Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Explaining Deep Multi-Class Time Series Classifiers

Ramesh Doddaiah
rdoddaiah@wpi.edu
Worcester Polytechnic Institute
Prathyush Parvatharaju
Worcester Polytechnic Institute
Elke Rundensteiner
Worcester Polytechnic Institute
Thomas Hartvigsen
Massachusetts Institute of Technology

Research Article

Keywords: Time Series, Classification, Explainability, Deep Learning

Posted Date: February 1st, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2531572/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Knowledge and Information Systems on March 4th, 2024. See the published version at https://doi.org/10.1007/s10115-024-02073-y.

Explaining Deep Multi-Class Time Series Classifiers

Ramesh Doddaiah[1*†], Prathyush S Parvatharaju[1†], Elke Rundensteiner[1] and Thomas Hartvigsen[2]

[1]Data Science, WPI, 100 Institute Rd, Worcester, 01609, Massachusetts, USA.
[2] CSAIL, MIT, 77 Massachusetts Ave, Cambridge, 02139, Massachusetts, USA .

*Corresponding author(s). E-mail(s): rdoddaiah@wpi.edu;
Contributing authors: psparvatharaju@wpi.edu; rundenst@wpi.edu; tomh@mit.edu;
†These authors contributed equally to this work.

Abstract

Explainability helps users trust deep learning solutions for time series classification. However, existing explainability methods for multi-class time series classifiers focus on one class at a time, ignoring relationships between the classes. Instead, when a classifier is choosing between many classes, an effective explanation must show what sets the chosen class apart from the rest. We now formalize this notion, studying the open problem of class-specific explainability for deep time series classifiers, a challenging and impactful problem setting. We design a novel explainability method, DEMUX, which learns saliency maps for explaining deep multi-class time series classifiers by adaptively ensuring that its explanation spotlights the regions in an input time series that a model uses specifically to its predicted class. DEMUX adopts a gradient-based approach composed of three interdependent modules that combine to generate consistent, class-specific saliency maps that remain faithful to the classifier's behavior yet are easily understood by end users. We demonstrate that DEMUX outperforms nine state-of-the-art alternatives on seven popular datasets when explaining two types of deep time series classifiers. We analyze run time performance, show the impacts of hyperparameter selection, and introduce a detailed study of perturbation methods for time series. Further, through a case

study, we demonstrate that DEMUX's explanations indeed highlight what separates the predicted class from the others in the eyes of the classifier.

Keywords: Time Series; Classification; Explainability; Deep Learning

1 Introduction

Background. Deep learning methods represent the state-of-the-art for tackling time series classification problems in important domains from finance [1] to healthcare [2, 3]. In these applications, we typically aim to classify a time series to be a member of one of many classes, referred to as *multi-class classification* in contrast to binary classification [4]. Despite the success of these deep learning methods, domain experts may not trust predictions from deep models as they are "opaque" and thus hard to understand. This lack of trust has the potential to hinder wide deployment of promising deep models for real-world applications [5]. By helping users better understand and thus trust their models, *explainability* has been recognized as a critical tool for successfully deploying deep learning models [6–13].

To explain a time series classification, we highlight the time steps that the model associates with the class it predicts via a saliency map [7, 9, 14]. However, in the multi-class problem setting where a classifier is choosing between many classes, an effective explanation must show *what sets the chosen class apart from the rest of the classes*. Thus the explanation should only highlight that particular subset of time steps that explains why that class was predicted compared to other classes. To further illustrate this with an example from computer vision, consider classifying images as either cats, dogs, foxes, or wolves. Fur is certainly evidence that an *animal* is in the image, but given choices between only *these* animals, fur is *specific* to no class. Therefore, an explanation should avoid highlighting fur [15]. In computer vision, it has been recognized as critical to leverage relationships between classes [15]. We conjecture that the same is equally true for time series, yet existing methods for time series ignore all relationships between classes [6, 8–10, 12]. Further, by ignoring this class-specificity issue, users may be prone to rationalize a model's prediction in the face of erroneous explanations, having been instilled with a false sense of confidence [16].

Motivating Example. Consider a deep multi-class time series classifier for ECG data [17] that classifies heartbeats into three classes: *Heart Attack*, *Cardiac Arrest*, and *Normal Heart*. Without knowing whether the classifier learned to recognize relevant critical regions of the time series for each class, a doctor may not trust its prediction. As shown in Figure 1 (middle), state-of-the-art methods would explain each class independently [7, 8, 12, 13]. However, by ignoring the relationships between classes, it appears as if the model uses most of the signal when predicting its top class, *Heart Attack*, even when it shares the same critical regions with Cardiac Arrest and Normal Heart classes as shown

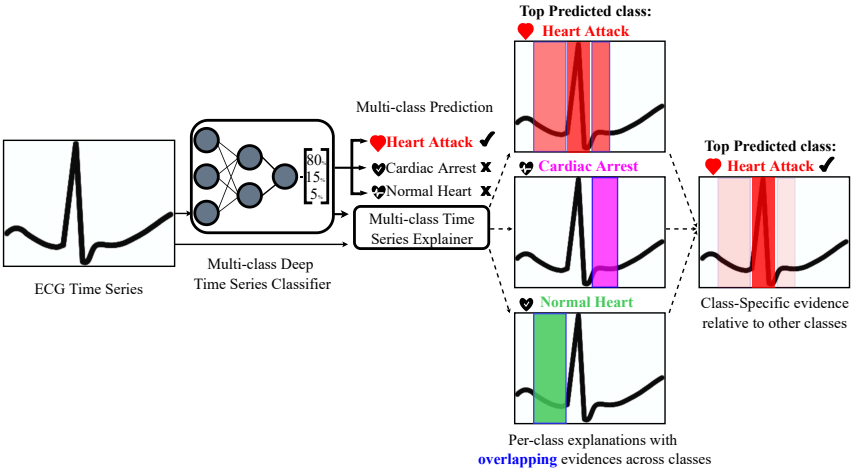


Fig. 1: Effective explanations of multi-class time series classifiers should show what regions of a time series set that class apart from the rest. Here, we show a deep classifier that predicts *Heart Attack* given an ECG. Explanations for individual classes (middle column) may have overlapping regions. However, as shown on right, the middle of the signal is what sets *Heart Attack* apart according to the classifier.

in blue. A class-specific explanation, on the other hand, should correct for the model using the same regions for multiple classes. In our example, it (rightmost column) explains that the burst (non-overlapping critical region) of the signal is what the model uses to predict *Heart Attack* instead of other classes.

State-of-the-Art. Explainablility for time series models has recently emerged as a promising direction to help users trust deep time series classifier models [6–10, 12, 18]. The most successful methods *learn to perturb* input time series to explain an opaque model's behavior in the vicinity of one instance [7, 8]. Intuitively, time steps that have a higher impact on model accuracy will be ranked higher. Most existing methods [5, 8, 9, 12, 13, 18] explain model behavior by perturbing each time step using either static, predefined values like zero or other time series instances from a "background" dataset. For example, PERT [7], which explains only binary deep time series classifiers, perturbs each time step by replacing it from a replacement time series sampled from the background dataset. DYNAMASK [8, 19], which also treats each class independently, uses static replacement strategies for each time step for deriving explanations for a multivariate classifier. It makes a binary decision if a feature is important or not. To-date, class-specific explanations, despite their recognized need in fields like computer vision [15], remain an open problem in time series. Typically, a successful multi-class classifier assigns high probability to one of the classes and lower probabilities to the rest. Evidence derived to explain the predicted class should be unique to that class, relative to other

classes. But existing time series explainable methods fail to incorporate the knowledge about relationships between classes.

Beyond lacking class-specificity, another well-known disadvantage of perturbation-learning methods is the high variance between explanations derived over multiple runs for the same time series instance as input [19–21]. High variability among explanations decreases a user's trust in an explainability method and must therefore be reduced.

Problem Definition. We study the open problem of *Class-Specific Explainability for Multi-Class Time Series Classifiers*: given a time series and a pre-trained multi-class classifier, we aim to generate a *class-specific saliency map* for the classifier's predicted class. A saliency map is a vector with one element per time step in the time series instance, where higher values of an element indicates a higher importance of this time step according to the classifier. To be *class-specific*, the saliency map should assign high importance only to time steps uniquely important to the predicted class (in contrast to also being important to other classes). This problem has multiple possibly conflicting objectives: a good saliency map should be class-specific, highlight only the most-relevant time steps, and still remain faithful to the model's behavior.

Challenges. Our problem is challenging for several reasons:

- *Class-Specificity*: Generating class-specific saliency maps requires knowledge of explanations across all classes. However, learning concurrently multiple explanations is hard, in particular for low-probability classes, with a model's predictions often highly variable in regions of low probability.
- *Local Fidelity*: We consider multi-class classifiers that predict probability distributions. Learning perturbations to explain these models must incorporate all class probabilities to remain faithful to the classifier's behavior. However, minor changes to the input can have a large effect on the predicted class distribution.
- *Temporal Coherence*: Time steps often depend on their neighbors' values. This implies that similarly for saliency maps neighboring time steps should have similar importance. While this encourages discovering important subsequences, thereby improving explainability, it conflicts with local fidelity and class-specificity. Hence, a trade-off must inherently be considered in any effective solution.
- *Consistent Saliency:* Perturbation-based explainability methods can create saliency maps that vary dramatically for the same instances when re-initialized. Yet to be useful in real-world applications, we should instead consistently generate similar explanations for the same time series

Proposed Solution. To derive class-specific explanations, we propose ***D**istinct T**E**mporal **MU**lticlass E**X**plainer (DEMUX)*, a novel model-agnostic, perturbation-based explainability method for multi-class time series models. DEMUX jointly learns saliency maps, with a focus on removing shared salient regions to generate a class-specific explanation for the model's top predicted class.

DEMUX is gradient-based approach that monitors changes in the classifier's predictions while perturbing values at each time step. It produces a saliency map for the classifier's top predicted class that preserves the classifier's prediction probability distribution across classes. To generate good perturbations, DEMUX learns to sample a replacement time-series per class from a background dataset using a clustering-based replacement selector. DEMUX avoids out-of-distribution replacement values by ensuring perturbations are like other time series the model has seen before for each class and for each time step, leading to more stable saliency maps.

This work extends our initial paper [22] in the following ways. We perform all experiments on two additional datasets. We then conduct an extensive hyperparameter study and a run-time analysis of all methods to provide a more comprehensive picture of DEMUX's performance. The new results in these experiments further demonstrate DEMUX as a robust and practical approach to explaining deep multi-class time series classifiers.

Contributions. Our main contributions are as follows:

- We identify and characterize the problem of class-specific saliency maps for deep multi-class time series classifiers.
- We introduce the first effective solution, DEMUX, which extends beyond recent work with three innovations: learning to remove shared saliency across classes (Class-Specificity), generating class-specific perturbations that are locally faithful (Local Fidelity and Temporal Coherence), and ensuring stability of saliency maps (Consistent Saliency) for given time series instances.
- Using seven real datasets, we conclusively demonstrate that DEMUX outperforms nine state-of-the-art alternatives, successfully generating class-specific explanations for two types of deep time-series classifiers.

2 Related Works

Deep learning models have achieved remarkable results in domains from time series forecasting to classification tasks. But these models are opaque in nature. Given that a substantial amount of data collected in high-stake domains like healthcare, [2, 3] finance [23] and autonomous vehicles [24, 25] are in the form of time series, the need to build explainable AI (XAI) for the time series domain is rapidly increasing. *Saliency maps* [19, 21] are among the many promising approaches to increasing transparency of a deep learning model. They correspond to importance scores that highlight the series regions contributing to the classifier's predictions. Below we characterize XAI methods according to the strategy they utilize to generate importance scores representing model explanations.

Perturbation methods. These methods [7, 8, 19] slightly change the input time series and compare the output to the baseline to create an importance ranking. The resulting saliency masks indicate importance of the time steps, explaining prediction for the given test instance. In time series, each time

step can have a range of valid or in-distribution values. Assuming static baselines replacement values or random time series from background datasets are good enough for each class-specific replacement tends to lead to sub-optimal importance scores for saliency maps.

Surrogate models. These methods [6, 12, 26] produce linear surrogate models that are trained to emulate the behavior of an opaque model locally for a set of perturbed versions of a given instance. If the linear model is a good enough local approximation, then its coefficients are treated as an explanation. Intuitively, if there is not a clear linear relationship between the set of perturbed instances and the opaque model's predictions, which is often the case in complex time series deep learning models, a linear model's coefficients fall short of generating a good explanation for a non-linear model's predictions.

Other methods. There are some popular methods [9, 11, 13, 18] that do not fall in one of the above categories. SHAP [13] does random permutations of the input features and average the marginal contribution of features to build importance Shapley scores. TIMESHAP is built upon KernelSHAP and extends it to the time series domain.

Unfortunately, none of the above category of methods address class-specific explanations challenges i.e., there has been little work developing explainability methods specifically for *multi-class* setting in time series, despite the recognition of the class-specific explainability importance in computer vision domain [15, 19]. Without leveraging knowledge from other classes during multi-class explanation learning, explanations from existing methods are less effective [15]. The methods mentioned above do not focus on generating *class-specific* perturbations and consistent explanations. Nor do they preserve the opaque model's prediction probability distribution across the classes while generating explanations. To the best of our knowledge, our work is the first to study learnable class-specific perturbations to explain opaque multi-class time series classifiers.

3 Methodology

3.1 Problem Definition

Assume we are given a set of N time series $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and a deep multi-class time series classifier $f : \mathbb{X} \to \mathcal{Y}$, where $\mathbb{X} \in \mathbb{R}^T$ is the T-dimensional feature space with and $\mathcal{Y} = \{1, 2, \ldots, C\}$ is the label space with C classes. Let us consider an instance-of-interest $\mathbf{x} \in \mathcal{D}$, a time series with T time steps along with a predicted probability distribution $\hat{\mathbf{y}} = f(\mathbf{x})$ over C classes where $\hat{\mathbf{y}} \in [0, 1]^C$ and $\sum_{i=1}^{C} \hat{\mathbf{y}}_i = 1$. The top predicted class z where $z = \text{argmax}\, \hat{\mathbf{y}}$ has predicted confidence $\hat{\mathbf{y}}_\mathbf{z}$ and is the class for which we seek an explanation.

Our goal is to learn a class-specific saliency map $\boldsymbol{\theta} \in [-1, 1]^T$ for the class of interest z where each element represents the importance of a corresponding time step. Positive values in $\boldsymbol{\theta}$ indicate evidence *for* class z, while negative values indicate evidence *against* class z.

We follow the lead of recent work on *perturbation-based* explanations for time series [7, 8], and assume the importance of a time step t should

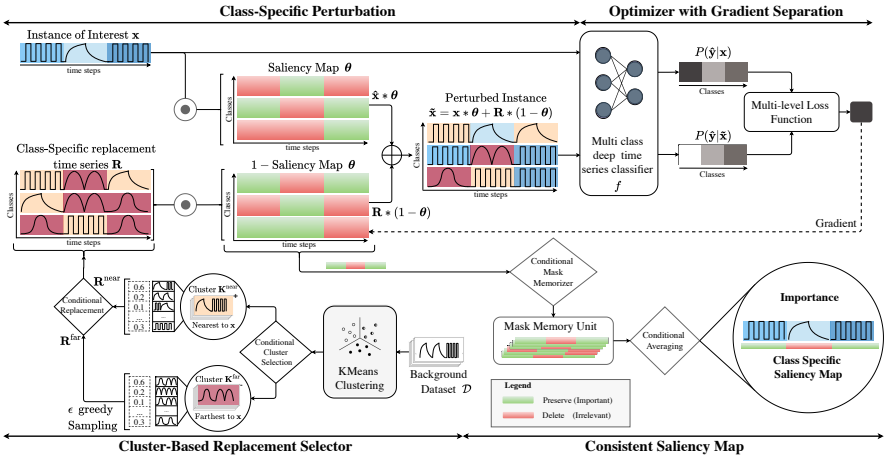


Fig. 2: DEMUX Architecture. Three model components collaborate to produce a saliency map for the predicted class. 1) The Cluster-Based Replacement Selector picks replacement time series. 2) The Class-Specific Perturbation Function uses these replacements to perturb the input. 3) The Mask Memory Unit produces Consistent Saliency Map.

reflect the expected scale of the change of $P(\hat{\mathbf{y}}_z|\mathbf{x})$ when $\mathbf{x}$ is perturbed: $|P(\hat{\mathbf{y}}_z|\mathbf{x}) - P(\hat{\mathbf{y}}_z|\tilde{\mathbf{x}})|$, where $\tilde{\mathbf{x}}$ is perturbed version of $\mathbf{x}$. A successfully learned saliency map $\boldsymbol{\theta}$ will assign high values to the regions in a time series where perturbations dramatically shift the model's predictions away from z. To ensure that explanations are simple and intuitive, we favor contiguous subsequences within the saliency map, indicating that neighboring time steps likely have similar importance, and that $\boldsymbol{\theta}$ be sparse if possible, assigning no importance to unimportant time steps.

3.2 Proposed Method: DEMUX

We propose the Distinct Temporal Multi-class Explainer (DEMUX), the first method to produce class-specific explanations for deep multi-class time series classifiers. DEMUX is model agnostic, so can explain any multi-class deep time series classifier f's predictions for an instance $\mathbf{x}$. By learning to discourage overlapping salient time steps between classes, DEMUX produces saliency maps that highlight the important class-specific time steps. As illustrated in Figure 2, DEMUX uses a novel class-specific replacement strategy to perturb time series and explain f's behavior. DEMUX's learning strategy encourages $\boldsymbol{\theta}$ to abide by a range of attractive behaviors and is accomplished via the design of a novel loss function that leads to explanations that are *unique* to the predicted class, *simple* and *easy* to understand, while being *consistent* between multiple runs.

DEMUX contains three key components that work together: (1) the *Cluster-Based Replacement Selector* learns which time series from the background

dataset $\mathcal{D}$ are the best for replacement-based perturbation. (2) The *Class-Specific Perturbation Function* learns a saliency map $\boldsymbol{\theta}$ for the top predicted class z relative to other classes by perturbing $\mathbf{x}$ and discovering the impact of each time step on f's prediction probability distribution $P(\hat{\mathbf{y}}|\mathbf{x})$ across all classes. (3) A *Mask Memory Unit* to *derive consistent Saliency Maps* which is achieved by building a repository of good saliency maps during learning that matches the predicted probability distribution $P(\hat{\mathbf{y}}|\mathbf{x})$ of f.

Cluster-Based Replacement Selection. Perturbation alters the values of the instance-of-interest $\mathbf{x}$, generating a synthetic time series $\tilde{\mathbf{x}}$. However, choosing which time steps to alter and the scale of the modifications can be highly impactful. In fact, dramatically changing $\mathbf{x}$ often creates an out-of-distribution perturbation, on which f's predictions are untrustworthy. PERT [7] overcomes this challenge by creating in-distribution perturbations via replacement time series $\mathbf{r}$, which replace time steps in $\mathbf{x}$ with time steps from background instances in $\mathcal{D}$. Similarly, MixUp [27] is a useful data augmentation strategy for computer vision, which interpolates between training instances. In this work, perturbations generated by mixing instances often results in in-distribution data. Inspired by PERT, we select a pair of time series from the background dataset $\mathcal{D}$ to provide evidence both *for* and *against* the class of interest z. However, multi-class replacement time series sampling is non-trivial as there can be many classes to choose from.

The best replacement time series may also vary by time step and per class, so we select replacement series $\mathbf{r}$ on a per-class and time step-by-time step basis. To provide evidence both *for* and *against* the class of interest z, the *Cluster-Based Replacement Selector* employs a clustering on $\mathcal{D}$—we use *KMeans* with *Kneedle* [28] to find a suitable number of clusters k for the given dataset $\mathcal{D}$ in our experiments. Thereafter, for each class, the Cluster-Based Replacement Selector chooses two clusters of time series representatives $\mathbf{K}^{\text{near}}$ and $\mathbf{K}^{\text{far}}$ from these k clusters. $\mathbf{K}^{\text{near}}$ denotes a cluster nearest to $\mathbf{x}$ and $\mathbf{K}^{\text{far}}$ a cluster furthest from $\mathbf{x}$ based on the *Euclidean* distance between $\mathbf{x}$ and the centroid of the two clusters, respectively. Then, we pick one replacement time series from $\mathbf{K}^{\text{near}}$ and one from $\mathbf{K}^{\text{far}}$

$$\mathbf{r}^{\text{far}} := \begin{cases} \mathbf{r}^{\text{far}} & \text{with probability } 1-\epsilon \\ \text{random } \mathbf{r} \in \mathbf{K}^{\text{far}} & \text{with probability } \epsilon \end{cases} \tag{1}$$

where $\mathbf{r}^{\text{far}}$ is one time series sampled from $\mathbf{K}^{\text{far}}$ with probability ϵ. We also sample $\mathbf{r}^{\text{near}}$ from the nearest cluster $\mathbf{K}^{\text{near}}$. We repeat this process for each of the C classes, accumulating the replacement series into respective matrices $\mathbf{R}^{near}, \mathbf{R}^{far} \in \mathbb{R}^{C \times T}$, denoting two replacement time series $\mathbf{r}^{near}$ and $\mathbf{r}^{far}$ per class. We employ epsilon greedy-based prioritized sampling [29] to explore and learn which replacement time series sampled from $\mathcal{D}$ are best and are accordingly weighted w for instance-of-interest $\mathbf{x}$.

To avoid disproportionate impact of large weights w early in training, we follow the literature [30] and employ the standard ϵ-greedy approach to balance

exploration and *exploitation* while DEMUX is being trained. Thus when ϵ is large, replacement series are sampled at random (*explore*). When ϵ is small, weights w are used to select the replacement series (*exploit*). To encourage early exploration and later exploitation, ϵ is exponentially decayed during training.

Class-Specific Perturbation. Next, we derive a class-specific saliency map $\boldsymbol{\theta}$ for z, the top predicted class, relative to other classes in $\hat{\mathbf{y}}$. We perturb the instance of interest $\mathbf{x}$ by performing time step and class-specific replacement using replacement time series matrices and a saliency map $\boldsymbol{\theta}$. Equation 2 shows a class-specific perturbation where $\mathbf{x}_i^*$ is the perturbed instance for the i-th class. $\mathbf{x}_i^*$ is passed to classifier f to observe the effects of the perturbation on f's predicted distribution $P(\hat{\mathbf{y}}|\mathbf{x})$.

We *learn* a perturbation function $\Phi : \mathbb{R}^{C\times T} \rightarrow \mathbb{R}^{C\times T}$ where the key component of Φ is a two-dimensional parameterized matrix $\boldsymbol{\theta} \in [-1, 1]^{C\times T}$. An element $\theta_{i,t}$ of this parameterized matrix represents the importance value of time step t for class i relative to other classes. Values in $\boldsymbol{\theta}$ close to 1 indicate strong evidence *for* the respective class i. Values close to -1 indicate evidence *against* the class i. Values near 0 imply no importance.

To consider both evidence *for* and *against* each class, our perturbation function Φ adaptively learns to replace values at each time step with replacement time series matrices. We thus replace the time step with $\mathbf{R}^{\text{near}}$, the representative of time series from the cluster nearest to $\mathbf{x}$ for $\theta_{i,t} < 0$. Similarly, when $\theta_{i,t} \geq 0$, $\mathbf{x}$ is replaced with the corresponding time step of $\mathbf{R}^{\text{far}}$, the representative of time series from the cluster farthest to $\mathbf{x}$. This way, DEMUX learns the degree of sensitivity of each time step per class. The function Φ generates perturbation $\tilde{\mathbf{x}}$ by performing time step-specific interpolation for each class:

$$\tilde{\mathbf{x}} = \boldsymbol{\theta}_i \odot \mathbf{x} + (1 - \boldsymbol{\theta}_i) \odot (\mathbb{1}_{\boldsymbol{\theta}_i<0} \odot \mathbf{R}_i^{\text{near}} + \mathbb{1}_{\boldsymbol{\theta}_i\geq 0} \odot \mathbf{R}_i^{\text{far}}) + g \quad (2)$$

Here, $\mathbb{1}_{\boldsymbol{\theta}_i<0}$ is a matrix-wise indicator function that returns 1 for elements of $\boldsymbol{\theta}_i$ that are less than 0 and $\mathbb{1}_{\boldsymbol{\theta}_i\geq 0}$ returns 1 for elements of $\boldsymbol{\theta}_i$ greater than or equal to 0. $\odot$ is the Hadamard product. For readability, we refer to this operation as $\Phi(\mathbf{x};\ \boldsymbol{\theta})$. Inspired by [19], We add a small amount of Gaussian noise g to avoid over-fitting $\boldsymbol{\theta}$ to extremely specific values. Using Equation 2, the final values of $\tilde{\mathbf{x}}$, the perturbed version of $\mathbf{x}$, are thus interpolations between the original time steps of $\mathbf{x}$ and the replacement series $\mathbf{R}_i^{\text{near}}$ or $\mathbf{R}_i^{\text{far}}$ where i is the i-th class in $\hat{\mathbf{y}}$ according to the scale of the corresponding value in $\boldsymbol{\theta}_i$. The conditional interval-specific operations on $\boldsymbol{\theta}$ are necessary to derive evidence both *for* and *against* class z.

The intuition behind adaptive time step specific interpolation is given by the three possible relationships between $P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\mathbf{x})$ and $P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\tilde{\mathbf{x}})$:

- $\boldsymbol{P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\mathbf{x}) < P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\tilde{\mathbf{x}})}$. After perturbing $\mathbf{x}$, the model's confidence in the top predicted class z *decreases*. In this case, we replace $\mathbf{x}$ with $R_{i=z}^{far}$, the corresponding value at the same time step of an instance from the furthest cluster. This way, the value is shifted towards signals from a different class. If

the values in $\mathbf{x}$ are crucial to predicting $P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\mathbf{x})$, their replacement should result in a drop in $P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\mathbf{x})$.

- $\boldsymbol{P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\mathbf{x}) = P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\tilde{\mathbf{x}})}$. Perturbing $\mathbf{x}$ has no impact on the classifier's prediction $P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\mathbf{x})$. This case indicates that inputs and prediction are independent. We find this is rare in practice.
- $\boldsymbol{P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\mathbf{x}) > P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\tilde{\mathbf{x}})}$. After perturbing $\mathbf{x}$, the model's confidence in the top predicted class z *increases*. In this case, we replace the values of $\mathbf{x}$ with $R^{near}_{i=z}$, the corresponding value of a time series sampled from the *nearest* cluster. This encourages $P(\hat{\mathbf{y}}_{\mathbf{i=z}}|\mathbf{x})$ to remain high, thereby accurately identifying values needed to predict the top class.

We initialize the values of $\boldsymbol{\theta}$ uniformly between $-1e^{-2}$ and $1e^{-2}$ to encode no prior assumptions about which time steps are most important. $\boldsymbol{\theta}$ is then updated iteratively throughout the training. We thus start with small values for $\boldsymbol{\theta}$ and effectively perform *forward selection* using gradient descent, steadily inserting only the most crucial timesteps to the prediction $P(\hat{\mathbf{y}}|\mathbf{x})$. This way, we encourage DEMUX to provide simpler explanations, as the default sum of $\boldsymbol{\theta}$ is small.

Learning Class-Specific Explanations. Class-specific saliency values $\boldsymbol{\theta} \in \mathbb{R}^{C\times T}$ are learned using a novel loss function (Equation 8) containing five key components. We optimize the loss function to derive a simple instance-specific explanation for the predicted distribution $P(\hat{\mathbf{y}}|\mathbf{x})$ for $\mathbf{x}$.

First, we encourage *Class-Specificity*: Overlapping salient regions between classes should be minimized. We design an objective called *shared saliency deletion* L_{SSD}, which learns to remove shared saliency amongst the top predicted class z and the rest of the classes, proportional to the respective class probability. This uniquely differentiates the top class salient time steps from the other classes. DEMUX learns unique class-specific saliency map $\boldsymbol{\theta}_z$ for the top predicted class z by altering $\boldsymbol{\theta}_z$ with respect to the corresponding importance values at the same time steps from other classes. Consider $\theta_{z,t}$ where t is the time step, if the importance value for time step t is high for multiple classes, it is considered to be a shared salient time step. The importance value $\theta_{z,t}$ is reduced by the importance value of time step t from $\boldsymbol{\theta}_{i,t}$ where $i \neq z$ represents a different class, and is weighted by the respective class confidence $P(\hat{\mathbf{y}}_i|\mathbf{x})$ to counter the influence of least confidence classes. Intuitively, L_{SSD} improves the importance of unique salient time steps and reduces the importance of shared salient time steps.

$$L_{SSD} = \lambda_5 \left(\frac{1}{|\mathcal{Y}|} \left(\boldsymbol{\theta}_z - \left| \sum_{i \neq z}^{\mathcal{Y}} (\boldsymbol{\theta}_z - P(\hat{\mathbf{y}}_i|\mathbf{x}) * \boldsymbol{\theta}_i) \right| \right)^2 \right) \tag{3}$$

Second, to derive instance-specific explanations from f, it is crucial to preserve *Local Fidelity* by preserving $P(\hat{\mathbf{y}}|\mathbf{x})$ for $\mathbf{x}$. Any changes to the predicted probability distribution $P(\hat{\mathbf{y}}|\mathbf{x})$ creates a disconnect between the derived explanation $\boldsymbol{\theta}_z$ and instance-of-interest $\mathbf{x}$. The second component of our loss function, L_{Prev}, encourages the perturbation function Φ to produce perturbations for

the top class z while preserving the distribution $P(\hat{\mathbf{y}}|\mathbf{x})$ for $\mathbf{x}$. To achieve this preservation, we use the Kullback-Liebler (KL) Divergence (Equation 4) to measure how the predicted distribution for the perturbed instance $P(\hat{\mathbf{y}}|\tilde{\mathbf{x}})$ differs from the reference distribution $P(\hat{\mathbf{y}}|\mathbf{x})$.

$$L_{\text{Prev}} = \lambda_1 \left((f(\mathbf{x}) - f(\Phi(\mathbf{x};\, \boldsymbol{\theta}_z)))_{KL} \right) \tag{4}$$

Third, to generate preserving low probabilities is a challenge as there are many possible perturbations that result in a low probabilities. Thus, for classes other than highest confidence class z we encourage the saliency map $\boldsymbol{\theta}$ to highlight the time steps that are responsible for maximizing the specific-class probability represented by Equation 5. Intuitively, we *preserve* the confidence for the top predicted class z, to derive explanations specific to instance-of-interest $\mathbf{x}$. For the rest of the classes, we maximize the confidences to find the minimal salient time steps that causes the prediction probability for the respective class to increase significantly. By making use of *gradient separation* and *multi-objective loss*, we learn saliency maps for z and the other classes distinctively and in parallel.

$$L_{\text{Max}} = \lambda_2 \left(\frac{1}{|\mathcal{Y}|} \sum_{i \neq z}^{\mathcal{Y}} (1 - f(\Phi(\mathbf{x};\, \boldsymbol{\theta}_i))) \right) \tag{5}$$

Fourth, to encourage simple explanations, with minimal and non-redundant salient time steps per class, we incorporate the L_{Budget} loss per class, similar to PERT [7] and DYNAMASK [8] methods. This component encourages the values of $\boldsymbol{\theta}$ to be as small as possible; intuitively, values that are not important should be close to 0 per our problem definition.

$$L_{\text{Budget}} = \lambda_3 \left(\frac{1}{|\mathcal{Y}|} \sum_{i=0}^{\mathcal{Y}} \left(\frac{1}{|T|} \sum_{t=0}^{T} |\theta_{i,t}| \right) \right) \tag{6}$$

Fifth, we encourage the saliency map to be *Temporally Coherent*: neighboring time steps should generally have similar important. To achieve this coherence, we add a time series regularizer L_{TReg} per class that minimizes the squared difference between neighboring saliency values in turn generating discriminative sub-sequences:

$$L_{\text{TReg}} = \lambda_4 \left(\frac{1}{|\mathcal{Y}|} \sum_{i=0}^{\mathcal{Y}} \left(\frac{1}{|T|} \sum_{t=0}^{T-1} (\theta_{i,t} - \theta_{i,t+1})^2 \right) \right) \tag{7}$$

Finally, all the loss terms are summed, each being scaled by a λ coefficient to balance the components depending on task-specific preferences in explanation behavior. Minimizing the total loss in Equation 8 leads to simple, class-specific saliency maps.

$$L(P(\hat{\mathbf{y}}|\mathbf{x});\, \boldsymbol{\theta}) = L_{\text{SSD}} + L_{\text{Prev}} + L_{\text{Max}} + L_{\text{Budget}} + L_{\text{TReg}} \tag{8}$$

Algorithm 1 DEMUX Algorithm

Require: $\mathbf{X}, \hat{\mathbf{y}}, z = \text{argmax}\,\hat{\mathbf{y}}, f, \mathcal{D}$
Ensure: Class-Specific saliency map $\boldsymbol{\theta}$ for class z

$\boldsymbol{\theta} \leftarrow$ random_uniform(low=-1e-2, high=1e-2)
for epoch in range(epochs); **do**
  for i in range(C classes); **do**
    $\mathbf{R}^{\text{near}}, \mathbf{R}^{\text{far}}, \mathbf{w}^{\text{near}}, \mathbf{w}^{\text{far}} \leftarrow$ Cluster-Based Replacement Selector$(\mathbf{X}, \mathcal{D})$
    $\tilde{\mathbf{x}} \leftarrow \boldsymbol{\theta}_i \odot \mathbf{x} + (1 - \boldsymbol{\theta}_i) \odot (\mathbb{1}_{\boldsymbol{\theta}_i < 0} \odot \mathbf{R}_i^{\text{near}} + \mathbb{1}_{\boldsymbol{\theta}_i \geq 0} \odot \mathbf{R}_i^{\text{far}}) + g$
    $L_{\text{Prev}} \leftarrow \lambda_1 \left((f(\mathbf{x}) - f(\Phi(\mathbf{x};\ \boldsymbol{\theta}_z)))_{KL} \right)$
    $L_{\text{Max}} \leftarrow \lambda_2 \left(\frac{1}{|\mathcal{Y}|} \sum_{i \neq z}^{\mathcal{Y}} (1 - f(\Phi(\mathbf{x};\ \boldsymbol{\theta}_i))) \right)$
    $L_{\text{Budget}} \leftarrow \lambda_3 \left(\frac{1}{|\mathcal{Y}|} \sum_{i=0}^{\mathcal{Y}} \left(\frac{1}{|T|} \sum_{t=0}^{T} |\theta_{i,t}| \right) \right)$
    $L_{\text{TReg}} \leftarrow \lambda_4 \left(\frac{1}{|\mathcal{Y}|} \sum_{i=0}^{\mathcal{Y}} \left(\frac{1}{|T|} \sum_{t=0}^{T-1} (\theta_{i,t} - \theta_{i,t+1})^2 \right) \right)$
    $L_{SSD} \leftarrow \lambda_5 \left(\frac{1}{|\mathcal{Y}|} \left(\boldsymbol{\theta}_z - \left| \sum_{i \neq z}^{\mathcal{Y}} (\boldsymbol{\theta}_z - P(\hat{\mathbf{y}}_i|\mathbf{x}) * \boldsymbol{\theta}_i) \right| \right)^2 \right)$
    weights $\leftarrow \mathbf{w}^{\text{near}} + \mathbf{w}^{\text{far}} * \frac{1}{2}$
    $L(P(\hat{\mathbf{y}}|\mathbf{x});\ \boldsymbol{\theta}) \leftarrow (L_{\text{SSD}} + L_{\text{Prev}} + L_{\text{Max}} + L_{\text{Budget}} + L_{\text{TReg}}) * \text{weights}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla J(\boldsymbol{\theta})$
    update_priorities$(L(P(\hat{\mathbf{y}}|\mathbf{x});\ \boldsymbol{\theta}))^{-1})$
    clamp$(\boldsymbol{\theta}, low = -1, high = 1)$
  end for
end for

Mask Memory Unit for Consistent Saliency Maps. A common problem with perturbation-based methods [7, 8, 19] is high variance between explanations, even given the same inputs multiple times. Also, during learning between epochs, the saliency maps can change dramatically [20]. This makes training challenging, in particular, to know when to stop training, especially for learning to perturb methods. We address this problem using a *Mask Memory Unit*, in which we store "good" saliency maps throughout training. A saliency map is considered to be good if the resulting prediction argmax $f(\Phi(\tilde{\mathbf{x}}))$ for the perturbed instance $\tilde{\mathbf{x}}$ is identical to the predicted class z for the instance of interest $\mathbf{x}$. The final saliency map generated corresponds to the average of the last few (A hyper parameter value of 10) good saliency maps in the repository. This significantly reduces the variance between the generated saliency maps offered by DEMUX as it results in generating consistent explanations for same time series instance across multiple runs.

4 Experimental Study

4.1 Datasets

We evaluate our method *DEMUX* on seven popular real-world multi-class time series datasets: ACSF1 [31], Plane [32], Trace [33], Rock [34],

SmallKitchen [32], ECG5000 [35], Meat [36]. Each is a popular publicly-available dataset [32] for multi-class time series classification. The summary statistics are provided in Table 1. We use the default train and test split provided. For each, we train a three-layered Fully Connected Network (FCN), and a Recurrent Neural Network (RNN) to serve as multi-class deep time series classifiers in need of explanations. Three-layered Fully Connected Network is considered as a strong baseline in time series classification [37]. Two models are then trained to achieve state-of-the-art accuracy [32] on their respective datasets.

Dataset	ACSF1	Plane	Trace	ECG5000	Meat	Rock	SmallKitchen
Num. Train Instances	100	20	100	500	60	500	375
Num. Test Instances	100	50	100	4500	60	500	375
Num. Time steps	1460	2844	275	140	500	500	720
Num. Classes	10	7	4	5	3	4	3

Table 1: Multi-Class datasets summary statistics.

4.2 Compared Methods

We compare our proposed method, DEMUX, to nine state-of-the-art explanation methods.

- *DYNAMASK (DYNA)* [8]. This method is designed for multi-variate time series, it is an extension of MP [19] but incorporates the time dependency at each time step for each feature and learns to perturb each time step with static values or adjacent time step values such that $P(\hat{\mathbf{y}}|\mathbf{x})$ decreases. Saliency values are then used as the final explanation.
- *PERT* [7]. This method is designed for binary univariate time series and learns to dynamically replace each time step using time step specific interpolation. It learns to perturb each time step such that $P(\hat{\mathbf{y}}|\mathbf{x})$ is preserved to derive instance-specific *for* and *against* evidence.
- *TSMULE (TMULE)* [6]. This method is an extension of LIME [12] to identify superpixel-like patterns, i.e., semantically related data regions, in time series data. Matrix profile, SAX transformation and uniform segmentation algorithms are used to generate time segments for the perturbation.
- *TIMESHAP (TSHAP)* [9]. This method builds upon SHAP and extends it to time series to explain important features and time steps contributing to the prediction. A temporal coalition pruning method aggregates time events to produce sub-sequences of important data regions.
- *SHAP* [13]. SHAP assigns Shapley values [38] to each time step, thus computing their contributions to $P(\hat{\mathbf{y}}|\mathbf{x})$. Each time step in the instance of interest

is replaced by a representative value observed at the corresponding time step, sampled from the background dataset.

- *RISE* [39]. The partial derivative of the opaque model's prediction $P(\hat{\mathbf{y}}|\mathbf{x})$ with respect to each time step is estimated empirically by randomly setting time steps to zero and summarizing its impact on the $P(\hat{\mathbf{y}}|\mathbf{x})$ to produce final explanation. This method was originally built for image domain.
- *LEFTIST (LEFT)* [11]. The partial derivative of $P(\hat{\mathbf{y}}|\mathbf{x})$ with respect to segments of time steps is estimated empirically by randomly *replacing* the corresponding values from a random instance from the background dataset, or with constants and summarizing its impact on $P(\hat{\mathbf{y}}|\mathbf{x})$.
- *LIME* [12]. Saliency values are derived from the coefficients of a linear surrogate model, trained to mimic the opaque model's behaviour in the feature space surrounding $\mathbf{x}$. The approach's success relies on the opaque model behaving linearly locally, which is rarely guaranteed. This method too was originally built for image domain.
- *Meaningful Perturbation (MP)* [19]. MP built for image domain learns to perturb each time step such that $P(\hat{\mathbf{y}}|\mathbf{x})$ decreases. Perturbation is achieved by combining squared exponential smoothing with additive Gaussian noise. Saliency values are then learned iteratively and are 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 Gated Recurrent Units (GRU) cells to serve as fairly standard multi-class time series classifiers [37] in need of explanations. Each dataset comes with a pre-defined split ratio. We train each model only on the training data, then explain their predictions for all test instances for each compared explainability method. All reported metrics are the result of the average over five runs to estimate the variance of the explainability methods. We optimize our proposed method using Adam [40] with a learning rate of $1e^{-3}$ and train for 5000 epochs, which we find empirically achieves convergence. We use the Weights & Biases framework [41] for experiment tracking. Our proposed method is implemented in PyTorch. For each compared method, we have used grid search to compute the optimum values of hyperparameters to get the best possible results. Our source code is publicly-available at https://github.com/rameshdoddaiah/DEMUX.

4.4 Metrics

We use two key popular metrics to evaluate saliency maps for time series under the intuition that a prediction is well-explained if it accurately ranks the class-specific time steps by their importance, as defined by changes in $P(\hat{\mathbf{y}}|\mathbf{x})$, and also returns only the most important time steps and the intersection of overlapping salient regions across the classes are minimal.

Measuring Insertion and Deletion Sensitivity Impact: AUC-Difference Metric [7, 39, 42]. Saliency maps can be evaluated by "inserting"

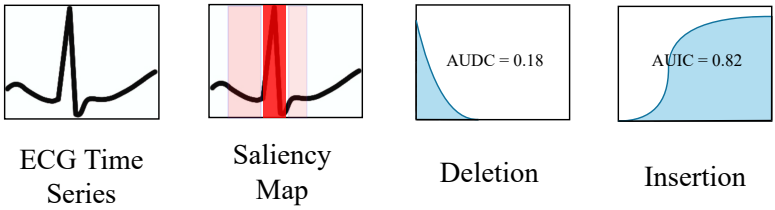


Fig. 3: Deletion and Insertion Metrics. ECG time serie is shown on the left. In the middle, saliency maps for the respective important regions are highlighted in the red. In the right, Deletion (AUDC) metrics shows area under the curve result of deleting important salient time steps from the time series, Insertion (AUIC) metrics shows area under the curve result of inserting important salient time steps to the time series. A Higher AUIC score and lower AUDC score generates a good explanation.

or "deleting" time steps from the time series instance based on the derived importance map and observing the changes in the opaque model's predictions [7, 39, 42]. A good saliency map is one that has ranked time steps such that when the most important time steps are inserted or deleted, there is a sharp change in the confidence of the model's prediction. This can be measured by computing the area under the deletion curve (AUDC) as time steps are deleted one by one. A lower value for this area indicates a better explanation. Analogously, insertion of a few important time steps 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, Equation 9 represents the difference between AUIC and AUDC where we expect the difference, AUC-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 as per PERT [7, 42] and RISE [39].

$$\text{AUC-Difference} = \text{AUIC} - \text{AUDC} \tag{9}$$

The metric alleviates the need for human evaluation and the error-prone collection of human ground-truth. This makes it fairer and truer to the classifier's own view on the problem. A pivotal choice in insertion and deletion tests is which reference input to use when deleting or inserting values to avoid spurious model classification. To achieve deletion during evaluation, we replace a to-be-deleted time step with the farthest cluster's centroid derived from the background data set D to provide in-distribution replacement values. Conversely, for insertion, we start with the the farthest cluster's centroid and iteratively replace the time steps with the values of the instance-of-interest. Figure 3 shows an example, illustrating a good explanation that has both a high AUIC score and a low AUDC score. This results in a high AUC-Difference, indicating that the explainability method has accurately ranked timesteps by their importance.

Measuring Class-Specificity of an Explanation: Intersection over Union. We leverage a standard intersection over union (IoU) metric from the image domain [15]. A simple and a good class-specific saliency map is the one that has no overlapping salient time steps across the classes with respect to the predicted class. This can be measured by intersection over union [15] of salient time steps of the predicted class saliency map with other classes' saliency maps. First, we map each per-class salient time steps to a *boolean value*, either 1 or 0 based on a preset range of thresholds [min=0, max=1, step=0.1]. Second, we find intersection of per-class saliency map with predicted class saliency map and count (IC) number of non-zero salient time steps. Third, we find the union of per-class saliency map with predicted class saliency map and count (UC) number of non-zero salient time steps.

Equation 10 represents the ratio of overlapping salient time steps. We repeat step two and three with different thresholds ranging from 0 to 1 at every 0.1 interval to compute area under the curve (IoU curve). Minimal overlapping salient time steps leads to a higher the number of class-specific salient time steps. The area under the curve IoU metric must be minimal to generate unique explanation.

$$\text{Intersection over Union (IoU)} = \frac{IC}{UC} \tag{10}$$

A higher AUC-Difference and higher IoU metric results in generating important but non-unique salient time steps as an explanation. A lower AUC-Difference and higher IoU metric results in generating unimportant and non-unique salient time steps as an explanation. A lower AUC-Difference and lower IoU metric results in generating unimportant and meaningless salient time steps as an explanation. An ideal multi-class explanation should have high AUC-Difference and low IoU, indicating the saliency values are correct *and* the class-specific information has been identified.

Ranking saliency methods: Combined Normalized Metric. Class-specific saliency maps are evaluated using quantifiable measures - *AUC-Difference* and *IoU*. *AUC-Difference* measures the quality of the derived explanation with respect to the black-box classifier f and ranges between -1 (low quality) and 1 (high quality). *IoU* indicates class-specificity and is normalized between 0 (class-specific) and 1 (general). To holistically compare explainability methods, we must consider both together. To achieve this comparison, we compute a normalized metric that combines *AUC-Difference* and *IoU*. This new metric, which we refer to as CNM (Combined Normalized Metric), ranges between 0 (low quality, class non-specific explanation) and 1 (high quality, class-specific explanation). Equation 11 shows CNM, where the AUC-Difference is prioritized as the lower bar for a model's success.

$$\text{Combined Normalized Metric (CNM)} = \frac{1}{2} * \left(\|\text{AUC-Difference}\| + \frac{\|1 - \text{IoU}\|}{2} \right) \tag{11}$$

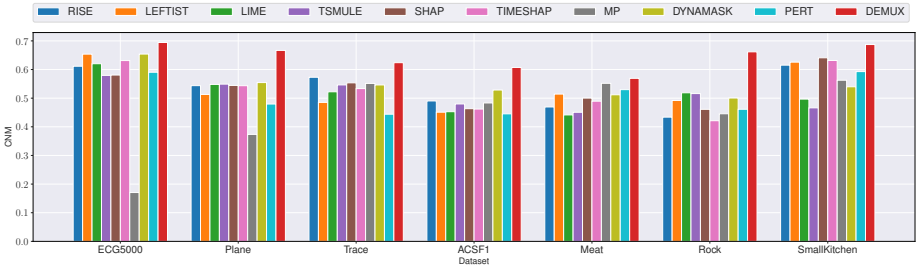


Fig. 4: Performance of the CNM metric for the FCN model.

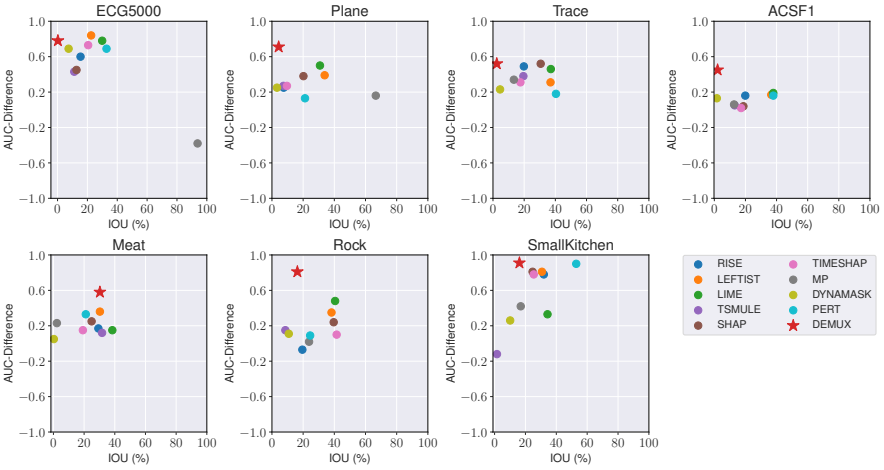


Fig. 5: Performance of the AUC-Difference ↑, IoU ↓ metrics with the FCN model. DEMUX (★) wins on all datasets.

4.5 Experimental Results

DEMUX finds locally faithful explanations per-class. To address the challenge of generating explanations being locally faithful to the classifier's behavior, we first measure how well all compared explainability methods have *ranked* the time steps by their importance via the AUC-Difference metric [7, 39]. Our results using Fully-Connected, and Recurrent Neural Networks can be found in Tables 2, and 3, respectively, and in Figures 4, 5, 6, and 7, where we evaluate each method on seven multi-class datasets. In most of the cases, DEMUX significantly outperforms all other methods on AUC-Difference by learning better rankings for time step importance. In general, learning to perturb, as done in DEMUX and PERT, outperforms the random-replacement methods RISE [11, 39], LIME [12], and SHAP [9] for both the long and short time series. This robust performance demonstrates that learned perturbations effectively produce higher-quality explanations than unlearned perturbations.

Datasets		Methods									
		RISE	LEFT	LIME	TMULE	SHAP	TSHAP	MP	DYNA	PERT	DEMUX
ECG5000	**AUC** ↑	0.60 (.005)	0.84 (.003)	0.78 (.012)	0.43 (.017)	0.45 (.133)	0.73 (.175)	-0.38 (.021)	0.69 (.005)	0.69 (.010)	0.78 (.007)
	IoU ↓	15.4%	22.5%	29.9%	11.2%	12.7%	20.5%	93.8%	7.41%	32.8%	0.24%
	CNM ↑	0.61	0.65	0.62	0.57	0.58	0.63	0.17	0.65	0.59	**0.69**
Plane	**AUC** ↑	0.25 (.009)	0.39 (.011)	0.50 (.005)	0.27 (.009)	0.38 (.011)	0.27 (.023)	0.16 (.001)	0.25 (.008)	0.13 (.007)	0.71 (.001)
	IoU ↓	7.52%	33.8%	30.8%	7.39%	20.2%	9.6%	66.6%	3.23%	21.2%	4.31%
	CNM ↑	0.54	0.51	0.54	0.55	0.54	0.54	0.37	0.55	0.47	**0.66**
Trace	**AUC** ↑	0.49 (.004)	0.31 (.007)	0.46 (.006)	0.38 (.015)	0.52 (.045)	0.31 (.067)	0.34 (.009)	0.23 (.007)	0.18 (.001)	0.52 (.004)
	IoU ↓	19.8%	36.9%	37.1%	19.5%	30.6%	17.6%	13.4%	4.54%	40.4%	2.44%
	CNM ↑	0.57	0.48	0.52	0.54	0.55	0.53	0.55	0.54	0.44	**0.62**
ACSF1	**AUC** ↑	0.16 (.007)	0.17 (.014)	0.19 (.006)	0.05 (.008)	0.04 (.005)	0.02 (.089)	0.06 (.011)	0.13 (.007)	0.16 (.047)	0.45 (.012)
	IoU ↓	19.9%	36.6%	37.9%	13.1%	18.7%	17.2%	12.7%	1.62%	37.9%	2.11%
	CNM ↑	0.49	0.45	0.45	0.47	0.46	0.46	0.48	0.52	0.44	**0.60**
Meat	**AUC** ↑	0.17 (.010)	0.36 (.006)	0.15 (.002)	0.12 (.003)	0.25 (.002)	0.15 (.033)	0.23 (.044)	0.05 (.001)	0.33 (.021)	0.58 (.007)
	IoU ↓	29.3%	30.3%	38.4%	31.7%	24.9%	19.2%	2.25%	0.27%	21.1%	30.3%
	CNM ↑	0.46	0.51	0.44	0.45	0.50	0.48	0.55	0.51	0.52	**0.57**
Rock	**AUC** ↑	-0.07 (.012)	0.35 (.015)	0.48 (.003)	0.15 (.043)	0.24 (.016)	0.10 (.007)	0.02 (.00)	0.11 (.021)	0.09 (.015)	0.81 (.001)
	IoU ↓	19.5%	38.2%	40.5%	8.6%	39.6%	41.5%	23.8%	10.8%	24.5%	16.3%
	CNM ↑	0.43	0.49	0.51	0.51	0.46	0.42	0.44	0.50	0.46	**0.66**
SmallKitchen	**AUC** ↑	0.78(.005)	0.81(.015)	0.33(.052)	-0.12(.02)	0.81(.005)	0.78(.007)	0.42(.001)	0.26(.001)	0.90(.001)	0.91(.008)
	IoU ↓	31.9%	30.7%	34.3%	1.53%	24.63%	25.38%	16.95%	10.1%	52.9%	16.1%
	CNM ↑	0.61	0.62	0.49	0.46	0.64	0.63	0.56	0.53	0.59	**0.68**

Table 2: Performance of the AUC-Difference ↑, IoU ↓ and CNM ↑ metrics with the FCN model. Parentheses indicate σ.

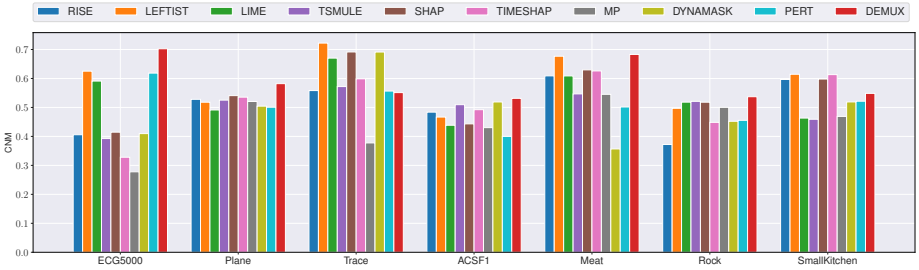


Fig. 6: Performance of the CNM ↑ metrics with the RNN model.

DEMUX finds class-specific salient time steps. We next find that most state-of-the-art methods generate per-class saliency maps independently, leading to overlapping salient regions across the classes. As shown in Tables 2 and 3, we also compare all methods according to the IoU metric [15], which measures how much saliency maps overlap between classes. As expected, DEMUX largely outperforms existing methods by learning to remove overlaps during training. Interestingly, DYNAMASK is competitive in three out of the seven datasets, though it has much lower AUC-Difference. Because DYNAMASK seems to sacrifice AUC for IoU, there appears to be a trade-off between these two metrics,

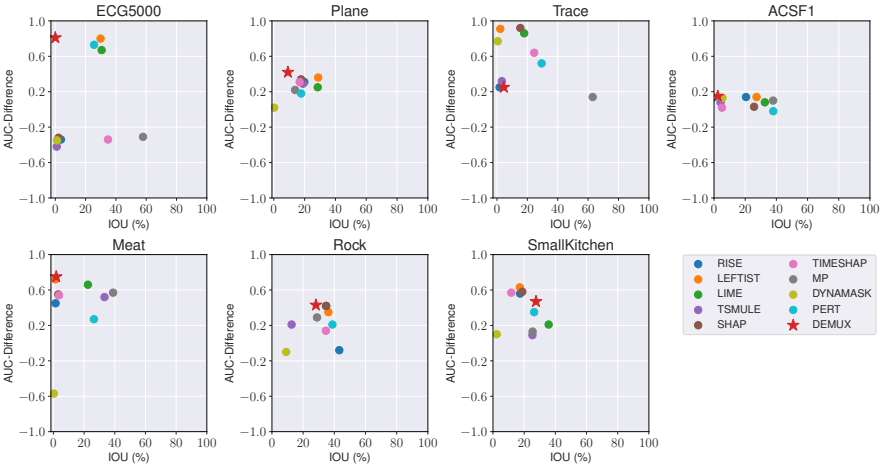


Fig. 7: Performance of the AUC-Difference ↑, IoU ↓ metrics with the RNN model. DEMUX (★) wins on most datasets.

Datasets		Methods									
		RISE	LEFT	LIME	TMULE	SHAP	TSHAP	MP	DYNA	PERT	DEMUX
ECG5000	**AUC** ↑	-0.34 (.005)	0.80 (.002)	0.67 (.028)	-0.42 (.040)	-0.32 (.030)	-0.34 (.075)	-0.31 (.005)	-0.35 (.001)	0.73 (.001)	0.81 (.007)
	IoU ↓	3.71%	29.9%	30.6%	1.0%	2.15%	34.77%	57.9%	1.14%	25.6%	0.01%
	CNM ↑	0.4	0.6	0.59	0.39	0.41	0.32	0.27	0.40	0.61	**0.70**
Plane	**AUC** ↑	0.31 (.009)	0.36 (.005)	0.25 (.005)	0.29 (.008)	0.34 (.003)	0.31 (.007)	0.22 (.001)	0.02 (.011)	0.18 (.007)	0.42 (.001)
	IoU ↓	19.8%	28.8%	28.5%	18.9%	17.7%	16.8%	13.8%	0.22%	17.7%	9.16%
	CNM ↑	0.52	0.51	0.49	0.52	0.54	0.53	0.52	0.50	0.50	**0.58**
Trace	**AUC** ↑	0.25 (.004)	0.91 (.006)	0.86 (.007)	0.32 (.029)	0.92 (.045)	0.64 (.037)	0.14 (.009)	0.77 (.007)	0.52 (.070)	0.25 (.004)
	IoU ↓	1.73%	2.16%	17.9%	3.2%	15.4%	24.5%	63.0%	0.50%	29.4%	4.52%
	CNM ↑	0.55	**0.72**	0.67	0.57	0.69	0.59	0.37	0.69	0.54	0.55
ACSF1	**AUC** ↑	0.14 (.005)	0.14 (.019)	0.08 (.004)	0.08 (.003)	0.03 (.013)	0.02 (.063)	0.10 (.020)	0.13 (.023)	-0.02 (.035)	0.15 (.007)
	IoU ↓	20.5%	27.3%	32.6%	4.2%	25.7%	5.1%	37.9%	5.43%	38.0%	2.43%
	CNM ↑	0.48	0.46	0.43	0.50	0.44	0.49	0.43	0.51	0.4	**0.53**
Meat	**AUC** ↑	0.45 (.010)	0.72 (.002)	0.66 (.001)	0.52 (.002)	0.55 (.001)	0.54 (.008)	0.57 (.021)	-0.57 (.031)	0.27 (.002)	0.75 (.007)
	IoU ↓	1.42%	1.28%	22.5%	33.3%	3.15%	3.6%	38.9%	0.30%	26.4%	1.81%
	CNM ↑	0.60	0.67	0.60	0.54	0.63	0.62	0.54	0.35	0.50	**0.68**
Rock	**AUC** ↑	-0.08 (.001)	0.35 (.021)	0.42 (.007)	0.21 (.081)	0.42 (.020)	0.14 (.065)	0.29 (.042)	-0.10 (.008)	0.21 (.003)	0.43 (.009)
	IoU ↓	43.2%	36.2%	34.7%	12.6%	34.78%	34.6%	28.9%	9.1%	38.8%	28.2%
	CNM ↑	0.37	0.49	0.51	0.52	0.51	0.44	0.50	0.45	0.45	**0.54**
SmallKitchen	**AUC** ↑	0.56 (.007)	0.63 (.010)	0.21 (.083)	0.09 (.012)	0.58 (.071)	0.57 (.023)	0.13 (.059)	0.10 (.004)	0.35 (.008)	0.47 (.017)
	IoU ↓	17.4%	17.2%	35.7%	25.3%	18.8%	11.7%	25.4%	2.4%	26.4%	27.6%
	CNM ↑	0.59	**0.61**	0.46	0.45	0.59	0.61	0.46	0.51	0.52	0.54

Table 3: Performance of the AUC-Difference ↑, IoU ↓ and CNM ↑ metrics with the RNN model. Parentheses indicate σ.

as expected. By and large, DEMUX significantly outperforms the state-of-the-art methods on both metrics, especially when considering them together on CNM metrics as shown in Figures 4 and 6. RNN model's low accuracy could

impact the explanation quality as seen on the Trace dataset in Figure 7 but w.r.t. IOU metrics, DEMUX remains competitive.

DEMUX learns consistent explanations. Some recent works express concern that perturbation methods can lead to different explanations [20] for the same instances when respective saliency masks are initialized randomly. We share this worry and provide some peace of mind: DEMUX achieves more-consistent explanations across multiple runs than other methods. Each saliency mask is re-initialized five times on all time series in each dataset. We report the standard deviation for the AUC-Difference metric as shown in Tables 2 and 3. DEMUX has a far-lower average standard deviation across all datasets when AUC-Difference and IoU are considered together.

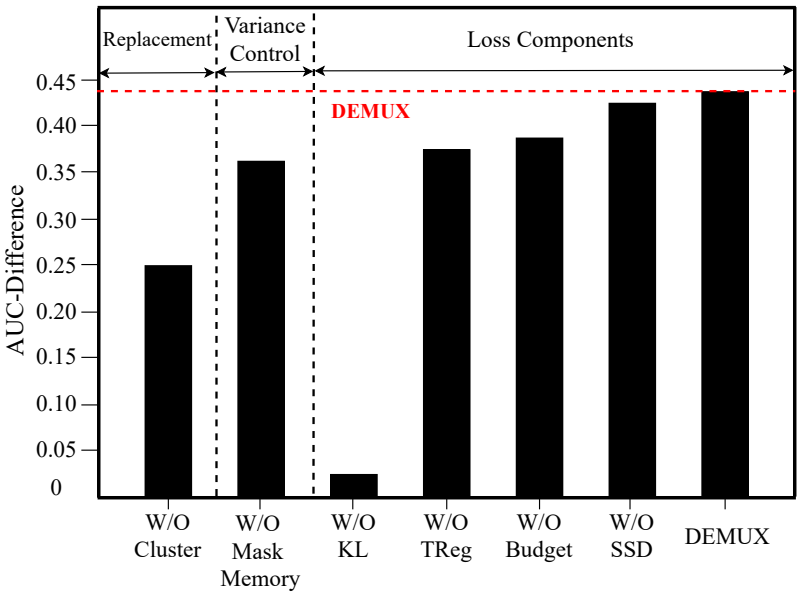
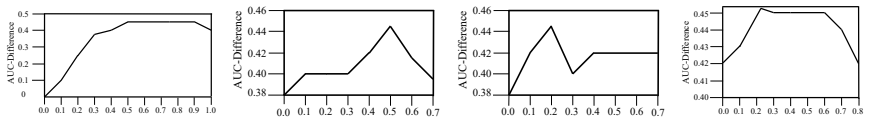


Fig. 8: DEMUX Ablation study results on ACSF1 dataset.

Ablation study. To demonstrate the need for each component of DEMUX, we perform an ablation study, removing different DEMUX components and reporting the AUC-Difference on the ACSF1 dataset [31]. We focus on the replacement strategy, mask memory unit, and each loss component. Our results in Figure 8 show that each component is necessary to achieve a good performance.

First, without the per-class time series replacement learning to sample component (*W/O Cluster*) AUC-Difference suffers when random time series with irrelevant time steps are used for replacement. This shows the need for selecting class-specific replacement values per time step to create per-class perturbations. *Second,* without the mask memory unit component (*W/O Mask Memory*), we

pick the final saliency map at 5000^{th} epoch as explanation and the AUC-Difference metric decreases slightly. This shows that Perturbation-based method may not consistently pick a good saliency map if stopped after a predefined number of epochs. *Third,* we compare DEMUX's performance while removing different components of the loss function. As expected, W/O KL clearly has a substantial impact on the final AUC-Difference, as without this component, there is no relationship between the saliency map and the model's prediction probability distribution. *Fourth,* the W/O TReg component has less of an impact but still contributes to DEMUX's state-of-the-art performance by learning discriminative subsequences, as without this component, DEMUX has no incentive to learn important subsequences and instead focuses on individual and sparse time steps. *Fifth,* the W/O Budget component has less of an impact but helps DEMUX to generate a simple explanation with non-redundant time steps. *Sixth,* learning without a class-specific loss component W/O SSD reduces DEMUX's AUC-Difference further, stressing the need for learning class-specific saliency maps. This component helps to learn distinct class-specific salient time steps. DEMUX includes all the above components needed to generate a simple, class-specific, and consistent salient subsequences for the top predicted class, and indeed achieves the best AUC-Difference.



(a) L_{Prev} coefficient (λ_1). (b) L_{Max} coefficient (λ_2). (c) L_{Budget} coefficient (λ_3). (d) L_{SSD} coefficient (λ_5).

Fig. 9: DEMUX hyperparameter study results on ACSF1 dataset.

Hyperparameter Study. Producing class-specific explanations with DEMUX involves balancing the four key hyperparameters: coefficients for L_{SSD}, L_{Prev}, L_{Max}, and L_{Budget}. We investigate the effects of tuning these coefficients in isolation on the ACSF1 dataset and report our results in Figure 9. For each case, we keep all unchanged parameters at their best-found values.

First, as shown in Figure 9a, we vary the coefficient λ_1 of L_{Prev} between 0 to 1. This maintains the prediction probability distribution across all the classes for a given instance and we find a value of 0.7 suffices. A low value for λ_1 fails to preserve the model's probability distribution, so the loss function may not encourage instance-specific explanations. There, we notice a gradual increase in AUC-Difference until it stabilizes before dropping for higher values of λ_1. Second, a low value for the coefficient of L_{Max} λ_2 does not generate good explanations for low probability predictions. Here a high value discards salient time steps from the explanations. AUC-Difference increases until λ_2 reaches 0.5 and decreases for values ≥ 0.5. The optimal value of 0.5 provides a simple and meaningful explanation as shown in Figure 9b. Third, too-low L_{Budget} coefficient

produces redundant salient time steps, while too high removes important salient time step. AUC-Difference gradually climbs up until this coefficient reaches a value of 0.2, and then it starts decreasing and later stabilizes. The optimal value of 0.2 provides a simple and meaningful explanation as shown in Figure 9c. Fourth, we vary the coefficients between 0 to 0.8 for L_{SSD} and find the optimal value is 0.2 as shown in Figure 9d. Too low means not enough focus on class-specificity, while a high value could remove important regions in favor of class-specificity.

Hyperparameter	**Search Space**	**Selected value**
1. Background dataset	10, 20, 30, 40, 50, 60, 70, 80, 90, 100	100%
Optimizer	SGD, Adam	Adam
2. Learning rate	0.0001, 0.001, 0.01, 0.1	0.01
3. Epochs	100, 500, 1000, 2000, 5000, 8000, 10000	5000
4. Distance measure	Euclidean, Random, Cluster, Mahalanobis	Cluster
5. Evaluation replacement	Zeros, Ones, Cluster Mean, Instance Mean Random Class Mean, Random Instance	Cluster Mean
6. Loss preservation	MSE, KL Divergence, Cosine, Hinge, Triplet	KL Divergence
7. Mask repository size	10, 20, 30, 40, 50, 60, 70, 80, 90, 100	10
8. Mask repository type	Last N, Last N Conditional	Last N Conditional
9. Number of clusters	Kneedle, Manual	Kneedle
10. Mask Normalization	clamp, softmax, sigmoid, none	clamp
11. Noise Mean	-1, 0, 1	0
12. Noise Std.	0.1, 0.01, 0.5	0.1

Table 4: DEMUX hyperparameter tuning table.

We perform a hyperparameter study to find the best hyperparameters for DEMUX. We search over a grid of parameters, then report the best parameters for DEMUX. The range of hyperparameters and the best selected value for each of the hyperparameters are shown in hyperparameter table 4. 1) 100% of the *test* data is used as background dataset to learn the best replacement time series for perturbing the given instance. 2) Between SGD and Adam optimizer, Adam is selected as it converges faster whilst performing significantly better on metrics compared to SGD. 3) A learning rate of 0.01 is selected along with maximum of 5000 iterations, which we find empirically achieves convergence. 4) Among the

various distance measurement methods such as Euclidean, Mahalanobis and Cluster, KMeans clustering is selected to sample replacement time series from respective clusters. 5) We evaluate the class-specific salient time steps in the generated Saliency Maps by replacing salient time steps with either zero, ones, random class mean values, cluster mean values, instance mean values, instance, and selected cluster mean as best method to evaluate the class-specific salient time steps. 6) We tried with various loss functions - mean square error (MSE), Kullback–Leibler (KL) Divergence, Cosine, Hinge, Triplet, and selected KL Divergence as the best loss function to generate class-specific saliency maps, as the opaque model predicts probability distribution across multiple classes. 7) Mask repository value of 10 is selected heuristically to obtain consistent explanation. 8) Mask repository stores a set of masks to derive consistent explanations and the selection criteria for storing a mask - save the last N good masks where N is defined by *Mask repository value* parameter. A saliency map is considered to be good if the resulting prediction argmax $f(\Phi(\tilde{\mathbf{x}}))$ for the perturbed instance $\tilde{\mathbf{x}}$ is identical to the predicted class z for the instance of interest $\mathbf{x}$. 9) Kneedle algorithm is used to automatically select the best number of clusters for KMeans Clustering. 10) To address the challenge of exploding gradients, we clamp the mask between -1 and 1. 11, and 12) To generate robust explanations, we introduce gaussian noise to the perturbations and heuristically we have found that a mean of 0 and std. of 0.1 works best.

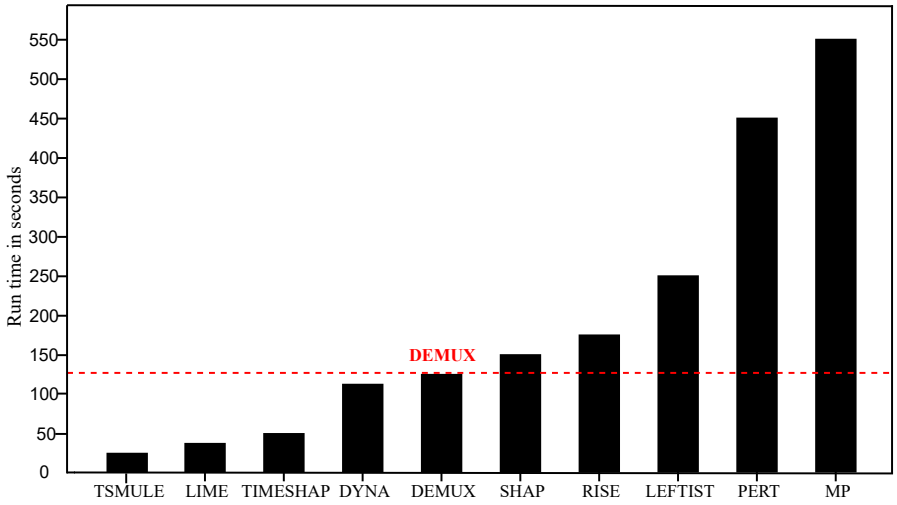


Fig. 10: DEMUX run time performance comparison.

Runtime Performance. To explore DEMUX's practicality, we perform a runtime comparison of DEMUX with the nine state-of-the-art comparisons on *ACSF1* dataset [31] to generate explanations for a three-layered fully-connected network. Our results are shown in Figure 10. On average, DEMUX takes 120

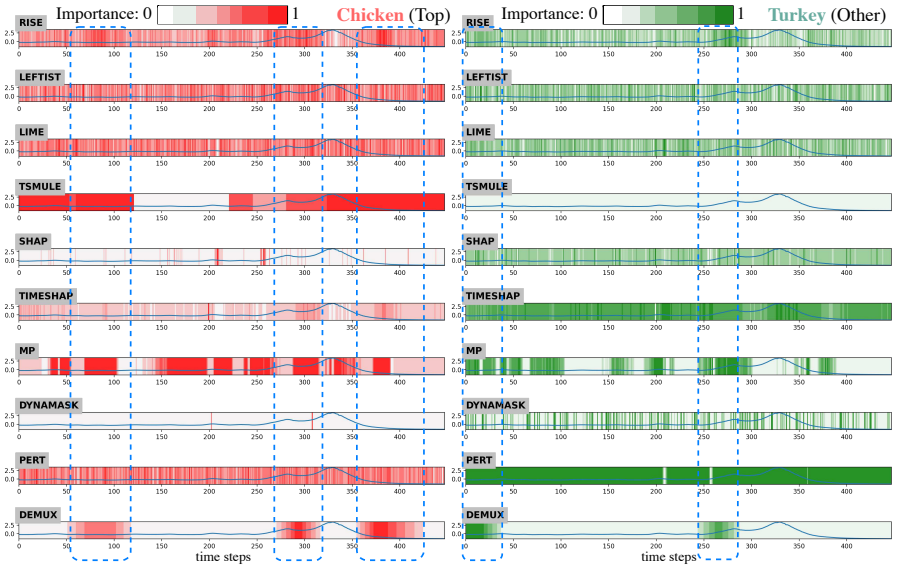


Fig. 11: MEAT [36] "Chicken" Class Case Study. The time series is shown in blue and saliency values are plotted against time steps in red for the top predicted chicken class and in green for another class, Turkey. DEMUX (the last row) highlights three unique, discriminative subsequences as the class-specific explanation, outlines by blue dashed boxes for the top predicted "Chicken" class, unlike the other methods. DEMUX highlights evidence for all classes, but only "Turkey" class is shown here due to space constraints.

seconds for each time series instance. Compared to the other methods, DEMUX is computationally faster than most, but TSMULE, LIME, TIMESHAP, and DYNA are slightly faster. We also observe similar pattern across other datasets irrespective of the time series length. This finding implies that runtime is not a consideration barring the use of DEMUX as compared to other methods.

Case Study on the MEAT [36] dataset. Finally, we describe a case study conducted using DEMUX to explain an Recurrent Neural Network's predictions for an instance from the MEAT dataset [36]. Each instance of MEAT dataset has 500 time steps. This is a multi-class classification task: MEAT is a food spectrograph dataset describing food types as class—a crucial task in food safety and quality assurance. There are three classes: *Chicken*, *Pork*, and *Turkey*, which are predicted with probabilities 88%, 11%, and 1%, respectively for the instance we consider.

Our case study results are shown in Figure 11, where each method's time step importance is **red** for class *Chicken* and **green** for *Turkey*, with darker hues indicating higher importance. By visually investigating discriminative subsequences, we see that the compared methods highlight several regions of input time series as important for the top predicted class *Chicken*. On

the contrary, DEMUX (the last row) succeeds to highlight the discriminative subsequences as class-specific evidence for the top predicted class *Chicken*. Unlike the compared methods, DEMUX has reduced overlapping salient regions between the top predicted class *Chicken* and class *Turkey* as shown in the blue dashed vertical boxes (*Pork* class is not shown due to space constraints). These results suggest that DEMUX's superiority over state-of-the-art methods is due to the ability to learn class-specific evidence with respect to other classes. Most of the existing methods look to identify redundant time steps as salient. These methods find many unimportant time steps instead of fewer high-importance time steps, failing to provide simple and meaningful explanations.

5 Conclusion

With this work, we identify class-specificity as a critical criteria for meaningful explanations of multi-class deep time series classifiers. Solving this open problem is essential for users who seek explanations for why a model predicted one class *in particular* [15]. We introduce DEMUX, the first solution to this open problem, which extends beyond recent methods for perturbation-based explainability for time series models by learning to encourage class-specificity during training via a novel loss function. With three interdependent modules, all of which are learned, DEMUX successfully produces class-specific explanations. In our experiments, we compare DEMUX to nine state-of-the-art explainability methods for three deep models on seven multi-class datasets. Our results demonstrate that DEMUX's explanations are (1) more class-specific than the alternatives, which leads to (2) higher quality according to popular explainability metrics, and (3) consistent explanations across multiple initializations.

References

[1] Dixon, M.F., London, J.: Financial forecasting with α-rnns: A time series modeling approach. In: Frontiers in Applied Mathematics and Statistics (2020)

[2] Fulcher, B.D., Jones, N.S.: hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. Cell systems **5** **5**, 527–5313 (2017)

[3] Che, Z., Purushotham, S., Cho, K., Sontag, D.A., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. Scientific Reports **8** (2018)

[4] Li, G., Choi, B., Xu, J., Bhowmick, S.S., Chun, K.-P., Wong, G.L.: Shapenet: A shapelet-neural network approach for multivariate time series classification. In: AAAI (2021)

[5] Tonekaboni, S., Joshi, S., Duvenaud, D., Goldenberg, A.: What went wrong and when? instance-wise feature importance for time-series models. Advances in Neural Information Processing Systems (2020)

[6] Schlegel, U., Vo, D.L., Keim, D.A., Seebacher, D.: Ts-mule: Local interpretable model-agnostic explanations for time series forecast models. In: PKDD/ECML Workshops (2021)

[7] Parvatharaju, P.S., Doddaiah, R., Hartvigsen, T., Rundensteiner, E.A.: Learning saliency maps to explain deep time series classifiers. Proceedings of the 30th ACM International Conference on Information & Knowledge Management (2021)

[8] Crabbe, J., van der Schaar, M.: Explaining time series predictions with dynamic masks. In: ICML (2021)

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

[10] Guidotti, R., Monreale, A., Spinnato, F., Pedreschi, D., Giannotti, F.: Explaining any time series classifier. International Conference on Cognitive Machine Intelligence (2020)

[11] Guillemé, M., Masson, V., Rozé, L., Termier, A.: Agnostic local explanation for time series classification. ICTAI, 432–439 (2019)

[12] Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier. CoRR **abs/1602.04938** (2016)

[13] Lundberg, S.M., Lee, S.-I.: A unified approach to interpreting model predictions. In: NIPS (2017)

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

[15] Shimoda, W., Yanai, K.: Distinct class-specific saliency maps for weakly supervised semantic segmentation. In: ECCV (2016)

[16] Kaur, H., Nori, H., Jenkins, S., Caruana, R., Wallach, H., Wortman Vaughan, J.: Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (2020)

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

(2020). https://doi.org/10.1038/s41467-020-15432-4

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

[19] Fong, R., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. ICCV, 3449–3457 (2017)

[20] Kumar, I.E., Venkatasubramanian, S., Scheidegger, C.E., Friedler, S.A.: Problems with shapley-value-based explanations as feature importance measures. In: ICML (2020)

[21] Fong, R., Patrick, M., Vedaldi, A.: Understanding deep networks via extremal perturbations and smooth masks. In: ICCV, pp. 2950–2958 (2019)

[22] Doddaiah, R., Parvatharaju, P., Rundensteiner, E., Hartvigsen, T.: Class-specific explainability for deep time series classifiers. 2022 IEEE International Conference on Data Mining (ICDM) (2022)

[23] Kelany, O.E., Aly, S., Ismail, M.A.: Deep learning model for financial time series prediction. 2020 14th International Conference on Innovations in Information Technology (IIT) (2020)

[24] Mehdiyev, N., Lahann, J., Emrich, A., Enke, D., Fettke, P., Loos, P.: Time series classification using deep learning for process planning: A case from the process industry. Procedia Computer Science **114**, 242–249 (2017)

[25] Zhang, Y., Zhang, Y., Zhang, Z., Bao, J., Song, Y.: Human activity recognition based on time series analysis using u-net. ArXiv **abs/1809.08113** (2018)

[26] Schlegel, U., Arnout, H., El-Assady, M., Oelke, D., Keim, D.A.: Towards a rigorous evaluation of xai methods on time series. 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 4197–4201 (2019)

[27] Zhang, H., Cissé, M., Dauphin, Y., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. ArXiv **abs/1710.09412** (2017)

[28] Satopaa, V., Albrecht, J.R., Irwin, D.E., Raghavan, B.: Finding a "kneedle" in a haystack: Detecting knee points in system behavior. 2011 31st International Conference on Distributed Computing Systems Workshops (2011)

[29] Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. CoRR **abs/1511.05952** (2016)

[30] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing atari with deep reinforcement learning. ArXiv **abs/1312.5602** (2013)

[31] Gisler, C., Ridi, A., Zufferey, D., Khaled, O.A., Hennebert, J.: Appliance consumption signature database and recognition test protocols. In: 2013 8th International Workshop on Systems, Signal Processing and Their Applications (WoSSPA), pp. 336–341 (2013). https://doi.org/10.1109/WoSSPA.2013.6602387

[32] Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The UCR Time Series Classification Archive. www.cs.ucr.edu/~eamonn/time_series_data/ (2015)

[33] Roverso, D.: Multivariate temporal classification by windowed wavelet decomposition and recurrent neural networks. (2000)

[34] Baldridge, A.M., Hook, S.J., Grove, C.I., Rivera, G.: The aster spectral library version 2.0. Remote Sensing of Environment **113**, 711–715 (2009). https://doi.org/10.1016/j.rse.2008.11.007

[35] Goldberger, A.L., Amaral, L.A.N., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.-K., Stanley, H.E.: Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. Circulation **101 23**, 215–20 (2000)

[36] Al-Jowder, O., Kemsley, E.K., Wilson, R.H.: Mid-infrared spectroscopy and authenticity problems in selected meats. Food Chemistry **59**(2), 195–201 (1997). https://doi.org/10.1016/S0308-8146(96)00289-0

[37] Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. IJCNN (2017)

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

[39] Petsiuk, V., Das, A., Saenko, K.: Rise: Randomized input sampling for explanation of black-box models. CoRR **abs/1806.07421** (2018)

[40] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2015)

[41] Biewald, L.: Experiment Tracking with Weights and Biases. Software available from wandb.com (2020). https://www.wandb.com/

[42] Hama, N., Mase, M., Owen, A.B.: Deletion and Insertion Tests in Regression Models (2022)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Table3.png
- Table4.png
- Table2.png
- Algorithm1.png
- Table1.png