BlockDTW: Efficient and Scalable Similarity Search Algorithm for Healthcare-Focused Time-Series

Alberto Zancanaro
Luxembourg Centre for Systems Biomedicine
Belval, Luxembourg
alberto.zancanaro@uni.lu

Leonardo Badia
Dept. Inf. Engineering
University of Padova, Italy
leonardo.badia@unipd.it

Giulia Cisotto
Dept. Mathematics, Informatics, and Geosciences, University of Trieste, Italy
giulia.cisotto@units.it

***Abstract*—Digital twins for engineering, finance, and especially personalized medicine often rely on time-series applications, such as computing the similarity between temporal signals. Dynamic Time Warping (DTW) remains the gold standard for robust alignment under temporal distortions, but its quadratic time and space complexity limits scalability and real-time usage. Existing methods such as FastDTW, PruneDTW, and SoftDTW attempt to address these issues, but often compromise on accuracy, differentiability, or flexibility. We introduce BlockDTW, a differentiable parallel approximation of DTW that divides time-series into non-overlapping blocks and computes local alignments. This reduces complexity to $\mathcal{O}(bN)$, enabling efficient training and inference. BlockDTW achieves good approximation relative to DTW with up to $8\times$ speedup, as shown in three tasks: synthetic frequency-varying sinusoids, Trace dataset prediction with an FFNN, and EEG reconstruction using hvEEGNet. Results are comparable to SoftDTW and PruneDTW, with significantly lower runtime.**

***Index Terms*—Medical digital twins; Time series; Dynamic time warping; Similarity search; EEG; Deep learning.**

I. Introduction

The concept of a digital twin mirrors the human mind's creation of mental imagery, enabling experimentation and early anomaly detection [1], [2]. In medicine, software representations of patients can support personalized treatments and timely interventions [3], [4]. Recent studies reviewed human digital twins for healthcare [5], e.g., wearable ECG data for real-time arrhythmia detection. Dynamic biometrics and other physiological signals (EEG, ECG, glucose levels) can be leveraged to build such twins, whose reliability depends on assessing time-series similarity between the twin and the real subject. Dynamic Time Warping (DTW) [6] is the reference similarity measure, aligning time-series non-linearly to minimize distance. However, its quadratic time complexity limits scalability. Several methods address this, such as FastDTW [7], PruneDTW [8], and SoftDTW [9], each trading off between speed, accuracy, and differentiability. Yet, DTW's sequential dynamic programming [10] still hinders parallelization, which could greatly accelerate GPU/TPU computation.

We propose BlockDTW, a parallel, differentiable DTW designed to bridge traditional alignment and deep learning for scalable, real-time, end-to-end systems, especially in medicine [11]. It splits time-series into non-overlapping blocks, computing and aggregating local alignments, reducing complexity via block size b instead of series length N. We tested it against SoftDTW and PruneDTW on synthetic sinusoids, FFNN prediction, and multi-channel EEG reconstruction with hvEEGNet [12], achieving comparable accuracy with significant speed-ups and GPU-parallelization potential.

This paper is organized as follows. In Section II, we review the background on DTW and the proposals made in the literature for differentiability and/or scalability. Section III details our proposed methods. We present numerical results in Section IV and conclude in Section V.

Leonardo Badia has been supported by the Italian PRIN project 2022PNRR "DIGIT4CIRCLE," project code P2022788KK. Giulia Cisotto acknowledges project D86-RIC-NA-CISOTTO funded by the University of Trieste for newly hired early and mid-career researchers.

II. Dynamic Time Warping

A. Foundational Work

Let us define $x[i]$ and $y[j]$ as two time-series with N and M samples, respectively, $i = 1, 2, \ldots, N$, and $j = 1, 2, \ldots, M$. The problem is to compute a measure of their similarity (or, their distance). A recent survey on the most commonly used distance measures is available in [13]. *Euclidean distance*, for example, is fundamental to compute the mean square error (MSE) and its variants (root MSE, mean absolute error) between x and y: these methods sum the (squared) difference between corresponding time points, giving an aggregated measure of their time-to-time similarity. To note, this assumes that the two time series have the same length (or, the last samples of the longer sequence are discarded). *Cross-correlation function*, on the other hand, quantifies the similarity between two time-series as the integral of their point-wise product, at any given shift in time of the second one w.r.t. the first. *Cosine similarity* has also been employed to quantify adherence of one time-series to another. However, it suffers from limitations for time-series and is confined to specific domains (e.g., information retrieval [14]). All the above measures turned out to be very sensitive to small distortions or shifts of one time-series w.r.t. the other.

Thus, in the 1970s the dynamic programming algorithm called *Dynamic Time Warping (DTW)* was introduced [6]. The idea to dynamically align two time-series was first conceptualized in 1975 in [15], then brought to its modern version by [6], and finally popularized by [16] for time-series analysis.

The algorithm operates in three main steps: (i) a cost matrix Δ is created, associating its rows and columns to the

amplitude values of $x(i)$ and $y(j)$, respectively. (ii) Starting from $\Delta(0,0)$, the value of each element is computed as

$$\begin{aligned}\Delta(i,j) = \delta(x[i], y[j]) \ &+ \min[\Delta(i-1, j-1), \\ &+ \Delta(i, j-1), \Delta(i-1, j)],\end{aligned} \tag{1}$$

if $i, j > 1$, otherwise $\Delta(i,j) = \delta(x(i), y(j))$, where $\delta(\cdot)$ is a distance function. (iii) The final DTW score is the lower right corner $\Delta(N, M)$ of the cost matrix. A pseudocode for this procedure is reported in Algorithm 1.

Algorithm 1 Full DTW Algorithm

$x \in \mathbb{R}^N, y \in \mathbb{R}^M$
$\Delta_{0,0} = 0, \Delta_{0,1} = 0, \Delta_{1,0} = 0$
$\Delta_{i,0} = \infty, \Delta_{0,j} = \infty$ for $i, j \geq 2$
for $(i = 1;\ i \leq N;\ i = i + 1)$ **do**
  for $(j = 1;\ j \leq M;\ j = j + 1)$ **do**
    $\Delta_{\min} = \min(\Delta_{i-1,j}, \Delta_{i-1,j-1}, \Delta_{i,j-1})$
    $\Delta_{i,j} = \delta(x_i, y_j) + \Delta_{\min}$
  end for
end for
Return $\Delta_{N,M}$

Formally, DTW is computed as it follows:

$$\text{DTW}(x, y) = \min_{A \in \mathcal{A}_{n,m}} (\langle A, \Delta(x, y) \rangle) \tag{2}$$

with $\Delta(x, y)$ as the distance matrix defined in Algorithm 1, $\mathcal{A}_{n,m} \subset \{0, 1\}^{N \times M}$ as the set of binary alignment matrices [9], i.e., the matrices that contain a path from the upper left corner to the lower right corner, and $\langle \cdot, \cdot \rangle$ the inner product. To create a path, $\rightarrow, \downarrow, \searrow$ steps are allowed, only. The overall DTW *score* between time-series x and y is given by the output of Algorithm 1.

This algorithm has been applied in many different domains and it provided large advantages compared to other measures of similarity, especially for its robustness to warping, distortion, and shift, enabling data analytics solutions to work with real-world time-series [17]. Unfortunately, in its original formulation (called *Full DTW*), DTW has both time and space complexity of $\mathcal{O}(NM)$. As $N \approx M$, as a matter of fact DTW scales as $\mathcal{O}(N^2)$ in time and memory. Since its first publication, many different versions have been proposed to address computational efficiency, without excessively sacrificing other important requirements such as accuracy, scalability, differentiability, non-negativity, and real-time constraints for applications like Brain-Computer Interface (BCI) [18].

B. State of the art in DTW computation

Here, we discuss previous DTW implementations aligned with our objectives. FastDTW [19] is a linear-time and space approximation of DTW, designed to scale by recursively computing DTW on downsampled series and refining the path within a constrained window. It achieves very good DTW approximation with runtime reductions of several orders of magnitude (complexity O(N)), but may miss optimal paths in highly non-linear alignments. Despite widespread use, recent works [7] question its real efficiency.

PruneDTW [8] accelerates exact DTW using Keogh's lower bound [20] to prune unlikely cells from the cost matrix. On UCR datasets, it achieved very good DTW approximation with up to 20× speedups, though its performance drops when sequences are highly distorted.

Stochastic DTW [21] offers a probabilistic ensemble of alignment paths, enabling uncertainty modelling (e.g., for seismic data). While robust to noise, its reliance on sampling increases computational cost, limiting real-time and deep learning integration.

SoftDTW [9] is the most common differentiable DTW approximation, reformulating DTW as a smooth loss by replacing the minimum in (2) with a soft-min operator. Given a series of n numbers, $[a_1, ..., a_n]$, the *soft* minimum operation is defined as

$$\min^{\gamma}(a_1, ..., a_n) = -\gamma \log \sum_{i=1}^{n} e^{-a_i/\gamma} \tag{3}$$

with $\gamma > 0$ being a smoothing hyperparameter. The lower γ, the closer (3) to the minimum. Then, SoftDTW is defined as

$$\text{SDTW}_{\gamma}(x, y) = \min^{\gamma}(\langle A, \Delta(x, y) \rangle, A \in \mathcal{A}_{n,m}) \tag{4}$$

$$= -\gamma \log \sum_{A \in \mathcal{A}_{n,m}} e^{\langle A, \Delta(x,y) \rangle / \gamma} \tag{5}$$

With SoftDTW, each sample of a time-series is compared with every sample of the other. This means that the memory usage and the number of computations scales as $\mathcal{O}(N^2)$.

It has to be noted that SoftDTW can assume negative values and it is not minimized when $x = y$. This limitation has been mitigated by a variation of SoftDTW called DTW divergence (DTW-div) [22]. DTW-div is simply derived from any measure of DTW by the following formula:

$$D_{\gamma}(x, y) = \text{DTW}_{\gamma}(x, y) - \frac{1}{2}(\text{DTW}_{\gamma}(x, x) + \text{DTW}_{\gamma}(y, y)) \tag{6}$$

DTW-div ensures non-negativity and minimization when the two time-series are very similar, at the cost of higher computation time w.r.t. a single DTW. Specifically, DTW-div is affected by a factor 3, as DTW needs to be computed three times (this could be overcome by parallel computation).

In [26], the authors proposed the Optimal Transport Warping (OTW) algorithm which showed linear time and space complexity and differentiability. Also, the authors claimed the algorithm is parallelized. However, performance was tested on different datasets w.r.t. this work (a simpler synthetic dataset including triangle or square signals with some amount of time shift, plus white noise) and different tasks (specifically, classification via deep learning and 1-nearest neighbour clustering, and hierarchical clustering), making it difficult to fairly compare the two works.

A very recent article introduced *TimePoint* [27], an algorithm whose alignment strategy relies on discrete rules (keypoint extraction and matching). While promising, this method does not allow differentiation or backpropagation.

TABLE I
STATE OF THE ART AND BLOCKDTW

Method	Time	Space	Approx	Diff.
Full DTW [6]	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	Baseline	No
FastDTW [19]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	Very good	No
PrunedDTW [8]	$\mathcal{O}(kN)$	$\mathcal{O}(N^2)$	Very good	*
Stochastic DTW [23]	$\mathcal{O}(kN^2)$	$\mathcal{O}(kN^2)$	Good	No
SoftDTW [9]	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	Good	Yes
SmoothDTW [24]	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	Good	***
DecDTW [25]	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	Very good	***
OTW [26]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	Good	Yes
TimePoint [27]	$\mathcal{O}(LL')^{\dagger}$	$\mathcal{O}(LL')$	Good	No
BlockDTW	$\mathcal{O}(bN)^{\dagger\dagger}$	$\mathcal{O}(bN)^{\dagger\dagger}$	Good	Yes

*only if *min* is replaced with *softmin*; **model-dependent; ***implicit differentiation (computationally heavier than native differentiation); † $L << N$, $L' << M$ number of keypoints in the two time-series; $^{\dagger\dagger}\mathcal{O}(b)$ in case of parallel implementation.

Table I reports a list of the most relevant and recent algorithms for DTW computation. They are compared based on their time and space complexity, approximation quality w.r.t. the Full DTW (assuming DTW gives the best distance path between the time-series, i.e., distance measurement accuracy), and differentiability. We included other methods, not discussed here due to space constraints.

In line with [19], we adopt an *abstraction* strategy, where DTW is computed on a reduced representation of the data. We call this solution *BlockDTW*, and we show how it is computationally efficient ($\mathcal{O}(N)$), accurate, differentiable, scalable, and feasible for real-time scenarios such as BCI.

III. METHODS

A. BlockDTW

To reduce memory consumption and computations to obtain DTW, we propose to divide the two signals into non-overlapping blocks and compute the SoftDTW (or, the SoftDTW divergence), between corresponding blocks, i.e., the first block of a signal with the first block of the other. This is repeated for the second block and the following ones as well. Formally, if we consider two time-series of the same length N and choose block size b, then each time-series is divided into $\frac{N}{b}$ blocks. Then, a DTW measure is obtained from each block (via SoftDTW), providing N/b independent measures, which are summed. Given N/b SoftDTW computations on a pair of b-long segments (i.e., each one with complexity of $\mathcal{O}(\left(\frac{N}{b}\right)^2)$, the overall complexity results in $\mathcal{O}(bN)$. A depiction of our *abstraction* strategy is shown in Fig. 1.

An implementation is shown in Algorithm 2. Note that if $\text{mod}(N, b) = 0$ (i.e., b divides N without remainder) then the *else* branch in the algorithm is unnecessary.

A formal definition of the final *score* Δ_B is given by

$$\Delta_B = \sum_{k=1}^{k=\lceil N/b \rceil} \text{SoftDTW}(x_{I_k}, y_{I_k}) \tag{7}$$

with I_k being the indices of the k-th block, and x_{I_k}, y_{I_k} the sample of the two time-series to compare for block k.

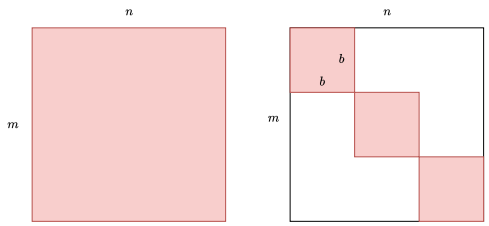

Fig. 1. Constraints for DTW computation for (left) SoftDTW and (right) BlockDTW. In line with [19], a constraint represents an abstraction strategy (i.e., a way to decrease resolution in favor of a lightweight computation).

Algorithm 2 BlockDTW Algorithm

$x \in \mathbb{R}^N, y \in \mathbb{R}^N$
$\Delta_B = 0$
for $(k = 0;\ k \leq \lfloor N/b \rfloor;\ k = k + 1)$ **do**
  if $k \leq \lfloor N/b \rfloor - 1$ **then**
    $I_k = [kb, kb{+}1, ..., (k{+}1)b - 1]$
  else
    $I_k = [kb, kb + 1, ..., N]$
  end if
  $x_k = x[I_k]$
  $y_k = y[I_k]$
  $\Delta_B = \Delta_B + \text{SoftDTW}(x_k, y_k)$
end for
Return Δ_B

B. Implementation

Our implementation of BlockDTW takes advantage from the most effective (to the best of our knowledge) implementation of SoftDTW. Currently, SoftDTW has not been implemented in the Pytorch framework. Based on some preliminary empirical tests, we adopted the implementation of the open-source package by [28], available on GitHub. Unfortunately, this implementation via CUDA [28] is limited to input sequences of up to 1024 samples due to GPU constraints. Our BlockDTW method overcomes this by processing longer signals in segmented blocks of up to 1024 samples. Importantly, most electrophysiological signals—such as EEG, ECG or EMG—are sampled at 250 Hz, 500 Hz, or 1000 Hz. At these sampling rates, 1024 samples correspond to segments lasting approximately 4, 2, or 1 s, respectively. These durations align with or even exceed the typical segment lengths used in real-world applications (e.g., 0.125–2 s commonly seen in electrophysiology [29]). Moreover, longer segments may become non-stationary or redundant when the lowest frequencies of interest are around 0.5–1 Hz, making our block-based approach both practical and effective.

Our implementation of BlockDTW is available on GitHub [30].

IV. RESULTS AND DISCUSSION

We show the promising performance of BlockDTW, in terms of accuracy w.r.t. SoftDTW and PruneDTW, its effectiveness as loss function for high-fidelity reconstruction

of multi-channel time-series recordings of brain activity (through electroencephalography), and its computational efficiency w.r.t. parameter b.

In the first case study, we generated a sinusoidal time-series with fundamental frequency f_1 and amplitude A_1, and computed its *similarity* w.r.t. a second sinusoid with amplitude A_2 but variable frequency (f_2). We computed the similarity using SoftDTW, PruneDTW (with two different levels of pruning, namely *bandwidth values* at 5 and 10), and BlockDTW with $b = 100$ and 150. Fig. 2 presents two examples with the fundamental frequency set to 1 Hz and 17 Hz, respectively. We can observe (Fig. 2(a)) that when the two sinusoids are

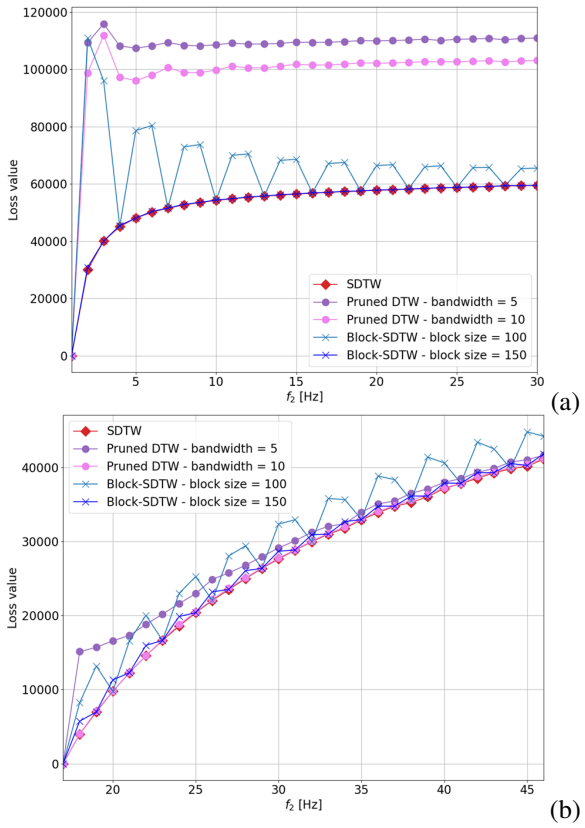

Fig. 2. Case study no.1 (variable-frequency sinusoids). Comparison between similarity measurements provided by SoftDTW, PruneDTW (with bandwidth set to 5 and 10), and BlockDTW (with $b = 100, 150$). (a) Fundamental frequency at 1 Hz. (b) Fundamental frequency at 17 Hz.

the same (e.g., $f_2 = f_1$), then all computations of DTW return a zero value. As soon as the frequency difference between the two curves increases, then SoftDTW increases smoothly, while BlockDTW approximates SoftDTW depending on the value of b. More specifically, the smaller b, the more sensitive it is to local variations between the two sinusoids and may eventually return a higher value compared to SoftDTW. On the other hand, in this case, PruneDTW provides a very poor approximation, no matter what value f_2 takes. Interestingly, we also tested the case of fundamental frequency at 17 Hz (see Fig. 2(b)), and observed that all curves start at the origin (i.e., when the sinusoids are the same) and then all of them stably increase. As expected, SoftDTW has a smoother pattern, as its algorithm requires averaging all possible paths in the cost matrix. PruneDTW shows much better performance compared to the previous case, especially for the bandwidth value set to 10 and larger f_2 values. On the other hand, BlockDTW is approaching SoftDTW with a periodicity that depends on the relationship between the sampling frequency, the block size and the frequency of the sinusoid.

In the second study, we predicted the last 100 samples of time-series included in four different public datasets, specifically: *Trace* inside the *tslearn* library [31], derived from nuclear power plants, and the *Wafer*, *Wine*, and *Ham* included in the benchmark *UCR Archive* [32].

We trained a feedforward neural network (FFNN) with 2 hidden layers, each one provided with 256 neurons. The hidden layers use a SELU activation function and batch normalization. No activation is applied to the output. For training, we used stochastic gradient descent with a learning rate of 0.001 for 60 epochs. No learning rate scheduler was used. One FFNN model was obtained for each distance measure: a first model was trained using SoftDTW, a second FFNN by using BlockDTW, and a third one by using PruneDTW. For both SoftDTW and BlockDTW, we set $\gamma = 1$ (default value). For BlockDTW, we used a block size of $b = 30$ samples. For PruneDTW, we set the bandwidth value to 10. Predictions are compared in Fig. 3. All algorithms (SoftDTW, PruneDTW, and BlockDTW) provided similar results, showing that BlockDTW can achieve the same high accuracy in the prediction task w.r.t. the state of the art. Slightly different behaviour is observed, depending on the specific type of waveform. However, a systematic comparison deserves a dedicated investigation, which is left for a future work.

Finally, in the third case study, we accurately reconstruct multi-channel EEG recordings, as derived from a popular public EEG dataset called *BCI competition IV dataset 2a* (sampling frequency of 250 Hz) [33]. We employed our recent model called *hvEEGNet* [12], based on a hierarchical variational autoencoder architecture [34], and engineered to be specific for high-fidelity reconstruction of multi-channel brain recordings. In our previous work [12], we employed SoftDTW to train the model, achieving very high performance in all the time-series included in the dataset. In Fig. 4, we show an example of how BlockDTW allows for a satisfactory reconstruction. We set $b = 125$ samples, corresponding to 0.5 s, i.e., the minimum time interval containing significant information about brain dynamics related to the task accomplished by the participants of the study [29]. As expected, the quality of the output is lower compared to the training based on SoftDTW (actually, DTW-div), but much faster.

A. Computational efficiency

We compare SoftDTW and BlockDTW in terms of computational time with variable input lengths and block sizes. Note that we tested them on two different hardware setups: an Intel(R) Core(TM) i7-8700 CPU@3.20GHz, and an Intel(R) Core(TM) i7-10750H CPU@2.60GHz. The software setup was the same in both cases: Linux Operating System

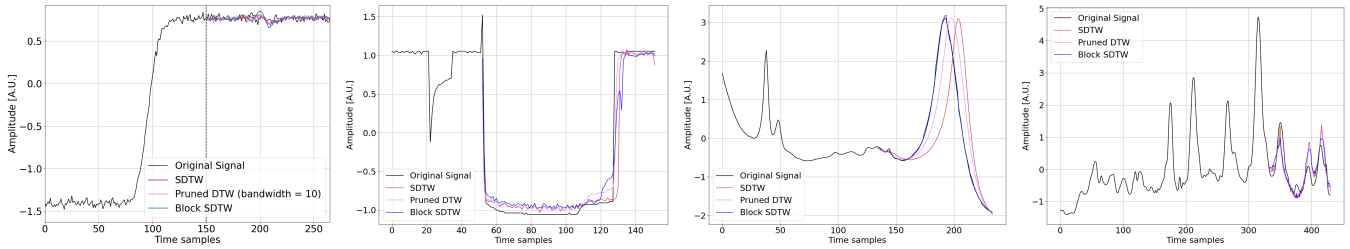

Fig. 3. Case study no.2: time-series prediction based on an FFNN model: (a) *Trace* dataset, (b) UCR Archive - *Wafer*, (c) UCR Archive - *Wine*, (d) UCR Archive - *Ham*. Every subplot reports four curves: the light blue one for the original time-series to predict, the orange one for the prediction based on SoftDTW, and the red one for the model trained with BlockDTW.

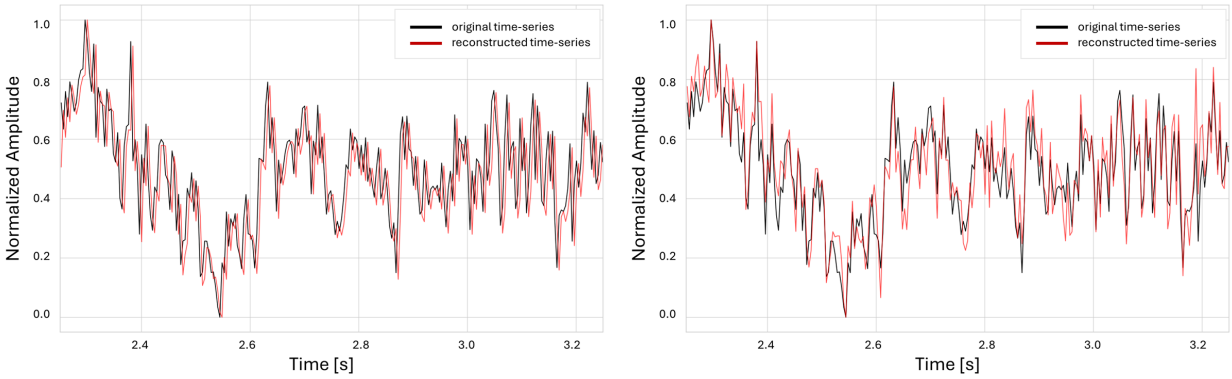

Fig. 4. Case study no.3: EEG reconstruction via hvEEGNet model. The latter was trained with (a) SoftDTW or (b) BlockDTW, respectively.

(Ubuntu), provided with 16 GB RAM. Repeated measures were collected in the same configuration (i.e., hardware setup, input length, block size) and then averaged to increase the statistical robustness. From Fig. 5, we observe that SoftDTW rapidly increases in input length (as expected due to quadratic scaling). On the other hand, BlockDTW generally performs better, depending on the specific hardware and the block size. As expected, the largest advantage is obtained when the block size is large and the input is long. For example, with two 1000 sample-long time-series as input, SoftDTW takes 40 ms to compute their similarity, while BlockDTW with $b = 50$ only 5 ms, reaching an $8\times$ time saving, as seen in Fig. 5(a). Note that PruneDTW is expected to have similar computational efficiency as BlockDTW, then we refer to a future extended work to expand the comparison.

V. Conclusions

With the growing role of digital twins in healthcare, especially for real-time monitoring and personalized treatment, efficient and accurate comparison of physiological time-series is essential. In this work, we presented BlockDTW, a differentiable and efficient variant of SoftDTW for computing DTW distances robust to shifts, warpings, and distortions common in real-world data. BlockDTW significantly speeds up DTW computation while preserving accuracy. It is also differentiable and potentially parallelizable — key features for deploying digital twins based on dynamic biometric data like EEG or ECG. We demonstrated its effectiveness in three tasks: aligning synthetic sinusoids, forecasting the *Trace* dataset and three different datasets of the benchmark UCR Archive with an FFNN, and reconstructing EEG signals using our hvEEGNet model. Future work will explore the application of BlockDTW to further tasks including nearest-neighbour search, hierarchical clustering, and anomaly detection. Also, we will compare it as a loss function for the hvEEGNet model w.r.t. other state of the art methods (e.g., PruneDTW). We also plan to implement a fully parallelized version of BlockDTW to further leverage GPU architectures. Finally, our approach will be compared to artificial neural networks-based alignment methods, such as TAP [35], which employs lightweight CNNs to directly predict alignment, and to stochastic or uncertainty-aware DTW variants like Stochastic DTW [21] and uDTW [36], which incorporate uncertainty into the alignment process.

References

[1] K. Duran and B. Canberk, "Digital twin enriched green topology discovery for next generation core networks," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 4, pp. 1946–1956, 2023.

[2] A. Buratto and L. Badia, "Massive opportunistic sensing with limited collaboration for age of information," in *Proc. IEEE Wirel. Commun. Netw. Conf. (WCNC)*, 2024, pp. 1–6.

[3] M. Roopa and K. Venugopal, "Digital twins for cyber-physical healthcare systems: Architecture, requirements, systematic analysis and future prospects," *IEEE Access*, vol. 13, pp. 44 963–44 996, 2025.

[4] A. Zancanaro, G. Cisotto, and L. Badia, "Modeling value of information in remote sensing from correlated sources," *Comput. Commun.*, vol. 203, pp. 289–297, 2023.

[5] Z. Johnson and M. J. Saikia, "Digital twins for healthcare using wearables," *Bioengineering*, vol. 11, no. 6, p. 606, 2024.

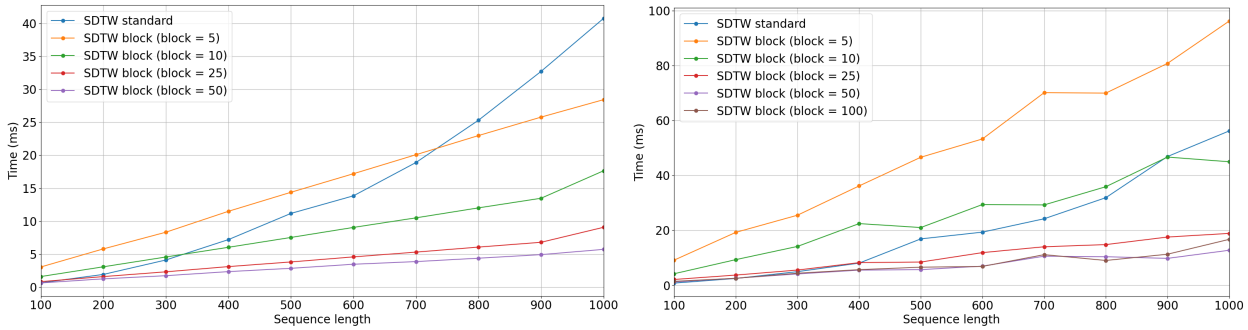

Fig. 5. Computational time of SoftDTW and BlockDTW w.r.t. the input length (expressed in number of samples). Hardware setups: (a) Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz. (b) Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz. Software setup: Linux Operating System (Ubuntu), provided with 16 GB RAM.

[6] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 26, no. 1, pp. 43–49, 1978.

[7] R. Wu and E. J. Keogh, "FastDTW is approximate and generally slower than the algorithm it approximates," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3779–3785, 2020.

[8] D. F. Silva and G. E. Batista, "Speeding up all-pairwise dynamic time warping matrix calculation," in *Proc. SIAM Int. Conf. Data Mining*, 2016, pp. 837–845.

[9] M. Cuturi and M. Blondel, "Soft-DTW: A differentiable loss function for time-series," in *Proc. Int. Conf. Machine Learning (ICML)*, vol. 70, 2017, pp. 894–903.

[10] A. Buratto, H. Tuwei, and L. Badia, "Optimizing sensor data transmission in collaborative multi-sensor environments," in *Proc. IEEE Int. Conf. Commun. Netw. Satellite (COMNETSAT)*, 2023, pp. 635–639.

[11] G. Cisotto, M. Capuzzo, A. V. Guglielmi, and A. Zanella, "Feature selection for gesture recognition in internet-of-things for healthcare," in *Proc. IEEE Int. Conf. Commun. (ICC)*. IEEE, 2020, pp. 1–6.

[12] G. Cisotto, A. Zancanaro, I. F. Zoppis, and S. L. Manzoni, "hvEEGNet: a novel deep learning model for high-fidelity EEG reconstruction," *Front. Neuroinf.*, vol. 18, p. 1459970, 2024.

[13] J. Paparrizos, H. Li, F. Yang, K. Wu, J. E. d'Hondt, and O. Papapetrou, "A survey on time-series distance measures," *arXiv preprint arXiv:2412.20574*, 2024.

[14] A. Raganato, Y. Scherrer, and J. Tiedemann, "An evaluation benchmark for testing the word sense disambiguation capabilities of machine translation systems," in *Proc. Language Res. Eval. Conf.*, 2020, pp. 3668–3675.

[15] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 23, no. 1, pp. 67–72, 1975.

[16] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 1994, pp. 359–370.

[17] C. Serantoni, A. Riente, A. Abeltino, G. Bianchetti, M. M. De Giulio, S. Salini, A. Russo, F. Landi, M. De Spirito, and G. Maulucci, "Integrating dynamic time warping and k-means clustering for enhanced cardiovascular fitness assessment," *Biomed. Signal Process. Control*, vol. 97, p. 106677, 2024.

[18] M. A. Mohamed, S. Mansour, P. Soulatiantork, K. K. Ang, P. K. Soon, and M. Arvaneh, "Improving common spatial patterns in brain-computer interface using dynamic time warping and EEG normalization," in *Proc. IEEE Int. Conf. Metrology Extend. Reality Artif. Intell. Neural Eng. (MetroXRAINE)*, 2023, pp. 1027–1032.

[19] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2007.

[20] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowl. Inf. Syst.*, vol. 7, no. 3, pp. 358–386, 2005.

[21] R. Cheverry, J. Edwards, and G. Caumon, "Seismic-to-well tie using stochastic dynamic time warping," in *Proc. GOCAD - RING Meeting*. ASGA, 2015.

[22] M. Blondel, A. Mensch, and J.-P. Vert, "Differentiable divergences between time series," in *Proc. Int. Conf. Artif. Intell. Stat.*, vol. 130, 2021, pp. 3853–3861.

[23] S. Nakagawa and H. Nakanishi, "Speaker-independent english consonant and japanese word recognition by a stochastic dynamic time warping method," *IETE J. Res.*, vol. 34, no. 1, pp. 87–95, 1988.

[24] L. Mensch and M. Blondel, "Differentiable dynamic programming for structured prediction and attention," in *Proc. Int. Conf. Machine Learning (ICML)*, vol. 80, 2018, pp. 3462–3471.

[25] M. Xu, S. Garg, M. Milford, and S. Gould, "Deep declarative dynamic time warping for end-to-end learning of alignment paths," in *Proc. Int. Conf. Learning Repres. (ICLR)*, 2023.

[26] F. Latorre, C. Liu, D. Sahoo, and S. C. Hoi, "Otw: Optimal transport warping for time series," in *Proc. Int. Conf. Acoustics Speech Signal Process. (ICASSP)*, 2023, pp. 1–5.

[27] R. S. Weber, S. Benishay, A. Lavrinenko, S. E. Finder, and O. Freifeld, "Timepoint: Accelerated time series alignment via self-supervised keypoint and descriptor learning," in *Proc. Int. Conf. Machine Learn. (ICML)*, 2025. [Online]. Available: https://openreview.net/forum?id=bUGdGaNFhi

[28] M. Maghoumi, E. M. Taranta, and J. LaViola, "DeepNAG: Deep non-adversarial gesture generation," in *Proc. Int. Conf. Intell. User Interf.*, 2021, pp. 213–223, code available at: https://github.com/Maghoumi/pytorch-softdtw-cuda.

[29] R. Yuvaraj, S. Samyuktha, J. Fogarty, J. S. Huang, S. Tan, and T. K. Wong, "Optimal eeg time window length for boredom classification using combined non-linear features," in *2024 32nd European Signal Processing Conference (EUSIPCO)*. IEEE, 2024, pp. 1756–1760.

[30] "BlockDTW: Block-wise (soft) Dynamic Time Warping," https://github.com/jesus-333/block-sdtw/tree/main, accessed: Aug. 11, 2025.

[31] M. Middlehurst, P. Schäfer, and A. Bagnall, "Bake off redux: a review and experimental evaluation of recent time series classification algorithms," *Data Mining Knowledge Disc.*, pp. 1–74, 2024.

[32] 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.

[33] B. Blankertz, G. Dornhege, M. Krauledat, K.-R. Müller, and G. Curio, "The non-invasive Berlin Brain-Computer Interface: Fast acquisition of effective performance in untrained subjects," *NeuroImage*, vol. 37, pp. 539–50, 09 2007.

[34] A. Vahdat and J. Kautz, "NVAE: A deep hierarchical variational autoencoder," *Adv. Neur. Inf. Proc. Syst.*, vol. 33, pp. 19 667–19 679, 2020.

[35] B. Su and J.-R. Wen, "Temporal alignment prediction for supervised representation learning and few-shot sequence classification," in *Int. Conf. Learning Repres.*, 2021.

[36] L. Wang and P. Koniusz, "Uncertainty-DTW for time series and sequences," in *Eur. Conf. Comp. Vis.*, 2022, pp. 176–195.