

Diffeomorphic Temporal Alignment Nets for Time-series Joint Alignment and Averaging

Ron Shapira Weber and Oren Freifeld

Abstract—In time-series analysis, nonlinear temporal misalignment remains a pivotal challenge that forestalls even simple averaging. Since its introduction, the Diffeomorphic Temporal Alignment Net (DTAN), which we first introduced in [1] and further developed in [2], has proven itself as an effective solution for this problem (the conference papers [1] and [2] are earlier partial versions of the current manuscript). DTAN predicts and applies diffeomorphic transformations in an input-dependent manner, thus facilitating the joint alignment (JA) and averaging of time-series ensembles in an unsupervised or a weakly-supervised manner. The inherent challenges of the weakly/unsupervised setting, particularly the risk of trivial solutions through excessive signal distortion, are mitigated using either one of two distinct strategies: 1) a regularization term for warps; 2) using the Inverse Consistency Averaging Error (ICAE). The latter is a novel, regularization-free approach which also facilitates the JA of variable-length signals. We also further extend our framework to incorporate multi-task learning (MT-DTAN), enabling simultaneous time-series alignment and classification. Additionally, we conduct a comprehensive evaluation of different backbone architectures, demonstrating their efficacy in time-series alignment tasks. Finally, we showcase the utility of our approach in enabling Principal Component Analysis (PCA) for misaligned time-series data. Extensive experiments across 128 UCR datasets validate the superiority of our approach over contemporary averaging methods, including both traditional and learning-based approaches, marking a significant advancement in the field of time-series analysis.

1 INTRODUCTION

Time-series data often exhibits a significant amount of misalignment (also known as nonlinear time warping); i.e., typically the observations are

$$(u_i)_{i=1}^N = (v_i \circ w_i)_{i=1}^N \quad (1)$$

where u_i is the i^{th} misaligned signal, v_i is the i^{th} latent aligned signal, w_i is a latent warp of the domain of v_i , and N is the number of signals. For technical reasons, the misalignment is usually viewed in terms of $T_i \triangleq w_i^{-1}$, the inverse warp of w_i , implicitly suggesting w_i is invertible. It is also typically assumed that $(T_i)_{i=1}^N$ belong to some nominal family of warps, parametrized by θ :

$$(v_i)_{i=1}^N = (u_i \circ T^{\theta_i})_{i=1}^N, \quad T_i = T^{\theta_i} \in \mathcal{T} \quad \forall i \in (1, \dots, N). \quad (2)$$

The nuisance warps, $(T^{\theta_i})_{i=1}^N$, create a fictitious variability in the range of the signals, confounding their statistical analysis.

To fix ideas, consider ECG recordings from healthy patients during rest. Suppose that the signals were partitioned correctly such that each segment corresponds to a heartbeat, and that these segments were resampled to have equal length (e.g., see Figure 1). Each resampled segment is then viewed as a distinct signal. The sample mean of these usually-misaligned signals (even when restricted to single-patient recordings) would not look like the iconic ECG sinus rhythm; rather, it would smear the correct peaks and valleys and/or contain superfluous ones. This is unfortunate as the sample mean has numerous applications in data analysis.

Moreover, even if one succeeds somehow in aligning a currently-available recording batch, upon the arrival of new data batches, the latter will also need to be aligned; i.e., one would like to generalize the inferred alignment from the original batch to the new data without having to solve a new optimization problem. This is especially the case if the new dataset is much larger than the original one; e.g., imagine a hospital solving the problem once, and then generalizing its solution, essentially at no cost, to align all the data collected in the following year. Finally, these issues become even more critical for multi-class data (e.g., healthy/sick patients),

where only in the original batch we know which signal belongs to which class; i.e., seemingly, the new data will have to be explicitly classified before its within-class alignment.

To further contextualize this concept, let us consider the application of dimensionality reduction in time-series data, using Principal Component Analysis (PCA) as an example. PCA is designed to identify the Principal Components (PCs) that capture the maximum variance in a dataset. For time series, the initial step in PCA involves centering the data by subtracting the mean sequence from each signal. However, if the sequences are misaligned, this mean might not accurately represent the true underlying structure. This, in turn, will lead to more variance at each time step, which will subsequently require more PCs to explain the data. Thus, unwarping the signals will allow for fewer PCs to be needed to effectively describe the data, as they are no longer compensating for the distortions caused by misalignment (see Figure 2 for an illustration).

A popular attempt to solve the problem relies on **pairwise alignments**. Let $u_i = (u_i(t))_{t=1}^n$ and $u_j = (u_j(t))_{t=1}^m$ be two real-valued discrete-time signals of lengths n and m , respectively. The optimal pairwise alignment of u_j towards u_i , under some dissimilarity measure D , is defined by

$$T^* = \arg \min_{T \in \mathcal{T}} D(u_i, u_j \circ T) \quad (3)$$

where \circ denotes function composition and \mathcal{T} is a family of *warps* (or warping functions); namely, every $T \in \mathcal{T}$ is a function $T : \Omega \rightarrow \mathbb{R}$ where $\Omega \subset \mathbb{R}$ is an interval containing $\{1, \dots, m\}$. For instance, Dynamic Time Warping (DTW) provides the optimal discrete warping path between the time indices of u_i and u_j via dynamic programming, where D is (usually) a Euclidean distance [3]. More generally, while u_i and u_j are defined over discrete domains (i.e., $\{1, \dots, n\}$ and $\{1, \dots, m\}$), the notation $u_j \circ T$ in Equation 3 implicitly assumes that the value of $u_j(t')$ at every $t' \in \mathbb{R}$ is determined, using interpolation techniques, from (possibly a subset of) the m given values, $(u_j(t))_{t=1}^m$.

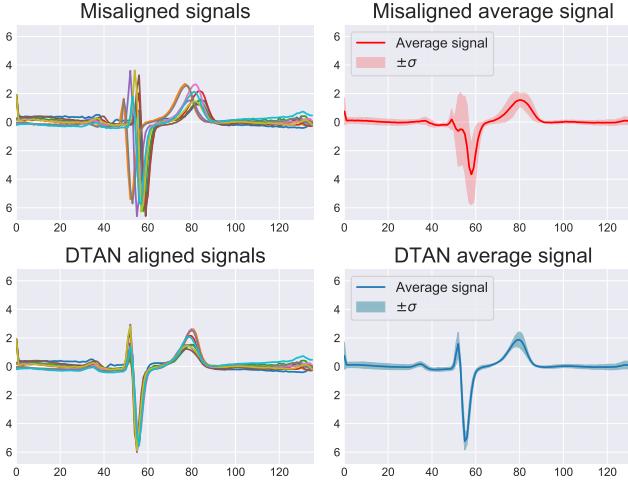


Fig. 1: An illustration of the joint-alignment problem in ECG data. The data shown is test data. Top: temporal misalignment between ECG signals and its effect on the sample mean (*ECGFiveDays* Dataset). Bottom: joint-alignment prediction by DTAN at test time.

In this paper, which extends our two conference papers [1, 2], we focus on continuously-defined warps that are order-preserving *diffeomorphisms*. A diffeomorphism (namely, a differentiable invertible function whose inverse is differentiable), is a natural choice for representing time warping [4]. Since spaces of diffeomorphisms are large, and to discourage unfavorable solutions, typically a regularization term, denoted by $T \mapsto \mathcal{R}(T; \lambda)$ and parameterized by *hyperparameters* (HP), λ , is added to the objective function; *e.g.*, \mathcal{R} might penalize lack of smoothness (in the machine-learning sense, not calculus) or large deviations from the identity map. Hence, Equation 3 is commonly replaced with

$$T^* = \arg \min_{T \in \mathcal{T}} D(u_i, u_j \circ T) + \mathcal{R}(T; \lambda) \quad (4)$$

where \mathcal{T} is a space of 1D diffeomorphisms from Ω into \mathbb{R} .

In the case of an ensemble of N signals, $(u_i)_{i=1}^N$ where $N > 2$, the pairwise approach usually does not generalize well, is prone to drift errors, and might introduce inconsistent solutions. This motivates approaches for **joint alignment** (JA), also known as global alignment or multiple-sequence alignment. The JA problem is often formulated as

$$(T_i^*)_{i=1}^N, \mu = \arg \min_{(T_i)_{i=1}^N \in \mathcal{T}, \mu} \sum_{i=1}^N D(u, u_i \circ T_i) + \mathcal{R}(T_i; \lambda) \quad (5)$$

where \mathcal{T} , $\mathcal{R}(\cdot; \lambda)$, and D are as before, T_i is the latent warp associated with u_i , and μ is a latent signal, conceptually thought of as the *average signal* (or *centroid*) of the ensemble. This optimization task may also be amortized via the training of a deep net (*e.g.*, [1, 6, 7, 2]).

We argue that this problem should be seen as a learning one, mostly due to the need for generalization. Particularly, we propose a novel deep-learning (DL) approach for the joint alignment of time-series data. More specifically, inspired by computer-vision and/or pattern-theoretic solutions for misaligned images (*e.g.*, congealing [8, 9, 10, 11, 12, 13], efficient diffeomorphisms [14, 15, 16, 17], and spatial transformer nets [18, 19, 20]), we introduce the Diffeomorphic Temporal Alignment Network (DTAN) which learns an input-dependent diffeomorphic time warping to its input signal

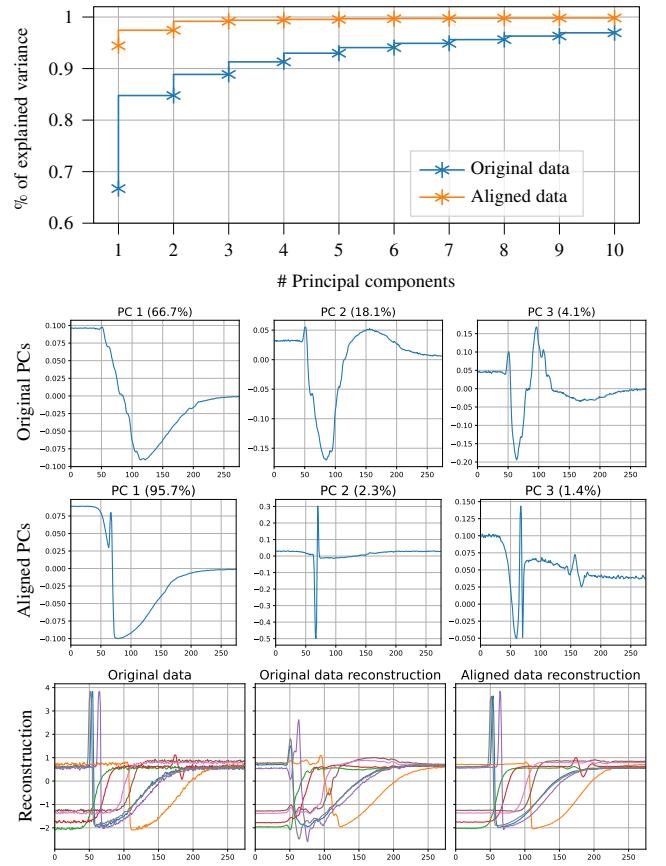


Fig. 2: The benefits of joint alignment for dimensionality reduction, evaluated on the *Trace* dataset [5] using Principal Component Analysis. The top panel shows the cumulative explained variance as a function of the number of Principal Components (PCs). The middle-top and middle-bottom panels depict the first 3 PCs of the original and DTAN-aligned data, respectively. The bottom panel illustrates the reconstruction of the original and (inverse warped) aligned data using the first 6 PCs.

to minimize a joint-alignment loss (see Figure 3 for a detailed illustration of the proposed model). The diffeomorphism family we use, called CPAB [14, 15], is based on the integration of piecewise affine velocity fields and will be further discussed in § 3.3. In the single-class case, DTAN is completely unsupervised. For multi-class problems, we propose a weakly-supervised method that results in a single model (for all classes) that learns how to perform within-class joint alignment. We demonstrate the utility of the proposed framework on real-world datasets with applications to time-series joint alignment, averaging, classification, and dimensionality reduction.

Below we list our 6 key contributions. Contributions 1-2-3 appeared in [1, 2] while contributions 4-5-6 are new.

- 1) DTAN, a DL framework for learning time series joint alignment and averaging [1].
- 2) RDTAN: a recurrent version of DTAN which predicts diffeomorphisms derived from non-stationary velocity fields [1].
- 3) A regularization-free objective function for the JA task - the *Inverse-Consistency Averaging Error* (ICAE) [2].
- 4) DTAN-MT: a multi-task version of DTAN that learns both time-series alignment and classification, resulting in better separation between the aligned classes.

METHOD	REG.-FREE	OPTIMIZATION	LEARNING	VL
EUCLIDEAN	✓	N/A	✗	✓
DBA	✓	EM	✗	✓
SOFTDTW	✗	L-BFGS	✗	✓
DTAN w/ WCSS	✗	DL TRAINING	✓	✗
DTAN w/ $\mathcal{L}_{\text{ICAE}}$	✓	DL TRAINING	✓	✓

TABLE 1: Comparing JA/averaging methods. Learning gives the ability to generalize JA to new data. VL indicates whether the method supports variable-length signals.

- 5) Evaluation of prominent time-series classification DL architectures as the backbone of the Temporal Transformer module.
 6) Analyzing DTAN’s effect on dimensionality reduction via PCA.
 We conclude this section with a timely remark: throughout the paper, the term “transformer net” should be understood in the same sense it was used in STN [18] or TTN [1] (*i.e.*, a Spatial or Temporal Transform Net), and is not to be confused with how it is used in Natural Language Processing or Vision Transformers.

2 RELATED WORK

Dynamic Time Warping (DTW) is a popular distance measure (or discrepancy) between a time-series pair [3, 22]. Given two signals of lengths n and m , DTW computes the best discrete alignment path in the $n \times m$ pairwise distance matrix. While its complexity is $O(nm)$, enforcing certain constraints on DTW results in a linear complexity. However, generalizing DTW from the pairwise case to the JA of multiple signals is prohibitively expensive since the complexity of finding the optimal discrete alignment between N signals of length n is $O(n^N)$. To overcome this limitation, several JA methods, working under the DTW geometry, were proposed. The DTW-Barycenter Averaging (DBA) [23, 24] employs an Expectation-Maximization (EM) approach to refine a signal that minimizes the sum of DTW distances from the data; *i.e.*, it alternates between finding μ (while fixing $(T_i)_{i=1}^N$),

$$\mu = \arg \min_u \sum_{i=1}^N D(u, u_i \circ T_i), \quad (6)$$

and finding discretely-defined $(T_i)_{i=1}^N$ (while fixing μ),

$$(T_i^*)_{i=1}^N = \arg \min_{(T_i)_{i=1}^N \in \mathcal{T}} \sum_{i=1}^N D(\mu, u_i \circ T_i). \quad (7)$$

SoftDTW [25], a soft-minimum variant of DTW, extends DBA. Instead of using EM, SoftDBA computes μ via gradient-based optimization. SoftDTW has one HP, γ , that controls the smoothness of the alignment ($\gamma = 0$ gives the original DTW score). SoftDTW-divergence [26] modifies SoftDTW to a proper positive-definite divergence. Both of these optimization-based methods *do not learn* how to find the JA of *new* data; *i.e.*, when new signals arrive, they must be run from scratch in order to achieve JA of the new ensemble. While it is possible to align the new data to the previously-found μ in a pairwise manner, this leads to inferior results (see § 5). Additionally, the time/memory complexity of SoftDTW is $O(mn)$. SoftDTW-div has an even worse complexity for a large n or m ; *e.g.*, results on HandOutlines (the largest UCR dataset in terms of $n \times N$) were not reported by [26], and when we tried to run SoftDTW (using tslearn [27]) on it, it failed due to memory limitations.

Other methods include the Global Alignment Kernel (GAK) [28] on which SoftDTW is based, DTW with Global Invariances which generalizes DTW/SoftDTW to both time and space [29], and Neural Time Warping that relaxes the original problem to a continuous optimization using a neural net (albeit limited in the number of signals it can jointly align) [30].

Spaces of Diffeomorphisms are often used for modeling warping paths between sequences; *e.g.*, [31, 32] proposed diffeomorphisms based on the square-root velocity function (SRVF) representation. However, the employment of diffeomorphisms in DL used to be hindered by the associated expensive computations and/or approximation/discretization schemes. For example, this is why diffeomorphisms could not initially be used effectively within a Spatial Transformer Net (STN) [18] since training the latter requires a large number of evaluations of both $x \mapsto T^\theta(x)$ and $x \mapsto \nabla_\theta T^\theta(x)$ (where θ parameterizes the chosen diffeomorphism family), and these quantities are computed at multiple values of x . This has changed, however, with the emergence of new methods [20, 33]. In particular, [20] built on the CPAB diffeomorphisms (see below) to propose the first diffeomorphic STNs.

CPAB Diffeomorphisms [14, 15]. The name CPAB, short for CPA-Based, stems from the fact that these parametric diffeomorphisms are based on Continuous Piecewise-Affine (CPA) velocity fields. Of note, in 1D, the CPAB warp, $x \mapsto T^\theta(x)$, has a closed form [14]. The expressiveness and efficiency of the CPAB warps make them an invaluable tool in DL (see, *e.g.*, [34, 20, 35, 1, 36, 37, 38, 7, 39, 40, 41, 42, 43, 44]) and thus this work uses them too. However, our method is not limited to this choice of \mathcal{T} .

A **Temporal Transformer Net (TTN)** is the 1D variant of the STN, where the latter is a DL module which, given a transformation family, predicts and applies a transformation to its input for a downstream task. Lohit et al. [45] use TTNs with discretized diffeomorphisms for learning rate-invariant discriminative warps. The SRVF framework was integrated into TTNs to either predict DTW-based warping functions [46], learn a generative model over the distribution of SRVF warps [47], and time-series JA [48]. However, computations in these nonparametric warps do not scale well with the signal length.

The **Diffeomorphic Temporal Alignment Net (DTAN)** [1] is a diffeomorphic TTN that, using the parametric and highly-expressive CPAB warps, offers an effective learning-based solution for JA and averaging. Weber et al. [1] based their DTAN implementation on libcpab [49]. Recently, Martinez et al. [7] released another CPAB library, Diffeomorphic Fast Warping (DIFW), which, while being similar to libcpab (and is, in fact, based on it), is even faster, largely due to the smart discovery of a closed-form gradient [7] for CPAB warps. Together with some other changes and an extensive HP tuning on the test data, this let them propose a DTAN implementation with SOTA results in terms of Nearest Centroid Classification (NCC) accuracy, a standard metric for time-series averaging. Henceforth will refer to the DTAN implementations from [1] and [7] as DTAN_{libcpab} and DTAN_{DIFW}, respectively. Lastly, ResNet-TW [6] also predicts CPAB warps albeit via the Large Deformation Diffeomorphic Metric Mapping framework [50].

Warp Regularization. As is typical with diffeomorphisms, CPAB warps too are usually regularized. In particular, the three works above [1, 6, 7], who all use the **within-class-sum-of-squares (WCSS) loss**, also use the following regularization from [14], $\mathcal{R}(T^{\theta_i}; \lambda) = \theta_i^\top \Sigma_{\text{CPA}}^{-1} \theta_i$. The matrix Σ_{CPA} is the covariance of a

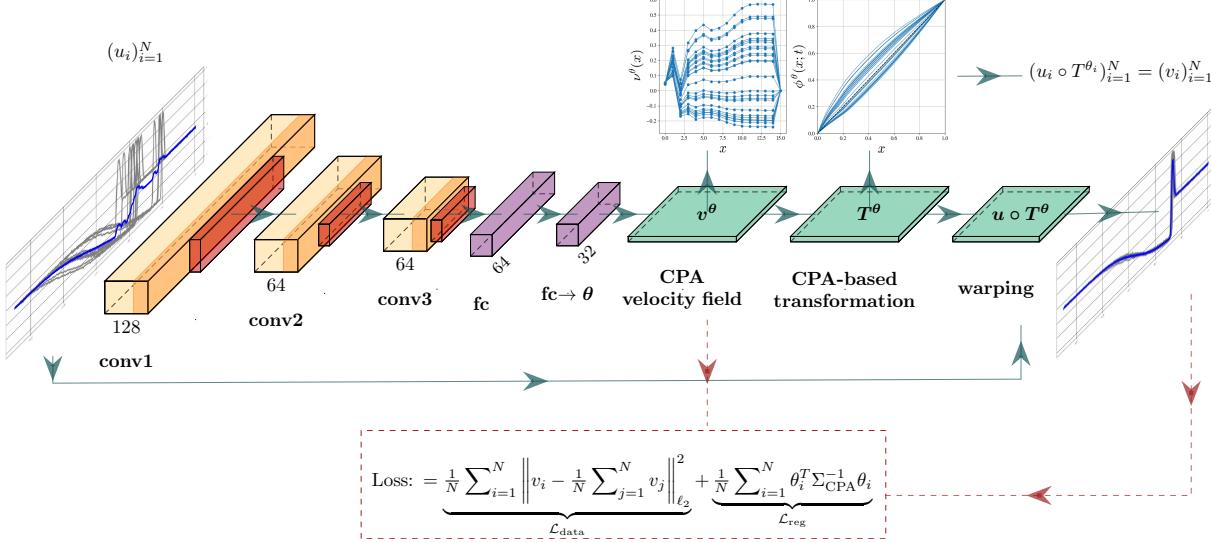


Fig. 3: DTAN joint alignment demonstrated on a class of the “Trace” dataset [21] with a simple 1D ConvNet backbone which was used in [1]. Signals are denoted in gray and their average in blue. Each Convolution layer is followed by a ReLU, Batch Normalization, and a Max-Pooling layer. The final Fully-Connected layer (fc) predicts the warping parameters, θ , of the CPA velocity fields, v^θ , which is then integrated to form a CPAB warp, T^θ . The latter, in turn, is applied to the input signal (u) to create the output, $u \circ T^\theta = v$. The loss consists of the empirical within-class variance (L_{data}) and a regularization term on θ (L_{reg}).

zero-mean Gaussian smoothness prior over CPA velocity fields and has two HPs: λ_Σ , which controls the overall variance, and λ_{smooth} , which controls the smoothness of the fields. Additionally, all these three methods predict a varying number of warps (denoted by N_{warps}), such that their composition yields the final warp.

We conclude the section with Table 1 that summarizes the differences between several JA/averaging methods and ours.

3 PRELIMINARIES

3.1 Temporal Transformer Nets

Given \mathcal{T} , a differentiable transformation family parameterized by θ , a Spatial Transformer (ST) layer performs a learnable input-dependent warp w.r.t a given objective function [18]. Reducing this from images (a 2D domain) to time series (1D), one obtains a Temporal Transformer (TT) layer. A TTN is a neural net with at least one TT layer. In more detail, let u denote the input of the TT layer. Its outputs consist of $\theta = f_{\text{loc}}(u)$ and $v = u \circ T^\theta$, where $T^\theta \in \mathcal{T}$ is a 1D warp parameterized by θ . The function $f_{\text{loc}} : u \mapsto \theta$ is itself a neural net called the localization net. Let w denote the parameters (also known as weights) of f_{loc} and let

$$\mathcal{L}((u_i, \theta_i(u_i; w))_{i=1}^N) \quad (8)$$

denote a loss function. Recall that, as usual, the back-propagation algorithm requires certain partial derivatives and note that one of these derivatives, $\nabla_\theta(T^\theta(\cdot))$, depends on the choice of \mathcal{T} .

The TTN consists of 3 modules:

- 1) **Localization network.** For an input signal, u , the localization network, f_{loc} , predicts the warp’s parameters; i.e., $f_{\text{loc}}(u) = \theta$. Any form of neural network architecture can be used for f_{loc} , as long as the output layer has d neurons, where $d = \dim(\theta)$.
- 2) **Parameterized grid generator.** This generator creates a discrete 1D grid of length M (where M is the length of the

signals): $G = (x_m)_{m=1}^M \subset [-1, 1]$ of evenly-spaced points which are later transformed by the parametrized warp, T^θ .

- 3) **Differentiable time-series resampler.** The output signal, v , is computed by interpolating the values of v at $T^\theta(G)$ from u . Let $x_{i,m}^{\text{new}} = T^{\theta_i}(x_m)$ and write the discrete-time i -th aligned signal as

$$v_i = (v_{i,m})_{m=1}^M = (v_{i,1}, \dots, v_{i,M}). \quad (9)$$

Note that due to the need to resample the signal, rather than having $v_i = u_i \circ T^{\theta_i}$, we need to also account for the resampling kernel. For the popular linear kernel, which is the one used in our work, we obtain (based on [18]),

$$v_{i,m} = \sum_{m'=1}^M u_{i,m'} \max(0, 1 - |x_{i,m}^{\text{new}} - m'|). \quad (10)$$

To propagate the loss to f_{loc} , the resampling kernel must be differentiable, which is the case for the linear kernel:

$$\frac{\partial v_{i,m}}{\partial u_{i,m'}} = \sum_{m'=1}^M \max(0, 1 - |p_{i,m}^{\text{warped}} - m'|) \quad (11)$$

$$\frac{\partial v_{i,m}}{\partial (p_{i,m}^{\text{warped}})} = \sum_{m'=1}^M u_{i,m'} \begin{cases} 0 & \text{if } |m' - p_{i,m}^{\text{warped}}| \geq 1 \\ 1 & \text{if } m' \geq p_{i,m}^{\text{warped}} \\ -1 & \text{if } m' < p_{i,m}^{\text{warped}} \end{cases}. \quad (12)$$

Here $v_{i,m}$ is the i^{th} warped signal at time point m , $u_{i,m'}$ is the input signal at time point m' and $p_{i,m}^{\text{warped}}$ is the m^{th} point of the sampling grid. The generalization of these results to multichannel time series is straightforward and thus omitted. In § 3.3 we will specify \mathcal{T} and will discuss its associated derivative, $\nabla_\theta(T^\theta(\cdot))$.

3.2 Deep-learning Time-series Architectures

The core module of the TTN is the localization network, f_{loc} , which predicts the transformation parameters θ . While in [1] we have

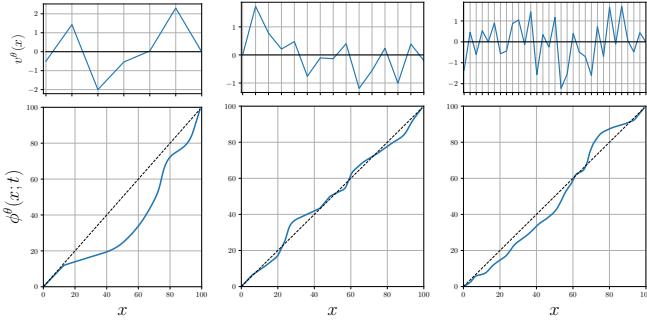


Fig. 4: CPAB warps for different partitions of $\Omega \in \{8, 16, 32\}$. Top: Continuous Piecewise-Affine (CPA) velocity fields. Bottom: The resulting CPAB warp, obtained via integration of v^θ .

used a simple Temporal Convolutional Neural Network (TCN), here we explore several other architectures.

Similar to Computer Vision and Natural Language Processing, the field of Time Series Classification (TSC) also saw a recent surge in DL-based classifiers. Traditionally, Recurrent Neural Network (RNN) [51, 52] were the go-to models for this task, as their time-dependent representation allowed them to capture temporal dependencies within signals. However, leveraging the expressive power of deeper and more recent architectures allowed TCNs to outperform RNNs while offering a more efficient training procedure. Fawaz et al., (2018) [53] provided an extensive review of such architectures for the TSC task. Their findings pointed to two architectures: Fully-Convolutional Networks (FCN) [54] and a 1D variant of the now quintessential Residual Network (ResNet) [55].

Another TCN-based architecture is InceptionTime [56]. It is a 1D variant of the Inception Network-v4 [57] and is composed of an ensemble of Inception modules. Arguing that frequency information is lost in current TSC models, a wavelet-based neural network structure called multilevel Wavelet Decomposition Network (mWDN) was proposed [58]. It preserves the advantage of multilevel discrete wavelet decomposition in frequency learning while still enabling the fine-tuning of learnable parameters via back-propagation. The model takes all or partial mWDN decomposed sub-series different frequencies as input features and updates its parameters globally for a downstream classification or a forecasting task. As different features at different frequencies are used, the authors coined the integration of mWDN features and a deep-classifier, $\psi(\cdot)$, as *Residual Classification Flow (RCF)*. Here, we utilize the mWDN framework for times-series alignment, which could be thought as *Residual Alignment Flow (RAF)*. In our experiments, $\psi(\cdot)$ is an *InceptionTime* module.

Given these recent advancements in DL for TSC, we explore the effect of the aforementioned architectures as the “backbones” of the localization networks in the TTN. In § 5 we provide an evaluation of TCN, RNN-FCN, mWDN, and, InceptionTime for time-series joint alignment and averaging under the DTAN framework.

3.3 Diffeomorphisms

As mentioned in § 1, \mathcal{T} needs to be specified. In the context of time warping, *diffeomorphisms* is a natural choice [4].

Definition 1 A (C^1) diffeomorphism is a differentiable invertible map with a differentiable inverse.

Working with diffeomorphisms usually involves expensive computations. In our case, since the proposed method explicitly incorporates

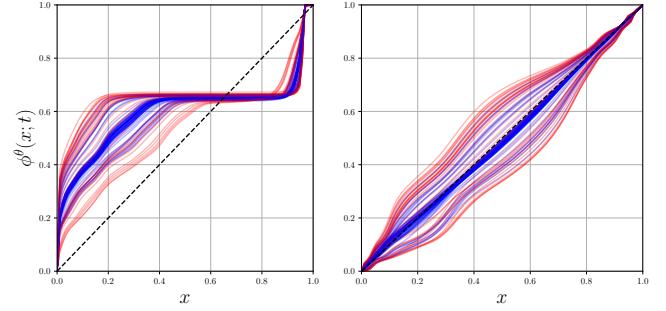


Fig. 5: The effect of the smoothness prior on the predicted warps in the *ECG200* dataset. Left: no prior. Right: $\lambda_\sigma = .01, \lambda_{smooth} = .01$. Color indicates class label.

them in a DL architecture, it is even more important (than in traditional non-DL applications of diffeomorphisms) to drastically reduce the computational difficulties. The reason is that during training, the quantities $x \mapsto T^\theta(x)$ and $x \mapsto \nabla_\theta T^\theta(x)$ are computed at multiple time points x and for multiple values of θ .

As mentioned in § 1, we have chosen to incorporate the CPAB transformation family into DTAN [1, 2]. These warps combine expressiveness and efficiency, making them a natural choice in a DL context [34, 20]. Other efficient and expressive diffeomorphisms (e.g., [17, 59, 60, 61]) can also be explored in the DTAN context, provided they also offer an efficient and highly-accurate way to evaluate $x \mapsto \nabla_\theta T^\theta(x)$ as CPAB warps do [62]. Below we briefly explain CPAB warps (restricting the discussion to 1D), and refer the reader to [14, 15, 62] for more details.

The name CPAB, short for CPA-Based, is due to the fact that these warps are based on Continuous Piecewise-Affine (CPA) velocity fields. The term “piecewise” is w.r.t. some partition, denoted by Ω , of the signal’s domain into subintervals. Let \mathcal{V} denote the linear space of CPA velocity fields w.r.t. such a fixed Ω , let $d = \dim(\mathcal{V})$, and let $v^\theta : \Omega \rightarrow \mathbb{R}$, a velocity field parametrized by $\theta \in \mathbb{R}^d$, denote the generic element of \mathcal{V} , where θ stands for the coefficient w.r.t. some basis of \mathcal{V} . The corresponding space of CPAB warps, obtained via integration of elements of \mathcal{V} , is

$$\begin{aligned} \mathcal{T} &\triangleq \left\{ T^\theta : x \mapsto \phi^\theta(x; 1) \text{ s.t. } \phi^\theta(x; t) \text{ solves} \right. \\ &\quad \left. \phi^\theta(x; t) = x + \int_0^t v^\theta(\phi^\theta(x; \tau)) d\tau \text{ where } v^\theta \in \mathcal{V} \right\}. \end{aligned} \quad (13)$$

It can be shown that these warps are indeed (C^1) diffeomorphisms [14, 15]. See Figure 4 for a typical warp. While v^θ is CPA, $T^\theta : \Omega \rightarrow \Omega$ is not (e.g., T^θ is differentiable, unlike v^θ). CPA velocity fields support an integration method that is faster and more accurate than typical velocity-field integration methods [14, 15]. The fineness of Ω controls the trade-off between expressiveness of \mathcal{T} on the one hand and the computational complexity as well as the dimensionality (i.e., the value of $d = \dim(\theta)$) on the other hand.

Initialization. Since $\theta = 0$ gives the identity map, we initialize the final layer of the localization network by sampling the weights from a zero-mean normal distribution (i.e. $w \sim \mathcal{N}(0, 10^{-5})$).

Optional zero-boundary conditions. If of interest, one can easily restrict the CPA fields to vanish at the endpoints of the domain, implying these points will be fixed points of the resulting warp, i.e. $v[0] = v[n] = 0$ (see Freifeld et al. [15] for more details).

Optional circularity constraint. Alternatively, one can enforce a circularity constraint by adding a linear constraint on the CPA velocity field, making it circularly continuous: $v[0] = v[n]$ (i.e.

enforcing the velocity at the starting point to be equal to the velocity at the end-point). See [36] for details.

The CPAB Gradient. Importantly, in 1D, the *CPAB gradient*, $\nabla_{\theta} T^{\theta}(x)$, has a closed-form expression which was recently discovered in [7]. The latter is more efficient and stable than the numerical solution and facilitates much faster training and inference time.

4 METHOD

To meet our goal, *i.e.*, solving the JA problem while being able to generalize its solution to the alignment of new data, we propose a DL-based method that includes a TTN with diffeomorphic TT layers. Particularly, here we choose \mathcal{T} to be a family of 1D CPAB warps [14, 15] and incorporate the latter within TT layers. Altogether, this lets us propose the first *Diffeomorphic Temporal Alignment Net (DTAN)* for time-series joint alignment.

4.1 Time Series Joint Alignment

Let u_i denote an input signal, let $\theta_i = \theta_i(u_i, \mathbf{w}) = f_{\text{loc}}(u_i, \mathbf{w})$ denote the corresponding output of the localization net $f_{\text{loc}}(\cdot, \mathbf{w})$ of weights \mathbf{w} , and let v_i , the warped signal, denote the result of warping u_i by $T^{\theta_i} \in \mathcal{T}$; *i.e.*, $v_i = v_i(u_i, \mathbf{w}) = u_i \circ T^{\theta_i}$. Consider first the case where all the u_i 's belong to the same class. As the variance of the observed $(u_i)_{i=1}^N$ is (at least partially) explained by the latent warps, $(T^{\theta_i})_{i=1}^N$, we seek to minimize the empirical variance of the collection of the warped signals, $(v_i)_{i=1}^N$. In other words, our data term in this setting is

$$\begin{aligned}\mathcal{L}_{\text{data}} &\triangleq \frac{1}{N} \sum_{i=1}^N \left\| v_i - \frac{1}{N} \sum_{j=1}^N v_j \right\|_{\ell_2}^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left\| u_i \circ T^{\theta_i} - \frac{1}{N} \sum_{j=1}^N u_j \circ T^{\theta_j} \right\|_{\ell_2}^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left\| u_i \circ T^{\theta_i} - \mu \right\|_{\ell_2}^2\end{aligned}\quad (14)$$

where $\|\cdot\|_{\ell_2}$ is the ℓ_2 norm and μ is the post-alignment average signal. Note this setting is unsupervised.

In the multi-class case, $\mathcal{L}_{\text{data}}$ is the sum of the within-class variances, often called the within-class sum of squares (WCSS):

$$\mathcal{L}_{\text{data}} \triangleq \sum_{k=1}^K \frac{1}{N_k} \sum_{i:y_i=k} \left\| u_i \circ T^{\theta_i} - \mu_k \right\|_{\ell_2}^2 \quad (15)$$

where K is the number of classes, y_i takes values in $\{1, \dots, K\}$ and is the class label associated with u_i (*namely*: $y_i = j$ if and only if u_i belongs to class j), and $N_k = |\{i : y_i = j\}|$ is the number of examples in class j . In this setting, the learning is partially (or weakly-)supervised: the labels, $(y_i)_{i=1}^N$ are known during the learning (but not during the test) while the within-class alignment remains unsupervised as in the single-class case. The same single network is responsible for aligning each of the classes; *i.e.*, \mathbf{w} does not vary with k .

Of importance is the fact that, unfortunately, it is possible to reduce $\mathcal{L}_{\text{data}}$ (even to zero!) by severely distorting the signals such that most of the inter-signal variability concentrates on a small region of the domain and this issue only worsens due to interpolation artifacts. We now present two complementary methods to avoid this issue:

Approach I: Regularizing the predicted warps by adding:

$$\mathcal{L}_{\text{reg}} \triangleq \sum_{i=1}^N (\theta_i^T \Sigma_{\text{CPA}}^{-1} \theta_i) \quad (16)$$

where Σ_{CPA} is the CPA covariance matrix (proposed by Freifeld *et al.* [14, 15]) associated with a zero-mean Gaussian smoothness prior over the CPA field. Akin to the standard formulation in, *e.g.*, Gaussian processes [63], Σ_{CPA} has two parameters: λ_{σ} , which controls the overall variance, and λ_{smooth} , which controls the smoothness of the field. A small λ_{σ} favors small warps (*i.e.*, close to the identity) and vice versa; similarly, the larger λ_{smooth} is, the more it favors CPA velocity fields that are almost purely affine and vice versa, as could be seen in Figure 5.

The prior also gives another way, an alternative to changing the resolution of Ω , to control the amount of expressiveness of the warps. The JA objective function, to be minimized w.r.t. \mathbf{w} , is

$$\mathcal{L}_{\text{JA}} = \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{reg}} \quad (17)$$

which corresponds to Equation 5, where $D(\cdot)$ is the euclidean distance and $\mathcal{R}(T; \lambda)$ is the smoothness prior over the CPA field with HP $\lambda = (\lambda_{\sigma}, \lambda_{\text{smooth}})$.

However, *optimal regularization is dataset-specific*. For example, penalizing deformations that are too large might not be ideal in many cases. Likewise, with a temporal smoothness prior, it is hard to determine the “right” amount of smoothness. Figure 6 illustrates the critical role of regularization on the barycenter computation using DBA, SoftDTW, and DTAN. Improper values of γ (for SoftDTW) or $\lambda_{\sigma}, \lambda_{\text{smooth}}$ (for DTAN) usually result in unrealistic warps or overly restrict the warps (*e.g.*, a *strong* prior for DTAN). This leads us to the second approach, one we have recently introduced in [2], which is regularization-free.

Approach II: Enforcing inverse-consistency between the post-alignment average sequence and the original samples. Specifically, we propose a new loss that is minimized when the average sequence is both a minimizer of the variance *and* consistent with its class. Concretely, we propose the Inverse Consistency Averaging Error loss (ICAE), defined as:

$$\mathcal{L}_{\text{ICAE}} \triangleq \sum_{k=1}^K \frac{1}{N_k} \sum_{i:y_i=k} \left\| \mu_k \circ T^{-\theta_i} - u_i \right\|. \quad (18)$$

$\mathcal{L}_{\text{ICAE}}$ measures how well the average signal, μ_k , fits each signal u_i in its class using the inverse warp $T^{-\theta_i}$. It does so by first aligning all of the signals in class k using the predicted warps, then computing their average μ_k , and finally warping μ_k back toward each u_i using $T^{-\theta_i}$, thereby ensuring consistency between them. See Figure 7 for an illustration of these two steps.

A key insight is that Equation 18 strongly discourages trivial solutions or unrealistic warps as this would result in a poor estimate of μ_k , which in turn would yield a high discrepancy between it and the original signals. In other words, the loss favors realistic deformations without the need to add a regularization term. This can be seen in Figure 8, where we show the training procedure for the *BeetleFly* dataset for both approaches: WCSS (without regularization) and the ICAE. Minimizing the WCSS distorts the signals to the point that they are no longer recognizable. In contrast, the results under the ICAE retain the key features of the data without enforcing any regularization. The full training procedure is described in Algorithm 1.

4.2 Variable-Length Joint Alignment

Our proposed $\mathcal{L}_{\text{ICAE}}$ also allows for the JA and averaging of variable-length sequences without having to use a specialized loss function or tweak the boundary conditions on T^{θ} (as mentioned in [1, 7] as a hypothetical possibility). Instead, our formulation (as

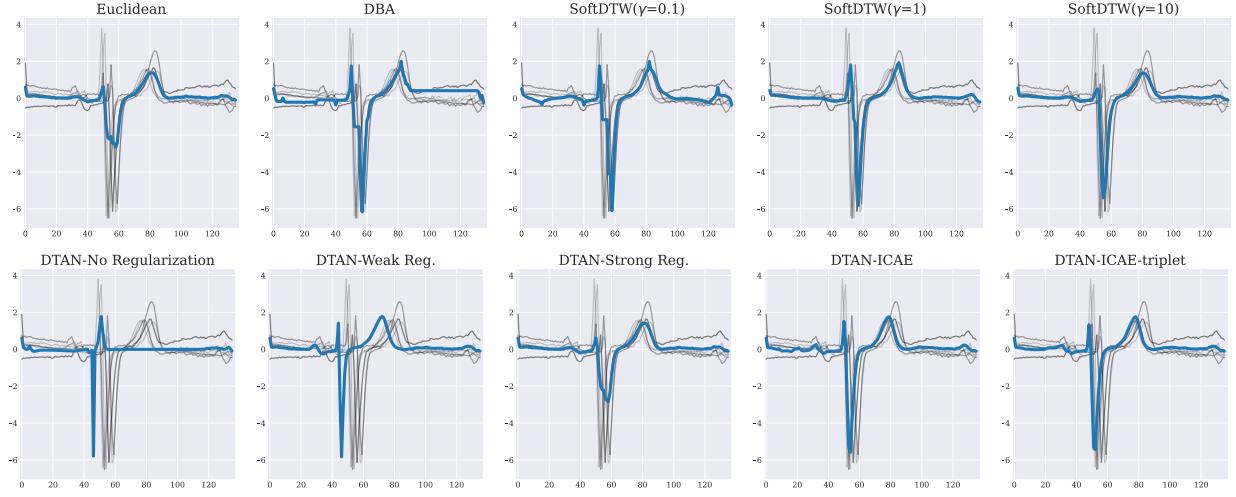
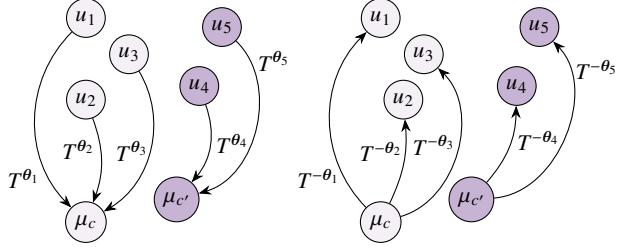


Fig. 6: The effect of the regularization HP. The figures shows 10 samples (gray) from the ECGFiveDays dataset with their estimated average (blue), and compares Euclidean averaging, DBA, SoftDTW, and several DTAN methods. DBA requires no HP but falls to poor local minima. SoftDTW’s barycenter is severely affected by the choice of its smoothing HP, γ : $\gamma = 0.1$ results in a visible ‘pinching’ effect while $\gamma = 10$ smoothens out peaks/valleys. DBA and SoftDTW are computed per class and do not learn how to generalize to new data, unlike DTAN which is learning-based and requires a single model for all classes. The regularization often used with DTAN has 2 HPs, $(\lambda_\sigma, \lambda_{\text{smooth}})$, where a *weak* regularization ($\lambda_\sigma, \lambda_{\text{smooth}} : .5, .01$) is insufficient and a *strong* regularization ($\lambda_\sigma, \lambda_{\text{smooth}} : .001, .1$), is too restrictive. $\mathcal{L}_{\text{ICAE}}$ and $\mathcal{L}_{\text{ICAE-triplet}}$ are regularization-free, yet provide barycenters that represent the data well.



(a) Centroids computed using forward warps (b) The ICAE loss computed using backward warps

Fig. 7: The Inverse Consistency Averaging Error loss in a two-class example. (a) The signals u_1, u_2 , and u_3 are in class c ; u_4 and u_5 are in class c' . Within each class, the centroid (μ_c or $\mu_{c'}$) is obtained by averaging the warped signals $((u_i \circ T^{\theta_i})_{i \in \{1,2,3\}}$ or $(u_i \circ T^{\theta_i})_{i \in \{4,5\}}$) using the forward warps. (b) The loss is computed using the backward warps; *i.e.*, we measure dissimilarity between each u_i and its class centroid, where the latter is first warped backward (“unwarped”) using $T^{-\theta_i}$ (the inverse of T^{θ_i}).

well as our code) handles both fixed and variable-length data. It does so in the following manner. First, the post-alignment average signal is produced by dividing, at each time step, the sum of the relevant values by the number of non-missing values. That is, for each time step t along the duration of the mean signal μ , we compute:

$$\mu[t] = \frac{1}{N_{\text{valid}}} \sum_{i:(u_i \circ T^{\theta_i})[t] \neq \text{null}}^N (u_i \circ T^{\theta_i})[t] \quad (19)$$

where N_{valid} is the number of signals whose domain includes a point mapped to t . Then, when μ is warped backward, Equation 18 is computed with no modifications. See, *e.g.*, Figure 9. From an implementation standpoint, we note that any null value in either

the input and/or loss would break the computational graph. To avoid for-loops and compute back-propagation in batches, it is computationally effective to first pad all samples with zeros (w.r.t. the longest signal) and create an indicator mask for missing values. The mask is also warped by T^θ in Equation 19.

4.3 Inverse Consistent Centroids Triplet Loss

While $\mathcal{L}_{\text{ICAE}}$ implies consistency, it is agnostic about the separation between different classes. That said, while metrics such as DTW are completely data-driven, DTAN is learning-based, and can be utilized to learn task-driven representations. As such, we introduce the centroid triplet loss into our framework to encourage inter-class separation. Traditionally, *e.g.* in classification tasks, a triplet loss is defined over a triplet (u_i^a, u_i^p, u_i^n) of an anchor, a positive, and a negative examples, respectively. As our task is intra-class JA and computing class averages (also known as centroids), adopting a centroid-based triplet loss is more adequate here [64]. We define the *Inverse Consistent Centroids Triplet Loss* over the triplet $(u_i^a, \mu_i^p, \mu_i^n)$ as

$$\begin{aligned} \mathcal{L}_{\text{ICAE-triplet}}(u_i^a, \mu_i^p, \mu_i^n) &\triangleq \\ \max(0, \|u_i^a - \mu_i^p \circ T^{-\theta_i}\|_{\ell_2}^2 - \|u_i^a - \mu_i^n \circ T^{-\theta_i}\|_{\ell_2}^2 + \alpha) \end{aligned} \quad (20)$$

where μ_i^p, μ_i^n are the positive and a negative class centroids, respectively, and α is the margin between them ($\alpha = 1$ in all our experiments and is dataset-independent). As both μ_i^p and μ_i^n are compared via an inverse warp, $\mathcal{L}_{\text{ICAE-triplet}}$ does not break the consistency between samples and their mean. The $\mathcal{L}_{\text{ICAE-triplet}}$ is used in tandem with $\mathcal{L}_{\text{ICAE}}$.

4.4 Recurrent DTAN

While often a coarse Ω suffices, the expressiveness of \mathcal{T} can be increased using a finer Ω at the cost of computation speed and a higher d [14, 15]. In fact, at the limit of an infinitely-fine Ω ,

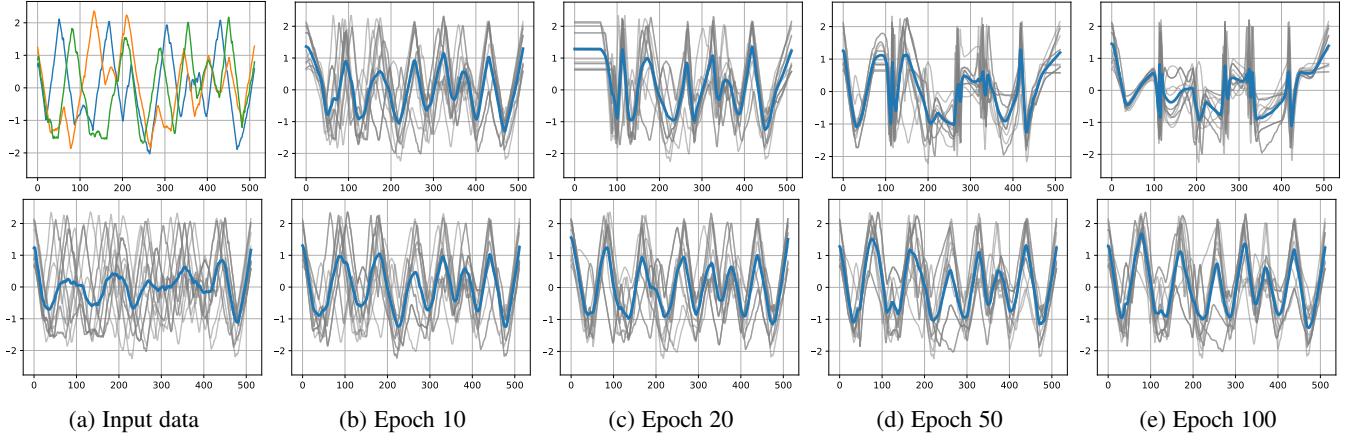


Fig. 8: Training procedure on the *BeetleFly* dataset. The first column depicts the input data (for better visualization, the top panel shows 3 random signals while the bottom 10 signals and their average are in blue). (**Top**) The Within-Class Sum of Squares (WCSS) loss reduces variance by applying an unrealistic deformation to the data, resulting in visible ‘pinching’ effect (*i.e.*, bad local minima). (**Bottom**) The proposed $\mathcal{L}_{\text{ICAЕ}}$, while requiring no regularization, avoids such an undesired solution by maintaining consistency between the average sequence and its class members.

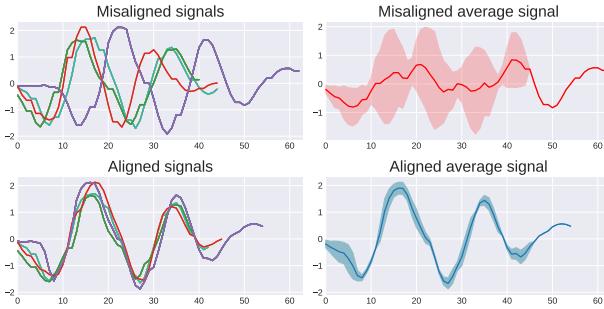


Fig. 9: JA of variable-length data (Dataset: *ShakeGestureWiimoteZ*) using the proposed $\mathcal{L}_{\text{ICAЕ}}$. Shaded area is \pm std. dev.

any diffeomorphism that is representable by integrating a Lipschitz-continuous stationary velocity field can be approximated by a CPAB diffeomorphism [14, 15]. Moreover, CPAB warps do not form a group under the composition operation [15] (even though they contain the identity warp and are closed under inversion); *i.e.*, the composition of CPAB warps is a diffeomorphism but usually not CPAB itself. Thus, a way to increase expressiveness without refining Ω is by composing CPAB warps [15]. Concatenating CPAB warps increases expressiveness beyond \mathcal{T} as it implies a non-stationary velocity field which is CPA w.r.t. Ω and piecewise constant w.r.t. time. Compositions increase dimensionality, but the overall cost of evaluating the composed warp scales better (in comparison with refinement of Ω), and it is also easier to infer the θ ’s. While this fact was not exploited in [20], we leverage it here as follows. We propose the Recurrent-DTAN (RDTAN), a net that recurrently applies nonlinear time warps, via diffeomorphic TT layers, to the input signal (Figure 13). By sharing the learned parameters by all the TT layers, an RDTAN increases expressiveness without increasing the number of parameters. While this is similar to, and inspired by, how Lin *et al.* [19] use a recurrent net with affine 2D warps, there is a key difference: since in the affine case zero-boundary conditions imply degeneracies, they explained they had to propagate warp parameters instead of the warped image as they would have liked. In contrast, as CPAB warps support optional zero-boundary conditions, propagating a warped signal

through an RDTAN is a non-issue.

4.5 Generalization via the Learned Joint-Alignment

Once the model is trained, a signal u (regardless whether it is a training or a test signal) is aligned as follows. First set $\theta = f_{\text{loc}}(u)$; *i.e.*, a forward pass of the net (an operation which is, as is usually the case in DL, simple and very fast). Next, obtain the aligned signal, v , by warping u by T^θ ; *i.e.*, set $v = u \circ T^\theta$. Especially useful and elegant is the fact that, in the multi-class case, the same single net aligns each new test signal, without knowing the label of the latter. This is in sharp contrast to other joint-alignment methods (*e.g.*, those based on DBA, SoftDTW, atlases, *etc.*) that require knowing the label of the to-be-aligned signal.

4.6 Time Series Averaging

The nuisance nonlinear misalignment distorts, among other things, the sample mean [65, 66]. As discussed in § 2, averaging under the DTW distance is a commonly-used solution to this issue [23, 24, 67, 25]; however, such non-learning DTW-based methods are computationally expensive. This is especially problematic since, as these methods do not generalize, each batch of new signals requires them to solve another optimization problem (*i.e.* consider the assignment step in the K-means algorithm). In contrast, as DTAN easily aligns new signals inexpensively and almost instantaneously via a forward pass, it also provides, in the single-class case, a mechanism for quickly averaging a new collection of previously-unseen signals. In other words, this is nothing more than computing the sample mean of the warped test data:

$$\mu = \frac{1}{N} \sum_{i=1}^N v_i = \frac{1}{N} \sum_{i=1}^N u_i \circ T^{\theta_i}. \quad (21)$$

4.7 Multi-task learning

In addition to the aforementioned objective functions, this work introduces DTAN to the notion of multitask learning. Inspired by the recent success of multitask learning in the context of time-series averaging [68], we propose to incorporate a classification objective

Algorithm 1: The JA training with an ICAE loss

Input: $N_{\text{epochs}}, f_{\text{loc}}$
Data: $(u_i, y_i)_{i=1}^N$
Output: $f_{\text{loc}}(\cdot)$, trained for joint alignment

- 1 **for** each epoch and each batch $j \in \{1, \dots, N_{\text{batches}}\}$ **do**
- 2 $\mathcal{L}_{\text{batch}} \leftarrow 0$
- 3 $(u_i, y_i)_{i=1}^{N_j} \leftarrow \text{batch}_j$
- 4 $(\theta_i)_{i=1}^{N_j} \leftarrow (f_{\text{loc}}(u_i))_{i=1}^{N_j}$
- 5 **for** $k \in \{1, \dots, K\}$ **do**
- 6 $\mu_k = \frac{1}{N_k} \sum_{i:y_i=k} (u_i \circ T^{\theta_i})$
- 7 $\mathcal{L}_{\text{ICAE}} = \frac{1}{N_K} \sum_{i:y_i=k} \|\mu_k \circ T^{-\theta_i} - u_i\|_{\ell_2}^2$
- 8 $\mathcal{L}_{\text{batch}} += \mathcal{L}_{\text{ICAE}}$
- 9 Perform an optimization step to minimize $\mathcal{L}_{\text{batch}}$

as a second task in the DTAN framework. As stated in [68], the classification objective is set to mitigate the chances of overlapping means between classes and serves as a complementary approach to $\mathcal{L}_{\text{ICAE-triplet}}$. Thus, to increase separability between classes, we propose to add a cross-entropy term to Equation 17:

$$\mathcal{L}_{\text{ce}} \triangleq - \sum_{i=1}^N y_i \log \tilde{y}_i. \quad (22)$$

where y_i are the true class labels and \tilde{y}_i are the predicted ones. In terms of architecture, we attach a fully-connected layer with a SoftMax activation to the penultimate layer (*i.e.* the embedding) of $f_{\text{loc}}(\cdot)$. Given a penultimate layer of $\text{dim} = M$, the additional classification head only adds $M \times K$ parameters to the final model. The classification framework is supervised w.r.t. the class labels, but still unsupervised w.r.t. the time-series alignment.

To control the trade-off between joint alignment and classification/separability, we introduce a hyperparameter λ_{ce} , which is set to 1 by default. Thus, the multitask loss function is defined as:

$$\mathcal{L}_{\text{multi}} \triangleq \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{reg}} + \lambda_{\text{ce}} \mathcal{L}_{\text{ce}} \quad (23)$$

In the case of RDTAN (to be discussed in § 4.4), the classification head is used only at the last recurrence of RDTAN.

4.8 Implementation

We adapted, to the 1D case, the implementation from libcpab [49] of the CPAB transformer layer, CPAB gradient, and the PyTorch C++ API. The close-formed gradient proposed in [7] is given by the DIFW package. We have implemented the CPAB regularization term, objective functions, and DTAN variants in PyTorch. Our code is available at <https://github.com/BGU-CS-VIL/RF-DTAN>.

5 EXPERIMENTS AND RESULTS

The evaluation of our approach was conducted on both synthetic and real-world data, using the popular *UCR time-series classification archive* benchmark which contains 128 datasets. Figure 12 depicts JA results on some of the UCR datasets. The rest of the section is structured as follows. First, we evaluate the effect of predicting non-stationary velocity fields via RDTAN in § 5.1. In § 5.2 we compare DTAN to state-of-the-art time-series JA and averaging methods on the UCR archive in a series of experiments. In § 5.3 we show the computational benefits of using DTAN compared with DTW-based approaches. In § 5.4 we provide new details regarding the effect of $f_{\text{loc}}(\cdot)$ and multi-task learning on JA. Finally, § 5.5 details how DTAN improves PCA.

5.1 Recurrent DTANs

Figure 13 displays the JA of synthetic data using RDTAN. We generated the synthetic data by perturbing four synthetic signals with random warps obtained via cumulative distribution functions sampled from a Dirichlet-distribution prior, as detailed in [1]. The top row presents the original latent sequences in red, the second row the perturbed synthetic signals in gray and their average in blue, and each of the subsequent three rows illustrates the alignment achieved in successive iterations of RDTAN. All four classes (columns) are aligned using the same single model. Consistent with our earlier discussion, the latent average and the latent warps are unknown during both training and testing, yet DTAN successfully recovers the latent signals via the successful JA.

RDTAN’s performance was assessed using the same recurrence number as in training. In certain cases, RDTAN benefits from applying extra warps beyond the training count. This improvement is feasible because the learned parameters are shared across all warps, akin to standard RNNs. Figure 14 demonstrates the impact of increasing the number of applied warps (up to 16) on the CBF dataset (which is a part of the UCR archive [5]), particularly in terms of NCC accuracy (defined below). The findings reveal that (a) DTAN, without recurrences, struggles with additional ones as it lacks relevant training, and (b) RDTAN either maintains robustness (RDTAN2) or shows enhanced performance (RDTAN3/4) with an increased number of recurrences.

5.2 Nearest Centroid Classification (NCC)

The most updated version [5] of the UCR archive has 128 datasets with inter-dataset variability in the number of samples, signal length, application domain, and the number of classes. Eleven of those datasets also present intra-dataset variability of the signal length; such datasets are referred to as variable-length (VL) datasets. In all of the experiments, we used the train/test splits provided by the archive. To quantify performances we used, as is customary, the NCC accuracy. This performance index is viewed as an evaluation metric for measuring how well each centroid describes its class members (and thus, implicitly, also measures the JA quality). The NCC framework has 2 steps: 1) compute the centroid, μ_k , for each class k of the *train* set; 2) label each *test* sample by the class of its closest centroid. As we explain below, Table 2, which summarizes the NCC results, is divided into several parts. The full results, together with many illustrative figures, train/test comparison, and additional evaluations, appear in our Supplemental Material (**SupMat**).

In all of our DTAN experiments, training was done via the Adam optimizer [69] for 1500 epochs, batch size of 64, N_p (the number of subintervals in the partition of Ω) was 16, and the scaling-and-squaring parameter (used by DIFW) was 8. These values were previously reported to yield the highest number of *Wins* in [7]. While in [1] we have used RDTAN, in Martinez et al. [7] the authors stacked TCNs sequentially. In this study, we fixed the number of recurrences to 4 as we did not find it necessary to stack InceptionTime models. The PyTorch TSAI implementation of the InceptionTime was taken from [70]. For DTW, DBA, and SoftDTW we used the tslearn package [27].

5.2.1 Part 1: 84 datasets – allowing an extensive HP search (previously-reported results).

An older version [21] of the UCR archive had only 85 datasets (a subset of the 128 mentioned above). Several previous works

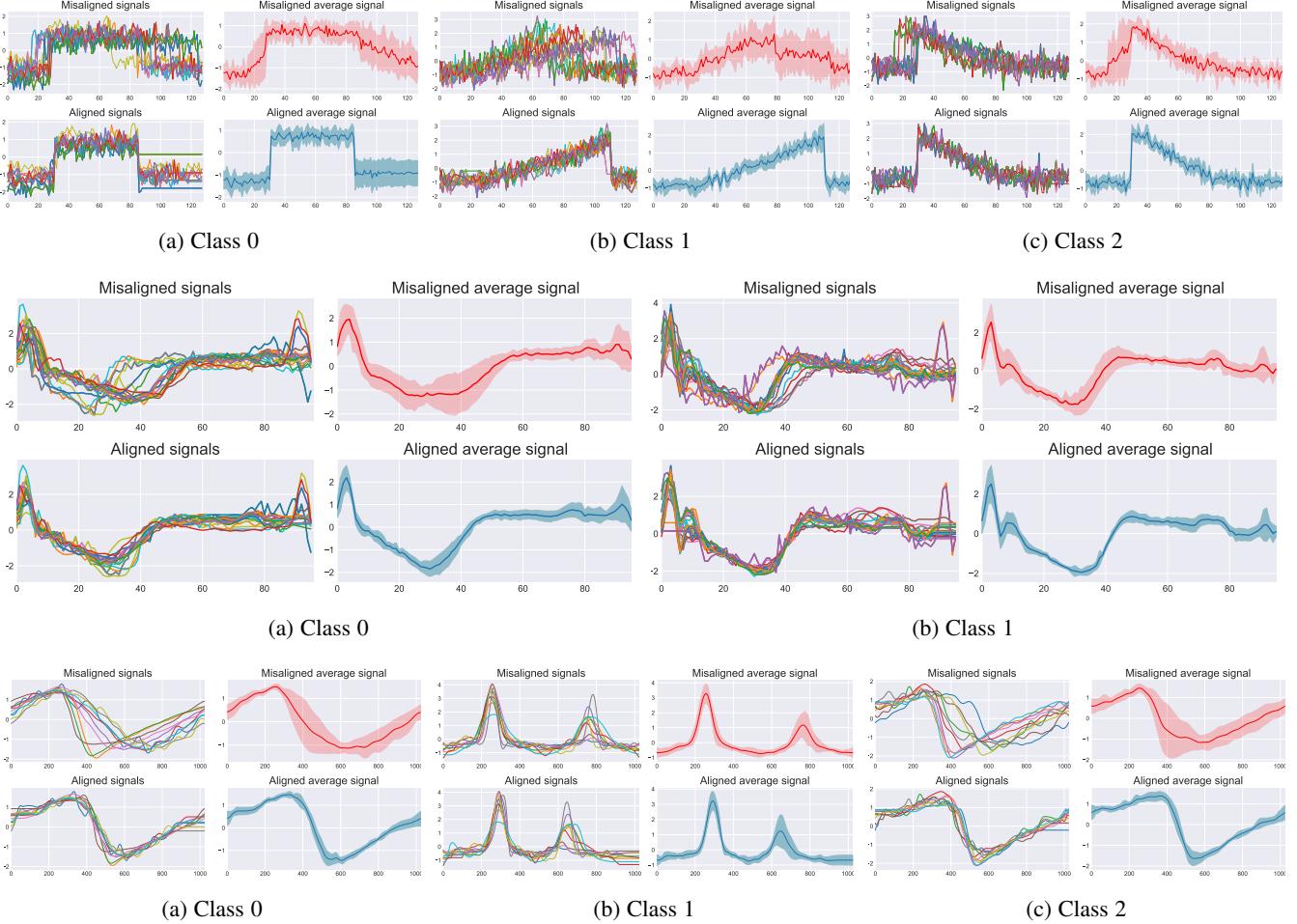


Fig. 12: Joint alignment and averaging of the (top) *CBF*, (middle) *ECG200*, and (bottom) *StarLightCurves* datasets using $\mathcal{L}_{\text{ICAЕ}}$. The shaded area corresponds to $\pm\sigma$.

reported results on only 84 datasets out of those 85, possibly due to the size of the largest dataset. Part 1 of Table 2 contains the results, on those 84 datasets, obtained by several key methods, as reported by their authors, as well as those obtained by a simple Euclidean averaging (*i.e.*, a no-alignment baseline). The methods are DBA, SoftDTW, DTAN_{libcpab}, ResNet-TW, and DTAN_{DIFW}. The regularization-free DBA requires no HP configurations. The SoftDTW methods have one HP for controlling the smoothness. Their results, reported in [26], were obtained by those authors using cross-validation. The other works [1, 6, 7] reported only their best results across different configurations. In [1] we have evaluated DTAN_{libcpab} using 12 different configurations per dataset (4 configurations for $(\lambda_\sigma, \lambda_{\text{smooth}})$ and 3 different numbers of recurrences). In [6], ResNet-TW used the same regularization configurations as in [1], but also tested varying numbers of ResNet blocks (4 to 8) per dataset. Martinez et al. [7] evaluated DTAN_{DIFW} using 96 different configurations (various options of $\lambda_\sigma, \lambda_{\text{smooth}}, N_p$, #stacked TCNs, boundary conditions, and the scaling-and-squaring parameter) per dataset. We note that: 1) tuning N_p and the boundary conditions is another form of tweaking the regularization; 2) as stated in (the supplemental material of) [7], their reported results were chosen among those 96 configurations, per dataset, based on the best performance on the test set.

5.2.2 Part 2: Regularization vs regularization-free DTAN

Part 1 of Table 2 suggests that increasing the number of tried HP configurations translates to better performance due to the large variability across the UCR datasets. However, the compact summary in Part 1 of Table 2 also hides an inconvenient truth: there is no *one-size-fits-all* configuration. For example, Martinez et al. [7] produced the best performance but this is largely due to an expensive search over a large number of HP configurations. In fact, inspecting the full results of either DTAN_{libcpab}, ResNet-TW, or DTAN_{DIFW}, reveals that the optimal choice of HP varies across the datasets and affects results drastically.

To demonstrate this crucial point, we ran a new set of experiments. We picked the HP configuration that according to [7] achieved the highest number of wins among their 96 configurations. Next, using that configuration we ran, on those 84 datasets, exactly the same DTAN but with 3 different losses: 1) WCSS plus the smoothness regularization (λ_σ and λ_{smooth} , 0.001 and 0.1, respectively); 2) our proposed $\mathcal{L}_{\text{ICAЕ}}$; 3) our proposed $\mathcal{L}_{\text{ICAЕ-triplet}}$. In the last 2 cases, which are regularization-free, the values of λ_σ and λ_{smooth} from that configuration were ignored. In all 3 cases, we used DTAN_{DIFW} with the same InceptionTime backbone [70] (in all 3 cases this gave better results than using a TCN). To account for random initializations and the stochastic nature of DL training, in each of the 3 cases we performed 5 runs on each dataset and report both the median and best results; see part 2 in Table 2. The

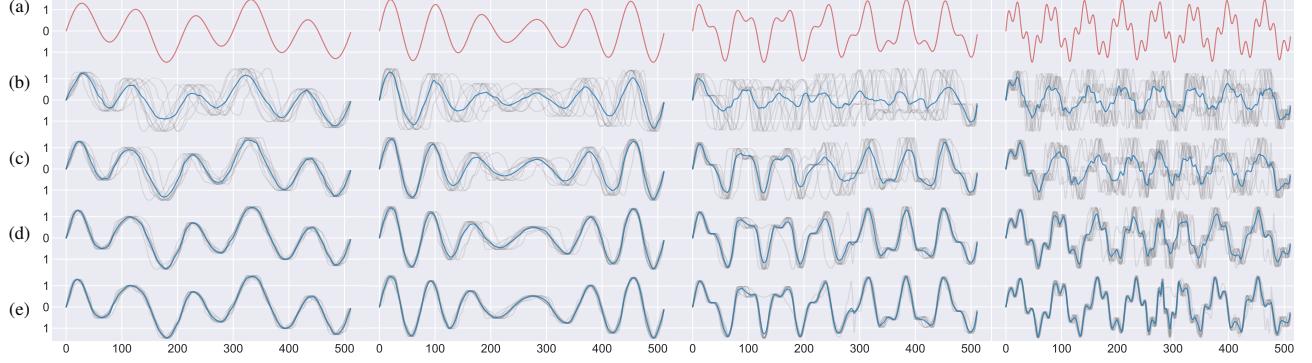


Fig. 13: Recurrent DTAN (RDTAN) JA of synthetic data. (a) latent source and (b) 10 perturbed signals (gray) and their average (blue). (c)-(e) RDTAN output at each recurrence, where the latent source signals are gradually recovered by finding the perturbed signals JA.

METHOD	OBJECTIVE	NCC _{median}	NCC _{best}	#CONFIGS	#DATASETS	#EXPERIMENTS
PART 1: ALLOWING HP SEARCH (PREVIOUSLY-REPORTED RESULTS)						
EUCLIDEAN	N/A	-	0.611	1	84	84
DBA [23]	DTW	-	0.657	1	84	84
SOFTDBA [25]	SOFTDTW	-	0.703	9	84	756
SOFTDBA [26]	SOFTDTW-DIV	-	0.708	9	84	756
DTAN _{labeled} [1]	WCSS + REG	-	0.705	12	84	1008
RESNet-TW [6]	WCSS + REG	-	0.711	20	84	1680
DTAN _{DIFW} [7]	WCSS + REG	-	0.749	96	84	8064
PART 2: SINGLE HP CONFIGURATION IN ALL DATASETS (SAME UCR DATASETS AS REPORTED BY OTHER WORKS ABOVE)						
DTAN _{DIFW} [1, 7]	WCSS + REG	0.604	0.607	1	84	84
DTAN _{DIFW} [2]	$\mathcal{L}_{\text{ICAE}}$	0.665	0.694	1	84	84
DTAN _{DIFW} [2]	$\mathcal{L}_{\text{ICAE}}\text{-triplet}$	0.707	0.739	1	84	84
PART 3: SINGLE HP CONFIGURATION IN ALL DATASETS (INCLUDING ADDITIONAL NEWER FIXED-LENGTH UCR DATASETS)						
DTAN _{DIFW} [1, 7]	WCSS	0.609	0.65	1	117	117
DTAN _{DIFW} [1, 7]	WCSS + REG	0.603	0.605	1	117	117
DTAN _{DIFW} [2]	$\mathcal{L}_{\text{ICAE}}$	0.656	0.686	1	117	117
DTAN _{DIFW} [2]	$\mathcal{L}_{\text{ICAE}}\text{-triplet}$	0.709	0.741	1	117	117
PART 4: SINGLE HP CONFIGURATION IN ALL DATASETS (FULL UPDATED UCR ARCHIVE, INCLUDING VARIABLE-LENGTH DATASETS)						
DTAN _{DIFW} [2]	$\mathcal{L}_{\text{ICAE}}$	0.623	0.653	1	128	128
DTAN _{DIFW} [2]	$\mathcal{L}_{\text{ICAE}}\text{-triplet}$	0.67	0.701	1	128	128

TABLE 2: Nearest Centroid Classification Accuracy.

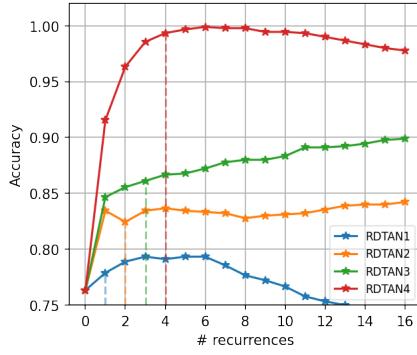


Fig. 14: Nearest Centroid Classifier (NCC) performance as a function of the number of recurrent wraps applied by RDTAN during inference, evaluated on the CBF dataset. Dashed vertical lines indicate the number of recurrences used during training.

results illustrate the merits of our regularization-free approach: a single HP configuration for the regularization, even the one stated

as the best, does not properly fit the entirety of the UCR datasets. In contrast, dropping the regularization term and using our $\mathcal{L}_{\text{ICAE}}$ increases performance by a large margin, which is only further increased when utilizing $\mathcal{L}_{\text{ICAE}}\text{-triplet}$, which increases separability between class centroids (a feat current DTW-based methods are incapable of) and achieves SOTA results.

5.2.3 Part 3 & 4: Using a single HP configuration in all of the 128 datasets.

To produce the results in part 3 of Table 2, we again repeated the procedure from part 2, except that 1) we added another case where the loss is only WCSS with no regularization, and 2) the results, on 117 datasets, also take into account additional fixed-length datasets that were added in the newer UCR archive. The results in, and conclusions from, Part 3 are consistent with Part 2. WCSS did slightly better than WCSS+Reg, probably since even though it distorts the signals, it makes it a bit easier (than in the WCSS+Reg case) to differentiate between classes. In any case, our losses outperform both of these methods. Part 4 extends the results

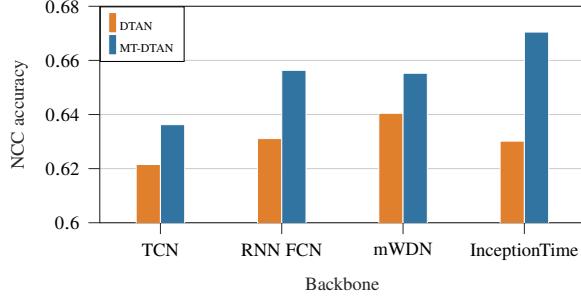


Fig. 15: DTAN vs. MT-DTAN comparison across different backbones on 113 datasets of the UCR archive [5] (2019 version) evaluated by the Nearest Centroid Classification (NCC) accuracy.

of Part 3 by adding, for the DTANs with our proposed losses, the 11 VL datasets (for a total of 128).

5.3 Computation-time Comparison

A key advantage of learning-based approaches is fast inference on new data. We performed several timing comparisons between DBA, SoftDTW (whose HP, $\gamma \in \{0.01, 0.1, 1\}$, must be searched in each dataset), and DTAN, trained with the proposed $\mathcal{L}_{\text{ICAE}}$. We used a machine with 12 CPU cores, 32Gb RAM, and an RTX 3090 GPU card. We chose a subset of the UCR archive, spanning different lengths and sample sizes, and compared the time it took to compute the centroids on the entire train set. Since DBA and SoftDTW are optimization-based we provide timing for two approaches: (1) barycenter computation time of a new batch ($N = 30$, average of 5 runs) and (2) computing DTW/SoftDTW between the batch and its barycenter (which, after warping, can be averaged again). For DTAN, this is just the inference time. Table 3 presents the result. On training data, for small datasets (in terms of n, N), SoftDTW/DBA is faster than DTAN, but this trend is reversed for the large ones. **SoftDTW and DBA run out of memory** on the largest dataset (HandOutlines). During inference, using DTAN is *orders of magnitude faster* ($\times 10$ – $\times 10^4$) than recomputing barycenters, and, on the larger datasets, is $\times 10$ faster than computing DTW/SoftDTW.

5.4 Multi-task Learning and Backbone Comparison

In this section, evaluation was performed on 113 (out of 128) datasets of the updated UCR archive [5] (*i.e.*, using all datasets, omitting the ones containing VL and/or too short for the max-pooling operators in some of the architectures). Again, we used the provided train/test splits given by the authors of the archive and used 20% of the train set as validation for choosing the best epoch. The experiments in the previous sections provided an in-depth evaluation of DTAN, given a fixed f_{loc} , across different numbers of recurrences, HP values, and objective functions. In this section, we fix $\lambda_{\sigma} = 0.1$, $\lambda_{\text{smooth}} = 0.5$, and focus on how the choice of f_{loc} affects DTAN’s performance and the effect of multitask learning. To this end, we chose the following architectures:

- 1) **TCN**: Temporal Convolutional Network, identical to the CNN from previous sections, but with an adaptive average pooling operator before the penultimate layer to maintain a fixed number of parameters w.r.t. the input’s length.
- 2) **RNN-FCN**: A Recurrent Neural Network (RNN) with a hidden layer of size=100 followed by a Fully-Convolutional

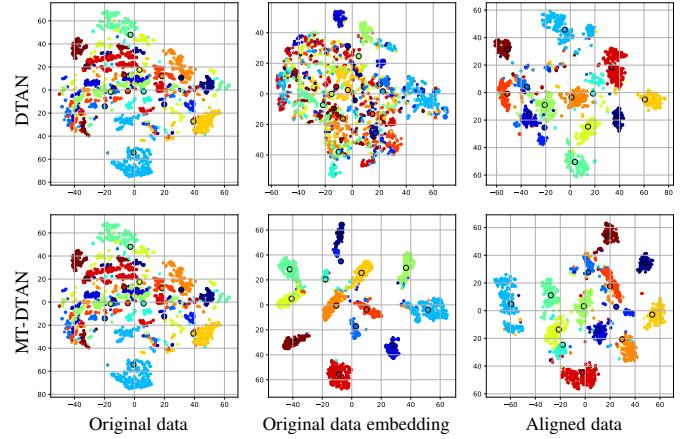


Fig. 16: t-SNE visualizations of the 14-class *FacesUCR* dataset are shown for DTAN (top) and MT-DTAN (bottom) using the *InceptionTime* backbone. DTAN effectively reduces the within-class variance in the original signal’s domain but fails to achieve similar results for latent features, also known as the embedding. Conversely, the multi-tasking framework, MT-DTAN, demonstrates improved separation both in the original domain and in the embedding space.

layer (FCN) with 3 blocks of [128, 256, 128] and a kernel size of [7, 5, 3] respectively.

- 3) **InceptionTime** [56]: composed of 5 Inception blocks (identical to the one used in previous sections).
- 4) **mWDN** [58]: composed of 3 Wavelet Blocks and an *InceptionTime* module for the *Residual Alignment Flow* (RAC) framework.

We have used the *tsai* PyTorch implementation for RNN-FCN, mWDN, and InceptionTime [70].

Results: Figure 15 shows the average NCC test accuracy of DTAN and MT-DTAN for the different architectures on 113 datasets of the UCR archive [5]. The overall best performance is achieved by MT-DTAN coupled with the *InceptionTime* architecture. This is consistent with the results presented in [56], where the authors show that *InceptionTime* produced the best performance for TSC. It is therefore unsurprising that it presented the largest performance gain when trained in the multi-task framework, as it was specifically designed for classification. We also note that the *mWDN* architecture provides the best performance for DTAN. Overall, the results indicate the importance of the choice of f_{loc} when it comes to time-series JA and deep TSC architectures are a good choice for this task.

Additionally, Figure 16 presents the t-SNE visualization of the *FacesUCR* dataset original data, learned embeddings, and the aligned data (using the *InceptionTime* architecture). Both DTAN and MT-DTAN alignment are sufficient for the t-SNE algorithm to provide adequate clustering in a 2-dimensional projection. The same cannot be said for the embeddings of the *InceptionTime* model, as DTAN is unable to provide good separation in the latent space. In comparison, MT-DTAN can provide a good separation between classes for both the aligned data and its embeddings. This helps to shed light on the performance gains achieved by MT-DTAN compared with the original model.

5.5 Principal Components Analysis

As discussed in § 1, time-series data pose particular issues when it comes to dimensionality reduction by, *e.g.*, PCA. While the

Dataset	$N_{samples}$	N_{class}	Length	DBA	$\text{SoftDTW}_{\gamma=0.01}$	$\text{SoftDTW}_{\gamma=0.1}$	$\text{SoftDTW}_{\gamma=1}$	$\text{DTAN}_{\text{ICAE}}$
Training time - full train set (sec)								
ECGFiveDays	23	2	136	0.90	0.65	0.64	0.31	157.39
Yoga	300	2	426	52.4104	265.493	283.566	50.3923	565.65
StarLightCurves	300	3	1024	1140.79	3399.90	964.21	441.33	2657.20
HandOutlines	1000	2	2709	N/A	N/A	N/A	N/A	6483.50
Inference time, averaged over 5 runs (sec)								
ECGFiveDays	30	1	136	0.3 ± 0.03	3.39 ± 1.32	2.22 ± 0.36	0.73 ± 0.25	0.02 ± 0.013
Yoga	30	1	426	4.21 ± 0.9	31.46 ± 5.92	27.58 ± 4.33	4.73 ± 0.38	0.02 ± 0.01
StarLightCurves	30	1	1024	19.08 ± 3.06	80.52 ± 12.71	61.5 ± 22.07	15.2 ± 0.15	0.02 ± 0.01
HandOutlines	30	1	2709	70.69 ± 28.39	209.2 ± 58.93	155.53 ± 18.37	68.54 ± 0.3	0.04 ± 0.02
Distance to barycenter using the corresponding metric, averaged over 5 runs (sec)								
ECGFiveDays	30	1	136	0.05 ± 0.0	0.04 ± 0.0	0.04 ± 0.0	0.05 ± 0.0	0.024 ± 0.0
Yoga	30	1	426	0.09 ± 0.0	0.25 ± 0.0	0.28 ± 0.0	0.3 ± 0.0	0.024 ± 0.0
StarLightCurves	30	1	1024	0.11 ± 0.01	1.39 ± 0.01	1.61 ± 0.0	1.76 ± 0.0	0.023 ± 0.0
HandOutlines	30	1	2709	0.4 ± 0.01	10.03 ± 0.01	11.5 ± 0.03	12.54 ± 0.07	0.045 ± 0.002

TABLE 3: Timing comparison. (Top) During the fitting/training step, SoftDTW/DBA are computed per class while DTAN_{ICAE} uses one model for all classes. (Middle) During inference, 30 new samples are averaged. Soft/DBA needs to be called again as it is optimization-based, while DTAN_{ICAE} requires a single forward pass. (Bottom) Finally, each new sample is compared to its train-set barycenter using the corresponding metric. **N/A = Out Of Memory** (on a machine with 12 CPU cores and 32Gb RAM).

relation between time-series data and PCA has been researched in the context of functional data analysis (*e.g.* functional-PCA [71, 72]) or neural computation [73], we focus on the effect of JA on the traditional PCA algorithm. In particular, after JA has been learned, PCA can be applied to the aligned time series to produce misalignment-robust principal components (PCs). Given a set of observations and the predicted warping parameters by DTAN, $(u_i, \theta_i)_{i=1}^N$ respectively, we apply PCA on the warped data. Given the Singular Value Decomposition (SVD) of $(v_i)_{i=1}^N$,

$$(v_i)_{i=1}^N = P \Lambda Q^T = \sum_i^N \sqrt{s_i} p_{i,j} q_i, \quad (24)$$

one can perform data reconstruction in its original domain, given the first k PCs:

$$\tilde{v}_j = \sum_i^k \sqrt{s_i} p_{i,j} q_i \quad (25)$$

$$\tilde{u}_j = \tilde{v}_j \circ T^{-\theta_j} \quad (26)$$

To evaluate the effect of DTAN JA on dimensionality reduction we provide results on the Trace dataset as a case study. Figure 2 (top) shows the cumulative explained variance by the first 10 PCs on both the original and aligned data. Since DTAN is set to minimize the within-class variance by reducing temporal variability, fewer PCs are required to explain the overall variance of the entire set w.r.t. the original data. This is also reflected in Figure 2 middle-to-bottom panels, where 1) the first three PCs are presented and 2) the reconstruction, performed by projecting the aligned data onto the first 6 PCs. **The first PC of the aligned data already explains 95.7% of the variance while the first three PCs of the original data combined explain only 88.9%.** The bottom panels demonstrate that, in contrast to the misaligned original data, 6 PCs adequately reconstruct the aligned data with high fidelity and also serve as a denoising procedure, suggesting that PCA and similar dimensionality reduction methods could be enhanced by DTAN.

6 CONCLUSION

In this work, we introduced DTAN, a novel deep learning framework for time-series Joint Alignment (JA), drawing upon a blend

of contemporary and traditional concepts such as Spatial Transformer Networks (STN; [18, 20]), efficient and highly-expressive diffeomorphisms [14, 15], and JA cost functions [9, 12, 74, 75]. DTAN facilitates unsupervised learning of alignments, with an extension to a weakly-supervised regime when class labels are available, enabling class-specific JA. The inherent challenges of unsupervised JA, particularly the risk of trivial solutions through excessive signal distortion, are mitigated through two distinct strategies: a regularization term for warps and the introduction of the Inverse Consistency Averaging Error (ICAE), a novel, regularization-free approach. Furthermore, we present RDTAN, an enhanced recurrent version of DTAN, which surpasses the original in terms of expressiveness and performance without an increase in parameter count. To augment class separation, we propose MT-DTAN, a multi-tasking extension of DTAN, optimized for simultaneous alignment and classification, alongside the Inverse Consistent Centroids Triplet Loss, derived from ICAE. Further evaluations demonstrate that architectures originally designed for time-series classification, when used as the backbone for the TTN, are equally effective in facilitating time-series alignment tasks. Finally, our findings underscore the efficacy of JA in enabling misalignment-robust and efficient Principal Component Analysis (PCA) for time-series data.

ACKNOWLEDGEMENTS

This work was supported by the Lynn and William Frankel Center at BGU CS, by the Israeli Council for Higher Education via the BGU Data Science Research Center, and by the Israel Science Foundation Personal Grant #360/21. R.S.W. was also funded in part by the BGU Kreitman School Negev Scholarship.

REFERENCES

- [1] R. S. Weber, M. Eyal, N. Skafte Detlefsen, O. Shriki, and O. Freifeld, “Diffeomorphic temporal alignment nets,” in *Advances in neural information processing systems*, vol. 32, 2019.

- [2] R. S. Weber and O. Freifeld, “Regularization-free diffeomorphic temporal alignment nets,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 30794–30826.
- [3] H. Sakoe, “Dynamic-programming approach to continuous speech recognition,” *1971 Proc. the International Congress of Acoustics, Budapest*, 1971.
- [4] D. Mumford and A. Desolneux, *Pattern theory: the stochastic analysis of real-world signals*. AK Peters/CRC Press, 2010.
- [5] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, “The ucr time series archive,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.
- [6] H. Huang, B. B. Amor, X. Lin, F. Zhu, and Y. Fang, “Residual networks as flows of velocity fields for diffeomorphic time series alignment,” *arXiv preprint arXiv:2106.11911*, 2021.
- [7] I. Martinez, E. Viles, and I. G. Olaizola, “Closed-form diffeomorphic transformations for time series alignment,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 15122–15158.
- [8] E. G. Miller, N. E. Matsakis, and P. A. Viola, “Learning from one example through shared densities on transforms,” in *CVPR*, vol. 1. IEEE, 2000, pp. 464–471.
- [9] E. G. Learned-Miller, “Data driven image models through continuous joint alignment,” *IEEE TPAMI*, vol. 28, no. 2, pp. 236–250, 2006.
- [10] G. B. Huang, V. Jain, and E. Learned-Miller, “Unsupervised joint alignment of complex images,” in *ICCV*. IEEE, 2007, pp. 1–8.
- [11] G. Huang, M. Mattar, H. Lee, and E. G. Learned-Miller, “Learning to align from scratch,” in *NIPS*, 2012, pp. 764–772.
- [12] M. Cox, S. Sridharan, S. Lucey, and J. Cohn, “Least squares congealing for unsupervised alignment of images,” in *CVPR*, vol. 2008. NIH Public Access, 2008, p. 1.
- [13] ——, “Least-squares congealing for large numbers of images,” in *ICCV*. IEEE, 2009, pp. 1949–1956.
- [14] O. Freifeld, S. Hauberg, K. Batmanghelich, and J. W. Fisher III, “Highly-expressive spaces of well-behaved transformations: Keeping it simple,” in *ICCV*, 2015.
- [15] ——, “Transformations based on continuous piecewise-affine velocity fields,” *IEEE TPAMI*, 2017.
- [16] M. Zhang and P. T. Fletcher, “Finite-dimensional Lie algebras for fast diffeomorphic image registration,” in *IPMI*, 2015.
- [17] ——, “Fast diffeomorphic image registration via fourier-approximated lie algebras,” *IJCV*, 2018.
- [18] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, “Spatial transformer networks,” in *Advances in neural information processing systems*, 2015, pp. 2017–2025.
- [19] C.-H. Lin and S. Lucey, “Inverse compositional spatial transformer networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2568–2576.
- [20] N. Skafte Detlefsen, O. Freifeld, and S. Hauberg, “Deep diffeomorphic transformer networks,” in *CVPR*, 2018.
- [21] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, “The ucr time series classification archive,” July 2015.
- [22] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.
- [23] F. Petitjean, A. Ketterlin, and P. Gançarski, “A global averaging method for dynamic time warping, with applications to clustering,” *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.
- [24] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh, “Dynamic time warping averaging of time series allows faster and more accurate classification,” in *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 2014, pp. 470–479.
- [25] M. Cuturi and M. , “Soft-dtw: a differentiable loss function for time-series,” *arXiv preprint arXiv:1703.01541*, 2017.
- [26] M. Blondel, A. Mensch, and J.-P. Vert, “Differentiable divergences between time series,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 3853–3861.
- [27] R. Tavenard, “tslearn: a machine learning toolkit dedicated to time-series data (2017),” *URL https://github.com/rtavenar/tslearn*, 2017.
- [28] M. Cuturi, “Fast global alignment kernels,” in *ICML*, 2011.
- [29] T. Vayer, L. Chapel, N. Courty, R. Flamary, Y. Soullard, and R. Tavenard, “Time series alignment with global invariances,” *arXiv preprint arXiv:2002.03848*, 2020.
- [30] K. Kawano, T. Kutsuna, and S. Koide, “Neural time warping for multiple sequence alignment,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3837–3841.
- [31] A. Srivastava, E. Klassen, S. H. Joshi, and I. H. Jermyn, “Shape analysis of elastic curves in euclidean spaces,” *IEEE TPAMI*, 2010.
- [32] A. Srivastava, W. Wu, S. Kurtek, E. Klassen, and J. S. Marron, “Registration of functional data using fisher-rao metric,” *arXiv preprint arXiv:1103.3817*, 2011.
- [33] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag, and A. V. Dalca, “An unsupervised learning model for deformable medical image registration,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9252–9260.
- [34] S. Hauberg, O. Freifeld, A. B. L. Larsen, J. W. F. III, and L. K. Hansen, “Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation,” in *AISTATS*, 2016.
- [35] N. Skafte Detlefsen and S. Hauberg, “Explicit disentanglement of appearance and perspective in generative models,” *NeurIPS*, 2019.
- [36] I. Kaufman, R. S. Weber, and O. Freifeld, “Cyclic diffeomorphic transformer nets for contour alignment,” in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 349–353.
- [37] G. Shacht, D. Danon, S. Fogel, and D. Cohen-Or, “Single pair cross-modality super resolution,” in *CVPR*, 2021.
- [38] P. Schwöbel, F. R. Warburg, M. Jørgensen, K. H. Madsen, and S. Hauberg, “Probabilistic spatial transformer networks,” in *UAI*, 2022.
- [39] N. Neifar, A. Ben-Hamadou, A. Mdhaffar, M. Jmaiel, and B. Freisleben, “Leveraging statistical shape priors in gan-based ECG synthesis,” *arXiv preprint arXiv:2211.02626*, 2022.
- [40] A. Kryeem, S. Raz, D. Eluz, D. Itah, H. Hel-Or, and I. Shimshoni, “Personalized monitoring in home healthcare: An assistive system for post hip replacement rehabilitation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 1868–1877.

- [41] H. Wang, F. Liu, Q. Zhou, R. Yi, X. Tan, and L. Ma, “Continuous piecewise-affine based motion model for image animation,” in *AAAI*, 2024.
- [42] A. Kryeem, N. Boutboul, I. Bear, S. Raz, D. Eluz, D. Itah, H. Hel-Or, and I. Shimshoni, “Action assessment in rehabilitation: Leveraging machine learning and vision-based analysis,” *Computer Vision and Image Understanding*, vol. 251, p. 104228, 2025.
- [43] I. Chelly, S. E. Finder, S. Ifergane, and O. Freifeld, “Trainable highly-expressive activation functions,” in *European Conference on Computer Vision*. Springer, 2024, pp. 200–217.
- [44] K. S. I. Mantri, X. Wang, C.-B. Schönlieb, B. Ribeiro, B. Bevilacqua, and M. Eliasof, “Digraf: Diffeomorphic graph-adaptive activation function,” in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 3649–3681. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/06cf4bae7ccb6ea37b968a394edc2e33-Paper-Conference.pdf
- [45] S. Lohit, Q. Wang, and P. 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*, 2019, pp. 12 426–12 435.
- [46] E. Nunez and S. H. Joshi, “Deep learning of warping functions for shape analysis,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 866–867.
- [47] E. Nunez, A. Lizarraga, and S. H. Joshi, “Srvfnet: A generative network for unsupervised multiple diffeomorphic functional alignment,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4481–4489.
- [48] C. Chen and A. Srivastava, “Srvfnet: Elastic function registration using deep neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4462–4471.
- [49] N. S. Detlefsen, “libcpab,” <https://github.com/SkafteNicki/libcpab>, 2018.
- [50] M. F. Beg, M. I. Miller, A. Trouvé, and L. Younes, “Computing large deformation metric mappings via geodesic flows of diffeomorphisms,” *IJCV*, 2005.
- [51] M. Hüskens and P. Stagge, “Recurrent neural networks for time series classification,” *Neurocomputing*, vol. 50, pp. 223–235, 2003.
- [52] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, “Recurrent neural network based language model.” in *Interspeech*, vol. 2, no. 3. Makuhari, 2010, pp. 1045–1048.
- [53] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series classification: a review,” *arXiv preprint arXiv:1809.04356*, 2018.
- [54] Z. Wang, W. Yan, and T. Oates, “Time series classification from scratch with deep neural networks: A strong baseline,” in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 1578–1585.
- [55] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European conference on computer vision*. Springer, 2016, pp. 630–645.
- [56] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, “Inceptiontime: Finding alexnet for time series classification,” *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.
- [57] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-first AAAI conference on artificial intelligence*, 2017.
- [58] J. Wang, Z. Wang, J. Li, and J. Wu, “Multilevel wavelet decomposition network for interpretable time series analysis,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2437–2446.
- [59] V. Arsigny, O. Commowick, X. Pennec, and N. Ayache, “A log-euclidean polyaffine framework for locally rigid or affine registration,” in *BIR*. Springer, 2006.
- [60] S. Durrleman, S. Allassonnière, and S. Joshi, “Sparse adaptive parameterization of variability in image ensembles,” *IJCV*, 2013.
- [61] S. Allassonniere, S. Durrleman, and E. Kuhn, “Bayesian mixed effect atlas estimation with a diffeomorphic deformation model,” *SIAM Journal on Imaging Sciences*, 2015.
- [62] O. Freifeld, “Deriving the CPAB derivative,” Ben-Gurion University, Tech. Rep., 2018.
- [63] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.
- [64] G. Doras and G. Peeters, “A prototypical triplet loss for cover detection,” in *ICASSP*. IEEE, 2020.
- [65] T. M. Wigley, K. R. Briffa, and P. D. Jones, “On the average value of correlated time series, with applications in dendroclimatology and hydrometeorology,” *Journal of climate and Applied Meteorology*, vol. 23, no. 2, pp. 201–213, 1984.
- [66] D. Gusfield, *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, 1997.
- [67] M. Cuturi and A. Doucet, “Fast computation of wasserstein barycenters,” in *International Conference on Machine Learning*, 2014, pp. 685–693.
- [68] T. Terefe, M. Devanne, J. Weber, D. Hailemariam, and G. Forestier, “Time series averaging using multi-tasking autoencoder,” in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2020, pp. 1065–1072.
- [69] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, 2014. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [70] I. Oguiza, “tsai - a state-of-the-art deep learning library for time series and sequential data,” Github, 2022. [Online]. Available: <https://github.com/timeseriesAI/tsai>
- [71] J. Dauxois, A. Pousse, and Y. Romain, “Asymptotic theory for the principal component analysis of a vector random function: some applications to statistical inference,” *Journal of multivariate analysis*, vol. 12, no. 1, pp. 136–154, 1982.
- [72] J. O. Ramsay and C. Dalzell, “Some tools for functional data analysis,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 53, no. 3, pp. 539–561, 1991.
- [73] A. H. Williams, B. Poole, N. Maheswaranathan, A. K. Dhawale, T. Fisher, C. D. Wilson, D. H. Brann, E. M. Trautmann, S. Ryu, R. Shusterman *et al.*, “Discovering precise

- temporal patterns in large-scale neural recordings through robust and interpretable time warping,” *Neuron*, vol. 105, no. 2, pp. 246–259, 2020.
- [74] G. Erez, R. S. Weber, and O. Freifeld, “A deep moving-camera background model,” in *European Conference on Computer Vision*. Springer, 2022, pp. 177–194.
- [75] N. Barel, R. S. Weber, N. Mualem, S. E. Finder, and O. Freifeld, “Spacejam: a lightweight and regularization-free method for fast joint alignment of images,” in *European Conference on Computer Vision*. Springer, 2024, pp. 180–197.