TX-Gen: multi-objective optimization for sparse counterfactual explanations for time-series classification

Huang, Q.; Kitharidis, S.; Bäck, T.H.W.; Stein, N. van; Marcelloni, F.; Madani, K.; Filipe, J.

TX-Gen: Multi-Objective Optimization for Sparse Counterfactual Explanations for Time-Series Classification

Qi Huang[1], Sofoklis Kitharidis[1], Thomas Bäck[1] and Niki van Stein[1]
[1]*Institute of Advanced Computer Science, Leiden University, Einsteinweg 55, Leiden, The Netherlands*
{*q.huang, s.kitharidis, t.h.w.baeck, n.van.stein*}*@liacs.leidenuniv.nl*

Keywords: Explainable Artificial Intelligence, Counterfactuals, Time-series Classification, Evolutionary Computation

Abstract: In time-series classification, understanding model decisions is crucial for their application in high-stakes domains such as healthcare and finance. Counterfactual explanations, which provide insights by presenting alternative inputs that change model predictions, offer a promising solution. However, existing methods for generating counterfactual explanations for time-series data often struggle with balancing key objectives like proximity, sparsity, and validity. In this paper, we introduce TX-Gen, a novel algorithm for generating counterfactual explanations based on the Non-dominated Sorting Genetic Algorithm II (NSGA-II). TX-Gen leverages evolutionary multi-objective optimization to find a diverse set of counterfactuals that are both sparse and valid, while maintaining minimal dissimilarity to the original time series. By incorporating a flexible reference-guided mechanism, our method improves the plausibility and interpretability of the counterfactuals without relying on predefined assumptions. Extensive experiments on benchmark datasets demonstrate that TX-Gen outperforms existing methods in generating high-quality counterfactuals, making time-series models more transparent and interpretable.

1 INTRODUCTION

The increasing adoption of machine learning models for time-series classification (TSC) in critical domains such as healthcare (Morid et al., 2023) and finance (Chen et al., 2016) has raised the demand for interpretable and transparent decision-making processes. However, the black-box nature of many high-performing classifiers, such as neural networks or ensemble models, limits their interpretability, making it difficult for practitioners to understand why a particular decision is made. In this context, counterfactual explanations have emerged as a valuable approach in Explainable AI (XAI) (Theissler et al., 2022), providing instance-specific insights by identifying alternative inputs that lead to different classification outcomes. Despite the growing body of work in this area, generating meaningful counterfactuals for time-series data presents unique challenges due to its sequential nature, dependency on temporal structure, and multi-dimensional complexity.

While counterfactual generation has been extensively explored for tabular and image data, methods tailored to time-series classification remain scarce and underdeveloped. The few existing methods often fail to balance key properties such as proximity, sparsity, and validity. Moreover, most approaches require high computational resources or rely on rigid heuristics, limiting their applicability in real-world scenarios. This gap motivates the development of an efficient, model-agnostic method capable of generating high-quality counterfactual explanations for time-series classifiers.

In this paper, we propose TX-Gen, a novel algorithm for generating counterfactual explanations in time-series classification tasks using a modified Non-dominated Sorting Genetic Algorithm II (NSGA-II). Unlike previous approaches that combine evolutionary computing with explainable AI (Zhou et al., 2024), TX-Gen leverages the power of evolutionary multi-objective optimization to find a diverse set of Pareto-optimal counterfactual solutions that simultaneously minimize multiple objectives, such as dissimilarity to the original time-series and sparsity of changes, while ensuring the classifier's decision is altered. Additionally, our approach incorporates a flexible reference-based mechanism, which guides the counterfactual generation process without relying on restrictive assumptions or predefined shapelets.

Contributions The key contributions of this work are as follows:

- We introduce TX-Gen, a novel framework for generating counterfactual explanations specifically tailored to time-series classification, which efficiently balances proximity, sparsity, and validity through multi-objective optimization.
- Our method employs a reference-guided approach to select informative subsequences, improving the plausibility and interpretability of counterfactuals.
- We demonstrate the effectiveness of TX-Gen across multiple benchmark datasets, showing that our approach outperforms existing methods in terms of sparsity, validity and proximity of the counterfactual examples.

In the following sections, we detail the methodology behind TX-Gen, present our experimental results, and discuss the implications of our findings for advancing explainability in time-series classification.

1.1 Problem Statement

We begin our problem formulation by first introducing the context of this research.

Given a collection (set) of univariate time series, denoted as $S = \{T_1, T_2, \ldots, T_n\}$, where each $T = \{t_i \in \mathbb{R}\}_{i=1}^{m}$ consists of observations orderly recorded across m timestamps. In the context of a time series classification (TSC) task, each time series T is associated with a true descriptive label $L \in \{L_1, \ldots, L_k\}$ from k unique categories. The objective of TSC is to develop an algorithmic predictor $f(\cdot)$ that can maximize the probability $P(f(T) = L)$ for $T \in S$ and also any unseen $T \notin S$ (provided that the true label of such a $T \notin S$ also equals L). Here, we presume the classifier provides probabilistic or logit outputs. Moreover, assume a function $g : \mathbb{R}^m \to \mathbb{R}^m$ that can element-wise perturb or transform any given time series T into a new *in-distribution* time series instance $\hat{T}$. We then say that $\hat{T}$ is a counterfactual of T regarding a TSC predictor f if $f(\hat{T}) \neq f(T)$ while the dissimilarity between T and $\hat{T}$ is *minimized* (Wachter et al., 2017; Delaney et al., 2021). It is noteworthy that counterfactual explanations are fundamentally instance-based explanations aimed at interpreting the predictions of the classifier being explained, rather than the true labels, regardless of whether the real labels are known or not.

Goal Considering a robust time series classification predictor f trained on the dataset S, and a time series instance T_e to be explained, this work proposes an explainable AI method, i.e., the aforementioned function g, which can identify the counterfactual of T_e concerning f. Notably, T_e may be either part of the dataset S or external to it.

2 BACKGROUND

2.1 Evaluation Metrics

The evaluation criteria for our counterfactual generation method are articulated through several key metrics, each addressing specific aspects of counterfactual quality and effectiveness:

- **Proximity**: Measures the element-wise similarity between the counterfactual instance and the target time series, typically by using the L_p metrics.
- **Validity**: Assesses whether the counterfactuals successfully alter the original class label, thereby reflecting their effectiveness.
- **Diversity**: Measures the variety in the generated counterfactual solutions for each to-be-explained time series instance.
- **Sparsity**: Quantifies the simplicity of the counterfactual by counting the number of element-wise differences between the generated counterfactual and the original series.

These metrics collectively provide a robust framework for evaluating the quality and utility of counterfactual explanations, emphasizing minimalism, diversity, validity, and closeness to the original instances.

2.2 Related Work

The evolution of explainable artificial intelligence (XAI) for generating counterfactuals for time series has been marked by significant advances in methods that provide clear, actionable insights into model decisions. This line of research has been initiated by (Wachter et al., 2017), introducing an optimization-based approach that focuses on minimizing a loss function to modify decision outcomes while maintaining minimal Manhattan distance from the original input. This foundational method encouraged further development of algorithms that enhance the quality and effectiveness of counterfactual explanations through additional modifications to the loss function.

Building upon the concept of influencing specific features within time series data, the introduction of local tweaking via a Random Shapelet Forest classifier (Karlsson et al., 2020) targets specific, influential shapelets for localized transformations within time series data. In contrast, global tweaking (Karlsson et al., 2018) uses a k-nearest neighbor classifier to guide broader transformations ensuring minimal yet meaningful alterations to change class outcomes.

Advancing these concepts, (Delaney et al., 2021) developed the Native Guide Counterfactual Explanation (NG-CF) which utilizes the Dynamic Barycenter Averaging with the nearest unlike neighbor. Subsequently, (Li et al., 2022) introduced the Shapelet-Guided Counterfactual Explanation (SG-CF), exploiting time series subsequences that are maximally representative of a class to guide the perturbations needed for generating counterfactual explanations. This method was further elaborated upon with the introduction of the Attention-Based Counterfactual Explanation for Multivariate Time Series (AB-CF) by (Li et al., 2023) which incorporates attention mechanisms to further refine the selection of shapelets and optimize the perturbations.

Moreover, TSEvo by (Höllig et al., 2022) utilizes the Non-Dominated Sorting Genetic Algorithm to strategically balance multiple explanation objectives such as proximity, sparsity, and plausibility while incorporating three distinct mutations. Most recently, Sub-SpaCE (Refoyo and Luengo, 2024) employs genetic algorithms too, with customized mutation and initialization processes, promoting changes in only a select few subsequences. This method focuses on generating counterfactual explanations that are both sparse and plausible without extensive alterations.

Despite the progress in generating counterfactual explanations for time-series classification, these existing approaches still face several significant limitations. Many methods, such as optimization-based or shapelet-guided approaches, rely heavily on predefined assumptions about the data, such as the importance of specific subsequences or the necessity for local perturbations. These assumptions often reduce the generalizability of the methods across diverse datasets and classifiers. Furthermore, most methods struggle to balance the trade-offs between proximity, sparsity, and diversity of the generated counterfactuals, often prioritizing one metric at the expense of others. This can lead to counterfactuals that are either too similar to the original instance to be meaningful or excessively altered, making them unrealistic or hard to interpret. Additionally, many existing approaches lack scalability, becoming computationally expensive when applied to larger or more complex time-series data. In contrast, our proposed method, TX-Gen, addresses these limitations by leveraging the flexibility of evolutionary multi-objective optimization to dynamically explore the solution space and by guiding the search using reference samples from the training data. This allows TX-Gen to generate diverse, sparse, and valid counterfactuals with greater computational efficiency, making it more applicable to real-world time-series classification tasks.

3 METHODOLOGY

The framework Our proposed method, TX-Gen, aims to jointly locate the subsequence of interest in a to-be-explained time series T while also transforming T into its counterfactual. This is achieved by mutually minimizing two objective functions under the guidance of reference samples, using a customized iterative optimization heuristic based on the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002). A high-level pseudocode of our algorithm is given in Algorithm 1 assuming the respect readers are familiar with the concepts of evolutionary algorithms (Back et al., 1997; Bäck et al., 2023) Following the self-explainable high-level pseudocode, the key components proposed in our algorithm are further explained.

3.1 Model-based Selection of References

In TSC, consider a time series instance T that belongs to category C. A counterfactual-based explainer can be viewed as a reference-guided method if it leverages example instances—originating from the same task but known to belong to categories other than C —to assist in *transforming* T into its counterfactual. Existing literature on reference-guided methods primarily identifies the reference instances by selecting those with low shape-based or distance-based similarities to the given instance to be explained, e.g., the nearest unlike neighbors as seen in (Delaney et al., 2021; Höllig et al., 2022). In contrast, our work proposes selecting reference samples solely based on the classifier's outputs.

Following the settings in Section 1.1, let $f(\cdot)$ be a probabilistic classifier for a TSC task Ω with k classes. That is, instead of predicting the exact label, f predicts the probability distribution over all candidate class labels for any given target time series T.

Definition 1 (Distance in the Classifier)**.** *For any pairs of time series (T_i, T_j) of the TSC task Ω, we define their Distance in the Classifier f as the Jensen-Shannon distance between $f(T_i)$ and $f(T_j)$:*

$$D_f(T_i, T_j) = \sqrt{\frac{KL(f(T_i) \mid P_m) + KL(f(T_j) \mid P_m)}{2}}, \tag{1}$$

where P_m is the element-wise mean of $f(T_i)$ and $f(T_j)$, and KL denotes the well-known Kullback-Leibler (KL) divergence (Endres and Schindelin, 2003). Apart from the properties that a valid metric holds, this distance is also bounded, $D_f(T_i, T_j) \in [0, 1]$ if we use 2 as the logarithm base in KL divergence.

With this definition, as described in Algorithm 2,

Algorithm 1: High-Level pseudocode for our counterfactual discovery algorithm TX-Gen

Data: Population size N; crossover probability p_c; mutation probability p_m, number of generations G; to-be-explained time series T; classifier f; a set of references S; number of final reference instances K
Result: Set of non-dominated counterfactual candidate solutions

Initialize population P_0 of size N as described in section 3.3.1
▷ Selection of references using Algorithm 2
$\psi \leftarrow$ `SelectReference`(T, S, f, K)
Generate the counterfactual candidate sets $\Theta_0 = \bigcup_{i=1}^{N} \{\hat{T}_i\}$ for all individuals in P_0 using the `Generate` function from Algorithm 5.
Evaluate the fitness of each individual in P_0 (Θ_0) using two objective functions (see section 3.2)
for $t = 1$ *to* G **do**
  Generate offspring Q_t from P_{t-1} by:
  - Select the parents for mating using binary tournament selection
  - `Crossover` the parents (with probability p_c) to obtain Q'_t using Algorithm 3
  - `Mutate` the individuals in Q'_t in place (with probability p_m) using Algorithm 4
  - Enumeratively expand each $\overrightarrow{X_i} \in Q'_t$ into K chromosomes and merge them into a new set

$$Q_t = \bigcup_{i=1}^{2N} \bigcup_{j=1}^{K} \overrightarrow{X_{i,j}},$$

  where each $\overrightarrow{X_{i,j}} = (x_{i,1}, x_{i,2}, j)$
  Obtain the counterfactual candidates Θ_t for all samples in Q_t using `Generate` function
  Evaluate the fitness of each candidate in Θ_t and assign the resulting values to respective individuals in Q_t
  Combine parent population P_{t-1} and offspring population Q_t into a combined population R_t of size $(2K+1)N$
  Perform non-dominated sorting on R_t and assign crowding distances to individuals within each front
  Select individuals for the next population P_t based on rank and crowding distance, ensuring the population size is N
end
Return the non-dominated individuals as well as their corresponding counterfactual candidates from the final population P_G

we can retrieve reference instances for a to-be-explained time series T from a candidate set that contains samples that both are predicted to belong to different categories than T and are close to T with respect to the distance in the classifier.

Algorithm 2: Select References for a Time Series.
Input: Target time series T; the classifier f; reference set S; number of reference samples K
Output: A set of reference samples $\psi \subseteq S$.

Function `SelectReference`(T, S, f, K)**:**
  foreach $T_i \in S$ **do**
    if $f(T_i) \neq f(T)$ **then**
      Compute $D_f(T, T_i)$ as in Definition 1
    else
      Set $D_f(T, T_i) \leftarrow 1.01$
    end
  end
  Get the set: $\psi \leftarrow \arg\min_{\substack{T_i \in S \\ |\psi|=K}} D_f(T, T_i)$
return ψ

3.2 The Objective Functions

To address the idea (see section 1.1) that a decent counterfactual $\hat{T} = \{\hat{t}_i \in \mathbb{R}\}_{i=1}^{m}$ shall closely resemble the target time series $T = \{t_i \in \mathbb{R}\}_{i=1}^{m}$ while flipping the prediction of a given classifier f, we propose to determine feasible candidates $\hat{T}^*$ by jointly **minimizing** two real-valued objectives as follow:

- The minimum Distance in Classifier between a counterfactual candidate $\hat{T}^*$ and the samples in the reference set ψ:

$$F_1(\hat{T}^*, \psi) = \begin{cases} \min D_f(\hat{T}^*, T_i),\ \forall T_i \in \psi & \text{if } f(\hat{T}^*) \neq f(T), \\ 1.01 & \text{if } f(\hat{T}^*) = f(T). \end{cases}$$

  The second condition is to penalize the case where $\hat{T}^*$ fails to flip the prediction of the classifier.

- Joint sparsity and proximity between $\hat{T}^*$ and T:

$$F_{2,1}(\hat{T}^*, T) = \frac{\sum_{i=[1,\ldots,m]} \mathbf{1}_{\hat{t}_i \neq t_i}}{m},$$

$$F_{2,2}(\hat{T}^*, T) = \frac{|\hat{T}^* - T|_2}{|\hat{T}^*|_2 + |T|_2},$$

$$F_2(\hat{T}^*, T) = \frac{1}{2}(F_{2,1}(\hat{T}^*, T) + F_{2,2}(\hat{T}^*, T))$$

  Here we normalize and combine the two indicators into one objective (F_2), where the sparsity ($F_{2,1}$) measures the degree of element-wise perturbation, and the second indicator $F_{2,2}$ aims to quantify the scale change between $\hat{T}^*$ and T.

In summary, the two objective functions can jointly ensure the found counterfactual solutions are valid and minimized. What is more, the proposed two objective functions are both bounded, saying $F_1 \in [0, 1]$ and $F_2 \in [0, 1]$.

3.3 Locating the Sequences of Interest

3.3.1 Chromosome Representation

Previous works that utilize evolutionary computation predominantly adopt a unified chromosome representation that encodes the entire counterfactual instance. In other words, the number of variables in each solution is at least as large as the number of timestamps in the target time series. Furthermore, these approaches typically apply crossover, mutation, and other evolutionary operators directly to the chromosomes to generate the next set of feasible candidates in one-go (Höllig et al., 2022; Refoyo and Luengo, 2024). One advantage of encoding the entire time series into the individual is that it can find multiple **Segment or Subsequence of Interest (SoI)** at the same time.

In contrast, our approach reduces the complexity of candidate solutions by representing each one with only three mutable integer variables, i.e., $X = [x_1, x_2, x_3]$, regardless of the number of timestamps in the target time series. Specifically, x_1 and x_2 denote the starting and ending indices of the single subsequence of interest considered in this individual, respectively. Meanwhile, x_3 represents the index of the reference sample in the set ψ, which is used to guide the discovery of potential counterfactual observations for the SoI (see section 3.4).

Similar to previous approaches, our method can also identify multiple SoI for each target time series by generating various counterfactual solutions. Furthermore, by leveraging the concept of Pareto efficiency in multi-objective optimization, our approach ensures that the diverse SoI discovered are all equally significant.

Initialization Several existing works explicitly incorporate low-level XAI strategies to identify the SoI in the target instance, thereby facilitating the counterfactual search process. These strategies include feature attribution methods, such as the GradCAM family (Selvaraju et al., 2017), which are employed in works like (Delaney et al., 2021; Refoyo and Luengo, 2024). Additionally, subsequence mining methods, such as Shapelet extraction (Ye and Keogh, 2009), are utilized in studies like (Li et al., 2022; Huang et al., 2024). Unlike other XAI techniques, we do not use this (possibly biased) initialization, instead the first population in our search algorithm is randomly sampled. Given a to-be-explained time series $T = \{t_i \in \mathbb{R}\}_{i=1}^{m}$ and its reference sample set ψ of size K, the variables in a individual solution $\overrightarrow{X_j} = [x_{j,1}, x_{j,2}, x_{j,3}]$ are step-by-step uniformly randomly sampled as:

$$x_{j,1} \sim \mathbb{U}(1, m-1)$$
$$x_{j,2} \sim \mathbb{U}(x_{j,1}, m)$$
$$x_{j,3} \sim \mathbb{U}(1, K)$$

3.3.2 Crossover

Crossover is a crucial operator in evolutionary algorithms, designed to create new offspring by combining decision variables from parent solutions. This process helps the algorithm exploit the search space within the current population. The parent solutions are chosen through a *selection* mechanism. In this work, we employ the well-known tournament selection method to select mated parents, and we refer interested readers to the literature (Goldberg and Deb, 1991) for more details.

Bearing in mind the idea behind crossover, our customized crossover operator is dedicated to fulfilling two additional criteria: (i) minimizing the intersection between the SoI of offspring. (ii) minimizing the total lengths of the SoI tracked by offspring.

The proposed crossover is demonstrated in Algorithm 3. The crossover operation by default produces two offspring. The function `GetUnique` returns the unique, non-duplicate values of its inputs in an array, sorted in ascending order. Notably, in line 22 of the pseudocode, we compare the total lengths of the SoI under two feasible crossover options and intentionally set the relational operator to $\leq$ instead of $<$. This ensures that, in cases where the SoIs of two parents do not overlap, their offspring will have minimal intersections. Between lines 33 and 40, we address the corner cases where the indices recorded in the parents are partially or entirely the same.

3.3.3 Mutation

In general, the mutation operators help to explore unforeseen areas of the solution space and prevent premature convergence. Unlike the crossover operator that mixes the SoIs from different parent solutions, the mutation operator proposed in this work aims to adjust the length of the SoI tracked by each individual selected for mutation. The pseudocode of our mutation strategy is provided in Algorithm 4.

The strategy employs a dynamic scaling of the SoI length for the new instance $\overrightarrow{Y}$ based on the SoI length of the previous instance $\overrightarrow{X}$. The new length is randomly sampled from a binomial distribution $\mathbf{Bin}(n, p)$, where the number of trials n is twice the length of the SoI in $\overrightarrow{X}$. This approach allows the SoI length to potentially increase almost exponentially between adjacent generations during optimiza-

Algorithm 3: The Crossover Operator.

Input: The parent individuals $\overrightarrow{X_1} = (x_{1,1}, x_{1,2}, x_{1,3})$ and $\overrightarrow{X_2} = (x_{2,1}, x_{2,2}, x_{2,3})$; crossover probability P_{cx}.

Output: The two offspring $\overrightarrow{Y_1} = (y_{1,1}, y_{1,2}, y_{1,3})$ and $\overrightarrow{Y_2} = (y_{2,1}, y_{2,2}, y_{2,3})$.

1 **Function** `GetUnique`(a_1, a_2, a_3, a_4)**:**
2 $\quad b_1, b_2, b_3, b_4 \leftarrow \text{AscendingSort}(a_1, a_2, a_3, a_4)$
3 $\quad \overrightarrow{\beta} \leftarrow (b_1)$
4 $\quad$ **if** $b2 \neq b1$ **then**
5 $\quad\quad \overrightarrow{\beta} \leftarrow \overrightarrow{\beta} \parallel (b_2)$ ▷ Concatenate b_2 to $\overrightarrow{\beta}$
6 $\quad$ **end**
7 $\quad$ **if** $b3 \neq b2$ **then**
8 $\quad\quad \overrightarrow{\beta} \leftarrow \overrightarrow{\beta} \parallel (b_3)$
9 $\quad$ **end**
10 $\quad$ **if** $b4 \neq b3$ **then**
11 $\quad\quad \overrightarrow{\beta} \leftarrow \overrightarrow{\beta} \parallel (b_4)$
12 $\quad$ **end**
13 **return** $\overrightarrow{\beta}$
14 **Function** `Crossover`$(\overrightarrow{X_1}, \overrightarrow{X_2}, P_{cx})$**:**
15 $\quad p_c \sim \mathbb{U}(0,1)$
16 $\quad \overrightarrow{\alpha} \leftarrow \texttt{GetUnique}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2})$
17 $\quad \overrightarrow{Y_1}, \overrightarrow{Y_2} \leftarrow \overrightarrow{X_1}, \overrightarrow{X_2}$
18 $\quad$ **if** $p_c \leq P_{cx}$ **then**
19 $\quad\quad$ **if** $|\overrightarrow{\alpha}| = 4$ **then**
20 $\quad\quad\quad A \leftarrow |x_{1,1} - x_{2,1}| + |x_{1,2} - x_{2,2}|$
21 $\quad\quad\quad B \leftarrow |x_{1,1} - x_{2,2}| + |x_{1,2} - x_{2,1}|$
22 $\quad\quad\quad$ **if** $A \leq B$ **then**
23 $\quad\quad\quad\quad y_{1,1} \leftarrow \min(x_{1,1}, x_{2,1})$
24 $\quad\quad\quad\quad y_{1,2} \leftarrow \max(x_{1,1}, x_{2,1})$
25 $\quad\quad\quad\quad y_{2,1} \leftarrow \min(x_{1,2}, x_{2,2})$
26 $\quad\quad\quad\quad y_{2,2} \leftarrow \max(x_{1,2}, x_{2,2})$
27 $\quad\quad\quad$ **else**
28 $\quad\quad\quad\quad y_{1,1} \leftarrow \min(x_{1,1}, x_{2,2})$
29 $\quad\quad\quad\quad y_{1,2} \leftarrow \max(x_{1,1}, x_{2,2})$
30 $\quad\quad\quad\quad y_{2,1} \leftarrow \min(x_{1,2}, x_{2,1})$
31 $\quad\quad\quad\quad y_{2,2} \leftarrow \max(x_{1,2}, x_{2,1})$
32 $\quad\quad\quad$ **end**
33 $\quad\quad$ **else if** $|\overrightarrow{\alpha}| = 3$ **then**
34 $\quad\quad\quad y_{1,1}, y_{1,2} \leftarrow \alpha_1, \alpha_2$
35 $\quad\quad\quad y_{2,1}, y_{2,2} \leftarrow \alpha_2, \alpha_3$
36 $\quad\quad$ **else**
37 $\quad\quad\quad y_{1,1}, y_{2,2} \leftarrow \alpha_1, \alpha_2$
38 $\quad\quad\quad y_{1,2} \sim \mathbb{U}(\alpha_1, \alpha_2)$
39 $\quad\quad\quad y_{2,1} \leftarrow y_{1,2}$
40 $\quad\quad$ **end**
41 $\quad$ **end**
42 **return** $(\overrightarrow{Y_1}, \overrightarrow{Y_2})$

tion, effectively simulating the idea of a binary search. Another vital parameter to specify a binomial distribution is p, the success rate of trials, which can be interpreted as the probability of extending ($p > 0.5$) or shrinking ($p < 0.5$) the SoI in this context. In lines 5 to 10 of Algorithm 4, given τ, a tolerable (or ideal) ratio of the SoI length to the length of the time series, the value p is determined such that the peak of the probability mass function of the binomial distribution is close to $\lfloor M \cdot \tau \rfloor$. Empirical illustrations of this concept are provided in Figure 1. As shown in Figure 1a, under the same τ, the success rate decreases (increases) as the SoI length increases (decreases). Additionally, for the same SoI length, an increase (decrease) in τ leads to a corresponding increase (decrease) in the likelihood of extending the SoI. Furthermore, Figure 1b demonstrates that the probability of sampling a new length close to the tolerable length is increased, enabling the adaptive adjustment of the SoI length, particularly discouraging excessively long SoIs. This can be observed as the *peak* of $n = 80$ (SoI length 40) is shifted further left of the tolerable length compared to that of $n = 40$ (SoI length 20).

Algorithm 4: The Mutation Operator.

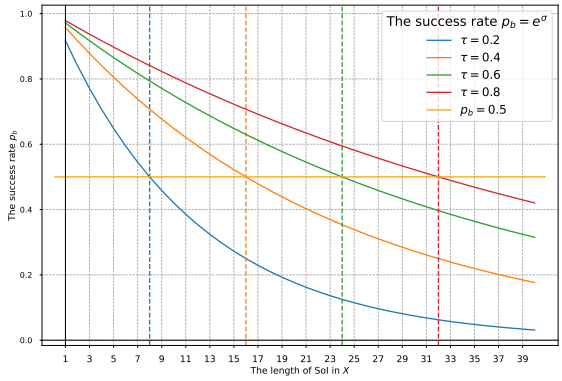
Input: The individual $\overrightarrow{X} = (x_1, x_2, x_3)$; mutation probability P_{mu}; tolerable SoI length ratio τ; M, the number of timestamps (length) of the target time series T.

Output: The offspring $\overrightarrow{Y} = (y_1, y_2, y_3)$ after mutation.

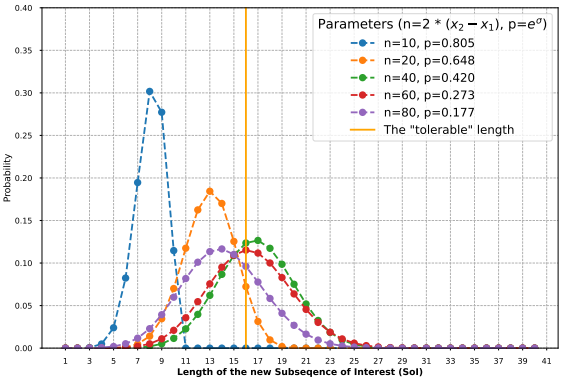
1 **Function** `Mutate`$(\overrightarrow{X}, P_{mu}, \tau, M)$**:**
2 $\quad \overrightarrow{Y} \leftarrow \overrightarrow{X}$
3 $\quad p_u \sim \mathbb{U}(0,1)$
$\quad\quad$ ▷ Configure the p of a $\textbf{Bin}(n,p)$
4 $\quad$ **if** $p_u \leq P_{mu}$ **then**
5 $\quad\quad$ **if** $\tau \in (0,1)$ **then**
6 $\quad\quad\quad \sigma \leftarrow \frac{\log(0.5)}{\tau} \cdot \frac{x_2 - x_1}{M}$
7 $\quad\quad\quad p_b \leftarrow e^{\sigma}$
8 $\quad\quad$ **else**
9 $\quad\quad\quad p_b \leftarrow 0.5$
10 $\quad\quad$ **end**
$\quad\quad\quad$ ▷ Determine the direction to scale
$\quad\quad p_s \sim \mathbb{U}(0,1)$
$\quad\quad\quad$ ▷ Sample the new SoI length
$\quad\quad l \sim \textbf{Bin}(2(x_2 - x_1), p_b)$
11 $\quad\quad$ **if** $p_s \leq 0.5$ **then**
12 $\quad\quad\quad y_2 \leftarrow \min(\max(y_1 + l, y_1 + 1), M)$
13 $\quad\quad$ **else**
14 $\quad\quad\quad y_1 \leftarrow \max(\min(y_2 - l, y_2 - 1), 1)$
15 $\quad\quad$ **end**
16 $\quad$ **end**
17 **return** $\overrightarrow{Y}$

3.4 Modeling of the New SoI

It is noteworthy that several previous works also separate counterfactual generation into two stages: the discovery of SoIs (or other forms of sensitivity analysis) and the subsequent generation or search for the actual values. While the crossover and mutation operators described above facilitate the search for optimal SoIs within the time series, the specific values



(a) An empirical plot of multiple success rates p_b versus the length of SoI in to-be-mutated individuals X under various tolerable SoI length ratios τ.



(b) An example Probability Mass Function (PMF) of a binomial distribution under our configuration. The tolerable SoI length ratio is set to $\tau = 0.4$, which results in a tolerable length of 16. And the five displayed PMFs are calculated based on SoI lengths 5, 10, 20, 30, and 40, respectively.

Figure 1: In both figures, the length of the to-be-explained time series is $M = 40$.

corresponding to these subsequences must still be determined to find true counterfactual explanations. Existing approaches to determining the values for SoIs can be broadly categorized into three types: 1. heuristic methods, which iteratively optimize specific criteria (Wachter et al., 2017; Höllig et al., 2022); 2. direct use of (sub)sequences from reference data, typically selected from the training set (Delaney et al., 2021; Li et al., 2022); 3. generation of values using machine learning models, such as generative models (Lang et al., 2023; Huang et al., 2024). Interestingly, the second strategy essentially complements the last. By combining these two approaches, it becomes feasible to obtain an SoI efficiently without being restricted to values from the reference data, which is the foundation of our proposed approach.

The proposed method relies on two assumptions. Given a univariate time series T of length m and one of its counterfactual candidate $\hat{T}$. Suppose the located subsequence of interest is between timestamps i and j where $i < j$, then the first assumption is as follows:

Assumption 1. *The time series $T = \{t_k \in \mathbb{R}\}_{k=1}^{m}$ and its counterfactual $\hat{T} = \{\hat{t}_k \in \mathbb{R}\}_{k=1}^{m}$ differ only in the subsequence of interest, i.e., $t_k = \hat{t}_k$, $\forall k \notin [i, j]$.*

While this assumption aims to ensure the true importance of the found SoI, the second assumption is developed to further restrict the values of SoI to a simplified form for efficient modeling and generation.

Assumption 2. *The difference between the SoI of the counterfactual candidate $\hat{T} = \{\hat{t}_k \in \mathbb{R}\}_{k=1}^{m}$ and that of the target time series $T = \{t_k \in \mathbb{R}\}_{k=1}^{m}$, denoted as $Z = \{\hat{t}_k - t_k\}_{k=i}^{j}$, is a stationary process where the current values are linearly dependent on its past values. Consequently, Z can be modeled by an autoregressive model of order p (AR-p) with a Gaussian error term.*

This assumption is straightforward and aims to define the nature of the difference between the SoIs. To the best of our knowledge, in contrast to previous approaches that directly obtain the observations (values) of counterfactuals, we are the first to model the difference between the counterfactual and the to-be-explained time series instance. Moreover, in section 3.1, it is mentioned that this method is reference-guided. By utilizing the reference instances found using Algorithm 2 and the chromosome representations, we now describe the methodology of generating counterfactual instances in Algorithm 5. Given

Algorithm 5: Generation of the SoI of counterfactual candidate
Input: The target time series $T = \{t_k \in \mathbb{R}\}_{k=1}^{m}$; the individual $\overrightarrow{X} = (x_1, x_2, x_3)$; a reference sample $\widetilde{T} = \{\tilde{t}_k \in \mathbb{R}\}_{k=1}^{m}$; the order (lags) of the autoregressive model p.
Output: The counterfactual candidate $\hat{T}$

Function `Generate`$(T, \overrightarrow{X}, \psi, p, m)$**:**
  $\hat{T} \leftarrow T$
  $l \leftarrow x_2 - x_1$
  $i,\ j \leftarrow \max(x_1 - p, 1),\ \min(x_2 + p, m)$
  $\zeta \leftarrow \{\tilde{t}_k - t_k\}_{k=i}^{j}$
  Fit an autoregressive model `AR`(p) on ζ
  Predict $\overline{\zeta} \leftarrow \{\bar{t}_k\}_{k=1}^{j-i+1}$ using the `AR`(p)
  $i^* \leftarrow \min(x_1, p)$
  $j^* \leftarrow i^* + l$
  $\hat{T}[x_1 : x_2] \leftarrow \overline{\zeta}[i^* : j^*] + T[x_1 : x_2]$
return $\hat{T}$

a target time series T and a chromosome representation of the SoI $\overrightarrow{X}$, the algorithm generates a counterfactual candidate based on a chosen instance $\hat{T}$ from

the reference set. In lines 4 and 5, the element-wise differences between the reference and the target are computed as a new time series ζ. Further, from lines 6 to 10, an autoregressive model of order p is fitted to learn the distribution of ζ through conditional maximum likelihood estimation. The in-sample predictions are then added to the SoI values of T to form the counterfactual SoI. In this context, the autoregressive model essentially acts as a process of denoising and *smoothing*.

4 EXPERIMENTAL SETUP

The experiments are performed using a diverse selection of time-series classification benchmarks and two popular time-series classifiers.

The UCR datasets (Bagnall et al., 2017) from the time-series classification archive (Chen et al., 2015) chosen in this study represent a diverse array of application domains in time-series classification, all of which are one-dimensional. Furthermore, they are categorized in Table 1 based on their domain specificity and characteristics, providing a comprehensive assessment of the proposed method's robustness and adaptability across varied types of time series data.

Table 1: The summary of the datasets along with the accuracy of the trained classifiers used in the experiments.

Dataset	Length	Train + Test Size	Balanced	Classes	Catch22	STSF
ECG200	96	100 + 100	No	2	0.83	0.87
GunPoint	150	50 + 150	Yes	2	0.94	0.94
Coffee	286	28 + 28	Yes	2	1.00	0.964
CBF	128	30 + 900	Yes	3	0.96	0.98
Beef	470	30 + 30	Yes	5	0.60	0.66
Lightning7	319	70 + 73	No	7	0.69	0.75
ACSF1	1460	100 + 100	Yes	10	0.87	0.82

For the classification tasks within our framework, each dataset was trained and tested using two specific classifiers: the *Catch22* classifier (C22) (Lubba et al., 2019) and the *Supervised Time Series Forest* (TSF) (Cabello et al., 2020). These classifiers were meticulously chosen not only for their computational efficiency and ease of implementation but also for their ability to produce probabilistic outputs. The generation of probabilistic outputs, rather than mere binary labels, is crucial for our methodology. It allows for the nuanced detection of subtle variations in the probability distributions across different classes. This feature is integral to our approach as it supports the generation of counterfactual explanations that are highly sensitive to minor but significant shifts in the data.

4.1 Hyperparameters Setup

The customized NSGA-II algorithm central to our approach is designed with several hyperparameters that can influence its operation and performance. The standard NSGA-II hyper-parameters (and their setting between brackets), population size (50), number of generations (50), the probability of crossover (0.7) and probability of mutation (0.7), are very common in evolutionary optimization algorithms, and are set based on the results of a few trials. A new hyperparameter, **Number of Reference Instances**, is set to 4 in this work. This parameter specifies both the number of reference cases used to evaluate the performance of a candidate solution and the number of *teachers* employed to guide the search for finding counterfactual examples. These hyper-parameters could be further optimized in future work.

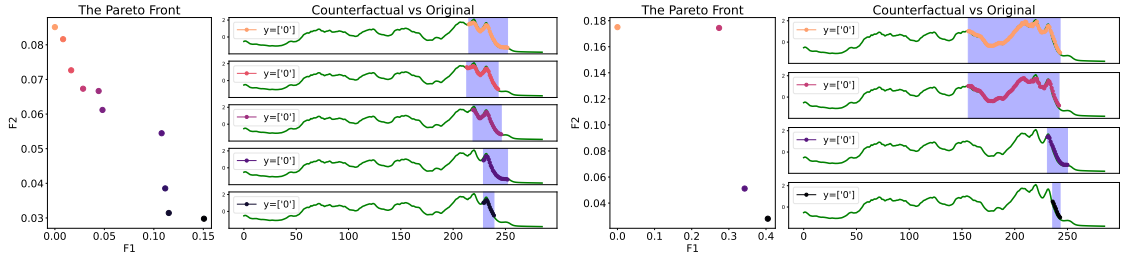
4.2 Metrics and Baselines

As introduced in section 4.2, five criteria can be used to assess the effectiveness of the counterfactual explanation algorithms. Given a to-be-explained time series $T = \{t_i \in \mathbb{R}\}_{i=1}^{m}$, a set of n counterfactuals candidates $\Theta = \{\hat{T}_j = \{\hat{t}_i \in \mathbb{R}\}_{i=1}^{m}\}_{j=1}^{N}$, and the classifier f, we define the five evaluation metrics as follow:

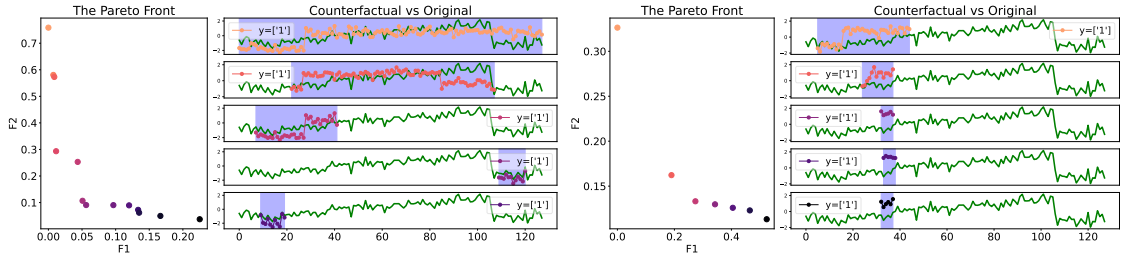
1. L1-Proximity$(T, \hat{T}) = \frac{|\hat{T}-T|}{|\hat{T}|+|T|}$
2. L2-Proximity$(T, \hat{T}) = \frac{|\hat{T}-T|_2}{|\hat{T}|_2+|T|_2}$
3. Validity$(T, \hat{T} \mid f) = \mathbf{1}_{f(T)\neq f(\hat{T})}$
4. Sparsity$(T, \hat{T}) = \frac{\sum_{i=[1,\ldots,m]} \mathbf{1}_{\hat{t}_i \neq t_i}}{m}$
5. Diversity$(T, \Theta \mid f) =$

$$\sum_{i=1}^{N-1} \Big(\prod_{j=i+1}^{N} \mathbf{1}_{\hat{T}_i \neq \hat{T}_j} \Big) \cdot \mathbf{1}_{f(T)\neq f(\hat{T}_i)},\ \hat{T}_i, \hat{T}_j \in \Theta$$

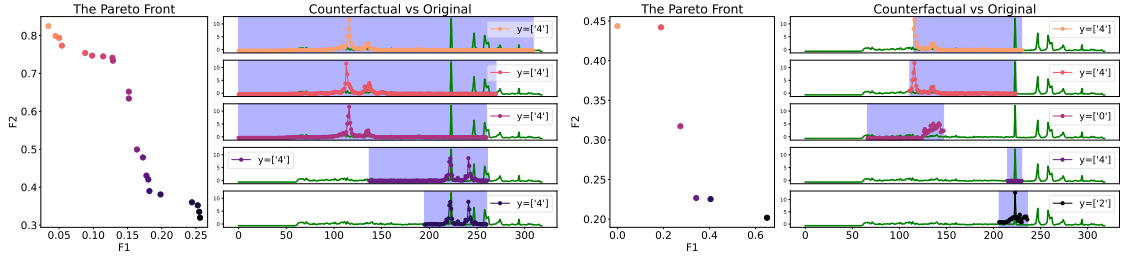
Among the five metrics, the first four are pairwise metrics. In contrast, Diversity is focused on measuring the number of unique and valid counterfactuals generated by the explainer for each target time series in a single run. In this work, we evaluate the performance of TX-Gen against four well-established algorithms which can be separated into two groups: w-CF (Wachter et al., 2017) and NativeGuide (Delaney et al., 2021), the hall-of-fame baselines; TSEvo (Höllig et al., 2022) and AB-CF (Li et al., 2023), the state-of-the-art counterfactual explainers.



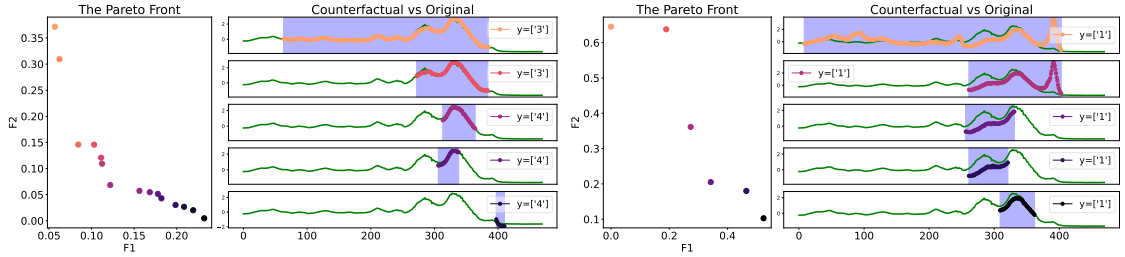
(a) An example of the Pareto-efficient counterfactuals for the test sample (20) of a binary classification task *Coffee* using Catch22 (left) and Supervised Time-series Forest (right).



(b) An example of the Pareto-efficient counterfactuals for the test sample (10) of a binary classification task *CBF* using Catch22 (left) and Supervised Time-series Forest (right).



(c) An example of the Pareto-efficient counterfactuals for the test sample (35) of a multi-class classification task *Lightning7* using Catch22 (left) and Supervised Time-series Forest (right).



(d) An example of the Pareto-efficient counterfactuals for the test sample (10) of a multi-class classification task *Beef* using Catch22 (left) and Supervised Time-series Forest (right).

Figure 2: Selection of examples of TX-Gen on different datasets and using different classifiers. In each subfigure, the left part shows the distribution of objective values (F_1 on the x axis and F_2 on the y axis) of the Pareto front. The right part of each figure displays the counterfactual examples and corresponding labels. The time series being explained is highlighted in green, while the counterfactual subsequence of interest (SoIs) are color-coded according to their position on the Pareto front.

Explainer	ECG200 Validity	Coffee Validity	GunPoint Validity	CBF Validity	Lightning7 Validity	ACSF1 Validity	Beef Validity
NG (C22)	0.56	0.46	0.43	0.59	–	–	0.83
w-CF (C22)	0.03	–	0.01	0.01	–	0.11	0.13
AB-CF (C22)	0.93	1.00	1.00	0.99	0.99	1.00	1.00
TSEvo (C22)	0.39	0.54	0.45	0.50	0.56	0.51	0.07
TX-Gen (C22)	1.00	1.00	1.00	1.00	1.00	1.00	1.00
NG (TSF)	0.35	0.50	0.50	–	–	–	0.10
w-CF (TSF)	–	–	–	0.01	0.11	0.17	
AB-CF (TSF)	0.74	1.00	1.00	1.00	0.99	0.99	0.97
TSEvo (TSF)	0.35	0.50	0.50	0.35	0.44	0.22	0.20
TX-Gen (TSF)	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 2: The **Validity** metric for each counterfactual XAI method on two different classifiers (TSF and C22) and seven different benchmark datasets.

5 RESULTS AND DISCUSSIONS

Our experimental results clearly demonstrate that TX-Gen consistently outperforms competing methods across several key metrics, as shown in Tables 2-5. In terms of *validity*, TX-Gen achieves a 100% success rate in generating valid counterfactuals across all datasets, regardless of the classifier used (Table 2). This significantly outperforms baseline methods like w-CF, which struggles with generating valid counterfactuals, achieving a mere $1\% \sim 17\%$ validity across different datasets and classifiers. TSEvo, while better than w-CF, also lags behind TX-Gen in validity, particularly when using the Catch22 classifier. AB-CF is in terms of validity only marginally worse than our proposed solution.

Sparsity is a key strength of TX-Gen, as demonstrated in Table 3. While methods like TSEvo also optimize for sparsity, TX-Gen achieves lower sparsity values across most datasets. In terms of proximity, Tables 4 and 5 show that TX-Gen performs competitively with state-of-the-art methods such as AB-CF and TSEvo, which also optimize for proximity. However, TX-Gen's use of multi-objective optimization allows it to maintain a strong proximity score while simultaneously achieving higher validity and diversity. The L1-Proximity and L2-Proximity results reflect that TX-Gen generates counterfactuals that are closer to the original time series, ensuring more interpretable and actionable insights.

As shown in the examples of Figure 2 and also in Table 6, our method generates a diverse set of Pareto-optimal solutions allowing for a wide range of plausible counterfactuals. All other methods only provide one counterfactual example per instance. This is particularly important in real-world applications, where generating multiple plausible alternatives can provide more comprehensive insights for end-users. The number of sub-sequences of the counterfactual examples by TX-Gen is always 1, while for TSEvo this is on average 13.8 sub-sequences, Native-Guide provides 1.8, w-CF 1.7 and AB-CF 1.8 sub-sequences on average over all datasets. In general, less sub-sequences are more informative for counterfactual examples. However, it could be a limitation that the proposed method only provides 1 sub-sequence, however, TX-Gen provides multiple examples per instance, one could combine the sub-sequences from these examples to alleviate this limitation. All results and source code can be found in our Zenodo repository[1].

In summary, TX-Gen's use of evolutionary multi-objective optimization and its reference-guided mechanism make it more effective than existing methods across key metrics. The method not only generates more valid and sparse counterfactuals but does so with a high degree of proximity and diversity, offering a significant advancement in the generation of explainable counterfactuals for time-series classification.

6 CONCLUSIONS

In this paper, we presented TX-Gen, a novel framework for generating counterfactual explanations for time-series classification tasks using evolutionary multi-objective optimization. Through extensive experimentation, we demonstrated that TX-Gen consistently outperforms state-of-the-art methods in terms of validity, sparsity, proximity, and diversity across multiple benchmark datasets. By leveraging the flexibility of the NSGA-II algorithm and incorporating a reference-guided mechanism, our approach ensures that the generated counterfactuals are both interpretable and computationally efficient.

The good performance of TX-Gen particularly in achieving 100% validity and maintaining high diversity while optimizing for proximity and sparsity, highlights its potential for real-world applications where model transparency is critical. Our proposed method strikes an effective balance between multiple conflicting objectives, offering a robust solution for generating meaningful counterfactuals in time-series classification.

Future work can explore several promising directions. First, further tuning of the hyper-parameters, particularly the number of reference instances, could lead to even greater improvements in performance. Additionally, extending TX-Gen to handle multivariate time-series data and real-time counterfactual generation could broaden its applicability to more complex, real-world scenarios.

[1] https://doi.org/10.5281/zenodo.13711886

Explainer	ECG200		Coffee		GunPoint		CBF		Lightning7		ACSF1		Beef	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
NG (C22)	0.403	0.310	0.962	0.131	0.052	0.065	0.989	0.067	–	–	–	–	1.000	0.000
w-CF (C22)	0.010	0.000	–	–	0.007	0.000	0.669	0.468	–	–	0.092	0.287	0.252	0.432
AB-CF (C22)	0.430	0.254	0.437	0.106	0.525	0.296	0.461	0.276	0.350	0.269	0.534	0.334	0.467	0.267
TSEvo (C22)	0.408	0.154	0.280	0.189	0.237	0.197	0.485	0.213	0.347	0.271	0.558	0.291	0.698	0.302
TX-Gen (C22)	0.113	0.137	0.016	0.016	0.121	0.161	0.043	0.052	0.044	0.056	0.041	0.086	0.040	0.091
NG (TSF)	0.967	0.096	0.932	0.169	0.963	0.061	–	–	–	–	–	–	0.667	0.470
w-CF (TSF)	–	–	–	–	–	–	–	–	0.006	0.000	0.818	0.385	0.207	0.397
AB-CF (TSF)	0.589	0.218	0.871	0.030	0.307	0.063	0.410	0.284	0.580	0.298	0.589	0.265	0.528	0.324
TSEvo (TSF)	0.154	0.105	0.180	0.072	0.170	0.084	0.194	0.076	0.140	0.072	0.512	0.338	0.128	0.040
TX-Gen (TSF)	0.113	0.131	0.054	0.037	0.051	0.046	0.049	0.029	0.030	0.027	0.053	0.081	0.033	0.044

Table 3: The **Sparsity** metric for each counterfactual XAI method on two different classifiers (TSF and C22) and seven different benchmark datasets.

Explainer	ECG200		Coffee		GunPoint		CBF		Lightning7		ACSF1		Beef	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
NG (C22)	0.121	0.086	0.098	0.260	0.010	0.011	0.383	0.220	–	–	–	–	1.000	0.000
w-CF (C22)	0.001	0.000	–	–	0.001	0.000	0.001	0.001	–	–	0.00003	0.00002	0.00003	0.0002
AB-CF (C22)	0.132	0.081	0.024	0.007	0.069	0.059	0.236	0.144	0.222	0.114	0.060	0.084	0.103	0.195
TSEvo (C22)	0.311	0.161	0.021	0.011	0.036	0.023	0.275	0.097	0.240	0.139	0.090	0.075	0.036	0.023
TX-Gen (C22)	0.062	0.076	0.002	0.002	0.039	0.067	0.032	0.037	0.045	0.062	0.008	0.017	0.047	0.133
NG (TSF)	0.164	0.118	0.026	0.011	0.062	0.029	–	–	–	–	–	–	0.515	0.365
w-CF (TSF)	–	–	–	–	–	–	–	–	0.0001	0.0000	0.035	0.061	0.0003	0.0001
AB-CF (TSF)	0.161	0.067	0.045	0.009	0.043	0.037	0.168	0.110	0.251	0.169	0.065	0.073	0.092	0.150
TSEvo (TSF)	0.067	0.054	0.016	0.005	0.015	0.012	0.073	0.029	0.083	0.041	0.067	0.074	0.143	0.110
TX-Gen (TSF)	0.056	0.060	0.007	0.004	0.012	0.019	0.030	0.020	0.039	0.062	0.012	0.020	0.020	0.036

Table 4: The **L1-Proximity** metric for each counterfactual XAI method on two different classifiers (TSF and C22) and seven different benchmark datasets.

Explainer	ECG200		Coffee		GunPoint		CBF		Lightning7		ACSF1		Beef	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
NG (C22)	0.182	0.068	0.105	0.259	0.046	0.034	0.478	0.216	–	–	–	–	1.000	0.000
w-CF (C22)	0.006	0.001	–	–	0.008	0.000	0.007	0.005	–	–	0.001	0.000	0.004	0.002
AB-CF (C22)	0.193	0.086	0.041	0.008	0.118	0.083	0.349	0.130	0.376	0.127	0.145	0.167	0.132	0.228
TSEvo (C22)	0.440	0.145	0.045	0.011	0.096	0.047	0.430	0.067	0.430	0.161	0.200	0.155	0.045	0.031
TX-Gen (C22)	0.134	0.105	0.017	0.008	0.076	0.098	0.121	0.076	0.117	0.127	0.036	0.061	0.084	0.207
NG (TSF)	0.175	0.113	0.036	0.016	0.091	0.046	–	–	–	–	–	–	0.561	0.395
w-CF (TSF)	–	–	–	–	–	–	0.001	0.000	0.058	0.128	0.058	0.128	0.002	0.000
AB-CF (TSF)	0.209	0.068	0.055	0.010	0.101	0.090	0.283	0.094	0.332	0.160	0.137	0.149	0.111	0.172
TSEvo (TSF)	0.169	0.094	0.043	0.007	0.051	0.038	0.181	0.046	0.210	0.113	0.125	0.105	0.357	0.255
TX-Gen (TSF)	0.133	0.107	0.027	0.011	0.047	0.057	0.112	0.062	0.100	0.102	0.045	0.060	0.060	0.104

Table 5: The **L2-Proximity** metric for each counterfactual XAI method on two different classifiers (TSF and C22) and seven different benchmark datasets.

Explainer	ECG200		Coffee		GunPoint		CBF		Lightning7		ACSF1		Beef	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
TX-Gen (C22)	6.95	2.73	7.46	3.52	10.70	4.08	14.45	4.39	25.07	8.93	24.86	8.49	20.80	6.60
TX-Gen (TSF)	4.89	0.92	4.79	0.82	4.12	1.26	5.27	1.00	5.89	1.67	6.58	2.28	6.13	1.28

Table 6: The **Diversity** metric for each counterfactual XAI method on two different classifiers (TSF and C22) and seven different benchmark datasets. (For all other methods this is always 1.00)

REFERENCES

Back, T., Fogel, D. B., and Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*. IOP Publishing Ltd., GBR, 1st edition.

Bäck, T. H., Kononova, A. V., van Stein, B., Wang, H., Antonov, K. A., Kalkreuth, R. T., de Nobel, J., Vermetten, D., de Winter, R., and Ye, F. (2023). Evolutionary algorithms for parameter optimization—thirty years later. *Evolutionary Computation*, 31(2):81–122.

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

Cabello, N., Naghizade, E., Qi, J., and Kulik, L. (2020). Fast and accurate time series classification through supervised interval search. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 948–953. IEEE.

Chen, J.-F., Chen, W.-L., Huang, C.-P., Huang, S.-H., and Chen, A.-P. (2016). Financial time-series data analysis using deep convolutional neural networks. In *2016 7th International conference on cloud computing and big data (CCBD)*, pages 87–92. IEEE.

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

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Delaney, E., Greene, D., and Keane, M. T. (2021). Instance-Based Counterfactual Explanations for Time Series Classification. In Sánchez-Ruiz, A. A. and Floyd, M. W., editors, *Case-Based Reasoning Research and Development*, pages 32–47, Cham. Springer International Publishing.

Endres, D. and Schindelin, J. (2003). A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860.

Goldberg, D. E. and Deb, K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In Rawlins, G. J. E., editor, *Foundations of Genetic Algorithms*, volume 1, pages 69–93. Elsevier.

Höllig, J., Kulbach, C., and Thoma, S. (2022). TSEvo: Evolutionary Counterfactual Explanations for Time Series Classification. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 29–36.

Huang, Q., Chen, W., Bäck, T., and van Stein, N. (2024). Shapelet-based Model-agnostic Counterfactual Local Explanations for Time Series Classification. Presented at the AAAI 2024 Workshop on Explainable Machine Learning for Sciences (XAI4Sci).

Karlsson, I., Rebane, J., Papapetrou, P., and Gionis, A. (2018). Explainable time series tweaking via irreversible and reversible temporal transformations.

Karlsson, I., Rebane, J., Papapetrou, P., and Gionis, A. (2020). Locally and globally explainable time series tweaking. *Knowledge and Information Systems*, 62(5):1671–1700.

Lang, J., Giese, M. A., Ilg, W., and Otte, S. (2023). Generating Sparse Counterfactual Explanations for Multivariate Time Series. In Iliadis, L., Papaleonidas, A., Angelov, P., and Jayne, C., editors, *Artificial Neural Networks and Machine Learning – ICANN 2023*, pages 180–193, Cham. Springer Nature Switzerland.

Li, P., Bahri, O., Boubrahimi, S. F., and Hamdi, S. M. (2022). SG-CF: Shapelet-Guided Counterfactual Explanation for Time Series Classification. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 1564–1569.

Li, P., Bahri, O., Boubrahimi, S. F., and Hamdi, S. M. (2023). Attention-Based Counterfactual Explanation for Multivariate Time Series. In Wrembel, R., Gamper, J., Kotsis, G., Tjoa, A. M., and Khalil, I., editors, *Big Data Analytics and Knowledge Discovery*, pages 287–293, Cham. Springer Nature Switzerland.

Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., and Jones, N. S. (2019). catch22: CAnonical time-series CHaracteristics: Selected through highly comparative time-series analysis. *Data Mining and Knowledge Discovery*, 33(6):1821–1852.

Morid, M. A., Sheng, O. R. L., and Dunbar, J. (2023). Time series prediction using deep learning methods in healthcare. *ACM Transactions on Management Information Systems*, 14(1):1–29.

Refoyo, M. and Luengo, D. (2024). Sub-SpaCE: Subsequence-Based Sparse Counterfactual Explanations for Time Series Classification Problems. In Longo, L., Lapuschkin, S., and Seifert, C., editors, *Explainable Artificial Intelligence*, pages 3–17, Cham. Springer Nature Switzerland.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626.

Theissler, A., Spinnato, F., Schlegel, U., and Guidotti, R. (2022). Explainable ai for time series classification: a review, taxonomy and research directions. *Ieee Access*, 10:100700–100724.

Wachter, S., Mittelstadt, B., and Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *SSRN Electronic Journal*.

Ye, L. and Keogh, E. (2009). Time series shapelets: A new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 947–956, New York, NY, USA. Association for Computing Machinery.

Zhou, R., Bacardit, J., Brownlee, A., Cagnoni, S., Fyvie, M., Iacca, G., McCall, J., van Stein, N., Walker, D., and Hu, T. (2024). Evolutionary computation and explainable ai: A roadmap to transparent intelligent systems.