WinTSR: A Windowed Temporal Saliency Rescaling Method for Interpreting Time Series Deep Learning Models

Md. Khairul Islam[1], **Judy Fox**[2]

[1]Computer Science Dept, University of Virginia
[2]School of Data Science, University of Virginia
mi3se@virginia.edu, ckw9mp@virginia.edu

Abstract

Interpreting complex time series forecasting models is challenging due to the temporal dependencies between time steps and the dynamic relevance of input features over time. Existing interpretation methods are limited by focusing mostly on classification tasks, evaluating using custom baseline models instead of the latest time series models, using simple synthetic datasets, and requiring training another model. We introduce a novel interpretation method, *Windowed Temporal Saliency Rescaling (WinTSR)* addressing these limitations. WinTSR explicitly captures temporal dependencies among the past time steps and efficiently scales the feature importance with this time importance. We benchmark WinTSR against 10 recent interpretation techniques with 5 state-of-the-art deep-learning models of different architectures, including a time series foundation model. We use 3 real-world datasets for both time-series classification and regression. Our comprehensive analysis shows that WinTSR significantly outranks the other local interpretation methods in overall performance. Finally, we provide a novel and open-source framework to interpret the latest time series transformers and foundation models.

1 Introduction

Time-series deep learning models have achieved unprecedented performance in recent years. However, the lack of explainability remains one of the key challenges for their widespread use. Explanations provide the necessary transparency to make reliable decisions, especially in sensitive data such as healthcare, finance, energy, traffic, weather, stocks, and many other science domains (Benidis et al. 2022). These explanations can be either *Global*, the logic and reasoning of the entire model, or *Local*, the model's specific decision on an instance. *Post-hoc* interpretation methods are generally applied after the model has already been trained, while *In-hoc* methods work during the model training time. *Model-agnostic* methods work on black-box models and do not require specific model architecture to work. Our proposed interpretation method is local, post-hoc, and model-agnostic.

Unlike image and text data, the interpretability of multivariate time series models has been relatively under-explored and difficult to visualize. By explaining time series models, one can highlight the importance of input features to the prediction of the model (Rojat et al. 2021), find intuitive patterns (Lim et al. 2021), and visualize saliency maps (Leung et al. 2023) without relying on the model architecture. This work focuses on the local interpretation techniques for deeper insights into the importance of temporal features. Interpretation methods commonly estimate how relevant each input feature is to the model's output. However, existing works are limited by: (1) benchmarking using simple baseline models (e.g. LSTM, GRU), not recent SOTA time series models that are used in practice (2) focusing mostly on classification tasks, which makes generalization difficult (3) not efficiently capturing temporal dependency (4) train another model to interpret one model.

We propose the *Windowed Temporal Saliency Rescaling (WinTSR)* method to address these challenges. It is benchmarked using the latest time series models of different architectures (including an LLM-based foundation model), tested on both classification and regression tasks, efficiently considers the temporal importance when interpreting, and does not require training a separate model. Our overall framework is summarized in Figure 1. In short, our contributions are,

1) A novel local interpretation method named *WinTSR* that calculates the time importance along the look-back window and rescales the feature importance based on it. This significantly improves the performance by capturing the delayed impact between input and output, while still being relatively fast.
2) Extensive analysis with 3 real-world datasets for classification and regression tasks, and interpretation evaluation using robust metrics.
3) Benchmark *WinTSR* with 5 state-of-the-art time series models (DLinear, MICN, SegRNN, iTransformer) including a foundation model (CALF) to demonstrate that WinTSR is generalizable and consistently outperforms in different model architectures.
4) A unified open-source framework that includes 20+ recent time series models (including 3 foundation models, Appendix E) with 10+ popular interpretation methods. Also, visualize and compare the multivariate

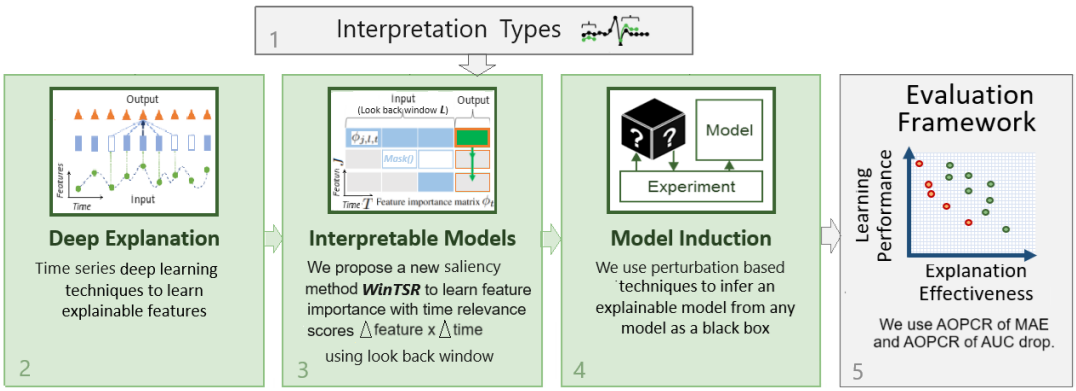


Figure 1: An overview of our framework with the proposed *Windowed Temporal Saliency Rescaling* (**WinTSR**) method.

Table 1: Summary of related works on local interpretation methods for time series. Most works focus on simple synthetic datasets for classification tasks with baseline models (e.g. GRU or LSTM).

Related Work	Method		Model	Dataset			Task	
	Gradient	Perturb.		Real	Synthetic	Time series	Class.	Regr.
FA (Suresh et al. 2017)		•	LSTM, CNN	•			•	
AFO (Tonekaboni et al. 2020)		•	RNN	•	•		•	
FP (Molnar 2020)		•	SVM	•			•	•
IG (Sundararajan, Taly, and Yan 2017)	•		ImageNet, LM, GCNN	•			•	
GS (Lundberg and Lee 2017)	•		CNN	•			•	
TSR (Ismail et al. 2020)	•		LSTM, CNN, Transformer		•	•	•	
DM (Crabbe and van der Schaar 2021)		•	RNN	•	•	•	•	•
EM (Enguehard 2023a)		•	RNN (1 layer GRU)	•	•	•	•	
WinIT (Leung et al. 2023)		•	GRU, LSTM, ConvNet	•	•	•	•	
ContraLSP (Liu et al. 2024d)		•	RNN (1 layer GRU)	•	•	•	•	
WinTSR (This work)		•	DLinear, SegRNN, MICN, iTransformer, CALF	•		•	•	•

temporal trends interpreted by different methods. **Code available** as open source in the Github [1].

2 Related Works

Time series interpretation methods cover a wide range of tasks and datasets (Rojat et al. 2021; Turbé et al. 2023). Table 1 summarizes the comparisons of our work with the related methods. *Gradient based* methods, such as Integrated Gradients (Sundararajan, Taly, and Yan 2017), and GradientSHAP (Erion et al. 2019) use the gradient of the model predictions to input features to generate importance scores. *Perturbation based* methods, such as Feature Ablation (Suresh et al. 2017), and Augmented Feature Occlusion (Tonekaboni et al. 2020) replace a feature or a set of features from the input using some baselines or generated masks and measure the importance based on the model output change. These methods are mainly proposed for image or language models and **do not consider the temporal dependencies in the time series data**.

Recent works like Dyna Mask (Crabbe and van der Schaar 2021), and Extremal Mask (Enguehard 2023a) focus on **learning the masks** to better perturb the input features. Model-based saliency methods, such as (Kaji et al. 2019; Lim et al. 2021; Islam et al. 2023), use the model architecture e.g. attention layers, to generate importance scores. ContraLSP (Liu et al. 2024d), proposed a contrastive learning method to learn locally sparsed perturbation. TIMEX (Queen et al. 2024) trained interpretable surrogates to learn stable explanations from latent space. However, TIMEX is limited to classification and assumes access to latent pretrained space. These methods overall have performed very well on synthetic data or real-world classification tasks. However, they **require training another model to interpret the target model**, which adds additional complexity. These are **not benchmarked for regression tasks** and often include algorithm design exclusively for classification (Queen et al. 2024). Also, **heavily uses simple RNN or LSTM baselines which are not state-of-the-art time series models**.

TSR (Ismail et al. 2020) improved interpretation by considering the temporal dimension separately. However, this comes with a heavy computational cost and was not benchmarked on real-world time series data. Feature Importance in Time, FIT (Tonekaboni et al. 2020) conditioned on the last input time step to calculate the distribution shift using KL-divergence of the predicted probability, but only supports classification tasks. WinIT (Leung et al. 2023) proposed a sliding window-based

[1] https://github.com/khairulislam/Timeseries-Explained

approach to calculate delayed feature importance for classification tasks.

So we need an interpretation method that is generalized for classification and regression, considers the time dependency of input data, and does not require training another model. In this work, we achieve this by, efficiently considering the time dependency, using a simple masking technique, and developing a novel framework that allows comparing these methods with SOTA time series models.

3 Windowed Temporal Saliency Rescaling

3.1 Problem Statement

We consider a multivariate multi-horizon time series setting with length T, the number of input features J, target outputs O, and total N instances. $X_{j,t} \in \mathbf{R}^{J \times T}$ is the input feature j at time $t \in \{0, \cdots, T-1\}$. Past information within a fixed look-back window L is used to forecast for the next τ_{max} time steps or the target class. The target output at time t is y_t. The black-box model f is defined as,

$$\begin{aligned} \hat{y}_t &= f(X_t) \\ \text{where, } X_t &= x_{t-(L-1):t} = [x_{t-(L-1)}, x_{t-(L-2)}, \cdots, x_t] \\ &= \{x_{j,l,t}\},\ j \in \{1, \cdots, J\},\ l \in \{1, \cdots, L\} \end{aligned} \tag{1}$$

where $\hat{y}_t$ is the predicted class or the forecast at $\tau \in \{1, \cdots, \tau_{max}\}$ time steps in the future. X_t is the input slice at time t of length L. An input feature at position (j, l) in the full input matrix at time step t is denoted as $x_{j,l,t}$.

The interpretation goal is to calculate the importance matrix, $\phi_t = \{\phi_{j,l,t}\}$ for each output $o \in O$ or prediction horizon $\tau \in \{1, \cdots, \tau_{max}\}$. This is a matrix of size $O \times J \times L$ for classification and $O \times \tau_{max} \times J \times L$ for regression. We find the relevance of the feature $x_{j,l,t}$ by masking it in the input matrix X_t and measuring the model output change,

$$\boldsymbol{\phi_{j,l,t}} = \mathbf{distance\_score}(\mathbf{f(X_t)},\ \mathbf{f(X_t \setminus x_{j,l,t})}) \tag{2}$$

where $X_t \setminus x_{j,l,t}$ is the input after masking/perturbing feature $x_{j,l,t}$. The *distance_score* can vary based on interpretation methods and prediction tasks. For example, l1-norm for regression and l1-norm or KL-divergence for classification.

3.2 Our Approach

We propose *Windowed Temporal Saliency Rescaling (WinTSR)* across the input time window to calculate the importance score matrix ϕ_t at a time t. Our method differs from previous approaches by accounting for the importance of a feature observation in a multi-horizon setting over multiple windows containing the target time step. The details are in the following Algorithm 1. The method returns an importance matrix ϕ which is later used to evaluate the interpretation performance.

For a distance score, we calculate the simple 'L1 distance' between the original and perturbed prediction for classification and regression. Unlike TSR, which uses the 'L1 distance' between the original and perturbed importance matrix returned from another interpretation method. This significantly improves our run time compared to TSR and removes the dependency on a second interpretation method. WinIT perturbs all values in a sliding window with and without the target feature, then finds the feature importance by subtracting them. Since we perturb only the individual feature, this reduces the computation overhead of perturbing a range of features and removes the dependency of choosing a best-fit sliding window.

The time relevance score enables us to skip less important time steps to speed up the computation similar to (Ismail et al. 2020). TSR uses the first feature value for masking, while WinIT uses feature augmentation (generated from past features). We generated random values from a normal distribution for masking and the input features are already normalized during the data pre-processing period.

Algorithm 1: WinTSR: Windowed Temporal Saliency Rescaling (Proposed)

Given: Input X_t, a time series model $f()$, look-back window L, number of features J.
Output: Feature importance matrix ϕ_t

baseline = $feature\_generator()$
for $l \leftarrow 0$ *to L* **do**
  # Mask all features at time l
  $X_t \setminus x_{:,l,t} = baseline_{:,l,t}$
  # Compute Time-Relevance Score
  $\Delta^{time}_{l,t} = |f(X_t) - f(X_t \setminus x_{:,l,t})|$
# to stabilize the temporal scaling normalize $\Delta^{time}_{:,t}$
for $l \leftarrow 0$ *to L* **do**
  for $j \leftarrow 0$ *to J* **do**
    # Mask feature j at lookback l
    $X_t \setminus x_{j,l,t} = baseline_{j,l,t}$
    # Compute Feature-Relevance
    $\Delta^{feature}_{j,l,t} = |f(X_t) - f(X_t \setminus x_{j,l,t})|$
    # Compute final importance
    $\phi_{j,l,t} = \Delta^{feature}_{j,l,t} \times \Delta^{time}_{l,t}$
return ϕ

4 Experimental Setup

We compare WinTSR to ten recent local and post-hoc interpretation methods. We evaluate them with five state-of-the-art deep learning time series models across three datasets for classification and regression tasks.

4.1 Datasets

We use the datasets shown in Table 2. Electricity and Traffic datasets contain the electricity consumption rate and traffic occupancy over the past hours respectively. The task is to forecast those values for the next 24 hours based on past observations and time-encoded features from the last 96 hours. The MIMIC-III dataset contains patient info and lab reports from a hospital. The goal is to predict whether the patient will die during their hospital stay, based on the

patient demographic info and lab reports over the last 48 hours. This is a private dataset but easy to apply for access and popularly used in related works (Leung et al. 2023; Enguehard 2023a). Details on the datasets and features are in Appendix A. The input values are standard normalized. The datasets are split into train validation test sets using the 8:1:1 ratio. The best model by validation loss is used in testing and for the rest of the experiments.

Table 2: Dataset descriptions with sample size (train, validation, test). Each data has an hourly frequency.

Dataset	Feature	Size	Window	Output	Task
Electricity	5	26.2k	96	24	Regression
Traffic	5	20.7k	96	24	Regression
MIMIC-III	32	22.9k	48	2	Classification

4.2 Models

We use five neural network architecture groups (Linear, CNN, RNN, Transformer, and LLM) for our experiment. Multiple models are chosen to generalize the proposed method across different network architectures. We show how the same interpretation method impacts different models. A complete list of available models in our framework is given in Appendix E. We selected these five models based on their state-of-the-art performance in their respective architecture. These models are : (1) **DLinear** (Zeng et al. 2023) - Linear, (2) **SegRNN** (Lin et al. 2023) - Recurrent Neural Network (RNN), (3) **MICN** (Wang et al. 2023) - Convolutional Neural Network (CNN), and (4) **iTransformer** (Liu et al. 2024b) - Transformer, and (5) **CALF** (Liu et al. 2024a) - A recent pretrained LLM model for generalized time series forecasting using cross-modal fine-tuning.

Table 3: Test results. The best results are in bold.

Dataset	Metric	DLinear	MICN	SegRNN	iTrans.	CALF
Electricity	MAE	0.36	0.39	0.27	**0.24**	0.28
	MSE	0.25	0.29	0.14	**0.11**	0.15
Traffic	MAE	0.36	0.27	0.30	**0.24**	0.32
	MSE	0.28	0.20	0.25	**0.17**	0.24
MIMIC-III	Acc	0.90	0.90	0.90	**0.91**	**0.91**
	AUC	0.77	0.77	0.78	**0.82**	0.77

We follow the Time-Series-Library (Wu et al. 2023) [2] implementation of these models. This ensures we follow the latest benchmarks. Table 3 shows the iTransformer model performs best across all cases. Details of the hyperparameters are listed in Appendix B.

4.3 Interpretation Methods

We use the following post-hoc interpretation analysis methods for comparison in this work: (1) Feature Ablation (FA, Suresh et al. (2017)) (2) Augmented Feature Occlusion (AFO, Tonekaboni et al. (2020)) (3) Feature Permutation (FP, Molnar (2020)) (4) Integrated Gradients (IG, (Sundararajan, Taly, and Yan 2017)) (5) Gradient Shap (GS, Lundberg and Lee (2017)) (6) Dyna Mask (DM, Crabbe and van der Schaar (2021)) (7) Extremal Mask Enguehard (2023a) (8) Windowed Feature Importance in Time (WinIT, Leung et al. (2023)) (9) Temporal Saliency Rescaling (TSR, Ismail et al. (2020)), and (10) Contrastive and Locally Sparse Perturbations (ContraLSP, Liu et al. (2024d)).

We choose them based on versatility across different model architectures and tasks. Captum (Kokhlikyan et al. 2020) [3] and Time Interpret (Enguehard 2023b) [4] libraries were used to implement the interpretation methods. Unlike (Enguehard 2023b), which runs the methods on the CPU, we implemented our framework to run all methods with GPU, thus increasing the interpretation speed. The baselines to mask inputs were randomly generated from the normal distribution, the raw inputs were also normalized. We excluded the methods which are classification only (e.g. FIT) or no public implementation is not available (e.g. CGS-Mask). For TSR, we used the best combination in their work (TSR with Integrated Gradients and $\alpha = 0.55$).

4.4 Evaluating Interpretation

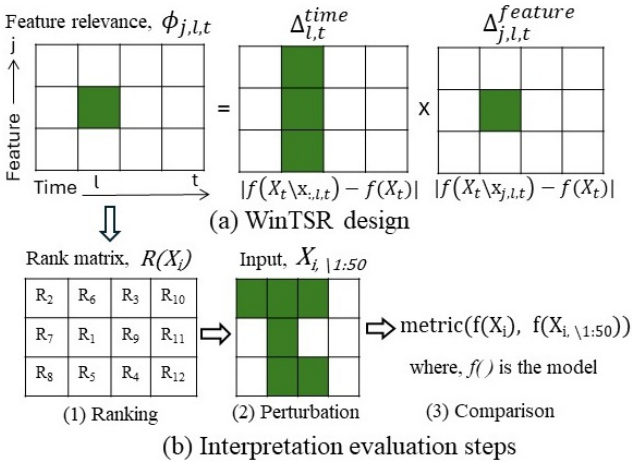


Figure 2: An illustration of (a) our proposed WinTSR method and (b) evaluating interpretation without ground truth. The above example is about calculating comprehensiveness at k=50% with three steps.

We follow (Ozyegen, Ilic, and Cevik 2022; Turbé et al. 2023) to evaluate interpretation when no interpretation ground truth is present. Figure 2 (b) briefly illustrates the evaluation framework. The steps are:

1) Sort relevance scores $\phi(X)$ so that $R_e(X_{i,t})$ is the e^{th} element in the ordered rank set $\{R_e(x_{i,t})_{e=1}^{L\times N}$. Here L is the look-back window and N is the number of features.

[2] https://github.com/thuml/Time-Series-Library

[3] https://captum.ai/

[4] https://josephenguehard.github.io/time_interpret

2) Find top $k\%$ (we used, $k \in \{5, 7.5, 10, 15\}$) features in this set, where $\mathbf{R}(x_{i,t}) \in \{\mathbf{R}_e(x_{i,t})\}_{e=1}^{k}$. Mask these top features or every other feature in the input.
3) Calculate the change in the model's output to the original output using different metrics. We use the AUC drop for classification (Leung et al. 2023) and Mean Absolute Error (MAE) for regression.

DeYoung et al. (2019) proposed to measure the *comprehensiveness* and *sufficiency* to ensure the faithfulness of the explained rationales. Which are similar to the precision and recall from Ismail et al. (2020). (1) *Comprehensiveness*: Were all features needed to make a prediction selected? Once important features are masked, the model should be less confident in its prediction. (2) *Sufficiency:* Are the top feature sufficient to make the prediction? This is achieved by masking all features except the top $k\%$. In summary, **the higher the comprehensiveness loss and the lower the sufficiency loss the better**. We define the set of top $k\%$ relevant features selected by the interpretation method for the i-th input X_i as $X_{i,1:k}$, the input after removing those features as $X_{i,\setminus 1:k}$. Then these two terms are calculated as:

$$\begin{aligned} \text{Comp.} &= \text{evaluation\_metric}(f(X_i),\ f(X_{i,\setminus 1:k})) \\ \text{Suff.} &= \text{evaluation\_metric}(f(X_i),\ f(X_{i,1:k})) \end{aligned} \tag{3}$$

For K bins of top $k\%$ features (we use top 5%, 7.5%, 10%, and 15% features, hence $K = 4$.), the aggregated comprehensiveness score (DeYoung et al. 2019) for the classification task is called the "Area Over the Perturbation Curve" (*AOPC*, Equation 4). For AUC drop, this will calculate the drop for each output class o after masking top $k\%$ features for each $k \in K$, then calculate the average drop.

Similarly for regression, (Ozyegen, Ilic, and Cevik 2022) defined the "Area Over the Perturbation Curve for Regression" (*AOPCR*, Equation 5). For MAE, it calculates the change in prediction for each output o and prediction horizon τ by masking top $k\%$ features for each $k \in K$ then takes the average. AOPC and AOPCR for sufficiency are calculated similarly after replacing $X_{i,\setminus 1:k}$ with $X_{i,1:k}$.

$$AOPC = \frac{\sum_{o,k}^{O,K} \text{metric}(f(X_i)_o, f(X_{i,\setminus 1:k})_o)}{K \times O} \tag{4}$$

$$AOPCR = \frac{\sum_{o,\tau,k}^{O,\tau_{max},K} \text{metric}(f(X_i)_{o,\tau}, f(X_{i,\setminus 1:k})_{o,\tau})}{K \times O \times \tau_{max}} \tag{5}$$

5 Results

This section shows the interpretation results and visualizations. Then discuss our time complexity and the effect of changing the lookback window.

5.1 Benchmark Evaluation

Table 4 shows the overall results. Our method performs the best or second best in most cases. This is consistent across different datasets and models. We ranked the methods for each dataset and model in terms of overall comprehensiveness and Sufficiency. Then we averaged the ranks in the rightmost columns and used for the final rank. **WinTSR achieves the best average rank in each dataset**, 1(1.4 $\pm$ 0.5), 1(1.4 $\pm$ 0.05), and 1(2.4 $\pm$ 1.5) in the Electricity, Traffic, and MIMIC-III respectively.

Integrated Gradient achieves the best results in a few cases for comprehensiveness in regression but fails in others. TSR performs significantly better for comprehensiveness in the MIMIC-III dataset, but its high sufficiency in the same dataset shows the top features it selects are not sufficient. Feature Ablation method also consistently performed well and achieved 2nd rank overall. We also see the mask learner methods, in practice do not interpret the SOTA models well.

5.2 Visualizing Interpretation

Visualizing the interpretations helps to understand their meaning. However, unlike images and texts, time series interpretations are harder to visualize and to verify intuitively. Here we 1) visualize persistent temporal patterns (trends present across the dataset) and 2) highlight the top features across time. Figure 3 shows the raw input feature (left) and the interpretation of these features (using normalized relevance/importance score). The MIMIC-III dataset is visualized with a heatmap due to many features (31), the other two datasets are shown with line plots. The interpretation results of the four selected methods are presented for comparison using the best-performing iTransformer model of 1st iteration.

The relevance scores shown here are for forecasting the target for the next hour ($\tau = 1$) or predicting mortality class ($o = 1$) for simplicity. Electricity and traffic features show a daily pattern, where the importance is highest at the most recent time step ($t = 96$) and the same time the previous day ($t = 72$). Sometimes at the last peak. This means, to predict the electricity consumption or traffic occupancy, past observations from recent times or the same daytime or last peak hour are important. For MIMIC-III the goal is to interpret which factors at which time were important in predicting the patient's death. Figure 3 (c) shows the top three points interpreted by the methods, where WinIT and TSR display the features important in the last 12 hours, whereas WinTSR and FA identify these features much earlier, within the first 12 hours, and then again around the last 12 hours. Temporal change of the important features is visible in WinTSR, WinIT, and TSR as they all consider temporal dependency.

5.3 Time Complexity

We summarize the run time information in Table 5 for some selected methods, Appendix 11 includes the complete results. Our WinTSR method's time complexity is $O(L + L \times J)$, where L is the lookback window and J is the number of features. The perturbation-based methods (FA,

Table 4: **Interpretation benchmark results**, higher comprehensiveness, and lower sufficiency are better. For each dataset, model, and evaluation method, the best results are in bold and the second-best results are underlined. '-' means results not run due to long runtime.

(a) **Electricity** (AOPCR of MAE)

Attribution Method	Comprehensiveness (↑)					Sufficiency (↓)					Rank
	DLinear	MICN	SegRNN	iTrans.	CALF	DLinear	MICN	SegRNN	iTrans.	CALF	(Avg ± Std)
FA	10.7	12.0	11.0	10.7	10.6	12.4	11.1	13.4	16.3	16.4	2(2.6 ± 0.5)
AFO	10.0	11.1	10.2	9.1	9.3	13.0	12.1	14.0	17.3	17.0	3(4.1 ± 0.7)
FP	7.0	9.4	7.9	7.0	7.6	16.0	14.1	15.6	19.3	18.9	10(8.2 ± 1.1)
IG	10.7	**13.7**	11.8	**16.1**	**14.7**	19.0	16.3	15.1	24.8	20.6	6(5.7 ± 4.6)
GS	7.7	10.3	9.1	8.2	9.6	17.4	14.6	15.3	20.4	19.7	8(7.4 ± 2.0)
DM	3.6	7.1	3.9	3.9	4.1	16.6	14.5	16.9	20.5	20.3	11(10.2 ± 0.9)
EM	8.7	9.7	4.8	5.4	8.3	13.6	13.2	16.6	19.2	18.0	8(7.4 ± 1.8)
WinIT	10.6	9.6	9.9	8.8	9.0	13.5	13.7	14.6	17.4	17.7	5(5.5 ± 1.4)
TSR	8.4	11.7	7.7	7.7	-	11.9	14.1	**12.0**	16.4	-	4(5.0 ± 2.8)
ContraLSP	7.6	9.7	4.6	6.0	8.5	13.8	12.5	16.3	18.6	17.4	7(7.3 ± 2.1)
WinTSR	**11.8**	12.7	**12.7**	12.7	13.0	**11.3**	**9.8**	12.1	**14.5**	**14.3**	**1(1.4 ± 0.5)**

(b) **Traffic** (AOPCR of MAE)

Attribution Method	Comprehensiveness (↑)					Sufficiency (↓)					Rank
	DLinear	MICN	SegRNN	iTrans.	CALF	DLinear	MICN	SegRNN	iTrans.	CALF	(Avg ± Std)
FA	10.4	10.1	13.4	9.3	12.4	22.4	16.4	17.9	22.8	22.3	2(2.8 ± 0.6)
AFO	8.5	8.5	12.1	7.6	9.4	23.6	18.2	19.7	24.6	24.3	5(4.9 ± 1.0)
FP	7.5	7.1	10.4	7.2	10.0	24.8	19.4	20.7	24.6	24.8	7(6.4 ± 1.1)
IG	10.9	9.4	10.8	**13.4**	**13.7**	24.1	18.1	21.2	24.8	23.7	4(3.8 ± 2.1)
GS	9.5	8.5	9.2	8.3	11.5	24.2	18.5	21.9	24.8	24.3	6(5.7 ± 1.3)
DM	3.5	4.4	4.1	3.8	4.0	26.3	21.4	23.9	27.9	28.1	11(10.6 ± 0.5)
EM	5.8	5.6	6.5	4.0	8.3	24.1	20.6	22.5	27.0	25.2	9(8.3 ± 1.1)
WinIT	7.3	6.1	11.7	6.3	7.2	25.3	21.1	22.3	25.7	26.5	8(8.0 ± 1.2)
TSR	8.7	7.0	13.5	9.6	-	**15.6**	**11.9**	19.6	22.6	-	3(3.0 ± 2.1)
ContraLSP	4.2	2.9	4.6	4.0	6.1	26.1	22.5	23.0	27.1	23.9	10(9.5 ± 2.0)
WinTSR	**11.2**	**10.8**	**14.3**	10.8	13.1	21.2	15.5	**16.5**	**21.3**	**20.8**	**1(1.4 ± 0.5)**

(c) **MIMIC-III** (AOPC of AUC drop)

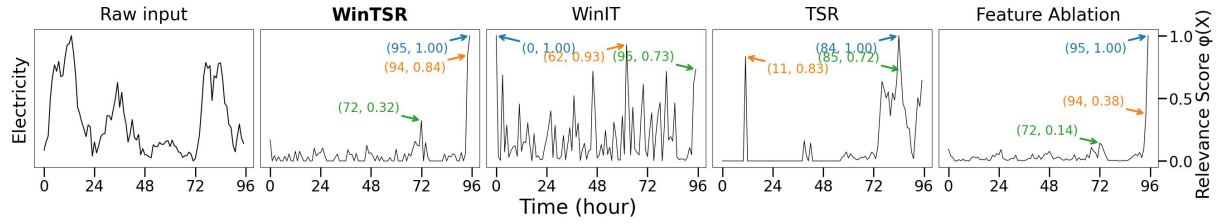
Attribution Method	Comprehensiveness (↑)					Sufficiency (↓)					Rank
	DLinear	MICN	SegRNN	iTrans.	CALF	DLinear	MICN	SegRNN	iTrans.	CALF	(Avg ± Std)
FA	0.61	0.50	0.51	0.46	0.82	0.49	0.46	0.30	0.49	0.83	2(3.6 ± 1.6)
AFO	0.53	0.46	0.45	0.42	0.82	0.45	0.43	**0.27**	0.48	0.83	3(4.8 ± 2.6)
FP	0.55	0.46	0.31	0.42	0.83	0.64	0.54	0.40	0.57	0.85	7(6.6 ± 2.0)
IG	0.51	0.48	0.28	0.42	0.83	0.61	0.54	0.48	0.51	0.84	8(6.9 ± 1.9)
GS	0.55	0.49	0.24	0.41	0.84	0.64	0.57	0.45	0.55	0.87	9(7.5 ± 2.9)
DM	0.48	0.43	0.28	0.40	0.72	0.56	0.56	0.54	0.54	0.78	11(8.2 ± 2.6)
EM	0.53	0.40	0.25	0.43	0.73	0.55	0.55	0.56	0.58	**0.74**	10(7.8 ± 3.0)
WinIT	0.50	0.45	0.46	0.40	0.83	0.53	0.44	0.31	0.51	0.84	6(6.3 ± 2.8)
TSR	**0.80**	**0.90**	**0.79**	**0.83**	-	0.90	0.95	0.87	0.91	-	5(6.0 ± 5.3)
ContraLSP	0.59	0.44	0.26	0.44	0.83	0.48	0.45	0.48	0.47	0.82	4(4.9 ± 2.6)
WinTSR	0.56	0.50	0.52	0.48	**0.85**	**0.44**	**0.42**	0.32	**0.47**	0.83	**1(2.4 ± 1.5)**

Abbreviations: *AOPC*: Area over the perturbation curve for classification, *AOPCR*: Area over the perturbation curve for regression, *FA*: Feature Ablation, *AFO*: Augmented Feature Occlusion, *FP*: Feature Permutation, *IG*: Integrated Gradients, *GS*: Gradient Shap, *DM*: Dyna Mask, *EM*: Extremal Mask, *WinIT*: Windowed Feature Importance in Time, *TSR*: Temporal Saliency Scaling with Integrated Gradients,*ContraLSP*: Contrastive and Locally Sparse Perturbation, *WinTSR*: Windowed Temporal Saliency Rescaling.

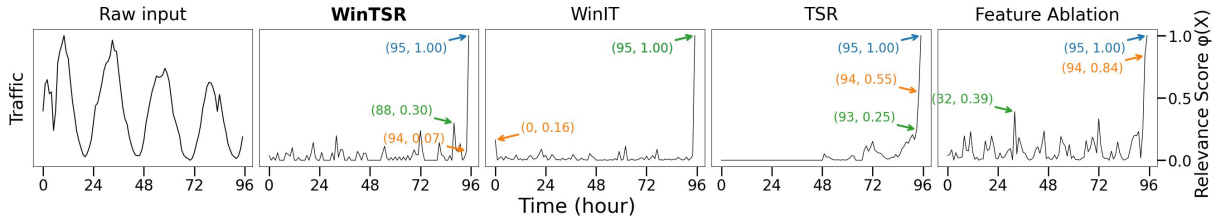
AFO, FP) have similar run-time efficiency $O(L \times J)$ since they also perturb each feature at each time point. WinIT has time complexity $O(L \times J)$, but since it needs to perturb a sliding window of feature each time, it is slower in practice. Gradient-based methods (IG, GS, DM) run the fastest. The *TSR* method is the slowest since it repeatedly applies the *IG* method across each time and feature column, then along the time axis to calculate the time relevance score. The time complexity is $O((L + L \times J) \times O_{IG})$ where O_{IG} is the time complexity of the Integrated Gradient (IG) method. In practice, **WinTSR is around 32 to 367 times faster than TSR**.

5.4 Varying Lookback Window

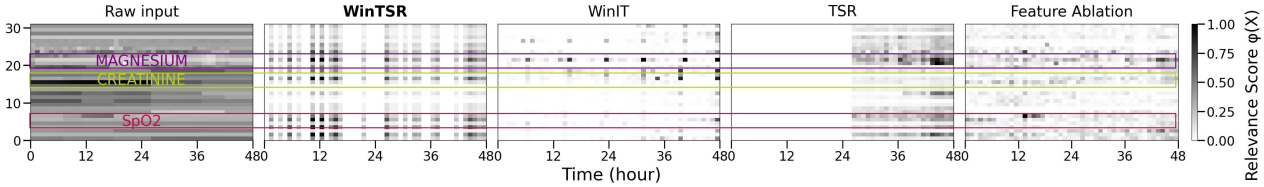
Since the lookback window size is an integral part of capturing temporal dependency, it is important to analyze the effect of changing the window size. By design, the WinIT method supports variable window length, where TSR and WinTSR compute over the whole training window size. We retrained the best-performing iTransformer model for



(a) Electricity



(b) Traffic



(c) MIMIC-III

Figure 3: Interpreted relevance scores (when $\tau = 1$ or $o = 1$) for an arbitrarily chosen example from each dataset, some selected methods, and the iTransformer model. For MIMIC-III the overall most important features of each method are annotated. Both electricity and traffic features show high importance for more recent features and a 24-hour pattern for most methods.

Table 5: **Runtime (minutes)**: A comparison of the interpretation methods averaged across iterations (See the full comparison in Appendix 11). *WinTSR* has a runtime similar to other perturbation-based methods, faster than *WinIT*, and 32 to 367 times faster than *TSR*. '-' means results not run due to long runtime.

Model	FA	WinIT	TSR	ContraLSP	WinTSR
DLinear (Linear)	1.8	2.8	176	2.4	1.9
MICN (RNN)	6.4	7.4	682	7.5	5.7
SegRNN (CNN)	2.6	3.6	725	3.2	2.5
iTransformer (Trans.)	5.2	6.2	1002	5.1	6.3
CALF (LLM)	39.8	36.6	-	48.7	43.3

Table 6: Interpretation performance with different lookback windows for methods considering temporal dependency.

Dataset	Method	Lookback = 24hr		Lookback = 48hr	
		Comp. (↑)	Suff. (↓)	Comp. (↑)	Suff. (↓)
Electricity	WinIT	14.4	23.0	11.5	17.9
	TSR	7.81	21.5	8.15	16.0
	WinTSR	**17.9**	**20.7**	**14.9**	**15.3**
Traffic	WinIT	9.10	25.5	7.50	26.1
	TSR	6.10	25.9	7.49	23.2
	WinTSR	**11.4**	**23.3**	**11.3**	**22.7**
				Lookback = 36hr	
MIMIC-III	WinIT	0.55	**0.63**	0.45	0.55
	TSR	**0.87**	0.93	**0.83**	0.91
	WinTSR	0.59	**0.63**	0.50	**0.49**

different lookback windows and interpreted it by comparing the 3 window-based methods (WinIT, TSR, WinTSR), specifically on temporal dependency. The results are shown in Table 6. We reduced the lookback to 24-hour and 48-hour for Electricity and Traffic (original data have 96-hour lookback). For MIMIC-III, we varied the lookback to 24-hour and 36-hour since the original data had a 48-hour lookback. WinTSR performs best or 2nd best in most cases, showing its robustness across different input window sizes.

6 Conclusion and Future Work

In this paper, we present a novel local interpretation method "Windowed Temporal Saliency Rescaling" that explicitly accounts for the dynamic temporal nature of input data and explains their features' importance. Through extensive experiments and metric comparisons, our analysis 1) **shows WinTSR provides a more accurate interpretation of temporal dependencies among features**; 2) benchmarks different neural network models: DLinear (Linear), SegRNN (RNN), MICN (CNN), iTransformer (Transformer), and CALF (LLM). 3) compares with ten widely used interpretation methods; 4) presents an easy-to-use framework by combining a popular time series library with interpretation libraries. This framework enables the quantitative measurement of time series interpretation across many recent models and methods.

For future work, we will identify higher-level patterns and trends in time series models by explicitly incorporating both spatial and temporal domains, to enhance the effectiveness and efficiency of AI interpretability in time series datasets. We will explore using the pre-trained foundation models to explain features in the time series domain.

References

Benidis, K.; Rangapuram, S. S.; Flunkert, V.; Wang, Y.; Maddix, D.; Turkmen, C.; Gasthaus, J.; Bohlke-Schneider, M.; Salinas, D.; Stella, L.; et al. 2022. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6): 1–36.

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*.

Crabbe, J.; and van der Schaar, M. 2021. Explaining Time Series Predictions with Dynamic Masks. In *International Conference on Machine Learning*.

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*.

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*.

Enguehard, J. 2023a. Learning Perturbations to Explain Time Series Predictions. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*.

Enguehard, J. 2023b. Time Interpret: a Unified Model Interpretability Library for Time Series. *arXiv preprint arXiv:2306.02968*.

Erion, G. G.; Janizek, J. D.; Sturmfels, P.; Lundberg, S. M.; and Lee, S.-I. 2019. Learning Explainable Models Using Attribution Priors. *ArXiv*, abs/1906.10670.

Islam, M. K.; Liu, Y.; Erkelens, A.; Daniello, N.; Marathe, A.; and Fox, J. 2023. Interpreting County-Level COVID-19 Infections using Transformer and Deep Learning Time Series Models. In *2023 IEEE International Conference on Digital Health (ICDH)*, 266–277.

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

Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J. Y.; Shi, X.; Chen, P.-Y.; Liang, Y.; Li, Y.-F.; Pan, S.; et al. 2024. Time-llm: Time series forecasting by reprogramming large language models. *The Twelfth International Conference on Learning Representations*.

Johnson, A. E. W.; Pollard, T. J.; Shen, L.; wei H. Lehman, L.; Feng, M.; Ghassemi, M. M.; Moody, B.; Szolovits, P.; Celi, L. A.; and Mark, R. G. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3.

Kaji, D. A.; Zech, J. R.; Kim, J. S.; Cho, S. K.; Dangayach, N. S.; Costa, A. B.; and Oermann, E. K. 2019. An attention based deep learning model of clinical events in the intensive care unit. *PLoS ONE*, 14.

Kitaev, N.; Kaiser, Ł.; and Levskaya, A. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Kokhlikyan, N.; Miglani, V.; Martin, M.; Wang, E.; Alsallakh, B.; Reynolds, J.; Melnikov, A.; Kliushkina, N.; Araya, C.; Yan, S.; and Reblitz-Richardson, O. 2020. Captum: A unified and generic model interpretability library for PyTorch. arXiv:2009.07896.

Leung, K. K.; Rooke, C.; Smith, J.; Zuberi, S.; and Volkovs, M. 2023. Temporal Dependencies in Feature Importance for Time Series Prediction. In *The Eleventh International Conference on Learning Representations*.

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.

Lin, S.; Lin, W.; Wu, W.; Zhao, F.; Mo, R.; and Zhang, H. 2023. Segrnn: Segment recurrent neural network for long-term time series forecasting. *arXiv preprint arXiv:2308.11200*.

Liu, P.; Guo, H.; Dai, T.; Li, N.; Bao, J.; Ren, X.; Jiang, Y.; and Xia, S.-T. 2024a. Taming Pre-trained LLMs for Generalised Time Series Forecasting via Cross-modal Knowledge Distillation. *arXiv preprint arXiv:2403.07300*.

Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2022a. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *The Tenth International Conference on Learning Representations*.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024b. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

Liu, Y.; Li, C.; Wang, J.; and Long, M. 2024c. Koopa: Learning non-stationary time series dynamics with koopman predictors. *Advances in Neural Information Processing Systems*, 36.

Liu, Y.; Wu, H.; Wang, J.; and Long, M. 2022b. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35: 9881–9893.

Liu, Z.; Zhang, Y.; Wang, T.; Wang, Z.; Luo, D.; Du, M.; Wu, M.; Wang, Y.; Chen, C.; Fan, L.; and Wen, Q. 2024d. Explaining Time Series via Contrastive and Locally Sparse Perturbations. In *Proceedings of the 12th International Conference on Learning Representations*, 1–21.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Molnar, C. 2020. *Interpretable machine learning*. Lulu. com.

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*.

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

Queen, O.; Hartvigsen, T.; Koker, T.; He, H.; Tsiligkaridis, T.; and Zitnik, M. 2024. Encoding time-series explanations through self-supervised model behavior consistency. *Advances in Neural Information Processing Systems*, 36.

Rojat, T.; Puget, R.; Filliat, D.; Del Ser, J.; Gelin, R.; and Díaz-Rodríguez, N. 2021. Explainable artificial intelligence (xai) on timeseries data: A survey. *arXiv preprint arXiv:2104.00950*.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning*.

Suresh, H.; Hunt, N.; Johnson, A. E. W.; Celi, L. A.; Szolovits, P.; and Ghassemi, M. 2017. Clinical Intervention Prediction and Understanding using Deep Networks. *ArXiv*, abs/1705.08498.

Tonekaboni, S.; Joshi, S.; Campbell, K.; Duvenaud, D. K.; and Goldenberg, A. 2020. What went wrong and when? Instance-wise feature importance for time-series black-box models. In *Neural Information Processing Systems*.

Trindade, A. 2015. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C58C86.

Turbé, H.; Bjelogrlic, M.; Lovis, C.; and Mengaldo, G. 2023. Evaluation of post-hoc interpretability methods in time-series classification. *Nature Machine Intelligence*, 5(3): 250–260.

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, H.; Peng, J.; Huang, F.; Wang, J.; Chen, J.; and Xiao, Y. 2023. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*.

Wang, S.; Wu, H.; Shi, X.; Hu, T.; Luo, H.; Ma, L.; Zhang, J. Y.; and Zhou, J. 2024. Timemixer: Decomposable multiscale mixing for time series forecasting. *The Twelfth International Conference on Learning Representations*.

Woo, G.; Liu, C.; Sahoo, D.; Kumar, A.; and Hoi, S. 2022. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *International Conference on Learning Representations*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*.

Yi, K.; Zhang, Q.; Fan, W.; Wang, S.; Wang, P.; He, H.; An, N.; Lian, D.; Cao, L.; and Niu, Z. 2024. Frequency-domain MLPs are more effective learners in time series forecasting. *Advances in Neural Information Processing Systems*, 36.

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*.

Zhang, T.; Zhang, Y.; Cao, W.; Bian, J.; Yi, X.; Zheng, S.; and Li, J. 2022. Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. *arXiv preprint arXiv:2207.01186*.

Zhang, Y.; and Yan, J. 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*.

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.; Sun, L.; Yao, T.; Yin, W.; Jin, R.; et al. 2022a. Film: Frequency improved legendre memory model for long-term time series forecasting. *Advances in neural information processing systems*, 35: 12677–12690.

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

A Dataset and Features

A.1 Electricity

The UCI Electricity dataset (Trindade 2015) contains the consumption of 321 customers from 2012 to 2014. We aggregated it on an hourly level. This data has been used as a benchmark in many time series forecasting models (Wu et al. 2023; Zeng et al. 2023; Ozyegen, Ilic, and Cevik 2022). Following (Wu et al. 2021) we use the past 96 hours to forecast over the next 24 hours. And we added four time-encoded features: month, day, hour, and day of the week.

A.2 Traffic

The UCI PEM-SF Traffic Dataset describes the occupancy rate (with $y_t \in [0, 1]$) of 440 SF Bay Area freeways from 2015 to 2016. It is also aggregated on an hourly level. Following (Wu et al. 2021) we used a look-back window of 96 hours, a forecast horizon of 24 hours, and the 821st user as the target variable. We added four time-encoded features: month, day, hour, and day of the week.

A.3 MIMIC-III Mortality

Table 7: List of MIMIC-III dataset features.

Features	Name
Static	Age, Gender, Ethnicity, ICU admission time
Laboratory	ANION GAP, ALBUMIN, BICARBONATE, BILIRUBIN, CREATININE, CHLORIDE, GLUCOSE, HEMATOCRIT, HEMOGLOBIN, LACTATE, MAGNESIUM, PHOSPHATE, PLATELET, POTASSIUM, PTT, INR, PT, SODIUM, BUN, WBC
Vitality	HeartRate, DiasBP, SysBP, MeanBP, RespRate, SpO2, Glucose, Temp

A multivariate real-world clinical time series dataset with a range of vital and lab measurements taken over time for over 40,000 patients (Johnson et al. 2016). It is widely used in healthcare and medical AI-related research, and also in time series interpretation (Tonekaboni et al. 2020; Leung et al. 2023). We follow the pre-processing procedure by (Leung et al. 2023) to drop patients with missing information and then aggregate. Among the 22988 patient data left in the dataset, 2290 died during their hospital stay. We use the measurements hourly over 48 hours to predict patient mortality (whether the patient died). Table 7 lists the clinical features used from the MIMIC-III patient dataset used in our experiments. There are four static features, twenty lab measurements, and eight vitality indicators.

B Parameters and Notations

The model and training parameters are chosen following (Wu et al. 2023) for a consistent comparison with the state-of-the-art. Table 8 and 9 list the training parameters and the model hyperparameters used during our experiments. Table 10 summarizes the notations mainly defined during the problem statement and interpretation evaluation framework.

Table 8: Training parameters.

Parameter	Value	Parameter	Value
epoch	10	optimizer	Adam
dropout	0.1	batch size	32
device	GPU	iterations	3
loss	MSE or Cross Entropy	learning rate	1e-3
seed	2024		

Table 9: Experiment configuration of the models.

Model	Hyper-parameters			
	Parameter	Value	Parameter	Value
DLinear	moving avg	25		
MICN	encoders	2	kernel	(18, 12)
	decoders	1	dimension	128
SegRNN	layers	1	dimension	128
	seg_len	24		
iTransformer	encoders	2	dimension	128
	n_heads	4	FCN dim	256
	factor	3		
CALF	gpt_layers	6	dimension	768
	tmax	20	lora_alpha	32
	lora_dropout	0.1		

Table 10: Notations used in our work.

Notation	Meaning
T	time series length
f	model
X	input features
J	number of features
L	input lookback window
y	model output
X_t	perturbed input
$\hat{y}$	model output on perturbed input
ϕ	feature relevance matrix
R	rank matrix created from ϕ
O	number of output
τ	output horizon
k	top % of features

C Interpretation methods

The following describes the interpretation methods we have compared within this paper.

1. ***Feature Ablation (FA):*** The difference in output after replacing each feature with a baseline. (Suresh et al. 2017).
2. ***Augmented Feature Occlusion (AFO):*** Tonekaboni et al. (2020) ablated the input features by sampling counterfactuals from the bootstrapped distribution.
3. ***Feature Permutation (FP):*** Permutes the input feature values within a batch and computes the difference between original and shuffled outputs (Molnar 2020).
4. ***Integrated Gradients (IG):*** Sundararajan, Taly, and Yan (2017) assigned an importance score to each input feature by approximating the integral of gradients of the model's output to the inputs.
5. ***Gradient Shap (GS):*** Lundberg and Lee (2017) approximated SHAP values by computing the expectations of gradients by randomly sampling from the distribution of baselines/references.

Table 11: **Runtime (minutes)**: The complete comparison of the interpretation methods averaged across iterations. *WinTSR* has a runtime similar to other perturbation-based methods. While *WinIT* is slightly slower and TSR is around 32-367 times slower.

Dataset	Model	FA	AFO	FP	IG	GS	DM	EM	WinIT	TSR	ContraLSP	WinTSR
Electricity	DLinear	1.83	1.97	1.91	2.29	1.48	1.44	1.36	2.8	176.3	2.4	1.9
	MICN	6.44	6.56	6.52	7.09	4.59	4.46	4.45	7.4	682.2	7.5	5.7
	SegRNN	2.64	2.78	2.71	2.84	2.01	1.96	1.89	3.6	725.4	3.2	2.5
	iTransformer	5.22	5.36	5.32	4.57	3.72	3.62	4.68	6.2	1002.9	5.1	6.3
	CALF	39.8	40.9	40.7	17.7	10.8	18.3	22.1	36.6	-	48.7	43.3
Traffic	DLinear	1.22	1.31	1.27	1.57	0.978	0.939	0.878	1.8	192.6	1.6	1.2
	MICN	4.38	4.46	4.43	4.59	3.11	3.01	2.96	5.0	517.6	5.0	3.8
	SegRNN	1.76	1.85	1.82	1.89	1.33	1.3	1.24	2.3	625.2	2.1	1.7
	iTransformer	3.54	3.64	3.6	3.16	2.52	2.48	3.12	4.2	609.1	3.5	4.3
	CALF	13.7	13.7	13.7	12.4	7.3	12.2	14.7	15.0	-	16.0	15.8
MIMIC-III	DLinear	1.85	2.23	2.05	0.361	0.3	0.461	0.439	6.8	176.0	4.0	4.4
	MICN	6.92	7.31	7.12	0.85	0.617	0.839	0.911	12.0	476.2	5.7	7.4
	SegRNN	3.82	4.71	4.69	0.772	0.483	0.594	1.03	7.9	373.3	4.6	5.9
	iTransformer	5.21	5.61	5.41	0.617	0.511	0.706	0.9	10.2	249.3	3.5	6.4
	CALF	102	103	102	7.77	3.99	26	38.9	106.2	-	19.8	60.2

6. ***Dyna Mask (DM):*** Crabbe and van der Schaar (2021) learned masks representing feature importance.
7. ***Extremal Mask (EM):*** Enguehard (2023a) improved the static perturbation from Dyna Mask by learning not only masks but also associated perturbations.
8. ***Windowed Feature Importance in Time (WinIT):*** Leung et al. (2023) explicitly accounted for the temporal dependence among observations of the same feature by summarizing its importance over a lookback window.
9. ***Temporal Saliency Rescaling (TSR):*** Ismail et al. (2020) proposed to separate the temporal dimension when calculating feature importance and rescaling it.
10. ***ContraLSP:*** Liu et al. (2024d) designed a contrastive learning-based masking method to learn locally sparse perturbations for better explaining feature relevance with and without top important features.

D Time Complexity

Table 11 shows the full run time comparison between the interpretation methods.

E Available Models

Our framework currently includes the following time series foundation models:

1. *CALF*: Aligns LLMs for time series forecasting with cross-modal fine-tuning (Liu et al. 2024a).
2. *TimeLLM*: Reprograms LLMs and its tokenization for better forecasting (Jin et al. 2024).
3. *GPT4TS*: Generalizes pretrained LLMs (GPT-2, Bert) for time series.

We include the following transformer-based and other recent time series models in our proposed framework: (1) *TimeMixer*, Wang et al. (2024) (2) *TSMixer*, Chen et al. (2023) (3) *iTransformer*, Liu et al. (2024b) (4) *TimesNet*, Wu et al. (2022) (5) *DLinear*, Zeng et al. (2023) (6) *PatchTST*, Nie et al. (2022) (7) *MICN*, Wang et al. (2023) (8) *Crossformer*, Zhang and Yan (2023) (9) *SegRNN*, Lin et al. (2023) (10) *Koopa*, Liu et al. (2024c) (11) *FreTS*, Yi et al. (2024) (12) *TiDE*, Das et al. (2023) (13) *LightTS*, Zhang et al. (2022) (14) *ETSformer*, Woo et al. (2022) (15) *Non-stationary Transformer*, Liu et al. (2022b) (16) *FEDformer*, Zhou et al. (2022b) (17) *Pyraformer*, Liu et al. (2022a) (18) *FiLM*, Zhou et al. (2022a) (19) *Autoformer*, Wu et al. (2021) (20) *Informer*, Zhou et al. (2021) (21) *Reformer*, Kitaev, Kaiser, and Levskaya (2020) (22) *Transformer*, Vaswani et al. (2017).

F Reproducibility Statement

Our source code and documentation are already publicly available on GitHub. The Electricity and Traffic datasets are publicly available. The private MIIMIC-III dataset can be accessed by following the steps at https://mimic.mit.edu/docs/gettingstarted/. In addition, we follow the procedures outlined in previous studies to preprocess the datasets. We provide singularity and docker definitions for container environment. We run the experiments on a single-node Linux server with 16 GB RAM and NVIDIA RTX GPU. We use a Python 3.12 environment with Pytorch 2.3.1 and Cuda 11.8. The random processes are seeded to ensure reproducibility.