Universiteit Leiden
Opleiding Informatica

A Dual-Technique Approach: Shapelet and TimeGAN for Counterfactual Explanations of Univariate Time Series

Name: Wei Chen

Date: 08/25/2023

1st supervisor: Dr. Niki van Stein
2nd supervisor: Prof.dr. T.H.W. Bäck

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

As the fast development of artificial intelligence (AI) continues, the concept of explainable artificial intelligence (XAI) has gained increasing attention lately. While a majority of efforts have been directed toward providing explanations in the image and tabular domains, considerably fewer methods have been introduced for time series data. Similarly, counterfactual-based XAI methods for explaining time series classification models have not received as much focus from experts. In this work, we leverage the **Shapelet Transform (ST)** and **TimeGAN** to propose a model-agnostic, instance-based (counterfactual-based) approach that provides counterfactual explanations for any time series classifier. We validate our method using a real-world time series classification dataset from the *UCR Time Series Archive*. Our results indicate that the counterfactual instances generated by **Time-CF** demonstrate superior performance in terms of the four metrics: closeness, omission, plausibility, and sparsity, when compared to other baseline methods.

Contents

1

Introduction

In recent years, the performance of Machine learning models became increasingly impressive. Advances in capability of data computation, development of sophisticated algorithms, and the proliferation of high-quality datasets have all contributed to this upward trend [31, 32]. Machine learning models and methods are being leveraged in almost every work of life, such as automated driving, banking, healthcare, aerospace, and geo-science, where detailed explanations for each decision are essential [30]. However, the recent rise in its black-box nature is a matter of considerable public concern. A black box model, as the name suggests, is inherently opaque. It is difficult for people to clearly explain why it predicts in a certain manner, as the underlying reasoning behind its decision is not easily comprehensible. In domains such as medicine and autonomous vehicles, where the occurrence of errors could be fatal, there is an urgent need either to provide **post-hoc** explanations based on the decisions made by the models or to build inherently explainable models before predictions **(ante-hoc)**.

In response to the opacity property of machine learning models, the field of XAI has emerged to address this issue. Although XAI methods have been effectively applied to domains such as images, text, and tabular data, there has been relatively less focus on time series data [33]. The main reason for this is temporal dependency within a time series instance, where each time point in a time series instance tends to rely on its predecessor. The temporal dependency therefore adds complexity to the interpretability and explainability of time series data particularly in developing XAI methods specifically for this type of data. In general, XAI approaches for time series data can be categorized into three types. First, time-points-based methods, which explore how important each time step and feature are to the prediction. Second, subsequences-based (or shapelets-based) methods, which consider the segments of the time series instance that will most significantly impact the outcome. Lastly, instance-based methods, which take all time steps within a time series instance into account[37].

For instance-based methods, in addition to generating feature-based and prototype-based explanations for predictions, counterfactual-based methods have recently gained wider popularity [37]. Counterfactual, as the name suggests, involves the generation of an instance, that does not exist in the original dataset. Interpretability and explainability are typically

achieved by generating at least one counterfactual instance based on the to-be-explained instance. Then, a comparison between these instances is made. Specifically, the most common approach taken by experts to transform the to-be-explained instance into a counterfactual one is based on perturbation. This means altering as few elements as possible from the to-be-explained instance, for it to be classified as a different class from the original one. By comparing the differences in the modified parts, end users can easily understand which factors influence the classifier.
In this work, the focus is on explaining the prediction by time series classifier using GAN to generate counterfactual instances.

1.1 Research Questions

In this thesis, we will investigate the following research questions to shed light on the challenges and solutions for providing convincing explanations for machine learning models, especially those that focus on time series data.

- How might the black-box nature of machine learning models be addressed to reduce opacity and enhance interpretability through the application of XAI principles?
- How can XAI methods be effectively applied to time series data and time series classification tasks, particularly considering the complexities introduced by the temporal dependency inherent to time series data?
- Given that there have been a number of XAI methods designed specifically for time series classification tasks, which among them can be considered particularly effective?
- The generation of counterfactual explanation is a useful approach within the instance-based category of methods for explaining time series data. What manner of generation can result in meaningful counterfactual explanations?
- How can we alter the given to-be-explained instance minimally yet effectively so that it is classified as a different class by the classifier?
- Which type of Generative Adversarial Network (GAN) is most suitable for generating counterfactual time series instances that adhere to the dataset distribution?

1.2 Outline

The remainder of this thesis is structured as follows: In the Related Works chapter (Chapter 2), we showcase the recent efforts on XAI methods for time series classification. In the Preliminary Concepts chapter, (Chapter 3), a comprehensive description of the preliminary concept of time classification series is described. Following this, we detail our local model-agnostic approach, particularly focusing on the generation of counterfactual instances in the Methodology chapter (Chapter 4). Subsequently, experiments and corresponding evaluations are presented in Chapter 5. Finally, we summarize our findings and draw our conclusion in the final chapter of the thesis 6.

2

Related Works

According to [37], time points-based explanation tend to be divide into *Attributions* and *Attentions*, both of which take into account the feature-importance of time series. **Lime** and **SHAP** are representative of attributions approaches. **Lime** operates by perturbing the to-be-explained instance and generating random instances around it with the assistance of a sparse linear model [29]. **SHAP**, on the other hand, borrows their ideas from the classic Shapley values in game theory, and it achieves explanation by computing the contribution of each feature to the prediction [20]. Nevertheless, they are not primarily tailored to explain time series data, which means that the explanations they produce for time series classifiers can be unconvincing, even though they can be adapted to accommodate time series data. In [10], Guillem´e *et al.* propose **LEFTIST** to adapt **Lime** to support time series classification. **LimeSegment** [35] extends Lime on the basis of **LEFTIST**, by introducing improved segmentation and perturbation algorithm. **TimeSHAP** [3], which is introduced by João Bento *et al.*, shares the same idea with **SHAP**. Its recurrent explainer is constructed based on KernelSHAP and is specifically adapted to handle time series data. In the domain of computer vision, **Class Activation Maps** (CAM) is an approach used for classification tasks by highlighting the most important regions of an image [44]. Building upon **CAM**, Selvaraju1 *et al.* propose a gradient-weighted class activation mapping technique, named **Grad-CAM** [34] to exclusively interpret deep learning models, such as CNNs. This technique does not mandate the presence of a Global Average Pooling (GAP) layer like the original CAM technique does. However, it is not explicitly designed for sequential data. An alternative approach to it, called **Salience-CAM** has been proposed by Zhou *et al.*, which is specifically designed to visually explain time series data in a point-to-point manner [45].

Instance-based explanation methods, particularly the Counterfactual-based direction have progressively emerged as alternative options to the time points-based strategies. In the context of counterfactuals, interpretability and explainability are attained by creating synthetic instances for visual comparison. Wachter *et al.* were pioneers in generating counterfactual instances to elucidate anomalies [39]. This involves simultaneously minimizing a prediction loss function that drives the prediction on counterfactuals towards a different

class, and a distance loss function that ensures the similarity between counterfactuals and to-be-explained instance. Recent efforts about using counterfactual instances as explanation are proposed essentially based on Wachter *et al.*'s theory. *Ates et al.* propose to explain multivariate time series data, named **CoMTE** [1]. In this method, it selects an instance from the training set as distractor, and then takes a specific interval from this distractor to replace a corresponding interval in the instance of the time series that is to be explained. This approach was among the first to generate explanation in the multivariate time series context. **Native-Guide** [6] is another counterfactual-based method, introduced by *Delaney et al*, following four desirable properties: Sparsity, Proximity, Plausibility, and Diversity. In essence, they find the nearest-unlike neighbor of the to-be-explained instance and generate counterfactual instance for explanation. **TimeX** in [8] utilizes the concept of Dynamic Barycenter Averaging (DBA) to generate prototype of the opposite class of the to-be-explained instance. Subsequently, the prototype is divided into equally sized intervals (or segments) to continually optimized the to-be-explained instance until a valid counterfactual instance is generated. **MG-CF** [17], **SG-CF** [16], and **SETS** [2] are also extensions of [39], but incorporate Shapelets into consideration. Specifically, **MG-CF** generates counterfactual instances by using **Shapelet Transform** [11] to extract all potential Shapelet candidates and then replaces the same part of the to-be-explained instance. Moreover, **SG-CF** adds an additional term to the previous formulation of [39], which enables the similarity between counterfactual instance and most salient shapelet, for more robust explanation. In **SETS** [2], *Bahri et al.* propose to generate counterfactuals for multivariate setting by leveraging **Shapelet Transform** to sample Shapelets from the raw dataset for each dimension. The Shapelets from the instance to be explained are contiguously replaced one by one with the corresponding sampled Shapelets in the opposite instance until the class of the generated instance is classified as the same as that of the opposite one. [14] and [5] employ Generative Adversarial Network (GAN) for counterfactual explanation [9]. However, unlike standard GAN structure, they additionally introduce pre-trained classifier to the optimization of loss function. In addition, the input of the generator is not random noise but a time series instance as they also incorporate the concept of the Conditional Generative Adversarial Network (cGAN) [23].

3

Preliminary Concept

In this section, we outline the preliminary concept of counterfactual generation for time series data, focusing specifically on the problem statement of univariate time series data. We also introduce four properties as benchmarks for the generation of meaningful counterfactual explanations.

3.1 Problem definition

Univariate time series data refers to time series data where the number of features (or attributes) is singular. Assume a univariate time series dataset

$$D = \{T_1, T_2, T_3, ..., T_n\} \tag{3.1.1}$$

where n is the number of instance within this dataset, and each time series instance T_i has a corresponding class label, i.e., $C = \{c, c'\}$. For a time series instance $T_i = \{t_1, t_1, t_3, ..., t_m\}$ with length m, its structure is arranged chronologically, with each time-step t_i represents a real value. To generate a meaningful counterfactual instance T_{cf} for a to-be-explained instance T_o, it is essential to utilize the training set of the time series data D to train a classifier f of interest beforehand. Once the pre-trained classifier is obtained, a counterfactual generation model M is used to perturb the to-be-explained instance T_o associated with its class, say c, to be classified as the opposite class c'.

3.2 Shapelets

Shapelets are sub-series (or sub-sequences) of a time series instance (as depicted by the green sub-sequences in figure 3.2.1). They can be extracted from time series data using the Shapelet Transform algorithm for the purpose of time series classification. In the context of XAI, Shapelets can be leveraged to provide more informative and intuitive explanations for undesired results compared to feature-based methods.

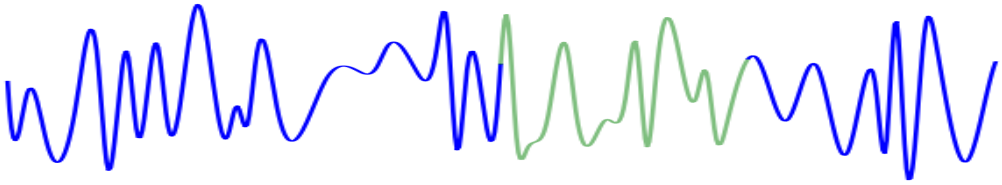


Figure 3.2.1: An example of a time series instance, where the blue curve line represents a univariate time series instance, while the green curve line is one of its Shapelets.

3.3 Properties of meaningful counterfactual explanations

Although there is no a consensus on the evaluation of counterfactual-based methods, closeness (or proximity), sparsity, plausibility (or interpretability), and diversity have been widely used in recent papers [1, 2, 6, 8, 16, 17, 24, 41]. In a similar vein, except for assessment on diversity that is replaced by omission, we utilize the remaining three measures for our evaluation. For more details on how we assess these four properties, refer to subsection 4.3.

- **Closeness**, as its name suggests, seeks to ensure that the to-be-explained instance is as similar to the counterfactual instance as possible [7].
- **Plausibility** is used to assess whether the counterfactual instance is generated within the distribution of raw data [22, 27].
- **Sparsity** serves as a gauge for the number of changes between the counterfactual instance and the to-be-explained instance [12, 13].
- **Omission** acts as a measure to demonstrate that how many classifiers a specific counterfactual-based method can explain, in other words, generate at least one counterfactual instance.

4

Methodology

We propose **Time-CF**, a local model-agnostic method to explain a specific instance for any classifier, whether they are traditional classifiers, e.g., k-nearest neighbor (KNN); or they are deep learning classifiers, e.g. Convolutional neural network (CNN). The details of the counterfactual generation model M are depicted in Figure 4.0.1. Specifically, to implement a counterfactual generation model, the process begins with the random sampling of a designated number of Shapelet candidates, the lengths of which are pre-specified within a certain range, from the training set using the Random Shapelet Transform (RST). Subsequently, these collected Shapelets are sorted by their respective information gain in descending order and then filtered to the top N, which means how many Shapelets are retained. Following this, each group of Shapelets corresponding to the same time interval, for example, 8:00am to 4:30pm, is brought to a variant of the GAN called TimeGAN [42], which is specifically designed for time series generation, in order to produce multiple Shapelets for diverse explanation. Once the artificial Shapelets are obtained, they are used

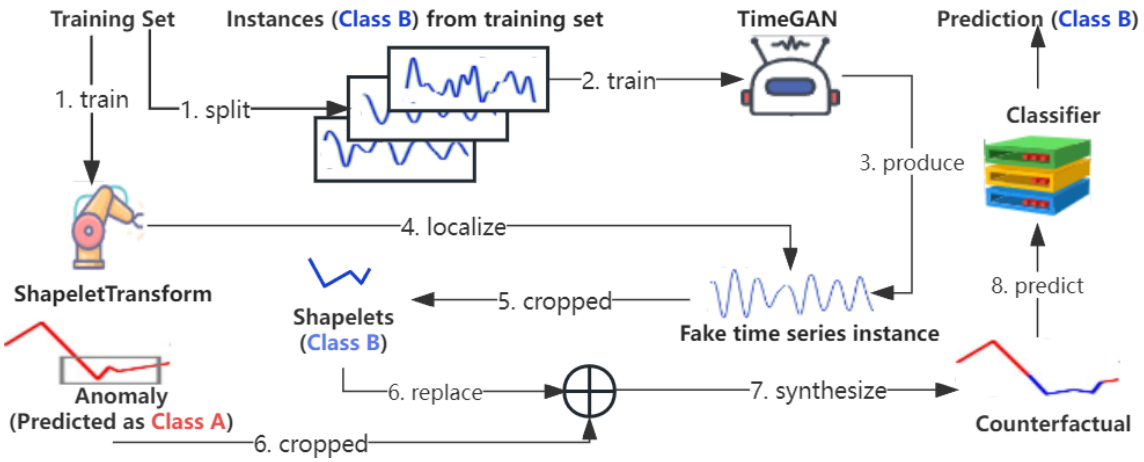


Figure 4.0.1: The structure of counterfactual generation model.

to replace the corresponding time interval of the to-be-explained instance Shapelet-by-Shapelet. This section delineates the detailed procedure for generating a counterfactual instance for explanation, which entails the extraction of discriminative Shapelet candidates and the ultilization of TimeGAN to generate meaningful counterfactual series.

4.1 Shapelets extraction

Algorithm 1: Shapelets Extraction

Data: Dataset: D, Random Shapelet Transform algorithm: $RST()$
Result: Shapelet candidates: S

$S \leftarrow \emptyset$;
$X_{train},\ y_{train},\ X_{test},\ y_{test} \leftarrow processDataSet(Dataset)$;
$S \leftarrow RST(len_{min},\ len_{max},\ ,\ max\_shapelets,\ time\_limit,\ X_{train})$;

Shapelets, also referred to as subsequences, constitute part of a time series instance.. Shapelets tend to provide important patterns and trends about a series, while a scattering of features (feature-based) might not. For this reason, perturbation should be applied on contiguous intervals instead of fragmented time-points. However, not all the Shapelets contained in a time series instance have crucial information that facilitates the correct prediction of classification. As previously discussed, with the aid of the Shapelet Transform (ST) algorithm, Shapelets with less discriminative information are filtered out, and those with relatively high discriminative power are retained for subsequent perturbation steps. In practice (See Algorithm 1), we utilize Random Shapelet Transform (RST), which accommodates time limitations, thus providing a time-saving advantage in comparison to the Shapelet Transform (ST). It takes as input the training set and outputs extracted Shapelets. The primary stages of the Shapelet Transform (ST) or Random Shapelet Transform (RST) process include candidate extraction, computation of information gain, and ultimately Shapelet selection.

4.1.1 Candidates Extraction

The Shapelet Transform (ST) is an algorithm designed for time-series classification tasks. It extracts Shapelets from a dataset (usually the training set) and considers these extracted Shapelets as transformation features. In this step, a subset of possible Shapelets (or subsequences) is randomly extracted from each time series instance within the dataset, with various lengths and starting positions. Suppose there is an univariate time series dataset with binary class $D = \{T_1, T_2, T_3, ..., T_n\}$, in which n is the number of instances within this dataset. For a given time series instance T_i with the length of m, a set that contains all its potential Shapelets Q_i with the length of k can be denoted as:

$$Q_i = \{p_1, p_2, ..., p_{m-k+1}\} \tag{4.1.1}$$

where $(m - k + 1)$ indicates how many distinct Shapelets can be extracted from T_i, and p_j is a Shapelet of T_i with length k. Further, the whole of the Shapelet candidates for the dataset is defined as:

$$S = \{Q_1, Q_2, Q_3, ..., Q_n\} \tag{4.1.2}$$

With the Random Shapelet Transform (RST), which does not consider all Shapelets, the size of the extracted Shapelets set Q_i for a given time series instance is likely to be less

than $(m - k + 1)$. Lastly, all Shapelets in each set Q_i are sorted together based on their information gain. The argument *max_shapelets* is used to specify the maximum number of Shapelets that should be retained in the end.
In practice, Random Shapelet Transform algorithm is introduced due to the long time Shapelet Transform (ST) takes to find all the possibilities. Random Shapelet Transform (RST) algorithm is essentially a variant of Shapelet Transform. The main difference is how Shapelets are extracted. In the Shapelet Transform (ST) algorithm, every distinct Shapelet is considered, which can be time-consuming, particularly when the length of the time series instance is long. In contrast, Random Shapelets Transform borrows the idea from the binary shapelet transform, significantly reducing the computation time. The mechanism of Random Shapelet Transform (RST) can be roughly summarized as continuously extracting Shapelet candidates and discarding those Shapelet candidates with low information gain in batches.

4.1.2 Information gain computation

Information Gain is a term used in the context of either decision tree or machine learning to select features, synonymous with Kullback–Leibler divergence. Hence, Shapelet Transform (ST) algorithm adopts it as a necessary metric to estimate the discriminative power of a Shapelet to further rank the extracted Shapelet candidates in the time series classification setting. Once the set of Shapelet candidates is obtained in extraction step, the step of information gain computation starts with calculating the distance between each Shapelet candidate and each sub-sequence of each time series instance. Equation 4.1.3 demonstrates how the distance between a particular Shapelet candidate s_i with length k and a time series instance T_j (from dataset D) is calculated, where d_i could be calculated using either *Euclidean distance formula* (See 4.3.2) or *Dynamic time warping distance.*

$$Distance(i,\ j) = \ min\{d_1, d_2, ..., d_{m-k+1}\} \tag{4.1.3}$$

Then the resulting ordered set of distances W_i for s_i will be sorted in ascending order as the following equation shown:

$$W_i =< (Distance(i,j), c),\ (Distance(i,j+3), c'),\ ...,\ (Distance(i,j+1), c') > \tag{4.1.4}$$

where $(Distance(i,j), c')$ is a tuple and c' represents the class of time series instance T_j is c'. Subsequently, a division point (or threshold) is introduced to split all elements in W_i into two parts. Lastly the information gain of s_i can be computed with a certain strategy, e.g., Entropy or Gini Impurity.

4.1.3 Shapelet Selection

After the information gain of every Shapelet candidate s_i is organized into a list, Shapelet candidates with class c, to which the to-be-explained instance belongs, are discarded, and only Shapelet candidates with class c' are retained for the follow-up replacement step. As the 3^{th} line in Algorithm 1 shown, Random Shapelet Transform (RST) generally needs five parameters, in which the range of the length of extracted Shapelet candiates is determined by the combination of len_{min} and len_{max}; *time_limit* demonstrates how long the operation of extraction will last and *max_shapelets* means the number of Shapelets expected to be extracted per class. Therefore, in addition to the Shapelet candidates whose class are c are discarded, the parameter *max_shapelets* also further restrict the size of the candidates.

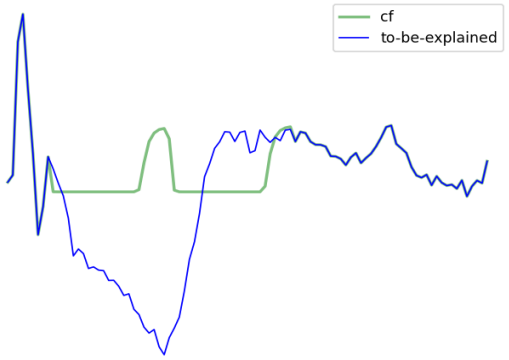


Figure 4.2.1: Time series counterfactual instance, where the blue curve line represents the to-be-explained (original) time series, while the green curve line stands for its corresponding generated counterfactual instance.

4.2 Couterfactual generation

Algorithm 2: Generating counterfactual explanations

Data: Shapelet candidates: S, to-be-explained instance with class c: T_o
Result: Counterfactual explanation with class c': T_{cf}

1 $S_{target} \leftarrow \emptyset$; /* A set of target Shapelets (real) from class c' */
2 $S_{fake} \leftarrow \emptyset$; /* A set of target Shapelets (fake) from class c' */
3 $CF \leftarrow \emptyset$; /* A set of counterfactual instances with class c' */
4 $f \leftarrow classifier.fit(X_{train}, y_{train})$; /* Trained a certain classifier */
5 **for** s_i *in* S **do**
6 $start\_pos, s_{len} \leftarrow s_i.info$;
7 **if** $s.class$!= $T_o.class$ **then**
8 $S_{target} \leftarrow crop(start\_pos, s_{len}, T_o.class, X_{train})$; /* Get the Shapelets starting from $start\_pos$ and end in $start\_pos + s_{len}$ from training set */
9 $S_{fake} \leftarrow TimeGAN(S_{target})$;
10 **for** s_{fake} *in* S_{fake} **do**
11 $T_{fake} = replace(T_o, s_{fake}, start\_pos, s_{len})$; /* Replace shapelets with the same time range, of T_o, with s_{fake} */
12 **if** $f(T_{fake}) == s.class$ **then**
13 $T_{cf} = T_{fake}$;
14 $CF.add(T_{cf})$; /* Store counterfactual instance */

Inspired by the positive performance of Shapelet-based method for time series classification tasks and by [5, 14, 38] that all leverage their respective Generative adversarial networks (GANs), We likewise apply a variant of the Generative adversarial networks, named *TimeGAN* as a core component of our counterfactual generation model M. Algorithm 2 depicts how the counterfactual instances is generated after the extraction of Shapelets. It first iterates over the Shapelet candidates set S. For each candidate s_i, it is essential to identify its start and end positions within the corresponding instance (See 6^{th} line in Algorithm 2). Next, *crop*() function is executed to extract the target Shapelets S_{target} by cropping each interval with the same time range (start and end positions) in

each target time series instance, that is, instance from class c'. Later on, *TimeGAN* begins with taking as input a group of sub-sequences S_{target} from class c' with the same time range and then outputs a number of fake (artificial) sub-sequences, i.e., Shapelets. The final process includes iterating through the set of target Shapelets (fake) and using the custom *replace*() method to replace the contiguous time-points (ranging from the start position and end position stored previously) of the to-be-explained time series instance T_o, with one of the target Shapelets (fake) generated by TimeGAN. More importantly, once a synthetic time series instance T_fake is yielded, having the pre-trained classifier f predict its class is indispensable. This step reserves the fake time series instances that are identified as counterfactual, for diverse explanations.
Figure 4.2.1 shows the visual comparison between generated counterfactual instance (green curve line) and the to-be-explained instance (blue curve line). The perturbation is acted on a contiguous segment of the original one and it results in a new instance classified as different class.
As mentioned in previous section, the most typical counterfactual-based method was proposed by *Wachter et al* [39], in which they introduce an optimization problem to simultaneously minimize the distance loss and the prediction loss.

$$\underset{T_{cf}}{\operatorname{argmin}} \max_{\lambda} \lambda(f(T_{cf}) - c')^2 + d(T_{cf}, T_o) \tag{4.2.1}$$

This equation (4.2.1) describes their attempt to find a counterfactual instance T_{cf} categorized as class c' (different from that of the original one) while being closer to the original instance, i.e., the to-be-explained instance. However, without the focus on contiguity, it cannot guarantee intuitive explanation [8]. In our counterfactual generation model M, it is also desirable that the generated counterfactual instance T_{cf} is close to the to-be-explained instance, and that the counterfactual instance is consistently predicted as another label c'. To meet the requirements, Shapelet Transform (ST) and *TimeGAN* that receives the Shapelets coming from the same intervals can jointly serve as an effective solution to greatly ensure the generated counterfactuals are close to the original one and to be classified as c'.

4.3 Evaluation measures

As mentioned in section 3.3, there are four properties leveraged to measure whether the generated counterfactual explanations are meaningful. Although the generated instance can be identified as counterfactual instance once it is classified as the target class, it is still necessary to further consider whether the way of generating counterfactual instances is not only effective but also efficient.

- **Closeness.** To measure the closeness between the counterfactual instance and the to-be-explained instance, both the Manhattan distance (L_1-norm) and the Euclidean distance (L_2-norm) are utilized. It is safe to say that the fewer perturbations performed, and the smaller the changes for each perturbation, the closer the distance between the to-be-explained instance and its counterfactual instances. The formula of L_1-norm (equation 4.3.1) and L_2-norm (equation 4.3.2) are shown as below:

$$L1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{4.3.1}$$

$$L2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4.3.2}$$

where n represents the length of an univariate time series instance, while x and y are different series.

- **Sparsity.** Another important metric for a meaningful counterfactual instance is similarity. It is similar to property closeness but the main difference is that closeness is evaluated by the distance between two instances in a multi-dimensional space $\mathbb{R}^n$, where n indicates the length of the univariate instances, while property sparsity focus on how many values of an instance are perturbed. Applying a few modifications that change an instance's class can easily provide intuition for analysis. For this reason, the number of perturbations on the to-be-explained instance should be minimized. Equation 4.3.3 is used to assess whether a counterfactual generation model M can generate counterfactual instance by perturbing little time-points on the to-be-explained one.

$$\text{Sparsity} = 1 - \frac{1}{n}\sum_{i=1}^{n} I(a_i \neq b_i) \tag{4.3.3}$$

  In this equation, n represents the length of a time series instance, while $\sum_{i=1}^{n} I(a_i \neq b_i)$ is a indicator function where it takes the value 1 when the value of the time-point i in series a (a_i) is not equal to the value of the time-point i in series b (b_i), or 0 otherwise. By analogy, similarity would be equal to 0 if a counterfactual instance is produced by changing all the time-points on the to-be-explained instance.

- **Plausibility.** Obtaining a generated instance that is also identified as a counterfactual instance does not necessarily mean it is relevant. In contrast, the instances generated can become out of touch with reality if some values are altered beyond the distribution of the raw data. For instance, ECG200 is a binary-class univariate dataset that collects electrocardiogram (ECG) signals with values typically ranging from -5 to 5. However, without incorporating intentional design into counterfactual generation model, perturbed values are more likely to deviate significantly, such as -500 or 100, which are implausible and fall outside the range of human indicators. To detect and evaluate out-of-distribution instances, the isolation forest method (Liu et al.) [18] is utilized.

- **Omission.** Many model-agnostic methods in XAI claim to explain any classifiers but often fail to live up to this assertion. Therefore, we introduce the measure of omission to evaluate the percentage of classifiers that a given time series XAI method is unable to explain. For example, given an instance to be explained, the omission for a classifier, such as CNN, for a certain dataset is calculated as follows:

$$\text{Omission (CNN)} = \frac{1}{K}\sum_{k=1}^{K} \begin{cases} 1 & \text{if at least one counterfactual instance is generated} \\ 0 & \text{otherwise} \end{cases} \tag{4.3.4}$$

  where K is the number of random seeds.

5

Experiment and Result

5.1 Experiment setup

As stated, Shapelet Transform (ST) and TimeGAN are both crucial components for counterfactual generation. Therefore, it is important to consider the setting of (hyper)parameters. See Table 5.1.1 and Table 5.1.2 for details. Additionally, in Appendix, we present representative counterfactual instances generated by various counterfactual-based methods, as shown in Figures A.0.1 and A.0.2.

To build the counterfactual generation model M, Random Shapelet Transform (RST) and TimeGAN are implemented with the help of two existing open-source libraries, namely sktime [19] and YData Synthetic [26], respectively. In addition, our approach is evaluated on each of four classifiers. Further, each classifier is trained with four different univariate datasets. In practice, there are several processes in our implementation involved in randomness. Specifically, they are classifiers (Convolutional Neural Networks (CNNs) Classifier, Diverse Representation Canonical Interval Forest Classifier (DrCIF), and Catch22 with Random Forest (RF) Classifier); Random Shapelet Transform; and TimeGAN. For this reason, the average of results for three repetitive experiments with random seeds 111, 222, and 333 are taken. On top of that, we consider two different instances from distinct classes, specifically the positive class and negative classes, for comparison. The main goal for this experiment is the focus on the evaluation of closeness, sparsity, and plausibility. Lastly,

Table 5.1.1: The selection of (hyper) parameters for Shapelet Transform (ST), where *min_shapelet_length* and *man_shapelet_length* represents the range of the size of the Shapelets extracted, both of which depends on *seq_len*, i.e., the length of the time series instance. Furthermore, *max_shapelets* is the maximum number of Shapelets retained, and *time_limit* indicates the time limit of the extraction process. Generally, the more time spent, the higher the information gain of the returned Shapelets.

Random Shapelet Transform			
max_shapelets	min_shapelet_length	max_shapelet_length	time_limit=1
20	0.1 * seq_len	0.5 * seq_len	1 minute

Table 5.1.2: The selection of (hyper) parameters for TimeGAN.

TimeGAN						
epochs	batch_size	lr	noise_dim	layers_dim	hidden_dim	gamma
500	4	5e-4	1	64	24	1

Table 5.2.1: The selection of (hyper) parameters for K-nearest neighbours (KNN) classifier. We can set the distance parameter to either *Euclidean* or *DTW* (Dynamic Time Warping). However, considering the extended training time that DTW may necessitate in comparison to the Euclidean distance, we opt to perform our experiment using the KNN model with the Euclidean distance metric.

Selection of (Hyper)parameters for K-nearest neighbors (KNN)	
distance	n_neighbours
Euclidean	1

for each dataset, its corresponding experiment will be suspended if minimum changes that make the to-be-explained instance classified as other class are found.

5.2 Classifiers

To examine the performance of the counterfactual generation model M more comprehensively, four popular time series classifiers from *sktime.classification module* are utilized in the experiments. The selection of the (hyper)parameters for each classifier is shown in Tables 5.2.1, 5.2.2, 5.2.3, and 5.2.4. Not all parameter settings are listed and unless otherwise stated, those parameters that are not presented in the Tables use the *sktime* default setting. The *sktime* version used in the experiment is 0.19.1.

K-nearest neighbours (KNN) Classifier with Euclidean distance

The k-nearest neighbors (KNN) algorithm can serve as a distance-based classifier for time series data. It operates by predicting labels for given time series instances based on their similarity to the training data. As such, it is widely used as a baseline for the experimental comparison between classifiers. Similarly, in our experiment, it is regarded as the fundamental model.

Convolutional Neural Networks (CNNs) Classifier

Arguably, LeNet-5 [15] proposed by *Y LeCun et al.* is the pioneer of modern CNN models. It has gone through rapid evolution and a majority of extension of CNN architecture have been developed and applied successfully in a wide range of artificial intelligence tasks. In 2017, *Zhao et. al* proposed the paper *Convolutional neural networks for time*

Table 5.2.2: The selection of (hyper) parameters for Convolutional Neural Networks (CNNs) Classifier, where all (hyper)parameters are set to the default settings provided by the *sktime* library.

Selection of (Hyper)parameters for Convolutional Neural Networks (CNNs)			
n_epochs	batch_size	kernel_size	filter_sizes
2000	16	7	[6, 12]

Table 5.2.3: The selection of (hyper)parameters for Diverse Representation Canonical Interval Forest Classifier (DrCIF). The *n_estimators* parameter specifies the number of estimators used to ensemble the trees. The *n_intervals* parameter is the number of intervals designated for feature extraction per representation per tree. Lastly, the *att_subsample_size* parameter sets the feature sub-sample size for each tree. All (hyper)parameters are set according to the recommended settings provided by the examples in the *sktime* library.

Selection of (Hyper)parameters for Diverse Representation Canonical Interval Forest (DrCIF)		
n_estimators	n_intervals	att_subsample_size
3	2	2

Table 5.2.4: The selection of (hyper)parameters for Catch22 Classifier. The *outlier_norm* parameter, when set to *True*, means that normalization is performed when encountering outliers or extreme values. Moreover, the *n_estimators* parameter determines the number of trees in the Random Forest, and the *criterion* parameter defines the method used to measure the quality of a split. All (hyper)parameters are set according to the recommended settings provided by the examples in the *sktime* library.

Selection of (Hyper)parameters for Catch22 Classifier with Random Forest (RF)		
outlier_norm	n_estimators	criterion
True	5	gini

series classification [43], where they adapted Convolutional Neural Networks (CNNs) to time series problems, leveraging convolutional layers to extract local dependencies in the sequences.

Diverse Representation Canonical Interval Forest Classifier (DrCIF)

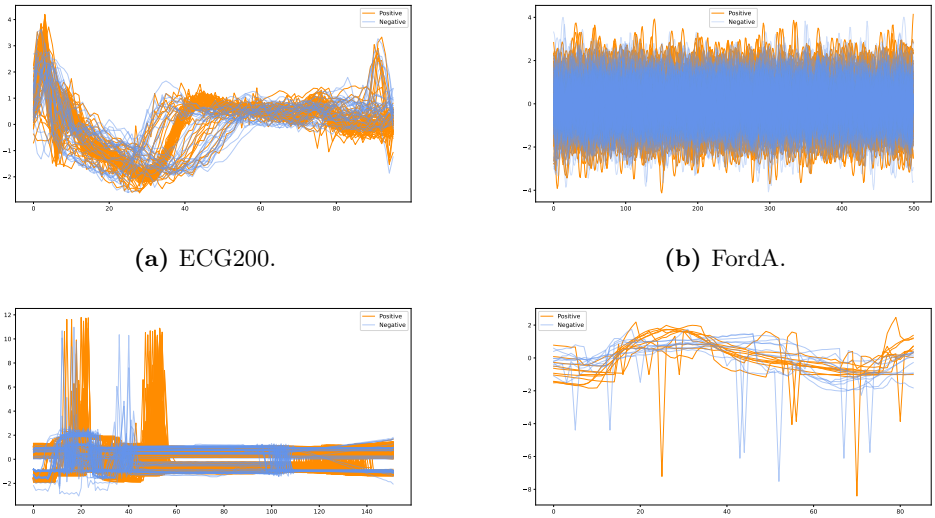
Diverse Representation Canonical Interval Forest Classifier (DrCIF) [21] belongs to the family of interval-based classifiers. It is developed atop the Canonical Interval Forest (CIF) algorithm. The main idea behind DrCIF is that it takes advantage of a series of representations for time series data, including catch22, and applies transformations to randomly selected intervals within the time-series data.

Catch22 Classifier with Random Forest (RF) Classifier

Catch22, a feature-based method, stands for Canonical Time-series Characteristics, with the number 22 indicating the number of features included in the feature set, known for its highly discriminatory power for classification. This classifier works by transforming the original time-series data with 22 features, providing highly efficient operations. The transformed data is then utilized to build an estimator, with the Random Forest (RF) Classifier being chosen.

5.3 Dataset

In terms of the experiment, four datasets are selected from the UCR archive [4]. All the datasets are binary class and have only one feature. See Table 5.3.1 for more details of the dataset structure. Figures 5.3.1a, 5.3.1b, 5.3.1c, and 5.3.1d show the distribution of the datasets, with two different classes distinguished by colour.



(a) ECG200.

(b) FordA.

(c) Wafer.

(d) MoteStrain.

Figure 5.3.1: Data distribution for training set.

Table 5.3.1: Dataset for experiment. *Length* represents the number of time-points (or time-steps) of a timer series instance, while *Classes Ratio* aims to show the number of instances for each class in training set. For example, *903:97* for dataset **Wafer** means there are 903 positive instances and 97 negative instances in it.

Dataset	Train size	Test size	Classes Ratio (Train set)	Length	Number of classes	Number of dimension
ECG200	100	100	69:31	96	2	1
FordA	3601	1320	1755:1846	500	2	1
Wafer	1000	6164	903:97	152	2	1
MoteStrain	20	1252	10:10	84	2	1

ECG200

This dataset, introduced by R. Olszewski's in his thesis *Generalized feature extraction for structural pattern recognition in time-series data* [25] focuses on the electrical activity of the heartbeat. In this context, the positive class is normal heartbeats and the negative class myocardial infarction.

Wafer

The *Wafer* dataset, also created by R. Olszewski in [25] records changes in sensor values during silicon wafer processing for semiconductor fabrication. It distinguishes between two labels: normal (positive) and abnormal (negative).Notably, this dataset is imbalanced, with the negative class constituting only 9.7% of the data. Therefore, it was selected for the experiment to evaluate its potential negative impact.

FordA

This dataset, designed for the IEEE World Congress on Computational Intelligence in 2008, consists of time series instances with 500 contiguous recordings, each corresponds to a reading of engine noise. The objectives of the classification problem is to detect outliers in automobile engine.

MoteStrain

This dataset, derived from *C. Guestrin et al.*'s paper *Online Latent Variable Detection in Sensor Networks* [36] is known as *MoteStrain*. It comprises data collected from two separate sensors: q8calibHumid and q8calibHumTemp. Each time series instance stands for a measurement of either humidity or temperature. The goal of the classification task is to determine the origin sensor of a particular data reading.

5.4 Evaluation and Result

As stated, experiments were conducted on different datasets for each classifier, with the exception of KNN. The random seeds used for these experiments were 111, 222, and 333. Tables 5.4.1, 5.4.2, 5.4.4, and 5.4.3 present the results. From these tables, it is evident that the quality of a dataset plays a crucial role in determining the performance of the counterfactual instance generation model. In the case of **ECG200**, its imbalanced class distribution of approximately 7:3 leads to outliers constituting the majority of generated counterfactual instances. For **Wafer**, which is even more extremely imbalanced, information gain (IG) struggles to extract Shapelets with discriminatory power, making it challenging for our model to generate meaningful counterfactual instances. In [40], they found that Information Gain (IG) tend not to extract Shapelets with discriminatory power in terms of imbalanced data sets. Hence, it is difficult for our generation model to produce counterfactual instances based on imbalanced dataset (ECG200 and Wafer). Moreover, the results for **MoteStrain** shows that the sparsity is guaranteed with over 90% while ensuring the average distance between the generated counterfactual instance and the original one is close. However, its small size of training data also causes problems such as overfitting and insufficient representational power. In these scenarios, Shapelet Transform is not only incapable of effectively capturing the relevant features, resulting in counterfactuals out of distribution although discovered Shapelets may fit these limited data plausibly well, but

Table 5.4.1: The results for **ECG200**. Attribute *Outliers* is the measure taken to see the percentage of out-of-distribution instances in the generated counterfactual instances (The lower, the better). However, the percentage for *Sparsity* represents the extent to which the original instance has been modified (The higher, the better).

ECG200 (Positive)						
Classifier	Accuracy	Closeness (L1)	Closeness (L2)	Sparsity	Outliers	Num CF
KNN	0.88	19.8	4.9	60%	81%	72
CNN	0.85	8.5	2.2	77%	100%	22
DrCIF	0.82	10.6	2.7	66%	44%	26
Catch22	0.80	26.8	5.4	57%	53%	12
ECG200 (Negative)						
KNN	0.88	32.3	4.4	76%	30%	72
CNN	0.85	14.1	4.5	63%	34%	134
DrCIF	0.82	10.8	2.7	73%	0%	71
Catch22	0.80	18.5	4.1	65%	40%	142

Table 5.4.2: The results for **FordA**. Attribute *Outliers* is the measure taken to see the percentage of out-of-distribution instances in the generated counterfactual instances (The lower, the better). However, the percentage for *Sparsity* represents the extent to which the original instance has been modified (The higher, the better).

FordA (Positive)						
Classifier	Accuracy	Closeness (L1)	Closeness (L2)	Sparsity	Outliers	Num CF
KNN	0.67	51.1	9.7	93%	0	155
CNN	0.88	141.1	15.0	74%	0	2
DrCIF	0.79	44.4	8.6	93%	0	890
Catch22	0.88	43.7	8.3	92%	0	148
FordA (Negative)						
KNN	0.67	13.2	6.1	99%	0	254
CNN	0.88	145.1	15.8	83%	0	175
DrCIF	0.79	127.7	15.2	79%	0	93
Catch22	0.88	83.6	13.8	90%	0	60

also outputting few number of counterfactual instances. On the other hand, generation model M demonstrates the best performance on the dataset **FordA**. Intuitively, the distribution of the two classes shown in Figure 5.3.1b is apparently distinct. The range of value for the negative instances is from -2 to 2, whereas that for the positive instances is somewhere between -4 and 4. As expected, our generation approach can provide interpretable explanations with low modifications and no outliers.

5.5 Comparative Study

In this section, we conduct a comparative analysis between our approach, **Time-CF**, and two other models: **mlxtend** (baseline model) [28] and **Native-Guide** [6]. This comparison is based on the four metrics detailed in sections 3.3 and 4.3, and we employ the same datasets, classifiers, and random seeds. Specifically, **mlxtend** [28] is an open source library in Python tailored for various data science tasks. It provides a counterfactual-based method that implements the original counterfactual generation model proposed by Wachter [39] to

Table 5.4.3: The results for **Wafer**. Attribute *Outliers* is the measure taken to see the percentage of out-of-distribution instances in the generated counterfactual instances (The lower, the better), while the percentage for *Sparsity* represents the extent to which the original instance has been modified (The higher, the better). On the other hand, attribute *Num CF* is the number of counterfactual instances generated. If it is 0, which means there is no counterfactual instance generated by generation model M then the record is set to None.

Wafer (Positive instance)						
Classifier	Accuracy	Closeness (L1)	Closeness (L2)	Sparsity	Outliers	Num CF
KNN	1	None	None	None	None	0
CNN	1	None	None	None	None	0
DrCIF	0.99	6.2	3.6	98%	0	33
Catch22	0.99	None	None	None	None	0
Wafer (Negative instance)						
KNN	1	None	None	None	None	0
CNN	0.99	None	None	None	None	0
DrCIF	0.99	12.7	4.5	95%	100%	24
Catch22	0.99	10.3	4.4	95%	100%	3

Table 5.4.4: The results for **MoteStrain**. Attribute *Outliers* is the measure taken to see the percentage of out-of-distribution instances in the generated counterfactual instances (The lower, the better). However, the percentage for *Sparsity* represents the extent to which the original instance has been modified (The higher, the better).

MoteStrain (Positive)						
Classifier	Accuracy	Closeness (L1)	Closeness (L2)	Sparsity	Outliers	Num CF
KNN	0.88	20.5	5.2	84%	48%	12
CNN	0.90	2.7	1.4	95%	0	7
DrCIF	0.74	18.5	4.7	80%	23%	3
Catch22	0.80	3.0	1.6	95%	32%	3
MoteStrain (Negative)						
KNN	0.88	1.6	0.8	94%	0	1
CNN	0.90	8.1	2.7	89%	0	4
DrCIF	0.74	9.2	4.6	87%	0	6
Catch22	0.80	6.8	2.9	88%	0	2

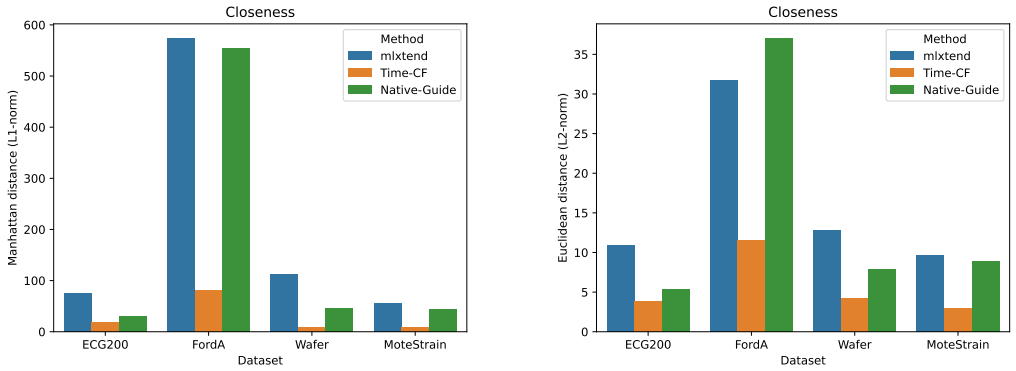


Figure 5.5.1: Closeness measures, where the results for L1-norm and L2-norm are shown in the left and right sub-figures, respectively.

explain a given instance. For **mlxtend**, We vary the parameter λ (0.4, 0.5, 1.0, 5.0, and 100) to probe into different aspects of the counterfactual instances. Additionally, **Native-Guide**, as referred to in chapter 2, deploys class activation weights (CAM) and weighted dynamic barycentre averaging (DBA) as two distinct methods to explain classifiers. However, since CAM can only be utilized with deep learning models that necessitate the addition of a global average pooling layer, we resort to their alternate method that employs DBA in our experiment, using its default parameter settings.

5.5.1 Closeness

As illustrated in figure 5.5.1, the results highlights closeness metrics. As anticipated, **mlxtend** as the baseline model showcases the least impressive performance. Conversely, counterfactual instances produced by our approach boast the smallest distance to the instances being explained across all datasets. This reveals our approach's proficiency in perturbing only the minimal, contiguous portion of the original instances.

5.5.2 Omission

Omission serves as a metric to determine the extent to which counterfactual-based methodologies are model-agnostic or can explain any classifier. Evaluating four time series classifiers across four datasets, we documented the proportion of classifiers that remained unexplained, signifying that the counterfactual-based approaches could not yield any counterfactual instances for a given to-be-explained instance. The data in figure 5.5.2 demonstrates that both **mlxtend** and **Native-Guide** fail to effectively handle a variety of classifiers across these datasets. In contrast, with the exception of the imbalanced dataset *Wafer*, our method can aptly discerns the informative features in time series data, thereby positioning itself as a model-agnostic tool that can, in theory, clarify any classifier.

5.5.3 Plausibility

In this experiment, we delve deeply into the percentage of generated instances identified as out of distribution. Given our emphasis on the importance of plausibility when generating counterfactual instances, we exclusively retain those counterfactuals with the lowest outlier rates to ensure heightened plausibility. As depicted in figure 5.5.3, our method outperforms

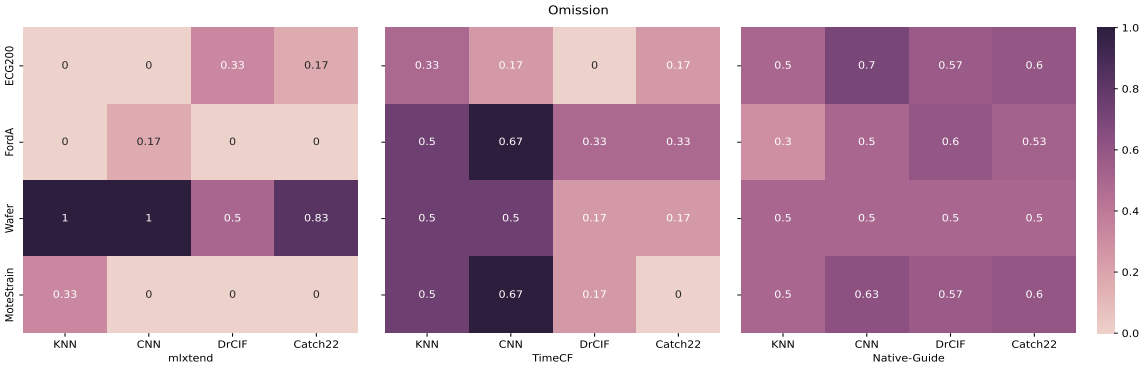


Figure 5.5.2: Assessment of plausibility (A lower value is preferable). Y-axis quantifies the percentage of generated counterfactual instances considered as out-of-distribution for datasets, where a value of 0 indicates complete adherence of all generated instances to the characteristics of their corresponding dataset.

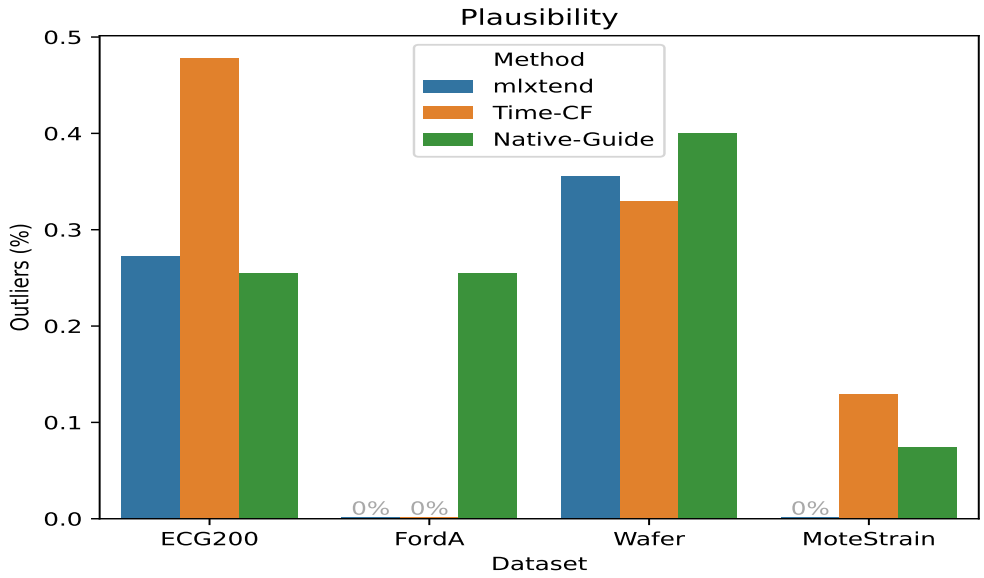


Figure 5.5.3: Assessment of plausibility (A lower value is preferable). Y-axis quantifies the percentage of generated counterfactual instances considered as out-of-distribution for datasets, where a value of 0 indicates complete adherence of all generated instances to the characteristics of their corresponding dataset.

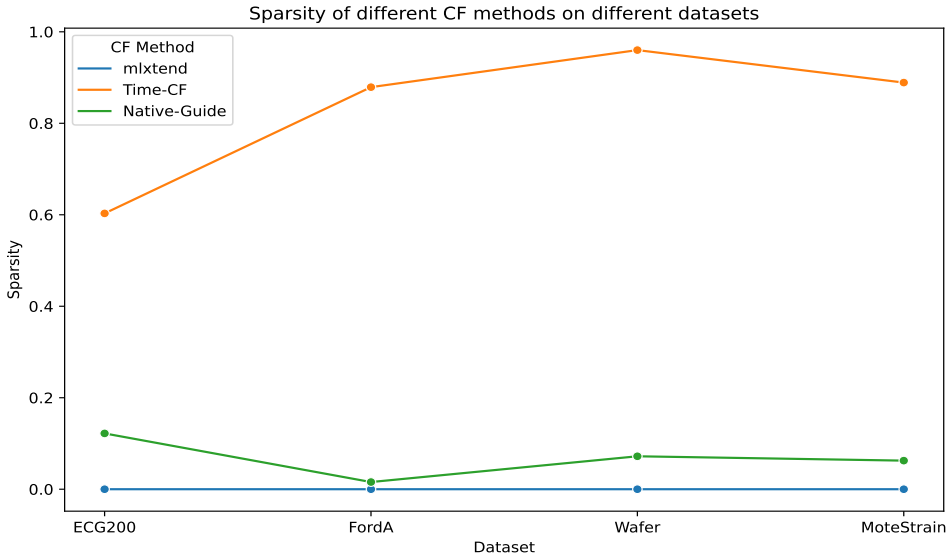


Figure 5.5.4: Measurement for sparsity (The higher, the better). This metric indicates the extent to which the number of time-steps is altered. A value of 0 means that the counterfactual instances are derived from the original instances that have alterations at every time-points.

others in terms of plausibility in dataset **FordA**. However, it shows worse performance than the others when using an imbalanced dataset.

5.5.4 Sparsity

Lastly, curve graph in figure 5.5.4 contrasts the sparsity level of each CF methods. When compared to the sparsity levels of **mlxtend** (blue) and **Native-Guide** (green), that of our approach consistently showcases superior performance. It is evident that both **mlxtend** and **Native-Guide** require modifications to nearly all the time-steps to alter the class of the to-be-explained instance from one to another, with values approaching a sparsity of 0%. In contrast, our approach only perturb a minor segment, thereby offering not only clearer visualization for analytical objectives but also a more intuitive comprehension of which portions primarily influence the classifier.

6

Conclusion

In this work, a model-agnostic, counterfactual-based XAI method, **Time-CF** was introduced to explain any anomaly for any classifiers concerning time series classification tasks. **Time-CF** chiefly utilizes Shapelets and TimeGAN to deliver meaningful counterfactual explanations, exhibiting exceptional performance in terms of closeness, omission, plausibility, and sparsity measures. The incorporation of Shapelets indicates that a contiguous segment of the time series instance is perturbed, thus offering intuitive insights for the analysis of anomalies for end-users. In addition, TimeGAN ensures that at least one counterfactual explanation is generated for explanation, provided the training set is both ample and balanced. TimeGAN also guarantees that the produced series adheres to the data distribution.

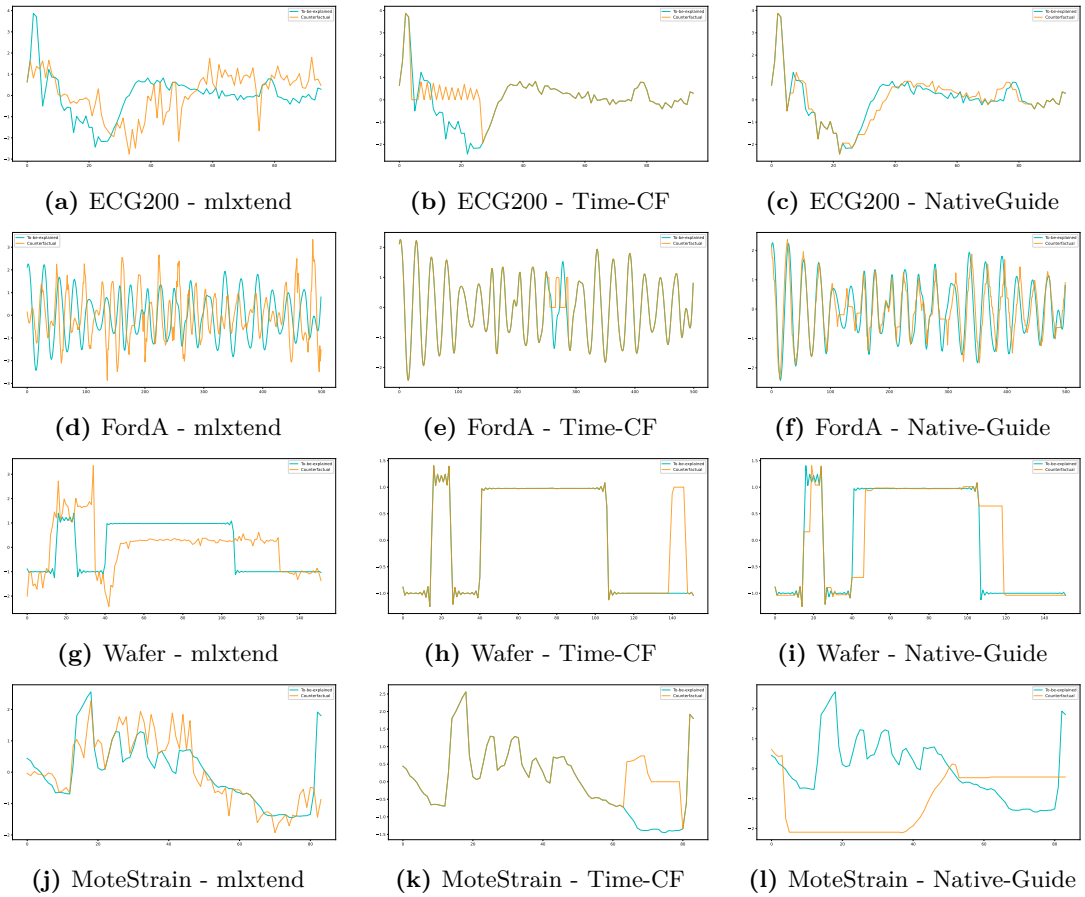
In future work, we aim to refine our methodology to cater to multi-class and multivariate time series classification tasks. Moreover, a more profound exploration of the synergies between TimeGAN and Shapelet Transform (ST) will be undertaken to address the suboptimal performance observed with imbalanced datasets.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. N. van Stein, as well as her PhD candidate, Qi Huang, for their relentless support and guidance during my research endeavors. Lastly, a special thanks to my girlfriend, Sichong, who has always been there for me. I love you.

A

Appendix



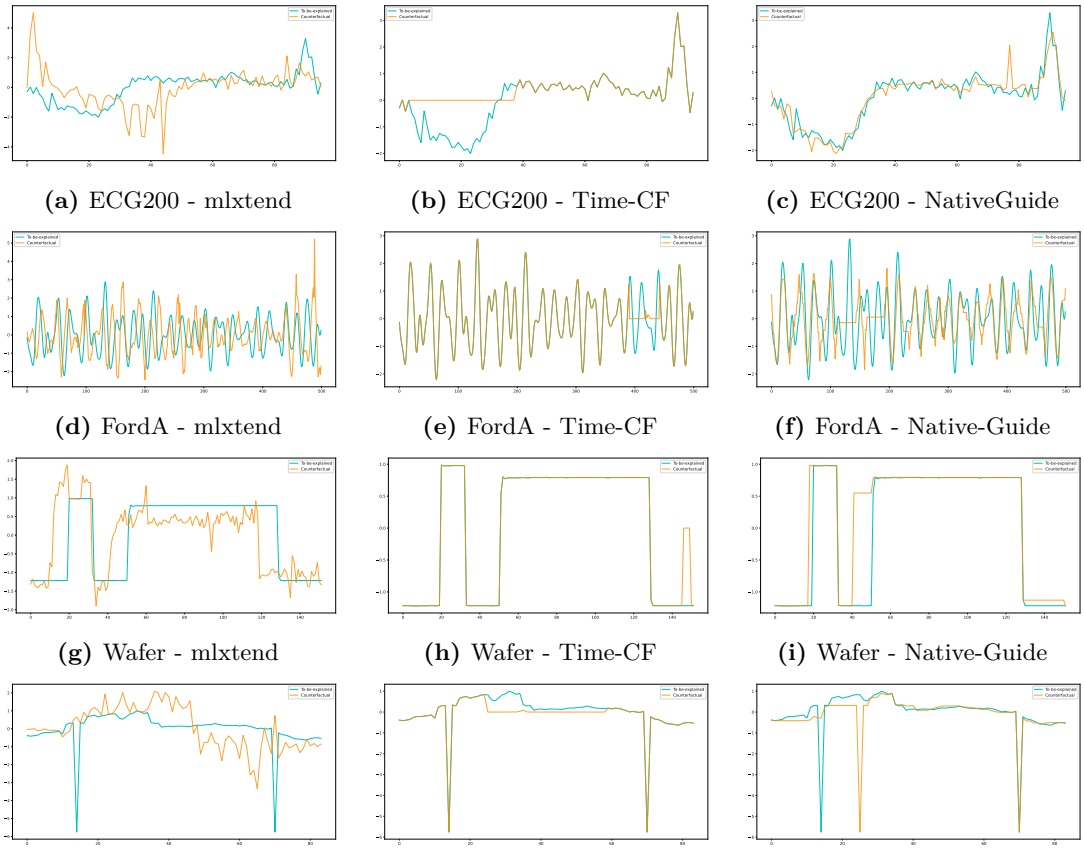
(a) ECG200 - mlxtend **(b)** ECG200 - Time-CF **(c)** ECG200 - NativeGuide

(d) FordA - mlxtend **(e)** FordA - Time-CF **(f)** FordA - Native-Guide

(g) Wafer - mlxtend **(h)** Wafer - Time-CF **(i)** Wafer - Native-Guide

(j) MoteStrain - mlxtend **(k)** MoteStrain - Time-CF **(l)** MoteStrain - Native-Guide

Figure A.0.1: Counterfactual instances generated using different methods, where the to-be-explained instance is labeled as positive.



(a) ECG200 - mlxtend **(b)** ECG200 - Time-CF **(c)** ECG200 - NativeGuide

(d) FordA - mlxtend **(e)** FordA - Time-CF **(f)** FordA - Native-Guide

(g) Wafer - mlxtend **(h)** Wafer - Time-CF **(i)** Wafer - Native-Guide

(j) MoteStrain - mlxtend **(k)** MoteStrain - Time-CF **(l)** MoteStrain - Native-Guide

Figure A.0.2: Counterfactual instances generated using different methods, where the to-be-explained instance is labeled as negative.

Bibliography

[1] Emre Ates, Burak Aksar, Vitus J Leung, and Ayse K Coskun. Counterfactual explanations for multivariate time series. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–8. IEEE, 2021.

[2] Omar Bahri, Soukaina Filali Boubrahimi, and Shah Muhammad Hamdi. Shapelet-based counterfactual explanations for multivariate time series. *arXiv preprint arXiv:2208.10462*, 2022.

[3] João Bento, Pedro Saleiro, André F Cruz, Mário AT Figueiredo, and Pedro Bizarro. Timeshap: Explaining recurrent models through sequence perturbations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2565–2573, 2021.

[4] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[5] Cassio F Dantas, Diego Marcos, and Dino Ienco. Counterfactual explanations for land cover mapping in a multi-class setting. *arXiv preprint arXiv:2301.01520*, 2023.

[6] Eoin Delaney, Derek Greene, and Mark T Keane. Instance-based counterfactual explanations for time series classification. In *Case-Based Reasoning Research and Development: 29th International Conference, ICCBR 2021, Salamanca, Spain, September 13–16, 2021, Proceedings 29*, pages 32–47. Springer, 2021.

[7] Eoin Delaney, Derek Greene, and Mark T Keane. Uncertainty estimation and out-of-distribution detection for counterfactual explanations: Pitfalls and solutions. *arXiv preprint arXiv:2107.09734*, 2021.

[8] Soukaïna Filali Boubrahimi and Shah Muhammad Hamdi. On the mining of time series data counterfactual explanations using barycenters. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3943–3947, 2022.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[10] Maël Guillemé, Véronique Masson, Laurence Rozé, and Alexandre Termier. Agnostic local explanation for time series classification. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 432–439. IEEE, 2019.

[11] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data mining and knowledge discovery*, 28:851–881, 2014.

[12] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *International Conference on Artificial Intelligence and Statistics*, pages 895–905. PMLR, 2020.

[13] Mark T Keane and Barry Smyth. Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In *Case-Based Reasoning Research and Development: 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8–12, 2020, Proceedings 28*, pages 163–178. Springer, 2020.

[14] Jana Lang, Martin Giese, Winfried Ilg, and Sebastian Otte. Generating sparse counterfactual explanations for multivariate time series. *arXiv preprint arXiv:2206.00931*, 2022.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Peiyu Li, Omar Bahri, Soukaïna Filali Boubrahimi, and Shah Muhammad Hamdi. Sg-cf: Shapelet-guided counterfactual explanation for time series classification. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 1564–1569. IEEE, 2022.

[17] Peiyu Li, Soukaina Filali Boubrahimi, and Shah Muhammad Hamd. Motif-guided time series counterfactual explanations. *arXiv preprint arXiv:2211.04411*, 2022.

[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[19] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J Király. sktime: A Unified Interface for Machine Learning with Time Series. In *Workshop on Systems for ML at NeurIPS 2019*.

[20] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[21] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. Hive-cote 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(11-12):3211–3243, 2021.

[22] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[23] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[24] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617, 2020.

[25] Robert Thomas Olszewski. *Generalized feature extraction for structural pattern recognition in time-series data*. Carnegie Mellon University, 2001.

[26] Ricardo Pereira. YData Synthetic: A package to generate synthetic tabular and time-series data, 2023. URL `https://github.com/ydataai/ydata-synthetic`. GitHub repository.

[27] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Feasible and actionable counterfactual explanations. 2020.

[28] Sebastian Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24), April 2018. doi: 10.21105/joss.00638. URL `https://joss.theoj.org/papers/10.21105/joss.00638`.

[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[30] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5): 206–215, 2019.

[31] Iqbal H Sarker. Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6):420, 2021.

[32] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.

[33] Udo Schlegel, Hiba Arnout, Mennatallah El-Assady, Daniela Oelke, and Daniel A Keim. Towards a rigorous evaluation of xai methods on time series. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4197–4201. IEEE, 2019.

[34] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[35] Torty Sivill and Peter Flach. Limesegment: Meaningful, realistic time series explanations. In *International Conference on Artificial Intelligence and Statistics*, pages 3418–3433. PMLR, 2022.

[36] Jimeng Sun, Spiros Papadimitriou, and Christos Faloutsos. Online latent variable detection in sensor networks. In *21st International Conference on Data Engineering (ICDE'05)*, pages 1126–1127. IEEE, 2005.

[37] Andreas Theissler, Francesco Spinnato, Udo Schlegel, and Riccardo Guidotti. Explainable ai for time series classification: A review, taxonomy and research directions. *IEEE Access*, 2022.

[38] Arnaud Van Looveren, Janis Klaise, Giovanni Vacanti, and Oliver Cobb. Conditional generative models for counterfactual explanations. *arXiv preprint arXiv:2101.10123*, 2021.

[39] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[40] Qiuyan Yan and Yang Cao. Optimizing shapelets quality measure for imbalanced time series classification. *Applied Intelligence*, 50:519–536, 2020.

[41] Wenzhuo Yang, Jia Li, Caiming Xiong, and Steven CH Hoi. Mace: An efficient model-agnostic framework for counterfactual explanation. *arXiv preprint arXiv:2205.15540*, 2022.

[42] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

[43] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.

[44] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[45] Linjiang Zhou, Chao Ma, Xiaochuan Shi, Dian Zhang, Wei Li, and Libing Wu. Saliencecam: Visual explanations from convolutional neural networks via salience score. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.