Reservoir computing approaches for representation and classification of multivariate time series

Filippo Maria Bianchi^{a,*}, Simone Scardapane^b, Sigurd Løkse^a, Robert Jenssen^a

^aMachine Learning Group, UiT the Arctic University of Norway, Hansine Hansens veg 18, 9019 Tromsø, Norway.

^bDepartment of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, Via
Eudossiana 18, 00184 Rome, Italy

Abstract

Classification of multivariate time series (MTS) has been tackled with a large variety of methodologies and applied to a wide range of scenarios. Among the existing approaches, reservoir computing (RC) techniques. which implement a fixed and high-dimensional recurrent network to process sequential data, are computationally 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 other classifiers, such as those exploiting fully trainable recurrent networks. In this paper we introduce the reservoir model space, an RC approach to learn vectorial representations of MTS in an unsupervised fashion. Each MTS is encoded within the parameters of a linear model trained to predict a low-dimensional embedding of the reservoir dynamics. Our model space yields a powerful representation of the MTS and, thanks to an intermediate dimensionality reduction procedure, attains computational performance comparable to other RC methods. As a second contribution we propose a modular RC framework for MTS classification, with an associated open source Python library. By combining the different modules it is possible to seamlessly implement advanced RC architectures, including our proposed unsupervised representation, bidirectional reservoirs, and non-linear readouts, such as deep neural networks with both fixed and flexible activation functions. Several RC architectures implemented with the proposed framework are compared to other MTS classifiers, including state-of-the-art recurrent networks 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.

Keywords: Reservoir computing, model space, time series classification, recurrent neural networks

1. 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, 2, 3, 4], civil engineering [5], action recognition [6], and speech analysis [7, 8]. The problem has been tackled by a wealth of different approaches, spanning from the definition of tailored distance measures over MTS [9, 10, 11], to the identification and modeling of short-time patterns in the form of dictionaries or shapelets [12]. In this paper, we focus on MTS classifiers based on recurrent neural networks (RNNs), which are models specifically conceived to process sequential data and to capture dependencies in time [7, 13]. While there are several variations of RNNs, they all share the same fundamental principle: the MTS is first processed sequentially by a dynamic (possibly adaptable) model, and then the sequence of its internal states generated over time is exploited to perform classification [14].

^{*}Corresponding author.

Email addresses: filippo.m.bianchi@uit.no (Filippo Maria Bianchi), simone.scardapane@uniroma1.it (Simone Scardapane), sigurd.lokse@uit.no (Sigurd Løkse), robert.jenssen@uit.no (Robert Jenssen)

Reservoir computing (RC) is a family of RNN models whose recurrent part is kept fixed and is either generated randomly, or by means of custom topologies for facilitating the information flow [15, 16, 17, 18]. Despite this strong architectural 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 can achieve excellent performance in many fields, including time series forecasting [19, 20, 21, 22, 23], process modelling [17], and speech analysis [8]. Since the reservoir is fixed, one needs to train only the readout, which provides an instant mapping between the internal representation of the reservoir and the task-specific output [16, 18]. 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.

Several works have shown that RC models are a remarkably efficient solution for MTS classification [24, 25, 26, 27]. The MTS is fed into the reservoir that generates a sequence of states over time and, in the simplest case, the last reservoir state (or the mean of all the states) becomes a representation of the input. Afterwards, the representation is processed by a classification algorithm for vectorial data [24]. However, despite its unrivalled training speed, this approach fails to achieve the same accuracy of competing state-of-the-art classifiers [28].

To improve the classification performance, in [29] the reservoir states associated to a MTS are projected onto the principal components of the reservoir states, relative to each class; a new sample is classified by identifying the subspace that yields its lower reconstruction error, in mean square sense. To learn more powerful representations from the sequence of reservoir states, an alternative approach originally proposed in [30] and later applied to MTS classification and fault detection [31, 28, 32], advocates to map the inputs in a "model-based" feature space where they are represented by statistical properties that better characterize each class. In this formulation, for each MTS a model is trained to predict its next input from the current reservoir state, and the model parameters become the MTS representation. As a drawback, this approach accounts only for those reservoir dynamics useful to predict the next input and could neglect meaningful information for MTS characterization, hence hindering the quality of the representation. To overcome this limitation, in this paper we significantly extend model-space criterion by proposing a new approach that disentangles from the constraints imposed in the original formulation.

Contributions of the paper

Our main contribution is the design of a novel unsupervised procedure to generate representations of the input MTS that extends the model space criterion. The proposed representation, called "reservoir model space", consists in the parameters of the one-step-ahead predictor that estimates the future reservoir state, as opposed to the future MTS input [30]. The prediction model in this case must account for all the internal reservoir dynamics to solve the prediction task and, therefore, we argue it conveys a more accurate characterization of the input MTS. Due to the large size of the reservoir, a naïve formulation of the model space would yield extremely large representations that would 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), which keeps separated the modes of variation among time steps and data samples.

As a second contribution, we introduce a unified RC framework (with associated open source Python library) for MTS classification that generalizes both classic RC architectures and more advanced ones. Our framework consists of four independent 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 used to perform the final classification. We present multiple variants for implementing each module, such as a bidirectional reservoir and nonlinear readouts. When the proposed reservoir model space is used as the MTS representation, we analyze in details the interactions with the other modules.

In our experimental evaluation, we compare several RC architectures implemented with our framework, classifiers based on fully trainable RNNs, and other baseline approaches, such as SVM classifiers configured with pre-computed kernels for MTS. Experiments are performed on multiple experimental benchmarks and

a real-world dataset of medical MTS. We show that RC classifiers are dramatically faster than the other methods and, when implemented using our proposed representation, also achieve superior classification accuracy.

Structure of the paper

Sec. 2 introduces the problem of classification of MTS with RNN models and reviews modern fully-trainable architectures and the approaches based on RC. Sec. 3 describes the proposed reservoir model space and a dimensionality reduction procedure for compressing the sequences of reservoir activations. Sec. 4 presents the unified RC framework for MTS classification, describing advanced approaches to build the reservoir and to implement the readout. Sec. 5 performs an extensive experimental evaluation of the techniques, and Sec. 6 reports our conclusions.

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.

2. Classification and representation learning with recurrent neural networks

We consider classification of generic F-dimensional MTS observed for 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 $\mathbf{X} = [\mathbf{x}(1), \dots, \mathbf{x}(T)]^T$. The problem of assigning a class label \mathbf{y} represented with one-hot encoding to the sequence \mathbf{X} can be framed as a density estimation problem:

$$p(\mathbf{y}|\mathbf{x}(T),\mathbf{x}(T-1),\ldots,\mathbf{x}(1)). \tag{1}$$

A commonly adopted approach in machine learning is to build the sequence model as the combination of an *encoding* function and a *decoding* function. The encoder is used to generate a representation of the input, while the decoder is a discriminative (or predictive) model that computes the posterior probability of the output given the representation provided by the encoder. Among all possible choices for the encoding method, RNNs [33] are a type of artificial neural network particularly suitable to model sequential data. An RNN is governed by the following state-update equation

$$\mathbf{h}(t) = f\left(\mathbf{x}(t), \mathbf{h}(t-1); \theta_{\text{enc}}\right),\tag{2}$$

where $\mathbf{h}(t)$ is the internal state of the RNN 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 (usually a sigmoid or hyperbolic tangent), and θ_{enc} are the adaptable weights of the RNN. The simplest (vanilla) formulation reads:

$$\mathbf{h}(t) = f\left(\mathbf{W}_{\text{in}}\mathbf{x}(t) + \mathbf{W}_{\text{r}}\mathbf{h}(t-1)\right),\tag{3}$$

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

From the sequence of the RNN states generated over time, described by the matrix $\mathbf{H} = [\mathbf{h}(1), \dots, \mathbf{h}(T)]^T$, it is possible to define an encoding (representation) $r(\mathbf{H}) = \mathbf{r_X}$ of the input sequence \mathbf{X} . Rather than accounting for the whole sequence of RNN states, however, it is common to represent the MTS only with a subset of \mathbf{H} . A common choice is to take $\mathbf{r_X} = \mathbf{h}(T)$, i.e. discard all states except the last one. Thanks to the ability of capturing temporal dependencies, the RNN can embed into its last state all the information required to reconstruct the original input [34].

Such a state becomes a fixed-size vectorial representation of the MTS and can be processed by standard machine learning algorithms. Specifically, the decoder maps the input representation $\mathbf{r}_{\mathbf{X}}$ into the output space, which contains all class labels \mathbf{y} in a classification task:

$$\mathbf{y} = g(\mathbf{r}_{\mathbf{X}}; \theta_{\text{dec}}), \tag{4}$$

where θ_{dec} are trainable parameters. In practice, $g(\cdot)$ can implemented by another (feed-forward) neural network or by a simpler linear model.

In the following, we describe two principal approaches for MTS classification with RNNs. The first is based on fully trainable architectures, the second on RC where the encoding is implemented by a RNN that is left untrained.

2.1. Fully trainable RNNs and gated architectures

In fully trainable RNNs, the encoder parameters θ_{enc} and the decoder parameters θ_{dec} are jointly learned within a common training procedure. To this end, given a set of MTS $\{\mathbf{X}[n]\}_{n=1}^{N}$ and associated labels $\{\mathbf{y}[n]\}_{n=1}^{N}$, we can train the model by minimizing an empirical cost:¹

$$\theta_{\text{enc}}^*, \theta_{\text{dec}}^* = \underset{\theta_{\text{enc}}, \theta_{\text{dec}}}{\text{arg min}} \frac{1}{N} \sum_{n=1}^{N} l\left(\mathbf{y}[n], g\left(r(f(\mathbf{X}[n]))\right)\right), \tag{5}$$

where $l(\cdot, \cdot)$ is a generic loss function (e.g., cross-entropy over the labels). If the encoder has adaptable weights, their gradient of (5) with respect to θ_{enc} and θ_{dec} can be computed by back-propagation through time [14].

To regularize the parameters values during the training phase, a common procedure is to add an ℓ_2 norm penalty to both the weights in the encoding and decoding functions, $r(\cdot)$ and $g(\cdot)$. The penalty term is controlled by a scalar parameter λ . Furthermore, it is also possible to include a dropout regularization, that randomly drops connection weights with probability p_{drop} during training [35]. In our experiments, in the encoding function we apply a dropout specific for recurrent architectures [36].

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 [37, 38]. 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 common solution is the long short-term memory (LSTM) network [39]; differently from the vanilla RNNs in (3), LSTM 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) [40], that provides a better memory conservation by using less parameters than the LSTM, but its state update requires an additional operation, hence a higher computational cost. Both LSTM and GRU still require to back-propagate through time the gradient of their loss.

2.2. Reservoir computing and output model space representation

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 (3), but with the fundamental difference that the encoder parameters $\theta_{\rm enc} = \{ \mathbf{W}_{\rm in}, \mathbf{W}_{\rm r} \}$ are randomly generated and left untrained (or, possibly, implemented according to a prefixed topology [41]). To compensate this lack of adaptability, when processing the input sequence a large recurrent layer (reservoir) generates a rich pool of heterogeneous dynamics, from which the ones useful to solve many different tasks can be drawn. 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 $\mathbf{W}_{\rm r}$, set to bring the system to the edge of stability [42]. The behaviour of the reservoir can therefore be controlled by simply modifying the following structural hyperparameters instead of training the internal weight matrices: 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 $\mathbf{W}_{\rm in}$, which controls the amount of nonlinearity

¹Note that MTS may have different lengths. For readability, in the paper we refer with T to the length of a single MTS, implicitly assuming that T is a function of the MTS itself.

in the processing units and, jointly with ρ , can shift the internal dynamics from a chaotic to a contractive regime. Finally, a Gaussian noise with standard deviation ξ can be added to the argument of the state update function (3) for regularization purposes [15].

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

$$\mathbf{y} = g(\mathbf{r}_{\mathbf{X}}) = \mathbf{V}_o \mathbf{r}_{\mathbf{X}} + \mathbf{v}_o \tag{6}$$

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

$$\theta_{\text{dec}}^* = \underset{\{\mathbf{V}_o, \mathbf{v}_o\}}{\operatorname{arg\,min}} \frac{1}{2} \left\| \mathbf{r}_{\mathbf{X}} \mathbf{V}_o + \mathbf{v}_o - \mathbf{y} \right\|^2 + \lambda \left\| \mathbf{V}_o \right\|^2, \tag{7}$$

which admits a closed-form solution [18]. The combination of an untrained reservoir and a linear readout defines the basic ESN model [15].

A powerful representation $\mathbf{r_X}$ is the *output model space* representation [30], obtained by first processing each MTS with a common reservoir and then fitting a linear model (one for each MTS), whose parameters become the MTS representation. Specifically, a ridge regression model is trained to implement an output function that performs one step-ahead prediction of each input MTS:

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

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 (6). In the following, we propose an alternative model space that yields a more expressive representation of the input MTS.

3. Proposed reservoir model space representation

In this section we introduce the main contribution of this paper, the *reservoir model space* for representing MTS. To make the model tractable, we reduce the dimensionality of reservoir features by means of a modified version of PCA, which deals with data represented as matrices rather than vectors.

3.1. Formulation of the reservoir model space

The great generalization capability of the reservoir is grounded on the large amount of different dynamical features it generates from the input time series. Indeed, the readout selects and combines the ones which are useful to accomplish a specific task. In the case of prediction, different combinations of these dynamics are accounted depending on the forecast horizon of interest. Therefore, by fixing a specific prediction step (e.g., 1 step-ahead) we implicitly ignore all those dynamics that are not particularly useful to solve the task. We argue that this potentially introduces a bias in the model space induced by output prediction, since some dynamical features that are not important for the prediction task can still be useful to characterize the MTS.

Therefore, we propose a variation of the output model space presented in Sec. 2.2, where each MTS is represented as the parameters of a linear model that predicts the next reservoir state, rather than the output sample. The prediction model in this case must account for all the dynamics in the reservoir to solve its task and we argue that provides a more accurate characterization of the input MTS. More formally, the following linear model is trained to implement the prediction of the next reservoir state

$$\mathbf{h}(t+1) = \mathbf{U}_h \mathbf{h}(t) + \mathbf{u}_h, \tag{9}$$

and, thus, $\mathbf{r_X} = \theta_h = [\text{vec}(\mathbf{U}_h); \mathbf{u}_h] \in \mathbb{R}^{R(R+1)}$ is our proposed representation of the input MTS.

Contrarily to the output model space representation, the prediction model here describes a generative model of the reservoir sequence, rather than of the input sequence. The learned feature space can capture both the data and their generative process. As for the output model space, a classifier that processes the reservoir model representation combines the explanatory capability of generative models with the classification power of the discriminative methods.

It is possible to see that $\mathbf{r}_{\mathbf{X}}$ characterize the following discriminative model

$$p\left(\mathbf{x}(t+1)|\mathbf{h}(t);\mathbf{r}_{\mathbf{X}}\right). \tag{10}$$

At the same time, the reservoir encoding that operates according to (3) can be expressed as

$$p\left(\mathbf{h}(t)|\mathbf{h}(t-1),\mathbf{x}(t)\right). \tag{11}$$

The state $\mathbf{h}(t-1)$ in (11) depends, in turn, on $\mathbf{x}(t-1)$ and $\mathbf{h}(t-2)$, and so on. By expressing those dependencies explicitly with the chain rule of probability and then plugging (11) in (10) one obtains

$$\prod_{t=1}^{T} p\left(\mathbf{x}(t+1)|\mathbf{x}(t),\mathbf{x}(t-1),\ldots,\mathbf{x}(1);\mathbf{r}_{\mathbf{X}}\right) = p\left(\mathbf{x}(T),\mathbf{x}(T-1),\ldots,\mathbf{x}(1);\mathbf{r}_{\mathbf{X}}\right),$$
(12)

Eq. (12) is a generative model [43], which provides a powerful characterization of the inputs and also induces a metric relationship between samples [44]. As a result, classification with the model space criterion can be categorized as a hybrid discriminative/generative method, and the representation $\mathbf{r_X}$ (learnt unsupervisedly) can be exploited in a variety of different tasks. Indeed, the model view characterization of a MTS has proven effective in anomaly detection [31], classification [28, 32], and to build a kernel similarity matrix [30].

3.2. Dimensionality reduction for reservoir states tensor

Due to the high dimensionality of the reservoir, the number of parameters of the prediction model in (9) would grow too large, making the reservoir model space representation intractable. Besides the undesired effects of producing a representation of the input in a high dimensional space without enforcing sparsity constraints, evaluating a ridge regression solution for each MTS would demand many computational resources, which is against our goal to design an efficient classifier.

In the context of RC, applying PCA to reduce dimensionality of the last reservoir state has shown to improve performance in the inference task [45, 46]. However, our proposed MTS representation no longer coincides with the last reservoir state, but derives from the whole sequence of states generated over time. In this case, standard PCA is unsuitable as it can be applied only to unidimensional data. To address this issue, hereinafter we introduce a dimensionality reduction procedure that extends PCA to the multidimensional case and keeps separated the modes of variation across time in the state sequences pertaining different samples.

We conveniently describe our dataset as a 3-mode tensor $\mathcal{H} \in \mathbb{R}^{N \times T \times R}$ and require a procedure to map $R \to D$ s.t. $D \ll R$, while maintaining the other dimensions unaltered. We note that for MTS of varying length, zero-padding (or a more elaborate interpolation procedure) is required to build the tensor. Dimensionality reduction on high-order tensors can be achieved through Tucker decomposition [47], which decomposes a tensor into a core tensor (the lower-dimensional representation) multiplied by a matrix along each mode. When only one dimension of \mathcal{H} is modified, Tucker decomposition becomes equivalent to applying a two-dimensional PCA on a specific matricization of \mathcal{H} [48]. In particular, to reduce the third dimension (R) one computes the mode-3 matricization of \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

$$\mathbf{C} = \frac{1}{NT - 1} \sum_{i=1}^{NT} \left(\mathbf{h}_i - \bar{\mathbf{h}} \right) \left(\mathbf{h}_i - \bar{\mathbf{h}} \right)^T.$$
 (13)

In (13), \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 \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

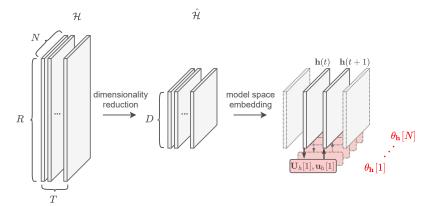


Figure 1: Schematic depiction of the procedure to generate the reservoir model space representation. For each input MTS $\mathbf{X}[n]$ a sequence of states $\mathbf{H}[n]$ is generated by a fixed reservoir. Those are the frontal slices (dimension N) of \mathcal{H} , but notice that in the figure lateral slices (dimension T) are shown. By means of dimensionality reduction, the reservoir features from R become D. A distinct linear model is trained to predict the columns of each frontal slice $\hat{\mathbf{H}}$ of $\hat{\mathcal{H}}$. The parameters $\theta_h[n]$ of the model trained on $\hat{\mathbf{H}}[n]$ become the representation of $\mathbf{X}[n]$.

orderings are lost, as reservoir states relative to different samples and generated in different time steps are mixed together. This may lead to a potential loss in the representation capability, as the existence of modes of variation in time courses within individual samples is ignored [49]. To address this issue, we consider as individual samples the matrices $\mathbf{H}_n \in \mathbb{R}^{T \times H}$, obtained by slicing \mathcal{H} across its first dimension. The sample covariance matrix in this case reads

$$\mathbf{S} = \frac{1}{N-1} \sum_{n=1}^{N} (\mathbf{H}_n - \bar{\mathbf{H}})^T (\mathbf{H}_n - \bar{\mathbf{H}}).$$
 (14)

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 $\hat{\mathcal{H}} = \mathcal{H} \times_3 \mathbf{E}$, 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, the whole time sequence of reservoir states generated by a MTS is considered as an observation. Accordingly, states pertaining to different MTS are grouped together and their temporal ordering is preserved. We notice that an analogous approach was adopted to reduce the dimensionality of images [50], which are characterized by multiple spatial dimensions, while the dimensions in our samples represent space and time, respectively.

After dimensionality reduction, the model in (8) becomes

$$\hat{\mathbf{h}}(t+1) = \mathbf{U}_h \hat{\mathbf{h}}(t) + \mathbf{u}_h, \tag{15}$$

where $\hat{\mathbf{h}}(\cdot)$ are the columns of a frontal slice $\hat{\mathbf{H}}$ of $\hat{\mathbf{H}}$, $\mathbf{U}_h \in \mathbb{R}^{D \times D}$, and $\mathbf{b}_h \in \mathbb{R}^D$. The representation will now coincide with the following vector of parameters $\mathbf{r}_{\mathbf{X}} = \theta_h = [\text{vec}(\mathbf{U}_h); \mathbf{u}_h] \in \mathbb{R}^{D(D+1)}$, whose dimensionality is controlled by the described dimensionality reduction procedure. A schematic description of the proposed unsupervised procedure to derive the reservoir model representation is shown in Fig. 1

4. A unified reservoir computing framework for time series classification

Apart from the model based criterion, in the last years several additional approaches have been proposed in the literature of RC for extending the basic ESN architecture. Most works focused on the design of more sophisticated reservoirs, readouts or representation of the input MTS and they can be considered independently one from the other. To evaluate their efficacy in the context of MTS classification, we introduce

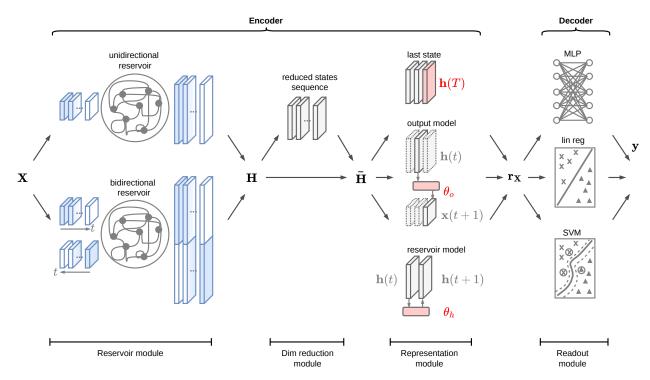


Figure 2: Overview of the RC framework for MTS classification. The ecoder generates a representation $\mathbf{r_X}$ of the MTS \mathbf{X} , while the decoder assign a label \mathbf{y} , given the representation. Several models are obtained by selecting variants for each module.

a unified framework for MTS classification 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.

A complete overview of all the combinations considered in this paper, including the proposed reservoir model space representation, is given in Fig. 2. The input MTS \mathbf{X} is processed by a reservoir, which is either unidirectional or bidirectional, and it generates the sequence of states \mathbf{H} over time. An optional dimensionality reduction step can be applied to reduce the number of reservoir features, and a new sequence $\bar{\mathbf{H}}$ is obtained. Three different approaches can be chosen to generate a representation $\mathbf{r}_{\mathbf{X}}$ of the input from the sequence of reservoir features: the last state of the sequence $\mathbf{h}(T)$, the output state model θ_o (Sec. 2.2), or the proposed reservoir state model θ_h (Sec. 3). The obtained representation $\mathbf{r}_{\mathbf{X}}$ is finally processed by a decoder (readout), which is implemented by a classifier for real-valued vectors. We consider three different classifiers, which are the standard linear readout implemented by ridge regression, a SVM classifier and a deep neural network (MLP). The classifier outputs the class label \mathbf{y} to be assigned to the MTS \mathbf{X} .

In the following, we describe the reservoir, dimensionality reduction and readout module, and we discuss the functionality of the variants implemented in our framework. A Python software library implementing the unified framework is publicly available online². The library is regularly updated and several other options are available to configure the classifier, beside the ones described in this paper.

4.1. Reservoir module

Several approaches have been proposed to extend the ESN reservoir to implement additional features, such as the capability of handling multiple time scales [51], or to simplify its large and randomized structure [52].

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

Of particular interest for the classification of MTS is the bidirectional reservoir, which is included in our framework as an optional replacement to the standard reservoir.

Bidirectional architectures have been successfully applied in RNNs to extract from the input sequences temporal features that account also for dependencies very far in time [7]. Within the RC framework, a bidirectional reservoir has been used in the context of time series prediction to collect future information, which is provided at training stage to improve the accuracy of the model [17]. The future information is not available during test and the model uses a standard reservoir to extract temporal features. On the other hand, in a classification setting the whole time series is given at once, in both training and test stages. In [46] the last state of a bidirectional reservoir is used as a fixed-size representation for classifying the input MTS.

Bidirectionality is implemented by feeding into the *same* reservoir an input sequence both in straight and reverse order, so that the following states are generated

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

$$\vec{\mathbf{h}}(t) = f\left(\mathbf{W}_{\text{in}}\vec{\mathbf{x}}(t) + \mathbf{W}_{\text{r}}\vec{\mathbf{h}}(t-1)\right),$$
(16)

where $\mathbf{\bar{x}}(t) = \mathbf{x}(T-t)$. The final states sequence of the bidirectional reservoir is obtained by concatenating at each time interval the backward and forward states $\mathbf{h}^b(t) = \left[\mathbf{\bar{h}}(t); \mathbf{\bar{h}}(t)\right] \in \mathbb{R}^{2R}$.

The representation of the input as the last state generated by the reservoir in this case would become $\mathbf{r_X} = \mathbf{h}^b(T)$. The main advantage of using a bidirectional reservoir in conjunction with the last state representation is that, compared to unidirectional architectures, temporal dependencies spanning longer time intervals can be captured. Since the reservoir trades its internal stability with a vanishing memory of the past inputs [42], at time T the state $\dot{\mathbf{h}}(T)$ maintains scarce information about the first inputs processed. On the other hand, $\dot{\mathbf{h}}(T)$ is more influenced by the first inputs and, therefore, $\mathbf{h}^b(T)$ summarizes well recent and past information. This representation improves classification results especially when important information are contained also in the first part of the input.

In the proposed reservoir model space formulation (see Sec. 3) a bidirectional reservoir modifies the representation $\mathbf{r}_{\mathbf{X}}$, as the linear model in (9) becomes

$$\left[\mathbf{h}(t+1); \tilde{\mathbf{h}}(t+1)\right] = \mathbf{U}_h^b \left[\tilde{\mathbf{h}}(t); \tilde{\mathbf{h}}(t)\right] + \mathbf{u}_h^b, \tag{17}$$

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 two distinct objectives, which are predicting the next state $\mathbf{h}(t+1)$ and reproducing the previous one $\mathbf{h}(t+1)$ (or equivalently their low-dimensional embeddings). Therefore, by combining the reservoir model space with a bidirectional reservoir, the prediction model must learn also a memorization task, as it yields at each time step both the previous and past reservoir state. We argue that such a model provides a more accurate representation of the input, by modeling at the same time the temporal dependencies in both time directions. We notice that the bidirectional reservoir produces a similar effect also in the output model space (see Sec. 2.2), where the linear model learns jointly to reproduce the previous input and to predict the next one.

4.2. Dimensionality reduction module

The dimensionality reduction module projects the sequence of reservoir activations on a lower dimensional subspace, using unsupervised criteria. Since the reservoir is characterized by a large number of neurons, dimensionality reduction applied on top of $\mathbf{h}(t)$ yields a more compact representation, which can provide a regularization to the model that enhances its generalization capability and simplifies its training [53]. 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 [54]. When dealing with a prediction task, dimensionality reduction is applied to a single sequence of reservoir states generated by the input MTS [45]. On the other hand, in a classification task each MTS is associated to a different sequence of

states [46]. If the MTS are represented by the last reservoir states, those are stacked into a matrix to which standard dimensionality reduction procedures were applied. However, as discussed in Sec. 3, when the whole set of representations is represented by a tensor 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. 3, dimensionality reduction is particularly important when implementing the proposed reservoir model space representation, to lower the computational complexity and reduce the risk of overfitting. Furthermore, dimensionality reduction becomes fundamental when using a bidirectional reservoir, as the state dimension is doubled and, accordingly, also the size of the representation increases. In particular, when the last state is used as representation its size is doubled, $\mathbf{r_X} \in \mathbb{R}^{2R}$. In the output and reservoir space representations instead, the size becomes $\mathbf{r_X} \in \mathbb{R}^{F(2R+1)}$ and $\mathbf{r_X} \in \mathbb{R}^{2R(2R+1)}$, respectively.

4.3. Readout module

The readout module classifies the representations by means of a linear model, a SVM or a neural network. In a standard ESN, the output layer is a linear readout that is quickly trained by solving a convex optimization problem. However, a simple linear model might not possess sufficient representational power for modeling the high-level embeddings derived from the reservoir states. For this reason, several authors proposed to replace the standard linear decoding function $g(\cdot)$ in (6) with a nonlinear model, such as support vector machines (SVMs) [20, 22] or MLPs [55, 56, 57].

In particular, MLP is an universal function approximator that can learn complex representations of the input by stacking multiple layers of neurons configured with non-linear activations, e.g., rectified linear units (ReLUs). Deep MLPs are known for their capability of disentangling factors of variations from high-dimensional spaces [58], and therefore can be more powerful and expressive in their instantaneous mappings from the representation to the output space than linear readouts. When paired with a RNN, the number of layers in the MLP determines the "feedforward" depth in the RNN [13].

Readouts implemented as MLPs accomplished only modest results in the earliest works on RC [16]. However, nowadays MLPs can be trained much more efficiently by means of sophisticated initialization procedures [59] and regularization techniques [35]. Indeed, the combination of ESNs with MLPs trained with modern techniques can provide a substantial gain in performance as compared to a linear formulation [46]. Following recent trends in the deep learning literature we also investigate endowing the deep readout with more expressive flexible nonlinear activation functions, namely Maxout [60] and kernel activation functions [61].

5. Experiments

In this section 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, and SVM classifiers configured with pre-computed kernels for MTS. Depending whether the input MTS in the RC-based model is represented by the last reservoir state ($\mathbf{r_X} = \mathbf{h}(T)$), or by the output space model (Sec. 2.2), or by the reservoir space model (Sec. 3), we refer to the models as lESN, omESN and rmESN, respectively. Whenever we use a bidirectional reservoir, a deep readout or a SVM readout we add the prefix "bi-", "dr-", and "svm-", respectively (e.g., bi-lESN or dr-bi-rmESN).

First, we introduce the MTS classification datasets under analysis and we explain our experimental setup. In Sec. 5.1, we compare the performance obtained on several benchmark datasets by RC classifiers configured with different representations, by classifiers based on fully trained RNNs, and by 1-NN classifier using DTW. In Sec. 5.2, we investigate whether performance in the RC-based models improves when the representations are generated with a bidirectional reservoir and/or processed with deep readouts. Finally, in Sec. 5.3 we process a real-world dataset and we further compare the classification performance with two time series kernels, using a SVM classifier.

Benchmark datasets. To provide an extensive evaluation of the performance of each classifier, we consider several benchmark classification datasets for MTS taken from the UCR³ and UCI repositories⁴. We excluded from the analysis only those MTS datasets that contain too few training samples to train a neural network model. We also included three univariate time series datasets, to show how the proposed approaches can be seamlessly applied also to the univariate case. Details of the datasets are reported in Tab. 1.

Table 1: 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.

Dataset	#V	Train	Test	#C	T_{min}	T_{max}	Source
Swedish Leaf	1	500	625	15	128	128	UCR
Chlorine Concentration	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	[62]
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
PEMS	963	267	173	7	144	144	UCI

Blood samples dataset. As real-world case study, 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 2004–2012.⁵ 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.

Experimental setup. For each dataset, we train the models 10 times using independent random parameters initializations and each model is configured with the same hyperparameters in all experiments. Reservoirs are sensitive to hyperparameter setting and, therefore, fine-tuning with independent validation procedures for each task is usually more important in RC models than in networks trained with gradient descent, such as LSTM and GRU. Nevertheless, we show that even by setting the hyperparameters according to unsupervised heuristics [63] the RC classifiers are robust and achieve superior performance, especially when configured with the proposed representation (rmESN).

To provide a significant comparison, *IESN*, *omESN* and *rmESN* always share the same randomly generated reservoir configured with the following hyperparameters: internal units R = 800; spectral radius $\rho = 0.99$; non-zero connections percentage $\beta = 0.25$; input scaling $\omega = 0.15$; noise level $\xi = 0.01$. When classification is performed with a ridge regression readout, we set the regularization $\lambda = 1.0$. The ridge

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

 $^{^4}$ archive.ics.uci.edu/ml/datasets.html

 $^{^5\}mathrm{The}$ dataset has been published in the AMIA Data Competition 2016

regression prediction models, used to generate the model-space representation in omESN and rmESN, are instead configured with $\lambda = 5.0$.

In each RC-based method, we apply dimensionality reduction, as it provides important computational advantages (in terms of both memory and CPU time), as well as a regularization that improves the overall generalization capability and robustness of the models. To determine the optimal number of subspace dimensions D, we evaluate how training time and average classification accuracy (computed with a k-fold cross-validation procedure) of the RC classifiers varies on the benchmark dataset in Tab. 1. We report the average results in Fig. 3. While the training time increases approximately linearly with D, it is possible to

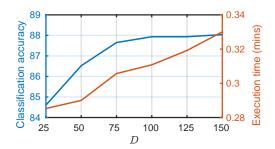


Figure 3: Classification accuracy and execution time for different dimensions D of the space with reduced dimensionality.

identify an "elbow" in the classification accuracy for D = 75, which is the value we select.

LSTM and GRU are configured with H=30 hidden units; the decoding function is implemented as a neural network with a dense layer of 20 hidden units followed by a softmax layer; the dropout probability is $p_{\rm drop}=0.1$; the ℓ_2 regularization parameter is $\lambda=0.0001$; gradient descent is performed with the Adam algorithm [64] and we train the models for 5000 epochs Finally, the 1-NN classifier uses FastDTW [9], implemented with the related Python library⁶.

5.1. 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 based on DTW. Evaluation is performed on the benchmark datasets in Tab. 1. The decoder is implemented by linear regression in the RC models and by a non-linear function in LSTM and GRU. Since all the other parameters in LSTM and GRU are learned with a non linear optimization technique, the non-linearities in the decoding function do not result in additional computational costs. Results are reported in Fig. 4. The first panel reports the mean classification accuracy and standard deviation from 10 independent runs on all benchmark datasets, while the second panel shows the average training time of each model in minutes on a logarithmic scale.

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, yielding consistently good results regardless of the random reservoir. The second-best accuracy is 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 RNN-based competitors, despite it relies on a linear classifier and the representations are learned in a complete unsupervised fashion.

In terms of execution time, the RC classifiers are much faster than the competitors, as the average time for train 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, hence demonstrating how it can attain the best performance without compromising the speed. On the other hand, the classifiers based on fully

⁶https://pypi.org/project/fastdtw/

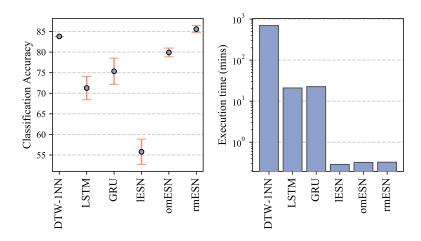


Figure 4: Comparison of the average results obtained on all benchmark datasets.

trainable RNNs, LSTM and GRU, require in average more than 20 minutes for training and processing the test data. Finally, 1-NN with DTW is much slower than the other methods despite the adopted "fast" DTW implementation [9]. 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 cases (see the supplementary material for the details).

5.2. 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 training 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 [60] and kernel activation functions (KAFs) [61]. Thanks to their adaptable parameters, trained jointly with the other MLP weights, these functions can learn more complicated relationships and improve the expressive capability of the model at every layer. 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 $p_{\rm drop} = 0.1$ and ℓ_2 regularization parameter $\lambda = 0.001$.

We repeat the models evaluation on the all the benchmark datasets and in Fig. 5 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. Indeed, the representation $\mathbf{r_X}$ provided by this model is the last state, which 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 slightly 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 we learn a model that also solves a memorization task and its parameters further characterize the input. However, the performance improvement for these model is limited, probably because the representations obtained with a unidirectional reservoir are already good enough.

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 at the end of 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 standard ReLUs,

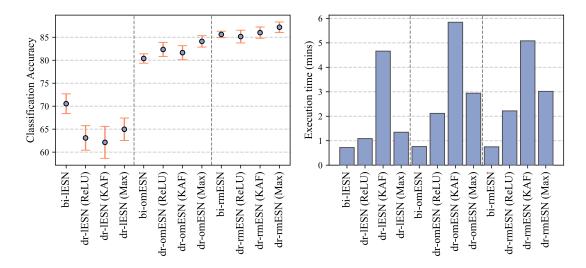


Figure 5: Classification accuracy and execution time when using RC classifiers with a bidirectional reservoir and deep readouts, configured with ReLUs, KAFs, and Maxout activations.

the training time is slightly higher, but we obtain significant improvements in the average classification accuracy. In particular, dr-omESN (Maxout) obtains almost the same performance of the basic version of rmESN, while dr-rmESN (Maxout) reaches the highest overall accuracy. 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. 6 we report the overall ranking, in terms of mean accuracy, of the 18 MTS classifier presented so far on the 14 benchmark datasets. On each dataset, 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 and bidirectional reservoir, performance can be further improved.

In the supplementary material we report the details of the aggregated results provided so far, along with a statistical test to asses the significance of the differences in the results obtained by the algorithms on multiple datasets.

5.3. Classification of blood samples MTS

As last experiment, we analyze the blood sample MTS and we evaluate the RC classifiers configured with a SVM readout. We consider only omESN and rmESN since, as demonstrated previously, they provide an optimal compromise between training efficiency and classification accuracy. As we focus on a kernel method to implement the decoding function (4), we compare the RC classifiers with two state-of-the-art kernels for MTS. The first is the learned pattern similarity (LPS) [11], 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) [10], which is based on an ensemble learning procedure wherein the clustering results of several Gaussian mixture models, 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 pre-computed 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

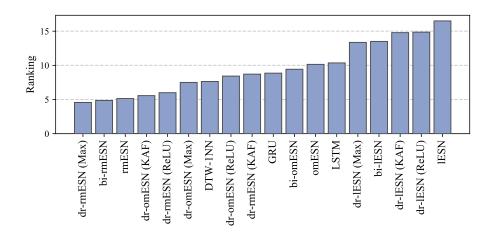


Figure 6: Ranking in terms of mean accuracy obtained by the MTS classifiers on 14 benchmark datasets. A lower value in ranking indicates better average accuracy.

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 are depicted in Fig. 7. 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. Nevertheless, 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. TCK yields results with the lowest standard deviation and is followed by the two versions of rmESN. The SVM readout increases the training time of the RC models, especially rmESN, but is still much lower than computing the TCK and LPS kernels.

6. Conclusions and future work

In this work we investigated several alternatives to build a classifier based on reservoir computing, focusing on unsupervised procedures to learn fixed-size representations of the input time series. As main contribution, 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 high-level dynamical features of the inputs. 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 [65]. However, in our case the reservoir is flexible and generic, as it can express a large variety of dynamical features of the input. Therefore, the reservoir captures relationships among the data, without posing constraints on the particular model underlying the data distribution. This provides two advantages: (i) different discriminative models can be

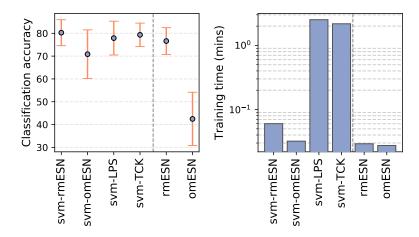


Figure 7: Classification accuracy obtained with SVM using different precomputed kernels. We also report the results obtained by rmESN and omESN on the same problem.

used in conjunction with the same reservoir model space representation and (ii) the same reservoir can be used to model data generated by different distributions.

To make the reservoir model space tractable we applied an unsupervised dimensionality reduction procedure, suitable for datasets represented as high-order tensors. Such dimensionality reduction greatly reduces computational time and memory usage, and provides a regularization that prevents overfitting, especially in complex discriminative classifiers.

We considered several benchmarks datasets for classification of multivariate time series, 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 proposed representation with these more sophisticated architectures provides only a minimal contribution to the accuracy, pointing to the strong informative content of this representation in terms of discriminative power. We concluded by considering a real-world case study of time series pertaining to blood samples and we compared our method with state-of-the-art kernels for multivariate time series. Even in this case, the features extracted by our reservoir model space criterion are fast to compute and highly informative, as they yield superior classification accuracy.

Acknowledgments

This work was partially funded by the Norwegian Research Council FRIPRO grant no. 239844 on developing the Next Generation Learning Machines. The authors would like to thank Arthur Revhaug, Rolv-Ole Lindsetmo and Knut Magne Augestad, all clinicians currently or formerly affiliated with the gastrointestinal surgery department at the University Hospital of North Norway, for preparing the blood samples dataset. The authors would also like to acknowledge Karl Øyvind Mikalsen, UiT, and Cristina Soguero-Ruiz, University of Rey Juan Carlos, for discussions regarding this data set.

References

References

[1] A. Kampouraki, G. Manis, C. Nikou, Heartbeat time series classification with support vector machines, IEEE Transactions on Information Technology in Biomedicine 13 (4) (2009) 512–518.

- [2] P. Buteneers, D. Verstraeten, B. Van Nieuwenhuyse, D. Stroobandt, R. Raedt, K. Vonck, P. Boon, B. Schrauwen, Real-time detection of epileptic seizures in animal models using reservoir computing, Epilepsy Research 103 (2-3) (2013) 124–134.
- [3] G. Clifford, C. Liu, B. Moody, L. Lehman, I. Silva, Q. Li, A. E. Johnson, R. G. Mark, AF classification from a short single lead ECG recording: The Physionet computing in cardiology challenge 2017, Computing in Cardiology 44 (2017) 1–4.
- [4] K. Ø. Mikalsen, F. M. Bianchi, C. Soguero-Ruiz, S. O. Skrøvseth, R.-O. Lindsetmo, A. Revhaug, 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.
- [5] E. Carden, J. Brownjohn, Arma modelled time-series classification for structural health monitoring of civil infrastructure, Mechanical Systems and Signal Processing 22 (2) (2008) 295–314.
- [6] D. Hunt, D. Parry, Using echo state networks to classify unscripted, real-world punctual activity, in: Engineering Applications of Neural Networks, Springer, 2015, pp. 369–378.
- [7] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Networks 18 (5-6) (2005) 602–610.
- [8] E. Trentin, S. Scherer, F. Schwenker, Emotion recognition from speech signals via a probabilistic echostate network, Pattern Recognition Letters 66 (2015) 4–12.
- [9] S. Salvador, P. Chan, Toward accurate dynamic time warping in linear time and space, Intelligent Data Analysis 11 (5) (2007) 561–580.
- [10] K. Ø. Mikalsen, F. M. Bianchi, C. Soguero-Ruiz, R. Jenssen, Time series cluster kernel for learning similarities between multivariate time series with missing data, Pattern Recognition 76 (2018) 569 – 581.
- [11] M. G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Mining and Knowledge Discovery 30 (2) (2016) 476–509.
- [12] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Mining and Knowledge Discovery 31 (3) (2017) 606–660.
- [13] R. Pascanu, C. Gulcehre, K. Cho, Y. Bengio, How to construct deep recurrent neural networks, in: Proc. 2017 International Conference on Learning Representations (ICLR), 2014.
- [14] A. Graves, A.-r. Mohamed, 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.
- [15] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note, GMD Technical Report 148 (34).
- [16] M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Computer Science Review 3 (3) (2009) 127–149.
- [17] A. Rodan, A. Sheta, H. Faris, Bidirectional reservoir networks trained using SVM+ privileged information for manufacturing process modeling, Soft Computing 21 (22) (2017) 6811–6824.
- [18] S. Scardapane, D. Wang, Randomness in neural networks: an overview, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 7 (2).

- [19] Z. Shi, M. Han, Support vector echo-state machine for chaotic time-series prediction, IEEE Transactions on Neural Networks 18 (2) (2007) 359–372.
- [20] D. Li, M. Han, J. Wang, Chaotic time series prediction based on a novel robust echo state network, IEEE Transactions on Neural Networks and Learning Systems 23 (5) (2012) 787–799.
- [21] A. Deihimi, H. Showkati, Application of echo state networks in short-term electric load forecasting, Energy 39 (1) (2012) 327–340.
- [22] F. Bianchi, S. Scardapane, A. Uncini, A. Rizzi, A. Sadeghian, Prediction of telephone calls load using Echo State Network with exogenous variables, Neural Networks 71 (2015) 204–213.
- [23] F. M. Bianchi, E. De Santis, A. Rizzi, A. Sadeghian, Short-term electric load forecasting using echo state networks and PCA decomposition, IEEE Access 3 (2015) 1931–1943.
- [24] M. Skowronski, 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.
- [25] Q. Ma, L. Shen, W. Chen, J. Wang, J. Wei, Z. Yu, Functional echo state network for time series classification, Information Sciences 373 (2016) 1–20.
- [26] F. Palumbo, C. Gallicchio, R. Pucci, A. Micheli, Human activity recognition using multisensor data fusion based on reservoir computing, Journal of Ambient Intelligence and Smart Environments 8 (2) (2016) 87–107.
- [27] P. Tanisaro, G. Heidemann, Time series classification using time warping invariant echo state networks, in: Proc. 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2016, pp. 831–836.
- [28] W. Aswolinskiy, R. Reinhart, 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.
- [29] A. Prater, Spatiotemporal signal classification via principal components of reservoir states, Neural Networks 91 (2017) 66 75. doi:10.1016/j.neunet.2017.04.008.
- [30] H. Chen, F. Tang, P. Tino, 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.
- [31] H. Chen, P. Tino, A. Rodan, X. Yao, Learning in the model space for cognitive fault diagnosis, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 124–136.
- [32] W. Aswolinskiy, R. Reinhart, J. Steil, Time series classification in reservoir-and model-space, Neural Processing Letters (2017) 1–21.
- [33] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, R. Jenssen, Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis, Springer, 2017.
- [34] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.
- [35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

- [36] W. Zaremba, I. Sutskever, O. Vinyals, Recurrent neural network regularization, arXiv preprint arXiv:1409.2329.
- [37] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks 5 (2) (1994) 157–166.
- [38] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International Conference on Machine Learning, 2013, pp. 1310–1318.
- [39] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.
- [40] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078.
- [41] A. Rodan, P. Tino, Minimum complexity echo state network, IEEE Transactions on Neural Networks 22 (1) (2011) 131–144.
- [42] F. M. Bianchi, L. Livi, C. Alippi, Investigating echo-state networks dynamics by means of recurrence analysis, IEEE Transactions on Neural Networks and Learning Systems 29 (2) (2018) 427–439.
- [43] A. Y. Ng, 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.
- [44] T. Jaakkola, D. Haussler, Exploiting generative models in discriminative classifiers, in: Advances in neural information processing systems, 1999, pp. 487–493.
- [45] S. Løkse, F. M. Bianchi, R. Jenssen, Training echo state networks with regularization through dimensionality reduction, Cognitive Computation 9 (3) (2017) 364–378.
- [46] F. M. Bianchi, S. Scardapane, S. Løkse, R. Jenssen, Bidirectional deep-readout echo state networks, in: European Symposium on Artificial Neural Networks, 2018.
- [47] L. R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.
- [48] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM review 51 (3) (2009) 455–500.
- [49] C. F. Beckmann, S. M. Smith, Tensorial extensions of independent component analysis for multisubject fmri analysis, Neuroimage 25 (1) (2005) 294–311.
- [50] D. Zhang, Z.-H. Zhou, (2d) 2pca: Two-directional two-dimensional pca for efficient face representation and recognition, Neurocomputing 69 (1-3) (2005) 224–231.
- [51] C. Gallicchio, A. Micheli, Echo state property of deep reservoir computing networks, Cognitive Computation 9 (3) (2017) 337–350. doi:10.1007/s12559-017-9461-9.
- [52] A. Rodan, P. Tino, Simple deterministically constructed cycle reservoirs with regular jumps, Neural Computation 24 (7) (2012) 1822–1852, pMID: 22428595. doi:10.1162/NECO_a_00297.
- [53] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural computation 15 (6) (2003) 1373–1396.
- [54] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, G. Rätsch, Kernel pca and de-noising in feature spaces, in: Advances in neural information processing systems, 1999, pp. 536–542.
- [55] W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural computation 14 (11) (2002) 2531–2560.

- [56] K. Bush, 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.
- [57] Š. Babinec, 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.
- [58] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, A. Y. Ng, Measuring invariances in deep networks, in: Advances in neural information processing systems, 2009, pp. 646–654.
- [59] X. Glorot, 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.
- [60] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, Y. Bengio, Maxout networks., Proc. 30th International Conference on Machine Learning (ICML).
- [61] S. Scardapane, S. Van Vaerenbergh, S. Totaro, A. Uncini, Kafnets: kernel-based non-parametric activation functions for neural networks, arXiv preprint arXiv:1707.04035.
- [62] Y. C. Sübakan, B. Kurt, A. T. Cemgil, B. Sankur, Probabilistic sequence clustering with spectral learning, Digital Signal Processing 29 (2014) 1–19.
- [63] F. M. Bianchi, L. Livi, C. Alippi, R. Jenssen, Multiplex visibility graphs to investigate recurrent neural network dynamics, Scientific reports 7 (2017) 44037.
- [64] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.
- [65] K. H. Brodersen, T. M. Schofield, A. P. Leff, C. S. Ong, E. I. Lomakina, J. M. Buhmann, K. E. Stephan, Generative embedding for model-based classification of fmri data, PLoS computational biology 7 (6) (2011) e1002079.

Supplementary Material

In the following, we provide the details of the aggregated results shown in the experimental section. Fig. 8 depicts the ranking of the accuracy achieved by each MTS classifier on the benchmark datasets (see details in Tab. 1). Best performance (higher accuracy) correspond to lower values in ranking and darker color code.

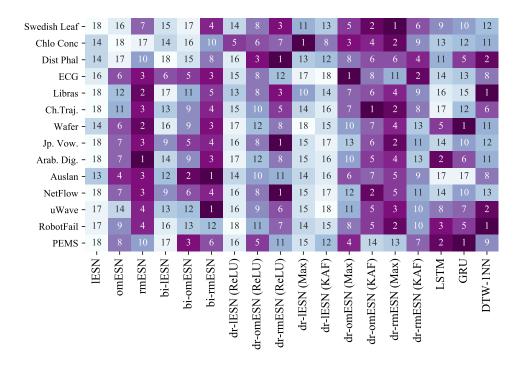


Figure 8: Ranking of the accuracy obtained by the MTS classifiers on benchmark classification dataset.

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.

In the tables below, we also report the detailed results obtained on each dataset. 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.

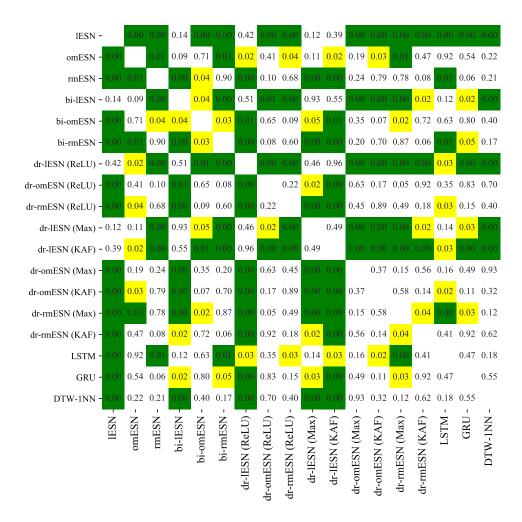


Figure 9: Results (p-values) of the post-hoc test. Yellow boxes indicate p-value < 0.05, Green boxes indicate p-value < 0.01.

Table 2: Results on Swedish Leaf dataset.

Swedish Leaf	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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	87.01	0.80	0.86	0.01	0.16
bi- $lESN$	70.98	4.34	0.69	0.05	0.37
bi-omESN	63.90	1.81	0.61	0.02	0.39
bi-rmESN	89.25	0.66	0.89	0.01	0.39
dr- $lESN$ $(ReLU)$	78.14	0.95	0.78	0.01	0.71
dr- $omESN$ $(ReLU)$	85.79	2.27	0.86	0.02	0.72
dr- $rmESN$ $(ReLU)$	92.64	0.77	0.93	0.01	1.41
dr- $lESN (Max)$	79.42	1.37	0.79	0.02	0.68
dr- $omESN$ (Max)	87.87	1.25	0.88	0.01	0.69
dr- $rmESN (Max)$	94.56	0.66	0.95	0.01	1.61
dr- $lESN$ (KAF)	78.43	2.03	0.78	0.02	2.46
dr- $omESN$ (KAF)	87.14	0.72	0.87	0.01	2.51
dr- $rmESN$ (KAF)	93.47	0.86	0.93	0.01	2.58
LSTM	83.58	0.71	0.83	0.01	8.60
GRU	82.24	1.62	0.82	0.02	9.39
DTW-1NN	78.72	_	0.78	_	329.99

Table 3: Results on Chlorine Concentration dataset.

Chlo Conc	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	Time (mins)
lESN	58.18	0.26	0.47	0.00	0.62
omESN	56.15	0.28	0.43	0.01	0.68
rmESN	57.02	0.41	0.48	0.01	0.70
bi- $lESN$	58.18	0.38	0.49	0.00	1.35
$bi ext{-}omESN$	57.99	0.56	0.48	0.01	1.40
bi-rmESN	63.72	0.62	0.59	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 4: Results on Distal Phalanx Outline dataset.

Dist Phal	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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	71.80	1.32	0.71	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	72.23	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	0.69	0.02	0.62
dr- $omESN$ (Max)	72.23	2.97	0.72	0.03	0.62
dr- $rmESN (Max)$	72.52	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)	72.52	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

Table 5: Results on Electrocardiography dataset.

ECG	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 6: Results on Libras dataset.

Libras	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 7: Results on Character Trajectory dataset.

Ch.Traj.	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 8: Results on Wafer dataset.

Wafer	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 9: Results on Japanese Vowels dataset.

Jp. Vow.	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 10: Results on Arabic Digits dataset.

Arab. Dig.	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 11: Results on Australian Sign Language Signs dataset.

Auslan	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 12: Results on Network Flow dataset.

NetFlow	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 13: Results on uWave dataset.

uWave	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 14: Results on Robotic Arm Failure dataset.

RobotFail	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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 15: Results on Peformance Measurement System dataset.

PEMS	Acc. (mean)	Acc. (std)	F1 (mean)	F1 (std)	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