Reservoir computing approaches for representation and classification of multivariate time series

Filippo Maria Bianchi*, Simone Scardapane, Sigurd Løkse, and Robert Jenssen

***Abstract*—Classification of multivariate time series (MTS) has been tackled with a large variety of methodologies and applied to a wide range of scenarios. Reservoir Computing (RC) provides efficient tools to generate a vectorial, fixed-size representation of the MTS that can be further processed by standard classifiers. Despite their unrivaled training speed, MTS classifiers based on a standard RC architecture fail to achieve the same accuracy of fully trainable neural networks. In this paper we introduce the *reservoir model space*, an unsupervised approach based on RC to learn vectorial representations of MTS. Each MTS is encoded within the parameters of a linear model trained to predict a low-dimensional embedding of the reservoir dynamics. Compared to other RC methods, our model space yields better representations and attains comparable computational performance, thanks to an intermediate dimensionality reduction procedure. As a second contribution we propose a modular RC framework for MTS classification, with an associated open-source Python library. The framework provides different modules to seamlessly implement advanced RC architectures. The architectures are compared to other MTS classifiers, including deep learning models and time series kernels. Results obtained on benchmark and real-world MTS datasets show that RC classifiers are dramatically faster and, when implemented using our proposed representation, also achieve superior classification accuracy.**

***Index Terms*—Reservoir computing, model space, time series classification, recurrent neural networks**

I. Introduction

The problem of classifying multivariate time series (MTS) consists in assigning each MTS to one of a fixed number of classes. This is a fundamental task in many applications, including (but not limited to) health monitoring [1], civil engineering [2], action recognition [3], and speech analysis [4]. The problem has been tackled by approaches spanning from the definition of tailored distance measures over MTS to the identification of patterns in the form of dictionaries or shapelets [5], [6], [7], [8]. In this paper we focus on classifiers based on recurrent neural networks (RNNs), which first process sequentially the MTS with a dynamic model, and then exploit the sequence of the model states generated over time to perform classification [9].

Reservoir computing (RC) is a family of RNN models whose recurrent part is generated randomly and then kept fixed [10], [11]. Despite this strong simplification, the recurrent part of the model (the reservoir) provides a rich pool of dynamic features which are suitable for solving a large variety of tasks. Indeed, RC models achieved excellent performance in time series forecasting [12], [13], and process modelling [14]. In machine learning, RC techniques were originally introduced under the name echo state networks (ESNs) [15]; in this paper, we use the two terms interchangeably.

RC-based classifiers represent the MTS either as the last or the mean of all the reservoir states and then process it with a classification algorithm for vectorial data [16], [17]. Despite their unrivaled training speed, these approaches fail to achieve the same accuracy of competing state-of-the-art classifiers [18]. To learn more powerful representations, an alternative approach originally proposed in [19] and later applied to MTS classification and fault detection [20], [18], advocates to map the inputs in a "model-based" feature space where the MTS is represented by the parameters of a model, trained to predict the next input from the current reservoir state. As a drawback, this approach accounts only for those reservoir dynamics useful to predict the next input and could neglect important information that characterize the MTS. To overcome this limitation, we propose a new model-space criterion that disentangles from the constraints imposed by this formulation.

Contributions of the paper: We propose an unsupervised procedure to generate MTS representations, called *reservoir model space*, that consists in the parameters of the one-step-ahead predictor that estimates the future *reservoir* state, as opposed to the future MTS input. As shown in our previous work [21], the reservoir states carry all the information necessary to reconstruct the phase space that, in turn, gives a complete knowledge of the underlying dynamical system generating the observed MTS. Therefore, a model capable of predicting the next reservoir state accounts for all the system dynamics and provides a much more accurate characterization of the MTS. Due to the large size of the reservoir, a naïve formulation of the model space yields extremely large representations that lead to overfit in the subsequent classifier and hamper the computational efficiency proper of the RC paradigm. We address this issue by training the prediction model on a low-dimensional embedding of the original dynamics. The embedding is obtained by applying to the reservoir states sequence a modified version of principal component analysis (PCA) for tensors, which keeps separated the modes of variation among time steps and data samples. The proposed representation is novel and, while our focus is on MTS, it naturally extends also to univariate time series.

As a second contribution, we introduce a unified RC framework (with an associated open source Python library) for MTS classification that generalizes both classic and advanced RC architectures. Our framework consists of four independent

*filippombianchi@gmail.com
F. M. Bianchi is with NORCE, The Norwegian Research Centre
S. Scardapane is with Sapienza University of Rome
S. Løkse and R. Jenssen are with UiT, the Arctic University of Tromsø.

modules that specify i) the architecture of the reservoir, ii) a dimensionality reduction procedure applied to reservoir activations, iii) the representation used to describe the input MTS, and iv) the readout that performs the final classification.

In the experiments, we compare several RC architectures implemented with our framework, with state-of-the-art time series classifiers, classifiers based on fully trainable RNNs, deep learning models, DTW, and SVM configured with kernels for MTS. The results obtained on several real-world dataset show that the RC classifiers are dramatically faster than the other methods and, when implemented using our proposed representation, also achieve a competitive classification accuracy.

Notation: we denote variables as lowercase letters (x); constants as uppercase letters (X); vectors as boldface lowercase letters ($\mathbf{x}$); matrices as boldface uppercase letters ($\mathbf{X}$); tensors as calligraphic letters ($\mathcal{X}$). All vectors are assumed to be columns. The operator $\|\cdot\|_p$ is the standard ℓ_p norm in Euclidean spaces. The notation $x(t)$ indicates time step t and $x[n]$ sample n in the dataset.

II. Preliminaries

We consider classification of generic F-dimensional MTS with T time instants, whose observation at time t is denoted as $\mathbf{x}(t) \in \mathbb{R}^F$. We represent a MTS in a compact form as a $T \times F$ matrix $\mathbf{X} = [\mathbf{x}(1), \dots, \mathbf{x}(T)]^T$ [1].

Common in machine learning is to express the classifier as a combination of an *encoding* and a *decoding* function. The encoder generates a representation of the input, while the decoder is a discriminative (or predictive) model that computes the posterior probability of the output given the encoder representation. An encoder based on an RNN [22] is particularly suitable to model sequential data, and is governed by the state-update equation

$$\mathbf{h}(t) = f\left(\mathbf{x}(t), \mathbf{h}(t-1); \theta_{\text{enc}}\right), \tag{1}$$

where $\mathbf{h}(t)$ is the RNN state at time t that depends on its previous value $\mathbf{h}(t-1)$ and the current input $\mathbf{x}(t)$, $f(\cdot)$ is a nonlinear activation function (e.g., a sigmoid or hyperbolic tangent), and θ_{enc} are adaptable parameters. The simplest (vanilla) formulation reads:

$$\mathbf{h}(t) = tanh\left(\mathbf{W}_{\text{in}}\mathbf{x}(t) + \mathbf{W}_{\text{r}}\mathbf{h}(t-1)\right), \tag{2}$$

with $\theta_{\text{enc}} = \{\mathbf{W}_{\text{in}}, \mathbf{W}_{\text{r}}\}$. The matrices $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{r}}$ are the weights of the input and recurrent connections, respectively.

From the sequence of the RNN states generated over time, $\mathbf{H} = [\mathbf{h}(1), \dots, \mathbf{h}(T)]^T$, it is possible to extract a representation $\mathbf{r}_{\mathbf{X}} = r(\mathbf{H})$ of the input $\mathbf{X}$. A common choice is to take $\mathbf{r}_{\mathbf{X}} = \mathbf{h}(T)$, since the RNN can embed into its last state all the information required to reconstruct the original input [23]. The decoder maps the MTS representation $\mathbf{r}_{\mathbf{X}}$ into the output space, which are the class labels $\mathbf{y}$ for a classification task:

$$\mathbf{y} = g(\mathbf{r}_{\mathbf{X}}; \theta_{\text{dec}}), \tag{3}$$

where $g(\cdot)$ can be a (feed-forward) neural network or a linear model, and θ_{dec} are the trainable parameters.

[1] Since MTS may have different lengths, T is a function of the MTS.

In the following, we describe two RNN-based approaches for MTS classification. The first is based on fully trainable architectures, the second on RC where the RNN encoder is left untrained.

A. Fully trainable RNNs and gated architectures

In fully trainable RNNs, given a set of MTS $\{\mathbf{X}[n]\}_{n=1}^N$ and associated labels $\{\mathbf{y}[n]\}_{n=1}^N$, the encoder parameters θ_{enc} and the decoder parameters θ_{dec} are jointly learned by minimizing an empirical cost:

$$\theta^*_{\text{enc}}, \theta^*_{\text{dec}} = \underset{\theta_{\text{enc}}, \theta_{\text{dec}}}{\arg\min} \frac{1}{N} \sum_{n=1}^{N} l\Big(\mathbf{y}[n], g\Big(r\big(f(\mathbf{X}[n])\big)\Big)\Big), \tag{4}$$

where $l(\cdot,\cdot)$ is a generic loss function (e.g., cross-entropy over the labels). The gradient of (4) with respect to θ_{enc} and θ_{dec} can be computed by back-propagation through time [9].

The parameters in the encoding and decoding functions are commonly regularized with a ℓ_2 norm penalty, controlled by a scalar λ. It is also possible to include a dropout regularization, that randomly drops connections during training with probability p_{drop} [24]. In our experiments, we apply a dropout specific for recurrent architectures [25].

Despite the theoretical capability of basic RNNs to model any dynamical system, in practice their effectiveness is hampered by the difficulty of training their parameters [26]. To ensure stability, the derivative of the recurrent function in an RNN must not exceed unity. However, as an undesired effect, the gradient of the loss shrinks when back-propagated in time through the network. Using RC models (described in the next section) is one way of avoiding this problem. Another solution is using the long short-term memory (LSTM) network [27], which exploits gating mechanisms to maintain its internal memory unaltered for long time intervals. However, LSTM flexibility comes at the cost of a higher computational and architectural complexity. A popular variant is the gated recurrent unit (GRU) [28], that provides a better memory conservation by using less parameters than LSTM.

B. Reservoir computing and output model space

To avoid the costly operation of back-propagating through time, the RC approach takes a radical different direction: it still implements the encoding function in (2), but the encoder parameters $\theta_{\text{enc}} = \{\mathbf{W}_{\text{in}}, \mathbf{W}_{\text{r}}\}$ are randomly generated and left untrained. To compensate for this lack of adaptability, a large recurrent layer, the *reservoir*, generates a rich pool of heterogeneous dynamics useful to solve many different tasks. The generalization capabilities of the reservoir mainly depend on three ingredients: (i) a high number of processing units in the recurrent layer, (ii) sparsity of the recurrent connections, and (iii) a spectral radius of the connection weights matrix $\mathbf{W}_{\text{r}}$, set to bring the system to the edge of stability [29]. The behaviour of the reservoir is controlled by modifying the following hyperparameters: the spectral radius ρ; the percentage of non-zero connections β; and the number of hidden units R. Another important hyperparameter is the scaling ω of the values in $\mathbf{W}_{\text{in}}$, which controls the amount of

nonlinearity in the processing units and, jointly with ρ, can shift the internal dynamics from a chaotic to a contractive regime [30]. A Gaussian noise with standard deviation ξ can also be added in the state update function (2) for regularization purposes [15].

In ESNs, the decoder (commonly referred as readout) is usually a linear model:

$$\mathbf{y} = g(\mathbf{r}_\mathbf{X}) = \mathbf{V}_o \mathbf{r}_\mathbf{X} + \mathbf{v}_o \tag{5}$$

The decoder parameters $\theta_{\text{dec}} = \{\mathbf{V}_o, \mathbf{v}_o\}$ can be learned by minimizing a ridge regression loss function

$$\theta^*_{\text{dec}} = \underset{\{\mathbf{V}_o, \mathbf{v}_o\}}{\arg\min} \frac{1}{2} \left\| \mathbf{r}_\mathbf{X} \mathbf{V}_o + \mathbf{v}_o - \mathbf{y} \right\|^2 + \lambda \left\| \mathbf{V}_o \right\|^2, \tag{6}$$

which admits a closed-form solution [11]. The combination of an untrained reservoir and a linear readout defines the basic ESN model [15].

A powerful representation $\mathbf{r}_\mathbf{X}$ is the *output model space* [19], [31], [32], obtained by first processing each MTS with the same reservoir and then training a ridge regression model to predict the input one step-ahead:

$$\mathbf{x}(t+1) = \mathbf{U}_o \mathbf{h}(t) + \mathbf{u}_o. \tag{7}$$

The parameters $\theta_o = [\text{vec}(\mathbf{U}_o); \mathbf{u}_o] \in \mathbb{R}^{F(R+1)}$ becomes the representation $\mathbf{r}_\mathbf{X}$ of the MTS, which is, in turn, processed by the classifier in (5). In the following, we propose a new model space that yields a more expressive representation of the input.

III. Proposed reservoir model space representation

In this section we introduce the main contribution of this paper, the *reservoir model space* for representing a (multivariate) time series, and a dimensionality reduction method that extends PCA to multidimensional temporal data. Related to our idea, but framed in a setting different from RC, are the recent deep learning architectures that learn unsupervised representations by predicting the future in a small-dimensional latent space with autoregressive models [33].

A. Formulation of the reservoir model space

The generalization capability of the reservoir is grounded on the large amount of heterogeneous dynamics it generates from the input. To predict the next input values, different dynamics are selected depending on the forecast horizon of interest. Therefore, when fixing the prediction step (e.g., 1 step-ahead) all those dynamics that are not useful to solve the task are discarded. This introduces a bias in the output model space, since the features that are not important for the prediction task can still be useful to characterize the MTS. Therefore, we propose a new model space, where each MTS is represented by the parameters of a linear model, which predicts the next reservoir state by accounting for all the reservoir dynamics. The linear model trained to predict the next reservoir state reads

$$\mathbf{h}(t+1) = \mathbf{U}_h \mathbf{h}(t) + \mathbf{u}_h, \tag{8}$$

and $\mathbf{r}_\mathbf{X} = \theta_h = [\text{vec}(\mathbf{U}_h); \mathbf{u}_h] \in \mathbb{R}^{R(R+1)}$ is our proposed representation.

The reservoir model space representation characterizes a generative model of the reservoir sequence

$$p\left(\mathbf{h}(T), \mathbf{h}(T-1), \ldots, \mathbf{h}(1); \mathbf{r}_\mathbf{X}\right). \tag{9}$$

The model (9) provides a characterization of both the input and the generative process of its high-level dynamical features, and also induces a metric relationship between samples [34]. A classifier that processes the reservoir model representation combines the explanatory capability of generative models with the classification power of the discriminative methods.

B. Dimensionality reduction for reservoir states tensor

Due to the high dimensionality of the reservoir, the number of parameters of the prediction model in (8) would grow too large, making the proposed representation intractable. Drawbacks in using large representations include overfitting and the high amount of computational resources to evaluate the ridge regression solution for each MTS. In the context of RC, applying PCA to reduce dimensionality of the last reservoir state has shown to improve performance achieved on the inference task [35]. Compared to non-linear methods for dimensionality reduction such as kernel-PCA or autoencoders [36], PCA provides competitive generalization capabilities when combined with RC models and can be computed quickly, thanks to its linear formulation [21].

Our proposed MTS representation do not coincides with the last reservoir state, but depends on the whole sequence of states generated over time. Therefore, we conveniently describe our dataset as a 3-mode tensor $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{N \times T \times R}$ and require a transformation to map $R \rightarrow D$ s.t. $D \ll R$, while maintaining the other dimensions unaltered. Dimensionality reduction on high-order tensors can be achieved through Tucker decomposition, which decomposes a tensor into a core tensor (the lower-dimensional representation) multiplied by a matrix along each mode. When only one dimension of $\boldsymbol{\mathcal{H}}$ is modified, Tucker decomposition becomes equivalent to applying a two-dimensional PCA on a specific matricization of $\boldsymbol{\mathcal{H}}$ [37]. Specifically, to reduce the third dimension (R) one computes the mode-3 matricization of $\boldsymbol{\mathcal{H}}$ by arranging the mode-3 fibers (high-order analogue of matrix rows/columns) to be the rows of a resulting matrix $\mathbf{H}_{(3)} \in \mathbb{R}^{NT \times R}$. Then, standard PCA projects the rows of $\mathbf{H}_{(3)}$ on the eigenvectors associated to the D largest eigenvalues of the covariance matrix $\mathbf{C} \in \mathbb{R}^{R \times R}$, defined as

$$\mathbf{C} = \frac{1}{NT-1} \sum_{i=1}^{NT} \left(\mathbf{h}_i - \bar{\mathbf{h}}\right) \left(\mathbf{h}_i - \bar{\mathbf{h}}\right)^T. \tag{10}$$

In (10), $\mathbf{h}_i$ is the i-th row of $\mathbf{H}_{(3)}$ and $\bar{\mathbf{h}} = \frac{1}{N} \sum_i^{NT} \mathbf{h}_i$. As a result of the concatenation of the first two dimensions in $\boldsymbol{\mathcal{H}}$, $\mathbf{C}$ evaluates the variation of the components in the reservoir states across all samples and time steps at the same time. Consequently, both the original structure of the dataset and the temporal orderings are lost, as the reservoir states relative to different samples and generated in different time steps are mixed together. This may lead to a potential loss

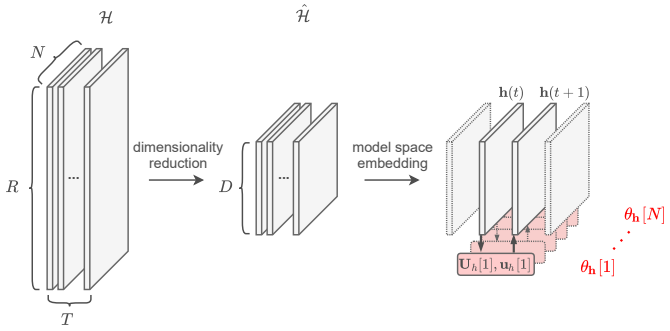

Fig. 1: Schematic depiction of the procedure to generate the reservoir model space representation. For each input MTS $\mathbf{X}[n]$ a sequence of states $\mathbf{H}[n]$ is generated by a fixed reservoir. Those are the frontal slices (dimension N) of $\boldsymbol{\mathcal{H}}$, but notice that in the figure lateral slices (dimension T) are shown. The proposed dimensionality reduction reduces the reservoir features from R to D. An independent model is trained to predict $\hat{\mathbf{H}}[n]$, the n-th frontal slice of $\hat{\boldsymbol{\mathcal{H}}}$, and its parameters $\theta_h[n]$ become the representation of $\mathbf{X}[n]$.

in the representation capability, as the existence of modes of variation in time courses within individual samples is ignored.

To address this issue, we consider as individual samples the matrices $\mathbf{H}_n \in \mathbb{R}^{T\times R}$, obtained by slicing $\boldsymbol{\mathcal{H}}$ across its first dimension (N). Our proposed sample covariance matrix reads

$$\mathbf{S} = \frac{1}{N-1}\sum_{n=1}^{N}\left(\mathbf{H}_n - \bar{\mathbf{H}}\right)^T\left(\mathbf{H}_n - \bar{\mathbf{H}}\right). \tag{11}$$

The first D leading eigenvectors of $\mathbf{S}$ are stacked in a matrix $\mathbf{E} \in \mathbb{R}^{R\times D}$ and the desired tensor of reduced dimensionality is obtained as $\hat{\boldsymbol{\mathcal{H}}} = \boldsymbol{\mathcal{H}} \times_3 \mathbf{E}$, where, $\times_3$ denotes the 3-mode product. Like $\mathbf{C}$, $\mathbf{S} \in \mathbb{R}^{R\times R}$ describes the variations of the variables in the reservoir. However, since the whole sequence of reservoir states is treated as a single observation, the temporal ordering in different MTS is preserved.

After dimensionality reduction, the model in (7) becomes

$$\hat{\mathbf{h}}(t+1) = \mathbf{U}_h\hat{\mathbf{h}}(t) + \mathbf{u}_h, \tag{12}$$

where $\hat{\mathbf{h}}(\cdot)$ are the columns of a frontal slice $\hat{\mathbf{H}}$ of $\hat{\boldsymbol{\mathcal{H}}}$, $\mathbf{U}_h \in \mathbb{R}^{D\times D}$, and $\mathbf{u}_h \in \mathbb{R}^D$. The representation will now coincide with the parameters vector $\mathbf{r}_\mathbf{X} = \theta_h = [\mathrm{vec}(\mathbf{U}_h); \mathbf{u}_h] \in \mathbb{R}^{D(D+1)}$, as shown in Fig. 1.

The complexity for computing the reservoir model space representations for all the MTS in the dataset is given by the sum of $\mathcal{O}(NTVH)$, the cost for computing all the reservoir states, and $\mathcal{O}(H^2NT + H^3)$, the cost of the dimensionality reduction procedure.

IV. A Unified Reservoir Computing Framework for Time Series Classification

In the last years, several works independently extended the basic ESN architecture by designing more sophisticated reservoirs, readouts or representations of the input. To evaluate their synergy and efficacy in the context of MTS classification, we introduce a unified framework that generalizes several RC architectures by combining four modules: i) a reservoir module, ii) a dimensionality reduction module, iii) a representation module, and iv) a readout module. Fig. 2 gives an overview of the models that can be implemented in the framework (including the proposed reservoir model space), by selecting one option in each module. The input MTS $\mathbf{X}$ is processed by a reservoir, which is either unidirectional or bidirectional, and it generates over time the states sequence $\mathbf{H}$. An optional dimensionality reduction step reduces the number of reservoir features and yields a new sequence $\bar{\mathbf{H}}$. Three different approaches can be chosen to generate the input representation $\mathbf{r}_\mathbf{X}$ from the sequence of reservoir states: the last element in the sequence $\mathbf{h}(T)$, the output state model θ_o (Sec. II-B), or the proposed reservoir state model θ_h. The representation $\mathbf{r}_\mathbf{X}$ is finally processed by a decoder (readout) that predicts the class $\mathbf{y}$.

In the following, we describe the reservoir, dimensionality reduction and readout modules, and we discuss the functionality of the variants implemented in our framework. A Python software library implementing the unified framework is publicly available online.[2]

A. Reservoir module

Several approaches have been proposed to extend the ESN reservoir with additional features, such as the capability of handling multiple time scales [38], or to simplify its large and randomized structure [39]. Of particular interest for the classification of MTS is the bidirectional reservoir, which can replace the standard reservoir in our framework. RNNs with bidirectional architectures can extract from the input sequence features that account for dependencies very far in time [40]. In RC, a bidirectional reservoir has been used in the context of time series prediction to incorporate future information, only provided during training, to improve the accuracy of the model [14]. In a classification setting the whole time series is given at once and, thus, a bidirectional reservoir can be exploited in both training and test to generate better MTS representations [35].

Bidirectionality is implemented by feeding into the *same* reservoir an input sequence both in straight and reverse order

$$\begin{aligned}\vec{\mathbf{h}}(t) &= f\left(\mathbf{W}_{\text{in}}\mathbf{x}(t) + \mathbf{W}_{\text{r}}\vec{\mathbf{h}}(t-1)\right),\\ \overleftarrow{\mathbf{h}}(t) &= f\left(\mathbf{W}_{\text{in}}\overleftarrow{\mathbf{x}}(t) + \mathbf{W}_{\text{r}}\overleftarrow{\mathbf{h}}(t-1)\right),\end{aligned} \tag{13}$$

where $\overleftarrow{\mathbf{x}}(t) = \mathbf{x}(T-t)$. The full state is obtained by concatenating the two state vectors in (13), and can capture longer time dependencies by summarizing at every step both recent and past information.

When using a bidirectional reservoir, the linear model in (8) defining the reservoir model space changes into

$$\left[\mathbf{h}(t+1); \overleftarrow{\mathbf{h}}(t+1)\right] = \mathbf{U}_h^b\left[\vec{\mathbf{h}}(t); \overleftarrow{\mathbf{h}}(t)\right] + \mathbf{u}_h^b, \tag{14}$$

where $\mathbf{U}_h^b \in \mathbb{R}^{2R\times 2R}$ and $\mathbf{u}_h^b \in \mathbb{R}^{2R}$ are the new set of parameters. In this case, the linear model is trained to optimize

[2] https://github.com/FilippoMB/Reservoir-Computing-framework-for-multivariate-time-series-classification

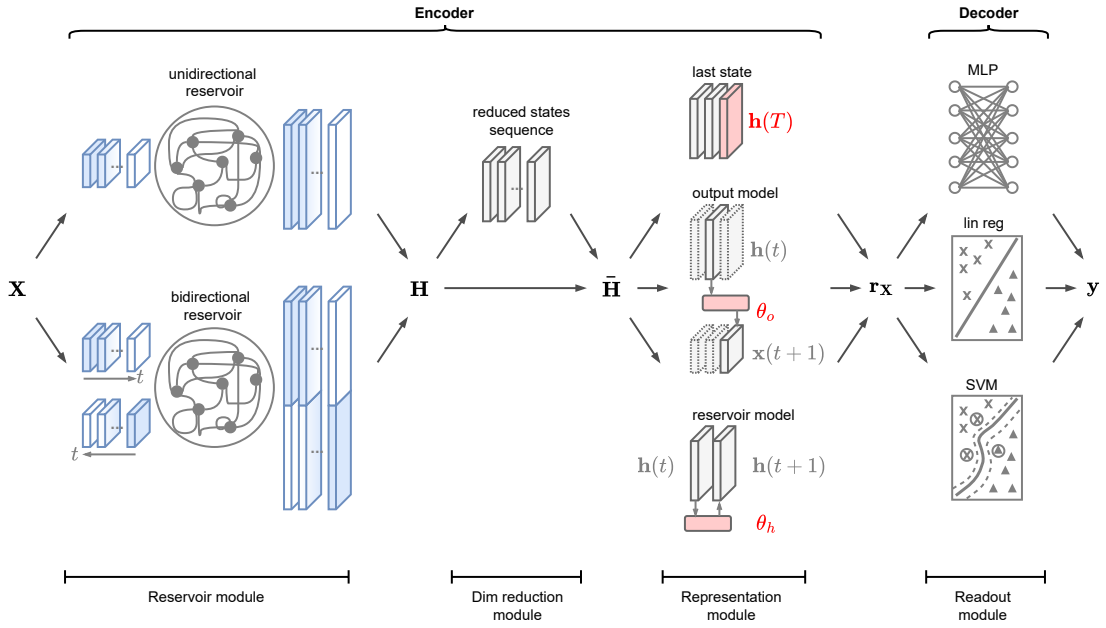

Fig. 2: Framework overview. The encoder generates a representation $\mathbf{r}_\mathbf{X}$ of the MTS $\mathbf{X}$, while the decoder predicts the label $\mathbf{y}$. Several models are obtained by selecting one variant for each module. Arrows indicate mutually exclusive choices

.

two distinct objectives: predicting the next state $\mathbf{h}(t+1)$ and reproducing the previous one $\overleftarrow{\mathbf{h}}(t+1)$ (or equivalently their low-dimensional embeddings). We argue that such a model provides a more accurate representation of the input, by modeling temporal dependencies in both time directions to jointly solve a prediction and a memorization task.

B. Dimensionality reduction module

The dimensionality reduction module projects the sequence of reservoir activations on a lower dimensional subspace, using unsupervised criteria. In the context of RC, commonly used algorithms for reducing the dimensionality of the reservoir are PCA and kernel PCA, which project data on the first D eigenvectors of a covariance matrix. When dealing with a prediction task, dimensionality reduction is applied to a single sequence of reservoir states generated by the input MTS [21]. On the other hand, in a classification task each MTS is associated to a different sequence of states [35]. If the MTS are represented by the last reservoir states, those are stacked into a matrix to which standard dimensionality reduction procedures were applied. When instead the whole set of representations is represented by a tensor, as discussed in Sec. III, the dimensionality reduction technique should account for factors of variation across more than one dimension.

Contrarily to the other modules, it is possible to implement a RC classifier without the dimensionality reduction module (as depicted by the skip connection in Fig. 2). However, as discussed in Sec. III, dimensionality reduction is particularly important when implementing the proposed reservoir model space representation or when using a bidirectional reservoir, in which cases the size of the representation $\mathbf{r}_\mathbf{X}$ grows with respect to a standard implementation.

C. Readout module

The readout module (decoder) classifies the representations and is either implemented as a linear readout, or a support vector machine (SVM) classifier, or a multi-layer perceptron (MLP). In a standard ESN, the readout is linear and is quickly trained by solving a convex optimization problem. However, a linear readout might not possess sufficient representational power for modeling the embeddings derived from the reservoir states. For this reason, several authors proposed to replace the linear decoding function $g(\cdot)$ in (5) with a nonlinear model, such as SVMs [12] or MLPs [41], [42], [43].

Readouts implemented as MLPs accomplished only modest results in the earliest works on RC [10]. However, nowadays MLPs can be trained much more efficiently by means of sophisticated initialization procedures [44] and regularization techniques [24]. The combination of ESNs with MLPs trained with modern techniques can substantially improve the performance as compared to a linear formulation [35]. Following recent trends in the deep learning literature we also investigate endowing the MLP readout with more expressive flexible nonlinear activation functions, namely Maxout [45] and kernel activation functions [46].

V. Experiments

We test a variety of RC-based architectures for MTS classification implemented with the proposed framework. We also compare against RNNs classifiers trained with gradient descent (LSTM and GRU), a 1-*NN* classifier based on the Dynamic Time Warping (DTW) similarity, SVM classifiers configured with pre-computed kernels for MTS, different deep learning architectures, and other state-of-the-art methods for time series classification. Depending whether the input MTS in the RC-based model is represented by the last reservoir state

($\mathbf{r}_\mathbf{X} = \mathbf{h}(T)$), or by the output space model (Sec. II-B), or by the reservoir space model (Sec. III), we refer to the models as *lESN*, *omESN* and *rmESN*, respectively. Whenever we use a bidirectional reservoir, a deep MLP readout, or a SVM readout we add the prefix "*bi-*", "*dr-*", and "*svm-*", respectively (e.g., *bi-lESN* or *dr-bi-rmESN*).

a) MTS datasets: To evaluate the performance of each classifier, we consider several MTS classification datasets taken from the UCR[3], UEA[4], and UCI repositories[5]. For completeness, we also included 3 univariate time series datasets. Details of the datasets are reported in Tab. I.

TABLE I: Time series benchmark datasets details. Column 2 to 5 report the number of variables (#V), samples in training and test set, and number of classes (#C), respectively. T_{min} is the length of the shortest MTS in the dataset and T_{max} the longest MTS. All datasets are available at our Github repository.

Dataset	**#V**	**Train**	**Test**	**#C**	**T_{min}**	**T_{max}**	**Source**
Swedish Leaf	1	500	625	15	128	128	UCR
Chlorine Conc.	1	467	3840	3	166	166	UCR
DistPhal	1	400	139	3	80	80	UCR
ECG	2	100	100	2	39	152	UCR
Libras	2	180	180	15	45	45	UCI
Ch.Traj.	3	300	2558	20	109	205	UCI
uWave	3	200	427	8	315	315	UCR
NetFlow	4	803	534	13	50	994	UEA
Wafer	6	298	896	2	104	198	UCR
Robot Fail.	6	100	64	4	15	15	UCI
Jp.Vow.	12	270	370	9	7	29	UCI
Arab. Dig.	13	6600	2200	10	4	93	UCI
Auslan	22	1140	1425	95	45	136	UCI
CMUsubject16	62	29	29	2	127	580	UEA
KickvsPunch	62	16	10	2	274	841	UEA
WalkvsRun	62	28	16	2	128	1918	UEA
PEMS	963	267	173	7	144	144	UCI

b) Blood samples dataset: As a case study on medical data, we analyze MTS of blood measurements obtained from electronic health records of patients undergoing a gastrointestinal surgery at the University Hospital of North Norway in 2004–2012.[6] Each patient is represented by a MTS of 10 blood sample measurements collected for 20 days after surgery. We consider the problem of classifying patients with and without surgical site infections from their blood samples, collected 20 days after surgery. The dataset consists of 883 MTS, of which 232 pertain to infected patients. The original MTS contain missing data, corresponding to measurements not collected for a given patient at certain time intervals, which are replaced by zero-imputation in a preprocessing step.

c) Experimental setup: For each dataset, we train the models 10 times using independent random parameters initializations. Each model is configured with the same hyperparameters in all the experiments. Since reservoirs are sensitive to the hyperparameter setting [47], a fine-tuning with independent cross-validation for each task is usually more important in classic RC models than in RNNs trained with gradient descent, such as LSTM and GRU. Nevertheless, we show that the proposed *rmESN* achieves competitive results even with fixed the hyperparameters. This indicates higher robustness and gives a practical advantage, compared to traditional RC approaches.

To provide a significant comparison, *lESN*, *omESN* and *rmESN* always share the same randomly generated reservoir configured with the following hyperparameters: number of internal units $R = 800$; spectral radius $\rho = 0.99$; non-zero connections percentage $\beta = 0.25$; input scaling $\omega = 0.15$; noise level $\xi = 0.001$. When classification is performed with a ridge regression readout, we set the regularization value $\lambda = 1.0$. The ridge regression prediction models, used to generate the model-space representation in *omESN* and *rmESN*, are configured with $\lambda = 5.0$. We always apply dimensionality reduction, as it provides important computational advantages (both in terms of memory and CPU time), as well as a regularization that improves the generalization capability and robustness of all RC models. For all experiments we select the number of subspace dimensions as $D = 75$, following a grid-search with k-fold cross-validation on the datasets of Tab. I (see the supplementary material for details).

LSTM and GRU are configured with $H = 30$ hidden units; the decoding function is implemented as a neural network with 2 dense layers of 20 hidden units followed by a softmax layer; the dropout probability is $p_{\text{drop}} = 0.1$; the ℓ_2 regularization parameter is $\lambda = 0.0001$; gradient descent is performed with the Adam algorithm [48] and we train the models for 5000 epochs Finally, the 1-*NN* classifier uses FastDTW [5], which is a computationally efficient approximation of the DTW[7]. We acknowledge additional approaches based on DTW [49], [50], which, however, are not discussed in this paper.

A. Performance comparison on benchmark datasets

In this experiment we compare the classification accuracy obtained on the representations yielded by the RC models, *lESN*, *omESN* and *rmESN*, by the fully trainable RNNs implementing either GRU or LSTM cells, and by the **1-NN** classifier based on DTW. Evaluation is performed on the benchmark datasets in Tab. I. The decoder is implemented by linear regression in the RC models and by a dense non-linear layer in LSTM and GRU. Since all the other parameters in LSTM and GRU are learned with gradient descent, the non-linearities in the decoding function do not result in additional computational costs. Results are reported in Fig. 3. The first panel reports the mean classification accuracy and standard deviation of 10 independent runs on all benchmark datasets, while the second panel shows the average execution time (in minutes on a logarithmic scale) required for training and testing the models.

The RC classifiers when configured with model space representations achieve a much higher accuracy than the basic *lESN*. In particular *rmESN*, which adopts our proposed representation, reaches the best overall mean accuracy and the low standard deviation indicates that it is also stable, i.e., it yields consistently good results regardless of the random initialization of the reservoir. The second-best accuracy is

[3] www.cs.ucr.edu/~eamonn/time_series_data

[4] https://www.groundai.com/project/the-uea-multivariate-time-series-classification-archive-2018/

[5] archive.ics.uci.edu/ml/datasets.html

[6] The dataset has been published in the AMIA Data Competition 2016

[7] we used the official Python library: https://pypi.org/project/fastdtw/

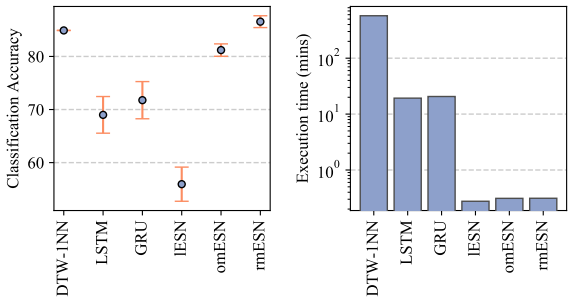

Fig. 3: Comparison of the average results obtained on all benchmark datasets.

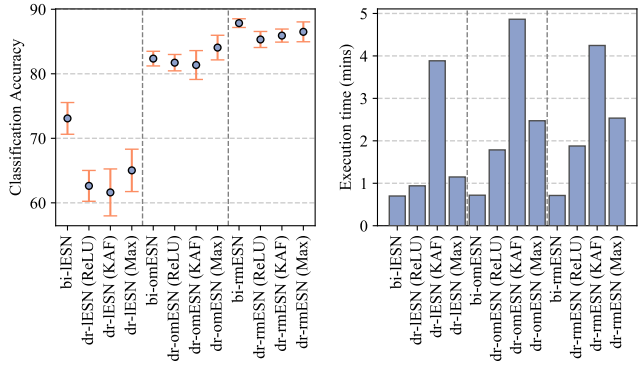

Fig. 4: Classification accuracy and execution time when using RC classifiers with a bidirectional reservoir and deep readouts, configured with ReLUs, KAFs, and Maxout activations.

obtained by 1-*NN* with DTW, while the classifiers based on LSTM and GRU perform only better than *lESN*. The results are particularly interesting since LSTM and GRU exploit supervised information to learn the representations $\mathbf{r_X}$ and they adopt a powerful non-linear discriminative classifier. On the other hand, the RC classifier configured with the model space representation outperforms the other RNN architectures, despite it relies on a linear classifier and the representations are learned in a complete unsupervised fashion.

In terms of execution time, all the RC classifiers are much faster than the competitors, as the average time for training and test is only few seconds. Remarkably, thanks to the proposed dimensionality reduction procedure, the *rmESN* classifier can be executed in a time comparable to *lESN*. The classifiers based on fully trainable RNNs, LSTM and GRU, require in average more than 20 minutes. Finally, 1-*NN* with DTW is much slower than the other methods despite the adopted "fast" implementation [5]. This is evident by looking at the huge gap in the execution time, which is more than 11 hours in average and goes beyond 30 hours in some datasets (see the supplementary material for the details).

B. Experiments with bidirectional reservoir and deep-readout

In this experiment we investigate how a bidirectional reservoir and a deep-readout, implemented by a MLP, influence classification accuracy and execution time in the RC-based classifiers. To further increase the flexibility of the deep readout, beside the standard rectified linear unit (ReLU), we also employ in the MLP more sophisticated transfer functions, namely Maxout [45] and kernel activation functions (KAFs) [46]. Thanks to their adaptable parameters, trained jointly with the other MLP weights, these functions can improve the expressive capability of the MLP classifier. We refer the reader to the original publications for details on their formulation. The deep readout is implemented with 3 layers of 20 neurons each and is trained for 5000 epochs, using a dropout probability $p_{\text{drop}} = 0.1$ and L_2 regularization parameter $\lambda = 0.001$.

We repeat the models evaluation on all the benchmark datasets and in Fig. 4 we report results in terms of classification accuracy and training time. We can see that both the bidirectional reservoir and deep readout improve, to different extents, the classification accuracy of each RC classifier. The largest improvement occurs for *lESN* when implemented with a bidirectional reservoir. This is expected since the last state representation in *lESN* depends mostly on the last observed values of the input MTS. Whenever the most relevant information is contained at the beginning of the input sequence or when the MTS are too long and the reservoir memory limitation forestall capturing long-term dependencies, the bidirectional architecture greatly improves the *lESN* representation. The bidirectional reservoir improves the performance also in *omESN* and *rmESN*. We recall that in these cases, rather than learning only a model for predicting the next output/state, when using a bidirectional reservoir the model also learns to solve a memorization task. The performance improvement for these model is lower than for *lESN*, probably because the representations obtained with a unidirectional reservoir are already good enough. Nevertheless, *bi-rmESN* reaches the highest overall accuracy.

A deep-readout enhances the capabilities of the classifier; improvements are larger in *lESN* and more limited in *omESN* and *rmESN*. Once again, this underlines that the weaker *lESN* representation benefits by adding more complexity in the pipeline. Even more than the bidirectional reservoir, a deep-readout trades greater modeling capabilities with more computational resources, especially when implemented with adaptive activation functions. Remarkably, when using Maxout functions rather than a standard ReLU, the training time is slightly higher, but there are significant improvements in the average classification accuracy. In particular, *dr-omESN (Maxout)* obtains almost the same performance of the basic version of *rmESN*. Another interesting result obtained by both Maxout and KAF is a reduction in the standard deviation of the accuracy, hence, a more robust classification.

In Fig. 5 we report the overall ranking, in terms of mean accuracy, of the 18 MTS classifier presented so far on the 17 classification datasets. On each dataset, the algorithms are ranked from 1 (best accuracy) to 18 (worst accuracy) and the table depicts the average of the ranks. It emerges that the proposed reservoir model space representation is the key factor to achieve the highest classification accuracy and that by introducing further complexity, by means of deep readouts

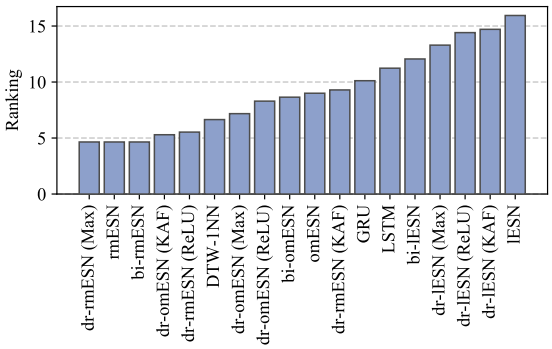

Fig. 5: Ranking in terms of mean accuracy obtained by the MTS classifiers on all the 14 datasets. A lower value in ranking indicates better average accuracy.

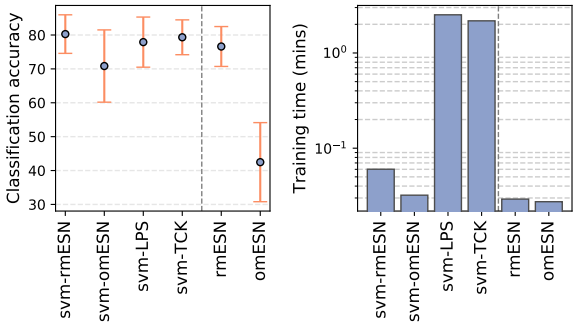

Fig. 6: Classification accuracy obtained with SVM using different precomputed kernels. We also report the results obtained by *rmESN* and *omESN* on the same problem.

and bidirectional reservoir, performance are further improved.

C. Classification of blood samples MTS

Here, we analyze the blood sample MTS and evaluate the RC classifiers configured with a SVM readout. We consider only *omESN* and *rmESN* since, as demonstrated in the previous experiments, they provide an optimal compromise between training efficiency and classification accuracy. Since we adopt a kernel method to implement the decoding function (3) (readout), we compare against two state-of-the-art kernels for MTS. The first is the learned pattern similarity (LPS) [51], which identifies segments-occurrence within the MTS by means of regression trees. Those are used to generate a bag-of-words type compressed representation, on which the similarity scores are computed. The second method is the time series cluster kernel (TCK) [52], which is based on an ensemble learning procedure wherein the clustering results of several Gaussian mixture models, which are fit many times on random subsets of the original dataset, are joined to form the final kernel.

For LPS and TCK, an SVM is configured with the precomputed kernels returned by the two procedures, while for *omESN* and *rmESN* we build a RBF kernel with bandwidth γ. We optimize on a validation set the SVM hyperparameters, which are the smoothness of the decision hyperplane, c, and bandwidth, γ (only *omESN* and *rmESN*). The hyperparameter space is explored with a grid search, by varying c in $[0.1, 5.0]$ with resolution 0.1 and γ in $[0.01, 1.0]$ with resolution 0.01. LPS is configured using 200 regression trees and maximum segments length 10. TCK is configured with 40 different random initializations and 30 maximum mixtures for each partition. RC classifiers use the same hyperparameters as in the previous experiments.

To compute the performance of the models, those are evaluated 15 times with independent random initializations and randomly shuffling and splitting the original dataset into training, validation, and test set, containing 70%, 10% and 20% of the original samples, respectively. Each time, we normalize the data by subtracting the mean and dividing by the standard deviation of each variable in the training set, excluding the imputed values. The results in terms of classification accuracy and training time are depicted in Fig. 6. For completeness, we report also the classification results obtained on this task by *omESN* and *rmESN*, with $g(\cdot)$ implemented as a linear readout. Also in this case, *rmESN* outperforms *omESN* either when it is configured with a linear or a SVM readout. As for the deep-readout, we notice that the more powerful decoding function improves the classification accuracy in *rmESN* only slightly, while the increment in *omESN* is much larger. The *svm-rmESN* manages to slightly outperform the SVM classifiers configured with LPS and TCK kernels. We notice standard deviations in all methods are quite high, since the train/validation/test splits are generated randomly at every iteration and, therefore, the classification task changes each time. *svm-TCK* yields results with the lowest standard deviation and is followed by *svm-rmESN* and *rmESN*. The SVM readout slightly increases the training time of the RC models, but they are still much faster than the TCK and LPS kernels.

VI. Comparison with deep learning baselines on the classification of univariate time series

Although the proposed framework is specifically designed for the classification of MTS, we conclude by considering additional experiments on univariate time series classification datasets [8]. Compared to the multivariate case, algorithms designed for this task can exploit stronger biases to attain high classification performance. We also notice that in the case of univariate time series we do not adopt the proposed extension of PCA for multivariate temporal data, but a regular PCA is used instead. Nevertheless, we show that our method can achieve competitive results compared to state-of-the-art methods for time series classification. We choose *rmESN* as the representative model of the RC classifiers, which provides a good trade-off between classification accuracy and training time. Tab. II reports the results obtained by *rmESN* and several different methods. We implement baselines based on popular deep learning architectures (MLP, FCN and ResNets) [53], [54], and report results where available on the original papers for BOSS [55], PROP [56], COTE [57], an advanced deep learning architecture that combines an LSTM with attention

[8]we used datasets from http://www.timeseriesclassification.com

TABLE II: Results on univariate TS classification. Best results are in bold, second best are underlined.

Dataset	MLP	FCN	ResNet	PROP	COTE	BOSS	LSTM-FCN	MMCL	TSML	rm-ESN
Adiac	24.8	14.3	17.4	35.3	23.3	30.2	**85.9**	72.6	73.7	81.2
Chl. Conc.	**87.2**	84.3	82.8	0.64	68.6	65.5	80.1	–	–	85.6
DistPhal	74.7	79.0	74.0	68.3	74.7	–	**81.7**	–	–	75.5
Earthquakes	79.2	80.1	78.6	71.9	–	80.7	**83.5**	–	–	79.7
ECG5000	93.5	94.1	93.1	65.0	–	89.0	94.7	–	–	**95.1**
FaceAll	88.5	92.9	83.4	84.8	89.5	75.9	**94.0**	–	76.7	93.5
FaceFour	83.0	93.2	93.2	90.1	90.1	**96.6**	94.3	–	95.5	**96.6**
GunPoint	93.3	**100**	99.3	99.3	99.3	**100**	**100**	–	98	**100**
ItalyPower	96.4	**97.0**	96.0	96.1	96.4	91.4	96.3	–	96.4	96.4
Lightning2	72.1	80.3	75.4	**88.5**	83.6	73.8	80.3	75.4	80.3	74.2
Swe. Leaf	89.3	96.6	95.8	91.5	95.4	85.9	**97.9**	–	93.0	94.5

to a CNN architecture (LSTM-FCN) [58], a model-metric co-learning methodology for sequence classification that learns in the model space (MMCL) [59], and a feature-based model (TSML) [6].

It is possible to see that the complex deep learning architecture LSTM-FCN achieves, on average, the best classification accuracy. On the other hand, the *rmESN* model equipped with a simple linear readout achieves results that are competitive to those obtained by much more complex models, while requiring only few seconds to be trained.

VII. Conclusions and future work

We proposed a RC classifier based on the reservoir model space representation, which can be categorized as a hybrid generative-discriminative approach. Specifically, the parameters of a model that predict the next reservoir states characterize the generative process of the dynamical input features. Such parameters are, in turn, processed by a discriminative decoder that classifies the original time series. Usually, in a hybrid generative-discriminative approach where data are assumed to be generated by a parametric distribution, the subsequent discriminative model cannot be specified independently from the generative model type, without introducing biases in the classification [60]. However, in our case the reservoir is flexible and generic, as it can extract a large variety of features from the underlying dynamical system, without posing constraints on the particular model underlying the data distribution. This provides two advantages: (i) different discriminative models can be used in conjunction with the same reservoir model space representation and (ii) the same reservoir can model data from different distributions.

To make the reservoir model space tractable we designed an unsupervised dimensionality reduction procedure, suitable for datasets represented as high-order tensors. Our dimensionality reduction greatly reduces computational time and memory usage and provides a regularization that prevents overfitting, especially in complex discriminative classifiers. Finally, we defined a unified framework and investigated several alternatives to build RC classifiers, focusing on unsupervised procedures to learn fixed-size representations of the MTS.

We considered several real-world datasets for classification of MTS, showing that the RC classifier equipped with the proposed representation achieves superior performance both in terms of classification accuracy and execution time. We analyzed how a bidirectional reservoir and a deep readout affect the performance (both in time and accuracy) of RC-based classifiers configured with different representations. We found that combining the reservoir model space with these more sophisticated architectures improves accuracy only slightly, pointing to the already strong informative content of this representation. We also considered a medical case study of blood samples time series and obtained superior performance compared to state-of-the-art kernels for MTS. We concluded by comparing with state-of-the-art methods on the classification of univariate time series and showed that, even on those tasks, our approach achieves competitive results.

References

[1] K. Ø. Mikalsen, F. M. Bianchi, C. Soguero-Ruiz, S. O. Skrøvseth, R.-O. Lindsetmo, A. Revhaug, and R. Jenssen, "Learning similarities between irregularly sampled short multivariate time series from EHRs," in Proc. 3rd International Workshop on Pattern Recognition for Healthcare Analytics at ICPR 2016, 2016.

[2] E. Carden and J. Brownjohn, "Arma modelled time-series classification for structural health monitoring of civil infrastructure," Mechanical Systems and Signal Processing, vol. 22, no. 2, pp. 295–314, 2008.

[3] D. Hunt and D. Parry, "Using echo state networks to classify unscripted, real-world punctual activity," in Engineering Applications of Neural Networks. Springer, 2015, pp. 369–378.

[4] E. Trentin, S. Scherer, and F. Schwenker, "Emotion recognition from speech signals via a probabilistic echo-state network," Pattern Recognition Letters, vol. 66, pp. 4–12, 2015.

[5] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," Intelligent Data Analysis, vol. 11, no. 5, pp. 561–580, 2007.

[6] C. O'Reilly, K. Moessner, and M. Nati, "Univariate and multivariate time series manifold learning," Knowledge-Based Systems, vol. 133, pp. 1–16, 2017.

[7] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," Data Mining and Knowledge Discovery, vol. 31, no. 3, pp. 606–660, 2017.

[8] M. G. Baydogan and G. Runger, "Learning a symbolic representation for multivariate time series classification," Data Mining and Knowledge Discovery, vol. 29, no. 2, pp. 400–422, 2015.

[9] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),. IEEE, 2013, pp. 6645–6649.

[10] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," Computer Science Review, vol. 3, no. 3, pp. 127–149, 2009.

[11] S. Scardapane and D. Wang, "Randomness in neural networks: an overview," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 7, no. 2, 2017.

[12] F. Bianchi, S. Scardapane, A. Uncini, A. Rizzi, and A. Sadeghian, "Prediction of telephone calls load using Echo State Network with exogenous variables," Neural Networks, vol. 71, pp. 204–213, 2015.

[13] F. M. Bianchi, E. De Santis, A. Rizzi, and A. Sadeghian, "Short-term electric load forecasting using echo state networks and PCA decomposition," IEEE Access, vol. 3, pp. 1931–1943, Oct. 2015.

[14] A. Rodan, A. Sheta, and H. Faris, "Bidirectional reservoir networks trained using SVM+ privileged information for manufacturing process modeling," Soft Computing, vol. 21, no. 22, pp. 6811–6824, 2017.

[15] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," GMD Technical Report, vol. 148, no. 34, 2001.

[16] Q. Ma, L. Shen, W. Chen, J. Wang, J. Wei, and Z. Yu, "Functional echo state network for time series classification," Information Sciences, vol. 373, pp. 1–20, 2016.

[17] M. Skowronski and J. Harris, "Minimum mean squared error time series classification using an echo state network prediction model," in Proc. 2006 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2006.

[18] W. Aswolinskiy, R. Reinhart, and J. Steil, "Time series classification in reservoir-and model-space: a comparison," in IAPR Workshop on Artificial Neural Networks in Pattern Recognition. Springer, 2016, pp. 197–208.

[19] H. Chen, F. Tang, P. Tino, and X. Yao, "Model-based kernel for efficient time series analysis," in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013, pp. 392–400.

[20] H. Chen, P. Tino, A. Rodan, and X. Yao, "Learning in the model space for cognitive fault diagnosis," IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 1, pp. 124–136, 2014.

[21] S. Løkse, F. M. Bianchi, and R. Jenssen, "Training echo state networks with regularization through dimensionality reduction," Cognitive Computation, vol. 9, no. 3, pp. 364–378, Jun 2017.

[22] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis. Springer, 2017.

[23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in neural information processing systems, 2014, pp. 3104–3112.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[25] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," arXiv preprint arXiv:1409.2329, 2014.

[26] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in International Conference on Machine Learning, 2013, pp. 1310–1318.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

[29] F. M. Bianchi, L. Livi, and C. Alippi, "Investigating echo-state networks dynamics by means of recurrence analysis," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 2, pp. 427–439, Feb 2018.

[30] L. Livi, F. M. Bianchi, and C. Alippi, "Determination of the edge of criticality in echo state networks through fisher information maximization," IEEE transactions on neural networks and learning systems, vol. 29, no. 3, pp. 706–717, 2018.

[31] Z. Gong, H. Chen, B. Yuan, and X. Yao, "Multiobjective learning in the model space for time series classification," IEEE transactions on cybernetics, vol. 49, no. 3, pp. 918–932, 2018.

[32] L. Wang, Z. Wang, and S. Liu, "An effective multivariate time series classification approach using echo state network and adaptive differential evolution algorithm," Expert Systems with Applications, vol. 43, pp. 237–249, 2016.

[33] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," arXiv preprint arXiv:1807.03748, 2018.

[34] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in Advances in neural information processing systems, 2002, pp. 841–848.

[35] F. M. Bianchi, S. Scardapane, S. Løkse, and R. Jenssen, "Bidirectional deep-readout echo state networks," in European Symposium on Artificial Neural Networks, 2018.

[36] F. M. Bianchi, L. Livi, K. Ø. Mikalsen, M. Kampffmeyer, and R. Jenssen, "Learning representations of multivariate time series with missing data," Pattern Recognition, vol. 96, p. 106973, 2019.

[37] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," SIAM review, vol. 51, no. 3, pp. 455–500, 2009.

[38] C. Gallicchio and A. Micheli, "Echo state property of deep reservoir computing networks," Cognitive Computation, vol. 9, no. 3, pp. 337–350, Jun 2017.

[39] A. Rodan and P. Tino, "Simple deterministically constructed cycle reservoirs with regular jumps," Neural Computation, vol. 24, no. 7, pp. 1822–1852, 2012, pMID: 22428595.

[40] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," Neural Networks, vol. 18, no. 5-6, pp. 602–610, 2005.

[41] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," Neural computation, vol. 14, no. 11, pp. 2531–2560, 2002.

[42] K. Bush and C. Anderson, "Modeling reward functions for incomplete state representations via echo state networks," in Proc. International Joint Conference on Neural Networks (IJCNN), vol. 5. IEEE, 2005, pp. 2995–3000.

[43] Š. Babinec and J. Pospíchal, "Merging echo state and feedforward neural networks for time series forecasting," in International Conference on Artificial Neural Networks. Springer, 2006, pp. 367–375.

[44] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[45] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks." Proc. 30th International Conference on Machine Learning (ICML), 2013.

[46] S. Scardapane, S. Van Vaerenbergh, S. Totaro, and A. Uncini, "Kafnets: kernel-based non-parametric activation functions for neural networks," arXiv preprint arXiv:1707.04035, 2017.

[47] F. M. Bianchi, L. Livi, C. Alippi, and R. Jenssen, "Multiplex visibility graphs to investigate recurrent neural network dynamics," Scientific reports, vol. 7, p. 44037, 2017.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[49] T. Górecki and M. Łuczak, "Multivariate time series classification with parametric derivative dynamic time warping," Expert Systems with Applications, vol. 42, no. 5, pp. 2305–2312, 2015.

[50] J. Mei, M. Liu, Y.-F. Wang, and H. Gao, "Learning a mahalanobis distance-based dynamic time warping measure for multivariate time series classification," IEEE transactions on Cybernetics, vol. 46, no. 6, pp. 1363–1374, 2015.

[51] M. G. Baydogan and G. Runger, "Time series representation and similarity based on local autopatterns," Data Mining and Knowledge Discovery, vol. 30, no. 2, pp. 476–509, 2016.

[52] K. Ø. Mikalsen, F. M. Bianchi, C. Soguero-Ruiz, and R. Jenssen, "Time series cluster kernel for learning similarities between multivariate time series with missing data," Pattern Recognition, vol. 76, pp. 569 – 581, 2018.

[53] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," Data Mining and Knowledge Discovery, vol. 33, no. 4, pp. 917–963, 2019.

[54] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in 2017 international joint conference on neural networks (IJCNN). IEEE, 2017, pp. 1578–1585.

[55] P. Schäfer, "Scalable time series classification," Data Mining and Knowledge Discovery, vol. 30, no. 5, pp. 1273–1298, 2016.

[56] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," Data Mining and Knowledge Discovery, vol. 29, no. 3, pp. 565–592, 2015.

[57] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: the collective of transformation-based ensembles," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 9, pp. 2522–2535, 2015.

[58] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," IEEE Access, vol. 6, pp. 1662–1669, 2017.

[59] H. Chen, F. Tang, P. Tino, A. G. Cohn, and X. Yao, "Model metric co-learning for time series classification," in Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[60] K. H. Brodersen, T. M. Schofield, A. P. Leff, C. S. Ong, E. I. Lomakina, J. M. Buhmann, and K. E. Stephan, "Generative embedding for model-based classification of fmri data," PLoS computational biology, vol. 7, no. 6, p. e1002079, 2011.

APPENDIX

A. Selection of the optimal number of subspace dimensions

To determine the optimal number of subspace dimensions D, we evaluate how the average training time and classification accuracy (computed with a k-fold cross-validation procedure) of the RC classifiers varies on the dataset in Tab. I.

We report the average results in Fig. 7.

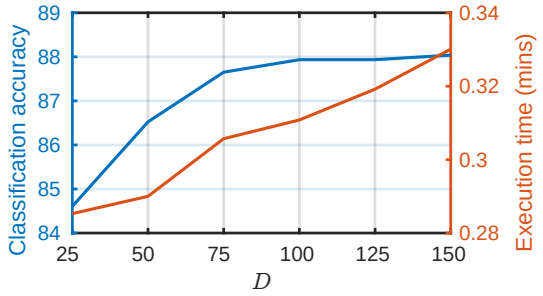

Fig. 7: Average classification accuracy and execution time for different dimensions D of the space with reduced dimensionality.

While the training time increases approximately linearly with D, it is possible to identify an "elbow" in the classification accuracy for $D = 75$, which is the value we select in all our experiments.

B. Statistical analysis of the results

In the following, we provide the details of the aggregated results reported in Sec. V-A and Sec. V-B. Fig. 8 depicts the ranking of the accuracy achieved by each MTS classifier on the benchmark datasets described in Tab. I. Best performance (higher accuracy) correspond to lower values in ranking and to a darker color code.

To evaluate the significance of the differences in performance obtained by the different MTS classifiers on the dataset, we first performed a Friedman test on the rankings. We obtained a p-value of $1.11e-16$, which indicates the presence of statistically significant differences. Then, we performed the Finner post-hoc test, to compute for each pair of classifiers if the difference in performance is statistically significant. In Fig. 9 we report the adjusted p-values obtained by testing the performance of each pair of classifiers. We highlighted in yellow test results with p-values lower than 0.05 and in green the results with p-values lower than 0.01.

We also report in Fig. 10 a critical-difference diagram based on the Wilcoxon-Holm method to detect pairwise significance. For details about the construction and interpretation of the diagram, we refer the reader to the related Python repository[9].

C. Detailed results on the benchmark datasets

The tables below report the detailed results obtained on each dataset by the time series classifiers analyzed in section Sec. V-A and Sec. V-B. For each algorithm we performed 10 independent runs and we report the mean accuracy, standard deviation accuracy, mean F1 score, standard deviation F1 score, and mean execution time (in minutes). For the Arabic Digits dataset we do not report the results for 1-NN with DTW, as the execution time for the simulation exceeded 48 hours.

[9]https://github.com/hfawaz/cd-diagram

TABLE III: Results on Swedish Leaf dataset.

Swedish Leaf	Accuracy	F1 score	Time (mins)
lESN	62.08±5.47	0.59±0.06	0.15
omESN	69.47±2.01	0.66±0.02	0.16
rmESN	94.51±0.80	0.93±0.01	0.16
bi-lESN	77.98±4.34	0.75±0.05	0.37
bi-omESN	70.90±1.81	0.68±0.02	0.39
bi-rmESN	96.25±0.66	0.89±0.01	0.39
dr-lESN (ReLU)	85.14±0.95	0.83±0.01	0.71
dr-omESN (ReLU)	92.79±2.27	0.91±0.02	0.72
dr-rmESN (ReLU)	96.64±0.77	0.94±0.01	1.41
dr-lESN (Max)	86.42±1.37	0.85±0.02	0.68
dr-omESN (Max)	93.87±1.25	0.92±0.01	0.69
dr-rmESN (Max)	95.56±0.66	0.95±0.01	1.61
dr-lESN (KAF)	84.43±2.03	0.82±0.02	2.46
dr-omESN (KAF)	92.14±0.72	0.87±0.01	2.51
dr-rmESN (KAF)	95.47±0.86	0.93±0.01	2.58
LSTM	89.58±0.71	0.86±0.01	8.60
GRU	88.24±1.62	0.86±0.02	9.39
DTW-1NN	81.72	0.79	329.99

TABLE IV: Results on Chlorine Concentration dataset.

Chlo Conc	Accuracy	F1 score	Time (mins)
lESN	68.18±0.26	0.57±0.00	0.62
omESN	76.15±0.28	0.63±0.01	0.68
rmESN	85.60±0.41	0.78±0.01	0.70
bi-lESN	58.18±0.38	0.49±0.00	1.35
bi-omESN	77.99±0.56	0.68±0.01	1.40
bi-rmESN	83.72±0.62	0.79±0.01	1.42
dr-lESN (ReLU)	80.79±2.09	0.80±0.02	1.10
dr-omESN (ReLU)	80.38±0.67	0.80±0.01	1.16
dr-rmESN (ReLU)	79.68±0.69	0.79±0.01	1.78
dr-lESN (Max)	85.95±1.21	0.86±0.01	1.21
dr-omESN (Max)	83.05±0.67	0.83±0.01	1.25
dr-rmESN (Max)	85.07±1.36	0.85±0.01	2.14
dr-lESN (KAF)	72.42±4.23	0.70±0.05	3.05
dr-omESN (KAF)	67.04±2.64	0.66±0.03	3.07
dr-rmESN (KAF)	81.21±2.63	0.81±0.03	3.33
LSTM	60.42±1.10	0.56±0.03	9.07
GRU	60.85±1.13	0.56±0.02	9.82
DTW-1NN	62.60	0.62	2414.91

TABLE V: Results on Distal Phalanx Outline dataset.

Dist Phal	Accuracy	F1 score	Time (mins)
lESN	68.92±0.54	0.67±0.01	0.06
omESN	67.48±0.29	0.63±0.00	0.07
rmESN	75.57±1.32	0.74±0.02	0.07
bi-lESN	67.34±1.08	0.65±0.01	0.20
bi-omESN	68.06±0.98	0.64±0.02	0.20
bi-rmESN	75.23±0.73	0.72±0.01	0.21
dr-lESN (ReLU)	67.77±0.84	0.68±0.01	0.50
dr-omESN (ReLU)	73.67±1.55	0.74±0.02	0.50
dr-rmESN (ReLU)	75.54±1.02	0.76±0.01	1.03
dr-lESN (Max)	69.35±2.35	0.65±0.02	0.62
dr-omESN (Max)	72.23±2.97	0.72±0.03	0.62
dr-rmESN (Max)	76.52±0.70	0.73±0.01	1.34
dr-lESN (KAF)	70.22±1.33	0.70±0.01	2.41
dr-omESN (KAF)	73.09±1.91	0.73±0.02	2.39
dr-rmESN (KAF)	75.52±2.00	0.72±0.02	2.70
LSTM	70.94±2.93	0.71±0.03	4.48
GRU	72.66±1.29	0.73±0.01	4.88
DTW-1NN	74.82	0.75	24.50

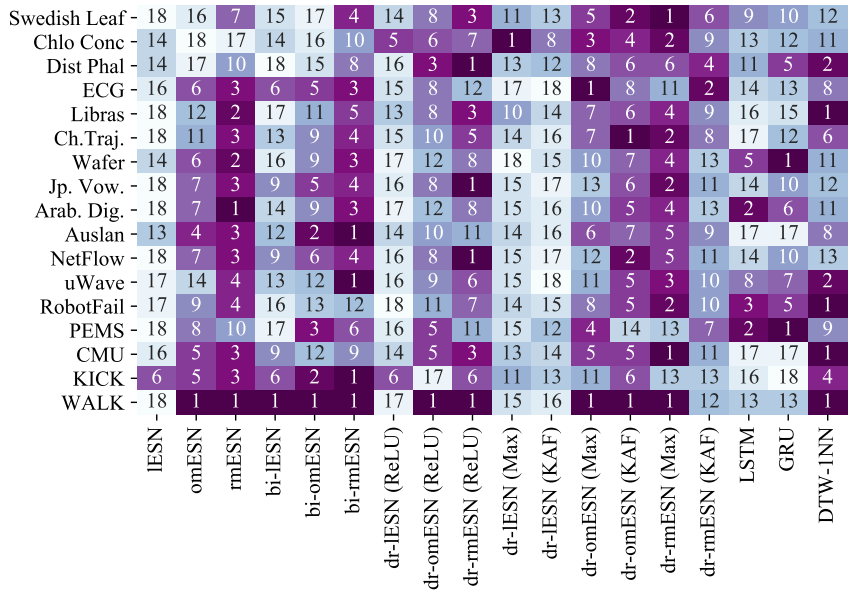

Fig. 8: Ranking of the accuracy obtained by the MTS classifiers on benchmark classification dataset.

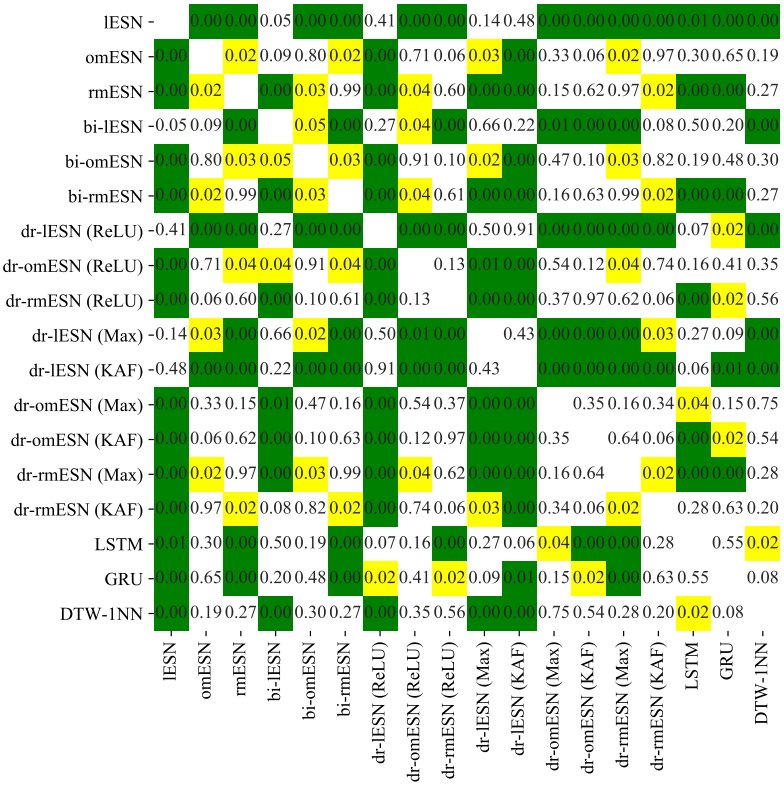

Fig. 9: Results (p-values) of the post-hoc test. Yellow boxes indicate p-value < 0.05, Green boxes indicate p-value < 0.01.

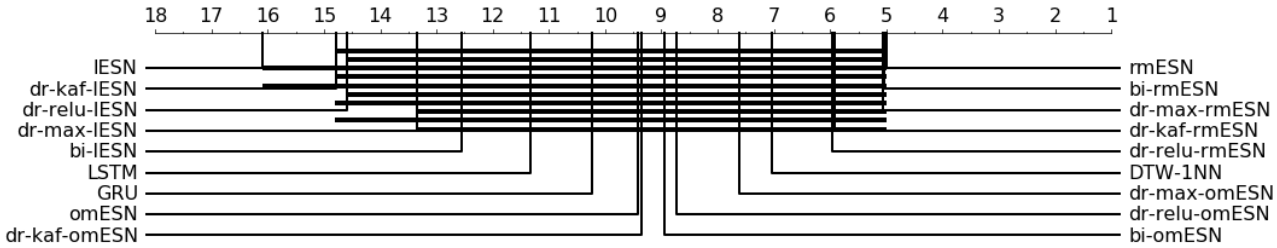

Fig. 10: Critical difference diagram.

TABLE VI: Results on Electrocardiography dataset.

ECG	Accuracy	F1 score	Time (mins)
lESN	69.00±2.19	0.81±0.01	0.05
omESN	84.60±0.49	0.89±0.00	0.05
rmESN	85.20±0.75	0.89±0.00	0.05
bi-lESN	84.60±2.06	0.89±0.01	0.16
bi-omESN	84.80±1.17	0.89±0.01	0.16
bi-rmESN	85.20±0.40	0.89±0.00	0.17
dr-lESN (ReLU)	71.60±4.03	0.79±0.04	0.17
dr-omESN (ReLU)	84.00±1.41	0.88±0.01	0.17
dr-rmESN (ReLU)	83.40±1.62	0.88±0.01	0.28
dr-lESN (Max)	68.80±3.43	0.76±0.03	0.20
dr-omESN (Max)	86.60±1.02	0.90±0.01	0.21
dr-rmESN (Max)	83.80±1.60	0.88±0.01	0.38
dr-lESN (KAF)	65.60±7.17	0.74±0.06	0.65
dr-omESN (KAF)	85.40±0.49	0.89±0.00	0.64
dr-rmESN (KAF)	84.00 ±1.10	0.88±0.01	0.69
LSTM	76.20±4.26	0.82±0.03	2.10
GRU	81.20±3.49	0.86±0.02	2.27
DTW-1NN	84.00	0.88	11.42

TABLE VII: Results on Libras dataset.

Libras	Accuracy	F1 score	Time (mins)
lESN	59.89±0.65	0.59±0.01	0.04
omESN	77.22±3.33	0.75±0.04	0.04
rmESN	88.11±1.43	0.88±0.02	0.04
bi-lESN	63.33±2.30	0.63±0.02	0.13
bi-omESN	77.78±0.99	0.77±0.01	0.13
bi-rmESN	86.00±0.65	0.86±0.01	0.14
dr-lESN (ReLU)	72.56±2.91	0.72±0.02	0.25
dr-omESN (ReLU)	80.78±2.29	0.80±0.02	0.26
dr-rmESN (ReLU)	87.22±1.76	0.87±0.02	0.48
dr-lESN (Max)	78.00±1.43	0.78±0.02	0.29
dr-omESN (Max)	84.44±2.17	0.84±0.02	0.30
dr-rmESN (Max)	86.67±0.79	0.87±0.01	0.62
dr-lESN (KAF)	72.22±2.25	0.72±0.02	1.11
dr-omESN (KAF)	79.67±2.40	0.79±0.02	1.11
dr-rmESN (KAF)	84.78±1.03	0.85±0.01	1.18
LSTM	68.22±2.62	0.68±0.03	1.17
GRU	71.56±4.60	0.71±0.05	1.25
DTW-1NN	88.33	0.88	9.52

TABLE VIII: Results on Character Trajectory dataset.

Ch.Traj.	Accuracy	F1 score	Time (mins)
lESN	21.41±7.01	0.17±0.06	0.46
omESN	91.39±0.91	0.91±0.01	0.50
rmESN	97.36±0.24	0.97±0.00	0.51
bi-lESN	51.11±8.37	0.49±0.09	1.01
bi-omESN	94.36±0.40	0.94±0.00	1.06
bi-rmESN	97.00±0.11	0.97±0.00	1.06
dr-lESN (ReLU)	44.05±5.12	0.43±0.05	0.82
dr-omESN (ReLU)	94.08±0.96	0.94±0.01	0.88
dr-rmESN (ReLU)	96.58±0.67	0.97±0.01	1.26
dr-lESN (Max)	44.71±4.81	0.44±0.05	0.87
dr-omESN (Max)	95.54±0.34	0.95±0.00	0.96
dr-rmESN (Max)	97.52±0.54	0.97±0.01	1.47
dr-lESN (KAF)	40.13±8.03	0.39±0.08	2.18
dr-omESN (KAF)	94.50±0.60	0.94±0.01	2.25
dr-rmESN (KAF)	97.59±0.23	0.97±0.00	2.38
LSTM	37.10±14.62	0.33±0.16	8.50
GRU	70.79±17.71	0.70±0.19	9.13
DTW-1NN	95.78	0.96	1218.31

TABLE IX: Results on Wafer dataset.

Wafer	Accuracy	F1 score	Time (mins)
lESN	89.35±0.09	0.94±0.00	0.22
omESN	95.71±1.05	0.98±0.01	0.24
rmESN	97.78±0.29	0.98±0.00	0.24
bi-lESN	88.91±0.32	0.94±0.00	0.52
bi-omESN	95.25±0.78	0.97±0.00	0.54
bi-rmESN	97.01±0.40	0.98±0.00	0.54
dr-lESN (ReLU)	88.50±0.95	0.94±0.00	0.53
dr-omESN (ReLU)	94.51±1.13	0.97±0.01	0.59
dr-rmESN (ReLU)	95.60±0.82	0.98±0.00	0.92
dr-lESN (Max)	88.30±1.85	0.94±0.01	0.61
dr-omESN (Max)	95.11±1.08	0.97±0.01	0.75
dr-rmESN (Max)	96.85±0.65	0.98±0.00	1.14
dr-lESN (KAF)	88.93±1.45	0.94±0.01	1.85
dr-omESN (KAF)	93.68±1.07	0.96±0.01	1.94
dr-rmESN (KAF)	95.69±1.00	0.98±0.01	2.02
LSTM	96.32±3.70	0.98±0.02	7.58
GRU	98.41±0.86	0.99±0.00	8.22
DTW-1NN	95.09	0.97	396.99

TABLE X: Results on Japanese Vowels dataset.

Jp. Vow.	Accuracy	F1 score	Time (mins)
lESN	80.00±5.37	0.80±0.05	0.04
omESN	95.35±0.46	0.95±0.00	0.05
rmESN	97.83±0.50	0.98±0.00	0.05
bi-lESN	94.05±0.70	0.94± 0.01	0.14
bi-omESN	97.35±0.40	0.97±0.00	0.15
bi-rmESN	97.62±0.46	0.98±0.00	0.15
dr-lESN (ReLU)	83.84±4.25	0.84±0.04	0.32
dr-omESN (ReLU)	94.76±0.86	0.95±0.01	0.44
dr-rmESN (ReLU)	98.14±0.44	0.97±0.00	0.67
dr-lESN (Max)	86.22±3.95	0.86±0.04	0.31
dr-lESN (KAF)	82.97±3.90	0.83±0.04	1.18
dr-omESN (Max)	93.41±0.40	0.93±0.00	0.46
dr-omESN (KAF)	93.57±0.46	0.94±0.01	1.33
dr-rmESN (KAF)	96.97±0.63	0.97±0.01	1.24
dr-rmESN (Max)	97.99±0.65	0.97±0.01	0.80
LSTM	92.70±1.36	0.93±0.01	1.15
GRU	94.00±2.21	0.94±0.02	1.24
DTW-1NN	93.51	0.94	19.23

TABLE XI: Results on Arabic Digits dataset.

Arab. Dig.	Accuracy	F1 score	Time (mins)
lESN	39.77±6.08	0.26±0.06	0.92
omESN	95.63±0.51	0.95±0.01	1.07
rmESN	98.12±0.21	0.98±0.00	1.16
bi-lESN	77.44±2.13	0.76±0.03	2.66
bi-omESN	94.92±0.27	0.95±0.00	2.80
bi-rmESN	96.46±0.44	0.96±0.00	2.90
dr-lESN (ReLU)	45.82±2.66	0.45±0.02	6.57
dr-omESN (ReLU)	92.48±0.32	0.92±0.00	10.08
dr-rmESN (ReLU)	95.39±0.52	0.95±0.01	15.42
dr-lESN (Max)	46.90±4.12	0.46±0.04	9.73
dr-lESN (KAF)	46.11±3.03	0.45±0.03	40.17
dr-omESN (Max)	94.01±0.44	0.94±0.00	16.86
dr-omESN (KAF)	91.66±0.59	0.92±0.01	44.52
dr-rmESN (Max)	96.10±0.35	0.96±0.00	22.18
dr-rmESN (KAF)	96.02±0.76	0.96±0.01	41.16
LSTM	96.61±0.69	0.97±0.01	82.41
GRU	95.98±2.91	0.96±0.03	90.82
DTW-1NN	–	–	$>$ 48 hours

TABLE XII: Results on Australian Sign Language dataset.

Auslan	Accuracy	F1 score	Time (mins)
lESN	1.35±0.26	0.00±0.00	0.34
omESN	94.53±0.43	0.94±0.00	0.39
rmESN	97.25±0.25	0.97±0.00	0.40
bi-lESN	56.94±0.95	0.56±0.01	0.80
bi-omESN	97.39±0.30	0.97±0.00	0.85
bi-rmESN	97.64±0.35	0.98±0.00	0.85
dr-lESN (ReLU)	1.31±0.28	0.01±0.00	2.09
dr-omESN (ReLU)	77.40±2.12	0.77±0.02	3.32
dr-rmESN (ReLU)	73.47±4.77	0.73±0.05	4.01
dr-lESN (Max)	1.31±0.21	0.01±0.00	2.09
dr-lESN (KAF)	1.09±0.08	0.00±0.00	7.65
dr-omESN (Max)	87.94±0.44	0.88±0.00	2.66
dr-rmESN (KAF)	85.75±0.87	0.86±0.01	8.56
dr-rmESN (Max)	88.70±1.38	0.89±0.01	4.49
dr-omESN (KAF)	84.53±2.37	0.84±0.02	8.75
LSTM	1.05±0.00	0.00±0.00	18.89
GRU	1.05±0.00	0.00±0.00	20.49
DTW-1NN	85.61	0.85	1650.32

TABLE XIII: Results on Network Flow dataset.

NetFlow	Accuracy	F1 score	Time (mins)
lESN	79.13±5.40	0.82±0.06	0.50
omESN	94.48±0.50	0.96±0.01	0.51
rmESN	96.96±0.54	0.98±0.01	0.52
bi-lESN	93.19±0.74	0.96±0.02	1.28
bi-omESN	95.48±0.43	0.96±0.01	1.26
bi-rmESN	96.75±0.50	0.98±0.01	1.17
dr-lESN (ReLU)	82.97±4.29	0.86±0.05	0.88
dr-omESN (ReLU)	93.89±0.90	0.97±0.02	0.90
dr-rmESN (ReLU)	97.27±0.47	0.98±0.01	1.27
dr-lESN (Max)	85.35±3.99	0.88±0.05	0.88
dr-lESN (KAF)	82.11±3.93	0.85±0.05	0.98
dr-omESN (Max)	92.54±0.44	0.95±0.01	1.54
dr-omESN (KAF)	92.70±0.50	0.95±0.01	2.36
dr-rmESN (Max)	96.11±0.66	0.98±0.02	2.48
dr-rmESN (KAF)	97.12±0.69	0.98±0.02	2.45
LSTM	91.84±1.39	0.95±0.02	8.72
GRU	93.13±2.25	0.96±0.03	9.42
DTW-1NN	92.08	0.94	407.73

TABLE XIV: Results on uWave dataset.

uWave	Accuracy	F1 score	Time (mins)
lESN	52.01±1.53	0.50±0.02	0.42
omESN	65.42±1.29	0.64±0.01	0.43
rmESN	88.88±0.52	0.89±0.01	0.44
bi-lESN	66.31±1.95	0.66±0.02	0.95
bi-omESN	68.22±1.28	0.67±0.01	0.97
bi-rmESN	90.51±1.16	0.90±0.01	0.97
dr-lESN (ReLU)	52.48±1.84	0.51±0.02	0.65
dr-omESN (ReLU)	71.03±1.80	0.71±0.02	0.66
dr-rmESN (ReLU)	84.86±0.83	0.85±0.01	1.05
dr-lESN (Max)	53.04±1.68	0.52±0.02	0.67
dr-lESN (KAF)	46.73±1.97	0.46±0.02	0.74
dr-omESN (Max)	70.47±2.98	0.70±0.03	0.72
dr-omESN (KAF)	70.51±2.24	0.70±0.02	0.76
dr-rmESN (Max)	89.39±1.45	0.89±0.01	1.38
dr-rmESN (KAF)	86.54±1.48	0.86±0.02	1.16
LSTM	72.52±1.71	0.72±0.02	21.88
GRU	79.49±2.65	0.79±0.03	22.99
DTW-1NN	89.46	0.89	189.54

TABLE XV: Results on Robotic Arm Failure dataset.

RobotFail	Accuracy	F1 score	Time (mins)
lESN	50.00±2.80	0.49±0.03	0.01
omESN	59.69±1.82	0.58±0.02	0.01
rmESN	64.38±1.17	0.63±0.01	0.01
bi-lESN	51.56±2.61	0.51±0.03	0.02
bi-omESN	55.94±3.34	0.52±0.04	0.02
bi-rmESN	56.88±1.25	0.55±0.01	0.02
dr-lESN (ReLU)	49.38±4.15	0.48±0.04	0.12
dr-omESN (ReLU)	57.50±3.62	0.56±0.04	0.14
dr-rmESN (ReLU)	62.81±1.17	0.61±0.01	0.45
dr-lESN (Max)	53.75±2.54	0.52±0.02	0.14
dr-lESN (KAF)	53.44±6.20	0.52±0.06	0.18
dr-omESN (Max)	61.56±2.12	0.60±0.02	0.18
dr-omESN (KAF)	57.81±3.95	0.57±0.04	0.20
dr-rmESN (Max)	66.25±1.88	0.64±0.02	0.72
dr-rmESN (KAF)	63.75±1.17	0.63±0.01	0.50
LSTM	64.69±3.22	0.62±0.03	0.67
GRU	63.75±2.30	0.62±0.02	0.72
DTW-1NN	68.75	0.67	0.41

TABLE XVI: Results on Peformance Measurement System.

PEMS	Accuracy	F1 score	Time (mins)
lESN	49.83±5.30	0.49±0.05	0.20
omESN	71.68±1.51	0.72±0.01	0.30
rmESN	70.40±3.79	0.70±0.04	0.21
bi-lESN	63.70±2.14	0.63±0.03	0.49
bi-omESN	73.87±1.61	0.74±0.02	0.72
bi-rmESN	72.37±2.02	0.72±0.02	0.53
dr-lESN (ReLU)	64.05±3.13	0.64±0.03	0.50
dr-omESN (ReLU)	72.49±1.66	0.73±0.02	9.81
dr-rmESN (ReLU)	69.48±3.86	0.69±0.04	1.03
dr-lESN (Max)	68.55±1.30	0.68±0.01	0.55
dr-lESN (KAF)	69.36±3.08	0.69±0.03	0.64
dr-omESN (Max)	72.72±1.12	0.73±0.01	13.99
dr-omESN (KAF)	71.79±1.48	0.72±0.02	9.94
dr-rmESN (Max)	69.02±3.48	0.69±0.04	1.48
dr-rmESN (KAF)	68.67±2.77	0.69±0.03	1.20
LSTM	85.57±1.57	0.86±0.02	118.64
GRU	89.67±1.51	0.90±0.02	125.98
DTW-1NN	70.52	0.70	80.99

TABLE XVII: Results on CMUsubject16 dataset.

CMU	Accuracy	F1 score	Time (mins)
lESN	56.89±8.62	0.62±0.13	0.14
omESN	96.53±0.01	0.96±0.01	0.29
rmESN	98.27±1.72	0.98±0.01	0.23
bi-lESN	94.82±1.72	0.95±0.02	0.38
bi-omESN	89.65±0.01	0.91±0.01	0.37
bi-rmESN	94.83±1.72	0.95±0.02	0.39
dr-lESN (ReLU)	62.06±0.12	0.66±0.07	0.19
dr-omESN (ReLU)	96.55±0.01	0.96±0.01	0.14
dr-rmESN (ReLU)	98.27±1.72	0.98±0.01	0.30
dr-lESN (Max)	68.96±3.44	0.74±0.01	0.18
dr-omESN (Max)	96.55±0.01	0.96±0.01	0.23
dr-rmESN (Max)	100.0±0.00	1.00±0.00	3.31
dr-lESN (KAF)	62.06±0.13	0.67±0.01	0.23
dr-omESN (KAF)	93.11±3.44	0.93±0.03	0.30
dr-rmESN (KAF)	96.55±0.01	0.97±0.01	0.27
LSTM	55.17±13.79	0.54±0.12	8.22
GRU	54.79±9.52	0.71±0.08	8.13
DTW-1NN	100.0	1.0	3.31

TABLE XVIII: Results on Kick versus Punch dataset.

KICK	Accuracy	F1 score	Time (mins)
lESN	60.37±0.49	0.66±0.01	0.16
omESN	65.41±5.39	0.58±0.08	0.14
rmESN	75.02±4.83	0.73±0.06	0.17
bi-lESN	60.37±7.34	0.67±0.05	0.51
bi-omESN	85.73±5.13	0.85±0.05	0.33
bi-rmESN	100.0±0.00	1.0±0.00	0.37
dr-lESN (ReLU)	60.37±0.55	0.74±0.02	0.15
dr-omESN (ReLU)	40.18±0.08	0.51±0.41	0.14
dr-rmESN (ReLU)	60.94±0.01	0.75±0.01	0.11
dr-lESN (Max)	55.34±3.94	0.36±0.36	0.12
dr-omESN (Max)	55.34±3.94	0.36±0.36	0.17
dr-rmESN (Max)	50.23±1.98	0.37±0.37	0.22
dr-lESN (KAF)	50.23±1.98	0.37±0.37	0.15
dr-omESN (KAF)	50.66±1.74	0.37±0.37	0.14
dr-rmESN (KAF)	60.50±0.01	0.75±0.01	0.26
LSTM	45.67±5.46	0.33±0.33	0.54
GRU	35.83±5.97	0.38±0.05	0.52
DTW-1NN	70	0.66	1.08

TABLE XIX: Results on Walk versus Run dataset.

WALK	Accuracy	F1 score	Time (mins)
lESN	53.12±3.12	0.69±0.02	0.36
omESN	100.0±0.00	1.0±0.00	0.33
rmESN	100.0±0.00	1.0±0.00	0.34
bi-lESN	100.0±0.00	1.0±0.00	0.90
bi-omESN	100.0±0.00	1.0±0.00	0.87
bi-rmESN	100.0±0.00	1.0±0.00	0.83
dr-lESN (ReLU)	59.37±3.12	0.73±0.03	0.42
dr-omESN (ReLU)	100.0±0.00	1.0±0.00	0.43
dr-rmESN (ReLU)	100.0±0.00	1.0±0.00	0.44
dr-lESN (Max)	71.87±3.12	0.81±0.01	0.37
dr-omESN (Max)	100.0±0.00	1.0±0.00	0.45
dr-rmESN (Max)	100.0±0.00	1.0±0.00	0.48
dr-lESN (KAF)	65.62±3.12	0.77±0.02	0.40
dr-omESN (KAF)	96.87±3.12	0.98±0.02	0.45
dr-rmESN (KAF)	100.0±0.00	1.0±0.00	0.48
LSTM	75.57±0.13	0.85±0.07	25.28
GRU	75.57±0.13	0.85±0.07	25.58
DTW-1NN	100.0	1.0	5.48