SrvfNet: A Generative Network for Unsupervised Multiple Diffeomorphic Functional Alignment

Elvis Nunez[1,3], Andrew Lizarraga[3], and Shantanu H. Joshi[2,3]
[1]Department of Electrical and Computer Engineering, [2]Department of Bioengineering, UCLA
[3]Ahmanson Lovelace Brain Mapping Center, Department of Neurology, UCLA
elvis.nunez@ucla.edu andrewlizarraga@mednet.ucla.edu s.joshi@g.ucla.edu

Abstract

We present SrvfNet, a generative deep learning framework for the joint multiple alignment of large collections of functional data comprising square-root velocity functions (SRVF) to their templates. Our proposed framework is fully unsupervised and is capable of aligning to a predefined template as well as jointly predicting an optimal template from data while simultaneously achieving alignment. Our network is constructed as a generative encoder-decoder architecture comprising fully-connected layers capable of producing a distribution space of the warping functions. We demonstrate the strength of our framework by validating it on synthetic data as well as diffusion profiles from magnetic resonance imaging (MRI) data.

1. Introduction

Owing to technological advances in imaging and sensing, the availability of commercial wearable devices, biosensors, and continuous recording instruments at reduced cost, there is an abundance of feature-rich scalar data. Such data arise from diverse applications and processes that produce continuous functions such as voltage, current from electronic meters, time-series data related to electroencephalography or electrocardiography from medical applications, scalar measures observed over spatial data from geological applications, or intensity-based features from spatial biological data. Recent advances in statistical shape and functional data analysis allow researchers to collectively study such data using geometric representations and structures [18].

An important step in the statistical analysis of multiple signals or recordings is matching or aligning them across the population. Pairwise matching or alignment is convenient for discriminative analysis such as classification and clustering, while the computation of the population mean is useful for inference. A typical approach for aligning multiple functional data involves a two step process: i) compute a mean in the appropriate space equipped with a choice of a metric by minimizing the variance of the set of functions, often iteratively in case a closed form solution is not available, and ii) align or register all the elements in the dataset to this mean, thereby establishing a correspondence across the entire population. The procedure for the computation of the mean is slow and typically yields a local solution if gradient descent is used. Further, in the case of signals and time-series, if one seeks invariance to reparameterization (or arbitrary nonlinear order-preserving warpings of the domain of the function), then one needs to employ dynamic time warping (DTW) procedures [14]. Although efficient algorithms exist, this process can quickly become computationally intensive for T-length data $O(T^N)$ as the size of the data (N) increases. If DTW is applied at every iteration until the convergence of the mean, the computational cost can become prohibitively expensive for a large population ($N > 500K$). A slightly faster yet sub-optimal solution is to compute a Euclidean (extrinsic) average, project it back onto the space of functions and, using it as a template, compute dynamic time warps with respect to this template. However, such an average template may not be optimal with respect to the underlying metric, and because it was computed by ignoring the nonlinear domain warps, may not preserve important geometric features present in the data.

In this paper, we propose a novel neural network architecture *SrvfNet* inspired from variational auto-encoders to simultaneously predict multiple functional data alignments in addition to the population templates.

1.1. Background and Related Work

Several researchers have proposed ideas for machine-learning based estimation of nonlinear alignment. Here, we discuss the approaches related to our work. A framework based on Gaussian process-based non-parametric priors was proposed by Kazlauskaite et al. [7] to learn a latent model for functional alignments. A deep-learning based approach, the deep canonical time warping (DCTW) was used to perform joint temporal registration, while at the same time maximizing the joint correlation of multiple time-series [19]. A sequence transformer network was proposed by Oh et al. [13] to perform stretching, compression, shifting, and flipping of the time-series signals to incorporate invariant matching of signals from clinical data. The approach by Lohit et al. [11] uses a temporal transformer network (TTN) to learn the inter-signal warping functions while performing classification simultaneously. The TTN performs a joint discriminative alignment of time-series by decreasing the intra-class variability and increasing the inter-class separation without using a fixed template for each class. Abid et al. [1] proposed Auto-warp, which learns and optimizes a metric distance on the set of unlabeled time-series. This is achieved by pairwise alignment of signals without diffeomorphic constraints on the warping functions. Recently, Weber et al. [15] have proposed the learning of diffeomorphisms in an unsupervised manner. Their approach uses temporal transformer layers to perform joint alignment of time-series and includes a loss function that aims to minimize the empirical variance of warped signals while regularizing the network using continuous piecewise-affine (CPA) covariance matrices [2]. Koneripalli et al. [10] also use autoencoders and temporal warping layers for unsupervised learning of diffeomorphisms. The reader is also referred to a previous approach by Nunez and Joshi [12] where a convolutional network is used for predicting warping functions when matching a pair of shapes.

1.2. Contributions

In this paper we develop a deep neural network architecture, SrvfNet, that allows for unsupervised multiple diffeomorphic alignment of functional data. We show that this generative-based architecture is capable of generating warping functions to warp data to a given template, and can simultaneously learn a suitable template when one is not provided. Our method is unsupervised in the sense that only the population of functions we wish to align are input–that is, we do not require pre-computed warping functions nor a priori labels/targets. Instead, our loss function aggregates an intrinsic geometric distance of the predicted alignments which is then iteratively minimized to improve the predicted alignments. Further, a key distinction between our work and others is in our shape representation of functional data through the use of the Square-Root Velocity Function (SRVF) [4]. Our network architecture is simple and consists only of fully-connected and regularization layers. The design of our network is inspired by the variational autoencoder (VAE) [8] in the sense that inputs are encoded onto a low-dimensional latent space which then serves to model the conditional posterior distribution of the warping functions. Contrary to the autoencoder design of VAEs, we do not measure a reconstruction error, nor do we impose a distribution assumption on the posterior distribution of the inputs given the latent variables. VAEs are trained to maximize the evidence lower bound (ELBO) function. While our loss functions partially build off of the ELBO function, we do not include the term that arises from the posterior distribution assumption made by VAEs. Instead, one novelty of our approach is in the replacement of this term with a geometric measure.

This generative approach distinguishes our work from recent developments such as that of [15, 10, 12] where the produced outputs are deterministic. Our approach has the added benefit of providing a statistical summary of the intrinsic relationship between the class of training data and the provided or predicted template. More specifically, our framework differs from [15] in the following ways: i) our architecture does not use any temporal transformer layers nor any CPA-based transformations [2], instead our network consists of simple feed-forward fully-connected layers with basic batch normalization and dropout regularizations; ii) our loss functions do not promote smoothness in the diffeomorphisms through a CPA-based regularization penalty and instead do so by penalizing the gradients of the predicted warps; iii) we approach this problem through a geometric lens and consider a representation of the data that encodes geometric variability rather than working with the raw data–consequently, our loss functions aim to minimize an intrinsic distance over a Hilbert sphere; iv) our framework is generative and provides a distribution over possible diffeomorphisms. Different from [10], we use a variational-based autoencoder rather than a deterministic one. Moreover, we do not aim to learn network weights by minimizing a mean squared reconstruction error, but rather minimize a geometric measure on the predicted alignments. To this end, our framework allows for direct penalties over learned diffeomorphisms to control for smoothing that in turn lead to more robust alignments. Our paper also differs from [12] where they used a supervised deep learning framework to predict warps between pairs of shapes. Unlike their approach, our network can not only predict pairwise warping between functions, but can also perform unsupervised template estimation of multiple functions taken together.

2. Methods

2.1. Shape Representation and Preliminaries

We first provide the reader a brief summary of the shape representation of curves that will be used through out this paper. We represent curves as one-dimensional parameterized functions. These functions are assumed to be first differentiable, and belonging to the class of $\mathbb{L}^2$ functions, denoted as $f \in \mathbb{L}^2([0,1],\mathbb{R})$, where an entire of collection of curves is denoted $\mathcal{F} \equiv \left\{ f_i \mid f_i \in \mathbb{L}^2([0,1],\mathbb{R}) \right\}$. We define a template as a designated fixed function $g \in \mathbb{L}^2([0,1],\mathbb{R})$, and we seek to align an entire collection of curves, $\mathcal{F}$, to the template curve.

When performing computations and shape comparisons of curves, we represent them as SRVFs [4, 5, 17]. Each f_i is represented by the SRVF map $f_i \mapsto q_i = \frac{\dot{f}_i}{\sqrt{\|\dot{f}_i\|}}$, and the space of such functions is denoted by $\mathcal{S}$. Adding the constraint, $\int_{[0,1]} \langle q(t), q(t)\rangle \, dt = 1$ under the standard $\mathbb{L}^2$ inner product $(\langle\cdot,\cdot\rangle)$ ensures that our shape representations are invariant to translation and scaling. As defined, the SRVFs are unit normalized, thus shape comparisons are equivalent to computing geodesic distances between SRVF points on the unit Hilbert Sphere $\mathcal{S}$ in $\mathbb{L}^2([0,1],\mathbb{R})$. We introduce an analogous notation for $\mathcal{S}$ where we designate a class of functions as $\mathcal{Q} \equiv \{q_i \mid q_i \in \mathcal{S}\}$.

When warping a function f to a specified template g, we solve the following optimization problem in terms of the Fisher-Rao metric via dynamic programming [16]

$$\underset{\gamma}{\operatorname{argmin}} \left\| q_f - \sqrt{\dot{\gamma}}(q_g \circ \gamma) \right\|^2, \tag{1}$$

where q_f, q_g are the respective SRVF representations of f, g, and $\gamma : [0,1] \to [0,1]$ is a reparameterization function. The γ obtained in this fashion acts as a suitable diffeomorphism between q_f and q_g, which in turn produces a warp from f to g in the original space. In fact, we need only recover $\dot{\gamma}$, since γ can be reconstructed via

$$\gamma(s) = \int_{[0,s]} \dot{\gamma}(t) dt. \tag{2}$$

In the general setting, we have a collection of curves $\mathcal{F}$ where we must make a justified choice of template. In this setting, the Kärcher mean [6] is a well-known average representation of the overall shape variability. Hence, the Kärcher mean, denoted q_μ, serves as a suitable template candidate and is given by

$$q_\mu = \underset{q}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \underset{\gamma}{\operatorname{argmin}} \, d(q, \sqrt{\dot{\gamma}}(q_i \circ \gamma))^2, \tag{3}$$

where $d(,)$ is the geodesic distance given by, $d(\psi_i, \psi_j) = \cos^{-1} \langle \psi_i, \psi_j \rangle$, where $\langle\cdot,\cdot\rangle$ is the standard $\mathbb{L}^2$ inner product.

2.2. Unsupervised Prediction of Warping Functions Under a Fixed Template

In this paper, we minimize a loss function that uses the chord distance $\left\| q - \sqrt{\dot{\gamma}}(q_i \circ \gamma) \right\|$ instead of the geodesic distance and apply deep learning to learn the desired template and to learn the proper warping functions without prior knowledge of the warps nor the Kärcher mean, effectively making this approach unsupervised. Formally, since the geodesic paths under the SRVF framework converge to a locally unique solution, the optimization problem in (1) produces a unique diffeomorphism, γ. However, in the unsupervised approach, we don't have prior information of the uniqueness of geodesic paths, so we specify the constraints $\gamma(0) = 0$, $\gamma(1) = 1$, and a non-decreasing condition on γ. This ensures that γ produced by the network is unique.

We start by assuming we have a template $\bar{q}$ and a collection $\mathcal{Q} \equiv \{q_i\}_{i=1}^N$ which we would like to warp to $\bar{q}$. As such, we seek to obtain $\{\gamma_i\}_{i=1}^N$ that solve equation (1). We assume that all functions $\bar{q}$, $\{q_i\}_{i=1}^N$ have been discretized into T uniformly spaced points over the domain $[0,1]$. We construct an unsupervised deep learning framework based on the variational autoencoder [8] to subsequently not only obtain the γ_i diffeomorphisms, but also an estimate of the distribution of γ_i's, which we denote Γ, that warp functions from the class $\mathcal{Q}$ to $\bar{q}$.

Adopting the standard autoencoder terminology, our SrvfNet architecure consists of an *encoder* and a *decoder*. We let ϕ denote the trainable parameters of the encoder network, and θ the parameters of the decoder. The encoder takes a function q and maps it to an ℓ-dimensional space via the reparameterization trick. In particular, the encoder outputs a mean $\mu^\phi(q) \in \mathbb{R}^\ell$ and diagonal covariance matrix $\Sigma^\phi(q) \in \mathbb{R}^{\ell\times\ell}$ which are then used to construct the low-dimensional representation $z \in \mathbb{R}^\ell$ defined by

$$z = \left(\Sigma^\phi(q)\right)^{\frac{1}{2}} \tilde{z} + \mu^\phi(q) \tag{4}$$

where $\tilde{z} \sim \mathcal{N}(0, I_\ell)$. This has the effect of modeling the posterior distribution of the latent variable z given q as $p^\phi(z|q) = \mathcal{N}(\mu^\phi(q), \Sigma^\phi(q))$ which is then transformed into a distribution over the diffeomorphisms by the decoder. We also invoke a prior on the latent variables $z \sim \mathcal{N}(0, I_\ell)$ to allow for efficient sampling over Γ. More specifically, once our network is trained, we can generate a sample from Γ by sampling $z \sim \mathcal{N}(0, I_\ell)$ and then pass z through the decoder. This gives rise to the first term in our training loss function where we push the posterior $p^\phi(z|q)$ to resemble the prior $p(z)$ and measure this disparity through the Kullback-Leibler (KL) divergence. Because the posterior and prior distributions are assumed to be normal, the KL divergence exhibits a closed-form solution. Letting $\mathcal{L}_{KL}(q) = D_{KL}\left(p^\phi(z|q) \,||p(z)\right)$, we have

$$\mathcal{L}_{KL}(q) = \frac{1}{2}\Big[\text{trace}\left(\Sigma^\phi(q)\right) - l + \left(\mu^\phi(q)\right)^T \mu^\phi(q) \\ - \log\left|\Sigma^\phi(q)\right|\Big]. \tag{5}$$

Once z is obtained using equation (4), it is propagated through the decoder which outputs $v^\theta \in \mathbb{R}^T$ which then goes through a diffeomorphic constraint satisfaction layer.

We construct γ by first transforming v^θ into an estimate of $\dot{\gamma}$ which we denote $\dot{\gamma}^\theta$. Similar to Lohit et al. [11], we first map v^θ onto the probability simplex by normalizing and then take the Hadamard product of the resulting output as given by

$$\dot{\gamma}^\theta = \frac{v^\theta}{\|v^\theta\|} \odot \frac{v^\theta}{\|v^\theta\|}. \tag{6}$$

In conjunction with equation (2), the estimated γ, denoted γ^θ, is then constructed as

$$\gamma^\theta(s) = \sum_{t=1}^{s} \dot{\gamma}^\theta(t). \tag{7}$$

Normalizing v^θ by $\|v^\theta\|$ in equation (6) ensures that all elements in $\frac{v^\theta}{\|v^\theta\|}$ are in the interval $[-1, 1]$, and taking the Hadamard product ensures all elements of $\dot{\gamma}^\theta$ are nonnegative, which leads to a non-decreasing γ^θ.

We further impose the constraint that $\gamma^\theta(0) = 0$ and $\gamma^\theta(1) = 1$. To promote smooth diffeomorphisms, we use linear interpolation to downsample $\gamma^\theta \in \mathbb{R}^T$ to $\tilde{\gamma}^\theta \in \mathbb{R}^{\tilde{T}}$ uniformly on $[0, 1]$ where $\tilde{T} < T$. We then use linear interpolation again to upsample $\tilde{\gamma}^\theta$ back to $\mathbb{R}^T$. With a slight abuse of notation, we refer to the upsampled $\tilde{\gamma}^\theta$ as γ^θ.

The second term in our training loss function is inspired by the Fisher-Rao metric defined in equation (1) and is given by

$$\mathcal{L}_{FR}(q, \gamma^\theta; \bar{q}) = \left\| \bar{q} - \sqrt{\dot{\gamma}^\theta}(q \circ \gamma^\theta) \right\|_2^2. \tag{8}$$

To penalize diffeomorphisms with large slopes and to further promote smoothness, we consider penalties on the norms of both the first and second derivatives of γ^θ. For the first derivative, we define

$$\mathcal{L}_\nabla\left(\gamma^\theta\right) = \|\dot{\gamma}^\theta\|_2^2, \tag{9}$$

and for the second derivative,

$$\mathcal{L}_{\nabla^2}\left(\gamma^\theta\right) = \|\ddot{\gamma}^\theta\|_2^2. \tag{10}$$

The loss function we consider is a weighted sum of $\mathcal{L}_{KL}, \mathcal{L}_{FR}, \mathcal{L}_\nabla$, and $\mathcal{L}_{\nabla^2}$ given by equations (5), (8), (9), and (10) with respective weights $\lambda_{KL}, \lambda_{FR}, \lambda_\nabla, \lambda_{\nabla^2}$. For a fixed template $\bar{q}$ and function q with predicted warping function γ^θ, the loss is therefore given by

$$\begin{aligned}\mathcal{L}_F = \lambda_{FR}\mathcal{L}_{FR}\left(q, \gamma^\theta; \bar{q}\right) + \lambda_{KL}\mathcal{L}_{KL}(q) + \lambda_\nabla\mathcal{L}_\nabla\left(\gamma^\theta\right) \\ + \lambda_{\nabla^2}\mathcal{L}_{\nabla^2}\left(\gamma^\theta\right)\end{aligned} \tag{11}$$

For a training batch $\{q_i\}_{i=1}^B$, we define the loss as

$$\begin{aligned}\mathcal{L}_F = \frac{1}{B}\sum_{i=1}^{B}\big[\lambda_{FR}\mathcal{L}_{FR}\left(q_i, \gamma_i^\theta; \bar{q}\right) + \lambda_{KL}\mathcal{L}_{KL}(q_i) \\ + \lambda_\nabla\mathcal{L}_\nabla\left(\gamma_i^\theta\right) + \lambda_{\nabla^2}\mathcal{L}_{\nabla^2}\left(\gamma_i^\theta\right)\big].\end{aligned} \tag{12}$$

2.3. Unsupervised Joint Prediction of the Template and Warping Functions

In the previous section we considered the case where a template $\bar{q}$ was provided to us a priori. This could be, for example, the Kärcher mean of the collection $\mathcal{Q}$ as defined by equation (3), or it could be some $q_i \in \mathcal{Q}$, or any other function in $\mathcal{S}$. In this section we will consider the case where the template is unknown to us, and we wish to estimate the optimal template to warp the collection $\mathcal{Q}$ to. We would like this predicted template to capture the geometric variability of $\mathcal{Q}$, and as such aim to integrate the notion of the Kärcher mean into our previous Fisher-Rao loss defined in equation (8). In particular, for batch $\{q_i\}_{i=1}^B$, we estimate the template as

$$\hat{q} = \Omega\left(\frac{1}{B}\sum_{i=1}^{B}\sqrt{\dot{\gamma}_i^\theta}\left(q_i \circ \gamma_i^\theta\right)\right) \tag{13}$$

where $\Omega(u) = \frac{u}{\|u\|_2}$ normalizes its input. In other words, we estimate the template as the Euclidean mean of the warped batch and then normalize to ensure $\hat{q} \in \mathcal{S}$.

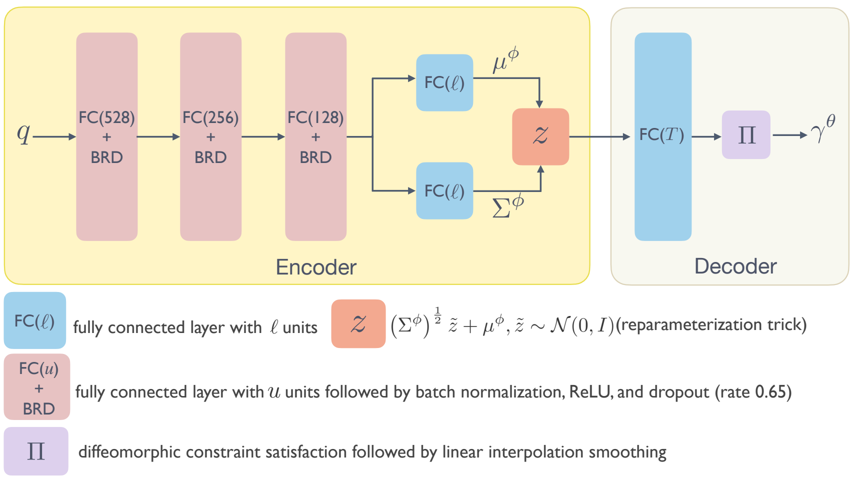
The batch template $\hat{q}$ can now be used as a surrogate for the template $\bar{q}$ as given in equation (8). For a given batch, our Fisher-Rao loss function takes the form

$$\hat{\mathcal{L}}_{FR}(q, \gamma^\theta; \hat{q}) = \left\| \hat{q} - \sqrt{\dot{\gamma}^\theta}(q \circ \gamma^\theta) \right\|_2^2 \tag{14}$$

where $\hat{q}$ is as defined in equation (13). The remaining terms in our loss function mirror those from the fixed template setting in equation (12). More precisely, we will again have weights $\hat{\lambda}_{FR}, \lambda_{KL}, \lambda_\nabla, \lambda_{\nabla^2}$ for the Fisher-Rao loss with template prediction, KL loss, and first and second derivative penalties as defined in equations (14), (5), (9), and (10), respectivly. In this setting, we aim to minimize the following loss defined for batch $\{q_i\}_{i=1}^B$

$$\begin{aligned}\mathcal{L}_E = \frac{1}{B}\sum_{i=1}^{B}\big[\hat{\lambda}_{FR}\hat{\mathcal{L}}_{FR}\left(q_i, \gamma_i^\theta; \hat{q}\right) + \lambda_{KL}\mathcal{L}_{KL}(q_i) \\ + \lambda_\nabla\mathcal{L}_\nabla\left(\gamma_i^\theta\right) + \lambda_{\nabla^2}\mathcal{L}_{\nabla^2}\left(\gamma_i^\theta\right)\big].\end{aligned} \tag{15}$$

While our setup does not explicitly output a predicted template, once our network has been trained we can obtain an estimate of the predicted template by computing equation (13) over the entire training data.



q
FC(528) + BRD
FC(256) + BRD
FC(128) + BRD
FC(ℓ)
μ^{ϕ}
FC(ℓ)
Σ^{ϕ}
z
Encoder
FC(T)
Π
γ^{θ}
Decoder
FC(ℓ) fully connected layer with ℓ units
z $(\Sigma^{\phi})^{\frac{1}{2}}\tilde{z}+\mu^{\phi}, \tilde{z}\sim\mathcal{N}(0,I)$(reparameterization trick)
FC(u) + BRD fully connected layer with u units followed by batch normalization, ReLU, and dropout (rate 0.65)
Π diffeomorphic constraint satisfaction followed by linear interpolation smoothing

Figure 1. Schematic of the SrvfNet architecture.

3. Network Design

The SrvfNet architecture is designed to operate on both the fixed template setting described in section 2.2 and the template prediction setting from section 2.3. The training regimes differ only in their respective loss functions given by equations (12) and (15).

3.1. SrvfNet Architecture

A schematic diagram of the SrvfNet architecture is provided in figure 1. Both our encoder and decoder networks consist only of fully-connected layers. We let BRD represent a batch normalization layer, followed by a ReLU activation, and finally a dropout layer with drop rate 0.65. Our encoder, then, consists of three fully-connected-BRD pairs with 528, 256, and 128 fully-connected units, respectively. The output of this last layer is then passed to two fully-connected layers, each with ℓ units. The first represents the mean $\mu^{\phi}(q)$, while the second represents the diagonal covariance matrix $\Sigma^{\phi}(q)$ (for numerical stability, this layer outputs $\log\left(\Sigma^{\phi}(q)\right)$ which is exponentiated to recover $\Sigma^{\phi}(q)$). $\mu^{\phi}(q)$ and $\Sigma^{\phi}(q)$ are then used to sample $z\sim\mathcal{N}\left(\mu^{\phi}(q),\Sigma^{\phi}(q)\right)$ as in equation (4).

Given z, we obtain γ^{θ} by passing z through the decoder network. Our decoder consists of a single fully-connected layer with T units which is then followed by a diffeomorphic constraint satisfaction and smoothing layer, which we denote Π, as discussed in section 2.2.

3.2. Training and Implementation Details

All network weights are initialized using a Glorot Uniform initialization [3]. We use a batch size of 512 and an Adam optimizer [9] with learning rate 10^{-3}. Loss function weights as defined in equations (12) and (15), as well as number of training epochs, are dataset-dependent and discussed further in subsequent sections. All models are implemented using TensorFlow on an Intel i7-7700K CPU @ 4.20GHz machine equipped with TITAN Xp GPUs.

4. Experiments

4.1. Datasets

Our data comprises two classes i) synthetic bump functions, and ii) diffusion profiles from MRI data. We first consider bump functions as shown in the first (original) column of figure 2. In this data we consider a collection of randomly generated sinusoidal functions characterized by their number of peaks, amplitudes, and phases. Each function is discretized to $T=300$ uniformly-spaced points on the interval $[0,1]$. We consider a two-bump dataset that consists of bump functions with two peaks and a three-bump dataset that consists of bump functions with three peaks.

We also consider a more realistic dataset consisting of fractional anisotropy (FA) values derived from white matter fiber bundles as depicted in the 'original' column of figure 6. In this dataset, each function is discretized to $T=100$ uniformly-spaced points on the interval $[0,1]$. We consider five bundle subclasses: i) the arcuate tract, ii) corpus callosum forceps minor (CCFmin), iii) cortico spinal tract (CST), iv) superior longitudinal fasciculus (SLF), and v) thalamic radiation tract (Th Rad).

4.2. Unsupervised Warping to a Fixed Template

In the fixed template setting, we perform two experiments on the bump datasets.

4.2.1 Two-to-Two Bumps Matching

We first generate a random bump function with two peaks and designate this as the template. We then randomly gener-

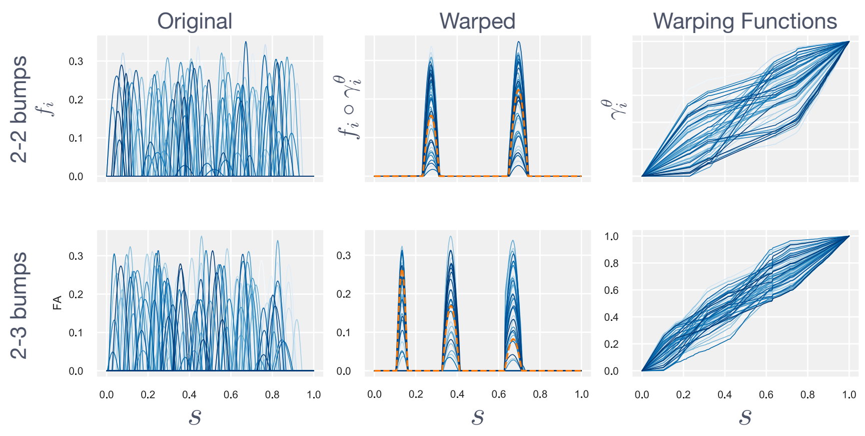


Figure 2. Two-to-Two and Two-to-Three bump matching with template (in orange).

ate $50,000$ bump functions to serve as our training dataset and $5,000$ more to serve as our test set. All bump functions are converted to their SRVF representations and are normalized to have unit length. We then train our network to minimize the fixed-template loss function given by equation (12). We train for 5000 epochs (approximately 30 minutes of training) and use a latent space dimension of $\ell = 150$. After training, we sample 70 bumps from our test set and obtain the predicted warping functions and apply them to the sampled functions. Figure 2 shows the test data, the warped data to a fixed template (orange) and the predicted warping functions γ_i^θ.

4.2.2 Two-to-Three Bumps Matching

In this experiment we reuse the two-bump training and test sets constructed in the previous experiment, but replace the template with a bump function that has three peaks. We train for 5000 epochs with a 150-dimensional latent space and visualize the performance on 70 randomly-sampled test functions as shown in the second row of figure 2. To clarify the nature of the warping functions learned by our network, we visualize the process on a single function in figure 3, where we see two peaks in the test function warp to two out of three peaks in the template (orange).

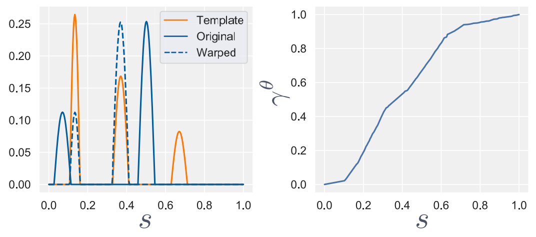


Figure 3. Two-to-Three bump matching with template (in orange).

4.2.3 Random Samples in the Encoding space of Warps

As discussed in section 2.2, our network allows sampling from the estimated space of diffeomorphisms that warp the training functions to the prescribed template by repeatedly sampling $z_i \sim \mathcal{N}(0, I_\ell)$ and passing this through the decoder. To qualitatively demonstrate that this procedure yields valid warping functions, we randomly sampled 200 z_i's and passed them through the decoder for the two-bump template model to obtain warping functions $\gamma^\theta(z_i)$. Figure 4 visualizes these results, which empirically shows the distribution over warps that encodes the intrinsic relationships between the class of two-bump functions and the template.

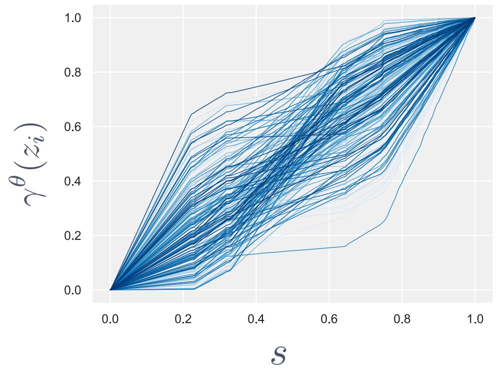


Figure 4. Distribution of γ for two-to-two bump matching with a fixed template.

4.3. Unsupervised Template Prediction

In the template prediction setting, we show results on synthetic data and diffusion profile data from fiber bundles.

4.3.1 Two-to-Two Bumps

Here we reuse the train and test datasets constructed for the two-bump dataset from section 4.2.1. However, instead of using a template $\bar{q}$, we minimize the loss function given in equation (15) and learn an appropriate template from the training data. We train for 1200 epochs with $\ell = 150$ and visualize the performance of our trained model on 70 randomly sampled test functions in figure 5.

4.3.2 Fractional Anisotropy (FA) Profiles

We train five different models on each of the bundles. Our training sets for Arcuate, CCFmin, CST, SLF, and Th Rad, have sizes 300K, 260K, 110K, 390K, and 160K, respectively. Test sets ranged in size from 20K to 90K. We again aim to minimize the loss function given in equation (15) and train for 1250 epochs (about $15 - 30$ minutes of training) with $\ell = 50$. We visualize the performance of our models on 70 randomly sampled functions from our test sets in figure 6. After the warping phase, one can see more pronounced patterns of shapes in the underlying FA values.

5. Discussion

5.1. Fixed Template

From figure 2 we observe that SrvfNet can successfully warp the functions in the test set to the template. Additionally, the produced warping functions appear as piecewise affine warps as expected when mapping a two-bump function to another bump function. We also replicated this experiment using the standard DTW method and found that the procedure took approximately 11 minutes to warp $50,000$ two-bump functions to a three-bump template on a 2.6GHz 6core Intel Core i7 processor. In contrast, the training phase of our network is slower and takes approximately 30 minutes to warp the training set to the prescribed template. While training time is slower, once the network is trained, the efficient implementation of the forward pass allows for extremely fast warpings of unseen functions belonging to the same class of training data.

5.2. Template Prediction

SrvfNet provides a convenient way to compute a population template by computing a mean of the set of warped inputs. Since all functions will be nonlinearly aligned to this underlying predicted template, their mean will resemble the predicted template. By simply visualizing the warped inputs as in figure 5 for the two-to-two bump experiment, we can also gain an insight into the types of functions that the model learns for the purpose of warping. For this two-to-two bump experiment, the network was able to learn a suitable two-bump template function, as evidenced by the two peaks in the warped outputs. Moreover, the predicted warping functions continue to exhibit the piecewise affine structure that we expect and observed in the fixed template case. We also note that while our network may have learned a template that minimized equation (15), this solution may not be unique as there may be several functions that yield the same objective. Indeed, in our experiments we found that different initializations lead to varying predicted templates with similar loss function values. To replicate this experiment using DTW we first need to construct a template as there is not one provided in this setting. Since our framework aims to construct a template that captures the geometric properties of the data, we would like our DTW template to also encode the shape variability of the training data. As discussed in 2.1, one such suitable template is the Kärcher mean in equation (3), from which our loss function (15) is based. However, solving equation (3) becomes prohibitively expensive for large datasets. As an example, solving (3) for $100K$ diffusion profiles would take roughly 25 hours on our platform. This is 150 times slower than our approach, which takes less than 30 minutes to train on $300K$ diffusion profiles, while also producing a template that encodes the shape variability of the profiles.

In general, we have no guarantees on the uniqueness of the predicted template. In all our experiments we assumed that the signal-to-noise ratio (SNR) is acceptable (> 15 dB after following standard processing techniques for MRI data). Further, in the case of functions and signals, where the noise levels are comparable to that of signal amplitudes, there may be ambiguity in generating the warping functions in the training phase. This may give rise to spurious peaks or valleys, which may not naturally occur in the training data. While we did not observe this behaviour in our experiments with bump functions, data generated from biological processes such as diffusion profiles may consist of segments where the signal to noise ratio may be reduced. Thus more validation of the method is needed in cases where such SNR assumptions are violated.

6. Conclusion

The proposed generative encoder-decoder network SrvfNet is capable of efficiently computing warping functions in an unsupervised manner that is also amenable to both fixed template and template prediction schemes. We validated our fixed template models on two experiments and found that the resulting warpings strongly resembled what we would expect from standard dynamic time warping approaches. Similarly, we validated our template prediction framework on both synthetic and FA data. In the synthesized bumps, we found that our predicted template closely resembled an exemplar function from the class of training data and displayed a prominent feature of two bumps. This suggests that our framework has learned the geometric structure of the data and has produced a more descriptive

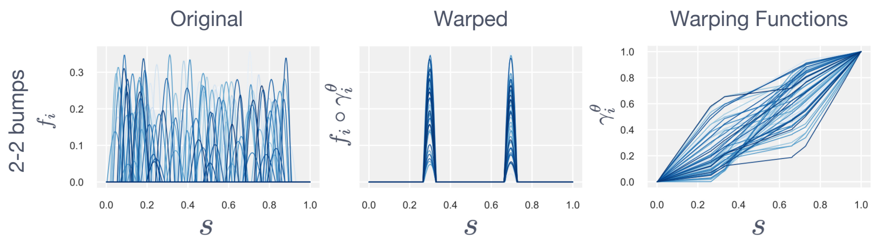


Figure 5. Warping of Two-to-Two bump functions with simultaneous template prediction.

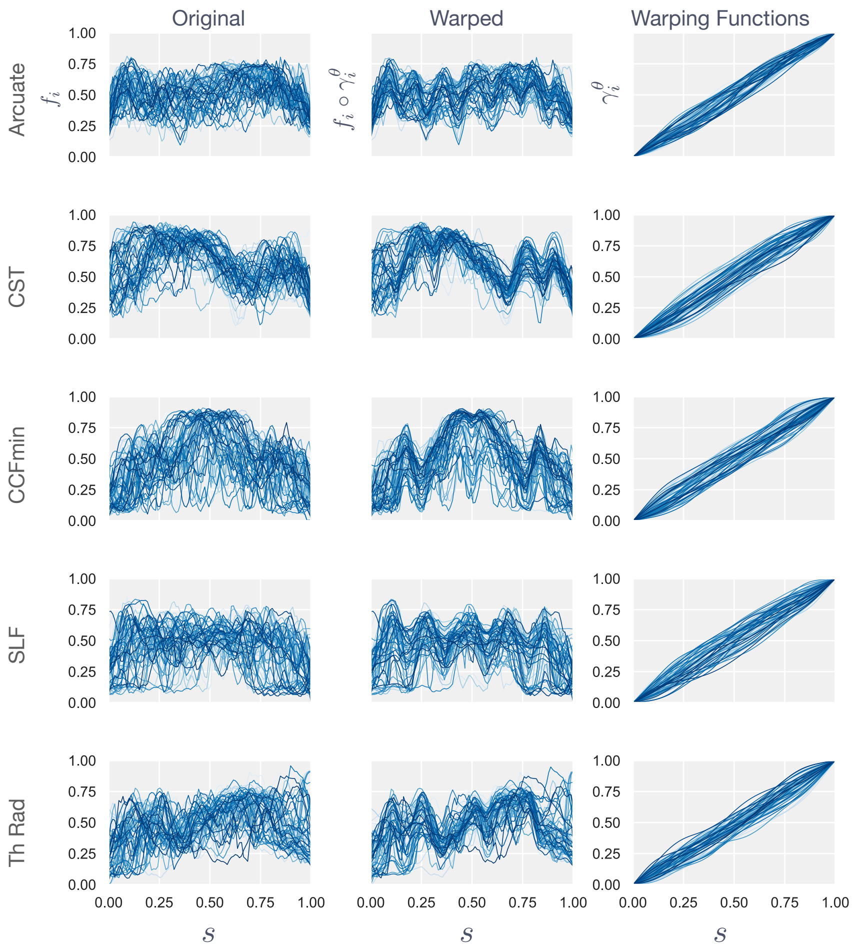


Figure 6. Warping of FA diffusion profiles with simultaneous template prediction.

template than a simple Euclidean mean, which would not necessarily preserve the geometric properties of the data. This is also demonstrated in the FA experiments where we observed distinct patterns in diffusion profiles in the warped test data compared to the original profiles while still retaining the overall structure of the original data. Importantly, our approach is not only capable of jointly aligning multiple signals and estimating a nonlinear template, but it also allows for sampling of warping functions from a space of distributions of encoded warps.

7. Acknowledgements

This research was partially supported by a fellowship from the NSF NRT Award #1829071 (EN) and the NIH NIAAA (National Institute on Alcohol Abuse and Alcoholism) awards R01-$AA026834$, R01-$AA025653$ (SHJ). We also thank Dr. Katherine L. Narr, UCLA, for providing the diffusion profile data.

References

[1] Abubakar Abid and James Y Zou. Learning a warping distance from unlabeled time series using sequence autoencoders. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[2] Oren Freifeld, Soren Hauberg, Kayhan Batmanghelich, and John W Fisher. Highly-expressive spaces of well-behaved transformations: Keeping it simple. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2911–2919, 2015. 2

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 5

[4] Shantanu H. Joshi, Eric Klassen, Anuj Srivastava, and Ian Jermyn. A novel representation for riemannian analysis of elastic curves in R^n. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, 2007. 2, 3

[5] Shantanu H. Joshi, E. Klassen, A. Srivastava, and I. Jermyn. Removing shape-preserving transformations in square-root elastic (SRE) framework for shape analysis of curves. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 387–398, 2007. 3

[6] Hermann Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30:509–541, 1977. 3

[7] Ieva Kazlauskaite, Carl Henrik Ek, and Neill DF Campbell. Gaussian process latent variable alignment learning. *arXiv preprint arXiv:1803.02603*, 2018. 1

[8] Diederik Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 12 2013. 2, 3

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015. 5

[10] Kaushik Koneripalli, Suhas Lohit, Rushil Anirudh, and Pavan Turaga. Rate-invariant autoencoding of time-series. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3732–3736, 2020. 2

[11] Suhas Lohit, Qiao Wang, and Pavan Turaga. Temporal transformer networks: Joint learning of invariant and discriminative time warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12426–12435, 2019. 2, 4

[12] Elvis Nunez and Shantanu. H. Joshi. Deep learning of warping functions for shape analysis. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3782–3790, 2020. 2

[13] Jeeheh Oh, Jiaxuan Wang, and Jenna Wiens. Learning to exploit invariances in clinical time-series data using sequence transformer networks. In *Proceedings of the 3rd Machine Learning for Healthcare Conference*, volume 85, pages 332–347, 2018. 2

[14] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978. 1

[15] Ron A Shapira Weber, Matan Eyal, Nicki Skafte, Oren Shriki, and Oren Freifeld. Diffeomorphic temporal alignment nets. *Advances in Neural Information Processing Systems*, 2019. 2

[16] Anuj Srivastava, Ian H. Jermyn, and Shantanu H. Joshi. Riemannian analysis of probability density functions with applications in vision. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 3

[17] Anuj Srivastava, Eric Klassen, Shantanu H. Joshi, and Ian H. Jermyn. Shape analysis of elastic curves in Euclidean spaces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33:1415–1428, 2011. 3

[18] Anuj Srivastava and Eric P. Klassen. *Functional and Shape Data Analysis*, volume 1. Springer, 2016. 1

[19] George Trigeorgis, Mihalis A Nicolaou, Björn W Schuller, and Stefanos Zafeiriou. Deep canonical time warping for simultaneous alignment and representation learning of sequences. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 40(5):1128–1138, 2017. 2