University of Cape Town

Modelling non-linearity in 3D shapes: A comparative study of Gaussian process morphable models and variational autoencoders for 3D shape data

Student:
Fabio Fehr
FHRFAB001

Supervisor:
Mr Allan Clark
Co-supervisor:
Assoc. Prof. Tinashe Mutsvangwa

Dissertation for the degree
Masters in Advanced Analytics

Department of Statistical Sciences

Division of Biomedical Engineering

June 24, 2021

Abstract

The presence of non-linear shape variation in 3D data is known to influence the reliability of linear statistical shape models (SSM). This problem is regularly acknowledged, but disregarded, as it is assumed that linear models are able to adequately approximate such non-linearities. Model reliability is crucial for medical imaging and computer vision tasks; however, prior to modelling, the non-linearity in the data is not often considered. The study provides a framework to identify the presence of non-linearity in using principal component analysis (PCA) and autoencoders (AE) shape modelling methods. The data identified to have linear and non-linear shape variations is used to compare two sophisticated techniques: linear Gaussian process morphable models (GPMM) and non-linear variational autoencoders (VAE). Their model performance is measured using generalisation, specificity and computational efficiency in training. The research showed that, given limited computational power, GPMMs managed to achieve improved relative generalisation performance compared to VAEs, in the presence of non-linear shape variation by at least a factor of six. However, the non-linear VAEs, despite the simplistic training scheme, presented improved specificity generative performance of at least 18% for both datasets.

Acknowledgements

I would like to thank Mr Allan Clark and Associate Professor Tinashe Mutsvangwa for the supervision and guidance. I also would like to thank Dr Marcel Lüthi for his technical support. Finally, I would like to acknowledge my friends and family for their unwavering support.

Contents

Abstract i

Acknowledgements ii

Acronyms v

Glossary vi

1 Literature Review Chapter 1
- 1.1 Introduction . . . 1
- 1.2 Nature of shape spaces . . . 4
  - 1.2.1 Higher order non-linearity . . . 6
  - 1.2.2 Articulation non-linearity . . . 7
  - 1.2.3 Rotational non-linearity . . . 8
- 1.3 Shape modelling techniques . . . 9
  - 1.3.1 Linear shape modelling . . . 10
  - 1.3.2 Non-linear shape modelling . . . 11
- 1.4 Shape model evaluation . . . 14

2 Aims and Objectives Chapter 16

3 Theoretical Background Chapter 17
- 3.1 Multivariate Gaussian Distribution . . . 17
- 3.2 Principal Component Analysis . . . 18
- 3.3 Singular Value Decomposition . . . 20
- 3.4 Point Distribution Models . . . 21
- 3.5 Autoencoders . . . 23
  - 3.5.1 Definition . . . 23
  - 3.5.2 Relationship to PCA . . . 23
  - 3.5.3 Parameters and hyperparameters . . . 24
    - Initialisation . . . 24
    - Epochs . . . 24
    - Batch size . . . 25
    - Layers . . . 25
    - Loss function . . . 25
    - Regularisation . . . 25
    - Optimisers . . . 26
    - Learning rate . . . 27
    - Batch Normalisation . . . 27
    - Activation functions . . . 27

3.6 Variational Autoencoders . . . 28
3.6.1 Definition . . . 29
3.6.2 Kullback-Liebler divergence . . . 31
3.6.3 Variational inference . . . 32
3.6.4 Reparameterisation trick . . . 33
3.7 Gaussian Process Morphable Models . . . 34
3.7.1 Definition . . . 34
3.7.2 Shape modelling with Gaussian Processes . . . 35
3.7.3 Kernel functions . . . 37
3.7.4 Nyström approximation . . . 38
3.8 Generalised Procrustes Analysis . . . 39
3.9 Shape Model Metrics . . . 40
3.9.1 Generalisation . . . 41
3.9.2 Specificity . . . 42

4 Data Chapter **43**
4.1 Shape data preprocessing . . . 44
4.2 Femur data . . . 45
4.3 Human body data . . . 46

5 Computation Specifications Chapter **48**
5.1 Hardware . . . 48
5.2 Software . . . 48

6 Identifying non-linearity Chapter **50**
6.1 Linear model equivalence . . . 50
6.2 Non-linear model exploration . . . 57

7 Modelling Chapter **61**
7.1 GPMM . . . 62
7.1.1 Experiment 1: Baseline GPMM . . . 63
7.1.2 Experiment 2: Kernel GPMM . . . 64
7.1.3 Experiment 3: Augmented GPMM . . . 68
7.1.4 Conclusion . . . 71
7.2 VAE . . . 72
7.2.1 Experiment 1: Single layer VAE . . . 73
7.2.2 Experiment 2: Deep VAE . . . 75
7.2.3 Experiment 3: Activations, Initialisations and Epochs . . . 80
7.2.4 Conclusion . . . 85
7.3 Model comparison . . . 86

8 Conclusion Chapter **88**

Bibliography **91**

Acronyms

ACM	Active Contour Models.
AE	Autoencoder.
ASM	Active Shape Models.
CNN	Convolutional Neural Network.
ELBO	Evidence Lower Bound.
ELU	Exponential Linear Units.
GAN	Generative Adversarial Network.
GPA	Generalised Procrustes Analysis.
GPMM	Gaussian Process Morphable Model.
HD	Hausdorff distance.
KL	Kullback-Liebler.
LOOCV	Leave-one-out cross-validation.
MCMC	Markov Chain Monte Carlo.
MM	Morphable Model.
MRI	Magnetic Resonance Imaging.
MSE	Mean Squared Error.
PCA	Principal Component Analysis.
PDM	Point Distribution Model.
PGA	Principal Geodesic Analysis.
ReLU	Rectified Linear Units.
RIMD	Rotation Invariant Mesh Difference.
SSM	Statistical Shape Modelling.
SVD	Singular Value Decomposition.
Tanh	Hyperbolic Tangent.
VAE	Variational Autoencoder.
VI	Variational Inference.

Glossary

Autoencoder A connected neural network architecture used to perform an identity mapping between input and output.

Backpropagation An algorithm that describes how the error of the neural network is propagated backwards through the layers of the network and used to update the weights.

Compactness A model metric of the variability explained using eigenvalues.

Correspondence Correspondence between two shapes maps every point on the first shape to its semantic equivalent point on the second shape.

Deformation The change in position of a point or collection of points on the shape.

Eigenfunction A non-zero linear operator that returns an eigenvector.

Eigenvalue A scaling factor for the eigenvector in a linear transformation.

Eigenvector A non-zero basis vector used in linear transformations.

Epoch One full parse of the training data through the model is one epoch.

Euclideanization Transforms the shape into a Euclidean feature space such that linear techniques are more appropriate.

FAUST An open source, human body 3D mesh dataset in correspondence. Named after the correspondence procedure Fine Alignment Using Scan Texture.

Gaussian distribution A symmetric, bell-shape distribution defined by its mean and covariance. Can be univariate for a single random variable or multivariate for a collection of random variables.

Gaussian process	A non-parametric, continuous representation of a multivariate Gaussian distribution, defined by a mean and covariance function.
Gaussian process morphable model	A continuous generalisation of a point distribution model using Gaussian processes.
Generalisation	The generalisation ability of a model measures its capability to represent unseen instances of the object class.
Hausdorff distance	Measures the maximum error, or greatest distance between two points or shapes.
Hyperparameters	An external configuration of variables used to alter a model. Hyperparameters are set through manual approaches.
Karhunen-Loeve expansion	A linear expansion which allows for dimension reduction and seen as a continuous analog for principal component analysis.
Kernel function	A positive semi-definite function that defines the covariance structure of data.
Kullback-Leibler divergence	A similarity measure between distributions that uses the expected value of the difference of information.
Landmark points	A point in correspondence that marks a specific part of the contour or structure of objects within the same shape class.
Manifold	A manifold is a topological space that locally resembles a Euclidean space.
Mesh	A point set with connectivity information.
Modes of variation	Basis vectors obtained when projecting the shape down into the tangent space. Often displayed by scaling shape parameters between Gaussian bounds.
Morphable model	An expansion on the point distribution model and precursor to the Gaussian process morphable model.
Parameters	An internal configuration of variables used to alter a model. Parameters are set through initialisation and optimisation.
PLY	Stanford Polygon Format - A file format for 3D surface geometry.
Point distribution model	A model that captures the variation of corresponding points between shapes in d-dimensional space.

Principal component analysis	A linear, multivariate, statistical technique that reduces the dimensionality of data. The variation in the data is extracted by a linear, orthogonal projection onto new variables, known as principal components.
Reference shape	An arbitrarily selected points domain from the dataset used as a reference for all other point variation.
Registration	The process to determine the transformation, or mapping, that relates points on a shape, to corresponding points on other shapes.
Segmentation	The process of detecting an object within a space and delineating its boundary.
Shape	An object which is independent of locational, rotational and scaling geometric information.
Shape descriptors	New features that represent the shape in a concise and informative format.
Shape parameters	The shape observations are linearly projected onto a new rotated coordinate frame known as the tangent space. The shape parameters define the observations in the tangent space.
Shape space	A multidimensional, curved manifold where objects of the same shape family will lie in close proximity.
Singular value decomposition	A mathematical factorisation of any matrix into three orthogonal matrices namely; left, right singular matricies and a diagonal matrix containing the singular values.
Specificity	A specific model should only generate instances of the object class that are similar to those in the training set.
Statistical shape model	A model that captures the variation between shapes using statistical methods.
STL	Stereolithography Format - A file format for 3D surface geometry.
Tangent space	A lower dimensional, orthogonal space tangent to the shape space that provides a linear approximation.
Variational autoencoder	A autoencoder model which encodes the inputs as a distribution over the latent space instead of arbitrary points.
Variational bayesian inference	An optimisation process used to find the best parameters for given distribution family that fits a target distribution.

1 | Literature Review

1.1 Introduction

Statistical shape modelling (SSM) is not a modern concept, but, due to the rise in accessible computational power and data-driven techniques, it has regained prominence in recent literature (Heimann and Meinzer, 2009; Brunton et al., 2014; Lüthi et al., 2018; Fouefack et al., 2020). Cootes, Baldock, and Graham, 2000 defines a shape in d-dimension(s) to be an object which is independent of locational, rotational and scaling geometric information. For example; suppose a square in 2-dimensions (2D), or cube in 3-dimensions (3D), is rotated around an axis, all features are shifted in the same direction and uniformly scaled which shrink or enlarge the shape. The new object is still part of the same shape family namely, a square or cube, respectively (Cootes, Baldock, and Graham, 2000). This is a widely accepted definition. However, in applications such as biomedical image analysis, the size of the object is a salient feature. In such applications, scale may be included in the definition of shape (Dryden and Mardia, 2016).

Statistical shape modelling is used to model the variation between shapes within the same shape family. This is particularly useful for anatomical shapes which do not have predefined mathematical representations. Statistical shape modelling has wide application and has traditionally been used for: facial recognition tasks (Blanz and Vetter, 1999), gait analysis (Sminchisescu and Jepson, 2004) and medical image analysis (Heimann and Meinzer, 2009). In more recent literature, SSM has been used in computer vision (Metaxas, 2012), mechanical engineering (Omrani et al., 2016), molecular biology (Gao et al., 2016b) and dentistry (Wu et al., 2016a).

Heimann and Meinzer, 2009 argues that the most common use for shape modelling in both 2D and 3D data is shape segmentation. This entails detecting an object within an image and delineating its boundary using a shape model. For instance, finding a face within a 2D image or clustering signature values - such as the eyes, nose and mouth points - from a 3D face mesh model (Williams and Ilieş, 2018). A notable application of segmentation is in medical image analysis. Segmentation is used to create 3D anatomical reconstructions of organs from different medical imaging modalities (Zhou et al., 2014).

Determining a point-to-point correspondence between two shapes is another useful application of shape modelling. Establishing correspondence requires mapping every point on a shape to its semantically equivalent point on another shape (Lüthi et al., 2018). While critical for explaining image data with shape models, determining correspondence is a challenging problem. A popular approach for establishing correspondence is through registration. It can be separated into two parts namely rigid alignment and non-rigid alignment. The former rigidly transforms and rotates the

shapes to a position of best fit without deformation. In contrast, non-rigid alignment is a non-uniform mapping between shapes (or images) which requires deformations of an elastic nature, stretching localized portions of the shape (Van Kaick et al., 2011). Registration is a core problem in both computer vision and medical imaging. Surveys by Van Kaick et al., 2011 and Tam et al., 2012 discuss in detail possible approaches to finding point-to-point correspondence for shape.

Similar to segmentation, shape modelling has also been used extensively for recognition tasks. Kass, Witkin, and Terzopoulos, 1988 made a fundamental contribution in the development of active contour models (ACM). The novel use of energy minimizing splines - used to detect edges and lines in images - coined ACM the analogy "Snakes". Cootes, Baldock, and Graham, 2000 extended the aforementioned ideas and contributed significantly to recognition using active shape models (ASM) - also referred to as "Smart Snakes". Active shape models provide an efficient fitting algorithm specialised to mimic the variation seen in training example images. Cootes, Baldock, and Graham, 2000 explains that a point distribution model (PDM) performs recognition tasks through learning the distribution of points. The variance between points is captured using a linear technique known as principal component analysis (PCA). Point distribution models are convenient for medical imaging and modelling human body movements due to their wide application, computational efficiency in training and minimal assumptions. However, the models lack flexibility when fitting to target data that is atypical to the training set (Cootes, Baldock, and Graham, 2000).

Blanz and Vetter, 1999 expanded upon the PDM framework by developing a morphable model (MM). The model was novel due to its ability to generate a 3D face mesh from a single 2D portrait by capturing both shape and texture. Due to their generative ability to morph between different objects, MMs have application in computer vision, animation, face synthesis and facial recognition. A shortcoming of MMs, described by the authors, is that their face model lacked flexibility to generalise to different ages or races, unseen in the training data (Blanz and Vetter, 1999). Lüthi et al., 2018 adapted this model by using Gaussian Processes to artificially add flexibility to any discretisation of points; this modern framework is called Gaussian process morphable models (GPMMs). This adapted modelling framework addressed a shortfall of MMs, through adding a principled manner to increase model flexibility, and requires minimal data to train. Moreover, GPMMs offered additional improvements in efficient training times and realistic shape generation.

Gaussian process morphable models, much like their precursors, use linear dimension reduction techniques. The aforementioned methods require the data both to be aligned and in-correspondence, prior to implementation. The alignment allows all coordinates to be relative to one another and places each shape on a multidimensional, curved manifold known as shape space (Klingenberg, 2020). Both Bowden, 2000 and Klingenberg, 2020 suggest that it is fair to assume that shapes within the same family should lie in close proximity in shape space, but one cannot assume that the multidimensional manifold is always locally smooth. This has implications for linear dimension reduction techniques as they perform an orthogonal projection from the non-linear shape space to a linear tangent space. Klingenberg, 2020 makes the analogy that the tangent space approximation to the shape space is comparable to the flat map approximation of our spherical earth surface. As the poles (Antarctica and Greenland) are distorted in size, the observations that are far from the mean shape will be distorted. When large variation is present between shapes or the shape

space is not locally smooth the orthogonal linear projection can lead to unreliable or unstable models (Bowden, Mitchell, and Sarhadi, 2000; Klingenberg, 2020).

Sozou et al., 1994 identified and visualised the non-linear shape space problem through a dependent, curved relationship in the tangent space. The relationship in tangent space is caused by variation present in the shapes. The orthogonal projection is unable to account for the dependent relationship and causes distortions to the shapes when modelled. The proposed solution was to use polynomial regression in the tangent space to account for the distortion caused by the projection. The adjustment allows for PDMs to deal with modes of variation which can move along polynomial paths. As a result, the points on an object surface were able to follow polynomials paths rather than linear paths as the shape parameters were varied (Sozou et al., 1994). The general use of polynomial regression in this context was limited. It performed poorly when the tangent space did not display dependent relationships and required the polynomial degree parameter to be specified beforehand. Sozou et al., 1997 furthered their approach by considering a multilayer perceptron framework, also known as an Autoencoder (AE). The aforementioned method addresses the shortcomings of a linear model's shape space projection. However, generative deep learning models requires a considerable amount of data to train and the training process is computationally expensive (Sozou et al., 1997).

Recent literature has seen a rise in the use of deep learning based shape models (Masci et al., 2015; Wu et al., 2016b; Tran and Liu, 2018; Tan et al., 2018). These models have been criticized for: requiring large amounts of data, computational inefficiencies and poor interpretability (Fouefack et al., 2020). Kingma and Welling, 2013 made a notable contribution by reformulating the AE network to integrate variational Bayesian inference, hence known as the Variational Autoencoder (VAE). This extension allowed the VAE to learn the parameters of a probability distribution. The additional distributional assumptions - similar to GPMMs - addresses the pitfalls of deep learning in interpretability, computationally efficiency in training and data requirements (Kingma and Welling, 2013).

Variational autoencoders augur well for statistical shape analysis. Nash and Williams, 2017 used a shape VAE on segmented, man-made shape objects and contrasted this method against linear alternatives. They found that VAE outperformed the linear alternatives on shapes with complex variation, such as the variation between aeroplane wing structures. They concluded that this method was computationally efficient in training, sampling and easy to scale to high dimensional data (Nash and Williams, 2017). Tan et al., 2018 expanded on the work of Nash and Williams, 2017 and applied a VAE to non-segmented anatomical shapes. The VAE was able to generate plausible new shapes with rich details from few samples.

Studies have alluded to the "non-linearity problem" and deemed it to be non-salient (Heimann and Meinzer, 2009; Brunton et al., 2014; Lüthi et al., 2018; Fouefack et al., 2020). However, the quality of the tangent space approximation is dependent on the scale of variation in the data (Klingenberg, 2020). Klingenberg, 2020 argues that the distortions caused by linear methods are only non-salient if the dataset occupy a small, sufficiently smooth neighbourhood of the shape space around the mean shape (Le and Kendall, 1993; Kendall et al., 2009; Small, 2012; Abdi and Williams, 2010).

Section 1.2 outlines the nature of shape spaces. Thereafter, the history of linear and non-linear shape modelling techniques shall be reviewed in Section 1.3. Finally, in

Section 1.4 the typical performance metrics and the evaluation of shape models will be defined.

1.2 Nature of shape spaces

The shape space is an abstract representation where each possible shape in the shape family is represented as a point on a multidimensional, non-linear manifold (Klingenberg, 2020). As mentioned, it is assumed that shapes coming from the same family should be in close proximity in shape space. Klingenberg, 2020 further explains that shape space is unique to the family of shapes and represents all the variation present in that class. The distance between points in shape space is relative to the magnitude of the difference between shapes (Klingenberg, 2020). Shape spaces are imperative as they allow for statistical operations such as estimating the mean and variation between shapes (Kendall et al., 2009).

Klingenberg, 2020 provided an intuitive visualisation of shape space. Figure 1.1 presents the shape space for 2D triangles as a sphere, displaying the northern hemisphere. An equilateral triangle is shown in at the north pole of the sphere, with six meridian lines showing isosceles triangles. The equator line displays collinear triangles and acts as a line of symmetry as the southern hemisphere mirrors the triangles seen in the northern hemisphere (Klingenberg, 2016).

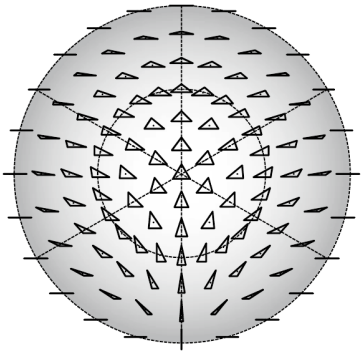

Figure 1.1: Shape space visualisation for triangles in 2D (Klingenberg, 2016). The figure presents the northern hemisphere of the spherical shape space where the "north pole" displays an equilateral triangle in the centre. The dashed perimeter of the sphere describes collinear triangles, where all three vertices lie on a straight line and the six vertical meridian lines show isosceles triangles.

Typically, shape spaces are multidimensional and difficult to visualise. Klingenberg, 2020 was able to visualise the shape space as the dimension of the shape space for triangles is 2D. There are three landmarks with two coordinates each which results in six dimensions. Four dimensions are subtracted due to the degrees of freedom lost during standardisation. One for scale, two for position and one for rotation $2N - 4$ (Klingenberg, 2016). In 3D, the dimensions are defined by the number of landmarks multiplied by three coordinates. The degrees of freedom lost are now: one for scale, three for position and three for rotation around all axes $3N - 7$ (Klingenberg, 2016). It is clear that as the number of landmark points increase the shape space quickly becomes multidimensional, non-linear and cumbersome for computations (Klingenberg, 2020).

The tangent space provides a local approximation to the shape space. Figure 1.2 presents the tangent space projection for 2D triangles, which is akin to the map-earth analogy previously mentioned (Klingenberg, 2016). In general, studies of shape variation orthogonally project the shape space into a flat tangent space. Tangent spaces have the same dimensions as their shape spaces. However, it is a local, linear approximation based around the mean shape which provides a space conducive for computations and statistics (Klingenberg, 2020). A set of orthogonal basis functions are used to project data into the tangent space. In shape modelling, these linear basis functions are known as modes of variation. The projected observations form the shape parameters of the tangent space. Shape parameters are assumed to be generated by probability distributions hence, used for statistical inference (Brunton et al., 2014).

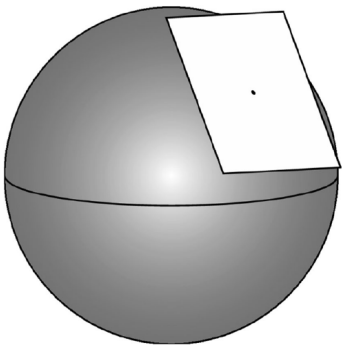

Figure 1.2: The tangent space approximation to shape space for 2D triangles (Klingenberg, 2016). The flat projection is tangent to the shape space at a reference shape commonly defined to be the mean shape. This provides a locally linear approximation to the shape space in the surroundings of this point.

Orthogonal modes of variation restrain point deviation in linear shape models. These models assume landmark variations are Gaussian distributed which forms a hyper-elliptical cluster in d-dimensional space around the mean. Figure 1.3 displays how a linear model's projection restricts the 3D landmark variations to lie within an elliptical cluster using orthogonal basis vectors. The average shape is shown as $\bar{x}$ in the centre of the modes of variation v_1, v_2, v_3.

Both Bowden, 2000 and Klingenberg, 2020 concur that large landmark variations can cause poor approximations when using linear models. Bowden, 2000 argues that orthogonal basis vectors would not be able to accurately capture landmark variations which form non-Gaussian clusters. In modern literature, linear shape analysis methods are justified by their simplicity, interpretability and arguably close approximations (Foueffack et al., 2020). Nevertheless, the quality of the linear approximation is determined by the non-linearity present in the data (Brunton et al., 2014). Pizer and Marron, 2017 identified that non-linearity in the data is caused by: poor correspondences between shapes, shapes variations which articulate (bend in an angular movement) and shapes which have a pivotal rotation variation about an axis. These types of deformations would cause large landmark variations resulting in poor linear projections from the shape space. Therefore, a shape linearity analysis should be conducted before model selection.

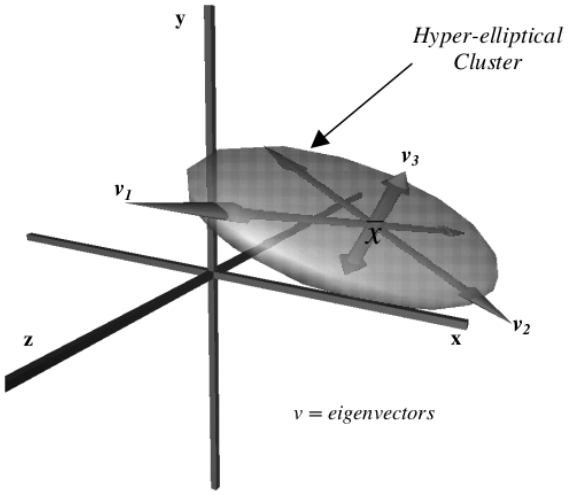

Figure 1.3: Hyper-ellipsoid in 3D space displaying the behaviour of orthogonal basis vectors. These vectors are used to project the data from shape space to the tangent space (Bowden, 2000).

1.2.1 Higher order non-linearity

Higher order non-linearity is caused by poor correspondences which is usually the result of incorrect labeling of data points. Tam et al., 2012 explains that point-to-point registration is typically performed using automated algorithms; however, erroneous labelling may still occur. Brunton et al., 2014 remarks that the quality of the registration directly influences the quality of the shape model. Higher order non-linearity can be mitigated by semi-automated approaches where a trained user carefully selects salient points on the boundary such as the nose tip in facial registration or joints centres in human movement analysis.

Figure 1.4 illustrates an example of higher order non-linearity caused by resampling on a 2D contour with two peaks. The black circle marks the highest point of the contour. The red and yellow circles mark the midpoints between the beginning and end of the contour, respectively. Figure 1.4 shows that the highest point can become inconsistent at shape (c) which causes poor correspondence between shapes.

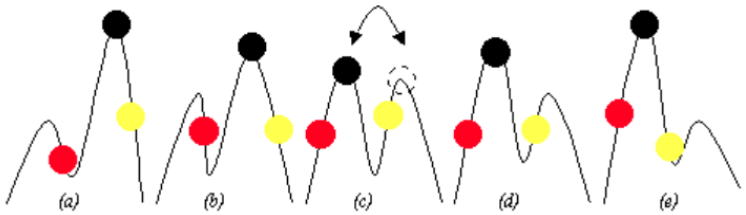

Figure 1.4: Higher order non-linearity caused by incorrect labeling of points on a two peaked, contour shape (Bowden, 2000). When considering shape (c) the peak orientation becomes inconsistent due to poor correspondence of the peaks.

This problem has been known to manifest itself as discontinuous clusters in tangent space. Furthermore, Bowden, 2000 found that higher order non-linearity results in linear models requiring more basis vectors to capture the deformations. In the next section we will consider another cause of non-linearity as a result of bending shape variations.

1.2.2 Articulation non-linearity

Shapes with curved bending variations cause non-linearity in the tangent space projections (Sozou et al., 1994; Bowden, Mitchell, and Sarhadi, 2000; Pizer and Marron, 2017). Sozou et al., 1994 used real chromosome data and artificial tadpole data to demonstrate this phenomenon. Both datasets contain thin worm structures that articulate back and forth illustrated in Figure 1.5.

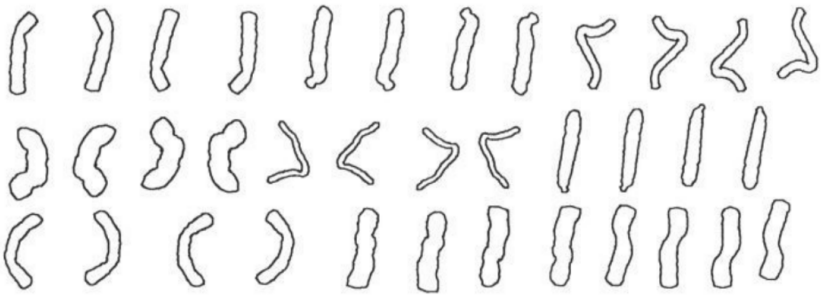

Figure 1.5: An example of real 2D chromosome shape data with non-linearity through pivoting about their centres. (Sozou et al., 1994).

Sozou et al., 1994 identified non-linearity in the chromosome data through non-linear dependencies in the tangent space. As a result, the modes of variation allow for distortions not seen in the training data. As seen before in Figure 1.3, linear techniques require the landmark variations to be small and Gaussian distributed about the mean, to be effectively modelled. Figure 1.6 illustrates the non-linear relationship, which resembles a second degree polynomial in the tangent space.

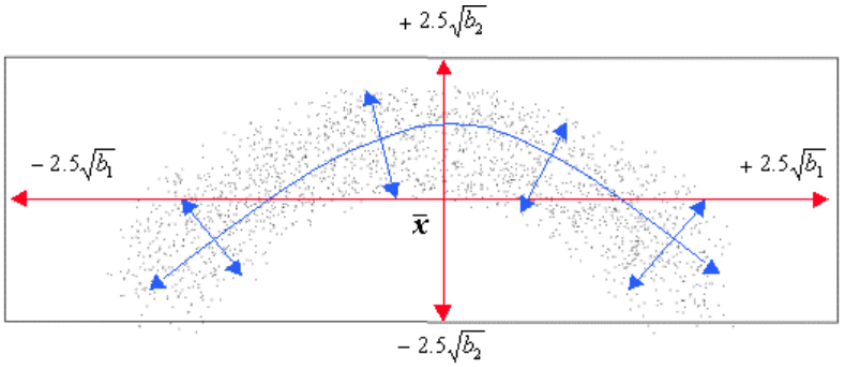

Figure 1.6: A scatter plot of shape parameters is presented. The non-linear dependency in tangent space is caused by bending variations between shapes. The basis vectors for linear (red) and non-linear (blue) techniques are overlayed (Bowden, 2000).

Figure 1.6 displays the first two sets of linear projections. The shape parameters are denoted by b_1, b_2 and are plotted on the x-axis and y-axis, respectively. The red orthogonal basis vectors are represented within Gaussian standard deviation bounds $[-2.5, 2.5]$ about the mean shape $\bar{\mathbf{x}}$. Notice that the mean does not lie within the centre of the shape parameter cluster and that the orthogonal vectors do not capture the true variation in the tangent space. These discrepencies prove that a linear combination of basis vectors can produce deformations not seen in the training set. Moreover, the non-linear polynomial PDM in blue is clearly better suited to the data.

Sozou et al., 1994 demonstrated that a polynomial regression PDM can successfully model curvature in the tangent space in comparison to linear techniques. However, Bowden, 2000 identified that this method suffers when modelling discontinuities. Another weakness of the polynomial regression PDM is in modelling non-linearity through rotational shape variations (Sozou et al., 1994). The next section elaborates further on rotational non-linearity.

1.2.3 Rotational non-linearity

Modern literature has emphasized the difficulty of modelling rotational non-linearity (Pizer and Marron, 2017; Lang, Kleinsteuber, and Hirche, 2018). Bowden, 2000 explored modelling rotational non-linearity with an artificial 2D robot arm dataset. Each robot arm, as seen in Figure 1.7, is constructed with three pivotal joints able to rotate in quarter circles. Large, rotational landmark variation is induced by design, mimicking complex human movement patterns (Bowden, 2000).

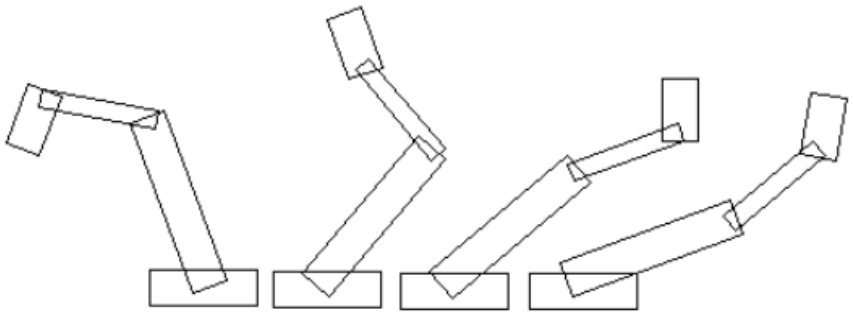

Figure 1.7: An example of the synthetic robot arm data with rotational non-linearity as a result of large, rotational movement. The robot arm comprises of three pivoting segments, which can rotate within a 90 degree radius, and a stationary base (Bowden, 2000).

Rotational non-linearity can be visualised through unrealistic distortions in the modes of variation. Figure 1.8 illustrates the first two modes of variation between Gaussian standard deviation bounds of the shape parameters. The first, primary mode captures the horizontal movement of the robot arm data. However, the positive and negative extremes of the primary mode display uncharacteristic distortions to the magnitude of the final segment. Similarly, the secondary mode of variation captures vertical movement, but portrays deformations of size and the inversion of the arm back upon itself (Bowden, 2000).

Sozou et al., 1997 contributed towards the identification of rotational non-linearity by developing a non-linear, multi-layer perceptron architecture that could successfully model rotational deformations in shape. This technique was trialed using a 2D clock face dataset which showed watch hands rotating around a central point. For contrast, the dataset was modelled initially using linear techniques which displayed a circular dependency pattern within the shape parameters. Similar to robot arm example, the modes of variation revealed unrealistic properties not seen in the data, such as extending and compressing the watch hands (Sozou et al., 1997).

The literature has concluded that modelling shape data - with the aforementioned non-linearity properties - using linear models can be unreliable. Modern linear shape

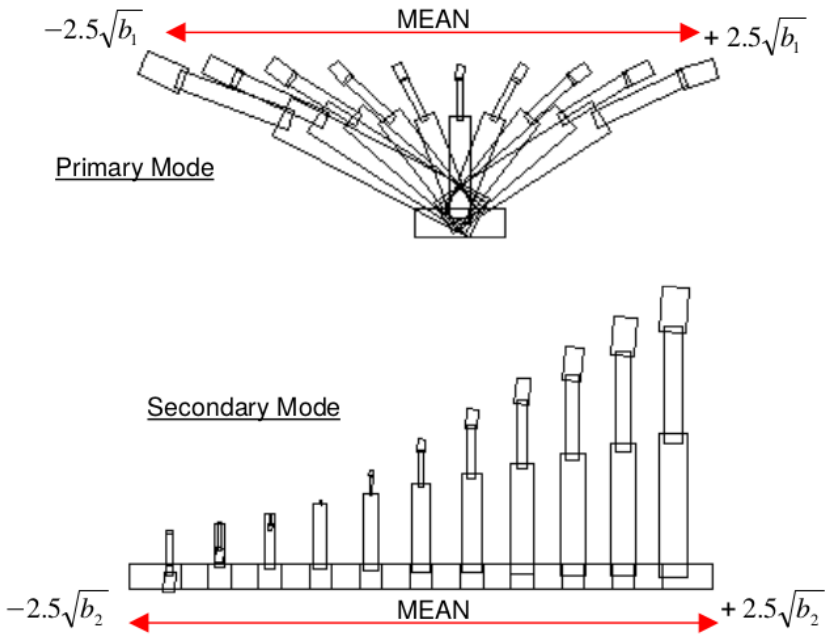

Figure 1.8: The primary and secondary modes of the robot arm datasets are displayed with clear distortions not seen in the training data (Bowden, 2000).

modelling techniques tend to disregard the non-linearity problem, the extent of which is dependent on the shape variation present in the data. The presence of non-linearity is identified through the following: visual rotations or bending deformations in the shapes; non-linear dependencies in the shape parameters; and unrealistic deformations in the modes of variation. This research will focus on 3D data of the human form as it has diverse shapes with large scale, non-linear landmark variations (Tan et al., 2018). The next section discusses the use of linear and non-linear shape modelling techniques in more detail.

1.3 Shape modelling techniques

Currently, numerous shape modelling paradigms exist, the most primitive of which is the use of a labeled example as a template or "golden shape". The template was adequate for rigid, non-anatomical shapes but the biological variation made it unsuitable for applications such as Magnetic Resonance Imaging (MRI) registration. Thereafter, simplistic "hand-crafted" models were used. These models are often single-use models and require complicated, time consuming construction (Yuille, Hallinan, and Cohen, 1992). Staib and Duncan, 1992 represented shape variation using Fourier descriptors, where the Fourier coefficients could alter the shape complexity. The approach was efficacious for closed boundary shapes, but lost popularity due to its inability to generalise to open boundary shapes.

Many alternative approaches were considered to model shapes such as: splines (Tsagaan et al., 2002), spherical harmonics (Kazhdan, Funkhouser, and Rusinkiewicz, 2003) and wavelets (Davatzikos, Tao, and Shen, 2003; Brunton et al., 2014). To discuss all of these methods in detail would be beyond the scope of the research. Thus, this study will focus on fitting 3D, generative techniques. The task of fitting a 3D shape model to ambiguous 3D data is important for many applications, such as recognition

(computer vision), human movement analysis (robotics), virtual avatar control (animation) and segmentation in surgical planning (medical imaging) (Brunton et al., 2014).

1.3.1 Linear shape modelling

Cootes et al., 1995 presented objects as sets of labelled points and examined the distribution of the corresponding points across all training shapes. In their approach, a reference shape is selected arbitrarily and all point variations, from the reference shape, are assumed to follow a multivariate Gaussian distribution. The use of point distribution models became increasingly popular as it is simple, greatly reduces the registration search space, allows probabilistic inference in model fitting, and is computationally efficient (Brunton et al., 2014). The classical statistical shape model, defined by Cootes, Baldock, and Graham, 2000, combined a PDM with PCA to reduce the dimensionality and capture the variation of corresponding points. As a result, any training object's shape can be approximated as a linear combination of the principal components, and new shapes could be generated by sampling from the shape parameters. This is a fundamental concept that laid the groundwork for many linear shape modelling techniques.

Blanz and Vetter, 1999 built on this by developing a novel linear model which was able to automatically generate 3D face shapes from images. The MM was able to uncover dense point-to-point correspondences between the vertices of the face and regulate unnatural faces through distributional assumptions. The model was able to transition smoothly between different expressions by adjusting shape parameters. Blanz and Vetter, 1999 manually parted their MM into four independent regions namely: eyes, nose, mouth and the surrounding head. These regions were fitted separately and then blended together afterwards, which achieved higher data accuracy than the global morphable model. Brunton et al., 2014 argues that part-based models are accurate as they manage to capture local and global deformations, but this only works well for applications where this partition is meaningful. The process of labeling part based data for medical imaging tasks is time consuming and difficult to define on anatomical shapes. Blending the boundaries between parts is challenging as regions are not necessarily continuous. Therefore, the study will focus on non-partitioned methods as they are widely applicable. Both MMs and classical PDMs share the same weakness of insufficient flexibility during model fitting. The models are restrained by the variation seen in the training sample. Moreover, Blanz and Vetter, 1999 used bootstrapping and optic flow algorithms to achieve improved correspondences on novel faces.

Lüthi et al., 2018 extended the morphable model to allow flexibility in model fitting. The MM framework was adapted by using a Gaussian process to model the distributions of an arbitrary discretisation of points. Training data exclusively defines the mean and covariance structure of a PDM and MM. In contrast, GPMMs are defined with using functions for the mean and covariance, allowing for flexibility to be added external of the data. Lüthi et al., 2018 explains that the covariance function, referred to as the kernel function, can be user specified and incorporate expert knowledge and non-linearity. Thus, the model can use kernel functions to augment the variation in the training data, allowing non-linear variation during model fitting (Wang et al., 2015).

A linear lower dimensional approximation is still achievable for GPMMs by using the Karhunen-Loeve expansion for Gaussian processes (Berlinet and Thomas-Agnan,

2011). Sampling shape parameters the Karhunen-Loeve expansion can generate new shape deformations from the reference domain. This requires calculating the equivalent eigenvalue and eigenvectors for a covariance kernel function. The eigenvalue and eigenvector calculation for user specified kernels are often not closed form, therefore the Nyström method for approximating eigenfunctions is used to get numerical approximations (Li, Kwok, and Lü, 2010). Gaussian distributions are prominent in shape modelling because of their Bayesian conjugacy property. This property allows for computational efficiency in model fitting due to the closed form solution of the posterior (Pizer and Marron, 2017). A stochastic data driven Markov chain Monte Carlo (MCMC) algorithm is used for GPMM model fitting which provides robust solutions through the use of probabilistic estimation (Schönborn et al., 2017).

Lüthi et al., 2018 successfully managed to add flexibility to linear PDMs in a generalisable, generative framework. The GPMM approach requires minimal data due to the use of kernel functions and ease of implementation through open source software Scalismo (`scalismo.org`). Gaussian process morphable models have performed well on local and global deformations in 3D modelling and medical imaging applications (Lüthi et al., 2018). Linear shape models are mathematically convenient and rely on the quality of the shape space linear projection. When forms of non-linearity are present, as a consequence of large landmark variation, the model's ability to represent natural biological variability is impaired (Huckemann, Hotz, and Munk, 2010; Zhang et al., 2015; Pizer and Marron, 2017). In the next section methods used to model shape non-linearity are explored.

1.3.2 Non-linear shape modelling

In the presence of non-linearity, non-linear shape models provide natural representations of variation. Heimann and Meinzer, 2009 confirms the need for non-linear models and argues that linear models inadequately approximate deformations such as bends and rotations. Non-linearity problems in shapes can be solved using two main approaches. The first transforms the shape into an Euclidean feature space, such that linear techniques may be more appropriate. This approach is known as Euclideanization (Pizer and Marron, 2017). The second approach uses non-linear shape models, which do not require Euclideanization.

Initial studies in shape Euclideanization transformed the data using log polar mappings, which removes non-linearity in shape space (Heap and Hogg, 1995). Pohl et al., 2006 expanded this idea in shape analysis by using LogOdds mapping. The mapping was used to embed the shape space into a linear LogOdds space, which improved the accuracy of the linear model for 3D medical imaging examples. A well-known method that used Euclideanization is the non-linear PCA analog Principal Geodesic Analysis (PGA) (Fletcher et al., 2004). They used geodesic distances between shapes, which calculates distance on the curve of a manifold, instead of Euclidean distances.

In recent studies, shape descriptors include Euclideanization. Shape descriptors are the representation of the shape features. Rostami et al., 2019 argues that using the 3D vertex position, or point cloud, shape descriptor exclusively is naive. Shape descriptors allow for shape variation to be modelled independently of posture (Wuhrer, Shu, and Xi, 2012), scale (Bronstein and Kokkinos, 2010) or rotation and translation (Gao et al., 2016a), which is useful in human body shape analysis. The use of shape descriptors has been shown to be particularly effective in deep learning shape models (Tan et al., 2018).

Deep generative techniques have gained prominence in the non-linear shape modelling field (Masci et al., 2015; Wu et al., 2016b; Tran and Liu, 2018; Tan et al., 2018). Deep learning methods can effectively scale to high dimensional data, which is more likely to have non-linearities, and compress the data more effectively than comparable linear methods (Nash and Williams, 2017). Other than PGA, autoencoders were used as non-linear analogs of PCA. Preliminary works used multilayer perceptrons and singular value decomposition (SVD) to achieve dimensionality reduction and compression (Bourlard and Kamp, 1988). Sanger, 1989 used a 3 layer neural network architecture, which was, unknowingly, similar to an AE. The network included an internal "bottleneck" of smaller dimension than either input "encoder" or output "decoder" layers. The smaller dimensional bottleneck was used to compress the encoded data from the earlier layer and the output layer decodes the compressed information to reconstruct the input (Sanger, 1989). In quick succession Kramer, 1991 used auto-associative neural networks to uncover both linear and non-linear correlations, without restriction on the type of the non-linearities existing in the data. Using the bottleneck idea, the author produced a comprehensive non-linear technique useful for identifying and removing correlations among variables, dimensionality reduction, visualization and exploratory data analysis. Moreover, the feature space resembled the actual distribution of the underlying system parameters (Kramer, 1991).

Sozou et al., 1997 was among the first to apply a multilayer perceptron to carry out non-linear PCA for shape analysis. The model used non-linear activations on the mapping layers and the bottleneck nodes were analogous to the shape parameters seen in linear models. They found that the technique was able to model highly non-linear shape spaces, but it was computationally expensive in training time and the quantity of data required (Sozou et al., 1997). This method provided a synonymous comparison with linear models as mathematically the optimal solution for an AE, with linear activations and a single hidden layer, results in PCA (Plaut, 2018). The flexible AE architecture is able to capture linearity, non-linearity, local and global deformations. However, this is dependent on hyperparameters such as: activation functions, number of nodes and error criterion for training (Blanz and Vetter, 1999; Bengio et al., 2009; Bagautdinov et al., 2018).

Another important generative model is the deep Boltzmann machine. Huang, Kalogerakis, and Marlin, 2015 used multi-layer Boltzmann machines to capture local and global shape variation. This part-based model proved to be effective at generating plausible shapes, shape segmentation and fine grained classification tasks. Boltzmann machines, due to their flexible nature, have also been used for speech data (Hinton et al., 2012), images and shape (Le Roux et al., 2011; Eslami et al., 2014). The shortfalls of these models, similar to other deep network techniques, is that they have proven to be difficult and time consuming to train. Moreover, they have shown to be more complicated to sample from than AEs (Nash and Williams, 2017).

Convolutional neural networks (CNN) are a popular choice for image synthesis. Similarly, using CNN for shape analysis is not uncommon. Wu et al., 2015 created a deep representation for 3D shapes using CNNs over volumetric representations instead of meshes. Their model, ShapeNets, learns the distribution of complex 3D shapes across different object categories and arbitrary poses. It performed well at both 3D shape completion and object recognition (Wu et al., 2015). ShapeNets has been used to analyse non-Euclidean manifolds and it has achieved a high performance in tasks such as shape description, retrieval, and correspondence (Masci et al., 2015). A limitation of CNN is that they are primarily applied to voxels, Nash and Williams, 2017

explain that volumetric methods are limited in terms of resolution due to the nature of the voxel grid. These grids require modelling of many redundant dimensions, such as empty space inside or outside the object itself. Tan et al., 2018 argues that the use of voxels is inefficient in shape analysis. The alternative representation of 3D meshes is proposed, as the shape structure is preserved without distortions from parameterisation. However, CNNs current application to 3D meshes is limited and non-trivial (Tan et al., 2018).

A powerful generative model used for both images and shape is the generative adversarial network (GAN). This architecture uses a generator network which maps low-dimensional latent samples to the data space. A discriminator network is then trained to distinguish between real and fake samples (Goodfellow et al., 2014). Wu et al., 2016b implemented a 3D GAN for object recognition which was effective at generating high-quality 3D samples. However, this method uses voxels and has no distributional assumptions, which makes evaluations and inference difficult. Although GANs ability to learn complex high-dimensional distributions is impressive, they are difficult to train as they suffer from model collapse, which causes them not to converge (Goodfellow et al., 2014; Dumoulin et al., 2016; Donahue, Krähenbühl, and Darrell, 2016). Recent studies in 3D shape and image analysis attempt to address the pitfalls of GANs by combining them with AEs (Larsen et al., 2015; Li et al., 2017).

Kingma and Welling, 2013 used the AE architecture to create an effective probabilistic directed model where the bottleneck layer inherits distributional properties. This alteration allows AEs to learn the parameters of a latent distribution from the input instead of an arbitrary function. Autoencoders reformulated in this way allowed the use of variational Bayesian inference to efficiently approximate intractable posterior distributions. Hence, giving them their name, Variational Autoencoders (VAEs) (Kingma and Welling, 2013). A VAE allows for inference and probabilistic sampling due to the smooth space learned by enforcing a prior on the latent distribution. They combine the strengths of local and global non-linear models whilst maintaining an efficient data-driven approach (Bagautdinov et al., 2018).

Variational autoencoders have made noteworthy contributions to shape analysis in recent literature (Nash and Williams, 2017; Bagautdinov et al., 2018; Tan et al., 2018). Bagautdinov et al., 2018 advocates VAEs effectiveness in 2D and 3D fitting tasks by demonstrating how they generalise for facial meshes with only 16 different examples. Their ability to perform well on few examples is important as high quality 3D mesh creation is expensive, time consuming and unrealistic in medical applications. Nash and Williams, 2017 use a VAE to synthesize 3D synthetic objects segmented into parts. They concluded that their shape VAE was effective at scaling to high dimensional data with non-linearities and could be trained quickly in comparison to other deep learning methods. Tan et al., 2018 used a mesh VAE to model anatomical non-segmented human bodies in different positions. The human body deformations were used as they have large scale non-linear point variations. Tan et al., 2018, following the work of Gao et al., 2016a, by used a rotation and translation invariant shape descriptor which improved the performance of their VAE substantially. Tan et al., 2018 claim that their model was easy to train with as few as 150 meshes and it could perform generation and interpolation tasks effectively. Both Tan et al., 2018 and Nash and Williams, 2017 use the Adam optimiser (Kingma and Ba, 2014) for model fitting, but it is not uncommon to use stochastic gradient descent to fit these models. This study shall only consider the use of Gaussian priors in the latent space

for simplicity although more sophisticated latent priors have been explored (Rezende and Mohamed, 2015; Kingma et al., 2016; Tomczak and Welling, 2017).

This research focuses on 3D generative techniques to capture non-linearities in shape data. Linear models are justified by: assuming the linear projection sufficiently approximates the non-linear shape space, their simplicity and computational efficiency. The literature suggests that a single layer AE with linear activations has an optimal solution equivalent to PCA. Moreover, the use of non-linear activations enables an AE to model both linear and non-linear deformations, such as rotation or bends in shape. Therefore, an AE is the ideal architecture for identifying non-linearity within 3D mesh data. Gaussian process morphable models are sophisticated, generative, linear shape models and argued to be user friendly, flexible and use minimal data (Lüthi et al., 2018). Within the non-linear generative modelling paradigm, VAEs are known to be computationally efficient, require minimal data and have impressive generative abilities on high dimensional, non-linear mesh data (Nash and Williams, 2017; Tan et al., 2018). A direct performance comparison between these two models will help bridge the gap between linear and non-linear 3D shape models. The next section will outline performance evaluation methods used in shape modelling.

1.4 Shape model evaluation

Bowden, 2000 elegantly summarises the non-linear problem as trying to model non-linearity accurately with the simplicity and speed of a linear model. Evaluation is challenging in shape modelling as true correspondences of biological shapes are often unknown. In consequence, ground truth correspondence measures are often used, which require manual setting of landmarks. A caveat of this approach is that it is laborious and time consuming for 3D data (Heimann and Meinzer, 2009). Nevertheless, assuming adequate correspondences, shape models are measured with respect to their generalisation to new objects in the shape family, generative sampling ability and variability explained by the parameters. The aforementioned concepts are defined, in shape modelling, by the following: generalization, specificity and compactness, respectively (Davies et al., 2003; Styner et al., 2003).

Generalisation describes the model's ability to accurately represent all valid instances in the shape family. It is a distance measure between new instances and the probability distribution estimated from the training shapes. To evaluate this we need to determine how well a model represents a target shape after being trained on typical shapes. The model is trained using a leave-one-out approach where all but one of the shape examples are used, with the last shape as a model projection test (Pizer and Marron, 2017; Lüthi et al., 2018). A common error metric for shape comparison is mean squared error (MSE), although an issue with this approach is that it may smooth out large errors. Therefore, in addition to MSE, the Hausdorff distance (HD) is considered in shape model evaluation. The HD measures the maximum error, or greatest distance, at a point (Cootes, Baldock, and Graham, 2000; Lüthi et al., 2018). The smaller the generalisation MSE and HD are, the better the model's ability to generate plausible shapes.

A model's ability to generate plausible instances of the object class is measured using specificity. This measures how well the probability distribution represents valid instances of the shape family. It is calculated by randomly sampling instances of the model then computing the average distance (Euclidean, geodesic or Hausdorff) between this sample and their nearest neighbours in the data (Brunton et al., 2014;

Pizer and Marron, 2017). There tends to be a trade-off in a model's ability to generalise well while still having a low specificity. This concept is akin to the bias-variance trade-off. As a form of regularisation, the variability explained by the model parameters is often considered.

Compactness measures the tightness of a probability distribution. The accumulated variance for a fixed number of parameters can be calculated as the determinant of the covariance matrix or a variability explained metric using eigenvalues (Styner et al., 2003; Pizer and Marron, 2017). This reaffirms the idea of parsimony. When comparing two models with identical generative ability and generalisation, we would prefer the model with less variance and fewer parameters (Sozou et al., 1997; Lüthi et al., 2018). Too many parameters in the model can cause it to overfit to the noise in the training data, but too simplistic models result in little shape detail being captured (Pizer and Marron, 2017).

Most modern techniques are criticised for being computationally inefficient during training and requiring large amounts of data to create reliable models. Proponents of GPMMs and VAEs mention that they have relatively efficient training and while requiring minimal data in their linear or non-linear paradigm, respectively (Lüthi et al., 2018; Tan et al., 2018). Following the ideas of Styner et al., 2003 and Davies et al., 2003 this research shall use generalisation and specificity to compare the accuracy of the model. The model complexity is not directly comparable as compactness requires eigenvalue components unique to the linear methods. In addition, the training data requirements and processing times shall be considered to compare computational efficiency.

2 | Aims and Objectives

When large variation is present between shapes a linear shape model's orthogonal projection, of the shape space, can be unreliable (Bowden, Mitchell, and Sarhadi, 2000; Klingenberg, 2020). This outlines the problem statement for this research. Linear shape models are assumed to adequately approximate non-linear data without comprehensive comparative analyses.

The literature revealed the following gaps: shape data may contains non-linearities that are not easily identifiable; linear shape models are assumed to adequately approximate non-linear shape data; there are no direct comparisons of linear and non-linear models for non-linear shape data.

This research has two aims. Firstly, provide a framework for identifying non-linearity in 3D shape data, using autoencoder (AE) models. This framework will be used to identify a linear and non-linear dataset for model comparison. Secondly, offer a comprehensive comparison study between two modern shape modelling techniques namely, Gaussian process morphable models (GPMMs) and variational autoencoders (VAEs). The models shall be evaluated according to their shape modelling accuracy and computational efficiency defined in the previous section. The objectives of this research are summarized below.

1) Develop a framework to identify non-linearity in 3D shape data through the use of autoencoder models.

2) Find the optimal GPMM (linear) and VAE (non-linear) models for the identified linear and non-linear datasets.

3) Compare the performance of the optimal shape modelling methods with regards to generalisation, specificity and training time.

This research aims to be relevant and reusable for many different applications. As a result, all statistical models and coding scripts are available on the following Github profile (`github.com/FJFehr`).

In the following chapter, a comprehensive theoretical background to the methods used in shape modelling is provided.

3 | Theoretical Background

The techniques described will use the following notation: matrices are denoted in upper case bold, vectors are denoted in lower case bold, and scalar elements are denoted in lower case italics (*e.g.*, $\mathbf{A}, \mathbf{a}, a$). The transpose and inverse operations shall be denoted by superscripts ^{T} and $^{-1}$ respectively. The identity matrix is a square matrix defined by $\mathbf{I}_n$.

The generic data matrix $\mathbf{X}_{n\times p}$ is comprised of n observations, p variables and has an arbitrary data point x_{ij} where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, p\}$.

3.1 Multivariate Gaussian Distribution

A univariate Gaussian distribution, often referred to as the normal distribution, has symmetric, bell-shape distribution. The mean μ and variance σ^2 parameters define the Gaussian distribution $N(\mu, \sigma^2)$. The univariate Gaussian density

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right),$$

describes a scalar random variable that is Gaussian distributed $X \sim N(\mu, \sigma^2)$ where $X \in \mathbb{R}$ (Johnson, Wichern, et al., 2002).

The multivariate Gaussian distribution is an extension of the univariate distribution, where a random matrix is Gaussian distributed $\mathbf{X}_{n\times p} \sim N_p(\boldsymbol{\mu}_{p\times 1}, \boldsymbol{\Sigma}_{p\times p})$. The p combined Gaussian variables will have density function

$$f(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{p}{2}}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

where $\boldsymbol{\mu}_{p\times 1}$ is a p dimensional vector of means and $\boldsymbol{\Sigma}_{p\times p}$ is a positive definite variance-covariance matrix (Johnson, Wichern, et al., 2002). Given a multivariate Gaussian distribution in $p = 2$ dimensions, the bivariate density function can be visualised by the Figure 3.1.

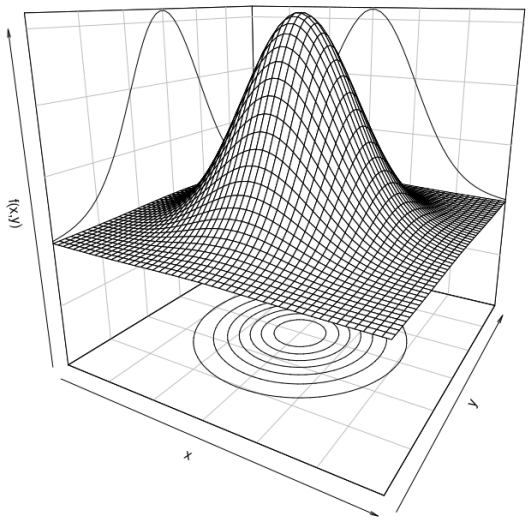

Figure 3.1: A standard normal, bivariate Gaussian distribution with mean vector $\boldsymbol{\mu} = [0, 0]^T$ and an identity covariance structure $\boldsymbol{\Sigma}_{2\times 2} = \boldsymbol{I}_2$.

Gaussian distributed random variables are uni-modal, symmetric and centered around the mean vector $\boldsymbol{\mu}$ (Johnson, Wichern, et al., 2002). Given two random vectors $\mathbf{x} = [x_1, \ldots, x_p]^T$ and $\mathbf{y} = [y_1, \ldots, y_p]^T$ with the following joint Gaussian distribution

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim N_{2p}\left(\begin{pmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{pmatrix} \right).$$

The marginal distribution on $\mathbf{x}$ is a Gaussian distribution of the form

$$\mathbf{x} \sim N_p(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx}).$$

The conditional distribution of $\mathbf{x}$, given that $\mathbf{y}$ is known, is a Gaussian distribution

$$\mathbf{x}|\mathbf{y} \sim N_p(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}),$$

where $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\Sigma}}$ are defined as

$$\begin{aligned} \bar{\boldsymbol{\mu}} &= \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_y) \\ \bar{\boldsymbol{\Sigma}} &= \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx} \end{aligned}$$

These convenient properties make the distribution useful for many applications (Johnson, Wichern, et al., 2002).

3.2 Principal Component Analysis

Principal Component Analysis (PCA) is a linear, multivariate statistical technique that reduces the dimensionality of data. The origin of PCA can be traced back over a century to Pearson (Pearson, 1901), but was formalised by Hotelling who coined the phrase *Principal Components* (Hotelling, 1933). These principal components are a new set of orthogonal variables which capture the variation in inter-correlated

dependent variables, shown in Figure 3.2. The goals of PCA are to: extract the most variation information from the data; and simplify and analyse the structure of the data (Abdi and Williams, 2010).

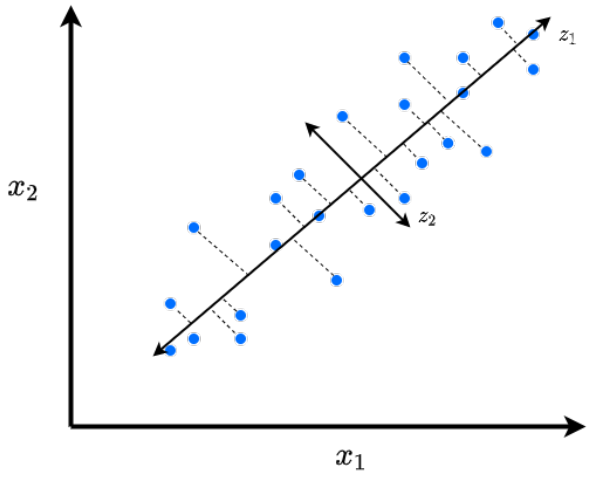

Figure 3.2: The data x_1 and x_2 are linearly projected onto a set of orthogonal variables z_1 and z_2, using PCA.

The data is preprocessed before analysis by centering each variable of $\mathbf{X}_{n\times p}$ such that the column mean is 0. If in addition each x_{ij} is scaled by $\sqrt{n}$, then this is known as *covariance PCA* since the matrix $\mathbf{X}^T\mathbf{X}$ is a covariance matrix. If the data is both centered and standardised to have a unit variance then $\mathbf{X}^T\mathbf{X}$ is a correlation matrix and the technique is known as *correlation PCA* (Abdi and Williams, 2010). For the purpose of this research we will only consider covariance PCA.

Assume the data matrix $\mathbf{X}_{n\times p}$ has mean vector $\boldsymbol{\mu}_{p\times 1}$ and covariance matrix $\boldsymbol{\Sigma}_{p\times p}$. Izenman, 2008 explains that PCA replaces the set of p unordered and correlated variables with k ordered and uncorrelated linear projections, $\boldsymbol{b}_1 \dots \boldsymbol{b}_k$, $(k \leq p)$ of the input variables

$$\begin{aligned} \boldsymbol{b}_1 &= \mathbf{z}_1^T\mathbf{X} = z_{11}\mathbf{x}_1 + \cdots + z_{1p}\mathbf{x}_p, \\ \boldsymbol{b}_2 &= \mathbf{z}_2^T\mathbf{X} = z_{21}\mathbf{x}_1 + \cdots + z_{2p}\mathbf{x}_p, \\ &\vdots \\ \boldsymbol{b}_k &= \mathbf{z}_k^T\mathbf{X} = z_{k1}\mathbf{x}_1 + \cdots + z_{kp}\mathbf{x}_p. \end{aligned} \tag{3.1}$$

The linear projections in Equation 3.1 are known as the *first k principal components* of $\mathbf{X}_{n\times p}$. Given high dimensional data, the input matrix $\mathbf{X}_{n\times p}$ has more features than observations such that $n << p$. The covariance matrix $\boldsymbol{\Sigma}_{p\times p}$ becomes large and inefficient to compute (Lata et al., 2009). The principle components can be derived using $\boldsymbol{\Sigma}_{n\times n} = \mathbf{X}\mathbf{X}^T$. We use this construct to derive the eigenvectors for $\boldsymbol{\Sigma}_{p\times p}$ as

$$\begin{aligned} (\mathbf{X}\mathbf{X}^T)\boldsymbol{\psi} &= \lambda\boldsymbol{\psi} \\ \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)\boldsymbol{\psi} &= \lambda(\mathbf{X}^T\boldsymbol{\psi}) \\ (\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\boldsymbol{\psi}) &= \lambda(\mathbf{X}^T\boldsymbol{\psi}) \end{aligned}$$

where $\boldsymbol{\psi}$ is the eigenvectors of $\mathbf{X}\mathbf{X}^T$. This shows that the eigenvectors for $\boldsymbol{\Sigma}_{p\times p}$ are given by $\mathbf{V} = (\mathbf{X}^T\boldsymbol{\psi})$. Principal component analysis uses the eigen decomposition to rewrite the positive definite covariance matrix as

$$\boldsymbol{\Sigma}_{p\times p} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T,$$

where the columns of $\boldsymbol{V}_{p\times m}$ are eigenvectors of $\boldsymbol{\Sigma}_{p\times p}$ and $\boldsymbol{\Lambda}_{m\times m}$ is a diagonal matrix of eigenvalues $\{\lambda_j\}$ of $\boldsymbol{\Sigma}_{p\times p}$ and $m \leq p$. Since the eigenvectors are orthogonal, $tr(\boldsymbol{\Sigma}) = tr(\Lambda) = \sum_{j=i}^{p} \lambda_j$. Izenman, 2008 further explains that the coefficient vectors $\mathbf{z}_j = [z_{1j}, \ldots, z_{pj}]^T$ are chosen such that the first k linear projections $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ of $\mathbf{X}$ are ranked in importance through their variances and listed in decreasing order of magnitude $var\{\boldsymbol{b}_1\} \geq \cdots \geq var\{\boldsymbol{b}_k\}$. As a result $\boldsymbol{b}_i$ is uncorrelated with $\boldsymbol{b}_j \; \forall \; i$ and j.

The principal components can be derived using a variance-maximisation technique. The goal is to find the k coefficients of $\mathbf{z}_j$ that maximise the variation of the linear projection $\boldsymbol{b}_j$. The variance of the components $var\{\boldsymbol{b}_j\} = var\{\mathbf{z}_j^T\mathbf{X}\} = \mathbf{z}_j^T\boldsymbol{\Sigma}\mathbf{z}_j$ are subject to the normalizations $\mathbf{z}_j^T\mathbf{z}_j = 1$, arranged in descending order and uncorrelated such that $cov(\boldsymbol{b}_i, \boldsymbol{b}_j) = \mathbf{z}_i^T\boldsymbol{\Sigma}\mathbf{z}_j = 0$. The first principal component is found by forming the Lagrangian equation

$$g(\mathbf{z}_1) = \mathbf{z}_1^T\boldsymbol{\Sigma}\mathbf{z}_1 - \lambda_1(1 - \mathbf{z}_1^T\mathbf{z}_1),$$

where λ_1 is a Lagrangian multiplier. Differentiating $g(\mathbf{z}_1)$ with respect to $\mathbf{z}_1$ and setting it to zero results in the equation

$$\frac{\delta g(\mathbf{z}_1)}{\delta \mathbf{z}_1} = 2(\boldsymbol{\Sigma} - \lambda_1\mathbf{I}_p)\mathbf{z}_1 = \mathbf{0}.$$

If $\mathbf{z}_1 \neq \mathbf{0}$ then λ_1 has to be the largest eigenvalue of $\boldsymbol{\Sigma}$ and $\mathbf{z}_1$ is the corresponding eigenvector. This process is repeated sequentially for all p variables with an added orthogonality constraint (Izenman, 2008).

Standard PCA assumes the data to be continuous, independent and that the relationship between variables can be expressed by linear combinations. If there exists second order dependencies or non-linearities between variables it can cause the linear technique to become unreliable at capturing the structure in the data. If the dataset is Gaussian distributed, then the linear combinations of the principal components will also be Gaussian distributed, therefore guaranteeing the principal components to be independent (Shlens, 2014).

3.3 Singular Value Decomposition

Wall, Rechtsteiner, and Rocha, 2003 defines the singular value decomposition (SVD) to be a method that allows any matrix $\mathbf{X}_{n\times p}$ to be factorised into three matrices

$$\mathbf{X}_{n\times p} = \mathbf{U}\mathbf{D}\mathbf{V}^T, \tag{3.2}$$

where $\mathbf{U}_{n\times p}$ and $\mathbf{V}_{p\times p}$ are orthogonal matrices containing the right and left singular vectors of $\mathbf{X}_{n\times p}$, respectively. The matrix $\boldsymbol{D}_{p\times p}$ is a diagonal matrix, where the singular values $[d_j]$, decrease sequentially (Venables and Ripley, 2013). The SVD

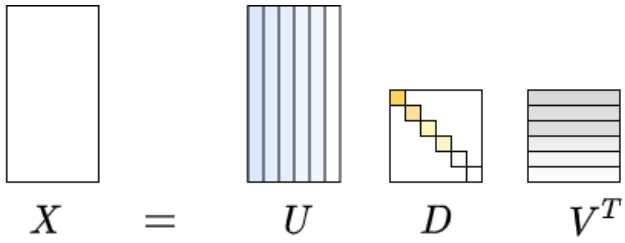

Figure 3.3: The SVD factorises the matrix $\mathbf{X}$ into 3 matrices. The orthogonal column vectors and diagonal values are coloured in descending order of their variation captured.

of a centered matrix $\mathbf{X}_{n\times p}$ can be used to express the *principal components* using Equation 3.2 and form

$$\begin{aligned}\frac{1}{n}\mathbf{X}^T\mathbf{X} &= \frac{1}{n}\mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{U}\mathbf{D}\mathbf{V}^T \\ &= \mathbf{V}\frac{\mathbf{D}^2}{n}\mathbf{V}^T\end{aligned}$$

where the format is similar to the eigen decomposition of the covariance matrix in Equation 3.2 (Hastie, Tibshirani, and Friedman, 2009). An equivalence can be drawn between these two techniques

$$\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \implies \mathbf{V}\frac{\mathbf{D}^2}{n}\mathbf{V}^T \; ; \quad \lambda_j = \frac{d_j^2}{n}$$

where the singular values relate to the eigenvalues (Venables and Ripley, 2013).

3.4 Point Distribution Models

Point distribution models (PDMs) capture the variation of corresponding points between shapes in d-dimensional space. Statistical analysis for biological shapes formed a preliminary foundation for Cootes et al., 1992 who later coined the term PDM (Kendall, 1989; Bookstein, 1997).

The construction of a PDM requires finding a mean shape and the modes of variation from a collection of training shapes. Lüthi et al., 2018 defines a single shape Γ_i, in a collection of training shapes $\Gamma_1, \ldots, \Gamma_n$, to be discrete set of points on the surface or boundary of the shape. A crucial assumption is that all Γ_i are in correspondence, meaning that x_k^i and x_k^j represent the same anatomical point on shape Γ_i and Γ_j, respectively (Lüthi et al., 2018). Given a shape in 2D, Γ_i is represented as a row vector

$$\mathbf{s}_i = [x_{1x}^i, x_{1y}^i \ldots x_{Nx}^i, x_{Ny}^i],$$

where $\mathbf{s}_i \in \mathbb{R}^{2N}$ and the x, y coordinates are row stacked. The matrix representation for all training shapes $\mathbf{S}_{n\times 2N}$ allows for variation between corresponding points to be modelled with a multivariate Gaussian distribution

$$\mathbf{S} \sim N_{2N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where the mean shape and covariance structure in 2D can be described by

$$\boldsymbol{\mu}_{1\times 2N} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{s}_i,$$

$$\boldsymbol{\Sigma}_{2N\times 2N} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{s}_i - \boldsymbol{\mu})(\mathbf{s}_i - \boldsymbol{\mu})^T.$$

Figure 3.4 describes a 2D hand shape where the variation of each coordinate is assumed to follow a Gaussian distribution.

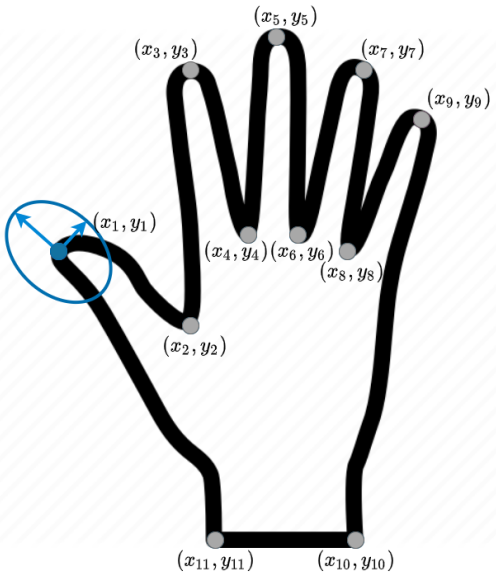

Figure 3.4: A 2D shape in with coordinate pairs describes the shape of a hand. The thumb tip coordinate displays the Gaussian distribution boundary of variation between other hand shapes. The boundary suggests that if all thumb tip coordinates were overlayed, we would see a bivariate Gaussian distribution of points within the boundary.

To reduce dimensionality, Cootes, Baldock, and Graham, 2000 uses PCA on the covariance matrix with model form

$$\mathbf{S}_{n\times 2N} = \mathbf{M} + \mathbf{VB}, \tag{3.3}$$

where $\mathbf{M}$ is a matrix with the mean $\boldsymbol{\mu}$ repeated on the columns, the eigenvectors $\mathbf{V}$ define a rotated coordinate frame and $\mathbf{B} = \mathbf{V}^T(\mathbf{S} - \boldsymbol{M})$ are the linear projections onto the rotated coordinate frame known as *shape parameters*. This is a matrix of $\mathbf{b}_j$ vectors equivalent to Equation 3.1. PCA assumes that the training set is Multivariate Gaussian distributed and thus the variance of $\mathbf{b}_j$ is given by $\sqrt{\lambda_j} = \sigma_j$ (Cootes, Baldock, and Graham, 2000). This suggests that if we set $\mathbf{b}_j$ between the bounds $\{-3\sqrt{\lambda_j}, 3\sqrt{\lambda_j}\}$ shapes similar to the training set can be generated.

3.5 Autoencoders

Autoencoders (AEs) are connected neural network architectures used to perform an identity mapping between input and output (Hassoun and Sudjianto, 1997). They were first introduced in the 1980's and used to compress data by forcing it through a 'bottleneck' network layer (Rumelhart, Hinton, and Williams, 1986; Sanger, 1989). The key advantage of using the neural network framework for capturing variation in lower dimensions is the theorem of universal approximation (Goodfellow, Bengio, and Courville, 2016). Hornik, 1991 states that a fully connected, feedforward network with a linear output layer, at least one hidden layer and with any non-linear activation function, can approximately map any finite-dimensional space to another. The theorem states that this can be done with zero error, given enough hidden units.

3.5.1 Definition

The network is split into two functions namely; encoder ϕ and decoder ψ as follows

$$\begin{aligned} &\phi : \mathbf{x} \rightarrow \mathbf{f} \\ &\psi : \mathbf{f} \rightarrow \tilde{\mathbf{x}} \\ &\phi^*, \psi^* = \underset{\phi,\psi}{\operatorname{argmin}} \; ||\mathbf{x} - (\psi \circ \phi)\mathbf{x}||, \end{aligned} \tag{3.4}$$

where ϕ maps the input $\mathbf{x}$ into a latent, lower dimensional space $\mathbf{f}$. This forms the compressive bottleneck known as the *code*. The decoder function ψ reconstructs the original input $\mathbf{x}$ from the latent space $\mathbf{f}$. The reconstruction $\tilde{\mathbf{x}}$ is found by parsing the data sequentially through the encoder ϕ and then the decoder ψ, also notated as $(\psi \circ \phi)\mathbf{x}$. The neural network equations

$$\begin{aligned} \phi : \mathbf{f} \; &= a_1(\mathbf{W}_1\mathbf{x}_i + \boldsymbol{\beta}_1) \\ \psi : \tilde{\mathbf{x}_i} &= a_2(\mathbf{W}_2\mathbf{f} \; + \boldsymbol{\beta}_2), \end{aligned}$$

denote how the data is passed through the network connections wrapped by the activation functions a_1, a_2. The weights $\mathbf{W}_1$, $\mathbf{W}_2$ and biases $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2$ in the encoder and decoder are optimised by minimising the reconstruction error. The error is calculated as the squared difference between the input and its reconstruction. Thus, AE's are able to recreate the data after a generalised linear or non-linear compression (Goodfellow, Bengio, and Courville, 2016).

Given a hand shape in 2D, Figure 3.5 describes how a single hidden layer (or more) AE model could be used to learn the variation of points and reduce the dimensionality of this dataset. The model would be able to replicate a PDM using a neural network structure, where the hidden bottleneck layer represents the shape parameters or the lower dimensional projection (Plaut, 2018)

3.5.2 Relationship to PCA

Plaut, 2018 shows that the mathematically optimal solution for an AE, with linear activations and a single hidden layer, results in PCA. The hidden layer spans the same vector subspace as the first principal components in PCA, without orthogonality constraints or ordered in descending order of variance. The weights are not equal to

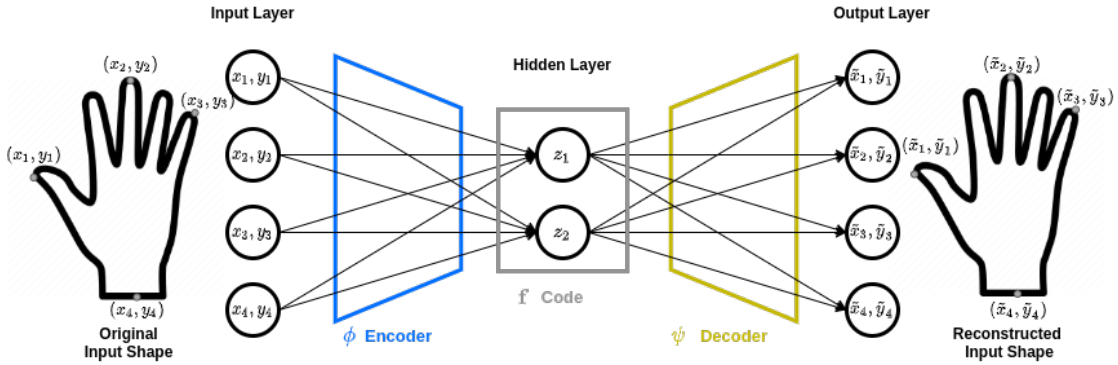

Figure 3.5: An AE model is used to capture the variation in a 2D hand shape. The model uses (x, y) coordinates as inputs to the encoder and trains the decoder to efficiently reconstruct the shape.

the principal components and not orthogonal, but can be recovered by applying SVD to the weight matrix of one of the two layers (Plaut, 2018).

Plaut, 2018 advocates the use of AE over PCA as it is able to process high-dimensional data with many observations. Implementation is easy and new data is easily added to the model (Plaut, 2018). However, AE's are computationally expensive with regards to training time and the amount of data required to train them in comparison to PCA (Sozou et al., 1997).

3.5.3 Parameters and hyperparameters

The parameters in a neural network architecture are an internal configuration of variables. These include: the weights and biases of each layer which are set through an initialisation and optimised by the data. The hyperparameters of a model are external variables and set manually through grid searches, cross-validation and prior knowledge.

Initialisation

Before an autoencoder can be trained, the weight parameters require initialisation. The weights are required to be asymmetrical - sufficiently different across layers - to allow for the parameters to be updated during the learning process, backpropagation. Géron, 2019 explains that when the weights are small or identical the network will be unable to update due to the gradient vanishing between layers. To provide asymmetry, the weights are commonly initialised using variations of Gaussian or uniform distributions. Notable variations, which provide training efficiencies, are proposed by Glorot (Glorot and Bengio, 2010), He (He et al., 2015) and LeCun (LeCun, Bengio, and Hinton, 2015).

Epochs

Gulli and Pal, 2017 defines a single epoch to be a full parse of the training data through the model. The number of epochs required is subjective to the model and data; however, it is usually set to sufficiently stabilise the validation accuracy (Gulli and Pal, 2017).

Batch size

The batch size hyperparameter defines the number of training instances observed before the model weights are updated (Gulli and Pal, 2017). This allows for multiple batches, also known as steps or iterations, within a single epoch which is faster and more memory efficient (Gulli and Pal, 2017). Géron, 2019 points out that large batch sizes can be computationally challenging to train and generalise poorly. It is suggested to use batch sizes between 2-32 as they tend to result in better models in less training time (Géron, 2019).

Layers

The number of layers and nodes in each layer are an important hyperparameter to consider in neural network architectures. In AEs, the dimension of the input and output layers are identical; however, the number of intermediate hidden layers and their node dimension, should be considered. Géron, 2019 explains that due to the universal approximation theorem a single hidden layer network can approximate even the most complex functions, provided it has enough nodes. When considering complex problems, deep multilayered networks can model these functions with fewer nodes than shallower networks (Géron, 2019). The deeper the network the more complexity can be captured. A caveat of this is that the deep network requires more data and time in training (Géron, 2019). In an AE network, the code layer of the network controls the complexity of the model. Too few nodes in the code layer will result in less variation captured by the model. As an effect, the model will have a high bias. Too many nodes in the code layer and the model will overfit the data, leading to no compression (Goodfellow, Bengio, and Courville, 2016).

Loss function

In a supervised context, the loss function evaluates the prediction result against the true value or label. This allows the model to assess and optimise its performance (Gulli and Pal, 2017). In regression examples the mean squared error (MSE) loss function is often used

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2,$$

where the difference between all predictions $\tilde{x}_i$ and true labels x_i are squared and aggregated. The MSE loss function is intuitive and penalises incorrect predictions heavily due to the squaring operation. Other loss functions could be considered such as: mean absolute error, mean percentage error and mean squared logistic error (Gulli and Pal, 2017). Classification examples require different loss functions that factor in the categorical data such as binary or categorical cross-entropy and hinge loss (Gulli and Pal, 2017).

Regularisation

Models are optimised to achieve the lowest loss on the training data; however, this can lead to over complex, inefficient models (Gulli and Pal, 2017). Gulli and Pal, 2017 presents two negative consequences of increasing the complexity; firstly the increase in running time and secondly the model will memorise the patterns in the training data leading to overfitting. This will allow the model to achieve perfect accuracy

on the training data but generalise poorly on unseen data. Regularisation restrict the models complexity to prevent the model from overfitting. In a neural network framework the loss function is penalised by constraining the weights using a ridge (L2) or lasso (L1) regularisation (Abu-Mostafa, Magdon-Ismail, and Lin, 2012).

$$L_1 : \mathcal{L}_{MSE} + \lambda \sum_{i=1}^{n} |w_i|$$
$$L_2 : \mathcal{L}_{MSE} + \lambda \sum_{i=1}^{n} |w_i|^2$$

The hyper parameter λ determines the magnitude of the regularisation. The L_2 regularisation preserves the weights by uniformly shrinking them to non-zero values. The L_1 regularisation tends to shrink the weights to zero allowing for a sparse representation (Gulli and Pal, 2017). Srivastava et al., 2014 presents another effective induced regularisation technique called dropout. The concept is to reduce the complexity of a fully connected network by dropping nodes and edges which results in a sparse representation.

Optimisers

The optimal weights for a neural network cannot be found analytically and thus, an empirical optimisation method is needed (Géron, 2019). The objective of optimisation is to minimise the loss function by adjusting the weights of the network. Figure 3.6 visually represents the optimisation process. The method iteratively adjusts the weights $\mathbf{w}$ to find the global minimum of the loss function $l(\mathbf{w}^*)$. The gradient $\frac{\delta l}{\delta \mathbf{w}}$ of the weights $\mathbf{w}$ is calculated at each iteration to direct the adjustment of the weights.

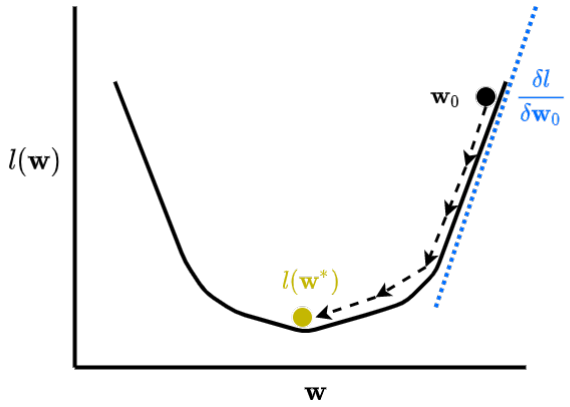

Figure 3.6: The loss function $l(\mathbf{w})$ is represented as a function of the weights $\mathbf{w}$ in a neural network architecture. The random initialisations of the weights $\mathbf{w}_0$ are optimised using the gradient to find a global minimum loss $l(\mathbf{w}^*)$.

Gulli and Pal, 2017 describes the optimisation process in two parts. First the data is passed through the model by a forward propagation where the prediction is evaluated by the loss function. Secondly, the negative gradient of the loss function is propagated back through the layers and used to update the weights. This process defines the learning process of the neural network known as backpropagation. Optimisers such as Stochastic gradient descent (Robbins and Monro, 1951) and Adam (Kingma and Ba, 2014) allow for efficient ways to find the optimum set of weights. The Adam

optimiser is a popular choice as it keeps track of prior gradients and has an adaptive learning rate (Géron, 2019)

Learning rate

Figure 3.6 demonstrates how the neural network learns from data using backpropagation. Each step, determined by the batch size, updates the weights. The magnitude of the update is determined by the gradient of the weight $\frac{\delta l}{\delta \mathbf{w}}$, a hyperparameter known as the learning rate α. The updated weights $\tilde{\mathbf{w}}$ are adjusted by

$$\tilde{\mathbf{w}} = \mathbf{w} - \boldsymbol{\alpha}\frac{\delta l}{\delta \mathbf{w}}$$

where $\boldsymbol{\alpha}$ is the appropriate dimension vector of repeated α values. Gulli and Pal, 2017 notes that when the learning rate is small the model will converge, albeit slowly, as the steps will be close to one another. However, if the learning rate is large, the model will descend to the optimum faster, but at the risk of missing the global minimum or non-convergence. Géron, 2019 discusses the advantages of a learning rate schedule which can reach an optimal solution more efficiently by starting the learning rate high and decreasing it through the training process.

Batch Normalisation

Training neural network architectures with multiple layers is challenging due to the number of parameters and hyperparameters. Ioffe and Szegedy, 2015 defines internal covariate shift to be the change of distribution of the inputs to layers in the network after each weight update. The parameters are optimised during backpropagation but weights are updated towards a changing target. Ioffe and Szegedy, 2015 proposes a normalisation for each training step which stabilises the learning process and reduces the number of required training epochs. Using batch normalisation allows for more efficient training by eliminating the need for small learning rates, careful initialisations or sophisticated regularisation techniques (Ioffe and Szegedy, 2015).

Activation functions

The activations in the nodes control how the data is passed through each layer. Figure 3.7 describes six common linear (blue) and non-linear (gold) activation functions. The nodes show a diagram of how the data is transformed from input by the activation, to the output. The linear, or identity, activation is commonly used as it is simple to optimise for gradient based algorithms (Goodfellow, Bengio, and Courville, 2016).

Early non-linear AE's used sigmoid logistic (σ) activations or hyperbolic tangent (tanh) activations. Sigmoidal functions are often used on the output layer to predict the probability of a binary outcome. By design, this activation tends to saturate the positive output to be 1 and negative output to be 0. As a result, they are known to be difficult to optimise using gradient based optimisers and therefore discouraged for hidden layers. The tanh activation typically performs better than a sigmoid activation as the behaviour about zero $tanh(0) = 0$ and $\sigma(0) = 1/2$ makes it easier to optimise (Goodfellow, Bengio, and Courville, 2016).

Géron, 2019 presents an issue with the aforementioned activations due to their restricted bounds. During optimisations such as gradient descent the gradient becomes smaller as it is processed through the layers until the weights become unchanged. As

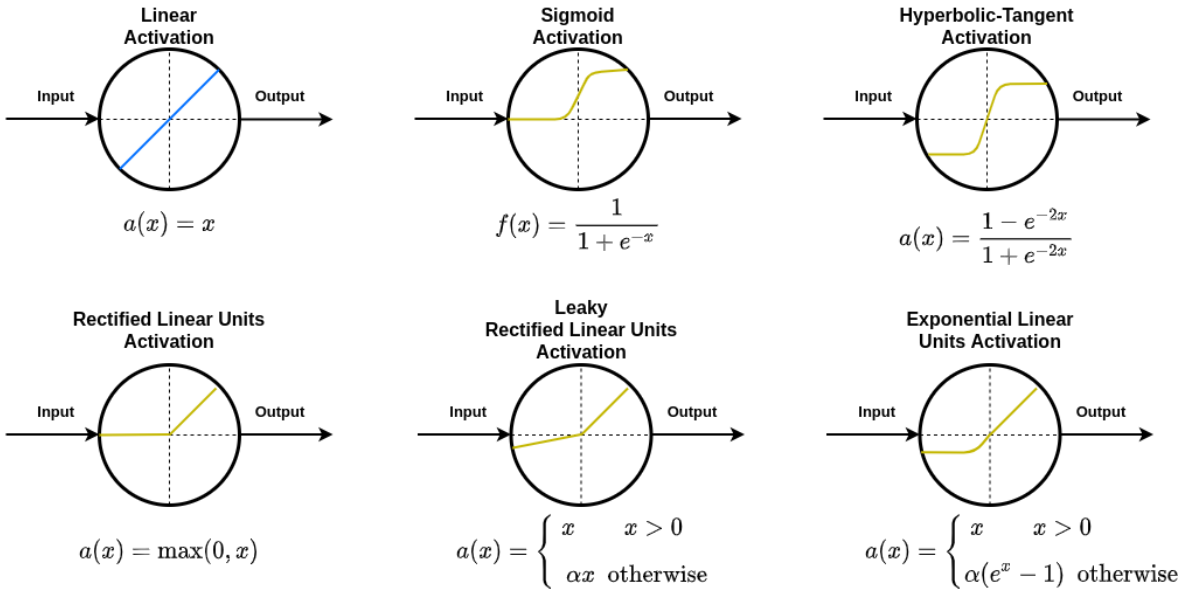

Figure 3.7: Each of the six nodes above show linear (blue) and non-linear (gold) activation functions.

a result, the training never converges to an adequate solution because of the gradient vanishing problem.

The Rectified Linear Units (ReLU) activation alleviates the gradient vanishing problem as it has an unbounded upper bound. Goodfellow, Bengio, and Courville, 2016 explains that the only difference between a linear activation and a ReLU activation is that ReLU outputs zero across half its domain. This changes the derivatives of the activation which allows for faster optimisation in the presence of second order effects which cause non-linearity (Goodfellow, Bengio, and Courville, 2016). The ReLU activation, although computationally appealing, suffers from the dying ReLU problem (Géron, 2019). This is seen in training when many activations of the network output zeros. This is a result of the weighted sum of the inputs being negative, which in turn, sets the gradient to zero when the ReLU activation is negative (Géron, 2019).

Variations of the ReLU function are used to revive the nodes and prevent the vanishing gradient problem. The leaky ReLU, as seen in Figure 3.7, is unbounded in both positive and negative directions. The small slope in the negative domain ensures that the outputs do not go to zero (Xu et al., 2015). Clevert, Unterthiner, and Hochreiter, 2015 provided another alternative variation known as the Exponential Linear Unit (ELU). This activation alleviated the vanishing gradient and dying ReLU problems (Géron, 2019). The ELU activation leads to faster convergence than the other ReLU variants; however, the function is slower to compute than ReLU (Géron, 2019).

3.6 Variational Autoencoders

In Section 3.5 the AE model was introduced as a compressive neural network architecture. The encoded latent space $\mathbf{f}$ in Equation 3.4 forms a discontinuous and irregular function which is not conducive to generative models. Kingma and Welling, 2013 proposed the variational autoencoder (VAE) which inherits the encoder and decoder architecture; however, encodes the inputs as a distribution over the latent space

instead of arbitrary points (Kingma and Welling, 2013). The reconstruction loss function includes an additional regularisation term to ensure the generative properties of the latent space. As a result, the latent space is coerced to a smooth, continuous space by the loss function (Odaibo, 2019). The distributional assumption organises similar observations to lie in close proximity in the latent space allowing effective generative sampling. In training, the VAE architecture inherits all the parameters and hyperparameters of AEs seen in Section 3.5.3.

3.6.1 Definition

Variational autoencoders are defined using a probabilistic framework. In contrast to the deterministic framework of an AE, the input is encoded as a probability distribution over the latent space. Thereafter, the latent space distribution is sampled and decoded to retrieve the reconstruction. The probabilistic encoder is defined as $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, where the approximate posterior distribution of the latent variable $\mathbf{z}$ is conditional on the given input $\mathbf{x}$ and parameterised by the collective network weights $\boldsymbol{\theta}$. Similarly, the probabilistic decoder is defined as $p_{\boldsymbol{\omega}}(\mathbf{x}|\mathbf{z})$ and describes the likelihood distribution parameterised by the collective network weights $\boldsymbol{\omega}$. The encoder and decoder functions are defined as

$$\begin{aligned}
&\phi : q_{\boldsymbol{\theta}(\mathbf{z}|\mathbf{x})} \\
&\psi : p_{\boldsymbol{\omega}(\mathbf{x}|\mathbf{z})} \\
&\boldsymbol{\theta}^*, \boldsymbol{\omega}^* = \underset{\theta,\omega}{\operatorname{argmin}}\ \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\omega}),
\end{aligned}$$

where the optimal weight parameters $\boldsymbol{\theta}^*$ and $\boldsymbol{\omega}^*$ are found by minimising the loss function. The latent distribution is typically Gaussian for simplicity; however, alternative priors have been considered (Rezende and Mohamed, 2015; Kingma et al., 2016; Tomczak and Welling, 2017). Figure 3.8 visualises a simple VAE model for a hand shape in 2D. Using a latent dimension of two, the parameters of a bivariate Gaussian distribution $[\mu_1, \mu_2, \sigma_1, \sigma_2]$ are trained in the encoder. The distribution is sampled to obtain the latent values $[z_1, z_2]$ and decoded to return our reconstruction.

The loss function is optimised by maximising the evidence lower bound (ELBO) also known as the variational lower bound (Kingma and Welling, 2013). The VAE loss function

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\omega}) = -[\underbrace{\mathbb{E}_{q_{\boldsymbol{\theta}}}[\log(p_{\boldsymbol{\omega}}(\mathbf{x}|\mathbf{z})]}_{(1)} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{(2)}], \tag{3.5}$$

wraps the ELBO with a negative sign thus, maximising the variational lower bound is equivalent to minimising the loss function (Odaibo, 2019). The ELBO in Equation 3.5 is split into two terms. The first term (1) is a measure of the likelihood of the reconstruction of the data. This is equivalent to the reconstruction error used for AEs. The second term (2) in the ELBO is used to regularise the loss function by constraining the latent distribution. The Kullback-Leibler divergence guides the posterior distribution of the encoder to return a standard normal Gaussian distribution (Odaibo, 2019).

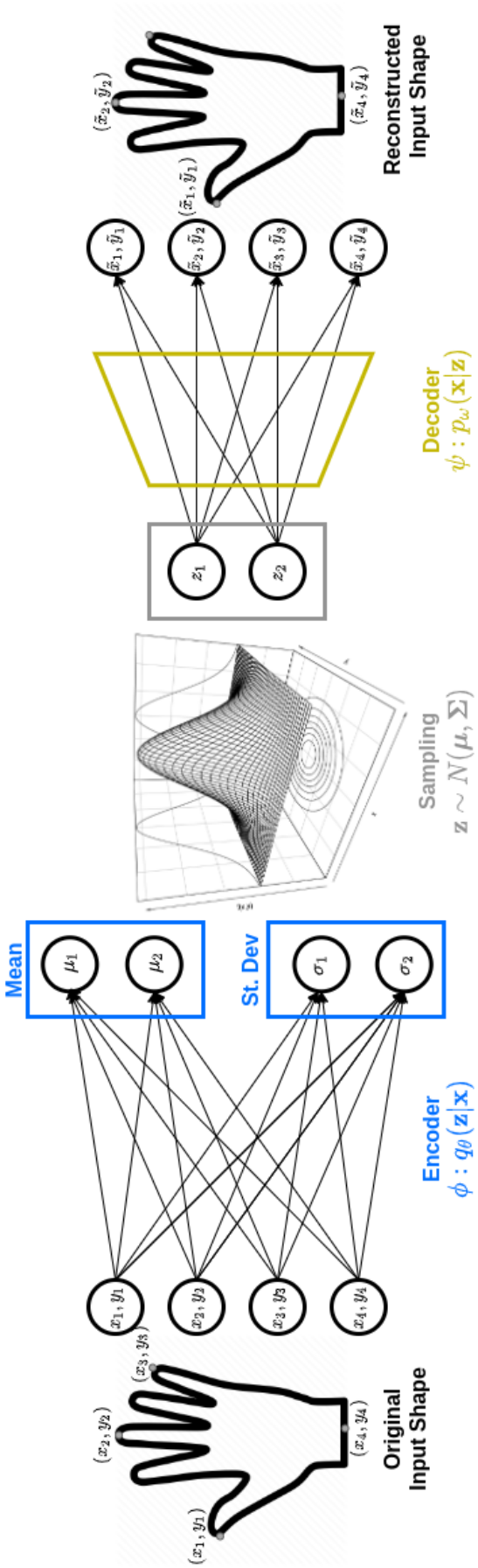

Figure 3.8: A VAE model is used to capture the variation in a 2D hand shape. The model uses coordinates as input to the encoder and a bivariate Gaussian distribution in the latent space. The distribution is sampled and decoded to reconstruct the shape.

3.6.2 Kullback-Liebler divergence

Odaibo, 2019 presents the Kullback-Liebler (KL) divergence as a similarity measure between distributions that uses the expected value of the difference of information (Odaibo, 2019). The information content of an event is defined to be inversely related to the probability of said event. Given two distributions $p(x)$ and $q(x)$, their information is defined by

$$\begin{aligned} I_p &= -\log(p(x)) \text{ and} \\ I_q &= -\log(q(x)), \end{aligned}$$

respectively. The difference in information of $p(x)$ and $q(x)$ is described by

$$\begin{aligned} \Delta I &= I_p - I_q \\ &= -\log p(x) + \log q(x) \\ &= \log\left(\frac{q(x)}{p(x)}\right). \end{aligned}$$

The KL takes the expectation of the difference in information

$$\begin{aligned} D_{\mathrm{KL}}(q(x)||p(x)) &:= \mathbb{E}_{q(x)}[\Delta I] \\ &= \int \Delta I q(x) dx \\ &= \int \log\left(\frac{q(x)}{p(x)}\right) q(x) dx, \end{aligned}$$

and similarly,

$$\begin{aligned} D_{\mathrm{KL}}(p(x)||q(x)) &:= \mathbb{E}_{p(x)}[\Delta I] \\ &= \int \Delta I p(x) dx \\ &= \int \log\left(\frac{p(x)}{q(x)}\right) p(x) dx. \end{aligned}$$

The equations are non-negative and non-symmetric. The non-symmetry implies the following relation $D_{\mathrm{KL}}(p(x)||q(x)) \neq D_{\mathrm{KL}}(q(x)||p(x))$. Hence, the KL is referred to as a divergence and not as a metric (Odaibo, 2019). The loss function of a VAE is derived from the KL divergence

$$D_{\mathrm{KL}}(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})),$$

where $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is the encoder approximate posterior distribution and $p(\mathbf{z}|\mathbf{x})$ is the real posterior distribution. Odaibo, 2019 manipulates this equation to derive

$$\log(p(\mathbf{x})) \geq \mathbb{E}_{q_{\boldsymbol{\theta}}}[\log(p_{\boldsymbol{\omega}}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})), \tag{3.6}$$

where the evidence lower bound, as seen before in Equation 3.5, is defined. Equation 3.6 shows that maximizing the ELBO maximizes the log probability of our data by

proxy and bounds the likelihood of the data (Odaibo, 2019). This is a fundamental concept of variational bayesian inference since maximization of the log probability directly is typically computationally intractable.

3.6.3 Variational inference

Statistical inference is the process of estimating properties, such as distributions, of a population based on statistics from a sample (Box and Tiao, 2011). Within the Bayesian paradigm, prior knowledge is described by a probability distribution and updated by new data. This inference is encapsulated by Bayes theorem

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)},$$

where the data x is generated from a probability distribution depending on the unknown parameter θ. Bayes theorem updates the posterior $p(\theta|x)$ using the prior distribution $p(\theta)$ of the unknown parameter and the likelihood from the data $p(x|\theta)$. The evidence $p(x)$ encapsulates the probability distribution of the data and is used as a normalisation factor. Unlike the previous terms, the evidence is often not explicitly known and requires calculation

$$p(x) = \int_{\theta} p(x|\theta)p(\theta)d\theta.$$

The calculation of the evidence can be intractable in higher dimensions and thus, numerical approximations are required (Fox and Roberts, 2012; Doersch, 2016). Two typical approaches for this numerical approximation are Markov chain Monte Carlo (MCMC) and variational inference (VI) methods. The VI method, although less accurate, is known to be computationally efficient and scalable to large datasets (Zhang et al., 2018).

Variational inference uses an optimisation process to find the best parameters of given distribution family. Zhang et al., 2018 explains that a family of distributions defines the search space and simplicity of the method. Once the family has been defined, the best parameters for the distribution are optimised by considering the KL divergence as seen in Section 3.6.2. The KL divergence is not sensitive to multiplicative coefficient and thus suitable to calculate the similarity between our approximate distribution and target distribution (Zhang et al., 2018).

Figure 3.9 outlines the VI process. The first part of the process requires a selection of the distribution family. Zhang et al., 2018 explains that the choice of distribution family determines the trade-off between being expressive enough to approximate the target distribution and simple enough to lead to a tractable approximation. If the family of distributions is not close to the target distribution the best approximation will return poor results (Zhang et al., 2018). The second stage of the process initialises and iteratively optimises the parameters according to the KL error. Finally the best parameters are selected which allows for the evidence to be estimated and used for inference.

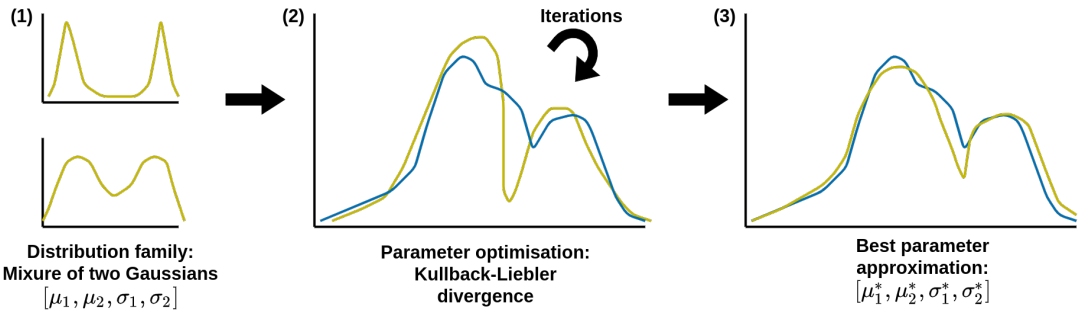

Figure 3.9: (1) The distribution family is selected as a mixture of two Gaussian distributions with initial parameters $[\mu_1, \mu_2, \sigma_1, \sigma_2]$. (2) The approximated distribution (gold) is fitted to the target distribution (blue). The parameters are optimised through multiple iterations of the optimiser using KL divergence. (3) The best parameters for the given family are selected $[\mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*]$

.

3.6.4 Reparameterisation trick

During a forward pass of the data through the VAE, the approximate latent distribution $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is sampled. Assuming a Gaussian distribution, the latent variable $\mathbf{z}$ is defined as

$$\mathbf{z} \sim q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = N(\boldsymbol{\mu}, \boldsymbol{\sigma}\mathbf{I}),$$

where the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ parameters are learned by the network. This process is stochastic in nature and does not allow for the backpropagation of errors through the network. Thus, the weight parameters of the VAE would be impossible to optimise. Kingma and Welling, 2013 provides a simple reparameterisation

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}$$
$$\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}),$$

which allows the random variable $\mathbf{z}$ to be deterministic. The element-wise product of the auxilary random variable $\boldsymbol{\epsilon}$ allows for a differentiable estimator (Kingma and Welling, 2013).

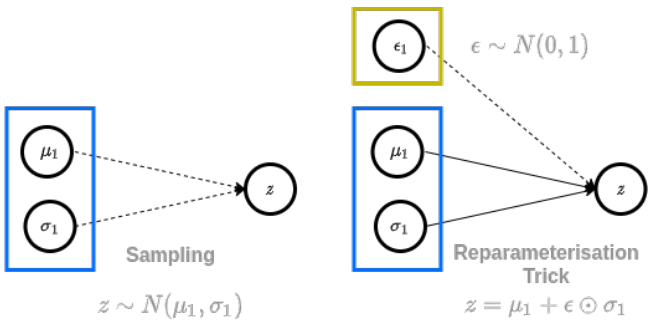

Figure 3.10: The latent variable $\mathbf{z}$ is sampled from a Gaussian distribution (dotted arrows). The reparameterisation trick provides a deterministic representation of the latent variable $\mathbf{z}$ (solid arrows). The auxiliary random variable $\boldsymbol{\epsilon}$ simulates the sampling process.

Figure 3.10 provides a visual representation of the reparameterisation trick. As a result, the reparameterisation allows the backpropagation of errors through the stochastic sampling step of VAEs allowing them to be optimised.

3.7 Gaussian Process Morphable Models

The interpretation of a Gaussian process morphable model (GPMM) can be seen as a continuous generalisation of a PDM (Lüthi et al., 2018). A reference shape defines the points domain and is selected arbitrarily from the dataset. The variation in the dataset is modelled by assuming the set of corresponding vectors, from the reference shape to all other shapes, are Gaussian distributed (Lüthi et al., 2018). Williams and Rasmussen, 2006 presents Gaussian processes as a generalisation of the multivariate Gaussian distribution to infinitely many vectors through defining a mean and covariance function. The covariance function, referred to as the kernel function, can be user specified to include expert knowledge not seen in the data (Williams and Rasmussen, 2006). The Karhunen-Loeve expansion allows for linear dimension reduction which is a continuous analog of PCA (Lüthi et al., 2018). As a result, a GPMM provides a continuous and flexible adaptation of linear point distribution models.

Gaussian processes shall be explained including their relationship to multivariate Gaussian distributions in Section 3.7.1. The application of Gaussian processes to shape modelling will be discussed in Section 3.7.2 and finally, the computational methods used to calculate the eigenfunctions and kernel functions will be mentioned.

3.7.1 Definition

To build some intuition around Gaussian processes consider $\mathbf{\Omega}$ to be a continuous domain where $\tilde{\mathbf{\Omega}} = [x_1, \ldots, x_n] \subset \mathbf{\Omega}$ is a discrete subset of that domain. Now consider the following discrete function

$$\tilde{h} : \tilde{\mathbf{\Omega}} \to \mathbb{R},$$

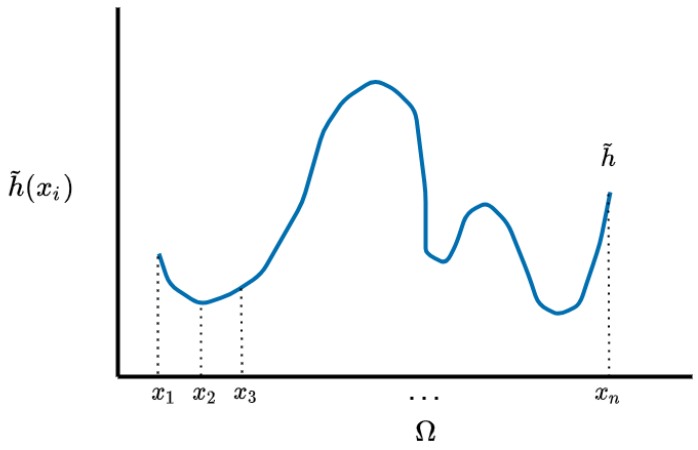

Figure 3.11: The discrete function $\tilde{h}$ is represented over the domain $\mathbf{\Omega}$.

which can be visualised by Figure 3.11. The function $\tilde{h}$ can be represented by a vector over the discrete domain $\tilde{\mathbf{\Omega}}$ with

$$\mathbf{h} = [\tilde{h}(x_1), \tilde{h}(x_2), \tilde{h}(x_3), \ldots, \tilde{h}(x_n)]^T \in \mathbb{R}^n.$$

The vector can be modelled as a multivariate Gaussian distribution, as seen in Section 3.1, over discrete functions

$$\mathbf{h} \sim N_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\mu}_{n\times 1}$ and $\boldsymbol{\Sigma}_{n\times n}$ are the mean vector and covariance matrix, respectively. Using this distribution a random function can be sampled and used to define a discrete function $\tilde{\mathbf{h}}$. A Gaussian process is defined over a continuous domain by defining a mean function $m : \boldsymbol{\Omega} \to \mathbb{R}$, and kernel function $k : \boldsymbol{\Omega} \times \boldsymbol{\Omega} \to \mathbb{R}^{1\times 1}$ such that

$$\mathbf{h} \sim GP(m, k).$$

This generalisation permits any discretisation of points, which infers infinitely many variables, making it a non-parametric model (Williams and Rasmussen, 2006). Once a discrete domain is selected, the Gaussian process returns a multivariate Gaussian distribution over the selected discrete domain (Lüthi and Bouabene, 2020). Thus, a Gaussian process is considered to be a continuous analog of a multivariate Gaussian distribution.

3.7.2 Shape modelling with Gaussian Processes

It has been shown that a Gaussian process is a continuous analog of a multivariate Gaussian distribution. Lüthi et al., 2018 uses Gaussian processes for shape modelling to model the deformations (landmark variation vector fields) from a reference shape (points domain). The shape contour in 2D, or surface in 3D, is a continuous domain which has been made discrete through a subset of corresponding landmark points. Given an example 2D hand shape dataset in correspondence, seen in Figure 3.12, with only two points we can select an arbitrary reference shape

$$\boldsymbol{\Gamma}_r = \{\mathbf{x} | \mathbf{x} \in \mathbb{R}^2\},$$

where the $\mathbf{x}$ is a vector of points in 2D. The deformation field u maps the reference shape $\boldsymbol{\Gamma}$ to all other shapes,

$$u : \boldsymbol{\Gamma} \to \mathbb{R}^2.$$

Each shape in the dataset has a set of corresponding vectors from the reference shape. These deformation vectors are assumed to follow a joint multivariate Gaussian distribution,

$$\begin{bmatrix} u(x_1) \\ u(y_1) \\ u(x_2) \\ u(y_2) \end{bmatrix} \sim N\left(\begin{pmatrix} \mu_1(x_1) \\ \mu_2(y_1) \\ \mu_1(x_2) \\ \mu_2(y_2) \end{pmatrix}, \begin{pmatrix} \sigma_{11}(x_1,x_1) & \sigma_{12}(x_1,y_1) & \sigma_{11}(x_1,x_2) & \sigma_{12}(x_1,y_2) \\ \sigma_{21}(y_1,x_1) & \sigma_{22}(y_1,y_1) & \sigma_{21}(y_1,x_2) & \sigma_{22}(y_1,y_2) \\ \sigma_{11}(x_2,x_1) & \sigma_{12}(x_2,y_1) & \sigma_{11}(x_2,x_2) & \sigma_{12}(x_2,y_2) \\ \sigma_{21}(y_2,x_1) & \sigma_{22}(y_2,y_1) & \sigma_{21}(y_2,x_2) & \sigma_{22}(y_2,y_2) \end{pmatrix} \right)$$

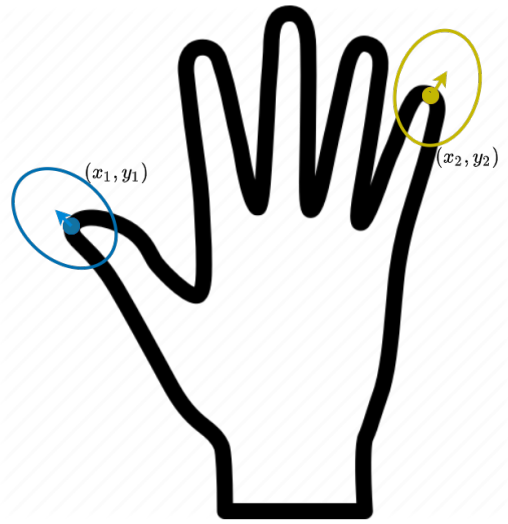

Figure 3.12: A 2D shape, with two coordinate pairs, describing the shape of a hand. The thumb and fifth digit tip coordinate displays the Gaussian distribution boundary of variation between other hand shapes. The boundary suggests that if all coordinates were overlayed, we would see a bivariate Gaussian distribution of vectors within the boundary.

where all pairwise deformation vectors are shown. This representation quickly becomes cumbersome for more than two points or more than two dimensions (Lüthi and Bouabene, 2020). Thus, a Gaussian process is defined over a continuous domain by defining a mean function $m : \mathbf{\Gamma}_r \to \mathbb{R}^2$, and kernel function $k : \mathbf{\Gamma}_r \times \mathbf{\Gamma}_r \to \mathbb{R}^{2\times 2}$ such that

$$u \sim GP(m, k).$$

A sampled shape $\mathbf{\Gamma}_i$ can be represented by sampling from the Gaussian process of deformations to obtain $\hat{u}$,

$$\mathbf{\Gamma}_i = \{x + \hat{u}(x) | x \in \mathbf{\Gamma}_r\},$$

where $\mathbf{\Gamma}_i$ is a deformation from the reference shape Γ_r. Gaussian processes can easily be scaled to 3D by defining the reference and deformation field in 3D. Point distribution models capture the variation between shapes using PCA (described in Section 3.2). Gaussian process morphable models use a continuous linear equivalent known as the Karhunen-Loeve expansion

$$u(x) \sim m(x) + \sum_{i=1}^{\infty} \alpha_i \sqrt{\lambda_i} \phi_i(x) \qquad \alpha \sim N(0, 1), \tag{3.7}$$

where λ_i and $\phi_i(x)$ are the corresponding eigenvalue and eigenfunctions, respectively. The parameter α_i are analogues to the shape parameters seen in Section 3.4 modelled by a standard Gaussian distribution and bounded to $\{-3, 3\}$. The Karhunen-Loeve expansion approximates the deformation function u through a linear projection using the mean function and an infinite number of orthogonal basis functions (Berlinet and

Thomas-Agnan, 2011). The orthogonal eigenfunctions, like their PCA counterpart, are uncorrelated and arranged in descending order of variation explained. Consequently, the expansion seen in Equation 3.7 reduced to a lower rank approximation $\tilde{u}$ using only r components

$$\tilde{u}(x) \sim m(x) + \sum_{i=1}^{r} \alpha_i \sqrt{\lambda_i} \phi_i(x) \qquad \alpha \sim N(0,1). \tag{3.8}$$

The model formulation in Equation 7.1 is a finite dimensional, parametric model analogous to a continuous generalisation of PDM. In contrast, the covariance structure of a GPMM no longer needs to be estimated from data, but can be specified theoretically using a positive semi-definite function k (Lüthi et al., 2018). In the next section kernel functions shall be discussed in more detail.

3.7.3 Kernel functions

The kernel function defines the covariance structure without estimation from data, making Gaussian processes a popular surface registration technique (Lüthi et al., 2018). Wang et al., 2015 suggests the use of kernels, improves the capability of modelling non-linear feature relationships. A kernel function can be user specified, but required to be a positive semi-definite function. A positive semi-definite kernel function $k : \mathbf{\Omega} \times \mathbf{\Omega} \to \mathbb{R}^{d \times d}$ in d dimension gives rise to a positive semi-definite matrix. A real matrix $\mathbf{K}_{n \times n}$ is positive semi-definite when satisfying

$$\mathbf{v}^T \mathbf{K} \mathbf{v} \geq 0 \qquad \forall\ \mathbf{V} \in \mathbb{R}^n,$$

where $\mathbf{v}$ is a real valued column vector. Lüthi et al., 2018 defines the simplest Gaussian process model to have a zero mean $m(x) = \mathbf{0}$ and a kernel function that enforces smooth deformations. The zero mean assumes that the reference shape is representative to the class of shapes and close to the theoretical mean shape (Lüthi et al., 2018). The simple scalar valued Gaussian kernel, also known as the radial basis function kernel

$$k_g(x, y) = \exp\left(-\frac{||x - y||^2}{\sigma^2}\right),$$

where the kernel is symmetric and defined by the parameter σ^2. The parameter σ^2 defines the range over which the deformation are correlated, with larger values for σ^2 resulting in more correlated smooth deformations (Lüthi et al., 2018). To model shape deformations in 3D, a matrix valued kernel

$$k_g(x, y) = s \cdot \mathbf{I}_{3 \times 3} k_g(x, y),$$

is required where the identity matrix $\mathbf{I}_{3 \times 3}$ signifies the x, y, z components (Lüthi et al., 2018). The parameter $s \in \mathbb{R}$ determines the scale of the deformations and is scalar multiplied. These parameters are obtained through cross-validation.

Point distribution models are criticised for inducing bias towards their training examples and not representing the shape space accurately (Le, Kurkure, and Kakadiaris, 2013). The covariance structure found though PCA, as seen in Section 3.4, can be

used as a linear, empirical kernel function (Lüthi et al., 2018). The Gaussian process $GP(m_{PDM}, k_{PDM})$ is defined with mean and covariance function

$$m_{PDM}(x) = \frac{1}{n}\sum_{i=1}^{n} u_i(x)$$
$$k_{PDM}(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(u_i(x) - m_{PDM}(x))(u_i(y) - m_{PDM}(y))^T,$$

where u_i and m_{PDM} is the i-th shape and mean shape, respectively. The result is a continuous analog to a PCA based shape model (Lüthi et al., 2018). Lüthi et al., 2018 explains the real flexibility of Gaussian process is in combining kernel functions. Hence, the empirical kernel can be used to augment smooth, non-linear kernels by inducing bias

$$k_{bs}(x,y) = k_{PDM}(x,y) + s \cdot \mathbf{I}_{3\times 3}\exp\left(-\frac{||x-y||^2}{\sigma^2}\right).$$

Lüthi et al., 2018 assumes this model's error to be spatially correlated, which suggests that if the model is unable to explain the structure empirically, then the neighbouring points deformation is likely to be smooth. User defined kernel function are able to incorporate expert insights to the model. However, the Karhunen-Loeve, seen in Equation 7.1, requires estimation of the eigenfunction $\phi_i(x)$ and eigenvalues λ_i of the kernel function, discussed in Section 3.7.4.

3.7.4 Nyström approximation

For simple kernels, the eigenfunction $\phi_i(x)$ and eigenvalues λ_i have closed form solutions (Amit, Grenander, and Piccioni, 1991; Grenander and Miller, 1998), for more complex kernels a numerical approximation is required. The Nyström numerical approximation method is used to calculate the eigenfunction, eigenvalue pairs $(\phi_i(x), \lambda_i)_{i=1}^r$ for the low rank approximation in Equation 7.1. Lüthi et al., 2018 explains that this is done by approximating the integral

$$\lambda_i\phi_i(x') = \int_\Omega k(x,x')\phi_i(x)d\rho(x), \tag{3.9}$$

which the pairs $\lambda_i\phi_i(x')$ satisfy the equation. Letting $d\rho(x) = p(x)dx$, where $p(x)$ is a density function defined on the domain Ω, points may be randomly sampled (Lüthi et al., 2018). These n sampled points of x', in the Equation 3.9, lead to the matrix eigenvalue problem

$$Ku_i = \lambda_i^{mat}u_i,$$

where $K(x_i, x_l)$ is the kernel matrix, u_i the eigenvector and λ_i^{mat} the corresponding eigenvalue (Lüthi et al., 2018). The eigenvalue λ_i^{mat} approximates the λ_i required, while the eigenfunction ϕ_i is approximated with

$$\tilde{\phi}_i(x) = \frac{\sqrt{n}}{\lambda_i^{mat}}k_X(x)u_i \approx \phi_i(x),$$

where $K_X(x)$ defines the block kernel matrix of the sampled values (Lüthi et al., 2018). This approximation improves with the sample size of n; however, increasing the sample size may be computationally infeasible. Therefore, a random SVD is used to efficiently approximate the eigenvalue, eigenvector pairs (Li, Kwok, and Lü, 2010). Further details regarding the Nyström approximation can be found through the work of (Williams and Rasmussen, 2006; Li, Kwok, and Lü, 2010; Lüthi et al., 2018). Lüthi et al., 2018 argues that the error of the approximation is assumed to be negligible and allows for efficient computation.

3.8 Generalised Procrustes Analysis

Ross, 2004 explains that generalised Procrustes analysis (GPA) is the evaluation of k sets of shapes where the sets are optimally superimposed. In shape analysis, rigid alignment is an important preprocessing step such that all observations comply with the definition of shape. Shape is all geometric information of an object once the location, orientation and scale of the object has been removed (Kendall, 1977; Cootes, Baldock, and Graham, 2000). Thus, generalised Procrustes analysis removes translation, rotation and scaling between observations such that only shape remains (Gower, 1975; Goodall, 1991).

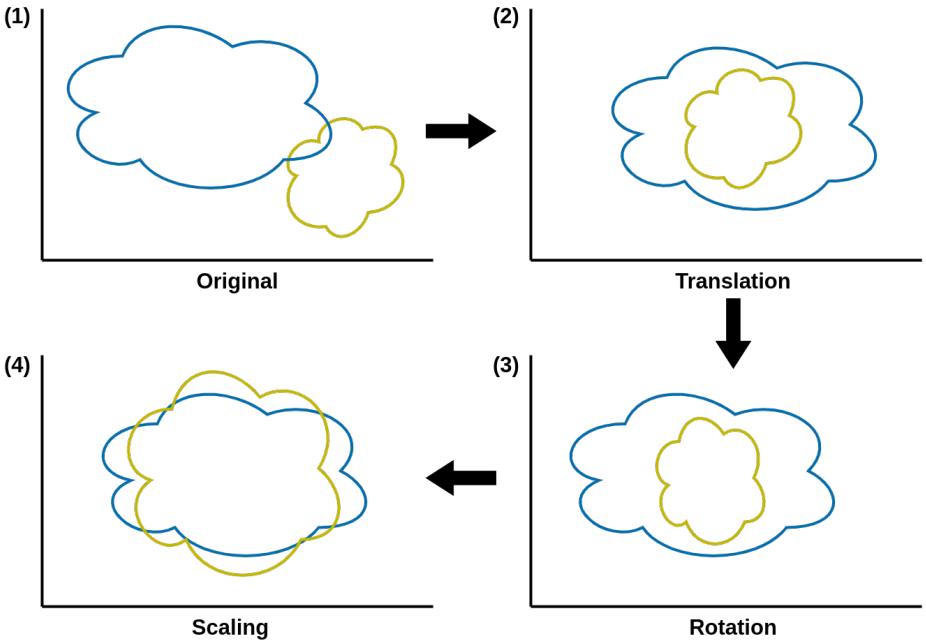

Figure 3.13: The process of rigid alignment is visualised given two anatomical shapes (1). The larger, blue observation is set as a target whereas the smaller gold observation is translated to a common center (2). The gold observation is rotated to align the two shapes (3). Finally, the shapes are uniformly scaled (4).

Figure 3.13 displays the process of rigid alignment for two anatomical shapes. The blue shape is set at the target and the gold shape is translated (2), rotated (3) and uniformly scaled (4) to rigidly align the observations. The function seen in Algorithm 1 outlines the GPA process by iteratively aligning all shapes to their mean (Ross, 2004). Initially, each observation is translated to a common center. Subsequently, the observations are rotated to align them to the mean shape. Finally, an isomorphic scaling maintains the proportions of the observations to scale them to a similar size.

Algorithm 1: Generalised procrustes analysis

```
Function generalisedProcrustesAnalysis(x_unaligned):
    /* x_unaligned is raw unaligned shape data                          */
    n = Number observations in x_unaligned
    currentMean = x_unaligned[0, :]      // Randomly initialise current mean shape
    updatedMean = x_unaligned[1, :]      // Randomly initialise updated mean shape
    x_aligned = x_unaligned                          // Initialise aligned shapes
    /* Iterate until convergence                                        */
    while currentMean ≠ updatedMean do
        currentMean = updatedMean                    // Set current mean shape
        for i in n do
            /* Set current shape                                        */
            currentShape = x_aligned[i, :]
            /* Apply translation to mean shape                          */
            currentShape = shapeTranslation(currentShape, currentMean)
            /* Apply rotation to mean shape                             */
            currentShape = shapeRotation(currentShape, currentMean)
            /* Apply scaling to mean shape                              */
            currentShape = shapeScaling(currentShape, currentMean)
            /* Update aligned shapes                                    */
            x_aligned[i, :] = currentShape
        updatedMean = shapeMean(x_aligned)           // Set updated mean shape
    /* Return aligned shape data                                        */
    return x_aligned
```

Ross, 2004 justifies GPA by its simplicity and ease of implementation. However, the process is not resistant to outliers or guaranteed to converge. Thus, a convergence threshold is set when the change in mean shape is significantly small (Ross, 2004).

3.9 Shape Model Metrics

The most common measures of dissimilarity between shapes $\mathbf{\Gamma}$ and $\mathbf{\Gamma}^*$ are functions of pointwise Euclidean distance (Davies et al., 2003; Styner et al., 2003). In 2D, the Euclidean pointwise distance between two corresponding shapes $\mathbf{\Gamma}$ and $\mathbf{\Gamma}^*$ is given by

$$
\begin{aligned}
d_1 &= \sqrt{(x_1 - x_1^*)^2 + (y_1 - y_1^*)^2} \\
d_1 &= \sqrt{(x_2 - x_2^*)^2 + (y_2 - y_2^*)^2} \\
&\vdots \\
d_n &= \sqrt{(x_n - x_n^*)^2 + (y_n - y_n^*)^2},
\end{aligned}
$$

where $\mathbf{\Gamma}$ and $\mathbf{\Gamma}^*$ have n corresponding points. Figure 3.14 displays two biological shapes and their corresponding point-wise distances d_1, d_2, d_3.

If the distances in Figure 3.14 are assigned numerical values $d_1 = 2, d_2 = 3, d_3 = 1$ the average distance would be 2. However, Cootes, Baldock, and Graham, 2000 argue that using an averaging metric smooths out large differences salient to shape modelling. In consequence, the Hausdorff distance (HD) is used as it calculates the maximum distance between shapes (Lüthi et al., 2018). For our example the a HD

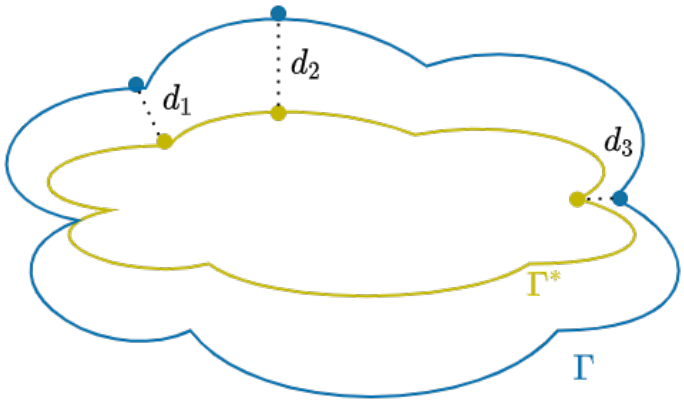

Figure 3.14: Two 2D biological shapes $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2$ are presented, described by three coordinate pairs. The shapes are assumed to be in correspondence and the euclidean distances between pairs is denoted as d_1, d_2, d_3.

returns a pessimistic value of 3. The performance of a shape model is evaluated by its ability to generalise to valid target shapes and the shape plausibility of its generative sampling. The following sections will consider both average and HD metrics while elaborating further on the evaluation approaches.

3.9.1 Generalisation

Generalisation measures a model's capability to represent unseen instances in the object class, typically calculated using a leave-one-out approach (Davies et al., 2003; Styner et al., 2003). The generalisation procedure is outlined by Algorithm 2.

Algorithm 2: Generalisation

Function `modelGeneralisation`($model$, x_{test})**:**
  /* $model$ is the trained model object */
  /* x_{test} is the testing shape data */
  n = Number observations in x_{test}
  /* Loop through n test observations */
  for j *in* n **do**
    $current = x_{test}[j, :]$ // Current observation
    $prediction$ = `predict`($model$, $current$) // Predicted observation
    $distance$ = `distance`($current$, $prediction$) // 3D pointwise distances
    $avgDist$ = `mean`($distance$) // Store the average pointwise distance
    $hausdorff$ = `max`($distance$) // Store the maximum pointwise distance
  /* Calculate the averages over all n */
  $avgDistGeneralisation$ = `mean`($avgDist$)
  $hausdorffGeneralisation$ = `mean`($hausdorff$)
  /* Return generalisation metrics */
  return $avgDistGeneralisation$, $hausdorffGeneralisation$

The function takes in a trained model and the test data, which in a leave-one-out training scheme is only a single test observation. Nonetheless, the function shows a general case for different cross-validation approaches by looping through all test observations. Pizer and Marron, 2017 explains that the model is fitted to the current

test observation and the distance between the prediction and the test data is stored. Lüthi et al., 2018 uses both average distance and HD for generalisation; thus, both metrics are stored and averaged over all testing observations. The smaller these values are the better the models ability to generalise.

A flexible model will have low generalisation; thus, fitting well to unseen examples. However, the flexible model will generate implausible shapes when sampled causing it to have a high specificity Pizer and Marron, 2017. This next section will discuss specificity in more detail.

3.9.2 Specificity

Specificity measures a model's generative ability to sample plausible instances of the same object class (Davies et al., 2003). The average distance between the generated sample and the testing data is calculated and the minimum distance among all training observations is stored (Brunton et al., 2014; Pizer and Marron, 2017). Hence, lower specificity scores result in better generative performance. The iteration parameter determines how many times the model is sampled. Lüthi and Bouabene, 2020 suggests that setting the iteration parameter to twenty should achieve stable results.

Algorithm 3 displays the method for calculating the specificity of a model. Similar to generalisation the inputs are a trained model, testing data and additionally an iteration parameter. For each iteration a random sample is generated from the model and compared to all items in the test data. In a leave-one-out training scheme there is only a single test case; however, the algorithm generalises to different schemes.

Algorithm 3: Specificity

```
1  Function modelSpecificity(model, x_test, iter):
       /* model is the trained model object                                  */
       /* x_test is the testing shape data                                   */
       /* iter is the number of iterations to sample                         */
2      n = Number observations in x_test
       /* Loop through iteration cycles                                      */
3      for l in iter do
4          randSample = sample(model)          // Randomly sample from model
           /* Loop through n test observations                               */
5          for j in n do
6              current = x_test[j, :]                    // Current observation
7              distance = distance(current, randSample)  // 3D pointwise distance
8              avgDist = mean(distance)    // Store the average pointwise distances
           /* Calculate the minimum over all n                               */
9          minDist = min(avgDist)
       /* Calculate the average over all iteration cycles                    */
10     specificity = mean(minDist)
       /* Return specificity metric                                          */
11     return specificity
```

Following the methodology of Lüthi et al., 2018, HD shall not be considered for specificity. This is due to the large variation seen in random samples which may be far from your target shape; thus, inflating the specificity unnecessarily. Moreover, the specificity calculation is computationally expensive assuming twenty iterations or more.

4 | Shape Data

This chapter provides a description of the shape data discussed in this research. Initially, shape data is defined for 3D shape objects. Thereafter, the registration pipeline for shape point-to-point correspondence is addressed. Finally, a motivation for the data used in this research is provided.

Shape is commonly defined to be all geometric information of an object once the location, orientation and scale of the object have been removed (Kendall, 1977; Cootes, Baldock, and Graham, 2000). Although inclusion of scale results in *form*, rather than shape, this definition of shape will include scale, as it is typically a salient feature in human biology-related shape analysis (Lüthi et al., 2018). Consequently, the definition of shape shall include features such as proportions, angles and size of the object.

Zheng, Li, and Szekely, 2017 explains that the shape contour is a set of connected landmark points, placed along the object boundary. A landmark point has many synonyms in literature such as: anchor point, model point, key point, and feature point. In this context, a landmark point is defined to be a point in correspondence, that marks a specific part of the contour or structure of an object, within the same shape class (Zheng, Li, and Szekely, 2017).

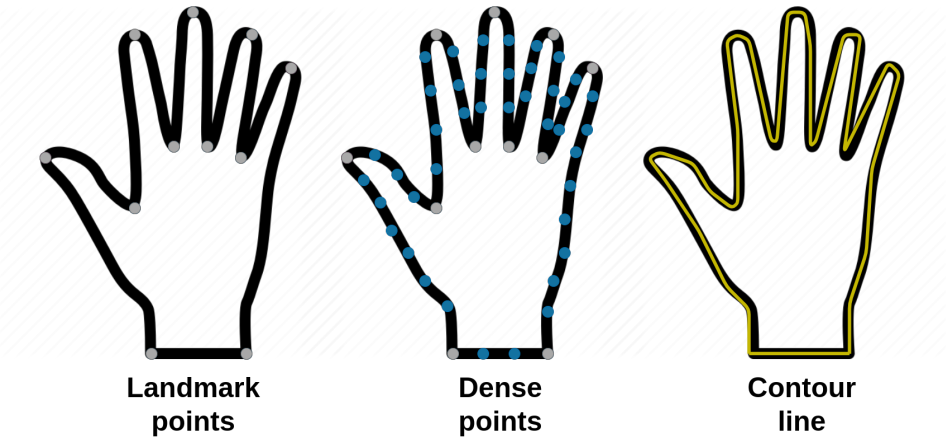

Figure 4.1: Three representations of a 2D hand shape with discrete landmark points, discrete dense points and a continuous contour line.

In Figure 4.1 the landmark points in 2D define the finger tips and turning point along the hand contour. Dense points along the boundary allow for better representations of the shape, such that infinitely many points would form a connected, continuous contour.

The contour representation can be extended to a surface in 3D. The surface coordinate points, often known as *vertices*, are connected together by *edges* to form non-overlapping triangular faces (Remondino, 2003). The collection of faces form a triangular *mesh*. Figure 4.2 represents a hand in 3D as both a point cloud and triangle mesh. The next section discusses the registration pipeline for shape point-to-point correspondence.

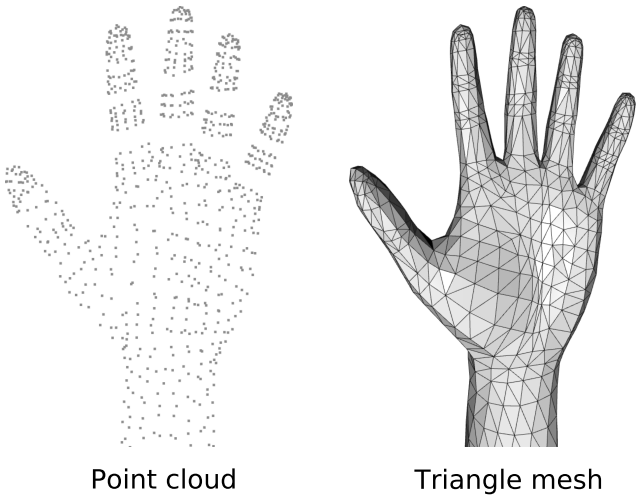

Figure 4.2: Two representations of a 3D hand shape. A cloud of coordinate points and a mesh of connected points.

4.1 Shape data preprocessing

Prior to modelling, shape data needs to be passed through a registration pipeline to establish correspondence between the shapes. This is non-essential for all shape modelling techniques, however this work will only consider shape data in correspondence. Establishing correspondence means determining a mapping of every point, on a shape, to its semantic equivalent point on another shape (Lüthi et al., 2018). Figure 4.3 shows the registration pipeline which consists of two parts namely; rigid alignment and non-rigid alignment.

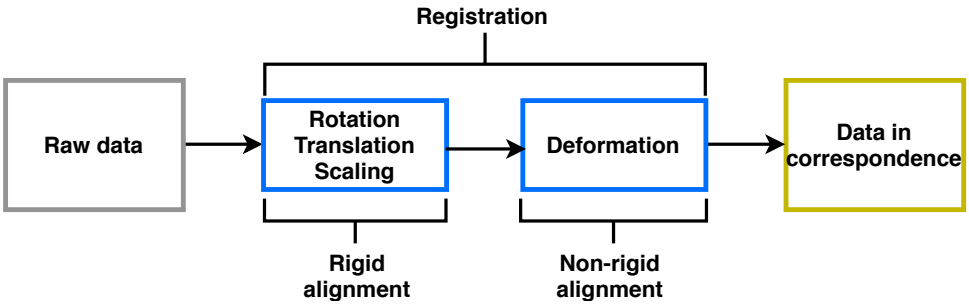

Figure 4.3: The shape data preprocessing pipeline to find correspondence between shapes.

Rigid alignment, or rigid registration, performs: translation, rotation and scaling transformations. This process aligns all target shapes $\mathbf{\Gamma}_i$, to an arbitrary selected reference shape $\mathbf{\Gamma}_R$. The uniform mapping between shapes does not undergo stretching or warping, hence regarded as rigid (Lüthi et al., 2018).

The rigid alignment process using the GPA algorithm (described in Section 3.8, which iteratively minimises the mean-squared distance between the two shapes (Gower, 1975; Goodall, 1991). In consequence, the landmark configurations are aligned and the data is fitted to a common shape space (Klingenberg, 2020).

Non-rigid alignment provides a non-uniform mapping between the target Γ_i and reference $\mathbf{\Gamma}_R$ shapes. Correspondence cannot be achieved without localised deformation of the shapes. This deforms the target in an elastic manner to find correspondence between shapes, with different geometries. Non-rigid transformations, often guided by a few manually annotated landmarks, are found by use of models such as splines or other shape models (Van Kaick et al., 2011). Thereafter, the shape data can be used for analysis.

The following sections selects two datasets for analysis: one with high suspected non-linear shape variation; and one with low suspected non-linear shape variation. Non-linearity is visualised by large landmark variations, in the form of rotations or bending deformations, between shapes (Tan et al., 2018). Thus, non-linear variation between anatomical shapes with many joints, such as human body shapes, is anticipated. Moreover, less non-linear shape variation between rigid shapes, such as bones, is expected.

4.2 Femur data

Rigid human bone datasets are regarded to exhibit linear shape related variation across the dataset (Lüthi et al., 2018; Fouefack et al., 2020). Lüthi et al., 2018; Fouefack et al., 2020 argue that the tangent space projection of linear models is sufficient for rigid shapes with a singular anatomical structure, such as bones within the same bone family. The caveat is for cases where the anatomical structure is severely diseased or has experienced trauma. Those scenarios are out of scope of this work. Visually, bones have fewer rotations and bending deformations when compared for shape differences, suggesting the shape variation is linear. Hence, both authors justify the use of linear models for human bone datasets. Thus, 3D human femur meshes shall be considered, due to their suspected low likelihood of exhibiting non-linear shape variation.

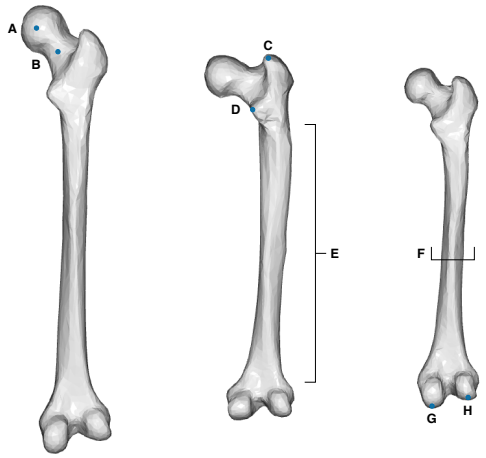

Figure 4.4: The diagram shows the posterior view of 3 different femur bones. **A**:Head, **B**:Neck, **C**:Greater trochanter, **D**:Lesser trochanter, **E**:Shaft length, **F**:Shaft width, **G**:Medial condyle, **H**:Lateral condyle.

Figure 4.4 displays the labeled posterior view of three femur meshes, selected from the dataset. The femur dataset was supplied by the course from the University of Basel, Statistical Shape Modelling: Computing the Human Anatomy (`www.futurelearn.com/courses/statistical-shape-modelling`), in STL file format (92mb in total) (Lüthi and Bouabene, 2020). The femurs were aligned using GPA, but scale between shapes was not removed as the length of the femur is a salient feature of the dataset. Correspondence was found using the Scalismo software (`scalismo.org`). Section 5.2 provides further information around the software and hardware specification in this research. The femur meshes provided were comprised of 109170 vertices and were uniformly decimated, to reduce the number of 3D vertices to 11994. The change in resolution was necessary to reduce computation requirements of the neural network architectures (10mb in total) for local computing. Finally, all femurs were uniformly scaled - thus not removing scale - such that all 3D coordinates lay between bounds of $[-1, 1]$.

The anatomical labels, in Figure 4.4, are defined to assist in describing the variation. Evidently, the variation between femur's shaft length and width are significant. Additionally, Figure 4.4 shows clear variation between the femoral head and neck orientation, between observations. Another notable source of variation is the distance and angle between the greater and lower trochanter. Finally, the orientation of the medial and lateral condyles show shape differences. We will now consider data expected to have high non linearity in shape variation.

4.3 Human body data

Human body, 3D scanned, datasets are known for their high complexity in landmark variation, due to height, mass and pose variations, between subjects. These types of data sets are often used to assess the effectiveness of non-linear methods as they display obvious rotation and bending deformations across subjects. (Bowden, Mitchell, and Sarhadi, 2000; Allen, Curless, and Popović, 2003; Brunton et al., 2014; Tan et al., 2018).

Rostami et al., 2019 discusses three open source human body mesh datasets: SCAPE, TOSCA and FAUST. The *Fine Alignment Using Scan Texture* (FAUST) dataset contains realistic deformations, built from a full body, high-accuracy, multi-stereo system. The registrations on the meshes were found through marking the subjects, with a high-frequency texture pattern, on key anatomical locations. The aforementioned process was novel as it allowed for ground truth correspondences (Bogo et al., 2014; Rostami et al., 2019). The FAUST dataset shall be selected for this research (http://faust.is.tue.mpg.de/), as it has both ground truth correspondences and realistic deformations in human pose variations.

Figure 4.5 displays the unaligned FAUST dataset. A training set of 100 meshes, in PLY file format (26mb in total), is supplied with ten subjects in ten different poses. Figure 4.5 shows a subset, where each observation portrays a single subject class, labeled $\{1, \ldots, 10\}$ in a different pose classes labeled $\{a, \ldots, j\}$. All meshes are in correspondence, with 6890 3D vertices. The subjects $1a$ and $9i$ illustrate the importance of GPA. Subject 1 is placed further forward than the other classes and the i-th pose class had clear full body rotation. Rigid alignment was conducted, without scaling as the body size was regarded an important feature and all landmarks lie between the $[-1, 1]$ bounds.

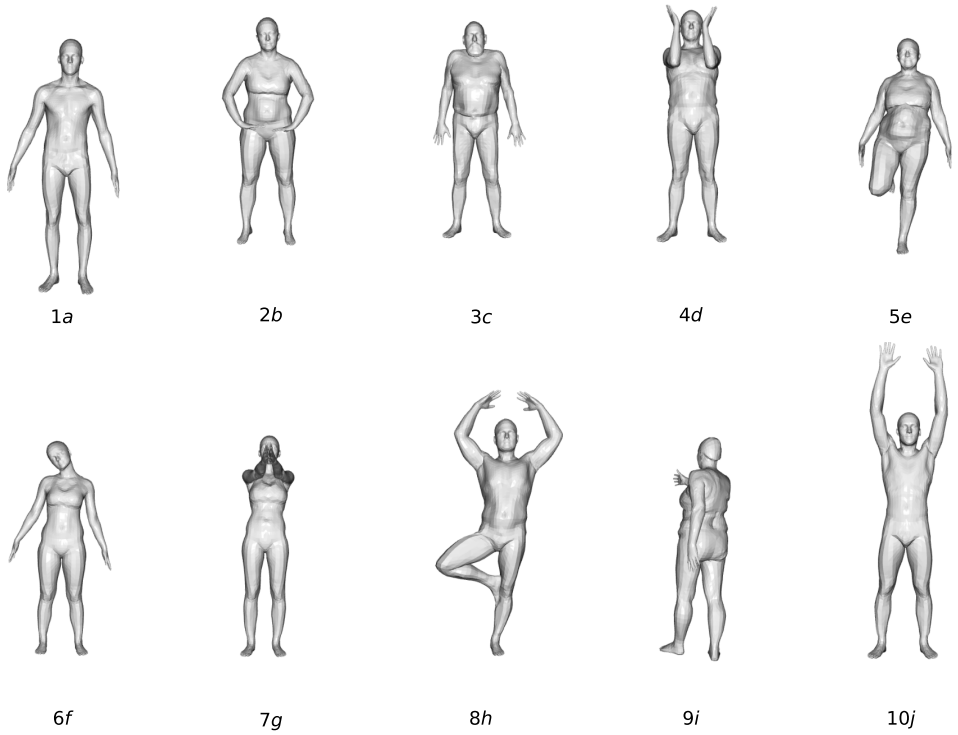

Figure 4.5: The FAUST dataset shows ten different subject classes $\{1, \ldots, 10\}$ each in ten different body pose classes $\{a, \ldots, j\}$.

The Figure 4.5 exhibits variation between the subject's body type and body position. The position variations are large rotations and bending deformations, with notable movements in the arms and the right leg of the subjects. These large landmark variations give reason to suspect that the deformations between shapes are non-linear. In Chapter 6 a framework will be developed to further identify the non-linearity in the data.

5 | Computation Specifications

The chapter outlines the hardware and software specifications used for the research. The aim is to allow for consistency and reproducibility when replicating the results presented in the study.

5.1 Hardware

Table 5.1 outlines the hardware used to conduct this research. A personal processing unit was used for all experiments for consistency.

Table 5.1: Hardware specifications

	Specifications
Model	Dell Latitude 7490
OS	Linux-Ubuntu 18.04.5
OS-type	64-bit
Memory (RAM)	16 gigabytes
Processor	Intel Core i7-8650U CPU - 1.90GHz x 8

High resolution shape data is naturally cumbersome; hence, cloud based platforms or GPU clusters are preferable for swifter computation of experiments. However, this study can be replicated using a personal computer. The next section will discuss the core programming languages used and software packages required.

5.2 Software

This section considers the software used to conduct this research. The programming languages used for this study are: Python (Van Rossum and Drake, 2009) primarily for deep learning and 3D visualisation; R (R Core Team, 2017) is used for visualisation and data wrangling; and Scala (Odersky et al., 2004) for GPMM models.

Table 5.2 categorised the aforementioned languages into three sectors: modelling, manipulation and visualisation. Modelling encompasses the software used for all models and experiments whereas, manipulation describes the main packages used to manipulate and transform the data. The visualisation sector outlines the software used to create 3D visualisations, plots and graphs.

Lüthi and Bouabene, 2020 provide a user-friendly tutorial and open source software to build and visualise Gaussian process morphable models. Therefore, the Scala based software, Scalismo, is used for implementing GPMMs. This software was developed at the University of Basel, Switzerland (`scalismo.org`). This language has

Table 5.2: Software language and package specifications

	Modelling	**Manipulation**	**Visualisation**
Python - *3.6.9*	keras - *2.3.1* scikit-learn - *0.23.1* tensorflow - *1.14.1*	numpy - *1.19.4*	open3d - *0.9.0.0* matplotlib - *3.2.1*
R - *3.6.3*		tidyverse - *1.3.0*	ggplot2 - *3.3.2*
Scala - *2.12.8*	Scalismo - *0.18*	Scala - *2.12.8*	Scalismo-ui - *0.14*

a steep learning curve and the software can be challenging to implement on different operating systems. The tutorials (`scalismo.org/docs`) and online community (`groups.google.com/g/scalismo`) are essential for a smooth implementation.

The Python programming language was used for the non-linear shape modelling paradigm. Packages such as Keras (Chollet et al., 2015), Tensorflow (Abadi et al., 2015) and Scikit-learn (Pedregosa et al., 2011), provide a simple and flexible framework for variational autoencoders and deep learning. The Open3d (Zhou, Park, and Koltun, 2018) visualisation package allows for easy displays of 3D shape objects, see (`open3d.org/`) for simple tutorials for this software.

Once all modelling has been completed the results were saved to CSV files for further analysis and visualisation. The R programming language allows for straightforward data manipulation using the Tidyverse (Wickham, 2016) software. Extensive plotting and visualisation options are provided using the GGplot2 (Wickham, 2016) framework. All code used in this research is available from (`github.com/FJFehr`) for further implementation details.

6 | Identifying non-linearity

Chapter 1 described how the quality of linear model's tangent approximation is determined by the non-linearity present in the data (Brunton et al., 2014). Non-linearity can visually be identified in data, through large scale landmark variation, such as rotations or bending deformations (Sozou et al., 1997; Bowden, 2000; Klingenberg, 2020). The femur bone and FAUST human pose datasets, selected in Chapter 4, provide a juxtaposition of visual non-linearity. Femur bones, as mentioned by Fouefack et al., 2020, are rigid shapes and visually display minor landmark variations between observations. In contrast, the FAUST dataset visually shows high landmark variation, including dynamic bending and rotational positions (Bogo et al., 2014).

The non-linearity present in the shape space can cause poor linear approximations in the tangent space (Bowden, 2000; Klingenberg, 2020). Non-linearity can be identified in the tangent space by plotting the linear projections and noting: non-linear dependencies (Sozou et al., 1994), clusters (Bowden, 2000) or if the shape parameters are not centered about the mean in tangent space (Kendall, 1989). Moreover, non-linearity can cause uncharacteristic landmark variations in the modes of variation, leading to implausible shape generation (Bowden, 2000). Therefore, it is important to consider the presence of non-linearity in the data, before undertaking model selection.

The autoencoder (AE) architecture is able to act as a linear and non-linear model (Plaut, 2018). Thus, in this chapter, an equivalence will be proved empirically between principal component analysis (PCA) and linear AEs for the 3D datasets. Thereafter, non-linear activation functions will be used while holding all other hyperparameters constant. Hyperparameters that shall remain constant will be: weight initialisation; optimiser; learning rate; regularisation; batch size; and number of epochs. See Section 3.5.3 for further details. This simple alteration will provide a comparison, which highlights the non-linearity captured by the model. Technical explanations of techniques shall be omitted in this chapter. Comprehensive explanations are provided in the theoretical background Chapter 3. All results are presented to four decimal places due to small changing values.

6.1 Linear model equivalence

The classical statistical shape model assumes that a shape, in 3D $\mathbf{s}_i \in \mathbb{R}^{3N}$, where the x, y, z coordinates are stacked, follows the univariate Gaussian distribution (see Section 3.1 for further details). Therefore, the shape dataset follows a multivariate Gaussian distribution

$$\mathbf{S} \sim N_{3N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{6.1}$$

where $\mathbf{S}_{n\times 3N}$ is a matrix representation for n training data (see Section 3.4). The distributional assumption infers that all training shapes are symmetrically distributed about the mean shape $\boldsymbol{\mu}$, with variation captured by $\boldsymbol{\Sigma}$.

The assumption of a multivariate Gaussian distribution for Equation 6.1 is often computationally infeasible to test. However, Table 6.1 considers the univariate normality of individual x, y, z coordinates and the multivariate normality of 3D $[x, y, z]$ points at a 5% significance level. The univariate Gaussian distribution was tested by using four different tests namely; Shapiro Wilk's test (Shapiro and Wilk, 1965), Jarque Bera's test (Jarque and Bera, 1987), D' Agostino's test (D'Agostino, 1972) and Lilliefors's test (Lilliefors, 1967). Whereas, the multivariate Gaussian distribution of the 3D points was tested using the Henze-Zirkler test (Henze and Zirkler, 1990). The results are reported as a proportion of individual x, y, z coordinates or 3D points that pass all tests of normality, respectively.

Table 6.1: Pointwise Gaussian Distribution Test

	Univariate normality proportion	Multivariate normality proportion
Femur	0.8714	0.9056
FAUST	0.0517	0

The results in Table 6.1 infer the Femur dataset should follow a multivariate Gaussian distribution and that the FAUST dataset does not. Roughly 90% of all 3D points for femurs follow a multivariate Gaussian distribution whereas, not a single coordinate or 3D point passes the normality tests for the FAUST dataset. The point distribution model uses PCA (described in Section 3.2) to linearly project the shape space into a lower dimensional space, which explains the linear variation in the data. Linear projected space provides an approximation

$$\tilde{\mathbf{S}}_{n\times 3N} = \mathbf{M} + \mathbf{VB}, \tag{6.2}$$

where $\mathbf{M}$ is a matrix with the mean $\boldsymbol{\mu}$ repeated on the columns, $\mathbf{V}$ a column matrix of eigenvectors and $\mathbf{B} = \mathbf{V}^T(\mathbf{S} - \boldsymbol{M})$. The shape parameters $\mathbf{b}_1, \ldots, \mathbf{b}_k$ $(k \leq 3N)$ are column vectors of $\mathbf{B}$ and are used to approximate the shape data $\mathbf{S}_{n\times 3N}$ (see Equation 3.1, for theoretical background).

Plaut, 2018 proved that a linear AE, a fully connected neural network with a single hidden layer, defined as

$$\begin{aligned} \phi : \mathbf{f} &= a_1(\mathbf{W}_1\mathbf{s}_i + \boldsymbol{\beta}_1) \\ \psi : \tilde{\mathbf{s}}_i &= a_2(\mathbf{W}_2\mathbf{f} + \boldsymbol{\beta}_2), \end{aligned} \tag{6.3}$$

could be used to approximate PCA (Plaut, 2018). Section 3.5 describes this model in detail. The reconstructed shape vertices $\tilde{\mathbf{s}}_i$, of dimension $1 \times 3N$, are individually passed through the network to train the weights $\mathbf{W}_i$ and biases $\boldsymbol{\beta}_i$. The activation functions a_1 and a_2 (Section 3.5.3) apply a linear transformation between the layers. The feature space reduction is a result of the dimension of the hidden layer $\mathbf{f}$ being less than the input dimension. Using singular value decomposition (SVD) (outlined in Section 3.3), Plaut, 2018 proved that the first k singular vectors of the weight

matrix $\mathbf{W}_2$ are equivalent to the first k eigenvectors of $\mathbf{V}$. This implies that the linear projection seen in Equation 6.2, can be recreated using linear AEs and SVD.

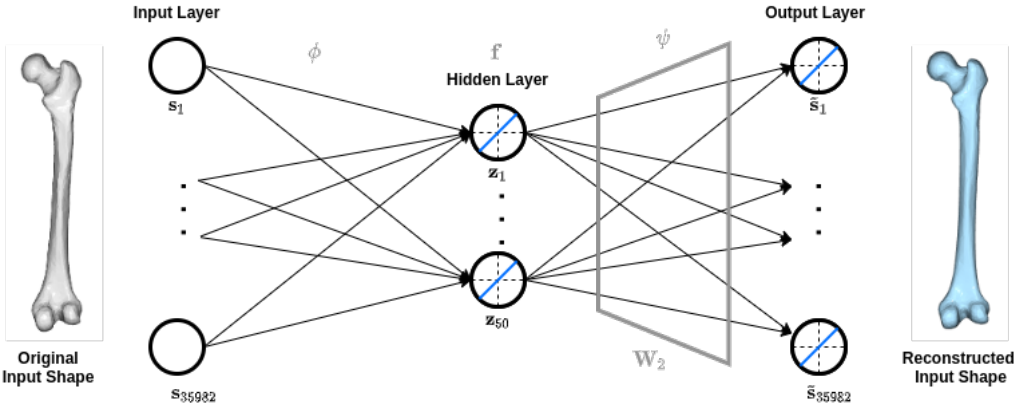

Figure 6.1: The autoecoder architecture used to replicate PCA for the Femur dataset has a single hidden layer of 50 nodes. The input and output dimension are the same and depend on the number of 3D vertices. The activation functions on both layers are linear.

The implementation from Plaut, 2018 has been adapted to 3D data, keeping the model hyperparameters the same, thus no experimentation of parameters was done. Section 3.5.3 provides more information on AE parameters and hyperparameters. The AE, as seen in Figure 6.1, has linear activations on both encoding ϕ and decoding ψ layers. The weights were initialised using a Glorot uniform initialisation. The model uses a hidden layer dimension of 50 and 100 for the Femur and FAUST datasets, respectively. The model was trained using: the Adam optimiser (Kingma and Ba, 2014); a learning rate of $1e^{-4}$; a regularisation parameter of $1e^{-5}$; a batch size of roughly a fourth of the data size; and 1000 epochs. All shapes in the respective datasets were used to train the models. The training time for the AE was roughly 210 seconds (3.5 minutes). Chapter 5 provides details regarding the computational specifications.

Figure 6.2 plots the tangent space projection for the models. The first two column vectors, $\mathbf{b}_1$ and $\mathbf{b}_2$, of the linear projection matrix $\mathbf{B}$ are plotted on the x and y axis, respectively. The tangent space projection is represented between Gaussian bounds of $[-3\sqrt{\lambda_j}, 3\sqrt{\lambda_j}]$ about the mean $\bar{x}$. Figure 6.2 confirms that the optimal solution for the linear AE is the PCA linear projection (Plaut, 2018).

Considering Figure 6.2, the Femur shape parameters (left) appear to be Gaussian distributed about the mean shape. The variations around the mean are small with few extreme values, displaying no clustering or non-linear relationships. In contrast, the FAUST tangent space (right) does not occupy a small neighbourhood around the mean shape. Table 6.2 tests the normality of the first two shape parameters, using the aforementioned tests. The Femur dataset passes all four univariate normality tests and the p-value proves that the first two shape parameters follow a bivariate Gaussian distribution. The FAUST dataset fails all normality tests, indicating that the first two shape parameters do not follow a bivariate Gaussian distribution.

The results in Table 6.2 and Figure 6.2 suggest that the linear model will cause distortions in modelling (Kendall, 1989; Kendall et al., 2009; Small, 2012).

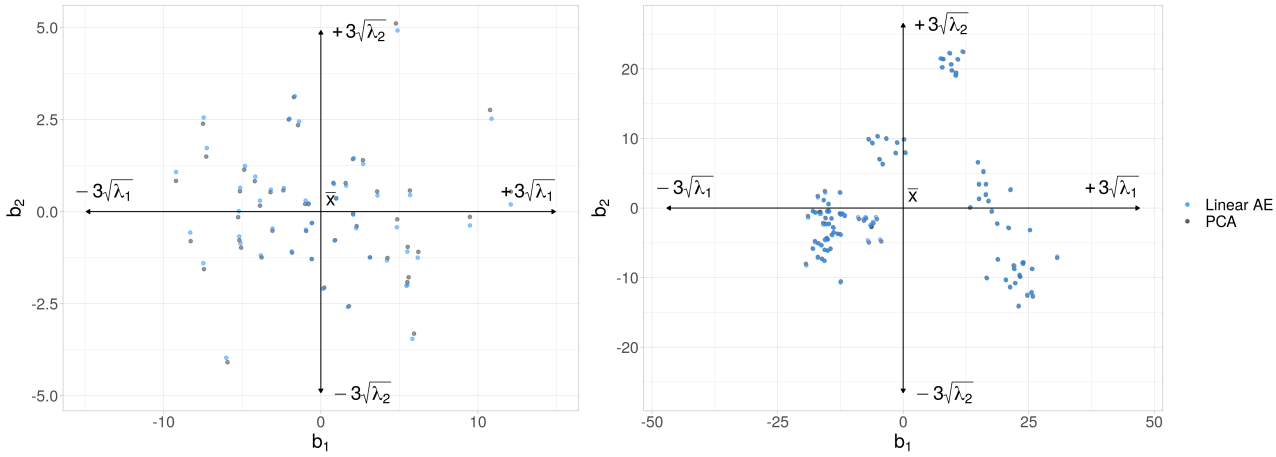

Figure 6.2: The Femur (left) and FAUST (right) datasets linear tangent space projection is visualised about the mean shape for $\mathbf{b}_1$ and $\mathbf{b}_2$. The PCA model (black) and the Linear AE (blue) are overlayed to justify their equivalence for both datasets.

Table 6.2: Shape Parameter Gaussian Distribution Test

	Univariate normality proportion	Multivariate normality p-value
Femur	1	0.6137
FAUST	0	0

Considering the FAUST shape parameters, the first parameter $\mathbf{b}_1$ is spread between $[-1\sqrt{\lambda_1}, 2\sqrt{\lambda_1}]$ and the second parameter $\mathbf{b}_2$ between $[-1\sqrt{\lambda_2}, 3\sqrt{\lambda_2}]$. The spread is not across the typical Gaussian range and form distinct clusters. The orthogonal projections display a quadratic relationship, indicating that there is non-linearity causing the dependency.

Sozou et al., 1997 suggested that the modes of variation display distortions unseen in the training data when in the presence of non-linearity. The modes of variation are displayed by varying only a single shape parameter $\mathbf{b}_j$ between $[-3\sqrt{\lambda_j}, 3\sqrt{\lambda_j}]$ while holding all others constant. When there is high non-linearity in the data, more modes of variation are needed to capture the variation. When there is low non-linearity in the data the transition between modes of variation form realistic deformations linearly about the mean (Sozou et al., 1997)

Figure 6.3 displays the modes of variation for the first three shape parameters ($\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$) of the Femur dataset. Each mode is shown as three standard deviations positively and negatively around the mean shape. The output from PCA (grey) and linear AEs (blue) capture the same variation highlighting their equivalence. When considering the tangent space projection seen in Figure 6.2 we noted that the first two shape parameters are centrally distributed about the mean. This is echoed by the mean shape being visually realistic and the modes of variation portraying plausible deformations, within the training set. The first mode captures the greatest variation in the data, which is the scale of the femur. The second and third modes of variation capture finer details, such as femur head and neck orientation, shaft thickness and

condyle orientation (see Figure 4.4 for anatomical labels).

Figure 6.4 compares the modes of variation between (PCA) (grey) and linear AE (blue), for the FAUST dataset. The tangent space projection for the FAUST dataset (Figure 6.2) revealed that the linear AE shape parameters closely approximated PCA. Consequently, the modes of variation are also closely approximated. The mean shape is placed far from other shape parameters in tangent space. This is visualised by our mean shape, in Figure 6.4, displaying unrealistic deformations in the arms, unseen in the training data. The shape parameters for the first mode of variation lie between $[-1\sqrt{\lambda_1}, 2\sqrt{\lambda_1}]$. The deformations observed between these bounds present plausible poses, with arms at a down, resting position and an arms raised, single legged position. The extremes in the first mode emphasize the unrealistic distortions, such as over extensions of the arms and uncharacteristic bending in the knee. The second mode of variation captures an orthogonal, horizontal movement in the arms. The shape parameters exist over the domain $[-1\sqrt{\lambda_2}, 3\sqrt{\lambda_2}]$. However, when all other modes are set to the mean, distortions not seen in the training data, such as small thin arms, are observed. The final mode of variation captures a walking pose, which includes a twisting rotation and the positive direction shows extreme deformations in the knee and arms.

The AE architecture, accompanied by SVD, has successfully managed to replicate the output seen from a classical point distribution model, using PCA. The FAUST dataset displays visual characteristics of non-linear dataset, such as rotation, bending and large variation not about the mean. This non-linearity was further visualised in the tangent space projections (Figure 6.2) and the modes of variation (Figure 6.4). The Femur dataset, does not exhibit the same characteristics. The next section presents how a non-linear AE could capture the non-linearity present in the data.

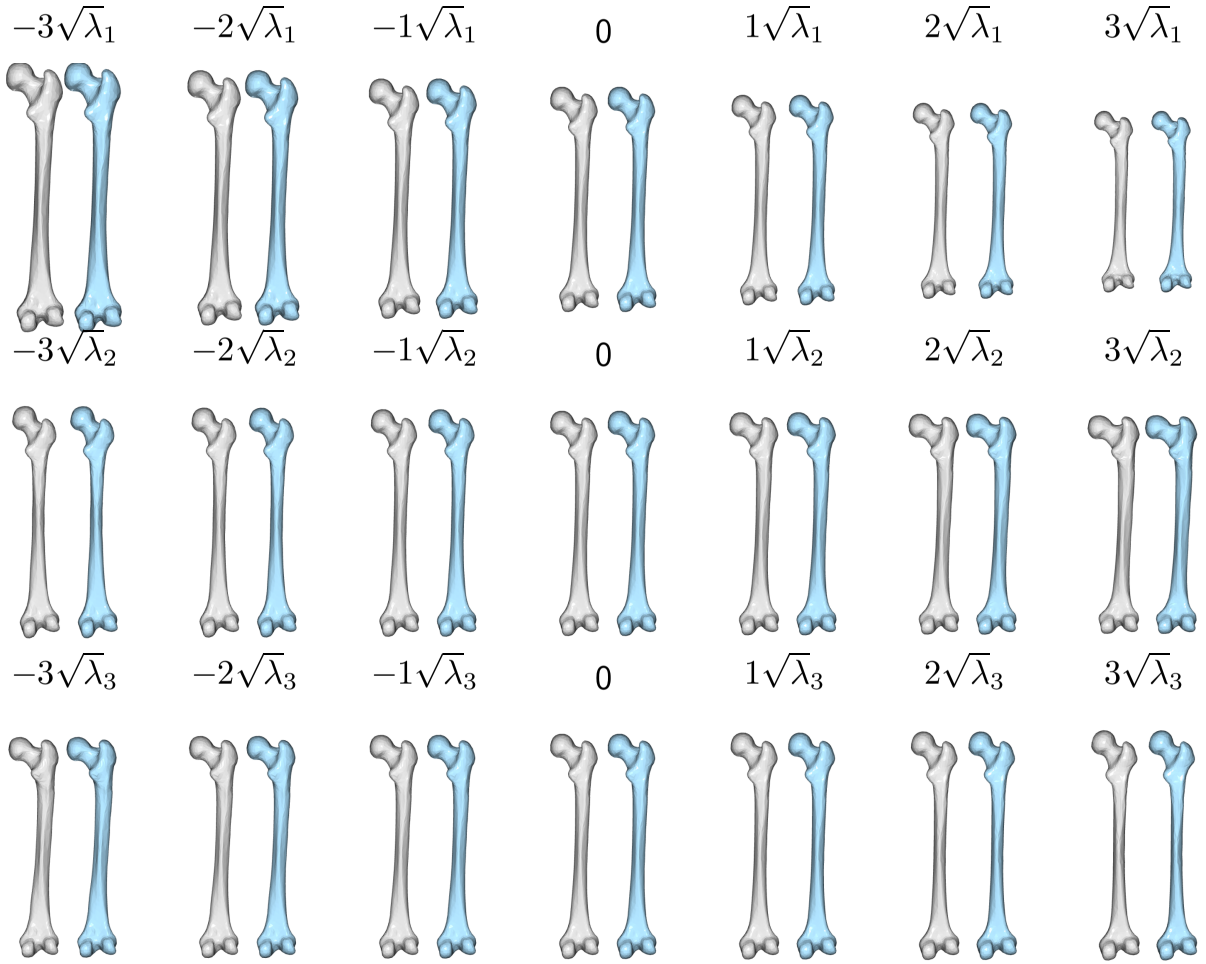

Figure 6.3: The modes of variation for the first three shape parameters are displayed for the Femur dataset. The figure rows describe the shape different parameters: the top row ($\mathbf{b}_1$), middle row ($\mathbf{b}_2$) and bottom row ($\mathbf{b}_3$). The first three standard deviations around the mean are shown for PCA (grey) and linear AE (blue).

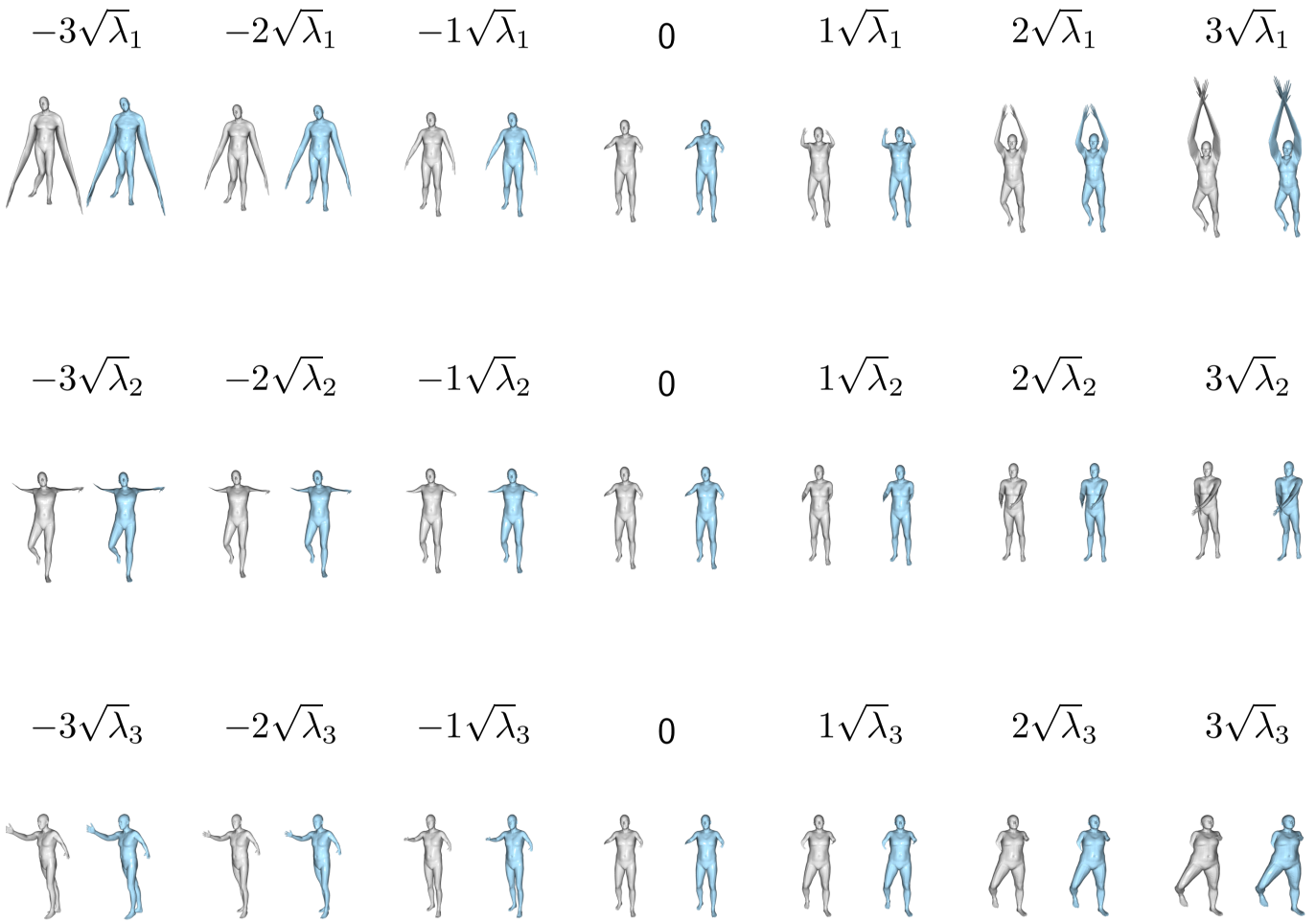

Figure 6.4: The modes of variation for the first three shape parameters are displayed for the FAUST dataset. The figure rows describe the shape different parameters: the top row ($\mathbf{b}_1$), middle row ($\mathbf{b}_2$) and bottom row ($\mathbf{b}_3$). The first three standard deviations around the mean are shown for PCA (grey) and linear AE (blue).

6.2 Non-linear model exploration

The previous section provided an equivalence between linear AEs and PCA, for the Femur and FAUST datasets. The non-linearity in the FAUST dataset was identified, initially through visual characteristics and subsequently through examining the tangent space and the modes of variation. The literature suggests that the non-linearity present in the data can be captured more effectively using non-linear models (Sozou et al., 1997; Heimann and Meinzer, 2009; Tan et al., 2018).

Autoencoders provide a flexible framework which allow for linear and non-linear models to be implemented simply by altering the activation functions. The AE, as seen in Figure 6.5, has linear activations on the encoding ϕ layer and non-linear hyperbolic tangent (tanh) activations on the decoding ψ layer (a_2 in Equation 6.3). The tanh activation is appropriate for the output layer as both datasets have been uniformly scaled to lie between $[-1, 1]$. All other training hyperparameters have been kept constant to isolate the effect of using non-linear activations. Thus, the model uses: the Adam optimiser; a learning rate of 1e4; a regularisation parameter of 1e5; a batch size of roughly a fourth of the data size; and 1000 epochs.

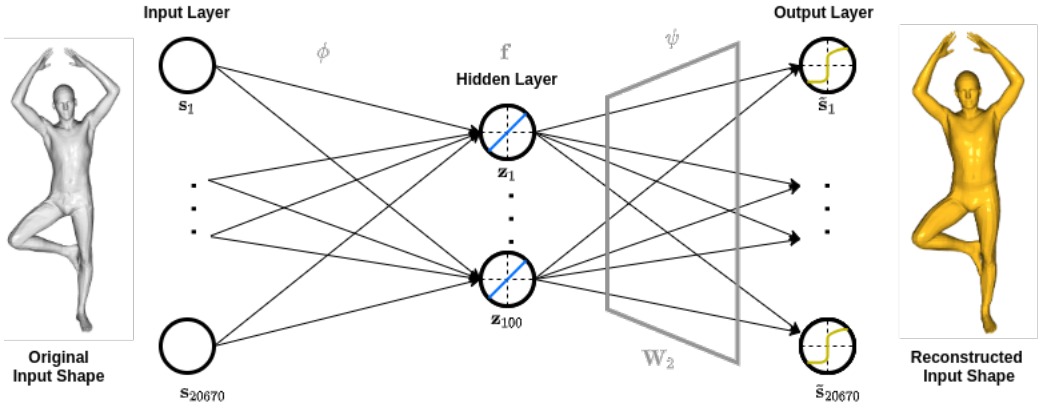

Figure 6.5: The autoecoder architecture used to replicate PCA for the FAUST dataset has a single hidden layer of 100 nodes. The input and output dimension are the same and depend on the number of 3D vertices. The linear activation function on the output layer has been substituted for the non-linear tanh function.

Figure 6.6 shows the tangent space projection for the linear and non-linear models, including a spline only to emphasize the patterns present in the two datasets. It is to be noted that the spines are not meant for prediction. Thus, the quality of fit is less important than the overall pattern observed. The Femur dataset (left) indicates minor differences between the linear and non-linear models. The shape parameters and splines are near identical suggesting the effect of the non-linear activation is negligible. This result is expected assuming the Femur data have few non-linearities. Due to the universal approximation theorem, if the shape space is linear, a non-linear AE will be able to approximate it (Hornik, 1991). The FAUST dataset (right) presents an offset rotation in the shape parameters, visualised by the spline representation. Although the non-linear model has not removed all the non-linearity present data, as suggested by Sozou et al., 1997, the model is able to capture a new set of shape parameters. This change in the shape parameters will affect the modes of variation.

Figure 6.7 show the Femur modes of variation for the linear AE (blue) and non-linear AE (gold). Just as the tangent space projection indicated minimal difference between

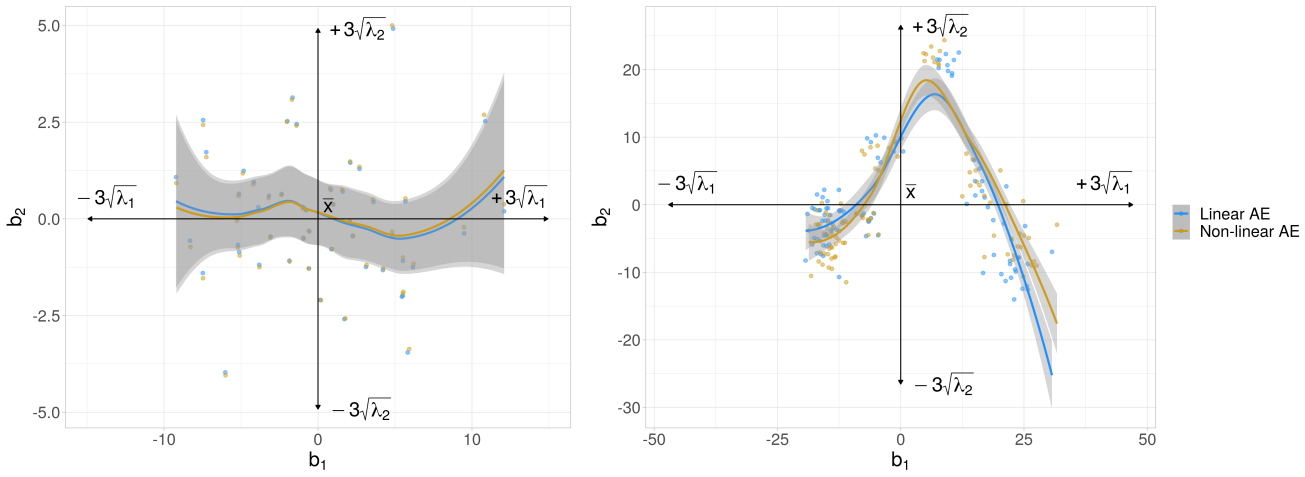

Figure 6.6: The Femur (left) and FAUST (right) datasets linear tangent space projection is visualised about the mean shape for $\mathbf{b}_1$ and $\mathbf{b}_2$. The linear AE (blue) and the non-linear AE (gold) are overlaid with a spline to contrast their differences and identify patterns.

the models, the modes of variation appear to be unchanged by the non-linear model.

The FAUST dataset modes of variation are presented in Figure 6.8 where the linear AE (blue) and non-linear AE (gold) are juxtaposed. The first mode of variation displays differences between the linear and non-linear model, although they capture similar vertical arm movement. The non-linear model appears to be capturing more information with regard to the body type. Considering the first mode of variation, the extreme positive mode shows a younger more slender figure while the alternative extreme presents an older overweight figure. The first mode of the non-linear model does not include neck or knee deformations in the positive extreme, although the negative extreme appears to be more distorted than the linear model. A clear difference between the models can be seen in the first and second modes, as the hands do not intersect for the non-linear model. These deformations are more realistic and closer to the training data. The third mode of variation captures the same theme between both models. However, the non-linear model is distorted in the negative extreme and the positive extreme captures a more natural bend in the knee.

The analysis has confirmed that the two datasets have different levels of non-linearity. The Femur dataset expresses low levels of non-linearity whereas the FAUST dataset has strong non-linear features. This was identified through: visual characteristics; examining the tangent space for non-linear dependencies; and the modes of variation showing deformations, unseen in the training data. Through a trivial adaptation of the linear AE, which was equivalent to PCA, new shape parameters were found for the FAUST dataset. The modes of variation appeared to capture more information and in some cases displayed more realistic deformations. This simple analysis demonstrates the potential of sophisticated non-linear models to capture the non-linearity in the data.

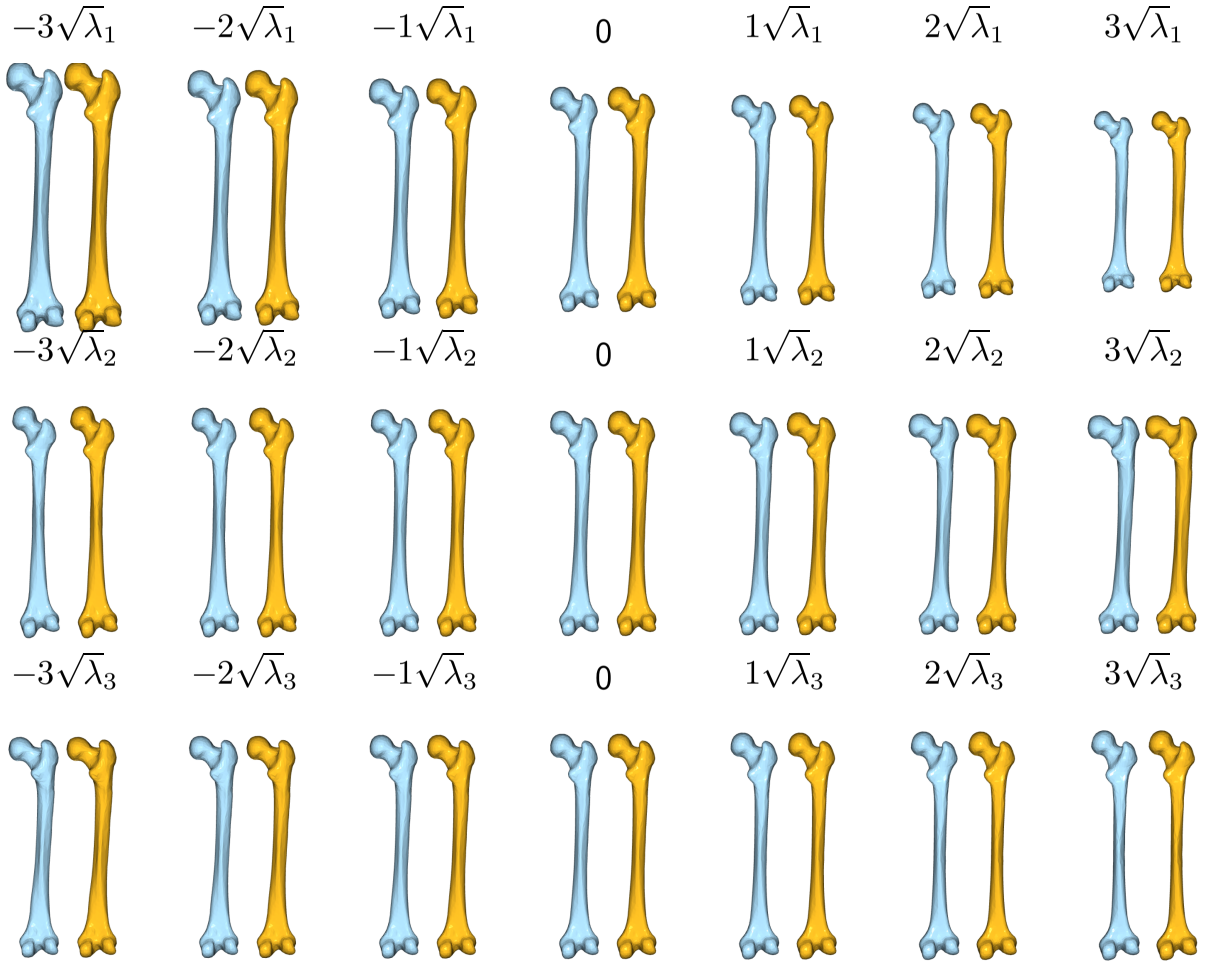

Figure 6.7: The modes of variation for the first three shape parameters are displayed for the Femur dataset. The figure rows describe the shape different parameters: the top row ($\mathbf{b}_1$), middle row ($\mathbf{b}_2$) and bottom row ($\mathbf{b}_3$). The first three standard deviations around the mean are shown for the linear AE (blue) and non-linear AE (gold).

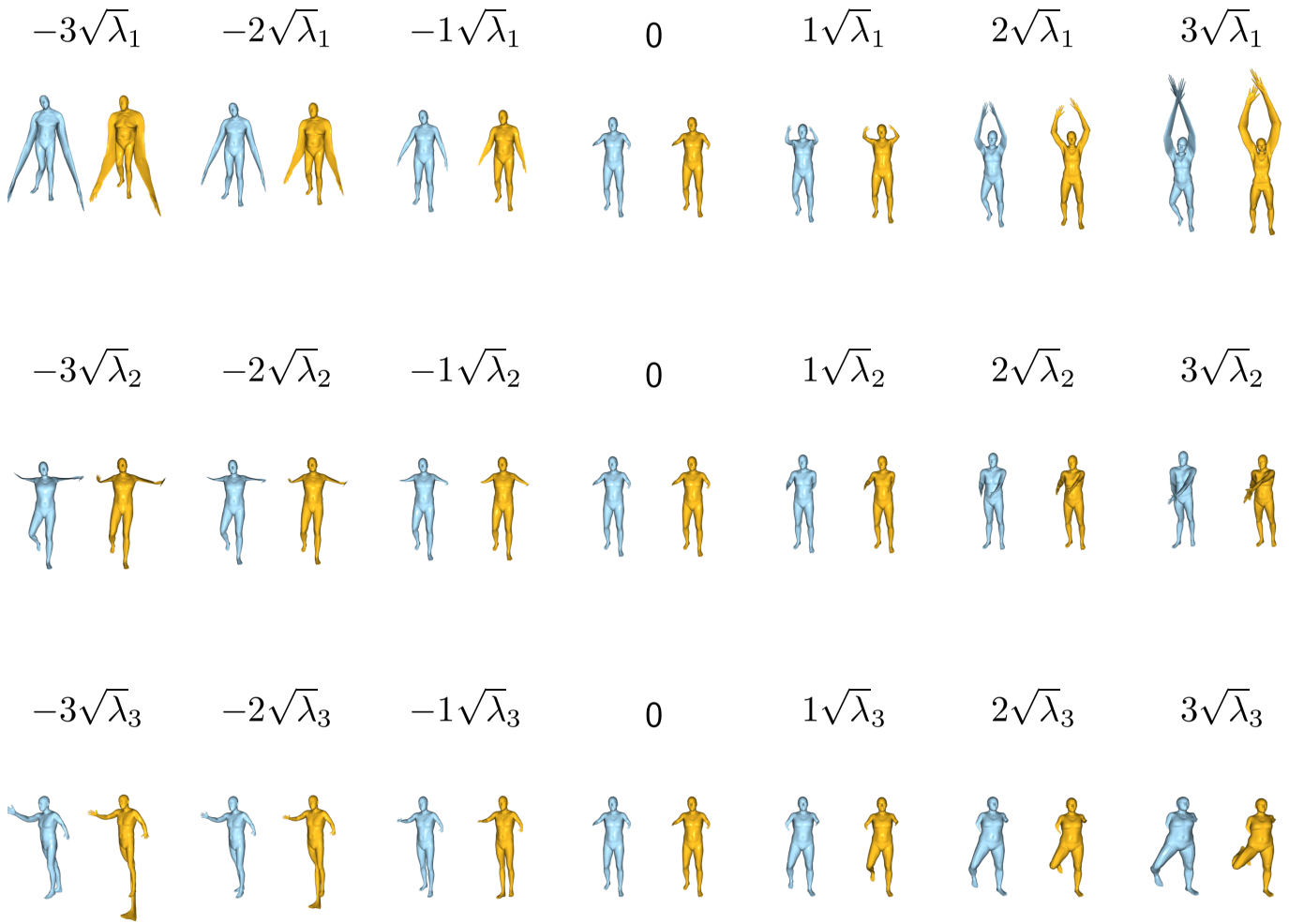

Figure 6.8: The modes of variation for the first three shape parameters are displayed for the FAUST dataset. The figure rows describe the shape different parameters: the top row ($\mathbf{b}_1$), middle row ($\mathbf{b}_2$) and bottom row ($\mathbf{b}_3$). The first three standard deviations around the mean are shown for the linear AE (blue) and non-linear AE (gold).

7 | Modelling

This chapter presents an experiment to find the optimal linear Gaussian process morphable models (GPMMs) and non-linear variational autoencoders (VAEs); and thereafter, compares their performance in model fitting, shape generation and computation time. A review of the literature, in Chapter 1, revealed a gap for a direct comparison between GPMMs and VAEs shape models, for 3D data. Previously, Chapter 6 identified that the Femur and FAUST datasets provide contrasting levels of non-linearity in shape variation. Hence, these 3D datasets would be ideal to compare linear and non-linear models. The aforementioned models will be evaluated using: Hausdorff distance (HD) generalisation; average distance generalisation; average distance specificity; and training time in seconds. Section 3.9 provides explicit explanations of the model metrics and algorithms for further perusal.

The training scheme for both linear and non-linear models is outlined in Algorithm 4. The models are trained using a leave-one-out cross-validation (LOOCV) approach, where the number of folds is equivalent to the number of observations. Typically, a hold out test set would be used to evaluate the final models; however, due to minimal observations the cross-validation error will be used as a proxy for the test error. The hardware and software specifications for the modelling, data manipulation and visualisation are described in Chapter 5.

Algorithm 4: Training scheme

Data: All shape data x

k = Number of folds

/* Loop through k folds */

for i *in* k **do**

  /* Train test split */

  $x_{test} = x[i, :]$

  $x_{train} = x[-i, :]$

  /* Train the VAE or GPMM model */

  $model =$ `Model`(x_{train}, θ) // Model parameters θ

  /* Store model training time */

  $time =$ `modelTime`$(model)$

  /* Store average distance and hausdorff generalisation */

  $generalisation =$ `modelGeneralisation`$(model,\ x_{test})$

  /* Store model specificity with 20 iterations */

  $specificity =$ `modelSpecificity`$(model,\ x_{test},\ iter = 20)$

Both Section 7.1 and Section 7.2 explore three different experiments to determine the optimal model. Each section starts with a simple baseline model and each experiment

sequentially increases the complexity of the model, in attempt to capture the non-linear variation. Due to local processing power - by use of a personal laptop, described in Chapter 5 - an extensive experimentation scheme is infeasible. Nevertheless, both sections return the best performing model, in their respective paradigm. Finally, the optimised GPMM and VAE models are compared directly for generalisation, specificity and training time.

7.1 GPMM

The literature supports the use of GPMMs for the following properties: their flexibility in model fitting; ability to generalise to different data; computational efficiency in model training; and minimal data requirements during training (Blanz and Vetter, 1999; Lüthi et al., 2018; Fouefack et al., 2020). An open source software Scalismo (`scalismo.org`) was used for the implementation of GPMM models (Lüthi et al., 2018). Although Scalismo has easy implementation, programming in Scala can be challenging, due to its functional nature (Odersky et al., 2004). A basic outline of the model is provided as a comprehensive definition of GPMMs is given in Section 3.7.

The GPMM is a continuous analog of a point distribution model (PDM) (discussed in Section 3.4). The model is defined by arbitrarily selecting a reference shape $\mathbf{\Gamma}_r = \{\mathbf{x}|\mathbf{x} \in \mathbb{R}^3\}$, which defines the 3D points domain for the model. The deformation field u, defined between the reference shape $\mathbf{\Gamma}_r$ and all other corresponding shapes $\mathbf{\Gamma}_i$, is modelled using a Gaussian process. The mean function $m : \mathbf{\Gamma}_r \to \mathbb{R}^3$ and kernel function $k : \mathbf{\Gamma}_r \times \mathbf{\Gamma}_r \to \mathbb{R}^{3\times3}$ are defined such that

$$u \sim GP(m, k),$$

allowing for a continuous representation of a multivariate Gaussian distribution. Point distribution models capture the variation between shape using the linear dimension reduction technique of PCA as described in Section 3.2. A continuous linear equivalent, known as the Karhunen-Loeve expansion, is applied with

$$\begin{aligned} \tilde{u}(x) &\sim m(x) + \sum_{i=1}^{r} \alpha_i \sqrt{\lambda_i} \phi_i(x), \\ \alpha &\sim N(0, 1), \end{aligned}$$

where λ_i and $\phi_i(x)$ are the corresponding eigenvalues and eigenfunctions, respectively. The parameter α_i are analogues to the PDM shape parameters. Similarly, the Karhunen-Loeve expansion provides a low rank approximation by using r orthogonal basis functions. Typically, model performance metrics are reported by comparing a single basis function $r = 1$ and the full model (Lüthi et al., 2018). This is valuable as it indicates the amount of variation explained by the least complex version of the model.

The mean m and covariance function k may be estimated from data - thus equivalent to the PDM - or may be user specified. The kernel function is required to be a positive, semi-definite function giving rise to a positive, semi-definite covariance matrix. A user specified kernel allows for added flexibility and non-linearity in training, external of the data (Wang et al., 2015).

The following sections conduct three experiments to find the optimal GPMM. A baseline GPMM, equivalent to a PDM, was constructed. Thereafter, the baseline GPMM was compared to a entirely functional approach. Romdhani, Gong, and Psarrou, 1999 argues that Gaussian kernels are able to capture the non-linear deformations between the shapes. The final experiment augmented the prior models in an attempt to improve the overall generalisation and specificity of the model.

7.1.1 Experiment 1: Baseline GPMM

The baseline GPMM is equivalent to a PDM; hence, referred to in the experimentation as PDM. The model uses the data to linearly estimate the mean and covariance of our distribution of points. The mean and covariance of the Gaussian process are defined as

$$\begin{aligned} m_{PDM}(x) &= \frac{1}{n}\sum_{i=1}^{n} u_i(x), \\ k_{PDM}(x,y) &= \frac{1}{n-1}\sum_{i=1}^{n}(u_i(x) - m_{PDM}(x))(u_i(y) - m_{PDM}(y))^T, \end{aligned} \tag{7.1}$$

where u_i and m_{PDM} define a shape observation and the mean shape, respectively. The 3D shape deformations u, are defined by the following Gaussian process

$$u \sim GP(m_{PDM}, k_{PDM}).$$

Thereafter, Karhunen-Loeve expansion is used to reduce the dimensions of the data, similar to the PCA model seen in Chapter 6. A full model - where r is selected to capture 99% of variation - is compared against a single basis model $r = 1$. For the full models the FAUST dataset used $r = 98$ and for the Femur dataset $r = 46$.

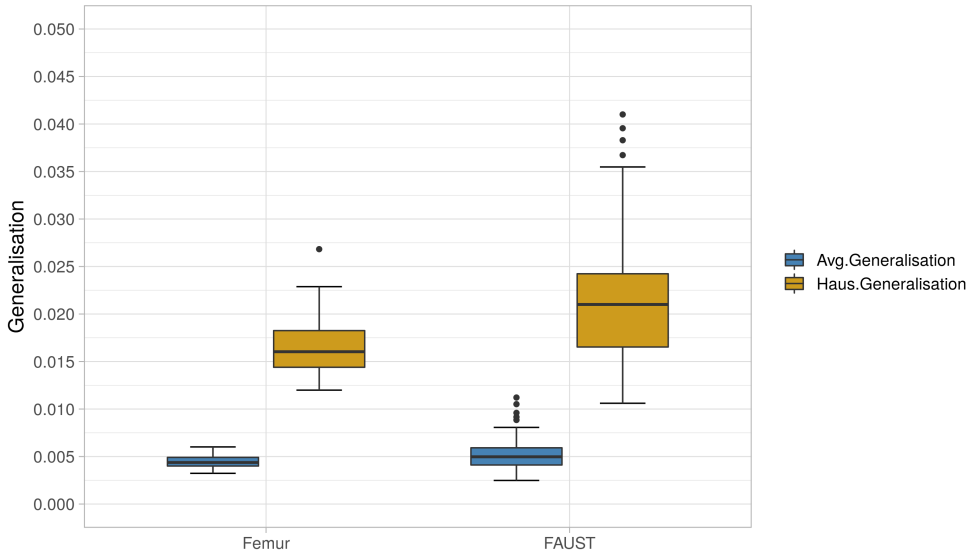

Figure 7.1: The generalisation distribution boxplots are visualised for the full model, baseline GPMMs. The plots display the generalisation using average distance (blue) and Hausdorff distance (HD) (gold), for both Femur and FAUST datasets.

Figure 7.1 reports only the full models' average generalisation and Hausdorff generalisation distributions, for both datasets. Generalisation assesses the models' ability to represent all instances in the shape family, by using a LOOCV approach. Section 3.9.1 provides an inclusive description and algorithm for generalisation. The average distance between two shapes was reported and contrasted against the maximum HD between two shapes. Lüthi et al., 2018 used boxplots to consider the distribution of the generalisation model metrics. Smaller values and tighter distributions indicate better performing models. In general, the HD is larger and more variable than the average distance. However, this was exacerbated by data with shape large variations, as seen in Figure 7.1.

The model metrics are presented in Table 7.1. A LOOCV approach was used to train the models. Thus, the average for all model metrics, across all folds, was reported. A single basis function, or first principle component (1st PC), is shown alongside the full model. The full model chooses the number of basis functions, such that 99% of variation is approximated (Lüthi et al., 2018). The training time for the full model and reduced model are equivalent, as the reduced model is a subset of the full model. Improved performance was seen by the full models for generalisation; however, this was at the cost of inferior, higher specificity. The specificity measures a model's generative ability to sample instances of the same object class. The training time may be interpreted as the average time taken, in seconds, to train a single GPMM model, during the LOOCV training scheme. The training time for these models were negligible, as all models required less than a second on average to train. However, it is important to mention that the fitting time for GPMMs is known to take more processing time than training (Lüthi et al., 2018). This was not within the scope of this study to pursue.

Table 7.1: The baseline GPMM (PDM) average model metrics are displayed for the Femur and FAUST datasets, respectively. The metrics are averaged over all cross-validation cycles.

Model	Avg.Gen	Haus.Gen	Specificity	Time (sec)
		Femur		
PDM (1st PC)	0.0248	0.0807	**0.0543**	**0.1091**
PDM (Full)	**0.0045**	**0.0166**	0.0612	**0.1091**
		FAUST		
PDM (1st PC)	0.1215	0.3540	**0.2326**	**0.7876**
PDM (Full)	**0.0052**	**0.0213**	0.2732	**0.7876**

The baseline GPMMs performance in: generalisation, specificity and training time, have been used to benchmark the following two experiments. The next experiment considered an entirely functional GPMMs using a Gaussian kernel function to capture the non-linear variation.

7.1.2 Experiment 2: Kernel GPMM

Lüthi et al., 2018 provides a comparison of different kernel functions and uses the Gaussian kernel function as a benchmark. The Gaussian kernel, also known as radial basis function kernel, is non-linear and could improve modelling performance in the presence of non-linear feature relationships (Romdhani, Gong, and Psarrou, 1999; Williams and Rasmussen, 2006; Wang et al., 2015). Romdhani, Gong, and Psarrou,

1999 justifies a Gaussian kernel, in kernel PCA, to model non-linear shape deformations. Hence, a Gaussian kernel was selected in an attempt to capture the non-linear shape variation.

The Gaussian kernel has a zero mean $m(x) = \mathbf{0}$ and estimates the covariance structure by enforcing smooth, global deformations. The kernel equation

$$k_g(x, y) = s \cdot \mathbf{I}_{3\times3} \exp\left(-\frac{||x - y||^2}{\sigma^2}\right), \tag{7.2}$$

describes the functional form. The parameter σ^2 defines the range over which the deformation are correlated, with larger values for σ^2 resulting in more correlated smooth deformations. The identity matrix $\mathbf{I}_{3\times3}$ signifies the x, y, z components (Lüthi et al., 2018). The parameter $s \in R$, determines the scale of the deformations and is scalar multiplied. For further information on kernel functions see Section 3.7.3.

More sophisticated kernels may perform better in the presence of non-linear shape variation, although for this experiment the simplest Gaussian kernel shall be implemented. Following the work of Lüthi et al., 2018, the parameters $\sigma = 100$ and $s = 100$ are used. A full parameter search for both datasets is out of scope for this work, due to computational limitations and difficulty of implementation. Hence, the parameters from Lüthi et al., 2018 are used as a benchmark and have been justified for bone data. Therefore, the 3D shape deformations u, are defined by the following Gaussian process

$$u \sim GP(\mathbf{0}, k_g^{(100,100)}).$$

Figures 7.2 and 7.3 compare the Gaussian kernel GPMM to the baseline GPMM for the linear Femur dataset. Both average and Hausdorff generalisations showed a substantial difference between the PDM and Gaussian process kernel alone. This disparity was not unexpected, as the covariance structure in the Gaussian process kernel model was only global, smooth deformations. Conversely, the PDMs deformations were informed directly by the data.

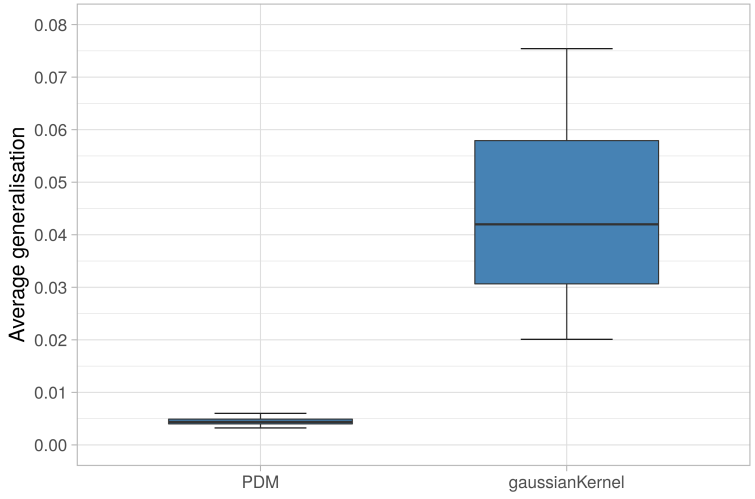

Figure 7.2: The average generalisation, for Gaussian kernel models, is compared against the Femur baseline GPMM, named PDM.

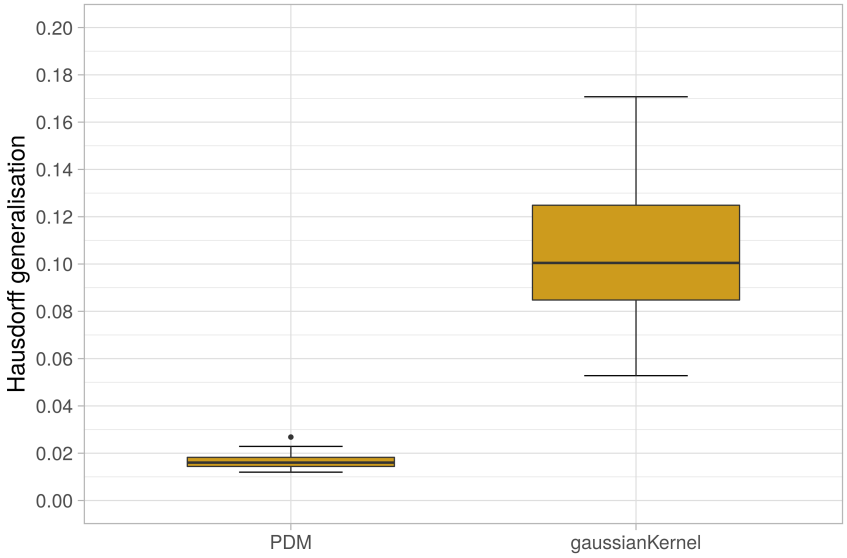

Figure 7.3: The Hausdorff generalisation, for Gaussian kernel models, is compared against the Femur baseline GPMM, named PDM.

Considering the FAUST non-linear dataset, the distributions of the average and Hausdorff generalisation showed a larger disparity. Figures 7.4 and 7.5 exhibited an increased difference in performance. Although the Gaussian kernel had non-linear properties, the large local and global deformations between observations resulted in the simple Gaussian model under-performing. For future exploration, more complex non-linear kernels or adjusting the parameters of the kernel may lead to improved performance.

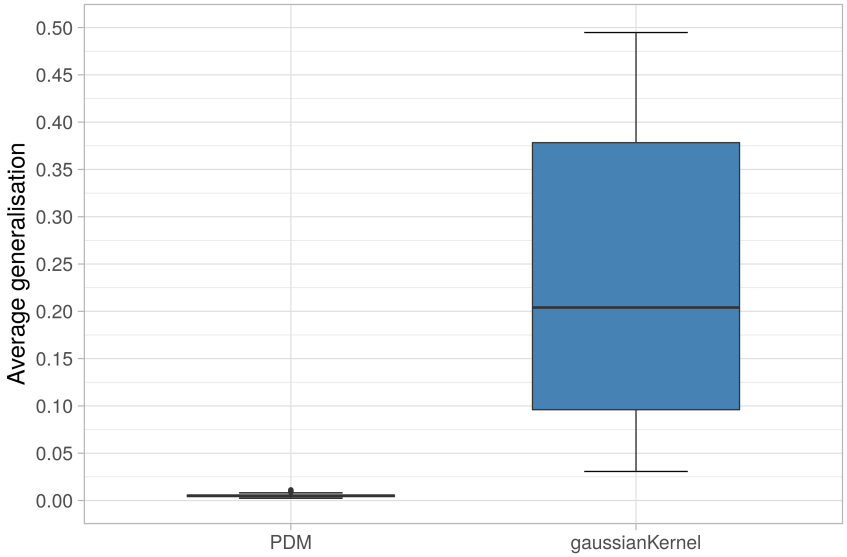

Figure 7.4: The average generalisation, for Gaussian kernel models, is compared against the FAUST baseline GPMM, named PDM.

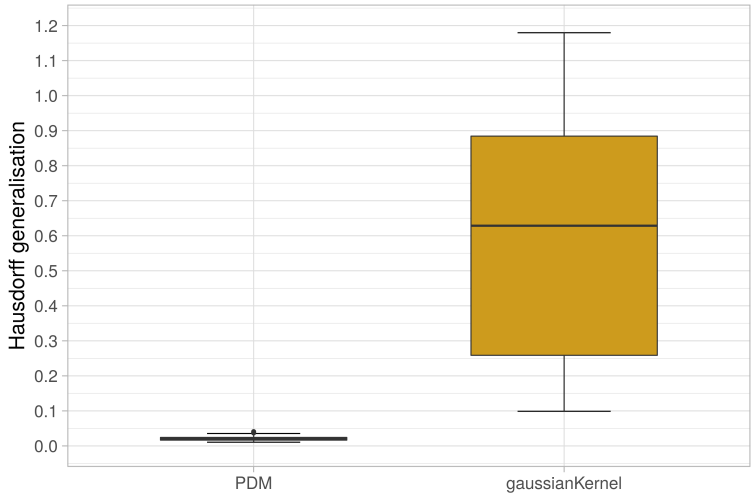

Figure 7.5: The Hausdorff generalisation, for Gaussian kernel models, is compared against the FAUST baseline GPMM, named PDM.

The average performance metrics for this experiment are summarised by Table 7.2. The single basis, Gaussian kernel models displayed an insignificant difference in generalisation, in comparison to the full model. However, we saw a clear improvement in specificity for the models with a single basis, as the full model appears to be over parameterised. When comparing the Gaussian kernel models against the baseline PDM, it was clear the baseline GPMM outperformed the purely functional model. This result was expected, but the large disparity in performance on the non-linear data was not expected. The Gaussian kernel function captures smooth global deformations, which, uninformed by data, were unable to capture the large deformations.

Table 7.2: The baseline GPMM average model metrics are displayed for the Femur and FAUST datasets, respectively. The metrics are averaged over all cross-validation cycles.

Model	Avg.Gen	Haus.Gen	Specificity	Time (sec)
		FEMUR		
PDM (1st PC)	0.0248	0.0807	**0.0543**	0.1091
PDM (full)	**0.0045**	**0.0166**	0.0612	0.1091
gaussianKernel (1st PC)	0.0597	0.1123	8.2245	**0.0893**
gaussianKernel (full)	0.0590	0.1123	16.3990	**0.0893**
		FAUST		
PDM (1st PC)	0.1215	0.3540	**0.2326**	0.7876
PDM (full)	**0.0052**	**0.0213**	0.2732	0.7876
gaussianKernel (1st PC)	0.2436	0.6203	7.8174	**0.1040**
gaussianKernel (full)	0.2435	0.6207	16.1413	**0.1040**

Evidently, the best performing model in this experiment for both datasets was the baseline full PDM. The average generalisation for the non-linear FAUST data, was roughly 50 times larger for the Gaussian kernel model, than the full PDM. In contrast, the linear Femur data only showed a disparity of 12 times. This suggests that the purely functional model had a relative, improved generalisation performance on linear data. These results may show improvements through an in-depth parameter search

for s and σ^2; however, as mentioned this was beyond the scope of the work due to difficulty of implementation. The next experiment harnesses the flexible nature of GPMMs, by combining the covariance structure of both prior models. This should allow the model to obtain information from the data and include non-linear global deformations.

7.1.3 Experiment 3: Augmented GPMM

Gaussian process morphable models provide a flexible modelling framework through the use of kernel functions. Kernel functions define the covariance structure and must give rise to a positive semi-definite matrix. Thus, kernels may be augmented by: addition of another positive semi-definite matrix; scalar multiplication; or matrix multiplication with another positive semi-definite matrix (Lüthi and Bouabene, 2020). These augmentations allow for bias and non-linearity to be introduced to the model.

This experiment combined the prior two experiments by adding their covariances. The empirical kernel, k_{PDM} from Equation 7.1, allowed the model to overfit the training data. The functional kernel, k_g from Equation 7.2, regularised it and induced bias. The biased augmented kernel function

$$k_{bs}(x, y) = k_{PDM}(x, y) + k_g(x, y),$$

used the same parameters seen in the prior experiment, $\sigma = 100$ and $s = 100$. As previously mentioned, these parameters are used directly from Lüthi et al., 2018 and may be altered. The mean functions were also added. The 3D shape deformations u, for both Femur and FAUST datasets, are defined by the following Gaussian process

$$u \sim GP(m_{PDM}, k_{bs}).$$

The combination of the prior experiments is expected to add bias to the baseline GPMM and add non-linear global deformations. Hence, improving generalisation and specificity performance metrics.

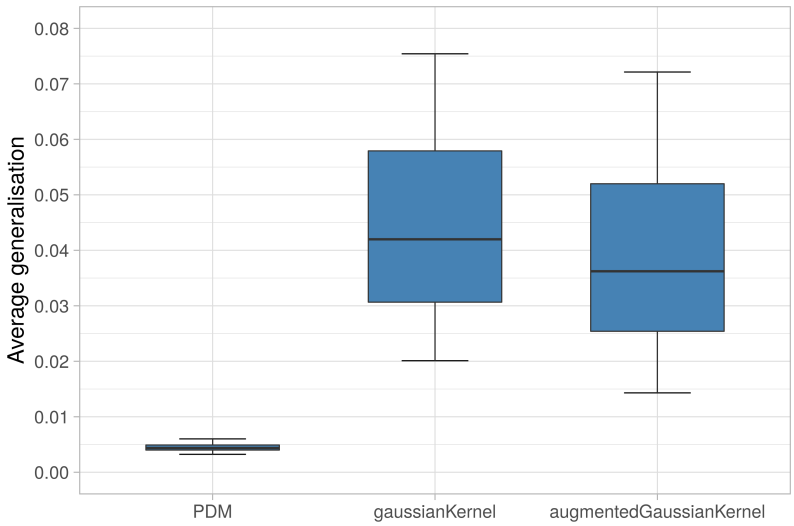

Figure 7.6: The average generalisation, for each experiment, is compared against the Femur baseline GPMM.

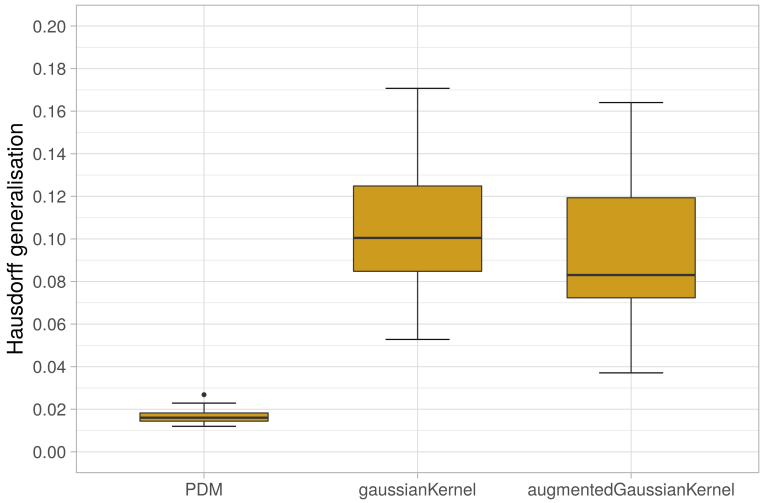

Figure 7.7: The Hausdorff generalisation, for each experiment, is compared against the Femur baseline GPMM.

The generalisation distributions for the Femur dataset are illustrated in Figure 7.6 and Figure 7.7. The augmented kernel improved both average and Hausdorff generalisation error metrics, in comparison to the Gaussian kernel. This was seen by the distribution shifting down. However, the contrast in performance to the PDM baseline model was still substantial. This suggested that the PDM performs well on the linear Femur data without the need of non-linear global deformations.

Figure 7.8 and 7.9 display a similar, and improved, trend for the non-linear FAUST dataset. The generalisation distributions for the augmented model showed smaller and tighter distributions than the Gaussian kernel. As expected the augmented model had less variance and improved performance on non-linear data than the Gaussian kernel model. Although the augmented model outperformed the Gaussian kernel, the model was still far from the performance of the PDM. This was a result of the local and global non-linear deformations present in the FAUST dataset. Improvements may be seen through deeper parameter exploration for s and σ^2 or complex kernels.

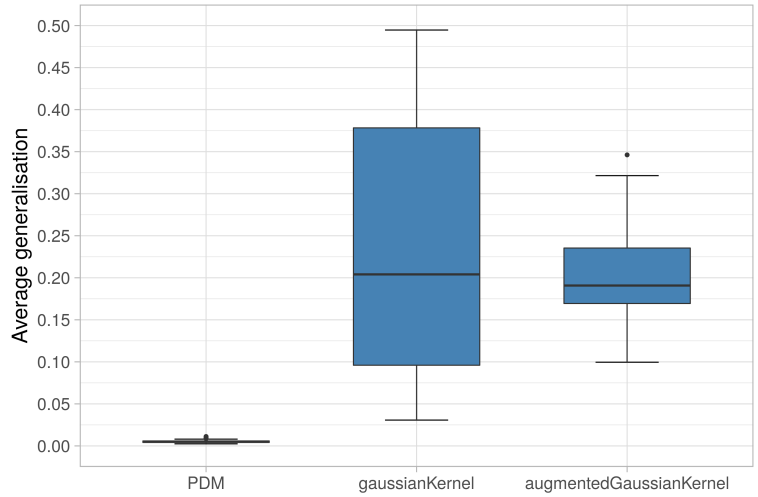

Figure 7.8: The average generalisation, for each experiment, is compared against the FAUST baseline GPMM.

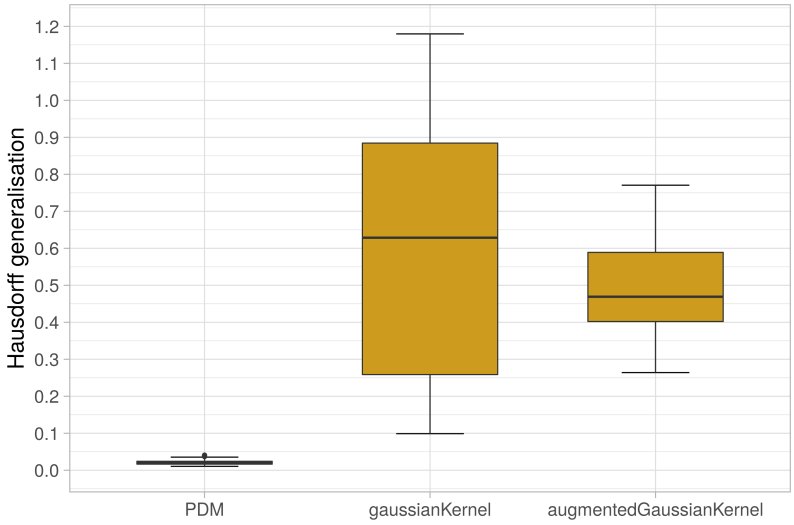

Figure 7.9: The Hausdorff generalisation, for each experiment, is compared against the FAUST baseline GPMM.

Table 7.3 summarises the results for all GPMM experiments. The improvements in generalisation, seen in the distribution plots, were echoed in the average metrics. The augmented, full model had marginally better specificity than the full Gaussian kernel model, across both datasets. Despite this, the specificity in these experiments were considerably inferior to the baseline GPMM. The processing time for the FAUST augmented model, although minimal, was relatively higher than all other models. In conclusion, the best experiment for both Femur and FAUST datasets was the baseline GPMM using a linear kernel estimated from data.

Table 7.3: The baseline GPMM average model metrics are displayed for the Femur and FAUST datasets, respectively. The metrics are averaged over all cross-validation cycles.

Model	Avg.Gen	Haus.Gen	Specificity	Time (sec)
	FEMUR			
PDM (1st PC)	0.0248	0.0807	0.0543	0.1091
PDM (full)	**0.0045**	**0.0166**	**0.0612**	0.1091
gaussianKernel (1st PC)	0.0597	0.1123	8.2245	**0.0893**
gaussianKernel (full)	0.0590	0.1123	16.3990	**0.0893**
augmentedGaussianKernel (1st PC)	0.0450	0.0943	8.0362	0.5839
augmentedGaussianKernel (full)	0.0441	0.0925	15.9672	0.5839
	FAUST			
PDM (1st PC)	0.1215	0.3540	**0.2326**	0.7876
PDM (full)	**0.0052**	**0.0213**	0.2732	0.7876
gaussianKernel (1st PC)	0.2436	0.6203	7.8174	**0.1040**
gaussianKernel (full)	0.2435	0.6207	16.1413	**0.1040**
augmentedGaussianKernel (1st PC)	0.2027	0.4910	7.9841	3.5572
augmentedGaussianKernel (full)	0.2027	0.4912	15.8455	3.5572

7.1.4 Conclusion

The optimal GPMM, for both Femur and FAUST datasets, has been finalised through three experiments: a baseline GPMM equivalent to a PDM; an entirely functional model using a Gaussian kernel; and finally an augmented model that combines both prior experiments.

The experiments showed that, given minimal data, augmented models can improve the generalisation performance over entirely functional Gaussian kernel models. This was particularly notable for non-linear data such as the FAUST dataset as the augmented kernel improved the distribution of the average and Hausdorff generalisation and substantially improved the specificity. The Gaussian kernel is able to capture global non-linear deformation; however, for the FAUST dataset the deformations are both local and global with complex variations. The simple model was unable to provide improvements in comparison to the linear baseline GPMM. Although out of scope of this research, improvements may be seen with more sophisticated kernels and training schemes.

On the linear Femur data, the entirely functional Gaussian kernel GPMM performed relatively better than the non-linear dataset. The global deformations between femur bones, such as femur length, were more easily captured by the functional model than the complex non-linear deformations present within FAUST. As expected the best performing model for the linear Femur data was the baseline GPMM.

The optimal GPMM, for both datasets, was found to be the linear baseline GPMM. This model is equivalent to a PDM and estimates the parameters directly from the data. This model outperformed all other experiments across all metrics. Although the non-linear Gaussian kernel models did not improve the performance, the simple experiments were a valuable comparison and showcased the flexibility of the GPMM framework. The experiments could be improved through an extensive parameter search or considering multi-scale Gaussian kernels that capture both global and local deformations (Lüthi et al., 2018). The baseline GPMM will be used to compare against the optimal VAE in the next section.

7.2 VAE

Variational autoencoders are structurally similar to autoencoders (AEs), but differ from the latter in that they encode the input as a distribution over the latent space (Kingma and Welling, 2013). A comprehensive theoretical background for VAE is presented in Section 3.6; hence, summarised in this section. The latent distribution is assumed to follow a Gaussian distribution (Section 3.1). This assumption provides a continuous space for probabilistic sampling. In training, the latent distribution is enforced through Kullback-Liebler (KL) divergence (Section 3.6.2), which regularises the mean squared error (MSE) loss function. Equation 7.3 describes the VAE loss function

$$\mathcal{L}(\theta, \omega) = -[\underbrace{\mathbb{E}_{q_\theta}[\log(p_\omega(\mathbf{x}|\mathbf{z})]}_{(1)} - \underbrace{D_{\mathrm{KL}}(q_\theta(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{(2)}], \tag{7.3}$$

where the weights for the probabilistic encoder θ and decoder ω are optimised by minimising the loss function. The variational lower bound is split into two parts: (1) the likelihood of the data, defined by the reconstruction loss MSE; and (2) the KL divergence, which guides the latent distribution of the encoder to return a standard Gaussian distribution. Odaibo, 2019 derives the KL divergence for a standard Gaussian distribution

$$D_{\mathrm{KL}}(q_\theta(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \sum_{j=1}^{J} \frac{1}{2}[1 + \log(\sigma_j^2) - \sigma_j^2 - \mu_j^2],$$

where J is the latent vector's dimension and the approximate distribution $q_\theta(\mathbf{z}|\mathbf{x})$, is parameterised by mean μ_j and variance σ_j^2. Apart from the aforementioned differences, the VAE architecture inherits all hyperparameters from AEs, outlined in Section 3.5.3.

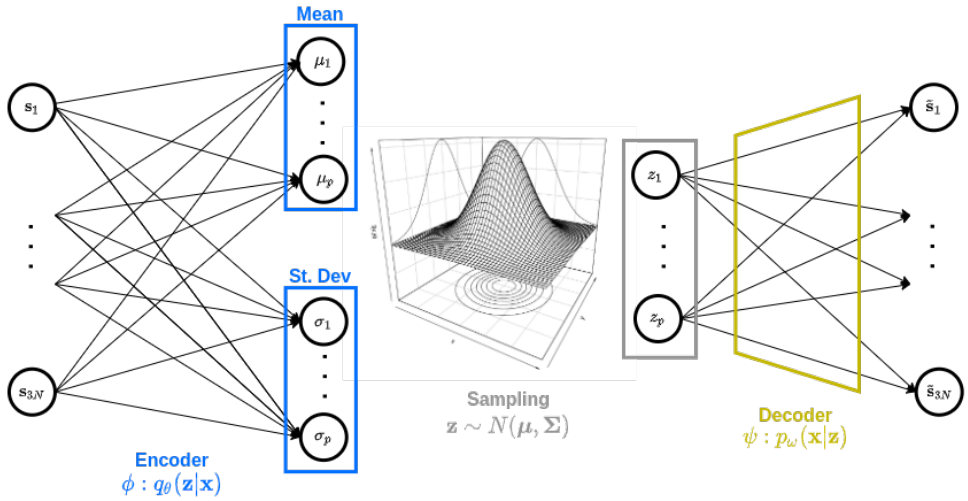

Figure 7.10: A single layer VAE visualisation for a 3D shape vector.

Figure 7.10 provides a visual presentation of a single layer VAE architecture. Although trivial, the input vector $\mathbf{s}_i \in \mathbb{R}^{3N}$ uses 3D shape vertices, where the x, y, z coordinates are row stacked. Each shape observation, in the dataset, is encoded as

latent, parameter vectors $\boldsymbol{\mu}_{p\times 1}$ and $\boldsymbol{\sigma}_{p\times 1}$. During a forward pass of the data, the parameters define the latent, multivariate Gaussian distribution from which is sampled. Thereafter, the sampled latent vector $\mathbf{z}$ is decoded to provide a reconstruction of the input shape $\tilde{\mathbf{s}}_i$.

Computation time for training hinders the application of deep models, such as VAEs (Sozou et al., 1997; Fouefack et al., 2020). The hyperparameters, size of the network and complexity of the data, influence the amount of computation required to train the model weights. A local optimum is determined, for each experiment, by examining the training and validation loss over all epochs. Increasing the number of epochs invariably increases the training time and decreases the training loss. Therefore, a local optimum is found when the validation error stabilises, visualised by a plateau or horizontal trajectory.

The following sections conduct three sequential experiments to find the optimal VAE hyperparameters. The first experiment is a baseline VAE, using the same hyperparameters considered in Chapter 6. The second experiment compares the baseline VAE to deep architectures, exploring the number of layers and the number of nodes hyperparameters. This added complexity is expected to improve the ability to capture non-linear shape variation Tan et al., 2018. Finally, the third experiment uses the best model structure in the second experiment to consider different weight initialisations, activation functions and number of training epochs required.

7.2.1 Experiment 1: Single layer VAE

A single layer, VAE model shall be used as a baseline for further experiments. Building from the non-linear, AE model in Chapter 6, the baseline model used the same hyperparameters. Both Femur and FAUST models were trained using: the Adam optimiser; a learning rate of $1e^{-4}$; tanh activations on the output layer; Glorot uniform weight initialisation; a batch size of roughly a fourth of the data size; and 1000 epochs. Similarly, the latent dimension for the Gaussian parameters was 50 and 100 for the Femur and FAUST datasets, hence named *latent*50 and *latent*100, respectively.

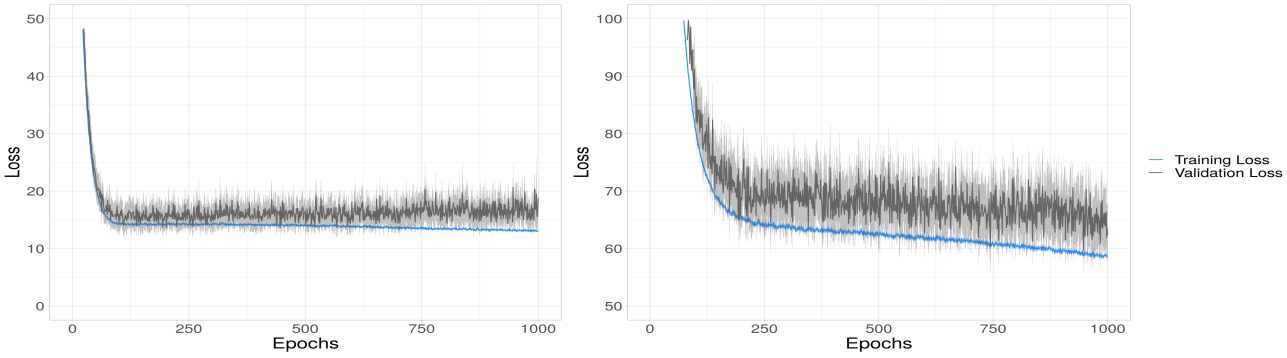

Figure 7.11: The figure illustrates the average training loss (blue) and average validation loss (black) across all cross-validation cycles, including a 95% confidence interval. The baseline models for the Femur *latent*50 (left) and FAUST *latent*100 (right) are displayed.

During model training the losses were reported for each epoch. Figure 7.11 plots the average training and validation losses across all cross-validation cycles, including a 95% confidence interval. Note that the training loss includes the KL divergence and

MSE, while the validation error is only MSE. The Femur dataset showed a clear stabilisation of the validation loss from as few as 100 epochs, indicating a local optimum had been found. Therefore, the model's performance will not improve by increasing the number of training epochs. In contrast, the FAUST validation loss continued to decrease after 1000 epochs, which indicated the models performance could improve by increasing the number of training epochs. The taken for the Femur dataset to run this experiment was on average 3.5 hours and the 6.3 hours for the FAUST dataset. Hence, due to limited computational capacity, 1000 epochs for the FAUST dataset was used as a baseline.

The distribution of the average distance and Hausdorff generalisations are reported in Figure 7.12 for both datasets. The distributions between datasets cannot be directly compared, although this provides a baseline generalisation distribution for further VAE experiments.

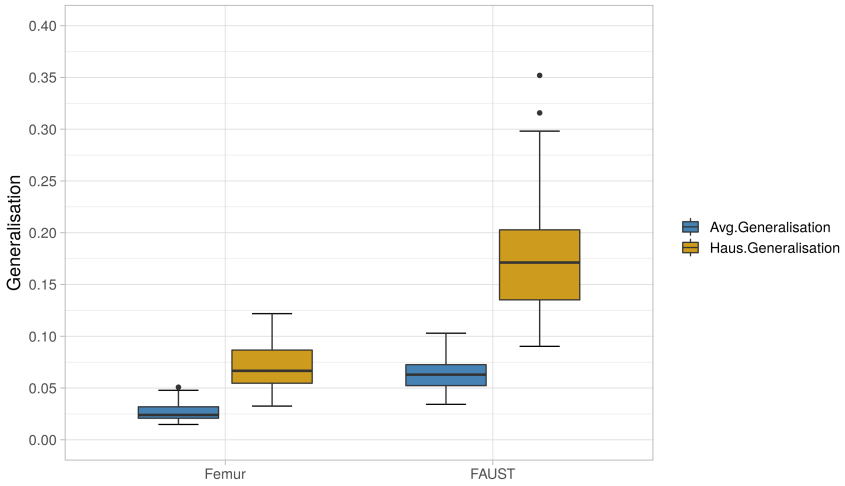

Figure 7.12: The generalisation distribution boxplots are visualised for the baseline VAEs. The plots display the generalisation using average (blue) and HD (gold).

Table 7.4: The baseline VAE average model metrics are displayed for the Femur and FAUST datasets, respectively. The metrics are averaged over all cross-validation cycles.

Model	Avg.Gen	Haus.Gen	Specificity	Time (sec)
		Femur		
latent50	0.0270	0.0715	0.0483	250
		FAUST		
latent100	0.0632	0.1756	0.2083	226

A comprehensive set of model metrics for the baseline models is reported in Table 7.4. The training time may be interpreted as the average time taken, in seconds, to train a single VAE model during the LOOCV training scheme. Thus, the total training time for any experiment can be calculated by the time in seconds multiplied by the number of data items - FAUST 100 and Femur 50. The aforementioned VAEs performance in: generalisation, specificity and training time, was used to benchmark

the next two experiments. The next experiment considered multilayered VAEs with different node configurations.

7.2.2 Experiment 2: Deep VAE

Tan et al., 2018 justifies the use of multi-layer VAEs for 3D mesh data. Their work suggests that an additional layer improves performance for complex structures, with large non-linear variations, such as human body datasets. Therefore, a single hidden layer with different node configurations was considered.

The parameters used were inherited from the prior experiment. Additionally, this experiment considered a single hidden layer with 256 and 512 nodes for both datasets (Tan et al., 2018). The latent dimensions examined were 50 and 128 for the Femur dataset and 100 and 128 for the FAUST dataset. Increasing the number of nodes and layers caused a comparable increase in training time and epochs required. Hence, a more comprehensive parameter search was determined to be infeasible.

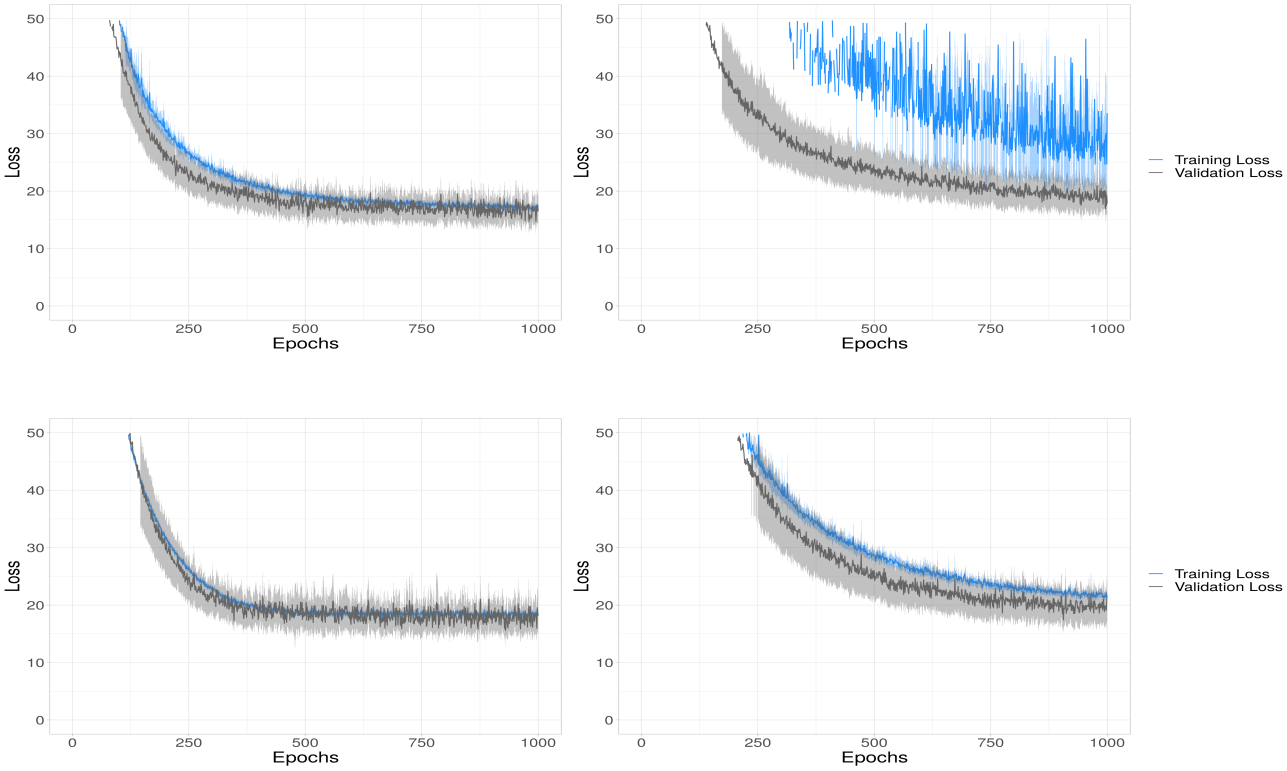

Figure 7.13: The Femur loss graphs are illustrated over 1000 epochs. The models shown are as follows: *latent*50*hidden*256 (top left), *latent*50*hidden*512 (top right), *latent*128*hidden*256 (bottom left) and *latent*128*hidden*512 (bottom right). The mean training loss (blue) and mean validation loss (black), across all cross-validation cycles, are shown including a 95% confidence interval.

Figures 7.13 and 7.14 describe the mean validation and training losses, over all cross-validation cycles. As previously mentioned, the training loss included the KL divergence and MSE, while the validation error was only MSE. A caveat of this, resulted in the validation loss being lower than the training loss. The pertinent feature was

the stabilisation of the validation loss, described by a horizontal trajectory. This suggested that a local optimum was found.

Figure 7.13 illustrates each experiments loss during training, for the Femur dataset. The experiments with a hidden layer dimension of 256 appeared to stabilize, as the validation loss followed a horizontal trajectory. The figure indicates that as few as 500 epochs could be used to achieve this local minimum. In contrast, the hidden layer dimension of 512, showed a slow stabilisation over 1000 epochs. The training loss for model *latent50hidden512* displayed erratic tendencies.

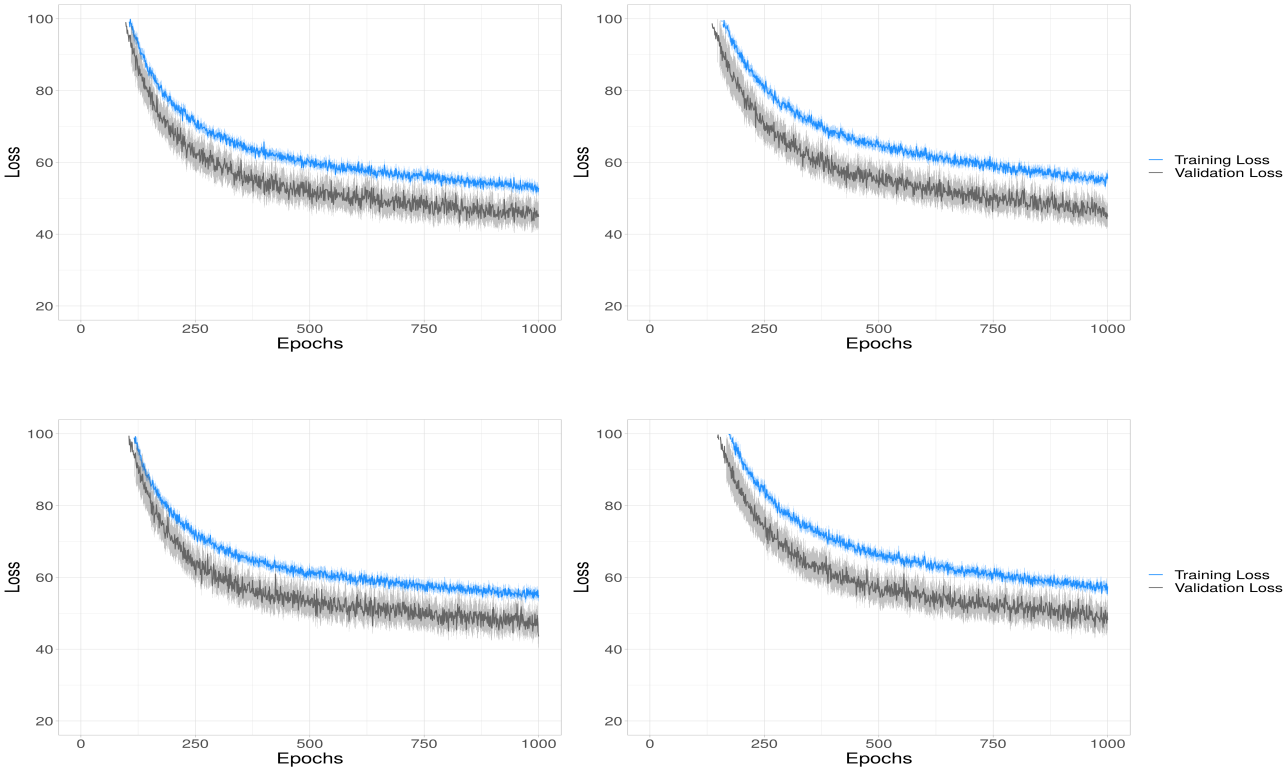

Figure 7.14: The FAUST loss graphs are illustrated over 1000 epochs. The models shown are as follows: *latent*100*hidden*256 (top left), *latent*100*hidden*512 (top right), *latent*128*hidden*256 (bottom left) and *latent*128*hidden*512 (bottom right). The mean training loss (blue) and mean validation loss (black), across all cross-validation cycles, are shown including a 95% confidence interval.

Similarly, Figure 7.14 describes the loss per epoch for the FAUST, deep VAE experiments. All experiments show a slow, continuing descent without validation loss stabilisation. This suggests the models would benefit from increasing the number of epochs, beyond what is feasible using local computation.

Figure 7.15 and 7.16 show the distributions of the average and Hausdorff generalisation for the linear Femur dataset. The single hidden layer VAE experiments were compared against the baseline VAE. The Femur models with the fewest parameters had the best generalisation performance. The baseline model *latent50* had the lowest median across both generalisation metrics. However, model *latent50hidden256* had a compact distribution and low median average generalisation, for the Femur dataset.

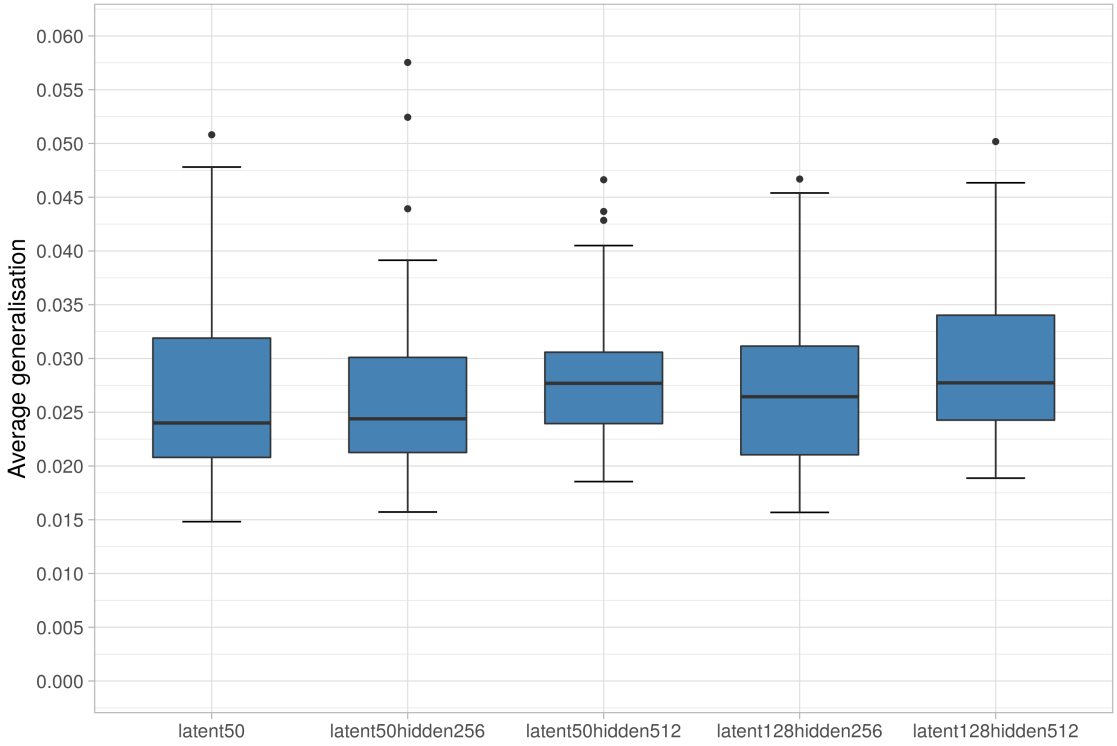

Figure 7.15: The average generalisation, for each experiment, is compared against the Femur baseline *latent50*.

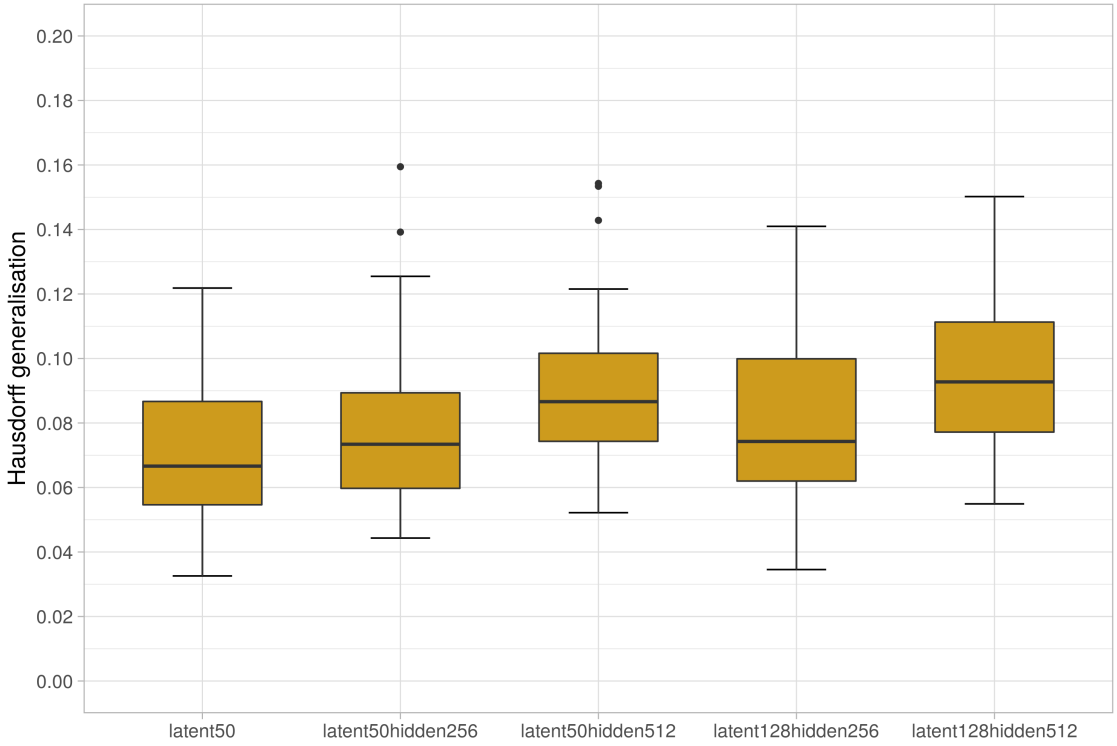

Figure 7.16: The Hausdorff generalisation, for each experiment, is compared against the Femur baseline *latent50*.

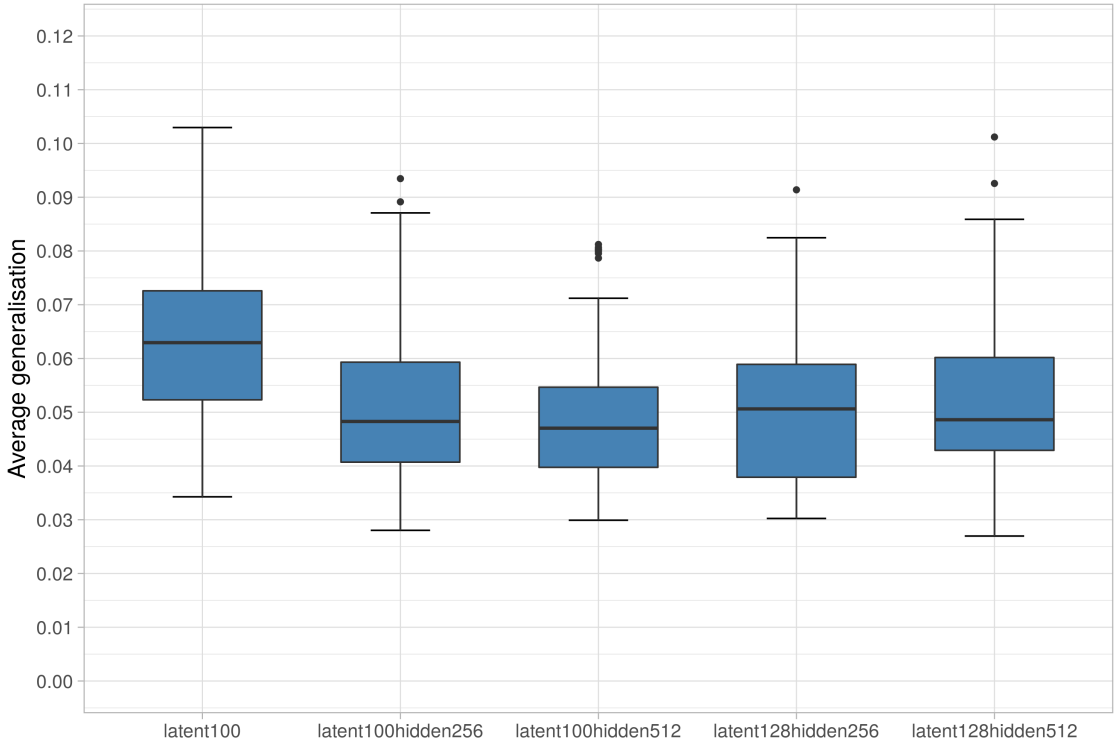

Figure 7.17: The average generalisation, for each experiment, is compared against the FAUST baseline *latent100*.

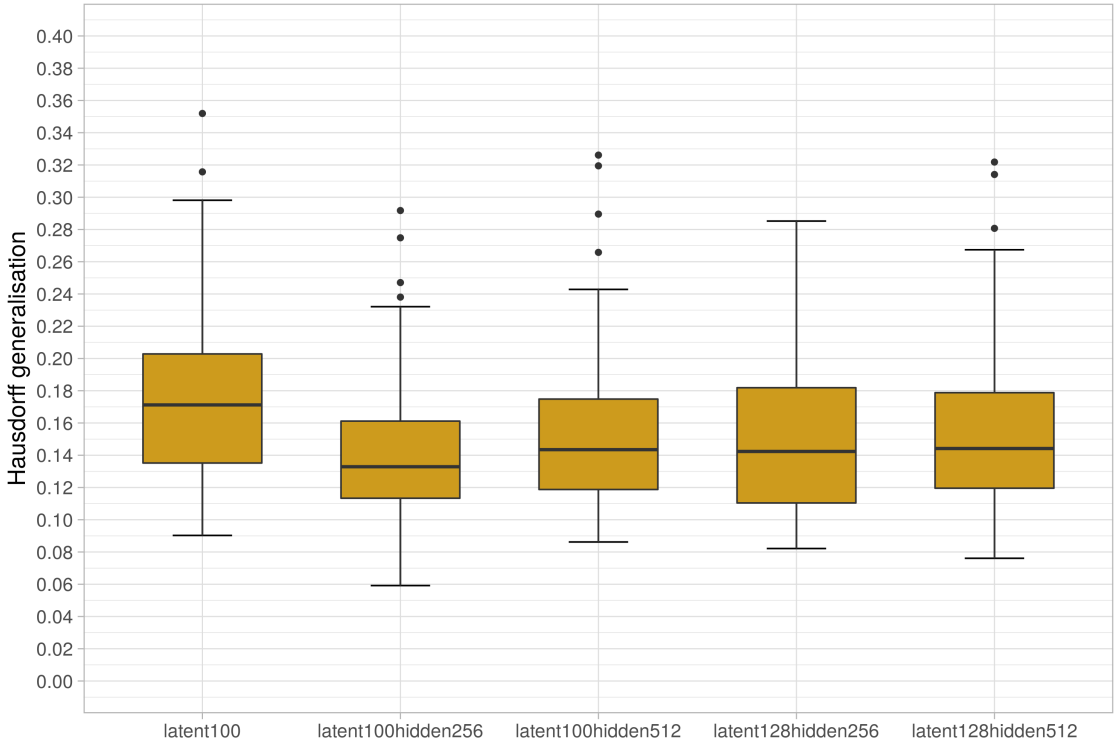

Figure 7.18: The Hausdorff generalisation, for each experiment, is compared against the FAUST baseline *latent100*.

Figure 7.17 and 7.18 display the distribution boxplots of the average and Hausdorff generalisation, for the non-linear FAUST dataset. The model with the lowest generalisation distribution and least variation display the best performance to generalise to new shapes. In contrast to the Femur dataset, the FAUST model experiments outperformed the baseline models in both average and Hausdorff generalisation, based on a visual inspection of the boxplots. This was expected as the large non-linear, local and global, deformations present in the FAUST dataset are captured by increasing the model complexity. The single hidden layer models with latent dimension 100 had the lowest generalisation.

A comprehensive set of average results is displayed by Table 7.5. The average for all metrics are presented across all validation cycles; hence, these results do not consider outliers. The model with the lowest average metrics and lowest, least varying metric distributions shall be considered the best. Subsequently, the best model's structure will be used for the final experiment.

The baseline Femur VAE was the optimal model for this experiment. This model had the lowest specificity, implying the best ability to generate similar observations. Moreover, the *latent50* Femur VAE had the fewest parameters, which required the least time to train - averaging only 250 seconds per model. The generalisation metrics coincide with the distributions, concluding that the baseline VAE had the lowest generalisation; thus, was best for the linear Femur data.

The best performing FAUST model was less clear. Considering specificity, the models with a single hidden layer of dimension 256 had the lowest score. However, the generalisation distributions showed the models with latent dimension 100 were the best performers. Therefore, *latent100hidden256* was selected as the best performing model, for the non-linear FAUST dataset, within this experiment.

Table 7.5: The deep VAEs average model metrics are compared against the baseline model, for the Femur and FAUST datasets, respectively. The metrics are averaged over all cross-validation cycles.

Model	Avg.Gen	Haus.Gen	Specificity	Time (sec)
		Femur		
latent50	0.0270	**0.0715**	**0.0483**	**250**
latent50hidden256	**0.0268**	0.0774	0.0498	895
latent50hidden512	0.0288	0.0906	0.0506	1767
latent128hidden256	0.0277	0.0790	0.0487	884
latent128hidden512	0.0296	0.0950	0.0499	1758
		FAUST		
latent100	0.0632	0.1756	0.2083	**226**
latent100hidden256	0.0508	**0.1428**	0.2065	413
latent100hidden512	**0.0495**	0.1531	0.2074	804
latent128hidden256	0.0500	0.1478	**0.2064**	405
latent128hidden512	0.0519	0.1545	0.2076	905

The baseline model, with *latent50*, and the deep VAE model *latent100hidden256*, were the best models for the Femur and FAUST datasets, respectively. This result proposes that deeper models were more adept in capturing the non-linear shape deformations;

however, may be improved with more training. The model structures in this experiment were used for the succeeding experiment. The following experiment considers alternative activation functions, weight initialisations and number of training epochs.

7.2.3 Experiment 3: Activations, Initialisations and Epochs

The final experiment explored alternative hyperparameters for the VAE models. The loss graphs, in the previous experiment, highlighted that the Femur models required fewer epochs to achieve a stable validation loss. In contrast, the FAUST models showed a continuing descent in validation loss over 1000 epochs. This suggested that a local optimum was not found; hence, more epochs would improve the models performance. The initial weights can influence the local optimum found by the network. Typically, deep learning models are initialised using variations of Gaussian (normal) or uniform distributions (Géron, 2019). Géron, 2019 mentions that both the Glorot uniform and He normal initialisations (He et al., 2015) provide fast convergence. Moreover, these initialisations pair well with leaky ReLU and exponential linear units (ELU) activation functions (Géron, 2019). Géron, 2019 puts forward that the combination of He normal initialisation and ELU activations reduces the vanishing gradient problem; thus, improves performance. A caveat is that ELU is slower to compute than the leaky ReLU activation function (Géron, 2019).

The following experiment used the optimal network structure from the succeeding experiment. The Femur model was trained using a reduced 250 epochs and considering both Glorot uniform and He normal initialisations. Since the baseline *latent50* structure was used, alternative activations on the hidden layers were invalid. In contrast, ELU and leaky ReLU activations were considered for the FAUST model, with latent dimension 100 and hidden layer 256. The models were trained for 2000 epochs and similarly, both Glorot uniform and He initialisations have been explored.

Figure 7.19 justifies the use of fewer epochs for the Femur models, as both experiments have found local optimums. In comparison, Figure 7.20 demonstrates that all FAUST experiments have yet to reach local optimums and would benefit from further increasing the training epochs. Moreover, the He normal experiments showed a slower descent than the Glorot uniform. The ELU Glorot uniform combination displayed the most stable validation loss trajectory.

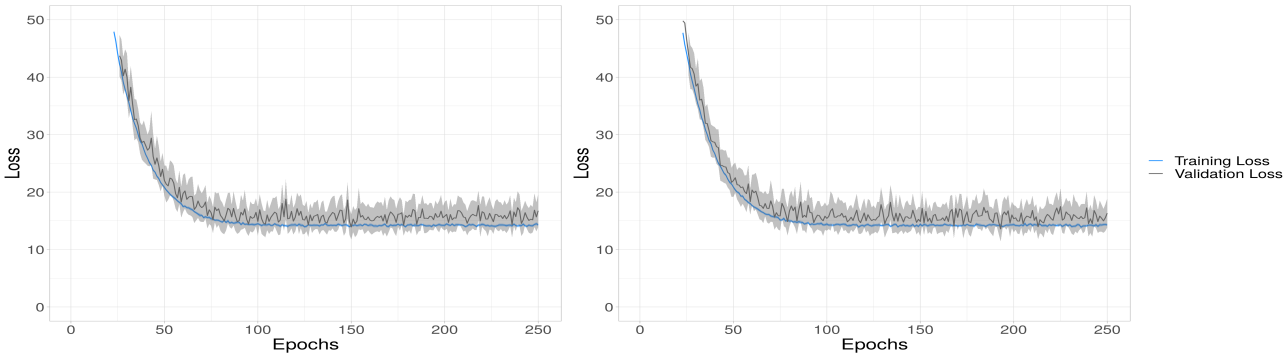

Figure 7.19: The Femur loss graphs are illustrated over 250 epochs. The models shown are as follows: $glorotUniform$ (left) and $heNormal$ (right). The mean training loss (blue) and mean validation loss (black), across all cross-validation cycles, are shown.

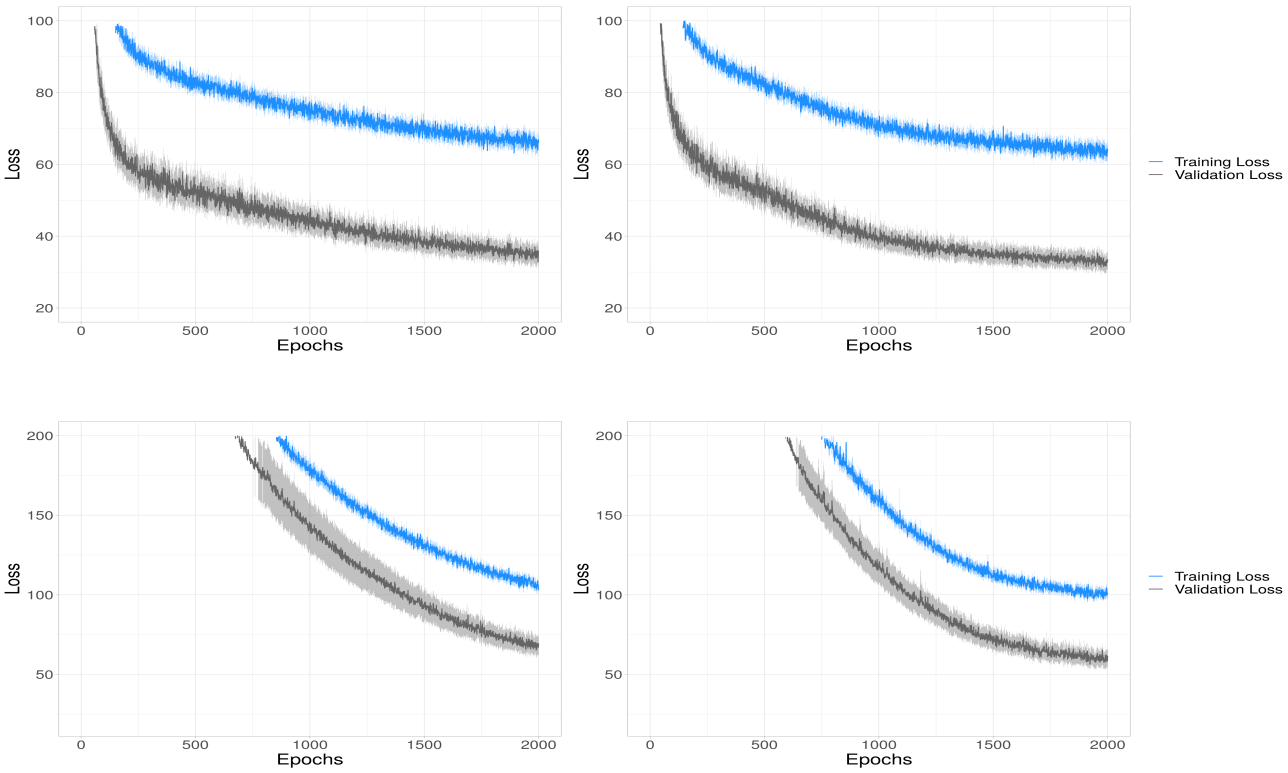

Figure 7.20: The FAUST loss graphs are illustrated over 250 epochs. The models shown are as follows: $leakyReluGlorotUniform$ (top left), $eluGlorotUniform$ (top right), $leakyReluHeNormal$ (bottom left) and $eluGlorotUniform$ (bottom right). The mean training loss (blue) and mean validation loss (black), across all cross-validation cycles, are shown including a 95% confidence interval.

Figure 7.21 and 7.22 show the distribution of the Femur models generalisation with 250 epochs and alternative initialisations. These models have been compared to the baseline model with 1000 epochs and a Glorot uniform initialisation. The He normal experiment displayed a marginally lower and more compact average and Hausdorff generalisation than the Glorot uniform initialisation.

Figure 7.23 and 7.24 compare the FAUST baseline VAE, optimal model from the prior experiment, to the current experimental models. The He normal initialisation experiments displayed large, poor generalisation distributions. This was anticipated due to their validation loss not reaching a local minimum. In contrast, the Glorot uniform models with 2000 epochs outperformed both prior models, with fewer epochs. Increasing the training epochs has provided the models with improved weights to learn the non-linear variations in the data; hence improving the generalisation performance. The combination of the ELU activation function and Glorot uniform initialisation had the lowest and least varying distributions across both generalisation metrics for the non-linear FAUST data.

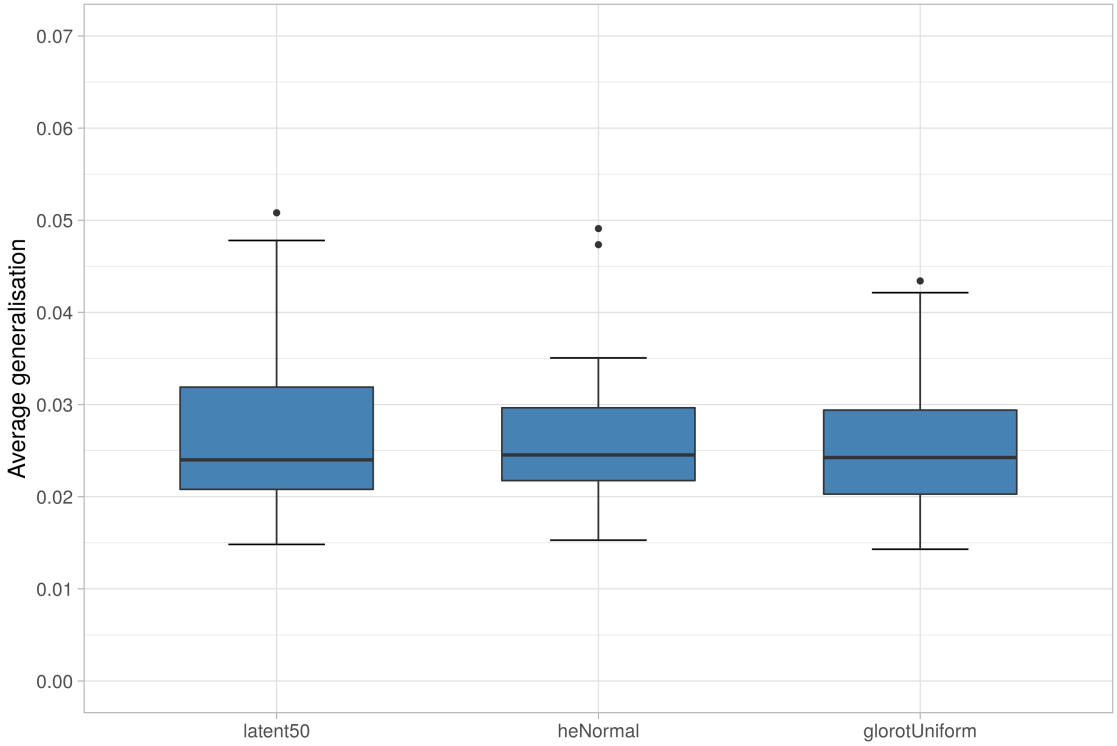

Figure 7.21: The average generalisation, for each experiment, is compared against the Femur baseline *latent50*.

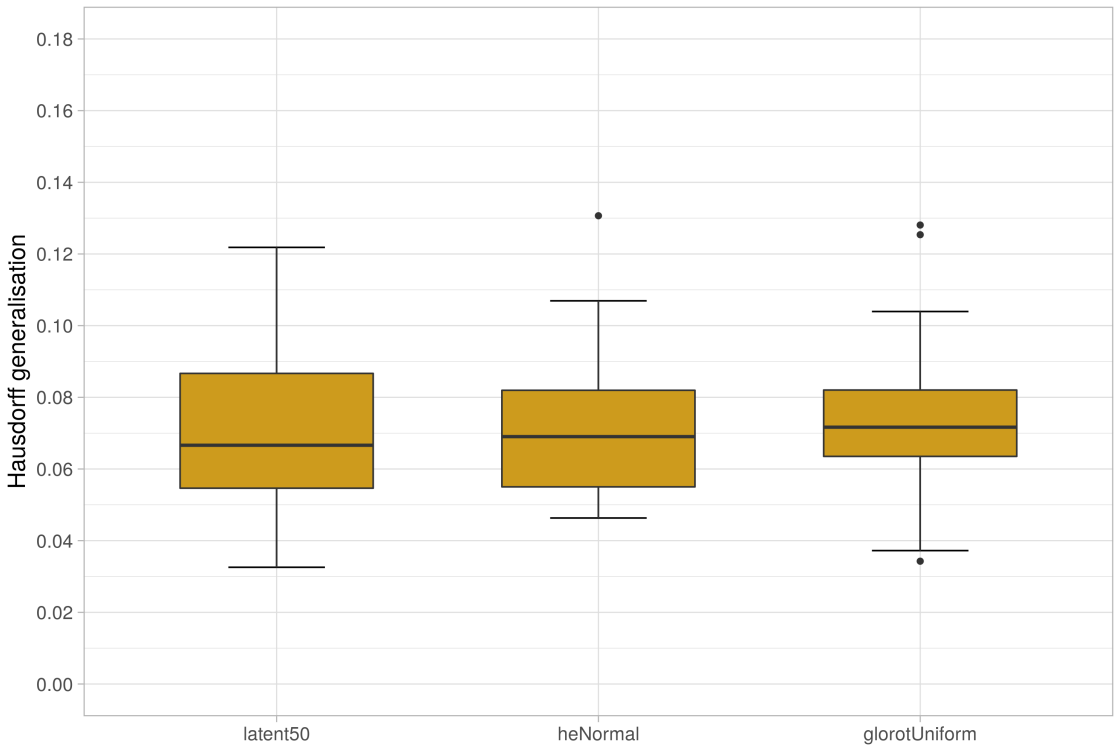

Figure 7.22: The Hausdorff generalisation, for each experiment, is compared against the Femur baseline *latent50*.

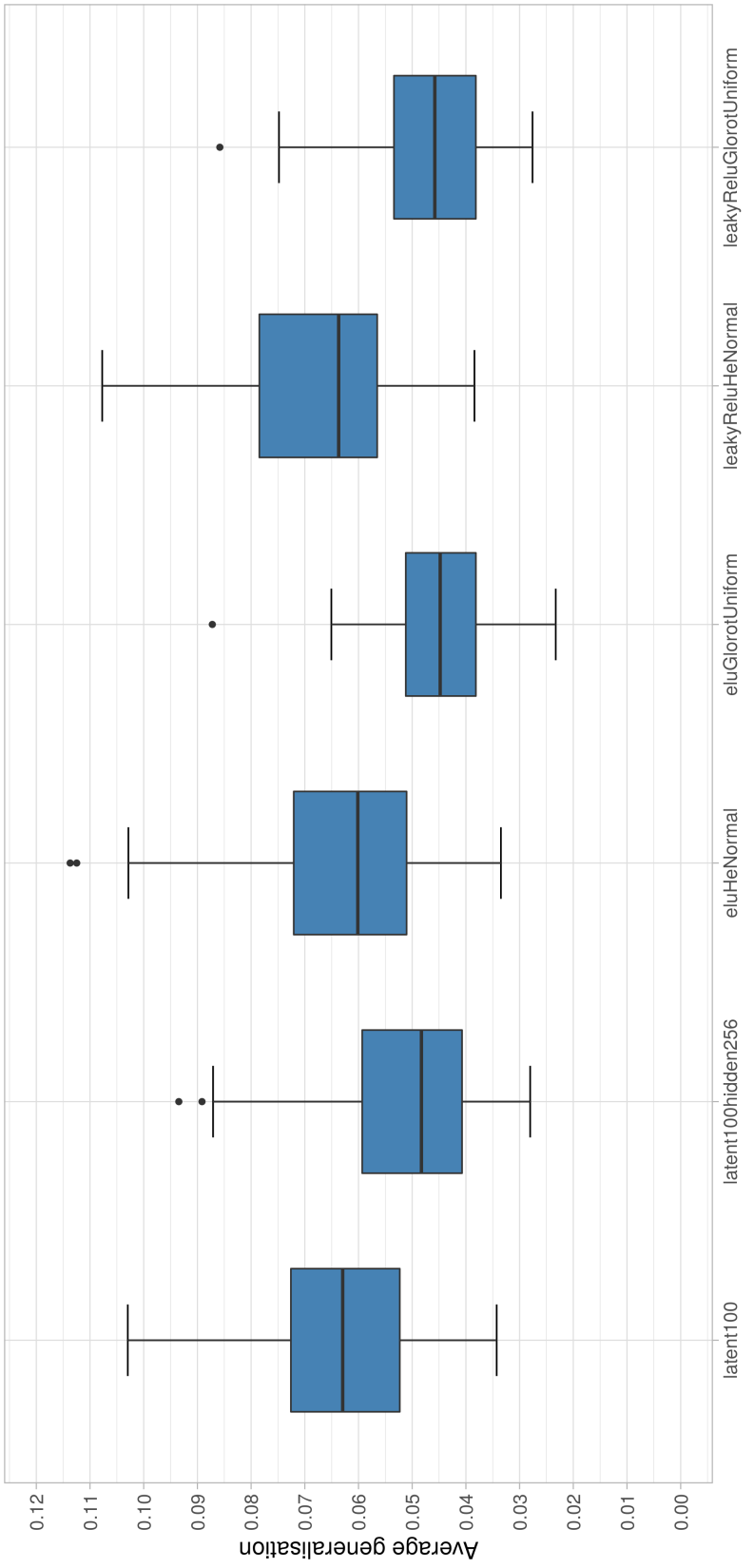

Figure 7.23: The average generalisation, for each experiment, is compared against the FAUST baseline *latent100* and the optimal model in experiment 2 *latent100hidden256*.

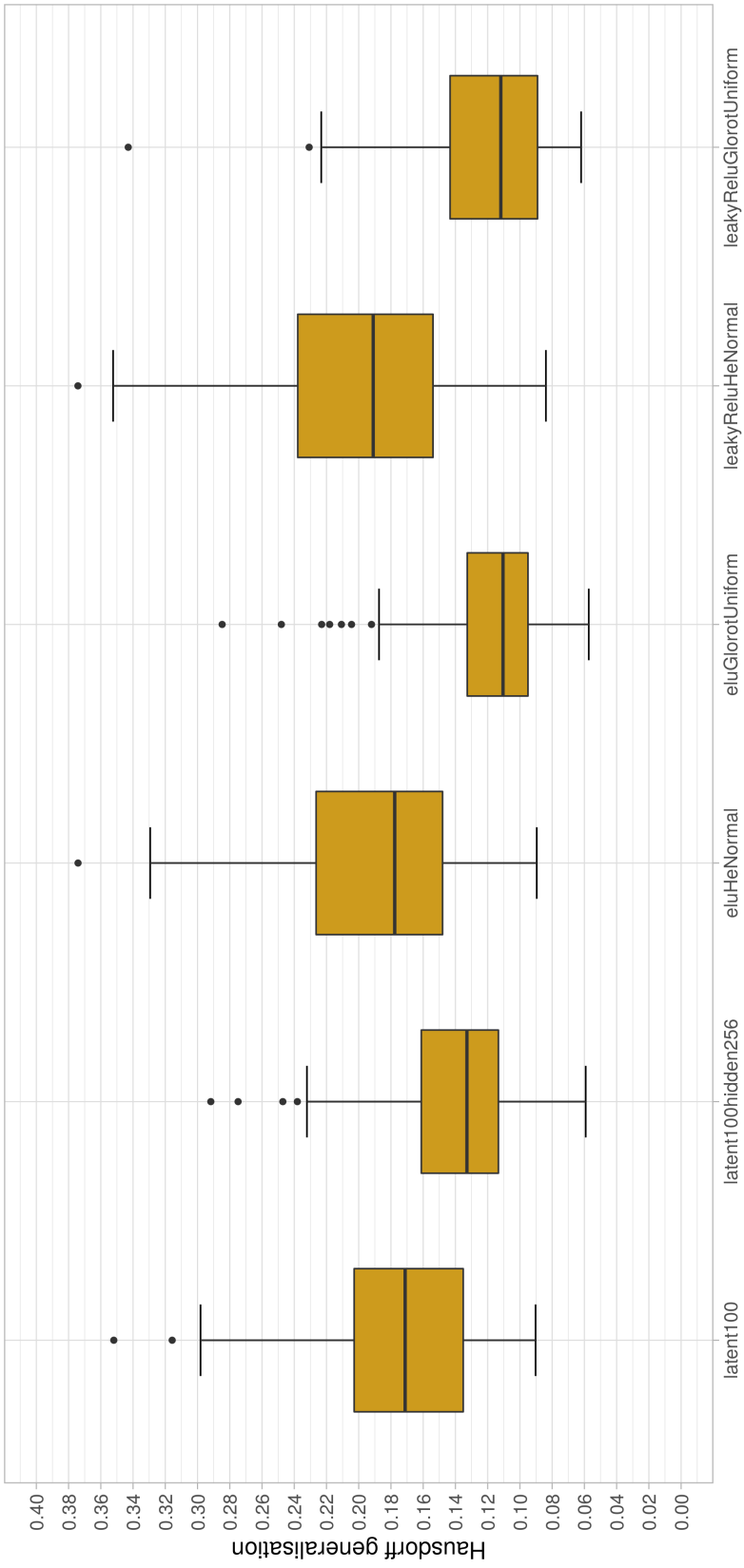

Figure 7.24: The Hausdorff generalisation, for each experiment, is compared against the FAUST baseline *latent100* and the optimal model in experiment 2 *latent100hidden256*.

Table 7.6 displays the average results for final VAE experiment. For the Femur dataset, as few as 250 training epochs improved the performance on the baseline model. Further training epochs overfit the linear Femur data and provided worse generalisation. Both He normal and Glorot uniform initialisations variants returned similar results; however, the Glorot uniform model presented more variation in generalisation. Therefore, the *HeNormal* model, trained with 250 epochs was the best Femur VAE.

Increasing the number of epochs greatly increased the computation time for the FAUST models. This extra computation presented an improved generalisation performance at the cost of specificity. Although out of scope of this work, the model fitting time for VAE were negligible once the weights have been trained. Generalisation was prioritised, as VAEs are known to have inherently strong generative properties (Nash and Williams, 2017; Bagautdinov et al., 2018). Therefore, the best FAUST model was *eluGlorotUniform* trained for 200 epochs, as it achieved the best overall generalisation.

Table 7.6: The average model metrics are compared against the baseline VAE and optimal model in the second experiment, for the Femur and FAUST datasets, respectively. The metrics are averaged over all cross-validation cycles.

Model	Avg.Gen	Haus.Gen	Specificity	Time (sec)
		FEMUR		
latent50	0.0270	0.0715	0.0483	250
heNormal	0.0259	**0.0704**	0.0474	**58**
glorotUniform	**0.0255**	0.0726	**0.0472**	62
		FAUST		
latent100	0.0632	0.1756	0.2083	**226**
latent100hidden256	0.0508	0.1428	**0.2065**	413
eluHeNormal	0.0627	0.1939	0.2131	1787
eluGlorotUniform	**0.0452**	**0.1200**	0.2231	1930
leakyReluHeNormal	0.0670	0.2032	0.2186	1774
leakyReluGlorotUniform	0.0471	0.1202	0.2168	1718

7.2.4 Conclusion

The optimal VAEs for both Femur and FAUST datasets have been concluded through three experiments: a baseline VAE, with the same parameters used in Section 6 for AEs; deep VAEs using parameter combinations informed by the literature (Tan et al., 2018); and finally, adjusting the hyperparameters and training epochs for improved performance.

Given these experiments, a simpler VAE model was sufficient to achieve a local minimum with as few as 250 epochs, for the linear, Femur dataset. The deformations in the linear Femur dataset are less complex and the VAE was able to find a local optimum promptly. In contrast, the non-linear FAUST data saw improvements through deeper VAEs, trained for more epochs. This is a result of the large local and global, non-linear shape variations present in the data. The final model's performance were

only an indication of the performance achieved on local computational power. Improved results would be seen through training the model until the validation loss has stabilised and a more comprehensive hyperparameter training scheme.

The optimal Femur VAE model was structurally similar to the AE used in Section 6. *heNormal* required at most 250 epochs to train and used a He normal initialisation. The optimal FAUST VAE had an additional hidden layer with ELU activation functions. This model required more than 2000 training epochs without achieving a stable validation loss, using Glorot uniform initial weights. The next section conducts a final comparison between the best performing VAE and GPMM, for the respective datasets.

7.3 Model comparison

The literature identified a gap for a direct comparison between GPMMs and VAEs, for non-linear data. The use of linear GPMMs was justified for their: simplicity in use and interpretation; minimal data requirements; and computational efficiency in training. A caveat of these models is that they rely on a linear projection to approximate the non-linear shape space. It was hypothesized that non-linear data would impair the linear model's performance (Pizer and Marron, 2017). In comparison, VAEs, in the non-linear paradigm, are known to be relatively computationally efficient, require minimal data and have strong generative abilities for non-linear data (Nash and Williams, 2017; Tan et al., 2018). These models were anticipated to have better performance for non-linear data.

The prior sections concluded the optimal linear and non-linear models for the Femur and FAUST datasets. The best performing GPMM experiment, for both datasets, was the baseline GPMM, that estimates the parameters directly from the data. In contrast, the best performing VAE, for the Femur and FAUST data, shared the following hyperparameters: the Adam optimiser; a learning rate of $1e^{-4}$; tanh activations on the output layer; and a batch size of roughly a fourth of the data size. Additionally, the optimal Femur VAE model was trained using: a single layer with latent dimension 50; at most 250 epochs to train; and He normal weight initialisations. The best performing VAE for the FAUST data had: an additional hidden layer with ELU activations; 2000, or more, training epochs; and used a Glorot uniform weight initialisation. These models are directly compared in Table 7.7, for generalisation ability, specificity and training time.

Table 7.7: The optimal average GPMM and VAE average model metrics are displayed for the Femur and FAUST datasets, respectively.

Model	Avg.Gen	Haus.Gen	Specificity	Time (sec)
		Femur		
GPMM-PDM (1st PC)	0.0248	0.0807	0.0543	**0.1091**
GPMM-PDM (full)	**0.0045**	**0.0166**	0.0612	**0.1091**
VAE-heNormal	0.0259	0.0704	**0.0474**	58
		FAUST		
GPMM-PDM (1st PC)	0.1215	0.3540	0.2326	**0.7876**
GPMM-PDM (full)	**0.0052**	**0.0213**	0.2732	**0.7876**
VAE-eluGlorotUniform	0.0452	0.1200	**0.2231**	1930

The linear Femur dataset was hypothesized to produce similar performance results for both shape models, as a result of the universal approximation theorem. However, the results in Table 7.7 reject this hypothesis. The full, GPMMs outperformed the optimal VAE models in both average and Hausdorff generalisation by approximately a factor of five. Comparing the simplified model, with a single basis, to the VAE we saw more similar performance. However, the optimal VAE was redeemed by its performance in specificity. The optimal VAEs specificity was superior by roughly 30%, which implies that the VAE had a better generative ability to sample plausible shapes for this data. There was a significantly large difference in training time for these models. Using the software and hardware specifications provided in Section 5.2, the VAE required almost a full minute to train a model, whereas the GPMM training process could be completed within a second. It is important to mention that the GPMM fitting time was longer than the VAE, but left for future research. In conclusion, for linear femur data, the GPMM model had better performance for 3D shape model fitting; however, the generative ability of a VAE was superior. Further exploration of VAE hyperparameters theoretically could yield an equivalence between the models performance, although this is beyond the scope of this study.

The FAUST, non-linear dataset was expected to impair the performance of the linear model. Unexpectedly, the full linear model outperformed the optimal VAE in average generalisation, by a factor of nine, and a factor of six for Hausdorff generalisation. This difference emphasizes the amount of training and complexity of tuning for VAE models. In theory, the optimal VAE should show an equivalence to the GPMM or improve the linear model, given the correct parameters and sufficient training. Nevertheless, within these experiments, the linear model's ability to represent all instances of the object class, was significantly better than the VAE. As seen before, the full GPMM's specificity was inferior by roughly 22%. The training time for the VAE was evidently more computationally expensive, requiring approximately 32 minutes to train a single model. Moreover, the epoch loss graphs proved that the performance should improve from further training. From these experiments, the linear model has achieved: significantly better performance for 3D shape model fitting, at a fraction of the computational cost. In comparison, the non-linear VAE provides a marginal improvement in performance for shape generation, shown by the improved specificity.

In conclusion, the comparison for linear and non-linear models had unexpected results. A linear, full GPMM, with parameters estimated from the data, significantly outperformed the non-linear VAE in 3D model fitting tasks. This result was illustrated in lower generalisation scores across both linear and non-linear datasets. This analysis is not sufficient to infer that the non-linear shape space has been sufficiently approximated by the linear projection. Given more computational power, data and a comprehensive training scheme, the results are expected to show an equivalence or even an improvement for non-linear models. Variational autoencoders are redeemed by the generative properties. Even without finding a local optimum in training their specificity scores were superior. This suggests that with more training even better performance may be achieved.

8 | Conclusion

The literature, in Chapter 1, defined the notion of non-linearity in shape space, which gave rise to two opposing shape modelling paradigms, namely, linear and non-linear models. The use of linear models are justified for their: simplicity; minimal data requirements; and computational efficiency in training. A caveat of this approach is that they rely on an assumption that the linear projection - from the shape space to the tangent space - adequately approximates the non-linearity in the shape space. Non-linear shape modelling methods oppose this assumption. These non-linear models are justified by: the universal approximation theorem, allowing them to capture all types of shape variation (Goodfellow, Bengio, and Courville, 2016); and their strong generative properties. In general, non-linear models are considered to be computationally expensive to train. Two modern techniques, the linear Gaussian process morphable model (GPMM) and non-linear variational autoencoder (VAE), were identified as the best candidates in their respective paradigm. The literature revealed that a direct performance comparison, for these two shape models, would bridge the gap between these two paradigms.

To test the assumption, that linear models sufficiently approximate the non-linearity in 3D shape spaces, two datasets were needed for comparison. The first objective, in Chapter 6, provided a framework for identifying non-linearity, in 3D shape data. Firstly, non-linearity was suspected through visual cues such as large scale bending or rotating deformations. Thereafter, the coordinate points were tested for both univariate and multivariate normality. The proportion of coordinate points that fail all tests for normality were presented. An equivalence between point distribution models (PDMs) and linear AEs was found for the shape data. This allowed a seamless comparison for non-linear models. The modes of variation, for these models, were examined for unrealistic deformations within Gaussian bounded shape parameters. The linear AE was then contrasted against a non-linear AE, which appeared to capture information that was different from the linear models. Hence, this suggests that non-linear models could be equivalent or present model performance improvements. This identification process confirmed that the Femur bone and FAUST human body datasets have different levels of non-linearity, ideal for a comparative analysis. Moreover, the non-linear AE models could potentially capture more information than linear models, in the presence of non-linear shape variation.

The previous objective concluded that non-linear models could theoretically capture more variation, than linear models, in the presence of non-linear shape variation. Thus, the second objective provided a comparison of two current techniques GPMM and VAE for the opposing paradigms. The optimal GPMM and VAE models were selected through a series of sequential experiments with increasing complexity.

For the GPMM experiments, the covariance was initially estimated directly from the data, which is equivalent to a simplistic, linear PDM - similar to the first objective. This provided a benchmark for further experimentation. The following GPMM experiments explored the use of Gaussian kernel functions and thereafter, augmented the data driven covariance with a functional Gaussian kernel. This design, inspired by Lüthi et al., 2018, attempted to add flexibility and non-linearity to the model, ultimately to improve the model's generalisation and specificity performance.

The non-linear VAE experimental design provided a simplistic hyperparameter search to attain the optimal: layer and node combination; weight initialisation; activation functions on the hidden layers; and number of training epochs. The initial experiment was a benchmark VAE, which used the identical hyperparameters seen in the first objective. Tan et al., 2018 motivates the second experiment, as their model uses a multi-layer VAE to capture the non-linearity in 3D human body datasets. The final experiment inherited the best model structure from the previous experiment and further explored the weight initialisation, activation functions and training epochs.

The optimal model experiments were directly compared, for both datasets, using: average generalisation, Hausdorff generalisation, specificity and model training time in seconds. Fitting time, however, was not recorded and left for further study. The generalisation metrics indicate a model's ability fit to unseen observations. A flexible model will have low generalisation, but when sampled will generate implausible observations, leading to poor specificity performance (Pizer and Marron, 2017). Specificity measures a model's generative ability. These results were conditional on the computational and data limitations of the experiments. The experiments were conducted on local hardware and software defined in Chapter 5; hence, the training and hyperparameter tuning schemes, for both linear and non-linear experiments, were restricted.

For rigid, 3D data, such as bones, the variations in the dataset are considered to be linear (Lüthi et al., 2018; Fouefack et al., 2020). For linear data, Chapter 6 demonstrated that the opposing paradigms of models should be equivalent. Under the aforementioned experimental conditions, the results showed that the linear GPMM significantly outperformed the non-linear VAE in generalisation, for the Femur dataset. This was unexpected as the non-linear model is not limited by an orthogonal linear projection from the shape space; hence, could capture more information, if a local optimum is found. The VAE managed to reach a local optimum within a minute of training. By comparison, the GPMM achieved an improved generalisation within under a second of training. Considering the specificity of each model, the VAE presented better, lower scores, confirmed by literature (Tan et al., 2018). The results on the linear data emphasizes the difference in computation required to train and tune the VAE for optimal performance. An equivalence of these models is theoretically attainable, for linear data, given the correct hyperparameters. However, this was infeasible due to the computational power and left for further research.

The FAUST dataset has been proven to contain non-linear shape variations between observations. Therefore, it was expected that the VAE models would demonstrate an equivalence or improved performance, in the presence of the non-linear shape variation. The experiment results in Chapter 7 were even further polarised. The full GPMM outperformed the optimal VAE in average generalisation, by a factor of nine, and a factor of six for Hausdorff generalisation. This result was conditional on the experimental design limitations. Tan et al., 2018 boasted improved results due to the use of their 3D mesh shape descriptor RIMD (Rotation Invariant Mesh Difference) (Gao et al., 2016b), compared to the 3D coordinates in correspondence

shape descriptor. This euclideanization technique would be valuable to explore for further research.

In theory, the deep VAE model should capture more information in the presence of non-linear shape variation. The experiments illustrated that, after approximately thirty minutes of training on average (2000 epochs), for a single model, the model did not reach a local optimum. This experimentation could be optimised through utilising different software, than what was presented in Chapter 5, and parallelisation. Local hardware restrictions could be alleviated using cloud based platforms or GPU clusters; therefore, allowing a more comprehensive training and tuning modelling scheme.

The non-linear model, although lacking in flexibility, was redeemed by its generative ability. Across both datasets, the VAE presented improved performances in the specificity metric, as anticipated by the literature (Tan et al., 2018). This result reinforces VAEs application for computer vision and generative tasks, which, once trained, allow for efficient and effective sampling. Conversely, the GPMMs: simplicity in application, fast training speed; and low generalisation error for minimal data, makes this model ideal for medical imaging tasks, especially for rigid, singular anatomical structures, such as bones.

In conclusion, this research provided a framework, using AE models, to identify non-linearity in 3D shape data. Thereafter, a comparative study for linear GPMMs and non-linear VAE shape models was conducted. Under the experimental conditions, the linear models outperformed the non-linear models in model fitting tasks, such as generalisation, across both datasets. The optimal VAE model, despite its premature training, had superior generative properties for both Femur and FAUST datasets. These results were unable to deduce if the GPMM had sufficiently captured the non-linearity present in the data. This was a result of insufficient computational power to sufficiently train the non-linear models. Nevertheless, this research provides a comparative case study for these techniques, using local computational power. This study provides ground-work and will guide further research around the non-linearity problem for 3D shape data.

For future work considering the non-linearity problem, for 3D shape data, this research can be used as a foundation. The study has highlighted the importance of computational resources. Through the use of cloud based computing or clusters, deep learning architectures - such as VAEs - could be trained more extensively with comprehensive hyperparameter tuning. The alteration would allow for an initial equivalence to be found between the two opposing paradigms; thereafter, tuning for improved performance for non-linear shape data. This approach would be able to conclude if the linear model had sufficiently captured the non-linearity present in the data. Software improvements could be made to reduce the training time and optimise the performance of the non-linear models. This research did not consider the model fitting time. Although, it was noted that GPMMs required more time during model fitting, this should be recorded in the computational time for future work. A comprehensive kernel analysis could be conducted for GPMM. Alternative kernels may provide generalisation and specificity improvements in modelling non-linear shape variation. Due to limited amounts of data, a hold out test set was not used and the validation loss was considered a proxy for testing error. An unseen testing set would provide more robust results. Future research should consider different shape descriptors. Shape descriptors such as RIMD, remove non-linearity and does not require data in correspondence; therefore, improved results would be expected.

Bibliography

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: http://tensorflow.org/.

Abdi, Hervé and Lynne J Williams (2010). "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4, pp. 433–459.

Abu-Mostafa, Yaser S, Malik Magdon-Ismail, and Hsuan-Tien Lin (2012). *Learning from data*. Vol. 4. AMLBook New York, NY, USA:

Allen, Brett, Brian Curless, and Zoran Popović (2003). "The space of human body shapes: reconstruction and parameterization from range scans". In: *ACM transactions on graphics (TOG)* 22.3, pp. 587–594.

Amit, Yali, Ulf Grenander, and Mauro Piccioni (1991). "Structural image restoration through deformable templates". In: *Journal of the American Statistical Association* 86.414, pp. 376–387.

Bagautdinov, Timur et al. (2018). "Modeling facial geometry using compositional VAEs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3877–3886.

Bengio, Yoshua et al. (2009). "Learning deep architectures for AI". In: *Foundations and trends® in Machine Learning* 2.1, pp. 1–127.

Berlinet, Alain and Christine Thomas-Agnan (2011). *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media.

Blanz, Volker and Thomas Vetter (1999). "A morphable model for the synthesis of 3D faces". In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 187–194.

Bogo, Federica et al. (2014). "FAUST: Dataset and evaluation for 3D mesh registration". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3794–3801.

Bookstein, Fred L (1997). *Morphometric tools for landmark data: geometry and biology*. Cambridge University Press.

Bourlard, Hervé and Yves Kamp (1988). "Auto-association by multilayer perceptrons and singular value decomposition". In: *Biological cybernetics* 59.4-5, pp. 291–294.

Bowden, Richard (2000). "Learning non-linear Models of Shape and Motion". PhD thesis.

Bowden, Richard, Tom A Mitchell, and Mansoor Sarhadi (2000). "Non-linear statistical models for the 3D reconstruction of human pose and motion from monocular image sequences". In: *Image and Vision Computing* 18.9, pp. 729–737.

Box, George EP and George C Tiao (2011). *Bayesian inference in statistical analysis*. Vol. 40. John Wiley & Sons.

Bronstein, Michael M and Iasonas Kokkinos (2010). "Scale-invariant heat kernel signatures for non-rigid shape recognition". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1704–1711.

Brunton, Alan et al. (2014). "Review of statistical shape spaces for 3D data with comparative analysis for human faces". In: *Computer Vision and Image Understanding* 128, pp. 1–17.

Chollet, Francois et al. (2015). *Keras*. URL: https://github.com/fchollet/keras.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2015). "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289*.

Cootes, T. F. et al. (1992). "Training Models of Shape from Sets of Examples". In: *BMVC92*. Ed. by David Hogg and Roger Boyle. London: Springer London, pp. 9–18. ISBN: 978-1-4471-3201-1.

Cootes, Tim, ER Baldock, and J Graham (2000). "An introduction to active shape models". In: *Image processing and analysis* 243657, pp. 223–248.

Cootes, Timothy F et al. (1995). "Active shape models-their training and application". In: *Computer vision and image understanding* 61.1, pp. 38–59.

D'Agostino, Ralph B (1972). "Small sample probability points for the D test of normality". In: *Biometrika* 59.1, pp. 219–221.

Davatzikos, Christos, Xiaodong Tao, and Dinggang Shen (2003). "Applications of wavelets in morphometric analysis of medical images". In: *Wavelets: Applications in Signal and Image Processing X*. Vol. 5207. International Society for Optics and Photonics, pp. 435–444.

Davies, Rhodri H et al. (2003). "Building optimal 2D statistical shape models". In: *Image and Vision Computing* 21.13-14, pp. 1171–1182.

Doersch, Carl (2016). *Tutorial on Variational Autoencoders*. cite arxiv:1606.05908. URL: http://arxiv.org/abs/1606.05908.

Donahue, Jeff, Philipp Krähenbühl, and Trevor Darrell (2016). "Adversarial feature learning". In: *arXiv preprint arXiv:1605.09782*.

Dryden, Ian L and Kanti V Mardia (2016). *Statistical shape analysis: with applications in R*. Vol. 995. John Wiley & Sons.

Dumoulin, Vincent et al. (2016). "Adversarially learned inference". In: *arXiv preprint arXiv:1606.00704*.

Eslami, SM Ali et al. (2014). "The shape boltzmann machine: a strong model of object shape". In: *International Journal of Computer Vision* 107.2, pp. 155–176.

Fletcher, P Thomas et al. (2004). "Principal geodesic analysis for the study of nonlinear statistics of shape". In: *IEEE transactions on medical imaging* 23.8, pp. 995–1005.

Fouefack, Jean-Rassaire et al. (2020). "Dynamic multi-object Gaussian process models: A framework for data-driven functional modelling of human joints". In: *arXiv preprint arXiv:2001.07904*.

Fox, Charles W and Stephen J Roberts (2012). "A tutorial on variational Bayesian inference". In: *Artificial intelligence review* 38.2, pp. 85–95.

Gao, Lin et al. (2016a). "Efficient and flexible deformation representation for data-driven surface modeling". In: *ACM Transactions on Graphics (TOG)* 35.5, pp. 1–17.

Gao, Zhanheng et al. (2016b). "Mesh generation and flexible shape comparisons for bio-molecules". In: *Computational and Mathematical Biophysics* 1.1.

Géron, Aurélien (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international*

conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, pp. 249–256.

Goodall, Colin (1991). "Procrustes methods in the statistical analysis of shape". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 53.2, pp. 285–321.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Goodfellow, Ian et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2672–2680.

Gower, John C (1975). "Generalized procrustes analysis". In: *Psychometrika* 40.1, pp. 33–51.

Grenander, Ulf and Michael I Miller (1998). "Computational anatomy: An emerging discipline". In: *Quarterly of applied mathematics* 56.4, pp. 617–694.

Gulli, Antonio and Sujit Pal (2017). *Deep learning with Keras*. Packt Publishing Ltd.

Hassoun, Mohamad H and Agus Sudjianto (1997). "Compression net-free autoencoders". In: *Workshop on Advances in Autoencoder/Autoassociator-Based Computations at the NIPS*. Vol. 97, pp. 605–611.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

He, Kaiming et al. (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

Heap, Tony and David Hogg (1995). *Automated pivot location for the cartesian-polar hybrid point distribution model*. University of Leeds, School of Computer Studies.

Heimann, Tobias and Hans-Peter Meinzer (2009). "Statistical shape models for 3D medical image segmentation: a review". In: *Medical image analysis* 13.4, pp. 543–563.

Henze, N and B Zirkler (1990). "A class of invariant consistent tests for multivariate normality". In: *Communications in statistics-Theory and Methods* 19.10, pp. 3595–3617.

Hinton, Geoffrey et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal processing magazine* 29.6, pp. 82–97.

Hornik, Kurt (1991). "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2, pp. 251–257.

Hotelling, Harold (1933). "Analysis of a complex of statistical variables into principal components." In: *Journal of educational psychology* 24.6, p. 417.

Huang, Haibin, Evangelos Kalogerakis, and Benjamin Marlin (2015). "Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces". In: *Computer Graphics Forum*. Vol. 34. Wiley Online Library, pp. 25–38.

Huckemann, Stephan, Thomas Hotz, and Axel Munk (2010). "Intrinsic shape analysis: Geodesic PCA for Riemannian manifolds modulo isometric Lie group actions". In: *Statistica Sinica*, pp. 1–58.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

Izenman, Alan Julian (2008). *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. 1st ed. Springer Publishing Company, Incorporated. ISBN: 0387781889.

Jarque, Carlos M and Anil K Bera (1987). "A test for normality of observations and regression residuals". In: *International Statistical Review/Revue Internationale de Statistique*, pp. 163–172.

Johnson, Richard Arnold, Dean W Wichern, et al. (2002). *Applied multivariate statistical analysis*. Vol. 5. 8. Prentice hall Upper Saddle River, NJ.

Kass, Michael, Andrew Witkin, and Demetri Terzopoulos (1988). "Snakes: Active contour models". In: *International journal of computer vision* 1.4, pp. 321–331.

Kazhdan, Michael, Thomas Funkhouser, and Szymon Rusinkiewicz (2003). "Rotation invariant spherical harmonic representation of 3 d shape descriptors". In: *Symposium on geometry processing*. Vol. 6, pp. 156–164.

Kendall, David G (1977). "The diffusion of shape". In: *Advances in applied probability* 9.3, pp. 428–430.

– (1989). "A survey of the statistical theory of shape". In: *Statistical Science*, pp. 87–99.

Kendall, David George et al. (2009). *Shape and shape theory*. Vol. 500. John Wiley & Sons.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Kingma, Durk P et al. (2016). "Improved variational inference with inverse autoregressive flow". In: *Advances in neural information processing systems*, pp. 4743–4751.

Klingenberg, Christian Peter (2016). "Size, shape, and form: concepts of allometry in geometric morphometrics". In: *Development genes and evolution* 226.3, pp. 113–137.

– (2020). "Walking on Kendall's Shape Space: Understanding Shape Spaces and Their Coordinate Systems". In: *Evolutionary Biology*, pp. 1–19.

Kramer, Mark A (1991). "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE journal* 37.2, pp. 233–243.

Lang, Muriel, Martin Kleinsteuber, and Sandra Hirche (2018). "Gaussian process for 6-DoF rigid motions". In: *Autonomous Robots* 42.6, pp. 1151–1167.

Larsen, Anders Boesen Lindbo et al. (2015). "Autoencoding beyond pixels using a learned similarity metric". In: *arXiv preprint arXiv:1512.09300*.

Lata, Y Vijaya et al. (2009). "Facial recognition using eigenfaces by PCA". In: *International Journal of Recent Trends in Engineering* 1.1, p. 587.

Le, Huiling and David G Kendall (1993). "The Riemannian structure of Euclidean shape spaces: a novel environment for statistics". In: *The annals of Statistics*, pp. 1225–1271.

Le, Yen H, Uday Kurkure, and Ioannis A Kakadiaris (2013). "PDM-ENLOR: Learning ensemble of local PDM-based regressions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1878–1885.

Le Roux, Nicolas et al. (2011). "Learning a generative model of images by factoring appearance and shape". In: *Neural Computation* 23.3, pp. 593–650.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *nature* 521.7553, pp. 436–444.

Li, Jun et al. (2017). "Grass: Generative recursive autoencoders for shape structures". In: *ACM Transactions on Graphics (TOG)* 36.4, pp. 1–14.

Li, Mu, James Tin-Yau Kwok, and Baoliang Lü (2010). "Making large-scale Nyström approximation possible". In: *ICML 2010-Proceedings, 27th International Conference on Machine Learning*, p. 631.

Lilliefors, Hubert W (1967). "On the Kolmogorov-Smirnov test for normality with mean and variance unknown". In: *Journal of the American statistical Association* 62.318, pp. 399–402.

Lüthi, M and G Bouabene (2020). "Statistical Shape Modelling: Computing the Human Anatomy". In: University of Basel, FutureLearn. URL: https://www.futurelearn.com/courses/statistical-shape-modelling.

Lüthi, M. et al. (2018). "Gaussian Process Morphable Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.8, pp. 1860–1873. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2017.2739743.

Masci, Jonathan et al. (2015). "Geodesic convolutional neural networks on riemannian manifolds". In: *Proceedings of the IEEE international conference on computer vision workshops*, pp. 37–45.

Metaxas, Dimitris N (2012). *Physics-based deformable models: applications to computer vision, graphics and medical imaging*. Vol. 389. Springer Science & Business Media.

Nash, Charlie and Christopher KI Williams (2017). "The shape variational autoencoder: A deep generative model of part-segmented 3D objects". In: *Computer Graphics Forum*. Vol. 36. Wiley Online Library, pp. 1–12.

Odaibo, Stephen (2019). "Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function". In: *arXiv preprint arXiv:1907.08956*.

Odersky, Martin et al. (2004). *The Scala language specification*.

Omrani, Emad et al. (2016). "Tribological study in microscale using 3D SEM surface reconstruction". In: *Tribology International* 103, pp. 309–315.

Pearson, Karl (1901). "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pizer, Stephen M and JS Marron (2017). "Object statistics on curved manifolds". In: *Statistical Shape and Deformation Analysis*. Elsevier, pp. 137–164.

Plaut, Elad (2018). "From principal subspaces to principal components with linear autoencoders". In: *arXiv preprint arXiv:1804.10253*.

Pohl, Kilian M et al. (2006). "Logarithm odds maps for shape representation". In: *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, pp. 955–963.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Remondino, Fabio (2003). "From point cloud to surface: the modeling and visualization problem". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 34.

Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational inference with normalizing flows". In: *arXiv preprint arXiv:1505.05770*.

Robbins, Herbert and Sutton Monro (1951). "A stochastic approximation method". In: *The annals of mathematical statistics*, pp. 400–407.

Romdhani, Sami, Shaogang Gong, and Ahaogang Psarrou (Jan. 1999). "A Multi-View Nonlinear Active Shape Model Using Kernel PCA". In: DOI: 10.5244/C.13.48.

Ross, Amy (2004). "Procrustes analysis". In: *Course report, Department of Computer Science and Engineering, University of South Carolina* 26.

Rostami, Reihaneh et al. (2019). "A Survey on Data-Driven 3D Shape Descriptors". In: *Computer Graphics Forum*. Vol. 38. Wiley Online Library, pp. 356–393.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Sanger, Terence D (1989). "Optimal unsupervised learning in a single-layer linear feedforward neural network". In: *Neural networks* 2.6, pp. 459–473.

Schönborn, Sandro et al. (2017). "Markov chain monte carlo for automated face image analysis". In: *International Journal of Computer Vision* 123.2, pp. 160–183.

Shapiro, Samuel Sanford and Martin B Wilk (1965). "An analysis of variance test for normality (complete samples)". In: *Biometrika* 52.3/4, pp. 591–611.

Shlens, Jonathon (2014). "A tutorial on principal component analysis". In: *arXiv preprint arXiv:1404.1100*.

Small, Christopher G (2012). *The statistical theory of shape*. Springer Science & Business Media.

Sminchisescu, Cristian and Allan Jepson (2004). "Generative modeling for continuous non-linearly embedded visual inference". In: *Proceedings of the twenty-first international conference on Machine learning*, p. 96.

Sozou, Peter D et al. (1994). "A Non-linear Generalisation of PDMs using Polynomial Regression." In: *BMVC*. Vol. 94. Citeseer, pp. 397–406.

Sozou, Peter D et al. (1997). "Non-linear point distribution modelling using a multilayer perceptron". In: *Image and Vision Computing* 15.6, pp. 457–463.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Staib, Lawrence H and James S Duncan (1992). "Boundary finding with parametrically deformable models". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence*, pp. 1061–1075.

Styner, Martin A et al. (2003). "Evaluation of 3D correspondence methods for model building". In: *Biennial International Conference on Information Processing in Medical Imaging*. Springer, pp. 63–75.

Tam, Gary KL et al. (2012). "Registration of 3D point clouds and meshes: A survey from rigid to nonrigid". In: *IEEE transactions on visualization and computer graphics* 19.7, pp. 1199–1217.

Tan, Qingyang et al. (2018). "Variational autoencoders for deforming 3d mesh models". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5841–5850.

Tomczak, Jakub M and Max Welling (2017). "VAE with a VampPrior". In: *arXiv preprint arXiv:1705.07120*.

Tran, Luan and Xiaoming Liu (2018). "Nonlinear 3d face morphable model". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7346–7355.

Tsagaan, Baigalmaa et al. (2002). "An automated segmentation method of kidney using statistical information". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 556–563.

Van Kaick, Oliver et al. (2011). "A survey on shape correspondence". In: *Computer Graphics Forum*. Vol. 30. Wiley Online Library, pp. 1681–1707.

Van Rossum, Guido and Fred L. Drake (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace. ISBN: 1441412697.

Venables, William N and Brian D Ripley (2013). *Modern applied statistics with S-PLUS*. Springer Science & Business Media.

Wall, Michael E, Andreas Rechtsteiner, and Luis M Rocha (2003). "Singular value decomposition and principal component analysis". In: *A practical approach to microarray data analysis*. Springer, pp. 91–109.

Wang, Lei et al. (2015). "Beyond covariance: Feature representation with nonlinear kernel matrices". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4570–4578.

Wickham, Hadley (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN: 978-3-319-24277-4. URL: https://ggplot2.tidyverse.org.

Williams, Christopher KI and Carl Edward Rasmussen (2006). *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA.

Williams, Reed M and Horea T Ilieş (2018). "Practical shape analysis and segmentation methods for point cloud models". In: *Computer Aided Geometric Design* 67, pp. 97–120.

Wu, Chenglei et al. (2016a). "Model-based teeth reconstruction." In: *ACM Trans. Graph.* 35.6, pp. 220–1.

Wu, Jiajun et al. (2016b). "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In: *Advances in neural information processing systems*, pp. 82–90.

Wu, Zhirong et al. (2015). "3d shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920.

Wuhrer, Stefanie, Chang Shu, and Pengcheng Xi (2012). "Posture-invariant statistical shape analysis using Laplace operator". In: *Computers & Graphics* 36.5, pp. 410–416.

Xu, Bing et al. (2015). "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853*.

Yuille, Alan L, Peter W Hallinan, and David S Cohen (1992). "Feature extraction from faces using deformable templates". In: *International journal of computer vision* 8.2, pp. 99–111.

Zhang, Chao et al. (2015). "Shell PCA: Statistical shape modelling in shell space". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1671–1679.

Zhang, Cheng et al. (2018). "Advances in variational inference". In: *IEEE transactions on pattern analysis and machine intelligence* 41.8, pp. 2008–2026.

Zheng, Guoyan, Shuo Li, and Gabor Szekely (2017). *Statistical shape and deformation analysis: methods, implementation and applications*. Academic Press.

Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun (2018). *Open3D: A Modern Library for 3D Data Processing*. cite arxiv:1801.09847Comment: http://www.open3d.org. URL: http://arxiv.org/abs/1801.09847.

Zhou, Xiangrong et al. (2014). "Development and evaluation of statistical shape modeling for principal inner organs on torso CT images". In: *Radiological physics and technology* 7.2, pp. 277–283.