2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) | 978-1-6654-4899-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/CVPRW53098.2021.00503

SrvfRegNet: Elastic Function Registration Using Deep Neural Networks

Chao Chen and Anuj Srivastava
Department of Statistics, Florida State University
Tallahassee, FL 32306, USA
{chao.chen,anuj}@stat.fsu.edu

Abstract

Registering functions (curves) using time warpings (reparameterizations) is central to many computer vision and shape analysis solutions. While traditional registration methods minimize penalized-$\mathbb{L}^2$ norm, the elastic Riemannian metric and square-root velocity functions (SRVFs) have resulted in significant improvements in terms of theory and practical performance. This solution uses the dynamic programming algorithm to minimize the $\mathbb{L}^2$ norm between SRVFs of given functions. However, the computational cost of this elastic dynamic programming framework – $O(nT^2k)$ – where T is the number of time samples along curves, n is the number of curves, and $k < T$ is a parameter – limits its use in applications involving big data. This paper introduces a deep-learning approach, named SRVF Registration Net or SrvfRegNet to overcome these limitations. SrvfRegNet architecture trains by optimizing the elastic metric-based objective function on the training data and then applies this trained network to the test data to perform fast registration. In case the training and the test data are from different classes, it generalizes to the test data using transfer learning, i.e., retraining of only the last few layers of the network. It achieves the state-of-the-art alignment performance albeit at much reduced computational cost. We demonstrate the efficiency and efficacy of this framework using several standard curve datasets.

1. Introduction

The registration or temporal alignment of functional, curve, shape, or activity data has been central to many computer vision problems, including shape analysis, activity recognition, and computational anatomy. Observations of multiple objects or actions, or multiple observations of the same object or action, may differ in the execution rates, causing a misalignment between observations. When comparing such observations for clustering, classification, or modeling, one needs to temporally register or align these curves to isolate this mis-registration variability and reach more natural solutions. In functional data analysis, this problem is often referred to as phase-amplitude separation [20].

While the importance and challenges of registration have been recognized well in the past, with several theoretical and practical ideas proposed, a satisfactory solution remains elusive, especially for massive datasets. Historically, the main issue was theoretical – the most commonly-used framework for registration relied on minimizing the classical Euclidean objective function (the $\mathbb{L}^2$ norm between the functions plus a regularization term) and that had fundamental theoretical flaws. The actual optimization is performed using the dynamic programming algorithm (DPA) or its adaptations. In recent years, the researchers have overcome the theoretical limitations using elastic Riemannian metrics and their excellent invariance properties. However, the optimization tool continues to be DPA, which can be computationally expensive for massive datasets. This paper explores deep neural networks to reach an architecture that can learn registration solutions from training data and then efficiently transfer the solutions to the test datasets.

1.1. Relevant Past Literature

We start by summarizing the past salient literature on function or curve registration, pointing out their strengths, limitations, and bottlenecks.

Dynamic Time Warping (DTW) Solutions: The most common approach for aligning functions, time-series, or curves is based on minimizing the $\mathbb{L}^2$ norm between observations. It is referred to by various names, including *dynamic time warping* or simply DTW [2, 28]. Given two scalar functions $f_1, f_2 : [0,T] \to \mathbb{R}$, this approach solves for a time-warping function $\gamma : [0,T] \to [0,T]$, often a boundary-preserving diffeomorphism, that minimizes the objective function $\|f_1 - f_2 \circ \gamma\|^2$, where $\|\cdot\|$ denotes the $\mathbb{L}^2$ norm. This optimization problem is degenerate in that one can make the difference arbitrarily small using drastic warpings. One handles that issue by imposing a roughness penalty on γ and limiting the search space of γ, accord-

978-1-6654-4899-4/21/$31.00 ©2021 IEEE
DOI 10.1109/CVPRW53098.2021.00503

ing to $\min_\gamma \left(\|f_1 - f_2 \circ \gamma\|^2 + \lambda \mathcal{R}(\gamma)\right)$. The optimization problem is commonly solved using the dynamic programming algorithm or DPA [3]. However, this solution has several flaws, including a lack of inverse consistency of the solution and low registration performance. Despite these flaws, this framework continues to be popular in the literature. Although several extensions of DTW have been proposed to improve alignment quality or applicability, the main limitations persist. Variations of DTW include Canonical time warping to align multi-modal data [43, 37, 38] and Soft-DTW to solve a simpler surrogate optimization problem [29].

An important theoretical advance in the field came from a Riemannian perspective – the introduction of elastic Riemannian metric and the Square-Root Velocity Function (SRVF) representation. Developed over several papers, including [42, 31, 32], this framework changed the objective function from $\mathbb{L}^2$ to an elastic distance. While this distance had excellent invariance properties, which helped avoid degeneracy and improve registration, its original form was still too complex to be of practical use. This issue was resolved by using SRVFs of functions rather than the functions themselves. The classical $\mathbb{L}^2$ norm between SRVFs of functions equaled the elastic distance between the functions, allowing for the dynamic programming solution to be readily applied.

Despite the superior theoretical properties and excellent registration performance, the DPA-based methods suffer from two significant issues. First, the computation cost becomes prohibitive if the length of time series data or the sample size increases dramatically [39]. Second, the DTW-based solutions do not transfer easily to unseen data – one has to apply the entire procedure to any data, old or new, to obtain registration.

Deep learning based methods: In recent years, there has been an exponential rise in the use of deep neural networks in various settings, including optimizations. Focusing on the registration and alignments, Jaderberg *et al.* [9] proposed the Spatial Transformer Network (STN), which aims to learn spatial warps from the training data and apply these warps to image data to enhance classification. Similarly, several papers proposed learning solutions to 2D (or even 3D) registration [4, 21, 41, 5, 1] using deep networks. For instance, Lohit *et al.* [19] introduced a Temporal Transformer Network (TTN) to learn time warpings to help improve the classification. TTN generates warping functions in a non-parametric way but is a supervised learning framework that requires labels for both training and test data. To overcome the limited training data, Terefeet *et al.* [35] introduced the semi-supervised multitasking autoencoder that requires class labels only during training. Another approach, termed Diffeomorphic Temporal Alignment Net (DTAN) [40], seeks unsupervised registration and estimates parametric warps using convolutional neural networks (CNNs). Although DTAN provides good results in unsupervised learning, it requires large training data. Similar to our goal here, Nunez and Joshi [24] used a deep-learning approach to curve registration. However, their supervised approach requires DTW to first generate training data for the network. The network is first trained on this data and then applied to the future data to reproduce DTW-type registration.

1.2. Proposed Approach

This paper combines the strengths of the elastic Riemannian framework and the deep neural networks, resulting in a fast and effective registration of massive functional or curve data. Our approach, named Square Root Velocity Function Registration Net (SrvfRegNet), is an unsupervised, learning-based registration approach that enjoys invariance properties and provides registration of unseen test data. Another feature of SrvfRegNet is that registration is "transferable", in the sense of [6, 25, 36], and one does not require large training data to train it.

2. Background: Elastic Function Registration

In this section, we summarize the main ideas from elastic Riemannian framework for functional or curve registration.

Let $\mathcal{F}$ be the set of all smooth scalar functions on an interval $[0, T]$. (One can develop frameworks for vector- or manifold-valued functions similarly.) Let Γ be the group of boundary-preserving diffemoprhisms from $[0, T]$ to itself, with the group operation being composition and the identity element being $\gamma_{id}(t) = t$. The right action of Γ on $\mathcal{F}$ is given by the $\mathcal{F} \times \Gamma \to \mathcal{F}$ with $(f, \gamma) = f \circ \gamma$, the composition of f by γ. Let the $\mathbb{L}^2$ norm of a function f be denoted by $\|f\| = \sqrt{\int_0^T f(t)^2 \, dt}$.

Given two functions $f_1, f_2 \in \mathcal{F}$, one seeks a warping function γ such that the composition $f_2 \circ \gamma$ is aligned as well as possible to f_1. The question is: What should be the objective function to define the optimality of γ? A convenient option is $\inf_{\gamma \in \Gamma} \|f_1 - f_2 \circ \gamma\|^2$, but this leads to degenerate solutions (this phenomenon is called the *pinching effect*) resulting in extreme time warpings. In other words, one can squeeze or pinch a large part of f_2 and make this cost function arbitrarily small. To avoid this situation, one frequently adds a penalty [18, 34, 16, 27, 13]: $\inf_{\gamma \in \Gamma} \left(\|f_1 - f_2 \circ \gamma\|^2 + \lambda \mathcal{R}(\gamma)\right)$, where $\mathcal{R}$ is a roughness penalty and λ is a positive number. However this solution is also not satisfactory. Firstly, the choice of λ is an important issue. Secondly, and more importantly, this solution is not symmetric. That is, if the roles of f_1 and f_2 are reversed then the resulting registration is not consistent with the previous registration. The shortcomings of the $\mathbb{L}^2$ norm, or its penalized versions, as an objective func-

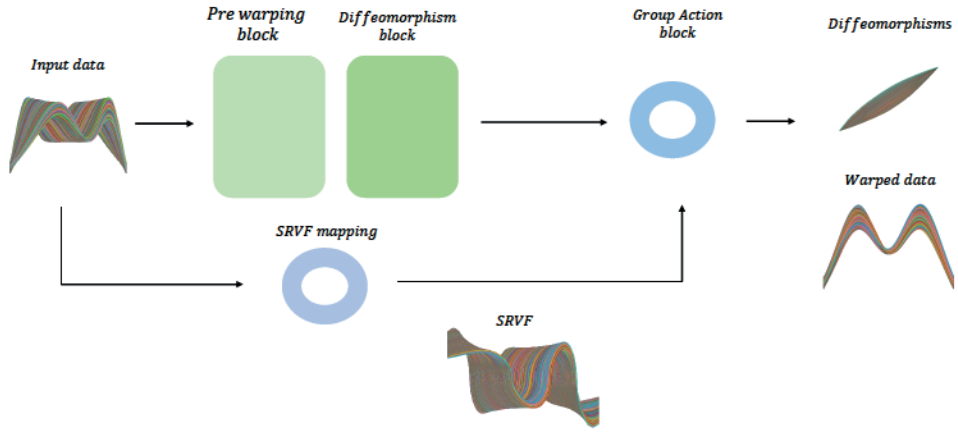

Figure 1. SrvfRegNet's architecture: The input data simultaneously passes through time-warping and SRVF-mapping blocks. The SRVF mapping computes SRVFs of input functions, and the Diffeomorphism block generates warping functions. These two are combined in the Group action block to output registered data.

tion for registering functions are more fundamental. It lacks the invariance property that is critical for use in registration tasks [33, 32]. In mathematical terms, for $f_1, f_2 \in \mathcal{F}$ and $\gamma \in \Gamma$, we have $\|(f_1 \circ \gamma) - (f_2 \circ \gamma)\| \neq \|f_1 - f_2\|$ in general. This lack of isometry under the action of Γ is the root cause of degeneracy and pinching during registration under the $\mathbb{L}^2$ norm.

A fundamentally better choice in this situation is to use an *elastic metric*. This metric, in conjunction with a mathematical representation called square-root velocity function (SRVF), provides a much better solution, in both theoretical properties and practical performance. As described in [10, 32], the SRVF of a function f is defined to be: $q(t) = \text{sign}(\dot{f}(t))\sqrt{|\dot{f}(t)|}$. We can easily map the SRVF q back to the original function f, up to a constant, using $f(t) = f(0) + \int_0^t |q(s)|q(s)ds$. One can show that for any $f \in \mathcal{F}$, its SRVF is square-integrable: $\|q\| < \infty$ or $q \in \mathbb{L}^2([0,T],\mathbb{R})$. The SRVF of a time-warped function $f(\gamma(t))$ is given by $(q \star \gamma)(t) \equiv q(\gamma(t))\sqrt{\dot{\gamma}(t)}$.

The main motivation for using SRVF in registration comes from the following invariance property: For any SRVFs $q_1, q_2 \in \mathbb{L}^2$ and $\gamma \in \Gamma$, we have that $\|q_1 - q_2\| = \|(q_2 \star \gamma) - (q_2 \star \gamma)\|$. A corollary to that result is that for any $q \in \mathbb{L}^2$ and $\gamma \in \Gamma$, the norm $\|q \star \gamma\| = \|q\|$. In other words, pinching is not possible under this metric. There are several other useful properties of this representation, including that adding a constant to a function does not change its SRVF. We refer the reader to [32] for a detailed discussion.

This setup leads to the following registration solution. Given two functions $f_1, f_2 \in \mathcal{F}$, with corresponding SRVFs $q_1, q_2 \in \mathbb{L}^2$, solve the optimization:

$$\gamma^* = \inf_{\gamma \in \Gamma} \|q_1 - q_2 \star \gamma\| \tag{1}$$

One can show that if γ^* is the optimal warping to align f_2 to f_1, then γ^{*-1} is the optimal warping to align f_1 to f_2. That is, this solution is *inverse consistent*. The infimum is approximated on a discrete-time grid using DPA. If each of the functions is sampled using T time points, then the computational cost is $O(T^2k)$, where k is typically a number much smaller than T.

To simultaneously register multiple functions $f_1, f_2, \ldots, f_n$, one first computes their mean under the elastic metric and then aligns them individually to that mean using Eqn. 1. Define the mean of given functions as the quantity:

$$\mu = \underset{q \in \mathbb{L}^2}{\text{argmin}} \sum_{i=1}^{n} \left(\inf_{\gamma_i} \|q - (q_i \star \gamma_i)\|^2 \right) . \tag{2}$$

The computation of μ is an iterative process wherein each iteration we: (1) Align the given functions to the current mean estimate and (2) Update the mean estimate using the arithmetic mean of the aligned functions (in the SRVF space). On convergence, if $\{\gamma_i^*\}$ denote the optimal time warpings inside the summation on the right, then the functions $\{f_i \circ \gamma_i^*\}$ are said to be registered.

If the n given functions are sampled at T time points each, then the computational cost of this registration process

is $O(nT^2k)$. In case T is very large, this process becomes computationally prohibitive.

3. SrvfRegNET:Elastic Registration Network

In this section, we introduce the architecture of the SrvfRegNet network designed to find an approximate but fast technique for alignment of large functional data. The network is made of several blocks with each block contributing a piece to the registration process.

3.1. Learnable Pre-warping Block

The first block takes in the input functional data (as discrete vectors) and generates latent features that are eventually mapped into warping functions. For this block, we adapt and utilize the model architecture introduced in [14, 30, 9, 40]. This block is composed of three 1-D temporal convolutional layers with 16-32-64 filters per layer (1D-CNN) [12]. Each convolutional layer is followed by one rectified linear activation function (ReLU) [23], max-pooling [22], and 1 dimensional-batch normalization (1D-BatchNorm) layer [8]. There is also one global averaging layer and a fully connected layer at the end of the block [17, 14, 30]. The dimensions of input to this block are $(n \times T)$, where n is the batch size and T is number of time points for each function.

This pre-warping block plays a role in the feature extraction mechanism using a hierarchical design. The 1D-CNN layer extracts temporal information, the ReLU step imposes a non-linear transformation on the latent features, the pooling layer downsamples the inputs, and 1D-BatchNorm enhances the stability of the training process. The global averaging layer further downsamples the features again and smooths out the noise. The fully connected layer is used to learn and form the warping functions from the latent features. It ensures that the length of each output is equal to T. Let the output of this block be denoted by $\{g_i\}$, where the length of each feature vector g_i is T.

3.2. Diffeomorphism Block

The next block is the diffeomorphism block that takes in the latent features extracted previously to form warping functions for each function individually. In this sense, it is simply a transformation block without any parameters to tune. The diffeomorphism block ensures that the resulting warping functions satisfy the properties of boundary-preserving diffemorphisms. There are two layers in this block: the warping layer and the smoothing layer.

- *Warping layer*: The warping layer constructs predicted warping functions using the formula

$$\hat{\gamma}_i(t) = T \frac{\sum_{s=0}^{t} g_i^2(s)}{\sum_{\tau=0}^{T} g_i^2(\tau)}, \quad i = 1, 2, \ldots, n\,. \tag{3}$$

and $t = 1, 2, \ldots, T$ is the time index. This equation guarantees that $\hat{\gamma}$ is montomically increasing and satisfies the boundary conditions ($\hat{\gamma}(0) = 0$ and $\hat{\gamma}(T) = T$).

- *Smoothing layer*: The warping functions generated thus far $\hat{\gamma}_i$ can lack smoothness and may lead to roughness in the output (warped) functions. Thus, we add a smoothing layer to the previous output to help preserve the geometric structure of the output. There are several ways to smooth a function and we use integration here. The smoothing layer reduces roughness of the warping functions according to:

$$\gamma_i(t) = T \frac{\sum_{s=0}^{t} \hat{\gamma}_i(s)}{\sum_{\tau=0}^{T} \hat{\gamma}_i(\tau)} \tag{4}$$

This layer results in smooth warping functions $\{\gamma_i\}$ that can now be applied to the input functional data, albeit in the SRVF space. The dimensions of the output of this stage are $(n \times T)$.

3.3. Group Action Block

This block performs the time warping of input functions, using the warping functions generated in the previous layer. Once again, this is a transformation block and does not have any parameters to optimize over in the learning stage. First it computes the SRVF of the given functions and then applies time warping on them using the equation:

$$Q_i(\gamma_i) = (q_i \star \gamma_i) = (q_i \circ \gamma_i)\sqrt{\dot{\gamma}_i}, i = 1, 2, \ldots, n\,. \tag{5}$$

3.4. Objective Function and Back Propagation

Next, we specify the objective function for training the complete network. We simplify the objective function given in Eqn. 2 to result in:

$$E(\gamma_2, \ldots, \gamma_n) = \sum_{i=1}^{n} \|Q_i(\gamma_i) - \bar{Q}\|^2, \tag{6}$$

where $\bar{Q} = \frac{1}{n}\sum_{i=1}^{n} Q_i(\gamma_{id})$. Note that this cost function represents a single iteration of the cost function specified in Eqn. 2, the solution of SRVF registration with DPA and where the solution required multiple iterations. While we use only one iteration in this paper, one can easily increase iterations by using multiple replicates of this set of blocks in the overall pipeline. The search for optimal network parameters in the learning step is performed through a gradient-descent on the objective function, *i.e.*, backpropagation. The parameters inside the pre-warping block are optimized to generate data-specific warping functions. In the pre-warping block, three 1D-CNN layers imply 7840

parameters, and the FC layer has $(T * L_{latent} + T)$ parameters, where the L_{latent} is the length of latent features generated by 1D-CNN layers and T is the time index. Thus, the total number of parameters in the SrvfRegNet is $7840 + (T * L_{latent} + T)$.

We focus on the gradient flow from the cost function to the pre-warping block, denoted by $\frac{\partial E}{\partial \gamma_i}$, and express it using the chain-rule: $\frac{\partial E}{\partial \gamma_i} = \frac{\partial E}{\partial Q_i}\frac{\partial Q_i}{\partial \gamma_i}$. The term $\frac{\partial E}{\partial Q_i}$ denotes the gradient of the cost function with respect to the warped SRVF and $\frac{\partial Q_i}{\partial \gamma_i}$ is the gradient of the warped SRVF with respect to the warping function. The expression for $\frac{\partial E}{\partial Q_i}$ is simply $\frac{\partial E}{\partial Q_i} = 2Q_i$, and the term $\frac{\partial Q_i}{\partial \gamma_i}$ is given by:

$$\frac{\partial (q_i \circ \gamma_i)\sqrt{\dot{\gamma}_i}}{\partial \gamma_i} = [\frac{\partial (q_i \circ \gamma_i)}{\partial \gamma_i}]\sqrt{\dot{\gamma}_i} + [\frac{\partial \sqrt{\dot{\gamma}_i}}{\partial \gamma_i}](q_i \circ \gamma_i) \quad (7)$$

$$= (\dot{q}_i \circ \gamma_i)\sqrt{\dot{\gamma}_i} + (q_i \circ \gamma_i)\frac{\ddot{\gamma}_i}{2\sqrt{\dot{\gamma}_i}} . \quad (8)$$

The downstream gradient of $\frac{\partial Q_i}{\partial \gamma_i}$ is composed of gradient of integral operation, a FC layer, and 1D-CNN blocks [15, 26].

4. Experimental Results

Next we present results for registration of scalar functional data using SrvfRegNet, on both simulated and popular real-world data. The SrvfRegNet was trained with Adam optimization algorithm [11] and the learning rate was set to be 0.001. These programs are implemented using Pytorch.

4.1. Alignment of Synthetic dataset

In the first experiment, we generate functional data according to the equation:

$$f_i(t) = z_{i,1}e^{\frac{-(t-1.5)^2}{2}} + z_{i,2}e^{\frac{-(t+1.5)^2}{2}}, \; t \in [-3, 3] \, , \quad (9)$$

where $z_{i,1}, z_{i,2} \sim N(0, (0.25)^2)$. Each function f_i is a bimodal function with variable peak locations and heights. A standard averaging of these functions is not a good representative of the data, and one needs registration of peaks and valleys (through time warpings) to improve statistical summaries. This model is used frequently to test functional registration algorithms in the statistics literature. We generate a training sample of size $n = 8,000$ and test data of size 2000, both from the same model. The length of each time series data is set to $T = 150$.

Figure 2 shows change in E during training. It shows that the training loss reduced significantly in 10 iterations. The time for training the network for the data of size $(8000, 150)$ is 23 seconds.

Figure 3 shows results of functional alignment obtained using the SrvfRegNet. The top row shows the results for the training data, while the bottom row is for the unseen test data. In each row, we display: (1) the original functions,

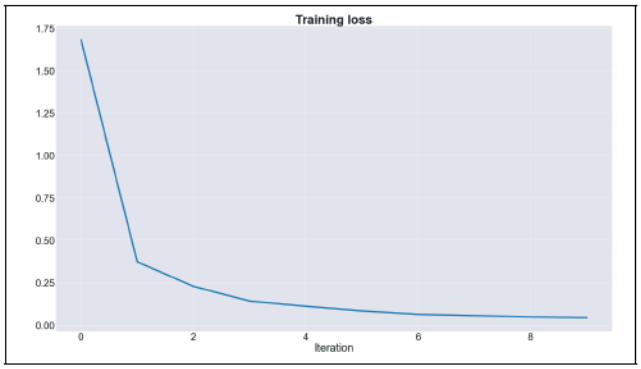

Figure 2. The training loss drops significantly in 10 iterations.

(2) their cross-sectional means, along with one-standard-deviation bands, (3) the aligned functions, and (4) the cross-sectional means of the aligned functions with one-standard-deviation bands. As these results show, the network is quite successful in aligning both training and test data. The within-class variance drops significantly from the original data to the warped data. Also, the time taken for registering the test data of size $(2000, 150)$ is only 0.42 seconds.

This experiment was conducted using an Nvidia GeForce GTX 1660 Ti graphic card. Even though this data is of modest size, the DTW method of Duncan et al. [7] takes around 3.25 hours to register the training data and needs a separate run to register the test data. Thus, the gain in time spent for alignment of test and training functions using SrvfRegNet is enormous.

As mentioned earlier, the past DPA-based techniques for functional alignment are not generalizable in the sense that they do not perform registration as training and test tasks. The registration code has to be rerun whenever new data are added. In contrast, the SrvfRegNet is trained on a training data and performs alignment on unseen data (although from the same underlying class) without a new training process. Furthermore, one can combine the idea of transfer learning with the SrvfRegNet to register functional data from a different model altogether. The use of transfer learning in SrvrfRegNet is explained in more detail later.

4.2. Alignment of Real-Life data

This section utilizes data from a well-known public repository – the UCR Time Series Classification Archive – to evaluate SrvfRegNet. We choose four datasets from different application domains, including ECGFiveDays (ECG), GunPointOldVer-susYoung (Motion), StarLightCurves (Sensor), and Yoga (Image). The computational environment and hyperparameter settings remain the same as synthetic data, but the epoch number increases to 100.

Figure 4 displays registration results obtained using

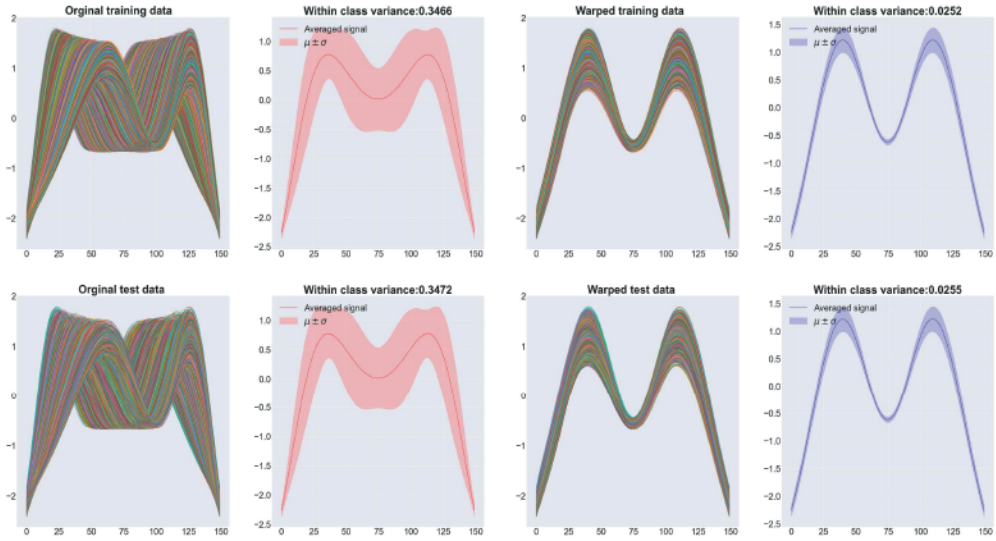

Figure 3. The first row shows the training data, and the second row shows the test data. Each row contains unaligned data, aligned data, and the corresponding functional means with one standard deviation bands.

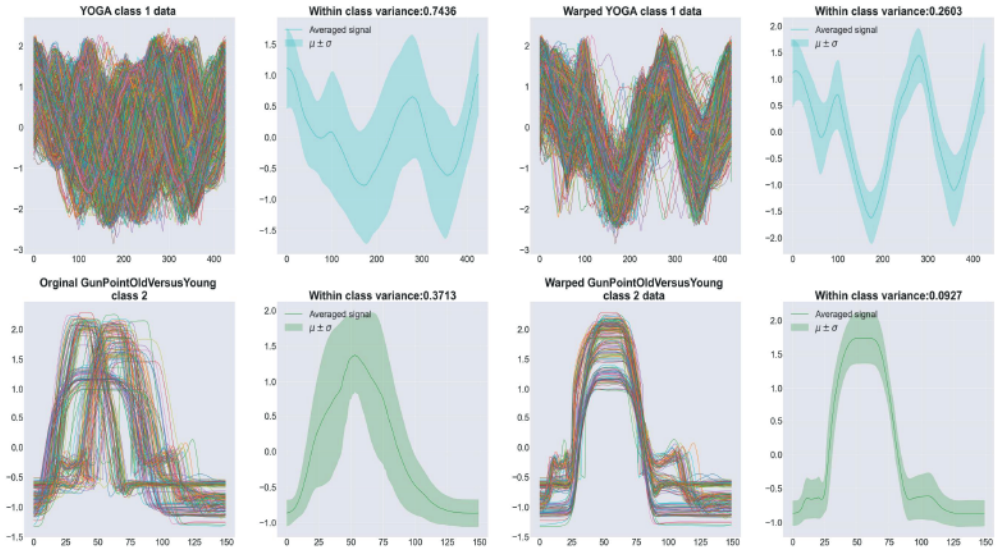

Figure 4. The figure shows the result of alignment using SrvfRegNet on the test data *GunPointOldVersusYoung* and *Yoga*.

SrvfRegNet on GunPointOldVersusYoung (top row) and Yoga datasets (bottom row). Each data has its training and test parts. The training parts of GunPointOldVersusYoung and Yoga are of sizes (137×426) and (71×150), respectively, while the test sizes are (1393×426) and (165×150), respectively. We trained SrvfRegNet with the training datasets and applied the resulting networks to the corresponding test sets. The first and third columns exhibit the original and the registered test data. The second and fourth columns show the mean functions along with a one-standard-deviation band around the mean. We can easily see that the registration improves significantly, and the with-in-class variances drop a lot due to registration.

To help visualize registration performance differently, we plot the t-SNE projections of the StarLightCurves dataset with three class labels in Fig. 5. The three classes are shown in different colors. The first column shows the original training (top) and test data (bottom), and the second column shows the aligned training (top) and test data (bottom). This figure shows that SrvfRegNet registration increases the inter-class dissimilarity and decreases the with-in-class variance. Note that the data was fed to the

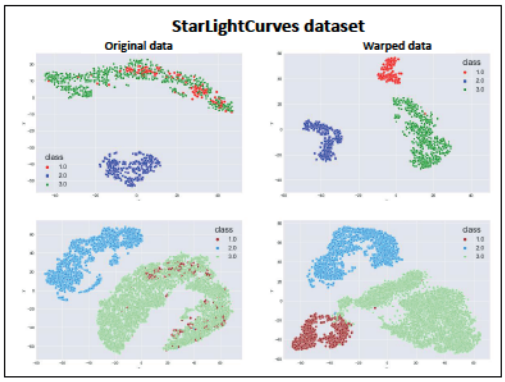

Figure 5. t-SNE visualization of the original and warped training and test data of the 3-class StarLightCurves. The first row is unaligned training data and the second row is aligned test data. Class 1 and Class 3 mixed heavily in unaligned data. We can view that how SrvfRegNet increases inter-class variance and decrease within-class variance.

network class by class, and therefore we expect the classes to separate further with the registration.

Transfer Learning (TL): Transfer learning is an approach in which a network trained for one data domain is applied, with some retraining, to data in a different domain. Specifically, one only needs to retrain the last few layers of a pre-trained model on the new data instead of training the whole network from scratch. While the classic TL only retrains the last few layers, it is sometimes beneficial to unfreeze more layers and retrain additional parameters depending on the training data's size. The more data we have, the more layers we can unfreeze and retrain.

The argument behind TL is that higher (or earlier) layers usually learn more generic features, and later layers learn specific task-related features. Thus, changing data domains necessitates retraining only the later layers. TL has been used widely for image labeling, natural language processing, and object detection. We combine TL with SrvfRegNet to improve performance in situations involving limited data for functional or curve registration. We demonstrate this approach using the ECGFiveDaysclass 1 data. The reason for choosing this dataset is its limited sample size; it has only five training samples. Fig. 6 shows how the SrvfReg-Net TL model achieves improved registration results despite small training data. Here one uses simulated data (shown in the leftmost column) to train the network, with the registration results shown in the second column. Then, we retrain two CNN blocks and one fully connected layer using the ECGFiveDaysclass 1 training data, as shown in the third and fourth columns. Finally, we apply this retrained network to the ECGFiveDaysclass 1 test data, and the last column shows the results.

	SrvfRegNet		SrvfRegNet with TL		DPA	
Data (TV)	**TV**	**Time (Tr. /Ela.)**	**TV**	**Time (Tr. /Ela.)**	**TV**	**Time (Ela.)**
ECG c1 (0.4478)	0.2561	9.21s./ 0.257s.	0.2566	**7.2s./ 0.22s.**	0.0847	93.82s.
ECG c2 (0.4857)	0.2495	30.12s./ 0.193s.	0.1995	**24.53s./ 0.22s.**	0.0671	95.96s.
GP c1 (0.2877)	0.1178	47s./ 0.20s.	0.1337	**39s./ 0.23s.**	0.1543	103.351s.
GP c2 (0.3713)	0.0877	44s./ 0.22s.	0.0927	**39s./ 0.23s.**	0.0891	43.132s
SLC c1 (0.1847)	0.039	8m.34s/ 2.54 s.	0.0385	**3m.42s. / 2.94s.**	0.0285	4.33 hr
SLC c2 (0.2237)	0.1722	13m.13s./ 4.14s	0.1719	**3m.36s./ 4.76s.**	**	18+ hr
SLC c3 (0.2685)	0.0665	24m.10s./ 9.42s	0.0701	**3m.50s./ 9.41s.**	**	18+ hr
Yg c1 (0.7436)	0.2564	2m.7s./ 1.15s.	0.2603	**1m.39s./ 1.14s.**	0.2182	1.24hr
Yg c2 (0.7362)	0.2165	2m.5s./ 1.24s.	0.2285	**1m.41s/ 1.2s.**	0.1715	1.41hr

Table 1. The table compares TVs of original datasets(underneath the name of data) with losses of aligned datasets. The Tr. and Ela. are training time and elapsed time. We list three alignment models : SrvfRegNet, SrvfRegNet with TL, and DPA, and examine their performances by measuring their losses and computational time. ** means unavailable.

Quantitative Evaulation: Next we apply SrvfRegNet to a number of public datasets and compare their results with the DPA-based registration. We evaluate the alignment performance of SrvfRegNet in three ways: visualization of the alignment, compare computational costs, and use a quantifiable registration metric. The metric that we use to measure alignment is the total variance (TV) under the $\mathbb{L}^2$ norm: $\frac{1}{n}\sum_{i=1}^{n}\|(f_i \circ \gamma_i) - \frac{1}{n}\sum_{i=1}^{n}(f_i \circ \gamma_i)\|^2$. Table 1 lists TVs obtained from SrvfRegNet, SrvfRegNet with TL, and the traditional Dynamic Programming Algorithm (DPA). These algorithms are applied to the training and the test data, and we calculate the total variance in each case. As the table shows, the SrvfRegNet (with and without TL) is able to provide total variance that is comparable to the DPA solution but an order of magnitude faster in execution. This is a remarkable accomplishment for a fully automatic, pre-trained solution to provide large-scale registrations at such efficient rates.

To further investigate the utility of transfer learning in function registration, we develop another model called the *adjusted-SrvrfRegNet*. The adjusted-SrvfRegNet model is composed of three additional CNN blocks and one FC layer to the SrvfReg-Net. That is, the SrvfRegNet and SrvfReg-Net with TL are made of three CNN blocks and one FC layer, and the adjusted-SrvfRegNet TL model contains five CNN blocks and two FC layers. Table 2 presents registration results for the StarLightCureves dataset for SrvfReg-Net, SrvfRegNet with TL, and adjusted-SrvfRegNet with TL. The SrvfRegNet is trained on the SLC training data and performs the registration on the test data. SrvfRegNet

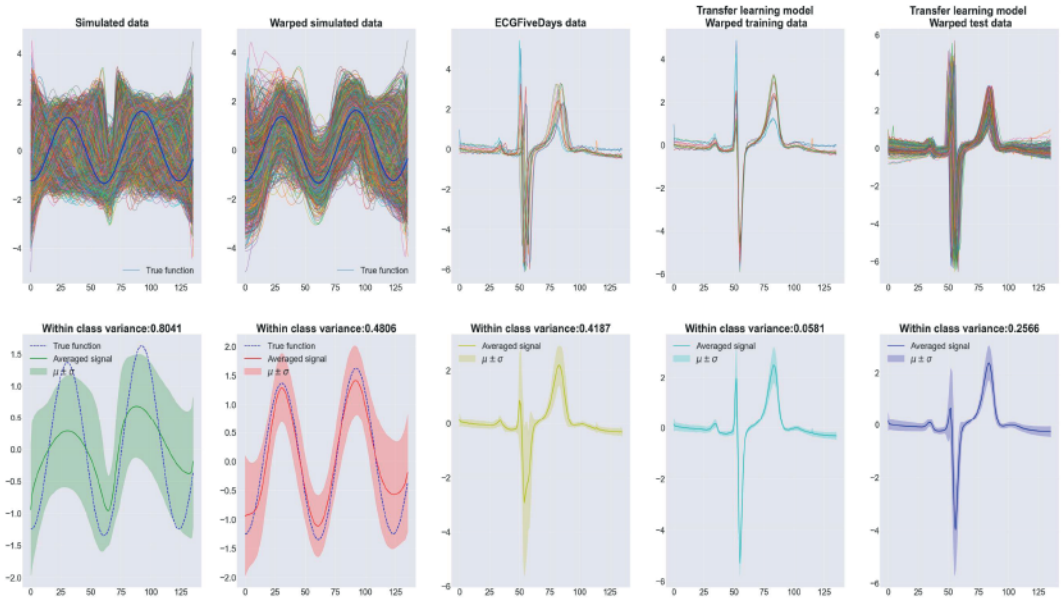

Figure 6. The first column is simulated data used to train SrvfRegNet, the second is warped training data, the third is the original **ECGFiveDays** retraining data, the fourth is registered retraining data and the fifth is registered test data.

	Unaligned data loss	SrvfRegNet	SrvfRegNet with TL (Unfreeze 2 CNN blocks & 1 FC layer)	Adjusted SrvfRegNet with TL (Unfreeze 2 FC layers)
SLC c1	0.1847	0.0390	0.0387	0.0385
SLC c2	0.2237	0.1722	0.1719	0.1689
SLC c3	0.2687	0.0665	0.0701	0.0648

Table 2. The adjusted SrvfRegNet with TL model outperforms the SrvfRegNet and SrvfRegNet with TL. The result highlights that we can obtain better alignment by only retrain two layers from the pre-trained adjusted SrvfRegNet.

with TL is retrained in the last two CNN blocks and one FC layer. The adjusted-SrvfRegNet wih TL is retrained in only the last two FC layers. We can see that the adjusted-SrvfRegNet with TL outperforms the other two models in the registration performance across all SLC classes.

We summarize results from Tables 1 and 2 as follows:

Total Variance: In terms of the total variance of the registered data, the DPA method performed better on ECG (ECGFiveDays) and Yg (Yoga) while SrvfRegNet did better on GP (GunPointOldVersusYoung) and SLC (StarLightCurves). Both these methods reduce total variance by a lot when compared to the original data. Note that the DPA could not handle SLC c2 and SLC c3 datasets due to their large sample size and lengths.

Computational Efficiency: The biggest advantage of SrvfRegNet over DP is in computational efficiency. The time difference between the two methods is less for smaller datasets, such as ECG c1, ECG c2, GP c1 and, GP c2 as their sizes are (428×136), (433×136), (150×150), and (165×150) respectively. However, the times differences are substantial in medium-size datasets, such as Yg c1, Yg c2, SLC c1, SLC c2, and SLC c3, where the sizes are (1393×426), (1607×426), (1177×1024), (2305×1024), (4754×1024), respectively. The SrvfRegNet is around 42 times faster than DPA on Yg dataset. The DP could not even be applied to the SLC c2 and c3 datasets due to their length and sample size. Overall, SrvfRegNet runs much faster than the DP model; and The SrvfRegNet model with TL ran faster than the SrvfRegNet model.

Generalization and transfer learning: The SrvfRegNet has the ability to train on one data and apply it on unseen test data. The DPA method does not have this property. The transfer learning with SrvfRegNet can be a key to the situation where the datasets are small and one needs registration on large unseen future data.

5. Summary

This paper develops the SrvfRegNet – a deep learning-based method – that combines the strengths of the elastic Riemannian framework with the efficiency of neural networks. The SrvfRegNet performs fast, registration on large test data and provides a good generalization to unseen future data. Additionally, the paper introduces transfer learning that integrates domain transfer with the SrvfRegNet to improve registration performances when the training data are limited.

References

[1] Guha Balakrishnan, Amy Zhao, Mert R Sabuncu, John Guttag, and Adrian V Dalca. An unsupervised learning model for deformable medical image registration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9252–9260, 2018. 2

[2] Richard Bellman and Robert Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959. 1

[3] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995. 2

[4] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[5] Bob D de Vos, Floris F Berendsen, Max A Viergever, Marius Staring, and Ivana Išgum. End-to-end unsupervised deformable image registration with a convolutional neural network. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 204–212. Springer, 2017. 2

[6] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR, 2014. 2

[7] Adam Duncan, A Srivastava, Xavier Descombes, and E Klassen. Geometric analysis of axonal tree structures. *DIFFCV: Differential Geometric Techniques in Computer Vision*, 2015. 5

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 4

[9] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *arXiv preprint arXiv:1506.02025*, 2015. 2, 4

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

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[12] Serkan Kiranyaz, Turker Ince, Ridha Hamila, and Moncef Gabbouj. Convolutional neural networks for patient-specific ecg classification. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2608–2611. IEEE, 2015. 4

[13] Alois Kneip and Theo Gasser. Statistical tools to analyze data representing a sample of curves. *The Annals of Statistics*, pages 1266–1305, 1992. 2

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 4

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 5

[16] Xiaoyan Leng and Hans-Georg Müller. Time ordering of gene coexpression. *Biostatistics*, 7(4):569–584, 2006. 2

[17] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 4

[18] Xueli Liu and Hans-Georg Müller. Functional convex averaging and synchronization for time-warped random curves. *Journal of the American Statistical Association*, 99(467):687–699, 2004. 2

[19] 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

[20] James S. Marron, James O. Ramsay, Laura M. Sangalli, and Anuj Srivastava. Functional data analysis of amplitude and phase variation. *Statistical Science*, 30(4):468–484, 2015. 1

[21] Tony C.W. Mok and Albert C.S. Chung. Fast symmetric diffeomorphic image registration with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[22] Jawad Nagi, Frederick Ducatelle, Gianni A Di Caro, Dan Cireşan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber, and Luca Maria Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347. IEEE, 2011. 4

[23] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 4

[24] Elvis Nunez and Shantanu H Joshi. Deep learning of warping functions for shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 866–867, 2020. 2

[25] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. 2

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[27] James O Ramsay and BW Silverman. Functional data analysis. *İnternet Adresi: http*, 2008. 2

[28] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978. 1

[29] David Schultz and Brijnesh Jain. Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces. *Pattern Recognition*, 74:340–358, 2018. 2

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[31] Anuj Srivastava, Eric Klassen, Shantanu H Joshi, and Ian H Jermyn. Shape analysis of elastic curves in euclidean spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1415–1428, 2010. 2

[32] Anuj Srivastava and Eric P Klassen. *Functional and shape data analysis*, volume 1. Springer, 2016. 2, 3

[33] Anuj Srivastava, Wei Wu, Sebastian Kurtek, Eric Klassen, and James Stephen Marron. Registration of functional data using fisher-rao metric. *arXiv preprint arXiv:1103.3817*, 2011. 3

[34] Rong Tang and Hans-Georg Müller. Pairwise curve synchronization for functional data. *Biometrika*, 95(4):875–889, 2008. 2

[35] Tsegamlak Terefe, Maxime Devanne, Jonathan Weber, Dereje Hailemariam, and Germain Forestier. Time series averaging using multi-tasking autoencoder. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1065–1072. IEEE, 2020. 2

[36] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010. 2

[37] George Trigeorgis, Mihalis A Nicolaou, Björn W Schuller, and Stefanos Zafeiriou. Deep canonical time warping for simultaneous alignment and representation learning of sequences. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1128–1138, 2017. 2

[38] George Trigeorgis, Mihalis A Nicolaou, Stefanos Zafeiriou, and Bjorn W Schuller. Deep canonical time warping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5110–5118, 2016. 2

[39] Lusheng Wang and Tao Jiang. On the complexity of multiple sequence alignment. *Journal of computational biology*, 1(4):337–348, 1994. 2

[40] Ron Shapira Weber, Matan Eyal, Nicki Skafte Detlefsen, Oren Shriki, and Oren Freifeld. Diffeomorphic temporal alignment nets. In *NeurIPS*, pages 6570–6581, 2019. 2, 4

[41] Xiao Yang, Roland Kwitt, Martin Styner, and Marc Niethammer. Quicksilver: Fast predictive image registration–a deep learning approach. *NeuroImage*, 158:378–396, 2017. 2

[42] Laurent Younes. Computable elastic distance between shapes. *SIAM Journal of Applied Mathematics*, 58(2):565–586, 1998. 2

[43] Feng Zhou and Fernando De la Torre. Generalized canonical time warping. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):279–294, 2015. 2