COUNTERFACTUAL EXPLANATIONS FOR TIME SERIES DATA VIA REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Counterfactual (CF) explanations are a powerful tool in Explainable AI (XAI), providing actionable insights into how model predictions could change under minimal input alterations. Generating CFs for time series, however, remains challenging: existing optimization-based methods are often instance-specific, impose restrictive constraints, and struggle to ensure both validity and plausibility. To address these limitations, we propose a reinforcement learning (RL) framework for counterfactual explanation in time series. Our actor–critic agent learns a policy in the latent space of a pre-trained autoencoder, enabling the generation of counterfactuals that balance validity and plausibility without relying on rigid handcrafted constraints. Once trained, the RL agent produces counterfactuals in a single forward pass, ensuring scalability to large datasets. Experiments on diverse benchmarks demonstrate that our approach generates valid and plausible counterfactuals, offering a reliable alternative to existing methods.

1 INTRODUCTION

Understanding the predictions of machine learning models is especially important in high-stakes areas such as healthcare monitoring, financial risk assessment, and space weather forecasting (Loh et al., 2022; Bharati et al., 2023; Černevičienė & Kabašinskas, 2024; Bussmann et al., 2020; Camps-Valls et al., 2025). In these domains, decisions informed by models can have significant consequences for human health, safety, or large-scale operations. While modern machine learning models, particularly deep neural networks, achieve remarkable accuracy, their black-box nature raises concerns about transparency, accountability, and trustworthiness (Dwivedi et al., 2023). Stakeholders in sensitive applications need more than predictions — they also need to understand why a decision was made and what factors influenced it.

Counterfactual explanations (CFEs) address this by asking a precise "what-if" question: what minimal, plausible changes to the input would flip the model's prediction (Wachter et al., 2017)? By specifying the smallest set of feature changes sufficient to alter the outcome, CFEs indirectly reveal which features are most influential and can be mapped to actionable recourse—that is, concrete steps a user could take to pursue a different decision. For example, in a loan application scenario, a CFE might show that if the applicant's annual income were slightly higher or their credit utilization ratio slightly lower, the model would have approved the loan instead of rejecting it. Such insights both clarify model reasoning and provide applicants with meaningful guidance for improving future outcomes.

While counterfactuals have been widely studied in tabular and image domains (Wachter et al., 2017; Mothilal et al., 2020; Looveren & Klaise, 2021; Guidotti, 2024; Verma et al., 2024), developing meaningful CFEs for time series remains comparatively less explored. For images, counterfactual changes are often intuitive and directly interpretable: a small alteration to a shape, color, or texture can be visually inspected, and the plausibility of the change can be judged at a glance. For tabular data, constraints on features are also relatively straightforward to define—certain attributes like age or gender are immutable, while others, such as income or credit score, can be adjusted within realistic ranges. In contrast, time series data present unique difficulties. The constraints that define a "plausible" modification are much less obvious, as sequences must preserve temporal dependencies and dynamic patterns that cannot be easily judged by visual inspection or simple feature rules. Naive changes often produce unrealistic patterns, and even optimization-based methods—which minimize

a mix of distance loss, prediction loss, and constraints to keep counterfactuals contiguous—can still generate implausible results because the modifications do not correspond to realistic temporal changes (Delaney et al., 2021; Li et al., 2022b). Moreover, these optimization methods typically operate on one instance at a time, making them slow and computationally expensive for large or high-dimensional datasets (Filali Boubrahimi & Hamdi, 2022; Li et al., 2022a). Together, these challenges make it difficult to balance plausibility and efficiency, limiting the practicality of CFEs for time series in large-scale applications.

To overcome these challenges, we propose a reinforcement learning (RL) framework for counterfactual explanation in time series classification. Instead of optimizing each instance separately, we frame counterfactual generation as a sequential decision-making process and adopt a Deep Deterministic Policy Gradient (DDPG) setup. In this framework, the actor network generates perturbations in the latent space of time series to form counterfactuals, while the critic network evaluates their quality. The training is guided by two key signals: (1) a reward when the counterfactual flips the classifier's prediction, which teaches the actor to generate valid counterfactuals, and (2) a penalty based on the distance between the counterfactual and the original instance (an L1+L2 proximity loss), which is added directly to the actor's loss to discourage unrealistic or overly large changes. By combining these signals, the actor learns to balance prediction flipping with proximity to the original input. Once trained, the actor can produce counterfactuals in batches, making the approach more scalable than traditional per-instance optimization.

The contributions of this paper are as follows: 1) we propose an RL-based approach that generates counterfactuals in batches, making it more scalable compared to traditional instance-based optimization methods; 2) instead of applying explicit constraints to raw time series data, we perturb in the latent space and use an L1+L2 distance penalty to keep the counterfactuals close to the original input, which helps maintain realistic structure; 3) the framework can be applied to any classifier, as it only requires access to the model's predictions (not gradients). The critic in the DDPG setup is trained against observed rewards (prediction flips and distance penalties), rather than relying on classifier gradients, which makes the method applicable even to black-box models; 4) our approach opens the door to leveraging RL techniques (e.g., policy learning, exploration strategies) for explanation, offering a more flexible alternative to handcrafted optimization objectives.

2 Related Work

Counterfactual explanations (CFEs) have received increasing attention because of their ability to provide intuitive and actionable insights into machine learning predictions. By identifying minimal changes that alter a model's decision, CFEs help users understand model behavior and explore alternative outcomes. Such explanations are especially valuable in high-stakes applications, where transparency and interpretability are as important as accuracy.

Early efforts in time series CFEs relied on instance-based heuristics, producing counterfactuals by perturbing each instance individually, often through segment replacement. For example, NG (Delaney et al., 2021) and MG-CF (Li et al., 2022b) construct counterfactuals by altering or replacing salient subsequences drawn from the data (e.g., CAM-highlighted regions, class motifs). Their main strength is interpretability, since the modifications are grounded in real observed patterns that domain experts can readily relate to. In addition, because they reuse real subsequences, these methods often produce results that appear plausible at the local level. However, they depend heavily on the quality of saliency or motif mining and can struggle with validity (failing to consistently flip the label) and global coherence (as a swapped segment may not integrate smoothly with the rest of the sequence), which reduces their robustness across datasets.

Building on this idea, optimization-based approaches formulate counterfactual generation as an optimization problem for each input instance. The classic formulation by Wachter et al. (2017), originally introduced for tabular data, has since been widely adopted as a baseline in time series studies (Li et al., 2022b;a; Bahri et al., 2022a). Wachter typically minimizes a loss that balances prediction validity (ensuring the label flips) with proximity to the original instance, often measured by L1 or L2 distance. While this formulation is simple and intuitive, proximity constraints alone cannot guarantee temporal plausibility, and naïve pointwise perturbations often disrupt sequential dynamics, leading to unrealistic counterfactuals. To address this limitation, constrained optimization variants have been proposed. For example, SG-CF (Li et al., 2022a) and TeRCE (Bahri et al.,

2022b) restrict perturbations to subsequences defined by shapelets or temporal rules, while TimeX (Filali Boubrahimi & Hamdi, 2022) enforces temporal coherence through barycenter averaging. These constraints improve local interpretability and coherence but introduce additional rigidity, as they rely on predefined structures that may not capture global dependencies, and they still suffer from the inefficiency of per-instance optimization.

A related direction uses saliency-guided masks to localize perturbations and improve interpretability. Methods such as CELS (Li et al., 2023) and Info-CELS (Li et al., 2025) learn perturbation masks guided by gradient information and the nearest unlike neighbor, enabling sparse and interpretable modifications without the need for predefined structures or rules. These approaches produce intuitive visual explanations by highlighting learned saliency maps. However, a key limitation is their reliance on the nearest unlike neighbor: counterfactuals are generated by interpolating between the original instance and this neighbor, with the interpolation rate controlled by the saliency map. If the nearest unlike neighbor exhibits temporal shifts or misalignments, the resulting counterfactuals can still be implausible, even if the label flips successfully.

Latent-space perturbation has recently emerged as a promising direction for improving plausibility. Glacier (Wang et al., 2024) perturbs representations in both the input space and autoencoder-derived latent space, guided by saliency explainers such as LimeSegment (Sivill & Flach, 2022). The saliency vectors serve as constraints that determine which time steps should be perturbed, helping to localize modifications and preserve temporal structure. While this approach improves temporal localization, its performance remains highly sensitive to the quality of the saliency explainer. If the saliency vectors highlight noisy or irrelevant regions, the generated counterfactuals may still lack plausibility and robustness.

All of the methods discussed above are fundamentally instance-level approaches, where counterfactuals are generated independently for each input through optimization or perturbation. This limits both scalability and generalization, as the process must be repeated from scratch for every new instance. Interestingly, Samoilescu et al. (2021) were among the first to propose framing counterfactual generation as a sequential decision-making RL process in tabular domains, where an agent learns to apply minimal perturbations that flip the classifier's decision. Despite its promise—being model-agnostic, not requiring gradients, and naturally supporting batch generation—this perspective has been largely overlooked in the context of time series counterfactuals.

Motivated by this gap, we revisit the RL perspective and extend it to time series classification. By framing counterfactual search as a sequential decision process in latent space, we design an RL "agent" that learns policies to generate counterfactuals tailored for time series data. Unlike instance-based optimization methods, once trained, the RL agent can efficiently produce counterfactuals in batches, providing a more scalable and generalizable solution for counterfactual explanation in time series.

3 Preliminary: Reinforcement Learning (RL)

Reinforcement learning (RL) provides a framework for sequential decision-making, where an agent interacts with an environment to maximize rewards (Sutton et al., 1998). At each step, the agent observes a state s, takes an action a according to a policy $\pi(a|s)$, and receives feedback in the form of a reward r and a new state s'. The goal is to learn a policy that maximizes expected returns.

Model-free RL methods do not assume knowledge of the environment's dynamics. Value-based methods learn an action-value function $Q(s, a)$, while policy-based methods directly optimize a parameterized policy. Actor–critic algorithms combine these ideas: the critic estimates $Q(s, a)$, and the actor updates the policy using guidance from the critic. This structure is particularly effective in high-dimensional, continuous control tasks.

Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015) is a widely used actor–critic algorithm designed for continuous action spaces. The actor outputs a deterministic action $a = \mu_\theta(s)$, and the critic estimates its value $Q_\phi(s, a)$. Training is stabilized through experience replay, which reuses past interactions, and target networks, which prevent divergence by slowly tracking the learned networks.

This setup aligns well with counterfactual generation for time series. We frame the problem as a sequential decision process: the agent starts from an input instance and applies incremental perturbations until the classifier's prediction flips. States correspond to the current representation of the instance together with conditioning information (e.g., original and target labels), actions represent perturbations applied to the instance, and rewards capture counterfactual desiderata such as validity, sparsity, and plausibility. The continuous nature of this problem makes DDPG particularly suitable: its actor–critic design enables efficient search, the replay buffer allows effective reuse of costly model queries, and target networks improve training stability. These properties make DDPG a natural backbone for our reinforcement learning framework for counterfactual generation in time series.

4 METHODOLOGY

4.1 PROBLEM STATEMENT

We consider the problem of generating counterfactual explanations for a black-box time series classifier $M : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ denotes the input space of time series and $\mathcal{Y}$ denotes the label set. Given an input sequence $x \in \mathcal{X}$ with predicted label $y_M = M(x)$, and a user-specified target label $y_T \in \mathcal{Y}, y_T \neq y_M$, the goal is to construct a counterfactual $x_{CF} = x + \delta$, where δ is a perturbation vector such that $M(x_{CF}) = y_T$. Ideally, x_{CF} should satisfy three key properties: (i) *validity*, ensuring that the classifier prediction flips to the target class; (ii) *proximity*, requiring the counterfactual to remain close to the original instance under suitable distance metrics such as L_1 or L_2; and (iii) *plausibility*, demanding that the counterfactual lies on or near the data manifold so that it corresponds to a realistic and temporally coherent sequence.

Instead of relying on instance-based optimization, where each counterfactual must be obtained by perturbing the input through the same optimization process for every instance, we employ an actor–critic reinforcement learning framework to learn a policy that generates counterfactuals efficiently. Once trained, the policy produces counterfactuals in batches through a single forward pass, offering greater scalability than per-instance optimization.

Formally, we formulate counterfactual generation as a Markov decision process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R), \quad s_t \in \mathcal{S}, ; a_t \in \mathcal{A}, ; s_{t+1} \sim P(\cdot \mid s_t, a_t), ; r_t = R(s_t, a_t), \tag{1}$$

where $\mathcal{S}$ denotes the state space, with s_t representing the instance at step t; $\mathcal{A}$ is the action space, where a_t corresponds to a perturbation applied to s_t; P defines the transition dynamics that map (s_t, a_t) to the next state s_{t+1}; and R is the reward function that provides feedback from the classifier. In particular, R is defined as a flip-label reward, taking value 1 if the counterfactual candidate $\hat{x}_{CF}$ is classified as the target label y_T, and 0 otherwise. The environment is thus the interaction between the time series instance and the classifier, which together determine both the state transitions and the reward signals.

To handle the high dimensionality and temporal structure of time series, we first train an autoencoder to obtain a lower-dimensional latent representation $z = \mathrm{enc}(x)$. Counterfactual generation is then performed in this latent space, where perturbations are smoother and more structured. The actor network μ_θ takes the latent state z as input and outputs a perturbed embedding z_{CF} corresponding to a counterfactual candidate. This representation is decoded back to the input space, $\tilde{x}_{CF} = \mathrm{dec}(z_{CF})$, which is then evaluated by the classifier to determine the reward.

We adopt a simplified deterministic actor–critic framework inspired by Deep Deterministic Policy Gradient (DDPG). The critic $Q_\phi(s, a)$ estimates the value of applying action a in state s and is trained by regressing directly to the observed (immediate) reward signal:

$$L_{\text{critic}} = \mathbb{E}\Big[\big(Q_\phi(s_t, a_t) - R_t\big)^2\Big]. \tag{2}$$

The actor is trained to maximize the critic's estimate of value through the deterministic policy gradient:

$$L_{\max} = -\,\mathbb{E}\big[Q_\phi(s_t, \mu_\theta(s_t))\big]. \tag{3}$$

To promote similarity between the counterfactual and the original input, we add a proximity regularizer that combines L_1 and L_2 distances:

$$L_{\text{prox}} = \beta \, \|\tilde{x}_{CF} - x\|_1 + \|\tilde{x}_{CF} - x\|_2^2. \tag{4}$$

Algorithm 1 Training procedure

Inputs: M: black-box model; proximity loss weight λ_p; total steps T; update interval U; batch size $|B|$; exploration std. σ; mixing coefficient β=0.5
Outputs: Trained actor network μ for counterfactual generation

1: Load pre-trained encoder enc and decoder dec
2: Initialize actor $\mu(\cdot;\theta_\mu)$ and critic $Q(\cdot;\theta_Q)$
3: Initialize replay buffer $\mathcal{D}$
4: Define reward function $f(\cdot,\cdot)$
5: **for** $t = 1$ to T **do**
6: Sample input time series x and target class y_T
7: $y_M \leftarrow M(x)$; $z \leftarrow enc(x)$
8: $z_{CF} \leftarrow \mu(z, y_M, y_T; \theta_\mu)$
9: $\tilde{z}_{CF} \leftarrow \text{clip}(z_{CF} + \varepsilon, -1, 1)$, $\varepsilon \sim \mathcal{N}(0, 0.1)$
10: $\tilde{x}_{CF} \leftarrow dec(\tilde{z}_{CF})$
11: $R \leftarrow f(M(\tilde{x}_{CF}), y_T)$
12: Store $(z, y_M, y_T, \tilde{z}_{CF}, R)$ in the replay buffer $\mathcal{D}$
13: **if** $t \bmod U = 0$ **then**
14: Sample batch $\mathcal{B}$ of size B from $\mathcal{D}$
15: Update critic by one-step gradient descent using:

$$\nabla_{\theta_Q} \frac{1}{|B|} \sum_{B} \left(Q(z, y_M, y_T, \tilde{z}_{CF}) - R \right)^2$$

16: Recompute $z_{CF} \leftarrow \mu(z, y_M, y_T; \theta_\mu)$, $x_{CF} \leftarrow dec(z_{CF})$

$$L_{\max} = -\tfrac{1}{|B|} \sum_{B} Q(z, y_M, y_T, z_{CF})$$

$$L_{\text{prox}} = \tfrac{1}{|B|} \sum_{B} \left(\beta \, \|x - x_{CF}\|_1 + \|x - x_{CF}\|_2^2 \right) \quad \text{with } \beta = 0.5$$

17: Update actor by one-step gradient descent using:

$$\nabla_{\theta_\mu} \left(L_{\max} + \lambda_p \, L_{\text{prox}} \right)$$

18: **end if**
19: **end for**

The final actor objective is then

$$L_{\text{actor}}^{\text{total}} = L_{\max} + \lambda_{\text{p}} \, L_{\text{prox}}. \tag{5}$$

During training, exploration is encouraged by injecting noise into the actor's proposed perturbations, gradually transitioning from uniform random actions to noisy perturbations around the actor's output. Through this combination, the actor learns to generate counterfactuals that maximize validity while preserving proximity, and the decoder ensures that the resulting sequences remain realistic and temporally coherent.

4.2 Training Workflow

Algorithm 1 summarizes the training pipeline of our reinforcement learning framework for counterfactual generation in time series, which is conducted on the training dataset. The procedure begins by loading a pre-trained encoder-decoder pair (line 1), which defines a structured latent space for applying perturbations. The actor and critic networks are initialized along with a replay buffer, and a reward function f is defined to convert classifier predictions relative to the target label into a reward signal (lines 2–4).

At each training step (lines 5–12), an input time series x and target label y_T are sampled. The input is encoded into a latent representation z, and the actor generates a candidate counterfactual latent z_{CF} conditioned on (z, y_M, y_T). Gaussian noise is added for exploration, and the perturbed latent $\tilde{z}_{CF}$ is clipped before being decoded into a candidate $\tilde{x}_{CF}$. The black-box classifier M evaluates

$\tilde{x}_{CF}$ to produce a reward $R = f(M(\tilde{x}_{CF}), y_T)$, and the interaction tuple $(z, y_M, y_T, \tilde{z}_{CF}, R)$ is stored in the replay buffer.

Every U steps, the framework performs a gradient update (lines 13–19). First, a minibatch from the replay buffer is used to update the critic by minimizing the squared error between its predicted value and the observed reward, thereby improving its approximation of the reward function. Next, the actor is updated with a combined objective: a policy loss, which encourages the actor to maximize the critic's Q-value and thus generate valid counterfactuals, and a proximity loss, which penalizes large deviations from the original input. The overall actor loss $L_{\text{max}} + \lambda_p L_{\text{prox}}$ balances validity with proximity.

Through repeated interactions with the black-box classifier and joint optimization of both actor and critic, the actor progressively learns a policy for generating counterfactuals that achieve the desired label change while remaining close to the original time series.

Notice that our setup is equivalent to a Markov decision process with a one-step horizon, which eliminates the need for bootstrapping to compute the critic's target, thereby increasing stability and simplifying the training pipeline.

4.3 Inference Workflow

Once the actor and supporting components have been trained, Algorithm 2 describes the inference procedure for generating counterfactual explanations on the testing dataset. The process begins by loading the pre-trained actor μ together with the encoder–decoder pair (line 1). The encoder maps the input instance x into its latent representation z (line 2), which provides a structured space in which the actor can operate. At the same time, the black-box model M is queried to obtain the original prediction y_M (line 3), which is used alongside the target label y_T to guide counterfactual generation.

Given these inputs, the actor network produces a counterfactual latent embedding z_{CF} (line 4). This representation is then decoded back into the input space using the decoder to yield the counterfactual sequence x_{CF} (line 5). The final counterfactual instance is then returned (line 6). In practice, this inference workflow enables efficient and scalable generation of counterfactuals, since the trained actor produces explanations in a single forward pass without requiring per-instance optimization.

Algorithm 2 Generating explanations

Inputs: x: original instance; y_T: target label; M: black-box model
Outputs: x_{CF}: counterfactual instance

1: Load pre-trained actor μ, encoder enc, and decoder dec
2: Compute latent representation $z \leftarrow enc(x)$
3: Obtain model prediction $y_M \leftarrow M(x)$
4: Generate counterfactual latent $z_{CF} \leftarrow \mu(z, y_M, y_T)$
5: Decode $x_{CF} \leftarrow dec(z_{CF})$
6: **return** x_{CF} as the counterfactual instance

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate our approach on a diverse collection of publicly available univariate time series datasets drawn from the University of California, Riverside (UCR) Time Series Classification Archive (Dau et al., 2019). The UCR archive is a widely used benchmark suite for time series classification and has been extensively adopted to assess both traditional and deep learning models. To ensure that our findings are representative across different domains and levels of complexity, we select 13 datasets spanning application areas such as spectro, ECG, human activity recognition (HAR), sensor, traffic, etc. Table 1 in the Appendix provides detailed statistics for all datasets, including sequence length, number of classes, and train–test splits. These datasets have been shown to yield strong performance with state-of-the-art deep classifiers (Ismail Fawaz et al., 2019), making them suitable for evaluating the quality of counterfactual explanations.

The chosen datasets also vary in size, covering scenarios from small-scale problems with fewer than 100 training instances to medium-scale collections with 100–250 training samples, and to larger-scale datasets containing thousands of examples. This diversity allows us to systematically investigate how our method scales with training set size. Importantly, the datasets are partitioned following the standard UCR setup, where training and testing sets are evenly sampled across classes to avoid class imbalance issues.

Baselines. We compare our reinforcement learning framework against several state-of-the-art counterfactual explanation methods for time series classification. These baselines span optimization-based, saliency-guided, and latent-space perturbation approaches, representing the primary categories of existing techniques.

Wachter. Wachter et al. (2017) introduced one of the earliest general-purpose counterfactual frameworks. The method formulates counterfactual search as minimizing a weighted loss that balances prediction validity with input proximity, typically measured by the L_1 norm. Although originally designed for tabular data, it has been widely adapted as a baseline in time series settings (Delaney et al., 2021; Li et al., 2022a).

TimeX. Building on Wachter, Filali Boubrahimi & Hamdi (2022) enhances plausibility by incorporating Dynamic Barycenter Averaging (DBA) into the loss. This encourages counterfactuals to move toward the centroid of the target class under dynamic time warping (DTW), thereby improving interpretability and contiguity of perturbations.

Info-CELS. Li et al. (2025) extend the saliency-guided counterfactual explainer CELS (Li et al., 2023) by removing the thresholding step in saliency map generation. This adjustment eliminates noise from hard thresholds, producing smoother perturbations and significantly improving the validity of counterfactuals, while maintaining sparsity and proximity.

Glacier (AE variants). Wang et al. (2024) proposes a unified framework that performs gradient-based counterfactual search either in latent space or directly in input space, under different temporal constraints. In this study, we focus on the latent-space variants (`Glacier-AE`), where optimization is performed on autoencoder representations, balancing prediction-margin loss with constraint penalties to enforce sparsity and temporal plausibility. In particular, local constraints derived from LIMESegment (Sivill & Flach, 2022) highlight influential subsequences, guiding perturbations toward instance-specific regions.

Black-box Classifiers. For all counterfactual explanation methods, we employed the Fully Convolutional Network (FCN) as the black-box classifier. This provides a consistent evaluation framework across methods while supporting gradient-based approaches such as Wachter, TimeX, Info-CELS, and Glacier. In addition, the FCN has demonstrated strong and stable classification performance across diverse UCR datasets (Ismail Fawaz et al., 2019), making it a reliable backbone and ensuring that counterfactual evaluations are not affected by poor predictive accuracy. Further architectural and training details of the FCN are provided in the Appendix.

Evaluation Metrics. We evaluated the performances of different counterfactual models in terms of three major metrics:

(1) Validity Metric (Flip Label Rate): This metric measures how often the generated counterfactuals successfully change the model's original prediction. Formally, it is defined as:

$$flip\_rate = \frac{num\_flipped}{num\_testsample}, \tag{6}$$

where *num_flipped* is the number of counterfactuals that flip the predicted label, and *num_testsample* is the total number of test inputs. A higher flip rate (closer to 1) indicates better validity, as more counterfactuals satisfy the label change requirement.

(2) Proximity Metrics (L_1, L_2, and L_∞). Proximity evaluates how close each counterfactual is to the original input. We report three common distance-based metrics: (a) the L_1 Distance (Manhattan distance) measures the total absolute change across all time steps; (b) the L_2 Distance (Euclidean distance): Emphasizes larger differences by squaring deviations; (c) the L_∞ Distance (Chebyshev distance): Captures the maximum absolute deviation.

(3) Plausibility Metrics. (IF, LOF, OC_SVM) To assess whether the generated counterfactuals resemble real data, we apply three unsupervised outlier detection methods: (a) **Isolation Forest (IF)**

(Liu et al., 2008): Detects anomalies by randomly partitioning data and isolating outliers with fewer splits; (b) **Local Outlier Factor (LOF)** (Breunig et al., 2000): Measures how much a sample deviates from its local neighborhood density; (c) **One-Class SVM (OC_SVM)** (Schölkopf et al., 2001): Learns the boundary of normal data to identify samples that fall outside the learned distribution. Lower outlier scores from these models indicate higher plausibility, meaning the counterfactuals are more consistent with the training data distribution.

5.2 Experimental Results

Validity (Flip Rate): Figure 1 compares the validity of counterfactuals across methods, measured by flip rate. The violin plot shows the distribution of flip rates across datasets. (*The black horizontal line and red dot represent the median and mean across datasets, respectively. This is kept consistent across all violin plots in the paper.*) Our RL framework consistently achieves strong performance, with validity concentrated in the 0.8–1.0 range and a high mean and median (0.95), reflecting reliable performance across datasets. This indicates that nearly all generated counterfactuals successfully flip the classifier's decision, a critical property for actionable explanations. TimeX and InfoCELS often reach near-perfect validity (≈1.0), but their performance is less reliable, dropping below 0.7 on certain datasets. Glacier is even more fragile: while it can achieve high validity on some datasets, it collapses to zero on others. These fluctuations reduce the overall robustness of the baselines. By contrast, RL maintains a compact distribution, whereas the wider spread of InfoCELS, TimeX, and Glacier reveals susceptibility to dataset-specific failures. Taken together, these results highlight that RL provides a stable and generalizable mechanism for generating valid counterfactuals in diverse time series domains.

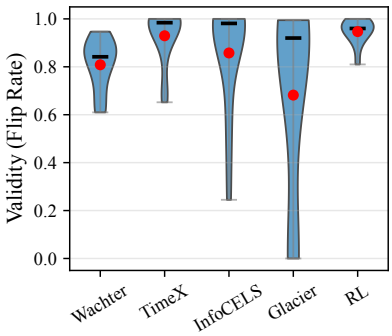

Figure 1: Validity evaluation measured by flip rate, where higher values indicate more valid counterfactuals.

Proximity (L1, L2, L_∞): Figure 2 reports the proximity of generated counterfactuals, where lower values indicate smaller perturbations from the original inputs. For L1 and L2 distances, InfoCELS, TimeX, and Wachter achieve the lowest values. InfoCELS attains the best proximity by perturbing only the most salient time steps, while TimeX and Wachter maintain relatively low distances through explicit distance-based penalties in their optimization objectives. By contrast, RL and Glacier operate in latent space rather than directly perturbing the input, which leads to higher L1 and L2 values after decoding, as modifications are distributed more broadly across the sequence. Although RL employs an elastic distance penalty within the actor network to constrain perturbations, the decoding process through the autoencoder inevitably spreads changes across multiple time steps, yielding larger aggregate distances.

For L_∞, which captures the largest pointwise deviation, RL ranks behind Wachter but demonstrates competitive performance with TimeX and InfoCELS, while clearly outperforming Glacier. This suggests that although RL produces broader perturbations—reflected in its higher L_1 and L_2 distances—it effectively avoids extreme spikes at individual time steps, leading to smoother and more controlled modifications. This behavior stems from the method's design: the autoencoder embedding promotes globally distributed adjustments rather than highly localized changes. As a result, RL yields counterfactuals that are less proximate in aggregate distance but remain better aligned with the underlying data manifold.

Plausibility (IF, LOF, OC_SVM): Figure 3 shows that RL maintains consistently low outlier scores across all three detectors. Under IF, RL remains within the 0–0.35 range, with both mean and median around 0.15, while the baselines span the full 0–1 interval with higher variance and frequent outliers. For LOF, RL stays below 0.3 across all datasets, achieving the lowest mean and median values (close to 0) and performing competitively with Wachter, TimeX, and InfoCELS, whereas Glacier again reaches up to 1.0. Under OC_SVM, RL clearly outperforms all baselines, keeping scores consistently below 0.1 with mean and median near zero, while Wachter and TimeX extend toward 0.4 and Glacier fluctuates widely between 0–1.

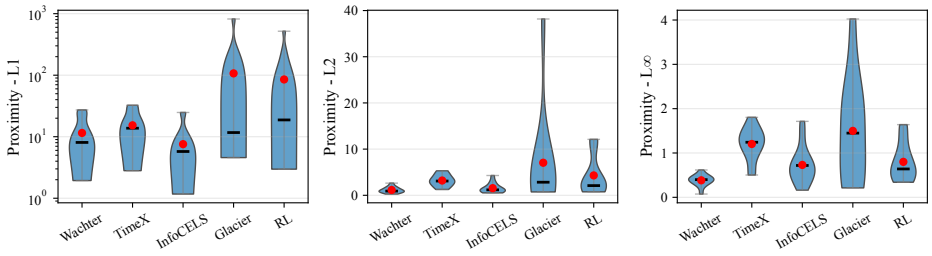

Figure 2: Proximity evaluation using three distance metrics. Lower values indicate counterfactuals closer to the original inputs.

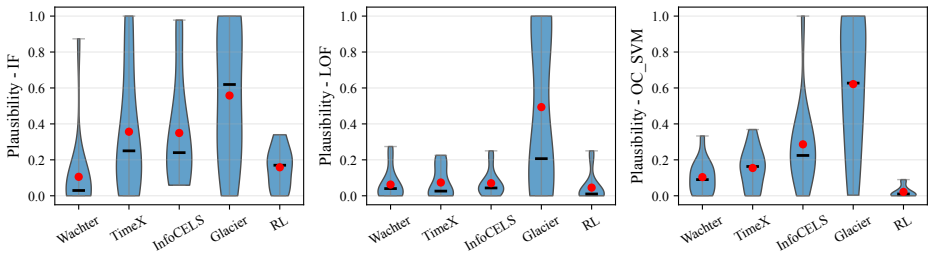

Figure 3: Plausibility evaluation using three outlier detection methods. Lower scores indicate more realistic counterfactuals.

These results demonstrate that RL counterfactuals align more closely with the data manifold, benefiting from latent-space policy learning that encourages globally coherent perturbations rather than localized, unrealistic changes. By operating in the latent space, RL captures structural regularities of the data, which translates into counterfactuals that are not only valid but also plausible under multiple outlier detection metrics. In contrast, the baselines either compromise plausibility for proximity—achieving lower distances at the cost of more out-of-distribution samples—or show instability across datasets, leading to less reliable counterfactuals. Overall, the evidence highlights RL as a method that balances validity, proximity, and plausibility, producing counterfactuals that are both effective for model explanation and realistic within the data distribution.

6 Conclusion

In this paper, we have developed an RL-based framework for counterfactual explanation in time series classification using Deep Deterministic Policy Gradient (DDPG). The framework treats counterfactual generation as a sequential decision process, where the agent learns to apply perturbations that flip the classifier's prediction. To guide learning, the reward function provides positive feedback for valid counterfactuals, while a proximity penalty discourages unrealistic deviations from the original input. After training, the actor can generate counterfactuals in batches, making the approach more scalable than instance-based optimization. Experimental results on 13 benchmark datasets demonstrate that our method achieves state-of-the-art performance, producing counterfactuals that are both valid and plausible.

Reproducibility Statement

To ensure reproducibility, we will release the full source code on an anonymous GitHub repository (https://github.com/bubbleblue0/Counterfactual-Explanations-for-Time-Series-Data-via-Reinforcement-Learning). Detailed descriptions of the datasets, hyperparameters, and model architectures (classifier, autoencoder, and actor–critic networks) are also provided in the Appendix, enabling researchers to replicate our experiments and results.

REFERENCES

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

Omar Bahri, Peiyu Li, Soukaina Filali Boubrahimi, and Shah Muhammad Hamdi. Temporal rule-based counterfactual explanations for multivariate time series. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1244–1249. IEEE, 2022b.

Subrato Bharati, M Rubaiyat Hossain Mondal, and Prajoy Podder. A review on explainable artificial intelligence for healthcare: Why, how, and when? *IEEE Transactions on Artificial Intelligence*, 5 (4):1429–1442, 2023.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. Explainable ai in fintech risk management. *Frontiers in Artificial Intelligence*, 3:26, 2020.

Gustau Camps-Valls, Miguel-Ángel Fernández-Torres, Kai-Hendrik Cohrs, Adrian Höhl, Andrea Castelletti, Aytac Pacal, Claire Robin, Francesco Martinuzzi, Ioannis Papoutsis, Ioannis Prapas, et al. Artificial intelligence for modeling and understanding extreme weather and climate events. *Nature Communications*, 16(1):1919, 2025.

Jurgita Černevičienė and Audrius Kabašinskas. Explainable artificial intelligence (xai) in finance: a systematic literature review. *Artificial Intelligence Review*, 57(8):216, 2024.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

Eoin Delaney, Derek Greene, and Mark T Keane. Instance-based counterfactual explanations for time series classification. In *International Conference on Case-Based Reasoning*, pp. 32–47. Springer, 2021.

Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, et al. Explainable ai (xai): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 55(9):1–33, 2023.

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

Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, 38(5):2770–2824, 2024.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

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

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

Peiyu Li, Omar Bahri, Soukaïna Filali Boubrahimi, and Shah Muhammad Hamdi. Cels: Counterfactual explanations for time series data via learned saliency maps. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 718–727. IEEE, 2023.

Peiyu Li, Omar Bahri, Pouya Hosseinzadeh, Soukaïna Filali Boubrahimi, and Shah Muhammad Hamdi. Info-cels: Informative saliency map-guided counterfactual explanation for time series classification. *Electronics*, 14(7):1311, 2025.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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

Hui Wen Loh, Chui Ping Ooi, Silvia Seoni, Prabal Datta Barua, Filippo Molinari, and U Rajendra Acharya. Application of explainable artificial intelligence for healthcare: A systematic review of the last decade (2011–2022). *Computer methods and programs in biomedicine*, 226:107161, 2022.

Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 650–665. Springer, 2021.

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

Robert-Florian Samoilescu, Arnaud Van Looveren, and Janis Klaise. Model-agnostic and scalable counterfactual explanations via reinforcement learning. *arXiv preprint arXiv:2106.02597*, 2021.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

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

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan Hines, John Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *ACM Computing Surveys*, 56(12):1–42, 2024.

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

Zhendong Wang, Isak Samsten, Ioanna Miliou, Rami Mochaourab, and Panagiotis Papapetrou. Glacier: guided locally constrained counterfactual explanations for time series classification. *Machine Learning*, pp. 1–31, 2024.

Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585. IEEE, 2017.

A Appendix

A.1 LLM Usage

GPT-5 was used as an assistive tool during the preparation of this manuscript. Specifically, GPT-5 was employed to help refine the writing style and improve the readability of our manuscript. All core research ideas, experiments, implementations, and analyses were conducted by the authors.

A.2 The details of datasets used in our experiments

This section provides detailed metadata of the UCR datasets employed in our experiments. For each dataset, we report the number of classes, sequence length, training and testing sizes, domain type, and the classification accuracy of the black-box model.

Table 1: UCR Datasets Metadata, Classifier Accuracy, and Autoencoder Best Validation Loss

ID	Dataset Name	$\mathcal{C}$	$\mathcal{L}$	DS train size	DS test size	Type	FCN Acc.	AE Val Loss
0	Chinatown	2	24	20	343	TRAFFIC	0.98	0.01
1	Coffee	2	286	28	28	SPECTRO	1.00	0.00
2	ECG200	2	96	100	100	ECG	0.91	0.06
3	FordA	2	500	3601	1320	SENSOR	0.93	0.22
4	FordB	2	500	3636	810	SENSOR	0.81	0.15
5	FreezerRegularTrain	2	301	150	2850	DEVICE	1.00	0.01
6	GunPoint	2	150	50	150	HAR	1.00	0.01
7	GunPointAgeSpan	2	150	135	316	HAR	1.00	0.01
8	GunPointMaleVersusFemale	2	150	135	316	HAR	1.00	0.01
9	GunPointOldVersusYoung	2	150	135	316	HAR	0.99	0.01
10	HandOutlines	2	2709	1000	370	IMAGE	0.82	0.04
11	TwoLeadECG	2	82	23	1139	ECG	1.00	0.01
12	Wafer	2	152	1000	6164	SENSOR	1.00	0.01

$\mathcal{C}$: number of classes; $\mathcal{L}$: sequence length. FCN Acc. reports classifier accuracy on the test set. AE Val Loss corresponds to the lowest validation loss achieved during autoencoder training.

A.3 The Architecture of FCN Model

The FCN architecture follows Wang et al. (2017) and the implementation of Ismail Fawaz et al. (2019). It consists of three one-dimensional convolutional blocks: the first applies 128 filters of size 8, the second 256 filters of size 5, and the third 128 filters of size 3. Each convolution is followed by batch normalization and a ReLU activation. A global average pooling layer aggregates temporal features, and a final dense layer with softmax activation outputs class probabilities. Training is performed with the Adam optimizer (learning rate 10^{-3}) and categorical cross-entropy loss. To improve convergence and prevent overfitting, we employ early stopping (patience 100, monitoring validation accuracy) and a learning rate scheduler that reduces the learning rate by a factor of 0.5 if the training loss plateaus for 30 epochs. Models are trained for up to 2000 epochs with a batch size of 16. For smaller datasets, we use a reduced mini-batch size of $\min(N/10, 16)$, where N denotes the number of training instances, to ensure stable optimization. The classification accuracy of the trained FCN on each test dataset is reported in Table 1.

A.4 The architecture of AutoEncoder

For counterfactual generation, we employed a convolutional autoencoder to provide a structured latent space for perturbations. The encoder comprises two one-dimensional convolutional layers with ReLU activations, each followed by max-pooling to progressively downsample the input, and a fully connected layer with `tanh` activation that maps the extracted features into a latent representation bounded in $[-1, 1]$, consistent with the requirements of the DDPG algorithm. The decoder reconstructs time series from this latent space by first expanding the latent code with a fully connected layer reshaped into a sequence, followed by causal one-dimensional convolutions and custom upsampling layers that preserve temporal ordering; a final convolutional layer with linear activation outputs a sequence with the original number of features, with cropping or last-value padding applied as needed to match the exact input length. The autoencoder was trained separately for each dataset using mean squared error (MSE) reconstruction loss and the Adam optimizer with a learning rate of 0.005. Training was performed for up to 2000 epochs with a batch size of 128, reduced to $\min(N/10, 128)$ for small datasets where N is the number of training samples, and employed early stopping with patience 50 together with learning rate reduction on plateau (factor 0.5, patience 30, minimum learning rate 10^{-6}). During training, the model was monitored using validation loss, and the weights with the lowest validation loss were saved. The best validation reconstruction losses obtained from these saved models are reported in Table 1.

A.5 Actor–Critic Architectures

The actor and critic networks used in our framework follow the standard design commonly adopted in deep reinforcement learning. Both models employ fully connected layers with layer normalization and ReLU activations, which provide stable training dynamics across tasks.

Specifically, the actor maps the state representation into a continuous action through a three-layer multilayer perceptron (MLP). Two hidden layers (each of size 256 with ReLU activations) are followed by an output layer with a $\tanh$ activation, which ensures bounded perturbations in the latent space. This network is responsible for generating candidate counterfactual representations. And the critic estimates the value of a state–action pair using a similar three-layer MLP. Two hidden layers of size 256 (with ReLU activations) are followed by a scalar output layer that predicts the Q-value. The critic provides feedback to guide the actor's updates by evaluating the expected reward of the generated counterfactuals.

A.6 Implementation Details of Reinforcement Learning

For counterfactual generation, we adopt the reinforcement learning framework implemented in the `alibi` library, using a Deep Deterministic Policy Gradient (DDPG) setup. The CFRL agent consists of an actor network, which generates perturbations, and a critic network, which estimates their quality. Both networks are trained jointly with additional sparsity and consistency losses to guide the learning process.

The training procedure is run for 50,000 steps, with dataset-specific batch sizes ranging from 16 to 256 depending on the dataset size (see code for mapping). The actor loss combines the policy gradient term (negative critic output) with proximity penalties, weighted by coefficients $\lambda_{\text{prox}} = 1$. The critic is trained by minimizing the squared error between its predicted Q-values and the observed rewards. To ensure stable training, Gaussian noise is added to the actor's output for exploration.

The predictor (black-box classifier) is kept fixed and only used to compute flip-label rewards. The autoencoder encoder–decoder pair is also fixed during RL training, providing the latent space for perturbations and reconstruction of counterfactuals. Rewards, success rates, and loss terms are monitored throughout training using Weights & Biases (wandb), and additional callbacks log intermediate samples and visualizations every 100 steps.

At the end of training, the learned actor network is able to generate counterfactuals in batches through a single forward pass, while the critic provides a learned notion of counterfactual quality. All reported results are based on the trained actor, without further fine-tuning at inference time.

A.7 The full table for our experimental results

In the main content, we visualize the overall performance of different methods using violin plots that aggregate results across all datasets. Tables 2-8 in the Appendix provide the full tabular results for each individual dataset. These results complement the violin plots in the main content by offering detailed, dataset-specific performance metrics. The reported metrics include **Validity** (measured by flip rate), **Proximity** (average L_1, L_2, and L_∞ distances), and **Plausibility** (measured by outlier scores from Isolation Forest (IF), Local Outlier Factor (LOF), and One-Class SVM (OC_SVM)). For proximity and plausibility, the averages are computed only over valid counterfactuals for each dataset, consistent with the procedure used in the violin plots, ensuring that evaluation reflects the quality of successful counterfactuals.

Table 2: Validity evaluation (Flip Rate) across UCR datasets. Higher values indicate more valid counterfactuals.

ID	Dataset	Wachter	TimeX	InfoCELS	Glacier	RL
0	Chinatown	0.8921	1.0000	0.2449	0.9038	0.9600
1	Coffee	0.7857	1.0000	1.0000	0.9286	1.0000
2	ECG200	0.8800	0.9800	1.0000	0.9200	0.9700
3	FordA	0.8674	0.9985	0.4955	0.6432	0.9400
4	FordB	0.6099	0.9222	0.9889	0.9444	0.8100
5	FreezerRegularTrain	0.6898	0.6519	0.9811	0.9825	1.0000
6	GunPoint	0.9467	1.0000	1.0000	0.9667	0.9200
7	GunPointAgeSpan	0.6171	0.8734	0.6582	0.0000	0.8800
8	GunPointMaleVersusFemale	0.8418	0.9968	0.9367	0.0000	0.9500
9	GunPointOldVersusYoung	0.8317	0.9841	0.8984	0.0000	0.9700
10	HandOutlines	0.7676	0.7108	0.9919	0.9946	1.0000
11	TwoLeadECG	0.8946	1.0000	1.0000	0.6119	0.9300
12	Wafer	0.8845	0.9655	0.9550	0.9685	0.9800

Table 3: Proximity evaluation measured by L_1 distance (mean values). Lower values indicate smaller perturbations from the original inputs.

ID	Dataset	Wachter	TimeX	InfoCELS	Glacier	RL
0	Chinatown	1.94	2.80	1.17	12.89	2.97
1	Coffee	17.08	6.18	6.10	10.17	18.74
2	ECG200	7.54	6.30	8.92	10.15	22.89
3	FordA	27.25	27.16	24.79	28.67	214.94
4	FordB	8.07	14.87	24.48	129.26	197.88
5	FreezerRegularTrain	23.54	17.56	3.40	10.49	17.20
6	GunPoint	6.61	12.65	6.91	35.12	17.49
7	GunPointAgeSpan	8.50	18.88	2.53	-	19.74
8	GunPointMaleVersusFemale	24.88	32.46	5.96	-	32.41
9	GunPointOldVersusYoung	6.10	10.13	1.93	-	17.97
10	HandOutlines	8.97	32.05	5.75	819.71	518.62
11	TwoLeadECG	4.11	4.42	4.09	9.63	11.25
12	Wafer	5.07	13.75	2.37	4.58	16.13

Table 4: Proximity evaluation measured by L_2 distance (mean values). Lower values indicate smaller perturbations from the original inputs.

ID	Dataset	Wachter	TimeX	InfoCELS	Glacier	RL
0	Chinatown	0.66	1.46	0.52	4.26	0.75
1	Coffee	1.51	1.27	1.12	1.04	1.50
2	ECG200	1.20	2.50	2.50	2.99	3.30
3	FordA	1.68	5.10	2.92	2.68	12.02
4	FordB	0.60	3.27	4.29	10.76	11.13
5	FreezerRegularTrain	1.90	4.25	1.26	1.19	1.99
6	GunPoint	0.91	3.02	1.65	6.36	1.88
7	GunPointAgeSpan	1.18	4.29	0.83	-	2.10
8	GunPointMaleVersusFemale	2.63	5.32	1.48	-	3.42
9	GunPointOldVersusYoung	0.79	2.58	0.67	-	1.96
10	HandOutlines	0.27	3.09	0.59	38.17	12.11
11	TwoLeadECG	0.84	1.95	1.18	2.33	1.66
12	Wafer	0.77	3.49	1.07	0.73	2.13

Table 5: Proximity evaluation measured by L_∞ distance (mean values). Lower values indicate smaller maximum deviations from the original inputs.

ID	Dataset	Wachter	TimeX	InfoCELS	Glacier	RL
0	Chinatown	0.42	1.02	0.37	2.22	0.34
1	Coffee	0.40	0.50	0.45	0.21	0.34
2	ECG200	0.47	1.40	1.39	1.80	1.23
3	FordA	0.36	1.80	0.86	0.63	1.64
4	FordB	0.25	1.24	1.71	2.30	1.57
5	FreezerRegularTrain	0.47	1.54	0.84	0.37	0.88
6	GunPoint	0.37	1.17	0.73	2.06	0.48
7	GunPointAgeSpan	0.47	1.48	0.50	-	0.50
8	GunPointMaleVersusFemale	0.62	1.45	0.75	-	0.78
9	GunPointOldVersusYoung	0.32	0.99	0.42	-	0.52
10	HandOutlines	0.07	0.52	0.16	4.02	0.54
11	TwoLeadECG	0.42	1.26	0.61	1.09	0.64
12	Wafer	0.34	1.24	0.72	0.26	0.94

Table 6: Plausibility evaluation measured by Isolation Forest (IF) outlier scores. Lower values indicate counterfactuals closer to the data manifold.

ID	Dataset	Wachter	TimeX	InfoCELS	Glacier	RL
0	Chinatown	0.00	0.00	0.60	1.00	0.34
1	Coffee	0.04	0.07	0.07	0.78	0.07
2	ECG200	0.07	0.25	0.39	0.33	0.03
3	FordA	0.00	1.00	0.94	0.00	0.00
4	FordB	0.87	1.00	0.98	0.00	0.00
5	FreezerRegularTrain	0.00	0.63	0.24	0.22	0.21
6	GunPoint	0.10	0.30	0.36	0.62	0.22
7	GunPointAgeSpan	0.06	0.21	0.12	-	0.17
8	GunPointMaleVersusFemale	0.01	0.08	0.22	-	0.20
9	GunPointOldVersusYoung	0.03	0.14	0.06	-	0.15
10	HandOutlines	0.17	0.30	0.15	0.34	0.25
11	TwoLeadECG	0.00	0.00	0.09	0.65	0.08
12	Wafer	0.02	0.65	0.33	0.32	0.34

Table 7: Plausibility evaluation measured by Local Outlier Factor (LOF) scores. Lower values indicate counterfactuals closer to the data manifold.

ID	Dataset	Wachter	TimeX	InfoCELS	Glacier	RL
0	Chinatown	0.00	0.05	0.25	0.99	0.00
1	Coffee	0.00	0.14	0.04	0.08	0.00
2	ECG200	0.00	0.00	0.03	0.21	0.00
3	FordA	0.00	0.00	0.00	0.00	0.00
4	FordB	0.00	0.00	0.00	0.00	0.00
5	FreezerRegularTrain	0.04	0.20	0.09	0.20	0.04
6	GunPoint	0.06	0.10	0.06	0.68	0.02
7	GunPointAgeSpan	0.06	0.02	0.10	-	0.02
8	GunPointMaleVersusFemale	0.27	0.19	0.04	-	0.21
9	GunPointOldVersusYoung	0.11	0.00	0.03	-	0.01
10	HandOutlines	0.22	0.23	0.20	0.99	0.25
11	TwoLeadECG	0.00	0.00	0.04	0.18	0.00
12	Wafer	0.05	0.03	0.04	0.10	0.04

Table 8: Plausibility evaluation measured by OC_SVM outlier scores. Lower values indicate counterfactuals closer to the data manifold.

ID	Dataset	Wachter	TimeX	InfoCELS	Glacier	RL
0	Chinatown	0.09	0.01	1.00	1.00	0.01
1	Coffee	0.00	0.29	0.35	0.23	0.00
2	ECG200	0.07	0.20	0.16	0.52	0.01
3	FordA	0.00	0.00	0.38	0.52	0.09
4	FordB	0.00	0.00	0.00	0.00	0.08
5	FreezerRegularTrain	0.03	0.27	0.12	0.41	0.01
6	GunPoint	0.17	0.18	0.35	0.76	0.03
7	GunPointAgeSpan	0.13	0.16	0.40	-	0.00
8	GunPointMaleVersusFemale	0.16	0.12	0.16	-	0.00
9	GunPointOldVersusYoung	0.14	0.15	0.13	-	0.01
10	HandOutlines	0.33	0.37	0.22	0.92	0.00
11	TwoLeadECG	0.04	0.07	0.18	0.63	0.00
12	Wafer	0.18	0.19	0.26	0.09	0.04