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

Maurice Kraus[*]
AI and ML Group
Technical University of Darmstadt
maurice.kraus@cs.tu-darmstadt.de

David Steinmann[*]
AI and ML Group
Technical University of Darmstadt
david.steinmann@tu-darmstadt.de

Antonia Wüst
AI and ML Group
Technical University of Darmstadt

Andre Kokozinski
Technical University of Darmstadt

Kristian Kersting
AI and ML Group
Technical University of Darmstadt
Centre for Cognitive Science
Hessian Center for AI (hessian.AI)
German Center for AI (DFKI)

Abstract

The reliability of deep time series models is often compromised by their tendency to rely on confounding factors, which may lead to incorrect outputs. Our newly recorded, naturally confounded dataset named P2S from a real mechanical production line emphasizes this. To avoid "Clever-Hans" moments in time series, i.e., to mitigate confounders, we introduce the method Right on Time (RioT). RioT enables, for the first time, interactions with model explanations across both the *time* and *frequency* domain. Feedback on explanations in both domains is used to steer models away from the annotated confounding factors. Dual-domain interactions are crucial to effectively address confounders in time series datasets. We empirically demonstrate that RioT can effectively guide models away from the wrong reasons in P2S as well as popular time series classification and forecasting datasets.

1 Introduction

Time series data is ubiquitous in today's world. Everything measured over time generates time series, e.g., energy load [14], sensor measurements in industrial machinery [19] or recordings of traffic data [16]. Various neural models are often applied to handle complex time series data [24, 2]. As in other domains, these can be subject to confounding factors ranging from simple noise or artifacts to complex shortcut confounders [15]. Intuitively, a confounder, also called "Clever-Hans" moment, can be a pattern in the data which is not relevant for the task, but correlates with it during model training. A model can learn this confounder (cf. Fig. 1) and use it instead of the relevant features to, e.g., make a classification. A confounded model does not generalize well to data without the confounder, which is problematic when employing models in practice [9]. For time series, confounders and their mitigation have yet to receive attention, where existing works make specific assumptions about settings and data [3].

[*]Equal contribution

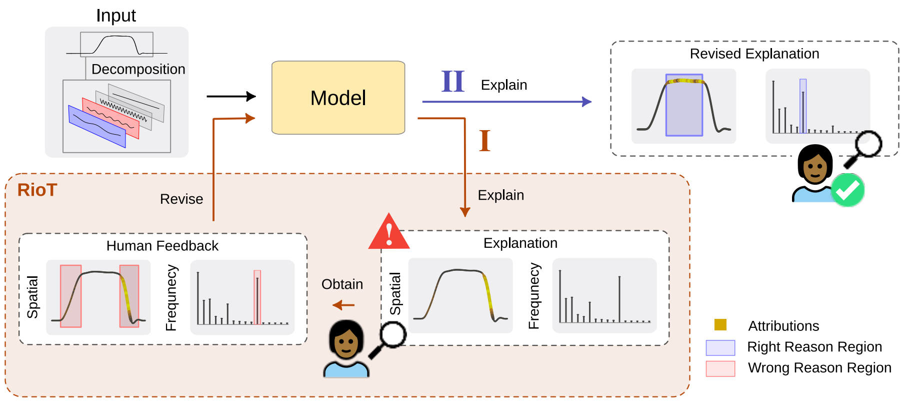

Figure 1: **I:** Explanations can reveal whether models rely on confounding factors in the input instead of relevant features. With RioT, a human can provide feedback on the spatial and frequency domain explanations for wrong reasons. This feedback is used to revise the model to not consider those regions. **II:** After revising via RioT, the model focuses on the right reasons instead.

Model explanations play a crucial role in uncovering confounding factors, but they are not sufficient to mitigate them. While an explanation can reveal that the model relies on incorrect factors, it does not alter the model's outcome. To change this, we introduce Right on Time (RioT), a new method following the core ideas of explanatory interactive learning (XIL) [31], i.e., utilizing feedback on explanations to mitigate confounders. RioT uses traditional explanation methods like Integrated Gradients (IG) [30] to detect whether models focus on the right or the wrong time steps and utilizes feedback on the latter to revise the model (Fig. 1, left). As confounding factors in time series data are not limited to the time domain, RioT for the first time enables interaction with the frequency domain to handle these confounders as well (Fig. 1, right).

With this work, we further introduce a new real-world, confounded dataset called PRODUCTION PRESS SENSOR DATA (P2S). It consists of sensor measurements from an industrial high-speed press, part of many important manufacturing processes in the sheet metal working industry. The sensor data used to detect faulty production is naturally confounded and thus causes incorrect predictions after training. P2S is the first time series dataset that contains explicitly annotated confounders, enabling evaluation and comparison of confounder 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 manually confounded datasets that SOTA neural networks on time series classification and forecasting can be affected by confounders. (2) We introduce RioT to mitigate confounders for time series data. The method can incorporate feedback not only on the time domain but also on the frequency domain. (3) By incorporating explanations and feedback in the frequency domain, we enable a new perspective on XIL, overcoming the important limitation that confounders must be spatially separable.[2]

2 Related Work

Explanatory Interactive Learning (XIL). While not prevalent for time series data, there has been some work on confounders and how to overcome them for other domains, primarily the image domain. Most notable, there is explanatory interactive learning, which describes the general process of revising a model's decision process based on human feedback [31, 25]. Within XIL, the model's explanations are used to incorporate the feedback back to the model, thus revising its mistakes [8]. Several methods apply the idea of XIL to image data. For example, Right for the Right Reasons (RRR) [23] and Right for Better Reasons (RBR) [27] use human feedback as a penalty mask on model explanations. Instead of penalizing wrong reasons, HINT [26] rewards the model for focusing on the correct part of the input. Although various XIL methods are often employed to address confounders in image data, their application to time series data remains unexplored. To bridge this gap, we introduce RioT, a method that incorporates the core principles of XIL to the unique characteristics of time series data.

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

Unconfounding Time Series. Next to approaches from interactive learning, there is also some other work on unconfounding time series models. This line of work is generally based on causal analysis of the time series model and data [7]. Methods like Time Series Deconfounder [3], SqeDec [11] or LipCDE [4], perform estimations on the data while mitigating the effect of confounders in covariates of the target variable. They generally mitigate the effect of the confounders through casual analysis and specific assumptions about the data generation. On the other hand, in this work, we tackle confounders within the target variate, and have no further assumption besides that the confounder is visible in the explanations of the model, where these previous methods cannot easily be applied.

3 Right on Time (RioT)

The core intuition of Right on Time (RioT) is to utilize human feedback to steer a model away from wrong reasons. It follows the general idea of XIL. We introduce RioT along the four steps of the XIL typology [8] (cf. Fig. 2): In *Select*, instances for feedback and following model revision are selected. Following previous XIL methods, we select all samples by default while not necessarily requiring feedback for all of them. Afterwards, *Explain* covers how model explanations are generated, before in *Obtain*, a human provides feedback on the selected instances. Lastly, in *Revise*, the feedback is integrated into the model to overcome the confounders.

Given is 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. In both cases, the ground-truth output of the full dataset is then $\mathcal{Y}$. 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 $\boldsymbol{y}$ for both tasks.

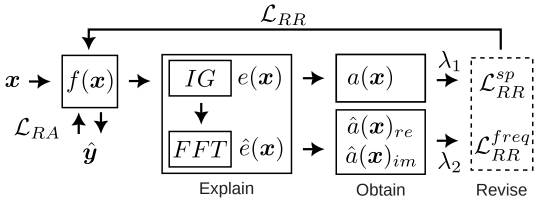

Figure 2: Overview of RioT: Model explanations $e(\boldsymbol{x})$ and annotations $a(\boldsymbol{x})$ are utilized in the *Revise* step to update the model via the right-reason loss $\mathcal{L}_{RR}$. RioT utilizes explanations and feedback in the time and frequency domain.

3.1 Explain

Given a pair of input $\boldsymbol{x}$ and model output $\hat{\boldsymbol{y}}$ for time series classification, an 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. $\boldsymbol{x}$. For an input element, large attribution means a large influence on the output. In the remainder of the paper, we drop f from the notation to declutter it, resulting in $e(\boldsymbol{x})$. We use integrated gradients (IG) [30] as explainer, an established gradient-based attribution method. However, we take the absolute value of the input to make it more suitable for time series and model revision (Eq. 1, further details in App. A.2).

$$e(\boldsymbol{x}) = |\boldsymbol{x} - \bar{\boldsymbol{x}}| \cdot \int_0^1 \frac{\partial f(\tilde{\boldsymbol{x}})}{\partial \tilde{\boldsymbol{x}}} \bigg|_{\tilde{\boldsymbol{x}} = \bar{\boldsymbol{x}} + \alpha(\boldsymbol{x} - \bar{\boldsymbol{x}})} d\alpha \quad (1)$$

$$e(\boldsymbol{x}) = \frac{1}{W} \sum_{i=1}^{W} e'_i(\boldsymbol{x}) \quad (2)$$

Attributions for Forecasting. In a classification setting, attributions are generated by propagating gradients back from the highest activated class to the inputs. However, there is 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 $e'_1(\boldsymbol{x}), \ldots e'_W(\boldsymbol{x})$. This number of explanations would make it even harder for humans to interpret the results, as the size of the explanation increases with W [20]. 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.

Attributions in the Frequency Domain. Time series data is often given in the frequency representation. Sometimes, this format is more intuitive for humans to understand than the spatial representations. As a result, providing explanations in this domain is essential. [32] 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 transfer 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 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 user feedback on confounding factors. For an input $\boldsymbol{x}$, a user can mark parts that are confounded, resulting in a feedback mask $a(\boldsymbol{x}) \in \{0,1\}^T$. In this binary mask, a 1 signals a potential confounder at this time step. Similarly, feedback can also be given on the frequency explanation, marking which elements in the frequency domain are potential confounders. 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 $\hat{A}$. As the annotated feedback masks have to come from human experts, obtaining them can become costly. In many cases, however, confounders occur systematically and it is therefore possible to apply the same annotation mask to multiple samples [25]. This can drastically reduce 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 [25, 23, 29] 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 annotated areas. To achieve model revision and good task performance, both losses are combined, where λ is a hyper-parameter to balance both parts of the combined loss $\mathcal{L}(\mathcal{X}, \mathcal{Y}, A, E) = \mathcal{L}_{\text{RA}}(\mathcal{X}, \mathcal{Y}) + \lambda \mathcal{L}_{\text{RR}}(A, E)$. The combined loss simultaneously optimizes the primary training objective and feedback alignment.

Time Domain Feedback. Masking parts of the time domain can be used to mitigate spatially locatable confounders. We use explanations E and annotations A in the spatial 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 input parts. This effectively trains the model to disregard the masked parts of the input 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, if the confounder is not locatable in time, giving spatial feedback cannot be used to revise the models' behavior. Thus, we utilize frequency explanations $\hat{E}$ and annotations $\hat{A}$ in the frequency right-reason loss:

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

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 at hand, it is possible to use RioT either in the spatial or frequency domain. Moreover, it is also possible to combine feedback in both domains, thus including two right-reason terms in the final loss. This results in two parameters λ_1 and λ_2 to balance the individual losses.

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

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 [10], is also possible.

Table 1: **Applying RioT mitigates confounders in time series classification (left) and forecasting (right).** Performance before and after applying RioT for spatial (Base_{sp}) and frequency ($\text{Base}_{\text{freq}}$) confounders separately. Unconfounded represents an ideal scenario without confounders.

Model	Fault Detection (ACC ↑)				Weather (MSE ↓)			
	FCN		OFA		PatchTST		TiDE	
	Train	Test	Train	Test	Train	Test	Train	Test
Unconf	0.99 ±0.00	0.99 ±0.00	1.00 ±0.00	0.98 ±0.02	0.26 ±0.03	0.08 ±0.01	0.25 ±0.02	0.03 ±0.00
Base_{sp}	**1.00** ±0.00	0.74 ±0.06	**1.00** ±0.00	0.53 ±0.02	**0.20** ±0.03	0.19 ±0.01	**0.22** ±0.03	0.15 ±0.01
+ RioT_{sp}	0.98 ±0.01	**0.93** ±0.03	0.96 ±0.08	**0.98** ±0.01	0.55 ±0.20	**0.14** ±0.01	0.25 ±0.03	**0.11** ±0.01
$\text{Base}_{\text{freq}}$	**0.98** ±0.01	0.87 ±0.03	**1.00** ±0.00	0.72 ±0.02	**0.63** ±0.09	0.24 ±0.04	**0.79** ±0.09	0.31 ±0.09
+ $\text{RioT}_{\text{freq}}$	0.94 ±0.00	**0.90** ±0.03	0.96 ±0.02	**0.98** ±0.02	0.96 ±0.02	**0.17** ±0.00	1.12 ±0.36	**0.22** ±0.01

4 Experimental Evaluations

Experimental Setup. We perform experiments on several datasets from the UCR/UEA repository [6], including FAULT DETECTION and SLEEP for classification and WEATHER for forecasting. In the evaluations, we compare several models with and without RioT on confounded versions of these datasets, as well as on the newly introduced dataset P2S. More details can be found in App. A.2.

Table 2: **Applying RioT overcomes the confounder in P2S.** Performance on confounded train set and the unconfounded test set. FCN learns the train confounder, resulting in a drop in test performance. Applying RioT with partial feedback *(2)* already yields good improvements, while adding feedback on the full confounder area *(4)* is even better. Unconfounded is the ideal scenario, specifically curated so that there is no confounder.

P2S (ACC ↑)	Train	Test
$\text{FCN}_{\text{Unconfounded}}$	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

Production Press Sensor Data (P2S). RioT aims to mitigate confounders in time series data. To assess our method, we need datasets with annotated real-world confounders. So far, there are no such datasets available. To fill this gap, we introduce PRODUCTION PRESS SENSOR DATA (P2S)[3], a dataset of sensor recordings with naturally occurring confounders. The sensor data comes from a high-speed press production line for metal parts, one of the sheet metal working industry's most economically significant processes. The task is to predict whether a run is defective based on the sensor data. The recordings include different production speeds, which, although not affecting part quality, influence process friction and applied forces. An expert identified regions in the time series that vary with production speed, potentially distracting models from relevant classification indicators, especially when no defect and normal runs of the same speed are in the training data. Thus, the run's speed is a confounder, challenging models to generalize beyond training. The default P2S setting includes normal and defect runs of different speeds, with an unconfounded setting of runs at the same speed. Further details on the dataset are available in App. B.

Confounders. To evaluate how well RioT can mitigate confounders in a more controlled setting, we add spatial (sp) or frequency (freq) shortcuts to the datasets from the UCR and Darts repositories. These create spurious correlations between patterns and class labels or forecasting signals in the training data, but are absent in validation or test data. We generate an annotation mask based on the confounder area or frequency to simulate human feedback. More details can be found in App. A.5.

4.1 Evaluations

Removing Confounders for Time Series Classification. We evaluate the effectiveness of RioT (spatial: RioT_{sp}, frequency: $\text{RioT}_{\text{freq}}$) in addressing confounders in classification tasks by comparing balanced accuracy with and without RioT. As shown in Tab. 1 (left), without RioT, both FCN and OFA overfit to shortcuts, achieving ≈100% training accuracy while having poor test performance.

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

Applying RioT significantly improves test performance for both models. In some cases, RioT even reaches the performance of the ideal reference (unconfounded) scenario.

Removing Confounders for Time Series Forecasting. Confounders are not exclusive to time series classification and can significantly impact other tasks, such as forecasting. In Tab. 1 (right), we outline that the models overfit to the confounders, but applying RioT improves test MSE. Specifically, in the frequency-confounded setting, the training data includes a recurring Dirac impulse as a distracting confounder instead of previous shortcut confounders (cf. App. A.5 for details). $\text{RioT}_{\text{freq}}$ alleviates this distraction and improves the test performance significantly.

In Tab. 8, we further investigate RioT's performance in handling both spatial and frequency confounders simultaneously. While models with RioT may not always match the ideal unconfounded scenario, it consistently improves classification and forecasting results in these confounded settings. Additional results on other datasets, models and feedback configurations are provided in App. A.6.

Removing Confounders in the Real-World. So far, our experiments have demonstrated the ability to counteract confounders within controlled environments. However, real-world scenarios often have more complex confounder structures, as in our newly proposed dataset P2S. Fig. 3 (top) highlights that the model focuses specifically on the two middle regions of a sample. With domain knowledge, it's clear that these regions shouldn't 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 incorrect regions, which can be mitigated by extending the annotated area. In Tab. 2, the model performance in these settings is presented. Without RioT, the model cannot generalize to data without the confounder. RioT with partial feedback already improves the performance *(2)* and improves even more with full feedback *(4)*. This highlights the effectiveness of RioT in real-world scenarios, even when not all confounders are (initially) known.

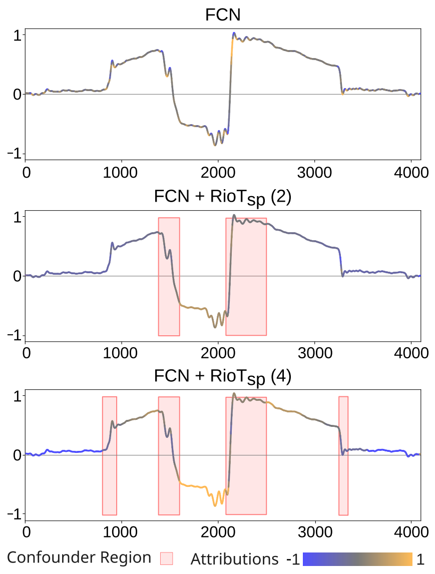

Figure 3: **Applying RioT makes the model ignore annotated confounder areas.** While FCN primarily focuses on confounder areas, applying RioT with partial feedback (middle) or full feedback (bottom) causes the model to ignore the confounder and focus on the remainder of the input.

Limitations. An important aspect of RioT is the feedback provided in the Obtain step. Integrating human feedback into the model is a key advantage of RioT, but can also be a limitation. While we have shown that a small fraction of annotated samples can be sufficient, and that annotations can be applied for many samples, they are still necessary for RioT. Additionally, like many other (explanatory) interactive learning methods, RioT assumes correct human feedback. Thus, considering repercussions of inaccurate feedback when applying RioT in practice is important.

5 Conclusion

In this work, we present Right on Time a method to mitigate confounding factors in time series data with the help of human 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 manually confounded datasets and the new and naturally confounded, real-world dataset P2S showcases that they are indeed subject to confounders. Our results, however, demonstrate that applying RioT to these models can mitigate confounders in the data. Furthermore, RioT is the first method to incorporate feedback in both time and frequency domains to mitigate confounders across both domains. Extending the application of RioT to multivariate time series represents a logical next step. Additionally, applying $\text{RioT}_{\text{freq}}$ to other modalities can offer there a more nuanced approach to confounder mitigation. It should be noted that while our method shows potential in its current iteration, interpreting attributions in time series data remains a general challenge.

Acknowledgment

This work 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 P2S. Furthermore, this work benefited from the HMWK project ”The Third Wave of Artificial Intelligence - 3AI”.

References

[1] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7):e0130140, 2015.

[2] K. Benidis, S. S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, F.-X. Aubet, L. Callot, and T. Januschowski. Deep Learning for Time Series Forecasting: Tutorial and Literature Survey. *ACM Computing Surveys*, 55(6):1–36, 2023.

[3] I. Bica, A. M. Alaa, and M. Van Der Schaar. 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.

[4] D. Cao, J. Enouen, Y. Wang, X. Song, C. Meng, H. Niu, and Y. Liu. Estimating treatment effects from irregular time series observations with hidden confounders. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[5] A. Das, W. Kong, A. Leach, S. Mathur, R. Sen, and R. Yu. Long-term Forecasting with TiDE: Time-series Dense Encoder. *ArXiv:2304.08424*, 2023.

[6] H. A. Dau, A. J. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. J. Keogh. The UCR time series archive. *ArXiv:1810.07758*, 2018.

[7] W. D. Flanders, M. Klein, L. Darrow, M. Strickland, S. Sarnat, J. Sarnat, L. Waller, A. Winquist, and P. Tolbert. A Method for Detection of Residual Confounding in Time-Series and Other Observational Studies. *Epidemiology*, 22(1):59–67, 2011.

[8] F. Friedrich, W. Stammer, P. Schramowski, and K. Kersting. A typology for exploring the mitigation of shortcut behaviour. *Nature Machine Intelligence*, 5(3):319–330, 2023.

[9] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[10] A. Graps. An introduction to wavelets. *IEEE Computational Science and Engineering*, 2(2):50–61, 1995.

[11] T. Hatt and S. Feuerriegel. Sequential deconfounding for causal inference with unobserved confounders. In *Proceedings of the Conference on Causal Learning and Reasoning (CLeaR)*, 2024.

[12] J. Herzen, F. Lässig, S. G. Piazzetta, T. Neuer, L. Tafti, G. Raille, T. V. Pottelbergh, M. Pasieka, A. Skrodzki, N. Huguenin, M. Dumonal, J. Kościsz, D. Bader, F. Gusset, M. Benheddi, C. Williamson, M. Kosinski, M. Petrik, and G. Grosch. Darts: User-Friendly Modern Machine Learning for Time Series. *Journal of Machine Learning Research (JMLR)*, 23(124):1–6, 2022.

[13] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery (DMKD)*, 34(6):1936–1962, 2020.

[14] I. Koprinska, D. Wu, and Z. Wang. Convolutional Neural Networks for Energy Time Series Forecasting. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2018.

[15] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1):1096, 2019.

[16] C. Ma, G. Dai, and J. Zhou. 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.

[17] Q. Ma, Z. Liu, Z. Zheng, Z. Huang, S. Zhu, Z. Yu, and J. T. Kwok. A survey on time-series pre-trained models. *ArXiv:2305.10716*, 2023.

[18] J. Martens. Deep learning via hessian-free optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

[19] N. Mehdiyev, J. Lahann, A. Emrich, D. Enke, P. Fettke, and P. Loos. Time Series Classification using Deep Learning for Process Planning: A Case from the Process Industry. *Procedia Computer Science*, 114:242–249, 2017.

[20] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence (AIJ)*, 267:1–38, 2019.

[21] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[22] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[23] A. S. Ross, M. C. Hughes, and F. Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[24] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall. 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.

[25] P. Schramowski, W. Stammer, S. Teso, A. Brugger, F. Herbert, X. Shao, H.-G. Luigs, A.-K. Mahlein, and K. Kersting. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence*, 2(8):476–486, 2020.

[26] R. R. Selvaraju, S. Lee, Y. Shen, H. Jin, S. Ghosh, L. Heck, D. Batra, and D. Parikh. Taking a HINT: Leveraging Explanations to Make Vision and Language Models More Grounded. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[27] X. Shao, A. Skryagin, W. Stammer, P. Schramowski, and K. Kersting. Right for Better Reasons: Training Differentiable Models by Constraining their Influence Functions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[28] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje. Not Just a Black Box: Learning Important Features Through Propagating Activation Differences. *ArXiv:1605.01713*, 2017.

[29] W. Stammer, P. Schramowski, and K. Kersting. 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.

[30] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[31] S. Teso and K. Kersting. Explanatory Interactive Machine Learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, 2019.

[32] J. Vielhaben, S. Lapuschkin, G. Montavon, and W. Samek. Explainable AI for Time Series via Virtual Inspection Layers. *ArXiv:2303.06365*, 2023.

[33] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[34] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin. One fits all: Power general time series analysis by pretrained LM. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

A Appendix

A.1 Impact Statement

Our research advances machine learning by enhancing the interpretability and reliability of time series models, significantly impacting human interaction with AI systems. By developing Right on Time (RioT), which guides models to focus on correct reasoning, we improve the transparency and trust in machine learning decisions. While human feedback can provide many benefits, one has also to be aware that it could be incorrect, and evaluate the consequences carefully.

A.2 Implementation and Experimental Details

Adaption of Integrated Gradients (IG). A part of IG is a multiplication of the model gradient with the input itself, improving the explanation's quality [28]. However, this multiplication makes some implicit assumptions about the input format. In particular, it assumes that there are no inputs with negative values. Otherwise, multiplying the attribution score with a negative input would flip the attribution's sign, which is not desired. For images, this is unproblematic because they are always equal to or larger than zero. In time series, negative values can occur and normalization to make them all positive is not always suitable. To avoid this problem, we use only the input magnitude and not the input sign to compute the IG attributions.

Computing Explanations. To compute explanations with Integrated Gradients, we followed the common practice of using a baseline of zeros. The standard approach worked well in our experiments, so we did not explore other baseline choices in this work. For the implementation, we utilized the widely-used Captum[4] library, where we patched the `captum._utils.gradient.compute_gradients` function to allow for the propagation of the gradient with respect to the input to be propagated back into the parameters.

Metrics. In our evaluations, we compare the performance of models on confounded and unconfounded datasets with and without RioT. For classification, we report balanced (multiclass) accuracy (ACC), and for forecasting the mean squared error (MSE). The corresponding mean absolute error (MAE) results can be found in App. A.6. We report average and standard deviation over 5 runs.

Model Training and Hyperparameters. For time series classification, we use the FCN model [17], with a slightly modified architecture for Sleep to achieve a better unconfounded performance (cf. App. A.2). Additionally, we use the OFA model [34]. For forecasting, we use the recently introduced TiDE model [5], PatchTST [21] and NBEATS [22] to highlight the applicability of our method to a variety of model classes.

To find suitable parameters for model training, we performed a hyperparameter search over batch size, learning rate, and the number of training epochs. We then used these parameters for all model trainings and evaluations, with and without RioT. In addition, we selected suitable λ values for RioT with a hyperparameter selection on the respective validation sets. The exact values for the model training parameters and the λ values can be found in the provided code.

To avoid model overfitting on the forecasting datasets, we performed shifted sampling with a window size of half the lookback window.

Code. For the experiments, we based our model implementations on the following repositories:

- FCN: `https://github.com/qianlima-lab/time-series-ptms/blob/master/model/tsm_model.py`
- OFA: `https://github.com/DAMO-DI-ML/NeurIPS2023-One-Fits-All/`
- NBEATS: `https://github.com/unit8co/darts/blob/master/darts/models/forecasting/nbeats.py`
- TiDE: `https://github.com/unit8co/darts/blob/master/darts/models/forecasting/tide_model.py`
- PatchTST: `https://github.com/awslabs/gluonts/tree/dev/src/gluonts/torch/model/patch_tst`

[4]`https://github.com/pytorch/captum`

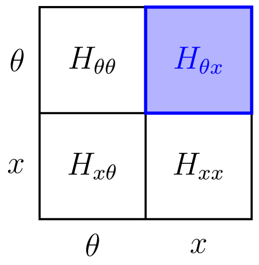

Figure 4: Illustration of the Hessian matrix with its respective sub-blocks. The mapping from x into θ is highlighted in blue.

All experiments were executed using our Python 3.11 and PyTorch code, which is available in the provided code. To ensure reproducibility and consistency, we utilized Docker. Configurations and Python executables for all experiments are provided in the repository.

Hardware. To conduct our experiments, we utilized single GPUs from Nvidia DGX2 machines equipped with A100-40G and A100-80G graphics processing units.

By maintaining a consistent hardware setup and a controlled software environment, we aimed to ensure the reliability and reproducibility of our experimental results.

A.3 Datasets

We perform experiments on various datasets. For classification, we focus mainly on the UCR/UEA repository [6], which holds a wide variety of datasets for this task. The data originates from different domains, e.g., health records, industrial sensor data, and audio signals. We select univariate datasets of a minimal size, resulting in FAULT DETECTION A, FORD A, FORD B, and SLEEP. We omit experiments on the very small datasets of UCR, as these are generally less suited for deep learning [13]. We use the splits provided by the UCR archive. For time series forecasting, we evaluate on three popular datasets of the Darts repository [12]: ETTM1, ENERGY, and WEATHER with 70%/30% train/test splits. These datasets are sufficiently large, allowing us to investigate the impact of confounding behavior in isolation without the risk of overfitting. We standardize all datasets as suggested by [33], i.e., rescaling the distribution of values to zero mean and a standard deviation of one.

A.4 Computational Costs of RioT

Training a model with RioT induces additional computational costs. The right-reason term requires computations of additional gradients. Given a model $f_\theta(x)$, parameterized by θ and input x, then computing the right reason loss with a gradient-based explanation method requires the computation of the mixed-partial derivative $\frac{\partial^2 f_\theta(x)}{\partial\theta\partial x}$, as a gradient-based explanation includes the derivative $\frac{\partial f_\theta(x)}{\partial x}$. While this mixed partial derivative is a second order derivative, this does not substantially increase the computational costs of our method for two main reasons. First, we are never explicitly materializing the Hessian matrix. Second, the second-order component of our loss can be formulated as a Hessian-vector product:

$$\frac{\partial\mathcal{L}}{\partial\theta} = g + \frac{\lambda}{2} H_{\theta x}(e(x) - a(x)) \tag{6}$$

where $g = \frac{\partial\mathcal{L}_{\text{RA}}}{\partial\theta}$ is the partial derivative of the right answer loss and if H is the full joint Hessian matrix of the loss with respect to θ and x, then $H_{\theta x}$ is the sub-block of this matrix mapping x into θ (cf. Fig. 4), given by $H_{\theta x} = \frac{\partial^2 f_\theta(x)}{\partial\theta\partial x}$. Hessian-vector products are known to be fast to compute [18], enabling the right-reason loss computation to scale to large models and inputs.

A.5 Details on Confounding Factors

In the datasets which are not P2S, we added synthetic confounders to evaluate the effectiveness of confounders. In the following, we provide details on the nature of these confounders in the four settings:

Classification Spatial. For classification datasets, spatial confounders are specific patterns for each class. The pattern is added to every sample of that class in the training data, resulting in a spurious correlation between the pattern and the class label. Specifically, we replace T time steps with a sine wave according to:

$$\mathit{confounder} := \sin(t \cdot (2 + j)\pi)$$

while $t \in \{0, 1, \ldots, T\}$ and j represents the class index, simulating a spurious correlation between the confounder and class index.

Classification Frequency. Similar to the spatial case, frequency confounders for classification are specific patterns added to the entire series, altering all time steps by a small amount. The confounder is represented as a sine wave and is applied additively to the full sequence ($T = S$):

$$\mathit{confounder} := \sin(t \cdot (2 + j)\pi) \cdot A$$

where A resembles the confounder amplitude.

Forecasting Spatial. For forecasting datasets, spatial confounders are shortcuts that act as the actual solution to the forecasting problem. Periodically, data from the time series is copied back in time. This "back-copy" is a shortcut for the forecast, as it resembles the time steps of the forecasting window. Due to the windowed sampling from the time series, this shortcut occurs at every second sample. The exact confounder formulation is outlined in the sketch below (Fig. 5), with an exemplary lookback length of 9, forecasting horizon of 3 and window stride of 6. This results in a shortcut confounder in samples 1 and 3 (marked red) and overlapping in sample 2 (marked orange).

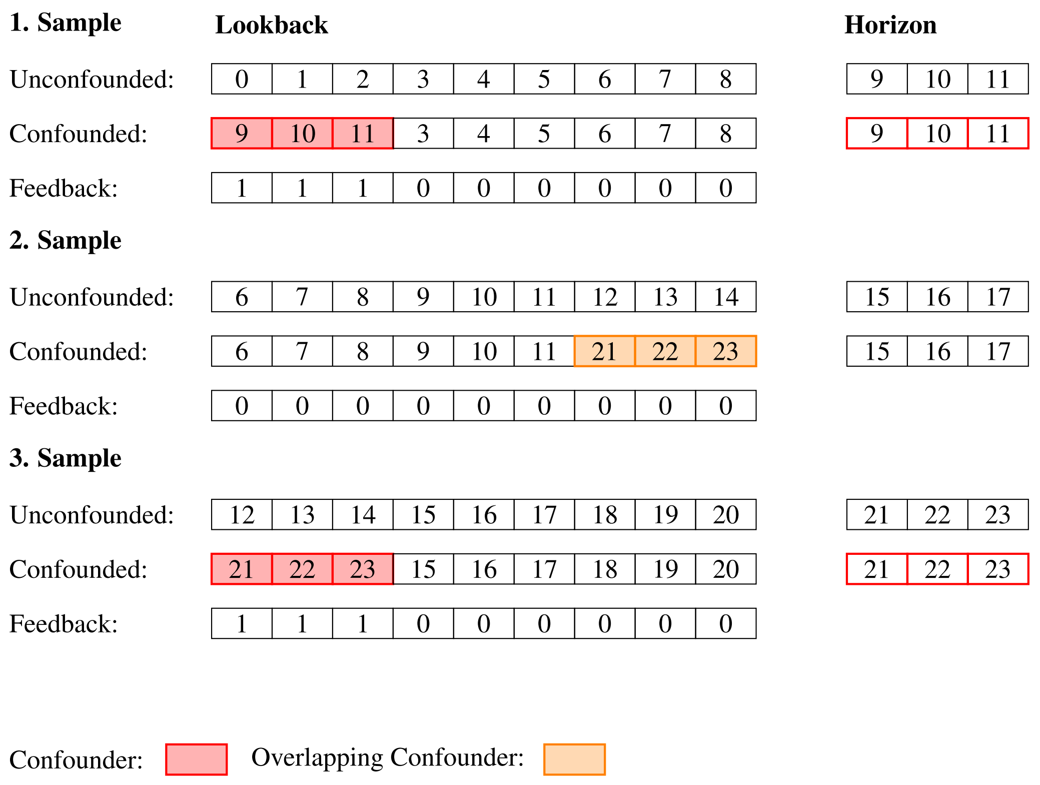
1. Sample
Lookback
Horizon
Unconfounded: 0 1 2 3 4 5 6 7 8 | 9 10 11
Confounded: 9 10 11 3 4 5 6 7 8 | 9 10 11
Feedback: 1 1 1 0 0 0 0 0 0
2. Sample
Unconfounded: 6 7 8 9 10 11 12 13 14 | 15 16 17
Confounded: 6 7 8 9 10 11 21 22 23 | 15 16 17
Feedback: 0 0 0 0 0 0 0 0 0
3. Sample
Unconfounded: 12 13 14 15 16 17 18 19 20 | 21 22 23
Confounded: 21 22 23 15 16 17 18 19 20 | 21 22 23
Feedback: 1 1 1 0 0 0 0 0 0
Confounder:
Overlapping Confounder:

Figure 5: Schematic overview of how the time series were confounded in the spatial forecasting experiments

Forecasting Frequency. This setting differs from the previous shortcut confounders. The frequency confounder for forecasting is a recurring Dirac impulse with a certain frequency, added every k

time steps over the entire sequence (of length S), including the forecasting windows. This impulse is present throughout all of the training data, distracting the model from the real forecast. The confounder is present at all time steps: $i \in \{n \cdot k | n \in \mathbb{N} \land n \cdot k \leq S\}$ with a strength of A:

$$confounder := A \cdot \Delta_i$$

In conclusion, confounders are only present in the training data, not validation or test data. We generate an annotation mask based on the confounder area or frequency to simulate human feedback. This mask is applied to all confounded samples except in our feedback scaling experiment.

A.6 Additional Experimental Results

This section provides further insights into our experiments, covering both forecasting and classification tasks. Specifically, it showcases performance through various metrics such as MAE, MSE, and accuracy, while also exploring different feedback configurations and the impact of invalid feedback.

Table 3: **Applying RioT mitigates confounders in time series classification.** Performance before and after applying RioT for spatial (SP Conf) and frequency (Freq Conf) confounders separately. Unconfounded represents the ideal scenario where the model is not affected by any confounder.

Model	Config (ACC ↑)	Fault Detection A		FordA		FordB		Sleep	
		Train	Test	Train	Test	Train	Test	Train	Test
FCN	Unconfounded	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
	Base$_{freq}$	**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
	+ RioT$_{freq}$	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	Unconfounded	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
	Base$_{freq}$	**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
	+ RioT$_{freq}$	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

Table 4: **RioT can successfully overcome confounders in time series forecasting.** MSE values (MAE values cf. Tab. 5) on the confounded training set and the unconfounded test set with Unconfounded being the ideal scenario where the model is not affected by any confounder.

Model	Config (MSE ↓)	ETTM1		Energy		Weather	
		Train	Test	Train	Test	Train	Test
NBEATS	Unconfounded	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
	Base$_{freq}$	**0.30** ±0.02	0.46 ±0.01	**0.33** ±0.04	0.36 ±0.04	**0.11** ±0.02	0.32 ±0.09
	+ RioT$_{freq}$	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	Unconfounded	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
	Base$_{freq}$	**0.45** ±0.03	0.91 ±0.16	**0.04** ±0.00	0.53 ±0.05	**0.63** ±0.09	0.24 ±0.04
	+ RioT$_{freq}$	0.91 ±0.07	• **0.66** ±0.04	2.45 ±4.59	• **0.38** ±0.06	0.96 ±0.02	• **0.17** ±0.00
TiDE	Unconfounded	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
	Base$_{freq}$	**0.06** ±0.01	0.69 ±0.08	**0.07** ±0.01	0.34 ±0.08	**0.79** ±0.09	0.31 ±0.09
	+ RioT$_{freq}$	0.07 ±0.01	• **0.49** ±0.07	**0.07** ±0.01	• **0.21** ±0.02	1.12 ±0.36	• **0.22** ±0.01

Feedback Generalization. Tab. 7 and Tab. 6 detail provided feedback percentages for forecasting and classification across all datasets, respectively. These tables report the performance of the

Table 5: **RioT can successfully overcome confounders in time series forecasting.** MAE values on the confounded training set and the unconfounded test set with Unconfounded being the ideal scenario where the model is not affected by any confounder.

Model	Config (MAE ↓)	ETTM1		Energy		Weather	
		Train	Test	Train	Test	Train	Test
NBEATS	Unconfounded	0.39 ±0.01	0.48 ±0.01	0.44 ±0.02	0.38 ±0.01	0.21 ±0.01	0.12 ±0.01
	Base$_{sp}$	**0.34** ±0.01	0.54 ±0.01	**0.44** ±0.03	0.77 ±0.01	**0.21** ±0.01	0.30 ±0.04
	+ RioT$_{sp}$	0.40 ±0.01	● **0.52** ±0.01	0.53 ±0.02	● **0.62** ±0.01	0.23 ±0.01	● **0.22** ±0.01
	Base$_{freq}$	**0.39** ±0.01	0.47 ±0.01	**0.45** ±0.03	0.45 ±0.03	**0.21** ±0.03	0.45 ±0.06
	+ RioT$_{freq}$	0.40 ±0.01	● **0.47** ±0.01	**0.45** ±0.03	● **0.44** ±0.02	0.59 ±0.22	● **0.39** ±0.01
PatchTST	Unconfounded	0.50 ±0.01	0.49 ±0.01	0.39 ±0.00	0.38 ±0.01	0.38 ±0.03	0.18 ±0.00
	Base$_{sp}$	**0.46** ±0.00	0.53 ±0.01	**0.41** ±0.00	0.78 ±0.01	**0.32** ±0.04	0.33 ±0.00
	+ RioT$_{sp}$	**0.46** ±0.01	● **0.52** ±0.01	0.51 ±0.00	● **0.53** ±0.01	0.54 ±0.12	● **0.28** ±0.00
	Base$_{freq}$	**0.53** ±0.01	0.81 ±0.07	**0.15** ±0.00	0.64 ±0.03	**0.58** ±0.03	0.41 ±0.05
	+ RioT$_{freq}$	0.92 ±0.05	● **0.80** ±0.02	0.97 ±0.86	● **0.57** ±0.02	0.65 ±0.01	● **0.40** ±0.01
TiDE	Unconfounded	0.36 ±0.01	0.48 ±0.01	0.40 ±0.01	0.38 ±0.02	0.36 ±0.02	0.13 ±0.00
	Base$_{sp}$	**0.32** ±0.01	0.54 ±0.01	**0.40** ±0.01	0.85 ±0.01	**0.32** ±0.03	0.29 ±0.01
	+ RioT$_{sp}$	0.34 ±0.01	● **0.51** ±0.01	0.57 ±0.01	● **0.58** ±0.01	0.35 ±0.03	● **0.24** ±0.01
	Base$_{freq}$	**0.18** ±0.01	0.74 ±0.06	**0.18** ±0.01	0.53 ±0.07	**0.65** ±0.05	0.49 ±0.09
	+ RioT$_{freq}$	0.19 ±0.01	● **0.60** ±0.05	**0.18** ±0.01	● **0.40** ±0.03	0.79 ±0.16	● **0.41** ±0.02

Table 6: Feedback percentage for forecasting across all datasets, reported for the TiDE model. Corresponding to (test) results shown in Fig. 6, a higher percentage indicates more feedback, lower is better.

Metric	Feedback	ETTM1		Energy		Weather	
		Spatial	Freq	Spatial	Freq	Spatial	Freq
MAE (↓)	0%	0.54 ±0.01	0.74 ±0.06	0.85 ±0.01	0.53 ±0.07	0.29 ±0.01	0.49 ±0.09
	5%	0.52 ±0.00	0.63 ±0.03	0.62 ±0.01	**0.40 ± 0.02**	0.28 ±0.01	0.43 ±0.03
	10%	0.52 ±0.00	0.63 ±0.03	0.61 ±0.01	**0.40 ± 0.02**	0.27 ±0.01	0.43 ±0.03
	25%	0.52 ±0.00	0.63 ±0.03	0.58 ±0.01	0.41 ±0.01	0.25 ±0.01	0.43 ±0.04
	50%	0.52 ±0.00	0.63 ±0.03	**0.57 ± 0.01**	0.41 ±0.01	**0.24 ± 0.01**	0.44 ±0.05
	75%	0.52 ±0.01	0.63 ±0.03	**0.57 ± 0.01**	0.41 ±0.01	**0.24 ± 0.01**	0.45 ±0.06
	100%	**0.51 ± 0.01**	**0.60 ± 0.05**	0.58 ±0.01	**0.40 ± 0.03**	**0.24 ± 0.01**	**0.41 ± 0.02**
MSE (↓)	0%	0.54 ±0.03	0.69 ±0.08	1.19 ±0.03	0.34 ±0.08	0.15 ±0.01	0.31 ±0.09
	5%	0.54 ±0.01	0.52 ±0.03	0.60 ±0.02	**0.20 ± 0.01**	0.14 ±0.01	0.24 ±0.02
	10%	0.53 ±0.01	0.52 ±0.03	0.57 ±0.02	**0.20 ± 0.01**	0.14 ±0.01	0.24 ±0.02
	25%	0.53 ±0.01	0.52 ±0.03	0.53 ±0.02	0.22 ±0.01	**0.11 ± 0.01**	0.24 ±0.03
	50%	0.53 ±0.01	0.52 ±0.03	**0.51 ± 0.02**	0.22 ±0.01	**0.11 ± 0.01**	0.25 ±0.04
	75%	0.52 ±0.01	0.51 ±0.03	0.52 ±0.02	0.22 ±0.01	**0.11 ± 0.01**	0.26 ±0.05
	100%	**0.48 ± 0.01**	**0.49 ± 0.07**	0.52 ±0.02	0.21 ±0.02	**0.11 ± 0.01**	**0.22 ± 0.01**

TIDE and FCN models, highlighting how different levels of feedback impact model outcomes on various datasets. Tab. 6 focuses on MAE and MSE for forecasting, while Tab. 7 presents ACC for classification.

As human feedback is an essential aspect of RioT, we investigate the required annotations and the potential to generalize annotations across samples. Our findings indicate that not every sample needs annotation. Fig. 6 shows that we can significantly reduce the amount of annotated data for classification and forecasting (cf. App. Tab. 7 and Tab. 6 for results on the other datasets). Even minimal feedback, such as annotating just 5% of the samples, substantially improves performance compared to no feedback.

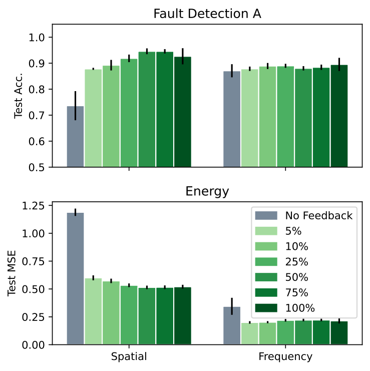

Figure 6: **RioT uses feedback efficiently.** Even with feedback on only a small percentage of the data, RioT can overcome confounders.

Table 7: Feedback percentage for classification across all datasets, reported for the FCN model. Corresponding to results shown in Fig. 6, a higher percentage indicates more feedback, higher is better.

Feedback	Fault Detection A (ACC ↑)		FordA (ACC ↑)		FordB (ACC ↑)		Sleep (ACC ↑)	
	Spatial	Freq	Spatial	Freq	Spatial	Freq	Spatial	Freq
0%	0.74 ±0.06	0.87 ±0.03	0.71 ±0.08	0.73 ±0.01	0.63 ±0.03	0.60 ±0.01	0.10 ±0.03	0.27 ±0.02
5%	0.88 ±0.00	0.88 ±0.01	0.81 ±0.03	0.80 ±0.03	0.66 ±0.03	**0.66 ± 0.02**	0.53 ±0.03	**0.49 ± 0.00**
10%	0.89 ±0.02	0.89 ±0.01	0.82 ±0.04	0.79 ±0.02	0.66 ±0.03	0.64 ±0.03	0.48 ±0.09	0.48 ±0.02
25%	0.92 ±0.01	0.89 ±0.01	0.83 ±0.02	0.78 ±0.01	0.67 ±0.02	0.65 ±0.01	0.49 ±0.08	0.42 ±0.08
50%	**0.95 ± 0.01**	0.88 ±0.01	0.82 ±0.03	0.81 ±0.05	0.67 ±0.02	0.65 ±0.00	**0.55 ± 0.03**	0.44 ±0.07
75%	**0.95 ± 0.01**	0.88 ±0.01	0.81 ±0.03	0.80 ±0.04	0.65 ±0.03	0.64 ±0.00	0.54 ±0.04	0.44 ±0.07
100%	0.93 ±0.03	**0.90 ± 0.03**	**0.84 ± 0.02**	**0.83 ± 0.02**	**0.68 ± 0.02**	0.65 ±0.01	0.54 ±0.05	0.45 ±0.07

Furthermore, the results on P2S highlights that annotations can be generalized across multiple samples. Once the confounder on P2S has been identified on a couple of samples, the expert annotations can be used on full dataset. The systematic nature of shortcut confounders suggest that generalizing annotations is an effective possibility to obtain feedback efficiently. While RioT does rely on human annotations, these findings highlight that it can work without extensive manual human interactions, and that obtained annotations can be utilized efficiently.

Sensitivity of RioT Regarding Invalid Feedback. We evaluated RioT's sensitivity to increasing amounts of invalid feedback for classification and forecasting in both the frequency and spatial domains. Instead of having annotations at the confounding area, the feedback instead marks random time steps (frequency components) as confounded. Fig. 7 outlines the results of this experiment. We observe that our method is relatively robust against invalid feedback. Notably, with 10% invalid feedback, there is minimal degradation in all evaluated scenarios, which should be a reasonable threshold for real-world tasks. In some scenarios (e.g. forecasting with spatial confounders), RioT can still provide substantial improvement even under high feedback noise. The difference in robustness can potentially be attributed to the different behavior of the models as well as dataset and confounder differences. Overall, the experiment indicates that RioT is generally robust against smaller amounts of invalid feedback, which even improves the usability of this method in practice.

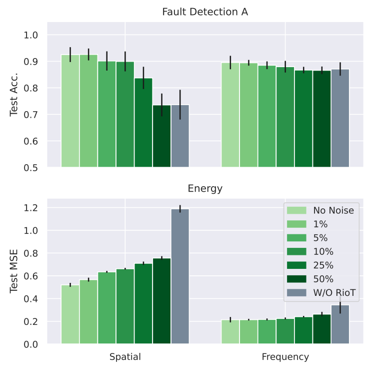

Figure 7: **RioT is robust against invalid feedback.** Results on Fault Detection A and Energy with increasing amounts of invalid feedback.

Removing Confounders for Time Series Forecasting. Tab. 5 reports the MAE results for our forecasting experiment across different models, datasets and configurations. It emphasizes how well each model performs on both the confounded training set and after applying RioT, with the Unconfounded configuration representing the ideal scenario unaffected by confounders.

Removing Multiple Confounders at Once. In the previous experiments, we illustrated that RioT is suitable for addressing individual confounding factors, whether spatial or frequency-based. Real-world time series data, however, often present a blend of multiple confounding factors that simultaneously may influence model performance.

We thus investigate the impact of applying RioT to both spatial and frequency confounders simultaneously (cf. Tab. 8), exemplary using FCN and TiDE. When Sleep is confounded in both domains, FCN without RioT overfits and fails to generalize. Addressing only one confounder does not mitigate the effects, as the model adapts to the other. However, combining feedback for both domains (RioT$_{\text{freq,sp}}$) significantly improves test performance, matching the frequency-confounded scenario (cf. Tab. 3). Tab. 8 (bottom) shows the impact of multiple confounders on the Energy dataset for forecasting. When faced with both spatial shortcut and noise confounders, the model overfits, indicated by lower training MSE. While applying either spatial or frequency feedback individually already shows some effect, utilizing both types of feedback simultaneously (RioT$_{\text{freq,sp}}$) results in the largest improvements, as

Table 8: **RioT can combine spatial and frequency feedback.** If the data is confounded in the time and frequency domain, RioT can combine feedback on both domains to mitigate confounders, superior to feedback on only one domain. Unconfounded represents the ideal scenario when the model is not affected by any confounder.

Sleep (Classification ACC ↑)	Train	Test
$\text{FCN}_{\text{Unconfounded}}$	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**
$\text{TiDE}_{\text{Unconfounded}}$	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
Energy (Forecasting MAE ↓)	**Train**	**Test**
$\text{TiDE}_{\text{Unconfounded}}$	0.40 ±0.01	0.38 ±0.02
$\text{TiDE}_{\text{freq,sp}}$	**0.30** ±0.01	0.70 ±0.02
$\text{TiDE}_{\text{freq,sp}}$ + RioT_{sp}	0.34 ±0.01	0.64 ±0.01
$\text{TiDE}_{\text{freq,sp}}$ + $\text{RioT}_{\text{freq}}$	0.36 ±0.01	0.60 ±0.01
$\text{TiDE}_{\text{freq,sp}}$ + $\text{RioT}_{\text{freq,sp}}$	0.39 ±0.01	• **0.55** ±0.01

both confounders are addressed. The performance gap between $\text{RioT}_{\text{freq,sp}}$ and the non-confounded model is more pronounced than in single confounder cases (cf. Tab. 4), suggesting a compounded challenge. Optimizing the deconfounding process in highly complex data environments thus remains an important challenge.

B Confounded Dataset from a High-speed Progressive Tool

The presence of confounders is a common challenge in practical settings, affecting models in diverse ways. As the research community strives to identify and mitigate these issues, it becomes imperative to test our methodologies on datasets that mirror the complexities encountered in actual applications. However, for the time domain, datasets with explicitly labeled confounders are not present, highlighting the challenge of assessing model performance against the complex nature of practical confounding factors.

To bridge this gap, we introduce P2S, a dataset that represents a significant step forward by featuring explicitly identified confounders. This dataset originates from experimental work on a production line for deep-drawn sheet metal parts, employing a progressive die on a high-speed press. The sections below detail the experimental approach and the process of data collection.

B.1 Real-World setting

The production of parts in multiple progressive forming stages using stamping, deep drawing and bending operations with progressive dies is generally one of the most economically significant manufacturing processes in the sheet metal working industry and enables the production of complex parts on short process routes with consistent quality. For the tests, a four-stage progressive die was used on a Bruderer BSTA 810-145 high-speed press with varied stroke speed. The strip material to be processed is fed into the progressive die by a BSV300 servo feed unit, linked to the cycle of the press, in the stroke movement while the tools are not engaged. The part to be produced remains permanently connected to the sheet strip throughout the entire production run. The stroke height of the tool is 63 mm and the material feed per stroke is 60 mm. The experimental setup with the progressive die set up on the high-speed press is shown in Fig. 8.

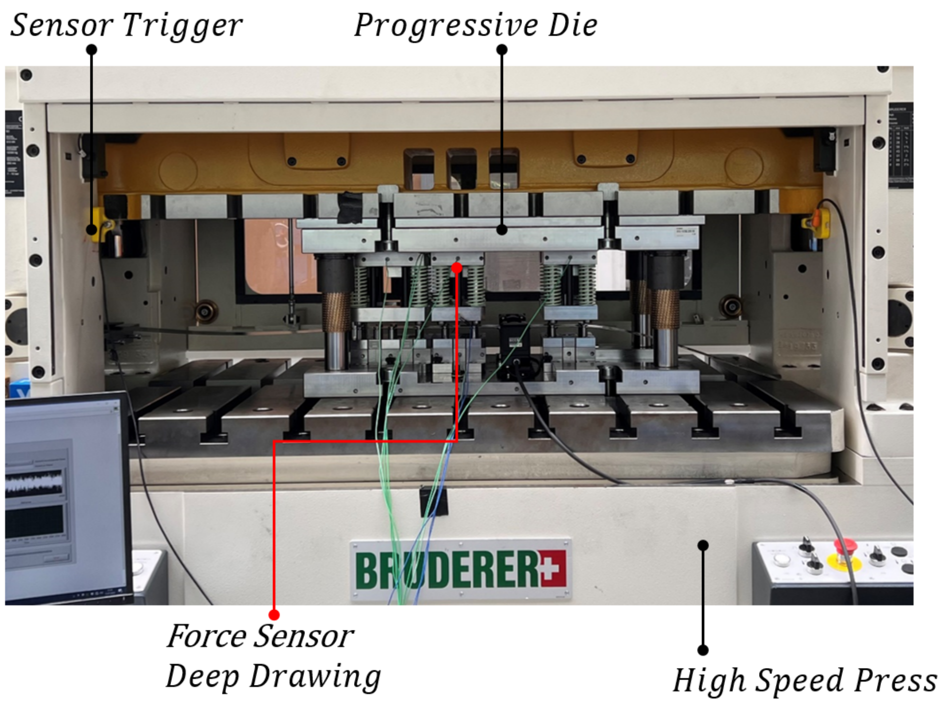

Figure 8: Experimental setup with high-speed press and tool as well as trigger for stroke-by-stroke recording of the data

The four stages include a pilot punching stage, a round stamping stage, deep drawing and a cut-out stage. In the first stage, a 3 mm hole is punched in the metal strip. This hole is used by guide pins in the subsequent stages to position the metal strip. During the stroke movement, the pilot pin always engages in the pilot hole first, thus ensuring the positioning accuracy of the components. In the subsequent stage, a circular blank is cut into the sheet metal strip. This is necessary so that the part can be deep-drawn directly from the sheet metal strip. This is a round geometry that forms small arms in the subsequent deep-drawing step that hold the component on the metal strip. In the final stage, the component is then separated from the sheet metal strip and the process cycle is completed. The respective process steps are performed simultaneously so that each stage carries out its respective process with each stroke and therefore a part is produced with each stroke. Fig. 9 shows the upper tool unfolded and the forming stages associated with the respective steps on the continuous sheet metal strip.

B.2 Data collection

An indirect piezoelectric force sensor (Kistler 9240A) was integrated into the upper mould mounting plate of the deep-drawing stage for data acquisition. The sensor is located directly above the punch and records not only the indirect process force but also the blank holder forces which are applied by spring assemblies between the upper mounting plate and the blank holder plate. The data is recorded at a sampling frequency of 25 kHz. The material used is DC04 with a width of 50 mm and a thickness of 0.5 mm. The voltage signals from the sensors are digitised using a "CompactRIO" (NI cRIO 9047) with integrated NI 9215 measuring module (analogue voltage input $\pm$ 10 V). Data recording is started via an inductive proximity switch when the press ram passes below a defined stroke height during the stroke movement and is stopped again as it passes the inductive proximity switch during the return stroke movement. Due to the varying process speed caused by the stroke speeds that have been set, the recorded time series have a different number of data points. Further, there are slight variations in the length of the time series withing one experiment. For this reason, all time series are interpolated to a length of 4096 data points and we discard any time series that deviate considerably from the mean length of a run (i.e., threshold of 3). A total of 12 series of experiments, shown in Tab. 9, were carried out with production rates from 80 to 225 spm. To simulate a defect, the spring hardness of

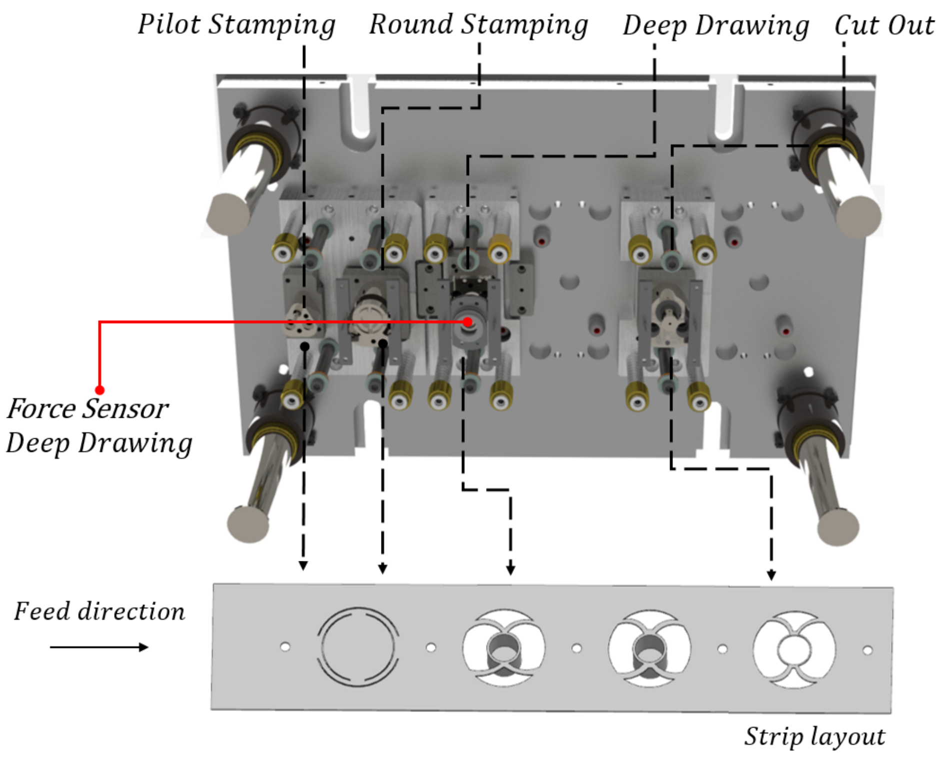

Figure 9: Upper tool unfolded and the forming stages associated with the respective steps on the passing sheet metal strip as well as the positions of the piezoelectric force sensors.

the blank holder was manipulated in the test series that were marked as *defect*. The manipulated experiments result in the component bursting and tearing during production. In a real production environment, this would lead directly to the parts being rejected.

B.3 Data characteristics

Fig. 10 shows the progression of the time series recorded with the indirect force sensor. The force curve characterises the process cycle during a press stroke. The measurement is started by the trigger which is activated by the ram moving downwards. The downholer plates touch down at point A and press the strip material onto the die. Between point A and point B, the downholder springs are compressed, causing the applied force to increase linearly. The deep drawing process begins at point B. At point C, the press reaches its bottom dead centre and the reverse stroke begins so that the punch moves out of the material again. At point D, the deep-drawing punch is released from the material and now the hold-down springs relax linearly up to point E. At point E, the downholder plate lifts off again, the component is fed to the next process step and the process is complete.

B.4 Confounders

The presented dataset P2S is confounded by the speed with which the progressive tool is operated. The higher the stroke rate of the press, the more friction is occurring and the higher is the impact of the downholder plate. The differences can be observed in Fig. 11. Since we are aware of these physics-based confounders, we are able to annotate them in our dataset. As the process speed increases, the friction that occurs between the die and the material in the deep-drawing stage changes, as the frictional force is dependent on the frictional speed. This is particularly evident in the present case, as deep-drawing oils, which can optimize the friction condition, were not used in the experiments. The affected area from friction of the punch are in 1380 to 1600 (start of deep drawing) and 2080 to 2500 (end of deep drawing). In addition, the impulse of the downholder plate affecting the die increases due to the increased process dynamics. If the process speed is increased, the process force

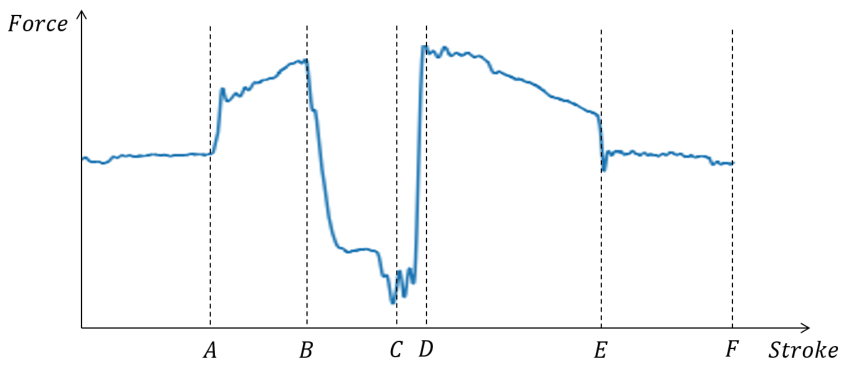

Figure 10: Force curve for one stroke. A) set down downholder plate B) start of deep drawing C) bottom dead centre D) deep drawing process completed E) downholder plates lift off F) measurement stops.

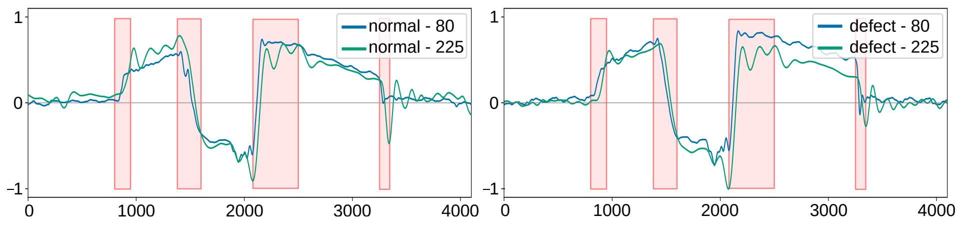

Figure 11: Samples of P2S with normal (left) and defect (right) setting at 80 and 225 strokes per minute. Areas that vary depending on the stroke rate and are considered confounding and marked red.

also increases in the ranges of the time series from 800 to 950 (downholder plate sets down) and 3250 to 3550 (downholder plate lifts).

In the experiment setting of Tab. 2, the training data set is selected in such a way that the stroke rate correlates with the class label, i.e., there are only normal experiments with small stroke rates and defect ones with high stroke rate. Experiment 1, 2, 3, 10, 11, 12 are the training data and the remaining experiments are the test data. To get a unconfounded setting where the model is not affected by any confounder, we use normal and defect experiments with the same speed in training and respectively test data. This results in experiments 1, 3, 5, 7, 9, 11 in the training set and the remaining in the test set.

Table 9: Overview of conducted runs on the high-speed press with normal and defect states at different stroke rates.

Experiment #	State	Stroke Rate	Samples
1	Normal	80	193
2	Normal	100	193
3	Normal	150	189
4	Normal	175	198
5	Normal	200	194
6	Normal	225	188
7	Defect	80	149
8	Defect	100	193
9	Defect	150	188
10	Defect	175	196
11	Defect	200	193
12	Defect	225	190
Total			2264