Neural Computing and Applications manuscript No.
(will be inserted by the editor)

An Active Adaptation Strategy for Streaming Time Series Classification based on Elastic Similarity Measures

Izaskun Oregi* · **Aritz Pérez** · **Javier Del Ser** · **Jose A. Lozano**

Received: date / Accepted: date

Abstract In streaming time series classification problems, the goal is to predict the label associated to the most recently received observations over the stream according to a set of categorized reference patterns. In online scenarios, data arise from non-stationary processes, which results in a succession of different patterns or *events*. This work presents an active adaptation strategy that allows time series classifiers to accommodate to the dynamics of streamed time series data. Specifically, our approach consists of a classifier that detects changes between events over streaming time series. For this purpose, the classifier uses features of the dynamic time warping measure computed between the streamed data and a set of reference patterns. When classifying a streaming series, the proposed pattern end detector analyzes such features to predict changes and adapt off-line time series classifiers to newly arriving events. To evaluate the performance of the proposed scheme, we employ the pattern end detection model along with dynamic time warping based nearest neighbor classifiers over a benchmark of ten time series classification problems. The obtained results present exciting insights into the detection accuracy and latency performance of the proposed strategy.

*: Corresponding author: izaskun.oregui@tecnalia.com

Izaskun Oregi, Javier Del Ser
TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Spain

Aritz Perez, Jose A. Lozano
Basque Center for Applied Mathematics (BCAM), 48009 Bilbao, Spain

Javier Del Ser, Jose A. Lozano
University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain

Keywords Time Series Classification, Streaming data, Deep Learning, Dynamic Time Warping

1 Introduction and Related Work

Time series are present in a large number of domains. In many application scenarios, time series are produced by dynamic processes, where data instances are streamed continuously at high speed, generating large volumes of samples that must be analyzed as fast as possible to comply with the limited memory and processing capabilities of current computer architectures [1, 2]. Illustrative examples of such continuously generated *streaming time series* (STS) include electricity supply data [3], human activity signals issued from wearable sensors [4] or car flow sequences [5], among many others.

It is common knowledge that successive measurements in time series are usually correlated with each other, i.e., there is dependence among observations. Such dependence has prompted the design of techniques that exploit correlations among the constituent measurements of time series to efficiently undertake diverse tasks which are defined either on streaming or conventional time series data. Such tasks include a manifold of challenging learning problems, such as forecasting [6], classification [7] and clustering [3]. Time series classification (TSC) is, arguably, among the most extensively studied problems [8, 9, 10, 11]. When formulated over STS, we define the streaming time series classification (STSC) as the problem of building a classifier that predicts the category of new incoming points based on a set of reference time series. Hence, the classifier issues a prediction every time a new point arrives over the stream.

Due to the dynamic nature of data sources, STS can also be non-stationary, in the sense that the temporal structure of the continuously flowing time series can evolve over time[1]. Accordingly, the STS can be thought as being composed of a concatenation of stationary intervals, each associated to an event that takes place for a limited period, such as a `jump` or `weight lifting` in a human activity signal (see figure 1). As a result, STS can be defined as the concatenation of stationary intervals related to different events.

When addressing TSC tasks defined over dynamic streaming time series data, it is of utmost importance that predictive models accommodate appropriately to changes between events. To this end, two main strategies can be followed depending on the mechanism utilized for adaptation [12]. On the one hand, passive (or blind) strategies use forgetting mechanisms or sliding window methods to adapt predictive models to non-stationarities of the received data [13, 14]. On the contrary, active (or informed) adaptation strategies employ auxiliary procedures to determine whether upcoming data have changed, and adjust the classifier accordingly [3, 8, 15].

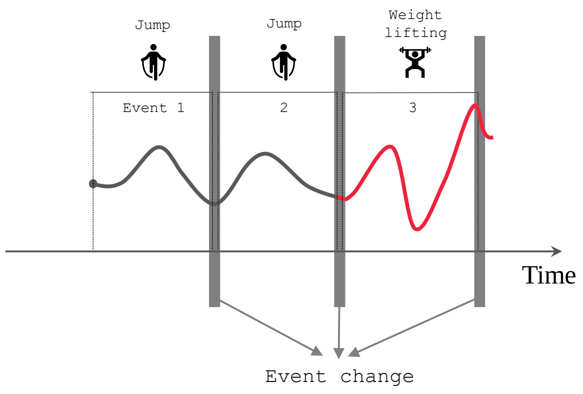


Fig. 1: Example of a streaming time series composed of two event types, `jump` and `weight lifting`. This work aims at detecting the changes between consecutive events so as to adapt the underlying STSC model.

In this context, this research work tackles the problem of developing an active adaptation strategy for streaming time series classifiers. Similarly to the change point detection problem [16], which aims at identifying abrupt changes in time series, the goal of our method is to detect changes in STS to actively adapt conventional (off-line) time series classifiers to upcoming events (see Figure 1). For this purpose, we develop the *pattern end detection* (PED) model. This procedure consists of a convolutional neural network (CNN) whose trainable parameters are learned from *streaming frames*, namely, images made up of similarity measurements computed between streamed observations and a set of reference patterns that are representative of the events in the STS. To ensure that the computation of such measurements meet the computational requirements of streaming scenarios (bounded memory and computation time), we resort to the on-line dynamic time warping (ODTW) [17, 13] – an adaptation of the conventional dynamic time warping for streaming scenarios – to determine the similarity among sequences. We assess the performance of the proposed PED when used together with DTW-based nearest neighbor classifiers to address STSC tasks defined over 10 benchmark problems from the `UCR Archive` [18]. The obtained experimental results show that an active adaptation strategy based on the proposed PED significantly outperforms other state-of-the-art distance-based classifiers used for STSC tasks. Furthermore, this work evidences the rich information embedded in streaming frames, which can be exploited for many other tasks apart from STSC.

The rest of the paper is organized as follows: Section 2 provides background information on STSC tasks and similarity measures. Section 3 details the proposed active strategy, including the concept of streaming frames and the pattern end detection model. Next, Section 4 presents the simulation setup, and Section 5 discusses the results drawn from the conducted experiments. Section 6 summarizes the main contributions of this work, and outlines future research directions. For the sake of understanding in subsequent derivations, Table 1 summarizes terms and abbreviations used throughout the paper. Additionally, some notation conventions have been established to describe the formulation: indexes are denoted with a small Latin letter; vectors with capital Latin letter; matrices with bold capital Latin letter; and datasets with a calligraphic capital Latin letter.

2 Background

In this section we pose the streaming time series classification problem (Section 2.1), as well as the fundamental concepts on the on-line version of the dynamic time warping (DTW) ESM (Section 2.2).

2.1 Problem Statement

As has been mentioned in the introduction, we assume that STS are composed of the concatenation of patterns/events belonging to different labels (see Figure

[1] Formally, a time series is said to be stationary when its statistical properties, such as mean and variance, do not depend on the timestamp at which it is measured, i.e., do not change over time.

Abbrv.	Meaning	Definition
STS	Streaming Time Series	Eqs. (1) and (2)
STSC	Streaming Time Series Classification	Section 2.1
PED	Pattern end Detection Model	Section 3.2
DTW	Dynamic Time Warping	Eq. (3)
ODTW	On-line Dynamic Time Warping	Eq. (10)
GS	Gold-standard scenario	A classification framework with a complete a priori knowledge on data
DTW-NN	DTW-based Nearest Neighbor Classifier	NN classifier with DTW as similarity measure
PED-NN	PED-based Nearest Neighbor Classifier	A DTW-NN model that uses the predicted changes of PED
ODTW-NN	ODTW-based Nearest Neighbor Classifier	NN classifier with DTW as similarity measure

Table 1: List of abbreviations.

1). Specifically, we consider that each event is represented by a time series whose data points arrive over time. Hence, the task of classifying a STS can be defined as the problem of labeling every streamed data point based on a set of known class labels. Formally, let

$$S = (s_1, \dots, s_t, \dots, s_n) \in \mathbb{R}^n \tag{1}$$

be a STS, where s_t is the t-th observation and n is the number of data points received so far. Following our assumption that S is the concatenation of labeled events, we can alternatively define a STS as

$$S = \langle S_1, \dots, S_k, \dots, S_K \rangle \in \mathbb{R}^n, \tag{2}$$

where $\langle \cdot \rangle$ denotes concatenation, $S_k = (s_1^k, \dots, s_{m_k}^k)$ is the k-th event with an associated label $l_k \in \{1, \dots, L\}$, and K is the total number of received events, so that $n = \sum_{k=1}^{K} m_k$ as per Equation (1). Based on this redefinition, the goal of the STSC problem is to carry out a classification of the stream every time a new data point s_t arrives in accordance with the class of the event to which it belongs, namely, l_k if $s_t \in S_k$.

2.2 Dynamic Time Warping for On-line Scenarios

Distance-based procedures, such as nearest neighbor (NN), are widely used to address TSC tasks [7, 19, 20]. In this context, elastic similarity measures (ESMs) are frequently used [21, 9, 22]. In general terms, ESMs are a family of similarity measures – including, among others, dynamic time warping (DTW) [23] – that shrink or stretch the time axis to find the best alignment between the time series under comparison. In this work, we use an on-line version of the DTW (ODTW [13, 17]) to develop the PED model. We begin this section by briefly introducing the naive DTW measure, with the aim of subsequently reviewing the ODTW used in the rest of the work.

2.2.1 Fundamentals of Dynamic Time Warping

Let us denote two time series as $X = (x_1, \dots, x_i, \dots, x_m)$ and $Y = (y_1, \dots, y_i, \dots, y_n)$, where $m, n \in \mathbb{N}$ represent the number of observations in each sequence. Mathematically, the DTW between X and Y is defined as

$$D_{m,n} = \underset{p \in \mathcal{P}}{\text{minimize}} \sum_{(i_q, j_q) \in p} ||x_{i_q} - y_{j_q}||_2 \tag{3}$$

subject to:

$$(i_q, j_q) \in \{1, ..., m\} \times \{1, ..., n\}, \tag{4}$$

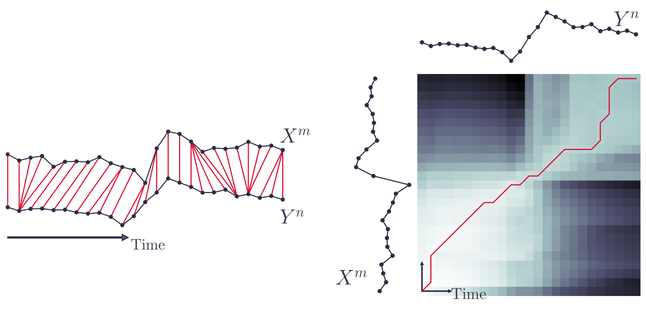
$$(i_1, j_1) = (1, 1), \tag{5}$$

$$(i_Q, j_Q) = (m, n), \tag{6}$$

$$(i_q - i_{q-1}, j_q - j_{q-1}) \in \{(0, 1), (1, 1), (1, 0)\}, \tag{7}$$

$$\forall q \in \{1, ..., Q\}, \tag{8}$$

where $||\cdot||_2$ represents the Euclidean distance, (i_q, j_q) the alignment between observations x_{i_q} and y_{j_q}, and $p = \{(i_q, j_q) \ : \ 1 \le q \le Q\} \in \mathcal{P}$ is a path – or sequence – of data point alignments that satisfies constraints (4) to (8). That is, DTW consists of finding the path p that aligns X and Y – from their beginning to their end, and without skipping a point or going backward in time – with minimal total Euclidean distance cost. Figure 2a depicts a feasible solution to this optimization problem, where each red line represents an alignment. In this context, we use p^* to denote the optimal path, that is, to refer to the sequence of alignments that results in the minimum distance, such that $D_{m,n} = \sum_{(i,j) \in p^*} ||x_i - y_j||_2$.



(a) Optimal path p^* (b) Measurement matrix $\mathbf{M}$

Fig. 2: DTW between X and Y time series, each composed of $m = n = 25$ data points.

Solutions to (3) can be efficiently obtained by dynamic programming methods [24], which leads to the following recursion:

$$D_{m,n} = ||x_m - y_n||_2 + \min \{D_{m-1,n}, D_{m-1,n-1}, D_{m,n-1}\},$$

(9)

where $D_{0,0} = 0$ and $D_{0,j} = D_{i,0} = \infty$ for $i = 1, 2, ..., m$ and $j = 1, 2, \ldots, n$. An example of the DTW computation through recurrence (9) is depicted in Figure 2b, where the red curve indicates the optimal path p^* between X^m and Y^n (black curves). Each pixel in the colored square represents the similarity $D_{i,j}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, m$, in such a way that $D_{1,1}$ and $D_{m,n}$ are located in the lower-left and upper-right corners, respectively.

2.2.2 On-line Dynamic Time Warping

In order to harness the advantages of DTW in streaming scenarios, the ODTW proposed in [13, 17] includes a forgetting mechanism that under-weights the contribution of past data observations for the similarity to evolve according to the characteristics of the streaming time series at hand. Specifically, the ODTW is defined between a stored reference time series X – with m data points – and a streaming time series Y^n – with n points received so far – as:

$$D_{m,n} = ||x_m - y_n||_2 + \min\left\{\rho D_{m-1,n-1}, D_{m-1,n}, \rho D_{m,n-1}\right\}, \tag{10}$$

where $\rho \in (0, 1]$ represents the memory parameter. It is important to note that, when the range of the memory is small (i.e., when $\rho \ll 1$), the ODTW substantially decreases the contribution of past observations of Y^n to the actual value of the similarity, thus favoring the ODTW to adjust immediately to newly arriving data. In contrast, when ρ increases, old data points acquire a higher relevance in the computation of the similarity measure, thereby making the similarity less reactive to changes in the stream. However, it provides a more realistic measure between the stored pattern and the underlying structure of the stream. Finally, when $\rho = 1$ – i.e., when ODTW takes into account all past observations – the recurrence in (10) becomes equal to Expression (9), hence, ODTW and DTW are completely equivalent to each other. The authors also used a computation/storage scheme to avoid unnecessary calculations when computing the ODTW, enabling the similarity to be updated in constant time.

It follows from recurrence (10) (also from expression (9)) that the computation of the ODTW (corr. DTW) requires intermediate results among all $X^i = (x_1, \ldots, x_i)$ for $i = 1, \ldots, n$ and $Y^j = (y_1, \ldots, y_j)$ for $j = 1, \ldots, m$ subsequences. That is, when calculating $D_{m,n}$, we compute the measurement matrix:

$$\mathbf{M} = \begin{pmatrix} D_{1,1} & D_{1,2} & \ldots & D_{1,n} \\ D_{2,1} & D_{2,2} & \ldots & D_{2,n} \\ \vdots & \vdots & & \vdots \\ D_{m,1} & D_{m,2} & \ldots & D_{m,n} \end{pmatrix} = (D_{i,j}) \in \mathbb{R}^{m \times n}, \tag{11}$$

where each entry is given by the similarity measure.

Actually, the measurement matrix contains useful information that can be analyzed to discover knowledge from data [25]. To provide evidence on the rich structure of $\mathbf{M}$, Figure 3 shows, for different values of the memory parameter, examples of the ODTW measurement matrix between a generic reference pattern (black sequence on the left) and an evolving streaming time series composed of 7 events (black sequence on the top). Images of matrix $\mathbf{M}$ in the first row of this figure correspond to the gold-standard (GS) scenario, namely, the measurement matrix with complete a priori knowledge on the event changes occurring over the stream. Thanks to this knowledge, in the GS, scenario the DTW similarity is automatically tailored to the streaming series change dynamics by initializing it at every change between consecutive events. As can be observed in this figure, the red curve indicating the alignment with the smallest ODTW between the reference pattern and each observation in the STS presents a *periodic* behavior, featuring an abrupt change within the boundary between consecutive events resulting from the initialization of the DTW. In other words, the initialization leads to coherent alignments between the reference pattern and the events held over the streaming time series.

Similarly, plots from rows 2 to 5 in Figure 3 display the measurement matrix $\mathbf{M}$ for short ($\rho = 0.0001^{1/m}$ with $m = 100$ the length of the reference time series), middle ($\rho = 0.01^{1/m}$), large ($\rho = 0.1^{1/m}$) and full ($\rho = 1$, which is equivalent to computing the naïve DTW) memory ranges [2]. In contrast to the GS scenario, in these cases stream changes are unknown, so the ODTW cannot be reinitialized. However, the reinitialization can be overridden by making the ODTW focus on recent observations, e.g., by using a small memory range through the choice of the ρ value. By doing so, $\mathbf{M}$ is more reactive to local changes, thus providing more detailed insights about the configuration of the streaming time series.

If we closely inspect the image maps of the measurement matrix, we notice that the structure of $\mathbf{M}$ delays with respect to the GS (first row) more severely as ρ increases. Accordingly, $p_{\min}$ (red curve depicting the position of the minimum ODTW measurement over time)

[2] By using these ρ values, we coerce the contribution of the data point m steps prior to the sample at hand be weighted by $0.0001, 0.1, 0.5, 1$

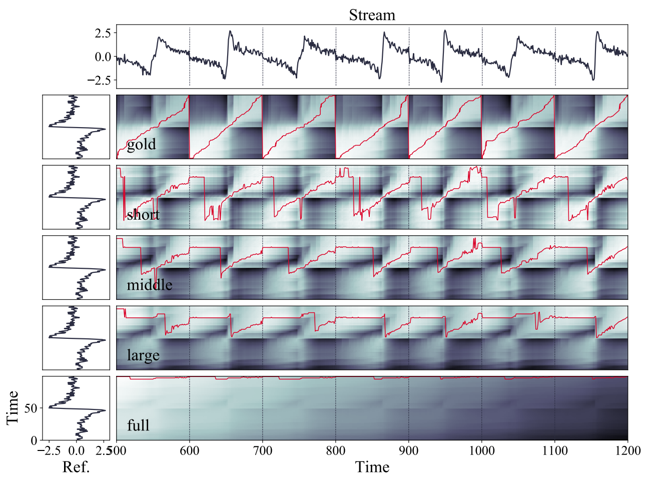


Fig. 3: Measurement matrix for gold-standard, short-, middle-, large- and full-range ($\rho = 1$) memory scenarios. In each plot, dark colors represent high $D_{i,j}$ values, whereas light colors correspond to small $D_{i,j}$ values. Vertical lines represent the boundaries (or transitions) between consecutive events (stationary intervals). In the gold-standard scenario, the red curve illustrates the optimal path p^*, computed separately within each event. In the remaining figures, the red curve depicts $p_{\min}$ (i.e., the position of the minimum ODTW measurement at each time stamp), which can be interpreted as the optimal path of the ODTW when configured with the corresponding memory parameter (short, middle, range, full).

features an increasing delay with ρ (compare red curves around pattern limits), meaning that the ODTW computed with large/full range memory needs more time to accommodate newly arriving events. For instance, in the full-range memory example ($\rho = 1$, i.e. the ODTW does not forget anything from the past), $p_{\min}$ yields from *matching* the last data points in the reference pattern with every point in the stream. Similarly, in the large-range memory example, $p_{\min}$ shows that ODTW requires at least half of the interval to forget the past and adapt to new stream events (remarkably, $p_{\min}$ is almost a diagonal in the second half of every event). By contrast, when using short- (and, to some extend, middle-) memory ranges, the adaptation of the ODTW is completed in the first half of the period, thus showing that the reference series aligns with newly appearing stream events properly.

3 The Active Adaptation Strategy

This section introduces in detail the proposed active adaptation strategy. As already mentioned, the objective of our procedure is to design a CNN classifier that aids conventional time series classifiers to cope with the STSC problem stated before in Section 2.1. To this end, we use images of the ODTW –streaming frames in Section 3.1, and a CNN binary classifier – the PED model in Section 3.2.

3.1 Streaming Frames

Rather than relying on single values of the similarity as described at the end of Section 2. Our active adaptation strategy uses sub-matrices of $\mathbf{M}$ to determine whether the sequence associated with an event has finished. Such sub-matrices, which we hereafter refer to as *streaming frames*, result from the computation of the ODTW between a reference pattern and the last ϖ observations of the streaming time series at hand. If we assume that y_n is the last streamed observation, the streaming frame is given by

$$\mathbf{M}^n = \begin{pmatrix} D_{1,n-\varpi+1} & \cdots & D_{1,n} \\ D_{2,n-\varpi+1} & \cdots & D_{2,n} \\ \vdots & \ddots & \vdots \\ D_{m,n-\varpi+1} & \cdots & D_{m,n} \end{pmatrix}, \tag{12}$$

namely, the last ϖ columns of the measurement matrix as per (11). Just as frames in a video stream, these sub-matrices are computed and stored in an on-line fashion, as new data points are received. To this end, we apply the ODTW computation/storage procedure presented in [17], under which we can update $\mathbf{M}$ incrementally along time. As mentioned previously in Section 1 (Introduction), the ODTW includes a computation/storage procedure to efficiently update the computation of the similarity by recording a small set of intermediate ODTW measurements over time. Hence, every time a new data point arrives, we do not compute the whole measurement matrix, but only the ODTW measurements of the arriving data. That is, if we receive y_{n+1} after computing $\mathbf{M}$ (see Expression (11)), we can use the elements of the last column of the measurement matrix ($D_{i,n}$ for $i = 1, \ldots, m$) to update it by just calculating

$$(D_{1,n+1}, D_{2,n+1}, \ldots, D_{m,n+1}), \tag{13}$$

which depend on the ODTW measurements concerning the received observation.

3.2 Pattern End Detection Model

This section describes the proposed PED model which, as stated previously, is used to detect label changes in streaming time series. In essence, PED consists of a binary CNN classifier learned from ODTW streaming frames. Just as in conventional CNN architectures [26], the classifier utilized in this work is composed of two main blocks: i) a set of convolutional layers, which extract features from raw data (in our case, ODTW streaming frames); and ii) a series of fully-connected neural layers, which map the extracted features to a binary label indicating whether a change among events has occurred at the timestamp corresponding to the input streaming frame.

Let $\mathcal{B} = \{(R_p, l_p)\}_{p=1}^{P}$ denote a set of reference patterns, where $R_p = (r_1^p, \ldots, r_i^p, \ldots, r_m^p)$ represents an event with an associated label $l_p \in \{1, \ldots, L\}$. We further define a streaming time series as $S = (s_1, \ldots, s_n)$, which has been previously labeled, so we know all labels occurring in the stream (see Section 2.1). In this context, the PED learning procedure is carried out by following the following steps in order:

1. **Computation of streaming frames:** we define a $(m \times n)$ image $\overline{\mathbf{M}}$ composed of P channels as:

$$\overline{\mathbf{M}} = [\mathbf{M}_1, \ldots, \mathbf{M}_p, \ldots, \mathbf{M}_P], \quad (14)$$

where $\mathbf{M}_p$ is the measurement matrix resulting from the computation of the ODTW between S and $R_p \in \mathcal{B}$ for a given value of the memory parameter ρ. Streaming frames are obtained by applying a fixed-size sliding window. That is, moving a window of w time units across the measurement matrix (sample by sample along the stream time axis), we extract a set of $n - w$ streaming frames given by:

$$\left\{\overline{\mathbf{M}}^w, \ldots, \overline{\mathbf{M}}^i, \ldots, \overline{\mathbf{M}}^n\right\}, \quad (15)$$

where $\overline{\mathbf{M}}^n = [\mathbf{M}_1^n, \ldots, \mathbf{M}_P^n]$ represents the n-th streaming frame as per (14). Red square areas overlaid in Figure 4 indicate $\overline{\mathbf{M}}^i$ and $\overline{\mathbf{M}}^n$ streaming frames.

2. **Construction of the database of labeled streaming frames $\mathcal{M}$:** the next step is to generate the streaming frame database by assigning labels to the streaming frames in $\overline{\mathbf{M}}^i$ for $i = w, \ldots, n$. To this end, each streaming frame is assigned a binary $\{0, 1\}$ label, such that frames falling within the 0 (1) category represent *no event change* (correspondingly, *event change*) in the stream. Accordingly, we generate the database of streaming frames as:

$$\mathcal{M} = \left\{\left(\overline{\mathbf{M}}^i, c_i\right)\right\}_{i=w}^{n}, \quad (16)$$

where the category c_i of the i-th streaming frame is set to 1 (i.e., *event change*) if the frame falls within a transition; and 0 (i.e., *no event change*) otherwise. Accordingly, in Figure 4, $\overline{\mathbf{M}}^i$ frame is categorized as 1 and $\overline{\mathbf{M}}^n$ as 0. Since the goal of PED is to detect limits between consecutive events, w needs to be small so that the procedure can detect event changes close to the instant where they truly occur.

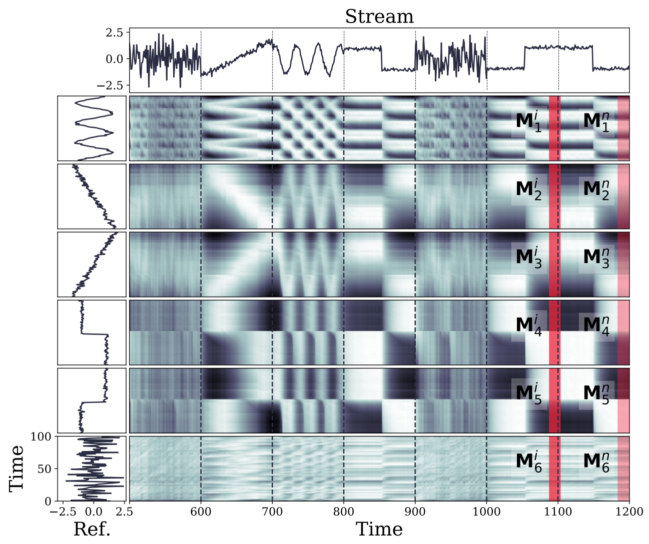


Fig. 4: Illustration of an online scenario composed of a streaming time series of $n = 1200$ observations (top) and a reference pattern database of $P = 6$ sequences (left). Event changes are highlighted with black vertical dashed lines. The set of vertical red squares depict the i-th ($\overline{\mathbf{M}}^i = [\mathbf{M}_1^i, \ldots, \mathbf{M}_6^i]$) and n-th ($\overline{\mathbf{M}}^n = [\mathbf{M}_1^n, \ldots, \mathbf{M}_6^n]$) streaming frames. The label of the former frame $\overline{\mathbf{M}}^i$ corresponds to $c_i = 1$ as the event change falls within the frame. Likewise, $\overline{\mathbf{M}}^n$ represents a `no event change` frame ($c_n = 0$). The sliding window size has been set to $w = 15$ samples.

Given a value of w, it is important to note that the sliding window method can yield a highly imbalanced database $\mathcal{M}$. In particular, if small window sizes are chosen, the number of streaming frames belonging to the *no event change* class is significantly higher than that of frames labeled as *event change*. It is widely acknowledged that, unless properly counteracted, learning from imbalanced databases can lead to imprecise models [27]. Hence, we apply a random under-sampling technique to decrease the number of *no event change* frames. Specifically, the utilized technique generates a balanced database $\mathcal{M}$ by keeping samples of the minority class, and by sampling uniformly at random as many frames from the majority class as those of the minority class.

3. **Training of the binary CNN classifier:** we train the CNN classifier over the generated database $\mathcal{M}$ of labeled streaming frames by using conventional

gradient-based backpropagation algorithms. This process yields the proposed PED that can be used to detect changes among events from new streaming frames.As we explain in detail in the following section, a variety of CNN structures have been considered for this purpose. Table 5 outlines such evaluated CNN structures.

4 Experimental Setup

In order to empirically assess the performance of the PED model and the value of streaming frames as information source, we design an experimental setup with two complementary goals in mind:

1. To evaluate the performance of the proposed PED approach when detecting event changes over STS.
2. To measure the efficiency of PED when used as a compounding part of an active adaptation strategy for STSC. That is, the objective is to analyze whether event changes predicted by the PED can be leveraged when addressing the on-line STSC task described in Section 2.1.

This section details how the above goals are approached experimentally, stressing on how performance is gauged in each case (Subsections 4.1 and 4.2), the STSC problems over which the evaluation is done (Subsection 4.3) and the model structure and learning methodology followed to train the PED model (Subsection 4.4).

4.1 Goal 1: Predictive Performance of the PED model

We first analyze the precision and the prediction delay of the PED model when detecting event changes in the streaming time series. To this end, we fit the trainable parameters of the CNN to each of the STSC databases. Then, we examine the PED model accuracy in terms of precision, recall and the distribution of the detection delay. In particular, the precision returns the proportion of correctly predicted changes among all predicted changes, i.e.,

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}, \tag{17}$$

where we consider that the PED model correctly identifies a change in the streaming time series if a streaming frame labeled as *event change* contains a real change of the test streaming time series. The recall returns the proportion of actual positives that are correctly identified, that is,

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}. \tag{18}$$

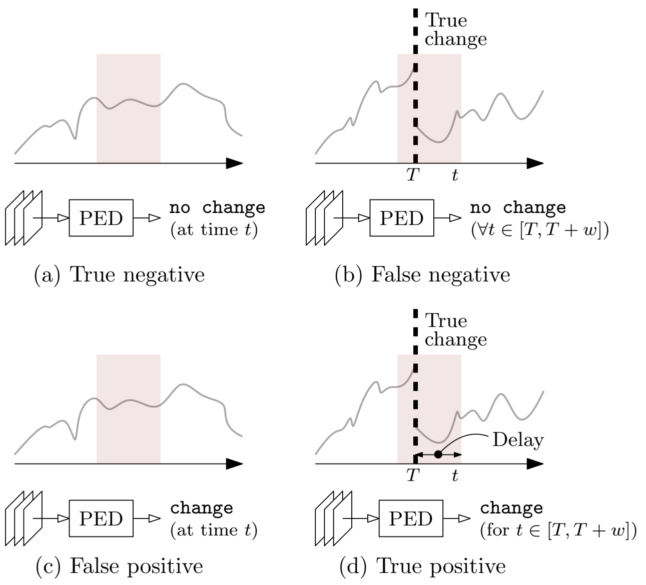


Fig. 5: Diagram showing (a) a true negative; (b) a false negative; (c) a false positive; and (d) a true positive when the PED model is used for detecting change events.

In this case, we assume a data point is received at each time stamp. Therefore, PED delivers as many predictions as observations received over the streaming time series. Figure 5 depicts a true negative, false negative, false positives and true positive in the context of this work.

On the other hand, the delay measures the distance (in time units) between the real event change and its estimation by the PED model. Hence, when a streaming frame is correctly labeled as *event change*, the delay is set to 0 if a true event change occurs at any of the time instants spanned by the streaming frame. We use the distribution of this difference to evaluate the lag when decided to trigger an active adaptation strategy as the one evaluated in the experiments (re-initialization of the DTW computation).

4.2 Goal 2: Performance of an active PED-based approach for STSC

To fulfill the second goal, we evaluate the performance of the PED by combining it with conventional time series classifiers such as DTW-based NN classifiers (DTW-NN) [7]. The active adaptation strategy embedding the PED model at its core consists of exploiting the event change detection flag issued by PED to trigger the adaptation of DTW-NN classifiers to upcoming events. That is, we use the output of PED when analyzing new streaming frames over time to i) let a DTW-NN classifier predict the label of the received stream sample (if PED model outputs *no change*); or ii) adapt the overall model to the new event by resetting the measure of similarity (if the PED issued *event change*).

In the experiments later discussed we compare the classification accuracy of the above strategy (a DTW-NN model using the predicted changes of PED, which we hereafter refer to as PED-NN) with (i) that of the GS (i.e., perfect a priori knowledge of the instants when event changes occur), and with (ii) that of a ODTW-based NN classifier (ODTW-NN) that passively adapts to event changes via a forgetting mechanism [17].

It is important to note the main difference between the PED-NN and the GS procedures: while GS is unrealistically aware of the event changes occurring in the stream, PED-NN depends on the predictions issued by PED. Thereby, GS is expected to perform better than the PED-NN approach, as it does not undergo any false positives, nor does it suffer from any delay in the detection of the event change. Consequently, GS serves as an upper bound of the performance that the PED-NN model could achieve. The closer the outcome of PED-NN is to that of GS, the more accurate the PED model can be considered to be. Therefore, GS can be thought of as being the *perfect* model because it essentially reduces to an off-line DTW-NN model performed separately for each event of the test stream.

However, in most realistic streaming scenarios it is difficult to know when an event will end. Therefore, to ensure a more realistic comparison we use the ODTW-NN classifier, so that we can assess the performance of passive and active event change adaptation strategies in STSC tasks. In this case we use `middle` ($0.1^{1/m}$) or `middle-short` ($0.01^{1/m}$) range memory values depending on the classification problem at hand (see Table 3 for a list of the specific values in use). That is, we use the same memory parameter values for both the PED model and the ODTW-NN approach for two reasons: i) to evaluate whether the proposed PED-based adaptation is less reactive than the passive adaptation strategy included in the ODTW-NN classifier; and ii) to compare both PED-NN and ODTW-NN classifiers in terms of accuracy. Moreover, an accurate PED model implies that the streaming time series can be divided into meaningful sub-sequences with critical event information, which can be used for other tasks apart from classification. Section 6 will focus on future research lines in this direction.

4.3 Generation of STSC Problems

Our STSC problems are built on 10 benchmark time series databases extracted from the `UCR Archive` [18] and the time series generation procedures provided in the supplementary material of [28]. The main features of these databases are summarized in Table 2. From left to right, columns denote the name, length of the patterns, total number of classes L, and the number of elements in each database. Among them, we have identified those synthetically generated databases with a symbol "$\Diamond$" which, as stated before, have been produced by following the procedure described in [28]. Parameters for the synthetic sequences have been set to 5 for the noise level, 10 for the shift level (except for `TWO PATTERNS` and `TWO PATTERNS MOD`, where the shift level is 5), and 10 for the warp level. Furthermore, none of the generated databases include *outlier patterns* or *observations* (check supplementary material in [28] for detailed information about each parameter). The rest of the databases in Table 2 correspond to those downloaded directly from the `UCR Archive`.

Table 2: Main features of benchmark databases.

Database ($\mathcal{U}$)	**Pattern length** (m)	**Num. of classes** (L)	**Num. patterns** (K)
`CBF` ($\Diamond$)	100	3	2223
`TWO PATTERNS` ($\Diamond$)	100	4	2444
`TWO PATTERNS MOD` ($\Diamond$)	100	2	2082
`SYNTHETIC CONTROL` ($\Diamond$)	100	6	2946
`RATIONAL` ($\Diamond$)	100	4	2444
`FACES UCR`	131	14	2205
`GUN POINT`	150	2	200
`PLANE`	144	7	210
`TWO LEAD ECG`	81	2	1162
`WAFER`	152	2	7164

For each database in Table 2, we construct a STSC problem which, we recall, consists of categorizing streaming time series based on a set of stored reference patterns (see Section 2.1). For this purpose, we split each database to craft the (i) reference pattern database, (ii) the PED training set, and (iii) the query set of streaming time series.

Let $\mathcal{U} = \{(U_k, l_k)\}_{k=1}^{K}$ be a benchmark database with K examples, where $U_k = (u_1, \ldots, u_m)$ represents the k-th sequence, and $l_k \in \{1, \ldots, L\}$ its class label. Given this notation, each set is given by:

– **Reference pattern database** ($\mathcal{B}$), which is a subset of $\mathcal{U}$ composed of L sequences, that is, one reference pattern per class. Each reference pattern in $\mathcal{B}$ results from the computation of the medoid over a randomly drawn set of 10 time series belonging to the class of the reference pattern at hand.
– **PED training database** ($\mathcal{D}_{\textbf{train}}$), which is used for fitting the PED model. It consists of a set of labeled streaming time series, each generated by randomly selecting and concatenating time series from a subset of $\mathcal{U}$. In this case, training STS are given

by:

$$\mathcal{D}_{\text{train}} = \{S_k^{\text{train}}\}_{k=1}^{K_{\text{train}}}, \tag{19}$$

where K_{train} represents the number of streaming time series in the training set $\mathcal{D}_{\text{train}}$. Each stream time series S_k^{train} is composed of $\beta_{\text{train}} = 10 \cdot L$ events (i.e., 10 events drawn uniformly at random for every class), yielding a total of $10 \cdot L \cdot m$ data points (see Section 2.1).

– **Query STS database ($\mathcal{D}_{\text{query}}$)**, which is created to assess the performance of the STSC task. Again, concatenating randomly selected sequences from a subset of $\mathcal{U}$, the query STS database consists of $K_{\text{query}} = 12$ streams, each composed of $\beta_{\text{query}} = 20 \cdot L$ events (i.e., 20 events per class selected uniformly at random without considering those in the training set, see Figure 6), and $20 \cdot L \cdot m$ data points. Formally, the set of query streaming time series is defined as:

$$\mathcal{D}_{\text{query}} = \{S_k^{\text{query}}\}_{k=1}^{K_{\text{query}}}, \tag{20}$$

where S_k^{query} denotes the k-th query STS.

Table 3: Summarized description of developed STSC problems.

Database ($\mathcal{U}$)	$\lvert\mathcal{B}\rvert$	$\mathcal{D}_{\text{train}}$		$\mathcal{D}_{\text{query}}$	
		K_{train}	β_{train}	K_{query}	β_{query}
`CBF` (◊)	3	50	30	12	60
`TWO PATTERNS` (◊)	4	37	40	12	80
`TWO PATTERNS MOD` (◊)	2	80	20	12	40
`SYN. CONTROL` (◊)	6	25	60	12	120
`RATIONAL` (◊)	4	37	40	12	80
`FACES UCR`	14	13	140	12	240
`GUN POINT`	2	80	20	12	40
`PLANE`	7	25	70	12	140
`TWO LEAD ECG`	2	80	20	12	40
`WAFER`	2	80	20	12	40

Table 3 shows a summarized description of the generated STSC problems. It is important to remark that the subsets ($\mathcal{U}_1$, $\mathcal{U}_2$ and $\mathcal{U}_3$) of time series used to build $\mathcal{B}$, $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{query}}$ as per the processes explained above are completely disjoint, i.e., no time series is utilized for constructing more than one of these databases. Figure 6 visually explains the disjoint nature of the generated databases.

4.4 Structure and Learning Process of the PED Model

Given one of the classification problems listed in Table 3, the memory parameter ρ, and the size of the sliding window w, we construct a PED model by following the steps described in Section 3.2. That is, we first compute the ODTW measurement matrix for each streaming time series in $\mathcal{D}_{\text{train}}$. Subsequently, we apply the sliding window to extract streaming frames, and label them to yield the set of supervised streaming frames $\mathcal{M}$. Finally, we learn a binary CNN classifier from $\mathcal{M}$. To this end, we split $\mathcal{M}$ into two non-overlapping subsets: training ($\mathcal{M}_{\text{train}}$) and validation ($\mathcal{M}_{\text{val}}$). The first set comprises 80% of the streaming frames in $\mathcal{M}$, and is used to train the CNN. The second set $\mathcal{M}_{\text{val}}$ contains the remaining 20%, and is utilized for validating the trained model.

Table 4: Main characteristics of the DNN training databases.

Database $\mathcal{U}$	Memory ρ		$\mathcal{M}_{\text{train}}$		$\mathcal{M}_{\text{val}}$	
	Range	Value	# frames	# streams	# frames	# streams
`CBF` (◊)	`middle` $0.1^{1/m}$	0.9772	18640	40	4660	10
`RATIONAL` (◊)	`middle` $0.1^{1/m}$	0.9772	18780	30	4382	7
`TP` (◊)	`middle` $0.1^{1/m}$	0.9772	18780	30	4382	7
`TP MOD` (◊)	`middle` $0.1^{1/m}$	0.9772	18360	60	6120	20
`TWO LEAD ECG`	`middle-short` $0.01^{1/m}$	0.9719	18360	60	6120	20
`SYN. CONTROL`◊	`middle` $0.1^{1/m}$	0.9772	18920	20	4730	5
`PLANE`	`middle-short` $0.01^{1/m}$	0.9685	22120	20	5530	5
`FACES UCR`	`middle` $0.1^{1/m}$	0.9826	22260	10	6678	3
`GUN POINT`	`middle` $0.1^{1/m}$	0.9848	18360	10	6120	10
`WAFER`	`middle` $0.1^{1/m}$	0.9850	18360	10	6120	10

For each STSC problem, Table 4 provides details on the memory and databases of the streaming frames considered for its training. Regarding the memory of the ODTW, we choose the value of ρ that performs the best for the PED model. To this end, we have considered three different CNN classifiers, each trained on frames of a particular ρ, and we have selected the parameter that produces the best validation AUC. In this design phase we have considered short ($\rho = 0.0001^{1/m}$), short-middle ($\rho = 0.01^{1/m}$) and middle ($\rho = 0.1^{1/m}$) range memories. These proposed values meet our claims made at the end of Section 2.1, where we stated that the ODTW should be computed with short-to-middle range memories. As mentioned at the end of Section 3.2, the size of the sliding window needs to be small, so as to enclose predictions of the PED close to the true limits between consecutive events. Thus, we use a fixed value of $w = 7$ time units for all the problems in Table 3.

Likewise, Tables 5a to 5c display the architecture of the CNN used for every STSC task. As can be observed in these tables, all classifiers consist of a series of stacked convolutional layers and a final fully-connected layer mapping the output of the last convolutional output to the binary target to be predicted. When backpropagating the gradients to learn the parameters of the model, we use the standard binary cross-entropy

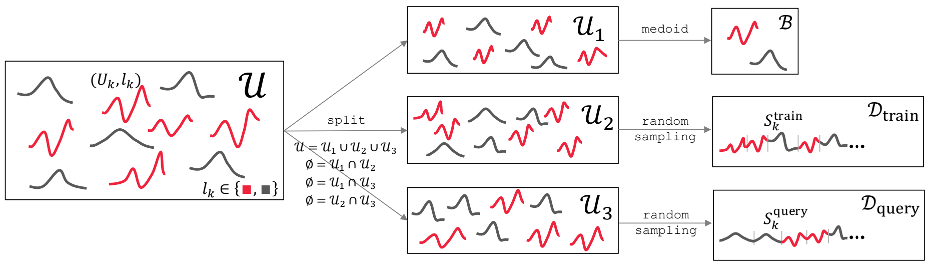

(U_k, l_k)
$\mathcal{U}$
$l_k \in \{$🟥$,$ ▨$\}$
split
$\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2 \cup \mathcal{U}_3$
$\emptyset = \mathcal{U}_1 \cap \mathcal{U}_2$
$\emptyset = \mathcal{U}_1 \cap \mathcal{U}_3$
$\emptyset = \mathcal{U}_2 \cap \mathcal{U}_3$
$\mathcal{U}_1$
$\mathcal{U}_2$
$\mathcal{U}_3$
medoid
random sampling
random sampling
$\mathcal{B}$
S_k^{train}
$\mathcal{D}_{\text{train}}$
S_k^{query}
$\mathcal{D}_{\text{query}}$

Fig. 6: Given a time series classification database $\mathcal{U}$, composed of two categories (grey and red), reference time series ($\mathcal{B}$) and streaming time series ($\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{query}}$) database generation process.

Table 5: DNN architecture for the considered STSC problems. The utilized epochs and batch sizes are specified in brackets next to the name of the databases.

Layer Type	**Description**	**Databases**
2D Convolutional	16 filters (3×3) stride $(1, 1)$, *same*, ReLu	`CBF` $(100, 100)$ `TWO PATTERNS` $(50, 100)$ `TWO PATTERNS MOD` $(50, 100)$ `SYNTHETIC CONTROL` $(25, 200)$ `RATIONAL` $(75, 100)$ `TWO LEAD ECG` $(50, 100)$ `GUN POINT` $(50, 100)$
2D Convolutional	32 filters (1×1) stride $(1, 1)$, *valid*, ReLu	
2D Max Pooling	Pool size (2×2), *valid*	
2D Convolutional	64 filters (1×1) stride $(1, 1)$, *valid*, ReLu	
2D Max Pooling	Pool size (2×2), *valid*	
Linear	20 units	
Dropout	0.3 rate	
Sigmoid	2 units	

(a)

Layer Type	**Description**	**Databases**
2D Convolutional	16 filters (3×3) stride $(1, 1)$, *same*, ReLu	`PLANE` $(20, 100)$ `FACES UCR` $(10, 500)$
2D Convolutional	1 filters (1×1) stride $(1, 1)$, *valid*, ReLu	
2D Convolutional	32 filters (3×3) stride $(1, 1)$, *same*, ReLu	
2D Max Pooling	Pool size (2×2), *valid*	
2D Convolutional	64 filters (3×3) stride $(1, 1)$, *same*, ReLu	
2D Convolutional	64 filters (3×3) stride $(1, 1)$, *valid*, ReLu	
2D Max Pooling	Pool size (2×2), *valid*	
Linear	20 units	
Dropout	0.3 rate	
Sigmoid	2 units	

(b)

Layer Type	**Description**	**Database**
2D Convolutional	16 filters (3×3) stride $(1, 1)$, *same*, ReLu	`WAFER` $(25, 100)$
2D Convolutional	32 filters (1×1) stride $(1, 1)$, *valid*, ReLu	
2D Max Pooling	32 filters (3×3) Pool size (2×2), *valid*	
2D Convolutional	64 filters (3×3) stride $(1, 1)$, *valid*, ReLu	
2D Convolutional	128 filters (1×1) stride $(1, 1)$, *same*, ReLu	
2D Max Pooling	Pool size (2×2), *valid*	
Linear	100 units	
Dropout	0.3 rate	
Linear	20 units	
Sigmoid	2 units	

(c)

loss and an Adam optimizer with a learning rate equal to 0.001.

Before proceeding with the discussion of the results from these experiments, we note that a public repository has been made available at `https://git.code.xx/active-stream-classifier.git`, which lists the databases, specifies the required Python packages and provides the scripts that reproduce all the results discussed in what follows.

5 Results and Discussion

We now discuss the obtained simulation outcomes, following the two-fold aim of the designed experimentation: to evaluate the accuracy of the proposed PED model when detecting changes between events (Subsection 5.1); and to analyze the contribution of PED when used to trigger active adaptation mechanisms in STSC problems (Subsection 5.2).

5.1 Performance Results of the PED Model

We recall that the detector is assumed to perform correctly if the predicted event changes *overlap* the true changes of the streaming time series. That is, given a streaming time series, we consider that PED correctly predicts event changes if the streaming frame corresponding to the i-th arriving observation, namely, $\overline{\mathbf{M}}^{i+1-w,i}$, is labeled as *event change*, and a real event change takes places between $i+1-w$ and i time instants.

This being said, we evaluate the accuracy of the PED by computing the precision and delay of its predicted changes for each database $\mathcal{U}$ in Table 3, and for each $S_k^{\text{query}} \in \mathcal{D}_{\text{query}}$. Results corresponding to `CBF`, `TWO PATTERNS`, `TWO PATTERNS MOD`, `WAFER` and `GUN POINT` databases are shown in Figure 7, whereas outcomes for `FACES UCR`, `RATIONAL`, `SYNTHETIC CONTROL`, `TWO LEAD ECG` and `PLANE` databases are depicted in Figure 8. In both cases, histograms on the top represent the distribution of the delay of predicted changes, where the red areas in the background lie between 0 and w, representing the temporal range over which true event changes take place. Accordingly, the more delay measurements fall within this band, the more accurate the PED is.

Table 6: Summary of the performance statistics of the PED model over the STSC problems under consideration. Regarding classification accuracy results, reported values correspond to mean and standard deviation of $\overline{\mathrm{ACC}}(i)$ as per Expression (21) over the entire simulated period.

Database $\mathcal{U}$	**Precision**	**Recall**	**Delay**	**Classification Accuracy**		
				PED-NN	GS	ODTW-NN
`CBF` (◊)	0.39 ± 0.06	0.41 ± 0.06	-0.06 ± 8.73	0.76 ± 0.24	0.88 ± 0.17	0.63 ± 0.42
`TWO PATTERNS` (◊)	0.26 ± 0.06	0.31 ± 0.07	-1.85 ± 18.31	0.47 ± 0.26	0.58 ± 0.30	0.40 ± 0.37
`TWO PATTERNS MOD` (◊)	0.32 ± 0.08	0.32 ± 0.08	0.14 ± 8.73	0.80 ± 0.23	0.85 ± 0.21	0.64 ± 0.43
`SYNTHETIC CONTROL` (◊)	0.84 ± 0.03	0.91 ± 0.03	0.98 ± 8.10	0.82 ± 0.17	0.88 ± 0.11	0.30 ± 0.15
`RATIONAL` (◊)	0.90 ± 0.04	0.90 ± 0.04	3.72 ± 4.07	0.53 ± 0.28	0.57 ± 0.28	0.46 ± 0.24
`FACES UCR`	0.67 ± 0.02	0.80 ± 0.02	0.43 ± 13.34	0.39 ± 0.20	0.45 ± 0.20	0.20 ± 0.21
`GUN POINT`	0.60 ± 0.09	0.62 ± 0.09	1.55 ± 8.24	0.74 ± 0.14	0.78 ± 0.12	0.53 ± 0.15
`PLANE`	0.96 ± 0.01	0.97 ± 0.01	4.08 ± 5.77	0.73 ± 0.34	0.87 ± 0.20	0.47 ± 0.40
`TWO LEAD ECG`	0.74 ± 0.06	0.74 ± 0.06	1.38 ± 2.81	0.69 ± 0.15	0.72 ± 0.12	0.68 ± 0.18
`WAFER`	0.56 ± 0.08	0.79 ± 0.09	-1.17 ± 24.89	0.55 ± 0.13	0.58 ± 0.14	0.51 ± 0.13

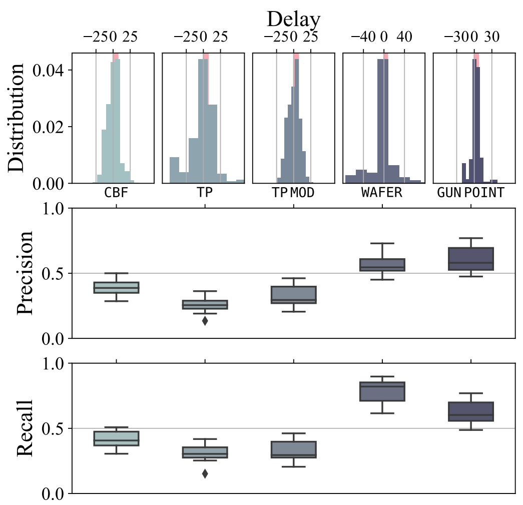


Fig. 7: PED model performance results for `CBF`, `TWO PATTERNS (TP)`, `TWO PATTERNS MOD (TP MOD)`, `WAFER` and `GUN POINT` STSC problems. On the top of the figure, histograms depict the distribution of the detection delay, where negative (positive) delays indicate the detector has predicted changes before (after) the real occurrence. In the lower part, boxplots illustrate the precision and recall of the PED model as per (17) and (18) respectively.

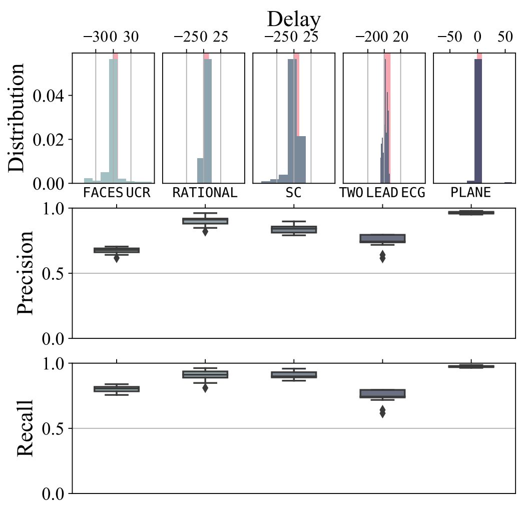


Fig. 8: PED model performance results for `FACES UCR`, `RATIONAL`, `SYNTHETIC CONTROL (SC)`, `TWO LEAD ECG` and `PLANE` classification problems. Similarly to Figure 7, histograms illustrate the distribution of the delay, and boxplots the PED model precision and recall.

When examining these figures in depth, we observe that PED trained over databases corresponding to Figure 8 shows delay distributions with lower variance than those in Figure 7. However, it is important to note that, given the shape of the patterns within this last group (see `CBF` in Figure 9), the limits established for these examples become unclear for the ODTW/DTW, due to their elasticity property. Hence, although the predictor appears to be quite imprecise, such an inaccuracy does not alter the main structure of the patterns, as will be later supported by the STSC performance results (see discussion at the end of Section 5.2).

Regarding the precision of PED, boxplots depicted at the bottom of Figures 7 and 8 illustrate the distribution of this score for each of the databases. According to Expression (17), the precision lies between 0 and 1, where 0 indicates failure, and 1 perfect performance. In line with the delay results, PED models learnt for databases in Figure 8 are more precise than those models in Figure 7.

A summarized overview of these results is given in Table 6, which numerically exposes the average precision and delay for each database. We can observe that except for `CBF`, `TWO PATTERNS`, `TWO PATTERNS MOD` and GUN POINT, the reported results for the considered benchmark are good in terms of precision. As argued previously, the limits between consecutive events in CBF, TWO PATTERNS, TWO PATTERNS MOD and GUN POINT databases are diffuse for the ODTW/DTW similarity as per a visual inspection of the events in Figures 9 and 10. Specifically, each black curve in these plots represents a streaming time series (last five events) of the $\mathcal{D}_{\text{query}}$ set; vertical dashed lines depict limits between consecutive events; and red dots indicate changes predicted by the PED, respectively. Since the boundaries between events is diffuse in these databases, streaming frames loose discrimination power, leading to a degradation of the precision and an increase of the delay of PED for these databases.

5.2 Efficiency of PED for STSC problems

We now proceed by evaluating the efficiency of PED when combined with a DTW-NN classifier to solve STSC problems. As stated in Section 4, we use two alternative NN-based classifiers to compare the performance of our PED based DTW-NN classifier: (i) gold-standard (GS) and (ii) the ODTW-based nearest neighbor (ODTW-NN) procedure. To this end, we define the classification accuracy for the i-th arriving query stream point of $S_k^{\text{query}} \in \mathcal{D}_{\text{query}}$ as:

$$\mathrm{ACC}_k(i) = \begin{cases} 1 \text{ if } \mathrm{PL}(i) = \mathrm{TL}(i), \\ 0 \qquad \text{otherwise,} \end{cases} \tag{21}$$

where $\mathrm{PL}(i)$ and $\mathrm{TL}(i)$ are the predicted and true label for the arriving point i, respectively. Accordingly, $\mathrm{TL}(i)$ is the same for all instant i belonging to the same event.

Results corresponding to CBF, RATIONAL and TWO LEAD ECG STSC problems are displayed at the bottom of Figure 9 and 10, where each curve shows the performance of PED-NN (green), GS (black) and ODTW-NN (gray) classifiers. More precisely, such plots display the accuracy over time averaged over the $K_{\text{query}} = 12$ streaming time series in $\mathcal{D}_{\text{query}}$, namely:

$$\overline{\mathrm{ACC}}(i) = \frac{1}{K_{\text{query}}} \sum_{k=1}^{K_{\text{query}}} \mathrm{ACC}_k(i), \tag{22}$$

where $\mathrm{ACC}_k(i)$ represents the accuracy of the i-th arriving point as per Equation (21).

We focus on the results obtained for CBF STSC problem (Figure 9a) due to its simplicity and straightforward interpretation. For this problem, we observe that both PED-NN and GS render similar average accuracy results, especially in the middle of every event. On the contrary, at the beginning/end of the event the predictions issued by the PED-NN model degrade, getting closer to those corresponding to the ODTW-NN approach. Indeed, if we pay attention to the outputs of PED (red dots), we observe that such predictions have an ample variability, which implies that the similarity is reset in the PED-NN model sooner/later than expected. As a result, the accuracy decreases around the true limits of the event.

For the sake of brevity, we have only described in detail the evolution of $\overline{\mathrm{ACC}}(i)$ over time for CBF, RATIONAL and TWO LEAD ECG representative experiments. The detailed results for the remaining databases are depicted in Figures 9 and 10. According to the division made in the previous section, we have grouped in Figure 9 the databases with diffuse event changes (e.g., CBF). On the other hand, results in Figure 10 correspond to databases with well-delimited event changes, such as RATIONAL and TWO LEAD ECG problems.

Averaged results obtained from all the STSC experiments are summarized in Table 6. For each approach in the benchmark (PED-NN, GS and ODTW-NN), the last three columns show the $\overline{\mathrm{ACC}}(i)$ results averaged for all data points arriving over the simulated period. As can be concluded from these results, both PED-NN and GS render similar performance levels in almost all the conducted experiments, which suggests that decisions made on predictions of the PED model do not affect the structure of the events composing the streaming time series. Moreover, if we compare the results of PED-NN and ODTW-NN classifiers, we can see that the former leads to better results. Hence, our proposed PED not only permits to efficiently adapt the NN classifier to upcoming events without disturbing any essential relationship among consecutive data points, but also yields more accurate results as it incorporates DTW (which uses information of the full event) instead of ODTW (which forgets information of the past as it moves forward in the event).

On a closing note, these results are conclusive in regards to the convenience of active adaptation strategies for streaming time series classification in contrast to passive adaptation methods such as the ODTW proposed in [17]. Furthermore, they emphasize the rich information about the stream dynamics contained in the measurement matrix. The use of this information has spanned several research lines that are outlined in the next section.

6 Conclusions and Future Work

In this work we have proposed to exploit the information embedded in streaming frames (namely, partial similarity measurements among all sub-sequences of streaming time series) towards identifying the change

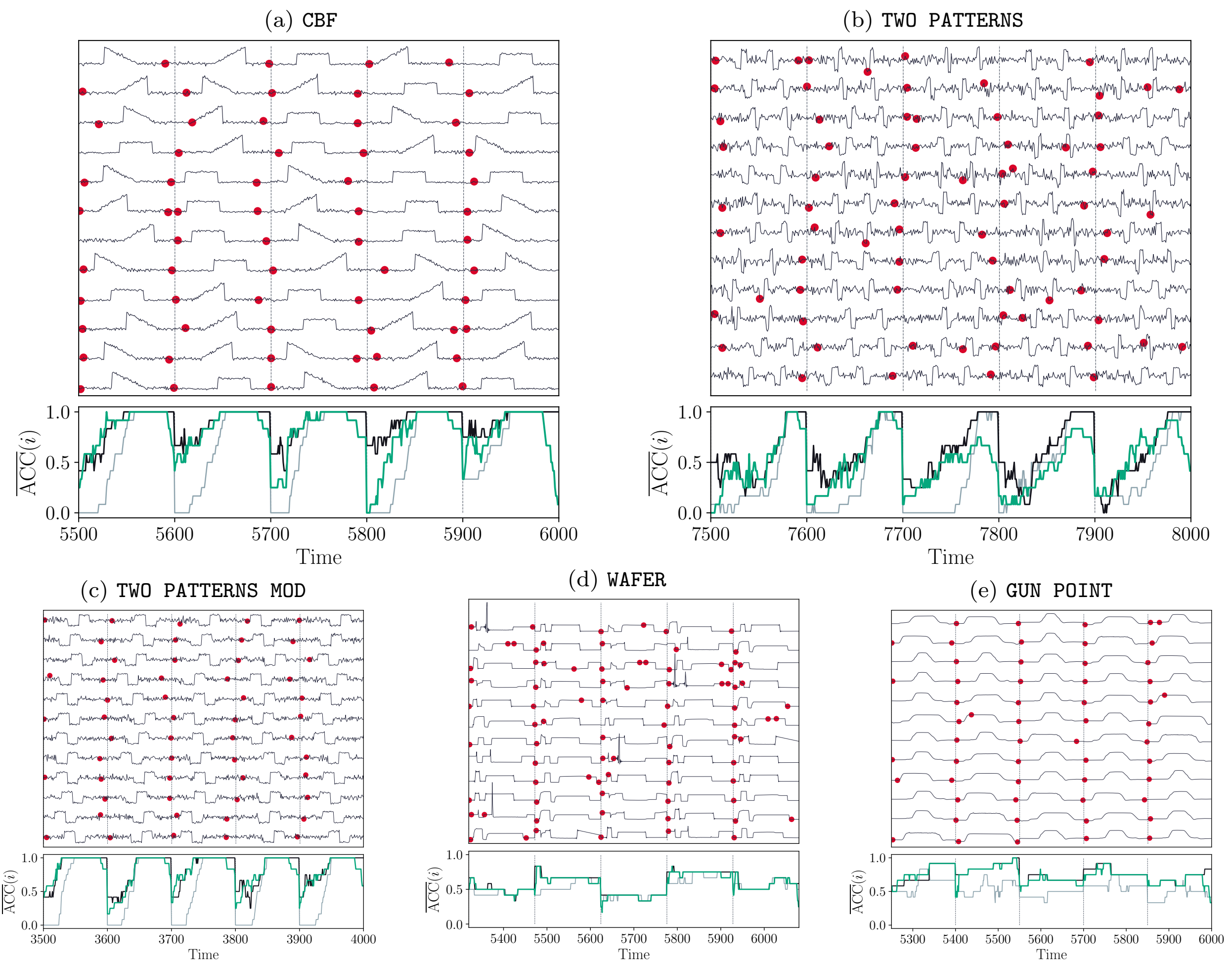


Fig. 9: Results of the PED model in terms of *event change* predictions (top) and classification accuracy of the PED-NN model over STSC problems (bottom) for (a) `CBF`, (b) `TWO PATTERNS`, (c) `TWO PATTERNS MOD`, (d) `WAFER` and (e) `GUN POINT` problems. On the top, black curves depict the last 5 events of each streaming time series in $\mathcal{D}_{\text{test}}$, black vertical lines depict event changes and red dots PED model change predictions. On the bottom, black, grey and green curves illustrate average accuracy scores $\overline{\mathrm{ACC}}$ along the last 5 events of test streams for GS, ODTW-NN and PED-NN classifiers, respectively.

between consecutive events in streaming time series. Our motivation for this purpose is to adapt DTW-NN models to such changes by leveraging the identification of such event changes. To this end, we have proposed PED, a model that categorizes streaming frames over time into *event change* or *no event change* classes. When a change is detected, a DTW-NN can adapt to the upcoming event by resetting the computation of the DTW on which it relies to predict the class of the arriving stream data.

A benchmark comprising 10 streaming time series classification databases has been designed to assess the performance of the proposed model. Specifically, we have compared it to that of i) the passive adaptation mechanism proposed in [17], and ii) a GS DTW-NN scheme that assumes perfect a priori knowledge of the instants at which event transitions occur to reset the DTW computation. The reported results of PED in terms of delay and false alarms have provided empirical insights on the utility of the proposed method when combined with DTW-based NN classifiers. In particular, classifiers endowed with the proposed PED and a GS detector have been found to behave similarly in terms of predictive performance, which ultimately buttress the suitability of streaming frames to detect event changes.

Several research directions rooted in this work are planned for the future aimed at investigating further uses of the information contained in streaming frames, including clustering and early classification formulated over streaming time series. Moreover, alternative deep neural network architectures can also be studied to de-

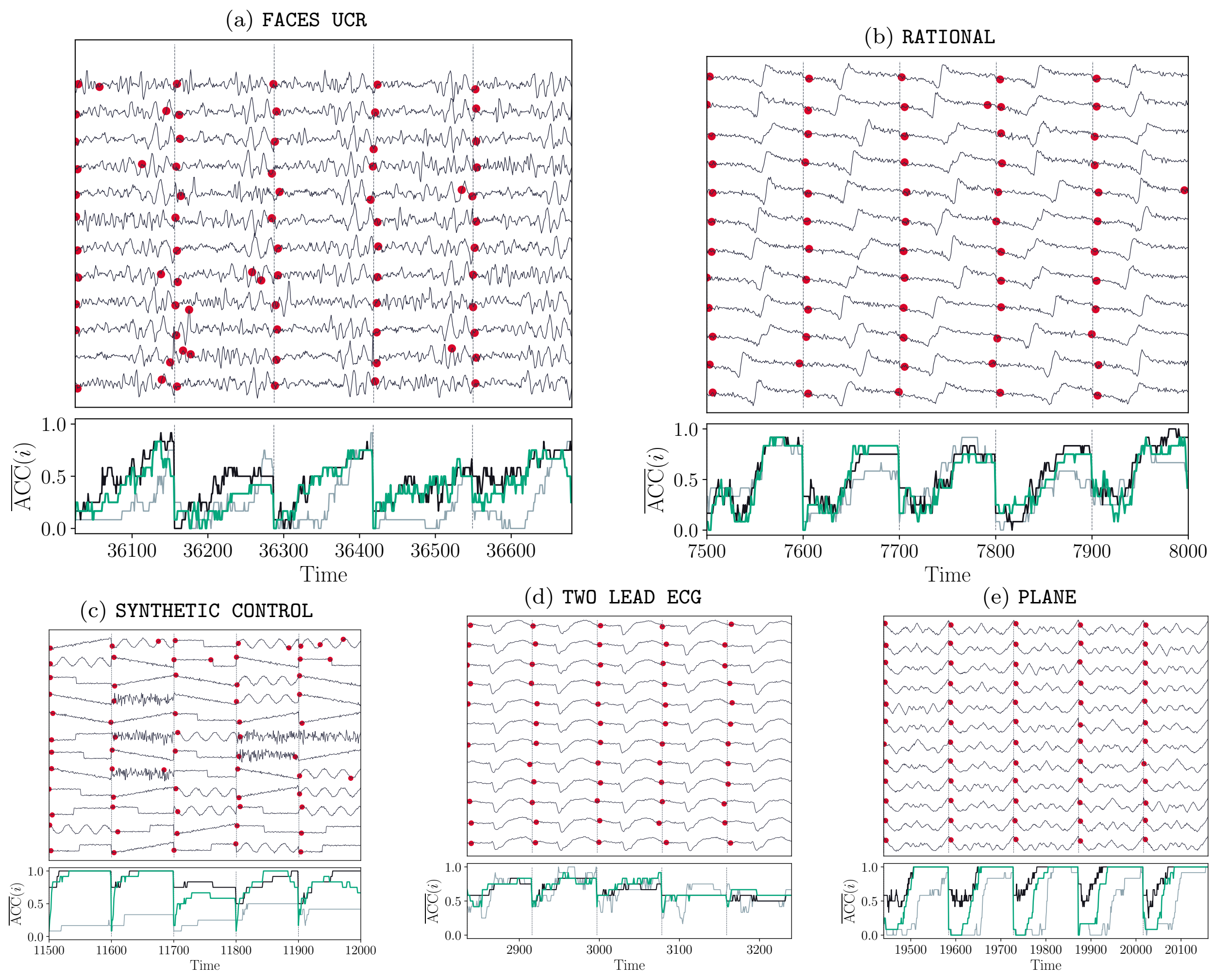


Fig. 10: PED model *change* predictions (top) and average accuracy of the PED-NN model (bottom) for (a) `FACES UCR`, (b) `RATIONAL`, (c) `SYNTHETIC CONTROL`, (d) `TWO LEAD ECG` and (e) `PLANE` STSC problems.

crease the variability of predicted event changes. Finally, given the potential of streaming frames to detect changes between events, further efforts will be invested towards examining whether streaming frames by themselves can discriminate among labels in streaming time series classification, without requiring any additional classifier.

Acknowledgements This research work has been supported by the Basque Government through the EMAITEK and ELKARTEK funding programs (3KIA, ref. KK-2020/00049), as well as by the Consolidated Research Group MATHMODE (IT1294-19) granted by the Department of Education of this institution. A. Pérez and J. A. Lozano are supported by the Basque Government through the BERC 2018-2021 program and by the Spanish Ministry of Economy and Competitiveness MINECO through BCAM Severo Ochoa excellence accreditation SEV-2017-0718. A. Pérez also acknowledges funding support from AEI/FEDER (UE) through project TIN2017-82626-R. J. A. Lozano is also supported by Spanish Ministry of Economy and Competitiveness MINECO through TIN2016-78365-R.

Author contributions

Conceptualization: I. Oregi, A. Pérez, J. Del Ser; Methodology: I. Oregi, A. Pérez; Formal analysis and investigation: I. Oregi, A. Pérez, J. Del Ser; Writing - original draft preparation: I. Oregi, A. Pérez; Writing - review and editing: I. Oregi, A. Pérez, J. Del Ser, J. A. Lozano; Funding acquisition: J. Del Ser, J. A. Lozano; Supervision: A. Pérez, J. Del Ser.

Conflict of interest

The authors declare no conflict of interest.

References

1. Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010.
2. Georg Krempl, Indre Žliobaite, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, et al. Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter*, 16(1):1–10, 2014.
3. Pedro Pereira Rodrigues, João Gama, and Joao Pedroso. Hierarchical clustering of time-series data streams. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):615–627, 2008.
4. Jorge-L Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, 2016.
5. Eric L Manibardo, Ibai Laña, Jesus L Lobo, and Javier Del Ser. New perspectives on the use of online learning for congestion level prediction over traffic data. *arXiv preprint arXiv:2003.14304*, 2020.
6. Ibai Lana, Javier Del Ser, Manuel Velez, and Eleni I Vlahogianni. Road traffic forecasting: Recent advances and new challenges. *IEEE Intelligent Transportation Systems Magazine*, 10(2):93–109, 2018.
7. Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.
8. Leandro L Minku, Allan P White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on knowledge and Data Engineering*, 22(5):730–742, 2009.
9. Zoltán Bankó and János Abonyi. Correlation based dynamic time warping of multivariate time series. *Expert Systems with Applications*, 39(17):12814 – 12823, 2012.
10. Charlotte Pelletier, Geoffrey I Webb, and François Petitjean. Temporal convolutional neural network for the classification of satellite image time series. *Remote Sensing*, 11(5):523, 2019.
11. Mahtab J Fard, Abhilash K Pandya, Ratna B Chinnam, Michael D Klein, and R Darin Ellis. Distance-based time series classification approach for task recognition with application in surgical robot autonomy. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 13(3), 2017.
12. João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.
13. Izaskun Oregi, Aritz Pérez, Javier Del Ser, and José A Lozano. On-line dynamic time warping for streaming time series. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 591–605. Springer, Cham, 2017.
14. Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.
15. Rodolfo C Cavalcante, Leandro L Minku, and Adriano LI Oliveira. Fedd: Feature extraction for explicit concept drift detection in time series. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 740–747. IEEE, 2016.
16. Samaneh Aminikhanghahi and Diane J Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.
17. Izaskun Oregi, Aritz Pérez, Javier Del Ser, and Jose A Lozano. On-line elastic similarity measures for time series. *Pattern Recognition*, 88:506–517, 2019.
18. Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.
19. Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48, 2010.
20. Amaia Abanda, Usue Mori, and Jose A Lozano. A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2):378–412, 2019.
21. Tomasz Górecki and Maciej Łuczak. Multivariate time series classification with parametric derivative dynamic time warping. *Expert Systems with Applications*, 42(5):2305–2312, 2015.
22. 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.
23. Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Workshop on Knowledge Discovery in*

Databases, pages 359–370. Seattle, WA, 1994.

24. Richard E Bellman and Stuart E Dreyfus. *Applied dynamic programming.* Princeton university press, 2015.
25. Feng Zhou, Fernando De la Torre, and Jessica K Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):582–596, 2013.
26. Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press Cambridge, 2016.
27. Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.
28. Usue Mori, Alexander Mendiburu, and Jose A Lozano. Similarity measure selection for clustering time series databases. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):181–195, 2015.