Right on Time: Revising Time Series Models by Constraining their Explanations

Maurice Kraus[1] *(✉), David Steinmann[1,2] *, Antonia Wüst[1], Andre Kokozinski[5], and Kristian Kersting[1,2,3,4]

[1] Artificial Intelligence and Machine Learning Group, TU Darmstadt
{maurice.kraus,david.steinmann}@cs.tu-darmstadt.de
[2] Hessian Center for Artificial Intelligence (hessian.AI), Darmstadt
[3] Centre for Cognitive Science, TU Darmstadt
[4] German Center for Artificial Intelligence (DFKI)
[5] Institute for Production Engineering and Forming Machines, TU Darmstadt

Abstract. Deep time series models often suffer from reliability issues due to their tendency to rely on spurious correlations, leading to incorrect predictions. To mitigate such shortcuts and prevent "Clever-Hans" moments in time series models, we introduce Right on Time (RioT), a novel method that enables interacting with model explanations across both the *time* and *frequency* domains. By incorporating feedback on explanations in both domains, RioT constrains the model, steering it away from annotated spurious correlations. This dual-domain interaction strategy is crucial for effectively addressing shortcuts in time series datasets. We empirically demonstrate the effectiveness of RioT in guiding models toward more reliable decision-making across popular time series classification and forecasting datasets, as well as our newly recorded dataset with naturally occuring shortcuts, P2S, collected from a real mechanical production line.

1 Introduction

Time series data is ubiquitous in today's world. Everything that is measured over time generates some form of time series, for example, energy load [15], sensor measurements in industrial machinery [21] or recordings of traffic data [18]. Complex time series data is often analyzed using various neural models [3,28]. However, as in other domains, these can be subject to spurious factors ranging from simple noise or artifacts to complex shortcuts [16]. Intuitively, a shortcut, also called "Clever-Hans" moment, is a spurious pattern in the data that correlates with the target task during training but lacks true relevance. If a model learns to rely on such patterns rather than meaningful features, its generalizability suffers, performing well on data with the shortcut but failing on data without it, which poses a significant challenge in real-world deployment [11]. While model explanations can help uncover these shortcuts, they do not resolve the issue on

*These authors share equal contribution.

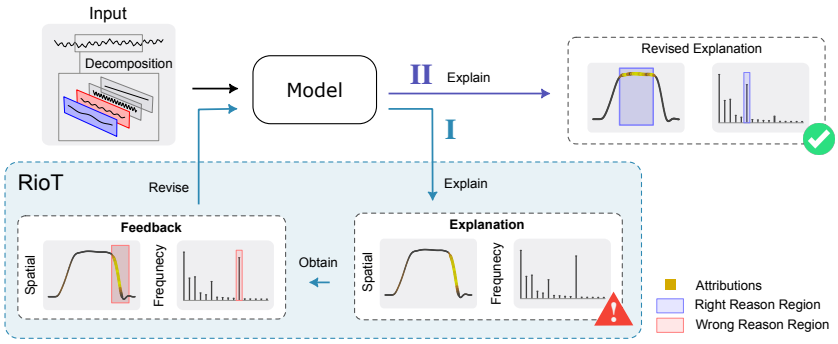

Fig. 1. Explanations reveal that the model relies on spurious factors in the input (red region) instead of relevant features (blue region). With RioT, the model can be guided away from these misleading patterns, whether they appear in the *spatial* or *frequency* domain. For this, RioT leverages feedback on incorrect explanations to steer the model toward more meaningful and reliable reasoning.

their own (cf. Fig. 1 **I**). Despite extensive research in other domains [36], shortcuts in time series models remain underexplored. Existing studies often have specific assumptions about settings and data [4], leaving a gap in understanding and mitigating shortcut learning in broader time series applications. To address this, we introduce Right on Time (RioT), a new method grounded in the principles of explanatory interactive learning (XIL) [38], which leverages feedback on explanations to mitigate shortcuts (cf. Fig. 1 **II**). RioT uses traditional explanation methods, such as Integrated Gradients (IG) [37], to assess whether the model attends to the correct time steps. It then incorporates feedback on shortcut areas to refine the model, improving robustness and generalization.

However, spurious factors in time series data extend beyond the time domain. For example, a consistent noise frequency in an audio signal can act as a shortcut without being tied to a specific point in time. RioT can handle these types of shortcuts by incorporating feedback in the frequency domain. To highlight the importance of shortcuts in time series data, we introduce a new real-world dataset with naturally occurring shortcuts, called Production Press Sensor Data (P2S). The dataset includes sensor measurements from an industrial high-speed press, essential to many manufacturing processes in the sheet metal working industry. The sensor data for detecting faulty production contains shortcuts and thus provokes incorrect predictions after training. Next to its industrial relevance, P2S is the first time series dataset that contains explicitly annotated shortcuts, enabling the evaluation of mitigation strategies on real data.

Altogether, we make the following contributions: (1) We show both on our newly introduced real-world dataset P2S and on several other datasets with manual shortcuts that SOTA neural networks on time series classification and forecasting are affected by these shortcuts. (2) We introduce RioT to mitigate shortcuts for time series data. The method can incorporate feedback on the time domain and the frequency domain. (3) By incorporating explanations and feed-

back in the frequency domain, we enable a new perspective on XIL, overcoming the important limitation that shortcuts must be spatially separable.

The paper is structured as follows: Sec. 2 provides a brief overview of related work on explaining time series and correcting model mistakes. Sec. 3 introduces our approach, while Sec. 4 describes our decoy methods and P2S. We then present a detailed evaluation and discussion in Sec. 5. Finally, Sec. 6 concludes the paper and outlines directions for future research.

2 Related Work

Explanations for Time Series. Explainable artificial intelligence offers various techniques to interpret machine learning models, many of which originated in image or text data before being adapted for time series [26]. Attribution methods explain models directly in the input space, while approaches like symbolic aggregation [17] and shapelets [42] provide higher-level insights (cf. [26,29] for a broader discussion on time series explanations). While explanation methods help identify shortcuts, they alone do not enable model revision. Thus, our approach begins with explanations to detect shortcuts and integrates feedback to mitigate them. Specifically, we use Integrated Gradients (IG) [37], which computes attributions via model gradients and is widely used for time series data [22,39].

Explanatory Interactive Learning (XIL). Research on shortcuts and their mitigation is growing, though it primarily focuses on visual data [36]. One direction is explanatory interactive learning (XIL), which entails methods that revise a model's decision-making based on human feedback [30,38]. A core aspect of XIL is using model explanations to correct mistakes, particularly to prevent Clever-Hans-like behavior, where models rely on spurious shortcuts [9,35]. Several XIL methods have been applied to image data. Right for the Right Reasons (RRR) [27] and Right for Better Reasons [32] penalize incorrect attributions, while HINT [31] rewards correct focus and [10] explore using multiple explainers. Despite their success in vision tasks, XIL approaches remain largely unexplored for time series. To address this, we introduce RioT, adapting XIL principles to the unique challenges of time series data.

Unconfounding Time Series. Apart from interactive learning approaches, some methods address confounding in time series models through causal inference [8]. Techniques like the Time Series Deconfounder [4], SqeDec [13], and LipCDE [5] estimate data while mitigating confounders in covariates of the target variable. They rely on causal analysis and specific assumptions about data generation. In contrast, our method focuses on shortcuts within the target variable itself, requiring no assumptions beyond the shortcut being detectable in model explanations - an area where existing causal methods are less applicable.

3 Right on Time (RioT)

The core idea of Right on Time (RioT) is to use feedback on model explanations to guide the model away from incorrect reasoning. Following the XIL

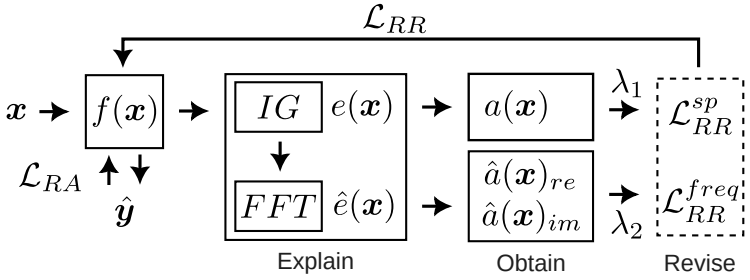

Fig. 2. RioT's explanation-based revision process. Input data $\boldsymbol{x}$ passes through the model $f(\boldsymbol{x})$ to generate explanations $e(\boldsymbol{x})$, receives annotated feedback $a(\boldsymbol{x})$, and is then fed back into the model. Integrated Gradients (IG) provides spatial explanations, while Fourier Transform (FFT) converts them into frequency-based explanations. Annotations can be applied in either or both domains and are leveraged by the right-reason loss ($\mathcal{L}_{RR}^{sp}$ and $\mathcal{L}_{RR}^{freq}$) to steer the model away from shortcuts in time or frequency.

paradigm, RioT is designed for seamless integration with other XIL methods. To ensure compatibility, we structure RioT around the four key steps identified by [9]: *Select*, *Explain*, *Obtain* and *Revise*. In *Select*, samples for feedback and model revision are selected. Following previous methods, we select all samples by default but also explore using subsets of the data. Afterwards, *Explain* covers model explanations before feedback is provided in *Obtain*. Lastly, in *Revise*, the feedback is integrated into the model to overcome the shortcuts. We introduce RioT along these steps in the following (as illustrated in Fig. 2).

Given a dataset $(\mathcal{X}, \mathcal{Y})$ and a model $f(\cdot)$ for time series classification or forecasting. The dataset consists of D many pairs of $\boldsymbol{x}$ and $\boldsymbol{y}$. Thereby, $\boldsymbol{x} \in \mathcal{X}$ is a time series of length T, i.e., $\boldsymbol{x} \in \mathbb{R}^T$. For K class classification, the ground-truth output is $\boldsymbol{y} \in \{1, \ldots, K\}$ and for forecasting, the ground-truth output is the forecasting window $\boldsymbol{y} \in \mathbb{R}^W$ of length W. The ground-truth output of the full dataset is described as $\mathcal{Y}$ in both cases. For a datapoint $\boldsymbol{x}$, the model generates the output $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$, where the dimensions of $\hat{\boldsymbol{y}}$ are the same as of $\boldsymbol{y}$.

3.1 Explain

Given a pair of input $\boldsymbol{x}$ and model output $\hat{\boldsymbol{y}}$ for time series classification, the explainer generates an explanation $e_f(\boldsymbol{x}) \in \mathbb{R}^T$ in the form of attributions to explain $\hat{\boldsymbol{y}}$ w.r.t. $\boldsymbol{x}$, where a large attribution value means a large influence on the output. In the remainder of the paper, explanations refer to the model f, but we drop f from the notation to declutter it, resulting in $e(\boldsymbol{x})$. We use IG [37] (Eq. 1) as an explainer, an established gradient-based attribution method. This method integrates the gradient along the path (using the integration variable α) from a baseline $\bar{\boldsymbol{x}}$ to the input x. It multiplies the result with the difference between baseline and input. However, we make some adjustments to the base method to make it more suitable for time series and model revision, namely taking the absolute value of the difference between $\boldsymbol{x}$ and $\bar{\boldsymbol{x}}$ (further details in

Appx. A.1). In the following, we introduce the modifications to use attributions for forecasting and to obtain explanations in the frequency domain.

$$e(\boldsymbol{x}) = |\boldsymbol{x} - \bar{\boldsymbol{x}}| \cdot \int_0^1 \frac{\partial f(\tilde{\boldsymbol{x}})}{\partial \tilde{\boldsymbol{x}}}\bigg|_{\tilde{\boldsymbol{x}}=\bar{\boldsymbol{x}}+\alpha(\boldsymbol{x}-\bar{\boldsymbol{x}})} d\alpha \quad (1) \qquad e(\boldsymbol{x}) = \frac{1}{W}\sum_{i=1}^{W} e'_i(\boldsymbol{x}) \quad (2)$$

Attributions for Forecasting. In a classification setting, attributions are generated by propagating gradients back from the model output (of its highest activated class) to the model inputs. However, there is often no single model output in time series forecasting. Instead, the model simultaneously generates one output for each timestep of the forecasting window. Naively, one could use these W outputs and generate as many explanations $e'_1(\boldsymbol{x}), \dots e'_W(\boldsymbol{x})$, where each $e'_i(\boldsymbol{x})$ is the IG explanation using the i-th time step from the forecasting window as a target instead of a classification label. This number of explanations would, however, make it even harder for humans to interpret the results, as the size of the explanation increases with W [23]. Therefore, we propose aggregating the individual explanations by averaging in Eq. 2. Averaging attributions over the forecasting window provides a simple yet robust aggregation of the explanations. Other means of combining them, potentially even weighted based on distance of the forecast in the future are also imaginable. Overall, this allows attributions for time series classification and forecasting to be generated similarly.

Attributions in the Frequency Domain. Time series data is often given in the frequency representation, and this format can sometimes be more intuitive for humans to understand than the typical spatial representation. Thus, providing explanations in this domain is essential. [40] showed how to obtain frequency attributions of the method Layerwise Relevance Propagation [1], even if the model does not operate directly on the frequency domain. We adapt this idea to IG: for an input sample $\boldsymbol{x}$, we generate attributions with IG, resulting in $e(\boldsymbol{x}) \in \mathbb{R}^T$ (Eq. 1 for classification or Eq. 2 for forecasting). We then interpret the explanation as a time series, with the attribution scores as values. To obtain the frequency explanation, we perform a Fourier transformation of $e(\boldsymbol{x})$, resulting in the frequency explanation $\hat{e}(\boldsymbol{x}) \in \mathbb{C}^T$ with $\hat{E}$ for the entire set.

3.2 Obtain

The next step of RioT is to obtain feedback on shortcuts. For an input $\boldsymbol{x}$, feedback marks input parts via a binary mask $a(\boldsymbol{x}) \in \{0,1\}^T$, where a 1 signals a potential shortcut at this time step. Thereby, masks $a(\boldsymbol{x}) = (0, \dots, 0)^T$ corresponds to no feedback for a sample. Similarly, feedback can also be given on the frequency explanation, marking which elements in the frequency domain are potential shortcuts. The resulting feedback mask $\hat{a}(\boldsymbol{x}) = (\hat{a}(\boldsymbol{x})_{re}, \hat{a}(\boldsymbol{x})_{im})$ can be different for the real $\hat{a}(\boldsymbol{x})_{re} \in \{0,1\}^T$ and imaginary part $\hat{a}(\boldsymbol{x})_{im} \in \{0,1\}^T$. For the whole dataset, we then have spatial annotations A and frequency annotations $\hat{A}$. Obtaining annotated feedback masks can become costly, particularly if the feedback comes from human experts. However, as shortcuts often occur sys-

tematically, it can be possible to apply annotations to many samples, drastically reducing the number of annotations required in practice.

3.3 Revise

The last step of RioT is integrating the feedback into the model. We apply the general idea of using a loss-based model revision [27,30,35] based on the explanations and the annotation mask. Given the input data $(\mathcal{X}, \mathcal{Y})$, we define the original task (or right-answer) loss as $\mathcal{L}_{RA}(\mathcal{X}, \mathcal{Y})$. This loss measures the model performance and is the primary learning objective. To incorporate the feedback, we further use the right-reason loss $\mathcal{L}_{RR}(A, E)$. This loss aligns model explanations $E = \{e(\boldsymbol{x}) | \boldsymbol{x} \in \mathcal{X}\}$ and user feedback A by penalizing the model for explanations in the annotated areas. To achieve model revision and a good task performance, both losses are combined, where λ is a hyperparameter to balance both parts of the combined loss $\mathcal{L}(\mathcal{X}, \mathcal{Y}, A, E) = \mathcal{L}_{\mathrm{RA}}(\mathcal{X}, \mathcal{Y}) + \lambda \mathcal{L}_{\mathrm{RR}}(A, E)$. Together, the combined loss simultaneously optimizes the primary training objective (e.g. accuracy) and feedback alignment.

Time Domain Feedback. Masking parts of the time domain is an easy way to mitigate spatially locatable shortcuts (Fig. 1, left). We use the explanations E and annotations A in the spatial version of the right-reason loss:

$$\mathcal{L}_{RR}^{sp}(A, E) = \frac{1}{D} \sum_{\boldsymbol{x} \in \mathcal{X}} (e(\boldsymbol{x}) * a(\boldsymbol{x}))^2 \tag{3}$$

As the explanations and the feedback masks are element-wise multiplied, this loss minimizes the explanation values in marked parts of the input. This effectively trains the model to disregard the marked parts for its computation. Thus, using the loss in Eq. 3 as right-reason component for the combined loss allows to effectively steer the model away from points or intervals in time.

Frequency Domain Feedback. However, feedback in the time domain is insufficient to handle every type of shortcut. If the shortcut is not locatable in time, giving spatial feedback cannot be used to revise the models' behavior. Therefore, we utilize explanations and feedback in the frequency domain to tackle shortcuts like in Fig. 1, (right). Given the frequency explanations $\hat{E}$ and annotations $\hat{A}$, the right-reason loss for the frequency domain is:

$$\mathcal{L}_{RR}^{fr}(\hat{A}, \hat{E}) = \frac{1}{D} \sum_{\boldsymbol{x} \in \mathcal{X}} \Big((\mathrm{Re}(\hat{e}(\boldsymbol{x})) * \hat{a}_{re}(\boldsymbol{x}))^2 + (\mathrm{Im}(\hat{e}(\boldsymbol{x})) * \hat{a}_{im}(\boldsymbol{x}))^2 \Big) \tag{4}$$

The Fourier transformation is invertible and differentiable, so we can backpropagate gradients to parameters directly from this loss. Intuitively, the frequency right-reason loss causes the masked frequency explanations of the model to be small while not affecting any specific point in time.

Depending on the problem, it is possible to use RioT either in the spatial or frequency domain. Moreover, it is also possible to combine feedback in both

domains, including two right-reason terms in the final loss. This results in two parameters λ_1 and λ_2 to balance the right-answer and both right-reason losses.

$$\mathcal{L}(\mathcal{X}, \mathcal{Y}, A, E) = \mathcal{L}_{\text{RA}}(\mathcal{X}, \mathcal{Y}) + \lambda_1 \mathcal{L}^{sp}_{\text{RR}}(A, E) + \lambda_2 \mathcal{L}^{fr}_{\text{RR}}(\hat{A}, \hat{E}) \tag{5}$$

It is important to note that giving feedback in the frequency domain allows a new form of model revision through XIL. Even if we effectively still apply masking in the frequency domain, the effect in the original input domain is entirely different. Masking out a single frequency affects all time points without preventing the model from looking at any of them. In general, having an invertible transformation from the input domain to a different representation allows to give feedback more flexible than before. The Fourier transformation is a prominent example but not the only possible choice for this. Using other transformations like wavelets [12], is also possible.

Computational Costs. Including RioT in the training of a model increases the computational cost. Computing the right reason loss term requires the computation of a mixed partial derivative: $\frac{\partial^2 f_\theta(x)}{\partial \theta \partial x}$. Even though this is a second-order derivative, it does not result in any substantial cost increases, as the second-order component of our loss can be formalized as a Hessian-vector product (cf. Appx. A.3), which is known to be fast to compute [20]. We also observed this in our experimental evaluation, as even the naive implementation of our loss in PyTorch scales to large models.

Source of Feedback. A key aspect of RioT is the feedback incorporated in the *Obtain* step, which can come from various sources, including automated methods, rule-based systems, foundation models, or human annotations. Automated approaches, such as rule-based heuristics or pre-trained foundation models, provide scalable and consistent feedback, reducing the reliance on manual labeling. However, human annotations remain valuable for ensuring accuracy, especially in complex cases where automated methods may introduce biases or fail to capture nuanced patterns. RioT is agnostic to the feedback source, allowing flexibility in its application.

4 Shortcuts in Time Series

Shortcuts, like those in images, naturally occur in time series data but are often less apparent. Developing effective mitigation methods requires datasets where shortcuts are explicitly annotated, yet no existing datasets provide such annotations, despite known biases in popular datasets [2]. To address this gap, we introduce several time series dataset decoy variants inspired by prior work on decoy data [27], allowing for controlled evaluation of shortcut mitigation strategies. To further evaluate shortcut mitigation under real-world conditions, we present P2S, a real-world dataset where shortcuts arise from sensor recording processes.

4.1 Decoy Shortcuts

Classification. For both spatial and frequency cases, we introduce the shortcut as a class-specific pattern embedded in each training sample. The **spatial** pattern

replaces m time steps with a sine wave defined as $s := \sin(t \cdot (2+j)\pi) \cdot A$ where $t \in \{0, 1, \ldots, m\}$ are the respective time steps, j represents the class index and A is the amplitude. In contrast, the **frequency** pattern is a similar sine wave, but it is *additively* applied to the full sequence ($m = T$).

Forecasting. For forecasting datasets, spatial decoys are more challenging due to window-based sampling and the complexity of the target. To address this, we design the shortcut as a "back-copy" of the forecast, where the decoy is equivalent to the actual solution. Due to the windowed sampling, the shortcut appears in every second sample. Given a sample of *lookback window* $\boldsymbol{x} \in \mathbb{R}^T$ and the *forecast horizon* $\boldsymbol{y} \in \mathbb{R}^W$. In the shortcut samples, we overwrite the first W entries of the lookback window with the future horizon values, yielding:

$$\boldsymbol{x}^{\mathrm{s}} = \Big(\underbrace{y_1, \ldots, y_W}_{\text{future horizon}}, \quad \underbrace{x_{W+1}, \ldots, x_T}_{\text{remaining lookback}} \Big).$$

while y remains unchanged. While this setting may seem constructed, similar patterns can emerge in real-world scenarios. For instance, in data transmission, glitches such as packet losses or duplications can subtly introduce irregularities into time series data, inadvertently creating forecasting shortcuts.

We model the forecasting frequency decoy as a recurring Dirac impulse with a specific frequency, added every k time steps: $i \in \{n \cdot k | n \in \mathbb{N} \wedge n \cdot k \leq T + W\}$ with a strength of A: *interference* $:= A \cdot \Delta_i$. The impulse is present within the lookback and forecasting window during training, representing an effective decoy distracting the model from the actual forecast.

4.2 Real-World Shortcuts: Production Press Sensor Data (P2S)

RioT aims to mitigate shortcuts in time series data. While the decoys above provide a controlled evaluation setting, they do not capture the complexity of real-world shortcuts. To rectify this, we introduce Production Press Sensor Data (P2S)[6], a dataset of sensor recordings with naturally occurring shortcuts.

The sensor data stems from a high-speed press production line for metal parts, one of the sheet metal working industry's most economically significant processes. Based on the sensor data, the task is to predict whether a run is defective. The recordings include different production speeds, which, although not affecting part quality, influence process friction, and applied forces. Fig. 3 shows samples recorded at different speeds from normal and defect runs, highlighting variations even within the same class. A domain expert identified regions in the time series that vary with production speed, potentially distracting models from relevant features, especially when no defect and normal runs of the same speed are in the training data. In these cases, the run's speed is a shortcut and makes it difficult to generalize to other speeds than those present in training. P2S also includes a specifically curated setup that matches run speeds during training to avoid the shortcut. Further details on the dataset are available in Sec. B.

[6] https://huggingface.co/datasets/AIML-TUDA/P2S

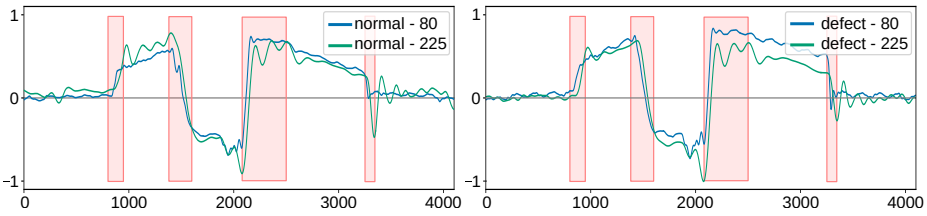

Fig. 3. Samples of P2S from normal (left) and defect (right) class at 80 and 225 strokes per minute. Areas of the time series that are especially sensitive to the stroke rate are considered a shortcut and marked red.

5 Experimental Evaluations

In this section, we investigate the effectiveness of RioT[7] to mitigate shortcuts in time series classification and forecasting, including revision in the spatial domain (RioT_{sp}) and the frequency domain ($\text{RioT}_{\text{freq}}$), as well as both jointly.

5.1 Experimental Setup

Data. For classification, we use datasets from the UCR/UEA repository [7]. We select available datasets of a minimal size (cf. Appx. A.2), which results in Fault Detection A, Ford A, Ford B, and Sleep. For time series forecasting, we evaluate on three popular datasets from the Darts repository [14]: ETTM1, Energy, and Weather. We split the data into training and test sets using a 70/30 ratio and 20% of the training set are used for validation. We apply the previously described decoys to the training sets and simulate feedback based on the shortcuts to generate annotation masks. In the real-world experiment, we utilize our newly introduced dataset P2S. The mask is applied to all samples except in our feedback scaling experiment. For the real-world test case, we consider our newly introduced dataset P2S. We standardize all datasets as suggested by [41], i.e., rescaling the distribution to zero mean and a standard deviation of one.

Models. For time series classification, we use the FCN model of [19], with a slightly modified architecture for Sleep to achieve better performance (cf. Appx. A.1). Additionally, we use the OFA model [43]. For forecasting, we use TiDE [6], PatchTST [24] and NBEATS [25] to highlight the applicability of our method to a variety of model classes.

Metrics. In our evaluations, we compare model performance on datasets with and without shortcuts, as well as with and without RioT. For classification, we report balanced (multiclass) accuracy (ACC), and mean squared error (MSE) for forecasting. The respective mean absolute error (MAE) results can be found in Appx. A.5. We report average and standard deviation over 5 runs.

[7] https://github.com/ml-research/RioT

Table 1. Applying RioT mitigates shortcuts in time series classification. Performance before and after applying RioT for spatial (Base_{sp}) and frequency ($\text{Base}_{\text{freq}}$) shortcuts. High training and low test accuracies indicate overfitting to the shortcut, which RioT successfully mitigates. *No Shortcut* represents the ideal scenario where the model is not affected by any shortcut.

Model	Config	Fault Detection A		FordA		FordB		Sleep	
	(ACC ↑)	Train	Test	Train	Test	Train	Test	Train	Test
FCN	No Shortcut	0.99 ±0.00	0.99 ±0.00	0.92 ±0.01	0.91 ±0.00	0.93 ±0.00	0.76 ±0.01	0.68 ±0.00	0.62 ±0.00
	Base_{sp}	**1.00** ±0.00	0.74 ±0.06	**1.00** ±0.00	0.71 ±0.08	**1.00** ±0.00	0.63 ±0.03	**1.00** ±0.00	0.10 ±0.03
	+ RioT_{sp}	0.98 ±0.01	**0.93** ±0.03	0.99 ±0.01	**0.84** ±0.02	0.99 ±0.00	**0.68** ±0.02	0.60 ±0.06	**0.54** ±0.05
	$\text{Base}_{\text{freq}}$	**0.98** ±0.01	0.87 ±0.03	**0.98** ±0.00	0.73 ±0.01	**0.99** ±0.01	0.60 ±0.01	**0.98** ±0.00	0.27 ±0.02
	+ $\text{RioT}_{\text{freq}}$	0.94 ±0.00	**0.90** ±0.03	0.83 ±0.02	**0.83** ±0.02	0.94 ±0.00	**0.65** ±0.01	0.67 ±0.05	**0.45** ±0.07
OFA	No Shortcut	1.00 ±0.00	0.98 ±0.02	0.92 ±0.01	0.87 ±0.04	0.95 ±0.01	0.70 ±0.04	0.69 ±0.00	0.64 ±0.01
	Base_{sp}	**1.00** ±0.00	0.53 ±0.02	**1.00** ±0.00	0.50 ±0.00	**1.00** ±0.00	0.52 ±0.01	**1.00** ±0.00	0.21 ±0.05
	+ RioT_{sp}	0.96 ±0.08	**0.98** ±0.01	0.92 ±0.03	**0.85** ±0.02	0.94 ±0.01	**0.65** ±0.04	0.52 ±0.22	**0.58** ±0.05
	$\text{Base}_{\text{freq}}$	**1.00** ±0.00	0.72 ±0.02	**1.00** ±0.00	0.65 ±0.01	1.00 ±0.00	0.56 ±0.02	**0.99** ±0.00	0.24 ±0.03
	+ $\text{RioT}_{\text{freq}}$	0.96 ±0.02	**0.98** ±0.02	0.78 ±0.04	**0.85** ±0.04	**1.00** ±0.00	**0.64** ±0.03	0.50 ±0.16	**0.49** ±0.04

5.2 Evaluations

Removing Shortcuts for Time Series Classification. We evaluate the effectiveness of RioT (spatial: RioT_{sp}, frequency: $\text{RioT}_{\text{freq}}$) in addressing shortcuts in classification tasks by comparing balanced accuracy with and without RioT. As shown in Tab. 1, without RioT, both FCN and OFA overfit to shortcuts, achieving ≈100% training accuracy, while having poor test performance. Applying RioT significantly improves test performance for both models across all datasets. In some cases, RioT even reaches the performance of the ideal reference (no shortcut) scenario as if there would be no shortcut in the data. Even on FordB, where the drop in training-to-test performance highlights the distribution shift of that dataset [2], RioT_{sp} is still beneficial. Similarly, $\text{RioT}_{\text{freq}}$ enhances performance on data with frequency shortcuts, though the improvement is less pronounced for FCN on Ford B, suggesting essential frequency information is sometimes obscured by $\text{RioT}_{\text{freq}}$. In summary, RioT successfully mitigates shortcuts in both domains, enhancing test generalization for FCN and OFA models.

Removing Shortcuts for Time Series Forecasting. Shortcuts are not exclusive to time series classification and can significantly impact other tasks, such as forecasting. In Tab. 2, we outline that spatial shortcuts cause models to overfit, but applying RioT_{sp} reduces MSE across datasets, especially for Energy, where MSE drops by up to 56%. In the frequency-shortcut setting, the training data includes a recurring Dirac impulse as a decoy (cf. Appx. A.4 for details). $\text{RioT}_{\text{freq}}$ alleviates this distraction and improves the test performance significantly. For example, TiDE's test MSE on ETTM1 decreases by 14% compared to the decoy setting.

In general, RioT effectively addresses spatial and frequency shortcuts in forecasting tasks. Interestingly, for TiDE on the Energy dataset, the performance with $\text{RioT}_{\text{freq}}$ even surpasses the no shortcut model. Here, the added frequency acts as a form of data augmentation, enhancing model robustness. A similar

Table 2. RioT can successfully overcome shortcuts in time series forecasting. MSE values (MAE values cf. Tab. 7) on the training set with and test set without shortcuts. *No Shortcut* is the ideal scenario where the model is not affected by shortcuts.

Model	Config (MSE ↓)	ETTM1		Energy		Weather	
		Train	Test	Train	Test	Train	Test
NBEATS	No Shortcut	0.30 ±0.02	0.47 ±0.02	0.34 ±0.03	0.26 ±0.02	0.08 ±0.01	0.03 ±0.01
	Base_{sp}	**0.24** ±0.01	0.55 ±0.01	**0.33** ±0.03	0.94 ±0.02	**0.09** ±0.01	0.16 ±0.04
	+ RioT_{sp}	0.30 ±0.01	**0.50** ±0.01	0.45 ±0.03	**0.58** ±0.01	0.11 ±0.01	**0.09** ±0.02
	$\text{Base}_{\text{freq}}$	**0.30** ±0.02	0.46 ±0.01	**0.33** ±0.04	0.36 ±0.04	**0.11** ±0.02	0.32 ±0.09
	+ $\text{RioT}_{\text{freq}}$	0.31 ±0.02	**0.45** ±0.01	**0.33** ±0.04	**0.34** ±0.04	0.81 ±0.48	**0.17** ±0.01
PatchTST	No Shortcut	0.46 ±0.03	0.47 ±0.01	0.26 ±0.01	0.23 ±0.00	0.26 ±0.03	0.08 ±0.01
	Base_{sp}	**0.40** ±0.02	0.55 ±0.01	**0.29** ±0.01	0.96 ±0.03	**0.20** ±0.03	0.19 ±0.01
	+ RioT_{sp}	**0.40** ±0.03	**0.53** ±0.01	0.44 ±0.00	**0.45** ±0.01	0.55 ±0.20	**0.14** ±0.01
	$\text{Base}_{\text{freq}}$	**0.45** ±0.03	0.91 ±0.16	**0.04** ±0.00	0.53 ±0.05	**0.63** ±0.09	0.24 ±0.04
	+ $\text{RioT}_{\text{freq}}$	0.91 ±0.07	**0.66** ±0.04	0.71 ±0.10	**0.38** ±0.06	0.96 ±0.02	**0.17** ±0.00
TiDE	No Shortcut	0.27 ±0.01	0.47 ±0.01	0.27 ±0.01	0.26 ±0.02	0.25 ±0.02	0.03 ±0.00
	Base_{sp}	**0.22** ±0.01	0.54 ±0.03	**0.28** ±0.01	1.19 ±0.03	**0.22** ±0.03	0.15 ±0.01
	+ RioT_{sp}	0.23 ±0.01	**0.48** ±0.01	0.53 ±0.02	**0.52** ±0.02	0.25 ±0.03	**0.11** ±0.01
	$\text{Base}_{\text{freq}}$	**0.06** ±0.01	0.69 ±0.08	**0.07** ±0.01	0.34 ±0.08	**0.79** ±0.09	0.31 ±0.09
	+ $\text{RioT}_{\text{freq}}$	0.07 ±0.01	**0.49** ±0.07	**0.07** ±0.01	**0.21** ±0.02	1.12 ±0.36	**0.22** ±0.01

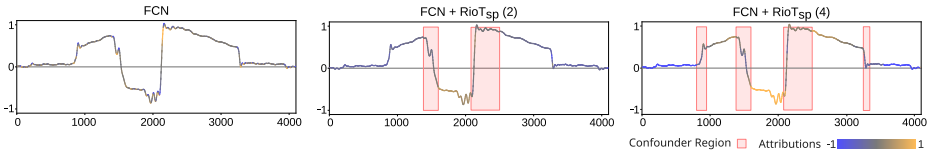

Fig. 4. Applying RioT lets the model ignore shortcut areas. While FCN primarily focuses on shortcuts, applying RioT with partial feedback (middle) or full feedback (bottom) causes the model to ignore the shortcut and focus on the remaining input.

behavior can be observed for NBEATS and ETTM1, where the decoy setting actually improves the model slightly, and RioT even improves upon that.

Removing Shortcuts in the Real-World. So far, our experiments have demonstrated RioT's ability to counteract shortcuts within controlled environments. However, real-world shortcuts, as in our new dataset P2S, often have more complex structures. The model explanations in Fig. 4 (top) reveal a focus on distinct regions of the sensor curve, specifically the two middle regions. With domain knowledge, it is clear that these regions should not affect the model's output. By applying RioT, we can redirect the model's attention away from these regions. New model explanations highlight that the model still focuses on (other) incorrect regions, which can be mitigated by extending the annotated area. In Tab. 3, the model performance (exemplarly with FCN) in these settings is presented. Without RioT, the model overfits to the shortcut. the test performance improves already with partial feedback (2) and even more with full feedback (4). These results highlight the effectiveness of RioT in real-world scenarios where not all shortcuts are initially known.

Table 3. Applying RioT overcomes shortcuts in P2S. Performance on the train set with and test set without shortcuts. FCN learns the train shortcut, resulting in lower test performance. Applying RioT with partial feedback (2) already yields good improvements, while adding feedback on the full shortcut area (4) is even better. *No Shortcut* is the ideal scenario, specifically curated so that there is no shortcut.

P2S (ACC ↑)	Train	Test
$\text{FCN}_{\text{No Shortcut}}$	0.97 ±0.01	0.95 ±0.01
FCN_{sp}	**0.99** ±0.01	0.66 ±0.14
FCN_{sp} + RioT_{sp} (2)	0.96 ±0.01	0.78 ±0.05
FCN_{sp} + RioT_{sp} (4)	0.95 ±0.01	**0.82** ±0.06

Table 4. RioT can combine spatial and frequency feedback. If the data contains time and frequency shortcuts, RioT can combine feedback on both domains to mitigate them, superior to feedback on only one domain. *No Shortcut* represents the ideal scenario when the model is not affected by any shortcuts.

Sleep (Classification ACC ↑)	Train	Test
$\text{FCN}_{\text{No Shortcut}}$	0.68 ±0.00	0.62 ±0.00
$\text{FCN}_{\text{freq,sp}}$	**1.00** ±0.00	0.10 ±0.04
$\text{FCN}_{\text{freq,sp}}$ + RioT_{sp}	0.94 ±0.00	0.24 ±0.02
$\text{FCN}_{\text{freq,sp}}$ + $\text{RioT}_{\text{freq}}$	**1.00** ±0.00	0.04 ±0.00
$\text{FCN}_{\text{freq,sp}}$ + $\text{RioT}_{\text{freq,sp}}$	0.47 ±0.00	**0.48** ±0.03
Energy (Forecasting MSE ↓)	**Train**	**Test**
$\text{TiDE}_{\text{No Shortcut}}$	0.28 ±0.01	0.26 ±0.02
$\text{TiDE}_{\text{freq,sp}}$	**0.16** ±0.01	0.74 ±0.02
$\text{TiDE}_{\text{freq,sp}}$ + RioT_{sp}	0.20 ±0.01	0.61 ±0.02
$\text{TiDE}_{\text{freq,sp}}$ + $\text{RioT}_{\text{freq}}$	0.22 ±0.01	0.55 ±0.02
$\text{TiDE}_{\text{freq,sp}}$ + $\text{RioT}_{\text{freq,sp}}$	0.25 ±0.01	**0.47** ±0.01

Removing Multiple Shortcuts at Once. In the previous experiments, we illustrated that RioT is suitable for addressing individual shortcuts, whether spatial or frequency-based. However, real-world time series data often presents a blend of multiple shortcuts that simultaneously influence model performance. Thus, we investigate the impact of applying RioT to both spatial and frequency shortcuts simultaneously (cf. Tab. 4), exemplarily using FCN and TiDE. When Sleep contains shortcuts in both domains, FCN without RioT overfits and fails to generalize. Addressing only one shortcut does not mitigate the effects, as the model adapts to the other. However, combining the respective feedback from both domains ($\text{RioT}_{\text{freq,sp}}$) significantly improves test performance, matching the frequency-shortcut scenario (cf. Tab. 1). Tab. 4 (bottom) shows the impact of multiple shortcuts on the Energy dataset, where the lower training MSE indicates overfitting. While applying either spatial or frequency feedback individually shows some effect, utilizing both types of feedback simultaneously ($\text{RioT}_{\text{freq,sp}}$) results in the largest improvements, as both decoys are addressed. The performance gap between $\text{RioT}_{\text{freq,sp}}$ and the no shortcut setting is more pronounced than in single shortcut cases (cf. Tab. 2). This highlights again the known challenge of removing multiple shortcuts at once, which is generally more complex than individual shortcuts [36].

Handling Feedback. As the annotations are a crucial component of RioT, we conduct two different ablation studies to evaluate its impact using the classification data set Fault Detection A and the forecasting data set Energy. The first experiment examines the required amount of feedback, while the second assesses robustness to noisy feedback. In particular if the feedback stems from domain experts, making excessive feedback requests is impractical. Thus our first experiment evaluates the performance of RioT when feedback is provided on only a portion of the dataset (Fig. 5). The findings reveal that full annotations are

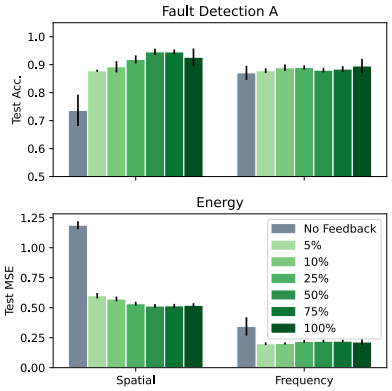

Fig. 5. RioT uses feedback efficiently. Even with feedback on only a small percentage of the data, RioT can overcome shortcuts.

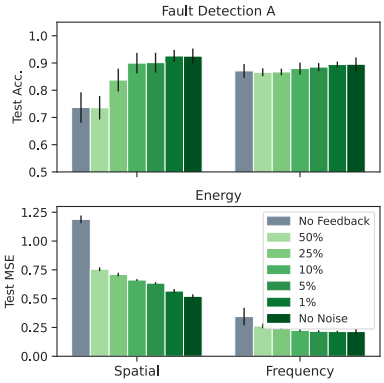

Fig. 6. RioT is robust against invalid feedback. Even with some percentage of random feedback, RioT overcomes the shortcuts.

unnecessary. Even with minimal feedback, such as annotating just 5% of the samples, RioT significantly outperforms scenarios with no feedback. While previous experiments assumed entirely accurate feedback, real-world applications often involve some degree of error. Therefore, we test the resilience of RioT to increasing levels of incorrect feedback (Fig. 6). Instead of accurately marking shortcut areas, random time steps or frequency components are incorrectly labeled as shortcuts. The results show that RioT maintains strong performance even with up to 10% invalid feedback, presenting only slight performance declines. In certain cases, like forecasting with spatial shortcuts, RioT can still achieve notable improvements despite high levels of feedback noise. To further evaluate whether annotations in different settings can also be incorporated via RioT, we conduct an additional ablation where the feedback is based on shaplets instead of the input domain directly (cf. Tab. 11 in the appendix). The results show that RioT can be effective in this setup as well, and is not limited to the specific explanation method and annotation modality shown in the other experiments (more details in Appx. A.5). In summary, RioT effectively generalizes from small subsets of feedback and remains robust against a moderate amount of annotation noise. Additionally, RioT can incorporate feedback in other settings with other types of explanations as well. These qualities demonstrate that RioT is well-equipped to manage the practical challenges associated with incorporating feedback.

Qualitative Insights into Model Encodings. Lastly, we examine the inner workings of a model by analyzing its latent representations under various configurations. In Fig. 7, t-SNE plots show OFA's feature encodings on Fault Detection A in three settings: trained on shortcut data (left) with poor class separation; after RioT regularization (center), where structure and separation improve; and trained on clean data (right), yielding clear clusters. This reflects the scores of the models (Tab. 1): the shortcut model reaches ≈50% accuracy,

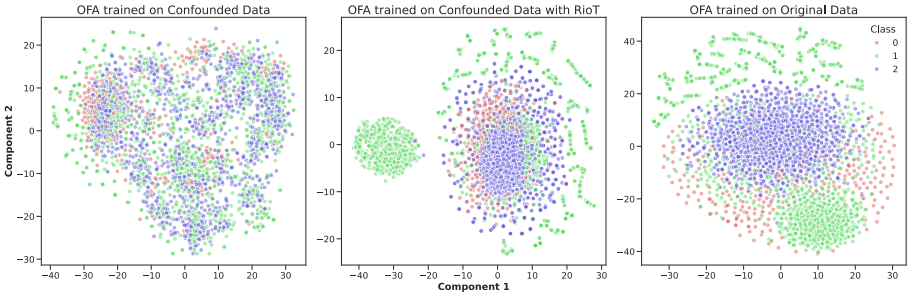

Fig. 7. t-SNE plots of OFA encodings for Fault Detection A. The left plot shows that a model trained with shortcuts shows minimal class separation. The middle plot shows the same setup but after RioT regularization, while the far right plot shows an model without shortcuts with clear class separation. Both RioT-regularized and model without shortcuts exhibit similar structures, highlighting the effectiveness of RioT.

whereas RioT boosts it to nearly 100%, matching the reference scenario with clean data. This further demonstrates RioT's ability to mitigate shortcuts and restore robust performance.

Limitations. A key aspect of RioT is the incorporation of feedback. While this is a major advantage of RioT, obtaining feedback can also present some challenges. Although we demonstrate that only a small fraction of annotated samples is needed, annotations remain essential. Moreover, like many interactive learning approaches, RioT assumes accurate feedback, making it important to consider potential issues from inaccuracies in practical applications. To reduce the need of manual feedback, one could explore automated feedback strategies instead or alongside manual feedback [34] (e.g., using an LLM to provide feedback or automated clustering of explanations to identify outliers). Such approaches may alleviate annotation costs but inevitably trade off some precision and can introduce new failure modes if the surrogate feedback is misaligned with task requirements. Another drawback of RioT is the increased training cost. Optimizing model explanations with gradient-based attributions requires computing a mixed-partial derivative. However, this can be efficiently handled using a Hessian-vector product, keeping the additional overhead manageable.

6 Conclusion

In this work, we present Right on Time (RioT) a method to mitigate shortcuts in time series data with the help of feedback. By revising the model, RioT significantly diminishes the influence of these factors, steering the model to align with the correct reasons. Using popular time series models on several controlled decoy datasets and the newly introduced, real-world dataset P2S with naturally occurring shortcuts, showcases that SOTA models are indeed subject to shortcuts. Our results demonstrate that applying RioT to these models can mitigate

shortcuts in the data. Furthermore, we have unveiled that addressing solely the time domain is insufficient for fully steering the model toward the correct reasons. To overcome this, we extended our method to incorporate feedback in the frequency domain, offering an additional mechanism for reducing reliance on shortcuts. Logical next steps are the extension of RioT to multivariate time series and the integration of various explainer types. Furthermore, exploring the usage of adaptive feedback mechanisms could prove to be beneficial, in particular in the context of multiple simultaneous shortcuts. Beyond time series, the application of RioT, especially $\text{RioT}_{\text{freq}}$, can also allow for a more nuanced approach to shortcut mitigation in other modalities.

Acknowledgments. This work was partly funded by the Federal Ministry of Education and Research (BMBF) project "XEI" (FKZ 01IS24079B), received funding by the EU project EXPLAIN, funded by the Federal Ministry of Education and Research (grant 01—S22030D). Additionally, it was funded by the project "The Adaptive Mind" from the Hessian Ministry of Science and the Arts (HMWK), the "ML2MT" project from the Volkswagen Stiftung, and the Priority Program (SPP) 2422 in the subproject "Optimization of active surface design of high-speed progressive tools using machine and deep learning algorithms" funded by the German Research Foundation (DFG). The latter also contributed the data for the P2S dataset. Furthermore, this work benefited from the HMWK project "The Third Wave of Artificial Intelligence – 3AI".

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. PLOS ONE **10**(7), e0130140 (2015)
2. Bagnall, A., Davis, L., Hills, J., Lines, J.: Transformation Based Ensembles for Time Series Classification. In: Proceedings of the SIAM International Conference on Data Mining (SDM) (2012)
3. Benidis, K., Rangapuram, S.S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Aubet, F.X., Callot, L., Januschowski, T.: Deep Learning for Time Series Forecasting: Tutorial and Literature Survey. ACM Computing Surveys **55**(6), 1–36 (2023)
4. Bica, I., Alaa, A.M., Van Der Schaar, M.: Time series deconfounder: estimating treatment effects over time in the presence of hidden confounders. In: Proceedings of the International Conference on Machine Learning (ICML) (2020)
5. Cao, D., Enouen, J., Wang, Y., Song, X., Meng, C., Niu, H., Liu, Y.: Estimating treatment effects from irregular time series observations with hidden confounders. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) (2023)
6. Das, A., Kong, W., Leach, A., Mathur, S., Sen, R., Yu, R.: Long-term Forecasting with TiDE: Time-series Dense Encoder. Transactions on Machine Learning Research (TMLR) (2023)
7. Dau, H.A., Bagnall, A.J., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.J.: The UCR time series archive. ArXiv:1810.07758 (2018)

8. Flanders, W.D., Klein, M., Darrow, L., Strickland, M., Sarnat, S., Sarnat, J., Waller, L., Winquist, A., Tolbert, P.: A Method for Detection of Residual Confounding in Time-Series and Other Observational Studies. Epidemiology **22**(1), 59–67 (2011)
9. Friedrich, F., Stammer, W., Schramowski, P., Kersting, K.: A typology for exploring the mitigation of shortcut behaviour. Nature Machine Intelligence **5**(3), 319–330 (2023)
10. Friedrich, F., Steinmann, D., Kersting, K.: One explanation does not fit XIL. ArXiv:2304.07136 (2023)
11. Geirhos, R., Jacobsen, J.H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., Wichmann, F.A.: Shortcut learning in deep neural networks. Nature Machine Intelligence **2**(11), 665–673 (2020)
12. Graps, A.: An introduction to wavelets. IEEE Computational Science and Engineering **2**(2), 50–61 (1995)
13. Hatt, T., Feuerriegel, S.: Sequential deconfounding for causal inference with unobserved confounders. In: Proceedings of the Conference on Causal Learning and Reasoning (CLeaR) (2024)
14. Herzen, J., Lässig, F., Piazzetta, S.G., Neuer, T., Tafti, L., Raille, G., Pottelbergh, T.V., Pasieka, M., Skrodzki, A., Huguenin, N., Dumonal, M., Kościsz, J., Bader, D., Gusset, F., Benheddi, M., Williamson, C., Kosinski, M., Petrik, M., Grosch, G.: Darts: User-Friendly Modern Machine Learning for Time Series. Journal of Machine Learning Research (JMLR) **23**(124), 1–6 (2022)
15. Koprinska, I., Wu, D., Wang, Z.: Convolutional Neural Networks for Energy Time Series Forecasting. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN) (2018)
16. Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., Müller, K.R.: Unmasking Clever Hans predictors and assessing what machines really learn. Nature Communications **10**(1), 1096 (2019)
17. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the ACM SIGMOD workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD) (2003)
18. Ma, C., Dai, G., Zhou, J.: Short-Term Traffic Flow Prediction for Urban Road Sections Based on Time Series Analysis and LSTM_bilstm Method. IEEE Transactions on Intelligent Transportation (T-ITS) **23**(6), 5615–5624 (2022)
19. Ma, Q., Liu, Z., Zheng, Z., Huang, Z., Zhu, S., Yu, Z., Kwok, J.T.: A survey on time-series pre-trained models. ArXiv:2305.10716 (2023)
20. Martens, J.: Deep learning via hessian-free optimization. In: Proceedings of the International Conference on Machine Learning (ICML) (2010)
21. Mehdiyev, N., Lahann, J., Emrich, A., Enke, D., Fettke, P., Loos, P.: Time Series Classification using Deep Learning for Process Planning: A Case from the Process Industry. Procedia Computer Science **114**, 242–249 (2017)
22. Mercier, D., Bhatt, J., Dengel, A., Ahmed, S.: Time to Focus: A Comprehensive Benchmark Using Time Series Attribution Methods. ArXiv:2202.03759 (2022)
23. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence (AIJ) **267**, 1–38 (2019)
24. Nie, Y., H. Nguyen, N., Sinthong, P., Kalagnanam, J.: A time series is worth 64 words: Long-term forecasting with transformers. In: Proceedings of the International Conference on Learning Representations (ICLR) (2023)

25. Oreshkin, B.N., Carpov, D., Chapados, N., Bengio, Y.: N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In: Proceedings of the International Conference on Learning Representations (ICLR) (2020)
26. Rojat, T., Puget, R., Filliat, D., Del Ser, J., Gelin, R., Díaz-Rodríguez, N.: Explainable Artificial Intelligence (XAI) on TimeSeries Data: A Survey. ArXiv:2104.00950 (2021)
27. Ross, A.S., Hughes, M.C., Doshi-Velez, F.: Right for the right reasons: Training differentiable models by constraining their explanations. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (2017)
28. Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.: The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery (DMKD) **35**(2), 401–449 (2021)
29. Schlegel, U., Arnout, H., El-Assady, M., Oelke, D., Keim, D.A.: Towards a Rigorous Evaluation of XAI Methods on Time Series. ArXiv:1909.07082 (2019)
30. Schramowski, P., Stammer, W., Teso, S., Brugger, A., Herbert, F., Shao, X., Luigs, H.G., Mahlein, A.K., Kersting, K.: Making deep neural networks right for the right scientific reasons by interacting with their explanations. Nature Machine Intelligence **2**(8), 476–486 (2020)
31. Selvaraju, R.R., Lee, S., Shen, Y., Jin, H., Ghosh, S., Heck, L., Batra, D., Parikh, D.: Taking a HINT: Leveraging Explanations to Make Vision and Language Models More Grounded. In: Proceedings of the International Conference on Computer Vision (ICCV) (2019)
32. Shao, X., Skryagin, A., Stammer, W., Schramowski, P., Kersting, K.: Right for Better Reasons: Training Differentiable Models by Constraining their Influence Functions. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) (2021)
33. Shrikumar, A., Greenside, P., Shcherbina, A., Kundaje, A.: Not Just a Black Box: Learning Important Features Through Propagating Activation Differences. ArXiv:1605.01713 (2017)
34. Stammer, W., Friedrich, F., Steinmann, D., Brack, M., Shindo, H., Kersting, K.: Learning by self-explaining. Transactions on Machine Learning Research (TMLR) (2024)
35. Stammer, W., Schramowski, P., Kersting, K.: Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
36. Steinmann, D., Divo, F., Kraus, M., Wüst, A., Struppek, L., Friedrich, F., Kersting, K.: Navigating Shortcuts, Spurious Correlations, and Confounders: From Origins via Detection to Mitigation. ArXiv:2412.05152 (2024)
37. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: Proceedings of the International Conference on Machine Learning (ICML) (2017)
38. Teso, S., Kersting, K.: Explanatory Interactive Machine Learning. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES) (2019)
39. Veerappa, M., Anneken, M., Burkart, N., Huber, M.F.: Validation of XAI explanations for multivariate time series classification in the maritime domain. Journal of Computational Science **58**, 101539 (2022)
40. Vielhaben, J., Lapuschkin, S., Montavon, G., Samek, W.: Explainable AI for Time Series via Virtual Inspection Layers. ArXiv:2303.06365 (2023)
41. Wu, H., Xu, J., Wang, J., Long, M.: Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In: Proceedings of the Conference on Neural Information Processing Systems (NeurIPS) (2021)

42. Ye, L., Keogh, E.: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Mining and Knowledge Discovery (DMKD) **22**, 149–182 (2011)
43. Zhou, T., Niu, P., Wang, X., Sun, L., Jin, R.: One fits all: Power general time series analysis by pretrained LM. In: Proceedings of the Conference on Neural Information Processing Systems (NeurIPS) (2023)