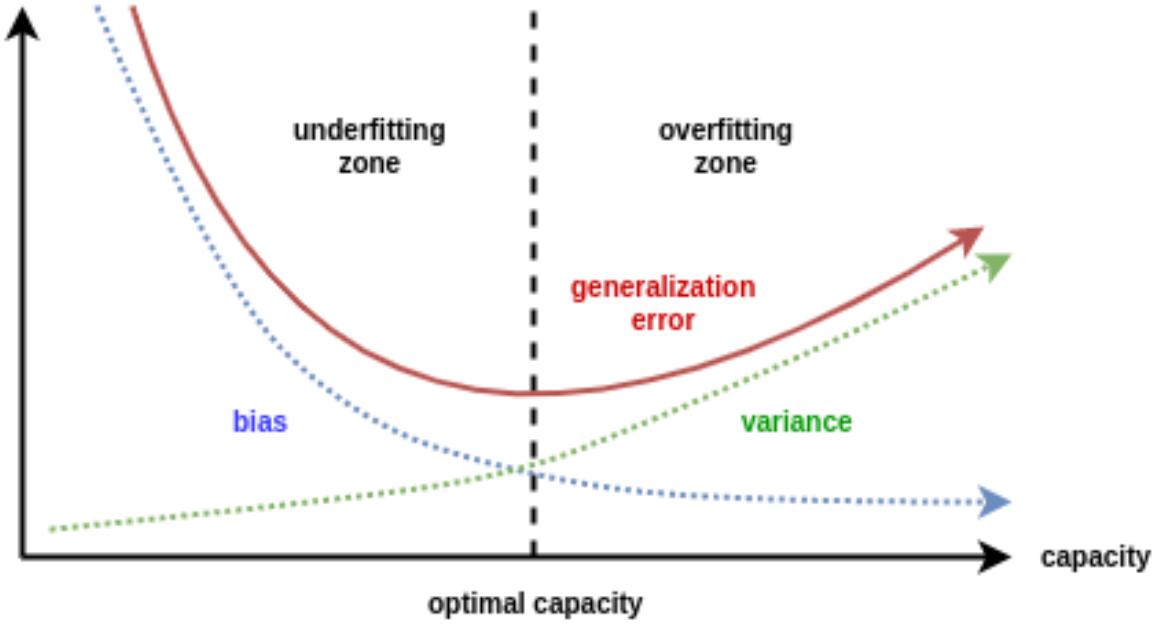
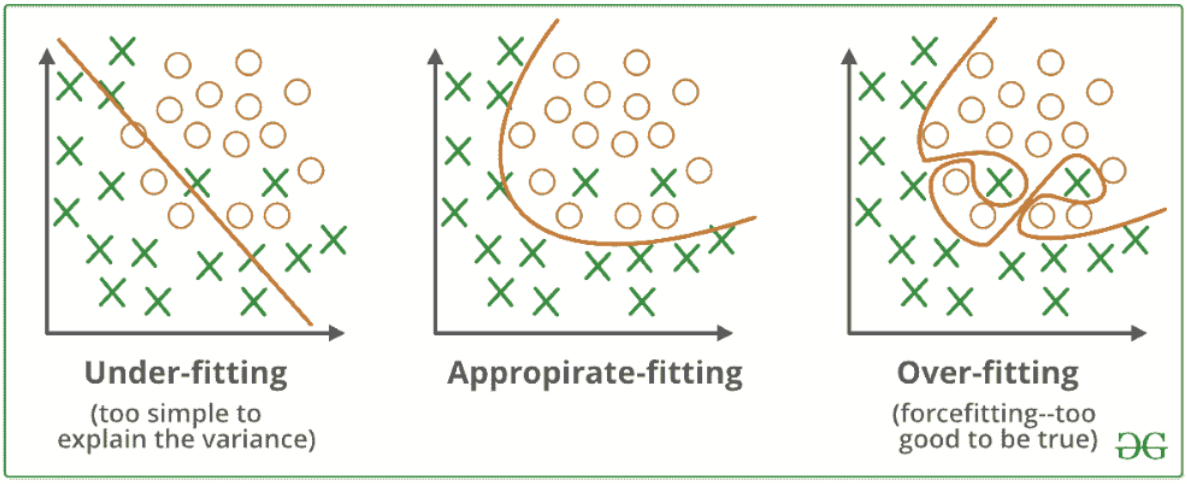


Figure 5.2: Overfitting and Underfitting



https://www.geeksforgeeks.org/regularization-in-machine-learning/

5.2 MLE - Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a Fisherian methodology for fitting model parameters to a data-distribution [57]. Consider a model, parameterized by a determinstic θ, so that,

$$x = h(\theta) + \epsilon$$

with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and θ deterministic. Then $x \sim \mathcal{N}(h(\theta), \sigma^2)$.

The probability density function of this Gaussian is defined as

$$p(x, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(x - h(\theta))^2}{2\sigma^2}. \tag{5.22}$$

Given data-samples x_i with $i = 1, ..., N$, Maximum Likelihood Estimation amounts to finding the model which results in the maximum joint probability. For $N = \infty$, the strong law of large numbers states

$$p(x|\theta) = \prod_{i=1}^{N} p(x_i, \theta). \tag{5.23}$$

Maximizing this quantity is an optimization problem. The product can be simplified into a more computable form by using the log-likelihood. Since the logarithm is strictly monotonous, extrema are unchanged.

$$\begin{aligned}
\theta_{opt}(x) &= \underset{\theta}{\text{argmax}} \prod_{i=1}^{N} p(x_i, \theta) \\
&= \underset{\theta}{\text{argmax}} \sum_{i=1}^{N} \ln(p(x_i, \theta)) \\
&= \underset{\theta}{\text{argmin}} - \sum_{i=1}^{N} \ln(p(x_i, \theta)) \\
&= \underset{\theta}{\text{argmin}} - \sum_{i=1}^{N} \left[\ln(\frac{1}{\sqrt{(2\pi\sigma^2)}}) + \frac{-(x_i - h(\theta))^2}{2\sigma^2} \right]. \\
&= \underset{\theta}{\text{argmin}} - \sum_{i=1}^{N} \frac{-(x_i - h(\theta))^2}{2\sigma^2} \\
&= \underset{\theta}{\text{argmin}} \sum_{i=1}^{N} (x_i - h(\theta))^2 * \text{const.}
\end{aligned}$$

Often const. $= \frac{1}{N}$. Note, that

$$\sum_{=1}^{N} (x - h(\theta))^2 * \frac{1}{N}$$

is often referred to as *mean squared error*. Minimizing the mean squared error returns optimal model parameters if the data distribution is gaussian. It can be proven that the estimator $h(\theta_{opt})$, is an unbiased minimum variance estimator (UMVE).

Summarizing, Maximum Likelihood Estimation amounts to computing

$$\theta_{opt}(x) = \underset{\theta}{\text{argmin}} \sum_{i=1}^{N} (x_i - h(\theta))^2 * \text{const.} \tag{5.24}$$

5.3 MAP - Maximum A-Posterior

Maximum A-Posterior (MAP) is a Bayesian methodology for fitting probabilistic model parameters to a data-distribution [12]. Consider a model, parameterized by θ, so that,

$$y = h(\theta) + \epsilon$$

with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $\theta \sim \mathcal{N}(0, \tau^2)$. Contrary to MLE we assume probabilistic model parameters.

By definition, the density function of this gaussian can be written as

$$p(\theta) = \frac{1}{\sqrt{2\pi\tau^2}} \exp \frac{\theta^2}{2\tau^2}.$$

This corresponds to the prior. In MAP we aim to find the posterior parameter distribution given the data. Consider the dataset x_i and note that

$$p(x, \theta) = p(\theta|x_1, .., x_N)p(x_1, .., x_N) = p(x_1, .., x_N|\theta)p(\theta) = \prod_{i=1}^{N}[p(x_i|\theta)]p(\theta),$$

where $p(x_1, .., x_N) = \text{const.}$ is determined by the data. Therefore

$$\begin{aligned}
\theta_{opt} &= \underset{\theta}{\operatorname{argmax}}\, p(\theta|x_1, .., x_N) \\
&= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N}[p(x_i|\theta)]p(\theta) \\
&= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N}[\ln(p(x_i|\theta)] - \ln(p(\theta)) \\
&= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N}[\frac{-(x_i - h(\theta))^2}{2\sigma^2}] - \frac{\theta^2}{2\tau^2} \\
&= \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N}[(x_i - h(\theta))^2] - \lambda\theta^2,
\end{aligned}$$

with $\lambda = \frac{\sigma^2}{N\tau^2}$. Note that the first term is equal to the mean squared error, as obtained during maximum likelihood estimation. The second term can be seen as a regularizer which constrains the model-parameters θ to be as small as possible. This is in accordance with Occam's razor. The loss obtained through maximum a-posterior coincides with a regularized maximum likelihood approach.

5.3.1 Connections between MLE and MAP

Assume a probability-distribution $p(x, \theta)$, as we have done in the maximum likelihood estimation (MLE) and maximum a-posterior (MAP) sections. Then, by the rules of probability

$$p(x, \theta) = p(x|\theta)p(\theta). \tag{5.25}$$

Since, in MLE we consider a deterministic parameter θ, the term $p(\theta)$ is irrelevant for optimization and the MLE objective

$$\theta_{opt} = \underset{\theta}{\operatorname{argmax}}\, p(x|\theta)$$

is equivalent to

$$\theta_{opt} = \underset{\theta}{\operatorname{argmax}}\, p(x, \theta).$$

The unregularized MLE assumption is that $p(\theta)$ is uniform, i.e. all θ are equally likely. No prior information is available.

For MAP consider

$$p(x, \theta) = p(\theta|x)p(x). \tag{5.26}$$

Since $p(x)$ does not depend on θ, the MAP objective

$$\theta_{opt} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|x)$$

is also equivalent to

$$\theta_{opt} = \underset{\theta}{\operatorname{argmax}}\, p(x, \theta).$$

However, this time the parameters θ are also drawn from a non-uniform prior distribution $p(\theta)$, often a gaussian distribution.

The loss-function resulting via MAP is equivalent to the loss-function resulting via MLE with an additional term. This methodology which introduces additional penalty-terms to the MLE-loss is called model regularization. For example, one might penalize the square of the parameter-magnitude so that the algorithm explores low parameter spaces more. This can help to avoid over-fitting by balancing small training errors with complex (and strongly biased) models. Regularization incorporates prior assumptions to derive terms which penalize deviations. Common methodologies are Ridge regularization and Lasso regularization. maximum likelihood estimation with Ridge regularization is equivalent to maximum a-posterior estimation with a gaussian prior $p(\theta)$.

Summarizing, MLE and MAP are different, although similar methodologies for maximizing the likelihood of $p(x, \theta)$ given some observed data x. MAP requires some prior knowledge on the model-parameters. MLE is equivalent to MAP if the prior knowledge is encoded in the objective through an additional regularization term.

5.4 PCA - Principal Component Analysis

Principal component analysis (PCA) is a popular linear algorithm for systems of the form

$$x = f(z) = Wz \tag{5.27}$$

or, in the probabilistic PCA setting

$$x = f(z) + \epsilon = Wz + \epsilon \tag{5.28}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ and $z \sim \mathcal{N}(0, I)$, so that

$$p(x|z) = \mathcal{N}(Wz, WW^T) \text{ and } p(x) = \mathcal{N}(Wz, WW^T)$$

since

$$\Sigma = \text{Covar}(x) = \mathbb{E}(Wz)\mathbb{E}(Wz)^T = W\mathbb{E}(z)\mathbb{E}(z)^T W^T = WW^T.$$

in the former case and

$$p(x|z) = \mathcal{N}(Wz, WW^T) \text{ and } p(x) = \mathcal{N}(Wz, WW^T + \sigma^2 I)$$

since

$$\Sigma = \text{Covar}(x) = \mathbb{E}(Wz+\epsilon)\mathbb{E}(Wz+\epsilon)^T = \mathbb{E}(Wz)\mathbb{E}(Wz)^T + \mathbb{E}(\epsilon)\mathbb{E}(\epsilon)^T = W\mathbb{E}(z)\mathbb{E}(z)^T W^T + \sigma^2 I = WW^T + \sigma^2 I$$

in the latter.

PCA can be motivated from a variety of directions. For simplicity, let us start with equation 5.27 and extend it to 5.28.

Consider white Gaussian noise $z \sim \mathcal{N}(0, I)$ and a mean-centered Gaussian data-distribution $p(x) = \mathcal{N}(0, \Sigma)$. Let $x, z \in \mathcal{R}^n$. Given the data-set matrix $X = (x_1, ..., x_N) \in \mathcal{R}^{N \times n}$, we can construct the singular value decomposition

$$X = U\Delta V^T, \tag{5.29}$$

with $U \in \mathcal{R}^N$, $\Delta \in \mathcal{R}^{N \times N}$ and $V \in \mathcal{R}^n$ (in the complex case we need to consider the conjugate transpose of V). Consider the question, which linear transformation of $z \sim \mathcal{N}(0, I)$ yields the distribution $p(x) = \mathcal{N}(0, XX^T)$. The answer is

$$x = Wz = UD^{1/2}z \tag{5.30}$$

as becomes obvious from equation 5.27. Note that U is the matrix of eigenvectors of the linear covariance matrix XX^T and that $D^{1/2}$ is the square-root of its eigenvalue matrix. Thus the linear transformation relating the data-distribution to isotropic Gaussian noise is defined by eigenspace of the covariance matrix. The eigenvalues are called the principal components, the eigenvectors are called principal directions.

Given a dataset X, we can perform the PCA decomposition and yield equation 5.30. By keeping only the d biggest eigenvectors ($D_- \in \mathcal{R}^{d \times d}$) and corresponding eigenvalues ($U_- \in \mathcal{R}^{n \times d}$) , a dimensionality reduction is performed. Note, that using

$$W = U_- D_-^{1/2}$$

in

$$x = Wz$$

leads to

$$x \sim \mathcal{N}(Wz, \Sigma)$$

where

$$\Sigma = WW^T = U_- D_- U^T \in \mathbb{R}^{n \times n}$$

is a singular matrix with rank $d < n$ but dimension n. Along the directions corresponding to the omitted eigenvectors no variability occurs. Since singular distributions can be hard to work with a common practice is adding noise (i.e. setting zero eigenvalues to small values). This addition of noise leads from equation 5.27 to equation 5.28. Consider equation 5.28

$$x = Wz + \epsilon$$

with $W \in \mathbb{R}^{n \times d}$, $z \in \mathbb{R}^d$ and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. From this equation, we obtain

$$p(x) \sim \mathcal{N}(Wz, \Sigma)$$

with $\Sigma = WW^T + \sigma^2 I \approx XX^T$. Consider the mean centered version $p(x) \sim \mathcal{N}(0, \Sigma)$ and perform MLE. with respect to Σ,

$$\Sigma_{opt} = \underset{\Sigma}{\text{argmax}} \prod_{i=1}^{N} \log(\frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp \frac{-x_i \Sigma^{-1} x_i^T}{2}$$

$$= \underset{\Sigma}{\operatorname{argmin}} \sum_{i=1}^{N} \log(\frac{1}{\sqrt{(2\pi)^n |\Sigma|}}) + \frac{-x_i \Sigma^{-1} x_i^T}{2}$$

$$= \underset{\Sigma}{\operatorname{argmin}} -\frac{N}{2} \log((2\pi)^n) - \frac{N}{2} \log(|\Sigma|) - \frac{1}{2} \sum_{i=1}^{N} x_i \Sigma^{-1} x_i^T$$

$$= \underset{\Sigma}{\operatorname{argmin}} -\frac{N}{2} \log(|\Sigma|) - \frac{1}{2} \sum_{i=1}^{N} x_i \Sigma^{-1} x_i^T$$

$$= \underset{\Sigma}{\operatorname{argmin}} -\frac{N}{2} \log(|\Sigma|) - \frac{1}{2} \sum_{i=1}^{N} x_i \Sigma^{-1} x_i^T. \tag{5.31}$$

To determine the extrema examine

$$-\frac{\partial}{\partial \Sigma} \frac{N}{2} \log(|\Sigma|) = -\frac{N}{2} \Sigma^{-1}$$

and

$$\frac{\partial}{\partial \Sigma} - \frac{1}{2} \sum_{i=1}^{N} x_i \Sigma^{-1} x_i^T = +\frac{1}{2} \Sigma^{-1} \sum_{i=1}^{N} x_i x_i^T \Sigma^{-1}.$$

leading to the condition

$$\frac{N}{2} \Sigma^{-1} + \frac{1}{2} \Sigma^{-1} \sum_{i=1}^{N} x_i^T x_i \Sigma^{-1} = 0.$$

Consider $\sum_{i=1}^{N} x_i x_i^T = XX^T = UDU^T$, according to the singular value decomposition (see equation 5.29) so that

$$\frac{N}{2} \Sigma^{-1} + \frac{N}{2} \Sigma^{-1} UDU^T \Sigma^{-1} = 0.$$

and thus

$$\frac{N}{2} + \frac{N}{2} \Sigma^{-1} UDU^T = 0 \rightarrow UDU^T = \Sigma$$

Aligned with the goal of dimensionality reduction, we only keep the d biggest eigenvalues and eigen-directions so that

$$\Sigma = UDU^T \approx U_- D_- U_-^T + \sigma^2 I.$$

Let $W = U_-(D_- - \sigma^2 I)^{1/2}$ so that

$$WW^T = U_- D_- U_-^T + \sigma^2 I \approx \Sigma.$$

D_- contains the d biggest eigenvalues and $\sigma = \frac{1}{n-d} \sum_{j=d+1}^{n} \lambda_j$. Thus, the dimensionality reduction results in

$$x = U_-(D_- - \sigma^2 I)^{1/2} z + \epsilon.$$

If we omit the noise term ϵ, this corresponds to a linear projection of the n dimensional data X onto a subspace of d dimension. This projection is injective, but not surjective. Through the noise-term it becomes bijective.

PCA is a very well understood and popular method. Being completely linear allows for closed form solutions, good interpret-ability and analysis. Linearity is at the same time its strength and main limitation. As a linear transformation of white-noise we can not create a model for nonlinear distributions.

5.5 Variational Inference and Expectation Maximization

Generative models of $p(x, \theta)$ generally involve a third, unmeasured variable $z \sim p(z)$ (the generating latent) so that

$$p(x, \theta) = \int_z p(x, z, \theta) dz. \tag{5.32}$$

In PCA we can solve analytically for the maximum likelihood estimator, assuming that z is drawn from isotropic Gaussian noise. For nonlinear models analytical approaches often fail. A popular theoretical framework for the analysis of 5.32 is variational inference [9].

Consider

$$\begin{aligned}
\log(p(x, \theta)) &= \log \int_z (\frac{p(x, z, \theta)}{q(z)} q(z)) \\
&\geq \int_z q(z) \log \frac{p(x, z, \theta)}{q(z)} \\
&= \mathbb{E}_{z \sim q(z)} (\log \frac{p(x, z, \theta)}{q(z)}) \\
&= \mathbb{E}_{z \sim q(z)} (\log(p(x, z, \theta)) - \mathbb{E}_{z \sim q(z)} (\log q(z)) \\
&= \text{ELBO}(p(x, z, \theta), q(z))
\end{aligned} \tag{5.33}$$

where the inequality comes from the Jensen Inequality. Note that $\mathbb{E}_{z \sim q(z)}(\log q(z)) = -H(q(z))$ is the negative entropy of $q(z)$. This relates maximum likelihood estimation to entropy maximization. While the integral in equation 5.32 might not be tractable, the lower bound, defined by the evidence lower bound (ELBO) often is. Furthermore

$$\begin{aligned}
&\log(p(x, \theta)) - \text{ELBO}(p(x, z, \theta), q(z)) \\
&\log(p(x, \theta)) - \int q(z) \log \frac{p(x, z, \theta)}{q(z)} dz \\
&= \int q(z) \log(p(x, \theta))) dz - \int q(z) \log \frac{p(z|x, \theta) p(x, \theta)}{q(z)} dz \\
&= \int q(z) \log \frac{q(z)}{p(z|y, \theta)} dz \\
&= D_{KL}(q(z), p(z|y, \theta)) dz
\end{aligned} \tag{5.34}$$

Here $\mathrm{D_{KL}}(\cdot, \cdot)$ denotes the D_{KL}-divergence between two distributions (see 5.5.2). Summarizing, we have derived two relationships:

$$\log(p(x, \theta) \geq \text{ELBO}(p(x, z, \theta), q(z)) = \mathbb{E}_{z \sim q(z)}(\log(p(x, z, \theta)) - \mathbb{E}_{z \sim q(z)}(\log(q(z)) \tag{5.35}$$

and

$$D_{KL}(q(z), p(z|x, \theta)) = \log(p(x, \theta)) - \text{ELBO}(p(x, z, \theta), q(z)) dz. \tag{5.36}$$

Expectation maximization algorithms are among the most effective for variational inference problems [34] [52]. They start with a minimization of the gap defined by equation 5.36 by finding $q(z)$ closest to $p(z|y, \theta)$. In essence, by computing the expected distribution $q(z)$. This is equivalent to maximizing the ELBO with respect to z at a fixed θ_t. If $p(z|x, \theta_t)$ can be evaluated, then one can directly set

$$q(z) = p(z|x, \theta_t).$$

Note, that this equality only holds if the D_{KL}-divergence is zero. Usually $q(z)$ is merely an approximation of the true latent-distribution $p(z|x, \theta_t)$. This step is called the Expectation-step because it requires finding the expectation of z given x and θ_t. Then one can compute the log-likelihood under $p(z|x, \theta_t) \approx q(z)$ (sometimes also called Q-function)

$$\mathbb{E}_{z\sim p(z|x,\theta_t)}[\log(p(x, z|\theta)] \approx \mathbb{E}_{z\sim q(z)}[\log(p(x, z|\theta)] \tag{5.37}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\int_z p(z|x_i;\theta_t)\log p(x_i, z|\theta)dz = \frac{1}{n}\sum_{i=1}^{n}\int_z q(z)\log p(x_i, z|\theta)dz = Q(\theta, \theta_t).$$

This quantifies how well the data coincides if latent-samples are drawn from $q(z)$.

In the next step, one aims to find θ_{t+1} which maximizes $Q(\theta, \theta_t)$. This step is called the Maximization-step because one maximizes the likelihood of the data x under the latent distribution $q(z)$ with respect to θ.

Repeated iterations of the Expectation and Maximization step are expected to converge, however, due to a sensitivity to starting parameters, not always to global optima. The Maximization step is equivalent to maximum likelihood estimation. The EM-algorithm can also be used for MAP [14]. In Gaussian Mixture Models $p(z|x, \theta_t)$ can be evaluated, allowing for direct iteration of the EM-algorithm by setting $q(z) = p(z|x, \theta_t)$. However, this direct evaluation is not always possible. In VAEs, $p(z|x, \theta_t)$ can not be evaluated analytically. Instead it is parameterized by an imposed prior distribution $p(z|x)$ and $q(z|x, \phi)$ is optimized w.r.t ϕ through gradient methods and a D_{KL}-divergence loss. The relation of ELBO maximization with iterative maximum likelihood estimation and Kullback-Leibler divergence minimization allows for information geometrical interpretations connecting MLE, EM and VAEs [14].

5.5.1 ELBO - Evidence Lower Bound

The lower bound obtained in equation 5.33 is an ubiquitous quantity in machine learning, especially in variational inference. Section 5.5 shows how it arises as a tractable optimization objective for equations of form 5.32. The results from [31] examine its decomposition into meaningful mathematical formulas. This provides insights into the components which are considered during the optimization process. Let us decompose the formulation of equation 5.35 into its empirical counterpart and use $q(z|x)$ instead of only $q(z)$. This is allowed since the derivation of the ELBO does not limit our choice of $q(z)$. We will also remove the explicit parameters θ to simplify notation. We write

$$\text{ELBO}(p(x, z), q(z|x)) = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{z_n\sim q(z_n|x_n)}(\log(p(x_n|z_n)) - \text{D}_{\text{KL}}(q(z_n|x_n), p(z_n)),$$

where we have n observations of data-samples x_n and latent-samples z_n respectively. Each observation is made with equal probability so that $q(x_n) = p(x_n) = \frac{1}{N}$. Furthermore, let $p(z)$ be an imposed prior (as is the case in many applications), i.e. $p(z|x_n) = p(z)$ does not depend on the observations. However, $q(z_n|x_n)$ does, since it corresponds to a computed map from input-space to latent-space. Examine

$$\text{D}_{\text{KL}}(q(z_n|x_n)), p(z_n)) = \frac{1}{N}\sum_{n=1}^{N} q(z_n|x_n)\frac{q(z_n|x_n)}{p(z_n))} = \sum_{n=1}^{N} q(z_n, x_n)\log\frac{q(z_n, x_n)}{p(z_n)p(x_n)}$$

$$= \sum_{n=1}^{N} q(z_n|x_n)q(x_n)(\log\frac{q(x_n|z_n)q(z_n)}{p(z_n)p(x_n)}) = \sum_{n=1}^{N} q(z_n|x_n)q(x_n)(\log\frac{q(z_n)}{p(z_n)} + \log\frac{q(x_n|z_n)}{p(x_n)})$$

$$= \mathrm{D}_{\mathrm{KL}}(q(z_n), p(z_n)) + \sum_{n=1}^{N} q(x_n|z_n)q(z_n)\log q(x_n|z_n) - \sum_{n=1}^{N} q(x_n|z_n)q(z_n)\log p(x_n)$$

$$= \mathrm{D}_{\mathrm{KL}}(q(z_n), p(z_n)) + \sum_{n=1}^{N} q(x_n, z_n)\log q(x_n|z_n) - \sum_{n=1}^{N} p(x_n)\log p(x_n).$$

$$= \mathrm{D}_{\mathrm{KL}}(q(z_n), p(z_n)) - H(q(x_n|z_n)) + H(q(x_n)).$$

$$= \mathrm{D}_{\mathrm{KL}}(q(z_n), p(z_n)) + I_q(x_n, z_n).$$

The negative of the conditional entropy $H(q(x_n|z))$ and entropy of $H(p(x_n)) = H(q(x_n)) = \log N$ appear. The mutual information of $q(x_n)$ and $q(z_n)$ is $I(x_n, z_n) = H(q(x_n)) - H(q(x_n|z))$ Thus we can write

$$\mathrm{ELBO}(p(x,z), q(z|x)) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{z_n \sim q(z_n|x_n)}(\log(p(x_n|z_n)) - \mathrm{D}_{\mathrm{KL}}(q(z_n), p(z_n)) - I_q(x_n, z_n). \tag{5.38}$$

Through this composition, it becomes apparent, how maximizing the ELBO is related to minimizing the mutual information between x_n and z_n.

Further decomposition into a mutual information $I_q(x, z)$, an entropy $H(q(z))$ and cross-entropy $H(q(z), p(z))$ term is possible. Consider

$$\mathrm{ELBO}(p(x,z), q(z|x)) = \mathbb{E}_{z\sim q(z|x)}(\log(p(x|z)) - \mathrm{D}_{\mathrm{KL}}(q(z), p(z)) - I(x, z)$$

$$= \mathbb{E}_{z\sim q(z|x)}(\log(p(x|z)) - I_q(x, z) + H(q(z)) - H(q(z), p(z)) \tag{5.39}$$

Summarizing, the ELBO maximization corresponds to the minimization of mutual information between x and z, maximization of the entropy of $q(z)$ and minimization of the cross-entropy between $q(z)$ and $p(z)$.

Note, that the prior $p(z)$ appears only in the last term. Contrary to the prior, the other functions are often parameterized and optimized numerically or analytically if possible. This highlights the importance of good priors, since they add negatively to the ELBO and are (generally) not optimizable. We show in section 9.2 and 9.4 how using the isotropic Gaussian prior requires a certain data structure for good generations.

5.5.2 The Kullback-Leibler divergence

The Kullback-Leibler (short KL) divergence (see [3] [5] [54]) is an ubiquitous quantity that reappears in many (generative) machine learning algorithms. Here we provide a geometrical intuition on its meaning and highlight some of its limitations and strengths. In machine learning, one often aims to maximize a likelihood $p(x, \theta)$ or log-likelihood $\log(p(x, \theta))$ by finding the optimal parameters θ. The gradient of the likelihood with respect to the parameters is defined as

$$s(x, \theta) = \frac{\partial \log(p(x, \theta))}{\partial \theta} \tag{5.40}$$

and called the score of the model. It can be shown that, under regularity conditions, the score of the true model $\tilde{\theta}$ is zero. In general, the score indicates parameter sensitivity. A high score means that changing this parameter causes big changes in the likelihood. It is possible to define the Fisher Information matrix as the covariance of the score

$$\mathbf{I}(\theta) = \mathbb{E}\left[\left(\frac{\partial \log p(x, \theta)}{\partial \theta}\right)\left(\frac{\partial \log p(x, \theta)}{\partial \theta}\right)^T\right] \tag{5.41}$$

$$= -\mathbb{E}\left[\frac{\partial^2 \log p(x|\theta)}{\partial\theta\partial\theta^T}\right] = \int_x \frac{\partial}{\partial\theta}\log(p(x,\theta)^2)p(x,\theta)dx.$$

The equalities are non-trivial and their derivation is not provided here. We refer to [79] for a more detailed analysis. Note, that this covariance matrix varies smoothly, depending on the location. It can be shown that $\mathbf{I}(\theta)$ is a Riemannian metric which provides the tangent space of θ with an inner product and can be used to define distances between distributions [53], [4]. The geodesic distance is the path-integral of distances induced in each point by the local covariance matrix. The idea is similar to the Mahalanobis distance [13], but, due to the covariance interpretation in terms of model parameter sensitivity (contrary to data-variability in Mahalanobis), one does not use the inverse of the covariance matrix.

Consider a path $\gamma(t)$ which is related to the displacement of the parameters θ by

$$d\theta = \frac{d\gamma(t)}{dt}dt.$$

The squared infinitesimal Riemannian distance with the Fisher-information matrix as metric tensor is given by

$$ds^2 = d\theta^T I(\theta)d\theta = \frac{d\gamma(t)}{dt}I(\theta)\frac{d\gamma(t)}{dt}dt. \tag{5.42}$$

When $I(\theta)$ is large, indicating a high likelihood sensitivity of a parameter, then the distance in that direction is also large. If we move in directions of high sensitivity (i.e. corresponding to high information content in that direction) we traverse a larger distance.

The distance between θ_1 and θ_2 becomes

$$d_G(\theta_1,\theta_2) = \int_0^1 \sqrt{\frac{d\gamma(t)}{dt}^T I(\gamma(t))\frac{d\gamma(t)}{dt}}\,dt = \int_0^1 ds, \tag{5.43}$$

where $\gamma(0) = \theta_1$ and $\gamma(1) = \theta_2$.

Through this geodesic distance, the closeness of distributions can be quantified. The geodesic distance is symmetric and a true distance. Since direct computation might be intractable, often quadratic form approximations are used, so called divergences. A famous case is the D_{KL}-divergence which is defined as

$$D_{KL}(p(x|\theta)\|q(x)) = \int p(x)\log\frac{p(x)}{q(x)}dx \tag{5.44}$$

For small displacements, it can be shown that

$$D_{KL}(p(x|\theta)\|p(x|\theta+d\theta)) \approx \frac{1}{2}\,d\theta^T I(\theta)d\theta.$$

Employing this approximation of the geodesic distance solves computational issues but destroys distance properties such as symmetry. Some algorithms, such as the variational autoencoder, use the D_{KL}-divergence as a regularizer, optimizing a parameterized distribution $q(z|x)$ so that it becomes similar to another distribution $p(z|x)$. Furthermore, minimizing the D_{KL}-divergence can be equivalent to maximum likelihood estimation [14]. Consider standard maximum likelihood estimation

$$\hat{\theta} = \arg\max_\theta \sum_{i=1}^N \log p_\theta(x_i) \tag{5.45}$$

and the discrete D_{KL}-divergence between the empirical data-distribution $x_i \sim p(x)$ and parameterized model distribution $q_\theta(x)$

$$D_{KL}(p(x)|q_\theta(x)) = \sum_{i=1}^{N} \log \frac{1}{q_\theta(x_i)} = -\sum_{i=0}^{N} \log q_\theta(x_i). \tag{5.46}$$

Minimizing equation 5.46 with respect to θ is equivalent to maximizing equation 5.45. However, the D_{KL}-divergence is not symmetric so that this equivalence does not hold for $D_{KL}(q_\theta(x)|p(x))$, although, if the zero is reachable, both have the same optimum. The D_{KL}-divergence requires both distributions to live in the same space and have significant overlap. If $q(x) = 0$ where $p(x) \neq 0$ the D_{KL}-divergence is not defined. If $p(x) = 0$ where $q(x) \neq 0$ the D_{KL}-divergence is zero. This can create differentiability problems and lead to unstable training. Thus, some algorithm replace it by other similarity measures, such as the Wasserstein distance. An popular example is the Wasserstein-GAN [6] which stabilizes training by replacing optimization with respect to a divergence with optimization with respect to the 1-Wasserstein distance [39].

5.6 VAE - Variational Auto-Encoder

The variational auto-encoder [15] is an extension of the auto-encoder. The latter considers a function

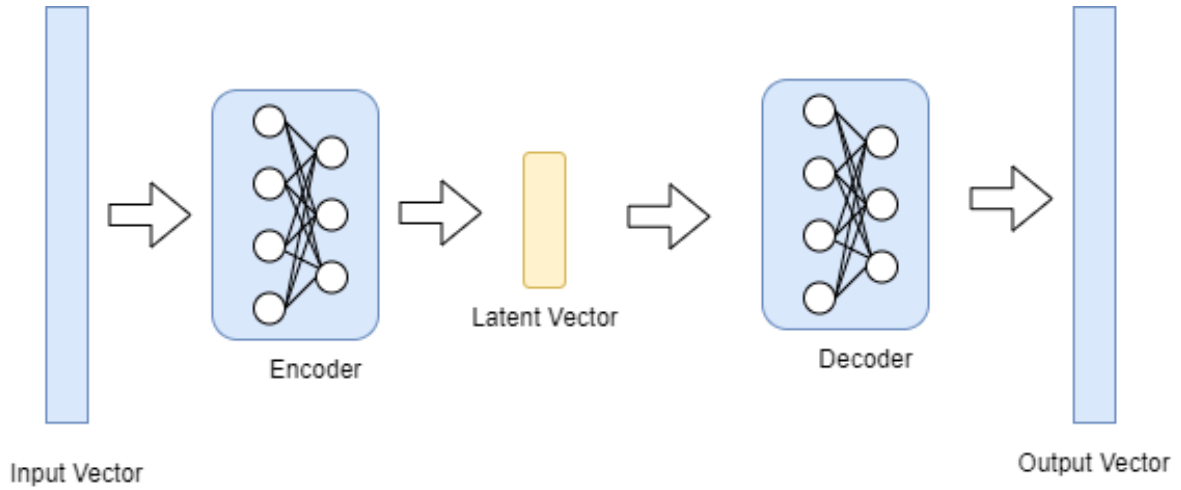
$$z = g_\phi(x)$$

(the encoder, parameterized by a neural network) to compress data and another function

$$x = f_\theta(z)$$

(the decoder, parameterized by a neural network) to decompress data. The dimensional space into which data-samples are mapped is called the latent space. Usually a dimensionality reduction is performed. Encoder and decoder are trained so that the encodings are mapped to

Figure 5.3: An Auto-Encoder



decodings which are as close to the originals as possible. The information in the input vector is compressed into the latent space and retrieved by the decoder. Model-optimization is performed via back-propagation, so that the reconstruction loss

$$\sum_{i=1}^{N} (x_i - f_\theta(z_i))^2 = \sum_{i=1}^{N} (x_i - f_\theta(g_\phi(x_i)))^2 \tag{5.47}$$

is maximized. Here x_i corresponds to the original data-sample and z_i to the corresponding low-dimensional encoding. Note that equation 5.47 corresponds to the maximum likelihood objective (see section 5.2) of a Gaussian data-distribution. If the data distribution does not align well with the Gaussian assumption, then other choices should be made. The auto-encoder enables the definition of a mapping from latent-space to data-space and vice-versa. The latent-space is unrestricted in its structure and the mapping deterministic. This is similar to (non-bayesian) gaussian process latent variable models [76], however through a neural network instead of a covariance parametrization.

The main difference between an auto-encoder and a variational auto-encoder is that the latter considers the latent vector z as coming from an imposed prior distribution with density $p(z)$ (similar to Bayesian Gaussian process latent variable models). This leads to variational inference with unobserved latent variable z. The prior distribution is often chosen as the isotropic Gaussian (as in principal component analysis). By learning a distribution, a structured, continuous latent-space, with neighboring latents corresponding to similar outputs, is learned. During training, each value has a certain probability of occurring and is mapped to the best corresponding decoding. The encoder learns a mapping from the output space to the latent distribution via $z \sim \mathcal{N}(\mu_z, \Sigma_z) = q_\phi(z|x)$ where $\mu_z, \Sigma_z = g_\phi(x)$ are parameterized by the neural network $q_\phi(z|x)$. Consider the decoder

$$x = f_\theta(z)$$

with $z \sim \mathcal{N}(0, \Sigma)$. Then

$$p_\theta(x|z) \sim \mathcal{N}(f_\theta(z), \Sigma) \text{ and } p(x, z) = p_\theta(x|z)p(z).$$

As in variational inference we can derive a lower bound on the data-likelihood

$$p_\theta(x) = \int p_\theta(x, z)dz \geq \int_z q_\phi(z|x) \log \frac{p_\theta(x, z)}{q_\phi(z|x)} dz = \text{ELBO}(p_\theta(x, z), q_\phi(z|x)).$$

Note, that

$$\begin{aligned}
\text{ELBO}(p_\theta(x, z), q_\phi(z|x)) &= \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(z|x)) - \mathbb{E}_{z\sim q_\phi(z|x)}(\log(q_\phi(z|x)) \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(z|x)p_\theta(z)) - \mathbb{E}_{z\sim q_\phi(z|x)}(\log(q_\phi(z|x)) \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(z|x)) + \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(z))) - \mathbb{E}_{z\sim q_\phi(z|x)}(\log(q_\phi(z|x)) \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(x|z)) - \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x), p_\theta(z)),
\end{aligned} \tag{5.48}$$

which is the usual expression for the objective employed in variational auto-encoders. Since $p_\theta(z)$ can not be evaluated directly it is parameterized by a (usually fixed) imposed prior distribution $p(z)$. This is often an isotropic Gaussian (similar to PCA). However other distributions can be chose, such as the asymmetric Laplace distribution [49].

Summarizing, the objective function of a VAE is defined as

$$\theta_{opt}, \phi_{opt} = \underset{\theta,\phi}{\text{argmax}} \left[\mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(x|z)) - \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x), p(z))\right] \tag{5.49}$$

with the first part corresponds to the log-likelihood of the data given a latent vector and a decoder parametrization, while the second part corresponding encourages a latent-distribution close to the prior. We can optimize the first part with (supervised) maximum likelihood estimation while term corresponds to a minimization of the D_{KL} divergence (unsupervised). This positions the VAE in the realm of semi-supervised models. For the evaluation of $\mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(x|z))$ Monte-Carlo Approximations can be used, so that

$$\mathbb{E}_{z\sim q_\phi(z|x)} \log(p_\theta(x|z) \approx \sum_{i=0}^{N} \log(p_\theta(x|z_i)). \tag{5.50}$$

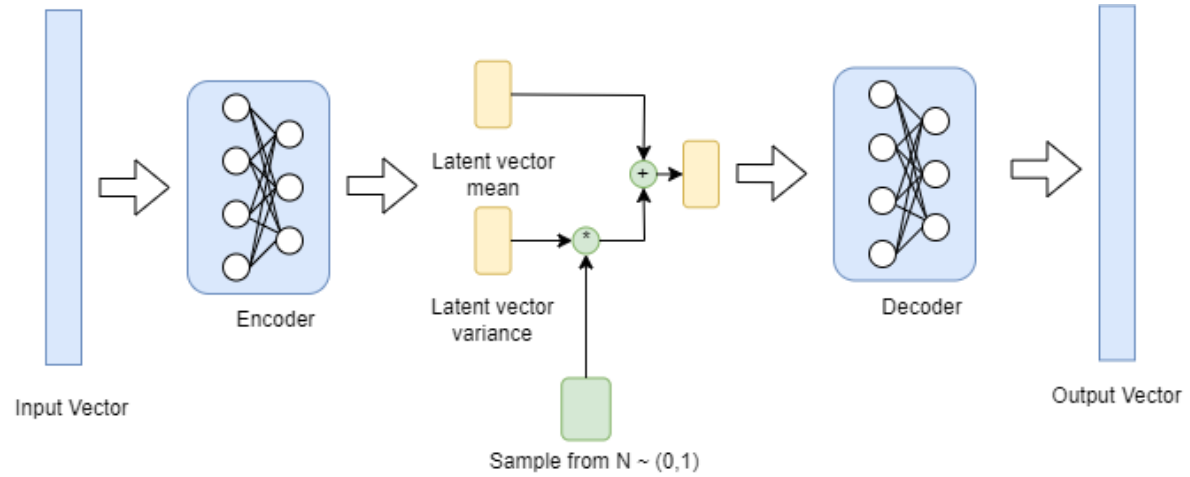
where $z_i \sim q_\phi(z|x)$. If one assumes $p_\theta(x|z) = \mathcal{N}(f_\theta(z), \sigma(z)^2 I)$, then the maximum likelihood estimator θ_{opt} is given by (see section 5.2)

$$\theta_{opt} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} (x_i - f_\theta(z_i))^2.$$

This is equivalent to the objective used in autoencoders (see equation 5.47). We can then write the objective function of equation 5.53 as a loss-function

$$\theta_{opt}, \phi_{opt} = \underset{\theta,\phi}{\operatorname{argmin}} \left[\sum_{i=1}^{N} (x_i - f_\theta(z_i))^2 + \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x), p(z)) \right] \tag{5.51}$$

Figure 5.4: The Variational Auto-Encoder



5.6.1 Potential Pitfalls

If the encoder perfectly maps different inputs x_i to the isotropic Gaussian, so that

$$q_\phi(z_i|x_i) = \mathcal{N}(0, I) \; \forall x_i \in X,$$

all information is lost since different samples x_i of the dataset X lead to exactly the same distribution of z_i. This event, which happens when the D_{KL}-loss is zero for multiple inputs x, is called mode-collapse, and is generally undesired. Instead $\sum_{i=0}^{N} q_\phi(z|x_i)$ should be a multi-modal distribution (with as many modes as training examples, each with different means and variances). Mode-collapse is caused by the regularizing term $\mathrm{D}_{\mathrm{KL}}(q_\phi(z|x), p(z))$, which encourages $q_\phi(z|x_i)$ to follow the prior $p(z)$ for all x_i. Thus the regularizing term is often softened though a weighting hyper-parameter β leading to the objective

$$\mathbb{E}_{z \sim q_\phi(z|x)}(\log(p_\theta(x|z)) - \beta \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x), p(z)). \tag{5.52}$$

If the prior $p(z)$ is the isotropic Gaussian, the actual encoder distribution $q_\phi(z|x_i)$ should deviate from $p(z)$ for all, but potentially one sample x_i. Choosing the right prior is non-trivial but very important. The wrong prior can destroy all applicability of the variational auto-encoder. Another issue can be an uninformative latent spaces due to posterior-collapse, a phenomenon in which some latent dimensions are filled with random noise and ignored completely during generation.

5.6.2 The re-parametrization trick

For neural network optimization the loss-function must be differentiable. However, if we consider equation 5.53 with the mean squared error loss, we obtain

$$\sum_{i=1}^{N}(x_i - f_\theta(z_i))^2 - \beta \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x_i), p(z)), \tag{5.53}$$

where z is a random variable and thus not directly differentiable. This is solved with a re-parametrization. Consider the Encoder $z = g_\phi(x)$. We want z to be a random variable, so that so that the decoder $f_\theta(g_\phi(x))$ remains differentiable with respect to θ. Consider an encoder of the form $\mu(x) = g_{\mu,\phi}(x),\ \Sigma(x) = g_{\Sigma,\phi}(x)$. We can write

$$z = \mu(x) + \Sigma(x)\epsilon \sim \mathcal{N}(\mu(x), \Sigma(x)),$$

with $\epsilon \sim \mathcal{N}(0, I))$. Due to this construction, also called the re-parametrization trick we can sample a random variable in differentiable way by a parametrization of $\mu(x)$ and $\Sigma(x)$. Often (but not always), $\Sigma(x) = \sigma(x)I$ is chosen as a diagonal covariance matrix. The re-parametrization trick can also be applied to non-gaussian distributions.

5.6.3 Connections to PCA

As shown in [64] and [88] VAE pursues PCA directions (see section 5.4) locally. Initially, they show that disentangled latent features (independent from each other) are sensitive to rotations of the latent space (the coordinate system matters). A disentangled representation is understood as encoding one specific "aspect" of the data, represented by a direction of independent variability. The transformation obtained by the principal component eigen-decomposition is orthogonal, i.e. the linear transformation matrix has pairwise orthogonal columns, and is thus an axes-preserving linear mapping. It aligns with the principal directions of variance in the data, enabling a representation as their linear combination. The loadings of each column can be gathered in a diagonal matrix. However, any invertible linear map can be multiplied to the axes-preserving transformation matrix without destroying reconstructive abilities. Then the columns are no longer disentangled since they correspond to linear combinations of the eigenvectors representing main variability directions. PCA performs eigenvalue decomposition on the covariance matrix Σ, finding its eigenvectors $\mathbf{v}_i$ and eigenvalues λ_i. Suppose we apply a rotation R (an orthogonal matrix, so $R^T R = I$) to the data matrix X, yielding a new data matrix X':

$$X' = XR$$

The covariance matrix of the rotated data is:

$$\Sigma' = \frac{1}{n}X'^T X' = \frac{1}{n}(XR)^T(XR) = \frac{1}{n}R^T X^T X R = R^T \Sigma R$$

Thus, the new covariance matrix Σ' is a similarity transformation of Σ. This means that Σ' and Σ have the same eigenvalues. Since the covariance matrices Σ and Σ' share the same eigenvalues, their eigenvectors (principal components) will be the same up to a rotation. The new eigenvectors $\mathbf{v}'_i$ of Σ' are related to the original eigenvectors $\mathbf{v}_i$ by the rotation matrix R:

$$\mathbf{v}'_i = R\mathbf{v}_i$$

Therefore, the principal components of the rotated data are just the rotated versions of the principal components of the original data. This shows that standard PCA is invariant to rotations of the latent space. However, a disentangled representation in a latent space is destroyed by rotation. The proof in [64] reformulates the VAE loss (within a specific but plausible parameter

range) in terms of active and passive variables. The latter are shown to have a low influence on the D_{KL}-regularization. If the decoder ignores the passive variables (i.e. its derivative with respect to passive variables is zero), the network is said to operate in a polarized regime. This assumption allows for a decomposition of the loss into a deterministic and stochastic part. In the following, the main expressions are stated. For more details, the reader is referred to [64]. The deterministic part is written as

$$L_{\text{rec}}(x_i) = \|\text{Dec}_\theta(\mu(x_i)) - x_i\|$$

the stochastic part as

$$\hat{L}_{\text{rec}}(x_i) = \mathbb{E}\|\text{Dec}_\theta(\mu(x_i)) - \text{Dec}_\theta(\text{Enc}_\phi(x_i))\|.$$

The latter is simplified to

$$\mathbb{E}_{\epsilon(x_i)} \|\text{Dec}_\theta(\mu(x_i)) - (\text{Dec}_\theta(\mu(x_i)) + J_i\epsilon(x_i))\|^2 = \mathbb{E}_{\epsilon(x_i)}\|J_i\epsilon(x_i)\|^2.$$

where J_i denotes the Jacobian of the decoder

$$J_i = \frac{\partial \text{DEC}(\mu(x_i))}{\partial \mu(x_i)}.$$

The stochastic loss minimization is formalized as optimization problem

$$\min_{V_i,\sigma_i} \sum_{x_i \in X} \log \mathbb{E}_{\epsilon(x_i)}\|J_i\epsilon(x_i)\|^2 \quad \text{s.t.} \quad \sum_{x_i \in X} D_{KL}(x_i) = C.$$

Every minimum of this optimization problem is global and so that the columns of the Jacobians J_i are orthogonal. This supports the statement that a VAE searches for a trade-off between good reconstructions and orthogonal directions. The effect is localized by the consideration of multiple Jacobians J_i for each input x_i. Rotations destroy disentanglement while maintaining orthogonality. The VAE objective with a Gaussian prior is invariant with respect to rotations. Due to the initialization dependent performance variance of VAE's both, disentangled and entangled representations might be learned. Several approaches exist to encourage disentangled representation learning. [88] shows how modifications of the training procedure (specifically, through a dataset manipulation) can be used to favor the disentangled representation. [49] uses an asymmetric prior distribution, so that the latent representation is no longer rotation invariant.

5.7 Comparing PCA and VAE

Building on machine learning basics, such as MLE (section 5.2) and MAP (section 5.3), we showed that PCA (section 5.4) can be considered a particularly simple (i.e. linear) form of variational inference (section 5.5) with an analytical solution. In more complicated (intractable) cases, a tractable lower bound (section 5.5.1) replaces the objective that is optimized. It can be formulated in different ways and be decomposed into several information theoretical terms. This decomposition highlights the importance of an adequate prior distribution. Furthermore, maximizing the ELBO is intimately related with a minimization of the D_{KL}-divergence (section 5.5.2)). The D_{KL}-divergence can be interpreted as a quadratic form approximation of the geodesic distance between probability distributions. Variational auto-encoders (section 5.6) are semi-supervised machine learning algorithms which maximize the ELBO objective through minimization of the D_{KL}-divergence and the reconstruction loss. The minimization of the D_{KL}-divergence corresponds to fitting a posterior mapping $q(z|x)$ to $p(z|x)$. Since $p(z|x)$ is not known and can usually not be evaluated, it is parametrized by an assumed prior distribution $p(z)$,

often the isotropic Gaussian. Thus, minimization of the D_{KL}-divergence encourages a latent-distribution where each sample x is mapped close to the prior. Possible problems can be mode-collapse, posterior collapse and implausible prior assumptions. We also mention that VAEs with an isotropic Gaussian prior can be considered a non-linear variant of PCA. As such, the reconstructive abilities are invariant to rotations, meaning that the learned representations are not necessarily disentangled. Figure 5.5 and figure 5.6 display the application of PCA and VAEs to a gaussian and non-gaussian data distribution. In the gaussian case, both models are capable of generating samples from the data-distribution. This experiment shows the equivalence of both models in linear regimes. In the non-linear case, PCA can find the principal axes of the data, but models a gaussian data-distribution which is not aligned with the real data-distribution. Contrary to this, the VAE can model the nonlinear data distribution. This shows that the VAE can learn a potentially non-linear map from isotropic gaussian noise to the data-distribution, while PCA always parametrizes a linear map. However, as displayed in figure 5.8, even the variational autoencoder might struggle to represent certain data distributions.

Figure 5.5: Test on a gaussian data distribution

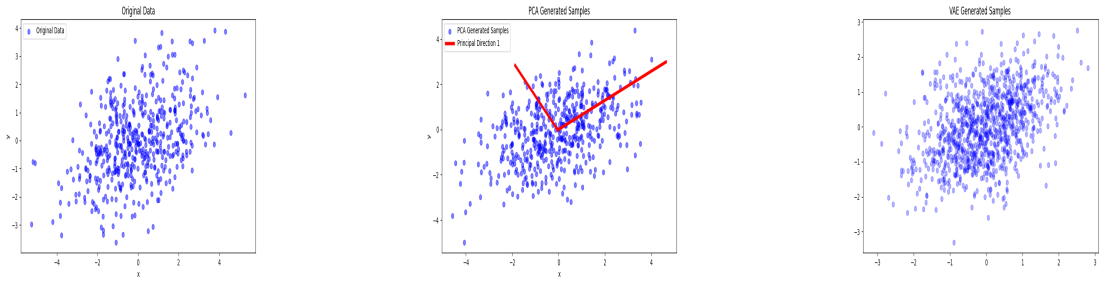


Figure 5.6: Test on a non-gaussian data distribution

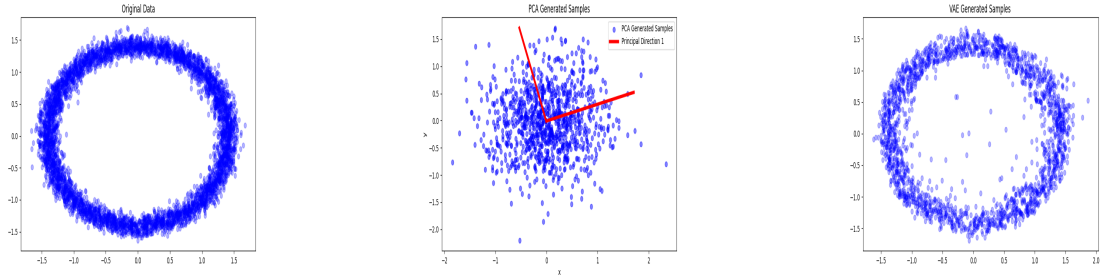


Figure 5.7: Test on a non-gaussian data distribution

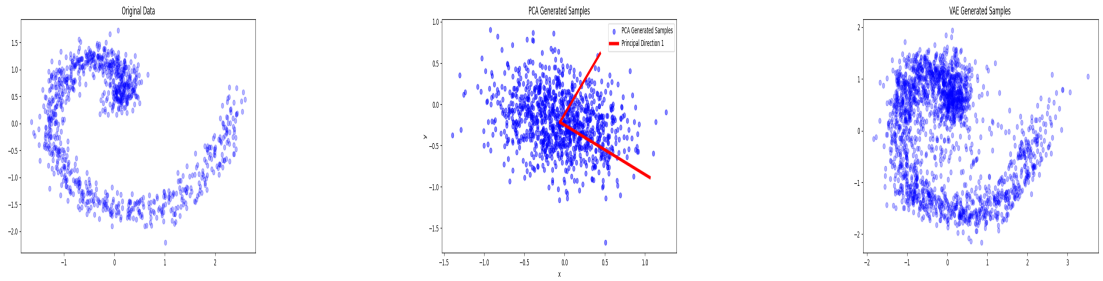
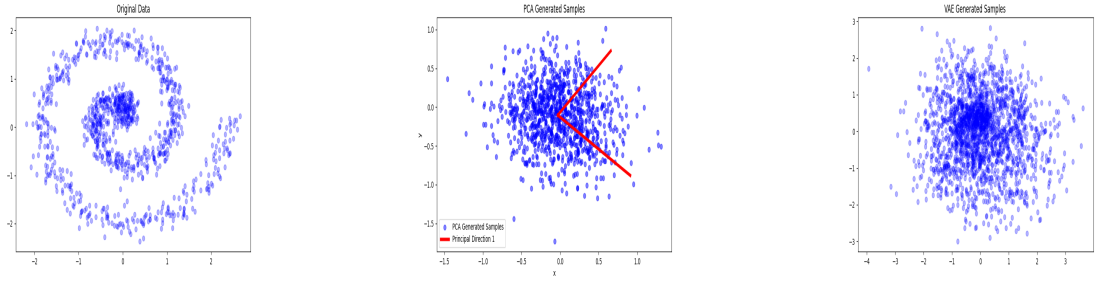


Figure 5.8: Test on a non-gaussian data distribution



5.8 Gradient descent

All machine learning problems can be viewed as optimization problems within a very large parameter space. Some metric, called *objective* or *loss* function, is defined. This metric corresponds to the performance of the model. A way of determining extreme points, when no direct analytical solution can be found, is through a numerical scheme called *gradient descent*. Consider a loss function $L(\theta)$. For example, in the supervised learning setting, we might choose a model

$$\vec{y} = h_\theta(\vec{x}) + \epsilon$$

with $\epsilon \sim \mathcal{N}(0, \tau^2)$ with corresponding loss-function

$$L(\theta) = \frac{1}{2}(h_\theta(\vec{x}) - \vec{y})$$

where

$$h_\theta(\vec{x}) = \vec{\theta}^T \Phi(\vec{x}) + \vec{b}.$$

Note, that for the linear kernel $\Phi(\vec{x}) = \vec{x}$ we can find the optimum analytically. However, we can choose any other continuous function. One can minimize $L(\theta)$, without knowing much about its form, by examining its Taylor-expansion and considering only the first few terms

$$L(\theta + s) \approx L(\theta) + \nabla L(\theta)^T * s.$$

The Goal is to learn θ which minimizes $L(\theta)$. Herefore, the simplest numerical update scheme (*gradient descent*) is:

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta L(\theta).$$

Then

$$L(\theta_{t+1}) = L(\theta_t - \gamma \nabla_\theta L(\theta)) \approx L(\theta_t) - \gamma \nabla_\theta L(\theta)^T \nabla_\theta L(\theta) \leq L(\theta_t)$$

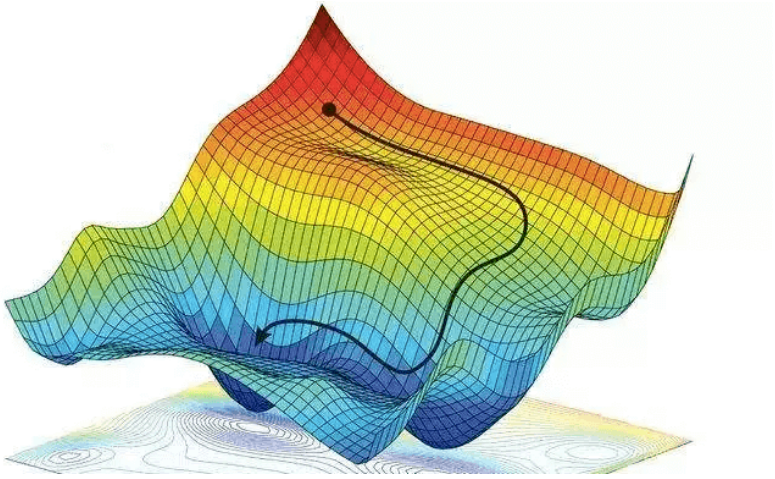
which proves

$$L(\theta_{t+1}) \leq L(\theta_t)$$

The weight θ_t is moved in the direction of its negative gradient. In essence, if θ_t has an increasing effect onto the *loss* function (infinitesimal increments of θ_t create changes proportional to $\nabla_\theta L(\theta_t)$), it is moved towards the negative direction of $\nabla_\theta L(\theta_t)$. This means that weights which increase $L(\vec{x}, \theta)$ are decreased, while weights which decrease $L(\theta)$ are increased. The parameter γ is called the *learning rate* and corresponds to a proportionality factor in the update scheme. Big values of γ mean big update steps, small values mean small update steps. If the loss-function is strictly convex and γ is chosen right, this scheme guarantees finding the parameters θ which minimize it. The algorithm can be written as:

1: **procedure** Simple Gradient Descent
2: **initialize parameters** θ_0

Figure 5.9: A loss landscape and a descent path



https://mriquestions.com/back-propagation.html

initialize threshold ϵ
initialize flag $converged = false$
initialize $k = 1$
repeat
  Compute $g = \frac{\partial L}{\partial \theta_{k-1}}$
  Update $\theta_k = \theta_{k-1} - \gamma * g$
  if $||\theta_k - \theta_{k-1}|| \leq \epsilon$ **then**
    $converged \leftarrow true$
  end if
until $converged == true$

If $L(\theta_k)$ has multiple local minima, or γ is too large, no convergence to a global minimum can be guaranteed. Optimizing the gradient descent algorithm to enable faster convergence to global minima is a field of on-going research. Problems that can occur, are getting stuck in sub-optimal minima, diverging loss due to exploding gradients or failing to converge due to vanishing gradients. A common methodology to address these problems is choosing adaptive learning-rate schemes in an informed manner. Figure 5.9 shows a two-dimensional loss landscape and a possible descent path following negative gradients.

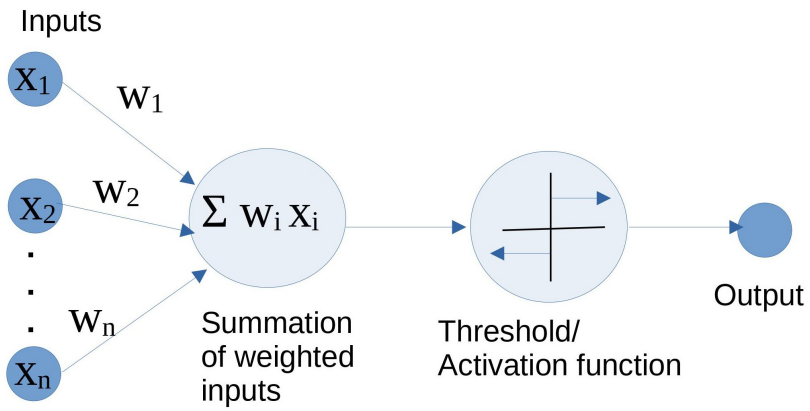
5.9 Artificial neural networks

An artificial neural network contains a mathematical model which can be viewed as a high level abstraction of a neuron. A node receives weighted inputs from its connected nodes and outputs a value if the sum of all inputs surpasses a specified threshold (see figure 5.10). A method called back-propagation is then used to update the weights and thresholds through gradient descent.
The computation across a single layer can be represented as a linear function which is passed through a nonlinear activation function,

$$f_{out}(x) = \sigma(W_\theta f_{in}(x) - b) \tag{5.54}$$

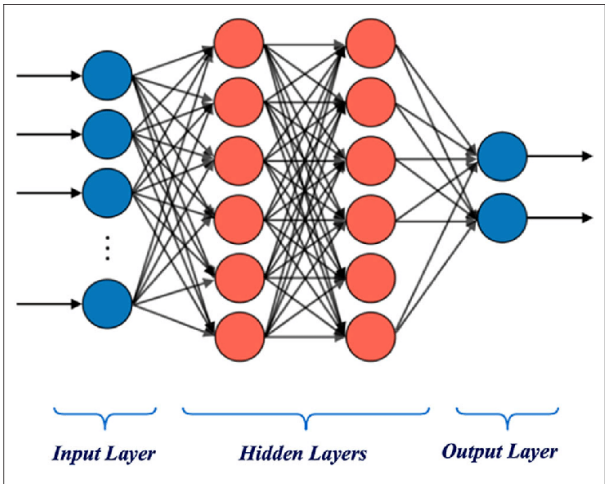
where $f_{in}(x) \in \mathbb{R}^n$ is the input feature vector corresponding to the output of n neurons, $f_{out}(x) \in \mathbb{R}^m$ the output feature vector corresponding to the output of m neurons, $W_\theta \in \mathbb{R}^{mxn}$ is a weight matrix and $b \in \mathbb{R}^n$ a bias vector representing the threshold for firing. The output of the linear computation is fed through an element-wise non-linear activation function. Neural Networks (see figure 5.11) correspond to a sequential application of equation 5.54. The non-linear activation

Figure 5.10: The perceptron / A single artificial neuron



https://www.analyticsvidhya.com/blog/2021/06/beginners-guide-to-universal-approximation-theorem/

Figure 5.11: Structure of a multilayer perceptron neural network



https://www.researchgate.net/figure/Structure-of-multilayer-perceptron-neural-network_fig2_342852638

function enables neural nets to learn any continuous function by the universal approximation theorem.

5.9.1 The universal approximation theorem

Artificial neural nets are very powerful function parameterizations due to the universal approximation theorem. This theorem states that any nonlinear function can be approximated by a multi-layer perceptron (see figure 5.11). It does not state how the multi-layer perceptron needs to be constructed (i.e. which depth, width and activation function is required) nor which approximation accuracy is obtained. Several formal proofs exist and all of them are quite involved. We will only provide an intuition. Consider a single neuron, which receives n inputs x_1, to x_n respectively. The values are summed and passed through a non-linear activation function. Consider as activation function a sigmoid function. Note, that when the summation of weighted input exceeds a threshold, then the output is one, if it is below a threshold, then the output is zero. For the sake of the argument, let us assume a very steep slope so that we consider these two as the only possible outputs. This constructs a step function. Within a thought experiment, let us try to construct a rectangular function. It is one only within a specific range and zero

elsewhere. Consider two step functions, each with a different threshold. Then just subtract one from the other. This results in a rectangular function with value one only above the lower threshold and below the upper threshold and zero elsewhere. This function can be constructed by two artificial neurons, whose outputs are subtracted. A generalization of this subtraction could be a weighted sum of the outputs. Like this we could construct rectangular functions with arbitrary values and arbitrary ranges. Note, that any continuous function can be approximated by rectangular functions. For reasons of visualization we only consider one-dimension, but the same holds in higher dimensions. This theorem is at the heart of artificial neural networks. Often, the exact function is not known. Therefore, one assumes that it can be represented by a neural net and that the parameters can be found through gradient descent. A downside of the universal approximation theorem is that there is no statement regarding the form of the neural network. Thus the depth, width and activation functions are determined mainly be trial and error.

5.10 Deep Learning Libraries

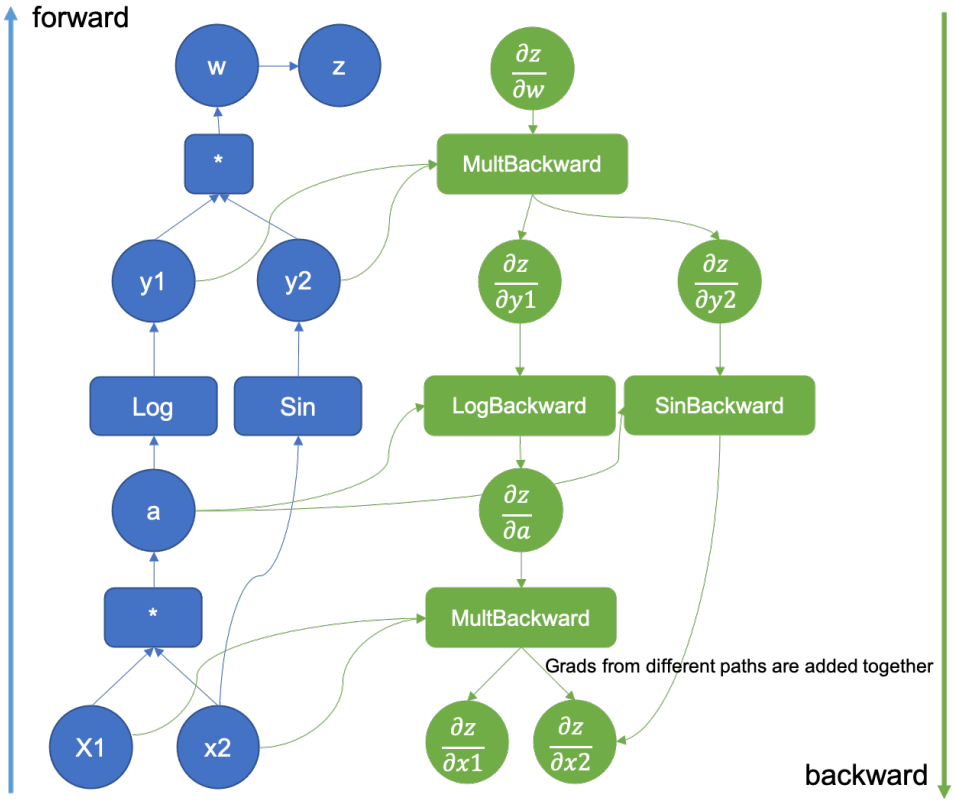
Within most deep learning library a complex, differentiable function is specified. Usually by terms of layer width, layer depth, activation functions (i.e. the model parameterization) and a loss-function. Then the output of this non-linear function, which is composed of the neural network and its loss function, is minimized. The problem of minimizing (or maximizing) a function is called an optimization problem. A common algorithm to tackle these problems is gradient-descent. For strictly convex function it is guaranteed to find the global minimum if the step size is chosen adequately. Functions containing complex artificial neural networks usually do not satisfy this strict convexity constraint. However, gradient descent still converges to local minima and exploration strategies can help to find the lowest possible minimum. These strategies are part of the optimization process. Since the implementation of such an algorithm for a large set of possible parameters is very labour intensive, programmers tend to rely on the deep learning libraries, such as PyTorch. Here, the optimization procedure is computed without requiring user-interference. The degrees of freedom the programmer has during the implementation of such an optimization problem are:

- Choice of data - this parameter corresponds to the model input x
- Parameterization of layers and activation functions of the model (described by $f_{\theta_{model}}$)
- Parameterization of the loss function (described by $l_{\theta_{loss}}$)
- Choice of the optimizer, defining the optimization strategies during back-propagation.

A very simple Pseudo-code example for the implementation of a neural network is:

1: **procedure** TRAIN MODEL
2: **load** data
3: **parameterize** model $f_{\theta_{model}}$
4: **parameterize** loss $l_{\theta_{loss}}$
5: **choose** optimizer $optimizer(\theta_{model})$
6: **for** iteration $\leq$ maximal number of iterations **do**
7: **for** x in data **do**
8: **compute model output** $y \leftarrow \mathrm{f}_{\theta_{model}}(x)$
9: **compute loss** loss $\leftarrow \mathrm{l}_{\theta_{loss}}(y)$
10: **compute gradients**
11: **update model parameters**
12: **end for**
13: iteration = iteration +1

Figure 5.12: An example of an augmented computational graph



https://pytorch.org/blog/computational-graphs-constructed-in-pytorch/

14: **end for**
15: **end procedure**=0

The *loss* object is a PyTorch-tensor. Its *backward* method propagates the gradient to other tensors with which it is connected. The optimizer is predefined by PyTorch and receives the model parameters. Its *step* method applies one optimization step to the model parameters. This optimization step depends on the tensor-gradients computed by the *backward* method.

5.10.1 Back-propagation, computation graphs and automatic differentiation

An important concept for the optimization of neural networks is back-propagation. This algorithm is the basis for the optimization process. It essentially consists in computing the gradients of the latest (output) layer and *propagating* it backwards by the chain rule to the previous layer. The process is repeated until the input-layer is reached. Through this the gradient of the output-function with respect to each parameter is computed. The parameters are then shifted in a proportional manner. PyTorch uses a computation graph to handle back-propagation. During the definition of the neural net tensors (e.g. y_1, y_2, x_1, x_2, z), the corresponding gradient is computed in the backwards path. Sums, Multiplications and special Operations like the log(), sin() or ReLu() are also included. If one performs special operations, for example by defining a loss-function which is not differentiable everywhere, it is possible to specify the gradient-computation manually. This specifies how the gradient flows through the custom function. An example definition of pytorch tensors is provided by

```
x = torch.tensor([0.5, 0.75], requires_grad=True)
v = x[0] * x[1]
v
```

`tensor(0.3750, grad_fn=<MulBackward0>).`

Notice, here the parameter

```
for i in range(0,max_epoch_number)
  batch_label = get_batch_label()
  model_input = get_input()
  prediction = model(input)
  loss = loss_fct(batch_label, model_input)
  loss.backwards()
  optimzer.step()
```

Notice, that the *loss.backwards()*, the *optimizer.step()* and the *model()* and functions are connected in the background. The programmer does not need to program this connection. Automatically, the optimizer step uses the gradient information from the backwards pass to update the model parameters.

5.10.2 Optimizer

Pytorch uses the optimizer class to perform parameter updates based on gradients computed in the backwards pass. Different classes exist. Commonly used types are the *SGD* (Stochastic Gradient Descent) and the *Adam* optimizers. Each allows for the specification of parameters, the most general being the learning rate. This parameter is a constant factor related to the update magnitude. During initialization the optimizer also receives the list of parameters which are to be updated. Usually these are simply the model parameters, however it is possible to use user-defined parameters. Allowing for example to update multiple models with the same optimizer or, as in Auto-decoders [86], to update a latent-space directly. In this thesis, we use the *Adam* optimizer. The name comes from *adaptive moment estimation*. It has little memory requirements, is computationally efficient and very appropriate for a wide variety of machine learning tasks. The optimizer allows for the specification of four parameters.

- α, corresponding to the learning rate, the proportion that weights are updated (common range is 0.1 to 1e-6)
- β_1, corresponding to the exponential decay rate for the first moment estimates (often 0.9)
- β_2, corresponding exponential decay rate for the second-moment estimates (often 0.999)
- ϵ, a very small number to prevent division by zero

Consider the momentum

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)[\frac{\partial L}{\partial \theta_t}]$$

and the quantity v_t (for root mean square propagation [41], [66])

$$v_t = \mathbb{E}\left[\frac{\partial L}{\partial \theta_t}\right]^2$$

$$= \beta_2 v_{t-1} + (1-\beta_2)[\frac{\partial L}{\partial \theta_t}]^2$$

$$= \beta_2 \mathbb{E}\left[\frac{\partial L}{\partial w_{t-1}}^2\right] + (1-\beta_2)\frac{\partial L}{\partial \theta_t}^2$$

Since both parameters are initialized at zero, they have a tendency to be biased towards zero as β_1 and β_2 go towards one. This allows for a combination of root mean square propagation and momentum by considering the update step

$$\theta_{t+1} = \theta_t - m_t \frac{\alpha}{\sqrt{v_t} + \epsilon}.$$

Note, that the momentum m_t adds a fraction of the previous gradient to the current gradient. If the previous gradient has been pointing consistently in the same direction, the momentum term will accumulate and accelerate optimization in that direction. The hyper-parameter β_1 is the momentum decay which exponentially decays past momentum vectors. The parameter v_t is a weighted moving average of the squared gradient corresponding to the variance of the gradient. If the variance is high, the step size is reduced.

The Adam optimizer uses

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \text{ and } \hat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

to update the weights according to

$$\theta_{t+1} = \theta_t - \hat{m_t} \frac{\alpha}{\sqrt{\hat{v_t}} + \epsilon}.$$

This corresponds to a warm-start, where the influence of momentum and root mean square propagation are incremented over time.

Chapter 6

Introduction into my contributions

Through usage of the software, algorithms and concepts explained in part one we aim to create a useful generative model for ischemic scar shapes.

In chapter 7 we examine the available pre-processing steps, provide summaries of used algorithms and highlight the results as well as potentially problematic processing artifacts. Without addressing these further, the 3D models provided by the pre-processing are accepted as ground-truth references for the examination of the generations.

In chapter 8 we explain the motivation for our data-representation. We explain an already implemented module for the numerical computation of universal ventricular coordinates, enabling a mapping from a template heart geometry (achieved through the pre-processing) to a cylindrical representation. This representation is visually close to the heart representation chosen by the American Heart Association. Following this mapping to a cylinder we begin with the actual implementations of my work. These include the parallel computation of a signed distance function, bridging C++ code and python via the c-types library. For completeness we provide short summaries of possible algorithms, but in the end use the already implemented components of the VTK-library. Following this, we attempt to fit an auto-encoder to this data representation. We see that model fitting to more than two data samples is not possible and provide as possible explanation the curse of dimensionality. We then reduce the dimensionality by exploiting the correlation of close signed distance values and the continuity of the signed distance function through mapping to basis spline coefficients. A detailed explanation of the underlying mathematics is provided. We then show that an auto-encoder can be fitted to the whole data-set. This concludes the part on the chosen data representation.

In the chapter 9 we first motivate our choice of algorithm and then examine the generative capabilities of principal component analysis and a variational auto-encoder. We show that neither principal component analysis nor variational auto-encoders create satisfactory scars. We provide a possible explanation by considering the extreme case of only two data samples and a one dimensional latent space, leading to the hypothesis that lacking shape overlap prevents plausible generations when using a gaussian latent distribution as prior. We align the scars before computing the signed distance function, essentially considering shape and position as independent from each other. Through this alignment the quality of generations is improved significantly. Further research leads us to consider and test the manifold hypothesis of data distributions. We provide an intuition for how such a manifold could look like in our case. We show that training on very few neighboring shapes with similar shape and locations (i.e. with sufficient overlap) enables plausible generations through simple principal component analysis. Furthermore, we show that the variational auto-encoder achieves superior generative quality on the same set of neighboring shapes and hypothesize that it is more robust against remaining

nonlinearities. Taking these insights, we train a variational auto-encoder on three sets of shapes with sufficient overlap. We show that certain visually striking features, which are distinct among the three groupings, are maintained in the generations and propose this mechanism for guiding generations through clustering the training data sets. Through comparison to smaller training data sets it becomes apparent that models trained on smaller groupings create more finer level features, some of which can be attributed to the pre-processing. This mechanism can enable the aimed generation and combination of specific features by training only on shapes which already contain them. Importantly we also mention that intelligent clustering which ensures sufficient overlap can render the need for shape-alignment obsolete. This might be preferred due to the likely correlation between the affected coronary, the location and the shape. Finally we examine the generations in the heart mesh and provide segmentation masks. These display clinically important characteristics, such as growth from endocardium to epicardium. Visual inspection of these masks indicates that the generations are plausible.

In chapter 10 we provide a summary of this work, the main problems we encountered, our solutions and the obtained results. We also outline future directions this work can take.

Chapter 7

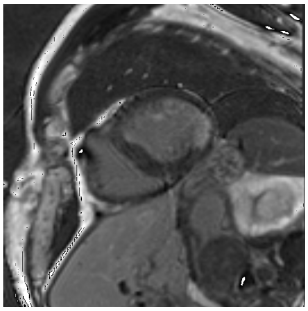
Part 2: Data acquisition

In this chapter the data acquisition and processing is explained. We start with the segmentation of patient data obtained from MRI scans. The contour lines obtained from the segmentation masks are used to fit a heart template geometry via B-spline based mesh morphing. The result is a heart with patient-specific geometry. The segmentation masks enable the assignment of vertex labels, differentiating between healthy myocardium and ischemic scar tissue. Since, for each patient-specific mesh, the number of vertices coincides with a template geometry, the cell values can be mapped back to the template. By representing scars in the template, the patient specific heart is abstracted away. Each case is represented as a scar within the same template heart.

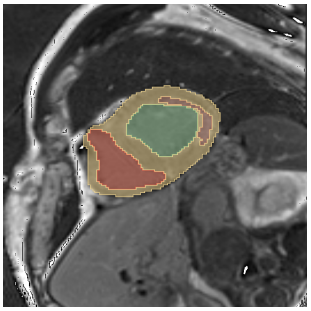
7.1 Segmentation

The patient specific data consists of several (usually ca. 5-10) horizontal slices of MRI images. Figure 7.1a shows such a MRI slice. The segmentation-masks are obtained by manual segmentation. An example is displayed in figure 7.1b. The segmentation masks differentiate between healthy myocardium (yellow), left-ventricle (green), right ventricle (red) and ischemic scar tissue (brown). The contour lines obtained from these masks on different vertical levels are displayed in figure 7.1c. Due to the low amount of MRI slices, the contour lines of the corresponding heart geometry are determined at a relatively low vertical resolution. The contour corresponding to the healthy myocardium mask outlines the outer heart geometry and corresponds to the endocardium, the contours of left and right ventricle masks correspond to the epicardium. These three contours define the patient-specific heart-wall geometry at respective heights.

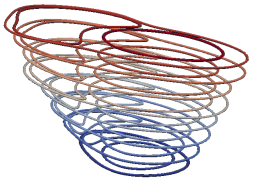
Figure 7.1: Fitting the left ventricle



(a) MRI scan



(b) MRI scar with masks



(c) Contour lines

7.2 Fitting the patient heart

The contour lines (see section 7.1), are used to obtain a 3D mesh of the patient specific heart with nonrigid image registration methods [67]. An available template geometry is fitted via free-form deformation, so that the epicardium and endocardium coincide with the segmentation contours. The template heart is visualized in figure 7.2a. Outcomes at specific time-steps of the morphing are visualized in figure 7.2b and 7.2c respectively. The algorithm (a free form deformation) is explained with detail in [67]. In the following we provide a high-level description. The method utilizes basis splines and an energy (or cost) function, which balances the mesh-curvature and line fitting to avoid physically unrealistic gradients in the heart walls.

Consider a three-dimensional domain (the image volume)

$$\Omega = \{(x, y, z) | 0 \leq x < X, 0 \leq y < Y, 0 \leq z < Z\} \tag{7.1}$$

with volume $V = XYZ$ within which the contours are located. Let $\Phi_{i,j,k}$ be a set of $n = n_x n_y n_z$ control points, where n_x, n_y, n_z denote the number of control points in the x, y, z direction respectively. These control points correspond to mesh vertices. Let $T(x, y, z)$ represent the contour lines and consider the B-spline

$$\tilde{T}(x, y, z) = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} B_l(u) B_m(v) B_n(w) \Phi_{i+l, j+m, k+n}$$

with $i = \lfloor x/n_x \rfloor - 1$, $j = \lfloor y/n_y \rfloor - 1$, $k = \lfloor z/n_z \rfloor - 1$, and $u = \lfloor x/n_x \rfloor - x/n_x$, $v = \lfloor y/n_y \rfloor - y/n_y$, $w = \lfloor z/n_z \rfloor - z/n_z$. For more details the reader is referred to [67].

The function B_i represents the i-th basis function of the B-splines, which in total are

$$B_0(u) = (1 - u)^3/6$$

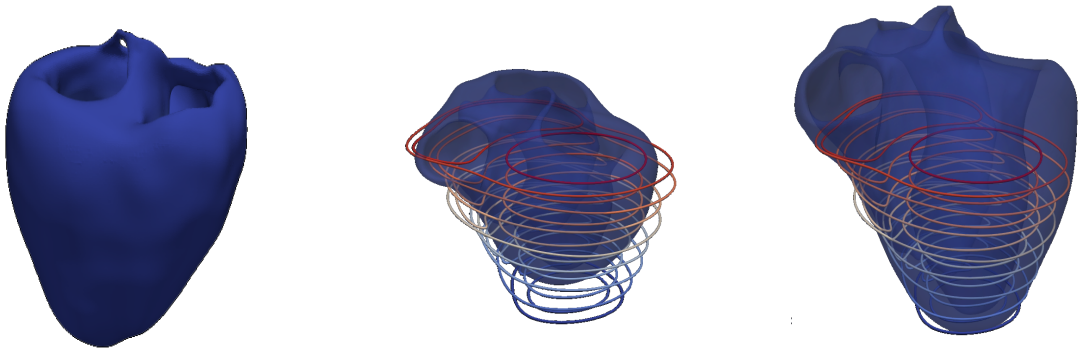
$$B_1(u) = (3u^3 - 6u^2 + 4)/6$$

$$B_2(u) = (-3u^3 + 3u^2 + 3u + 1)/6$$

$$B_3(u) = u^3/6$$

The number of control points of n_x,n_y,n_z are hyper-parameters of the B-spline transformation and correspond to the resolution of the control point mesh Φ. Choosing the right amount of control points is a design choice which balances the trade-off between the computational

Figure 7.2: Fitting the left ventricle



(a) Template heart (b) Early timestep (c) Late timestep

complexity and the accuracy of the approximation. A multi-resolution approach is used to compute $\tilde{T}(x, y, z)$ as linear combination of B-splines $\tilde{T}_r$ with respective resolution (i.e number of control points $n_r = n_{x,r} n_{y,r} n_{z,r}$), via

$$\tilde{T}(x, y, z) = \sum_{r=0}^{L} \tilde{T}_r(x, y, z).$$

This enables a separation of the coarser deformations from the finer high-resolution deformations and results in a more computationally efficient approach. The low-resolution forms are approximated at lower computational cost. The resolution is incremented in an iterative manner to approximate regions which require high resolution. This hierarchical approach enables much faster convergence, since low-resolution details do not need to be approximated with a high resolution mesh. To penalize smoothness of the mesh-deformation a penalty term C is computed:

$$C_{\text{smooth}} = \frac{1}{V} \int_o^X \int_0^Y \int_0^Z [\frac{\partial^2 \tilde{T}}{\partial^2 x^2} + \frac{\partial^2 \tilde{T}}{\partial y^2} + \frac{\partial^2 \tilde{T}}{\partial z^2}] dx dy dz.$$

The penalty-term for line fitting is computed as

$$C_{\text{fit}} = \frac{1}{V} (T(x, y, z) - \tilde{T}(x, y, z))^2$$

for all $(x, z, z) \in N$ and zero elsewhere. Both together form the optimization loss:

$$C = C_{\text{fit}} + C_{\text{smooth}}$$

The algorithm which fits the mesh to the contour lines can be described by the following Pseudocode

procedure FIT MESH TO CONTOURS
  initialize control points $\Phi_{r=0}$ by minimizing C_{fit}
  repeat
    $\nabla C \leftarrow \frac{\partial C(\Phi_r)}{\partial \Phi_r}$
    while $||\nabla C|| > \epsilon$ **do**
      $\Phi_r \leftarrow \Phi_r + \mu \frac{\nabla C}{||\nabla C||}$
      $\nabla C \leftarrow \frac{\partial C(\Phi_r)}{\partial \Phi_r}$
    end while
    increase control point resolution Φ_r to Φ_{r+1}
  until finest resolution is reached
end procedure

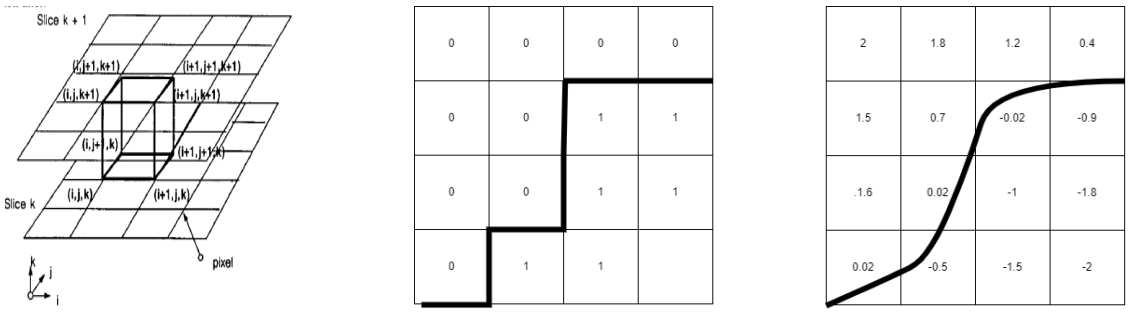
More details and references can be found in [67]. Once the patient specific heart is obtained, its cell values can be filled according to the by assignment to the closest segmentation mask. Cells are labeled as left ventricle myocardium, right ventricle myocardium and ischemic scar tissue.

7.3 Iso-surface extraction

Implicit shape representations define a field of level-set values in 3D-space. The shape boundary is represented by a specific level set. Often the zero level. In a binary representation with values zero and one, the shape boundary is defined by the transition. In the continuous case, the cell values often correspond to the distance to a surface. The binary values (healthy myocardium and ischemic scar tissue) obtained from the patient-data (see section 7.2) define an implicit binary shape representation (right ventricle, left ventricle, ischemic scar). For representations of this type, the marching cubes algorithm [45] can be used for surfaces extraction. Here we present a detailed explanation of the algorithm, encompassing both binary and continuous scalar

fields. Figure 7.3a displays the marching cube which, as the name suggests, is translated over the domain. Figure 7.3b displays a discrete field with respective iso-surface. Figure 7.3c displays a continuous field with respective iso-surface. The marching cube algorithm can be summarized in the following four bullet points:

Figure 7.3: Marching cubes



(a) the cube (extracted from [45])

(b) a discrete field

(c) a continuous field

1. **Cube Configuration:** The scalar field is divided into cubes formed by eight neighboring voxels. For each cube, an 8-bit index is generated. In a binary field, the value at each vertex is either zero or one. In a continuous field, the value can vary, with negative values typically indicating inside the object and positive values indicating outside.

   For both cases, the index is computed by treating each vertex value as a bit in an 8-bit integer, where the bit is set to one if the value exceeds the threshold (commonly zero) and zero else. This index determines which of the 256 possible cube configurations applies.

   Figure 7.4b displays a cube with its 8-bit index.

2. **Lookup Table:** A precomputed lookup table maps each index to a set of edges, indicating where the surface intersects the cube. This table is essential for efficiently determining the surface geometry without recalculating the intersections for each cube configuration from scratch. Each 8-bit index points to its respective surface configuration. Figure 7.4a displays some of the possible surface configurations.

3. **Edge Interpolation:** The precise position where the surface intersects an edge of the cube is determined by linear interpolation of the scalar values at the edge's vertices.

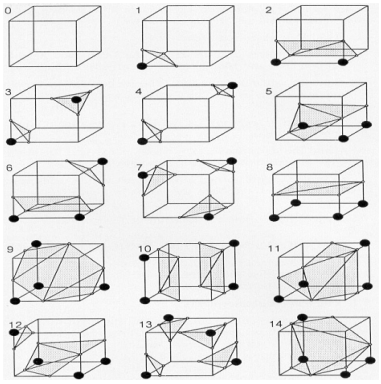
   - In the **binary case**, the scalar values are 0 or 1 and the intersection is typically approximated at the midpoint of the edge whose vertices have values zero and one.
   - In the **continuous case** (e.g., signed distance function), the intersection can be determined more accurately through interpolation

   $$p = \frac{|f(v_1)|}{|f(v_1)| + |f(v_2)|} \times v_2 + \frac{|f(v_2)|}{|f(v_1)| + |f(v_2)|} \times v_1$$

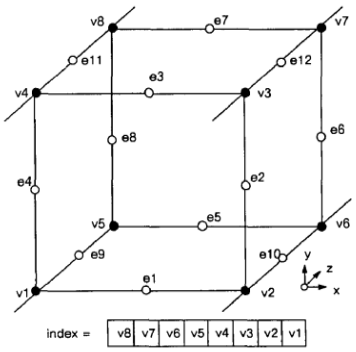
   where $f(v_1)$ and $f(v_2)$ are the scalar values at the vertices v_1 and v_2. This interpolation enables the extraction of a smooth surface even when the underlying grid is relatively coarse.

4. **Surface Construction:** Using the computed intersection points, the algorithm constructs surface triangles within the current cube. These triangles are generated according to the lookup table. This specifies the mesh connectivity.

Figure 7.4: Look-up table and 8-bit index (extracted from [45])



(a) Look-up table



(b) 8-bit index for a cube

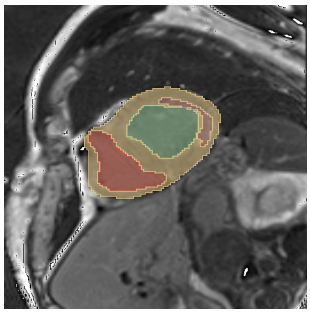
The marching cubes algorithm is versatile and can be applied to a variety of domains. In medical imaging it is used to reconstruct surfaces of organs from volumetric scans. In computer graphics, it is employed to generate meshes from defined scalar fields. The choice between binary and continuous scalar fields depends on the application's specific needs for surface detail and smoothness. In this work, marching cubes is applied to the binary volumetric field obtained during the data acquisition and to a continuous signed distance function which is used during the generative process.

7.4 Acquisition Results and Processing Artifacts

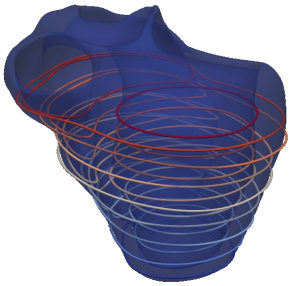
A disadvantage of our processing methodology is that the vertical resolution of the MRI scans and consequently also the segmentation masks is low. Only approximately ten horizontal slices exist for each heart. The current approach identifies the nearest mask for each cell and assigns corresponding values. The result are nonphysical processing artifacts. For example, the cell-values have a block like structure and sometimes the extracted scar displays seemingly separated "blocks". In reality, the tissue structure is most likely smooth and connected. Furthermore, MRI scans are usually not available at the base. Scar tissue there is not captured, leading to nonphysical holes. In figure 7.5c the disjoint, block like, structure of the obtained scar is visualized. Figure 7.5c also displays seemingly disconnected tissue structures. Furthermore, it does not display any scar tissue at the base. However, it is likely that the scar tissue is connected and also covers the base. Another example of such a case is visualized in figure 2.4c. In section 8, which describes the scar representation in a disk domain, the missing data at the heart base is visible as holes in the scar.

For the purpose of this work, the obtained scars, including processing artifacts, will be considered ground-truth. This needs to be considered when evaluating the representative power of the data.

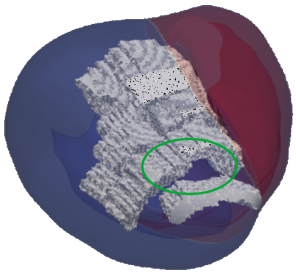
Figure 7.5: Assignment of cell values based on the Segmentation Mask



(a) segmentation mask with scar mask



(b) Heart with contour lines



(c) Obtained Scar (view from below)

Chapter 8

Part 2: Data Representation

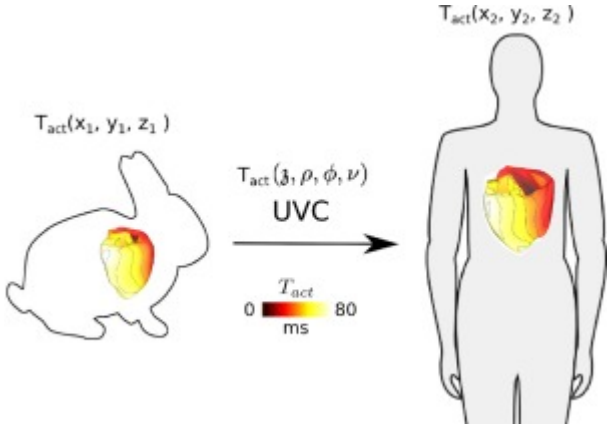
A first step in designing machine learning applications is choosing a data-representation and algorithm. Shape analysis and generation is an extensively studied field in medical imaging technology. The main models are statistical shape models [10] and their non-linear alternative, Gaussian morphable models [46]. Both require point-to-point correspondence between shapes. This correspondence can be achieved by manual landmark annotation, non-rigid image registration or both. Good libraries exits which implement many methodologies for shape modeling and generation [10] [26]. A major limiting factor of these models is that the shapes must have equal topology. While this is plausible for many organic geometries, such as mammalian hearts, many bone-structures, kidneys and more, our data-samples contain topologically differing shapes. It is unclear if this dissimilarity can be attributed to the processing (and thus be eliminated by additional processing steps) or if it is inherent in the data itself. Thus we consider the extracted scars, including processing artifacts, as ground-truth and attempt to generate shapes while allowing for differing topologies. This makes standard statistical shape models unsuited for our problem. We consider a shape representation as signed distance function in a compact domain. For the mesh vertices on which the signed distance is defined we achieve point-to-point correspondence. We then apply algorithms such as principal component analysis, gaussian process latent variable models, gaussian morphable models and variational auto-encoders to the vertex values. However, this corresponds to the modeling of continuous signed-distance functions and not necessarily the implicit shape.

In the section 8.1 and 8.2 we present our data representation. Section 8.3 elucidates how the high-dimensional feature vector prevents model fitting and how basis-spline coefficients can be used as a low-dimensional features.

8.1 Universal ventricular coordinates

In clinical practice the left ventricle of the heart is represented by a Standardized Myocardial Segmentation and Nomenclature provided by the American Heart Association. This representation flattens the heart into a disk, with the apex at the center and the base on the border. 17 regions are distinguished. These regions enable a standardized communication regarding locations in the heart. For example, clinicians define scar-shapes by the degree of occupancy and transmurality within each standardized region. To approach the clinical setting, we choose to map the left ventricle to a cylinder. The height is representative of the wall-thickness at respective positions. The mapping into the disk requires the usage of standardized ventricular coordinates. Each point in the template heart is assigned four coordinates, ρ, Φ, θ and γ. These four coordinates (also called universal ventricular coordinates) enable the representation in a cylinder.

Figure 8.1: Mammalian hearts in universal coordinates (extracted from [7])



In this section the procedure for mapping sets of cardiac ventricles to topologically equivalent representations is presented. This mapping is used to compare quantities of interest between different heart geometries and presented in [7]. It is based on the knowledge that mammalian hearts share a common structure with generally four-chambers and two ventricles. Figure 8.1 visualizes a mapping from a rabbit heart to a human heart via universal ventricular coordinates. Describing tensors on a generic heart is important for comparison and quantification across geometries. For example, mammalian hearts express electrical heterogeneity with respect to trans-mural, apicobasal and left-right gradients [7]. The segmentation of AHA-regions already describes various heart geometries in a generalized framework. However, [7] highlights the low-resolution of this mapping and provides a more detailed way of computing a generalized representation of mammalian hearts. Universal ventricular coordinates are presented. The four coordinates are

- ρ, which represents the distance between the endocardium and epicardium, i.e. the transmurality.
- Φ, which represents the circumferential distance from the long axis of right ventricle and left ventricle respectively.
- r, which corresponds to the distance traveled along the long axis of the ventricles from apex to base, i.e. the apicobasal coordinate.
- γ, which is used to distinguish between right and left ventricle.

The computation of these coordinates is explained in [7]. We provide a short summary. Given the surfaces for the heart base, epicardium, endocardium of both ventricles and septum, the universal ventricular coordinates can be obtained by solving the Laplace equations with Dirichlet boundary conditions applied onto each surface.

- For the trans-mural coordinate, the epicardium is assigned $\rho = 1$, the endocardium is assigned $\rho = 0$ (see figure 8.2).
- For the apicobasal coordinate, the base is assigned $r = 0$ and the apex is assigned $r = 1$ (see figure 8.3).
- For the rotational coordinate, the left ventricle endocardium is assigned $\Phi = \pm\pi$ at the wall where it combines with the right ventricle. Furthermore, $\Phi = \pm\frac{\pi}{2}$ at the outer junctions of left and right ventricle, $\Phi = 0$ at the middle of the septum and $\Phi = \pm\frac{\pi}{2.5}$ at the inner junctions of left and right ventricle (see figure 8.4).

Figure 8.2: Boundary conditions for the transmural coordinate (extracted from [7])

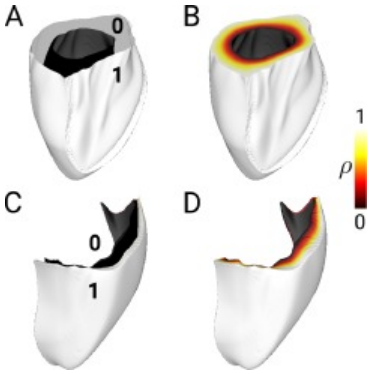
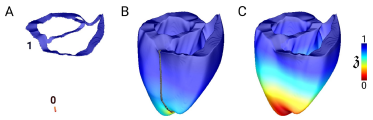


Figure 8.3: Boundary conditions for the apicobasal coordinate (extracted from [7])



- To seperate left and right ventricle, the left ventricle endocardium gets $\gamma = 0$, the right ventricle endocardium gets $\gamma = 1$, the septum gets $\gamma = 0.5$ (see figure 8.5).

The Laplace equation is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

where

$$y_b(x) = f(x) \quad \forall x \in \partial\Omega$$

correspond to the boundary conditions on the boundary $\partial\Omega$.

Computing the coordinates corresponds to finding a solution f, so that the specified boundary conditions are satisfied. Since the Laplace equations appear in heat-flow one can formulate the problem as fixing a temperature at the specified boundaries and computing heat flow until a stationary temperature distribution is achieved. The stationary distribution corresponds to the universal ventricular coordinate. The heat flow is computed numerically on a discretization of the domain, e.g. by the finite element method. We will not go deeper into the numerical solving procedure and point to [7] for additional information.

Figure 8.4: Boundary conditions for the rotational coordinate (extracted from [7])

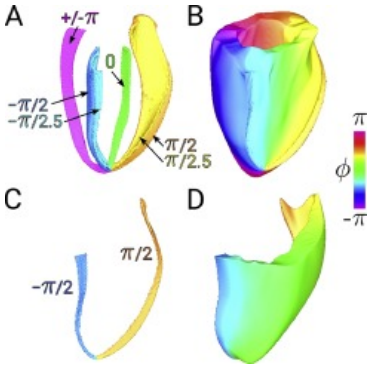
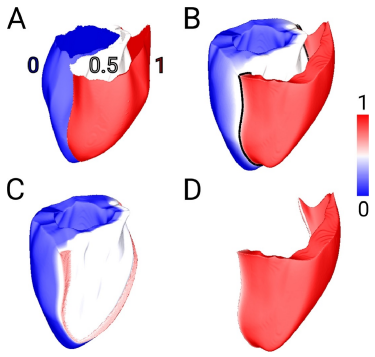


Figure 8.5: Boundary conditions for the left and right ventricle (extracted from [7])



Once the universal ventricular coordinates are computed, each point of the left ventricle is mapped to a cylinder with coordinates r for the radius, ρ for the height and Φ for the angular coordinate. The end-result is a cylindrical mesh with the same connectivity information as the heart mesh.

Figure 8.6: Representation as heart

Figure 8.7: Representation as disk

8.2 Signed distance representation

Within the disk-mesh, a signed distance function is computed. Each node in the disk-mesh is labeled with its minimal distance to the scar boundary. The signed distance function is described by a function $d(p, S)$, where p is a point in (usually three dimensional) space. This function is defined over the whole domain. Its values are so that $d(p, S) < 0$ if p is inside the shape and $d(p, S) > 0$ if p is outside the shape. Values of p which are positioned exactly on the surface boundary correspond to $d(p, S) = 0$. The signed distance function requires the shape-boundary to be closed, allowing for a distinction between inner and outer space. The shape is represented only implicitly in form of a function. Recovering the shape-surface requires usage of other algorithms. We use marching cubes (see section 7.3) to extract surface information.

Computing the signed distance function requires two objects. A set of points in space, as well as surface-information of the shape boundary. These objects can be used to assign to each point the distance to the surface. In our case, the shape-information is in the vertex values of the heart-mesh. Some values are assigned to the healthy myocardium and others to the scar tissue. Similar to a signed distance function the representation is implicit, however in a discrete manner. By using the *Extract-Surface* function (which implements marching cubes) provided by *paraview* and *vtk* we convert the implicit scar representation into a surface-mesh representation. We then compute the signed distance information for each points within the disk. The relevant algorithms perform a computation of Point-Plane distances and a Boolean function which differentiates between inner and outer points. The computation of the distance and the boolean function can be separated. One employs a point-plane distance algorithm. The other uses the winding-number or ray-casting algorithm.

8.2.1 Point-Plane Distance algorithm

Let

$$P_i = (x_i, y_i, z_i),\ i = 0, 1, ...N-1$$

be a set of N three-dimensional coordinates corresponding to a set of discrete points. Let

$$\vec{a_j} = (x_a, y_a, z_a)_j,$$

$$\vec{b_j} = (x_b, y_b, z_b)_j,$$

$$\vec{c_j} = (x_c, y_c, z_c)_j$$

with

$$j = 0, 1, ...M-1$$

be a set of vertices corresponding to a cell of a surface mesh with M triangular faces. The normalized normal vector of the plane can be computed as

$$\vec{n} = \frac{(a_j \times b_j)(a_j \times c_j)}{||(a_j \times b_j)(a_j \times v_j)||},$$

where $\times$ denotes the cross product. To find the projection of a generic $3D$ point

$$\vec{p} = (x_p, y_p, z_p)$$

onto the plane consider the decomposition

$$\vec{p} = \alpha\vec{n} + \beta_1\vec{n}_1^{\perp} + \beta_2\vec{n}_2^{\perp},$$

where $\alpha\vec{n}$ corresponds to the distance of $\vec{p}$ to the plane, $\beta_1\vec{n}_1^{\perp} + \beta_2\vec{n}_2^{\perp}$ corresponds to the coordinates of the projection and $\vec{n}_1^{\perp}$, $\vec{n}_2^{\perp}$ are mutually orthogonal vectors of the plane. Define the plane equation

$$\vec{n} \cdot \vec{x} + d = 0.$$

Compute d for the plane using any point within the plane (for example $\vec{a_j}$, $\vec{b_j}$ or $\vec{c_j}$) as $\vec{x}$. Then the minimal distance of $\vec{p}$ to this plane is given by

$$\vec{n} \cdot \vec{p} + d = \beta_1\vec{n}_1^{\perp} + \beta_2\vec{n}_2^{\perp}.$$

The projected coordinate can be written as

$$\vec{p}_{proj.} = \alpha\vec{n} = \vec{p} - \beta_1\vec{n}_1^{\perp} - \beta_2\vec{n}_2^{\perp} = \vec{p} - (\vec{n} \cdot \vec{p} + d) = \vec{p} - (\vec{n} \cdot \vec{p} - \vec{n}\vec{a_j}).$$

8.2.2 Winding number algorithm

The winding surface algorithm (see [2]) computes the solid angle to point $\vec{p}$ for every triangle of the surface mesh defined by

$$\vec{a_j} = (x_a, y_a, z_a)_j,$$
$$\vec{b_j} = (x_b, y_b, z_b)_j,$$
$$\vec{c_j} = (x_c, y_c, z_c)_j,$$

and sums them. The solid angle is computed as

$$\Omega_i = 2 \cdot \text{atan2}\left(\vec{n}_i \cdot \vec{ap}_i, \|\vec{ap}_i\|\|\vec{bp}_i\|\|\vec{cp}_i\| + (\vec{bp}_i \cdot \vec{cp}_i)\|\vec{ap}_i\| + (\vec{cp}_i \cdot \vec{ap})_i\|\vec{bp}_i\| + (\vec{ap}_i \cdot \vec{bp}_i)\|\vec{cp}_i\|\right),$$

where $\vec{ap}_i$,$\vec{bp}_i$,$\vec{cp}_i$ respectively correspond to the distance between $\vec{a}_i$, $\vec{b}_i$, $\vec{c}_i$ and $\vec{p}_{proj.}$. The vector $\vec{n}$ corresponds to the normal vector of the triangle and is computed via

$$\vec{n} = \vec{ab} \times \vec{ac}.$$

The winding number for point $\vec{p}$ is computed as the sum of all solid angles and is

$$w = \sum_{i=0}^{T} \Omega_i,$$

where T is the number of triangles within the surface mesh. If $\vec{p}_{proj.}$ is within the surface, then $w \approx 4\pi$, else $w \approx 0$. The winding number algorithm is quite robust but computationally expensive, although it can be parallelized heavily. For complex meshed, with the possibility of self-intersections this algorithm is recommended.

8.2.3 Ray Casting Algorithm

A method to determine whether $\vec{p}$ is inside or outside the surface mesh is via a ray-casting algorithm (see [62]). This method is less robust than the winding-number algorithm, but often good enough, easier to implement and requires less computation. Consider the point $\vec{p}$ and define a ray

$$\vec{y}_{ray} = \vec{p} + \gamma\vec{r}.$$

Consider the three vertices $\vec{a_j}, \vec{b_j}, \vec{c_j}$ of triangle j and the three corresponding edges

$$\vec{e_1} = \vec{a_j} - \vec{b_j},$$
$$\vec{e_2} = \vec{a_j} - \vec{c_j}$$

and

$$\vec{e_3} = \vec{b_j} - \vec{c_j}.$$

To find the intersections of the ray with each edge by solving the linear system of equations

$$\vec{y}_{\text{ray}} = \vec{p} + \gamma\vec{r} = \vec{a}_j + u\vec{e}_1 + v\vec{e}_2, \quad \text{where } u \geq 0,\ v \geq 0,\ u + v \leq 1.$$

One can solve for u,v and γ via linear programming. If γ is positive and u, v satisfy $u \geq 0$, $v \geq 0$, $u + v \leq 1$, then the ray intersects the triangle. If the number of intersections is odd, then the $\vec{p}_{proj.}$ is not within the triangle.

8.2.4 Parallel computation in VTK

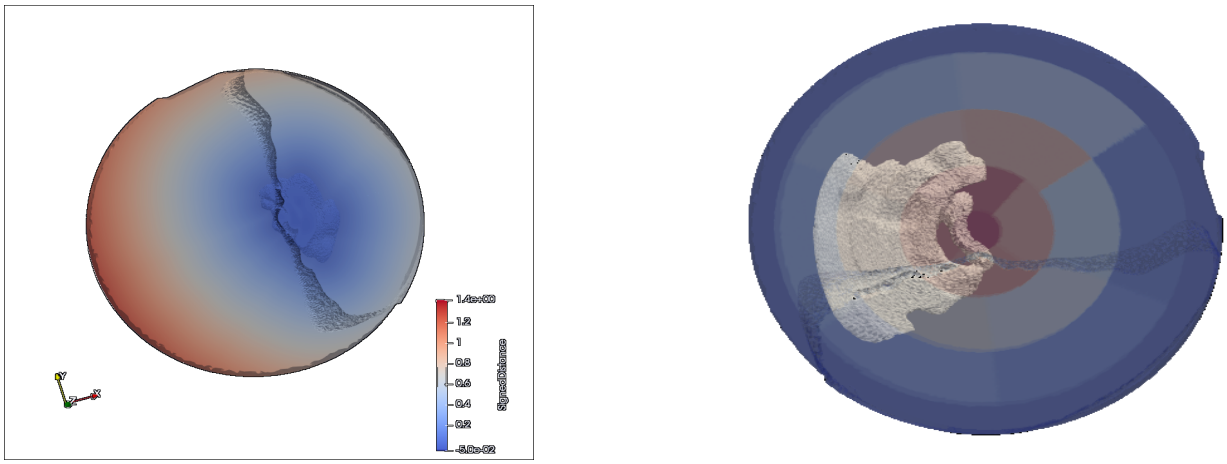
The *vtk*-library provides a *FindClosestPoint()* function. This function accepts a surface-mesh and a $3D$ coordinate as input. It returns the *closest point* on the surface as well as the corresponding distance. Additionally, the *selector*-class has a function *IsInsideSurface()*, which is used to determine the sign of the returned distance. It employs the ray casting algorithm.

Since the disk contains 600000 points, the computational efficiency becomes relevant. Running the *FindClosestPoint()* and *IsInsideSurface()* in a python *for-loop* takes a significant amount of time. It is better to write the code in C++ and connect it to python with the *c-types* library. Using *vtks* parellizability and running the for-loop in several threads with *vtkSMPTools* results in a significant speed-up. Since the *selector*-class with the *IsInsideSurface()* -method is not thread-safe, it must be initialized within the loop for each thread. Contrasting this, the *CellLocator* -class can be initialized only once for each shape due to its thread-safe computation. However, in principle the winding number algorithm and ray casting algorithm are both parallelizable and it is only due to the reliance on the *selector* class that this part must be initialized for each thread. The function we use to compute the signed distance values is

1: **function** ComputeSignedDistance(SignedDistanceVector, pointSet, Point_number, surface, transformed)
2: Initialize `cellLocator`(surface)
3: numberOfPoints ← Get number of points from surface
4: grains ← `Point_number` / 28 + 1
5: vtkSMPTools::**For** (0, Point_number, grains, &(int startPtId, int endPtId))
6: Initalize `selector`(surface)
7: `double`* p - *a pointer to the current point coordinates*
8: `double` closestp[3] - *array to store the closest point's coordinates*
9: `double` dist2 - *distance squared from the current point to the closest point on surface*
10: `vtkIdType` cellID - *identifier for the closest cell*
11: `int` subID - *identifier for the sub-cell of the closest cell*
12: `vtkNew<vtkGenericCell>` genericCellPtr: *pointer to a generic cell object*
13: **For** `int`ptId from startPtId to endPtId:
14: p ← pointSet[ptId] - *Point coordinates at index ptId in* ***`pointSet`***
15: **If** `selector`.IsInsideSurface(p)
16: sign ← -1
17: **else**
18: sign ← 1
19: `cellLocator`.FindClosestPoint(p, closestp, genericCellPtr, cellId, subId, dist2)
20: SignedDistanceVector[ptId] ← sign × $\sqrt{\text{dist2}}$
21: **End parallel For**
22: **end function**

This computation returns a vector with 600000 signed distances, one for each point of the mesh. Figure 8.8 displays an implicit signed distance function and an extracted iso-surface. Recovering the scar-shape involves iso-surface extraction with the marching cubes algorithm (see section 7.3). The iso-surface is defined on the zero level set of the signed distance field. No important features are lost.

Figure 8.8: Zero level set of a signed distance field



(a) a signed distance field

(b) iso-surface on the zero level

8.3 Dimensionality reduction with basis splines

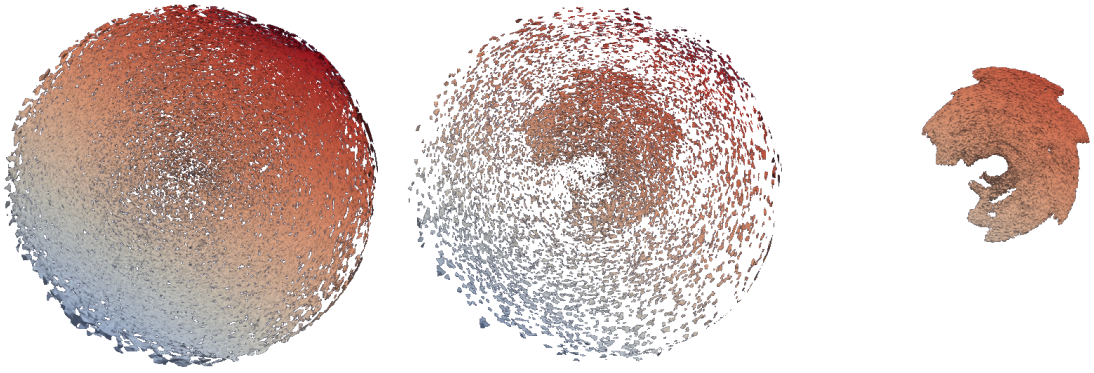
The scar is represented as a signed distance function (SDF) in the $3D$-domain of a disk, i.e. $s = f(x, y, z)$. The signed distance values are available on discrete points as a vector of ca. 600000 signed distance values. Initial tests with an (non-variational) auto-encoder show that this high dimensionality prevents model-fitting. We are unable to fit our model to more than a single data-sample. Figure 8.9 displays the training progress on a single shape. Only points with negative values are shown. Each of the 600000 discrete points is updated individually. This independent consideration of each pixel leads to a very high-dimensional optimization problem. However, it is a-priori known that the signed distance function is smooth and that close pixel values are highly correlated. We use this prior knowledge and represent each function as basis-spline coefficients. The curse of dimensionality is related to optimization with many independent

Figure 8.9: Over-fitting a variational auto-encoder to the high dimensional signed distance

(a) Epoch 0

(b) Epoch 2000

(c) Epoch 5000

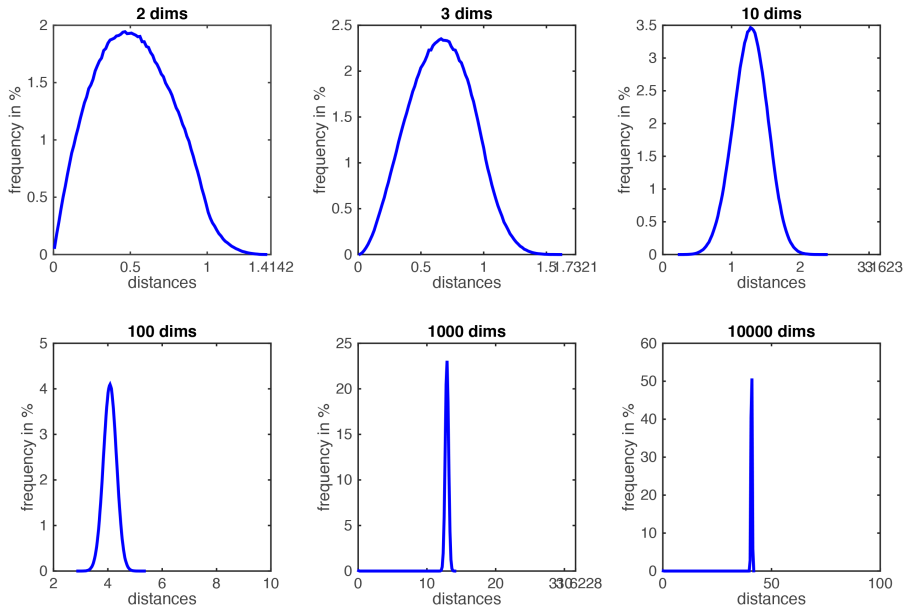


components. We refer the reader to [17] for an insightful perspective on high dimensional optimization. An exponential scaling law relates training error and feature-dimensionality, indicating that high dimensional feature-vectors makes convergence difficult. Optimization of our auto-encoder model occurs via the Euclidean distance between input and output pairs. Consider

$$\text{dist}(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_d - y_d)^2}.$$

When sampling points uniformly random from the d-dimensional space, the distance distribution tends to become concentrated around a mean value. This effect is called concentration of measure [17]. Figure 8.10 displays the euclidean distance between points drawn uniformly from space. With increasing dimensions, the distances converge to higher and more similar values. For training on a single data-sample this is not problematic since input and output

Figure 8.10: Concentration of measure in high dimensions



https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html

are fixed and there is no need to distinguish between pairs. As soon as two data-samples are considered, this can make training difficult. Finding an optimal separating hyperplane between high dimensional samples is difficult considering that nearly all points have nearly equal distance.

The signed distance function is a continuous function. This continuous function can be represented as a linear combination of weighted basis functions. We call this methodology of representing lines, surfaces, or $3D$-functions, in terms of continuous basis function coefficients, a B-spline representation. The continuous function is mapped to few coefficients. Through this representation we alleviate the problem of high-dimensional feature vectors and incorporate prior-knowledge of function smoothness.

By mapping the function to coefficients of basis splines, the dimensionality is reduced hundredfold. Figure 8.15, displays the fitting process on the basis-spline representation. It becomes possible to fit an auto-encoder to all 44 data-samples.

8.3.1 Bezier functions

We explain Bezier curves and build up to Basis-splines. Bezier-curves can be used to represent functions as linear combination of Bernstein polynomials. Consider a $1D$-function and d control points c_i with $i = 0, ..., d-1$. Consider the parametrization of the Bernstein polynomials, with $u \in [0, 1]$ and degree d,

$$B_{i,d}(u) = \binom{d}{i}(1-u)^{d-i}u^i.$$

Often we will omit the degree index d. If $d = 4$, then the cubic bernstein polynomials are defined as

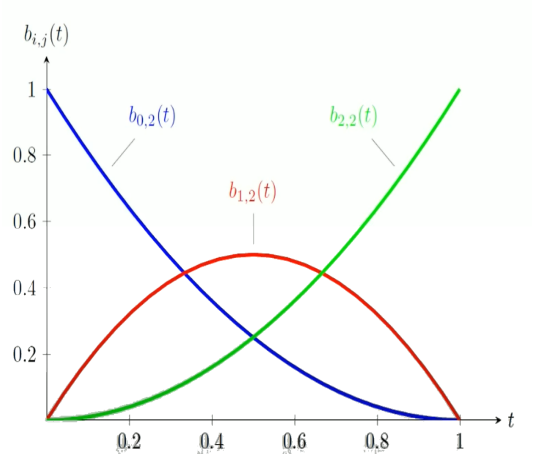
$$B_0(u) = (1-u)^3,$$
$$B_3(u) = 3u(1-u)^2,$$
$$B_2(u) = 3u^2(1-u),$$

and

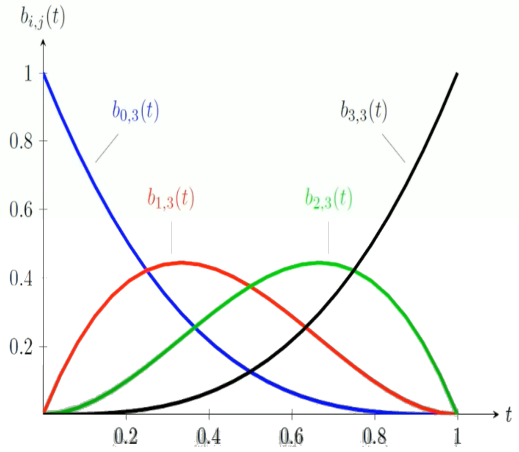
$$B_3(u) = u^3.$$

Figure 8.11a and 8.11b display quadratic and cubic bernstein polynomials respectively. The construction of a degree d bezier-curve requires $d+1$ control points. Note, that, for cubic

Figure 8.11: Bernstein polynomials



(a) quadratic Bernstein polynomials



(b) cubic Bernstein polynomials

https://www.youtube.com/watch?v=qhQrRCJ-mVg

Bernstein polynomials, we can write

$$B(u) = [B_0, ..., B_3](u) = [u^3, u^2, u, 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = \vec{u}^T \tilde{M}.$$

The Bernstein polynomials are used to define a Bezier-curve as

$$\mathbf{f}(u) = \sum_{i=0}^{d} c_i \, B_{i,d}(u)$$

which can be written in matrix form

$$f(u) = B\vec{c} = \vec{u}^T M \vec{c},$$

where $\vec{c} = [c_0, c_1, ..., c_d]^T$, $\vec{u}^T = [u^d, u^{d-1}, ..., 1]$ and M is a matrix of scalars. The index d corresponds to the number of bernstein-polynomials (and is thus related to their degree). There are as many bernstein polynomials as there are control points c_i. Furthermore, each bernstein polynomial influences all control points. For most practical purposed $d = 3$ is sufficient. A cubic Bezier curve can also be written as

$$f(u) = (1-u)^3 c_0 - 3t(1-t)^2 c_1 + 3t^2(1-t)c_2 + t^3 c_3.$$

Assume there is a function $r(u)$ which is provided as vector of discrete values at n points, i.e. $r \in \mathbb{R}^n$. This function can be approximated as a Bezier-curve by determining the coefficients $c = [c_0, c_1, ..., c_d]^T$ which minimize the mean squared error

$$E(c) = \left(r(u) - \sum_{i=0}^{d} c_i B_{i,d}(u) \right)^2 \approx \sum_{j=0}^{n} \left(r[j] - \sum_{i=0}^{d} c_i B_{i,d}[j] \right)^2 \tag{8.1}$$

or in matrix notation

$$E(c) = \left(\vec{r} - B\vec{c} \right)^2 .$$

Similarly, the same concept can be applied to $2D$-surfaces. Herefore, consider m_x control points in x-directions, m_y control points in y-direction and a total of $m = m_x m_i$ control points. Then,

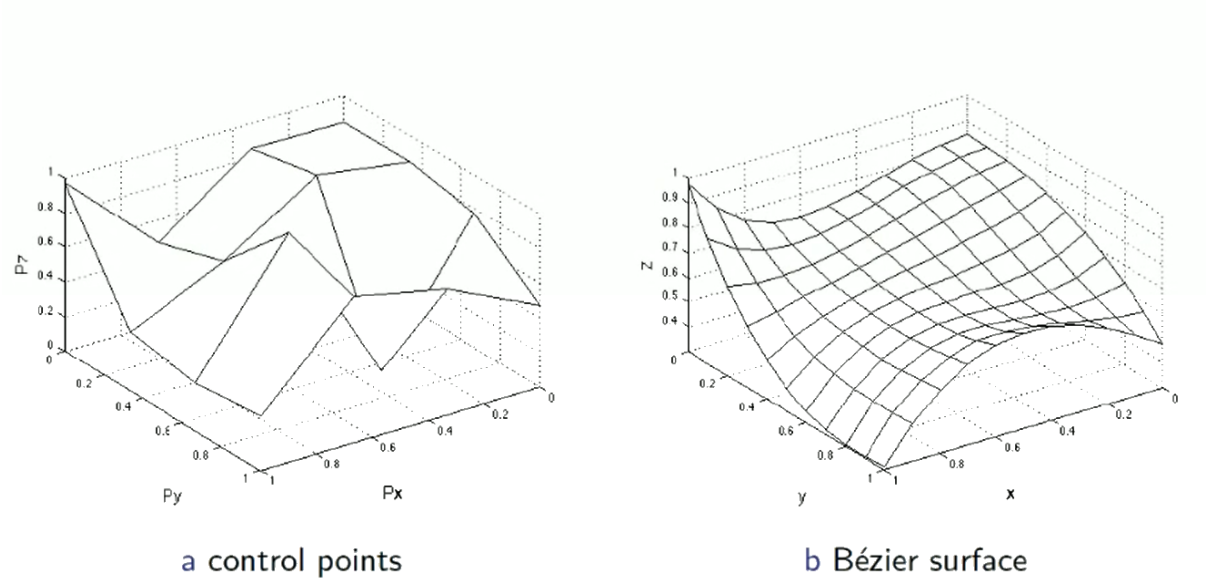
$$B_i(u) = B_i^x(u) B_i^y(u)$$

and the Bezier-curve is defined analogously

$$f(u) = \sum_{i=0}^{m-1} c_i B_i(u) = B\vec{c}.$$

Figure 8.12 displays control-points and a constructed $2D$-surface respectively.

Figure 8.12: A Bezier surface



https://www.youtube.com/watch?v=qhQrRCJ-mVg

8.3.2 Basis splines

The Basis-spline (short B-spline) approach approximates functions by the linear combination of section-wise defined bezier curves. Each bezier curve is restricted to a finite range of influence. This has the advantage of increased control and removes the dependency between basis function degree and number of control points. However, this section-wise construction introduces continuity issues and requires a specific, recursive definition.

A d degree B-spline defined by n control points will consist of $n - d$ basis functions. For

example, a cubic B-spline defined by 6 control points $P_0, ..., P_6$ consists of 3 Bezier curves. The support of each basis-function is defined through a knot vector $k \in n + d + 1$. The knot vector contains values in the range $[u_{min}, u_{max}]$ in a non-decreasing manner. Typically, as in section 8.3.1, $u_{min} = 0$ and $u_{max} = 1$, but other ranges are also possible. The first and last element have multiplicity $d + 1$, meaning, that the value u_{min} is repeated in index 0 to $d - 1$ and u_{max} in index $n + d + 1$ to $n + 1$. All other values are chosen with uniform intervals and multiplicity one. Uniform intervals are not strictly necessary but make application easier. The knot vector $k = [u_0, u_1, ...u_{(n+d+1)}]$ determines the local support of each basis function B_i with $i \in [0, .., n - 1]$. The knot-vector helps define a special set of basis functions which fulfill continuity conditions. These functions are calculated recursively via the Cox-de-Boor formula. These basis-functions are non-negative and have local support.
One starts with base case (degree 0) functions defined as

$$B_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

and then recursively constructs higher degree functions via

$$B_{i,d}(u) = \frac{u - u_i}{u_{i+d} - u_i} N_{i,d-1}(u) + \frac{u_{i+d+1} - u}{u_{i+d+1} - u_{i+1}} N_{i+1,d-1}(u),$$

where d denotes the degree. Note, that if $u \notin [u_i, u_{i+1}]$, then $N_{i,0}(u) = 0$. This ensures that the basis-function of the i-th control point only has local support. Through this, the knot-vector influences the support region of each basis function $B_{i,d}(u)$. At the boundaries, values are repeated for continuity. Figure 8.13 displays a set of basis-functions with local support and their knot values. For each control point we obtain one Bezier-function, i.e. $i \in [0, m - 1]$ The B-spline is defined as

$$f(u) = \sum_{i=0}^{d} c_i B_{i,d}(u) = Bc$$

A function $r(u)$ which is represented as vector of discrete values at n points i.e, $\vec{r} \in \mathbb{R}^n$ can be approximated by determining the coefficients $c = [c_0, c_1, ..., c_d]^T$ which minimize the mean squared error

$$E(c) = \left(r(u) - \sum_{i=0}^{d} c_i B_{i,d}(u) \right)^2 = (\vec{r} - B\vec{c})^2.$$

B-splines allow for independent degree d and control point number n. The local support property enables easier optimization and introduces a sparse structure into the B-matrix, which can be exploited for computational efficiency.

8.3.3 Application to the signed distance function

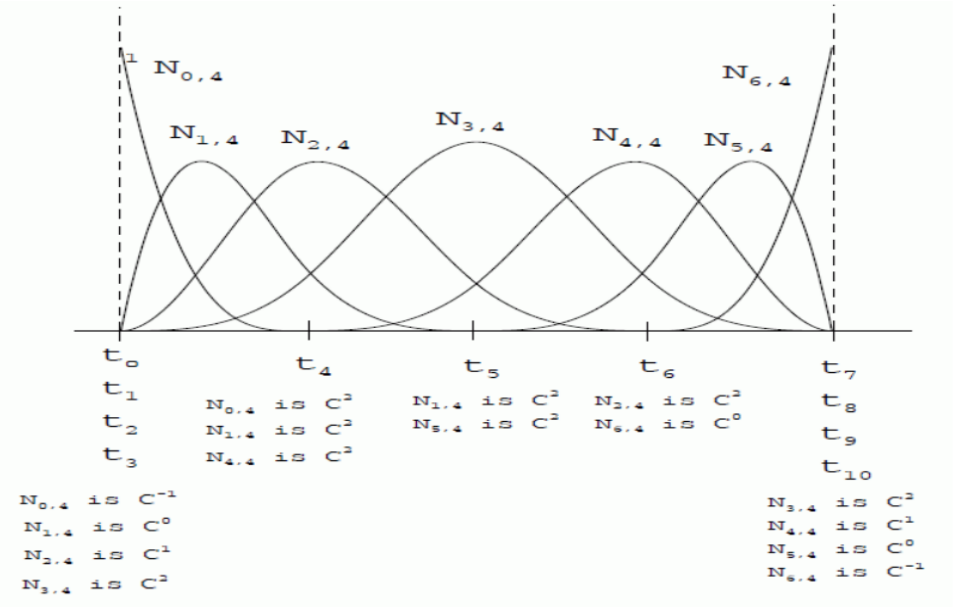
We apply the Basis-spline approximation to a $3D$ signed distance function by defining multiple $1D$ basis functions. We choose the l.i. coordinate lines r, θ, z in cylindrical coordinates as basis directions and define m_r, m_θ, m_z control-points in each direction. The $3D$-domain is discretized into $m = m_r m_\theta m_z$ control points whose coefficients define the m-dimensional vector $\vec{c}$. The Basis-function in $3D$ is defined as the product of the $1D$ Basis-functions functions, which are each constructed via the Cox-de-Boor recursion (for brevity the index d for degree is omitted), to

$$B_i(u) = B_i^r(u) B_i^\theta(u) B_i^z(u).$$

The B-spline curve is defined as linear combination

$$f(u) = \sum_{i=0}^{m-1} c_i B_i(u) = B\vec{c}.$$

Figure 8.13: Basis functions and knot values



https://www.youtube.com/watch?v=qhQrRCJ-mVg

B has dimension $m \times n$, where $B_{i,j} = B_i(r_j, \theta_j, z_j)$.

Given the n-dimensional signed distance vector s, the mean squared error can be constructed

$$E(c) = (\vec{s} - B\vec{c})^2.$$

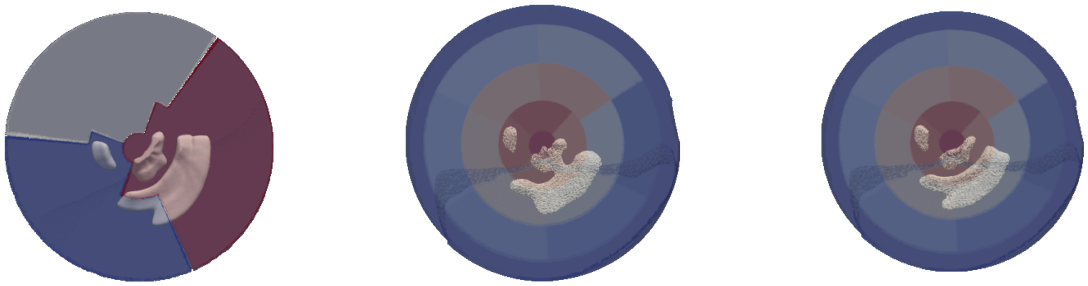
Minimizing with respect to $\vec{c} \in \mathbb{R}^m$ yields the optimal weights. The analytical solution for the optimal parameter values is

$$\vec{c} = (B^T B)^{-1} B^T \vec{s}.$$

Once $\vec{c}$ has been obtained, $\vec{s}$ can be reconstructed through $\vec{s} = B\vec{c}$.

To evaluate the required number of control points in each direction, we visually inspect the reconstructions and compare the results to the originals. Figure 8.14 displays a scar projected onto basis splines with varying number of control points. Less control points correspond to a stronger dimensionality reduction. The B-spline projection creates a smooth surface, which alleviates some of the processing issues explained in section 7.4. However, it can also have undesired effects. Figure 8.14b displays a bridge between two initially disconnected parts. We choose a resolution which reconstructs the signed distance function nearly lossless for all scars and consider processing artifacts as ground-truth.

Figure 8.14: Projection onto basis splines



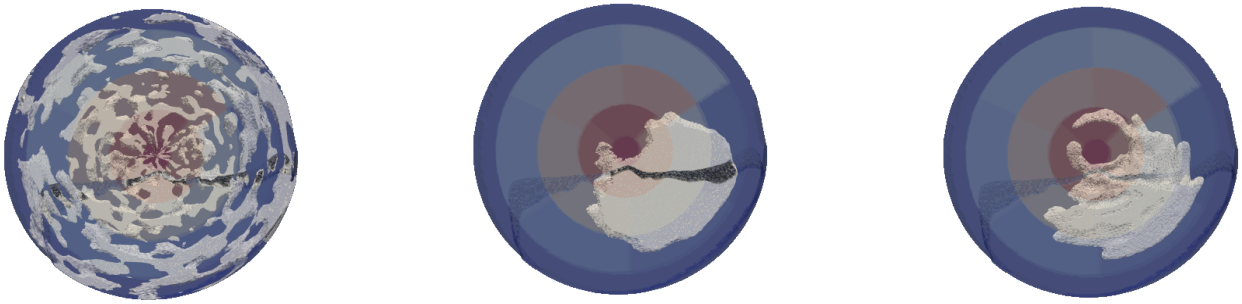
(a) original (b) low resolution (c) high resolution

8.3.4 Results

During computation, we store the B and $(B^T B)^{-1}$ matrices in a sparse format. For our purposes we choose 30, 50 and 5 control points in θ, ρ, z coordinates respectively. This corresponds to a nearly exact reconstruction of the original signed distance function, including all processing artifacts. The projection reduces dimensionality by a factor of one hundred and enables overfitting an auto-encoder to the whole dataset. Figure 8.15 visualizes model output snapshots during the training process. Whereas training on the high-dimensional signed distance function (see figure 8.9) optimizes every point individually, training on the B-spline representation is coarse grain.

Figure 8.15: Overfitting a variational auto-encoder to the B-spline coefficients

(a) Epoch 0 (b) Epoch 2000 (c) Epoch 5000



Chapter 9

Part 2: Algorithm and Generations

Generative models span from actor-critic methods such as generative adversarial networks, to statistical kernel-parametrization with Gaussian Process latent variable models, to variational auto-encoders over normalizing flows, diffusion models and, finally, linear principal component analysis. In principle, all of these models can be formulated in a way that enables learning a low-dimensional latent space distribution. However, the simplest models for these purposes tend to be variational auto-encoders and principal component analysis.

Generative adversarial networks implicitly minimize a divergence objective through adversarial training. While they achieve state of the art results in generative tasks, the implicit optimization through adversarial training makes interpretation and training difficult. Statistical covariance parametrization (as common in Gaussian processes) are among the most interpretable models but can suffer from a high computational complexity. In general, these models require a high amount of expertise for optimization, although, recently, many python libraries (such as *gpytorch* [24]) enable their usage in a deep-learning framework similar to *pytorch*. While popular statistical computational anatomy, this model class is relatively unexplored for generative purposes. Diffusion networks have enjoyed great success in image-generation with applications such as Open-AIs DALL-E [59] and Stability AIs Stable-Diffusion [65]. However, these models require a high number of iterations for sampling, making them slow. They can be considered a special case of variational auto-encoders, where transitions in a markov chain are modeled by the latter. Contrary to many of these methods, which replace a maximum likelihood objective with a lower bound or optimize the real objective implicitly through adversarial training, normalizing flows can be optimized with respect to the maximum likelihood. However, they require the computation of the transformation Jacobian which is computationally expensive, especially for expressive models.

In our work we will apply variational auto-encoders and principal component analysis to generate data-samples. The latter is a well-understood linear data-analysis method which is easy to implement and use. By virtue of its analytical solution no numerical optimization procedure is needed. The former corresponds to a (at first view) conceptually simple model class with relatively low computational bottlenecks. It is easy to implement and train and widely used for dimensionality reduction tasks and generative purposes. These models are explained in section 5.4 and section 5.6. We use *sklearn* to program the principal component analysis model and *pytorch* for the variational auto-encoder.

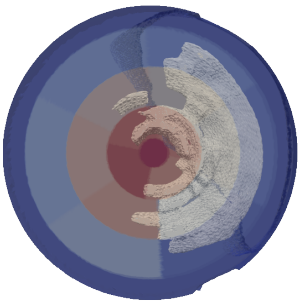
9.1 Naive PCA

Through PCA we compute a linear map from isotropic gaussian noise to the data-distribution. This requires meaningful point-to-point correspondence which is not a-priori given between

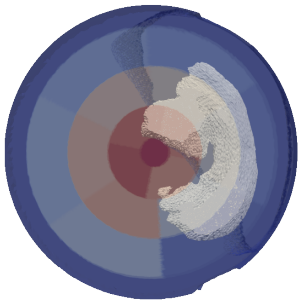
the scar meshes. Common approaches (e.g. used in statistical shape models) establish correspondence through non-rigid registration to a template mesh. However, due to the different topologies of the scar meshes, this is not possible. No point-to-point correspondence can be trivially established between topologically different domains. However, since the signed distance values are available on identical point-sets (the heart mesh is the same for each scar), correspondence is provided for heart mesh vertices and their signed distance values. The signed distance function function implicitly represents scar shapes which can be recovered through iso-surface extraction at the zero level-set. While the point-to-point correspondence is correspondence of signed-distance values and not shape landmarks, we attempt the direct application of principal component analysis. We reduce the dimensionality through a mapping to B-spline coefficients and use the respective coefficient vector as representation of a shape. In a first attempt we model the data-distribution through PCA.

Fig. 9.1 displays a scar shape and its reconstruction using 8 and 43 principal components. As expected, using 8 principal components keeps the main form but disposes of small variabilities. Using 43 components represents the shape nearly lossless. We use 43 principal components

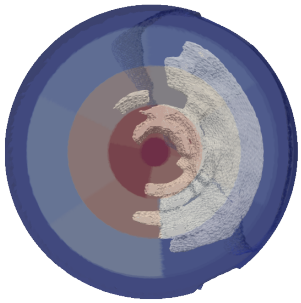
Figure 9.1: Principal component reconstructions



(a) Original scar shape



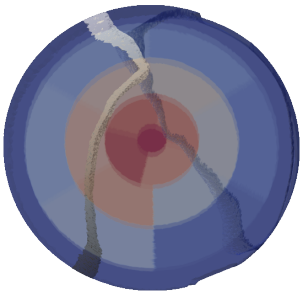
(b) Reconstruction with 8 principal directions



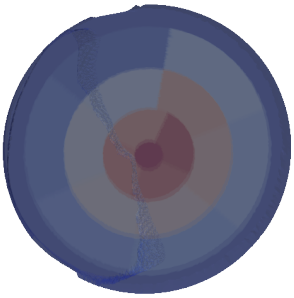
(c) Reconstruction with 43 principal directions

to create new scars by sampling from the modeled distribution. Visual inspection reveals that some generated shapes are implausible. Fig 9.2a shows a completely nonphysical shape and Fig. 9.2b corresponds to a purely positive signed distance function from which no shape can be extracted. This indicates that the shape distribution represented by B-spline coefficients can not be represented by a linear transformation of white noise.

Figure 9.2: Samples from the PCA distribution



(a) Sample 1



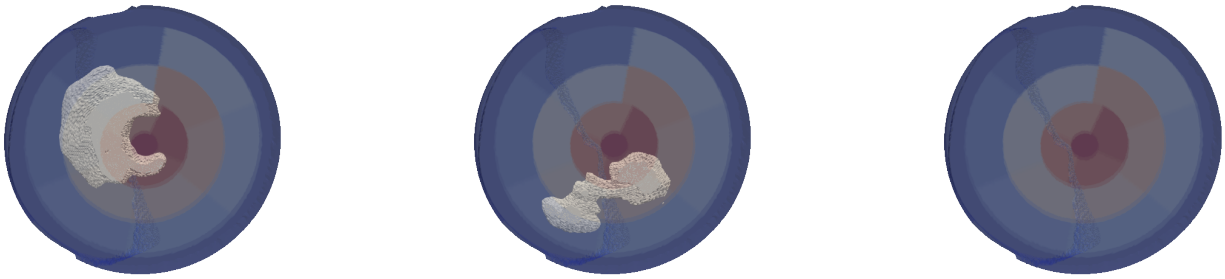
(b) Sample 2

9.2 Naive VAE

We train a variational auto-encoder on 44 shapes represented by the basis spline coefficients. We choose the mean squared error as reconstruction loss. The decoder output correspond to generated B-spline coefficients, which are mapped to signed distance functions. This function implicitly represents generated scar shapes which can be recovered through iso-surface extraction at the zero level-set. We choose an isotropic Gaussian as prior distribution of the latent-variables. Regularization with the D_{KL}-divergence encourages encoding close to this prior distribution. After the training, the encoder is disposed of. Latent variables are sampled from the isotropic Gaussian and decoded.

Figure 9.4 displays some of the generated shapes. Fig. 9.3a displays a shape which (through visual inspection) can be considered plausible. However, figure 9.3b and figure 9.3c are questionable generations. The scars in our training data revolve around the disk center and are either circular, semi-circular shaped or elliptical (see figure 9.14). Figure 9.3b does not display these features and looks a bit out of place when compared to the available samples. Figure 9.3c corresponds to a purely positive signed distance function from which no scar shape can be extracted.

Figure 9.3: Generations (VAE trained on basis spline coefficients)

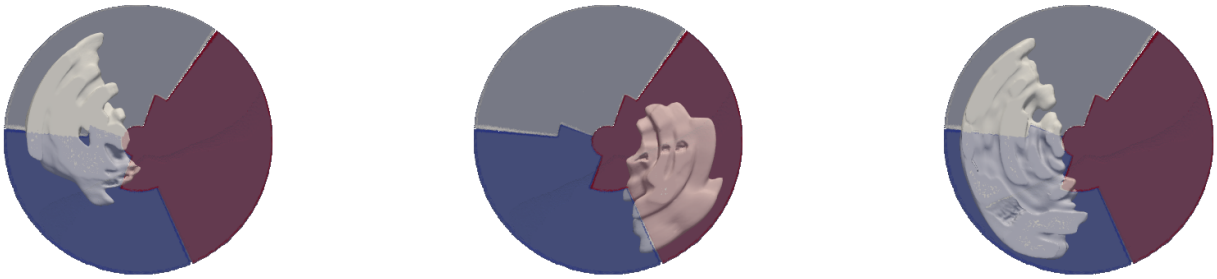


(a) a VAE generation (b) a VAE generation (c) a VAE generation

It is likely, that these issues are due to the low amount of available data samples. Consider the extreme case, where only two samples are available. Consider training a variational auto-encoder on these two shapes with a one dimensional latent space. Each of the two samples is encoded to a Gaussian distribution with mean and covariance. Due to the D_{KL}-divergence term, the encoding is encouraged to be as close the isotropic Gaussian as possible. Avoiding mode collapse (where both distributions become the isotropic Gaussian exactly), leads to a multi-modal distribution with two peaks. During sampling from this multi-modal distribution, sometimes a latent variable is decoded which has equal probability of belonging to either shape. For these samples, a trained decoder generates a sample which minimizes the reconstruction loss with respect to both shapes. This reconstruction is the euclidean mean of the signed-distance function.

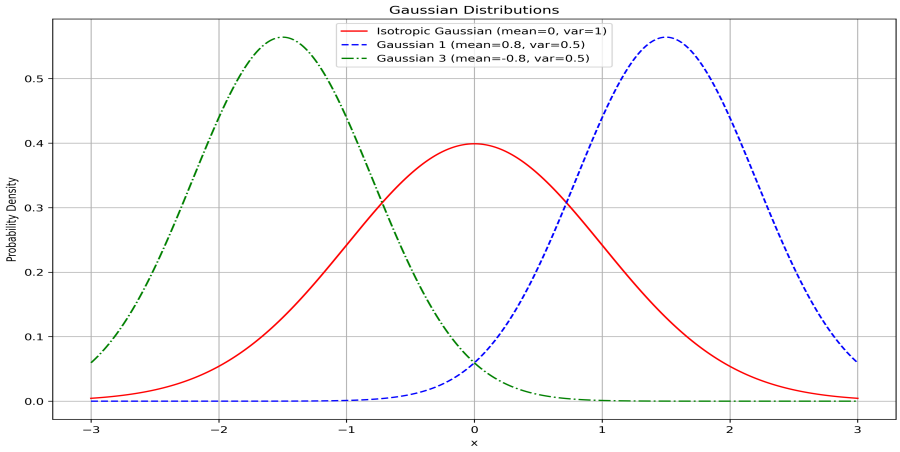
Figure 9.5 displays the isotropic Gaussian and two latent distribution corresponding to the two training data-samples. The latent-value which intersects both distributions, leads to a generation which minimizes the reconstruction loss to both shapes. If the two data-samples correspond to figure 9.4a and 9.4b, then the euclidean mean is a purely positive signed distance function. If the data-samples correspond to figure 9.4a and 9.4c, the euclidean mean contains positive and negative values. This example illustrates how the lack of overlapping scars can lead to implausible generations (for example purely positive signed distance functions) for some latent codes.

Figure 9.4: Data-sample examples



(a) Sample 1 (b) Sample 2 (c) Sample 3

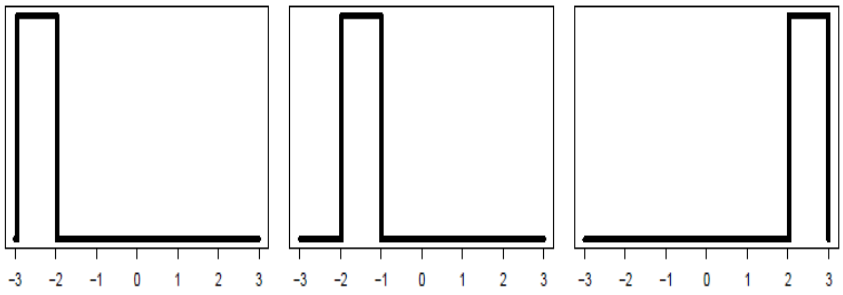
Figure 9.5: 1D latent space visualization with two training samples



9.3 On the mean of distributions

The problematic elucidated in section 9.2 is related to the representation as signed distance function and the euclidean mean squared error (also called the $L2$ distance) as reconstruction loss. Note, that the signed distance function can be interpreted as a distribution simply by normalizing it so that all values sum up to one. Consider the three distributions (interpretable as shapes) displayed in figure 9.6. The three shapes have the same mean squared error to each other, i.e. equal L_2 distance. However, intuitively we can see that the shape represented by p_2 should be closer to p_1 than p_3. The L_2 distance is not able to capture this. The Kullback-Leibler (D_{KL}) divergence (see section 5.5.2) is an approximation of the distance between distributions with strong theoretical background, however, it is not well defined for these shapes, since they have no overlap. In the domain where $p_1 \neq 0$ we have $p_2 = 0$ and $p_3 = 0$. This would lead to a Kullback-Leibler divergence of zero or be ill-defined with division by zero. Thus neither the L_2 distance, nor the D_{KL} divergence are suited to measure similarity of these distributions. Another type of distance based on optimal transport is robust to the issues which arise with the D_{KL} divergence and captures several properties which are not represented by the L_2 distance. This distance is called the Wasserstein distance (see [39], [82] , [58]). For illustrative purposes only, we contrast some of its properties with properties of the L_2 distance to highlight the problematic than can arise from using the L_2 distance as reconstruction loss. As mentioned in section 9.2, during variational training a latent sample can appear which has equal probability of belonging to two inputs. Thus its optimal reconstruction must be so that it minimizes the reconstruction loss to both shapes. If the reconstruction loss is the $L2$ distance (equivalent to the mean squared error), then the optimal reconstrucion corresponds to the euclidean mean. In

Figure 9.6: Three densities p_1, p_2, p_3. Each pair has the same distance in L_2. However in Wasserstein distance p_1 and p_2 are closer.



https://www.stat.cmu.edu/%7Elarry/=sml/Opt.pdf

the Wasserstein space, the distribution which minimizes the summed Wasserstein-distance to all other distributions is often called the Wasserstein barycenter. The upper image in figure 9.7 displays two distributions (interpretable as shapes) in red and blue. The black dotted line show the euclidean mean (i.e. the L_2 barycenter). The green dotted line displays the wasserstein barycenter. Note that the bell shaped form of the initial distributions is preserved by the latter and destroyed by the former. Note, that if the two initial shapes were aligned, both barycenters would coincide, i.e. the L_2 distance and the Wasserstein distance are equivalent. The Wasserstein distance can also be used to define geodesics between distributions. The lower image in figure 9.7 shows an interpolation along the geodesic connecting the two initial distributions. In fact, the Wasserstein distance gives rise to a Riemannian manifold, which, by definition, is locally isomorphic to Euclidean space. Figure 9.8 displays another example

Figure 9.7: Top: Two distributions p_1 (blue), p_2 (red) and the barycenters with the L_2 and wasserstein distance. Bottom: Interpolation from p_1 to p_2 along the Wasserstein geodesic.

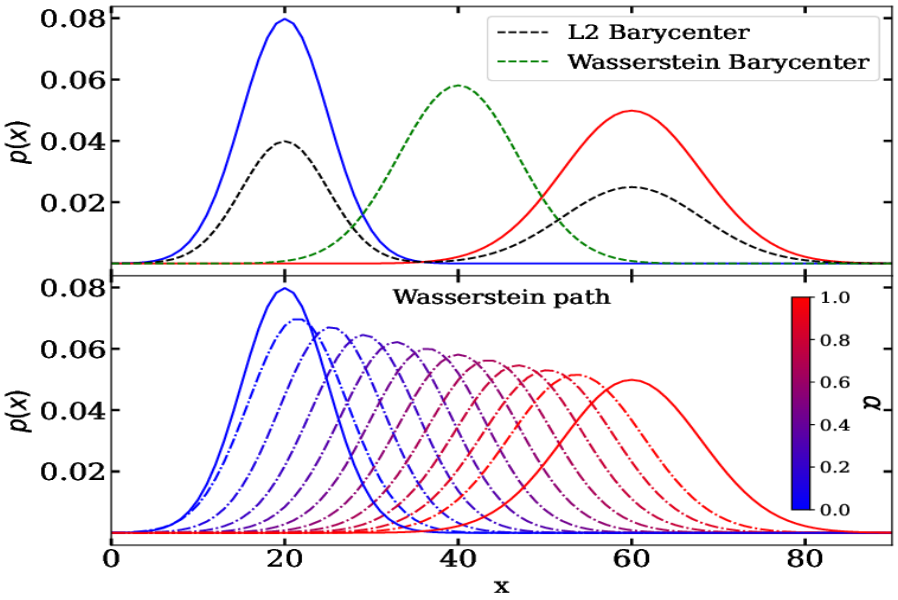
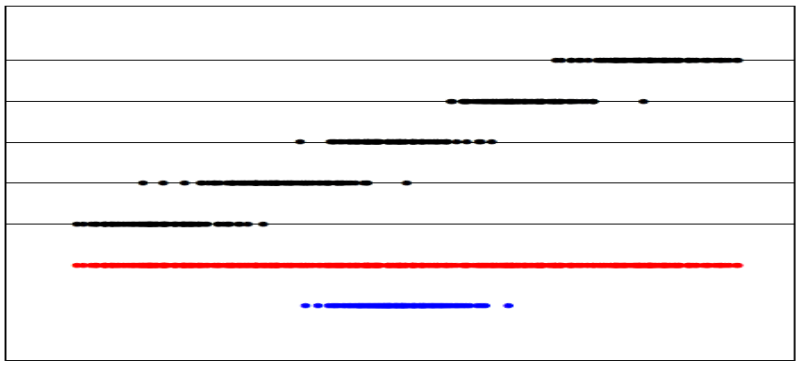


Image from *A Novel Optimal Transport-Based Approach for Interpolating Spectral Time Series* [60]

which illustrates the euclidean mean and Wasserstein barycenter between data-samples. We

Figure 9.8: The top five lines show five, one-dimensional datasets. The red points are what happens if we simple average the give empirical distributions. The blue dots show the Wasserstein barycenter which, in this case, can be obtained simply by averaging the order statistics.



https://www.stat.cmu.edu/%7Elarry/=sml/Opt.pdf

mention these examples merely to point out the problematic related to the L_2 distance to quantify the quality of reconstructions. Computation of wasserstein distances (and geodesics) gets complex very quickly. Within the context of this work, we do not provide any experiments with it. Instead, we require the shapes to be sufficiently similar so that the L_2-distance can be used. One way of encouraging such a similarity is by aligning the shapes and/or by restriction to infinitesimal neighborhoods (since the manifold is locally isomorphic to euclidean space). Future work could examine the effect of a Wasserstein reconstruction loss. The curious reader is referred to [39] for a first (intuitive) introduction into optimal transport, to [82] and [58] for the (more rigorous) mathematical theory behind the Wasserstein space and to [87] for an implementation of a variational autoencoder with Wasserstein loss for the latent distributions and the reconstructions. In contrast, [77] develops a variational autoencoder with Wasserstein loss only in the latent space.

9.4 Shape alignment

The low data density leads to implausible generations due to lacking shape overlap (see section 9.2). As a solution we perform a pre-processing step which aligns the shapes. In doing so we consider shape and position as independent from each other. This is not true. The diseased coronary is correlated with the location, size and shape of the scar. However, this assumption is a reasonable first approximation and alleviates the limitations posed by the data. The correlation could potentially be modeled through post-processing.

We morph the shapes with the goal of increasing overlap through a diffeomorphic mapping, allowing for some bending and scaling. This normalizes position and size and considers these factors independent from shape. We choose a transformation class which enables a computable inverse and restrict the map so that is maintains desired properties. The transformation parameters are chosen by minimization of an energy-function between the target and the morphed shape. This energy-function considers a distance between shapes and a penalty term for strong bending and scaling.

9.4.1 Optimization objective

The optimization procedure is similar to diffeomorphic image registration. Analogously, we search for a transformation which maps between shapes and is invertible. We choose a loss, or

energy, which quantifies the distance between the shape boundary S_1 and S_2. It is defined as the integral of the closest distance

$$d(S_1, S_2) = \int_{x \in S_1} d(x, S_2) dx,$$

where $d(x, S_2) = \min(|x - x_2|^2)$ with $x_2 \in S_2$ denotes the smallest squared distance from $x \in S_1$ to the surface S_2. Furthermore, we add a term which penalizes the type of transformation. In image registration such a penalty might be constructed to be a gradient regularization (imposing a smoothness constraint), an elastic regularization (penalizing bending and ensuring physicality) and a diffusion regularization (penalizing scaling). We choose a transformation which bijectively maps points in the disk to points in the disk. It is defined in cylindrical coordinates, acting independently in each of the coordinates, as

$$(\rho, \theta, z) \mapsto f(\rho, \theta, z) = (f_\rho(\rho), f_\theta(\theta), f_z(z)).$$

The z coordinate is kept unchanged:

$$f_z(z) = z$$

and the transformations affecting the polar coordinates ($\rho' = f_\rho(\rho)$ and $\theta' = f_\theta(\theta)$) are defined implicitly by

$$2\pi\rho' - a\sin(2\pi\rho') - b\cos(2\pi\rho') + b = 2\pi\rho + a\sin(2\pi\rho) + b\cos(2\pi\rho) - b$$

and

$$\theta' - \mu\sin\theta' - \nu\cos\theta' = \theta + \mu\sin\theta + \nu\cos\theta + \alpha.$$

Here, we have $\rho \in [0, 1]$ and θ is cyclic modulus 2π. This implicit definition is bijective and smooth and thus diffeomorphic. However, we still employ penalty terms to constrain the transformation. The total energy-function results in

$$\mathcal{L}_{energy}(a, b, \mu, \nu, \alpha) = \mathcal{L}_{similarity}(a, b, \mu, \nu, \alpha) + \lambda \mathcal{L}_{regularisation}(a, b, \mu, \nu, \alpha), \tag{9.1}$$

where $\mathcal{L}_{regularisation}$ is composed as the sum of the individual penalty terms, a, b, μ, ν, α, corresponds to the transformation parameters and $\mathcal{L}_{similarity} = \sum_{i=0}^{N} d(S_1, S_i)$ measures similarity between the shapes (S_1, S_2). A possible term for gradient regularization is

$$\mathcal{L}_{\text{gradient}} = \int \left(\left(\frac{d\rho'}{d\rho} \right)^2 + \left(\frac{d\theta'}{d\theta} \right)^2 \right) d\rho d\theta,$$

which in discrete settings can be written as

$$\mathcal{L}_{\text{gradient}} = \sum_{i=1}^{N-1} \left(\left(\frac{\rho'_{i+1} - \rho'_i}{\rho_{i+1} - \rho_i} \right)^2 + \left(\frac{\theta'_{i+1} - \theta'_i}{\theta_{i+1} - \theta_i} \right)^2 \right) (\rho_{i+1} - \rho_i)(\theta_{i+1} - \theta_i).$$

A possible term for elastic regularization is

$$\mathcal{L}_{\text{elastic}} = R_{\text{elastic}} = \int \left(\left(\frac{d\rho'}{d\rho} - 1 \right)^2 + \left(\frac{d\theta'}{d\theta} - 1 \right)^2 \right) d\rho d\theta,$$

which in discrete settings can be written as

$$\mathcal{L}_{\text{elastic}} = \sum_{i=1}^{N-1} \left(\left(\frac{\rho'_{i+1} - \rho'_i}{\rho_{i+1} - \rho_i} - 1 \right)^2 + \left(\frac{\theta'_{i+1} - \theta'_i}{\theta_{i+1} - \theta_i} - 1 \right)^2 \right) (\rho_{i+1} - \rho_i)(\theta_{i+1} - \theta_i).$$

It penalizes deviations from the identity transform. A possible term for diffusion regularization is

$$\mathcal{L}_{\text{diffusion}} = R_{\text{diffusion}} = \int \left(\left(\frac{d^2\rho'}{d\rho^2} \right)^2 + \left(\frac{d^2\theta'}{d\theta^2} \right)^2 \right) d\rho d\theta,$$

which in discrete settings can be written as

$$\mathcal{L}_{\text{diffusion}} = \sum_{i=2}^{N-1} \left(\left(\frac{\rho'_{i+1} - 2\rho'_i + \rho'_{i-1}}{(\rho_{i+1} - \rho_i)^2} \right)^2 + \left(\frac{\theta'_{i+1} - 2\theta'_i + \theta'_{i-1}}{(\theta_{i+1} - \theta_i)^2} \right)^2 \right) (\rho_{i+1} - \rho_i)(\theta_{i+1} - \theta_i).$$

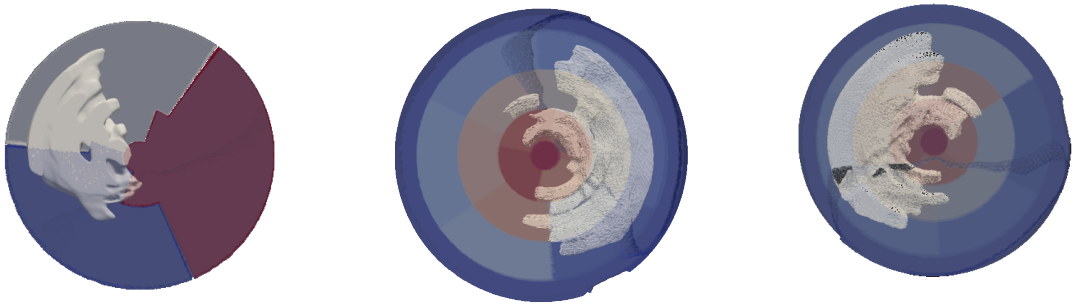
Each of these terms is weighted and summed to the similarity loss. By adjusting the respective weights we can control for the amount of bending, stretching and scaling.

9.4.2 Transformations and generations

We choose a reference shape and align all other shapes to it by choosing a transformation which minimizes the energy function. This is done via gradient based optimization (see section 5.8).

Figure 9.9a display the reference shape. Figure 9.9b displays a shape before the transformation and 9.9c displays the shape after the transformation. The main features are maintained while the shape overlap is increased. This causes the overlap of negative regions in the signed distance representation and enables the use of the mean squared error as reconstruction loss by avoiding the generation of purely positive signed distance functions. The normalization is applied as

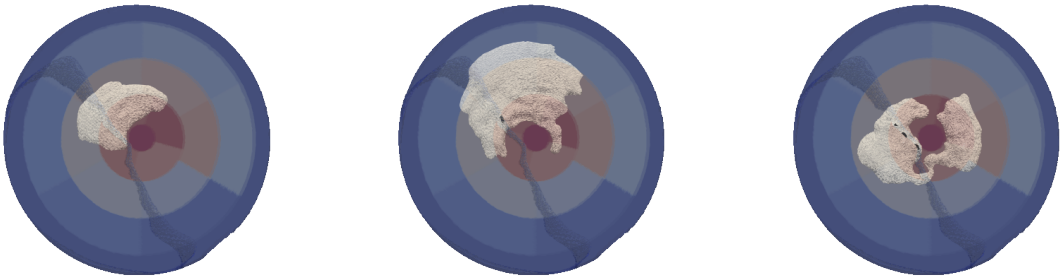
Figure 9.9: Shape Transformation



(a) Reference shape (b) Original Shape (c) Transformed Shape

a pre-processing step of the training data. This provides a generative model of shapes in the same normalized location. The generated shapes can be moved to any other location as a post-processing step. Figure 9.10 displays examples. Figure 9.10a and 9.10b can both be considered plausible scars. Figure 9.10c displays some nonphysical features. Our experiments show that the position normalization alleviates the issue of purely positive signed distance generations. However, the generations lack features, such as circular topologies (see figure 9.21a).

Figure 9.10: Generations on normalized shapes



(a) generated shape 1 (b) generated shape 2 (c) generated shape 3

9.5 Manifold hypothesis

A common assumption in data science is the manifold hypothesis [20], [44], which states that real world data tends to live on a low dimensional manifold. A manifold is a topological space M that satisfies the following conditions:

- There exists a collection of open sets $\{U_i\}_{i\in I}$ that cover M, i.e., $\bigcup_{i\in I} U_i = M$.
- For each U_i, there exists a homeomorphism $\varphi_i : U_i \to \mathbb{R}^n$ (where n is a fixed integer) such that $\varphi_i(U_i) \subseteq \mathbb{R}^n$. The pair (U_i, φ_i) is called a *chart*.
- The maps φ_i and φ_j must satisfy the *compatibility condition*: for any $x \in U_i \cap U_j$, the map $\varphi_j \circ \varphi_i^{-1} : \varphi_i(U_i \cap U_j) \to \varphi_j(U_i \cap U_j)$ is a smooth (or continuously differentiable, depending on the context) map.

An atlas $\mathcal{A}$ on a manifold M is a collection of charts $\{(U_i, \varphi_i)\}_{i\in I}$ such that:

- The sets $\{U_i\}_{i\in I}$ cover M, i.e., $\bigcup_{i\in I} U_i = M$,
- For each pair of charts (U_i, φ_i) and (U_j, φ_j), the map $\varphi_j \circ \varphi_i^{-1} : \varphi_i(U_i \cap U_j) \to \varphi_j(U_i \cap U_j)$ is smooth (or continuously differentiable).

An atlas is called a smooth atlas if the transition maps between charts are smooth.

Figure 9.11: A manifold with its charts

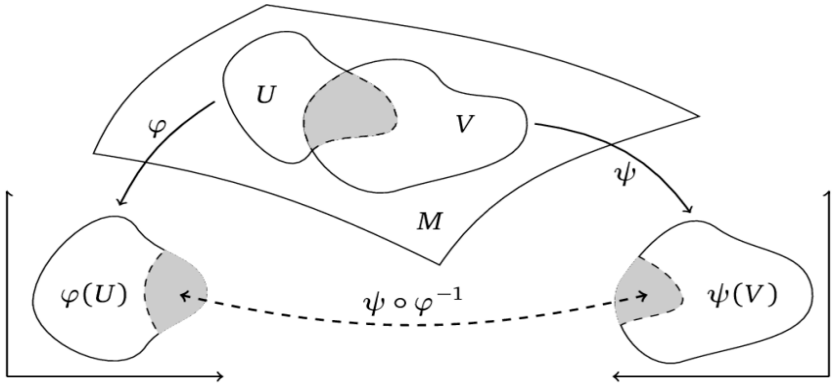


image from the book *Quaternion Algebras (pp.605-627)* [83]

Let M be a smooth manifold and $p \in M$ a point on the manifold. The tangent space of M at p, denoted by T_pM, is a vector space that consists of the tangent vectors at the point

p. There are two common equivalent definitions of the tangent space. The tangent space at a point $p \in M$, T_pM, is the set of derivations at p. A derivation is a linear map $\delta_p : C^\infty(M) \to \mathbb{R}$ that satisfies the Leibniz rule:

$$\delta_p(fg) = \delta_p(f)g(p) + f(p)\delta_p(g),$$

for all smooth functions $f, g \in C^\infty(M)$, where $\delta_p(f)$ represents the derivative of f at p in the direction of the tangent vector. This definition captures the idea of a tangent vector as a linear map acting on smooth functions. The tangent space can also be understood as the set of equivalence classes of smooth curves passing through p. Let $\gamma : (-\epsilon, \epsilon) \to M$ be a smooth curve such that $\gamma(0) = p$. The tangent vector to the curve γ at p is given by the derivative of γ at $t = 0$, denoted $\dot{\gamma}(0)$. The tangent space T_pM is the set of all such derivatives of smooth curves through p. In both definitions, the tangent space is a vector space that consists of the directions in which one can move infinitesimally from the point p on the manifold. Figure 9.12 visualizes a manifold, its tangent space at point p, a curve $\gamma(t)$ and its tangent vector v. The tangent bundle of M, denoted by TM, is the disjoint union of the tangent spaces at each point of M. It is defined as:

$$TM = \bigsqcup_{p \in M} T_pM.$$

The tangent bundle is itself a smooth manifold with twice the dimensionality of M and the following key properties:

- **Projection map:** There is a natural projection map $\pi : TM \to M$, which projects a pair $(p, v) \in TM$ to the point $p \in M$:

$$\pi(p, v) = p.$$

- **Smooth structure:** The tangent bundle TM inherits a smooth structure from the manifold M. It is given a smooth atlas where each chart of M is "lifted" to a chart on TM. If we fix a matric in the tangent spaces of the tangent bundle we say that we have a Rimaniann structure on the manifold M

Figure 9.12: A manifold and its tangent space

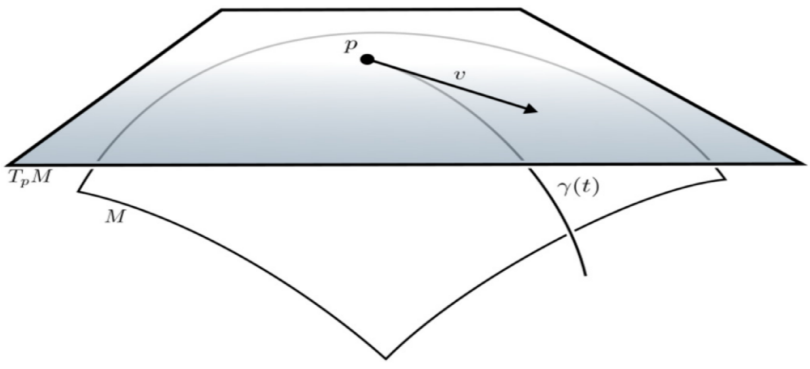


image from *Visualization of Nil, Sol, and SL2(R) geometries* [55]

Figure 9.13a visualizes a manifold in form of a circle and its tangent bundle.

Figure 9.13: Linear approximation of a manifold

(a) A circle and its tangent bundle

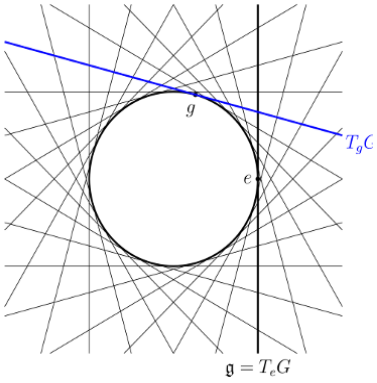
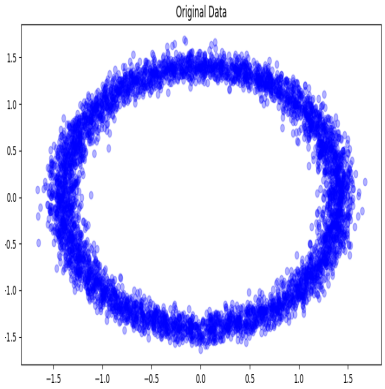


image from
https://math.stackexchange.com/questions/2328045/tangent-spaces-of-so3

(b) A data distribution on a circle



Let us consider a simple data distribution on a manifold, for example on a circle. An example is visualized in 9.13b. Note, that the data distribution on the circle is locally isomorphic to a gaussian data-distribution. While principal component analysis can not model the non-linear data manifold (see section 5.7) we can approximate the circle by a number of gaussian distributions which individually can be modeled by principal component analysis and then reconstruct an approximation of the data manifold as the combination of all principal component models. We have shown in section 5.7 that a variational auto-encoder can directly sample from the data manifold when it is a circle, but that it might struggle for more complex manifolds. Nonetheless, its ability to model non-linear maps make it more robust for modeling data distributions on non-euclidean manifolds.

With reference to the manifold hypothesis [20], [44] we hypothesize that our data is distributed on some differentiable manifold. Figure 9.14 shows how such a manifold could look like. "Similar" shapes are located in neighborhoods of each other. This assumption is strengthened by our data representation as a signed distance function which, through normalization so that its values integrate to one, can be interpreted as a probability distribution. The field of information geometry (see [53]) and optimal transport (see [58]) both formalize the existence of a differentiable manifold on the set of probability distributions. Information geometry utilizes the fisher information matrix (see section 5.5.2) as a Riemannian metric tensor and employs two, the exponential and the logarithmic connection, whereas the Wasserstein space, arising from optimal transport, is a true Riemannian manifold with the Wasserstein distance as metric and the Levi-Civita connection. In both models, geodesics between probability distributions can be formalized, although their computation might be complex. Thus it is reasonable to assume that our data lies on a differentiable manifold with infinitesimal neighborhoods and tangent spaces being isomorphic to euclidean space. We choose a set of similar shapes and assume that they are located within an approximately euclidean neighborhood. In analogy to the distribution on a circle we expect that it is possible to model the data-distribution within an infinitesimal neighborhood using PCA. Figure 9.15 displays a chosen set of three shapes. While the shapes displayed in figure 9.15a and figure 9.15b are very similar, the shape in figure 9.15c is more dissimilar, indicating a bigger geodesic distance. We will assume that all three shapes are within an approximately euclidean neighborhood of the data manifold. In manifold learning, a common methodology approximates the data manifold by a set of euclidean neighborhoods [51].

Figure 9.14: The shape manifold

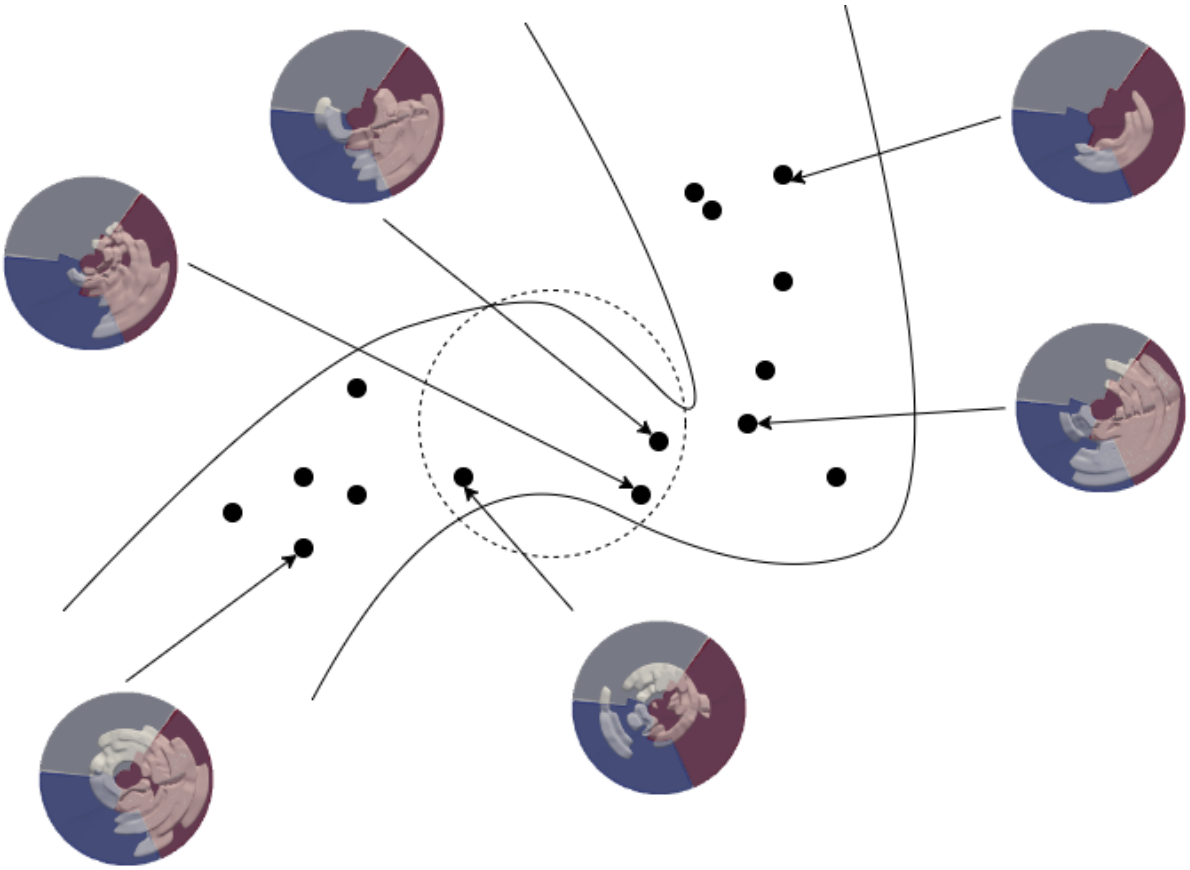
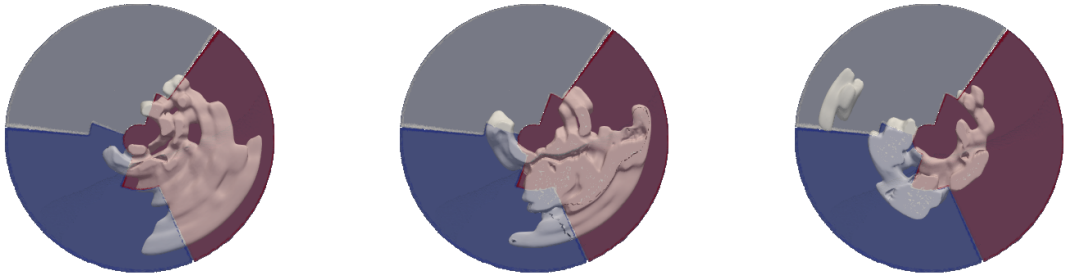


Figure 9.15: Shapes from a neighbourhood



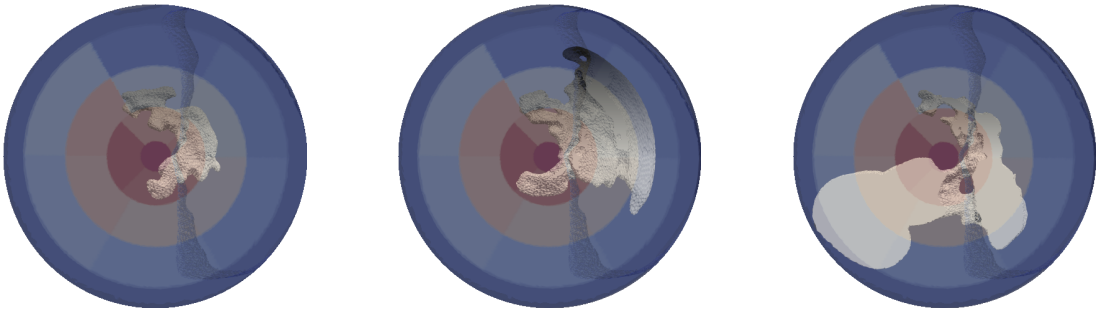
(a) Shape 1 (b) Shape 2 (c) Shape 3

9.5.1 PCA

We apply PCA on the B-spline coefficients representing the shapes displayed in figure 9.15. If the manifold hypothesis holds, we expect that sampling from the model distribution produces visually plausible generations. Figure 9.16 displays some generations. The generative quality is clearly improved when compared to naive PCA on the whole data space (see section 9.1). This indicates that restricting the model to a set of similar shapes can improve the generative quality. The generations displayed in figure 9.16b and 9.16a are good generations. The shape visualized in figure 9.16c displays an implausible part in form of exaggerated scar tissue growth towards the left of the disk. This might be caused by the disjoint scar tissue and the more dissimilar shape of figure 9.16c.

Supporting the hypothesis of euclidean neighborhoods, we achieve generations of better quality by restricting the training data to similar shapes. However, some implausible generations occur, indicating that the chosen samples are not sufficiently similar to be considered within the same euclidean space.

Figure 9.16: Generation within a neighborhood

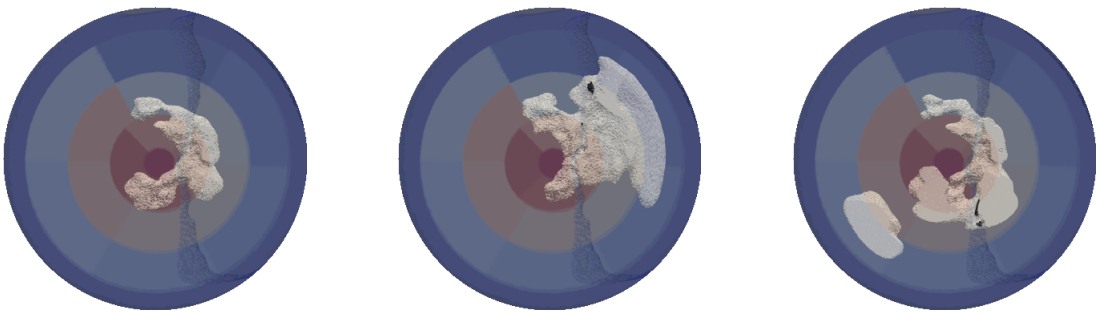


(a) generated shape 1 (b) generated shape 2 (c) generated shape 3

9.5.2 VAE

We train a variational auto-encoder on the samples displayed in figure 9.15. Figure 9.17 displays the generations. Visual inspection indicates that better generations are obtained when compared with PCA based models. While figure 9.17a and figure 9.17b are similar in quality to the PCA generated samples, figure 9.17c is of better quality This indicates that by employing non-linear methods on nearly euclidean neighborhoods of the data-manifold we achieve similar results to PCA if the data really can be assumed to lie in a euclidean neighborhood. However, the variational auto-encoder is more robust to deviations. As shown in section 5.7, the variational auto encoder can also model data distributions which lie on a curved, non euclidean manifold. For example, while the data-distribution modeled through PCA did not generate the disjoint scar tissue (see figure 9.16c), the variational auto-encoder creates data samples with a similar disjoint scar tissue (see figure 9.17c). The generations visualized in figure 9.17a and figure 9.17b are visually appealing and hard to distinguish from the samples in the data set while being novel. Many fine level details are maintained.

Figure 9.17: Generation within a neighborhood



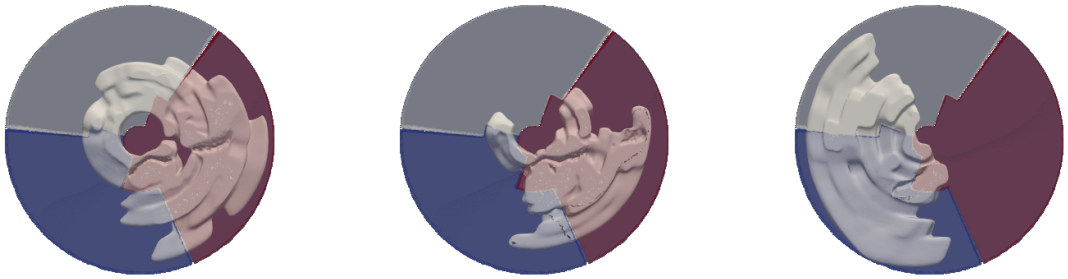
(a) generated shape 1 (b) generated shape 2 (c) generated shape 3

9.5.3 Clustering

In another experiment we cluster the scars into three groups. These groups are called the circular, longitudinal and semicircle clusters, with the naming suggestive of visually striking characteristics. Some shapes can belong to both groups. Since shapes within a group are similar,

they can be considered neighborhoods of the low-dimensional latent-space. We train variational

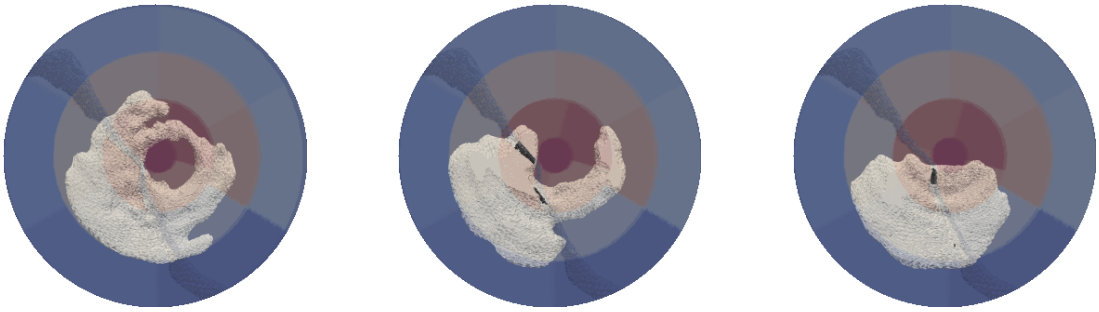
Figure 9.18: Feature based groups



(a) Circular shaped scar (b) Semi-circular shaped scar (c) longitudinally shaped scar

auto-encoders on each group separately and examine the generations. Visual inspection indicates that the visually striking features (circular, longitudinal, semicircle) are maintained in the generations.

Figure 9.19: Feature based generations



(a) Circular shaped generation (b) Semi-circular shaped generation (c) longitudinally shaped generation

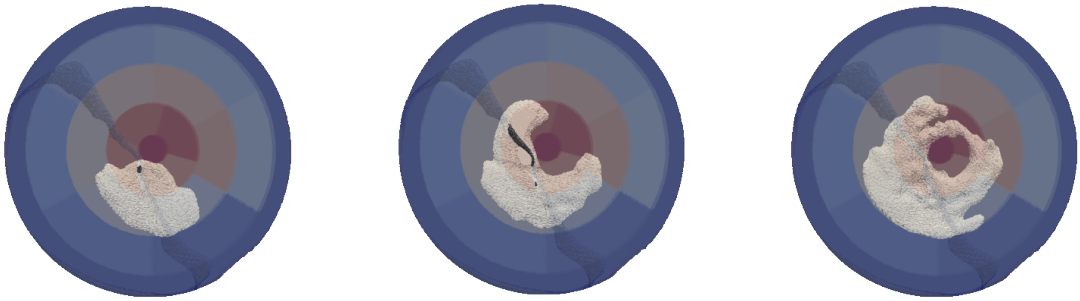
9.6 Generated scars

Our experiments examine training on the full (but transformed) dataset (see section 9.4) and training on a small set of similar shapes 9.5. It is striking that the generations obtained by training on a small set of similar shapes contain many of the more detailed features found in the dataset. For example, the scars have a block like structure which likely originates from the processing (see section 7.4). Contrary to this, the generations created on the full dataset are smoothened and only contain the most general features, such as the overall shape and size. Training on three groups (see section 9.5.3), each containing visually striking features that we want to generate, strikes a middle ground. While generations with circular topology (which did not occur after training on the full aligned dataset) can be created, the generations are smoothened and do not display the block like structure found in the dataset. Further research could examine the best similarity measure, iterative methods with differently sized groups and automatized clustering (with K nearest neighbors or Gaussian Mixture Models) and more.

We train variational auto-encoders separately on clusters corresponding to a circular, longitudinal and semicircular shape and visualize the generations in the disk (see figure 9.20), heart (see figure 9.21) and as segmentation masks (see figure 9.22). The generations display visually plausible features indicating that they can be considered ischemic scars. For example, the cir-

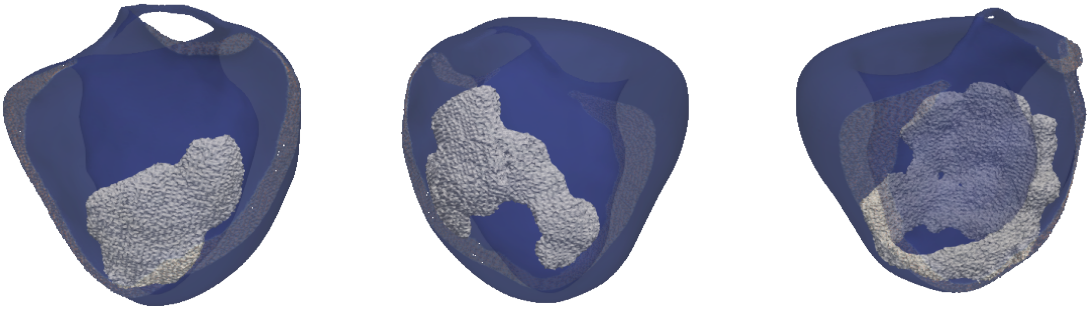
cular, longitudinal and semicircular structure of the respective cluster is preserved. The size and shape is similar to samples in the data set. Furthermore, the segmentations show that the generated scars tend to grow from endocardium to epicardium. This is in accordance with clinical knowledge.

Figure 9.20: Generations in disk



(a) Generation 1 in disk (b) Generation 2 in disk (c) Generation 3 in disk

Figure 9.21: Generations in heart



(a) Generation 1 in heart (b) Generation 2 in heart (c) Generation 3 in heart

Figure 9.22: Segmenations

(a) Segmenation 1 (b) Segmenation 2 (c) Segmenation 3

Chapter 10

Part 2: Conclusion

We provide a theoretically motivated methodology which enables the generation of novel scar shapes. For this purpose, we examined shape representations and algorithms for their suitability. Due to topological differences, we do not use standard statistical shape models but a choose a signed distance representation with a variational auto-encoder. In first experiments, we highlight the high-dimensionality of our representation and related difficulties. Our solution is mapping the high-dimensional signed distance feature vector onto basis spline coefficients. We highlight the unsuitability of the isotropic Gaussian prior with an L_2 reconstruction set for our sparse dataset with low shape overlap. Considering shape and position as independent we perform a normalizing transformation based on non-rigid image registration. Experiments show that the quality of the generations is improved and purely positive signed distance generations are avoided. Furthermore, we provide theoretical motivations supporting the manifold hypothesis. In analogy to manifold learning, we show that even linear methods (PCA) can be employed for plausible generations when the training data consists only of a few selected similar shapes. We provide a visual comparison to non-linear generations from a VAE, indicating that these are more robust to remaining non-linearity. We show how clustering into meaningful groups can be used to guide the generative process. Visual comparison to samples generated by training on bigger data sets shows that training on fewer shapes generates more of the high frequency features. In a last step we provide visualization of generated scars within the disk and heart representation. Furthermore, we create an artificial segmentation to model the data format which is commonly available to clinicians. The generated shapes are visually appealing and clinically relevant features, such as growth from endocardium to epicardium, are maintained. Our work considers the scars obtained from processing as ground-truth. However we point out several processing artifacts. Among these are the low-vertical resolution of MRI scans (leading to a block-wise structures and disjoint parts) and missing scans in the apex (leading to shapes with nonphysical holes). Summarizing, this work provides insights into the data and challenges associated to generative algorithms. We provide solutions regarding high-dimensionality, and low-quality generations due to data sparsity and show that training in neighborhoods of few, but similar, shapes can be advantageous. This is related to the manifold hypothesize which we test by applying linear methods. Our results suggest the applicability of linear methods in euclidean tangent spaces. However, nonlinear methods are more robust to approximation errors by virtue of their ability to model distributions on nonlinear manifolds.

We can now answer the questions stated in the problem formulation

- What kind of features are most relevant?

  A user would likely care about the segmental occupancy degree and transmurality of a scar. These are related to the severity of infarction. By labeling the generated scars, a gallery can be filtered.

- What is an adequate shape representation?

  Due to differing topologies, statistical shape models can not be applied trivially. By using a signed distance function we can generate shapes from data samples with differing topologies, as long as the shape overlap is adequate. This overlap dictates the generation due to the pixel wise reconstruction loss. The high dimensionality can cause issues for model fitting. We use a dimensionality reduction by representing the function as basis spline coefficients.

- Which algorithm is best suited for shape generation?

  For very similar shapes, principal component analysis can produce signed distance functions from which a plausible scar can be extracted. However, variational auto-encoders are more robust against nonlinearities and work better for more dissimilar shapes. Usage of the gaussian prior distribution and a pixel wise reconstruction loss, requires that the shapes are aligned so that the signed distance functions have overlapping regions with negative values. Clustering techniques can be used to guide the generations. Intelligent clustering can render the need for shape alignment obsolete. This might be preferred, due to the correlation between the affected coronary and the scar.

Looking to the future, steps towards an automated clustering could be examined. We showed how generations performed on few structurally similar data-samples are qualitatively better than generations on the whole data-set. It is unclear which distance measure is the best, however several possibilities can be explored. To name a few, the Wasserstein-distance [23] and our shape distance (see section 9.4) are promising candidates. Possible clustering algorithms are K-means and Gaussian Mixture Models. Given a shape distance, evaluation methods can consider the closeness of the generated shapes to the data samples. Far generations can be discarded, while close shapes can be added to the training data. This increases the data density and generative capabilities. The generated scars can be labeled with their segmental occupancy and transmurality and stored in a gallery. Filters can be applied to select shapes with relevant labels.

By aligning the shapes, we consider it to be independent from the position. However this simplifying assumption is not true in general. Additional research which correlates shape with position might be useful. Users familiar with the dataset and cardiomyopathy can do manually placement of scars by applying knowledge of physically realistic locations. Alternatively, the training data does not have to normalized with respect to position, orientation and scaling, as long as it has sufficient overlap. This can be advantageous since it removes the need for a post-processing step.

An interesting approach could also be model fitting to provided segmentation masks, similar to gaussian morph-able models [46]. Through this, a clinician could draw the scar into the MRI scan. A disadvantage of this approach is the purely two-dimensional nature of MRI scans.

While we used a variational autoencoder to describe the data-distribution, comparisons with gaussian process models could be insightful. Such comparisons have been done in **??** for non-linear statistical shape models. Direct application of a gaussian process model on a statistical shape model has not been considered due to the complications unequal topologies create for point-to-point correspondence. However, similar to free form deformations [70], which establish correspondence in the domain within which the shape is located as opposed to the shape itself, it could be interesting to examine the applicability of gaussian process models on the signed distance representation. Recent development of user-friendly libraries for such models make research in this direction attractive.

Bibliography

[1] Jazmin Aguado-Sierra, Constantine Butakoff, Renee Brigham, Apollo K. Baron, Guillaume Houzeaux, Jose M. Guerra, Francesc Carreras, David Filgueiras-Rama, Paul A. Iaizzo, Tinen L. Iles, and Mariano Vazquez. Hpc framework for in-silico trials on 3d virtual human cardiac population to assess drug-induced arrhythmic risk. *medRxiv preprint*, September 2022.

[2] D Alciatore and Rick Miranda. A winding number and point-in-polygon algorithm. *Glaxo Virtual Anatomy Project Research Report, Department of Mechanical Engineering, Colorado State University*, 1995.

[3] Shun-ichi Amari. Divergence, optimization and geometry. In *International conference on neural information processing*, pages 185–193. Springer, 2009.

[4] Shun-ichi Amari. Information geometry. *Japanese Journal of Mathematics*, 16(1):1–48, 2021.

[5] Shun-ichi Amari and Andrzej Cichocki. Information geometry of divergence functions. *Bulletin of the polish academy of sciences. Technical sciences*, 58(1):183–195, 2010.

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[7] Jason D. Bayer, Anton J. Prassl, Ali Pashaei, Juan F. Gomez, Antonio Frontera, Aurel Neic, Gernot Plank, and Edward J. Vigmond. Universal ventricular coordinates: A generic framework for describing position within the heart and transferring data. *Medical Image Analysis*, 45:83–93, 2018.

[8] Helen Suzanna Becker. *An information-theoretic unsupervised learning algorithm for neural networks*. PhD thesis, CAN, 1993. UMI Order No. GAXNN-82765.

[9] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[10] Ghazi Bouabene, Thomas Gerig, Marcel Lüthi, Andreas Forster, Dennis Madsen, Dana Rahbani, and P Kahr. Scalismo: Scalable image analysis and shape modelling, 2023.

[11] Hadrien Calmet. *Large-scale CFD and micro-particle simulations in a large human airways under sniff condition and drug delivery application*. Phd dissertation, Universitat Politècnica de Catalunya, 2020.

[12] Denis Cousineau and Sebastien Helie. Improving maximum likelihood estimation using prior probabilities: A tutorial on maximum a posteriori estimation and an examination of the weibull distribution. *Tutorials in Quantitative Methods for Psychology*, 9(2):61–71, 2013.

[13] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.

[14] Ming Ding. The road from mle to em to vae: A brief tutorial. *AI Open*, 3:29–34, 2022.

[15] Carl Doersch. Tutorial on variational autoencoders, 2016. cite arxiv:1606.05908.

[16] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 3:409–425, 1999.

[17] David L Donoho et al. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS math challenges lecture*, 1(2000):32, 2000.

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[19] Yunlong Duan, Zhen Zhou, Xiaogang Zhang, and Xin Liu. Modeling cardiac stimulation by a pacemaker, with accurate electrophysiological simulations. *Journal of Computational Cardiology*, 20:305–319, 2023.

[20] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[21] Fabio J. Fehr. Modelling non-linearity in 3d shapes: A comparative study of gaussian process morphable models and variational autoencoders for 3d shape data. 2021.

[22] Mark K. Friedberg and Andrew N. Redington. Right versus left ventricular failure. *Circulation*, 129(9):1033–1044, 2014.

[23] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a wasserstein loss. *Advances in neural information processing systems*, 28, 2015.

[24] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

[25] Bernhard L. Gerber, Julie Darchis, Jean-Benoît le Polain de Waroux, Gabin Legros, Anne-Catherine Pouleur, David Vancraeynest, Agnès Pasquet, and Jean-Louis Vanoverschelde. Relationship between transmural extent of necrosis and quantitative recovery of regional strains after revascularization. *JACC: Cardiovascular Imaging*, 3(7):720–730, 2010.

[26] Amir Gholami, Andreas Mang, Klaudius Scheufele, Christos Davatzikos, Miriam Mehl, and George Biros. A framework for scalable biophysics-based image analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, 2017.

[27] J. Goldberger, S. Gordon, and H. Greenspan. Unsupervised image-set clustering using an information theoretic framework. *IEEE Transactions on Image Processing*, 15(2):449–458, 2006.

[28] Mark Golob, Richard L. Moss, and Naomi C. Chesler. Cardiac tissue structure, properties, and performance: A materials science perspective. *Annals of Biomedical Engineering*, 42(10):2003–2013, 2014.

[29] Rubén González Viera. Numerical study of aerodynamic systems for drag reduction in cars. Master's thesis, Universitat Politècnica de Catalunya, 2023.

[30] Yulan Guo, Zhiwei Lei, Tianjia Wan, Xing Xu, and Jianbo Wang. A survey of deep learning-based 3d shape generation. *arXiv preprint arXiv:2002.04579*, 2020.

[31] Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, 2016.

[32] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR, 2022.

[33] Shao-Lun Huang, Xiangxiang Xu, and Lizhong Zheng. An information-theoretic approach to unsupervised feature selection for high-dimensional data. *IEEE Journal on Selected Areas in Information Theory*, 1(1):157–166, May 2020.

[34] David R Hunter and Kenneth Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37, 2004.

[35] Clemens Isert. *Quantum Mechanics-Based Geometric Deep Learning for Drug Discovery.* Phd dissertation, ETH Zurich, 2023.

[36] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b: A 7–billion-parameter language model engineered for superior performance and efficiency. *arXiv preprint arXiv:2310.06825*, 2023.

[37] Moien Ab Khan, Muhammad Jawad Hashim, Halla Mustafa, May Yousif Baniyas, Shaikha Khalid Buti Mohamad Al Suwaidi, Rana AlKatheeri, Fatmah Mohamed Khalfan Alblooshi, Meera Eisa Ali Hassan Almatrooshi, Mariam Eisa Hazeem Alzaabi, Reem Saif Al Darmaki, and Shamsa Nasser Ali Hussain Lootah. Global epidemiology of ischemic heart disease: Results from the global burden of disease study. *Cureus*, 12(7):e9349, July 2020.

[38] Jon Kleinberg. An impossibility theorem for clustering. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.

[39] Soheil Kolouri, Se Rim Park, Matthew Thorpe, Dejan Slepcev, and Gustavo K. Rohde. Optimal mass transport: Signal processing and machine-learning applications. *IEEE Signal Processing Magazine*, 34(4):43–59, 2017.

[40] Sebastian Krittian, Stefan Schmitt, Christoph Lorenz, Michael A. Brockmann, and Tobias Preusser. Prediction of 3d cardiovascular hemodynamics using fluid-structure interaction models. *Journal of Computational Surgery*, 1(1):97–107, 2011.

[41] Thomas Kurbiel and Shahrzad Khaleghian. Training of deep neural networks based on distance measures using rmsprop, 2017.

[42] Pierre Lahoud, Reinhilde Jacobs, Seyed Ali Elahi, Maxime Ducret, Wout Lauwers, G. Harry van Lenthe, Raphaël Richert, and Mostafa EzEldeen. Developing advanced patient-specific in silico models: A new era in biomechanical analysis of tooth autotransplantation. *Journal of Endodontics*, 50(6):820–826, 2024.

[43] Hao Liu, João S. Soares, John Walmsley, David S. Li, Samarth Raut, Reza Avazmohammadi, Paul Iaizzo, Mark Palmer, Joseph H. Gorman III, Robert C. Gorman, and Michael S. Sacks. The impact of myocardial compressibility on organ-level simulations of the normal and infarcted heart. *Scientific Reports*, 11:13466, 2021.

[44] Gabriel Loaiza-Ganem, Brendan Leigh Ross, Rasa Hosseinzadeh, Anthony L Caterini, and Jesse C Cresswell. Deep generative models through the lens of the manifold hypothesis: A survey and new connections. *arXiv preprint arXiv:2404.02954*, 2024.

[45] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery.

[46] Marcel Lüthi, Thomas Gerig, Christoph Jud, and Thomas Vetter. Gaussian process morphable models. *IEEE transactions on pattern analysis and machine intelligence*, 40(8):1860–1873, 2017.

[47] C. Jud M. Lüthi, T. Gerig and T. Vetter. Gaussian process morphable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15 40, no. 8, pp. 1860-1873, 1 Aug. 2018r.

[48] F. Maes, D. Vandermeulen, and P. Suetens. Medical image registration using mutual information. *Proceedings of the IEEE*, 91(10):1699–1722, 2003.

[49] Emile Mathieu, Tom Rainforth, N. Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders, 2019.

[50] Thomas Maxwell, Boonthanome Nouanesengsy, Andy Bauer, Aashish Chaudhary, and John Patchett. Exploratory climate data visualization and analysis using dv3d and paraview in uv-cdat. *Kitware Source*, 04 2013.

[51] Marina Meilă and Hanyu Zhang. Manifold learning: What, how, and why. *Annual Review of Statistics and Its Application*, 11, 2024.

[52] Loc Nguyen. Tutorial on em algorithm. 2020.

[53] Frank Nielsen. An elementary introduction to information geometry. *Entropy*, 22(10):1100, 2020.

[54] Frank Nielsen. The many faces of information geometry. *Not. Am. Math. Soc*, 69(1):36–45, 2022.

[55] Tiago Novello, Vinícius da Silva, and Luiz Velho. Visualization of nil, sol, and sl2(r)˜ geometries. *Computers Graphics*, 91:219–231, 2020.

[56] David Oks, Cristóbal Samaniego, Guillaume Houzeaux, Constantine Butakoff, and Mariano Vázquez. Fluid–structure interaction analysis of eccentricity and leaflet rigidity on thrombosis biomarkers in bioprosthetic aortic valve replacements. *International Journal for Numerical Methods in Biomedical Engineering*, 38(12):e3649, 2022.

[57] Jian-Xin Pan, Kai-Tai Fang, Jian-Xin Pan, and Kai-Tai Fang. Maximum likelihood estimation. *Growth curve models and statistical diagnostics*, pages 77–158, 2002.

[58] V.M. Panaretos and Y. Zemel. *An Invitation to Statistics in Wasserstein Space*. Creative Media Partners, LLC, 2020.

[59] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[60] Mauricio Ramirez, G. Pignata, Francisco Förster, Santiago Gonzáles-Gaitán, Claudia Gutiérrez, B. Ayala, Guillermo Cabrera-Vives, Márcio Catelan, Alejandra Muñoz Arancibia, and Jonathan Pineda. A novel optimal transport-based approach for interpolating spectral time series: Paving the way for photometric classification of supernovae, 09 2024.

[61] Sudhir Rao. Unsupervised learning: An information theoretic framework. 2008.

[62] Harvey Ray, Hanspeter Pfister, Deborah Silver, and Todd A Cook. Ray casting architectures for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):210–223, 1999.

[63] Alhusseini MI Fazal M Tang S Roney CH Rogers AJ Lee A Wang PJ Clopton P Rubin DL Narayan SM Niederer S Baykaner T. Razeghi O, Kapoor R. Atrial fibrillation ablation outcome prediction with a machine learning fusion framework incorporating cardiac computed tomography. 01 2023.

[64] Michal Rolinek, Dominik Zietlow, and Georg Martius. Variational autoencoders pursue pca directions (by accident), 2019.

[65] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[66] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[67] D. Rueckert, L.I. Sonoda, C. Hayes, D.L.G. Hill, M.O. Leach, and D.J. Hawkes. Non-rigid registration using free-form deformations: application to breast mr images. *IEEE Transactions on Medical Imaging*, 18(8):712–721, 1999.

[68] Santiago Sanchez, Eduardo V. Garcia, Jose Luis Serna, Eric W. Remme, and Thor Edvardsen. The impact of myocardial fibrosis on left ventricular function during acute coronary syndromes: insights from cardiovascular magnetic resonance imaging. *Journal of Cardiovascular Magnetic Resonance*, 17(1):1–10, 2015.

[69] William Schroeder, K. Martin, and William Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics*. 01 2006.

[70] Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, 1986.

[71] Sandeep Singh Sengar, Affan Bin Hasan, Sanjay Kumar, and Fiona Carroll. Generative artificial intelligence: A systematic review and applications. *arXiv preprint arXiv:2405.11029v1*, 2024.

[72] Daanish Mohammed Shariff, H Abhishek, D Akash, et al. Artificial (or) fake human face generator using generative adversarial network (gan) machine learning model. In *2021 fourth international conference on electrical, computer and communication technologies (ICECCT)*, pages 1–5. IEEE, 2021.

[73] Joshua Steyer, Lourdes Patricia Martínez Díaz, Laura Anna Unger, and Axel Loewe. Simulated excitation patterns in the atria and their corresponding electrograms. In *Functional Imaging and Modeling of the Heart*, 2023.

[74] Julian Suk, Christoph Brune, and Jelmer M. Wolterink. Se(3) symmetry lets graph neural networks learn arterial velocity estimation from small datasets. In *Functional Imaging and Modeling of the Heart (FIMH)*, pages 445–454. Springer Nature, 2023.

[75] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

[76] Michalis Titsias and Neil D Lawrence. Bayesian gaussian process latent variable model. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 844–851. JMLR Workshop and Conference Proceedings, 2010.

[77] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders, 2019.

[78] Eric J. Topol, Francisco Lopez-Jimenez, and Andrea N. DeMaria. Artificial intelligence in the diagnosis and management of disease: Overview and recommendations. *Journal of the American College of Cardiology*, 73(17):2137–2152, 2019.

[79] Roshan Reddy Upendra, Richard Simon, Suzanne M. Shontz, and Cristian A. Linte. Deformable image registration using vision transformers for cardiac motion estimation from cine cardiac mri images. In *Functional Imaging and Modeling of the Heart: 12th International Conference, FIMH 2023, Lyon, France, June 19–22, 2023, Proceedings*, page 375–383, Berlin, Heidelberg, 2023. Springer-Verlag.

[80] Mariano Vázquez, Guillaume Houzeaux, Seid Koric, Antoni Artigues, Jazmin Aguado-Sierra, Ruth Arís, Daniel Mira, Hadrien Calmet, Fernando Cucchietti, Herbert Owen, et al. Alya: Multiphysics engineering simulation toward exascale. *Journal of computational science*, 14:15–27, 2016.

[81] Marco Viceconti, Francesca Pappalardo, Baudouin Rodriguez, Matthew Horner, M. Bischoff, and Lara Montoya. In silico trials: Verification, validation and uncertainty quantification of predictive models used in the regulatory evaluation of biomedical products. *Methods*, 94:146–160, 2016.

[82] C. Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003.

[83] John Voight. *Hyperbolic plane*, pages 605–627. 06 2021.

[84] Shinelle Whiteman, Yusuf Alimi, Mark Carrasco, Jerzy Gielecki, Anna Zurada, and Marios Loukas. Anatomy of the cardiac chambers: A review of the left ventricle. *Translational Research in Anatomy*, 23:100095, 2021.

[85] R Willems, EKPM Kruithof, KLPM Janssens, MJM Cluitmans, O van der Sluis, PHM Bovendeerd, and CV Verhoosel. Isogeometric-mechanics-driven electrophysiology simulations of ventricular tachycardia. *Lecture Notes in Computer Science*, 13958:97–106, 2023.

[86] Amir Zadeh, Yao-Chong Lim, Paul Pu Liang, and Louis-Philippe Morency. Variational auto-decoder: A method for neural generative modeling from incomplete data, 2021.

[87] Shunkang Zhang, Yuan Gao, Yuling Jiao, Jin Liu, Yang Wang, and Can Yang. Wasserstein-wasserstein auto-encoders. *CoRR*, abs/1902.09323, 2019.

[88] Dominik Zietlow, Michal Rolinek, and Georg Martius. Demystifying inductive biases for β-vae based architectures, 2021.