Learning Saliency Maps to Explain Deep Time Series Classifiers

Prathyush S. Parvatharaju*
Worcester Polytechnic Institute
Worcester, MA, USA
psparvatharaju@wpi.edu

Ramesh Doddaiah*
Worcester Polytechnic Institute
Worcester, MA, USA
rdoddaiah@wpi.edu

Thomas Hartvigsen
Worcester Polytechnic Institute
Worcester, MA, USA
twhartvigsen@wpi.edu

Elke A. Rundensteiner
Worcester Polytechnic Institute
Worcester, MA, USA
rundenst@wpi.edu

Abstract

Explainable classification is essential to high-impact settings where practitioners require *evidence* to support their decisions. However, state-of-the-art deep learning models lack transparency in how they make their predictions. One increasingly popular solution is attribution-based explainability, which finds the impact of input features on the model's predictions. While this is popular for computer vision, little has been done to explain deep time series classifiers. In this work, we study this problem and propose PERT, a novel perturbation-based explainability method designed to explain deep classifiers' decisions on time series. PERT extends beyond recent perturbation methods to generate a saliency map that assigns importance values to the timesteps of the instance-of-interest.

First, PERT uses a novel Prioritized Replacement Selector to learn which alternative time series from a larger dataset are most useful to perform this perturbation. Second, PERT mixes the instance with the replacements using a Guided Perturbation Strategy, which learns to what degree each timestep can be perturbed without altering the classifier's final prediction. These two steps jointly learn to identify the fewest and most impactful timesteps that explain the classifier's prediction. We evaluate PERT using three metrics on nine popular datasets with two black-box models. We find that PERT consistently outperforms all five state-of-the-art methods. Using a case study, we also demonstrate that PERT succeeds in finding the relevant regions of the input time series.

Keywords

Time series classification; Explainability; Deep learning

ACM Reference Format:
Prathyush S. Parvatharaju, Ramesh Doddaiah, Thomas Hartvigsen, and Elke A. Rundensteiner. 2021. Learning Saliency Maps to Explain Deep Time Series Classifiers. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD,Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482446

*Equal Contribution

CIKM '21, November1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482446

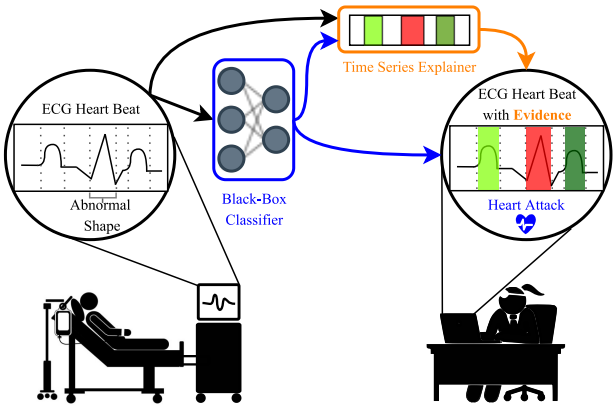


Figure 1: Example of Time Series Saliency Map highlighting input time steps contributing to classification result.

1 Introduction

Background. Deep neural networks are the state-of-the-art solution to many time series classification problems in domains such as healthcare [18, 41] and finance [13, 38]. However, expert end users often do not trust predictions from such "black box" models [32]. As the complexity of these models has skyrocketed, the gap between machine-extracted patterns and data-driven decision making has also widened. To address this problem, recent works have shown that users are more likely to trust a model's predictions if they are shown *what evidence the model used most* [9]. Attribution-based explainability methods thus aim to discover to what degree a classifier used each of its input features.

Motivating Example. Consider a black-box deep time series classifier that detects abnormal heartbeats, or arrhythmia, in ECG data [14, 31]. Without knowing whether or not the model looked at the proper region of the heartbeat, a doctor will not trust its prediction. As shown in Figure 1, one solution is to attribute the prediction to the evidence that the model used most to make its decision. In this example, wave-like shapes are shown in the input ECG time series, one of which is labeled as abnormal. These shapes are all part of a single cardiac cycle. The classifier accurately predicted the ECG is part of a *Heart attack* due to the presence

of unexpected spike in the middle of the cycle. Shown in orange, the explainability method finds that the black-box classifier indeed found the unexpected spike as the evidence for this classification, as shown in red. The expected shapes are reported as green bars.

State-of-the-Art. Recently, attribution-based explainable AI (XAI) has rapidly become popular in the literature [6, 15, 17, 25, 32–34, 37, 40, 44, 45]. However, most of these XAI methods are designed for images [8, 9, 21, 29, 32, 39, 42, 47] and text [1, 20, 22, 36]. while very little has been done for time series. Time series values are unidirectionally correlated. Also, during classification, short contiguous subsequences often contain much of the discriminative information [46]. However, existing explainability methods treat all input features *independently*, ignoring correlations and possibly disrupting discriminative subsequences.

Some of the most recent and promising explainability methods have begun to derive attribution-based explanations by *perturbing* inputs and then observing the effects on the black-box model's predictions. However, current perturbation strategies are developed specifically for computer vision where images are transformed into the same space (meaning, values ranging from [0 to 255]). By darkening pixels of an image or altering their hues, different qualities of a black-box model can be tested. Unfortunately, these approaches do not have clear analogies for time series where the range of a series' values depends entirely on its domain. Some very recent works [2, 10, 12, 16, 25] have only just begun to fill this gap by adapting image and text approaches for time series.

Problem Definition. In this work, we study the problem of *Learning Perturbations* to produce attribution-based explanations for a black-box classifier's prediction for a given time series instance. Our goal is to generate a saliency map, or one value per time step of an input time series, where higher values indicate stronger dependence of the model on the time step. A good explanation method must adapt to the specific input instance. This is a multi-objective optimization problem. Namely, a good saliency map must accurately highlight the most relevant time steps while remaining intuitive to the end-user by finding the *fewest* possible time steps.

Challenges. Despite the importance of explaining deep time series classifiers, three open challenges remain:

- *Heterogeneous series*: Time series within the same dataset can be highly variable, even within the same class. To best explain a classifier's prediction for a given series, a perturbation strategy must be adaptive across both time series and classes.
- *Perturbing long series*: Perturbation methods rely on generating *perturbed* versions of input time series. As the series length increases, the range of possible perturbations grows high. It makes finding perturbations computationally long.
- *Conflicting objectives*: Meaningful explanations are often made at the expense of their accuracy. An explanation highlighting only a few key inputs is intuitive but might fail to explain the model if the black-box actually distributes its focus across many inputs.

Proposed Solution. To derive attribution-based explanations we propose a model-agnostic, perturbation-based time series explainer - ***PE****rturbation by Prioritized* ***R****eplacemen****T*** (PERT). PERT uses a gradient-based search algorithm to discover the impact of perturbing each time step on the black-box model's predictions, producing a learned saliency map. It jointly learns to sample a replacement time-series from an available background dataset containing many time series using a *prioritized replacement selector*. It regularizes the perturbations to ensure perturbations remain similar to other series the model has seen before. As a result, it this way adapts to the given dataset and better yet to each time step of the instance-of-interest.

Contributions Our main contributions are as follows:

- We identify and characterize the open problem of *Learning Perturbations* for deep time series classifiers.
- We propose the first *Learning Perturbations* solution to this open problem which integrates two novel components, jointly learning to solve two tasks: gathering the appropriate time series to aid in perturbation of a specific interest, and *how* to perturb each timestep to best tease out its impact on the model's predictions.
- Using nine popular real-world publicly-available datasets, we conclusively demonstrate that our proposed method performs better on all three proposed metrics compared to five state-of-the-art explainability methods for different types of black-box time series classifiers, for multiple key metrics.

2 Related Works

As machine learning models have become more complex, the need for robust explanations of their predictions has become essential to garner the trust required for experts to react [6, 8, 9, 17, 37, 39, 40, 42, 44, 45, 47]. However, there remains very little focus explicitly on explaining deep time series classifiers [6, 44]. To-date none of the explaining deep time series classifiers have developed *perturbation*-based explanations, despite their clear success in a variety of settings [8, 9]. To the best of our knowledge, our method is the first learnable perturbation approach to explaining black-box time series classifiers.

One popular family of explainability methods develops linear surrogate models that are trained to emulate the behavior of a black-box model for a set of perturbed versions of a given instance [32, 33]. If the linear model is a good enough local approximation, then its coefficients are treated as an explanation. This assumes that if the two models produce the same outputs they must use the inputs in the same way. To train the linear model, a set of perturbed versions of the input instance is created and the black-box model is used to produce one class probability per element. Then, the linear surrogate is trained to predict the same probabilities as the black-box model. Intuitively, if there is not a clear linear relationship between the set of perturbed instances and the black-box model's predictions, a linear model's coefficients cannot be treated as explanations.

For time series, which are often high-dimensional and complex, this assumption of linearity is particularly unrealistic, as supported by the vast amount of successful distance-based time series classifiers, many of which explicitly rely on non-linear relationships between series [7, 27, 43]. Further, prior works generate perturbations using ad-hoc hand-crafted perturbation functions such as random replacement of features with either zero [32] or randomly-selected instances from a larger dataset [12, 21].

To avoid the unrealistic linearity assumption, game theory based methods like SHAP [21] and its variants [2, 12, 21, 25], generate explanations by computing each feature-wise marginal contribution for the black box model's predictions. In practice, these methods are extremely slow to compute all permutations to learn the marginal contribution of each feature. Another approach to alleviate the need for linear surrogate models by explicitly *learning* to perturb features of the inputs and measuring the effect on the black-box model's prediction [8, 9, 29, 44]. If an input can be changed dramatically without impacting the prediction, then it is deemed unimportant.

Some works attempt to extract such relationships by randomly setting input features to zero [29], smoothing local features [9], or using a generative model to produce new estimates [44]. This is a flexible approach that readily allows for the encoding of prior knowledge into the explanations by encouraging different behaviors during optimization such as contiguous regions of equally-important features [8] or finding very few important time steps [9]. However, most prior works have explicitly focused on explaining computer vision models, as opposed to time series and their perturbation functions assume the capture of solely visual features [8]. In images, the proper *range* for values in the perturbations is clearly known as each pixel is described by *color*. This is in contrast to the far more dynamic and dataset-specific ranges common to time series. A successful perturbation-based explainability method for time series data must carefully adapt to the proper range and dynamics of the respective time series dataset.

3 Methodology

3.1 Problem Definition

Assume we are given a set of N time series $\mathcal{D} = \{X^1, \ldots, X^N\}$ and a black-box classifier $f_c : \mathbb{X} \rightarrow \mathcal{Y}$, where $\mathbb{X} \in \mathbb{R}^T$ is the T-dimensional feature space and $\mathcal{Y}$ is the label space. Let us consider one instance-of-interest $X \in \mathcal{D}$ where $X = [x_1, \ldots, x_T]$ along with a class-of-interest C, the prediction for which we would like an explanation.

Our goal is to learn a saliency map $\theta = [\theta_1, \ldots, \theta_T]$ where $\theta_t \in [-1, 1]$ indicates the *importance* of time step t with respect to $P(C|X)$, the probability of class C as predicted by black-box f_c. The scale of the real-valued θ_t reflects the importance of the corresponding time step x_t.

While the notion of *importance* is challenging to quantify, we follow the lead of recent work on perturbation-based explainability [8, 9] and assume that it is a reflection of the *expected scale of the change of* $P(C|X)$ *when* X *is perturbed*: $|P(C|X) - P(C|\tilde{X})|$, where $\tilde{X}$ is a perturbed version of X. Naturally, the perturbation-based explanation problem is thus to derive a *perturbation function* $f_p(X)$ that can be used to discover the impact of perturbing each time step x_t on the predicted probability of class C. The more a time step can be changed without impacting the probability, the less important the time step. Using this perturbation function, a saliency map θ may be derived. A successfully learned saliency map θ will assign high values to the most-impactful time steps, effectively ranking them by their impact on $P(C|X)$. Intuitively, an explanation that is *simpler* is often easier to understand, and so we also prefer that $\sum_{t=1}^{T} \theta_t$ be small so as to encourage unimportant timesteps to go to zero.

3.2 Proposed Method: PERT

To derive an explanation for the class prediction made by a black-box model f_c, we propose a model-agnostic, perturbation-based method specific to time series that uncovers the importance of each timestep to the model's final prediction. We refer to our method as ***PE**rturbation by Prioritized **R**eplacemen**T** (PERT)*. Perturbation involves modifying the time steps from X. As illustrated in Figure 2, PERT adaptively employs the dataset $\mathcal{D}$, learning to sample a *replacement* time series R which is used to mix each time step of the instance-of-interest $\hat{X}$ with the corresponding time step from R to create in-distribution perturbations. Furthermore, by querying the black-box model's predictions for a perturbed version of the instance-of-interest X, PERT discovers how sensitive predictions are to perturbations in each times step. To encode these ideas, PERT contains two components: (1) A *Prioritized Replacement Selector* that learns to perform weighted sampling of the time series in $\mathcal{D}$, finding the best replacement time series R specific to instance X and (2) A *Guided Perturbation Function* that learns θ_t to perturb X and discover the impact of each timestep on the black-box model's final prediction. These two steps jointly learn to generate a *saliency map* with the fewest and most impactful timesteps that explain the classifier's prediction.

3.2.1 Prioritized Replacement Selector. To perturb the values of a time series, we must choose new values to replace them with. However, this choice is highly impactful in time series since the shapes and trends of the signals may be altered dramatically as we show in our Experiments in Section 4.5.2. To avoid such massive changes, which might lead to time series unlike any the classifier has seen before, we instead opt to replace values with those of other time series in dataset $\mathcal{D}$. Thus the task of the first component of PERT, the Prioritized Replacement Selector, is to choose which time series is appropriate to use from the background dataset for replacement of the timesteps of instance X. Intuitively, the choice of best replacement time series may vary by timestep and so we achieve this on a timestep-by-timestep basis, choosing different replacement series and corresponding values for each timestep of X. In practice, to provide evidence both *for* and *against* class C, the Prioritized Replacement Selector chooses two time series representatives R_C and R_O from $\mathcal{D}$ where R_C is a series from class C and R_O is a series not from class C. As discussed in Section 3.2.2, these series will be used together to generate one final perturbation $\tilde{X}$.

To acquire R_C and R_O, we split $\mathcal{D}$ into two subsets: $\mathcal{D}_C$ contains all samples from the class-of-interest C, and $\mathcal{D}_O = \mathcal{D} - \mathcal{D}_C$ contains all remaining, or opposing, instances. All instances in each set are assigned individual weights w to serve as their *priorities*, similar to Prioritized Experience Replay [35]. We then sample one replacement time series R from each set by using the softmax of the set's respective weights w_O and w_C to parameterize two independent categorical distributions:

$$\mathrm{P}(R_C^i) = w_C^i$$
$$\mathrm{P}(R_O^i) = w_O^i$$

where R_C^i is the i-th instance of dataset $\mathcal{D}_C$ and w_C^i is its corresponding weight. Intuitively, instances with higher weights w will

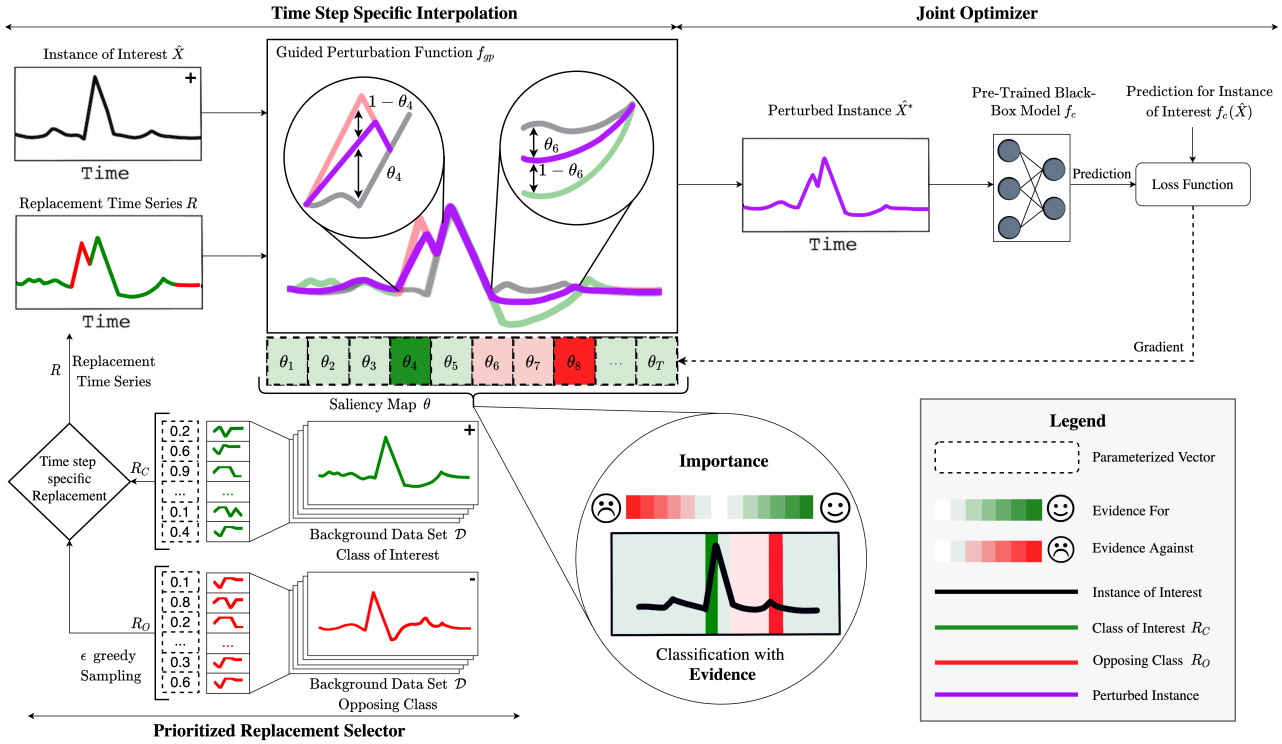


Figure 2: PERT Architecture

have a higher likelihood of being selected. Since w will eventually be learned (Section 3.2.3), the chosen representatives will differ according to the instance X, resulting in adaptive explanations.

As proposed, values of w that are large early in training may be exploited. Therefore, we follow the literature [23] and employ the standard ϵ-greedy approach to balancing *exploration* and *exploitation* while PERT is being trained:

$$R_C = \begin{cases} R_C \text{ (itself)} & \text{with probability } 1-\epsilon \\ \text{random } R_C \in \mathcal{D}_C & \text{with probability } \epsilon \end{cases} \quad (1)$$

Thus when ϵ is large, replacement series are picked randomly and when ϵ is small, weights w are used to select the replacement series. Selection of R_O is performed the same way. To encourage early exploration and later exploitation, ϵ is exponentially decayed during training.

3.2.2 Guided Perturbation Function. Next, we compute the perturbation $\tilde{X}$, which is a modified version of instance X using our *Guided Perturbation Function* and replacement series R_C and R_O. $\tilde{X}$ will subsequently be fed to the black-box model f_c to observe how this perturbation has affected its prediction. To achieve this, we *learn* a perturbation function $f_p : \mathbb{X} \rightarrow \mathbb{X}$. The key component of f_p is a vector of learnable weights $\theta \in [-1, 1]^T$, which will serve as the final saliency map and be the explanation of the model's prediction. Higher values of θ indicate stronger evidence *for* class C while lower values are evidence *against* class C

To take into account both evidence *for* and *against* class C, our perturbation function f_p chooses between replacing values at each timestep with those from R_C and R_O adaptively. To make it so that $\theta_t < 0$ indicates that its corresponding time step X_t is evidence *against* class C, we replace the time step with R_C, the representative of class C. Similarly, when $\theta_t \geq 0$, X_t is replaced with the corresponding timestep of R_O. This way, PERT learns the degree of sensitivity of each time step. The function f_p generates its final perturbation $\tilde{X}$ by performing *timestep-specific interpolation*:

$$\tilde{X} = \theta \odot X + (1-\theta) \odot (\mathbb{1}_{\theta<0} \odot R_C + \mathbb{1}_{\theta \geq 0} \odot R_O) + g \quad (2)$$

where $\mathbb{1}_{\theta<0}$ is a vector-wise indicator function that returns 1 for elements of θ that are less than 0 and $\mathbb{1}_{\theta \geq 0}$ returns 1 for elements of θ greater than or equal to 0. $\odot$ is the Hadamard product. For readability, we refer to this operation as function $f_p(X; \theta)$. In practice, we also add a small amount of Gaussian noise g to avoid overfitting θ to extremely specific values, similar to [9]. Using Equation 3.2.2, the final values of $\tilde{X}$ are thus interpolations between the original timesteps of X and the replacement series R_C or R_O according to the scale of the corresponding value in θ.

We initialize the values of θ uniformly between $-1e^{-2}$ and $1e^{-2}$ to encode no prior assumptions on which time steps are most important and are iteratively updated throughout the training. We start with small values for θ and thus perform *forward selection*, steadily inserting only the most crucial timesteps to the prediction $P(C|\tilde{X})$. This way, we encourage the system to provide simpler explanations, as the default sum of θ is small. Alternatively, θ can be initialized to be close to 1 and followed by *backward* selection. However, we show in our experiments 3 that this approach is inferior.

3.2.3 Optimizing PERT. A major benefit of *learning to perturb* for explainability is that many behaviors of good explanations can be encouraged during optimization via a loss function. In this work, we learn the saliency values θ iteratively with respect to a novel loss function (Equation 6) that contains three key components, each of which works together to provide accurate and simple explanations for a black-box time series classifier. There are three components in our loss function: $L_{\text{preservation}}$, L_{budget} , and L_{TV}. The first component, $L_{\text{preservation}}$ encourages the perturbation function f_p to produce perturbations for which the black-box classifier makes the *same* prediction as it did for the instance-of-interest $\hat{X}$:

$$L_{\text{preservation}} = \lambda_1 \left(\frac{1}{\|\hat{X}\|} \sum_{t=0}^{T} (f_c(\hat{X}) - f_c(f_p(\hat{X}; \theta)))^2 \right) \quad (3)$$

where $f_c(\hat{X})$ is black-box classifier's predicted probability for the instance-of-interest $\hat{X}$ and $f_c(\hat{X}^*)$ is black-box classifier's prediction for the perturbed instance X^*. By minimizing the squared difference between these terms, we encourage f_p to *preserve* the prediction made on $\hat{X}$. To balance the goal of explainability and simplicity, we incorporate a L_{budget} on the saliency values, it encourages the final saliency values θ to be small with fewest possible timesteps:

$$L_{\text{budget}} = \lambda_2 \left(\frac{1}{\|\theta\|} \sum_{t=0}^{T} |\theta_t| \right) \quad (4)$$

Third, we add the intuition that neighboring timesteps and short contiguous subsequences should be roughly equal in importance. We achieve this through *Total Variance Normalization* [9], minimizing the squared difference between neighboring saliency values:

$$L_{\text{TV}} = \lambda_3 \left(\frac{1}{\|\theta\|} \sum_{t=0}^{T-1} (\theta_t - \theta_{t+1})^2 \right) \quad (5)$$

Finally, the loss terms are summed and each is associated with a coefficient to allow for re-scaling depending on task-specific preferences in explanation behavior. We scale the final loss function according to the sampling weights w_O and w_C.

The final loss for PERT is shown in Equation 6.

$$L(P(\hat{X}); \theta) = (L_{\text{preservation}} + L_{\text{budget}} + L_{\text{TV}}) * \frac{1}{2}(w_O + w_C) \quad (6)$$

We jointly optimize θ and w by minimizing $L(P(\hat{X}); \theta)$. To prioritize the replacement time series R_C and R_O which help minimize $L(P(\hat{X}); \theta)$, we use $L(P(\hat{X}); \theta)^{-1}$ to update the corresponding replacement sample weights w_O and w_C.

4 Experiments

4.1 Datasets

We evaluate our method on nine real-world time-series datasets: Wafer [26], GunPoint [30], Computers [5], Earthquakes [5], FordA [5], FordB [5], CricketX [5], PTB [11], ECG [26].

Each is a popular and publicly-available dataset for time series classification, and the summary statistics are provided in Table 1. For each dataset, we train a three-layered Fully-Connected Network (FCN) and a multi-node Recurrent Neural Network (RNN) to serve as black-box classifiers in need of explanations. Both models achieve nearly state-of-the-art performance for each task. Our experiments here show the results of both FCN and RNN networks. Each dataset comes with default train and test splits.

4.2 Compared Methods

We compare our proposed method, PERT, to one baseline and five state-of-the-art explanation methods for both FCN and RNN black-box models.

- *Random.* Each time step is assigned a random saliency value between -1 to 1 from a uniform distribution. This approach serves as a baseline for all methods.
- *RISE* [29]. The partial derivative of the black-box model's prediction $P(C|\hat{X})$ with respect to each time step is estimated empirically by randomly setting timesteps to zero and summarizing its impact on the $P(C|\hat{X})$.
- *LEFTIST* [12]. The partial derivative of $P(C|\hat{X})$ with respect to each time step is estimated empirically by randomly *replacing* the timestep with a corresponding value from a random instance from the background dataset and summarizing its impact on $P(C|\hat{X})$.
- *LIME* [32]. Saliency values are derived from the coefficients of a linear surrogate model, trained to mimic the behavior of the black-box model in the feature space surrounding $\hat{X}$. The success of this approach relies on the model behaving linear locally, which is rarely guaranteed in practice.
- *SHAP* [21]. SHAP assigns Shapley values [4] to each time step, thus computing their contributions to $P(C|\hat{X})$. Each time step is replaced by every value observed at the corresponding time step across all other instances in the background dataset.
- *Meaningful Perturbation (MP)* [9]. MP learns to perturb each time step such that $P(C|\hat{X})$ decreases. Perturbation is achieved by combining squared exponential smoothing with additive gaussian noise. Saliency values are then learned iteratively and are ultimately used as the final explanation.

4.3 Implementation Details

For each dataset, we train a three-layer FCN and a 10-node single layer RNN with GRU cells to serve as black-box time series classifiers in need of explanations. We train each model only on the training data, then explain their predictions for all testing instances using each compared explainability method. All reported metrics are the result of averaging over five runs to estimate the variance of the explainability methods.

We optimize our proposed method using Adam [19] with a learning rate of $1e^{-3}$ and train for 5000 epochs, which empirically achieves convergence. We used Weights & Biases [3] for experiment tracking and visualization. Our proposed method is implemented in PyTorch [28] and our code is publicly-available at https://github.com/kingspp/timeseries-explain.

4.4 Metrics

We use three key metrics to evaluate saliency maps for time series under the intuition that a prediction is well-explained if it accurately ranks the timesteps by their importance, as defined by changes in $P(C|\hat{X})$ *and* returns only the most-important timesteps.

Dataset	Wafer	GunPoint	Computers	Earthquakes	FordA	FordB	CricketX	PTB	ECG
Num. Train Instances	1000	50	250	322	3601	3636	390	1456	100
Num. Test Instances	6164	150	250	139	1320	810	390	1456	100
Num. Timesteps	152	150	720	512	500	500	300	187	96
FCN Accuracy (%)	99	99	80	75	96	92	81	98	98
RNN Accuracy (%)	99	99	79	75	96	92	80	98	98

Table 1: Summary statistics for the real-world datasets and the Accuracy of our corresponding FCN and RNN models.

Methods	Datasets								
	Wafer	GunPoint	Computers	Earthquakes	FordA	FordB	CricketX	PTB	ECG
Random	-0.02 (.01)	0.02 (.01)	0.01 (.01)	-0.01 (.01)	-0.03 (.01)	-0.01(.01)	-0.01 (.01)	0.06 (.04)	0.01 (.06)
RISE [29]	0.22 (.01)	0.16 (.01)	-0.01 (.02)	0.23 (.05)	0.15 (.02)	0.11 (.01)	0.42 (.01)	0.07 (.05)	0.14 (.07)
LEFTIST [12]	0.53 (.02)	0.15 (.03)	-0.16 (.01)	0.15 (.03)	0.16 (.01)	0.15 (.01)	-0.01 (.01)	0.51 (.01)	0.55 (.01)
LIME [32]	0.06 (.01)	0.10 (.01)	0.06 (.03)	-0.01 (.01)	0.01 (.02)	0.01 (.01)	-0.01 (.01)	0.18 (.07)	0.09 (.06)
SHAP [21]	-0.19 (.01)	-0.01 (.01)	0.10 (.01)	0.71 (.03)	0.23 (.01)	-0.17 (.01)	0.09 (.01)	-0.15 (.01)	-0.11 (.09)
MP [9]	0.49 (.01)	0.03 (.01)	0.15 (.01)	0.33 (.01)	0.48 (.01)	0.37 (.01)	0.43 (.01)	0.33 (.01)	-0.16 (.00)
PERT	**0.74 (.01)**	**0.56 (.01)**	**0.95 (.01)**	**0.93 (.01)**	**0.83 (.01)**	**0.82 (.01)**	**0.69 (.01)**	**0.58 (.01)**	**0.60 (.01)**

Table 2: Performance of the AUC-difference metric with the FCN black-box model. Parentheses indicate σ. Compared methods are separated into four groups: Random perturbation, linear surrogate model, game theory method, and learned perturbations.

AUC-Difference. Saliency maps can be evaluated by "inserting" or "deleting" timesteps from the time-series instance based on the derived importance map and observing the changes in the black-box model's predictions [29].

Intuitively, a good saliency map is one that has ranked timesteps such that when the most important timesteps are deleted, there is a sharp drop in the confidence of the model's prediction. This can be measured by computing the area under the *deletion* curve (AUDC) as timesteps are deleted one by one. A lower value for this area naturally indicates a better explanation. Analogously, insertion of a few important timesteps should result in the largest possible increase in the confidence of the model's prediction, thereby creating a large area under the *insertion* curve (AUIC). To ease comparisons, we merge these two measures into one metric by computing the difference, AUC-Difference = AUIC − AUDC. Ideally, the difference should be large (1.0), implying the saliency maps having a high AUIC (1.0) and a low AUDC (0.0) is a good explanation. This metric alleviates the need for human evaluation and annotation and makes it more fair and true to the classifier's own view on the problem. To achieve deletion during evaluation we replace a to-be-deleted timestep with the corresponding timestep from the opposing class's mean. Conversely, for insertion, we iteratively replace the mean time series from the opposing class with the original inputs of the instance-of-interest.

Confidence Suppression Game. Another popular approach to evaluating saliency maps is to find the smallest number of timesteps required to suppress the confidence of the black-box model by a given percentage [9]. We thus ask each explanation to play this *Confidence Suppression Game* and return the proportion of timesteps that must be deleted to reduce $P(C|\hat{X})$ by each value in a set of percentages. To achieve deletion, we once again replace timesteps with corresponding values from the opposing class mean.

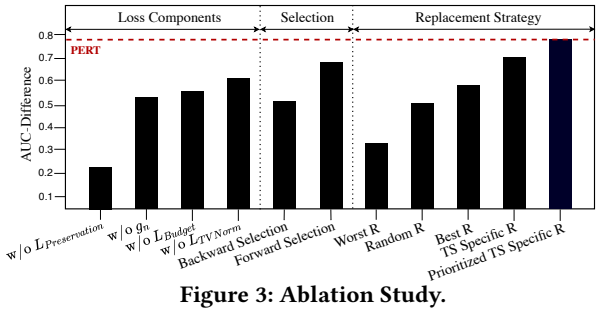


Figure 3: Ablation Study.

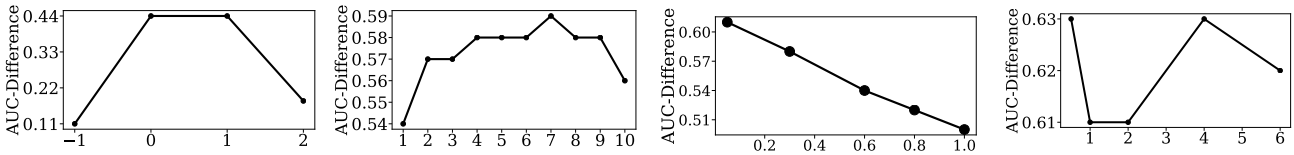
Minimality of Saliency Maps. A good explanation returns *only* the most important timesteps, thus allowing a user to consider only a few regions of the input. Thus we compute the *sum* of a saliency map, under the intuition that it should be small, as the goals of explainability and simplicity are jointly optimized, we expect importance of unimportant timesteps to go to zero leading to simpler explanations with fewest possible timesteps. To avoid naive solutions with very small and equally distributed saliency values, we report the normalized *variance* of the saliency map. A high variance indicates a larger difference between the minimum and maximum saliency values.

4.5 Experimental Results

4.5.1 PERT successfully finds the most-important timesteps. First, we measure how well all compared explainability methods have *ranked* the timesteps by their importance via the AUC-Difference metric discussed in Section 4.4. Our results using Fully-Connected

Methods	Datasets								
	Wafer	GunPoint	Computers	Earthquakes	FordA	FordB	CricketX	PTB	ECG
Random	0.01 (.01)	0.03 (.01)	0.01 (.01)	0.04 (.01)	0.01 (.01)	0.01(.01)	-0.01 (.01)	0.07 (.04)	0.01 (.06)
RISE [29]	0.13 (.01)	0.10 (.01)	-0.01 (.02)	0.23 (.05)	0.15 (.01)	0.11 (.02)	0.42 (.01)	0.10 (.05)	0.19 (.07)
LEFTIST [12]	0.16 (.01)	0.15 (.03)	-0.16 (.01)	0.53 (.03)	0.15 (.02)	0.15 (.01)	-0.10 (.01)	0.42 (.01)	0.51 (.01)
LIME [32]	0.07 (.01)	0.02 (.01)	0.05 (.03)	-0.02 (.01)	0.01 (.01)	0.01 (.01)	0.03 (.01)	0.12 (.07)	0.09 (.06)
SHAP [21]	-0.15 (.01)	-0.01 (.01)	0.10 (.01)	**0.80 (.03)**	0.23 (.01)	-0.17 (.01)	0.30 (.01)	-0.14 (.01)	0.08 (.09)
MP [9]	0.55 (.01)	0.02 (.01)	0.16 (.01)	0.30 (.01)	0.47 (.01)	0.39 (.01)	0.23 (.01)	0.30 (.01)	-0.15 (.01)
PERT	**0.78 (.01)**	**0.48 (.01)**	**0.92 (.01)**	**0.82 (.01)**	**0.70 (.01)**	**0.70 (.01)**	**0.68 (.01)**	**0.52 (.01)**	**0.57 (.01)**

Table 3: Average performance of the AUC-difference metric with the RNN black-box model.



(a) Mean g_n. **(b) $L_{\text{preservation}}$ coefficient (λ_1).** **(c) L_{budget} coefficient (λ_2).** **(d) L_{TV} coefficient (λ_3).**

Figure 4: PERT Hyperparameter Study

Network and Recurrent Neural Network can be found in Tables 2 and 3 respectively. In all cases, PERT significantly outperforms all other methods, with the largest improvement seen in the Wafer, Gun-Point, Computers, FordA, FordB, and CricketX datasets. This strongly indicates that the saliency maps produced by PERT have indeed ranked the timesteps far more accurately in terms of their individual effect on the black-box model's prediction confidence $P(C|\hat{X})$. We also note that in general, learning to perturb (PERT and MP) outperforms the random-replacement methods (Random, RISE, LEFTIST), linear surrogate method (LIME) and game theory method (SHAP), demonstrating the value in this class of perturbation-based explainability methods. When time series instances are short, such as for the GunPoint and Wafer datasets, RISE and SHAP produce comparable results to PERT as they rely on a densely-sampled feature space surrounding each instance-of-interest. For longer series, covering this space efficiently becomes extremely challenging, motivating *learned* perturbation [9]. We can see a similar pattern in results with FCN and RNN networks on the same datasets.

Next, we use the *Confidence Suppression Game* to measure the smallest number of timesteps that must be deleted to have the black-box model's prediction drop by a set of percentages. We report our findings for the Wafer, ECG, Computers, Earthquakes, Ford-A, Ford-B, and CriketX datasets in Table 4, and due to the space constraints the remainder of the results for PTB and GunPoint datasets are included with our publicly-available code, where we find the same trends. Overall, PERT successfully achieves the desired suppression levels by deleting far fewer timesteps than the compared methods. This indicates that PERT indeed finds the fewest and most important timesteps. As expected, all methods also outperform the *Random* baseline with very few exceptions. We also find the same trend to be true for the RNN classifier, the results for which are included with our publicly-available code.

Finally, we use *Minimality of Saliency* as a measure of simplicity of the derived saliency map. The results in Table 4 indicates that the saliency maps derived by PERT indeed highlight the fewest possible timesteps necessary to maintain f_c's prediction of $P(C|\hat{X})$ compared to state-of-the-art methods. The high variance of saliency maps demonstrates PERT's ability to avoid naive solutions with small saliency values.

4.5.2 Ablation Study. We next compare alternative approaches to some of PERT's key components with respect to the AUC-Difference metric, the results for which are shown in Figure 3. We target this study at three categories: loss components, selection criteria, and replacement strategies. First, we compare PERT's performance while removing different components of the loss function. As expected, $L_{\text{preservation}}$ clearly has the largest impact on the final AUC-Difference, as without this component, there is no relationship between the saliency map and the model's prediction. The remaining components g_n, L_{Budget} and L_{TVNorm} have less of an impact but still contribute to PERT's state-of-the-art performance.

Second, PERT uses *Forward Selection*, starting with small saliency values and progressively *increasing* them during training. However, another option is *Backward Selection*, where saliency values are initially large and are *decreased* during training. Interestingly, the *Forward Selection* is significantly better than *Backward Selection*, indicating that carefully *adding* time steps produces a better ranking whilst preserving the fewest possible timesteps. Third, we experiment with five alternative time series replacement strategies to test the importance of the choice of replacement time series R's impact on black-box model f_c's prediction $P(C|\hat{X})$. Initially we consider an exhaustive search to find a single *best* (and *worst*) R from the

		Confidence suppression game ↓										Minimality of Saliency	
Dataset	Method	10%	20%	30%	40%	50%	60%	70%	80%	90%	95%	Sum (%) ↓	$\|Variance\|$ ↑
Wafer [26]	Random	0.79	0.82	0.86	0.87	0.87	0.88	0.89	0.91	0.91	0.92	50	0.0016
	RISE [29]	0.57	0.57	0.58	0.59	0.59	0.59	0.61	0.61	0.61	0.61	82	0.0001
	LEFTIST [12]	0.56	0.57	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.59	78	0.0001
	LIME [32]	0.69	0.73	0.82	0.82	0.83	0.84	0.84	0.86	0.86	0.88	42	0.0001
	SHAP [21]	0.78	0.78	0.78	0.78	0.78	0.78	0.79	0.79	0.79	0.79	52	0.0004
	MP [9]	0.41	0.47	0.53	0.54	0.55	0.56	0.57	0.58	0.68	0.73	6	0.0041
	PERT	**0.40**	**0.40**	**0.46**	**0.47**	**0.48**	**0.49**	**0.51**	**0.53**	**0.55**	**0.59**	**1**	**0.0153**
ECG [5]	Random	0.40	0.41	0.41	0.43	0.43	0.43	0.44	0.45	0.45	0.45	50	0.0001
	RISE [29]	0.40	0.40	0.40	0.40	0.40	0.41	0.41	0.41	0.41	0.41	76	0.0001
	LEFTIST [12]	0.30	0.30	0.30	0.30	0.30	0.31	0.31	0.31	0.31	0.31	60	0.0001
	LIME [32]	0.71	0.72	0.74	0.77	0.79	0.81	0.81	0.84	0.85	0.64	42	0.0001
	SHAP [21]	0.62	0.62	0.62	0.63	0.63	0.63	0.63	0.64	0.64	0.64	64	0.0001
	MP [9]	0.51	0.51	0.51	0.52	0.52	0.52	0.52	0.52	0.53	0.53	3	0.0005
	PERT	**0.21**	**0.21**	**0.21**	**0.21**	**0.21**	**0.22**	**0.22**	**0.22**	**0.22**	**0.22**	**2**	**0.0034**
Computers [5]	Random	0.69	0.73	0.75	0.77	0.79	0.81	0.83	0.85	0.90	0.95	50	0.0001
	RISE [29]	0.92	0.92	0.93	0.93	0.93	0.94	0.94	0.94	0.94	0.94	46	0.0001
	LEFTIST [12]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	49	0.0001
	LIME [32]	0.60	0.62	0.65	0.68	0.7	0.71	0.73	0.75	0.78	0.82	23	0.0001
	SHAP [21]	0.99	0.99	0.99	0.99	0.99	1.00	1.00	1.00	1.00	1.00	51	0.0001
	MP [9]	0.75	0.76	0.78	0.81	0.81	0.82	0.82	0.83	0.84	0.88	4	0.0005
	PERT	**0.50**	**0.50**	**0.50**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**2**	**0.0017**
Earthquakes [5]	Random	0.97	0.97	0.98	0.98	0.99	0.99	0.99	0.99	1.00	1.00	50	0.0001
	RISE [29]	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	74	0.0001
	LEFTIST [12]	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1.00	1.00	61	0.0001
	LIME [32]	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.99	0.99	43	0.0001
	SHAP [21]	0.99	0.99	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	51	0.0001
	MP [9]	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	4	0.0001
	PERT	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**	**2**	**0.0039**
Ford-A [5]	Random	0.75	0.78	0.82	0.84	0.86	0.88	0.89	0.91	0.92	0.95	50	0.0001
	RISE [29]	0.76	0.76	0.76	0.76	0.76	0.76	0.77	0.77	0.77	0.78	55	0.0001
	LEFTIST [12]	0.78	0.78	0.79	0.80	0.81	0.82	0.83	0.84	0.84	0.85	49	0.0001
	LIME [32]	0.71	0.75	0.78	0.81	0.83	0.84	0.86	0.87	0.89	0.92	44	0.0001
	SHAP [21]	0.93	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	47	0.0001
	MP [9]	0.66	0.67	0.70	0.72	0.75	0.77	0.78	0.81	0.83	0.87	3	0.0010
	PERT	**0.63**	**0.64**	**0.65**	**0.66**	**0.66**	**0.67**	**0.67**	**0.68**	**0.69**	**0.72**	**2**	**0.0012**
Ford-B [5]	Random	0.73	0.74	0.77	0.79	0.81	0.83	0.84	0.85	0.86	0.88	50	0.0001
	RISE [29]	0.73	0.81	0.85	0.86	0.86	0.86	0.86	0.86	0.87	0.87	48	0.0001
	LEFTIST [12]	0.72	0.73	0.75	0.77	0.80	0.82	0.83	0.84	0.85	0.86	49	0.0001
	LIME [32]	0.71	0.72	0.74	0.77	0.79	0.81	0.81	0.84	0.85	0.87	30	0.0001
	SHAP [21]	0.71	0.72	0.74	0.75	0.77	0.81	0.81	0.82	0.83	0.84	40	0.0009
	MP [9]	0.71	0.71	0.73	0.74	0.76	0.77	0.78	0.79	0.82	0.83	4	0.0008
	PERT	**0.70**	**0.71**	**0.71**	**0.72**	**0.73**	**0.73**	**0.74**	**0.74**	**0.76**	**0.77**	**2**	**0.0010**
CricketX [24]	Random	0.21	0.27	0.35	0.41	0.49	0.61	0.72	0.89	0.93	1.00	50	0.0001
	RISE [29]	0.15	0.15	0.26	0.35	0.45	0.55	0.63	0.91	0.91	0.95	74	0.0001
	LEFTIST [12]	0.20	0.26	0.29	0.29	0.30	0.50	0.50	0.50	0.50	0.50	51	0.0001
	LIME [32]	0.14	0.19	0.23	0.26	0.29	0.31	0.34	0.37	0.41	0.48	39	0.0001
	SHAP [21]	0.19	0.29	0.33	0.36	0.38	0.39	0.40	0.41	0.43	0.45	33	0.0005
	MP [9]	0.09	0.11	0.12	0.15	0.17	0.18	0.19	0.21	0.21	0.25	2	**0.0040**
	PERT	**0.06**	**0.07**	**0.07**	**0.08**	**0.08**	**0.09**	**0.10**	**0.11**	**0.12**	**0.15**	**1**	**0.0037**

Table 4: Confidence suppression metric performance for seven key datasets with the FCN black-box model. Lower values are better for Saliency sum and Confidence suppression game, Higher values are better for Saliency variance. ↓ indicates *the lower the better.* ↑ indicates *the higher the better.*

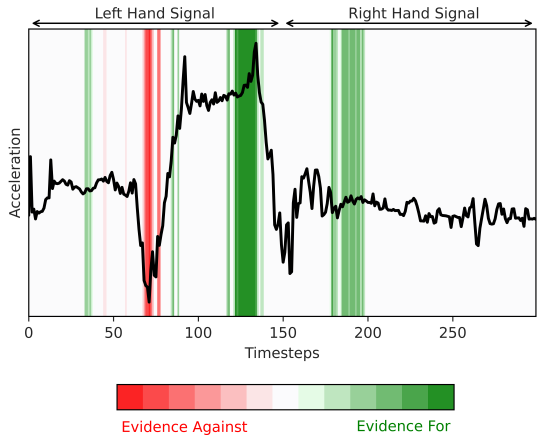


Figure 5: CricketX "No-Ball" Class Case Study. The signal is shown in black. Important time steps for the class-of-interest are shown in bright green and while bright red indicates important time steps for the opposing class.

background dataset $\mathcal{D}$. From this experiment, we notice that the AUC-Difference metric is sensitive to the choice of replacement time series R. However, by observing the relationship between the predicted confidence of f_c for instance-of-interest $\hat{X}$, $P(C|\hat{X})$ and for the perturbed instance X^*, $P(C|\hat{X}^*)$, we propose dual replacement sampling, where we sample a replacement time series from class-of-interest R_C and from opposing class R_O, and perform time step specific replacement which results in a substantial improvement in AUC-Difference metric, and when coupled with prioritization outperforms all other replacement strategies.

4.5.3 Hyperparameter Study. Producing explanations with PERT involves balancing the four key hyperparameters: The mean of Gaussian noise g_n and the $L_{\text{preservation}}$, L_{budget}, and L_{TV} coefficients.

We investigate the effects of tuning the coefficients of each in isolation in Figure 4 on Wafer dataset, keeping all unchanged parameters at their best-found values. As shown in Figure 4a, we change the mean of the gaussian noise from -1 to 2 and notice that the optimal value lies between 0 and 1, indicating that a small amount of additive gaussian noise leads to the largest improvement in PERT's performance. This confirms the intuition behind the compared method MP [9], which employs a similar strategy. Next, we find a non-linear trade-off for the $L_{\text{preservation}}$ coefficient: Too-low means not enough focus on producing good explanations, too-high begins to ignore the other crucial parts of the loss function, as shown in Figure 4b. Third, as expected, a higher L_{budget} leads to lower AUC-Difference, as shown in Figure 4c. This is because there is a trade-off between the L_{budget} and $L_{\text{preservation}}$, where the higher the coefficient on L_{TV} forces PERT to minimize the values of the saliency map during training, eventually ignoring the preservation task entirely. Finally, the results for the L_{TV} coefficient are shown in Figure 4d, where we observe that the final AUC-Difference is quite robust to these changes in the coefficient, ranging only between 0.61 and 0.63. Compared methods use default hyperparameters.

4.5.4 Case Study on the CricketX dataset. Finally, we provide a case study, using our proposed PERT method to explain the black-box model's prediction for one instance from the CricketX dataset [24]. CricketX contains length-300 time series and is a binary classification task: An umpire wearing a sensor on their wrists calls either *No Ball* or *Wide Ball* during a cricket match. The *No Ball* class has bursts of movement on the left side, indicating the umpire raised their left hand. The *Wide Ball* class has left-side *and* right-side bursts, indicating the umpire also raised their right hand. For this case study, we choose one instance from the *No Ball* class, which is known to be represented as as a spike in the left side of the time series. In Figure 5, we show the raw instance in black along with the saliency map produced by PERT colored according to the scale of the values in the saliency map. Intuitively, the brighter the colors, the more important the corresponding time series.

First, multiple bursts only on the left-side of the time series are an indication of strong evidence for the *No Ball* class, PERT accurately highlighted the highest burst in the brightest green region. Further, evidence for the *No Ball* class also appears on in the right-hand signal where there would have been a signal had it been *Wide Ball*. PERT's saliency map effectively highlights that the black-box classifier accurately used these important regions to predict *No Ball*. Additionally, this example gives us insight into the black-box model's prediction: The end of the left-hand signal (once the umpire's hand is raised and stabilized) is the strongest evidence supporting the model's decision. Second, bright red color bars indicate strong evidence for the opposite class had this been a *Wide Ball* (opposite class). This strongly indicates that the underlying model is using the information at these two regions far more than the others, exactly as expected for this example.

5 Conclusion

In this work, we identify the need for attribution-based explanations for deep neural network-based time series classifiers. We then design PERT, a method that learns to highlight the timesteps that are most responsible and the degree to which they are important for the classifier's prediction. By adaptively *learning* a perturbation function, PERT creates explanations for input time series that are catered specifically to the dataset, time series instance, and individual timesteps, deriving evidence both *for* and *against* the model's prediction. This evidence, presented as one importance value per timestep, thus allows an end-user to understand what information their classifier was using during classification, increasing their trust in their model. In our experiments, we conclusively demonstrate that PERT accurately discovers the most-important timesteps as it outperforms five state-of-the-art alternatives on three key metrics on nine datasets.

6 Acknowledgements

This research was supported by the U.S. Dept. of Education grant P200A180088, NSF grant 1910880, NSF CSSI: FAIN: 2103832, Oak Ridge Associated Universities CA W911NF-16-2-0008 and W911NF-20-2-0232. The research was performed using computational resources supported by the Academic and Research Computing group at Worcester Polytechnic Institute.

References

[1] Inès Arous, Ljiljana Dolamic, Jie Yang, Akansha Bhardwaj, Giuseppe Cuccu, and Philippe Cudré-Mauroux. Marta: Leveraging human rationales for explainable text classification. In *AAAI Conference on Artificial Intelligence*, 2021.

[2] J. Bento, P. Saleiro, F. Cruz, M. Figueiredo, and P. Bizarro. Timeshap: Explaining recurrent models through sequence perturbations. *ArXiv*, abs/2012.00073, 2020.

[3] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[4] A. Charnes, B. Golany, M. Keane, and J. Rousseau. Extremal principle solutions of games in characteristic function form: Core, chebychev and shapley value generalizations. In *Econometrics of Planning and Efficiency*, pages 123–133. Springer Netherlands, Dordrecht, 1988.

[5] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[6] Eoin Delaney, Derek Greene, and Mark T. Keane. Instance-based counterfactual explanations for time series classification. *ArXiv*, abs/2009.13211, 2020.

[7] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *CoRR*, abs/1809.04356, 2018.

[8] R. Fong, M. Patrick, and A. Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *ICCV*, pages 2950–2958, 2019.

[9] R. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *ICCV*, pages 3449–3457, 2017.

[10] Christopher Frye, Colin Rowat, and Ilya Feige. Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1229–1239. Curran Associates, Inc., 2020.

[11] A. Goldberger, L. A. Amaral, L. Glass, Jeffrey M. Hausdorff, P. Ivanov, R. Mark, J. Mietus, G. Moody, C. Peng, and H. Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation*, 101 23:E215–20, 2000.

[12] Maël Guillemé, V. Masson, Laurence Rozé, and A. Termier. Agnostic local explanation for time series classification. *ICTAI*, pages 432–439, 2019.

[13] Tadaaki H. Bankruptcy prediction using imaged financial ratios and convolutional neural networks. *Expert Systems with Applications*, 117:287 – 299, 2019.

[14] Shenda Hong, Yuxi Zhou, Junyuan Shang, Cao Xiao, and Jimeng Sun. Opportunities and challenges in deep learning methods on electrocardiogram data: A systematic review. *Computers in biology and medicine*, 122:103801, 2020.

[15] A. A. Ismail, M. Gunady, H. Bravo, and S. Feizi. Benchmarking deep learning interpretability in time series predictions. *ArXiv*, abs/2010.13924, 2020.

[16] Aya Abdelsalam Ismail, Mohamed Gunady, Hector Corrada Bravo, and Soheil Feizi. Benchmarking deep learning interpretability in time series predictions. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6441–6452. Curran Associates, Inc., 2020.

[17] Deepak A Kaji, John R Zech, Jun S Kim, Samuel K Cho, Neha S Dangayach, Anthony B Costa, and Eric K Oermann. An attention based deep learning model of clinical events in the intensive care unit. *PloS one*, 14(2):e0211057, 2019.

[18] D. Kale, D. Gong, Z. Che, Y. Liu, G. Medioni, R. Wetzel, and P. Ross. An examination of multivariate time series hashing with applications to health care. *ICDM*, pages 260–269, 01 2015.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[20] Sawan Kumar and P. Talukdar. Nile : Natural language inference with faithful natural language explanations. In *ACL*, 2020.

[21] Scott L. and Su-In L. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems 30*, pages 4765–4774, 2017.

[22] Tao Lei, R. Barzilay, and T. Jaakkola. Rationalizing neural predictions. In *EMNLP*, 2016.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

[24] A. Mueen, E.J. Keogh, and N E. Young. Logical-shapelets: an expressive primitive for time series classification. In *KDD*, pages 1154–1162, 2011.

[25] F. Mujkanovic, V. Doskoč, M. Schirneck, P. Schäfer, and T. Friedrich. timexplain–a framework for explaining the predictions of time series classifiers. *ArXiv*, abs/2007.07606, 2020.

[26] R. Olszewski, R. Maxion, and D. Siewiorek. Generalized feature extraction for structural pattern recognition in time-series data. *PhD thesis*, 2001.

[27] Paul A. Parker, Scott H. Holan, and Nalini Ravishanker. Nonlinear time series classification using bispectrum-based deep convolutional neural networks, 2020.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[29] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *CoRR*, abs/1806.07421, 2018.

[30] C. Ratanamahatana and Eamonn J. Keogh. Three myths about dynamic time warping data mining. In *SDM*, pages 506–510. SIAM, 2005.

[31] A. Ribeiro, M. Ribeiro, G. Paixão, D. Oliveira, P. Gomes, A. Canazart, P. Ferreira, C. Andersson, P. Macfarlane, and et al. Automatic diagnosis of the 12-lead ecg using a deep neural network. *Nature Communications*, 11(1), Apr 2020.

[32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016.

[33] Udo S., Hiba A., Mennatallah E., Daniela O., and Daniel A.K. Towards a rigorous evaluation of xai methods on time series. *CoRR*, abs/1909.07082, 2019.

[34] Udo S., Daniela O., Daniel A.K., and Mennatallah E. An empirical study of explainable ai techniques on deep learning models for time series tasks. *ArXiv*, abs/2012.04344.

[35] T. Schaul, John Quan, Ioannis Antonoglou, and D. Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2016.

[36] Cansu Sen, Thomas Hartvigsen, Biao Yin, Xiangnan Kong, and Elke Rundensteiner. Human attention maps for text classification: Do humans and neural networks focus on the same words? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[37] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153, 2017.

[38] Sima Siami-Namini and Akbar Siami Namin. Forecasting economics and financial time series: Arima vs. lstm. *arXiv preprint arXiv:1803.06386*, 2018.

[39] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[40] H. Song, D. Rajan, J. Thiagarajan, and A. Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 4091–4098. AAAI press, 2018.

[41] Nils Strodthoff, Patrick Wagner, Tobias Schaeffter, and Wojciech Samek. Deep learning for ecg analysis: Benchmarks and insights from ptb-xl, 2020.

[42] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328, 2017.

[43] Ahmed Tealab, Hesham Hefny, and Amr Badr. Forecasting of nonlinear time series using ann. *Future Computing and Informatics Journal*, 2(1):39–47, 2017.

[44] Sana Tonekaboni, Shalmali Joshi, David Duvenaud, and Anna Goldenberg. What went wrong and when? instance-wise feature importance for time-series models. *Advances in Neural Information Processing Systems*, 2020.

[45] Yanbo Xu, Siddharth Biswal, Shriprasad R Deshpande, Kevin O Maher, and Jimeng Sun. Raim: Recurrent attentive and intensive model of multimodal patient monitoring data. In *Proceedings of the 24th ACM SIGKDD international conference on Knowledge Discovery & Data Mining*, pages 2565–2573, 2018.

[46] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956. ACM, 2009.

[47] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.