A More details on the mathematical formulation

A.1 Remarks on definition for perturbation functions

In the main paper, we define the perturbation function. There are two remarks on the definition as follow.

Remark A.1. The perturbation function is not defined as a binary function $\mathrm{Perturb}(v_{feature}, p_{feature})$, because the perturbation may not solely depend on the original value of the feature itself. For example, it may depend on the background distribution of feature values in the dataset.

Remark A.2. If perturbation is based on sampling and uses random sampled data, then $\mathrm{Perturb}_{feature}(p_{feature})$ should be monotonic to $p_{feature}$ when the same data is sampled.

A.2 Remarks on IRFFT

Remark A.3. Note that when FFT is computed for purely real input, the output is Hermitian-symmetric, i.e. the negative frequency terms are just the complex conjugates of the corresponding positive-frequency terms, and the negative-frequency terms are therefore redundant. As a result, we only need to consider the lowest $\left[\frac{N}{2}\right] + 1$ frequencies, and use Inverse real-input FFT (IRFFT) to reconstruct the seasonal component, which auto-completes the negative-frequency terms with the complex conjugates of the corresponding positive-frequency terms before calculating the Inverse FFT.

A.3 Mathematical formulas of comprehensiveness and sufficiency

In the main paper, we choose comprehensiveness and sufficiency (DeYoung et al., 2019) for performance metrics on real-world datasets. The mathematical formulas of comprehensiveness (κ) and sufficiency (ϵ) are defined as follows:

$$\kappa(x, e) = \frac{1}{N+1} \sum_{l=0}^{N} [f(x) - f(\tilde{x}^{(l)})], \tag{26}$$

$$\epsilon(x, e) = \frac{1}{N+1} \sum_{l=0}^{N} [f(x) - f(\hat{x}^{(l)})], \tag{27}$$

where x is a time series instance, e is an explanation which assigns feature importance for all timesteps, N is the length of series, f is the prediction function of the model that outputs the probability of the true label, $\tilde{x}^{(l)}$ is the instance with l most important features removed, and $\hat{x}^{(l)}$ is the instance with l most important features present.

B More details on models and the training process

B.1 Model architecture

We use two kinds of models, CNN (LeCun et al., 1989) and LSTM (Hochreiter and Schmidhuber, 1997), as is mentioned in the main paper.

B.1.1 CNN

The architecture in our CNN is listed as follows:

- 2 convolution layers with ReLU activation and max pooling.
- 2 linear layers with a ReLU activation.
- 1 sigmoid layer.

The configurations in our CNN are:

- Default kernel size in 2 layers: 5 and 3.

- Channels in 2 layers: 16 and 16.
- Pooling kernel size in 2 layers: 2 and 2.
- Linear layers dimension: 16$\times$($length$-8) $\rightarrow$ 16 $\rightarrow$ 1 (1 for binary output, $num\_classes$ for multiple class output).
- Dropout: 0.25.

B.1.2 LSTM

The architecture in our LSTM is listed as follows:

- 1 LSTM layer.
- 2 linear layers with a ReLU activation.
- 1 sigmoid layer.

The configurations in our LSTM are:

- Default hidden size: 64.
- Linear layers dimension: 64 $\rightarrow$ 16 $\rightarrow$ 1 (1 for binary output, $num\_classes$ for multiple class output). .
- Dropout: since gradient of LSTM in PyTorch can only be calculated in training mode where dropout cannot be disabled, the dropout rate of 0 is selected for experiments.
- For CricketX (Mueen et al., 2011) dataset, the number of LSTM layers is 2, and the LSTM is bidirectional.

B.2 Train-test split

In synthesized datasets, $train : test = 8 : 2$. In PTB (Goldberger et al., 2000) and Wafer (Olszewski, 2001) dataset, $train : test = 9 : 1$. In CricketX (Mueen et al., 2011) dataset, $train : test = 7 : 3$. To balance the proportion of labels in training sets and test sets in all real-world datasets, the order of instances is randomly shuffled.

B.3 Hyper-parameters for training

On synthesized *trend* dataset, number of epochs is 20. On synthesized *season* dataset, number of epochs is 50. On synthesized *remainder* dataset, number of epochs for CNN is 50.On other synthetic datasets, number of epochs is 500 and number of epochs for LSTM is 300. On Cricket_X dataset, number of epochs is 200. We use an Adam optimizer (Kingma and Ba, 2014) for training. The initial learning rate is always 0.001, with an exponent scheduler with decay 0.999.

B.4 Information of computer resources

The computer resources used in the experiments are:

- CPU: Intel Core i9-13900k.
- GPU: Nvidia RTX 4080 Super.
- Memory: 64 GB.

C More details on synthesized datasets

C.1 The main synthetic datasets

Each synthetic datasets is generated by five different processes(note that $\varepsilon_t \sim \mathcal{N}(0,1)$):

- Gaussian dataset: Gaussian noise with zero mean and unit variance.

$$X_t = \varepsilon_t$$

- Harmonic dataset: Independent sequences sampled from a harmonic function. A sinusoidal wave was used with $f = 0.25$.

$$X(t) = \sin(0.25\pi f t) + \varepsilon_t$$

- PseudoPeriodic dataset: Independent sequences sampled from a pseudo period function, where, $A_t \sim \mathcal{N}(0, 0.5)$ and $f_t \sim \mathcal{N}(2, 0.01)$

$$X(t) = A_t \sin(2\pi f_t t) + \varepsilon_t$$

- AutoRegressive dataset: Independent sequences of an autoregressive time series process, where, $p = 1$ and $\varphi = 0.9$

$$X_t = \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

- NARMA dataset: Independent sequences of non–linear autoregressive moving average (NARMA) time series, where, the equation is given below, where $n = 10$ and $U \sim U(0, 0.5)$ is a uniform distribution.

$$X_t = 0.3X_{t-1} + 0.05X_{t-1} \sum_{i=0}^{n-1} X_{t-i} + 1.5U\left(t - (n-1)\right) * U(t) + 0.1 + \varepsilon_t$$

Every instance is of length 100 and are re-indexed from 0 to 99 after generated. 50% instances are added bias = 1 from step 25 to step 74. Instances with added bias are labeled positive, and the others are labeled negative. All datasets contain 10,00 instances, in which 800 is for training, 200 is for testing.

C.2 Trend, seasonal and remainder datasets

All time series have the length of 100. The seasonal component has a 50% probability of containing a sine wave of period $T = 10$ with a random phase and an amplitude of 1. To introduce disturbance to frequency domain, it also has an independent 50% probability of containing a sine wave of period $T = 20$ and another 50% probability of containing a sine wave of period $T = 25$, all with random phases and amplitudes of 1. The trend has 50% probability of being $t_t^i = (\frac{t}{50} - 1)^3 - (\frac{t}{50} - 1)$, with another 50% probability of being $t_t^i = -(\frac{t}{50} - 1)^3 + (\frac{t}{50} - 1)$. The remainder contains a normally distributed noise with $\mu = 0$ and $\sigma = 0.05$, and has 50% probability of containing 1 to 3 randomly distributed spikes, each raising a random timestep by 0.5. In the seasonal dataset, only series containing the component of the sine wave of period $T = 10$ are labeled positive. In the trend dataset, only series containing a trend of $t_t^i = -(\frac{t}{50} - 1)^3 + (\frac{t}{50} - 1)$ are labeled positive. In the remainder dataset, only series with spikes are labeled positive. Hence, the portions of labels are balanced in all datasets. All datasets contain 10,000 instances, in which 8,000 is for training, 2,000 is for testing.

D More detailed results of the experiments

D.1 Detailed results on remainder dataset

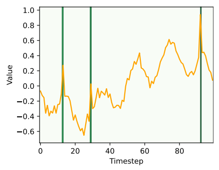
Our method significantly outperforms other methods on *remainder* dataset in both metrics, as is shown in Table 3. The result shows that our proposed method truly identifies the locations of important features in remainder. In addition, our method assigns almost zero saliency value to seasonal and trend components, successfully explaining the model by pointing out that it focuses on the remainder component in the time series. We also find that FO and AFO have a high value of precision but a relatively low recall, indicating that these two methods rarely highlight time steps unless being very confident. On the contrary, SHAP has a low precision, indicating that it identifies a huge number of redundant timesteps as salient.

Figure 5 shows an example where some existing methods fail to identify critical timesteps in the *remainder* dataset, while our method finds them accurately and precisely. The failure of these methods often occurs when there are multiple spikes. Since the label is related to the existence of spikes, the presence of each spike and the model's prediction are in a logical OR relation. In this situation,

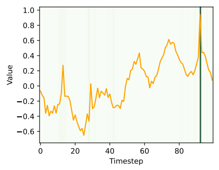
Table 3: Average precision and recall scores on the *remainder* dataset. Best performance in boldface. We report the mean and sample standard deviations in 5 runs.

	CNN		LSTM	
	Precision	Recall	Precision	Recall
IG	0.56±0.08	0.68±0.01	0.28±0.06	0.65±0.03
DeepSHAP	0.61±0.05	0.70±0.01	0.08±0.07	0.13±0.10
FO	0.85±0.04	0.45±0.13	0.43±0.23	0.36±0.12
AFO	0.94±0.02	0.56±0.19	0.35±0.11	0.34±0.10
LIME	0.02±0.01	0.29±0.21	0.02±0.00	0.17±0.01
SHAP	0.03±0.00	0.23±0.02	0.03±0.00	0.23±0.04
STR-Saliency	**0.99±0.01**	**0.93±0.02**	**0.79±0.05**	**0.66±0.10**

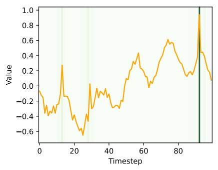
the marginal difference when a timestep is perturbed is almost zero, which leads to the failure of methods that perturb a single feature at a time like IG. Our method does not encounter this problem, because the optimization of a perturbation is done as a whole on all features, thus enabling it to learn to perturb all important timesteps.



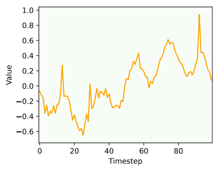
(a) STR-Saliency



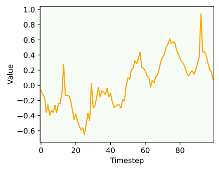
(b) IG



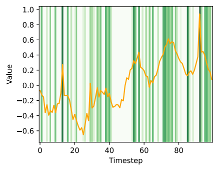
(c) DeepSHAP



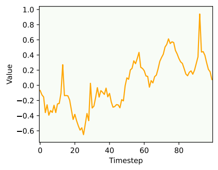
(d) FO



(e) AFO



(f) SHAP



(g) LIME

Figure 5: Sample of saliency maps generated by all seven saliency methods on an instance from *remainder* dataset. STR-Saliency correctly identifies all three spikes. IG and DeepSHAP only identify one of the three spikes. FO, AFO and LIME fail to identify any spike. SHAP highlights a lot of redundant timesteps.

D.2 Detailed results on real datasets

Table 4: Comprehensiveness (κ) and sufficiency (ϵ) over various saliency methods on PTB dataset. Best performance in boldface.

	CNN		**LSTM**	
	Comprehensiveness	Sufficiency	Comprehensiveness	Sufficiency
IG	0.250	0.110	0.154	0.104
DeepSHAP	0.278	0.105	0.186	0.108
FO	0.238	0.222	0.210	0.139
AFO	0.338	0.096	0.207	0.149
SHAP	0.297	0.091	0.146	0.108
LIME	0.287	0.151	0.138	0.258
STR-Saliency	**0.437**	**0.066**	**0.268**	**0.096**

The detailed results in section 3.2 of the main paper are listed in Table 4, 5 and 6. Our proposed method outperforms other saliency methods in almost all metrics. Note that in some situations, the

Table 5: Comprehensiveness (κ) and sufficiency (ϵ) over various saliency methods on Wafer dataset. Best performance in boldface.

	CNN		**LSTM**	
	Comprehensiveness	Sufficiency	Comprehensiveness	Sufficiency
IG	0.665	0.015	0.602	-0.143
DeepSHAP	0.670	**0.011**	0.609	-0.135
FO	0.433	0.136	0.518	-0.082
AFO	0.435	0.136	0.547	-0.068
SHAP	0.294	0.136	0.235	0.004
LIME	0.322	0.306	0.182	0.102
STR-Saliency	**0.678**	0.078	**0.613**	**-0.160**

Table 6: Comprehensiveness (κ) and sufficiency (ϵ) over various saliency methods on CricketX dataset. Best performance in boldface.

	CNN		**LSTM**	
	Comprehensiveness	Sufficiency	Comprehensiveness	Sufficiency
IG	0.618	-0.263	0.482	-0.204
DeepSHAP	0.628	**-0.275**	0.495	-0.218
FO	0.557	-0.220	0.340	-0.113
AFO	0.483	-0.179	0.404	-0.149
SHAP	0.272	-0.042	0.191	-0.028
LIME	0.171	0.145	0.093	0.152
STR-Saliency	**0.642**	-0.245	**0.516**	**-0.247**

sufficiency value is negative, suggesting that the interpretation methods may be selecting the positive features and hiding the negative features to make the model predict a higher probability than the original prediction.

D.3 Ablation Study

We conduct ablation experiments to prove the necessities of some important design components in our method. More details can be found at supplementary material.

D.3.1 Necessity of decomposition

To prove the necessity of the decomposition, we try to skip some steps in the decomposition process, and evaluate the performances on both the *remainder* dataset and real-world datasets. Table 7 shows precision and recall on *remainder* dataset, as well as comprehensiveness and sufficiency on PTB dataset. In the "Way of decomposition" column, "Remainder" means treating the whole time series as remainder, without decomposing, and "Remainder+Seasonal" means decomposing the time series into remainder and seasonal components, *etc.* As is shown in the table, ignoring some components in the decomposition process reduces performance of the saliency map. On the *remainder* dataset, our original method with seasonal-trend-remainder decomposition performs the best. On PTB dataset, since comprehensiveness and sufficiency does not take the seasonal map into account, the extraction of seasonal component does not contribute to performance. Though our original method is not the best in comprehensiveness, it achieves better sufficiency than all ablated methods.

Table 7: Precision & recall over synthetic *remainder* dataset and comprehensiveness & sufficiency over PTB dataset, on CNN model.

	Remainder dataset		PTB dataset	
Way of decomposition	Precision	Recall	Comp.	Suff.
Remainder	0.704	0.738	0.394	0.087
Remainder+Seasonal	0.633	0.749	0.395	0.085
Trend	0.427	0.906	**0.469**	0.067
Trend+Seasonal	0.451	0.922	0.462	0.069
Remainder+Trend	0.968	**0.949**	0.452	0.068
All	**0.986**	**0.949**	0.437	**0.066**

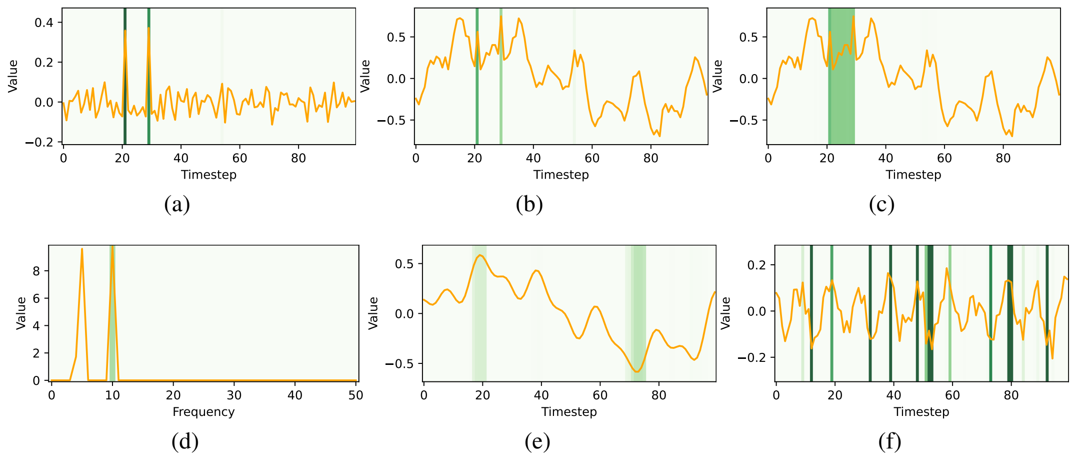


Figure 6: (a): The original map on remainder. (b): Map without decomposition and perturbed as remainder. (c): Map without decomposition and perturbed as trend. (d): The original map on seasonal. In this case the maps on trend and remainder are almost zero-valued. (e): Map on trend, without seasonal extraction. (f): Map on remainder, without seasonal extraction.

Figure 6 (a)-(c)shows the differences between the saliency maps without decomposing and the original saliency map. In the original map, two spikes are precisely identified with high saliency values. In the second one, though the two spikes are also highlighted, the saliency value of the second spike is relatively lower, and another unimportant step is also assigned a non-zero saliency value. In the third map, the whole segment is highlighted, which is not precise enough.

We also visualize the comparison between the original method and the one that skipped the seasonal extraction, both on a time series instance in *seasonal* dataset. As is shown in Figure 6(d)-(f), the trend and remainder map of the latter method overmarks a lot of timesteps as salient, while the original one precisely highlights the critical frequency. When some frequency components are deterministic, the seasonal part is essential for identifying the important features without overmarking a lot of timesteps.

D.3.2 Perturbation function on trend

We try three different perturbation functions on the trend component, as defined in 12, 13 and 14. According to quantitative experiments, the three perturbation functions have similar performances in precision, recall, comprehensiveness and sufficiency. Since the mean value among instances only needs to be calculated for one time, while perturbation through sampling requires multiple calls to the model which is time-consuming, and the calculation of 12 is roughly half of 13, we choose 12 as our perturbation function.

E Limitations

As far as we know, there are some limitations of our work:

- Lack of further utilization of STR-Saliency: STR-saliency forms separate saliency maps on different components given by STR decomposition. The trend, season and remainder parts have clear meaning, which makes corresponding saliency maps more meaningful. This provides a new point of view to the interpretation of time series models from simply the importance of each time step to the importance of the trend around some time steps or sudden changes at certain time steps or even the importance of different frequency components. We can develop methods to use STR-saliency in different ways in the future.
- Computation complexity: the optimization is time consuming. We have some ideas to solve this problem. First, we can adjust the hyper-parameters to sacrifice a little performance for better complexity. Second, we are exploring a method to generate perturbation by sampling, which may reduce the computation complexity.
- The method is dedigned for univariate time series. This leads to some comparison problems with methods on multivariate time series and limits the selection of datasets. However, we think it can be expanded to multivariate cases. Future works are expected.