Advances in Data Analysis and Classification (2021) 15:911–936
https://doi.org/10.1007/s11634-021-00437-8

REGULAR ARTICLE


# Learning multivariate shapelets with multi-layer neural networks for interpretable time-series classification

**Roberto Medico[1] · Joeri Ruyssinck[1] · Dirk Deschrijver[1] · Tom Dhaene[1]**

Received: 27 August 2019 / Revised: 17 November 2020 / Accepted: 13 February 2021 /
Published online: 4 March 2021


## Abstract

Shapelets are discriminative subsequences extracted from time-series data. Classifiers using shapelets have proven to achieve performances competitive to state-of-the-art methods, while enhancing the model's interpretability. While a lot of research has been done for univariate time-series shapelets, extensions for the multivariate setting have not yet received much attention. To extend shapelets-based classification to a multidimensional setting, we developed a novel architecture for shapelets *learning*, by embedding them as trainable weights in a multi-layer Neural Network. We also investigated the introduction of a novel learning strategy for the shapelets, comprising of two additional terms in the optimization goal, to retrieve a reduced set of uncorrelated shapelets. This paper describes the proposed architecture and presents results on ten publicly available benchmark datasets, as well as a comparison with existing state-of-the-art methods. Moreover, the proposed optimization objective leads the model to automatically select smaller sets of uncorrelated shapelets, thus requiring no additional manual optimization on typically important hyper-parameters such as number and length of shapelets. The results show how the proposed approach achieves competitive performance across the datasets, and always leads to a significant reduction in the number of shapelets used. This can make it faster for a domain expert to match shapelets to real patterns, thus enhancing the interpretability of the model. Finally, since the shapelets learnt during training can be extracted from the model they can

✉ Roberto Medico
roberto.medico@ugent.be

Joeri Ruyssinck
joeri.ruyssinck@ugent.be

Dirk Deschrijver
dirk.deschrijver@ugent.be

Tom Dhaene
tom.dhaene@ugent.be

[1] IDLab, iGent Tower - Department of Information Technology, Ghent University - imec, Technologiepark-Zwijnaarde 126, 9052 Ghent, Belgium

serve as meaningful insights on the classifier's decisions and the interactions between different dimensions.

**Keywords** Shapelets · Time-series classification · Machine learning · Neural networks

**Mathematics Subject Classification** 68T07 Artificial neural networks and deep learning · 62H30 Classification and discrimination · 62M10 Time series

## 1 Introduction

Multivariate time-series classification is receiving an increasing interest given the ubiquity and availability of such data in several domains, including e.g. industrial manufacturing (in the form of machinery's' sensor readings), medical measurements or Internet-of-Things applications. Sensor data is mostly multivariate and often the interactions between different sensors are of interest for the analysis. A well-established and prolific research exists for univariate time-series classification (Bagnall et al. 2017), and recently efforts have also been made to adapt these existing methodologies, or come up with novel approaches, for the multivariate case (Karim et al. 2019; Wang et al. 2016). When dealing with time-series, often the interest is not only to have a satisfactory classification performance but also to obtain insights on the data. In machine learning, this is often a challenge, given that most powerful classifiers are black-box. However, it is still possible to obtain interpretable results by making use of white-box features, which a classifier can then base its predictions on. One recently proposed approach for univariate time-series classification that relies on interpretable features is based on *shapelets*, i.e. maximally discriminative subsequences of time-series data. Shapelets were originally proposed in Ye and Keogh (2009) as an innovative supervised motif discovery algorithm, where univariate shapelets are searched within a time-series exhaustively among all possible candidates (subsequences) using a decision-tree-like approach. Each candidate subsequence is evaluated according to the information gain obtained at each node, using the distance between the shapelet and each time-series and an optimally chosen threshold for the split. The length of the shapelets is a hyperparameter of the algorithm. Given the brute force nature of the approach, this strategy does not scale well with the size of the data, making it inapplicable for larger datasets. An extension of the original idea was introduced in Rakthanmanon and Keogh (2013) with the FastShapelets algorithm, where the authors proposed to convert each time-series into its SAX (Symbolic Aggregate approXimation) representation (Lin et al. 2007), and perform the random search for shapelets in this new lower dimensional space. In Hills et al. (2014) the authors proposed to separate the discovery process from the classification task (Shapelet Transform): first, shapelets are extracted using the FastShapelets algorithm, and afterwards the training data is projected onto a new feature space, by computing the minimum distance of each time-series with each shapelet. This is done by sliding a shapelet over each time-series, computing the distances with all its subsequences and finally finding the minimum distance. Using this new feature space as input for traditional classifiers such as SVM or Random Forest has

shown to improve classification accuracy compared to the original tree-based approach (Lines et al. 2012). All above methods were proposed for univariate time-series, but some research also investigated possible extensions to the multi-dimensional case. In Cetin et al. (2015), the authors propose an ensemble method of shapelet-based decision trees built independently on each univariate dimension. In Karlsson et al. (2016), a tree-based ensemble (Random Shapelet Forest) is built using many trees, constructed by random sampling both the dimension and the shapelets in each tree. The predictions of individual trees are then ensembled by majority voting to obtain the final classifications. Another approach is to convert a multivariate dataset into a univariate representation by e.g. concatenating the dimensions, and then applying existing discovery algorithms to the new representation. In Patri et al. (2015), this representation is obtained by interleaving time-series subsequences extracted from multiple channels. In Bostrom and Bagnall (2017), the authors propose an extension of Shapelet Transform to the multivariate case by extracting multi-dimensional shapelets: (i) independently from each dimension, (ii) dependently across dimensions, maintaining the phase and (iii) independently across dimensions, looking for the optimal location in each dimension during matching. All the approaches mentioned so far are based on the enumeration (or random search) of potential candidates (subsequences), and can be categorized as *discovery* algorithms, in the sense that shapelets are looked for among subsequences of the training data, and their quality is evaluated according to a defined criterion (e.g. information gain in tree-based approaches). Within this category, each shapelet is constrained to be a subsequence of existing data. An approach alternative to discovery that does not require computationally heavy candidate search is shapelets *learning*, where shapelets are learnt from scratch directly from the data using a machine learning model. Since shapelets are learnt and updated by the model, they are not constrained to be subsequences of the training data. Shapelets learning was originally proposed in Grabocka et al. (2014) for univariate time-series data, where a logistic regression classifier is trained to jointly learn shapelets and weights. Shapelets are initialized with a rough guess (the authors propose random or KMeans-based initialization) and iteratively refined during the learning process. The learning model proposed (with $\hat{\mathbf{y}}$ as the approximated binary target labels, $\mathbf{M}$ as the minimum distances between each shapelets and the training data, $\mathbf{W}$, $W_0$ as the linear weights) is:

$$\hat{y}_i = W_0 + \sum_{k=1}^{K} M_{i,k} W_k, \quad \forall i \in 1, \ldots, N \tag{1}$$

where $K$ is the number of shapelets and $N$ the number of time-series in the training data. The features used are the minimum distances $\mathbf{M}$, as proposed in Lines et al. (2012). A regularized logistic loss between approximated $\hat{\mathbf{y}}$ and real $\mathbf{y}$ targets is then optimized via stochastic gradient descent. An extension of this technique to the multivariate case was recently also proposed in Wang and Wu (2017), where the authors use multivariate distances between subsequences and shapelets as predictors $\mathbf{M}$ for a linear logistic classifier. The work presented in our paper contributes to this research direction, and extends the existing approaches to non-linear decision boundaries, by embedding the shapelets learning in the architecture of a Neural Network model, thus

leveraging the recent advances in computational power granted by GPUs and allowing the learning on a bigger scale by deepening the architecture. Specifically, the main contributions presented in this paper are the following:

1. A novel architecture for learning multivariate shapelets, based on *embedding* the shapelets as *trainable weights* of a multi-layer Neural Network model to allow for *non-linear* decision boundaries, as discussed in Sect. 2.2;
2. The introduction of a novel *optimization objective* comprising additional terms to control the number and correlation of the learnt shapelets, as described in Sect. 3.1;
3. An evaluation *benchmark* of the proposed model on ten publicly available multivariate time-series datasets, presented in Sect. 4.

## 2 Multi-layer neural network

The following definitions will introduce symbols and notations used throughout the paper. We use the words *channel* and *dimension* interchangeably, referring to the number of different measurements which each time-series in the data consists of.

### 2.1 Definitions

**Definition 1** *(Multivariate Time-Series Dataset, MVD)* A multivariate time-series dataset is a collection of $N$ multivariate time-series instances. Each time-series consists of $Q$ sequential measurements of $C$ different channels. Such dataset is defined as $\mathbf{T}^{N\times Q\times C}$.

**Definition 2** *(Multivariate Shapelets Matrix, MSM)* A multivariate shapelet matrix is a matrix consisting of $K$ multivariate shapelets of the same length. If the length is defined as $L$, the shapelets matrix can be defined as $\mathbf{S}^{K\times L\times C}$.

**Definition 3** *(Minimum Distance Matrix, MDM)* The minimum distance matrix $\mathbf{M}^{N\times K}$ between an MVD $\mathbf{T}^{N\times Q\times C}$ and an MSM $\mathbf{S}^{K\times L\times C}$ is defined as:

$$M_{i,j} = \min_k dist(T^i_{k,k+L}, S_j) \quad \forall i \in \{1,\ldots,N\}, j \in \{1,\ldots,K\} \tag{2}$$

i.e. $M_{i,j}$ is the minimum distance between all subsequences of length $L$ of the $i$-th time-series from the $j$-th shapelet. The distance function $dist$ is the Euclidean distance in a $C$-dimensional space.

### 2.2 Architecture

The model proposed in this paper embeds the shapelets learning process in the architecture of a Neural Network by introducing custom layers (Distance Layers), responsible of computing the MDM as in Eq. (2), between input data and shapelets (Fig. 1). Each Distance Layer $\mathrm{DL_k}$ outputs the values of the MDM between $K$ shapelets of length $L_k$ and the input data. The description of the model (and algorithm) is generic for a classification task with $P$ classes, with $P \geq 2$.

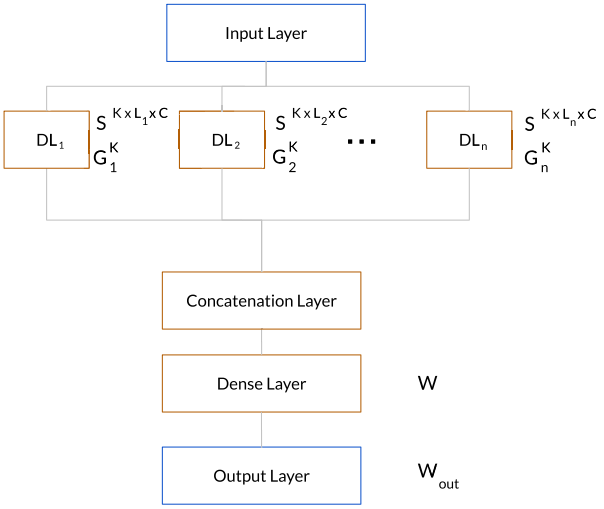



**Fig. 1** Architecture of the proposed multi-layer Neural Network, with 1 hidden layer. The parameters associated with each layer are shown on the right. $DL_k$ indicates the $k_{th}$ Distance Layer, $S$ is the shapelet matrix, $W$ and $W_{out}$ are fully-connected weights, and G are additional gating parameters as introduced in Sect. 2.2.2

### 2.2.1 Layers

The network consists of three kind of layers:

1. *Distance layer*: this layer receives a batch of size $B$ of multivariate time-series from the training data $\mathbf{T}^{B\times Q\times C}$, and outputs the Minimum Distance Matrix $\mathbf{M}^{B\times K}$ computed as in (2). This layer consists therefore of $K$ neurons, one for each shapelet. The trainable weights of this layer correspond to the shapelets being learnt, $\mathbf{S}$. It follows from (2) that:

$$M_{i,j} \geq 0 \quad \forall\, i, j,$$

   therefore no activation is applied to the output.
2. *Hidden Layer(s)*: one or multiple fully-connected layers with input $\mathbf{M}$, weights $\mathbf{W}$ and output $f(\mathbf{Y})$, where $f(\cdot)$ is the ReLU activation (Glorot et al. 2011):

$$f(\mathbf{Y}) = \max(0, \mathbf{Y})$$

   and $\mathbf{Y}$ is the pre-activation

$$\mathbf{Y} = \mathbf{M} \cdot \mathbf{W} + \mathbf{b}.$$

3. *Output layer*: the final layer consists of $P$ neurons, where $P$ is the number of classes, and has associated weights $\mathbf{W_{out}}$. The pre-activation $\mathbf{Y}$ is fed into a softmax

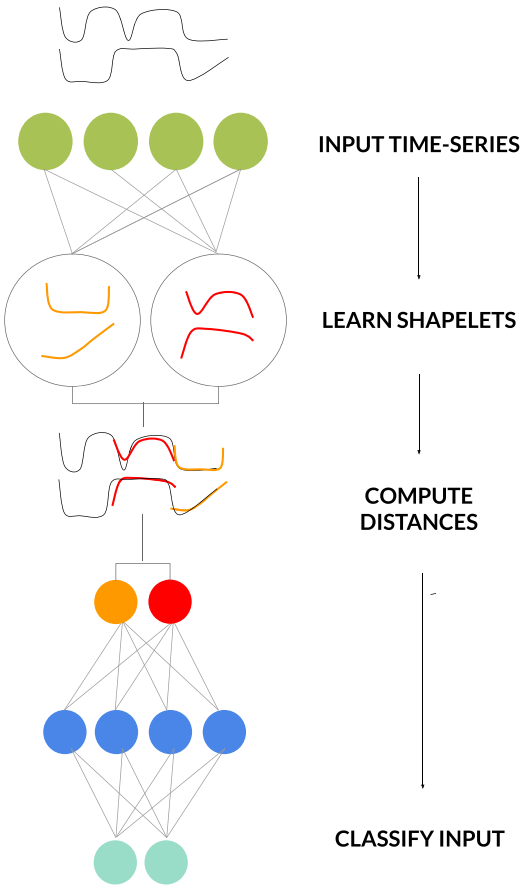



**Fig. 2** Example of the proposed model, using 2 shapelets (red and orange) of the same length on a 2D time-series input, and 1 hidden dense layer. A 2D input time-series is fed to the model, which updates the shapelets and computes the minimum distances. Finally, the distances are used as features for the classification (color figure online)

activation for binary or multiclass classification:

$$\sigma(\mathbf{Y})_j = \frac{e^{Y_j}}{\sum_{i=1}^{P} e^{Y_i}} \quad \forall j = 1, \ldots, P$$

The $p^{th}$ class that satisfies

$$\sigma(\mathbf{Y})_p > \sigma(\mathbf{Y})_k \quad \forall k \in \{1, \ldots, P\} \backslash \{p\}$$

is chosen as the predicted class. In case of a tie, the predicted class is chosen randomly among the tying candidates.

As shown in Fig. 1, in practice the model learns shapelets of different sizes by using multiple Distance Layers. The outputs of each of these layers (i.e. the distances) are concatenated before being fed to the hidden layer(s).

### 2.2.2 Learning objective

The proposed learning objective comprises of three components: classification loss $H$, *redundancy* $C$ and *correlation* $E$

*Classification term,* $H$: Similar to Grabocka et al. (2014), the classification task is solved by minimizing the (binary or categorical) cross-entropy between the predicted output $\hat{\mathbf{y}}$ and the labels $\mathbf{y}$ defined as:

$$H_y(\hat{y}) = -\sum_i y_i \log(\hat{y_i})$$

*Redundancy term,* $C$: In this work, we propose to modify the learning objective by adding two terms to the loss, aimed at limiting the number of shapelets found, and reducing their correlation. To achieve this, we introduce in each Distance Layer additional parameters $\mathbf{G} \in \mathbb{R}^K$. Based on these, we introduce a term $C$ in the loss, defined as:

$$C = \sum_{k=1}^{K} h_\theta(G_k)/K, \quad C \in [0, 1]$$

where $h_\theta(\cdot)$ is the sigmoid function and $\theta$ a parameter to control its saturation:

$$h_\theta(G_k) = \frac{1}{1 + e^{-\theta G_k}}$$

In other words, the values $\mathbf{h}_\theta(\mathbf{G}) = [h_\theta(G_1), \ldots, h_\theta(G_K)]$ - referred to as *gating* parameters throughout the reminder of this text - act as *gates* and dynamically *select* the shapelets among the pool of available candidates. The $\mathbf{G}$ parameters are thus directly responsible of pushing the output of the sigmoid function towards higher (lower) values as to select (discard) specific shapelets.

Additional trainable parameters are often used in Neural Network-based architectures (not limited to TSC tasks) to achieve ad-hoc tasks, such as dynamic gating, pruning etc. Some examples using trainable custom weights include channel gating networks (Hua et al. 2019), dynamic channel pruning (Gao et al. 2018). In the domain of TSC, in Raychaudhuri et al. (2017) e.g. the Authors introduce additional parameters to "mask" specific dimensions in multivariate time-series data while learning shapelets.

The $C$ term forces the number of selected shapelets to be as low as possible. Since the gating parameters should be as close to binary values as possible, i.e. $h_\theta(G_k) \in \{0, 1\},\ \forall k$, the value of $\theta$ is on purpose set to be $\gg 1$. Note that, unlike when used as an activation for a fully-connected layer, in this case the tendency of the sigmoid function to saturate is actually a property that we can exploit. Indeed, given that the training is joint and small changes are applied to the $\mathbf{G}$ parameters during training, we would like to *amplify* these in the actual output of the sigmoid function, to eventually force the parameters to be close to binary values. Throughout our experiments, we found

the setting $\theta = 100$ is appropriate for this goal, offering an activation with enough steepness to capture these small changes.
To be able to learn the gating parameters, these need to be part of the (differentiable) computational graph of our model, so that gradient updates can be back-propagated through the network during training. To achieve this, we modify the Distance Layer as to compute the following output on a mini-batch of size $B$:

$$\begin{aligned}\mathbf{M}' \in \mathbb{R}^{B\times K} &= \mathbf{M} \cdot \mathbf{diag}(h_\theta(G_1), \ldots, h_\theta(G_K)) \\ &= \begin{bmatrix} M_{1,1} & \ldots & M_{1,K} \\ \vdots & \ddots & \vdots \\ M_{B,1} & \ldots & M_{B,K} \end{bmatrix} \begin{bmatrix} h_\theta(G_1) & \ldots & 0 \\ & \ddots & \\ 0 & \ldots & h_\theta(G_K) \end{bmatrix}\end{aligned}$$

In other words, the $k^{th}$ shapelet is ignored during training as $h_\theta(G_k) \to 0$, since the corresponding distances are set to 0 regardless of the input.
*Correlation term, $E$*: A second additional term $E$ controls how correlated the selected shapelets are, again with the goal of limiting their number and helping the model discard redundant patterns:

$$E = \max_{i,j \in U} |\text{Corr}(\mathbf{M}')_{ij}|, \quad E \in [0, 1]$$

where $\text{Corr}(\mathbf{M}') \in \mathbb{R}^{K\times K}$ is the correlation matrix of $\mathbf{M}'$, and $U = \{i, j : i > j \ \ \forall\, i, j \in [1, K]\}$, i.e. $U$ is the upper triangular matrix of $Corr$, excluding the diagonal values.

In simpler words, the two additional terms $C$ and $E$ try to lead the model to learn (i) a smaller subset of shapelets and (ii) uncorrelated patterns. The rationale behind the introduction of these terms is that, without additional constraints, the model typically converges to sub-optimal solutions using irrelevant (random noise), redundant (motifs) or many similar (correlated) shapelets. Indeed, while a shapelets-based model can achieve good accuracy when using a larger set of shapelets, it loses its interpretability as many irrelevant patterns are also learnt. Additionally, this allows one to avoid the optimization of number and length of shapelets, which are crucial hyper-parameters in shapelets-based models (Ye and Keogh 2009; Grabocka et al. 2014).

**Combined objective** As both terms $C$ and $E$ are to be minimized, the complete learning objective problem can be formulated as:

$$\begin{aligned}&\min_{\mathbf{S},\mathbf{G},\mathbf{W}} (1-\alpha) H_y(\hat{y}) + \alpha(C + E) \\ &\text{where } \alpha \in \left[0, \frac{1}{2}\right]\end{aligned} \tag{3}$$

Note that shapelets $\mathbf{S}$, network weights $\mathbf{W}$ and $\mathbf{G}$ parameters all appear in Eq. (3) and are therefore jointly updated during training gradient descent and back-propagation. The parameter $\alpha$ plays a very important role in the loss, as it sets the trade-off between

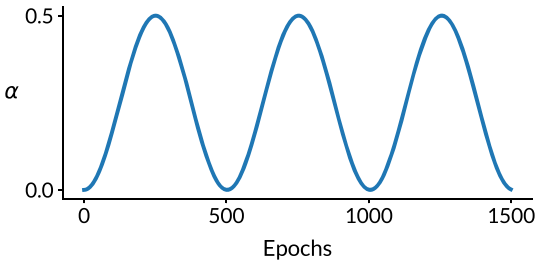



**Fig. 3** Cyclic scheduling of $\alpha$, the trade-off parameter for the classification/compactness balance

classification performance and number of shapelets to retain. A too high value of $\alpha$ might lead the model to learn a very small set of shapelets, degrading the classification performance (at the limit, the model can disregard the $H$ term). Vice versa, setting $\alpha$ too low will lead the model to discard few shapelets (or none at all). While its optimal value could be cross-validated, in practice it is advised to fully explore both objectives. To do this, we propose a dynamic cyclic scheduling of the value of $\alpha$, similar to what has been proposed for the learning rate in Smith (2017): at each epoch the value of $\alpha$ is changed following the sinusoidal cycles shown in Fig. 3, with range in $[0, 0.5]$. This allows the model to start by favoring the classification task initially, and gradually increasing $\alpha$ to discard irrelevant shapelets, and again decreasing it to refine the selected shapelets for classification. This cycle is repeated to keep the balance between the two objectives of maximizing classification accuracy and minimizing the number of shapelets used.

## 3 Algorithm

The following sections describe the algorithm, with a discussion on the parameters initialization (most importantly, shapelets initialization) and a description of the complete training and evaluation framework. All the code written for this work is in Python, the Neural Network architecture was defined in Keras[1] using a Tensorflow[2] backend for gradient-based computations.

### 3.1 Parameters initialization

The algorithm requires the initialization of the network parameters: $\mathbf{W}$, $\mathbf{G}$ and $\mathbf{S}$. Weights $\mathbf{W}$ are initialized using Xavier uniform initialization (Glorot and Bengio 2010); the gating parameters $\mathbf{G}$ are initialized to small random values sampled from $\mathcal{N}(10^{-1}, 10^{-3})$. Note that this implies that $h_\theta(\mathbf{G}) \approx \mathbf{1}$ since $\theta = 100$. The choice for the initialization of the shapelets $\mathbf{S}$ clearly has a big impact on the model performance and the final learnt representation for the shapelets. To allow a fair comparison and mitigate this effect, we chose to initialize the shapelets as in Grabocka et al. (2014), using KMeans centroids on each channel.

[1] https://keras.io.

[2] https://tensorflow.com.

### 3.2 Training

The training phase of our approach is illustrated in Algorithm 1.

1 **Inputs**: Training data ($\mathbf{T}$, $\mathbf{y_T}$), validation data ($\mathbf{V}$,$\mathbf{y_V}$), maximum number of shapelets $K_{max}$, lengths of shapelets $L$, patience, maxEpochs
2 –
3 $S = initShapelets(K_{max},\ L)$
4 $i = 0$, $bestAcc = 0$, noImprovement $= 0$
5 $minShapelets = length(S)$
6 earlyStop $= FALSE$
7 –
8 **while** $i \leq maxEpochs \wedge \neg earlyStop$ **do**
9 $alpha = getAlpha(i)$
10 $forward\_pass(alpha)$
11 $valAcc = computeAccuracy(\mathbf{V}, \mathbf{y}_v)$
12 $numShapelets = computeNumberShapelets()$
13 –
14 **if** $(valAcc > bestAcc)\ OR\ (numShapelets < minShapelets\ AND\ valAcc == bestAcc)$ **then**
15 $bestAcc = valAcc$
16 $minShapelets = numShapelets$
17 $noImprovement = 0$
18 **else**
19 noImprovement+ $= 1$
20 **if** $noImprovement > patience$ **then**
21 earlyStop $= TRUE$
22 **end**
23 **end**
24 –
25 $backpropagate\_error()$
26 $S, G, W = update\_parameters()$
27 $i = i + 1$
28 **end**

**Algorithm 1:** Training algorithm of the proposed approach

The algorithm takes as input a training time-series dataset $\mathbf{T}$ with labels $\mathbf{y_T}$, a validation set $\mathbf{V}$ with labels $\mathbf{y_V}$, as well as the following parameters:

1. *maxEpochs*: the maximum number of training epochs;
2. *patience*: the amount of successive epochs with no improvement of the validation score that is tolerated before early stopping the training process;
3. $K_{max}$, $L$: the initial number of shapelets, and their lengths.

During each training epoch, the input data is fed in small batches into the network; the current value for $\alpha$ is computed according to the epoch number, as in Fig. 3, and a forward pass is run on the data (cf. lines 10 – 11). Afterwards, the prediction error is backpropagated through the network and weights, gating parameters and shapelets are updated using the computed gradients (cf. lines 27 – 28). The training can stop for two reasons:

– the maximum number $maxEpochs$ of iterations is reached;

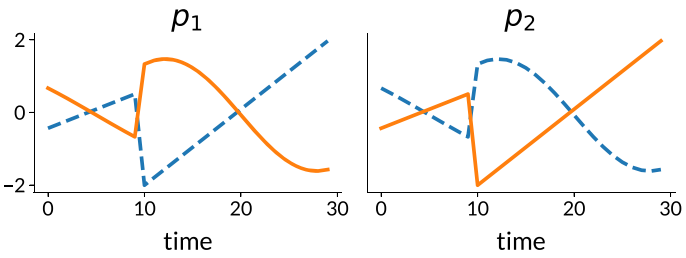



**Fig. 4** Synthetic dataset: the injected patterns $p_1$ and $p_2$

– there has been no improvement of the validation target metric for more than $patience$ epochs. Note that improvement is defined (cf. lines 15 – 25) as either (i) an increase in validation accuracy or (ii) a decrease in number of shapelets used, if the accuracy is kept the same. This is to ensure that a model that uses fewer shapelets must be preferred, accuracy being equal.

## 4 Experiments

We evaluated the proposed architecture as a whole, as well as its individual components (namely, introduction of non-linear boundary and automatic selection of shapelets). First, a proof-of-concept on a synthetic dataset is introduced to illustrate and validate the rationale and advantages of our proposed method. The subsequent subsections describe the experimental methodology and real-world datasets, while results can be found in Sect. 5. In all experiments, the input parameters for the training were set as follows: $maxEpochs = 5000$, batch size $B = 32$, $patience = 200$, $K_{max} = 2 * num\_classes$, $L = \{0.125,\ 0.25,\ 0.33,\ 0.4\}$ expressed as fraction of the input length $Q$. The model initializes $K_{max}$ shapelets for every length in $L$. Note that here $K_{max}$ and $L$ are not treated as hyper-parameters and cross-validated, but on purpose initialized to broad values, as the goal of the proposed model is to automatically select those during learning. The optimization task is solved using an Adam optimizer (Kingma and Ba 2014), with learning rate 0.001.

### 4.1 Synthetic dataset: XOR classification

To better illustrate and explain the advantages introduced by our method over existing techniques to learn shapelets, we applied it on a synthetic dataset. The dataset consists of a random bi-variate time-series, where two patterns $p_1$ and $p_2$ (Fig. 4) are injected, as in Fig. 5. Specifically, the dataset $D$ consists of 2000 samples $D_{i=1,\ldots,2000}$, labelled according to the following criteria:

$$(p_1 \in D_i) \oplus (p_2 \in D_i)$$

where $\oplus$ is the logical XOR, and $p_k \in D_i$ is true iff the pattern $p_k$ was injected in the $i_{th}$ time-series $D_i$.

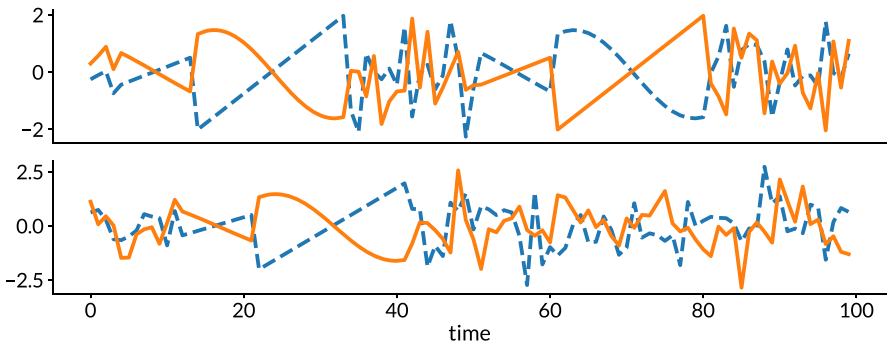



**Fig. 5** Synthetic dataset: two bi-variate time-series of different classes (top class 0, bottom class 1), where two patterns $p_1$ and $p_2$ are injected

If the model is able to learn shapelets similar to those (and only those) two patterns (Fig. 4), the classification problem using the distances to these would correspond to a XOR classification problem (cf. Fig. 6), which is known to be unsolvable for linear models (Minsky and Papert 2017). The goal is to investigate whether our approach is able to reduce the initial set of shapelets to the two patterns, as well as correctly classify each sample.

According to the values of $K_{max}$ and $L$ defined above, we ran our model with a total of $K = 16 \gg 2$ shapelets, as well its linear variant (i.e. with no hidden layers). This setting corresponds to most real-world scenarios, where the optimal number of shapelets is not known a priori. While a model using all $K$ shapelets should also find the patterns and achieve perfect accuracy, the shapelets will contain many irrelevant and redundant patterns, very similar to each other (cf. Fig. 7, bottom). Note that the linear model is also able to separate the classes but using more than two shapelets, as the problem could become linear in a higher-dimensional space. However, only the introduction of a non-linear boundary will guarantee that the desired patterns are the only ones retrieved. Figs. 7 and 8 show the gating parameters $h_\theta(\mathbf{G})$ over time and the shapelets found by running respectively the linear and non-linear variant of our method. Patterns are marked in green (red) if they are kept (discarded) by the model. We can observe how both variants manage to get rid of most of the redundant patterns, but using a non-linear classification boundary improves the reduction even further, while conserving the performance.

While this example is of course a simplification w.r.t real-world datasets, it illustrates the two key advantages of our proposed model, namely automatic selection of shapelets and the non-linear boundary for the classification.

Finally, Fig. 9 illustrates the evolution during training of the individual loss terms $H$, $C$ and $E$, as well the accuracy on the validation set, using the linear (top) and non-linear (bottom) model. The vertical dashed lines mark the epochs when the model early stops due to the plateauing of the validation accuracy. It can be observed how both models find quickly the patterns (accuracy rises to 1) and gradually drop the irrelevant or correlated patterns, as illustrated by the decrease over time of $C$ and $E$. However, the non-linear boundary allows to drop the third (redundant) shapelet, as

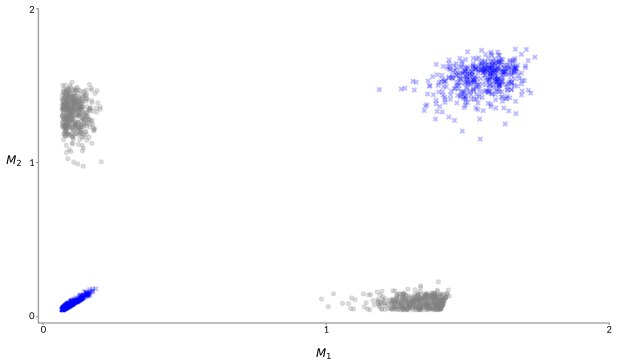



**Fig. 6** Synthetic dataset: distances $M_1$ and $M_2$ from the shapelets learnt with the non-linear model. As shown, separating the two classes corresponds to a XOR classification problem in 2D

shown by the steeper decrease of the correlation term $E$, and effectively retaining only the two desired patterns.

### 4.2 Datasets

To evaluate the model on different multivariate datasets, we selected a subset based on time-series length, number of channels and classes (cf. Table 1) from the multivariate collection described in Bagnall et al. (2018) and publicly available[3]. The chosen datasets are the following:

- *Libras*: this dataset contains 15 classes of 24 instances each, where each class references to a hand movement type in the Brazilian sign language. The hand movement is represented as a 2D curve performed by the hand in a period of time.
- *Epilepsy*: this datasets gather data from six participants conducting 4 different activities (walking, running, sawing, seizure mimicking) while wearing a tri-axial accelerometer on the wrist.
- *NATOPS*: this dataset contains 3D coordinates obtained from sensors on hands, elbows, wrists and thumbs, recorded while performing six kinds of gestures.
- *UWaveGestureLibrary*: this dataset contains 3D coordinates of simple gestures generated from accelerometers.
- *FingerMovements*: This dataset contains measurements of a normal subject sitting and pressing with the index and little fingers specific keys in a self-chosen order and timing.
- *SelfRegulationSCP1/SCP2*: This dataset contains measurements of the cortical potentials taken from a healthy/artificially respirated ALS patient, asked to move a cursor up and down on a computer screen.
- *ArticularyWordRecognition*: this datasets contains sensor data collected from native speaker pronouncing 25 words.

[3] https://timeseriesclassification.com.

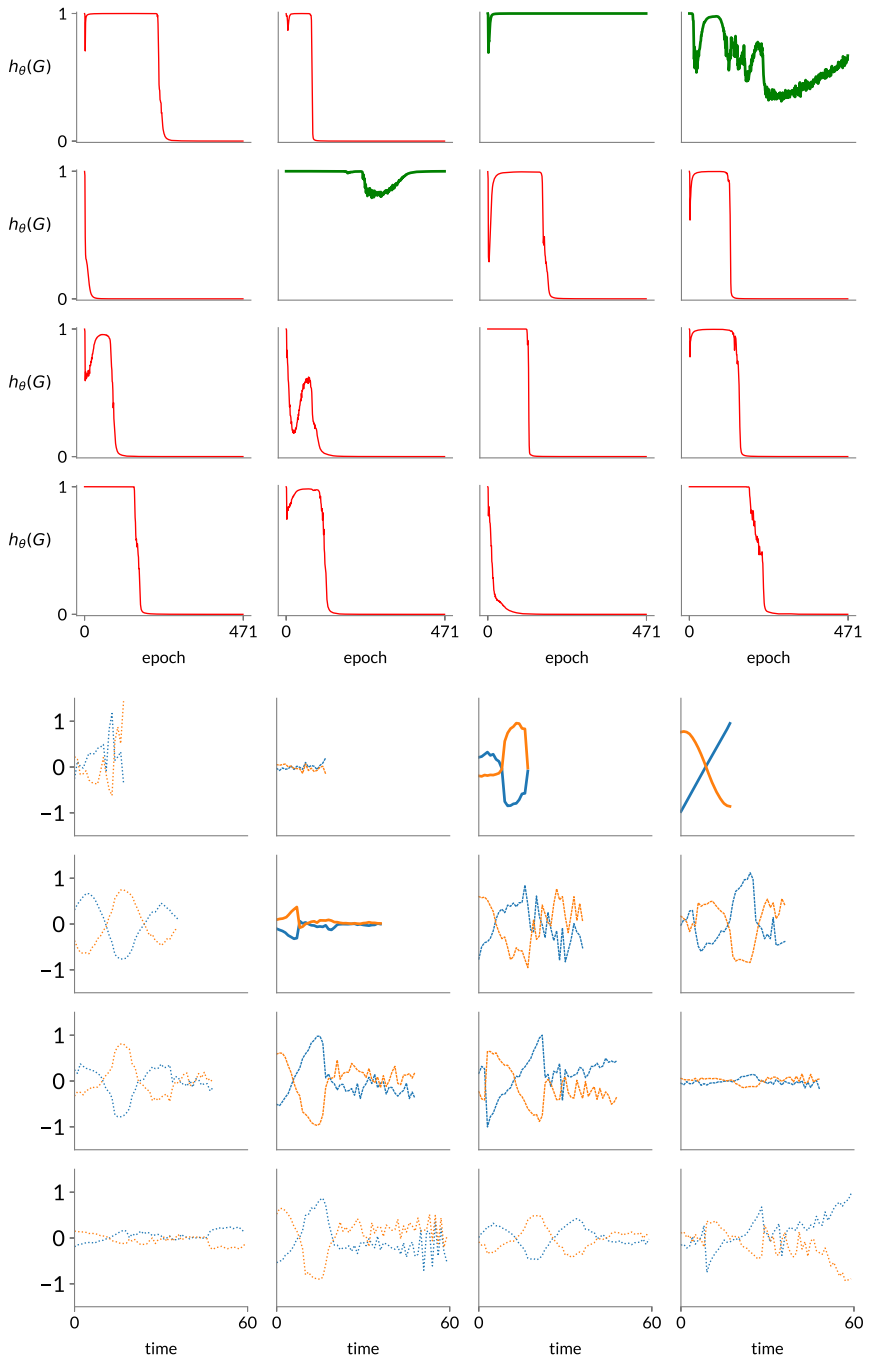



**Fig. 7** Synthetic dataset: gating parameters (top) and shapelets (bottom) for our model using a linear classification boundary. The curves in green/red (top) correspond to selected/discarded shapelets; below, the shapelets are drawn with solid/dashed lines accordingly (color figure online)

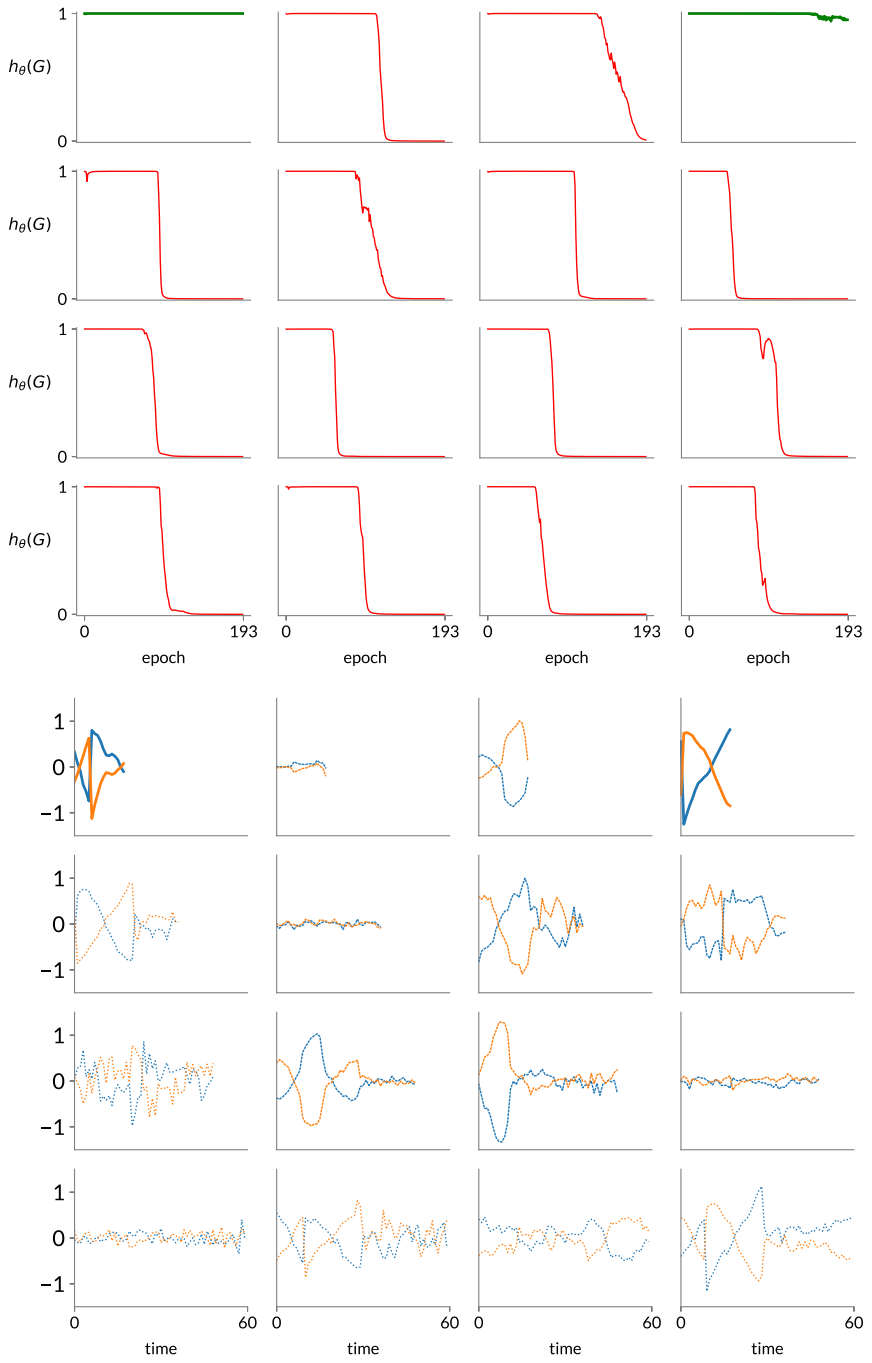



**Fig. 8** Synthetic dataset: gating parameters (top) and shapelets (bottom) for our model using a non-linear classification boundary. The curves in green/red (top) correspond to selected/discarded shapelets; below, the shapelets are drawn with solid/dashed lines accordingly (color figure online)

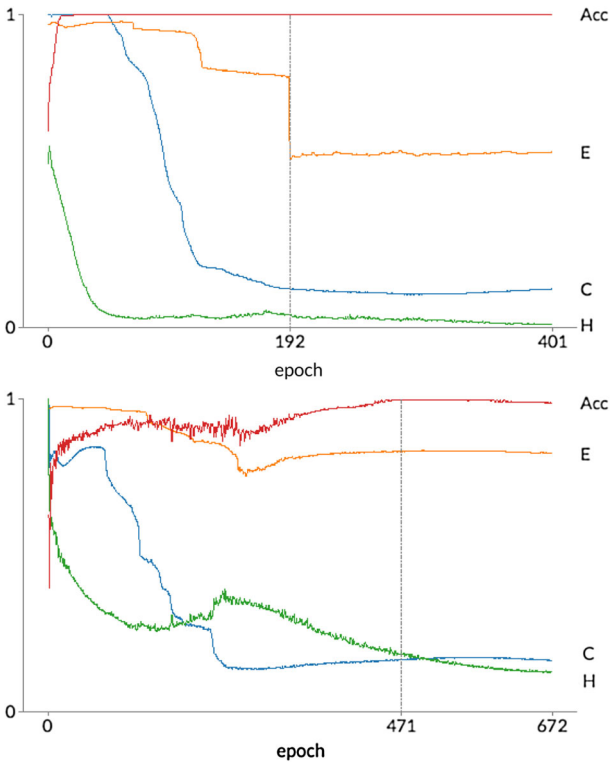



**Fig. 9** Synthetic dataset: evolution during training of the different terms in the loss (E: Correlation, C: Redundancy, H: Classification Loss), as well as the validation accuracy for the linear (top) and non-linear (bottom) model

– *RacketSports*: this dataset contains measurements (3D coordinates for gyroscope and accelerometer) from students wearing a smart watch and playing badminton or squash; the problem is to identify the sport played and the type of stroke made.
– *Handwriting*: this dataset contains measurements from a smart watch worn by the subjects while writing the 26 letters of the alphabet. 3D accelerometer readings (x, y, z) are provided.

The last column of Table 1 indicates whether the distribution of the classes for all datasets is balanced or not. For the imbalanced datasets, the classes distributions are shown in Fig. 10. Further details on the datasets can be found in Bagnall et al. (2018).

### 4.3 Preprocessing

Before being fed to the model, the datasets were preprocessed in the following way:

1. Each time-series (from each dimension) is normalized to $\mathcal{N}(0, 1)$;
2. The training data is randomly shuffled to guarantee diversity in the training batches.

**Table 1** Benchmark datasets: properties of the datasets used for evaluation, where $N$ is the number of time-series samples, $Q$ their length and $C$ the number of dimensions

| dataset | N | Q | C | classes | balanced |
|---|---|---|---|---|---|
| Libras | 360 | 45 | 2 | 15 | yes |
| Epilepsy | 275 | 206 | 3 | 4 | no |
| NATOPS | 360 | 51 | 24 | 6 | yes |
| UWaveGestureLibrary | 440 | 315 | 3 | 8 | yes |
| FingerMovements | 416 | 50 | 28 | 2 | yes |
| SelfRegulationSCP1 | 561 | 896 | 6 | 2 | yes |
| SelfRegulationSCP2 | 380 | 1152 | 7 | 2 | yes |
| ArticularyWordRecognition | 575 | 144 | 9 | 25 | yes |
| RacketSports | 303 | 30 | 6 | 4 | no |
| Handwriting | 1000 | 152 | 3 | 26 | no |

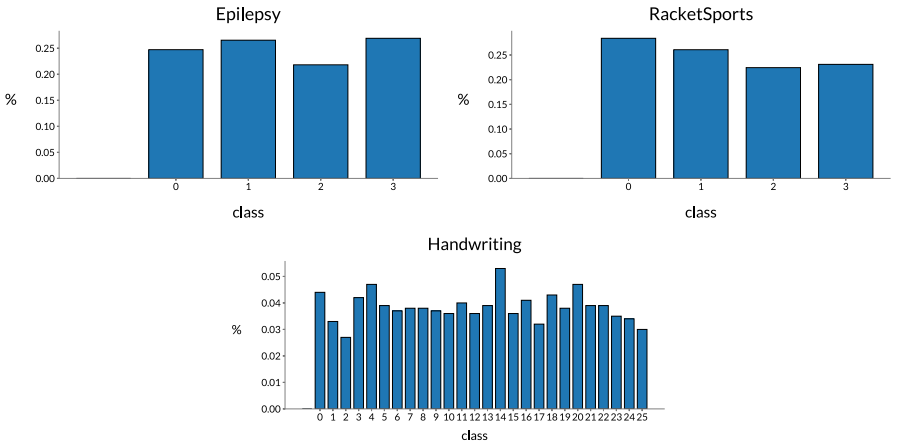



**Fig. 10** Benchmark datasets: class distribution for the imbalanced datasets

### 4.4 Validation approach

Each model/hyperparameters setting is validated on multiple resampling of the data; specifically, each dataset in Table 1 is split in 10 train/test set pairs with the following proportions:

1. Training: 85% of the total data (15% of this data is held-out and used for early stopping during training);
2. Test: 15% of the total data;

All splits are stratified, i.e. the class distribution is kept the same for training and test sets.

## 5 Results

This section discusses the results of the proposed methodology, in terms of both classification performance and shapelets interpretability. We carried out experiments both for well-established time-series classification approaches as well as an ablation study on the proposed approach. In details, the following approaches are compared:

1. 1-Nearest Neighbor under multi-variate Dynamic Time Warping ($DTW$), as implemented in Tavenard et al. (2017);
2. Time Series Forest ($TSF$), as proposed in Deng et al. (2013) and implemented in Löning et al. (2019);
3. Learning Time-Series Shapelets ($LTS_{full}$), as proposed in Grabocka et al. (2014) and adapted to the multivariate case;
4. Learning Time-Series Shapelets ($LTS_{red}$), as above but using the proposed automatic selection of shapelets;
5. Generalized Multivariate Shapelets Model ($GMSM_{full}$), the extension of $LTS$ using a non-linear classification boundary;
6. Generalized Multivariate Shapelets Model Reduced ($GMSM_{red}$), the proposed approach, using a non-linear classification boundary and embedded selection of shapelets.

Approaches 4, 5 and 6 are variants in the ablation study.

### 5.1 Performance

The results of all models compared across multiple datasets are shown in Table 2. The last rows show the average rank across datasets for each model, as well as the p-value resulting from a two-sided Wilcoxon signed rank-test (Wilcoxon 1992) between each classifier and our approach ($GMSM_{red}$). The p-values reported are computed using the obtained accuracy values for each fold, for each dataset (10 folds $\times$ 10 datasets). As shown, our approach $GMSM_{red}$ and its full variant $GMSM_{full}$ consistently match or improve the performance of the linear versions; moreover, our approach $GMSM_{red}$ outperforms all models in terms of average rank across datasets. Finally, the significance tests show how $GMSM_{red}$ is significantly different (p-value $< 0.1$) from all models except its full variant $GMSM_{full}$, showing that the selection of shapelets does not negatively impact the model.

These results show how our method is able to achieve competitive performance, while automatically selecting the number of shapelets needed. Indeed, Table 3 shows the effect of the selection process on the number of shapelets $K$, compared to the full scenario. The proposed approach learns a subset that is on average significantly smaller (around 60% across datasets) than $K$ for both linear and non-linear variant, while retaining if not improving (cf. Table 2) the performance.

**Table 2** Benchmark datasets: Average (and standard deviation) of the test accuracy for the different methods obtained by resampling the data 10 times

| Dataset | DTW | TSF | $LTS_{full}$ | $LTS_{red}$ | $GMSM_{full}$ | $GMSM_{red}$ |
|---|---|---|---|---|---|---|
| Libras | 0.839 (0.039) | 0.822 (0.030) | 0.844 (0.040) | 0.837 (0.053) | **0.850** (0.047) | **0.863** (0.038) |
| Epilepsy | 0.890 (0.038) | **0.929** (0.032) | 0.919 (0.060) | 0.914 (0.055) | 0.929 (0.051) | **0.931** (0.033) |
| NATOPS | 0.787 (0.046) | 0.772 (0.044) | 0.841 (0.044) | 0.852 (0.031) | **0.861** (0.028) | **0.857** (0.032) |
| UWaveGL | **0.961** (0.015) | 0.905 (0.019) | 0.917 (0.046) | 0.939 (0.032) | 0.932 (0.033) | **0.945** (0.034) |
| FingerMov | 0.500 (0.047) | **0.544** (0.042) | **0.535** (0.044) | 0.529 (0.055) | 0.535 (0.072) | 0.524 (0.060) |
| SelfRegSCP1 | 0.842 (0.029) | **0.860** (0.032) | 0.852 (0.054) | 0.844 (0.041) | 0.854 (0.037) | **0.861** (0.044) |
| SelfRegSCP2 | 0.507 (0.035) | **0.560** (0.042) | 0.500 (0.049) | 0.53 (0.052) | 0.540 (0.047) | **0.540** (0.031) |
| AWRecogn | **0.995** (0.006) | 0.969 (0.022) | 0.968 (0.009) | **0.983** (0.011) | 0.963 (0.016) | 0.975 (0.012) |
| RacketSp | **0.920** (0.034) | 0.887 (0.047) | 0.891 (0.055) | 0.878 (0.023) | 0.88 (0.053) | **0.891** (0.031) |
| Handwriting | **0.851** (0.028) | 0.644 (0.032) | **0.830** (0.022) | 0.821 (0.016) | 0.800 (0.031) | 0.812 (0.025) |
| Average Rank | 3.6 | 3.8 | 3.8 | 4.0 | 3.5 | **2.3** |
| p-value | 0.07603 | 0.00013 | 0.01685 | 0.05951 | 0.11383 | / |

At the bottom, average rank and results of a two-sided Wilcoxon signed rank-test (p-value) against $GMSM_{red}$ are reported. Top 2 methods per dataset are marked in bold; in case of a tie, the method with smaller standard deviation is marked

**Table 3** Benchmark datasets: reduction in the number of shapelets used after dynamic selection, for both linear (LTS) and non-linear (GMSM) variants

| dataset | K | $LTS_{red}$ | $GMSM_{red}$ |
|---|---|---|---|
| Libras | 120 | 0.66 | **0.53** |
| Epilepsy | 32 | 0.63 | **0.57** |
| NATOPS | 48 | **0.62** | 0.79 |
| UWaveGestureLibrary | 64 | 0.66 | **0.60** |
| FingerMovements | 16 | **0.59** | 0.82 |
| SelfRegulationSCP1 | 16 | **0.42** | 0.65 |
| SelfRegulationSCP2 | 16 | **0.76** | 0.81 |
| ArticularyWordRecognition | 200 | 0.57 | **0.37** |
| RacketSports | 32 | 0.61 | **0.51** |
| Handwriting | 208 | 0.86 | **0.73** |

Reduction is expressed as a fraction of the initial number of shapelets $K$
The model achieving the largest reduction are marked in bold

### 5.2 Interpretability

In addition to the competitive results w.r.t. classification performance shown in Table 2, a significant additional advantage of a shapelet-based classifier is its interpretability: since these discriminative patterns are learnt from the data, visualizing the shapelets can aid experts in interpreting the classifier's decisions as well as offer insights on the underlying data. While interpretability is not directly measurable or quantitatively comparable across different approaches, in this section we (i) show a few examples of the shapelets learnt by the model, to illustrate how they actually correspond to patterns in the data and (ii) show how the distances-from-shapelets are useful features for the classification, by visualizing a 2D manifold using T-SNE (Maaten and Hinton 2008).

A subset of shapelets learnt for the datasets *UWaveGL*, *Epilepsy* and *Articulary-WordRecognition* are shown in Figs. 11, 12 and 13 respectively. It can be observed how the shapelets closely match specific patterns in the data, and different lengths are able to capture different shorter/longer trends. Moreover, since the learnt shapelets are multivariate, it can be interesting to note how not all shapelets match closely each channel. As an example, the shapelets shown in Fig. 12 have a better match on a some channels (the first channel in the subset shown) than others, to show that the model can pick up discriminative behaviors also when those happen on a subset of the channels only.

Figure 14 shows a manifold learnt on the distances computed using the learnt shapelets. To be able to visualize these, we used T-SNE to project the distances form a high-dimensional space into a 2D plane. For a subset of the datasets, each plot highlights the samples belonging to a specific class. It can be observed that across datasets the embedding created by the distances groups sample from the same class together, in a limited number of tight clusters. This allows a non-linear model to separate the clusters in the higher dimensional space, thus obtaining good accuracy in the classification task.

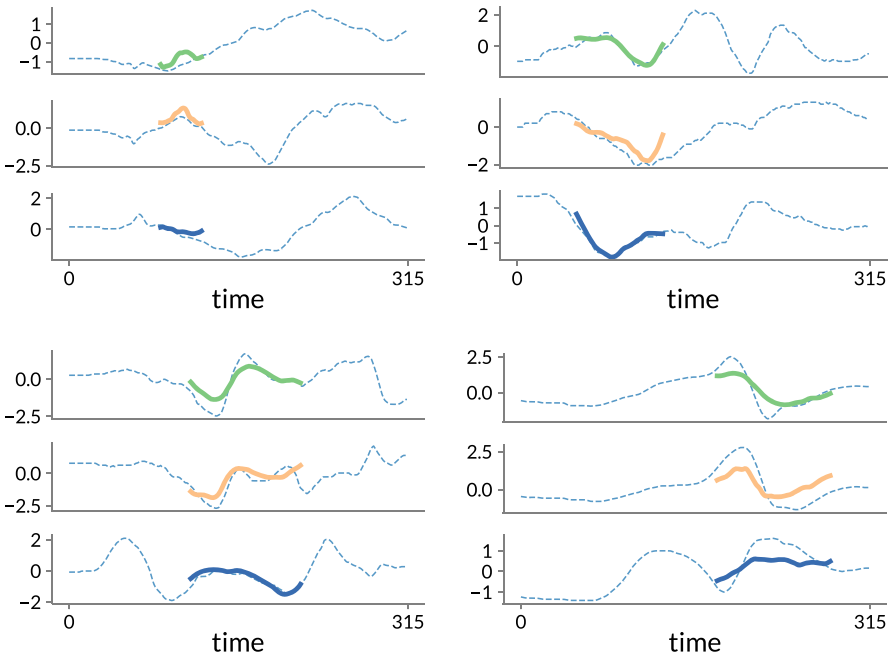



**Fig. 11** UWaveGestureLibrary dataset: 3D shapelets (subset) learnt on the dataset

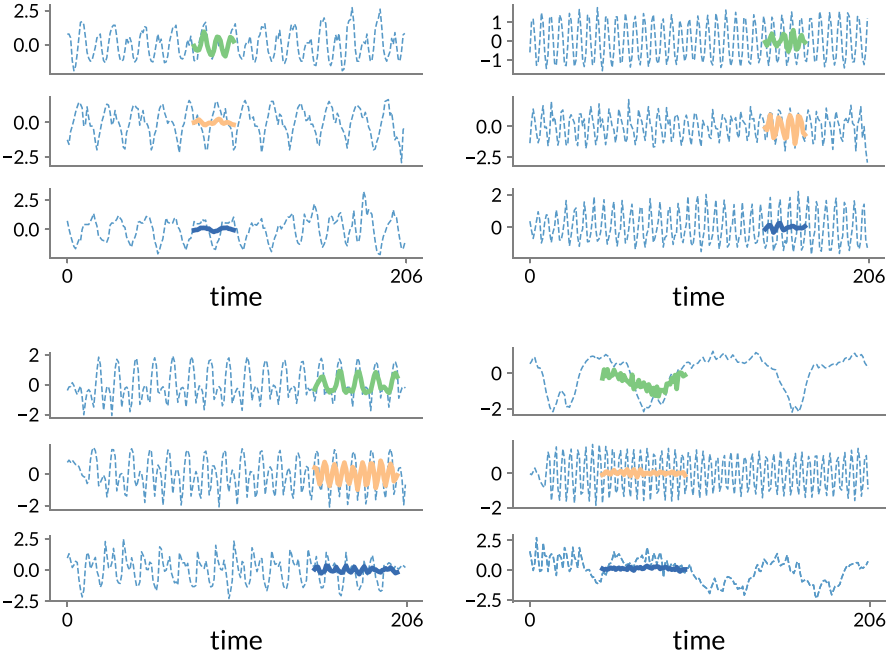



**Fig. 12** Epilepsy dataset: 3D shapelets (subset) learnt on the dataset

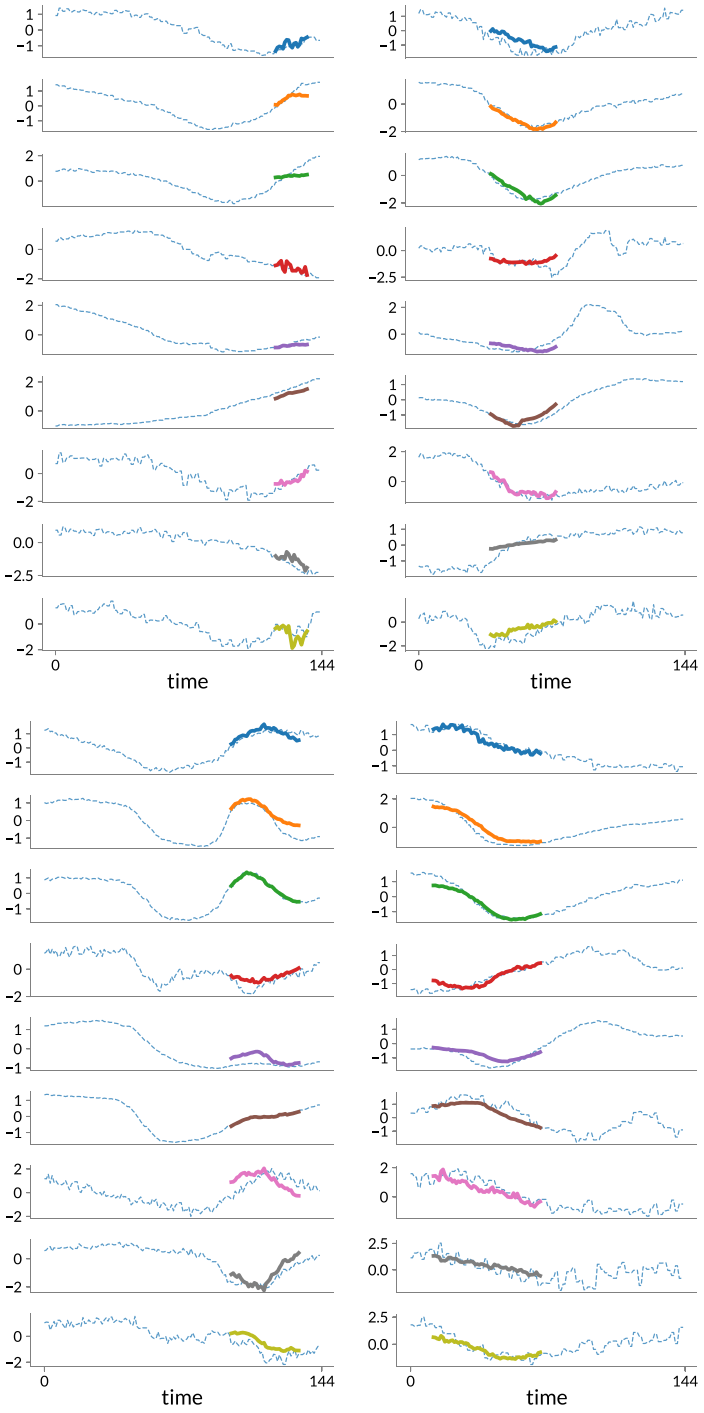
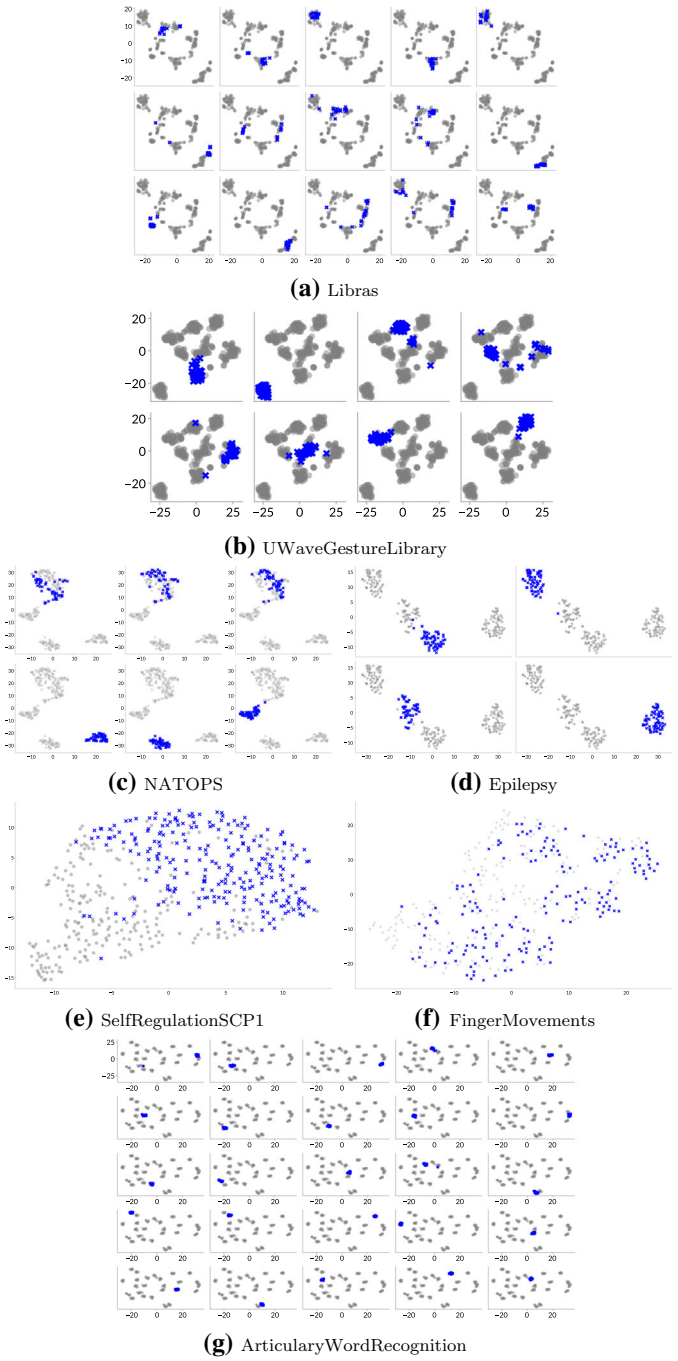



**Fig. 13** ArticularyWordRecognition dataset: 9D shapelets (subset) learnt on the dataset



**(a)** Libras

**(b)** UWaveGestureLibrary

**(c)** NATOPS

**(d)** Epilepsy

**(e)** SelfRegulationSCP1

**(f)** FingerMovements

**(g)** ArticularyWordRecognition

**Fig. 14** Benchmark datasets: T-SNE manifold on the distances obtained for some datasets. Samples belonging to a class are plotted as blue crosses, while others as gray circles. Across datasets, the distances often manage to isolate each class in a few (1 or 2) small clusters

Based on the observations above, the shapelets learnt by our approach have the properties to (i) match closely the input data, thus effectively representing realistic patterns and (ii) be effective features for time-series classification, using their distances to the input data. It follows that, once trained, the model can be used not only as a classification tool but also to uncover discriminative patterns, by visualizing the learnt shapelets; since these represent specific behaviors in the data, they are indeed immediately *interpretable* by a domain expert, who can match them with specific real events. Additionally, since the proposed approach learns *reduced* sets of shapelets, this investigation process is expedited.

It is worth noting that, while theoretically the model can find shapelets on datasets with any number of dimensions, the interpretability (and even basic visualization), understood as the ability of a domain expert to link shapelets to real events, gets more difficult as the dimensionality increases. Moreover, a hard limitation on the number of dimensions that the approach can handle is related to the memory of the underlying hardware used (e.g. the GPU): an increase of 1 dimension corresponds to a $K * L$-fold increase in the number of parameters to store in memory (as each shapelet has one more channel of dimension L).

## 6 Conclusions

This paper presents a novel approach for learning multivariate shapelets for classification tasks. It shows how a shapelets-based classifier can be generalized to learn any non-linear decision boundary, by embedding shapelets learning into the architecture of a Neural Network. In contrast to existing expensive discovery-based methods, this approach is ready-to-use for (and can benefit from) larger scale datasets, given the possibility to add more hidden layers and make the architecture deeper. In this work, we also proposed a modified learning objective, to allow the model to automatically identify a smaller number of uncorrelated shapelets. This also implies that no expensive hyper-parameters optimization is required for parameters such as number and length of shapelets, crucial for a shapelet-based model's performance. Results on ten benchmark datasets showed: (i) how the proposed approach achieves competitive performance against shapelet-based classifiers as well as state-of-the-art time-series classification models and (ii) how the number of shapelets used is dynamically reduced by approximately 40% (in average) using the proposed selection strategy, while preserving or improving the performance with the full set of shapelets. Finally, the shapelets can be visualized and used by a domain expert to uncover specific patterns in the data and match them to actual events.

**Acknowledgements** This research received funding from the Flemish Government (AI Research Program). We also would like to thank the authors in Bostrom and Bagnall (2017) for providing the datasets for evaluation.

## References


Bagnall A, Dau HA, Lines J, Flynn M, Large J, Bostrom A, Southam P, Keogh E (2018) The UEA multivariate time series classification archive, 2018. arXiv preprint arXiv:1811.00075

Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min Knowl Disc 31(3):606–660

Bostrom A, Bagnall A (2017) A shapelet transform for multivariate time series classification. arXiv preprint arXiv:1712.06428

Cetin MS, Mueen A, Calhoun VD (2015) Shapelet ensemble for multi-dimensional time series. In: Proceedings of the 2015 SIAM international conference on data mining. SIAM, pp 307–315

Deng H, Runger G, Tuv E, Vladimir M (2013) A time series forest for classification and feature extraction. Inf Sci 239:142–153

Gao X, Zhao Y, Dudziak Ł, Mullins R, Xu C-z (2018) Dynamic channel pruning: feature boosting and suppression. arXiv preprint arXiv:1810.05331

Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp 249–256

Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp 315–323

Grabocka J, Schilling N, Wistuba M, Schmidt-Thieme L (2014) Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 392–401

Hills J, Lines J, Baranauskas E, Mapp J, Bagnall A (2014) Classification of time series by shapelet transformation. Data Min Knowl Disc 28(4):851–881

Hua W, Zhou Y, De Sa CM, Zhang Z, Suh GE (2019) Channel gating neural networks. In: Proceedings of the 32nd advances in neural information processing systems (NeurIPS), pp 1884–1894

Karim F, Majumdar S, Darabi H, Harford S (2019) Multivariate LSTM-FCNS for time series classification. Neural Netw 116:237–245

Karlsson I, Papapetrou P, Boström H (2016) Generalized random shapelet forests. Data Min Knowl Disc 30(5):1053–1085

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980

Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing sax: a novel symbolic representation of time series. Data Min Knowl Disc 15(2):107–144

Lines J, Davis LM, Hills J, Bagnall A (2012) A shapelet transform for time series classification. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 289–297

Löning M, Bagnall A, Ganesh S, Kazakov V, Lines J, Király FJ (2019) sktime: a unified interface for machine learning with time series. In: Workshop on systems for ML at NeurIPS 2019

Lvd Maaten, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605

Minsky M, Papert SA (2017) Perceptrons: an introduction to computational geometry. MIT Press, Cambridge

Patri OP, Kannan R, Panangadan AV, Prasanna VK (2015) Multivariate time series classification using inter-leaved shapelets. In: NIPS 2015 time series workshop. NIPS

Rakthanmanon T, Keogh E (2013) Fast shapelets: a scalable algorithm for discovering time series shapelets. In: Proceedings of the 2013 SIAM international conference on data mining. SIAM, pp 668–676

Raychaudhuri DS, Grabocka J, Schmidt-Thieme L (2017) Channel masking for multivariate time series shapelets. arXiv preprint arXiv:1711.00812

Smith LN (2017) Cyclical learning rates for training neural networks. In: 2017 IEEE winter conference on applications of computer vision (WACV). IEEE, pp 464–472

Tavenard R, Faouzi J, Vandewiele G, Divo F, Androz G, Holtz C, Payne M, Yurchak R, Rußwurm M, Kolar K, Woods E (2017) tslearn: a machine learning toolkit dedicated to time-series data. https://github.com/rtavenar/tslearn. Accessed 1 Aug 2019

Wang H, Wu J (2017) Boosting for real-time multivariate time series classification. In: AAAI, pp 4999–5000

Wang L, Wang Z, Liu S (2016) An effective multivariate time series classification approach using echo state network and adaptive differential evolution algorithm. Expert Syst Appl 43:237–249

Wilcoxon F (1992) Individual comparisons by ranking methods. In: Breakthroughs in statistics. Springer, pp 196–202

Ye L, Keogh E (2009) Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 947–956