Generating Sparse Counterfactual Explanations For Multivariate Time Series

Jana Lang
Section Computational Sensomotorics
Hertie Institute for Clinical Brain Research
Centre for Integrative Neuroscience
University of Tübingen, Germany
jana.lang@uni-tuebingen.de

Martin Giese
Section Computational Sensomotorics
Hertie Institute for Clinical Brain Research
Centre for Integrative Neuroscience
University of Tübingen, Germany
martin.giese@uni-tuebingen.de

Winfried Ilg
Section Computational Sensomotorics
Hertie Institute for Clinical Brain Research
Centre for Integrative Neuroscience
University of Tübingen, Germany
winfried.ilg@uni-tuebingen.de

Sebastian Otte
Neuro-Cognitive Modeling
University of Tübingen, Germany
sebastian.otte@uni-tuebingen.de

Abstract

Since neural networks play an increasingly important role in critical sectors, explaining network predictions has become a key research topic. Counterfactual explanations can help to understand why classifier models decide for particular class assignments and, moreover, how the respective input samples would have to be modified such that the class prediction changes. Previous approaches mainly focus on image and tabular data. In this work we propose SPARCE [1], a generative adversarial network (GAN) architecture that generates SPARse Counterfactual Explanations for multivariate time series. Our approach provides a custom sparsity layer and regularizes the counterfactual loss function in terms of similarity, sparsity, and smoothness of trajectories. We evaluate our approach on real-world human motion datasets as well as a synthetic time series interpretability benchmark. Although we make significantly sparser modifications than other approaches, we achieve comparable or better performance on all metrics. Moreover, we demonstrate that our approach predominantly modifies salient time steps and features, leaving non-salient inputs untouched.

1 Introduction

With the advent of machine learning for decision making in critical sectors like healthcare, predictive maintenance, or traffic, serious concerns have been raised about the trustworthiness of these algorithms. In recent years, the field of explainable artificial intelligence (XAI) has therefore gained increasing popularity. While manifold techniques for explaining tabular data and image classifiers have been proposed, temporal data has largely been neglected. In contrast to image data, time series interpretability poses manifold challenges, including the presence of distinct time and space dimensions and an increased difficulty of visualizing information in a meaningful way. Recent work has raised strong concerns about the adaptability of prevalent XAI methods to multivariate time series [Ismail et al., 2020].

[1] Code: https://github.com/janalang/SPARCE

Preprint. Under review.

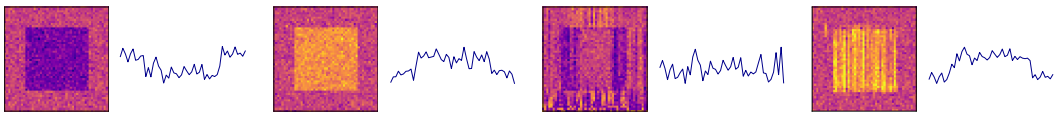

Figure 1: Counterfactuals generated using a state-of-the-art approach [Nemirovsky et al., 2020] and our approach for a multivariate time series. Columns represent features and rows represent time steps. The curves arranged right to the boxes depict respective sequences for one of the center features.

Counterfactual Explanations Derived from philosophical reasoning, *counterfactual explanations* try to find modifications to an input query so that the classification changes to a desired class [Wachter et al., 2017]. Features of the input query can be mutable, i.e. the values can and may be modified, or immutable. A valid counterfactual should only modify mutable input features [Karimi et al., 2021]. Meaningful counterfactual explanations can guide users towards a better understanding of decisions made by a system. If a classifier predicts a certain disease risk based on a patient's medical record, it is helpful to understand not only what factors led to the decision, but also what factors would have to change and in which way to minimize the risk.

1.1 Objectives For Counterfactual Explanations

Precision, Similarity & Realism A valuable counterfactual explanation is close to the original data point, looks plausible and realistic and suggests actionable modifications [Dandl et al., 2020, Keane and Smyth, 2020]. The choice of distance functions to measure the actionability of a counterfactual has been a topic of discussion. The original approach by Wachter et al. [2017] iteratively minimizes the distance between the predicted class for the counterfactual and the target class (via $\mathcal{L}_2$ norm) as well as the distance between query and counterfactual (via $\mathcal{L}_1$ norm) using gradient descent. Dandl et al. [2020] additionally assess realism of the generated counterfactual by measuring how likely it is that the counterfactual stems from the observed data distribution.

Sparsity Dandl et al. [2020] implement sparsity as the $\mathcal{L}_0$ norm between query and counterfactual, that measures how many features were changed to go from the original data point to the counterfactual. Mothilal et al. [2020] do not include sparsity into the loss function, but modify the generated counterfactuals post-hoc using a greedy algorithm to set increasingly more features with smaller modifications zero until the prediction changes. In contrast, Keane and Smyth [2020] define a rigid threshold for sparsity stating that a good counterfactual for tabular data may only modify up to two features. Adapting this paradigm to time series, Delaney et al. [2021] only allow for modification of one single contiguous section of the time series. Others only ensure feature sparsity, while modifying all time steps of the sequence [Ates et al., 2021].

Similarity vs. Sparsity Figure 1 demonstrates why similarity alone does not guarantee actionability. The counterfactual generated by our approach makes sparse, but more substantial modifications, while the counterfactual generated using a state-of-the-art approach makes minor changes in all time steps and features. If solely regularized by the $\mathcal{L}_1$ norm (i.e. the similarity constraint), the latter would be preferred. Taking the actionability of the counterfactual into account, one would most likely prefer the counterfactual generated by SPARCE despite the higher $\mathcal{L}_1$ loss. As a consequence, sparsity plays a central role in our approach.

1.2 Generative Approaches

To generate more realistic and plausible counterfactuals, while overcoming high computational costs of iterative optimization methods, generative adversarial networks (GANs) have recently been introduced for the generation of counterfactual explanations [Nemirovsky et al., 2020, Van Looveren et al., 2021]. GANs have become popular for generating realistic looking fake images by training a generator to create fake samples that a discriminator would erroneously perceive as real samples [Goodfellow et al., 2014]. GAN-based architectures for counterfactual search add a classifier to the standard GAN approach. In this way, the generator learns to produce realistic looking counterfactuals that change the classifier's prediction to a target class.

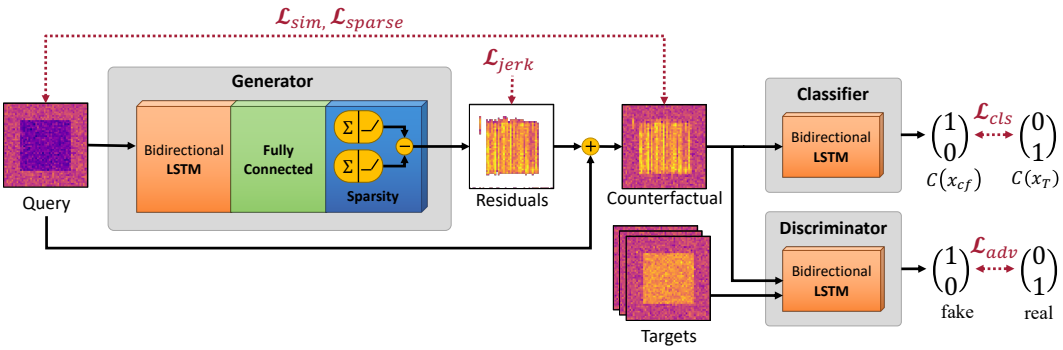

Figure 2: Schematic illustration of our GAN-based approach for counterfactual search. Inputs are divided into query and target time series (displayed as heatmaps) according to the desired target class. A recurrent generator with sparsity activation generates residuals for each query. Residuals are added to the corresponding query to create a counterfactual explanation. A pretrained sequence classifier predicts the class label of the counterfactual. A recurrent discriminator tries to distinguish counterfactuals from real targets.

While Nemirovsky et al. [2020] only evaluate their model on image and tabular data, Van Looveren et al. [2021] also assess their approach on univariate time series. Both approaches use $\mathcal{L}_1$ or $\mathcal{L}_2$ norms as regularization terms that act on the generator's loss function. Particularly for multivariate time series, this formulation is problematic, since it creates proximate, but not sparse counterfactuals. Indeed, sample counterfactuals generated by Van Looveren et al. [2021] modify every single time step of the query sequence. In some domains, this might be necessary. However, it is questionable whether such a counterfactual explanation would have any explanatory power. Besides, it is unclear whether these modifications could actually be acted upon in reality. Our approach is thus designed to create truly sparse counterfactual explanations for multivariate time series without compromising other important objectives of counterfactuals, including realism, similarity, and plausibility.

2 Method

Motivated by the insufficient adaptation of counterfactual approaches to multivariate time series, we propose SPARCE: a novel framework to efficiently generate SPARse Counterfactual Explanations for multivariate time series data. Our approach aims to change the class label of an original time series to a target class (*precision*). Generated counterfactuals should be within the distribution of the original data points (*realism*) and stay as close as possible to the query sequence (*similarity*). In contrast to related approaches for multivariate time series, we postulate that counterfactuals are time- and feature-sparse, i.e. that only a subset of features and time steps is modified (*sparsity*). Finally, for applications where time series evolve smoothly over time, we aim to modify the original data point in a temporally plausible manner (*smoothness*).

2.1 Generating Counterfactual Explanations

Basing our approach on a generator-discriminator architecture, we ensure realism of the generated samples. In line with Nemirovsky et al. [2020] we define a modified generator $\mathcal{G}$ which learns to generate residuals $\delta = \mathcal{G}(\mathbf{x}_q)$ from the input sample $\mathbf{x}_q$. In contrast to standard GANs, the generator does not use a random seed, but real samples as inputs. Thus, original samples are first divided into queries $\mathbf{x}_q$ and targets $\mathbf{x}_t$. Targets are samples labeled as the target class $\mathbf{c}_t$ and are used as real examples for the discriminator $\mathcal{D}$. The query subset contains all other samples and is presented as inputs to the generator $\mathcal{G}$. Residuals created by the generator are added to the query to produce a counterfactual ($\mathbf{x}_{cf} = \mathbf{x}_q + \delta$). A pre-trained classifier $\mathcal{C}$ determines the class prediction for the generated counterfactual. At the same time, the counterfactual is presented to the discriminator as a fake sample. In combination with real target samples, the discriminator tries to distinguish between real and fake (i.e. generated) samples. The realism of the generated counterfactual examples increases as the generator learns to fool the discriminator. The classifier prevents the generator from producing zero-residuals, i.e. from learning the identity function.

$$\mathcal{L}_{adv} = -\log(\mathcal{D}(\mathbf{x}_q + \mathcal{G}(\mathbf{x}_q))) \quad (1) \qquad \mathcal{L}_{cls} = -\mathbf{c}_t \log(\mathcal{C}(\mathbf{x}_q + \mathcal{G}(\mathbf{x}_q))) \quad (2)$$

$$\mathcal{L}_{sim} = \|\mathbf{x}_q - \mathbf{x}_{cf}\|_1 \quad (3) \qquad \mathcal{L}_{sparse} = \|\mathbf{x}_q - \mathbf{x}_{cf}\|_0 \quad (4)$$

$$\mathcal{L}_{jerk} = \sum_{t=0}^{T-1} \|\delta^{t+1} - \delta^t\|_2 \quad (5)$$

$$\mathcal{L}_G = \mathbb{E}_{\mathbf{x}_q}\left[\lambda_1 \mathcal{L}_{adv} + \lambda_2 \mathcal{L}_{cls} + \lambda_3 \mathcal{L}_{sim} + \lambda_4 \mathcal{L}_{sparse} + \lambda_5 \mathcal{L}_{jerk}\right] \quad (6)$$

$$\mathcal{L}_D = \frac{1}{2}\,\mathbb{E}_{\mathbf{x}_q}\left[-\log(\mathcal{D}(\mathbf{x}_q))\right] - \mathbb{E}_{\mathbf{x}_q}\left[\log(1 - (\mathcal{D}(\mathbf{x}_q + \mathcal{G}(\mathbf{x}_q))))\right] \quad (7)$$

Generator The generator is realized with a many-to-many sequence prediction model trained to generate modifications to a query sequence. To capture temporal dependencies in the input, different types of sequence models can be chosen, including long short-term memories (LSTMs), gated recurrent units, or temporal convolutional neural networks. Input and output of the generator are of the same shape. Loss functions for generator and discriminator derive from the minimax loss suggested by Goodfellow et al. [2014]. The generator maximizes the discriminator's estimate that the counterfactual is real (Equation 1). One important aspect of the generator is the subtractive dual ReLU [Nair and Hinton, 2010] output in the sparsity layer. Instead of a single linear output the two contrastive outputs allow the network to produce positive and negative residuals while it is still easy to generate exact zero-residuals ($\delta = ReLU(\delta_{pos}) - ReLU(\delta_{neg})$).

Immutable Features In case of immutable features in the original dataset, the generator only produces residuals for all mutable features. In this specific case, the input to the generator is larger than its output. Generated residuals for the mutable features are then likewise added to the respective mutable features in the query sequence. All immutable features of the query instance remain untouched.

Discriminator The discriminator takes on the role of distinguishing between real samples (i.e. samples from the original dataset) and fake samples (i.e. generated counterfactuals). It aims to maximize its estimate that the counterfactual is fake and the query is real (Equation 7). It is implemented as a binary many-to-one sequence classification model with sigmoid activation that takes in a multivariate time series and produces a probability between 0 and 1, indicating whether the given sample looks like a real or fake sample. As the counterfactuals begin to look more realistic, the discriminator's accuracy drops towards 50% (chance).

Classifier Unlike vanilla GANs, a counterfactual GAN needs a third neural network, the classifier. In our approach the classifier is realized with a many-to-one sequence classification model. The classifier is pretrained on the original dataset and learns to classify the label of a sequence. In contrast to Nemirovsky et al. [2020], our classifier does not only distinguish between samples which belong and samples that do not belong to the target class. Instead, we train a full classifier which learns to distinguish all classes in the original dataset. That said, our classifier can either be binary (with sigmoid activation) or multi-class (with softmax activation) in case of two or multiple original class labels, respectively. This property allows us to flexibly alter the desired target class for the generated counterfactuals without retraining the classifier. Moreover, our approach could also simultaneously be trained on all target classes. In this case, generating counterfactuals for different target classes would not require retraining of any network element of our approach.

Regularization The combination of adversarial loss $\mathcal{L}_{adv}$ and classification loss $\mathcal{L}_{cls}$ loss ensures that the generated counterfactual changes the class label, while resembling a sample from the original data distribution. The classification loss between the predicted class for the counterfactual and the target class is derived from the cross-entropy loss (Equation 2). In line with other counterfactual approaches, we apply the $\mathcal{L}_1$ norm as a similarity regularization term $\mathcal{L}_{sim}$ on the generator loss (Equation 3). Importantly, we also use the $\mathcal{L}_0$ norm as a real sparsity constraint $\mathcal{L}_{sparse}$ which ensures that the number of modifications stays low (Equation 4). It was shown that $\mathcal{L}_0$ regularization effectively fosters sparse hidden state updates in RNNs [Gumbsch et al., 2021]. To address the sequentiality of time series, we introduce another regularization term, the *jerk* constraint $\mathcal{L}_{jerk}$. This

term ensures that changes are evenly distributed over time by penalizing large differences between modifications in consecutive time steps (Equation 5). Additional weighting factors λ_{1-5} allow each component of the generator loss to be switched on or off to meet the specific needs of individual datasets. A more fine-grained weighting with weighting factors between 0 and 1 enables a direct influence on the loss balance (Equation 6).

2.1.1 On the Sparsity of Generated Counterfactuals

One key difference of our model in comparison with other counterfactual approaches is the clear distinction between similarity and sparsity. The combination of the sparsity constraint $\mathcal{L}_{sparse}$ and the sparsity layer as part of the generator architecture produces truly sparse counterfactuals with zero-residuals in a number of time steps and features. Importantly, we let the system inherently learn the trade-off between realism, precision, similarity, sparsity and smoothness during the training process. As a consequence, unlike other counterfactual approaches for time series, there is no need to define a fixed number of time steps and features that which may be changed. On the same lines, there is not only one specific section of the series which can be modified. Instead, we demonstrate that our approach identifies and modifies salient time steps and features while leaving most non-salient time steps and features untouched.

3 Experiments

Our approach is evaluated on three different multivariate time series datasets in comparison with three related counterfactual methods. The evaluated tasks comprise two movement datasets for multi-class classification and one synthetic time series interpretability benchmark for binary classification. Human motion datasets are anonymized and cannot be mapped back to individual subjects.

3.1 Datasets

MotionSense: The human motion dataset *MotionSense* [Malekzadeh et al., 2019] (Open Database License ODbL) provides multivariate time series collected by accelerometer and gyroscope sensors of a smartphone stored in a subject's pocket as they perform different actions. Actions include walking downstairs, walking upstairs, sitting, standing, walking and jogging. For this work, we only used active movement sequences and thus excluded sitting and standing trials which yielded a total number of 11194 samples. Each time series was truncated to a length of 100 time steps. All twelve features describing attitude, gravity, rotation and user acceleration are treated as mutable.

Catching: The *Catching* dataset [Lang et al., 2021] (provided by personal permission) contains multivariate two-dimensional movement trajectories of healthy and pathological ball catching trials over 60 time steps. At each time step, 20 features capture the catcher's arm position as well as the position of the ball. Each of the 1975 catching trials is assigned a label indicating the subject's disease status: healthy control, patient with Autism Spectrum Disorder or patient with Spinocerebellar Ataxia. All features specifying the catcher's body posture are defined as mutable features, while the two features describing the ball position are treated as immutable.

Moving Box: The synthetic *Moving Box* dataset was introduced to benchmark interpretability in time series predictions [Ismail et al., 2020]. It portrays a wide range of temporal and spatial properties commonly found in multivariate time series. Each time series spans 50 time steps and 50 features of which only a subset is salient. Samples are assigned a binary label (0: negative class, 1: positive class) and have a defined start and end point of salient time steps and features per sample. In this dataset, all features are mutable. We used a representative subset containing 13950 samples with boxes of different sizes and at varying positions as well as a variety of generating time series processes.

3.2 Approaches

ICS: We loosely follow Wachter et al. [2017] for an implementation of an iterative counterfactual search algorithm. Each counterfactual is initialized with a random uniform distribution between the minimum and maximum values of the query sequence. The class of the generated counterfactual is predicted using a pretrained classifier. We use the $\mathcal{L}_2$ distance to measure the classification loss and the unweighted $\mathcal{L}_1$ norm to enforce similarity.

All following approaches are based on GANs combined with a pretrained classifier. To account for temporal dependencies in the data, generator and discriminator are implemented as bidirectional LSTMs [Hochreiter and Schmidhuber, 1997]. The generator is a two-layer many-to-many bidirectional LSTM with 256 hidden neuron and dropout of 0.4. The discriminator is built up as a one-layer many-to-one bidirectional LSTM with 16 hidden neurons, sigmoid output activation and dropout of 0.4. For both networks, the final LSTM layer is followed by a fully-connected output layer.

GAN: This approach consists of a counterfactual LSTM-GAN producing complete counterfactuals based on query sequences. The fully-connected output layer of the generator is followed by a tanh activation. The generator loss is regularized using the $\mathcal{L}_1$ norm to optimize the distance between counterfactual and query.

CounteRGAN: This approach is a time series specific implementation of Nemirovsky et al. [2020] and implements an LSTM generator that produces residuals based on query sequences. All other aspects of the implementation are equal to the GAN approach.

SPARCE: Our approach likewise generates residuals instead of complete counterfactuals. In comparison to CounteRGAN, we additionally regularize the generator loss via sparsity and smoothness constraints (cf. Section 2.1). Moreover, we add weighting factors λ_{1-5} to enable the (de-)activation of single regularization constraints if required. Most importantly, the LSTM generator implemented in our approach does not conclude with a linear or tanh activation layer, but instead uses a custom sparsity layer of two interoperating ReLU activations (cf. Section 2.1).

3.3 Evaluation Metrics

Realism: In line with Yoon et al. [2019], we use *t-distributed stochastic neighbor embedding* (*t-SNE*) for a visual assessment of the in-distributionness of the generated counterfactuals [Van der Maaten and Hinton, 2008]. We separately plot query and target samples of the original dataset along with the counterfactuals generated by each approach to determine whether the generated counterfactuals rather resemble queries or targets.

Precision: Classification error of generated counterfactuals is measured by the $\mathcal{L}_2$ norm between the classifier's prediction for a counterfactual sequence and the target class. The metric is indicated as the average distance across all test samples. The lower the metric, the higher the precision of the counterfactual approach. A precision value of 0.0 means that all generated counterfactuals were correctly classified as the target class.

Similarity: The $\mathcal{L}_1$ distance between each query and the corresponding counterfactual is used to assess similarity. The metric is averaged over all test samples and normalized using the number of time steps and features in the dataset. Lower values indicate higher mean proximity of the generated counterfactuals to the corresponding queries.

Sparsity: Generated counterfactuals of each approach are evaluated on the number of modified time steps and features to transform the query into the counterfactual using the $\mathcal{L}_0$ norm between queries and corresponding counterfactual examples. Values are averaged and normalized in the same way as the similarity metric. Here lower values represent higher average sparsity, i.e. fewer modifications in the time and feature dimensions. In the case of immutable features in the dataset, the sparsity metric is only computed on all mutable features. As a consequence, the maximum sparsity value equals 1.0 indicating that all features in all time steps have been modified in each counterfactual.

Smoothness: This time series specific metric is assessed with the $\mathcal{L}_2$ distance between modifications of consecutive time steps. High values indicate large differences between modifications in subsequent steps. Lower values represent modifications that are more smoothly distributed over the course of the sequence. This metric is likewise averaged across all samples and normalized using the number of time steps and features.

4 Results

4.1 Quantitative Evaluation

All results are reported on the held-out subsets for testing (20% of each dataset). Unless otherwise stated, results are averaged over five repetitions with random seeds. The target class for counterfactuals

Table 1: Quantitative results for all datasets

Dataset	Measure	ICS	GAN	CounteRGAN	SPARCE (Ours)
Catching	Precision	0.24 ± .05	**0.00** ± .00	**0.00** ± .00	0.01 ± .01
	Similarity	1.66 ± .04	0.22 ± .02	0.12 ± .01	**0.09** ± .04
	Sparsity	1.00 ± .00	1.00 ± .00	1.00 ± .00	**0.27** ± .10
	Smoothness	0.55 ± .01	0.07 ± .01	**0.01** ± .01	**0.01** ± .01
MotionSense	Precision	0.37 ± .07	**0.00** ± .00	**0.00** ± .01	0.04 ± .06
	Similarity	1.32 ± .01	0.71 ± .21	0.33 ± .09	**0.22** ± .13
	Sparsity	1.00 ± .00	1.00 ± .00	1.00 ± .00	**0.22** ± .14
	Smoothness	0.58 ± .00	0.09 ± .01	**0.03** ± .02	0.04 ± .03
Moving Box	Precision	0.99 ± .00	**0.00** ± .00	0.01 ± .01	**0.00** ± .00
	Similarity	1.32 ± .00	0.87 ± .17	0.59 ± .05	**0.40** ± .06
	Sparsity	1.00 ± .00	1.00 ± .00	1.00 ± .00	**0.30** ± .05
	Smoothness	0.29 ± .00	0.12 ± .01	0.03 ± .00	**0.02** ± .00

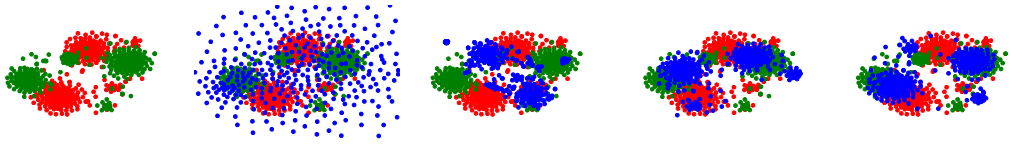

Figure 3: t-SNE plots for *Moving Box* dataset. Queries are plotted in red, targets in green and generated counterfactuals in blue.

is healthy control for *Catching*, walking upstairs for *MotionSense* and class 1 for *Moving Box*. ICS is performed for 100 steps ($\lambda_{init} = 1.0$, max. λ steps = 10). The loss is minimized with Adam [Kingma and Ba, 2015] optimization ($lr = 0.4, \beta_1 = 0.9, \beta_2 = 0.999$). All GAN-based approaches are trained for 100 epochs in batches of 32 samples using Adam optimization ($lr = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$). For all quantitative metrics, lower values represent better performance. The best value for each metric is printed in bold numbers.

Considering the *Catching* dataset, GAN and CounteRGAN achieve a precision of 100%, however closely followed by our approach (Table 1). SPARCE outperforms ICS, GAN and CounteRGAN on the similarity and sparsity of generated counterfactuals and shares the best smoothness value with the CounteRGAN approach. It can be seen that no tested approach besides ours can generate sparse counterfactuals. This observation also holds for the *MotionSense* and *Moving Box* datasets. SPARCE reaches the best or second-best performance on each metric in spite of making considerably sparser modifications than the other approaches.

4.2 Realism

We qualitatively assess the in-distributionness of generated counterfactuals for the synthetic *Moving Box* dataset via t-SNE visualization ($components = 2, perplexity = 4.4, iterations = 300$). In Figure 3, the first subplot illustrates the distribution of queries and targets in the original dataset. The remaining subplots additionally show the distribution of counterfactuals generated by the respective approaches. While counterfactuals generated by ICS lie within but also largely out of the original distribution, those generated by GAN form separate groups next to queries and targets. Since the task of counterfactual search is to find samples that modify a query sample to look like a target, counterfactuals generated by CounteRGAN and SPARCE show the most promising distributions. Indeed, counterfactuals of both approaches modify queries in a way that the resulting sequences approximate and even overlap with target samples.

4.3 Saliency

Since salient features and time steps are known upfront for the synthetic *Moving Box* dataset, we compare the overlap with time steps modified by each approach. A perfect counterfactual would only modify salient inputs. We first visually compare modifications for queries with boxes of different

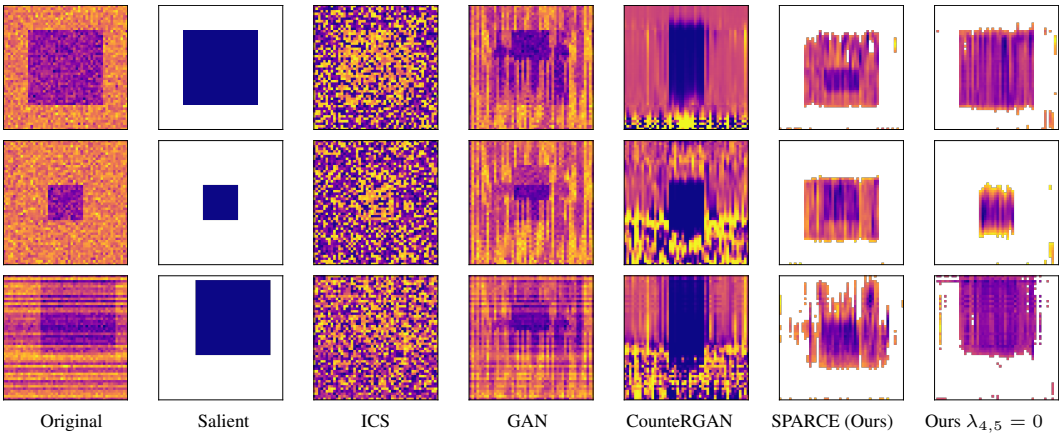

Figure 4: Predefined salient inputs vs. counterfactual modifications for the *Moving Box* dataset.

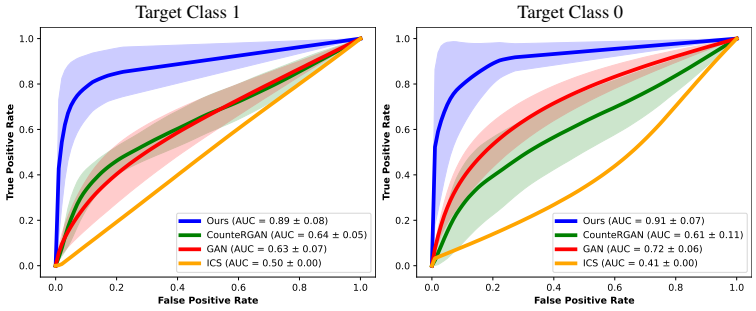

Figure 5: Mean ROC curve measuring the overlap between predefined salient inputs and counterfactual modifications for the *Moving Box* dataset. Shaded areas describe one standard deviation of the mean ROC.

sizes and positions (Figure 4). In all heatmaps, the x-axis represents the feature axis and time is on the y-axis. In the second column, the salient features and time steps corresponding to each query are shown in color. All remaining sub-figures demonstrate the modifications to the queries. White spaces are zero-residuals (i.e. sparse time steps and features without modifications). Darker colors indicate stronger modifications.

ICS largely fails to identify salient points in the input. All GAN-based methods detect the position of most salient inputs. However, GAN and CounteRGAN additionally modify non-salient inputs. In contrast, SPARCE modifies far fewer inputs overall and focuses on salient inputs. For this dataset, the performance of our approach can be further improved by switching off $\lambda_{4,5}$, i.e. sparsity and jerk regularization. This shows that sparsity is primarily induced by the sparsity layer. It also demonstrates that the application of the jerk constraint depends on the problem. Here, a clear value increase marks the transition from non-salient to salient inputs. In human motion datasets, in contrast, smooth movements are natural and desired.

We furthermore assess the salience overlap in a quantitative manner via the receiver operating characteristic (ROC) curve in combination with the area under the curve (AUC) score. Higher AUC scores indicate better discrimination performance between salient and non-salient inputs. Figure 5 visualizes mean ROC curves over five repetitions for both target classes. In both cases, we see that our approach produces counterfactuals that show a substantially higher overlap with predefined salient inputs than other approaches. Visual and quantitative evaluation therefore demonstrates that our approach creates sparse counterfactual explanations and is also suitable for the identification of salient inputs in multivariate time series.

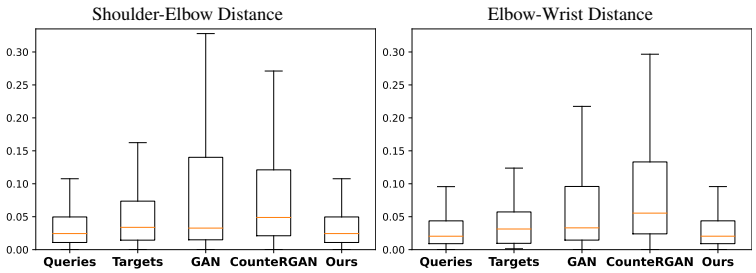

Figure 6: Measured distances between body parts for originals and counterfactuals for the *Catching* dataset. Orange horizontal lines denote the median distance. Boxes contain all values between lower and upper quartiles.

4.4 Geometric Plausibility

To assess geometric plausibility of the *Catching* dataset, we compute the Euclidean distances between body parts for the original dataset and the generated counterfactuals (Figure 6). ICS is excluded from the figure, since the corresponding values lie outside of the displayed area. Counterfactuals generated by our approach most closely resemble the body-part distances found in the original data. This can indicate higher geometric plausibility of our generated counterfactuals. In order to fully inspect geometric plausibility, however, the angles at which the joints are positioned in relation to one another would also have to be examined.

5 Conclusions

We proposed GAN architecture for generating time- and feature-sparse counterfactual explanations for multivariate time series. Our approach extends previous methods by a custom sparsity layer and additional loss regularization for sparsity and smoothness. In extensive experiments, we demonstrate that in spite of making substantially sparser modifications SPARCE achieves comparable or superior performance on common metrics for counterfactual search. Benchmarking our approach on a synthetic interpretability dataset, we show that it can also be used for feature attribution. The application to real-world human motion datasets demonstrates that our approach generates sparser and more plausible counterfactuals than related approaches.

The design of our approach allows for a flexible change of the desired target class, as well as an easy adaptation of the counterfactual value function catering to the needs of other applications. Future extensions can consider other applications (e.g. weather, stocks) and domain-specific regularization terms. In critical sectors such as healthcare, misinterpretation of systems can have severe consequences. XAI systems should thus always be validated by human experts. To enhance the understandability of generated counterfactuals, further work can investigate the visualization of explanations for end-users (e.g. in textual or visual form).

Acknowledgments and Disclosure of Funding

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) and the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes) for supporting Jana Lang. Martin Giese received support from the BMG project SStepKiZ and the EU ERC SYNERGY Grant RELEVANCE.

References

E. Ates, B. Aksar, V. J. Leung, and A. K. Coskun. Counterfactual explanations for multivariate time series. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–8. IEEE, 2021.

S. Dandl, C. Molnar, M. Binder, and B. Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pages 448–469. Springer, 2020.

E. Delaney, D. Greene, and M. T. Keane. Instance-based counterfactual explanations for time series classification. In *International Conference on Case-Based Reasoning*, pages 32–47. Springer, 2021.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

C. Gumbsch, M. V. Butz, and G. Martius. Sparsely changing latent states for prediction and planning in partially observable domains. *Advances in Neural Information Processing Systems*, 34, 2021.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

A. A. Ismail, M. Gunady, H. Corrada Bravo, and S. Feizi. Benchmarking deep learning interpretability in time series predictions. *Advances in neural information processing systems*, 33:6441–6452, 2020.

A.-H. Karimi, B. Schölkopf, and I. Valera. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 353–362, 2021.

M. T. Keane and B. Smyth. Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In *International Conference on Case-Based Reasoning*, pages 163–178. Springer, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

J. Lang, M. A. Giese, M. Synofzik, W. Ilg, and S. Otte. Early recognition of ball catching success in clinical trials with rnn-based predictive classification. In *International Conference on Artificial Neural Networks*, pages 444–456. Springer, 2021.

M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi. Mobile sensor data anonymization. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, IoTDI '19, pages 49–58, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6283-2. doi: 10.1145/3302505.3310068. URL `http://doi.acm.org/10.1145/3302505.3310068`.

R. K. Mothilal, A. Sharma, and C. Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 607–617, 2020.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

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

L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

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

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

J. Yoon, D. Jarrett, and M. Van der Schaar. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems*, 32, 2019.

A Appendix

A.1 Additional Results

A.1.1 Saliency Heatmaps

Figure 7 visualizes heatmaps for four additional queries from the *Moving Box* dataset in comparison to counterfactual modifications made by ICS, GAN, CounteRGAN and our approach (SPARCE). In this dataset, salient inputs are known in advance. A reasonable counterfactual explanation shows a high overlap of non-zero modifications and salient inputs (second column), as well as zero-modifications and non-salient inputs. In the figure, zero-residuals are plotted in white. The figure demonstrates that our approach generates sparse counterfactuals with a clear distinction between salient and non-salient time steps and features. The modifications made by our approach show a high overlap with the original saliency heatmap. Other approaches also partially recognize salient inputs, but modify salient as well as non-salient inputs. Although these approaches also successfully change the class label and remain similar to the query, they fail to create sparse counterfactuals. As pointed out in the main paper, sparsity is a key factor in ensuring the actionability of counterfactual explanations.

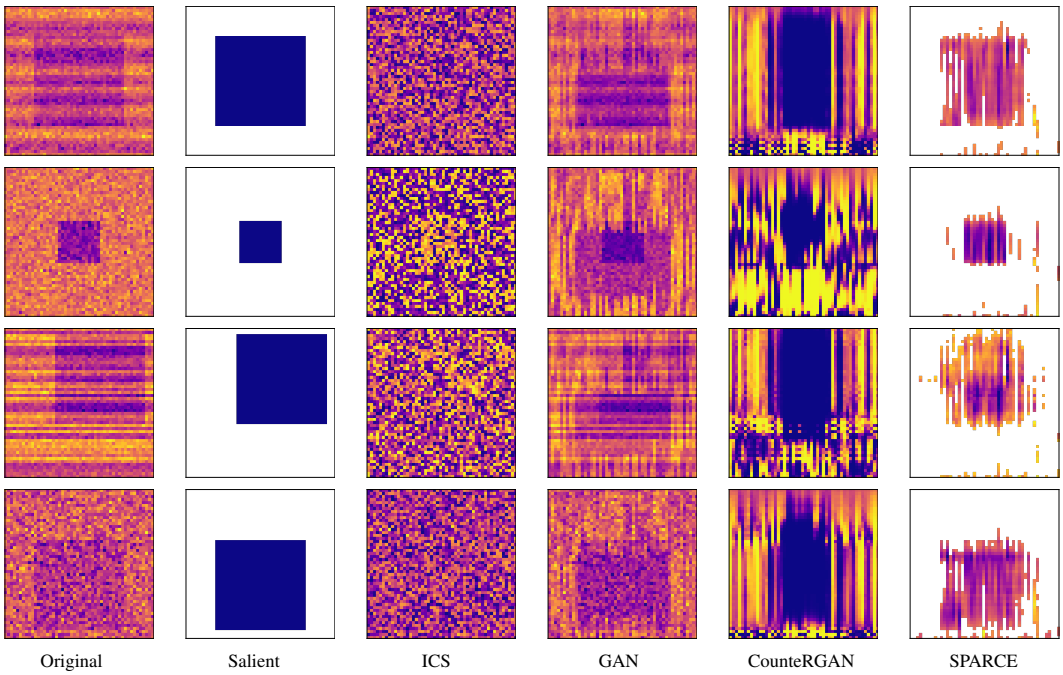

Figure 7: Visualization of predefined salient inputs in comparison to counterfactual modifications for the *Moving Box* dataset.

A.1.2 Influence of Different Target Classes

In contrast to other approaches, our classifier is not trained to distinguish samples belonging to the target class from other samples. Instead, it learns to discriminate all classes that are present in the original dataset. As a consequence, we can flexibly switch the target class for counterfactuals without retraining the classifier. Table 2 denotes the performance of our approach on all metrics for the four action categories of *MotionSense* (dws: walking downstairs, ups: walking upstairs, wlk: walking, jog: jogging). Our approach is able to generate counterfactuals with high informative value regardless of the selected target class. The results for *Catching* and *Moving Box* datasets are provided in Tables 3, 4.

A.1.3 Influence of Regularization Terms

In addition to the similarity constraint used in most related approaches, we apply sparsity and jerk regularization. To investigate the importance of these additional factors, we report results of experimental runs activating either only $\mathcal{L}_{adv}$, $\mathcal{L}_{cls}$ and $\mathcal{L}_{sim}$ ($\lambda_{1,2,3} = 1$), or additionally $\mathcal{L}_{sparse}$ ($\lambda_{1,2,3,4} = 1$), or additionally $\mathcal{L}_{jerk}$ ($\lambda_{1,2,3,5} = 1$) or all regularization terms ($\lambda_{1,2,3,4,5} = 1$). As shown in Table 5 the sparsest counterfactuals for

Table 2: Influence of different target classes for *MotionSense*

$\mathbf{c}_t$:	dws	ups	wlk	jog
Precision	$0.04 \pm .04$	$0.04 \pm .06$	$0.19 \pm .08$	$0.04 \pm .07$
Similarity	$0.21 \pm .10$	$0.22 \pm .13$	$0.21 \pm .09$	$0.24 \pm .09$
Sparsity	$0.35 \pm .15$	$0.22 \pm .14$	$0.42 \pm .17$	$0.45 \pm .15$
Smoothness	$0.02 \pm .01$	$0.04 \pm .03$	$0.03 \pm .01$	$0.03 \pm .01$

Table 3: Influence of different target classes for *Catching*

$\mathbf{c}_t$:	Ataxia	Autism	Control
Precision	0.11	0.04	0.01
Similarity	0.15	0.13	0.09
Sparsity	0.47	0.42	0.27
Smoothness	0.01	0.01	0.01

Table 4: Influence of different target classes for *Moving Box*

$\mathbf{c}_t$:	0	1
Precision	0.00	0.00
Similarity	0.52	0.40
Sparsity	0.33	0.30
Smoothness	0.02	0.02

the *Catching* dataset are generated activating all five loss terms. However, the most significant difference in sparsity between other approaches and ours does not come from the $\mathcal{L}_1$ sparsity regularization, but from the sparsity layer. The meaningfulness of jerk regularization can vary between datasets and can - as shown here - easily be switched off if high differences between features in subsequent time steps are expected or even desired. Results for the other datasets are reported in Tables 6, 7.

Table 5: Influence of regularization terms for *Catching*

	$\lambda_{1,2,3} = 1$	$\lambda_{1,2,3,4} = 1$	$\lambda_{1,2,3,5} = 1$	$\lambda_{1,2,3,4,5} = 1$
Precision	$\mathbf{0.01} \pm .01$	$\mathbf{0.01} \pm .01$	$0.02 \pm .03$	$\mathbf{0.01} \pm .01$
Similarity	$\mathbf{0.08} \pm .02$	$0.14 \pm .04$	$0.10 \pm .03$	$0.09 \pm .04$
Sparsity	$0.32 \pm .06$	$0.33 \pm .13$	$0.28 \pm .14$	$\mathbf{0.27} \pm .10$
Smoothness	$\mathbf{0.01} \pm .00$	$0.02 \pm .01$	$\mathbf{0.01} \pm .01$	$\mathbf{0.01} \pm .01$

Table 6: Influence of regularization terms for *MotionSense*

	$\lambda_{1,2,3} = 1$	$\lambda_{1,2,3,4} = 1$	$\lambda_{1,2,3,5} = 1$	$\lambda_{1,2,3,4,5} = 1$
Precision	$\mathbf{0.01} \pm .01$	$0.03 \pm .04$	$0.03 \pm .04$	$0.04 \pm .06$
Similarity	$0.25 \pm .15$	$0.23 \pm .11$	$0.29 \pm .13$	$\mathbf{0.22} \pm .13$
Sparsity	$\mathbf{0.22} \pm .11$	$\mathbf{0.22} \pm .08$	$0.32 \pm .16$	$\mathbf{0.22} \pm .14$
Smoothness	$0.06 \pm .04$	$\mathbf{0.03} \pm .03$	$\mathbf{0.03} \pm .03$	$0.04 \pm .03$

Table 7: Influence of regularization terms for *Moving Box* (only one repetition for columns 1-3)

	$\lambda_{1,2,3} = 1$	$\lambda_{1,2,3,4} = 1$	$\lambda_{1,2,3,5} = 1$	$\lambda_{1,2,3,4,5} = 1$
Precision	**0.00**	**0.00**	**0.00**	$\mathbf{0.00} \pm .00$
Similarity	0.39	**0.38**	0.42	$0.40 \pm .06$
Sparsity	**0.26**	0.35	0.29	$0.30 \pm .05$
Smoothness	**0.02**	**0.02**	**0.02**	$\mathbf{0.02} \pm .00$

A.2 Dataset Statistics

All datasets used in this work contain multivariate time series. Sequences are either truncated to equal lengths or are of equal length by design. Table 8 denotes the number of samples per class, the total number of samples and the number of features and time steps for each dataset. Note that for *MotionSense*, classes 2 and 3 (sitting and standing) are excluded in all experiments.

Table 8: Samples per class for all datasets

	Samples	Time steps	Features	$\mathbf{c}_t$=0	$\mathbf{c}_t$=1	$\mathbf{c}_t$=2	$\mathbf{c}_t$=3	$\mathbf{c}_t$=4	$\mathbf{c}_t$=5
Catching	1 975	60	20	522	789	664	-	-	-
MotionSense	13 950	100	12	1 283	1 317	3 364	3 042	1 536	3 408
Moving Box	17 600	50	50	8 806	8 794	-	-	-	-

A.3 Computing Times

All experiments are conducted on a Dell Precision 7920 Tower computer with an Intel Xeon Silver processor running at 2.10 GHz, 64 GB RAM, Windows 10 and an NVIDIA Quadro RTX 5000 GPU. Computing times include one complete training run (for GAN-based models) and the generation of counterfactuals for the entire test set (Table 9).

Table 9: Computing times (in seconds) for all dataset

	ICS	GAN	CounteRGAN	Ours
Catching	693 ± 34	432 ± 5	435 ± 7	398 ± 4
MotionSense	$2\,678 \pm 13$	$1\,483 \pm 21$	$1\,660 \pm 23$	$1\,669 \pm 32$
Moving Box	$1\,755 \pm 5$	$22\,789 \pm 1\,071$	$22\,290 \pm 1\,198$	$21\,138 \pm 914$