ETH Zürich
Computer Vision Laboratory
Department of Information Technology and Electrical Engineering
Prof. Dr. Ender Konukoglu

Seminar for CSE

specialization Robotics

spring semester 2021

A review of the voxelmorph and mermaid libraries for medical image registration

---

Submitted by:

Luca Adrian Blum
E-Mail: lblum@student.ethz.ch
Master in Computational Science and Engineering
Matriculation number: 15-923-923

Issue date: 06.07.2021

Submission date: 29.08.2021

Abstract

Two popular learning-based medical image registration frameworks (voxelmorph and mermaid) were reviewed. Furthermore, voxelmorph was trained on intra-modal and inter-modal MR brain images and the corresponding registration performances were evaluated.

Acknowledgments

I am deeply grateful to my advisor Professor Ender Konukoglu for giving me the chance to write my seminar paper in such a fascinating area of research. Additionally, I am thankful for his excellent lectures that introduced me to medical image analysis, which helped me a lot with this seminar work and sparked my interest in this field even more.
I would like to thank Dr. Kristine Haberer for introducing me to the usage of the cluster, which allowed me to conduct my own experiments with the frameworks.

Contents

List of Figures

List of Abbreviations

MR	Magnetic resonance
MRI	Magnetic resonance imaging
SSD	Sum of Squared Differences
NCC	Normalized Cross-Correlation
LCC	Local Cross-Correlation
MI	Mutual Information
KL	Kullback Leibler
LDDMM	Large Deformation Diffeomorphic Metric Mapping
NN	Neural Network
CNN	Convolutional Neural Network
MAP estimate	Maximum a posteriori estimate
OMT	optimal mass transport
vSVF	vector momentum-parameterized stationary velocity field
ASVM	affine-vSVF-Mapping
RDMM	Region-specific Diffeomorphic Metric Mapping

1 Introduction

In this seminar work two registration frameworks are reviewed and evaluated on external data. Section 2 describes the problem of medical image registration and introduces the relevant notation. Section 3 presents important building block of the two frameworks. Section 4 and 5 analyzes the learning based mermaid and voxelmorph framework and reviews the underlying papers. In this report, *mermaid* is used as a synonym for *learning-based mermaid*. Subsequently we tried to apply both frameworks for different scenarios in section 6. The datasets were T1-weighted and T2-weighted volumetric MR brain images. The networks were trained and evaluated intra-modal on the T1-weighted and T2-weighted dataset and inter-modal on both datasets together.

2 Medical Image Registration

Medical image registration tackles the problem of comparing images when they are taken at different times, with different modalities or from different subjects. The terminology for the individual settings can be found in Figure 1 (Konukoglu 2021) where the different colors indicate the difficulty of the registration problem (green easiest, red hardest).

	Same subject's images	Different subjects' images
Same modality	Intra-subject, intra-modality	Inter-subject, intra-modality
Different modality	Intra-subject, inter-modality	Inter-subject, inter-modality

Figure 1: Classification of registration problems (Konukoglu 2021)

Possible applications are (Konukoglu 2021):

- aligning images of different modalities
- longitudinal analysis
- atlas-based segmentation
- population analysis
- image analysis for interventions

To make the images comparable, image registration aligns them through spatial transformation $\varphi(x)$ (Konukoglu 2021). In general there are parametric and non-parametric approaches. As the name suggests, parametric transformation models parametrize the transformation which makes them relatively low dimensional (Yang et al. 2017b). Examples are rigid, similarity and affine transformations or B-spline models (Rueckert et al. 1999). In contrast to parametric models, non-parameteric models have a parameter (or a parameter vector) for each voxel and therefore parametrize the transformation locally (Yang et al. 2017b). This results in a high-dimensional optimization which makes the computation more expensive than parametric approaches. Examples are displacement-based registration models like the classical Horn and Schunck optical flow (Horn and Schunck 1981) or LDDMM (Beg et al. 2005).
Formally image registration can be described as following

Definition 2.1 (image registration (adapted from Yang et al. 2017b))
Let Ω be an open subset of $\mathbb{R}^d$. Given a moving (source) image $M(x)$ and a target image $T(x)$ with $x \in \Omega$, the goal of image registration is to find a transformation $\varphi : \Omega \to \Omega$, which maps the moving image to the target image in such a way that the deformed moving

image is similar to the target image:

$$(M \circ \varphi)(x) \approx T(x)$$

where d denotes the spatial dimension, x is the spatial coordinate of the fixed target image T and Ω is the image domain of the fixed target image (Yang et al. 2017b). Usually it is desired that φ fulfills some smoothness conditions. To get meaningful deformations the transformation should be bijective because the transformation should not create holes and no folding should appear (Younes 2010). Different approaches make further constraints on the smoothness of the transformation. Generally the problem can be formulated as an energy minimization problem

$$E(\varphi) = Reg(\varphi) + \frac{1}{\sigma^2} Sim((M \circ \varphi)(x), T(x)) \tag{1}$$

where σ is a balancing constant, $Reg(\varphi)$ is the regularizer of the transformation and $Sim(\cdot)$ is an image dissimilarity measure, which becomes small if the images are similar to each other (Yang et al. 2017b). The regularizer penalizes spatially irregular transformations and therefore encourages the transformation to fulfill the smoothness constraints (Yang et al. 2017b). Additionally for non linear transformation, the regularizer is needed to make the problem well-posed (Konukoglu 2021). The choice of the dissimilarity measure depends on the task.

2.1 Metrics

In the following the focus is on intensity based metrics because they can be used for non-parametric registration like voxelmorph or mermaid . There are other metrics that depend on features/landmarks (e.g. thin plate splines (Bookstein 1989)).

2.1.1 Sum of Squared Differences SSD

As the name suggests, SSD just sums the squared differences at each location.

$$\mathcal{L}(\varphi) = \sum_{x \in \Omega} \|(M \circ \varphi)(x) - T(x)\|_2^2 \tag{2}$$

The loss term gets small when the transformed moving image is similar to the the target image. It is important to note that there is generally no unique solution to the image registration problem 1 with this loss function (Konukoglu 2021). Also one can use another norm than the squared L2-norm like the L1-norm to get a more robust loss (Konukoglu 2021). Additionally it assumes that the intensities are comparable across the images and therefore it is inappropriate for inter-modality registration (Konukoglu 2021).

2.1.2 Normalized Cross-Correlation NCC

The normalized cross-correlation is given by

$$\mathcal{L}(\varphi) = \frac{1}{|\Omega|}\frac{1}{\sigma_T \sigma_M}\sum_{x\in\Omega}(T(x) - \mu_T)((M \circ \varphi)(x) - \mu_M) \tag{3}$$

where $|\Omega|$ is the number of pixels in the target image, σ_T, μ_T and σ_M, μ_M are the standard deviations and means of the target and transformed moving image (Konukoglu 2021). The standard deviations and means can be approximated empirically. Note that this is the 2D version of the Pearson correlation coefficient and it assumes pixel intensities as independent random variables (Konukoglu 2021). Hence, it can be used for inter-modality registration. It takes values between -1 and 1 where an absolute value of 1 means a perfect linear correlation and value of 0 means no linear dependency. For the image registration problem 1 we want the absolute value as high as possible and thus it makes sense to maximize NCC^2 which is equivalent to minimize $-NCC^2$ (Konukoglu 2021).

2.1.3 Mutual Information MI

The Mutual Information is given by

$$\mathcal{L}(\varphi) = D_{KL}(p(T, M)|p(T)p(M)) \tag{4}$$

where $D_{KL}(\cdot|\cdot)$ is the Kullback-Leibler KL divergence that measures the distance between the joint distribution of the intensities $p(T, M)$ and the product of the marginal distributions $p(T)p(M)$ (Konukoglu 2021). Like NCC it assumes that pixel intensities are independent random variables (Konukoglu 2021). Thus, it can be used for inter-modality registration. A loss of 0 corresponds to two independent images which means $p(T, M) = p(M)p(T)$ (Konukoglu 2021). Therefore MI should be maximized for image registration (Konukoglu 2021) and thus $-MI$ should be minimized for the general image registration problem 1. KL divergence and the corresponding probability distributions can be approximated with histograms.

2.2 Large deformation diffeomorphic metric mapping LDDMM

In order to get meaning full deformation there are two cases we want to avoid (Younes 2010, p. 183):

- No creation of holes by the transformation. This means that for every point $y \in \Omega$ there should be a $x \in \Omega$ such that $y = \varphi(x)$
- No folds should occur by the transformation. This means that for two distinct points x and x' in Ω they should not be mapped to the same point y in Ω

Therefore the transformation should be bijective in Ω. Additionally the transformation and its inverse should be smooth. In this context the smoothness of φ and its inverse is at least once differentiable and invertible, which means that the transformation φ should be diffeomorphic (Polzin 2018, p. 61).

Definition 2.2 (C^1-Diffeomorphisms on Ω, adapted from (Younes 2010))
A C^1-diffeomorphism of Ω is a continously differentiable bijection $\varphi : \Omega \to \Omega$ such that φ^{-1} is continuously differentiable. Formally:

- *φ is bijective*
- $\varphi \in C^1$
- $\varphi^{-1} \in C^1$

The term *large deformation diffeomorphic metric mapping* was introduced by Beg et al. 2005. This registration approach is different to other approaches because it gives diffeomorphic mappings and allows for large deformation. Large deformations are not always possible for so called elastic models that uses a sum of identity and a displacement $u : \Omega \to \mathbb{R}^d$ i.e. $\varphi(x) = x + u(x)$ because of the regularization of u (Polzin 2018). Additionally it can happen that folding occurs and therefore the mapping is not diffeomorphic (Christensen 1994). Christensen, Rabbitt, and Miller 1996 developed the large deformation model that ensures diffeomorphic mappings where the transformation of the domain is given by the endpoint of the flow of a time-dependent velocity vector field $v_t : \Omega \to \mathbb{R}^d, t \in [0, 1]$ specified by the ODE $\dot{\phi}_t = v_t(\phi_t)$ with initial conditions $\phi_0 = x$. Hence the registration is defined by the spatio-temporal velocity field $\hat{v}(x, t)$ which means from the view of an particle that the final transformation is obtained by following the velocity field over time (Yang et al. 2017b).
Beg et al. 2005 estimated the optimal transformation for the *relaxation* formulation (5) via the basic variational problem in the space of smooth velocity vector fields V on the domain Ω:

$$\hat{v}_t = \underset{v_t:\phi_1=v_t(\phi_t),\phi_0=x}{\arg\min} \left(\int_0^1 \|v_t\|_V^2 \, dt + \frac{1}{\sigma^2} \left\| I_0 \circ \phi_1^{-1} - I_1 \right\|_{L^2}^2 \right) \tag{5}$$

The first term can be seen as a regularizer on the time-dependent velocity vector field v_t and the norm is chosen such that the resulting transformation is diffeomorphic. The second term measures the dissimilarity of the target and transformed source image. The optimal velocity of this cost function allows to compute the desired transformation φ through time integration of the velocity field:

$$\varphi = \phi_1 = \phi_0 + \int_0^1 \hat{v}_t(\phi_t) dt \tag{6}$$

Beg et al. 2005 derived the Euler-Lagrange equation which the minimizer of the variational problem 5 has to satisfy. With the Euler-Lagrange equation the problem can be solved numerically. An important observation is that in the Euler-Lagrange equation a smoothing operator is applied which yields smooth velocity fields. Therefore, depending on the choice of the smoothing operator, a diffeomorphic transformation can be computed. An alternative optimization can be done with the *shooting* formulation. It uses the insight that the optimal solution needs to be a straight line (i.e. a geodesic) connecting the two images and therefore one needs to optimize only over the intercept and the slope of the line which reduces the complexity of the optimization drastically (Yang et al. 2017b). The slope corresponds to the initial momentum and the intercept to the initial map which is the identity map (Yang et al. 2017b). Hence the complete spatio-temporal deformation φ is determined by the initial momentum. The initial momentum and velocity 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 (Yang et al. 2017b). Due to this smoothing operator the velocity will be smooth and therefore diffeomorphic mappings can be created. In contrast, the initial momentum does not have to be smooth.

3 Deep Learning

This section introduces non-standard deep learning building blocks used by the frameworks. It is assumed that the reader is familiar with the basics of fully connected neural networks NN and convolutional neural networks CNN. Otherwise Goodfellow, Bengio, and Courville 2016 provides an excellent resource to enter the field of deep learning.

3.1 Activation Functions

Usually every hidden unit in a neural network applies a non-linearity which is called activation function. Typically this activation function is used after the affine transformation with the weight matrix (for fully connected NNs) or with the convoluational matrix (for CNNs). Common activation functions like sigmoid, tanh or ReLU are shown in Figure 2. Additionally the PReLU activation function is shown because it is used by the mermaid framework.
The formula for PReLU(x) is given by (Yang et al. 2017b)

$$PReLU(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases}$$

where a is a parameter that is learned when training the network.

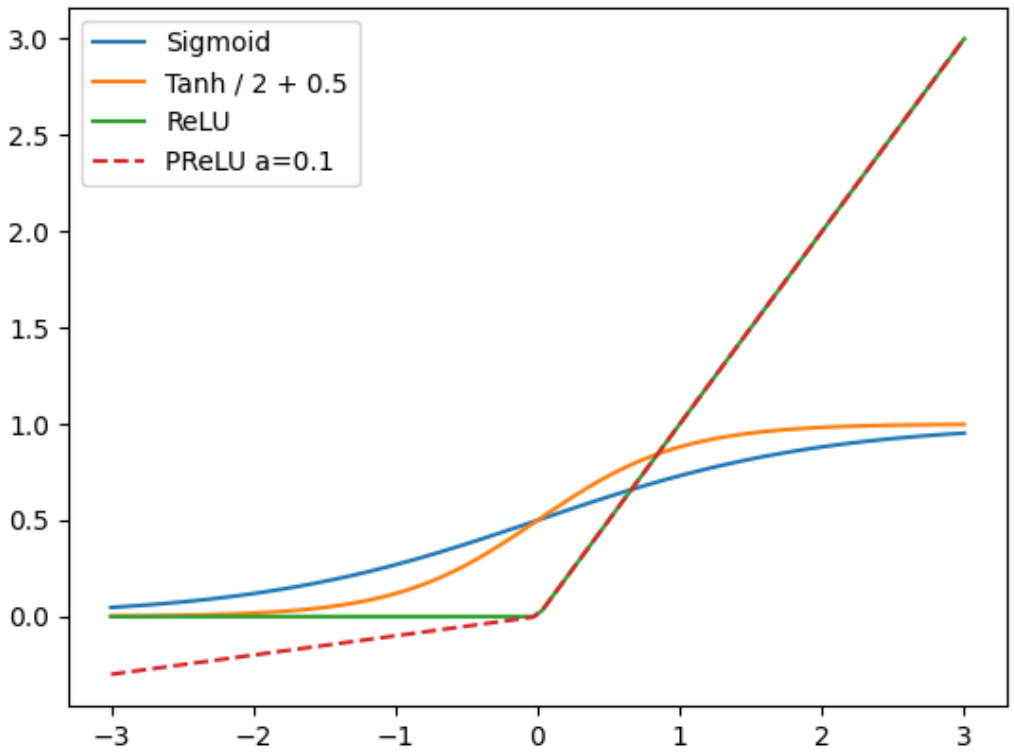

Figure 2: Activation functions

3.2 Dropout

Dropout was first proposed by Hinton et al. 2012. It was introduced to prevent neural networks from overfitting. During training each hidden unit is randomly omitted with probability p (Hinton et al. 2012). This means for the output of a hidden layer

$$\mathbf{y} = f(\mathbf{W}\ \mathbf{x}) \circ \mathbf{m} \qquad m_i \sim Bernoulli(1-p)$$

where $\mathbf{y}$ is the layer output, $f(\cdot)$ is the activation function, $\mathbf{W}$ is the layer weight matrix, $\mathbf{x}$ is the layer input, and $\mathbf{m}$ is the layer dropout mask, with each element m_i being 0 with probability p (Labach, Salehinejad, and Valaee 2019). During testing there is no dropout. Thus all neurons are used. Because the network was smaller during training the hidden layer outputs need to be scaled by $(1-p)$ (Hinton et al. 2012).

$$\mathbf{y} = (1-p)f(\mathbf{W}\ \mathbf{x})$$

Dropout can be seen as model averaging over an ensemble of neural networks. There is no need to train and average many different independent neural networks (e.g. bagging) because random dropout allows to train a large number of different networks (Hinton

et al. 2012). For each training sample there is almost certainly another network but all of these share the same weights for the hidden units (Hinton et al. 2012). This makes training much more efficient. At test time, the full network is used which can be interpreted as a mean network (Hinton et al. 2012). This mean network approximates the geometric mean of the ensemble's outputs (Labach, Salehinejad, and Valaee 2019). This is different from bagging which uses the arithmetic mean. Additionally bagging trains many models independently where as dropout trains only one model.

Many dropout methods have been proposed and Labach, Salehinejad, and Valaee 2019 provides a great overview. Our focus is on *Monte-Carlo* dropout since it is used by the mermaid framework. Monte-Carlo dropout can be used to estimate the model uncertainty (Gal and Ghahramani 2016). Gal and Ghahramani 2016 provided a Bayesian view of dropout and showed that it is mathematically equivalent to an approximation to a probabilistic deep Gaussian process. A deep Gaussian process outputs a probability distribution and is a Bayesian machine learning model (Labach, Salehinejad, and Valaee 2019). This predictive probability distribution has the form

$$
\begin{aligned}
p(y|x, D) &= \int p(y, w|D, x)dw \\
&= \int \frac{p(y, w|D, x)}{p(w|D, x)} p(w|D, x)dw \\
&= \int p(y|w, D, x)p(w|D, x)dw \\
&= \int p(y|w, x)p(w|D)dw
\end{aligned}
$$

where y is the output of the network, x the input and D the data that were used for training the network. In comparison, the frequentists approach uses the maximum likelihood estimate which picks the model weights such that the probability of observing the training data is maximized. This gives only a point estimate and thus can not be used for uncertainty quantification. Usually it is not feasible to calculate the predictive probability distribution exactly because we would need to integrate over all possible models. The solution is to restrict the model on a set of random variables w which are the weight matrices (Yang et al. 2017a). The next problem arises from the calculation of the posterior of the weights $p(w|D)$ because it is usually intractable. The solution is to use a tractable variational distribution as replacement (Yang et al. 2017a). Gal and Ghahramani 2016 have shown that Monte Carlo sampling can estimate characteristics of the predictive probability distribution (e.g. mean and standard deviation). Further they have demonstrated that dropout minimises the Kullback-Leibler divergence between an approximate distribution and the posterior of a deep Gaussian process.

To get estimates for the characteristics of the predictive probability distribution, dropout is enabled during testing. The training process is the same as for standard dropout. To

estimate the mean and the variance of an output, the same input is given to the network T times with standard dropout but with varying randomly generated dropout masks (Gal and Ghahramani 2016). Estimates for the mean and variance are given by

$$\mathbb{E}[\mathbf{y}] = \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_{\mathbf{t}}(\mathbf{x})$$

$$Var(\mathbf{y}) = \tau^{-1}\mathbf{I}_D + \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_{\mathbf{t}}(\mathbf{x})^T\hat{\mathbf{y}}_{\mathbf{t}}(\mathbf{x}) - \mathbb{E}[\mathbf{y}]^T\mathbb{E}[\mathbf{y}]$$

where $\hat{\mathbf{y}}_{\mathbf{t}}(\mathbf{x})$ is the t^{th} output of the network given inputs $\mathbf{x}$ and τ is a constant determined by the model structure (Labach, Salehinejad, and Valaee 2019). $\hat{\mathbf{y}}_{\mathbf{t}}(\mathbf{x})$ is a row vector and the sum is over the outer products (Gal and Ghahramani 2016). $\mathbf{I}_D$ is the $D \times D$ identity matrix where D is equal to the length of $\hat{\mathbf{y}}_{\mathbf{t}}(\mathbf{x})$. Therefore the actual prediction of the network is the mean and the variance can be used for uncertainty quantification.

3.3 U-net

The U-net architecture was developed by Ronneberger, Fischer, and Brox 2015 for image segmentation. It is an autoencoder where the encoder is called the *contracting path* and the decoder is called *expansive path*. The contracting path repeatedly applies a downsampling block that consists of two 3x3 valid convolutions with stride 1, followed by a ReLU activation and a 2x2 max pooling layer with stride 2 (Ronneberger, Fischer, and Brox 2015). Each downsampling block doubles the number of feature channels through the first 3x3 convolution. This means for the output dimensions x^c_{out}, y^c_{out} and c^c_{out} of such a block given by the input dimensions x^c_{in}, y^c_{in} and c^c_{in}

$$x^c_{out} = \frac{x^c_{in} - 4}{2}$$
$$y^c_{out} = \frac{y^c_{in} - 4}{2}$$
$$c^c_{out} = 2 * c^c_{in}$$

So far this is a typical fully convolutional network. The interesting part can be found in the expansive path. An upsampling block consists of an up-convolution (upsampling with scale 2 + 2x2 valid convolution with stride 1), two 3x3 valid convolutions with stride 1, followed by a ReLU and a concatenation with the correspondingly cropped feature map from the contracting path (Ronneberger, Fischer, and Brox 2015). Therefore we have skip connections. The first 3x3 convolution as well as the 2x2 up-convolution halves the number

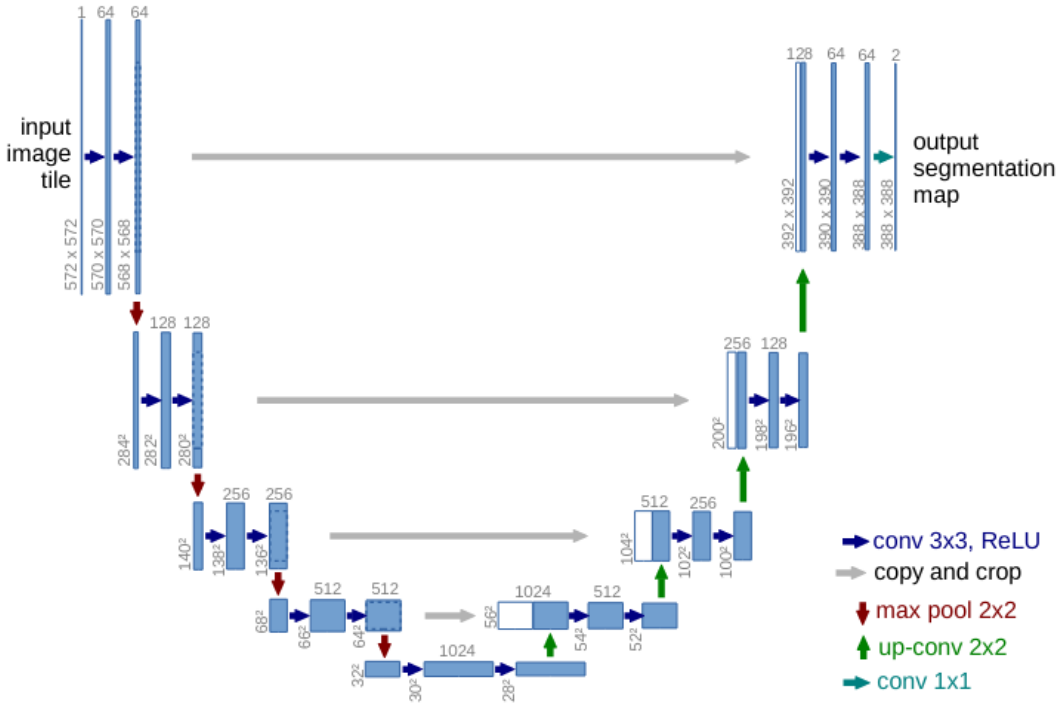

Figure 3: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. (Ronneberger, Fischer, and Brox 2015)

of feature channels(Ronneberger, Fischer, and Brox 2015). The output dimensions x^e_{out}, y^e_{out} and c^e_{out} of an upsampling block is given by the input dimensions x^e_{in}, y^e_{in} and c^e_{in}

$$x^e_{out} = 2 * x^e_{in} - 4$$
$$y^e_{out} = 2 * y^e_{in} - 4$$
$$c^e_{out} = \frac{c^e_{in}}{2}$$

This is more or less the symmetric behavior of the contracting part and thus yields an u-shaped architecture (Ronneberger, Fischer, and Brox 2015). The final layer applies a 1x1 valid convolution with stride 1 to map each 64-component feature vector to the desired number of classes (Ronneberger, Fischer, and Brox 2015). This architecture won several image segmentation competitions. Therefore it is used commonly and was adapted to other tasks like image registration. Differences to the original u-net architecture are for example the usage of padded convolution, different stride, different kernel size, different number of kernels, different scale for the upsampling, different number of levels for the

contracting and expansive path and other loss function. Nevertheless the u-shape or the symmetric behavior mostly remained.

4 Mermaid

The development of the framework started with a supervised deep learning approach. The network took as input a target image T and a moving image M and predicted the initial momentum patch-wise. This allowed to compute the final diffeomorphic displacement field and the corresponding warped moving image. A disadvantage was that supervised training data for the initial momentum was needed. Therefore the model can be interpreted as a surrogate for the initial momentum generation method. Nevertheless state-of-the-art performance could be achieved. The next step focused on learning the registration metric. Specifically the goal was to train spatially varying regularizer by a CNN. The succeeding model ASVM extended the initial quicksilver network. The input is still the target image T and the moving image M, and the output is the deformation map and the corresponding warped moving image. The first part of the network is a multi-step affine network that takes the target image T and the moving image M as input and predicts the affine mapping and the warped moving image. These are the input for the vSVF registration which predicts the initial momentum and therefore the final mapping and final warped moving image. The vSVF registration can also operate in a multi-step fashion. The corresponding velocity field of the predicted momentum was limited to be stationary. The training was unsupervised and comparable and sometimes better performance to popular registration tools was achieved. The last paper fused the idea of metric learning and the ASVM model. Particularly, a time- and spatially-varying regularizer was proposed and integrated into the ASVM model. Additionally, the velocity was no more limited to be static. This lead to the RDMM model which can predict mappings and warped moving images regularized with spatio-temporal regularizer.

4.1 Quicksilver

Following the first two papers of the mermaid library are described. They introduce *Quicksilver* a fast predictive image registration framework. *Quicksilver* uses a supervised deep learning approach where the network predicts the initial momentum of an LDDMM solution and therefore generates diffeomorphic transformations φ (Yang, Kwitt, and Niethammer 2016). Given the initial momentum, the complete spatio-temporal deformation φ is determined (Yang et al. 2017b). The deformation can be computed by first calculating the velocity from the initial momentum $v = Km$ and then integrating the velocity field over unit time (Yang et al. 2017b). K is a smoothing operator that governs the theoretical properties of LDDMM and therefore the velocity will be smooth and thus

the deformation, too (Yang et al. 2017b). In contrast, the initial momentum does not have to be smooth and hence the neural network does not need to take this into account (Yang et al. 2017b). Also, this allows to predict the initial momentum patch-by-patch which is crucial for large images (Yang et al. 2017b). Another advantage of the initial momentum parametrization is that the initial momentum is zero in homogenous regions (Yang, Kwitt, and Niethammer 2016). This can be an issue for velocity or displacement field parametrization because a homogenous region provides locally no information (Yang, Kwitt, and Niethammer 2016). Given a trained model, dramatic speed-ups compared to direct optimization of the LDDMM problem 5, while maintaining high prediction accuracy can be achieved (Yang et al. 2017b). Additionally an uncertainty quantification of the deformation field is provided.

4.1.1 Model

The input to the model is the moving and target image. The output is the initial momentum to calculate the displacement field φ. The training is supervised, which means that for each moving and target training image a corresponding initial momentum needs to be provided. Since there can be large images and the memory in modern GPUs is limited, it is not feasible to predict the entire displacement field (Yang et al. 2017b). Rather they used a patch based prediction. Therefore, the entire 3D image prediction is accomplished patch-by-patch via a sliding window approach (Yang et al. 2017b). After all patches of an input have been collected, the whole initial momentum can be reconstructed by averaging the overlapping regions (Yang et al. 2017b). Afterwards smoothing with K to get the velocity v and time-integration is done on the whole image to get the final displacement field φ. However, it should be noted that during training only the patch predictions are required and no reconstruction of the whole initial momentum is performed. In summary, in a sliding window fashion the model takes as input a patch of the target and the corresponding patch of the moving image and predicts a patch of the initial momentum which can be used to calculate the whole diffeomorphic displacement field φ.
To get the initial momentum two networks are used. Both networks have the same structure. The only difference is the input and the corresponding output. First the *prediction network* is trained patch-wise with moving and target images and the ground truth initial momentum. Then the target image T is warped to the moving image M using the predicted momentum from the prediction network. Afterwards the *correction network* is trained patch-wise with the moving image M and the warped target image $(T \circ \varphi^{-1})(x)$ to predict the difference between ground truth momentum and the predicted momentum from the prediction network. Lastly, the initial momentum and the correction momentum are added together to get the final momentum. Figure 4 shows the complete architecture.

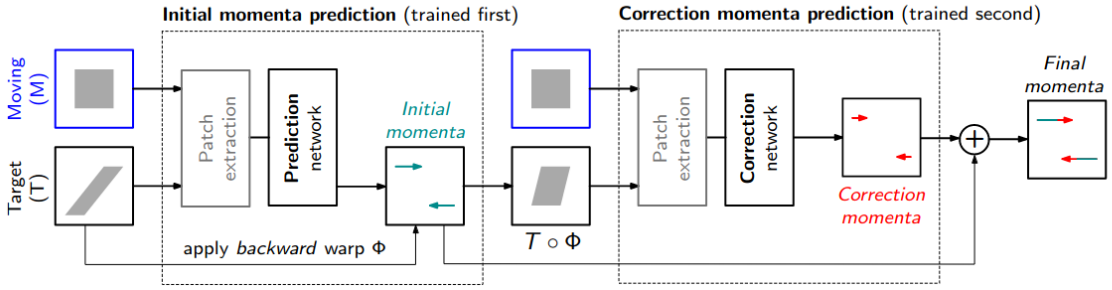

Figure 4: Quicksilver architecture (Copied from Yang et al. 2017b) The prediction network takes as input a moving image M and a target image T. It tries to predict the initial momentum patch-wise. The training is supervised and therefore a ground truth initial momentum is needed. After the prediction, the target image T is warped to the moving image M. Then the correction network takes the moving image M and the warped target image $(T \circ \varphi^{-1})(x)$ as input and tries to predict the difference between the ground truth initial momentum and the predicted initial momentum. Finally, the initial momentum and the correction momentum are added to get the final momentum

Both networks have the same structure and Yang et al. 2017a built the networks with an encoder-decoder architecture where the encoder consists of two parallel encoder. They used two encoders because it can be beneficial for multi-modal image registration (Yang et al. 2017a). Each encoder learns features from the moving/target image patches independently. The architecture can be found in Figure 5 (Yang et al. 2017b). One encoder contains two blocks of three 3x3x3 3D convolutional layers with stride 1, padding 1 and PReLU activation function. Each block is followed by a 2x2x2 3D convolution with a stride of 2, padding of 1 and PReLU activation. The 2x2x2 convolution with stride 2 can be seen as pooling operation as described in Springenberg et al. 2015. After the two blocks, both learned features are concatenated and sent to three parallel decoders. They used for each spatial dimension (x, y and z) one decoder because they observed that this is much easier to train than one large decoder. Yang et al. 2017b used a decoder structure that is the inverse of the encoder, expect that the number of features is doubled (due to the concatenation of two encoder outputs). Additionally the final layer applies no PReLU activation. Another reason to use convolution and transpose of convolution to perform pooling/unpooling is the fact that the two encoders perform pooling independently which prevented them from using the pooling index for unpooling in the decoder. They use the 1-norm as loss function because it tolerates outliers and generates sharper momentum predictions. Additionally, they chose stochastic gradient descent for optimization.
To allow uncertainty quantification, Yang, Kwitt, and Niethammer 2016 incorporated dropout with probability of 0.2 after each convolutional layer except for those used as pooling/unpooling layers and the final layer. As described in Section 3 Monte-Carlo dropout can be used to quantify the uncertainty. This means that dropout stays enabled

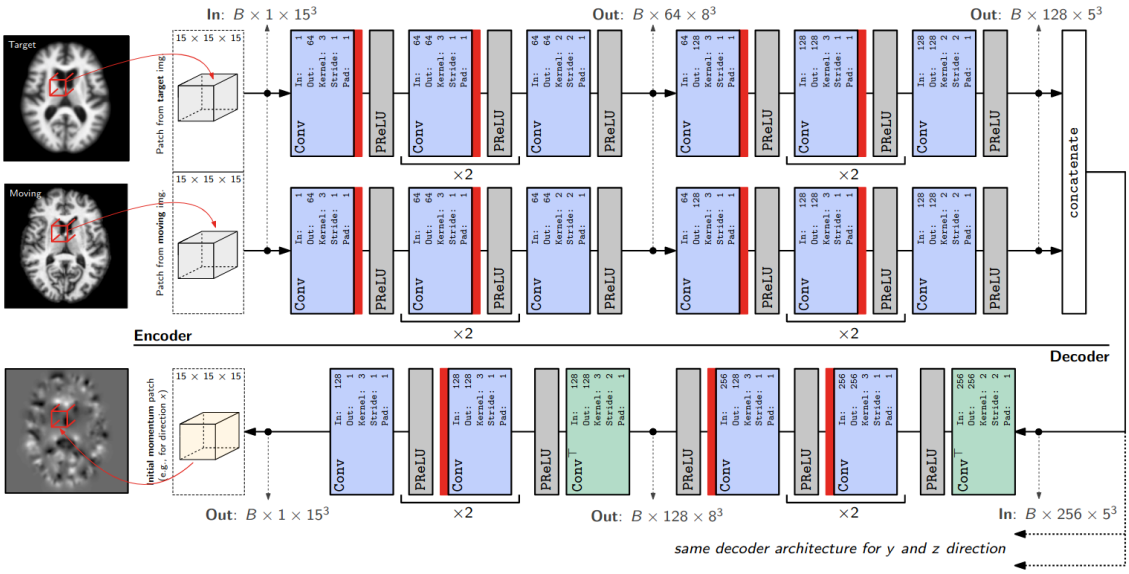

Figure 5: Quicksilver network architecture (Copied from Yang et al. 2017b) The network takes two 3D patches from the moving and target image as the input, and outputs 3 3D initial momentum patches (one for each of the x,y and z dimensions respectively; for readability, only one decoder branch is shown in the figure). In case of the deterministic network the dropout layers ■ are removed. *Conv*: 3D convolution layer. $Conv^T$: 3D transposed convolution layer. Parameters for the *Conv* and $Conv^T$ 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.

during testing and that the prediction and the variance can be calculated from T forward passes of the same input x with different random masks.

Using a sliding window approach with voxel stride 1 can easily lead to a huge number of patch predictions. To reduce the number of predictions, Yang et al. 2017a used patch pruning and a large voxel stride (14 for 15x15x15 patches). Patch pruning ignores all the patches from the background (Yang et al. 2017a). This works because for constant image regions the initial momentum is zero (Yang et al. 2017a). The large voxel stride can be justified, since the initial momentum has compact support at edges and the spatial shift invariance from the pooling/unpooling layers (Yang et al. 2017a). According to Yang et al. 2017a this allows to reduce the patch prediction by 99.995% for 3D brain images of size 229x193x193.

4.1.2 Results and Conclusion

The quicksilver model retains all theoretical properties of LDDMM and therefore predicts diffeomorphic transformation. Additionally the model allows to take large strides for patch-wise prediction, without substantial decrease in registration accuracy (Yang et al. 2017a). This enables to predict fast and accurate deformation predictions. However,

note that there is no additional regularization for training the network, because it was already encoded when computing the initial momentum fields to obtain training data.

4.2 Metric Learning for Image Registration

In this paper Niethammer, Kwitt, and Vialard 2019 introduced a spatially-adaptive regularizer. Usually the regularizer is just constant and therefore the same spatial regularity is applied everywhere. This is not realistic because different deformation scales are present in different image regions and thus the regularization should be adapated accordingly (Niethammer, Kwitt, and Vialard 2019). The whole model takes as input a target image T and a moving image M and predicts the corresponding diffeomorphic deformation. It can be splitted into two main parts. The first part is a model that predicts the diffeomorphic deformation for a moving image M given the target image T. This is comparable to the quicksilver network expect that they restrict the resulting velocity to be constant in time. Niethammer, Kwitt, and Vialard 2019 call this vector momentum-parameterized stationary velocity field *vSVF* registration model. Here it was not the focus of them and the architecture is assumed to be given. Their approach could be generalized to non static velocity fields like LDDMM which would require a spatial and temporal varying regularizer. On top of this they built the spatial varying regularization which is another CNN. This network takes as input a momentum vector field (predicted by the vSVF) and a moving image M and predicts a smoothed vector field. The training is unsupervised. Since the final mapping φ is influenced by the momentum and the regularizer they optimized jointly over both networks.

4.2.1 Model

Remember that the velocity and the vector momemtum where linked by $m = Lv$. L can be seen as a spatially invariant smoothing operator that encodes the desired level of smoothness (Niethammer, Kwitt, and Vialard 2019). Niethammer, Kwitt, and Vialard 2019 used multi-Gaussian kernels given by

$$v = \left(\sum_{i=0}^{N-1} w_i G_i \right) \star m, \qquad w_i > 0, \qquad \sum_{i=0}^{N-1} w_i = 1$$

where $\star$ is a convolution, G_i is a normalized Gaussian centered at zero with standard deviation σ_i and w_i is a positive weight. The standard deviations σ_i are set manually and the weights $w_i(x)$ need to be defined. To encourage smooth solutions it is beneficial to penalize small σ's. This was done through the introduction of *optimal mass transport* OMT. The standardized version is defined as

$$\widehat{OMT}(w) = \left| log \frac{\sigma_{N-1}}{\sigma_0} \right|^{-r} \sum_{i=0}^{N-1} w_i \left| log \frac{\sigma_{N-1}}{\sigma_i} \right|^r$$

The motivation behind this is that if we order the standard deviations and the corresponding weights $0 < \sigma_0 < \sigma_1 < ... < \sigma_{N-1}$, for smooth transformations we want large standard deviations, which means that we want large values for weights with a higher index/standard deviations. Thus OMT penalizes weight distributions that gives larger values for weights with small index/standard deviations.
In order to get a spatial varying regularizer Niethammer, Kwitt, and Vialard 2019 introduced *localized* multi-Gaussian kernels which let the weights vary spatially $w_i(x)$. To get diffeomorphic transformations the weights were set to

$$w_i(x) = G_{\sigma_{small}} \star \omega_i(x)$$

where $\omega_i(x)$ are pre-weights which are all convolved with the same Gaussian with small standard deviation (Niethammer, Kwitt, and Vialard 2019). They used 0.02 in 2D and 0.05 in 3D. To guarantee diffeomorphic transformations the pre-weights have to be not too degenerated. Thus they used color-TV regularization for the pre-weights. In summary, there are two regularizer added to the vSVF formulation which regularize the spatial weights through OMT and the pre-weights through color-TV.
The computation of the velocity from the momentum changes as follows (Niethammer, Kwitt, and Vialard 2019)

$$v_0(x) = \sum_{i=0}^{N-1} \sqrt{w_i(x)} \int_y G_i(|x-y|) \sqrt{w_i(y)} m_0(y) dy$$

which is for spatially constant weights just the standard multi-Gaussian approach. The overall variance of the multi-Gaussian is bounded by the variances of its component (manually chosen) and therefore one can specify a desired regularity level (Niethammer, Kwitt, and Vialard 2019).
Therefore values for the pre-weights $\omega_i(x)$ are needed, which allows the computation of the spatially varying weights $w_i(x)$ (given σ_{small}) which defines the localized multi-Gaussian kernels (given σ_i) and therefore allows to compute the velocity. These pre-weights were optimized through the use of a CNN. The input is the transformed moving image and the output are the pre-weights. The architecture is given by

$$\begin{aligned} &conv(d+1, n_1) \rightarrow BatchNorm \rightarrow lReLU \\ &\rightarrow conv(n_1, N) \rightarrow BatchNorm \rightarrow weighted - linear - softmax \end{aligned}$$

where $conv(a, b)$ takes as input a channels and outputs b channels. They used a kernel size of 5 (5x5 in 2D and 5x5x5 in 3D). Additionally n_1 was set to 20. The weighted-linear softmax ensures that the pre-weights are positive and sum up to 1.
The joint optimization is done in a global and a local stage. In the initial global stage Niethammer, Kwitt, and Vialard 2019 chose a reasonable set of global Gaussian weights and optimized only over the momenta. The global weights were linear with respect to their associated variances (Niethammer, Kwitt, and Vialard 2019

$$w_i = \frac{\sigma_i^2}{\sum_{j=0}^{N-1} \sigma_j^2}$$

This helps the local optimization because it can be used as a reasonable starting point for the local stage (Niethammer, Kwitt, and Vialard 2019). This is comparable to quicksilver expect that the training is unsupervised, the velocity is temporal static and the smoothing of the momenta is done with multi-Gaussian kernels. In the next local stage the momenta and the parameters of the CNN to get spatially-localized weights were optimized together. The standard deviations of the Gaussian for the localized multi-Gaussian kernels were chosen as $\{0.01, 0.05, 0.1, 0.2\}$. The similarity measure was chosen as NCC with $\sigma = 0.1$. Thus the total loss function consisted of the NCC similarity measure, an L2 regularization for the velocity field, OMT regularization for the spatial weights, color-TV regularization for the pre-weights and some constraints to ensure diffeomorphic transformations given by the vSVF formulation.

4.2.2 Results and Conclusion

Niethammer, Kwitt, and Vialard 2019 have proposed an approach to learn spatially varying regularizer. The regularizer is learned by a CNN and can be integrated to existing deformable registration models. Additionally, they achieved good performance on both synthetic and real data.

4.3 Networks for Joint Affine and Non-parametric Image Registration ASVM

Shen et al. 2019 have introduced an end-to-end deep-learning framework which consists of an affine registration and a non-linear vSVF registration which they call *ASVM*(affine-vSVF-Mapping). The affine registration is done with a multi-step network. The non-linear registration is formulated as a vSVF model and has a momentum generation network and a vSVF component. Also, this part can be refined with multiple steps. An overview of the overall architeture can be seen in Figure 6.

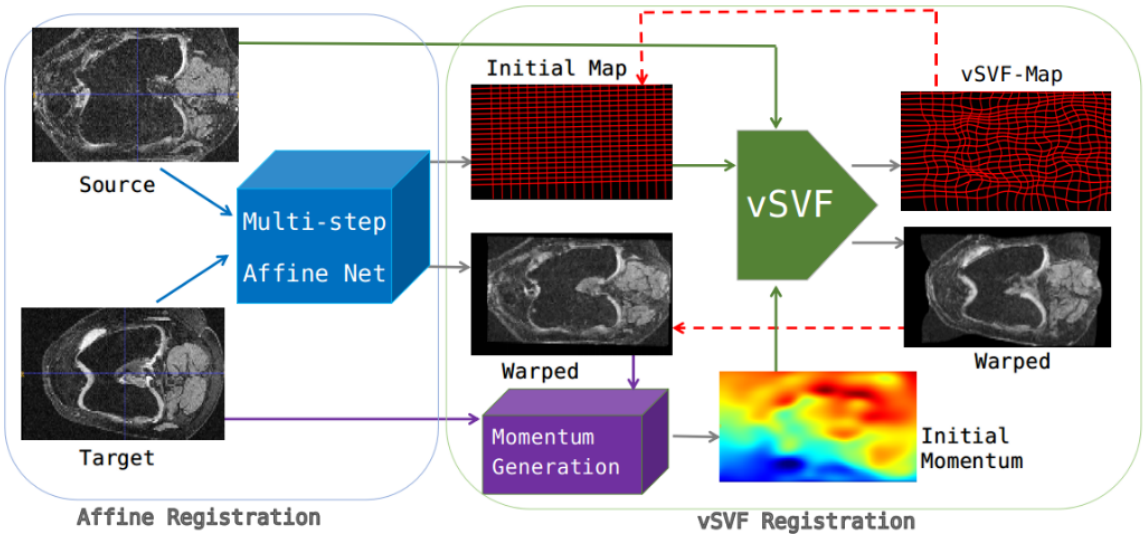

Figure 6: ASVM architecture (Copied from Shen et al. 2019) The framework consists of affine (left) and vSVF (right) registration components. The affine part outputs the affine map and the affinely warped source image. The affine map initializes the map of the vSVF registration. The affinely warped image and the target image are input into the momentum generation network to predict the momentum of the vSVF registration model. The outputs of the vSVF component are the composed transformation map and the warped source image, which can be either taken as the final registration result or fed back (indicated by the dashed line) into the vSVF component to refine the registration solution.

4.3.1 Multi-step Affine Network

The multi-step affine network predicts the affine deformation. Shen et al. 2019 have observed that training a single deep convolutional network does not perform well in practice. Therefore they composed the affine transformation from several steps which results in better accuracy and stability. Thus the network was designed as recurrent network, which progressively refines the predicted transformation as shown in Figure 7 (Shen et al. 2019). The DNN module is a convolutional neural network with two fully connected layers at the end.

Since interpolation after each step would lead to numerical instabilities and dissipation, they updated the affine transformation directly. Let the affine parameters be $\Gamma = (A\ b)$ where $A \in \mathcal{R}^{dxd}$ and $b \in \mathcal{R}^d$ and d is the image dimension. The update is given by

$$
\begin{aligned}
A_{(t)} &= \tilde{A}_{(t)} A_{(t-1)} \\
b_{(t)} &= \tilde{A}_{(t)} b_{(t-1)} + \tilde{b}_{(t)} \\
s.t.\ A_{(0)} &= I, \quad b_{(0)} = 0
\end{aligned}
$$

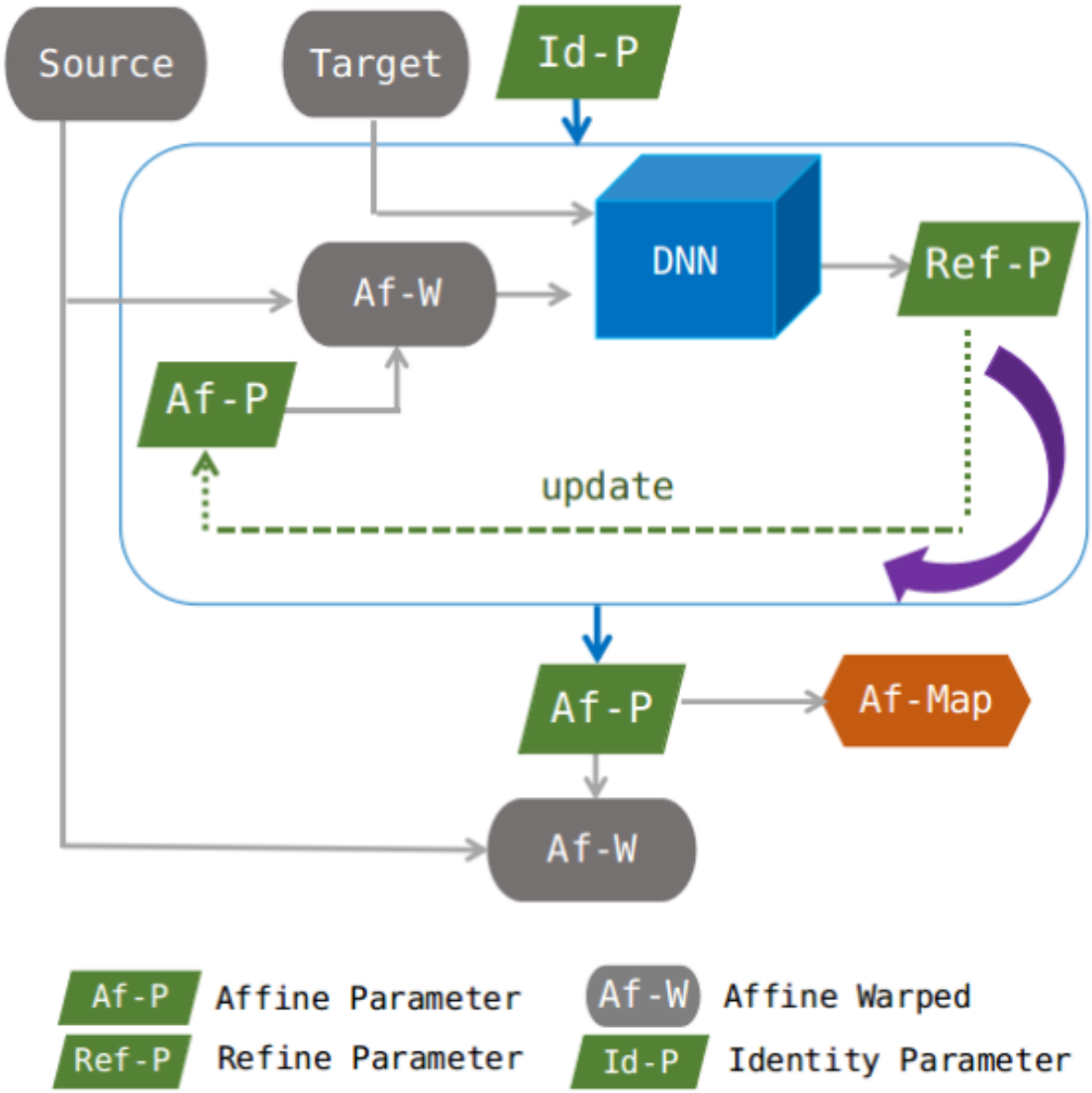

Figure 7: Multi-step affine network architecture (Copied from Shen et al. 2019) Multi-step affine network structure. As in a recurrent network, the parameters of the affine network are shared by all steps. At each step, the network outputs the parameters to refine the previously predicted affine transformation. I.e., the current estimate is obtained by composition (indicated by dashed line). The overall affine transformation is obtained at the last step.

where $\tilde{A}_{(t)}$ and $\tilde{b}_{(t)}$ is the output of the network and $A_{(t)}$ and $b_{(t)}$ the updated linear transformation (Shen et al. 2019). Finally, after the multi-step affine network the transformation is given by

$$\varphi_{(affine)}(x, \Gamma) = A_{(t_{final})}x + b_{(t_{final})}$$

The loss function for training has three parts. First a image similarity loss L_{a-sim}, a regularization loss L_{a-reg} and a symmetry regularization loss L_{a-sym} (Shen et al. 2019). They have chosen a multi-kernel LNCC loss which is a weighted sum of LNCCs with different window sizes.

$$L_{a-sim}(M,T,\Gamma) = \sum_i \omega_i LNCC_{s_i}(M \circ \varphi_{(affine)}, T)$$
$$s.t. \quad \varphi_{(affine)}(x,\Gamma) = Ax + b \quad and \quad \sum_i \omega_i = 1, \quad \omega_i \geq 0$$

The regularization loss L_{a-reg} penalizes deviations of the composed affine transform from the identity

$$L_{a-reg}(\Gamma) = \lambda_{ar} \left(\|A - I\|_F^2 + \|b\|_2^2 \right)$$

where $\|\cdot\|_F^2$ is the Frobenius norm and $\lambda_{ar} \geq 0$ is an epoch dependent weight factor that is large at the beginning to constrain large deformations and then gradually decaying to zero (Shen et al. 2019).
The symmetry loss L_{a-sym} encourages that the transformation computed from the moving image M to the target image T is the inverse of the transformation computed from the target image T to the moving image M (Shen et al. 2019). That means that

$$A^{tm}(Ax + b) + b^{tm} = x$$

where $A^{tm}\, b^{tm}$ is the affine transformation mapping the target image T to the moving image M. Thus both affine transformations (moving to target and target to moving) need to be computed. The loss is given by

$$L_{a-sym}(\Gamma, \Gamma^{tm}) = \lambda_{as} \left(\left\| A^{tm}A - I \right\|_F^2 + \left\| A^{tm}b + b^{tm} \right\|_2^2 \right)$$

where λ_{as} is a chosen constant (Shen et al. 2019). Hence the overall loss is

$$\begin{aligned} L_a(M,T,\Gamma,\Gamma^{tm}) = & L_{a-sim}(M,T,\Gamma) + L_{a-sim}(T,M,\Gamma^{tm}) + \\ & L_{a-reg}(\Gamma) + L_{a-reg}(\Gamma^{tm}) + \\ & L_{a-sym}(\Gamma,\Gamma^{tm}) \end{aligned}$$

The final affine mapping $\varphi_{(affine)}$ is then given to the vSVF registration.

4.3.2 vSVF

The non-linear registration uses a simpler form of LDDMM where the velocity field is temporal static (vSVF). The benefit is that it allowed them to explicitly control spatial smoothness. The deep network predicts the momentum which gets subsequently smoothed to obtain the velocity field, instead of a network that has to predict a smooth vector field directly (Shen et al. 2019). The architecture is given in Figure 8.

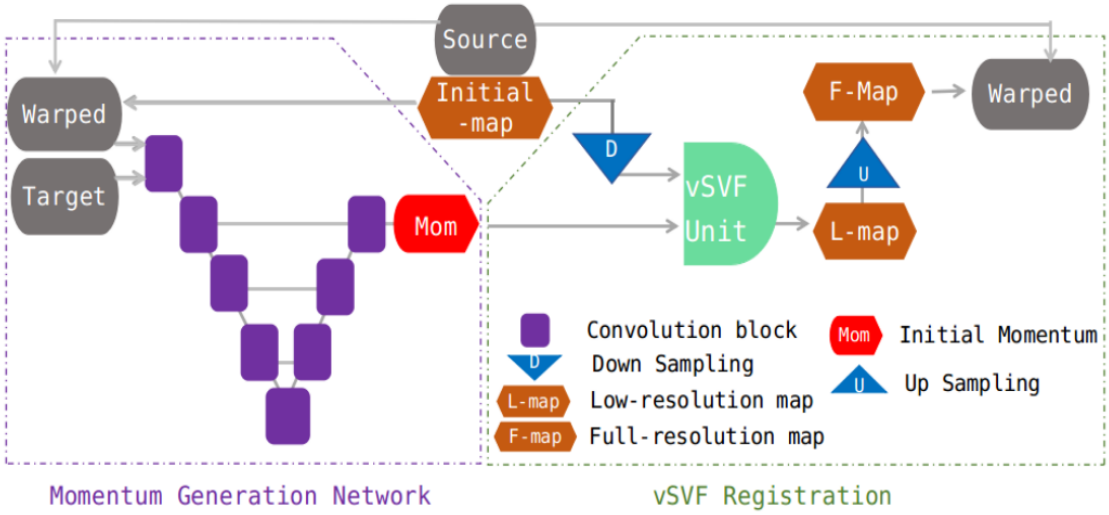

Figure 8: vSVF registration (Copied from Shen et al. 2019) vSVF registration framework illustration (one step), including the momentum generation network and the vSVF registration. The network outputs a low-resolution momentum. The momentum and the down-sampled initial map are input to the vSVF unit outputting a low-resolution transformation map, which is then up-sampled to full resolution before warping the source image.

The momentum generation network has a four-level U-net architecture with residual links (Shen et al. 2019). In order to reduce memory requirements, the network outputs a low resolution momentum which is achieved by removing the last decoder level. The vSVF unit operates also on low-resolution map and thus the initial map needs to be down sampled. The initial map can be the affine transformation or the map obtained from a previous vSVF step (Shen et al. 2019). The vSVF unit gets as input the low-resolution momentum and the initial map and outputs the final low-resolution map (Shen et al. 2019). Lastly this mapping is upsampled to get the full-resolution map. Like the affine network, the loss for the vSVF part of the network consists of a similarity loss L_{v-sim}, a regularization loss L_{v-reg} and a symmetry loss L_{v-sym} (Shen et al. 2019). They defined the similarity loss $L_{v-sim}(M, T, \varphi)$ as multi kernel LNCC. Further they regularized the velocity field as following

$$L_{v-reg}(m_0) = \lambda_{vr} \|v\|_L^2 = \lambda_{vr} < m_0, v_0 > \tag{7}$$

$$v_0 = Km_0 \tag{8}$$

where K is implemented as convolution with a multi-Gaussian kernel. Finally the symmetric loss was defined by

$$L_{v-sym}(\varphi, \varphi^{tm}) = \lambda_{vs} \left\| \varphi \cdot \varphi^{tm} - id \right\|_2^2$$

where id denotes the identity map and φ^{tm} is the mapping from the target image T to the moving image M (Shen et al. 2019). They define the complete loss as

$$\begin{aligned} L_v(M, T, \varphi, \varphi^{tm}, m_0, m_0^{tm}) = & L_{v-sim}(M, T, \varphi) + L_{v-sim}(T, M, \varphi^{tm}) + \\ & L_{v-reg}(m_0) + L_{v-reg}(m_0^{tm}) + \\ & L_{v-sym}(\varphi, \varphi^{tm}) \end{aligned}$$

For multi-step vSVF the loss changes to

$$\begin{aligned} & \sum_{\tau=1}^{T} L_v(M, T, \varphi_{(\tau)}, \varphi_{(\tau)}^{tm}, m_{0(\tau)}, m_{0(\tau)}^{tm}) \\ & \varphi_{(\tau)}(x, 0) = \varphi_{(\tau-1)}(x, 1) \\ & \varphi_{(\tau)}^{tm}(x, 0) = \varphi_{(\tau-1)}^{tm}(x, 1) \end{aligned}$$

which means that the initial map for the next step is just the final map of the previous step.

4.3.3 Results and Conclusion

In summary, Shen et al. 2019 introduced an end-to-end 3D image registration approach that consists of a multi-step affine network and a deformable registration network using a momentum-based SVF algorithm. Their model outputs a transformation map which includes an affine pre-registration and a vSVF non-parametric deformation in a single forward pass. Their experiments have shown that this approach achieves comparable and sometimes better performance to popular registration tools with a dramatically reduced computation time. Additionally, the deformation are regular and symmetric.

4.4 Region-specific Diffeomorphic Metric Mapping RDMM

The last paper can be seen as a fusion and generalization of the previous two works. Shen, Vialard, and Niethammer 2019 introduced RDMM which is a non-parametric approach that extends the traditional LDDMM approach such that spatio-temporal regularization is possible. Spatial varying regularization is desirable because there are different scales of motion in different regions of the images. Temporal varying regularization is needed such that the regularizer can move with the deformation. The paper has an extensive theoretical part covering the derivation of the RDMM approach and is not covered here. The focus is on a high level understanding and the deep learning framework.
They have shown that RDMM can assure diffeomorphic transformation in the continuum. This means that in practice it can happen that for example folding happens due to inaccuracies when discretizing the evolution equations and when discretizing the determinant of the Jacobian (Shen, Vialard, and Niethammer 2019). As before the velocity can be computed by the convolution with a kernel $v = K \star m$. Shen, Vialard, and Niethammer 2019 defines the kernel and the corresponding convolution as a multi-Gaussian kernel

$$v(x,t) = K(x,t) \star m = \sum_{i_0}^{N-1} w_i(x,t) K_{\sigma_i} \star (w_i(x,t)m), \quad w_i(x,t) \geq 1 \tag{9}$$

where K_{σ_i} is a pre-defined Gaussian kernel with $\sigma_0 \leq \sigma_1 \leq ... \leq \sigma_{N-1}$.
The spatio-temporal weights are specified at inital time $t = 0$ and are then advected via v(x,t). In order to get diffeomorphic transformations the initial weight $w(x,0)$ needs to be sufficiently smooth. Therefore the optimization is done over the initial *pre-weights* $h_i(x,0)$. They are related as

$$\begin{aligned} w_i(x,0) &= G_\sigma h_i(x,0) \\ s.t. \quad & h_i(x,0) \geq 0 \end{aligned} \tag{10}$$

where G_σ is a fixed Gaussian with small standard deviation (Shen, Vialard, and Niethammer 2019). In addition they constrained $\sum_{i=0}^{N-1} h_i^2(x,0)$ to locally sum to one. Thus they optimize over the momentum and the initial pre-weights $h_i(x,0)$. The spatio-temporal regularizer was regularized to encourage simpler transformations This was achieved through an OMT penalty and a range loss on the weights

$$Reg(\{h_i(x,0)\}, T) = \lambda_{OMT}(T) OMT(\{h_i(x,0)\}) + \lambda_{Range}(T) Range(\{h_i(x,0)\}$$

where λ_{OMT} and λ_{Range} are scale factors and T refers to the iteration/epoch (Shen, Vialard, and Niethammer 2019). The OMT loss is given by

$$\widehat{OMT}(\{h_i(x,0)\},) = \left|log\frac{\sigma_{N-1}}{\sigma_0}\right|^{-r} \sum_{i=0}^{N-1} h_i(x,0) \left|log\frac{\sigma_{N-1}}{\sigma_i}\right|^{r}$$

where r is a chosen power. The motivation behind this is that if we order the standard deviations $0 < \sigma_0 < \sigma_1 < ... < \sigma_{N-1}$ and the corresponding weights for the Gaussians in Eq. (9), we should assign higher values to weights with large corresponding standard deviation to get smooth transformations. Actually, they penalized the squares of the weights to make the penalty more consistent with a standard multi-Gaussian regularizer. The range loss is given by

$$Range = \|G_\sigma \star (h(x,0) - w_0)\|$$

where w_0 is the pre-defined initial weight (Shen, Vialard, and Niethammer 2019). Thus the loss penalizes differences between the initial weight from the pre-defined weight. Since it is difficult to jointly optimize over the momentum and the pre-weights at the beginning of the optimization/training, the pre-weights are constrained by the Range loss (Shen, Vialard, and Niethammer 2019). To solve the original model, the influence of the range penalty needs to diminish while the influence of the OMT term needs to increase during training (Shen et al. 2019). Hence λ_{OMT} and λ_{Range} are epoch dependent. Additionally, they added a symmetry loss as in ASVM to encourage symmetric consistency. Shen, Vialard, and Niethammer 2019 RDMM framework can be used with pre-defined regularizer (e.g. foreground and background), as registration model where the spatio-temporal regularizer is estimated (traditional optimization of the underlying equations) or as registration model where the spatio-temporal regularizer is obained via an end-to-end deep network. In the following the deep learning approach is discussed.

4.4.1 Learning framework

Shen, Vialard, and Niethammer 2019 uses a similar approach as ASVM. The architecture is shown in Figure 9.

The multi-step affine network is the same as in ASVM and predicts the affine mapping. This is forwarded to the non-parametric network and can be used as initial map. The non-parametric module is also similarly build as the ASVM but has a LDDMM or RDMM unit instead of a vSVF unit.

The non-parametric module allows for iterative refinement where the output of the previous iteration can be used as the initial map for the next iteration. The non-parametric module operates on low-resolution maps and map compositions (Shen, Vialard, and Niethammer 2019). The final mapping is obtained by upsampling (Shen, Vialard, and Niethammer 2019). The LDDMM unit uses a 3D U-net for momemtum prediction. The RDMM has another 3D U-net for pre-weights prediction. Therefore we have 2 networks

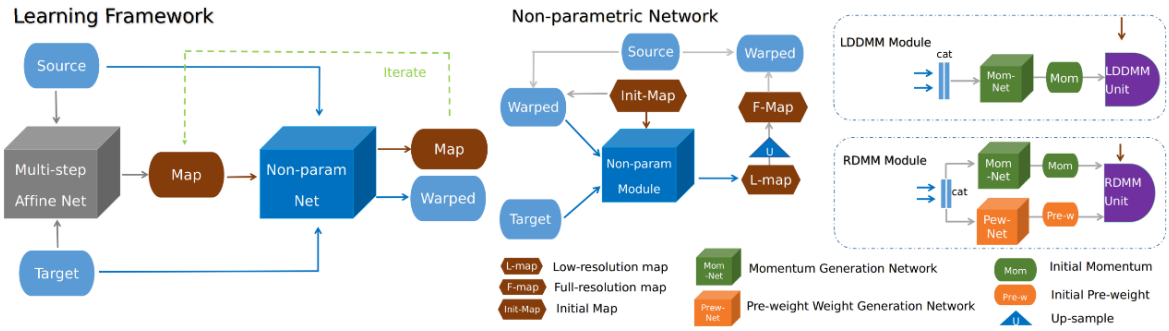

Figure 9: RDMM architecture (Copied from Shen, Vialard, and Niethammer 2019) A multi-step affine-network first predicts the affine transformation map followed by an iterable non-parametric registration to estimate the final transformation map. The LDDMM component uses one network to generate the initial momentum. RDMM also uses a second network to predict the initial regularizer pre-weights. The RDMM evolution equations are integrated at low-resolution (based on the predicted initial conditions) to save memory. The final transformation map is obtained via upsampling.

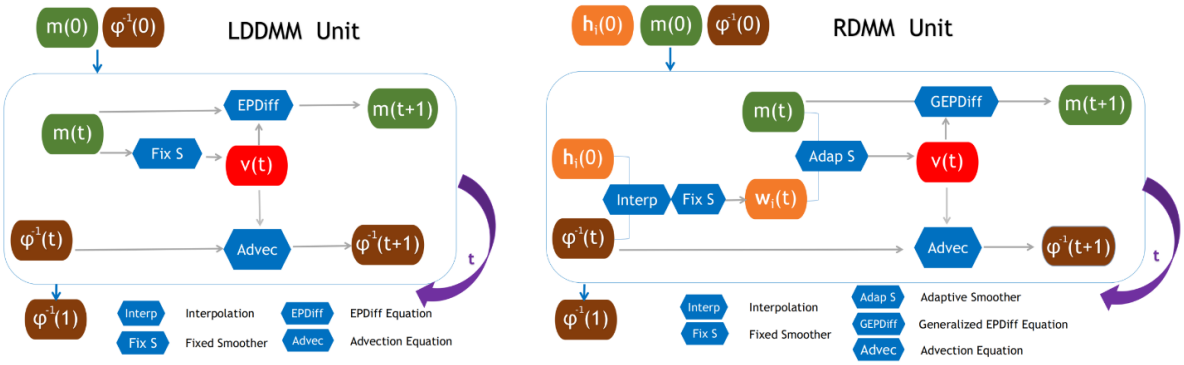

Figure 10: LDDMM (left) and RDMM (right) units (Copied from Shen, Vialard, and Niethammer 2019) The LDDMM unit iteratively solves the Euler-Poincaré equation given the initial momentum and initial map and outputs the final mapping. The velocity is obtained by a smoothing with a fixed multi-Gaussian kernel. The RDMM unit takes as input the initial momentum, the initial map and the pre-weights. Then it can map the pre-weights with the current mapping and interpolates the result. It follows a smoothing with a fixed Gaussian to get the initial weights (Eq. 10). Together with the momentum the velocity can be computed through adaptive Gaussian smoothing (Eq. 9). This allows to advect the current mapping to the next time step. Additionally a modified Euler-Poincaré equation is solved to get the next momentum. This procedure is iteratively applied until $t = 1$ is reached.

for LDDMM (affine + momentum) and 3 networks for RDMM (affine + momentum + pre-weights). Both units are shown in Figure 10.

4.4.2 Results and Conclusion

Shen, Vialard, and Niethammer 2019 introduced RDMM which allows the regularizer to be time and spatially-varying and is an extension of LDDMM registration. Diffeormophic transformations can be still computed with sufficiently regular regularizers. Further, their experiments showed that RDMM is flexible and fast at test time.

5 Voxelmorph

The first model (voxelmorph) of the voxelmorph framework was a CNN with U-net structure that used an extension of the NCC loss as the similarity measure and L2-regularization (Balakrishnan et al. 2018). This was extended with a segmentation loss that allowed to add segmentation data during training to improve the performance (Balakrishnan et al. 2019). The input to the network is a target image and a moving image and the output is a displacement field and the aligned moving image. Later an adapted version was commonly used that predicts a stationary velocity field which can be time integrated to get diffeomorphic mappings. Next a probabilistic network was designed. The input and output is the same as in the previous model. Additional segmentation data during training can still be provided. The difference is that the prediction network predicts the mean and variance of a random latent variable, which allows to sample a velocity field and therefore through time integration the deformation field. The velocity field was modeled to be stationary and thus diffeomorphic mappings can be generated. The next work proposed an atlas generation network that can align the atlases to a given image. The templates can be unconditional or conditioned on some attributes like age or sex. The input to the network is an attribute vector (potentially empty) and an image to which the generated template is aligned. The output of the network is the atlas and the corresponding mapping to a given image. The registration part uses an adapted voxelmorph model. Subsequently Synthmorph was introduced. It is a registration network that was trained without providing real images. Instead they generated in a sophisticated way synthetic images and corresponding labels and trained an adapted voxelmorph model. They achieved a registration framework that is robust to different modalities. Finally, HyperMorph was proposed. It is a hypernetwork that approximates a landscape of registration networks for a range of hyperparameter values, by learning a continuous function of hyperparameter. This enables for for fast hyperparameter tuning at test-time. The input to the network are the loss hyperparameters and the output is the weights for the registration network that allows to register two images. They used an adapted voxelmorph model for registration. Also, segmented data during training is supported.

5.1 VoxelMorph

The first two papers *An Unsupervised Learning Model for Deformable Medical Image Registration (Balakrishnan et al. 2018)* and *VoxelMorph: A Learning Framework for Deformable Medical Image Registration (Balakrishnan et al. 2019)* introduces a deep neural network approach for non-rigid pairwise image registration. The transformation φ is characterized by a displacement field u

$$\varphi = Id + u$$

where Id is the identity transform. The network is a CNN with a U-net architecture that can be seen in Figure 11. It predicts the displacement mapping given a moving image M and a target image T and is trained in a unsupervised fashion.

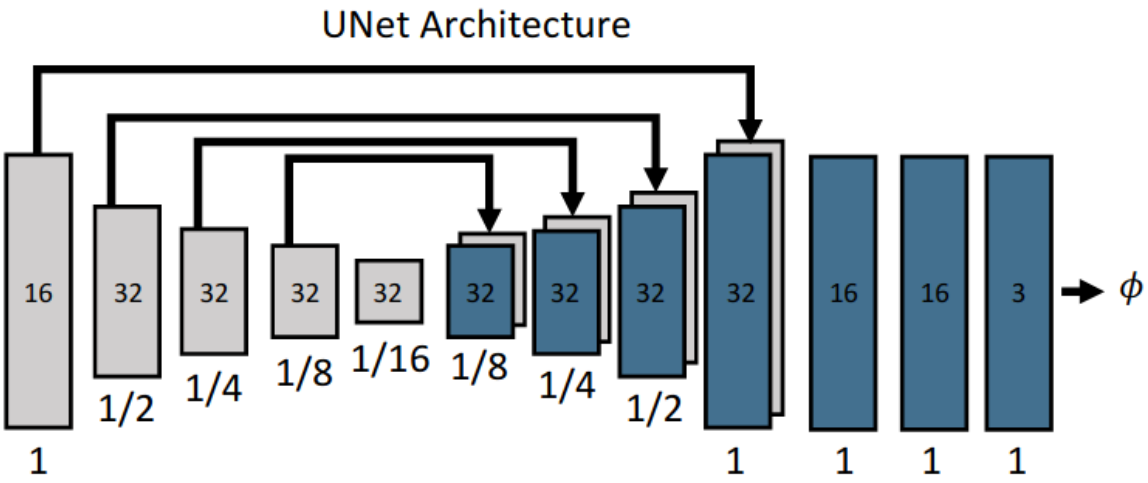

Figure 11: Voxelmorph CNN architecture (Copied from Balakrishnan et al. 2019) Each rectangle represents a 3D volume, generated from the preceding volume using a 3D convolutional network layer. The spatial resolution of each volume with respect to the input volume is printed underneath. In the decoder, they use several 32-filter convolutions, each followed by an upsampling layer, to bring the volume back to full resolution. Arrows represent skip connections, which concatenate encoder and decoder features. The full-resolution volume is further refined using several convolutions.

When warping the moving image M, a (subpixel) voxel location $p' = p + u(p)$ in M is computed (Balakrishnan et al. 2019). Since image values are only defined at integer locations, Balakrishnan et al. 2018 used linear interpolation of the values at the corresponding eight neighboring voxels

$$m \circ \varphi(p) = \sum_{q \in \mathrm{Z}(p')} m(q) \prod_{d \in \{x,y,u\}} (1 - |p'_d - q_d|)$$

where $\mathrm{Z}(p')$ are the voxel neighbors of p' and d iterates over the dimensions of Ω. This allowed them to backpropagate errors during optimization because gradients or subgradi-

ents (0 for gradient of absolute value at 0) can be computed. The loss function contains an image similarity measure and a smoothing penalty

$$L_{us}(T, M, \varphi) = L_{sim} + \lambda L_{smooth}(\varphi)$$

where λ is a regularization parameter. Balakrishnan et al. 2019 tried the MSE loss and the local cross-correlation of T and $M \circ \varphi$. The later is an extension of the cross correlation measure defined in section 2.1. It is given by

$$LCC(T, M, \varphi) = \sum_{p \in \Omega} \frac{\left(\sum_{p_i} \left(T(p_i) - \hat{T}(p)\right) \left([M \circ \varphi](p_i) - \left[\hat{M} \circ \varphi\right](p)\right)\right)^2}{\left(\sum_{p_i} \left(T(p_i) - \hat{T}(p)\right)^2\right) \left(\left(\sum_{p_i} [M \circ \varphi](p_i) - \left[\hat{M} \circ \varphi\right](p)\right)^2\right)}$$

where $\hat{T}(p)$ and $\left[\hat{M} \circ \varphi\right](p)$ denote local mean intensity images (Balakrishnan et al. 2018)

$$\hat{T}(p) = \frac{1}{n^3} \sum_{p_i} T(p_i)$$

where p_i iterates over a n^3 volume around p. They used $p = 9$. In contrast to MSE the local cross-correlation is more robust to intensity variations between images. Since a higher LCC yields a better alignment, the negative LCC was used for the minimization of the similarity loss $L_{sim}(T, M, \varphi) = -LCC(T, M, \varphi)$ (Balakrishnan et al. 2019).
Smooth transformations to a certain degree are desirable. This was encouraged through the use of the regularization loss $L_{smooth}(\varphi)$. They used L2 regularization

$$L_{smooth}(\varphi) = \sum_{p \in \Omega} \|\nabla u(p)\|^2$$

where the spatial gradients are approximated using neighboring voxels. Note that the regularization loss operates on the displacement field whereas the similarity loss operates on the target image T and the warped moving image M.
Additionally they added the possibility to provide auxiliary segmentations available in training. This should help the network to learn better registration. Note that during testing no auxiliary information is needed. Additionally, partially segmented images are supported and thus no fully segmented image is needed. The overview of the complete framework can be seen in Figure 5.
A region in the target image T and the warped moving image M corresponding the the same anatomical structure should overlap well. Therefore an additional loss was introduced to encourage this behaviour. The overlap was quantified through the use of the Dice score

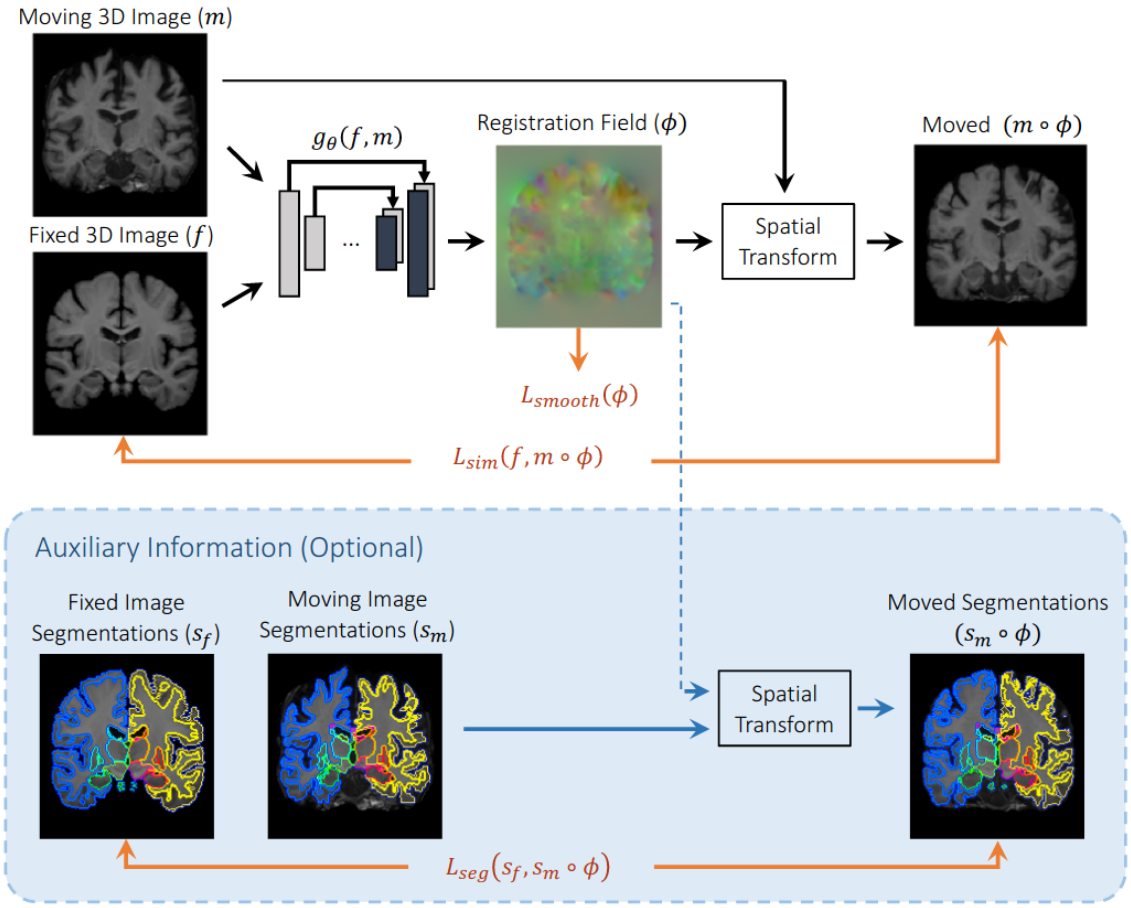

Figure 12: Voxelmorph overview (Copied from Balakrishnan et al. 2019) The displacement field for registering a moving image M to a target image T is predicted. During training, M is warped with $\varphi = \phi$ using a spatial transformer funtion. Optionally, auxiliary information can be leveraged during training (blue box).

$$Dice(s_T^k, s_M^k \circ \varphi) = 2\frac{|s_T^k \cap (s_M^k \circ \varphi)|}{|s_T^k| + |s_M^k \circ \varphi|}$$

where s_T^k and $s_M^k \circ \varphi$ are the voxels of the structure k for the target T and the warped moving image $M \circ \varphi$ (Balakrishnan et al. 2019). A Dice score of 1 indicates that the regions matches perfectly, and a score of 0 indicates no overlap (Balakrishnan et al. 2019). Therefore they defined the segmentation loss as:

$$L_{seg}(s_T, s_M \circ \varphi) = -\frac{1}{K}\sum_{k=1}^{K} Dice(s_T^k, s_M^k \circ \varphi)$$

Note that the segmentation loss operates on the segmentation of the target image T and the warped segmentation of the moving image M. Because anatomical labels are categorical they could not use a naive implementation of linear interpolation to compute the label of $s_M^k \circ \varphi$. Therefore s_T and s_M was designed as image volumes with K channels, where each channel is a binary mask specifying the spatial domain of a particular structure

(Balakrishnan et al. 2019). Further $s_M^k \circ \varphi$ was computed by spatially transforming each channel of s_M using linear interpolation. Then the dice score could be computed by multiplication and addition of s_T and $s_M^k \circ \varphi$ for the numerator and denominator respectively (Balakrishnan et al. 2019).
Combined with the unsupervised loss the total loss L_a is given by

$$L_a(T, M, s_T, s_M, \varphi) = L_{us}(T, M, \varphi) + \gamma L_{seg}(s_T, s_M \circ \varphi)$$

where γ is a regularization parameter.

5.1.1 Results and Conclusion

In summary, the network predicts a displacement field given the target image T and moving image M. The loss function for training consisted of a similarity measure and a displacement field regularization. Further, (partially) segmented training data can be used to train the network. Therefore an additional loss for the segmentation data was introduced. The training was unsupervised. Balakrishnan et al. 2019 have shown that adding the segmentation loss leads to significant improvements. Further, they have shown that the dataset does not have to be large. Already 100 samples could achieve state-of-the-art performance. Additionally, the network can produce fast predictions.

5.2 Unsupervised Learning for Fast Probabilistic Diffeomorphic Registration

The next two papers *Unsupervised Learning for Fast Probabilistic Diffeomorphic Registration* (Adrian V. Dalca et al. 2018) and *Unsupervised Learning of Probabilistic Diffeomorphic Registration for Images and Surfaces* (Adrian V. Dalca et al. 2019b) introduced a probabilistic generative model which can offer uncertainty quantification and providing diffeomorphic mappings. Additionally the learning can be done unsupervised. They choose to work with stationary velocity field representation to get diffeomorphic transformations. Therefore the deformation field is defined by

$$\frac{\partial \varphi^{(t)}}{\partial t} = v(\varphi^{(t)}$$

where $\varphi^{(0)} = Id$ is the identity transformation and t is time (Balakrishnan et al. 2018). To obtain the final registration field, the stationary velocity field needs to be integrated over $t = [0, 1]$. They used *scaling and squaring* proposed by Arsigny et al. 2016. To predict the diffeomorphic transformation a variational inference approach was used. An overview of the method can be found in Figure 13.
The latent variable z parametrizes the desired transformation function φ_z. It is a random variable and the goal is to estimate the posterior registration probability of z given the

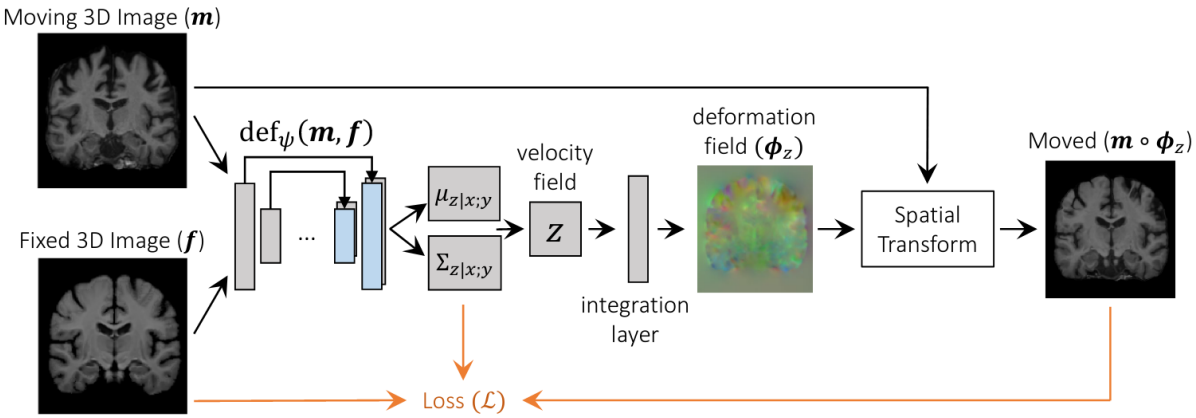

Figure 13: Overview of probabilistic architecture (Copied from Adrian V. Dalca et al. 2019b) The first part of the network, $def_\psi(M, T)$ takes the input images (target image $T = f$ and moving image $M = m$) and outputs the approximate posterior probability parameters representing the velocity field mean $\mu_{z|M;T}$, and variance $\Sigma_{z|M;T}$. A velocity field z is sampled and transformed to a diffeomorphic deformation field φ_z using squaring and scaling integration layers. Finally, a spatial transformation warps M to obtain $M \circ \varphi$.

target image T and moving image M $p(z|M;T)$. This allows to get the most likely registration field φ_z for a new registration pair (T, M) via MAP estimation. In their work they modeled z as a stationary velocity field to get diffeomorphic transformations. The prior probability is modeled as multivariate normal distribution (Adrian V. Dalca et al. 2019b)

$$p(z) = \mathcal{N}(z, 0; \Sigma_z)$$

$\Sigma_z^{-1} = \Lambda_z = \lambda L$ where Λ_z is a precision matrix and λ denotes a parameter controlling the scale of the velocity field z (Adrian V. Dalca et al. 2019b). This encourages spatial smoothness of the velocity field. $L = D - A$ is defined as the Laplacian of a neighborhood graph defined on the voxel grid where D is the graph degree matrix and A is a voxel neighbourhood adjacency matrix (Adrian V. Dalca et al. 2019b). The degree matrix is a diagonal matrix that stores the degree of each vertex (i.e. the number of edges attached to each vertex) and the adjacency matrix is a square matrix where the element of the matrix indicate whether pairs of vertices are connected or not. Computing the posterior probability $p(z|M;T)$ is intractable. Therefore they used a variational approach and introduced an approximate posterior probability $q_\psi(z|T;M)$ parametrized by ψ. Then the Kullback-Leibler divergence was minimized which yielded the negative variational lower bound.

$$\begin{aligned}
&\min_{\psi} KL\left[q_\psi(z \mid T; M) \| p(z \mid T; M)\right] \\
=&\min_{\psi} KL\left[q_\psi(z \mid T; M) \| p(z)\right] - \mathbb{E}_q\left[log\ p(T \mid z; M)\right] + const
\end{aligned} \tag{11}$$

It is called variational lower bound because it is a lower bound for the log evidence

$$\sum_{i=1}^{N} log\ p(T^{(i)}; M^{(i)})$$

where N is the number of samples. The approximate posterior $q_\psi(z \mid T; M)$ is modeled as multivariate normal (Adrian V. Dalca et al. 2019b)

$$q_\psi(z \mid T; M) = \mathcal{N}(z; \mu_{z|T,M}, \Sigma_{z|T,M})$$

where $\Sigma_{z|T,M}$ is diagonal. Adrian V. Dalca et al. 2019b showed that a non-diagonal matrix does not provide a significant improvement. $\mu_{z|T,M}$ and $\Sigma_{z|T,M}$ can be interpreted as voxel-wise mean and variance respectively (Adrian V. Dalca et al. 2019b). The probability of the target image T given the latent variable z and the moving image M is modeled as noisy observation of the warped image M (Adrian V. Dalca et al. 2019b).

$$p(T \mid z; M) = \mathcal{N}(T; M \circ \varphi, \sigma_I^2 \mathcal{I})$$

Therefore every term in the variational lower bound is modeled as multivariate Gaussian. To estimate the mean and the variance of the approximate posterior $q_\psi(z \mid T; M)$ a CNN was used with a U-net structrure. It takes as input the target image T and the moving image M and outputs the mean and variance of the approximate posterior $\mu_{z|T,M}$ and $\Sigma_{z|T,M}$. The network consists of a convolutional layer with 32 filter, four downsampling layers with 64 convolutional filters and a stride of two and *three* upsampling convolutional layers with 64 filters. Adrian V. Dalca et al. 2018 used only three upsampling steps to fit all operations in current GPU card memory. Therefore the velocity field is predicted at every two voxels. The loss for the neural network is the negative variational lower bound from Eq. 11

$$\begin{aligned}
L(\psi; T, M) =& - \mathbb{E}_q\left[log\ p(T \mid z; M)\right] + KL\left[q_\psi(z \mid T; M) \| p(z)\right] \\
&= \frac{1}{2\sigma^2 K} \sum_k \left\| T - M \circ \varphi_{z_k} \right\|^2 + \\
&\frac{1}{2}\left[tr\left(\lambda D \Sigma_{z|T,M} - log \Sigma_{z|T,M}\right) + \mu_{z|T,M}^T \Lambda_z \mu_{z|T,M}\right]
\end{aligned} \tag{12}$$

The derivation of this formula is provided in the supplementary material of Adrian V. Dalca et al. 2019b. K is the number of samples used to approximate the expectation. The first part encourages the target image T and the warped moving image M to be similar and the second term encourages the approximate posterior to be close the the prior $p(z)$. They used σ^2 and λ as fixed hyper-parameters and set K=1.
After the mean and the variance $\mu_{z|T,M}$ and $\Sigma_{z|T,M}$ of the approximate posterior have been predicted, a velocity field needs to be sampled $z_k \sim \mathcal{N}(\mu_{z|T,M}, \Sigma_{z|T,M})$. Adrian V. Dalca et al. 2018 used the "re-parametrization trick" for a Gaussian

$$z_k = \mu_{z|T,M} + \sqrt{\Sigma_{z|T,M}} r$$

where r is a sample from a standard normal distribution $r \sim \mathcal{N}(0, I)$. After z_k is sampled a deformation field φ_{z_k} can be computed by scaling and squaring. Finally the moving image M can be mapped on to the target image T.
In summary, the model takes as input the target image T and the moving image M and computes the mean and variance of the approximate posterior $\mu_{z|T,M}$ and $\Sigma_{z|T,M}$. This allows to sample a velocity field from the approximate posterior $z_k \sim \mathcal{N}(\mu_{z|T,M}, \Sigma_{z|T,M})$ and therefore to compute a diffeomorphic mapping φ_{z_k} that warps the moving image M. Thus the network outputs $\mu_{z|T,M}$, $\Sigma_{z|T,M}$ and $M \circ \varphi_{z_k}$. For new image pairs, the most likely velocity field $\hat{z}_k$ is obtained by (Adrian V. Dalca et al. 2019b)

$$\hat{z}_k = \arg\max_{z_k} p(z_k \mid T, M) = \mu_{z|T;M}$$

Therefore the images are feeded to the neural network which outputs the desired mean $\mu_{z|T;M}$. This can be used to compute the diffeomorphic mapping through integration (scaling and squaring). At test time the covarinace is not used (Adrian V. Dalca et al. 2019b).
Adrian V. Dalca et al. 2019b extended the model to support additional segmentation information during training. As Balakrishnan et al. 2019 has shown this can improve the registration result. Adrian V. Dalca et al. 2019b focused on the case where one anatomical structure is segmented. In order to incorporate segmentation the Kullback-Leibler divergence and therefore the negative variational lower bound is given by (Adrian V. Dalca et al. 2019b)

$$\min_{\psi} KL\left[q_\psi(z \mid T; M) \| p(z \mid T, s_T; M, s_M)\right] \tag{13}$$

$$= \min_{\psi} KL\left[q_\psi(z \mid T; M) \| p(z)\right] - \mathbb{E}_q\left[log\ p(T \mid z; M)\right] - \mathbb{E}_q\left[log\ p(s_T \mid z; s_M)\right] \tag{14}$$

where s_T and s_M represent the N surface coordinates of the anatomical structure for the target image T and the moving image M. They modeled $p(s_T \mid z; s_M)$ as multivariate Gaussian

$$p(s_T \mid z; s_M) = \mathcal{N}(s_T; s_M \circ \varphi_z, \sigma_s^2 \mathcal{I})$$

The total loss is given by

$$L(\psi; T, s_T M, s_M) = \frac{1}{2\sigma_I^2 K} \sum_k \|T - M \circ \varphi_{z_k}\|^2 + \tag{15}$$

$$\frac{1}{2\sigma_s^2 K} \sum_k \|s_T - s_M \circ \varphi_{z_k}\|^2 + \tag{16}$$

$$\frac{1}{2} \left[tr\left(\lambda D \Sigma_{z|T,M} - log \Sigma_{z|T,M}\right) + \mu_{z|T,M}^T \Lambda_z \mu_{z|T,M} \right] \tag{17}$$

where the additional second term encourages the deformation φ_{z_k} to warp the moving surface s_M close to the fixed surface s_T (Adrian V. Dalca et al. 2019b). To compute this term they used a surface distance function $sd(x, s)$ that returns for a location x the Euclidian distance to the closest surface point in s. Since $\sum_n sd(a[n], b) \neq \sum sd(b[n], a)$ due to potential asymmetries in the surfaces, the distance $\|s_T - s_M \circ \varphi_{z_k}\|^2$ was approximated by computing $sd(\cdot, \cdot)$ in both directions (Adrian V. Dalca et al. 2019b)

$$\|s_T - s_M \circ \varphi_{z_k}\|^2 \approx \frac{1}{2} \sum_n sd(s_T[n] \circ \varphi_z^{-1}, s_M) + \frac{1}{2} \sum_n sd(s_M[n] \circ \varphi_z, s_T)$$

5.2.1 Results and Conclusion

Adrian V. Dalca et al. 2018 proposed a probabilistic model for diffeomorphic registration. They have shown that the simplifying diagonal approximation to the velocity covariance $\Sigma_{z|T,M}$ adds voxel-independent noise to every velocity field sample z_k, but the resulting deformation fields are well behaved because of the smoothing priors and diffeomorphic representation. Further diffeomorphic mappings are guaranteed and the network is fast at test time. Additionally, providing segmentation information during training helps the network to achieve better performance.

5.3 Learning Conditional Deformable Templates with Convolutional Networks

Adrian V Dalca et al. 2019a presented a learning framework for building atlases. It is possible to create universal and conditional templates with a probabilistic model and efficient alignment of images to these atlases. There exist already different methods but

often they need a costly global optimization. Additionally, a single template may be inadequate at capturing the variability in a large dataset. Existing methods often solve this problem by dividing the data into subgroups, usually along a single attribute, and computing separate templates for each group (Adrian V Dalca et al. 2019a). Thus each template is created with fewer images, which leads to sup-optimal templates (Adrian V Dalca et al. 2019a). They propose a learning-based approach that can compute on-demand conditional deformable templates. The conditioning can be done with multiple attributes, continuous or discrete (Adrian V Dalca et al. 2019a). An overview of the model can be seen in Figure 14.

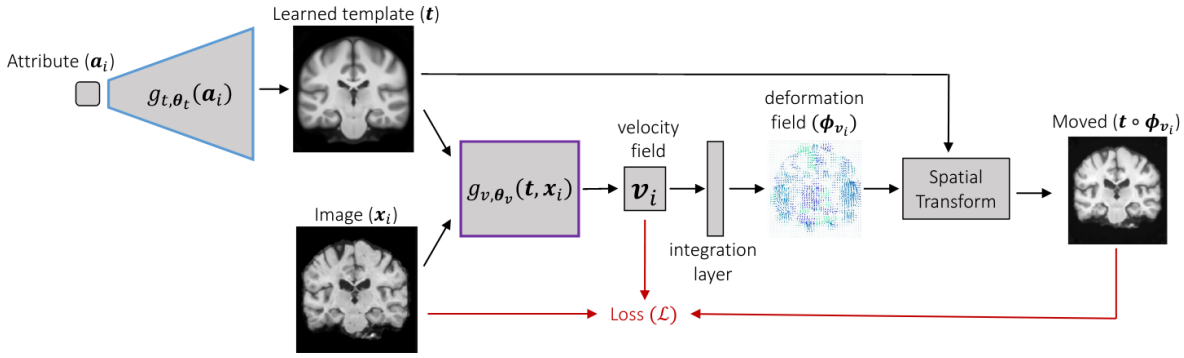

Figure 14: Overview of template creation model (Copied from Adrian V Dalca et al. 2019a) The network takes as input an image and an optional attribute vector. The upper network $g_{t,\theta_i}(\cdot)$ outputs a template, which is then registered with the input image by the second network $g_{v,\theta_v}(\cdot)$. The loss function, derived from the negative log likelihood of the generative model, leverages the template warped into $t \circ \varphi_{v_i}$.

The model takes as input an attribute vector to condition the template on (this could be empty for global templates) and an image x_i. It outputs the template and the corresponding mapping to align the template t to the image x_i.

The first network takes as input a vector of attributes and outputs a (conditional) template. Class attributes can be encoded as one-hot representations and continuous attributes as scalars (Adrian V Dalca et al. 2019a). For an unconditioned atlas no attribute vector can be provided. They defined the network by $g_{t,\theta_t}(a_i) = t$. Adrian V Dalca et al. 2019a implemented two versions, depending on whether an unconditional or conditional template is estimated. The conditional version consists of a decoder that includes a fully connected layer, followed by several blocks of upsampling, convolutional, and ReLu activation layers (Adrian V Dalca et al. 2019a). The second, unconditional version has no inputs and simply consists of a learnable parameter at each pixel (Adrian V Dalca et al. 2019a).

The second network takes as input a template and an image x_i and aligns the template to the image x_i. The architecture is similar to the architecture in subsection 5.1. However the terminology for this new setting is different. Previously a target image T and a moving

image M was feeded to the voxelmorph network and the warped moving image M was returned. Here a template t and an image x_i is feeded to the network. The network still predicts a stationary velocity v_i which can be integrated to get the warped template $t \circ \varphi_{v_i}$ where φ_{v_i} follows a displacement formulation $\varphi_{v_i} = Id + u_v$. The network is defined as $g_{v,\theta_v}(t, x_i) = v_i$. The set of all stationary velocity fields of the training set is given by $V = \{v_i\}$

The optimization should learn the parameter $\hat{\theta} = \{\hat{\theta}_t, \hat{\theta}_v\}$ defining the networks and was done using stochastic gradient descent. The loss function is given by (Adrian V Dalca et al. 2019a)

$$\begin{aligned} L(\theta_t, \theta_v, v_i, x_i, a_i) &= -\, log\ p_\theta(x_i \mid v_i, a_i) - log\ p_\theta(v_i) \\ &= -\frac{1}{2\sigma^2} \left\| x_i - g_{t,\theta_t}(a_i) \circ \varphi_{v_i} \right\|^2 \\ &\quad - \gamma \left\| \overline{u} \right\|^2 - \lambda_d \frac{d}{2} \sum_i \left\| u_i \right\|^2 + \frac{\lambda_a}{2} \sum_i \left\| \nabla u_i \right\|^2 \end{aligned} \tag{18}$$

The last three terms follow from the assumption that the probability of the set of stationary velocity field is given by

$$p_\theta(V) \propto exp\{-\gamma \left\| \overline{u} \right\|^2\} \prod_i \mathcal{N}(u_i, 0, \Sigma_u)$$

where $\overline{u} = 1/n \sum_i u_i$ is the average displacement and $\Sigma_u^{-1} = L$ where $L = \lambda_d D - \lambda_a C$ is a relaxation of the Laplacian of a neighborhood graph with graph degree matrix D and the pixel neighbourhood adjacency matrix C (Adrian V Dalca et al. 2019a). Therefore the log velocity prior is given by

$$log\ p_\theta(V) = -\gamma \left\| \overline{u} \right\|^2 - \lambda_d \frac{d}{2} \sum_i \left\| u_i \right\|^2 + \frac{\lambda_a}{2} \sum_i \left\| \nabla u_i \right\|^2 + const$$

where d is the neighbourhood degree (Adrian V Dalca et al. 2019a). The first term encourages a small average deformation across the dataset, encouraging a central, unbiased template (Adrian V Dalca et al. 2019a). They approximated the average deformation $\overline{u}$ with a weighted running average $\overline{u} \sim \sum_{k=K-c}^{K} u_k$ where u_k is the displacemnet at iteration k, K is the current iteration and c was set to 100. The second and third terms encourage templates that minimize deformation size and smoothness, respectively, and γ, λ_d and λ_a are hyperparameters (Adrian V Dalca et al. 2019a).

The first term of the network loss in Eq. 18 follows from the assumuption

$$p(x_i \mid v_i, a_i) = \mathcal{N}(x_i; g_{t,\theta_t}(a_i) \circ \varphi_{v_i}, \sigma^2 \mathcal{I})$$

where σ^2 represents the additive image noise (Adrian V Dalca et al. 2019a).

5.3.1 Results and Conclusion

In summary, Adrian V Dalca et al. 2019a introduced a model that can predict unconditional or conditional atlases and the corresponding mapping to align these atlases to an image given the attribute vector (for conditional atlas) and the image. This method is much faster at test time than traditional atlas generation methods. Additionally it can generate templates conditioned on a set of attributes. Further they have shown that with about a day of training, they can produce unconditional atlases similar in quality and utility to a widely used atlas that took weeks to produce.

5.4 Learning image registration without images

Hoffmann et al. 2020 introduced a learning strategy for contrast-invariant image registration without requiring imaging data. They tried to solve the problem that learning-based techniques can only register images with similar modality and geometric contrast. In order to train a registration network, they built a generative model for creating training data with a wide range of variability in contrast and geometric shapes. This should help the registration to generalize well. The model can be split into two parts. The first part creates target image T, moving image M, segmentation for target image s_T and segmentation for moving image s_M. The second part is a neural network that tries to align the moving image segmentation to the target image segmentation. They used the voxelmorph architecture presented in subsection 5.1 with an adapted loss function. Additionaly, the prediction of the network is a stationary velocity field that can be integrated to obtain diffeomorphic mappings. An overview of the method can be seen in Figure 15.

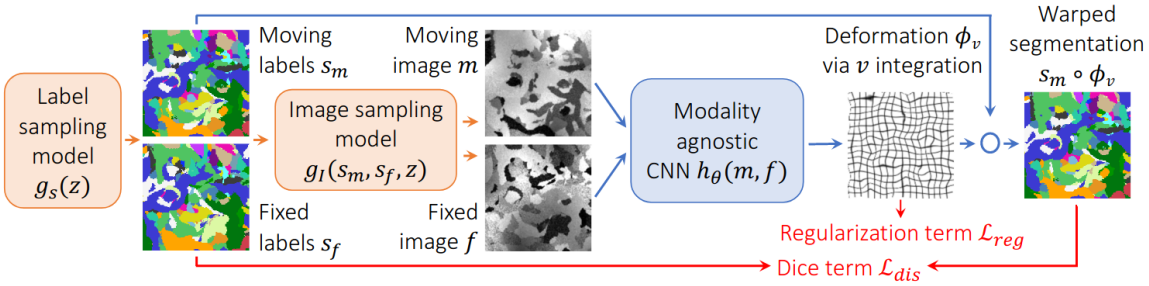

Figure 15: Overview of SynthMorph (Copied from Hoffmann et al. 2020) A pair of 3D label maps $\{s_m = s_m, s_T = s_f\}$ is synthesized and then the corresponding 3D images $\{M, T\}$ based on a generative model that covers a wide range of shapes an contrasts. The images are used to train a U-net style network and the label maps are used in the loss to measure alignment

Hoffmann et al. 2020 achieved contrast-invariance and robustness to anatomical variability by requiring no real training data, but instead synthesizing arbitrary contrasts and shapes through a generative process. They started from scratch, synthesizing first two 3D paired label maps $\{s_M, s_T\} = g_s(z)$ given a random seed z. Then two 3D intensity

volumes $\{M, T\} = g_i(s_M, s_T, z)$ were synthesized based on the maps $\{s_M, s_T\}$ and seed z (Hoffmann et al. 2020). This can be used to train the network on arbitrary contrasts and shapes at each iteration and therefore removing the dependency on specific modalities and shapes of previous learning-based methods (Hoffmann et al. 2020). Additionally, this procedures allows to use a similarity loss that measures label overlap independent of image contrast. They used dice metric for the dissimilarity measure and L2 regularization for the velocity field

$$L(\varphi, s_M, s_T,) = -\frac{2}{J}\sum_{j=1}^{J}\frac{\mid (s_M^j \circ \varphi) \otimes s_T^j \mid}{\mid (s_M^j \circ \varphi) \oplus s_T^j \mid} + \lambda\frac{1}{2}\|\nabla u\|^2$$

where J is the number of lables, s^j represents the one-hot encoded label $j \in \{1, 2, ..., J\}$ of label map s and $\otimes$ and $\oplus$ denote voxel-wise multiplication and addition, respectively.

5.4.1 Generating data

The following paragraph is mainly copied from Hoffmann et al. 2020 because it was written in a compact and well understandable form.
In order to get data, first label maps with J labels are created. This was done by drawing J smoothly varying noise images $p_j (j \in \{1, 2, ..., J\})$ by sampling voxels from a standard distribution at reduced resolution r_p and upsampling to full size. Then each image p_j is warped with a random smooth deformation φ_j (described below) to obtain $\tilde{p_j} = p_j \circ \varphi_i$ (Hoffmann et al. 2020). Third an input label map s is created by assigning, for each voxel k of s, the label j among the images $\tilde{p_j}$ that has the highest intensity (Hoffmann et al. 2020). An illustration of this procedure can be found in Figure 16.

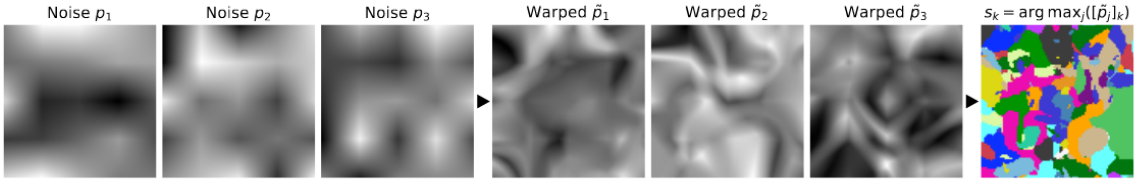

Figure 16: Generation of label maps (Copied from Hoffmann et al. 2020) A set of smoothly varying 3D noise images pj $(j \in \{1, 2, ..., J\})$ is sampled from a standard distribution, then warped by random deformation fields φ_j to cover a range of spatial scales and shapes. A label map s is synthesized from the warped images $\tilde{p_j} = p_j \circ \varphi_j$ for each voxel k of s, by determining for which $\tilde{p_j}$ that voxel has the highest intensity,and assign the corresponding label j. The example uses J=26.

Given a selected label map s, Hoffmann et al. 2020 generated two new label maps by deforming s with two random smooth diffeomorphic transformation φ_M and φ_T (described below) using nearest-neighbor interpolation. This generated the segmentation of the moving $s_M = s \circ \varphi_M$ and target images $s_T = s \circ \varphi_T$. If there were anatomical segmentations

available, they selected two separate subjects at random and also deformed the segmentations with a diffeomorphic transformation. Next the gray-scale images $\{M, T\}$ were generated. Given a segmentation map s, they draw the intensities of all image voxels that are associated with label j as independent samples from the normal distribution $\mathcal{N}(\mu_j, \sigma_j^2)$. Hoffmann et al. 2020 sample the mean μ_j and standard deviation σ_j for each label from continuous distributions $\mathcal{U}(a_\mu, b_\mu)$ and $\mathcal{U}(a_\sigma, b_\sigma)$, respectively, where a_μ, b_μ, a_σ and b_σ are hyperparameters. To simulate partial volume effects they convolved the image using an anisotropic Gaussian kernel $K(\sigma_i = 1, 2, 3)$ where $\sigma_i = 1, 2, 3 \sim \mathcal{U}(0, b_K)$. Additionally they corrupted the image with a spatially varying intensity-bias field B. The voxels of the bias field B were sampled from a normal distribution $\mathcal{N}(0, \sigma_B^2)$ at reduced resolution r_B, where $\sigma_B^2 \sim \mathcal{U}(0, b_B)$. Then the bias field was upsampled to full size, and the exponential of each voxel was taken to yield non-negative values. Finally the bias field is applied by element-wise multiplication. The final images M and T were obtained through min-max normalization and contrast augmentation through exponentiation, using the normally distributed parameter $\gamma \sim \mathcal{N}(0, \sigma_\gamma^2)$ such that $M = \tilde{M}^{exp(\gamma)}$ and $T = \tilde{T}^{exp(\gamma)}$ where $\tilde{\cdot}$ is the normalized image.

The random transformations were obtained by integrating random static velocity fields v_j. The voxels of v_j were drawn independently from a normal distribution $\mathcal{N}(0, \sigma_j^2)$ at reduces resolution r_p, where $\sigma_j \sim \mathcal{U}(0, b_p)$ is sampled uniformly and each velocity field is upsampled to full size (Hoffmann et al. 2020). The same process is used for φ_M and φ_T with hyperparameters r_v and b_v. In total there are 12 hyperparamters where three are resolution parameters $\{r_p, r_v, r_B\}$, eight defining the boundaries of uniform distributions $\{b_p, a_\mu, b_\mu, a_\sigma, b_\sigma, b_B, b_K, b_v\}$ and one parameter for the variance of a Gaussian σ_γ.

5.4.2 Results and Conclusion

Hoffmann et al. 2020 have shown through experiments that SynthMorph achieves registration accuracy matching or outperforming classical methods and is significantly faster. If brain labels are available the result can be further improved. Additionally, SynthMorph is robust to different modalities.

5.5 HyperMorph: Amortized Hyperparameter Learning for Image Registration

Hoopes et al. 2021 proposed an end-to-end strategy to learn the effect of registration hyperparameters. This should replace the traditional hyperparamter tuning process that usually requires considerable computational and human effort, which may lead to suboptimal parameter choices. HyperMorph is a hypernetwork that approximates a landscape of registration networks for a range of hyperparameter values, by learning a continuous function of hyperparameters (Hoopes et al. 2021). Therefore only a single HyperMorph

model needs to be trained. This allows to estimate optimal deformation fields for arbitrary image pairs and any hyperparameter value from a continuous interval (Hoopes et al. 2021). Additionally it enables for fast hyperparameter tuning at test-time.
The hypernetwork should learn the effect of the loss hyperparameters Λ on a given registration network (Hoopes et al. 2021). It is shown in Figure 17.

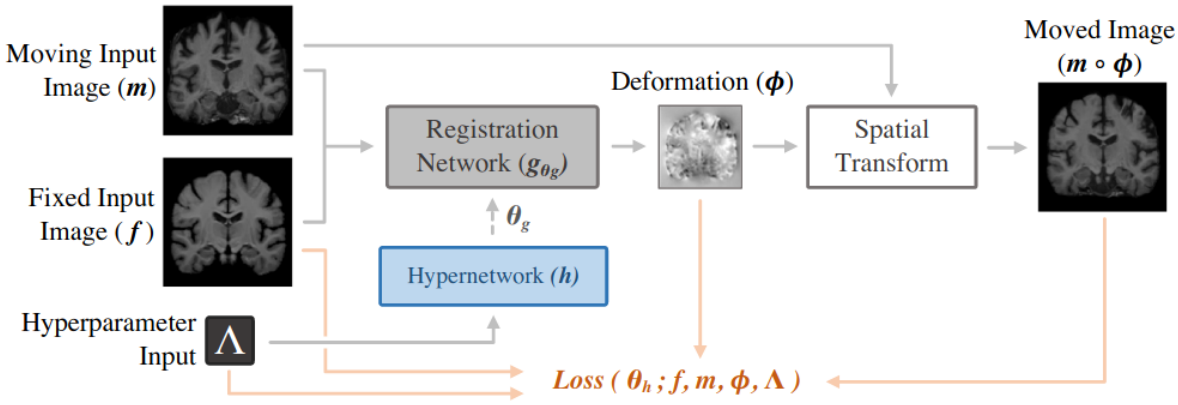

Figure 17: HyperMorph model (Copied from Hoopes et al. 2021) A hypernetwork (blue) learns to output the parameters of a registration network given registration hyperparameters Λ. HyperMorph is trained end-to-end, exploiting implicit weight-sharing among the full landscape of registration networks within a continuous interval of hyperparameter values.

The registration network takes as input a target image T and a moving image M and outputs a mapping $g_{\theta_g}(M, T) = \varphi$ that aligns the moving image to the target image. θ_g are the parameters of the registration network. The hypernetwork takes as input the loss hyperparameters Λ and outputs the registration network parameters $h_{\theta_h}(\Lambda) = \theta_g$. Hoopes et al. 2021 proposed to learn optimal hypernetwork parameters θ_h using stochastic gradient methods, optimizing the loss

$$L_h(\theta_h; D) = \mathbb{E}_{\Lambda \sim p(Lambda)} \left[L(\theta_h; D, \Lambda) \right]$$

where D is a dataset of images, $p(\Lambda)$ is a prior probability over the hyperparameters and $L(\cdot)$ is a registration loss involving hyperparameters Λ. They used the following loss function

$$L_h(\theta_h; D) = \mathbb{E}_{\Lambda} \left[\sum_{M,T \in D} \left((1 - \lambda)\, L_{sim}(T, M \circ \varphi; \lambda_{sim}) + \lambda L_{reg}(\varphi; \lambda_{reg}) \right) \right]$$

The loss term L_{sim} measures image similarity and might involve hyperparameters λ_{sim} and L_{reg} enforces spatial regularity of the deformation field and might also involve hyperparameters λ_{reg} (Hoopes et al. 2021). λ balances the relative importance of the two terms (Hoopes et al. 2021). Hence the set of hyperparameters is given by $\Lambda = \{\lambda, \lambda_{sim}, \lambda_{reg}\}$.

They used standard similarity metrics like MSE, local NCC and normalized MI. The later two have each one hyperparameter. As regularization they choose L2 regularization. Additionally, the deformation field φ was parametrized with a stationary velocity field v (Hoopes et al. 2021). Thus diffeomorphic deformation can be obtained through integration. Therefore the voxelmorph architecture from subsection 5.1 can be used as registration network. The loss can be extended to support semi-supervised learning for training with segmented data. The total loss expands to

$$L_h(\theta_h; D) = \mathbb{E}_\Lambda \sum_{M,T \in D} [(1-\lambda)(1-\gamma) L_{sim}(T, M \circ \varphi; \lambda_{sim}) + \lambda L_{reg}(\varphi; \lambda_{reg}) + (1-\lambda)\gamma L_{seg}(s_T, s_M \circ \varphi)]$$

where L_{seg} is a segmentation similarity metric (e.g. Dice score) and s_T and s_M are the segmentation maps of the target and moving image (Hoopes et al. 2021).

5.5.1 Results and Conclusion

Hoopes et al. 2021 have shown in their experiments that HyperMorph is much faster than training individually models with different hyperparametrs. Additionally, HyperMorph is able to yield similar optimal hyperparameters than traditional methods. Further, it is more robust to initialization. Therefore tedious hyperparameter search can be automated and performed much faster.

6 Applications

This section shows the results from the usage of the voxelmorph framework for three different datasets. First intra-modal volumetric T1-weighted MR brain images were used to train and evaluate, afterwards intra-modal volumetric T2-weighted MR brain images and lastly inter-modal volumetric T1-weighted - T2-weighted MR brain images. Unfortunately the learning-based mermaid framework easyreg could not be made to work. Therefore only voxelmorph was tested. First preprocessing of the data was needed. There were two other frameworks used for this task. First skull stripping was done with HD-BET (Isensee et al. 2019). An automated brain extraction tool for multi-sequence MRI using neural networks. It outperformed five other publicly available brain extraction algorithms and it can perform independent brain extraction on various different MRI sequences and is not restricted to precontrast T1-weighted sequences. Additionally it is straightforward to apply. Furthermore, affine alignment and normalization was done with intensity-normalization (Reinhold et al. 2019). It contains various methods to normalize the intensity of different modalities of magnetic resonance images. Additionally it provides the ability to co-register

a set of MR images to the mean image via an affine transformation. The network architecture was the same for the following case studies. We used the default voxelmorph model which is shown in Figure 11. The optimizer was Adam with a learning rate of 1e-4. Each model was trained for 6 days, resulting in about 120 epochs with 60 pairs per epoch. This corresponds to 7200 iterations. In every epoch a random sample of pairs were generated and given to the network. In comparison, 150'000 iterations were used for the experiments of Balakrishnan et al. 2019. Thus, our model was trained only for 5% of the iterations. In Figure 18 you can see how the training and testing loss decreases with the number of epochs for the intra-modal T2-weighted brain scan network. This behaviour was similar for the other two cases.

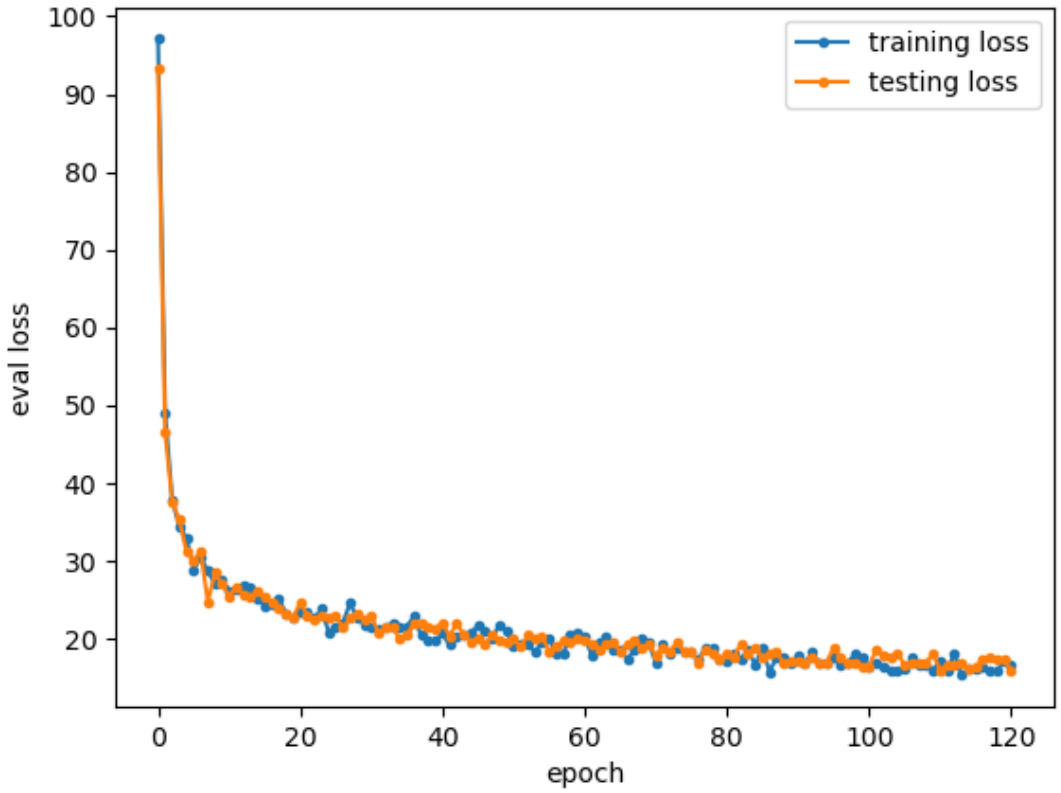

Figure 18: Loss for intra-modal T2-weighted brain scans network

The corresponding losses were approximated by evaluating 8 pairs each. The plot shows that the training loss and testing loss is more or less the same. This indicates that the network did not over fit. Nevertheless, a further decrease could be expected for training with more epochs.

6.1 intra-modal T1-weighted MR brain images

The intra-modal brain scans were volumetric T1-weighted MR images. There were 897 samples. First skull-stripping was applied with HD-BET. Then co-registering to the mean image and nyul normalization was performed with intensity-normalization. Afterwards the voxelmorph network could be trained and evaluated. The loss function consisted of a mean squared error with a weight of 1 and L2 regularization of the displacement field with a weight of 0.01. After six days of training 122 epochs could be reached. Therefore the network was trained with 7320 iterations.
A prediction of the trained network can be seen in Figure 19. It shows an example of T1-weighted MR coronal middle slices. In the third column the predicted warped moving image is shown. It illustrates that the model is capable of registering the global structures. Examples for this are indicated by the green ellipses. However, the network has difficulties with finer structures shown by the red ellipses. Further training could improve this issue and Balakrishnan et al. 2019 demonstrated the capabilities of the model.

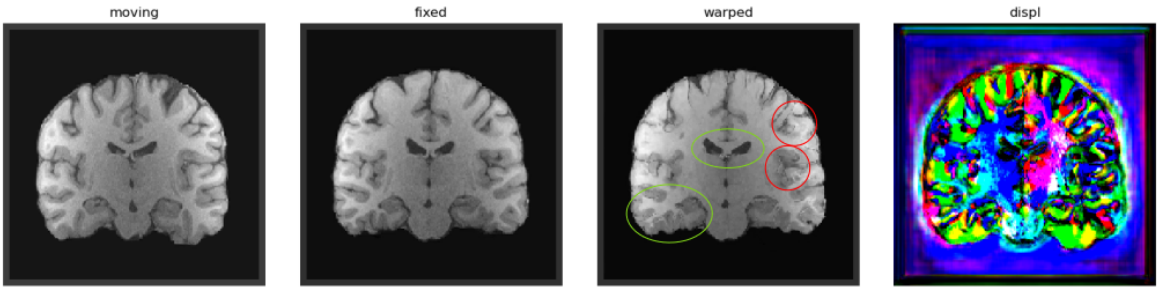

Figure 19: Example intra-modal T1-weighted MR coronal slices extracted from input pairs (columns 1-2), resulting $M \circ \varphi$ (column 3) and the displacement field (column 4). Displacement in each spatial direction is mapped to each of the RGB color channels.

In Figure 20 the progress of the network is shown. The top row shows the sagital, coronal and traverse middle slices of the target image. The middle row shows the warped moving image after 39 epochs and the bottom row after 122 epochs. The ellipses illustrate that the model gets better with more training. For example the ellipse for the sagital view shows that the boundaries get sharper. In addition, the circle on the bottom in the coronal view indicates that finer structures have been learned. This is further confirmed by the other ellipses

6.2 intra-modal T2-weighted MR brain images

This dataset contained volumetric T2-weighted MR images. There were 897 samples and the preprocessing steps and the loss function were the same as for the intra-modal T1-weighted MR brain images. The network was trained for 120 epochs which corresponds to 7200 iterations.

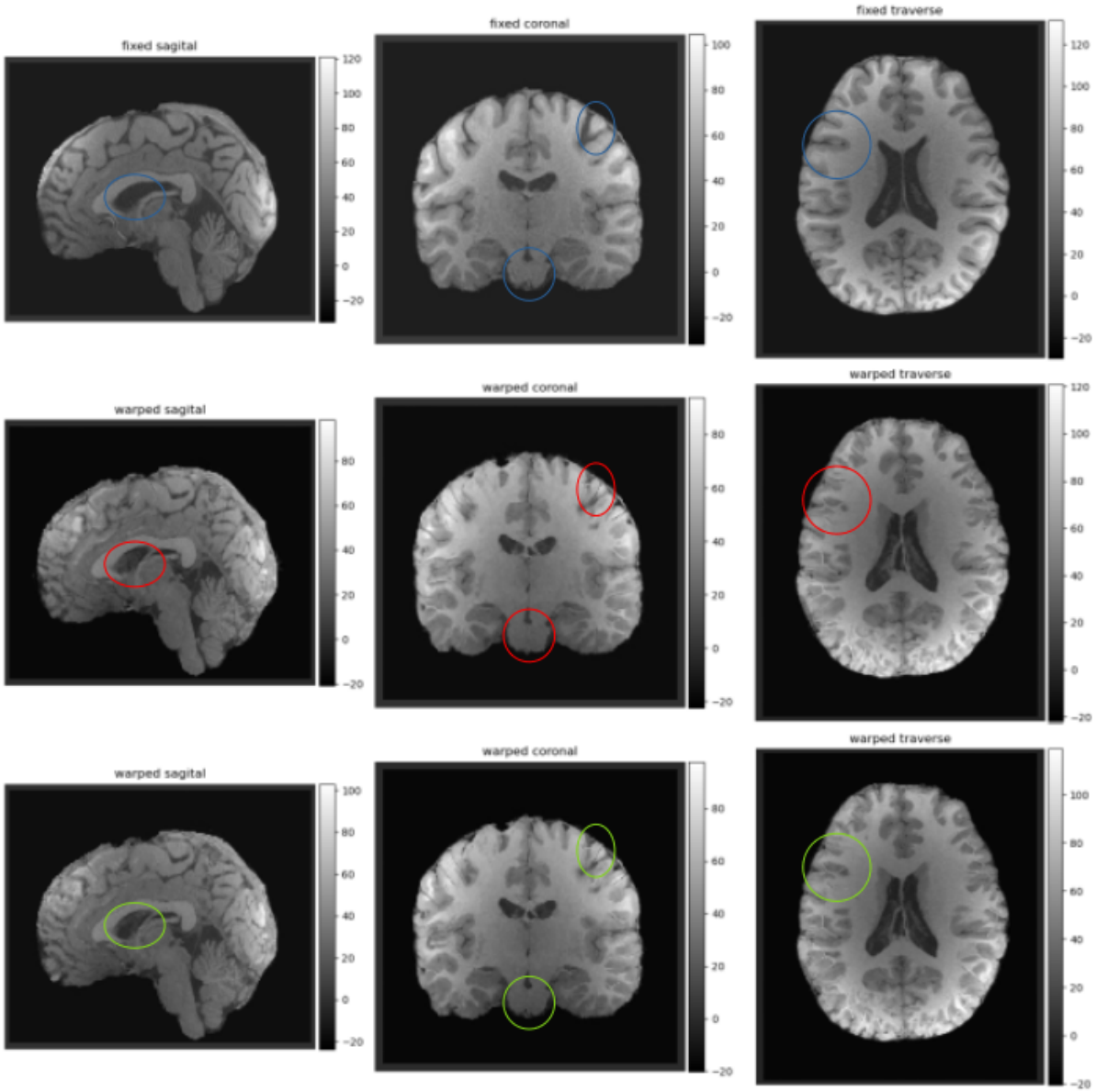

Figure 20: Example intra-modal T1-weighted MR slices extracted from the target image (top), the warped moving image (middle) after 39 epochs and the warped moving image after 122 epochs (bottom).

In Figure 21 an example prediction of the final network is given. The behaviour is comparable to the intra-modal T1-weighted model. The network has learned global structures, but has difficulties with finer details. This is again pointed out by the green and red ellipses. The green ellipse in the middle shows that this region is already pretty good. The green ellipse to the right demonstrates that the model has started to learn the desired anatomy. However, examples for issues are demonstrated by the red ellipses. The lower red ellipse illustrates that the network still needs to learn global trends and not just finer structures.

The improvement of the network is demonstrated in Figure 22. The sagital view is really noisy but the ellipses show that distinctive structures have improved with more training.

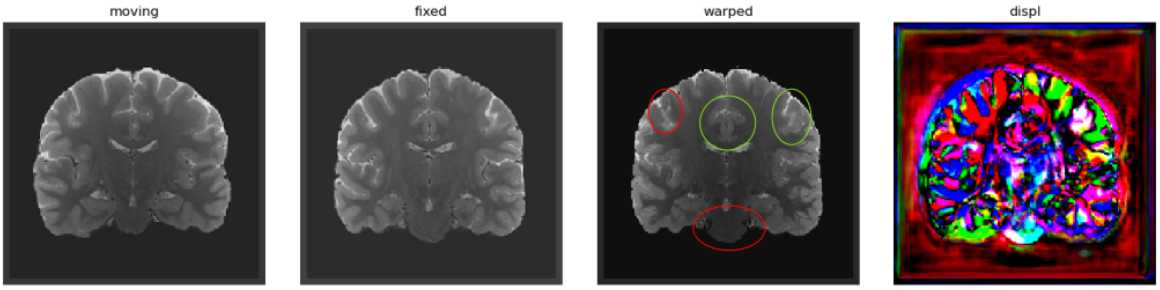

Figure 21: Example intra-modal T2-weighted MR coronal slices extracted from input pairs (columns 1-2), resulting $M \circ \varphi$ (column 3) and the displacement field (column 4). Displacement in each spatial direction is mapped to each of the RGB color channels.

The coronal view is much less noisy and the same behavior can be observed. The ellipse in the traverse view illustrates how the displacement of the anatomy is learned better.

6.3 inter-modal T1-weighted - T2-weighted MR brain images

The inter-modal brain scans were from a volumetric T1-weighted MRI dataset and a volumetric T2-weighted MRI dataset. There were 897 samples each. First skull-stripping was applied with HD-BET. This was done for each modality independently because HD-BET can handle skull-stripping T2-weighted MR images. Next each modality was co-registered to the corresponding mean image and nyul normalized with intensity-normalization. Normalization is not necessary but it allowed to reuse the preprocessed datasets from the two intra-modal cases. Local cross correlation with a weight of 1 was used as similarity measure and L2 regularization of the displacement field with a weight of 0.01 was added. Next the voxelmorph network could be trained and evaluated. The training process has reached 127 epochs, which corresponds to 7620 iterations.
Figure 23 presents the performance for T1-weighted - T2-weighted registration (top), T1-weighted - T1-weighted registration (middle) and T2-weighted - T2-weighted registration (bottom). The results for the latter two are slightly worse than the corresponding intra-modal models. This can be expected because learning inter-modal registration is harder. The performance for the T1-weighted - T2-weighted registration is much worse. So far, not much has been learned apart from the contour line. Nevertheless, some structures can already be recognised. It is remarkable that there is a lot of movement in the x-direction for the left side of the brain and no movement on the right side for the inter-modal case. The improvement of a T2-weighted - T1-weighted registration task is presented in Figure 24. As already shown in the previous Figure, the inter-modal performance is not so good. However, improvements can be observed. The middle row shows a prediction after 42 epochs and the bottom row after 127 epochs. The sagital view demonstrates that distinctive structures have been registered better with more training. Furthermore, the

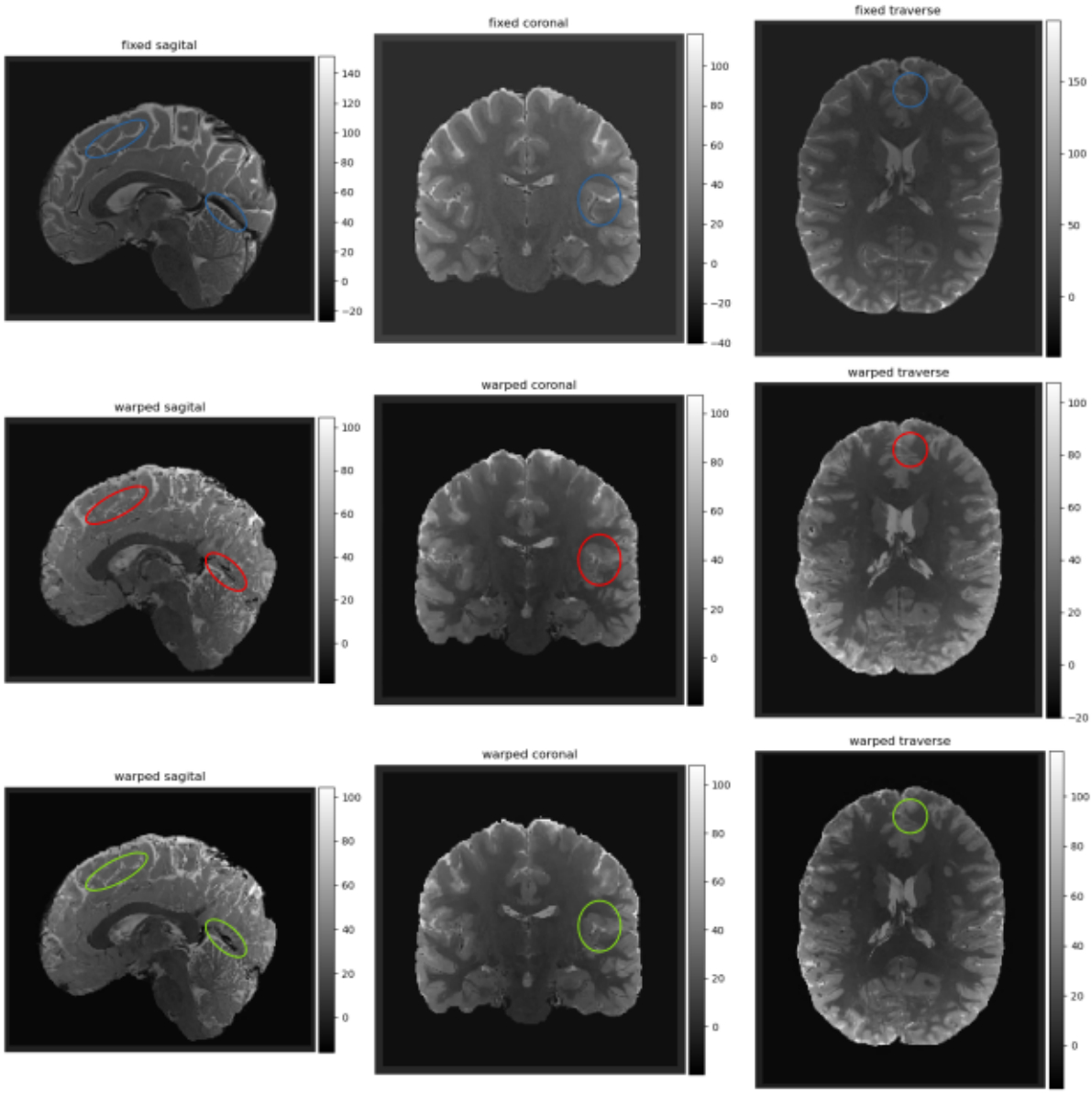

Figure 22: Example intra-modal T2-weighted MR slices extracted from the target image (top), the warped moving image (middle) after 40 epochs and the warped moving image after 120 epochs (bottom).

coronal view reveals that major errors get corrected over time. This is further confirmed by the traverse view.

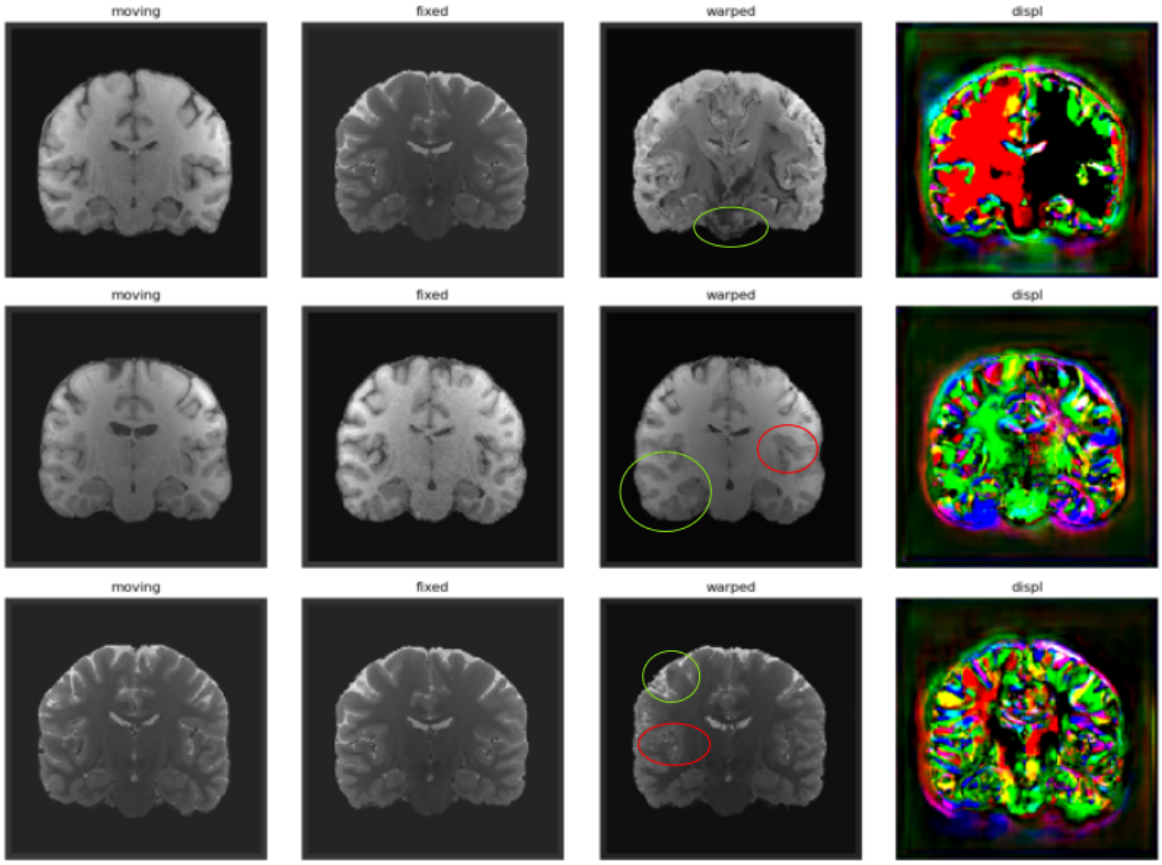

Figure 23: Example inter-modal MR coronal slices extracted from input pairs (columns 1-2), resulting $M \circ \varphi$ (column 3) and the displacement field (column 4). Displacement in each spatial direction is mapped to each of the RGB color channels. The top row shows the registration result for T1-weighted - T2-weighted registration, the middle row T1-weighted - T1-weighted registration and the bottom row T2-weighted - T2-weighted registration

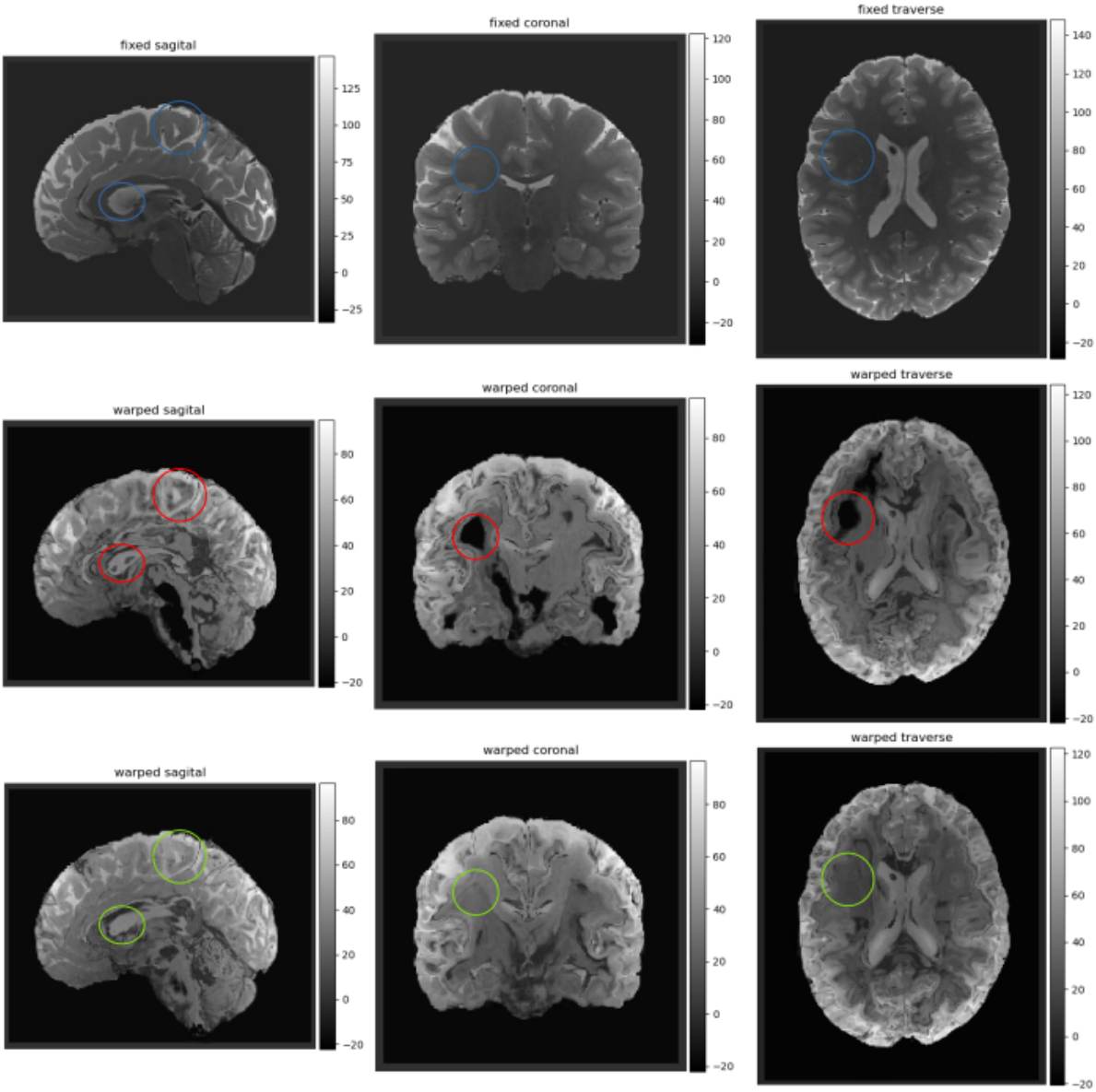

Figure 24: Example inter-modal T2-weighted - T1-weighted MR slices extracted from the target image (top), the warped moving image (middle) after 42 epochs and the warped moving image after 127 epochs (bottom).

7 Conclusion and Outlook

Two learning based frameworks for medical image registration were studied. Mermaid introduced spatio-temporal regularizer which is desirable because there are different scales of motion in different regions and at different times. This extends the traditional LDDMM theory and can still guarantee diffeomorphic transformations. Furthermore, the ASVM architecture removes the need of affine alignment of the data before training a network. This is more convenient and reduces the time requirement for pre-processing. Voxelmorph showed in various cases how helpful segmentation data during training can be. This was especially demonstrated remarkably with the SynthMorph architecture. There they could achieve a high performance for intra and inter-modal settings. Especially the later one is hard to achieve.

Unfortunately mermaid could not be tested. Although several hours were spent trying to get it to work, further efforts could potentially solve the problem. This would allow to directly compare both frameworks. A future task could be to create a detailed documentation of the mermaid framework and to provide an interface. This would help to use the framework for people who are not directly related to the project.

Voxelmorph has a python interface and gives three examples to show the capabilities. A documentation could help to adapt and use the model more dynamically. Nevertheless, the examples are really helpful.

Applying voxelmorph to the different datasets showed that the models could capture global trends but struggles with finer structures. Nevertheless, the achieved performance is not so good. The reason is that the networks have only been trained for around 7200 iterations. In comparison, Balakrishnan et al. 2019 used 150'000 iterations. Therefore our model was trained only for 5% of the iterations. More training was not used because it would be out of scope of this seminar work. Training for so many iterations with the current implementation and hardware setup would take roughly 120 days. Nonetheless, Balakrishnan et al. 2019 have already shown that the model is capable of achieving state-of-the-art performance for similar experimental setups.

The training of the networks was done with the default parameters. This is almost surely not the optimal choice. A large hyperparameter tuning study could be done to achieve better performance. Probably HyperMorph could be used for this task. Furthermore, the networks could be trained with more data or for a longer time. Additionally, the networks could be evaluated for anatomies other than the brain.

Last updated: 29.08.2021

References

Arsigny, Vincent, Olivier Commowick, Xavier Pennec, and Nicholas Ayache. 2016. "A Log-Euclidean Framework for Statistics on Diffeomorphisms." *Larsen R., Nielsen M., Sporring J. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006* 4190:924–931.

Balakrishnan, Guha, Amy Zhao, Mert Sabuncu, John Guttag, and Adrian V. Dalca. 2018. "An Unsupervised Learning Model for Deformable Medical Image Registration." *CVPR: Computer Vision and Pattern Recognition,* 9252–9260.

Balakrishnan, Guha, Amy Zhao, Mert Sabuncu, John Guttag, and Adrian V. Dalca. 2019. "VoxelMorph: A Learning Framework for Deformable Medical Image Registration." *IEEE TMI: Transactions on Medical Imaging* 38 (8): 1788–1800.

Beg, Mirza Faisal, Michael I. Miller, Alain Trouvé, and Laurent Younes. 2005. "Computing Large Deformation Metric Mappings via Geodesic Flows of Diffeomorphisms." *International Journal of Computer Vision* 61:139–157.

Bookstein, F.L. 1989. "Principal warps: thin-plate splines and the decomposition of deformations." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 (6): 567–585.

Christensen, G.E., R.D. Rabbitt, and M.I. Miller. 1996. "Deformable templates using large deformation kinematics." *IEEE Transactions on Image Processing* 5 (10): 1435–1447.

Christensen, Gary Edward. 1994. *Deformable Shape Models For Anatomy.*

Dalca, Adrian V, Marianne Rakic, John Guttag, and Mert R Sabuncu. 2019a. "Learning Conditional Deformable Templates with Convolutional Networks." *NeurIPS: Neural Information Processing Systems.*

Dalca, Adrian V., Guha Balakrishnan, John Guttag, and Mert Sabuncu. 2019b. "Unsupervised Learning of Probabilistic Diffeomorphic Registration for Images and Surfaces." *Medical Image Analysis* 57:226–236.

Dalca, Adrian V., Guha Balakrishnan, John Guttag, and Mert R Sabuncu. 2018. "Unsupervised Learning for Fast Probabilistic Diffeomorphic Registration." *MICCAI: Medical Image Computing and Computer Assisted Intervention, LNCS* 11070:729–738.

Gal, Yarin, and Zoubin Ghahramani. 2016. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.* arXiv: 1506.02142 [stat.ML].

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

Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. *Improving neural networks by preventing co-adaptation of feature detectors.* arXiv: 1207.0580 [cs.NE].

Hoffmann, Malte, Benjamin Billot, Juan Eugenio Iglesias, Bruce Fischl, and Adrian V. Dalca. 2020. *Learning image registration without images.* arXiv: 2004.10282 [cs.CV].

Hoopes, Andrew, Malte Hoffmann, Bruce Fischl, John Guttag, and Adrian V Dalca. 2021. "HyperMorph: Amortized Hyperparameter Learning for Image Registration." *arXiv preprint arXiv:2101.01035.*

Horn, Berthold, and Brian Schunck. 1981. "Determining Optical Flow." *Artificial Intelligence* 17 (August): 185–203.

Isensee, Fabian, Marianne Schell, Irada Pflueger, Gianluca Brugnara, David Bonekamp, Ulf Neuberger, Antje Wick, et al. 2019. *Automated brain extraction of multi-sequence MRI using artificial neural networks,* August.

Konukoglu, Ender. 2021. *Medical Image Analysis, Lecture 5: Image Registration.* ETH Zürich.

Labach, Alex, Hojjat Salehinejad, and Shahrokh Valaee. 2019. *Survey of Dropout Methods for Deep Neural Networks.* arXiv: 1904.13310 [cs.NE].

Niethammer, Marc, Roland Kwitt, and François-Xavier Vialard. 2019. "Metric Learning for Image Registration." *CoRR* abs/1904.09524. arXiv: 1904.09524. http://arxiv.org/abs/1904.09524.

Polzin, Thomas. 2018. "Large Deformation Diffeomorphic Metric Mappings– Theory, Numerics, and Applications." PhD diss., University of Lübeck.

Reinhold, Jacob C, Blake E Dewey, Aaron Carass, and Jerry L Prince. 2019. "Evaluating the impact of intensity normalization on MR image synthesis." In *Medical Imaging 2019: Image Processing,* vol. 10949, 109493H. International Society for Optics and Photonics.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. 2015. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* arXiv: 1505.04597 [cs.CV].

Rueckert, D., L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes. 1999. "Nonrigid registration using free-form deformations: Application to breast MR images." *IEEE Transactions on Medical Imaging* 18:712–721.

Shen, Zhengyang, Xu Han, Zhenlin Xu, and Marc Niethammer. 2019. "Networks for Joint Affine and Non-parametric Image Registration." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 4224–4233.

Shen, Zhengyang, François-Xavier Vialard, and Marc Niethammer. 2019. "Region-specific Diffeomorphic Metric Mapping." *NeurIPS: Neural Information Processing Systems.*

Springenberg, Jost Tobias, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2015. *Striving for Simplicity: The All Convolutional Net.* arXiv: 1412.6806 [cs.LG].

Yang, Xiao, Roland Kwitt, and Marc Niethammer. 2016. "Fast Predictive Image Registration." In *DLMIA: International Workshop on Deep Learning in Medical Image Analysis,* 48–57.

Yang, Xiao, Roland Kwitt, Martin Styner, and Marc Niethammer. 2017a. *Fast Predictive Multimodal Image Registration.* arXiv: 1703.10902 [cs.CV].

Yang, Xiao, Roland Kwitt, Martin Styner, and Marc Niethammer. 2017b. "Quicksilver: Fast predictive image registration–a deep learning approach." *NeuroImage* 158:378–396.

Younes, Laurent. 2010. *Shapes and Diffeomorphisms.* Vol. 2. Springer, Berlin, Heidelberg.