

Counterfactual Explainable AI (XAI) Method for Deep Learning-Based Multivariate Time Series Classification

Alan G. Paredes Cetina¹, Kaouther Benguessoum¹, Raoni Lourenço¹, Sylvain Kubler¹,

¹SnT, University of Luxembourg

alan.paredes@uni.lu, kaouther.benguessoum@uni.lu, raoni.lourenco@uni.lu, sylvain.kubler@uni.lu

Abstract

Recent advances in deep learning have improved multivariate time series (MTS) classification and regression by capturing complex patterns, but their lack of transparency hinders decision-making. Explainable AI (XAI) methods offer partial insights, yet often fall short of conveying the full decision space. Counterfactual Explanations (CE) provide a promising alternative, but current approaches typically prioritize either accuracy, proximity or sparsity – *rarely all* – limiting their practical value. To address this, we propose CONFETTI, a novel multi-objective CE method for MTS. CONFETTI identifies key MTS subsequences, locates a counterfactual target, and optimally modifies the time series to balance prediction confidence, proximity and sparsity. This method provides actionable insights with minimal changes, improving interpretability, and decision support. CONFETTI is evaluated on seven MTS datasets from the UEA archive, demonstrating its effectiveness in various domains. CONFETTI consistently outperforms state-of-the-art CE methods in its optimization objectives, and in six other metrics from the literature, achieving $\geq 10\%$ higher confidence while improving sparsity in $\geq 40\%$.

Code — <https://github.com/serval-uni-lu/confetti>

1 Introduction

Deep Learning (DL) models, such as Convolutional Neural Networks (CNNs) (Wang, Yan, and Oates 2017a), Recurrent Neural Networks (RNNs) (Liu et al. 2020), and Transformer-based models (Schäfer and Leser 2018), have recently shown strong performance in multivariate time series (MTS) tasks (Ismail Fawaz et al. 2019; Khan et al. 2023). However, their lack of transparency makes it hard for decision makers to interpret predictions (Ruiz et al. 2021). Explainable AI (XAI) methods help decision makers understand the rationale behind DL model predictions (Adadi and Berrada 2018). Even high-quality explanations may not reveal the full decision space. For example, an AI system might advise a broker to sell a stock due to recent volatility, and XAI could highlight the spike as the key factor. Yet the broker might not realize that a slight drop in volatility could have changed the recommendation. Counterfactual Explanations (CE), a subset of XAI, help fill this gap by showing

how small input changes can alter predictions, thus revealing critical decision factors (Spinnato et al. 2022).

State-of-the-art CE methods for MTS mostly rely on shapelet-based techniques (Bahri, Boubrahimi, and Hamdi 2022), attention mechanisms (Li et al. 2023), or hybrid instance- and rule-based approaches (Spinnato et al. 2023). However, they often optimize a single objective, either (i) maximizing *prediction confidence*, which may require large changes in the original time series, (ii) promoting *sparsity*, which can reduce explanation accuracy, or (iii) improving *proximity*, which focuses on minimizing the distance from the original instance but may lead to out-of-distribution instances. This narrow focus limits the usefulness of CE, as all of these objectives are important to be jointly considered to generate actionable and understandable insights.

To address the lack of multi-objective CE methods for MTS classification, we introduce a novel method called CONFETTI (COuNterFactual Explanations for mulTivariate Time series). It employs a four-step approach: it first locates the closest instance from the opposite class as the counterfactual target, then uses Class Activation Maps (CAMs) (Zhou et al. 2016) to identify the most influential subsequences. Next, it substitutes values from the target into the original series to create an initial CE, and finally optimizes this CE to balance prediction confidence, proximity, and sparsity, while ensuring plausibility, and validity by design.

In summary, the contributions of the paper are:

- A novel multi-objective CE method for MTS that optimizes prediction confidence, proximity, and sparsity, while ensuring plausibility and validity by design;
- A comprehensive benchmark against CoMTE, SETS, and TSEvo CE methods conducted on seven datasets from the UEA archive using two model architectures;
- An ablation study of the initial stage showing how the search for most influential subsequence through feature weights affects performance.
- A sensitivity analysis of our method parameters, examining how variations in their values affect performance;

Section 2 provides a review of the CE literature for MTS. Section 3 introduces the concepts used in our approach and formally define the problem. Section 4 details the workings of CONFETTI. Section 5 presents the experimental setup and results. Section 6 summarizes our contributions.

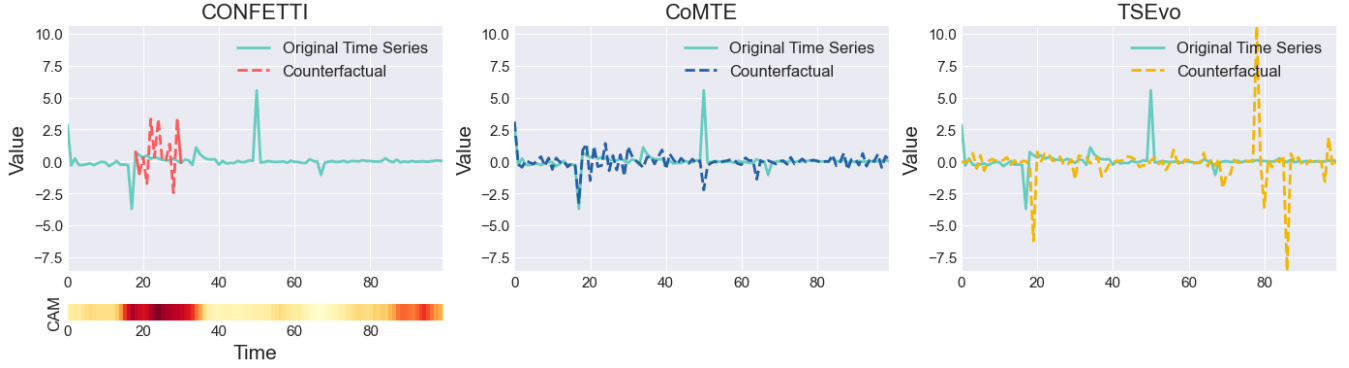


Figure 1: Counterfactual explanations generated by i) CONFETTI; ii) CoMTE; and iii) TsEVO for a single dimension of a multivariate time series. Below CONFETTI example is the Class Activation Map of the Nearest Unlike Neighbour as a heatmap, showing in red the most relevant timesteps.

2 Related Work

CE are expected to satisfy several desiderata outlined in the XAI literature (Guidotti 2024): (i) *Validity*: the CE results in a different predicted class compared to the original instance; (ii) *Confidence*: the predicted probability associated with the target class should be sufficiently high, providing a measure of how strongly the model supports the CE outcome; (iii) *Sparsity*: the number of changes relative to the original instance is minimal; (iv) *Proximity*: the magnitude of these changes is as small as possible; (v) *Plausibility*: the CE should have feature values that lie within the distribution of a reference population, avoiding unrealistic values.

CoMTE (Ates et al. 2021) was the first CE method for MTS, which substitutes one channel of a MTS instance at a time with the corresponding channel of the nearest unlike neighbor (NUN). SETS (Bahri, Boubrahimi, and Hamdi 2022) uses the Shapelet Transformer to identify the most relevant subsequences per class (shapelets). Then, it substitutes the identified shapelets of the instance of interest for those of the NUN. LASTS (Spinnato et al. 2023) takes advantage of the growing sphere algorithm to create a latent space around the target instance, identifying the NUN. It builds a neighborhood around this NUN to generate a set of CEs. AB-CE (Li et al. 2023) uses a sliding window technique to generate multiple candidate subsequences, which are then evaluated based on Shannon Entropy to select the most informative ones. It then proceeds to swap the corresponding instance subsequences to be explained for the candidate subsequences one by one until the classification label changes. TSEvo (Höllig, Kulbach, and Thoma 2022) formulates the search for CEs as a multi-objective optimization problem that incorporates proximity, sparsity, and plausibility. It initializes a population of candidate counterfactuals for a MTS instance and uses a customized version of NSGA-II (Deb et al. 2002) to evolve them. Individuals are evaluated based on the defined objectives, ranked using non-dominated sorting, and selected via tournament selection that considers both rank and crowding distance. Offspring are generated via crossover and mutation, repeated until the target number of generations is reached.

Table 1: Summary of CE XAI methods for MTS. **Generation** columns show which objectives are optimized during CE generation, while **Evaluation** columns indicate which aspects are later assessed in the studies. Approach: He = Heuristic, Sh = Shapelet, At = Attention Mechanism, La = Latent space, Mo = Multi-objective. Scope: In = Instance-based, Su = Subsequence-based, Tp = Time-point based.

Method	Approach	Scope	Generation					Evaluation				
			Validity	Confidence	Sparsity	Plausibility	Proximity	Validity	Confidence	Sparsity	Plausibility	Proximity
CoMTE	He	In										
SETS	Sh	Su										
AB-CE	At	Su										
LASTS	La	Su										
TSEvo	Mo	Tp										
CONFETTI	Mo	Tp										

Limitations While all methods ensure Plausibility by design, only a few approaches (notably ours and LASTS) also enforce Validity by design. Beyond these requirements, most methods focus on a single optimization goal, either minimizing the distance between the original instance and its CE (e.g., LASTS, SETS) or maximizing the prediction confidence of the CE (e.g., CoMTE, AB-CE). Methods that attempt to address multiple criteria simultaneously, such as TSEvo and CoMTE, differ in how they explore the search space: TSEvo uses uninformed population-based exploration, making it costly and inefficient for high-dimensional or long time series, whereas CoMTE employs a more structured search, but still does not incorporate prior knowledge about the input space. To highlight the limitations of current methods,

Figure 1 illustrates an example of CE generated by our method, CoMTE and TSEvo. The panel below CONFETTI shows the CAM of the NUN used to guide its changes. CoMTE and TSEvo largely replace the series with the NUN,

resulting in low sparsity. By focusing on CAM-highlighted regions, CONFETTI achieves sparse and interpretable CEs. To address the limitations raised, we propose CONFETTI, which incorporates informed guidance through relevant subsequences and NUNs to perform the search process more effectively, drawing inspiration from CE techniques in other domains such as computer vision (Dandl et al. 2020; Navas-Palencia 2021).

3 CE Problem Definition

This section defines the core concepts and notation used in CONFETTI.

Basic Notation. Scalars are denoted by lowercase letters (e.g., $a, b, t, d, \alpha, \beta$). Sets are denoted by uppercase letters (e.g., A), and their elements by uppercase letters with subscripts (e.g., A_i). When the elements of a set are vectors (e.g., a timeseries), we use superscripts to denote its elements (e.g., A_i^t is the t^{th} element of A_i). We use two comma-separated superscript indices to indicate a contiguous sequence of elements of a vector (e.g., $A_i^{s,e} = \{A_i^s, A_i^{s+1}, \dots, A_i^e\}$). When each A_i^t is itself a vector (e.g., a MTS), square brackets are used to select a specific element of it, so that $A_i^t[d]$ denotes the d -th element of A_i^t . We also define the operator $\parallel$ to denote the concatenation of sequences. For example, $A_i \parallel A_j = A_i^0, A_i^1, \dots, A_i^{|A_i|}, A_j^0, A_j^1, \dots, A_j^{|A_j|}$.

Definition 1 (Classifier, Instance, Prediction). A *classifier* f is an estimator trained with MTS samples. We denote by $X_i \in X$ the i^{th} *instance* of a set of MTSs X . A *prediction* $f(X_i)$ is the class predicted by the classifier f for X_i .

Since an instance X_i is a MTS, let us note that it can be sliced along its temporal and channel dimensions.

Definition 2 (Subsequence). A *subsequence* $X_i^{s,e} \subseteq X_i$ is a contiguous segment of the original MTS instance X_i from time s to time e , so $X_i^{s,e} = \{X_i^s, X_i^{s+1}, \dots, X_i^e\}$, with $0 \leq s < e < |X_i|$. Each X_i^t is a vector that contains the values of all channels at time t .

Definition 3 (Nearest Unlike Neighbor - NUN). Given a dataset X , a classifier f , a distance metric between two instances $d(X_a, X_b)$, and an instance $X_i \in X$, we denote the NUN of X_i an instance $nun \in X \mid f(nun) \neq f(X_i)$, and $\nexists X_s \in X \mid f(X_s) \neq f(X_i)$ and $d(X_s, X_i) < d(nun, X_i)$.

Definition 4 (Counterfactual Set, Counterfactual). Given a dataset X , a classifier f that outputs the prediction $f(X_i)$ for an instance $X_i \in X$, we denote by $C(X_i)$ the set of *counterfactuals* for X_i , where each $C_j \in C(X_i)$ satisfies $f(C_j) \neq f(X_i)$.

To ensure that the CE values are within the data manifold, CONFETTI – as detailed in section 4.2 – and many state-of-the-art methods construct counterfactuals by altering an instance X_i with values from its *nun*, thus promoting plausibility (Theissler et al. 2022).

Definition 5 (Prediction Confidence). Given a classifier f , an instance X_i , and a target class c , we define the *prediction confidence* for class c as the probability $P(f(X_i) = c)$

assigned by the classifier f . This value quantifies the probability that the model considers class c to be the correct prediction for instance X_i .

Definition 6 (Minimality). Given an instance X_i and a CE $C_j \in C(X_i)$, we define *minimality* as the distance $d(X_i, C_j)$ between the original instance and its CE, according to a predefined distance metric $d(\cdot, \cdot)$. A lower value of d indicates that the CE requires fewer or smaller changes to alter the prediction of the model, and therefore it is considered to be more minimal.

Problem Statement. Given a classifier f , an instance $X_i \in X$ with predicted label $f(X_i)$, and a predefined distance metric $d(\cdot, \cdot)$, the objective is to find a set of *counterfactuals* $C(X_i)$ such that for each *counterfactual instance* $C_j \in C(X_i)$ it holds that $f(C_j) \neq f(X_i)$, the distance $d(X_i, C_j)$ is as *minimal* as possible, and the *prediction confidence* $P(f(C_j) = c)$, for some class $c \neq f(X_i)$, is as large as possible.

4 CONFETTI

The process behind CONFETTI consists of four main stages. First, a time series instance X_i is selected, and its *nun* is identified as the counterfactual target. Second, the most influential subsequences affecting the prediction are identified. Third, values from *nun* are substituted into X_i to generate an initial CE that serves as the first element of the counterfactual set $C(X_i)$, denoted by C_0 . Fourth, C_0 is passed to the optimization stage, which optimizes prediction confidence, proximity, and sparsity, producing an optimized set of CEs that belong to $C(X_i)$. These stages are respectively detailed in Sections 4.2 to 4.5. We also outline the entire process in Algorithm 1.

4.1 Feature Weight Vector

An optional component of CONFETTI is the use of a feature-weight vector W_i to identify the most relevant contiguous subsequence within a MTS. This vector assigns an importance score to each time step, and subsequences with the highest cumulative weight are selected as candidates for modification. The weights can be obtained with techniques such as CAM (Zhou et al. 2016), which requires a DL model with global average pooling to highlight key time steps by averaging feature maps before classification, enabling targeted rather than full-series perturbations. When a feature-weight vector W_i is not available, for example due to the model architecture, CONFETTI skips the Subsequence extraction and Naive Stage steps (see Section 4.3 & 4.4) and proceeds directly to the optimization stage, in which case the method operates in a fully model-agnostic manner.

4.2 Retrieve the NUN

We define $findNUN(\cdot)$ as the function that identifies the nearest unlike neighbor (*nun*) of a query instance X_i . It first filters the reference set R to include only instances with a predicted label different from $f(X_i)$. A k -nearest neighbors search is then performed among these using a chosen distance metric. Candidates with classifier confidence above

threshold θ are retained. If none meet this criterion, the function returns no neighbor and the process halts. Otherwise, it returns the closest valid nun and its label $f(nun) = c$.

Algorithm 1: Algorithm of CONFETTI

Input: Model f ; instance X_i ; reference data R ; CAM weights of R W ; confidence weight α ; threshold θ ; partitions p ; generations g ; population size z ; crossover probability p_c , mutation probability p_m

```

1:  $(nun, c) \leftarrow \text{findNUN}(f, X_i, R, \theta)$ 
2: if  $nun = \perp$  then
3:   return FAIL
4: else
5:   if  $\exists W$  then
6:      $\ell \leftarrow 2$ 
7:      $s, e \leftarrow \text{findSubsequence}(W_{nun}, \ell)$ 
8:      $C_0 \leftarrow X_i^{0, s-1} \parallel nun^{s, e} \parallel X_i^{e+1, |X_i|-1}$ 
9:     while  $P(f(C_0) = c) < \theta$  do
10:       $\ell \leftarrow \ell + 1$ 
11:       $s, e \leftarrow \text{findSubsequence}(W_{nun}, \ell)$ 
12:       $C_0 \leftarrow X_i^{0, s-1} \parallel nun^{s, e} \parallel X_i^{e+1, |X_i|-1}$ 
13:     end while
14:      $C(X_i) \leftarrow \{C_0\}$ 
15:      $S(C(X_i)) \leftarrow \{[0, 0]\}$ 
16:      $h \leftarrow \ell$ 
17:   else
18:      $C(X_i) \leftarrow \emptyset$ 
19:      $S(C(X_i)) \leftarrow \emptyset$ 
20:      $h \leftarrow |X_i|/2$ 
21:   end if
22:    $l \leftarrow 1$ 
23:   while  $l \leq h$  do
24:      $k \leftarrow \left\lfloor \frac{l+h}{2} \right\rfloor$ 
25:      $s, e \leftarrow \text{findSubsequence}(W_{nun}, k)$ 
26:      $H \leftarrow \text{referencePoints}(2, p)$ 
27:      $P_0 \leftarrow \text{binarySampling}(X_i, nun, s, e, z, H)$ 
28:      $\text{evaluateObjectives}(P_0, \alpha, \theta)$ 
29:      $P_g, S_g \leftarrow \text{nsgaIII}(P_0, g, z, p_c, p_m, \theta)$ 
30:     if  $P_g = \emptyset$  then
31:        $l \leftarrow k + 1$ 
32:     else
33:        $C(X_i) \leftarrow C(X_i) \cup P_g$ 
34:        $S(C(X_i)) \leftarrow S(C(X_i)) \cup S_g$ 
35:        $h \leftarrow k - 1$ 
36:     end if
37:   end while
38:   return  $C(X_i), \arg\max S(C(X_i)) \times [\alpha, 1 - \alpha]$ 
39: end if

```

4.3 Subsequence extraction

This step uses the classifier’s feature-importance vector of nun , denoted by $W_{nun} = \{W_{nun}^0, W_{nun}^1, \dots, W_{nun}^{|nun|-1}\}$, where W_{nun}^t is the importance weight associated with time step t . This vector, obtained using CAM, is used to locate the most relevant subsequence of length ℓ . A sliding window is applied to W_{nun} to identify the contiguous segment with

Algorithm 2: FINDSUBSEQUENCE(W_{nun}, ℓ)

Input: Importance weights W_{nun} ; subsequence length ℓ
Output: Indices s, e such that $nun^{s, e}$ is the most relevant subsequence of length ℓ

```

1:  $s \leftarrow 0$ 
2:  $maxSum \leftarrow \sum_{i=0}^{\ell-1} W_{nun}^i$ 
3:  $currSum \leftarrow maxSum$ 
4: for  $i = 1$  to  $|W_{nun}| - \ell$  do
5:    $currSum \leftarrow currSum - W_{nun}^{[i-1]} + W_{nun}^{[i+\ell-1]}$ 
6:   if  $currSum > maxSum$  then
7:      $maxSum \leftarrow currSum$ 
8:      $s \leftarrow i$ 
9:   end if
10: end for
11:  $e \leftarrow s + \ell - 1$ 
12: return  $s, e$ 

```

the largest cumulative weight, thereby restricting the perturbation to a smaller set of time steps. Although inspired by (Delaney, Greene, and Keane 2021), CONFETTI adapts this mechanism for MTS by averaging CAM values across channels, as implemented in the FINDSUBSEQUENCE(W_{nun}, ℓ) function (Algorithm 2).

4.4 Naive Stage

In this step, the values of instance X_i are altered by substituting them with the corresponding values from the same time steps in its nun . This stage is referred to as *naive* because it applies the modification uniformly across all channels. Specifically, for a subsequence $X_i^{s, e} \subseteq X_i$ spanning a set of time steps, all values are replaced with those from the equivalent subsequence in nun , yielding the counterfactual instance C_0 .

The process begins by setting the subsequence length $\ell = 2$, identifying the target class $f(nun) = c$, and selecting the most influential subsequence $nun^{s, e} \in nun$ based on CAM weights, with $e - s = \ell$. The corresponding segment $X_i^{s, e}$ in X_i is then replaced with $nun^{s, e}$, creating a candidate C_0 . The classifier’s confidence in predicting class c for C_0 , denoted $P(f(C_0) = c)$, is computed. If this value meets or exceeds the user-defined threshold θ , the counterfactual is accepted. Otherwise, ℓ is incremented and the process repeats until the threshold is satisfied.

In the next stage, we input the optimization algorithm’s final time window (s, e) ; the optimization will occur within this window. In the most extreme scenario, the *naive stage* algorithm will replace all values of X_i with those of nun .

4.5 Optimization

CONFETTI aims to optimize the prediction confidence of the target class c , the distance between CE $C_j \in C(X_i)$ and X_i , and the number of changes made to a CE C_j . To evaluate the number of changes between CE $C_j \in C(X_i)$ and X_i , we use the Hamming distance (Hamming 1950):

$$\text{Hamming}(X_i, C_j) = \sum_{m=1}^t \sum_{n=1}^u \mathbf{1}_{[X_i^m[n] \neq C_j^m[n]]}$$

That is, the total number of elements changed between X_i and C_j in all t time steps and u channels

Our multi-objective optimization problem is then formalized as follows:

$$\begin{aligned} \max \quad m_1 &= \sum_{C_i \in C(X_i)} P(f(C_j) = c) \\ \min \quad m_2 &= \frac{1}{t \cdot u} \cdot \sum_{C_i \in C(X_i)} \text{Hamming}(X_i, C_j) \\ \min \quad m_3 &= \sum_{C_i \in C(X_i)} \text{dist}(X_i, C_j) \\ \text{s.t.} \quad P(f(C_j) = c) &\geq \theta, \forall C_j \in C(X_i) \end{aligned} \quad (1)$$

In (1), we maximize m_1 , the model’s confidence, defined as the sum of predicted probabilities for class c over all $C_j \in C(X_i)$ under f . The second objective, m_2 , promotes sparsity by summing the distances between X_i and each C_j , normalized by $t \cdot u$, reflecting the proportion of changed elements across time and channels. With m_3 , we aim to minimize the distance between X_i and each C_j . This distance can take the form of L1 norm, L2 norm or other distance metrics. A constraint ensures that each $C_j \in C(X_i)$ is classified as c by f with probability at least $\theta \in [0, 1]$.

If a solution is found within the time window, the subsequence length ℓ is reduced by one, and the process repeats until no further solution is found. CONFETTI uses NSGA-III for optimization, chosen for its ability to maintain solution diversity via reference points generated using the Das-Dennis method (Das and Dennis 1998) in a 3-objective space with k partitions. The initial population is generated via Binary Random Sampling, and the algorithm applies Two-Point Crossover and Bit-Flip Mutation. It runs for a fixed number of generations to evolve toward an optimal solution. Each generation produces a population P_g and the values of the objectives of $S_g = \{\forall x \in P_g | [m_1, m_2]\}$ that are added to the sets $C(X_i)$ and $S(C(X_i))$ respectively.

Lastly, CONFETTI includes a weighting parameter $\alpha \in [0, 1]$ that allows users to balance the relative importance of confidence and sparsity when selecting the best CE. We combine these two objectives using a weighted sum, applying the vector $[\alpha, 1 - \alpha]$. The CE $C_j \in C(X_i)$ that obtains the highest weighted score is selected as the best candidate for instance X_i . Ultimately, CONFETTI returns the full set $C(X_i)$ and the best candidate.

Correctness of the approach. Our method ensures that each $C_j \in C(X_i)$ is generated through controlled optimization, progressively reducing modifications from the initial CE $C_0 \in C(X_i)$. The following theorem shows that every C_j after the naive stage maintains or improves similarity to X_i , measured by Hamming distance.

Theorem 1. *Let f be the classifier, X_i the instance to be explained, and $\theta \in [0, 1]$ a confidence threshold. Suppose*

CONFETTI is executed with these inputs and returns a set of counterfactuals $C(X_i)$. Let $C_0 \in C(X_i)$ denote the initial counterfactual generated during the naive stage (Lines 5–12). Define $\text{Hamming}(C_j, X_i)$ as the number of non-matching values between C_j and X_i across all time steps and channels—that is.

Then, for all $C_j \in C(X_i)$, it holds that $\text{Hamming}(C_j, X_i) \leq \text{Hamming}(C_0, X_i)$

Context. The initial CE C_0 is constructed as $C_0 \leftarrow X_i^{0,s-1} \parallel \text{null}^{s,e} \parallel X_i^{e+1,|X_i|-1}$. If the replaced subsequence spans $\ell = e - s + 1$ timesteps and the series has d channels, then all $\ell \cdot d$ entries in that region differ from X_i , giving $\text{Hamming}(C_0, X_i) = \ell \cdot d$

Proof. Throughout CONFETTI’s Algorithm (Algorithm 1) the following properties hold:

1. **Window bound.** At every entry to the NSGA-III optimisation loop the current subsequence length satisfies $k \leq \ell$
Reason: the outer loop sets $k \leftarrow \lfloor \ell/2 \rfloor$; afterwards the binary-search update $h \leftarrow k - 1$ can only *decrease* the upper bound ℓ .
2. **Window locality.** For a candidate C_j produced by BINARY SAMPLING, CROSSOVER, or MUTATION we have

$$\text{Hamming}(C_j, X_i) \leq k \cdot d.$$

Reason: These operators modify *only* the current window $X_i^{s,e}$ of length k , affecting at most k time steps and $k \cdot d$ time–channel entries.

Now choose any CE C_j that the algorithm finally returns. Because C_j comes from some generation of the NSGA-III loop, Property (2) yields $\text{Hamming}(C_j, X_i) \leq k \cdot d$. By Property (1) we further have that $k \leq \ell$; chaining the two inequalities, we obtain:

$$\text{Hamming}(C_j, X_i) \leq \ell \cdot d = \text{Hamming}(C_0, X_i).$$

□

5 Experimental Evaluation

To assess the ability of CONFETTI to solve its problem statement, we focus on answering the following research questions in the experiments:

- (RQ1) Can CONFETTI *effectively* generate significant counterfactuals for MTS (Section 5.1)?
- (RQ2) Does CONFETTI’s naive stage contribute significantly to the quality of the generated counterfactuals (Section 5.2)?
- (RQ3) How *sensitive* is CONFETTI to the configuration of its internal parameters and components (Section 5.3)?
- (RQ4) How *efficient* is CONFETTI compared to current alternatives in the state-of-the-art (Section 5.1)?

Models & Datasets The experiments are conducted with (i) two models: Fully Convolutional Network (FCN), Residual Network (ResNet) implemented in (Wang, Yan, and Oates 2017b), which have demonstrated strong performance on MTS classification and support CAM extraction (Section 4); (ii) seven datasets from the Multivariate TSML Archive (Bagnall et al. 2018) (see Table 2), selected to cover diverse time series lengths, channel counts, and class numbers.

Table 2: Dataset Properties

Dataset	Length	Dimensions	Classes
ArticularlyWordRecognition	144	9	25
BasicMotions	100	6	4
Epilepsy	207	3	4
ERing	65	4	6
Libras	45	2	15
NATOPS	51	24	6
RacketSports	30	6	4

Note. We refer to ArticularlyWordRecognition as AWR in the subsequent text.

Evaluation criteria CE quality is assessed using six criteria and eight metrics, as follows:

1. **Sparsity (SPA)**: For each instance, measures the proportion of unchanged time-channel entries between a CE and its original, averaged over all instances:

$$\frac{1}{|X_i|} \sum_{X_i \in X} \sum_{C_j \in C(X_i)} \left(1 - \frac{\text{Hamming}(X_i, C_j)}{t \cdot d} \right)$$

Dividing by the number of time steps t and channels d normalizes the Hamming distance to $[0, 1]$; higher values indicate sparser counterfactuals.

2. **Counterfactual Confidence (CONF)**: For each instance, measures how much its counterfactuals reduce the model’s confidence in the original class $f(X_i)$, averaged over all instances. Higher values indicate lower confidence in the original class:

$$\frac{1}{|X_i|} \sum_{X_i \in X} \sum_{C_j \in C(X_i)} (1 - P(f(C_j) = f(X_i)))$$

3. **Plausibility**: We use the **yNN** score (Pawelczyk et al. 2021), adapted to MTS as in (Höllig, Kulbach, and Thoma 2022), to measure how well the predicted class of a CE $C_j \in C(X_i)$ is supported by its local neighborhood. For each C_j , we compute its $k = 5$ nearest neighbors using DTW and average the yNN score across all instances. Higher values indicate stronger support from the data distribution.

$$\frac{1}{|X|} \sum_{X_i \in X} \sum_{C_j \in C(X_i)} 1 - \frac{\sum_{R_h \in kNN(C_j)} \mathbf{1}_{[f(C_j) = f(R_h)]}}{k}$$

4. **Proximity**: Evaluated in three ways: (i) l_1 measures overall deviation between $C_j \in C(X_i)$ and X_i without regard

to location or size of changes; (ii) l_2 penalizes larger differences more heavily; (iii) DTW assesses temporal similarity. Lower values indicate better proximity.

5. **Coverage (COV)**: Proportion of test instances X for which at least one CE is found (the higher the better):

$$\frac{1}{|X|} \sum_{X_i \in X} \mathbf{1}_{C(X_i) \neq \emptyset}$$

6. **Validity (VAL)**: Proportion of CEs that successfully change the original prediction:

$$\frac{1}{|X|} \sum_{X_i \in X} \frac{1}{|C(X_i)|} \sum_{C_j \in C(X_i)} \mathbf{1}_{[f(X_i) \neq f(C_j)]}$$

Baselines CONFETTI is compared with three state-of-the-art methods described in Section 2: CoMTE (Ates et al. 2021), SETS (Bahri, Boubrahimi, and Hamdi 2022), and TSEvo (Höllig, Kulbach, and Thoma 2022). We did not include AB-CE (Li et al. 2023) due to the lack of publicly available code, which prevented a reproducible implementation. LASTS (Spinnato et al. 2022) was also excluded because it produces multiple types of explanations (counterfactuals, rules, and a surrogate model) rather than focusing solely on counterfactuals, making a fair and direct comparison with our counterfactual-only evaluation framework impractical.

The parameter configurations are as follows: CoMTE is configured to use one distractor, with a maximum of 100 attempts and 100 iterations; SETS is configured with a minimum shapelet length defined as either three time steps or one-tenth of the series length (whichever is greater), and a maximum shapelet length set to either half of the series length or one time step longer than the minimum (whichever is greater), ensuring shapelets scale appropriately with series size; finally, TSEvo is limited to a maximum of 100 epochs. All remaining parameters were kept at their default values provided by TSInterpret.

Implementation Setup CONFETTI was implemented in Python 3.12 using Keras 3.8.0 for modeling and sktime 0.36.0 for data handling. Baselines were built with TSInterpret 0.4.7. Experiments ran on a MacBook Pro (Apple M4 Max, macOS Sequoia 15.5, 36GB RAM). We used the official UEA train/test splits from sktime. Minor edits were made to TSInterpret to fix a CoMTE threading bug.

5.1 Benchmark

To ensure fair comparison, we first evaluate the two metrics optimized by CONFETTI, limiting comparisons to methods that explicitly target each metric (Table 1). We compare *Counterfactual Confidence* with CoMTE and SETS, and *Sparsity* with CoMTE and TSEvo. We then report all eight metrics across all methods. Finally, we assign ranks (in parentheses) via pairwise method comparisons per dataset. A method ranks higher only if it outperforms another on every dataset (excluding missing values), with statistical significance confirmed by a paired Wilcoxon signed-rank test (Wilcoxon 1992) at $\alpha = 0.05$; otherwise, ranks are shared.

Counterfactual Confidence Table 3 presents the benchmark results for *Counterfactual Confidence*. Across all seven datasets, CONFETTI consistently outperforms CoMTE and SETS. With a stricter threshold ($\theta = 0.95$), it achieves a mean confidence of 0.98, exceeding the best baseline (CoMTE, 0.86) by 0.02–0.22 across datasets—most notably on NATOPS, improving from 0.76 to 0.98.

Table 3: Confidence scores across methods for each dataset. Values are averaged over models. Ranks appears in parentheses in the column headers

Dataset	CoMTE (2)	SETS (2)	CONFETTI	
			$\alpha=0.5$	$\alpha=0.5$
			$\theta=0.51$ (2)	$\theta=0.95$ (1)
ARW	0.953	0.940	0.726	0.978
BasicMotions	0.917	0.487	0.611	0.965
ERing	0.701	0.766	0.770	0.981
Epilepsy	0.837	0.778	0.636	0.972
Libras	0.952	0.820	0.719	0.973
NATOPS	0.755	*	0.709	0.976
RacketSports	0.932	0.780	0.756	0.980

Rank 1 values are highlighted in bold.

* Failed to produce results.

Table 4: Sparsity scores across methods for each dataset. Values are averaged over models. Ranks appears in parentheses in the column headers.

Dataset	CoMTE (3)	TSEvo (4)	CONFETTI	
			$\alpha=0.0$	$\alpha=0.5$
			$\theta=0.51$ (1)	$\theta=0.51$ (2)
ARW	0.731	0.002	0.926	0.912
BasicMotions	0.486	0.003	0.822	0.798
ERing	0.681	0.029	0.913	0.876
Epilepsy	0.461	0.011	0.822	0.802
Libras	0.247	0.033	0.850	0.794
NATOPS	0.719	0.001	0.880	0.861
RacketSports	0.562	0.012	0.942	0.912

Note: Rank 1 values are highlighted in bold.

The default setting ($\theta = 0.51$) further demonstrates CONFETTI’s adaptability. While its mean confidence (0.70) is slightly below CoMTE’s, it remains competitive on four datasets and clearly outperforms SETS on BasicMotions, ERing, Epilepsy, and RacketSports. Thus, $\theta = 0.95$ is preferable for high-stakes scenarios requiring high reliability, whereas $\theta = 0.51$ offers a solid baseline for general use.

Sparsity Table 4 reports results for *sparsity*, with ranks assigned as in Table 3. CONFETTI again outperforms both CoMTE and TSEvo on all datasets. With full emphasis on sparsity ($\alpha = 0.0$), it reaches an average score of 0.88, clearly ahead of CoMTE (0.56) and TSEvo (0.01). Even

the balanced setting ($\alpha = 0.5$) performs strongly, averaging 0.85. These results show that CONFETTI offers precise control over the sparsity–confidence trade-off, with $\alpha = 0.0$ suited for minimizing feature changes and $\alpha = 0.5$ as a robust general setting.

Evaluation Metrics Overview Table 5 and 6 report average scores for all methods using FCN and ResNet, along with ranking (in parenthesis). Best-performing methods are shown in bold. Multiple bold entries indicate no significant difference.

All CONFETTI variants consistently achieved 100% coverage (COV), matching CoMTE and TSEvo, meaning CEs were generated for every instance. In terms of validity (VAL), CONFETTI outperformed TSEvo and SETS and, while tied in rank with CoMTE, was the only method to reach a perfect score of 1 across both models and all datasets, indicating every CE changed the model prediction. By contrast, CoMTE ranged between 0.79–0.80 and 0.91–0.93.

Table 5: Performance of CE methods on FCN for all evaluation metrics. Values represent average scores across all datasets. Ranking position are in parentheses

Metric	Baselines			CONFETTI		
	CoMTE	SETS	TsEVO	$\alpha=0.0$ $\theta=0.51$	$\alpha=0.5$ $\theta=0.51$	$\alpha=0.5$ $\theta=0.95$
COV	100 (1)	94.17 (1)	100 (1)	100 (1)	100 (1)	100 (1)
VAL	0.93 (1)	0.77 (2)	0.80 (2)	1.00 (1)	1.00 (1)	1.00 (1)
CONF	0.86 (2)	0.76 (3)	0.80 (2)	0.59 (4)	0.69 (3)	0.97 (1)
SPA	0.54 (4)	0.02 (5)	0.01 (5)	0.88 (1)	0.85 (2)	0.81 (3)
L_1	283.55 (4)	923.58 (5)	954.54 (5)	99.76 (1)	111.80 (2)	146.19 (3)
L_2	27.24 (4)	56.79 (5)	56.50 (5)	16.01 (1)	16.83 (2)	19.78 (3)
DTW	26.49 (4)	48.16 (5)	49.19 (5)	15.40 (1)	16.09 (2)	18.93 (3)
yNN	0.99 (1)	0.99 (1)	0.99 (1)	0.99 (1)	0.99 (1)	0.99 (1)

Table 6: Performance of CE methods on ResNet for all evaluation metrics. Values represent average scores across all datasets. Ranking position are in parentheses.

Metric	Baselines			CONFETTI		
	CoMTE	SETS	TsEVO	$\alpha=0.0$ $\theta=0.51$	$\alpha=0.5$ $\theta=0.51$	$\alpha=0.5$ $\theta=0.95$
COV	100 (1)	93.24 (1)	100 (1)	100 (1)	100 (1)	100 (1)
VAL	0.91 (1)	0.76 (2)	0.79 (2)	1.00 (1)	1.00 (1)	1.00 (1)
CONF	0.87 (1)	0.76 (2)	0.79 (2)	0.59 (3)	0.71 (2)	0.98 (1)
SPA	0.57 (4)	0.02 (5)	0.01 (5)	0.88 (1)	0.85 (2)	0.82 (3)
L_1	269.82 (4)	920.46 (5)	954.39 (5)	96.50 (1)	106.08 (2)	121.78 (3)
L_2	25.90 (3)	56.98 (4)	56.22 (4)	14.94 (1)	15.61 (2)	16.47 (3)
DTW	25.25 (4)	47.72 (5)	48.62 (5)	14.22 (1)	14.79 (2)	15.68 (3)
yNN	0.99 (1)	0.99 (1)	0.99 (1)	0.99 (1)	0.99 (1)	0.99 (1)

Regarding sparsity, CONFETTI clearly outperforms all baselines. All tested configurations achieved significantly higher sparsity scores (0.81–0.88) than CoMTE (0.56), and vastly exceeded SETS and TSEvo (both < 0.02). Sparsity remained consistently high across configurations, confirm-

ing CONFETTI’s consistently in generating minimal yet effective CEs.

CONFETTI also excelled in proximity metrics (l_1 , l_2 , and DTW), achieving the lowest average distances across all settings and outperforming baselines by wide margins. Notably, proximity was not explicitly optimized, yet the CEs remained close to the original X_i , enhancing interpretability. Since values are either retained from X_i or replaced with those from nun , similar numbers of changes across methods result in comparable DTW distances.

Finally, all methods (incl., CONFETTI) achieve high yNN scores (0.99), confirming plausibility within the data distribution. Together, these findings answer **RQ1** and **RQ4** positively: CONFETTI consistently generates valid, interpretable counterfactuals for MTS (RQ1) and outperforms or matches all baselines across metrics and datasets, demonstrating strong practical value (RQ4).

Execution Times Lastly, we also show the execution times of the different methods across both models. We present our results in Table 7.

Table 7: Execution time comparison (in seconds) of CE methods for FCN and ResNet. Values represent mean and standard deviation across all datasets.

Methods	FCN	ResNet
CoMTE	341.24 $\pm$ 77.52	308.73 $\pm$ 93.42
CONFETTI $\alpha = 0.0$	204.47 $\pm$ 118.07	335.37 $\pm$ 201.95
CONFETTI $\alpha = 0.5$	199.55 $\pm$ 115.59	339.00 $\pm$ 208.56
CONFETTI $\theta = 0.95$	223.85 $\pm$ 120.57	368.93 $\pm$ 221.73
SETS	84.01 $\pm$ 88.48	89.02 $\pm$ 95.13
TsEVO	8625.38 $\pm$ 2509.38	9126.82 $\pm$ 2615.85

As shown in the table, SETS is approximately an order of magnitude faster than the other methods, while TsEVO is roughly an order of magnitude slower. CONFETTI and CoMTE fall in the middle with comparable execution times, although CoMTE exhibits much lower variance across datasets. In contrast, CONFETTI shows a larger standard deviation, which is expected given its optimization process. The window-based search depends on the length and structure of each time series, making execution time more sensitive to dataset characteristics. This reflects a broader trade-off between methods that operate at the subsequence or timestep level and those that work at the channel level. Nevertheless, the fact that our subsequence-based approach remains competitive with CoMTE and significantly outperforms its most similar competitor TsEVO highlights the strength and practical relevance of CONFETTI.

5.2 Ablation Study

To respond to RQ2 we performed an ablation study of the naive stage of CONFETTI. We carried out this experiment using the default hyperparameter settings ($\alpha=0.5$, $\theta = 0.51$) in two different versions: one in which we provided CAM weights and the other in which we did not provide them. We evaluated the generated CE across all metrics and datasets and performed a Wilcoxon test (Wilcoxon 1992) with Holm

correction to identify whether there are any significant differences for each metric. We present our results in Table 8.

Table 8: Performance comparison of CONFETTI with and without Naive Stage for FCN and ResNet. Results are averaged across all datasets. A parenthesis (*) indicates a statistically significant difference (Wilcoxon + Holm).

Metric	FCN		ResNet	
	Normal	Ablation	Normal	Ablation
COV	100.00	99.40	100.00	100.00
SPA	0.84	0.81	0.84	0.80
CONF	0.58	0.58	0.55	0.55
VAL	1.00	1.00	1.00	1.00
L_1	107.01 (*)	145.22 (*)	110.74 (*)	168.13 (*)
L_2	15.65 (*)	19.61 (*)	15.69 (*)	20.59 (*)
DTW	14.96 (*)	18.98 (*)	14.87 (*)	19.84 (*)
yNN	0.99	0.99	0.99	0.99
Exec. Time	162.76 (*)	221.44 (*)	302.84 (*)	384.31 (*)

First, we noticed that we did not find significant differences in coverage, sparsity, confidence, validity, and plausibility. This result is beneficial because we prove that CONFETTI can achieve great results for models that do not have any feature importance mechanisms, and thus its application is relevant as a model-agnostic method. However, we found significant differences in performance across all proximity distance metrics, meaning that CEs produced with the aid of the naive stage resemble closer to the original instances, proving that using feature importance weights has indeed a positive impact. Furthermore, we also observed a significant difference in execution time. This improvement comes from the reduced size of the initial optimization windows. In the naive stage, the largest subsequence is located much more quickly, whereas the ablation variant always begins its optimization from a subsequence length fixed at half of the time series, resulting in a noticeably slower search.

5.3 Sensitivity Analysis

To evaluate the impact of parameters θ and α , we conducted two sensitivity analyses: one varying α with fixed $\theta=0.51$, and another varying θ with fixed $\alpha=0.5$. Specifically, we tested $\alpha \in [0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0]$ and $\theta \in [0.55, 0.65, 0.75, 0.85, 0.95]$. Figure 2 shows how Sparsity, Confidence, and Proximity metrics respond to these parameter changes.

Balancing Optimization Objectives We first examine the effect of α , which balances prediction confidence and sparsity during optimization. Lower α favors sparsity; higher values prioritize confidence.

CONFETTI’s sparsity remains stable for intermediate α values, with notable changes only at the extremes ($\alpha = 0$ or 1). Thus, the optimization is sensitive to α mainly when it is pushed to its limits.

Effect of Confidence Requirement Figure 2 (subplot iii) shows that increasing α steadily improves *confidence*, from just above 0.60 to ≈ 0.80 at $\alpha = 1$. A similar trend appears

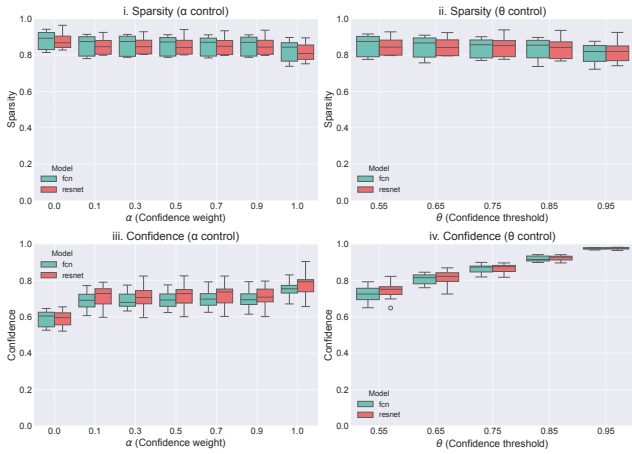


Figure 2: Effect of parameter configurations on counterfactual generation. Panels show: (i) Sparsity as α varies; (ii) Sparsity as θ varies; (iii) Confidence as α varies; and (iv) Confidence as θ varies.

in the sensitivity to θ : sparsity remains stable, while confidence increases with higher thresholds. Even at the lowest setting ($\theta = 0.55$), CONFETTI produces CEs averaging ≈ 0.75 in confidence, well above the required minimum. These results suggest that boosting prediction confidence generally requires modifying only a few time steps, reinforcing the benefit of CAM-guided perturbations. Notably, α has no effect on *Coverage*: CONFETTI generates CEs for all instances across datasets, regardless of the weighting. This invariance is by design, as the CE search process is independent of α . Similarly, CONFETTI’s consistency to increasing θ likely reflects properties of the classifier f , which yields a valid *num* for every X_i . These findings answer **RQ3**: CONFETTI is robust to parameter choices. θ influences confidence but not sparsity, and α only affects outcomes at extremes. Across a broad range of settings, CONFETTI reliably generates confident, sparse, and valid CEs with minimal tuning.

5.4 Fitness Function Analysis

We conducted an empirical experiment to verify that our proposed fitness function was the right approach. To this end, we executed CONFETTI’s CE search using all combinations of objectives. The tested configurations were Confidence-Proximity (CO_PR), Confidence-Sparsity (CO_SP), Sparsity-Proximity (SP_PR), and Confidence-Sparsity-Proximity (CO_SP_PR). After evaluating all resulting CEs, we applied the Friedman test to assess whether ranking differences were statistically significant. Whenever the Friedman test (Friedman 1937) indicated significance at $p \leq 0.05$, we followed with the Nemenyi test (Nemenyi 1963) to determine which specific rankings differed. Only configurations with a Nemenyi p value below 0.05 were considered significantly different. The corresponding critical difference diagram is shown in Figure 3.

As shown in Figure 3, we first note that we do not report coverage, validity, or plausibility. We found no significant

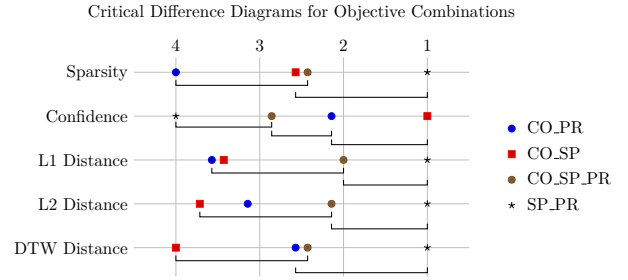


Figure 3: Critical Difference Diagram showing the average ranking (lower is better) of different objective combinations on counterfactual generation metrics. The lines group combinations that are not significantly different from each other. The different combinations are i) Confidence-Proximity (CO_PR); ii) Confidence-Sparsity (CO_SP); iii) Confidence-Sparsity-Proximity (CO_SP_PR); and iv) Sparsity-Proximity (SP_PR).

cant ranking differences for these metrics; all configurations reached identical scores (100%, 1.0, and 0.99). For sparsity, we observed a significant difference between CO_PR and SP_PR, driven by their consistent ranking of 4th and 1st respectively across all datasets. This is expected since SP_PR is the only configuration that does not incorporate confidence. However, once sparsity is included in the objective function (e.g., CO_SP), we no longer observe a significant difference with respect to SP_PR.

For confidence, we detected significant differences between CO_SP and SP_PR, as well as between the full configuration CO_SP_PR and SP_PR, but not between CO_PR and SP_PR. This indicates that pairing confidence exclusively with sparsity or proximity does not substantially affect confidence performance, whereas including all three objectives leads to a slight decrease.

Regarding proximity metrics, we observe a consistent pattern in which SP_PR differs significantly from both CO_SP and CO_PR (with the exception of DTW distance). However, no significant differences are found once all three objectives are combined in the fitness function.

At first glance, SP_PR might appear to be the best option, but its poor confidence performance makes it unsuitable, given the importance of confidence in CE desiderata. The next best configurations for confidence were CO_PR and CO_SP_PR. Between these two, CO_PR is significantly different from SP_PR in sparsity. Overall, our results indicate that including all three objectives provides the most balanced alternative. Although CO_SP_PR never achieved the top rank, it consistently remained close enough that its differences were not statistically significant.

5.5 Choice of Proximity Distance

As discussed in Section 4.5, the objective m_3 can be instantiated with any distance function. To assess whether the choice of distance metric influences performance, we ran CONFETTI using several alternatives: Euclidean distance, Canonical Time Warping (CTW) (Zhou and Torre 2009),

Dynamic Time Warping (DTW) (Sakoe and Chiba 2003), and Soft-DTW (Cuturi and Blondel 2017). Following the same procedure described in Section 5.4, we applied a Friedman test to the resulting performance metrics to evaluate whether any metric led to statistically different rankings. In this analysis, we found no significant differences across any proximity metric for any of the CE performance measures. The results are presented in Table 9.

Table 9: Performance of CONFETTI given different distance metrics for proximity for all evaluation metrics. Values represent mean and standard deviation across all datasets.

Metric	ctw	dtw	euclidean	softdtw
COV	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00
VAL	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00
SPA	0.85 $\pm$ 0.05	0.85 $\pm$ 0.05	0.84 $\pm$ 0.07	0.84 $\pm$ 0.06
CONF	0.59 $\pm$ 0.08	0.59 $\pm$ 0.08	0.58 $\pm$ 0.05	0.59 $\pm$ 0.09
L_1	120.97 $\pm$ 133.22	120.97 $\pm$ 133.22	101.99 $\pm$ 110.69	120.55 $\pm$ 134.17
L_2	17.41 $\pm$ 18.62	17.41 $\pm$ 18.62	15.08 $\pm$ 15.40	17.46 $\pm$ 18.69
DTW	15.38 $\pm$ 17.37	15.38 $\pm$ 17.37	14.47 $\pm$ 15.20	15.59 $\pm$ 17.59
yNN	0.99 $\pm$ 0.01	0.99 $\pm$ 0.01	0.99 $\pm$ 0.00	0.99 $\pm$ 0.01

6 Conclusion

We introduced a novel method for generating CEs for MTS that effectively balances sparsity, proximity, and prediction confidence while ensuring plausibility and validity by design. Our approach leverages prior knowledge through CAM as an attention mechanism, enabling modifications of the most relevant subsequences. The experimental results confirmed that our method successfully generated valid CEs for all instances tested. Furthermore, it significantly outperformed existing state-of-the-art methods in sparsity, proximity, and also in counterfactual confidence, particularly at high values of the parameter θ . Sensitivity analyzes demonstrate that modifying even a small number of time steps within the most relevant subsequence can substantially enhance prediction confidence while maintaining high sparsity. Lastly, our ablation study shows that CONFETTI maintains almost the same performance even when attention mechanisms are removed, which confirms that the method can operate in a model-agnostic setting. However, this comes at the cost of worse proximity values and longer execution times.

7 Acknowledgements

This work is supported in whole or in part by the National Research Fund, Luxembourg. Grant references 18959931 and 18435508. The authors acknowledge the contributors to the UEA and UCR Time Series Repositories. Their work in creating, curating, and sharing these datasets has been essential for this research.

References

Adadi, A.; and Berrada, M. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6: 52138–52160.

- Ates, E.; Aksar, B.; Leung, V. J.; and Coskun, A. K. 2021. Counterfactual explanations for multivariate time series. In *2021 international conference on applied artificial intelligence (ICAPAI)*, 1–8. IEEE.
- Bagnall, A.; Dau, H. A.; Lines, J.; Flynn, M.; Large, J.; Bostrom, A.; Southam, P.; and Keogh, E. 2018. The UEA multivariate time series classification archive, 2018. arXiv:1811.00075.
- Bahri, O.; Boubrahimi, S. F.; and Hamdi, S. M. 2022. Shapelet-based counterfactual explanations for multivariate time series. *arXiv preprint arXiv:2208.10462*.
- Cuturi, M.; and Blondel, M. 2017. Soft-dtw: a differentiable loss function for time-series. In *International conference on machine learning*, 894–903. PMLR.
- Dandl, S.; Molnar, C.; Binder, M.; and Bischl, B. 2020. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, 448–469. Springer.
- Das, I.; and Dennis, J. E. 1998. Normal-boundary intersection: A new method for generating the Pareto surface in non-linear multicriteria optimization problems. *SIAM journal on optimization*, 8(3): 631–657.
- 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. arXiv:2009.13211.
- Friedman, M. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200): 675–701.
- Guidotti, R. 2024. Counterfactual Explanations and How to Find Them: Literature Review and Benchmarking. *Data Mining and Knowledge Discovery*, 38(5): 2770–2824.
- Hamming, R. W. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2): 147–160.
- 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)*, 29–36. IEEE.
- Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4): 917–963.
- Khan, M.; Wang, H.; Nguenbaye, A.; and Elfatyany, A. 2023. End-to-end multivariate time series classification via hybrid deep learning architectures. *Personal and Ubiquitous Computing*, 27(2): 177–191.
- Li, P.; Bahri, O.; Boubrahimi, S. F.; and Hamdi, S. M. 2023. Attention-based counterfactual explanation for multivariate time series. In *International Conference on Big Data Analytics and Knowledge Discovery*, 287–293. Springer.
- Liu, Y.; Gong, C.; Yang, L.; and Chen, Y. 2020. DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction. *Expert Systems with Applications*, 143: 113082.

- Navas-Palencia, G. 2021. Optimal counterfactual explanations for scorecard modelling. *arXiv preprint arXiv:2104.08619*.
- Nemenyi, P. B. 1963. *Distribution-free multiple comparisons*. Princeton University.
- Pawelczyk, M.; Bielawski, S.; Heuvel, J. v. d.; Richter, T.; and Kasneci, G. 2021. Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *arXiv preprint arXiv:2108.00783*.
- Ruiz, A. P.; Flynn, M.; Large, J.; Middlehurst, M.; and Bag-nall, A. 2021. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2): 401–449.
- Sakoe, H.; and Chiba, S. 2003. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1): 43–49.
- Schäfer, P.; and Leser, U. 2018. Multivariate Time Series Classification with WEASEL+MUSE. *arXiv:1711.11343*.
- Spinnato, F.; Guidotti, R.; Monreale, A.; Nanni, M.; Pedreschi, D.; and Giannotti, F. 2023. Understanding Any Time Series Classifier with a Subsequence-based Explainer. *ACM Transactions on Knowledge Discovery from Data*, 18(2): 1–34.
- Spinnato, F.; Guidotti, R.; Nanni, M.; Maccagnola, D.; Paciello, G.; and Farina, A. B. 2022. Explaining crash predictions on multivariate time series data. In *International Conference on Discovery Science*, 556–566. Springer.
- 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.
- Wang, Z.; Yan, W.; and Oates, T. 2017a. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 1578–1585.
- Wang, Z.; Yan, W.; and Oates, T. 2017b. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, 1578–1585. IEEE.
- Wilcoxon, F. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*, 196–202. Springer.
- Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2921–2929.
- Zhou, F.; and Torre, F. 2009. Canonical time warping for alignment of human behavior. *Advances in neural information processing systems*, 22.