Adversarial regularization for explainable-by-design time series classification

Yichang Wang, Rémi Emonet, Elisa Fromont, Simon Malinowski, Romain Tavenard

Adversarial Regularization for Explainable-by-Design Time Series Classification

Yichang Wang*, Rémi Emonet†, Elisa Fromont‡, Simon Malinowski*, Romain Tavenard§
* *Univ Rennes, IRISA UMR 6074, Rennes, France*
† *Univ Lyon, Lab Hubert Curien UMR 5516, Saint-Etienne, France*
‡ *Univ Rennes, IUF, IRISA UMR 6074, Rennes, France*
§ *Univ Rennes, LETG UMR 6554, Rennes, France*
{yichang.wang, elisa.fromont, simon.malinowski}@irisa.fr,
remi.emonet@univ-st-etienne.fr, romain.tavenard@univ-rennes2.fr

***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. This shapelet-based classification is both accurate and explainable since the shapelets are time series themselves and thus can be visualized and be provided as a classification explanation. In this paper, we claim that not all shapelets are good visual explanations and we propose a simple, yet also accurate, adversarily regularized EXplainable Convolutional Neural Network, XCNN, that can learn shapelets that are, by design, suited for explanations. We validate our method on the usual univariate time series benchmarks of the UCR repository.**

***Index Terms*—Time Series, Shapelets, Adversarial Networks, Explainable AI, Convolutional Neural Networks**

I. 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 classifying univariate time series. 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. 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 [1]) 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 [3], [4], 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 II). Jointly learning a shapelet-based representation of the series in the dataset and classifying the series according to this representation [5], [6] allowed to obtain discriminative shapelets in a much more efficient way. An example of such learned shapelets, obtained with the method from [6], is given in blue in Figure 1 (left). However, if the learned shapelets are definitively discriminative, they are often very different (visually) from actual pieces of a real series in the dataset. As such, these shapelets might not be suited to explain a particular classifier's decision. Note that the same interpretability issue arises with ensemble classifiers such as [7] where one decision depends on the presence of multiple shapelets. One of the main challenges nowadays is to provide Machine Learning (ML) methods that are both accurate and self-explanatory, i.e. provide mechanisms to explain their decisions to human users since, in many scenarios, it may be risky, unacceptable, or simply illegal, to let artificial intelligent systems make decisions without any human supervision [8].

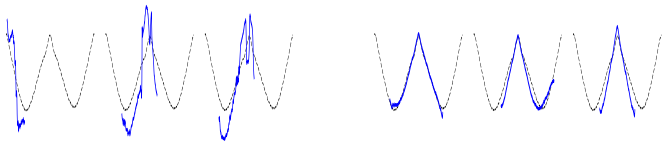


Fig. 1: The three most discriminative shapelets obtained for the dataset DiatomSizeReduction using (left column) Learning Shapelets or (right column) our XCNN architecture.

In this paper, we make use of a simple convolutional network to classify time series and we show how one can leverage the principle of adversarial learning to regularize the parameters of this network such that it learns shapelets that could be more useful to interpret the classifier's decision. Section II presents the most related work. We detail our XCNN method in Section III. In Section IV, we show quantitative and qualitative results on the usual time series benchmarks [9]: XCNN performance are on par with comparable state-of-the-art methods and our explainable-by-design method provides new types of explanations for neural network's predictions.

Elisa Fromont is supported by the HyAIAI (Hybrid Approaches for Interpretable AI) Inria "Défis" project. Romain Tavenard is partly funded by ANR through project MATS (ANR-18-CE23-0006).

II. Related Work

In this section we review the literature on shapelet-based Time Series Classification (TSC) and on tools for understanding black box model predictions.

A. Time Series Classification

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 [3] but in this work, the search space of all possible shapelets is explored exhaustively which makes the method intractable in practice. This high time complexity has led to the use of heuristics in order to select the shapelets more efficiently. In Fast Shapelets (FS) [4], the authors rely on quantized time series and random projections in order to accelerate the shapelet search but they sacrifice the accuracy, as reported in [2]. The Shapelet Transform (ST) [5] consists in transforming time series into a feature vector whose coordinates represent distances between the time series and the shapelets selected beforehand. However as in [3], the shapelets selection step makes the method 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 [10]. On the other hand, [6] 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 [11]. 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 [9]. A review of these models can be found in [2].

Finally, ensemble-based methods, such as COTE [7] or HIVE-COTE [12], 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 I) 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 [3] or ST [5]), yields interpretable results which can be used to explain the classifier's decisions (compared to ensemble approaches or unconstrained approaches such as [6] or [12]), and exhibits good classification accuracy (compared to FS [4]).

B. Model Interpretability

Among the vast number of existing classifiers, some are considered self-explanatory (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 (i.e. dedicated to one sample) explanations, and black-box dependent versus agnostic. In this category, state-of-the-art methods are Local Interpretable Model-agnostic Explanations (LIME and Anchors) [13], [14] and SHapley Additive exPlanations (SHAP) [15]. 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. However, these methods, contrarily to XCNN, are not able to show what has been learned and is used by the classifier to explain a particular decision.

GradCAM [16] is a popular local visualization method designed to explain neural networks decisions on image classification tasks. It uses gradient-based methods to highlight (with a heat map) the discriminative pixels on a given input test image. This method was adapted in MTEX-CNN [17] as an explanation and feature selection tool for multivariate time series (MTS) classification tasks which is a closer setting to ours. In [17], the authors proposed to stack 2D and 1D convolution sequentially to capture the important feature(s) and the important time stamp(s) for the time series. The prediction results are explained by inspecting the input MTS using GradCAM on both the variable and temporal dimensions.

[18] has a similar goal as ours (to produce interpretable discriminative shapelets) and build on both the work from [5] (in this case the candidate shapelets are extracted with a piece-wise aggregate approximation) and from [6] 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 [9] and they provide visualizations for only a couple of the learned shapelets.

The work from [19] is the closest to ours. Contrarily to ours, they decouple the shapelet learning phase and the classification process resulting in a quite different adversarial architecture. Their classification process is made using the shapelet transform method [5] but, in this case, the candidate shapelets are dynamically generated for each input time series. In our case, this is learned by a simple CNN for all the dataset. In [19], an adversarial regularization is also used to constrain the generated shapelets to be similar to real pieces of the series. However, the regularization is imposed on the result of the convolutions (i.e. the feature maps) and not on the convolutions themselves as we

propose to do in this paper. This is a different philosophy: we believe that the pattern detectors, i.e. the convolutions, are the shapelets. They believe that the shapelets are the series output by the convolution operation which might, in our opinion, have a very different shape than the original input signal. This difference of regularization may hinder the interpretability of the learned shapelets but this aspect is not studied in details in [19]. Besides, the proposed method does not allow global explanations (in addition to local ones) as can be done with our method. However, according to the results reported in [19], their method is more accurate than ours since it gives better results than LS [6], which gives similar results to our method, as shown in the experiments. The work proposed in [19] thus has a different trade-off explainability/accuracy than us.

Finally [20] also proposes a time series classification method. The authors propose to extract various symbolic representations from the time series and train a logistic regression model on top of these representations. The logistic regression weights are then inspected (using GradCAM) to extract the most discriminative features and localize the most important time series subparts. This method necessitates to discretize the original signal (and thus lose some information), it is not self-explanatory (the explanations are post-hoc) and we believe that showing the shapelets, as we can do in our method, is an important feature for explaining decisions.

III. TS Classification with Regularized Shapelets

In this section, we present our architecture, XCNN, to learn interpretable discriminative shapelets for time series classification. Our base time series *classifier* is a Convolutional Neural Network (CNN). As explained in Section II, this model is very similar in spirit to the Learning Shapelet (LS) model presented in [6]. Both LS and CNN slide the shapelets on the series to compute local (dis)similarities. 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:

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

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:

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

Note that here, the higher $D(z_{i:i+L}, S)$, the more similar the shapelet is to the subseries. We will loosely refer to the convolution filters of our classifier as *Shapelets* in the following.

A. XCNN Architecture

Inspired by previous work on adversarial training (e.g. [21]), 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. 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 training procedures are explained in the next subsection.

Contrarily to GANs, our adversarial architecture does not rely on a generator to produce fake samples from a latent space. XCNN iteratively modifies the shapelets (i.e. the convolution filters of the classifier) such that they become close to subseries from the training set. The type of data given as input to the discriminator is another major difference between a GAN and XCNN: in a GAN, the discriminator is fed with complete instances, while in XCNN, 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 between shapelets and training time series portions ($\hat{x}$, see the following section for more details on those). This process allows the discriminator to alter the shapelets for better interpretability.

B. Loss Function

As for GANs, our optimization process alternates between losses attached to the subparts of our 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, so that the regularized-shapelets become similar to a subsequence of time series. 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 θ_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) [22]:

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

where $\mathbb{P}_S$ is the empirical distribution over the shapelets, $\mathbb{P}_x$ is the empirical distribution over the training subseries, and $\hat{x} = \epsilon x + (1-\epsilon)\tilde{x}$, where ϵ is drawn uniformly at random from the interval $[0, 1]$.

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:

$$L_r(\theta_s) = -\mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_S}[D(\tilde{x})] \tag{1}$$

IV. Experiments

In this section, we will detail the training procedure for XCNN and present both quantitative and qualitative experimental results.

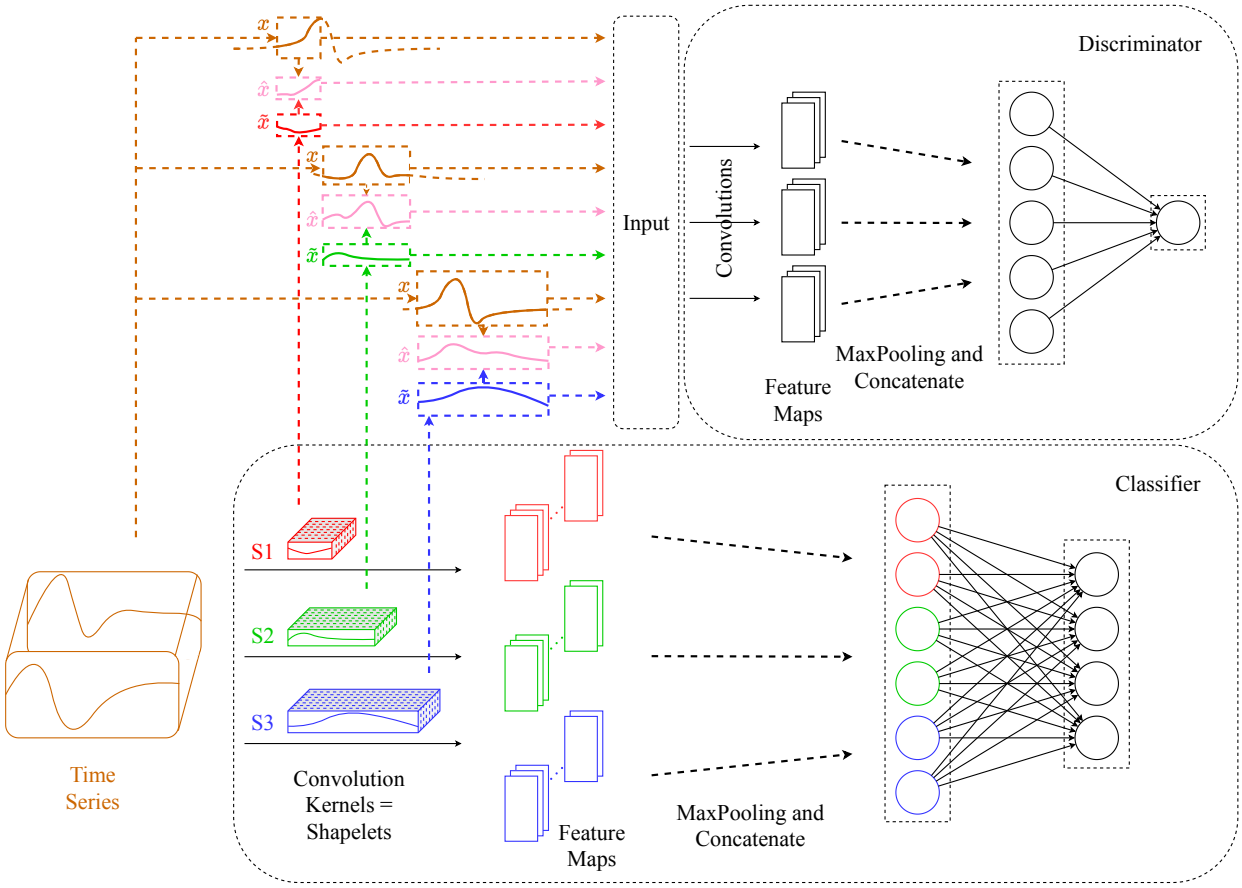


Fig. 2: Adversarial architecture of our proposed explainable CNN (XCNN).

A. Experimental Setting

1) Competitors: We provide experiments about the quality (for explanations) of our learned shapelets as well as their quality for classification. As explained in Section II, our most relevant competitor is Learning Shapelets (LS) from [6] as it also describes a shapelet-based model where the shapelets are learned and where a single model is used for classification. The quality (for explanations) of the shapelets produced by [3] and [4] is, by design, perfect since the shapelets are true subpart of the original series so we do not compare with them but only with the shapelets learned by [6]. However, we compare our classification performance to [3], Fast Shapelets [4] and the recent ELIS [18].

2) Datasets: We use the 85 univariate time series datasets from the UCR/UEA repository [9] for which most of our baselines results are already available.[1]

Note that our CNN-based method may also be suited for multivariate time series but giving "intuitive" explanations for multivariate data is far from obvious and we decided to focus only on univariate ones in this paper. 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.

3) Architecture details and parameter setting: We have implemented the XCNN model using TensorFlow [23] following the general architecture illustrated in Fig. 2. The classifier is composed of one 1D convolution layer with ReLU activation, followed by a max-pooling layer along the temporal dimension and a fully connected layer with a soft-max activation. The shapelets use a Glorot uniform initializer [24] 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 [6] but without resorting to hyper-parameter search: we consider 3 groups of $20 \times n_{\text{cl}}$ shapelets of length $0.2T$, $0.4T$ and $0.6T$, where n_{cl} 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 soft-max in the last layer. For optimization, we use Adam optimizer with a standard parameterization ($\alpha = 10^{-3}$, $\beta_1 = 0.9$ 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.

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

discriminator and regularizer losses).

Experimental results are reported in terms of test accuracy and aggregated over five random initializations. All experiments are run for $8,000$ training epochs.

B. Qualitative results for explainability

We first illustrate the evolution of a shapelet during the training process. Then we describe how we compute the shapelet contributions to the classification of one (or multiple) example(s) and validate that our adversarial regularization actually ensures that shapelets are visually similar to the training data. And finally we show, in three different ways, how shapelets that look like subseries are better suited to explain decisions.

We believe that the Euclidean distance is the most understandable distance for human eyes so all the figures that show shapelets and series will be displayed using this distance even though it is not the one optimized during XCNN training.

1) Evolution of a shapelet during training process: We illustrate our training process and its impact on a single shapelet in Figure 3. 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 3a and 3d 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.

2) Shapelet contributions: The computation of the contribution of a shapelet to a decision is based on GradCAM ("Gradient-weighted Class Activation Mapping") [16]. GradCAM is a very popular method used in computer vision to understand which parts of an original image is used by a trained neural network to make a particular classification decision. The "interesting" parts are shown using a heat map on the original image. We recall that in a convolutional neural network, a *feature map* is the output of a particular layer of neurons. It somehow (ignoring the activation function) shows the response of a given convolution filter to the output of the previous layer. GradCAM computes the feature importance α_k^c of the feature map A^k on the classification decision c. This is computed after the final pooling layer which transforms all spatial positions (for images) A_{ij}^k of the k^{th} feature map to a single value F^k. The filter importance weight α_k^c, for a given input image (omitted for conciseness), is calculated with: $\alpha_k^c = \frac{\partial y^c}{\partial F^k}$ where y^c is the output of the network for class c.

Compared to the image classifiers used in [16], in our time series classification problem (1-dimensional) we are interested in both the **positive and negative contributions** of each learned shapelet on the classification of the (set of) series (whereas in [16] only the positive contributions matter). Those contributions are defined for a trained network and a given time series Z_i (implicitly present in the partial derivatives) as: $p_k(Z_i) = ReLU\left(\frac{\partial y^c}{\partial F^k}\right)$ and $n_k(Z_i) = -ReLU\left(-\frac{\partial y^c}{\partial F^k}\right)$

As F^k is obtained from a global max pooling ($F^k = \max_t A_t^k$), each shapelet contribution can be associated to a timestamp $t = \arg\max_{t'} A_{t'}^k$, allowing us to localize the contribution. To produce a heat map with the positive contributions, we follow the same principle as in [16]: $L_{mask}(Z_i) = \sum_k p_k(Z_i)\tilde{A^k}(Z_i)$. where $\tilde{A^k}$ is a vector of all zeros but at position $t = \arg\max_{t'} A_{t'}^k$ (where A_t^k is stored).

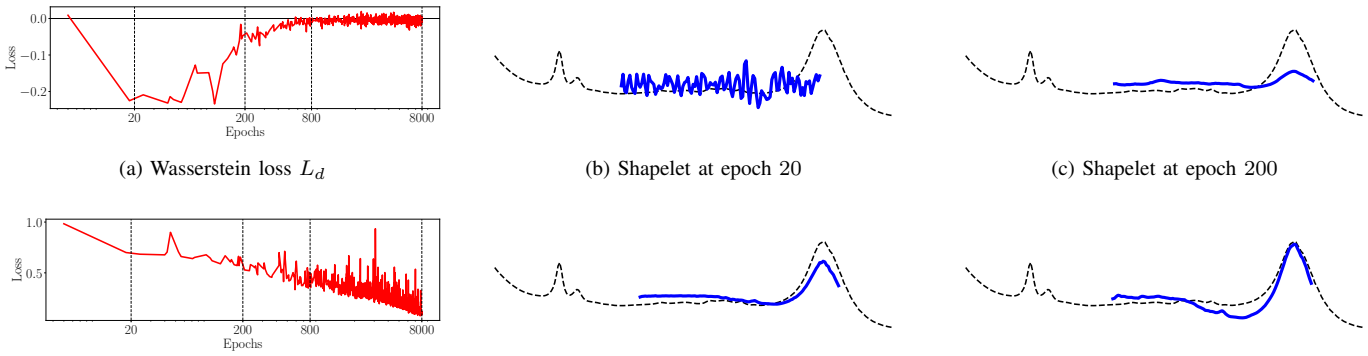
To obtain the **global positive contribution** of a shapelet k given a set of n time series examples, we compute

$$gp_k = \frac{1}{n}\sum_{i=1}^{n} p_k(Z_i). \tag{2}$$

The shapelets shown in Fig. 4 are the 3 most contributing shapelets, according to this global criterion. In Fig. 4, the shapelets learned by XCNN seem visually closer to the time series than the shapelets learned by LS. We then computed the average L_2 between a shapelet and a subpart of a time series over all the shapelets learned by XCNN and by LS for a given dataset, computed at the best matching point of the closest time series in the dataset (also in terms of L_2). The results are given in Fig. 5. This scatter-plot shows that, even if the optimized distance between the shapelets and the input series in the neural network is not the L_2 one (it is the dot product), our adversarial regularization allows XCNN to obtain closer (in terms of L2) shapelets than LS which are deemed more suited for explanations.

3) Gradient-based explanations with XCNN shapelets: Since we use a neural network classifier, we could directly benefit from the standard gradient-based explanations, as also discussed in [25], to show what parts of a given time series example is important for the classifier to take its classification decision. These explanations would also be the ones produced by post-hoc methods such as LIME [13]. For lack of space, we do not show examples of such explanations but the interested reader can find many examples in [25] or in [20].

These, nowadays standard, gradient-based explanations are interesting but do not show the inner working of the classifier and, in particular, the reason why some parts of the input series were particularly useful for the classification. We believe that our ability, with XCNN, to show the shapelets that were learned and used to make the classification gives a different type of information than the gradient-based one. To illustrate this, we overlay in Fig. 6 and 7 the three most positively (resp. negatively on the right) contributing shapelets on the time series at their best matching location (using L2 distance), with number of total positively and negatively shapelets noted in the captions. Note that on the left side, the horizontal axis gives the length of the series (in black) while on the right, it gives the length of the shapelets which is at most 60% of the length of the series. We do not show the original series for the negative shapelets since, by definition, they are very far from the original series. In Fig. 7 there is no negative shapelet used to discriminate the series of this dataset. This is due to the fact



(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

Fig. 3: Illustration of the evolution of a shapelet during training (for the Wine dataset).

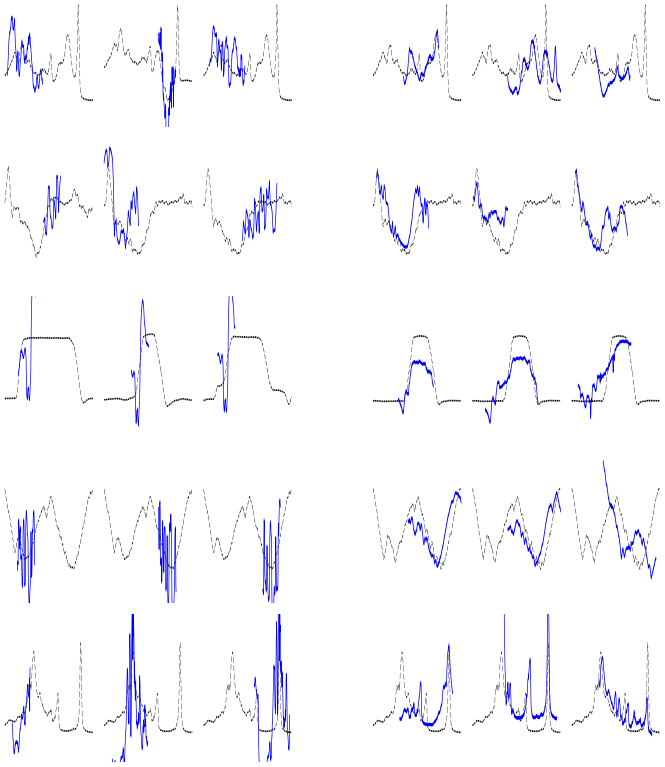


Fig. 4: Three most discriminative shapelets obtained for the datasets Beef, ECG200, GunPoint, Herring, OliveOil (rows 1 to 5, resp.) using (left column) LS or (right column) XCNN. The average discriminative power of the shapelets is evaluated using Eq. 2 and each shapelet is superimposed over its best matching time series in the test set.

that the series for all the classes are very similar except for very small changes in the slope of the bump or in the size of the plateau at the top of the bump. These small changes can be captured by the positive shapelets but many of them are used to succeed in discriminating the classes.

We can also use our method to show the shapelets that

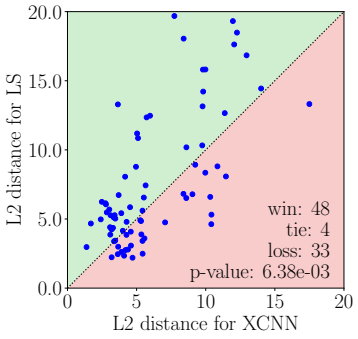


Fig. 5: Average over all the shapelets learned by XCNN and by LS for a given dataset, of the L_2 distances between a shapelet and a subpart of a time series at the best matching point of the closest time series in the dataset.

most contribute to the classification of all examples of a *given class*. This is useful when one wants to understand the class characteristics. The global relative positive contribution of one shapelet considering all series from a given class is:

$$rp_k(c) = ReLU\left(\frac{1}{N^c}\sum_{i=1}^{N^c}\left(p_k^c(Z_i) - \frac{1}{n_{\text{cl}}-1}\sum_{\substack{j=1\\ j\neq c}}^{n_{\text{cl}}-1} p_k^j(Z_i)\right)\right)$$

where N^c is the number of examples in class c, and n_{cl} is the total number of classes in the dataset. One can compute $rn_k(c)$ similarly by replacing p_k^j with n_k^j. The time series shown in black in Fig. 8 and 9 is the average over all examples of a given class. With these figures, we can draw similar conclusions as the previous ones but for an entire class.

C. Quantitative Results

XCNN is able to learn, by design, shapelets that are discriminative and suited for explanations. 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

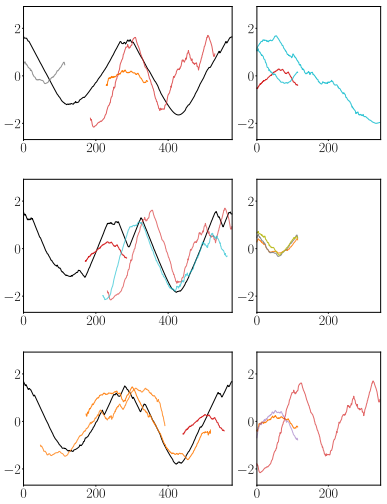


Fig. 6: Three most positively (left) and negatively (right) contributing shapelets for a random series (in black) of some of the classes (class 0, 1, 2, resp.) of the Car test set. Note that there were different positive and negative for different series, *e.g.* there were in total 92 positive shapelets used for the decision of the first test series and 148 negative ones.

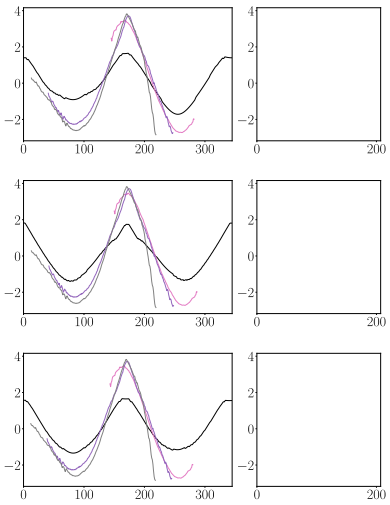


Fig. 7: Three most positively (left) and negatively (right) contributing shapelets for a random series (in black) of some of the classes (class 0, 1, 2, resp.) of the DiatomSizeReduction test set. Note that there were different positive and negative for different series, *e.g.* there were in total 240 positive shapelets used for the decision of the first test series and 0 negative ones.

Shapelets [3]), much more accurate than very fast random shapelet selection-based methods (our baseline is FS [4]) and as accurate and as fast as single model shapelet learning methods (our baselines are LS [6] and ELIS [18]).

1) Accuracy: We analyze the accuracies obtained by FS, LS, ELIS and our XCNN method on the 85 datasets using scatter plots. We compare FS versus XCNN in Fig. 10, LS

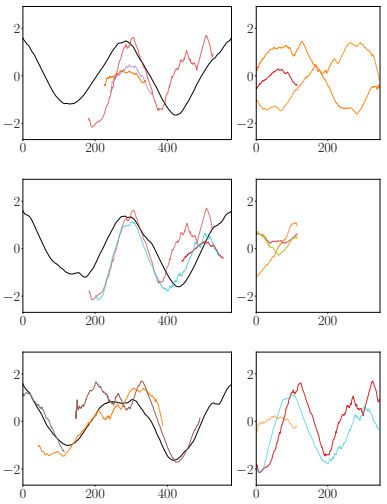


Fig. 8: Three most positively (left) and negatively (right) globally contributing shapelets for some of the classes (class 0, 1, 3, resp.) of the Car test set.

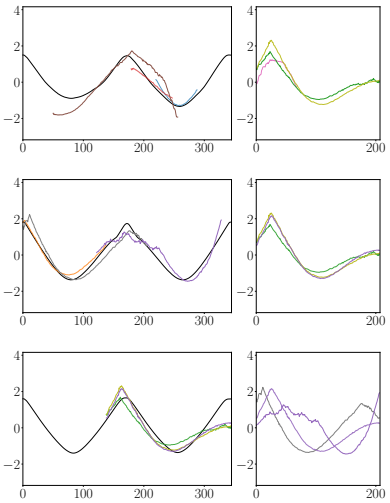


Fig. 9: Three most positively (left) and negatively (right) globally contributing shapelets for some of the classes (class 0, 1, 2, resp.) of the DiatomSizeReduction test set.

versus XCNN in Fig. 11 and ELIS versus XCNN in Fig. 12. We also show how a simple CNN (without the adversarial regularization) compares against LS in Fig. 13. We indicate the number of *win/tie/loss* for our method and we provide a Wilcoxon significance test [26] 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. Fig. 10 shows that, as expected, our method yields significantly better performance than FS. It gives similar results (not significantly better nor worse on average) than ELIS for 52 datasets for which ELIS terminated in 48 hours. However for 33 datasets ELIS took more than 48 hours to complete. Compared to

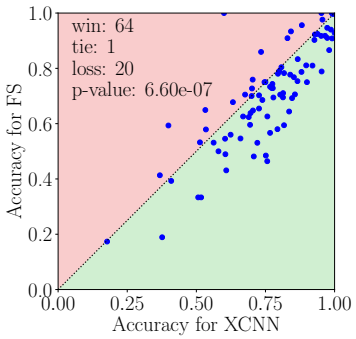


Fig. 10: Accuracy comparison between Fast Shapelets (FS) and XCNN on 85 datasets (each point is a dataset) of the UCR/UEA repository [9].

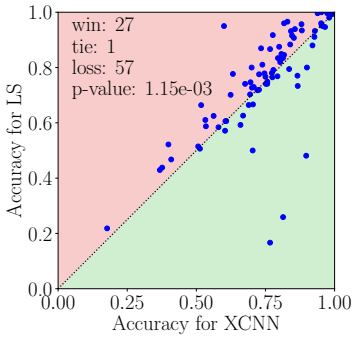


Fig. 11: Accuracy comparison between Learning Shapelets (LS) and XCNN on 85 datasets (each point is a dataset) of the UCR/UEA repository [9].

LS, for most datasets, the difference in accuracy is low, with a small edge (significant) for LS. On three datasets (namely HandOutlines, NonInvasiveFetalECGThorax1 and OliveOil), our XCNN method and its regularization seems to be strongly positive (and detrimental on one dataset), in terms of generalization. A simple CNN that would correspond to the classifier of our XCNN alone seems to give slightly better (non significant) results than LS (and thus than our XCNN). 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.

TABLE I: Complexity of four different shapelet-based TSC algorithms (Shapelet [3], FS [4], LS [6] and XCNN). n is the number of examples in the training set, T is the average length of the time series, n_{shap} is the number of selected shapelets (if set *a priori*), and n_{cl} is the number of classes.

Shapelet	FS	LS and XCNN (per epoch)
$O(n^2 \cdot T^4)$	$O(n \cdot T^2)$	$O(n \cdot (T^2 n_{\text{shap}} + n_{\text{shap}} \cdot n_{\text{cl}}))$

2) Training Time: We provide a theoretical complexity study (see Table I) of all the baselines and of our XCNN method. Our method is based on a classifier and a discriminator, and both of them are simple CNNs. So the complexity of our algorithm ($O(n \cdot (T^2 n_{\text{shap}} + n_{\text{shap}} \cdot n_{\text{cl}}))$) is related to training a CNN and should depend mainly on the number of examples (n),

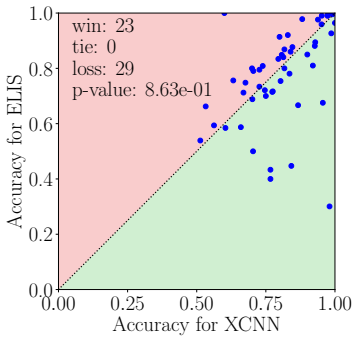


Fig. 12: Accuracy comparison between ELIS and XCNN on 85 of the UCR/UEA [9] datasets.

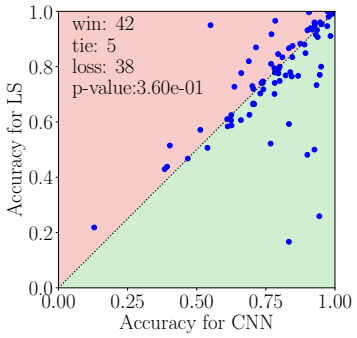


Fig. 13: Accuracy comparison between Learning Shapelets (LS) and a simple CNN on 85 of the UCR/UEA [9] datasets.

the average length of the time series (T), and the number of classes (n_{cl}), since the latter is used to decide the number of shapelets to be learned. Note that for both LS and XCNN, the provided complexity is the one for a single iteration of the algorithm since the number of iterations required for such algorithms to converge is highly data dependent.

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 [3] 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 XCNN is identical so these results were surprising. We suspected that the JAVA implementation of LS was not well optimized and we used the implementation of LS method from tslearn [27] using Keras[2] with TensorFlow as backend. With this implementation, the training phase took only 71 minutes for LS on this dataset (compared to 142 for XCNN) which shows that the time difference between the two algorithms is mainly related to the implementation (and the hyper-parameters related to the

[2]https://keras.io/

number of epochs).

V. Conclusion

We have presented a new shapelet-based time series classification method that produces shapelets that are, by design, better suited to explain decisions. 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 both discriminative but also meaningful 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 but with shapelets that can be used in many different way to explain the decisions.

In future work, we would like to first investigate the use of an additional regularization term to be able to determine automatically a minimal set of necessary 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 our approach to explain anomaly detections using neural network architectures such as convolutional auto-encoders or generative networks.

Acknowledgment

The authors would like to thank the archivists of the UCR/UEA time series classification repository for all the resources they made available which greatly help all the researchers working on the analysis of time series.

References

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

[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*, vol. 31, no. 3, pp. 606–660, 2017.

[3] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proceedings of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data mining*, 2009, pp. 947–956.

[4] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 668–676.

[5] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proceedings of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data mining*, 2012, pp. 289–297.

[6] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data mining*, 2014, pp. 392–401.

[7] 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*, vol. 27, no. 9, pp. 2522–2535, 2015.

[8] 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*, vol. 51, no. 5, 2018.

[9] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.

[10] I. Karlsson, P. Papapetrou, and H. Bostrom, "Generalized random shapelet forests," *Data Mining and Knowledge Discovery*, vol. 30, no. 5, pp. 1053–1085, Sep 2016.

[11] 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*, 2017, pp. 1578–1585.

[12] 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)*, vol. 12, no. 5, p. 52, 2018.

[13] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.

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

[15] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017, pp. 4768–4777.

[16] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *IEEE International Conference on Computer Vision, ICCV*, 2017, pp. 618–626.

[17] R. Assaf, I. Giurgiu, F. Bagehorn, and A. Schumann, "MTEX-CNN: Multivariate Time Series EXplanations for Predictions with Convolutional Neural Networks," in *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 952–957.

[18] 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)*, 2018, pp. 497–508.

[19] Q. Ma, W. Zhuang, S. Li, D. Huang, and G. W. Cottrell, "Adversarial dynamic shapelet networks," in *The Thirty-Fourth Conference on Artificial Intelligence (AAAI)*, 2020, pp. 5069–5076.

[20] T. Le Nguyen, S. Gsponer, I. Ilie, M. O'Reilly, and G. Ifrim, "Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations," *Data Mining and Knowledge Discovery*, pp. 1–40, 2019.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[22] 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.

[23] 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. [Online]. Available: http://tensorflow.org/

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, vol. 9. PMLR, 2010, pp. 249–256.

[25] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

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

[27] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, "Tslearn, a machine learning toolkit for time series data," *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-091.html