Deep Learning of Warping Functions for Shape Analysis

Elvis Nunez[1,2], and Shantanu H. Joshi[2,3]
[1]Department of Applied Mathematics and Statistics, Johns Hopkins University
[2]Ahmanson Lovelace Brain Mapping Center, Department of Neurology, UCLA
[3]Department of Bioengineering, UCLA
elvis_nunez@jhu.edu s.joshi@g.ucla.edu

Abstract

Rate-invariant or reparameterization-invariant matching between functions and shapes of curves, respectively, is an important problem in computer vision and medical imaging. Often, the computational cost of matching using approaches such as dynamic time warping or dynamic programming is prohibitive for large datasets. Here, we propose a deep neural-network-based approach for learning the warping functions from training data consisting of a large number of optimal matches, and use it to predict optimal diffeomorphic warping functions. Results show prediction performance on a synthetic dataset of bump functions and two-dimensional curves from the ETH-80 dataset as well as a significant reduction in computational cost.

1. Introduction

A key ingredient in rate or parameterization-invariant matching of shapes of one-dimensional functions or curves is a cost function (either an L_1 or an L_2 norm) that includes a nonlinear one-one differentiable function with a differentiable inverse (diffeomorphism) that controls the rate of change of one function (shape) with respect to the other. This transformation can be conveniently described by a group action if an appropriate representation or mapping of shapes or functions is defined, or by a direct non-linear transformation of the coordinates of the curves or graphs of functions. In both cases, the transformation results in a composition of a shape with a diffeomorphic warping function. While the full problem of shape analysis for curves also involves an invariant matching over a set of translations, scalings, and rotations, the main computation for both curves and one-dimensional functions requires an optimization over a set of diffeomorphic warping functions. There have been several algorithms proposed to solve this optimization problem. A classical, widely-used algorithm is dynamic time warping (DTW), which is amenable to a global solution using dynamic programming [13]. This algorithm has a computational complexity of $O(T^2)$ for T samples or points along the shape. In this paper, instead of solving the dynamic programming (DP) problem explicitly, we outline a deep learning (DL) framework for predicting the warping functions by learning from pairwise matches from a training dataset.

1.1. Motivation

Our motivation for learning to predict reparameterizations is multifold.

Computational cost: Direct computation of alignment, either exclusively using dynamic time warping or by optimizing a cost function to find geodesics as in [3, 4, 14], incurs a computational cost. This cost is negligible for small-sized functional or curve-shape datasets with sizes on the order of a few thousands. However, for an order increase of tenfold, hundredfold, or larger (data sizes $> 100K$ or > 1 million), the computational cost becomes prohibitive. In real-world applications, such data arise from biological shapes [2, 5], clinical time-series data of heart rates [11], or functional magnetic resonance imaging signals (fMRI) [8]. Thus, we suggest that a prediction or an evaluation-type of an approach has the potential to yield sizable cost savings as compared to a direct computation-type of an approach. In spirit, this motivation is similar to the idea proposed by Yang et al. who predict the momentum-parameterization of diffeomorphisms using a deep learning approach to achieve a speedup in image registration [17].

Shape learning and classification: As proposed by Thompson [15], the task of comparing the deformations of the objects, rather than a precise definition of the object itself, often yields more interesting information. The temporal warps or spatial reparameterizations (diffeomorphisms) can be analyzed under a statistical learning framework including tangent-space principal component analysis or linear discriminant analysis. Under a deep-learning framework for warping functions, one would not only have the predicted warping function, but also the weights associated with the underlying model training from the population. Thus the

goal of shape learning or shape understanding may potentially benefit from having more information about the population. We also note that Lohit et al. [10] aims to achieve this by jointly learning the time warping functions associated with temporal human activity. There, the approach is classification-focused and is trained to give the best performance with respect to classification.

Novel descriptors: Finally, the output of the intermediate layers can serve as a rich feature descriptor for the shape of the population. This can be achieved by following the process of *feature extraction* and obtaining the projection of a novel shape as a mapping of the intermediate layers of the neural network, or by modeling the ensembles of the activation layer for a population level shape descriptor.

1.2. Background and related work

Several ideas have been proposed for learning-based prediction of the warping functions. Here, we outline them in brief. Kazlauskaite et al. proposed a method for automatically learning functional alignments based on a probabilistic model built on non-parametric priors [6]. Another idea, deep canonical time warping (DCTW) for learning warping functions (for multiple time-series) uses deep learning to achieve simultaneous temporal alignment and maximal correlation of time-series in a common subspace [16]. The work by Oh et al. [12] uses a sequence transformer network to learn linear transformations along the temporal axis (stretch, compress, flip and/or shift the signal) to identify and account for invariances in clinical time-series data. A more recent approach by Lohit et al. [10] uses a temporal transformer network (TTN) to learn warping functions in the context of classification. Their network performs learning as well as class-aware discriminative alignment jointly for time-series classification including action trajectories by reducing the intra-class variability and also increasing the inter-class separation. Notably, in their framework the prediction of the warping functions is achieved without an explicit template for the class.

1.3. Contributions

The contributions of this paper are as follows. We propose a deep learning based framework for predicting diffeomorphic warps giving rise to invariant matching of one-dimensional functions and two-dimensional curves. The network architecture is simple and consists of a convolutional layer followed by three dense layers. We propose a choice of three loss functions that measure discrepancies between i) warping functions, ii) linear combination of the coordinates (graphs in case of functions) and warping functions, and a iii) linear combination of the shape representations and warping functions, and demonstrate that the latter yields the best performance. We train the network on warping functions obtained using dynamic programming and show prediction results for a large set (~500K) of data for synthetic bump functions and two-dimensional curves from the ETH-80 dataset [9].

Our architecture differs from [10] in the following ways: i) we permit negative inputs into the fully connected layers by using leaky reLU activation functions with parameter 0.1, ii) we train our network directly on the shapes and warping functions to minimize loss functions that penalize shape matching differences as opposed to minimizing loss functions that penalize classification errors, and iii) we do not enforce a positive monotonicity in the network output and instead try to learn this constraint.

Additionally, different from the approach in [10], we achieve reparameterization-invariant matching for pairs of curves by integrating loss functions that aim to find the shortest distance between points in a Hilbert sphere.

This paper is organized as follows. Section 2 outlines the preliminaries for the shape representation and matching problem. Section 3 outlines the deep learning architecture including the choice of loss functions, followed by results in section 4 and discussion in section 5.

2. Shape representation preliminaries

Throughout this paper, we will consider a parameterized representation for two-dimensional curves and functions. We will let p denote a parameterized curve such that $p : D \equiv [0, 2\pi] \to \mathbb{R}^2$. In this paper we will only consider those ps that are differentiable and their first derivative is in $L^2(D, \mathbb{R}^n)$. For a one-dimensional function, we assume that $p : D \to \mathbb{R}$ with a slight abuse of notation. Though the following discussion is for two-dimensional curves, the theory holds for one-dimensional functions. For the purpose of studying the shape of p, we will represent it using the square-root velocity function (SRVF) [3, 4, 14] defined as $q : D \to \mathbb{R}^n$, where $q(t) = \dot{p}(t)/\sqrt{\|p(\dot{t})\|}$. We note that for every $q \in L^2(D, \mathbb{R}^n)$ there exists a curve p (unique up to a translation) such that the given q is the SRVF function of that p. This curve is recoverable using the equation: $p(t) = \int_0^t q(s)\|q(s)\|ds$.

To achieve invariance to scale, we re-scale all curves to be of length 2π. This causes the SRVF functional representation for these curves to be identified as elements of a hypersphere in the Hilbert manifold $L^2(D, \mathbb{R}^2)$. In this paper, we will use the notation $\mathcal{C}^o$ to denote this hypersphere.

We impose a standard L^2 metric on the tangent space of this hypersphere as follows. Since $\mathcal{C}^o$ is a sphere in $L^2([0, 2\pi], \mathbb{R}^n)$, its tangent space at a point q is given by: $T_q(\mathcal{C}^o) = \{v \in L^2([0, 2\pi], \mathbb{R}^n) | \langle v, q\rangle = 0\}$. Here, $\langle v, q\rangle$ denotes the inner product in $L^2([0, 2\pi], \mathbb{R}^n)$: $\langle v, q\rangle = \int_0^{2\pi} (v(t).q(t))_{\mathbb{R}^n} dt$. In this paper, we deal with one-dimensional functions ($n = 1$) and two-dimensional curves

$(n = 2)$. This standard metric on $L^2([0, 2\pi], \mathbb{R}^n)$ restricts to one on $\mathcal{C}^o$ and is used to compute geodesics between shapes.

Representing a parameterized curve $p(t)$ by its SRVF function $q(t)$, and imposing the constraint $\int_D \langle q(t), q(t)\rangle \, dt = 2\pi$, makes it invariant to translation and scaling. Further, the rotation and the re-parameterization variability is accounted for as follows.

For two-dimensional curves, a rotation is an element of $SO(2)$, the special orthogonal group of 2×2 matrices, and a re-parameterization is an element of Γ, the set of all orientation-preserving diffeomorphisms of D. The rotation and re-parameterization of a curve p are both denoted by the actions of $SO(2)$ and Γ on its SRVF. While the action of $SO(2)$ is denoted by multiplication: $SO(n) \times \mathcal{C}^o \to \mathcal{C}^o, \quad (O, q(t)) = Oq(t)$, the action of Γ is derived as follows. For a $\gamma \in \Gamma$, the composition $p \circ \gamma$ denotes its re-parameterization; the SRVF of the re-parameterized curve is given by $q(\gamma(t))\sqrt{\dot{\gamma}(t)}$, where q is the SRVF of p. This gives us the action $\Gamma \times \mathcal{C}^o \to \mathcal{C}^o, \quad (q, \gamma) = (q \circ \gamma)\sqrt{\dot{\gamma}}$.

Next, we enable comparisons between functions by computing the shortest path between them. Since the space $\mathcal{Q}$ is a Hilbert sphere, the shortest path between two points (shapes) q_1 and q_2 can be expressed analytically as,

$$\chi_t(q_1; v) = \cos\left(t \cos^{-1}\langle q_1, q_2\rangle\right) q_1 + \sin\left(t \cos^{-1}\langle q_1, q_2\rangle\right) v, \tag{1}$$

where $t \in [0, 2\pi]$ and the initial tangent vector $v \in T_{q_1}(\mathcal{Q})$ is given by $v = q_2 - \langle q_1, q_2\rangle q_1$. Then the shortest distance between the two shapes q_1 and q_2 in $\mathcal{Q}$ is given by

$$d(q_1, q_2) = \int_0^{2\pi} \sqrt{\langle \dot{\chi}_t, \dot{\chi}_t\rangle} dt. \tag{2}$$

The distance in Eqn. 2 can be made invariant by searching over all reparaemterizations γ as

$$d_\gamma(q_1, q_2) = \min_\gamma d(q_1, \sqrt{\dot{\gamma}} q_2 \circ \gamma). \tag{3}$$

We use dynamic programming to minimize Eqn. 3 to find the optimal reparameterization γ_{DP} as the minimizer $\gamma_{DP} = \arg\min_\gamma d(q_1, \sqrt{\dot{\gamma}} q_2 \circ \gamma)$.

In the following discussion with a slight abuse of notation, we refer to the γ_{DP} obtained using dynamic programming as γ without the subscript DP and will refer to the predicted warping function using deep learning as $\hat{\gamma}$. Next, we outline the framework for learning this warping function by considering a training dataset of pairwise matchings.

3. Deep Learning Architecture

In an effort to minimize training time, we constructed our network to be relatively simple with sufficient complexity to be applicable to varying datasets. Indeed, we use the same architecture across various datasets as discussed further in section 4.

Drawing inspiration from TTN [10], our network also consists of a convolutional layer. However, this is then followed by three dense layers as illustrated in Figure 1.

As presented, this network operates on two-dimensional curves; that is, given $p_1 : D \to \mathbb{R}^2$ and $p_2 : D \to \mathbb{R}^2$, the network aims to find the optimal diffeomorphism mapping p_2 to p_1. The curves p_1 and p_2 are first downsampled by selecting T evenly spaced points in the interval $[0, 2\pi]$ and then evaluating the curves at these points. As such, p_1 and p_2 can be regarded as matrices each of size $T \times 2$. With a slight abuse of notation, we refer to these matrix representations of p_1 and p_2 as p_1 and p_2. Concatenating the two matrices yields $[p_1, p_2] \in \mathbb{R}^{T \times 4}$. This is what is then fed through the network.

The input is first passed through a convolutional layer with 32 filters of size 3×3 and a unit stride. The output of this layer is then fed through the standard reLU activation function. After flattening this output, it is then fed through two successive fully-connected layers of sizes 256 and 128, respectively. Both of these layers have a leaky reLU activation function with parameter 0.1 and both are followed by dropout layers with a drop probability of 0.25. The output layer has T neurons and, to restrict the output to be in D, the activation function is given by $\Omega(x) = 2\pi\sigma(x)$ where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

For $p_1, p_2 : D \to \mathbb{R}$, the network remains the same except the convolution layer now has filters of size 2×2 and the input to the network is $[p_1, p_2] \in \mathbb{R}^{T \times 2}$.

In contrast to [10], we do not enforce a non-decreasing output but rather try to learn it from the data. Moreover, our network aims to reproduce warps obtained from DTW rather than warps that yield the best classification results.

3.1. Choice of loss functions

Given discretized curves p_1 and p_2, our network outputs $\hat{\gamma}$, which is an estimate of the γ obtained by solving

$$\gamma = \arg\min_{\bar{\gamma}} ||q_1 - \sqrt{\dot{\bar{\gamma}}}(q_2 \circ \bar{\gamma})||_2^2 \tag{4}$$

where q_1 and q_2 are the SRVF representations of p_1 and p_2, respectively.

Since we desire $\hat{\gamma}$ to be as close to γ as possible, it is natural to consider their squared ℓ_2 difference as a measure of similarity. As such, one possible loss function is

$$\mathcal{L}_1(\gamma, \hat{\gamma}) = ||\gamma - \hat{\gamma}||_2^2. \tag{5}$$

After warping p_2, we expect the distance between p_1 and this warped p_2 to be smaller than that of p_1 and p_2. As such, we can also aim to minimize $||p_1 - p_2 \circ \hat{\gamma}||_2^2$ with respect to

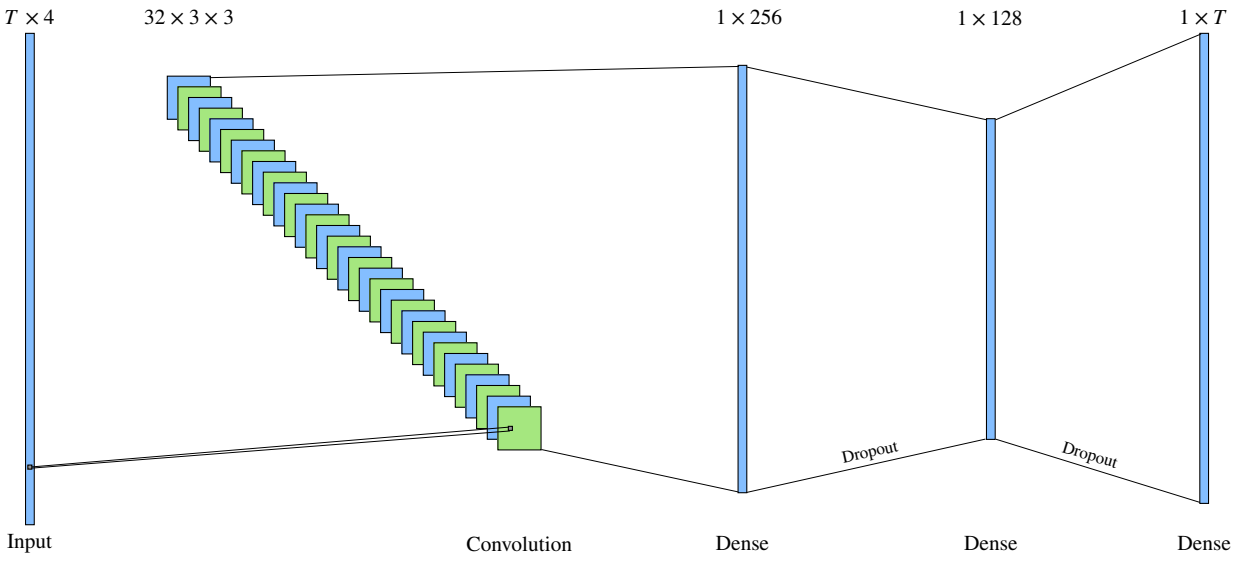


Figure 1. Deep neural architecture of the prediction network.

$\hat{\gamma}$. However, this loss does not make use of the true value γ, rendering the use of deep learning moot. As such, we can simply consider a linear combination of this and $\mathcal{L}_1$:

$$\mathcal{L}_2(\gamma, \hat{\gamma}) = a||p_1 - p_2 \circ \hat{\gamma}||_2^2 + b||\gamma - \hat{\gamma}||_2^2. \quad (6)$$

where $a, b \in \mathbb{R}_{>0}$.

Since the true γ is obtained by solving (4), it makes sense to include this function in our loss. As such, the final loss function we consider is again a linear combination of this function and $\mathcal{L}_1$

$$\mathcal{L}_3(\gamma, \hat{\gamma}) = a||q_1 - \sqrt{\dot{\hat{\gamma}}}(q_2 \circ \hat{\gamma})||_2^2 + b||\gamma - \hat{\gamma}||_2^2. \quad (7)$$

Because we do not impose a nonnegativity constraint on $\dot{\hat{\gamma}}$, instead of considering loss (7), we will consider the equivalent (8) for numerical stability

$$\mathcal{L}_3(\gamma, \hat{\gamma}) = a||q_1 - \frac{(\dot{p_2} \circ \hat{\gamma})\dot{\hat{\gamma}}}{\sqrt{|(\dot{p_2} \circ \hat{\gamma})\dot{\hat{\gamma}}|}}|| + b||\gamma - \hat{\gamma}||_2^2. \quad (8)$$

4. Results

4.1. Data Generation

We consider two separate applications of time warping: to one and two-dimensional curves. One-dimensional curves were synthesized by appending sinusoidal waves of varying phase and amplitudes. We refer to these curves as *bumps* and characterize them by their number of peaks. Examples of one, two, three, and four-bump curves is given in figure 2.

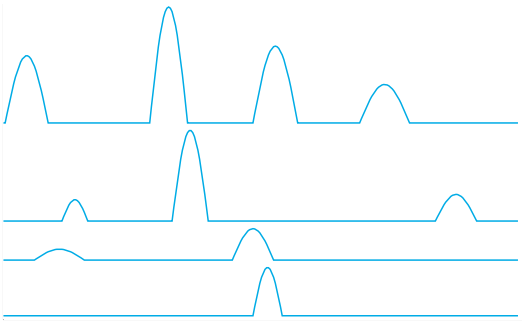


Figure 2. Examples of synthesized bumps.

Each curve was discretized to $T = 300$ points. Amplitudes were sampled uniformly on the interval $[0.15, 3]$ and wave lengths were chosen to be a percentage of T chosen uniformly on the interval $[5, 10]$.

Two-dimensional curves were constructed by first modifying the contours in the ETH-80 [9] dataset. In particular, each contour was manually inspected and modified so that the curve did not exhibit any holes. Figure 3 illustrates this process on an example of a cow contour from the dataset. This process yields closed curves in $\mathbb{R}^2$ with no holes.

Since the dataset consists of PNG images, we first binarize the image and then apply the Moore-Neighbor tracing algorithm to extract the curve boundary. This process gives a set of points in $\mathbb{R}^2$. Using linear interpolation, the curve is downsampled to $T = 300$ points so that the curve is an element of $\mathbb{R}^{300\times 2}$. This process is applied to each of the modified ETH-80 curves, yielding 3280 curves belonging to one of eight evenly distributed classes: apple, car, cow, cup, dog, horse, pear, and tomato. Within each

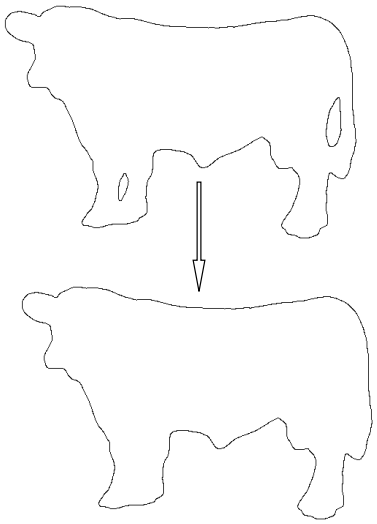


Figure 3. Hole removal from ETH-80 contour dataset.

class, each curve is enumerated according to its original file name; afterwards, every possible image pair $(i, j), j > i$ is considered. Since each class has 410 curves, there are $\frac{410*409}{2} = 83845$ such pairs. Across all 8 classes, this gives a total of 670760 curve pairs.

For each curve pair (i, j), the optimal diffeomorphism that warps curve j to curve i was computed using DTW. With the curves and diffeomorphisms at hand, the data is split into training/validation/testing in a 70/10/20 split, respectively, for a total of 469532 training pairs.

In the one-dimensional case, for each bump curve we generate 650000 random pairs so as to be consistent with the size of the two-dimensional dataset. The optimal alignment diffeomorphism is then found using DTW for each pair and the same training/validation/test split is applied. This is repeated for one, two, three, and four-bump curves so that we have 450000 training pairs for each bump class.

4.2. Loss function performance

Using the one-bump dataset, we trained the network using the $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_3$ losses and used $a = 1$ and $b = \frac{1}{4}$ for $\mathcal{L}_2$ and $\mathcal{L}_3$. Figure 4 depicts the average value of the $\mathcal{L}_1$ loss on the validation data when trained using each loss. We see that the $\mathcal{L}_3$ loss gives outputs that are most similar, in the ℓ_2 sense, to the desired output. Consequently, we use the $\mathcal{L}_3$ loss to train our network on the remaining bump datasets. The $\mathcal{L}_3$ loss also gives the best performance on the two-dimensional dataset as depicted in figure 5.

4.3. Model performance

Figure 6 illustrates the performance of our network trained on one, two, three, and four-bump datasets separately using the $\mathcal{L}_3$ loss

The first row plots γ (in green), $\hat{\gamma}$ (in red) and γ_I (in blue) where $\gamma_I(t) = t$. The second row is a correspondence plot between p_1 and p_2 as determined by γ_I. The third row is a correspondence plot between p_1 and p_2 as determined by γ and the fourth row is a correspondence plot between p_1 and p_2 as determined by $\hat{\gamma}$.

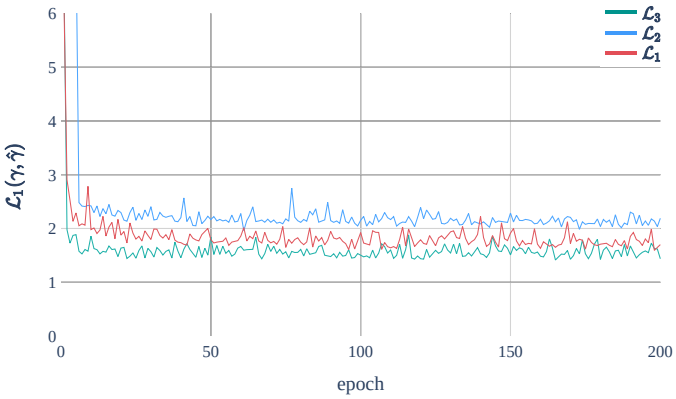


Figure 4. Average $\mathcal{L}_1$ loss on 1-d validation data.

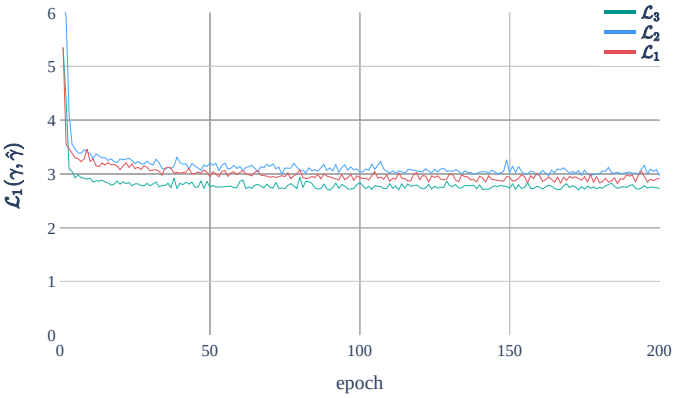


Figure 5. Average $\mathcal{L}_1$ loss on 2-d validation data.

Similarly, figure 7 illustrates the performance of our network when trained on the full 2d-dataset consisting of all classes.

Figures 8 and 9 show the average error $\mathcal{L}_1$ for the one-dimensional (bump) data and the two-dimensional curves for different bump and curve types. It is observed that in the case of bumps, the lowest error is obtained for the one-bump case, whereas the highest error is obtained for the three-bump case, with the two and three-bump cases giving similar errors. For curves, rotund shapes such as apples, cups, pears, and tomatoes yielded lower errors compared to shapes with articulated features such as cows, dogs, and horses.

4.4. Implementation and computational cost

The network was trained with a batch size of 32 for 200 epochs using an Adam optimizer [7] with a learning rate of

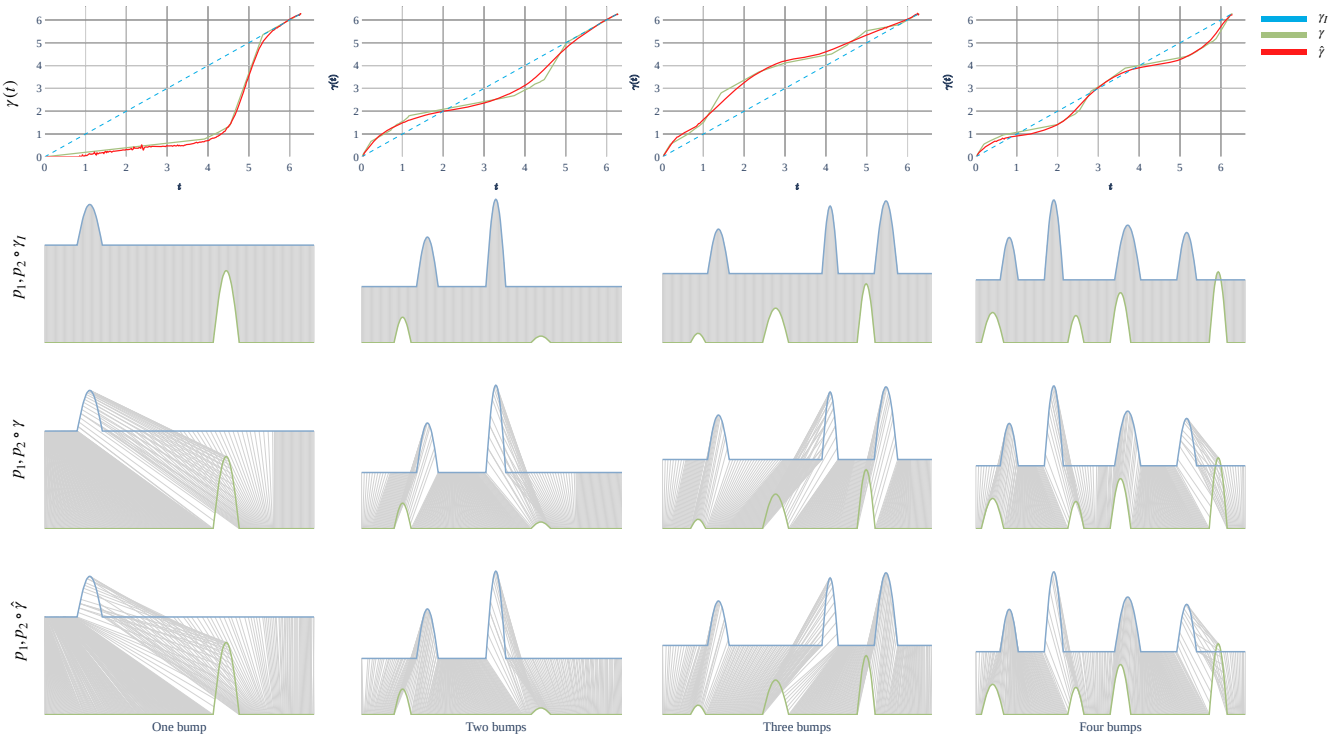


Figure 6. Performance for the one, two, three, and four-bump cases. Top: Warping functions from DP (γ) and DL ($\hat{\gamma}$). Matching by an identity warp (2^{nd} row), DP γ (3^{rd} row), and deep learning $\hat{\gamma}$ (4^{th} row).

Table 1. Time (in seconds) to obtain 100,000 warps on an i7-7700K CPU @ 4.20GHz (dynamic programming) and two NVIDIA GP102 TITAN Xp GPUs (deep learning).

	Dynamic Programming	Deep Learning
Bumps	45130.011	14.047
Shapes	5519.849	5.639

0.001, and exponential decay rates of 0.9 and 0.999 for the first and second moment estimates, respectively.

Table 1 shows comparisons of computational costs between the dynamic programming approach and the deep learning prediction approach. All experiments were performed on an Intel i7-7700K CPU @ 4.20GHz. The machine was equipped with 2 TITAN Xp GPUs for deep learning. The network was implemented and trained using TensorFlow [1]. The dynamic programming was executed on the same machine. It is observed that the deep learning warping prediction approach was approximately 3000 times faster for one-dimensional functions (bumps) and 900 times faster for two dimensional curves.

5. Discussion

We presented a deep learning approach for predicting warping functions that achieve rate-invariant alignment in the case of functions and reparameterization-invariant matching for two-dimensional curves. While we listed shape learning and novel shape representation as potential applications, in this paper, our primary motivation was demonstrating reduced computational cost. The network architecture was simple to construct and has similarities with the approaches in [10] and [12]. We experimented with three loss functions, the first of which only plenalized the cost between warping functions. The second penalized the cost between the coordinates of curves and functions, and the third penalized the cost between their SRVFs; both of these penalized the cost between warping functions as well. We showed that the latter yielded the best results. While, visually, the predicted warping, and consequently the ensuing matching, appear close to each other, we also observed cases where the predicted function failed to achieve an optimal warping. We also noted that, occasionally, the dynamic programming algorithm partially failed to achieve a good match.

In the one-dimensional case, figure 6 suggests our network is able to perform reasonably well in aligning curves when the curves are relatively close to one another. Figure 10 (a) is an example where the alignment of the curves would require significant stretching and we see that both the DP and DL solutions fail to achieve this. Because the training data is derived from the DP warps, performance of our model must be measured relative to the DP performance.

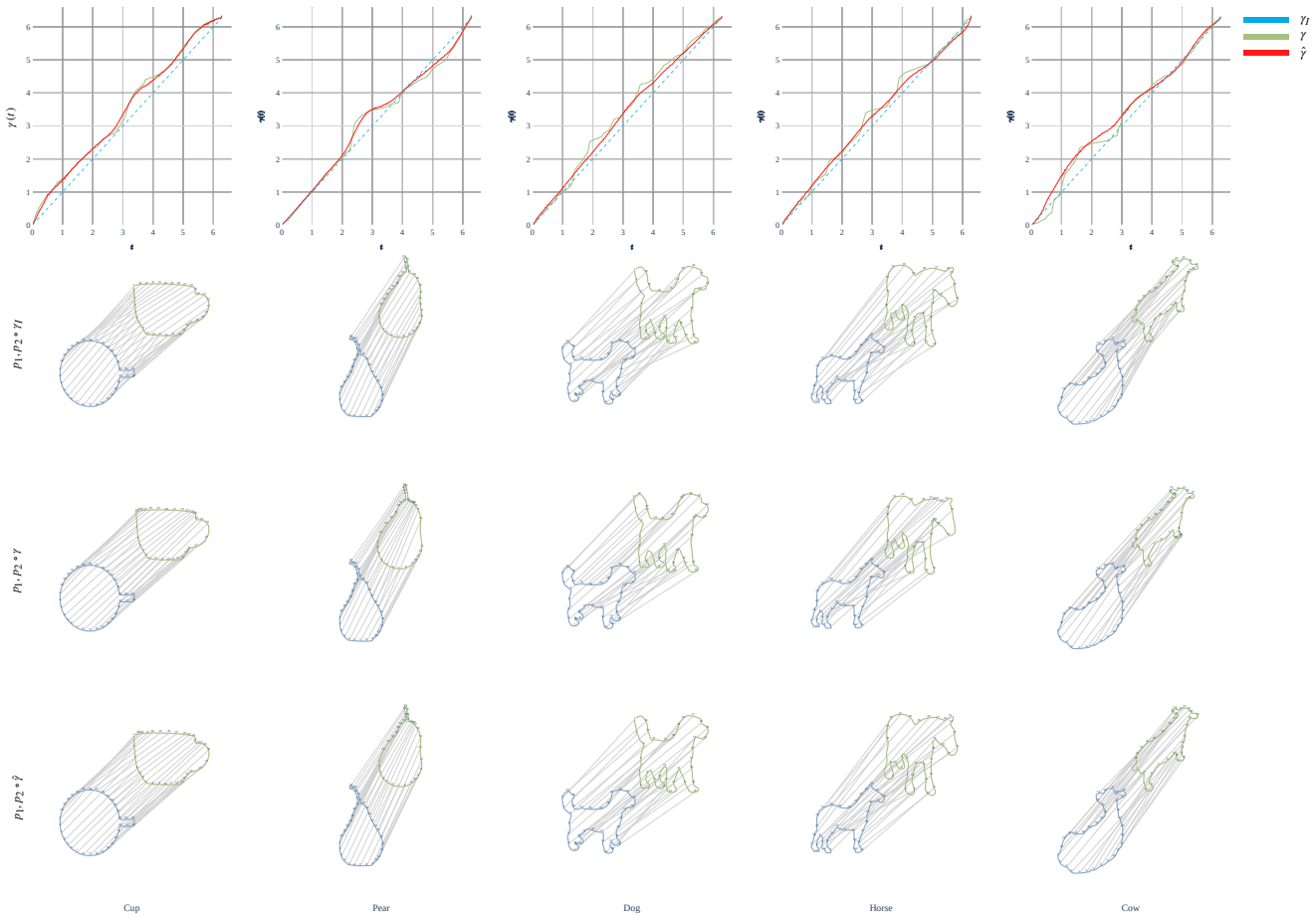


Figure 7. Performance for 2D curves. Top: Warping functions from DP (γ) and DL ($\hat{\gamma}$). Matching by an identity warp (2^{nd} row), DP γ (3^{rd} row), and deep learning $\hat{\gamma}$ (4^{th} row).

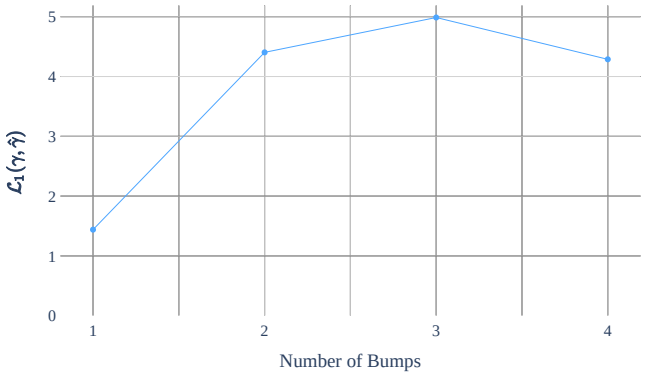


Figure 8. Average $\mathcal{L}_1$ loss on test data for all bumps.

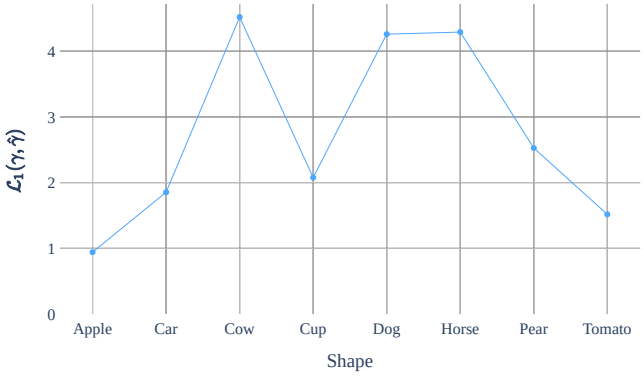
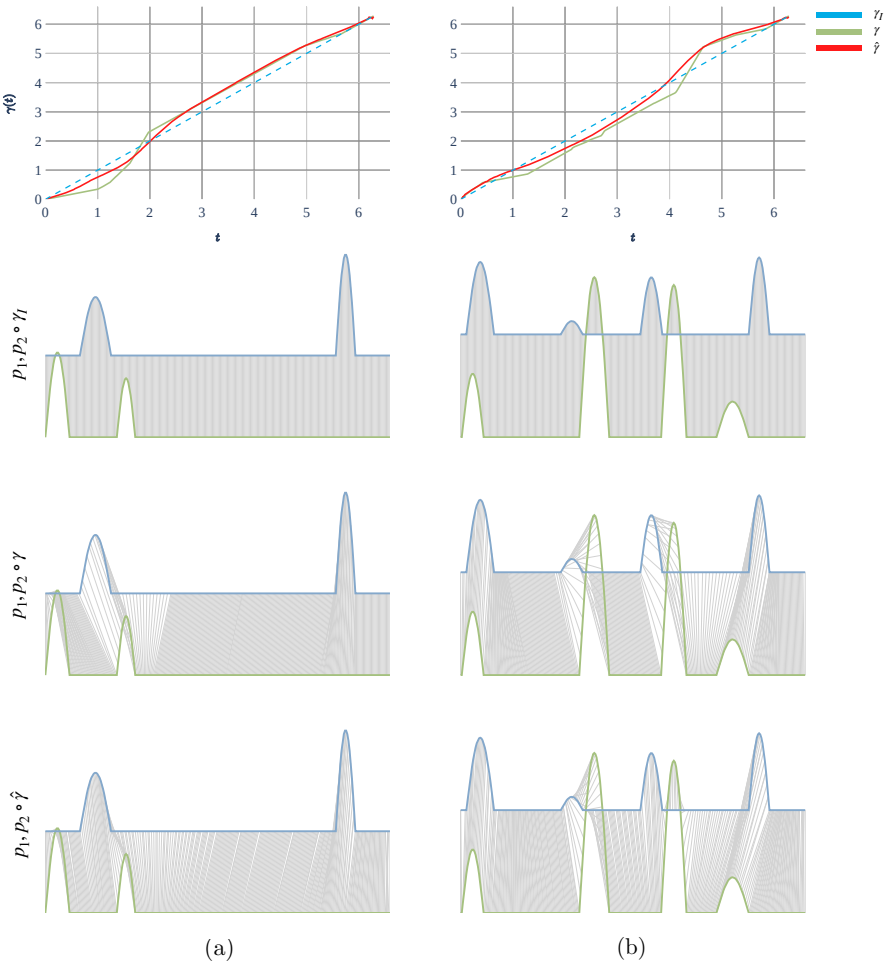


Figure 9. Average $\mathcal{L}_1$ loss on test data for shapes.

While our framework offers similar performance to DP in relatively simple curves, its architectural simplicity limits its flexibility and performance on difficult curves. For example, figure 10 (b) depicts a curve where DP succeeds but DL fails to match every bump correctly. Future work can aim to enforce positive monotonicity in the predicted warps as in [10]. However, more complicated architectures should be explored. In particular, the convolution filter size should be examined and chosen so as to span the dimension of the curve and be large enough to capture variations in the



(a) (b)

Figure 10. (a) A two-bump example where both DP and DL fail to match bumps. (b) A four-bump example where DP successfully matches all bumps but DL fails to match the third.

curve throughout its entire domain.

When trained on the two-dimensional shape data, we see that the performance is comparable to DP for simple shapes. In figure 7 we see that DL is able to perform alignments similar to DP in both the cup and pear shapes. For more complicated shapes like dogs and horses, detailed artifacts such as legs, tails, and ears can be matched reasonably well but performance is not as strong as in the DP case. This, again, may be attributed to the simplicity of the network and may possibly improve under a more complex network.

6. Acknowledgments

This research was partially supported by NIH National Institute on Alcohol Abuse and Alcoholism awards K25-AA024192 and R01-AA026834.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016. 6

[2] Shantanu H Joshi, Ryan P Cabeen, Anand A Joshi, Bo Sun, Ivo Dinov, Katherine L Narr, Arthur W Toga, and Roger P Woods. Diffeomorphic sulcal shape analysis on the cortex. *IEEE Transactions on Medical Imaging*, 31(6):1195–1212, 2012. 1

[3] Shantanu H Joshi, E. Klassen, A. Srivastava, and I. Jermyn. A novel representation for Riemannian analysis of elastic curves in R^n. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7, 2007. 1, 2

[4] 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. 1, 2

[5] Shantanu H Joshi, Katherine L Narr, Owen R Philips, Keith H Nuechterlein, Robert F Asarnow, Arthur W Toga, and Roger P Woods. Statistical shape analysis of the corpus callosum in schizophrenia. *NeuroImage*, 64:547–559, 2013. 1

[6] Ieva Kazlauskaite, Carl Henrik Ek, and Neill Campbell. Gaussian process latent variable alignment learning. In *Proceedings of Machine Learning Research*, volume 89, pages 748–757, 2019. 2

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

[8] David S Lee, Joana Loureiro, Katherine L Narr, Roger P Woods, and Shantanu H Joshi. Elastic registration of single subject task based fMRI signals. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 154–162. Springer, 2018. 1

[9] Bastian Leibe and Bernt Schiele. Analyzing appearance and contour based methods for object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 409–415, 2003. 2, 4

[10] Suhas Lohit, Qiao Wang, and Pavan Turaga. Temporal transformer networks: Joint learning of invariant and discriminative time warping. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12426–12435, 2019. 2, 3, 6, 7

[11] Jeeheh Oh, Maggie Makar, Christopher Fusco, Robert McCaffrey, Krishna Rao, Erin E Ryan, Laraine Washer, Lauren R West, Vincent B Young, John Guttag, et al. A generalizable, data-driven approach to predict daily risk of clostridium difficile infection at two large academic health centers. *Infection Control & Hospital Epidemiology*, 39(4):425–433, 2018. 1

[12] 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, 6

[13] 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

[14] 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*, 33:1415–1428, 2011. 1, 2

[15] D. W. Thompson. *On Growth and Form*. Cambridge University Press, 1943. 1

[16] 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

[17] Xiao Yang, Roland Kwitt, Martin Styner, and Marc Niethammer. Quicksilver: Fast predictive image registration–a deep learning approach. *NeuroImage*, 158:378–396, 2017. 1