Neurocomputing 666 (2026) 132318

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Enhancing randomized recurrent neural networks with explainable attribution methods

Francesco Spinnato [a,b,*], Andrea Ceni [a], Andrea Cossu [a], Riccardo Guidotti [a,b], Claudio Gallicchio [a], Davide Bacciu [a]

[a] University of Pisa, Largo B. Pontecorvo, 3, Pisa, 56127, Italy
[b] ISTI-CNR, Via Giuseppe Moruzzi, 1, Pisa, 56124, Italy

HIGHLIGHTS

- We propose a method to enhance randomized recurrent neural networks by weighting their hidden states using attribution scores derived from explainable AI techniques.
- Theoretical analysis shows that our proposed aggregation corresponds to a second-order residual RNN formulation, whose Jacobian spectral radius is automatically adapted based on XAI-driven attribution values.
- Aggregating hidden states using XAI-derived attribution scores improves classification performance across several datasets, outperforming standard baselines, especially under noisy conditions.

ARTICLE INFO

Keywords:
Recurrent neural networks
Explainable ai
Reservoir computing
Echo state networks

ABSTRACT

Recurrent Neural Networks (RNNs) are well-suited for temporal data modeling but remain limited by their high training computational cost. As a lightweight alternative, randomized RNNs mitigate this issue by employing a fixed, randomly initialized recurrent layer combined with a simple, trainable output layer. To classify a given input sequence, randomized RNNs usually rely on the final reservoir state, which can be suboptimal when relevant temporal information is sparse or masked by noise. In this work, we investigate how explainable attribution methods can improve the performance of randomized RNNs in classification tasks. In particular, we adopt gradient-based attribution explainability techniques to weigh reservoir states according to their relevance to the final prediction. We theoretically justify the effectiveness of our approach through linear stability analysis, offering geometric intuition via an estimation of the variability of the recurrent dynamics by means of explainability techniques. Our experimental evaluation spans 30 binary and 10 multiclass time series classification tasks, comparing several randomized recurrent models. Results show that explainability-guided weighting can improve classification performance in noisy scenarios.

1. Introduction

Deep neural networks have achieved remarkable performance across various domains, including environmental science, financial forecasting, and medical diagnosis [26]. However, their inherent complexity and lack of transparency often hinder their practical adoption, particularly in sensitive applications where interpretability is crucial [2]. To address these limitations, Explainable Artificial Intelligence (XAI) has emerged as a critical research field, aiming to interpret the internal mechanisms of complex deep neural networks and provide insights into their decision-making processes [13].

XAI techniques encompass a diverse range of methodologies designed to make black-box models more interpretable [24]. Traditionally, these methods have focused on enhancing transparency for human stakeholders. More recently, however, research has begun exploring how XAI can also serve as a means to improve model performance by using explanations as feedback during the learning process [1].

Among deep neural network architectures, Recurrent Neural Networks (RNNs) have demonstrated particular effectiveness in processing temporal data and time series [22,23,33]. However, training RNNs with backpropagation through time—the most widely used

* Corresponding author at: University of Pisa, Largo B. Pontecorvo, 3, Pisa, 56127, Italy.
Email address: francesco.spinnato@di.unipi.it (F. Spinnato).

https://doi.org/10.1016/j.neucom.2025.132318
Received 11 June 2025; Received in revised form 10 October 2025; Accepted 1 December 2025
Available online 3 December 2025

learning algorithm for recurrent models—is computationally expensive, requiring significant resources and extended training times [36]. These challenges often hinder the practical deployment of traditional RNNs.

Randomized RNNs, in particular those based on reservoir computing paradigm [16,20], offer an efficient alternative by leveraging a fixed, randomly initialized recurrent layer (called *reservoir*) combined with a simple, trainable output layer (called *readout*) [3,17]. For time series classification tasks, RNNs process the input sequence one step at a time, updating their hidden state after each step. Typically, classification is then performed by relying on the state obtained after processing the last step of the sequence. Alternatively, the readout can take as input the average of the states across all steps, thus assigning an equal weight to each state. However, these approaches exhibit limitations, particularly in scenarios where crucial information is unevenly distributed across time or masked by noise. In such cases, the network should assign more importance to those states carrying the largest amount of information useful for the final classification.

To this end, in this work we extend the analysis presented in [30], further investigating the use of gradient-based, post-hoc explainability methods to enhance randomized RNNs' performance in time series classification, particularly in the presence of noisy inputs. We support the soundness of our approach with a theoretical analysis based on linear stability, and complement it with geometric intuition derived from estimating the variability of the recurrent dynamics using attribution explainability techniques. We also introduce a multiclass extension to expand its scope beyond the binary classification case. Experimentally, we benchmark three additional randomized recurrent models on 30 binary classification datasets and introduce experiments on 10 multiclass tasks.

The remainder of the paper is organized as follows. In Section 2, we review related work relevant to our approach. Section 3 introduces the foundational concepts needed to understand our proposal. Our methodology is presented in Section 4, with a supporting mathematical analysis provided in Section 5. We evaluate the proposed approach in Section 6, and conclude in Section 7 with a summary of our findings and suggestions for future research.

2. Related work

RNNs have become a central focus in XAI research, where attribution and saliency methods are increasingly used to enhance their interpretability across supervised tasks such as time series classification, in both classical [29,35] and continual learning settings [8], as well as in regression and forecasting tasks [11]. This is particularly relevant for randomized recurrent architectures such as Echo State Networks (ESNs) [20], which demonstrate strong performance on such tasks but are often criticized for their lack of transparency [35].

Saliency-based methods, such as *Integrated Gradients* [34], *GradientSHAP* [21], and *Gradient*Input* [28], provide explanations by assigning importance scores to each input feature or time step based on their contributions to the model's predictions. *Integrated Gradients*, for instance, accumulates gradients along a path from a baseline to the actual input, thus quantifying feature influence in a principled manner [34]. *GradientSHAP* similarly compares model predictions against baseline inputs, but incorporates random sampling, often leading to computational efficiency gains over *Integrated Gradients* [21]. These methods have been extensively employed to explain sequence models, highlighting temporal regions critical for predictions, and, among them, those most relevant to our work will be formally presented later in Section 3.

Beyond interpretation, recent studies have exploited these attribution techniques to actively improve model performance. Apicella et al. [1], for example, introduced feature masking strategies guided by saliency explanations. Their approach either removes or softly attenuates less important features, significantly improving accuracy in image classification tasks. However, this method assumes the availability of correct class explanations at inference, which limits practical applicability. Spinnato et al. [32] propose using saliency-based XAI techniques in order to reduce the dimensionality of training data, and improve the efficiency of a deep learning model. However, the approach is tailored to the specific task of car crash prediction. *Saliency Guided Training* proposed by Ismail et al. [14], incorporates saliency into the training process itself. Low-attribution features are systematically masked during training, and a consistency loss encourages stable predictions despite input perturbations. This method results in models with more sparse and meaningful gradients, effectively improving interpretability and robustness. However, it incurs increased computational complexity due to iterative masking and multiple gradient evaluations per training batch. Attention mechanisms offer another avenue, embedding explainability directly into neural architectures [27]. Attention models learn dynamic weight assignments for input features or hidden states, implicitly providing interpretability through learned importance weights. Such approaches have demonstrated performance gains by enabling models to selectively focus on informative input segments [18].

Contrary to the aforementioned approaches, we propose applying gradient-based attribution methods, such as *Gradient*Input* and *GradientSHAP*, to weigh the hidden states of a randomized RNN before feeding them to the readout layer. Unlike learned attention mechanisms or saliency-guided training, our method retains the advantages of the reservoir computing paradigm, by introducing only a lightweight, one-shot retraining of the readout layer with the weighted hidden states derived from attribution maps. We show that attribution-driven weighting of the hidden states represents a practical compromise, maintaining computational efficiency while significantly improving noise robustness and prediction accuracy.

3. Setting the stage

In this section, we introduce all the necessary concepts to understand our proposal.

3.1. Data

Definition 3.1 (Time Series).
A univariate time series is an ordered sequence of data points indexed in time, $\mathbf{x} = [x_1, x_2, \dots, x_T]$, where each x_t is the observation at time t, and T is the total number of observations.

Time series can be collected in time series datasets.

Definition 3.2 (Time Series Dataset).
A time series dataset is a collection of multiple time series, $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2} \dots, \mathbf{x}_N\} \in \mathbb{R}^{N \times T}$

In classification tasks, accompanying the dataset is a label vector that assigns a categorical label to each time series.

Definition 3.3 (Time Series Classification).
Given a time series dataset $\mathbf{X} \in \mathbb{R}^{N \times T}$ with corresponding labels $\mathbf{y} \in \{0, \dots, C-1\}^N$, where C is the number of classes, the objective of time series classification is to train a model f to predict a target for each time series, such that $\hat{\mathbf{y}} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_N)]$ where $\hat{y} = f(\mathbf{x})$ is the predicted label for the time series $\mathbf{x}$.

The aim is to ensure that the predicted labels $\hat{\mathbf{y}}$ are as close as possible to the true labels $\mathbf{y}$. When $C = 2$, the task is called *binary classification*, while when $C > 2$ the task is called multiclass classification.

3.2. Randomized recurrent neural networks

Time series classification can be performed using RNNs [10]. These models rely on a recurrent state-update equation that enables learning of a fixed-size memory state capturing information from previous time steps. The final output is obtained by applying a (usually linear) projection to this hidden memory state. In this work, we focus on *randomized* RNNs belonging to the family of reservoir computing, which differ from

fully trainable RNNs as they initialize the parameters of the state-update equation in a structured manner while learning only the linear output projection. Specifically, we consider the Echo-State Network (ESN) [20], the Random Oscillators Network (RON) [4,7], the Euler State Network (EuSN) [12], and the Residual ESN (RES-ESN) [6] as our randomized RNN models. The ESN is defined by the following state-update equation:

$$\mathbf{h}_t = \sigma(W\mathbf{h}_{t-1} + V x_t + \mathbf{b}), \tag{1}$$

where $\mathbf{W}$ and $\mathbf{V}$ are the hidden-to-hidden and input-to-hidden weight matrices, respectively, $\mathbf{b}$ is the bias term, and x_t represents the input at time t. The activation function σ is typically chosen as the hyperbolic tangent. The recurrent state $\mathbf{h}_t$ serves as a memory of past time steps up to t.

To ensure the desired dynamical properties, $\mathbf{W}$ is initialized uniformly in $[-1, 1]$ and then scaled such that its spectral radius takes the value ρ. A common choice is $\rho < 1$, which is often required to satisfy the Echo-State Property [19], ensuring that the asymptotic behavior of the ESN remains independent of its initial conditions. The input weight matrix $\mathbf{V}$ is initialized uniformly in $[-2, 2]$ and subsequently scaled by an input scaling factor ν. The bias $\mathbf{b}$ is randomly sampled from the range $[-1, 1]$.

The RON network [4] is composed of a randomly connected set of damped oscillators. The state-update equation reads:

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \tau \mathbf{z}_t, \tag{2}$$

$$\mathbf{z}_t = \mathbf{z}_{t-1} + \tau(\sigma(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{V}x_t + \mathbf{b}) - \boldsymbol{\gamma} \odot \mathbf{h}_{t-1} - \boldsymbol{\epsilon} \odot \mathbf{z}_{t-1}), \tag{3}$$

where $\mathbf{z}_t$ is a latent state used to compute $\mathbf{h}_t$, τ is a time-step constant, and $\boldsymbol{\gamma}$ and $\boldsymbol{\epsilon}$ are the frequency and damping vector coefficients, respectively. The matrices $\mathbf{W}$ and $\mathbf{V}$ are initialized as in an ESN and kept fixed, as is the bias.

The EuSN [12] is a recent addition to the family of reservoir computing models, designed to excel in classification tasks. Its effectiveness stems from a carefully structured parameterization of the reservoir, incorporating a numerical integration scheme inspired by Euler's method. The state update rule of the EuSN is formulated as:

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \epsilon\, \sigma\big((\mathbf{A} - \gamma I)\mathbf{h}_{t-1} + \mathbf{V}x_t + \mathbf{b}\big), \tag{4}$$

Here, $\epsilon > 0$ denotes the step size of Euler integration derived from a continuous-time formulation, $\mathbf{A}$ is an antisymmetric matrix governing the internal dynamics, $\mathbf{I}$ is the identity matrix, and $\gamma \in [0, 1)$ is a diffusion parameter that controls the rate of dissipation.

The ResESN model integrates a temporal residual connection mechanism, enabling it to flexibly adapt to a wide range of tasks, performing robustly on both regression (e.g., time series forecasting and memory-based processing) and classification [6]. Its state update equation is defined as:

$$\mathbf{h}_t = \alpha\, \mathbf{O}\, \mathbf{h}_{t-1} + \beta\, \sigma(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{V}x_t + \mathbf{b}). \tag{5}$$

In this formulation, $\mathbf{O}$ represents a randomly initialized orthogonal matrix, while α and β are tunable hyperparameters that influence the scaling of the linear and nonlinear components, respectively. The random orthogonal matrix $\mathbf{O}$ is constructed by generating a matrix with entries drawn uniformly at random, then computing its QR decomposition and taking the orthogonal factor, as described in [6]. By adjusting these coefficients, one can control the reservoir's dynamical regime and memory characteristics.

Independent of the architecture type, the last hidden state output, $\mathbf{h}_T$, is forwarded to the readout component of the randomized RNN, which is the only trained part of the network. The readout typically involves a linear transformation followed by a nonlinear activation function, depending on the task. For binary classification $\hat{y} = \sigma(\mathbf{W}_{\text{out}}\mathbf{h}_T + \mathbf{b}_{\text{out}})$, where $\mathbf{W}_{\text{out}}$ represents the weights of the readout layer, $\mathbf{b}_{\text{out}}$ is the bias, and σ is a sigmoid activation function. For multiclass tasks, the readout applies a softmax activation instead of a sigmoid. Specifically, the prediction is given by $\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_{\text{out}}\mathbf{h}_T + \mathbf{b}_{\text{out}})$, where $\mathbf{W}_{\text{out}}$ and $\mathbf{b}_{\text{out}}$ retain their roles as the output weights and bias vector, respectively. The softmax function normalizes the output into a probability distribution over the target classes, making it suitable for tasks involving more than two categories. Since the recurrent transformation processed by the reservoir is fixed, the readout can be trained in closed form (e.g., Ridge regression).

3.3. Attribution methods

In this work, we use gradient-based XAI attribution methods to infer the importance of points in the time series. These techniques essentially perform sensitivity analysis, i.e., they can highlight the importance of each observation in the time series. The foundation of these approaches is the analysis of the gradient of the network's output with respect to an input instance, as it informs about the sensitivity of the output to the input features, offering insights into how they affect the network's classification. XAI approaches differ in how they exploit this gradient, and in other information they use [35]. However, their output is always a vector, $\boldsymbol{\phi}$, containing the importance of observations in the input.

We focus on three gradient-based approaches for explaining model predictions: *Gradient*, *Gradient*Input*, and *GradientSHAP*. Each method provides different insights into how the input features influence the model's output.

The simplest approach, *Gradient*, computes the gradient of the predicted output with respect to each observation in the input time series. This method measures the local sensitivity of the prediction to small changes in the input. Mathematically, it is defined as:

$$\boldsymbol{\phi} = \left[\frac{\partial \hat{y}}{\partial x_1}, \ldots, \frac{\partial \hat{y}}{\partial x_T}\right]. \tag{6}$$

Since the gradient represents the steepest direction of change, it highlights which input features have the greatest influence on the prediction. However, *Gradient* does not account for the scale of the input, which can limit its interpretability.

The second approach, *Gradient*Input* [28], refines this idea by multiplying the gradient with the corresponding input value:

$$\boldsymbol{\phi} = \left[\frac{\partial \hat{y}}{\partial x_1} x_1, \ldots, \frac{\partial \hat{y}}{\partial x_T} x_T\right]. \tag{7}$$

Geometrically, *Gradient*Input* computes the directional derivative along the direction defined by the input itself. This approach enhances interpretability by considering both the gradient and the magnitude of the input.

Finally, we consider *GradientSHAP* [21], an approximation of Integrated Gradients [34] that introduces a baseline, i.e., a reference point against which feature contributions are measured. The method estimates the impact of each feature by comparing the model's predictions on actual input data to its predictions on baseline data. It computes the expected gradients over a distribution of interpolated inputs between the input $\mathbf{x}$ and a baseline $\mathbf{z}$:

$$\boldsymbol{\phi} = \mathbb{E}_{\mathbf{x}' \sim D(\mathbf{z}, \mathbf{x})}\left[\frac{\partial \hat{y}}{\partial x_1}(x_1 - x_1'), \ldots, \frac{\partial \hat{y}}{\partial x_T}(x_T - x_T')\right]. \tag{8}$$

Here, $\mathbf{x}'$ denotes samples drawn from the distribution $D(\mathbf{z}, \mathbf{x})$, defined as the uniform distribution over linear interpolations between the baseline $\mathbf{z}$ and the input $\mathbf{x}$. The term $\frac{\partial \hat{y}}{\partial x_t}(x_t - x_t')$ quantifies the contribution of each feature by weighting the gradient with the difference between the input and its corresponding perturbed version. By averaging over multiple such perturbations, *GradientSHAP* provides a more stable estimate of feature importance, with respect to the previous approaches, reducing sensitivity to local variations and noise while capturing the effect of each input dimension relative to the chosen baseline.

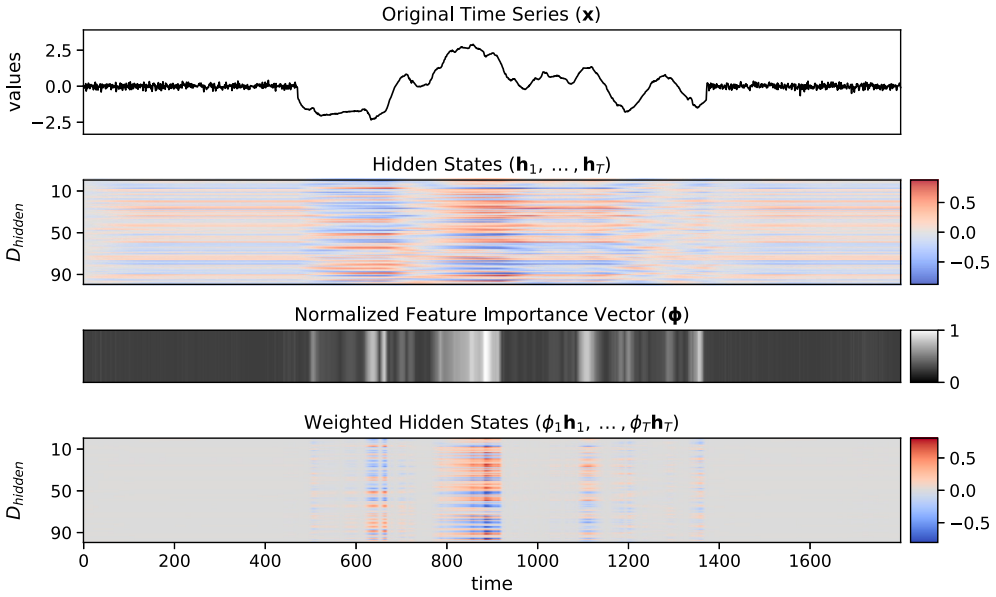


Fig. 1. Top: a time series from the *WormsTwoClass* dataset with Gaussian noise at the beginning and end. Second from the top: the raw hidden states for each timestep computed by the ESN. Third from the top: the importance of each timestep of the input time series from *Gradient*Input*. Bottom: the hidden states weighted by their importance.

These approaches are easily extended to multiclass tasks by independently computing the contribution toward each of the classification targets. In summary, each of these methods offers a different perspective on how the model processes input data, with *Gradient* capturing local sensitivity, *Gradient*Input* incorporating feature magnitude, and *GradientSHAP* providing a more global, expectation-based attribution mechanism.

4. Enhancing randomized recurrent neural networks

The core idea of this work is to enhance randomized RNNs by weighting the reservoir output, based on the importance of each observation in a time series $\mathbf{x}$. For this purpose, we propose extending the readout function f_{res}, introducing a generalized method for aggregating hidden states, formally:

$$\mathbf{h}' = f_{\text{res}}(\mathbf{x}, \boldsymbol{\phi}) = \frac{\sum_{t=1}^{T} \phi_t \mathbf{h}_t}{\sum_{t=1}^{T} \phi_t}, \tag{9}$$

where $\mathbf{h}_t$ is the hidden state at time t, obtained from one of the methods introduced in Section 3, and $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_T]$ is a vector collecting all scalar weights assigned to each hidden state. The aggregated state $\mathbf{h}'$ is provided as input to the readout to perform classification. Eq. (9) is a general form that encompasses existing aggregation approaches. Consider, for example, a weight vector $\boldsymbol{\phi}$ consisting solely of zeros except for the final element, which is one, i.e., $\boldsymbol{\phi} = [0, 0, \ldots, 0, 1]$. In this scenario, the aggregation process ignores all hidden states except for the last, $\mathbf{h}_T$. This is the most common approach to classification. Alternatively, if $\boldsymbol{\phi} = \left[\frac{1}{T}, \frac{1}{T}, \ldots, \frac{1}{T}\right]$, i.e., if we assign an equal weight of $\frac{1}{T}$ to each hidden state, the aggregation method transitions to computing the average of all hidden states, i.e., $\bar{\mathbf{h}}$. Beyond these specific scenarios, any other weight configuration produces a weighted average, $\mathbf{h}'$.

Our intuition is that assigning larger weights to the hidden states corresponding to the most relevant parts of the input time series can yield better performance than simpler methods such as averaging the states or using only the last hidden state. Consider, for example, Fig. 1. On top, we have a time series $\mathbf{x}$ with Gaussian noise at the beginning and the end. Below the time series, we have the 100-dimensional hidden state computed for each time step of the time series. Intuitively, selecting the last hidden state to classify such a time series would be suboptimal, as the information for classification is mixed with noisy values. Similarly, taking the average of the hidden states would dilute the information contained in the hidden states, given that approximately half of this time series is composed of Gaussian noise.

Our challenge is to find a weight vector $\boldsymbol{\phi}$ such that $\phi_i > \phi_j$ when x_i contains values that are important or discriminative for the task, whereas x_j contains values that are not, such as noise. Naturally, our approach does not know where the noise is positioned in advance but only receives the raw data as input. It is important to compute the weight vector efficiently, possibly without requiring an iterative optimization process. One of the main advantages of randomized RNNs is that they do not require learning in the recurrent component. Introducing an optimization problem over the hidden states' trajectory would reduce this benefit.

Hence, we propose inferring the weight vector $\boldsymbol{\phi}$ using attribution methods in a one-shot fashion, i.e., train a *base model* on the time series classification task, compute the weights $\boldsymbol{\phi}$ using an XAI technique from Section 3, and retrain the readout with the aggregated hidden state $\mathbf{h}'$. Notably, the readout is only retrained once and, since it is a linear predictor, the closed-form solution can be computed very efficiently. The procedure is detailed step-by-step in Algocf Algorithm 1 for training and Algocf Algorithm 2 for inference. Both algorithms cover the binary classification case, while an extension to multiclass classification is presented in the last paragraph of this section.

Training. The training procedure in Algocf Algorithm 1 requires a training set, X, with labels $\mathbf{y}$, as well as reservoir and readout functions, f_{res} and f_{read}, respectively, along with an attribution method, e. The process begins with the reservoir generating outputs for the entire training set. The hidden states, $\bar{\mathbf{h}}$, are equally weighted using the (unweighted) average, $\frac{1}{T}$ (line 1). We employ this average as the *base model* because, empirically, it demonstrates greater stability compared to the common approach of considering only the last hidden state. Next, the readout function is trained on the averaged hidden states (line 2). Following this, the attribution method is initialized and fitted (line 3). This phase does not involve any parameter optimization but simply provides the attribution method with a *background* dataset, which is used to perform perturbations necessary for computing the gradients of the outputs with

Algorithm 1: Training.

Data: $\mathbf{X}$ - training set, $\mathbf{y}$ - training labels, f_{res} - reservoir function, f_{read} - readout function, e - attribution method,
Result: f_{read} - trained readout, e - fitted attribution method

1 $\bar{\mathbf{h}} \leftarrow f_{\text{res}}(\mathbf{X}, \frac{1}{T})$; // Unweighted average of the hidden states
2 $f_{\text{read}} \leftarrow train(f_{\text{read}}, \bar{\mathbf{h}}, \mathbf{y})$; // Train the readout
3 $e \leftarrow fit(e, \mathbf{X})$; // Fit the attribution method
4 $\mathbf{\Phi} \leftarrow e(\mathbf{X}, f_{\text{res}}, f_{\text{read}})$; // Get feature importance
5 $\mathbf{h}' \leftarrow f_{\text{res}}(\mathbf{X}, \mathbf{\Phi})$; // Weighted average of the hidden states
6 $f_{\text{read}} \leftarrow train(f_{\text{read}}, \mathbf{h}', \mathbf{y})$; // Re-train the readout
7 **return** f_{read}, e

Algorithm 2: Inference.

Data: $\mathbf{X}$ - test set, f_{res} - reservoir function, f_{read} - trained readout function, e - fitted attribution method,
Result: $\hat{\mathbf{y}}$ - predicted labels

1 $\mathbf{\Phi} \leftarrow e(\mathbf{X}, f_{\text{res}}, f_{\text{read}})$; // Get feature importance
2 $\mathbf{h}' \leftarrow f_{\text{res}}(\mathbf{X}, \mathbf{\Phi})$; // Weighted average of the hidden states
3 $\hat{\mathbf{y}} \leftarrow f_{\text{read}}(\mathbf{h}', \mathbf{y})$; // Predict labels
4 **return** $\hat{\mathbf{y}}$

respect to the inputs. Once trained, the attribution vectors for each time series are computed and stored in the matrix $\mathbf{\Phi}$. The attribution function, e, not only requires data but must also access both the reservoir and readout functions to compute the gradients. The output of these attribution methods consists of values ranging from negative to positive, indicating the contribution toward either class. Since our focus is on the magnitude of the importance vector rather than its sign, we take the absolute value, formally:

$$\boldsymbol{\phi} = [|\phi_1|, \dots, |\phi_T|]. \tag{10}$$

Fig. 1 (third from the top) shows the importance vector extracted by *Gradient*Input* (the lighter the color, the more important the observation), and at the bottom, the hidden states weighted by the importance vector. Once $\mathbf{\Phi}$ is computed, it is used to weight the output of the reservoir (line 5), which is used as a new training input for the readout (line 6). The outputs of this training phase are a trained readout function and an attribution method.

Inference. The inference phase, outlined in Algocf Algorithm 2, takes as input the test set, the reservoir function, and the trained readout and attribution method. This phase consists of three steps. First, the attribution method is applied to the test set using the reservoir with simple average weights and the trained readout to compute the new weights (line 1). These weights are then used to compute the weighted hidden states for the test set (line 2). Finally, the readout function utilizes these weighted hidden states to predict the test labels (line 3).

Multiclass tasks. For multiclass tasks, algocfs Algorithms 1 and 2 must be slightly adapted. The main issue is that the attribution method provides C different ways to weigh each of the T hidden states, i.e., $[\boldsymbol{\phi}_0, \dots, \boldsymbol{\phi}_{C-1}] \in \mathbb{R}^{T \times C}$. Thus, the simplest way to deal with this problem is to aggregate them across the class axis. We propose the most natural option, i.e., after Eq. (10), taking the sum, which corresponds to considering the overall contribution of the t-th timestep, formally:

$$\phi_t^{\text{sum}} = \sum_{c=0}^{C-1} \phi_{t,c}. \tag{11}$$

In Section 6, we present preliminary experiments using this approach.

5. Mathematical analysis

We conduct a theoretical investigation of the recurrent dynamics in the context of weighted aggregation of hidden states driven by XAI techniques, offering insights supported by linear stability analysis into the effectiveness of our approach.

Theorem 5.1. *A randomized RNN with weights $(\mathbf{W}, \mathbf{V}, \mathbf{b})$, as defined in Eq. (1), that sends a weighted average of its hidden states to the output classifier, as defined in Eq. (9), is equivalent to the following randomized recurrent formulation of the second order:*

$$\mathbf{h}'_t = \mathbf{h}'_{t-1} + \phi_t \, \sigma\Big(\phi_{t-1}^{-1}\mathbf{W}(\mathbf{h}'_{t-1} - \mathbf{h}'_{t-2}) + \mathbf{V}x_t + \mathbf{b}\Big), \tag{12}$$

which sends the last hidden state to the output classifier. In Eq. (12), the weights ϕ_t are assumed to be normalised to have the total sum equal to 1, i.e. $\sum_{t=1}^{T} \phi_t = 1$.

Proof. Let us define the new hidden state variable $\mathbf{h}'_t = \phi_1\mathbf{h}_1 + \phi_2\mathbf{h}_2 + \cdots + \phi_t\mathbf{h}_t$, assuming we have normalised the weights to have sum $\sum_{t=1}^{T} \phi_t = 1$. In particular, we can write $\mathbf{h}'_t = \mathbf{h}'_{t-1} + \phi_t\mathbf{h}_t$, from which, it holds:

$$\mathbf{h}_t = \frac{\mathbf{h}'_t - \mathbf{h}'_{t-1}}{\phi_t}. \tag{13}$$

The ESN state-update equation reads $\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{V}x_t + \mathbf{b})$, which can be equivalently written in the $\mathbf{h}'$ variable as $\frac{\mathbf{h}'_t - \mathbf{h}'_{t-1}}{\phi_t} = \sigma(\mathbf{W}(\frac{\mathbf{h}'_{t-1} - \mathbf{h}'_{t-2}}{\phi_{t-1}})) + \mathbf{V}x_t + \mathbf{b})$. This leads to the following recurrence of the second order in the variable $\mathbf{h}'$:

$$\mathbf{h}'_t = \mathbf{h}'_{t-1} + \phi_t \, \sigma\Big(\phi_{t-1}^{-1}\mathbf{W}(\mathbf{h}'_{t-1} - \mathbf{h}'_{t-2}) + \mathbf{V}x_t + \mathbf{b}\Big).$$

□

Remark 5.1. Eq. (12) effectively models a *residual RNN* where the nonlinear residual depends on the previous two hidden states at times $t-1$ and $t-2$, the reciprocal of previous SHAP value at time $t-1$, the current SHAP value at time t, and the current input at time t. Intuitively, the new hidden state $\mathbf{h}'_t$ undergoes a significant change from $\mathbf{h}'_{t-1}$ whenever the weight ϕ_t takes a value substantially greater than zero.

To better understand and justify the advantages of our approach for long-range information propagation, we derived an equivalent formulation of our model (Theorem 5.1) that reveals a key mathematical property: the spectral radius of the Jacobian can be systematically biased toward the value of 1. This is a crucial insight, as a spectral radius near 1 implies sustained information flow, mitigating vanishing or exploding of information over long sequences or deep computations. By formalizing this behavior, we demonstrate how our model inherently supports more stable and effective transmission of information across extended temporal or spatial dependencies, providing a theoretical foundation for its superior performance in tasks requiring long-range interactions.

We start by analyzing the Jacobian of the second order recurrent equivalent formulation of our approach. Let us denote $\mathbf{H}_t = \begin{pmatrix} \mathbf{h}'_t \\ \mathbf{h}'_{t-1} \end{pmatrix}$. This allows us to write the second order recurrence of Eq. (12) as a first order recurrence in the augmented hidden state $\mathbf{H}_t$, as: $\mathbf{H}_t = \mathbf{F}(\mathbf{H}_{t-1}, x_t)$. More precisely, it can be written as

$$\mathbf{H}_t = \begin{pmatrix} \mathbf{f}(\mathbf{H}_{t-1}, x_t) \\ \mathbf{h}'_{t-1} \end{pmatrix} = \begin{pmatrix} \mathbf{h}'_{t-1} + \phi_t \, \sigma\Big(\phi_{t-1}^{-1}\mathbf{W}(\mathbf{h}'_{t-1} - \mathbf{h}'_{t-2}) + \mathbf{V}x_t + \mathbf{b}\Big) \\ \mathbf{h}'_{t-1} \end{pmatrix}.$$

Thus, the Jacobian $\mathbf{J}_t = \frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_{t-1}}$, is the following block matrix:

$$\mathbf{J}_t = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{h}'_{t-1}} & \frac{\partial \mathbf{f}}{\partial \mathbf{h}'_{t-2}} \\ \frac{\partial \mathbf{h}'_{t-1}}{\partial \mathbf{h}'_{t-1}} & \frac{\partial \mathbf{h}'_{t-1}}{\partial \mathbf{h}'_{t-2}} \end{bmatrix} = \begin{bmatrix} \mathbf{I} + \mathbf{M}_t & -\mathbf{M}_t \\ \mathbf{I} & \mathbf{0} \end{bmatrix}, \tag{14}$$

where

$$\mathbf{M}_t = \phi_t \mathrm{diag}(\sigma'_{t-1})\phi_{t-1}^{-1}\mathbf{W}^\top. \tag{15}$$

In the reservoir computing literature, a commonly accepted rule of thumb [19] is to configure the recurrent weight matrix $\mathbf{W}$ such that its spectral radius is around 1. This condition promotes the so-called echo state property, which guarantees that the influence of past inputs on the network state gradually fades over time.

Numerous studies have demonstrated that, as the spectral radius of $\mathbf{W}$ approaches 1, the network exhibits an increased short-term memory capacity [5,15]. In other words, a spectral radius close to 1 allows the network to retain information about past inputs for longer periods, enhancing its ability to process temporal dependencies.

This behavior has been shown to be critical for the effective classification of time series using RNNs, particularly when employing only the final hidden state as a compact representation—or encoding—of the entire input sequence [12]. In such cases, preserving temporal information across time steps becomes essential for accurate downstream predictions.

In this context, we present the following theorem, which formalizes the relationship between the spectral radius of $\mathbf{W}$ and the corresponding spectral characteristics of the equivalent second-order RNN defined in Eq. (12).

Theorem 5.2. *The following XAI-driven upper bound holds for the spectral radius of $\mathbf{J}_t$:*

$$\rho(\mathbf{J}_t) \leq \max\left(1, \frac{\phi_t}{\phi_{t-1}}\ \rho(\mathbf{W})\right). \tag{16}$$

In particular, if $\phi_t \leq \phi_{t-1}$ then the spectral radius tends to decrease, while if $\phi_t > \phi_{t-1}$, then the spectral radius tends to increase.

Proof. We will need Lemma 5.1 to prove this theorem. Lemma 5.1 implies that the spectrum of the Jacobian $\mathbf{J}_t$ is composed of the union of $\{1\}$ and the spectrum of $\mathbf{M}_t = \phi_t \mathrm{diag}(\sigma'_{t-1})\phi_{t-1}^{-1}\mathbf{W}^\top$. In particular, we have that $\rho(\mathbf{J}_t) = \max(1, \rho(\mathbf{M}_t))$. Define $\mathbf{D}_t = \mathrm{diag}(\sigma'_{t-1})$. Now, since the entries of $\mathbf{D}_t$ are all in $[0, 1]$, then we have that $\rho(\mathbf{D}_t\mathbf{W}) \leq \rho(\mathbf{W})$. Moreover, since $\mathbf{M}_t = \frac{\phi_t}{\phi_{t-1}}\mathbf{D}_t\mathbf{W}$, and $\frac{\phi_t}{\phi_{t-1}}$ is a positive scalar value, it implies that $\rho(\mathbf{M}_t) = \frac{\phi_t}{\phi_{t-1}}\rho(\mathbf{D}_t\mathbf{W}) \leq \frac{\phi_t}{\phi_{t-1}}\rho(\mathbf{W})$, from which it follows the thesis, $\rho(\mathbf{J}_t) \leq \max\left(1, \frac{\phi_t}{\phi_{t-1}}\ \rho(\mathbf{W})\right)$. □

Lemma 5.1. *Consider the block matrix*

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}+\mathbf{M} & -\mathbf{M} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n \times 2n}, \tag{17}$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix, $\mathbf{0} \in \mathbb{R}^{n\times n}$ is the zero matrix, and $\mathbf{M} \in \mathbb{R}^{n\times n}$ is a square matrix. Then, the eigenvalues of $\mathbf{J}$ are either the eigenvalues of $\mathbf{M}$ (with the same multiplicity within $\mathbf{M}$) or 1 with multiplicity n.

Proof. See Appendix A for the proof. □

Remark 5.2. Theorem 5.2 has a nice geometric interpretation. If the SHAP values decrease from time step $t-1$ to time step t, i.e. if $\phi_t \leq \phi_{t-1}$ then the equivalent second-order RNN formulation acts by shrinking the volume of information in the space of hidden states at time t. This contraction of the information is guided by the lower importance as indicated by the SHAP calculation. On the other hand, if the SHAP values increase from time step $t-1$ to time step t, i.e. if $\phi_t > \phi_{t-1}$, then the equivalent second-order RNN formulation acts by expanding the volume of information in the space of hidden states at time t. This expansion of the information is guided by the higher importance as indicated by the SHAP calculation. Figuratively speaking, our approach uses SHAP values as an XAI-informed gate mechanism for controlling information flow—selectively modulating the extent to which input information is integrated at each time step, akin to the gating mechanisms in GRU and LSTM networks.

Table 1
Dataset description, in terms of number of instances in the training (N_{train}) and test sets (N_{test}), length (T), number of classes (C), and type.

Dataset	N_{train}	N_{test}	T	C	type
Adiac	390	391	176	37	Image
ArrowHead	36	175	251	3	Image
BME	30	150	128	3	Simulated
Beef	30	30	470	5	Spectro
BeetleFly	20	20	512	2	Image
BirdChicken	20	20	512	2	Image
Chinatown	20	343	24	2	Traffic
Coffee	28	28	286	2	Spectro
Computers	250	250	720	2	Device
DistalPhalanxOutlineCorrect	600	276	80	2	Image
DodgerLoopGame	20	138	288	2	Sensor
DodgerLoopWeekend	20	138	288	2	Sensor
ECG200	100	100	96	2	ECG
ECGFiveDays	23	861	136	2	ECG
Earthquakes	322	139	512	2	Sensor
Fish	175	175	463	7	Image
Fungi	18	186	201	18	HRM
GunPoint	50	150	150	2	Motion
GunPointAgeSpan	135	316	150	2	Motion
GunPointOldVersusYoung	136	315	150	2	Motion
Ham	109	105	431	2	Spectro
Haptics	155	308	1092	5	Motion
Herring	64	64	512	2	Image
ItalyPowerDemand	67	1029	24	2	Sensor
Lightning7	70	73	319	7	Sensor
Meat	60	60	448	3	Spectro
MiddlePhalanxOutlineCorrect	600	291	80	2	Image
PhalangesOutlinesCorrect	1800	858	80	2	Image
Plane	105	105	144	7	Sensor
PowerCons	180	180	144	2	Power
ProximalPhalanxOutlineCorrect	600	291	80	2	Image
ShapeletSim	20	180	500	2	Simulated
SonyAIBORobotSurface1	20	601	70	2	Sensor
SonyAIBORobotSurface2	27	953	65	2	Sensor
Strawberry	613	370	235	2	Spectro
ToeSegmentation1	40	228	277	2	Motion
ToeSegmentation2	36	130	343	2	Motion
TwoLeadECG	23	1139	82	2	ECG
Wine	57	54	234	2	Spectro
WormsTwoClass	181	77	900	2	Motion

In summary, this formulation provides insight into why the proposed approach may lead to improved performance. By recasting the hidden states weighted aggregation as a second-order recurrent model, the analysis reveals how the importance weights, derived from attribution methods, act as a modulation mechanism for the flow of information across time. Specifically, the spectral radius of the Jacobian is automatically adapted by the local ratio of consecutive weights. For this reason, we can avoid tuning the spectral radius, one of the most sensitive hyperparameters for reservoir computing models, and use the same spectral radius across all experiments. In the following empirical evaluation, we test these theoretical properties on time series classification, showing how the proposed mechanism selectively integrates informative segments of the time series while attenuating the influence of irrelevant or noisy parts.

6. Experiments

We perform tests on four different Randomized RNN: the Echo State Network (ESN) [16], the Random Oscillators Network (ron) [4], the Residual Echo State Network (RES-ESN) [6], and the Euler State Network (EUSN) [12]. For each RNN, we consider as baselines the most common approaches for aggregating hidden states, namely *(i)* using only the last hidden state (last) and *(ii)* the average of all hidden states (AVG), as well as a naive reference *(iii)* that computes a weighted average of hidden states with randomly-generated weights (RND). The latter serves as a simple sanity check to evaluate whether

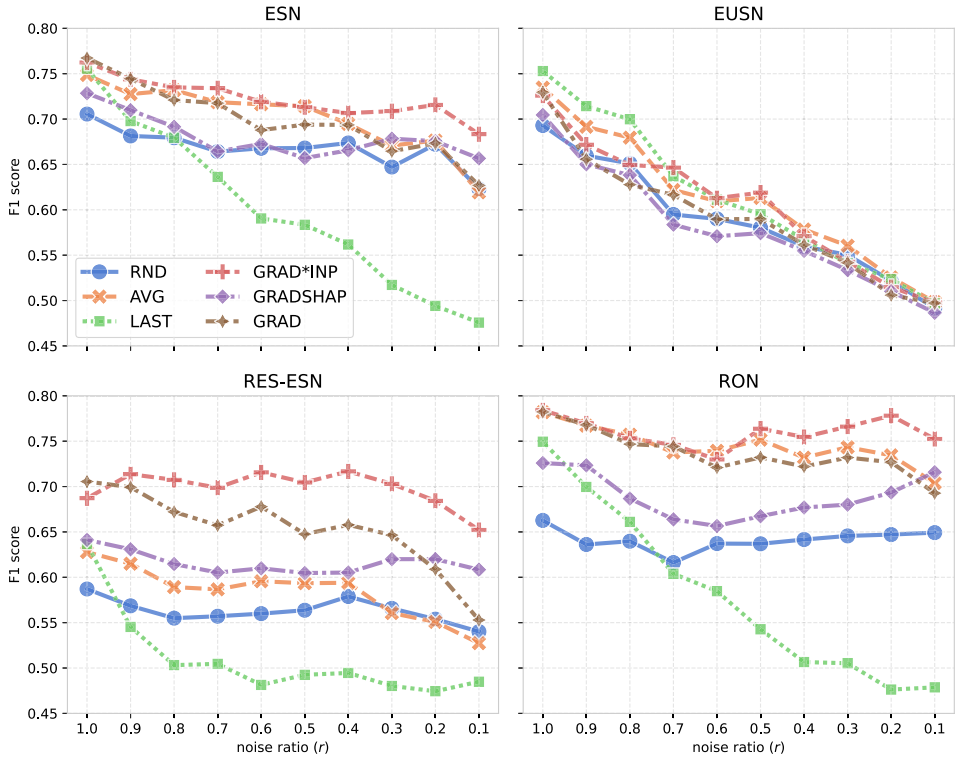


Fig. 2. Average F1 of the benchmarked aggregation approaches for the 30 **binary classification** UCR datasets, and for each of the four randomized RNNs, starting from no added noise ($r = 1.0$), to 10x noise ($r = 0.1$).

the other methods capture meaningful structure beyond what could be obtained by chance. Then, in terms of our proposal, for each RNN we consider our weighted average of hidden states computed using 3 different XAI approaches: *Gradient* (grad), *Gradient*Input* [28] (grad*inp), and *GradientSHAP* (gradshap) [21].

We show how our approach is able to improve learning in randomized RNNs effectively across a wide variety of time series classification benchmarks without ad-hoc tuning of hyperparameters on each benchmark. This approach aligns with recent studies on time series classification [23,25,31], which prioritize building classifiers that generalize well across tasks using a reasonable set of hyperparameters, rather than fine-tuning for optimal performance on specific tasks. Our goal is to isolate the effect of different aggregation methods as much as possible; thus, in this case, excessive hyperparameter fine-tuning would confound the comparison and be detrimental to our objective. Further, for reservoir computing models, we have several theoretical insights that help us choose reasonable values for the hyperparameters (for example, the spectral radius is usually set very close, but smaller than 1 due to the Echo-State Property).

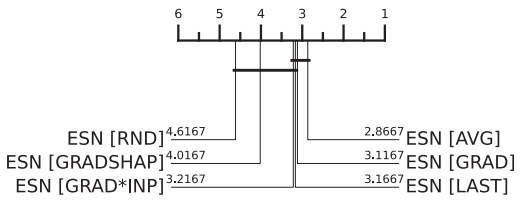
All networks feature fully connected reservoirs with 100 units, a spectral radius of 0.99, and an input scaling of 1 [19]. Specifically, the ESN employs a leak rate of 0.01. The ron model is characterized by $\boldsymbol{\gamma} = \mathbf{1}$, $\boldsymbol{\epsilon} = \mathbf{1}$, and $\tau = 0.1$ [4], while RES-ESN is defined by $\alpha = 0.9$ and $\beta = 0.1$ [6]. Finally, EUSN operates with $\epsilon = 0.01$, $\gamma = 0.1$ [12]. To determine the importance of the points in the time series, we need a *base model* to retrieve them. For this purpose, we use AVG, given that it is the most stable baseline against noise, as shown in the following experiments. Once $\boldsymbol{\phi}$ is retrieved, we use it to retrain the readout on the weighted average of hidden states. We use LBFGS as the readout optimizer, as it provides stable convergence and efficient training for relatively small networks, achieving good solutions with few iterations. The maximum number of iterations is set to 1000.

We propose two types of benchmarks: binary classification and multiclass classification.[1] For binary classification, we benchmark our approach on 30 datasets from the UCR time series classification repository [9], whose characteristics are summarized in Table 1 ($C = 2$). For the multiclass task, we propose experiments on 10 multiclass datasets also detailed in Table 1 ($C > 2$), using the aggregation proposed in Section 4. We purposely chose datasets with diverse properties, such as the number of instances varying from 20 to 1800, sequence lengths ranging from 20 to 1092, and spanning multiple types, including *Image*, *Traffic*, *Sensor*, *Motion*, *Power*, *ECG* (Electrocardiogram), *Spectro*, *Device*, *HRM* (High Resolution Melt), and *Simulated*. For each dataset, we evaluate the performance of all models both on the original time series and on augmented versions where random Gaussian noise is appended. In particular, the appended noise has a standard deviation equal to 10 % of the original time series. We control the amount of noise through a ratio parameter $0 < r \leq 1$, where $r = 1$ denotes the original series without noise, and smaller values of r correspond to longer appended noisy segments. Specifically, for a time series of length T, the number of noise points added is computed as $T^{noise} = \frac{T}{r} - T$. We vary r from 0.1 (i.e., the noise is ten times the length of the original series) to 1.0 in steps of 0.1. All time series are standardized using Z-score normalization prior to evaluation.

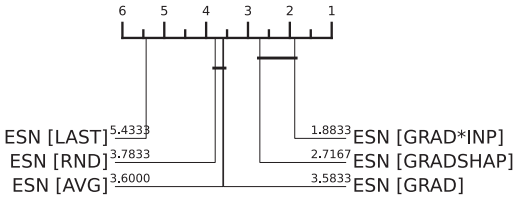
6.1. Binary classification results

In Fig. 2, we report the performance in terms of the F1-score for all approaches, first on the original datasets and then by adding random Gaussian noise to each time series, for all four randomized RNNs: ESN (top-left), EUSN (top-right), RES-ESN (bottom-left), RON (bottom-right). The average across all the datasets with the standard deviation

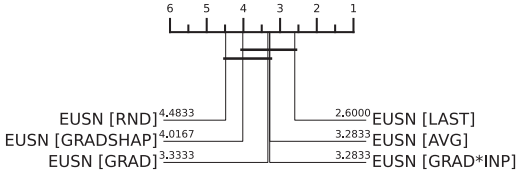
[1] Code and datasets are available at https://github.com/fspinna/xai_enhanced_esn. System: IBM SYSTEM POWER AC922 Compute Nodes with 2.7GHz POWER9 CPUs, limited to 16-threads, 32GB of RAM.



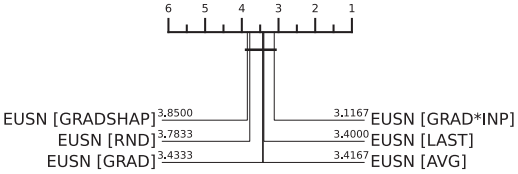
(a) ESN, $r = 1.0$ (no noise)



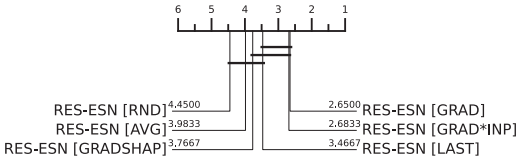
(b) ESN, $r = 0.1$ (10x noise)



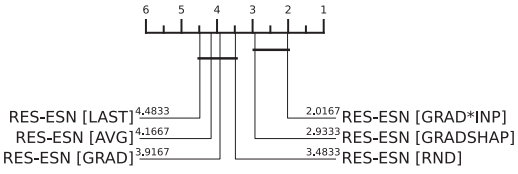
(c) EUSN, $r = 1.0$ (no noise)



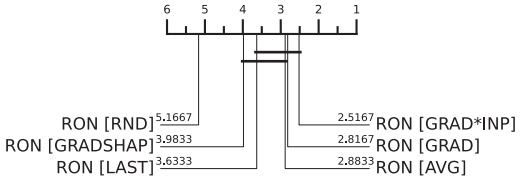
(d) EUSN, $r = 0.1$ (10x noise)



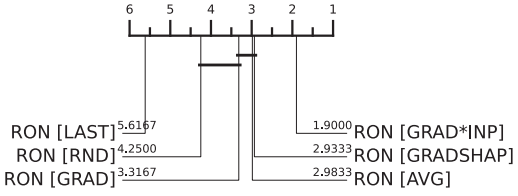
(e) RES-ESN, $r = 1.0$ (no noise)



(f) RES-ESN, $r = 0.1$ (10x noise)



(g) RON, $r = 1.0$ (no noise)



(h) RON, $r = 0.1$ (10x noise)

Fig. 3. Critical Difference Plots for the four different randomized RNNs on **binary classification** datasets in the two extreme noise cases, i.e., $r = 1.0$ (no noise), and $r = 0.1$ (10x noise). Best models have lower rank, and are shown on the right of each plot. Models connected with a horizontal line are statistically tied.

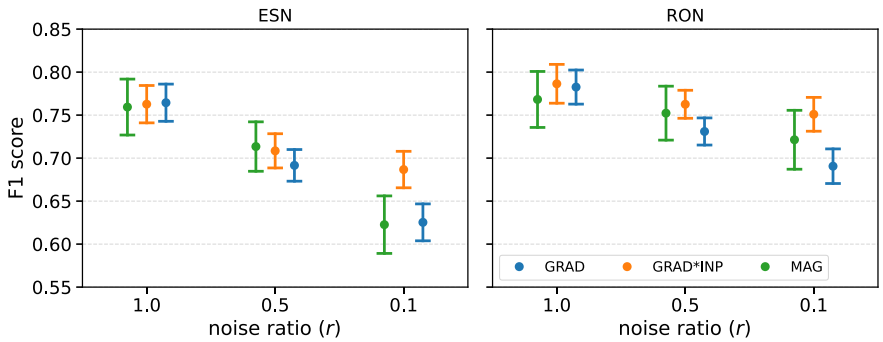


Fig. 4. Average F1 score of the benchmarked aggregation methods on the 30 **binary classification** UCR datasets, for ESN and RON, across noise ratios 1.0 (no noise), 0.5 (medium noise), and 0.1 (high noise). Error bars indicate the standard error of the mean.

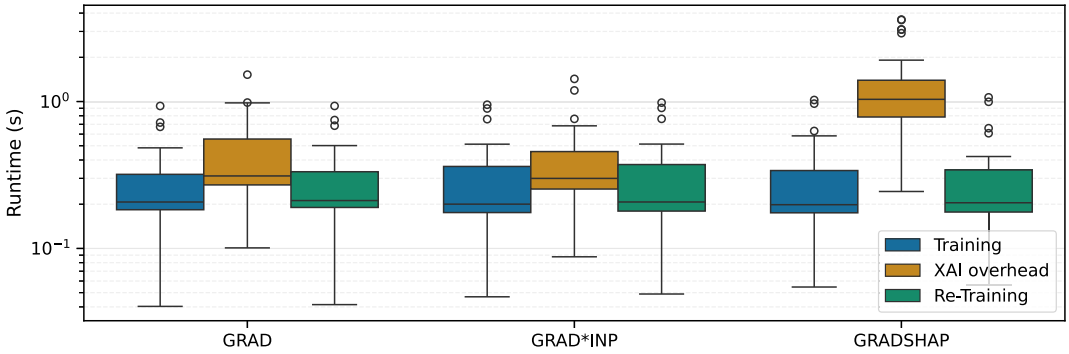


Fig. 5. Boxplots of training, XAI overhead, and retraining runtimes (in seconds) for a standard ESN. Lower is better.

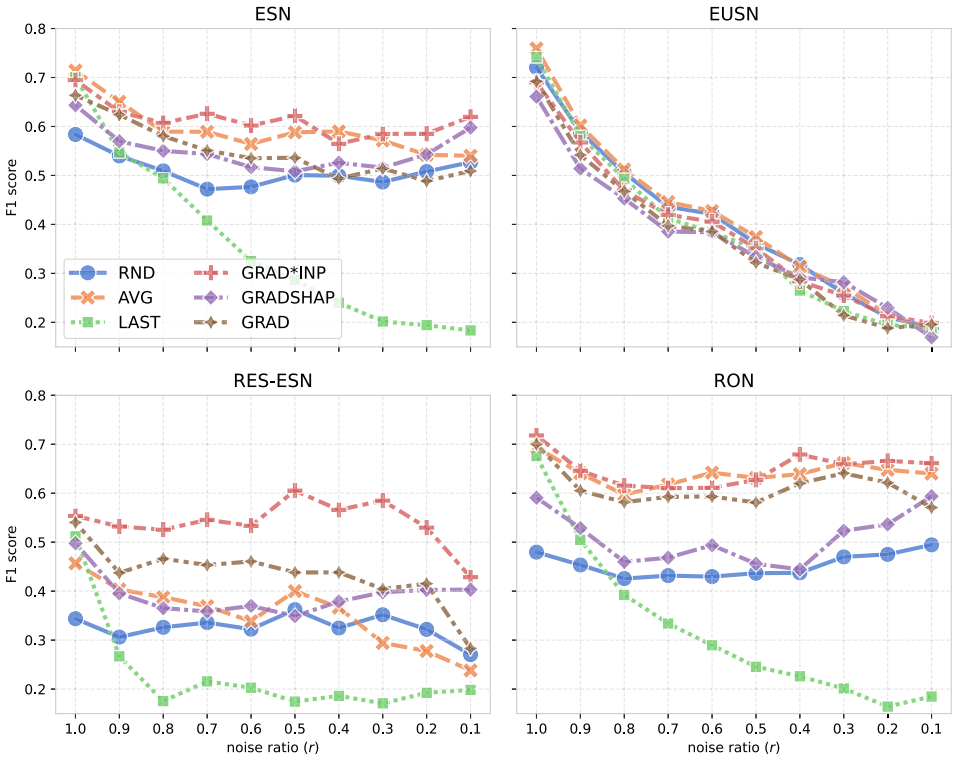


Fig. 6. Average F1 of the benchmarked aggregation approaches for the 10 **multiclass classification** UCR datasets, and for each of the four randomized RNNs, starting from no added noise ($r = 1.0$), to 10x noise ($r = 0.1$).

is also reported in Tables B.2–B.5 in Appendix B. Overall, the most effective aggregation method for ESN, RES-ESN, and RON is GRAD*INP, which consistently achieves strong performance. While results on the original, noise-free datasets (i.e., $r = 1.0$) are close across methods, GRAD*INP clearly outperforms others in the presence of noise, making it the most robust choice overall. On the original datasets, AVG, LAST, and GRAD aggregations are comparable to GRAD*INP. However, LAST's performance drops quickly as the noise increases, while AVG and GRAD follow a less steep trend. Interestingly, their performances become comparable to random weighting as r approaches 0.1 (10x noise). Further, although GRADSHAP is a more sophisticated approach, it performs worse than AVG and GRAD. This is likely due to its random baseline extraction, which could be subpar on time series data. In summary, GRAD*INP is more robust, both on the original data, and in the presence of noise, while its computational overhead is minimal. A notable exception to these experiments is EUSN, for which the performance of all methods decays as noise increases. We hypothesize that, since EUSN operates at the edge of stability, a sufficiently large noisy input can push the dynamics beyond the stability boundary, resulting in poor generalization.

While the average performance is a good indicator of the superiority of GRAD*INP, it is not sufficient to determine whether such differences are statistically significant. For this reason, in Fig. 3, we propose the use of Critical Difference Plots, which offer an intuitive visualization of model performance based on their rank positions in terms of F1 scores, for the two extreme noise settings, i.e., $r = 1.0$ (no noise) and $r = 0.1$ (10x noise). Models are arranged from right to left, with lower ranks indicating better performance. Models connected by a horizontal bar are considered to have statistically tied performance. The CD-plot employs a one-sided Holm-corrected Wilcoxon signed-rank test with a significance threshold of 0.05. With the exception of EUSN, for which nearly all methods are statistically tied, GRAD*INP is either statistically tied with the top-performing models in the absence of noise or emerges as the best model under high noise ($r = 0.1$), being statistically tied only with GRAD*SHAP in a few cases. This indicates that GRAD*INP achieves robust and competitive performance under substantial label noise, while

preserving its effectiveness in noise-free settings. Fig. B.7 presents CD plots for all methods combined, and shows that the best overall approach pairs GRAD*INP hidden states aggregation with RON.

6.2. Input magnitude

To disentangle the contribution of gradient information from that of the input itself, we introduce an additional baseline that simply weights the hidden states by their input magnitude, $|x_t|$ (MAG). This sanity check allows us to assess whether the performance improvements of GRAD*INP arise from explainable attribution or are largely due to input-based weighting.

Results are reported in Fig. 4 for ESN and RON. Each point represents the average F1 score across datasets, with error bars indicating the standard error of the mean, enabling a visual comparison of robustness across different noise levels. They largely follow the same trends observed in Fig. 2. Without noise ($r = 1.0$), all methods perform comparably, although MAG exhibits slightly higher variance. At moderate noise levels ($r = 0.5$), the behavior of MAG closely tracks that of GRAD*INP (again with higher variance), while RON paired with GRAD falls notably behind. Under high noise ($r = 0.1$), however, GRAD*INP clearly outperforms both GRAD and MAG, highlighting the added value of incorporating gradient information. These findings confirm that while input magnitude alone can act as an efficient baseline, capturing part of the signal and filtering out noise, the full GRAD*INP method provides more robust performance across noise regimes. Numerical results are reported in Table B.6.

6.3. Runtime

The overall computational cost of our method can be decomposed into three components: the initial training of the ESN, the computation of weights using XAI methods, and the subsequent re-training. In Fig. 5, we report the boxplots of training time, XAI overhead, and re-training time (in seconds) for a standard ESN using the average (AVG) as the aggregation method (RND and LAST share the same computational complexity), evaluated on the original 30 datasets and depending on the chosen XAI weighting approach. Lower values correspond to faster performance. In general, for GRAD and GRAD*INP, the XAI overhead is comparable to, though slightly higher than, the training and re-training times. Consequently, these approaches are approximately three times slower than a standard randomized recurrent neural network, but still fast in absolute terms, with the interquartile range of runtimes well below one second. The overall computational cost remains modest, especially when considering the typically much higher training times required by fully trainable recurrent neural networks. GRADSHAP introduces a noticeably higher overhead, yet it remains reasonably efficient for our datasets, with a median runtime of around one second. The results for other ESN variants, presented in Fig. B.8, exhibit the same overall trends.

6.4. Multiclass classification results

In Fig. 6, we provide results on 10 multiclass datasets, in terms of the F1-score for all approaches, first on the original datasets and then with added random Gaussian noise to each time series, for all four randomized RNNs: ESN (top-left), EUSN (top-right), RES-ESN (bottom-left), and RON (bottom-right). Similar to the binary classification case, the GRAD*INP approach appears more effective for the ESN, RES-ESN, and RON, whereas EUSN once again shows a consistent performance drop with increasing noise, regardless of the weighting strategy. However, in the multiclass setting, aggregating hidden states using a simple average becomes more competitive. Indeed, while GRAD*INP generally achieves higher rankings than AVG, the CD-plots in Fig. C.9 are less conclusive, indicating statistical ties for all methods except ESN. This suggests that the weighting approach is indeed promising, but more extensive experimentation is required to validate its robustness and generalizability across diverse multiclass scenarios. Further plots and tables can be found in Appendix C.

7. Conclusion

In this work, we have proposed to improve the robustness and accuracy of randomized recurrent neural networks by weighting their hidden states using gradient-based post-hoc attribution techniques. We have provided a theoretical justification for this weighting strategy and extended it to multiclass classification settings. Our extensive experimental evaluation, covering 30 binary and 10 multiclass time series classification tasks, across four randomized RNNs, has demonstrated that explainability-guided weighting can lead to improved performance, particularly in noisy scenarios. This marks a step toward rethinking the role of explainable AI, moving beyond purely interpretive use toward a more integrated function that can support and enhance the learning process. Such integration strengthens the synergy between the model and the explanations it produces, enabling more adaptive and effective learning mechanisms with potential impact on real-world applications.

A current limitation of our proposal is its restriction to univariate time series. In future work, we aim to generalize the approach to multivariate time series, a non-trivial task due to the need to model inter-channel dependencies. We also plan to explore other learning tasks, such as time series regression and forecasting, to assess the generality of our method across broader temporal modeling challenges.

CRediT authorship contribution statement

Francesco Spinnato: Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Andrea Ceni:** Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Andrea Cossu:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Riccardo Guidotti:** Writing – review & editing, Writing – original draft, Supervision, Funding acquisition. **Claudio Gallicchio:** Writing – review & editing, Writing – original draft, Methodology, Funding acquisition. **Davide Bacciu:** Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study has been partially funded by the Italian Project Fondo Italiano per la Scienza FIS00001966 “MIMOSA”, by the European Community Horizon 2020 programme under the funding schemes ERC-2018-ADG G.A. 834756 “XAI”, G.A. 101070212 “FINDHR”, G.A. 101120763 “TANGO”, by the European Commission under the NextGeneration EU programme - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) Project: “SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics” - Prot. IR0000013 - Av. n. 3264 del 28/12/2021, and M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - “FAIR” - Future Artificial Intelligence Research” - Spoke 1 “Human-centered AI”, EU EIC project EMERGE (Grant No. 101070918). The authors gratefully acknowledge the contributors of the datasets available through the UEA & UCR Repositories. Their efforts in collecting and sharing these datasets have been invaluable to this research.

Appendix A. Proof of Lemma 5.1

Here is the detailed proof of Lemma 5.1. For convenience, we repeat the statement to be proved.

Lemma A.1. *Consider the block matrix*

$$\mathbf{J} = \begin{bmatrix} \mathbf{I} + \mathbf{M} & -\mathbf{M} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n \times 2n},$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix, $\mathbf{0} \in \mathbb{R}^{n\times n}$ is the zero matrix, and $\mathbf{M} \in \mathbb{R}^{n\times n}$ is a square matrix. Then, the eigenvalues of $\mathbf{J}$ are either the eigenvalues of $\mathbf{M}$ (with the same multiplicity within $\mathbf{M}$) or 1 with multiplicity n.

Proof. Consider the block matrix

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}+\mathbf{M} & -\mathbf{M} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2\,n\times 2\,n},$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix, $\mathbf{0} \in \mathbb{R}^{n\times n}$ is the zero matrix, and $\mathbf{M} \in \mathbb{R}^{n\times n}$ is a square matrix. We are interested in the eigenvalues and their multiplicities for $\mathbf{J}$, in terms of the eigenvalues of $\mathbf{M}$. Let $\lambda \in \mathbb{C}$ be an eigenvalue of $\mathbf{J}$, with corresponding eigenvector $\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \in \mathbb{C}^{2\,n}$. Then,

$$\mathbf{J}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} = \lambda\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \quad \Rightarrow \quad \begin{cases} (\mathbf{I}+\mathbf{M})\mathbf{x} - \mathbf{M}\mathbf{y} = \lambda\mathbf{x} \\ \mathbf{x} = \lambda\mathbf{y} \end{cases}. \tag{A.1}$$

Substituting $\mathbf{x} = \lambda\mathbf{y}$ into the first equation gives:

$$(\mathbf{I}+\mathbf{M})(\lambda\mathbf{y}) - \mathbf{M}\mathbf{y} = \lambda^2\mathbf{y} \tag{A.2}$$

$$\Rightarrow \left[\lambda(\mathbf{I}+\mathbf{M}) - \mathbf{M} - \lambda^2\mathbf{I}\right]\mathbf{y} = 0. \tag{A.3}$$

Simplifying the expression:

$$\left[\lambda\mathbf{I} + \lambda\mathbf{M} - \mathbf{M} - \lambda^2\mathbf{I}\right]\mathbf{y} = 0 \tag{A.4}$$

$$\Rightarrow [\mathbf{M}(\lambda-1) + \lambda(1-\lambda)\mathbf{I}]\,\mathbf{y} = 0. \tag{A.5}$$

Let μ be an eigenvalue of $\mathbf{M}$, with eigenvector $\mathbf{v}$, so $\mathbf{M}\mathbf{v} = \mu\mathbf{v}$. Substituting into the above gives the scalar condition:

$$[\mu(\lambda-1) + \lambda(1-\lambda)]\,\mathbf{v} = 0. \tag{A.6}$$

Thus,

$$\mu(\lambda-1) + \lambda(1-\lambda) = 0 \to \lambda^2 - (\mu+1)\lambda + \mu = 0. \tag{A.7}$$

This quadratic equation factors as:

$$\lambda^2 - (\mu+1)\lambda + \mu = (\lambda-1)(\lambda-\mu). \tag{A.8}$$

Therefore, for each eigenvalue μ of $\mathbf{M}$, the matrix $\mathbf{J}$ has two eigenvalues:

$$\lambda = \mu, \quad \lambda = 1. \tag{A.9}$$

Let $\mu_1, \ldots, \mu_k$ be the distinct eigenvalues of $\mathbf{M}$, with corresponding algebraic multiplicities $m_{\mu_1}, \ldots, m_{\mu_k}$. Then, we have that:

- the algebraic multiplicity of $\lambda = \mu_i$ in $\mathbf{J}$ is m_{μ_i};
- the algebraic multiplicity of $\lambda = 1$ in $\mathbf{J}$ is:

$$\sum_{i=1}^{k} m_{\mu_i} = n. \tag{A.10}$$

It remains to determine geometric multiplicities. For that purpose, notice that (A.5) can be written as:

$$[\mathbf{M}(\lambda-1) + \lambda(1-\lambda)\mathbf{I}]\mathbf{y} = 0 \Rightarrow (\lambda-1)(\mathbf{M} - \lambda\mathbf{I})\mathbf{y} = 0, \tag{A.11}$$

so the geometric multiplicity of $\lambda = \mu$ in $\mathbf{J}$ equals that of μ in $\mathbf{M}$.

For $\lambda = 1$, the eigenvalue equation becomes:

$$[\mathbf{M}(1-1) + 1(1-1)\mathbf{I}]\mathbf{y} = 0 \Rightarrow 0 = 0 \tag{A.12}$$

which imposes no restriction on $\mathbf{y}$. On the other hand, from the original system of (A.1), we have $\mathbf{x} = \mathbf{y}$, thus the first equation in (A.1) becomes:

$$(\mathbf{I}+\mathbf{M})\mathbf{x} - \mathbf{M}\mathbf{x} = \mathbf{x} \Rightarrow \mathbf{x} = \mathbf{x}. \tag{A.13}$$

Thus, any vector of the form $\begin{bmatrix}\mathbf{x}\\ \mathbf{x}\end{bmatrix}$ is an eigenvector for $\lambda = 1$, and the geometric multiplicity is:

$$\dim\left\{\begin{bmatrix}\mathbf{x}\\ \mathbf{x}\end{bmatrix} : \mathbf{x} \in \mathbb{C}^n\right\} = n \tag{A.14}$$

□

Appendix B. Tables and plots for binary classification

See Tables B.2–B.6, Figs. B.7 and B.8

Table B.2
Average F1 score and standard deviation of each attribution method using ESN across all **binary classification** UCR datasets, evaluated from noise-free conditions ($r = 1.0$) to high noise levels ($r = 0.1$, i.e., $10\times$ noise). Higher scores indicate better performance; best results are highlighted in bold.

r	RND	AVG	LAST	GRAD	GRAD*INP	GRADSHAP
1.0	0.71 ± 0.16	0.75 ± 0.17	0.76 ± 0.16	**0.77** ± 0.17	0.76 ± 0.17	0.73 ± 0.15
0.9	0.68 ± 0.15	0.73 ± 0.17	0.70 ± 0.17	**0.74** ± 0.15	**0.74** ± 0.15	0.71 ± 0.16
0.8	0.68 ± 0.13	**0.73** ± 0.16	0.68 ± 0.15	0.72 ± 0.16	**0.73** ± 0.17	0.69 ± 0.15
0.7	0.66 ± 0.13	0.72 ± 0.15	0.64 ± 0.14	0.72 ± 0.15	**0.73** ± 0.15	0.66 ± 0.14
0.6	0.67 ± 0.12	**0.72** ± 0.16	0.59 ± 0.12	0.69 ± 0.15	**0.72** ± 0.15	0.67 ± 0.14
0.5	0.67 ± 0.14	**0.71** ± 0.16	0.58 ± 0.11	0.69 ± 0.15	**0.71** ± 0.15	0.66 ± 0.14
0.4	0.67 ± 0.15	0.69 ± 0.18	0.56 ± 0.11	0.69 ± 0.16	**0.71** ± 0.16	0.67 ± 0.15
0.3	0.65 ± 0.13	0.67 ± 0.15	0.52 ± 0.08	0.66 ± 0.15	**0.71** ± 0.14	0.68 ± 0.15
0.2	0.67 ± 0.15	0.68 ± 0.16	0.49 ± 0.07	0.67 ± 0.16	**0.72** ± 0.15	0.68 ± 0.13
0.1	0.62 ± 0.16	0.62 ± 0.18	0.48 ± 0.07	0.63 ± 0.17	**0.68** ± 0.17	0.66 ± 0.15

Table B.3
Average F1 score and standard deviation of each attribution method using EUSN across all binary UCR datasets, evaluated from noise-free conditions ($r = 1.0$) to high noise levels ($r = 0.1$, i.e., 10× noise). Higher scores indicate better performance; best results are highlighted in bold.

r	RND	AVG	LAST	GRAD	GRAD*INP	GRADSHAP
1.0	0.69 ± 0.15	0.73 ± 0.17	**0.75** ± 0.18	0.73 ± 0.16	0.73 ± 0.17	0.70 ± 0.13
0.9	0.66 ± 0.14	0.69 ± 0.17	**0.71** ± 0.17	0.66 ± 0.16	0.67 ± 0.15	0.65 ± 0.14
0.8	0.65 ± 0.14	0.68 ± 0.16	**0.70** ± 0.14	0.63 ± 0.16	0.65 ± 0.14	0.64 ± 0.13
0.7	0.59 ± 0.14	0.62 ± 0.16	0.64 ± 0.14	0.62 ± 0.14	**0.65** ± 0.14	0.58 ± 0.12
0.6	0.59 ± 0.12	**0.61** ± 0.13	**0.61** ± 0.14	0.59 ± 0.13	**0.61** ± 0.13	0.57 ± 0.11
0.5	0.58 ± 0.12	0.61 ± 0.13	0.60 ± 0.12	0.59 ± 0.11	**0.62** ± 0.10	0.57 ± 0.10
0.4	0.56 ± 0.11	**0.58** ± 0.12	0.57 ± 0.11	0.56 ± 0.11	0.57 ± 0.10	0.55 ± 0.10
0.3	0.55 ± 0.07	**0.56** ± 0.08	0.54 ± 0.08	0.54 ± 0.09	0.54 ± 0.09	0.53 ± 0.08
0.2	0.52 ± 0.08	**0.53** ± 0.08	0.52 ± 0.09	0.51 ± 0.08	0.51 ± 0.09	0.51 ± 0.07
0.1	0.49 ± 0.05	**0.50** ± 0.05	**0.50** ± 0.05	**0.50** ± 0.04	**0.50** ± 0.06	0.49 ± 0.07

Table B.4
Average F1 score and standard deviation of each attribution method using RES-ESN across all **binary classification** UCR datasets, evaluated under noise-free conditions ($r = 1.0$) to high noise levels ($r = 0.1$, i.e., 10× noise). Higher scores indicate better performance; best results are highlighted in bold.

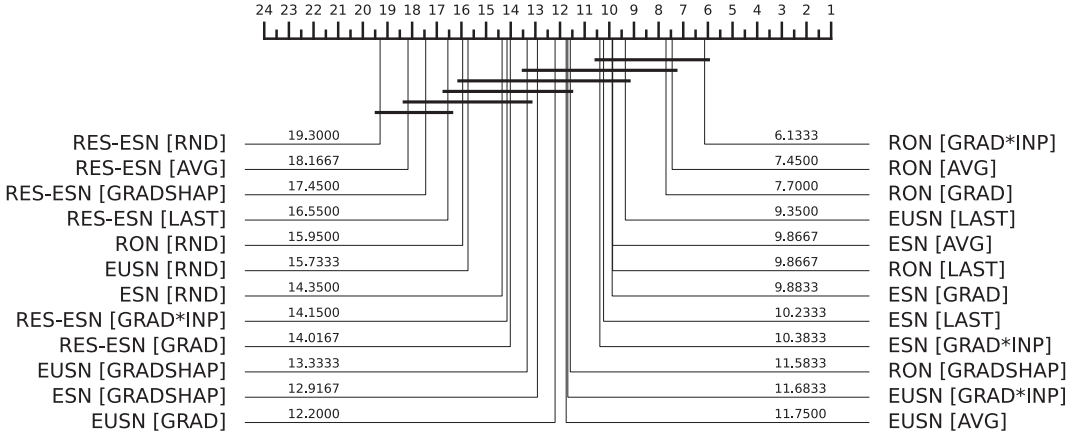
r	RND	AVG	LAST	GRAD	GRAD*INP	GRADSHAP
1.0	0.59 ± 0.11	0.63 ± 0.16	0.64 ± 0.14	**0.71** ± 0.16	0.69 ± 0.19	0.64 ± 0.12
0.9	0.57 ± 0.12	0.61 ± 0.17	0.55 ± 0.11	0.70 ± 0.16	**0.71** ± 0.18	0.63 ± 0.13
0.8	0.55 ± 0.13	0.59 ± 0.16	0.50 ± 0.08	0.67 ± 0.16	**0.71** ± 0.17	0.61 ± 0.11
0.7	0.56 ± 0.10	0.59 ± 0.16	0.50 ± 0.07	0.66 ± 0.16	**0.70** ± 0.16	0.61 ± 0.11
0.6	0.56 ± 0.11	0.60 ± 0.15	0.48 ± 0.06	0.68 ± 0.15	**0.72** ± 0.15	0.61 ± 0.12
0.5	0.56 ± 0.12	0.59 ± 0.16	0.49 ± 0.05	0.65 ± 0.17	**0.70** ± 0.15	0.60 ± 0.13
0.4	0.58 ± 0.13	0.59 ± 0.15	0.49 ± 0.08	0.66 ± 0.15	**0.72** ± 0.15	0.61 ± 0.14
0.3	0.57 ± 0.12	0.56 ± 0.16	0.48 ± 0.05	0.65 ± 0.14	**0.70** ± 0.15	0.62 ± 0.13
0.2	0.55 ± 0.12	0.55 ± 0.15	0.47 ± 0.04	0.61 ± 0.16	**0.68** ± 0.12	0.62 ± 0.13
0.1	0.54 ± 0.14	0.53 ± 0.15	0.48 ± 0.05	0.55 ± 0.17	**0.65** ± 0.14	0.61 ± 0.14

Table B.5
Average F1 score and standard deviation of each attribution method using RON across all **binary classification** UCR datasets, evaluated from noise-free conditions ($r = 1.0$) to high noise levels ($r = 0.1$, i.e., 10× noise). Higher scores indicate better performance; best results are highlighted in bold.

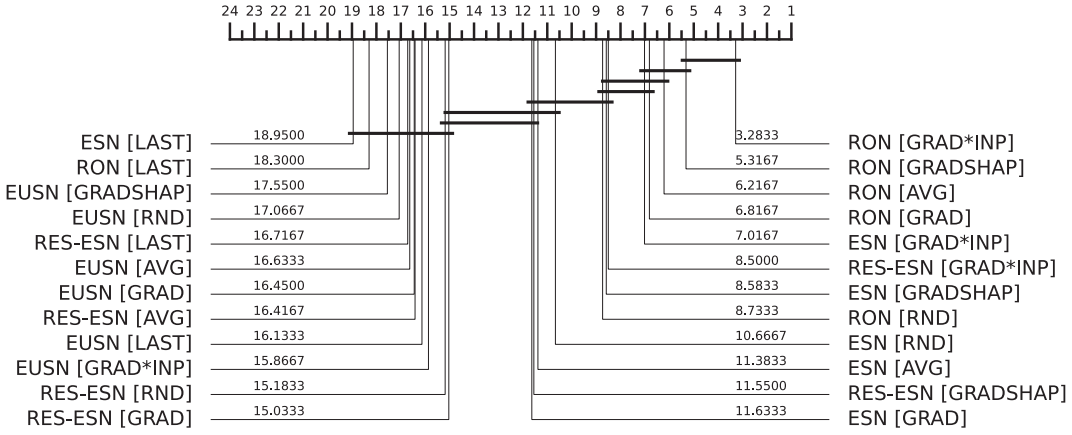
r	RND	AVG	LAST	GRAD	GRAD*INP	GRADSHAP
1.0	0.66 ± 0.12	**0.78** ± 0.16	0.75 ± 0.17	**0.78** ± 0.16	**0.78** ± 0.18	0.73 ± 0.16
0.9	0.64 ± 0.12	**0.77** ± 0.16	0.70 ± 0.17	**0.77** ± 0.15	**0.77** ± 0.17	0.72 ± 0.14
0.8	0.64 ± 0.11	**0.76** ± 0.16	0.66 ± 0.15	0.75 ± 0.14	0.75 ± 0.15	0.69 ± 0.15
0.7	0.62 ± 0.11	0.74 ± 0.14	0.60 ± 0.15	0.74 ± 0.14	**0.75** ± 0.14	0.66 ± 0.13
0.6	0.64 ± 0.11	**0.74** ± 0.15	0.58 ± 0.13	0.72 ± 0.11	0.73 ± 0.13	0.66 ± 0.13
0.5	0.64 ± 0.11	0.75 ± 0.15	0.54 ± 0.09	0.73 ± 0.13	**0.76** ± 0.13	0.67 ± 0.14
0.4	0.64 ± 0.12	0.73 ± 0.16	0.51 ± 0.09	0.72 ± 0.13	**0.75** ± 0.15	0.68 ± 0.15
0.3	0.65 ± 0.11	0.74 ± 0.15	0.51 ± 0.08	0.73 ± 0.14	**0.77** ± 0.15	0.68 ± 0.12
0.2	0.65 ± 0.13	0.73 ± 0.17	0.48 ± 0.07	0.73 ± 0.14	**0.78** ± 0.14	0.69 ± 0.14
0.1	0.65 ± 0.13	0.70 ± 0.17	0.48 ± 0.04	0.69 ± 0.15	**0.75** ± 0.15	0.72 ± 0.14

Table B.6
Average F1 score and standard deviation of GRAD, GRAD*INP, MAG, using ESN and RON across all **binary classification** UCR datasets, evaluated in noise-free conditions ($r = 1.0$), medium ($r = 0.5$) and high noise levels ($r = 0.1$). Higher scores indicate better performance; best results for each network are highlighted in bold.

r	ESN			RON		
	GRAD	GRAD*INP	MAG	GRAD	GRAD*INP	MAG
1.0	**0.77** ± 0.17	0.76 ± 0.17	0.76 ± 0.18	**0.78** ± 0.16	**0.78** ± 0.18	0.77 ± 0.18
0.5	0.69 ± 0.15	**0.71** ± 0.15	**0.71** ± 0.16	0.73 ± 0.13	**0.76** ± 0.13	0.75 ± 0.17
0.1	0.63 ± 0.17	**0.68** ± 0.17	0.62 ± 0.18	0.69 ± 0.15	**0.75** ± 0.15	0.72 ± 0.19

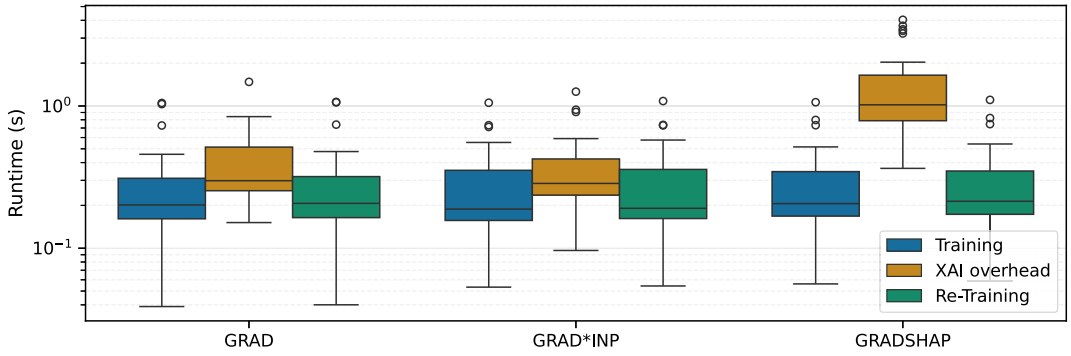


(a) All RNNs, $r = 1.0$ (no noise)

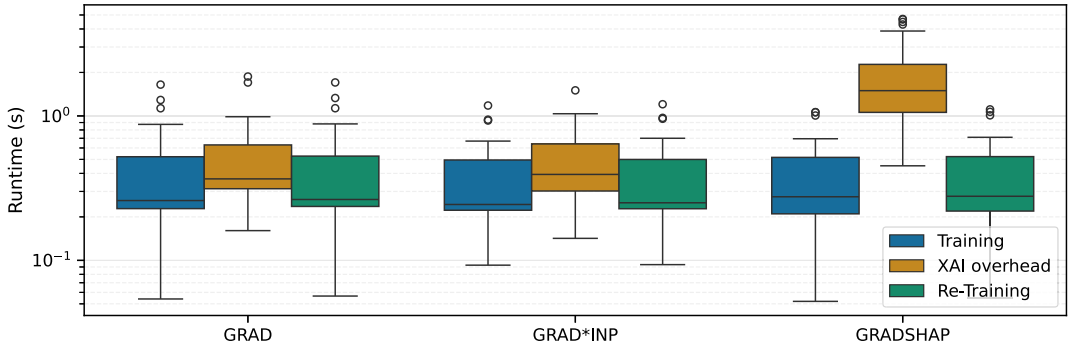


(b) All RNNs, $r = 0.1$ (10x noise)

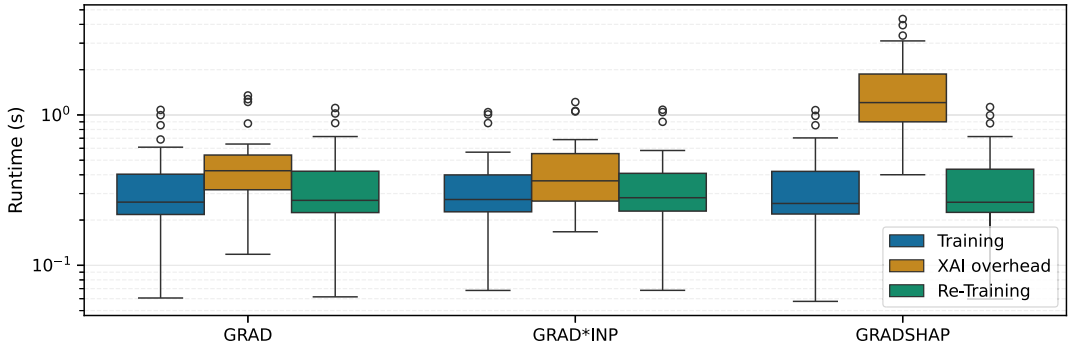
Fig. B.7. Global Critical Difference Plots across the four different randomized RNNs on **binary classification** datasets in the two extreme noise cases, i.e., $r = 1.0$ (no noise), and $r = 0.1$ (10x noise). Best models have lower rank, and are shown on the right of each plot. Models connected with a horizontal line are statistically tied.



(a) EUSN.



(b) RES-ESN.



(c) RON.

Fig. B.8. Boxplots of training, XAI overhead, and retraining runtime (in seconds) by ESN type. Lower is better.

Appendix C. Tables and plots for multiclass classification

See Tables C.7–C.10, Figs. C.9 and C.10

Table C.7
Average F1 score and standard deviation of each attribution method using ESN across all **multiclass classification** UCR datasets, evaluated from noise-free conditions ($r = 1.0$) to high noise levels ($r = 0.1$, i.e., 10× noise). Higher scores indicate better performance; best results are highlighted in bold.

r	RND	AVG	LAST	GRAD	GRAD*INP	GRADSHAP
1.0	0.58 ± 0.26	**0.71** ± 0.19	0.70 ± 0.20	0.66 ± 0.15	0.69 ± 0.17	0.64 ± 0.17
0.9	0.54 ± 0.25	**0.65** ± 0.21	0.55 ± 0.22	0.62 ± 0.19	0.63 ± 0.18	0.57 ± 0.20
0.8	0.51 ± 0.23	0.59 ± 0.21	0.49 ± 0.23	0.58 ± 0.22	**0.61** ± 0.23	0.55 ± 0.19
0.7	0.47 ± 0.22	0.59 ± 0.21	0.41 ± 0.25	0.55 ± 0.23	**0.63** ± 0.23	0.54 ± 0.20
0.6	0.48 ± 0.22	0.56 ± 0.18	0.33 ± 0.20	0.54 ± 0.20	**0.60** ± 0.20	0.52 ± 0.22
0.5	0.50 ± 0.20	0.59 ± 0.23	0.29 ± 0.18	0.54 ± 0.23	**0.62** ± 0.25	0.51 ± 0.20
0.4	0.50 ± 0.19	**0.59** ± 0.21	0.24 ± 0.15	0.49 ± 0.22	0.56 ± 0.22	0.53 ± 0.22
0.3	0.49 ± 0.21	0.57 ± 0.19	0.20 ± 0.11	0.51 ± 0.20	**0.58** ± 0.18	0.52 ± 0.20
0.2	0.51 ± 0.20	0.54 ± 0.19	0.19 ± 0.14	0.49 ± 0.17	**0.59** ± 0.20	0.54 ± 0.20
0.1	0.53 ± 0.19	0.54 ± 0.22	0.18 ± 0.12	0.51 ± 0.21	**0.62** ± 0.23	0.60 ± 0.22

Table C.8
Average F1 score and standard deviation of each attribution method using EUSN across all **multiclass classification** UCR datasets, evaluated from noise-free conditions ($r = 1.0$) to high noise levels ($r = 0.1$, i.e., 10× noise). Higher scores indicate better performance; best results are highlighted in bold.

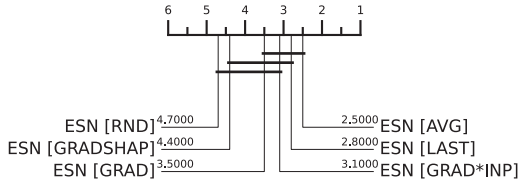
r	RND	AVG	LAST	GRAD	GRAD*INP	GRADSHAP
1.0	0.72 ± 0.27	**0.76** ± 0.22	0.74 ± 0.21	0.69 ± 0.20	0.69 ± 0.21	0.66 ± 0.20
0.9	0.59 ± 0.24	**0.60** ± 0.23	0.58 ± 0.22	0.54 ± 0.22	0.57 ± 0.22	0.51 ± 0.21
0.8	**0.51** ± 0.28	**0.51** ± 0.27	0.49 ± 0.26	0.47 ± 0.24	0.47 ± 0.25	0.45 ± 0.23
0.7	0.44 ± 0.26	**0.45** ± 0.24	0.41 ± 0.22	0.40 ± 0.25	0.42 ± 0.22	0.39 ± 0.21
0.6	0.42 ± 0.23	**0.43** ± 0.23	0.38 ± 0.22	0.39 ± 0.20	0.41 ± 0.20	0.38 ± 0.21
0.5	0.36 ± 0.25	**0.37** ± 0.26	0.35 ± 0.21	0.32 ± 0.23	0.35 ± 0.21	0.33 ± 0.20
0.4	**0.32** ± 0.20	0.31 ± 0.19	0.27 ± 0.16	0.29 ± 0.16	0.28 ± 0.17	0.29 ± 0.18
0.3	0.26 ± 0.18	0.27 ± 0.17	0.22 ± 0.15	0.21 ± 0.14	0.25 ± 0.14	**0.28** ± 0.17
0.2	0.21 ± 0.15	0.22 ± 0.15	0.19 ± 0.13	0.19 ± 0.13	0.21 ± 0.14	**0.23** ± 0.16
0.1	0.19 ± 0.13	0.19 ± 0.13	0.19 ± 0.13	**0.20** ± 0.11	**0.20** ± 0.13	0.17 ± 0.10

Table C.9
Average F1 score and standard deviation of each attribution method using RES-ESN across all **multiclass classification** UCR datasets, evaluated from noise-free conditions ($r = 1.0$) to high noise levels ($r = 0.1$, i.e., 10× noise). Higher scores indicate better performance; best results are highlighted in bold.

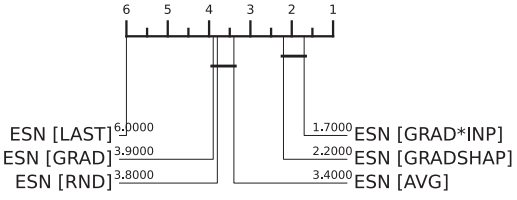
r	RND	AVG	LAST	GRAD	GRAD*INP	GRADSHAP
1.0	0.34 ± 0.16	0.46 ± 0.21	0.51 ± 0.23	0.54 ± 0.24	**0.55** ± 0.25	0.50 ± 0.21
0.9	0.31 ± 0.14	0.40 ± 0.20	0.27 ± 0.17	0.44 ± 0.24	**0.53** ± 0.24	0.40 ± 0.20
0.8	0.33 ± 0.13	0.39 ± 0.19	0.18 ± 0.13	0.47 ± 0.24	**0.53** ± 0.24	0.37 ± 0.20
0.7	0.34 ± 0.15	0.37 ± 0.19	0.22 ± 0.13	0.45 ± 0.20	**0.55** ± 0.22	0.36 ± 0.21
0.6	0.32 ± 0.13	0.34 ± 0.17	0.20 ± 0.13	0.46 ± 0.24	**0.53** ± 0.27	0.37 ± 0.19
0.5	0.36 ± 0.13	0.40 ± 0.20	0.17 ± 0.10	0.44 ± 0.24	**0.61** ± 0.23	0.35 ± 0.20
0.4	0.32 ± 0.13	0.37 ± 0.15	0.19 ± 0.13	0.44 ± 0.24	**0.57** ± 0.26	0.38 ± 0.18
0.3	0.35 ± 0.15	0.29 ± 0.20	0.17 ± 0.12	0.40 ± 0.25	**0.58** ± 0.27	0.40 ± 0.23
0.2	0.32 ± 0.14	0.28 ± 0.16	0.19 ± 0.12	0.42 ± 0.21	**0.53** ± 0.24	0.40 ± 0.24
0.1	0.27 ± 0.15	0.24 ± 0.16	0.20 ± 0.12	0.28 ± 0.24	**0.43** ± 0.20	0.40 ± 0.23

Table C.10
Average F1 score and standard deviation of each attribution method using RON across all **multiclass classification** UCR datasets, evaluated from noise-free conditions ($r = 1.0$) to high noise levels ($r = 0.1$, i.e., 10× noise). Higher scores indicate better performance; best results are highlighted in bold.

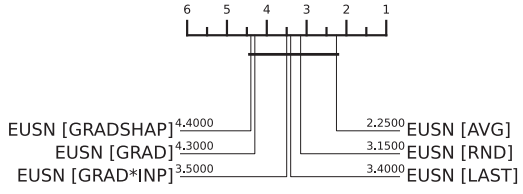
r	RND	AVG	LAST	GRAD	GRAD*INP	GRADSHAP
1.0	0.48 ± 0.24	0.69 ± 0.26	0.68 ± 0.21	0.70 ± 0.18	**0.72** ± 0.21	0.59 ± 0.20
0.9	0.45 ± 0.20	0.64 ± 0.21	0.50 ± 0.25	0.61 ± 0.20	**0.65** ± 0.21	0.53 ± 0.21
0.8	0.43 ± 0.22	0.60 ± 0.23	0.39 ± 0.26	0.58 ± 0.24	**0.62** ± 0.23	0.46 ± 0.18
0.7	0.43 ± 0.21	**0.62** ± 0.21	0.33 ± 0.26	0.59 ± 0.23	0.61 ± 0.24	0.47 ± 0.22
0.6	0.43 ± 0.21	**0.64** ± 0.18	0.29 ± 0.18	0.59 ± 0.17	0.61 ± 0.19	0.49 ± 0.18
0.5	0.44 ± 0.20	**0.63** ± 0.22	0.25 ± 0.16	0.58 ± 0.23	**0.63** ± 0.21	0.46 ± 0.21
0.4	0.44 ± 0.18	0.64 ± 0.21	0.23 ± 0.13	0.62 ± 0.20	**0.68** ± 0.19	0.44 ± 0.21
0.3	0.47 ± 0.20	**0.66** ± 0.21	0.20 ± 0.12	0.64 ± 0.21	**0.66** ± 0.20	0.52 ± 0.22
0.2	0.48 ± 0.21	0.65 ± 0.22	0.16 ± 0.10	0.62 ± 0.21	**0.67** ± 0.20	0.54 ± 0.21
0.1	0.49 ± 0.21	0.64 ± 0.25	0.18 ± 0.12	0.57 ± 0.23	**0.66** ± 0.22	0.59 ± 0.19



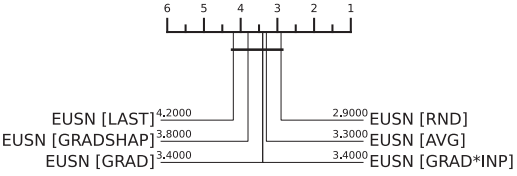
(a) ESN, $r = 1.0$ (no noise)



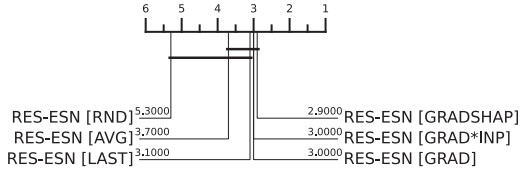
(b) ESN, $r = 0.1$ (10x noise)



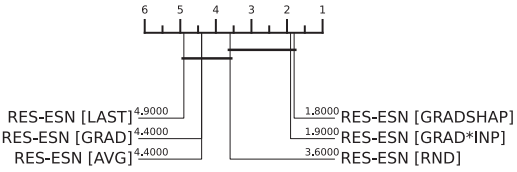
(c) EUSN, $r = 1.0$ (no noise)



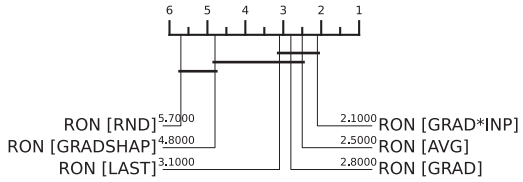
(d) EUSN, $r = 0.1$ (10x noise)



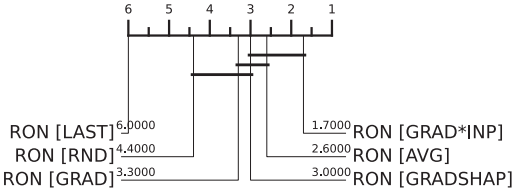
(e) RES-ESN, $r = 1.0$ (no noise)



(f) RES-ESN, $r = 0.1$ (10x noise)

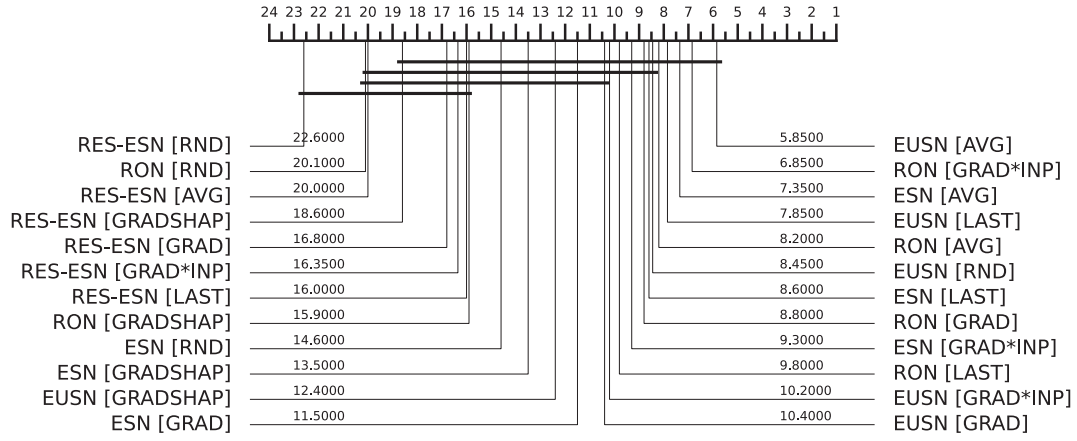


(g) RON, $r = 1.0$ (no noise)

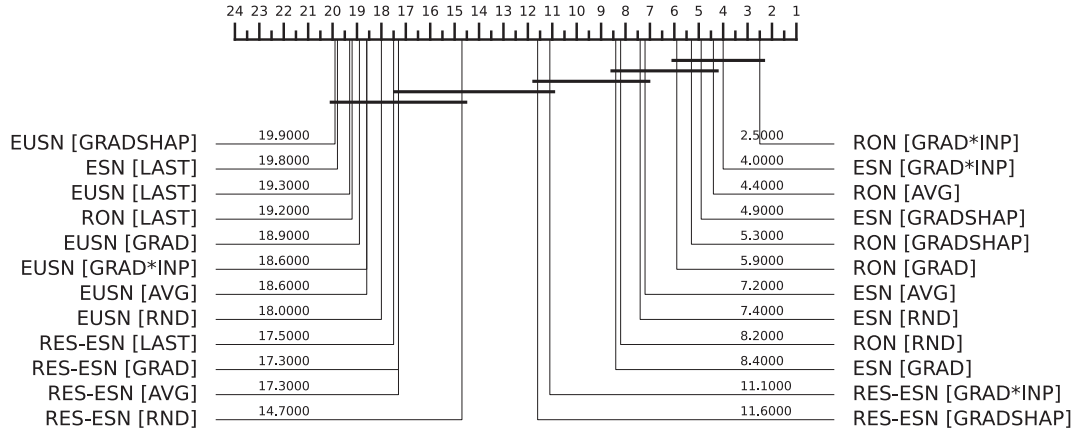


(h) RON, $r = 0.1$ (10x noise)

Fig. C.9. Critical Difference Plots for the four different randomized RNNs on **multiclass classification** datasets in the two extreme noise cases, i.e., $r = 1.0$ (no noise), and $r = 0.1$ (10x noise). Best models have lower rank, and are shown on the right of each plot. Models connected with a horizontal line are statistically tied.



(a) All RNNs, $r = 1.0$ (no noise)



(b) All RNNs, $r = 0.1$ (10x noise)

Fig. C.10. Global Critical Difference Plots across the four different randomized RNNs on **multiclass classification** datasets in the two extreme noise cases, i.e., $r = 1.0$ (no noise), and $r = 0.1$ (10x noise). Best models have lower rank, and are shown on the right of each plot. Models connected with a horizontal line are statistically tied.

Data availability

Data and code are publicly available at https://github.com/fspinna/xai_enhanced_esn.

References

[1] A. Apicella, et al., Strategies to exploit XAI to improve classification systems, in: xAI (1), Springer, 2023, pp. 147–159.
[2] A.B. Arrieta, et al., Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI, Inf. Fusion 58 (2020) 82–115.
[3] F.M. Bianchi, S. Scardapane, S. Løkse, R. Jenssen, Reservoir computing approaches for representation and classification of multivariate time series, IEEE Trans. Neural Netw. Learn. Syst. 32 (2020) 2169–2179.
[4] A. Ceni, A. Cossu, M. Stölzle, J. Liu, C. Della Santina, D. Bacciu, C. Gallicchio, Random oscillators network for time series processing, in: AISTATS 2024, 2024.
[5] A. Ceni, C. Gallicchio, Edge of stability echo state network, IEEE Trans. Neural Netw. Learn. Syst. (2024).
[6] A. Ceni, C. Gallicchio, Residual echo state networks: residual recurrent neural networks with stable dynamics and fast learning, Neurocomputing 597 (2024) 127966.
[7] A. Cossu, A. Ceni, D. Bacciu, C. Gallicchio, Sparse reservoir topologies for physical implementations of random oscillators networks, in: Proceedings of the 4th International Conference on AI-ML Systems, 2024, pp. 1–9.
[8] A. Cossu, F. Spinnato, R. Guidotti, D. Bacciu, Drifting explanations in continual learning, Neurocomputing 597 (2024) 127960.
[9] H.A. Dau, A. Bagnall, K. Kamgar, C.C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, E. Keogh, The UCR time series archive, IEEE/CAA J. Autom. Sin. 6 (2019) 1293–1305.
[10] J.L. Elman, Finding structure in time, Cogn. Sci. 14 (1990) 179–211, https://doi.org/10.1207/s15516709cog1402_1
[11] W. Freeborough, T. van Zyl, Investigating explainability methods in recurrent neural network architectures for financial time series data, Appl. Sci. 12 (2022) 1427.
[12] C. Gallicchio, Euler state networks: non-dissipative reservoir computing, Neurocomputing 579 (2024) 127411.
[13] R. Guidotti, et al., A survey of methods for explaining black box models, ACM Comput. Surv. 51 (2019) :93:1–:93:42.
[14] A.A. Ismail, H. Corrada Bravo, S. Feizi, Improving deep learning interpretability by saliency guided training, Adv. Neural Inf. Process. Syst. 34 (2021) 26726–26739.
[15] H. Jaeger, Short Term Memory in Echo State Networks, 2001.
[16] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note, Bonn, Germany: German National Research Center for Information Technology GMD Technical Report 148 (2001) 13.
[17] H. Jaeger, Adaptive nonlinear system identification with echo state networks, Adv. Neural Inf. Process. Syst. 15 (2002).
[18] Y. Liu, Z. Shao, N. Hoffmann, Global attention mechanism: Retain information to enhance channel-spatial interactions, arXiv preprint arXiv:2112.05561, (2021).
[19] M. Lukoševičius, A practical guide to applying echo state networks, in: Neural Networks: Tricks of the Trade, second ed., Springer, 2012, pp. 659–686.
[20] M. Lukosevicius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comput. Sci. Rev. 3 (2009) 127–149.
[21] S.M. Lundberg, S. Lee, A Unified Approach to Interpreting Model Predictions, 2017, 4765–4774.
[22] L.R. Medsker, L. Jain, et al., Recurrent neural networks, Des. Appl. 5 (2001) 2.
[23] M. Middlehurst, P. Schäfer, A. Bagnall, Bake off redux: a review and experimental evaluation of recent time series classification algorithms, Data Min. Knowl. Discov. 38 (2024) 1958–2031.
[24] T. Miller, Explanation in artificial intelligence: insights from the social sciences, Artif. Intell. 267 (2019) 1–38.
[25] A.P. Ruiz, M. Flynn, J. Large, M. Middlehurst, A. Bagnall, The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 35 (2021) 401–449.

[26] W. Samek, G. Montavon, S. Lapuschkin, C.J. Anders, K.R. Müller, Explaining deep neural networks and beyond: a review of methods and applications, Proc. IEEE 109 (2021) 247–278.
[27] A. de Santana Correia, E.L. Colombini, Attention, please! a survey of neural attention models in deep learning, Artif. Intell. Rev. 55 (2022) 6037–6124.
[28] A. Shrikumar, et al., Not just a black box: learning important features through propagating activation differences, CoRR abs/1605.01713 (2016).
[29] I. Šimić, V. Sabol, E. Veas, Xai methods for neural time series classification: A brief review, arXiv preprint arXiv:2108.08009, (2021).
[30] F. Spinnato, A. Cossu, R. Guidotti, A. Ceni, C. Gallicchio, D. Bacciu, Enhancing echo state networks with gradient-based explainability methods, in: 32nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2024, Bruges, Belgium, October 9-11, 2024, 2024, https://doi.org/10.14428/ESANN/2024.ES2024-78
[31] F. Spinnato, R. Guidotti, A. Monreale, M. Nanni, Fast, interpretable and deterministic time series classification with a bag-of-receptive-fields, IEEE Access (2024).
[32] F. Spinnato, R. Guidotti, M. Nanni, D. Maccagnola, G. Paciello, A.B. Farina, Explaining crash predictions on multivariate time series data, in: International Conference on Discovery Science, Springer, 2022, pp. 556–566.
[33] F. Spinnato, C. Landi, Pyrregular: A unified framework for irregular time series, with classification benchmarks, arXiv preprint arXiv:2505.06047, (2025).
[34] M. Sundararajan, et al., Axiomatic attribution for deep networks, in: ICML, PMLR, 2017, pp. 3319–3328.
[35] A. Theissler, et al., Explainable AI for time series classification: a review, taxonomy and research directions, IEEE Access 10 (2022) 100700–100724.
[36] P.J. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (1990) 1550–1560.