Ad-Hoc Explanation for Time Series Classification

A. Abanda amaiabanda@gmail.com[1], U. Mori usue.mori@ehu.es [2], and J. A. Lozano ja.lozano@ehu.eus [1,2]

[1]Basque Center for Applied Mathematics (BCAM), Mazarredo Zumarkalea, 14, 48009 Bilbao, Spain
[2]Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, Manuel de Lardizabal 1, 20018 Donostia-San Sebastian, Spain

August 29, 2022

Abstract

In this work, a perturbation-based model-agnostic explanation method for time series classification is presented. One of the main novelties of the proposed method is that the considered perturbations are interpretable and specific for time series. In real-world time series, variations in the speed or the scale of a particular action, for instance, may determine the class, so modifying this type of characteristic leads to ad-hoc explanations for time series. To this end, four perturbations or transformations are proposed: warp, scale, noise, and slice. Given a transformation, an interval of a series is considered relevant for the prediction of a classifier if a transformation in this interval changes the prediction. Another novelty is that the method provides a two-level explanation: a high-level explanation, where the robustness of the prediction with respect to a particular transformation is measured, and a low-level explanation, where the relevance of each region of the time series in the prediction is visualized. In order to analyze and validate our proposal, first some illustrative examples are provided, and then a thorough quantitative evaluation is carried out using a specifically designed evaluation procedure.

Keywords: Time series classification, agnostic, ad-hoc explanation, time series transformations, robustness

1 Introduction

Machine learning models are used daily in a wide range of application domains, both in research and in real-word problems. Specifically, the analysis of temporal data has become one of the most popular tasks. Among the challenges when working with time series, one of the most common tasks is Time Series Classification (TSC). The objective of this tasks is finding a function $f : T \rightarrow y$ that maps the space of time series to the space of labels, based on the example pairs in a training set $D = (T_1, y_1), \ldots, (T_n, y_n)$, where y_i is the label associated to the time series T_i, and the time series are assumed to be independent and identically distributed. Over the years, machine learning models, and TSC models in particular, have become more and more complex -also accurate-, leading to a decrease in the transparency of the decisions they make and, hence, in their interpretability.

Even if a model obtains a good performance in a given problem, being able to understand and interpret its predictions is a key aspect to trust the model, specially in some critical domains, such as biomedicine. In order to deal with this issue, a new field of research, called Explainable Artificial Intelligence (XAI) [1], has greatly emerged in recent years. The goal of explainability is to explain the decision of a specific model with respect to a particular instance. For image data, for instance, the explanation of a model for a given instance is usually shown as a saliency map [2], where each pixel is colored depending on its relevance in the prediction of the model.

Explanation methods can be categorized in several ways: regarding the model-dependence, they can be model-dependent (intrinsic) or model-independent (agnostic), while, depending on their scope, they can be local or global. Intrinsic explanation methods are limited to the particular model (or types of model) they are designed for, while agnostic explanation methods are general approaches for obtaining an explanation of any model. Local methods provide explanations for a particular data instance, while global explanation methods provide a single explanation for a whole dataset (the coefficients of a linear model, for instance).

Recently, one of the most popular perturbation-based explanation methods, the Local Interpretable Model-agnostic Explanations (LIME) [3], has been adapted to time series classification [4]. The LIME method, from where we depart in this paper, consists of locally approximating the classifier by an interpretable model. To this end, a synthetic neighbourhood is generated around a given instance by *perturbing* it, and then, the explanation is obtained by learning an interpretable model using the neighbours (see [3] for more details). In [4], in order to adapt the method to time series and take the temporal information into account, the perturbations to generate the neighbours are defined as transformations applied to one or more intervals (subsequences) of the time series, instead of to isolated points. These transformations consists of replacing the given interval(s) by another subsequence, such as an interval of random noise, the linear interpolation between the first and last point of the interval, or subsequences extracted from

the time series of the training set.

The main drawback of these methods is that the neighbourhoods generated with these transformations are not realistic. For instance, if a time series represents the electricity consumption of a country (as in the ItalyPowerDemand dataset from the UCR repository [5]), replacing a given interval of the series by random noise does not produce a realistic neighbour. In particular, the generated neighbour does not have a semantic meaning, since an interval of random noise can not be interpreted from an electricity consumption point of view.

Our proposal aims at going a step further, and provides a local model-agnostic explanation for TSC based on a more realistic neighbourhood for time series. To this end, the vicinity of a time series is generated by applying transformations that are more natural for time series data. More specifically, in this work, we consider 4 transformations based on [6, 7, 8]: warp, scale, noise and slice. With these transformations, the explanations provided by our method have an interpretation: an interval is important because, if a certain transformation is applied to this interval, the prediction will change. More concretely, given a transformation type, the proposed method provides two explanations: on one hand, the robustness for the prediction of the classifier with respect to that transformation and, on the other, the relevance of each region of the series in the prediction.

The main contribution of our work is that, while standard TSC explanation methods just reveal the relevant regions of a given time series, our method provides an additional knowledge: a region of the series is important because, if a specific transformation is applied to this region, the classifier is likely to label the transformed time series into another class. This information can be very useful in many real-life problems, such as, for instance, in an scenario where the time series represent the water flow of a water distribution network and the objective is to find water leaks [9]. Given a time series that represents a water leak, our method reveals which region of the series is relevant for classifying it as a water leak and, which transformation should be applied to this region to transform the series into a non-leak flow. In particular, if a region is shown to be relevant with respect to the scale transformation of level xx, it means that if this subsequence is scaled with a magnitude of xx (moved down, for instance), the classifier would label the series into the non-leak category. In conclusion, the explanation would give us information about the time localization and magnitude of the leak.

The rest of the work is organized as follows: in Section 2, the state-of-the-art of explanation methods for time series classification is introduced, while the proposed transformations for time series are presented in 3. The explanation method is thoroughly described in Section 4, while the experimentation is presented in Section 5. Finally, the main conclusions are drawn in Section 6.

2 State-of-the-art

In the field of TSC, the growing interest in explainability has been materialized in several proposals. In Table 1, we have summarized the existing explanation methods for TSC, including the name of the method, the explanation type, the model-dependence and scope.

As it can be seen in the table, there are different forms in which the explanations are provided in TSC; the most common explanation type is given in the form of a saliency map, where the relevance of each point of a time series in the prediction of the classifier is computed and then visualized. Other types of methods include explanations based on shapelets [10, 11] (in which a subsequence or set of subsequences of the time series that are relevant for the classification are provided) or counter-examples [12, 13] (in which, given a target time series, a similar series that is classified into another class is shown).

Regarding the model-dependence, most of the proposed explanations methods for TSC are intrinsic approaches [11, 12, 13, 14, 15, 16, 17, 18, 19, 20], that is, they are limited to a particular classifier or set of classifiers. Among these proposals, we can distinguish between different approximations: in [14, 16], for instance, two feature based classifiers are presented, in which the time series are transformed into some specific feature spaces and the relevance of each feature in the classification is computed. Then, the explanation is computed by translating the relevance back to the time domain. In AI-PR-CNN [11], on the contrary, the authors propose a shapelet based TSC method, in which adversarial training is used to obtain shapelets that are similar to some subsequences of the real series. The explanation is given directly by the shapelets found by the method during the training of the classifier. In [15], the main idea is to find the minimum number of changes that need to be applied to a time series to change the decision of the Random Shapelet Forest classifier [21]. Another explanation method is presented in [17], where, given a K-NN classifier, the explanation is given by the closest time series that belongs to the other class (with respect to a given distance measure). The last group of intrinsic methods is a set of explanation methods that are specifically designed for DNN. In some cases, the gradients included in the DNN are exploited for obtaining the explanation [20, 12], while others employ AutoEncoders [13] or exploit specific architectures of the DNN that enable explanations, such as the Class Activation Map (CAM) layer [18, 19]. As mentioned previously, this type of methods are limited to the particular classifier(s) they are obtained from.

Contrary to intrinsic methods, agnostic methods are capable of explaining the decision of any classifier. In the field of TSC, on which this work focuses, to the best of our knowledge, four local explainers have been proposed [4, 22, 23, 24]. In [24], one of the most popular perturbation-based explanation methods, the Local Interpretable Model-agnostic Explanations (LIME) [3], is directly applied to time series. In [4], instead, the LIME method is adapted to time series by applying some transformations to intervals in the time domain (subsequences of the time series). An extension of the LIME method called

SHAP [25] is used for the explanation method proposed in [22], where transformations are applied to intervals of the time series in both the time and frequency domain. Finally, a Local Agnostic Subsequence-based Time Series explainer (LASTS) is presented in [23]. The explanations are provided in the form of exemplar and counter-exemplar time series (synthetic time series that are classified into the same or into a different class of the target time series, respectively). The authors employ autoencoders (AE) [26] to create synthetic neighbours around the target time series, together with a shapelet tree for designing an explanation that provides subsequences that must, or must not, be contained in a time series to obtain a particular outcome.

Method	**Type**	**Model-dependence**	**Scope**
AI-PR-CNN [11]	Shapelets	Intrinsic* (DNN)	Local
MTEX-CNN [12]	Saliency	Intrinsic* (DNN)	Local
CEM [13]	Counter-examples	Intrinsic* (DNN)	Local
SAX-VSM [14]	Saliency	Intrinsic	Local
Tweaking RSF [15]	Counter-examples	Intrinsic	Local
MrSEQL-SM [16]	Saliency	Intrinsic	Global
Native-Guides [17]	Counter-examples	Intrinsic* (K-NN)	Local
CAM [18, 19]	Saliency	Intrinsic* (DNN)	Local
Dynamic Masks [20]	Saliency	Intrinsic* (DNN)	Local
LEFTIST [4]	Saliency	Agnostic	Local
TimeXplain [22]	Saliency	Agnostic	Local
LASTS [23]	Counter-examples	Agnostic	Local
LIME TS [24]	Saliency	Agnostic	Local

Table 1: Existing explanations methods for TSC and their characteristics. The * indicates that these methods are not limited to one specific classifier but to a type of classifiers. The type is shown between parenthesis (DNN refers to Deep Neural Networks and K-NN refers to TSC method employing the K-Nearest Neighbour classifiers).

A particular aspect that complicates the design of the proposal of explanation methods is that the evaluation is usually carried out in a qualitative way. In TSC, most of the explanation methods are evaluated visually by performing a *case of study* [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 23]. However, there are also some proposals in which efforts have been made to carry out a quantitative evaluation. In these works, the authors usually propose a specifically designed quantitative measure that depends on the particular classifier or problem being explained [11, 20, 4, 23]. A more particular evaluation methodology is employed by Guillemé *et al.* in [4], where a user study is carried out in a dataset in which characteristic patterns of the classes are relatively easy to identify visually. In this way, users are asked to answer some questions regarding the discriminative parts of the series, and the answers are compared with explanations obtained by their method. Lastly, a promising work is

presented in [24], where the authors propose a model agnostic evaluation methodology for TSC explanation methods (in the form of saliency maps). In this work, the time series are perturbed according to the relevance of each point in the prediction. The main underlying idea is that perturbations on the relevant parts should change the decision of the classifier more frequently than perturbations on the non-relevant parts.

3 Time series Transformations

As stated in previous sections, the explanation method that we will present in this work is based on LIME [3]. As such, it is based on generating a local interpretable model using a neighborhood of the target instance. In this context, in order to generate the neighborhood, the target instance is perturbed using a set of pre-defined transformations. The main achievement of this work is to include specific transformations for time series that will result in realistic and semantically interpretable neighbors. In this section, we present the set of transformations for time series that we are going to consider for our explanation method.

Time series can be analyzed in both the time and frequency domain, and thus, the transformations could be defined in both these spaces. However, in TSC and more specifically in TSC explainability, it is much more common to work in the time domain. Indeed, to the best of our knowledge, the only work in TSC explainability where the series are represented in the frequency domain is [22]. The main reason for this is that transformations in the time domain can be applied to specific temporal instances (or intervals), they are usually more intuitive and easier to interpret. In this sense, our transformations will be defined in the time domain.

Once we focus on the time domain, there are many methods that are useful to generate synthetic time series, for example time series bootstrapping [27, 28, 29], o Generative Adversarial Networks (GAN) [30]. The main objective of these methods is to produce time series that resemble the time series in a given database of series (follow the same probability distribution). These methods do not allow to fully control the characteristics of the generated time series, and hence a generate and test type of methodology would have to be used (in combination with a distance measure such as in [3]) with the aim of selecting the neighboring time series. In order to provide a more efficient approach, we will generate the vicinity of a time series by using a set of realistic and controlled transformations for time series, for which we follow the ideas presented in [6, 7, 8].

In [6], the authors state that a similarity measure for time series should be robust to a set of transformations and they propose the scale, warp, noise, and outliers transformations. In [7], the authors claim that time series with unequal lengths appear naturally in real-world problems due to reasons such as variations in the frequency of measurements (warp) or variations in the starting or/and ending index of the time series (slice), and propose

transformations to simulate this kind of time series. Lastly, time series data augmentation is addressed in [8], where two methods are proposed: time series slice extraction and window warping. Inspired by the previous works, we decided to consider 4 transformations: warp, scale, noise, and slice. In our case, transformations that involve intervals of the time series are considered, so the outlier transformation -which applies to certain indexes of the series but not to a whole interval- has not been taken into account. In the following sections, the considered transformations are presented in detail.

3.1 Warp

This transformation is a deformation of the series in the time axis (x-axis), which produces a compression or expansion (depending on the warp level) of the values of a series in a given interval. Time series with an interval warped at different warp levels appear frequently in problems such as the GunPoint dataset from the UCR repository, in which a sensor on the wrist of a person measures the movements in the x-axis. There are two classes in this problem, the Gun and the Point class; in the former, a person has a gun on his/her hip, takes the gun to point it at a target for one second, and brings the gun back to the hip. In the latter, the same action is simulated without the gun, that is, the person points at a target with a finger. Figure 1 shows two time series from this dataset, one from each class. The interval that represents the gun rising can be seen as a warped version of the interval that represents the hand rising (or vice-versa). In particular, in the time series from the Gun class, this movement is made in a quicker manner than in the time series from the Point class. In this case, this interval is considered discriminative with respect to the warp transformation, because if we compress/expand it, the classifier could change its class prediction.

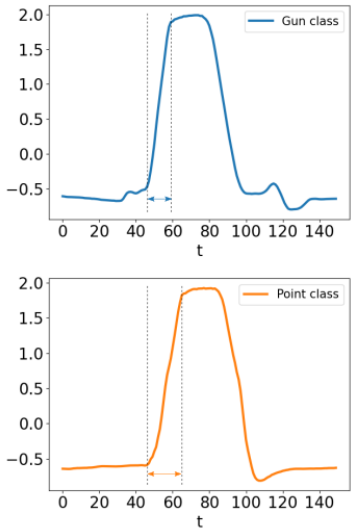


Figure 1: Two time series from the GunPoint dataset. The time series from the Gun class can be seen as a warped version of the time series from the Point class (or vice-versa), where the warp has been applied in the marked interval.

In order to synthetically create warped versions of a reference time series $T = (t_1, \ldots, t_i, \ldots, t_l)$, we propose the warp transformation; given an interval $[s, e]$ in which the transformation is applied (such that $0 < s < e < l$) and a warp level k_w, the warped version of T is defined by:

$$T' = (t'_1, \ldots, t'_i, \ldots, t'_{s+(e-s)*k_w+(l-e)}) \tag{1}$$

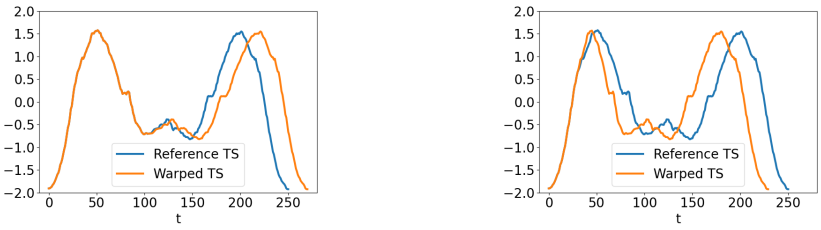
where

$$t'_i = \begin{cases} t_i, & \text{if } i \leq s \\ (w_1 t_p + w_2 t_q)/k_w, & \text{if } s < i \leq s + k_w(e-s) \\ t_{e+i-(s+k_w(e-s))}, & i > s + k_w(e-s) \end{cases} \tag{2}$$

where $t_p = \max_{j=1,\ldots,l} [jk_w] < i$ and $t_q = \min_{j=1,\ldots,l} [jk_w] \geq i$. The notation [ ] refers to the nearest integer and the weights are defined by $w_1 = k_w - (i - p)$ and $w_2 = k_w - (q - i)$. The first and last parts in (2) correspond to the interval that remains from the reference time series. The second part corresponds to the warped interval, which is a weighted average of the two nearest warped points of the reference time series.

Note that this transformation compresses ($k_w < 1$) or expands ($k_w > 1$) an interval of the reference series, thus the transformed series will be shorter or larger than the reference series. Two examples of synthetic warped versions of

a reference time series from the ArrowHead dataset from the UCR repository are shown in Figure 2.



(a) $k_w = 1.2$ in $[100, 200]$. (b) $k_w = 0.7$ in $[30, 100]$.

Figure 2: A reference time series from the ArrowHead dataset and two examples of synthetic warped version.

3.2 Scale

This transformation is a deformation of the series in the y-axis. It produces an upward or downward displacement (depending on the scale level) of the values of a series in a given interval. Time series with different scale levels can be found, for example, in the Adiac dataset from the UCR repository. In this dataset, images of 37 classes of diatoms (unicellular algae) are converted into time series, and time series from different classes differ in the scale in several intervals (see Figure 3). The interpretation of this fact is that these two diatoms have very similar shapes, but differ in the amplitude of the shape in some intervals. As such, the scale in those intervals is considered as relevant for the prediction.

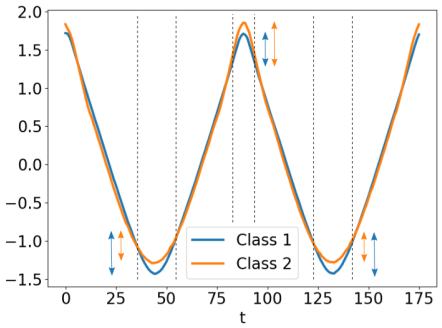


Figure 3: Two time series from different classes of the Adiac dataset that differ in the scale level in the marked intervals.

We propose the scale transformation of a reference time series $T = (t_1, \ldots, t_i, \ldots, t_l)$ as: given an interval in which the transformation is

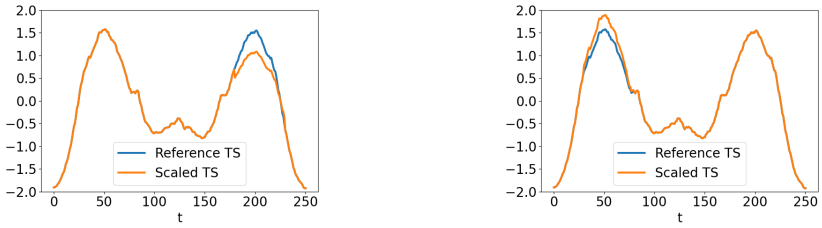
applied $[s, e]$ (such that $0 < s < e < l$) and a scale level k_s, the scaled version of the series is:

$$T' = (t'_1, \ldots, t'_i, \ldots, t'_l) \tag{3}$$

where

$$t'_i = \begin{cases} t_i * k_s, & \text{if } s \leq i \leq e \\ t_i, & \text{otherwise} \end{cases} \tag{4}$$

A reference time series from the ArrowHead dataset and two examples of synthetic scaled versions are shown in Figure 4.



(a) $k_s = 0.7$ in $[180, 230]$.

(b) $k_s = 1.2$ in $[30, 80]$.

Figure 4: A reference time series from the ArrowHead dataset and two examples of synthetic scaled versions.

3.3 Noise

The noise transformation consists of adding noise to the series in a given interval. An example of time series with different noise levels that determine the class can be seen in the ECG200 dataset from the UCR repository, for instance. In this dataset, the electrical activity of a heartbeat is recorded in two scenarios (normal heartbeat and a myocardial infarction). Figure 5 shows two time series from different classes that differ in the noise level in the marked interval. In these two examples, it can be seen that the heartbeat in both scenarios is very similar, but the normal heartbeat is smoother than the heartbeat of myocardial infarction. In this case, a noise transformation in this interval may make the classifier predict the series into another class and, hence, the noise level in the given interval is considered as discriminative.

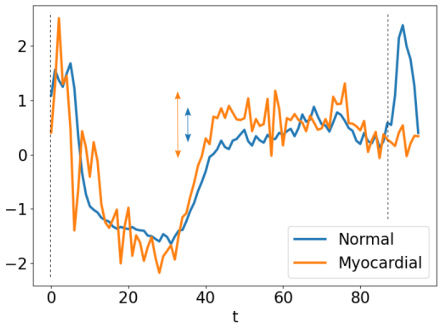


Figure 5: Two time series from different classes of the ECG200 dataset that differ in the noise level in the marked interval.

Given a reference time series $T = (t_1, \ldots, t_i, \ldots, t_l)$, an interval in which the transformation is applied $[s, e]$ (such that $0 < s < e < l$) and a noise level k_n, the noise-transformed version of the series is defined by:

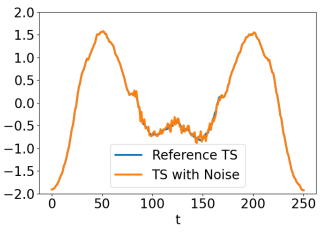
$$T' = (t'_1, \ldots, t'_i, \ldots, t'_l) \tag{5}$$

where

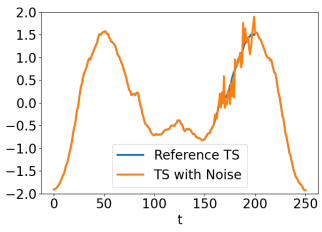
$$t'_i = \begin{cases} t_i + \mathcal{N}(0, \frac{A * k_n}{100}), & \text{if } s \leq i \leq e \\ t_i, & \text{otherwise} \end{cases} \tag{6}$$

with $A = |max(T) - min(T)|$ the amplitude of the series. As Equation 6 shows, the added noise is a Gaussian noise $\mathcal{N}(\mu, \sigma)$, with $\mu = 0$ and a σ that depends on the amplitude of the time series and the noise level k_n. In this way, for $k_n = 5$, for example, the standard deviation is set to 5% of the amplitude of the series.

A reference time series from the ArrowHead dataset and two examples of synthetic versions with noise are shown in Figure 6.



(a) $k_n = 2$ in $[80, 170]$.



(b) $k_n = 7$ in $[160, 200]$.

Figure 6: A reference time series from the ArrowHead datasets and two examples of synthetic versions with noise.

3.4 Slice

The slice transformation selects a subsequence of the time series and aligns it at the beginning of the reference time series. This transformation can be seen as selecting and shifting a subsequence of the time series. As the authors of [7] pointed out, for many time series that represent audio, video, or sensor recordings, the exact moment at which the recording starts and finishes may vary. For instance, for a time series that represents the audio recording of a person saying a word, the location of subsequence that represents the word may be different depending on the exact starting and ending point of the recording.

Sliced time series can be found, for example, in the AllGestureWiimoteY dataset from the UCR repository. This dataset consists of the y-axis recording of 10 individuals performing 10 different gestures measured by the Nintendo Wiimote controller. Two series from different classes are shown in Figure 7; it can be seen that both time series have very similar shapes, but the time series from class 5 is a sliced version of the time series from class 7. That is, if the recording of the time series from class 7 had started a bit later and had finished before, the classifier would probably have classified it in class 5. In this way, the marked interval and, in particular, its location are considered as relevant for the prediction.

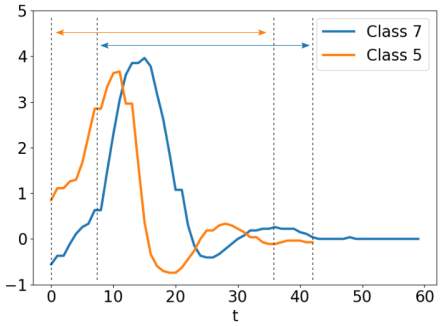


Figure 7: Two time series from the AllGestureWiimoteY dataset. The time series from class 5 can be seen as a sliced version of the time series from class 7.

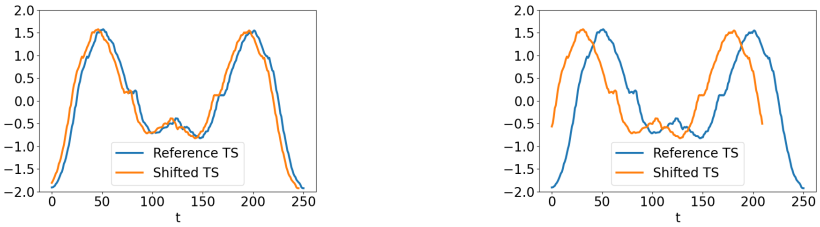
Given a reference time series $T = (t_1, \ldots, t_i, \ldots, t_l)$ and a random interval $[s, e]$, the slice transformation consists of removing the intervals $[1, s)$ and $(e, l]$ from the reference series. In this way, in contrast to the previous transformations, the slice transformations does not modify the series in the $[s, e]$ interval, but it is applied in $[1, s) \cup (e, l]$. The sliced version of T is defined by:

$$T' = (t'_1, \ldots, t'_i, \ldots, t'_{e-s}) \tag{7}$$

where

$$t'_i = t_{s+i}, \qquad i = 1, \ldots, e - s \tag{8}$$

This transformation involves the time axis of the series, and hence, the transformed time series are shorter than the reference series. A reference time series from the ArrowHead dataset and two examples of sliced versions are shown in Figure 8.



(a) Sliced in $[5, 250]$. (b) Sliced in $[20, 230]$.

Figure 8: A reference time series from the ArrowHead dataset and two examples of synthetic sliced versions.

To sum up, the proposed transformations enable the creation of synthetic time series that (i) have a semantic interpretation -as we have shown in the examples above- and (i) seem more realistic than those obtained by the perturbations proposed for other existing explanation methods for TSC, such as [4, 22]. In [20], the authors do not provide illustrative examples of the perturbed time series.

Note that our method can be applied to any time series from any type of TSC problem, but it is particularly interesting for problems in which the considered transformations are meaningful and semantically interpretable. Note that this requirement holds in most TSC problems (regardless the application domain), but there are some examples in which it is not fulfilled; in the StarLigthCurves dataset from the UCR repository, for instance, the time series represent the brightness of celestial objects as a function of time, and the noise transformation in an interval with very low brightness could produce a synthetic time series with a negative brightness values in this interval. In this case, our method with the noise transformation would not be suitable.

4 Time Series Classification Explanation Method

Given a time series in a labelled dataset, a classifier, and a transformation, the method provides a two level explanation: the high-level explanation describes the robustness of the prediction with respect to the transformation, while the low-level explanation displays the relevance of each region of the series in the

prediction.

4.1 High-level explanation

The high-level explanation can be summarized in 3 steps (see Figure 9):

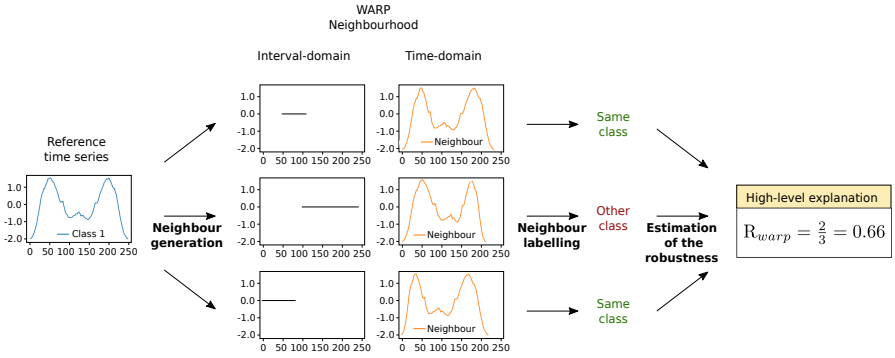


Figure 9: Scheme of the high-level explanation with an example of time series from the ArrowHead dataset and warp transformation with $k_w = 0.7$.

Neighbour generation. We create neighbour time series by sampling a random interval of the reference series and applying the transformation to this interval. For this, a random interval generation is needed, such that all the time indexes of a series have the same probability to appear in an interval. We employ an approach called *randomized unwrap the circle method* [31], which considers the intervals as wrapped around the perimeter of a circle (described in Algorithm 1).

Algorithm 1: Randomized unwrap the circle method

Input : Parameters of Beta prime distribution α, β
Output: Random interval in $[0, 1]$
Sample x from BetaPrime(α, β)
Sample u uniformly on $[-x, 1]$
Return $[0, 1] \cap [u, u + x]$

Neighbour labelling. In this step, the generated neighbours are labelled by the classifier. Some of the transformations involve deformations in the time axis of the time series, giving rise to neighbours with larger or shorter lengths than the reference time series. Since many benchmark classifiers for TSC are not adapted to varying length series [7], this leaves us two options: to employ a pre-processing step to equal the lengths of all the series, or to limit our method to classifiers that can handle series of varying lengths. The first option involves

adding information to the shorter series and removing information from the larger series, while the second works with all the information but restricts our method to a shorter number of classifiers. In this work, in order modify the time series as little as possible, the second option has been adopted. In addition, since we consider classifiers that are able to handle unequal length time series, our method can be directly applied to obtain explanations for this type of time series.

Estimation of the robustness. We consider that the prediction is robust with respect to the transformation if all the neighbours are classified into the same class of the reference time series. In the following, we call $I = I_{=} \cup I_{\neq}$ the set of generated random intervals, where $I_{=}$ refers to those intervals that result in neighbours from the same class of the reference series, and $I_{\neq}$ to those that result in neighbours from other classes. As such, in this step, the robustness of a prediction with respect to a transformation, R_{transf}, is quantified by measuring the percentage of neighbours that are classified into the same class of the reference time series:

$$\mathrm{R}_{transf} = |I_{=}|/|I|$$

where the operator $|\ |$ refers to the cardinal. R_{transf} varies in the range $[0, 1]$, where a value of 0 means that the prediction is completely sensitive to the transformation, since all the considered neighbours are labelled into another class. A value of 1, in contrast, means that the prediction is completely robust with respect to the transformation, since all the considered neighbours are labelled into the same class of the reference series.

4.2 Low-level explanation

The low-level explanation consists of computing the relevance of each region of the time series in the prediction. In this step, two possible scenarios are considered: the prediction is completely robust with respect to a transformation ($\mathrm{R}_{transf} = 1$), or the prediction is somewhat sensitive to a transformation ($\mathrm{R}_{transf} < 1$). In the first case, a transformation never affects the prediction, and hence, the low-level explanation is that there are no intervals that have a special impact on the prediction, with respect to the transformation. In the second case, the low-level explanation is computed following the next steps (see Figure 10).

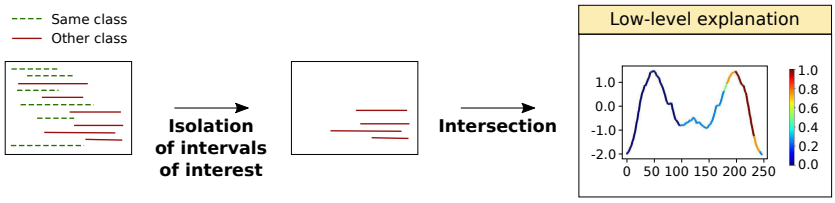


Figure 10: Scheme of the low-level explanation with an example of a time series from the ArrowHead dataset.

Isolation of intervals of interest. An interval is considered relevant for the prediction if a transformation applied to this interval changes the prediction of the classifier (i.e., those intervals in $I_{\neq}$). In some regions of the time series, however, there may be intervals both from $I_{\neq}$ and from $I_{=}$. In order to guarantee that a region of the series is relevant for the prediction, we define the intervals of interest as those intervals from $I_{\neq}$ that do not have many intervals from $I_{=}$ in the vicinity.

Specifically, an interval $i_{\neq}$ is considered of interest if it is mainly surrounded by intervals from $I_{\neq}$. Based on this idea, the procedure to identify the intervals of interest is summarized as follows:

(i) For each interval $i_{\neq} \in I_{\neq}$, find the K closest intervals in $I = I_{=} \cup I_{\neq}$ employing the Hausdorff distance[1].

(ii) If more than $K/2$ of those intervals belong to $I_{\neq}$, $i_{\neq}$ is considered an interval of interest. On the contrary, if at least $K/2$ of those intervals belong to $I_{=}$, $i_{\neq}$ is not considered an interval of interest.

Note that the distribution of the intervals of I between $I_{\neq}$ and $I_{=}$ varies for each time series. As such, using a fixed value of the parameter K for all the time series would bias the vicinity towards the predominant interval type ($I_{\neq}$ or $I_{=}$). Hence, we set the parameter K as a proportion of the number of intervals in $I_{\neq}$.

Intersection. This step consists of summarizing and representing the information of the interval of interest in the original time series. The idea is that, the more intervals of interest that contain a time index i, the more relevant this index is for the prediction. As such, given a time series $T = (t_1, \ldots, t_i, \ldots, t_l)$, we compute the number of isolated intervals that contain the index i for $i = 1 \ldots l$ and this values are stored in a vector

[1]Given two intervals $A = [a_1, a_2]$ and $B = [b_1, b_2]$ in $\mathbb{R}$, the Hausdorff distance [32] is defined by $d_H(A, B) = \max\{|a_1 - b_1|, |a_2 - b_{12}|\}$. Note that the slice transformation is applied in the union of intervals instead of in intervals (Section 3.4), thus the extension for unions [33] is used for this transformation.

$w = (w_1, \ldots, w_i, \ldots, w_l)$. The explanation, thus, can be seen as a weight vector w, in which w_i represents the relevance of the time index i in the prediction. Then, the time series is coloured depending on these values. The red colour indicates that these time indexes are contained in many relevant intervals, so this region is important for the prediction, while the blue colour, indicates the opposite. Note that the colorbar is normalized to $[0, 1]$.

Note that, given a time series, a transformation and a classifier, the cost of computing an explanation is dominated by the *neighbour labelling* step. In this step, the generated neigbours are labelled by the corresponding classifier, so the time that this process involves significantly varies depending on the classifier itself and the length of the time series.

5 Experimentation

In this section, we first present the experimental set-up, followed by three cases of study of the explanations provided by our method in some example time series chosen from the UCR repository. Then, a quantitative evaluation is presented, for which we adapt the evaluation methodology presented in [24]. This procedure quantifies the informativeness of explanation methods for TSC, and we adjust it to the context of the proposed transformations. All the code has been developed in Python and is publicly available[2].

5.1 Set-up

The experimentation has been carried out considering several parameters of the transformations. The warp and scale levels considered in this experimentation, k_w and k_s, are $\{0.7, 0.8, 0.9, 1.1, 1.2, 1.3\}$, while the considered noise levels, k_n are $\{1, 3, 5, 7, 9\}$. Other sets of parameters could be considered, but we limit our experimentation to the mentioned values; we think that they lead to a reasonable variety in the transformations, while creating realistic neighbours with no extreme modifications in the time series. Each transformation level is independently studied. The slice transformation does not depend on any parameter but, in order to ensure a minimum length in the generated neighbours, we set the minimum interval length to $0.3 * l$, where l is the length of the series that is the object of study.

Regarding the neighbour generation process, the size of the neighbourhood is set to 500, while the parameters in 1 are set to $\alpha = 8$ and $\beta = 18$. such that the probability of a index to be covered by an interval is 0.3 [31]. In this way, each time index appears in approximately 150 of the 500 generated neighbours. In the computation of the intervals of interest, the parameter K is set to $0.1^*|I_{\neq}|$.

[2]https://gitlab.bcamath.org/aabanda/tscexplanation.

Lastly, regarding the classifiers, as mentioned before, in this experimentation we employ classifiers that handle varying length time series. As such, 3 benchmark and diverse classifiers for TSC have been chosen: from the category of elastic or distance-based classifiers, the 1-NN-DTW distance, from shapelet-based classifiers, the Shapelet Transform (ST), and, from the dictionary-based category, the Bag-of-SFA-Symbols (BOSS).

5.2 Case of study

In this section, the proposed explanation method is studied in some example time series extracted from 3 datasets of the UCR repository: GunPoint, Coffee and ItalyPowerDemand. The GunPoint and Coffee datasets have been chosen because they have already been used before in methods concerning TSC explainability [18, 16, 14]. The ItalyPowerDemand dataset has been chosen with the aim of including a less used dataset from another more economic context; the energy consumption.

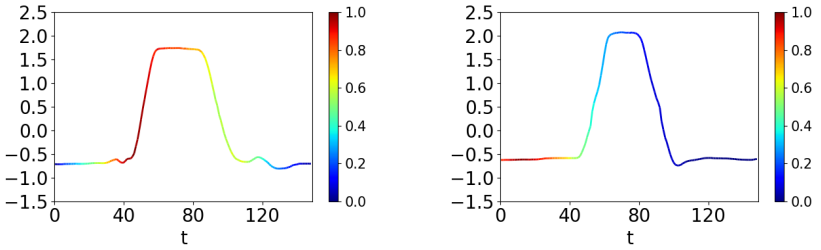
Note that our method provides an independent explanation for each transformation and they do not need to be consistent; each transformation leads to an explanation with a particular meaning. In the same manner, the explanations provided for different classifiers in the same time series can be different, since the explainability is dependent on the decision of the classifier.

5.2.1 GunPoint dataset

From the GunPoint dataset, introduced in Section 3.1, we have randomly[3] chosen a time series from each class for the examples shown in Figure 11. Figure 11a shows the explanation obtained for the warp transformation ($k_w = 0.7$) and the 1-NN-DTW classifier in a time series from the Gun class. The R_{warp} is 0.45, which indicates that the output of the classifier in this time series is quite sensitive to the warp transformation - more than a half of the neighbours generated by warping random intervals of the series are classified into the Point class-. Our method states that the most discriminant region is the part in which the subject raises the gun from the hip; if the subject does this movement faster, the time series is likely to be classified into the other class.

The explanation for a time series from the Point class for the scale transformation ($k_s = 1.2$) and the 1-NN-DTW classifier is shown in Figure 11b. The R_{scale} is 0.64, so the prediction is rather robust to the scale transformation (35% of the scale transformed neighbours are classified into the other class). The explanation indicates that the most discriminative region is the part in which the subject is pretending to hold the gun on the hip; if the vertical position of his/her hand was lower than what it is, the 1-NN-DTW would probably classify this time series into the other class.

[3]For the sake of reproducibility, the index of each example time series is specified: time series 52 (Fig. 11a) and 1 (Fig. 11b)



(a) Warp ($k_w = 0.7$) with the 1-NN-DTW classifier, R_{warp}= 0.45.

(b) Scale ($k_s = 1.2$) with the 1-NN-DTW classifier, R_{scale}= 0.64

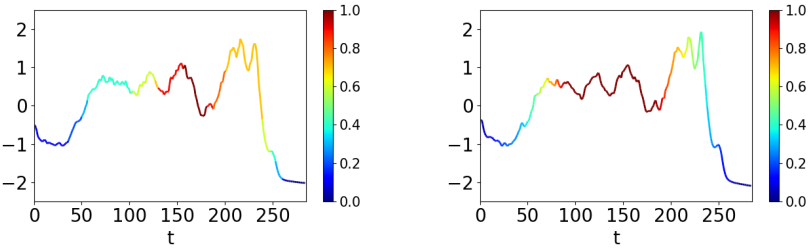
Figure 11: Two explanations provided by our method in a time series from the Gun class (a) and Point class (b) from GunPoint dataset.

5.2.2 Coffee dataset

The Coffee dataset is a binary dataset composed of time series that represent food spectrographs of two types of coffee: arabica and robusta. We have randomly[4] chosen a time series from each class for the examples shown in Figure 12. Figure 12a shows the explanation obtained for a time series from the Arabica class with the noise transformation ($k_n = 9$) and the BOSS classifier. In this case, the output of the BOSS classifier is very robust to the noise transformation ($R_{noise} = 0.94$); it is very difficult to mislead the classifier by adding noise to the time series. An interesting finding is that the explanation obtained by our method is very similar to that reported in [16] for a time series from the same class, even when the methods are fundamentally different. The authors point out that the highlighted regions correspond to the chlorogenic acid and caffeine contents of the coffee blends, i.e., the regions that discriminate between the Arabica and the Robusta coffee types [34].

An explanation obtained for a time series from the Robusta class with the slice transformation and the ST classifier is shown in Figure 12b. The prediction of the ST classifier in this time series is quite sensitive to the slice transformation, with a R_{slice} of 0.48. Analogously, the explanation obtained by our method is very similar to that presented in [16], where the highlighted regions correspond to the discriminative parts reported in [34].

[4]For the sake of reproducibility, the index of each example time series is specified: time series 52 (Fig. 11a) and 1 (Fig. 11b)



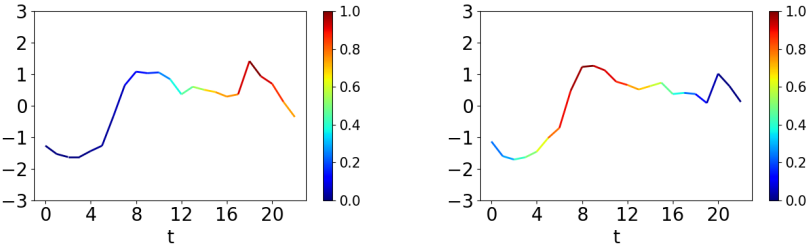
(a) Noise ($k_n = 9$) with the BOSS classifier, R_{noise}= 0.94.

(b) Slice with the ST classifier, R_{slice}= 0.48.

Figure 12: Two explanations provided by our method in a time series from the Arabica class (a) and Robusta class (b) from Coffee dataset.

5.2.3 ItalyPowerDemand dataset

ItalyPowerDemand is a dataset in which the time series represent the six-month electrical power demand (with 4 observations per month) in Italy. There are two classes, where the time series in class 1 represent the electric demand from October to March (cold semester), and the time series in class 2 from April to September (warm semester). We have computed our explanation in two[5] time series from the test set: a time series from the cold semester (Figure 13a) and a time series from the warm semester (Figure 13b).



(a) Warp ($k_w = 0.7$) with the ST classifier, R_{warp}= 0.67.

(b) Scale ($k_s = 0.7$) with the 1-NN-DTW classifier, R_{scale}= 0.77.

Figure 13: Two explanations provided by our method in a time series from the cold semester class (a) and warm semester class (b) from ItalyPowerDemand dataset.

In Figure 13a, the explanation for the warp transformation (with $k_w = 0.7$) and the ST classifier is shown for a time series from the cold semester. It can

[5]For the sake of reproducibility, the index of each example time series is specified. Time series 13 (Fig. 13a) and 1 (Fig. 13b).

be seen that the most relevant part is the peak in the interval $[17, 21]$ (from the beginning of February to the beginning of March). In particular, if it is compressed to a shorter interval, the ST classifier would label the time series into the warm semester class. From a semantic point of view, the interpretation is that this peak of the electric demand at the end of the semester is wider in the cold semester than in the warm semester. So, if we compress the time axis in this interval and narrow this peak, the ST would classified the time series in the warm semester.

In Figure 13b, instead, we have computed the explanation for the scale transformation (with $k_s = 0.7$) and the 1-NN-DTW classifier. The explanation shows that the most relevant part of the time series is in the interval $[6, 10]$ (from the beginning of May to the beginning of June). More concretely, if the subsequence in this interval is scaled down in the y-axis, the 1-NN-DTW classifier would label the time series into the cold semester class. From a semantic point of view, the interpretation is that the increase in the electric demand in the first part of the semester is lower in the cold semester than in the warm semester.

5.3 Quantitative evaluation on UCR

5.3.1 Experimental Design

In this section, the quantitative evaluation of the proposed method is presented, for which we will adapt the methodology presented in [24] to our context. Given a time series and an explanation, the authors propose perturbing the time series by adding noise to the important and non-important regions of the series. The idea behind their method is that perturbations on the important regions should change the class prediction more frequently than perturbations on the non-important regions.

The important and non-important regions are defined based on the distribution of the values of the weight vector w computed in Section 4: the $p\%$ most important indexes are those indexes with the $p\%$ highest values of w, and we will call the set of these indexes p_+. Analogously, the $p\%$ least important indexes are those corresponding to the $p\%$ lowest values of w, and we will call the set of these indexes p_-. In their work, however, the only considered perturbation is noise, and it is added independently to each time index. The adaptation to this evaluation method that we propose, allows the considered transformations to be employed as perturbations (not only noise).

The general evaluation procedure is summarized in Figure 14: given a classifier and a dataset divided into train/test sets, the classifier is trained using the train set and the explanations for all the time series in the test set, $w^1, \ldots, w^m$ (where m is the number of time series in the test set), are computed. Then, perturbed versions of the test set are computed for different values of p (in this work, 3 values of p are considered: 10, 50, 90). More specifically, if X is the test set, X^{p-} is a modified version of the test set in which the least important $p\%$ of the time indexes are perturbed in all the time

series of this test set. Analogously, X^{p+} is a modified version of the test set in which the most important $p\%$ of the time indexes are perturbed in all the time series of this test set. Then, the labels of all the series in the perturbed test sets are predicted by the classifier; given a perturbed test set, X^{10+} for instance, the classifier is employed to obtain the corresponding labels Y^{10+}. In order to measure the frequency in which the labels of a perturbed test set are different from the labels of the true test set, the consensus between them is defined by:

$$C^{p+} = \frac{1}{m} \sum_{s=0}^{m} \mathbb{1}_{y_s = y_s^{p+}}$$

where

$$\mathbb{1}_{y_s = y_s^{p+}} = \begin{cases} 1 & \text{if} \quad y_s = y_s^{p+} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The consensus varies from 0 to 1, where 0 means that all the labels in Y^{p+} are different from those in Y, while 1 means that all the time series in the perturbed test set are classified in the same class as the original time series. The case for perturbations p_- is analogous. The consensus is computed sequentially for the different values of $p = (10, 50, 90)$, such that C^{p-} and C^{p+} form two curves (as shown at the bottom of Figure 14).

The area under each curve, $eLoss^1$ and $eLoss^2$ correspondingly, is computed employing the trapezoidal rule. Lastly, the area between both curves (named in the original work as $\Delta eLoss$, equation 10) is calculated; if it is positive, the explanation is considered informative and if it is negative, uninformative.

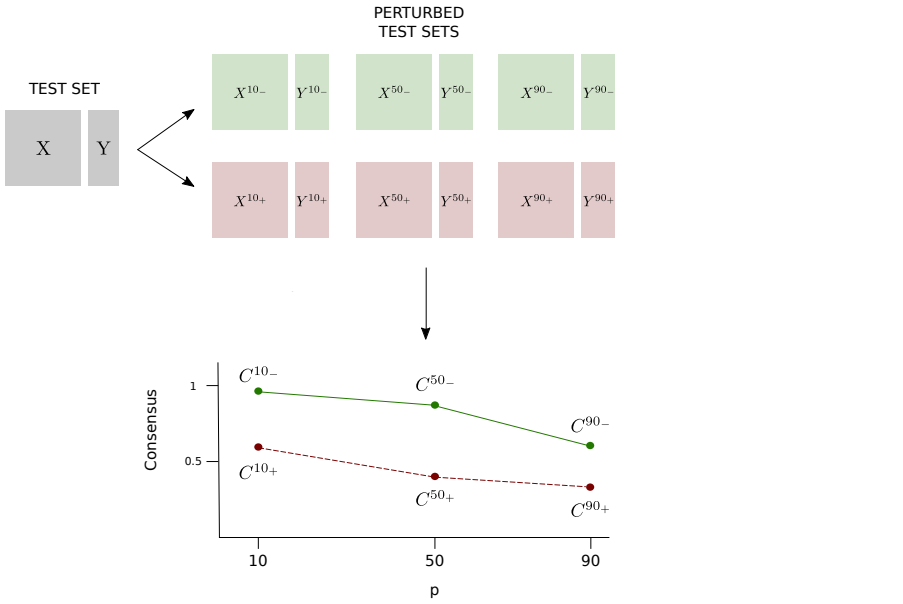


Figure 14: Scheme of the evaluation methodology.

$$\Delta eLoss = eLoss^1 - eLoss^2 \tag{10}$$

In the original work, given the $p\%$ most important indexes of w, the transformation is directly applied to these indexes. In the context of our work, the transformations are applied interval-wise instead of index-wise and, hence, an interval in the $p\%$ most important indexes is needed. As such, two scenarios are considered: 1) the $p\%$ most important time indexes form a single interval or 2) the $p\%$ most important time indexes form more than one interval. In the first scenario, a random interval is sampled in this range, and the perturbation is applied to this interval. In the second scenario, the longest interval is chosen, a random interval is sampled in this range and the perturbation is applied to this interval. The case of $p\%$ least important time indexes is analogous.

Lastly, note that the authors of the original work [24] employ different classifiers to obtain the explanations and to evaluate the explanations (i.e., to label the modified test sets). In our work, since the explanation depends on the classifier for which it has been calculated, the same classifier is employed for both purposes.

5.3.2 Results

The evaluation is carried out in 10 datasets from the UCR repository. Among the datasets with univariate time series of equal length with no missing values,

we have chosen a dataset from each of the categories defined in [35]. Given the high computational cost required for evaluating our method in an entire dataset, we have excluded those from the audio, sound and traffic categories because of their high dimensions. For the same reason, within the selected categories, we have chosen the dataset with the smallest dimension. The chosen datasets and the corresponding categories are: PowerCons (device), ECG200 (ECG), EOGHorizontalSignal (EOG), InsectEPGSmallTrain (EPG), PigCVP (hemodynamics), ArrowHead (image), GunPoint (motion), ItalyPowerDemand (sensor), CBF (simulated) and Coffee (spectro). Note that the proposed method is a general framework that could be used to explain the decision of a time series classifier applied in any application domain.

Recall that, in some cases, the prediction may be robust with respect to a given transformation and hence, there is no important region for this instance, classifier and transformation. As such, the evaluation is carried out using only those instances for which the R_{transf} is lower than 1. In order to provide more information to the reader, we also report the dataset-robustness with respect to a transformation, measured by the number of instances in the test set in which the prediction is robust to the transformation.

With the considered settings, $\Delta eLoss$ varies in the range $[-2, 2]$, and an explanation is considered informative if the corresponding $\Delta eLoss$ is positive. The evaluation procedure is repeated 10 times in order to remove the effect of randomness in the interval-wise perturbation process. Table 2 reports the mean $\Delta eLoss$ (and the mean dataset-robustness between parenthesis) for each combination of classifier and dataset across the different transformation levels. The detailed results for all the transformation levels are provided as supplementary material.

	Warp			Scale		
	1-NN-DTW	BOSS	ST	1-NN-DTW	BOSS	ST
PowerCons	**0.74** (0.88)	0.54 (0.68)	0.44 (0.66)	**0.68** (0.77)	0.65 (0.85)	0.62 (0.78)
ECG200	**0.52** (0.81)	0.45 (0.51)	0.42 (0.54)	**0.72** (0.69)	0.48 (0.75)	0.29 (0.88)
EOGHorizontalSignal	0.61 (0.95)	**0.63** (0.60)	0.08 (0.56)	**0.57** (0.58)	0.43 (0.48)	0.08 (0.57)
InsectEPGSmallTrain	- (1)	**0.34** (0.99)	0.18 (0.95)	- (1)	0.27 (0.75)	**0.49** (0.76)
PigCVP	**0.67** (0.95)	0.25 (0.72)	0.30 (0.16)	**0.38** (0.66)	0.14 (0.94)	0.04 (0.72)
ArrowHead	**0.61** (0.90)	0.42 (0.49)	0.19 (0.59)	**0.62** (0.41)	0.25 (0.49)	0.19 (0.86)
GunPoint	**0.63** (0.91)	0.46 (0.84)	0.36 (0.93)	**0.60** (0.39)	0.35 (0.87)	0.24 (0.82)
ItalyPowerDemand	0.28 (0.71)	**0.38** (0.29)	0.34 (0.73)	**0.53** (0.77)	0.28 (0.65)	0.45 (0.93)
CBF	0.41 (0.89)	0.50 (0.99)	**0.56** (0.88)	**0.57** (0.94)	0.47 (0.98)	0.45 (0.92)
Coffee	- (1)	**0.68** (0.87)	0.29 (0.69)	**0.66** (0.55)	0.39 (0.89)	0.18 (0.87)

	Noise			Slice		
	1-NN-DTW	BOSS	ST	1-NN-DTW	BOSS	ST
PowerCons	**0.38** (0.80)	0.16 (0.90)	0.07 (0.92)	**0.59** (0.01)	*-0.04* (0.07)	0.15 (0.30)
ECG200	**0.24** (0.64)	0.17 (0.72)	0.15 (0.69)	**0.44** (0.03)	0.05 (0.54)	0.08 (0.53)
EOGHorizontalSignal	**0.64** (0.68)	0.42 (0.52)	0.08 (0.57)	**0.14** (0.34)	*-0.10* (0.30)	0.01 (0.15)
InsectEPGSmallTrain	- (1)	**0.24** (0.99)	0.17 (0.77)	- (1)	**0.82** (0.86)	0.58 (0.50)
PigCVP	**0.50** (0.78)	0.31 (0.91)	0.08 (0.68)	0.07 (0.16)	**0.63** (0.83)	*-0.11* (0.12)
ArrowHead	**0.54** (0.52)	0.27 (0.36)	0.43 (0.56)	0.10 (0.07)	**0.24** (0.27)	*-0.10* (0.50)
GunPoint	**0.45** (0.57)	0.05 (0.73)	0.43 (0.51)	*-0.01* (0.02)	0.65 (0.81)	**0.75** (0.49)
ItalyPowerDemand	**0.10** (0.75)	0.06 (0.68)	0.09 (0.86)	**0.57** (0.01)	*-0.04* (0.07)	0.22 (0.30)
CBF	0.12 (0.98)	0.11 (0.99)	**0.14** (0.92)	**0.41** (0.01)	0.08 (0.31)	0.04 (0.01)
Coffee	**0.14** (0.65)	0.08 (0.94)	0.09 (0.52)	**0.65** (0.00)	0.05 (0.86)	0.56 (0.86)

Table 2: The mean $\Delta eLoss$ obtained for the considered transformations across the different transformation levels. The mean dataset-robustness is shown between parenthesis. For each dataset and transformation, the best $\Delta eLoss$ is shown in bold.

Table 2 shows that the mean $\Delta eLoss$ is positive in almost all the dataset-transformation-classifier combinations, which means that the explanation is informative. In fact, there are only 6 cases (of the 120 cases) in which it is negative (they are marked in italics). The results clearly validate our explanation method, since, perturbing the important parts change the prediction of the classifier more frequently than perturbing the non-important parts, regardless of the classifier, transformation or dataset.

If we analyze the results transformation-wise, it can be seen that all the negative values, that is, the non-informative explanations, are for the slice transformation. However, there are also high positive values for this transformation. This could indicate that this particular transformation is not suitable for the datasets in which our method obtains negative values. If we analyze the results with respect to the classifier, instead, in the majority of the datasets and transformations, the best explained classifier (highlighted in bold) is the 1-NN-DTW, which obtains the highest values in 28 of the 40 dataset-transformation combinations. It is followed by the BOSS classifier, which obtains the best explanations in 8 cases, while the ST classifier wins only in 4 cases.

The dataset-robustness is generally very high (except for the slice transformation) in all the cases, which means that it is rather hard to change the prediction of the classifiers employing the considered transformations. This

can be due to the distribution of the classes within each dataset, or due to the capacity of the classifiers employed in this work to handle the transformations.

It is worth noting that the dataset-robustness depends on the parameter of the transformation (as shown in the tables presented in the supplementary material); for the warp and scale transformations, the closer the parameter is to 1, the greater the dataset-robustness, while for the noise transformation, the lower the parameter, the greater the dataset-robustness (which seems reasonable in both cases, since those are the parameters that involve the slightest modifications).

There are some interesting findings related with the classifiers that deserve attention; for example, the prediction of the 1-NN-DTW classifier is found to be robust to all the transformations in the InsectEPGSmallTrain dataset and for the warp transformation in the Coffee dataset for all the considered transformations levels. In addition, the 1-NN-DTW is found to be the most robust classifier for the warp transformation in 7 of the 10 datasets, which backs our expectations, since it is an elastic classifier specifically designed to deal with time warping. Lastly, the BOSS classifier is known to be robust to noise [4], but in our experimentation it is found to be the most robust classifier with respect to the noise transformation in only 5 of the 10 considered datasets.

6 Conclusions and Future Work

In this work, an explanation method for time series classification that provides realistic -and specific to time series- explanations is proposed. For this, 4 transformations for time series are presented (warp, scale, noise and slice) and a synthetic neighbourhood of a time series is created by applying these transformations to random intervals of the series. The method provides explanations at two levels: in the high-level, the robustness of a classifier's prediction with respect to a given transformation is measured, while in the low-level, the relevance of each region of the series in the prediction is computed.

In the experimentation, we first visually validate the explanation provided by our method in some time series extracted from datasets of the UCR repository. Taking advantage of the semantic meaning of the time series, our methods show, for example, that the speed of the action recorded in the GunPoint dataset is discriminant. In the latter, the proposed method is quantitatively evaluated by adapting an existing evaluation methodology [24] to the context of our transformations. The methodology consists of perturbing the time series in the important and non-important regions (according to the explanation), and checking whether the perturbations in the important regions change the prediction of the classifier more frequently than perturbations in the non-important regions. The results show that this holds in almost all the considered combinations of dataset-classifier-transformation, which confirms the informativeness of our method. Moreover, the dataset-robustness

(measured by the number of time series in the test set for which the prediction is robust to a transformation), give some insights into how robust the classifiers are to the considered transformations; the 1-NN-DTW classifier, for example, which is supposed to be very robust to time warping, is shown to be the most robust classifier with respect to the warp transformation in 7 of the 10 considered datasets.

Lastly, there are many possible directions for the future work. For instance, it could be interesting to extend our method to the multi-variate time series classification scenario. Also, given that the time series found in many real-life scenarios are streaming series [36], a challenging future work could be to extend our method to this type of time series by considering the corresponding specificities. Moreover, in this method, the transformations have been studied independently, while it could be interesting to study other transformations or combinations of transformations, for example, by successively applying different transformations to random intervals. In the same manner, other time series generation techniques (such as time series bootstrapping [29]) could be used to generate the neighbourhood of a given instance. To conclude, we think that our method is a first attempt at empirically studying the robustness in TSC, and it could be interesting to continue in this direction.

Acknowledgments

This research is supported by the Basque Government through the BERC 2018-2021 program and by Spanish Ministry of Economy and Competitiveness MINECO through BCAM Severo Ochoa excellence accreditation SEV-2017-0718, as well as through project TIN2017-82626-R funded by (AEI/FEDER, UE) and acronym GECECPAST. In addition, by the Research Group IT1244-19 programs (Basque Government), PID2019-104966GB-I00 (Spanish Ministry of Economy, Industry and Competitiveness) and Elkartek project 3KIA (KK2020/00049). A. Abanda is also supported by the Grant BES-2016-076890. The authors would like to thank the people who contributed to the UCR time series repository.

References

[1] A. Barredo, N. Díaz-Rodríguez, J. Del, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable Artificial Intelligence (XAI): Concepts , Taxonomies, Opportunities and Challenges Toward Responsible AI, Information Fusion (2020) 82–115doi:10.1016/j.inffus.2019.12.012. URL https://doi.org/10.1016/j.inffus.2019.12.012

[2] T. Narayan, M. Tscherepanow, B. Wrede, A Saliency Map based on Sampling an Image into Random Rectangular Regions of Interest, Pattern Recognition 45 (9) (2012) 3114–3124. doi:10.1016/j.patcog.2012.02.

009.
URL http://dx.doi.org/10.1016/j.patcog.2012.02.009

[3] M. T. Ribeiro, C. Guestrin, Why Should I Trust You? Explaining the Predictions of Any Classifier, The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016) 1135–1144.

[4] M. Guilleme, L. Roze, V. Masson, A. Termier, Agnostic local explanation for time series classification, International Conference on Tools with Artificial Intelligence (ICTAI) (2019) 432–439doi:10.1109/ICTAI.2019.00067.

[5] A. Bagnall, J. Lines, W. Vickers, E. Keogh, The UEA & UCR Time Series Classification Repository.
URL www.timeseriesclassification.com

[6] P. Esling, C. Agon, Time-Series Data Mining, ACM Computing Surveys 45 (1) (2012) 1–34. doi:10.1145/2379776.2379788.
URL http://dl.acm.org/citation.cfm?doid=2379776.2379788

[7] C. W. Tan, E. Keogh, G. I. Webb, Time Series Classification for Varying Length Series, arXiv:1910.04341v1.

[8] A. L. Guennec, S. Malinowski, R. Tavenard, Data Augmentation for Time Series Classification using Convolutional Neural Networks, International Workshop on Advanced Analytics and Learning on Temporal Data (AALTD16).

[9] A. Blázquez-García, A. Conde, U. Mori, J. A. Lozano, Water leak detection using self-supervised time series classification, Information Sciences 574 (2021) 528–541. doi:10.1016/j.ins.2021.06.015.

[10] L. Ye, E. Keogh, Time series shapelets: A novel technique that allows accurate, interpretable and fast classification, Data Mining and Knowledge Discovery 22 (1-2) (2011) 149–182. doi:10.1007/s10618-010-0179-5.

[11] Y. Wang, E. Fromont, S. Malinowski, Learning Interpretable Shapelets for Time Series Classification through Adversarial Regularization, arXiv preprint arXiv:1906.00917arXiv:arXiv:1906.00917v2.

[12] R. Assaf, I. Giurgiu, F. Bagehorn, A. Schumann, MTEX-CNN: Multivariate Time series EXplanations for Predictions with Convolutional Neural Networks, IEEE International Conference on Data Mining (ICDM) (2019) 952–957doi:10.1109/ICDM.2019.00106.

[13] J. L. B, E. Zugasti, X. D. Carlos, Contrastive Explanations for a Deep Learning Model on Time-Series Data, Vol. 1, Springer International Publishing, 2020. doi:10.1007/978-3-030-59065-9.
URL http://dx.doi.org/10.1007/978-3-030-59065-9{_}19

[14] P. Senin, SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model, IEEE International Conference on Data Mining (ICDM) (2013) 1175–1180doi:10.1109/ICDM.2013.52.

[15] I. Karlsson, J. Rebane, P. Papapetrou, Explainable Time Series Tweaking via Irreversible and Reversible Temporal Transformations, IEEE International Conference on Data Mining (ICDM)doi:10.1109/ICDM.2018.00036.

[16] T. Le, N. Severin, G. Iulia, I. Martin, G. Ifrim, Interpretable Time Series Classification using Linear Models and Multi-Resolution Multi-Domain Symbolic Representations, Data Mining and Knowledge Discovery 33 (4) (2019) 1183–1222. doi:10.1007/s10618-019-00633-3.
URL https://doi.org/10.1007/s10618-019-00633-3

[17] E. Delaney, D. Greene, M. T. Keane, Instance-Based Counterfactual Explanations for Time Series Classification, International Conference on Case-Based ReasoningarXiv:arXiv:2009.13211v1.

[18] H. Ismail, F. Germain, L. I. P.-A. Muller, J. Weber, Deep Learning for Time Series Classification: a Review, Data Mining and Knowledge Discovery 33 (4) (2019) 917–963. doi:10.1007/s10618-019-00619-1.
URL https://doi.org/10.1007/s10618-019-00619-1

[19] Z. Wang, W. Yan, T. Oates, Time Series Classification from Scratch with Deep Neural Networks: a Strong Baseline, International Joint Conference on Neural Networks (IJCNN) (2017) 1578–1585.

[20] J. Crabb, M. Van der Schaar, Explaining Time Series Predictions with Dynamic Masks, International Conference on Machine LearningarXiv:arXiv:2106.05303v1.

[21] I. Karlsson, P. Papapetrou, H. Boström, Generalized random shapelet forests, Data Mining and Knowledge Discovery 30 (5) (2016) 1053–1085. doi:10.1007/s10618-016-0473-y.

[22] F. Mujkanovic, V. Doskoč, M. Schirneck, TimeXplain – a Framework for Explaining the Predictions of Time Series Classifiers, arXiv preprint arXiv:2007.07606 (2020) 1–17arXiv:arXiv:2007.07606v1.

[23] R. Guidotti, A. Monreale, F. Spinnato, D. Pedreschi, F. Giannotti, Explaining Any Time Series Classifier, 2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI) (2020) 167–176doi:10.1109/CogMI50398.2020.00029.

[24] T. T. Nguyen, T. L. Nguyen, G. Ifrim, A Model-Agnostic Approach to Quantifying the Informativeness of Explanation Methods for Time Series Classification, International Workshop on Advanced Analytics and Learning on Temporal Data (AALTD20).

[25] S. M. Lundberg, S.-i. Lee, A Unified Approach to Interpreting Model Predictions, Proceedings of the 31st international conference on neural information processing systems (Section 2) (2017) 1–10. arXiv:arXiv:1705.07874v2.

[26] G. E. Hinton, R. R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science 313 (5786) (2016) 504–507.

[27] W. Härdle, J. P. Horowitz, J., Kreiss, Bootstrap Methods for Time Series, International Statistical Review 30 (2003) 3–26. doi:10.1016/B978-0-444-53858-1.00001-6.

[28] S. Lahiri, Resampling Methods for Dependent Data, Vol. 46, 2003. doi:10.1198/tech.2004.s795.

[29] D. Rajapaksha, C. Bergmeir, LIMREF: Local Interpretable Model Agnostic Rule-Based Explanations for Forecasting, with an Application to Electricity Smart Meter Data, 2022.

[30] J. Yoon, D. Jarrett, Time-series Generative Adversarial Networks, Advances in Neural Information Processing System (NeurIPS) (2019) 1–11.

[31] L. Devroye, P. Epstein, J.-R. Sack, On Generating Random Intervals and Hyperrectangles, Journal of Computational and Graphical Statistics 2 (3) (1993) 291–307.

[32] F. Hausdorff, Set Theory, 2nd Edition, Republished by Americal Mathematical Society (AMS) – Chelsea 2005, 1962.

[33] M. F. Barnsley, Superfractals, Australian National University, Canberra.

[34] R. Briandet, E. K. Kemsley, R. H. Wilson, Discrimination of Arabica and Robusta in Instant Coffee by Fourier Transform Infrared Spectroscopy and Chemometrics, Journal of Agricultural and Food Chemistry (1996) 170–174.

[35] A. Bagnall, J. Lines, W. Vickers, E. Keogh, The UEA and UCR time series classification repository.
URL http://www.timeseriesclassification.com

[36] B. Hu, Y. Chen, E. Keogh, Classification of streaming time series under more realistic assumptions, Data Mining and Knowledge Discovery 30 (2) (2016) 403–437. doi:10.1007/s10618-015-0415-0.