Time Elastic Neural Networks

Pierre-François Marteau

Time Elastic Neural Networks

Pierre-François Marteau, PIERRE-FRANCOIS.MARTEAU@UNIV-UBS.FR
Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)
Université Bretagne Sud
Vannes, France.

Editor: My editor

Abstract

We introduce and detail an atypical neural network architecture, called time elastic neural network (teNN), for multivariate time series classification. The novelty compared to classical neural network architecture is that it explicitly incorporates time warping ability, as well as a new way of considering attention. In addition, this architecture is capable of learning a dropout strategy, thus optimizing its own architecture.

Behind the design of this architecture, our overall objective is threefold: firstly, we are aiming at improving the accuracy of instance based classification approaches that shows quite good performances as far as enough training data is available. Secondly we seek to reduce the computational complexity inherent to these methods to improve their scalability. Ideally, we seek to find an acceptable balance between these first two criteria. And finally, we seek to enhance the explainability of the decision provided by this kind of neural architecture.

The experiment demonstrates that the stochastic gradient descent implemented to train a teNN is quite effective. To the extent that the selection of some critical meta-parameters is correct, convergence is generally smooth and fast.

While maintaining good accuracy, we get a drastic gain in scalability by first reducing the required number of reference time series, i.e. the number of teNN cells required. Secondly, we demonstrate that, during the training process, the teNN succeeds in reducing the number of neurons required within each cell. Finally, we show that the analysis of the activation and attention matrices as well as the reference time series after training provides relevant information to interpret and explain the classification results.

The comparative study that we have carried out and which concerns around thirty diverse and multivariate datasets shows that the teNN obtains results comparable to those of the state of the art, in particular similar to those of a network mixing LSTM and CNN architectures for example.

Keywords: Time series matching, Time elastic attention, Kernel methods, Neural Networks, Time series classification.

1 Introduction

This paper presents a neural network architecture for time series classification that explicitly incorporates time warping capability. Behind the design of this architecture, our overall objective is threefold: firstly, we are aiming at improving the accuracy of instance based classification approaches that shows quite good performances as far as enough training data

is available. Secondly we seek to reduce the computation time inherent to these methods to improve their scalability. In practice, we seek to find an acceptable balance between these first two criteria. And finally, we seek to enhance the explainability of the decision provided by this kind of neural architecture.

The approach we develop in this study is rooted into the theory of kernels Schoenberg (1938), which are essentially similarity measures to which an inner product corresponds in the so-called Reproducing Kernel Hilbert Space.

More precisely, the proposed architecture, called time elastic Neural Network (teNN) is derived directly from the Dynamic Time Warping Kernel (KDTW) proposed in Marteau and Gibet (2014b) and its novelty, compared to classical neural networks is the following.

1. The full network architecture is an assembly of competitive sub-networks called teNN cells. Each teNN cell is associated with three main components: i) an abstract time series, called a reference, ii) an activation matrix and iii) an attention matrix.

2. Within a cell, the output of any elementary neuron is the sum of its inputs (at most three) multiplied by the local kernel that evaluates the pairwise matching of time series samples (thus, the samples of the input time series are compared to that of the reference time series).

3. The inverse of the bandwidth of the local kernel is a parameter (ν) that is learned during training. A large value means high local attention, while a small value means a low attention, an area where we do not care any sample comparison. All attention parameters (inverse of the bandwidth values) are gathered within a teNN cell into an attention matrix.

4. each elementary neuron is associated to an activation weight that is learned during the training. Therefore, inactivated neurons after training can be dropped to simplify the neural architecture. In a way, the network is able to optimize its own architecture. All activation weights in a teNN cell are gathered into an activation matrix.

5. Finally, the samples of the reference time series are also learned during training.

While maintaining a good accuracy, we expect a drastic gain in scalability by first reducing the number of references, i.e. the number of necessary teNN cells. Secondly, we expect to be able to reduce the number of required neurons within a cell. Finally, we expect that the analysis of the activation and attention matrices as well as the references after learning will provide relevant information to interpret and explain the classification results.

The remaining par of the article is organized as follows. In section 2, we present a brief history of relevant works in the domain of time elastic matching, going from early definition of elastic distances to elastic kernel. Then, as the proposed architecture is essentially inspired from the implementation of KDTW, we detail the way this kernel has been constructed, some of its properties and its implementation in section 3. Section 4 is dedicated to the presentation of the proposed neural architecture. The differentiation of the teNN cells is detailed in section 5. A stochastic gradient descent is proposed to minimize a categorical cross entropy loss covering the entire teNN architecture. Section 6 shows some results obtained on synthetic and real datasets. We confront here our expectations to the experimental

reality. In section 7 we compare teNN accuracy to the state of the art in multivariate time series classification before concluding this study.

2 A brief history of time elastic matching and the root of time elastic neural network

The following survey on time elastic matching, that spans more than a century, as depicted in Fig.1, is indeed not exhaustive. We only focus on the works or results which seem enlightening to us in the context of the study presented in this article.

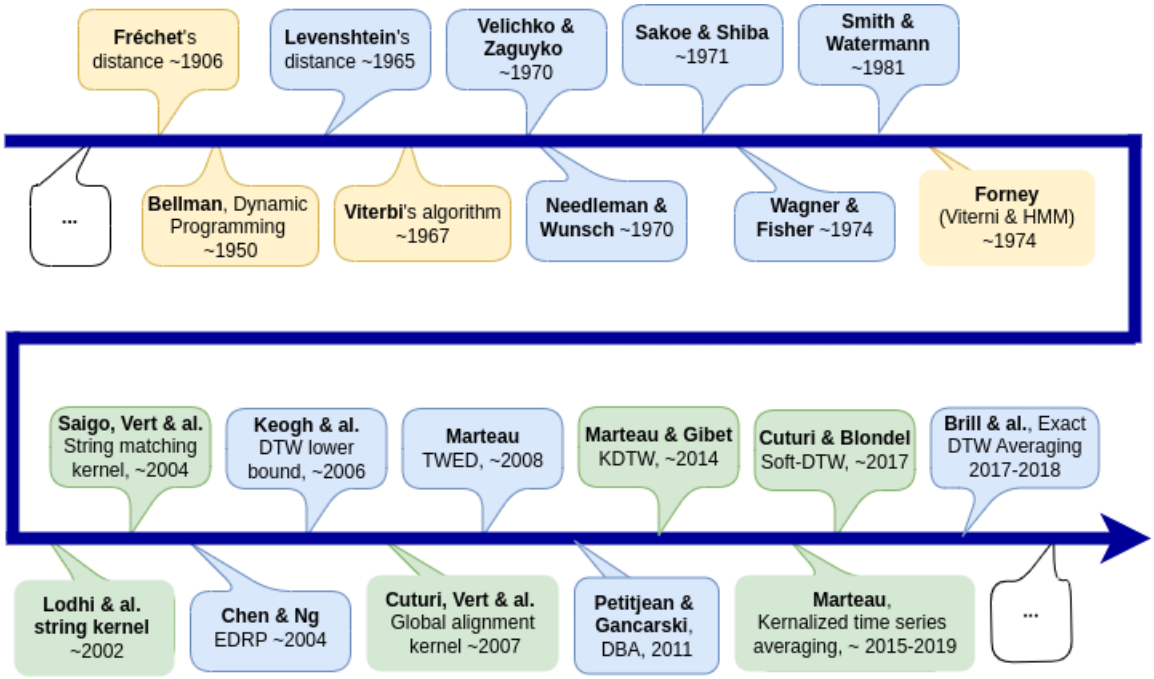

Figure 1: A non exhaustive history of time elastic matching for time series comparison. The founding work is presented in orange, the work on elastic distances in blue and the work on elastic kernel in green.

The concept of temporal elastic matching between two curves, x and y, was historically introduced by Maurice Fréchet Fréchet (1906) in the form of an eponymous distance. According to its mathematical formulation given in Eq.1, this pairwise distance is defined as an optimization problem in the space of monotonically increasing (temporal) functions. Formally, Fréchet defined the pairwise distance measure F between two time series x and y as:

$$F(x,y) = \underset{\alpha,\beta}{Inf} \ \underset{t\in[0,1]}{Max} \left\{ d\Big(x(\alpha(t)),\, y(\beta(t))\Big) \right\} \tag{1}$$
$$\forall t,\ \ \delta_t > 0,\ \ \alpha(t) \leq \alpha(t+\delta_t) \ \texttt{and} \ \beta(t) \leq \beta(t+\delta_t)$$

with $\alpha(t)$ and $\beta(t)$ two monotonic increasing temporal functions on which the optimization applies.

Hence, according to Fréchet's metaphoric illustration, the distance between two (3D) trajectories followed by a man accompanied with his dog is the minimum length of a leash required to connect the dog and its owner as they walk freely, but without going backward, along their respective paths from one endpoint to the other.

To our knowledge, this is the first time that the temporal way in which curves are traveled has been explicitly taken into account into the design of a distance between two time series. Elastic time matching was born.

Obviously, this formal definition was impossible to calculate efficiently in 1906, and it was not until the 1960s and 1970s, with the advent of the early computers, that we saw the first implementations of distance or similarity functions sharing with Fréchet's distance this concept of temporal elasticity. Earlier and latest implementations of such time elastic distances are based on the optimality principle developed by Bellman Bellman (1957). Once the so-called *Dynamic Programming* (DP) algorithm has been proposed by Bellman to solve complex (exponential) resource allocation problems in polynomial time, applications to time elastic matching spread rapidly across the computer science community. The Viterbi algorithm Viterbi (1967), used in Hidden Markov Model Forney (1973) to align with some elasticity a sequence of observable with a sequence of hidden states, paved the way to the use of DP in the scope of elastic matching. Subsequently to the Viterbi algorithm, Dynamic time Warping (DTW) was proposed Velichko and Zagoruyko (1970); Sakoe and Chiba (1971) in the context of speech recognition and then widely generalized to numerous application areas. About at the same time, Needlemann and Wunsch Needleman and Wunsch (1970) proposed an eponym algorithm to evaluate, using DP, the global maximal alignment of two strings. It has been widely used in bio-informatics, to align protein or nucleotide sequences. One can also mention the Levenshtein distance (also called edit distance) for string comparison Levenshtein (1966) that was originally proposed in the 1960s found ten years later a DP implementation Wagner and Fischer (1974) solving the pairwise distance evaluation in $O(n^2)$ complexity that greatly generalized its use, in particular as a spell checker and guesser. Its adaptation to the field of bioinformatics was proposed by Smith and Watermann Smith and Waterman (1981). Subsequently, other proposals seeking to satisfy the triangular inequality unsatisfied by DTW emerged such as the Edit Distance with Real Penalty (EDR)Chen and Ng (2004) and the Time Warp Edit Distance (TWED) Marteau (2008).

Meanwhile, the advent of support vector machines (SVM) in the early 1990s shed light on the theory of kernels Schoenberg (1938), opening a new path toward the development of time elastic kernel. First string kernels have been proposed Lodhi et al. (2002); Saigo et al. (2004) then elastic kernels for time series matching were designed, in particular the Global Alignment Kernel (GAK), Cuturi et al. (2007a) and the KDTW kernel Marteau and Gibet (2014b).

Some of these elastic distances have been intensively evaluated on numerous classification tasks such as in Bagnall et al. (2016) and Middlehurst et al. (2024). The study by Paparrizos et al. (Paparrizos et al. (2020)) specifically focusing on distance and kernels evaluation on 1NN classification tasks exploiting a set of 128 dataset from the UCR repository Chen et al. (2015) is likely to be the most exhaustive. Fig.2 showing that kernels com-

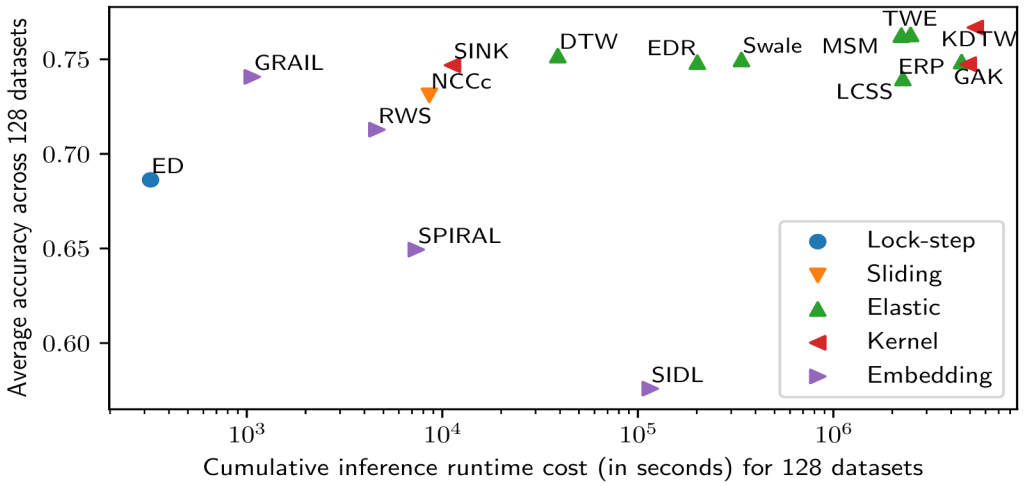

Figure 2: Ranking of elastic measures according to Paparrizos et al. study (Figure from Paparrizos et al. (2020)). Distance and kernel measures are evaluated on 128 datasets from the UCR archive.

pete advantageously (KDTW, GAK) in terms of accuracy, but at the cost of computational efficiency.

To conclude this short survey, elastic dissimilarity or similarity functions have more recently been the subject of complementary research to solve the problem of estimating the time elastic mean of a set of time series. On the similarity track essentially based on the DTW we can highlight the DBA algorithm Petitjean et al. (2011) and a proposal for exact calculation of a DTW average Brill et al. (2017). Regarding the kernel track, we mainly identify Soft-DTW Cuturi and Blondel (2017) and a kernelized version of DBA called TEKA (Time Elastic Kernilized Averaging) Marteau (2019).

Algorithmically speaking, DP allows to reduce computational complexity of all these time elastic measures to a polynomial function of the time series lengths, in general of degree 2.

3 The KDTW kernel

We detail in the following subsections the construction of the KDTW kernel, on which we largely relied to develop the teNN architecture.

3.1 Few definitions

The following definitions will be used through out the article.

Definition 1 *Time series:*

1. *A (discrete) time series is considered through out this article as a sequence* $x = x(1), x(2), \cdots, x(|x|)$ *of multidimensional samples* $x(i) \in S \cup \{\Lambda\}$*, where* $|x|$ *is the length of* x *and* Λ *is the null sample element. In general* $S \subset \mathbb{R}^d$*, with* $d \in \mathbb{N}^+$*, but it*

could be also a set of finite discrete symbols for instance. In the remaining part of the article we will consider that $S \subset \mathbb{R}^d$

2. *Let* x_n*, with* $0 \leq n < |x|$*, be the truncated time series obtained from* x *up to sample* n *(*$x_n = x(0)x(1), \cdots, x(n)$*).*

3. $^r x$ *will denote the time series obtained when reverting time series* x*, basically* $^r x(i) = r(|x| - i)$*, for all* $i \in \{0, 1, \cdots, |x| - 1\}$

4. *Finally, by convention, if* $k < 0$ *or* $k > |x|$ *we consider that* $x(k) = \Lambda$ *(padding).*

Let $\mathcal{U}$ *be the set of time series and* $\Omega \in \mathcal{U}$ *be the null time series (time series of length* 0*). We will denote* $\mathcal{U}_n = \{x \in \mathcal{U} \ \texttt{s.t.} \ \ |x| \leq n\}$ *the set of time series whose size is lower or equal to* n*.*

Definition 2 *Alignment map: Let* π *be an ordered alignment map between two finite non empty sequences of successive integers of length* n *and* m *respectively. Basically* π *is a finite sequence of pairs of integers* $\pi(l) = (i_l, j_l)$ *for* $l \in \{0, ..., |\pi| - 1\}$*, satisfying the following conditions*

1. $0 \leq i_l, \forall l \in 0, .., |\pi| - 1$

2. $i_l \leq i_{l-1} + 1, \forall l \in 1, .., |\pi| - 1$

3. $j_l \leq j_{l-1} + 1, \forall l \in 1, .., |\pi| - 1$

4. $i_{l-1} < i_l$ *or* $j_{l-1} < j_l, \forall l \in \{1, .., |\pi| - 1\}$

$\pi_1(l) = i_l$ *and* $\pi_2(l) = j_l$ *are the two coordinate access functions for the* l^{th} *pair of mapped integers so that* $\pi(l) = (\pi_1(l), \pi_2(l))$*.*

We denote by $\tilde{\pi}$ *the alignment map symmetric to* π*, namely,* $\tilde{\pi}_1 = \pi_2$ *and* $\tilde{\pi}_2 = \pi_1$*.*

For all $n \geq 1$ *and* $m \geq 1$*, let* $\Pi_{n,m}$ *be the set of alignment maps* π *such that the two sets of mapped integers* π *are* $\{1 \cdots n\} \times \{1 \cdots m\}$*.*

Fig.3 gives an example of an alignment map that corresponds to an alignment path traversing the $n \times m$ grid while satisfying the conditions specified in Definition 2.

3.2 Dynamic Time Warping

Using the previous definition of an alignment map and corresponding path, the DTW measure between two time series x and y is straightforwardly defined as:

$$\delta_{dtw}(x, y) = \min_{\pi \in \Pi_{|x|,|y|}} \sum_{i=1}^{|\pi|} (x(\pi_1(i)) - y(\pi_2(i)))^2 \tag{2}$$

However, solving directly the optimization problem that defines DTW (Eq.2) is difficult since the number of available paths corresponding to a valid alignment map in a $n \times m$ grid is known to be a Delannoy's number, $D(n, m)$ Banderier and Schwer (2005). When n and m are in the same order, this number asymptotically increases as $D(n) = \frac{c\alpha^n}{\sqrt{n}}(1 + O(n^{-1}))$

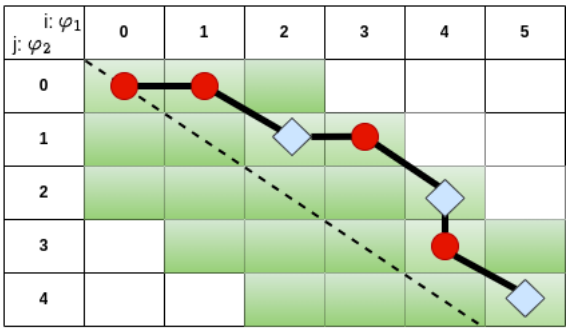

Figure 3: Example of an alignment path corresponding to the alignment map $(0,0)(0,1)(1,2)(1,3)(2,4)(3,4)(4,5)$. The white squares correspond to substitution operations and black circles to either deletion or insertion operations.

where $n \times n$ is the size of the square grid, $\alpha \approx 5,828$ and $c =\approx 0,5727$. Hence, the above DTW optimization problem consists in searching an optimal path in a set of paths whose cardinal increases exponentially with the length of the compared time series.

This is where Bellman's optimality principle Bellman (1957) and the dynamic programming paradigm come into play, allowing to derive in a recursive way the optimal alignment path with a quadratic computational time complexity Sakoe and Chiba (1971).

$$\begin{aligned}\delta_{dtw}(x(n),y(m)) &= (x(n),y(m))^2 \\ &+ Min \begin{cases} \delta_{dtw}(x(n-1),y(m)) \\ \delta_{dtw}(x(n-1),y(m-1)) \\ \delta_{dtw}(x(n),y(m-1)) \end{cases} \end{aligned} \tag{3}$$

To further reduce the time complexity, Sakoe and Chiba proposed to limit the search space to a symmetric corridor disposed around the main diagonal of the grid. The green cells of the grid presented in Fig. 3 illustrates this kind of corridor.

3.3 Kernelization of DTW

Since it has been shown Lei and Sun (2007); Marteau and Gibet (2014a) that it is not possible to derive directly definite (positive or negative) kernels from the elastic distances mentioned previously, including DTW, the kernelization of DTW has attracted some attention during the last decade. Global alignment Kernel (GAK) Cuturi et al. (2007b), Kernalized DTW (KDTW) Marteau and Gibet (2014b) and soft-DTWCuturi and Blondel (2017) are some of the most prominent approaches in this area. As our proposal for a time elastic neural network is directly derived from KDTW, we detail below how this positive definite kernel has been originally elaborated.

Definition 3 *π-embeding: For all $n > 1$ and all $\pi \in \Pi_{n,n}$, we introduce two projections for time series , basically two vectorized representations, $\varphi_{\pi_1} : \mathbb{U}_n \rightarrow U_{2n,\pi}$ and $\varphi_{\pi_2} : \mathbb{U}_n \rightarrow U_{2n,\pi}$. These projections are uniquely induced by the alignment map π. Here $U_{2n,\pi} \subset (\mathbb{R}^d \cup \{\Lambda\})^{2n}$ can be considered as a subset of times series whose lengths are at most $2n$.*

Note that the maximal length of an alignment map, as defined by Def.2, allowing to align two time series of length n is $2n$.

Then, the principle of constructing these two projections is simple. Given any alignment map $\pi \in \mathcal{M}_n$ and any time series x, we traverse π step by step from $l = 1$ to $l = \pi$, while applying the following rules:

1. *if both indexes $\pi_1(l)$ and $\pi_2(l)$ increase, then, we set $\varphi_{\pi_1}(x)(l) = x(\pi_1(l))$ and $\varphi_{\pi_2}(x)(l) = x(\pi_2(l))$,*

2. *if only index $\pi_1(l)$ increases, then we set in $\varphi_{\pi_1}(x)(l) = x(\pi_1(l))$ and $\varphi_{\pi_2}(x)(l) = \varphi_{\pi_2}(x)(l-1)$,*

3. *if only index $\pi_2(l)$ increases, then we set $\varphi_{\pi_1}(x)(l) = \varphi_{\pi_1}(x)(l-1)$ and $\varphi_{\pi_2}(x) = x(\pi_2(l))$,*

4. *when we reach the end of π, if the lengths of $\varphi_{\pi_1}(x)$ (respectively $\varphi_{\pi_2}(x)$) is shorter than $2n$, then we insert Λ into the remaining slots.*

If we consider the example given in Fig.3, for any time series $x \in U_6$ correspond two projections in $U_{12} \subset (\mathbb{R}^d \cup \{\Lambda\})^{2\times 6}$ given the alignment map depicted in Figure 3. These projections are:

$$\begin{aligned} \varphi_{\pi_1}(x) &= [x(0), x(1), x(2), x(3), x(4), x(4), x(6), \Lambda, \Lambda, \Lambda, \Lambda, \Lambda] \\ \varphi_{\pi_2}(x) &= [x(0), x(0), x(1), x(1), x(2), x(3), x(4), \Lambda, \Lambda, \Lambda, \Lambda, \Lambda] \end{aligned} \tag{4}$$

Finally, for any $x \in \mathbb{U}_n$ and $\pi \in \Pi_{n,n}$, we denote $\mathcal{P}_\pi(x) = \{\varphi_{\pi_1}(x), \varphi_{\pi_2}(x)\}$ the set of projections (or parts) for time series x induced by π. Note that all these projections are sequences whose lengths are $2n$.

Proposition 4 *If kernel $k(.,.)$ is positive definite on $\mathbb{R}^d \cup \{\Lambda\}$ then $\forall n \geq 1$ and $\forall \pi \in \Pi_{n,n}$, then*

$$k_\pi(a, b) = \prod_{l=1}^{2n} k(a(l), b(l)) \tag{5}$$

is a p.d. kernel on $(\mathbb{U}_{2n})$

Proposition 5 *If kernel $k(.,.)$ is positive definite on $\mathbb{R}^d \cup \{\Lambda\}$ then $\forall n \geq 1$ and $\forall \pi \in \Pi_{n,n}$, then*

$$K_\pi(x, y) = \sum_{\varphi(x) \in \mathcal{P}_\pi(x)} \sum_{\varphi(y) \in \mathcal{P}_\pi(y)} \prod_{l=1}^{2n} k(\varphi(x)(l), \varphi(y)(l)) \tag{6}$$

is a p.d. kernel on $(\mathbb{U}_n)$.

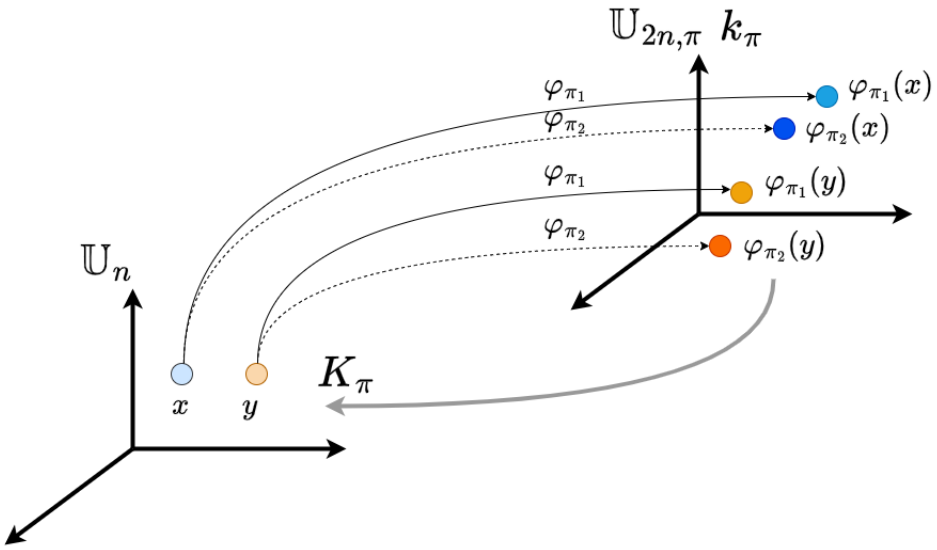

Figure 4: Projections generated by the alignment path π. To each time series in $\mathbb{U}_n$ correspond two series (embeddings) in the space $\mathbb{U}_{2n}$. The existence of a kernel in the embedding space allows for the construction of an elastic kernel back into the time series space $\mathbb{U}_n$.

Proof of proposition 5 is a direct consequence of the Haussler's *R-convolution* kernel theorem Haussler (1999). Indeed, since $k(x, y)$ is a p.d. kernel on $(\mathbb{R}^d \cup \{\Lambda\})$, and, considering the sets of parts $\mathcal{P}_\pi(x)$ and $\mathcal{P}_\pi(y)$ associated respectively to the sequences x and y, the conditions for the Haussler's *R-convolution* are satisfied.

Note that $K_\pi(x, y)$ simply rewrites as

$$\begin{aligned} K_\pi(x,y) = &\prod_{l=1}^{2n} k(\varphi_{\pi_1}(x)(l), \varphi_{\pi_2}(y)(l))) \\ &+ \prod_{l=1}^{2n} k(\varphi_{\pi_2}(x)(l), \varphi_{\pi_1}(y)(l))) \\ &+ \prod_{l=1}^{2n} k(\varphi_{\pi_1}(x)(l), \varphi_{\pi_1}(y)(l))) \\ &+ \prod_{l=1}^{2n} k(\varphi_{\pi_2}(x)(l), \varphi_{\pi_2}(y)(l))) \end{aligned} \tag{7}$$

In practice, $k(a, b) = \frac{1}{3}e^{-\nu(a-b)^2}$ is chosen as the local positive kernel defined on $\mathbb{R}^d \cup \{\Lambda\}$, with $\nu \in \mathbb{R}^+$ and considering for instance $(a - \Lambda)^2 = \infty$, $foralla \in \mathbb{R}^d$.

Let $\mathcal{C}_{2n} \subset \mathcal{M}_{2n}$ be a subset of paths then the following holds.

Proposition 6 *For any $n > 1 \in \mathbb{N}$, any $\pi \in \mathcal{C}_{2n} \subset \mathcal{M}_{2n}$, and any $(x, y) \in (\mathbb{U}_n)^2$, then the following kernel is positive definite on $\mathbb{U}_n$*

$$KDTW(x, y) = \sum_{\pi \in \mathcal{C}_{2n}} K_\pi(x, y) \tag{8}$$

The three previous propositions are also a consequence of the fact that the set of positive definite kernels is a closed (w.r.t. pointwise convergence) convex cone stable under addition and multiplication.

The choice for the local kernel $k(a, b) = \frac{1}{3}e^{-\nu(a-b)^2}$ in the embedding space allows for a probabilistic interpretation of $KDTW$. It provides a probability (up to a normalization factor) for the matching of samples a and b. As we multiply these local matching probabilities along the alignment path π, kernel k_π provides a distribution of probability (up to a normalization factor) on the set of alignment paths (represented by the four alignments parts listed in Eq.7). And finally, $KDTW(x, y)$ can be understand as the sum of the probability of the admissible paths that align x and y.

More details about the proofs and interpretation of $KDTW(x, y)$ can be found in Marteau and Gibet (2014b).

The following equations define a recursive implementation (thanks to the dynamic programming solution) of KDTW that can be evaluated with a quadratic complexity ($O(n^2)$) both in time and space. It includes a corridor, h, symmetric to the main diagonal of the alignment grid.

$$KDTW(x, y) = \mathcal{C}_h(x, y) + \tilde{\mathcal{C}}_h(x, y) \tag{9}$$

with the two recursive equations starting with $p = |x|$ and $q = |y|$.

$$\begin{aligned} \mathcal{C}_h(x_p, y_q) &= \frac{1}{3}e^{-\nu(x(p)-y(q))^2} \\ &\sum \begin{cases} h(p-1, q)\mathcal{C}_{e,h}(x_{p-1}, y_q) \\ h(p-1, q-1)\mathcal{C}_{e,h}(x_{p-1}, y_{q-1}) \\ h(p, q-1)\mathcal{C}_{e,h}(x_p, y_{q-1}) \end{cases} \end{aligned} \tag{10}$$

$$\begin{aligned} \tilde{\mathcal{C}}_h(x_p, y_q) &= \frac{1}{3} \\ &\sum \begin{cases} h(p-1, q)\tilde{\mathcal{C}}_h(x_{p-1}, y_q)(e^{-\nu(x(p)-y(p))^2} + e^{-\nu(x(q)-y(q))^2}) \\ \frac{1}{2}\delta_{pq}h(p-1, q-1)\tilde{\mathcal{C}}_h(x_{p-1}, y_{q-1})e^{-\nu(x(p)-y(p))^2} \\ \frac{1}{2}h(p, q-1)\tilde{\mathcal{C}}_h(x_p, y_{q-1})(e^{-\nu(x(p)-y(p))^2} + e^{-\nu(x(q)-y(q))^2}) \end{cases} \end{aligned} \tag{11}$$

By construction, the first term in Eq.9 evaluates over all the considered alignment paths the two first products listed in Eq.7, while the second term in Eq.9 evaluates over all the alignment paths the two last products listed in Eq.7.

In the two terms of the recursive equation (Eq.10 and Eq.11), the $h(p, q)$ function represents a symmetric corridor that can be defined along the main diagonal of the alignment grid.

In the second term (Eq.11) δ_{pq} is the Kronecker's symbol: $\delta(p, q) = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$

3.4 The forward and backward kernel evaluation matrices

The recursive equations of KDTW (Eq.9,10,11) allow to construct a $|x| \times |y|$ forward matrix $F(x, y)$ whose elements $F(x, y)_{i,j}$ are nothing but the evaluation of the kernel on the prefix time series x_i and y_j, namely $KDTW(x_i, y_j)$, i.e. the sum of the probabilities of all the alignment paths that align x and y up to samples i and j respectively, or in other words, the sum of the probabilities of all the the alignment paths connecting cell $(0, 0)$ to cell (i, j) of the alignment grid.

Similarly, we construct the backward alignment matrix, $B(x, y)$, from the reverse time series ${}^r x$ and ${}^r y$. More precisely, $B(x, y)$ is defined such that $B(x, y)_{i,j} = KDTW({}^r x_i, {}^r y_j)$.

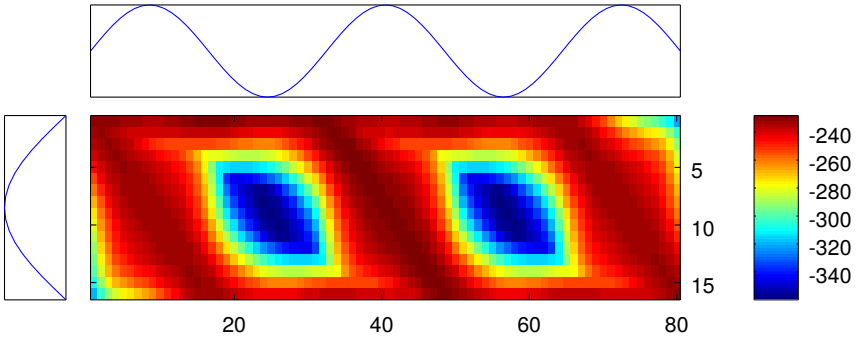

Figure 5: Forward Backward matrix (logarithmic values) for the alignment of a positive halfwave with a sinus wave. The dark red color represents high probability cells, while dark blue color represents low probability cells.

And the Forward-Backward alignment matrix $FB(x, y)$ is defined as the point-wise multiplication of the forward and backward matrices

$$FB(x, y)_{i,j} = F(x, y)_{i,j} \cdot B(x, y)_{|x|-i,|y|-j} \tag{12}$$

Hence, cell (i, j) of the FB matrix evaluate (up to a constant) the sum of the probabilities of all the existing alignment paths between x and y that traverse cell (i, j).

The FB matrix and its two components F and B provide interpretive information about the alignment process. As an example, Fig. 5 presents the Forward-Backward matrix corresponding to the alignment of a positive half-wave with a sine wave. The three areas of likely alignment paths are clearly identified in dark red color, while the low probability areas are encoded in dark blue color.

In section 5 the F and B matrices will be used to differentiate the KDTW kernel, whose evaluation is provided by the core cell of the teNN architecture detailed in the following subsections.

4 Time Elastic Neural Nets

As shown in Fig.2 and in Table **??** of the experimentation section, k-NN classifiers based on elastic distances or kernels are fairly good classification models if sufficient training data is

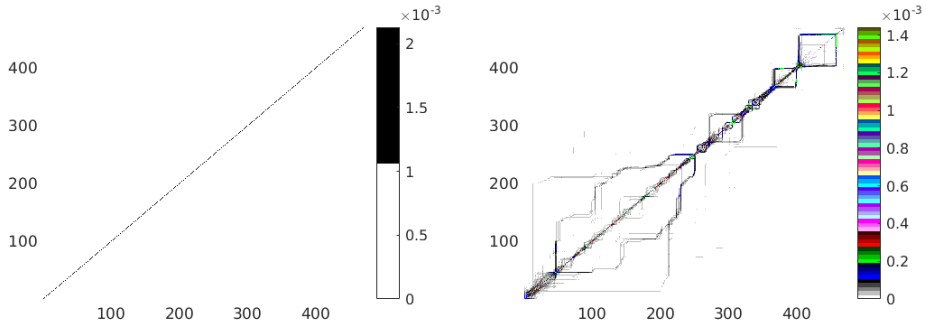

Figure 6: UCR Beef dataset: left Sakoe-Chiba 'optimal' corridor, right all the best DTW alignment paths (symmetrized), (from Soheily-Khah and Marteau (2019).

available. But, unfortunately, finding neighbors in large, multi-dimensional datasets scales very poorly (especially when distance or kernel evaluation is obtain with a quadratic complexity). In addition, decision results are difficult to explain or interpret, as the local geometry of the multi-dimensional integration space is often highly non-linear and difficult to visualize.

In this context, the motivation behind the development of the time-elastic neural network (teNN) is the improvement of k-NN classifiers based on the KDTW measure, an improvement that can be quantified in terms of efficiency, accuracy and interpretability of results.

4.1 KDTW as a network of cells

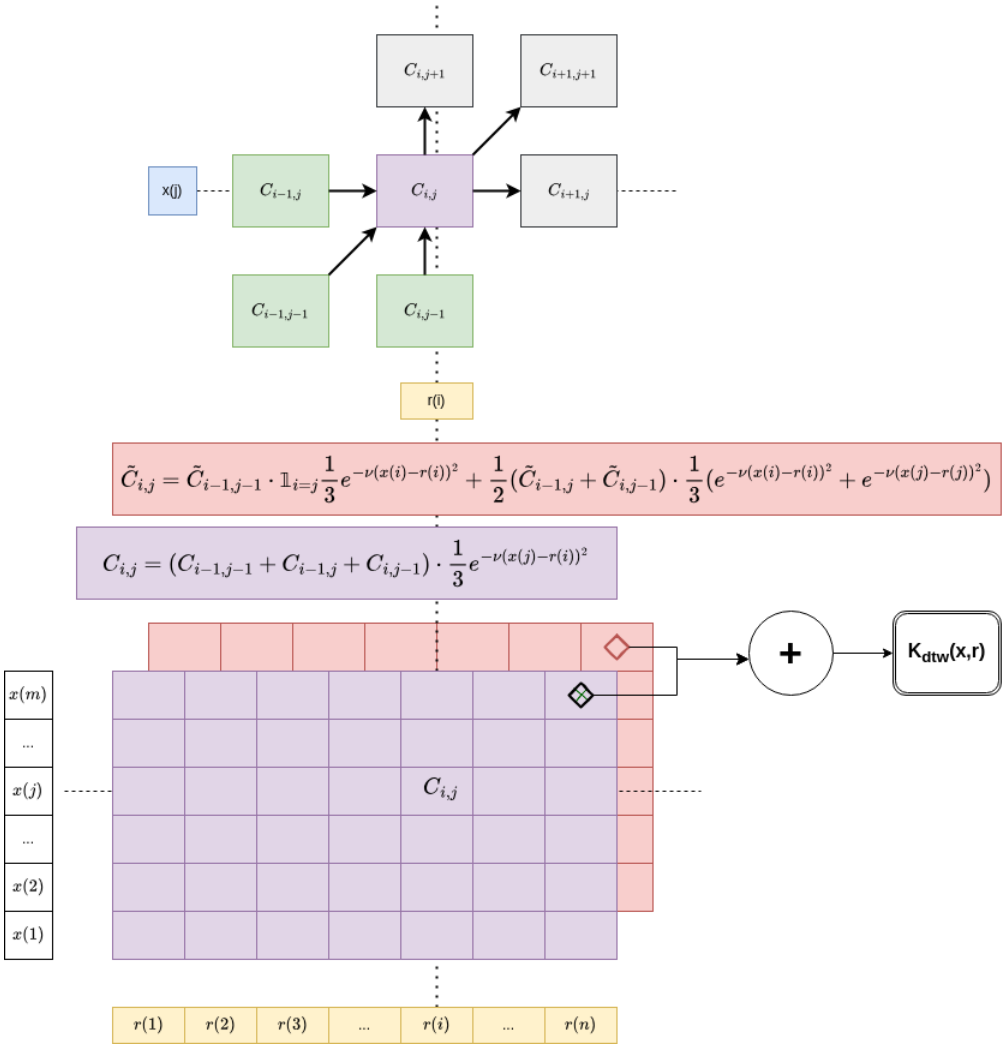

Figure 7: Network architecture dedicated to the computation of the $KDTW$ kernel.

Four main considerations guided the design of the teNN architecture.

1. The evaluation of the KDTW measure is akin to disseminating information in a network of cells. teNN is an attempt to push this analogy as far as possible.

2. To speed up k-NN classifiers, one can significantly reduce thesize of the train set by selecting only reference instances that generally characterize class boundaries, e.g. the support vectors involved in support vector machines, for example. However, another path can be followed, which consists of considering reference instances as parameters of the model that can be optimized during the training phase.

3. The ν meta-parameter which enters into the calculation of the local kernel $k(a,b) = \frac{1}{3}e^{\nu(a-b)^2}$ is constant in the definition of KDTW, although it could be adjusted along the alignment paths. By making the ν a time-dependent parameter, it may become possible to take into account a varying selectivity of the local alignment kernel capable of encoding an original form of sequential attention.

4. The corridor defined by the $h(.,.)$ function has a fixed shape (usually it is defined as a rectangle symmetrically adapted along the main diagonal of the alignment grid). As shown in Fig.6, the best alignment paths may be confined inside a potentially sparse corridor of any shape. Somehow, the teNN architecture is aiming at learning the shape of the corridor.

As it is the case for most time elastic measures, the computation of $KDTW(x,r)$ between a reference time series, R, and time series x consists in progressively evaluating the cells of two grids, $[C_{i,j}]$ (Eq.10) and $[\tilde{C}_{i,j}]$ (Eq.11), following a process that corresponds to the propagation (from left to right) of partial information into a well defined network architecture. This architecture is depicted in Fig.7. Namely each cell (i,j) of these two grids are systematically connected to three previous ones, cells $(i-1,j)$, $(i-1,j-1)$ and $(i,j-1)$, except for the initial cells $(0,j)$ and $(i,0)$, $i,j \in \{0,1,\cdots\}$. The summation of the upper right cells of the two grids gives the final result, i.e. $KDTW(x,r)$.

4.2 teNN elementary layer

To build a single teNN layer (or component) presented in Fig.8, we are making three major changes to the previous KDTW network architecture.

First, each layer is composed of a reference matrix $[R(i,k)]$ ($i \in \{0,\cdots,n-1\}$ and $k \in \{0,\cdots,d-1\}$), with $R(i,k) \in \mathbb{R}$, which is trained during the learning phase to best represent a time series subset. More precisely, once trained, R will correspond to a virtual time series that somehow maximizes the sum of pairwise similarity between itself and each of the time series of the considered subset, given the other parameters of the network introduced below.

Second, to each pair of cells ($C(i,j)$ and $\tilde{C}(i,j)$) we add an activation weight. All activation weights are gathered into an activation matrix, Ac, whose elements $Ac(i,j) \in [0;1]$, $\forall i,j = \{0,\cdots,n-1\}$, are aiming at quantifying the activation of the cells at position (i,j). If $Ac(i,j) \to 0$, then the cell is inactive and the probability for an alignment path to traverse it tends towards 0. During the training phase, this activation matrix highlights the likely alignment paths and consequently reduces the size of the alignment search space. Basically, it will extract the optimized shape of the alignment corridor.

The last main change is the attention matrix $[A_{t,R}(i,k)]$ ($i \in \{0,\cdots,n-1\}$ and $k \in \{0,\cdots,d-1\}$). This matrices is attached to the reference R (hence the index R in $A_{t,R}$). The elements of this matrix are positive or null and correspond to the parameter ν that appears in the local alignment kernel used in KDTW.

$$\kappa(r(i),x(j)) = e^{-\sum_{k=1}^{d} A_{t,R}(r(i)-x(j))^2} \tag{13}$$

When $A_{t,R}(i,k) = 0$, then the local differences at time stamp i and dimension k, $(r(i,k) - x(j,k))^2$, no longer play any role. Conversely, when $A_{t,R}(i,k)$ takes high values, then the

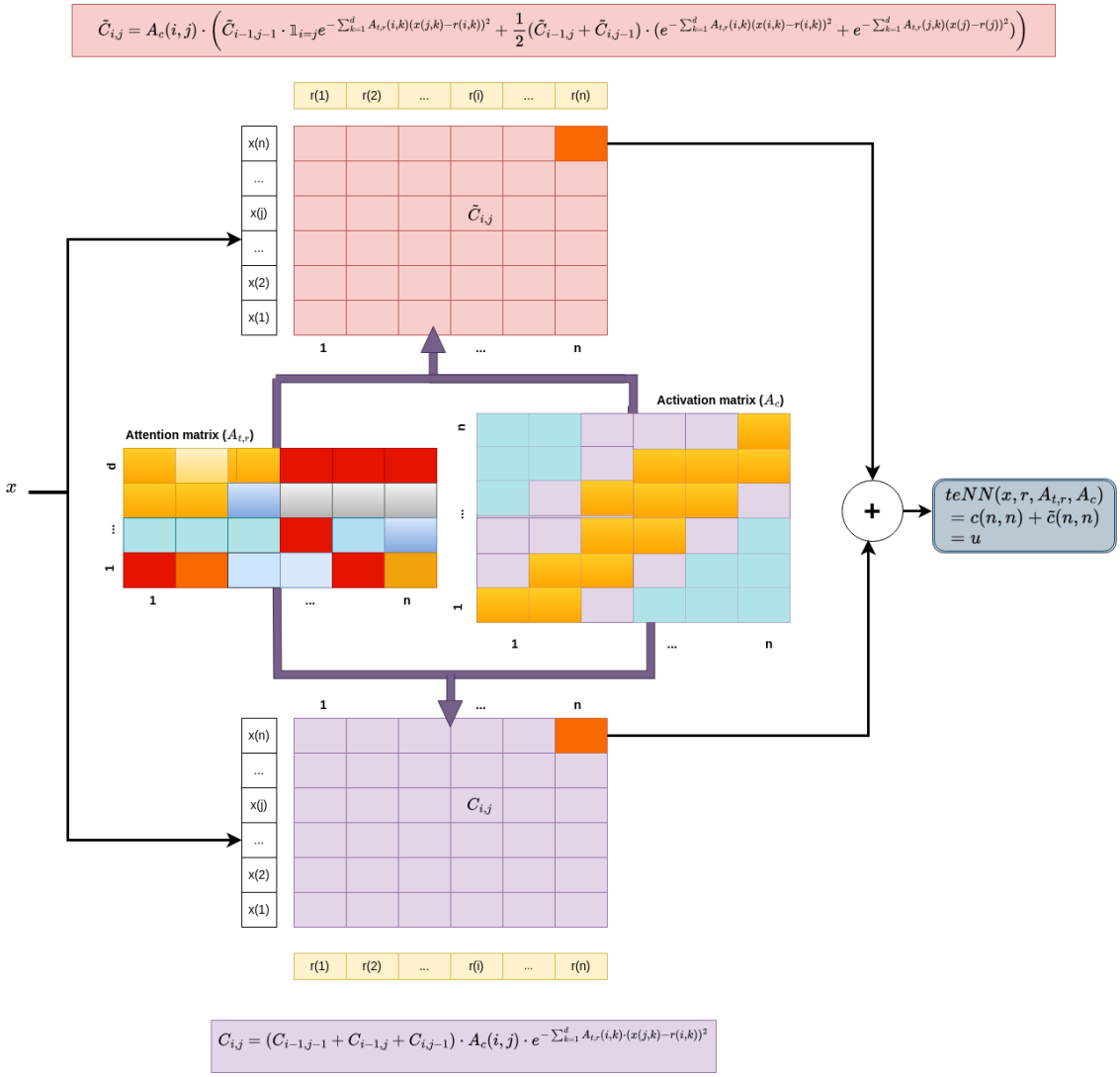

Figure 8: teNN network layer.

local alignment kernel becomes very selective at time stamp i and dimension k. Consequently, visualizing the contents of these matrices enables us to identify the spatio-temporal locations of highly selective patterns, i.e. the area of the reference time series where the network is particularly attentive, and the "don't care" areas where it is mostly inattentive.

In addition, although KDTW allows the management of time series of variable length, we adopt for sake of simplification an architecture able to process time series of length at most n. If time series shorter than n are involved, a padding with Λ samples is used.

4.2.1 Managing several references per category

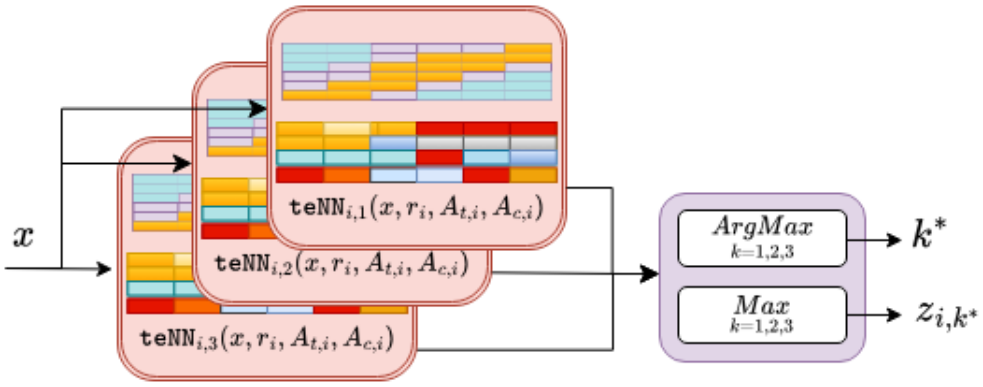

Figure 9: When several references are used to represent a category, the layer whose output is maximum (given an input x) is selected.

For some applications, it may be necessary to deal with heterogeneous categories, that is to say containing separated clusters of dissimilar time series. In such situation, the proposed architecture can manage several references in parallel for these heterogeneous categories as depicted in Fig. 9. When an input x is submitted to a set of elementary layers attached to the same category, the best layer is selected, that is to say the one providing the highest similarity measure in output. This selection is used during training (only the parameters of the best reference will be updated) and during exploitation.

4.3 teNN full network

The complete teNN architecture is depicted in Fig.10. This is a parallelization of teNN elementary layers. Each reference R_i is associated to a category y_i. We can consider either a single reference per category or several, depending on the classification task to be processed. The number of references per category is a meta parameter of the model. The expectation will be that the size of the set of R_i (the number of teNN layers) will be much smaller than the training set.

As each teNN_i layer output provides, up to a normalization factor, an estimate of the sum of the probabilities of all alignment paths between the tested time series and the reference R_i, a final normalization function acting as a softmax layer is preferred over a pure softmax layer.

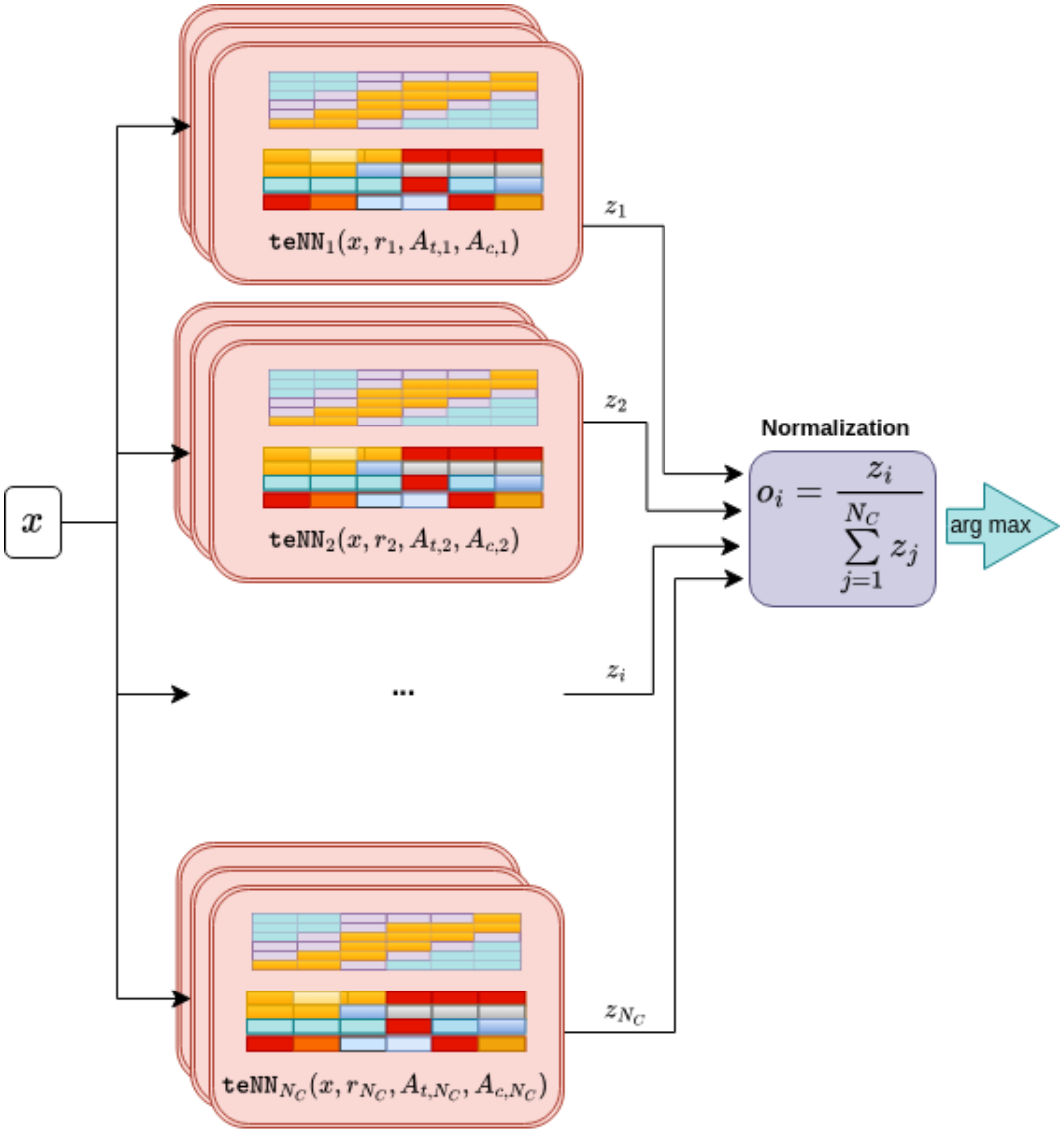

Figure 10: teNN complete network architecture.

The latter could produce numerical instability due to the exponentiation of non-normalized KDTW values.

The proposed normalizing function, f, is simply with $z^t = [z_1, z_2, \cdots, z_C]$:

$$o_i = f(z)_i = \frac{z_i}{\sum_{j=1}^{C} z_j} \tag{14}$$

5 teNN differentiation

To implement a training procedure for the teNN architecture based on stochastic gradient descent, we need to differentiate the cells of this architecture according to the parameters $(r, A_c, A_{t,R})$. Below we proceed step by step to provide these required derivatives.

5.1 teNN cell derivatives

Let begin with a single teNN cell. Recall that

$$\texttt{teNN}(x, r, A_{t,R}, A_c) = C_{n,n} + \tilde{C}_{n,n} = u \tag{15}$$

Let $k(i,j) = \frac{1}{3} \cdot A_c(i,j) \cdot e^{-\sum_{k=0}^{d-1} A_{t,R}(i,k)(r(i,k)-x(j,k))^2)}$ be the local kernel evaluation at cell (i,j) of the alignment grid ($i, j \in \{0, 1, \cdots, n-1\}$, $k \in \{0, 1, \cdots, d-1\}$)

5.1.1 $C_{n,n}$ term differentiation

Let us first consider the derivative of $C_{n,n}$ according to the three types of parameters that come into play in the teNN cell. By decomposing the derivative process around position (i,j) we get that

$$\frac{\partial C(n,n)}{\partial r(i,k)} = \sum_j \left(\frac{\partial C(n,n)}{\partial k(i,j)} \cdot \frac{\partial k(i,j)}{\partial r(i,k)} \right) \tag{16}$$

$$\frac{\partial C(n,n)}{\partial A_{t,R}(i,k)} = \sum_j \left(\frac{\partial C(n,n)}{\partial k(i,j)} \cdot \frac{\partial k(i,j)}{\partial A_{t,R}(i,k)} \right) \tag{17}$$

$$\frac{\partial C(n,n)}{\partial A_c(i,j)} = \frac{\partial C(n,n)}{\partial k(i,j)} \cdot \frac{\partial k(i,j)}{\partial A_c(i,j)} \tag{18}$$

using the derivatives of the local kernel we have

$$\frac{\partial k(i,j)}{\partial r(i,k)} = \frac{2}{3} \cdot A_c(i,j) \cdot A_{t,R}(i,k)(r(i,k) - x(j,k)) \cdot e^{-\sum_{k=0}^{d-1} A_{t,R}(i,k)(r(i,k)-x(j,k))^2)} \tag{19}$$

$$\frac{\partial k(i,j)}{\partial A_{t,R}(i,k)} = \frac{1}{3} \cdot A_c(i,j) \cdot (r(i,k) - x(j,k)) \cdot e^{-\sum_{k=0}^{d-1} A_{t,R}(i,k)(r(i,k)-x(j,k))^2)} \tag{20}$$

$$\frac{\partial k(i,j)}{\partial A_c(i,j)} = \frac{1}{3} \cdot e^{-\sum_{k=0}^{d-1} A_{t,R}(i,k)(r(i,k)-x(j,k))^2)} \tag{21}$$

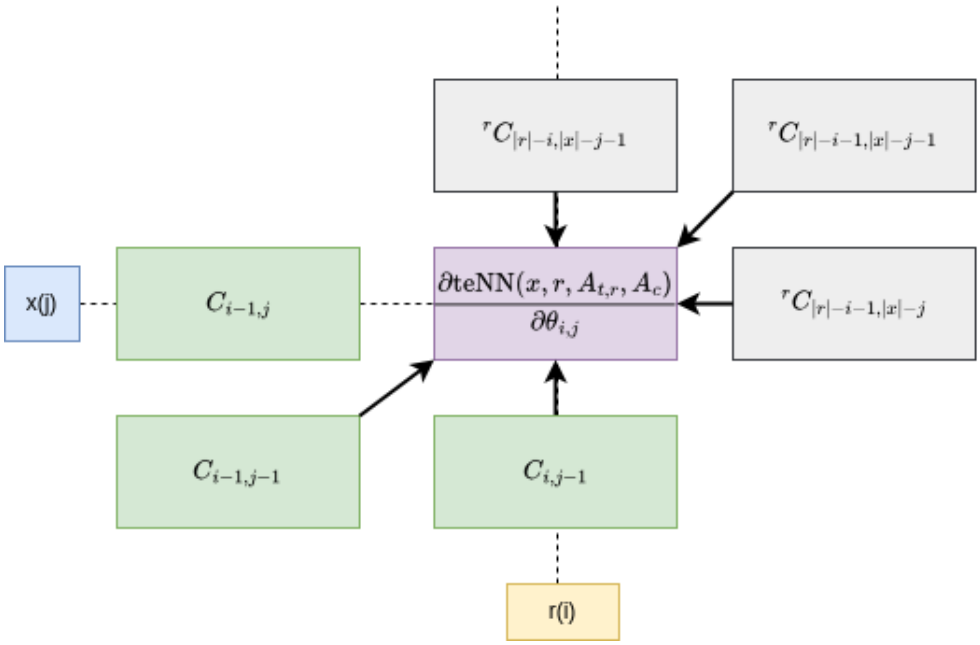

Figure 11: teNN Differentiating cell $C_{i,j}$ (similarly $\tilde{C}_{i,j}$) according to a parameter $\theta_{i,j}$ (that only depends on cell (i,j)).

For the term $\frac{\partial C_{n,n}}{\partial k(i,j)}$, we make use of the forward matrix $[C_{i,j}]$ and its backward counterpart $[^{r}C_{i,j}]$.

As $k(i,j)$ only occurs in the evaluation of cell $C_{i,j}$, Fig.11 shows how forward and backward matrices intervene into the determination of the term $\frac{\partial C_{n,n}}{\partial k(i,j)}$ by looking around cell $C_{i,j}$. Basically, we only have to consider all the paths that traverse cell Ci,j to derive $\frac{\partial C_{n,n}}{\partial k(i,j)}$.

According to Fig.11 the sum $\mathcal{S}_{i,j}$ of all the probabilities of the alignment paths that traverse cell $C_{i,j}$ is given by

$$\begin{aligned}\mathcal{S}_{i,j} =&(C_{i-1,j-1} + C_{i-1,j} + C_{i,j-1}) \cdot k(i,j) \cdot \\ &(^{r}C_{|r|-i-1,|x|-j-1} + {}^{r}C_{|r|-i-1,|x|-j} + {}^{r}C_{|r|-i,|x|-j-1})\end{aligned} \tag{22}$$

Hence, we get

$$\begin{aligned}&\frac{\partial C_{n,n}}{\partial k(i,j)} = \frac{\partial \mathcal{S}_{i,j}}{\partial k(i,j)} \\ &= (C_{i-1,j-1} + C_{i-1,j} + C_{i,j-1}) \cdot (^{r}C_{|r|-i-1,|x|-j-1} + {}^{r}C_{|r|-i-1,|x|-j} + {}^{r}C_{|r|-i,|x|-j-1})\end{aligned} \tag{23}$$

5.1.2 $\tilde{C}_{n,n}$ term differentiation

The procedure to obtain the derivatives of the term $\tilde{C}_{n,n}$ is similar to that used for the term $C_{n,n}$. We introduce first the function $f(i,j) = \frac{1}{2}(k(i,i) + k(j,j))$ and decompose the partial differentiation as follows

$$\frac{\partial \tilde{C}(n,n)}{\partial r(i,k)} = \sum_j \left(\frac{\partial \tilde{C}(n,n)}{\partial f(i,j)} \cdot \frac{\partial f(i,j)}{\partial r(i,k)} \right) \tag{24}$$

$$\frac{\partial \tilde{C}(n,n)}{\partial A_{t,R}(i,k)} = \sum_j \left(\frac{\partial \tilde{C}(n,n)}{\partial f(i,j)} \cdot \frac{\partial k(i,j)}{\partial A_{t,R}(i,k)} \right) \tag{25}$$

$$\frac{\partial \tilde{C}(n,n)}{\partial A_c(i,j)} = \frac{\partial \tilde{C}(n,n)}{\partial f(i,j)} \cdot \frac{\partial f(i,j)}{\partial A_c(i,j)} \tag{26}$$

Two distinct cases appear for the derivatives of the term $f(i,j)$, depending on whether $i = j$ or not. The formulation below compiles the two possibilities

$$\frac{\partial f(i,j)}{\partial r(i,k)} = \frac{1 + \mathbb{1}_{i=j}}{6} \cdot A_c(i,j) \cdot A_{t,R}(i,k)(r(i,k) - x(i,k)) \cdot e^{-\sum_{k=0}^{d-1} A_{t,R}(i,k)(r(i,k)-x(i,k))^2)} \tag{27}$$

$$\frac{\partial f(i,j)}{\partial A_{t,R}(i,k)} = \frac{1 + \mathbb{1}_{i=j}}{6} \cdot A_c(i,j) \cdot (r(i,k) - x(j,k)) \cdot e^{-\sum_{k=0}^{d-1} A_{t,R}(i,k)(r(i,k)-x(j,k))^2)} \tag{28}$$

$$\frac{\partial f(i,j)}{\partial A_c(i,j)} = \frac{1}{6} \cdot (e^{-\sum_{k=0}^{d-1} A_{t,R}(i,k)(r(i,k)-x(i,k))^2} + e^{-\sum_{k=0}^{d-1} A_{t,R}(j,k)(r(j,k)-x(j,k))^2}) \tag{29}$$

And finally, the exploitation of the forward and backward matrices leads to

$$\begin{aligned} \frac{\partial \tilde{C}_{n,n}}{\partial f(i,j)} =& (\mathbb{1}_{i=j}\tilde{C}_{i-1,j-1} + \tilde{C}_{i-1,j} + \tilde{C}_{i,j-1}) \cdot \\ & ({}^r\mathbb{1}_{i=j}\tilde{C}_{|r|-i-1,|x|-j-1} + {}^r\tilde{C}_{|r|-i-1,|x|-j} + {}^r\tilde{C}_{|r|-i,|x|-j-1}) \end{aligned} \tag{30}$$

5.2 The loss function for the full teNN architecture training

In the absence of knowledge on the data, the Categorical Cross-Entropy (CCE) loss seems to be an acceptable choice for training the teNN architecture, although it is known to be sensitive to imbalance data and outliers. Usually, in most neural networks architecture optimization, CCE is used in conjunction with a *softmax* output layer. For the teNN architecture however, we use it with the normalizing function o (Eq.14). In addition to the CCE loss, we add two regularization terms to force the search for parsimonious solutions. We use the L_1 norm to constrain the matrices A_c and A_t of all teNN cells to be sparse. $\lambda_1 \geq 0$ and $lambda_2 \geq 0$ are two meta parameters that weight the importance of these two regularizing terms in the final loss function.

$$\mathcal{L}(X,y) = -\sum_{l=1}^{|X|}\sum_{i=1}^{C} y_{l,i} \ln(o_{l,i}) + \lambda_1 \sum_{i=1}^{C} ||A_{t,i}||_{L_1} + \lambda_2 \sum_{ki1}^{C} ||A_{c,i}||_{L_1} \tag{31}$$

where y_i is a one-hot vector and $o_{l,i}$ the normalized output vector of teNN layer i (corresponding to the i^{th} category) when x_l is presented as input.

The training of the teNN architecture corresponds therefore to the following optimization problem

$$\begin{aligned} \Theta^* = \underset{\Theta}{Min}\ & \mathcal{L}(o, y) \\ s.t.\ \ & 0 \leq A_{c,i} \leq 1 \\ & 0 \leq A_{t,i} \\ & \texttt{for}\ \ i \in \{1, 2, \cdots, C\} \end{aligned} \tag{32}$$

Θ is the set of parameters that are optimized, namely $\{(A_{c,k}, A_{t,k}, r_k)\}$ for $k \in \{1, 2, \cdots, C\}$.

Differentiating the CEL function according to any parameter θ_i characterizing any teNN layer i can be expressed as:

$$\frac{\partial \mathcal{L}(X, y)}{\partial \theta} = \frac{\partial \mathcal{L}(o, y)}{\partial \theta} = \frac{\partial \mathcal{L}(o, y)}{\partial z_i} \frac{\partial z_i}{\partial \theta} \tag{33}$$

where o_i is the output of teNN layer i.

$\frac{\partial o_i}{\partial \theta}$ is obtained through the process described from Eq.15 to Eq.30.

And it is easy to get that

$$\frac{\partial \mathcal{L}(o_i, y)}{\partial z_i} = \frac{1}{z_i} \left(\frac{z_i}{\sum_{j=1}^{C} z_j} - y_i \right) \tag{34}$$

where y is a one hot vector.

Finally, to solve the optimization problem, which is quadratic but not convex, we adopt a classical stochastic gradient descent.

5.2.1 Algorithmic complexity of the training procedure

For the previous differentiation procedure, the evaluation of the forward and backward alignment matrices $C_{n,n}$ and $\tilde{C}_{n,n}$ requires $4n^2$ local kernel evaluations, whose complexity is in $O(d)$. As n^2 derivatives need to be evaluated for the A_c (activation) matrix, nd derivatives for the attention matrix $A_{t,R}$ and nd derivatives for the reference time series R, we get that the overall complexity requirement to differentiate a teNN cell is about $4 \cdot d \cdot n^2 + n^2 + 2n \cdot d$ macro operations, leading to a $O(d \cdot n^2)$ time complexity and a $O(n^2 + n \cdot d)$ space complexity. hence, for a full teNN architecture containing N_r cells, the computational time and space complexities are $O(N_r \cdot d \cdot n^2)$.

5.3 Training procedure for teNN

Algorithm 1 presents the implementation of a stochastic gradient descent that seeks to minimize the CCE loss on a teNN architecture, namely a set of N_c teNN cells $\{(\text{teNN}_k, y_k, f_k),\ k = 1 \cdots N_c\}$. It takes as arguments a set of labeled time series (X, y), the max number of epochs, the batch size. The main meta parameters are

1. ν_0, the bandwidth of the local kernel which is used to initialize the attention matrices A_t.
2. α_0, used to initialize the activation matrices A_c.

Algorithm 1: Stochastic gradient descent on the categorical cross entropy loss for the teNN architecture (Eq.31)

Data: X, y, L, dim $\{(\text{teNN}_k, y_k, nc_k, f_k),\ k = 1 \cdots N_c\}$, max_epoch, $batch_size$, ν_0, α_0, η, λ_1, λ_2
Result: The trained teNN
$nepoch \leftarrow 0$;
$nbatch \leftarrow max(1, floor(|X|/batch\_size))$;
$C \leftarrow set\_of\_categories(y)$;
$n \leftarrow length\_of\_time\_series(X)$;
$R_k \leftarrow InitializeWithCentroidR(y_k, X, y, nc_k)$, for $k = 0$ to N_c;
$A_{t,k} \leftarrow \nu_0$, for $k = 0$ to N_c;
$A_{c,k} \leftarrow \alpha_0$, for $k = 0$ to N_c;
while $nepoch < max\_epoch$ **do**
  $list\_of\_batch \leftarrow Random\_split(X, y, nbatch)$;
  for $nc = 0$ *to* $nbatch$ **do**
    $X_b, y_b \leftarrow list\_of\_batch[nc]$;
    $Gr[N_c, L, dim] \leftarrow 0$; $GA_c[N_c, n, n] \leftarrow 0$; $GA_t[N_c, L, dim] \leftarrow 0$;;
    for $l = 0$ *to* $|X_b|$ **do**
      $x \leftarrow X_b(l)$;
      $sumZ \leftarrow 0$;
      for $i = 0$ *to* $|C|$ **do**
        $i^* \leftarrow argmax_k\ \{\text{teNN}_k(x, r_k, A_{t,k}, A_{c,k})$ s.t. $y_k = C_i\}$;
        $z_{i^*} \leftarrow \text{teNN}_{i^*}(x, r_{i^*}, A_{t,r_{i^*}}, A_{c,i^*})$;
        $sumZ \leftarrow sumZ + z_{i^*}$;
        $\nabla r_{i^*}, \nabla A_{t,i^*}, \nabla A_{c,i^*} \leftarrow Grads(z_{i^*})$;
      end
      for $i = 0$ *to* $|C|$ **do**
        $o_{i^*} \leftarrow \frac{z_{i^*}}{sumZ}$;
        if $y_{i^*} = y_b(l)$ **then**
          loss $\leftarrow$ loss $-np.log(o_{i^*})$;
          $Gr(i^*) \leftarrow Gr(i^*) - \nabla r_{i^*} * (1/sumZ - 1/z_{i^*})$;
          $GA_t(i^*) \leftarrow GA_t(i^*) - \nabla A_{t,i^*} * (1/sumZ - 1/z_{i^*})$;
          $GA_c(i^*) \leftarrow GA_c(i^*) - \nabla A_{c,i^*} * (1/sumZ - 1/z_{i^*}$;
        end
        else
          $Gr(i^*) \leftarrow Gr(i^*) - \nabla r_{i^*} * 1/sumZ$;
          $GA_t(i^*) \leftarrow GA_t(i^*) - \nabla A_{t,i^*} * 1/sumZ$;
          $GA_c(i^*) \leftarrow GA_c(i^*) - \nabla A_{c,i^*} * 1/sumZ$;
        end
      end
    end
    for $k = 0$ *to* N_c **do**
      $R_k \leftarrow r_k + \eta Gr(k)/(||Gr(k)|| + \epsilon)/f_k$;
      $A_{t,k} \leftarrow A_{t,k} + \eta GA_t(k)/(||GA_t(k)|| + \epsilon)/f_k$;
      $A_{c,k} \leftarrow A_{c,k} + \eta GA_c(k)/(||GA_c(k)|| + \epsilon)/f_k$;
    end
  end
  if *No Progress* **then**
    `/* No accuracy gain on train data or loss reduction over a given number of epochs */`
    $\eta \leftarrow \eta/1.05$;
  end
  $nepoch \leftarrow nepoch + 1$;
end

Algorithm 2: Averaging a set of time series by means of a gradient descent on the KDTW kernel.

Data: X, L, dim, ν_0, max_epoch
Result: An estimate of the centroid of X
$nepoch \leftarrow 0$;
$M \leftarrow medoid_of(X)$;
while $nepoch < max\_epoch$ **do**
  $Gr[L, dim] \leftarrow 0$;;
  for $x \in X$ **do**
    $G_r \leftarrow G_r + \nabla KDTW(x, M, \nu_0)$ /* see Eq.16 and Eq.24 */
  end
  $M \leftarrow M + \eta G_r(k)/(||G_r(k)||)$;
  $nepoch \leftarrow nepoch + 1$;
end

3. n_r the number of references used to represent each category ($N_r = n_r \dot{N}_c$).

4. the relaxation coefficient η

5. the λ_1 which weights the sparsity penalty of the activation matrices (A_c).

6. the parameter λ_2 which weights the sparsity penalty of the attention matrices (A_t).

The initial references $\{r_u\}_{u=1,\cdots,N_r}$ are determined using algorithm 2 which average a set of time series thanks to a gradient descent on the KDTW kernel that exploits Eq.16 and Eq.24. If several references per category are required, spectral clustering is applied first with the required number of clusters, then the algorithm 2 is run on each discovered subset of time series.

6 Validation

For validating the teNN architecture and its training algorithm, we provide hereinafter some experiment carried out on three datasets, two univariate (BME, ECG200)and one multivariate (ERing) datasets from the Time Series Classification (TSC) website [1].

The meta parameters used for these experiments are given in the following table:

Table 1: Selected values for the teNN meta parameters

Parameter	value	comment
λ_t	$1e-3$	sparsity penalty for the attention matrices
λ_a	$1e-3$	sparsity penalty for the activation matrices
η	$1e-1$	relaxation coefficient
batch_size	64	
ν_0	$1e-3$	initialization of the attention matrices A_t
α_0	1.0	initialization of the activation matrices A_c
n_r	1	the number of reference time series per category.

1. https://timeseriesclassification.com/d

The following subsections present for each of these three data sets the contents of matrices A_t, A_c and R, once the learning procedure has converged. We then evaluate the degree of parsimony of the matrices A_t and A_c. We finally carry out an ablation study to estimate the impact of each of these parameter matrices on the accuracy metric.

6.1 The BME dataset

BME (Begin-Middle-End) is a synthetic univariate dataset provided by Laboratoire d'informatique de Grenoble(LIG), at Université Grenoble Alpes. It contains with three classes as shown in Fig.12:

1. Category 1 series (Begin), are characterized by the presence of a small positive peak arising at the initial period.

2. Category 2 series (Middle) are characterized by the absence of any peak at the beginning or are the end of the series.

3. Category 3 series (End) are characterized by the presence of a positive peak arising at the final period.

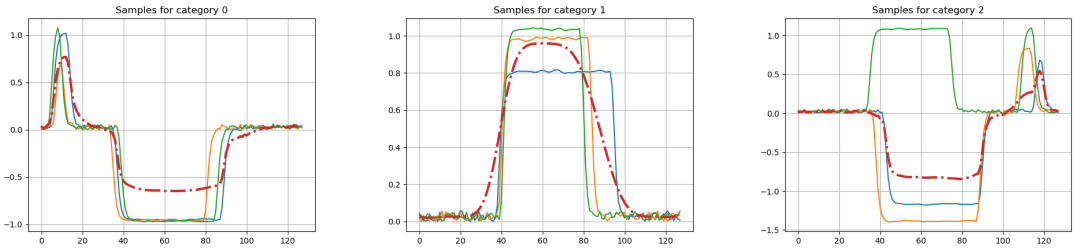

Figure 12: Samples of the BME dataset (3 samples per category are presented). The time elastic centroid used to initialize the references of the teNN cells are represented in red dotted lines.

All series, whatever their category, are constituted by a central plate. The central plates may be positive or negative. Hence, the discriminant part is the presence or absence of a positive peak, located at the beginning or at the end of the series.

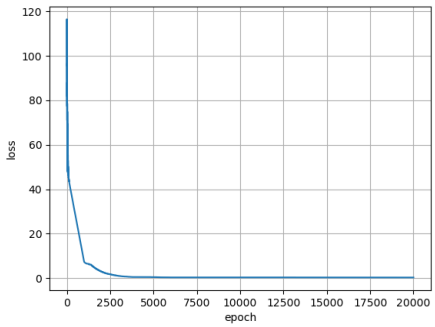

Figure 13: Loss(epoch) for the BME dataset.

Fig.13 shows the loss function as a function of the epoch for the BME dataset. The convergence appears regular and smooth although a large number of iteration (> 1000) is required to reach a minimum.

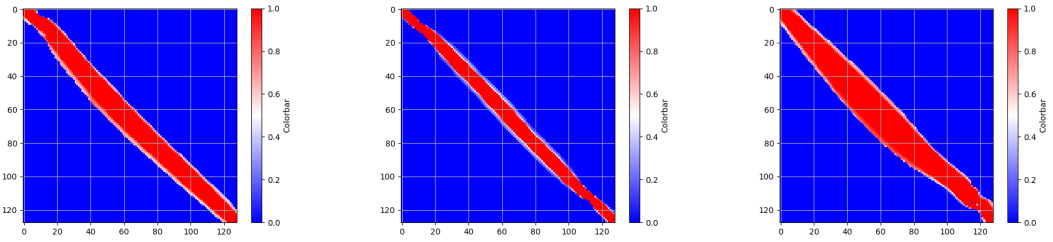

Figure 14: Activation matrices for the BME dataset. From left to right, activation for categories 1 (Begin), 2 (Middle) and 3 (End).

The activation matrices A_c, one for each of the three categories, are presented as colored encoded images in Fig.14. Red color corresponds to 1 level of activation, while blue color encodes a zero level of activation. Clearly, the teNN architecture has been able to learn corridors that limit the search spaces for the alignment paths quite sharply.

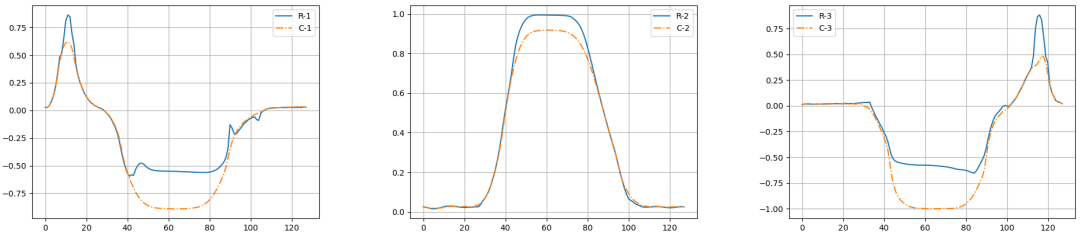

Figure 15: Reference ($\{r_i\}$) time series for the BME dataset (blue curve, plain line). The centroid of the category used for initializing the teNN cell is presented in orange dotted line. Time is the horizontal axis.

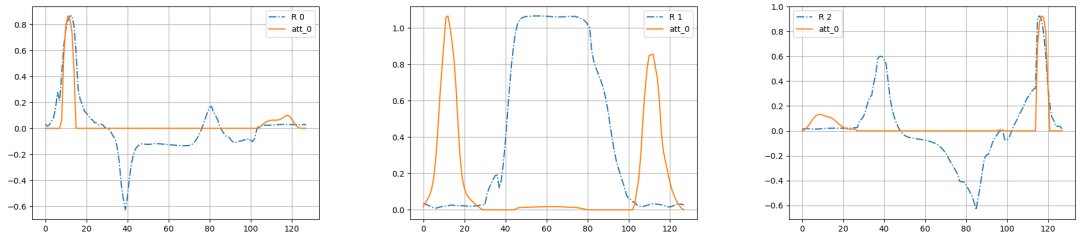

Figure 16: Attention weights for the BME dataset (orange curve, plain line). The corresponding reference vector is presented in blue dotted line. Time is the horizontal axis.

Since the BME dataset contains univariate time series, the attention matrices A_t reduce to attention vectors that are associated to each of the reference vectors R. These vectors are shown in Fig. 16.

As expected, the attention of the teNN components are focused on the discriminant part of the reference time series, basically at the beginning and the end of the time series to test the presence or absence of a discriminative peak. The centers of the time series are characterized by a quasi-null attention, meaning that the associated subsequences have no weight in the computation of the output of the teNN component.

Fig. 15 shows the initial references for each category in dotted lines. The reference obtained once the training is completed is indicated in solid blue lines. For categories 1 and 3, the central part of the final references having zero attention weight is becoming closer to 0. Here, optimizing simultaneously on the references R and on the attention matrices $A_{t,R}$ is somehow redundant: it sets a null attention on the central plate and move the reference towards zero, the average value between positive and negative plates.

For this example, we could have proposed two reference vectors for categories 1 and 2, to account for positive or negative plateaus in the central part of the series. However, as

shown in figure reffig:BME-attentions, attention weights offer a more parsimonious solution, allowing the use of a single reference per category.

6.2 The ECG200 dataset

For this dataset Olszewski et al. (2001), each univariate series is a subsequence of an electrocardiogram corresponding to a single heartbeat. The two classes gather respectively normal heart rhythm time series (category 0) and a myocardial infarction (category 1) time series (typical shapes are given in Fig.17 from Olszewski et al. (2001)). Few time series extracted randomly from the train dataset for each category are presented in Fig.18 along with the estimate centroid time series presented in red dash-dot line.

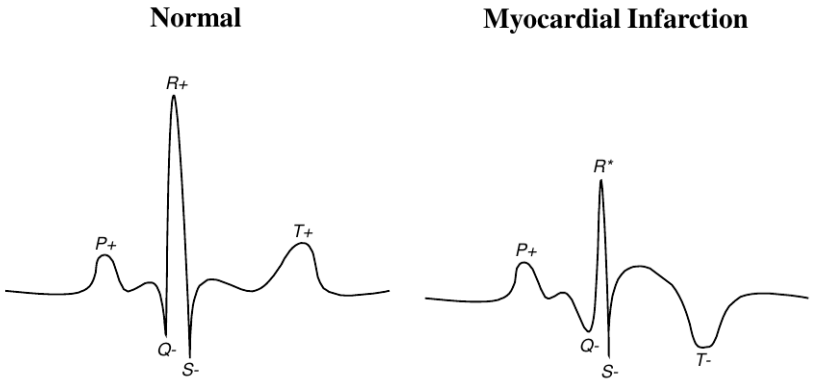

Figure 17: ECG200 categories, Normal is category 0 and Myocardial Infarction is category 1. Figures are from Olszewski et al. (2001)

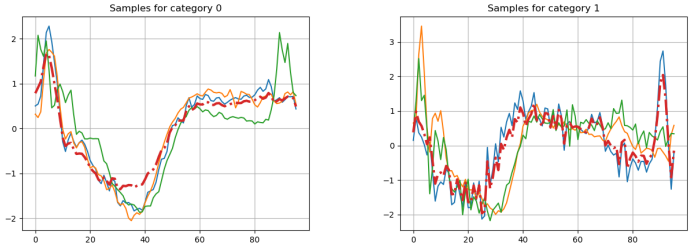

Figure 18: Samples of the ECG200 dataset (3 samples per category are presented). The time elastic centroid used to initialize the references of the teNN cells are represented in red dotted lines.

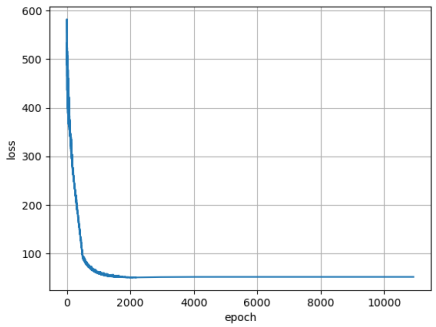

Figure 19: Loss(epoch) for the ECG200 dataset.

Fig.19 shows the loss function as a function of the epoch for the ECG200 dataset. Here again the convergence appears regular and smooth. The slope of the curve is steep, although a large number of iteration is required before reaching a minimum.

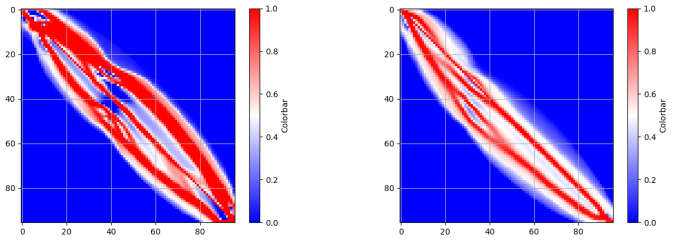

Figure 20: Activation matrices for the ECG200 dataset. From left to right, activation for category -1 and 1.

The activation matrices A_c, one for each of the three categories, are presented as colored encoded images in Fig.20. Red color corresponds to a level of maximum activation, while blue color encodes a zero level of activation and white color corresponds to in-between activation levels ($\approx$ .5). On this example, we show that the teNN architecture has been able to learn complex corridors whit 'holes' that forbid the passing of alignment paths. Here again, the search spaces for the alignment paths is reduced quite sharply.

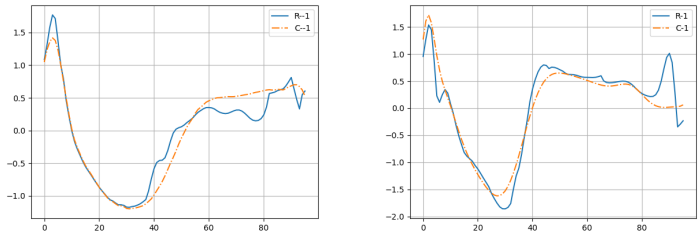

Figure 21: $\{R_i\}$) time series for the ECG200 dataset (blue curve, plain line). The centroid of the category used for initializing the teNN cell is presented in orange dotted line. Time is the horizontal axis.

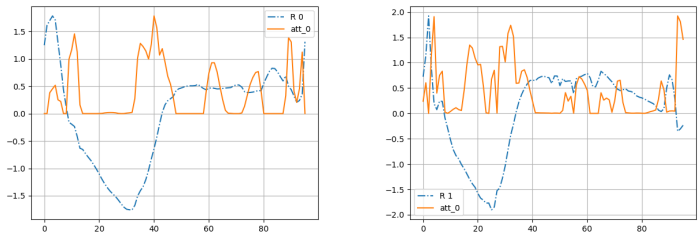

Figure 22: Attention weights for the ECG200 dataset. Left, category 'Normal' (0), right, category Abnormal (1)

The ECG200 dataset is composed with univariate time series, hence the attention matrices A_t reduce again to attention vectors that are associated to each of the reference vectors R shown in Fig.21. The attention vectors are shown in Fig. 22 in plain orange curves, while the reference vectors are presented in blue dash-dot curves. The interpretation is not obvious and required some medical expertise, but one can see that the attention vectors focus mostly on the shape of the big negative valley and on the shape of the subsequent plateau.

6.3 The ERing dataset

The ERing dataset Wilhelm et al. (2015) is composed with multivariate times series in 4 dimensions ($d = 4$) that characterize captured hand and finger gestures belonging to 6 categories. These data have been captured using a finger ring and an electromagnetic sensor. Fig.23 shows few samples for the 6 categories.

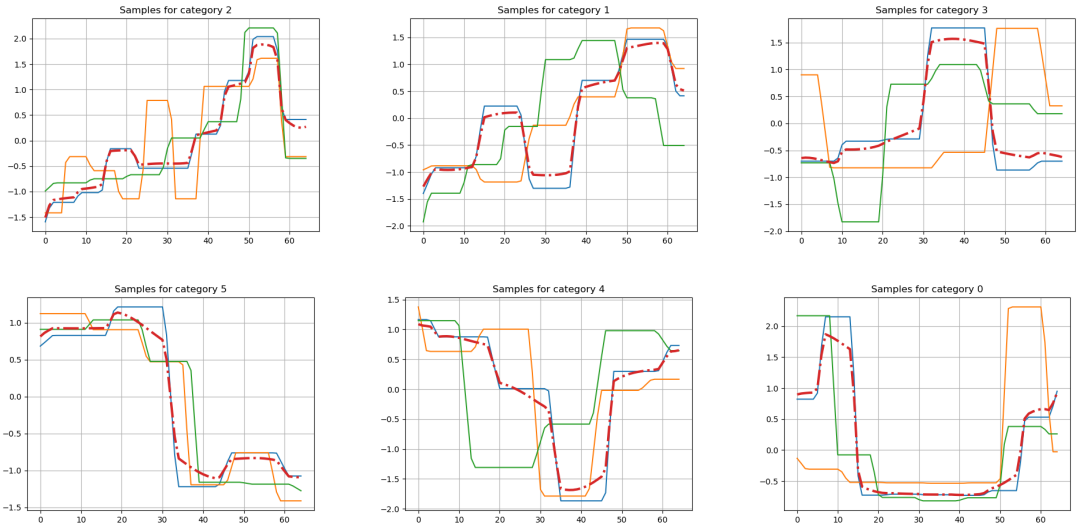

Figure 23: Samples of the ERing dataset. The time elastic centroid used to initialize the references of the teNN cells are represented in red dotted lines.

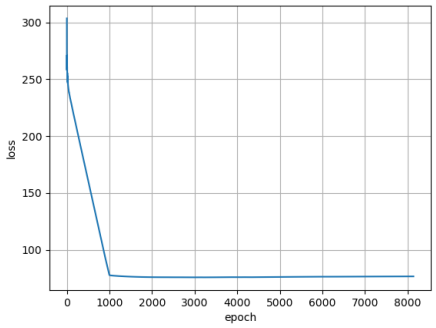

Figure 24: Loss as a function of the epoch for the ERing dataset.

Similarly to the previous examples, for the ERing multivariate data, the loss function presented in Fig.24 shows a smooth and regular convergence of the training process.

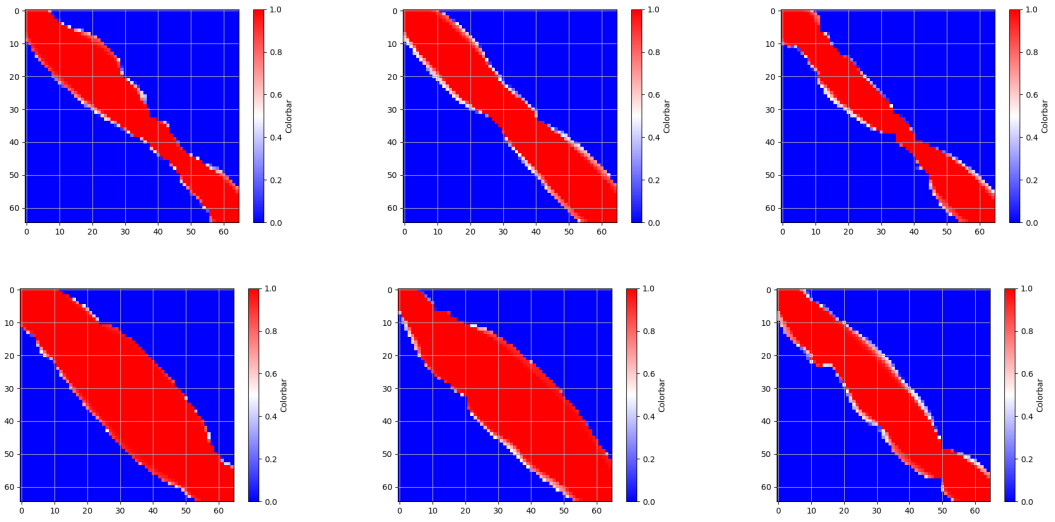

Figure 25: Activation matrix for the ERing dataset.

The activation matrices given in Fig.25 are very contrasted. Either the neurons are activated (red pixels) or deactivated (blue pixels). On this example, the teNN cells were able to significantly reduce the alignment search spaces, as shown by the narrow corridors.

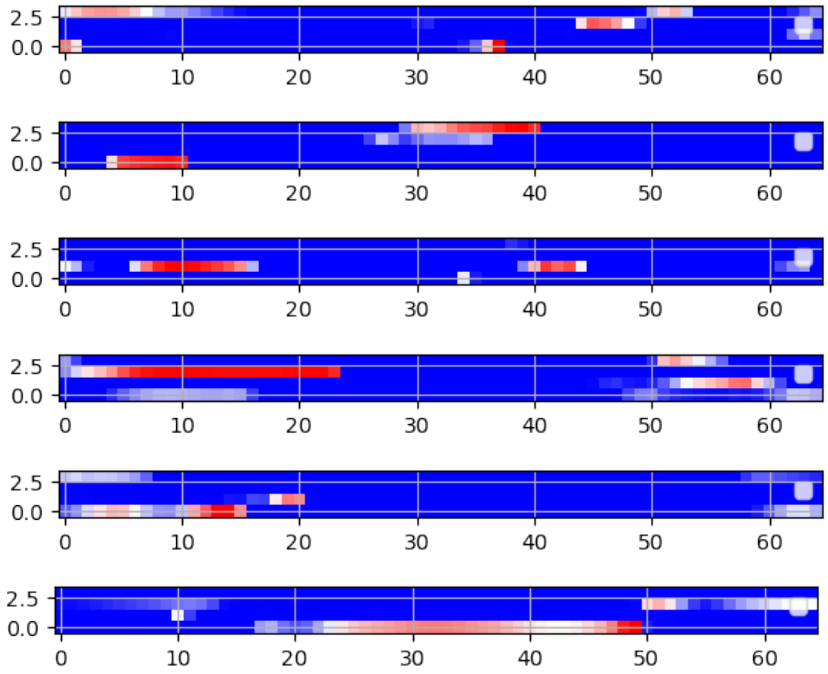

Figure 26: Attention weights for the ERing dataset. Red pixels stand for high attention, blue ones for low attention. White pixels identify in between level of attention.

As the time series are multivariate, the attention matrices do not reduce to vectors. They are shown in Fig.26. From this example, we see that, after training, the teNN architecture was able to focus its attention on a spatio-temporal basis, basically, at any given time, attention is generally focused on a small subset of dimensions. The attention matrix is sparse which was expected using a L1 penalty (meta parameter λ_2 in Eq.31).

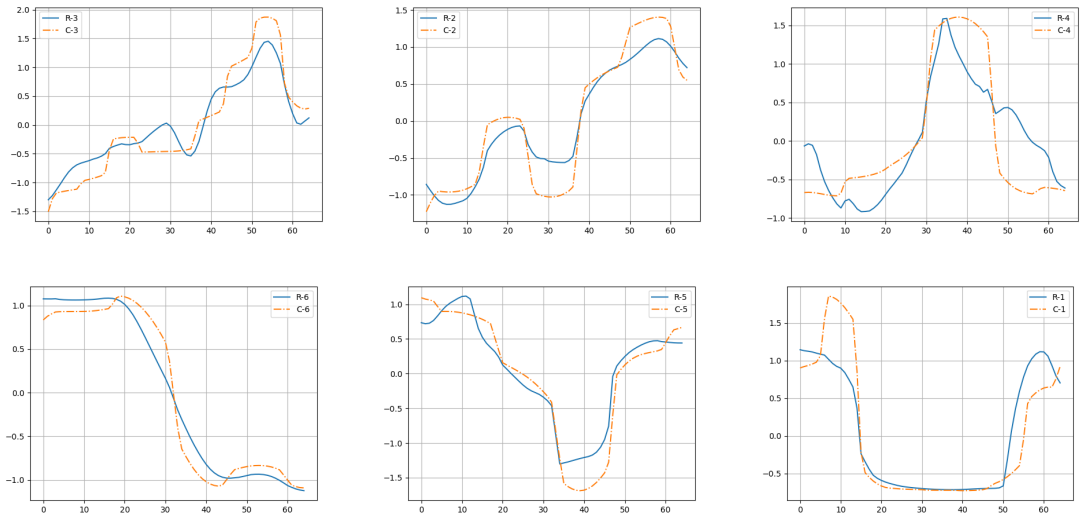

Figure 27: Reference ($\{R_i\}$) time series for the ERing dataset.

Fig.27 shows the reference time series after learning. For this example, they are staying close to the initial centroids although some details of the shapes have been erased or added. Without a precise knowledge of the motion capture process, we cannot interpret further this example.

6.4 How parsimonious is the teNN model?

Beyond limiting the number of class references, we need to estimate the sparsity of the teNN components, namely the A_c and A_t matrices. The sparsity of any matrix can be evaluated as the % of zero elements it contains. The sparsity of the matrix A_c estimates the size of the useful corridor in which the alignment paths could be confined. For the matrix A_t, it indicates the area of the reference time series R on which the network focuses its attention. Zero values correspond to unimportant local sample alignments. Although the teNN architecture is based on the KDTW global alignment kernel, A_t sparsity allows extracting local segments with high attention interconnected via "don't care" segments. Somehow, this provides teNN with local alignment capability.

Note that there is no reason for the matrix R, which represents a reference time series, to be sparse.

Table 2: Sparsity (in %) of the A_c and A_t matrices of the teNN architecture

	BME	ECG200	ERing
A_c	84.6	73.37	68.0
A_t	57.8.5	25.52	81.3

Table 2 shows the sparsity of the matrices A_c and A_t once the teNN architecture has been trained on the BME, ECG200 and ERing datasets with meta-parameters λ_t and λ_c set to the value $1e^{-2}$. For the synthetic BME dataset the sparsity of matrix A_c reaches almost 84%, while it is a bit less for the ECG200 and the ERing datasets ($\approx 70\%$). The sparsity of the A_t matrix is also significant ($\approx 60\%$) for the BME and ERing datasets.

6.5 Ablation study

Table 3: Accuracies (in %) obtained when none, one or several components of the teNN architecture are not optimized. First value corresponds to the last minimum training error configuration, while the second value corresponds to the minimum loss configuration. Best accuracies are in bold characters.

	BME	ECG200	ERing
R only	72/71.33	84/82	86.67/84.44
A_c only	74.67/74.67	78/78	86.30/86.30
A_t only	**100**/85.33	87/87	93.70/91.48
R and A_c	70.67/70.67	90/91	86.67/85.19
R and A_t	100/99.33	84/87	93.70/86.30
A_c and A_t	100/83.33	86/88	94.44/89.26
Full model	**100/100**	**93/93**	**95.56/95.56**
DTW	89.33	80.00	93.33
KDTW	98.66	83	92.59

To evaluate the importance of the three main components (A_c, A_t and R matrices) which compose the teNN architecture, we have carried out an ablation study and compared the classification accuracies obtained on the three previous datasets. Table 3 presents the accuracies obtained for the fully optimized teNN architecture (last raw), and for configurations in which one or to of the three components (A_c, A_t and R) are not optimized. When A_t is not optimized, the attention matrix is initialized by a constant ν value selected such as minimizing the training error of a one near neighbor classifier (using a leave one out procedure).

Clearly, this study shows that the joint optimization of the three components leads to the best accuracies. Apparently, the component having the most impact on the drop in accuracy is the attention matrix A_t. The R and A_c component have a significant impact on the ECG200 task, but not so much on the BME and ERing tasks. The A_c component, as

already mentioned, primarily serves to balance precision and parsimony. Not optimizing it, as expected, does not have a catastrophic impact on accuracy.

As a baseline, we added in the table 3 the accuracies of the first DTW and KDTW near neighbor classifiers. We observe that the full teNN architecture achieves better classification accuracies even though it uses a single reference time series R to represent each category.

6.6 Wrapping-up the first results

At the light of these few previous examples, we have shown that with a single reference time series per category, the teNN classifier outperforms the 1-NN classifier based on KDTW or DTW, which leads to a significant speedup. Furthermore, we have established that:

1. the stochastic gradient descent is effective to train the teNN architecture, as the loss functions decreases rapidly in general,
2. the teNN architecture is able to learn sparse attention matrices that reduce the computational cost of the teNN cellls and can be interpretable if expertise is available.
3. the neural network is able to learn part of its interconnection by deactivating unnecessary neurons. This makes it possible to learn complex alignment corridors and opens prospects for optimizing the implementation of teNN, notably by reducing its memory requirements and computational complexity.

7 Experimentation

To further assess the performance of the teNN architecture on classification tasks and to compare it with some methods of the state of the art methods, we follow the thorough empirical study that has been designed and carried out in Baldán and Benítez (2021).

7.1 Datasets

The datasets made up of multivariate time series on which the evaluation is carried out are those used in this previous study Baldán and Benítez (2021). These are 30 datasets from various fields (economy, health, biology, energy, industry, etc.), freely accessible on the UEA archive[2].

The characteristics of these datasets (size, length of series, dimension) are given in table 4 directly provided in Baldán and Benítez (2021).

Since the current implementation of the teNN architecture requires a fixed length for time series, when the series in a dataset are of variable size, we reduce them all to the length of the longest series by zero-padding.

7.2 Compared methods

In Baldán and Benítez (2021), the authors developed a feature based approach called Compexity Measures and Features for Multivariate Time series (CMFM) to improve the interpretability of classification results. More precisely, time series are represented with 41 descriptive features (in particlar curvature, linearity, Shannon entropy, skewness, trend, etc.).

2. `https://timeseriesclassification.com/`

Table 4: Datasets information from the UEA repository

Dataset	Train	Test	Length	Dims	Class
ArticularyWordRecognition	275	300	144	9	25
AtrialFibrillation	15	15	640	2	3
BasicMotions	40	40	100	6	4
CharacterTrajectories	1422	1436	60-182	3	20
Cricket	108	72	1197	6	12
DuckDuckGeese	50	50	270	1345	5
EigenWorms	128	131	17984	6	5
Epilepsy	137	138	206	3	4
EthanolConcentration	261	263	1751	3	4
ERing	30	270	65	4	6
FaceDetection	5890	3524	62	144	2
FingerMovements	316	100	50	28	2
HandMovementDirection	160	74	400	10	4
Handwriting	150	850	152	3	26
Heartbeat	204	205	405	61	2
InsectWingbeat	25000	25000	2-22	200	10
JapaneseVowels	270	370	7-29	12	9
Libras	180	180	45	2	15
LSST	2459	2466	36	6	14
MotorImagery	278	100	3000	64	2
NATOPS	180	180	51	24	6
PenDigits	7494	3498	8	2	10
PEMS-SF	267	173	144	963	7
PhonemeSpectra	3315	3353	217	11	39
RacketSports	151	152	30	6	4
SelfRegulationSCP1	268	293	896	6	2
SelfRegulationSCP2	200	180	1152	7	2
SpokenArabicDigits	6599	2199	4-93	13	10
StandWalkJump	12	15	2500	4	3
UWaveGestureLibrary	120	320	315	3	8

Several models of classification are evaluated, namely C5.0 with boosting (CMFM-C5.0B), Random Forest (CMFM-RF), Support vector machine (CMFM-SVM) and 1-Nearest Neighbors with Euclidean distance (CMFM-1NN-ED).

In addition, they included in their experimental study the following SOTA models :

- 1-Nearest Neighbors with Euclidean distance (1NN-ED).

- 1-Nearest Neighbors with DTW distance using multidimensional points (1NN-DTW-D), with or without normalization **??**.

- 1-Nearest Neighborsbased on the sum of 1 dimensional DTW distances (1NN-DTW-I), with or without normalization **??**.

- Multivariate LSTM with fully Convolutional Networks (MLSTM-FCN) **??** with the settings specified by the authors.

- Word ExtrAction for time SEries cLassification plus Multivariate Unsupervised Symbols and dErivatives (WEASEL + MUSE) [36] with the settings specified by their authors.

- Local Cascade Ensemble for Multivariate data classification (LCEM) **??**, optimized hyper-parameters for each dataset.

- Random Forest for Multivariate (RFM) algorithm, from the sklearn library, applied to the transformation proposed in the LCEM paper **??**.

- Extreme Gradient Boosting for multivariate (XGBM), Extreme Gradient Boosting algorithm, from the xgboost library,applied to the transformation proposed in the LCEM paper **??**.

All the reported classification results from these methods presented in Table 6 are from Fauvel et al. (2020) and Baldán and Benítez (2021). We use them for comparison purposes, to position the teNN architecture within the state of the art in the field.

7.3 Heuristic selection for the teNN meta parameters

The convergence of the training process depends strongly on the choice of three meta-parameters: η, the relaxation parameter, $\nu 0$, the initialization of the attention matrices and n_r the number of reference time series per category.

The heuristic we have adopted consists in selecting the highest values for η and $\nu 0$ and the lowest value for n_r so that the learning procedure converges.

Table 5: Selected values for the teNN meta parameters

Parameter	value	comment
λ_t	$1e-6$	sparsity penalty for the attention matrices
λ_a	$1e-6$	sparsity penalty for the activation matrices
η	$1e-1$	relaxation coefficient
batch_size	size of dataset	
α_0	1.0	initialization of the activation matrices A_c

For the other meta parameters, we have retained the values listed in Table 5.

Table 6: Accuracy results on the UEA repository datasets: accuracy (%), average accuracy, median, average rank, and Win/Loss/Tie Ratio. Best results are in bold.

Datasets	CMFM+ C5.0B	CMFM+ RF	CMFM+ SVM	CMFM+ 1NN-ED	LCEM	XGBM	RFM	MLSTM - FCN	WEASEL + MUSE	ED-1NN	DTW-1NN-I	DTW-1NN-D	ED-1NN (norm)	DTW-1NN-I (norm)	DTW-1NN-D (norm)	teNN-lm	teNN-ml
ArticularyWordRecognition	91	99	97.7	98.3	**99.3**	99	99	98.6	**99.3**	97	98	98.7	97	98	98.7	98.3	98.3
AtrialFibrilation	20	20	26.7	13.3	**46.7**	40	33.3	20	26.7	26.7	26.7	20	26.7	26.7	22	33.3	33.3
BasicMotions	85	97.5	92.5	95	**100**	**100**	**100**	**100**	**100**	67.5	**100**	97.5	67.6	**100**	97.5	100	100
CharacterTrajectories	95.3	97.1	95.9	90.7	97.9	98.3	98.5	**99.3**	99	96.4	96.9	99	96.4	96.9	98.9	99.2	99.2
Cricket	86.1	97.2	95.8	97.2	98.6	97.2	98.6	98.6	98.6	94.4	98.6	**100**	94.4	98.6	**100**	99.2	99.2
DuckDuckGeese	42	52	44	40	37.5	40	40	**67.5**	57.5	27.5	55	60	27.5	55	60	42.0	42.0
EigenWorms	81.7	88.5	84	79.4	52.7	55	**100**	80.9	89	55	60.3	61.8	54.9	61.8	NA	0	0
Epilepsy	88.4	**100**	97.8	95.7	98.6	97.8	98.6	96.4	99.3	66.7	97.8	96.4	66.6	97.8	96.4	93.5	93.5
Ering	80.7	**93**	92.6	90.4	20	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	96.3	96.3
EthanolConcentration	35	33.5	32.7	30.4	37.2	42.2	**43.3**	27.4	31.6	29.3	30.4	32.3	29.3	30.4	32.3	31.2	24.3
FaceDetection	54	54.8	57.9	50.5	61.4	**62.9**	61.4	55.5	54.5	51.9	51.3	52.9	51.9	52.9	NA	65.8	65.8
FingerMovements	44	52	46	53	59	53	56	**61**	54	55	52	53	55	52	53	63.0	63.0
HandMovementDirection	33.8	28.4	32.4	18.9	**64.9**	54.1	50	37.8	37.8	27.9	30.6	23.1	27.8	30.6	23.1	45.91	45.9
Handwriting	16.5	28.2	18.4	24.9	28.7	26.7	26.7	54.7	53.1	37.1	50.9	**60.7**	20	31.6	28.6	40.7	40.7
Heartbeat	74.1	76.6	73.2	62	76.1	69.3	**80**	71.4	72.7	62	65.9	71.7	61.9	65.8	71.7	72.7	72.7
InsectWingbeat	NA	**64.0**	10	25.8	22.8	23.7	22.4	10.5	12.8	11.5	12.8	NA	NA	NA	NA	55.4	55.4
JapaneseVowels	82.4	87.6	76.5	72.2	97.8	96.8	97	**99.2**	97.8	92.4	95.9	94.9	92.4	95.9	94.9	97.3	97.3
Libras	83.9	86.7	83.3	82.8	77.2	76.7	78.3	**92.2**	89.4	83.3	89.4	87.2	83.3	89.4	87	86.7	86.7
LSST	63.1	**65.2**	64.8	50	**65.2**	63.3	61.2	64.6	62.8	45.6	57.5	55.1	45.6	57.5	55.1	43.4	43.4
MotorImagery	49	51	50	44	**60.0**	46	55	53	50	51	39	50	51	50	NA	58.0	58.0
NATOPS	81.7	81.7	75	73.9	91.6	90	91.1	**96.1**	88.3	85	85	88.3	85	85	88.3	93.3	93.3
PenDigits	93.3	95.1	94.9	94.4	97.7	95.1	95.1	**98.7**	96.9	97.3	93.9	97.7	97.3	93.9	97.7	96.2	96.2
PEMSF	96.5	**100**	69.4	77.5	94.2	98.3	98.3	65.3	70.5	73.4	71.1	70.5	73.4	71.1	NA	74.0	72.1
PhonemeSpectra	22.4	28.7	25	15.8	**28.8**	18.7	22.2	27.5	19	10.4	15.1	15.1	10.4	15.1	15.1	862	8.44
RacketSports	72.4	80.9	80.9	71.1	**94.1**	92.8	92.1	88.2	91.4	86.4	84.2	80.3	86.8	84.2	80.3	86.8	86.8
SelfRegulationSCP1	81.2	81.2	79.2	70.3	83.9	82.9	82.6	**86.7**	74.4	77.1	76.5	77.5	77.1	76.5	77.5	86.3	86.0
SelfRegulationSCP2	53.9	41.7	46.1	50	**55**	48.3	47.8	52.2	52.2	48.3	53.3	53.9	48.3	53.3	53.9	56.1	56.1
SpokenArabicDigits	93.9	96.9	97.5	92.6	97.3	97	96.8	**99.4**	98.2	96.7	96	96.3	96.7	95.9	96.3	98.7	98.7
StandWalkJump	26.7	33.3	20	13.3	40	33.3	**46.7**	**46.7**	33.3	20	33.3	20	20	33.3	20	40.0	40.0
UWaveGestureLibrary	64.1	77.2	73.8	75.3	89.7	89.4	90	85.7	**90.3**	88.1	86.9	**90.3**	88.1	86.8	**90.3**	86.9	86.9
Average Rank	12	8.5	11	13	**5.1**	8	6.4	6.2	7	12	10	8.9	12	11	9.4	6.2	6.4
Win/Loss/Tie Ratio	0/30/0	4/26/0	0/30/0	0/30/0	**7/23/2**	1/29/1	5/25/2	10/20/2	3/27/3	0/30/0	1/29/1	3/27/2	0/30/0	1/29/1	2/28/2	5/25/5	5/25/5

7.4 Results

Two teNN models were evaluated. The first, teNN-lm, corresponds to the last minimum learning error strategy. The second, teNN-ml, corresponds to the last minimum loss strategy. The classification accuracies obtained by these two models are shown in the last two columns of the Table 6.

The last column of the Table 6 shows the average rank achieved by each model.

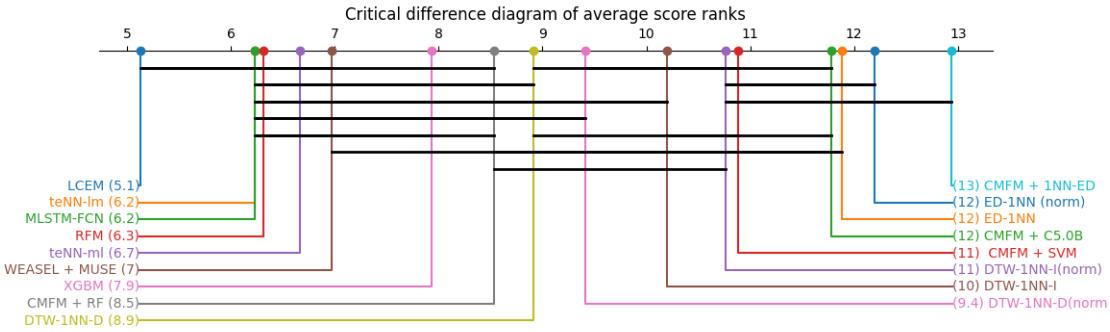

Figure 28: Critical difference diagram

As a post-hoc test, we considered the Wilcoxon signed rank test. The null hypothesis for this test is that the difference between the accuracies of the two compared models is zero. If the p-value is below the chosen significance level (0.05), we reject the null hypothesis and conclude that there is a significant difference between the accuracies of the two models. Otherwise, we consider that there is no significant difference between the two models.

Fig. 28 shows the Critical Difference Diagram (CDD) for the set of evaluated models. CDD is a way of visualizing post-hoc test statistics. Firstly, in a block design scenario, the values within each block are ranked, and the average rank across all blocks for each treatment is plotted along the x axis. A crossbar is then drawn over each group of treatments that do not show a statistically significant difference among themselves. Here, the significance level is set to 0.05 (less than 5%).

Based on this benchmark, we conclude that the teNN architecture has classification performance very similar to that of state-of-the-art models, in particular Random Forest and MLSTM-FCN models. The only model that performs slightly better, although not significantly so according to the Wilcoxon post-hoc analysis, is the LCEM model, which is a meta-model combining Random Forest and XGBoost. The gain/loss/tie ratios are very close between teNN and the best models, LCEM, CMFM+RF and MLSTM-FCN.

Furthermore, the two tested decision strategies, last minimum training error (teNN-lm) and last minimum loss (teNN-ml) lead to the similar classification accuracies, although teNN-lm achieves a slightly better average rank.

8 Conclusion

We have introduced and detail in this article an atypical neural network architecture for multivariate time series classification purpose. More precisely, we have proposed a neural network architecture that explicitly incorporates time warping capability. Behind the design of this architecture, our overall objective was threefold: firstly, we were aiming at improving the accuracy of instance based classification approaches that shows quite good performances as far as enough training data is available. Secondly we have seek to reduce the computational complexity inherent to these methods to improve their scalability. In practice, we have tried to find an acceptable balance between these first two criteria. And finally, we have seek to enhance the explainability of the decision provided by this kind of neural architecture.

The approach we have develop is rooted into the theory of kernels Schoenberg (1938), which are essentially similarity measures to which an inner product corresponds in the so-called Reproducing Kernel Hilbert Space.

The novelty of the teNN, compared to classical neural networks is the following.

1. The complete network architecture is an assembly of competitive subnetworks called teNN cells. Each teNN cell is associated with three main components: i) an abstract time series, called reference, ii) an activation matrix and iii) an attention matrix.

2. In a cell, the output of any elementary neuron is the sum of its inputs (at most three) multiplied by the local kernel which evaluates the pairwise correspondence of time series samples (thus, samples of the input time series input are compared to that of the reference time series).

3. For each neuron, the inverse of the local kernel bandwidth is a parameter (ν) that is learned during training. A high value means high local attention, while a low value means low attention, basically an area in the time series where we don't care about sample comparison. All attention parameters (inverse of bandwidth values) are grouped within a teNN cell into an attention matrix.

4. each elementary neuron is associated with an activation weight learned during training. Therefore, neurons inactivated after training can be removed to simplify the neuronal architecture. In a way, the network is able to learn a drop-out strategy, hence optimizing its own architecture. All activation weights in a teNN cell are grouped into an activation matrix.

5. Finally, the reference time series samples are also adapted during training, which provides interpretable discriminative cues in areas where attention is high.

The experiment demonstrates that the stochastic gradient descent implemented to train a teNN is quite effective. To the extent that the selection of some critical meta-parameters is correct, convergence is generally smooth and fast.

While maintaining good accuracy, we show a drastic gain in scalability by first reducing the required number of reference time series, i.e. the number of teNN cells required. Secondly, we demonstrate that, during the training process, the teNN succeeds in reducing the number of neurons required within each cell. Finally, we show that the analysis of the

activation and attention matrices as well as the reference time series after training provides relevant information to interpret and explain the classification results.

The comparative study that we have carried out and which concerns around thirty diverse and multivariate datasets shows that the teNN obtains results comparable to those of the state of the art, in particular similar to those of a network mixing LSTM and CNN architectures for example.

However, the study of TeNN architecture is still in its infancy. The impact of meta-parameters (initialization of attention matrices, number of reference time series, relaxation parameter, etc.) needs to be studied in detail.

The current implementation of the architecture itself also needs to be optimized to reduce memory requirements, in particular, inactivated neurons have to be effectively pruned to recover memory space. Very long and high dimensional time series will not fit in memory. Segmentation or multiresolution approaches could be considered to handle this issue. Finally, the implementation could be adapted to run on highly parallel computing platforms, i.e. GPUs and FPGAs.

9 Bibliography

References

Anthony Bagnall, Aaron Bostrom, James Large, and Jason Lines. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version, 2016.

Francisco J. Baldán and José M. Benítez. Multivariate times series classification through an interpretable representation. *Information Sciences*, 569:596–614, 2021. ISSN 0020-0255.

Cyril Banderier and Sylviane Schwer. Why delannoy numbers? *Journal of Statistical Planning and Inference*, 135(1):40–54, November 2005. ISSN 0378-3758. doi: 10.1016/j.jspi.2005.02.004. URL `http://dx.doi.org/10.1016/j.jspi.2005.02.004`.

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

Markus Brill, Till Fluschnik, Vincent Froese, Brijnesh J. Jain, Rolf Niedermeier, and David Schultz. Exact mean computation in dynamic time warping spaces. *CoRR*, abs/1710.08937, 2017. URL `http://arxiv.org/abs/1710.08937`.

L. Chen and R. Ng. On the marriage of lp-norm and edit distance. In *Proceedings of the 30th International Conference on Very Large Data Bases*, pages 792–801, 2004.

Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015.

M. Cuturi, J-P. Vert, O. Birkenes, and T. Matsui. A Kernel for Time Series Based on Global Alignments. In *Proceedings of ICASSP'07*, Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on, pages II–413 – II–416, Honolulu, HI, April 2007a. IEEE.

M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *IEEE ICASSP 2007*, volume 2, pages II–413–II–416, April 2007b. doi: 10.1109/ICASSP.2007.366260.

Marco Cuturi and Mathieu Blondel. Soft-DTW: a differentiable loss function for time-series. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 894–903, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Kevin Fauvel, Élisa Fromont, Véronique Masson, Philippe Faverdin, and Alexandre Termier. Local cascade ensemble for multivariate data classification. *ArXiv*, abs/2005.03645, 2020.

G.D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973. doi: 10.1109/PROC.1973.9030.

Maurice Fréchet. *Sur quelques points du calcul fonctionnel*, volume 22. Rendiconti del Circolo Matematico di Palermo, 1906.

David Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999.

Hansheng Lei and Bingyu Sun. A study on the dynamic time warping in kernel machines. In *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, pages 839–845, 2007.

Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, feb 1966. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, March 2002. ISSN 1532-4435.

P. F. Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):306–318, 2008.

Pierre-François Marteau and Sylvie Gibet. Constructing positive elastic kernels with application to time series classification. *IEEE Trans. on Neural Networks and Learning Systems*, http://arxiv.org/abs/1005.5141:1–14, 2014a.

Pierre-François Marteau. Times series averaging and denoising from a probabilistic perspective on time-elastic kernels. *International Journal of Applied Mathematics and Computer Science*, 29(2):375–392, June 2019.

Pierre-François Marteau and Sylvie Gibet. On Recursive Edit Distance Kernels with Application to Time Series Classification. *IEEE Trans. on Neural Networks and Learning Systems*, pages 1–14, June 2014b.

Matthew Middlehurst, Patrick Schäfer, and Anthony Bagnall. Bake off redux: a review and experimental evaluation of recent time series classification algorithms, 2024.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

Robert Thomas Olszewski, Roy Maxion, and Dan Siewiorek. *Generalized feature extraction for structural pattern recognition in time-series data*. PhD thesis, USA, 2001. AAI3040489.

John Paparrizos, Chunwei Liu, Aaron J. Elmore, and Michael J. Franklin. Debunking four long-standing misconceptions of time-series distance measures. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 1887–1905, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367356.

François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

H. Sakoe and S. Chiba. A dynamic programming approach to continuous speech recognition. In *Proceedings of the 7th International Congress of Acoustic*, pages 65–68, 1971.

I. J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536, nov 1938.

T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

Saeid Soheily-Khah and Pierre-François Marteau. Sparsification of the alignment path search space in dynamic time warping. *Applied Soft Computing*, 78:630 – 640, 2019. ISSN 1568-4946.

V. M. Velichko and N. G. Zagoruyko. Automatic recognition of 200 words. *International Journal of Man-Machine Studies*, 2:223–234, 1970.

Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269, April 1967.

Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974. ISSN 0004-5411. doi: 10.1145/321796.321811.

Mathias Wilhelm, Daniel Krakowczyk, Frank Trollmann, and Sahin Albayrak. ering: multiple finger gesture recognition with one ring using an electric field. In *Proceedings of the 2nd International Workshop on Sensor-Based Activity Recognition and Interaction*, iWOAR '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334549.