Embedding the Learning of Multivariate Shapelets in a Multi-Layer Neural Network

Roberto Medico
Ghent University - imec
Ghent, Belgium
roberto.medico@ugent.be

Joeri Ruyssinck
Ghent University - imec
Ghent, Belgium
joeri.ruyssinck@ugent.be

Dirk Deschrijver
Ghent University - imec
Ghent, Belgium
dirk.deschrijver@ugent.be

Tom Dhaene
Ghent University - imec
Ghent, Belgium
tom.dhaene@ugent.be

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 a novel initialization strategy for the shapelets, based on meaningful multidimensional motif discovery using the Matrix Profile, a recently proposed time series analysis tool. This paper describes the proposed architecture and presents results on seven publicly available benchmark datasets. Our results show how the proposed approach achieves competitive performance across the datasets, and, in contrast with the existing discovery-based methods, is applicable to larger-scale datasets. Moreover, the proposed motif-based initialization strategy helps the model convergence and performance, as well as improving interpretability of the learnt shapelets. Finally, the shapelets learnt during training can be extracted from the model and serve as meaningful insights on the classifier's decisions and the interactions between different dimensions.

CCS CONCEPTS

• **Information systems** → **Temporal data**; • **Computing methodologies** → **Neural networks**; *Supervised learning by classification*; • **Mathematics of computing** → *Time series analysis*;

KEYWORDS

Shapelets, Time-series Classification, Machine Learning, Neural Network

KDD, 2018, London, UK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Roberto Medico, Joeri Ruyssinck, Dirk Deschrijver, and Tom Dhaene. 2018. Embedding the Learning of Multivariate Shapelets in a Multi-Layer Neural Network. In *Proceedings of ACM KDD conference (KDD).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 machineries' 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 [1], and recently efforts have also been made to adapt these existing methodologies, or come up with novel approaches, for the multivariate case. 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 sub-sequences of time-series data. Shapelets were originally proposed in [20] as an innovative supervised motif discovery algorithm, where univariate shapelets are searched within a time-series exhaustively among all possible candidates (sub-sequences) 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 hyper-parameter 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 [17] with the FastShapelets algorithm, where the authors proposed to convert each time-series into its SAX representation [13], and perform the random search for shapelets in this new lower dimensional space. In [9] 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 [14].

All above methods were proposed for univariate time-series, but some research also investigated possible extensions to the multi-dimensional case. In [3], the authors propose an ensemble method of shapelet-based decision trees built independently on each univariate dimension. In [11], 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 [16], this representation is obtained by interleaving time-series subsequences extracted from multiple channels. In [2], the authors propose an extension of Shapelet Transform to the multivariate case by extracting multi-dimensional shapelets: (1) independently from each dimension, (2) dependently across dimensions, maintaining the phase and (3) 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 criteria (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 [8] 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, $\mathbf{W}, \mathbf{W_0}$ 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 [14]. 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 [19], 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 Section 2.2;
(2) The introduction of a novel *motif-based initialization* strategy, based on the computation of meaningful multivariate motifs with the multi-dimensional Matrix Profile [22], as described in Section 3.1;
(3) An evaluation *benchmark* (with source code available at [15]) of the proposed model on seven publicly available multivariate time-series datasets, presented in Section 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 2.1 (**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 2.2 (**Multivariate Shapelets Matrix**, MSM).* A multivariate shapelet matrix is a matrix consisting of K multivariate shapelets. If the length of each shapelet is defined as L, the shapelets matrix can be defined as $\mathbf{S}^{K \times L \times C}$.

*Definition 2.3 (**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:

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

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 *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 a custom layer (Distance Layer), responsible of computing the MDM as in (2), between input data and the associated trainable weights, corresponding to the shapelets. This Distance Layer is the first layer the input goes through in the network, and its outputs correspond to the values of the MDM, $\mathbf{M}$. The description of the model (and

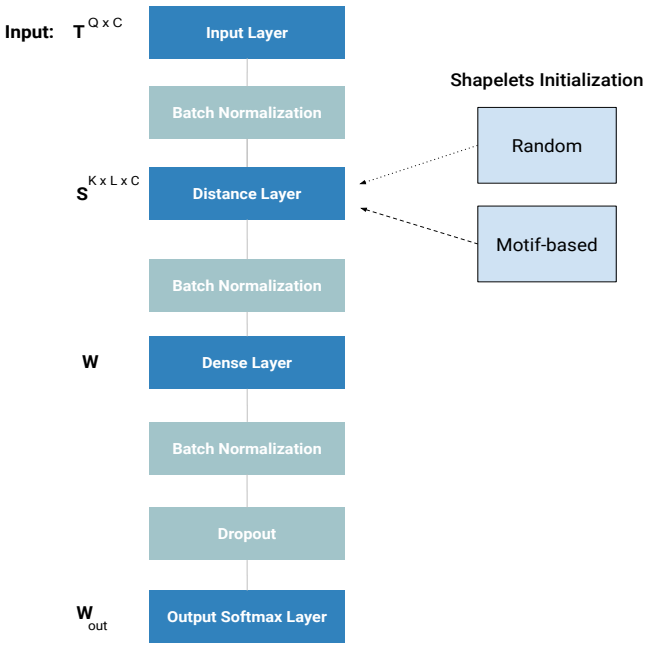


Figure 1: Architecture of the proposed multi-layer Neural Network, with 1 hidden layer. The parameter (or input) associated with each layer is shown on the left. The initialization strategy for shapelets is shown as an input for the Distance Layer.

algorithm) is generic for a classification task with P classes, with $P \geq 2$.

2.2.1 Layers. The network consists of five kind of layers:

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

$$M_j \geq 0 \quad \forall j,$$

therefore no activation is applied to the output.

(2) *Hidden Layer(s)*: one or multiple fully-connected layers with input $\mathbf{M}$, weights $\mathbf{W}$, bias $\mathbf{b}$ and output $\mathbf{Y}$, with ReLU activation [7]:

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

. where $\mathbf{Y}$ is the result of

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

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

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

is chosen as the predicted class.

(4) *Dropout layer*: This layer implements dropout [18] to add regularization and control overfitting of the network.

(5) *Batch Normalization layer*: This layer normalizes the activations of the preceding layer [10] to add regularization and help convergence. A Batch Normalization layer is also added after the Input Layer to ensure data is normalized to $N(0, 1)$.

2.2.2 Loss Function. The network is trained by optimizing the (binary or categorical) cross-entropy between predicted output $\hat{\mathbf{y}}$ and labels $\mathbf{y}$ defined as:

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

The optimization problem can therefore be formulated as:

$$\min_{\mathbf{S}, \mathbf{W}, \mathbf{W_{out}}, \mathbf{b}, \mathbf{b_{out}}} H_y(\hat{y}) \tag{3}$$

Note that the shapelets $\mathbf{S}$ appear in (3) and are therefore updated as network weights using gradients during back-propagation. In this work, the optimization problem in (3) is solved with an Adam [12] optimizer. A schematic of the overall architecture (using 1 hidden layer) is shown in Fig. 1. A Batch Normalization layer follows distance layer and hidden layer (Dense Layer). The hidden layer is also followed by a Dropout layer.

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. The code is presented here as pseudo-code for ease of readability, however we encourage reproducibility and make the complete code for the algorithm and all experiments publicly available at [15]. All the code written for this work is in Python, the Neural Network architecture was defined in Keras [4] and the optimization problem for training is solved using a Tensorflow [5] backend.

3.1 Parameters initialization

The algorithm requires the initialization of the network parameters: $\mathbf{W}$,$\mathbf{W_{out}}$, $\mathbf{b}$,$\mathbf{b_{out}}$ and $\mathbf{S}$. Weights $\mathbf{W}$ and $\mathbf{W_{out}}$ are initialized using Xavier uniform initialization [6] while biases $\mathbf{b}$ and $\mathbf{b_{out}}$ are initialized to 0.

The choice for the initialization of the weights representing the shapelets $\mathbf{S}$ clearly has a big impact on the model performance and the final learnt representation for the shapelets. The following sections illustrate two possible strategies for such initialization.

3.1.1 Random initialization. One immediate possibility is to initialize the shapelets by randomly sampling multivariate subsequences of the training data to be used as candidates, and letting the model update them during learning. This strategy has the advantages of requiring no additional computational cost and introducing diversity in the shapelets initialization. However, the network may

struggle to adapt these initial guesses into meaningful *discriminative* shapelets, focusing its updates more on the other trainable parameters.

3.1.2 Motif-based initialization. As an alternative way to initialize shapelets, we propose in this work a *motif-based* strategy that employs the Matrix Profile (MP), a time-series mining tool originally introduced for univariate data in [23] and recently extended to the multi-dimensional case in [22]. The Matrix Profile of a time-series T is another time-series obtained by retrieving the 1-Nearest Neighbor distance of every subsequence to each other, where the subsequence length is an input parameter. This tool can be employed for several use cases in time-series data analysis [23]. Indeed, following from the way it is constructed, the minima of the MP represent the top motifs in the time-series, while the maxima are its discords. In this work, we propose to use the multi-dimensional version of the MP to initialize the shapelets. As observed by the authors in [22], in most cases motifs spanning all dimensions of a multivariate time-series are not very informative, since one or more channels are likely to be irrelevant or noisy. Therefore, the *k-dimensional Matrix Profile* [22] is computed by using the *k-dimensional distance*, which is the distance computed using only k out of C dimensions, for a given number of dimensions (k). The mSTAMP algorithm ([21]) computes the k-dimensional MP for every combination of k dimensions out of C. For details on the algorithm's performance and correctness, the reader can refer to [22]. Note that this initialization strategy adds a very low computational cost to the algorithm, given the efficiency and speed of the MP implementation [22].

The pseudo-code of the algorithm for the initialization of the shapelets is illustrated in Algorithm 1. The main idea is to random sample B multivariate time-series from the training data for every class p. These samples are then processed to be used as input for the k-dimensional MP: corresponding channels of length Q for each sample $b \in \{1, \ldots, B\}$ are concatenated to form a $C \times (B \cdot L)$ dataset (referred to as *multi_sample* in the pseudo-code) which is then fed to the mSTAMP algorithm [22] which computes the k-dimensional MP for $k = 1, \ldots, C$. The top k-dimensional motifs for each k can then be retrieved by locating the minima of the MP, and be used as initialization for the multivariate shapelets. Note that this process initializes $P \times C$ shapelets, since for each class p a shapelet is extracted using the motif from each k-dimensional MP. When the number of shapelets K is set lower than $P \times C$, the algorithm picks a random k for each class, instead of using all the possible values of k (i.e. $k \in \{1, \ldots, C\}$).

For ease of presentation, in the pseudo-code we omit the check on the location of the motifs in the algorithm: since corresponding channels are concatenated, a motif is only valid if it does not overlap between two successive samples in the concatenated representation. If such a case occurs, then the motif is discarded and the second top-motif is used and so on, until the desired number of motifs is retrieved. Fig. 2 illustrates this process on three concatenated samples from the dataset *ArticularyWordUL* (one of the evaluation datasets, as described later): the motif shown as a dashed red subsequence (its nearest neighbor is also marked in the same way) is discarded, since it overlaps onto two different samples (separated by vertical dotted black lines). The second top-motif, marked as a

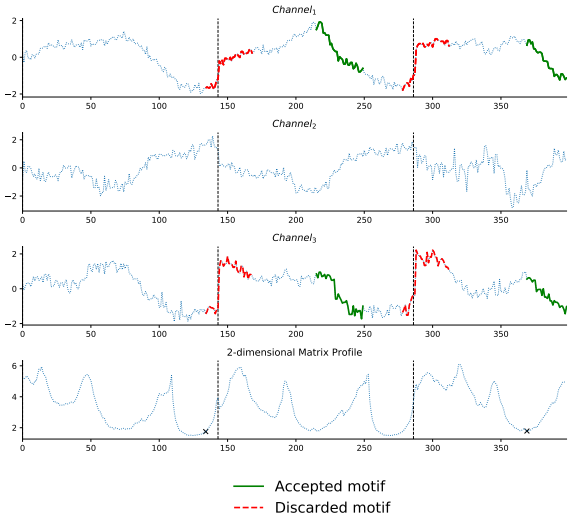


Figure 2: Multi-dimensional Matrix Profile-based shapelets initialization. Motifs overlapping two different samples (separated by a black vertical line) are discarded.

solid green line (as is its nearest neighbor), is used instead. Note that the motifs in this example are computed with a 2-dimensional Matrix Profile, therefore they are meaningful along two dimensions out of the three existing. The lower plot shows the 2-dimensional MP, where the starting locations for the motifs are marked with a black cross.

Note that we perform a random sampling in Algorithm 1 using a limited number of samples B (typically 10) since mSTAMP scales quadratically with the length of the time-series and we are only interested in retrieving an initial good approximation for the shapelets, whose shape will be refined by the learning algorithm. Also, the algorithm makes no checks on the similarity between motifs for different classes. It is important to observe that, in this step, we are only extracting motifs (i.e. most common patterns) from each class, rather than *discriminative* motifs: this means that very similar motifs could be chosen for different classes, and, in the worst case, most shapelets will be similar to each other if a common motif is present across all classes. However, in most cases this approach allows one to start with a sensitive initialization for the shapelets, since the motifs extracted with mSTAMP are *meaningful*, in the sense that each k-dimensional MP potentially represents a different level and depth of interactions between the channels and therefore its motifs capture a potentially relevant shapelet for classification.

3.2 Training

The training phase of the algorithm is illustrated in Algorithm 2.

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

(1) *EPOCHS*: the maximum number of training epochs

```
Inputs: T, B, L
forall class c in classes do
    multi_sample = empty()
    forall channel ch in channels do
        R = random_sample(T, B, c, ch)
        R = concatenate(R)
        multi_sample.add(R)
    end
    MP = mSTAMP(multi_sample, sub_len)
    while dimension < size(channels) do
        motif_location = MP[dimension].argmin()
        shapelet = empty()
        forall channel ch in channels do
            shapelet.add(multi_sample[ch, motif_location :
              motif_location + sub_len])
        end
        S.add(shapelet)
        dimension+ = 1
    end
end
```

Algorithm 1: Matrix Profile-based shapelets initialization

```
Inputs: T, y, V, PATIENCE, EPOCHS, init_strategy, L
if init_strategy is 'motif-based' then
    S = discover_shapelets(T, L)
else
    S = random_candidates(T, L)
i = 0, best_acc = 0, no_improvement = 0
EARLY_STOP = FALSE
while i ≤ EPOCHS ∧ ¬EARLY_STOP do
    forward_pass()
    val_acc = compute_accuracy(V)
    if val_acc > best_acc then
        best_acc = val_acc
    else
        no_improvement+ = 1
        if no_improvement > PATIENCE then
            EARLY_STOP = TRUE
        end
    end
    backpropagate_error()
    update_parameters()
    i+ = 1
end
```

Algorithm 2: Training algorithm

(2) *PATIENCE*: the amount of successive epochs with no decrease of the validation loss that is tolerated before early stopping the training process
(3) *init_strategy*: the initialization strategy, as discussed in Section 3.1
(4) *L*: length of the shapelets

During each training epoch, the input data is fed in small batches into the network; afterwards, the prediction error is backpropagated through the network and weights, biases and shapelets are updated using the computed gradients. To avoid overfitting, the training can stop for two reasons:

- The maximum number *EPOCHS* of iterations is reached
- There has been no decrease of the validation loss for more than *PATIENCE* epochs

These checks help to control overfitting on the training data, especially when the network becomes deeper, i.e. more hidden layers are used.

Table 1: 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
ArticularyWordLL	575	143	3	25
ArticularyWordT1	575	143	3	25
ArticularyWordUL	575	143	3	25
HandwritingGyroscope	1500	152	3	26
HandwritingAccelerometer	1000	151	3	26
uWaveGesture	440	314	3	8
ArabicDigit	8798	94	13	10

4 EXPERIMENTS

We evaluated the proposed architecture paying equal attention to classification performance (accuracy) and shapelets interpretability. The following subsections describe the experimental methodology and datasets, while results can be found in Section 5.

4.1 Datasets

Given the large amount of parameters to learn (shapelets and weights of the hidden layers), the approach proposed in this paper is better suited for larger datasets. However, not many *larger* multivariate time-series datasets are publicly available. Recently, in [2], the authors made available a collection of multivariate datasets with a uniform format collected from different public sources. Since the size of most of these datasets is below 100, we selected seven with the "largest" size out of the publicly available ones (with some exceptions, as discussed later): note that the sizes of these datasets are still not ideal for a deeper architecture, therefore the experiments are performed using only a limited amount of layers. However, one of the major strengths of the proposed model is that, once data is available, it can scale and grow deeper by adding hidden layers to the existing architecture. The datasets used for evaluation are the following: *ArticularyWordLL*, *ArticularyWordUL*, *ArticularyWordT1*, *uWaveGesture*, *HandwritingGyroscope*, *HandwritingAccelerometer* and *Arabic Digits*. Note that, despite the larger size, *PenDigits* was not chosen due to the very short time-series length (only 9 datapoints), that makes shapelets extraction meaningless. A similar observation is true for *JapaneseVowels* (where the length is 30). *PEMS* was not suitable for a straightforward application of this approach given the large amount of dimensions (144). The properties of the chosen datasets are summarized in Table 1. Notice that these may differ from those reported in [2], since there the authors report training set size only. However, in our experiments we used the full data instead of the pre-existing train/test splits.

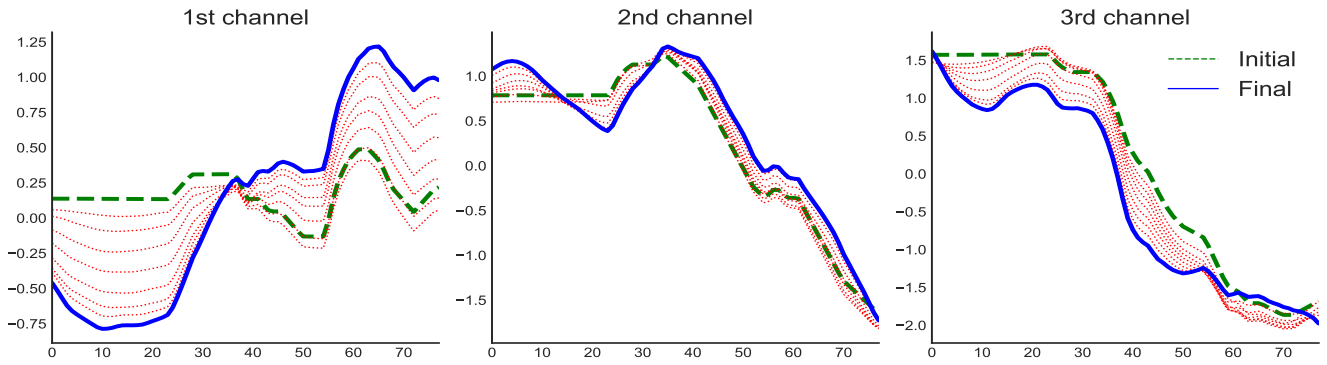


Figure 3: Learning process of a 3-dimensional shapelet for the *uWaveGesture* dataset. The shapelet was initialized with the motif-based strategy. The green dashed curve is the initial guess, while the solid blue curve is the final learnt shapelet. The learning updates are shown in dotted red curves, drawn every 50 training epochs.

4.2 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 $N(0, 1)$
(2) Small additive random noise, uniformly sampled from $[0, 0.1)$ is added to the data (to avoid constant subsequences, where the MP algorithm might fail)
(3) The training data is randomly shuffled to guarantee diversity in the training batches

4.3 Hyper-parameters search

The hyper-parameters to be optimized are:

(1) Shapelets initialization strategy: *random (R)* or *motif-based (MP)*
(2) Length of the shapelets: L
(3) Number of shapelets: K
(4) Number of hidden layers: hl
(5) L_2-regularization in the hidden and output layers: reg

As previously discussed, increasing the number of hidden layers hl results in a larger number of parameters. Given the relatively small size of the datasets, we chose to only employ 1 hidden layer, since the purpose of the experiments is to show the benefit of introducing non-linearity in the decision boundary, and not to offer an exhaustive benchmark of the performance. Note that when $hl = 0$ the optimization problem in (3) reduces to a linear classifier, using the minimum distances computed by the Distance Layer as features and $\mathbf{W_{out}}$ as weights. This allows for an indirect comparison with the multivariate extension of the original learning approach proposed in [19]. Note, however, that our implementation with no hidden layers still has significant differences with [19]: usage of batch normalization, Adam optimizer and initialization strategy are all novel contributions introduced in this work.

While for $hl \in \{0, 1\}$ and $init \in \{MP, R\}$ (where MP stands for MP-based initialization and R for random), all possible values are used, reg, L and K are optimized via grid search in the following spaces:

(1) $reg \in \{0.01, 0.1\}$
(2) $L \in \{0.15, 0.25\}$
(3) Number of shapelets $K \in \{2, 3\}$

where L is expressed as a fraction of the total length Q, and K is a multiplicative factor of the number of classes P.

The other parameters of the network were fixed as followed:

(1) $EPOCHS = 500$
(2) $PATIENCE = 25$
(3) Batch size $b = 64$
(4) Learning rate $lr = 10^{-3}$
(5) Dropout probability $dp = 0.10$
(6) Number of nodes in hidden layer(s): $2 \cdot P$

By an early inspection of the training process, a learning rate of 10^{-3} was found to be suitable across the datasets therefore no further optimization was made for this parameter.

4.4 Validation approach

Each dataset in Table 1 was split in train/validation/test with the following proportions:

(1) Training: 85 % of the total data
(2) Validation: 15 % of the training data
(3) Test: 15 % of the total data

Given the limited size of the datasets, to mitigate the effect of randomness, this split was repeated three times, and the reported performance of each model is the average across all splits. The splits are stratified, i.e. the samples' class distribution is kept the same for training, validation and test set.

5 RESULTS

This section discusses the results of the proposed methodology, in terms of both shapelets interpretability and classification performance.

5.1 Interpretability

One of the advantages 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

Table 2: Results of the grid search hyper-parameters optimization. For each dataset, the best performing models using either zero or one hidden layer are shown. Validation and test accuracy are averaged across the three splits.

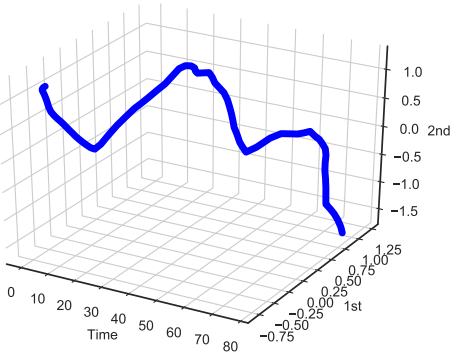
dataset	hl	init	reg	L	K	val_acc	test_acc
ArticularyWordUL	0	R	0.01	0.25	75	0.806 ± 0.02	0.816 ± 0.06
	1	MP	0.10	0.25	75	**0.811** ± 0.05	**0.820** ± 0.04
ArticularyWordT1	0	MP	0.01	0.15	50	0.964 ± 0.01	0.954 ± 0.03
	1	MP	0.10	0.25	50	**0.977** ± 0.03	**0.958** ± 0.02
ArticularyWordLL	0	R	0.01	0.25	75	0.923 ± 0.02	0.870 ± 0.04
	1	R	0.10	0.25	75	**0.937** ± 0.02	**0.904** ± 0.04
uWaveGesture	0	MP	0.10	0.25	24	**0.953** ± 0.02	**0.929** ± 0.02
	1	MP	0.01	0.25	24	0.942 ± 0.01	0.899 ± 0.05
HandwritingAccelerometer	0	MP	0.01	0.15	78	**0.849** ± 0.03	**0.816** ± 0.04
	1	R	0.10	0.25	52	0.812 ± 0.03	0.764 ± 0.03
HandwritingGyroscope	0	MP	0.1	0.15	78	**0.943** ± 0.01	**0.916** ± 0.02
	1	R	0.1	0.25	78	0.929 ± 0.01	0.911 ± 0.01
ArabicDigit	0	R	0.01	0.25	30	0.994 ± 0.005	0.994 ± 0.003
	1	R	0.10	0.15	30	**0.995** ± 0.002	**0.994** ± 0.003

classifier's decisions. In this framework, shapelets are multivariate time-series, hence one can opt to visualize them independently channel by channel, or the interactions between pairs of dimensions a 3D plane.

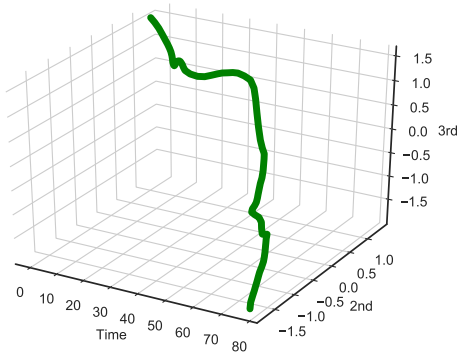
In Fig. 3 the learning process of one 3D shapelet for the *uWaveGesture* dataset is shown, where each subplot represents a different channel of the shapelet. The x-axis represents the time axis, while the y-axis is the value of the datapoints. The initial guess after initialization is shown as a dashed green line, while the final shapelet is shown as a solid blue curve. The learning process is illustrated with dotted red curves, where one curve is drawn every 50 epochs. In this case, the shapelet was initialized with the motif-based strategy. As illustrated by the plots, the learning process starts with the initial guess of the shapelet, and, during the learning process, the model can adapt and change each dimension of the shapelet more (as for the 1st channel) or less (2nd and 3rd channel) significantly. Since the learnt shapelets are multivariate, it is interesting to obtain a higher-dimensional visualization of how the various channels interact over time. Fig. 4 shows the same shapelet as Fig. 3, highlighting the multi-dimensional aspect of it: since the shapelets capture interactions between multiple channels, it is possible to visualize those on a 3D plane, by using the x-axis for the time dimension and y- and z-axis for two chosen channels. This visualization gives an overview on how different dimensions evolve and interact with each other over time. The first three plots (a), (b) and (c) show interactions between different channels, while (d) is a 3D visualization using all channels (*uWaveGesture* is a 3-dimensional time-series dataset). The fourth dimension (time) is in this case coded using color, changing from lighter to darker in intensity.

5.2 Performance

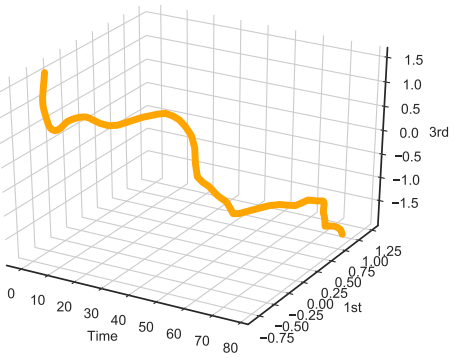
This section presents the results via hyper-parameters optimization on the chosen datasets. All the performance scores indicate the *accuracy* of the multiclass classifier. The results allow for a comparison between a linear (0 hidden layers) and non-linear (1+ hidden layers) classifier, as well as between the different initialization strategies for the shapelets. For each dataset, we report in Table 2 the results of best performing models (using either 0 or 1 hidden layer) on the validation set, and the relative test score. The best validation and test score between the two per dataset is marked in bold. Note that all scores are the average between the three train/validation/test splits as explained in Section 4. The introduction of a hidden layer (and consequently, allowing the model to learn non-linear decision boundaries) achieves competitive performance with the simpler linear version (improving performance in four out of seven cases), and the motif-based initialization (marked as MP) results in the top-performing model for five out of seven datasets. As pointed out in Section 4, the datasets used for evaluation are not ideal for the proposed approach, given the limited size. Additionally, the grid search for hyper-parameters optimization was carried out on a limited range. Indeed, the main purpose of this section is to display the applicability of the proposed approach (embedding shapelets learning in a Neural Network architecture) rather than an exhaustive benchmark. For a more complete and robust performance evaluation, more suitable datasets would be needed. Then the advantages of the proposed model (namely, non-linear decision boundary and the possibility to scale up by deepening the model) can be more meaningfully assessed. To easily visualize the impact of the initialization strategy, Fig. 5 shows an overview of the validation performance across different models (i.e. using different hyper-parameters): each boxplot represents the validation accuracy scores obtained by the best three combinations of parameters found via grid search for each initialization strategy. Each plot relates to one dataset, where the x-axis indicates the initialization strategy, while the y-axis the validation accuracy. As shown by the plots, the models using the proposed motif-based initialization strategy achieve in average better validation scores than a random initialization of the shapelets. Note that in the case of *ArabicDigit*, the better performance of a



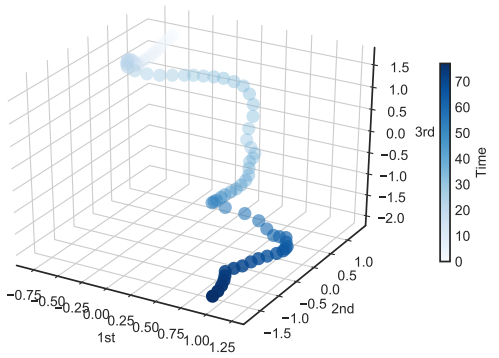
(a) Interaction between 1st and 2nd channel



(b) Interaction between 2nd and 3rd channel



(c) Interaction between 1st and 3rd channel



(d) Interaction between all channels

Figure 4: 3D visualizations of a multi-dimensional shapelet learnt on the *uWaveGesture* dataset. The curves in (a),(b) and (c) show the interaction between different channels over time, while (d) shows the interaction between all channels, with time coded as color intensity.

random initialization is possibly due to the very high similarity between different channels across samples: Fig. 6 shows the first two channels of one sample from class 1 (top) and one from class 10 (bottom). More than half of the time-series is almost constant across channels, and these regions are most likely to be picked by the motif-based strategy as initial shapelets. An ad-hoc preprocessing of this data (e.g. shortening or removing these constant regions) might help improving the MP initialization performance for this case.

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 investigated the impact of the initialization strategies for the multi-dimensional shapelets and proposed a novel motif-based initialization, to allow the learning to start from meaningful initial guesses, thus helping convergence and interpretability. This approach employs the multi-dimensional Matrix Profile [22] to discover top motifs on a small random sample of each class, and uses these as initial guesses for the shapelets, to be refined during the learning process. Our results on seven benchmark datasets showed: 1) how the proposed approach achieves competitive performance on publicly available datasets, by allowing non-linear classification boundaries, and 2) how the motif-based initialization strategy for the shapelets can improve

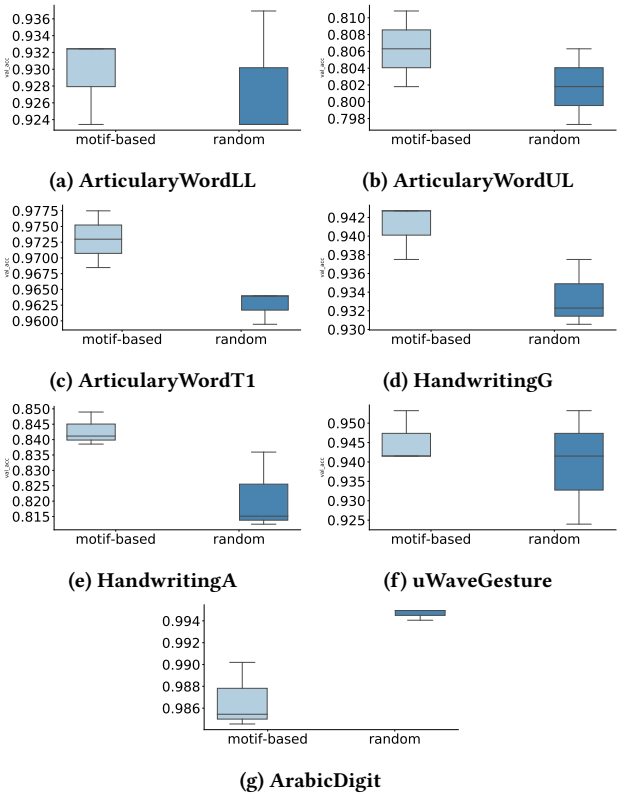


Figure 5: Validation performance on seven datasets using different initialization strategies. Each boxplot is computed using the scores of the top three performing models for each initialization strategy.

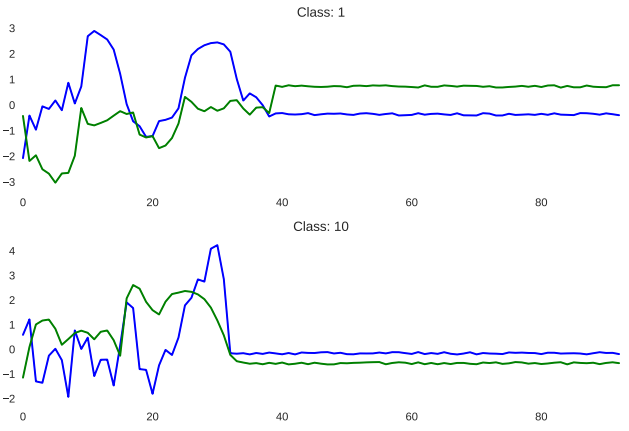


Figure 6: First two channels of two samples of different classes from the *ArabicDigit* dataset.

classification performance and interpretability, in comparison with a random initialization.

ACKNOWLEDGMENTS

This research was carried out in the framework of the FWO-SBO project "Hypermodeling strategies for Operational Optimization" (HYMOP). We also would like to thank the authors in [2] for providing the datasets for evaluation.

REFERENCES

[1] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 31, 3 (2017), 606–660.

[2] Aaron Bostrom and Anthony Bagnall. 2017. A Shapelet Transform for Multivariate Time Series Classification. (2017). arXiv:1712.06428

[3] Mustafa S Cetin, Abdullah Mueen, and Vince D Calhoun. 2015. Shapelet ensemble for multi-dimensional time series. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 307–315.

[4] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras. (2015).

[5] Martín Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). https://www.tensorflow.org/ Software available from tensorflow.org.

[6] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 249–256.

[7] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 315–323.

[8] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. 2014. Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 392–401.

[9] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. 2014. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery* 28, 4 (2014), 851–881.

[10] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. 448–456.

[11] Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. 2016. Generalized random shapelet forests. *Data mining and knowledge discovery* 30, 5 (2016), 1053–1085.

[12] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. (2014). arXiv:1412.6980

[13] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15, 2 (2007), 107–144.

[14] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. 2012. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 289–297.

[15] Roberto Medico. 2018. Multivariate Shapelets Learning. https://github.com/rob-med/multi_shapelets. (2018).

[16] Om Prasad Patri, Rajgopal Kannan, Anand V Panangadan, and Viktor K Prasanna. 2015. Multivariate time series classification using inter-leaved shapelets. In *NIPS 2015 Time Series Workshop. NIPS*.

[17] Thanawin Rakthanmanon and Eamonn Keogh. 2013. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 668–676.

[18] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15, 1 (2014), 1929–1958.

[19] Haishuai Wang and Jun Wu. 2017. Boosting for Real-Time Multivariate Time Series Classification.. In *AAAI*. 4999–5000.

[20] Lexiang Ye and Eamonn Keogh. 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, 947–956.

[21] Chin-Chia Michael Yeh. 2017. mSTAMP. https://github.com/mcyeh/mstamp. (2017).

[22] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. 2017. Matrix Profile VI: Meaningful Multidimensional Motif Discovery. In *2017 IEEE 17th International Conference on Data Mining (ICDM)*.

[23] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 1317–1322.