UNCERTAINTY QUANTIFICATION, IMAGE SYNTHESIS AND DEFORMATION PREDICTION FOR IMAGE REGISTRATION

Xiao Yang

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2017

Approved by:

Marc Niethammer

Roland Kwitt

J. S. Marron

Martin Styner

Vladimir Jojic

ABSTRACT

Xiao Yang: UNCERTAINTY QUANTIFICATION, IMAGE SYNTHESIS AND DEFORMATION PREDICTION FOR IMAGE REGISTRATION
(Under the direction of Marc Niethammer)

Image registration is essential for medical image analysis to provide spatial correspondences. It is a difficult problem due to the modeling complexity of image appearance and the computational complexity of the deformable registration models. Thus, several techniques are needed: Uncertainty measurements of the high-dimensional parameter space of the registration methods for the evaluation of the registration result; Registration methods for registering healthy medical images to pathological images with large appearance changes; Fast registration prediction techniques for uni-modal and multi-modal images.

This dissertation addresses these problems and makes the following contributions: 1) *A framework for uncertainty quantification of image registration results is proposed.* The proposed method for uncertainty quantification utilizes a low-rank Hessian approximation to evaluate the variance/covariance of the variational Gaussian distribution of the registration parameters. The method requires significantly less storage and computation time than computing the Hessian via finite difference while achieving excellent approximation accuracy, facilitating the computation of the variational approximation; 2) *An image synthesis deep network for pathological image registration is developed.* The network transforms a pathological image into a 'quasi-normal' image, making registrations more accurate; 3) *A patch-based deep learning framework for registration parameter prediction using image appearances only is created.* The network is capable of accurately predicting the initial momentum for the Large Deformation Diffeomorphic Metric Mapping (LDDMM) model for both uni-modal and multi-modal registration problems, while increasing the registration speed by at

least an order of magnitude compared with optimization-based approaches and maintaining the theoretical properties of LDDMM.

Applications of the methods include 1) Uncertainty quantification of LDDMM for 2D and 3D medical image registrations, which could be used for uncertainty-based image smoothing and subsequent analysis; 2) Quasi-normal image synthesis for the registration of brain images with tumors with potential extensions to other image registration problems with pathologies and 3) deformation prediction for various brain datasets and T1w/T2w magnetic resonance images (MRI), which could be incorporated into other medical image analysis tasks such as fast multi-atlas image segmentation, fast geodesic image regression, fast atlas construction and fast user-interactive registration refinement.

ACKNOWLEDGEMENTS

First of all I would like to express my greatest appreciation for my Ph.D advisor Dr. Marc Niethammer. His patience and wisdom when guiding me through research problems has been crucial for my past 5 years. Besides, the freedom I got from him to explore various research and career opportunities has been priceless. Furthermore, he is an awesome person, and has always supported me in research and in life. I cannot thank him enough for this. Finally, his great collection of books helped me a lot in various aspects of my research, and I greatly appreciate that (thank you for the "Clean Code" book by the way! xiè xiè nǐ xiǎo mǎ gē!).

I would also like to thank Dr. Roland Kwitt for the collaboration for the past two years. It was always a joy working, brainstorming, and discussing with him. I also learned a lot of paper writing skills from him (xiè xiè nǐ nán gē!). Also, I would like to thank my other thesis committee members, Dr. Steve Marron, Dr. Martin Styner and Dr. Vladimir Jojic for their suggestions for and help with my thesis. Specifically, Dr. Steve Marron has been very helpful for all statistical questions, and his expertise has helped me tremendously. Dr. Martin Styner has been my go to person for choosing appropriate medical image datasets for all my experiments. Dr. Vladimir Jojic has given me many valuable suggestions related to machine learning.

I am grateful for all my other collaborators, including Dr. Stephen Aylward, Xu Han and Eunbyung Park for their great contributions to all my publications. I also want to thank all my fellow students in the lab: Dr. Liang Shan, Dr. Tian Cao, Dr. Yi Hong, Xu Han, and Heather Couture, for all the discussions that helped the progress of my research. Special thanks go to Eunbyung Park for his guidance and for giving me the initiative to do deep learning research, and to Nikhil Singh for his help with the PyCA package and the OASIS dataset.

Finally I want to thank my parents, Chaokun Yang and Yanrong Lv, for their everlasting support and love throughout the years. Also thanks to all my friends in and out of Chapel Hill for their support. You bring a lot of fun to my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

LDDMM	Large Deformation Diffeomorphic Metric Mapping
LO	LDDMM optimization
LP	LDDMM prediction network
LPC	LDDMM prediction + correction network

CHAPTER 1: Introduction

1.1 Motivation

Image registration is a crucial task for medical image analysis. It is the process of finding the best deformation to spatially transform a moving image to a target image. Image registration is widely used, for example, to detect shape changes for image time-series (Evans, 2006), for treatment planning (Rosenman et al., 1998), to monitor disease progression (Fripp et al., 2007) and for multi-model image alignment (Caplan et al., 2011). Many registration methods have been proposed for various registration tasks, but some general problems remain for all these methods:

1. Most registration methods do not provide statistical evaluations of the registration result;
2. General registration methods cannot deal with appearance inconsistencies due to pathologies;
3. Registration methods, especially non-parametric ones, have a large computation time requirement.

I provide a detailed discussion of the three aforementioned problems below.

1.1.1 Uncertainty Quantification for Image Registration

As stated, many registration methods have been proposed to provide high-quality spatial deformations for images. However, most of them do not provide any measures of registration uncertainty. Hence these methods do not enable users to assess the trustworthiness of the deformation. Furthermore, uncertainty measures could be useful for follow-up image analysis tasks, such as uncertainty-based smoothing as in (Simpson et al., 2011). Not having uncertainty quantification is particularly problematic in the case of complex, deformable (e.g., elastic, fluid, diffusion) registration methods as these methods have a large number of parameters to model a deformation, and

the high flexibility of these deformation models can result in large ambiguities of the generated deformations, making uncertainty quantification even more necessary.

Different approaches for modeling image registration uncertainty have been proposed. For affine registration, physical landmarks have been used to estimate the distribution of the target registration error for the whole image volume (Fitzpatrick and West, 2001). This approach, however, does not easily apply to non-parametric registration methods since landmarks only assess local deformations. Thus a large number of landmarks is required to assess deformable image registration approaches, which requires work-extensive manual labeling. Because of this, methods have been introduced to quantify different types of registration uncertainty using probabilistic models of the registration model and the image itself[1]. (Kybic, 2010) proposes a bootstrap sampling strategy to sample candidate voxels to be used for the image matching function of a cubic B-spline model, and computes registrations for every sampled subset of the image voxels. The resulting deformations are used to calculate the variance of the displacements quantifying the registration uncertainty. (Watanabe and Scott, 2012) also evaluate the registration uncertainty of a B-spline model by generating multiple deformation fields, but they instead sample registration parameters. Specifically, after the registration parameters are generated via optimizing the registration energy, they assume a Gaussian distribution for registration parameters on each B-spline grid point. Then they sample the Gaussian distribution to generate multiple deformation fields. These deformation fields are used to warp the moving image to get multiple synthesized target images. After that, they register the moving image to these synthesized target images, and compute the deformation errors between the synthesized deformations and the registration result. The distribution of the deformation errors are then approximated using a Gaussian distribution, and the variance of the Gaussian is used as the uncertainty of the deformation. More complex methods include (Simpson et al., 2011) where a variational Bayesian model is used to model the posterior probability of the deformation parameters

[1]Note that evaluating image registration errors directly requires landmarks and/or labels for region of interest. However, other measures of registration uncertainty, for example, how a registration result depends on the image itself are also of interest

of a B-spline registration, and Monte-Carlo sampling methods for elastic registration (Risholm et al., 2013) and LDDMM (Zhang et al., 2013).

Existing methods mainly focus on parametric deformable registration models such as the B-spline method, or require sampling of a high-dimensional deformation parameter space (e.g., for the LDDMM model), which is computationally demanding. Thus, an efficient approach for uncertainty quantification for non-parametric complex deformation models is very much needed.

For this problem, I propose a variational approximation method for uncertainty quantification of the registration model in a time-efficient manner. Chapter 2.2.3.4 describes the method and shows experimental results.

1.1.2 Image Reconstruction for Pathological Image Registration

Atlas-to-image registration provides spatial information to map anatomical locations from the atlas image to a specific patient. This procedure is crucial for atlas-based segmentation, which is a common technique for example to segment lesions in brains with pathologies (Cabezas et al., 2011). However, large pathologies may produce appearances vastly different from normal image appearances. This can result in large misregistration if the appearance changes are falsely accounted for by the deformation. This is especially problematic for deformable registration models, where the flexibility of the model allows for localized deformations. Since deformation registration models are often used to capture subtle deformations and mass effects of pathologies (Zacharaki et al., 2009), solving the misregistration problem becomes important.

Many methods have been proposed for atlas-to-image registration with large pathologies. They can be separated into several general approaches. The first group of methods requires a pathology segmentation. The simplest approach is cost function masking (Brett et al., 2001). In this method, the lesion area is ignored when calculating image similarity. This method is very general but can be problematic if the pathology areas contain important anatomical structures. Other methods that fall into this group are:

- Joint segmentation and registration methods that mitigate missing correspondences via setting the loss function value of the segmented pathological area to be constant (Chitphakdithai and Duncan, 2010);
- Geometric metamorphosis that separates the deformation of healthy brain areas from lesion changes (Niethammer et al., 2011);
- Registration methods accounting for deformation and intensity changes that require user input of pathology location as initializations of the segmentation step (Wang et al., 2012).

While effective, the required segmentation or localization may not be available for general use. The second group of methods does not require user input. Instead, these methods attempt to model pathology changes mathematically, for example, by using tumor growth models for deformable image registration of brain images (Gooya et al., 2012; Kyriacou et al., 1999). For this group of methods, the effectiveness of the mathematical model strongly affects the registration accuracy, and they are limited to the specific registration problems the models are designed for.

Apart from the aforementioned methods, two alternatives exist: (1) using a robust cost function (Reuter et al., 2010) or a mutual saliency map (Ou et al., 2011) to mitigate the effect of outliers (pathology in this case) or, instead, (2) learning desired mappings between image types from large-scale image databases. Nowadays, given the increasing amount of image data available, the second approach has drawn much attention. A learned mapping allows for image synthesis from one image type to another. This technique has been extensively explored to synthesize magnetic resonance imaging sequences (Jog et al., 2013), to facilitate multi-modality registration (Roy et al., 2014; Cao et al., 2014) and to segment lesions (Roy et al., 2010). For our problem, the goal is to synthesize quasi-normal images from images with lesions to simplify atlas-to-image registration in the presence of pathologies. Using image synthesis rather than a robust cost-function or a mutual saliency map allows reconstructing structural information to guide registration even in highly pathological areas. One existing method for image synthesis is the low-rank-plus-sparse (LRS) technique (Liu et al., 2015). This method synthesizes quasi-normal brain images from pathological images and simultaneously estimates a quasi-normal atlas. This approach decomposes images into

normal (low-rank) and lesion (sparse) parts. The low-rank part then constitutes the synthesized quasi-normal images, effectively removing lesion effects. By learning from data, no prior lesion information is required. However, the LRS decomposition itself requires good image alignment, hence decomposition and registration have to be interleaved to obtain good results. Hence, a method that synthesizes quasi-normal images in an end-to-end fashion without additional registration and tumor modeling would be very beneficial.

I propose, in chapter 3.4.3, a variational autoencoder framework for quasi-normal image synthesis that fulfills the needs stated above and provides competitive registration accuracy.

1.1.3 Deep Learning for Image Registration Prediction

Image registration is typically formulated as an optimization problem (Modersitzki, 2004), optimizing the parameters of a transformation model to achieve the best possible agreement between a transformed source and a target image, subject to transformation constraints. Apart from simple low-dimensional parametric models (e.g., rigid or affine transformations), more complex, high-dimensional parametric or non-parametric registration models are able to capture subtle, localized image deformations. However, these methods, in particular, the non-parametric approaches, have a very large numbers of parameters. Therefore, numerical optimization to solve the registration problems becomes computationally costly, even with acceleration by graphics processing units (GPUs).

While computation time may not be overly critical for imaging studies of moderate size, the currently increasing sample sizes and image sizes of medical image data frequently results in the registration process being very time-consuming. As a case in point, the UK Biobank study is, at the moment, the world's largest health imaging study and will image "the brain, bones, heart, carotid arteries and abdominal fat of 100,000 participants" using magnetic resonance (MR) imaging within the next few years[2]. Furthermore, the voxel sizes of MR images of human brains have decreased from 2x2x2 mm^3 not too long ago to smaller than 1x1x1 mm^3 as in the human connectome project (Van Essen et al., 2013). Attempts at speeding-up deformable image registration

[2] `www.ukbiobank.ac.uk`

have primarily focused on GPU implementations (Shams et al., 2010), but these approaches are still relatively slow. For example, a GPU-based registration of a 128x128x128 image volume using the Large Deformation Diffeomorphic Metric Mapping (LDDMM) registration model will take about 10 minutes on a current GPU (e.g., an Nvidia TitanX), which is too slow for large-scale image analysis tasks. Thus, very fast deformable registration approaches are needed to time- and cost-efficiently analyze very large imaging studies and to allow their use as building blocks for more advanced image analysis algorithms.

Recent work has focused on accelerating computations, for example, by using a finite-dimensional approximation of LDDMM (Zhang and Fletcher, 2015) which achieves a roughly 25x speed-up over a standard LDDMM optimization-based solution for 3D images of 128x128x128. An alternative approach to improve registration speed is to predict deformation parameters, or deformation parameter update steps in the optimization via a regression model, instead of directly minimizing a registration energy (Gutiérrez-Becker et al., 2016; Chou et al., 2013; Wang et al., 2015; Cao et al., 2015). The resulting predicted deformation fields can either be used directly or as an initialization of a subsequent optimization-based registration. However, the high dimensionality of the deformation parameters, as well as the highly non-linear relationship between the images and the parameters, poses a significant challenge. Among these methods, (Chou et al., 2013) propose a multi-scale linear regressor which only applies to affine deformations and low-rank approximations of non-linear deformations. (Wang et al., 2015) predict deformations by key-point matching using sparse learning followed by dense deformation field generation with radial basis function interpolation. The performance of their method heavily depends on the accuracy of the key point selection. (Cao et al., 2015) use a semi-coupled dictionary learning method to directly model the relationship between image appearance and the deformation parameters of the LDDMM model (Beg et al., 2005). However, only a linear relationship is assumed between image appearance and the deformation parameters. Lastly, (Gutiérrez-Becker et al., 2016) use a regression forest based on hand-crafted features to learn update steps for a B-spline registration model. While these methods work well in their constrained problem settings, there is a need for a more powerful, end-to-end deformation prediction method

that 1) models the highly nonlinear relationship between the image appearance and deformation parameters; 2) only takes images as inputs and 3) can directly predict the deformation with high accuracy and fast computation speed.

In chapter 5, I propose a deep learning framework for fast deformation prediction that achieves at least an order of magnitude of speedup compared with a GPU-accelerated optimization-based registration approach, and at the same time achieves state-of-the-art registration accuracy.

1.1.4 Deep Learning for Augmented Deformation Prediction

An extension of the previous problem is augmented deformation prediction, where apart from predicting deformations, the model also learns additional image features that help address more complex image registration problems. One example is multi-modal image registration. Multi-modal image registration seeks to estimate spatial correspondences between image pairs from different imaging modalities (or protocols). In general image registration, these correspondences are estimated by finding the spatial transformation which makes a transformed source image most similar to a fixed target image. For unimodal image registration, image similarity should be high if images are close to identical, which allows using simple image similarity measures such as the sum of squared intensity differences (SSD) between image pairs. Assessing image similarity across modalities is substantially more difficult as image appearance can vary widely, e.g., due to different underlying physical imaging principles. In fact, these differences are desired as they can, for example, highlight different tissue properties in brain imaging. Hence, more sophisticated multi-modal similarity measures are required. Furthermore, image registration results are driven by both the chosen similarity measure and the chosen deformation model. Hence, especially for multi-modal image registration, where assessing image similarity becomes challenging, considering the similarity measure jointly with the deformation model is important.

The most popular multi-modal similarity measure is mutual information (MI) (Viola and Wells, 1997), but other *hand-crafted* multi-modal similarity measures have also been proposed (Meyer et al., 1998; Hermosillo et al., 2002; Lorenzen et al., 2006). These approaches *assume* properties characterizing good image alignment instead of *learning* them from data. Hence, more recently,

learning-based approaches to measure image similarity have been proposed. These techniques include measuring image similarity by comparing observed and learned intensity distributions via KL-divergence (Guetter et al., 2005); or learning the similarity of image pixels/voxels or image features (e.g., Fourier/Gabor features) via max-margin structured output learning (Lee et al., 2009), boosting (Michel et al., 2011), or deep learning (Cheng and Zheng, 2015; Simonovsky et al., 2016). Some methods avoid a complex similarity measure by applying image synthesis to the source or target image to change the task to unimodal registration (Roy et al., 2013; Jog et al., 2013; Van Nguyen et al., 2015). However, the registration performance then heavily depends on the synthesis accuracy. Moreover, these similarity measures still require optimizations to register images. Hence, the large computation cost for large-scale image data analysis is still problematic. In summary, a new method that simultaneously models the deformation parameters *and* the similarity measure from the different imaging modalities would be desirable.

For this problem, I propose, in chapter 6, applying the deep learning method from chapter 5 to show the validity of the network in learning complex image similarity measures while, at the same time, predicting accurate image deformations.

1.2 Thesis Statement

Thesis: *Registration uncertainty can be modeled using a variational approximation that exploits the low-rank property of the image similarity measure in a time- and memory-efficient manner. Furthermore, deep learning can be used for pathological-to-quasi-normal image synthesis and for fast and accurate prediction of uni-modal and multi-modal registrations*.

The main contributions of the thesis are:

1. *For uncertainty quantification for image registration:* I create a Laplace approximation based method to model the local multivariate Gaussian distribution at the optimal solution for the LDDMM method, and an efficient approach to accurately approximate the covariance matrix of the approximated Gaussian distribution as the uncertainty measure.
2. *For quasi-normal image synthesis:* I propose a variational deep learning approach to synthesize a quasi-normal image from an image with a large pathology which only requires

pathology segmentation at training time. Furthermore, image synthesis uncertainty from the network is used to guide image registration for better registration accuracy.

3. *For fast image registration:* I build a general deep network architecture to learn the relationship between deformation parameters and image appearance and to efficiently predict deformation parameters for both uni-modal and multi-modal image registration applications.

1.3 Overview of Chapters

The thesis is organized as follows: Chapter 2 introduces background for image registration, diffeomorphic deformations and deep learning. Chapter 2.2.3.4 discusses the variational approach to model uncertainty of LDDMM registration. Chapter 3.4.3 discusses the deep network image synthesis method for pathological image registration. Chapter 5 proposes a deep learning method for fast deformation prediction. Chapter 6 extends the previous chapter to the augmented deformation prediction problem, and specifically discusses the multi-modal image deformation prediction task. Chapter 7 concludes the thesis with a discussion of its contributions and potential future research directions.

CHAPTER 2: Background

This chapter provides necessary background to this dissertation. Section 2.1 introduces image registration which is used for all the projects in my dissertation. I will cover various types of image registration models, discuss the theory of diffeomorphic deformation, and introduce the LDDMM registration model, which is a powerful non-parametric image registration model that, in theory, guarantees diffeomorphic deformations, and is heavily explored in my dissertation. Section 2.2 introduces the basics of deep learning, discusses convolutional networks, which is the basic deep network structure for modern computer vision research, summarizes current applications of deep networks for medical image analysis problems, and briefly review the loss functions for different deep learning tasks.

2.1 Image Registration

Given a moving image M and target image T, image registration aims to find a deformation map $\Phi : \mathbb{R}^d \to \mathbb{R}^d$, where d is the spatial dimension of the image, to map the moving image to the target image in such a way that the deformed moving image is similar, in some similarity measures, to the target image, i.e., $M \circ \Phi^{-1}(x) \approx T(x)$. Due to the importance of image registration, a large number of different approaches have been proposed (Modersitzki, 2004; Hill et al., 2001; Sotiras et al., 2013; Oliveira and Tavares, 2014). Typically, these approaches are formulated as optimization problems, where one seeks to minimize an energy of the form

$$E(\Phi) = \mathrm{Reg}[\Phi] + \frac{1}{\sigma^2}\mathrm{Sim}[I_0 \circ \Phi^{-1}, I_1] \ , \tag{2.1}$$

where $\sigma > 0$ is a balancing constant, $\mathrm{Reg}[\cdot]$ regularizes the spatial transformation, Φ, by penalizing spatially irregular (for example, non-smooth) spatial transformations, and $\mathrm{Sim}[\cdot, \cdot]$ is an image

dissimilarity measure, which becomes small if images are similar to each other. In general, one distinguishes between parametric and non-parametric transformation models (Modersitzki, 2004). Parametric registration methods model the deformation via a low dimensional parameterization. Non-parametric registration methods, on the other hand, model the deformation locally, with a parameter (or a parameter vector) for each voxel. Choosing a suitable deformation model depends on the registration task, specifically, the object to register to and the expected transformations. For example, for simple registrations of rigid bodies, the rigid transformation model is enough; for inter-subject non-rigid organ registrations, the elastic model and the fluid model can be better choices. I discuss various kinds of deformation models below.

2.1.1 Parametric Registration Models

Among the parametric registration models, the most commonly used ones are:

- *Affine transformation.* This deformation model only models transformations that preserve collinearity. In general, an affine transformation models translations, rotations, scaling and shears and aspect ratio changes. Mathematically, an affine transformation maps pixel/voxels from the position $\overrightarrow{x}$ in the moving image to the position $\overrightarrow{y}$ in the target image by the following equation:

$$\overrightarrow{y} = A\overrightarrow{x} + \overrightarrow{b}$$

  where A is a 2×2 matrix for 2D images and 3×3 for 3D images. A even simpler variant of the affine transformation is rigid transformation, which only models scaling, rotation and translation. This deformation model is simple and has been widely implemented for fast image alignment, or as initialization of complex registration models.
- *B-spline deformation.* B-spline registration (Rueckert et al., 1999) parameterizes the deformation by representing the image space with a grid. By moving the grid points which imply a dense spatial transformation, via a B-spline interpolation, the whole deformation field can be calculated.

2.1.2 Non-parametric Registration Models

As discussed before, non-parametric registration models parameterize the transformation locally, so their numbers of parameters are much larger compared to parametric methods, and the optimization process is more complex. The most direct non-parametric approach is to represent voxel displacements as $u(x) = \Phi(x) - x$, where $\Phi(x)$ is the deformation map and x is the identity map. Regularization is necessary for non-parametric approaches to avoid ill-posedness of the optimization problem, and to ensure smoothness of the deformation. There are several notable non-parametric registration methods with different parameterizations and regularizations (Modersitzki, 2004), as shown below.

Elastic registration. This registration method is based on linear elastic theory (Broit, 1981). For elastic registration, the moving and target images are regarded as elastic bodies before and after the deformation. The deformation parameter is the displacement field $u(x)$, and the regularization energy is

$$\int_{\Omega} \frac{\nu}{4} \sum_{j,k=1}^{d} (\partial_{x_j} u_k + \partial_{x_k} u_j)^2 + \frac{\lambda}{2} (\text{div } u)^2 dx \ , \tag{2.2}$$

where d is the dimension of the image data, x indicates the pixel/voxel location, ν and λ denote the so-called Lamé constants and Ω denotes the imace space.

Fluid registration. In fluid registration (Bro-Nielsen and Gramkow, 1996), the image deformation is described as a time-dependent process, where the displacement u is a function of location x and time t. Typically $t \in [0, 1]$. The biggest difference between the elastic registration and the fluid registration is the regularization energy. In elastic registration, the regularization penalizes the displacement u, while in fluid registration the regularization is on u_t, the derivative of the displacement with respect to time t. In other words, elastic models are characterized by a smooth displacement field, while fluid models are characterized by a smooth velocity field that changes along the time. For fluid models, the regularizer can be written as

$$\nu \nabla^2 v + (\lambda + \nu) \nabla \nabla \cdot v, \ v = \partial_t u \ . \tag{2.3}$$

The PDE of transforming the image over time depends on the registration model formulation.. For example, in LDDMM the image transport equation is

$$I_t + \nabla I^T v = 0$$

where I is the image and $v = u_t$ is the velocity.

Diffusion registration. This registration model is proposed by (Horn and Schunck, 1993), and it is different from both elastic registration and fluid registration since it is not motivated by physical models, but by smoothing properties of the displacement. The regularizer of the diffusion registration is given by (Horn and Schunck, 1993) as

$$\frac{1}{2}\sum_{l=1}^{d}\int_{\Omega}||\nabla u_l||^2 dx \quad . \tag{2.4}$$

Diffusion registration has a lot of similarities with elastic registration and fluid registration. The regularization energy of the diffusion registration can be seen as the energy for the elastic registration without $(\text{div } u)^2$. It is also very simple to extend diffusion registration to velocity-based methods similar to fluid registration. It is also possible to warp the image multiple times in a greedy fashion, such as the Demon registration algorithm (Thirion, 1998).

Curvature registration. The three mentioned non-parametric registration methods suffer from the same problem: they are sensitive to affine pre-registration. Curvature registration is introduced in (Fischer and Modersitzki, 2004) to circumvent this problem. In curvature registration, the regularizer is

$$\frac{1}{2}\sum_{l=1}^{d}\int_{\Omega}|(\Delta u_l)^2 dx \tag{2.5}$$

The idea of this regularizer is to minimize the curvature of the deformation field. Since this regularizer does not penalize the displacement u or the gradient of the displacement (∇u), it does not penalize affine transformations and hence eliminates the need for affine pre-registration.

2.1.3 Diffeomorphisms

In image registration, diffeomorphic transformations are often (though not always) desirable. A diffeomorphism can be considered as a smooth one-to-one mapping with a smooth inverse. Since this is very important for many medical image registration tasks where the deformation needs to be diffeomorphic and valid warpings to both the moving image space and the target image space are desired (e.g., in brain registration and in atlas-based population analysis), diffeomorphic image registration plays an important role in current medical image analysis research. I now discuss the LDDMM image registration model, which is a fluid-based image registration model that generates diffeomorphic deformations.

2.1.4 LDDMM Relaxation and Shooting based models

LDDMM is a non-parametric registration method which represents the transformation via spatio-temporal velocity fields. In particular, the sought-for mapping, Φ, is obtained via an integration of a spatio-temporal velocity field $v(x,t)$ for unit time, where t indicates time and $t \in [0,1]$, such that $\Phi_t(x,t) = v(\Phi(x,t),t)$ and the sought-for mapping is $\Phi(x,1)$. To single-out desirable velocity-fields, non-spatial-smoothness at any given time t is penalized by the regularizer $\mathrm{Reg}[\cdot]$, which is applied to the velocity field instead of the transform Φ directly. Specifically, LDDMM aims at minimizing the energy[1] (Beg et al., 2005)

$$E(v) = \int_0^1 \|v\|_L^2 \, dt + \frac{1}{\sigma^2}\|M \circ \Phi^{-1}(1) - T\|^2,$$
$$\text{s.t.} \quad \Phi_t(x,t) = v(\Phi(x,t),t), \ \Phi(x,0) = \mathrm{id} \quad (2.6)$$

where $\sigma > 0$, $\|v\|_L^2 = \langle Lv, v\rangle$, L is a self-adjoint differential operator[2], id is the identity map, and the differential equation constraint for Φ can be written in Eulerian coordinates as $\Phi_t^{-1} + D\Phi^{-1}v = 0$, where $\Phi_t(x,t)$ is the derivative of Φ with respect to time t, and D is the Jacobian matrix. In this

[1]When clear from the context, I suppress spatial dependencies for clarity of notation and only specify the time variable. E.g., I write $\Phi^{-1}(1)$ to mean $\Phi^{-1}(x,1)$.

[2]Note that I define $\|v\|_L^2$ here as $\langle Lv, v\rangle$ instead of $\langle Lv, Lv\rangle = \langle L^\dagger Lv, v\rangle$ as for example in Beg et al. (Beg et al., 2005).

LDDMM formulation, which is called the relaxation formulation of LDDMM, the registration is parameterized by the full spatio-temporal velocity field $v(x,t)$. When implementing this algorithm, a discretization for the time t is needed. This results in multiple discrete steps to propagate of Φ from $t=0$ to $t=1$. The number discrete propagation steps is call the number of time steps. The problem with the relaxation formulation of LDDMM is that the deformation parameter is the full spatial-temporal velocity field, which can require a large amount of memory for 3D image registration problems with large number of timesteps. Besides, the forward propagation path from the velocity field is not necessary a geodesic before convergence of the numerical optimization. The solution to this is the *shooting* formulation (Vialard et al., 2012a; Niethammer et al., 2011), which parameterizes the deformation via the initial momentum vector field $m_0 = m(0)$ and the initial map $\Phi^{-1}(0)$, from which the map Φ can be computed for any point in time. The geodesic equations, in turn, correspond to the optimality conditions of Eqn. (2.6). Essentially, the shooting formulation enforces these optimality conditions of Eqn. (2.6) as a constraint. In effect, one then searches only over geodesic paths, as these optimality conditions are geodesic equations. They can be written in terms of the momentum m alone. In particular, the momentum is the dual of the velocity v, which is an element in the reproducing kernel Hilbert space V; m and v are connected by a positive-definite, self-adjoint differential smoothing operator K by $v = Km$ and $m = Lv$, where L is the inverse of K. Given m_0, the complete spatio-temporal deformation $\Phi(x,t)$ is determined.

Specifically, the energy to be minimized for the shooting formulation of LDDMM is (Singh et al., 2013b)

$$E(m_0) = \langle m_0, Km_0 \rangle + \frac{1}{\sigma^2}||M \circ \Phi^{-1}(1) - T||^2, \quad \text{s.t.} \tag{2.7}$$

$$\begin{aligned} m_t + \text{ad}_{*v} m &= 0, \\ m(0) &= m_0, \\ \Phi_t^{-1} + D\Phi^{-1} v &= 0, \\ \Phi^{-1}(0) &= \text{id}, \\ m - Lv &= 0 \ , \end{aligned} \tag{2.8}$$

where id is the identity map, and the operator ad_* is the dual of the negative Jacobi-Lie bracket of vector fields, i.e., $\text{ad}_v w = -[v, w] = Dvw - Dwv$. The optimization approach is similar to the one for the relaxation formulation. I.e., one determines the adjoint equations for the shooting formulation and uses them to compute the gradient with respect to the unknown initial momentum m_0 (Singh et al., 2013b; Vialard et al., 2012a). Based on this gradient an optimal solution can, for example, be found via a line-search or by a simple gradient descent scheme.

2.2 Deep Networks

Learning deep artificial neural networks has become an important topic in the machine learning, computer vision and medical image analysis communities. This is due to the great performance of deep networks, and specifically of deep convolutional networks, on image analysis tasks. Most deep networks (feed forward networks) can be viewed as an extension of the neural network, and can be considered as a generalization of a linear or logistic regression. During training, the network learns a function that maps the network input to the desired output using a training dataset. The parameters being optimized during training are the layer weights in the neural network. Mathematically, given input x, one single layer of a neural network can be written as the following function

$$y = \sigma(W^T x + b) \tag{2.9}$$

where W and b are the weights in the neural network layer, and the $\sigma(\cdot)$ is a non-linear function, and is generally referred to as an 'activation function', which resembles the activation of a neuron during signal transmission. When stacking multiple layers to form a neural network, the resulting function for the whole neural network is

$$y = \sigma_1(W_1^T \sigma_2(W_2^T \sigma_3(...) + b_2) + b_1) \ . \tag{2.10}$$

Such a multi-layer feed forward network is called a multi-layered perceptron. When the number of layers in the network is very large, the network is considered 'deep' and is generally called a 'deep network'.

2.2.1 Deep Convolutional Networks

Deep convolutional networks are deep networks that use convolution layers. A convolution layer is different from a fully connected layer. In a fully connected layer, each neuron is connected to all the neurons in the previous layer. In a convolution layer, however, each neuron is only connected to a few neighboring neurons in the previous layer via a set of weights, and the same set of weights is shared across all neurons in the current layer. This can be viewed as performing a convolution operation, where a set of filters (network weights) are used to perform convolution across the whole image space, hence the name 'convolution layer'. This weight sharing feature is very useful for image analysis tasks because similar simple (line, corner, circle, etc) or complex (object shape, human face, tumors, etc) structures can occur in multiple locations in a single image. Pooling layers, which perform patch-wise down-sampling, are often used in convolutional network after convolution layers to decrease the image size. This is useful for reducing the network size and ensuring translational invariance for the following convolutions. For classification tasks, a set of fully-connected layers are usually added at the end of the convolutional network to predict class labels. However, this is not needed for pixel-to-pixel or voxel-to-voxel image regression networks for tasks like image synthesis and patch-wise image segmentation.

2.2.2 Convolutional Network's Classification and Regression Criteria

In most cases, deep learning tasks can be summarized as one of two tasks: classification, where a fixed integer from a set of integers is the final output, and regression, where a continuous number is the final output. A loss function is usually used to measure the similarity between the network output and the ground truth, and it is used to guide the network training process. There are various criteria that can be used as loss functions for classification and regression tasks, and I briefly discuss them here.

2.2.2.1 Classification

Cross Entropy. Cross entropy comes from information theory. It measures the similarity between two *probabilities*, and it is widely used as *the* loss function for classification problems in the deep learning community. This is because it is very natural to model the network output to be the probability of the input being a specific class by using a sigmoid layer at the end of the network for binary-label classification

$$f(x) = \frac{1}{1 + e^{-x}}$$

or a softmax layer for multi-label classification

$$f_i(x) = \frac{e^{x_i - \text{shift}}}{\sum_j e^{x_j - \text{shift}}}$$

where shift $= \max_i(\{x_i\})$. Generally speaking, for binary classification tasks using cross entropy as the criterion, the ground truth label is either 0 or 1, and the network output is a value between 0 and 1. A sigmoid layer is used as the final layer in the network to ensure the network output range is $[0, 1]$. For multi-label classification tasks, the ground truth label can be chosen from n values, and the network output is usually a 1D vector of length n. Usually a softmax layer is used as the network output layer to ensure that the sum of the network outputs (i.e., the probability that the input belongs to certain class) is 1.

Mathematically, for binary classification, suppose the network output is p, meaning the input has a probability of p to belong to class 1 and $1 - p$ to belong to class 0, then the binary cross entropy loss can be written as

$$E(p, t) = -t\log p - (1 - t)\log(1 - p) \ .$$

where t is the ground truth label and $t \in \{0, 1\}$. For multi-class classification, where the network output is a 1D vector of length n, the cross entropy can be written as

$$E(p,t) = -\sum_{i=1}^{n} f(i,t)\text{log}(p(i)), \quad f(i,t) = \begin{cases} 1, \text{ if } i = t \ ; \\ 0, \text{otherwise} \ . \end{cases}$$

Here, t is the ground truth label and $t \in \{1, 2, \cdots, n\}$.

Hinge Loss. Hinge loss is generally used for maximum-margin classification, i.e., the classification problem where the goal is to maximize the minimal distance of the training data (i.e., the margin) to the decision hyperplane. This is especially the case for support vector machines. However, there are several research papers that use hinge loss as the classification criterion for deep networks (Kamper et al., 2016; Tang, 2013) as well. When using hinge loss as the criterion, there are no specific non-linear layers such as sigmoid layers or softmax layers added to the end of the network. For binary classification, the network predicts a value p, and the hinge loss can be written as

$$E(p,t) = \max(0, \text{margin} - pt)$$

where margin is the decision boundary (i.e., value for the decision hyperplane), and t is the ground truth label and $t \in \{-1, 1\}$. For multi-class classification, the network output p is a 1D vector of length n, and suppose the correct label value is t_{val} and $t_{\text{val}} \in \{1, 2, \cdots, n\}$, then the hinge loss can be written as

$$E(p, t_{\text{val}}) = \sum_{i=1, i \neq t_{\text{val}}}^{n} \max(0, \text{margin} - p(t_{\text{val}}) + p(i)) \ .$$

2.2.2.2 Regression

Sum-of-square loss. This criterion is a classic choice and is widely used in regression tasks due to the ease of computing the gradient and the solution being stable, i.e., for a small perturbation of the data point, the regression parameters only change slightly. This means when using the sum of square loss, the regression parameters are continuous functions of the data. For example, given the network

output value $\mathbf{x}$ and the ground truth value $\mathbf{y}$, the mean square error is simply $E(x) = ||x - y||^2$, and the gradient with respect to x is $2(x - y)$.

l_1 loss. The l_1 loss is another popular choice for the regression task, and can be written as $E(x) = ||x - y||$. The gradient of this loss is

$$G(x) = \partial_x E(x) = \begin{cases} 1, & x > y \ , \\ -1, & x < y \ , \\ 0, & x = y \ , \end{cases}$$

where the gradient at $x = y$ is simply defined to be zero. Compared with sum of square loss, l_1 loss is not a stable criterion because l_1 loss is not continuous with respect to the data. However, the l_1 loss has several advantages: it is robust against outliers, and for image synthesis tasks it generates sharper edges compared with the sum of square loss, making it a favorable choice in some image synthesis networks such as in `pix2pix` (Isola et al., 2016).

2.2.3 Current Applications of Deep Network in Medical Image Analysis

With the fast research progress in the deep learning community, many areas in medical image analysis have seen the use of deep networks with outstanding performance. Below, I will mainly discuss the recent development of the following four major medical image analysis areas where deep learning has been applied in: image classification, detection, image segmentation and image registration.

2.2.3.1 Image Classification

Classfication in medical image analysis can primarily be separated into two categories: whole image classification and object classification. For whole image classification, the image (or set of images) is sent into the network and the network produces a single class label prediction as the output. This formulation is usually used for disease diagnosis. In these tasks the number of subjects in the training dataset is generally small, and transfer learning plays an important role in

deep learning based methods for these tasks. Transfer learning is, in this context, using networks pre-trained with natural images on medical images. There are two widely applied methods for using a pre-trained network: using the pre-trained network as a feature extractor on the medical image data to perform classification with other machine learning tools, and fine-tuning the pre-trained network on medical image data and to use the network to directly perform classification. There are papers (Antony et al., 2016; Kim and Hwang, 2016) comparing the performance between these two approaches, but no conclusion has been reached on which one is better. The network structures evolve from Stacked Autoencoders (Suk and Shen, 2013) and Deep Boltzmann Machines (Brosch and Tam, 2013; Plis et al., 2014) to convolutional neural networks (Menegola et al., 2016; Payan and Montana, 2015; Hosseini-Asl et al., 2016) or even complex networks with newly defined 'edge-to-edge', 'edge-to-node' and 'node-to-graph' convolution filters, where the convolution is based on topological locality instead of spatial locality, to mimic the connectivity pattern for the analysis of the human connectome (Kawahara et al., 2017).

Another application area is object classification, which classifies a part of the image, instead of the whole image, into different categories. Example tasks include lung nodule classification (Shen et al., 2015), skin-lesion classification (Kawahara et al., 2016) and nuclear cataracts classification (Gao et al., 2015). Multi-stream networks, where image patches of different downsampling scales are used as network inputs, are often used in these tasks. For the majority of the tasks, only 2D images are used as the network input either due to the nature of the problem (e.g., skin lesion image classification) or the additional complexity of using 3D images. However, 3D information from 3D medical images has been explored to achieve better classification accuracy, such as using multiple 2D patches with different planes of a cube as network input (Setio et al., 2016) and directly using 3D convolutional networks for classification from 3D image patches (Nie et al., 2016b).

2.2.3.2 Detection

There are two main areas for detection in medical image analysis: landmark/region localization and object/lesion detection. For landmark and region localization, the majority of the proposed methods solve the 3D image volume parsing problem by decomposing the 3D space into 3 2D

orthogonal planes. The localization task is done in each 2D plane and the final 3D location is found by summarizing from the orthogonal 2D planes, such as using the intersection point of the three orthogonal 2D planes with the highest classification likelihood. This is currently preferred for 3D medical image localization due to its simplicity compared to direct 3D volume computation. These approaches either change the localization problem to a set of 2D whole image classification problem where landmark/region locations are generated by combining the locations of the orthogonal 2D slices with the desired landmarks (Yang et al., 2015; de Vos et al., 2016), or to regression problems where the network output for each 2D slice is a probability heatmap (Payer et al., 2016), and the location with the highest probability in 3 dimensions are selected as the landmarks. However, there are also several approaches to directly detect landmark/region in the 3D image space using 3D convolutions on 3D patches with some success (Zheng et al., 2015; Ghesu et al., 2015).

For object/lesion detection, where the task is to detect a small region of the whole image (e.g., lesions) instead of single voxels (landmarks) or whole image volume segments (regions), the techniques used for object classification are often utilized here since these two tasks are very similar, such as using multi-stream networks to perform multi-resolution object detection (Roth et al., 2016).

2.2.3.3 Segmentation

Medical image segmentation via deep learning has seen a huge increase in popularity, and various network structures have been proposed. A prominent example is the U-net (Ronneberger et al., 2015) for 2D image segmentation. The network structure is similar to an autoencoder (Rumelhart et al., 1986), which can be separated into an encoder and a decoder. The encoder takes the network input and produces features, and the decoder takes the features generated from the encoder to reconstruct the original input. Usually the network structure of the decoder is the inverse of the encoder. However, in U-net the network input and output is different from an autoencoder: it takes a 2D image as the network input, and generates a label map as the output. The network has an equal number of pooling and unpooling layers and has skip connections between opposing convolution layers before/after the pooling/unpooling operations. Using the same number of pooling and unpooling layers means that the entire 2D image can be processed by U-net in one

pass, which is more efficient than patch-based networks that predict classification result for one patch at a time or even only for the center voxel of the whole patch. The skip connections in this case concatenate the input features of the pooling layers to the output of the unpooling layers using the same patch size, effectively skipping network layers between them. This operations enables U-net to predict more precise segmentation results. Several papers extend the U-net architecture by, e.g., extending the network to take 3D images as input using 3D convolutions (Milletari et al., 2016a), and adding ResNet-like short skip connections in addition to the existing connections in the U-net (Drozdzal et al., 2016), with promising results. Other deep learning techniques are also explored for segmentation, such as using recurrent networks to use neighboring information for patch-to-patch segmentation (Xie et al., 2016) and patch-to-center-voxel segmentation (Andermatt et al., 2016), as well as 3D convolution networks for patch-to-center-voxel segmentation (Korez et al., 2016).

2.2.3.4 Registration

Compared to the previously discussed areas, deep learning is a relatively new approach for medical image registration. However, it is being actively explored with many promising results. There are various works to use deep learning for medical image registration, and they can be separated into three approaches[3]. The first one is used for feature based image registration, where autoencoders are used to extract better image features (Wu et al., 2013). The second approach uses a deep network to learn the image similarity measure, and uses the network as the image matching part of the registration energy during optimization (Cheng and Zheng, 2015; Gutierrez-Becker et al., 2017). Finally, deep networks can be used to directly predict deformation parameters and this idea is applied to affine registration (Miao et al., 2016) and LDDMM (Yang et al., 2016b, 2017a).

[3]It is worth noting that for multi-modal image registration and pathological image registration tasks, there is another approach that uses a deep network for cross-modal or quasi-normal image synthesis. The synthesized image can then be used in traditional registration methods, such as in (Yang et al., 2016a).

CHAPTER 3: Uncertainty Quantification of LDDMM using a Low-rank Hessian Approximation

This chapter[1] develops a variational approximation method for quantifying uncertainty of the LDDMM registration model. Specifically, I develop a method to estimate uncertainty in the deformation parameters of the shooting formulation (Vialard et al., 2012b) of LDDMM. At the optimal solution, I assume a local multivariate Gaussian distribution to approximate the posterior distribution of the deformation parameters, and approximate the covariance matrix through the inverse of the approximated energy Hessian to quantify uncertainty of the registration parameters. Using the Hessian of the energy to estimate the covariance matrix of parameters has been discussed for large scale inverse problems in other application domains (Flath et al., 2011; Kalmikov and Heimbach, 2014). For high dimensional parameter spaces, computing the full Hessian is prohibitive due to large memory requirements. Therefore, I develop a method to compute Hessian-vector products for the LDDMM energy. This allows me to efficiently compute and store an approximation of the Hessian. In particular, I directly approximate the covariance matrix by exploiting the low-rank structure of the image mismatch Hessian. My framework therefore allows uncertainty analysis for LDDMM at a manageable computational cost.

Sec. 3.1 discusses the relationship of the covariance matrix of the parameter distribution and the Hessian of the energy function. Sec. 3.2 introduces my framework to compute the Hessian via tangent linear models. Sec 3.3 shows my strategy to approximate the covariance using a low-rank Hessian approximation. Sec. 3.4 shows experimental results for both synthetic and real data.

[1]The work presented in this chapter is based on the paper (Yang and Niethammer, 2015).

3.1 Covariance Matrix and the Hessian of the Registration

Consider a Gaussian random vector θ of dimension N_θ with mean value θ^* and covariance matrix Σ_θ. Its joint probability density function can be written as

$$P(\theta) = (2\pi)^{-\frac{N_\theta}{2}} |\Sigma_\theta|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\theta-\theta^*)^T \Sigma_\theta^{-1}(\theta-\theta^*)\right] \ . \tag{3.1}$$

In image registration, one typically minimizes the energy given by the negative log-likelihood of the posterior distribution. As the negative log-likelihood of the multivariate Gaussian is given by

$$E(\theta) = -\ln(P(\theta)) = \frac{N_\theta}{2}\ln 2\pi + \frac{1}{2}\ln|\Sigma_\theta| + \frac{1}{2}(\theta-\theta^*)^T \Sigma_\theta^{-1}(\theta-\theta^*) \ , \tag{3.2}$$

computing the Hessian of $E(\theta)$ with respect to θ results in $H_{E(\theta)} = \Sigma_\theta^{-1}$ and directly relates the covariance matrix of the multivariate Gaussian model to the Hessian of the energy through its inverse.

In my method, I assume that, at optimality, the LDDMM energy can be locally approximated by a second order function and hence by a multivariate Gaussian distribution. In statistics this approach is called the Laplace approximation (Azevedo-filho, 1994). In particular, I make use of the shooting based formulation of LDDMM (Vialard et al., 2012b) which parameterizes the spatial deformation by an initial momentum (or, equivalently, by an initial velocity field) and associated evolution equations describing the space deformation over time. Specifically, the registration parameter in shooting based LDDMM is the initial momentum m, which is the dual of the initial velocity v, an element in a reproducing kernel Hilbert space V. The initial momentum belongs to V's dual space V^*, and it is connected with v via a positive-definite, self-adjoint differential operator $L : V \to V^*$ such that $m = Lv$ and $v = Km$. Here the operator K denotes the inverse of L. The energy of LDDMM with the dynamic constraints of the shooting equations (Vialard et al., 2012b) can then be written as

$$E(m_0) = \langle m_0, Km_0\rangle + \frac{1}{\sigma^2}||I(1) - I_1||^2 \ , \tag{3.3}$$

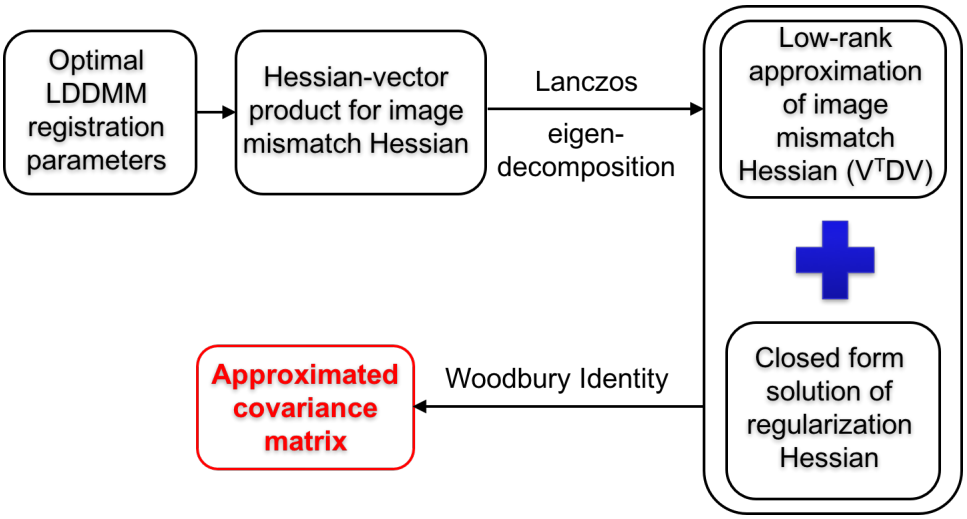


Figure 3.1: Proposed framework for covariance estimation for LDDMM by low-rank Hessian approximation.

$$m_t + \mathrm{ad}_v^* m = 0,\ m(0) = m_0, \qquad I_t + \nabla I^T v = 0,\ I(0) = I_0 \qquad m - Lv = 0\ , \tag{3.4}$$

where the operator ad^* is the dual of the negative Jacobi-Lie bracket of vector fields: $\mathrm{ad}_v w = -[v, w] = Dvw - Dwv$, and I_0 and I_1 indicate the source and the target images for registration respectively. Formally, the Hessian of this energy is given by

$$H_{m_0} = 2K + \frac{\partial^2 \frac{1}{\sigma^2} ||I(1) - I_1||^2}{\partial m_0^2}\ . \tag{3.5}$$

Computing this Hessian is not straightforward, because $I(1)$ only indirectly depends on m_0 through the dynamic constraints and m_0 can become very high-dimensional, making computation and storage challenging. Figure 3.1 shows the framework of my proposed solution, which solves the problem above by using a low-rank Hessian approximation. I discuss the detailed strategy for computing the Hessian in Sec. 3.2.

3.2 Tangent Linear Models for Hessian-vector Product Computation

To avoid computation of the full Hessian, I instead compute Hessian-vector products. This enables me to make use of efficient iterative methods such as the Lanczos method (Lanczos, 1950)

to perform eigen-decomposition of the Hessian, which I exploit to compute an approximation of the Hessian and the covariance matrix.

The equivalent of Hessian-vector products for LDDMM can be computed using the second variation of the LDDMM energy. Specifically, the second variation in the direction δm_0 can be written as

$$\delta^2 E(m_0; \delta m_0) := \frac{\partial^2}{\partial \epsilon^2} E(m_0 + \epsilon \delta m_0)|_{\epsilon=0} = \langle \delta m_0, \nabla^2 E \delta m_0 \rangle. \tag{3.6}$$

Here $\nabla^2 E$ denotes the Hessian of $E(m_0)$. Using this formulation, I can read off the Hessian-vector product $\nabla^2 E \delta m_0$ from the second variation. Computing this second variation of the LDDMM shooting energy can be accomplished by linearizing both the forward equations for shooting as well as the associated adjoint equations around the optimal solution (the solution of the registration problem). The resulting linearized forward and adjoint equations are called tangent linear model (TLM) and tangent linear adjoint model (TLAM). Below I give the detailed derivation of the TLM and TLAM for LDDMM shooting.

Remember that I want to compute the Hessian-vector product for the initial momentum version of the shooting formulation of LDDMM

$$E(m_0) = \langle m_0, K m_0 \rangle + \frac{1}{\sigma^2} ||I(1) - I_1||^2 \ , \tag{3.7}$$

with dynamic constraints

$$\text{momentum evolution: } m_t + \text{ad}_v^* m = 0, \ m(0) = m_0 \ , \tag{3.8}$$

$$\text{image evolution: } \quad I_t + \nabla I^T v = 0, \ I(0) = I_0 \ , \tag{3.9}$$

$$\text{momentum-velocity transformation: } \quad m - Lv = 0 \ . \tag{3.10}$$

I add the dynamic constraints into the energy using time-dependent adjoint variables $\hat{m}$, $\hat{I}$ and $\hat{v}$ through time and space $[0,1]^d$. This gives me the Lagrangian

$$\mathcal{L}(m_0, I, v, \hat{m}, \hat{I}, \hat{v}) = E(m_0) + \int_0^1 \langle \hat{m}, m_t + \text{ad}_v^* m \rangle + \langle \hat{I}, I_t + \nabla I^T v \rangle + \langle \hat{v}, m - Lv \rangle dt \ . \quad (3.11)$$

Computing the second variation for the Lagrangian requires computing

$$\begin{aligned}
\sigma^2 \mathcal{L} =& \frac{\partial^2}{\partial \epsilon^2} \mathcal{L}(m_0 + \epsilon\delta m_0, I + \epsilon\delta I, v + \epsilon\delta v, \hat{m} + \epsilon\delta\hat{m}, \hat{I} + \epsilon\delta\hat{I}, \hat{v} + \epsilon\delta\hat{v})|_{\epsilon\to 0} \qquad (3.12)\\
=& \frac{\partial^2}{\partial \epsilon^2} \Big(\langle m_0 + \epsilon\delta m_0, K(m_0 + \epsilon\delta m_0) \rangle + \frac{1}{\sigma^2} ||I(1) + \epsilon\delta I(1) - I_1||^2 + \\
& \int_0^1 \langle \hat{m} + \epsilon\delta\hat{m}, m_t + \epsilon\delta m_t + \text{ad}^*_{v+\epsilon\delta v}(m + \epsilon\delta m) \rangle + \\
& \langle \hat{I} + \epsilon\delta\hat{I}, I_t + \epsilon\delta I_t + \nabla(I^T + \epsilon\delta I^T)(v + \epsilon\delta v) \rangle + \\
& \langle \hat{v} + \epsilon\delta\hat{v}, m + \epsilon\delta m - L(v + \epsilon\delta v) \rangle dt \Big)|_{\epsilon\to 0} \ .
\end{aligned}$$

Computing Eqn. 3.12 results in

$$\begin{aligned}
\sigma^2 \mathcal{L} =& \underbrace{\langle \delta m_0, 2K\delta m_0 \rangle}_{1} + \underbrace{\frac{2}{\sigma^2} \langle \delta I(1), \delta I(1) \rangle}_{2} + \qquad (3.13)\\
2 \int_0^1 \Big(& \underbrace{\langle \delta\hat{m}, \delta m_t + ad_v^* \delta m + ad_{\delta v}^* m \rangle}_{3} + \underbrace{\langle \hat{m}, ad_{\delta v}^* \delta m \rangle}_{4} + \\
& \underbrace{\langle \delta\hat{I}, \delta I_t + \nabla I^T \delta v + \nabla \delta I^T v \rangle}_{5} + \underbrace{\langle \hat{I}, \nabla \delta I^T \delta v \rangle}_{6} + \\
& \underbrace{\langle \delta\hat{v}, \delta m - L\delta v \rangle}_{7} \Big) dt \ .
\end{aligned}$$

Here, I label every part of the equation to help follow the computations. For the transformations in the equations below, all parts are labeled according to the number of the original component in Eqn. 3.13. I assume periodic boundary conditions for the momentum and the velocity. Furthermore, I assume

that $\delta I(0) = 0$, i.e., the initial image is fixed. Thus I can perform the following transformations

$$3 : \int_0^1 \langle \delta\hat{m}, \delta m_t \rangle \, dt = \int_0^1 \langle -\delta\hat{m}_t, \delta m \rangle \, dt + \langle \delta\hat{m}(1), \delta m(1) \rangle - \langle \delta\hat{m}(0), \delta m(0) \rangle \ ,$$

$$3 : \langle \delta\hat{m}, ad_v^* \delta m \rangle = \langle ad_v \delta\hat{m}, \delta m \rangle \ ,$$

$$3 : \langle \delta\hat{m}, ad_{\delta v}^* m \rangle = \langle ad_{\delta v} \delta\hat{m}, m \rangle = \langle -ad_{\delta\hat{m}} \delta v, m \rangle = \langle \delta v, -ad_{\delta\hat{m}}^* m \rangle \ ,$$

$$4 : \langle \hat{m}, ad_{\delta v}^* \delta m \rangle = \langle ad_{\delta v} \hat{m}, \delta m \rangle = \langle -ad_{\hat{m}} \delta v, \delta m \rangle = \langle \delta v, -ad_{\hat{m}}^* \delta m \rangle \ ,$$

$$5 : \int_0^1 \langle \delta\hat{I}, \delta I_t \rangle \, dt = \int_0^1 \langle -\delta\hat{I}_t, \delta I \rangle \, dt + \langle \delta\hat{I}(1), \delta I(1) \rangle \ ,$$

$$5 : \langle \delta\hat{I}, \nabla I^T \delta v \rangle = \langle \nabla I \delta\hat{I}, \delta v \rangle \ ,$$

$$5 : \langle \delta\hat{I}, \nabla \delta I^T v \rangle = \langle \delta I, -div(v\delta\hat{I}) \rangle \ ,$$

$$6 : \langle \hat{I}, \nabla \delta I^T \delta v \rangle = \langle \nabla \delta I \hat{I}, \delta v \rangle = \langle \delta I, -div(\delta v \hat{I}) \rangle \ ,$$

$$7 : \langle \delta\hat{v}, L\delta v \rangle = \langle L\delta\hat{v}, \delta v \rangle \ .$$

Putting these transformations into Eqn. 3.13, I get

$$\begin{aligned}
\sigma^2 \mathcal{L} = & \underbrace{\langle \delta m_0, 2K\delta m_0 \rangle}_{1} + \underbrace{\frac{2}{\sigma^2} \langle \delta I(1), \delta I(1) \rangle}_{2} - \underbrace{\langle \delta\hat{m}(0), \delta m(0) \rangle}_{3} + \underbrace{\langle \delta\hat{I}(1), \delta I(1) \rangle}_{5} + \\
& \int_0^1 \Big(\underbrace{\langle \delta\hat{m}, \delta m_t + ad_v^* \delta m + ad_{\delta v}^* m \rangle + \langle \delta m, -\delta\hat{m}_t \rangle + \langle \delta m, ad_v \delta\hat{m} \rangle + \langle \delta v, -ad_{\delta\hat{m}}^* m \rangle}_{3} + \\
& \underbrace{\langle \delta m, ad_{\delta v} \hat{m} \rangle + \langle \delta v, -ad_{\hat{m}}^* \delta m \rangle}_{4} + \\
& \underbrace{\langle \delta\hat{I}, \delta I_t + \nabla I^T \delta v + \nabla \delta I^T v \rangle + \langle \delta I, -\delta\hat{I}_t \rangle + \langle \delta v, \nabla I \delta\hat{I} \rangle + \langle \delta I, -div(v\delta\hat{I}) \rangle}_{5} + \\
& \underbrace{\langle \delta v, \nabla \delta I \hat{I} \rangle + \langle \delta I, -div(\delta v \hat{I}) \rangle}_{6} + \underbrace{\langle \delta\hat{v}, \delta m - L\delta v \rangle + \langle \delta m, \delta\hat{v} \rangle + \langle \delta v, L\delta\hat{v} \rangle}_{7} \Big) dt \ .
\end{aligned}$$

Combining these terms, I get

$$\begin{aligned}\sigma^2\mathcal{L} =&\langle \delta m_0, 2K\delta m_0 - \delta\hat{m}(0)\rangle + \langle \delta I(1), \frac{2}{\sigma^2}\delta I(1) + \delta\hat{I}(1)\rangle + \\ &\langle \delta\hat{m}(1), \delta m(1)\rangle + \\ \int_0^1 \Big(&\langle \delta\hat{m}, \delta m_t + ad_v^*\delta m + ad_{\delta v}^* m\rangle + \\ &\langle \delta\hat{I}, \delta I_t + \nabla I^T\delta v + \nabla\delta I^T v\rangle + \\ &\langle \delta\hat{v}, \delta m - L\delta v\rangle + \\ &\langle \delta v, -ad_{\delta\hat{m}}^* m - ad_{\hat{m}}^*\delta m + \nabla I\delta\hat{I} + \nabla\delta I\hat{I} - L\delta\hat{v}\rangle + \\ &\langle \delta I, -\delta\hat{I}_t - div(\delta v\hat{I} + v\delta\hat{I})\rangle + \\ &\langle \delta m, -\delta\hat{m}_t + ad_{\delta v}\hat{m} + ad_v\delta\hat{m} + \delta\hat{v} = 0\rangle\Big) dt \ .\end{aligned} \tag{3.14}$$

Regarding everything inside the integration as optimality conditions, and extracting the boundary condition as $\delta m(0) = \delta m_0, \quad \delta I(0) = 0, \quad \delta\hat{m}(1) = 0, \quad \delta\hat{I}(1) = -\frac{2}{\sigma^2}\delta I(1)$, I finally obtain the equation for computing Hessian-vector products

$$\nabla^2 E\delta m_0 = 2K\delta m_0 - \delta\hat{m}(0) \tag{3.15}$$

together with the tangent linear model (TLM)

$$\begin{cases} \delta m_t + ad_{\delta v}^* m + ad_v^*\delta m = 0, \quad \delta m(0) = \delta m_0 \\ \delta I_t + \nabla\delta I^T v + \nabla I^T\delta v = 0, \quad \delta I(0) = 0 \\ \delta m - L\delta v = 0 \end{cases} \tag{3.16}$$

and the tangent linear adjoint model (TLAM)

$$\begin{cases} -\delta\hat{m}_t + ad_{\delta v}\hat{m} + ad_v\delta\hat{m} + \delta\hat{v} = 0, \quad \delta\hat{m}(1) = 0 \\ -\delta\hat{I}_t - div(\delta v\hat{I} + v\delta\hat{I}) = 0, \quad \delta\hat{I}(1) = -\frac{2}{\sigma^2}\delta I(1) \\ -ad^*_{\delta\hat{m}}m - ad^*_{\hat{m}}\delta m + \nabla I\delta\hat{I} + \nabla\delta I\hat{I} - L\delta\hat{v} = 0 \ . \end{cases} \tag{3.17}$$

Note that the TLM and TLAM are in fact the linearized versions of the forward equations:

$$\begin{cases} m_t + ad^*_v m = 0, \quad m(0) = m_0, \\ I_t + \nabla I^T v = 0, \quad I(0) = I_0, \\ m - Lv = 0 \ , \end{cases} \tag{3.18}$$

and the adjoint equations:

$$\begin{cases} -\hat{m}_t + ad_v\hat{m} + \hat{v} = 0, \quad \hat{m}(1) = 0, \\ -\hat{I}_t - \text{div}(v\hat{I}) = 0, \quad \hat{I}(1) = -\frac{2}{\sigma^2}(I(1) - I_1), \\ -ad^*_{\hat{m}}m + \nabla I\hat{I} - L\hat{v} = 0 \end{cases} \tag{3.19}$$

for LDDMM shooting. Hence, a more direct derivation could simply be obtained by linearizing the forward model and the adjoint model instead of computing the second variation.

Solving these equations for a given initial condition δm_0 then allows the computation of the Hessian-vector product (in a functional sense) as Eqn. 3.15. Here, $2K\delta m_0$ can be computed directly and $\delta\hat{m}(0)$ is the perturbation of the adjoint of the momentum propagation constraint at $t = 0$, which is obtained efficiently through a forward-backward sweep through the TLM (Eqn. 3.16) and TLAM (Eqn. 3.17).

3.3 Covariance Estimation with Low-rank Hessian Approximation

To estimate the covariance, a straightforward way is inverting the Hessian of the full energy. This is not feasible for standard LDDMM because the number of parameters is so large that saving or computing the inverse of the full Hessian is prohibitive[2]. Another possibility is to approximate the full energy Hessian. Note that the Hessian of the LDDMM energy can be separated into the Hessian of the regularization energy and the Hessian of the image mismatch energy. Thus, I can separately calculate Hessian-vector products for these two parts based on Eqn. 3.15 as:

$$H_m^{\text{regularization}}\delta m_0 = 2K\delta m_0 \ , \quad H_m^{\text{mismatch}}\delta m_0 = -\delta\hat{m}(0) \ .$$

A simple low-rank pseudoinverse as an approximation of the Hessian would result in approximation

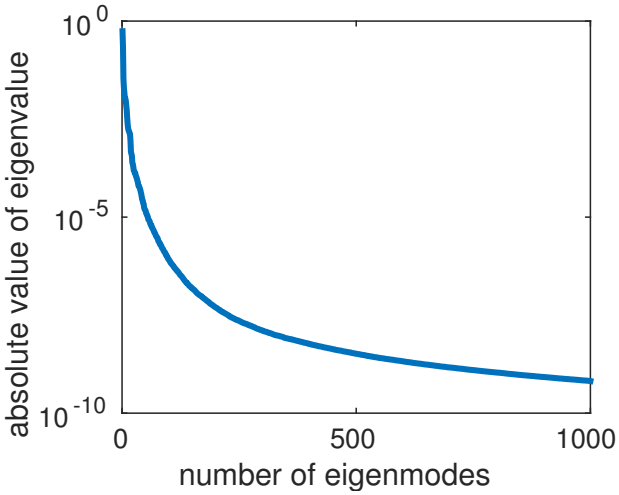


Figure 3.2: First 1000 largest absolute eigenvalues of image mismatch Hessian for a 100×140 pixels heart registration case using initial momentum LDDMM.

errors for both $H_m^{\text{regularization}}$ *and* the H_m^{mismatch}. This can be partially avoided by using the *exact* inverse of the Hessian of the regularization combined with an *approximation* for the image mismatch Hessian. To compute the covariance matrix, I realize that for many ill-posed inverse problems, the spectrum of the absolute values of the eigenvalues of the image mismatch Hessian decays rapidly to

[2]Note that this would be possible when using a landmark-based LDDMM variant where the parameterization of the deformation becomes finite-dimensional.

zero. Fig. 3.2 shows an example of the largest 1000 absolute values of the eigenvalues for a 2D 100×140 pixels heart registration case for initial momentum LDDMM. By computing only a few[3] dominant eigenmodes (using an iterative eigensolver such as the Lanczos method) of the image mismatch Hessian with respect to the initial momentum, I can accurately approximate the Hessian with much less memory and computational effort. Suppose I approximate the image mismatch Hessian with k dominating eigenmodes as

$$H_{m(k)}^{\text{mismatch}} \approx V_{m(k)}^T D_{m(k)} V_{m(k)} \quad .$$

Here $D_{m(k)}$ is a $k \times k$ diagonal matrix, where the diagonal elements are the eigenvalues; $V_{m(k)}$ is a $k \times n$ matrix, where n is the number of all parameters, and each row of V_k is an eigenvector. For simplicity I write $H_m^{\text{regularization}}$ as H_m^{reg}. I can then approximate the covariance matrix Σ_m as

$$\Sigma_m = (H_m^{\text{reg}} + H_m^{\text{mismatch}})^{-1} \approx (H_m^{\text{reg}} + V_{m(k)}^T D_{m(k)} V_{m(k)})^{-1} \quad . \tag{3.20}$$

Since I have a closed-form solution for the inverse of H_m^{reg}, I can directly apply the Woodbury identity to the right hand side of equation 3.20 and obtain

$$\Sigma_m \approx (H_m^{\text{reg}})^{-1} - (H_m^{\text{reg}})^{-1} V_{m(k)}^T (D_{m(k)}^{-1} + V_{m(k)} (H_m^{\text{reg}})^{-1} V_{m(k)}^T)^{-1} V_{m(k)} (H_m^{\text{reg}})^{-1} \quad . \tag{3.21}$$

The advantage of this formulation is that $D_{m(k)}^{-1} + V_{m(k)} (H_m^{\text{reg}})^{-1} V_{m(k)}^T$ is a small $k \times k$ matrix, and its inverse can be computed easily using dense algebra.

In LDDMM I could quantify the uncertainty through the covariance with respect to either the initial momentum or its corresponding initial velocity. If I use the initial momentum, the approximation of $(D_{m(k)}^{-1} + V_{m(k)} (H_m^{\text{reg}})^{-1} V_{m(k)}^T)^{-1}$ will be accurate for small k, because the image mismatch Hessian for the initial momentum has a rapidly decreasing eigen-spectrum. However,

[3] In my experiments, I use 200 eigenmodes (3.8% of the total eigenmodes) for the 2D synthetic test case, 500 eigenmodes (1.8% of the total eigenmodes) for the 2D heart image test case, and 1000 eigenmodes (0.08% of the total eigenmodes) for the 3D test case.

the inverse of the regularization kernel would be $(H_m^{\text{reg}})^{-1} = (2K)^{-1} = \frac{1}{2}L$, which is a non-smooth kernel. In experiments, using the non-smooth kernel significantly increases the difference between approximations for different k's, especially when k is small. This means that the low-rank approximation of the Hessian is no longer stable across different k's, and it is not reliable to estimate the covariance matrix. On the other hand, in the initial velocity formulation, $(H_m^{\text{reg}})^{-1}$ is a smoothing kernel, and it further decreases the approximation difference for different k's. This is beneficial for our low-rank covariance approximation. Unfortunately, the image mismatch Hessian with respect to the initial velocity does not have a fast-decreasing spectrum due to the implicit smoothness of the initial velocity.

I want to use the rapidly decreasing eigen-spectrum of the momentum-based formulation, but at the same time I want to avoid its rough kernel when calculating the covariance matrix. My solution is to use the low-rank approximation of the initial *momentum* image mismatch Hessian to approximate the covariance with respect to the initial *velocity*. Recall the relation between momentum and velocity: $v = Km$. This means the Jacobian of v with respect to m is $J_{\frac{v}{m}} = K$. Thus by change of variables, I can obtain my final approximation of the covariance with respect to the initial velocity as

$$\Sigma_v = J_{\frac{v}{m}} \Sigma_m J_{\frac{v}{m}}^T \approx \frac{K}{2} - \frac{1}{4} V_{m(k)}^T (D_{m(k)}^{-1} + V_{m(k)} \frac{L}{2} V_{m(k)}^T)^{-1} V_{m(k)} \quad . \tag{3.22}$$

Eqn. 3.22 has the advantage of both using $(D_{m(k)}^{-1} + V_{m(k)}(H_m^{\text{reg}})^{-1} V_{m(k)}^T)^{-1}$ as the inverse part, which already gives accurate approximations, and avoiding multiplication of the inverse part with the non-smooth kernel L, making the overall covariance estimation accurate.

3.4 Experiments

3.4.1 Experimental Settings

I evaluate my proposed model using synthetic and real data. In the following experiments, L corresponds to the invertible and self-adjoint Sobolev operator, $L = a\Delta^2 + b\Delta + c$, with $a = 9 \times 10^{-4}$, $b = -6 \times 10^{-2}$, and $c = 1$; $\sigma = 0.1$. For the eigen-decomposition I use PROPACK (Larsen, 1998),

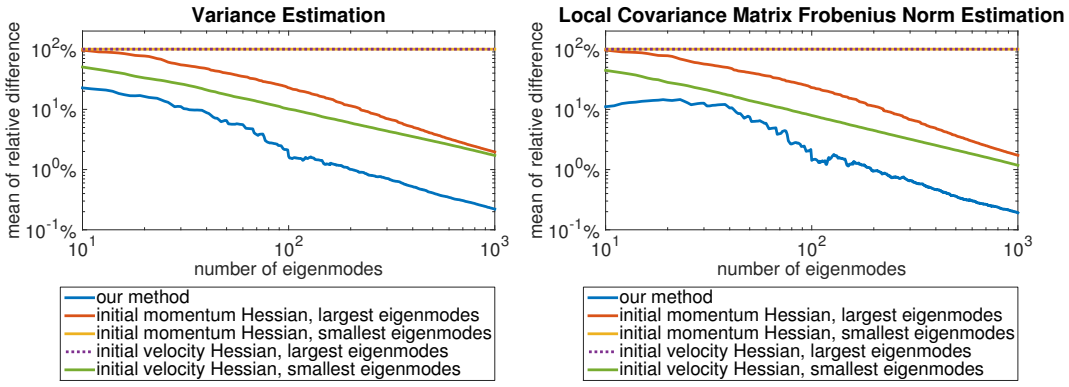


Figure 3.3: Relative low-rank approximation differences of the covariance matrix for the initial velocity for synthetic data. Initial momentum/velocity Hessian: Use initial momentum Hessian (via change of variables) or initial velocity Hessian to approximate. Largest/smallest eigenmodes: use the largest or the smallest eigenmodes to perform pseudo-inversion.

which implements a Lanczos bidiagonalization method. Computing 200 dominant eigenvectors for the 2D square synthetic example in Sec. 3.4.2, which gives a very accurate Hessian estimation, requires less than 3 min in MATLAB; however computing the full Hessian requires more than 30 min. Hence, my method is an order of magnitude faster. All images below are rescaled to a $[0, 1]$ range.

3.4.2 Synthetic example

The synthetic example is a simple registration of an expanding square. Figure 3.4 shows the source image, the target image, and the final registration result. The size of the image is 51×51 pixels, thus the size of the Hessian is 5202×5202. Since this Hessian is small, I compute the full covariance matrix using finite differences as the ground truth for my covariance estimation. I compare my method with the low rank pseudoinverse using both the initial momentum Hessian and the initial velocity Hessian.

I approximate two uncertainty measures: the variance of each parameter, and the spatially localized covariance matrix for each image pixel[4]. I compare the proposed approximation method with direct pseudo-inversion using both the initial velocity Hessian and the initial momentum

[4]In 2D this amounts to computing $\begin{pmatrix} \sigma^2_{xx}(i,j) & \sigma^2_{xy}(i,j) \\ \sigma^2_{xy}(i,j) & \sigma^2_{yy}(i,j) \end{pmatrix}$ for each pixel location (i, j), i.e., not considering non-local cross-variances.

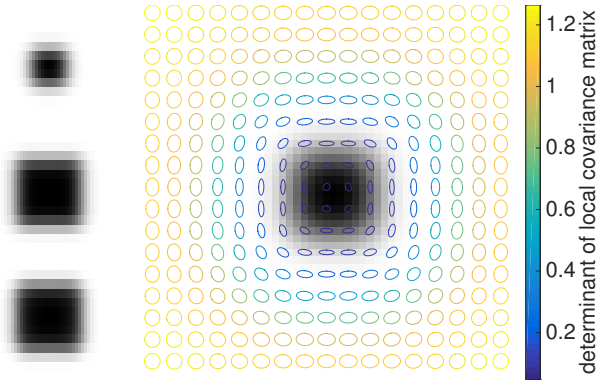


Figure 3.4: Square registration case. Left: top to bottom: source image, target image, warped result. Right: Uncertainty visualization by ellipses, mapped on source image.

Hessian combined with a change of variables. For the pseudo-inversion, both using the smallest eigenmodes and using the largest eigenmodes are attempted. Fig. 3.3 shows the mean of relative differences

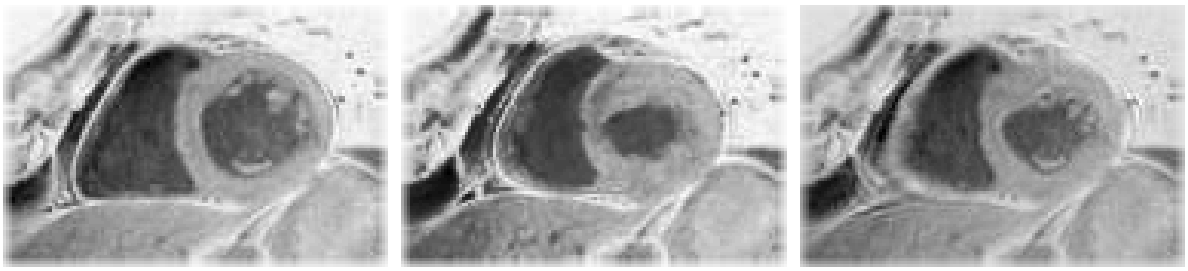
$$\text{mean}\left(\frac{|v_{\text{approx}}(i) - v_{\text{gt}}(i)|}{v_{\text{gt}}(i)}\right)_{i=1\cdots N}$$

with respect to the ground truth for different methods, where i indicates the ith value among all N values, v_{approx} is the approximated result and v_{gt} is the ground truth result. My method outperforms the other methods even with very few eigenmodes selected. In Fig 3.3, while 100 eigenmodes result in a relative error of around 1% for my method, around 1000 are required for the other approaches to achieve similar accuracy.

To visualize the uncertainty information, I extract the local covariance matrices from the approximated covariance matrix and visualize these matrices as ellipses on the source image. Fig. 3.4 shows the uncertainty visualization for the synthetic data. The ellipses are estimated using 200 dominant eigenmodes from the image mismatch Hessian. The color indicates the determinant of local covariance matrices. The closer to the center of the square, the smaller the determinant is, indicating a more confident registration result closer to the center. Furthermore, the uncertainty along the edge is larger than the uncertainty perpendicular to the edge, which is consistent with the aperture problem of image registration.

3.4.3 Real Image Example

2D heart image. I use cardiac data from the Sunnybrook cardiac MR database (Radau et al., 2009). The image corresponds to a beating heart of a 63 years old normal individual at two different time points. I cropped the axial images to a common 2D rectangular region around the heart. Fig. 3.5 shows the heart image and registration result. The size of the heart image



(a) Source image (b) Target image (c) Warped result

Figure 3.5: Heart registration test case.

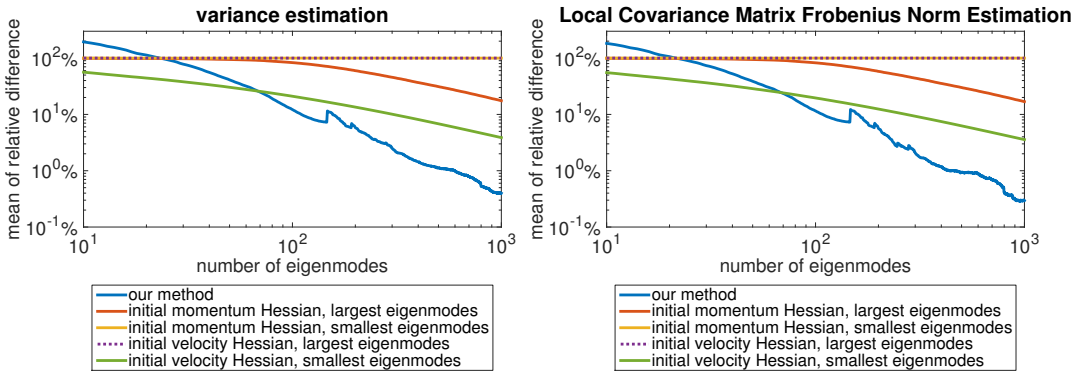


Figure 3.6: Relative low-rank approximation differences of the covariance matrix for the initial velocity for heart data.

is 100×140 pixels, resulting in a 28000×28000 Hessian. Fig. 3.6 shows the relative difference of both variance estimation and local covariance matrix Frobenius norm estimation for the 2D heart case. Relative to other methods, my method has higher relative difference when using the first 70 eigenmodes. However, the smallest mean relative difference using 70 eigenmodes is as large as 24.9% for variance estimation, and 23.5% for local covariance estimation.

Thus, one cannot get a reliable estimation using such a small number of eigenmodes. When the number of eigenmodes selected is larger than 70, my method always achieves better accuracy. Furthermore, my method can achieve a reasonably high accuracy much faster than other low-rank methods. For example, while the pseudoinverse of the initial velocity Hessian needs 1000 eigenmodes to achieve a 3.87% mean relative variance difference, my method only needs 237 eigenmodes to get the same accuracy.

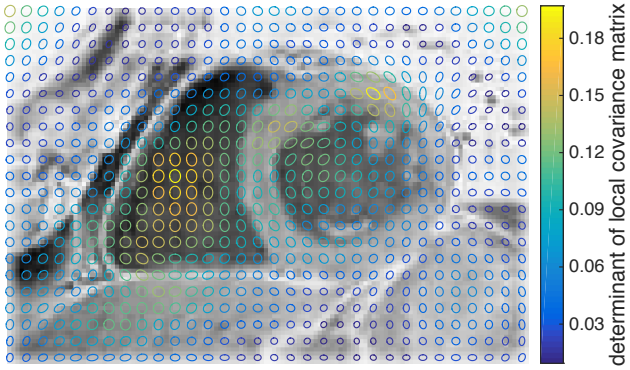


Figure 3.7: Uncertainty visualization of heart test case mapped on source image.

I select the top 500 eigenmodes for approximation and achieve a mean relative difference for variance of 1.13%, and for the Frobenius norm of the local covariance matrix of 0.94%. Using the pseudoinverse of the full Hessian gives a mean relative difference of 6.81% and 6.27% for the initial velocity Hessian, and 31.17% and 30.04% for the initial momentum Hessian. Fig. 3.7 shows the uncertainty visualization for the initial velocity on the source image. From the image, I see that the area inside the ventricle has high uncertainty, indicating low deformation confidence in the isotropic area. Also, there exists high uncertainty at the upper right edge of the atrium. This indicates high uncertainty for shifting along the edge.

3D brain image. Here the data consists of two MR images ($75 \times 90 \times 60$ voxels) of a macaque monkey from the UNC-Wisconsin Rhesus Macaque Neurodevelopment Database (Young et al., 2017) at 6 months and 12 months of age. The size of this Hessian is $1,215,000 \times 1,215,000$. This means that the full Hessian requires 10 Terabytes of memory to store, making storing the Hessian and calculating its inverse infeasible. Thus no approximation accuracy evaluation is done in this case. I calculate the largest 1000 eigenmodes for covariance approximation. For visualization of uncertainty, I use the trace of the local covariance matrix. Fig. 3.8 shows the 3D test case as well as the uncertainty visualization. I can see that, although the deformation is very small due to the small

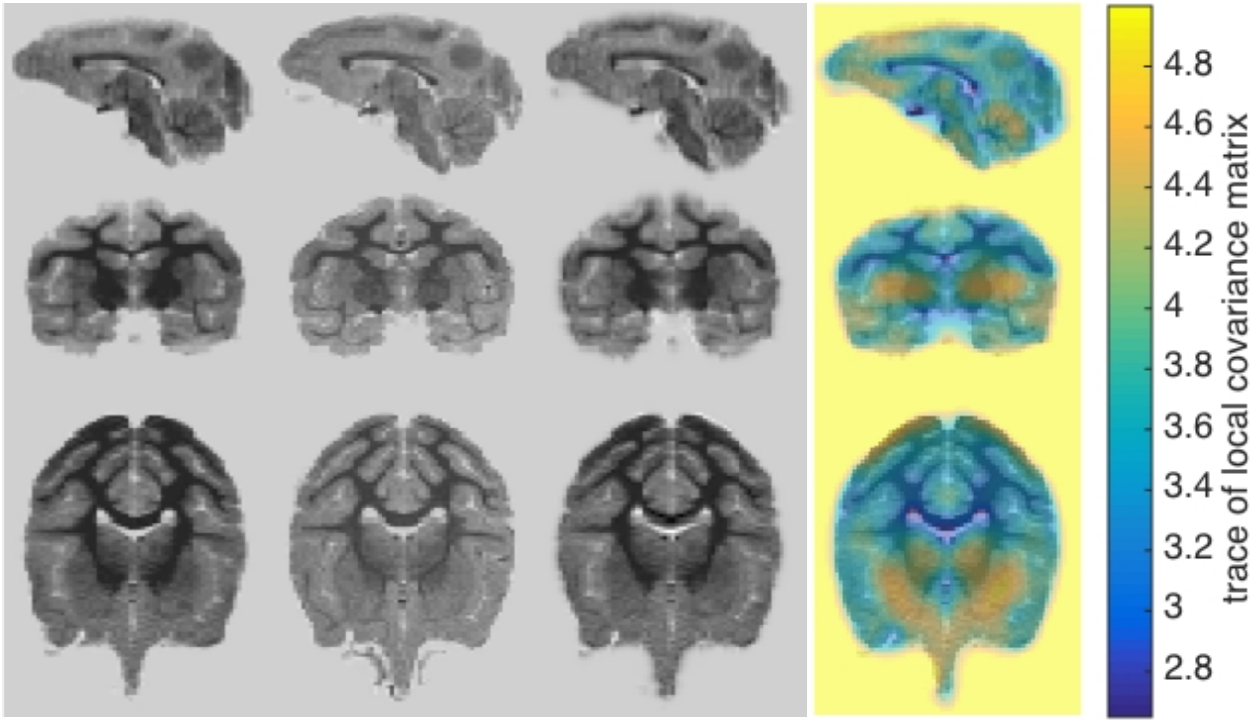


Figure 3.8: 3D monkey brain test case. Left to right: source image, target image, warped result, visualization of trace of local covariance matrix on the source image.

overall intensity change, my method can still capture isotropic areas that have very small changes.

These areas have a higher uncertainty compared to others.

CHAPTER 4: Improving Pathological Image Registration with a Variational Denoising Autoencoder

This chapter[1] presents my variational denoising autoencoder framework for image synthesis from pathological images to quasi-normal images to simplify the pathological image registration problem and to improve registration accuracy. In this chapter the following contributions are made: *First*, similar to (Liu et al., 2015), the proposed method directly maps a pathology image to a synthesized *quasi-normal* image, but no registration is needed in this process. *Second*, a deep variational autoencoder network is constructed to learn this mapping and it is trained using stochastic gradient variational Bayes (Kingma and Welling, 2013). *Third*, since the normal appearance of pathological tissue is unknown per se, I propose loss-function masking and pathology-like "structured noise" to train the autoencoder model. These strategies ignore mappings between image regions without known correspondence, and artificially create areas with known correspondence which can be used for training, respectively. *Fourth*, based on the variational formulation, estimation of the reconstruction uncertainty of the predicted quasi-normal image are made and used to adjust/improve the image similarity measure so that it focuses more on matching areas of low uncertainty.

The rest of this chapter is organized as follows: Sec. 4.1 introduces the basic theory from Denoising Variational Autoencoding, and discuss its relationship with the pathological-to-quasi-normal image synthesis task. Sec. 4.2 shows the proposed image synthesis network, and discuss strategies for training the network. Sec 4.3 discusses the uncertainty-weighted registration method, and Sec. 4.4 shows experiments using both synthetic and real image data.

4.1 Denoising Variational Autoencoder

I start with an introduction to the autoencoder family, starting from the basic traditional autoencoder to the denoising autoencoder and the variational autoencoder, which are two extensions

[1]The work presented in this chapter is based on the previous paper (Yang et al., 2016a).

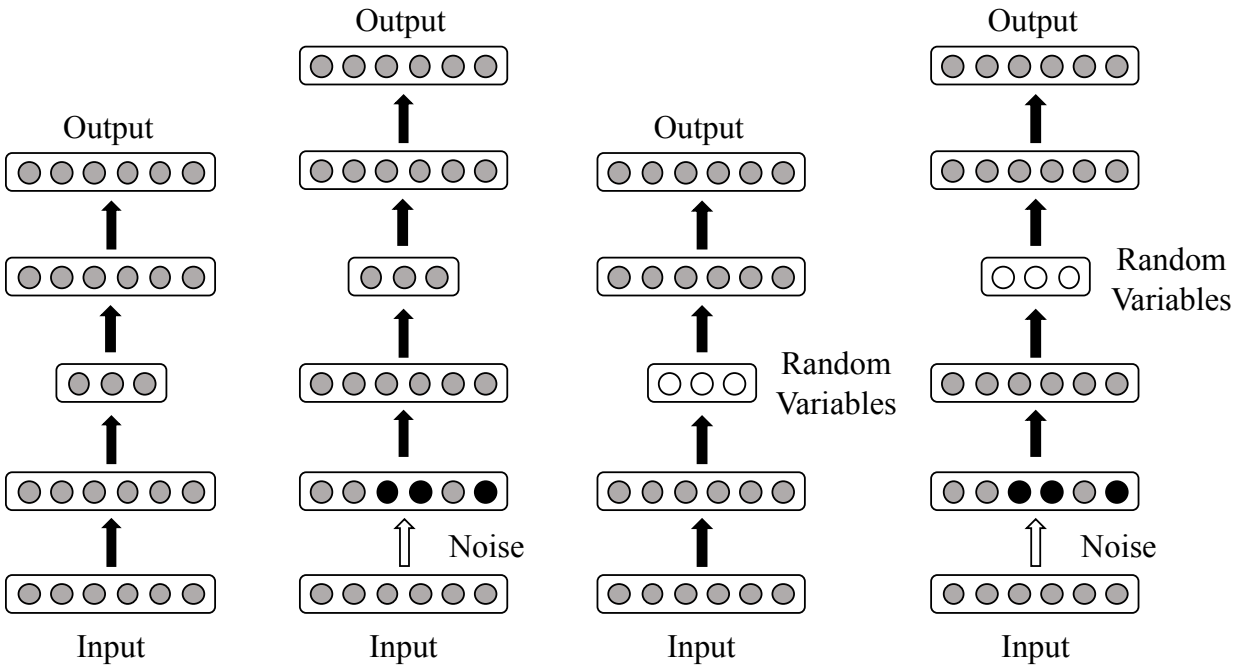


Figure 4.1: Autoencoder structures. Rounded rectangles means layers, and circles means layer values. (a): traditional autoencoder; (b): denoising autoencoder; (c): variational autoencoder; (d): denoising variational autoencoder. All network structures aim to reconstruct the input as the output. Denoising autoencoders add random noise to the input sending it to the network. For variational autoencoders, the hidden layer (layer with white circles) are random variables.

of the autoencoder theory, and finally to the combination of the two models: denoising variational autoencoder, which is also the network structure used in this task. Fig. 4.1 shows an illustration of the 4 autoencoder structures.

4.1.1 Autoencoder

An autoencoder (Lecun, 1987; Ballard, 1987) is a network that is trained to reconstruct the input $\boldsymbol{x}$ as its output. Internally, it has a hidden layer $\boldsymbol{h}$ that *encodes* features of the input, which is shown as the layers with 3 nodes in Fig. 4.1. An autoencoder usually consists of two parts: an *encoder* that takes the input and generates the hidden layer code, and a *decoder* which takes the hidden layer value and generates the reconstructed output. Traditionally, the number of values of the hidden layer code is smaller than the input data size. This enables the autoencoder to extract features that ensembles the training data, and avoids learning an identity mapping function, which is not desired. However, modern approaches tend to use overcomplete autoencoders with regularizations.

Such network structures can learn complex data distributions without overfitting (Goodfellow et al., 2016). Formally, the learning process of an autoencoder can be described as minimizing a loss function

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) \ , \tag{4.1}$$

where L is the loss function, f is the nonlinear encoder function and g is the decoder function. Autoencoders are usually used for dimensionality reduction and feature learning.

4.1.2 Denoising Autoencoder

The denoising autoencoder is introduced in (Vincent et al., 2008) as an unsupervised learning method to extract features from input data. Specifically, it is used for regularization to avoid learning an identity function when the network capacity is large. Different from Eqn. 4.1, the loss function for the denoising autoencoder is

$$L(\boldsymbol{x}, g(f(\widetilde{\boldsymbol{x}}))) \ , \tag{4.2}$$

where $\widetilde{\boldsymbol{x}}$ is the input $\boldsymbol{x}$ that is corrupted by some form of noise. The network then learns to reconstruct the clean input $\boldsymbol{x}$ using the corrupted input $\widetilde{\boldsymbol{x}}$. As shown in (Alain and Bengio, 2014), adding noises to the input, which is a way of adding regularization, forces the network to learn the structure of the data distribution.

4.1.3 Variational Autoencoder

Using a deep network as a generative model for computer vision problems, e.g., for image synthesis, image denoising, inpainting, and image super-resolution has been a research topic of great interest recently (Karpathy et al., 2016). Unfortunately, traditional autoencoders are not designed as generative models. A solution is the variational autoencoder (Kingma and Welling, 2013), where units in the hidden layer $\boldsymbol{h}$ are random variables. The decoder part of the variational autoencoder can then be used as a generative model. In my work, I use the idea of variational autoencoders for image synthesis. The following discussions, including both the theoretical explanation of the

variational autoencoder and the reparameterization trick, are taken from (Kingma and Welling, 2013).

I now discuss the theoretical foundation of the variational autoencoder. Given the input data $\boldsymbol{x}$ and the random variable hidden layer of the variational autoencoder $\boldsymbol{h}$, the encoding and decoding operations could be written as $q_\Phi(\boldsymbol{h}|\boldsymbol{x})$ and $p_\Theta(\boldsymbol{x}|\boldsymbol{h})$, where Φ and Θ are the parameters of the encoder and the decoder. The goal of training a variational autoencoder is to find the posterior distribution

$$p_\Theta(\boldsymbol{h}|\boldsymbol{x}) \propto p_\Theta(\boldsymbol{x}|\boldsymbol{h})p(\boldsymbol{h}) \tag{4.3}$$

Unfortunately the integral of the marginal likelihood $p_\Theta(\boldsymbol{x})$ is not tractable, thus the posterior distribution is also intractable. The solution to this problem is to use a variational posterior distribution $q_\Phi(\boldsymbol{h}|\boldsymbol{x})$ to approximate the true posterior distribution. This can be viewed as optimizing the encoder parameters to generate the parameters of a tractable distribution to approximate the intractable posterior from the decoder. This is done by minimizing the Kullback-Leibler (KL) divergence between these two distributions

$$\begin{aligned} D_{\text{KL}}(q_\Phi(\boldsymbol{h}|\boldsymbol{x})||p_\Theta(\boldsymbol{h}|\boldsymbol{x})) &= \mathbb{E}_{q_\Phi(\boldsymbol{h}|\boldsymbol{x})}\left[\log \frac{q_\Phi(\boldsymbol{h}|\boldsymbol{x})}{p_\Theta(\boldsymbol{h}|\boldsymbol{x})}\right] \\ &= \log p_\Theta(\boldsymbol{x}) - \mathbb{E}_{q_\Phi(\boldsymbol{h}|\boldsymbol{x})}\left[\log \frac{p_\Theta(\boldsymbol{h},\boldsymbol{x})}{q_\Phi(\boldsymbol{h}|\boldsymbol{x})}\right] \quad . \end{aligned} \tag{4.4}$$

Since the data $\boldsymbol{x}$ is independent of the latent variable $\boldsymbol{h}$, $\log p_\Theta(\boldsymbol{x})$ in Eqn. (4.4) is constant with respect to $\boldsymbol{h}$ and Φ. Thus, minimizing the KL-divergence is equivalent to maximizing the term $\mathbb{E}_{q_\Phi(\boldsymbol{h}|\boldsymbol{x})}(\log p_\Theta(\boldsymbol{h},\boldsymbol{x}) - \log q_\Phi(\boldsymbol{h}|\boldsymbol{x}))$. Since the KL-divergence is non-negative, I have $\mathbb{E}_{q_\Phi(\boldsymbol{h}|\boldsymbol{x})}[\log p_\Theta(\boldsymbol{h},\boldsymbol{x}) - \log q_\Phi(\boldsymbol{h}|\boldsymbol{x})] \leq \log p_\Theta(\boldsymbol{x})$, This term is called the variational lower bound of the data likelihood $\mathcal{L}_{\text{VAE}}$, i.e.,

$$\begin{aligned} \mathcal{L}_{\text{VAE}} &= \mathbb{E}_{q_\Phi(\boldsymbol{h}|\boldsymbol{x})}\left[\log \frac{p_\Theta(\boldsymbol{h},\boldsymbol{x})}{q_\Phi(\boldsymbol{h}|\boldsymbol{x})}\right] \\ &= -D_{\text{KL}}(q_\Phi(\boldsymbol{h}|\boldsymbol{x})||p_\Theta(\boldsymbol{h})) + \mathbb{E}_{q_\Phi(\boldsymbol{h}|\boldsymbol{x})}[\log p_\Theta(\boldsymbol{x}|\boldsymbol{h})] \quad , \end{aligned} \tag{4.5}$$

where the first term can be regarded as the regularizer, matching the variational posterior to the prior of the latent variable (Gaussian distribution in our case), and the second term is the expected network output likelihood w.r.t. the variational posterior $q_\Phi(\boldsymbol{h}|\boldsymbol{x})$. During training, the optimization algorithm maximizes this variational lower bound.

Directly optimizing the autoencoder based on 4.5 is difficult because the hidden layer node values, which are outputs of the encoder, are random variables as in $\boldsymbol{h} \sim q_\Phi(\boldsymbol{h}|\boldsymbol{x})$. This means that during network back-propagation, one needs to differentiate through the sampling process with respect to the encoder parameter Φ. The solution, proposed in (Kingma and Welling, 2013), is call the reparameterization trick. Specifically, it separates the random variable from the encoder output, making the encoder a deterministic model. For example, in my formulation I assume the variational posterior distribution to be a multivariate Gaussian distribution $\boldsymbol{h} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is a vector and $\boldsymbol{\Sigma}$ is a diagonal matrix. I can then rewrite $\boldsymbol{h}$ as

$$\boldsymbol{h} = \boldsymbol{\mu} + \boldsymbol{\Sigma}\epsilon \ , \ \epsilon \sim \mathcal{N}(0, \boldsymbol{I}) \tag{4.6}$$

where $\boldsymbol{I}$ is an identity matrix. This equation enables me to set the encoder output to the parameters of the Gaussian distribution $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ instead of the sampled values. Then during the network parameter update step, I can first take the derivative of $\boldsymbol{h}$ with respect to the prior distribution parameters μ and σ, and then calculate the gradient of μ and σ through neural network back-propagation. This approach separates the random variable ϵ from the network, making the whole network tractable back-propagation.

4.1.4 Denoising Variational Autoencoder

(Im et al., 2015) propose the denoising variational autoencoder as an improvement to the variational autoencoder. The change they propose is very straightforward and similar to the denoising autoencoder: the data $\boldsymbol{x}$ is corrupted by random noise to be $\widetilde{\boldsymbol{x}}$, which is used as the network input, and the goal is to reconstruct $\boldsymbol{x}$. The data corruption distribution can be written as $p(\widetilde{\boldsymbol{x}}|\boldsymbol{x})$, and the variational posterior distribution is then $\widetilde{q}_\Phi(\boldsymbol{h}|\boldsymbol{x}) = \int q_\Phi(\boldsymbol{h}|\widetilde{\boldsymbol{x}})p(\widetilde{\boldsymbol{x}}|\boldsymbol{x})d\widetilde{\boldsymbol{x}}$. If the

original variational posterior distribution is a Gaussian, this new posterior can be regarded as a mixture of Gaussians where the mixture weight is the data corruption distribution. This means that, theoretically, the posterior distribution for a denoising variational autoencoder has a better representation power than that for a traditional variational autoencoder. As shown in (Im et al., 2015), the variational lower bound for a denoising autoencoder is

$$\mathcal{L}_{\text{DVAE}} = \mathbb{E}_{\widetilde{q}_\Phi(\boldsymbol{h}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{h},\boldsymbol{x})}{q_\Phi(\boldsymbol{h}|\widetilde{\boldsymbol{x}})}\right] \geq \mathcal{L}_{\text{VAE}} = \mathbb{E}_{\widetilde{q}_\Phi(\boldsymbol{h}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{h},\boldsymbol{x})}{\widetilde{q}_\Phi(\boldsymbol{h}|\boldsymbol{x})}\right] \quad . \tag{4.7}$$

This means that the denoising variational lower bound is higher than the original one, leading to a smaller KL-divergence between the true and the approximated posterior.

For my problem I want to synthesize quasi-normal images from pathological images. To do this, I regard lesions as a type of structured noise. Removing lesion appearance is then equivalent to removing noise in the denoising autoencoder theory. In other words, By using the pathological images $\widetilde{\boldsymbol{x}}$ as the network input to a denoising variational autoencoder, I aim to obtain the quasi-normal, "clean" images as $\boldsymbol{x}$.

4.2 Network Structure and Training Strategy

4.2.1 Network Architecture

Fig. 4.2 shows the structure of my denoising variational encoder-decoder network. The input is a brain image $\mathbf{x}$ with intensities normalized to $[0, 1]$. The encoder network consists of convolution followed by max-pooling layers (`ConvPool`), and the decoder has max-unpooling layers followed by convolution (`UnpoolConv`). I choose max-unpooling instead of upsampling as the unpooling operation, because upsampling ignores the pooling location for each pooling patch, which results in severe image degradation. The encoder and decoder are connected by fully connected layers (`FC`) and the re-parameterization layer (`Reparam`) (Kingma and Welling, 2013). As discussed in Sec. 4.1.3, this layer takes the parameters for the variational posterior as input, which in my case is the mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of the Gaussian distribution, and generates a sampled value from the variational posterior. This enables me to compute the gradient of $-D_{\text{KL}}(q_\Phi(\boldsymbol{h}|\boldsymbol{x})||p_\theta(\boldsymbol{h}))$

for Φ using the variational parameters instead of the sampled value, which is not differentiable for Φ. Below I discuss specific techniques implemented for my task.

4.2.2 Training Normal Brain Appearance Using Pathology Images

A model of normal brain appearance would ideally be learned from a large number of healthy brain images with a consistent imaging protocol. My goal, instead, is to learn a mapping from a pathological image to a quasi-normal image, i.e., train a denoising autoencoder for the lesion 'noise', and maximize the denoising variational lower bound. This poses two challenges: *first*, in general, I do not know what the normal appearance in a pathological area should be; *second*, pathological images may exhibit spatial deformations not seen in a normal subject population (such as the mass effect for brain tumors). To mitigate these problems, I learn the brain appearance from the *normal areas of the pathological brain images* only. This can be accomplished by 1) introducing lesion-like structured noise (i.e., circles filled with the mean intensity of the normal brain area for brain tumor cases) via the `QuasiLesion` layer in Fig. 4.2, and 2) *loss function masking*, i.e., ignoring lesion-areas during learning. Suppose I have the lesion segmentation for the training data. For loss-function masking, I first change the input with structured noise $\widetilde{\boldsymbol{x}}$ to $\widetilde{\boldsymbol{x}}_{\text{normal}}$ using the following rule: if $\widetilde{\boldsymbol{x}} \in$ Normal, then $\widetilde{\boldsymbol{x}}_{\text{normal}} = \widetilde{\boldsymbol{x}}$; otherwise, (i.e., $\widetilde{\boldsymbol{x}} \in$ Lesion) $\widetilde{\boldsymbol{x}}_{\text{normal}} = a + \mathcal{N}(0, \sigma)$. This prevents the network from using tumor-appearance. Experiments show only small differences for different settings of a and σ. However, performance suffers when σ is too high, and setting $a = 0$ increases the mean intensity error for the whole image. In my model, I set a to the mean intensity value of the normal area and $\sigma = 0.03$. Second, I set my network output likelihood for $\boldsymbol{x}_{\text{output}}$ to

$$\log p_\theta(\boldsymbol{x}_{\text{output}}|\boldsymbol{h})_{\text{normal}} = \begin{cases} |\boldsymbol{x}_{\text{output}} - \boldsymbol{x}|, & \boldsymbol{x}_{\text{output}} \in \text{Normal} \\ 0, & \boldsymbol{x}_{\text{output}} \in \text{Lesion}. \end{cases} \tag{4.8}$$

Hence, I disregard any errors in the lesion area during back-propagation. I refer to this two-step strategy as *loss-function masking*.

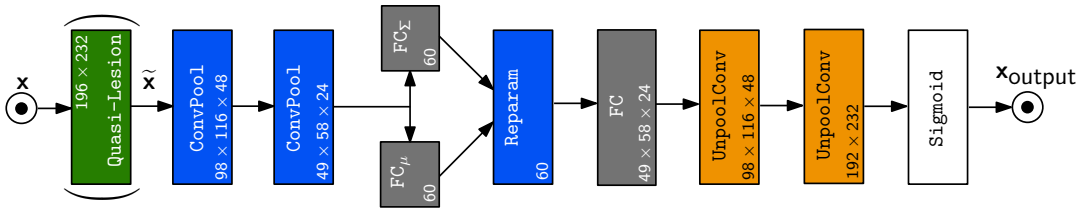


Figure 4.2: Network structure (numbers indicate the data size).

The network is trained using a minibatch of training images as network inputs. For each image $\boldsymbol{x}$ in the minibatch, the training procedure is: 1) sample one corrupted input $\widetilde{\boldsymbol{x}}$ from $p(\widetilde{\boldsymbol{x}}|\boldsymbol{x})$, 2) mask out the lesion area to get $\widetilde{\boldsymbol{x}}_{\text{normal}}$, 3) sample one $\boldsymbol{h}$ from $q_\Phi(\boldsymbol{h}|\widetilde{\boldsymbol{x}}_{\text{normal}})$ and obtain a reconstructed image $\boldsymbol{x}_{\text{output}}$ from the network and 4) calculate the denoising lower bound $\mathcal{L}_{\text{DVAE}}$ with the change in Eqn. (4.8). The network parameters are then updated with respect to the minibatch by back-propagation.

4.3 Registration guidance via Network Uncertainty

During testing, due to the small amount of data available and the possibly large appearance differences among training cases, it is useful to utilize the uncertainty of the reconstructed image to guide registration. In my case, I sample $\boldsymbol{h}$ from the approximated posterior $q_\Phi(\boldsymbol{h}|\boldsymbol{x})$ to generate multiple reconstruction images $\boldsymbol{x}_{\text{output}}$ with different $\boldsymbol{h}$. Then, I choose the mean of the sampled images $\mu_{\boldsymbol{x}_{\text{output}}}$ as the reconstruction result, and the (local) standard deviation $\Sigma_{\boldsymbol{x}_{\text{output}}}$ as uncertainty measure. I define areas of high uncertainty as those areas with large variance, and, for registration, my method down-weights the contribution of those areas to the image similarity measure. I simply use $w(\boldsymbol{x}_{\text{output}}) = \exp(-\Sigma^2_{\boldsymbol{x}_{\text{output}}} \times 2000)$ as a local weight for the image similarity measure in my experiments[2]. This function ensures that the weight drops to near 0 for a large standard deviation. Note that this is different from cost function/pathology masking. Cost function masking uses a simple binary mask, which is equivalent to setting the weight of the lesion area to zero. My uncertainty-based weighting, on the other hand, *downweights* ambiguous areas in the reconstruction process which may not be highly reliable for registration. My uncertainty weight is in $[0, 1]$. Hence,

[2]Other, potentially better choices are of course possible.

structural information is rarely discarded completely as in cost-function masking. My experimental results in Sect. 4.4 show that this is indeed desirable.

4.4 Experimental Results

4.4.1 Experiment Setting

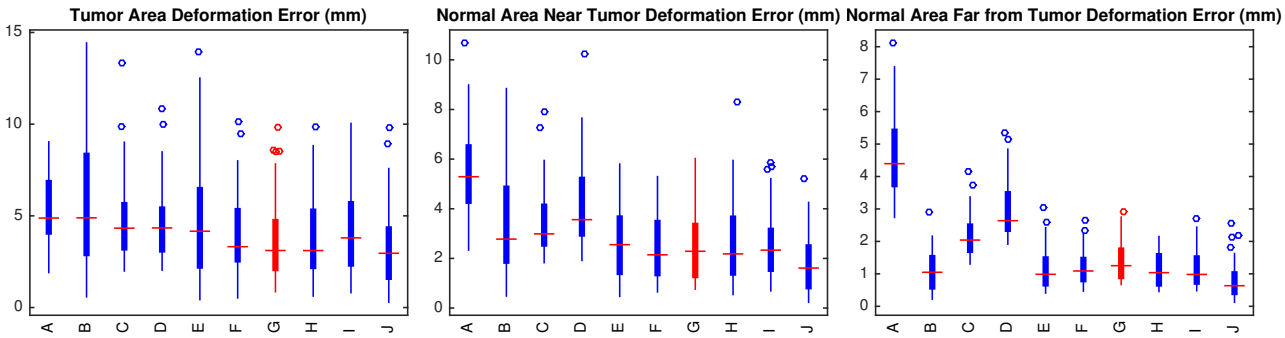


Figure 4.3: *Mean deformation error* of all synthetic tumor test cases for various models. My model is highlighted in red. Masking tumor area = MT. Add structured noise = ASN. Use uncertainty for registration = UR. *(A):* affine registration; *(B):* register to tumor image; *(C):* low-rank-sparse (LRS) with registration; *(D):* LRS w/o registration; *(E):* MT, no ASN, no UR; *(F):* MT, ASN, no UR; *(G):* MT, ASN, UR; *(H):* network trained with clean images, ASN; *(I):* Use uncertainty on tumor image directly; *(J):* cost function masking.

I evaluate my model in two experiments: one using 2D synthetic images, and one with real BRATS tumor images. The image intensity range is $[0, 1]$. I implement the network with Torch and use the *rmsprop* (Tieleman and Hinton, 2012) optimization algorithm; I set the learning rate to 0.0001, the momentum decay to 0.1 and the update decay to 0.01. Further, I use a batch size of 16, and, for a training dataset with 500 images of size 196×232, training 1000 epochs takes about 10 hours on a 2012 NVIDIA Titan GPU. For data augmentation, I apply random shifts up to 10 pixels in both directions for a training image and add zero-mean Gaussian noise with standard deviation of 0.01. During testing, I sample 100 images for each test case, and calculate their mean and standard deviation. All images for training and testing are extracted from the same slice of their original 3D images, which are pre-aligned to a 3D ICBM T1 atlas (Fonov et al., 2011) using affine registration and judged to be limited to having in-plane deformations. I use `NiftyReg` (Modat et al., 2010) (with standard settings) together with normalized cross correlation (NCC) to register

the 2D ICBM atlas slice to the reconstructed result. Note that I modified `NiftyReg` to integrate image uncertainty into the cost function and used a large number of B-spline control points (19×23 for a 196×232 image). This ensures that displacements are large enough to capture the mass effect observed in the BRATS data. B-spline registration approaches similar to `NiftyReg` have successfully been used for registrations of various difficulty (Ou et al., 2014); and given sufficient degrees of freedom poor registration performance is likely due to an unsuitable similarity measure, which should be investigated in future work. To capture even larger deformations, `NiftyReg` could easily be replaced by a fluid-based registration approach. The focus here is to *synthesize* quasi-normal images and to exploit them and their associated reconstruction uncertainty for registration. For my images, 1 pixel corresponds to 1mm $\times$ 1mm.

For comparison, I use the low-rank-plus-sparse (LRS) method (Liu et al., 2015), which is an alternative approach to image synthesis for tumor images. I select the parameters maximizing $2 \times NCC_{\text{tumor}} + NCC_{\text{normal}}$ for the training data. Due to high computational cost of current LRS approaches (Han et al., 2017; Liu et al., 2015), I use 50 training images for each case. Furthermore, to demonstrate that using synthesized images in fact improves registration accuracy, I also compare my method against using the reconstruction uncertainty map in combination with the original tumor image for registration.

4.4.2 Synthetic Data

I use 436 brain images from the OASIS (Marcus et al., 2007) cross-sectional dataset as base images. This chosen dataset is a mix of 43% Alzheimer's and 57% control subjects. I create a synthetic tumor dataset by registering random OASIS images to random BRATS 2015 T1c images (to account for the mass effect of tumors) with tumor area masking, followed by pasting the BRATS' tumor regions into the OASIS images. I generate 500 training and 50 testing images using separate OASIS and BRATS images.

Fig. 4.3 shows boxplots of mean deformation errors of different areas per test case, with respect to the ground truth deformation obtained by registering the atlas to the normal image (i.e., without added tumor). The highlighted boxplot is the network model trained with tumor images, added

quasi-tumor (i.e., structured noise) and using uncertainty weighting for the registration. I evaluate the deformation error for three areas: 1) the tumor areas, 2) normal areas within 10mm from the tumor boundary (*near tumor*) and 3) normal areas more than 10mm away from the boundary (*far from tumor*). By evaluating all three areas I can assess how well the mass effect is captured. This is generally only meaningful for my synthetic experiment. Landmarks (outside the tumor area) are more suitable for real data. For the tumor areas, my method (MT+ASN+UR) outperforms most other methods. For the normal areas, the registration difference between my method and directly registering to the original tumor image is very small, especially compared with the LRS method which tends to remove fine details.

Tumor Area Mean error [mm]

Data Percentile	99.7%	75%	50%	25%	0.3%
Affine (Baseline)	9.03	6.79	4.88	4.14	1.91
Use tumor image directly	14.43	8.28	4.89	2.97	0.58
LRS+registration	9.01	5.59	4.33	3.28	1.99
LRS, no registration	8.49	5.35	4.34	3.16	2.04
Network trained with clean image	8.83	5.21	3.21	2.32	0.62
My model (MT)	12.50	6.40	4.16	2.29	0.43
My model (MT+ASN)	8.00	5.26	3.31	2.62	0.52
My model (MT+ASN+UR)	7.83	4.66	3.11	2.15	0.86
Cost function masking	7.58	4.26	2.96	1.67	0.29

Normal Area Near Tumor ($\leq 10mm$) Mean error [mm]

Data Percentile	99.7%	75%	50%	25%	0.3%
Affine (Baseline)	8.99	6.48	5.29	4.32	2.33
Use tumor image directly	8.34	4.81	2.78	1.90	0.49
LRS+registration	5.95	4.08	2.99	2.60	1.83
LRS, no registration	7.65	5.17	3.56	3.00	1.92
Network trained with clean image	5.94	3.44	2.19	1.45	0.54
My model (MT)	5.80	3.61	2.55	1.45	0.45
My model (MT+ASN)	5.29	3.43	2.14	1.40	0.65
My model (MT+ASN+UR)	6.02	3.31	2.28	1.32	0.76
Cost function masking	4.25	2.44	1.61	0.88	0.23

Tabel 4.1 shows the detailed statistics of deformation error for synthetic test cases. Compared to using the tumor image directly for registration, my model decreases the 99.7% upper limit of the

Normal Area Far from Tumor ($> 10mm$) Mean error [mm]

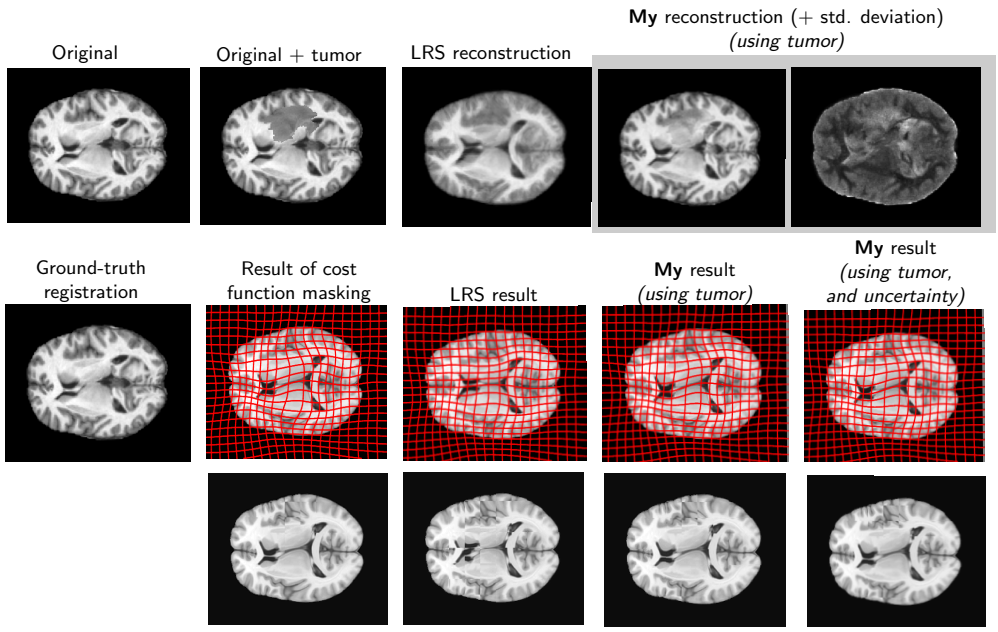
Data Percentile	99.7%	75%	50%	25%	0.3%
Affine (Baseline)	7.37	5.38	4.39	3.76	2.74
Use tumor image directly	2.16	1.49	1.04	0.61	0.22
LRS+registration	3.37	2.46	2.04	1.74	1.30
LRS, no registration	4.84	3.46	2.64	2.39	1.92
Network trained with clean image	2.14	1.49	1.03	0.70	0.47
My model (MT)	2.42	1.45	0.99	0.70	0.41
My model (MT+ASN)	2.23	1.43	1.09	0.83	0.47
My model (MT+ASN+UR)	2.75	1.72	1.25	0.93	0.67
Cost function masking	1.63	0.99	0.63	0.44	0.12

Table 4.1: Statistics table for synthetic test cases for all methods. Masking tumor area = MT. Add structured noise = ASN. Use uncertainty for registration = UR.

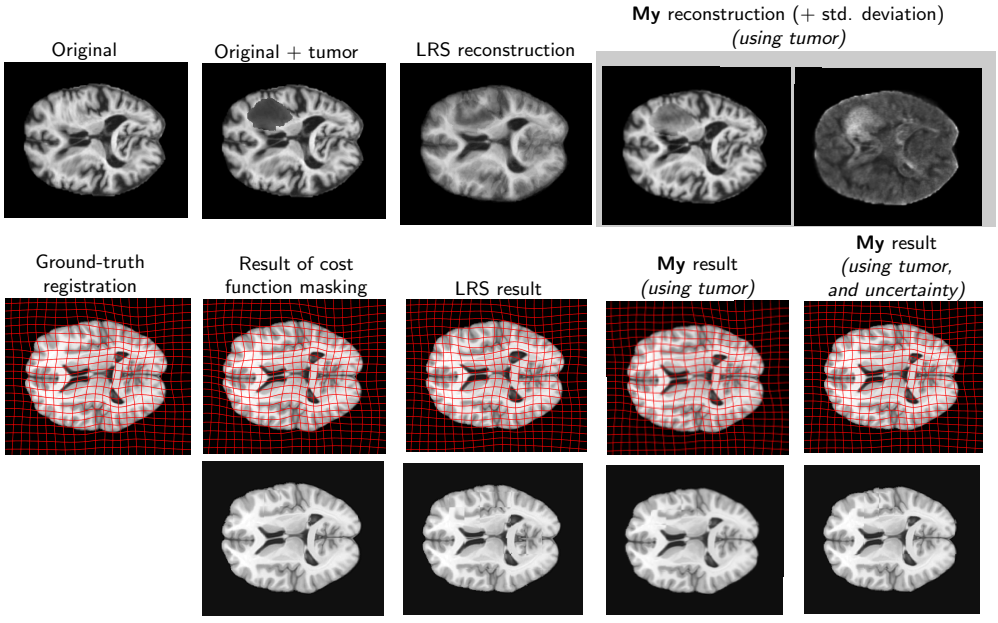
mean of the tumor area deformation error from 14.43mm to 7.83mm, the mean error from 5.62mm to 3.60mm, and the standard deviation from 3.49mm to 2.16mm. The significantly decreased deformation error only causes a small increase of mean deformation error for the normal area, from 1.36mm to 1.49mm. The only method performing better than my model for this *synthetic* test is cost function masking, which *requires* tumor segmentation. Fig. 4.4 shows example synthetic image test cases. Notice that for Fig. 4.4 (a) the LRS method erroneously reconstructs the upper lateral ventricle, resulting in a wrong deformation. In summary, my model performs consistently better than the low-rank method for image reconstruction and atlas registration.

4.4.3 Real Data

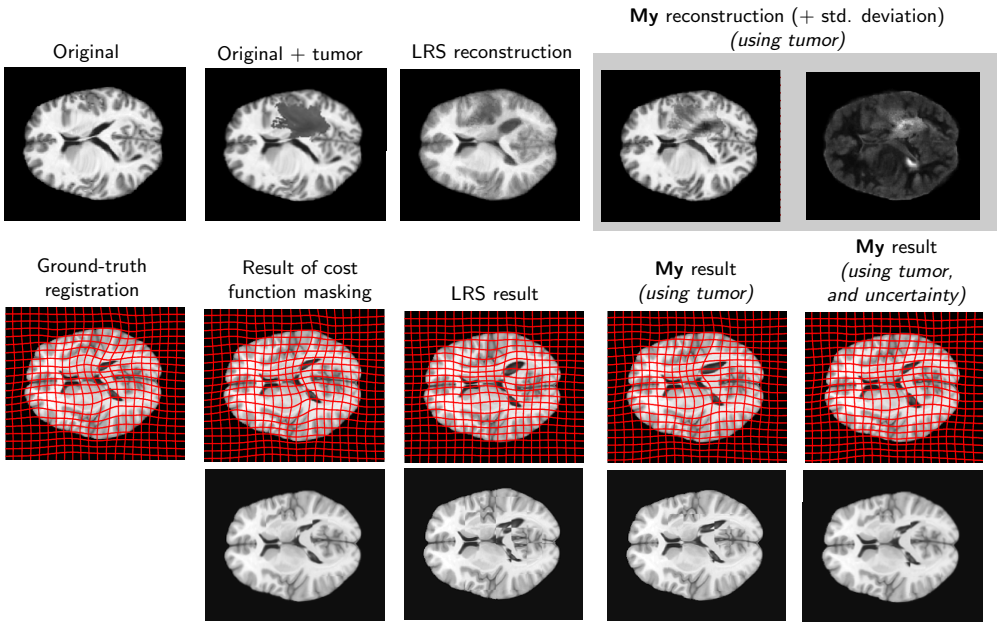
I also evaluate my network using the BRATS 2015 training dataset (Menze et al., 2015), which contains 274 images. This is a very challenging dataset due to moderate sample size and high variations in image appearance and acquisition. I use cross-validation, and partition the dataset into 4 sets of 244 training images and 30 testing images, resulting in a total of 120 test cases. For preprocessing, I standardize image appearance using adaptive histogram equalization (Zuiderveld, 1994). For evaluation, I manually label, on average, 10 landmarks per case around the tumor area and at major anatomical structures for the test images. I report the target registration error for the landmarks in Table 4.2. My method still outperforms most methods, including LRS


Original
Original + tumor
LRS reconstruction
My reconstruction (+ std. deviation)
(using tumor)
Ground-truth registration
Result of cost function masking
LRS result
My result
(using tumor)
My result
(using tumor, and uncertainty)

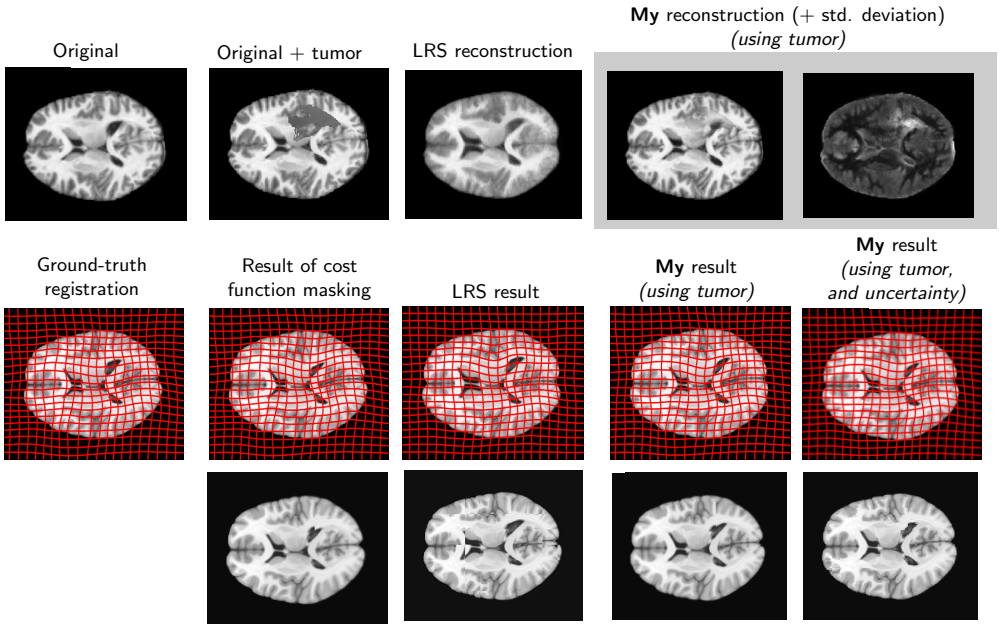
(a)


Original
Original + tumor
LRS reconstruction
My reconstruction (+ std. deviation)
(using tumor)
Ground-truth registration
Result of cost function masking
LRS result
My result
(using tumor)
My result
(using tumor, and uncertainty)

(b)



(c)



(d)

Figure 4.4: Exemplary synthetic tumor test case reconstruction and checkerboard comparison with ground truth registration. Best viewed zoomed-in.

	Mean error [mm]					**Max. error** [mm]				
Data Percentile	99.7%	75%	50%	25%	0.3%	99.7%	75%	50%	25%	0.3%
`Affine (Baseline)`	11.29	6.90	5.06	3.72	2.28	19.25	12.61	9.83	7.04	3.52
Use tumor image directly	6.32	4.20	3.29	2.77	**1.54**	20.48	12.47	7.52	5.22	**1.95**
Cost function masking	6.12	4.22	3.12	2.65	1.89	21.00	11.75	6.98	4.97	2.65
LRS+registration	5.26	3.77	3.06	2.74	1.88	12.04	8.20	6.30	5.45	3.54
LRS, no registration	6.15	4.30	3.25	2.79	2.12	14.62	9.61	6.83	5.72	4.05
Tumor image+uncertainty	5.52	3.79	3.08	2.65	1.81	13.91	8.76	6.24	**4.95**	2.63
My model (no uncertainty)	5.08	3.63	**2.98**	2.64	1.66	12.79	8.12	6.21	4.96	2.82
My model (with uncertainty)	**4.74**	**3.52**	3.02	**2.61**	1.83	**11.77**	**7.99**	**5.93**	5.08	2.48

Table 4.2: Statistics for landmark errors over the BRATS test cases. The best results in each category are marked in **bold**.

without registration. Although, the difference of my model and LRS+registration is not statistically significant, the figures in combination with my synthetic results suggest that my method is overall preferable. Note also that LRS requires image registrations for *each* decomposition iteration and introduces blurring to the brain's normal area (see Fig. 4.5), while my method does not suffer from these problems. Moreover, it is interesting to see that cost function masking performs worse than my method. This could be explained by the observation that in cases where the tumors are very large, cost function masking hides too much of the brain structure, making registration inaccurate. Fig. 4.5 shows one exemplary BRATS test case. Because the tumor covers the majority of the white matter in the left hemisphere, cost function masking removes too much information from the registration. As a result, the left lateral ventricle is mis-registered. Combining my network reconstructed image and uncertainty information, my registration result is much better.

4.4.4 Modeling quasi-tumor appearance

One interesting problem is the choice of quasi-tumor appearance. In my work, I use the mean normal brain area intensity as the appearance, while other choices, such as using simulated[3] tumor appearance or random noise, are also reasonable. To show the effect of quasi-tumor appearance choice on the registration result, I conduct additional experiments using 4 textures to create quasi-tumors: 1) real tumors of the BRATS dataset, 2) mean intensity (my approach), 3) random constant intensities and 4) random noise. Registration performance for all 4 methods is similar, with 2) having

[3]Real tumor appearance is not known in such areas.

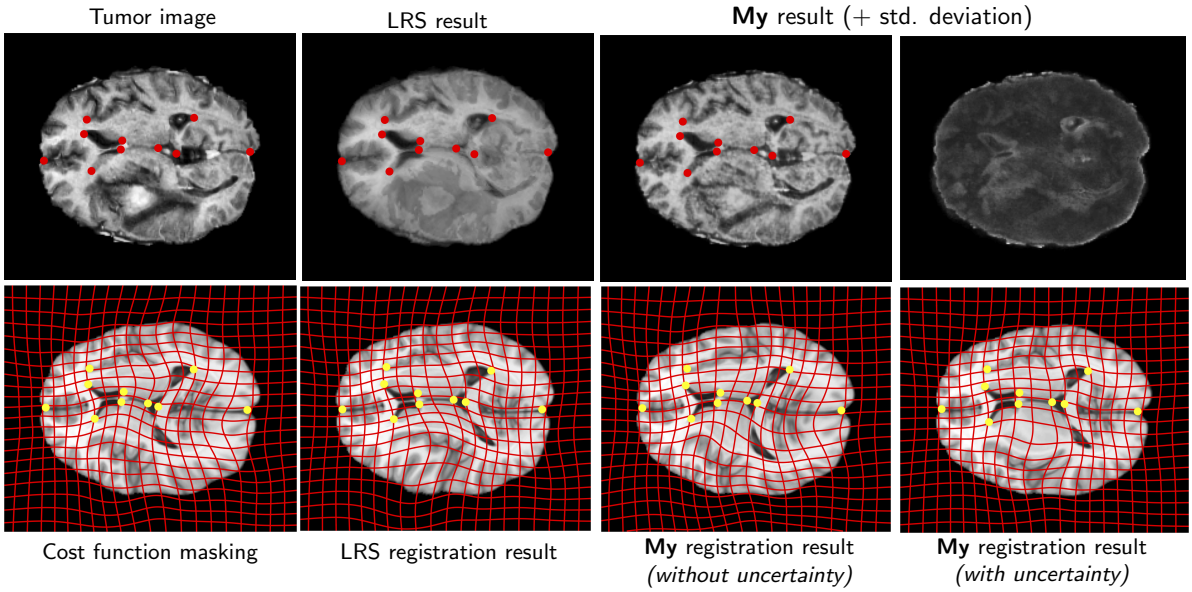


Figure 4.5: Exemplary BRATS test case with landmarks for test image (*top row*) and warped atlas (*bottom row*).

lower registration error in normal areas (e.g., median of 1.07/2.78mm compared to 1.28/2.84mm using 1) for synthetic/BRATS data). A possible reason why using tumor appearance is not superior is the limited training data available ($\sim$200 images). For a larger dataset with more tumor appearance examples to learn from, using tumor appearance could potentially be a better choice.

CHAPTER 5: Fast Predictive Image Registration using Deep Learning

This chapter[1] proposes a deep learning framework for fast image deformation prediction. In this chapter, I change the image registration task into a regression problem where the deformation parameters are predicted using the image appearances, and choose a deep network as a method for prediction. The end result is a framework that generates accurate deformations with an order of magnitude of speed-up compared with optimization-based approaches. The detailed contributions of this chapter are as follows:

- *Convenient parameterization:* Diffeomorphic transformations are desirable in medical image analysis applications to smoothly map between fixed and moving images or to and from an atlas image. Methods, such as LDDMM, with strong theoretical guarantees exist, but are typically computationally very demanding. On the other hand, direct prediction, e.g., of optical flow (Weinzaepfel et al., 2013; Dosovitskiy et al., 2015) is fast, but the regularity of the obtained solution is unclear as it is not considered within the regression formulation. I demonstrate that the momentum-parameterization for LDDMM shooting (Vialard et al., 2012b) is a convenient representation for regression approaches as (i) the momentum is typically compactly supported around image edges and (ii) there are no smoothness requirements on the momentum itself. Instead, smooth velocity fields are obtained in LDDMM from the momentum representation by *subsequent* smoothing. Hence, by predicting the momentum, the proposed method retains all the convenient mathematical properties of LDDMM and, at the same time, is able to predict diffeomorphic transformations *fast*. As the momentum has compact support around image edges, no am-

[1]The work presented in this chapter is based on the previous papers (Yang et al., 2016b, 2017a).

biguities arise within uniform image areas (in which predicting a velocity or deformation field would be difficult).

- *Fast computation:* My method uses a sliding window to locally predict the LDDMM momentum from image patches. I experimentally show that by using patch pruning and a large sliding window stride, my method achieves dramatic speedups compared to the optimization approach, while maintaining good registration accuracy.
- *Uncertainty quantification:* I extend my network to a Bayesian model which is able to determine the uncertainty of the registration parameters and, as a result, the uncertainty of the deformation field. This uncertainty information could be used, e.g., for uncertainty-based smoothing (Simpson et al., 2011), or for surgical treatment planning, or could be directly visualized for qualitative analyses.
- *Correction network:* Furthermore, I propose a correction network to increase the accuracy of the prediction network. Given a trained prediction network, the correction network predicts the difference between the ground truth momentum and the predicted result. The difference is used as a correction to the predicted momentum to increase prediction accuracy. Experiments show that the correction network improves registration results to the point where optimization-based and predicted registrations achieve a similar level of registration accuracy on various registration experiments.
- *Extensive validation:* I extensively validate my fast predictive image registration approach on the four validation datasets of Klein et al. (Klein et al., 2009) and demonstrate registration accuracies on these data sets on par with the state-of-the-art registration approaches. Of note, these registration result are achieved using a model that was trained on an entirely different dataset (images from the OASIS dataset (Marcus et al., 2007)).

The remainder of the chapter is organized as follows. Sec. 5.1 reviews the registration parameterization of the shooting-based LDDMM registration algorithm. Sec. 5.1 introduces my deep network architecture for deformation parameter prediction, the Bayesian formulation of my network, as well as my strategy for speeding up the deformation prediction. Sec. 5.3 discusses the *correction network*

and the reason why it further improves the registration prediction accuracy over an existing prediction network. Finally, Sec. 5.4 presents experimental results for *atlas-to-image* and *image-to-image* registration.

5.1 Momentum Parameterization for Deformation Prediction

As discussed in Chap. 2, given the moving image M and the target image T, the shooting formulation of LDDMM image registration can be written as

$$E(m_0) = \langle m_0, Km_0 \rangle + \frac{1}{\sigma^2}||M \circ \Phi^{-1}(1) - T||^2, \quad \text{s.t.} \tag{5.1}$$

$$\begin{aligned} m_t + \text{ad}_v^* m &= 0, \\ m(0) &= m_0, \\ \Phi_t^{-1} + D\Phi^{-1} v &= 0, \\ \Phi^{-1}(0) &= \text{id}, \\ m - Lv &= 0. \end{aligned} \tag{5.2}$$

where m_0 is the initial momentum, Φ^{-1} is the inverse deformation field, id is the identity map, and the operator ad^* is the dual of the negative Jacobi-Lie bracket of vector fields, i.e., $\text{ad}_v w = -[v, w] = Dvw - Dwv$. In my framework, I predict the initial momentum m_0 given the moving and target images in a patch-by-patch manner. Using the initial momentum for patch-based prediction is a convenient parameterization because (i) the initial momentum is generally not smooth, but is compactly supported at image edges and (ii) the initial velocity is generated by applying a smoothing kernel K to the initial momentum. Therefore, the smoothness of the deformation does not need to be specifically considered during the parameter prediction step, but is imposed *after* the prediction. Since K governs the theoretical properties or LDDMM, a strong K assures diffeomorphic transformations[2], making predicting the initial momentum an ideal choice. However, predicting alternative parameterizations such as the initial velocity or directly the displacement fields

[2]See (Beg et al., 2005; Dupuis et al., 1998) for the required regularity conditions.

would make it difficult to obtain diffeomorphic transformations. Furthermore, it is hard to predict initial velocity or displacements for homogeneous image regions, as these regions locally provide no information from which to predict the spatial transformation. In these regions the deformations are purely driven by regularization. This is not a problem for the initial momentum parameterization, since the initial momentum in these areas, for image-based LDDMM, is zero. This is because for image-based LDDMM (Vialard et al., 2012b; Niethammer et al., 2011) the momentum can be written as $m(x,t) = \lambda(x,t)\nabla I(x,t)$, where λ is a scalar field and ∇I is the spatial gradient of the image. Hence, for homogeneous areas, $\nabla I = 0$ and consequentially $m = 0$. Fig. 5.1 illustrates this graphically. In summary, the initial momentum parameterization is ideal for my patch-based prediction method.

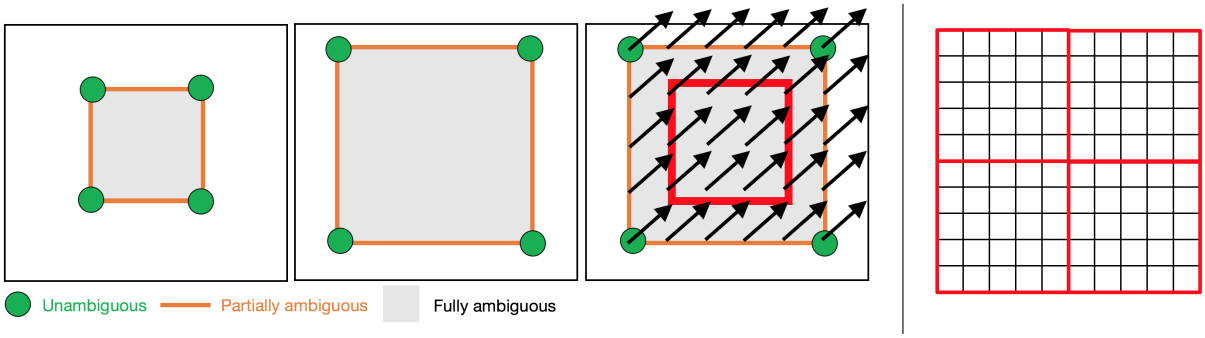


Figure 5.1: **Left:** The LDDMM momentum parameterization is ideal for patch-based prediction of image registrations. Consider registering a small to a large square with uniform intensity. Only the corner points suggest clear spatial correspondences. Edges also suggest spatial correspondences, however, correspondences between *individual* points on edges remain ambiguous. Lastly, points interior to the squares have ambiguous spatial correspondences, which are established purely based on regularization. Hence, predicting velocity or displacement fields (which are spatially dense) from patches is challenging in these interior areas, in the absence of sufficient spatial context. On the other hand, LDDMM theory shows that the optimal momentum m to match images can be written as $m(x,t) = \lambda(x,t)\nabla I(x,t)$, where $\lambda(x,t) \mapsto \mathbb{R}$ is a spatio-temporal scalar field and $I(x,t)$ is the image at time t. Hence, in spatially uniform areas (where correspondences are ambiguous) $\nabla I = 0$ and consequentially $m(x,t) = 0$. This is highly beneficial for prediction as the momentum only needs to be predicted at image edges. **Right:** Furthermore, as the momentum is not spatially smooth, the regression approach does not need to account for spatial smoothness, which allows predictions with non-overlapping or hardly-overlapping patches. This is not easily possible for the prediction of displacement or velocity fields since these are expected to be spatially dense and smooth, which would need to be considered in the prediction.

5.2 Prediction Network Structure

Fig. 5.2 shows the structure of the initial momentum prediction network, and Table 5.1 shows the detailed configuration of the network. I first discuss the deterministic version of the network without dropout layers. I then introduce the Bayesian version of my network where dropout layers are used to convert the architecture into a probabilistic deep network. Finally, I discuss my strategy for patch pruning to reduce the number of patches needed for whole image prediction.

5.2.1 Deterministic Network

In my formulation, the network input(s) are two 3D image patches, extracted from the same location of the moving and target images, respectively. These are the same locations with respect to image grid coordinates as the images are still unregistered at this point. The network output is the predicted initial vector-valued momentum patch separated into the x, y and z dimensions respectively. The prediction network consists of two parts: an *encoder* and a *decoder* which I describe next.

Encoder. The encoder consists of two parallel encoders which learn features from the moving/target image patches independently. I use a VGG-style network that stacks multiple convolution layers at the same image scale to increase the receptive field, and use pooling to decrease the feature size while increasing the number of features. Each encoder contains two blocks of three $3 \times 3 \times 3$ convolution+PReLU (He et al., 2015) layers, followed by another $2 \times 2 \times 2$ convolution+PReLU with a stride of two, as shown in the **Encoder** part in Fig. 5.2. The latter essentially performs the pooling operation. The number of features in the first block is $F = 64$ and increases to $F = 128$ in the second block. The learned features from the two encoders are then concatenated and sent to three parallel decoders (one for each dimension x, y, z).

Decoder. Each decoder's structure is the inverse of the encoder, except that the number of features is doubled ($F = 256$ in the first block and $F = 128$ in the second block) as the initial input is obtained from *two* encoders. I use the 3D transpose convolution layers (Long et al., 2015) shown as the cyan layers in Fig. 5.2 to perform "unpooling", and omit the non-linearity after the final convolution

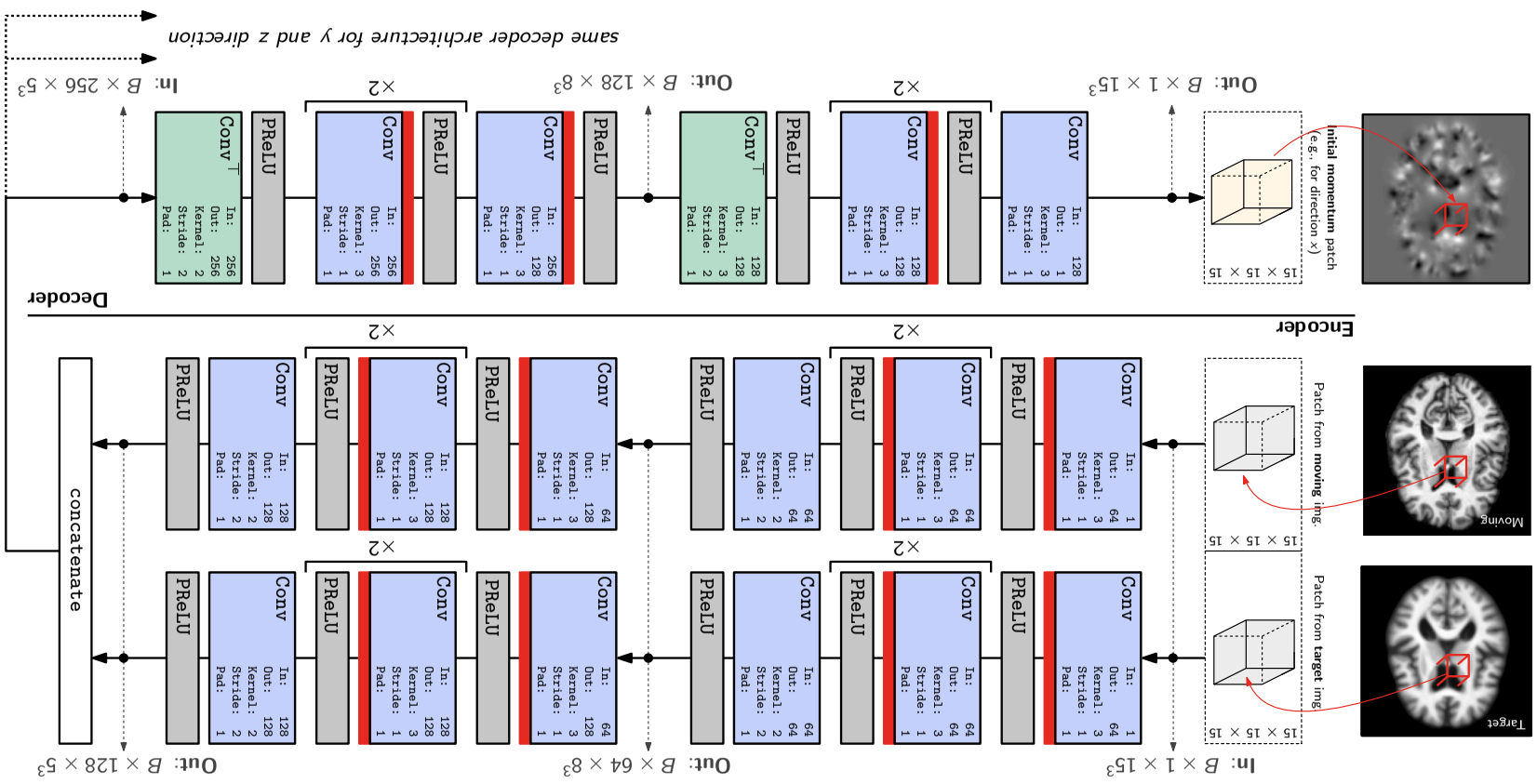


Figure 5.2: 3D (probabilistic) network architecture. The network takes two 3D patches from the moving and target image as the input, and outputs 3 3D initial momentum patches (one for each of the x, y and z dimensions respectively; for readability, only one decoder branch is shown in the figure). In case of the deterministic network, see Sec. 5.2.1, the dropout layers, illustrated by ■, are removed. `Conv`: 3D convolution layer. `Conv`$^\intercal$: 3D transposed convolution layer. Parameters for the `Conv` and `Conv`$^\intercal$ layers: `In`: input channel. `Out`: output channel. `Kernel`: 3D filter kernel size in each dimension. `Stride`: stride for the 3D convolution. `Pad`: zero-padding added to the boundaries of the input patch. Note that in this illustration B denotes the batch size.

layer. The idea of using convolution and the transpose convolution to learn the pooling/unpooling operation is motivated by (Springenberg et al., 2014), and it is especially suited for my network as the two encoders perform pooling independently which prevents me from using the pooling index for unpooling in the decoder. During training, I use the l_1 norm between the predicted and the desired momentum to measure the prediction error. When predicting the deformation parameters for the whole image, I follow a sliding window strategy to predict the initial momentum in a patch-by-patch manner and then average the overlapping areas of the patches to obtain the final prediction result.

One question that naturally arises is why to use independent encoder/decoder in the prediction network. For the decoder part, I observed that an independent decoder structure is much easier to train than a network with one large decoder (twice the number of features of a single decoder in my network) to predict the initial momentum in all dimensions simultaneously. In my experiments, such a combined network easily got stuck in poor local minima. As to the encoders, experiments do not show an obvious difference of the prediction accuracy between using two independent encoders and one single large encoder. However, such a two-encoder strategy is beneficial when extending the approach to multi-modal image registration (Yang et al., 2017b). Hence, using a two-encoder strategy here will make the approach easily retrainable for multi-modal image registration. In short, my network structure can be viewed as a multi-input multi-task network where each encoder learns features for one patch source, and each decoder uses the shared image features from the encoders to predict one spatial dimension of the initial momenta.

5.2.2 Probabilistic Network

I extend my architecture to a probabilistic network using dropout (Srivastava et al., 2014), which can be viewed as (Bernoulli) approximate inference in Bayesian neural networks (Gal and Ghahramani, 2015, 2016). Given a training input $\mathbf{X}$ and its corresponding output $\mathbf{Y}$, I aim to find weights $\mathbf{W}$ of the convolution layers that optimize the likelihood that given input $\mathbf{X}$, the network is likely to generate $\mathbf{Y}$. I define this likelihood as $p(\mathbf{Y}|\mathbf{W}, \mathbf{X})$, i.e., measured via the l_1-norm in my network. Given input $\mathbf{X}$ and output $\mathbf{Y}$, the goal is to find the posterior distribution of the

convolutional weights $\mathbf{W}$, i.e.,

$$p(\mathbf{W}|\mathbf{Y},\mathbf{X}) \propto \frac{p(\mathbf{Y}|\mathbf{W},\mathbf{X})p(\mathbf{W})}{p(\mathbf{Y})} \ , \tag{5.3}$$

where $p(\mathbf{W}) \sim \mathcal{N}(\mathbf{0}, \frac{1}{\lambda} \times \mathbf{I})$ is the prior distribution over the convolutional weights, λ is the weight decay parameter[3] and $p(\mathbf{Y})$ is constant. As this posterior is generally unknown, I use a variational posterior $q(\mathbf{W})$ to approximate the true posterior. This is usually done by minimizing the Kullback-Leibler (KL) divergence between the two distributions

$$\begin{aligned} D_{\mathrm{KL}}(q(\mathbf{W}) \,||\, p(\mathbf{W}|\mathbf{X},\mathbf{Y})) &= \mathbb{E}_{q(\mathbf{W})}(\log \frac{q(\mathbf{W})}{p(\mathbf{W}|\mathbf{X},\mathbf{Y})}) \\ &= \log(p(\mathbf{Y})) + D_{\mathrm{KL}}(q(\mathbf{W}) \,||\, p(\mathbf{W})) + \mathbb{E}_{q(\mathbf{W})}(\log \frac{1}{p(\mathbf{Y}|\mathbf{W},\mathbf{X})}) \ . \end{aligned} \tag{5.4}$$

In this equation $D_{\mathrm{KL}}(q(\mathbf{W}) \,||\, p(\mathbf{W}))$ can be seen as the regularization term and $\mathbb{E}_{q(\mathbf{W})}(\log \frac{1}{p(\mathbf{Y}|\mathbf{W},\mathbf{X})})$ is the data likelihood term. By defining the approximating variational posterior for the weights $\mathbf{W}_i$ of the i-th convolution layer as

$$q(\mathbf{W}_i) = \mathbf{M}_i \cdot \mathrm{diag}([z_{i,j}]_{j=1}^{K_i}), \quad z_{i,j} \sim \mathrm{Bernoulli}(p_i) \ , \tag{5.5}$$

I see that sampling from $q(\mathbf{W}_i)$ is equivalent to dropout (with probability p) after the i-th convolution layer with weights $\mathbf{M}_i$. The variational posterior of the overall network weights $\mathbf{W}$ is then $q(\mathbf{W}) = \prod_{i=1}^{N} q(\mathbf{W}_i)$, were N is the number of layers in the network. In my implementation, I add dropout layers after all convolutional layers except for those used as pooling/unpooling layers, and train the network using stochastic gradient descent (SGD). *According to (Gal and Ghahramani, 2015), this is equivalent to minimizing the KL-divergence between the true and the variational posterior distribution of* $\mathbf{W}$.

[3]It is possible to set λ to 0, meaning no weight decay is used during the deep network optimization. In this case the prior distribution $p(\boldsymbol{W})$ changes to a constant. This also means that $D_{\mathrm{KL}}(q(\mathbf{W}) \,||\, p(\mathbf{W}))$ in Eqn. 5.4 is then the negative of entropy of q plus a constant, which still acts as a regularization term.

Network evaluation. For testing, I keep the dropout layers to maintain the probabilistic property of the network, and sample the network to obtain multiple momentum predictions for one moving/target image pair. I then choose the sample mean as the prediction result, and perform LDDMM shooting using all the samples to generate multiple deformation fields. The local variance of these deformation fields can then be used as an uncertainty estimate of the predicted deformation field. When selecting the dropout probability, a probability of 0.5 provides the largest variance, but may also enforce too much regularity for a convolutional network, especially in my case where dropout layers are added after every convolution layer. In my experiments, I use a dropout probability of 0.2 as a balanced choice.

Network structure	Output data size
Input	$2 \times 15 \times 15 \times 15$
2 independent encoders	
Encoder input	$15 \times 15 \times 15$
Conv (1 input, 64 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$64 \times 15 \times 15 \times 15$
Conv (64 input, 64 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$64 \times 15 \times 15 \times 15$
Conv (64 input, 64 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$64 \times 15 \times 15 \times 15$
Conv (64 input, 64 output, $2 \times 2 \times 2$ filter, 2 stride, 1 padding) + PReLU	$64 \times 8 \times 8 \times 8$
Conv (64 input, 128 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$128 \times 8 \times 8 \times 8$
Conv (128 input, 128 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$128 \times 8 \times 8 \times 8$
Conv (128 input, 128 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$128 \times 8 \times 8 \times 8$
Conv (128 input, 128 output, $2 \times 2 \times 2$ filter, 2 stride, 1 padding) + PReLU	$128 \times 5 \times 5 \times 5$
Concatenate outputs from 2 encoders	$256 \times 5 \times 5 \times 5$
3 independent decoders	
Decoder input	$256 \times 5 \times 5 \times 5$
Conv^{T}(256 input, 256 output, $2 \times 2 \times 2$, 2 stride, 1 padding) + PReLU	$256 \times 8 \times 8 \times 8$
Conv (256 input, 256 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$256 \times 8 \times 8 \times 8$
Conv (256 input, 256 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$256 \times 8 \times 8 \times 8$
Conv (256 input, 128 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$128 \times 8 \times 8 \times 8$
Conv^{T}(128 input, 128 output, $3 \times 3 \times 3$, 2 stride, 1 padding) + PReLU	$128 \times 15 \times 15 \times 15$
Conv (128 input, 128 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$128 \times 15 \times 15 \times 15$
Conv (128 input, 128 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding) + PReLU + Dropout	$128 \times 15 \times 15 \times 15$
Conv (128 input, 1 output, $3 \times 3 \times 3$ filter, 1 stride, 1 padding)	$15 \times 15 \times 15$
Concatenate outputs from 3 decoders (final momentum prediction)	$3 \times 15 \times 15 \times 15$

Table 5.1: Detailed probablistic network configuration, together with data size after every convolution operation.

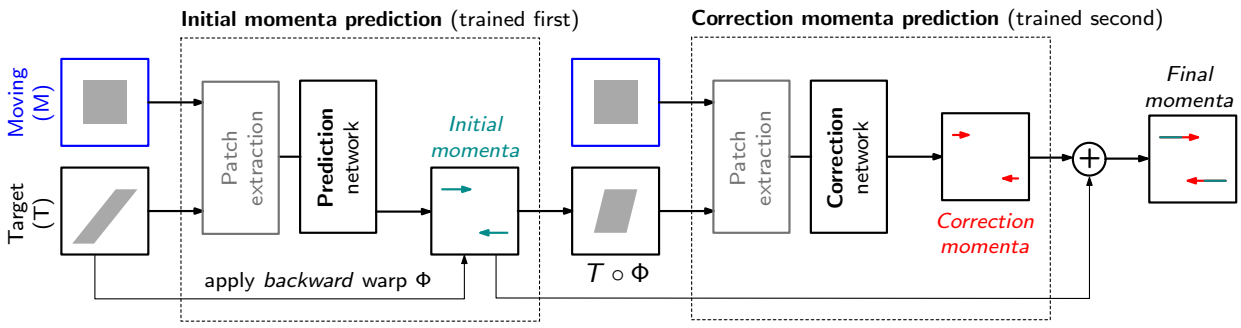


Figure 5.3: The full prediction + correction architecture for LDDMM momenta.

5.2.3 Speeding Up Whole Image Prediction with Patch Pruning

As discussed in Sec. 5.2.1, I use a sliding-window approach to predict the deformation parameters (the momenta for LDDMM) patch-by-patch for a whole image. Thus, computation time is proportional to the number of the patches I need to predict. When using a 1-voxel sliding window stride, the number of patches to predict for a whole image could be substantial. For a typical 3D image of size $128 \times 128 \times 128$ using a $15 \times 15 \times 15$ patch for prediction will require more than 1.4 million patch predictions. Hence, I use two techniques to drastically reduce the number of patches needed for deformation prediction. First, I perform patch pruning by ignoring all patches that belong to the background of both the moving image and the target image. This is justified, because according to LDDMM theory the initial momentum in constant image regions, and hence also in the image background, should be zero. Second, I use a large voxel stride (e.g., 14 for $15 \times 15 \times 15$ patches) for the sliding window operations. This is reasonable for my initial momentum parameterization because of the compact support (at edges) of the initial momentum and the spatial shift invariance I obtain via the pooling/unpooling operations. By using these two techniques, I can reduce the number of predicted patches for one single image dramatically. For example, by 99.995% for 3D brain images of dimension $229 \times 193 \times 193$.

5.3 Correction Network

There are two main shortcomings of the deformation prediction network: (i) the complete iterative numerical approach typically used for LDDMM registration is replaced by a *single* predic-

tion step. Hence, it is not possible to recover from any prediction errors; (ii) to facilitate training a network with a small number of images, to make predictions easily parallelizable, and to be able to perform predictions for large 3D image volumes, the prediction network predicts the initial momentum *patch-by-patch*. However, since patches are extracted at the same spatial grid locations from the moving and target images, large deformations may result in drastic appearance changes between a source and a target patch. In the extreme case, corresponding image information may no longer be found for a given source and target patch pair. This may happen, for example, when a small patch-size encounters a large deformation. While using larger patches would be an option, this would require a network with substantially larger capacity (to store the information for larger image patches and all meaningful deformations) and would also likely require much larger training data sets[4].

To address these shortcomings, I propose a two-step prediction approach to improve overall prediction accuracy. The first step is my already described prediction network. I refer to the second step as the *correction network*. The task of the correction network is to compensate for prediction errors of the first prediction step. The idea is grounded in two observations: The first observation is that patch-based prediction is accurate when the deformation inside the patch is small. This is sensible as the initial momentum is concentrated along the edges, small deformations are commonly seen in training images, and less deformation results in less drastic momentum values. Hence, more accurate predictions are expected for smaller deformations. My second observation is that, given the initial momentum, I am able to generate the whole geodesic path using the geodesic shooting equations. Hence, I can generate two deformation maps: the forward warp Φ^{-1} that maps the moving image to the coordinates of the target image, and the backward warp Φ mapping the target image back to the coordinates of the moving image. Hence, after the first prediction step using my prediction network, I can warp the target image back to the moving image M via $T \circ \Phi$. I can then train the *correction network* based on the difference between the moving image M and the

[4]In fact, I have successfully trained prediction models with as little as ten images using all combinations of pair-wise registrations to create training data (Yang et al., 2017b). This is possible, because even in such a case of severely limited training data the number of *patches* that can be used for training is very large.

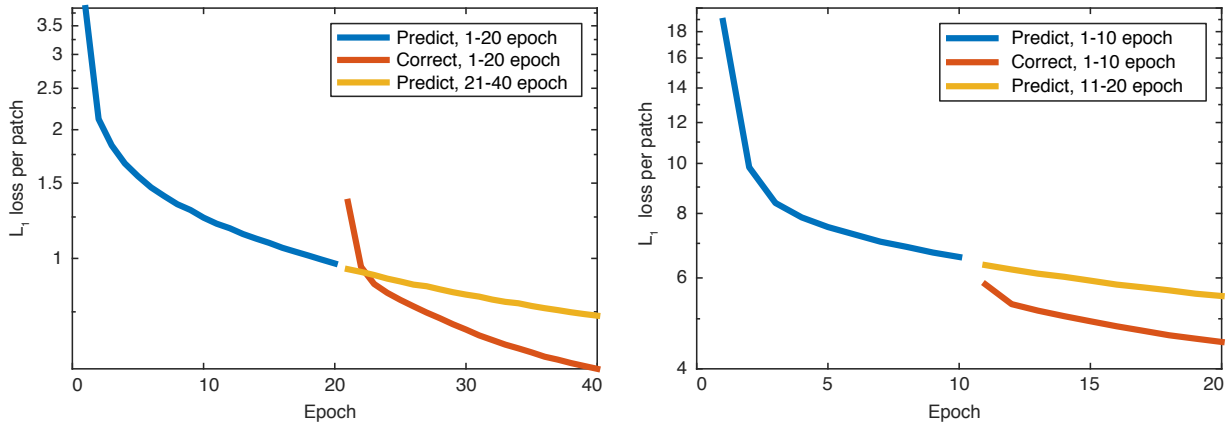
warped-back target image $T \circ \Phi$, such that it makes adjustments to the initial momentum predicted in the first step by my prediction network. Because M and $T \circ \Phi$ are in the same coordinate system, the differences between these two images are small as long as the predicted deformation is reasonable, and more accurate predictions can be expected. Furthermore, the correction for the initial momentum is then performed in the original coordinate space (of the moving image) which allows me to obtain an overall corrected initial momentum, m_0. This is, for example, a useful property when the goal is to do statistics with respect to a fixed coordinate system, for example, an atlas coordinate system.

Fig. 5.3 shows a graphical illustration of the resulting two-step prediction framework. In the framework, the correction network has the same structure as the prediction network, and the only difference is the input of the networks and the output they produce. Training the overall framework is done sequentially. First, I train the prediction network using training images and the ground truth initial momentum obtained by numerical optimization of the LDDMM registration model. Then I use the *predicted* momentum from the prediction network to warp the target images in the training dataset back to the moving image. I use the moving images and the warped-back target images to train the correction network. The correction network learns to predict the *difference* between the ground truth momentum and the predicted momentum from the prediction network. During test time, the outputs from the prediction network and the correction network are summed up to obtain the final predicted initial momentum. This summation is justified from the LDDMM theory as it is performed in a fixed coordinate system, which is the coordinate system of the moving image. Experiments show that my prediction+correction approach results in lower training and testing error compared with only using a prediction network, as shown in Sec. 5.4.

5.4 Experiments

5.4.1 Data and Settings

I evaluate my method using two 3D brain image registration experiments. The first experiment is designed to assess *atlas-to-image* registration. In this experiment the moving image is always the atlas image. The second experiment addresses general *image-to-image* registration.



(a) `Atlas-to-Image` (b) `Image-to-Image`

Figure 5.4: Log_{10} plot of l_1 training loss per patch. The loss is averaged across all iterations for every epoch for both the Atlas-to-Image case and the Image-to-Image case.

For the *atlas-to-image* registration experiments, I use 3D image volumes from the OASIS longitudinal dataset (Marcus et al., 2007). Specifically, I use the first scan of all subjects, resulting in 150 brain images. I select the first 100 images as my training target images and the remaining 50 as my test target images. I create an unbiased atlas (Joshi et al., 2004) from all training data using `PyCA`[5] (Singh et al., 2013b,a), and use the atlas as the moving image. I use the LDDMM shooting algorithm to register the atlas image to all 150 OASIS images. The obtained initial momenta from the training data are used to train my network; the remaining momenta are used for validation.

For the *image-to-image* registration experiment, I use all 373 images from the OASIS longitudinal dataset as the training data, and randomly select target images from different subjects for every image, creating 373 registrations for training. For testing, I choose the four datasets (`LPBA40`, `IBSR18`, `MGH10`, `CUMC12`) evaluated in (Klein et al., 2009). I perform LDDMM shooting for all training registrations, and follow the evaluation procedure described in (Klein et al., 2009) to perform pairwise registrations within all datasets, resulting in a total of 2168 registration (1560 from `LPBA40`, 306 from `IBSR18`, 90 from `MGH10`, 132 from `CUMC12`) test cases.

[5] `https://bitbucket.org/scicompanat/pyca`

All images used in my experiments are first affinely registered to the ICBM MNI152 nonlinear atlas (Grabner et al., 2006) using `NiftyReg` and intensity normalized via histogram equalization[6] prior to atlas building and LDDMM registration. All 3D volumes are of size $229 \times 193 \times 193$ except for the LPBA dataset ($229 \times 193 \times 229$), where I add additional blank image voxels for the atlas to keep the cerebellum structure. LDDMM registration is done using `PyCA` (Singh et al., 2013b) with sum-of-squared-differences (SSD) as the image similarity measure. I set the parameters for the regularizer of LDDMM[7] to $L = -a\nabla^2 - b\nabla(\nabla\cdot) + c$ as $[a, b, c] = [0.01, 0.01, 0.001]$, and σ in Eqn. 5.1 to 0.2. I use a $15\times15\times15$ patch size for deformation prediction in all cases, and use a sliding window with step-size 14 to extract patches for training. The network is implemented in `PyTorch`, and optimized using `Adam` (Kingma and Ba, 2014), where I set the learning rate to 0.0001 and keep the remaining parameters at their default values. I train the prediction network for 10 epochs for the image-to-image registration experiment and 20 epochs for the atlas-to-image experiment. The correction networks are trained using the same number of epochs as their corresponding prediction networks. Fig. 5.4 shows the l_1 training loss per patch averaged for every epoch. For both, the atlas-to-image and the image-to-image cases, using a correction network in conjunction with a prediction network results in lower training error compared with training the prediction network for more epochs.

5.4.2 Atlas-to-image Experiments

For the atlas-to-image registration experiment, I test two different sliding window strides for my patch-based prediction method: stride = 5 and stride = 14. I trained additional prediction networks predicting the initial velocity $v_0 = Km_0$ and the displacement field $\Phi^{-1}(1) - \text{id}$ of LDDMM to show the effect of different deformation parameterizations on deformation prediction accuracy.

[6]Of course, more advanced image intensity normalization techniques could be used in this step for potentially better results.

[7]This regularizer is too weak to assure a diffemorphic transformation based on the *sufficient* regularity conditions discussed in (Beg et al., 2005). For these conditions to hold in 3D, L would need to be at least a differential operator of order 6. However, as long as the obtained velocity fields v are finite over the unit interval, i.e., $\int_0^1 \|v\|_L^2 \, dt < \infty$ for an L of at least order 6, I will obtain a diffeomorphic transform (Dupuis et al., 1998). In the discrete setting, this condition will be fulfilled for finite velocity fields. To side-step this issue, models based on Gaussian or multi-Gaussian kernels (Bruveris et al., 2012) could also be used instead.

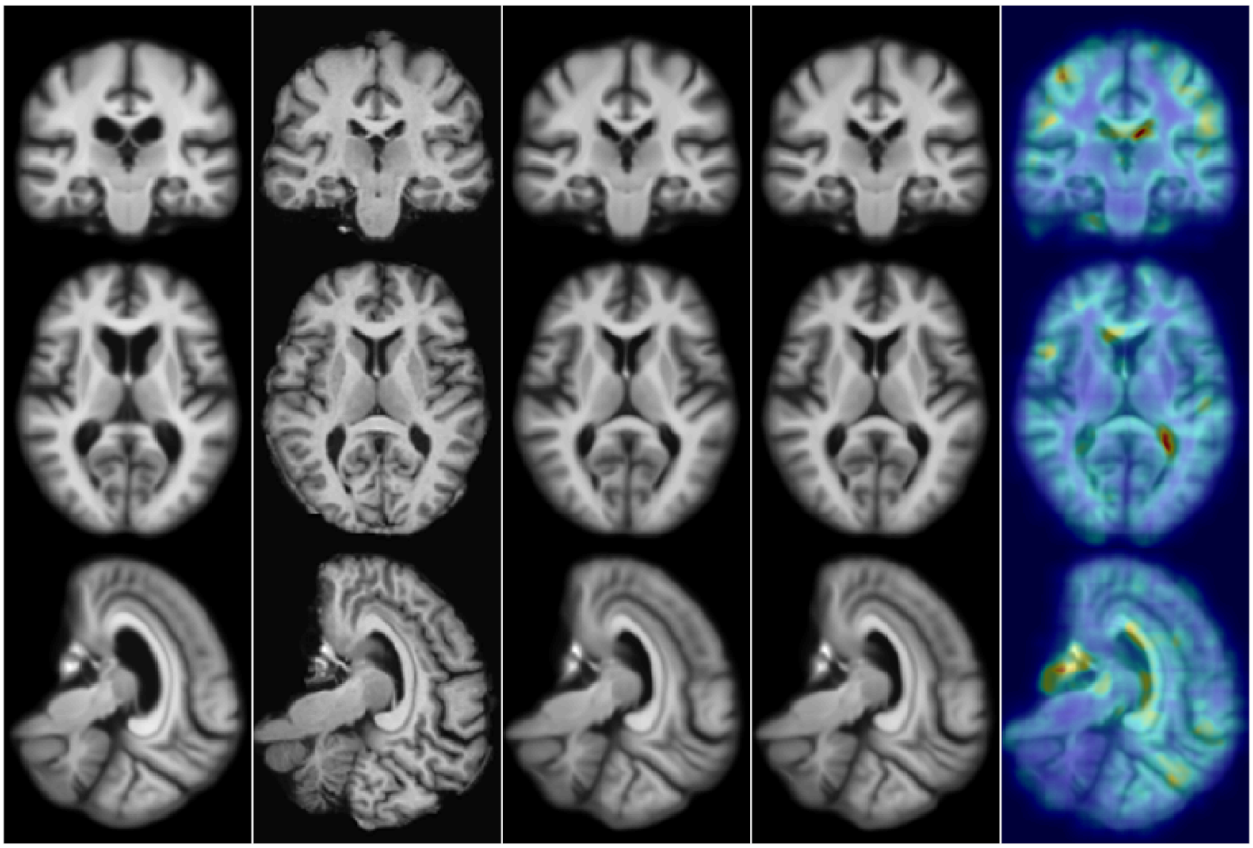


Figure 5.5: Atlas-to-image registration example. From *left* to *right*: moving (atlas) image, target image, deformation from optimizing LDDMM energy, deformation from using the mean of 50 samples from the probabilistic network with stride=14 and patch pruning, and the uncertainty map as square root of the sum of the variances of the deformation in x, y, and z directions mapped onto the predicted deformation result. The coloring indicates the level of uncertainty, with red = high uncertainty and blue = low uncertainty (best-viewed in color).

I generate the predicted deformation map by integrating the shooting equation Eqn. 5.2 for the initial momentum and the initial velocity parameterization respectively. For the displacement parameterization I can directly read-off the map from the network output. I quantify the deformation errors per voxel using the voxel-wise two-norm of the deformation error with respect to the result obtained via numerical optimization for LDDMM using `PyCA`. Table 5.2 shows the error percentiles over all voxels and test cases.

I observe that the initial momentum network has better prediction accuracy compared to the results obtained via the initial velocity and displacement parameterization in both the 5-stride and 14-stride cases. This validates my hypothesis that momentum-based LDDMM is better suited for

	Deformation Error [voxel]							**det$J > 0$**
Data Percentile	0.3%	5%	25%	50%	75%	95%	99.7%	
`Affine`	0.0613	0.2520	0.6896	1.1911	1.8743	3.1413	5.3661	N/A
`D`, velocity, stride 5	0.0237	0.0709	0.1601	0.2626	0.4117	0.7336	1.5166	100%
`D`, velocity, stride 14	0.0254	0.075	0.1675	0.2703	0.415	0.743	1.5598	100%
`D`, deformation, stride 5	0.0223	0.0665	0.1549	0.2614	0.4119	0.7388	1.5845	56%
`D`, deformation, stride 14	0.0242	0.0721	0.1671	0.2772	0.4337	0.7932	1.6805	0%
`P`, stride 14 + PR, 50 samples	0.0166	0.0479	0.1054	0.1678	0.2546	0.4537	1.1049	100%
`D`, stride 5	0.0129	0.0376	0.0884	0.1534	0.2506	0.4716	1.1095	100%
`D`, stride 14	0.013	0.0372	0.0834	0.1359	0.2112	0.3902	0.9433	100%
`D`, stride 14, 40 epochs	0.0119	0.0351	0.0793	0.1309	0.2070	0.3924	0.9542	100%
`D`, stride 14 + correction	**0.0104**	**0.0309**	**0.0704**	**0.1167**	**0.185**	**0.3478**	**0.841**	100%

Table 5.2: Test result for *atlas-to-image* registration. `D`: deterministic network; `P`: probabilistic network; stride: stride length of sliding window for whole image prediction; velocity: predicting initial velocity; deformation: predicting the deformation field; correction: using the correction network. The **det$J > 0$** column shows the ratio of test cases with only positive-definite determinants of the Jacobian of the deformation map to the overall number of registrations (100% indicates that all registration results were diffeomorphic). My initial momentum networks are highlighted in **bold**. The best results are also highlighted in **bold**.

patch-wise deformation prediction. I also observe that the momentum prediction result using a smaller sliding window stride is slightly worse than the one using a stride of 14. This is likely the case, because in the atlas-to-image setting, the number of patches sent into the encoder that extract features from the atlas image is very limited, and using a stride of 14 during the training phase further reduces the available data from the atlas image. Thus, during testing, the encoder will perform very well for the 14-stride test cases since it has already seen all the input atlas patches during training. For a stride of 5 however, unseen atlas patches will be input to the network, resulting in reduced registration accuracy[8]. In contrast, the velocity and the displacement parameterizations result in slightly better predictions for smaller sliding window strides. That this is not the case for the momentum parameterization suggests that it is easier for the network to learn to predict the momentum, as it indeed has become more specialized to the training data which was obtained with a stride of 14. One of the important properties of LDDMM shooting is its ability to generate diffeomorphic deformations. To assess this property, I calculate the local Jacobians of the resulting

[8]This behavior could likely be avoided by randomly sampling patch locations during training instead of using a regular grid. However, since I aim at reducing the number of predicted patches I did not explore this option and instead maintained the regular grid sampling.

deformation maps. Assuming no flips of the entire coordinate system, a diffeomorphic deformation map should have positive definite Jacobian everywhere, otherwise foldings occur in the deformation maps. I calculate the ratio of test cases with positive Jacobian of the deformation maps to all test cases, shown as **det**$J > 0$ in Table 5.2. I observe that the initial momentum and the initial velocity networks indeed generate diffeomorphic deformations in all scenarios. However, the deformation accuracy is significantly worse for the initial velocity network. Predicting the displacement directly cannot guarantee diffeomorphic deformations even for a small stride. This is unsurprising as, similar to existing optical flow approaches (Weinzaepfel et al., 2013; Dosovitskiy et al., 2015), directly predicting displacements does not encode deformation smoothness. Hence, the initial momentum parameterization is the preferred choice among my three tested parameterizations as it achieves the best prediction accuracy and guarantees diffeomorphic deformations. Furthermore, the initial momentum prediction including the correction network with a stride of 14 achieves the best registration accuracy overall among the tested methods, even outperforming the prediction network alone trained with more training iterations (**`D`, stride 14, 40 epochs**). This demonstrates that the correction network is capable of improving the initial momentum prediction beyond the capabilities of the original prediction network.

Fig. 5.5 shows one example atlas-to-image registration case. The predicted deformation result is very similar to the deformation from LDDMM optimization. I compute the square root of the sum of the variance of the deformation in the x, y and z directions to quantify deformation uncertainty, and visualize it on the rightmost column of the figure. The uncertainty map shows high uncertainty along the ventricle areas where drastic deformations occur, as shown in the moving and target images.

5.4.3 Image-to-image Experiments

In this experiment, I use a sliding window stride of 14 for both the prediction network and the correction network during evaluation. For image-to-image registration, I follow the approach in (Klein et al., 2009) and calculate the target overlap (TO) of labeled brain regions after registration: $TO = \frac{l_m \cap l_t}{l_t}$, where l_m and l_t indicate the corresponding labels for the moving image (after

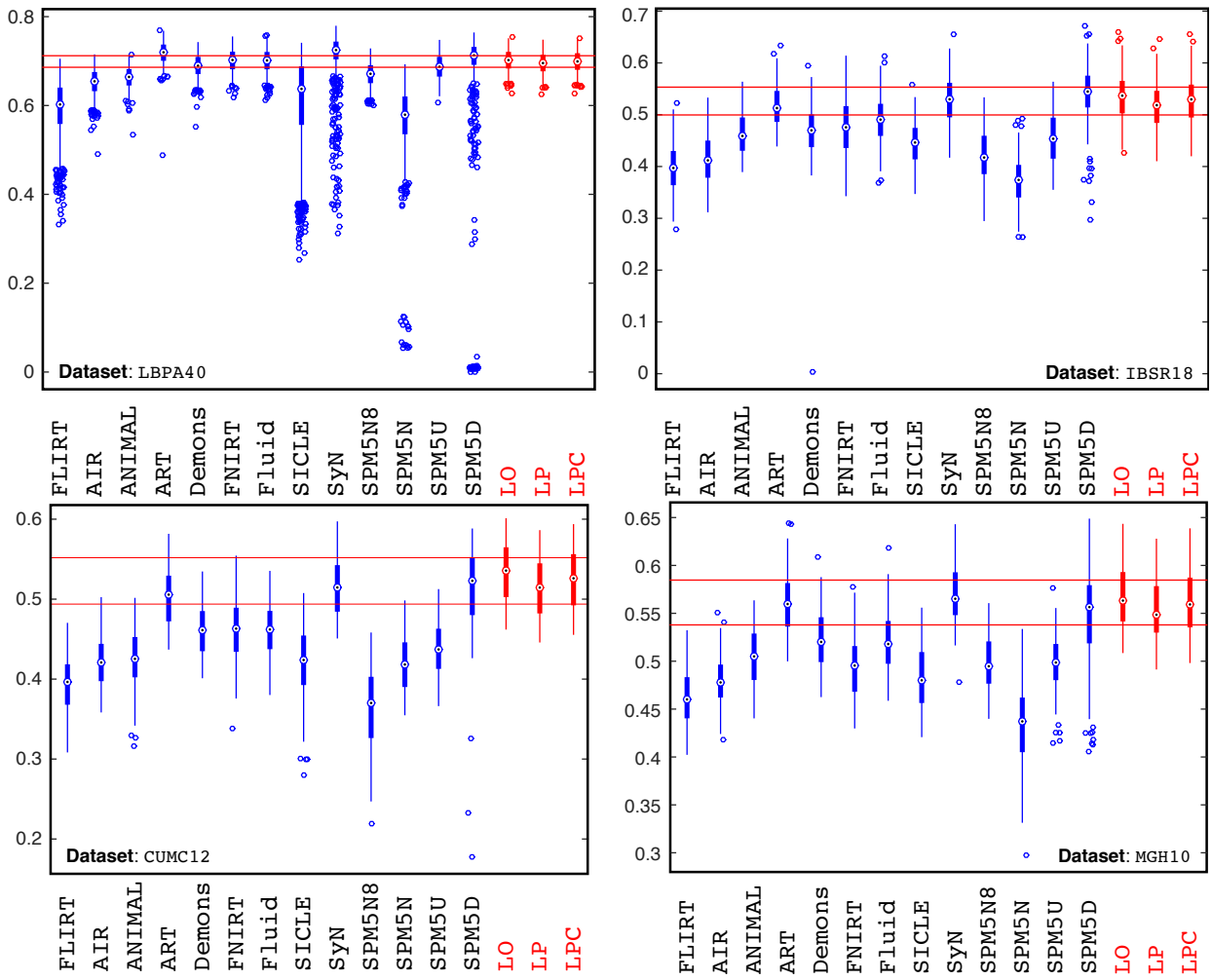
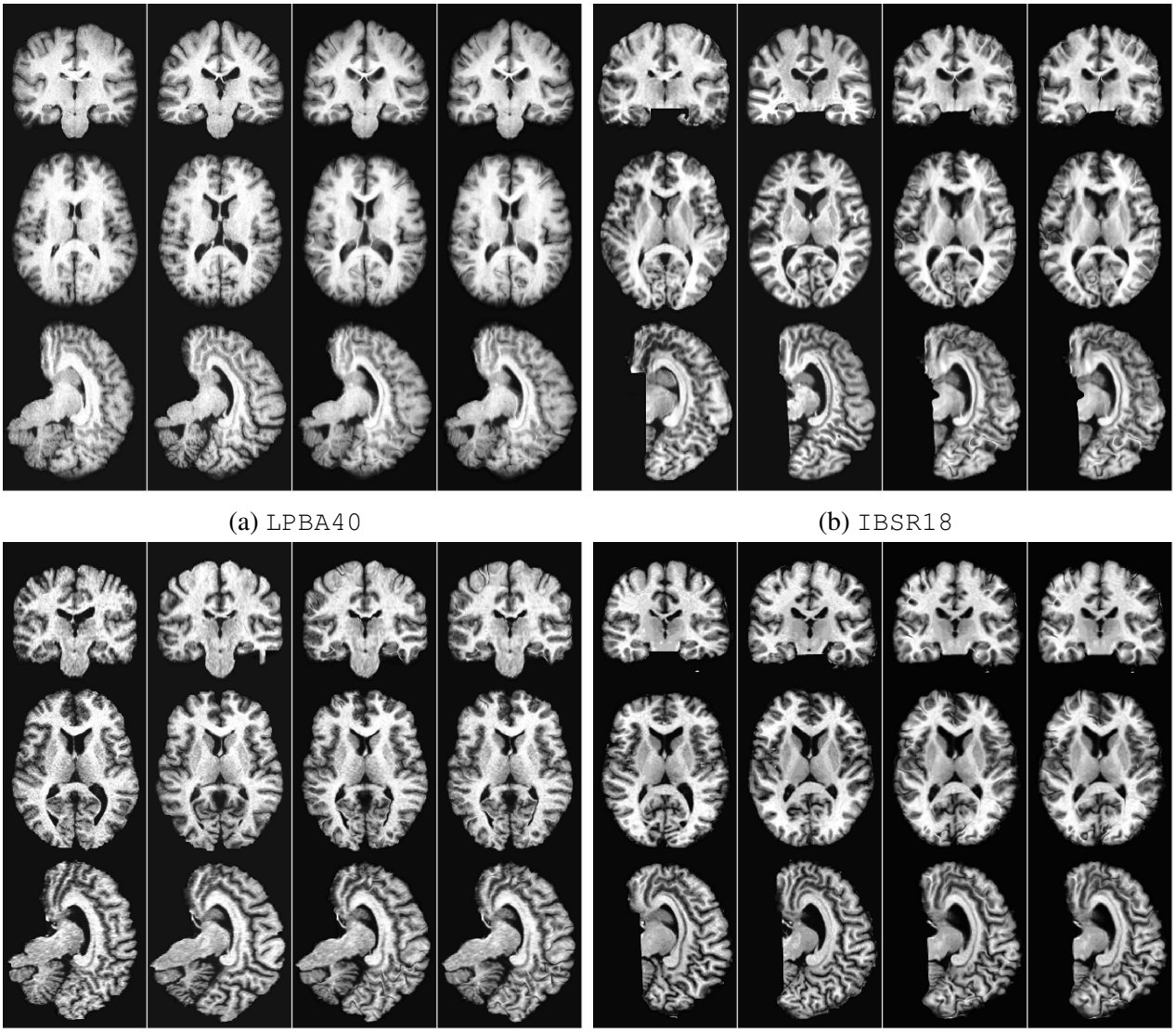


Figure 5.6: Overlap by registration method for the *image-to-image* registration case. The boxplots illustrate the mean target overlap measures averaged over all subjects in each label set, where mean target overlap is the average of the fraction of the target region overlapping with the registered moving region over all labels. LDDMM-based methods are highlighted in red. `LO` = LDDMM optimization; `LP` = prediction network; `LPC` = prediction network + correction network. Horizontal red lines show the `LPC` performance in the lower quartile to upper quartile (best-viewed in color).

registration) and the target image. I then evaluate the mean of the target overlap, averaged first across all labels for a single registration case and then across all registration cases within one dataset. The evaluation results for other methods tested in (Klein et al., 2009) are available online. I compare my registration approaches to these results. I choose three LDDMM-based methods for evaluation: (i) the numerical LDDMM optimization approach (`LO`) as implemented in `PYCA`, which acts as an upper bound on the performance of my prediction methods; and two flavors of my method: (ii) only the prediction network (`LP`) and (iii) the prediction+correction network (`LPC`). Fig. 5.6 shows the evaluation results. Several points should be noted: first, the LDDMM optimization performance is on par with `SyN` (Avants et al., 2008), `ART` (Ardekani et al., 2005) and the SPM5 DARTEL Toolbox (`SPM5D`) (Ashburner, 2007). This is reasonable as these methods are all non-parametric diffeomorphic or homeomorphic registration methods, allowing the modeling of large deformations between image pairs. Second, using only the prediction network results in a slight performance drop compared to the numerical optimization results (`LO`), but the result is still competitive with the top-performing registration methods. Furthermore, using the correction network boosts the deformation accuracy nearly to the same level as the LDDMM optimization approach (`LO`). The red horizontal lines in Fig. 5.6 show the lower and upper quartiles of the target overlap score of the prediction+correction method. Compared with other methods, my prediction+correction network achieves top-tier performance for label matching accuracy at a small fraction of the computational cost. Lastly, in contrast to many of the other methods the proposed method produces virtually no outliers. One can speculate that this may be the benefit for learning to predict deformations from a *population* of data, which may result in a prediction model which conservatively rejects unusual deformations.

To study the differences of registration algorithms *statistically*, I performed paired t-tests with respect to the target overlap scores between my LDDMM variants (`LO`, `LP`, `LPC`) and the methods in (Klein et al., 2009). My null-hypothesis is that the methods show the same target overlap scores. I use Bonferroni correction to safe-guard against spurious results due to multiple comparisons by dividing the significance level α by 204 (the total number of statistical tests). The



(a) LPBA40 (b) IBSR18

(c) CUMC12 (d) MGH10

Figure 5.7: Example test cases for the image-to-image registration. For every figure from left to right: moving image, target image, registration result from optimizing LDDMM energy, and registration result from prediction+correction network.

significance level for rejection of the null-hypothesis is $\alpha = 0.05/204$. I also computed the mean and the standard deviation of pair-wise differences between my LDDMM variants and these other methods. Table 5.3 shows the results. I observe that direct numerical optimization of the shooting LDDMM formulation via PyCA (LO) is a highly competitive registration method and shows better target overlap scores than most of the other registration algorithms for all four datasets (LPBA40,

IBSR18, CUMC12, and MGH10). Notable exceptions are ART (on LPBA40), SyN (on LBPA40), and SPM5D (on IBSR18). However, performance decreases are generally very small: -0.017, -0.013, and -0.009 mean decrease in target overlap ratio for the three aforementioned exceptions, respectively. Specifically, a similar performance of LO to SyN, for example, is expected as SyN (as used in (Klein et al., 2009)) is based on a *relaxation* formulation of LDDMM, whereas LO is based on the shooting variant of LDDMM. Performance differences may be due to differences in the used regularizer and the image similarity measure. In particular, where SyN was used with Gaussian smoothing and cross-correlation, I used the sum-of-squared intensity differences as the image similarity measure and a regularizer involving up to second order spatial derivatives.

LO is the algorithm that my predictive registration approaches (LP and LPC) are based on. Hence, LP and LPC are not expected to show improved performance with respect to LO. However, similar performance for LP and LPC would indicate high quality predictions. Indeed, Table 5.3 shows that my prediction+correction approach (LPC) performs similar (with respect to the other registration methods) to LO. A slight performance drop with respect to LO can be observed for LPC and a slightly bigger performance drop for LP, which only uses a single prediction model, but no correction model. Just as for LO, performance of the predictive models is generally high and performance decreases are small.

To assess statistical equivalence of the top performing registration algorithms I performed paired two one-sided tests (paired TOST) (Wellek, 2010) using a relative threshold difference of 2% with Bonferroni correction. In other words, my null-hypothesis is that methods show a relative difference of larger than 2%. Rejection of this null-hypothesis at a significance level of $\alpha = 0.05/204$ then indicates statistical equivalence. Table 5.4 shows the paired TOST results. For a relative threshold difference of 2%, LPC can be considered statistically equivalent to LO for all four datasets and to many of the other top methods (e.g., LPC *vs.* SyN on MGH10 and IBSR18).

Overall, these statistical tests confirm that my predictive models, in particular LPC, are highly competitive registration algorithms. Computational cost, however, is very small. This is discussed in detail in Sec. 5.4.4.

Dataset: LPBA40								
	FLIRT	AIR	ANIMAL	ART	Demons	FNIRT	Fluid	SLICE
LO	0.108 ± 0.054	0.049 ± 0.021	0.039 ± 0.029	-0.017 ± 0.013	0.012 ± 0.014	0.001 ± 0.014	0.001 ± 0.013	0.097 ± 0.1
LP	0.102 ± 0.054	0.043 ± 0.02	0.033 ± 0.029	-0.024 ± 0.013	0.006 ± 0.014	-0.006 ± 0.014	-0.005 ± 0.013	0.091 ± 0.1
LPC	0.106 ± 0.054	0.046 ± 0.021	0.037 ± 0.029	-0.02 ± 0.013	0.009 ± 0.014	-0.002 ± 0.014	-0.002 ± 0.013	0.095 ± 0.1
	SyN	SPM5N8	SPM5N	SPM5U	SPM5D	LO	LP	LPC
LO	-0.013 ± 0.05	0.032 ± 0.018	0.13 ± 0.07	0.015 ± 0.017	0.03 ± 0.16	N/A	0.006 ± 0.003	0.003 ± 0.002
LP	-0.02 ± 0.05	0.025 ± 0.018	0.124 ± 0.07	0.009 ± 0.017	0.023 ± 0.16	-0.006 ± 0.003	N/A	-0.004 ± 0.002
LPC	-0.016 ± 0.05	0.029 ± 0.018	0.127 ± 0.07	0.012 ± 0.017	0.027 ± 0.16	-0.003 ± 0.002	0.004 ± 0.002	N/A
Dataset: IBSR18								
	FLIRT	AIR	ANIMAL	ART	Demons	FNIRT	Fluid	SLICE
LO	0.136 ± 0.025	0.119 ± 0.03	0.07 ± 0.027	0.018 ± 0.022	0.064 ± 0.034	0.057 ± 0.026	0.044 ± 0.019	0.088 ± 0.029
LP	0.118 ± 0.022	0.101 ± 0.028	0.052 ± 0.025	0 ± 0.021	0.047 ± 0.032	0.039 ± 0.023	0.026 ± 0.018	0.07 ± 0.027
LPC	0.129 ± 0.024	0.112 ± 0.03	0.063 ± 0.027	0.01 ± 0.022	0.058 ± 0.033	0.049 ± 0.025	0.036 ± 0.019	0.08 ± 0.029
	SyN	SPM5N8	SPM5N	SPM5U	SPM5D	LO	LP	LPC
LO	0.005 ± 0.024	0.112 ± 0.034	0.161 ± 0.042	0.08 ± 0.030	-0.009 ± 0.035	N/A	0.018 ± 0.007	0.007 ± 0.004
LP	-0.013 ± 0.024	0.094 ± 0.032	0.144 ± 0.042	0.062 ± 0.027	-0.026 ± 0.035	-0.018 ± 0.007	N/A	-0.01 ± 0.004
LPC	-0.002 ± 0.024	0.105 ± 0.034	0.154 ± 0.043	0.073 ± 0.029	-0.016 ± 0.035	-0.007 ± 0.004	0.01 ± 0.004	N/A

Table 5.3: Mean and standard deviation of the difference of target overlap score between LDDMM variants (LDDMM optimization (LO), the proposed prediction network (LP) and prediction+correction network (LPC)) and all other methods for the *image-to-image* experiments. The cell coloring indicates significant differences calculated from a pair-wise t-test: green indicates that the row-method is statistically significantly *better* than the column-method; red indicates that the row-method is statistically significantly *worse* than the column-method, and blue indicates the difference is not statistically significant (best-viewed in color). We use Bonferroni correction to safe-guard against spurious results due to multiple comparisons by dividing the significance level α by 204 (the total number of statistical tests). The significance level for rejection of the null-hypothesis is $\alpha = 0.05/204$. Continued on next page.

Dataset: CUMC12								
	FLIRT	AIR	ANIMAL	ART	Demons	FNIRT	Fluid	SLICE
LO	0.14 ± 0.02	0.111 ± 0.019	0.108 ± 0.031	0.031 ± 0.01	0.072 ± 0.012	0.071 ± 0.019	0.073 ± 0.017	0.115 ± 0.03
LP	0.12 ± 0.017	0.092 ± 0.017	0.089 ± 0.031	0.012 ± 0.01	0.052 ± 0.01	0.052 ± 0.017	0.053 ± 0.015	0.096 ± 0.031
LPC	0.131 ± 0.018	0.102 ± 0.018	0.1 ± 0.031	0.023 ± 0.01	0.063 ± 0.011	0.062 ± 0.018	0.064 ± 0.016	0.107 ± 0.031
	SyN	SPM5N8	SPM5N	SPM5U	SPM5D	LO	LP	LPC
LO	0.020 ± 0.011	0.169 ± 0.029	0.114 ± 0.019	0.1 ± 0.015	0.022 ± 0.049	N/A	0.02 ± 0.004	0.009 ± 0.002
LP	0.001 ± 0.011	0.149 ± 0.028	0.095 ± 0.017	0.076 ± 0.013	0.003 ± 0.048	-0.02 ± 0.004	N/A	-0.011 ± 0.003
LPC	0.012 ± 0.011	0.16 ± 0.028	0.106 ± 0.018	0.087 ± 0.013	0.013 ± 0.048	0.009 ± 0.002	0.011 ± 0.003	N/A
Dataset: MGH10								
	FLIRT	AIR	ANIMAL	ART	Demons	FNIRT	Fluid	SLICE
LO	0.104 ± 0.016	0.087 ± 0.015	0.062 ± 0.022	0.005 ± 0.016	0.044 ± 0.013	0.071 ± 0.018	0.043 ± 0.016	0.083 ± 0.017
LP	0.091 ± 0.016	0.073 ± 0.016	0.049 ± 0.023	-0.008 ± 0.017	0.03 ± 0.013	0.058 ± 0.018	0.03 ± 0.015	0.07 ± 0.017
LPC	0.098 ± 0.016	0.081 ± 0.015	0.057 ± 0.022	0 ± 0.016	0.038 ± 0.013	0.065 ± 0.018	0.037 ± 0.016	0.077 ± 0.017
	SyN	SPM5N8	SPM5N	SPM5U	SPM5D	LO	LP	LPC
LO	-0.002 ± 0.015	0.069 ± 0.02	0.135 ± 0.041	0.07 ± 0.024	0.023 ± 0.047	N/A	0.013 ± 0.004	0.006 ± 0.002
LP	-0.015 ± 0.016	0.055 ± 0.02	0.121 ± 0.041	0.057 ± 0.025	0.01 ± 0.048	-0.013 ± 0.004	N/A	-0.008 ± 0.003
LPC	-0.008 ± 0.015	0.063 ± 0.02	0.129 ± 0.041	0.065 ± 0.024	0.018 ± 0.047	-0.006 ± 0.002	0.008 ± 0.003	N/A

Table 5.3: Continued from previous page.

Dataset: LPBA40					
	ART	SyN	LO	LP	LPC
LO			N/A	✓	✓
LP			✓	N/A	✓
LPC			✓	✓	N/A

Dataset: IBSR18					
	ART	SyN	LO	LP	LPC
LO		✓	N/A		✓
LP	✓			N/A	
LPC		✓	✓		N/A

Dataset: CUMC12					
	ART	SyN	LO	LP	LPC
LO			N/A		✓
LP		✓		N/A	
LPC			✓		N/A

Dataset: MGH10					
	ART	SyN	LO	LP	LPC
LO		✓	N/A		✓
LP				N/A	✓
LPC	✓		✓	✓	N/A

Table 5.4: Pairwise TOST, where I test the null-hypothesis that for the target overlap score for each row-method, t_{row}, and the target overlap score for each column-method, t_{column}, $\frac{t_{\text{row}}}{t_{\text{column}}} < 0.98$, or $\frac{t_{\text{row}}}{t_{\text{column}}} > 1.02$. Rejecting the null-hypothesis indicates that the row-method and column-method are statistically equivalent. Equivalence is marked as ✓s in the table. We use Bonferroni correction to safe-guard against spurious results due to multiple comparisons by dividing the significance level α by 204 (the total number of statistical tests). The significance level for rejection of the null-hypothesis is $\alpha = 0.05/204$.

5.4.4 Runtime Study

I assess the runtime of the proposed method on a single Nvidia Titan X (Pascal) GPU. Performing LDDMM optimization using the GPU-based implementation of `PyCA` for a $229 \times 193 \times 193$ 3D brain image takes approximately 10.8 minutes. Using my prediction network with a sliding window stride of 14, the initial momentum prediction time is, on average, 7.63 seconds. Subsequent geodesic shooting to generate the deformation field takes 8.9 seconds, resulting in a total runtime of 18.43 seconds. Compared to the LDDMM optimization approach, my method achieves a $35\times$ speed up. Using the correction network together with the prediction network doubles the computation time, but the overall runtime is still an order of magnitude faster than direct LDDMM optimization. Note that, at a stride of 1, computational cost increases about 3000-fold in 3D, resulting in runtimes of about $5^1/_2$ hours for 3D image registration (eleven hours when the correction network is also used). Hence the initial momentum parameterization, which can tolerate large sliding window strides, is essential for fast deformation prediction with high accuracy while guaranteeing diffeomorphic deformations.

Since, I predict the whole image initial momentum in a patch-wise manner, it is natural to extend my approach to a multi-GPU implementation by distributing patches across multiple GPUs. I

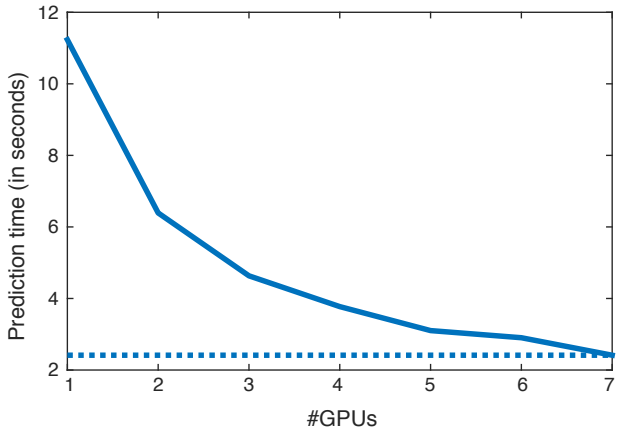


Figure 5.8: Average initial momentum prediction time (in seconds) for a single $229 \times 193 \times 193$ 3D brain image case using various number of GPUs.

assess the runtime of this parallelization strategy on a cluster with multiple Nvidia GTX 1080 GPUs; the initial momentum prediction result is shown in Fig. 5.8. By increasing the number of GPUs, the initial momentum prediction time decreases from 11.23 seconds using 1 GPU to 2.41 seconds using 7 GPUs. However, as the number of GPUs increases, the communication overhead between GPUs becomes larger which explains why computation time is not equal to $(11.23/\text{number of GPUs})$ seconds. Also, when I increase the number of GPUs to 8, the prediction time slightly increases to 2.48s. This can be attributed to the fact that `PyTorch` is still in Beta-stage, and according to the documentation, better performance for large numbers of GPUs (8+) is being actively developed[9]. Hence, I expect faster prediction time using a large number of GPUs in the future. Impressively, by using multiple GPUs, the runtime can be improved by two orders of magnitude over a direct (GPU-based) LDDMM optimization. Thus, my method can readily be used in a GPU-cluster environment for ultra-fast deformation prediction.

[9] `http://pytorch.org/docs/notes/cuda.html/#cuda-nn-dataparallel-instead`

CHAPTER 6: Augmented Deformation Prediction using Deep Learning

This chapter[1] shows the extension of the prediction framework in chapter 5 to augmented deformation prediction problems. Augmented deformation prediction aims to predict deformation parameters that may not be easily obtainable based on the input image appearance. For example, a goal of augmented deformation prediction can be to predict an image registration result which was obtained using images with landmarks or image segmentations, where the both the image and the landmark/segmentation information was used in the image similarity measure. The goal of the augmented prediction algorithm is then to predict the registration results using only the images. I demonstrate how the prediction framework could be easily applied to augmented prediction problems by changing the way the initial momenta for the training data are generated. The goal is to let the network implicitly learn the needed image properties apart from deformation parameter prediction for the augmented prediction task.

In this chapter I discuss two augmented prediction tasks: multi-modal image registration, which my training strategy is successfully applied to, and deformation prediction for image+label matching, where the strategy fails to improve the registration accuracy. I also discuss the reason for the different outcomes for these two tasks, and potential solutions to improve the result.

Sec. 6.1 discusses the strategy for training an augmented deformation prediction network. Sec. 6.2 shows the experimental setting and results for the multi-modal image registration task, and Sec 6.3 discusses the result for the image + label registration task, why the method fails and potential solutions.

[1]The multi-modal image registration experiments discussed in this chapter is based on the paper (Yang et al., 2017b).

6.1 Network and Training strategy

The network structure used in this chapter is the same as that in chapter 5. For the augmented deformation prediction tasks, one straightforward way is to train the network to predict the momenta generated via LDDMM with a modified image loss function. For example, I could create an LDDMM optimization algorithm with normalized cross correlation, mutual information or other advanced multi-modal image similarity measures as the image loss function, and generate initial momenta using this LDDMM algorithm to train the network. The problem with this approach is that the performance of the network is constrained by the LDDMM optimization, as seen in chapter 5, and a manually designed image similarity measure can hinder the registration performance. Thus, a potentially better approach is to generate the initial momenta in a simpler scenario where the additional task, apart from deformation prediction, does not exist, and let the network learn the more complex image similarity measure. For example, for multi-modal image registration prediction, this means generating the initial momenta via uni-modal image registration optimization, and train the network using multi-modal image inputs to predict the uni-modal initial momenta. This has the benefit of eliminating the potential inaccuracy problems from using a hand-crafted image similarity measure, and lets the deep network learn the needed image features implicitly. Of course, this method is based on the assumption that the simplified version of the image data (e.g., uni-modal images for multi-modal image registration) is available. For some cases, e.g., image+label registration prediction, I need to combine both image matching and label matching during the registration optimization phase, thus the proposed method above will not work.

6.2 Experiments for Multi-modal Image Registration

6.2.1 Data and Settings

I assess my approach on the IBIS 3D Autism Brain image dataset (Wolff et al., 2015), containing 375 T1w/T2w brain images ($229 \times 193 \times 193$) of 2 year old subjects. I first register all images affinely to the ICBM 152 atlas (Grabner et al., 2006) and select 359 images for training and the remaining 16 images for testing. For training, I randomly select 359 T1w-T1w image pairs and

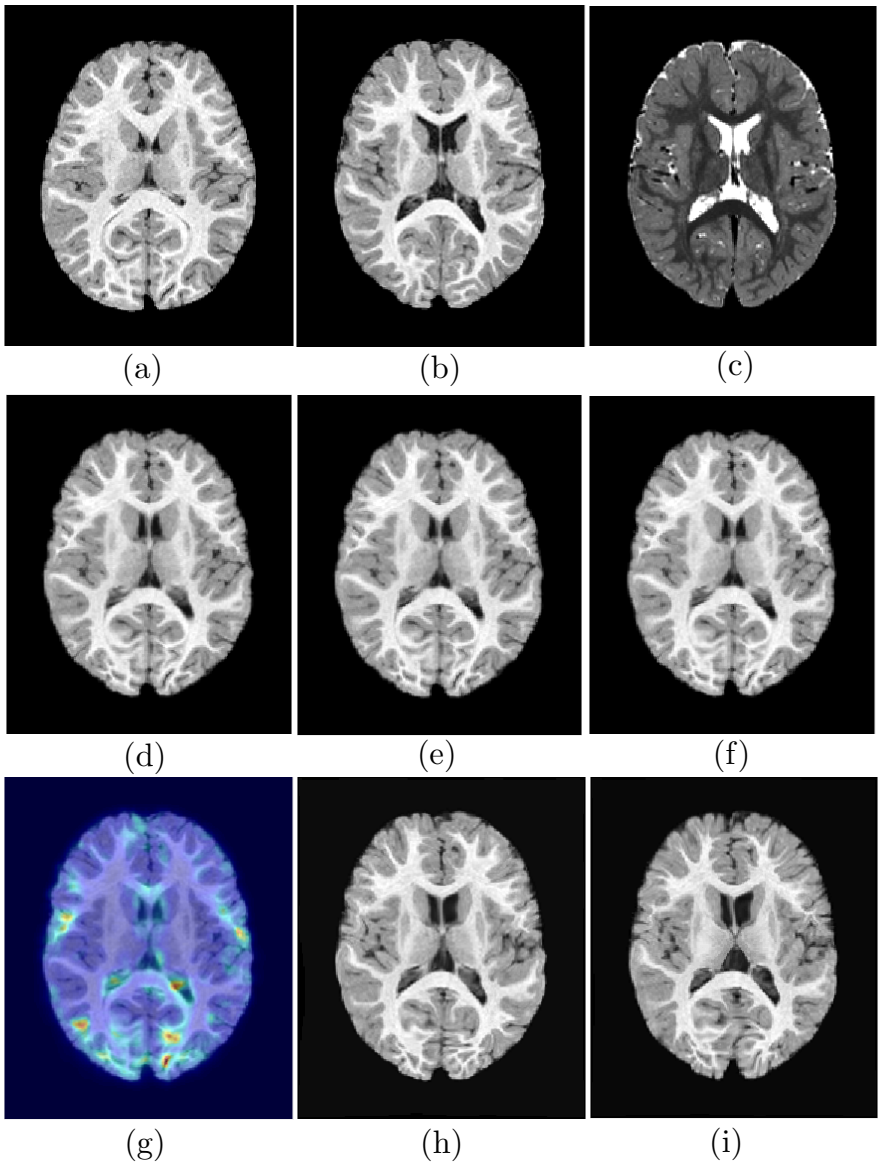


Figure 6.1: Exemplary test case. (a) T1w moving image; (b) T1w target image; (c) T2w target image; (d) deformation result by LDDMM optimization for T1w-T1w registration; (e)-(f) deformation prediction result from T1w-T1w/T1w-T2w data; (g) uncertainty of predicted T1w-T2w deformation as the 2-norm of the sum of variances of deformation fields in all spatial directions, mapped on the predicted T1w-T2w wrapped image. Yellow = more uncertainty, blue = less uncertainty; (h)+(i) `NiftyReg` registration result for T1w-T1w/T1w-T2w pair.

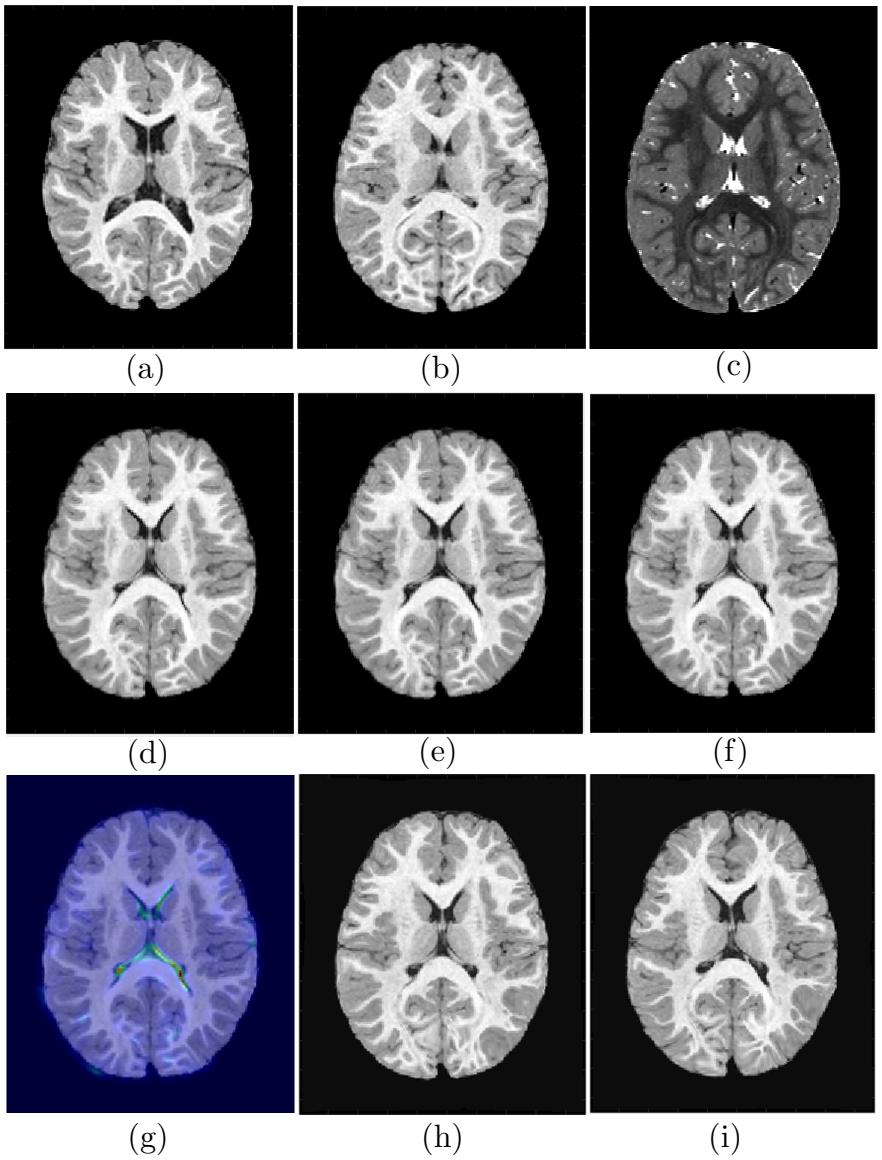


Figure 6.2: Another exemplary test case. Subfigure contents are the same as Fig. 6.1

	Deformation Error w.r.t LDDMM optimization on T1w-T1w data [voxel]						
Data Percentile	0.3%	5%	25%	50%	75%	95%	99.7%
Affine (Baseline)	0.1664	0.46	0.9376	1.4329	2.0952	3.5037	6.2576
Ours, T1w-T1w data	0.0353	0.0951	0.1881	0.2839	0.416	0.714	1.4409
(Yang et al., 2016b), T1w-T2w data	0.0582	0.1568	0.3096	0.4651	0.6737	1.1106	2.0628
Ours, T1w-T2w data	0.0551	0.1484	0.2915	0.4345	0.6243	1.0302	2.0177
Ours, T1w-T2w data, 10 images	0.0663	0.1782	0.3489	0.5208	0.752	1.2421	2.3454
	Prediction/Optimization error between T1w-T2w and T1w-T1w [voxel]						
Data Percentile	0.3%	5%	25%	50%	75%	95%	99.7%
Ours	0.0424	0.1152	0.2292	0.3444	0.4978	0.8277	1.6959
`NiftyReg` (Baseline)	0.2497	0.7463	1.8234	3.1719	5.1124	8.9522	14.4666

Table 6.1: Evaluation results for the 3D dataset. The results of my method are highlighted in green. (Yang et al., 2016b) is the network structure with one single encoder.

perform LDDMM registration using PyCA[2] on the GPU with SSD as image similarity measure. I set the parameters for the LDDMM regularizer $L = a\nabla^2 + b\nabla + c$ to $[a, b, c] = [0.01, 0.01, 0.001]$, and σ in Eqn.5.1 to 0.2. *I then train the network to predict the momenta generated from the T1w-T1w registrations using their corresponding T1w and T2w images*. The network is implemented in `Torch` on a TITAN X GPU and is optimized (over 10 epochs) using `rmsprop` with a learning rate of $= 0.0001$, momentum decay $= 0.1$ and update decay $= 0.01$. For testing, I perform T1w-T2w pairwise registration predictions for all 16 test images, excluding self-registrations. This results in a total of 240 test cases. Each prediction result is compared to the T1w-T1w registrations obtained via LDDMM optimization (used as ground truth). The patch size for the 3D network is $15 \times 15 \times 15$ and I use a sliding window size of 14 for both training and testing. For comparison to the ground truth deformation from LDDMM optimization, I trained another network using T1w-T1w data to perform prediction on the T1w-T1w registration cases. This network serves as the "upper limit" of my multimodal network's potential performance. I also implemented the architecture from (Yang et al., 2016b) and train the network using the T1w-T2w data for comparison. The deformation errors are calculated as the 2-norm of the voxel-wise difference between the predicted deformations and the deformations obtained from LDDMM optimization.

Tab. 6.1(top) lists the evaluation results: my multimodal network (T1w-T2w) greatly reduces deformation error compared to affine registration, and only has a slight accuracy loss compared to

[2] `https://bitbucket.org/scicompanat/pyca`

the T1w-T1w network. This demonstrates registration consistency (to the T1w-T1w registration result) of my approach achieved by *learning* the similarity measure between the two modalities. Moreover, the deformation error data percentiles of Tab. 6.1 show that my network achieves a slight deformation error decrease of 2.1% to 7.3% for all data percentiles compared to (Yang et al., 2016b). This is likely due to less overfitting by using two small encoders.

I also test my network for registration tasks with limited training data. To do so, I randomly choose only 10 out of the 359 training images to perform pairwise registration, generating 90 T1w-T1w registration pairs. I then use these 90 registrations to train my T1w-T2w network model. Tab. 6.1 shows that although the network used only 10 images for training, performance only decreases slightly in comparison to my T1w-T2w network using 359 image pairs for training. Hence, by using patches, my network model can also be successfully trained with a limited number of training images.

To further test my network's consistency in relation to the T1w-T1w prediction results, I calculate the deformation error of my T1w-T2w network w.r.t the T1w-T1w network. For comparison, I also run `NiftyReg` (Modat et al., 2010) B-spline registration on both T1w-T1w and T1w-T2w test cases using normalized mutual information (NMI) with a grid size of 4 and a bending energy weight of 0.0001; I compare the deformation error between T1w-T2w and T1w-T1w registrations, see Tab. 6.1(bottom). Compared to `NiftyReg`, my method is more consistent for multimodal prediction. Fig. 6.1 and Fig. 6.2 shows two test cases: `NiftyReg` generates large differences in the ventricle area between the T1w-T1w and T1w-T2w cases, while my approach does not. I attribute this result to the shortcomings of NMI and not to `NiftyReg` as a registration method. I also computed the 2-norm of the sum of variances of deformation fields in all directions as the uncertainty of the deformation, shown in Fig. 6.1 and Fig. 6.2. I observe high uncertainty around the ventricle, due to the drastic appearance change in this area between the moving and the target image.

Computation time. On average, my method requires 24.46s per case on a Maxwell NVIDIA

TITAN X. Compared to (GPU) LDDMM optimization, I achieve a 36x speedup. Further speedups can be achieved by using multiple GPUs for independent patch predictions.

6.3 Experiments for Image+label Registration

For this experiment, the goal is to predict deformations that give both a good image matching result and a good label matching result. This is important because label matching is, in many cases, the goal of image registration, such as in multi-atlas image segmentation (Iglesias and Sabuncu, 2015). For labels in areas with high intra-subject variations and homogeneous image intensity, registration methods with only intensity-based matching criterion might result in label misalignment.

Specifically, my goal is to predict a momentum matching both the image and labels of the moving and target image using image appearance only. In this setting, I know the label maps for the training data, but not for the testing data. When training a model for this problem, the input data is the moving and target image pair and the prediction output is the initial momentum for LDDMM. When generating the training data, I change the original image similarity term in the LDDMM energy function 5.2, which is simply the sum of the square of the image intensity differences $\frac{1}{\sigma^2}||M \circ \Phi^{-1}(1) - T||^2$, to

$$\frac{1}{\sigma^2_{\text{image}}}||M \circ \Phi^{-1}(1) - T||^2 + \frac{1}{\sigma^2_{\text{label}}} \sum_{i=1}^{N_L} ||L_{M,i} \circ \Phi^{-1}(1) - L_{T,i}||^2 \tag{6.1}$$

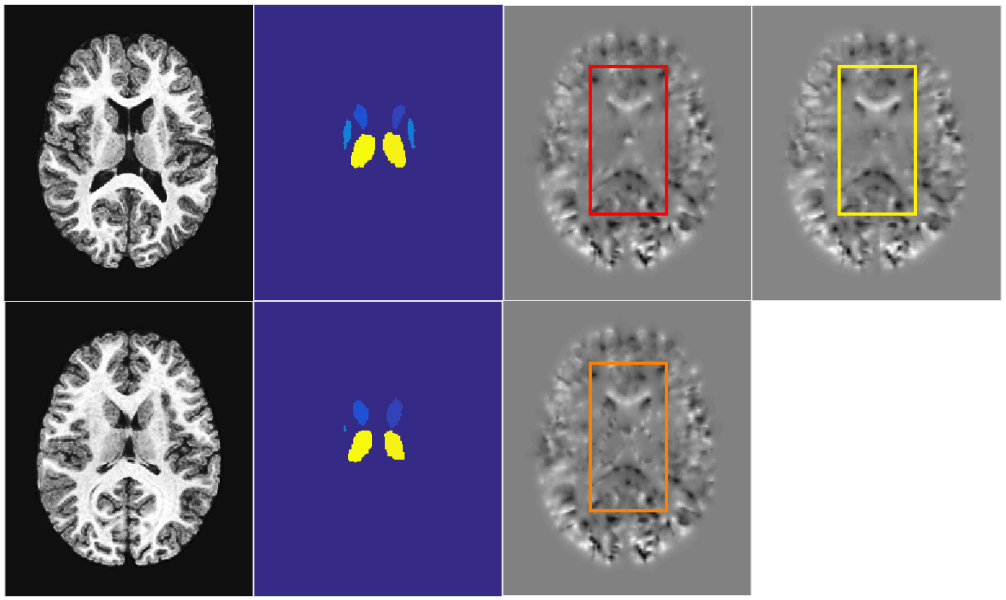
where M and T are the moving and target images, L_M and L_T are their corresponding label maps, N_l indicates the total number of label types for the label map, and i indicates the ith label type. The goal is to use LDDMM optimization to generate initial momenta that match the image and the labels at the same time, resulting in a deformation that has good label overlap and image appearance matching. Similar to Sec. 6.2, I use the moving and target image pairs as network input and predict the generated momenta. In my experimental setting, I use 16 images with labels for subcortical areas from the IBIS 3D brain image dataset (Wolff et al., 2015) and perform pairwise registrations, generating 240 training cases. For the LDDMM setting, the regularization kernel setting is the same

as in Sec. 6.2, and σ_{image} and σ_{label} are set to 0.24 and 0.34. The network learning parameter is also the same as in Sec. 6.2.

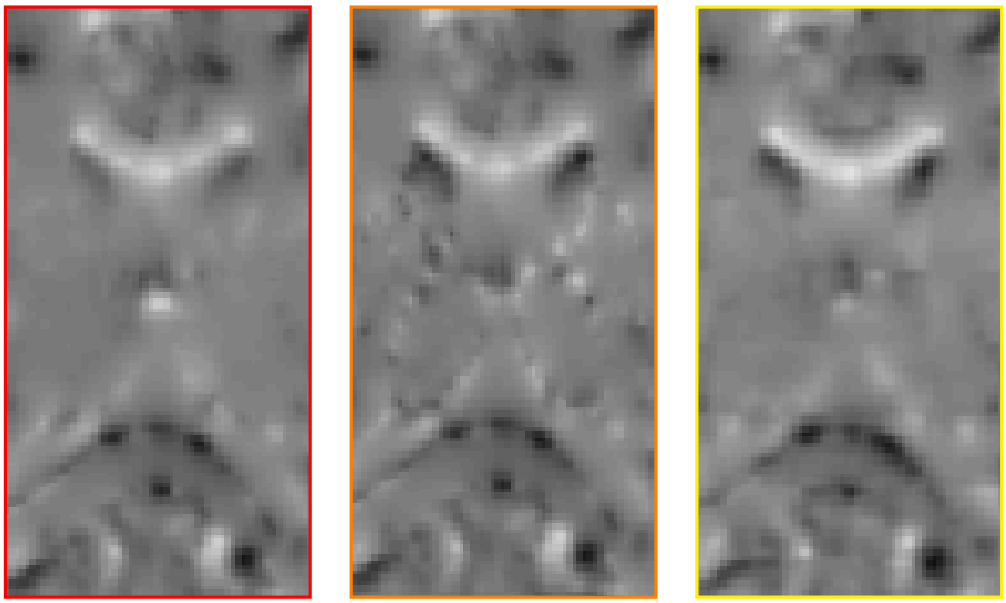
Unfortunately, for this task my network is unable to predict the initial momentum around the boundary of the label map, which is essential for label alignment. See Fig 6.3 as an example. Notice the small initial momenta near the boundary of the label area for the image + label matching LDDMM optimization version (middle image in (b)). These initial momenta are essential for achieving a high label overlapping value, but the deep network fails to predict them. I made several modifications to the network to attempt to improve the result without success. These attempts include:

1. Adding spatial information to the network. Examples include two networks where the first one takes the image patches and the location of the center voxel of the patchs in the images as input, and the second one takes the patch from the identity deformation map with the image patches as the input.
2. Using a larger patch size. For this network I increased the patch size from $15 \times 15 \times 15$ to $31 \times 31 \times 31$.
3. Using a deeper network structure. I created a residual-network style encoder-decoder network and doubled the number of convolution layers.

Several observations can be made from these attempts. First, adding spatial information does not improve the result. This is reasonable because spatial information can only indicate which areas potentially contain the initial momentum for label matching, and in the end the initial momentum prediction still depends on the image patch appearance. Failing to improve prediction using a deeper network structure means that the network capacity is possibly not the bottleneck of this task. In the end, I think the reason why momentum prediction fails is the low contrast for the subcortical areas. Compared to areas where the momentum prediction is more accurate (e.g. ventrical areas, curvatures), the intensity difference between the label area (subcortical region) and the neighboring white matter is small. Furthermore, the scale and range of the label matching momenta are much



(a)



(b)

Figure 6.3: Example case for the image+label matching task. (a): example data. From left to right: 1st column: moving (top) and target (bottom) images. 2nd column: moving(top) and (target) label map. 3rd column: initial momentum generated from LDDMM optimization by matching image only (top) and matching image+label (bottom). 4th column: predicted initial momentum from deep network. (b): Zoomed in area of the initial momentums for optimization via matching images only (left), optimization via matching image+label (middle) and network prediction (right). Notice that there are small momentum alongside the boundary of the labels of the moving image in the middle figure (via matching image+label), but not in the left (via matching image only) and the right (via deep network prediction) images.

smaller than other large deformation momentums, which can be another reason why predicting these label matching momenta fails.

There are several possible solutions for the problems mentioned above:

1. Adding image contrast enhancement as a preprocessing step. The intuition behind this is simple: the contrast for the labeled image area is low, so increasing the contrast may increase the prediction accuracy. However, this either means full-image contrast enhancement which may not be desired, or requires information of the label area location, which is by default not available during testing.
2. Combining image registration prediction with image segmentation. For this setting, I can create a deep network to segment the label area on the image, and use the segmentation map as additional input to the registration prediction network. Or, I can create a multi-task network where the input is the image patches, and the network outputs both the segmentation map and the initial momentum prediction. This structure is similar to the Mask R-CNN network (He et al., 2017), which is an object instance segmentation framework that has 3 output network branches (bounding box regression, classification, object segmentation map) and achieves state-of-the-art semantic segmentation results. The potential problem for this approach is that segmenting the brain areas can be difficult especially when ambiguous boundaries exist, which is the case for subcortical areas.
3. Reducing the effect of pooling/unpooling operations. Pooling operations are used in deep learning to reduce the feature size from the lower level layers and to help extract high-level features. Besides, pooling operations also introduces shift-invariance for the features, which is beneficial for classification tasks and non-pixel-to-pixel regression problems. However, for my problem where each voxel has an initial momentum vector, pooling reduces the feature detail at higher level layers, which can be harmful when predicting initial momenta for small anatomical structures. Some methods that use a similar encoder-decoder structure mitigate this problem by using skip connections. Skip connections concatenate the features from the encoder layers to the decoder layers using

the same feature size, thus avoiding pooling/unpooling operations between these layers. Examples include `U-net` (Ronneberger et al., 2015) and `V-net` (Milletari et al., 2016b) for medical image segmentation, as well as `pix2pix` (Isola et al., 2016) for pixel-wise image-to-image translation. For my task, a network where every layer in the encoder is connected to the corresponding decoder layer can potentially improve the prediction result. This can be easily done for networks with a single encoder (e.g. the network in (Yang et al., 2016b)), while for a network with multiple encoders as in (Yang et al., 2017a) it is less clear how skip connections can be applied.

These 3 approaches all have potential to increase the label matching prediction accuracy, Of course there are other possible ways to improve the prediction result, such as using more image information, e.g., using both T1 weighted and T2 weighted images as input. All these methods can be explored in future research.

CHAPTER 7: Discussion

This chapter summarizes the contributions of this dissertation in Sec. 7.1, followed by a discussion of future work in Sec 7.2.

7.1 Summary of Contributions

This dissertation studies three problems in medical image registration: uncertainty quantification for image registration, quasi-normal image synthesis for pathological image registration, and fast predictive image registration. I propose a method for each one of the problems stated above, and the contributions can be restated with discussions on how they are accomplished as follows:

1. *For uncertainty quantification for image registration: I created a Laplace approximation based method to model the local multivariate Gaussian distribution at the optimal solution for the LDDMM method, and an efficient approach to accurately approximate the covariance matrix of the approximated Gaussian distribution as the uncertainty measure.* In Chapter 2.2.3.4, the LDDMM optimization energy is revisited as the negative logarithm of the posterior distribution of the registration parameters, and I apply a Laplace approximation to this posterior distribution for variational approximation. The Laplace approximation is formulated as a multi-variate Gaussian distribution, and the covariance matrix, which is the inverse of the Hessian of the LDDMM energy, can be used for uncertainty quantification. Calculating and inverting the Hessian is challenging in terms of both computation time and storage, due to the high-dimensional parameterization of LDDMM. To solve these tasks, I proposed two key solutions. First, I calculate Hessian-vector multiplications to avoid computing and storing the full Hessian. Second, I separate the LDDMM Hessian into a regularization Hessian, whose inverse has a closed-form solution, and an image mismatch Hessian, which I approximate using a low-rank eigen-decomposition.

Using the Woodbury identity on these two components, I get a low-rank Hessian inverse approximation method that is computationally more efficient than the finite-difference method, and achieves much higher accuracy compared with directly inverting the full LDDMM energy Hessian with low-rank eigenmodes. The variance of the covariance matrix for the initial velocity is used for uncertainty quantification, and indicates high uncertainty for isotropic areas and areas with ambiguous deformations.

2. *For quasi-normal image synthesis: I propose a variational deep learning approach to synthesize a quasi-normal image from an image with a large pathology which only requires pathology segmentation at training time. Furthermore, image synthesis uncertainty from the network is used to guide image registration for better registration accuracy.*

   The method for training such a deep network for quasi-normal image synthesis is discussed in Chapter 3.4.3. Specifically, I use a denoising variational autoencoder, and treat the pathologies as noise. The goal then is to remove the pathological noise, which is consistent with the theory for a denoising autoencoder. To train the network with tumor images, I use two techniques. First, for the real tumor areas, I apply *loss function masking* which sets the mismatch between the tumor areas of the synthesized image and the ground truth image to 0. Second, I add fake tumors to the image and train the network to recover the original brain structures replaced by the fake tumors. After the synthetic image is generated, I utilize the variational autoencoding theory and perform Monte-Carlo sampling to obtain the image reconstruction uncertainty. I demonstrate improved registration accuracy by using the uncertainty information to guide image registration.

3. *For fast image registration: I build a general deep network architecture to learn the relationship between deformation parameters and image appearance and to efficiently predict deformation parameters for both uni-modal and multi-modal image registration applications.*

   I discuss the deep network architecture for fast deformation prediction from image appearance in Chapter 5 and Chapter 6. In particular, I use a deep encoder-decoder network

structure, and perform the deformation prediction in a patch-wise manner for 3D image registration prediction. I focus on LDDMM registration prediction, and use the initial momentum as the parameterization to ensure diffeomorphisms from patch-wise prediction. I also provide a probabilistic version of the network which can be sampled to calculate uncertainties in the predicted deformations. Finally, I introduce a correction method to further increase the registration accuracy for the existing prediction network. This method uses another deep network to predict the difference between the predicted deformation parameters and the deformation parameters from LDDMM optimization. The inputs for this correction network are the moving image and the warped back target image; At testing time, I sum up the outputs from both the prediction network and the correction network as the final prediction result. The method is tested on uni-modal atlas-to-image registration as well as uni- and multi- modal image-to-image registration. I also explored other augmented deformation prediction problems and discussed potential solutions to increase prediction accuracy for these problems.

These contributions support the thesis statement in Chapter 1, which I revisit here:

Registration uncertainty can be modeled using a variational approximation that exploits the low-rank property of the image similarity measure in a time- and memory-efficient manner. Furthermore, deep learning can be used for pathological-to-quasi-normal image synthesis and for fast and accurate prediction of uni-modal and multi-modal registrations.

7.2 Future work

There are several directions where my work can be explored in the future. I discuss them in the following sections.

7.2.1 Uncertainty Quantification for Image Registration

One potential idea is to apply the Laplace approximation method to other medical image analysis tasks, such as image regression and atlas building. This works because, for many models that solve these problems, the energy functions of the latent parameters consist of a regularization

part and a matching part, which can be regarded as the prior and the likelihood in the Bayesian theory. Also, many of these energy functions can be approximated in quadratic forms. Hence, computing the Laplace approximation for these models amounts to computing the Hessian of the energy and its inverse. By utilizing the second variation technique discussed in Chapter 2.2.3.4, this can be done straightforwardly. However, there are unsolved problems for these tasks that are worth investigating:

1. *Handling the Hessian matrix with negative eigenmodes.* For the Hessian of the energy to be meaningful, a prerequisite is to optimize the energy function to convergence. Doing so ensures that the Hessian of the energy is positive definite or semi-positive definite (such as in the regression problems on the Grassmannian manifold (Hong et al., 2017)). This is relatively straightforward for simple problems, but for complex non-convex optimization tasks it can take a very long time to get the energy function to convergence. Thus, apart from using advanced optimization techniques to speed up the convergence, one interesting question arises for the scenarios where the optimization is close to convergence with an indefinite Hessian. In this case, how should the Hessian be fixed so that the useful information can be extracted for uncertainty quantification? This is left for future research.
2. *Propagation of the uncertainty*. This can be further expanded into two problems. First, for regression problems, it is useful to calculate uncertainty not only on the initial condition, but along the geodesics. (Hong et al., 2017) proposed a way to do so on the Grassmannian, but the experiment of that work only focuses on shape regression, where the data size as well as the size of the Hessian are relatively small. This makes calculating and inverting the full Hessian an easy task and performing covariance propagation via direct matrix multiplication. For problems with large data size such as image regression, such an approach will no longer be computationally possible due to the computation constraints. Thus some ways to approximate uncertainty propagation for large data regression will be useful. Second, my proposed method performs uncertainty quantification by calculating the variance of the registration parameters, which is the initial velocity in Chapter 2.2.3.4.

It would be more interesting to calculate the uncertainty on the resulting deformations. It is worth noting that for my deep learning based deformation prediction task, I am able to quantify variance of the deformation from the deep learning model uncertainty. However, it is still useful to explore uncertainty quantification on the LDDMM registration model itself. Again, an efficient computation method for large data uncertainty propagation is needed for this task.

7.2.2 Pathological Image Reconstruction

A very interesting question is how to extend my quasi-normal image synthesis network to 3D images. This is a difficult task because 3D medical images are very large, and using the whole image as the network input is not feasible. In the initial experiments, we implemented a 2.5D network which reconstructs 14 slices at once. Training the network on 500 2.5D training cases takes 3 days, which, while not fast, is feasible. One possible approach is to learn mappings for 3D patches using the patch location as an additional feature, which would enable us to train on a much larger dataset (patches) at a reasonable computational cost. The maximal 3D patch size is decided by the memory capacity of the GPU and the network structure. For the deep learning deformation framework in Chapter 5, the 3D patch size is $15 \times 15 \times 15$, but there are works that, compared with Chapter 5, use smaller networks with much larger input patch sizes (e.g. $128 \times 128 \times 64$ on a Tesla K40 GPU for image segmentation in (Milletari et al., 2016b).)

Also, designing a more "lesion-like" noise model is an interesting direction. In my work I choose uniform intensity shapes as the fake lesion, and argue that the reason why using real lesions in the fake lesion step does not improve network performance is due to the limited number of lesion samples in the training dataset. Thus, using a larger dataset with a larger and more diverse set of lesion appearances to improve the synthesis result would be interesting to explore.

Finally, apart from using variational autoencoder for image synthesis, one can explore generative adversarial networks (GANs) for our image synthesis task. GANs are already used for various image synthesis and image modality transfer problems (Isola et al., 2016; Zhu et al., 2017; Nie et al., 2016a). Moreover, researchers have shown that GANs can be used for image impainting (Yeh et al.,

2016), which sheds light on using GANs for quasi-normal image registration by, again, treating the pathologies as random noise. This has potential to generate better pathological area reconstruction, and is left for future research.

7.2.3 Deep Learning for Deformation Prediction

My deformation prediction framework is very general and can be directly applied to many other registration techniques. For non-parametric registration methods with pixel/voxel wise registration parameters (e.g., elastic registration (Modersitzki, 2004), or stationary velocity field (Vercauteren et al., 2009) registration approaches), our approach can be directly applied for parameter prediction. For parametric registration methods with local control, such as B-splines, we could attach fully connected layers to the decoder to reduce the network output dimension, thereby predicting low-dimensional registration parameters for a patch. Of course, the patch pruning techniques may not be applicable for these methods if the parameter locality cannot be guaranteed.

Furthermore, this framework opens up possibilities for various extensions and applications. Exciting possibilities are, for example, using the framework for fast multi-atlas segmentation, fast image geodesic regression, fast atlas construction, or fast user-interactive registration refinements (where only a few patches need to be updated based on local changes). Furthermore, extending the deformation prediction network to more complex registration tasks could also be beneficial, as discussed in Chapter 6. Other potential areas include joint image-label registration for better label-matching accuracy; multi-scale-patch networks for very large deformation prediction; deformation prediction for registration models with anisotropic regularizations; and end-to-end optical flow prediction via initial momentum parameterization. For the correction step, other methods could also be explored, either by using different network structures, or by recursively updating the deformation parameter prediction using the correction approach (e.g., with a sequence of correction networks where each network corrects the momenta predicted from the previous one). Finally, since our uncertainty quantification approach indicates high uncertainty for areas with large deformation or appearance changes, utilizing the uncertainty map to detect pathological areas could also be an interesting research direction.

BIBLIOGRAPHY

Alain, G. and Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. *J. Mach. Learn. Res.*, 15(1):3563–3593.

Andermatt, S., Pezold, S., and Cattin, P. (2016). Multi-dimensional gated recurrent units for the segmentation of biomedical 3d-data. In *DLMIA/MICCAI*, pages 142–151.

Antony, J., McGuinness, K., O'Connor, N. E., and Moran, K. (2016). Quantifying radiographic knee osteoarthritis severity using deep convolutional neural networks. In *ICPR*, pages 1195–1200.

Ardekani, B. A., Guckemus, S., Bachman, A., Hoptman, M. J., Wojtaszek, M., and Nierenberg, J. (2005). Quantitative comparison of algorithms for inter-subject registration of 3D volumetric brain MRI scans. *Journal of Neuroscience Methods*, 142(1):67 – 76.

Ashburner, J. (2007). A fast diffeomorphic image registration algorithm. *NeuroImage*, 38(1):95 – 113.

Avants, B., Epstein, C., Grossman, M., and Gee, J. (2008). Symmetric diffeomorphic image registration with cross-correlation: Evaluating automated labeling of elderly and neurodegenerative brain. *MedIA*, 12(1):26 – 41. Special Issue on The Third International Workshop on Biomedical Image Registration WBIR 2006.

Azevedo-filho, A. (1994). Laplace's method approximations for probabilistic inference in belief networks with continuous variables. In *In de Mantaras*, pages 28–36. Morgan Kaufmann.

Ballard, D. H. (1987). Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, pages 279–284. AAAI Press.

Beg, M. F., Miller, M. I., Trouvé, A., and Younes, L. (2005). Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International journal of computer vision*, 61(2):139–157.

Brett, M., Leff, A. P., Rorden, C., and Ashburner, J. (2001). Spatial normalization of brain images with focal lesions using cost function masking. *Neuroimage*, 14(2):486–500.

Bro-Nielsen, M. and Gramkow, C. (1996). Fast fluid registration of medical images. In Höhne, K. H. and Kikinis, R., editors, *Visualization in Biomedical Computing: 4th International Conference, VBC'96 Hamburg, Germamy, September 22–25, 1996 Proceedings*, pages 265–276.

Broit, C. (1981). *Optimal Registration of Deformed Images*. PhD thesis, Philadelphia, PA, USA. AAI8207933.

Brosch, T. and Tam, R. C. (2013). Manifold learning of brain mris by deep learning. *MICCAI*, 16 Pt 2:633–40.

Bruveris, M., Risser, L., and Vialard, F.-X. (2012). Mixture of kernels and iterated semidirect product of diffeomorphisms groups. *Multiscale Modeling & Simulation*, 10(4):1344–1368.

Cabezas, M., Oliver, A., Llad, X., Freixenet, J., and Cuadra, M. B. (2011). A review of atlas-based segmentation for magnetic resonance brain images. *Computer Methods and Programs in Biomedicine*, 104(3):e158 – e177.

Cao, T., Singh, N., Jojic, V., and Niethammer, M. (2015). Semi-coupled dictionary learning for deformation prediction. In *ISBI*, pages 691–694.

Cao, T., Zach, C., Modla, S., Powell, D., Czymmek, K., and Niethammer, M. (2014). Multi-modal registration for correlative microscopy using image analogies. *Medical Image Analysis*, 18(6):914 – 926.

Caplan, J., Niethammer, M., II, R. M. T., and Czymmek, K. J. (2011). The power of correlative microscopy: multi-modal, multi-scale, multi-dimensional. *Current Opinion in Structural Biology*.

Cheng, X. and Zheng, L. Z. Y. (2015). Deep similarity learning for multimodal medical images. *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.*, pages 1–5.

Chitphakdithai, N. and Duncan, J. S. (2010). Non-rigid registration with missing correspondences in preoperative and postresection brain images. In *MICCAI*, pages 367–374, Berlin, Heidelberg. Springer.

Chou, C.-R., Frederick, B., Mageras, G., Chang, S., and Pizer, S. (2013). 2D/3D image registration using regression learning. *CVIU*, 117(9):1095–1106.

de Vos, B. D., Wolterink, J. M., de Jong, P. A., Viergever, M. A., and Igum, I. (2016). 2d image classification for 3d anatomy localization: employing deep convolutional neural networks.

Dosovitskiy, A., Fischery, P., Ilg, E., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766.

Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., and Pal, C. (2016). The importance of skip connections in biomedical image segmentation. In *DLMIA/MICCAI*, pages 179–187.

Dupuis, P., Grenander, U., and Miller, M. I. (1998). Variational problems on flows of diffeomorphisms for image matching. *Quarterly of applied mathematics*, pages 587–600.

Evans, A. C. (2006). The nih mri study of normal brain development. *NeuroImage*, 30(1):184 – 202.

Fischer, B. and Modersitzki, J. (2004). A unified approach to fast image registration and a new curvature based registration technique. *Linear Algebra and its Applications*, 380:107 – 124.

Fitzpatrick, J. M. and West, J. B. (2001). The distribution of target registration error in rigid-body point-based registration. *IEEE Transactions on Medical Imaging*, 20(9):917–927.

Flath, H. P., Wilcox, L. C., Akçelik, V., Hill, J., van Bloemen Waanders, B., and Ghattas, O. (2011). Fast algorithms for Bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial Hessian approximations. *SIAM Journal on Scientific Computing*, 33(1):407–432.

Fonov, V., Evans, A. C., Botteron, K., Almli, C. R., McKinstry, R. C., and Collins, D. L. (2011). Unbiased average age-appropriate atlases for pediatric studies. *Neuroimage*, 54(1):313–327.

Fripp, J., Crozier, S., Warfield, S. K., and Ourselin, S. (2007). Automatic segmentation of articular cartilage in magnetic resonance images of the knee. In *Proceedings of the 10th international conference on Medical image computing and computer-assisted intervention*, MICCAI'07, pages 186–194, Berlin, Heidelberg. Springer-Verlag.

Gal, Y. and Ghahramani, Z. (2015). Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv:1506.02158*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *ICML*.

Gao, X., Lin, S., and Wong, T. Y. (2015). Automatic feature learning to grade nuclear cataracts based on deep learning. *IEEE Transactions on Biomedical Engineering*, 62(11):2693–2701.

Ghesu, F. C., Georgescu, B., Zheng, Y., Hornegger, J., and Comaniciu, D. (2015). Marginal space deep learning: Efficient architecture for detection in volumetric image data. In *MICCAI*, pages 710–718.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gooya, A., Pohl, K. M., Bilello, M., Cirillo, L., Biros, G., Melhem, E. R., and Davatzikos, C. (2012). GLISTR: Glioma image segmentation and registration. *TMI*, 31(10):1941–1954.

Grabner, G., Janke, A. L., Budge, M. M., Smith, D., Pruessner, J., and Collins, D. L. (2006). Symmetric atlasing and model based segmentation: An application to the hippocampus in older adults. In *MICCAI*, pages 58–66.

Guetter, C., Xu, C., Sauer, F., and Hornegger, J. (2005). Learning based non-rigid multi-modal image registration using Kullback-Leibler divergence. In *MICCAI*.

Gutiérrez-Becker, B., Mateus, D., Peter, L., and Navab, N. (2016). Learning optimization updates for multimodal registration. In *MICCAI*, pages 19–27.

Gutierrez-Becker, B., Mateus, D., Peter, L., and Navab, N. (2017). Guiding multimodal registration with learned optimization updates.

Han, X., Yang, X., Aylward, S., Kwitt, R., and Niethammer, M. (2017). Efficient registration of pathological images: A joint pca/image-reconstruction approach. In *ISBI*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. *arXiv preprint arXiv:1703.06870*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, Washington, DC, USA.

Hermosillo, G., Chefd'Hotel, C., and Faugeras, O. (2002). Variational methods for multimodal image matching. *International Journal of Computer Vision*, 50(3):329–343.

Hill, D. L., Batchelor, P. G., Holden, M., and Hawkes, D. J. (2001). Medical image registration. *Physics in medicine and biology*, 46(3):R1.

Hong, Y., Yang, X., Kwitt, R., Styner, M., and Niethammer, M. (2017). Regression Uncertainty on the Grassmannian. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pages 785–793, Fort Lauderdale, FL, USA. PMLR.

Horn, B. K. P. and Schunck, B. G. (1993). "determining optical flow": A retrospective. *Artif. Intell.*, 59(1-2):81–87.

Hosseini-Asl, E., Gimel'farb, G. L., and El-Baz, A. (2016). Alzheimer's disease diagnostics by a deeply supervised adaptable 3d convolutional network. *CoRR*, abs/1607.00556.

Iglesias, J. E. and Sabuncu, M. R. (2015). Multi-atlas segmentation of biomedical images: A survey. *Medical Image Analysis*, 24(1):205 – 219.

Im, D. J., Ahn, S., Memisevic, R., and Bengio, Y. (2015). Denoising criterion for variational auto-encoding framework. *CoRR*, abs/1511.06406.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*.

Jog, A., Roy, S., Carass, A., and Prince, J. L. (2013). Magnetic resonance image synthesis through patch regression. In *ISBI*.

Joshi, S., Davis, B., Jomier, M., and Gerig, G. (2004). Unbiased diffeomorphic atlas construction for computational anatomy. *NeuroImage*, 23:151–160.

Kalmikov, A. G. and Heimbach, P. (2014). A hessian-based method for uncertainty quantification in global ocean state estimation. *SIAM Journal on Scientific Computing*, 36(5):S267–S295.

Kamper, H., Wang, W., and Livescu, K. (2016). Deep convolutional acoustic word embeddings using word-pair side information. In *ICASSP*, pages 4950–4954.

Karpathy, A., Abbeel, P., Brockman, G., Chen, P., Cheung, V., Duan, R., Goodfellow, I., Kingma, D., Ho, J., Houthooft, R., Salimans, T., Schulman, J., Sutskever, I., and Zaremba., W. (2016). *Generative Models*. `https://blog.openai.com/generative-models/`.

Kawahara, J., BenTaieb, A., and Hamarneh, G. (2016). Deep features to classify skin lesions. In *ISBI*, pages 1397–1400.

Kawahara, J., Brown, C. J., Miller, S. P., Booth, B. G., Chau, V., Grunau, R. E., Zwicker, J. G., and Hamarneh, G. (2017). Brainnetcnn: Convolutional neural networks for brain networks; towards predicting neurodevelopment. *NeuroImage*, 146:1038 – 1049.

Kim, H. and Hwang, S. (2016). Scale-invariant feature learning using deconvolutional neural networks for weakly-supervised semantic segmentation. *CoRR*, abs/1602.04984.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. In *ICLR*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. *CoRR*, abs/1312.6114.

Klein, A., Andersson, J., Ardekani, B. A., Ashburner, J., Avants, B., Chiang, M.-C., Christensen, G. E., Collins, D. L., Gee, J., Hellier, P., Song, J. H., Jenkinson, M., Lepage, C., Rueckert, D., Thompson, P., Vercauteren, T., Woods, R. P., Mann, J. J., and Parsey, R. V. (2009). Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration. *NeuroImage*, 46(3):786 – 802.

Korez, R., Likar, B., Pernuš, F., and Vrtovec, T. (2016). Model-based segmentation of vertebral bodies from mr images with 3d cnns. In *MICCAI*, pages 433–441.

Kybic, J. (2010). Bootstrap resampling for image registration uncertainty estimation without ground truth. *Image Processing, IEEE Transactions on*, 19(1):64–73.

Kyriacou, S. K., Davatzikos, C., Zinreich, S. J., and Bryan, R. N. (1999). Nonlinear elastic registration of brain images with tumor pathology using a biomechanical model. *IEEE Transactions on Medical Imaging*, 18(7):580(13).

Lanczos, C. (1950). An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bureau Standards*, 45(4):255–282.

Larsen, R. M. (1998). Lanczos bidiagonalization with partial reorthogonalization.

Lecun, Y. (1987). *PhD thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models)*. Universite P. et M. Curie (Paris 6).

Lee, D., Hofmann, M., Steinke, F., Altun, Y., Cahill, N., and Schölkopf, B. (2009). Learning similarity measure for multi-modal 3D image registration. In *CVPR*.

Liu, X., Niethammer, M., Kwitt, R., Singh, N., McCormick, M., and Aylward, S. (2015). Low-rank atlas image analyses in the presence of pathologies. *TMI*, 34(12):2583–2591.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440.

Lorenzen, P., Prastawa, M., Davis, B., Gerig, G., Bullitt, E., and Joshi, S. (2006). Multi-modal image set registration and atlas formation. *MedIA*, 10(3):440–451.

Marcus, D. S., Wang, T. H., Parker, J., Csernansky, J. G., Morris, J. C., and Buckner, R. L. (2007). Open access series of imaging studies (OASIS): Cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. *J. Cognitive Neurosci.*, 19(9):1498–1507.

Menegola, A., Fornaciali, M., Pires, R., de Avila, S. E. F., and Valle, E. (2016). Towards automated melanoma screening: Exploring transfer learning schemes. *CoRR*, abs/1609.01228.

Menze, B. H., Reyes, M., and Leemput, K. V. (2015). The multimodal brain tumor image segmentation benchmark (BRATS). *TMI*, 34(10):1993–2024.

Meyer, C., Boes, J., Kim, B., and Bland, P. (1998). Evaluation of control point selection in automatic, mutual information driven, 3D warping. In *MICCAI*.

Miao, S., Wang, Z. J., and Liao, R. (2016). A cnn regression approach for real-time 2d/3d registration. *IEEE Transactions on Medical Imaging*, 35(5):1352–1363.

Michel, F., Bronstein, M., Bronstein, A. M., and Paragios, N. (2011). Boosted metric learning for 3D multi-modal deformable registration. In *ISBI*.

Milletari, F., Navab, N., and Ahmadi, S. (2016a). V-net: Fully convolutional neural networks for volumetric medical image segmentation. *CoRR*, abs/1606.04797.

Milletari, F., Navab, N., and Ahmadi, S.-A. (2016b). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 565–571. IEEE.

Modat, M., Ridgway, G., Taylor, Z., Lehmann, M., Barnes, J., Hawkes, D., Fox, N., and Ourselin, S. (2010). Fast free-form deformation using graphics processing units. *Comput. Methods Prog. Biomed*, 98(3):278–284.

Modersitzki, J. (2004). *Numerical methods for image registration*. Oxford University Press on Demand.

Nie, D., Trullo, R., Petitjean, C., Ruan, S., and Shen, D. (2016a). Medical image synthesis with context-aware generative adversarial networks. *arXiv preprint arXiv:1612.05362*.

Nie, D., Zhang, H., Adeli, E., Liu, L., and Shen, D. (2016b). Miccai. pages 212–220.

Niethammer, M., Hart, G. L., Pace, D. F., Vespa, P. M., Irimia, A., Horn, J. D., and Aylward, S. R. (2011). Geometric metamorphosis. In *MICCAI*, pages 639–646, Berlin, Heidelberg. Springer.

Oliveira, F. P. and Tavares, J. M. R. (2014). Medical image registration: a review. *Computer methods in biomechanics and biomedical engineering*, 17(2):73–93.

Ou, Y., Akbari, H., Bilello, M., Da, X., and Davatzikos, C. (2014). Comparative evaluation of registration algorithms in different brain databases with varying difficulty: Results and insights. *TMI*, 33(10):2039–2065.

Ou, Y., Sotiras, A., Paragios, N., and Davatzikos, C. (2011). DRAMMS: Deformable registration via attribute matching and mutual-saliency weighting. *Med. Image Anal.*, 15(4):622–639.

Payan, A. and Montana, G. (2015). Predicting alzheimer's disease: a neuroimaging study with 3d convolutional neural networks. *CoRR*, abs/1502.02506.

Payer, C., Štern, D., Bischof, H., and Urschler, M. (2016). Regressing heatmaps for multiple landmark localization using cnns. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II*, pages 230–238.

Plis, S. M., Hjelm, D. R., Salakhutdinov, R., Allen, E. A., Bockholt, H. J., Long, J. D., Johnson, H. J., Paulsen, J. S., Turner, J. A., and Calhoun, V. D. (2014). Deep learning for neuroimaging: a validation study. *Frontiers in Neuroscience*, 8:229.

Radau, P., Lu, Y., Connelly, K., Paul, G., Dick, A., and Wright, G. (2009). Evaluation framework for algorithms segmenting short axis cardiac MRI. *The MIDAS Journal*, 49.

Reuter, M., Rosas, D. H., and Fischl, B. (2010). Highly accurate inverse consistent registration: a robust approach. *Neuroimage*, 53(4):1181–1196.

Risholm, P., Janoos, F., Norton, I., Golby, A. J., and Wells, W. M. (2013). Bayesian characterization of uncertainty in intra-subject non-rigid registration. *Medical image analysis*, 17(5):538–555.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241.

Rosenman, J. G., Miller, E. P., Tracton, G., and Cullip, T. J. (1998). Image registration: An essential part of radiation therapy treatment planning. *Int J Radiat Oncol Biol Phys*, 40:197–205.

Roth, H. R., Lu, L., Liu, J., Yao, J., Seff, A., Cherry, K., Kim, L., and Summers, R. M. (2016). Improving computer-aided detection using convolutional neural networks and random view aggregation. *IEEE Transactions on Medical Imaging*, 35(5):1170–1181.

Roy, S., Carass, A., Jog, A., Prince, J. L., and Lee, J. (2014). MR to CT registration of brains using image synthesis. *SPIE Medical Imaging*, 9034:903419–903419–8.

Roy, S., Carass, A., and Prince, J. (2013). Magnetic resonance image example based contrast synthesis. *TMI*, 32(12):2348–2363.

Roy, S., Carass, A., Shiee, N., Pham, D. L., and Prince, J. L. (2010). MR contrast synthesis for lesion segmentation. In *ISBI*, pages 932–935.

Rueckert, D., Sonoda, L. I., Hayes, C., Hill, D. L., Leach, M. O., and Hawkes, D. J. (1999). Nonrigid registration using free-form deformations: application to breast MR images. *IEEE transactions on medical imaging*, 18(8):712–721.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA.

Setio, A. A. A., Ciompi, F., Litjens, G., Gerke, P., Jacobs, C., van Riel, S. J., Wille, M. M. W., Naqibullah, M., Snchez, C. I., and van Ginneken, B. (2016). Pulmonary nodule detection in ct images: False positive reduction using multi-view convolutional networks. *IEEE Transactions on Medical Imaging*, 35(5):1160–1169.

Shams, R., Sadeghi, P., Kennedy, R. A., and Hartley, R. I. (2010). A survey of medical image registration on multicore and the gpu. *IEEE Signal Processing Magazine*, 27(2):50–60.

Shen, W., Zhou, M., Yang, F., Yang, C., and Tian, J. (2015). Multi-scale convolutional neural networks for lung nodule classification. In *IPMI*, pages 588–599.

Simonovsky, M., Gutiérrez-Becker, B., Mateus, D., Navab, N., and Komodakis, N. (2016). A deep metric for multimodal registration. *MICCAI*.

Simpson, I., Woolrich, M., Groves, A., and Schnabel, J. (2011). Longitudinal brain MRI analysis with uncertain registration. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011*, pages 647–654.

Singh, N., Hinkle, J., Joshi, S., and Fletcher, P. (2013a). A hierarchical geodesic model for diffeomorphic longitudinal shape analysis. In *IPMI*, pages 560–571.

Singh, N., Hinkle, J., Joshi, S., and Fletcher, P. (2013b). A vector momenta formulation of diffeomorphisms for improved geodesic regression and atlas construction. In *ISBI*, pages 1219–1222.

Sotiras, A., Davatzikos, C., and Paragios, N. (2013). Deformable medical image registration: A survey. *IEEE transactions on medical imaging*, 32(7):1153–1190.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. (2014). Striving for simplicity: The all convolutional net. CoRR abs/1412.6806.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958.

Suk, H.-I. and Shen, D. (2013). Deep learning-based feature representation for ad/mci classification. In *MICCAI*, pages 583–590.

Tang, Y. (2013). Deep learning using support vector machines. *CoRR*, abs/1306.0239.

Thirion, J.-P. (1998). Image matching as a diffusion process: an analogy with maxwell's demons. *Medical Image Analysis*, 2(3):243 – 260.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop. Coursera: Neural Networks for Machine Learning.

Van Essen, D. C., Smith, S. M., Barch, D. M., Behrens, T. E., Yacoub, E., Ugurbil, K., and WU-Minn HCP Consortium (2013). The WU-Minn human connectome project: an overview. *NeuroImage*, 80:62–79.

Van Nguyen, H., Zhou, K., and Vemulapalli, R. (2015). Cross-domain synthesis of medical images using efficient location-sensitive deep network. In *MICCAI*.

Vercauteren, T., Pennec, X., Perchant, A., and Ayache, N. (2009). Diffeomorphic demons: Efficient non-parametric image registration. *NeuroImage*, 45(1):S61–S72.

Vialard, F.-X., Risser, L., Rueckert, D., and Cotter, C. J. (2012a). Diffeomorphic 3D image registration via geodesic shooting using an efficient adjoint calculation. *IJCV*, 97(2):229–241.

Vialard, F.-X., Risser, L., Rueckert, D., and Cotter, C. J. (2012b). Diffeomorphic 3D image registration via geodesic shooting using an efficient adjoint calculation. *International Journal of Computer Vision*, 97(2):229–241.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103. ACM.

Viola, P. and Wells, W. (1997). Alignment by maximization of mutual information. *IJCV*, 24(2):137–154.

Wang, B., Prastawa, M., Awate, S., Irimia, A., Chambers, M., Vespa, P., Horn, J. V., and Gerig, G. (2012). Segmentation of serial MRI of TBI patients using personalized atlas construction and topological change estimation. In *ISBI*, pages 1152–1155.

Wang, Q., Kim, M., Shi, Y., Wu, G., and Shen, D. (2015). Predict brain MR image registration via sparse learning of appearance & transformation. *MedIA*, 20(1):61–75.

Watanabe, T. and Scott, C. (2012). Spatial confidence regions for quantifying and visualizing registration uncertainty. In *Biomedical Image Registration*, pages 120–130. Springer.

Weinzaepfel, P., Revaud, J., Harchaoui, Z., and Schmid, C. (2013). DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, pages 1385–1392.

Wellek, S. (2010). *Testing statistical hypotheses of equivalence*. CRC Press.

Wolff, J. J., Gerig, G., Lewis, J., Soda, T., Styner, M., Vachet, C., Botteron, K., Elison, J., Dager, S., Estes, A., Hazlett, H., Schultz, R., Zwaigenbaum, L., and Piven, J. (2015). Altered corpus callosum morphology associated with autism over the first 2 years of life. *Brain*, 138(7):2046–2058.

Wu, G., Kim, M., Wang, Q., Gao, Y., Liao, S., and Shen, D. (2013). Unsupervised deep feature learning for deformable registration of mr brain images. In *MICCAI*, pages 649–656.

Xie, Y., Zhang, Z., Sapkota, M., and Yang, L. (2016). Spatial clockwork recurrent neural network for muscle perimysium segmentation. In *MICCAI*, pages 185–193.

Yang, D., Zhang, S., Yan, Z., Tan, C., Li, K., and Metaxas, D. (2015). Automated anatomical landmark detection ondistal femur surface using convolutional neural network. In *ISBI*, pages 17–21.

Yang, X., Han, X., Park, E., Aylward, S., Kwitt, R., and Niethammer, M. (2016a). Registration of pathological images. In *SASHIMI*. Springer.

Yang, X., Kwitt, R., and Niethammer, M. (2016b). Fast predictive image registration. In *DLMI-A/MICCAI*, pages 48–57.

Yang, X., Kwitt, R., and Niethammer, M. (2017a). Quicksilver: Fast predictive image registration-a deep learning approach. *arXiv:1703.10908*.

Yang, X., Kwitt, R., Styner, M., and Niethammer, M. (2017b). Fast predictive multimodal image registration. In *ISBI*.

Yang, X. and Niethammer, M. (2015). Uncertainty quantification for lddmm using a low-rank hessian approximation. In *MICCAI*, pages 289–296.

Yeh, R., Chen, C., Lim, T. Y., Hasegawa-Johnson, M., and Do, M. N. (2016). Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*.

Young, J. T., Shi, Y., Niethammer, M., Grauer, M., Coe, C. L., Lubach, G. R., Davis, B., Budin, F., Knickmeyer, R. C., Alexander, A. L., et al. (2017). The unc-wisconsin rhesus macaque neurodevelopment database: A structural mri and dti database of early postnatal development. *Frontiers in neuroscience*, 11.

Zacharaki, E. I., Hogea, C., Shen, D., Biros, G., and Davatzikos, C. (2009). Non-diffeomorphic registration of brain tumor images by simulating tissue loss and tumor growth. *NeuroImage*, 46:762–774.

Zhang, M. and Fletcher, P. (2015). Finite-dimensional Lie algebras for fast diffeomorphic image registration. In *IPMI*, pages 249–260.

Zhang, M., Singh, N., and Fletcher, P. T. (2013). Bayesian estimation of regularization and atlas building in diffeomorphic image registration. In *Information Processing in Medical Imaging*, pages 37–48. Springer.

Zheng, Y., Liu, D., Georgescu, B., Nguyen, H., and Comaniciu, D. (2015). 3d deep learning for efficient and robust landmark detection in volumetric data. In *MICCAI*, pages 565–572.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*.

Zuiderveld, K. (1994). Graphics gems iv. chapter Contrast Limited Adaptive Histogram Equalization, pages 474–485. Academic Press Professional, Inc., San Diego, CA, USA.