Information Sciences 645 (2023) 119334

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Explaining time series classifiers through meaningful perturbation and optimisation

Han Meng [a,b,*], Christian Wagner [b], Isaac Triguero [a,b,c,d]

[a] Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, United Kingdom
[b] Lab for Uncertainty in Data and Decision Making (LUCID), School of Computer Science, University of Nottingham, United Kingdom
[c] DaSCI Andalusian Institute in Data Science and Computational Intelligence, University of Granada, Granada, Spain
[d] Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain

ARTICLE INFO

Keywords:
Time series classification
Post-hoc explanation
Saliency-based explanation
Perturbation method
Optimisation approach

ABSTRACT

Machine learning approaches have enabled increasingly powerful time series classifiers. While performance has improved drastically, the resulting classifiers generally suffer from poor explainability, limiting their applicability in critical areas. Saliency-based methods designed to highlight the critical features are one of the most promising approaches to improving this explainability. Here, current techniques commonly rely on artificially perturbing the features, using, for example, random noise or 'zeroing' these features. We first demonstrate that an important drawback of these methods is that the perturbations used can result in unrealistic assessments of the classifier, since the perturbations force the data outside their original distribution. We articulate how this can result in poor identification of critical features, and hence misleading explanations. In order to address this issue and identify the most important features for the output of a black-box model, we propose a dual approach through meaningful perturbation and optimisation. First, leveraging a mechanism originally proposed in image analysis, a generative model is trained to create within-distribution perturbations of the input. These are then used to reliably evaluate whether a set of features is critical. Second, a greedy-based segmentation and identification strategy is proposed to search for the smallest set of critical features. Experiments show that the proposed approach addresses the out-of-distribution problem and identifies fewer critical features than existing methods. In combination, both aspects of the proposed approach offer a qualitative advance towards generating meaningful and robust explanations in the context of time series classification.

1. Introduction

Time Series Classification (TSC) is a hot research area in machine learning [1]. Unlike time series forecasting, which makes predictions based on historical records, TSC focuses on categorising sequences of real values recorded in a temporal order; it is encountered in various real-world applications [2,3]. In recent years, with the evolution of technology and the wide accessibility of sensor data, e.g. via the Internet of Things, more and more time series data are becoming available. The need to exploit these data

* Corresponding author at: Computational Optimisation and Learning (COL) Lab and Lab for Uncertainty in Data and Decision Making (LUCID), School of Computer Science, University of Nottingham, United Kingdom.

E-mail addresses: Han.Meng1@nottingham.ac.uk (H. Meng), Christian.Wagner@nottingham.ac.uk (C. Wagner), Isaac.Triguero@nottingham.ac.uk (I. Triguero).

https://doi.org/10.1016/j.ins.2023.119334
Received 14 September 2022; Received in revised form 12 June 2023; Accepted 16 June 2023

Available online 22 June 2023

drives advances of TSC models. Among such models, Artificial Intelligence (AI) techniques have achieved outstanding performance [4,5]. However, many of these methods, e.g. deep learning, are black boxes, and their lack of explainability becomes an impediment to deploying them in real-world applications, especially in critical areas, such as finance [2] and medicine [3]. Here, users who employ these AI techniques need the reasons for the classification results to support them in making decisions or taking actions, which has boosted the development of eXplainable AI (XAI).

Currently, XAI has undoubtedly achieved notable momentum in the AI community. One of the main aims of XAI is to provide explanations for humans to help them understand and gain the appropriate trust in modern AI models. In the XAI literature, explainability refers to two distinct high-level concepts: post-hoc explainability or interpretability, and transparency (also referred to as self-explainability, e.g. in [6]). The former captures the ability to explain a given output generated by an AI model, while the latter describes the property of an AI model as being interpretable by a human,[1] i.e. including its components, mechanisms and training process. With respect to the former, post-hoc methods can usually be applied to a wide range of algorithms since many post-hoc methods are model-agnostic [7]. Moreover, post-hoc methods will not affect the predictive performance of a classifier, while some self-explainable classifiers sacrifice their performance for interpretability [8]. Although it is contested whether self-explainable methods have to sacrifice their performance for interpretability [9], post-hoc analysis approaches may relieve the builders of the model of the burden of thinking about how to provide explanations and enable them to focus on the predictive performance only. In addition, post-hoc analysis can provide different kinds of explanations, such as sample-based counterfactual explanations [10] or explanations of the importance of each feature [11,12]. However, post-hoc analyses also face the risk of generating unreliable explanations, which means that the explanations provided by post-hoc approaches may not faithfully reflect how the classifier works [9].

Based on their type, the explanations provided are classified into global explanations and local explanations [6]. Global explanations aim to reveal the global behaviour of the classifier on the whole dataset, while local explanations focus on the behaviour on a specific instance. Due to the heterogeneity of real-world datasets, a model might focus on different features when making predictions on different sets of instances. Thus, the development of local instance-wise explanations has drawn increasing attention in recent years [13], and is the main focus of the present paper.

Saliency (or feature importance) methods are among the most common approaches to providing explanations through post-hoc analysis, where the most important features are identified for a certain input. They were originally proposed to explain computer vision and language processing models and have also been adopted in the context of TSC [14,15]. In time series, the real values at certain time steps can be regarded as features, similar to pixels in images. Typical ways to obtain saliency-based explanations in TSC include gradient backpropagation-based methods [4] and perturbation-based methods [16]. Gradient-based methods compute the gradients of the membership of a given class w.r.t. the input features to explain the classifier. Therefore, they can only be applied to classifiers whose gradients are accessible. In contrast, perturbation-based methods are usually model-agnostic. They perturb certain features and then measure the changes in the classification. In time series, most works perturb the features by replacing them with zeros [16]. This is useful, as it allows the discovery of saliency by flagging features which are (or are not) relevant to a given classification. However, they may result in inputs that are not aligned with the distribution of the training dataset. In the XAI field, this is known as the out-of-distribution (OOD) problem [17]. It has been widely recognised in image [18] and language processing [19] models, but in the context of TSC, it has not been articulated or received much attention. Building on our preliminary work [20], the present paper mitigates the OOD problem by putting forward a novel generative model to learn the distribution of the training dataset in order to generate within-distribution perturbed inputs for saliency discovery.

Another challenge for saliency discovery is to efficiently identify the smallest group of salient features. These features are necessary for a classifier to predict a specific output. Inspired by Fong's work in computer vision [21], saliency discovery in the TSC context has been cast as an optimisation problem [14], which can informally be described as: *what is the smallest feature set of the current input that would maximally change the classifier's output?* The solution to such an optimisation problem provides a pathway to help users focus on the minimal number of features necessary for the classifier to produce one output rather than another, something which helps identify the reasons for this output. However, this solution can only be attained by an exhaustive search of a feature space whose size increases exponentially with the dimension of the input features, and is an NP-hard problem [22–24]. One group of current solutions uses gradient descent techniques, where an auxiliary neural network is trained to identify the salient features [14,18,21]. However, in order to optimise the auxiliary neural network, these methods need the gradients of the classifiers. Another group approximates the optimal solution to the optimisation problem by using search-based methods, including Beam Search [23], Greedy method [24] and Genetic Algorithm [22]. These methods do not need the inner structures of the classifiers and can be applied to any classifier. To the best of our knowledge, such search-based methods have not been explored in the context of TSC.

In the present paper, we propose a framework to tackle the OOD problem and the difficulty of searching for the smallest salient feature set in the context of TSC. The specific contributions of this paper are summarised as follows:

- A novel model-agnostic framework is proposed to explain TSC through a post-hoc approach.
- The OOD problem in explaining TSC is articulated and addressed by designing a generative model to generate within-distribution perturbed inputs.
- A novel greedy-based segmentation and identification search strategy is proposed to identify the key features that are necessary for the classifier to produce a particular output.

[1] The notion of who this human is, is a crucial aspect of assessing a model's transparency, i.e. it may be transparent to its creating engineer, but not to an end user.

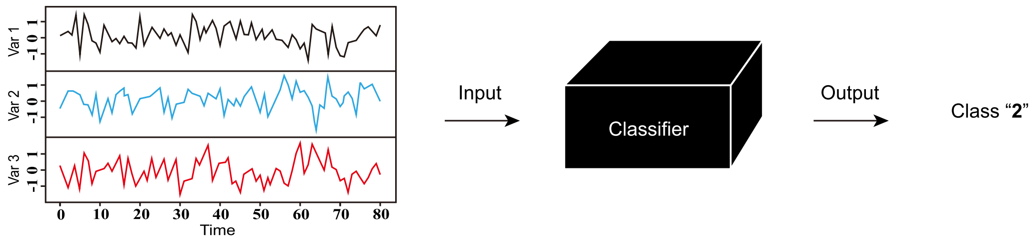


Fig. 1. An output, $p(c|\boldsymbol{x})$ = "2", is produced by a classifier for a MTS. This MTS has 3 variables and the length of these sequences is 80, which is expressed by $\boldsymbol{x} \in \mathbb{R}^{3\times 80}$.

- In addition to saliency explanations, this framework also provides counterfactual explanations based on the identified features and the proposed generative model, which provides the users more insights into the working mechanisms of the classifier.

The remainder of this paper is organised as follows. Section 2 gives the background of this work. Then, in Section 3, the proposed framework is described in detail. Section 4 and Section 5 provide detailed information about the experimental design and the thorough analysis of the proposed framework, respectively. Section 6 provides the conclusion and descriptions of avenues for future work.

2. Background

This section gives a brief overview of the work related to XAI in the context of TSC. First, the preliminaries for TSC are provided in Section 2.1. After that, the classic saliency-based explaining methods adopted to explaining TSC are reviewed in Section 2.2. Then, optimisation-based saliency identification techniques are introduced, and their advantages as well as challenges are discussed in Section 2.3. Finding the smallest subset of relevant features may resemble the problem of feature selection [25], but it has different goals and faces different challenges, which we believe necessary and useful to clarify. Therefore, we analyse these differences in Section 2.4.

2.1. Time series classification

A time series is a sequence of real values recorded in temporal order. A time series can be univariate, where the values are collected from only one sensor, or multivariate, where the real values are collected from multiple sensors. Nowadays, with the advancement of technique, more and more multivariate time series are being collected, which makes classification as well as its explanation more challenging. An example of a multivariate time series (MTS) can be seen in Fig. 1. In this paper, a time series is denoted by $\boldsymbol{x} \in \mathbb{R}^{D\times T}$, where D is the number of variables (or the number of sensors from which the values are collected; for univariate time series, $D = 1$) and T is the total number of time steps. In this paper, a value at a certain time step is regarded as one feature. For example, $\boldsymbol{x}_{i,t}$ is the feature representing the value of the i-th variable at time t. A TSC can be denoted by $p(c|\boldsymbol{x})$, which predicts the class c of the input time series $\boldsymbol{x}$. The remainder of this paper is going to target MTS, but the proposed explanatory framework is also applicable to univariate time series.

2.2. Classic saliency-based explanation

Saliency-based methods are one of the most commonly adopted approaches to explain black-box classifiers [15]. The term "saliency" was originally used in explaining image models, where the most important pixels for an output produced by a classifier are highlighted in a saliency map [26]. In classic saliency-based explanations, a saliency map is obtained by assigning each input pixel an importance score, which describes the influence of that pixel on the classifier's output. However, this idea is also applicable to explaining TSC. In a saliency map, the most important features are highlighted. An example of a classic saliency-based explanation for a TSC is shown in Fig. 2a. Currently, methods providing classic saliency-based explanations for black-box classifiers can be roughly categorised into three groups: *gradient-based* methods, *perturbation-based* methods, and the recently developed *meaningful perturbation-based* methods.

Gradient-based methods: These methods are based on the assumption that when a salient feature varies locally, it will cause a significant change in the output. The importance of one feature is evaluated by the gradient of the membership of a given class in the output w.r.t. this feature. A large gradient is translated into a salient feature. Popular gradient-based methods include Simple Gradient (SG) [26], Integrated Gradient (IG) [27], DeepLIFT [28], and others. These methods can provide saliency-based explanations very quickly but their explanations might be misleading [29]. It is known that SG methods fail to model saturation [28]. Taking a model, $y = \max(0, 1 - x_1 - x_2)$, for example, as long as $x_1 + x_2 > 1$, the gradient of the output w.r.t. the input features will always zeros, because in this case the model is "saturated" and its output will be always "0" no matter what the values of x_1 and x_2 are. IG [27] is proposed to mitigate this issue. This method numerically calculates the path integral of the gradients of a certain output w.r.t. an input feature to evaluate the importance of this feature. The path starts from a user-defined baseline and ends with the current input. Therefore, IG needs a baseline to generate explanations. The authors of [27] set the baseline to "zeros" input, which might

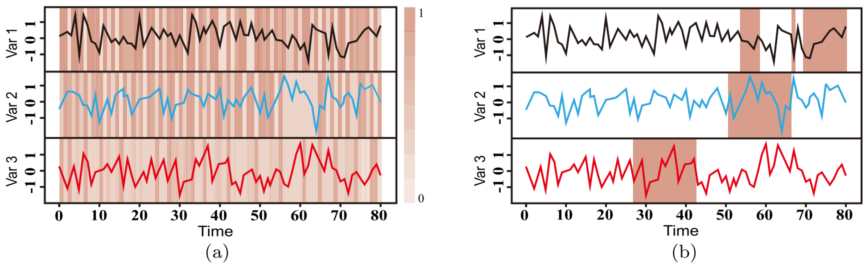


Fig. 2. (a) A classic saliency-based explanation; (b) An optimisation-based saliency-based explanation. Typical saliency-based explanations focus on assigning each feature with an importance score, while optimisation-based saliency-based explanations focus on identifying a group of features rather than exactly estimating the importance of each feature.

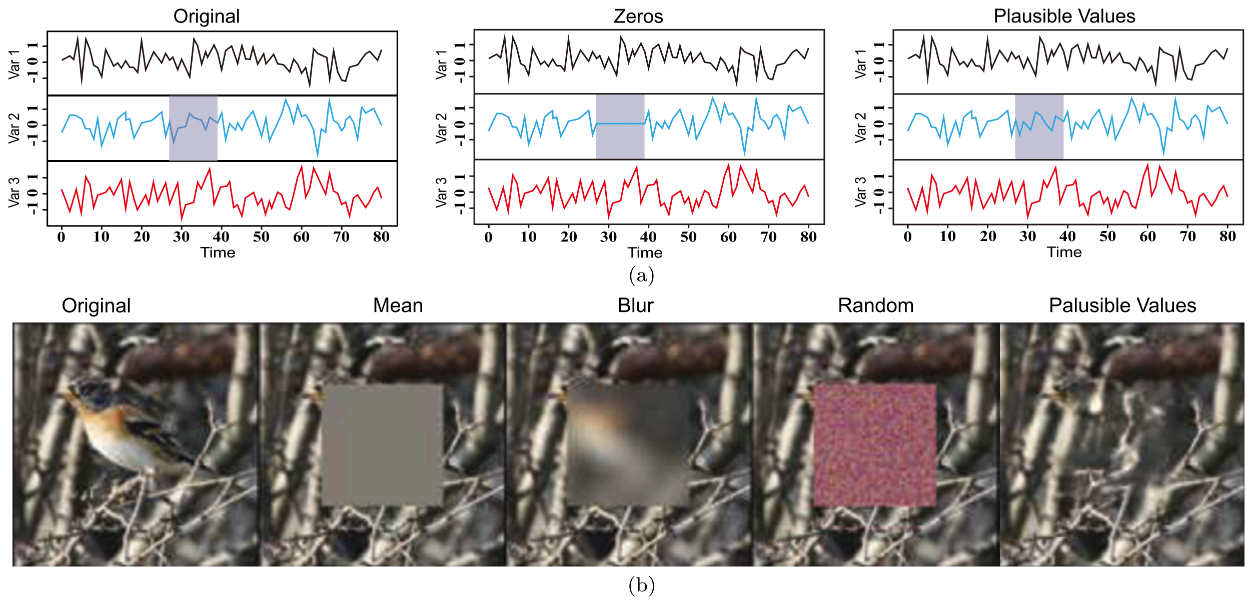


Fig. 3. (a) Time series perturbed inputs generated by replacing features with zeros and plausible values that inferred from the training dataset; (b) An example of the OOD problem in an image classification problem, where the centre region is perturbed by artificial strategies, including mean, blur, random, and plausible values inferred from the dataset. (This image is taken from [18]).

risk feeding classifiers with inputs that classifiers did not observe during the training stage, which is known in the XAI domain as 'the OOD problem'. Different baselines will result in different explanations, but the guidance for choosing a suitable baseline has not been developed [30]. Similar to IG, DeepLIFT also needs to choose suitable baselines, which is not an easy task. In addition, since these methods calculate the saliency based on gradients, they can only be applied to classifiers for which the gradients are accessible.

Perturbation-based methods: These methods perturb the input features and then measure the changes in the classifiers' outputs. These methods are based on the assumption that there are salient features whose perturbation will produce significant changes in the output. LIME [11] is one of the most prominent of these frameworks and can also be regarded as a kind of perturbation-based method, since the first step in the LIME framework is to generate perturbed inputs. The difference between LIME and typical perturbation-based methods is that LIME needs to train a local proxy model to evaluate the importance of the features rather than directly measuring the changes in the outputs. When using perturbation-based methods, one of the key points is how to perturb the features. Currently, in the context of TSC, the features are perturbed by replacing them with zeros [31,16] or with the weighted average of their neighbouring features [14]. However, these artificial perturbation operations might generate OOD inputs, as is shown in Fig. 3a. In the context of time series, the OOD problem might not be as obvious to humans as it is in image problems, as shown in 3b, but it might result in the same meaningless explanations as those in image problems. From Fig. 3b, the human might easily judges that the artificially generated perturbed inputs are not aligned with the distribution of the original dataset. In other words, the classifier did not observe such images during the training stage. Therefore, the behaviour of a classifier on these OOD inputs might not faithfully reflect what it has learned from the training dataset, and thus, the provided explanations might be meaningless.

Meaningful perturbation-based methods: These methods aim to mitigate the OOD problem in the classic perturbation-based methods. This is realised by replacing the features to be perturbed with plausible alternative values that are inferred from the distribution of the training datasets, as shown in Fig. 3. Therefore, in this paper, we call these methods meaningful perturbation-based methods, which means the perturbed inputs are meaningful to the classifier. In the literature, these methods are usually expressed through a marginalisation form [19,18]:

$$\begin{aligned} p(c|\boldsymbol{x}_{\backslash r}) &= \int p(c|\boldsymbol{x}_r^*, \boldsymbol{x}_{\backslash r}) p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\backslash r}) d\boldsymbol{x}_r^* \\ &= \mathbb{E}_{\boldsymbol{x}_r^* \sim p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\backslash r})}[p(c|\boldsymbol{x}_r^*, \boldsymbol{x}_{\backslash r})] \end{aligned} \tag{1}$$

where r denotes a subset of the input features, $\boldsymbol{x}_r$ denotes a partition of the input $\boldsymbol{x} = \boldsymbol{x}_r \cup \boldsymbol{x}_{\backslash r}$, and $p(c|\boldsymbol{x}_{\backslash r})$ denotes the classifier's output when features $\boldsymbol{x}_r$ are perturbed by replacing them with plausible alternative values. The $p(c|\boldsymbol{x}_{\backslash r})$ can be also interpreted as the output of the classifier when the $\boldsymbol{x}_r$ are unknown (or removed from the input) [18], since the contributions of $\boldsymbol{x}_r$ are marginalised out [19,18]. The difference between $p(c|\boldsymbol{x}_{\backslash r})$ and $p(c|\boldsymbol{x})$ measures the importance of $\boldsymbol{x}_r$. The $p(\boldsymbol{x}_r|\boldsymbol{x}_{\backslash r})$ in (1) describe the distribution of plausible values of $\boldsymbol{x}_r$ conditioned by $\boldsymbol{x}_{\backslash r}$, which guarantees that the perturbed inputs are aligned with the training datasets. However, the exact distribution $p(\boldsymbol{x}_r|\boldsymbol{x}_{\backslash r})$ is hard to achieve. Current promising approaches use generative models to approximate this distribution from the training dataset [19,18], from which within-distribution perturbed inputs can be generated. Meaningful perturbation-based methods have now been used to explain image models [18] and language processing models [19], but in the context of TSC, their performance has not been explored.

2.3. Optimisation-based saliency computation

Optimisation-based saliency computations also aim to identify the most important features, but they reformulate their goal as: *what is the smallest feature set of the current input that would maximally change the classifier's output?* These methods focus on identifying a group of features rather than assigning an importance score to each feature [21,18]. An example of this kind of explanation is shown in Fig. 2b. Compared with classic saliency methods, optimisation-based saliency computations have their advantages. For example, optimisation-based saliency explanations can help humans focus on the key features, which helps them to explore the reasons for an output, while in a classic saliency explanation, humans have no insights into how many features they should focus on to understand an output. Definitely, we can set a threshold for a classic saliency explanation to obtain a binary saliency map, where the features whose importance scores are above this threshold are considered important but otherwise are not important. A suitable threshold should ensure that the important features are identified and the unimportant features are excluded. However, it may be a challenge to set a suitable threshold. The suitable thresholds might differ for different problems or even depending on the instances belonging to the same problem.

$$\begin{aligned} \text{Binary Input} &: x_1, x_2, x_3 \\ \text{Binary Classifier} &: \mathrm{F}(x_1, x_2, x_3) = x_1 \wedge x_2 \wedge x_3 \\ \text{Current Input} &: \mathrm{F}(0, 0, 1) = 0 \end{aligned} \tag{2}$$

Another advantage of optimisation-based saliency explanations is that they avoid the difficulty of obtaining exact importance scores. This difficulty comes from various aspects; one of them is model saturation [28]. For example, (2) shows a binary classifier of which the three input features are also binary. The classifier produces "1" when all input features are "1", otherwise it produces "0". Herein, a classification task, $\mathrm{F}(0,0,1) = 0$, is going to be explained using classic perturbation-based saliency methods. If x_1 and x_2 are perturbed independently, which means that only one feature is perturbed each time, the classifier will maintain its prediction, since the classifier is saturated to produce "0" as long as one of the input features is "0". In this circumstance, both the importance of x_1 and x_2 are evaluated as zero. However, if x_1 and x_2 are perturbed together by replacing them with "1", the classifier will flip its prediction. This means that x_1 and x_2 indeed make their contributions to the current prediction, but their contributions can only be evaluated together. This means that to obtain the importance score of one feature we not only need to evaluate its independent contribution by perturbing this feature alone, but we also need to evaluate its cooperative contributions with other features by perturbing feature sets containing this feature. The evaluation of the exact importance needs to consider all subsets of the input features, which is an NP-hard problem [22–24]. Therefore, the evaluation of an exact importance score is hard to achieve for many real-world applications. In contrast, optimisation-based methods only focus on identifying a feature set that is effective at changing the output of the classifier, for example in (2), they only need to identify the feature set $\{x_1, x_2\}$ and do not need to waste time on estimating the importance score of these three features.

Although having advantages, optimisation-based saliency explanations face several challenges. The first comes from the difficulty in identifying the *smallest* feature set, which can only be identified through an exhaustive search. Current solutions mitigate this challenge through two kinds of approaches: optimising an auxiliary neural network selector [14,18] and using heuristic search methods [23,22,24]. The former approaches adopt gradient descent optimisation methods to train an auxiliary neural network to identify feature sets that can maximally change the classifier's output. A sparsity penalty is added to the loss function to obtain a small set. However, these approaches need the gradients of the classifiers, so they are not model-agnostic and cannot be applied to classifiers whose gradients are not accessible. The approaches in the other group are model-agnostic and can be applied to any classifier. Typical heuristic search methods adopted in saliency computation include Beam Search [23], Greedy method [24] and Genetic Algorithm [22]. Currently, these methods are adopted in explaining language processing models [24,22], while in the context of TSC, these methods have not been explored. The reason for this might be that MTS are usually high-dimensional, where these search methods face challenges in searching for a good solution. One of the motivations of this paper is to address this challenge.

Another challenge that optimisation-based saliency methods face is the OOD problem. The basis of optimisation-based methods it to judge whether a group of features can maximally change the classifier's output. Currently, this is achieved through perturbation-based methods by perturbing these features and evaluating the changes in the output. Therefore, similar to classic perturbation-based saliency methods, optimisation-based saliency computations also need to address the OOD problem. However, many current optimisation-based saliency methods do not consider the OOD problem [14,24,23].

2.4. Saliency-based explanation and feature selection

Saliency-based explanation may seem similar to the feature selection [25] and feature weighting problem [32], but it has different goals and faces different challenges. Feature selection is to resolve a binary optimisation problem to select the most representative features, while feature weighting is to solve a continuous optimisation problem so as to assign a weight to each feature. Both of these problems are addressed in the data preprocessing stage, aiming to improve the model's performance and enhance the model's interpretability. However, in this paper, the saliency-based explanation does not aim to improve the classifier's performance but rather to identify the relevant features for a given classification so as to explain a classifier which has been already trained. A saliency-based explanation is only meaningful for a particular classification because the classifier might focus on different features when making classifications for different inputs. So for each classification to be explained, a search operation needs to be carried out separately. In contrast, feature selection (or feature weighting) only needs to do a search one time, during the model construction stage. In terms of interpretability, feature selection (or feature weighting) might do better for global interpretability, since the number of input features is reduced globally (or the weights of the features are the same globally), while saliency-based explanation aims to provide local interpretability since the identified relevant features only help to explain the model's behaviour on a specific input.

In addition to the different goals, saliency-based explanation also faces different challenges. Both feature selection and saliency-based explanation need to evaluate the influence of certain features on the model. In feature selection, if we employ wrapper methods [33], this is realised by training the same model with different features and then checking the accuracy (or whatever metrics of interest we are maximising/minimising). If some features are important, removing them from the input will affect the model's performance significantly. However, in the context of saliency-based explanation, the model has been already trained before looking for how to explain it, and the number of input features is fixed, so we can not physically remove features like that in the feature selection. Therefore, how to evaluate the influence of certain features is a challenge that needs to be addressed in this paper.

Both feature selection and saliency-based explanation need to search over the input feature space to identify the smallest feature set that includes all important or relevant features. For time series, feature selection is commonly adopted at the series level [34], which means that each series is regarded as one feature. If one series is considered important, the whole series is maintained, but otherwise, it is removed. However, this is not what we wish to obtain in a saliency-based explanation. A classifier might only focus on a limited number of the features of a series to make a prediction. Therefore, we hope to search for the really necessary features rather than the whole series. Of course, we can consider each feature of a time series individually and carry out search operations to identify the necessary features. However, in this case, the search space would increase exponentially with the number of features, which makes the problem hard to be addressed efficiently in the context of high-dimensional data [25,35,36]. Recently, in feature selection for tabular data, Cooperative Coevolutionary Algorithms (CCs) have been used to address this challenge [36,35,25]. The input features are first grouped into a number of groups. The features belonging to the same group have strong correlations. Then, the same number of Evolutionary Algorithms (EAs) are carried out over each group and evolve cooperatively to search for the most representative features. Through such a "divide and conquer" strategy, the original complex high-dimensional optimisation problem is divided into a number of simple low-dimensional optimisation problems, which can be addressed more easily. However, such a strategy is not suitable to be directly adopted for the TSC problem. For tabular data, each feature might have a clear meaning for humans, while for time series, a feature at a specific time step might not be so meaningful. Humans might understand what a time series means better by the features within the continuous time steps, for example, an up or down trend. Therefore, to provide better interpretability of a TSC, a saliency-based explanation needs to consider how to search for the most important features within continuous time steps rather than the isolated features.

3. Methodology

In this section, the problem that we are going to address is formulated in Section 3.1. Then, an overview of our proposed framework is provided in Section 3.2. After that, the modules, including the design of the generative model (Section 3.3) and the design of the search method (Section 3.4), which compose this framework, are described in detail.

3.1. Problem definition

Let $p(c|\boldsymbol{x})$ be a TSC that produces the class label of $\boldsymbol{x}$. Assume that we have a particular input $\boldsymbol{x}^*$ for which the classifier's output is $p(c|\boldsymbol{x}^*) = c_1$. Our main goal is to explain, using a saliency-based explanation, why the classifier predicts that $\boldsymbol{x}^*$ belongs to class c_1 rather than another class. In the final explanation, the key features of $\boldsymbol{x}^*$ that are necessary for the classifier to produce this output are identified. If these features had not taken their actual values but other plausible values instead, the classifier would have produced a different output. Therefore, in this paper, these key features are also called 'supporting features' and denoted by $\boldsymbol{x}^*_s$. The set $\boldsymbol{x}^*$, which includes all input features, is definitely supporting. However, a smaller set would provide more insight into explaining the output. Therefore, in order to provide better explanations, we are trying to find the smallest supporting feature set to help users who use this classifier understands why $p(c|\boldsymbol{x}^*) = c_1$.

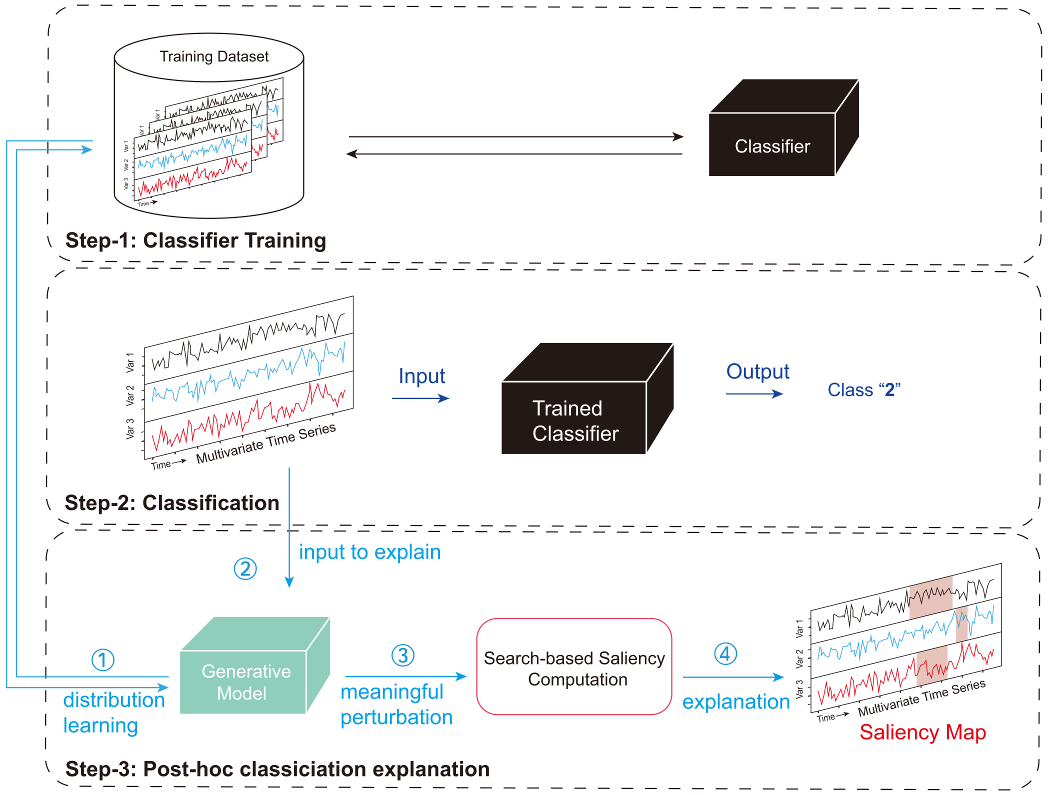


Fig. 4. Explaining a classifier through a post-hoc approach. In the first step, the classifier is trained on an MTS training dataset. Then, the trained classifier is used to classify an instance of the MTS, of which the output is explained in the third step.

3.2. *Pipeline of the proposed framework*

Fig. 4 shows the pipeline of the proposed framework that explains TSC through a post-hoc approach. Algorithm 1 provides the corresponding pseudo-code for this framework. Since we are taking a post-hoc approach, a classifier p is first trained on the dataset before we try to explain. Then, an output is produced for a given instance, $\boldsymbol{x}$. To explain this output, a generative model G is trained to learn the distribution of the dataset on which the classifier is trained (line 1 in Algorithm 1) and then used to generate perturbed inputs for the input to be explained (line 5 in Algorithm 1). Since the training dataset of the classifier is fixed, the generative model only needs to be trained one time and can be used to explain any input. The structure of the generative model and the training strategy are described in Section 3.3. After that, a greedy-based segmentation and identification search method is designed to identify the smallest supporting feature set of this input. This method is composed of segmentation (line 2 and line 10 in Algorithm 1) and identification (lines 4 to 6 in Algorithm 1) steps. A detailed description of these two steps is provided in Section 3.4. The connection between the proposed generative model and the search method is that the generative model is primarily used to create realistic inputs for evaluating importance of specific features, while the greedy segmentation strategy is employed to identify the most important feature set for a given output. During the greedy search process, the generative model is used to generate realistic inputs. Finally, an explanation is generated for this output through a saliency map, where a small supporting feature set is identified.

Algorithm 1: The pseudo-code for the proposed framework .

```
   Input : Time Series $\boldsymbol{x}$, Classifier $p$, Training Dataset $D$
1  $G$ ← Training the generative model;
2  Segments ← Segmentation($\boldsymbol{x}$);
3  while True do
4      $\boldsymbol{W}_0$ ← Initialising saliency maps by Segments;
5      $\boldsymbol{W}_{best}$ ← Searching for best saliency map by $(\boldsymbol{x}, G, \boldsymbol{W}_0, p)$ ;
6      Identified Segments ← $\boldsymbol{W}_{best}$ ;
7      if the length of the Identified Segment = 1 then
8          break;
9      end
10     Segments ← Segmentation (Identified Segments) ;
11 end
   Return: $\boldsymbol{W}_{best}$
```

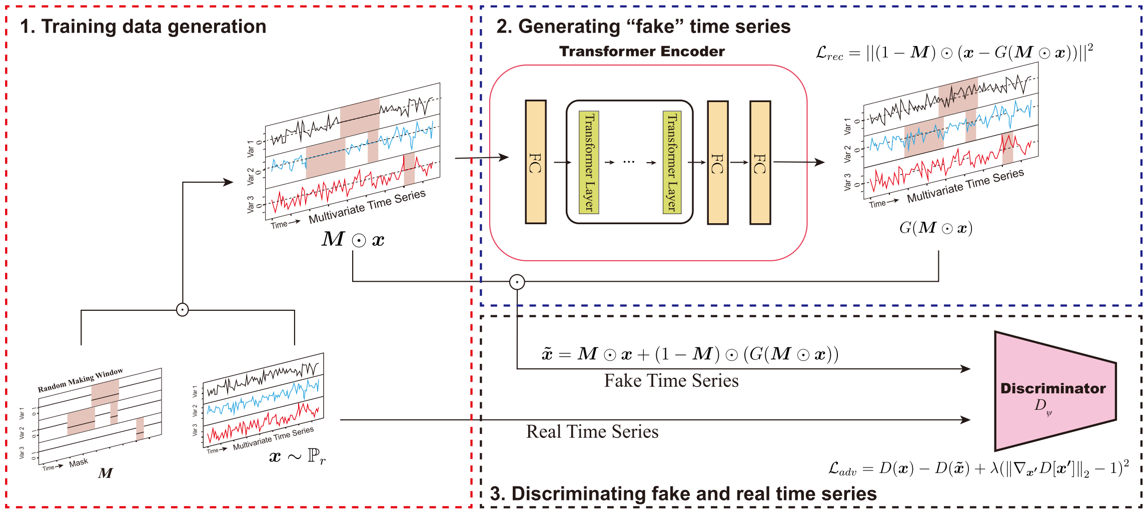

1. Training data generation
2. Generating “fake” time series
Transformer Encoder
FC
Transformer Layer
Transformer Layer
FC
FC
$\mathcal{L}_{rec} = ||(1-M)\odot(x - G(M\odot x))||^2$
Multivariate Time Series
$M \odot x$
$G(M\odot x)$
Random Making Window
Mask
M
$x \sim \mathbb{P}_r$
$\tilde{x} = M\odot x + (1-M)\odot(G(M\odot x))$
Fake Time Series
Real Time Series
Discriminator
D_ψ
$\mathcal{L}_{adv} = D(x) - D(\tilde{x}) + \lambda(\|\nabla_{x'}D[x']\|_2 - 1)^2$
3. Discriminating fake and real time series

Fig. 5. The structure of the proposed generative model. The training data are generated by masking features within some randomly chosen time intervals. Then, the generative model is trained to generate plausible alternative values for these masked features. To enhance the quality of the MTS generated by the generator, a discriminator is added to distinguish the fake MTS (generated by the generative model) and the real MTS (sampled from the dataset).

3.3. Designing the generative model

The basis of this framework is to determine whether a group of features are supporting features, something which is accomplished by (1). If $p(c|\boldsymbol{x}^*_{\backslash r}) \neq p(c|\boldsymbol{x}^*)$, $\boldsymbol{x}^*_r$ are supporting features. Here, a generative model is designed for MTS to estimate the necessary distribution, $p(\boldsymbol{x}^*_r|\boldsymbol{x}^*_{\backslash r})$. The structure of the proposed generative model is shown in Fig. 5. In the following part of this section, first, the method to generate the training data for this generative model is described. Then, we give a detailed description of the structure of the generative model. Finally, the objectives to be optimised and the training strategy are given.

3.3.1. Training data generation

To train the generative model to estimate the distribution, $p(\boldsymbol{x}^*_r|\boldsymbol{x}^*_{\backslash r})$, the first step is to obtain its inputs, $\boldsymbol{x}^*_{\backslash r}$, and outputs, $\boldsymbol{x}^*_r$. We sample MTS data from the dataset, and for each MTS, a binary mask is created to mask out certain features through element-wise multiplication, $\boldsymbol{x}^* \odot \boldsymbol{M}$, creating an input for the generative model. The complementary components, $\boldsymbol{x}^* \odot (\mathbf{1}-\boldsymbol{M})$, are the target the generative model tries to generate.

3.3.2. Architecture of the generative model

The aim of the generative model is to create plausible values for $\boldsymbol{x}^*_r$ conditioned by $\boldsymbol{x}^*_{\backslash r}$. This is very similar to time series imputation problems, where the missing components of a time series need to be filled in. However, in the missing values imputation problem, they do not have the actual ground truth values. Therefore, imputation models attempt to learn the temporal dependence of the time series using the observed values. The missing parts are then filled in based on the learned temporal dependence. In our scenario, the time series we have are complete. We have the ground truth for each feature, so we do not need to predict them. Our goal is to generate alternative plausible values for certain features, which can maintain the perturbed time series within the original distribution. Nevertheless, although time series missing imputation models have a different goal, they can be applied in this situation.

Prior to this work, we compared the performance of various state-of-the-art methods for imputing missing values [37] with that of two generative methods we designed. One of the methods is based on Transformers [38], and the other is based on Bidirectional RNNs [39]. The results of this comparison are provided in the supplementary material.[2] In those results we observe that the Transformer is the best performing method, and that is why we have adopted it in this paper. Specifically, the original time series is first embedded by a fully connected layer and then fed into a Transformer encoder, which is composed of several Transformer layers. Then, the output of the Transformer encoder is fed into a fully connected layer followed by a hyperbolic tangent (tanh) activation function, which aims to constrain the predicted values into [-1, 1].

However, it is important to note that the results presented in the supplementary material indicate that although the imputation ability of the Transformers is superior to the other methods, that does not significantly influence the result of the final explanations (i.e. the results of the deterioration test, see Table 3).

[2] https://github.com/menghan1994/ETSC_through_Meainingful_Perturbation_and_Optimisation.

3.3.3. Optimisation objective

The generative model aims to produce the masked component. Therefore, one of the loss functions is a reconstruction loss: $\mathcal{L}_{rec} = ||(1 - \boldsymbol{M}) \odot (\boldsymbol{x} - G(\boldsymbol{M} \odot \boldsymbol{x}))||^2$, where G denotes the whole generative model. This loss measures the distance between the target features of the original inputs and those generated by the generative model. In addition, to enhance the quality of the MTS generated by the model, we adopt the adversarial training strategy. Specifically, we take the Wassersten Generative Adversarial Networks (WGANs) [40] diagram to train the generative model. WGANs are a modification of the classic GANs [41] whose discriminator has no activation function in its final layer. WGANs are better at learning stability and avoiding model collapse than are the classic GANs [42]. In this paper, we use the gradient penalty proposed in [40] to implement the Lipschitz constraint. The adversarial loss is

$$\mathcal{L}_{adv} = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] + \lambda \mathop{\mathbb{E}}_{\boldsymbol{x'}\sim\mathbb{P}_{x'}}[(\|\nabla_{\boldsymbol{x'}} D[\boldsymbol{x'}]\|_2 - 1)^2] \tag{3}$$

where D is the discriminator; P_r is the training dataset distribution; P_g is the fake dataset distribution from which the MTS are generated by $\tilde{\boldsymbol{x}} = \boldsymbol{M} \odot \boldsymbol{x} + (1 - \boldsymbol{M}) \odot (G(\boldsymbol{M} \odot \boldsymbol{x}))$, being the combination of the generated masked features and the non-masked parts of the real data; $P_{x'}$ is the distribution sampled uniformly along straight lines between pairs of points sampled from P_r and P_g [40]; λ is a parameter that controls the strength of the gradient penalty.

3.3.4. Training strategy

Following the training diagram of GANs [41], the discriminator D and the generator G are optimised iteratively through stochastic gradient descent. The discriminator is optimised by minimising the adversarial loss $\mathcal{L}_{adv}$ in (3) and the generator is optimised by minimising the overall generative loss which is expressed by

$$\mathcal{L}_{rec} = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[\mathcal{L}_{rec} + \lambda_{dis} D(\tilde{\boldsymbol{x}})] \tag{4}$$

where λ_{dis} is a hyperparameter controlling the strength of the penalty of the adversarial loss.

3.4. Search-based method for saliency computation

The previous steps have provided the components necessary to judge whether a group of features are supporting features for the given classification. The next step of our proposed framework is to find the smallest supporting feature set. However, the smallest supporting feature set can only be obtained by an exhaustive search, which is impossible when the feature space is huge. Therefore, we propose to use heuristic search-based methods to find approximate solutions within a reasonable time. This section first introduces the objective (or fitness) function guiding the search. Then, we design a method to address the difficulties faced by classic search methods when searching a huge feature space.

3.4.1. Fitness function for search-based saliency explanation

In this section, we are going to describe how to evaluate whether a saliency explanation is ideal or not and then formulate these evaluation metrics as a fitness function. A smaller value of the fitness means a better saliency explanation.

First, the identified features in an ideal saliency explanation should be supporting features, which means that if these features were replaced by other plausible values, the classifier would produce a different output. This is formulated as follows:

$$\text{Sup}(\boldsymbol{W}) = \begin{cases} 1 & \text{if } p(c|\boldsymbol{x}) = p(c|\boldsymbol{x}_{\backslash r}) \\ 0 & \text{if } p(c|\boldsymbol{x}) \neq p(c|\boldsymbol{x}_{\backslash r}) \end{cases} \tag{5}$$
$$\text{where } \boldsymbol{x}_{i,j} \in \boldsymbol{x}_{\backslash r}, \text{ if } \boldsymbol{W}_{i,j} = 1$$

where the saliency map is represented by a binary mask $\boldsymbol{W} \in \mathbb{R}^{D\times T}$; $\boldsymbol{W}_{d,t} = 1$ means the feature $\boldsymbol{x}_{d,t}$ is identified; $\text{Sup}(\boldsymbol{W}) = 0$ means the identified features are supporting features. To calculate $p(c|\boldsymbol{x}_{\backslash r})$ using (1), it is necessary to sample from the distribution $p(\boldsymbol{x}_r|\boldsymbol{x}_{\backslash r})$. [43] has shown that keeping the dropout layer active during the inference stage and running the forward process multiple times is equivalent to sampling from the learned distribution. Therefore, (1) can be reformulated as

$$\begin{aligned} p(c|\boldsymbol{x}_{\backslash r}) &= \int p(c|\boldsymbol{x}_r^*, \boldsymbol{x}_{\backslash r}) p(\boldsymbol{x}_r^*|\boldsymbol{x}_{\backslash r}) d\boldsymbol{x}_r^* \\ &= \int p(c|g_{\theta,\hat{\epsilon}}(\boldsymbol{x}_{\backslash r}), \boldsymbol{x}_{\backslash r}) d\epsilon \\ &= \mathbb{E}_{\hat{\epsilon}\sim p(\epsilon)} p(c|G_{\theta,\hat{\epsilon}}(\boldsymbol{x}_{\backslash r}), \boldsymbol{x}_{\backslash r}) \end{aligned} \tag{6}$$

where G represents the generative model; $p(\epsilon)$ is a product of Bernoulli distribution with probabilities of $1 - p$, and p represents the dropout rate in the training stage; the $\boldsymbol{\theta}$ represent the weights of the final fitted generative model.

Second, sparsity is also a desired property of an ideal explanation, which means that the size of the identified supporting feature set should be small. This helps in providing clear explanations. This objective is expressed by

$$\text{Spar}(\boldsymbol{W}) = \|\boldsymbol{W}\|_1 \tag{7}$$

where $\|\cdot\|_1$ denotes the L1-norm of the binary mask $\boldsymbol{W}$. A smaller value of $\text{Spar}(\boldsymbol{W})$ means fewer features are identified.

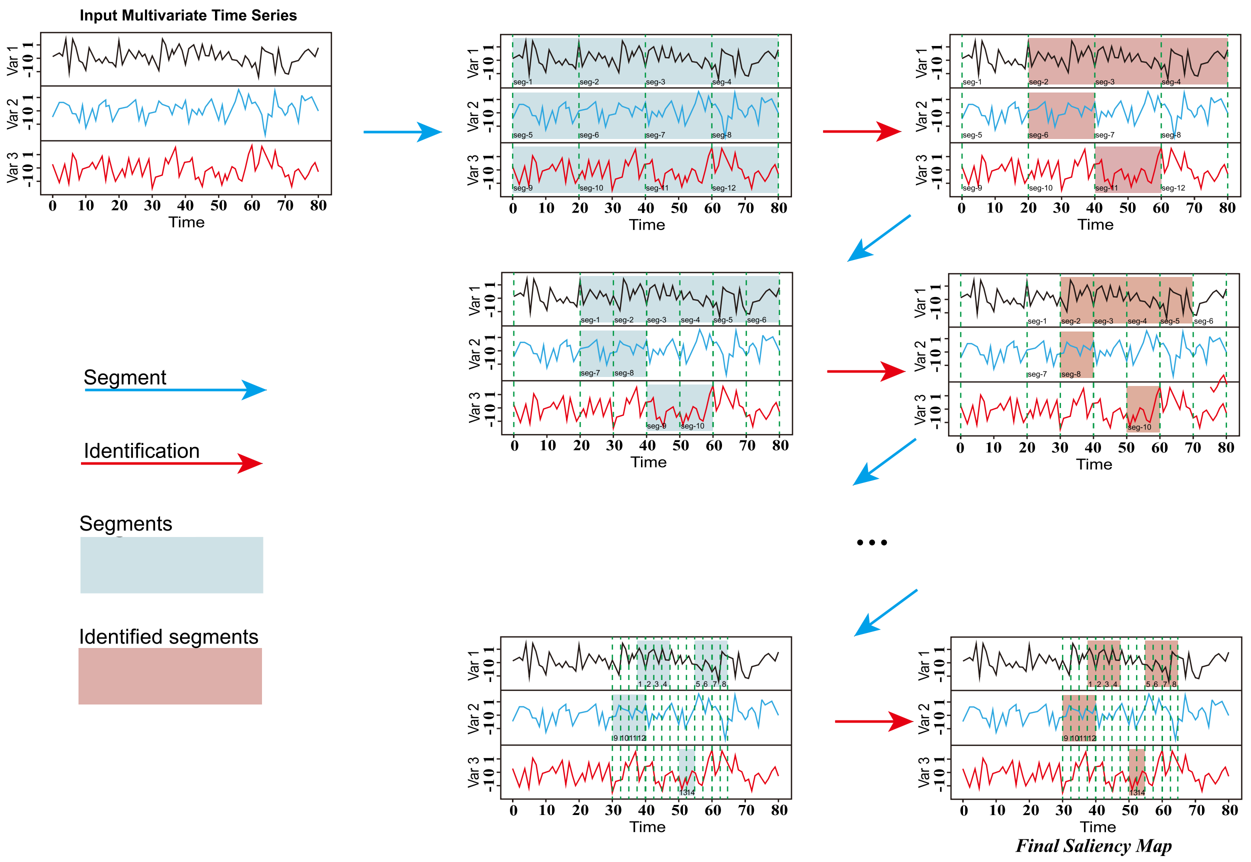


Fig. 6. Proposed greedy-based segmentation and identification search method. The input MTS is segmented into several segments. In the following step, the smallest supporting segments are identified through a binary search method. In the next segment step, the previously identified segments are further segmented into smaller segments. Then, the adopted binary search model is implemented on these smaller segments. This process continues until the length of the segments is one.

Finally, similar to [14], we assume that in the context of MTS, features within continuous time steps provide more insights for humans to understand the behaviour of a classifier than those scattered over discrete time steps. This is similar to the pixels in images, where the pixels within a continuous region make more sense for humans than those scattered everywhere. Therefore, in an ideal saliency explanation, we hope the identified features can be continuous in time, which can be realised by minimising the following objective:

$$\text{Conti}(\boldsymbol{W}) = \frac{1}{DT} \sum_{d=0}^{D} \sum_{t=0}^{T-1} |\boldsymbol{W}_{d,t} - \boldsymbol{W}_{d,t+1}| \tag{8}$$

The final fitness function for the task at hand is the combination of the above three objectives:

$$\text{Fitness}(\boldsymbol{W}) = \lambda_e \text{Sup}(\boldsymbol{W}) + \lambda_s \text{Spar}(\boldsymbol{W}) + \lambda_c \text{Conti}(\boldsymbol{W}) \tag{9}$$

where λ_e, λ_s, and λ_c are three parameters that control the priority of these objectives in the process of searching for an ideal explanation. Finding supporting features has the highest priority, therefore, the fitness value of a saliency map, of which the identified features are supporting features, should always be smaller than that of which the identified features are not supporting features. This priority is realised by setting $\lambda_e > \lambda_s \text{Spar}(\boldsymbol{W}) + \lambda_c \text{Conti}(\boldsymbol{W})$ for any $\boldsymbol{W}$. Similarly, the sparsity requirement has the second priority, which is maintained by setting $\lambda_s \text{Spar}(\boldsymbol{W}) > \lambda_c \text{Conti}(\boldsymbol{W})$ for any solution. Because $\text{Sup}(\boldsymbol{W}) \subset [0, DT]$ and $\text{Conti}(\boldsymbol{W}) \subset [0, 1]$, in this paper, we set $\lambda_e = 10DT$, $\lambda_s = 1.0$ and $\lambda_c = 0.1$ to maintain the defined priority.

3.4.2. *Greedy-based segmentation and identification search method*

After defining the fitness function, the challenge we are facing now is how to obtain an ideal explanation in a reasonable time. MTS are often high-dimensional, on which classic search methods face difficulties in obtaining ideal solutions. Here, we mitigate this difficulty by the proposed greedy-based segmentation and identification search method, of which an overview is shown in Fig. 6. The overall search operation is composed of a number of sequential sub-steps. Each sub-step contains two operations: segmentation and identification. At the beginning, the original input MTS is segmented into a number of non-overlap segments. Each segment contains the features within a certain time interval and is considered as one “super-feature” during the next identification step. In

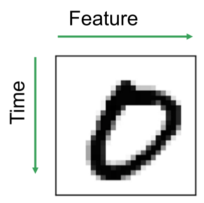


Fig. 7. Treating MNIST images as MTS. Each image is regarded as an MTS with 28 time steps, each of which has 28 features.

the identification steps, a binary heuristic search method is used to search for the smallest group of supporting segments over the feature space composed of these “super-features”. Then in the next segment step, the identified supporting “super-features” in the previous identification step are further segmented into smaller segments, on which the binary search method is carried out again. The two operations are carried out iteratively until the “super-features” can not be segmented further (the length of the identified segments is one).

In the identification steps, we are dealing with a binary optimisation problem. In the literature, many classic heuristic methods can be adopted, such as Genetic Algorithm (GA) [44] and Binary Particle Swarm Optimisation (BPSO) [45]. Both of these classic methods have their advantages as well as limitations. In this work, we implement GA and BPSO in the identification steps. We will show in the experiment that these two methods achieve very similar performance. One thing that has to be noted here is that, although we take GA and BPSO, users can use any binary search method in the identification steps. We believe a suitable search method will help in achieving better results. Besides, although we focus on the TSC context, the proposed greedy-based segmentation and identification search strategy is more broadly applicable to any domain, such as images.

4. Experimental design

In this section, we are going to provide details of the experimental design. First, we describe the adopted TSC datasets and the classifier we are going to explain. Then, the experimental parameters, including the hyperparameters of the generative model and the settings of the adopted search methods, are provided. Finally, the benchmarking explaining methods that are going to be compared with are presented, and the quantitative evaluation metric is described.

4.1. Datasets

To validate the performance of our proposed framework, we adopt the UEA TSC archive, which is a well-known benchmarking archive containing 30 TSC datasets collected from a wide range of applications [46]. In addition, we also use the handwriting images dataset, MNIST [47] in our experiments. Although it was not originally collected as an MTS, we can treat it as an MTS, as did [15]. We take MNIST here for visualisation purposes, which helps to obtain some useful qualitative evaluations. The details of the pre-processing methods are as follows.

UEA TSC Archive: Detailed information of these datasets can be obtained from [46] and a summary of these 30 datasets is given in Table 1. As can be seen in the table, the time series in some datasets have too many features or time steps, e.g. DuckDuckGeese, EigenWorms, MotorImagery, and PEMS-SF. Providing explanations for these problems is very time consuming. In this paper, we only consider the 26 medium-size datasets. The datasets, including CharacterTrajectories (30%), InsectWingbeat (57%), JapaneseVowels (45%), and SpokenArabicDigits (57%), have missing values, of which the percentage of the missing features are given in brackets. We will first fill in the missing values in these datasets with zeros. In the following section, we will show that the classifier achieves better performance than random guessing on these datasets. Since the main aim of this work is to explain the classifier, filling missing values with zeros is acceptable. Second, the features of every dimension are normalised using the min–max normalization method and then rescaled into $[-1, 1]$. The rescaling operation aims to keep the range of the real-world MTS be consistent with that generated by the generative model, of which the activation function in the final layer is tanh.

Treating MNIST as an MTS: The original 28 × 28 images are treated as an MTS with 28 time steps, each of which has 28 features. Here, the y-axis is treated as the time steps and the x-axis as the features, which is shown in Fig. 7. Similarly, the images are normalised using the min–max normalization method and then rescaled into [-1, 1].

4.2. Classifier

The novelty of this paper is the model-agnostic framework that explains any classifier through a post-hoc approach. Designing a high-performance TSC is not our goal. Therefore, we only implement a simple classifier based on RNNs for our experiments, where Long Short Term Memory (LSTM) cells [48] are used to encode the whole MTS and the hidden states at the final step are fed into a fully-connected layer followed by a soft-max layer to produce the probability of each class. The class with the highest probability is regarded as the predicted class. In our experiments, the dimension of the hidden states of this classifier is set to 128. The accuracy of the classifier on MNIST is 98%, while the accuracy of the classifier varies among the datasets in the UEA archive. As shown in Table 2, the classifier performs excellently on some problems, such as JapaneseVowels, SpokenArabicDigits, and PenDigits. However, on other problems, such as AtrialFibrillation, EthanolConcentration and FingerMovements, its performance is not better than random guesses,

Table 1
A summary of the 30 datasets in the UEA TSC Archive.

	Dataset	Training cases	Test cases	Dimensions	Length	Classes
0	ArticularyWordRecognition	275	300	9	144	25
1	AtrialFibrillation	15	15	2	640	3
2	BasicMotions	40	40	6	100	4
3	CharacterTrajectories	1422	1436	3	182	20
4	Cricket	108	72	6	1197	12
5	**DuckDuckGeese**[*]	50	50	1345	270	5
6	**EigenWorms**[*]	128	131	6	17984	5
7	Epilepsy	137	138	3	206	4
8	EthanolConcentration	261	263	3	1751	4
9	ERing	30	270	4	65	6
10	FaceDetection	5890	3524	144	62	2
11	FingerMovements	316	100	28	50	2
12	HandMovementDirection	160	74	10	400	4
13	Handwriting	150	850	3	152	26
14	Heartbeat	204	205	61	405	2
15	InsectWingbeat	30000	25000	200	22	10
16	JapaneseVowels	270	370	12	29	9
17	Libras	180	180	2	45	15
18	LSST	2459	2466	6	36	14
19	**MotorImagery**[*]	278	100	64	3000	2
20	NATOPS	180	180	24	51	6
21	PenDigits	7494	3498	2	8	10
22	**PEMS-SF**[*]	267	173	963	144	7
23	PhonemeSpectra	3315	3353	11	217	39
24	RacketSports	151	152	6	30	4
25	SelfRegulationSCP1	268	293	6	896	2
26	SelfRegulationSCP2	200	180	7	1152	2
27	SpokenArabicDigits	6599	2199	13	93	10
28	StandWalkJump	12	15	4	2500	3
29	UWaveGestureLibrary	120	320	3	315	8

[*] These 4 datasets are not considered in this paper. Providing explanations for these problems would be very time-consuming.

Table 2
The accuracy of the classifier and random guessing on the problems in UEA archive. The RNNs-based classifier performs well on the 17 datasets shown in bold.

	Dataset	Accuracy			**Dataset**	Accuracy	
		RNNs	Guess			RNNs	Guess
0	**ArticularyWordRecognition**	**0.89**	0.04	15	**InsectWingbeat**	**0.29**	0.10
1	AtrialFibrillation	0.33	0.33	16	**JapaneseVowels**	**0.94**	0.11
2	**BasicMotions**	**0.73**	0.25	17	**Libras**	**0.71**	0.07
3	**CharacterTrajectories**	**0.67**	0.05	18	**LSST**	**0.62**	0.07
4	**Cricket**	**0.79**	0.08	20	**NATOPS**	**0.86**	0.17
7	**Epilepsy**	**0.54**	0.25	21	**PenDigits**	**0.99**	0.10
8	EthanolConcentration	0.27	0.25	23	**PhonemeSpectra**	**0.14**	0.03
9	**ERing**	**0.79**	0.17	24	**RacketSports**	**0.76**	0.25
10	FaceDetection	0.59	0.50	25	SelfRegulationSCP1	0.53	0.50
11	FingerMovements	0.52	0.50	26	SelfRegulationSCP2	0.69	0.50
12	HandMovementDirection	0.34	0.25	27	**SpokenArabicDigits**	**0.98**	0.10
13	Handwriting	0.03	0.04	28	**StandWalkJump**	**0.47**	0.33
14	Heartbeat	0.70	0.50	29	**UWaveGestureLibrary**	**0.54**	0.13

where we assume that the classifier also performs a random guessing classification. This means that the classifier's outputs are not dependent on the inputs. Under this circumstance, explaining the outputs through saliency maps does not make sense. Therefore, we only take the datasets on which the classifier's performance is notably better than random guesses in our experiments, which are shown in bold in Table 2. Among them, 17 datasets shown in bold meet this requirement. The inputs in the test datasets are used in our explaining experiments. To save experiment time, for the problems, InsectWingbeat, PhonemeSpectra, and SpokenArabicDigits, we randomly sample 1000 inputs from the test datasets to do experiments while for other datasets all the inputs in the test dataset are used in our experiments. Finally, 17 datasets are used in our experiment. In the supplementary materials, we also provide the results for a convolutional-based classifier.

4.3. Experimental parameters

This section gives the parameters adopted for our experiments. In our experiments, we separately adopted the BPSO and GA methods for the identification steps in our proposed framework. The BPSO method is implemented using the package provided by

[45] and the GA method is implemented by the package provided by Ryan Solgi.[3] In our experiments, we tuned the parameters for these two methods and the satisfying settings are shown as follows. For the BPSO method, the adopted parameters are

- Population size: 100 (the number of particles in the swarm)
- Inertia weight (w): 1 (the influence of a particle's previous velocity on its current velocity)
- Cognitive coefficient (c1): 5 (the influence of a particle's personal best position on its velocity update)
- Social coefficient (c2): 5 (the influence of the swarm's global best position on the velocity update of a particle)

For the GA method, the adopted parameters are

- Population size: 100 (the number of candidate in the population)
- Mutation rate: 0.2 (the probability of altering one or more genes in an individual's chromosome)
- Crossover rate: 0.5 (the probability of exchanging genetic material between two parent individuals during reproduction)
- Parents portion: 0.5 (the proportion of individuals in the population that are selected as parents for performing crossover and mutation operations to generate offspring)

For these two methods, at the very beginning, the solutions are initialised randomly. Then, in the following optimisation sub-steps, the initial solutions are set to be all-ones solutions. During the sub-steps, both of these two methods stop when the values of the fitness function do not improve over 20 successive iterations.

4.4. Comparison methods

The saliency methods we are going to compare with are *Gradient-based methods*, *LIME* and *LIME-G*. These methods are widely adopted to explain classifiers and taken as benchmarking methods in many works [15,14].

1. *Gradient-based Methods*: The first gradient-based method we consider is SG, where the saliency scores are obtained by directly calculating the gradient of the probability of the predicted class w.r.t. the input features [26]. In addition, we also consider the IG method, which numerically integrates the gradients along a path between the current input and a user-defined baseline [27]. Following the setting of [27], we set the baseline in our experiments to zero input.

2. *LIME*: LIME [11] provides importance scores by learning a transparent proxy model in the local region near the input. It first perturbs the input data and creates a series of artificial data. We realise it by randomly selecting the features of the input MTS and replacing them with zeros. Then, the classifier produces outputs for these artificial data. After that, a linear regression model is fitted, of which the input features are binary vectors indicating the "presence" (by 1) or "absence" (by 0) of the corresponding original features and the outputs are the probability score of the predicted classes. Finally, the absolute values of the weights of the fitted linear model are regarded as the importance scores of the corresponding features.

3. *LIME-G*: The first step of LIME is to create a series of artificial data by perturbing the original input. This is realised through zeros replacement, which might create OOD inputs. Similar to [49], we also implement LIME-G, where G indicates that the artificial dataset is created with the help of the designed generative model. Specifically, to create one perturbed input, we randomly select a number of original features to be perturbed and then generate possible alternative values for them using the generative model. Then, the original features are replaced by these alternative values to create a perturbed input. The other steps in obtaining a saliency map are the same as the LIME method.

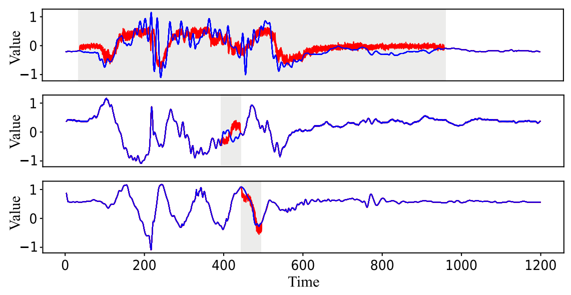
4.5. Quantitative evaluation metric

The "best" saliency maps should compactly identify supporting features for the outputs, which means that the classifier could produce a different output by perturbing a minimal number of features. Here, we use the quantitative metric adopted in [21,18] to evaluate the performance of a saliency explanation. Specifically, for a given saliency explanation, we calculate the minimal number of features that are necessary to be perturbed to change the classifier's output. The final saliency maps of our method are binary. Therefore, we can directly obtain this metric by calculating how many features are identified. However, classic saliency-based methods assign each feature a saliency score. To make these saliency maps comparable with our method, we successively select features in order of their saliency scores and then replace them with plausible values generated by the generative model until the classifier produces a different output. Whether the classifier would produce a different output is estimated through (1), where $\boldsymbol{x}_r$ represents the selected supporting feature. Finally, the selected features form binary saliency maps. Then, the number of the selected features is regarded as this metric. In this paper, we call this metric Smallest Deletion Features (SDF).

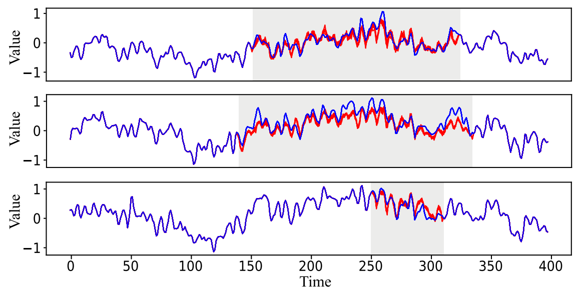
5. Results and discussion

In this section, we are going to systematically evaluate the performance of our proposed framework. First, we quantitatively and qualitatively evaluate the performance of the generative model in mitigating the OOD problem in Section 5.1, which is the basis

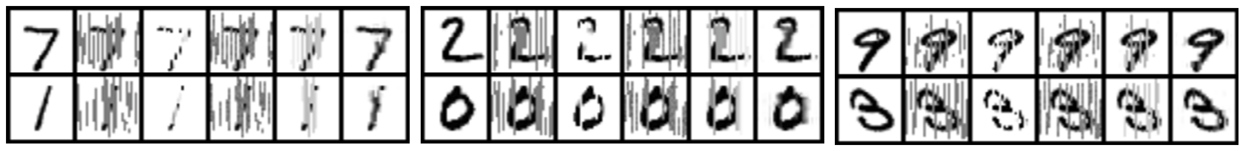
[3] https://github.com/rmsolgi/geneticalgorithm.



(a) Cricket



(b) HandMovementDirection



(c) MNIST

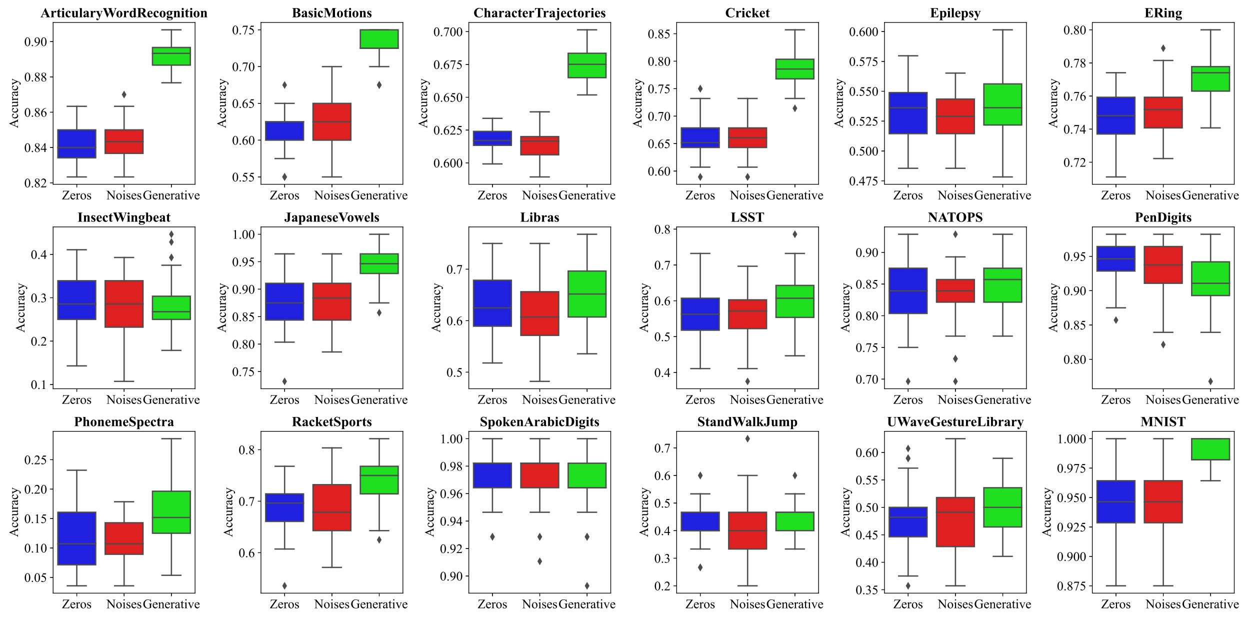
Fig. 8. The performance of the generative model in creating plausible values for certain features. In (a) and (b), the blue time series are the original time series and the red subsequences are generated by the generative model. In (c), the first column and second column are the original input and the input with some features randomly selected and masked. The third column to last column is perturbed input created by the weighted average of neighbouring features [14], by random noise, by the average values of corresponding dimensions and by the values created by the generative model, respectively.

for providing meaningful explanations. Further, the generative model will also be used in Section 5.2 to quantitatively evaluate the explaining results provided by adopted methods. If the OOD problem is not addressed, in other words, if the generative model does not approximate the distribution of the training dataset well, the evaluation methods will also be unfair. However, with only saliency-based explanations, users sometimes might still not understand the working mechanism of the classifier. Fortunately, our framework can provide counterfactual explanations, which will be described in Section 5.3. There are some stochastic processes in the proposed framework, but the explanations provided of a given classification should not change when we repeat the explaining process. Therefore, in Section 5.4, we carry out a stability analysis of the proposed framework. Finally, in Section 5.5, ablation studies are carried out to analyse the contributions of the generative model and the proposed search strategy separately.

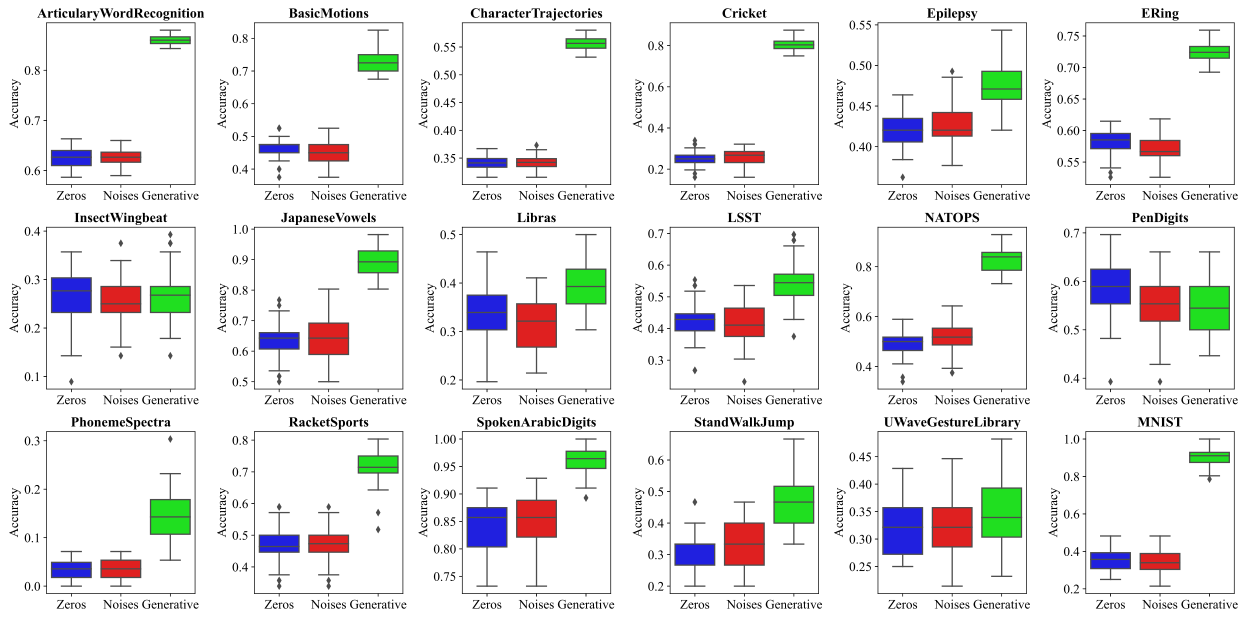
5.1. The performance of the generative model

We first qualitatively evaluate the performance of the generative model by visualising the perturbed inputs for the MNIST hand writing data, which is shown in Fig. 8. In Fig. 8a and Fig. 8b, the blue time series are the original time series and the red subsequences are generated by the generative model. The results show that the created values are closed to the target. In Fig. 8c, we also compare the performance of the generative model with an artificial method on the instance in the MNIST dataset. As is shown in the figure, some features are selected randomly and then perturbed through artificial methods and our generative model. For MNIST inputs, the within-distribution inputs should look similar to digits written on the white background. We can easily judge that the inputs created by the artificial methods do not meet this requirement. For example, noise is scattered over the background that should be white, or the original sharp features become blurry or even missing. In contrast, the generative model creates suitable values to replace the selected features, creating much more realistic inputs, which suggests that it performs better in creating within-distribution perturbed inputs than do artificial methods.

In addition to the above qualitative evaluation, we can also quantitatively evaluate the performance of the generative model with artificial methods. Inspired by [17], we cast this problem as the measurement of the model's robustness to perturbation, which is carried out as follows. For each input in the test dataset, a certain percentage of features is randomly selected and then perturbed using zeros and random noise, as is common practice in the literature, and using the alternative values created by the generative model. The percentages of features to be perturbed here are {20%, 50%}. Then, the accuracy of the classifier on these perturbed inputs is measured, where the perturbed inputs are assumed to have the same labels as the original inputs. In this scenario, we expect the performance of all classifiers to deteriorate in line with increases in the percentage of perturbed features. However, we also expect that features replaced with random or arbitrary (e.g. zeros) replacement (which will tend to be OOD) will result in overall worse performance compared to features replaced with within-distribution replacements –by the generative model proposed. The experiment is repeated 50 times for each percentage of features to perturb, and the accuracy is shown through boxplots in Fig. 9. Overall, the accuracy of the classifiers is higher on the perturbed inputs created by the generative model than on those perturbed artificially, which suggests that the classifiers are more robust to the perturbed inputs created by the designed generative model. This means that when some features are masked, the generative model could generate suitable alternative values, on which the classifier might be able to make correct predictions, like the inputs created by the generative model in Fig. 8c. If these masked features are filled with artificial values, creating input that the classifier did not observe during the training stage, like the artificial perturbed inputs in Fig. 8c, the classifier's behaviour might be unpredictable and have a higher probability of making wrong predictions. However, the



(a) 20% values are masked out



(b) 50% values are masked out

Fig. 9. Model's prediction accuracy on the test datasets, when a certain number of features are perturbed using "zeros", "random noise", and alternative values generated by the generative model. Better performance in creating within-distribution inputs yields higher model's higher accuracy.

generative model does not always enjoy higher accuracy than artificial methods. For the InsectWingbeat dataset, perturbing by zeros results in higher accuracy than the generative model. The reason might be that there are many missing values in this dataset (about 57%), and in our experiments, we fill them with zeros. In this circumstance, perturbing with zeros replacement might be a better choice. However, for other datasets, the generative model consistently performs better in generating within-distribution perturbed inputs than do the artificial methods.

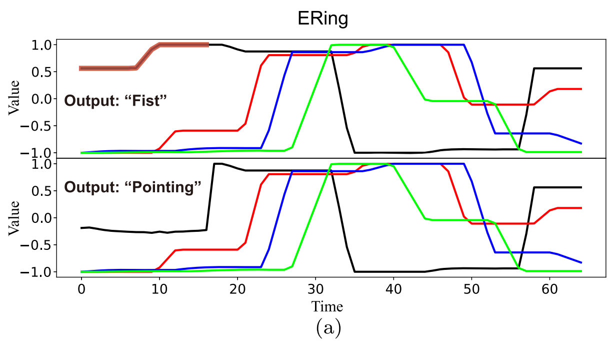
5.2. Quantitative evaluation of explanations

In this section, we calculate the SDF scores to quantitatively evaluate the saliency explanation provided by the adopted explaining methods. The calculation results are shown in Table 3. The results show that our method achieves the smallest SDF scores, which sug-

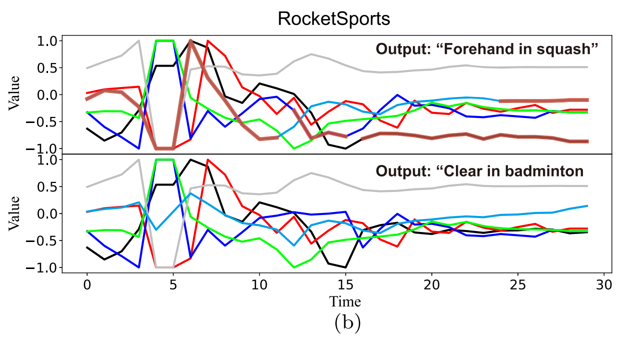
Table 3
The SDF results of the adopted saliency explaining methods (above) and the time required to obtain one explanation (below).

Dataset	IG	SG	LIME	LIME-G	Proposed Framework	
					with BPSO	with GA
ArticularyWordRecognition	349.00(25.75)	401.50(20.17)	645.00(38.77)	591.65(40.38)	**83.50(7.23)**	89.00(7.50)
	0.02s(0.00)	0.01s(0.00)	5.30s(0.43)	9.31s(0.87)	27.59s(1.43)	34.02s(1.85)
BasicMotions	86.00(6.00)	465.00(26.92)	125.00(8.32)	130.70(10.04)	73.50(4.86)	**60.00(3.22)**
	0.02s(0.00)	0.01s(0.00)	6.07s(0.56)	10.45s(0.87)	13.45s(0.90)	14.08s(0.83)
CharacterTrajectories	112.00(9.81)	84.50(7.63)	108.50(7.82)	93.27(6.53)	32.50(2.55)	**28.00(2.75)**
	0.02s(0.00)	0.01s(0.00)	3.73s(0.20)	6.51s(0.58)	23.46s(1.58)	24.21s(2.28)
Cricket	8.49(0.54)	22.23(2.18)	25.62(2.02)	29.30(2.29)	12.12(1.06)	**8.15(0.59)**
	0.08s(0.01)	0.05s(0.01)	29.26s(2.55)	29.90s(2.50)	142.43s(9.82)	141.92s(11.15)
Epilepsy	8.00(0.48)	63.50(3.36)	7.50(0.41)	6.42(0.36)	**4.00(0.32)**	5.50(0.45)
	0.02s(0.00)	0.01s(0.00)	4.39s(0.29)	6.89s(0.53)	26.41s(1.89)	29.80s(2.94)
ERing	26.50(2.64)	56.50(4.49)	16.00(1.45)	14.86(1.18)	13.00(0.83)	**12.00(0.67)**
	0.02s(0.00)	0.01s(0.00)	5.63s(0.36)	10.92s(1.09)	17.04s(1.16)	7.29s(0.72)
HandMovementDirection	70.50(4.45)	86.50(5.49)	372.00(36.72)	364.44(29.79)	72.00(6.93)	**34.00(2.89)**
	0.03s(0.00)	0.02s(0.00)	8.56s(0.68)	13.16s(1.25)	102.22s(9.21)	101.71s(6.97)
InsectWingbeat	19.50(1.93)	65.50(4.29)	23.50(2.11)	44.00(2.56)	15.50(1.18)	**14.02(0.77)**
	0.02s(0.00)	0.01s(0.00)	8.23s(0.67)	14.85s(0.98)	33.10s(9.79)	31.08s(1.59)
JapaneseVowels	84.00(6.08)	164.50(14.09)	101.50(6.32)	88.67(8.09)	38.50(1.99)	**31.00(2.73)**
	0.02s(0.00)	0.00s(0.00)	7.89s(0.43)	10.67s(0.96)	43.57s(3.06)	12.30s(0.93)
Libras	7.50(0.54)	18.00(1.02)	8.50(0.76)	7.74(0.65)	**6.00(0.46)**	7.00(0.36)
	0.03s(0.00)	0.00s(0.00)	4.93s(0.25)	7.67s(0.63)	5.06s(0.30)	4.80s(0.25)
LSST	12.50(0.95)	52.00(3.60)	10.50(0.71)	8.46(0.68)	13.00(1.23)	**8.00(0.74)**
	0.02s(0.00)	0.01s(0.00)	5.18s(0.49)	8.46s(0.48)	7.22s(0.66)	6.58s(0.50)
NATOPS	446.00(42.97)	656.00(48.58)	822.50(73.36)	818.67(50.49)	**111.50(8.38)**	128.00(11.78)
	0.03s(0.00)	0.01s(0.00)	10.66s(1.01)	18.98s(1.29)	10.59s(1.05)	9.04s(0.71)
PenDigits	5.00(0.47)	11.00(1.08)	5.50(0.50)	5.43(0.27)	**3.00(0.26)**	**3.00(0.20)**
	0.01s(0.00)	0.01s(0.00)	3.17s(0.25)	4.27s(0.38)	2.12s(0.17)	2.27s(0.20)
PhonemeSpectra	77.00(6.94)	153.50(9.71)	562.00(120.66)	632.27(82.72)	79.00(5.19)	**71.00(3.69)**
	0.02s(0.00)	0.01s(0.00)	8.09s(0.57)	15.24s(1.48)	50.25s(4.40)	46.93s(3.90)
RacketSports	34.50(1.94)	116.50(7.01)	44.00(2.26)	41.61(2.69)	25.50(1.46)	**20.50(1.25)**
	0.03s(0.00)	0.01s(0.00)	4.32s(0.43)	6.54s(0.63)	4.93s(0.28)	5.16s(0.31)
SelfRegulationSCP1	16.00(1.38)	**3.00(0.24)**	**3.00(0.24)**	3.04(0.29)	**3.00(0.22)**	**3.00(0.22)**
	0.03s(0.00)	0.02s(0.00)	9.22s(0.73)	10.47s(0.58)	35.51s(3.08)	38.70s(3.21)
SpokenArabicDigits	219.00(14.77)	935.50(75.66)	638.00(39.57)	688.94(64.33)	115.00(8.01)	**93.50(5.86)**
	0.02s(0.00)	0.01s(0.00)	4.91s(0.44)	6.68s(0.46)	21.41s(1.42)	21.91s(1.58)
StandWalkJump	10.50(0.66)	25.00(1.40)	29.00(1.61)	30.58(1.81)	12.00(0.68)	**8.00(0.44)**
	0.08s(0.01)	0.06s(0.00)	32.19s(2.13)	32.44s(1.66)	151.23s(14.26)	164.26s(14.31)
UWaveGestureLibrary	27.00(1.61)	29.00(1.48)	36.00(3.26)	38.26(3.65)	**21.00(1.81)**	22.50(1.23)
	0.02s(0.00)	0.01s(0.00)	4.42s(0.34)	8.38s(0.74)	52.47s(3.50)	56.89s(3.79)
MNIST	104.00(10.06)	167.00(15.72)	90.50(8.30)	95.51(6.70)	27.50(2.36)	**18.00(1.56)**
	0.02s(0.00)	0.00s(0.00)	4.01s(0.30)	7.98s(0.76)	9.43s(0.66)	10.71s(0.96)

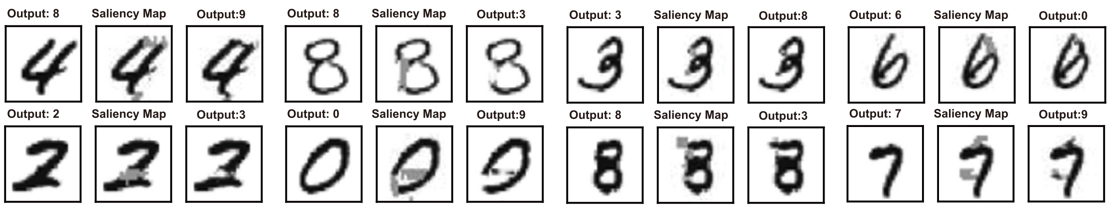
gests that fewer features are identified in the saliency maps. This helps to provide better explanations. Our framework is compatible with any binary search method, so we separately use the classic BPSO and GA methods in the identification steps of our framework. The results show that these two methods attain a very similar performance, suggesting that our framework is not sensitive to the adopted search method. Therefore, users can adopt the search method they want in the identification steps. From the aspect of time expense, gradient-based methods are very fast, since they only need several times back-propagation. But our framework needs much more forward propagation, and, besides, the generation of the perturbed inputs also takes time. Therefore, our method needs more time. However, for most of the problems here, our method can provide explanations within 20 seconds. For some applications where time constraints are not so strict, our method can be acceptable. However, if we take the quality of explanations into account, for example, in the MNIST dataset, our method identifies about 30 important features. But the others identify nearly or more than 100 features. Therefore, it is worth sacrificing time for better explanations.



(a)



(b)



(c)

Fig. 10. Counterfactual explanations for instances taken from ERing (a) and RocketSports (b) dataset. The figures above are original time series, where the most important features are highlighted. The figures below are counterfactual inputs, where the important features take alternative values. (c): Counterfactual explanations for instances from MNIST dataset.

5.3. *Counterfactual explanations*

By now, we have shown that our framework can provide explanations through sparser saliency maps than other methods, from which users can clearly identify the features that are relevant for the predicted class. However, only knowing which features are relevant is not enough. People might want to see how these relevant features would affect the classifier's output. If these features took other values, what would be the output of the classifier? Answers to this question can provide more insights for ordinary people to understand the classifier [10].

Fortunately, our framework can easily provide counterfactual inputs to answer this question. A counterfactual input is also a concrete MTS, which is very similar to the original input but has a different class predicted by the classifier. The only difference between the counterfactual input and the original input are the features identified in the saliency map. Our framework can easily provide counterfactual inputs by replacing the identified features with alternative values generated by the designed generative model. Examples of counterfactual inputs are shown in Fig. 10. These counterfactual inputs can further explain the classifier by telling users how these features would affect the classifier's output. For example, if the identified features of input with the label of "4", took large values (shown by dark pixels in the figure), the classifier would produce a label "9".

5.4. *Stability analysis*

In this section, stability analyses on the explanation results are carried out. By stability, we mean that the explanations provided should be reproducible, which means that the explanations provided for a given classification should not vary too much if we repeat the explaining process multiple times. In our framework, there are stochastic processes existing in the heuristic search process and the generation of the perturbed inputs, which makes it seem as if stability would be hard to achieve, as shown in Fig. 11, for the two different explanations provided for one input in the CharacterTrajectories dataset when the explaining process is carried out multiple times. However, we argue that this instability mainly results from a high number of degrees of freedom in the input space that can change the output of the classifier. For example, the input shown in Fig. 11a is predicted as class "1". However, the predicted class can be changed into one of the other 19 classes by perturbing different features. During the heuristic search, the stochastic processes make it uncertain which features are to be perturbed.

However, if we want to know what group of features of the input predicted as class "1" input, if replaced by other plausible values, can lead the classifier to produce class "4", will we obtain more stable explanations at the end? Answers to this question can be obtained easily. We only need to modify the fitness function (5) to (10):

$$\mathrm{Sup}(\boldsymbol{W}) = \begin{cases} 1 & \text{if } p(c|\boldsymbol{x}_{\setminus r}) \neq \text{desired class} \\ 0 & \text{if } p(c|\boldsymbol{x}_{\setminus r}) = \text{desired class} \end{cases} \tag{10}$$

where $\boldsymbol{x}_{i,j} \in \boldsymbol{x}_{\setminus r}$, if $\boldsymbol{W}_{i,j} = 1$

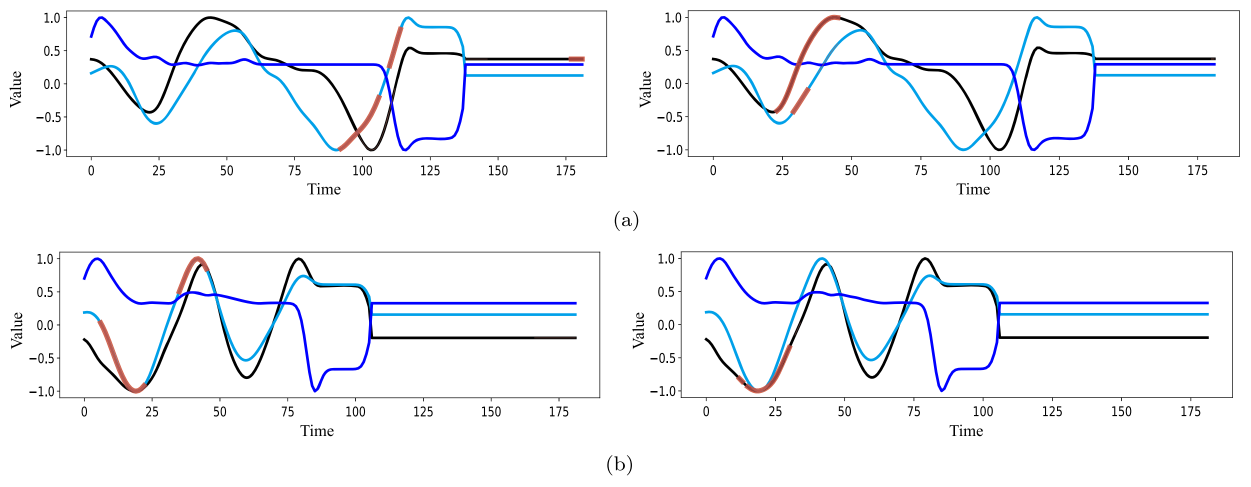


Fig. 11. Different explanations are provided for the two inputs in the CharacterTrajectories datasets, where the most important features are highlighted by red lines.

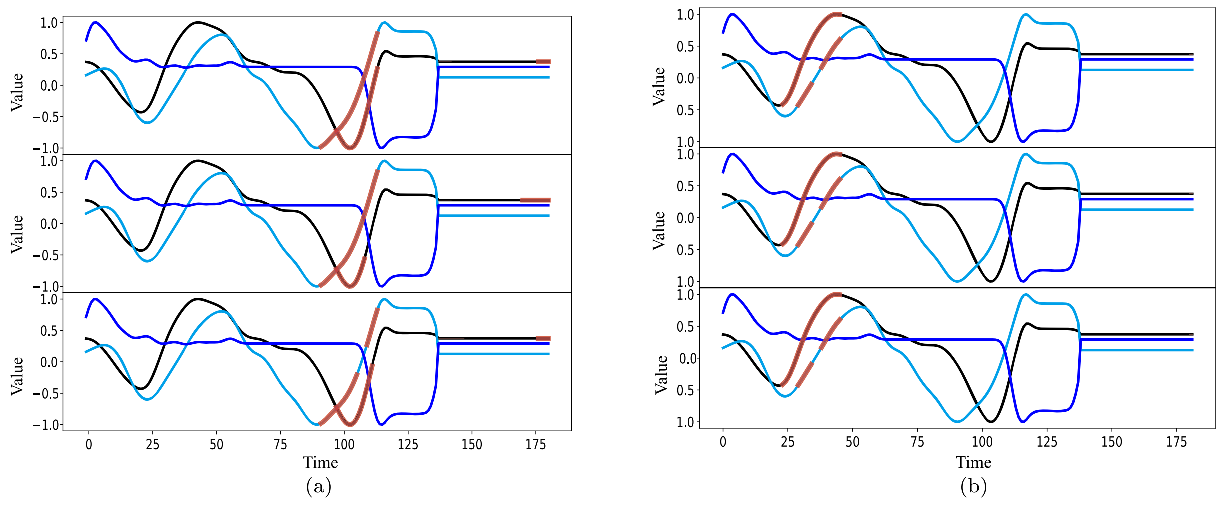


Fig. 12. The saliency maps when the desired class is set to class “4” (a) and class “11” (c).

Fig. 12 shows the saliency maps when the desired class is assigned before explaining. It can be seen that the final saliency maps are much more stable than that in the Fig. 11.

5.5. Ablation studies

In this section, we carry out ablation studies to analyse the contributions of the two key components in our framework, the designed generative model and the greedy-based segmentation and identification search strategy.

5.5.1. Contribution of the generative model

Fig. 13 shows the saliency maps provided for given classifications using our framework but with different strategies for generating the perturbed inputs, including using the generative model and using artificial perturbation methods (zeros and random noise). The results show that the generative model really helps in generating satisfying saliency maps. The identified features are compact and located in the really meaningful regions rather than scattered among the meaningless background. We believe that these improvements result from the mitigation of the OOD problem.

5.5.2. Contribution of the greedy-based segmentation and identification strategy

We propose a greedy-based segmentation and identification search strategy to mitigate the challenge in searching over the huge feature space. We compare the saliency maps provided using our proposed search strategy with those provided without this search strategy. The latter saliency maps are provided by applying the same binary search algorithm (BPSO or GA) over the original feature space, where each feature is considered as one segment. As shown in Fig. 14, without the proposed search method, the identified features are scattered everywhere, while our method provides compact saliency maps. Because the proposed strategy regarded the

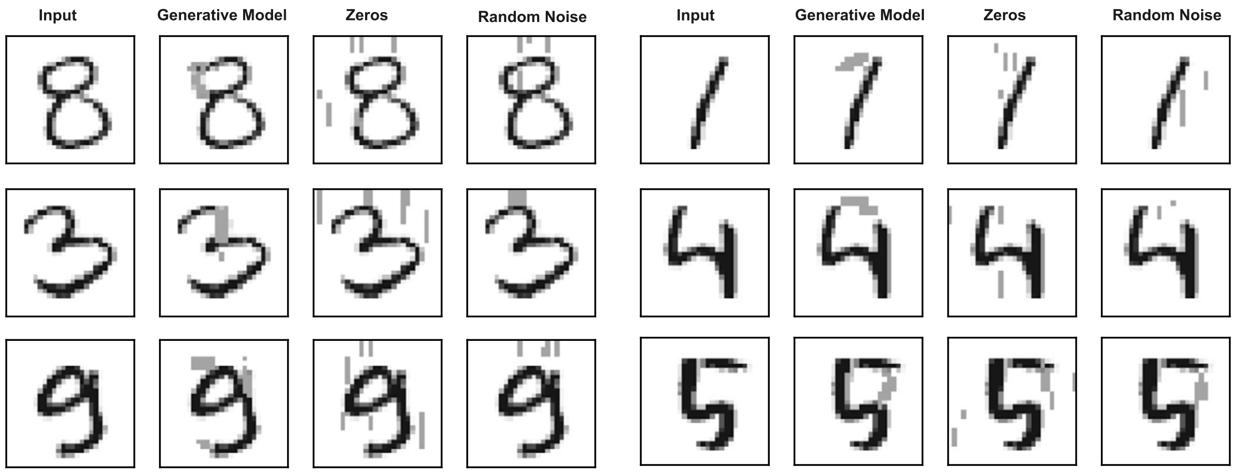


Fig. 13. The saliency maps provided by our framework but using different perturbation strategies, including using the generative model, using zeros, and random noise.

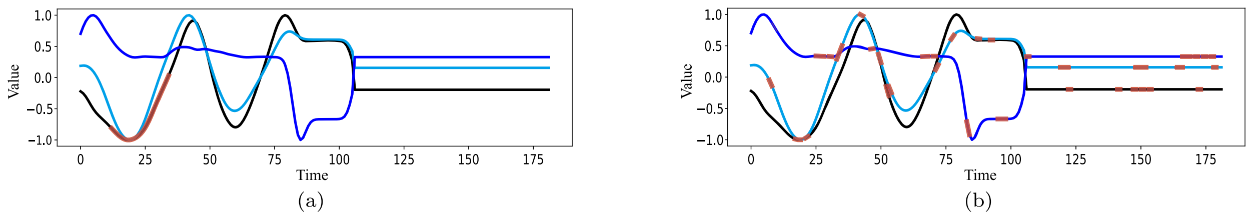


Fig. 14. The saliency maps for the given inputs (a) provided with our proposed search strategy and (b) provided by applying binary search method directly over the original features space.

features within continuous time steps as one "super-feature", the final identified features also tend to be continuous in time. In addition, our search strategy does better in terms of sparsity. On average for the MNIST, only 38.33 features are identified as relevant with this strategy while without our strategy 69.09 features are so identified. Moreover, this strategy can also notably speed up the explaining process: on average for the MNIST, it takes about 20 s to generate an explanation, but direct searching needs about 100 s.

6. Conclusions and future work

This paper promotes the development of XAI a step further in TSC. We develop a model-agnostic framework to provide saliency-based explanations for TSC through a post-hoc approach. Two challenges are addressed in this work. First is the widely acknowledged OOD problem. The accurate distribution of the training dataset might not be accessible. Therefore, we design a generative model for MTS to approximate this distribution. The results of the experiments demonstrate that the classifiers are more resistant to perturbed inputs produced by the generative model than to those produced by artificial perturbation techniques, which suggests that the generative model is effective at producing within-distribution perturbed inputs. Another challenge comes from the huge search space, which increases exponentially with the number of features an MTS has. This challenge is addressed by the proposed greedy-based segmentation and identification strategy. The search space is significantly reduced, and on it, the classic search methods, including BPSO and GA methods, can achieve satisfying results. The results of ablation studies show that the generative model and the proposed search strategy are necessary components to provide satisfying explanations. Besides saliency-based explanations, this framework has the potential to provide counterfactual explanations, which makes more sense to users and is very useful for understanding the classifier. Although we focus on TSC problems in this work, our proposed framework can be applied to any problem, such as image and language processing problems.

However, this framework does not go without its limitations. The critical point is the design of the generative model. If the generative model can not effectively produce within-distribution perturbed inputs, the OOD problem can not be addressed, and the final explanations might still be meaningless. Therefore, the performance of this framework is upper-bounded by the performance of the generative model. In the future, we hope we can develop a more powerful generative model to improve this bound. Besides, to provide an explanation, a search needs to be carried out separately, which is expensive in terms of time, making this framework difficult to use for online applications. We will also do more work in providing explanations with high efficiency in the future.

CRediT authorship contribution statement

Han Meng designed the research, conducted the experiments, analyzed and wrote the manuscript. Isaac Triguero and Christian Wagner contributed to the writing of the manuscript.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Han Meng reports financial support was provided by China Scholarship Council and University of Nottingham. Isaac Triguero reports financial support and equipment, drugs, or supplies were provided by Spanish Scientific Research Council. Isaac Triguero reports equipment, drugs, or supplies was provided by NVIDIA Corp.

Data availability

Data is available publicly as is highlighted within the paper.

Acknowledgements

The first author is funded by the University of Nottingham and the China Scholarship Council (CSC) from the Ministry of Education of P.R. China. This work is also supported by projects A-TIC-434-UGR20 and PID2020-119478GB-I00. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ins.2023.119334.

References

[1] J. Faouzi, Time series classification: a review of algorithms and implementations, in: Machine Learning (Emerging Trends and Applications), Proud Pen, 2022.
[2] P. Giudici, E. Raffinetti, Shapley-Lorenz explainable artificial intelligence, Expert Syst. Appl. 167 (2021) 114104.
[3] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Accurate and interpretable evaluation of surgical skills from kinematic data using fully convolutional neural networks, Int. J. Comput. Assisted Radiol. Surg. 14 (2019) 1611–1617.
[4] R. Chen, X. Yan, S. Wang, G. Xiao, DA-Net: dual-attention network for multivariate time series classification, Inf. Sci. 610 (2022) 472–487.
[5] Z. Zheng, Z. Zhang, L. Wang, X. Luo, Denoising temporal convolutional recurrent autoencoders for time series classification, Inf. Sci. 588 (2022) 159–173.
[6] W. Ding, M. Abdel-Basset, H. Hawash, A.M. Ali, Explainability of artificial intelligence methods, applications and challenges: a comprehensive survey, Inf. Sci. 615 (2022) 238–292.
[7] U. Kamath, J. Liu, Explainable Artificial Intelligence: An Introduction to Interpretable Machine Learning, Springer, 2021.
[8] Z.C. Lipton, The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery, Queue 16 (2018) 31–57.
[9] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nat. Mach. Intell. 1 (2019) 206–215.
[10] Explanation sets: a general framework for machine learning explainability, Inf. Sci. 617 (2022) 464–481.
[11] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?" Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.
[12] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017.
[13] S. Tonekaboni, S. Joshi, K. Campbell, D.K. Duvenaud, A. Goldenberg, What went wrong and when? Instance-Wise Feature Importance for Time-Series Black-Box Models, in: Advances in Neural Information Processing Systems, vol. 33, Curran Associates, Inc., 2020, pp. 799–809.
[14] J. Crabbé, M. Van Der Schaar, Explaining time series predictions with dynamic masks, in: Proceedings of the 38th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, vol. 139, 2021, pp. 2166–2177.
[15] A.A. Ismail, M. Gunady, H. Corrada Bravo, S. Feizi, Benchmarking deep learning interpretability in time series predictions, in: Advances in Neural Information Processing Systems, vol. 33, Curran Associates, Inc., 2020, pp. 6441–6452.
[16] J. Bento, P. Saleiro, A.F. Cruz, M.A. Figueiredo, P. Bizarro, TimeSHAP: explaining recurrent models through sequence perturbations, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD'21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 2565–2573.
[17] P. Hase, H. Xie, M. Bansal, The Out-of-Distribution problem in explainability and search methods for feature importance explanations, in: Advances in Neural Information Processing Systems, vol. 34, 2021.
[18] C.-H. Chang, E. Creager, A. Goldenberg, D. Duvenaud, Explaining image classifiers by counterfactual generation, in: International Conference on Learning Representations, 2019.
[19] S. Kim, J. Yi, E. Kim, S. Yoon, Interpretation of NLP models through input marginalization, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2020, pp. 3154–3167.
[20] H. Meng, C. Wagner, I. Triguero, Feature importance identification for time series classifiers, in: 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2022, pp. 3293–3298.
[21] R.C. Fong, A. Vedaldi, Interpretable explanations of black boxes by meaningful perturbation, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3449–3457.
[22] Q. Du, J. Xu, Model-agnostic local explanations with genetic algorithms for text classification, in: The 33rd International Conference on Software Engineering &, Knowledge Engineering, 2021.
[23] M.T. Ribeiro, S. Singh, C. Guestrin Anchors, High-precision model-agnostic explanations, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[24] K. Vafa, Y. Deng, D. Blei, A. Rush, Rationales for sequential predictions, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online and Punta Cana, Dominican Republic, Association for Computational Linguistics, 2021, pp. 10314–10332.
[25] X.-F. Song, Y. Zhang, D.-W. Gong, X.-Z. Gao, A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data, IEEE Trans. Cybern. (2021) 1–14.
[26] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: visualising image classification models and saliency maps, in: Workshop at International Conference on Learning Representations, 2014.
[27] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in: D. Precup, Y.W. Teh (Eds.), The 34th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, vol. 70, 2017, pp. 3319–3328.
[28] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: D. Precup, Y.W. Teh (Eds.), The 34th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, vol. 70, 2017, pp. 3145–3153.
[29] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, B. Kim, Sanity checks for saliency maps, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018.
[30] P. Sturmfels, S. Lundberg, S.-I. Lee, Visualizing the impact of feature attribution baselines, Distill 5 (2020) e22.
[31] R. Assaf, I. Giurgiu, F. Bagehorn, A. Schumann, MTEX-CNN: multivariate time series explanations for predictions with convolutional neural networks, in: 2019 IEEE International Conference on Data Mining (ICDM), 2019, pp. 952–957.
[32] I. Niño-Adan, D. Manjarres, I. Landa-Torres, E. Portillo, Feature weighting methods: a review, Expert Syst. Appl. 184 (2021) 115424.
[33] R. Kohavi, G.H. John, Wrappers for feature subset selection, Artif. Intell. 97 (1997) 273–324.
[34] R. Espinosa, F. Jiménez, J. Palma, Multi-surrogate assisted multi-objective evolutionary algorithms for feature selection in regression and classification problems with time series data, Inf. Sci. 622 (2023) 1064–1091.
[35] M. García-Torres, F. Gómez-Vela, B. Melián-Batista, J.M. Moreno-Vega, High-dimensional feature selection via feature grouping: a variable neighborhood search approach, Inf. Sci. 326 (2016) 102–118.
[36] X.-F. Song, Y. Zhang, Y.-N. Guo, X.-Y. Sun, Y.-L. Wang, Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data, IEEE Trans. Evol. Comput. 24 (2020) 882–895.
[37] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, Y. Li Brits, Bidirectional recurrent imputation for time series, in: Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018.
[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017.
[39] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (1997) 2673–2681.
[40] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, Improved training of Wasserstein GANs, in: Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017.
[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, vol. 27, Curran Associates, Inc., 2014.
[42] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein GAN, arXiv, 2017.
[43] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, in: Proceedings of the 33rd International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 48, PMLR, New York, New York, USA, 2016, pp. 1050–1059.
[44] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1998.
[45] L.J.V. Miranda, PySwarms, a research-toolkit for particle swarm optimization in python, J. Open Sour. Softw. 3 (2018).
[46] A. Bagnall, H.A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, E. Keogh, The UEA multivariate time series classification archive, arXiv preprint, arXiv:1811.00075, 2018.
[47] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.
[48] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.
[49] C. Agarwal, A. Nguyen, Explaining image classifiers by removing input features using generative models, in: Proceedings of the Asian Conference on Computer Vision (ACCV), 2020.