

Explainable time-series forecasting with sampling-free SHAP for Transformers

Matthias Hertel^{1*}, Sebastian Pütz¹, Ralf Mikut¹, Veit Hagenmeyer¹, Benjamin Schäfer¹

¹Institute for Automation and Applied Informatics (IAI), Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany.

*Corresponding author(s). E-mail(s): matthias.hertel@kit.edu;

Contributing authors: sebastian.puetz@kit.edu; ralf.mikut@kit.edu; veit.hagenmeyer@kit.edu; benjamin.schaefer@kit.edu;

Abstract

Time-series forecasts are essential for planning and decision-making in many domains. Explainability is key to building user trust and meeting transparency requirements. Shapley Additive Explanations (SHAP) is a popular explainable AI framework, but it lacks efficient implementations for time series and often assumes feature independence when sampling counterfactuals. We introduce SHAPformer, an accurate, fast and sampling-free explainable time-series forecasting model based on the Transformer architecture. It leverages attention manipulation to make predictions based on feature subsets. SHAPformer generates explanations in under one second, several orders of magnitude faster than the SHAP Permutation Explainer. On synthetic data with ground truth explanations, SHAPformer provides explanations that are true to the data. Applied to real-world electrical load data, it achieves competitive predictive performance and delivers meaningful local and global insights, such as identifying the past load as the key predictor and revealing a distinct model behavior during the Christmas period.

Keywords: Explainable AI (XAI), time series, forecasting, Transformers, electrical load, Shapley Additive Explanations (SHAP)

1 Introduction

Time-series forecasting plays a critical role in numerous domains, including logistics, retail, finance, healthcare, business services, traffic management, and energy systems [1]. In the energy sector, forecasts are becoming increasingly important due to the ongoing transition to renewable energy sources [2], which is essential to mitigate the impacts of anthropogenic climate change [3]. The inherent variability of renewable generation makes accurate forecasts of electrical load and generation crucial to ensure real-time balance between supply and demand [4]. Moreover, forecasts enable early detection of critical grid conditions and help prevent equipment overloads through demand-side response and redispatch measures [5].

Across domains, modern forecasting models increasingly rely on complex deep learning architectures [6, 7]. These models often involve a large number of parameters, which makes their internal processes opaque to humans. This has driven the development of Explainable Artificial Intelligence (XAI) methods, which aim to improve the understanding of how machine learning models operate [8]. Human interpretability is valuable for several reasons. First, it can enhance user trust and facilitate the adoption of deep learning in real-world applications [9]. Second, explanations help users assess the reliability of predictions and decide how to act on them [10]. Third, they assist developers in debugging and refining models—for example, by revealing spurious patterns or “Clever Hans” effects [11]. Finally, XAI supports compliance with emerging regulations that demand transparency in AI systems. The European Union’s AI Act, for instance, mandates transparency for applications

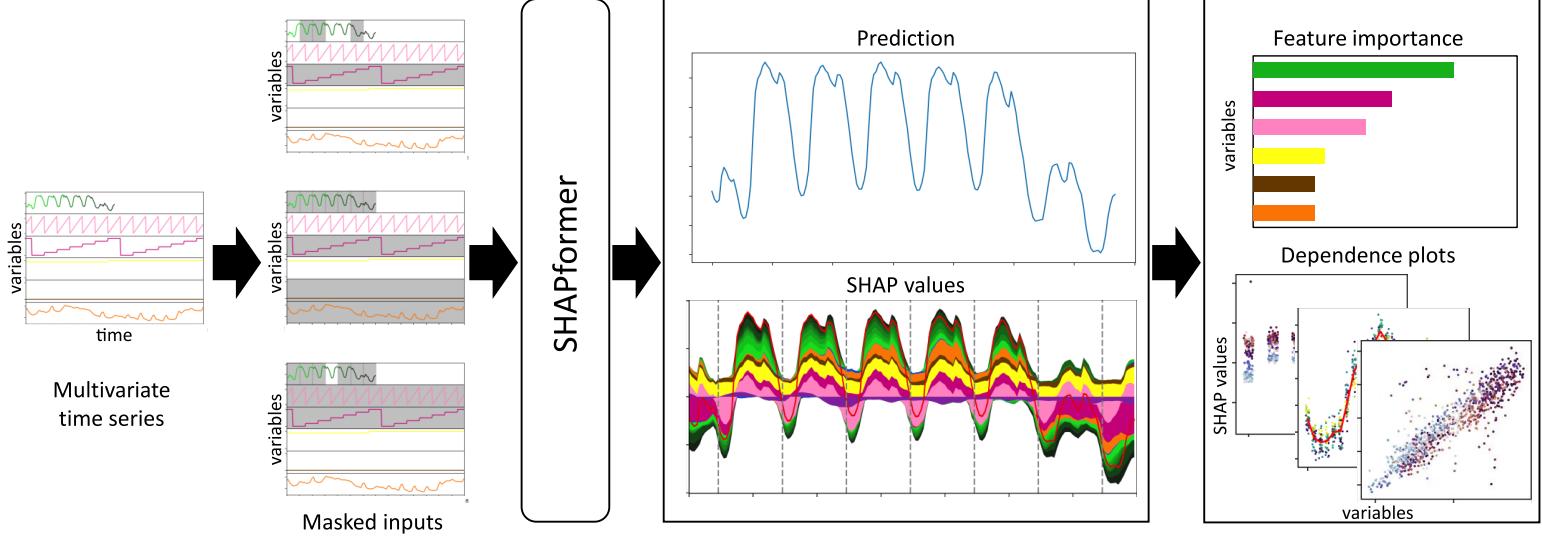


Fig. 1: Overview of the proposed method. The model receives the past target and covariate time-series as input. The features are grouped as indicated by colors. SHAPformer makes predictions based on masked inputs. SHAP values are derived from the marginal feature contributions, defined as the difference of the prediction with and without a feature group. Multiple local explanations of forecasts get combined into global feature importance values and feature dependence plots.

that involve humans or critical infrastructure such as energy systems [12]. Although not always required, XAI can support human oversight and improve transparency [10], and may justify future regulatory standards for transparency in AI systems.

While comparably simple models, including linear regressors and generalized additive models, are inherently interpretable, more complex and powerful models often require post-hoc explanation methods to make their predictions understandable [13]. Post-hoc explanation methods are often perturbation-based, which means that they systematically modify the model inputs and observe the corresponding changes in the model output. A widely used post-hoc XAI framework is Shapley Additive Explanations (SHAP) [14]. It is the most popular XAI framework in the energy sector [15] and has been used to explain electrical load forecasts [16] produced by models such as long short-term memories [17, 18], multilayer perceptrons [19] and tree-based algorithms [20–23]. Based on cooperative game theory, SHAP estimates the contribution of each feature to the prediction and satisfies the efficiency property, which means that the contributions sum up to the prediction [14]. SHAP is more informative than mere feature attribution methods, as it gives not only feature importance but also the direction (positive or negative) and magnitude of a feature’s impact on the prediction. SHAP supports local explanations of individual predictions, and global explanations such as feature importance scores and dependence plots, which are aggregations of many local explanations [24]. Other post-hoc XAI methods, including LIME [25], Grad-CAM [26] and Layer-wise Relevance Propagation (LRP) [27, 28], only highlight important inputs without indicating how they affect the prediction, or do not fulfill the efficiency property.

Several algorithms for the estimation of SHAP values have been developed [29], which quantify the marginal contribution of each feature by evaluating a model based on subsets of features. These estimations typically follow one of two strategies: (a) sampling absent features from a marginal or conditional distribution, or (b) replacing absent features with a predefined baseline value. Both approaches have limitations. Sampling is computationally intensive and can produce unrealistic inputs that fall outside the data distribution, leading to off-the-manifold evaluations [30]. Baseline substitution requires careful selection of the baseline and only explains deviations from that specific reference point. TimeSHAP [31] is a SHAP version for time series that estimates SHAP values for variables and time steps based on replacing data points with the mean value. WindowSHAP [32] estimates SHAP values for time windows, where absent windows are sampled or replaced.

For Transformer models [33], it is common to use attention weights as proxies for feature importance, visualizing them to highlight influential inputs [34, 35]. However, this practice remains controversial, as the interpretability of attention mechanisms is debated [36, 37]. The Temporal Fusion Transformer (TFT) [38] computes feature importance values with a dedicated feature selection layer, but it gives no information about how

the features influence the prediction, and it lacks the efficiency property of SHAP. An efficient implementation of SHAP for time-series Transformers is currently missing [16].

In this context, we present a new algorithm to estimate SHAP values for time-series forecasting models based on attention manipulation [39]. Our contributions are as follows:

1. We introduce SHAPformer, a Transformer-based forecasting model that enables the efficient calculation of exact SHAP values by grouping features and applying attention manipulation [39] to exclude absent feature groups. This approach eliminates the need for sampling or setting baseline values. An overview of the method is shown in Figure 1.
2. We validate SHAPformer’s explanations on a synthetic load forecasting dataset with known ground truth explanations, demonstrating its ability to accurately capture the underlying patterns.
3. We showcase the benefits of SHAPformer applied to empirical load data from the transmission system operator (TSO) TransnetBW [40]. SHAPformer is several orders of magnitude faster than the widely used SHAP Permutation Explainer and provides meaningful local and global insights.

2 Results

We evaluate SHAPformer on two datasets:

- Synthetic dataset: We generate a synthetic dataset exhibiting daily, weekly, and annual seasonality with dependencies on exogenous covariates (holidays and a multiplier), for which ground truth explanations are available. The details of the data generation are presented in Section 4.3. 100,000 examples are used for training and 10,000 each for validation and testing.
- TransnetBW dataset: This dataset contains hourly electrical load measurements from the German TSO TransnetBW for the years 2015–2019 [40] and weather data from the Copernicus ERA5 reanalysis model [41]. The last six months are used for testing and the six months before for validation.

SHAPformer is compared against a standard Transformer model explained by two SHAP algorithms, the Permutation Explainer and a Custom Masker, as well as against state-of-the-art forecasting models (detailed in Section 4). All models have one week context length and forecast the 168 hourly values for the following week. Forecast accuracy, training time, and inference time for all methods are summarized in Table 1. For the real-world dataset, average results from five repeated runs with different model initializations are reported. Additional metrics and the standard deviations for the five runs are reported in Appendix A.

Table 1: Test root mean squared error (RMSE), training time and inference time on the synthetic dataset and real dataset. Inference time is the time needed to create one explanation for the Permutation Explainer, Custom Masker and SHAPformer, and to create one forecast for all others.

Approach	SHAP values	Forecast error (RMSE)		Training time [h]		Inference time [s]	
		Synthetic	Real [MW]	Synthetic	Real	Synthetic	Real
Persistence baseline	no	0.152	652.3	-	-	-	-
Linear Regression	no	0.149	553.7	0.00	0.00	0.00	0.00
XGBoost	no	0.119	387.0	0.01	0.00	0.00	0.00
Temporal Fusion Transformer	no	0.059	390.8	5.20	1.84	0.01	0.03
Transformer	no	0.059	263.1	0.90	0.26	0.01	0.01
+ Permutation Explainer	approximate	"	"	"	"	1124.16	484.34
+ Custom Masker	approximate	"	"	"	"	7.84	3.54
SHAPformer	exact	0.060	265.9	10.55	3.46	21.90	0.60

2.1 Fast calculation of exact SHAP values while maintaining forecast quality

As SHAPformer is based on the Transformer architecture, we compare it to a Transformer forecasting model. Explanations for the Transformer are generated with two SHAP algorithms: (1) the Permutation Explainer, which samples absent features from background data under the assumption of feature independence; and (2) a Custom Masker, which groups features into seven load-related groups (one for each of the past seven days) and

one group for each exogenous variable. SHAP values are then computed per group, instead of per feature, by jointly sampling all features within a group.

SHAPformer generates explanations more than $50\times$ faster than the Permutation Explainer on the synthetic data, and more than $800\times$ faster on the real-world data. Remarkably, this is achieved although SHAPformer calculates exact SHAP values based on all feature group subsets, whereas the Permutation Explainer and Custom Masker approximate SHAP values based on ten random feature permutations. The Custom Masker achieves a $136\text{--}143\times$ speedup by reducing the number of feature subsets through grouping. In addition, SHAPformer uses attention manipulation to drop absent feature groups and thereby eliminates the need for sampling and repeated model evaluations, so that it becomes feasible to run it with all feature group subsets. However, SHAPformer’s inference speed advantage comes at the cost of increased training time, ranging from two to ten times longer than that required by TFT or the standard Transformer. The inference times of the Transformer and SHAPformer are larger on the synthetic data than on the real-world data, because the models are larger (see Appendix B for the hyperparameters) and the synthetic dataset contains one covariate more than the real-world dataset.

TFT produces explanations with a single model evaluation, making it the fastest Transformer-based explanation method. However, it is not directly comparable to the SHAP-based approaches, as it provides only feature importance scores but does not provide information on how individual features influence the predictions.

SHAPformer’s forecast accuracy is comparable to or better than the compared models. All evaluated models outperform the persistence baseline, which simply predicts the value from one week earlier. The Transformer achieves the lowest forecast error on both datasets, with SHAPformer close behind, with a forecast error that is only about a 1% higher. On the real-world dataset, both the Transformer and SHAPformer outperform Linear Regression, XGBoost [42], and TFT. The larger forecast error of TFT on the real-world data stems from (a) a larger variance across experiment repetitions (but SHAPformer and the Transformer outperform TFT in all runs), and (b) from large forecast errors before Christmas, where TFT’s causal attention prohibits the model from using the holiday feature of the later days when predicting the first days in the forecast horizon.

2.2 Successful validation of the explanations on synthetic data

Having demonstrated SHAPformer’s forecast accuracy and fast inference, we validate its explanations using synthetic data with known ground truth explanations. The synthetic time series incorporate daily, weekly, and annual seasonality, along with dependencies on covariates. SHAP is used on the data generation process in order to compute ground truth explanations. Details of the dataset and ground truth computation are provided in Section 4.3.

SHAPformer’s global feature importance is close to the ground truth, except that it underestimates the importance of the month feature (Figure 2A). In contrast, the Permutation Explainer and Custom Masker deviate from the ground truth, particularly for key features such as the past target (called “load” as for the real-world data), hour of day, and day of week. TFT is more difficult to interpret, as it produces separate sets of feature importance values for past and future features. Nonetheless, it is observable that neither of the sets resembles the ground truth, nor does a linear combination of the two. Notably, both SHAPformer and the Permutation Explainer effectively filter out irrelevant features such as the two noise covariates, while the other methods assign them importance values greater than zero, although the noise covariates are not related to the target variable.

In addition to feature importance, we compare SHAPformer’s learned feature dependencies (Figure 2B) with the ground truth (Figure 2C), where the strongest interacting variable is indicated by the color of the dots. Overall, SHAPformer’s feature dependencies align well with the ground truth, with the only exception being the underestimation of the month feature importance. In panel (a) of Figure 2B-C, the previous week’s load has a linear effect on the prediction. The same past load results in a higher SHAP value at night (bright dots) than during the day (dark dots), because a normal daytime load is already exceptionally high for the night. In panel (b), the dependence on the hour of day forms a half-sine wave, which is exactly the daily pattern used in the synthetic data generation. This interacts with the multiplier: negative multipliers (dark blue dots) decrease the effect of the hour of day, whereas positive values (light green to yellow) increase the effect. Panel (c) reflects the multiplicative shrinking effect used in the data generation to model weekends (days five and six), with SHAP values closer to zero than on weekdays – negative at night, positive during the day. Panel (d) shows that holidays shrink the predicted load by increasing negative past loads and reducing positive ones, while non-holidays exhibit a reversed, but less pronounced, effect. In panel (e), the month feature has a sinusoidal effect in the ground truth, which SHAPformer partially captures, though with much reduced amplitude. Finally, panel (f) shows an X-shaped pattern for the multiplier feature: higher multipliers amplify positive previous loads and reduce negative ones, whereas lower multipliers have the opposite effect.

Taken together, the fact that SHAPformer’s feature importance and feature dependencies resembles the ground truth confirms that it captures the true dependencies present in the data. Next to the global explanations, SHAPformers local explanations are also consistent with the ground truth, which is shown in Appendix C. The dependence plots and local explanations created with the Permutation Explainer and Custom Masker are presented in Appendix D for comparison.

2.3 Insights into load forecasts on real-world TSO data

With SHAPformer validated on synthetic data, we examine its global explanations for the last 12 months of the real-world dataset (Figure 3). In this case, no ground truth is available, so the quality of the explanations is assessed based on domain knowledge.

SHAPformer identifies the past load as the most important predictor, followed by the day of the week and the hour of the day. Features such as month, temperature, and holidays contribute less and show similar levels of importance, while precipitation has almost no importance. The explanations of the Permutation Explainer and Custom Masker are similar to each other but differ from SHAPformer’s explanations, emphasizing hour of day and day of week, and assigning much lower importance to the past load than SHAPformer. For TFT, the interpretation is not straightforward, as it outputs two distinct sets of feature importance – one for past inputs and one for future covariates – with an unclear relationship between them. In TFT’s case, the most important past feature is the load, while hour of day, temperature, and day of week are the most important future features.

Figure 3B displays SHAPformer’s learned feature dependencies. SHAP values are expressed in terms of standardized electrical load, where a value of 1.0 corresponds to one standard deviation (approximately 1550 MW). In panel (a), a linear relationship between the previous week’s load and the forecasted load is observed. Notably, SHAP values are higher at night (bright dots) for the same load value – likely because a load considered low during the day can be unusually high at night. Panel (b) shows the typical daily load pattern with lower values at night and two peaks around noon and in the evening. On Sundays (yellow dots), SHAP values tend to be closer to zero. Panel (c) illustrates that the predicted load is lower on Saturdays and Sundays, especially during the day (dark dots). Holidays, shown in panel (d), consistently reduce the predicted load, although the effect is weaker at night (bright dots). In panel (e), the month feature reflects a seasonal pattern with higher loads in winter – except in December, likely due to reduced industrial activity during the Christmas period. Finally, panel (f) shows that temperature increases the predicted load on cold days, particularly when daytime temperatures (bright dots) are below 15 °C or nighttime temperatures (dark dots) are below 0 °C.

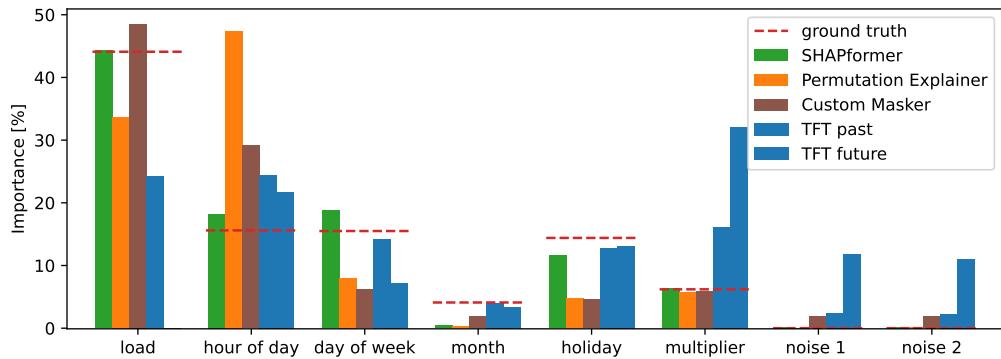
Figure 4 presents two local SHAPformer explanations for forecasts in December. In both cases, the hour of day feature lowers the predictions at night and shows two daily peaks – mirroring the patterns seen in the dependence plots. The day of week feature reduces the predictions on weekends, as expected. Interestingly, in the left-hand example, the month feature has a positive effect despite being in December – likely because the previous week falls in November, which is represented in the month feature of the encoder input. In contrast, the right-hand example, which is later in December, shows a strongly negative month effect. The holiday feature has a positive impact during weekdays, suggesting that the model associates December with lower loads in general due to the long holiday season, except during weeks without holidays. In the left-hand example, the cold temperature in the early forecast days raises the prediction, while the warmer temperature in the last two days decreases the prediction. In the right-hand example, the temperature is more constant and has a consistently positive effect on the prediction. The effect of the past load also differs in the two examples: in the left-hand plot, the past load increases the prediction on weekdays, whereas in the right-hand plot, it decreases the prediction from days 3 to 7 – especially influenced by the last two input days which show a lower average load than the rest.

Overall, the observed dependence patterns align well with domain knowledge, reinforcing confidence in SHAPformer’s predictions. The December pattern – where a lower load is treated as the default and non-holidays as exceptions – is unexpected, but can be plausibly explained by the extended holiday period during that time.

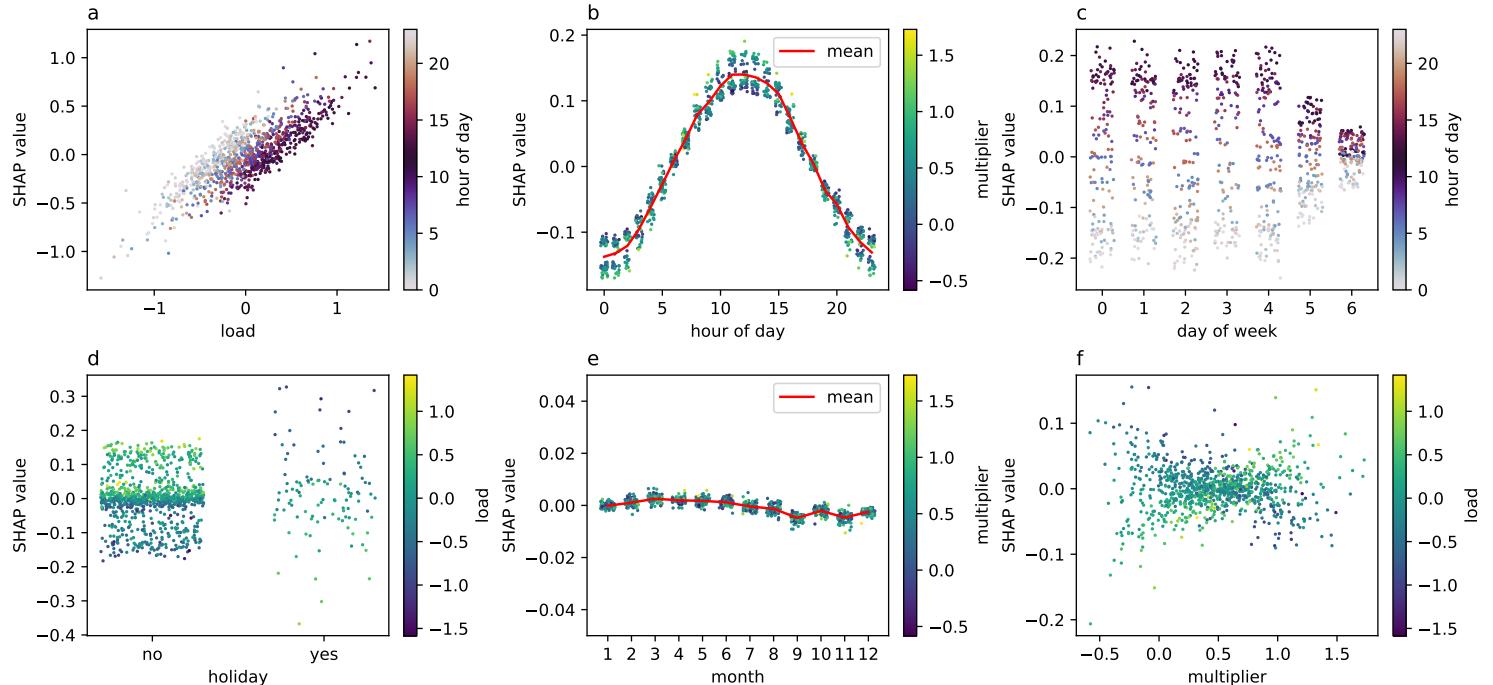
3 Discussion

We introduced SHAPformer, an accurate and efficient approach for explainable time-series forecasting. SHAPformer enables the generation of SHAP explanations for Transformer-based models several orders of magnitude faster than the established SHAP Permutation Explainer. This significant speedup results from two key mechanisms: grouping input features to reduce the number of coalitions, and estimating marginal contributions without sampling by manipulating the attention weights. Both mechanisms contribute to reduced runtime. The

A Feature importance



B SHAPformer dependence plots



C Ground truth dependence plots

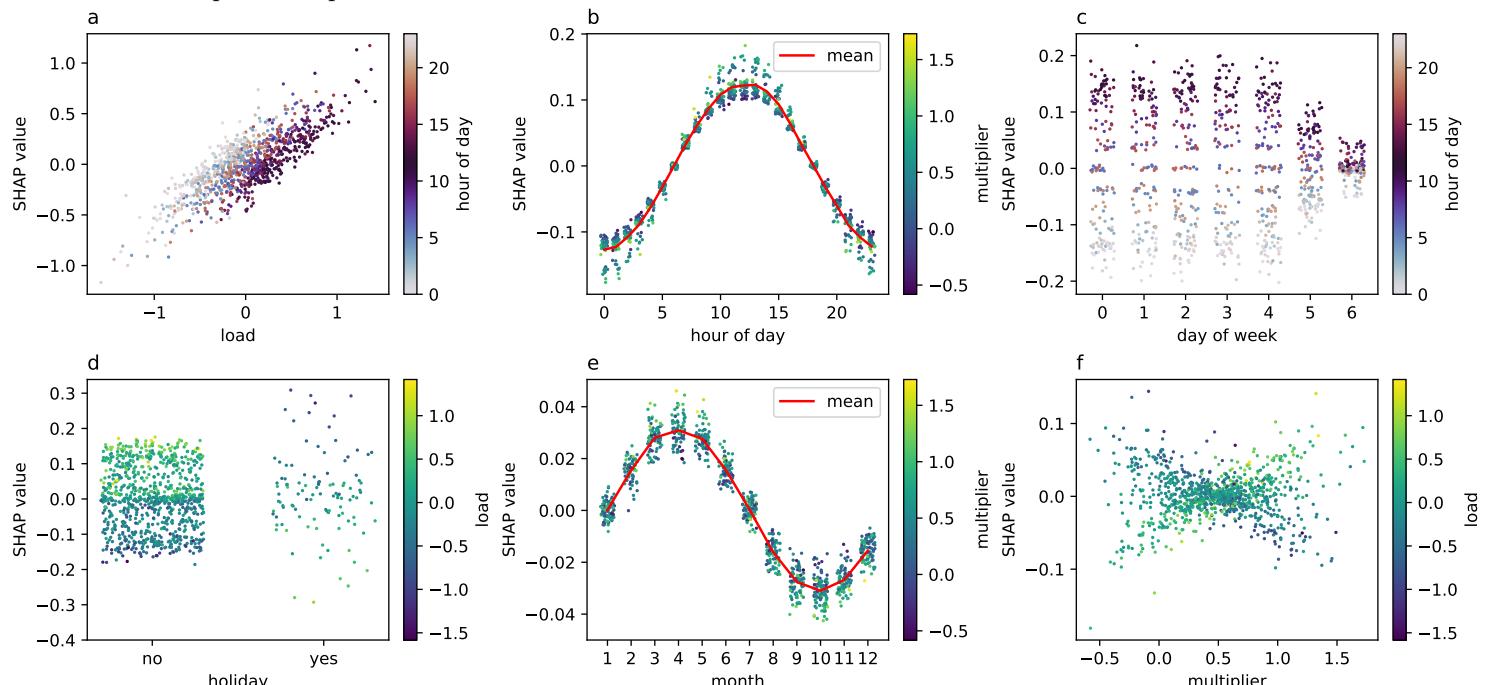
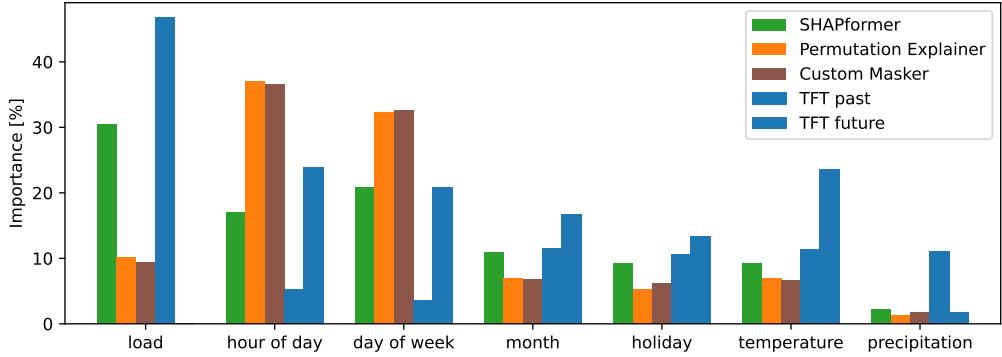


Fig. 2: Global explanations on the synthetic test data. A: SHAPformer approximates ground truth feature importances well (TFT: Temporal Fusion Transformer). B and C: dependence on the six most important features. The strongest interacting feature is shown in color (interactions mean that the SHAP value depends on the interacting variable). For discrete variables, noise was added in the x-direction for visibility reasons.

A Feature importance



B SHAPformer dependence plots

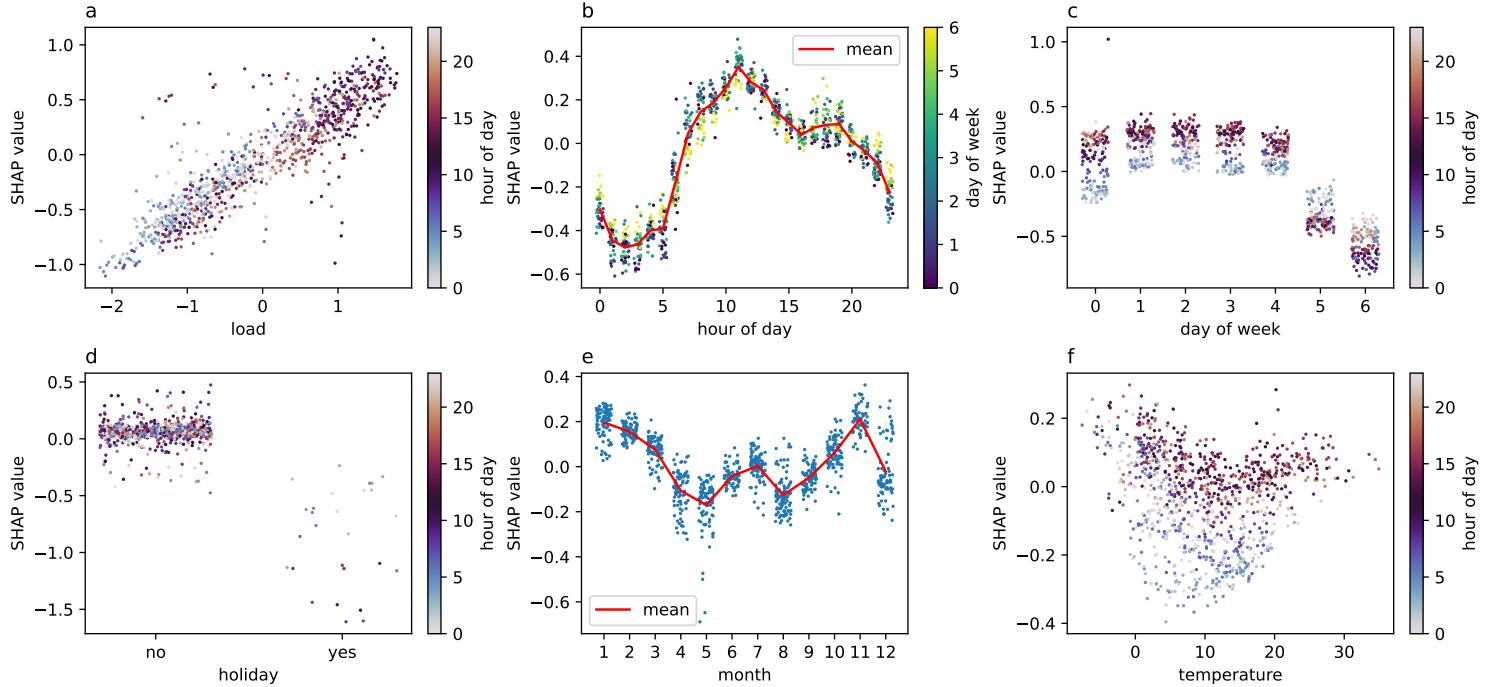


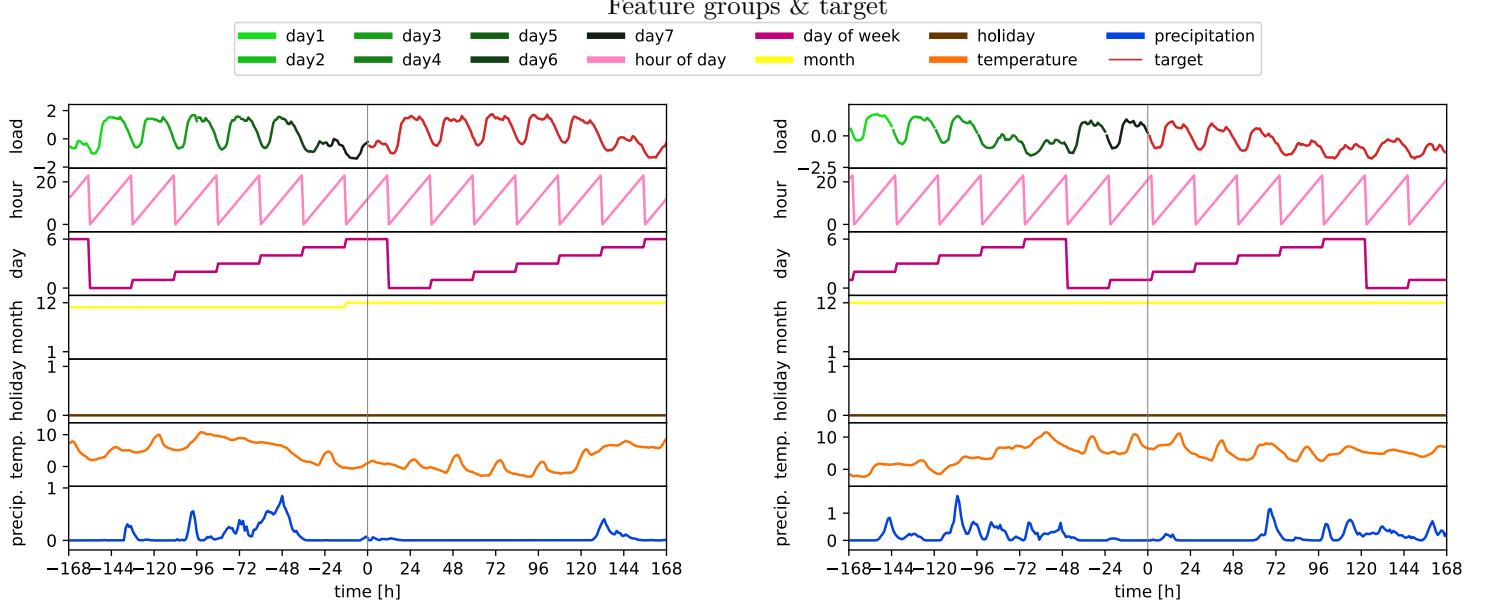
Fig. 3: Global explanations on real-world load data from TransnetBW. A: Feature importance scores by SHAPformer, a Transformer model explained using the Permutation Explainer and the Custom Masker, and Temporal Fusion Transformer (TFT). B: Dependence plots from SHAPformer, with feature values on the x-axis and corresponding SHAP values on the y-axis. The strongest interacting variable is indicated by color. For discrete variables, noise was added in the x-direction for visibility reasons.

Custom Masker, which applies feature grouping but not attention manipulation, is already faster than the Permutation Explainer. However, only SHAPformer achieves inference times below one second per explanation on the real-world data.

SHAPformer's fast inference comes at the cost of increased training time. Training with masked inputs requires the model to learn a more complex structure across varying feature subsets, and since it only sees partial inputs in each iteration, the effective amount of training data per epoch is reduced. Nevertheless, for applications where many forecasts need to be explained, the reduced inference time justifies the additional training effort.

The explanation runtime of SHAPformer increases exponentially with the number of feature groups, because the model is evaluated for all 2^N subsets of N feature groups. In our setup with seven input days and six or seven covariates (depending on the dataset), this full enumeration is computationally feasible and desirable as it yields the most accurate SHAP estimates. However, in scenarios with more feature groups, such as longer input sequences or additional covariates, this exhaustive approach may become impractical. In such cases, SHAPformer can be adapted to evaluate only a subset of the feature coalitions, enabling a runtime that scales linearly with N .

A Data



B SHAPformer

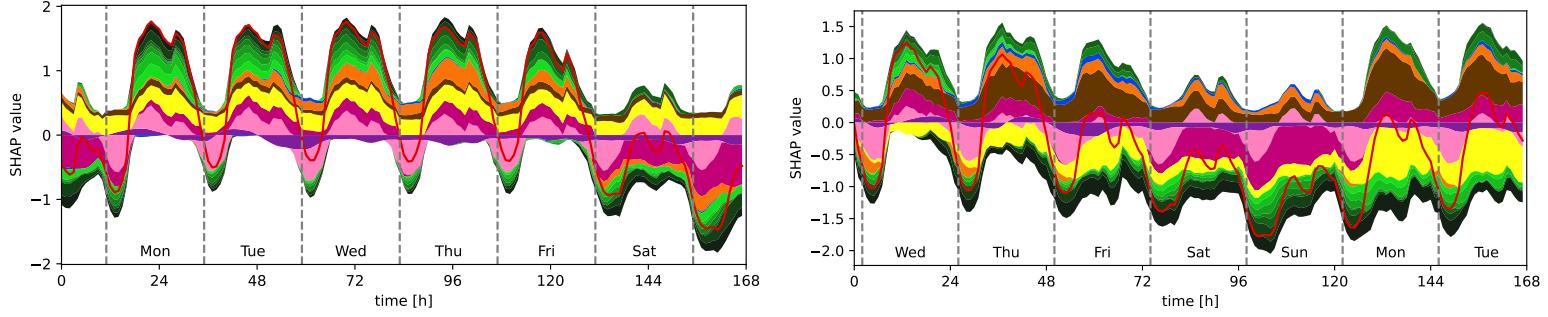


Fig. 4: Local explanations of examples from the real-world load data of TransnetBW. The start of the forecast horizon on the left-hand side is December 1, 2019, 13:00 and on the right-hand side December 17, 2019, 22:00. Panel B shows the prediction as a red line.

SHAPformer has been successfully validated on synthetic data, as it closely reproduces the ground truth in terms of feature importance, dependence patterns, and feature interactions. It effectively filters out irrelevant noise features and captures key relationships, including daily patterns, multiplicative effects of holidays and weekends, and temperature interactions. The only exception is the month feature, whose importance SHAPformer appears to underestimate, likely because its influence in the data generation process is small and comparable to the noise added to the target time series (see Section 4.3).

We publish the synthetic dataset and the ground truth explanations, so that they can be used in the future for the evaluation of XAI methods¹. However, note that this ground truth is created based on multiple assumptions, such as that the ground truth should resemble SHAP values and it should respect dependencies between features (e.g. the hour of day can be inferred from the jumps in the day of week feature). Other explanations might be valid under different assumptions, especially when the goal is to create explanations that are true to a model that has learned different dependencies. We view this evaluation approach as complementary to existing explanation metrics [43] and believe that it can be extended to other data modalities in the future.

Existing SHAP algorithms are either true to the model or true to the data [44]. Conditional sampling tends to produce explanations that are true to the data, while off-the-manifold sampling generates explanations that better reflect the model's internal behavior [30]. We argue that SHAPformer achieves both: it is true to the model and true to the data. This is made possible by its sampling-free design and training strategy, which builds robustness to absent features. In the presence of correlated features, SHAPformer learns to predict accurately using any of the correlated inputs, distributing the predictive contribution among them. This contrasts with

¹The synthetic dataset with ground truth explanations is available on GitHub: <https://github.com/KIT-IAI/SHAPformer>

models trained on the full feature set, which may rely on arbitrary feature combinations, leading to explanations that misrepresent underlying data relationships. SHAPformer’s strong alignment with the ground truth explanations on synthetic data supports this claim.

Applying SHAPformer to real-world load data provides valuable insights into how different features influence the model’s predictions. The most important predictor is the load from the previous week, followed by the day of the week and the hour of the day. Other features – such as month, temperature, and holidays – also contribute, while precipitation has minimal impact. This contrasts with the Transformer model explained by the Permutation Explainer or Custom Masker, which emphasizes temporal features more heavily and downplays the role of the past load. This difference highlights how global explanations can help differentiate between models with similar forecast accuracy but different learned dependencies. In our view, a forecasting model can reasonably rely on both calendar-based inputs or lagged load values to estimate typical load patterns. SHAP values are instrumental in uncovering which of these cues the model relies on. Given that SHAPformer aligns more closely with the ground truth explanations on synthetic data, we are confident that its explanations on real-world data are likewise more faithful to the underlying data relationships than those from the comparison methods.

In the future, the efficient SHAP calculation of SHAPformer can be adopted for other models than Transformers, by training them on feature subsets, which allows to run them on feature subsets at inference time. This can be achieved by using feature dropout instead of attention manipulation. Training models in this way not only facilitates explainability but may also improve robustness to known anomalies and missing values. By dropping out these values with attention manipulation or equivalent mechanisms, such models can adapt more effectively to real-world data imperfections.

We release SHAPformer as a Python package to make it available to researchers and practitioners². This will facilitate its application and evaluation across a range of domains beyond electrical load forecasting. Given its successful validation on synthetic data, we are confident that SHAPformer will generalize well to other use cases.

4 Methods

This section introduces SHAP and two algorithms to estimate SHAP values, the Permutation Explainer and the Custom Masker, in Section 4.1. Next, our method SHAPformer is described in Section 4.2. Then, the synthetic dataset and ground truth generation processes are described in Section 4.3. Finally, the compared forecasting models are described in Section 4.4.

4.1 Shapley Additive Explanations (SHAP)

Shapley values

Shapley values are a concept from game theory that is used to compute a player’s contribution to the outcome of a cooperative game [45]. The player’s contribution is defined as how much the game outcome changes on average when the player enters the game, considering all permutations of players entering the game. The Shapley value definition is the unique solution for the computation of players’ contributions that fulfill the efficiency axiom (the sum of the contributions adds up to the outcome), the symmetry axiom (two identical players receive equal contributions), the dummy axiom (a player that does not contribute to any coalition gets a contribution of zero) and the additivity axiom (the contribution to the sum of two games equals the sum of the contributions of the two games) [14, 46].

Explaining machine learning models with SHAP

The concept of Shapley values is used in machine learning to compute the contribution of features to the prediction of a model [14, 46]. For this, the features are the players and the model prediction is the game outcome. The exact Shapley values usually cannot be computed for computational reasons and because it is not possible to integrate over the unknown distributions of absent features. Therefore, the Shapley values are estimated in practice and are then called SHAP values, in order to distinguish them from the exact Shapley values. The naive algorithm to compute SHAP values works as follows: For a set of features $\{v_1, \dots, v_n\}$, consider all permutations p_j , $1 \leq j \leq n!$. For a permutation p_j , go through all features in the order of the permutation.

²The code for SHAPformer is available on GitHub, with example files demonstrating its use: <https://github.com/KIT-IAI/SHAPformer>

For a feature v_i , let S be the features that precede v_i in p_j . Then, the marginal contribution of v_i is computed as:

$$\phi(v_i, p_j) = f(S \cup \{v_i\}) - f(S) \quad (1)$$

where $f(X)$ is the model prediction using a set of features X . The marginal contribution is the effect on the model prediction when adding v_i to the feature set. The SHAP value for v_i is defined as the average marginal contribution of v_i in all permutations. It can be computed more efficiently by iterating over all 2^n feature subsets instead of all $n!$ feature permutations:

$$\text{SHAP}(v_i) = \sum_{S \subseteq V \setminus \{v_i\}} \frac{(n - 1 - |S|)! \cdot |S|!}{n!} \cdot (f(S \cup \{v_i\}) - f(S)) \quad (2)$$

The first part of the formula is a weight, defined by the fraction of permutations in which v_i gets added to S , and the second part is the marginal contribution of v_i , defined as the difference of the model prediction with and without v_i . Note that to compute all feature contributions, it is necessary to compute the model predictions for all 2^n feature subsets $S \subseteq V$ (also called coalitions) [14, 46].

Usually, a machine learning model trained on a set of features requires all features for inference, which means that it does not allow for predictions based on feature subsets. Therefore, a common approach to estimate predictions based on feature subsets is to marginalize out the effect of the absent features via a Monte Carlo integration – that is, by sampling their values k times from a background dataset (e.g., the training data). Consequently, the model is called k times for each of the feature subsets.

Permutation Explainer

The Permutation Explainer is a more efficient way of estimating SHAP values by calculating marginal contributions only for a subset of the coalitions [46]. It samples K random feature permutations. For each permutation p_j , it starts with the empty feature set and iteratively adds features in the order of the permutation. The marginal contribution of each feature is computed as the prediction with that feature minus the prediction without that feature, i.e. $f(S \cup \{v_i\}) - f(S)$ as above, where S is the set of features preceding v_i in p_j . As in the naive algorithm, predictions based on feature subsets are estimated by a Monte Carlo integration over the absent features (i.e. by sampling their values from background data). Each permutation is used twice, where features are added once in the forward and once in the backward direction. The SHAP value estimate is then computed as the average marginal contribution of a feature over all sampled permutations.

The Permutation Explainer is commonly used in practice, but it has two drawbacks:

1. It is costly to compute for large feature sets. In our forecasting setup on the TransnetBW dataset with 168×7 past features and 168×6 future features (2 184 features in total), ten feature permutations (each run forward and backward), and 100 samples to estimate distributions of absent features, $2 184 \times 10 \times 2 \times 100 = 4 368 000$ model calls are needed to compute a single local explanation of a forecast. The runtimes reported in Table 1 are measured with ten permutations. As this is not feasible to evaluate for many samples, we instead reduce the number of permutations to two to generate the local and global explanations shown in the figures.
2. When feature dependencies are not considered during Monte Carlo sampling, unrealistic samples are created, which are out of distribution of the training data and can potentially lead to arbitrary model behavior. For example, when calendar features are sampled independently, it can happen that Monday midnight is followed by Friday noon. Other examples of unrealistic counterfactuals are 30°C in winter, quick jumps in the electrical load or temperature values, high temperatures or loads at night, and many more.

Note that many algorithms exist that estimate SHAP values [29] and not all of them require sampling. For text and images, usually no sampling is required. Instead, patches of images are blurred or replaced by uncolored pixels. Words are replaced by “...” or removed entirely. Similarly, it is possible to replace time series features by a baseline value of zero, their negative values or the values in reversed order, as is done in perturbation methods [47]. However, choosing a baseline value is not straightforward, and the chosen baseline affects the explanations. After standardization of the continuous time-series features, a value of zero represents the time series’ mean, so applying a zero baseline would only explain the difference to the mean. When a non-holiday is chosen as the baseline for the holiday feature, non-holidays will be attributed SHAP values of zero, as the feature value and the baseline value are the same. For the day-of-week feature, no meaningful sequence of weekdays can be chosen as a baseline, as every sequence appears equally often, whereas constant values (e.g. seven consecutive Mondays or 168 times midnight) are unrealistic.

Custom Masker

We develop a Custom Masker to mitigate the sampling and efficiency issues of the Permutation Explainer. Given an example for which an explanation is to be computed, and a feature coalition (i.e. subset of the features), the Custom Masker defines how the features absent in the coalition get replaced in order to generate alternative samples which are used in the Monte Carlo estimation.

First, we reduce the number of features by defining groups of features that get altered together. The past load is split into seven groups of 24 features, representing the past seven days. All features belonging to the same exogenous time series form one feature group. This makes 13 feature groups in total for the real-world data, and 14 feature groups in total for the synthetic data which exhibits one exogenous feature more. As a result, the model is called $13 \times 10 \times 2 \times 100 = 26\,000$ times to create a local explanation of a forecast on the real-world data.

The Custom Masker samples each feature group differently, respecting the characteristics of the feature and creating more realistic samples than the Permutation Explainer. For a load feature group, 24 consecutive values from the training data are sampled, so that these 24 values form a smooth and realistic curve in themselves (but not necessarily realistic in the context of the other load values and exogenous features). The month values are increased by a constant, randomly chosen offset between 0 and 11 months. In the same way, the day-of-week and hour-of-day values are offset by 0 to 6 days, respectively 0 to 23 hours. For the temperature feature, the entire temperature curve is replaced by 336 consecutive values from the training data.

From local to global explanations

The local explanations of m samples are aggregated to feature importance values by summing up the absolute SHAP values of a feature group in all samples, and then calculating percentages on the summed up values.

Another form of global model explanation are the dependence plots shown in Figures 2 and 3. For these, m samples result in m dots, each with the feature value on the x-axis and the SHAP value on the y-axis. One explanation results in 168 SHAP values for the 168 predicted time steps, but only the SHAP values for the 24-hour forecast (i.e. the 24th value in the forecast horizon) are shown in the dependence plots. Similarly, there are multiple feature values that could be shown on the x-axis (168 for the past load and 336 for each of the covariates), of which we chose the load of one week before the predicted time step for the load plot, and the future value of the predicted time step for all other plots.

4.2 SHAPformer

Attention manipulation

As the main building block of the Transformer [33], the attention mechanism [48] computes a context-dependent embedding of a query vector given an arbitrary number of equally-dimensioned key vectors as context. More formally, there is a query vector q and key vectors k_1 to k_n . With each key vector k_i there is a value vector u_i associated. The attention score a_i of a key vector k_i is computed as $a_i = \frac{q^T k_i}{\sqrt{d}}$, where d is the vector dimension.

The attention scores a_i to a_n are then soft-maxed by the formula $\alpha_i = \frac{\exp(a_i)}{\sum_{j=1}^n \exp(a_j)}$ in order to compute attention weights α_1 to α_n which sum up to one. The output o of the attention mechanism is finally computed as $o = \sum_{i=1}^n \alpha_i \cdot u_i$.

Attention manipulation [39] allows dropping out individual vectors k_i by setting their attention score to $a_i = -\infty$, so that the attention weight α_i becomes zero after the softmax operation. This means that the weight of vector u_i in the weighted sum is set to zero, and therefore no information from u_i is contained in the attention output and the following layers of the model have no access to u_i .

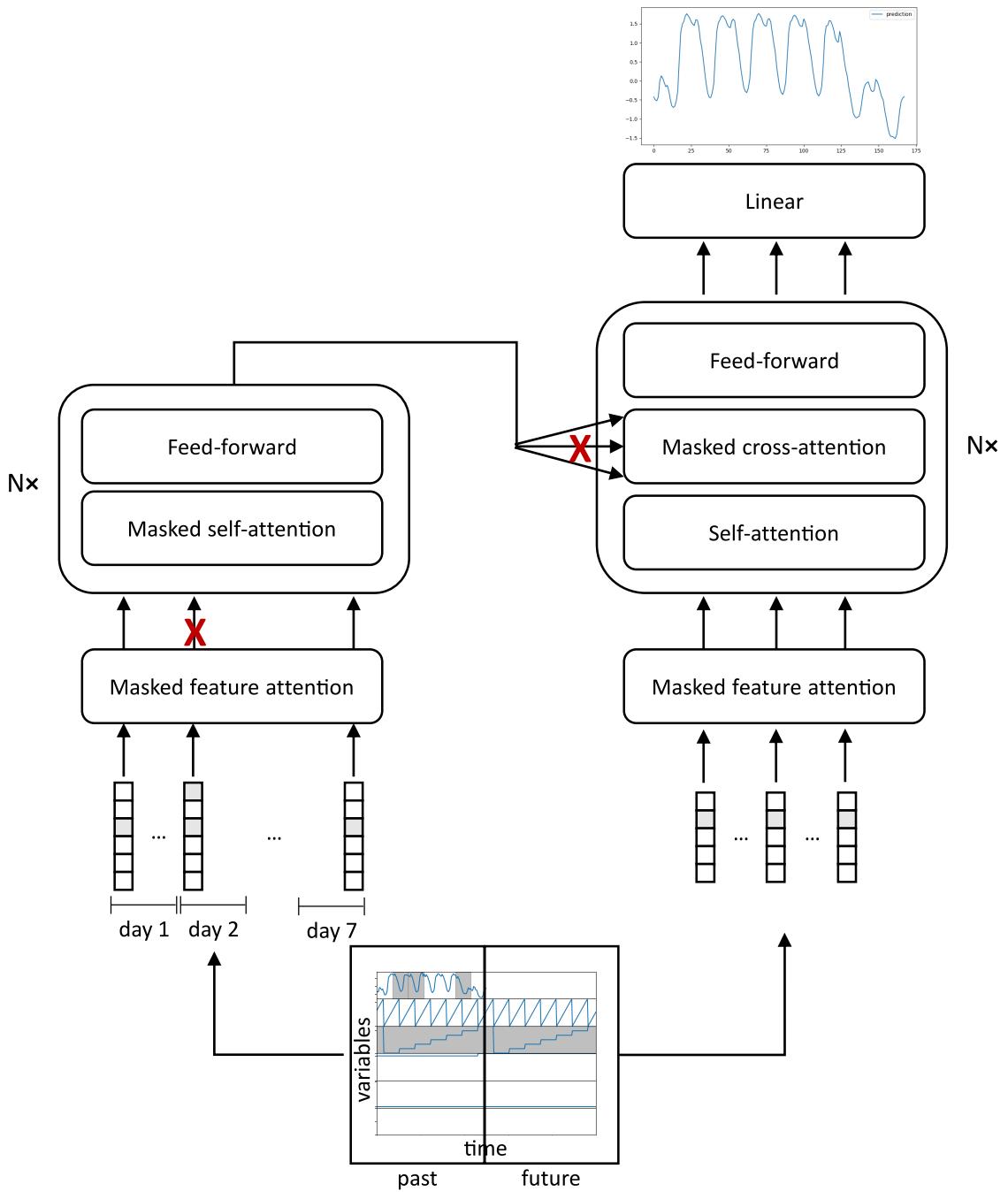
In the following, we make use of attention manipulation to restrict the model's access to feature groups that are absent in a coalition.

SHAPformer architecture

Figure 5A shows the SHAPformer architecture. The model has a lookback size of 168 and a forecast horizon of 168. The 168 past feature vectors are given to the encoder. Each past vector contains the load feature and exogenous features. The 168 future feature vectors are fed to the decoder. Each future vector contains the exogenous features.

SHAPformer uses a masked feature-attention mechanism to compute time step embeddings, as shown in Figure 5B. This mechanism computes an embedding of the feature values belonging to one time step. For each

A SHAPformer architecture



B Masked feature attention

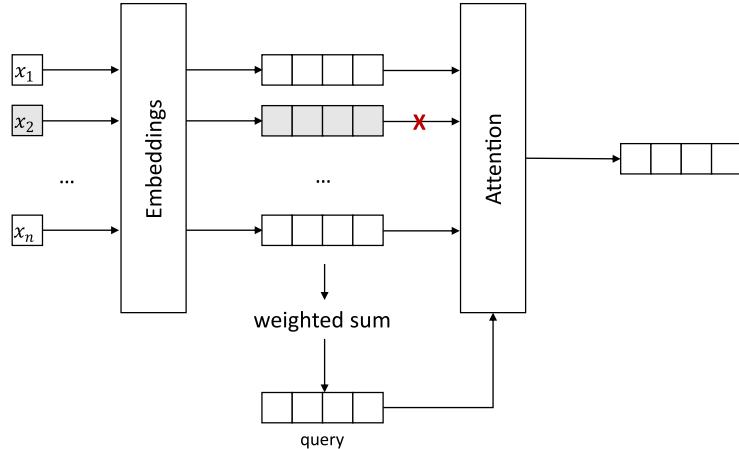


Fig. 5: Overview of the SHAPformer architecture (A) and the masked feature attention mechanism (B). N encoder and decoder blocks are stacked. The masked feature attention is used to drop out variables (v_2 in the example in B). Masked self-attention and masked cross-attention are used to drop out time steps (days 2, 3 and 6 in the example in A).

variable, the model maintains a learned embedding. For categorical variables with k possible values, k randomly initialized embedding vectors are used. For continuous variables, a linear layer is used to embed the feature values. A query vector is computed as a weighted sum of the embedded variables, where dropped-out features get zero weight and all others weight one. Then, attention is used to create the final embedding, based on the query vector and the embedded feature values as key vectors. In the masked feature-attention mechanism, the attention weight of absent features is set to zero, so that the model has no access to their embeddings. With this approach, SHAPformer can be run on subsets of the exogenous features. A positional encoding is added to the embedding vectors to maintain their order.

For the load feature, masked self-attention in the encoder and masked cross-attention in the decoder are used to restrict access to 24 time steps belonging to the same load feature group.

The encoder gets 168 embeddings for the 168 past time steps as input. Multiple Transformer encoder layers are stacked, each consisting of self-attention and linear layers – both equipped with add and norm mechanisms (not shown in the figure). The self-attention is masked, i.e. the attention value for absent time steps is set to zero, so that they are not used to calculate the attention output.

The decoder gets 168 vectors for the 168 future time steps as input. It consists of multiple Transformer decoder layers using self-attention and cross-attention to access the encoder output. The cross-attention is masked, i.e. the attention value for absent time steps is set to zero, so that the decoder has no access to them. Finally, a linear layer is used as prediction head to transform the 168 output vectors of the decoder into 168 scalar values, which are the model prediction for the next 168 time steps.

During model training, random masks are sampled for every example that is given to the model, so that each exogenous feature group and each day has a 50% chance of being masked. Thereby, the model learns to create robust forecasts using subsets of the features. Without masked training, the model could potentially behave arbitrarily when a feature is absent at inference time.

Owen values

Owen values [49] are a variant of SHAP values for games with a coalitional structure. First, the contributions of the coalitions are computed, and then they are broken down further into contributions of the individual players. The formula is slightly different from the formula for SHAP values:

$$Owen(v_i) = \sum_{R \subseteq M \setminus \{k\}} \sum_{T \subseteq B_k \setminus \{v_i\}} \frac{1}{|M| \cdot |B_k|} \frac{1}{\binom{|M|-1}{|R|}} \frac{1}{\binom{|B_k|-1}{|T|}} (f(Q \cup T \cup \{v_i\}) - f(Q \cup T)) \quad (3)$$

with $M = \{B_1, \dots, B_l\}$ being the set of coalitions, B_k the coalition containing v_i , and $Q = \bigcup_{r \in R} B_r$ the union of features over a subset of coalitions R . We use the Owen value formula in SHAPformer in order to treat the past load equally as the exogenous features on the coalition level, and then break the past loads' SHAP value down further into SHAP values of the seven input days.

Note that in our case, when the load of a past day is absent, the other features of that day are also absent, as an effect of the masked self-attention in the encoder and masked cross-attention in the decoder. We assume that this does not affect the Owen values, because the past day's features are used as context information for the past day's load – when the load is not available to the model, the effect of the past day's features on the forecast is negligible.

4.3 Synthetic dataset

Data generation

In order to verify the explanations from SHAPformer, we create a large synthetic dataset for time-series forecasting. Since the data generation process is known, it is possible to judge the explanations returned from SHAPformer and other XAI methods.

The dataset consists of 120 000 examples. Each example contains two weeks of hourly values from the target variable and multiple covariates. The first week of the target variable is used as input to the model, and the model has to predict the second week of the target variable. We call the target variable "load", as in our real dataset. For the covariates, both weeks are used as input to the model. That is, we assume a perfect forecast of the covariates for the next week.

The generation of an example follows a multi-step procedure:

- A month $\in [1, 12]$, start weekday $\in [0, 6]$ and start hour $\in [0, 23]$ are sampled with uniform probability.

- A base load is sampled uniformly in the range $(-0.5, 0.5)$.
- The month has an additive effect on the base load. A value of $0.1 \cdot \sin(\text{month} \cdot 2\pi/12)$ gets added to the base load.
- A daily load curve of 24 values is generated as $\text{factor} \cdot (\sin(h - 0.5 \cdot \pi) + \alpha \cdot \sin(h \cdot 2\pi/24) + \beta \cdot \cos(h \cdot 2\pi/24))$, with $\text{factor} \in (0.5, 1)$, $\alpha, \beta \in (-0.5, 0.5)$ uniformly sampled.
- A uniformly sampled pattern of 24 values with mean 0 and standard deviation 0.1 is added to the daily load. We call the result the day_pattern and use it as the load curve for workdays.
- The Saturday pattern deviates from the workday pattern by a multiplicative factor and an additive deviation. It is generated as $s_1 \cdot \text{day_pattern} + \text{normal}(0, 0.1)$, with $s_1 \in (0.5, 0.9)$ uniformly sampled, and $\text{normal}(0, 0.1)$ generating 24 values from a normal distribution with mean 0 and standard deviation 0.1.
- The Sunday pattern is $s_2 \cdot \text{day_pattern} + \text{normal}(0, 0.1)$, with $s_2 \in (0.2, s_1)$.
- From the workday, Saturday and Sunday patterns, the 336 hourly values are produced by repeating the daily pattern, Saturday pattern and Sunday pattern, beginning from the start day of week and start hour of day. Each day has a 10% chance of being a holiday. On holidays, the Sunday pattern is always used, independent of the weekday.
- A temperature curve is generated by a random walk starting from a uniformly sampled value in $(-0.5, 0.5)$ and each step sampled from a normal distribution with mean 0 and standard deviation 0.02. The temperature has a multiplicative effect on the load. That is, the load is rescaled by the temperature as $\text{load} \cdot (0.5 + 0.5 \cdot \text{temperature})$.
- Finally, random noise is added to the load time series, sampled from a normal distribution with mean 0 and standard deviation 0.05.

The following covariates are used as inputs for the forecasting models: hour of day, day of week, month (all categorical), holiday (binary), temperature and two uncorrelated noise features (all continuous).

100 000 examples are used for training, 10 000 for validation and 10 000 for test. Two examples are shown in Figure C1A.

Ground truth explanations

In order to validate the explanations from the different XAI methods presented in Section 2.2, we calculate ground truth SHAP values for the synthetic dataset. This is possible because we have access to the data generation procedure and can therefore explain the true data dependencies with SHAP. As SHAP is model agnostic, it can be used to explain an arbitrary function $f(x)$. Usually, in the context of XAI, f is a machine learning model, but we set f to the data generation process in order to calculate ground truth explanations. The inputs x to the data generation process consist of the past load, hour of day, day of week, month, holiday, multiplier and two noise features, each represented as a time series of 168 past and – for all inputs except for the load – 168 future values, and the output is the target load of the next week.

We compute ground truth explanations for the examples in the test set of the synthetic dataset. To do so, we use the SHAP Permutation Explainer on the data generation process. For a given test example and a permutation, the Permutation Explainer computes marginal contributions of the inputs based on subsets of the inputs. The inputs that are absent, i.e. not contained in the subset, are resampled 1000 times following the sampling process described above, and an alternative target load curve is generated using the data generation process and the resampled inputs. These 1000 alternative targets are then averaged, and the averaged values are used as the expected target load curve given the subset of the inputs. From the target load curves generated with different input subsets, marginal contributions and SHAP values are computed with the procedure described in Section 4.1.

While sampling alternative inputs, we make sure that no unrealistic combinations of inputs are generated. In particular, when an input x_1 depends on an input x_2 that is contained in the active subset, we do not resample x_1 . The following dependencies are respected: If the day of the week is in the active subset, the hour of day is not resampled (as it can be inferred from the day beginnings and endings). If the holiday feature is in the active subset and there is at least one holiday in the example, the hour of day is not resampled (for the same reason). If the load of the past week is in the active subset, all calendric information as well as the multiplier of the last week is not resampled (because they affect the load), as well as the load patterns for the workday, Saturday and Sunday (as changing any of these would affect the past load).

Note that the ground truth SHAP values are meant to be true to the data [44] under the assumptions stated above. However, a machine learning model could learn different patterns, so that explanations being true to the model instead of true to the data would deviate from our ground truth.

4.4 Baselines

Persistence baseline

The persistence baseline predicts the value from one week before the predicted time step. It thereby respects daily and weekly seasonalities, but no exogenous dependencies. In its simplicity, it is inherently interpretable.

Linear Regression

The linear regression model is based on the following features: the load from 168 time steps before the predicted time step, the hour of day (sine and cosine encoded), the day of week (sine and cosine encoded), the month (sine and cosine encoded), whether it is a holiday (binary) and the temperature and precipitation values. As the model is linear, an explanation can be derived from the model coefficients.

XGBoost Regressor

The Extreme Gradient Boosting Regressor (XGBoost) uses the same features as the linear regression. The default hyperparameters, 100 estimators and a maximum depth of three, are used. The model is able to return feature importance values, but no information on how the features impact the prediction is available.

Time-series Transformer

The time-series Transformer is the same model architecture as SHAPformer, but the model is trained without masking, so that the model always receives the full information about the last week and exogenous features for the next week. The Permutation Explainer and Custom Masker are used with the time-series Transformer to generate explanations.

Temporal Fusion Transformer

The Temporal Fusion Transformer [38] is an encoder-decoder model based on an LSTM encoder and an LSTM decoder [50], followed by a single Transformer [33] layer. It uses causal masking in the Transformer layer to prevent the model from using information of time steps after the prediction time step. We use the same features as for SHAPformer for both the encoder and the decoder, and no static covariates. Crucially, the model’s first layer is a variable selection network, which computes two sets of variable weights – one for the encoder and one for the decoder – that can be interpreted as feature importance values.

Supplementary information. For supplementary information on the data and methods, as well as detailed results from the comparison methods, please refer to the Appendix.

Acknowledgements. During the preparation of this work, the authors used ChatGPT 4o to improve the clarity and fluency of the written text. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Declarations

Funding

The authors gratefully acknowledge funding by the Helmholtz Association under the program “Energy System Design” and the Helmholtz Association’s Initiative and Networking Fund through Helmholtz AI.

Data availability

The real-world load data is publicly available and downloadable via open-power-system-data.org [40]. The exogenous weather data is publicly available and downloadable via cds.climate.copernicus.eu [41]. The synthetic dataset with ground truth explanations is available via GitHub under <https://github.com/KIT-IAI/SHAPformer>.

Code availability

SHAPformer is available as a Python package via GitHub under <https://github.com/KIT-IAI/SHAPformer>.

Author contribution

M.H.: Conceptualization, Methodology, Software, Investigation, Writing - original draft, Visualization; **S.P.:** Methodology, Writing - review & editing; **R.M.:** Conceptualization, Supervision, Funding acquisition, Writing - review & editing; **V.H.:** Supervision, Funding acquisition, Writing - review & editing; **B.S.:** Conceptualization, Funding acquisition, Writing - review & editing.

References

- [1] Petropoulos F, Apiletti D, Assimakopoulos V, Babai MZ, Barrow DK, Ben Taieb S, et al. Forecasting: theory and practice. *International Journal of Forecasting*. 2022 Jul;38(3):705–871. <https://doi.org/10.1016/j.ijforecast.2021.11.001>.
- [2] Sweeney C, Bessa RJ, Browell J, Pinson P. The future of forecasting for renewable energy. *WIREs Energy and Environment*. 2020;9(2):e365. <https://doi.org/10.1002/wene.365>.
- [3] Calvin K, Dasgupta D, Krinner G, Mukherji A, Thorne PW, Trisos C, et al. IPCC, 2023: Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, H. Lee and J. Romero (eds.)]. IPCC, Geneva, Switzerland. Intergovernmental Panel on Climate Change (IPCC); 2023. Available from: <https://doi.org/10.59327/IPCC/AR6-9789291691647>.
- [4] Machowski, Jan and Lubosny, Zbigniew and Bialek, Janusz W and Bumby, James R. Power system dynamics: stability and control. John Wiley & Sons; 2020.
- [5] Haben S, Arora S, Giasemidis G, Voss M, Vukadinović Greetham D. Review of low voltage load forecasting: Methods, applications, and recommendations. *Applied Energy*. 2021 Dec;304:117798. <https://doi.org/10.1016/j.apenergy.2021.117798>.
- [6] Lim B, Zohren S. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2021 Feb;379(2194):20200209. Publisher: Royal Society. <https://doi.org/10.1098/rsta.2020.0209>.
- [7] Su L, Zuo X, Li R, Wang X, Zhao H, Huang B. A systematic review for transformer-based long-term series forecasting. *Artificial Intelligence Review*. 2025 Jan;58(3):80. <https://doi.org/10.1007/s10462-024-11044-2>.
- [8] Ali S, Abuhmed T, El-Sappagh S, Muhammad K, Alonso-Moral JM, Confalonieri R, et al. Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. *Information Fusion*. 2023 Nov;99:101805. <https://doi.org/10.1016/j.inffus.2023.101805>.
- [9] Das A, Rad P.: Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey. arXiv. ArXiv:2006.11371 [cs]. Available from: <http://arxiv.org/abs/2006.11371>.
- [10] Panigutti C, Hamon R, Hupont I, Fernandez Llorca D, Fano Yela D, Junklewitz H, et al. The role of explainable AI in the context of the AI Act. In: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency. FAccT '23. New York, NY, USA: Association for Computing Machinery; 2023. p. 1139–1150.
- [11] Lapuschkin S, Wäldchen S, Binder A, Montavon G, Samek W, Müller KR. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*. 2019 Mar;10(1):1096. Publisher: Nature Publishing Group. <https://doi.org/10.1038/s41467-019-08987-4>.
- [12] European Commission.: Artificial Intelligence Act. Available from: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L_202401689#art_13.
- [13] Adadi A, Berrada M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*. 2018;6:52138–52160. <https://doi.org/10.1109/ACCESS.2018.2870052>.

- [14] Lundberg SM, Lee SI. A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems. vol. 30. Canada; 2017. p. 4768 – 4777. Available from: <https://dl.acm.org/doi/proceedings/10.5555/3295222>.
- [15] Machlev R, Heistrene L, Perl M, Levy KY, Belikov J, Mannor S, et al. Explainable Artificial Intelligence (XAI) techniques for energy and power systems: Review, challenges and opportunities. Energy and AI. 2022 Aug;9:100169. <https://doi.org/10.1016/j.egyai.2022.100169>.
- [16] Baur L, Ditschuneit K, Schambach M, Kaymakci C, Wollmann T, Sauer A. Explainability and Interpretability in Electric Load Forecasting Using Machine Learning Techniques – A Review. Energy and AI. 2024 May;16:100358. <https://doi.org/10.1016/j.egyai.2024.100358>.
- [17] Wu K, Gu J, Meng L, Wen H, Ma J. An explainable framework for load forecasting of a regional integrated energy system based on coupled features and multi-task learning. Protection and Control of Modern Power Systems. 2022 Jun;7(1):24. <https://doi.org/10.1186/s41601-022-00245-y>.
- [18] Henriksen E, Halden U, Kuzlu M, Cali U. Electrical Load Forecasting Utilizing an Explainable Artificial Intelligence (XAI) Tool on Norwegian Residential Buildings. In: 2022 International Conference on Smart Energy Systems and Technologies (SEST); 2022. p. 1–6. Available from: <https://ieeexplore.ieee.org/document/9898500>.
- [19] Bolstad DA, Cali U, Kuzlu M, Halden U. Day-ahead Load Forecasting using Explainable Artificial Intelligence. In: 2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT); 2022. p. 1–5. ISSN: 2472-8152. Available from: <https://ieeexplore.ieee.org/document/9817538>.
- [20] Li M, Wang Y. Power load forecasting and interpretable models based on GS_XGBoost and SHAP. Journal of Physics: Conference Series. 2022 Feb;2195(1):012028. Publisher: IOP Publishing. <https://doi.org/10.1088/1742-6596/2195/1/012028>.
- [21] Lee YG, Oh JY, Kim D, Kim G. SHAP Value-Based Feature Importance Analysis for Short-Term Load Forecasting. Journal of Electrical Engineering & Technology. 2023 Jan;18(1):579–588. <https://doi.org/10.1007/s42835-022-01161-9>.
- [22] Han Y, Sha X, Grover-Silva E, Michiardi P. On the impact of socio-economic factors on power load forecasting. In: 2014 IEEE International Conference on Big Data (Big Data); 2014. p. 742–747. Available from: <https://ieeexplore.ieee.org/document/7004299>.
- [23] Wang J, Yu B, Chen X, Dai G, Liu W, et al. An interpretable short-term electrical load forecasting model based on SHapley Additive exPlanations—A case study in Haidian, Beijing. Electric Power Systems Research. 2025 Oct;247:111769. <https://doi.org/10.1016/j.epsr.2025.111769>.
- [24] Lundberg SM, Erion G, Chen H, DeGrave A, Prutkin JM, Nair B, et al. From local explanations to global understanding with explainable AI for trees. Nature Machine Intelligence. 2020 Jan;2(1):56–67. Publisher: Nature Publishing Group. <https://doi.org/10.1038/s42256-019-0138-9>.
- [25] Ribeiro MT, Singh S, Guestrin C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16. New York, NY, USA: Association for Computing Machinery; 2016. p. 1135–1144. Available from: <https://dl.acm.org/doi/10.1145/2939672.2939778>.
- [26] Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In: 2017 IEEE International Conference on Computer Vision (ICCV); 2017. p. 618–626. ISSN: 2380-7504. Available from: <https://ieeexplore.ieee.org/document/8237336>.
- [27] Bach S, Binder A, Montavon G, Klauschen F, Müller KR, Samek W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. PLOS ONE. 2015 Oct;10(7):e0130140. Publisher: Public Library of Science. <https://doi.org/10.1371/journal.pone.0130140>.

- [28] Achtibat R, Hatefi SMV, Dreyer M, Jain A, Wiegand T, Lapuschkin S, et al. AttnLRP: Attention-Aware Layer-Wise Relevance Propagation for Transformers. In: Proceedings of the 41st International Conference on Machine Learning. PMLR; 2024. p. 135–168. ISSN: 2640-3498. Available from: <https://proceedings.mlr.press/v235/achtibat24a.html>.
- [29] Chen H, Covert IC, Lundberg SM, Lee SI. Algorithms to estimate Shapley value feature attributions. *Nature Machine Intelligence*. 2023 Jun;5(6):590–601. Publisher: Nature Publishing Group. <https://doi.org/10.1038/s42256-023-00657-x>.
- [30] Frye C, Mijolla Dd, Begley T, Cowton L, Stanley M, Feige I. Shapley explainability on the data manifold; 2021. Available from: <https://openreview.net/forum?id=OPyWRrcjVQw>.
- [31] Bento J, Saleiro P, Cruz AF, Figueiredo MAT, Bizarro P. TimeSHAP: Explaining Recurrent Models through Sequence Perturbations. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. KDD '21. New York, NY, USA: Association for Computing Machinery; 2021. p. 2565–2573. Available from: <https://doi.org/10.1145/3447548.3467166>.
- [32] Nayebi A, Tipirneni S, Reddy CK, Foreman B, Subbian V. WindowSHAP: An efficient framework for explaining time-series classifiers based on Shapley values. *Journal of Biomedical Informatics*. 2023 Aug;144:104438. <https://doi.org/10.1016/j.jbi.2023.104438>.
- [33] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is All you Need. In: Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc.; 2017. p. 5998–6008.
- [34] Clark K, Khandelwal U, Levy O, Manning CD.: What Does BERT Look At? An Analysis of BERT’s Attention. arXiv. ArXiv:1906.04341. Available from: <http://arxiv.org/abs/1906.04341>.
- [35] Hertel M, Ott S, Schäfer B, Mikut R, Hagenmeyer V, Neumann O. Evaluation of Transformer Architectures for Electrical Load Time-Series Forecasting. In: Proceedings 32. Workshop Computational Intelligence. vol. 1; 2022. p. 93. Available from: https://library.oapen.org/bitstream/handle/20.500.12657/59840/external_content.pdf?sequence=1#page=105.
- [36] Jain S, Wallace BC. Attention is not Explanation. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics; 2019. p. 3543–3556. Available from: <https://aclanthology.org/N19-1357>.
- [37] Wiegreffe S, Pinter Y. Attention is not not Explanation. In: Inui K, Jiang J, Ng V, Wan X, editors. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics; 2019. p. 11–20. Available from: <https://aclanthology.org/D19-1002>.
- [38] Lim B, Arik SO, Loeff N, Pfister T. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*. 2021 Oct;37(4):1748–1764. <https://doi.org/10.1016/j.ijforecast.2021.03.012>.
- [39] Deisereth B, Deb M, Weinbach S, Brack M, Schramowski P, Kersting K. ATMAN: Understanding Transformer Predictions Through Memory Efficient Attention Manipulation. *Advances in Neural Information Processing Systems*. 2023 Dec;36:63437–63460.
- [40] Wiese F, Schlecht I, Bunke WD, Gerbaulet C, Hirth L, Jahn M, et al. Open Power System Data – Frictionless data for electricity system modelling. *Applied Energy*. 2019 Feb;236:401–409. <https://doi.org/10.1016/j.apenergy.2018.11.097>.
- [41] Copernicus Climate Change Service.: Climate and energy indicators for Europe from 1979 to present derived from reanalysis. ECMWF. Available from: <https://cds.climate.copernicus.eu/doi/10.24381/cds.4bd77450>.
- [42] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016. p. 785–794.

ArXiv:1603.02754 [cs]. Available from: <http://arxiv.org/abs/1603.02754>.

- [43] Nauta M, Trienes J, Pathak S, Nguyen E, Peters M, Schmitt Y, et al. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. ACM Comput Surv. 2023 Jul;55(13s):295:1–295:42. <https://doi.org/10.1145/3583558>.
- [44] Chen H, Janizek JD, Lundberg S, Lee SI.: True to the Model or True to the Data? arXiv. ArXiv:2006.16234. Available from: <http://arxiv.org/abs/2006.16234>.
- [45] Shapley, Lloyd S . A Value for n-person Games. Contributions to the Theory of Games Annals of Mathematical Studies Princeton University Press. 1953;28:307–317. <https://doi.org/https://doi.org/10.1515/9781400881970-018>.
- [46] Molnar C. Interpreting machine learning models with SHAP: a guide with Python examples and theory on Shapley values. First edition ed. München, Germany: Christoph Molnar c/o MUCBOOK, Heidi Seibold; 2023.
- [47] Schlegel U, Keim DA. A Deep Dive into Perturbations as Evaluation Technique for Time Series XAI. In: Longo L, editor. Explainable Artificial Intelligence. Cham: Springer Nature Switzerland; 2023. p. 165–180.
- [48] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. In: Bengio Y, LeCun Y, editors. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings; 2015. Available from: <http://arxiv.org/abs/1409.0473>.
- [49] Owen G. Values of Games with a Priori Unions. In: Henn R, Moeschlin O, editors. Mathematical Economics and Game Theory. Berlin, Heidelberg: Springer; 1977. p. 76–88.
- [50] Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Computation. 1997 Nov;9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [51] Biewald L.: Experiment Tracking with Weights and Biases. Available from: <wandb.com>.
- [52] Kingma DP, Ba J.: Adam: A Method for Stochastic Optimization. arXiv. ArXiv:1412.6980 [cs]. Available from: <http://arxiv.org/abs/1412.6980>.
- [53] Loshchilov I, Hutter F.: Decoupled Weight Decay Regularization. arXiv. ArXiv:1711.05101 [cs, math]. Available from: <http://arxiv.org/abs/1711.05101>.

Appendix A Detailed results on the real-world dataset

Detailed results on the empirical dataset from TransnetBW are given in Table A1. Five metrics are evaluated and means and standard deviations from five runs are reported. The baseline, linear regression and XGBoost are deterministic, therefore they have a standard deviation of zero.

Table A1: Forecast errors on the TransnetBW dataset, evaluated with five different metrics. Means and standard deviations over five runs are reported.

Model	MAE (scaled)	MSE (scaled)	MAE [MW]	RMSE [MW]	MAPE [%]
Persistence baseline	0.255 ± 0.000	0.177 ± 0.000	395.5 ± 0.0	652.3 ± 0.0	6.17 ± 0.00
Linear Regression	0.255 ± 0.000	0.128 ± 0.000	395.0 ± 0.0	553.7 ± 0.0	6.19 ± 0.00
XGBoost	0.177 ± 0.000	0.062 ± 0.000	274.2 ± 0.0	387.0 ± 0.0	4.18 ± 0.00
TFT	0.163 ± 0.016	0.065 ± 0.016	251.9 ± 24.3	390.8 ± 49.9	3.94 ± 0.41
Transformer	0.127 ± 0.005	0.028 ± 0.002	197.3 ± 8.3	263.1 ± 9.1	2.98 ± 0.12
SHAPformer	0.131 ± 0.006	0.029 ± 0.002	203.3 ± 8.9	265.9 ± 9.6	3.09 ± 0.13

Appendix B Hyperparameter optimization

The hyperparameters of the Temporal Fusion Transformer, the Transformer and SHAPformer on synthetic data were optimized using Bayesian Optimization and Weights and Biases (wandb) [51]. For SHAPformer on real-world data, the same hyperparameters were used as for the Transformer, but with a lower learning rate, which stabilizes the masked training. The selected hyperparameters are given in Table B2. Adam [52] and AdamW [53] were used as training algorithms, using the mean squared error loss function.

Table B2: The chosen hyperparameters for the different models.

Model	Hyperparameter	Synthetic	Real
Temporal Fusion Transformer	d_model	256	1024
	batch size	4	16
	heads	8	8
	optimizer	AdamW	AdamW
	learning rate	0.00067	0.00017
	decay rate	0.37	0.93
	layers	8	2
	d_model	64	128
	heads	4	2
	optimizer	AdamW	Adam
Transformer	batch size	16	64
	learning rate	0.00015	0.000010
	decay rate	1.00	1.00
	layers	7	2
	d_model	512	128
	heads	2	2
	optimizer	AdamW	Adam
	batch size	16	64
	learning rate	0.00010	0.000001
	decay rate	0.96	1.00
SHAPformer	d_model	512	128
	heads	2	2
	optimizer	AdamW	Adam
	batch size	16	64
	learning rate	0.00010	0.000001
	decay rate	0.96	1.00
	d_model	512	128
	heads	2	2
	optimizer	AdamW	Adam
	batch size	16	64

Appendix C Local SHAPformer explanations on synthetic data

Local explanations of two synthetic examples are shown in Figure C1. On the left-hand side, the forecast horizon starts on a Friday. The hour of day, day of week, and holiday features influence the prediction in a pattern resembling the base load curve: a half-sine shape with higher loads during the day and lower loads at night. In this case, Sunday is also a holiday, so the holiday and day-of-week effects are both small compared to typical weekdays, resulting in a lower overall load. The previous Sunday (input day 3) exerts a negative influence on the predicted Sunday, as its particularly low load leads to a reduced forecast for the following week. The final day of the forecast horizon is also a holiday, where the holiday feature compresses the load curve—reducing positive daytime loads and amplifying negative nighttime loads. The ground truth looks similar to the SHAPformer explanation. Only the effect of the seven days is summarized, so the effect of input day 3 on the Sunday is not visible. On the right-hand side, the prediction starts on a Friday evening and the load is affected by a temperature increase. This is visible in the explanation of the forecast, where the temperature SHAP values increase in magnitude over time. In comparison to the ground truth, SHAPformer underestimates the month effect, as observable through the lower amplitudes in the explanation than in the ground truth for these two features.

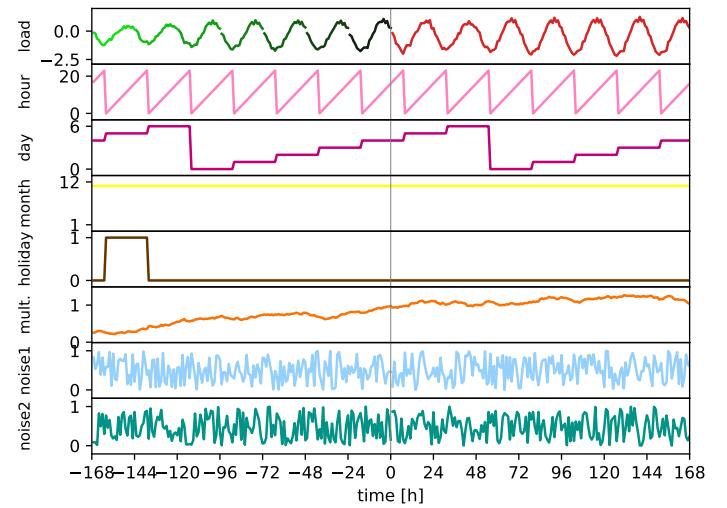
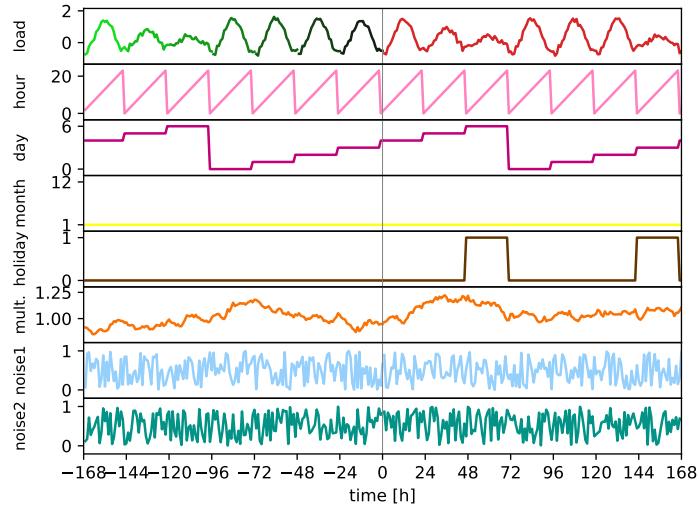
Appendix D Transformer explained with SHAP

This appendix shows the global and local explanations for the Transformer model created with the Permutation Explainer and the Custom Masker on the two datasets.

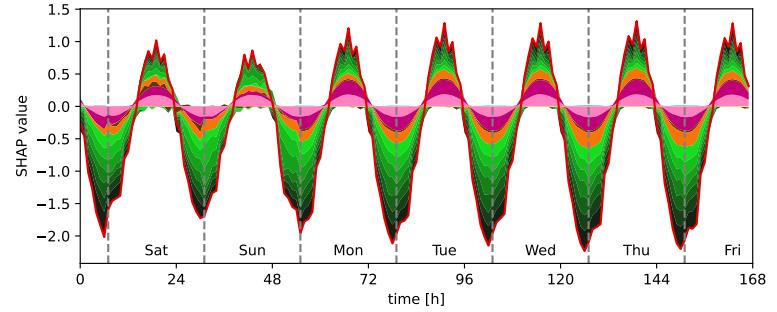
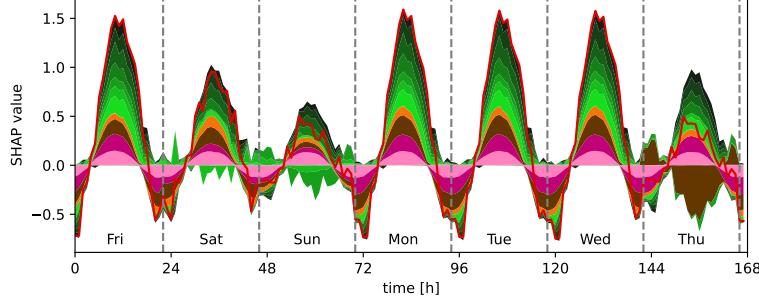
D.1 Synthetic data

See Figure D2 for the global explanations and Figure D3 for the local explanations on the synthetic data.

A Synthetic examples



B SHAPformer explanations



C Ground truth

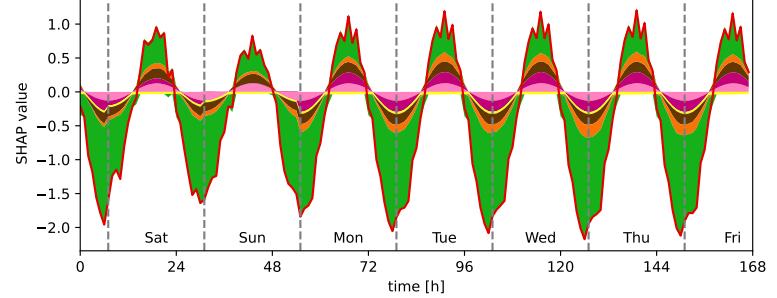
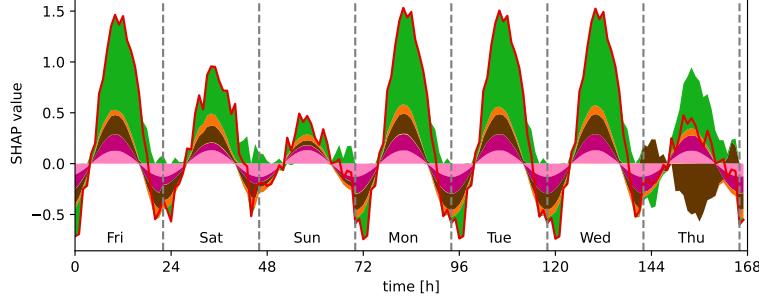
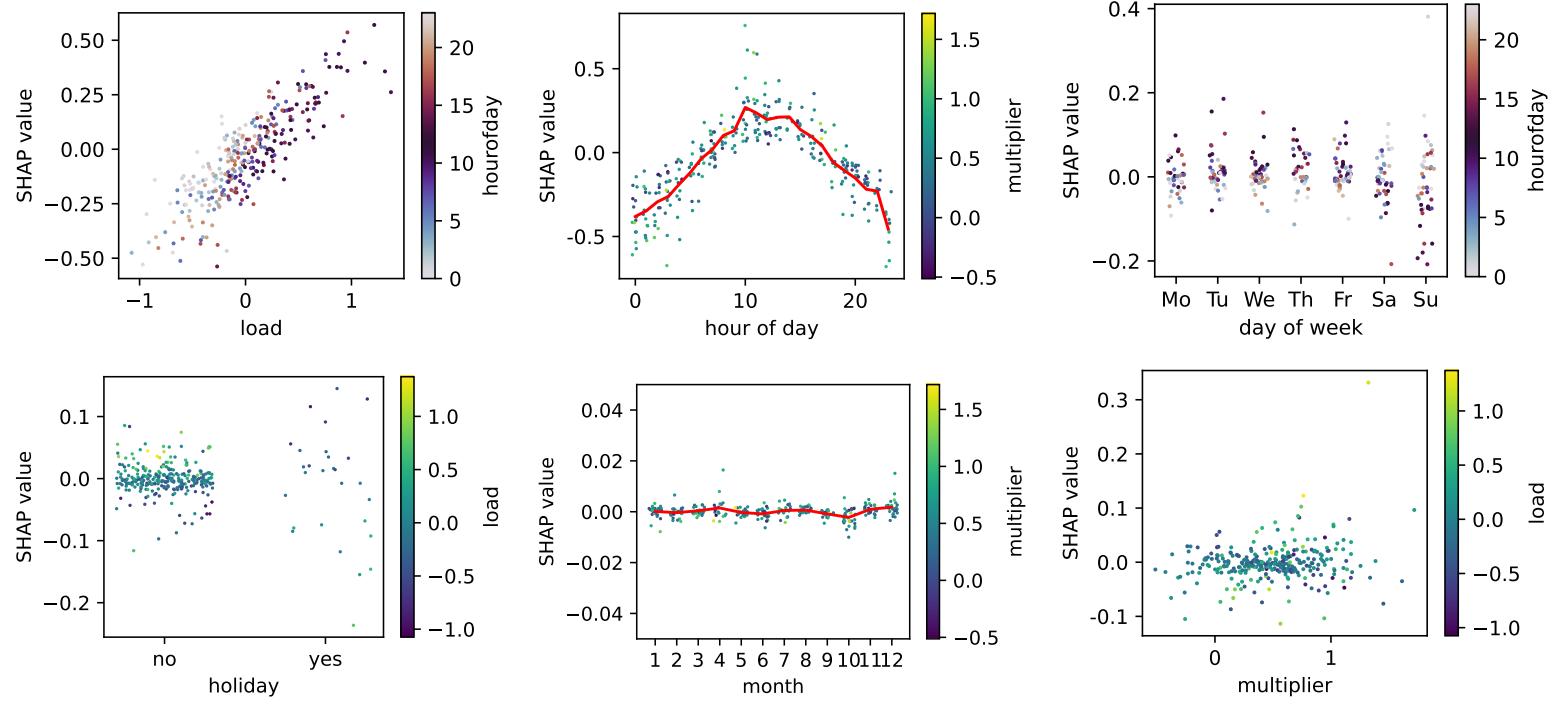


Fig. C1: Local explanations of two different synthetic examples (left and right). Left: the holiday feature affects the last predicted day. Right: The increasing multiplier has an increasing effect.

D.2 TransnetBW data

See Figure D4 for the global explanations and Figure D5 for the local explanations on the TransnetBW data.

A Permutation Explainer



B Custom Masker

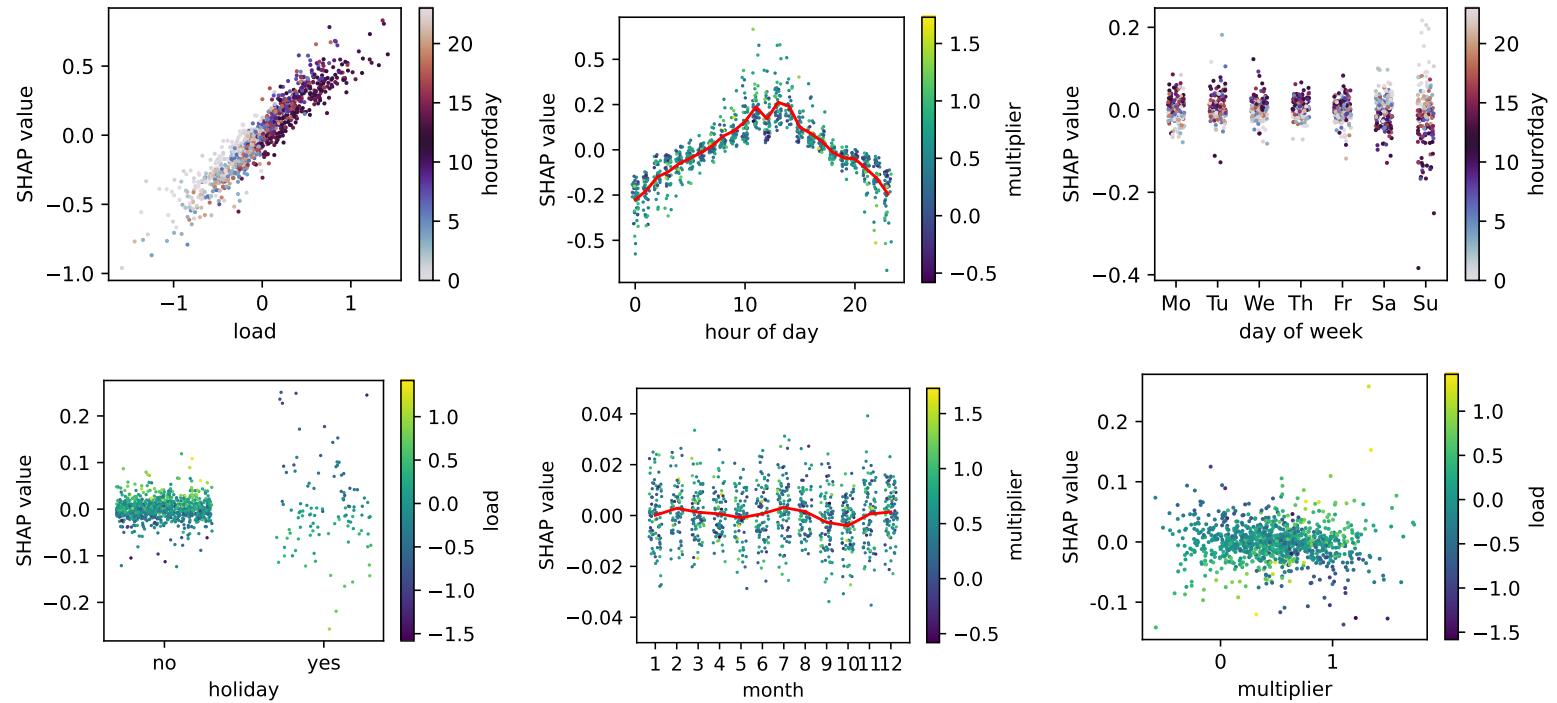
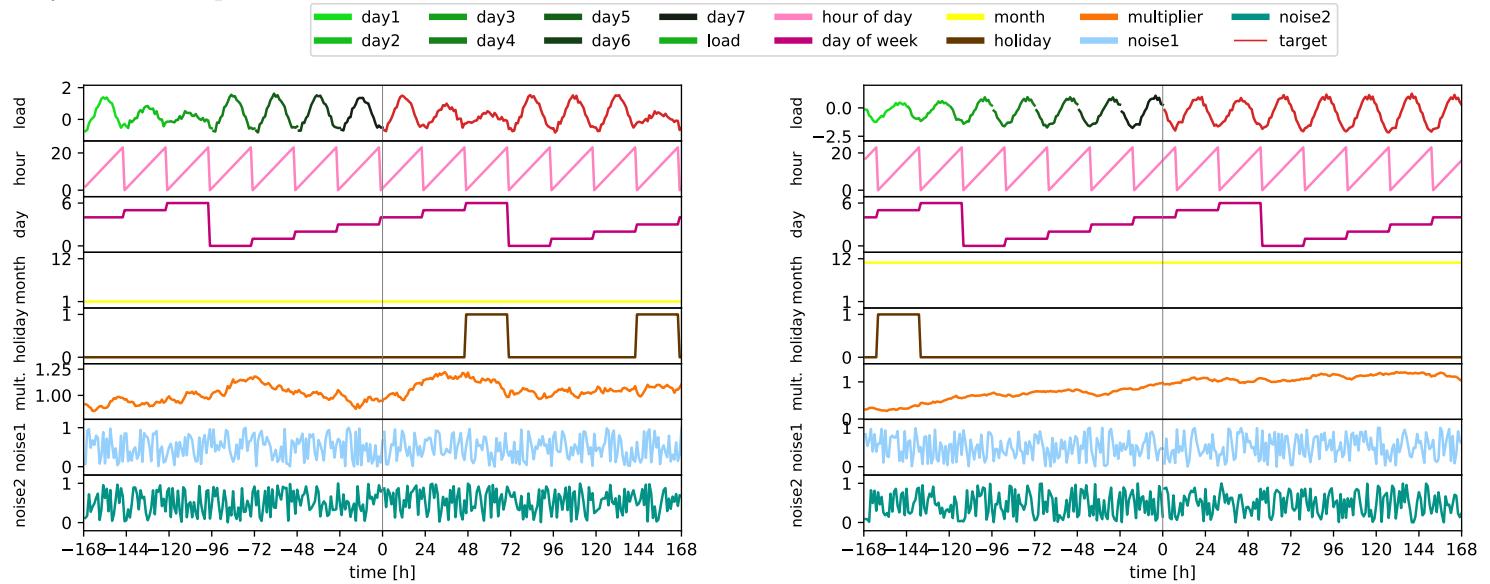
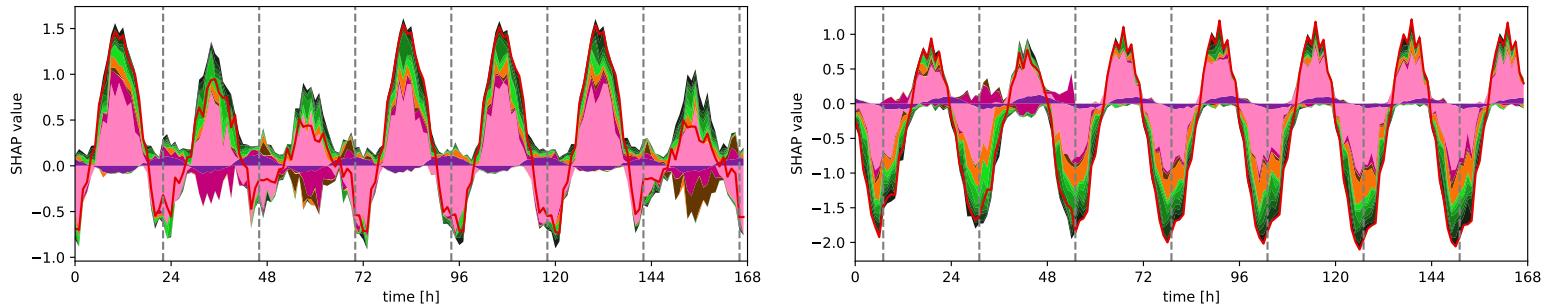


Fig. D2: Dependence plots on synthetic data created with the Permutation Explainer and the Custom Masker. For discrete variables, noise was added in the x-direction for visibility reasons.

A Synthetic examples



B Explanations from Permutation Explainer



C Explanations from Custom Masker

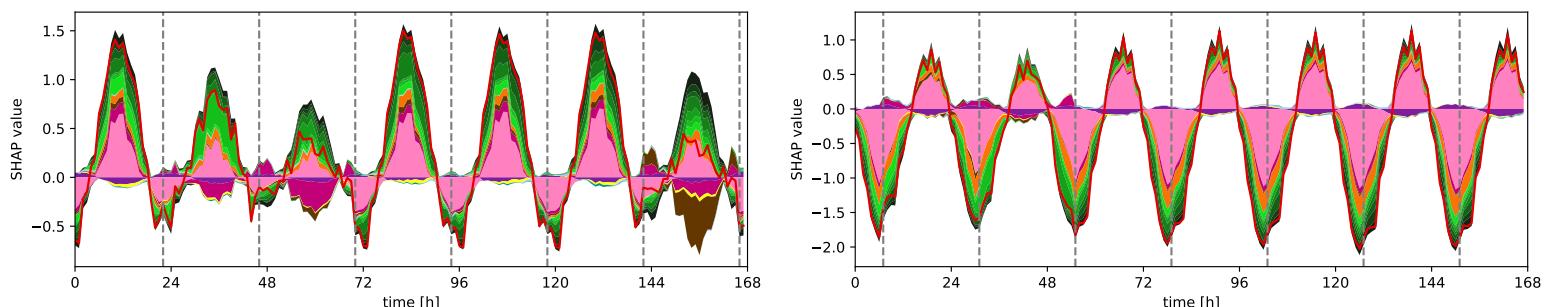
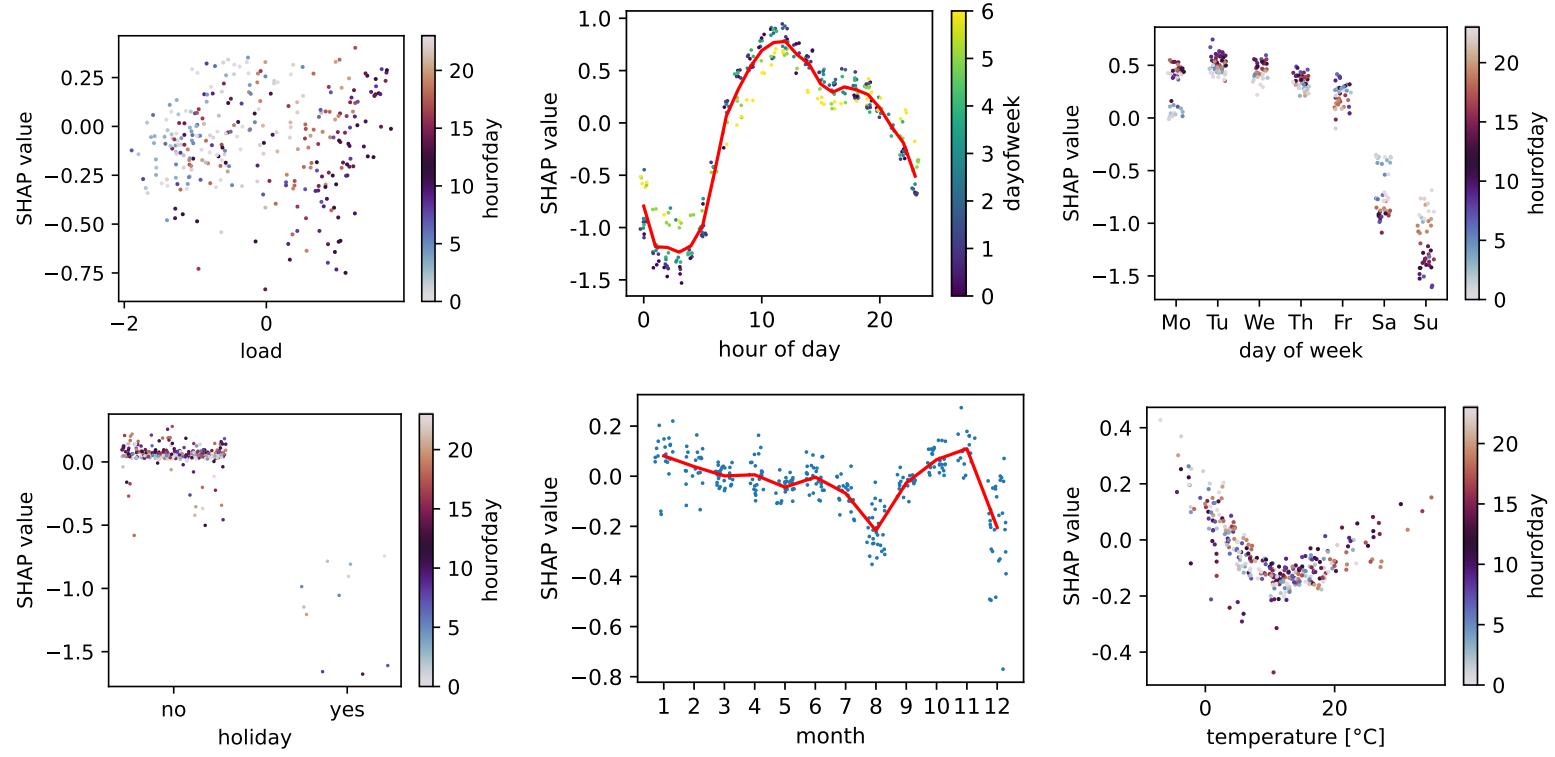


Fig. D3: Local explanations of synthetic examples created with the Permutation Explainer and the Custom Masker.

A Permutation Explainer



B Custom Masker

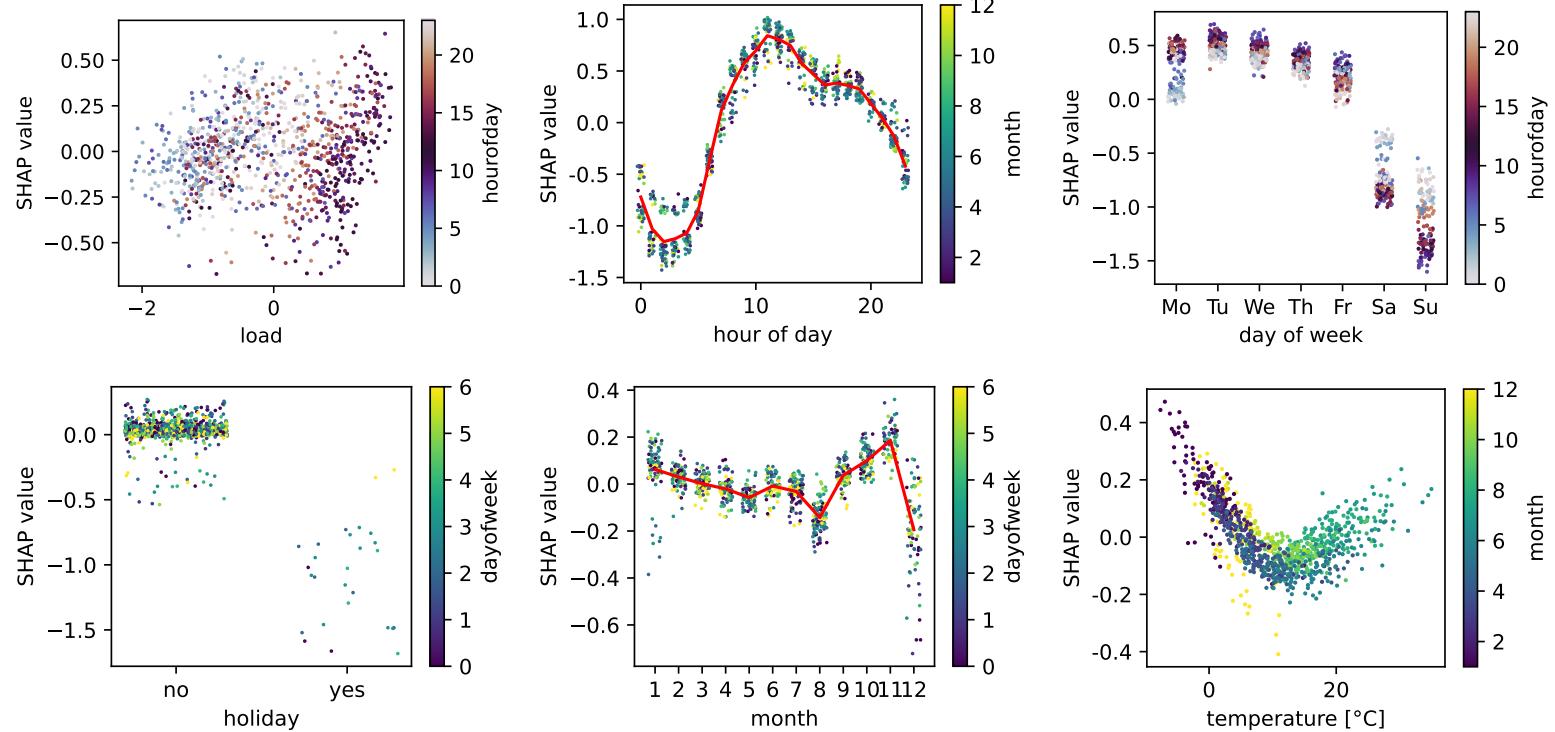


Fig. D4: Dependence plots on TransnetBW data created with the Permutation Explainer and the Custom Masker. For discrete variables, noise was added in the x-direction for visibility reasons.

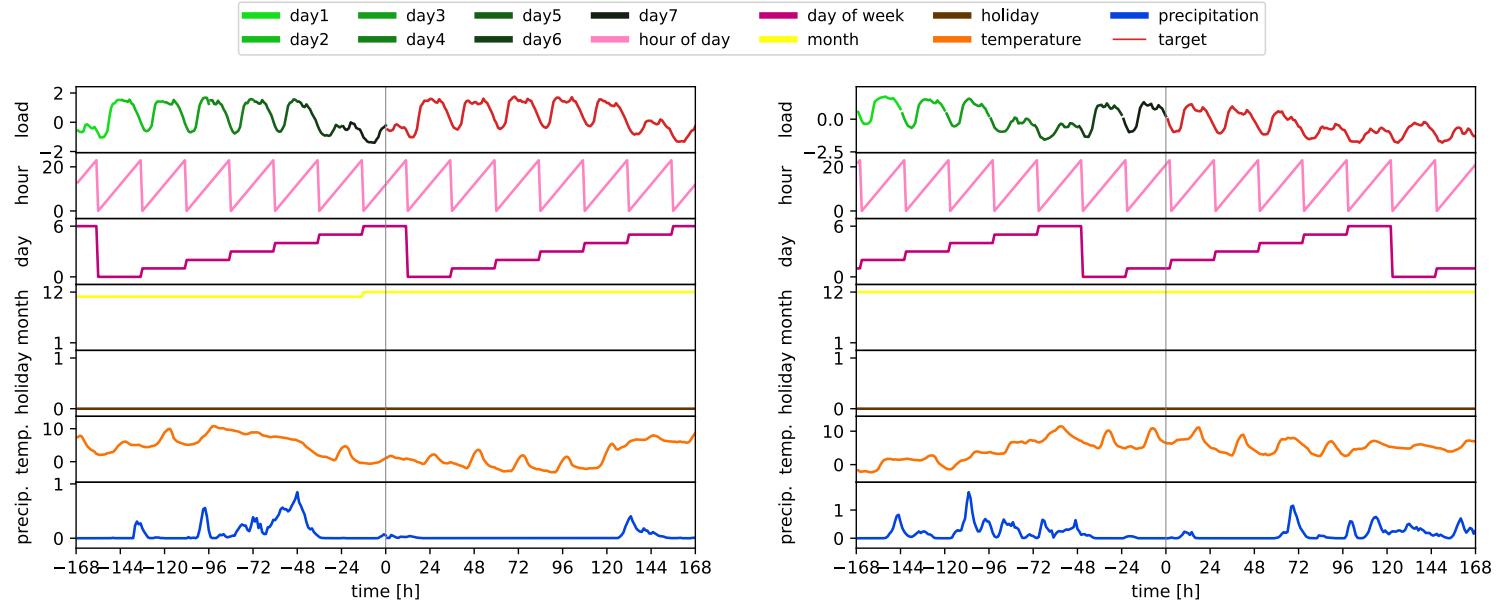
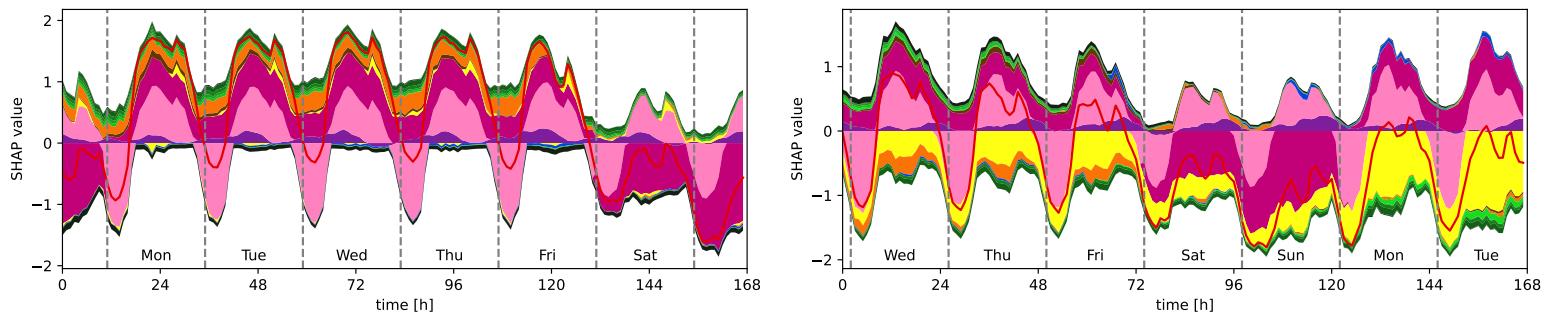
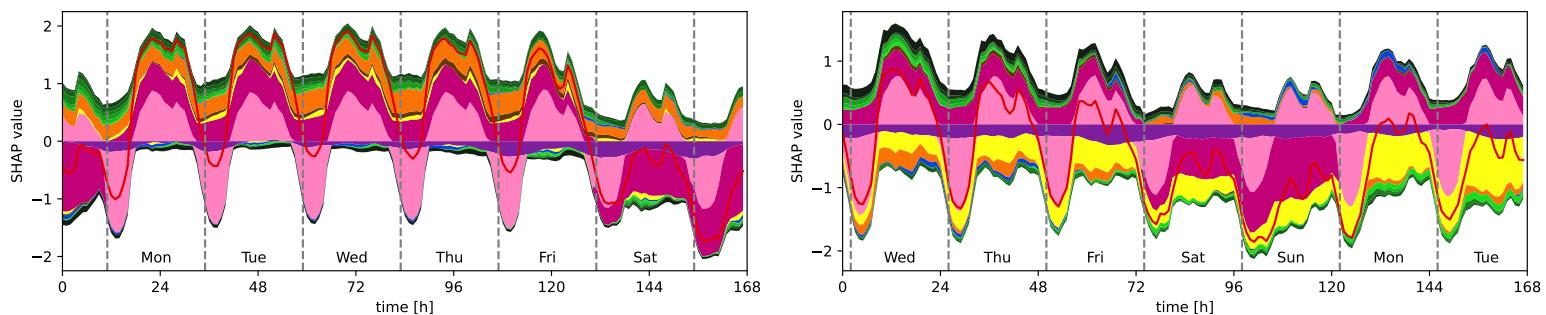
A Data**B Permutation Explainer****C Custom Masker**

Fig. D5: Local explanations of TransnetBW examples created with the Permutation Explainer and the Custom Masker.