PatchDecomp: Interpretable Patch-Based Time Series Forecasting

Hiroki Tomioka, Genta Yoshimura

Mitsubishi Electric Corporation, Japan
Hiroki.Tomioka@ay.MitsubishiElectric.co.jp

Abstract

Time series forecasting, which predicts future values from past observations, plays a central role in many domains and has driven the development of highly accurate neural network models. However, the complexity of these models often limits human understanding of the rationale behind their predictions. We propose PatchDecomp, a neural network-based time series forecasting method that achieves both high accuracy and interpretability. PatchDecomp divides input time series into subsequences (patches) and generates predictions by aggregating the contributions of each patch. This enables clear attribution of each patch, including those from exogenous variables, to the final prediction. Experiments on multiple benchmark datasets demonstrate that PatchDecomp provides predictive performance comparable to recent forecasting methods. Furthermore, we show that the model's explanations not only influence predicted values quantitatively but also offer qualitative interpretability through visualization of patch-wise contributions.

Introduction

Time series data, which record values that change over time, are crucial across various domains, including manufacturing, logistics, and healthcare. The versatility of time series forecasting (TSF) enables its applications in various domains, such as the prediction electricity consumption, traffic flow, product sales, and inventory levels. With the advancement of machine learning technologies, numerous forecasting methods have been proposed, ranging from relatively simple linear transformation-based methods to more complex approaches that utilize recurrent neural networks (RNNs), multilayer perceptrons (MLPs), and Transformers (Vaswani et al. 2017). In recent years, there has been a growing trend of incorporating patching techniques that treat input data as subsequences rather than in a point-wise manner, thereby significantly enhancing the performance of TSF. Additionally, some forecasting frameworks utilize not only the time series history of the target variable but also other exogenous variables (covariates) for predictions. By incorporating information on external factors that cannot be captured using historical target data alone, these approaches promise further improvements in accuracy, leading to the expansion of these models to real-world problems.

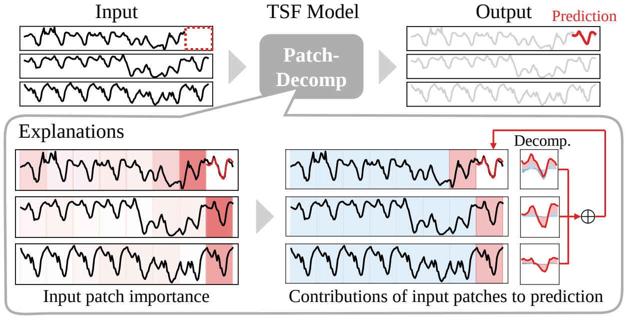


Figure 1: TSF framework and explanations from PatchDecomp. PatchDecomp performs predictions by dividing input time series into patches, providing users with explanations as the input patch importance (the intensity of red in the lower left figure) and the contributions of input patches to the prediction (the area charts in the lower right). For clarity, only the patches with high importance are depicted in red for each variable; others are in blue.

Advancements in studies based on MLPs and Transformers, along with the dramatic increase in computational power, have rapidly improved forecasting accuracy. However, neural networks are generally regarded as complex black-box models, making it difficult to understand their internal behaviors. This lack of interpretability is undesirable in practical applications. In real-world settings such as manufacturing systems, users are unlikely to trust predictive models that merely produce outputs without any interpretable rationale. Deploying such opaque models in domains where transparency and safety are critical entails substantial risks. Users can assess the validity of these forecasts if the reasons behind the predictions are presented. This topic is called interpretability and is being actively studied in the field of explainable artificial intelligence (XAI). For neural network-based TSF methods, enhancing interpretability to mitigate risks and promoting applicability to real-world problems is a significant challenge. Some existing methods attempt to improve interpretability by visualizing linear weight matrices (Zeng et al. 2023), decomposing predicted values (Oreshkin et al. 2019; Olivares et al. 2023; Challu et al. 2023), and presenting variable importance and attention matrix weights over time (Lim et al. 2021). However, these approaches fail to explain the contributions of each variable's subsequence, including exogenous variables, to the predicted values, rendering their interpretability insufficient for practical use.

In this paper, we propose PatchDecomp, a neural network-based TSF method with interpretability. PatchDecomp handles the input variables, including exogenous variables, by dividing them into subsequences (patches) and decomposing the predicted values according to the contribution of each input patch (Figure 1). By performing contribution decomposition based on the entire processing of the model from input to output, this model is capable of rigorously calculating the correspondence between inputs and outputs, rather than being limited to partial interpretability through the visualization of the attention map. This model is designed to improve interpretability, and achieving state-of-the-art performance is not the focus of this study. However, as a result of extensive experiments, we confirmed the competitive predictive accuracy of PatchDecomp. The main contributions are summarized as follows:

- We propose an interpretable patch-based TSF model called PatchDecomp. This model can deal with exogenous variables in addition to the target and explain the contribution of each variable's patch to the prediction.
- We conduct TSF tasks on multiple datasets and demonstrate that the proposed method is comparable to recent forecasting methods in terms of its predictive accuracy.
- By conducting both qualitative and quantitative analyses, we demonstrate that the proposed method achieves superior interpretability in TSF.

Related Work

Time Series Forecasting

Although various TSF methods have been studied for a long time, deep learning models have become mainstream. Recently, models based on MLPs and Transformers have largely replaced RNN-based models, such as DeepAR (Salinas et al. 2020), for TSF trends. N-BEATS (Oreshkin et al. 2019) is an MLP-based method that stacks predicted trend and seasonal components. NBEATSx (Olivares et al. 2023) was developed subsequently to handle exogenous variables, along with NHITS (Challu et al. 2023), which is hierarchical and capable of processing multiple frequencies. TSMixer (Chen et al. 2023), which applies MLP-Mixer, and TiDE (Das et al. 2023), which features an encoder–decoder structure using MLPs, can handle exogenous variables. Moreover, Transformer fully utilizes attention mechanisms, and the development of models incorporating these architectures, such as Informer (Zhou et al. 2021), Autoformer (Wu et al. 2021), and FEDformer (Zhou et al. 2022), is one of the most popular directions. Temporal Fusion Transformer (TFT) (Lim et al. 2021) consists of an LSTM encoder and a multi-head attention decoder and utilizes exogenous variables to predict future time series. While conventional Transformer-based methods tokenize data by time steps and apply attention, iTransformer (Liu et al. 2023) represents a paradigm shift by tokenizing data by variables instead of time.

Furthermore, a noteworthy idea is the patching technique introduced by PatchTST (Nie et al. 2022), which enables the model to capture local time series information that cannot be grasped point-wise by splitting the input time series into patches before feeding them into the Transformer encoder. The patching technique has had a significant influence on subsequent research and has become essential for enhancing forecasting models. TimeXer (Wang et al. 2024) also adopts this technique and comprises patch-wise self-attention and cross-attention with exogenous variables.

The proposed method herein adopts the attention mechanism and also employs a patching technique to forecast future values based on the patches of the input variables.

Interpretability of TSF Models

In the modern era, where AI technologies are spreading throughout society, understanding the behavior of AI models is critically important and is being actively studied across a wide range of areas within machine learning. Several researchers are working on time series tasks (Zhao et al. 2023; Ozyegen, Ilic, and Cevik 2022; Arsenault, Wang, and Patenaude 2025); however, much of this work remains limited to time series classification problems.

Studies on the interpretability of time series tasks can be broadly divided into two categories: post-hoc interpretation methods and inherently interpretable models (Zhao et al. 2023).

Post-hoc interpretation methods extract interpretable information by applying it to pretrained models. LIME (Ribeiro, Singh, and Guestrin 2016) and SHAP (Lundberg and Lee 2017) fall into this category. Although these methods can be applied independently to the predictive model, they have the drawbacks of relatively high computational costs and difficulty providing precise explanations of the model behavior because they perform an approximation.

On the other hand, inherently interpretable models have build-in mechanisms to present the rationale behind their predictions. While it is necessary to incorporate explanatory mechanisms into the model, explanations that are more understandable to users (compared with post-hoc methods) can be provided with careful design. For example, simple models, such as DLinear and NLinear (Zeng et al. 2023) possess interpretability in their linear weight matrices. By visualizing these weights, users can understand the variables that strongly contribute to predictions. In addition, N-BEATS, NBEATSx, and NHITS, can decompose the predicted time series into trend and seasonal components, allowing us to understand which variables or components contribute to the forecasted values. TFT enhances interpretability by calculating the attention weights for each time step and the feature importance for each variable. Transformer-based methods often regard attention weights as a foundation for interpretability, as visualizing these weights provides insights into which inputs are more focused by the model (Schockaert, Leperlier, and Moawad 2020; Gangopadhyay et al.

2021; Liu et al. 2023). However, in the context of natural language processing, while attention weights provide interpretation to some extent, they do not possess sufficient capability to fully explain model behavior (Serrano and Smith 2019; Jain and Wallace 2019), and the effectiveness of attention weights in interpreting TSF models remains a controversial topic.

In all the aforementioned models, the extent to which each segment of the variables, including exogenous variables, contributes to the predicted values remains unclear. In this paper, we propose an inherently interpretable model that decomposes the contributions of input patches to predicted values. Our approach does not simply visualize attention weights; rather, it establishes an apparent correspondence between input and output data by considering the entire processing of the architecture that includes the attention mechanism.

Methodology

In this study, we address the problem of predicting future time series values over H time steps, based on the observed time series history over L time steps. The current time series history of the target variable $\boldsymbol{y}$ is denoted as $\boldsymbol{y}_{:L} = \{y_1, \cdots, y_L\}$, whereas the future time series we aim to predict is represented as $\boldsymbol{y}_{-H:} = \{y_{L+1}, \cdots, y_{L+H}\}$.

We denote D_{hist} exogenous variables that can be observed from the current time series history, along with the target variable $\boldsymbol{y}_{:L}$, as $\boldsymbol{x}^{\text{hist}} \in \mathbb{R}^{(D_{\text{hist}}+1)\times L}$. We denote D_{futr} exogenous variables, such as weather forecasts and calendar information, for which the time series is known in advance up to the future time steps, as $\boldsymbol{x}^{\text{futr}} \in \mathbb{R}^{D_{\text{futr}}\times(L+H)}$. In addition, we describe static variables that do not change over time, such as product IDs and categories, as $\boldsymbol{x}^{\text{stat}} \in \mathbb{R}^{D_{\text{stat}}}$.

The objective of TSF is to find a model $\mathcal{F}$ that can accurately predict the future time series $\boldsymbol{y}_{-H:}$ using exogenous variables as inputs, as follows:

$$\hat{\boldsymbol{y}}_{-H:} = \mathcal{F}(\boldsymbol{x}^{\text{hist}}, \boldsymbol{x}^{\text{futr}}, \boldsymbol{x}^{\text{stat}}) \in \mathbb{R}^H, \tag{1}$$

where $\hat{\boldsymbol{y}}_{-H:}$ represents the predicted future values over H time steps.

As shown in Figure 2, PatchDecomp consists of two components: a patch encoder, which divides the time series into patches for each variable and encodes them into latent vectors, and a patch decoder, which associates the latent vectors of the input and output patches and decodes them into predicted values.

Patch Encoder

First, the time series is standardized using reversible instance normalization (RevIN) (Kim et al. 2021) for each variable and then divided into patches of length P, starting from the current time. If the past length L or future length H is not divisible by the patch length P, zero padding is applied at the beginning or end. Consequently, the numbers of past and future patches are given by $N_{\text{hist}} = \lceil \frac{L}{P} \rceil$ and $N_{\text{futr}} = \lceil \frac{H}{P} \rceil$, respectively.

Next, for the i-th variable's t-th patch $\boldsymbol{x}_{i,t}^{\text{patch}} \in \mathbb{R}^P$, a linear transformation is applied to obtain $\boldsymbol{z}_{i,t}^{\text{patch}} \in \mathbb{R}^D$. Temporal information is encoded via positional encoding through linear transformation or embedding, resulting in $\boldsymbol{z}_t^{\text{pos}} \in \mathbb{R}^{(N_{\text{hist}}+N_{\text{futr}})\times D}$. Additionally, to embed the information from the static variables $\boldsymbol{x}^{\text{stat}} \in \mathbb{R}^{D_{\text{stat}}}$, a linear transformation or embedding is applied to obtain $\boldsymbol{z}^{\text{stat}} \in \mathbb{R}^D$, which is also added.

Subsequently, by applying a residual block that combines the MLP and skip connections N_{enc} times, each patch is encoded into a D-dimensional latent vector while considering the temporal positional information and static variables:

$$\boldsymbol{z}_{i,t}^{\text{src}} = \text{Encode}(\boldsymbol{z}_{i,t}^{\text{patch}} + \boldsymbol{z}_t^{\text{pos}} + \boldsymbol{z}^{\text{stat}}) \in \mathbb{R}^D. \tag{2}$$

The resulting $\boldsymbol{z}^{\text{src}} \in \mathbb{R}^{N_{\text{patch}}\times D}$, which is formed by stacking the vectors $\boldsymbol{z}_{i,t}^{\text{src}}$ for all patches, corresponds to the latent representation for the input, where $N_{\text{patch}} = (1 + D_{\text{hist}} + D_{\text{futr}})N_{\text{hist}} + D_{\text{futr}}N_{\text{futr}}$ denotes the total number of patches.

Furthermore, the latent representation corresponding to the output can be obtained by encoding similarly, excluding $\boldsymbol{z}_{i,t}^{\text{patch}}$:

$$\boldsymbol{z}_t^{\text{tgt}} = \text{Encode}(\boldsymbol{z}_t^{\text{pos}} + \boldsymbol{z}^{\text{stat}}) \in \mathbb{R}^D. \tag{3}$$

$\boldsymbol{z}^{\text{tgt}} \in \mathbb{R}^{N_{\text{futr}}\times D}$ is also acquired by stacking $\boldsymbol{z}_t^{\text{tgt}}$.

Patch Decoder

By using the representations of the output patches $\boldsymbol{z}^{\text{tgt}}$ as the query and the representations of the input patches $\boldsymbol{z}^{\text{src}}$ as the key and value in a multi-head attention mechanism with N_{head} heads, it is possible to relate the contributions of the input patches to the output patches in a decomposable manner:

$$\boldsymbol{z}^{\text{mha}} = \text{MultiHeadAttention}(\boldsymbol{z}^{\text{tgt}}, \boldsymbol{z}^{\text{src}}, \boldsymbol{z}^{\text{src}}) \in \mathbb{R}^{N_{\text{futr}}\times D}. \tag{4}$$

We obtain the latent vector for the patches corresponding to the prediction output by adding the sum of the element-wise product of the latent representation of the input patches $\boldsymbol{z}^{\text{src}}$ and the bias vector $\boldsymbol{w}^{\text{bias}} \in \mathbb{R}^{N_{\text{patch}}\times D}$:

$$\boldsymbol{z}^{\text{pred}} = \boldsymbol{z}^{\text{mha}} + \text{Sum}(\boldsymbol{z}^{\text{src}} * \boldsymbol{w}^{\text{bias}}) \in \mathbb{R}^{N_{\text{futr}}\times D}. \tag{5}$$

Finally, we obtain the predicted values $\hat{\boldsymbol{y}}_{-H:}$ by applying a linear transformation to $\boldsymbol{z}^{\text{pred}}$ to convert it into the patch dimension and then reverting it to the original scale using the mean and standard deviation calculated by RevIN.

Decomposition of the prediction

In the patch encoder, the i-th variable's t-th patch $\boldsymbol{x}_{i,t}^{\text{patch}}$ is independently mapped to a representation vector $\boldsymbol{z}_{i,t}^{\text{patch}}$. On the other hand, in the patch decoder, we utilize multi-head attention only once to associate the input and output patches and subsequently apply linear transformations. When the dimension of each head's value is d_v, multi-head attention typically computes attention via matrix multiplication between an attention weight of dimension $N_{\text{futr}} \times N_{\text{patch}}$ and a value of dimension $N_{\text{patch}} \times d_v$. However, we first compute the element-wise product of tensors of dimension $N_{\text{futr}} \times N_{\text{patch}} \times 1$ and $1 \times N_{\text{patch}} \times d_v$, then sum along the N_{patch} dimension. This method calculates the same attention while

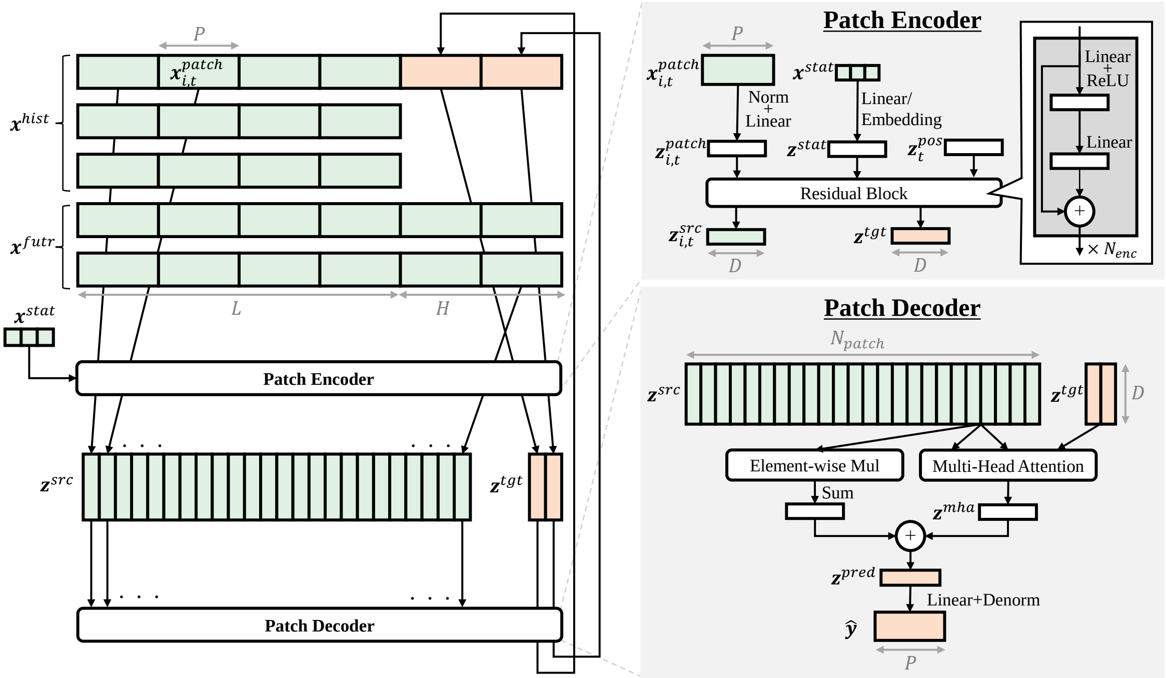


Figure 2: Architecture of PatchDecomp

providing the contributions along the N_{patch} dimension. Additionally, the bias vectors are also computed for each patch. Thus, this architecture can decompose the output according to the contributions of each input patch through the series of processes in the encoder and decoder. This provides the model's predictive rationale more directly than the visualization of attention maps.

Experiments

Experimental Settings

Datasets. We conducted long-term TSF (LTSF) using seven datasets frequently used as benchmarks: ETTh1, ETTh2, ETTm1, ETTm2, Weather, Electricity (ECL), and Traffic. These datasets did not contain exogenous variables, and in the experimental setup, each dataset was considered a univariate time series. By contrast, to evaluate the performance in TSF tasks that include exogenous variables, we used the electricity price forecasting (EPF) datasets (Lago et al. 2021), which contain real data from five electricity markets (NP, PJM, BE, FR, and DE). This dataset included two different exogenous variables for each electricity market, such as system load and wind generation. In addition, we used the month, day of the week, and hour as exogenous variables.

Baselines. As the baseline of LTSF without exogenous variables, we employed the following models: PatchTST (Nie et al. 2022), NBEATSx (Olivares et al. 2023), NHITS (Challu et al. 2023), TFT (Lim et al. 2021), DLinear (Zeng et al. 2023), TSMixer (Chen et al. 2023), Autoformer (Wu et al. 2021), iTransformer (Liu et al. 2023), and TiDE (Das et al. 2023). For the EPF task, we selected models capable of handling exogenous variables, specifically NBEATSx, NHITS, TFT, TSMixer, and TiDE.

Experimental Details. The proposed and existing methods were implemented and evaluated using NeuralForecast (Olivares et al. 2022). In all the experiments, the learning rate was set to 10^{-3}, and the loss function was the mean absolute error (MAE). The data were split into training, validation, and test sets in chronological order to ensure no overlap between the sets. The training was terminated when the prediction accuracy on the validation set no longer improved (early stopping). The hyperparameters that yielded the best prediction accuracy in the validation set were explored using Optuna (Akiba et al. 2019). Although the patch length P can be set in two ways: fixed and variable length, variable length patches do not necessarily align with the user's intuition. For example, calculating contributions for patches of length 17 hours for hourly data could potentially hinder the user's understanding. Considering the affinity with interpretability, this study adopted fixed-length patches. For the LTSF task, the optimal value was determined through hyperparameter tuning from $P \in \{12, 24, 48\}$, whereas P was specifically set to 24 to align with daily units for the EPF task. All the experiments under each condition were conducted in five trials using different seed values on an NVIDIA Quadro RTX8000 GPU.

Accuracy

In the LTSF task, the past time series length was set to $L = 512$, and the future length was configured to four options: $H \in \{96, 192, 336, 720\}$. The quantitative prediction

		PatchDecomp		PatchTST		NBEATSx		NHITS		TFT		DLinear		TSMixer		Autoformer		iTransformer		TiDE	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.362	0.412	0.405	0.436	0.658	0.550	0.703	0.555	0.610	0.519	0.380	0.424	0.435	0.459	0.513	0.497	0.432	0.459	**0.356**	**0.408**
	192	**0.383**	**0.427**	0.415	0.443	0.730	0.582	0.758	0.583	0.648	0.544	0.404	0.442	0.465	0.477	0.507	0.489	0.491	0.495	0.418	0.450
	336	**0.383**	**0.431**	0.439	0.461	0.809	0.619	0.792	0.604	0.623	0.545	0.442	0.467	0.528	0.520	0.485	0.492	0.517	0.518	0.462	0.482
	720	**0.406**	**0.453**	0.457	0.483	0.871	0.655	0.843	0.631	0.624	0.562	0.440	0.476	0.538	0.538	0.543	0.538	0.548	0.547	0.457	0.488
ETTh2	96	0.264	**0.339**	**0.263**	0.340	0.319	0.373	0.330	0.375	0.286	0.363	0.268	0.346	0.353	0.413	0.327	0.396	0.373	0.427	0.275	0.351
	192	**0.316**	**0.374**	0.321	0.379	0.395	0.421	0.409	0.421	0.337	0.393	0.318	0.380	0.412	0.450	0.343	0.406	0.408	0.449	0.325	0.384
	336	0.341	**0.395**	0.354	0.406	0.451	0.458	0.461	0.457	0.360	0.410	0.343	0.401	0.432	0.465	**0.336**	0.404	0.443	0.474	0.350	0.405
	720	**0.366**	**0.422**	0.372	0.424	0.485	0.485	0.477	0.474	0.379	0.432	0.367	0.423	0.484	0.501	0.381	0.439	0.483	0.502	0.374	0.427
ETTm1	96	0.567	0.501	0.322	0.374	0.528	0.479	0.560	0.485	0.610	0.504	**0.310**	**0.368**	0.439	0.441	0.582	0.511	0.443	0.452	0.321	0.378
	192	0.569	0.502	0.356	0.394	0.602	0.520	0.569	0.502	0.639	0.515	**0.333**	**0.383**	0.448	0.448	0.594	0.519	0.470	0.465	0.345	0.393
	336	0.567	0.502	0.384	0.409	0.669	0.549	0.655	0.536	0.619	0.519	**0.353**	**0.395**	0.467	0.459	0.598	0.522	0.487	0.476	0.362	0.403
	720	0.595	0.523	0.441	0.445	0.767	0.599	0.730	0.571	0.629	0.530	**0.397**	**0.420**	0.506	0.480	0.619	0.536	0.534	0.502	0.410	0.430
ETTm2	96	0.190	0.290	**0.166**	**0.259**	0.183	0.279	0.186	0.278	0.203	0.298	0.168	0.265	0.340	0.379	0.256	0.339	0.340	0.380	0.173	0.269
	192	0.230	0.315	**0.215**	**0.294**	0.245	0.322	0.249	0.320	0.245	0.327	**0.215**	0.297	0.400	0.411	0.279	0.352	0.388	0.406	0.220	0.302
	336	0.275	0.345	**0.261**	**0.328**	0.302	0.362	0.307	0.358	0.284	0.351	**0.261**	0.329	0.439	0.433	0.310	0.371	0.434	0.432	0.265	0.331
	720	0.349	0.390	0.337	0.377	0.392	0.419	0.396	0.412	0.352	0.395	**0.334**	**0.376**	0.494	0.471	0.369	0.408	0.481	0.465	0.336	0.378
Weather	96	**0.171**	0.212	0.176	**0.198**	0.196	0.200	0.199	0.201	0.174	0.214	0.179	0.210	0.340	0.325	0.254	0.286	0.369	0.343	0.184	0.218
	192	**0.215**	0.248	0.219	**0.236**	0.248	0.243	0.251	0.242	0.216	0.246	0.223	0.246	0.392	0.351	0.277	0.298	0.424	0.371	0.228	0.251
	336	**0.264**	0.281	0.269	**0.273**	0.299	0.284	0.301	0.283	0.267	0.281	0.267	0.279	0.420	0.370	0.300	0.311	0.451	0.389	0.271	0.283
	720	**0.316**	**0.313**	0.322	**0.313**	0.360	0.330	0.362	0.326	0.318	0.314	0.322	0.316	0.413	0.373	0.333	0.329	0.432	0.384	0.325	0.320
ECL	96	0.148	0.237	0.149	0.250	0.158	0.257	0.155	0.253	0.281	0.346	**0.140**	**0.236**	0.218	0.309	0.266	0.333	0.237	0.332	0.141	0.237
	192	0.168	0.254	0.159	0.262	0.175	0.272	0.170	0.268	0.398	0.421	**0.149**	**0.245**	0.237	0.324	0.310	0.364	0.259	0.349	**0.149**	0.246
	336	0.181	0.269	0.171	0.277	0.188	0.289	0.184	0.284	0.831	0.674	0.160	0.261	0.253	0.338	0.285	0.353	0.272	0.361	**0.158**	**0.258**
	720	0.242	0.318	0.221	0.311	0.231	0.326	0.222	0.319	0.910	0.710	**0.198**	**0.296**	0.321	0.392	0.476	0.479	0.332	0.405	0.200	0.298
Traffic	96	0.418	**0.289**	0.399	0.312	**0.381**	0.300	0.383	0.301	0.528	0.346	0.417	0.305	0.453	0.326	0.611	0.428	0.439	0.352	0.415	0.302
	192	0.431	**0.294**	0.426	0.319	0.409	0.315	**0.408**	0.315	0.610	0.392	0.433	0.311	0.470	0.332	0.631	0.442	0.457	0.359	0.428	0.308
	336	0.433	**0.297**	0.425	0.322	0.425	0.323	**0.423**	0.322	0.690	0.429	0.440	0.317	0.473	0.333	0.572	0.402	0.467	0.364	0.438	0.315
	720	0.480	**0.329**	0.472	0.353	0.477	0.356	**0.463**	0.348	1.063	0.646	0.486	0.348	0.584	0.376	0.767	0.510	0.564	0.402	0.482	0.344

Table 1: LTSF results. The best results are in bold, and the second best are underlined.

accuracies of the proposed and baseline methods are summarized in Table 1.

The performance varied across the seven datasets; however, PatchDecomp recorded high prediction accuracies on several datasets. Notably, impressive results were obtained for ETTh1 and ETTh2. These results can be attributed to the fact that ETTh1 and ETTh2 exhibit significantly pronounced periodicity below the patch length compared with the other datasets, which aligns well with the characteristics of PatchDecomp, which utilizes patch segmentation for forecasting.

Table 2 presents the results for EPF tasks. In this experiment, the past length was set to $L = 168$ and the future length was set to $H = 24$. Similar to the LTSF task, the results for the EPF task exhibited variability depending on the data source (market). However, PatchDecomp and TFT tended to achieve higher accuracies.

To statistically verify the performance differences between methods for the LTSF and EPF tasks, critical difference diagrams (Demšar 2006) were created (Figure 3). The Conover's test (Conover and Iman 1979), a type of nonparametric test, was utilized for post-hoc analysis. In the evaluation of the LTSF based on the mean squared error (MSE) (Figure 3(a)), the proposed method statistically belonged to the highest-ranking group, and it was ranked in the second highest group when evaluated based on MAE (Figure 3(b)). Similar results were observed for the EPF task as well (Figure 3(c) and (d)). Thus, PatchDecomp does not fall short of the baseline methods in terms of prediction accuracy.

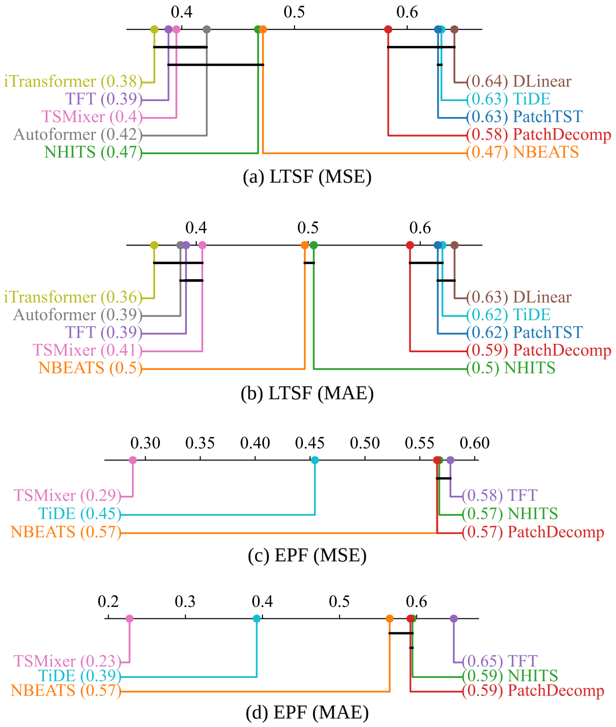
Interpretability

Focusing on the EPF task with exogenous variables, we demonstrate the interpretability of PatchDecomp from both qualitative and quantitative perspectives. As necessary, we clarify the characteristics of the proposed method by comparing it with TFT, which provides interpretability in terms of feature importance and attention weights. For TFT, the importance of a specific variable at a given time can be calculated on a point-wise basis by combining these two types of characteristics.

Qualitative Evaluation. The key aspect of the interpretability of PatchDecomp lies in its ability to decompose predicted values based on the contributions of each input patch. Figure 4 shows an example of the decomposition results for the predicted values in the BE market. The x-axis represents time, and 0 indicates the current time point. The first row (y) shows the time series of the target electricity

	PatchDecomp		NBEATSx		NHITS		TFT		TSMixer		TiDE	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
NP	18.32	2.26	18.31	2.27	17.86	2.22	**16.72**	**2.11**	46.33	4.09	27.20	2.95
PJM	**28.81**	**3.23**	30.88	3.42	31.88	3.39	32.42	3.40	78.66	5.50	44.15	4.19
BE	**182.93**	**5.36**	184.70	5.54	186.74	5.60	187.65	5.37	321.08	8.29	217.41	6.57
FR	218.42	4.70	219.09	4.72	218.75	4.73	**215.81**	**4.59**	341.97	7.60	242.97	5.80
DE	65.18	4.96	61.83	4.80	59.15	4.68	**58.99**	**4.58**	211.49	9.15	139.03	7.63

Table 2: EPF results. The best results are in bold, and the second best are underlined.



(a) LTSF (MSE)

(b) LTSF (MAE)

(c) EPF (MSE)

(d) EPF (MAE)

Figure 3: Critical difference diagrams. The horizontal axis corresponds to the performance rankings of each method, with methods positioned further to the right on the graph indicating a relatively higher performance. The black cross-bars connect methods that do not exhibit statistically significant differences.

price, where the black and red lines represent the actual and predicted values, respectively. The subsequent rows show the exogenous variables used for the prediction, listed from top to bottom: system load (exogenous 1), generation (exogenous 2), month, day of the week (week_day), and hour. In the right column (decomp.), the contributions of each variable to the predicted values are indicated by red lines, and the sum of the red lines for each variable aligns with the final predicted value. Moreover, for each variable, the contributions of the individual patches are displayed as an area chart, with the background color of the patches corresponding to the respective colors in the chart. This enables a precise understanding of the extent to which data contained in the input patches influenced the actual predicted values. Consequently, it is easier to identify the dominant input data affecting the predictions and pinpoint the data responsible for any unnatural predictions, which is expected to significantly contribute to practical applications. For instance, in this case, we can interpret that the patch immediately before the prediction (pink) and the future system load and generation (exogenous 1 and 2, gray) strongly contribute to the prediction. Patch-level explanations, rather than point-wise explanations, can become a user-friendly approach in multi-horizon forecasting problems, as they allow hourly data to be interpreted at granularities such as ”data from yesterday” or ”data from the same day of the previous week.” Additionally, Figure 5 shows a local explanation for a specific time-point prediction, where the contribution of each patch is represented by the intensity of the color. This intensity was calculated from the sum of the absolute values (areas) of the contributions of each patch, as shown in Figure 4. Figure 6 shows global explanations that illustrate the prediction contributions of each patch across the entire test dataset for both the proposed method and TFT. Global explanations elucidate the behavior of the model for the target dataset. When comparing the global explanations of the proposed method and TFT, it is evident that TFT exhibited a distribution of contributions that lacks clarity across various variables and time points, making it challenging to interpret the rationale behind the predictions. By contrast, the proposed method highlighted the contribution of a smaller number of patches. Specifically, the patch immediately preceding the prediction of the target electricity price (y) as well as the future patches for system load (exogenous 1) and generation (exogenous 2), strongly contributed to the predictions.

Quantitative Evaluation. To quantitatively evaluate the interpretability of TSF models, we introduce the concept of comprehensiveness (DeYoung et al. 2019). Comprehensiveness measures how much the output changes when the most important $k\%$ of features from the input are removed; the greater the informational value of the removed input for the output, the larger this metric becomes. In practice, we quantified comprehensiveness using the area over the perturbation curve for regression (AOPCR) (Ozyegen, Ilic, and Cevik 2022). By applying MAE as a benchmark, it can be expressed as follows:

$$AOPCR_k = \frac{\sum_t^T \mathrm{MAE}\left(\mathcal{F}(\boldsymbol{x}_t), \mathcal{F}(\boldsymbol{x}_{t,\backslash k})\right)}{TH}. \qquad (6)$$

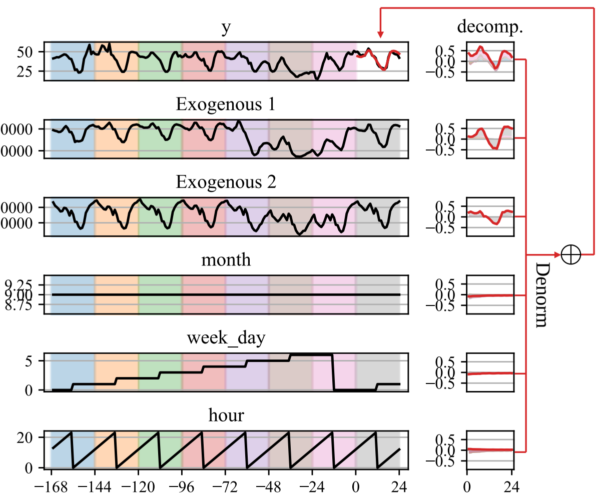


Figure 4: Decomposition of the contributions of the input patches. Each row represents a variable, with the colors corresponding to the input patches. The right column displays the area charts of the prediction contributions for each variable, where the final prediction is obtained by summing the contributions of all the variables.

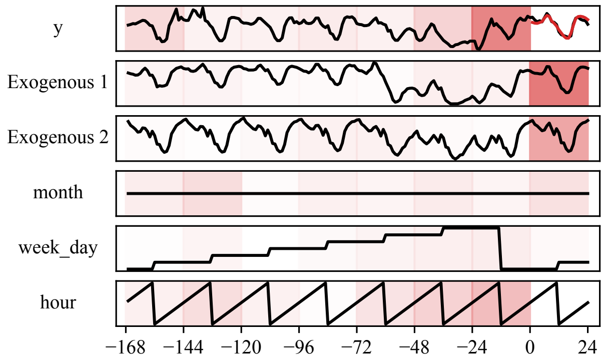


Figure 5: Local explanation. It represents the prediction contributions for each input patch at a specific time by the intensity of the color.

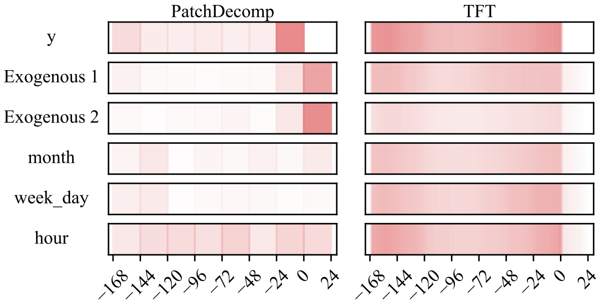


Figure 6: Global explanations of PatchDecomp and TFT. The darker the color, the higher the importance of that part across the entire test dataset.

Here, $\mathcal{F}$ represents the forecasting model, $\boldsymbol{x}_t$ denotes the t-th input, and $\boldsymbol{x}_{t,\setminus k}$ refers to the input after removing the top $k\%$ of values with high prediction contributions from the t-th input. T is the length of the test data and H is the prediction horizon. In this experiment, we set $K = \{5.0, 7.5, 10.0, 12.5, 15.0\}$ and $k \in K$. For each k, we computed the values individually, and we did not calculate the average $AOPCR = \frac{1}{|K|}\sum_k AOPCR_k$ across the entire set K. Additionally, we removed input values by replacing the values in the patches with the mean values of the variables in the entire test data. We calculated $AOPCR_k$ in four different ways: removing patches with high contributions from PatchDecomp (PatchDecomp), removing random patches from PatchDecomp (random), removing pointwise inputs from TFT (TFT-point), and removing inputs aggregated by patch-wise importance from TFT (TFT-patch) (Figure 7). In all five market datasets, PatchDecomp outperformed both the random and TFT values. This result quantitatively demonstrates that the proposed method can effectively highlight patches with a high contribution to the predictions.

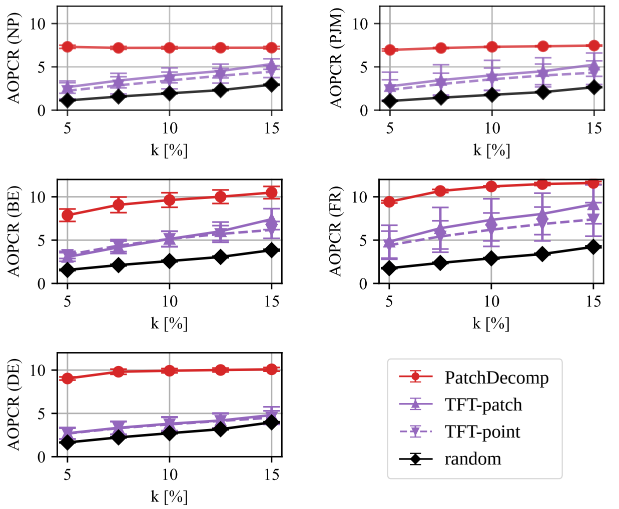


Figure 7: $AOPCR_k$ for $k \in \{5.0, 7.5, 10.0, 12.5, 15.0\}$. The error bars represent the standard deviation.

Conclusion

In this paper, we introduced PatchDecomp, a TSF method that decomposes predicted values into contributions of the input subsequences, explaining how much each subsequence of any variable, including exogenous variables, contributes to the prediction. We conducted experiments on an LTSF task, which consisted of seven types of data, and an electricity price forecasting task, which included five market datasets with exogenous variables. The results demonstrated that the proposed method can achieve a prediction accuracy comparable to that of recently proposed forecasting methods. Furthermore, through the visualization of prediction contributions and quantitative evaluation using AOPCR, we clarified that the interpretability of PatchDecomp was improved compared with that of TFT, which is also an interpretable model.

Currently, PatchDecomp cannot present the contributions of static exogenous variables to predictions, and we plan to address this issue in future studies.

Acknowledgments

We are grateful to Professor Yasushi Sakurai and Professor Yasuko Matsubara of the University of Osaka for their helpful discussions and valuable advice regarding this study.

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2623–2631.

Arsenault, P.-D.; Wang, S.; and Patenaude, J.-M. 2025. A survey of explainable artificial intelligence (XAI) in financial time series forecasting. *ACM Computing Surveys*, 57(10): 1–37.

Challu, C.; Olivares, K. G.; Oreshkin, B. N.; Ramirez, F. G.; Canseco, M. M.; and Dubrawski, A. 2023. NHITS: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 6989–6997.

Chen, S.-A.; Li, C.-L.; Yoder, N.; Arik, S. O.; and Pfister, T. 2023. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*.

Conover, W. J.; and Iman, R. L. 1979. On multiple-comparisons procedures. *Los Alamos Sci. Lab. Tech. Rep. LA-7677-MS*, 1: 14.

Das, A.; Kong, W.; Leach, A.; Mathur, S.; Sen, R.; and Yu, R. 2023. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan): 1–30.

DeYoung, J.; Jain, S.; Rajani, N. F.; Lehman, E.; Xiong, C.; Socher, R.; and Wallace, B. C. 2019. ERASER: A benchmark to evaluate rationalized NLP models. *arXiv preprint arXiv:1911.03429*.

Gangopadhyay, T.; Tan, S. Y.; Jiang, Z.; Meng, R.; and Sarkar, S. 2021. Spatiotemporal attention for multivariate time series prediction and interpretation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3560–3564. IEEE.

Jain, S.; and Wallace, B. C. 2019. Attention is not explanation. *arXiv preprint arXiv:1902.10186*.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

Lago, J.; Marcjasz, G.; De Schutter, B.; and Weron, R. 2021. Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. *Applied Energy*, 293: 116983.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 95–104.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32.

Lim, B.; Arık, S. Ö.; Loeff, N.; and Pfister, T. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4): 1748–1764.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2023. iTransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.

Olivares, K. G.; Challú, C.; Garza, F.; Canseco, M. M.; and Dubrawski, A. 2022. NeuralForecast: User friendly state-of-the-art neural forecasting models. PyCon Salt Lake City, Utah, US 2022.

Olivares, K. G.; Challu, C.; Marcjasz, G.; Weron, R.; and Dubrawski, A. 2023. Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx. *International Journal of Forecasting*, 39(2): 884–900.

Oreshkin, B. N.; Carpov, D.; Chapados, N.; and Bengio, Y. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*.

Ozyegen, O.; Ilic, I.; and Cevik, M. 2022. Evaluation of interpretability methods for multivariate time series forecasting. *Applied Intelligence*, 1–17.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "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*, 1135–1144.

Salinas, D.; Flunkert, V.; Gasthaus, J.; and Januschowski, T. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191.

Schockaert, C.; Leperlier, R.; and Moawad, A. 2020. Attention mechanism for multivariate time series recurrent model interpretability applied to the ironmaking industry. *arXiv preprint arXiv:2007.12617*.

Serrano, S.; and Smith, N. A. 2019. Is attention interpretable? *arXiv preprint arXiv:1906.03731*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Wang, Y.; Wu, H.; Dong, J.; Qin, G.; Zhang, H.; Liu, Y.; Qiu, Y.; Wang, J.; and Long, M. 2024. Timexer: Empowering transformers for time series forecasting with exogenous variables. *arXiv preprint arXiv:2402.19072*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34: 22419–22430.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 11121–11128.

Zhao, Z.; Shi, Y.; Wu, S.; Yang, F.; Song, W.; and Liu, N. 2023. Interpretation of time-series deep models: A survey. *arXiv preprint arXiv:2305.14582*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, 27268–27286. PMLR.

Appendix

LTSF Datasets Description

A description of the long-term time series forecasting datasets is presented in Table 3.

Electricity Transformer Temperature (ETT). The data from two locations in China, collected from July 2016 to July 2018, are referred to as ETTh1 and ETTh2 for data sampled at hourly intervals, and ETTm1 and ETTm2 for data sampled at 15 min intervals (Zhou et al. 2021).

Weather. This dataset consists of meteorological measurements from the Weather Station of the Max Planck Biogeochemistry Institute in Germany for the year 2020 (Wu et al. 2021).

Electricity Consuming Load (ECL). The dataset consists of power consumption (kWh) collected every 15 min for each client from 2012 to 2014, which has been aggregated into hourly data (Li et al. 2019).

Traffic. The dataset contains road occupancy rates (ranging from 0 to 1) for general roads in the San Francisco Bay Area, collected from January 2015 to December 2016 (Lai et al. 2018; Wu et al. 2021).

EPF Datasets Description

This dataset consists of actual electricity prices from five electricity markets (NP, PJM, BE, FR, and DE) (Lago et al. 2021). The description of the datasets is summarized in Table 4.

Hyperparameters for LSTF

The search range for the hyperparameters is summarized in Table 5. The final selected hyperparameters varied depending on the trials.

Hyperparameters for EPF

The search range for the hyperparameters is summarized in Table 6. The final selected hyperparameters varied depending on the trials.

Experimental Details

In LTSF, the batch size was set to 8, while the windows batch size was set to 128. In EPF, these values were set to 16 and 512, respectively. However, owing to memory constraints, the windows batch size was set to 64 specifically for TFT, which requires significantly more memory, in LTSF. For the same reason, TSMixer was also configured with a batch size of 4 instead of 8 in some LTSF experiments. The random seed values were specified within the program at the start of each experiment. Hyperparameter optimization using Optuna (Akiba et al. 2019) was conducted with a maximum of 20 epochs for LTSF, terminating the training if no improvement in the evaluation metric was observed over 5 epochs. For EPF, the maximum limit was set to 2000 epochs, and training was stopped if no improvement was seen over 20 epochs.

Global Explanations

We conducted five trials using different random seeds for each of the five market datasets in the EPF dataset. Figure 8–12 show the global explanations from the five trials of PatchDecomp and TFT. The TFT-point visualizes the predicted contributions calculated at each time step, whereas TFT-patch aggregates and visualizes the contributions at the patch level. PatchDecomp emphasized approximately the same patches when trained on the same market data, even when the models were initialized with different random seeds. By contrast, TFT exhibited a less distinct distribution of predicted contributions, with significant variations in the contributions of certain market data across trials. These results indicate that PatchDecomp provides a higher and more robust level of interpretability than TFT.

Dataset	Dim	Sampling Frequency	(Train, Valid, Test)
ETTh1	7	1 h	(8640, 2880, 2880)
ETTh2	7	1 h	(8640, 2880, 2880)
ETTm1	7	15 min	(34560, 11520, 11520)
ETTm2	7	15 min	(34560, 11520, 11520)
Weather	21	10 min	(36887, 10539, 5269)
ECL	321	1 h	(18414, 5260, 2630)
Traffic	862	1 h	(12282, 3508, 1754)

Table 3: Description of LTSF datasets

	Market	Exogenous variable 1	Exogenous variable 2	Period	(Train, Valid, Test)
NP	the Pennsylvania-New Jersey-Maryland market	2 day-ahead system load	day-ahead wind generation	01-01-2013 to 24-12-2018	(36504, 5448, 10464)
PJM	the Nord Pool market	day-ahead load	2 day-ahead COMED load	01-01-2013 to 24-12-2018	(36504, 5448, 10464)
BE	the Belgium markets	day-ahead load	day-ahead total France generation	09-01-2011 to 31-12-2016	(36504, 5448, 10464)
FR	the France markets	day-ahead load	day-ahead total France generation	09-01-2011 to 31-12-2016	(36504, 5448, 10464)
DE	the Germany markets	day-ahead zonal load	day-ahead wind and solar generation	09-01-2012 to 31-12-2017	(36504, 5448, 10464)

Table 4: Description of EPF datasets

	Patch Size	Hidden Size	Heads	Layers	Units	Blocks	Window Size	Dropout
PatchDecomp	12,24,48	32,64,128,256	4,8	1,2,3,4	-	-	-	uniform(0.0,0.5)
PatchTST	12,24,48	32,64,128,256	4,8	-	-	-	-	uniform(0.0,0.5)
NBEATSx	-	-	-	-	32,64,128,256	-	-	uniform(0.0,0.5)
NHITS	-	-	-	-	32,64,128,256	-	-	uniform(0.0,0.5)
TFT	-	32,64,128,256	4,8	-	-	-	-	uniform(0.0,0.5)
DLinear	-	-	-	-	-	-	11,25,51	-
TSMixer	-	32,64,128,256	-	-	-	1,2,4,6,8	-	uniform(0.0,0.5)
Autoformer	-	32,64,128,256	4,8	-	-	-	-	uniform(0.0,0.5)
iTransformer	-	32,64,128,256	4,8	1,2,3,4	-	-	-	uniform(0.0,0.5)
TiDE	-	32,64,128,256	-	1,2,3,4	-	-	-	uniform(0.0, 0.5)

Table 5: Hyperparameters tuning spaces for LTSF

	Hidden Size	Heads	Layers	Units	Blocks	Dropout
PatchDecomp	16,32,64,128,256,512	4,8	1,2,3,4	-	-	uniform(0.0,0.5)
NBEATSx	-	-	-	16,32,64,128,256,512	-	uniform(0.0,0.5)
NHITS	-	-	-	16,32,64,128,256,512	-	uniform(0.0,0.5)
TFT	16,32,64,128,256,512	4,8	-	-	-	uniform(0.0,0.5)
TSMixer	16,32,64,128,256,512	-	-	-	1,2,4,6,8	uniform(0.0,0.5)
TiDE	16,32,64,128,256,512	-	1,2,3,4	-	-	uniform(0.0, 0.5)

Table 6: Hyperparameters tuning spaces for EPF

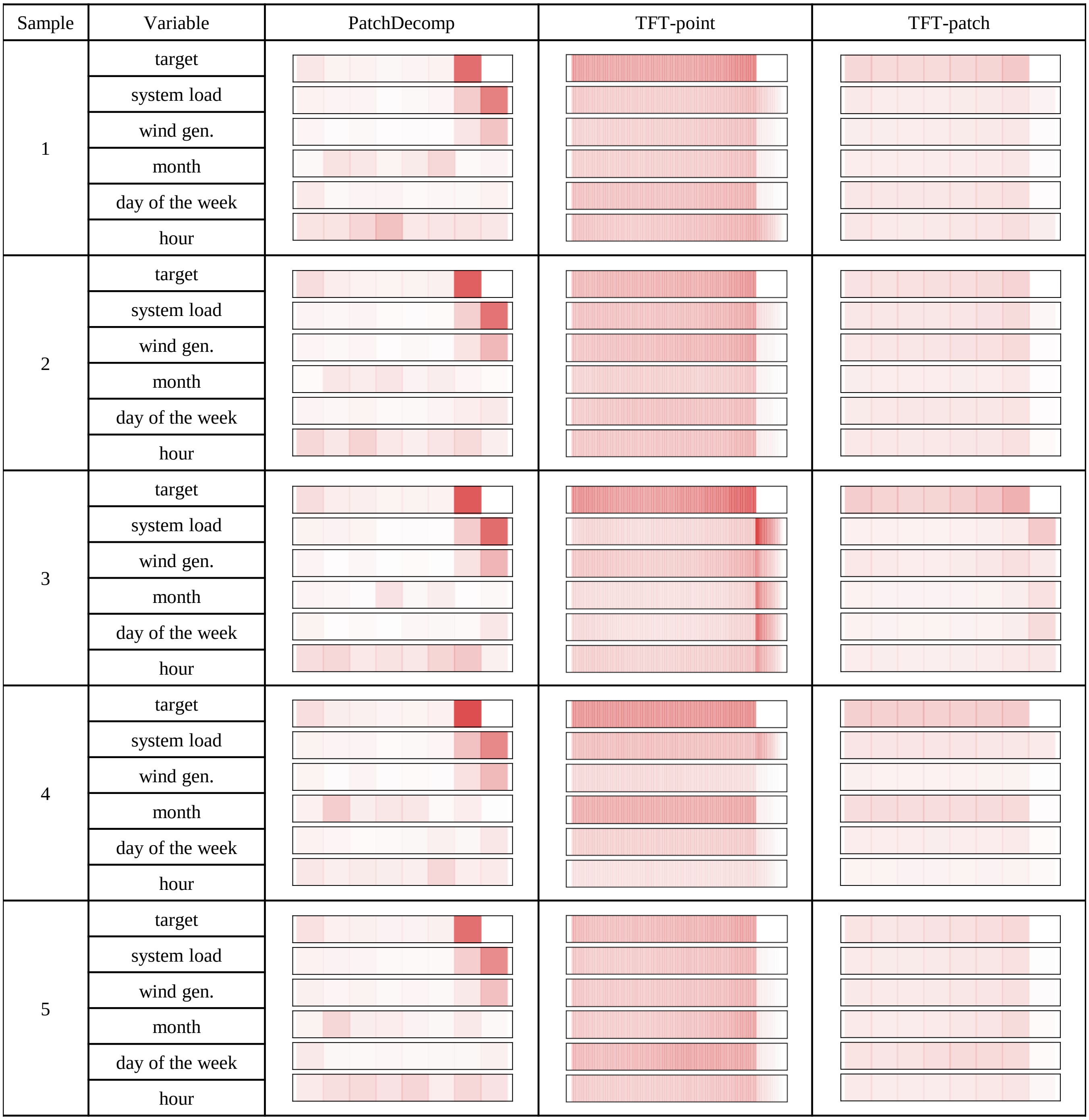


Figure 8: Global explanations on the NP dataset. The term “gen.” indicates “generation.”

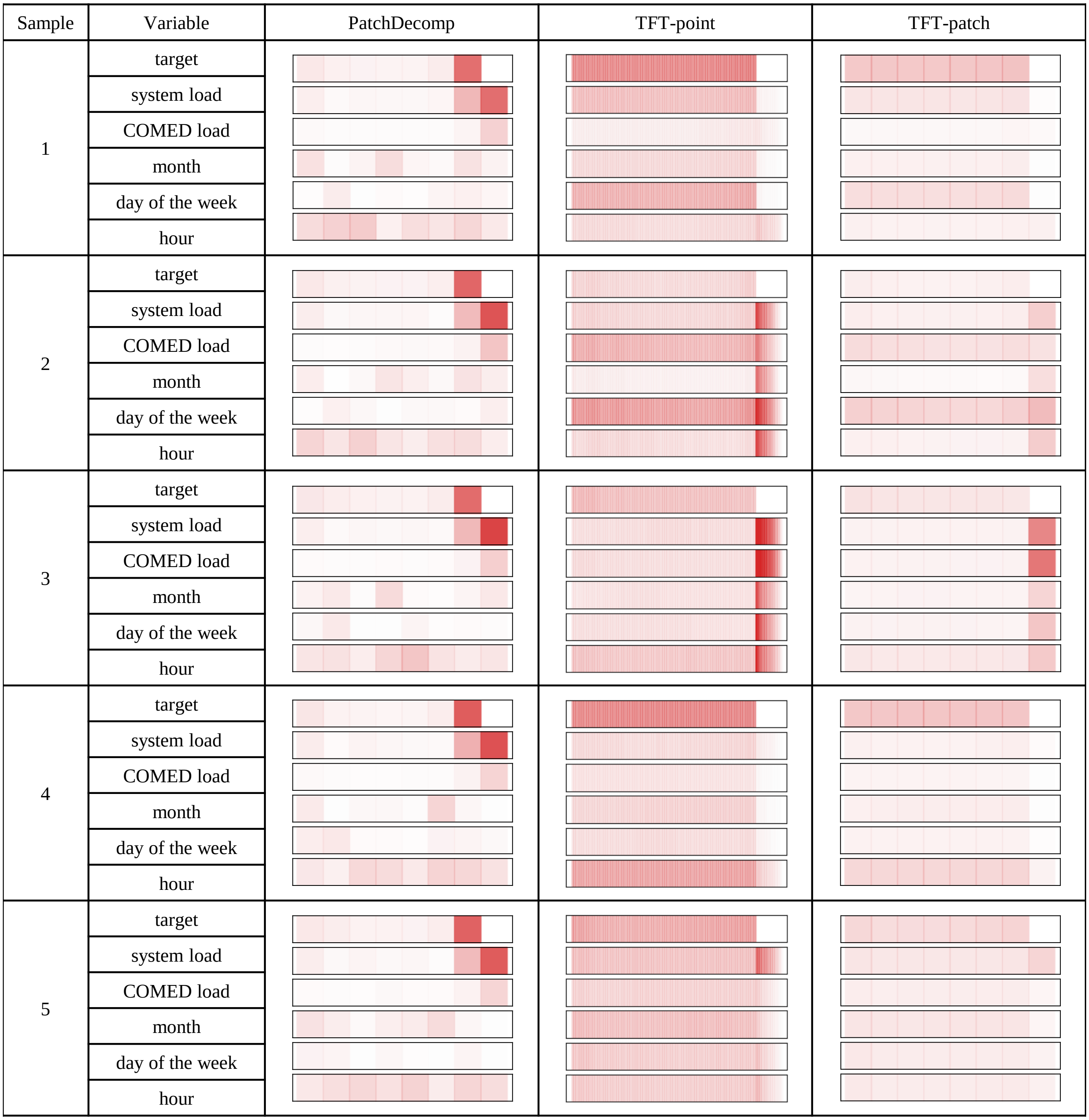


Figure 9: Global explanations on the PJM dataset

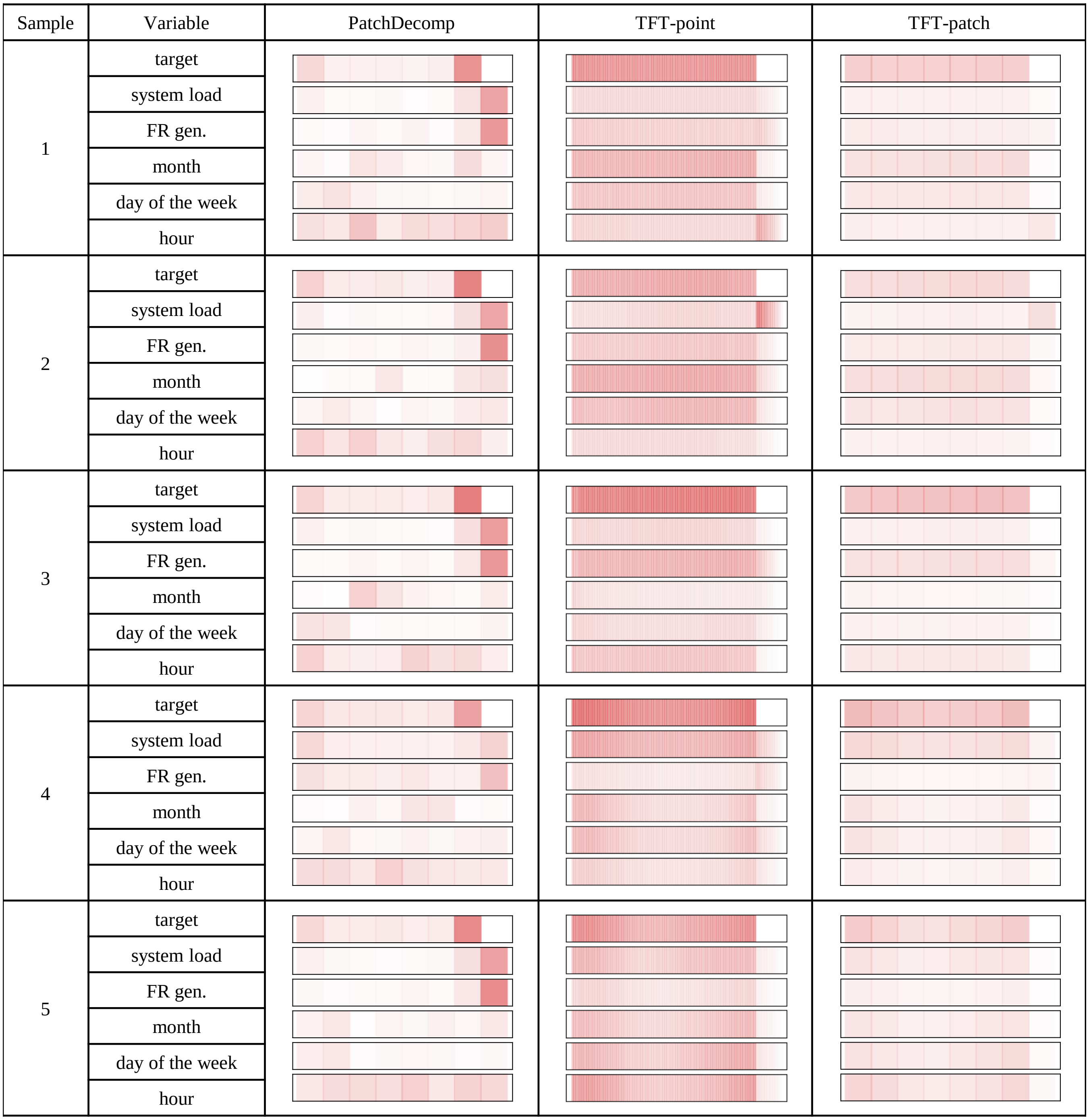


Figure 10: Global explanations on the BE dataset. The term “gen.” indicates “generation.”

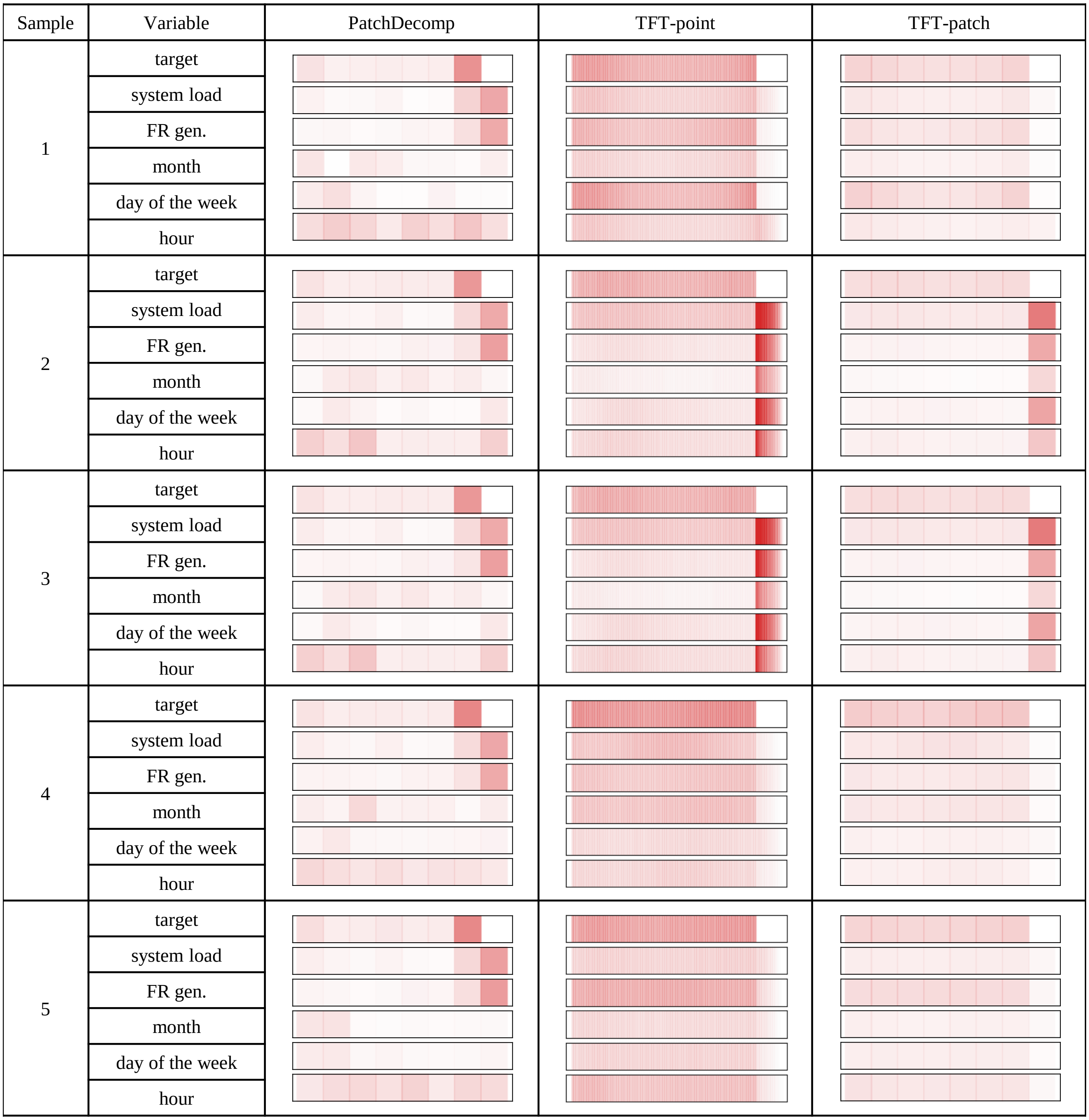


Figure 11: Global explanations on the FR dataset. The term “gen.” indicates “generation.”

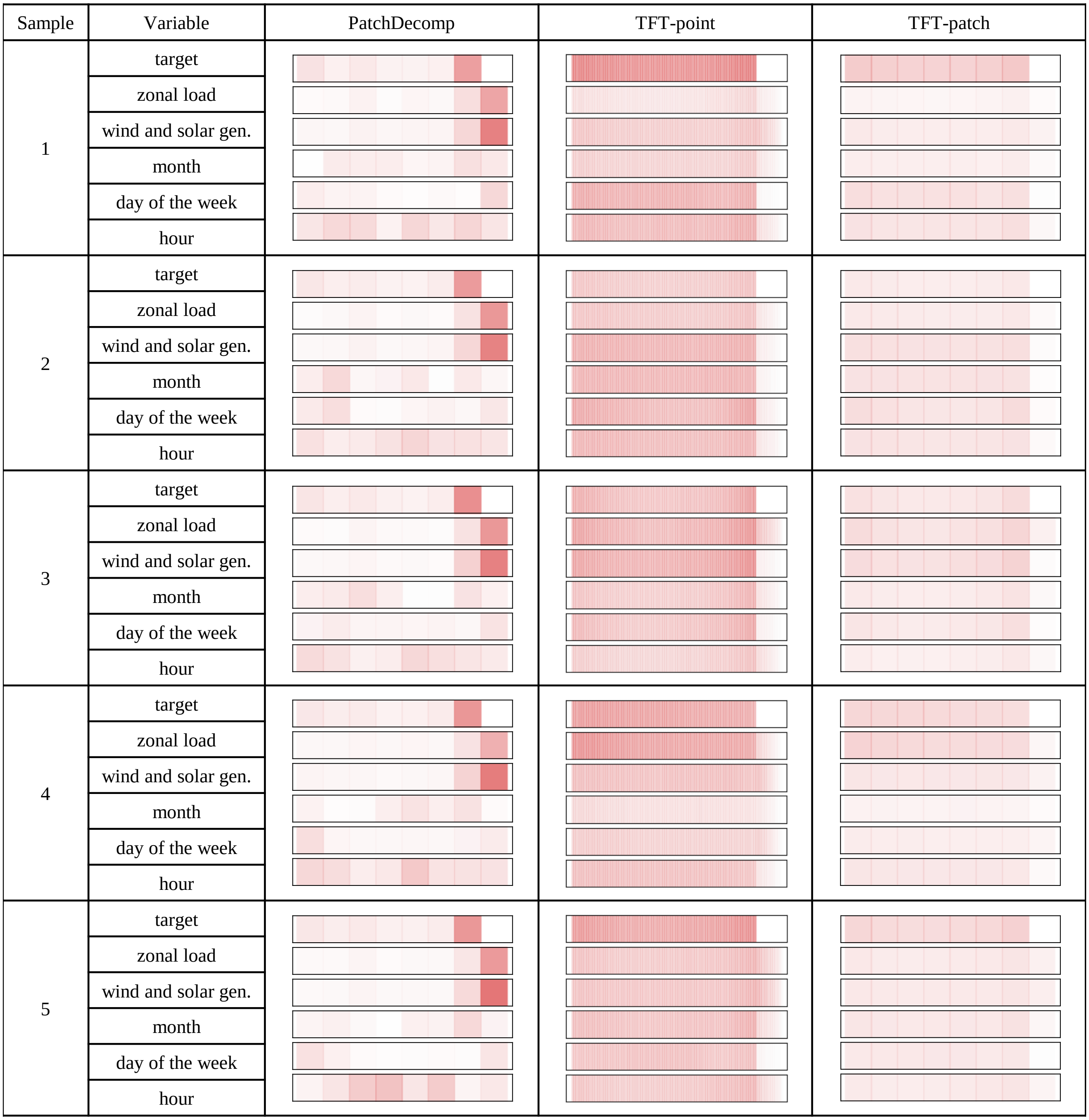


Figure 12: Global explanations on the DE dataset. The term “gen.” indicates “generation.”