# GRSF Counterfactual Generation Using Gradient Based Method.

**Gabin Vrillault**
ENSICAEN - École Nationale Supérieure d'Ingénieurs de Caen
6 Boulevard Maréchal Juin, 14050 Caen, France
gabin.vrillault@ecole.ensicaen.fr

### Abstract

AI systems are becoming increasingly present in our daily lives, and their decisions can have significant impacts on individuals and society as a whole. Their ability to make predictions and decisions can be critical in various domains, such as healthcare, finance, or autonomous driving. However, these systems are often perceived as black boxes, making it difficult to understand their decision-making processes and to trust their predictions. AI explainability research aims to address this issue by providing insights into how AI systems make decisions, and to help users understand how to interpret their predictions. Furthermore, these systems can be vulnerable to a variety of attacks, which can compromise their integrity and reliability. The AI security research field aims to address these challenges by developing methods and techniques to ensure that AI systems are robust and resilient. One of the key challenges in AI security is to ensure that AI systems are not vulnerable to adversarial attacks, which uses maliciously crafted inputs to deceive AI systems into making incorrect predictions or decisions. In this work, we inspire from AI security technics to generate counterfactual samples for a time-series classifier: GRSF (Generalized Random Shapelet Forest) which is an explainable oriented time-series classifier. Our method is inspired from the work done in adversarial sample generation using gradient based solutions.

## Introduction

The main idea behind *GRSF* (Karlsson, Papapetrou, and Boström 2016) is to build a set of decision trees, where each feature corresponds to a shapelet. The decision condition on an internal node is the presence or absence of a Shapelet in a test time series example. As mentioned in (Karlsson et al. 2020), *GRSF*, while being performant on a large collection of time series datasets, is still an opaque classification model. Hence the need for explainability and interpretability of the model. There are many ways to explain a model, but the approach that we will focus on in this work is the counterfactual generation. Counterfactual samples are minimally modified samples that are "close" using a distance to a base sample from a base class, but are classified as a sample from an other class. They provide a way to understand the decision boundaries of our model and thus can be used to determine to what extent our model can be trusted and to identify potential biases in the model. In AI security, adversarial samples are similar to counterfactual samples, but in this case the minimality is not a requirement, the main goal is to generate samples that are systematically misclassified by the model. These two concepts are closely related, and in this work we will use adversarial sample generation inspiration to generate counterfactual samples for *GRSF*. The chosen method uses gradient based approach greatly inspired by adversarial sample generation from (Shafahi et al. 2018). Since *GRSF* is not a gradient based classifier, we rely on gradient based surrogate models that mimic our non differentiable model to generate counterfactuals.



## Related Work

Counterfactual generation has been widely explored in the literature, it has become a cornerstone of post-hoc explainability. They provide actionable, instance-specific insights into model behavior by adding minimal changes to input feature that alter the prediction (Verma, Dickerson, and Hines 2020). Hence the usefulness of counterfactual explanations for interpreting black-box models (Byrne 2019), their feature-level reasoning is particularly aligned with human understanding

In time series classification, counterfactual explanations have gained increasing attention. (Ates et al. 2021; Delaney, Greene, and Keane 2021), explored methods to generate plausible counterfactuals for time series data, highlighting the challenges posed by temporal dependencies and data continuity.

More specifically, the generation of counterfactuals for *GRSF* has been studied in (Karlsson et al. 2020), where the authors propose a method that minimally perturbs key parts of Shapelets until the model's prediction flips. This approach emphasizes interpretability by anchoring changes in human-interpretable components of the time series.

Generative models have also been explored for counterfactual generation. (Nemirovsky et al. 2020) GAN-based framework (CounterGAN) to synthesize realistic counterfactual instances. Similarly (Van Looveren et al. 2021) propose a framework for generating conditional and contrastive multimodale counterfactual explanations, focusing on generating meaningful alternatives that satisfy desired outcome constraints while maintaining plausibility.

# Problem Formulation

## Definitions

**L2 distance** used to compare time series, is defined as follows:

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

where $x$ and $y$ are two time series of length $n$.

**DTW distance** (Dynamic Time Warping) is a distance measure that allows to compare time series of different lengths by aligning them in a non-linear way. It is more robust than Euclidean distance to shifts and scaling of the time series, making it particularly useful for time series classification tasks. We use this distance measure to compare time series in our counterfactual generation method. It is defined as follows:

$$d_{DTW}(x, y) = \min_{\pi} \sum_{(i,j)\in\pi} (x_i - y_j)^2 \tag{2}$$

where $\pi$ is a warping path that aligns the two time series $x$ and $y$.

For our counterfactual generation problem, we define the following notations:

- $C_b$ is the class of the base sample $b$.
- $C_t$ is the target class.
- $\mathcal{G}$ is the classifier we want to generate counterfactuals for (*GRSF*).
- $x$ is the counterfactual sample.
- $b$ is the base instance of $x$ we want to modify to generate a counterfactual.
- $t$ is the target of class $C_t$ that $x$ should use as a reference to be classified as $C_t$.
- $\mathcal{S}$ is the surrogate model used to generate the counterfactuals.
- $D(x, y)$ is the distance between two time series $x$ and $y$, which can be either L2 or DTW distance.

## Counterfactual Generation Problem

Since our solution is inspired by the generation process of adversarial samples, we are able to perform targeted counterfactual generation, meaning that we can modify a sample from class $C_b$ to be classified as a sample from class $C_t$.

We can thus formulate the counterfactual generation problem as follows:

$$\begin{aligned}
&\text{Find } x \text{ such that:}\\
&D(x, b) \leq \epsilon\\
&D(x, t) \leq \delta\\
&\mathcal{G}(x) = C_t\\
&\mathcal{G}(t) = C_t\\
&\mathcal{G}(b) = C_b
\end{aligned}$$

where $\epsilon$ is the maximum allowed distance between the base sample $b$ and the counterfactual sample $x$, and $\delta$ is the maximum allowed distance between the target sample $t$ and the counterfactual sample $x$.

We define the following loss function that can be used to generate counterfactuals:

- **L2 distance loss:** This loss function is used to generate counterfactuals that are close to the target sample $t$.

$$L_2(x, t) = d(x, t) \tag{3}$$

# Time Series Modifications

## Experimental Setup

We use the UCR archive (Dau et al. 2019) to evaluate our method. The UCR archive is a collection of time series datasets that are widely used in the literature to evaluate time series classification algorithms. We tested our method on the following datasets:

- **Beef** - 60 samples, 470 points, 5 classes.
- **ECG200** - 200 samples, 96 points, 2 classes.
- **BME** - 180 samples, 128 points, 3 classes.

To train and test the *GRSF* model, we split the datasets into training and test sets, using 80% of the samples for training and 20% for testing. We use the *GRSF* implementation from the *wildboar* python library (Samsten 2020; Karlsson et al. 2020; Karlsson, Papapetrou, and Boström 2016) to use the model as a classifier. We implemented the counterfactual generation method in python, using the *pytorch* library (Paszke et al. 2019) to perform the gradient descent and the backward step. We have used three surrogate model architectures to generate counterfactuals:

- **Simple feed-forward neural network** - A simple feed-forward neural network with one hidden layer a linear input layer and ReLU activation function.
- **Convolutional neural network** - A 1D convolutional neural network with a single convolutional layer and ReLU activation function.
- **LSTM (Long Short-Term Memory)** - A recurrent neural network classifier that uses stacked LSTM layers with configurable depth and dropout regularization

For each experiment the surrogate models are trained using the same training set as the GRSF model. It is important to note that in a real-world scenario, the surrogate model would be trained on a different dataset than the one used to train the GRSF model, as the original dataset may not be available. The training and testing splits are randomly generated for each dataset and re generated for each pair of *GRSF* and surrogate model.

## Approximation of GRSF

The key idea of our method is to mimic the decision boundaries of the *GRSF* model. In order to validate the effectiveness of our assumption, we need to ensure that the surrogate model have the same prediction as the *GRSF* model on the test set. We have computed the fidelity by comparing the predictions of both models in the table 1.

| Surrogate Model | Fidelity (%) Beef | Fidelity (%) ECG200 | Fidelity (%) BME |
|---|---|---|---|
| FNN | 0.75 | 0.83 | 1.00 |
| CNN | 0.75 | 0.88 | 1.00 |
| LSTM | 0.75 | 0.90 | 1.00 |

Table 1: Fidelity of the surrogate models to the GRSF model on the test set.

The table 1 shows that the surrogate models are able to approximate the decision boundaries of the *GRSF* model with a high fidelity and thus validate our assumption that the surrogate model can be used to generate counterfactuals for the *GRSF* model.

## Global Perturbation

The global perturbation process is based on a gradient descent and a backward step. The algorithm was inspired by the work done by (Shafahi et al. 2018) that proposed a method to generate adversarial samples to insert poison samples in a DNN model.

**Algorithm 1:** Counterfactual Example Generation
**Input:** target instance $t$, base instance $b$, learning rate $\lambda$
**Initialize:** $x_0 \leftarrow b$
**Define:** $L_p(x) = Loss(x, t)$
**for** $i = 1$ to maxIters **do**
Forward step: $\widehat{x_i} = x_{i-1} - \lambda \nabla_x L_p(x_{i-1})$
Backward step: $x_i = (\widehat{x_i} + \lambda\beta b)/(1 + \beta\lambda)$
**end for**

We set the parameter $\beta$ to a value between 0.5 and 0.6 based on empirical results. The $L_2(x, t)$ loss function is used to generate counterfactuals that not only change the class of the base sample $C_b$ to $C_t$, but also try to make the counterfactual sample $x$ as close as possible to the target sample $t$. The parameter $\beta$ is used to make the counterfactual sample "look realistic" in the sense that it would fool a human observer into thinking that the sample has not been modified. For instance, on the figure 1, we can see that the counterfactual sample is very similar visually to the target sample, in both global and local perturbation cases, the differences could probably not be detected by a human observer. Also, to ensure a minimal perturbation, we check at each iteration the validity of the current counterfactual sample $x_i$ and break the loop if the prediction of the *GRSF* model is $C_t$. It is important to note that our approach has does not reach absolute minimal perturbation of the base sample, as it does not use a distance threshold to stop the perturbation process. It would be interesting to use an other loss function definition to seek the absolute minimal perturbation.

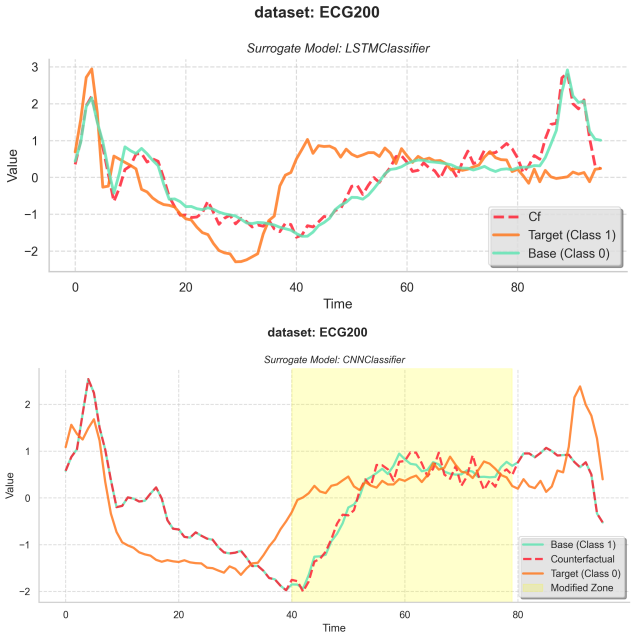


Figure 1: Top: Global perturbation counterfactual with base and target sample. Bottom: Local perturbation counterfactual showing region-specific modifications.

## Local Perturbation

For the local perturbation strategy, we adopt a methodology similar to that used in the global perturbation approach. However, there is a new parameter introduced to create a binary mask mechanism used during the gradient optimization process to constrain the perturbation to a specific temporal segment of the time series. This parameter enables the generation of region-specific counterfactuals, which are particularly valuable for interpretability. It is important to note that we did not implement a mechanism to automatically select the region to modify. This local perturbation strategy facilitates a more refined analysis of the model's decision boundaries on specific segments of the input time series.

# Results

We evaluate our counterfactual generation method on three datasets from the UCR archive. The following table presents the experimental results, showing the accuracy of both the GRSF model and the surrogate model on the test set and the validity rate of the generated counterfactuals. We consider a counterfactual to be valid if it is classified as the target class $C_t$ by the GRSF model.

The results demonstrate that our gradient-based approach can successfully generates valid counterfactuals with relatively low distances to the base sample on average. When we analyze visually the generated counterfactuals, we observe that they are very similar to the base sample, with only slight modifications in the time series. Although some counterfactuals are very noisy and thus not very realistic. We did not

| Dataset | GRSF accuracy | Surrogate accuracy | Validity (%) | AVG DTW distance |
|---|---|---|---|---|
| Beef | 0.83 | 0.92 | 96.0 | 1.17 |
| ECG200 | 0.93 | 0.90 | 66.0 | 4.06 |
| BME | 1.00 | 1.00 | 60.0 | 8.37 |

Table 2: Experimental results for counterfactual generation across different datasets using a simple feed-forward neural network surrogate model.

| Dataset | GRSF accuracy | Surrogate accuracy | Validity (%) | AVG DTW distance |
|---|---|---|---|---|
| Beef | 0.83 | 0.91 | 76.0 | 1.41 |
| ECG200 | 0.93 | 0.90 | 6.0 | 0.72 |
| BME | 1.00 | 1.00 | 94.0 | 1.29 |

Table 3: Experimental results for counterfactual generation across different datasets using a convolutional neural network surrogate model.

| Dataset | GRSF accuracy | Surrogate accuracy | Validity (%) | AVG DTW distance |
|---|---|---|---|---|
| Beef | 0.83 | 0.92 | 48.00 | 1.02 |
| ECG200 | 0.95 | 0.9 | 30.0 | 1.08 |
| BME | 1.00 | 1.00 | 38.0 | 1.26 |

Table 4: Experimental results for counterfactual generation across different datasets using a LSTM surrogate model.

formally evaluate the realism of the generated counterfactuals. The validity percentages of the counterfactuals fluctuates across datasets and surrogate models, with the simple feed-forward neural network achieving the highest validity rates on average. Thus, we can see that the choice of surrogate model is crucial for the effectiveness of counterfactual generation using our method.

The complete implementation of our method and all experimental code are available on GitHub[1], ensuring full reproducibility of our results. The repository includes the dataset loading scripts, surrogate model implementations, the counterfactual generation algorithms described in this work and a simple web-based interface to visualize and interact with the generation process and results.

## Comparison with a Baseline Method

To validate the effectiveness of our method, we compare our counterfactual generation approach with a baseline method. We only consider the global perturbation strategy for this comparison. The baseline method is a repetition of random perturbations on the base sample $b$ until a counterfactual sample $x$ is found that is classified as the target class $C_t$ by the *GRSF* model. The random perturbation is done by adding a random noise to the base sample $b$ and checking if the perturbed sample is classified as the target class $C_t$.

The results of the baseline method presented in table 5 show that the method is able to generate counterfactuals with a high validity rate on average, but with a higher average DTW distance compared to our method. Also the generated counterfactuals do not retain any similarity to the base sample, our method has a better control over the generated counterfactuals.

[1]https://github.com/gabinvr/grsf-counterfactual-generation

| Dataset | Validity (%) | AVG DTW distance |
|---|---|---|
| Beef | 81.82 | 2.62 |
| ECG200 | 90.91 | 1.38 |
| BME | 1.00 | 2.25 |

Table 5: Baseline method results for counterfactual generation across different datasets.

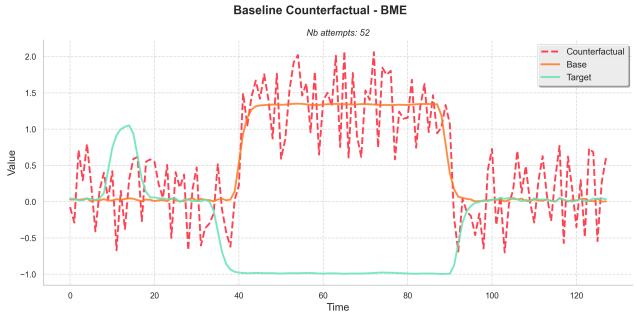


Figure 2: Counterfactual generated by the baseline method. The counterfactual is very noisy.

## Conclusion

In this work, we presented a method to generate counterfactual samples for the GRSF algorithm using a gradient based approach. We showed that it is possible to generate counterfactuals for GRSF by using our method against a gradient based surrogate model. We also showed that it is possible to generate both global and local counterfactuals relatively efficiently. Our method is capable of generating counterfactual with specific properties, such as being close to the base sample. Thus allowing a better understanding of the decisions made by the *GRSF* model. Moreover we have been able to test our method on three different datasets from the UCR archive, showing that it is possible to generate counterfactuals for GRSF on different time series datasets with an average validity of 79% with a carefully selected surrogate model. Finally we have compared our method with a baseline method that generates counterfactuals by random perturbations, showing that our method is able to generate meaningful counterfactuals with a lower average DTW distance to the base sample.

## References


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.

Byrne, R. M. 2019. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *IJCAI*, 6276–6282. California, CA.

Dau, H. A.; Bagnall, A.; Kamgar, K.; Yeh, C.-C. M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C. A.; and Keogh, E. 2019. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica* 6(6):1293–1305.

Delaney, E.; Greene, D.; and Keane, M. T. 2021. Instance-based counterfactual explanations for time series classification. In *International conference on case-based reasoning*, 32–47. Springer.

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

Karlsson, I.; Papapetrou, P.; and Boström, H. 2016. Generalized random shapelet forests. *Data mining and knowledge discovery* 30:1053–1085.

Nemirovsky, D.; Thiebaut, N.; Xu, Y.; and Gupta, A. 2020. Countergan: Generating realistic counterfactuals with residual generative adversarial nets. *arXiv preprint arXiv:2009.05199*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. 8024–8035.

Samsten, I. 2020. isaksamsten/wildboar: wildboar 1.0. 3.

Shafahi, A.; Huang, W. R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems* 31.

Van Looveren, A.; Klaise, J.; Vacanti, G.; and Cobb, O. 2021. Conditional generative models for counterfactual explanations. *arXiv preprint arXiv:2101.10123*.

Verma, S.; Dickerson, J.; and Hines, K. 2020. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596* 2(1):1.