# Learning Interpretable Shapelets for Time Series Classification through Adversarial Regularization

Yichang Wang[1], Rémi Emonet[2], Elisa Fromont[1], Simon Malinowski[1], Etienne Menager[1], Loïc Mosser[1], and Romain Tavenard[3]

[1] *Univ Rennes, Inria, CNRS, IRISA, Rennes, France*
[2] *Laboratoire Hubert Curien UMR 5516, Univ Lyon, Saint-Etienne, France*
[3] *Univ Rennes, LETG, IRISA, Rennes, France*

### Abstract

Times series classification can be successfully tackled by jointly learning a shapelet-based representation of the series in the dataset and classifying the series according to this representation. However, although the learned shapelets are discriminative, they are not always similar to pieces of a real series in the dataset. This makes it difficult to interpret the decision, i.e. difficult to analyze if there are particular behaviors in a series that triggered the decision. In this paper, we make use of a simple convolutional network to tackle the time series classification task and we introduce an adversarial regularization to constrain the model to learn more interpretable shapelets. Our classification results on all the usual time series benchmarks are comparable with the results obtained by similar state-of-the-art algorithms but our adversarially regularized method learns shapelets that are, by design, interpretable.

## 1   Introduction

A time series (TS) $Z$ is a series of time-ordered values, $Z = \{z^{(1)}, z^{(2)}, \ldots, z^{(T)}\}$ where $z^{(t)} \in \mathbb{R}^d$, $T$ is the length of our time series and $d$ is the dimension of the feature vector describing each data point. If $d = 1$, $Z$ is said univariate, otherwise it is said multivariate. In this paper, we are interested in the Time Series Classification (TSC) task. We are given a training set $\mathcal{T} = \{(Z_1, y_1), \ldots, (Z_n, y_n)\}$, composed of $n$ time series $Z_i$ and their associated labels $y_i$ (target variable). Our aim is to learn a function $h$ such that $h(Z_i) = y_i$, in order to predict the labels of new incoming time series. The time series classification problem has been studied in countless applications (see for example [22]) ranging from stock exchange evolution, daily energy consumption, medical sensors, videos, etc.

Many methods have been developed to tackle this problem (see [2] for a review). One very successful category of methods consists in "finding" discriminative phase-independent subsequences, called *shapelets*, that can be used to classify the series. In the first papers about shapelet-based time series classification [26, 18], the shapelets were directly extracted from the training set and the selected shapelets could be used *a posteriori* to explain the classifier's decision. However, the shapelet enumeration and selection processes were either very costly or the selection was fast but did not yield good performance (as discussed in Section 2). Jointly learning a shapelet-based representation of the series in the dataset and classifying the series according to this representation [15, 11] allowed to obtain discriminative shapelets in a much more efficient way. An example of such a learned shapelet, obtained with the method from [11], is given in Figure 1 (top). However, if the learned shapelets are definitively discriminative, they are often different from actual pieces of a real series in the dataset. As such, the classification decision is difficult to interpret, i.e. it is difficult to determine what particular behavior in a time series triggered the classification decision. Note that the same interpretability issue arises with ensemble classifiers such as [3] where one decision depends on the presence of multiple shapelets. One
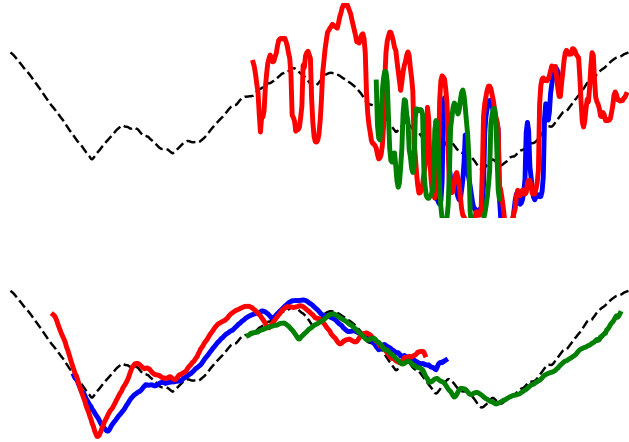
Figure 1: Example test time series and three most discriminative shapelets used for its classification for a baseline [11] (top) and for our proposed AI↔PR-CNN model (bottom) on the Herring classification problem.

of the main challenge nowadays is to enrich Machine Learning (ML) systems, and in particular black box models such as neural networks, so that they have the ability to explain their outputs to human users. In many scenarios, it may be risky, unacceptable, or simply illegal, to let artificial intelligent systems make decisions without any human supervision [12]. Hence, it is necessary for ML systems to provide an explanation of their decisions to all the humans concerned.

In this paper, we make use of a simple convolutional network to classify time series and we show how one can use adversarial techniques to regularize the parameters of this network such that it learns shapelets that could be more useful to interpret the classifier's decision. Section 2 presents the related work on time series classification, interpretability of models and adversarial training. We present our adversarial parameter regularization method in Section 3. In Section 4, we show quantitative and qualitative results on the usual time series benchmarks [4] that are both on par with state-of-the-art methods and very interesting to interpret the neural network predictions.

## 2 Related Work

In this section we review the literature on Time Series Classification (TSC), on tools for understanding black box model predictions and on adversarial training.

### 2.1 Time Series Classification

In the TSC literature, two main families of approaches have been designed. First, a dedicated metric can be used to compare the time series. In this case the decision is based on the resulting similarities. For example, [21] uses Dynamic Time Warping (DTW) to find an optimal alignment between time series and provides an alignment cost that can be used to assess the similarity. Another family of methods is based on the extraction of features in the time series. Among these works, shapelet-based classifiers have attracted a lot of attention from the research community.

Shapelets are discriminative subseries that can either be extracted from a set of time series or learned so as to minimize an objective function. They have been introduced in [26], in which a binary decision tree is built, whose nodes are shapelets and whose subtrees contain subsets of time series that contain or not that shapelet. In this work, shapelets are extracted from a training set of time series and building the decision tree requires to test all possible subseries from the training set, which makes the

method intractable for large-scale learning with an overall time complexity of $O(n^2 \cdot T^4)$ where $n$ is the number of training time series and $T$ is the average length of the time series in the training set. This high time complexity has led to the use of heuristics in order to select the shapelets more efficiently. In [18] (Fast Shapelets), the authors rely on quantized time series and random projections in order to fasten the shapelet search. Note however that these improvements in time complexity are obtained at the cost of a lower classification accuracy, as reported in [2]. The Shapelet Transform (ST) [15] consists in transforming time series into a feature vector whose coordinates represent distances between the time series and the shapelets selected beforehand. It hence needs to select a shapelet set (as in [26]) before transforming the time series. The resulting vectors are then given to a classifier in order to build the decision function. The training time complexity for ST is also in $O(n^2 \cdot T^4)$ [8], which makes it unfit for large scale learning.

In order to face the high complexity that comes with search-based methods, other strategies have been designed for shapelet selection. On the one hand, some attention has been paid to random sampling of shapelets from the training set [14]. On the other hand, Grabocka *et al.* [11] showed that shapelets could be learned using a gradient-descent-based optimization algorithm. The method, referred to as Learning Shapelets (LS) in the following, jointly learns the shapelets and the parameters of a logistic regression classifier. This makes the method very similar in spirit to a neural network with a single convolutional layer followed by a fully connected classification layer and where the convolution operation is replaced by a sliding-window local distance computation. A min-pooling aggregator should then be used for temporal aggregation.

Closely related to shapelet-based methods (as stated above), variants of Convolutional Neural Networks (CNN) have been introduced for the TSC task [25]. These are mostly mono-dimensional variants of CNN models developed in the Computer Vision field. Note however that most models are rather shallow, which is likely to be related to the moderate sizes of the benchmark datasets present in the UCR/UEA archive [4]. A review of these models can be found in [8].

Finally, ensemble-based methods, such as COTE [3] or HIVE-COTE [16], that rely on several of the above-presented standalone classifiers are now considered state-of-the-art for the TSC task. Note however that these methods tend to be computationally expensive, with high memory usage and difficult to interpret (as stated in Section 1) due to the combination of many different core classifiers.

In this paper, we propose a method that is scalable (compared to methods such as Shapelets [26] or ST [15]), yields interpretable results which can be used to explain the classifier's decision (compared to ensemble approaches or unconstrained approaches such as [11] or [16]), and exhibits good classification accuracy (compared to FS [18]).

## 2.2   Model Interpretability

Among the vast number of existing classifiers, some are easily interpretable (e.g. decision trees, classification rules), while others are difficult to interpret (e.g. ensemble methods, neural networks that can be considered as black-boxes). Interpretation of black box classifiers usually consists in designing an interpretation layer between the classifier and the human level. Two criteria refine the category of methods to interpret classifiers: global versus local explanations, and black-box dependent versus agnostic. In this category, state-of-the-art methods are Local Interpretable Model-agnostic Explanations (LIME and Anchors) [19, 20] and SHapley Additive exPlanations (SHAP) [17]. SHAP values come with the black-box local estimation advantages of LIME, but also with theoretical guarantees. A higher absolute SHAP value of an attribute compared to another means that it has a higher predictive or discriminative power.

In this paper, we are interested in making the decision of a neural network understandable. We follow the concept of interpretable shapelet as in [11]: for a TSC model, a simple explanation should not directly come from the vector of attributes describing each point of each time series but rather from some discriminative shapelets internally learned to produce an intermediate representation to classify the series. Solutions such as LIME, Anchors and SHAP which are not designed to inspect the internal representation of a model are thus not well suited for our problem.

Fang *et al.* [7] have a similar goal as ours (to produce interpretable discriminative shapelets) and build on both the works from [15] (in this case the candidate shapelets are extracted with a piecewise aggregate approximation) and from [11] to automatically refine the "handcrafted" shapelets. Contrarily to our method, there is no explicit constraint on the learning process that ensures the interpretability of the shapelets. Besides, their experimental validation makes it hard to fully grasp the benefits and limitations of the proposed method since the algorithm is evaluated on a small subset of UCR/UEA datasets [4] and they provide visualizations for only a couple of the learned shapelets.

## 2.3 Adversarial Training

Adversarial training of neural networks has been popularized by Generative Adversarial Networks (GANs) [10] and their numerous variants.[1] A GAN is a combination of two neural networks: a generator and a discriminator which compete against each other during the training process to reach an equilibrium where the discriminator cannot distinguish between the generator outputs and real training data. In a GAN, the adversarial network is used to push the generator towards producing data as similar to real data as possible. Other (non generative) adversarial training settings have been studied, for example in the context of domain adaptation [24]. In this case, the adversarial network is used to regularize the latent representation learned by the classifier such that it becomes domain-independent. The recent work from [27] also uses adversarial regularization to constrain the latent representation of an autoencoder to follow a given distribution.

In this paper, we propose an adversarial regularization approach which is unique as 1) we use a non-generative adversarial approach, 2) we do not work on a latent representation or on the output of a generator but rather on the CNN convolution filters (i.e., it is used as a parameter regularization), and, 3) we leverage this regularization to encourage interpretability, by making the convolution filters similar to real subseries from the training data.

# 3 Learning Interpretable Shapelets

In this section, we present our approach to learn interpretable discriminative shapelets for time series classification.

Our base time series *classifier* is a Convolutional Neural Network (CNN). As explained in Section 2, this model is very similar in spirit to the Learning Shapelet (LS) model presented in [11].

Both LS and CNN slide the shapelets on the series to compute local (dis)similarities. The main difference between the classifier of LS and that of our method is the (dis)similarity between a shapelet and a series. LS uses a squared Euclidean distance between a portion of the time series $Z$ starting at index $i$ and a shapelet $S$ of length $L$:

$$D(z_{i:i+L}, S) = \sum_{l=1}^{L} \left( z^{(i+l-1)} - S^{(l)} \right)^2. \tag{1}$$

The smaller this distance, the closer the shapelet is to the considered subseries. In a CNN, the feature map is obtained from a convolution, and hence encodes cross-correlation between a series and a shapelet:

$$D(z_{i:i+L}, S) = \sum_{l=1}^{L} z^{(i+l-1)} \cdot S^{(l)}. \tag{2}$$

Note that here, the higher $D(z_{i:i+L}, S)$, the more similar the shapelet is to the subseries.

As shown in Figure 2 (bottom), the convolutional layer of this classifier is made of three parallel convolutional blocks with shapelets of different lengths (red, green, blue) to be comparable with the

---

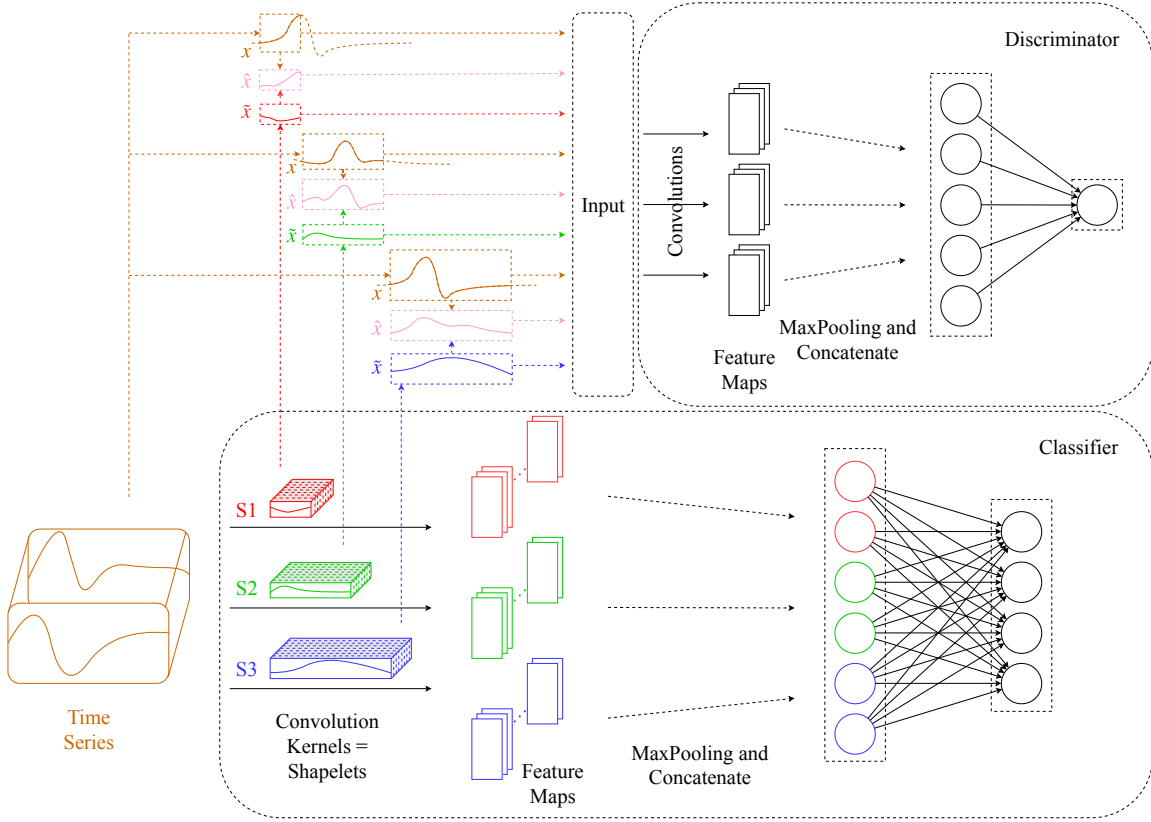[1]See `https://github.com/hindupuravinash/the-gan-zoo` for a list.

Figure 2: Architecture of our proposed Adversarially Input↔Parameter Regularized CNN (AI↔PR-CNN)

structure proposed in LS. We will loosely refer to the convolution filters of our classifier as *Shapelets* in the following.

Inspired by previous works on adversarial training (see e.g. Section 2), in addition to our CNN classifier, we make use of an adversarial neural network (the discriminator at the top of Figure 2) to regularize the convolution parameters of our classifier. This regularization acts as a soft constraint for the classifier to learn shapelets as similar to real pieces of the training time series as possible.

This novel regularization strategy is referred to, in the following, as Adversarial Input↔Parameter Regularization (AI↔PR) and the corresponding model is named AI↔PR-CNN.

Contrarily to GANs, our adversarial architecture does not rely on a generator to produce fake samples from a latent space. The AI↔PR strategy iteratively modifies the shapelets (i.e. the convolution filters of the classifier) such that they become close to subseries from the training set. To execute this strategy, the discriminator is trained to distinguish between real subseries from the training set and the shapelets. During the regularization phase, the discriminator updates the shapelets so that they become more and more similar to real subseries.

To obtain the best trade-off between the discriminative power of the shapelets (i.e. the final classification performance) and their interpretability, our training procedure alternates between training the discriminator and the classifier.

The type of data given as input to the discriminator is another major difference between a GAN and AI↔PR-CNN: in a GAN, the discriminator is fed with complete instances, while in AI↔PR-CNN, the discriminator takes subseries as input. These subseries can either be shapelets from the classifier model (denoted as $\tilde{x}$ in Figure 2), portions of training time series (denoted as $x$) or interpolations
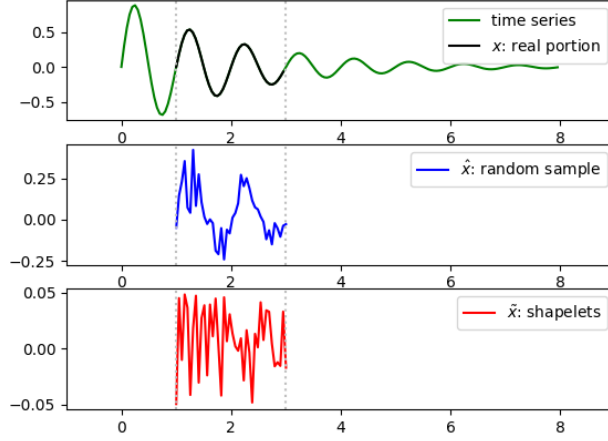
Figure 3: An example of samples provided as input to the discriminator.

between shapelets and training time series portions ($\hat{x}$, see the following section for more details on those), as illustrated in Figure 3. This process allows the discriminator to alter the shapelets for better interpretability.

## 3.1 Loss Function

As for GANs, our optimization process alternates between losses attached to the subparts of our AI↔PR-CNN model. Here, each training epoch consists of three main steps that are (i) optimizing the classifier parameters for correct classification, (ii) optimizing the discriminator parameters to better distinguish between real subseries and shapelets and (iii) optimizing shapelets to fool the discriminator. Each of these steps is attached to a loss function that we describe in the following.

Firstly, a multi-class cross entropy loss is used for the classifier. It is denoted by $L_c(\theta_c)$ where $\theta_c$ is the set of all classifier parameters.

Secondly, our discriminator is trained using a loss function derived from the Wasserstein GANs with Gradient Penalty (WGAN-GP) [13]:

$$L_d(\theta_d) = \mathop{\mathbb{E}}_{\tilde{x} \sim \mathbb{P}_S} [D(\tilde{x})] - \mathop{\mathbb{E}}_{x \sim \mathbb{P}_x} [D(x)]$$
$$+ \lambda \mathop{\mathbb{E}}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} \left[ (||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2 \right] \tag{3}$$

where $\mathbb{P}_S$ is the empirical distribution over the shapelets, $\mathbb{P}_x$ is the empirical distribution over the training subseries, and

$$\hat{x} = \epsilon x + (1 - \epsilon)\tilde{x}, \tag{4}$$

where $\epsilon$ is drawn uniformly at random from the interval $[0, 1]$ (cf. Figure 3) .

Thirdly, shapelets are updated to fool the discriminator by optimizing on the loss $L_r(\theta_s)$ where $\theta_s \subset \theta_c$ is the set of shapelet coefficients:

$$L_r(\theta_s) = - \mathop{\mathbb{E}}_{\tilde{x} \sim \mathbb{P}_S} [D(\tilde{x})]. \tag{5}$$

## 3.2 Learning Algorithm

Algorithm 1 presents the whole training procedure to update the parameters of our AI↔PR-CNN model. At each epoch of this algorithm, the three steps presented above are executed sequentially.

6

---

**Algorithm 1:** Learning Interpretable Shapelet

---

**Require:** number of shapelets $n_S$

**Require:** random initialization for the classifier/discriminator/shapelets $\theta_c, \theta_d, \theta_s \subset \theta_c$

**Require:** gradient penalty coefficient $\lambda$

**Require:** number of epochs $n_{\text{epochs}}$, mini-batch size $m$

**Require:** number of classifier/discriminator/regularization mini-batches per epoch $n_c, n_d, n_r$

**Require:** optimizer (Adam) hyperparameters $\alpha, \beta_1, \beta_2$

1  **for** $i = 1, \ldots, n_{epochs}$ **do**

2     **for** $t = 1, \ldots, n_c$ **do**

3        **for** $j = 1, \ldots, m$ **do**

4           Sample a pair $(Z_j, y_j)$ from the training set

5           $\hat{y}_j \leftarrow h_{\theta_c}(Z_j)$

6           $L_c^{(j)} \leftarrow \text{CrossEntropy}(y_j, \hat{y}_j)$

7        **end**

8        $\theta_c \leftarrow \text{Adam}(\nabla_{\theta_c} \frac{1}{m} \sum_{j=1}^{m} L_c^{(j)}, \theta_c, \alpha, \beta_1, \beta_2)$

9     **end**

10     **for** $t = 1, ..., n_d$ **do**

11        **for** $j = 1, ..., m$ **do**

12           Sample a shapelet $\tilde{x}_j$ from the set $\theta_s$, a subseries $x_j$ from the training set and a random number $\epsilon \sim U[0, 1]$

13           $\hat{x}_j \leftarrow \epsilon x_j + (1 - \epsilon)\tilde{x}_j$

14           $L_d^{(j)} \leftarrow D(\tilde{x}_j) - D(x_j) + \lambda(||\nabla_{\hat{x}} D(\hat{x}_j)||_2 - 1)^2$

15        **end**

16        $\theta_d \leftarrow \text{Adam}(\nabla_{\theta_d} \frac{1}{m} \sum_{j=1}^{m} L_d^{(j)}, \theta_d, \alpha, \beta_1, \beta_2)$

17     **end**

18     **for** $t = 1, \ldots, n_r$ **do**

19        **for** $j = 1, \ldots, n_S$ **do**

20           $\tilde{x}_j \leftarrow \theta_s[j]$

21           $L_r^{(j)} \leftarrow -D(\tilde{x}_j)$

22        **end**

23        $\theta_s \leftarrow \text{Adam}(\nabla_{\theta_s} \frac{1}{n_S} \sum_{j=1}^{n_S} L_r^{(j)}, \theta_s, \alpha, \beta_1, \beta_2)$

24     **end**

25 **end**

---

Note that in the second step (lines 10–17), sampling classifier shapelets, as well as sampling subseries from the training set, is performed uniformly at random.

# 4 Experiments

In this section, we will detail the training procedure for the AI↔PR-CNN and present both quantitative and qualitative experimental results.

## 4.1 Experimental Setting

As explained in Section 2, our most relevant competitor is Learning Shapelets (LS) from [11] as it also describes a shapelet-based model where the shapelets are learned and where a single model is used for classification. In the following sections, all the results presented for LS are retrieved from the UCR/UEA

(a) Wasserstein loss $L_d$      (b) Shapelet at epoch 20      (c) Shapelet at epoch 200

(d) Cross-entropy loss $L_c$      (e) Shapelet at epoch 800      (f) Shapelet at epoch 8000
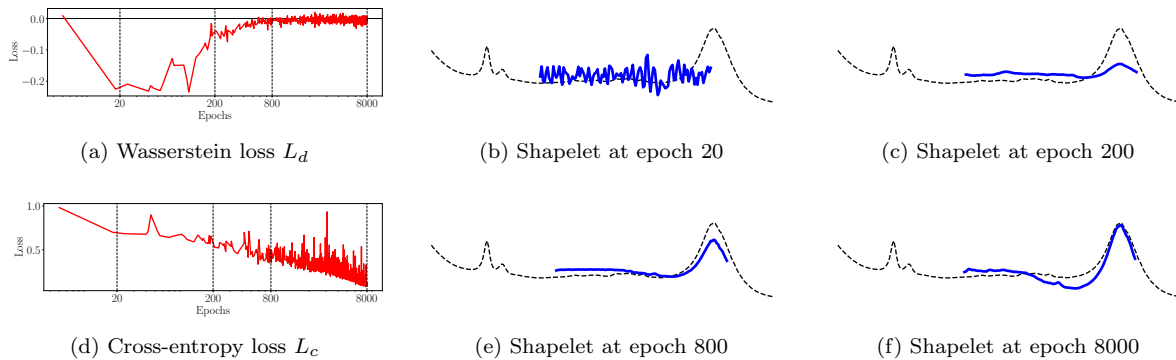
Figure 4: Illustration of the evolution of a shapelet during training (for the Wine dataset).

repository [4] and the shapelets presented for LS are obtained using the *tslearn* implementation [23].

### 4.1.1 Datasets

To compare our proposed method with [26, 18, 11], we use the 85 univariate time series datasets from the UCR/UEA repository for which all the baselines are available [4].[2] Note that our CNN-based method is not, by design, limited to univariate time series. However, for a fair comparison, we limited ourself to these datasets for this study. The datasets are significantly different from one to another, including seven types of data with various number of instances, lengths, and classes. The splits between training and test sets are provided in the repository.

### 4.1.2 Architecture details and parameter setting

We have implemented the AI↔PR-CNN model using TensorFlow [1] following the general architecture illustrated in Figure 2. The classifier is composed of one 1D convolution layer with ReLU activation, followed by a maxpooling layer along the temporal dimension and a fully connected layer with a softmax activation. The shapelets use a Glorot uniform initializer [9] while the other weights are initialized uniformly (using a fixed range). For each dataset, three different shapelet lengths are considered, inspired by the heuristic from [11] but without resorting to hyper-parameter search: we consider 3 groups of $20 \times n_{\text{classes}}$ shapelets of length $0.2T$, $0.4T$ and $0.6T$, where $n_{\text{classes}}$ is the number of classes in the dataset and $T$ is the length of the time series at stake.

The convolution filters of the classifier, i.e. the shapelets, are given as input to the discriminator which has the same structure as the classifier, but with shorter convolution filters (100 filters of size $0.06T$, $0.12T$ and $0.18T$) and a single-neuron $tanh$ activation instead of the softmax in the last layer. For optimization, we use Adam optimizer with a standard parametrization ($\alpha = 10^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$) and each epoch consists in $n_c = 15$ (resp. $n_d = 20$ and $n_r = 17$) mini-batches of optimization for the classifier loss (resp. discriminator and regularizer losses).

Experimental results are reported in terms of test accuracy and aggregated over five random initializations. All experiments are run for 8000 training epochs. The authors are devoted to the reproduciblility of the results.

## 4.2 Qualitative Results

Our method aims at producing interpretable results in the sense that shapelets should be similar to sub-parts of some series from the dataset. We first validate that our AI↔PR scheme actually ensures

---

[2]See `http://www.timeseriesclassification.com/singleTrainTest.csv` for all used datasets and baseline results.

(a) Learning Shapelets [11]

(b) AI↔PR-CNN
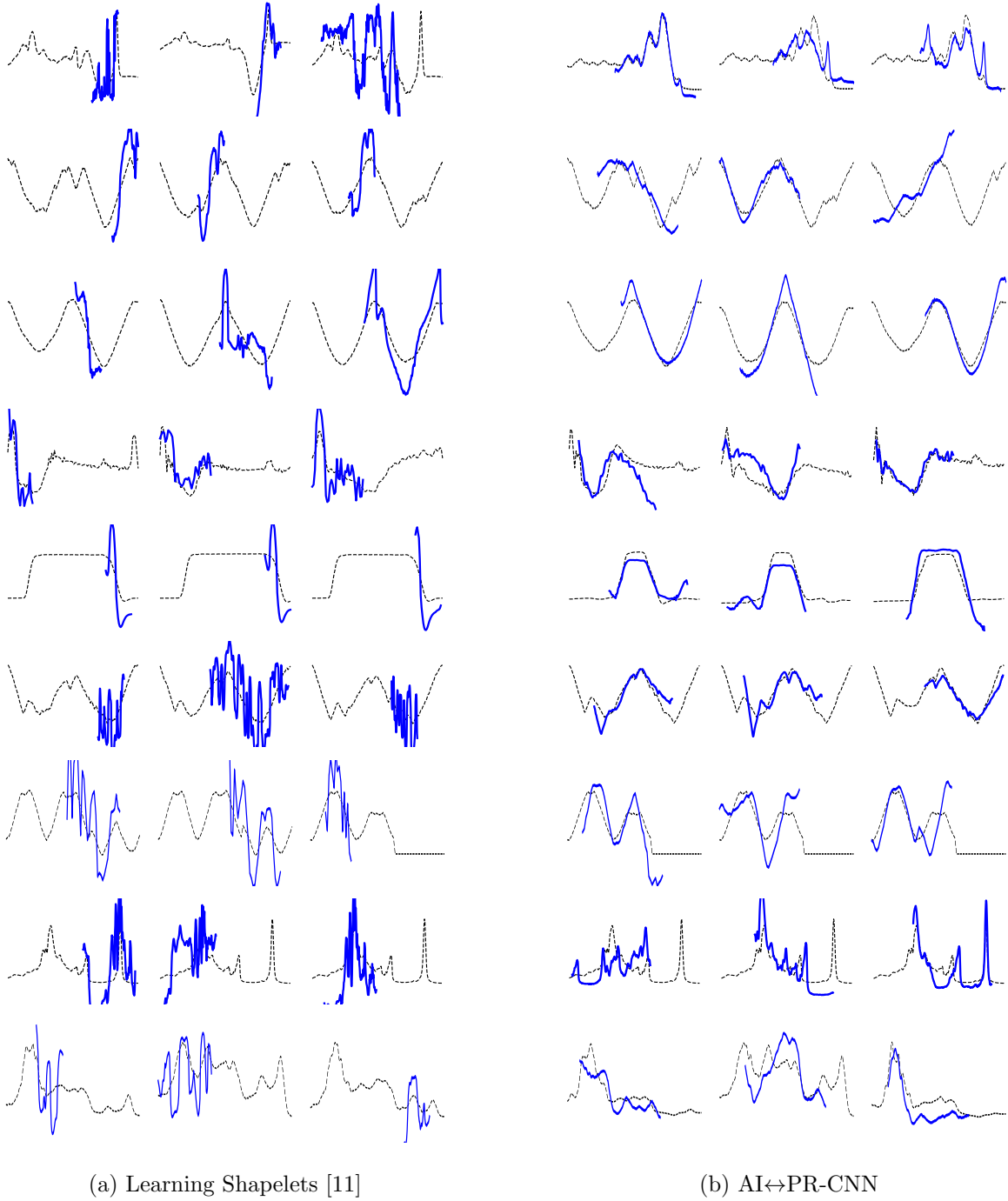
Figure 5: The three most discriminative shapelets obtained for the datasets Beef, Car, DiatomSizeReduction, ECG200, GunPoint, Herring, MiddlePhalanxOutlineCorrect, OliveOil and Strawberry (rows 1 to 9, respectively) using (a) Learning Shapelets or (b) our AI↔PR-CNN architecture. The average discriminative power of the shapelets is evaluated using Eq. 7 and each shapelet is superimposed over its best matching time series in the training set.
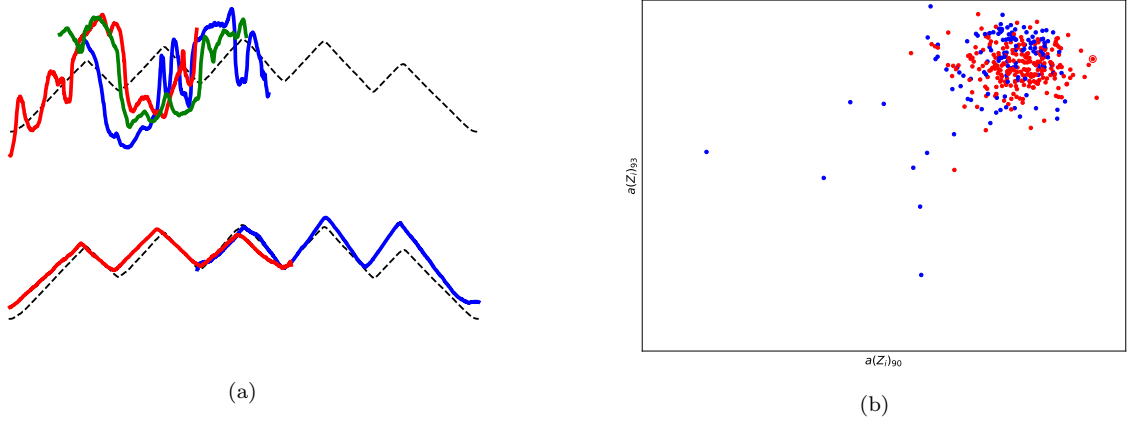
Figure 6: Explaining the decision for a test time series from the HandOutlines dataset. (a) The series together with the shapelets that were prominent for the classification decision. The results for LS are shown at the top (3 shapelets) and ours AI↔PR-CNN at the bottom (2 shapelets, see (b)). (b) All time series of the dataset shown in a 2D embedding using the activation values for the two shapelets from (a). The series shown in (a) is circled in red (top right corner) and belongs to the "red" class.

that shapelets are similar to the training data. Then we show how shapelets that look like subseries are helpful to make the decision process interpretable.

We first illustrate our training process and its impact on a single shapelet in Figure 4. In this figure, we show the evolution of a given shapelet for the Wine dataset at epochs 20, 200, 800 and 8,000. One can see from the loss values reported in Figures 4a and 4d that these correspond to different stages in our learning process. At epoch 20, the Wasserstein loss is far from the 0 value ($L_d = 0$ corresponds to a case where the discriminator cannot distinguish between shapelets and real subseries), and this indeed corresponds to a shapelet that looks very different from an actual subseries. As epochs go, both the Wasserstein loss $L_d$ and the cross-entropy one $L_c$ get closer to 0, leading to both realistic and discriminative shapelets.

To further check the effect of our regularization, we focus on the most discriminative shapelets for a bunch of datasets, as it would be misleading to look at a random shapelet: a shapelet might well be similar to a series but useless for the classification. The discriminative power, for class $j$, of the shapelet at index $s$ with respect to the $i$-th time series in the training set is evaluated as:

$$\mathcal{P}_{i,j}(s) = a(Z_i)_s \cdot w_{s,j} \tag{6}$$

where $a(Z_i)_s$ is the $s$-th component (i.e. the one that corresponds to shapelet $s$) of the activation map for the time series $Z_i$ and $w_{s,j}$ is the weight connecting that $s$-th component to the $j$-th output in the logistic layer of our classifier. As we aim at evaluating the overall discriminative power of a shapelet in a multi-class setting, and given that we use of a softmax activation at the input of our logistic layer, we can define the cross-class discriminative power of a shapelet as:

$$\mathcal{P}_i(s) = \sum_j \left( a(Z_i)_s \cdot (w_{s,j} - \max_{j'} w_{s,j'}) \right)^2 . \tag{7}$$

This is the criterion that we use to rank our shapelets in terms of discriminative power and to select the three most discriminative shapelets in Figure 5. This figure shows a significant improvement in terms of adequation of the shapelets to the training time series when using our AI↔PR-CNN model in place of a standard LS one. Examples of the shapelets learned using only the classifier part of our neural network architecture (a simple CNN) are shown in Figure 7. This figure reveals that an
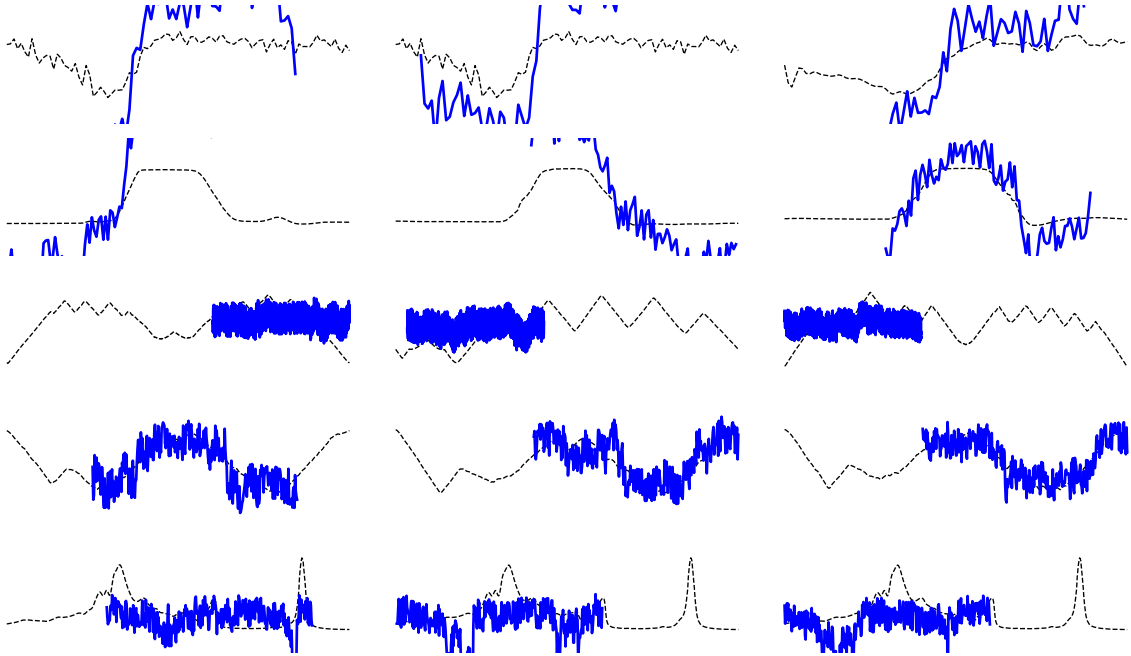
10

Figure 7: Shapelets obtained on the datasets (from top to bottom) ECG200, GunPoint, HandOutlines, Herring and OliveOil learned using only the bottom part of the architecture presented in Figure 2 (a simple CNN) superimposed over the best matching time series in the training set.

unregularized network fails at generating interpretable shapelets just as LS does. This shows that the actual benefit in interpretability indeed comes from our AI↔PR scheme. Our regularization strategy allows to generate shapelets that are both discriminative and representative of the training data.

Another important aspect, in terms of interpretability, is the explanation that can be provided to an end-user to explain a classification decision. For a given test time series, we produce two representations that help the user understand and trust the decision of a classifier. First, in Figure 1 and Figure 6a, we present the shapelets that were the most important to make a classification decision (according to Equation 7). One can notice that in both cases, shapelets extracted by AI↔PR-CNN better fit the time series at stake and hence help the end-user focus on the particular pattern in the time series that leads to the decision (e.g. the series of three peaks for HandOutlines or the overall shape of the central hump for Herring). Next, in Figure 6b, we present a 2D embedding of all the time series of the dataset, using the two most important shapelets for the considered time series. One can see that the considered time series (circled in red) lies in a part of the space where there are only "red class" time series. With these two representations for a test time series, the end-user: knows what are the most important shapelets (and their location) used by the model for its decision, and, can be convinced that these shapelets are good or sufficient to isolate the time series into a given class. When the considered time series correspond to actual subseries as with our method, this allows the end-user to better understand the decision process.

## 4.3   Quantitative Results

Our AI↔PR is able to recover shapelets that are discriminative and similar to the input, as expected. We want to quantify if this is achieved at the expense of classification accuracy and/or computation time. Our goal is to be much faster than exhaustive shapelet search methods (our baseline is Shapelets [26]), much more accurate than very fast random shapelet selection-based methods (our baseline is FS [18]) and as accurate and as fast as single model shapelet learning methods (our baseline is LS [11]).
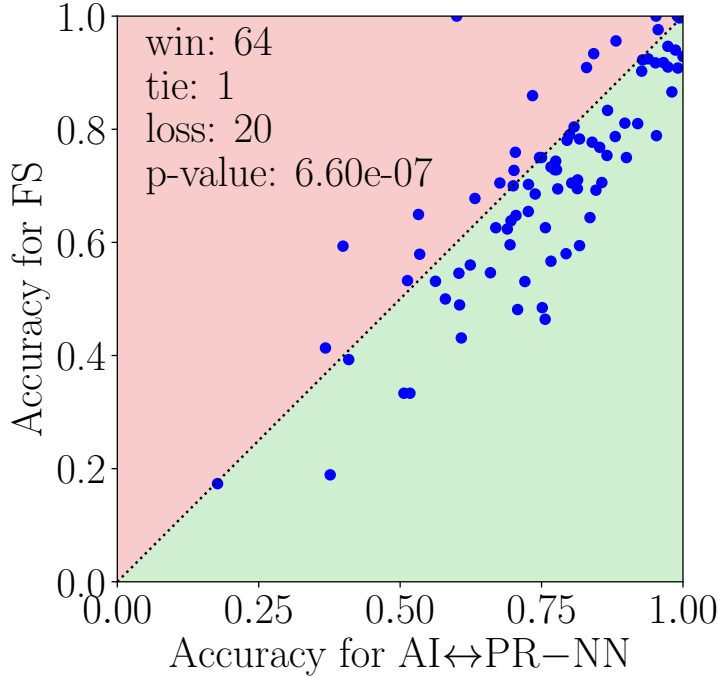
Figure 8: Accuracy comparison between Fast Shapelets and our AI↔PR method on 85 datasets (each point is a dataset) of the UCR/UEA repository [4].

### 4.3.1 Accuracy

We analyze the accuracies obtained by FS, LS and our AI↔PR-CNN method on the 85 datasets using scatter plots.[3] The results of the shapelet-based baselines used in this section come from [2] (the results for Shapelets [26] are not available because the method already does not scale on small size datasets). We compare FS versus AI↔PR-CNN in Figure 8 and LS versus AI↔PR-CNN in Figure 9. We also show how a simple CNN (without the adversarial regularization) compares against LS in Figure 10. We indicate the number of *win/tie/loss* for our method and we provide a Wilcoxon significance test [6] with the resulting $p$-value ($> 0.01$: none of the two methods is significantly better than the other). The points on the diagonal are datasets for which the accuracy is identical for both competitors. Figure 8 shows that, as expected, our method yields significantly better performance than FS. Compared to LS, for most datasets, the difference in accuracy is low, with a small edge (significant) for LS: on average for the 85 datasets, LS obtains an accuracy of 0.77 whereas AI↔PR-CNN obtains an accuracy 0.76. On three datasets (namely HandOutlines, NonInvasiveFetalECGThorax1 and OliveOil), our AI↔PR-CNN method and its regularization seems to be strongly positive (and detrimental on one dataset), in terms of generalization. The simple CNN seems to give slightly better (non significant) results than LS (and thus than our AI↔PR-CNN): on average for the 85 datasets, the simple CNN obtains an accuracy of 0.8. This means that our backbone neural network architecture is a good candidate to jointly learn interpretable shapelets and classify time series with little loss on accuracy.

### 4.3.2 Training Time

We provide both a theoretical complexity study (see Table 1) of all the baselines and of our AI↔PR-CNN method. Some complexities were already given in Section 2. Our method is based on a classifier and a

---

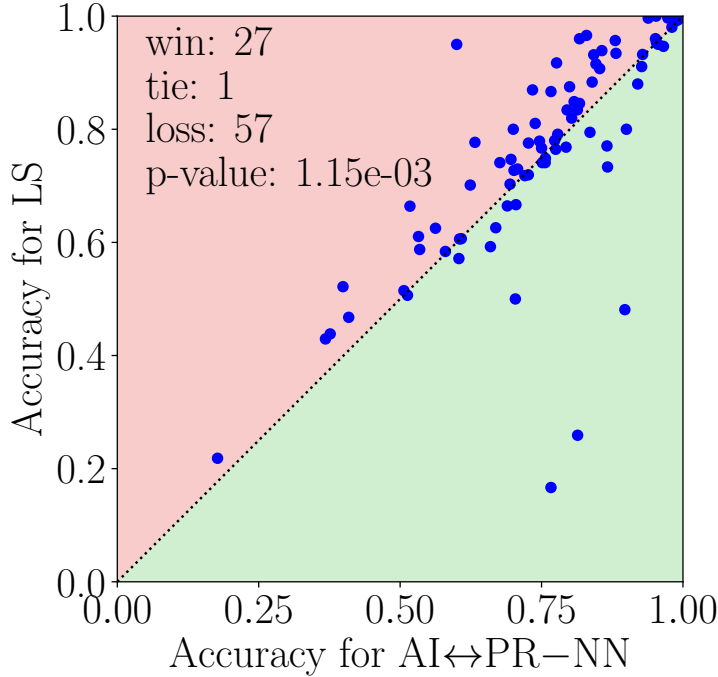[3]See Appendix A for detailed dataset information and accuracy.

Figure 9: Accuracy comparison between Learning Shapelets and our AI↔PR method on 85 datasets (each point is a dataset) of the UCR/UEA repository [4].

Table 1: Complexity of four different shapelet-based TSC algorithms (Shapelet [26], FS [18], LS [11] and AI↔PR-CNN). $n$ is the number of examples in the training set, $T$ is the average length of the time series, and $c$ is the number of classes.

| Shapelet | FS | LS | AI↔PR-CNN |
|---|---|---|---|
| $O(n^2 \cdot T^4)$ | $O(n \cdot T^2)$ | $O(n \cdot T^2 \cdot c)$ | $O(n \cdot T^2 \cdot c)$ |

discriminator, and both of them are simple CNNs. So the complexity of our algorithm ($O(n \cdot T^2 \cdot c)$) is related to training a CNN and should depend mainly on the number of examples ($n$), the average length of the time series ($T$), and the number of classes ($c$, since the latter is used to decide the number of shapelets to be learned). Note that for both LS and AI↔PR-CNN, the parameter $n$ could be considered as a (quite big) constant since the number of epochs (i.e. the number of times the algorithm "sees" the entire dataset) is fixed in the experiments. However, in LS, this number still depends on $n$ whereas it is fixed once and for all (to 8000) in AI↔PR-CNN. This difference is in favor of LS for small datasets and in favor of AI↔PR-CNN for larger ones.

To have a better grasp on the actual training time of all methods, we ran the methods on a single dataset (ElectricDevices) and recorded the CPU time. The experiments were conducted on a Debian Cluster using Intel(R) Xeon(R) CPU E5-2650 v4 Processor (12 core 2.20 GHz CPU) with 32GB memory. The results are averaged over five runs. The implementation code of our baselines is taken from [2] (as for the accuracy results). As expected, the original Shapelet [26] method does not finish in 48 hours for this medium size dataset. FS finishes in 12.1 minutes, LS finishes in 2323 minutes, and our method takes 142 minutes. The theoretical complexity of LS and AI↔PR-CNN is identical so these results were surprising. We suspected that the JAVA implementation of LS was not well optimized and we re-implemented the LS method with Keras[4]. With this new implementation, the training
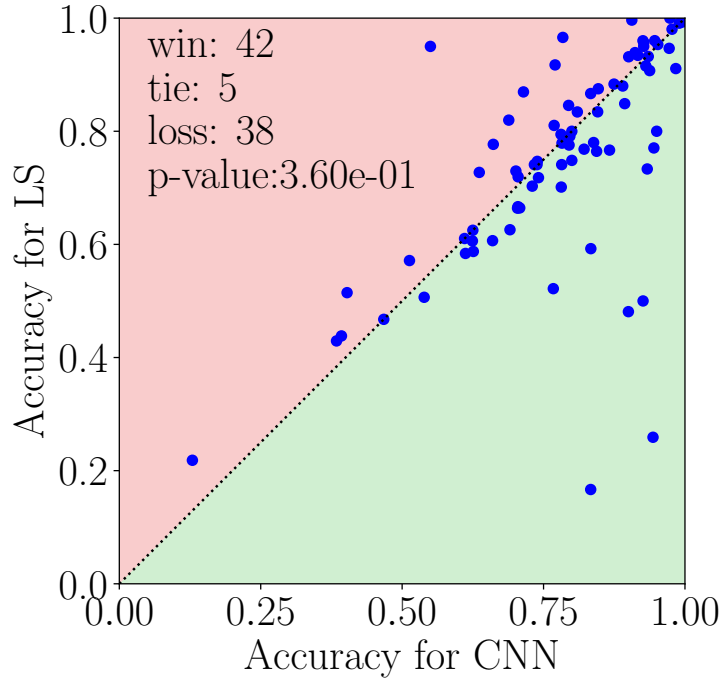
---
[4] https://keras.io/

Figure 10: Accuracy comparison between Learning Shapelets and a simple CNN on 85 datasets of the UCR/UEA repository [4].

phase took only 71 minutes for LS on this dataset (compared to 142 for AI↔PR-CNN) which shows that the time difference between the two algorithms is mainly related to the implementation (and the hyper-parameters related to the number of epochs).

# 5 Conclusion

We have presented a new shapelet-based time series classification method that produces interpretable shapelets. The shapelets are deemed interpretable because they are similar to pieces of a real series and can thus be used to explain a particular model prediction. The method is based on a novel adversarial architecture where one convolutional neural network is used to classify the series and another one is used to constrain the first network to learn interpretable shapelets. Our results show that the expected trade-off between accuracy and interpretability is satisfactory: our classification results are comparable with similar state-of-the-art methods while our shapelets are interpretable.

We believe that the proposed adversarial regularization method could be used in many more applications where the regularization should be put on the parameters instead of the latent representation of the networks as done, for example, with Generative Adversarial Networks.

In future work, we would like to first investigate the use of an additional regularization term, based on the group lasso [5], to be able to determine automatically a minimal set of necessary interpretable shapelets. We also want to use our regularization on other types of data (such as multivariate time series, spatial data, graphs) and in a deep(er) CNN. Furthermore, we would like to adapt this architecture for unsupervised anomaly detection in time series with interpretable clues using neural network architectures such as convolutional auto-encoders or generative networks.

# References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, May 2017.

[3] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

[4] A. Bagnall, J. Lines, W. Vickers, and E. Keogh. The uea & ucr time series classification repository. `www.timeseriesclassification.com`.

[5] K. Bascol, R. Emonet, E. Fromont, and J.-M. Odobez. Unsupervised Interpretable Pattern Discovery in Time Series Using Autoencoders. In A. Robles-Kelly, M. Loog, B. Biggio, F. Escolano, and R. Wilson, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 10029, pages 427–438. Springer International Publishing.

[6] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, Dec. 2006.

[7] Z. Fang, P. Wang, and W. Wang. Efficient learning interpretable shapelets for accurate time series classification. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 497–508. IEEE, 2018.

[8] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller. Deep learning for time series classification: a review. *ArXiv*, abs/1809.04356, 2018.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[11] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 392–401, 2014.

[12] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Survey*, 51(5), 2018.

[13] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[14] I. Karlsson, P. Papapetrou, and H. Bostrom. Generalized random shapelet forests. *Data Mining and Knowledge Discovery*, 30(5):1053–1085, Sep 2016.

[15] J. Lines, L. M. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 289–297, 2012.

[16] J. Lines, S. Taylor, and A. Bagnall. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5):52, 2018.

[17] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*, pages 4768–4777, 2017.

[18] T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. pages 668–676, 05 2013.

[19] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[20] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1527–1535, 2018.

[21] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

[22] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005.

[23] R. Tavenard. tslearn: A machine learning toolkit dedicated to time-series data, 2017. `https://github.com/rtavenar/tslearn`.

[24] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2962–2971, 2017.

[25] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1578–1585, 2017.

[26] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 947–956, 2009.

[27] J. Zhao, Y. Kim, K. Zhang, A. Rush, and Y. LeCun. Adversarially regularized autoencoders. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5902–5911, 2018.

# A Dataset information and accuracy comparison

| DatasetName | nb_train | nb_test | length | class | FS | LS | CNN | AI↔PR |
|---|---|---|---|---|---|---|---|---|
| Adiac | 390 | 391 | 176 | 37 | 0.5934 | 0.5217 | 0.7673 | 0.3990 |
| ArrowHead | 36 | 175 | 251 | 3 | 0.5943 | 0.8457 | 0.7943 | 0.8171 |
| Beef | 30 | 30 | 470 | 5 | 0.5667 | 0.8667 | 0.8333 | 0.7667 |
| BeetleFly | 20 | 20 | 512 | 2 | 0.7000 | 0.8000 | 0.8000 | 0.7000 |
| BirdChicken | 20 | 20 | 512 | 2 | 0.7500 | 0.8000 | 0.9500 | 0.9000 |
| Car | 60 | 60 | 577 | 4 | 0.7500 | 0.7667 | 0.8667 | 0.7500 |
| CBF | 30 | 900 | 128 | 3 | 0.9400 | 0.9911 | 0.9900 | 0.9867 |
| ChlorineConcentration | 467 | 3840 | 166 | 3 | 0.5464 | 0.5924 | 0.8336 | 0.6596 |
| CinCECGTorso | 40 | 1380 | 1639 | 4 | 0.8594 | 0.8696 | 0.7145 | 0.7341 |
| Coffee | 28 | 28 | 286 | 2 | 0.9286 | 1.0000 | 1.0000 | 1.0000 |
| Computers | 250 | 250 | 720 | 2 | 0.5000 | 0.5840 | 0.6120 | 0.5800 |
| CricketX | 390 | 390 | 300 | 12 | 0.4846 | 0.7410 | 0.7385 | 0.7513 |
| CricketY | 390 | 390 | 300 | 12 | 0.5308 | 0.7179 | 0.7410 | 0.7205 |
| CricketZ | 390 | 390 | 300 | 12 | 0.4641 | 0.7410 | 0.7821 | 0.7564 |
| DiatomSizeReduction | 16 | 306 | 345 | 4 | 0.8660 | 0.9804 | 0.9771 | 0.9804 |
| DistalPhalanxOutlineAgeGroup | 400 | 139 | 80 | 3 | 0.6547 | 0.7194 | 0.7050 | 0.7266 |
| DistalPhalanxOutlineCorrect | 600 | 276 | 80 | 2 | 0.7500 | 0.7790 | 0.7826 | 0.7464 |
| DistalPhalanxTW | 400 | 139 | 80 | 6 | 0.6259 | 0.6259 | 0.6906 | 0.6691 |
| Earthquakes | 322 | 139 | 512 | 2 | 0.7050 | 0.7410 | 0.7338 | 0.6763 |
| ECG200 | 100 | 100 | 96 | 2 | 0.8100 | 0.8800 | 0.8900 | 0.9200 |
| ECG5000 | 500 | 4500 | 140 | 5 | 0.9227 | 0.9322 | 0.9351 | 0.9287 |
| ECGFiveDays | 23 | 861 | 136 | 2 | 0.9977 | 1.0000 | 1.0000 | 0.9977 |
| ElectricDevices | 8926 | 7711 | 96 | 7 | 0.5790 | 0.5875 | 0.6259 | 0.5346 |
| FaceAll | 560 | 1690 | 131 | 14 | 0.6260 | 0.7485 | 0.8000 | 0.7568 |
| FaceFour | 24 | 88 | 350 | 4 | 0.9091 | 0.9659 | 0.7841 | 0.8295 |
| FacesUCR | 200 | 2050 | 131 | 14 | 0.7059 | 0.9390 | 0.9117 | 0.8566 |
| FiftyWords | 450 | 455 | 270 | 50 | 0.4813 | 0.7297 | 0.7011 | 0.7077 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Fish | 175 | 175 | 463 | 7 | 0.7829 | 0.9600 | 0.9257 | 0.8171 |
| FordA | 3601 | 1320 | 500 | 2 | 0.7871 | 0.9568 | 0.9273 | 0.8803 |
| FordB | 3636 | 810 | 500 | 2 | 0.7284 | 0.9173 | 0.7704 | 0.7765 |
| GunPoint | 50 | 150 | 150 | 2 | 0.9467 | 1.0000 | 0.9733 | 0.9733 |
| Ham | 109 | 105 | 431 | 2 | 0.6476 | 0.6667 | 0.7048 | 0.7048 |
| HandOutlines | 1000 | 370 | 2709 | 2 | 0.8108 | 0.4811 | 0.9000 | 0.8973 |
| Haptics | 155 | 308 | 1092 | 5 | 0.3929 | 0.4675 | 0.4675 | 0.4091 |
| Herring | 64 | 64 | 512 | 2 | 0.5313 | 0.6250 | 0.6250 | 0.5625 |
| InlineSkate | 100 | 550 | 1882 | 7 | 0.1891 | 0.4382 | 0.3927 | 0.3764 |
| InsectWingbeatSound | 220 | 1980 | 256 | 11 | 0.4894 | 0.6061 | 0.6242 | 0.6051 |
| ItalyPowerDemand | 67 | 1029 | 24 | 2 | 0.9174 | 0.9602 | 0.9466 | 0.9514 |
| LargeKitchenAppliances | 375 | 375 | 720 | 3 | 0.5600 | 0.7013 | 0.7813 | 0.6240 |
| Lightning2 | 60 | 61 | 637 | 2 | 0.7049 | 0.8197 | 0.6885 | 0.8033 |
| Lightning7 | 70 | 73 | 319 | 7 | 0.6438 | 0.7945 | 0.7808 | 0.8356 |
| Mallat | 55 | 2345 | 1024 | 8 | 0.9761 | 0.9501 | 0.9271 | 0.9561 |
| Meat | 60 | 60 | 448 | 3 | 0.8333 | 0.7333 | 0.9333 | 0.8667 |
| MedicalImages | 381 | 760 | 99 | 10 | 0.6237 | 0.6645 | 0.7079 | 0.6895 |
| MiddlePhalanxOutlineAgeGroup | 400 | 154 | 80 | 3 | 0.5455 | 0.5714 | 0.5130 | 0.6039 |
| MiddlePhalanxOutlineCorrect | 600 | 291 | 80 | 2 | 0.7285 | 0.7801 | 0.8385 | 0.7732 |
| MiddlePhalanxTW | 399 | 154 | 80 | 6 | 0.5325 | 0.5065 | 0.5390 | 0.5130 |
| MoteStrain | 20 | 1252 | 84 | 2 | 0.7772 | 0.8834 | 0.8746 | 0.8395 |
| NonInvasiveFatalECGThorax1 | 1800 | 1965 | 750 | 42 | 0.7104 | 0.2590 | 0.9435 | 0.8137 |
| NonInvasiveFatalECGThorax2 | 1800 | 1965 | 750 | 42 | 0.7537 | 0.7705 | 0.9450 | 0.8656 |
| OliveOil | 30 | 30 | 570 | 4 | 0.7333 | 0.1667 | 0.8333 | 0.7667 |
| OSULeaf | 200 | 242 | 427 | 6 | 0.6777 | 0.7769 | 0.6612 | 0.6322 |
| PhalangesOutlinesCorrect | 1800 | 858 | 80 | 2 | 0.7436 | 0.7646 | 0.8438 | 0.7751 |
| Phoneme | 214 | 1896 | 1024 | 39 | 0.1735 | 0.2184 | 0.1292 | 0.1772 |
| Plane | 105 | 105 | 144 | 7 | 1.0000 | 1.0000 | 1.0000 | 0.9524 |
| ProximalPhalanxOutlineAgeGroup | 400 | 205 | 80 | 3 | 0.7805 | 0.8341 | 0.8098 | 0.7951 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ProximalPhalanxOutlineCorrect | 600 | 291 | 80 | 2 | 0.8041 | 0.8488 | 0.8935 | 0.8076 |
| ProximalPhalanxTW | 400 | 205 | 80 | 6 | 0.7024 | 0.7756 | 0.7951 | 0.7268 |
| RefrigerationDevices | 375 | 375 | 720 | 3 | 0.3333 | 0.5147 | 0.4027 | 0.5067 |
| ScreenType | 375 | 375 | 720 | 3 | 0.4133 | 0.4293 | 0.3840 | 0.3680 |
| ShapeletSim | 20 | 180 | 500 | 2 | 1.0000 | 0.9500 | 0.5500 | 0.6000 |
| ShapesAll | 600 | 600 | 512 | 60 | 0.5800 | 0.7683 | 0.8217 | 0.7933 |
| SmallKitchenAppliances | 375 | 375 | 720 | 3 | 0.3333 | 0.6640 | 0.7040 | 0.5173 |
| SonyAIBORobotSurface1 | 20 | 601 | 70 | 2 | 0.6855 | 0.8103 | 0.7687 | 0.7388 |
| SonyAIBORobotSurface2 | 27 | 953 | 65 | 2 | 0.7901 | 0.8751 | 0.8468 | 0.7996 |
| StarLightCurves | 1000 | 8236 | 1024 | 3 | 0.9178 | 0.9466 | 0.9721 | 0.9655 |
| Strawberry | 613 | 370 | 235 | 2 | 0.9027 | 0.9108 | 0.9838 | 0.9270 |
| SwedishLeaf | 500 | 625 | 128 | 15 | 0.7680 | 0.9072 | 0.9376 | 0.8528 |
| Symbols | 25 | 995 | 398 | 6 | 0.9337 | 0.9317 | 0.9005 | 0.8422 |
| SyntheticControl | 300 | 300 | 60 | 6 | 0.9100 | 0.9967 | 0.9967 | 0.9733 |
| ToeSegmentation1 | 40 | 228 | 277 | 2 | 0.9561 | 0.9342 | 0.9167 | 0.8816 |
| ToeSegmentation2 | 36 | 130 | 343 | 2 | 0.6923 | 0.9154 | 0.9308 | 0.8462 |
| Trace | 100 | 100 | 275 | 4 | 1.0000 | 1.0000 | 1.0000 | 0.9900 |
| TwoLeadECG | 23 | 1139 | 82 | 2 | 0.9245 | 0.9965 | 0.9061 | 0.9385 |
| TwoPatterns | 1000 | 4000 | 128 | 4 | 0.9083 | 0.9933 | 0.9958 | 0.9910 |
| UWaveGestureLibraryAll | 896 | 3582 | 945 | 8 | 0.7887 | 0.9534 | 0.9520 | 0.9531 |
| UWaveGestureLibraryX | 896 | 3582 | 315 | 8 | 0.6946 | 0.7912 | 0.7965 | 0.7786 |
| UWaveGestureLibraryY | 896 | 3582 | 315 | 8 | 0.5958 | 0.7030 | 0.7300 | 0.6943 |
| UWaveGestureLibraryZ | 896 | 3582 | 315 | 8 | 0.6382 | 0.7468 | 0.7390 | 0.6960 |
| Wafer | 1000 | 6164 | 152 | 2 | 0.9968 | 0.9961 | 0.9972 | 0.9935 |
| Wine | 57 | 54 | 234 | 2 | 0.7593 | 0.5000 | 0.9259 | 0.7037 |
| WordSynonyms | 267 | 638 | 270 | 25 | 0.4310 | 0.6066 | 0.6599 | 0.6082 |
| Worms | 181 | 77 | 900 | 5 | 0.6494 | 0.6104 | 0.6104 | 0.5325 |
| WormsTwoClass | 181 | 77 | 900 | 2 | 0.7273 | 0.7273 | 0.6364 | 0.7013 |
| Yoga | 300 | 3000 | 426 | 2 | 0.6950 | 0.8343 | 0.8457 | 0.8133 |