Automatic scoring of X-rays in Psoriatic Arthritis

Rambojun, Adwaye

Award date:
2020

Awarding institution:
University of Bath

Automatic scoring of X-rays in Psoriatic Arthritis

submitted by

Adwaye M. Rambojun

for the degree of *Doctor of Philosophy*

of the

University of Bath

Department of Mathematical Sciences

May 2020

COPYRIGHT

Declaration of any previous Submission of the Work

The material presented here for examination for the award of a higher degree by research has not been incorporated into a submission for another degree.

.....

Adwaye Mihirkiran Rambojun

Declaration of Authorship

.....

Adwaye Mihirkiran Rambojun

Summary

In this thesis, we investigate the potential for automation in the scoring process of Psoriatic Arthritis x-rays. We focus on the identification of bones structures through a latent space shape model that is driven by a Gaussian Process.

We describe a top to bottom approach to designing such a model that includes the data collection and annotation process. We highlight the importance of capturing and modelling uncertainties associated with having automated systems in medical imaging. The main tool for this is noise models in a Bayesian setting.

The main mathematical contribution we make takes the form of a shape model for which we perform an exact Bayesian marginalisation of the model parameters. These parameters include the shape and the pose. We define a dependence structure that models the uncertainties present in a segmentation task. We show that the Active Appearance Model of Cootes et al. [2001] falls under our framework.

We believe that this is significant as previous work has only focused on the real world performance of the models as opposed to the probabilistic interpretation. Such an interpretation is important as it allows us to better understand the model uncertainties.

Acknowledgements

First and foremost, I want to thank my supervisors Tony, Neill and Will for their guidance and continued support of my PhD study. I very much enjoyed all the discussions (sometimes quite lively) we had over the last three years.

I would also like to thank all of my friends at the Department of Mathematical Sciences at the University of Bath and the SAMBa team. I was lucky and am grateful to have had the support of my sister, my mother and my father during my PhD. Finally, writing this thesis would be impossible without the constant moral support provided by Shreshta Sookwareea. Thank you for keeping me sane during this period.

This work is dedicated to my father, Hurrydutt Rambojun, who passed away on July 2, 2018 after a brave and exemplary battle against cancer. You showed us how to live and this thesis is as much your work as it is mine.

This research was supported by a scholarship from the EPSRC Centre for Doctoral Training in Statistical Applied Mathematics at Bath (SAMBa), under the project EP/L015684/1.

To Those Who Believe In The Elegance Of The Universe.

Contents

1 Introduction **1**

1.1 Problem Statement and Motivation 1

1.2 Medical Background 3

1.2.1 The Human Hand 3

1.2.2 Scoring for Damage 5

1.3 Mathematical Model of a Hand X-ray 7

1.4 Related Work 8

1.5 Evaluation of our work 9

1.6 Outline of the Thesis 10

2 Data Aquisition **12**

2.1 Introduction 12

2.2 Annotation 13

2.3 Curve outline pre-processing 14

2.3.1 Parametric curve representation 16

2.3.2 Imposing shape correspondence 20

2.4 Intensity normalisation 21

2.4.1 Histogram Matching 22

2.4.2 Histogram matching the dataset 23

3 Mathematical Tools **24**

3.1 Introduction 24

3.2 Bayesian Inference 24

3.2.1 Setting 26

3.2.2 Variational Inference 28

3.2.3 Segmentation as a Bayesian Problem 29

3.3 Gaussian Process Latent Variable Models 32

3.3.1 Gaussian Processes: A quick overview 32

3.3.2 Gaussian Process Regression 32

3.3.3 GPLVM 34

3.3.4 Probabilistic PCA and dual PPCA 35

3.3.5 Back Constrained GPLVM . . . 39
3.3.6 Variational GPLVM . . . 40
3.3.7 Computing $\mathcal{I}_1$ . . . 42
3.3.8 Computing $\mathcal{I}_2$ . . . 42
3.3.9 Final Bound . . . 44
3.3.10 Summary . . . 45
3.4 Radial Basis Function Interpolation . . . 46
3.4.1 Setting . . . 46
3.4.2 Interpolation error estimate . . . 46

4 Mathematical representation of shape 50
4.1 Introduction . . . 50
4.2 Shape Definition . . . 51
4.2.1 Shape correspondence . . . 52
4.3 Statistical shape models . . . 53
4.3.1 Linear Shape Model . . . 54
4.3.2 Latent Space Representation through GPLVMs . . . 55
4.3.3 Discussion . . . 56
4.4 Shape modelling through GPLVMs . . . 57
4.4.1 Choosing the latent space dimension . . . 58
4.4.2 Single Bone outline data . . . 60
4.4.3 Multiple Bone outline data . . . 62
4.5 Concluding remarks . . . 63

5 Mathematical representation of texture 64
5.1 Introduction . . . 64
5.2 Texture Definition . . . 65
5.3 Texture Modelling . . . 68
5.4 Discriminative Texture Modelling with Neural Networks . . . 69
5.4.1 Some Notation . . . 69
5.4.2 Patch Based Texture Modelling . . . 70
5.4.3 Purely convolutional semantic segmentation . . . 72
5.5 Experimental results . . . 74
5.5.1 Patch-net . . . 75
5.5.2 U-net . . . 79
5.5.3 Concluding remarks . . . 83

6 Bone identification through shape modelling 85
6.1 Introduction . . . 85

6.2 Active Latent Space Shape Model . . . 86
6.2.1 Latent Space Representation . . . 88
6.2.2 Data term . . . 89
6.2.3 Setting . . . 90
6.2.4 Marginalisation and model fitting . . . 90
6.2.5 Objective function . . . 92
6.3 Discussion . . . 93
6.3.1 Uncertainty propagation . . . 93
6.3.2 Full Bayesian Marginalisation . . . 94
6.3.3 Marginalisation Error Estimate . . . 94
6.4 Experimental Results . . . 96
6.4.1 Data term V_u . . . 98
6.4.2 Coarse to fine approach . . . 100
6.4.3 Model evaluation . . . 102
6.5 AAMs interpreted as ALSSM . . . 104
6.5.1 Latent space representation . . . 104
6.5.2 Data term . . . 105
6.5.3 Marginalisation and objective function . . . 105

7 Final Conclusions and Outlook 106
7.1 Outlook . . . 106
7.1.1 Full hand model . . . 106
7.1.2 Clinical applications . . . 107
7.1.3 Fully Generative Models . . . 107
7.2 Statistical models of deformation fields . . . 108

A Appendix 109
A.1 Product of Gaussians . . . 109
A.2 Stochastic Gradient Descent . . . 112
A.3 Introduction to Neural Networks . . . 116
A.3.1 Setting . . . 116
A.3.2 Activation Functions . . . 117
A.3.3 Discrete kernel operations . . . 118
A.3.4 Layers . . . 121
A.3.5 Training . . . 127
A.4 Experiment Results . . . 129
A.4.1 ARAND score . . . 130
A.4.2 Tabulated results . . . 130

Chapter 1

Introduction

In this thesis we develop an automated bone identification tool. This tool is to be used by rheumatoligists who score hand x-rays of Psoriatic Arthritis Patients (PsA). This chapter introduces this work and the philosophy we developed as part of this work.

We give our motivation in section 1.1. We introduce some medical concepts in section 1.2 where we describe the anatomy of the human hand and the scoring methods used by rheumatologists. We mathematically formulate the hand anatomy in section 1.3. We then give an overview of the current state of things and how our work compares to the literature in section 1.4. Assessing a multi disciplinary work such as this one should not be done in the traditional way. We discuss this in section 1.5. We finish this chapter by describing the set up of this thesis in section 1.6.

1.1 Problem Statement and Motivation

Psoriatic Arthritis (PsA) is an inflammatory arthritis that affects up to 20% of people having the skin condition Psoriasis (Ogdie et al. [2012]). The most common symptoms are swelling and tenderness at the joints in the hands and wrist along with damage to the nails. The inflammation leads to pain and stiffness. Untreated inflammation will lead to damage of the joint structures, impaired physical function and reduced quality of life. Early intervention to suppress inflammation is required to preserve function and prevent damage accumulation. Genetics, obesity and ethnicity are contributing factors to the onset of the disease.

The aims of drug treatment are to improve pain and prevent damage though the suppression of inflammation. The most widely available tool to assess dam-

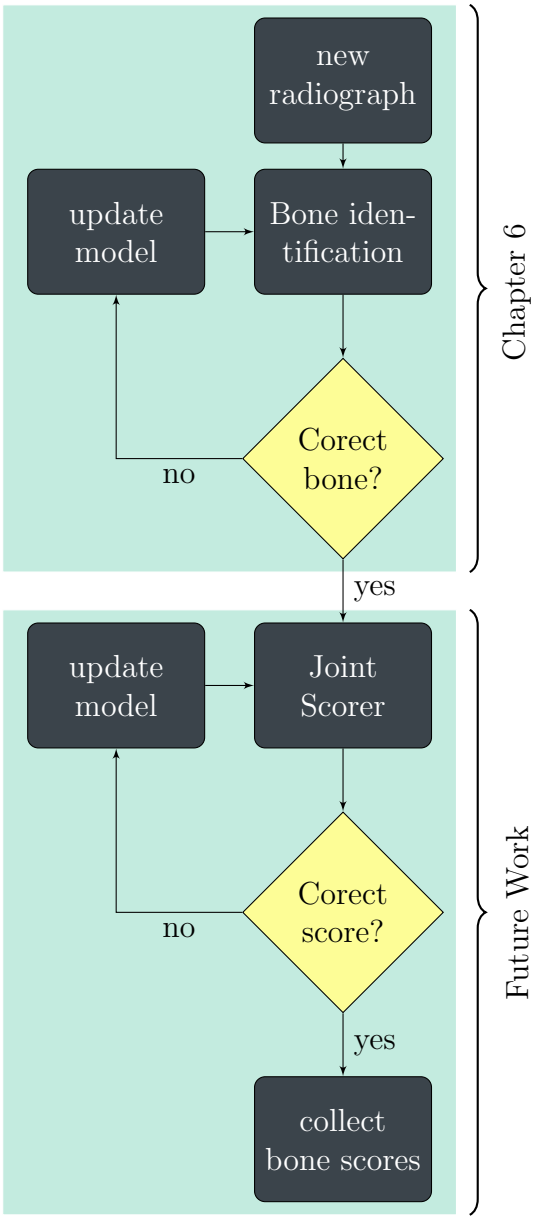


Figure 1-1: The flowchart shows the semi supervised learning process for an automatic scoring system.

age is the plain radiograph (X-Ray). Clinicians use plain radiographs as part of routine clinical care to ensure treatment is suppressing damage progression. Little is known about the natural course of damage progression determined by plain radio-graphs in PsA because progression is slow over years and decades.

Longitudinal cohort studies are conducted to answer clinical questions relating to disease progression, prognosis and response to treatment. However little clinical data has been reported because the quantification of damage on x-rays (scoring methods) are time consuming to conduct. Moreover, drug treatments for PsA tends to be expensive in a rationed system. Consequently, such clinical data needs to be collected to increase the impact of prescriptions made. The

easiest way to achieve this is to score existing x-rays for damage in a faster and more efficient manner.

Our proposed solution is to create an automated system that would assist rheumatologists in scoring x-rays. One important aspect of this system is experience acquisition. As rheumatologists see more cases of PsA, they become better at detecting the disease and at assessing its severity. We therefore favor a system that assimilates information from new, unseen cases via online learning.

Figure 1-1 shows a flowchart of our proposed diagnosis system. It consists of two main processes. The first is the bones identifier, whose aim is to find the location of each bone in the hand x-ray. The second system will use the output of the bone identifier to score the bone for damage. In the initial stages of the deployment, both of these systems will operate under the supervision of a trained rheumatologist. If the first system fails to delineate a bone, the rheumatologist will correct the output by delineating the bone themselves. The new bone outline is stored in a stack of wrong labels and the bone identifier is updated when this stack reaches a certain capacity. The same process is used with the bone scorer.

Having such a semi supervised system makes it easy for online learning to happen. The assimilation of the new example is done via additional training of the machine learning models that drive the bone identifier and the bone scorer. Moreover, it solves the problem of a lack of initial training data for training and testing such models. As we later show, collecting and annotating quality data for our models is a time consuming process. The process of correcting the prediction from new example also serves the purpose of data collection.

1.2 Medical Background

It is of paramount importance to understand the expectations of the medical community in projects like this one. Failure to do so might cause a divergence in the research direction. Hence, for completeness, we introduce the hand anatomy and briefly describe the scoring process that PsA x-rays undergo.

1.2.1 The Human Hand

The human hand has three types of bones, namely the carpals, the metacarpal and the phalanges. There are 8 carpal bones, 5 metacarpal (MP) bones (1 for

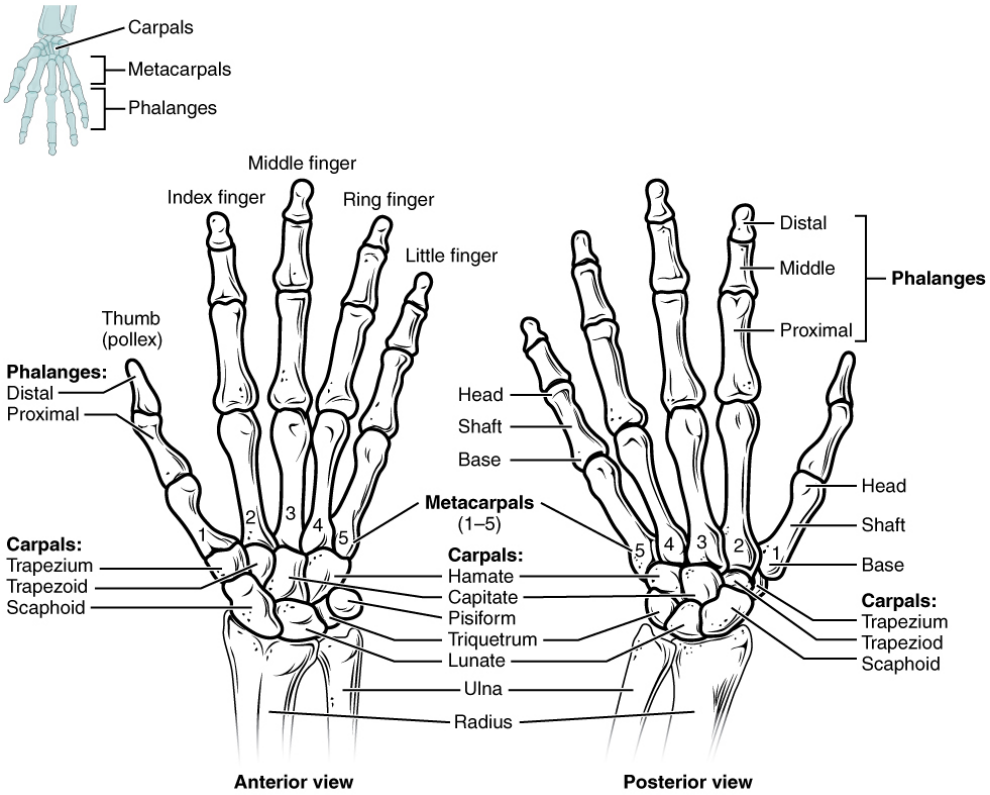


Figure 1-2: The Figure shows the bones of the hand along with their labels. The image is courtesy of the book of Anatomy and Physiology by Openstax (Betts et al. [2014]).

Area	Bone Name	Abbreviation
Phalanges	Distal Phalange	DP
	Middle Phalange	MP
	Proximal Phalange	PP
Metcarpals	Metacarpal	MC
Carpals	Trapezium	Tm
	Trapezoid	Td
	Capitate	C
	Hamate	H
	Scaphoid	S
	Lunate	L
	Triquetrum	Tr
	Pisiform	P
Joints	Distal Interphalangeal Joint	DIP
	Proximal Interphalangeal Joint	PIP
	Metacarpophalangeal Joint	MCP
	Carpometacarpal Joint	CMC

Table 1.1: The Table shows the bones of the hand along with their commonly used abbreviations.

each finger) and 14 phalangial bones. With the exception of the thumb, the phalanges in each fingers are are the proximal phalanges (PP), the middle phalanges (MP) and the distal phalanges (DP). As for the thumb, it only has the proximal

and distal phalanges. Each finger in the hand is labelled with a number with the 1, 2, 3, 4 and 5 representing the thumb, the index finger, the middle finger, the ring finger and the small finger respectively.

Starting from the DP the area between the bones (joints) are the distal interphalangeal joint (DIP), the proximal interphalangeal joint (PIP), the metacarpophalangeal joint (MCP) and the carpometacarpal joint. As the names suggest they are the joints between the DP and the MP; the MP and the PP; the PP and the MP and the MP and the carpals respectively.

The joints are of particular interest to rheumatologists as this is where most of the damage due to PsA manifests itself. We describe this in section 1.2.2. Table 1.1 summarise the abbreviations used for the hand bones. Figure 1-2 shows the locations of the different bones in the hands.

1.2.2 Scoring for Damage

The current standard in the severity assessment of Psioratic Arthritis (Psa) involves the grading of hand X-rays by experts. The scores take into account three types of damage which are illustrated in Figure 1-3:

(1) **Bone Erosion**: The spacing between bones is the same but the bone erodes,

(2) **Joint Space Narrowing (JSN)**: The bone is intact but the distance separating them becomes smaller,

(3) **Osteoproliferation**: The joint space is intact but extra bone starts to grow at the joint.

Rheumatologists specialising in PsA will usually grade each condition in each joint of a hand or feet using four scoring techniques. **Steinbrocker** (Rahman et al. [1998]) is a global scoring technique that will take into account the radiographic features of tissue as well as bone. The **Ratingen** (Wassenberg et al. [2001]) scoring rule measures osteoproliferation and erosion at each joint. The **Modified sharp** (Sharp et al. [1985]) and **Sharp-van der Heijde** or SvH (Van der Heijde et al. [2000]) scoring rules measure erosion and JSN at each joint respectively. Rheumatologists then combine those scores to come up with a global PsA score.

One metric used to measure the performance of a scoring system is the Interobserver intraclass correlation coefficient (ICC). The ICC is used to assess conformity among scorers. Its aim is in effect to make sure that the scores suffers

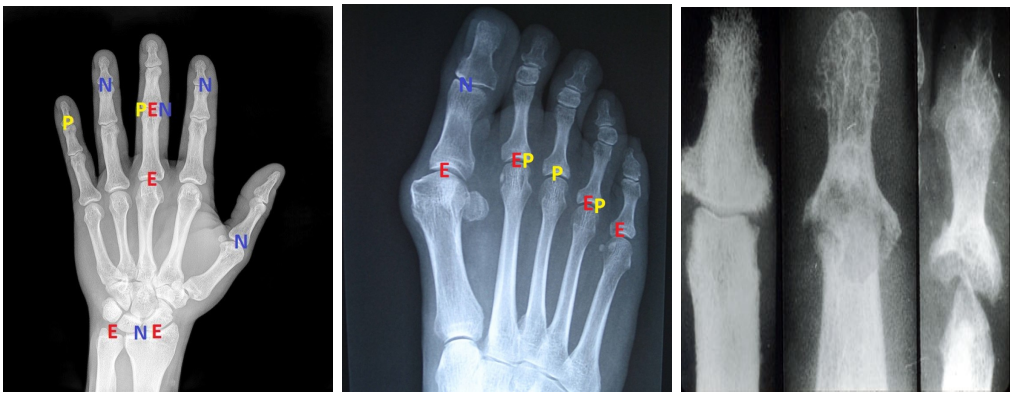


Figure 1-3: X-ray showing the different type of joint degradation with P denoting proliferation; E denoting erosion; N denoting joint space narrowing in a hand. The right most picture shows an enlarged picture of each type of damage in a joint. From left to right one can see JSN, osteoproliferation and bone erosion.

Ratingen Destruction score	
Score value	**Meaning**
0	normal
1	1 or more erosions with $< 10\%$ of the joint area destroyed
2	1 or more erosions with $11 - 25\%$ of the joint area destroyed
3	$26 - 50\%$ of the joint area destroyed
4	$51 - 75\%$ of the joint area destroyed
5	$> 75\%$ of the joint area destroyed or bony ankylosis
Ratingen Proliferation score	
Score value	**Meaning**
0	normal
1	1-2 mm proliferation or bone growth covering 25% of original size
2	2-3 mm proliferation or bone growth covering 25% of original size
3	>3 mm proliferation or bone growth covering 25% of original size
4	bony ankylosis

Table 1.2: The Table shows the Ratingen Score values and their corresponding symptoms. The Ratingen score was developed specifically to assess the x-rays of PsA patients. Bony Ankylosis is the condition where the joint becomes stiff due to extensive damage.

from the least amount of bias. However, Table 1.2 shows that the score depends on a subjective interpretation of the damage extent.

Indeed, the damage is expressed as a percentage of the total area. There is therefore a need to measure the area before (normal) and after the onset of PsA. This is almost impossible as x-rays are usually taken once the patient has developed PsA. Scorers do use other joints in the hand as a proxy for normality, but this also can be misleading. Another issue that comes out of subjectivity is the notion of what is normal. This is made even harder when one considers the natural progression of the joint appearance.

Scoring is also a time consuming exercise. If one were to score an x-ray using the Ratingen method, then each joint in the hand needs to be scored. This would require the careful analysis of damaged areas. This is not made easy by the fact that a joint might suffer from multiple damage types, the presence of which affects one's ability to ascertain the exact type of damage.

1.3 Mathematical Model of a Hand X-ray

As we are dealing with hand x-rays, it is important to have a mathematical formulation of the anatomical description of the hand. Let $u : \Omega \subset \mathbb{R}^2 \to [0, 255]$ be a hand x-ray. Each bone in the hand can be described as an open subset of Ω. There are 19 phalanges in a hand that we treat as subsets $O_i, i = 0, ..., 18$ of Ω.

The boundaries ∂O_i of these areas take the form of closed non-overlapping curves $\boldsymbol{m}_i : [0, 1] \to \mathbb{R}^2$. We have $N_b = 19$ such curves in a hand x-ray. The location of each bone is given by the midpoint $\boldsymbol{d}_i = (d_x^{(i)}, d_y^{(i)}), i = 1, .., 19$ of its bounding curve. Given a parametric representation $\boldsymbol{m}_i : [0, 1] \to \mathbb{R}^2$ of the curve bounding bone i, the midpoint is

$$\arg\min_{\boldsymbol{x}} \int_0^1 \|\boldsymbol{m}_i(s) - \boldsymbol{x}\|^2 \mathrm{d}s = \frac{\int_0^1 \boldsymbol{m}_i(s)\mathrm{d}s}{|\partial O_i|} \tag{1.1}$$

where $|\partial O_i|$ is the circumference of the bone. The orientation of the bone with respect to each other is controlled by the angle ψ_i the $i-$th bone and the horizontal. For our purposes, $(\psi_i, \boldsymbol{d}^{(i)}, \boldsymbol{m}_i)$ fully parametrise the shape of the hand. ψ_i can be found by considering the orientation of bones with respect to each other. We describe in Chapter 4 how to extract the shape and pose (denoted in this section by $\psi_i, \boldsymbol{d}^{(i)}$) of each bone from this description.

Usually, the regions O_i can be identified by a change in the intensity profile. This is controlled by the so-called texture information of the image which we describe in Chapter 5. This formulation of a hand x-ray allows us to use an Active Latent Space Shape Model (ALSSM) to model the shape of bones. As we show in Chapter 6, this involves using a shape model and a texture model to identify the bones of the index finger.

1.4 Related Work

The use of computer vision in medical imaging goes back to the invention of Magnetic Resonance Imaging. It was necessary at the time to be able to reconstruct the organ being imaged. Bertero and Boccacci [1998] is a good reference for the methods used in that field. In addition to reconstructing the image of the organ, there was a need to be able to identify the region of the image that contained the organ. Segmentation and registration algorithms were invented to deal with these problems. One noteworthy registration algorithm is the Lucas-Kanade model introduced in Lucas et al. [1981]. More complex registration algorithms that seek to align landmarks were later introduced. Modersitzki [2003] is a good reference for the algorithms used in image registration. Other algorithms that rely on the clustering of pixel values can be found in Haralick and Shapiro [1985].

Nowadays, machine learning algorithms are being used for transfer learning via a labelled data set. Greenspan et al. [2016] highlights the promise of Deep Learning in the field of medical imaging. For example Bar et al. [2015] use Convolutional Neural Networks (CNN) to locate abnormalities in Chest X-rays. Deep learning requires a lot of data for training and testing and tend to exhibit a black box behaviour.

The examples we have just given all fall into the discriminative category. Generative models of appearance tend to exhibit a better control on the uncertainties being propagated. The Active Appearance Model (AAM) of Cootes et al. [2001] falls into this category, although the model matching procedure does involves a discriminative step.

Kauffman [2009] provides a detailed study of automating scoring in Rheumatoid Arthritis. We believe that it is the piece of work that most closely resembles ours. Indeed, much of the material there would complement this thesis. The main tool used by Kauffman [2009] is the Active Appearance model of Cootes et al. [2001], which is the reference when it comes to statistical modelling of images. Lindner et al. [2013] uses Constrained Local Models for medical segmentation and has patented a bone identification technology for x-rays. A successful application of the AAM in a medical setting is demonstrated by Minciullo et al. [2018].

1.5 Evaluation of our work

The aim of projects like the one we present is to ultimately make machine learning and computer vision play an integral part in medical diagnosis. Getting approval from the concerned bodies is a lengthy process. One needs to have sufficient proof that the method presented works. As we enter a scientific age where the lines between disciplines get blurred, it is important for scientists to engage in a discussion about how best to evaluate research outcomes.

In the machine learning community, the performance of models is usually evaluated using the training set and the test set diagnosis. In the case of a classifier for example, one would treat the test set accuracy[1] to be a good representation of performance. These metrics are also compared to those of other models to ascertain whether it is an improvement over the current norm. Medical research, on the other hand, concerns itself with large trials when it comes to evaluating new methods. Van der Heijde et al. [2005], for example, compare the performance of scoring methods when used in large cohort studies.

Even when it comes to metrics, the notion of what should be used and what should not deviates between the two communities. Rambojun et al. [2019] use the Adjusted Rand Score or ARAND (Rand [1971]) on a test set data to evaluate the performance of the ALSSM presented in Chapter 6. However, the ARAND score is not used in the medical community and might not be a good enough metric to prove the reliability of such models.

Hence, even though a model is accepted as a good one by the machine learning community, it might not be viewed as such by the medical community. One reason for this is the lack of mutual understading regarding the approval process of the two communities. Collaborative works such as ours helps with this problem. Another problem is the way the approval process in the medicaly community works.

Minciullo et al. [2018] is an example where a computer vision method was rigorously tested by the medical community. However, this piece of work uses the Active Appearance Model which has been around since 2001 (Cootes et al. [2001]). This is an example where an extablished model in the machine learning community takes time to find itself accepted by the medical community.

[1]The percentage of the test data set that was correctly classified

In our opinion, there needs to be a shift in the way digital medical technology is evaluated. Currently, models are evaluated first from the machine learning community and then by the medical community. There needs to be a greater shift towards a joint assessment. Such an approach is inevitable given the rise of machine learning and artificial intelligence in day to day life. We are strong proponents of the systems illustrated in Figure 1-1. Indeed, such systems need to be made accessible to as many qualified practitioners as possible so that studies like the one performed by Minciullo et al. [2018] can be done more easily.

It is with this mindset that we believe that our work and any future work of the same nature should be evaluated in a hybrid manner. Convergence of our models is monitored in the typical machine learning fashion. For example, we use the accuracy versus the iteration number plot in Chapter 5 to conclude that our texture discriminator is trained. However, assessing this individual model for the medical community should be done alongside its other components (which is the shape model). Moreover this should be done so as to satisfy the demands of the medical community when it comes to their approval process.

We finish this section by noting that, even though new technology evaluation appears to be different between the two mentioned communities, they are basically doing the same thing. What we want is a technology that is robust to all possible scenarios. This is a fast process in the machine learning community as the evaluation involves numerical computations on an already labelled dataset. On the other hand, the medical community performs the labelling as well as the evaluation. It is the labelling part that is the most time consuming part. The hybrid system we show in Figure 1-1 will help with labelling while live testing for robustness.

1.6 Outline of the Thesis

The main body of this thesis is found in Chapters 4, 5 and 6. Chapter 4 is dedicated to the treatment of shape as a mathematical concept. We also describe how we can use the formulation of shape that we adopt to build a statistical model of a shape. We re-interpret the Active Shape Model (ASM) of Cootes et al. [1995] in a probabilistic way. In particular we show that under the dual treatment of probabilistic Principal Components Analysis (PCA), we are able to extend the ASM to model non linear maps from a latent space. We do this by

using Gaussian Process Latent Variable Models (GPLVM).

Chapter 5 is dedicated to defining texture. We do not have a generative texture model in this thesis, but instead consider ways in which we could build a discriminator. We use a CNN to predict areas of a hand x-ray that correspond to a one edge.

Our main contribution comes in Chapter 6. We argue that a GPLVM provides us with a better latent space shape model when viewed from a model specification point of view. We also show how to build a Bayesian model that incorporates uncertainty from the shape model, the texture model and the image quality. We perform a full marginalisation of the joint likelihood by using a Radial Basis Function (RBF). This is a significant contribution as we are able to provide error bounds on the integration we perform.

Much of the work leading to this thesis involved collecting and pre-processing data. We describe this process in Chapter 2. We build an annotation software used to delineate areas of interest. We then show how we can induce shape correspondence on our dataset, which consists of bones outlines. We believe that this is a novelty in itself as it reduces the complexity of delineating bone areas. Traditionally, landmarks had to be manually placed in specific regions. By using curves and their geometric properties as part of the data pre-processing, we eliminate the need for this exact placing of landmarks.

Chapter 3 is dedicated to key mathematical concepts used throughout this thesis. We describe variational Bayesian inference and how it is used in the Variational GPLVM that we use to determine the dimensionality of the shape model latent space. We then describe GPLVMs and interpret it as the dual of Probabilistic PCA. This allows us to extend the ASM model of Cootes et al. [1995]. We then give some error bounds on RBF interpolation of compactly supported functions.

Chapter 2

Data Aquisition

2.1 Introduction

We describe in this chapter how we collected the data used in this thesis. The Royal National Hospital for Rheumatic Diseases (RNHRD) has a database of X-rays taken from PsA patients. Much of these X-rays have been scored for PsA damage. These measurements were taken as part of the Long term Outcomes in Psoriatic Arthritis II (LOPAS II) study, the aim of which is to assess work disability in PsA.

A subset of the X-rays with no damage was collected. The data collection was conducted according to the principles of the Declaration of Helsinki and ethical approval was obtained from the National Research Ethics Services (NRES) Committee South West Wales Panel D. All patients included in this study gave full written informed consent for participation.

Each patient that had an X-ray scored has been assigned with a study number by the RNHRD. The X-rays were assigned this number along with the year that the X-ray was taken. For example a patient with study number CPSA001 who had a hand X-ray taken on January 4, 2015 would have the hand X-ray assigned the identifier `CPSA001h2015`. If that X-ray had been a foot then the identifier would change to `CPSA001f2015`. All of the X-rays were stored in a portable network graphics (png) format due to its lossless property.

The X-rays came with no information on the shape of bones. We describe in section 2.2 how this annotation was performed. The result of the annotation is an ordered point distribution representing curve outlines. These need to be cleaned for use in the shape model we describe in chapter 6. We describe in section 2.3

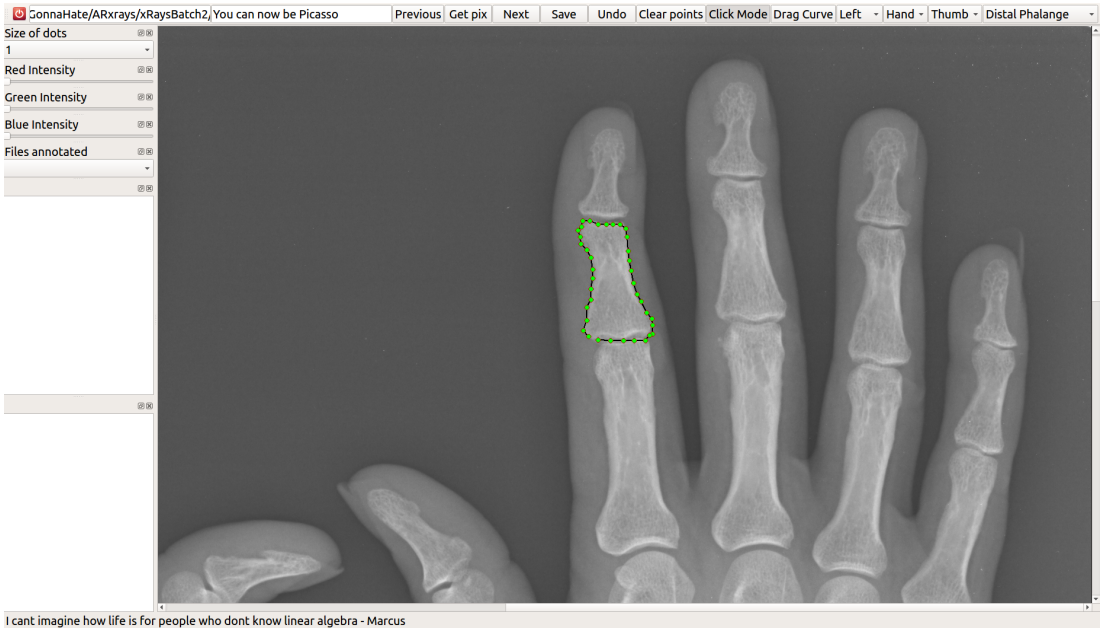


Figure 2-1: The figure shows the functionality of the Aspax Annotator 1.0 beta. It allows the user to mark the location of a bone by drawing a set of points around the bone that act as a discretised version of a curve.

how these points were upsampled and aligned.

Not all the collected X-rays came from the same health centre. As a result, they exhibit intensity variations coming from different exposure settings in the X-ray machines. This is something we want to get rid of as this will improve the performance of the texture models we describe in Chapter 5. We describe how this is done in section 2.4

2.2 Annotation

The data annotation process consisted of manually extracting bone outlines from the X-rays we collected. This was done by a trained rheumatologist[1] and hence was an expensive and time consuming process. This is why we only annotated the index finger of the right hand. This finger was chosen as it was previously shown to exhibit the most amount of damage in Tillett et al. [2016].

Outlines of the MC,PP,MP and DP were drawn using a piece of software called the ASPAX annotator. This was written in python3 using PyQt (Summerfield [2007]). It is a user interface that allows the user to draw polylines around each bone in a hand. As part of this annotation process, landmarks were placed on each bone. Figure 2-1 shows the annotator in use. 103 hand X-rays were thus

[1]Dr William Tillett

annotated.

The shape of each bone was saved as a point distribution array that represents a discretised version of the bone outline. We assume that this outline is a closed curve and hence the data collected for the n-th X-ray can be expressed as $\boldsymbol{y}_n = (\boldsymbol{y}_n^{(0)}, .., \boldsymbol{y}_n^{(J)})$, where $\boldsymbol{y}_n^{(j)} \in \mathbb{R}^2$ is the coordinate of the j-th point on the discretised curve.

2.3 Curve outline pre-processing

We want to use the curve outlines that we have drawn to model the shape of bones. As we describe in section 4.2, our definition of shape requires us to remove variations coming from rigid transformations of the bones as well as those coming from the parametric representation we adopt. In other words, we want each coordinate on one curve to be in correspondence with the same coordinates from other curves (see section 4.2.1).

Hence, the curves drawn using the ASPAX annotator have to be processed to remove the following sources of variability coming from the annotation process.

(a) The number of points d_i in each saved example is not the same.

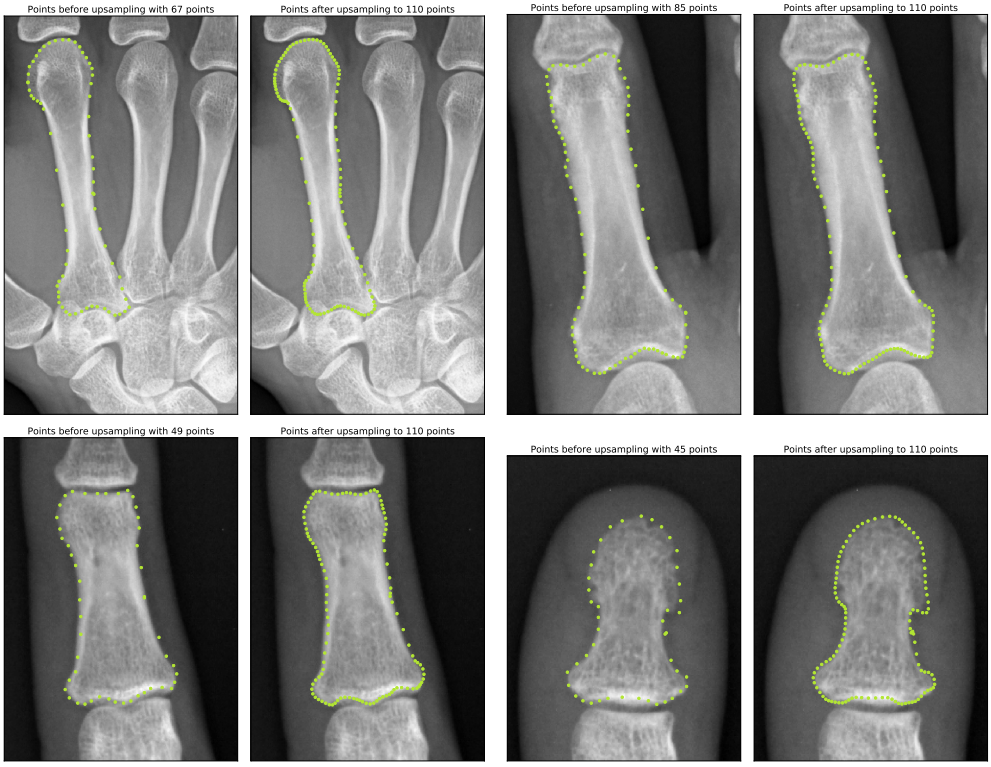
(b) The curves are not aligned as the X-rays were not all in the same pose.

(c) The curves do not have the same arc length.

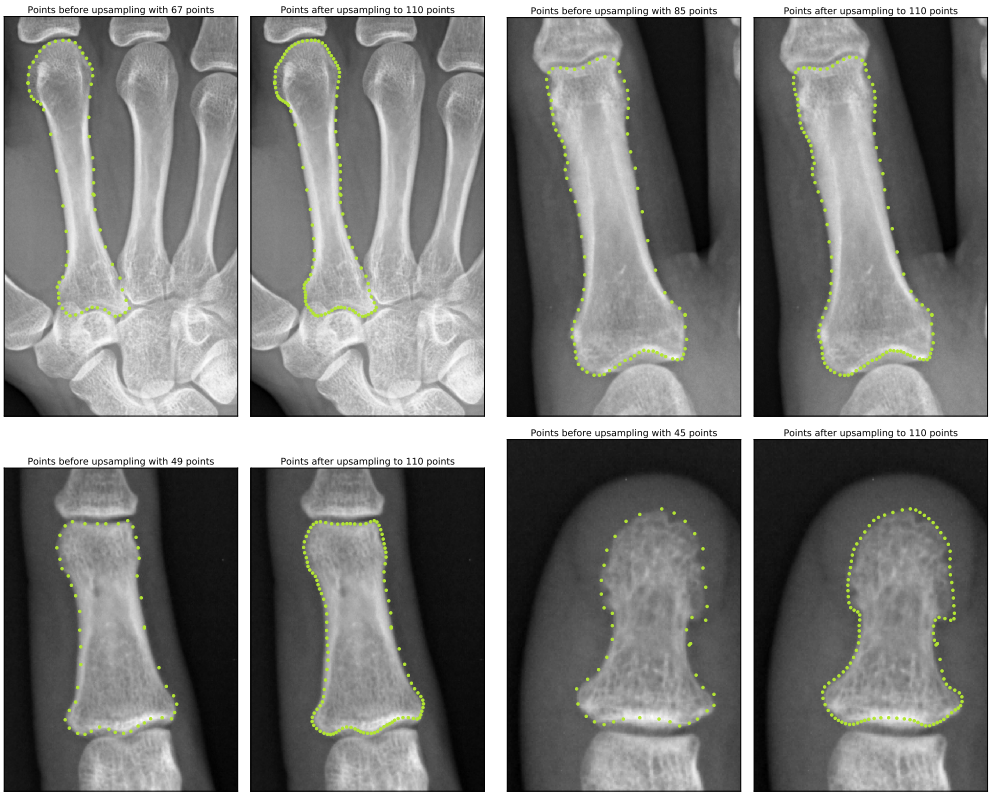
(d) The curves are not in parametric correspondence.

We show how to fix (a) in section 2.3.1. It is important to have the same number of points representing the same bone so as to be able to build statistical models. Indeed, we want this data to be saved as an array $\mathbf{Y} \in \mathbb{R}^{N \times P}$. P here is the dimension of the features of our data set, which in this case is the number of points that represent a bone outline as a curve. We have that $P = 2D$, where D is the number of points representing the discretised bone curve. N is the number of annotated curves. This allows us to use the methods described in section 4.3 to build a shape model.

Once (a) is done, we have curves that are not in shape correspondence. As described in section 4.2.1, this involves curves of unit arc length being in alignment and in parametric correspondence. Removing the variability described in (b),(c) and (d) is equivalent to putting the curves in shape correspondence. We

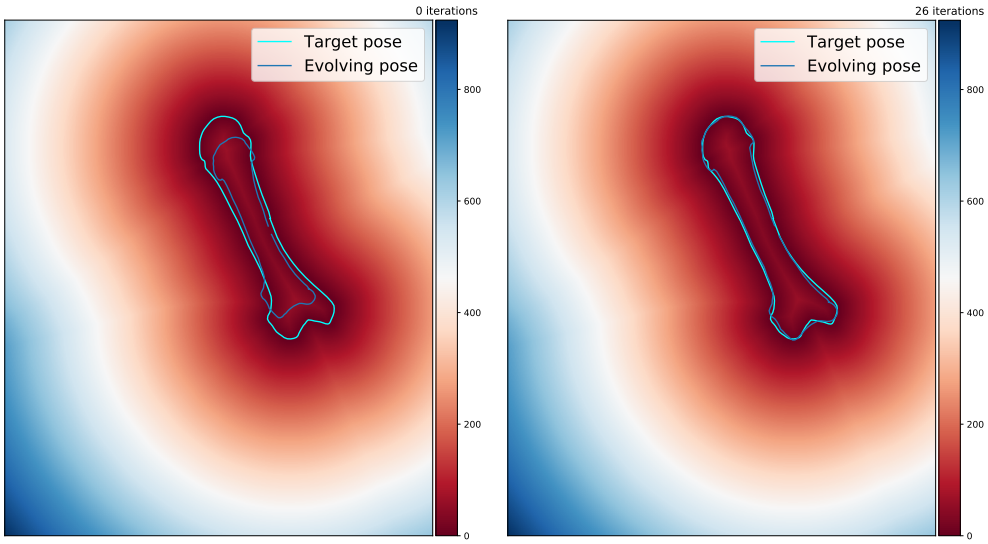


(a) Upsampling using a Fourier representation the bounding curve.

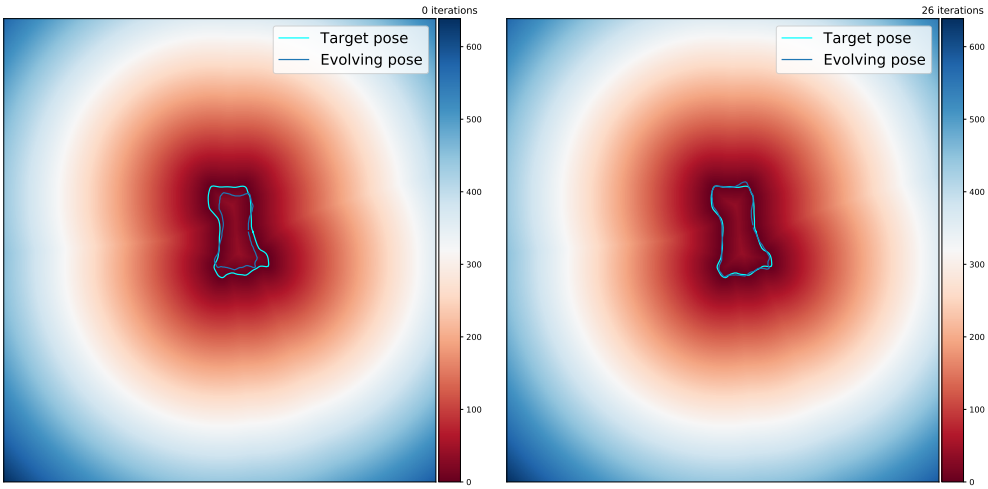


(b) Upsampling using a spline representation for the bounding curve.

Figure 2-2: The Figure shows the point distribution for the MC, PP, MP, and DP (clockwise from top left). In each case, the raw annotation is shown on the left and the upsampled points are shown on the right. The curves were upsampled to $D = 110$ points.



(a) MC curve being transformed onto a template.



(b) MP curve being transformed onto a template.

Figure 2-3: The figure shows the process of minimising $\sum_{d=1}^{D-1} \mathcal{D}_{\text{temp}}(\mathrm{T}_{r,\phi} \circ \hat{\boldsymbol{y}}^{(d)})$ with respect to the scale r and rotation angle ϕ where $\mathcal{D}_{\text{temp}}$ is the distance transform from the template curve and $\hat{\boldsymbol{y}}$ is curve that is to be made invariant to scale and rotation. The template curve is shown in cyan and its distance transform is shown as a colormap. The blue curve is being transformed to have the same scale and orientation as the cyan curve.

descibe this process in section 2.3.2. Once this is done, the only variation we see comes from shape and any statistical model we fit to this cleaned data will be a statistical shape model (as opposed to ones capturing pose variation).

2.3.1 Parametric curve representation

To have the same number of points in each discretised curve, a parametric form is fit to each curve and is used to sample the outline with the same number of points D. Let $\boldsymbol{m}_\theta : [0, 1] \to \mathbb{R}^2$ be a parametric representation for a bone outline with parameters θ. Then, fitting the representation to the current curve data

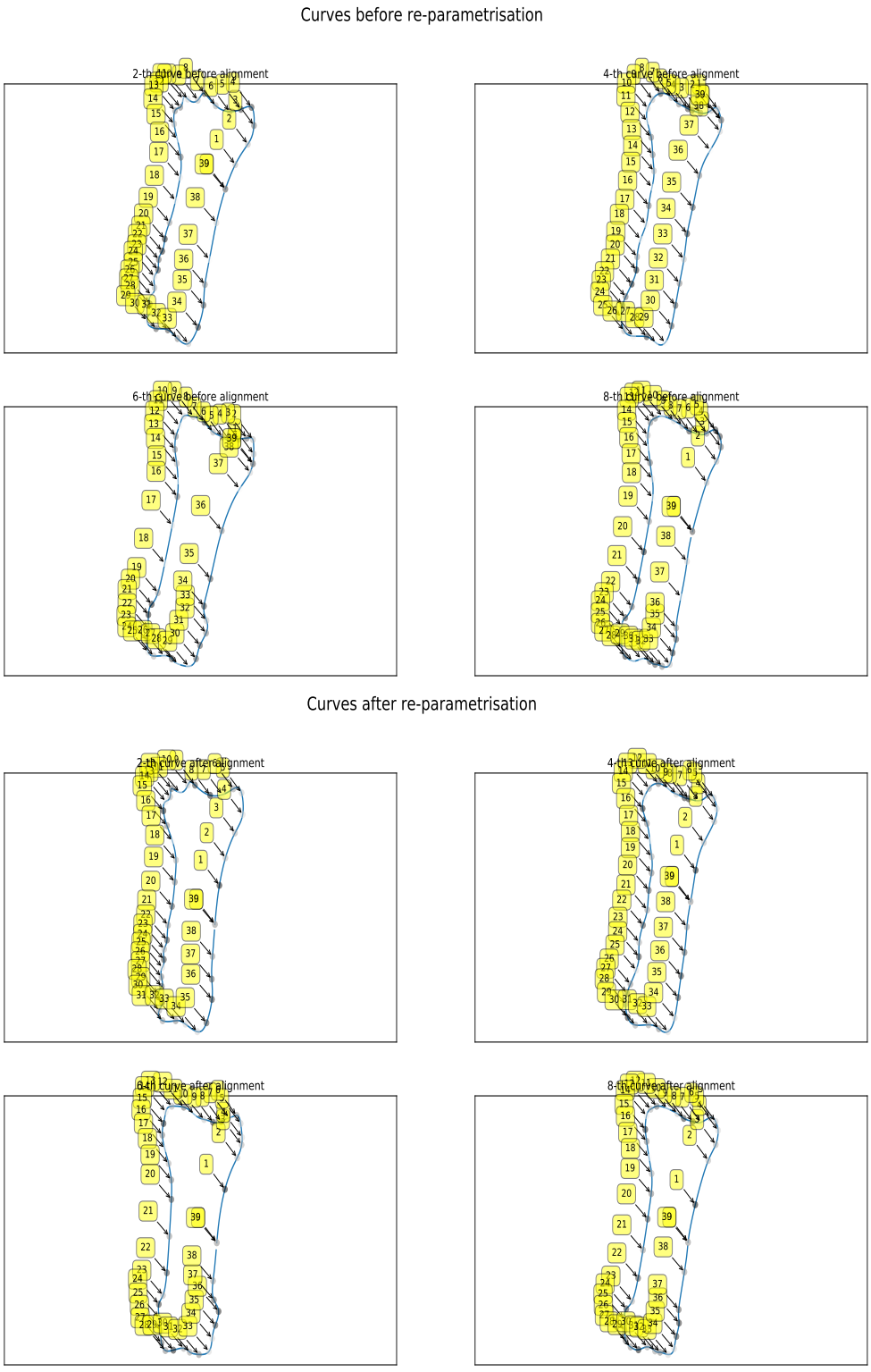


Figure 2-4: The Figure shows a set of PP curves that have been aligned but not yet set in parametric correspondence (top) and the same set of PP curves that have been now set in parametric correspondence (bottom). The bottom curves now exhibit shape correspondence.

$\boldsymbol{y} \in \mathbb{R}^{2J}$ is equivalent to finding

$$\theta^* = \arg\min_{\theta} \sum_{j=0}^{J} (\boldsymbol{m}_\theta(t_j) - \boldsymbol{y}^{(j)})^T (\boldsymbol{m}_\theta(t_j) - \boldsymbol{y}^{(j)}) \tag{2.1}$$

Let D be the desired number of points in the discretised representation of the curve. We set $s_d = \frac{d}{D}$ for $d = 0, ..., D-1$. Then the upsampled discretised curve is given by $\hat{\boldsymbol{y}} = (\boldsymbol{m}_{\theta^*}(s_0), ..., \boldsymbol{m}_{\theta^*}(s_{D-1}))$. Running this optimisation for each curve outlines yields a parametric representation $\boldsymbol{m}_n$ for each $\hat{\boldsymbol{y}}_n$.

The curve outlines drawn were closed curves with $\boldsymbol{y}^{(J)} = \boldsymbol{y}^{(0)}$. This implies that $t_0 = 0$ and $t_J = 1$. However, it is not clear what the values t_j for $0 < j < J$ should be. Ideally, equation (2.1) should be a minimised with respect t_j as well. We seek to upsample the curves in such a way that the point density around areas of high gradient in the curve is higher. Hence, our choice of t_j should depict this beaviour.

As shown in Figure 2-2 the raw data already was collected in such a way. Hence using a uniform parametrisation will force the parametric curve to have such a behaviour in these areas. Consequently, we found that setting $t_j = j/J$ worked well when trying to upsample the curves. Due to the continuity of the representation, if D is not too different from J, then this behaviour is preserved for the finer discretisation s_d. We now show two possible parametric representations for $\boldsymbol{m}_\theta$.

2.3.1.1 Fourier representation

One can express a closed curve as a Fourier expansion as follows:

$$\boldsymbol{f}_\theta(t) = (x(t), y(t)) = \left(\sum_{l=0}^{L-1} \alpha_l \cos(2l\pi t) + \beta_l \sin(2l\pi t), \sum_{l=0}^{L-1} \gamma_l \cos(2l\pi t) + \delta_l \sin(2l\pi t) \right) \tag{2.2}$$

For each curve, the set of coefficients is

$$\boldsymbol{\theta} = (\alpha_1, ..., \alpha_{L-1}, \beta_1, ..., \beta_{L-1}, \gamma_1, ..., \gamma_{L-1}, \delta_1, ..., \delta_{L-1}).$$

It was found by doing a least-squares fit in SciPy (Jones et al. [2001–]). Such a representation ensures that the curves are closed due to the periodic nature of the basis functions. The results are shown in Figure 2-2a.

2.3.1.2 Spline representation

We can represent a 1-d function with domain $[0, 1]$ with piecewise polynomials. Let $0 =: t_0 < t_1 < ... < t_J := 1$ be the control points as previously described and define intervals $I_j = [t_j, t_{j+1})$ for $j = 0, ..., J-1$. Then we have that a polynomial spline is given by

$$\mathcal{S}_\theta(t) = \sum_{j=0}^{J-1} \mathbb{I}_{t \in I_j} \mathcal{P}_j(t) \tag{2.3}$$

where $\mathcal{P}_j$ is a polynomial of degree O given by

$$\mathcal{P}_j(t) = a_{j,0} + \sum_{o=1}^{O} a_{j,o}(t - t_j)^o \tag{2.4}$$

The parameters are given by

$$\boldsymbol{\theta} = \{a_{j,o} : o = 0, .., O, \quad j = 0, .., J-1\}$$

They are found by solving a linear system which consist of the following conditions

$$\begin{aligned} \mathcal{P}_j(t_j) &= y^{(j)} \\ \mathcal{P}_{j-1}(t_j) &= y^{(j)} \end{aligned} \tag{2.5}$$

for $j = 0, .., J-1$. We impose the following gluing condition for $o = 1, .., O-2$ that ensures picewise smoothness of up to degree $O-1$

$$\frac{\mathrm{d}^o}{\mathrm{d}t^o}\mathcal{P}_j(t_j) = \frac{\mathrm{d}^o}{\mathrm{d}t^o}\mathcal{P}_{j-1}(t_j) \tag{2.6}$$

for $o = 1, .., O-1$. To impose periodicity we have set the boundary coundition given by

$$\mathcal{P}_{d-1}(1) = \mathcal{P}_0(0) \tag{2.7}$$

We use a cubic spline (O=3) representation on each coordinate of the curve. That is

$$\boldsymbol{f}_\theta(t) = (\mathcal{S}_x(t), \mathcal{S}_y(t)) \tag{2.8}$$

We fit the curves and perform the upsampling by using the interpolation module in SciPy (Jones et al. [2001–]). The results are shown in Figure 2-2b.

2.3.2 Imposing shape correspondence

After upsampling, we have a set of discretised curves $\hat{\mathbf{Y}} = (\hat{\boldsymbol{y}}_0, .., \hat{\boldsymbol{y}}_{N-1})$ for each bone in the index finger. To impose shape correspondence on $\hat{\mathbf{Y}}$, we remove the effect of rotation and scaling from the centred curve outlines. Once this is done, we use the process described in Campbell and Kautz [2014] to induce parametric correspondence on these curves.

2.3.2.1 Removing rotation and scale

We want to rotate and scale the curves in our dataset to match the orientation and size of a template curve $\hat{\boldsymbol{y}}_{\text{temp}}$ chosen from the dataset. To do this, we build a map representing the distance transform from this curve, i.e. for $\boldsymbol{x} \in \mathbb{R}^2$ we define

$$\mathcal{D}_{\text{temp}}(\boldsymbol{x}) = \min_{d \in \{0,\dots,D-1\}} \|\boldsymbol{x} - \hat{\boldsymbol{y}}^{(d)}_{\text{temp}}\| \tag{2.9}$$

Now let $\mathrm{T}_{r,\phi}$ represent a scaled rotation on $\mathbb{R}^2$, i.e. for $\boldsymbol{x} = (x_0, x_1)$ we have that

$$\mathrm{T}_{r,\phi} \circ \boldsymbol{x} = (rx_0\cos(\phi) - rx_1\sin(\phi), rx_0\cos(\phi) + rx_1\sin(\phi)). \tag{2.10}$$

Then we have that for an upsampled curve $\hat{\boldsymbol{y}}$, solving the following minimisation problem

$$(r^*, \phi^*) = \arg\min_{r,\phi} \sum_{d=1}^{D-1} \mathcal{D}_{\text{temp}}(\mathrm{T}_{r,\phi} \circ \hat{\boldsymbol{y}}^{(d)}) \tag{2.11}$$

and gives us a scaling factor r^* and an angle of rotation ϕ^* that would transform $\hat{\boldsymbol{y}}$ so that it has the same scale and orientation as the template curve $\hat{\boldsymbol{y}}_{\text{temp}}$.

We perform this operation for each curve in the upsampled dataset. This can be seen in Figure 2-3 We have then for each curve $\hat{\boldsymbol{y}}_n$, a transformation T_n, that when applied to $\hat{\boldsymbol{y}}_n$ yields a dataset that is invariant to rigid tranformations. We denote this data set by $\hat{\mathbf{Y}}_{\mathrm{T}} = (\mathrm{T}_0 \circ \hat{\boldsymbol{y}}_0, ..., \mathrm{T}_{N-1} \circ \hat{\boldsymbol{y}}_{N-1})$. Without loss of generality, we can assume that at this point, the data we have consists of curves of unit arc length.

2.3.2.2 Setting the curves in parametric correspondence

From now on, we assume that $\hat{\mathbf{Y}} = \hat{\mathbf{Y}}_{\mathrm{T}}$. We introduce the following notation. Let $\mathbf{S} = (\boldsymbol{s}_0, ..., \boldsymbol{s}_{N-1})$ where $\boldsymbol{s}_n = (s_n^{(0)}, ..., s_n^{(D-1)})$ be the arc length at which

the curves $\hat{\mathbf{Y}}$ are sampled. That is for a discrete curve $\hat{\boldsymbol{y}}_n$ in our dataset with parametric representation $\boldsymbol{m}_n$ we have that $\hat{\boldsymbol{y}}_n = \left[\boldsymbol{m}_n\left(s_n^{(0)}\right), ..., \boldsymbol{m}_n\left(s_n^{(D-1)}\right)\right]$.

As we describe in section 4.2.1, the defining factor for curves of unit length to be in parametric correspondence is for them to have the same geometry at the equivalent parametric positions. In our case, this means that the geometric properties of our discrete curves must be the same at every coordinate $\hat{\boldsymbol{y}}_n^{(d)}$.

This can be done by finding the arc length $s_n^{(d)}$ for each curve and each coordinate from which to sample the data point $\hat{\boldsymbol{y}}_n^{(d)}$ from the parametric form $\boldsymbol{m}_n$. Note that this form needs to be found again as we have rotated and scaled our data in section 2.3.2.1. We use an energy minimisation approach similar to that used by Campbell and Kautz [2014] to achieve this. We recover $\mathbf{S}$ by minimising the following energy function:

$$\begin{aligned} E(\mathbf{S}) = &\sum_{n=0}^{N-1}\left[1 - \exp\left(-\frac{\gamma}{2D}\sum_{d=0}^{D-1}(\kappa_n(s_n^{(d)}) - \bar{\kappa}^{(d)})^2\right)\right] \\ &+ \sum_{n=0}^{N-1}\sum_{d=0}^{D-1}\left[s_n^{(d+1)} - s_n^{(d)} - \frac{1}{D}\right]^2 \end{aligned} \tag{2.12}$$

where κ_n is the curvature operator applied to the $n-$th curve example and $\bar{\kappa}^{(d)}$ is the mean curvature at coordinate d. The first term gathers the sampling points so that they are close to the mean curvature at their respective coordinates. The second term ensures that they are somewhat equally spaced.

This optimisation also requires the monotonicity constraint $s_n^{(d+1)} > s_n^{(d)}$ for all n and for $d = 0, ..., D-1$. We use the identity

$$s_n^{(d)} = \frac{\sum_{k=0}^{d}\exp(r_n^{(d)})}{\sum_{k=0}^{D-1}\exp(r_n^{(k)})} \tag{2.13}$$

to force this constraint and optimise (2.12) with respect to $r_n^{(d)}$ instead using SciPy (Jones et al. [2001–]). We show the result of this optimisation in Figure 2-4.

2.4 Intensity normalisation

As we describe in section 5.2, the texture around a pixel neighbourhood is the normalised filtered intensities around that neighbourhood. We find that different

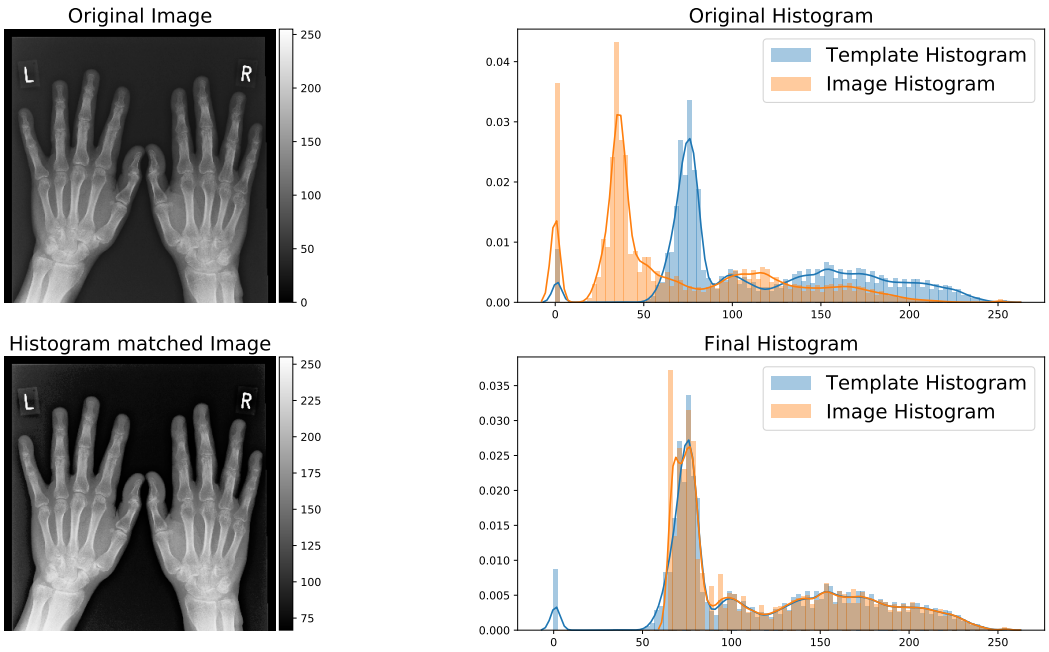


Figure 2-5: The Figure shows the intensity normalisation process. The input X-ray is shown on the top left. The same X-ray is shown in the bottom left after its histogram was matched to that of a template X-ray. The change in the histograms is shown on the right plots, where it can be seen that the input X-ray now has the same histogram as the template X-ray.

test centres participating in the LOPAS II study have different exposure settings on their X-ray acquisition process. Hence, there is a need to correct the X-rays in our data for exposure differences. We perform histogram matching to achieve this.

2.4.1 Histogram Matching

Given two images $\mathbf{U}_1, \mathbf{U}_2 \in [0, 255]^{N_x \times N_y}$ we want to induce a transformation on $\mathbf{U}_2$ such that the distribution of pixel intensities of the two images match each other. This process is called histogram matching. We call $\mathbf{U}_2$ the input image and $\mathbf{U}_1$ the target image. We briefly describe how this is done.

We split the range $[0, 255]$ into B equal intervals $\mathcal{I}_b$ (bins). Each image $\mathbf{U}_1, \mathbf{U}_2$ induces a cumulative density function F_1, F_2 respectively on these intervals. Now let $\mathcal{I}_2$ be the interval for which $F_2(\mathcal{I}_2) = F_1(\mathcal{I}_1)$ for some $\mathcal{I}_1$. That is $\mathcal{I}_1$ and $\mathcal{I}_2$ are the intervals at which the CDF of $\mathbf{U}_1$ and $\mathbf{U}_2$ match each other.

Now, if for the input image, the locations at which the pixels fall into $\mathcal{I}_2$ are known, then we can replace the pixel values at these locations with some pixel value that falls into the interval $\mathcal{I}_1$. This process shifts the mass on the pixel

values around until the quantiles in each image are the same. This is repeated for all the B intervals we have defined.

2.4.2 Histogram matching the dataset

To correct the intensity variations introduced by the different exposure settings, we choose a template X-ray that was captured at the RNHRD[2]. We then sequentially perform the operation just described on the X-rays coming from different acquisition centres. We choose an example from the RNHRD because it is the centre that generated the most data. The result of histogram matching on a test X-ray is shown in Figure 2-5.

[2]Royal National Hospital for Rheumatic Diseases

Chapter 3

Mathematical Tools

3.1 Introduction

We give a short introduction to the concepts used in this thesis. The core philosophy underpinning the models we later introduce is Bayesian Inference. We dedicate section 3.2 to a discussion of the Bayesian philosophy. We also demonstrate the use of variational inference, where a so called variational distribution is used to approximate an intractrable integral. We then show in section 3.2.3 how one can think of segmentation as a Bayesian problem.

We introduce Gaussian Process Latent Variable models in section 3.3. These are used in Chapter 4 to model shape. We show how these are derived from a latent model Directed Acyclic Graph (DAG) in section 3.3.4. This derivation reveals an important dual relationship between probabilistic principal components analysis and GPLVMs. We use this dual relationship later in section 6.2.1 as part of our argument justifying a move away from AAMs (Cootes et al. [2001]).

One of the contributions of this thesis is the exact marginalisation that we present in section 6.2.4. This uses a Radial Basis Function (RBF) interpolant to approximate an intractable integral. We introduce RBF interpolation in section 3.4. We also give some error estimate for the integration we perform in chapter section 6.2.4.

3.2 Bayesian Inference

The Bayesian philosophy comes from the belief that one cannot measure every state of a system. Models that we use to explain the world will therefore have an inherent uncertainty associated to it. The major drawback of mathemati-

cal modelling is that models come from the subjective human observer. What Bayesians truly seek to quantify is the uncertainty we should observe given the assumptions we make about the system. Probability theory provides us with a framework within which this can be attempted.

In the eyes of the Bayesian Scientist, every statement coming from a particular model of a system is a statement about the distibution of the state of the system given the model used to explain it. Hence, Bayesian Inference is very attractive when it comes to formalising this belief. Mathematically, one would like to make statements about events that are conditioned upon each other. This builds on the definition of independent events. We say A, B are independent events iff $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$. We also define the conditional probability of A given B as

$$\mathbb{P}(A|B) := \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

Coming back to the more general Bayesian philosophy, this is useful as one can then make statements about a model $\mathcal{M}$ which predicts states $\boldsymbol{y}$ of the system that take the form $p(\mathcal{M}|\boldsymbol{y})$. I.e., we make a probabilistic statement about how good the model is given the observations we have of the system $\boldsymbol{y}$. We usually have an expression for $p(\boldsymbol{y}|\mathcal{M})$. Bayes' Rule allows us to invert this probability:

Theorem 3.1 (Bayes' Rule)**.** *Let $(\mathcal{Z}, \mathcal{F}, \mathbb{P})$ be a probability space, with $\mathcal{Z}$ being the sample space, $\mathcal{F}$ being the $\sigma-$algebra and $\mathbb{P}$ being the probability measure. Let $A, B \in \mathcal{F}$ be events, then we have that:*

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}$$

Bayesian methods have experienced a rise in popularity in the statistics community since the discovery of Markov Chain Monte Carlo (MCMC) methods (Hastings [1970]) which made it possible to iteratively compute $p(\mathcal{M}|\boldsymbol{y})$. In more recent years, cheaper and more powerful computers along with research into efficient sampling methods have greatly increased the popularity of Bayesian Inference. For example in uncertainty quantification, one seeks to quantify the error in the spatial dependent diffusion coefficient a through the evolution of the following PDE:

$$-\nabla \cdot (a \nabla u) = f \text{ on } D \subset \mathbb{R}^d$$

Using Multilevel Monte Carlo methods (Giles [2008]) along with efficient

solvers, one is able to compute the distribution of a given noisy observations of the system. More recently, Rynn et al. [2019] uses Bayesian Inversion to get a distribution instead of a point value for the conductivity of a material. In the machine learning community, priors are put on the weights of neural networks (Kingma and Welling [2013]) to quantify their uncertainty. The computation in this case relies on variational optimisation instead of sampling methods such as MCMC.

This all shows that the Bayesian Philosophy is starting to spread in all aspects of science and hints towards a general movement where scientists start to acknowledge the need to quantify the deviation of models from reality. In particular, science is moving towards recognising that from our subjective point of view, there cannot be just one state of the observed universe. We rather have to treat our observations as realisations of some hidden process. While this is happening, new discoveries are making it easier to apply this philosophy in practice through Bayesian inference. At the risk of being self referential, it is unclear whether the change of philosophy caused the rise of Bayesian inference or vice versa.

3.2.1 Setting

Traditionally, we set up a **likelihood** model $\boldsymbol{y} \sim p(\boldsymbol{y}; \theta)$ for the observed vector of data $\mathbf{Y} = (\boldsymbol{y}_0, ..., \boldsymbol{y}_{N-1})$. In other words, data is just realisations of a random variable having probability density function (pdf) $p(\boldsymbol{y}; \theta)$. Here θ parametrises $p(\boldsymbol{y}; \theta)$, and is sometimes called the model parameters. One can then calculate the likelihood of the data observations which we denote by $p(\mathbf{Y}; \theta) = \prod_{n=0}^{N-1} p(\boldsymbol{y}_n; \theta)$. The aim of model fitting then is to find the optimal value of θ which explains the observed data. This is achieved traditionally via Maximum Likelihood Estimation (MLE) where we find $\arg\max_\theta p(\mathbf{Y}; \theta)$.

In Bayesian Inference, one would treat the paramaters of the likelihood model to be random variables themselves. These would have their own **prior** distributions $p(\theta; \alpha)$. The likelihood model is now a conditional distribution $p(\boldsymbol{y}|\theta; \alpha)$ with α now acting as hyper-parameters. The main aim of Bayesian Inference is to quantify the uncertainty in model parameters. The prior distributions on the data likelihood model is what captures this uncertainty.

Introducing a likelihood model and a prior distribution automatically generates a dependence structure in the model. A Directed Acyclic Graph (DAG) is

used to simplify the dependence structure of variables. This can be seen in Figure 3-1. The arrows represent the dependence of the variables with respect to each other. In our case, $\mathbf{Y}$ depends on θ. An arrow thus points from θ to $\mathbf{Y}$. Here, we point out that the dependence structure that we refer to is not necessarily a statement about causation. Rather, we are saying that if we knew the value of α, we would be able to model the data through the prior and likelihood model.

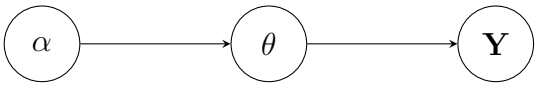


Figure 3-1: The Figure shows a simple DAG. The arrows shows the dependence structure of the variables

One is usually interested in the posterior distribution, which using Theorem 3.1 is given by

$$p(\theta|\mathbf{Y};\alpha) = \frac{p(\mathbf{Y}|\theta;\alpha)p(\theta;\alpha)}{p(\mathbf{Y};\alpha)}. \tag{3.1}$$

We call

$$\begin{aligned} p(\mathbf{Y};\alpha) &= \int p(\mathbf{Y},\theta;\alpha)d\theta \\ &= \int p(\mathbf{Y}|\theta;\alpha)p(\theta;\alpha)d\theta \end{aligned} \tag{3.2}$$

the marginal likelihood. The posterior distribution is what quantifies the uncertainty with the parameters of the model we want to fit. One would like to sample from this posterior. Most sampling techniques require the normalising factor of the distribution to be known. The normalising factor in equation (3.2) is often hard to compute, making traditional sampling techniques inappropriate.

Markov Chain Monte Carlo methods are popular when trying to sample from target distributions with intractable normalising factors. These methods have several drawbacks. The samples come from a Markov Chain and hence are correlated. To get independent draws from the target distribution, one would have to thin the chain, i.e. only keep samples from the chain at every l steps. l here is the number of steps before the autocorrelation is sufficiently low for the n-th and $n+l$-th sample to be considered uncorrelated. Secondly, these methods can be very slow with runtimes of the order of weeks being very common. In general, MCMC methods are not used in Machine Learning for these reasons.

Instead of sampling from the posterior, we can maximise the posterior distribution with respect to θ, thus obtaining the mode of the posterior. This method is called maximum a posterior probability (MAP) estimation. The result is a point estimate $\arg\max_\theta p(\mathbf{Y}|\theta;\alpha)p(\theta;\alpha)$. The objective to be maximised is a function of α. Often, this is an unknown value and needs to be computed as well.

Recently, there has been a strong focus in the machine learning community to directly optimise the marginal likelihood with respect to the hyperparameters α. Doing this is equivalent to finding the distribution parametesrised by α that explains the data. We describe how this is done in the next section.

3.2.2 Variational Inference

In what follows, we drop α from our notation. Our aim is to maximise the marginal posterior introduced in the previous section or equivalently, to maximise the log marginal likelihood

$$\log p(\mathbf{Y}) = \int p(\mathbf{Y}|\theta)p(\theta)\mathrm{d}\theta$$

Often, the above integral is intractable. We can instead optimise the lower bound induced by the introduction of another distribution $\mathcal{Q}(\theta)$ as follows:

$$\begin{aligned}
\log p(\mathbf{Y}) &= \log\left(\int \frac{p(\mathbf{Y},\theta)}{\mathcal{Q}(\theta)}\mathcal{Q}(\theta)\mathrm{d}\theta\right) \\
&\geq \int \log\left(\frac{p(\mathbf{Y},\theta)}{\mathcal{Q}(\theta)}\right)\mathcal{Q}(\theta)\mathrm{d}\theta \\
&\quad \text{By Jensen's inequality} \qquad (3.3)\\
&= \int \log\left(\frac{p(\mathbf{Y}|\theta)p(\theta)}{\mathcal{Q}(\theta)}\right)\mathcal{Q}(\theta)\mathrm{d}\theta \\
&= \int \log\left(p(\mathbf{Y}|\theta)\right)\mathcal{Q}(\theta)d\theta - \int \log\left(\frac{\mathcal{Q}(\theta)}{p(\theta)}\right)\mathcal{Q}(\theta)\mathrm{d}\theta =: \mathcal{L}(\mathcal{Q})
\end{aligned}$$

We also have that

$$
\begin{aligned}
&\log p(\mathbf{Y}) - \mathcal{L}(\mathcal{Q}) \\
=&\log\left(\int \frac{p(\mathbf{Y},\theta)}{\mathcal{Q}(\theta)}\mathcal{Q}(\theta)\mathrm{d}\theta\right) - \int \log\left(\frac{p(\mathbf{Y},\theta)}{\mathcal{Q}(\theta)}\right)\mathcal{Q}(\theta)\mathrm{d}\theta \\
=&\log\left(\int p(\mathbf{Y},\psi)\mathrm{d}\psi\right) - \int \log\left(\frac{p(\mathbf{Y},\theta)}{\mathcal{Q}(\theta)}\right)\mathcal{Q}(\theta)\mathrm{d}\theta \\
&\text{Through cancellation of } Q\text{, and the introduction of dummy variable } \psi \text{ to replace } \theta \\
=&\int \log\left(\int p(\mathbf{Y},\psi)\mathrm{d}\psi\right)\mathcal{Q}(\theta)\mathrm{d}\theta - \int \log\left(\frac{p(\mathbf{Y},\theta)}{\mathcal{Q}(\theta)}\right)\mathcal{Q}(\theta)\mathrm{d}\theta \\
&\text{as } \mathcal{Q}(\theta) \text{ should integrate to 1} \\
=&-\int \log\left(\frac{p(\mathbf{Y},\theta)}{\int p(\mathbf{Y},\psi)\mathrm{d}\psi\mathcal{Q}(\theta)}\right)\mathcal{Q}(\theta)\mathrm{d}\theta \\
=&-\int \log\left(\frac{p(\mathbf{Y},\theta)}{p(\mathbf{Y})\mathcal{Q}(\theta)}\right)\mathcal{Q}(\theta)\mathrm{d}\theta \\
=&-\int \log\left(\frac{p(\theta|\mathbf{Y})}{\mathcal{Q}(\theta)}\right)\mathcal{Q}(\theta)\mathrm{d}\theta \\
=&\int \log\left(\frac{\mathcal{Q}(\theta)}{p(\theta|\mathbf{Y})}\right)\mathcal{Q}(\theta)\mathrm{d}\theta \\
=:&\mathrm{D}_{\mathrm{KL}}(\mathcal{Q}||p_{\mathrm{post}})
\end{aligned}
\tag{3.4}
$$

where $\mathrm{D}_{\mathrm{KL}}(\mathcal{Q}||p_{\mathrm{post}})$ is the Kullback-Leibler divergence from the posterior $p(\theta|\mathbf{Y})$ to $\mathcal{Q}(\theta)$. Making the lower bound $\mathcal{L}(\mathcal{Q})$ a tight one is equivalent to finding $\mathcal{Q}(\theta)$ that minimises $\mathrm{D}_{\mathrm{KL}}(\mathcal{Q}||p_{\mathrm{post}})$. In other words, we are approximating the posterior with $\mathcal{Q}$. We call $\mathcal{Q}$ in such a setting **the variational distribution**. Using variational analysis and the convexity of the Kullback-Leibler divergence, we find that $\mathcal{Q}(\theta) \propto p(\theta|\mathbf{Y})$ is a minimiser.

However, this fact alone does not solve the problem of an intractable integral. Moreover, we seek to get an expression whose gradients can be computed so that some sort of gradient based optimisation can be done. $\mathcal{Q}$ need to be chosen so that these conditions are satisfied. Hence variational inference, in general, tends to be more of an art when it comes to choosing the right form of $\mathcal{Q}$. In section 3.3.6 we show how such a distribution can be chosen when fitting a Variational GPLVM.

3.2.3 Segmentation as a Bayesian Problem

The task of any segmentation process is, given a noisy image u_0, to find the sets O_i whose union forms the domain Ω of the intensity function. We point the

reader to Younes [2010] for a mathematical presentation of segmentation based on geometrical considerations.

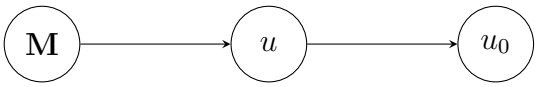
3.2.3.1 Mumford Shah

The Mumford Shah model (Vese and Chan [2002]) is a popular denoising tool that also identifies regions O_i for $i = 1, ..., N$ as being areas where the image intensity is smooth. The aim of the model is to identify the ideal, uncorrupted image intensity profiles and the edge set $\mathbf{M} = \{\partial O_i\}$.

As described in Section 3.2.1, we set a likelihood model $p(u_0|u, \mathbf{M}; \mathrm{X})$ and priors $p(\mathbf{M})$ and $p(u|\mathbf{M})$ where u is the ideal image intensity function.

$$\begin{aligned} p\left(\cup_{i=1}^{N} O_i | u_0\right) &= p\left(\mathbf{M}, u | u_0\right) \\ &\propto p(u_0|u)p(u|\mathbf{M})p(\mathbf{M}) \end{aligned} \tag{3.5}$$

This generates the following DAG:



For simplicity, we set $N = 2$. The acquisition noise is treated as Gaussian noise with a certain variance r, that is:

$$p(u_0|u) \propto \frac{1}{\sqrt{2r\pi}} \exp\left(-\frac{1}{2r}\int_{\Omega} \left(u(\boldsymbol{x}) - u_0(\boldsymbol{x})\right)^2 \mathrm{d}\boldsymbol{x}\right) \tag{3.6}$$

The edge set is assumed to be a smooth curve. That is $\mathbf{M} = \boldsymbol{m}$ for some curve. A prior is set on the length of the curve. As in the previous equation, we have a variance parameter w controlling the length of the curve.

$$p(\mathbf{M}) = \exp\left(-\frac{1}{2w}|\boldsymbol{m}|\right) \tag{3.7}$$

The ideal intensity profile is also forced to be smooth inside the different regions. A variance parameter s controls how smooth the function should be. In the case of the two region segmentation algorithm the, prior is given by:

$$p(u|\mathbf{M}) \propto \frac{1}{\sqrt{2s\pi}} \exp\left(-\frac{1}{2s}\int_{O} \nabla|u(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} - \frac{1}{2s}\int_{O^c} |\nabla u(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x}\right) \tag{3.8}$$

To find the edge set, the negative log of the posterior is optimised and the ideal image u^* and the edge set Γ^* are given by:

$$\arg\min_{u,\Gamma} \frac{1}{r}\int_{\Omega}\left(u(\boldsymbol{x})-u_0(\boldsymbol{x})\right)^2 \mathrm{d}\boldsymbol{x} + \frac{1}{s}\int_{O}\nabla|u(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x} + \frac{1}{s}\int_{O^c}|\nabla u(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x} + \frac{1}{w}|\partial O| \tag{3.9}$$

The solution to equation (3.9) involves the evolution of the intensity function u as well as that of the edge set $\mathbf{M}$. The dynamics for such an optimisation is described in Younes [2010]. It closely ressembles the set of equations for the moving front or Stefan Problem (Rubinšteĭn [2000]). The numerical solution of this problem has been well studied and allows one to solve this problem numerically. Moreover, Ambrosio and Tortorelli [1990] provides an approximation that can be optimised instead of the full functional in equation (3.9).

3.2.3.2 Snakes Model

The snakes model (Kass et al. [1988]) is a simpler segmentation algorithm that only requires the evolution of a curve on the image domain. For simplicity, we again set the number of regions to be two. Like in the Mumford Shah, the edge set is considered to be a smooth curve $\boldsymbol{m}$. However, we have an expression for $p(\mathbf{M}|u)$ that also incorporates the smoothness of the curve as a regulariser. The model does not consider any acquisation noise and hence, $u = u_0$. We have

$$p(\mathbf{M}|u) \propto \exp\left(-\frac{1}{2}\int V_u\left(\boldsymbol{m}(t)\right)\mathrm{d}t - \lambda\int\left|\nabla\boldsymbol{m}(t)\right|^2\mathrm{d}t - \beta\int\left|\Delta\boldsymbol{m}(t)\right|^2\right) \tag{3.10}$$

where V_u is a potential that increases away from areas of high gradient in the image function u. Ideally, the curve should lie at an area where the gradient is high. Instead of forcing smoothness on the ideal intensity function u, it requires smoothness of the boundary and hence, a prior is set on the lapacian. We would also like the curve to be the one of minimum length, hence the term involving the gradient of the curve.

3.3 Gaussian Process Latent Variable Models

We introduce Gaussian Process Latent Variable Models. They can be interpreted as a non-linear dimensionality reduction technique. It extends on the ideas of Probabilistic Principal Component Analysis as presented in Tipping and Bishop [1999]. Titsias and Lawrence [2010] later introduced a bayesian version of GPLVM (Bayesian GPLVM) where priors were introduced on the weights. We explore the numerics of this model later in this Chapter. We first describe Gaussian Processes and then move on to describing GPLVMs.

3.3.1 Gaussian Processes: A quick overview

Definition 3.2 (Gaussian Processe)**.** *Let $Q \in \mathbb{N}$, a stochastic process $Z : \Omega \times \mathbb{R}^Q \to \mathbb{R}$ is called a Gaussian process if for any finite collection $\boldsymbol{t}_0, ..., \boldsymbol{t}_{N-1} \in \mathbb{R}^Q$, we have that $\boldsymbol{Z} = Z(\boldsymbol{t}_0), ..., Z(\boldsymbol{t}_{N-1})$ jointly follows a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{K} \in \mathbb{R}^{Q \times Q}$ the joint density of $\boldsymbol{Z}$ is given by:*

$$\mathcal{N}(\boldsymbol{Z}|\boldsymbol{\mu}, \boldsymbol{K}) = \frac{1}{\sqrt{det(2\pi \boldsymbol{K})}} \exp\left(-\frac{1}{2}(\boldsymbol{Z} - \boldsymbol{\mu})^T \boldsymbol{K}^{-1}(\boldsymbol{Z} - \boldsymbol{\mu})\right)$$

A Gaussian process comes with a symmetric, positive definite kernel k that gives the covariance between the Gaussian Process evaluated at two indexing points

$$k(\boldsymbol{t}_1, \boldsymbol{t}_2) = \mathrm{Cov}(Z(\boldsymbol{t}_1), Z(\boldsymbol{t}_2)) \tag{3.11}$$

The covariance matrix in Definition 3.2 is given by $\mathbf{K}_{i,j} = k(\boldsymbol{t}_i, \boldsymbol{t}_j)$ for $i, j = 0, ..., N-1$. See Rasmussen [2006] for more details of how to handle covariance kernels. The Gaussian process also has a mean function

$$m(\boldsymbol{t}) = \mathbb{E}[Z(\boldsymbol{t})] \tag{3.12}$$

We have that $\boldsymbol{\mu} = (m(\boldsymbol{t}_0), ..., m(\boldsymbol{t}_{N-1}))$. One would often define a Gaussian process by specifying the covariance kernel and mean function explicitly. We then write $\boldsymbol{Z} \sim \mathrm{GP}(m(\cdot), k(\cdot, \cdot))$ to specify the distribution of the Gaussian Process Z.

3.3.2 Gaussian Process Regression

In Gaussian process regression, we observe a signal (dependent variable) $\mathbf{Y} = [\boldsymbol{y}_0, ..., \boldsymbol{y}_{N-1}]^T \in \mathbb{R}^{N \times P}$ (N data points with P features) that come with a set

of independent variables $\mathbf{t} = [\boldsymbol{t}_0, ..., \boldsymbol{t}_{N-1}]^N \in \mathbb{R}^{N\times Q}$. They are assumed to be related by:

$$\boldsymbol{y}_n = \boldsymbol{f}(\boldsymbol{t}_n) + \boldsymbol{\eta}_n \quad n = 0, ..., N-1 \tag{3.13}$$

where each component of $\boldsymbol{f}$, $f_p \sim \mathrm{GP}(m_p(\cdot), k_p(\cdot,\cdot)), p = 0, ..., P-1$ is a Gaussian process and $\boldsymbol{\eta}_n$ are Multivariate Gaussian random variables with mean zero and diagonal covariance matrix $\beta^{-1}\mathbf{I}_N$ (zero mean Gaussian noise). Such a setting is called a multiple-output Gaussian process. Each of the component of $\boldsymbol{f}$ are assumed to be independent. Hence, the full density is given by

$$f(\mathbf{Y}) = \prod_{p=0}^{P-1} \frac{1}{\sqrt{|2\pi(\mathbf{K}_p + \sigma^2\mathbf{I}_N)|}} \exp\left(-\frac{1}{2}(\mathbf{Y}_{:,p} - \boldsymbol{\mu}_p)^T(\mathbf{K}_p + \sigma^2\mathbf{I}_N)^{-1}(\mathbf{Y}_{:,p} - \boldsymbol{\mu}_p)\right) \tag{3.14}$$

where $\mathbf{Y}_{:,p}$, the p-th column of $\mathbf{Y}$ has components that represents the value of the p-th feature for each data point; $\mathbf{K}_p \in \mathbb{R}^{N\times N}$ has entries given by $k_p(\boldsymbol{t}_i, \boldsymbol{t}_j)$ for $i, j = 0, ..., N-1$; and $\boldsymbol{\mu}_p$ has components given by $[m_p(\boldsymbol{t}_0), ..., m_p(\boldsymbol{t}_{N-1})]$. Covariance kernels characterise the similarity structure of the data set and hence they are often a function of distance. In such cases, they are called isotropic.

Definition 3.3. *Let $k : \mathbb{R}^Q \times \mathbb{R}^Q \to \mathbb{R}$ be a positive definite kernel. We say that k is a* **stationary** *kernel if $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^Q$ we can write $k(\boldsymbol{x}, \boldsymbol{y}) = k_S(\boldsymbol{x} - \boldsymbol{y})$ for some function $k_S : \mathbb{R}^Q \to \mathbb{R}$ We further call a stationary kernel isotropic if we can write $k_S(\boldsymbol{x} - \boldsymbol{y}) = k_I(\|\boldsymbol{x} - \boldsymbol{y}\|)$ for some euclidean norm $\|\cdot\|$ and some function $k_I : \mathbb{R}_{\geq 0} \to \mathbb{R}$.*

Predicting an unseen $\boldsymbol{y}^*$ for an unseen value of $\boldsymbol{t}^*$ is called krigging. This involves finding the posterior mean and variance for that value of $\boldsymbol{t}^*$. For one component y_p^* of the feature vector $\boldsymbol{y}^*$, these are given by:

$$\begin{aligned} \mu_p^* &= k_p(\mathbf{t}, \boldsymbol{t}^*)^T(\mathbf{K}_p + \beta^{-1}\mathbf{I}_N)^{-1}\mathbf{Y}_{p,:} \\ k_p^*(\boldsymbol{t}^*, \boldsymbol{t}^*) &= k_p(\boldsymbol{t}^*, \boldsymbol{t}^*) - k_p(\mathbf{t}, \boldsymbol{t}^*)^T(\mathbf{K}_p + \beta^{-1}\mathbf{I}_N)^{-1}k_p(\mathbf{t}, \boldsymbol{t}^*). \end{aligned} \tag{3.15}$$

where we write $k_p(\mathbf{t}, \boldsymbol{t}^*) = [k_p(\boldsymbol{t}_0, \boldsymbol{t}^*), ..., k_p(\boldsymbol{t}_{N-1}, \boldsymbol{t}^*)]$. Then we have that $y_p^*|\mathbf{Y} \sim \mathcal{N}(\mu_p^*, k_p^*(\boldsymbol{t}^*, \boldsymbol{t}^*))$ is a univariate Gaussian random Variable.

3.3.3 GPLVM

Gaussian Process Latent Variable (GPLVM) models are a form of dimensionality reduction technique that make the same assumption as in equation 3.13. The two differences are $Q << P$ and we only have an observation of $\mathbf{Y}$. We wish to then find the latent space position matrix $\mathbf{t}$.

The data we observe is now thought of as realisations of a multiple-output Gaussian process $\mathbf{f}$ with a covariance operator $k(\boldsymbol{t}_i, \boldsymbol{t}_j)$ that is corrupted by noise with covariance operator $\beta^{-1}\delta_i^j$. The latent space $\mathbb{R}^Q$ is now the indexing set of the Gaussian process. Previously, we had a different covariance kernel for each component of the multiple-output Gaussian Process. In the case of the GPLVMs used in this paper, however, we will have the same covariance structure across the outputs of the Gaussian Process. One such kernel is the Automatic Relevance Determination (ARD) kernel given by:

$$k(\boldsymbol{t}, \boldsymbol{t}') = \sigma^2 \exp\left(-\frac{1}{2}\sum_{q=0}^{Q-1} \alpha_q (t_q - t'_q)^2\right). \tag{3.16}$$

$\sigma, \alpha_0, ..., \alpha_{Q-1}$ are hyper-parameters that need to be identified. Note that we assumed that the covariance operator is the same across the outputs of $\boldsymbol{f}$ which allows us to write

$$\begin{aligned} \boldsymbol{\mu}_{\text{post}}(\boldsymbol{t}^*) &= k(\mathbf{t}, \boldsymbol{t}^*)^T (\mathbf{K} + \beta^{-1}\mathbf{I}_N)^{-1}\mathbf{Y} \in \mathbb{R}^P \\ K_{\text{post}}(\boldsymbol{t}^*, \boldsymbol{t}^*) &= k(\boldsymbol{t}^*, \boldsymbol{t}^*) - k(\mathbf{t}, \boldsymbol{t}^*)^T (\mathbf{K} + \beta^{-1}\mathbf{I}_N)^{-1} k(\mathbf{t}, \boldsymbol{t}^*) \in \mathbb{R} \end{aligned} \tag{3.17}$$

for the distribution of a new indexing point $\boldsymbol{t}^*$ of the GPLVM. Using this equation, one can generate new instances of the data coming from indexing locations that correspond to none of the training data. Note that this distribution can be re-interpreted as white noise around a mean $\boldsymbol{\mu}_{\text{post}}$ as there is no correlation between the components of the features at location $\boldsymbol{t}^*$. A pictorial representation of a GPLVM is given in Figure 3-2.

Our aim is to optimise the likelihhood $l(\mathbf{Y}) := \prod_{p=0}^{P-1} f(\boldsymbol{y}_p|\mathbf{t}, \beta)$ by finding the optimal value of the latent variable for each data observation. This is done by finding $\arg\max_{\mathbf{t},\theta} log f(\mathbf{Y})$ where f is given in equation (3.14) and θ are the hyper parameters of the covariance kernels. For the ARD kernel, these are the length scales α_q and the variance σ^2. Note that this yields the following objective function for a GPLVM with zero mean

$$L(\mathbf{t}, \theta) = \frac{PN}{2}\log(2\pi) + \frac{P}{2}\log|\mathbf{K}| + \frac{1}{2}\text{tr}(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T). \tag{3.18}$$

A GPLVM tries to match the variance across the dataset by through the trace term. This matching is regularised by the determinant of the covariance matrix which prevents it from being too specific to the data.

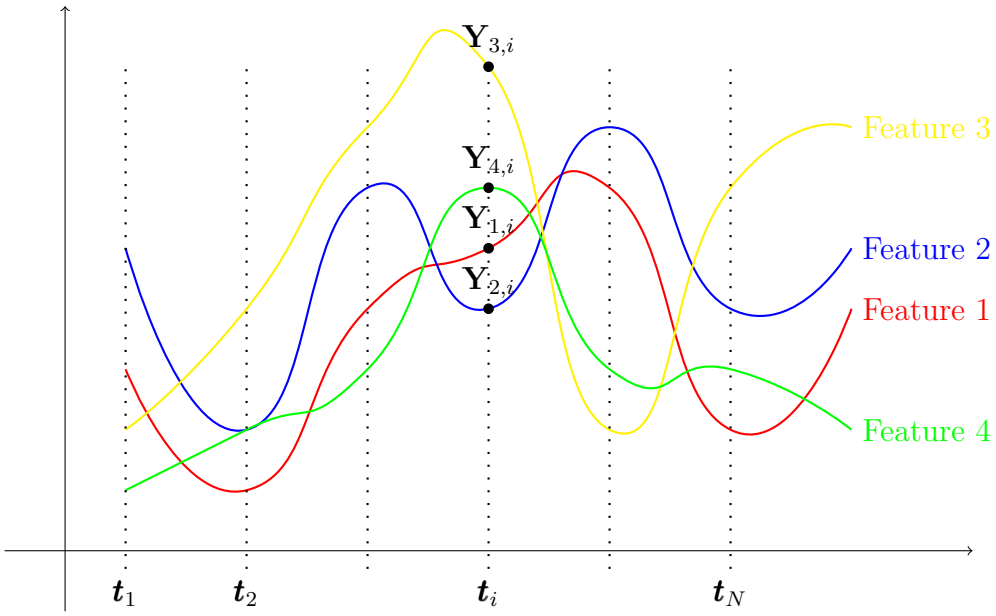


Figure 3-2: The figure shows a GPLVM. The coloured line represent one Gaussian process. In this case any one of those lines is a feature of the data $\mathbf{Y}$ that we observe ($P = 4$). Hence we see N of those features which occur at the corresponding latent space variable $\boldsymbol{t}_n$. What we have are P Gaussian processes that have indexing set $\boldsymbol{t}_n$ (the latent space). Each coloured curve represents one feature for the N data point and each vertical line represents one data point with P features.

3.3.4 Probabilistic PCA and dual PPCA

We describe a dual formulation of GPLVMs that use the formulation given by Lawrence [2005]. We assume that data is the result of a linear transformation on some latent space as follows:

$$\mathbf{Y}^T = \mathbf{W}\mathbf{t}^T + \eta \tag{3.19}$$

where $\mathbf{W} \in \mathbb{R}^{P\times Q}$ is a linear map from $\mathbb{R}^Q$ to $\mathbb{R}^P$; $\eta \in \mathbb{R}^{N\times P}$ is the observation error with $\eta_{i,j}$ being i.i.d Gaussians centred at zero and with variance β^{-1}; $\mathbf{Y} \in \mathbb{R}^{N\times P}$ is the observed data; and $\mathbf{t} \in \mathbb{R}^{N\times P}$ is are the latent variables. This induces the DAG in Figure 3-3.

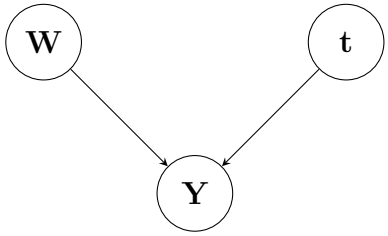


Figure 3-3: The DAG shows the linear latent space noise model. Marginalising over the latent space $\mathbf{t}$ yields PPCA while marginalising over the linear map $\mathbf{W}$ yields a GPLVM.

In such a latent space model, one can either focus on modelling the linear map or the latent space. This is done by marginalising over one of the two and then looking at the resulting distribution. We state the following lemma which allows us to do this.

Lemma 3.4. *Consider*

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$$

where $\boldsymbol{y} \in \mathbb{R}^P, \boldsymbol{x} \in \mathbb{R}^Q$ *and* $\boldsymbol{W} \in \mathbb{R}^{P\times Q}$. *Let* $f(\boldsymbol{y}|\,\boldsymbol{W}, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}|\,\boldsymbol{W}\boldsymbol{x}, \beta \boldsymbol{I}_P)$ *and* $f(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{0}, \alpha \boldsymbol{I}_Q)$, *then under the above model, we have that*

$$f(\boldsymbol{y}|\,\boldsymbol{W}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \alpha\, \boldsymbol{W}\boldsymbol{W}^T + \beta \boldsymbol{I}_P)$$

Proof. We have that

$$\begin{aligned} f(\boldsymbol{y}|\mathbf{W}) &= \int f(\boldsymbol{y}|\mathbf{W}, \boldsymbol{x}) f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ &\propto \int \exp\left(-\frac{1}{2\beta}(\boldsymbol{y} - \mathbf{W}\boldsymbol{x})^T(\boldsymbol{y} - \mathbf{W}\boldsymbol{x}) - \frac{1}{2\alpha}\boldsymbol{x}^T\boldsymbol{x}\right) \end{aligned} \tag{3.20}$$

Looking at the inner products,

$$
\begin{aligned}
&\frac{1}{\beta}(\boldsymbol{y}-\mathbf{W}\boldsymbol{x})^T(\boldsymbol{y}-\mathbf{W}\boldsymbol{x})+\frac{1}{\alpha}\boldsymbol{x}^T\boldsymbol{x}\\
=&\boldsymbol{x}^T\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)\boldsymbol{x}-2\frac{1}{\beta}\boldsymbol{y}^T\mathbf{W}\boldsymbol{x}+\frac{1}{\beta}\boldsymbol{y}^T\boldsymbol{y}\\
=&\boldsymbol{x}^T\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)\boldsymbol{x}\\
&-2\boldsymbol{x}^T\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)^{-1}\mathbf{W}^T\frac{\boldsymbol{y}}{\beta}\\
&+\frac{1}{\beta^2}\boldsymbol{y}^T\mathbf{W}\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)^{-1}\mathbf{W}^T\boldsymbol{y}\\
&+\frac{1}{\beta}\boldsymbol{y}^T\boldsymbol{y}-\frac{1}{\beta^2}\boldsymbol{y}^T\mathbf{W}\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)^{-1}\mathbf{W}^T\boldsymbol{y}\\
=&\left(\boldsymbol{x}-\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)^{-1}\mathbf{W}^T\frac{\boldsymbol{y}}{\beta}\right)^T\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)\\
&\times\left(\boldsymbol{x}-\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)^{-1}\mathbf{W}^T\frac{\boldsymbol{y}}{\beta}\right)\\
&+\boldsymbol{y}^T\left(\frac{1}{\beta}\mathbf{I}_P-\frac{1}{\beta^2}\mathbf{W}\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)^{-1}\mathbf{W}^T\right)\boldsymbol{y}
\end{aligned}
\tag{3.21}
$$

The first quadratic form gets integrated out in equation (3.20). We note that by using the Sherman-Woondbury matrix inversion formula

$$
(\beta\mathbf{I}_P+\mathbf{W}(\alpha\mathbf{I}_Q)\mathbf{W}^T)^{-1}=\frac{1}{\beta}\mathbf{I}_P-\frac{1}{\beta^2}\mathbf{W}\left(\frac{1}{\beta}\mathbf{W}^T\mathbf{W}+\frac{1}{\alpha}\mathbf{I}_Q\right)^{-1}\mathbf{W}^T \tag{3.22}
$$

Hence giving the result. □

3.3.4.1 Latent space marginalisation

The following proposition gives the distribution of the data when the latent space variables have been marginalised.

Proposition 3.5. *Let $\boldsymbol{t}_{n,:}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}_Q)$, then under the assumptions of equation (3.19), after marginalising $\boldsymbol{t}$, the n-th row of $\boldsymbol{Y}$ follows a multivariate normal given by $f(\boldsymbol{z}|\,\boldsymbol{W})=\mathcal{N}(\boldsymbol{z}|\mathbf{0},\,\boldsymbol{W}\boldsymbol{W}^T+\beta^{-1}\boldsymbol{I})$*

Proof. The $n-$th row of $\mathbf{Y}$ is given by $\boldsymbol{y}=\mathbf{W}\boldsymbol{x}$ where $\boldsymbol{x}=\mathbf{t}_{n,:}^T$. We also have that $f(\boldsymbol{y}|\mathbf{W},\boldsymbol{x})=\mathcal{N}(\boldsymbol{y}|\mathbf{W}\boldsymbol{x},\beta\mathbf{I}_P)$. Applying lemma 3.20 completes the proof. □

Proposition 3.5 yields the following negative log-likelihood

$$\mathcal{L}_w(\mathbf{Y}) = \frac{NP}{2}(\log(2\pi|\beta\mathbf{I}_P + \mathbf{W}\mathbf{W}^T|) + \sum_{n=0}^{N-1} \mathbf{Y}_{n,:}^T(\beta\mathbf{I}_P + \mathbf{W}\mathbf{W}^T)^{-1}\mathbf{Y}_{n,:} \quad (3.23)$$

Fitting the model would amount to maximising the above with respect to the linear map. Tipping and Bishop [1999] show that the above is maximised when

$$\mathbf{W} = \mathbf{U}_Q(\Sigma_Q - \beta\mathbf{I}_P)^{\frac{1}{2}} \quad (3.24)$$

where

$$\mathbf{U}_Q = \begin{bmatrix} | & | & .... & | \\ \boldsymbol{u}_0 & \boldsymbol{u}_1 & .... & \boldsymbol{u}_{Q-1} \\ | & | & .... & | \end{bmatrix}, \quad \Sigma_Q = \begin{bmatrix} \sigma_0 & 0 & 0 & .... & 0 \\ 0 & \sigma_1 & 0 & .... & 0 \\ . & . & . & & \\ . & . & & . & \\ 0 & 0 & 0 & & \sigma_{Q-1} \end{bmatrix}$$

Here $\sigma_q, \boldsymbol{u}_q$ form the eigenvalue pair of $\frac{1}{P}\mathbf{Y}^T\mathbf{Y}$ with $\sigma_q \geq \sigma_{q+1}$.

3.3.4.2 Linear map marginalisation

The following proposition gives the distribution of the data when the linear map has been marginalised.

Proposition 3.6. *Let $\boldsymbol{W}_{p,:} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_Q)$ then the assumptions of equation (3.19), after marginalising $\boldsymbol{W}$ the p-th column of $\boldsymbol{Y}$ follows a multivariate normal given by $f(\boldsymbol{z}|\boldsymbol{t}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \boldsymbol{t}\boldsymbol{t}^T + \beta^{-1}\boldsymbol{I}_N)$.*

Proof. Let $\boldsymbol{w}_p$ be the p-th row of $\mathbf{W}$ expressed as a column vector, then we have that the p-th column of $\mathbf{Y}$ is given by $\boldsymbol{y} = \mathbf{t}\boldsymbol{w}_p$. We then have that $f(\boldsymbol{y}|\mathbf{t}, \boldsymbol{w}_p) = \mathcal{N}(\boldsymbol{y}|\mathbf{t}\boldsymbol{w}_p, \beta\mathbf{I}_N)$. Applying lemma 3.20 completes the proof. □

Proposition 3.6 yields the following negative log-likelihood

$$\mathcal{L}_t(\mathbf{Y}) = \frac{NP}{2}(\log(2\pi|\beta\mathbf{I}_N + \mathbf{t}\mathbf{t}^T|) + \sum_{p=0}^{P-1} \mathbf{Y}_{:,p}^T(\beta\mathbf{I}_N + \mathbf{t}\mathbf{t}^T)^{-1}\mathbf{Y}_{:,p} \quad (3.25)$$

Fitting the model would amount to maximising the above with respect to the latent space variables. Lawrence [2005] show that the above is maximised when

$$\mathbf{t} = \mathbf{V}_Q(\Lambda_Q - \beta\mathbf{I}_N)^{\frac{1}{2}} \quad (3.26)$$

where

$$\mathbf{U}_Q = \begin{bmatrix} | & | & .... & | \\ \boldsymbol{v}_0 & \boldsymbol{v}_1 & .... & \boldsymbol{v}_{Q-1} \\ | & | & .... & | \end{bmatrix}, \quad \Sigma_Q = \begin{bmatrix} \lambda_0 & 0 & 0 & .... & 0 \\ 0 & \lambda_1 & 0 & .... & 0 \\ . & . & . & & \\ . & . & & . & \\ 0 & 0 & 0 & & \lambda_Q \end{bmatrix}$$

Here $\lambda_q, \boldsymbol{v}_q$ form the eigenvalue pair of $\frac{1}{N}\mathbf{Y}\mathbf{Y}^T$ with $\lambda_q \geq \lambda_{q+1}$.

3.3.4.3 Discussion

Propositions 3.5 and 3.6 provide a dual treatment of the same model. Equation (3.24) treats one data sample $\mathbf{Y}_{n,:} \in \mathbb{R}^P$ as a multivariate Gaussian. Its covariance structure is modelled after the inner product $\mathbf{Y}^T\mathbf{Y}$ of the data and the dimensionality reduction comes from the principal subspace of the data as shown in equation (3.24). This type of model is called probabilistic principal component analysis (PPCA). By sending the variance β to zero, we note that we recover the traditional PCA solution in equation (3.24).

Equation (3.26) on the other hand treats one feature $\mathbf{Y}_{:,p} \in \mathbb{R}^N$ as a multivariate Gaussian. Its covariance structure is modelled after the outer product $\mathbf{Y}\mathbf{Y}^T$ of the data and the dimensionality reduction comes from the principal subspace of the features as shown in equation (3.26).

Moreover, marginalising out the map allows us to recover the latent space position of each data point. Note that equation (3.26) is the likelihood of a multi output Gaussian process where the covariance structure for each component f_p is given by $K_p(\boldsymbol{t}, \boldsymbol{s}) = \boldsymbol{t}^T\boldsymbol{s}$ and where one has a diagonal i.i.d Gaussian noise model with variance β. We can extend this model to allow $K_p(\boldsymbol{t}, \boldsymbol{s})$ to be a general covariance kernel. This yields a Gaussian Process Latent Variable model.

3.3.5 Back Constrained GPLVM

The geometry of the latent space is guided by how well the latent point covariance matrix $\mathbf{K} + \beta^{-1}\mathbf{I}$ matches the data covariance $\mathbf{Y}\mathbf{Y}^T$ as shown in equation (3.18). Assuming that we are using an isotropic kernel, the data examples that are close together in data space will have latent point positions that are also close together. However, it is not guaranteed that latent point positions that are close

together will generate data examples that are also close to each other. Lawrence and Quiñonero-Candela [2006] propose to modify the objective function in equation (3.18). The new model is called a Back-Constraind GPLVM.

Let ϕ be an isotropic kernel and let $\mathbf{v} \in \mathbb{Q} \times \mathbb{N}$. We replace the latent space position by one that is forced to be close to each other via the following kernel projection

$$\boldsymbol{s}_n = \left(\sum_{j=0}^{N-1} v_{0,j} \phi(\boldsymbol{t}_m, \boldsymbol{t}_j), ..., \sum_{j=0}^{N-1} v_{Q-1,j} \phi(\boldsymbol{t}_m, \boldsymbol{t}_j) \right) \tag{3.27}$$

and instead maximise

$$L(\mathbf{t}, \theta, \mathbf{v}) = \frac{PN}{2} \log(2\pi) + \frac{P}{2} \log |\mathbf{K}(\mathbf{s}, \mathbf{s})| + \frac{1}{2} \mathrm{tr}(\mathbf{K}(\mathbf{s}, \mathbf{s})^{-1} \mathbf{Y}\mathbf{Y}^T). \tag{3.28}$$

with respect to $\mathbf{t}, \theta, \mathbf{v}$. This regularises the objective function by introducing a local distrance preserving term. This is referred to as a back constraint by Lawrence and Quiñonero-Candela [2006].

3.3.6 Variational GPLVM

We have shown in the section 3.3.4.2 that a GPLVM can be thought of a latent noise model where the mapping has been marginalised over. We now want to marginalise over the latent space variables of the GPLVM.

Let $\mathbf{Y} \in \mathbb{R}^{N \times P}$ be data coming from a noise corrupted zero mean GPLVM with covariance operator $K : \mathbb{R}^Q \times \mathbb{R}^Q \to \mathbb{R}$.

$$\mathbf{Y} = \mathbf{F} + \eta \tag{3.29}$$

where $\eta \in \mathbb{R}^{N \times P}$ is the observation error with $\eta_{i,j}$ being i.i.d Gaussians centred at zero and with variance β^{-1}; and $\mathbf{F} \in \mathbb{R}^{N \times P}$ are realisation from a multi-output Gaussian Process mapping the latent space $\mathbf{t} \in \mathbb{R}^{N \times Q}$ onto the data space.

Our aim is to perform the following marginalisation

$$p(\mathbf{Y}) = \int p(\mathbf{Y}|\mathbf{t}) p(\mathbf{t}) \mathrm{d}\mathbf{t} \tag{3.30}$$

with the following priors and variational distribution on the latent space:

$$
\begin{aligned}
p(\mathbf{t}) &= \prod_{n=0}^{N-1} \mathcal{N}\left(\boldsymbol{t}_n | \mathbf{0}, \mathbf{I}_Q\right) \\
\mathcal{Q}(\mathbf{t}) &= \prod_{n=0}^{N-1} \mathcal{N}\left(\boldsymbol{t}_n | \boldsymbol{\mu}_n, \Sigma_n\right)
\end{aligned}
\tag{3.31}
$$

That is, the rows of $\mathbf{t}$ are assumed to be independent. Using Jensen's inequality, this becomes:

$$
\begin{aligned}
\mathcal{L}(\mathcal{Q}) &= \int \log\left(p(\mathbf{Y}|\mathbf{t})\right) \mathcal{Q}(\mathbf{t}) \mathrm{d}\mathbf{t} - \int \log\left(\frac{\mathcal{Q}(\mathbf{t})}{p(\mathbf{t})}\right) \mathcal{Q}(\mathbf{t}) \mathrm{d}\mathbf{t} \\
&= \sum_{p=0}^{P-1} \int \log\left(p(\mathbf{Y}_{:,p}|\mathbf{t})\right) \mathcal{Q}(\mathbf{t}) \mathrm{d}\mathbf{t} - \int \log\left(\frac{\mathcal{Q}(\mathbf{t})}{p(\mathbf{t})}\right) \mathcal{Q}(\mathbf{t}) \mathrm{d}\mathbf{t}
\end{aligned}
\tag{3.32}
$$

This integral consists of two parts

1.

$$
\begin{aligned}
\mathcal{I}_1 &= \int \log\left(\frac{\mathcal{Q}(\mathbf{t})}{p(\mathbf{t})}\right) \mathcal{Q}(\mathbf{t}) \mathrm{d}\mathbf{t} \\
&= \prod_{n=0}^{N-1} \int \log\left(\frac{\mathcal{N}\left(\boldsymbol{t}_n | \boldsymbol{\mu}_n, \Sigma_n\right)}{\mathcal{N}\left(\boldsymbol{t}_n | \mathbf{0}, \mathbf{I}_Q\right)}\right) \mathcal{N}\left(\boldsymbol{t}_n | \boldsymbol{\mu}_n, \Sigma_n\right) \mathrm{d}\boldsymbol{t}_n \\
&= \prod_{n=0}^{N-1} \left[-\frac{Q}{2} - \frac{1}{2}\left(\mu_n^T \mu_n + \mathrm{tr}(\Sigma_n)\right) - \frac{Q}{2} \log|\Sigma_n|\right]
\end{aligned}
\tag{3.33}
$$

which we compute in section 3.3.7. Note that this is the K-L divergence from $\mathcal{Q}(\mathbf{t})$ to $p(\mathbf{t})$, which we denote by $\mathrm{D}_{\mathrm{KL}}(Q_t||p_t)$

2.

$$
\begin{aligned}
\mathcal{I}_2 &= \sum_{p=0}^{P-1} \int \log\left(p(\mathbf{Y}_{:,p}|\mathbf{t})\right) \mathcal{Q}(\mathbf{t}) \mathrm{d}\mathbf{t} \\
&\geq \sum_{p=0}^{P-1} \log\left[\int \exp\left(\mathbb{E}_{\mathcal{Q}(\mathbf{t})}\left[\log \mathcal{N}\left(\mathbf{Y}_{:,p} | \boldsymbol{\mu}_{MN}, \beta^{-1}\mathbf{I}_N\right)\right]\right) p(\mathbf{Z}_{:,p}) \mathrm{d}\mathbf{Z}_{:,p}\right] \\
&- P \times \frac{\beta}{2} \mathbb{E}_{\mathcal{Q}(\mathbf{t})}\left[\mathrm{tr}\left(\mathbf{K}_{NN} - \mathbf{Q}_{NN}\right)\right]
\end{aligned}
\tag{3.34}
$$

which we caculate and describe in section 3.3.8. Here, we have introduced inducing points $\mathbf{Z}$, which are draws from the same distribution as $\mathbf{F}$.

3.3.7 Computing $\mathcal{I}_1$

Using the distributions in equation (3.31) and setting $\boldsymbol{t}_n = \boldsymbol{t}$, $\boldsymbol{\mu}_n = \boldsymbol{\mu}$ and $\Sigma_n = \Sigma$ for ease of notation, we get that

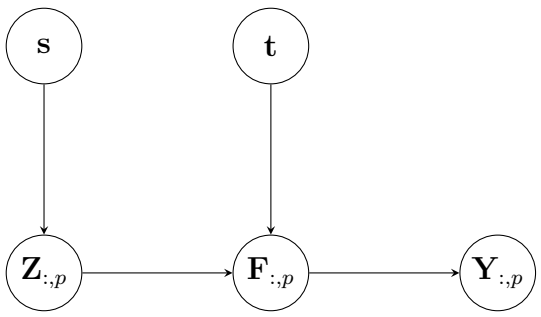
$$\begin{aligned}
&\int \log\left(\frac{\mathcal{N}(\boldsymbol{t}|\boldsymbol{\mu},\Sigma)}{\mathcal{N}(\boldsymbol{t}|\mathbf{0},\mathbf{I}_Q)}\right)\mathcal{N}(\boldsymbol{t}|\boldsymbol{\mu},\Sigma)\,\mathrm{d}\boldsymbol{t} \\
&= \int\left[-\frac{1}{2}(\boldsymbol{t}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{t}-\boldsymbol{\mu}) - \frac{1}{2}\boldsymbol{t}^T\boldsymbol{t} - \frac{Q}{2}\log|\Sigma|\right]\mathcal{N}(\boldsymbol{t}|\boldsymbol{\mu},\Sigma)\,\mathrm{d}\boldsymbol{t} \\
&= -\frac{1}{2}\mathrm{tr}(\mathbf{I}_Q) - \frac{1}{2}\left(\mu^T\mu + \mathrm{tr}(\Sigma)\right) - \frac{Q}{2}\log|\Sigma|
\end{aligned} \tag{3.35}$$

3.3.8 Computing $\mathcal{I}_2$

Recall that for a multi output Gaussian Process that is corrupted by noise, the likelihood is given by $p(\boldsymbol{y}|\boldsymbol{t}) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{K}_p + \beta^{-1}\mathbf{I}_N)$. Hence we have that

$$\mathcal{I}_2 = -\sum_{p=0}^{P-1}\mathbb{E}_{\mathcal{Q}(t)}\left[\frac{1}{2}\mathbf{Y}_{:,p}^T(\mathbf{K}_p + \beta^{-1}\mathbf{I}_N)^{-1}\mathbf{Y}_{:,p} + \log\left(\sqrt{|(2\pi(\mathbf{K}_p + \beta^{-1}\mathbf{I}_N)|}\right)\right]. \tag{3.36}$$

with $\mathbb{E}_{\mathcal{Q}(t)}$ denoting an expectation with respect to the variational distribution on $\mathbf{t}$ and $\mathbf{Y}_{:,p}$ denoting the p-th column of the data. $\mathbf{t}$ appears in the inverse of the covariance martix and hence the first integral is hard to compute. Titsias and Lawrence [2010] use the method described by Titsias [2009] where a set of M inducing variables $\mathbf{Z}$ is used to facilitate the computation of equation (3.32). $\mathbf{Z} \in \mathbb{R}^{M\times P}$ is assumed to be realisations of the same Gaussian process as the GPLVM at latent point locations $\mathbf{s} \in \mathbb{R}^{M\times Q}$. The DAG below introduces an augmented model where the values of the $\mathbf{Y}$ depend on the values of the inducing points.



For ease of notation let $\boldsymbol{z} = \mathbf{Z}_{:,p}$, $\boldsymbol{f} = \mathbf{F}_{:,p}$ and $\boldsymbol{y} = \mathbf{Y}_{:,p}$. We approximate $p(\boldsymbol{y}|\mathbf{t},\mathbf{s})$ with $p(\boldsymbol{y}|\mathbf{t})$. Under the above DAG, we have that $p(\boldsymbol{y}|\mathbf{t},\mathbf{s}) = \int\int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{z},\mathbf{t})p(\boldsymbol{z}|\mathbf{s})\mathrm{d}\boldsymbol{z}\mathrm{d}\boldsymbol{f}$. Omiting the dependence on the latent space pa-

rameters, and by using the variational distribution $\mathcal{Q}(\boldsymbol{z}, \boldsymbol{f}) = p(\boldsymbol{f}|\boldsymbol{z})\phi(\boldsymbol{z})$ on $\boldsymbol{z}$ and $\boldsymbol{f}$, we have that:

$$\begin{aligned}
&\log p(\boldsymbol{y}) \\
=&\log \int\int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{z})p(\boldsymbol{z})\mathrm{d}\boldsymbol{z}\mathrm{d}\boldsymbol{f} \\
\geq&\int\int \log\left(\frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{f}|\boldsymbol{z})\phi(\boldsymbol{z})}\right) p(\boldsymbol{f}|\boldsymbol{z})\phi(\boldsymbol{z})\mathrm{d}\boldsymbol{z}\mathrm{d}\boldsymbol{f} \\
=&\int \phi(\boldsymbol{z})\left[\int \log\left(p(\boldsymbol{y}|\boldsymbol{f})\right) p(\boldsymbol{f}|\boldsymbol{z})\mathrm{d}\boldsymbol{f} + \int \log\left(\frac{p(\boldsymbol{z})}{\phi(\boldsymbol{z})}\right) p(\boldsymbol{f}|\boldsymbol{z})\mathrm{d}\boldsymbol{f}\right]\mathrm{d}\boldsymbol{z} \\
=&\int \phi(\boldsymbol{z})\left[\int \log\left(p(\boldsymbol{y}|\boldsymbol{f})\right) p(\boldsymbol{f}|\boldsymbol{z})\mathrm{d}\boldsymbol{f} + \log\left(\frac{p(\boldsymbol{z})}{\phi(\boldsymbol{z})}\right)\right]\mathrm{d}\boldsymbol{z}
\end{aligned} \tag{3.37}$$

Note that, for the first integral, we have that $p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \beta^{-1}\mathbf{I})$ and $p(\boldsymbol{f}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}_{MN}, \mathbf{K}_{NN} - \mathbf{Q}_{NN})$. The second distribution is a posterior Gaussian where

- $\mathbf{K}_{NN} = K(\mathbf{t}, \mathbf{t})$
- $\mathbf{K}_{MM} = K(\mathbf{s}, \mathbf{s})$
- $\mathbf{K}_{NM}^T = \mathbf{K}_{MN} = K(\mathbf{s}, \mathbf{t})$
- $\mathbf{Q}_{NN} = \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}$
- and $\boldsymbol{\mu}_{MN} = \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\boldsymbol{z}$.

We now perform the inner integration:

$$\begin{aligned}
&\int \log\left(p(\boldsymbol{y}|\boldsymbol{f})\right) p(\boldsymbol{f}|\boldsymbol{z})\mathrm{d}\boldsymbol{f} \\
=&\int \left[-\frac{\beta}{2}(\boldsymbol{y}-\boldsymbol{f})^T(\boldsymbol{y}-\boldsymbol{f}) + \frac{N}{2}\log\left(\frac{\beta}{2\pi}\right)\right]\mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}_{MN}, \mathbf{K}_{NN} - \mathbf{Q}_{NN})\mathrm{d}\boldsymbol{f} \\
=&\int -\frac{\beta}{2}\left[\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{f}^T\boldsymbol{y} + \boldsymbol{f}^T\boldsymbol{f}\right]\mathcal{N}(\boldsymbol{f}|\mu_{MN}, \mathbf{K}_{NN} - \mathbf{Q}_{NN})\mathrm{d}\boldsymbol{f} + \frac{N}{2}\log\left(\frac{\beta}{2\pi}\right) \\
=&-\frac{\beta}{2}\left[\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{\mu}_{MN}^T\boldsymbol{y} + \boldsymbol{\mu}_{MN}^T\boldsymbol{\mu}_{MN} + \mathrm{tr}(\mathbf{K}_{NN} - \mathbf{Q}_{NN})\right] + \frac{N}{2}\log\left(\frac{\beta}{2\pi}\right) \\
=&\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{MN}, \beta^{-1}\mathbf{I}_N\right) - \frac{\beta}{2}\mathrm{tr}(\mathbf{K}_{NN} - \mathbf{Q}_{NN})
\end{aligned} \tag{3.38}$$

Substituting this in equation (3.39), we have that

$$
\begin{aligned}
&\log p(\boldsymbol{y}|\mathbf{t},\mathbf{s}) \\
&\geq \int \phi(\boldsymbol{z}) \left[\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{MN},\beta^{-1}\mathbf{I}_N\right) - \frac{\beta}{2}\mathrm{tr}(\mathbf{K}_{NN}-\mathbf{Q}_{NN}) + \log\left(\frac{p(\boldsymbol{z})}{\phi(\boldsymbol{z})}\right)\right] \mathrm{d}\boldsymbol{z} \\
&= \int \phi(\boldsymbol{z}) \left[\log\left(\frac{\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{MN},\beta^{-1}\mathbf{I}_N\right)p(\boldsymbol{z})}{\phi(\boldsymbol{z})}\right)\right] \mathrm{d}\boldsymbol{z} - \frac{\beta}{2}\mathrm{tr}(\mathbf{K}_{NN}-\mathbf{Q}_{NN})
\end{aligned}
\tag{3.39}
$$

Putting this inside the integral with respect to $\boldsymbol{t}$

$$
\begin{aligned}
&\int \log\left(p(\boldsymbol{y}|\mathbf{t})\right)\mathcal{Q}(\mathbf{t})\mathrm{d}\mathbf{t} \\
&= \int \log\left(\int\int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{z},\boldsymbol{t},\mathbf{s})p(\boldsymbol{z}|\mathbf{s})\mathrm{d}\boldsymbol{z}\mathrm{d}\boldsymbol{f}\right)\mathcal{Q}(\mathbf{t})\mathrm{d}\mathbf{t} \\
&\geq \int \left(\int \phi(\boldsymbol{z})\left[\log\left(\frac{\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{MN},\beta^{-1}\mathbf{I}_N\right)p(\boldsymbol{z})}{\phi(\boldsymbol{z})}\right)\right]\mathrm{d}\boldsymbol{z} - \frac{\beta}{2}\mathrm{tr}(\mathbf{K}_{NN}-\mathbf{Q}_{NN})\right)\mathcal{Q}(\mathbf{t})\mathrm{d}\mathbf{t} \\
&= \int \phi(\boldsymbol{z})\mathbb{E}_{\mathcal{Q}(\mathrm{t})}\left[\log\left(\frac{\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{MN},\beta^{-1}\mathbf{I}_N\right)p(\boldsymbol{z})}{\phi(\boldsymbol{z})}\right)\right]\mathrm{d}\boldsymbol{z} - \frac{\beta}{2}\mathbb{E}_{\mathcal{Q}(\mathrm{t})}\left[\mathrm{tr}(\mathbf{K}_{NN}-\mathbf{Q}_{NN})\right]
\end{aligned}
\tag{3.40}
$$

Using variational calculus, we find that $\phi(\boldsymbol{z})$ needs to be proportional to

$$
\mathbb{E}_{\mathcal{Q}(\mathrm{t})}\left[\log\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{MN},\beta^{-1}\mathbf{I}_N\right)\right]p(\boldsymbol{z}).
$$

We also want $\phi(\boldsymbol{z})$ to be a probability distribution, and hence we can set it to be $p(\boldsymbol{z})$. By reversing the jensen inequality we get that:

$$
\begin{aligned}
&\int \phi(\boldsymbol{z})\mathbb{E}_{\mathcal{Q}(\mathrm{t})}\left[\log\left(\frac{\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{MN},\beta^{-1}\mathbf{I}_N\right)p(\boldsymbol{z})}{\phi(\boldsymbol{z})}\right)\right]\mathrm{d}\boldsymbol{z} \\
&\leq \log\left(\int \exp\left\{\mathbb{E}_{\mathcal{Q}(\mathrm{t})}\left[\log\left(\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{MN},\beta^{-1}\mathbf{I}_N\right)p(\boldsymbol{z})\right)\right]\right\}p(\boldsymbol{z})\mathrm{d}\boldsymbol{z}\right)
\end{aligned}
\tag{3.41}
$$

3.3.9 Final Bound

The final lower bound for $p(\mathbf{Y})$ is given by

$$
\begin{aligned}
&\mathcal{L}(\mathcal{Q}) \\
&= \sum_{p=0}^{P-1}\log\left[\int \exp\left(\mathbb{E}_{\mathcal{Q}(\mathrm{t})}\left[\log\mathcal{N}\left(\mathbf{Y}_{:,p}|\boldsymbol{\mu}_{MN},\beta^{-1}\mathbf{I}_N\right)\right]\right)p(\mathbf{Z}_{:,p})\mathrm{d}\mathbf{Z}_{:,p}\right] \\
&- P\times\frac{\beta}{2}\mathbb{E}_{\mathcal{Q}(\mathrm{t})}\left[\mathrm{tr}\left(\mathbf{K}_{NN}-\mathbf{Q}_{NN}\right)\right] - \mathrm{D}_{\mathrm{KL}}(Q_t||p_t)
\end{aligned}
\tag{3.42}
$$

Note that now, $\mathbf{t}$ does not appear inside the inverse of a matrix inside the expectation with respect to $\mathcal{Q}(\mathbf{t})$. Instead, when using a squared exponential kernel such as

$$k(\boldsymbol{t}, \boldsymbol{t}') = \sigma^2 \exp\left(-\frac{1}{2}\sum_{q=0}^{Q-1} \alpha_q (t_q - t'_q)^2\right)$$

the expectation with respect to $\mathcal{Q}$ looks like the convolution of two Gaussians, which has a closed form expression. Moroever, the integration of the exponential with respect to $\mathbf{Z}_{:,p}$ is one that can be computed as it will take the form of inner products integrated against a Gaussian. The exact form of this integral is given in Titsias and Lawrence [2010].

3.3.10 Summary

We have thus found a lower bound for $\int p(\mathbf{Y}|\mathbf{t})p(\mathbf{t})\mathrm{d}\mathbf{t}$ by introducing new realisations of the Gaussian Process. Note that these realisations and their latent space positions $\mathbf{s}$ need to be found. This is done by optimising the lower bound with respect to these latent spoint parameters. The final expression will also be a function of the variational means $\boldsymbol{\mu}_n$ and Σ_n. They parametrise the distribution on $\mathbf{t}$ and act as a proxy for the point estimates on $\mathbf{t}$ made in the regular GPLVM.

It is worth mentioning that (3.42) is a lower bound on $p(\mathbf{Y}|\mathbf{s})$ as opposed to being lower bound for $p(\mathbf{Y})$. However, the literature treats $\mathbf{s}$ as hyperparameters that need to be learned. Strictly speaking, $\mathbf{s}$ are not random variables. They are parameters that allow us to approximate $\int \log(p(\mathbf{Y}|\mathbf{t}))\mathcal{Q}(\mathbf{t})\mathrm{d}\mathbf{t}$.

3.4 Radial Basis Function Interpolation

We use a Radial Basis Function (RBF) interpolant in Chapter 6 to turn an intractable integral into a tractable one. We first introduce RBF kernel interpolation and analyse the interpolation error. Using results from functional analysis and material from Fröhlich [2013], we then give error estimates for the approximation we make in the marginalisation in (6.7).

3.4.1 Setting

Consider a function $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ which we wish to approximate using Radial Basis Functions

$$\phi_l(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi v^2}} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{p}_l\|^2}{2v^2}\right) \tag{3.43}$$

We assume that we know the value of the function f over a set $\mathcal{X} := \{\boldsymbol{p}_0, ..., \boldsymbol{p}_{L-1} : \boldsymbol{p}_i \neq \boldsymbol{p}_j \forall i \neq j\}$. Usually, interpolating f using ϕ_l would require us to solve the system given by

$$\begin{bmatrix} \phi_0(\boldsymbol{p}_0) & . & . & . & \phi_{L-1}(\boldsymbol{p}_0) \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ \phi_0(\boldsymbol{p}_{L-1}) & . & . & . & \phi_{L-1}(\boldsymbol{p}_{L-1}) \end{bmatrix} \begin{bmatrix} w_0 \\ . \\ . \\ . \\ w_{L-1} \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{p}_0) \\ . \\ . \\ . \\ f(\boldsymbol{p}_{L-1}) \end{bmatrix} \tag{3.44}$$

The interpolant will then be given by

$$\mathcal{I}_f(\boldsymbol{x}) = \sum_{l=0}^{L-1} \frac{w_l}{\sqrt{2\pi v^2}} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{p}_l\|^2}{2v^2}\right) \tag{3.45}$$

3.4.2 Interpolation error estimate

In Interpolation Theory one is interested in controlling the maximum difference between the interpolant and the function f within a bounded domain Ω, i.e, one wants to bound $\max_{x\in\Omega} |f(\boldsymbol{x}) - \mathcal{I}_p(\boldsymbol{x})|$. We have the following Lemma from Fröhlich [2013]

Lemma 3.7. *Let Ω be a cube in $\mathbb{R}^d$ and let $\mathcal{I}_f$ be the radial basis function interpolant through a set $\mathcal{X} := \{\boldsymbol{p}_0, ..., \boldsymbol{p}_{L-1}\} \subset \Omega$ of a smooth, differentiable function*

f. Then we have that

$$\max_{x\in\Omega} |f(\boldsymbol{x}) - \mathcal{I}_p(\boldsymbol{x})| \leq \exp\left(-\frac{\log(h_{\Omega,\mathcal{X}})}{h_{\Omega,\mathcal{X}}}\right) \|f\|_{\phi,\Omega}$$

where

$$\|f\|^2_{\phi,\Omega} = \frac{1}{(\sqrt{2\pi})^d} \int_\Omega \frac{\hat{f}^2(\omega)}{\hat{\mathcal{I}}_f(\omega)} d\omega$$

is constant with Where $\hat{f}$ and $\hat{\mathcal{I}}_f$ denote the Fourier transform of f and $\mathcal{I}_f$ respectively, and

$$h_{\Omega,\mathcal{X}} := \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{p}\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{p}\|_2$$

Remark 3.8. *Fröhlich [2013] requires that the function to be interpolated lies in the so call native space of the interpolant. For our purposes, this requires for $f(\boldsymbol{x}) := p(\boldsymbol{M}|u)\Big|_{\boldsymbol{x}}$ to satisfy*

$$\int_\Omega \frac{\hat{f}^2(\omega)}{\hat{\mathcal{I}}_f(\omega)} d\omega \in \mathbb{R}$$

. We hence require that the Fourier transform of f decays at most as fast as that of a Gaussian. Let Ω be a square bounded domain on $\mathbb{R}^2$ and let $\Gamma \subset \mathbb{R}^2$ be the boundary of some open set in Ω. Define V_u to be given by

$$V_u(\boldsymbol{x}) = \min_{\boldsymbol{y}\in\Gamma} \|\boldsymbol{x} - \boldsymbol{y}\| \tag{3.46}$$

Then for

$$\begin{aligned} f :&\mathbb{R}^2 \to \mathbb{R} \\ &\boldsymbol{x} \mapsto \exp(-V_u(\boldsymbol{x})) \end{aligned} \tag{3.47}$$

We have that f behaves like a Gaussian at its tails. This ensures that the Fourier transform decays like a Gaussian at infinity, gauranteeing that f lies in the Native space of the RBF interpolant.

What interests us ins Lemma 3.7 is the behaviour of $h_{\Omega,\mathcal{X}}$.

Lemma 3.9. *There exists a sequence $\mathcal{X}_i$ s.t. $\lim_{i\to\infty} h_{\Omega,\mathcal{X}_i} = 0$*

Proof. Consider a general $\mathcal{X}$. We define the following subset of Ω

$$B_l(\mathcal{X}) = \{\boldsymbol{x} \in \Omega : \|\boldsymbol{x} - \boldsymbol{p}_l\|_2 = \min_{\boldsymbol{p}\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{p}\|_2\} \tag{3.48}$$

which contains all the elements of Ω which is in the maximum range of some $\boldsymbol{p}_l \in \mathcal{X}$. This often referred to as the Voronoi Cell associated with the site $\boldsymbol{p}_l$. We set

$$\|B_l(\mathcal{X})\| := \sup_{x \in B_l} \|\boldsymbol{p}_l - \boldsymbol{x}\| \tag{3.49}$$

Then we have that

$$h_{\Omega,\mathcal{X}} = \max_{l=0,..,L-1} \|B_l(\mathcal{X})\| \tag{3.50}$$

We define a neighbourhood structure on the elements of $\mathcal{X}$. Let $\boldsymbol{p}_l \in \mathcal{X}$. Then $\boldsymbol{p}_k \in \mathcal{X}$ is a neighbour of $\boldsymbol{p}_l$ if there exists a curve $\boldsymbol{m} : [0,1] \to \mathbb{R}^d$ with $\boldsymbol{m}(0) = \boldsymbol{p}_l$ and $\boldsymbol{m}(1) = \boldsymbol{p}_k$ s.t. $\boldsymbol{m} \cap B_l \cap B_k = \boldsymbol{m}$. Equivalently, $\boldsymbol{p}_l \sim \boldsymbol{p}_k$ if $B_k \cap B_l \neq \varnothing$. Let $d_i = \max_{p \sim q \in \mathcal{X}_i} \|\boldsymbol{p} - \boldsymbol{q}\|$. Consider now $\boldsymbol{p}_l \sim \boldsymbol{p}_k$. From the definition of a Voronoi cell we have that for $\boldsymbol{x} \in B_k(\mathcal{X}_i)$ and $\boldsymbol{y} \in B_k(\mathcal{X}_i) \cup \{\boldsymbol{p}_k\}$,

$$\begin{aligned} \|\boldsymbol{p}_k - \boldsymbol{x}\| &< \|\boldsymbol{p}_l - \boldsymbol{y}\| \\ &\text{setting } \boldsymbol{y} = \boldsymbol{p}_k \\ &< d_i \Rightarrow \\ \|B_k(\mathcal{X}_i)\| &< d_i \end{aligned} \tag{3.51}$$

For each such pair of neighbours, we choose a curve such that $\|\boldsymbol{m}\|_2 = \|\boldsymbol{p}_l - \boldsymbol{p}_k\|_2$ and pick a point halfway on the curve. We construct $\mathcal{X}_{i+1}$ to be the union of $\mathcal{X}_i$ and the points that we just created out of the neighbouring structure of $\mathcal{X}_i$.

As Ω is a box in $\mathbb{R}^d$, we can define $\mathcal{X}_0$ to be the set of points lying on the corners of Ω. Under the above construction rule, the new points are added on lines joining adjacent elements of $\mathcal{X}_0$, as illustrated below.

$\mathcal{X}_0$ $\mathcal{X}_1$

In fact, with $\mathcal{X}_0$ described above, $B_l(\mathcal{X}_i)$ is a square domain for every l and every i and each new point is added on the corners of $B_l(\mathcal{X}_i)$. For such a construction, we have that d_0 is the distance between diagonal neighbours. Moreover, for all $p \in \mathcal{X}_0$ we have that:

$$d_0 = \max_{q : q \sim p \in \mathcal{X}_0} \|\boldsymbol{p} - \boldsymbol{q}\|_2$$

As we are adding new points that lie halfway between neighbours we necessarily have that $d_{i+1} = \frac{1}{2}d_i$ yielding

$$\|B_l(\mathcal{X}_i)\| \leq \frac{1}{2^i}d_0 \quad \forall l \Rightarrow h_{\Omega,\mathcal{X}_i} \leq \frac{1}{2^i}d_0 \tag{3.52}$$

Hence we have created a sequence whose limit is zero as i goes to infinity.

□

Chapter 4

Mathematical representation of shape

4.1 Introduction

PsA damage causes the shape of bones to change through erosion, fusion and proliferation. It is thus important to identify a representation for bone shapes which allows us to detect these shape changes. In this chapter, we describe how we can use statistical models of shapes for this purpose. In particular, we show how one can build latent shape representations for shapes. These treat shapes as realisations from a random variable that is in turn the result of a hidden process.

Currently, the most popular latent space representation involves a linear map from the latent space to the curve space as originally introduced by Cootes et al. [1995]. We argue that a move towards more general mappings should be favoured as this captures a wider varieties of shapes. This can be easily done by instead using a latent space representation that is characterised by a GPLVM (Lawrence [2005]) such as the one used by Prisacariu and Reid [2011] to set a prior on the Mumford Shah model.

Section 4.2 is dedicated to introducing shapes as closed curves and defining a family of shapes in terms of deformations. We then describe in section 4.3 how by treating a discretised curve as a point cloud, one can use dimensionality reduction techniques to build a statistical model of a shape. In particular, we introduce and describe the PCA representation used in the Active Shape Model (ASM) by Cootes et al. [1995]. In section 4.3.2, we use the dual representation of PCA described in section 3.3.4 to extend this representation to include non-linear maps from the latent space by using a GPLVM. We then demonstrate the use of

GPLVMs as latent space shape representation models in section 4.4.

4.2 Shape Definition

Kendall [1984] describes shapes as being geometric information that is invariant to similarity transformations. Building on this notion, two objects have the same shape if they can be translated, rotated or scaled to match one another exactly. Hence, when speaking of a family of shapes, we wish to isolate the effect of such transformations from the definition.

If two distinct shapes come from the same family, then after undergoing an alignment process through some similarity tansform, they would still differ from each other. The size of this difference is what characterises the family of shapes. At this point, one shape can be mapped exactly onto the other by some non-linear transformation $\mathcal{T}$. We define a family of shapes as shapes that can be transformed onto each other through some small deformation $\mathcal{T}$, after having undergone some initial alignment. The range of shapes that is admissible within a family is then captured by some constraint on $\mathcal{T}$. This could, for example, be a constraint on the norm of $\mathcal{T}$.

One way to characterise this family of shapes is to model $\mathcal{T}$ as a deformation that is applied on $\mathbb{R}^2$. I.e one would induce a diffeomorphic flow that transforms space in a global way. Younes [2010] explores how diffeomorphisms can be used in image analysis. These diffeomorphisms can be built by inducing a time-dependent flow field. These methods are popular in the field of image registration, where one tries to align one image onto another. When modelling $\mathcal{T}$ in such a way, the family of shapes is then characterised by a norm that one would set on this flow field. This norm is usually a time integral of a sobolev norm on the flow field that defines the diffeomorphism.

In general, shapes appearing in images are the boundary of some open domain O which is a subset of the image function domain. As we are dealing with bones, which have a smooth boundary, we can treat their shapes as being smooth closed non overlapping curves. As we want to remove the effect of similarity transformations, we can force them to be centered around zero. More formally, shape are functions coming from the set

$$\mathcal{C} = \{\boldsymbol{m} \in C^1\left([0,1];\mathbb{R}^2\right) : \boldsymbol{m}(0) = \boldsymbol{m}(1),\, \boldsymbol{m}(t) \neq \boldsymbol{m}(s) \text{ for } 0 < s \neq t < 1 \\ \text{and } \int_0^1 \boldsymbol{m}(t)\mathrm{d}t = (0,0)\} \tag{4.1}$$

This interpretation now provides us with a new characterisation of shape families. Given a template shape $\bar{\boldsymbol{m}} \in \mathcal{C}$ we can model the small deformation $\mathcal{T}$ through a curve $\boldsymbol{\zeta} \in \mathcal{C}$. Applying $\mathcal{T}$ on $\bar{\boldsymbol{m}}$ would be equivalent to performing the operation $\bar{\boldsymbol{m}} + \boldsymbol{\zeta}$. The norm we would then use on $\mathcal{T}$ would be an integral curve norm on $\boldsymbol{\zeta}$.

4.2.1 Shape correspondence

As we have previously described, we can check whether two shapes are from the same family if, after having undergone some alignment they differ from each other by a small amount. Two curves exhibit shape correspondence if they are in alignment as well as bein in paramteric correspondence.

Let two curves $\boldsymbol{m}_1, \boldsymbol{m}_2$ represent two distinct members from the same shape family. Without loss of generality, we assume that they are of unit arc length. They are in alignment if for an arc length parametrisation of the curves, there exists $t \in [0,1)$ for which $\|\boldsymbol{m}_1(s) - \boldsymbol{m}_2(s+t \mod 1)\|_2$ is minimised for any $s \in [0,1]$. Here, t plays the role of an offset at which we start measuring the arc length. $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ are in **parametric correspondence** if they have the same geometry at any s along their arc length. The geometry of a curve is captured by its derivatives and one can quantify it using a function of the form

$$\mathcal{G} \circ \boldsymbol{m} = \sum_{r=0}^{R} g_r \circ \frac{\mathrm{d}^r}{\mathrm{d}s^r}\boldsymbol{m}. \tag{4.2}$$

where $\mathcal{G}$ is a linear combination of derivative operators g_r of order r. Parametric correspondence means that if one were to sample a point from $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ at the same arc length, then the points would correspond to each other. Hence for an aligned curve that is in parametric correspondence, we have that $\|\boldsymbol{m}_1(s) - \boldsymbol{m}_2(s)\|_2$ is minimised for all $s \in [0,1]$.

In this thesis we align pairs of curves as follows. First the distance transform given by

$$\mathcal{D}(\boldsymbol{x}) = \min_{\boldsymbol{y} \in \boldsymbol{m}_1} \|\boldsymbol{x} - \boldsymbol{y}\|_2$$

of a template curve $\boldsymbol{m}_1$ is found. Then the following minimisation is performed with respect to a similarity transform T

$$\mathrm{T}^* = \arg\min_{\mathrm{T}} \int_0^1 \mathcal{D}(\mathrm{T} \circ \boldsymbol{m}_2(s))ds.$$

$\mathrm{T}^* \circ \boldsymbol{m}_2$ and $\boldsymbol{m}_1$ are now in alignment. One can then use the geometric information of aligned curves to put them in parametric correspondence. Campbell and Kautz [2014] define an energy that takes the same form as equation (4.3) that, when minimised, gives a new parametrisation which puts aligned curves in parametric correspondence.

$$E(t) = \|\mathcal{G} \circ \boldsymbol{m}_1(s) - \mathcal{G} \circ \boldsymbol{m}_2((s+t) \mod 1)\|_2 \tag{4.3}$$

We use a similar type of energy minimisation approach in (2.12) to set our data in parametric correspondence.

4.3 Statistical shape models

The previous definition we have given about shape families uses a norm to measure how far away shape examples are from each other and then choosing how big this norm should be. A more practical approach would be to fit a distribution on a data set of similar shapes and use this distribution to define the central tendency of this shape family.

From now on, we treat shapes as closed C^1 curves. Given a traning set of shape outlines, we model them as curves that we put in shape correspondence. We describe this process in section 2.3.2. We then discretise each curve so that for the training data $\mathbf{Y} = (\boldsymbol{y}_0, ..., \boldsymbol{y}_{N-1})^T$ we have that $\boldsymbol{y}_i = (\boldsymbol{y}_i^{(0)}, .., \boldsymbol{y}_i^{(d_i)})$, where $\boldsymbol{y}_i^{(k)} = (x_k, y_k)$ is the coordinate of the k-th point on the discretised curve. The aim of statistical shape modelling is to find a multivariate distribution f from which we can sample new shapes.

In the field of statistical shape modelling, the concept of a shape family is defined by the observed data. The most common approach is to model the deviation away from a template shape within that family of shapes. The extent by which shapes are allowed to deviate from each other is learned from the observed data. One could then say that the concept of a shape family is defined by the observed examples from this family.

4.3.1 Linear Shape Model

The simplest way to model deviations of shapes away from each other is through a linear combination of orthogonal vectors. Cootes et al. [2001] does this by modelling shape as a mean shape to which orthogonal variation modes are added. The Linear Shape Model is characterised by

$$\boldsymbol{y} = \bar{\boldsymbol{y}} + \sum_{q=0}^{Q-1} \omega_q \mathrm{v}_q \tag{4.4}$$

where $\bar{\boldsymbol{y}}$ is the mean shape and v_k are the Q orthogonal variation modes. Given a training set $\mathbf{Y}$, v_q are the eigenvectors of $\frac{1}{N}(\mathbf{Y} - \bar{\mathbf{Y}})^T(\mathbf{Y} - \bar{\mathbf{Y}})$.

That is, Cootes et al. [2001] performs a dimensionality reduction on $\mathbf{Y}$ via its principal component analysis. One can use the eigenvalues of $\frac{1}{N}(\mathbf{Y} - \bar{\mathbf{Y}})^T(\mathbf{Y} - \bar{\mathbf{Y}})$ as a proxy to the variance allowed in the weights ω_q. However, the classical formulation of the ASM does not provide us with a distribution over the space of shapes. We provide a probabilistic version of ASM by adopting the formulation of section 3.3.4 where

$$\mathbf{Y}^T = \mathbf{W}\mathbf{t}^T + \eta \tag{4.5}$$

for some diagonal Gaussian noise η with variance β. We also set a Gaussian prior on the rows of $\mathbf{t}$ which as zero mean and a diagonal covariance given by $\alpha\mathbf{I}_Q$. Recall that we show in section 3.3.4 that PCA is the result of marginalising the latent space in the above linear relationship. Lemma 3.4 provides the following expression for the resulting distribution.

$$f(\boldsymbol{y}|\mathbf{W}) = \mathcal{N}(\boldsymbol{y}|\bar{\boldsymbol{y}}, \alpha\mathbf{W}\mathbf{W}^T + \beta\mathbf{I}_P)$$

where $\bar{\boldsymbol{y}}$ is the empirical mean of $\mathbf{Y}$. After maximising the induced likelihood with respect to $\mathbf{W}$ and sending the variance of the observation error η to zero one gets that the linear map $\mathbf{W}$ is given by the matrix of principal components of $\mathbf{Y}$. One then recovers the ASM from equation (4.5) where the rows of $\mathbf{Y}$ are expressed as a linear combination of the orthogonal variation modes described in equation (4.4).

4.3.2 Latent Space Representation through GPLVMs

Interpreting ASM as probabilistic PCA allows us to generate new shapes via sampling as we have a well-defined distribution over the space of shapes. However, what we are seeking is a distribution that is indexed by the latent space position of each shape example. We argue our case for this requirement in section 6.2.1. Lawrence [2005] provides a dual treatment of the model in equation (4.5) that recovers a distribution that is dependent on the latent position.

In fact, as we have shown in section 3.3.4.1 and as argued by Lawrence [2005], this dual treatment is easily extended to treat the shape $\boldsymbol{y}$ as a realisation of a GPLVM. GPLVMs have two advantages over a PCA representation. Firstly, the distribution is expressed as a posterior Gaussian that is dependent on the latent point position. Secondly, the linear relationship between observed data and latent space is relaxed to allow for more general mappings. GPLVMs can also be extended to set a prior over the latent space, hence adopting a fully Bayesian treatment of latent space models as described in section 3.3.6.

For completeness, we summarise the content of section 3.3 when applied to our curve dataset $\mathbf{Y}$. We say that $\mathbf{Y}$ is the realisation of a GPLVM if each row of the centered data $\hat{\mathbf{Y}} = \mathbf{Y} - \bar{\boldsymbol{y}}$ is a noise corrupted realisation of a Gaussian Process with covariance operator given by $k(\cdot,\cdot) : \mathbb{R}^Q \times \mathbb{R}^Q \to \mathbb{R}$ and whose indexing set locations or latent point positions need to be estimated. Here, $\bar{\boldsymbol{y}} \in \mathbb{R}^{N \times 2D}$ is the matrix with rows equal to the mean of $\mathbf{Y}$. As is customary in the Gaussian Process literature, we use a Gaussian noise model with a diagonal covariance matrix variance $\frac{1}{\beta}\mathbf{I}_N$. The data likelihood is given by:

$$f(\hat{\mathbf{Y}}) = \prod_{d=0}^{2D-1} \frac{1}{\sqrt{\det\left(2\pi\left(\mathbf{K} + \frac{1}{\beta}\mathbf{I}_N\right)\right)}} \exp\left(-\frac{1}{2}\hat{\mathbf{Y}}_{:,d}^T\left(\mathbf{K} + \frac{1}{\beta}\mathbf{I}_N\right)^{-1}\hat{\mathbf{Y}}_{:,d}\right) \quad (4.6)$$

where $\mathbf{K}$ is a symmetric positive matrix with entries given by $k(\boldsymbol{t}_i, \boldsymbol{t}_j)$. The model is fitted by optimising this with respect to the latent point positions and the covariance operator hyper-parameters.

Given the latent point positions $\mathbf{t} \in \mathbb{R}^{N \times Q}$ for the training data $\mathbf{Y}$, the shape generation process for a GPLVM model takes the form of a posterior Gaussian distribution $f(\boldsymbol{y}^*|\boldsymbol{t}^*, \mathbf{t}, \mathbf{Y})$ with mean and variance given by

$$
\begin{aligned}
K_{\text{post}}(\boldsymbol{t}^*) &= k(\boldsymbol{t}^*, \boldsymbol{t}^*) - k(\mathbf{t}, \boldsymbol{t}^*)^T \left(\mathbf{K} + \frac{1}{\beta}\mathbf{I}_N\right)^{-1} k(\mathbf{t}, \boldsymbol{t}^*) \\
\boldsymbol{\mu}_{\text{post}}(\boldsymbol{t}^*) &= \bar{\boldsymbol{y}} + k(\mathbf{t}, \boldsymbol{t}^*)^T \left(\mathbf{K} + \frac{1}{\beta}\mathbf{I}_N\right)^{-1} \hat{\mathbf{Y}}
\end{aligned}
\tag{4.7}
$$

where $k(\mathbf{t}, \boldsymbol{t}^*) = [k(\boldsymbol{t}_0, \boldsymbol{t}^*), ..., k(\boldsymbol{t}_{N-1}, \boldsymbol{t}^*)] \in \mathbb{R}^N$.

4.3.3 Discussion

The shape generation process for both the GPLVM model and the PPCA model involves adding a mean to some variation modes. In the case of the PPCA representation in equation (4.4), the variation modes consists of the basis elements for the data space. Equation (4.7) on the other hand uses a basis expansion given by $\left(\mathbf{K} + \frac{1}{\beta}\mathbf{I}_N\right)^{-1} \hat{\mathbf{Y}}$ as the variation modes. The weights for a new example is then given by how similar the new data example is to the already observed examples through the covariance $k(\boldsymbol{t}^*, \mathbf{t})$.

Another interpretation of the GPLVM model comes from the fact that predicting unseen examples using Gaussian processes is analogous to using basis functions to learn a general function mapping. Indeed, due to their properties, covariance functions behave as basis functions in Reproducing Kernel Hilbert spaces, and hence can be used to approximate functions. The data examples then serve as weights for the basis functions.

Both of the models we have described so far do not make any explicit assumptions about the space of shape families. Instead their properties are learned from data. In particular, when using a GPLVM to represent a discretised curve, the samples produced from the posterior distribution in equation (4.7) are not necessarily smooth curves. They, in fact take the form of a curve that can be thought of as white noise. One way to get around this problem is to use a non parametric curve representation as the data $\mathbf{Y}$. For example, we could express curves using an elyptic Fourier representation given by

$$
\boldsymbol{m}(s) = \left(\sum_{j=0}^{N_a-1} a_j \cos(2j\pi s) + b_j \sin(2j\pi s), \sum_{j=0}^{N_a-1} c_j \cos(2j\pi s) + d_j \sin(2j\pi s) \right)
\tag{4.8}
$$

and then use a GPLVM or a PCA representation on the Fourier coefficients $\boldsymbol{y} = (a_0, ..a_{N_a-1}, b_0, ..b_{N_a-1}, c_0, ..c_{N_a-1}, d_0, ..d_{N_a-1})$ instead of on the curve points.

Even in this case, we would require the higher Fourier frequencies to be small so as not to get curves samples that are not smooth. We choose to model a point cloud directly as it is more useful for the bone identification algorithm that we later present in Chapter 6.

Even though the GPLVM gives us a posterior distribution from which we can sample curves, we are not interested in these samples per se. Instead, we consider the posterior mean to be the shape that is generated at the corresponding latent position. The posterior variance then gives us a measure of how confident we are that the shapes being generated is one that comes from our shape family.

4.4 Shape modelling through GPLVMs

The main purpose of a shape model in this thesis is the role of shape prior in the bone identification algortithm that we present in Chapter 6. As we later describe, the GPLVM prior will take the form of a generative shape model. Hence, we want the shape generation process to capture as much variation as is allowed within the shape family it is modelling.

We investigate three types of generative GPLVM models:

(a) a shared latent space for different shape classes,

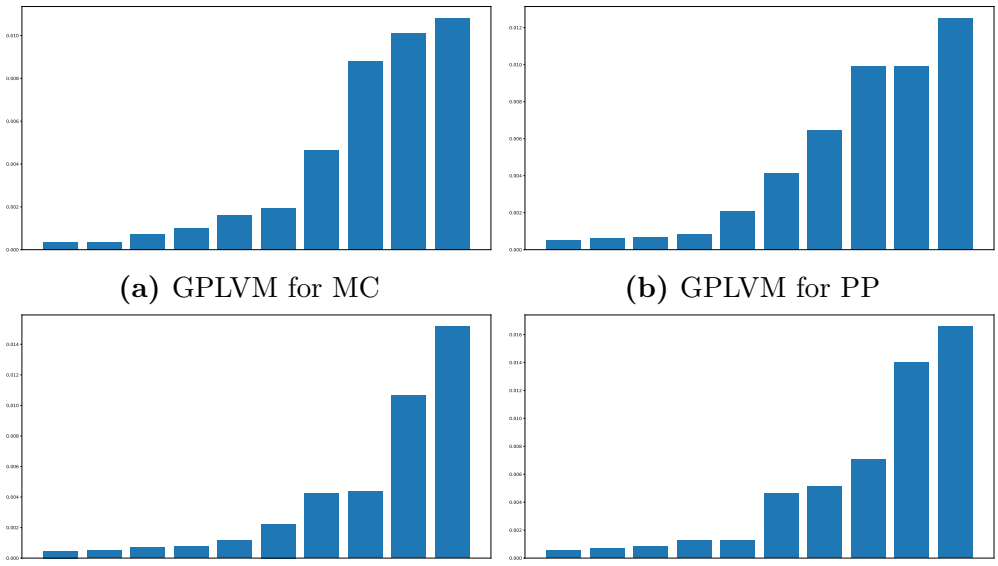
(b) an individual latent space for each shape class, and

(c) a latent space for a compound object.

In particular, we show that when it comes to modelling a single shape, it is better for our purpose to have one latent space per class as opposed to one latent space for multiple shape classes.

We show the shape generation process for the three different models we have described in Figures 4-2, 4-3 and 4-4. In each figure, the middle and right plots show the posterior variance given by equation (4.7) using a colour plot. The rightmost figure shows the most dominant latent space dimension while the middle plot shows the next two dominant dimension. We explain how the dimensions are sorted in section 4.4.1.

The latent space plot on the right shows how the variance changes as one moves through the first two sorted dimensions while the next two dimensions are



(a) GPLVM for MC **(b)** GPLVM for PP

(c) GPLVM for MP **(d)** GPLVM for DP

Figure 4-1: The Figures shows a bar plot of the sorted lengthscales α_q^{-1} for Variational GPLVMs that uses the kernel given in equation (3.16) to model the bone shapes of the right index finger when we set the latent space dimension $Q = 10$.

kept constant, the value of which is given by the location of the bold black cross in the middle plot. The same applies for the middle plot. The left most plot show the shape generated at the latent point positions given by the black cross in the latent space plots.

We use Variational GPLVMs in all of our experiments. The reason we do so is that it allows us to choose the dimension of the latent space as we describe next. It is worth at this point to mention what this means for our shape generation process. In a Variational GPLVM, we marginalise out $\boldsymbol{t}$ to fit the model through an approximation. This approximation yields values for the hyperparameters that would not necessarily be optimal if we were able to optimise $p(\mathbf{Y}|\mathbf{t})p(\mathbf{t})$ directly.

From a modelling point of view though, we still have that $p(\mathbf{Y}|\mathbf{t})$ is a Gaussian Process. We hence maintain that $p(\boldsymbol{y}^*|\boldsymbol{t}^*, \mathbf{Y}, \mathbf{t})$ should have the distribution given by equation (4.7). In our case, the point values we choose for $\mathbf{t}$ are prior means that we recover from fitting the Variational GPLVM.

4.4.1 Choosing the latent space dimension

The latent space dimension is to be chosen. Using the covariance kernel we describe in equation (3.16), we are able to sort the dimensions in order of importance through the length scales. Large values of α_q means small contributions of the

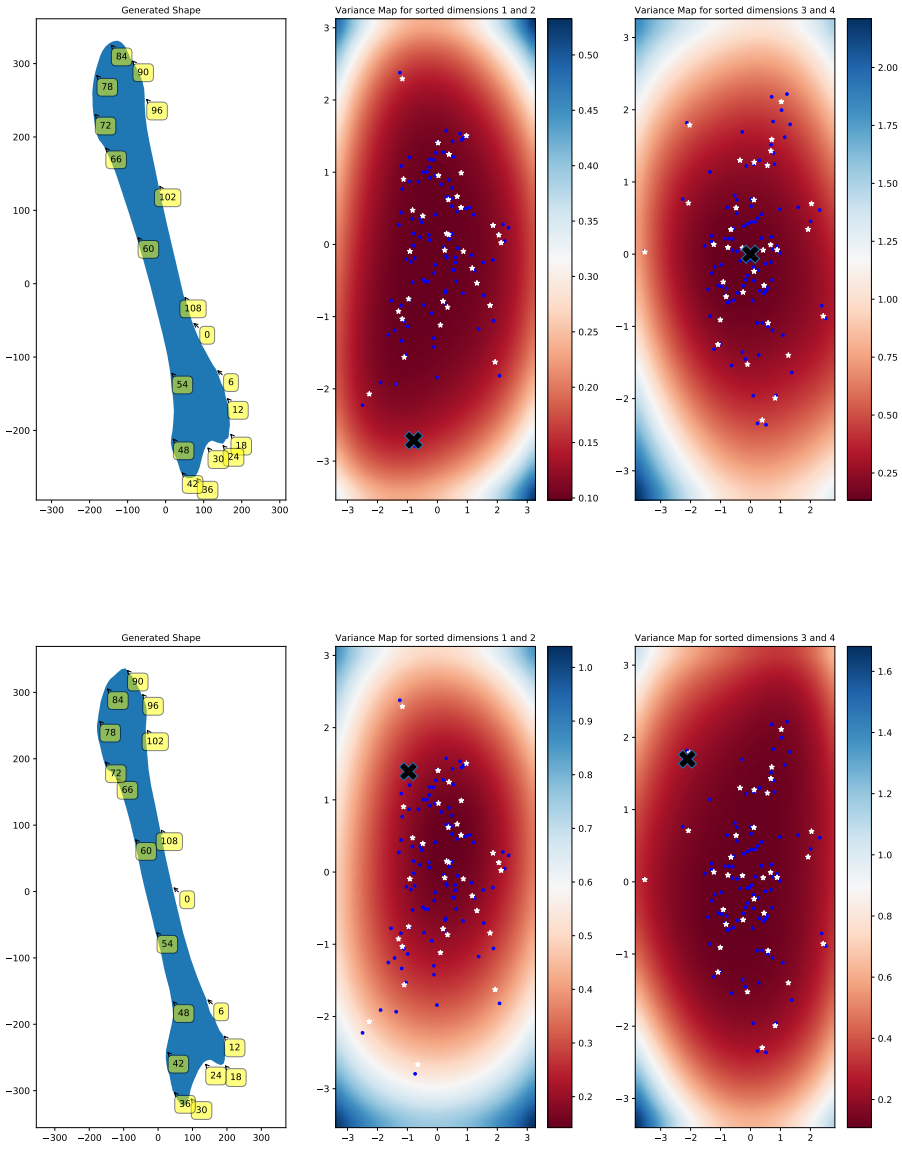


Figure 4-2: The figure shows a Variational GPLVM modelling the MC in the right index finger. The curve points were upsampled from the hand drawn curve outlines using a spline representation. The latent space dimension for each case is 4. The right-most plot shows how the posterior variance varies with the two most dominant dimensions of the latent space while the middle one shows the same relationship but with the two least dominant dimensions.

q-th dimension of the latent space to the variability of the model.

In practice, we choose a large latent space dimension and look at the length scale plots. We choose the dimension based on how fast the lengthscales decay. Figure 4-1 shows the values of the length scale when we set $Q = 10$. It can be seen that the lengthscales start to level from dimension 5 onwards. This suggests that latent space dimension of $Q = 4$ is warranted.

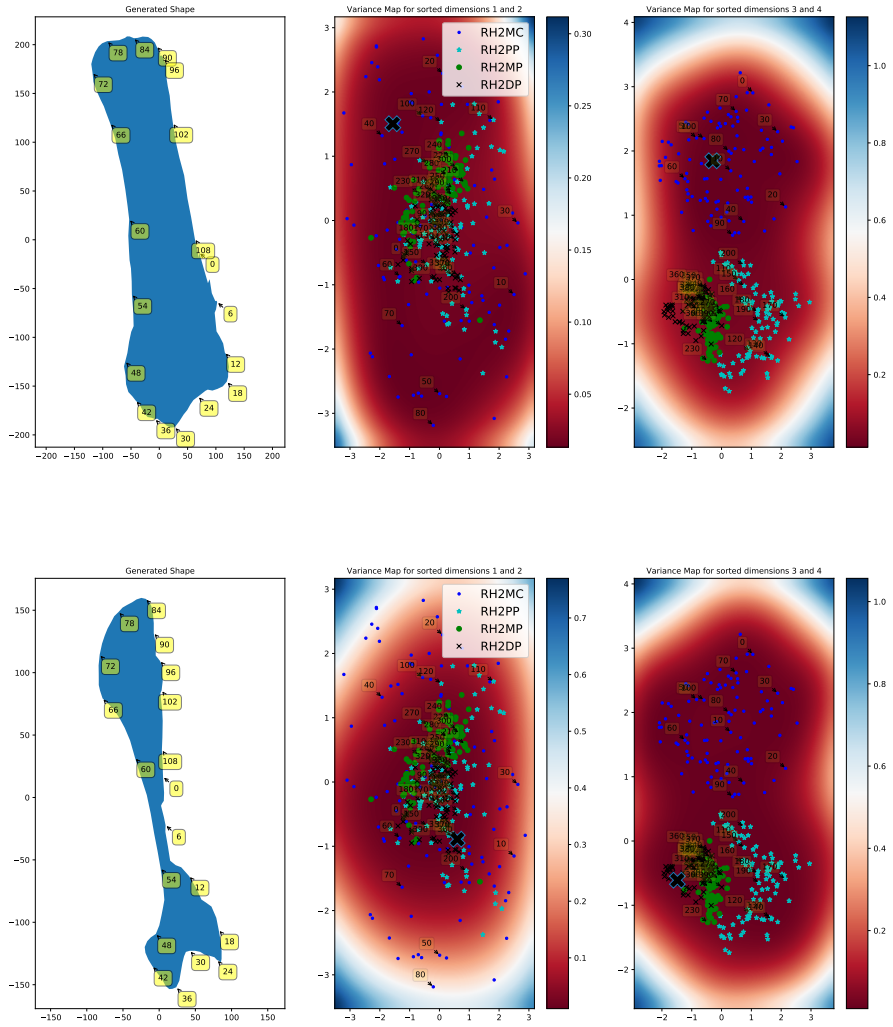


Figure 4-3: The figure shows a Variational GPLVM modelling the four bones in the right index finger. The curve points were upsampled from the hand drawn curve outlines using a spline representation. The latent space dimension for each case is 4. The right-most plot shows how the posterior variance varies with the two most dominant dimensions of the latent space while the middle one shows the same relationship but with the two least dominant dimensions.

4.4.2 Single Bone outline data

We use a Variational GPLVM to model the outline of a single bone. For the shared latent space model, the data takes the form of all the outlines of the right index finger bones that have been aligned. As for the single latent space models, the data consists of the outlines from only one bone class. We use the process described in section 4.4.1 to choose the latent space dimension for each model. The models are fitted using a Tensorflow (Abadi et al. [2015]) implementation by the University of Bath Centre for the Analysis of Motion, Entertainment Research and Applications (CAMERA) group.

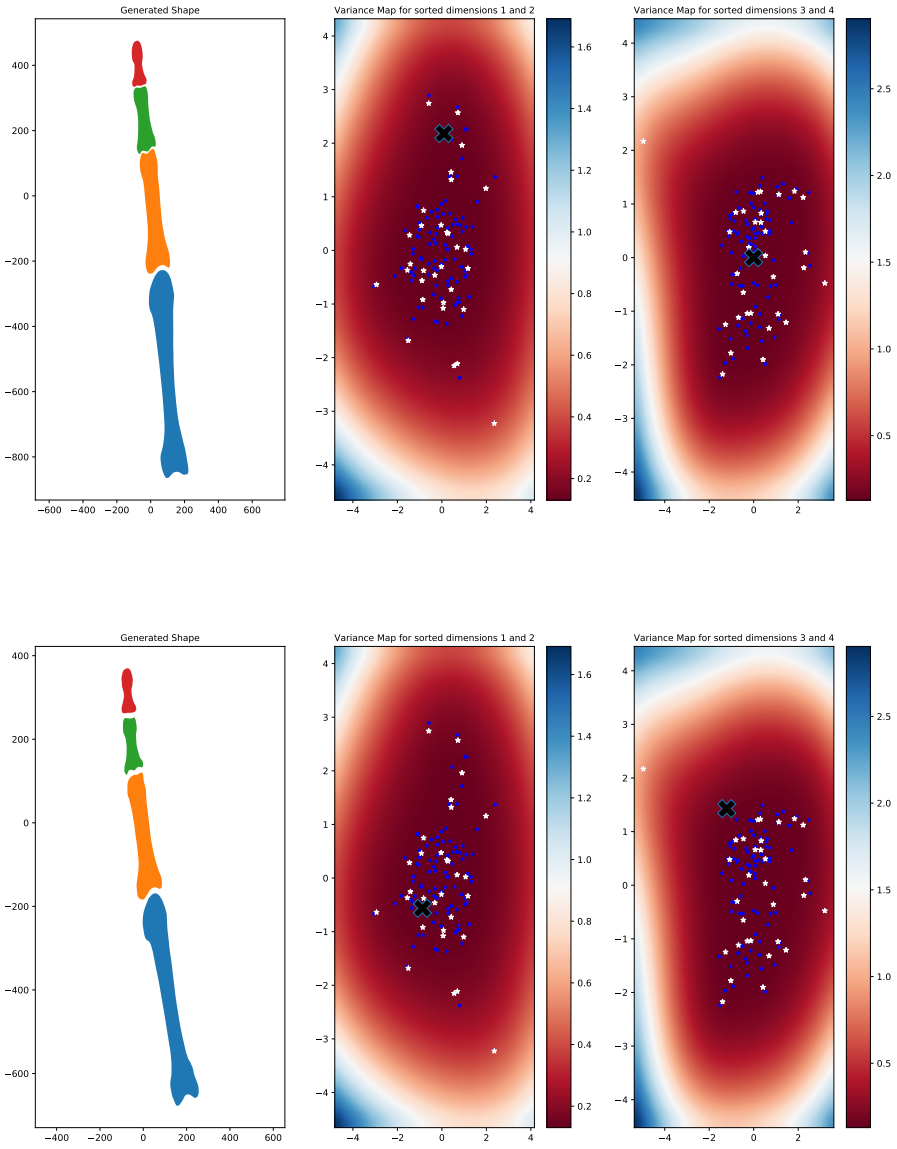


Figure 4-4: The figure shows a Variational GPLVM modelling the right index finger. The curve points were upsampled from the hand drawn curve outlines using a spline representation. The latent space dimension for each case is 4. The right-most plot shows how the posterior variance varies with the two most dominant dimensions of the latent space while the middle one shows the same relationship but with the two least dominant dimensions.

We show the results of the shape generation process for the single latent space model in Figure 4-2. The black dots on the latent space plots represent the mean of the prior given in equation (3.31) that we set on the latent space positions for each data point while the white stars are the inducing points $\mathbf{Z}$ that allow use to marginalise out the latent point position in section 3.3.6.

We do not plot the inducing points in the shared latent space models in Figure 4-3 but instead show only the mean of the latent point positions for each data example. The MC examples are shown as blue dots; the PP as cyan stars; the MP as green dots; and the DP as black crosses [1]. It can be seen that the first two dominant dimensions perform some form of clustering on the data examples. As is to be expected, the DP and the MP are close together in the latent space due to their similar sizes. As the MC is the biggest bone in the right index finger, it lies further away from the other classes in the latent space.

We find that the shared latent space model struggles to perform the shape generation task that we are interested in. In particular, Figure 4-3 shows that it cannot generate a DP from a latent point position that corresponds to a DP example. It is therefore preferable to use one latent space representation per shape class.

To understand why this is the case, recall that in equation (4.7) the shape generation process involves modelling a posterior perturbation given by

$$k(\boldsymbol{t}^*, \mathbf{t}) \left(\mathbf{K} + \frac{1}{\beta} \mathbf{I}_N \right)^{-1} \hat{\mathbf{Y}}$$

to which the mean shape is added. The mean shape of one bone in the case of the dataset consisting of the three phalanges and the meta carpal will not necessarily represent the mean shape of any one class. Hence, generating a sensible bone shape depends on the ability of $k(\boldsymbol{t}^*, \mathbf{t}) \left(\mathbf{K} + \frac{1}{\beta} \mathbf{I}_N \right)^{-1} \hat{\mathbf{Y}}$ to capture variations away from this mean shape.

Moreover, the variation that needs to be added to generate a shape at latent point position $\boldsymbol{t}^*$ can be viewed as a weighted sum of the variation that was already observed through $\left(\mathbf{K} + \frac{1}{\beta} \mathbf{I}_N \right)^{-1} \hat{\mathbf{Y}}$. Hence, if one wants to generate, say a DP, then the latter will share some characteristics with a MC. Given that this is the case, it is hard to add enough variation to the mean shape so as to generate a sensible shape.

4.4.3 Multiple Bone outline data

The data used is a set of aligned bones from the index finger. We show the results of the shape generation process in Figures 4-4. The black dots on the

[1] Recall that MC stands for metacarpal, PP for proximal phalanx, MP for middle phalanx and DP for distal phalanx

latent space plots represent the mean of the prior given in equation (3.31) that we set on the latent space positions for each data point. The white stars are the inducing points $\mathbf{Z}$ that allow use to marginalise out the latent point position in section 3.3.6.

The Variational GPLVM mostly captures variation coming from the relative positions of bones within the right index finger. In other words, it is modelling the relative pose of bones with respect to each other. We hence use this model to initialise the pose parameters of our bone identification algorithm in Chapter 6.

4.5 Concluding remarks

We have shown that when a dual probabilistic treatment of the latent space map is done, the ASM model uses a GPLVM as the latent space representation. This probabilistic treatment equips GPLVM shape models with an inherent density. We later use this property to marginalise out the shape generation process in our bone identification algorithm in Chapter 6.

It is possible to build a latent space out of an ASM model. From a shape generation point of view, the latent space is given by the weights ω_q in equation (4.4). One could then build a density on this space by looking at the distribution of the training weight set. However, from a Bayesian point of view, it is unclear how this density propagates through to the data space. GPLVMs on the other hand, have this property built into the model. Finally, GPLVMs relaxes the linearity assumption on the shape generation process, which we believe is very important.

Chapter 5

Mathematical representation of texture

5.1 Introduction

Erosion and fusion as well as the tissue swelling that is common around PsA joint damage are characterised by a shape change as well as a texture change within the x-ray. Hence it is important for us to model the texture of an image. Moreover, the model we describe in chapter 6 uses texture information to guide the shape model matching. We show in this chapter how one can use learnable image filters to extract the relevant texture information.

More precisely, given an image $u : \Omega \to [0, 255]$, and a subset of the image domain A, the task that we are trying to perform is to model a potential $V_u : \Omega \to [0, \infty)$ such that for a pixel location $\boldsymbol{x}$ we have that

$$\mathbb{P}(\boldsymbol{x} \in A|u) \propto \exp\left(-\frac{1}{2}V_u\right) \tag{5.1}$$

This is the data term in the snakes model that we describe in equation (3.10). We later extend this notion of data term in Chapter 6 to be the energy of a shape model within an image domain .

The earliest attempts at modelling V_u involved using simple edge detectors, examples of which include thresholding and gradient based method such as the Canny edge detector (Canny [1987]). Once the edge set Γ is found, V_u takes the form of a distance transform given by

$$D_\Gamma(\boldsymbol{x}) = \min_{y \in \Gamma} \|\boldsymbol{y} - \boldsymbol{x}\|_2 \tag{5.2}$$

Note that, given annotated data, modelling V_u could be seen as a supervised learning task. Cootes et al. [2012] uses regression trees to determine which direction one should move in to find the area of interest A. Even though, a heat map of votes is built, this is in effect, modelling the global gradient of V_u.

This chapter is organised as follows. We give a definition for texture in section 5.2. We then show two ways that one can form statistical models of texture in section 5.2 where we justify our use of a discriminative model as opposed to a generative model. We then show how the discriminative task can be performed using Neural Networks in Section 5.4. In particular we introduce the two architectures we use in sections 5.4.2.1 and 5.4.2.1. The results of these architectures are presented in section 5.5. We describe how we aim to build V_u from output of these models in section 5.5.3.

5.2 Texture Definition

The literature does not have a rigorous mathematical treatment of texture, the definition of which changes based on the task at hand. If we were to adopt the philosophy present in the mathematical definition shape, then texture would be what remains after one removes the effect of exposure and luminosity from an image. Unlike shape, however, texture tends to exibit more variations and hence properly defining what it means to remove these effects is more challenging.

For the purpose of this thesis, we view the texture of an image at a spatial location $\boldsymbol{x}$ within the image domain to be the intensity profile in a neighbourhood of that image when the effects of exposure and luminosity have been removed. Hence, before modelling texture, we would have to normalise the image in some way.

We could also define texture by the representation we choose to adopt to model it. One of the earliest representations comes from the definition of scale space given by Lindeberg [1994] where images are transformed via some kernel indexed by t

$$L_t(\boldsymbol{x}) = K_t \star u(\boldsymbol{x}) \tag{5.3}$$

The scale space representation is in a loose sense what the human visual cortext would perceive as the object is moved further away from the observer. Lindeberg [1994] concluded that K_t should be a Gaussian kernel of variance t centered

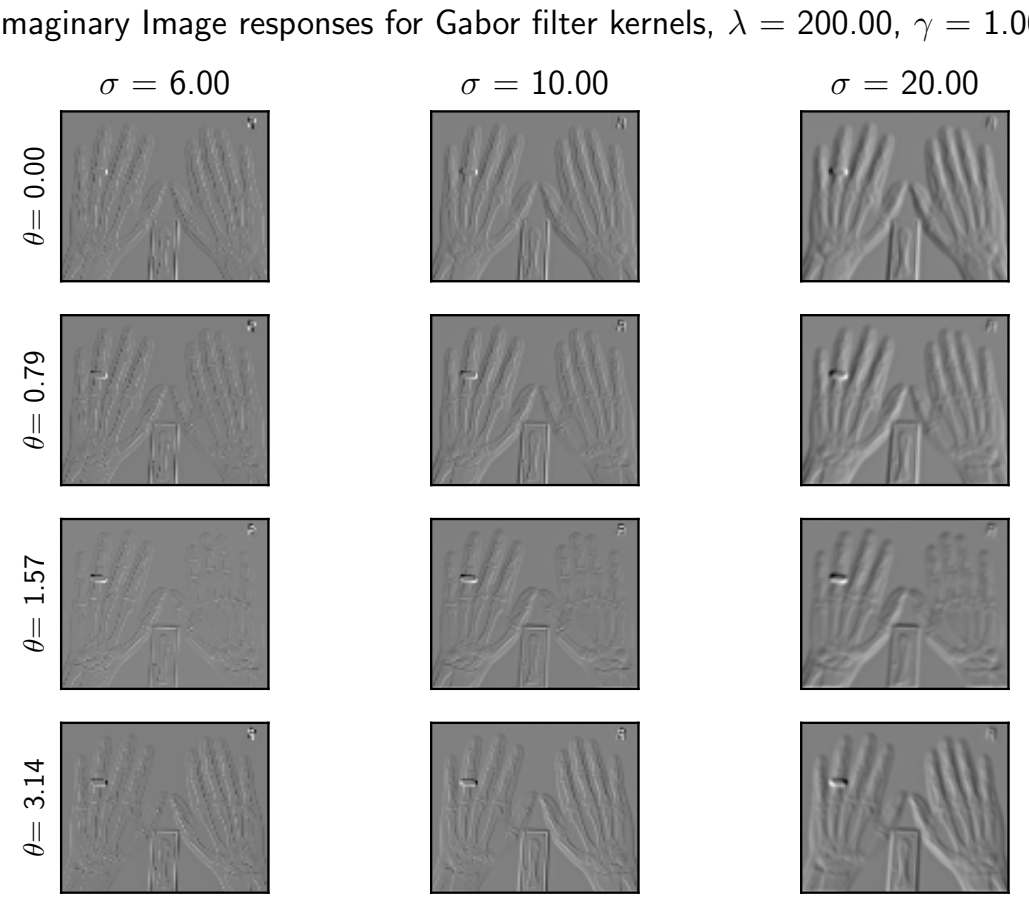


Figure 5-1: The Figure shows the imaginary part of a gabor filtered image for different values of θ and σ. The imaginary response is good to model the occurence of edges in images.

around $\boldsymbol{x}$. In his interpretation, all the texture information comes from blurring images with Gaussian kernels of varying length scales.

The Fourier decomposition of an image is also a popular way of representing texture in the signal processing literature. Images can be thought of as a superimposition of sinusoidal waves of varying frequencies and orientation. The higher frequencies tend to represent finer details in the image that correspond to areas of high variation within the image surface. Hence, texture modelling methods tend to focus on isolating the different components of this superimposition. This is done by using band pass filters that cut off certain frequencies.

One popular such filter is the Gabor filter. The kernel used is given by

$$K(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x_\theta^2 + \gamma^2 y_\theta^2}{2\sigma^2}\right) \exp\left(i\left[2\pi\frac{x_\theta}{\lambda} + \psi\right]\right) \tag{5.4}$$

where $x_\theta = x\cos\theta + y\sin\theta$ and $y_\theta = -x\sin\theta + y\cos\theta$. The angle of incidence θ

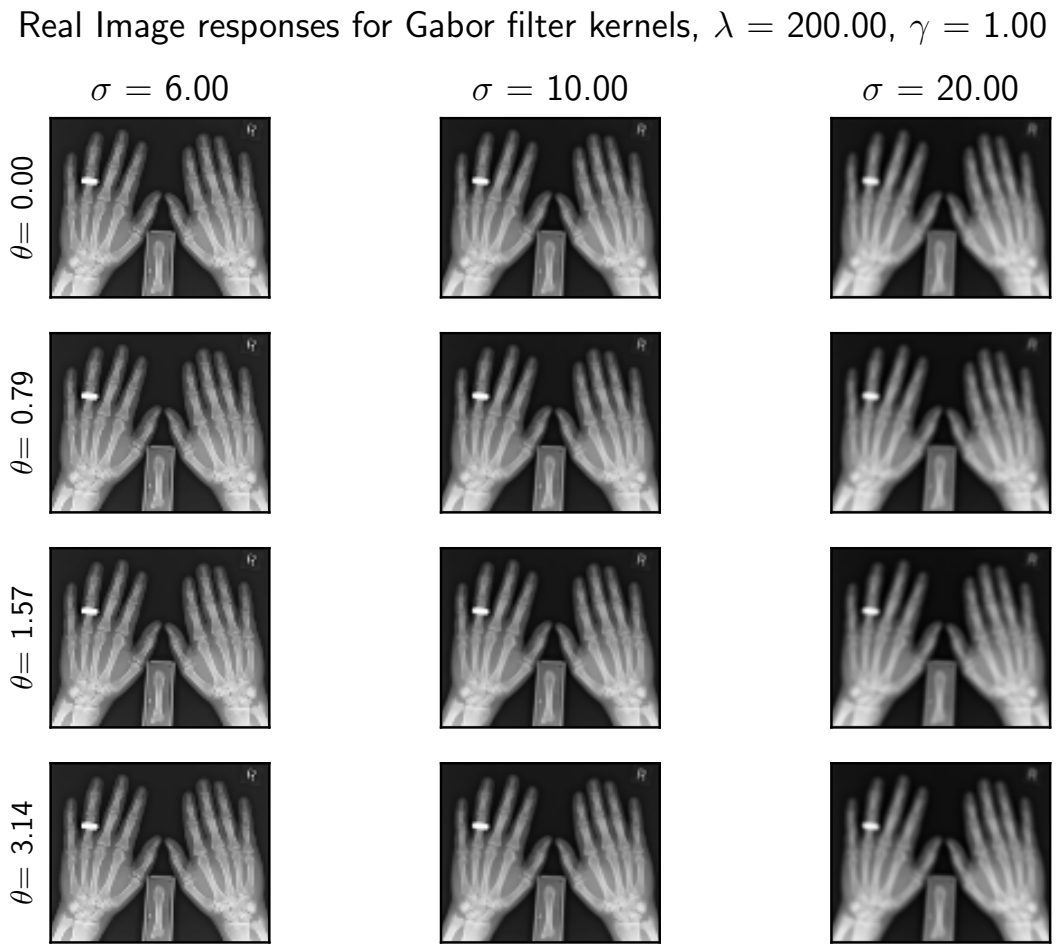


Figure 5-2: The Figure shows the real part of a gabor filtered image for different values of θ and σ. As opposed to the imaginary part, the real part is not affected by the angle θ. It is equivalent to blurring the image with a Gaussian blur.

determines the orientation of the wave that is being extracted. The Filter is split into two parts, the real and the imaginary.

The real part is controlled by the blurring length scale σ and the stretching parameter γ. σ determines the scale at which the features are being extracted in the same way that a Gaussian blur operates. γ stretches space and controls the direction in which most of the blurring occurs. A smaller value for γ means less bluring in the y_θ direction. The imaginary part is controlled by the wavelength λ, and a phase shift ψ. These control the frequency of the wave to be isolated. Figures **??** and 5-2 show an image which has been convolved with Gabor filters.

We finish this section by adopting the following representation for texture. Given a filter bank $(K_0, K_1, ..., K_{L-1})$ consisting of L filters, we define the texture of an image around a spatial location $\boldsymbol{x}$ to be the stack $(K_0 \star u(\boldsymbol{x}), K_1 \star u(\boldsymbol{x}), ..., K_{L-1} \star u(\boldsymbol{x}))$ of length L of filtered image intensities. Here, $K_i \star u(\boldsymbol{x})$ is

the filtered image intensity at pixel location $\boldsymbol{x}$.

5.3 Texture Modelling

We have defined a representation for image texture. As we have previously described for shape modelling, we would like to fit a statistical model for damage detection. There are two main approaches one could use when modelling texture. The first one is the **generative approach**. That is, one fits a distribution to the stack of filtered image intensities and new texture examples are generated by sampling.

In fact an extension to Cootes et al. [1995] is presented by Cootes et al. [2001] and has texture as part of the modelling. This model is called the Active Appearance Model (AAM). It treats appearance as a mixture of texture and shape information. The training data is given by $\mathbf{M} = (\mathrm{M}_0, .., \mathrm{M}_{N-1})$ where $\mathrm{M}_i = (\boldsymbol{y}_i, \boldsymbol{f}_i) \in \mathbb{R}^{2D+DL}$ is a vector of spatial locations concatenated with a vector of texture intensities $\boldsymbol{f}$ at these respective spatial locations. As in the ASM, the appearance vector is given by

$$\mathrm{M} = \bar{\mathrm{M}} + \sum_{q=0}^{Q-1} \psi_q \mathrm{u}_q \tag{5.5}$$

where $\bar{\mathrm{M}}$ is the mean appearance and u_q are orthogonal variation modes. The vector $\boldsymbol{f}$ is usually obtained after applying a texture extracting filter to the training image set. The problem with such an approach is that knowing which filter to use is hard.

More recently, Variational Autoencoders (Doersch [2016]) have been used to generate image textures. These are similar to latent space shape models that we have described in Chapter 4. They are an extension of autoencoders which use Neural Networks to compress information about data into a lower dimensional case. By setting a prior which is learned on this low dimensional space, Variational Autoencoders allows one to sample images by sampling from this low dimensional space.

Providing a definition for texture, as we have loosely done, provides one with a sense of direction when it comes to trying to model them. However, in practice, modelling texture is done with a certain goal in mind. We can treat texture extraction as an intermediate step in a bigger supervised learning task. I.e.,

given an image u and a function $\mathcal{F}$ that represents a supervised learning task, we have that

$$\mathcal{F} \circ u(\boldsymbol{x}) = g\left(K_0 \star u(\boldsymbol{x}), K_1 \star u(\boldsymbol{x}), ..., K_{L-1} \star u(\boldsymbol{x})\right) \tag{5.6}$$

where g is a function $\mathbb{R}^L \mapsto \mathbb{R}$ mapping the texture onto the output space of the supervised learning task.

This is called the **discriminative approach** to texture modelling. It involves learning the kernels $K_0, .., K_{L-1}$ that best do a classification or regression task. We focus our attention on performing a classification task with image textures. That is, we aim to characterise a spatial image location based on the texture at that location.

5.4 Discriminative Texture Modelling with Neural Networks

The goal we wish to achieve is to areas within an x-ray that corresponds to a bone edge or to a bone region. I.e., we want to perform the classification task given by

$$\mathbb{P}(\boldsymbol{x} \in A|u) = \mathcal{F} \circ u(\boldsymbol{x}) \tag{5.7}$$

where A is either a bone edge or an actual bone. The function $\mathcal{F}$ can be modelled by a convolutional neural network (CNN). As described in section A.3.4.2, convolutional layers can be thought of as learnable filters that extract texture from images. Hence, we choose to find the locations of bone edges in an x-ray by using a neural network (NN) function f_{net}.

5.4.1 Some Notation

We use the notation of section A.3 to describe NNs. For clarity, we summarise this notation here. We denote the output of an NN with L layers by $\mathbf{Z}_L$. The neural network function f_{net} is a composition of simpler functions given by equation (A.24). Each layer in a neural network is mapped to each other via some transformation given by $\mathbf{Z}_l = \phi_l \circ \mathrm{T}_l\left(\mathbf{Z}_{l-1}\right)$.

A fully connected layer maps its input $\mathbf{Z}_l \in \mathbb{R}^{N_l}$ to the output $\mathbf{Z}_{l+1} \in \mathbb{R}^{N_{l+1}}$ via a matrix multiplication with weight vectors $\mathbf{W}_l \in \mathbb{R}^{N_l \times N_{l+1}}$ to which a bias $\boldsymbol{b}_l \in \mathbb{R}^{N_{l+1}}$ is added. This is followed by the application of an activation function.

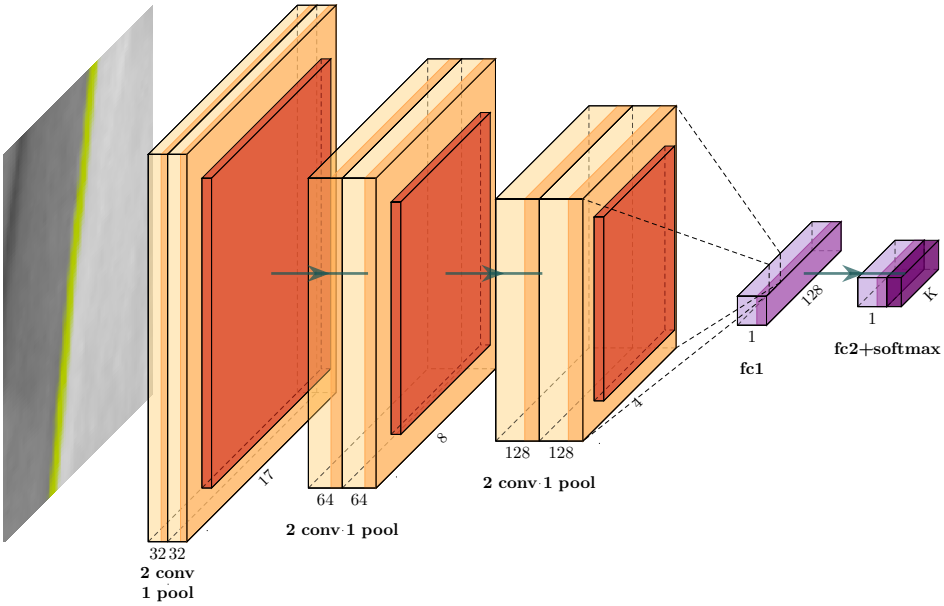


Figure 5-3: The Figure shows the patch-net architecture described in section 5.4.2.1. It has $M = 3$ downsampling layers, $N = 1$ fully connected layer and an initial patch size of $N_0 = 17$.

We denote this operation using $\phi_l\left(\mathbf{W}_l\mathbf{Z}_l + \boldsymbol{b}_l\right)$.

A convolutional layer maps its input $\mathbf{Z}_l \in \mathbb{R}^{N_l^x \times N_l^y \times C_l}$ to the output $\mathbf{Z}_{l+1} \in \mathbb{R}^{N_{l+1}^x \times N_{l+1}^y \times C_{l+1}}$ via a kernel operation with a kernel $\mathbf{W}_l \in \mathbb{R}^{S_l^x \times S_l^y \times C_l \times C_{l+1}}$ to which a bias $\boldsymbol{b}_l \in \mathbb{R}^{N_{l+1}^x \times N_{l+1}^y \times C_{l+1}}$ is added. Like a fully connected layer, this is followed by the application of an activation function. We denote this operation by $\phi_l\left(\mathbf{W}_l \star \mathbf{Z}_l + \boldsymbol{b}_l\right)$. C_{l+1} is the number of features (texture) being extracted from the image present in the previous layer.

Convolutional layers are usually followed by a pooling layer, which we represent by $\boldsymbol{K}_l \star \mathbf{Z}_l$. This is used to change the scale of the image by making it smaller. For an input of size $N_l^x \times N_l^y \times C_l$, pooling layers output a tensor of size $\left\lfloor \frac{N_l^x}{2} \right\rfloor \times \left\lfloor \frac{N_l^y}{2} \right\rfloor \times C_l$. Section A.3 has a more thorough presentation of NN layers.

5.4.2 Patch Based Texture Modelling

The most naive way of building f_{net} would be to split the image into patches and then feed these patches into a CNN. The CNN would output a vector of length N_c that would have the probability of the centre pixel belonging to each one of the classes. We call such an architecture patch-net.

5.4.2.1 Patch-net Architecture

The architecture consists of M downsampling layers followed by N fully connected layers for a total of $L = 3M + N + 1$ layers. Because of the fully connected layers, the square input spatial dimensions $N_0 \times N_0$ should be predefined. Figure 5-3 shows the architecture for $M = 3, N = 1$ and $N_0 = 17$.

Downsampling layers: Each downsampling layer consists of two convolutional layers followed by a pooling layer. The first convolutional layer doubles the number of features from the previous layer. The second one preserves the number of features. The pooling layer halves the size of the twice convolved image in each spatial dimension. Hence we have for $l = 1, .., M$

$$\mathbf{Z}_{3l} = \boldsymbol{K}_{3l} \star \phi_{3l-1} \circ \mathbf{W}_{3l-1} \star \phi_{3l-2} \circ \mathbf{W}_{3l-2} \star \mathbf{Z}_{3(l-1)} \tag{5.8}$$

where $\mathbf{W}_{3l-1}, \mathbf{W}_{3l-2}$ are convolutional kernels of sizes $S \times S \times 2C_{3(l-1)} \times 2C_{3(l-1)}$ and $S \times S \times C_{3(l-1)} \times 2C_{3(l-1)}$ respectively. These are applied with padding $P = \left\lfloor \frac{S}{2} \right\rfloor$ for odd S and strides $d_x = d_y = 1$. The pooling kernel $\boldsymbol{K}_{3l}$ has size 2 in each spatial dimension. It is applied with a stride of 2 on the unpadded output of the 2 convolutional layers. As per equation (A.36), this halves the size of each spatial dimension of the input.

Fully connected layers: For $l = M + 1, ..., M + N$

$$\mathbf{Z}_l = \phi_l \left(\mathbf{W}_l \mathbf{Z}_{l-1} + \boldsymbol{b}_l \right) \tag{5.9}$$

where $\mathbf{W}_l$ is a matrix of weights of size $N_l \times N_{l-1}$ and $\boldsymbol{b}_l$ is the bias term of size N_l. Note that for an input of size $N_0 \times N_0$, we require the $N_M = \left\lfloor \frac{N_0^2}{2^{2M}} \right\rfloor C_M$. That is, we want the number of columns in $\mathbf{W}_l$ to be equal to the total number of neurons present in the final downsampling layer.

Output Layer: The output layer is a fully connected layer where the activation function is the softmax as given in equation (A.26).

$$\mathbf{Z}_{M+N+3} = \sigma \left(\mathbf{W}_{M+N+3} \star \mathbf{Z}_{M+N+2} + \boldsymbol{b}_{M+N+3} \right) \tag{5.10}$$

$\mathbf{W}_{M+N+3}$ is the matrix of weights of size $N_{M+N+2} \times N_c$ where N_c is the number of classes.

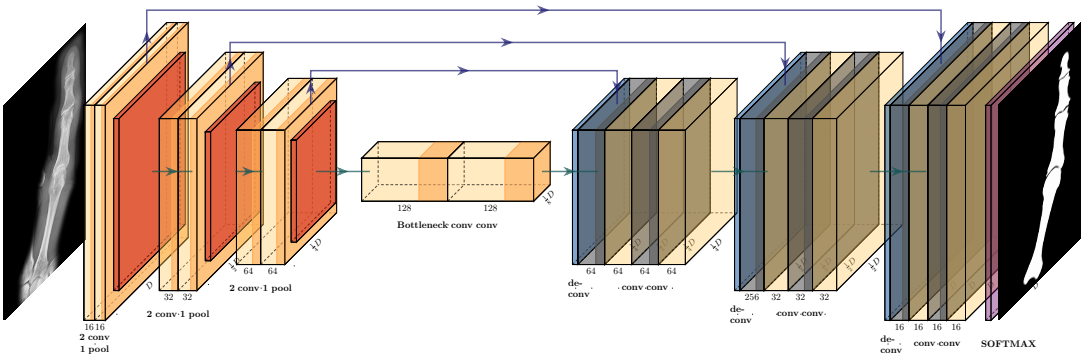


Figure 5-4: The Figure shows the U-net architecture described in section 5.4.3.1. It has $M = 3$ downsampling layers and $C_1 = 16$. The arrows represent the mapping from one layer to the next. The skip connections are shown as archs above the main architecture. The network outputs a segmentation map as shown in the figure. It takes as input a masked image with the mask drawn around the finger.

5.4.3 Purely convolutional semantic segmentation

Using patch-net to make f_{net} involves breaking an image down into patches and then reshaping the collection of predictions into the correct shape. This involves a lot of preprocessing and can be very computationally intensive.

A purely convolutional architecture, on the other hand could get rid of all this processing by instead producing an output having the same spatial dimensions as the input. Given an input image tensor $\mathbf{U} \in \mathbb{R}^{N_0^x \times N_0^y}$ We would then get an prediction map $\mathbf{Z}_L \in \mathbb{R}^{N_0^x \times N_0^y \times N_c}$ such that

$$\mathbf{Z}_L[i, j, h] = \mathbb{P}\left((i, j) \in \text{ class } h\right). \tag{5.11}$$

where (i, j) is the discrete pixel location within the image tensor $\mathbf{U}$.

Such architectures have become very popular for semantic segmentation. Ronneberger et al. [2015] shows how one can use a purely convolutional architecture to segment images of arbitrary sizes. The resulting CNN is called U-net. We describe the architecture in the next section.

5.4.3.1 U-net Architecture

The U-net architecture consists of M downsampling layers, 1 bottleneck layer and M upsampling layers for a total of $L = 6M + 3$ layers. Figure 5-4 shows the architecture for $M = 3$.

Downsampling layers: Each downsampling layer consists of two convolutional layer followed by a pooling layer. The first convolutional layer doubles the number

of features from the previous layer. The second one preserves the number of features. The pooling layer halves the size of the twice convolved image in each spatial dimension. Hence we have for $l = 1, .., M$

$$\mathbf{Z}_{3l} = \boldsymbol{K}_{3l} \star \phi_{3l-1} \circ \mathbf{W}_{3l-1} \star \phi_{3l-2} \circ \mathbf{W}_{3l-2} \star \mathbf{Z}_{3(l-1)} \tag{5.12}$$

where $\mathbf{W}_{3l-1}, \mathbf{W}_{3l-2}$ are convolutional kernels of sizes $S \times S \times 2C_{3(l-1)} \times 2C_{3(l-1)}$ and $S \times S \times C_{3(l-1)} \times 2C_{3(l-1)}$ respectively. These are applied with padding $P = \lfloor \frac{S}{2} \rfloor$ for odd S and strides $d_x = d_y = 1$. The pooling kernel $\boldsymbol{K}_{3l}$ has size 2 in each spatial dimension. It is applied with a stride of 2 on the unpadded output of the 2 convolutional layers. As per equation (A.36), this halves the spatial dimensions of the input to the next downsampling or bottle neck layer.

Bottleneck layers: Once the image has undergone downsampling, it goes through 2 convolutional layers. These two layers are often called the bottleneck layers. As in the previous downsampling layer, the first convolutional layer doubles the number of features from the previous downsampling layer. The second layer keeps the number of features constant. Thus we have:

$$\mathbf{Z}_{3M+2} = \phi_{3M+2} \circ \mathbf{W}_{3M+2} \star \phi_{3M+1} \circ \mathbf{W}_{3M+1} \star \mathbf{Z}_{3M} \tag{5.13}$$

where $\mathbf{W}_{3M+2}, \mathbf{W}_{3M+1}$ are convolutional kernels of sizes $S \times S \times 2C_{3M} \times 2C_{3M}$ and $S \times S \times C_{3M} \times 2C_{3M}$ respectively. These are applied with padding $P = \lfloor \frac{S}{2} \rfloor$ for odd S and strides $d_x = d_y = 1$.

Upsampling layers: The input to the upsampling layer undergoes a transposed convolution. The output from the second convolutional layer in a downsampling layer is chosen such that is has the same size as the current upsampled output. It is then concatenated with the ouput. This is demonstrated in Figure 5-4. The augmented tensor is then passed through 2 convolutional layers. Hence, we have for $l = M + 1, .., 2M$

$$\mathbf{Z}_{3l+2} = \phi_{3l+2} \circ \mathbf{W}_{3l+2} \star \phi_{3l+1} \circ \mathbf{W}_{3l+1} \star \left[\mathbf{Z}_{6M+2-3l}, \left(\phi_{3l} \circ \mathbf{W}_{3l} \star \hat{\mathbf{Z}}_{3l-1} \right) \right] \tag{5.14}$$

where $\mathbf{W}_{3l+2}, \mathbf{W}_{3l+1}$ are convolutional kernels of sizes , $S \times S \times \frac{C_{3l-1}}{2} \times \frac{C_{3l-1}}{2}$; $\mathbf{W}_{3l}$ is a transposed convolutional kernel of size $S \times S \times C_{3l-1} \times \frac{C_{3l-1}}{2}$; and $\hat{\mathbf{Z}}_{3l-1}$ is the output of the previous bottleneck or upsampling layer that has been dilated by a factor of two in each dimension (see section A.3.4.4). All of the kernel operations are applied with padding $P = \lfloor \frac{S}{2} \rfloor$ for odd S and strides $d_x = d_y = 1$.

Output Layer: The output layer consists of a a convolutional layer and the application of a softmax as in equation (A.26)

$$\mathbf{Z}_{6M+3} = \boldsymbol{\sigma} \circ \mathbf{W}_{6M+3} \star \mathbf{Z}_{6M+3} \tag{5.15}$$

where $\mathbf{W}_{6M+3}$ is a convolutional kernel with size $S \times S \times C_{6M+3} \times N_c$. The softmax acts on spatial slices $\mathbf{W}_{6M+3} \star \mathbf{Z}_{6M+3}[i, j, :]$ of the tensor that have length N_c. The output of the network is therefore a spatial map of class probabilities.

5.5 Experimental results

We use both of the models described above on the x-ray data that we have. Each x-ray in our training and test set comes with a raw annotation that takes the form of the hand drawn curve outline described in section 2.2. We adjust the intensity of the training data as described in section 2.4. We are interested in building a classifier that predicts where the bones are. The aim of these models is to use them as the data term in the bone identification algorithm that we present in Chapter 6.

We do not have a set of annotations that covers the whole hand and hence we could only train the models described above on the right index finger. This means that we had to be careful when designing the architectures and when pre-processing the training data. Nevertheless, we found that both patch-net and U-net extended the knowledge gained from this area to the rest of the hand where they achieve a good performance. We present the results for patch-net and U-net in sections 5.5.1 and 5.5.2 respectively.

In both patch-net and U-net the training data we use is artificially modified so as to only expose the areas of the hand for which labels are available. What we are seeking is a model that is able to perform the texture discrimination task over the whole hand. To measure this, we want our labels to cover the whole hand. Hence, even though we report the training and testing accuracy for the Neural Networks we train, these should not be regarded as a good measure of performance of the individual models. The accuracy of a classifier on a batch of input examples is defined to be the number of correct predictions made divided by the total number of examples in that batch.

The loss function we use in our experiments is the cross entropy loss. For a

Algorithm 1: `posSamp`: Algorithm to sample positive patches for patch-net in an annotated image

input : Mask $\mathcal{M}$ showing locations of positive examples with a value of 1

output : Set $\mathcal{P}$ of positive patches coordinates, $\mathcal{N}$ of negative patches coordinates

parameter: spacing s, patch size p, Poisson sampling rate θ, max distance from true patch $d_{\max}$

for (x, y) *in* $\{(x, y) : \mathcal{M}(x, y) = 1\}$ **do**
  if $\|(x, y) - \mathcal{P}\|_1 > s * p$ **then**
    $\mathcal{P} \leftarrow \text{append}(\mathcal{P}, (x, y))$;
    $\mathcal{S} \leftarrow$ `negSamp`$(\mathcal{M}, (x, y), \theta, d_{\max})$ (algorithm 2);
    $\mathcal{N} \leftarrow \text{append}(\mathcal{N}, \mathcal{S})$;
  end
end

batch of size B that is fed into a NN, let $\mathbf{Z}_L \in \mathbb{R}^{B \times N_c}$ be the output of a neural network classifier and let $\bar{\mathbf{Z}} \in \mathbb{R}^{B \times N_c}$ be the labels for that particular batch where

$$\bar{\mathbf{Z}}[b, n] = \begin{cases} 1 & \text{if the } b - \text{th example is in class } n \\ 0 & \text{otherwise} \end{cases}$$

Then the cross entropy loss on this batch is given by

$$R(\Theta; \bar{\mathbf{Z}}, \mathbf{Z}_L) = -\frac{1}{B} \sum_{b=0}^{B-1} \sum_{n=0}^{N_c-1} \bar{\mathbf{Z}}[b, n] \log \left(\mathbf{Z}_L[b, n] \right) \tag{5.16}$$

Where Θ represents the neural network weights $\mathbf{W}_l$ and biases $\boldsymbol{b}_l$ for $l = 1, ..., L$. The loss becomes a function of the NN parameters through the mapping induced on the input by the NN.

5.5.1 Patch-net

We use Patch-net on two tasks. The first one is finding the edges separating the bone from the tissue and the second one is finding the pixels belonging bones within an image.

5.5.1.1 Data

The data for the Patch-net takes the form of square patches centred around pixels that represent a particular area in the x-ray. Figure 5-5 shows how the patches were sampled by using a mask that shows the region of interest. Figure 5-5c shows the patches used to train the bone discriminator while figure 5-5d shows

Algorithm 2: `negSamp`: Algorithm to sample negative patches for patchnet in an annotated image

input : Mask $\mathcal{M}$ showing the coordinates of the positive examples with a 1, coordinates of a true patch (x, y), patchsize p
output : Set $\mathcal{S}$ of coordinates of negative patches
parameter: Poisson sampling rate λ, max distance from true patch $d_{\max}$

$N \leftarrow \text{Poi}(\lambda)$;
for $i \leftarrow 1$ **to** N **do**
  $z \leftarrow \text{unif}[p/2+1, p/2+d_{\max}]$;
  $w \leftarrow \text{unif}[p/2+1, p/2+d_{\max}]$;
  while $\sum \mathcal{M}(x+z, y+w) > 0$ **do**
    $z \leftarrow \text{unif}[p/2+1, p/2+d_{\max}]$;
    $w \leftarrow \text{unif}[p/2+1, p/2+d_{\max}]$;
    $z \leftarrow z \times (2 \times \text{ber}(0.5) - 1)$ (right or left of true patch);
    $w \leftarrow w \times (2 \times \text{ber}(0.5) - 1)$ (above or below of patch);
  end
  $\mathcal{S} \leftarrow \text{append}(\mathcal{S}, (x+z, y+w))$
end

the patches that were used to train the bone edge discriminator.

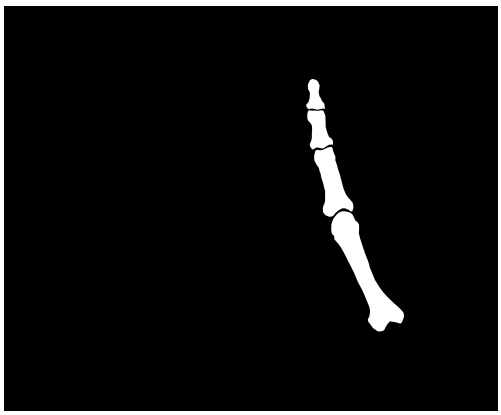
We devised an algorithm to iteratively sample a collection of positive and negative patches. The positive patches come from areas where the mask has a non zero pixel value. We sample each positive sample to have a distance of at least *sp* from the center of all positive patches that have been collected so far. We show how this is done in Algorithm 1.

For each positive patch that is sampled, we sample a random number of negative samples as per algorithm 2. The number of negative patches to be sampled around a positive patch is given by a Poisson random variable with rate λ. For each dimension of the image, the coordinate of the false patch is chosen to be a random at a random distance d from the centre of the positive patch. We sample d from the set $\{p+1, ..., p+d_{\max}\}$ with the uniform measure.

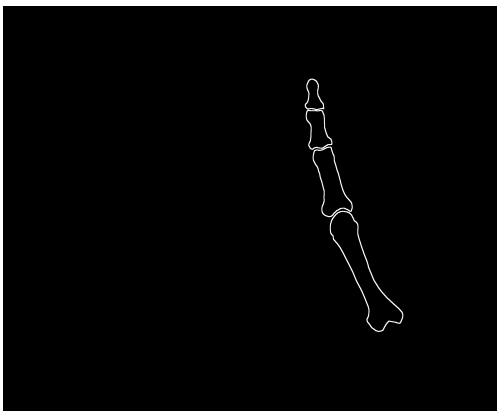
5.5.1.2 Training

We use $M = 2$ downsampling layers and $N = 1$ fully connected layer in the bone region discriminator. We find that using a patch width[1] of 9 works well for the bone region discriminator. We use $M = 3$ downsampling layers and $N = 1$ fully connected layer in the edge discriminator. We use a patch width of 17.

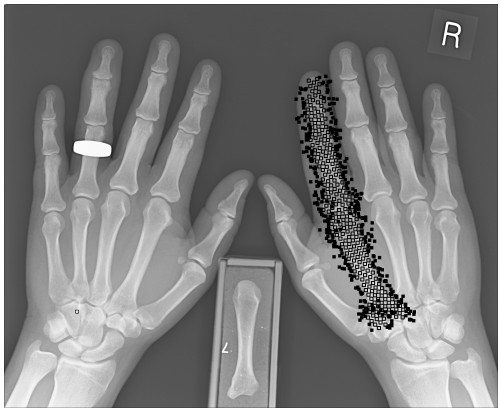
[1]Size of input image described in section 5.4.2.1 which was denoted by N_0



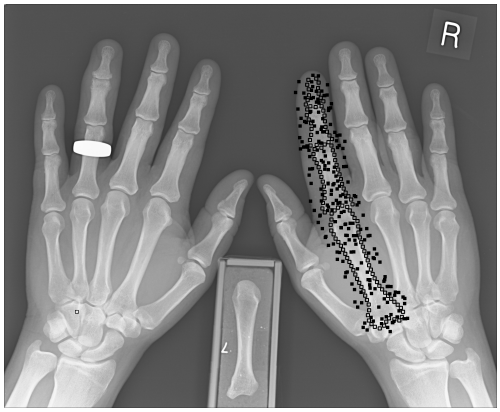
(a) Mask showing the bones.



(b) Mask showing the bone edges.



(c) Samples generated by the mask in Figure 5-5c.

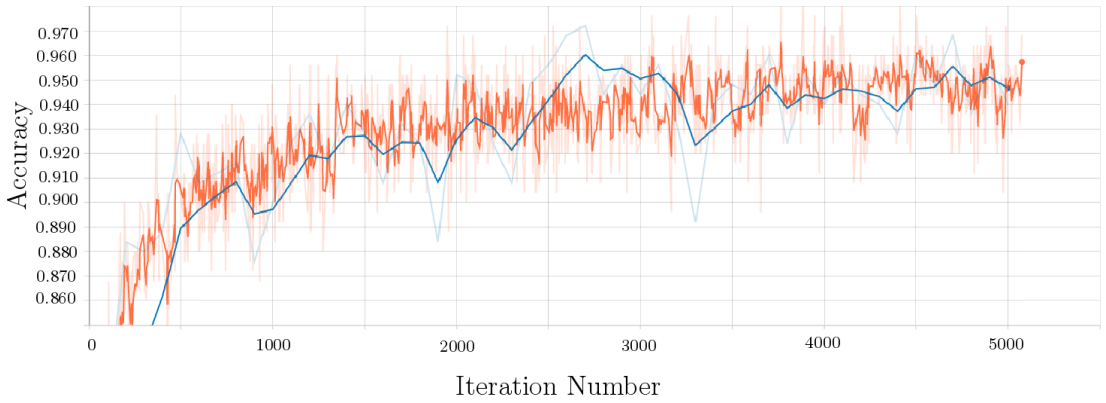


(d) Samples generated by the mask in Figure 5-5d.

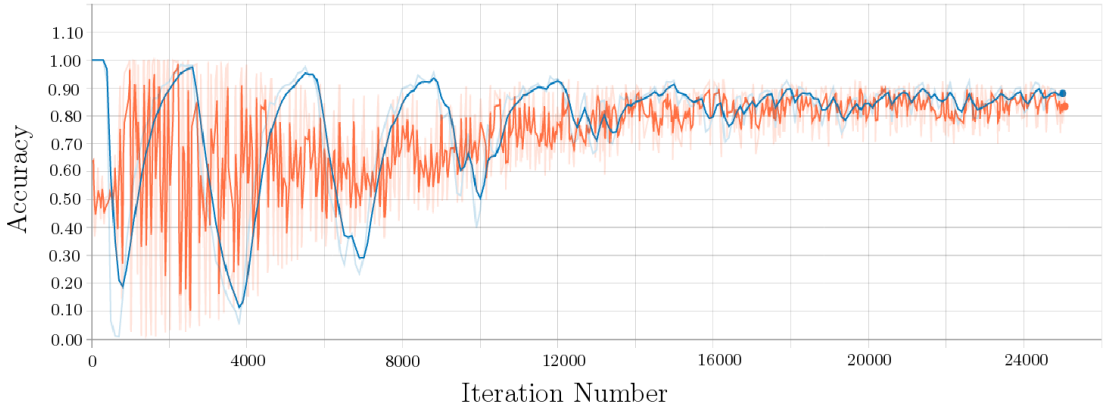
Figure 5-5: The Figure shows two masks and the samples they generate that are used to train patch-net. The black squares are the sampled negative patches while the empty squares show the sampled positive samples.

For each discrminator that we train, we split the patches that have been collected in a training and validation set. We use an Adam optimiser (see section A.2.0.6) with a learning rate of 0.0001. We train the model for 200 epochs and with a batch size of 400. We report a test accuracy of 94.66% for the edge discriminator and 88.75% for the bone discriminator. Figure 5-6 shows the trace plot for the accuracy.

The trace plots do not depict any over fitting as the test set accuracy and the training set accuracy do not diverge from each other. Our models do not achieve a very high accuracy rate when compared to modern Neural Network performances. This is partly because our sampling algorithm introduces false negatives.This has a higher chance of happening in the bone discriminator as depicted by Figure 5-5c, which explains the higher accuracy of the edge detector. The algorithm starts to sample patches inside the MC of the middle finger and treats them as negative samples.



(a) Edge discriminator trace plot.



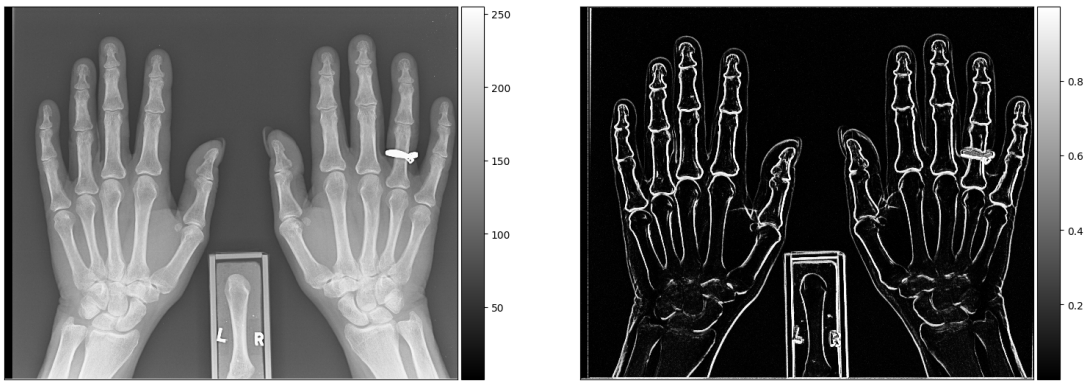
(b) Bone discriminator trace plot.

Figure 5-6: The Figure shows the accuracy for the patch-net discriminators evolve as training progresses. The x-axis shows the stochastic gradient descent iteration number and the y-axis shows the accuracy at the corresponding step. We show the test set accuracy (blue) and the training set accuracy (red). The cyclical behaviour or the test set accuracy for the bone discriminator comes from alternating between easy to pedict batches and hard to predict bacthes.

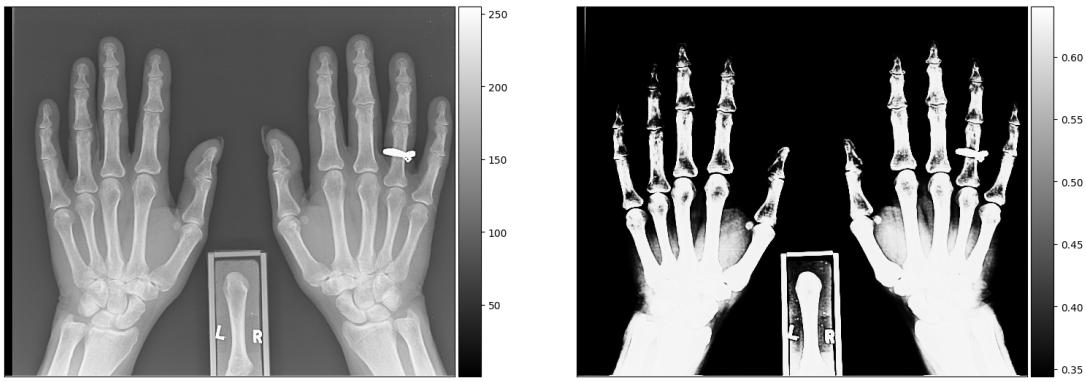
5.5.1.3 Prediction

We test the models on full hand x-rays. A typical x-ray has a resolution of 2400×2000 pixels. This amounts to running the prediction step on 4.8×10^6 patches. Using patch-net to create predictions for one image is hence very expensive.

Algorithm 3 shows how the prediction can be done faster. We use a sliding window on the image. The window selects a section of the image on which we run tensorflow's patch extraction function. This operation is parallelised on a GPU. Then for each patch that is extracted, we find the class probabilities using f_{net}. The resulting predictions are then reshaped and glued together after each pass of the window so as to build the probability map. We assume in algorithm 3 that the image is padded so that the windowing operation is valid.



(a) Edge discriminator prediction.



(b) Bone discriminator prediction.

Figure 5-7: The Figure shows a full hand prediction for the patch-net edge and bone discriminators. The right plots shows the probability of a pixel being either a bone edge or a bone.

Figure 5-7 shows the result of running patch-net on two whole images that have not been annotated. We can see that the model predicts areas of the hand that it was not trained on. We also see that the model is more confident when predicting edges than when predicting bones. This might be due to the fact that there are more false positives in the bone discriminator training data than in the bone edge discriminator training data.

5.5.2 U-net

As described by Ronneberger et al. [2015], we use U-net as a bone region discriminator. We find that it works better for this task. We find that it does not work as well as patch-net for an edge discriminator task.

5.5.2.1 Data

As with patch-net, we did not have a full set of annotations for whole hand x-rays. We masked the areas for which annotations were not available and cropped the image around this masked area. As for the labels, they do not take the form

Algorithm 3: `predPatch`: Algorithm for fast prediction with patch-net.

input : image $\mathbf{U}$ of size $N_x \times N_y$, patch-net predictor f_{net}
output : Probability map $\mathbf{P}$ of size $N_x \times N_y \times K$ showing the probability of pixel i, j being in class k for $k = 1, .., K$
parameter: Size of sliding window $S_x \times S_y$, patchsize p

$n_x \leftarrow \left\lfloor \frac{N_x}{S_x} \right\rfloor$;
$n_y \leftarrow \left\lfloor \frac{N_y}{S_y} \right\rfloor$;
$\mathbf{U} \leftarrow \texttt{pad}(\mathbf{U}, \texttt{xborder} = p, \texttt{yborder} = p)$;
for $i \leftarrow 0$ **to** $n_x - 1$ **do**
  for $j \leftarrow 0$ **to** $n_y - 1$ **do**
    $\mathbf{C} \leftarrow \mathbf{U}[iS_x - \left\lfloor \frac{p}{2} \right\rfloor + 1 : (i+1)S_x + \left\lfloor \frac{p}{2} \right\rfloor + 1, jS_x - \left\lfloor \frac{p}{2} \right\rfloor + 1 : (j+1) * S_y + \left\lfloor \frac{p}{2} \right\rfloor + 1]$ (sliding window operation);
    $\mathbf{C} \leftarrow \texttt{tf.extract\_patches}(\mathbf{C})$;
    $\mathbf{C} \leftarrow \texttt{reshape}(\mathbf{C}, (n_x n_y, p, p))$;
    $\mathbf{C} \leftarrow f_{net}(\mathbf{C})$;
    $\mathbf{C} \leftarrow \text{reshape}(\mathbf{C}, (n_x, n_y, K))$;
    $\mathbf{C} \leftarrow \mathbf{C}[iS_x : (i+1)S_x, jS_y : (j+1)S_y]$;
    extra borders are removed;
    if $j = 0$ **then**
      $\mathbf{R} \leftarrow \mathbf{C}$;
    end
    else
      $\mathbf{R} \leftarrow \texttt{right-append}(\mathbf{R}, \mathbf{C})$;
    end
  end
  if $i = 0$ **then**
    $\mathbf{P} \leftarrow \mathbf{R}$;
  end
  else
    $\mathbf{P} \leftarrow \texttt{down-append}(\mathbf{P}, \mathbf{R})$;
  end
end
return P

of a single class assignment. Instead for each training image, we have an image whose pixel values show the class to which the corresponding pixel in the image is assigned to.

The artificially masked areas of the training images would interfere with the output of the model. To fix this, we treat pixels coming from that area as being from a different class. For a model seeking to discriminate between K classes, we end up having a label with $K + 1$ classes. This can be seen in Figure 5-8.

Just like the data for patch-net, the data here will consist of false negatives

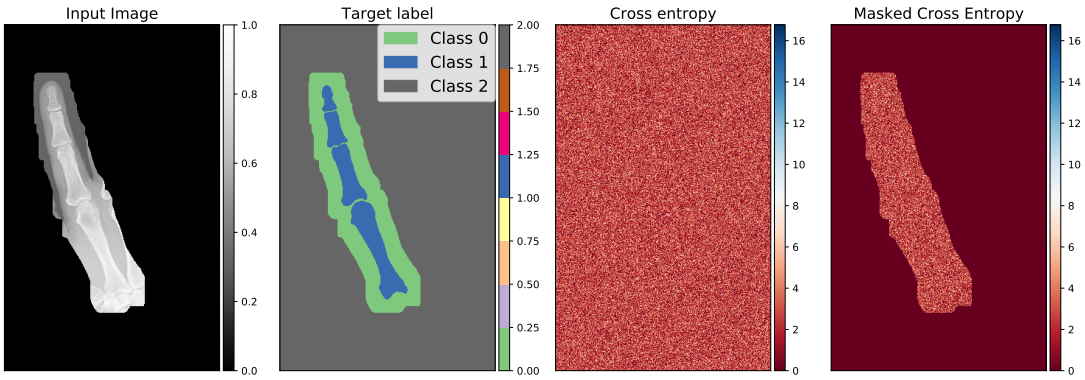


Figure 5-8: The Figure shows the input image, the target label, the cost map and the cost map after it has been masked by the background class (left to right) for U-net. Note that the cost map is shown at initialisation, which explains its salt and pepper appearance.

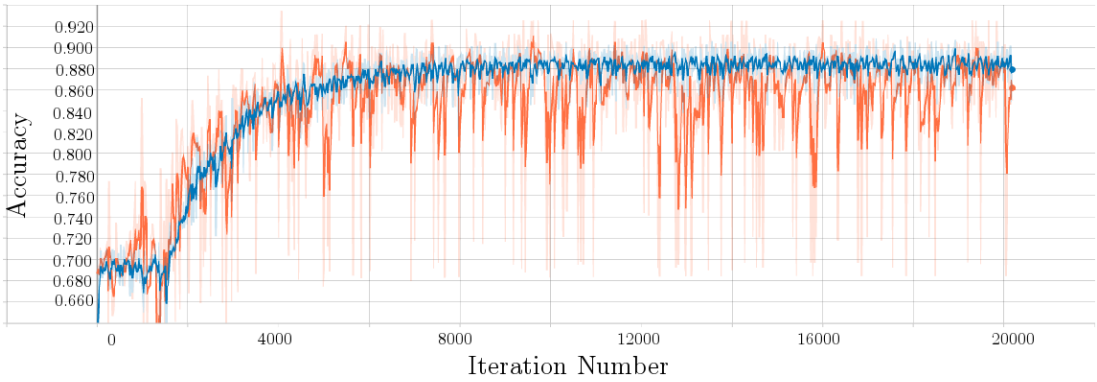


Figure 5-9: The Figure shows the accuracy for the U-net discriminators evolve as training progresses. The x-axis shows the stochastic gradient descent iteration number and the y-axis shows the accuracy at the corresponding step. We show the test set accuracy (blue) and the training set accuracy (red).

especially at the base of the index finger. As we do not have annotations for these areas, some parts of the bones in the metar carpal soup and from the adjacent fingers are wrongly labelled as not being bones.

5.5.2.2 Training

As previously mentioned, we do not want the model to learn features from the blurred out background. We modify the cost function we use so that this does not happen. We use the background label to create an image that has pixel values of ones for the area showing the finger and pixel values of zero for the area showing the masked background.

We then multiply this with the cost map. Pixels of the output map corresponding to the masked background hence have a zero contribution to the cost

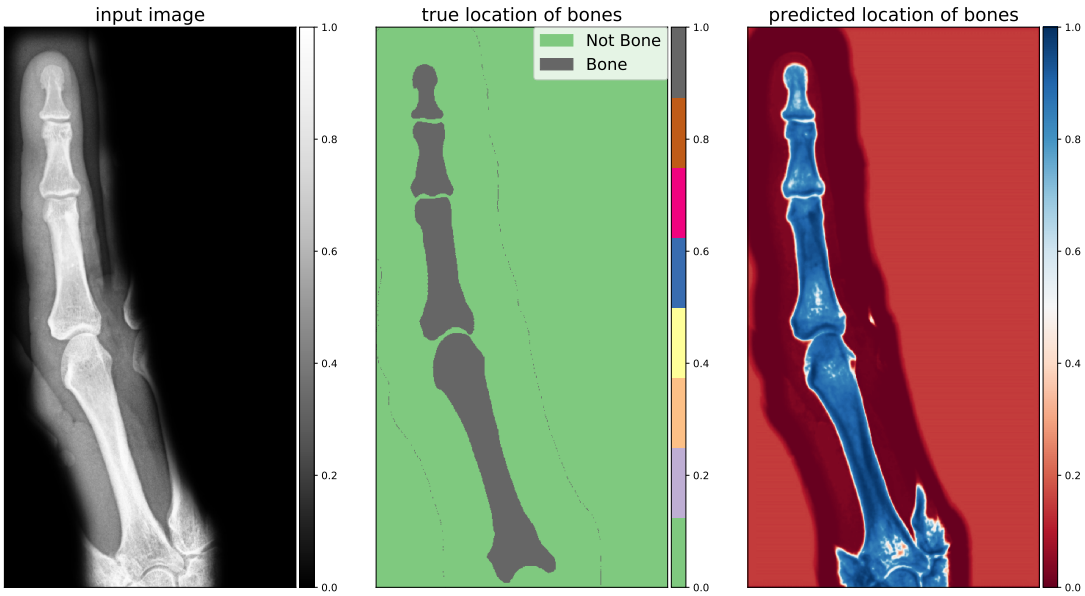


Figure 5-10: The Figure shows a test set input image, the true location of the bone in that image and the prediction for the bone location made by U-net (left to right).

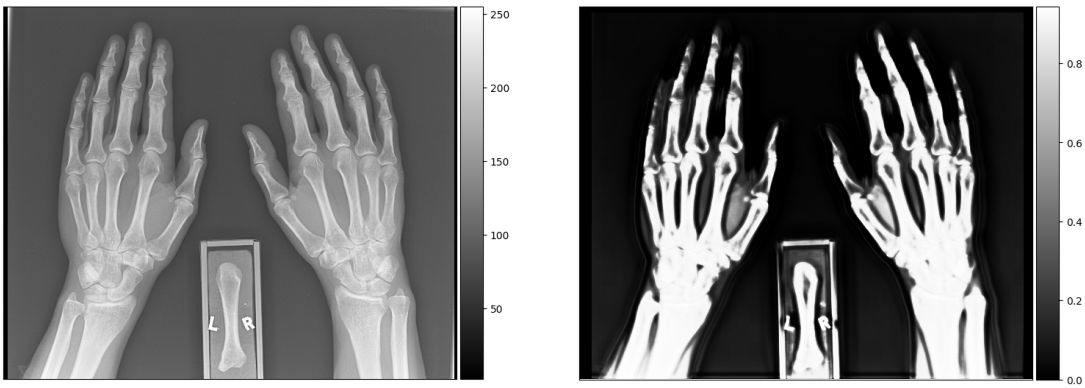


Figure 5-11: The Figure shows a full hand prediction for the U-net bone discriminator. The right plots shows the probability of a pixel being either a bone edge or a bone while the left plot shows the input image.

and they back propagate a gradient of zero. This ensures that the model does not learn anything about these pixels.

We use $M = 4$ downsampling layers with $C_1 = 16$ features for the first downsampling layer. We split the data into a training set and validation set. We use a momentum optimiser (see section A.2.0.6) with a momentum $\gamma = 0.8$ and a decaying learning rate. The learning rate decays with the epochs according to the folowing formula

$$\eta_i = \eta_0 r^i \tag{5.17}$$

where the r is the decay rate, η is the initial learning rate and i is the epoch

number. We use a batch size of 1 with $r = 0.95$ and $\eta = 0.01$. As described by Ronneberger et al. [2015], the high momentum ensures that information from the previous batch gets transferred to the current batch during training. We train the model for 300 epochs. We report a test accuracy of 88.64%. Figure 5-9 shows the trace plot for the accuracy.

To deal with images of different sizes during training, we can either resize them or pad them to the same dimensions. We found that both approaches worked well. The results we present use padded images.

5.5.2.3 Prediction

We want to see how well the model performs when run on full sized images. We show the predictions for the validation data used during training. The model predicts the bones locations for this set as shown in Figure 5-10. However, it does not perform as well for full sized images as shown in Figure 5-11. This is because it was not trained to discriminate full sized images. We believe that this problem will be solved for data which have a full set of annotations.

As opposed to patch-net, using U-net for prediction is a much faster operation. Moreover, the convolutional architecture allows prediction to be performed on images of arbitrary sizes. One problem though, is the fact that the image loses dimensions as it gets downsampled and upsampled again. The prediction maps hence need to be padded back to the original input dimensions when comparing it with the input.

5.5.3 Concluding remarks

In chapter 6, we want a potential that guides a shape model towards a likely candidate for that shape. The way we do this is inspired by the snakes model (Kass et al. [1988]). That is, we build V_u to model the areas that are likely to contain a bone edge. Given a bone region O we have that

$$\mathbb{P}(\boldsymbol{x} \in \partial O|u) \propto \exp\left(-\frac{1}{2}V_u(\boldsymbol{x})\right). \tag{5.18}$$

We have modelled $\mathbb{P}(\boldsymbol{x} \in \partial O|u)$ directly by using patch-net. Once the bone edge set Γ is found, we set V_u to be the distance transform D_Γ from this set. This is given by

$$D_\Gamma(\boldsymbol{x}) = \min_{y \in \Gamma} \|\boldsymbol{y} - \boldsymbol{x}\|_2 \tag{5.19}$$

However, finding the kernels that would cause V_u to model only the edge set of the bones is hard. This can be seen in Figure 5-7b where the bone edge discriminator also finds the edges around tissue. Hence, we sought to instead model $\mathbb{P}(\boldsymbol{x} \in O|u)$, through U-net. The hope was that, by finding the outlines of the bone regions, we would remove the contribution of the tissue edge set from the potential V_u.

The incomplete training data that we have does not allow us to do this for the full hand. Moreover, unless trained with a weight applied on the edge set of the image (see Ronneberger et al. [2015]), U-net does a bad job of finding the gaps between the bones. The potential that we then build from the output of U-net is more useful as a global initialisation for a shape matching model. Once the templates are initialised, the output from an edge detector such as patch-net can then be used to fit the shape model to a new image. We describe how this is done in more detail in section 6.4.

Chapter 6

Bone identification through shape modelling

6.1 Introduction

Bone identitification forms an integral part of our system. We present in this chapter a hybrid model that uses a discriminator and a shape generator to identify bones in a hand X-ray. The literature has a lot of such algorithms. We inspire ours from the snake model and from its extension the Active Shape model of Cootes et al. [1995].

Active Shape Models involves the use of a statistical model to regularise a segmentation task taking the form of (3.10). Later, a texture model was incoporated into the model as an extension to the potential V_u (the data comparison term). The joint model was called an Active Appearance Models (AAM) Cootes et al. [2001]. Shape and texture are broken into linear combinations of a mean and variation modes which are learned from training examples of aligned images. These have been used in facial recognition tasks inEdwards et al. [1998]. A weakness of AAMs is that they require the alignment of features in training examples. Kruger et al. [2015] performs the training and alignment in one optimisation.

The strengths of Shape models in segmentation tasks lies in the fact that they can capture the modes of variability of shapes very well. For that to happen, the training data needs to only exhibit variation due to shape and not due to relative and absolute spatial locations. This means that AAMs, on their own struggle to model compound objects. AAMs can be combined to form a compound AAM as in Contrained Local Models (CLMs) Cristinacce and Cootes [2006] which themselves are an extension to Felzenszwalb and Huttenlocher [2005]. Minciullo et al.

[2018] shows a successful application of all these techniques in a medical imaging segmentation task.

We introduce the Active Latent Space Shape Model (ALSSM). This uses a a mapping from a latent space to generate shape examples. Our model incorporates uncertainty from the shape generation process, the shape matching process and the image to be segmented. We develop a fully bayesian framework that allows us to marginalise out each step of the fitting procedure. We make our case for using a discriminator as the data comparison term or potential V_u, which turns our model into a hybrid one.

We describe the dependence structure within the model in section 6.2. We then describe the latent space generation process in section 6.2.1, where we justify a move away from traditional linear decomposition of shapes. We show how we can have a probabilistic interpretation of the data term in section 6.2.2.

Our main contribution is in section 6.2.4, where we approximate the data term using radial basis functions and perform the full bayesian marginalisation. This yields an objective function in section 6.2.5 which we can optimise to fit the model. We discuss the approach we have taken and the contributions we make in section 6.3. In particular, we show why the approximation we use is justified when considering the errors it introduces.

We show in section 6.4 how one can use a coarse to fine strategy to partially solve the problem of initialising pose parameters. We finish the chapter by showing how we can interpret the AAM (Cootes et al. [2001]) to be part of the ALSSM framework in section 6.5.

6.2 Active Latent Space Shape Model

In what follows, we will use the following notation:

(a) $\mathrm{X} \in \mathbb{R}^Q$ represents the latent space position of a shape example generated from a latent space representation of shape (shape model).

(b) $\mathbf{F} = (\mathrm{F}_0, .., \mathrm{F}_{D-1}) \in \mathbb{R}^P$, with $\mathrm{F}_d = (\mathrm{F}_d^{(x)}, \mathrm{F}_d^{(y)})$, is the vector of representing the shape outline generated from a shape model such as a GPLVM. Here $P = 2D$ where D is the discretisation used on the curve outlines when training the shape model.

(c) $\mathbf{M} \in \mathbb{R}^P$ is the observed shape in an image. It is given by $\mathbf{M} = \mathbf{F} + \boldsymbol{\eta}$ where $\boldsymbol{\eta}$ is a white noise process. Similar to $\mathbf{F}$, we have that $\mathrm{M}_d = (\mathrm{M}_d^{(x)}, \mathrm{M}_d^{(y)})$ are the coordinates of the d-th point on a discretised curve.

(d) $\mathrm{T} = (r, \psi, d_x, d_y)$ represents the pose parameters. It defines a similarity transform on $\mathbb{R}^2$.

(e) u is the observed image.

Building on the DAG in Figure 6-1, we define a new class of shape models which we call Active Latent Space Shape Models. A shape $\mathbf{F}$ is generated from a latent space X. This generative process is captured by a conditional density $p(\mathbf{F}|\mathrm{X})$. $p(\mathbf{M}|u, \mathbf{F}, \mathrm{T})$ specifies the mapping of the generated shape onto observable real world examples of such shapes that appears in an image u.

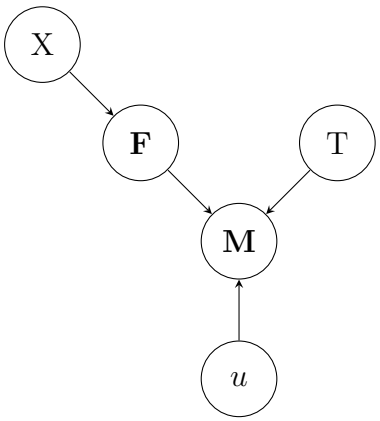


Figure 6-1: The Figure shows how the shape $\mathbf{M}$ depends on the shape model $\mathbf{F}$ and its prior X. We also assume that shape is generated from the observed image u

By imposing in Figure 6-1 that $\mathbf{M}$ depends on the model generated shape $\mathbf{F}$ and the image u, we are looking for the model generated shape that fits some candidate shape in the image u. Equivalently, we are looking for the model generated shape that best matches a candidate shape from the image. We simplify this problem further by making the following assumption

$$p(\mathbf{M}|u, \mathbf{F}, \mathrm{T}) = p(\mathbf{M}|\mathbf{F}, \mathrm{T})p(\mathbf{M}|u) \tag{6.1}$$

This splits the inference into 2 parts. On one part, we have the shape model defined by the prior $p(\mathbf{M}|\mathrm{X})$ and on the other, we have the data term $p(\mathbf{M}|u)$ which specifies the type of interaction between the generated shape $\mathbf{M}$ and the image u. More importantly though, such a factorisation along with the introduction of $p(\mathbf{M}|\mathbf{F})$ allows us to specify a fitting procedure where the uncertainty at each step can be quantified.

6.2.1 Latent Space Representation

$p(\mathbf{F}|\mathrm{X})$ has two purposes. The first one is to quantify the uncertainty in the generated shape given a value for the latent space parameters X. We use a Gaussian model for this purpose, with mean $\boldsymbol{\mu}(\mathrm{X}) \in \mathbb{R}^P$ and diagonal covariance matrix given by $\mathrm{K}(\mathrm{X})\mathbf{I}_P \in \mathbb{R}^{P\times P}$ depending on the latent variable X. The uncertainty we have just mentioned is captured by K(X). That is, the higher the variance of a given shape, the less likely it is to be a valid one.

The second purpose is to define the nature of the mapping from the latent space onto the space of shapes. We treat $\boldsymbol{\mu}(\mathrm{X})$ to be this mapping. The most likely candidate for this is a statistical shape model. Cootes et al. [1995] uses a PCA decomposition in the Active Shape Model (see section 4.3). Such a decomposition has been shown to be a very good candidate in the literature. We, however, choose to use a GPLVM as our statistical shape model as we believe that they are a better candidate when viewed from a model specification point of view. We make our case for this choice here.

We have previously expressed the need for a latent space dependent distribution on the space of shapes in section 4.3.2. Note that when using a GPLVM, the shape generation uncertainty is already part of the model. Were we to use the PCA decomposition of Cootes et al. [1995], we would have to model this uncertainty through further analysis of the distribution of the weights[1] ω_q in equation (4.4) to come up with a sensible latent space indexed variance K(X).

GPLVMs also relaxes the linearity constraints on the shape generation process when viewed as a mapping from the latent space. We show in section 4.3.3 that when one considers the expression for the posterior mean of a GPLVM, one realises that it is in fact a linear combination of data examples. This seems to counter our claim about a GPLVM being able to model more complex mappings. At this point, we refer the reader to equation (4.5) where the map is from the latent space and not from data space. Moreover, it is clear from equation (6.2) that the shape generation process happens via some non-linear transformation on the latent space X.

We again stress that our arguments come from model specification considerations. We do not claim that a GPLVM will necesarily deliver superior results to a PCA decomposition of shape space. In fact, the results shown by Cristinacce and

[1] these weights are the equivalent of a latent space in the ASM

Cootes [2006] suggests that ASMs are very powerful tools when used in medical imaging.

Now that we have justified the use of a GPLVM as our shape prior, let us define the form that the latter takes. As previously described, a GPLVM automatically induces a multivariate Gaussian distribution $p(\mathbf{F}|\mathrm{X})$. In particular, under the setting of section 4.4 the predictive density of a GPLVM with covariance kernel k that has been trained on shape data $\mathbf{Y} \in \mathbb{R}^{N\times P}$ is a multivariate normal $N(\mathbf{F}|\boldsymbol{\mu}(\mathrm{X}), \mathrm{K}(\mathrm{X})\mathbf{I})$ where

$$\begin{aligned} \mathrm{K}(\mathrm{X}) &= k(\mathrm{X},\mathrm{X}) - \mathrm{K}(\mathbf{t},\mathrm{X})^T\mathbf{K}^{-1}\mathrm{K}(\mathbf{t},\mathrm{X}) \\ \boldsymbol{\mu}(\mathrm{X}) &= \mathrm{K}(\mathbf{t},\mathrm{X})^T\mathbf{K}^{-1}\mathbf{Y} \end{aligned} \tag{6.2}$$

$\mathbf{t} \in \mathbb{R}^{N\times Q}$ is the matrix of latent parameters whose n-th row $\boldsymbol{t}$ is the latent space value of the $n-$th training example. $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the training data covariance matrix where the entries are given by $k(\boldsymbol{t}_i, \boldsymbol{t}_j)$. $\mathrm{K}(\mathbf{t},\mathrm{X}) \in \mathbb{R}^N$ is the vector having entries $k(\boldsymbol{t}_i, \mathrm{X})$.

6.2.2 Data term

The observed image has many candidates for shapes that can be extracted from it. This relationship is captured by $p(\mathbf{M}|u)$. This term is analogous to the potential function V_u in equation (3.10). In the model that we present in this thesis, we set

$$p(\mathbf{M}|u) \propto \exp\left(-\frac{1}{D}\sum_{d=0}^{D-1} V_u(\mathrm{M}_d)\right) \tag{6.3}$$

We have that

$$V_u(\boldsymbol{x}) = \min_{\boldsymbol{y}\in\Gamma} \|\boldsymbol{x}-\boldsymbol{y}\| \tag{6.4}$$

where Γ is some set in the image domain whose texture resembles that of an object boundary. We describe how we can use discrminative texture models (see Chapter 5) to build the function V_u in section 6.4.1. Note that V_u has support in the image domain $\Omega \subset \mathbb{R}^2$.

Equation (6.3) is similar to the data term used by Cristinacce and Cootes [2006]. However, instead of specifying a distribution for $p(\mathbf{M}|u)$, Cristinacce and Cootes [2006] provides a Gibbs type distribution for $p(u|\mathbf{M})$. Computationally,

this does not change the marginalisation that we perform next. However, in our case, the form of the function V_u depends on the image, and so, from a conceptual point of view, it is more appropriate to define $p(\mathbf{M}|u)$.

6.2.3 Setting

Hence, our shape model consists of a shape $\mathbf{M}$ which is controlled by a shape parameter $\mathbf{F}$. $\mathbf{F} = [F_0, .., F_{2D-1}]$ is itself the realisation of a conditional multi-output Gaussian Process that comes from the fitting of a GPLVM on the curve outlines delineating bone regions to be segmented. The shape parameters X are the latent space values of this GPLVM. The pose parameters $\mathrm{T} = (r, \psi, d_x, d_y)$ define a rigid transformation $\mathrm{T} \circ \mathbf{M}$ on $\mathbf{M}$ given by

$$T \circ (\mathrm{M}_d^{(x)}, \mathrm{M}_d^{(y)}) = (r * \mathrm{M}_d^{(x)} \cos\psi - \mathrm{M}_d^{(y)} \sin\psi, r * \mathrm{M}_d^{(x)} \sin\psi + \mathrm{M}_d^{(y)} \cos\psi) + (d_x, d_y) \tag{6.5}$$

The DAG is characterised by the following conditional densities:

- $p(\mathbf{M}|u) \propto \exp(-\sum_{d=0}^{D-1} V_u(\mathrm{M}_d))$
- $p(\mathbf{M}|\mathbf{F}, \mathrm{T}) = \frac{1}{\sqrt{2\pi\beta^2}} \exp\left(-\frac{\|T \circ \mathbf{M} - \mathbf{F}\|^2}{2\beta^2}\right)$
- $p(\mathbf{F}|\mathrm{X}) = \frac{1}{\sqrt{2\pi \mathrm{K}(\mathrm{X})}} \exp\left(-\frac{\|\mathbf{F} - \boldsymbol{\mu}(\mathrm{X})\|^2}{2\mathrm{K}(\mathrm{X})}\right)$ (from equation (6.2))

We set the latent space priors and shape priors to be Dirac functions. We can consider them to be hyperparameters of our model which need to be found. The curve points M_j can only be defined inside the image domain Ω and hence, $p(\mathbf{M}|u)$ has support in Ω^D, which is a bounded $2D$-dimensional cube in $\mathbb{R}^{2D}$.

6.2.4 Marginalisation and model fitting

We wish to fit the model to a new data point (image) u. This is achieved by maximising

$$\log p(u; \mathrm{X}, \mathrm{T}) = \log \int_{\Omega^D} \int_{\mathbb{R}^P} p(u, \mathbf{M}, \mathbf{F}; \mathrm{X}, \mathrm{T}) \mathrm{d}\mathbf{F} \mathrm{d}\mathbf{M} \tag{6.6}$$

with respect to X, T. We have that

$$
\begin{aligned}
&\int_{\Omega^D}\int_{\mathbb{R}^P} p(u,\mathbf{M},\mathbf{F})\mathrm{d}\mathbf{F}\mathrm{d}\mathbf{M} \\
&=\int_{\Omega^D}\int_{\mathbb{R}^P} p(\mathbf{M}|u)p(\mathbf{M}|\mathbf{F};\mathrm{T},\mathrm{X})p(\mathbf{F};\mathrm{X})\mathrm{d}\mathbf{F}\mathrm{d}\mathbf{M} \\
&=\int_{\Omega^D} p(\mathbf{M}|u)\left[\int_{\mathbb{R}^P} p(\mathbf{M}|\mathbf{F};\mathrm{T},\mathrm{X})p(\mathbf{F};\mathrm{X})\mathrm{d}\mathbf{F}\right]\mathrm{d}\mathbf{M}
\end{aligned}
\tag{6.7}
$$

Using the fact that for $D-$dimensional vector, $\|\boldsymbol{a}\|^2 = \sum_{d=0}^{D-1} a_j^2$, we have that we can integrate in a component wise fashion. To integrate out $\mathbf{F}$, we look at its components and without loss of generality, we can re-index $\mathrm{T}\circ\mathbf{M} = [T\circ \mathrm{M}_0, ..., T\circ\mathrm{M}_{2D-1}]$. Then the $d-$th component of the integral is given by:

$$
\begin{aligned}
&\int_{\mathbb{R}} \mathcal{N}\left(F_d|\boldsymbol{\mu}(\mathrm{X})_d, \mathrm{K}(\mathrm{X})\right)\mathcal{N}\left(F_d|T\circ\mathrm{M}_d,\beta^2\right)\mathrm{d}F_d \\
&=\int_{\mathbb{R}} \mathcal{N}\left(F_d|\frac{\boldsymbol{\mu}(\mathrm{X})_d\beta^2 + T\circ\mathrm{M}_d\mathrm{K}(\mathrm{X})}{\mathrm{K}(\mathrm{X})+\beta^2}, \frac{\beta^2\mathrm{K}(\mathrm{X})}{\mathrm{K}(\mathrm{X})+\beta^2}\right)\mathcal{N}\left(\mathrm{M}_d|T^{-1}\circ\boldsymbol{\mu}(\mathrm{X})_d, \beta^2+\mathrm{K}(\mathrm{X})\right)\mathrm{d}F_d \\
&\quad \text{using Lemma A.1} \\
&=\mathcal{N}\left(\mathrm{M}_d|T^{-1}\circ\boldsymbol{\mu}(\mathrm{X})_d, \beta^2+\mathrm{K}(\mathrm{X})\right)
\end{aligned}
\tag{6.8}
$$

Gathering the components of the integral back, equation (6.7) becomes

$$
\begin{aligned}
&\int_{\Omega^D} p(\mathbf{M}|u)\left[\int_{\mathbf{F}} p(\mathbf{M}|\mathbf{F};\mathrm{T},\mathrm{X})p(\mathbf{F};\mathrm{X})\mathrm{d}\mathbf{F}\right]\mathrm{d}\mathbf{M} \\
&=\int_{\Omega^D} p(\mathbf{M}|u)\mathcal{N}\left(\mathbf{M}|\mathrm{T}^{-1}\circ\boldsymbol{\mu}(\mathrm{X}), \left(\beta^2+\mathrm{K}(\mathrm{X})\right)\mathbf{I}\right)\mathrm{d}\mathbf{M}
\end{aligned}
\tag{6.9}
$$

The term $p(\mathbf{M}|u)$ makes the above intractable, however we note that it can be approximated well by an RBF expansion Radial Basis Function (RBF). $p(\mathbf{M}|u)$ is a function with domain $\Omega\in\mathbb{R}^{2D}$. We therefore use RBF kernels

$$
\phi_l(\mathbf{M}) = \frac{1}{\sqrt{2\pi v^2}}\exp\left(-\frac{\|\mathbf{M}-\boldsymbol{p}_l\|^2}{2v^2}\right)
$$

centred around $\boldsymbol{p}_l\in\mathbb{R}^{2D}$ with scalar spread v to build the following interpolant:

$$
\mathcal{I}_p(\mathbf{M}) = \sum_{l=0}^{L-1} w_l\phi_l(\mathbf{M}) \tag{6.10}
$$

where w_l are the weights and $\boldsymbol{p}_l$ are the nodes we describe in Section 3.4. This approximation allows us to turn the integral in (6.9) into one consisting of products

of Gaussians, thus making it tractable as follows:

$$\begin{aligned}
&\int_{\Omega^D} \mathcal{I}_p(\mathbf{M}) \mathcal{N}\left(\mathrm{M}_j \middle| \mathrm{T}^{-1} \circ \boldsymbol{\mu}(\mathrm{X})_j, \beta^2 + \mathrm{K}(\mathrm{X})\right) \mathrm{d}\mathbf{M} \\
=&\sum_{l=0}^{L-1} w_l \int_{\Omega^D} \mathcal{N}\left(\mathbf{M} \middle| \boldsymbol{p}_l, v^2\mathbf{I}\right) \mathcal{N}\left(\mathbf{M} \middle| T^{-1} \circ \boldsymbol{\mu}(\mathrm{X})_j, \left(\beta^2 + \mathrm{K}(\mathrm{X})\right)\mathbf{I}\right) \mathrm{d}\mathbf{M} \\
=&\sum_{l=0}^{L-1} \tilde{w}_l \mathcal{N}\left(\mathrm{T}^{-1} \circ \boldsymbol{\mu}(\mathrm{X}) \middle| \boldsymbol{p}_l, \left[\beta^2 + \mathrm{K}(\mathrm{X}) + v^2\right]\mathbf{I}\right) \\
=&\sum_{l=0}^{L-1} \frac{\tilde{w}_l}{\sqrt{2\pi(v^2 + \beta^2 + \mathrm{K}(\mathrm{X}))}} \exp\left(-\frac{\|\mathrm{T}^{-1}\boldsymbol{\mu}(\mathrm{X}) - \boldsymbol{p}_l\|^2}{2(v^2 + \beta^2 + \mathrm{K}(\mathrm{X}))}\right)
\end{aligned} \tag{6.11}$$

where, using Lemma A.1,

$$\tilde{w}_l = w_l \int_{\Omega^D} \mathcal{N}\left(\mathbf{M} \middle| \frac{\boldsymbol{p}_l(\beta^2 + \mathrm{K}(\mathrm{X})) + \mathrm{T}^{-1}\boldsymbol{\mu}(\mathrm{X})v}{v^2 + \beta^2 + \mathrm{K}(\mathrm{X})}, \frac{v^2(\beta^2 + \mathrm{K}(\mathrm{X}))}{v^2 + \beta^2 + \mathrm{K}(\mathrm{X})}\right) \mathrm{d}\mathbf{M} \tag{6.12}$$

is the coefficient of the $l-$th basis function of the RBF expansion of $p(\mathbf{M}|u)$ reweighted by the Gaussian measure of the region Ω. We note that we can make Ω^D arbitrarily big by simply extending the domain of $p(\mathbf{M}|u)$ beyond the image domain. This would force its Gaussian Measure to get closer and closer to 1. We can then assume without loss of generality that $\tilde{w}_l = w_l$.

6.2.5 Objective function

Equation (6.11) yields the following objective function to be minimised

$$L(\mathrm{X}, \mathrm{T}) = \sum_{l=0}^{L-1} \frac{w_l}{\sqrt{2\pi(v^2 + \beta^2 + \mathrm{K}(\mathrm{X}))}} \exp\left(-\frac{\|\mathrm{T}^{-1}\boldsymbol{\mu}(\mathrm{X}) - \boldsymbol{p}_l\|^2}{2(v^2 + \beta^2 + \mathrm{K}(\mathrm{X}))}\right) \tag{6.13}$$

where we have ommitted the weighting given in equation (6.12) of the basis function weights w_l. This is equivalent to a convolution of the form

$$l(\mathrm{X}, \mathrm{T}) = \mathcal{N}(0, \left(\beta^2 + \mathrm{K}(\mathrm{X})\right)\mathbf{I}) \star \mathcal{I}_p(\mathrm{T}^{-1}\boldsymbol{\mu}(\mathrm{X})) \tag{6.14}$$

which we can approximate with

$$l(\mathrm{X}, \mathrm{T}) = \mathcal{N}(0, \left(\beta^2 + \mathrm{K}(\mathrm{X})\right)\mathbf{I}) \star p(\mathbf{M}|u)\Big|_{\mathbf{M}=\mathrm{T}^{-1}\boldsymbol{\mu}(\mathrm{X})} \tag{6.15}$$

Fitting the shape model to a new image is thus similar to fitting the snakes model as described in Section 3.2.3. We evolve a discretised curve $\mathbf{M}$ on a potential V_u.

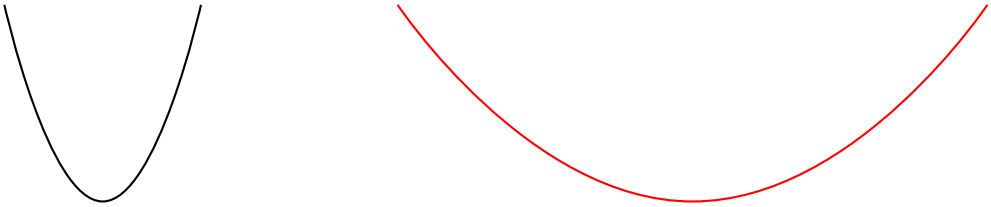


Figure 6-2: The black curve (left) is a local minimum of $p(\mathbf{M}|u)$. After Gaussian smoothing, the gradient around the local minimum has been decreased as shown on the red curve(right).

However, the optimisation takes place with respect to the latent space parameters that regularise the shape of the curve as opposed to the curve points directly.

6.3 Discussion

6.3.1 Uncertainty propagation

We have presented a model where the shape is regularised by a GPLVM constraint. We have split the uncertainty into different parts by introducing $p(\mathbf{M}|\mathbf{F})$ and $p(\mathbf{F}|\mathrm{X})$. In particular, we are able to control how certain we are about the effect of the shape model by varying the hyperparameter β.

In fact, during fitting, the uncertainty from each part of our model gets transferred into the final objective function through gaussian blurring. This blurring causes the gradient around a local minimum of $\exp(-V_u)$ to decrease in magnitude as shown in Figure 6-2. Adopting the snakes formulation of a segmentation problem, this blurring allows for more variation in the shape of the curve minimising the snakes energy. Most of this behaviour in our model is controlled by β in $p(\mathbf{M}|\mathbf{F}, \mathrm{T})$ which controls the balance between the observed data and the learned space of shapes in our minimisation.

$p(\mathbf{M}|\mathbf{F})$ takes the form of Gaussian noise, and hence, draws from this distribution will lack the smoothness that we expect in shapes appearing in images. However, this does not affect the fitting of the model. Moreover, in the fitting procedure, we are only interested in the posterior mean of the GPLVM shape model. This is smooth as the kernel that we use is a smooth one. Comparing this to the snakes algorithm, it is this smoothness on the mean shape $\boldsymbol{\mu}(\mathrm{X})$ that replaces the smoothness term in equation (3.10).

6.3.2 Full Bayesian Marginalisation

The typical way in the literature to marginalise intractable integrals such as the one in equation (6.9) is to use a variational distribution. The main guarantees of such an approximation given in the literature relies on the approximation being a lower bound on the integral. As far as we know, no estimates of the error are given. The RBF approximation we make in our marginalisation has error estimates that are well studied, which are presented in section 3.4.

Moreover, variational integration usually results in an objective function that differs from the original marginal data likelihood. This happens through the introduction of a new distribution q that is different from the marginals of the model DAG. In our case, we get an objective function that only depends on the original formulation of the model DAG.

6.3.3 Marginalisation Error Estimate

Let

$$\begin{aligned} f : \Omega^D \subseteq \mathbb{R}^P &\to \mathbb{R} \\ \boldsymbol{x} &\mapsto p(\mathbf{M}|u)\Big|_{\mathbf{M}=\boldsymbol{x}} \end{aligned}$$

be compactly supported and let $\mathcal{I}_p$ be the interpolant described in equation (6.10). Let

$$h_{\Omega^D,\mathcal{X}} := \sup_{\boldsymbol{x}\in\Omega^D} \min_{\boldsymbol{p}\in\mathcal{X}} \|\boldsymbol{x}-\boldsymbol{p}\|_2$$

where $\mathcal{X}$ is the set of interpolating nodes $\{\boldsymbol{p}_0, ..., \boldsymbol{p}_{L-1}\}$. We have the following bound from Fröhlich [2013]

$$\max_{\boldsymbol{x}\in\Omega^D} \left|\mathcal{I}_p(\boldsymbol{x}) - f(\boldsymbol{x})\right| \leq \exp\left(-\frac{\log(h_{\Omega^D,\mathcal{X}})}{h_{\Omega^D,\mathcal{X}}}\right) \tag{6.16}$$

Using this, we have that for any normal density $\mathcal{N}(\boldsymbol{x})$

$$
\begin{aligned}
&\left| \int_{\Omega^D} \mathcal{I}_p(\boldsymbol{x}) \mathcal{N}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} - \int_{\Omega^D} f(\boldsymbol{x}) \mathcal{N}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \right| \\
\leq & \int_{\Omega^D} \left| \left(\mathcal{I}_p(\boldsymbol{x}) - f(\boldsymbol{x}) \right) \mathcal{N}(\boldsymbol{x}) \right| \mathrm{d}\boldsymbol{x} \\
\leq & \max_{\boldsymbol{x} \in \Omega^D} \left| \mathcal{I}_p(\boldsymbol{x}) - f(\boldsymbol{x}) \right| \int_{\Omega^D} \mathcal{N}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\
& \text{using Holder's inequality} \\
\leq & C \exp\left(- \frac{\log(h_{\Omega^D, \mathcal{X}})}{h_{\Omega^D, \mathcal{X}}} \right)
\end{aligned}
\tag{6.17}
$$

Hence we get a bound on approximating the integral in equation (6.9) using an RBF interpolant. We now show that we get a bound for the approximation of (6.14) by (6.15). If we now set the mean of the normal density $\mathcal{N}(\boldsymbol{x})$ to be zero, we have for any $\boldsymbol{y} \in \Omega$

$$
\begin{aligned}
&\left| f \star \mathcal{N}(\boldsymbol{y}) - \mathcal{I}_p \star \mathcal{N}(\boldsymbol{y}) \right| \\
= & \left| \int_{\Omega^D} \mathcal{I}_p(\boldsymbol{x}) \mathcal{N}(\boldsymbol{x} - \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} - \int_{\Omega^D} f(\boldsymbol{x}) \mathcal{N}(\boldsymbol{x} - \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} \right| \\
\leq & \int_{\Omega^D} \left| \left(\mathcal{I}_p(\boldsymbol{x}) - f(\boldsymbol{x}) \right) \mathcal{N}(\boldsymbol{x} - \boldsymbol{y}) \right| \mathrm{d}\boldsymbol{x} \\
\leq & \max_{\boldsymbol{x} \in \Omega^D} \left| \mathcal{I}_p(\boldsymbol{x}) - f(\boldsymbol{x}) \right| \int_{\Omega^D} \mathcal{N}(\boldsymbol{x} - \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} \\
& \text{using Holder's inequality} \\
\leq & C \exp\left(- \frac{\log(h_{\Omega^D, \mathcal{X}})}{h_{\Omega^D, \mathcal{X}}} \right)
\end{aligned}
\tag{6.18}
$$

where C is a constant depending on f and whose form is given in Lemma 3.7. Equations (6.17) and (6.18) both give us guarantees about the error we get in the approximation we make in the marginalisation and the subsequent use of the smoothed data likelihood as the objective function. In particular, we are able, through the right choice of interpolating point grid as described in Lemma 3.9 to send the error in these approximations to zero.

The error estimates for the RBF interpolation of $p(\mathbf{M}|u)$ rely heavily on its compact support. However, we seem to violate this assumption in equation (6.12). We show in Lemma 3.9 that for any cube domain in $\mathbb{R}^P$ we can find a good approximation to $p(\mathbf{M}|u)$. Hence, even though the size of Ω^D is increasing, we can still find a good interpolant, provided Ω^D stays compact.

Algorithm 4: Multi-level fitting of a compound ALSSM. The objective function E is given by equation (6.19).

Multi level ALSSM Fitting;
Input : Image u_0
Number of levels H and number of models per level n_h
Initialised pose and latent space parameters
$\mathrm{X}_l^0, \mathrm{T}_l^0, l = 0, .., n_0$
Output: $\mathrm{X}_l^{H-1}, \mathrm{T}_l^{H-1}, l = 0, .., n_{H-1}$
for $h = 0$ **to** $H - 1$ **do**
  $\{\mathrm{X}_l^*, \mathrm{T}_l^* : l = 0, .., n_h - 1\} \longleftarrow \arg\min_{\Theta,Q} E(\mathrm{X}, \mathrm{T})$;
  if $h \neq H - 1$ **then**
    Extract $\{\mathrm{T}_l^{h+1} : l = 0, ..., n_{h+1} - 1\}$ from $\{\mathrm{T}_l^* : l = 0, ..., n_h\}$
  else
    $\mathrm{T}_l^{H-1} \longleftarrow \mathrm{T}_l^*$;
    $\mathrm{X}_l^{H-1} \longleftarrow \mathrm{X}_l^*$;
  end
end

We stress that the approximation we make allows us to marginalise out $\mathbf{M}$, but plays no role in the numerical fitting of the model. Hence, we never have to compute any parameters of the RBF interpolants in equation (6.10). In fact, in practice, we minimise

$$p(\mathbf{M}|u)\Big|_{\mathrm{T}^{-1}\boldsymbol{\mu}(\mathrm{X})}$$

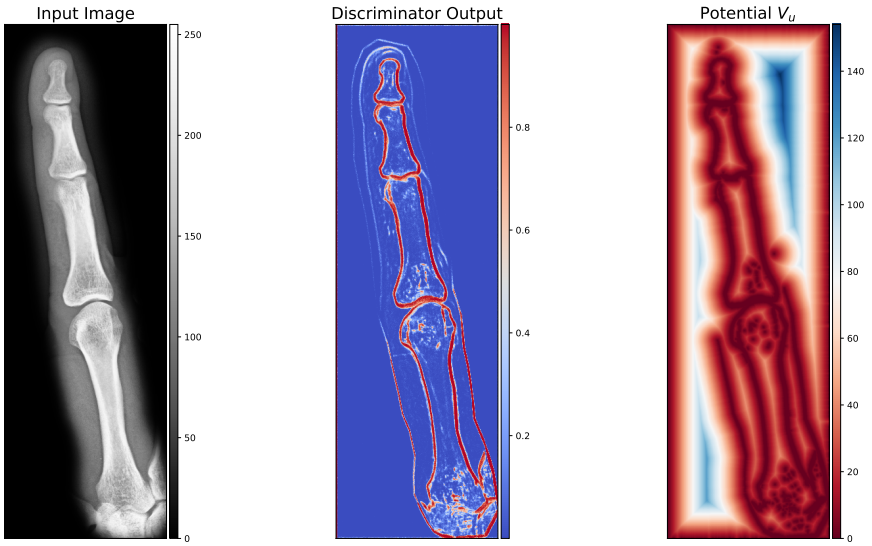
with respect to T, X. We are able to do this because V_u is a functional of the output of a discriminative texture model which we are able to evaluate as we later describe in section 6.4.1. Furthermore, we are able to calculate numerical gradients of V_u very easily, which experiments have shown to be a good approximation of the true gradient.

6.4 Experimental Results

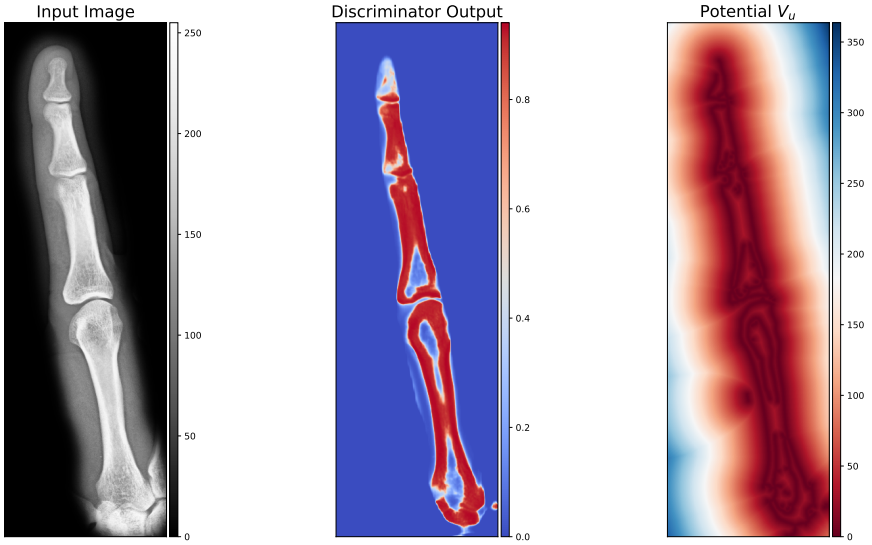
The main aim of this thesis is to apply computer vision and machine learning techniques to the problem of PsA diagnosis. Hence, our dataset comes from PsA hand radiographs. Given a GPLVM and a discriminator that were both trained on PsA X-rays, we minimise $-log\left(p(\mathbf{M}|u)\right)$. That is, our objective function is given by

$$E(\mathrm{X}, \mathrm{T}) := \frac{1}{D} \sum_{i=0}^{D-1} V_u\left(\mathrm{T}^{-1}\boldsymbol{\mu}(\mathrm{X})_i\right) \tag{6.19}$$

This is different from equation (6.15) as it is missing the variance terms K(X)



(a) Potential built from p-net output.



(b) Potential built from u-net output.

Figure 6-3: The Figures shows the distance transform from the edge set that p-net and u-net find. We set V_u to be this distance transform. The left-most figure is the masked input image, the middle figure is the probability map generated by the two discriminators and the left most map is the distance transform from the edge-set.

and β^2. This is because in practice, the numerics perform better on our dataset when these are ommitted. In particular, because of the small number of data points available, using the variance term K(X) forces the latent space parameters to stay close to big data clusters. This prevents the algorithm from exploring new shapes when it comes to fitting the model to a new example.

As mentioned earlier, the parameter β balances the uncertainty between the data term V_u and the model generated shape $\mathbf{F}$. In the absence of K(X), β is

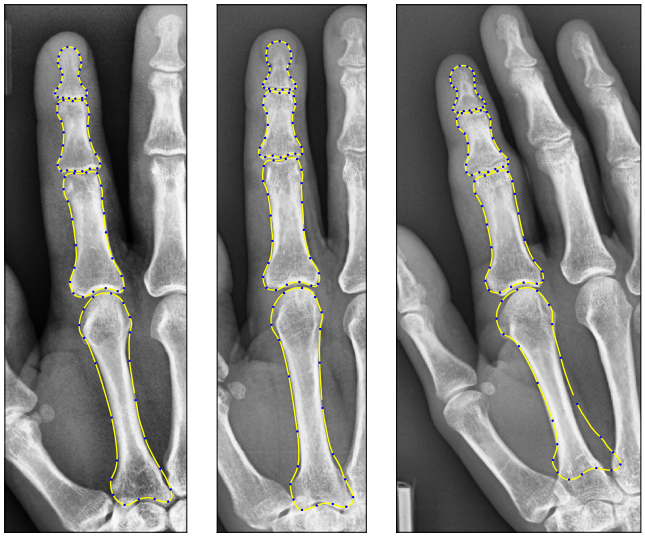


Figure 6-4: The Figures show the result of fitting an ALSSM when using a GPLVM that models the compound interactions of the bones in the right index finger. The bone outlines have the correct pose. The pose parameters for each bone is extracted and used as initialisation in ALSSMs where a GPLVM models the shape of that respective bone.

only a scaling factor in the objective function in equation (6.19). It would affect the model fiting were we to optimise equation (6.15) due to the presence of the exponential term.

Sending β to zero has significance in the marginalisation of $\mathbf{F}$ in equation (6.9). This essentially turns $p(\mathbf{M}|\mathbf{F})$ into a Dirac centred around $\mathbf{F}$. From a model specification however, we maintain that the Gaussian $p(\mathbf{M}|\mathbf{F})$ should be kept as it is a more realistic representation of the uncertainties present in a segmentation task.

In the following sections, we run our model on unlabelled masked images of the right hand index. This is because most of the models we use in this thesis draw their training data from this region of the hand. It would be unfair for us to expect good results on the whole hand.

6.4.1 Data term V_u

We run p-net and U-net on our test data and build potentials V_u as described in section 5.5.3. Let Γ be the edge set of bone regions. Then we have that

$$V_u(\boldsymbol{x}) = \min_{y \in \Gamma} \|\boldsymbol{y} - \boldsymbol{x}\|_2 \tag{6.20}$$

The output of p-net shows the location of the edges directly. We find the edge set Γ by simple thresholding. As for the output of u-net, we use a simple gradient

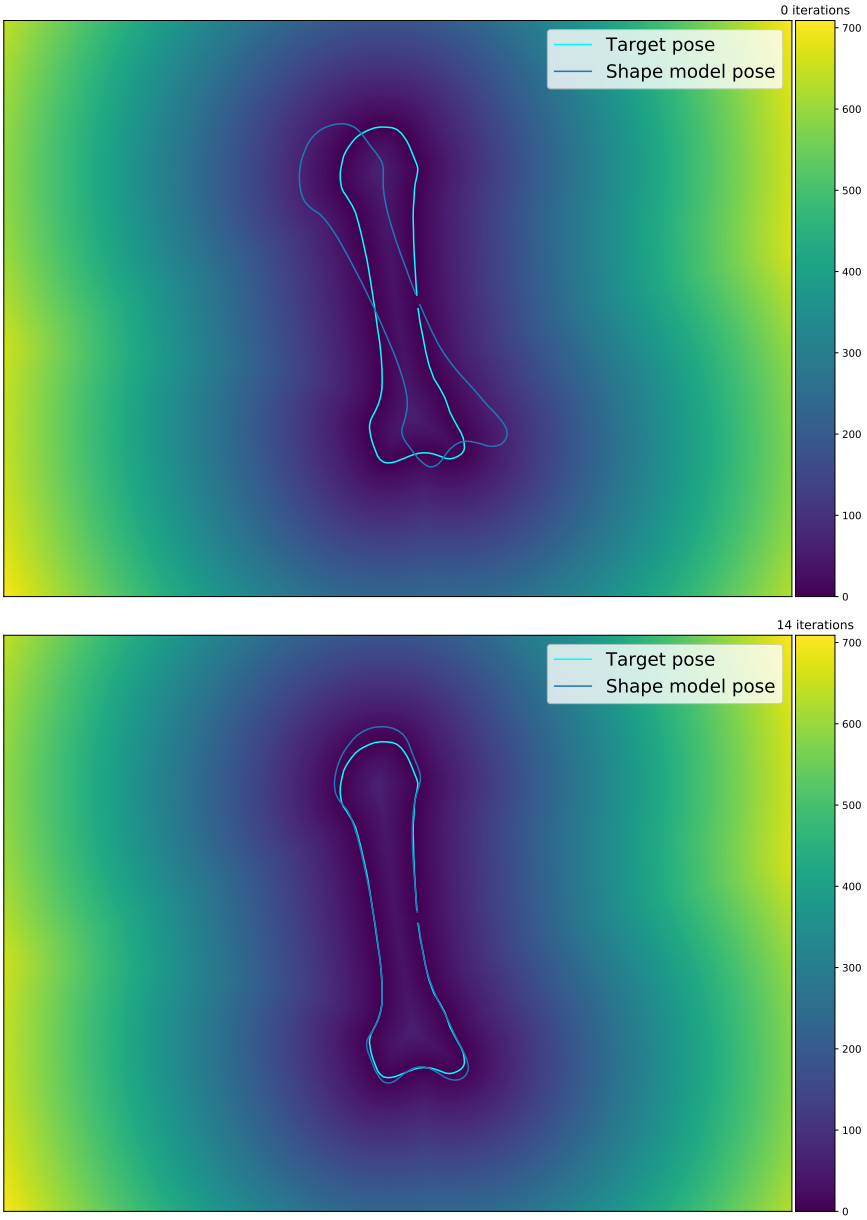


Figure 6-5: The Figures show how the pose is extracted from the full finger model. The bone of interest is isolated from the model and a distance transform is built. We then rotate the shape from the single bone model so that it lies on the zero level set of the distance transform.

based edge detector on the probability map that it generates. Recall that the output of u-net shows the probability of a pixel belonging to a bone region. The fact that this output is smooth allows us to use gradient based edge detectors to find the edges. The result of these operations is shown in Figure 6-3.

Figure 6-6: The Figures show the ALSSM output for the DP, MC, MP and PP (clockwise from top left). The shape template are initialised by using the pose extracted from the coarser ALSSM model fit shown in Figure 6-4.

6.4.2 Coarse to fine approach

6.4.2.1 Description

In an image where the set Γ is quite big and complex, V_u suffers from multiple local minima. This is because locally, the edges of bones look the same. The fact that we are not exploring V_u in a pointwise manner when minimising (6.19) does not guarantee that the solution will not get stuck in a local minima.

Hence, it is important to properly initialise the pose parameters when fitting the model. In medical imaging, we have the advantage that there is a strong prior on the relative pose of objects to be identified. One way to use this prior is to actually fit a shape model that incorporates this information.

More formally, we use a coarse to fine approach in the model fitting. The coarseness is defined by the number of latent space models in the ALSSM fitting.

At level h we have n_h latent spaces $\{\mathrm{X}_l^h; l = 0, ..., n_h - 1\}$ that generate n_h different groups of shapes. These groups $\{\mathbf{M}_l^h; l = 0, ..., n_h - 1\}$ are also accompanied by their respective pose parameters $\{\mathrm{T}_l^h; l = 0, ..., n_h - 1\}$. The latent space captures the relative positions of shapes within the group.

In the next level $h + 1$, the individual groups split and we now have n_{h+1} latent shape models for the smaller, but more numerous groups. It is easy to extract the pose parameters that maintain the relative position of objects when switching levels. Hence, each time the ALSSM undergoes fitting, the location of each object in the image frame is properly initialised. Algorithm 4 shows the steps of the multi level fitting we have just described.

6.4.2.2 Example using bone outline data

In the coarsest level, we use the GPLVM shown in Figure 4-4 as our shape model. The latter models the relative position of the four bones in the index finger. We minimise equation (6.19) where now the shape is generated from the full finger GPLVM. The result of this fit is shown in Figure 6-4. We now have a set of curves that delineate the outlines of the bones in the hand. Note that this will not be perfect as the shape model is mostly able to explore the relative positions of bones.

We use the outlines of the bones from this shape to extract the pose information for each individual bone. Let $\mathbf{M}^{(t)}$ be the curve delineating a bone that the coarser model generates. We call this the target outline. For a GPLVM modelling the shape of this single bone, we generate a source outline $\mathbf{M}^{(s)}$ from the mean of the training latent space position. We then align this source shape on the outline of the target shape (generated from fitting the coarser shape model).

We use the same process as in section 2.3.2.1. We build the distance transform $\mathcal{D}_t$ of the target outline. We then find the pose parameters for that bone outline by performing the following minimisation

$$(r^*, \phi^*) = \arg\min_{r,\phi} \sum_{d=1}^{D-1} \mathcal{D}_t(\mathrm{T} \circ \mathrm{M}_d^{(s)}) \tag{6.21}$$

We now have an initial set of pose parameters for each bone in the index finger. We initialise the shapes using these and now minimise equation (6.19) by this time using a GPLVM that is generating a single bone shape (one for the MC, PP, MP and DP). The result is shown in Figure 6-6.

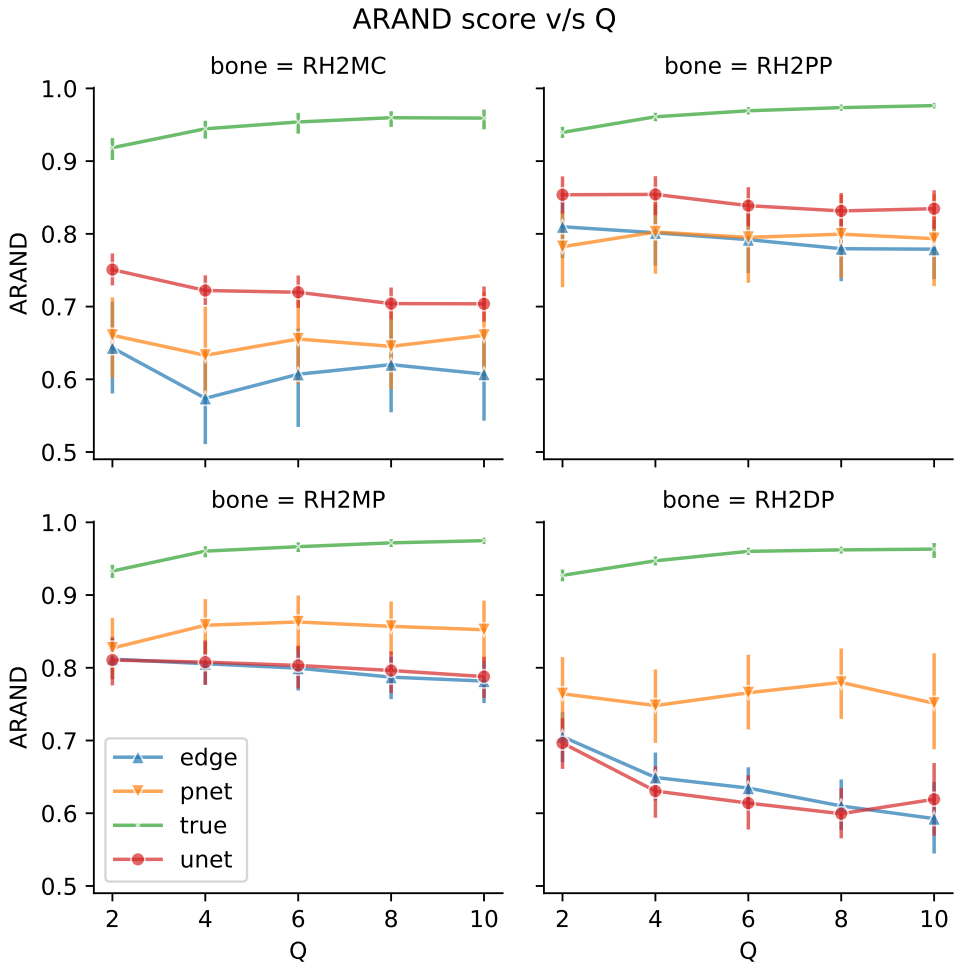


Figure 6-7: The Figure shows how the average ARAND score between the model generated bone area and the true area varies with the dimension Q of the GPLVM latent space in a 10-fold validation. For each training-test set, we use four methods to build the potential function V_u: p-net outline predictions (pnet); u-net region predictions (unet); a gradient based edge detector (edge); and the true bone outline (true). The vertical lines represent error bars (standard deviation) of each mean.

6.4.3 Model evaluation

Given the correct pose, we want to evaluate whether the GPLVM latent space can match the shape observed on a new example. We use the Adjusted RAND or ARAND (Rand [1971]) score to measure how well our model identifies the bone. The ARAND score is mainly used to measure the accuracy of clustering methods. In our case, by considering the pixels inside and outside the bone curve, we are able to cluster pixels based on whether they are part of the bone being investigated or not. We describe the ARAND score in section A.4. We also report the L-2 distance between the true outline and the model generated outline.

Using these two scores, we evaluate the goodness of fit of our ALSSM when

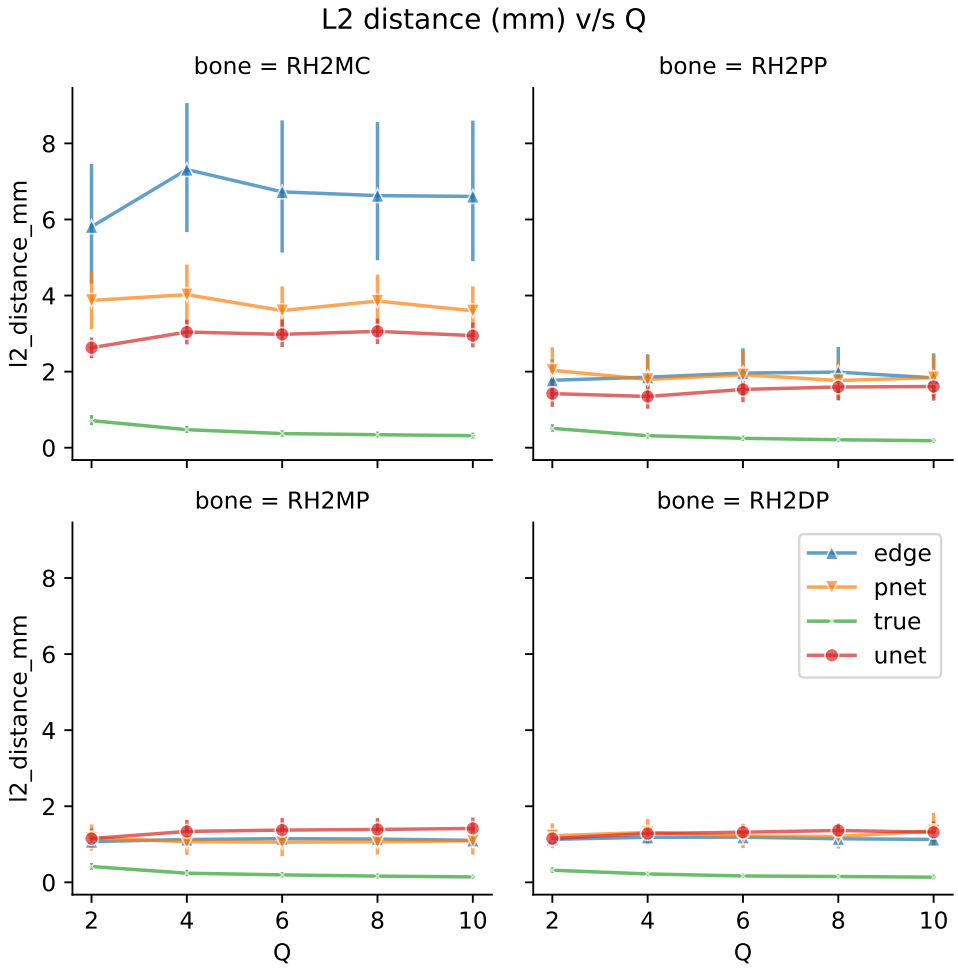


Figure 6-8: The Figure shows how the average L-2 distance (mm) between the model generated bone outline and the true outline varies with the dimension Q of the GPLVM latent space in a 10-fold validation. For each training-test set, we use four methods to build the potential function V_u: p-net outline predictions (pnet); u-net region predictions (unet); a gradient based edge detector (edge); and the true bone outline (true). The vertical lines represent error bars (standard deviation) of each mean.

the latent space dimension of the GPLVM varies. We initially only wish to measure the performance of our ALSSM, and hence use the true outline to build the potential function V_u. We then perform the same experiments by using potential functions resulting from a gradient based edge detector, p-net edge predictions and u-net region predictions. Hence, we are able to assess the goodness of fit of different potential functions as well. We perform a 10 fold validation on each bone in the right hand index finger when using the four different potential functions described above for $Q = 2, 4, 6, 10$. We use the hand drawn outlines in section 2.2 as the true outlines.

We report our full findings in section A.4 and show the accuracy variation of

our model with the latent space dimension Q in Figures 6-7 and 6-8. We can observe that both the average ARAND score and the average L-2 distance improve as the dimension Q of the latent space increases when using the true potential. This is to be expected as the higher the number of dimensions in the latent space, the bigger is the range of small scale variations that the model captures.

However we do not observe the same behaviour when using other types of potential. This is because these potential functions fail to capture fine scale shape variations that the higher dimensional latent space can model. In fact, we actually observe a decrease in performance as the dimension Q increases.

We also notice that the simple edge detector can be out performed by a Neural Network architecture. This is especially apparent when looking at the MC. This is because Neural Networks are able to separate the MC edges from the metacarpal soup at the base of the finger, which is something that simple edge detectors struggle to do. We thus propose to use convolutional architectures such as u-net, which have the potential to perform better than gradient based edge detectors. These architectures also do not require much more processing time as shown in section A.4.

6.5 AAMs interpreted as ALSSM

In our formulation of ALSSMs, we have treated $\mathbf{M}$ to only represent the shape of objects. However, employing the formulation of Cootes et al. [2001], we can treat $\mathbf{M}$ as being a vector of locations (x_i, y_i) along with sampled texture values $\boldsymbol{f}_i = (f_i^0, .., f_i^{d-1})$ in the neighbourhood of (x_i, y_i). Hence we have that for $\mathbf{M} = (\mathrm{M}_0, .., \mathrm{M}_D)$, $\mathrm{M}_j = (x_j, y_j, \boldsymbol{f})$. To define the ALSSM, we only need to specify the latent space representation $\boldsymbol{\mu}(\mathrm{X})$ and the data term $p(\mathbf{M}|u)$.

6.5.1 Latent space representation

Cootes et al. [2001] interpret Images of the same object as being a warped mean appearance to which orthogonal variation modes are added. A warp in our case is any diffeomorphism acting on the domain of the image intensity function. Hence an image is given by:

$$u(\boldsymbol{x}) = \bar{\mathbf{u}}(\mathrm{T} \circ \boldsymbol{x}) + \sum_{q=0}^{Q-1} \omega_q \mathrm{v}_q(\mathrm{T} \circ \boldsymbol{x}) \tag{6.22}$$

where $\bar{\mathbf{u}}$ is the mean image (appearance) and v_k are Q orthogonal variation modes, that is we have $\mathrm{v}_{l1} \cdot \mathrm{v}_{l2} = \mathbb{I}_{l1}^{l2}$. Given training data $\mathbf{Y}$ that has the same form as $\mathbf{M}$, fitting this representation is equivalent to performing PCA on the data. That is, the orthogonal variation mode are given by the first Q eigenvectors of the matrix $\frac{1}{N}(\mathbf{Y} - \bar{\mathbf{Y}})^T(\mathbf{Y} - \bar{\mathbf{Y}})$. Where $\bar{\mathbf{Y}}$ is the matrix of row average of $\mathbf{Y}$. With $\mathrm{X} = \boldsymbol{\omega}$ being the weights applied to the orthogonal variation modes and Λ being the matrix where the columns correspond to $\boldsymbol{v}_q$, we have that

$$\boldsymbol{\mu}(\mathrm{X}) = \bar{\mathbf{Y}} + \Lambda\boldsymbol{\omega} \tag{6.23}$$

As shown in Section 4.3.2, such a latent space representation is not so different from our GPLVM representation of shape. Indeed, under the DAG in Figure 3-3, it is a dual formulation of our latent space model when the covariance structure we use is based on euclidean inner products.

6.5.2 Data term

As $\mathbf{M}$ now contains texture information, it is possible to directly compare how well the texture fits the current image. Hence we have that

$$p(\mathbf{M}|u) = \frac{1}{\sqrt{2\pi v^2}} \exp\left(\frac{1}{2v^2}\sum_{j=0}^{D-1} \|\boldsymbol{f}_j - \mathcal{F} \star u(x_j, y_j)\|^2\right) \tag{6.24}$$

In the above equation, we have introduced a filter $\mathcal{F}$ applied to the image at location (x_i, y_j) that produces a stack of texture values. For example Krüger et al. [2017] uses Gabor Filters to generate $\boldsymbol{f}_j$.

6.5.3 Marginalisation and objective function

Using the notation in equation (6.3) and the marginalisation procedure we introduce in Section 6.2.4, we have that the objective function is given by equation (6.19). In this case however, the pose parameter only acts on the first 2 coordinates of $\mathrm{M} = (x, y, \boldsymbol{f})$ and hence, only acts on the first 2 coordinates of $\boldsymbol{\mu}(\mathrm{X})$. We have that

$$V_u(\mathrm{M}) = \|\boldsymbol{f} - \mathcal{F} \star u(x, y)\|^2 \tag{6.25}$$

We again point out that for such a potential function with support in $U \times [0, 255]^{D-2}$, the assumptions made in Section 3.4 hold and hence the marginalisation we perform in equation (6.11) is still valid. The objective function that we have hence recovered is exactly the one that is minimised in Cootes et al. [2001]. Hence we have shown that AAMs fall under the ALSSM DAG framework.

Chapter 7

Final Conclusions and Outlook

We have shown a top to bottom process for building a segmentation tool for rheumatologists. We have shown how the data was collected, cleaned and processed in Chapter 2. We have shown how the data was used to build a statistical model of the hand in Chapter 4. Shape correspondence was an important part of this thesis. Using a curve to represent shapes allows us to eliminate the need to align points during annotation, hence speeding this process up.

We use CNNs in Chapter 5 to model texture in a discriminative fashion. We have compared the performance of two architectures, one of which delivers good results for bone region discrimination (U-net). The patch-net architecture was shown to perform better at a edge detection task.

We argued in Chapter 6 for a move towards models that are a more realistic representation of the mapping one can have between a latent space and the shape space. We also have shown that a GPLVM shape model can be viewed as the dual of a probabilistic treatment of the ASM. We maintain that model specification is an important part of solving problems such as the one presented here. This not only allows for better interpretation of the models but also to have a more realistic representations of uncertainties present in the system.

7.1 Outlook

7.1.1 Full hand model

A major weakness of this Thesis is the lack of labelled data which prevents us from performing a full hand segmentation. Hence the most achievable improvement that we can suggest over our work involves collecting bone outline data for

all the bones present in a hand x-ray. This will cause U-net to have a better performance. We will also be able to add more levels to the coarse to fine approach we describe in section 6.4.2, and hence be able to segment a whole hand x-ray.

We do anticipate though that the pose initialisation will be an issue when trying to segment a whole hand. However, this issue has been tackled in the literature and we believe that it can be overcome. For example, Cootes et al. [2012] induces a potential for a global search scheme by using random forest voting.

7.1.2 Clinical applications

We believe that our model can be used for semi supervised longitudinal studies involving the joint space width. This is an important metric used to assess JSN and proliferation. The shape of bones from the same individual can be extracted from x-rays taken at different time points. This will allow us to log the progression of JSN and proliferation for individuals that have PsA and those that do not. The process will be semi supervised in the sense that it will involve a human to assess whether the bone was correctly segmented as shown in Figure 1-1.

However, the work done in this thesis will have to be supplemented with more research on the texture of bones suffering from the various types of damage described in section 1.2. One way to do this is to isolate bones that suffer from certain types of damage and performing a regression task on its texture. Again, the semi supervised system shown in Figure 1-1 will allow us to label (assign scores to the respective bones) while building example of images suffering from a particular type of damage.

Such an approach to the deployment and improvement of our system fits with our philosophy of having new ways of evaluating digital health care technology. We believe that the entire process described above will provide the health care sector with a valuable proof of concept. We hope then, that our philosophy towards deployment and testing will become more widespread.

7.1.3 Fully Generative Models

We have shown how one can use the ALSSM to also include generative texture information in section 6.5.1. We believe that we can use a GPLVM to generate texture as well as shape. One interesting question however, involves using two

different types of generative process: one for shape and another one for texture. More specifically, it would be interesting to see if a Variational Autoencoder (Doersch [2016]) an be used alongside a GPLVM shape model under the ALSSM framework. These have been shown to do a good job of generating texture.

7.2 Statistical models of deformation fields

Recall that in section 4.2 we describe shape variations within a family to be the effect of small perturbations of the shape domain. We have used curves to represent this perturbation. This led to statistical models of shapes being statistical models of point clouds. We would like to explore in the future the possibility of directly modelling the distribution of the small perturbations.

Generating a new shape would then be equivalent to perturbing the domain of a template curve. Taking the latent space approach, each latent space would generate a diffeomorphism on $\mathbb{R}^2$, that when applied to a template shape, would generate another shape from the same family. Wang et al. [2006] uses a Gaussian Process to model a dynamical system. This hints towards the possibility of fitting distributions on gradient flows that usually characterise diffeomorphisms on $\mathbb{R}^2$.

Appendix A

Appendix

A.1 Product of Gaussians

We use the following result for the marginalisation we perform in section 6.2.4.

Lemma A.1. *Let* $f_1(x) = \mathcal{N}(x|\mu_1, \sigma_1^2)$ *and* $f_2 = \mathcal{N}(x|\mu_2, \sigma_2^2)$*, then we have that* $f_1(x)f_2(x)$ *is a scaled Gaussian with mean* $\mu = \left(\frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2}\right)\sigma^2$ *and variance* $\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$*. The scaling factor is given by*

$$\frac{1}{\sqrt{2\pi(\sigma_1^2 + \sigma_2^2)}} \exp\left(-\frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}\right)$$

Proof. We complete the square inside the exponential. We neglect the negative sign for ease of notation

$$\begin{aligned} &\frac{1}{2\sigma_1^2}(x - \mu_1)^2 + \frac{1}{2\sigma_2^2}(x - \mu_2)^2 \\ =&\frac{x^2}{2}\left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}\right) - \frac{2x}{2}\left(\frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2}\right) + \frac{\mu_1^2}{\sigma_1^2} + \frac{\mu_2^2}{\sigma_2^2} \end{aligned} \tag{A.1}$$

Comparing the form of a Gaussian quadratic form to equation (A.1) we get that $\frac{1}{\sigma^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}$ as required. To get the mean we introduce σ into the term involving x as follows:

$$
\begin{aligned}
&\frac{2x}{2}\left(\frac{\mu_1}{\sigma_1^2}+\frac{\mu_2}{\sigma_2^2}\right)\\
=&\frac{2x}{2}\left(\frac{\mu_1}{\sigma_1^2}+\frac{\mu_2}{\sigma_2^2}\right)\frac{\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}{\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}\\
&\text{which yields the result for } \mu\\
=&\frac{2x}{2\sigma^2}\left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)
\end{aligned}
\tag{A.2}
$$

we can now use this to complete the square in equation (A.1)

$$
\begin{aligned}
&\frac{x^2}{2}\left(\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}\right)-\frac{2x}{2}\left(\frac{\mu_1}{\sigma_1^2}+\frac{\mu_2}{\sigma_2^2}\right)+\frac{\mu_1^2}{\sigma_1^2}+\frac{\mu_2^2}{\sigma_2^2}\\
&=\frac{1}{2\sigma^2}\left(x-2x\mu\right)+\frac{\mu_1^2}{\sigma_1^2}+\frac{\mu_2^2}{\sigma_2^2}\\
&\text{using the expanded form for } \mu \text{ in equation (A.2)}\\
&=\frac{1}{2\sigma^2}\left(x-\mu\right)^2+\frac{\mu_1^2}{\sigma_1^2}+\frac{\mu_2^2}{\sigma_2^2}-\frac{1}{2\sigma^2}\left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2
\end{aligned}
\tag{A.3}
$$

Now we simplify the expression that is independent of x in equation (A.3).

$$
\begin{aligned}
&\frac{\mu_1^2}{\sigma_1^2}+\frac{\mu_2^2}{\sigma_2^2}-\frac{1}{2\sigma^2}\left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2\\
=&\frac{1}{2\sigma_1^2\sigma_2^2}\left[\sigma_2^2\mu_1^2+\sigma_1^2\mu_2^2-\frac{\mu_1^2\sigma_2^4+2\mu_1\sigma_1^2\mu_2\sigma_2^2+\mu_2^2\sigma_1^4}{\sigma_1^2+\sigma_2^2}\right]\\
=&\frac{1}{2\sigma_1^2\sigma_2^2}\left[\frac{\sigma_2^2\sigma_1^2\mu_1^2+\sigma_1^4\mu_2^2+\sigma_2^4\mu_1^2+\sigma_1^2\sigma_2^4\mu_2^2-\mu_1^2\sigma_2^4-2\mu_1\sigma_1^2\mu_2\sigma_2^2-\mu_2^2\sigma_1^4}{\sigma_1^2+\sigma_2^2}\right]\\
=&\frac{1}{2\sigma_1^2\sigma_2^2}\left[\frac{\sigma_2^2\sigma_1^2\mu_1^2-2\mu_1\sigma_1^2\mu_2\sigma_2^2+\sigma_1^2\sigma_2^4\mu_2^2}{\sigma_1^2+\sigma_2^2}\right]\\
=&\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}
\end{aligned}
\tag{A.4}
$$

Putting equations (A.4) and (A.3) together and putting the quadratic forms in an exponential yields

$$
\begin{aligned}
&\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(-\frac{1}{2\sigma_1^2}\left(x-\mu_1\right)^2\right)\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(-\frac{1}{2\sigma_2^2}\left(x-\mu_2\right)^2\right)\\
=&\frac{1}{\sqrt{2\pi\sigma}}\exp\left(-\frac{1}{2\sigma^2}\left(x-\mu\right)^2\right)\frac{1}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}}\exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right)
\end{aligned}
\tag{A.5}
$$

as required.

□

A.2 Stochastic Gradient Descent

We describe some Stochastic Gradient Descent algorithms as background material for the training of the Neural Network models in Chapter 5. In statistics and machine learning, the objective function to be optimised is often an empirical expectation

$$R(\Theta; \mathbf{Y}) = \frac{1}{N} \sum_{i=0}^{N-1} r(\Theta; \boldsymbol{y}_i) \tag{A.6}$$

Often, because of the dimensionality of data $\mathbf{Y}$, it becomes necessary to optimise the above on a smaller subset of the data to make the optimisation tractable. This turns equation (A.6) into

$$R(\Theta; \mathbf{Y}^{(t)}) = \frac{1}{B} \sum_{i=0}^{B-1} r(\Theta; \boldsymbol{y}_{t_i}) \tag{A.7}$$

where $B < N$ and $\{\boldsymbol{y}_{t_i} : i = 0, .., B-1\} \subset \{\boldsymbol{y}_i : i = 0, .., N-1\}$.

Batching in such a way introduces stochasticity into the algorithm. The sequence $\Theta^{(t)}, t = 0, 1, ...$ that is generated from an iterative scheme hence becomes a stochastic process. More formally, the update step becomes

$$\Theta^{(t+1)} = \Theta^{(t)} - v^{(t)} \tag{A.8}$$

where $v^{(t)}$ drives the stochasticity of the process. There are two main factors to consider when building SGD algorithms

1. **Stochasticity due to the mini-batch**
   We want the chain $v^{(t)}$ to reach its steady state distribution with respect to the stochaticity induced by the random batching of the data.

2. **Efficient direction and step size computation**
   Quasi newton methods tend to be too computationally intractable when the data dimension is too big. Hence, more efficient alternatives to linesearch and descent direction computation need to be found.

We describe a few algorithms that aim to achieve these goals. For ease of notation, we write $\nabla_\Theta R(\Theta^{(t)}) = \nabla_\Theta R(\Theta^{(t)}; \mathbf{Y}^{(t)})$. We call η the learning rate. It is usually pre defined. It is analogous to the step size α that one takes when using gradient descent update steps of the form

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \alpha \nabla_x f(\boldsymbol{x}) \tag{A.9}$$

where $f : \mathbb{R}^M \to \mathbb{R}$ is a function to be minimised with respect to $\boldsymbol{x} \in \mathbb{R}^M$. All the operations defined below are element wise in nature unless otherwise stated.

A.2.0.1 Momentum (Qian [1999])

The momentum algorithm seeks to achieve a steady state via the accumulation of information from previous gradients. The step-size η needs to be determined

$$v^{(t)} = \gamma v^{(t-1)} + \eta \nabla_\Theta R(\Theta^{(t)}) \text{ for } 0 < \gamma < 1 \tag{A.10}$$

This can be thought of a moving average over a sliding window, the length of which is controlled by the momentum γ.

A.2.0.2 Nesterov accelerated gradient (Nesterov [1983])

The Nesterov accelerated gradient algorithm seeks to improve on the momentum algorithm. It does so by evaluating the gradient at the next possible parameter estimation. This is analogous to the stepping used in implicit Ordinary differential equation solvers, where one solves a system to find the value of the function at the next time step.

$$v^{(t)} = \gamma v^{(t-1)} + \eta \nabla_\Theta R(\Theta^{(t-1)} - \gamma v^{(t-1)}) \text{ for } 0 < \gamma < 1 \tag{A.11}$$

A.2.0.3 Adagrad (Duchi et al. [2011])

Define

$$\mathrm{G}^{(t)} = \sum_{k=0}^{t-1} \left[\nabla_\Theta R(\Theta^{(k)})\right]^2 \tag{A.12}$$

The update step is given by

$$v^{(t)} = \frac{\eta}{\sqrt{\mathrm{G}^{(t)} + \epsilon}} \nabla_\Theta R(\Theta^{(t)}) \tag{A.13}$$

for $0 < \epsilon \approx 10^{-8}$. Adagrad tries to fix the problem of choosing the step size. Even though the learning rate still has to be chosen, as the algorithm progresses, dimensions of Θ that have had significant updates will change by smaller amounts due to the scaling used. This might be a problem as certain dimensions of Θ might not get updated at all.

A.2.0.4 RMSprop (Tieleman and Hinton [2012])

Let $x^{(t)} \in \mathbb{R}^d, t = 0, 1, ...$ stationary stochastic process with zero mean then we define

$$\begin{aligned} \mathrm{E}[x^2]^{(t)} &= \gamma \mathrm{E}[x^2]^{(t-1)} + (x^{(t)})^2 \\ &= \sum_{i=0}^{t} \gamma^{t-i} (x^{(i)})^2 \end{aligned} \tag{A.14}$$

as being an estimate of the second moment of the stochastic process. RMSprop is an attempt at fixing the fast decay of Adagrad by instead using this estimate as a per-dimensional scaling in the following way

$$v^{(t)} = \frac{\eta}{\sqrt{\mathrm{E}[g^2]^{(t)} + \epsilon}} g^{(t)} \tag{A.15}$$

where $g^{(t)} = \nabla_\Theta R(\Theta^{(t)})$. Adagrad makes the assumption of an autocorrelation decays slowly with lag. RMSprop makes a more realistic assumption about this rate of decay which can be controlled by the hyperparameter γ as shown in equation (A.14).

A.2.0.5 Adadelta (Zeiler [2012])

Then for $0 < \epsilon \approx 10^{-8}$

$$v^{(t)} = \sqrt{\frac{\mathrm{E}[v^2]^{(t-1)} + \epsilon}{\mathrm{E}[g^2]^{(t)} + \epsilon}} g^{(t)} \tag{A.16}$$

A big problem in the machine learning community is tuning the learning rate to fit the current optimisation problem. Adadelta removes the need to choose it, as the step size is determined by the ratio $\sqrt{\frac{\mathrm{E}[v^2]^{(t-1)} + \epsilon}{\mathrm{E}[g^2]^{(t)} + \epsilon}}$. The latter provides a way to balance information from past iterations with information from the current iteration about the geometry of the random surface.

A.2.0.6 Adam (Kingma and Ba [2014])

For predefined decay rates $0 < \beta_1, \beta_2 < 1$ we define for $i = 1, 2$

$$\begin{aligned} m_i^{(t)} &= \beta_i m_i^{(t-1)} + (1 - \beta_i)(g^{(t)})^i \\ &= (1 - \beta_i^t) \sum_{k=0}^{t} \beta^{t-k} (g^{(k)})^i \end{aligned} \tag{A.17}$$

Like equation (A.14), the above is a biased estimate for the i-th moment of the gradient where past values of the gradient contribute less to the current moment estimate. We the define

$$\hat{m}_i^{(t)} = \frac{m_i^{(t)}}{1 - \beta_i^t} \tag{A.18}$$

which turns equation (A.17) into unbiased estimates of the $i-$th moment of the stochastic gradient g_t. As we then have:

$$\begin{aligned} \mathbb{E}[\hat{m}_i^{(t)}] &= \frac{1}{1-\beta_i^t}\mathbb{E}\left[(1-\beta_i)\sum_{k=0}^{t}\beta^{t-k}(g^{(k)})^i\right] \\ &= \frac{1}{1-\beta_i^t}\mathbb{E}\left[(g^{(t)})^i\right](1-\beta_i)\sum_{k=0}^{t}\beta^{t-k} \\ &= \mathbb{E}\left[(g^{(t)})^i\right] \end{aligned} \tag{A.19}$$

It also mitigates the problem of $m_i^{(t)}$ being close to zero in the initial stages of the optimisation. The update step is given by:

$$v^{(t)} = \frac{\eta}{\sqrt{\hat{m}_2^{(t)}} + \epsilon}\hat{m}_1^{(t)}. \tag{A.20}$$

Adam tries to solve the stochaticity problem and the step size and direction problem by exploiting the assumed stationarity of the stochastic process $g^{(t)}$. It still requires a learning rate to be specified, but like Adagrad and RMSprop, these only affects the initial behaviour of the algorithm.

A.3 Introduction to Neural Networks

We give an introduction on Neural Networks to support the material in Chapter 5. Traditionally, artificial neural networks (ANNs) were an attempt to solve the supervised learning problem where a signal $\boldsymbol{y}$ was being explained from a set of observed features $\boldsymbol{x}$ through the map

$$\boldsymbol{y} = f(\boldsymbol{x}) \tag{A.21}$$

It had been known for a long time that the human visual cortex and the human cognitive system relied on a highly non linear map f for general inference. Much work of the early neural network framework was based on the description made by Hebb [2005] of the human cerebral cortex. It was then understood that an artificial brain would be based on a large number of logic gates.

Attempts at simulating this network of logic gates such as the one by Farley and Clark [1954]. Hence, even though ANNs had been known in the literature since the 1940s, the lack of processing power and the difficulty in approximating neuron activations made them very unpopular. It was not until the description of backpropagation by Werbos [1974] that the research community started to delve into ANNs again. The arrival of parallel computing in the 1980s further accelerated the development of ANNs.

A.3.1 Setting

From now on, we assume that Neural Networks are being used for image data. Data $\mathbf{Z}_0$ is fed through a neural network and is mapped through succesive layers via a composition of an affine transform with a non linear activation function. Hence, each layer l is related to the previous by:

$$\mathbf{Z}_l = \phi_l \circ \mathrm{T}_l\left(\mathbf{Z}_{l-1}\right) \tag{A.22}$$

where ϕ_l is the activation function and T_l is the affine transformation.

In general, the affine transformation is parametrised by a set of weights and biases which we denote by θ_l. The weights $\mathbf{W}_l$ define the linear part of the transformation and the bias $\boldsymbol{b}_l$ defines an offset similar to the ones used in linear regressions. The output of a neural network with L layers is denoted by $\mathbf{Z}_L$ while the input is denoted by $\mathbf{Z}_0$. They are related by the ANN map

$$\mathbf{Z}_L = f_{net}\left(\mathbf{Z}_0\right) \tag{A.23}$$

where

$$f_{net}\left(\mathbf{Z}_0\right) = \phi_L \circ \mathrm{T}_L \circ \phi_{L-1} \circ \mathrm{T}_{L-1} \circ ... \circ \phi_1 \circ \mathrm{T}_1\left(\mathbf{Z}_0\right) \tag{A.24}$$

We shall employ the convention used in computer tensor algebra when indexing arrays. I.e, for an array $\boldsymbol{a}$ of size $N_x \times N_y \times N_z$, we shall use $\boldsymbol{a}[i, j, k]$ to denote its entries. We also use the notation $[h_1 : h_2, i_1 : i_2, j_1 : j_2]$ to denote the discrere interval $\{h_1, h_1 + 1, ..., h_2\} \times \{i_1, i_1 + 1, ..., i_2\} \times \{j_1, j_1 + 1, ..., j_2\}$. We denote by $\boldsymbol{a}[h_1 : h_2, i_1 : i_2, j_1 : j_2]$ the ordered slice of the array corresponding to the set defined by $\{h_1, h_1 + 1, ..., h_2\} \times \{i_1, i_1 + 1, ..., i_2\} \times \{j_1, j_1 + 1, ..., j_2\}$.

A.3.2 Activation Functions

The middle layers of neural networks consist of neurons. The neurons are elements of the vector $\mathbf{Z}_l$. In traditional ANN theory, these neurons are made to behave like logic gates. Hence activation functions were used that were approximations of the indicator function

$$\mathbf{1}(x) = \begin{cases} 1 & \text{if } x > 1 \\ 0 & \text{otherwise} \end{cases} \tag{A.25}$$

In fact Funahashi [1989] showed that we actually require activation functions to be basis functions. We give a few examples in Table A.1 of some commonly used activation functions. Note that most of these can be used as so called universal function approximators.

Much of the focus in the deep learning community has been to find activation functions whose gradients would behave reasonably well during training. A common problem in training is that of vanishing gradients. For example, for large values of $|x|$, the derivative of the sigmoid function in Table A.1 tends to 0. This means that during training, neurons that have a big pre-activation [1] will receive little updating. This is similar to getting stuck in a local extremum.

A common activation used for the output layer is the softmax function. It acts on a vector of length N_c (usually number of classes in a classification task). It is given by

[1] the state or value of the neuron before passing through the activation function

Name	$f(x)$	$f'(x)$
Sigmoid	$\frac{1}{1+e^{-x}}$	$\frac{e^x}{(1+e^x)^2}$
Tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\frac{4}{e^x+e^{-x}}$
Arctan	$\tan^{-1}(x)$	$\frac{1}{x^2+1}$
Relu	$x\mathbf{1}(x)$	$\mathbf{1}(x)$

Table A.1: The table shows different activations and their derivatives.

$$\boldsymbol{\sigma}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{n=0}^{N_c-1} e^{z_n}} \tag{A.26}$$

The output of the softmax at index i usually represents the probability that the input data is in class i.

A.3.3 Discrete kernel operations

Before describing the layers in ANNs, we give a description of the generalised discrete kernel convolution of a 2-d kernel with a 2-d array. We have a kernel $\boldsymbol{K} \in \mathbb{R}^{S^x \times S^y}$ applied to an image $\mathbf{U} \in \mathbb{R}^{N^x \times N^y}$. The operation is split in two. First, a window of size $S^x \times S^y$ is selected inside the padded image domain. Padding is the process of appending and prepending the image with P_x rows and P_y columns. Then, each element in this window is multiplied by the corresponding entry in the kernel. The second part of the operation is a reduction operation f_r on the newly created array.

$$\boldsymbol{K} \star \mathbf{U}[m,n] = f_r\left(\boldsymbol{K} \odot \mathbf{U}\left[s_x(m) : s_x(m) + S^x - 1, s_y(n) : s_y(n) + S^y - 1\right]\right) \tag{A.27}$$

s_x and s_y map the target coordinates of the output map to the source coordinate in the input map. It is given by

$$\begin{aligned} s_x(m) &= -P_x + md_x \\ s_y(m) &= -P_y + md_y \end{aligned} \tag{A.28}$$

d_x and d_y are the stride lengths of the kernel. They are the jumps that the kernel makes as the window of operation moves along the spatial dimensions of the array. Hence, as per equation (A.28), the source pixels are then separated by $d_x - 1$ and $d_y - 1$ pixels in each dimension. For an image that has been padded, we adopt the convention $\mathbf{U}[x, y] = 0$ when $(x, y) \notin [0 : N^x - 1, 0 : N^y - 1]$.

Definition A.2 (Kernel Window Operation)**.** *Consider a one dimensional kernel of size S being applied to an array of size N with padding P and stride d. We call the the interval $[s(m) : s(m) + S - 1]$ the kernel window of of operation at source location $s(m)$. Here $s(m)$ is given by*

$$s(m) = -P + md \tag{A.29}$$

The kernel window of operation for a kernel acting on many dimensions is given by the product of the windows of operation in each dimension. The size of the output is given by the value of m in equation (A.28) for which the kernel window $[s_x(m) : s_x(m) + S^x - 1, s_y(n) : s_y(n) + S^y - 1]$ is completely inside the padded image. Without loss of generality, we solve

$$\begin{aligned} s_x(m) &= (N^x - 1) - (S^x - 1) + P_x \Leftrightarrow \\ -P_x + md_x &= (N^x - 1) - (S^x - 1) \Leftrightarrow \\ md_x &= (N^x - 1) - (S^x - 1 + 2P) \Leftrightarrow \\ m &= \left\lfloor \frac{N^x - S^x + 2P}{d_x} \right\rfloor + 1 \end{aligned} \tag{A.30}$$

Solving equation (A.30) for N_{l+1} is equivalent to finding the number of partitions one can make in an array of size $N_L^x - 2\left(\left\lfloor \frac{S^x}{2} \right\rfloor - P^x\right)$ through jumps of size d_x that stay within the array.

Lemma A.3. *Consider a one dimensional kernel of size S being applied to an array of size N with padding P and stride S. The size of the output array is given by*

$$N_{out} = \left\lfloor \frac{N - S + 2P}{d} \right\rfloor + 1 \tag{A.31}$$

Figure A-1 shows a kernel operation when the number of valid jumps match with the dimension of the image perfectly, resulting in the same size image. Figures A-2 shows what happens when a lack of padding causes information to be lost. Figure A-3 shows how this problem can be fixed by using padding.

In general, one would use a stride of 1 and a padding of size $\left\lfloor \frac{S}{2} \right\rfloor$ in each dimension to preserve the size of an image. It is common to want to half the size of the image in each dimension, in which case a kernel size of 2 with a stride of 2 and no padding is used.

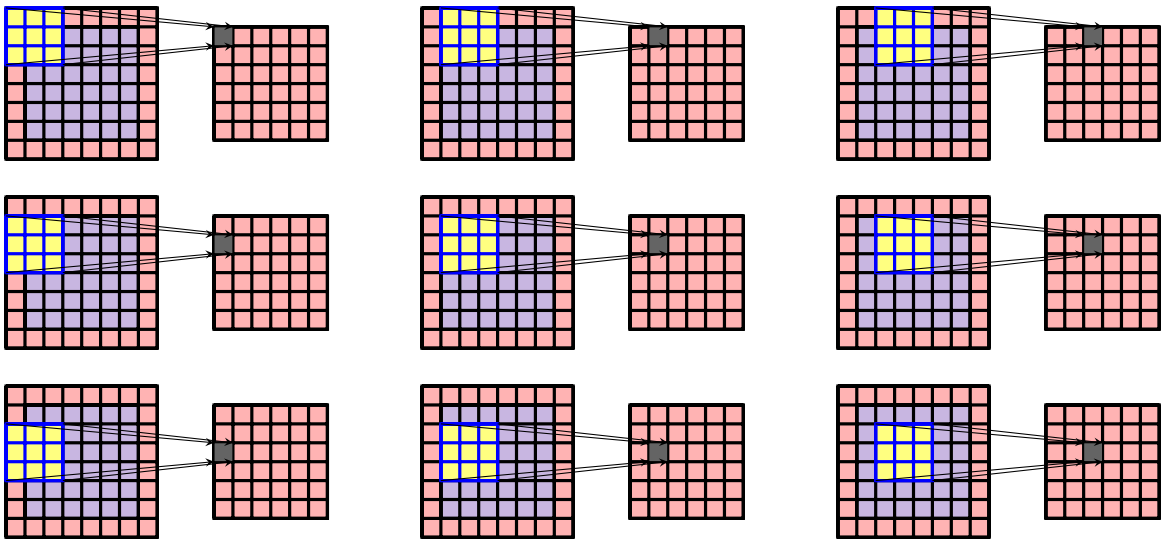


Figure A-1: The Figure shows a kernel of size 3×3 being applied to an image of size 6×6 with a stride of 1 in each dimension and a padding of 1. This yields an output of size 6×6 as per equation (A.31).

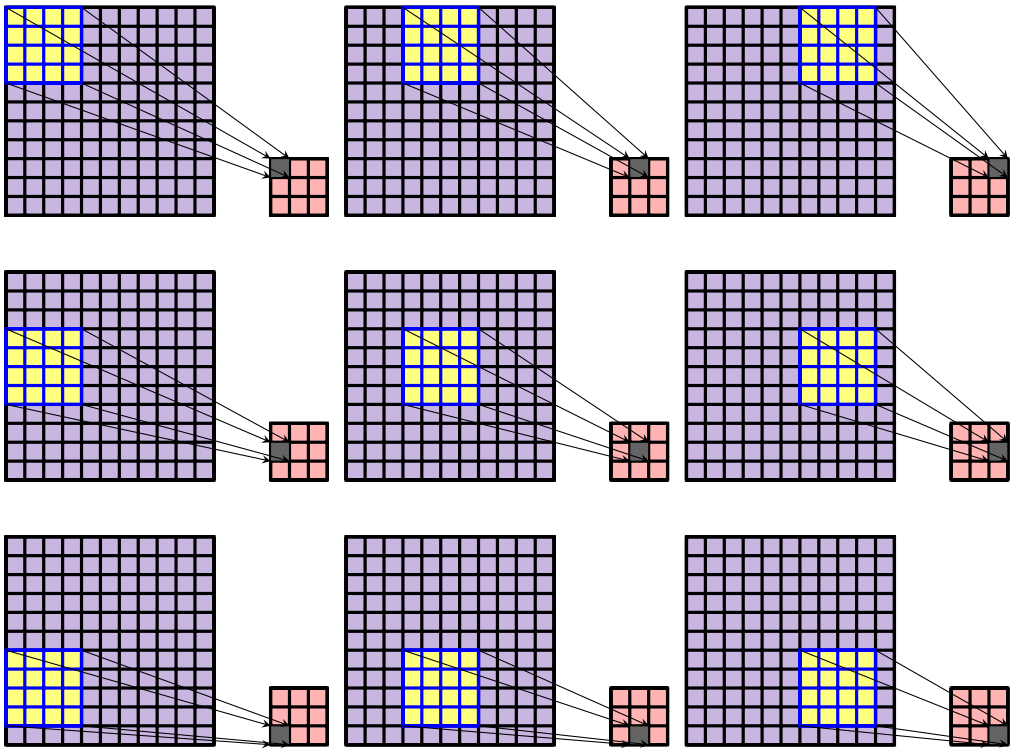


Figure A-2: The Figure shows a kernel of size 4×4 being applied to an image of size 11×11 with a stride of 3 in each dimension and a padding of 0. This yields an output of size 3×3 as per equation (A.31). This is an example where the last row and column of the image do not contribute to the output.

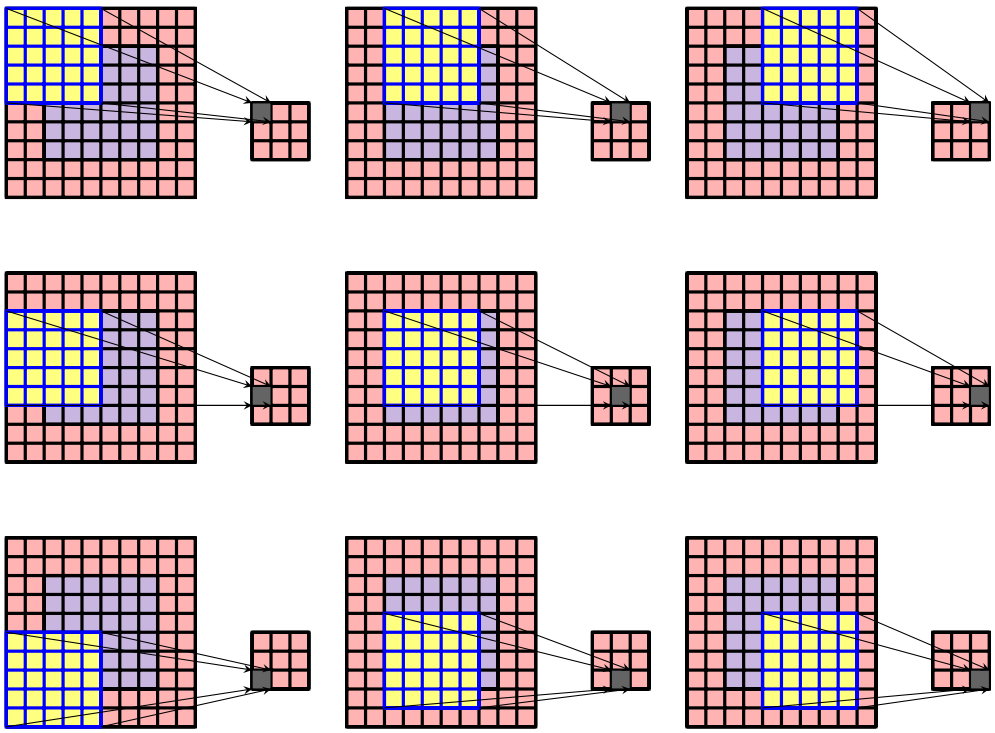


Figure A-3: The Figure shows a kernel of size 5×5 being applied to an image of size 6×6 with a stride of 3 in each dimension and a padding of 2. This yields an output of size 3×3 as per equation (A.31).

A.3.4 Layers

The types of layers in a Neural Network are generally characterised by the type of affine transformation between layers. The affine transformation dictates the level of connectivity between successive layers.

A.3.4.1 Fully Connected Layers

Fully connected layers are characterised by a matrix multiplication between layers that acts as the affine map T. That is for $\mathbf{Z}_l \in \mathbb{R}^{N_l}$

$$\mathbf{Z}_l = \phi_l \left(\mathbf{W}_l \mathbf{Z}_{l-1} + \boldsymbol{b}_l \right) \tag{A.32}$$

where $\mathbf{W}_l \in \mathbb{R}^{N_{l-1} \times N_l}$ is the weight matrix and $\boldsymbol{b}_l \in \mathbb{R}^{N_l}$. If each element of $\mathbf{Z}_l$ is treated as a neuron, then the weight matrix $\mathbf{W}_l$ defines a dense connection between layers where each neuron in the current layer is mapped to each neuron in the next layer as shown in Figure A-4.

A.3.4.2 Convolutional Layers

Convolutional Layers on the other hand, use a kernel convolution between layers as the linear part of the Affine map. Equation (A.22) becomes

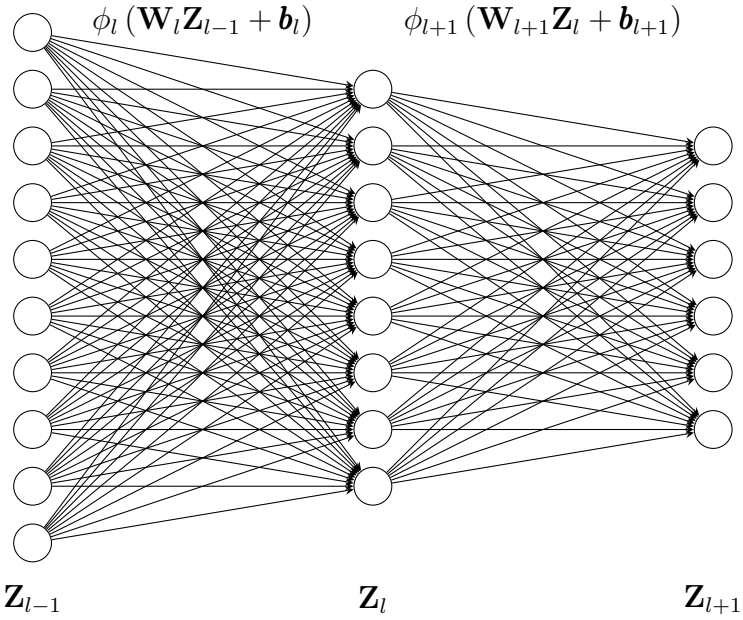


Figure A-4: The Figure shows the relationship between three fully connected layers of a Neural Network. The components of $\mathbf{Z}_l$ represent the neurons in the network. These are represented by the circles. The Weight matrix $\mathbf{W}_l$ creates a dense connection between individual neurons between layers.

$$\mathbf{Z}_{l+1} = \phi_l\left(\mathbf{W}_l \star \mathbf{Z}_l + \boldsymbol{b}_l\right) \tag{A.33}$$

One of the motivations of Convolutional Layers is that the dense connection in fully connected layers are not invariant to rotations and shifts of objects present in image space. Using a sliding window of weights such as a kernel convolution, partly addresses this issue.

Moreover, for a long time, researchers have focused on finding filtering methods that would extract texture information from images. Canny [1987] showed how to use a variety of filters to detect edges in an image. Later, Gabor filters were used by Jain and Farrokhnia [1991] for text recognition. Convolutional layers are an extension of such efforts, where now, the filter is specifically designed for the inference at hand through backpropagation.

In general, data between convolution layers take the form a multi channel image. That is $\mathbf{Z}_l \in \mathbb{R}^{N_l^x \times N_l^y \times C_l}$ where N_l^x, N_l^x are the width and height of the image[2] respectively and C_l is the number of channels. $\mathbf{W}_l \in \mathbb{R}^{S_l^x \times S_l^y \times C_l \times C_{l+1}}$ is a discrete convolution kernel with horizontal spatial extent S_l^x and vertical spatial extent S_l^y defining a mapping $\boldsymbol{K}_l : \mathbb{R}^{N_l^x \times N_l^y \times C_l} \to \mathbb{R}^{N_{l+1}^x \times N_{l+1}^y \times C_{l+1}}$. We usually

[2] N_l^x, N_l^x are the spatial dimensions of $\mathbf{Z}_l$

have that S_l^x, S_l^y are odd. Setting $S = S_l^y = S_l^x$ for ease of notation, we have

$$(\mathbf{W}_l \star \mathbf{Z}_l)[m,n,o] = \sum_{h=0}^{C_l-1} \sum_{0\leq i\leq S-1} \sum_{0\leq j\leq S-1} \mathbf{W}_l[i,j,h,o]\mathbf{Z}_l[s_x(m)+i, s_y(n)+j, h] \tag{A.34}$$

Equation (A.34) is a discrete convolution operation as defined in Section A.3.3. In this case, f_r is the reduce sum operation across the channels and within the window of operation. s_x, s_y are as in section A.3.3 and are defined by equation (A.28). The bias term $\boldsymbol{b}_l \in \mathbb{R}^{N_{l+1}^x \times N_{l+1}^y \times C_{l+1}}$ is a constant along the first 2 dimensions. That is, each output channel gets shifted by a constant term.

The padding P_x and P_y and the strides d_x and d_y determine the dimension of the output map. The output size N_{l+1}^x and N_{l+1}^y of each dimension is given by equation (A.31).

A.3.4.3 Pooling Layers

One central aspect of Convolutional Neural networks is that they try to replicate the early efforts in the computer vision community to represent the human visual cortex. The Nyquist sampling theorem along with descriptions by Koenderink [1984] of the human visual cortex have led researchers to conclude that it is necessary to represent images at different scales or sizes. Scale space was described by Lindeberg [2001] which then led to the development of feature extractors such as Scale Invariant Feature Transform algorithm by Lowe [2004].

For convolutional layers to replicate those efforts, pooling layers were introduced, which would downsample the image after a convolutional operation. Moreover, they impose an additional layer of translation invariance to the image, as the value of the pooling layer loses some spatial information about the source image.

Like a convolutional layer, a pooling layer consists of a kernel operation followed by an activation function. Pooling layers however differ in two ways. Firstly, the activation function that is applied after the kernel operation is the identity function. Secondly, the kernel used is a slicing kernel with entries that are all ones. Moreover, unlike in convolutional layers, the kernel weights do not get updated during training.

We have the input $\mathbf{Z}_l \in \mathbb{R}^{N_l^x \times N_l^y \times C_l}$. The kernel has size $\boldsymbol{K}_l \in \mathbb{R}^{S_l^x \times S_l^y}$ and

consists of only ones and maps the input onto the output $\mathbf{Z}_{l+1} \in \mathbb{R}^{N_{l+1}^x \times N_{l+1}^y \times C_l}$. We can rewrite equation (A.27) as

$$\boldsymbol{K}_l \star \mathbf{Z}_l[m,n,h] = f_r\left(\mathbf{Z}_l\left[s_x(m) : s_x(m) + S^x - 1, s_y(n) : s_y(n) + S^y - 1, h\right]\right) \tag{A.35}$$

Common reduction operation used are

- the max pooling operation given by $f_r(\boldsymbol{x}) = \max\{x_0, ..., x_{n-1}\}$;
- the average pooling operation given by $f_r(\boldsymbol{x}) = \frac{1}{n}\sum_{i=0}^{n-1} x_i$; and
- the median pooling operation given by $f_r(\boldsymbol{x}) = \text{median}\{x_0, ..., x_{n-1}\}$

In pratice, pooling is performed with a kernel of size 2 in each spatial dimension and a stride of length 2 on an unpadded image to produce an output of half the size of the input in each dimension. Indeed, with these values of d, P and S, equation (A.31) yields

$$N_{l+1} = \left\lfloor \frac{N_l}{2} \right\rfloor \tag{A.36}$$

for all spatial dimensions.

A.3.4.4 Transposed Convolution

As we have previously described, images are downsampled, or compressed, by applying a convolutional layer and a pooling layer. It is often desirable to upsample the image in deeper layers. Moreover, we want the type of upsampling to be application specific by letting the operation be updated through backpropagation. This is achieved through the transpose convolution operation.

As in Section A.3.4.2, $\mathbf{Z}_l \in \mathbb{R}^{N_l^x \times N_l^y \times C_l}$ is the input to the layer, and $\mathbf{W}_l \in \mathbb{R}^{N_l^x \times N_l^y \times C_l \times C_{l+1}}$ is a discrete convolution kernel with horizontal spatial extent S_l^x and vertical spatial extent S_l^y.

Before applying the kernel operation to $\mathbf{Z}_l$, the latter is first augmented by mapping it to a tensor $\hat{\mathbf{Z}}_l \in \mathbb{R}^{\hat{N}_l^x \times \hat{N}_l^y \times C_l}$ where $\hat{N}_l^x = \delta_x N_l^x$ and $\hat{N}_l^y = \delta_y N_l^y$. We call $\delta_x, \delta_y > 1 \in \mathbb{N}$ the horizontal and vertical dilations respectively. We have that

$$\hat{\mathbf{Z}}_l[i,j,k] = \begin{cases} \mathbf{Z}_l\left[\frac{i}{\delta_x}, \frac{j}{\delta_y}, k\right] & \text{if } i \mod \delta_x = 0 \text{ and } j \mod \delta_y = 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.37}$$

Then the output to the layer is given by

$$\mathbf{Z}_{l+1} = \phi_l\left(\mathbf{W}_l \star \hat{\mathbf{Z}}_l + \boldsymbol{b}_l\right) \tag{A.38}$$

where

$$\left(\mathbf{W}_l \star \hat{\mathbf{Z}}_l\right)[m,n,o] = \sum_{k=0}^{C_l-1} \sum_{0\le i\le S-1} \sum_{0\le j\le S-1} \mathbf{W}_l[i,j,k,o]\hat{\mathbf{Z}}_l\left[s_x(m)+i, s_y(n)+j, k\right] \tag{A.39}$$

and $\boldsymbol{b}_l$ is the bias term as in section A.3.4.2. This is equivalent to adding $\delta_y - 1$ after every column of $\mathbf{Z}_l$ and $\delta_x - 1$ rows after every row of $\mathbf{Z}_l$ before convoluting it with a kernel. It is common to not use striding when using the transposed convolution. In fact dilation can be considered as the dual of the striding operation as it negates the effect that the latter has on size. This dual treatment is extended by Dumoulin and Visin [2016] who also defines a dual padding. However, we treat a transposed convolution as a dilated convolution operation with stride 1. By using equation (A.31) we get the following equation for the output size

$$\begin{aligned} N_{l+1}^x &= N_l^x \delta_x - S_l^x + 2P + 1 \\ N_{l+1}^y &= N_l^y \delta_y - S_l^y + 2P + 1 \end{aligned} \tag{A.40}$$

Using a kernel size of $2P+1$ with a dilation of 2 will cause the dimension to double. It is common in the literature to use $S = 3$ and $P = 1$. It is also common practice to use have $S \ge \delta_x$ to capture local dependencies.

A.3.4.5 Skip Connections

Skip connections arise when the output of a previous layer is appended to the input of a deeper layer. That is

$$\mathbf{Z}_{l+1} = \phi_{l+1} \circ \mathrm{T}_{l+1}\left(\mathbf{Z}_l, \mathbf{Z}_k\right) \tag{A.41}$$

where $k < l$. These usually arise in purely convolutional neural networks, where the dimension of the current layer is augmented along the channels by concatenating the input with the output of a previous convolutional layer. In these

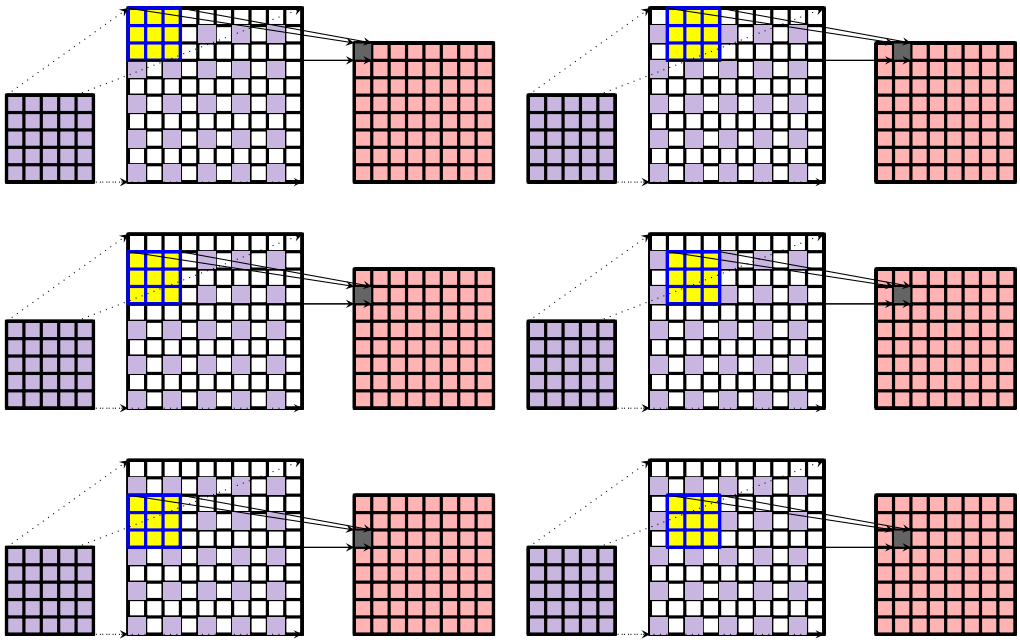


Figure A-5: The Figure shows a transposed convolution operation of a kernel of size 3×3 with an image of size 5×5. A dilation of 2 is used with no padding. The output has size 8 as per equation (A.40).

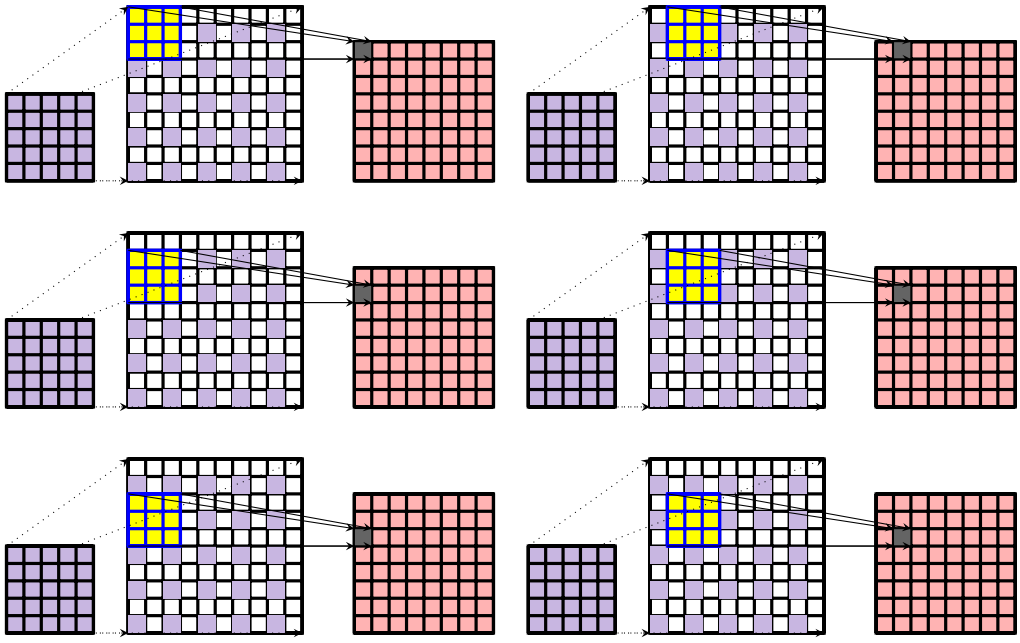


Figure A-6: The Figure shows a transposed convolution operation of a kernel of size 3×3 with an image of size 5×5. A dilation of 2 is used with no padding. The output has size 8×8 as per equation (A.40).

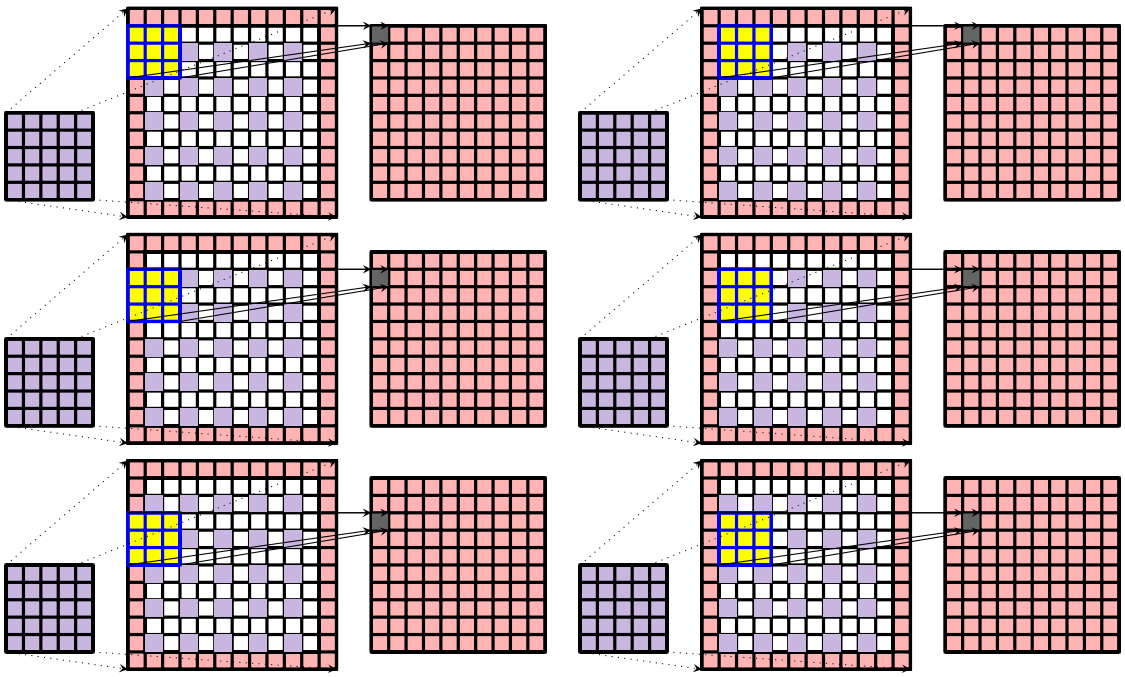


Figure A-7: The Figure shows a transposed convolution operation of a kernel of size 3×3 with an image of size 5×5. A dilation of 2 is used with no padding. The output has size 8×8 as per equation (A.40).

situations, it is necessary for $\mathbf{Z}_k$ and $\mathbf{Z}_l$ to have the same spatial dimension. Figure 5-4 shows the use of skip connections in the unet architecture, which we later describe. However, it is possible to have more general maps $\phi_{l+1} \circ \mathrm{T}_{l+1}$ in equation (A.41).

A.3.5 Training

Given training data $\mathbf{X} = (\boldsymbol{x}_0, ..., \boldsymbol{x}_{N-1})^T$ and their corresponding labels $\hat{\mathbf{Y}} = (\hat{\boldsymbol{y}}_0, ..., \hat{\boldsymbol{y}}_{N-1})^T$, we want to find the set of weights and biases $\Theta = \{\theta_l, l = 1, .., L\}$ such that the difference between the predictions $\boldsymbol{y} = f_{net}(\boldsymbol{x})$ and the labels $\hat{\boldsymbol{y}}$ is minimised. As $\boldsymbol{y}$ is a function of the set Θ, we can define an energy on Θ through a loss function between the predictions and labels as follows

$$R(\Theta) = \sum_{i=0}^{N-1} r(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) \tag{A.42}$$

Where r is a loss function. We $\Theta^* = \arg\min_\Theta R(\Theta)$ that minimises the loss $r(\boldsymbol{y}, \hat{\boldsymbol{y}})$ between labels and predictions.

A.3.5.1 Backpropagation and Forward propagation

Minimising equation (A.42) is usually done through stochastic gradient descent as described in section A.2. The gradient $\nabla_\Theta R$ is calculated through the chain

rule.

Theorem A.4 (Chain Rule)**.** *Let $f : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \mathbb{R}^K \to \mathbb{R}^M$ be differentiable functions with continuous derivatives. Let $h := f \circ g$ be the composition of the 2 functions, then the jacobian of h is given by the matrix multiplication of the jacobian of g evaluated at x and the jacobian of f evaluated at $g(x)$.*

$$J_h(x) = J_g(x) J_f\left(g(x)\right)$$

The recursive nature of f_{net} and the chain rule provides an efficient algorithm for calculating the gradients $\frac{\partial L}{\partial \theta_l}$. Using equation (A.22), we have that

$$\frac{\partial \mathbf{Z}_l}{\partial \theta_l} = \phi_l'\left(\mathrm{T}_l \circ \mathbf{Z}_{l-1}\right) \frac{\partial \mathrm{T}_l}{\partial \theta_l}\left(\mathbf{Z}_{l-1}\right) \tag{A.43}$$

Using this an equation (A.24) we have that for any l,

$$\frac{\partial R}{\partial \theta_l} = \frac{\partial R}{\partial \mathbf{Z}_L} \prod_{k=0}^{L-l} \frac{\partial \mathbf{Z}_{L-k}}{\partial \theta_{L-k}} \tag{A.44}$$

where $\prod$ here denotes matrix multiplication. The process of recursively calculating the gradient for the parameters in each layer given in equation (A.44) is called **back propagation**. $\dfrac{\partial R}{\partial \theta_l}$ requires the computation of $\mathbf{Z}_l, \mathbf{Z}_{l+1}, .., \mathbf{Z}_L$. Hence each time back propagation is carried out, the output of each layer has to be recomputed. The process of passing the input data forward through the network to generate an output is called **Forward Propagation**. The training of a neural network iteratively alternates between the backpropagation and the forward propagation steps.

As described in section A.2, the data is split into minibatches $\mathbf{X}^{(t)}, \mathbf{Y}^{(t)}$ of size B. One epoch is the number of SGD steps such that the whole data has passed through the algorithm and is given by $\left\lfloor \frac{N}{B} \right\rfloor + 1$. As the batching is random, it is not guaranteed that every data point will pass through the SGD algorithm in one epoch. However sampling methods can be devised to ensure that this happens.

A.3.5.2 Dropout

Dropout works by randomly setting the weight and bias contribution of a neuron in a layer to zero. The probability of any one neuron being switched off in this manner in a layer is called the **dropout rate**. This is depicted in Figure A-8. Drop out is performed during training. It reduces the dependence of the prediction on any one particular neuron as the latter is not optimised in all the runs of the gradient descent algorithm. This is demonstrated by Srivastava et al. [2014].

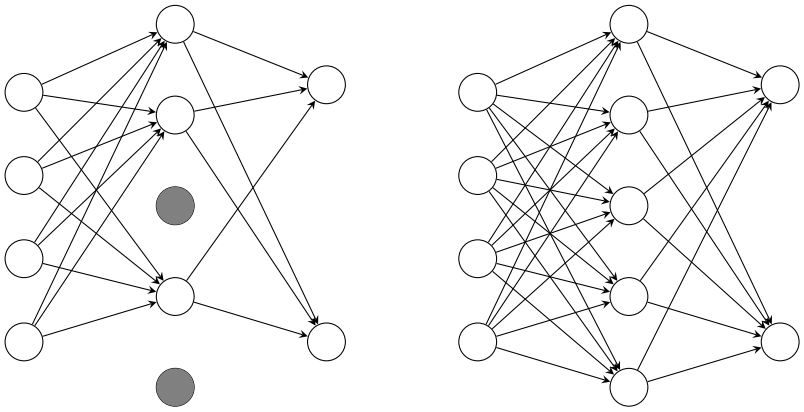


Figure A-8: The figure shows two instances of a fully connected layer. On the left, dropout is applied, with the result that some of the hidden neurons have no contribution to the value of the output neurons.

It is now widely accepted that in the community that dropout does indeed speed up training by reducing the overfitting effect.

Dropout can also be viewed from a model selection point of view. Consider a layer with P neurons on which dropout is applied. This creates a stochastic classifier during training. One can think of dropout as creating 2^P different neural networks which undergo sparse training at the dropout layer. Hence, when predicting, keeping dropout on that layer would create a random prediction that is analogous to the predictions created by random forests. One would then create an ensemble of predictions from which the mode is reported.

However, neural networks tend to be expensive to evaluate and creating ensemble predictions is not feasible. Srivastava et al. [2014] shows that dropout need only by applied when training. During test time and when using the trained network, dropout need not be used. In Figure A-8, the diagram on the left would represent the fully connected layer during training while the diagram on the left would be that same layer when using the neural network on new examples.

A.4 Experiment Results

In this section, we tabulate the results of the experiments we carry out in section 6.4.3. We perform a 10 fold cross validation on the ALSSM fitting procedure. At each run, 10% of the data set is omitted from the training set and the fitted model is tested on this set. This is performed 10 times so that the whole dataset is covered in this way. For each test example, we compute the ARAND score

between the model generated classification and the true classification and the L-2 distance between the generated outline and the true outline.

A.4.1 ARAND score

By fitting a curve outline to a bone in an X-ray, we are able to classify pixels as being inside or outside the curve, or alternatively as having positive or negative labels. Consider the following:

(a) Let c^+ be the number of pixels that are correctly classified as positive by the model;

(b) c^- be the number of pixels that are correctly classified as negative by the model;

(c) b^+ be the number of pixels that are incorrectly classified as positive by the model; and

(d) b^- be the number of pixels that are incorrectly classified as negative by the model

Then the ARAND index is given by

$$\text{ARAND} = \frac{c^+ + c^-}{c^+ + c^- + b^+ + b^-} \tag{A.45}$$

A.4.2 Tabulated results

Hence the following table shows mean and standard deviation for the ARAND score, the L-2 pixel distance, the average L-2 distance in millimeters, the time the model takes to find a bone outline data, and the time taken to build the potential function for each test batch. The experiments are repeated for $Q = 2, 4, 6, 8, 10$ using four potential functions. The potential functions generated using the true outline, a gradient based edge detector, u-net (5.4.2.1) and p-net (5.4.3.1). We extract the pixel spacing in millimetres from the dicom file that stores the X-ray image.

			ARAND		l2-distance pixel		l2_distance_mm		fitting time	potential time
			mean	std	mean	std	mean	std	mean	mean
Q	bone	potential								
2	RH2DP	edge	0.71	0.16	9.09	6.18	1.13	0.82	0.33	2.36
		pnet	0.76	0.23	9.31	9.66	1.23	1.36	0.51	149.88
		true	0.93	0.03	2.56	1.03	0.32	0.14	0.44	0.84
		unet	0.70	0.16	9.13	5.19	1.15	0.76	0.43	3.88
	RH2MC	edge	0.64	0.33	44.89	60.80	5.81	7.93	0.41	2.36
		pnet	0.66	0.28	29.53	24.59	3.87	3.41	0.50	150.06
		true	0.92	0.07	5.81	3.32	0.71	0.42	0.48	0.85
		unet	0.75	0.10	20.75	7.03	2.63	1.09	0.48	3.84
	RH2MP	edge	0.81	0.14	8.66	6.47	1.07	0.79	0.38	2.37
		pnet	0.83	0.20	8.98	11.11	1.16	1.53	0.48	150.53
		true	0.93	0.03	3.38	1.73	0.42	0.22	0.44	0.84
		unet	0.81	0.14	8.96	6.52	1.15	0.98	0.45	3.84
	RH2PP	edge	0.81	0.20	14.37	20.45	1.77	2.51	0.40	2.37
		pnet	0.78	0.26	15.34	18.87	2.03	2.68	0.52	151.40
		true	0.94	0.03	4.09	1.94	0.51	0.27	0.46	0.85
		unet	0.85	0.12	10.84	10.60	1.42	1.57	0.46	3.82

Continued on next page

			ARAND		l2-distance pixel		l2_distance_mm		fitting time	potential time
			mean	std	mean	std	mean	std	mean	mean
Q	bone	potential								
4	RH2DP	edge	0.65	0.15	9.49	4.64	1.18	0.64	0.51	2.34
		pnet	0.75	0.25	9.94	10.56	1.31	1.46	0.73	149.83
		true	0.95	0.02	1.79	0.70	0.22	0.08	0.66	0.85
		unet	0.63	0.17	10.09	4.84	1.28	0.73	0.64	3.83
	RH2MC	edge	0.57	0.33	56.28	67.49	7.31	8.86	0.57	2.37
		pnet	0.63	0.31	30.45	24.85	4.02	3.42	0.73	144.97
		true	0.94	0.05	3.89	2.31	0.47	0.26	0.67	0.88
		unet	0.72	0.09	23.48	8.34	3.04	1.33	0.67	3.78
	RH2MP	edge	0.81	0.15	9.07	7.12	1.13	0.90	0.53	2.34
		pnet	0.86	0.18	7.94	11.38	1.07	1.64	0.66	148.09
		true	0.96	0.03	1.92	1.27	0.24	0.19	0.68	0.85
		unet	0.81	0.13	10.23	8.01	1.34	1.20	0.69	3.78
	RH2PP	edge	0.80	0.20	15.17	22.51	1.85	2.70	0.59	2.36
		pnet	0.80	0.26	13.69	18.29	1.80	2.48	0.75	151.58
		true	0.96	0.02	2.60	1.38	0.31	0.15	0.70	0.86
		unet	0.85	0.11	10.20	9.89	1.35	1.50	0.69	3.80

Continued on next page

			ARAND		l2-distance pixel		l2_distance_mm		fitting time	potential time
			mean	std	mean	std	mean	std	mean	mean
Q	bone	potential								
6	RH2DP	edge	0.63	0.14	9.55	4.22	1.19	0.57	0.67	2.35
		pnet	0.77	0.24	9.22	9.95	1.22	1.38	0.96	151.13
		true	0.96	0.01	1.38	0.61	0.17	0.06	0.92	0.84
		unet	0.61	0.16	10.37	4.75	1.32	0.72	0.87	3.78
	RH2MC	edge	0.61	0.32	51.78	68.82	6.73	9.04	0.77	2.34
		pnet	0.66	0.28	27.74	22.48	3.61	3.05	0.96	150.09
		true	0.95	0.06	3.08	1.97	0.37	0.20	0.91	0.86
		unet	0.72	0.10	23.08	9.57	2.98	1.52	0.94	3.82
	RH2MP	edge	0.80	0.14	9.29	7.12	1.15	0.89	0.72	2.36
		pnet	0.86	0.17	7.87	11.60	1.06	1.66	0.89	150.31
		true	0.97	0.02	1.60	1.13	0.20	0.15	0.90	0.84
		unet	0.80	0.13	10.48	8.28	1.37	1.24	0.89	3.80
	RH2PP	edge	0.79	0.19	15.84	23.57	1.96	2.97	0.75	2.34
		pnet	0.80	0.27	14.37	19.02	1.92	2.66	0.98	150.20
		true	0.97	0.01	2.06	1.19	0.25	0.12	0.89	0.84
		unet	0.84	0.12	11.56	10.48	1.53	1.56	0.90	3.76

Continued on next page

			ARAND		l2-distance pixel		l2_distance_mm		fitting time	potential time
			mean	std	mean	std	mean	std	mean	mean
Q	bone	potential								
8	RH2DP	edge	0.61	0.16	9.25	3.47	1.15	0.44	0.83	2.34
		pnet	0.78	0.22	9.05	9.96	1.20	1.42	1.19	150.77
		true	0.96	0.01	1.25	0.41	0.15	0.05	1.16	0.84
		unet	0.60	0.16	10.72	4.77	1.36	0.73	1.07	3.76
	RH2MC	edge	0.62	0.33	50.99	67.52	6.63	8.96	1.07	2.34
		pnet	0.65	0.28	29.57	23.44	3.86	3.19	1.20	150.91
		true	0.96	0.04	2.82	1.55	0.34	0.16	1.20	0.85
		unet	0.70	0.10	23.82	8.76	3.06	1.41	1.10	3.77
	RH2MP	edge	0.79	0.14	9.23	6.24	1.14	0.75	0.92	2.35
		pnet	0.86	0.18	8.00	10.88	1.07	1.55	1.12	150.90
		true	0.97	0.02	1.33	0.84	0.16	0.10	1.21	0.85
		unet	0.80	0.13	10.65	8.36	1.39	1.25	1.08	3.77
	RH2PP	edge	0.78	0.20	16.10	24.28	1.99	3.04	0.93	2.35
		pnet	0.80	0.27	13.20	17.22	1.77	2.45	1.21	155.00
		true	0.97	0.01	1.71	0.77	0.21	0.08	1.21	0.86
		unet	0.83	0.12	12.05	10.39	1.60	1.57	1.10	3.78

Continued on next page

			ARAND		l2-distance pixel		l2_distance_mm		fitting time	potential time
			mean	std	mean	std	mean	std	mean	mean
Q	bone	potential								
10	RH2DP	edge	0.59	0.17	8.98	3.40	1.13	0.41	0.98	2.24
		pnet	0.75	0.23	10.02	9.60	1.35	1.37	1.48	146.08
		true	0.96	0.03	1.07	0.45	0.14	0.06	1.51	0.80
		unet	0.62	0.17	10.12	4.92	1.32	0.79	1.28	3.72
	RH2MC	edge	0.61	0.33	50.35	67.22	6.61	9.08	1.31	2.33
		pnet	0.66	0.27	27.54	22.73	3.60	3.14	1.47	148.72
		true	0.96	0.06	2.59	1.46	0.32	0.16	1.59	0.83
		unet	0.70	0.11	22.89	8.67	2.95	1.40	1.44	3.78
	RH2MP	edge	0.78	0.14	8.96	5.86	1.10	0.71	1.14	2.32
		pnet	0.85	0.19	8.19	11.64	1.09	1.62	1.32	148.87
		true	0.97	0.01	1.15	0.49	0.14	0.06	1.41	0.82
		unet	0.79	0.14	10.93	7.84	1.42	1.17	1.37	3.77
	RH2PP	edge	0.78	0.20	14.85	20.87	1.84	2.62	1.22	2.32
		pnet	0.79	0.29	13.84	19.38	1.84	2.69	1.51	149.11
		true	0.98	0.01	1.50	0.61	0.18	0.07	1.40	0.82
		unet	0.83	0.12	12.04	10.98	1.61	1.67	1.37	3.76

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Luigi Ambrosio and Vincenzo Maria Tortorelli. Approximation of functional depending on jumps by elliptic functional via t-convergence. *Communications on Pure and Applied Mathematics*, 43(8):999–1036, 1990.

Yaniv Bar, Idit Diamant, Lior Wolf, and Hayit Greenspan. Deep learning with non-medical training used for chest pathology identification. In *Medical Imaging 2015: Computer-Aided Diagnosis*, volume 9414, page 94140V. International Society for Optics and Photonics, 2015.

Mario Bertero and Patrizia Boccacci. *Introduction to inverse problems in imaging*. CRC press, 1998.

J Gordon Betts, Peter DeSaix, Eddie Johnson, Jody E Johnson, Oksana Korol, Dean H Kruse, Brandon Poe, James A Wise, Kelly A Young, et al. Anatomy and physiology. 2014.

Neill DF Campbell and Jan Kautz. Learning a manifold of fonts. *ACM Transactions on Graphics*, 33(4), 2014.

John Canny. A computational approach to edge detection. In *Readings in computer vision*, pages 184–203. Elsevier, 1987.

Tim F Cootes, Mircea C Ionita, Claudia Lindner, and Patrick Sauer. Robust and accurate shape model fitting using random forest regression voting. In *European Conference on Computer Vision*, pages 278–291. Springer, 2012.

Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active Appearance Model. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 23(6), June 2001.

David Cristinacce and Timothy F Cootes. Feature detection and tracking with constrained local models. In *Bmvc*, volume 1, page 3. Citeseer, 2006.

Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12 (Jul):2121–2159, 2011.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

Gareth J Edwards, Timothy F Cootes, and Christopher J Taylor. Face recognition using active appearance models. In *European conference on computer vision*, pages 581–595. Springer, 1998.

BWAC Farley and W Clark. Simulation of self-organizing systems by digital computer. *Transactions of the IRE Professional Group on Information Theory*, 4(4):76–84, 1954.

Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 61(1):55–79, 2005.

Fabian Fröhlich. Approximation and analysis of probability densities using radial basis functions. Master's thesis, Technische Universität München, Germany, 2013.

Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

Michael B Giles. Multilevel monte carlo path simulation. *Operations Research*, 56(3): 607–617, 2008.

Hayit Greenspan, Bram van Ginneken, and Ronald M Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016.

Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

Anil K Jain and Farshid Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern recognition*, 24(12):1167–1186, 1991.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed ¡today¿].

Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

Joost A. Kauffman. *Automated radiographic assessment of hands in rheumatoid arthritis*. PhD thesis, 2009.

David G Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121, 1984.

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

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

Jan J Koenderink. The structure of images. *Biological cybernetics*, 50(5):363–370, 1984.

Julia Kruger, Jan Ehrhardt, and Heinz Handels. Probabilistic appearance models for segmentation and classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1698–1706, 2015.

Julia Krüger, Jan Ehrhardt, and Heinz Handels. Statistical appearance models based on probabilistic correspondences. *Medical image analysis*, 37:146–159, 2017.

Neil Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005.

Neil D Lawrence and Joaquin Quiñonero-Candela. Local distance preservation in the gp-lvm through back constraints. In *Proceedings of the 23rd international conference on Machine learning*, pages 513–520. ACM, 2006.

Tony Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(1-2):225–270, 1994.

Tony Lindeberg. Scale-space theory. 2001.

Claudia Lindner, Shankhar Thiagarajah, J Mark Wilkinson, Gillian A Wallis, Timothy F Cootes, arcOGEN Consortium, et al. Fully automatic segmentation of the proximal femur using random forest regression voting. *IEEE transactions on medical imaging*, 32(8):1462–1472, 2013.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

Luca Minciullo, Matthew J Parkes, David T Felson, and Timothy F Cootes. Comparing image analysis approaches versus expert readers: the relation of knee radiograph features to knee pain. *Annals of the rheumatic diseases*, 77(11):1606–1609, 2018.

Jan Modersitzki. *Numerical methods for image registration*. Oxford university press, 2003.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, volume 269, pages 543–547, 1983.

Alexis Ogdie, SinÃ©ad Langan, Thorvardur Love, Kevin Haynes, Daniel Shin, Nicole Seminara, Nehal N. Mehta, Andrea Troxel, Hyon Choi, and Joel M. Gelfand. Prevalence and treatment patterns of psoriatic arthritis in the UK. *Rheumatology*, 52 (3):568–575, 12 2012. ISSN 1462-0324. doi: 10.1093/rheumatology/kes324. URL https://doi.org/10.1093/rheumatology/kes324.

Victor Adrian Prisacariu and Ian Reid. Nonlinear shape manifolds as shape priors in level set segmentation and tracking. In *CVPR 2011*, pages 2185–2192. IEEE, 2011.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

P Rahman, DD Gladman, RJ Cook, Y Zhou, G Young, and D Salonen. Radiological assessment in psoriatic arthritis. *Rheumatology*, 37(7):760–765, 1998.

Adwaye M Rambojun, William Tillett, Neill DF Campbell, and Tony Shardlow. 032 a novel human-assisted computer algorithm for identification of hand bones on plain radiographs in psoriatic arthritis. *Rheumatology*, 58(Supplement_3):kez106–031, 2019.

William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

Carl Edward Rasmussen. Gaussian processes for machine learning. In . MIT Press, 2006.

O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. URL http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a. (available on arXiv:1505.04597 [cs.CV]).

LI Rubinšteĭn. *The stefan problem*, volume 8. American Mathematical Soc., 2000.

James AJ Rynn, Simon L Cotter, Catherine E Powell, and Louise Wright. Surrogate accelerated bayesian inversion for the determination of the thermal diffusivity of a material. *Metrologia*, 56(1):015018, 2019.

John T Sharp, Gilbert B Bluhm, Andrew Brook, Anne C Brower, Mary Corbett, John L Decker, Harry K Genant, J Philip Gofton, Neal Goodman, Arvi Larsen, et al. Reproducibility of multiple-observer scoring of radiologic abnormalities in the hands and wrists of patients with rheumatoid arthritis. *Arthritis & Rheumatism*, 28 (1):16–24, 1985.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Mark Summerfield. *Rapid Gui Programming with Python and Qt*. Hall, Prentice, 1st edition, 2007.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

William Tillett, Gavin Shaddick, Deepak Jadon, Graham Robinson, Eleanor Korendowych, and Neil McHugh. Novel Composite Radiographic Score for Longitudinal Observational Studies of Psoriatic Arthritis: A Proof-of-concept Study. *The Journal of rheumatology*, pages jrheum–150114, 2016.

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61 (3):611–622, 1999.

Michalis K Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

Michalis K Titsias and Neil D Lawrence. Bayesian Gaussian process latent variable model. In *International Conference on Artificial Intelligence and Statistics*, pages 844–851, 2010.

DMFM Van der Heijde, H PAULUS, and P SHEKELLE. How to read radiographs according to the Sharp/van der Heijde method. Discussion: Heterogeneity in rheumatoid arthritis radiographic trials. Issues to consider in a metaanalysis. *Journal of rheumatology*, 27(1):261–263, 2000.

DMFM Van der Heijde, J Sharp, S Wassenberg, and DD Gladman. Psoriatic arthritis imaging: a review of scoring methods. *Annals of the Rheumatic Diseases*, 64(suppl 2):ii61–ii64, 2005.

Luminita A Vese and Tony F Chan. A multiphase level set framework for image segmentation using the Mumford and Shah model. *International journal of computer vision*, 50(3):271–293, 2002.

Jack Wang, Aaron Hertzmann, and David J Fleet. Gaussian process dynamical models. In *Advances in neural information processing systems*, pages 1441–1448, 2006.

S Wassenberg, V Fischer-Kahle, G Herborn, and R Rau. A method to score radiographic change in psoriatic arthritis. *Zeitschrift für Rheumatologie*, 60(3):156–166, 2001.

Paul Werbos. Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974.

Laurent Younes. *Shapes and diffeomorphisms*, volume 171. Springer Science & Business Media, 2010.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.