Università di Pisa
Dipartimento di Informatica

Technical Report: TR-09-22

# On the Predictive Effects of Markovian and Architectural Factors of Echo State Networks

Claudio Gallicchio, Alessio Micheli
Dipartimento di Informatica, Università di Pisa,
Largo B. Pontecorvo, 3, 56127 Pisa, Italia
*e-mail:gallicch@di.unipi.it, micheli@di.unipi.it*

November 23, 2009

ADDRESS: Largo B. Pontecorvo 3, 56127 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

# On the Predictive Effects of Markovian and Architectural Factors of Echo State Networks

Claudio Gallicchio, Alessio Micheli
Dipartimento di Informatica, Università di Pisa,
Largo B. Pontecorvo, 3, 56127 Pisa, Italia
*e-mail:gallicch@di.unipi.it, micheli@di.unipi.it*

November 23, 2009

**Abstract**

Echo State Networks (ESNs) represent an emerging paradigm for modeling Recurrent Neural Networks (RNNs). In this report we try to identify and investigate some of the main aspects that can be accounted for the success and limitations of this class of models. Independently of the architectural design, we first show the effect on ESNs behavior due to the contractivity of the state transition function and the related Markovian bias. The purpose of our study is also to give an insight on how and why a larger reservoir may improve the predictive performance. We identify four key factors which can influence the performance of ESNs: input variability, multiple time-scales dynamics, non-linear interactions among units and regression in a high dimensional state space. Several variants of the basic ESN model are introduced in order to study these main factors. The proposed variants are tested on four datasets: the Mackey-Glass chaotic time series, the 10th order NARMA system, and two predictive tasks on a symbolic sequence domain with Markovian/anti-Markovian flavor. Experimental evidence shows that all the key identified factors have a major role in determining ESNs performances.

## 1 Introduction

Echo State Networks (ESNs) [16, 21] constitute a novel approach for efficiently modeling Recurrent Neural Networks (RNNs). An ESN (typically) consists in a large and sparsely connected *reservoir* layer of recurrent neurons, connected to a simple *readout* layer of linear neurons. The striking feature of ESNs is that the recurrent part of the network is initialized according to a specific criterion and then is left untrained. The only trained part is the linear readout, which can be adapted by using efficient linear regression. ESNs have been successfully applied in several sequential domains, such as non linear system identification [17], robot control [29, 15, 14], speech processing [33], time series prediction and noise modeling [21]. Besides this widespread success in applications, a number of open issues still remain and motivate the research effort in this area. Some of the main research topics on ESNs [20] focus on the optimization of reservoirs towards specific problems [15, 30, 31], the role of topological organization of

reservoirs [43] and the properties of reservoirs that are responsible for successful or unsuccessful applications [28, 10]. In particular, this last topic, considered also in relation with the (usually) high dimensionality of reservoirs, is of a special interest for the aims of this paper.

It is a known fact that RNNs initialized with contractive state transition functions are able to discriminate among different (recent) input histories even prior to learning [12, 36], according to a Markovian organization of the state dynamics. Input sequences sharing a common suffix drive the same network into states which are close to each other proportionally to the length of the common suffix. The Markovian characterization of contractive mappings has been studied in the contexts of Iterated Function Systems (IFSs), variable length memory predictive models, fractal theory and for describing the bias of trainable RNNs initialized with small weights [12, 36, 37].

Such analysis also applies to ESNs due to the contractive setting of the state transition function. In particular, ESNs exploit the consequences of Markovianity of state dynamics in combination with a typically high dimensionality of the recurrent reservoir. The importance of a richly varied ESN state dynamics within a large number of reservoir units has been theoretically and experimentally pointed out in ESN literature (e.g. [16, 21, 37, 39]), although not completely analyzed. Moreover, high dimensional reservoir constitutes the basis to argue an universal approximation property with bounded memory of ESNs, even in presence of a linear readout layer [37]. Indeed, although the Markovian organization of the reservoir state space rules the dynamics of ESNs, it is known (e.g.[39, 26, 19]) that large reservoirs show a goodness of predictive results on sequence tasks which is almost proportional to the number of reservoir units. The Markovian characterization of the reservoir state space is therefore not sufficient to completely explain the performances of the model.

These points open interesting issues, motivating our investigation on the factors which may influence the model behavior and on the assessment of their relative importance.
Markovianity of state dynamics is considered in relation to the issue of identifying relevant factors which might determine success and limitations of the ESN model on predictive tasks.
The aspect of high reservoir dimensionality is studied by asking to which extent performance improvements obtained by increasing the number of reservoir units is due to a larger number of recurrent dynamics or to the effect of the possibility to regress an augmented state space.
Relating to the same issue, we also propose a study of the different architectural factors of ESN design which allow the model units to effectively diversify their activations and lead to enrichment of the reservoir dynamics. This is done by measuring and comparing the effects on the final predictive accuracy (*performance* in the following) due to the inclusion of individual factors and combination of factors in the design of ESNs. This study also investigates the effectiveness of the characteristic of sparsity among reservoir units connections, which is commonly claimed to be a crucial feature of ESN modeling.

Recently, there has been a growing interest in studying architectural variants and simplifications of the basic ESN model. In particular, a number of reservoir models with a simpler architecture than ESN have been proposed. A model with self-recurrent connections only, linear reservoir neurons and unitary input-to-reservoir weights, the so called "Simple ESN" (SESN) was presented

in [7]. A feed-forward variant of ESN, the "Feed-Forward ESN" (FFESN), was introduced in [3], while in [4] a further simplification of the model with reservoir units organized into a tapped delay line was proposed. Other recent works on the effects of the ESN components on the predictive performance, such as the settling time and spectral radius, can be found in [38]. Our work, being directed towards a deeper understanding of the comparative predictive performance effects of different architectural factors of ESN design, can also be intended in these research directions as well.

The rest of this report is organized as follows. Section 2 reviews RNNs in the framework of sequence transduction processing. Section 3 provides an introduction to the main concepts of ESN modeling, using the same framework introduced in Section 2. Section 4 focuses on the Markovian organization of reservoirs state dynamics deriving from contractivity of state transition functions. Section 5 introduces the identified architectural factors of ESN design and the corresponding architectural variants proposed to the basic ESN model. Experimental results are illustrated in Section 6, by firstly discussing the influence of Markovianity on ESN performance based on two ad-hoc designed tasks, and then assessing the relevance of the proposed architectural factors for standard sequence processing tasks showing a significant effect of the reservoir dimensionality. Finally, Section 7 summarizes the main general results of the report as well as a number of simple outcomes for practical use of ESNs.

# 2 A Framework for Recurrent Sequences Processing

In this paper we are interested in processing sequence domains. Let $\mathcal{U}$ be an input space, then by $\mathcal{U}^*$ we denote the set of all possible sequences of finite length on $\mathcal{U}$. In the same way we denote by $\mathcal{Y}$ an output space and by $\mathcal{Y}^*$ the set of all finite length sequences on $\mathcal{Y}$. A function $\mathcal{T}$, mapping a sequence domain into another sequence domain, is called a *sequence transduction*:

$$\mathcal{T} : \mathcal{U}^* \rightarrow \mathcal{Y}^* \tag{1}$$

We denote by $\mathbf{u} \in \mathcal{U}$ an input (vector) element, while $s(\mathbf{u}) \in \mathcal{U}^*$ is an input sequence. In particular, if $s(\mathbf{u})$ is of length $n$, then we can show its elements by using the notation $s(\mathbf{u}) = [\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n)]$, where $\mathbf{u}(1)$ is the oldest entry and $\mathbf{u}(n)$ is the most recent one. An empty input sequence is denoted by $s(\mathbf{u}) = [\,]$. An output element and an output sequence are likewise represented by $\mathbf{y} \in \mathcal{Y}$ and $s(\mathbf{y}) \in \mathcal{Y}^*$, respectively. For our purposes, a sequence transduction $\mathcal{T}$ can be usefully decomposed into an *encoding function* $\tau$ and an *output function* $g$, as follows

$$\mathcal{T} = g \circ \tau \tag{2}$$

The encoding function $\tau$ maps an input sequence into a sequence of elements in a (hidden) feature space $\mathcal{H}$:

$$\tau : \mathcal{U}^* \rightarrow \mathcal{H}^* \tag{3}$$

An element in the feature space $\mathcal{H}$ is denoted by $\mathbf{x}$. The output function maps a sequence on $\mathcal{H}$ into a sequence of output elements:

$$g : \mathcal{H}^* \rightarrow \mathcal{Y}^* \tag{4}$$

Note that the encoding function and the output function are defined as sequence transductions themselves.
Both the encoding and the output functions can be computed by resorting to element-wise applied recurrent functions:

$$\begin{aligned} &\hat{\tau} : \mathcal{H} \times \mathcal{U} \to \mathcal{H} \\ &\mathbf{x}(n) = \hat{\tau}(\mathbf{x}(n-1), \mathbf{u}(n)) \\ &\\ &\hat{g} : \mathcal{H} \to \mathcal{Y} \\ &\mathbf{y}(n) = \hat{g}(\mathbf{x}(n)) \end{aligned} \tag{5}$$

where $\mathbf{u}(n)$, $\mathbf{x}(n)$ and $\mathbf{y}(n)$ are the input, feature and output elements for step $n$, respectively. A sequence transduction $\mathcal{T}$ can be qualified in several ways. We say that $\mathcal{T}$ is ([8, 11])

- *Input-Output Isomorphic*, or *sequence-to-sequence*, if it associates an output element $\mathbf{y}(n)$ for each input element $\mathbf{u}(n)$. In this case the input sequence $s(\mathbf{u})$ is always the same length as the output sequence $s(\mathbf{y})$. A particular instance concerns the case of next-step prediction tasks.
  Another noteworthy case is when a single output element $\mathbf{y}$ is associated to each input sequence $s(\mathbf{u})$, generally in correspondence to the last input entry. In this case the output space contains only vectors of a fixed size and the transduction is said to be a *sequence-to-element* transduction. This case describes sequence classification and real valued output time-series prediction tasks.

- *Causal* if the output computed for $\mathbf{u}(n)$ depends on $\mathbf{u}(n)$ itself and a number of input entries which are in a causal relation to $\mathbf{u}(n)$. A very common kind of causality is always assumed when processing temporal sequences. *Temporal causality* means that the output computed by $\mathcal{T}$ at a certain time step $n$ does only depend on the actual input $\mathbf{u}(n)$ and the previous input entries $\mathbf{u}(n-1), \mathbf{u}(n-2), \ldots, \mathbf{u}(1)$.

- *Stationary* if the function applied by $\mathcal{T}$ to compute an output element in correspondence of the input entry $\mathbf{u}(n)$ does not change as the input entry changes.

- *Adaptive* if the function computed by $\mathcal{T}$ is learned from data. Opposite to adaptive transductions are *fixed* transductions, which are defined a-priori for a class of data or tasks. A sequence transduction $\mathcal{T}$ as in equation (2) is fully adaptive if both the encoding and the output functions are adaptive, and partially adaptive is only one of them is adaptive. Adaptive transductions are preferable to fixed ones because of their broader applicability, but on the other hand they might require expensive training algorithms which could even make the application on real world problems almost infeasible.

Recurrent Neural Networks (RNNs) [13] are neural network models capable of computing transductions on sequence domains according to the decomposition elucidated above. In the simplest RNN architecture there are three layers of units: an input layer, an hidden layer of recurrent units which compute the encoding function $\hat{\tau}$, and a layer of output units which compute the output

function $\hat{g}$. In this case, the input, hidden and output spaces are real subspaces denoted by $\mathbb{R}^{N_U}$, $\mathbb{R}^{N_R}$ and $\mathbb{R}^{N_Y}$, respectively[1]. The element-wise application of the encoding function is the *state transition function*:

$$\mathbf{x}(n) = \hat{\tau}(\mathbf{x}(n-1), \mathbf{u}(n)) = f(\mathbf{W}_{in}\mathbf{u}(n) + \hat{\mathbf{W}}\mathbf{x}(n-1)) \tag{6}$$

where $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U+1}$ is the input-to-hidden weight matrix also containing the biases for the hidden neurons, $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent weight matrix, and $f$ is the activation function which usually is a nonlinearity of sigmoidal type. The feature element computed by the network at pass $n$ is called the *state* of the network at pass $n$, i.e. $\mathbf{x}(n)$. The state at $n = 0$, i.e. $\mathbf{x}(0)$, is called the *initial state* of the network. We also introduce an iterated version of the element-wise encoding function $\hat{\tau}$ of equation (6):

$$\tilde{\tau} : \mathbb{R}^{N_R} \times (\mathbb{R}^{N_U})^* \rightarrow \mathbb{R}^{N_R}$$

$$\forall \mathbf{x} \in \mathbb{R}^{N_R}, \;\; \forall s(\mathbf{u}) \in (\mathbb{R}^{N_U})^* :$$

$$\tilde{\tau}(\mathbf{x}, s(\mathbf{u})) = \begin{cases} \hat{\tau}(\tilde{\tau}(\mathbf{x}, [\mathbf{u}(1), \ldots, \mathbf{u}(n-1)]), \mathbf{u}(n)) & \text{if } \; s(\mathbf{u}) = [\mathbf{u}(1), \ldots, \mathbf{u}(n)] \\ \mathbf{x} & \text{if } \; s(\mathbf{u}) = [\,] \end{cases} \tag{7}$$

where $\tilde{\tau}(\mathbf{x}, s(\mathbf{u}))$ is the state of the network which has been driven by the input sequence $s(\mathbf{u})$ starting from initial state $\mathbf{x}$.
The output layer computes the function $\hat{g}$:

$$\mathbf{y}(n) = \hat{g}(\mathbf{x}(n)) = f_{out}(\mathbf{W}_{out}\mathbf{x}(n)) \tag{8}$$

where $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_R+1}$ is the hidden-to-output weight matrix (plus the biases) and $f_{out}$ is the activation function of the output units.
The class of sequence transductions computed by RNNs, as described here, can be classified as fully adaptive (as both the encoding function and the output function can be learned from data) and stationary. In the following we also assume temporal causality, as we only deal with temporal sequences.
RNNs are theoretically very powerful, indeed they have been shown to be Turing-equivalent [32]. Moreover, they have been successfully applied to many real world application domains, such as signal processing, speech recognition, financial forecasting, biological sequence processing, and robotics, just to name a few (see for instance [22] and the references therein).

One of the major issues concerning RNN models regards learning. In general, both the encoding and the output functions are adapted by a training algorithm which is responsible for properly adjusting the weight matrices $\mathbf{W}_{in}$, $\hat{\mathbf{W}}$ and $\mathbf{W}_{out}$ in a task dependent way. A number of training algorithms have been proposed for RNNs, most of them are based on a gradient-descending technique. BackPropagation Through Time (BPTT) [40] and Real-Time Recurrent Learning (RTRL) [41] are among the others two of the most known implementations of this technique. Unfortunately, several drawbacks are involved in using this standard class of training algorithms. One of the major problems is their high computational training costs, which could make their applications to several real world domains almost infeasible. Other main problems concern the possibility

[1] $N_R$ refers to the dimensionality of the state space representation, i.e. of the reservoir for ESNs.

of getting stuck in local minima, slow convergence and the problem of learning long term dependencies [44]. These issues motivate the efforts in designing new solutions for training RNNs, which is still an open area of ongoing research.

In this paper we consider an alternative RNN modeling approach which goes by the name of Reservoir Computing. Reservoir Computing (RC) [23] is a denomination for a class of RNN models that share some basic common features. Reservoir networks are characterized by a conceptual separation between a recurrent dynamical part (the *reservoir*) and a simple non-recurrent output tool (the *readout*). In the most basic setting, the reservoir is implemented by a large (high dimensional) and random layer of recurrent hidden neurons, which is initialized according to some criterions and then left untrained. The readout is implemented through a layer of (typically) linear neurons and is adapted by using a simple and efficient training algorithm for (non-recurrent) feedforward networks.

RC is also claimed to have a strong biological plausibility. Indeed several relationships have been discovered between the properties of animal brains and reservoir networks. Examples of these relationships can be found in [27, 5, 6, 42, 9].

Reservoir networks implement causal, stationary and partially adaptive sequence transductions in which the encoding function (the state transition function) is realized by a fixed dynamical reservoir, and the output function is realized by the adaptive readout. The key observation about reservoirs is that as long as they satisfy some very easy-to-check properties, they are able to discriminate among different input histories even in the absence of training. In this way it is possible to restrict the training just to a simple recurrent-free readout. This eliminates the recurrent dependencies in the weight adjusting process and leads to a very efficient RNN design.

RC includes several classes of RNN models, including the popular Echo State Networks (ESNs) [16], Liquid State Machines (LSMs) [24] and other approaches such as BackPropagation Decorrelation (BPDC) [34, 35] and Evolino [30]. In this paper we focus on the ESN approach. The next Section introduces the basics of ESN modeling in the framework defined so far.

# 3 Echo State Networks

## 3.1 Model

Echo State Networks have been introduced in [16] and further investigated in many other works (e.g. [18, 17]). An ESN consists in an input layer of $N_U$ units, a number of $N_R$ recurrent hidden units (the reservoir) and an output layer of $N_Y$ typically linear and non recurrent units (the readout). The basic equations describing the computation carried out by an ESN are the same as equations (6) and (8):

$$\begin{aligned} \mathbf{x}(n) &= \hat{\tau}(\mathbf{x}(n-1), \mathbf{u}(n)) = f(\mathbf{W}_{in}\mathbf{u}(n) + \hat{\mathbf{W}}\mathbf{x}(n-1)) \\ \mathbf{y}(n) &= \hat{g}(\mathbf{x}(n)) = \mathbf{W}_{out}\mathbf{x}(n) \end{aligned} \tag{9}$$

where $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U+1}$ is the input-to-reservoir weight matrix, $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent reservoir weight matrix and $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_R+1}$ is the reservoir-

to-output weight matrix. Equation (9) also describes the basic ESN architecture which is referred in this paper as standard ESN. Moreover, hyperbolic tangent is typically used as reservoir activation function (i.e. $f = tanh$), while the output of the network is a linear combination of the reservoir output. An ESN computes a sequence transduction in which the encoding function (state transition function) is fixed and the output function is adapted from training data. In particular, the reservoir of the network computes the element-wise encoding function $\hat{\tau}$, while the readout computes the element-wise version of the output function, $\hat{g}$. This means that only the weights to the output layer of the network are adjusted, i.e. $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$ are fixed and only $\mathbf{W}_{out}$ is adapted. Figure 1 illustrates the architecture of an ESN. Not every choice of $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$ leads to a valid echo state network. Indeed the reservoir must satisfy the so called *echo state property*. This is discussed in the following.

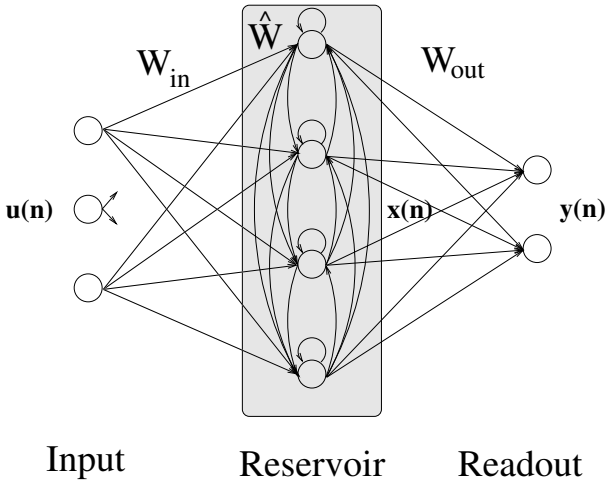


**Figure 1:** Basic architecture of an ESN with a number of $N_U = 3$ input units, $N_R = 4$ reservoir units and $N_O = 2$ output units.

## 3.2 Echo State Property and Contractivity

A valid ESN satisfies the so called *echo state property* [16]. This says that the state in which the network is after being driven by a long input sequence does only depend on the input sequence itself. The dependence on the initial state of the network is progressively lost, as the length of the input sequence goes to infinity. Equivalently, the current state $\mathbf{x}(n)$ of the network is a function of its past input history independently of initial state values. In formulas, the echo state property may be expressed as follows:

$$\begin{gathered} \forall s_n(\mathbf{u}) = [\mathbf{u}(1), \ldots, \mathbf{u}(n)] \in (\mathbb{R}^{N_U})^n \quad \text{input sequence of length } n, \\ \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R} : \\ \|\tilde{\tau}(\mathbf{x}, s_n(\mathbf{u})) - \tilde{\tau}(\mathbf{x}', s_n(\mathbf{u}))\| \to 0 \ \text{ as } \ n \to \infty \end{gathered} \tag{10}$$

which means that the distance between the states in which the network is driven by an input sequence of length $n$ approaches zero (for any choice of the initial

states) as $n$ goes to infinity. In [16] a valid echo state network has also been characterized as "input forgetting", "state contracting" and "state forgetting". Moreover, two conditions have been provided as necessary and sufficient, respectively, for a network (with *tanh* as activation function) having echo states. The necessary condition is that the *spectral radius* (the largest eigenvalue in absolute value) of the reservoir recurrent weight matrix is less than one

$$\rho(\hat{\mathbf{W}}) < 1 \tag{11}$$

If this condition is violated, the dynamical reservoir is locally asymptotically unstable at the zero state $\mathbf{0} \in \mathbb{R}^{N_R}$ and echo states cannot be guaranteed if the null sequence is an admissible input for the system. The sufficient condition for the presence of echo states is that the *largest singular value* of $\hat{\mathbf{W}}$ is less than unity,

$$\sigma(\hat{\mathbf{W}}) < 1 \tag{12}$$

The Euclidean norm of $\hat{\mathbf{W}}$ is equal to its largest singular value provided that $\hat{\mathbf{W}}$ is a square matrix, thus the sufficient condition can be restated as $\|\hat{\mathbf{W}}\|_2 < 1$. This condition ensures global stability of the system and thus the presence of echo states.

Another very important characteristic of echo state networks is *contractivity* of its state dynamics. The state transition function $\hat{\tau}$ is contractive if it satisfies the following condition:

$$\begin{gathered} \exists K \in \mathbb{R}, 0 < K < 1, \;\; \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R}, \;\; \forall \mathbf{u} \in \mathbb{R}^{N_U} : \\ \|\hat{\tau}(\mathbf{x}, \mathbf{u}) - \hat{\tau}(\mathbf{x}', \mathbf{u})\| \leq K \|\mathbf{x} - \mathbf{x}'\| \end{gathered} \tag{13}$$

in other words, $\hat{\tau}$ must be Lipschitz continuous with a parameter $K$ less than unity. Contractivity of the state transition function ensures the echo state property of equation (10). Indeed, if $\hat{\tau}$ is contractive with parameter $K < 1$, then for every input sequence of length $n$, $[\mathbf{u}(1), \ldots, \mathbf{u}(n)]$, and for every initial states $\mathbf{x}$ and $\mathbf{x}'$:

$$\begin{aligned} &\|\tilde{\tau}(\mathbf{x}, [\mathbf{u}(1), \ldots, \mathbf{u}(n)]) - \tilde{\tau}(\mathbf{x}', [\mathbf{u}(1), \ldots, \mathbf{u}(n)])\| \\ &= \|\hat{\tau}(\tilde{\tau}(\mathbf{x}, [\mathbf{u}(1), \ldots, \mathbf{u}(n-1)]), \mathbf{u}(n)) - \hat{\tau}(\tilde{\tau}(\mathbf{x}', [\mathbf{u}(1), \ldots, \mathbf{u}(n-1)]), \mathbf{u}(n))\| \\ &\leq K \|\tilde{\tau}(\mathbf{x}, [\mathbf{u}(1), \ldots, \mathbf{u}(n-1)]) - \tilde{\tau}(\mathbf{x}', [\mathbf{u}(1), \ldots, \mathbf{u}(n-1)])\| \\ &\leq \ldots \\ &\leq K^{n-1} \|\tilde{\tau}(\mathbf{x}, [\mathbf{u}(1)]) - \tilde{\tau}(\mathbf{x}', [\mathbf{u}(1)])\| \\ &= K^{n-1} \|\hat{\tau}(\tilde{\tau}(\mathbf{x}, [\,]), \mathbf{u}(1)) - \hat{\tau}(\tilde{\tau}(\mathbf{x}', [\,]), \mathbf{u}(1))\| \\ &= K^{n-1} \|\hat{\tau}(\mathbf{x}, \mathbf{u}(1)) - \hat{\tau}(\mathbf{x}', \mathbf{u}(1))\| \\ &\leq K^n \|\mathbf{x} - \mathbf{x}'\| \end{aligned} \tag{14}$$

which clearly approaches 0 as $n \to \infty$. Note that this argument is valid for any norm in which $\hat{\tau}$ is a contraction, hence contractivity of the state transition function *in any norm* is a sufficient condition for the echo state property[2]. On the other hand, if contractivity of the state transition function cannot be guaranteed, at least definitely for any couple of network states that can be

[2]In the original definition of the echo state property in [16], the Euclidean norm $\|\cdot\|_2$ is used. However, as finite-dimensional norms are all equivalent, if the echo state property holds for any given norm, then it also holds for the Euclidean norm, and the original echo state property is satisfied.

reached after a long common input sequence, the presence of echo states, in general, cannot be guaranteed as well.
For networks using the hyperbolic tangent as activation function, contractivity in the Euclidean norm is very easily ensured if the condition $\sigma(\hat{\mathbf{W}}) = \|\hat{\mathbf{W}}\|_2 < 1$ holds. Indeed, in this case $\hat{\tau}$ is always a contraction in this norm:

$$\begin{aligned}
&\|\hat{\tau}(\mathbf{x}, \mathbf{u}) - \hat{\tau}(\mathbf{x}', \mathbf{u})\|_2 \\
&= \|tanh(\mathbf{W}_{in}\mathbf{u} + \hat{\mathbf{W}}\mathbf{x}) - tanh(\mathbf{W}_{in}\mathbf{u} + \hat{\mathbf{W}}\mathbf{x}')\|_2 \\
&\leq max(|tanh'|)\|\hat{\mathbf{W}}(\mathbf{x} - \mathbf{x}')\|_2 \\
&\leq \|\hat{\mathbf{W}}\|_2\|\mathbf{x} - \mathbf{x}'\|_2
\end{aligned}$$

This leads to the sufficient condition of equation (12) proposed in [16]. Note that even if $\hat{\tau}$ is not contractive in the Euclidean norm, it could be contractive in another norm, and in this case the echo state property would hold anyway. In [2], a different norm ($D$-norm) is introduced for which the contraction condition of the state transition function is less restrictive than the condition in equation (12). For simplicity, however, in this work we use only contractivity in the Euclidean norm.

### 3.3 Obtaining and Training ESNs

To build up a valid echo state network it is possible to start with a randomly generated matrix $\hat{\mathbf{W}}_{random}$ and then rescale it to satisfy the required condition (equation (11) or (12)). For the sufficient condition, this is done as follows:

$$\hat{\mathbf{W}} = \frac{\sigma}{\sigma(\hat{\mathbf{W}}_{random})}\hat{\mathbf{W}}_{random} \tag{15}$$

which ensures that the obtained matrix $\hat{\mathbf{W}}$ has a maximum singular value equal to $\sigma$. The same could be done to rescale the spectral radius of $\hat{\mathbf{W}}$ to a desired value. In most of the ESN literature, the spectral radius of $\hat{\mathbf{W}}$ is scaled to meet the necessary condition ($\rho(\hat{\mathbf{W}}) < 1$), which does however not ensure echo states. In this work, instead, as we are mainly interested in contractivity, we focus on the stronger condition on the Euclidean norm ($\sigma(\hat{\mathbf{W}}) < 1$), which is quite restrictive but always guarantees the contractivity of the state transition function and thus the echo state property.
Moreover, the typical recipe for echo state network preparation prescribes that $\hat{\mathbf{W}}$ is a sparse matrix with a fixed (usually less than 20%) percentage of connectivity. The intuition behind this is that a sparsely connected recurrent reservoir would ensure a rich and loosely coupled pool of dynamics from which the readout should take advantage. However, it has been noted (e.g. [43]) that reservoir units (even with sparse connectivity) may exhibit coupled behaviors, and thus the original intuition is probably misleading. Nonetheless, sparsely connected reservoirs are however preferable to fully connected ones for computational reasons. In this work we consider both fully connected and sparsely connected ESNs.
In the following, the Euclidean norm is always assumed and the symbol $\sigma$ is used to refer the Euclidean norm of the recurrent weight matrix $\hat{\mathbf{W}}$ (properly scaled as in (15)). The value of $\sigma$ is also called the *contraction coefficient* of the network, as it governs the contractivity (at least in the Euclidean norm) of

the state transition function. As a consequence, the value of the contraction coefficient should be properly set to match the target system dynamics. As a rule of thumb, a larger $\sigma$ corresponds to slower dynamics and longer memory of the target system and a smaller $\sigma$ corresponds to faster dynamics and smaller memory [16].

Consider a supervised learning task on a sequence domain described by a training set of $N_{train}$ examples $(\mathbf{u}(n), \hat{\mathbf{y}}(n))_{n=1}^{N_{train}}$, where $[\mathbf{u}(1), \ldots, \mathbf{u}(N_{train})]$ is the training input sequence and $[\hat{\mathbf{y}}(1), \ldots, \hat{\mathbf{y}}(N_{train})]$ is the target output sequence. The goal of the ESN training procedure is to find the appropriate values of the reservoir-to-output weights of matrix $\mathbf{W}_{out}$ which minimize the averaged squared error on the training set. In an off-line setting this can be accomplished in a very simple way by computing the reservoir states corresponding to the presentation of the input sequence to the network and then using these states as input for training the readout. Let denote by $\mathbf{x}(1), \ldots, \mathbf{x}(N_{train})$ these states. If the reservoir is a valid reservoir, the echo state property of equation (10) ensures that, after a sufficiently long input sequence, the state of the network is a function of the input sequence only, and the dependence on the initial conditions have died out. For this reason, we can dismiss the first $N_{transient}$ states of the network which may still be affected by the initial conditions and consider only the remaining states $\mathbf{x}(N_{transient}+1), \ldots, \mathbf{x}(N_{train})$. The time steps corresponding to the dismissed states are also known as the initial *transient*, or *washout*, of the network. As this transient depends on how fast the state dynamics forgets the initial conditions, a longer initial transient is generally required for reservoirs with a larger contraction coefficient. To make the notation more compact, we denote by $\mathbf{X}$ the matrix whose columns are the reservoir states $\mathbf{x}(N_{transient}+1), \ldots, \mathbf{x}(N_{train})$. In a similar way, the corresponding target vectors $\hat{\mathbf{y}}(N_{transient}+1), \ldots, \hat{\mathbf{y}}(N_{train})$ are column-wise arranged into the matrix $\hat{\mathbf{Y}}$.

For sequence-to-element transductions, for which the output is computed after the complete presentation of the input sequence, the training set is composed of $N_{train}$ examples $(s_n(\mathbf{u}), \hat{\mathbf{y}}(n))_{n=1}^{N_{train}}$, where $s_n(\mathbf{u})$ is the $n$-th input sequence and $\hat{\mathbf{y}}(n)$ is the corresponding target output. In this case, the state of the network corresponding to the $n$-th input sequence is the state in which the network is driven after a complete presentation of the sequence $s_n(\mathbf{u})$. Moreover, to account for the initial transient, each input sequence is actually presented a prescribed number of consecutive times to the reservoir before the corresponding state is stored in $\mathbf{X}$.

Training the readout is then a simple problem of linear regression consisting in minimizing the squared error

$$\|\mathbf{W}_{out}\mathbf{X} - \hat{\mathbf{Y}}\|^2 \tag{16}$$

This is usually solved by using a pseudo-inversion of matrix $\mathbf{X}$:

$$\mathbf{W}_{out} = \hat{\mathbf{Y}}\mathbf{X}^{+} \tag{17}$$

The readout of an ESN can also be trained in an on-line fashion. This can be done by using the Recursive Least Mean Squares algorithm, as pointed out in [17], while the Least Mean Squares algorithm is usually unsuitable because of the high eigenvalue spread of the correlation matrix of $\mathbf{X}$ [20, 17].

In this paper we use off-line training with pseudo-inversion.

# 4 Markovian Characterization of ESNs

Contractivity of the state transition function is also responsible for putting a bias on the theoretical computational capabilities of a RNN. In [12] it has been proved that the class of RNNs with *contractive state transition function* and bounded state space is equivalent to (can be approximated arbitrarily well by) the class of definite memory machines (DMMs), i.e. they can only compute functions with a bounded memory on the input history. This architectural bias has been qualified as Markovian in [36]. In fact, as a consequence of the contractivity of the state dynamics, the influence of old inputs in determining the actual state of the network gradually dies out. If two any input sequences share a common suffix (i.e. the last input elements), then they will drive the network into close states, and these states will be closer to each other for longer common suffixes, a condition which can be related to the echo state property of equation (10). Through a contractive initialization of the state transition function, the state dynamics of RNNs exhibits the capacity of discriminating between input sequences in a Markovian way even prior to learning. Thus, if a task of interest only involves Markovian processes then it can be faced by a contractive RNN in which the state transition function is left untrained, such is the case of ESNs. Note that ESNs are characterized by contractivity, as explained above, and have a bounded state space under very mild assumptions (e.g. for bounded activation functions like the hyperbolic tangent). Therefore the ESN dynamics are clearly characterized by a Markovian nature. This is also supported by the results on the "fading memory" property ([16]) accounting for the closeness of ESN representations of sequences sharing the most recent inputs. We can thus expect a good performance of this class of models on Markovian tasks, and a poorer one on tasks with a less prominent Markovian nature or which are non-Markovian at all (in particular anti-Markovian tasks). To face non-Markovian tasks a form of adaptation of the state dynamics is needed, which leads out to the aim of the paper.

The intrinsic Markovian characteristic of reservoir dynamics is called in this paper the *Markovian factor*. This factor characterizes every ESN with contractive dynamics independently of the architectural design. In this paper the contraction coefficient is used to control the Markovian factor, with smaller $\sigma$ corresponding to a more prominent Markovianity.

The relevance of this factor can be highlighted introducing specific experiments aimed at showing that it characterizes easy/hard tasks independently of the architectural design (see Section 6.2).

There is another interesting question arising from the observation that the class of ESNs is equivalent to the class of DMMs. As contractivity seems to be the ruling characteristic of reservoirs, then one could expect that even a smaller contractive reservoir, or a contractive reservoir of neurons with any possible architecture of connectivity (e.g. only self recurrent connections) would perform well on Markovian tasks just as standard ESNs would do. If this intuition were true, then even a single-unit reservoir (even a linear one) could be used to solve any Markovian task as long as it is characterized by a contractive state dynamics.

However, several works have reported that larger reservoirs of neurons have better predictive performances than smaller ones on different non-linear tasks (e.g. see [39]). Increasing the dimensionality of reservoirs seems to be the sim-

pler way to get better performances. Moreover, as pointed out in [28], different reservoir instances, obtained with the same scaling settings of the recurrent weight matrix, may result in different performances on the same task.

Our investigation about the key architectural factors which may improve large reservoirs performance stems from these observations. Accordingly, several architectural variants of the basic ESN model are introduced in Section 5.

# 5 Architectural Factors of ESNs

Even though reservoirs dynamics are governed by the Markovian factor, there still are several other factors, related to the architectural design, which might influence the richness of the Markovian dynamics and thus the prediction performance of ESNs. Indeed ESNs with the same contractive coefficient but different topologies can lead to different results on the same task. At the same time, the richness of the dynamics is related to the growth of the number of units (reservoir dimensionality). It is therefore interesting to investigate the factors determining the differentiation among the units.

We identified four architectural factors which may have an impact on ESNs performance, namely: input variability, multiple time-scale dynamics, non-linear interactions among units and regression in a high dimensional feature space. Each factor is described more extensively in the following:

**Input Variability.** This refers to the possibility for the reservoir units of looking at each element of the input sequence under several points of view. The configuration without input variability is obtained fixing the same values of the input weights for all the reservoir units. Input variability is implemented through a random initialization of the input-to-reservoir weight matrix $\mathbf{W}_{in}$ with different values among units.

**Multiple Time-Scale Dynamics.** This refers to the possibility for the reservoir units to behave as dynamical systems with different time-scale dynamics. Supporting multiple time-scales is equivalent to a reservoir having individual neurons with different contractive dynamics. In this paper we implement this possibility by controlling the reservoir recurrent weight matrix $\hat{\mathbf{W}}$. As said in Section 3, the contractive dynamics of the reservoir is governed by the contraction coefficient of the state transition function, which in the Euclidean norm is $\sigma = \|\hat{\mathbf{W}}\|_2$. However, individual neurons may exhibit different contractive dynamics if $\hat{\mathbf{W}}$ is arranged in a proper way. In particular, we study the case in which $\hat{\mathbf{W}}$ is a diagonal matrix. In this case $\sigma = \max_{i=1,\ldots,N_R} \sigma_i$, where $\sigma_i$ is the contraction coefficient of each single unit $i$. This situation is equivalent to a RNN with self-recurrent connections only in the hidden layer. If different self-recurrent weights are used, then each neuron is able to show different contractive dynamics while being driven by the same input sequence. The Markovian behavior of the ESN does not only depend on the global value of $\sigma$ but on the whole set of dynamics determined by the single values $\sigma_i, \ \forall i = 1, \ldots, N_R$. We conjecture that this architectural factor is crucial to achieve a rich dynamics of the global transition function corresponding to good predictive performance.

**Non-linear Interactions among Units.** This factor refers to the presence of non-linear interactions among unit activations in a reservoir. It is implemented by using a $\hat{\mathbf{W}}$ matrix with non-zero extra-diagonal values, i.e. the reservoir units are mutually connected. Note that using mutually and self-recurrent connectivity (including also non-zero diagonal values), the different units dynamics due to the multiple time-scale factor are enriched by the recurrent non-linear combination of the dynamics of each unit. In this paper we consider both the cases of a dense recurrent matrix (fully connected reservoir) and a sparse recurrent matrix (sparsely connected reservoir). Note that since the very first work on ESNs [16] this architectural factor is one of those that are claimed to be responsible for producing "rich" reservoir dynamics, and thus good predictive performance. We propose to study this point in comparison to the other three identified and to show how it really influences the performance of reservoirs. Moreover, through a comparison between the sparsely and the fully connected architectures we intend to investigate if a full connectivity among neurons is actually needed to observe an influence of the non-linear interactions among units factor or if a small number of recurrent connections among neurons is sufficient.

**Regression in a High Dimensional Feature Space.** This factor refers to the influence that having a high dimensional reservoir may have in the predictive accuracy of an ESN. This last point has actually much to do with the readout part of ESNs. The conjecture here is that a linear readout might perform much better in a high dimensional reservoir state space. To assess and to measure the effect of this factor we introduce a particular reservoir variant which is called $\varphi$-ESN in which a smaller number of recurrent reservoir units (thus a smaller number of network recurrent dynamics) is projected into an higher dimensional space by a non-linear mapping. This study should give some insight on how much of the goodness of ESN performance is due to a sufficient number of network dynamics and how much of it is due to the augmented possibility of discriminating input patterns in a high dimensional feature space (even with a linear readout).

We propose to investigate how the identified factors influence the reservoir dynamics by studying several architectural variants on the main ESN model of equation (9) (see Figure 1). In particular, we consider the following architectures:

- **ESN *full***. This corresponds to the basic ESN model described in Section 3, where $\hat{\mathbf{W}}$ is characterized by full connectivity.
- **ESN *sparse***. As above, but $\hat{\mathbf{W}}$ is a sparse matrix. This distinction is useful to study the effects of non linear interactions among reservoir units even in presence of a very small number of such interactions.
- **DESN**[3]. Stands for Diagonal Echo State Network, in which $\hat{\mathbf{W}}$ is a diagonal matrix whose diagonal entries $w_{11}, w_{22}, \ldots, w_{N_R N_R}$ are all the same

[3]In literature the acronym DESN is also used to refer a different approach called Decoupled ESN model [43].

and such that $|w_{ii}| = \sigma, \quad \forall i = 1, 2, \ldots, N_R$ (see Figure 2). In this case $\sigma$ is also equal to the spectral radius of $\hat{\mathbf{W}}$. A particular case is when $w_{ii} = \sigma, \quad \forall i = 1, \ldots, N_R$, that corresponds to a reservoir having only positive self-recurrent connections:

$$x_i(n) = f(\mathbf{W}_{in}\mathbf{u}(n) + \sigma x_i(n-1)) \ \ \forall i = 1, 2, \ldots, N_R \tag{18}$$

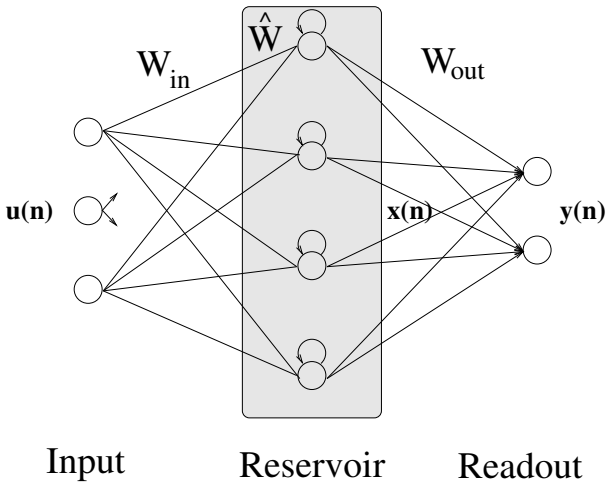


**Figure 2:** Diagonal ESN architectures with a number of $N_U = 3$ input units, $N_R = 4$ reservoir units and $N_O = 2$ output units. The same recurrent weight is shared by every reservoir unit for a DESN, while different recurrent weights are possible in a RDESN.

The individual neurons of a DESN implement dynamical systems which are all governed by the same contractivity properties (the same Markovianity), but which may look at the same input sequence through different input weights depending on the presence or absence of input variability. In particular, in the absence of input variability, all neurons in the reservoir of a DESN are identical, and the dynamics of a DESN is the same as the dynamics of a single neuron reservoir. If input variability is added, every reservoir neuron may differentiate its dynamics and thus performance prediction is expected to be better in this latter case.

- **RDESN**. Stands for Random Diagonal Echo State Network, which consists in a DESN with possibly different self-recurrent weights. If $\sigma$ is the contraction coefficient of the reservoir, then the diagonal weights of $\hat{\mathbf{W}}$ are chosen according to a uniform distribution in $[-\sigma, \sigma]$ (see Figure 2).

$$x_i(n) = f(\mathbf{W}_{in}\mathbf{u}(n) + w_{ii}x_i(n-1)) \ \ \forall i = 1, 2, \ldots, N_R \tag{19}$$

Also in this case the contractivity coefficient $\sigma$ is equal to the spectral radius of $\hat{\mathbf{W}}$. The reservoir of a RDESN has $N_R$ self recurrent units, each of which is characterized by contractive dynamics with different timescales. This variant is mainly used to evaluate the influence that multiple time-scale dynamics might have on the model performance, in particular when compared with the DESN architecture. An architectural variant

very similar to this one is known in ESN literature as the Simple ESN (SESN) model [7]. However, in the basic SESN setting the input weights to the reservoir are always fixed to 1.0 and the recurrent units are typically linear.

- **$\varphi$-ESN**. This variant accounts for the last architectural factor. The reservoir state space is projected into a higher dimensional feature space through a non-linear random mapping. The encoding function $\hat{\tau}$ computed by the reservoir of the ESN is further decomposed into an encoding function $\hat{\tau}'$ and a feature mapping $\varphi$:

$$\hat{\tau} = \varphi \circ \hat{\tau}' \tag{20}$$

where $\varphi : \mathbb{R}^{N_R} \rightarrow \mathbb{R}^{N_\varphi}$, and $\mathbb{R}^{N_\varphi}$ is the new augmented reservoir state space. The reservoir is subdivided into a recurrent part, which implements $\hat{\tau}'$ and acts as a standard dynamic reservoir, and a feed-forward (static) part, which implements the function $\varphi$:

$$\varphi(\mathbf{x}(n)) = f_\varphi(\mathbf{W}_\varphi \mathbf{x}(n)) \tag{21}$$

where $\mathbf{W}_\varphi \in \mathbb{R}^{N_\varphi \times N_R + 1}$ is the weight matrix (plus the bias) for the connections from the recurrent part to the feed-forward part of the reservoir. $\mathbf{W}_\varphi$ is set with random values in a bounded range. In our settings, the hyperbolic tangent is used as non-linear activation function $f_\varphi$. As the reservoir state space is now $\mathbb{R}^{N_\varphi}$, the matrix $\mathbf{W}_{out}$ of equation (9) is a $N_Y \times N_\varphi + 1$ matrix. Figure 3 illustrates the architecture of a $\varphi$-ESN.

Note that the recurrent part of a $\varphi$-ESN may be arranged according to one of the previously described architectures. We can thus have $\varphi$-ESN *full*, $\varphi$-ESN *sparse*, $\varphi$-DESN and $\varphi$-RDESN.

# 6 Experimental Results

The experiments presented in the following aimed at testing the empirical effects of the architectural factors introduced in Section 4 and 5. Firstly, in Section 6.2, we use two tasks to show the condition underlying the ESN state space organization, i.e. the Markovian assumption. Under such extreme condition we show that the Markovian factor dominates the behavior of the model and then complex architectures are even not necessary. In particular, the Markovian task is designed to be an easy task that can be solved by a 1-unit model, while the anti-Markovian task is conceived to be an hard task independently of the architectural design.
Then, in Section 6.2.2, we use two well-known tasks (Mackey-Glass and NARMA system) for which large ESN architectures are useful to achieve high predictive accuracies (showing that 1-unit is not sufficient independently of the value of the contraction coefficient).
On these two last tasks, where the architectural design has a relevant role, in Section 6.3 we show and evaluate the progressive positive influence on the ESN performance of the architectural factors introduced in Section 5. For the sake of completeness we tested the effects of the architectural factors also on the first

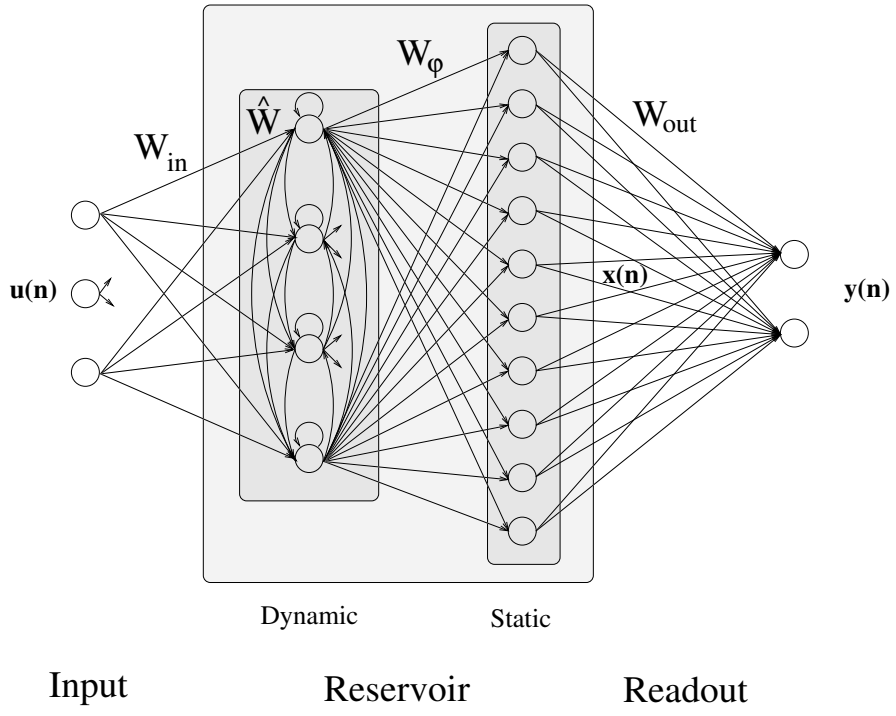


**Figure 3:** Architecture of a $\varphi$-ESN, with a number of $N_U = 3$ input units and $N_O = 2$ output units. The dynamic part of the reservoir contains $N_R = 4$ recurrent units, while the static part is implemented through a feed-forward layer of $N_\varphi = 10$ feed-forward units.

two task (Markovian/anti-Markovian), allowing us to provide a further support to the result of Section 6.2 on the major role of the Markovian factor.

The next Section introduces details and experimental conditions on the tasks used in all the experiments.

## 6.1 Tasks

We considered four tasks. The first two are prediction tasks on a symbolic sequence domain, where the targets are of a distinct Markovian and anti-Markovian nature, respectively. The other tasks are one-step prediction tasks on the Mackey-Glass chaotic time series and on a 10-th order non-linear NARMA system.

To evaluate the performance we computed the mean squared and the standard deviation of errors. The mean squared error (measure) is computed as follows:

$$E = \frac{1}{N}\sum_{n=1}^{N}(y(n) - \hat{y}(n))^2 \tag{22}$$

where $y(n)$ and $\hat{y}(n)$ respectively denote the actual output of the model and the target one for pass $n$, while $N$ is the number of samples on which the error function is evaluated. A number of $N_{trials} = 10$ independent repetitions[4] of each experiment has been carried out and the average results are reported in the following.

The following subsections describe the considered tasks more in detail.

### 6.1.1 Markovian/Anti-Markovian Symbolic Sequences

For this task we considered symbolic sequences on an input alphabet of 10 symbols, $\mathcal{A} = \{a, \ldots, j\}$. Each element in a sequence was selected according to a uniform distribution over $\mathcal{A}$. The length of the sequence $s$ is denoted by $|s|$. The symbols in $s$ are denoted by $s(1), s(2), \ldots, s(|s|)$, where $s(1)$ is the oldest symbol and $s(|s|)$ is the most recent one. A mapping function $M : \mathcal{A} \to \{0.1, \ldots, 1.0\}$ is defined, such that $M(a) = 0.1, M(b) = 0.2, \ldots, M(j) = 1.0$. Hence, each input element is defined by $u(n) = M(s(n))$. On such sequences we defined two kind of sequence-to-element tasks with a Markovian nature.
A first kind of Markovian tasks is obtained by associating to each sequence a target defined by the equation

$$\hat{y}(s) = \sum_{n=1}^{|s|}\frac{u(n)}{\lambda^{|s|-n}} \tag{23}$$

where $\lambda > 1$ is a real number that controls the degree of Markovianity of the task. Indeed the target value associated to a string $s$ depends on each symbol $s(n)$, weighted by a value which exponentially decreases with decreasing $n$, i.e. recent entries have a greater influence on the target than older ones. The distance among target values reflects the length of common suffix of the input sequences, i.e. sequences with similar suffixes lead to similar target values, whereas the closeness of targets is proportional to length of the similar suffix. Hence, this task is an instance in the class of Markovian processes (in particular

[4]This number turned out to be sufficient as explained at the beginning of Section 6.3.

those related to positive feedback systems). It would provide an example of easy task for ESNs, for which we expect an excellent performance. Every target value obtained by applying the equation (23) was then rescaled to the range $[-1, 1]$.
The second task is an anti-Markovian task, in which the target value associated to $s$ is computed as

$$\hat{y}(s) = \sum_{n=1}^{|s|} \frac{u(n)}{\lambda^{n-1}} \tag{24}$$

where again $\lambda > 1$ controls the degree of Markovianity, but in this case the weight associated to each input entry is greater for older entries and smaller for more recent ones. It is designed to be an hard task for ESNs, in the sense that we expect that ESNs perform quite bad on it. Also for this task, the target values were rescaled to the range $[-1, 1]$.
A number of $N_{train} = 500$ and $N_{test} = 100$ input sequences, of length between 50 and 100, were used for training and testing, respectively, while each input sequence was fed three consequently times to the networks to account for the transient. We used $\lambda = 2$ for the Markovian task and $\lambda = -2$ for the anti-Markovian one.

### 6.1.2 Mackey-Glass Time Series

The Mackey-Glass time series [25] is a standard benchmark for chaotic time series prediction models, on which ESNs have been successfully applied (e.g. [16][21]) showing a very good performance. It is defined by the following differential equation:

$$\frac{\partial u(t)}{\partial t} = \frac{0.2u(t-\alpha)}{1 + u(t-\alpha)^{10}} - 0.1u(t) \tag{25}$$

The most used values for $\alpha$ are 17 and 30, where for $\alpha > 16.8$ the system has a chaotic attractor. We used $\alpha = 17$. The task consists in a next-step prediction of a discrete version of the equation (25).
For each repetitions of the experiments, we generated 10000 time steps of the series, of which the first $N_{train} = 5000$ were used for training and the last $N_{test} = 5000$ were used for testing. An initial transient of $N_{transient} = 1000$ time steps was discarded before training the readout. Every element of the Mackey-Glass sequence was shifted by $-1$ and passed through the *tanh* function as in [16, 21].

### 6.1.3 10th Order NARMA System

This task consists in predicting the output of a 10-th order non-linear autoregressive moving average (NARMA) system. The task has been introduced in [1] and has been tackled by ESN models in [17] and [4]. The input of the system is a sequence of elements $u(n)$ randomly chosen according to a uniform distribution over $[0, 0.5]$. The output of the target system is computed as:

$$\hat{y}(n) = 0.3\hat{y}(n-1) + 0.05\hat{y}(n-1)(\sum_{i=1}^{10} \hat{y}(n-i)) + 1.5u(n-10)u(n-1) + 0.1 \tag{26}$$

Given the input value $u(n)$, the task is to predict the corresponding value of $\hat{y}(n)$. The training set was made up of $N_{train} = 2200$ input-target examples, of which $N_{transient} = 200$ were used as initial transient. A sequence of length $N_{test} = 2000$ was used for testing.

## 6.2 Markovian Factor

We tested ESNs with one single (self-recurrent) neuron versus ESNs with a number of 100 reservoir units ($N_R = 1$ or 100) and contractive dynamics ruled by the same value of $\sigma$ ranging in the interval $[0.1, 0.9]$ with a step size of 0.1. Reservoirs with multiple neurons posses a full connectivity, and are initialized according to Section 3.3. In particular, weight values in $\hat{\mathbf{W}}$ are drawn from a uniform distribution over $[-1, 1]$ and then $\hat{\mathbf{W}}$ is rescaled to the specific contraction coefficient $\sigma$. Reservoirs with one single unit are actually unidimensional DESNs, where for the Markovian sequences task the setting is done according to the simplest case, i.e. equation (18), and using a linear activation function. Weights for the input-to-reservoir connections were initialized according to a uniform distribution over $[-0.1, 0.1]$.

### 6.2.1 Markovian/Anti-Markovian Tasks

Figure 4 shows the results of this experiment obtained for the Markovian sequences task. It is evident that for the appropriate value of the contraction

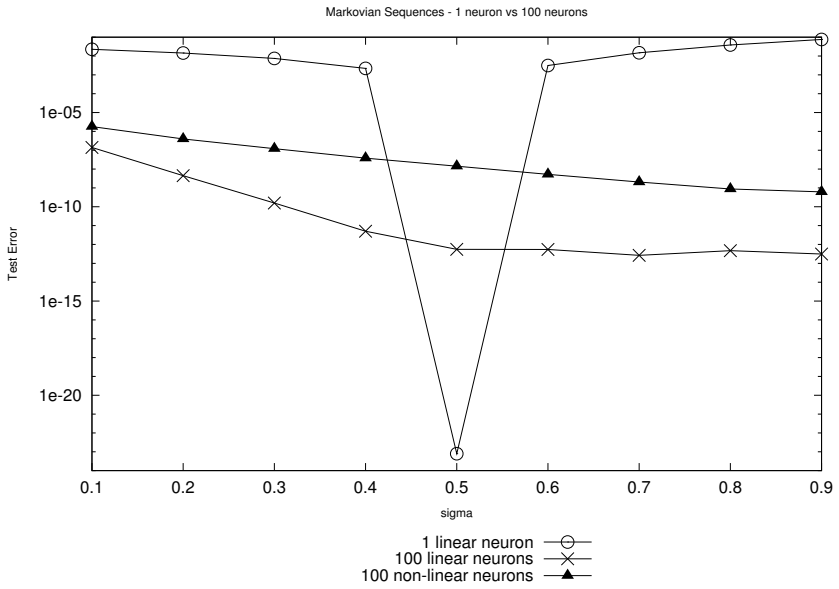


**Figure 4:** Mean squared test errors on the Markovian sequences task for reservoirs of 1 linear unit (DESN) and 100 units (ESN *full*), with linear and non-linear activation functions. Results are reported for increasing values of the contraction coefficient $\sigma$.

coefficient the single unit model with linear activation function is able to reproduce the target dynamics with almost no error, beating models with larger reservoirs. The point of best performance is actually due to the particular choice

of the parameter $\lambda = 2$ in the definition of the task (see equation (23)), which indeed corresponds to a value of $\sigma = 0.5$ in the linear unit dynamics. These results represent a strong evidence that for this particular task, with such a distinct Markovian nature, the Markovian factor is the dominant factor influencing the predictive performance of ESNs. In such a case more complex architectures are even not necessary. None of the architectural factor included in the ESN *full* could improve the ad-hoc solution fitted by the single neuron with appropriate setting of parameters. Note that, however, to obtain the best performance, the contraction coefficient of the neuron must match the right degree of Markovianity of the target system. If this does not happen, then other architectural factors may be useful to improve the performance. In fact, for values of the contraction coefficient other than 0.5, the single unit model is always outperformed by the full configuration of ESN with multiple units.

The results obtained for the anti-Markovian sequences task are reported in Figure 5. As expected, none of the tested architectures was able to solve this

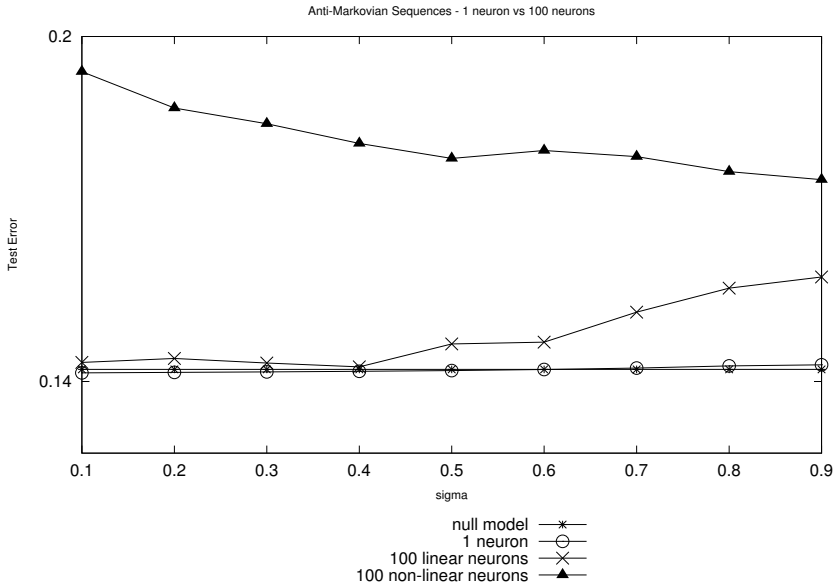


**Figure 5:** Mean squared test errors on the anti-Markovian sequences task for reservoirs with 1 unit and 100 units, with linear and non-linear activation functions. Results are reported for increasing values of the contraction coefficient $\sigma$. The error obtained by a null model is also reported for a further comparison.

task with a satisfactory accuracy. Through a comparison between Figures 4 and 5, it is indeed apparent that the error on the anti-Markovian sequences task is several orders of magnitude higher than the error on the Markovian sequences one. For a further comparison, we also reported in the same Figure the results obtained with the *null model*, i.e. a trivial model whose output is always equal to the mean target value of the training set. In particular, single unit models performed just as the null model for every choice of the contraction coefficient. Moreover, larger reservoirs led to even poorer results, especially for a non-linear activation function. None of the architectural factor included in the ESN *full*

could help to face the task. These observations further remark the importance of the Markovian factor, showing that tasks of a remarkable anti-Markovian kind cannot be solved with ESN models, independently of the choice of the contraction coefficient and of the other architectural details.

### 6.2.2 Mackey-Glass and NARMA System Tasks

Figures 6 and 7 provide the results for the Mackey-Glass and the NARMA system tasks for the 1 unit versus 100 units setting.

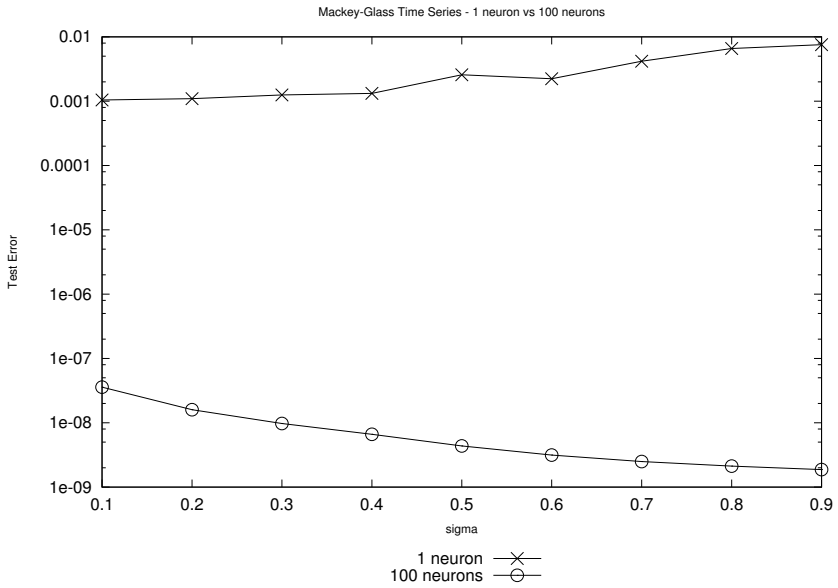


**Figure 6:** Mean squared test errors on the Mackey-Glass task for reservoirs of 1 unit and 100 units (ESN *full*), with non-linear activation function. Results are reported for increasing values of the contraction coefficient $\sigma$.

As results show, independently of the contraction coefficient, on both these tasks the multiple neurons architecture outperformed the single neuron one. Even though the Markovian factor continues to characterize reservoir dynamics, it is evident that larger architectures can exploit the increased dimensionality to obtain better performances, especially for large value of sigma.

### 6.2.3 Remarks on Markovian Factor Results

As explained in Section 4, contractivity is the key feature of valid reservoirs of neurons and allows ESNs to solve Markovian tasks even without training the recurrent connections. However, the results presented in this Section point out that just contractivity is not enough. Indeed, for (simple) tasks with a distinct Markovian nature even one single contractive neuron (with linear activation function) is able to outperform a larger reservoir. In particular, this is possible when the Markovian dynamics of the target process is known and can thus be reproduced in the neuron. However, for highly non-linear tasks, which do not obey a predefined Markovian rule, single contractive neurons turned out to be

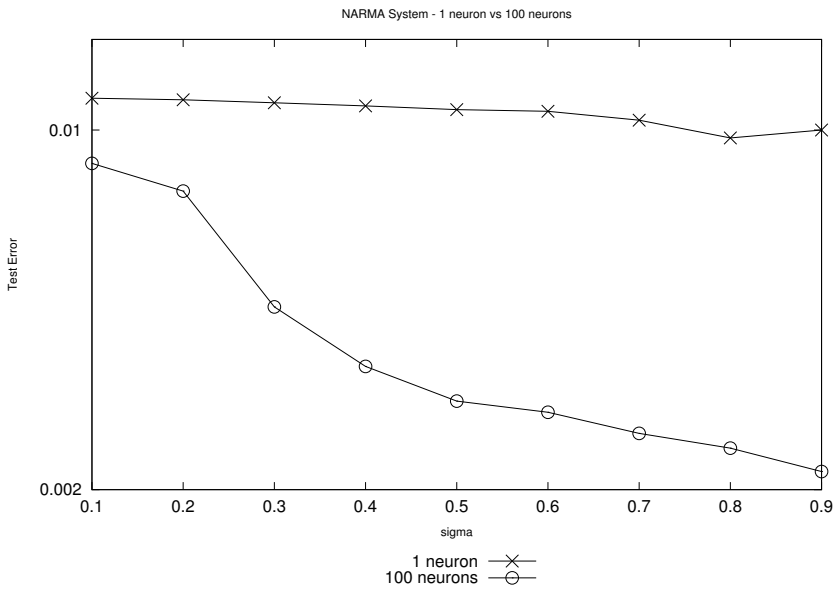


**Figure 7:** Mean squared test errors on the NARMA system task for reservoirs of 1 unit and 100 units, with non-linear activation function. Results are reported for increasing values of the contraction coefficient $\sigma$.

inferior to larger reservoir organized as in standard ESNs. In these cases the Markovian factor turn out to be not sufficient to completely characterize the behavior of the ESN. The problem is then in understanding the features that make a reservoir of neurons with fixed dynamics able to successfully solve highly non-linear tasks. In Section 5 we identified four possible key architectural factors that may improve the performance of fixed reservoirs. The results concerning the architectural factors are reported in the next subsection.

## 6.3 Architectural Factors

We tested the proposed architectural variants on the same benchmark datasets introduced in Section 6.1, using the same experimental framework of Section 6.2. The reported results present the performance of the tested models for increasing reservoir dimensionality, allowing us to show the progressive effect of differentiation introduced among units due to each architectural factor. The reservoir dimensionality considered varied in $N_R = 1, 10, 100, 300, 500$, while for the $\varphi$-ESN model we used a number of recurrent reservoir units varying as $N_R = 1, 5, 10, 100, 200$, with a projection into a $N_\varphi = 500$ dimensional feature space. Weight values in matrix $\mathbf{W}_\varphi$ were randomly chosen according to a uniform distribution over $[-1.0, 1.0]$. For ESNs *full* and ESNs *sparse*, the recurrent reservoir weight matrix $\hat{\mathbf{W}}$ was initialized with random weight values according to a uniform distribution over $[-1, 1]$. For the case of sparse reservoirs, we used a percentage of connectivity equal to 5%. $\hat{\mathbf{W}}$ was then rescaled to the desired value of the contraction coefficient. The weights in matrix $\mathbf{W}_{in}$ were selected according to a uniform distribution in $[-0.1, 0.1]$. In reservoirs not support-

| | no input var. | input var. |
|---|---|---|
| **DESN** | $1.0619 \times 10^{-2}$ | $2.9994 \times 10^{-7}$ |
| **RDESN** | $1.0112 \times 10^{-8}$ | $1.5195 \times 10^{-9}$ |
| **ESN *full*** | $2.2556 \times 10^{-9}$ | $3.9408 \times 10^{-10}$ |
| **ESN *sparse*** | $2.8331 \times 10^{-9}$ | $4.2421 \times 10^{-10}$ |

**Table 1:** Mean squared test errors on the Mackey-Glass task for DESNs, RDESNs, ESNs *full* and ESNs *sparse*, with and without input variability. Errors are reported for a number of reservoir units of $N_R = 500$ and a constant value of the contraction coefficient $\sigma = 0.9$.

ing input variability, all weights in $\mathbf{W}_{in}$ were randomly initialized to the same value. For every task, we tested networks with the same value of the contraction coefficient $\sigma$. For the Mackey-Glass and the NARMA system tasks we set the value of $\sigma$ to 0.9 (which provide the best setting founded in Section 6.2.2). For the Markovian/anti-Markovian tasks a value of $\sigma = 0.5$ was used. In the next subsections we report the results of these experiments together with an analysis of the influence of the single architectural factor.

The predictive performance of DESNs, RDESNs and ESNs with and without input variability is provided in Figures 8, 9, 10 and 11 for the Mackey-Glass, NARMA system, Markovian sequences and anti-Markovian sequences, respectively. Tables 1, 2, 3 and 4 report the averaged squared errors for the same tasks and with $N_R = 500$.

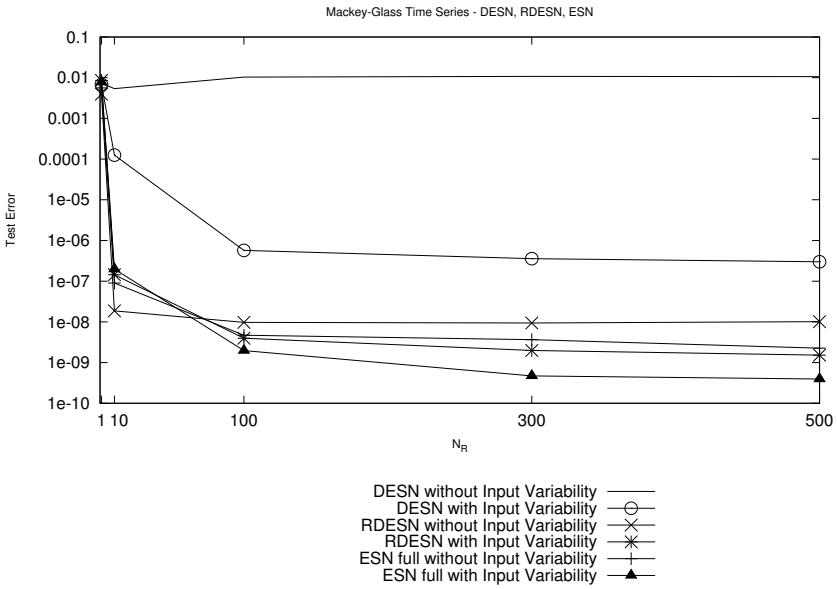


**Figure 8:** Mean squared test errors on the Mackey-Glass task for DESNs, RDESNs, ESNs *full*, with and without input variability. Errors are reported for increasing reservoir dimensionality and a constant value of the contraction coefficient $\sigma = 0.9$.

As a first general result, the graphs show that the test error initially dra-

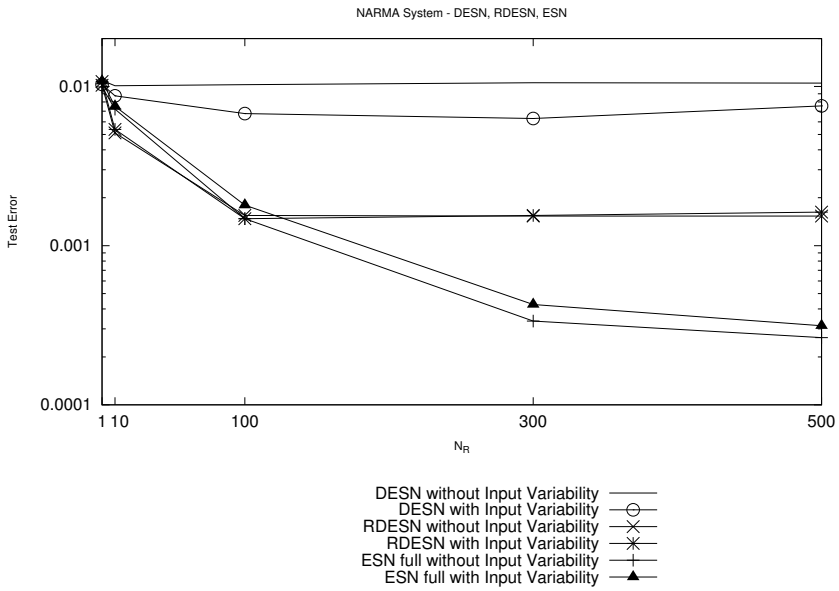


**Figure 9:** Mean squared test errors on the NARMA system task for DESNs, RDESNs, ESNs *full*, with and without input variability. Errors are reported for increasing reservoir dimensions and a constant value of the contraction coefficient $\sigma = 0.9$.

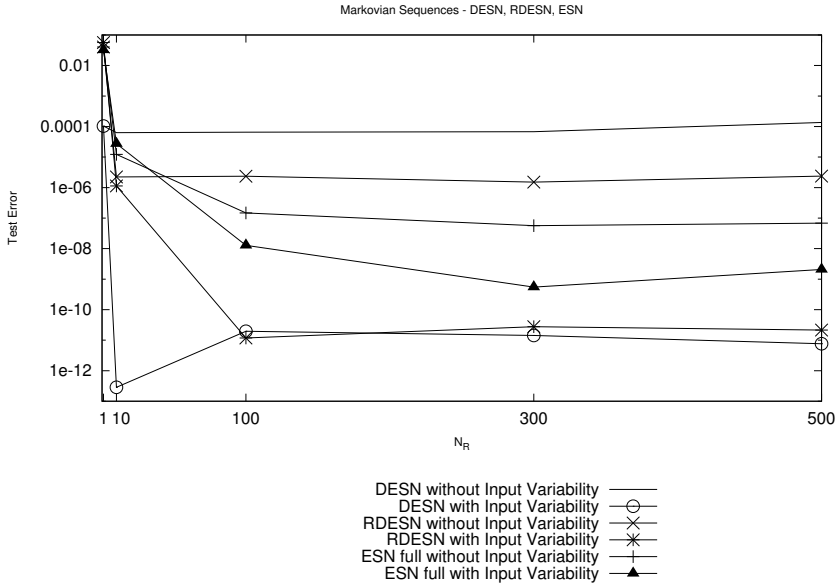


**Figure 10:** Mean squared test errors on the Markovian sequences task for DESNs, RDESNs, ESNs *full*, with and without input variability. Errors are reported for increasing reservoir dimensions and a constant value of the contraction coefficient $\sigma = 0.5$.

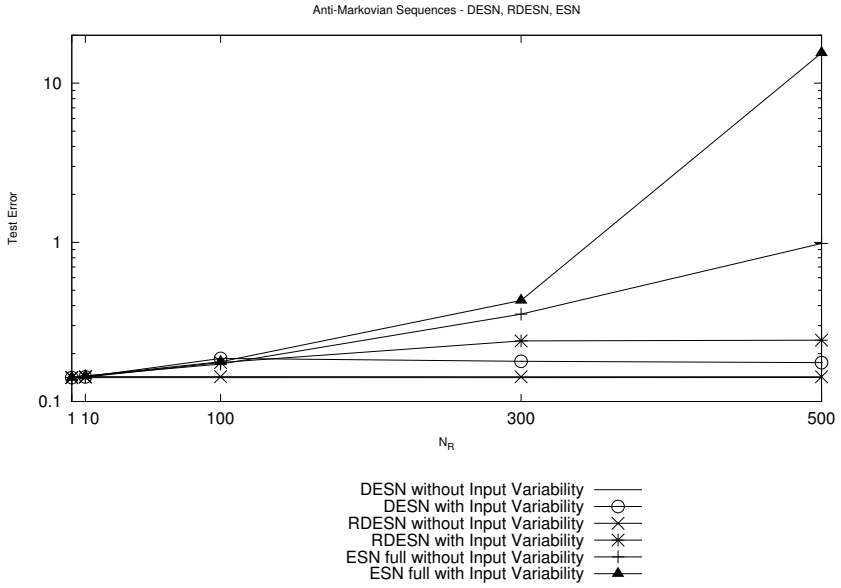


**Figure 11:** Mean squared test errors on the anti-Markovian sequences task for DESNs, RDESNs, ESNs *full*, with and without input variability. Errors are reported for increasing reservoir dimension and a constant value of the contraction coefficient $\sigma = 0.5$.

| | **no input var.** | **input var.** |
|---|---|---|
| **DESN** | $1.0503 \times 10^{-2}$ | $7.5553 \times 10^{-3}$ |
| **RDESN** | $1.5352 \times 10^{-3}$ | $1.6236 \times 10^{-3}$ |
| **ESN *full*** | $2.6440 \times 10^{-4}$ | $3.1413 \times 10^{-4}$ |
| **ESN *sparse*** | $2.8160 \times 10^{-4}$ | $3.2208 \times 10^{-4}$ |

**Table 2:** Mean squared test errors on the NARMA system task for DESNs, RDESNs, ESNs *full* and ESNs *sparse*, with and without input variability. Errors are reported for a number of reservoir units of $N_R = 500$ and a constant value of the contraction coefficient $\sigma = 0.9$.

| | **no input var.** | **input var.** |
|---|---|---|
| **DESN** | $1.3656 \times 10^{-4}$ | $7.6448 \times 10^{-12}$ |
| **RDESN** | $2.3728 \times 10^{-6}$ | $2.1435 \times 10^{-11}$ |
| **ESN *full*** | $6.8621 \times 10^{-8}$ | $2.0681 \times 10^{-9}$ |
| **ESN *sparse*** | $9.0994 \times 10^{-8}$ | $1.6285 \times 10^{-9}$ |

**Table 3:** Mean squared test errors on the Markovian sequences task for DESNs, RDESNs, ESNs *full* and ESNs *sparse*, with and without input variability. Errors are reported for a number of reservoir units of $N_R = 500$ and a constant value of the contraction coefficient $\sigma = 0.5$.

| | no input var. | input var. |
|---|---|---|
| **DESN** | $1.4149 \times 10^{-1}$ | $1.7555 \times 10^{-1}$ |
| **RDESN** | $1.4289 \times 10^{-1}$ | $2.4279 \times 10^{-1}$ |
| **ESN *full*** | $9.8374 \times 10^{-1}$ | $1.5496 \times 10$ |
| **ESN *sparse*** | 1.0527 | 9.4444 |

**Table 4:** Mean squared test errors on the anti-Markovian sequences task for DESNs, RDESNs, ESNs *full* and ESNs *sparse*, with and without input variability. Errors are reported for a number of reservoir units of $N_R = 500$ and a constant value of the contraction coefficient $\sigma = 0.5$.

matically decreased[5] as the number of units in the reservoir increased, i.e. along with the increasing diversification on the reservoir dynamics induced by the architectural factors. (An expected exception is due to the anti-Markovian task where no choice of architectural factors can satisfactorily solve the task and the more complex models overfitted the data). Note also that the influence of the factors led to a saturation effect approaching the maximum number of units.
As expected, for the DESN model without input variability (corresponding to a single unit dynamics) we found the same error value for every reservoir dimensionality, except for small statistical fluctuations due to the random nature of the experimental settings.
The variance of the test errors over the 10 repetitions of each experiment was found to be not significant and is not reported in detail. For instance, for the Mackey-Glass task the variance ranged from a minimum of $5.1732 \times 10^{-22}$ (in correspondence of an averaged squared error of $3.9408 \times 10^{-10}$) for $N_R = 500$, to a maximum of $1.0543 \times 10^{-4}$ (corresponding to an averaged squared error of $7.2033 \times 10^{-3}$) for $N_R = 1$.

The discussion on the effect of each architectural factor is detailed in the following basing on the order introduced in Section 5. The comparison between the predictive performance of ESNs with full connectivity and ESNs with sparse connectivity is discussed in Section 6.3.3. For this reason, the performance of ESN *sparse* is not reported in the comparative Figures in this Section.

### 6.3.1 Input Variability

The effect of the input variability factor on the predictive performance can be firstly evaluated by a comparison between the test errors of the DESN model with and without input variability. Results show that the presence of input variability improved the predictive performance of the DESN model for Mackey-Glass, NARMA system and Markovian sequences tasks. In particular, for the Mackey-Glass task, for $N_R = 500$, this architectural factor alone was able to enhance the predictive performance of five orders of magnitude (see Table 1). The improvement is also noteworthy for the Markovian sequences task (Table 3), while it is much more limited for the NARMA system one (Table 2). On the other hand, for the anti-Markovian sequences task we observed a negative influence of this architectural factor, which in fact led to worse results (Table 4).

As a second aspect, the impact on the performance due to input variability in presence of other architectural factors (RDESNs and ESNs with and without

[5]Note that the test errors in the plots are reported in logarithmic scale.

input variability) is reduced with respect to the absolute amount of the improvement for the DESN model. However, it still provides an improving effect on the performance, as detailed in the following subsections.

Note that the best result on the Markovian sequences task is obtained by the DESN model (initialized according to the setting of equation (18)) with input variability. This remarks the importance and prevalence of this architectural factor on this last task. For the specificity of the task (and especially because of the linearity of the target), we found that linear variants of the models proposed outperformed the non linear ones. This is illustrated in Figure 12 and Table 5. From a comparison between Tables 3 and 5, we can see that for every variant the linear versions of the model beat the non-linear counterparts. In particular it is very striking the result obtained with DESN without input variability. Indeed this architecture functionally correspond to one single neuron with a fixed self-recurrent weight and a fixed input weight. This model outperformed every other model with a greater number of functionally different neurons, except for the DESN variant with input variability. As said in Section 6.2, the reason for this great performance is in the fact that the dynamics of the target to be learned is actually very well fitted by the dynamics of one single neuron if its parameters are properly set.

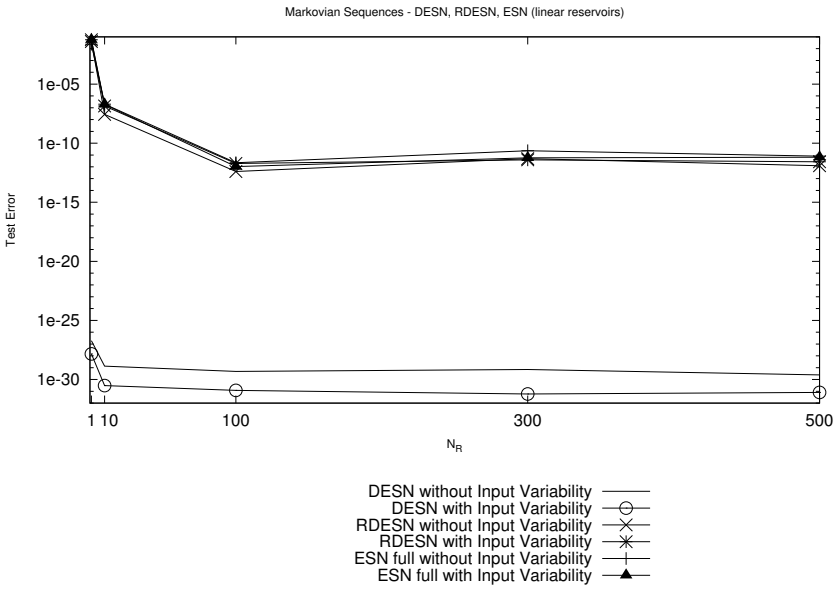


**Figure 12:** Mean squared test errors on the Markovian sequences task for ESN variants with linear activation function. Errors are reported for increasing reservoir dimension and a constant value of the contraction coefficient $\sigma = 0.5$.

For the anti-Markovian sequences task, by contrast, input variability increased the test errors for both RDESNs and ESNs (Figure 11). Moreover, note that the performance prediction on the anti-Markovian sequences task worsened for every choice of the architectural ESN variant. ESN models were not able to satisfactorily face this task, as expected. The averaged test errors increased as the reservoir dimensionality was increased, and mean error values greater than

| | **no input var.** | **input var.** |
|---|---|---|
| **DESN** | $2.4661 \times 10^{-30}$ | $8.0326 \times 10^{-32}$ |
| **RDESN** | $1.2328 \times 10^{-12}$ | $2.6542 \times 10^{-12}$ |
| **ESN *full*** | $8.0427 \times 10^{-12}$ | $6.4403 \times 10^{-12}$ |
| **ESN *sparse*** | $8.0515 \times 10^{-12}$ | $2.5100 \times 10^{-12}$ |

**Table 5:** Mean squared test errors on the Markovian sequences task for linear variants of DESN, RDESN, ESN *full* and ESN *sparse*, with and without input variability. Errors are reported for a number of reservoir units of $N_R = 500$ and a constant value of the contraction coefficient $\sigma = 0.5$.

four (corresponding to output values out of the target range) were found for $N_R = 500$, which is a clear signal that the readout overfitted the training data (see Table 4).

### 6.3.2 Multiple Time-Scale Dynamics

The influence of multiple time-scales dynamics can be observed by comparing the performance of DESN, which does not support it, with the performance of RDESN, which does. Again, an inspection of Figures 8, 9 and 10, and Tables 1, 2 and 3 reveals that the presence of multiple time-scales support is helpful for reservoirs.
To isolate the contribution given by this architectural factor we can focus on the difference between DESN without input variability and RDESN without input variability. For $N_R = 500$ the improvement was of almost six orders of magnitude for the Mackey-Glass task (Table 1), of one order of magnitude for the NARMA system task (Table 2) and of two orders of magnitude for the Markovian sequences task (Table 3). The contribution given by this factor alone is thus superior to the one brought about by input variability alone, at least for the Mackey-Glass and the NARMA system tasks. On the other hand, for the Markovian sequences task the improvement due to input variability alone outperformed the improvement due to multiple time-scales alone.

Moreover, note that the combined effect of both the factors together can lead to better performances. This is visible for the Mackey-Glass task (see Table 1), for which the RDESN with input variability variant led to an increase in performance of almost seven orders of magnitude with respect to DESN without input variability, of two orders of magnitude with respect to DESN with input variability and of one order of magnitude with respect to RDESN with no input variability (for a reservoir dimensionality of $N_R = 500$). We can say that for the Mackey-Glass dataset the interaction between input variability and multiple time-scales actually combined the single factor improvements, and led to a performance which is worse than the best recorded (ESNs *full* with input variability) of less than one order of magnitude. For the NARMA system task the results of RDESNs with or without input variability were the same (Table 2), suggesting that in this case supporting both multiple time-scales and input variability was actually equivalent as supporting multiple time-scales only. For the Markovian sequences task we found an appreciable effect of the combination of the two architectural factors. In fact, the performance of the RDESN model with input variability was five orders of magnitude better than the performance of the same model without input variability. However, the combination of input

variability and multiple time-scales led to a result which was worse than what obtained with input variability only (Table 3).
For the anti-Markovian task, the RDESN architecture without input variability performed like the DESN model without input variability, which correspond to the best performance obtained for this task. The interaction between input variability and multiple time-scales support worsened the result a little more than input variability alone (Table 4).

As a further remark, note that DESN and RDESN models provide dynamics originating from a set of single neurons dynamics only. Even though the global reservoir dynamics is ruled by a unique value of the contraction coefficient, the variety introduced by input variability and multiple time-scales dynamics for increasing reservoir dimension was able to differentiate the single Markovian dynamics. This actually resulted in an enrichment of the reservoir dynamics sufficient to produce the significant performance improvement observed here and in Section 6.3.1.

### 6.3.3 Non-linear Interactions Among Units

We investigated the performance improvement obtained by introducing non-linear interactions among reservoir neurons. As for the other architectural factors, we found that the presence of non-linear interactions among neurons enhanced the predictive performance of the tested models on the Mackey-Glass, NARMA system and Markovian sequences tasks, while made it worse on the anti-Markovian one.
In absence of the input variability factor, for $N_R = 500$, the improvement with respect to the DESN model without input variability was of seven orders of magnitude for the Mackey-Glass time series (Table 1), of two orders of magnitude for the NARMA System (Table 2) and of almost four orders of magnitude for the Markovian sequences task (Table 3). These results show that non-linear interactions among reservoir units turned out to be a very effective factor for the case of the NARMA system task (Figure 9), while its role was less crucial for the Mackey-Glass one (Figure 8). On this last task, in fact, its impact was similar to the one brought about by the multiple time-scales factor.

The combination of input variability and non-linear interactions led to a performance improvement of almost one order of magnitude for both the Mackey-Glass (Table 1) and the Markovian sequences tasks (Table 3). No appreciable improvement due to this combination was found for the NARMA system dataset (Table 2). From Figure 11 it is clearly apparent that for the anti-Markovian task including non-linear interactions among neurons led to a worsening of the results. The combined effect of units interactions and input variability made the predictive performance even poorer.

Another remarkable fact is that ESNs with full connectivity and ESNs with sparse connectivity led to almost the same results for all the four tasks examined (see Tables 1, 2, 3, 4). Figure 13 illustrates the averaged test errors for ESN *full* and ESN *sparse* on the four tasks, with and without input variability. What is interesting is that even a relatively small number of interconnections inside a reservoir (5% in our experimental setting) had nearly the same impact on the predictive performance as the presence of connections among all neurons. Our results also support the idea that the original intuition of sparsity as a characteristic of reservoirs with rich dynamics is actually misleading. The

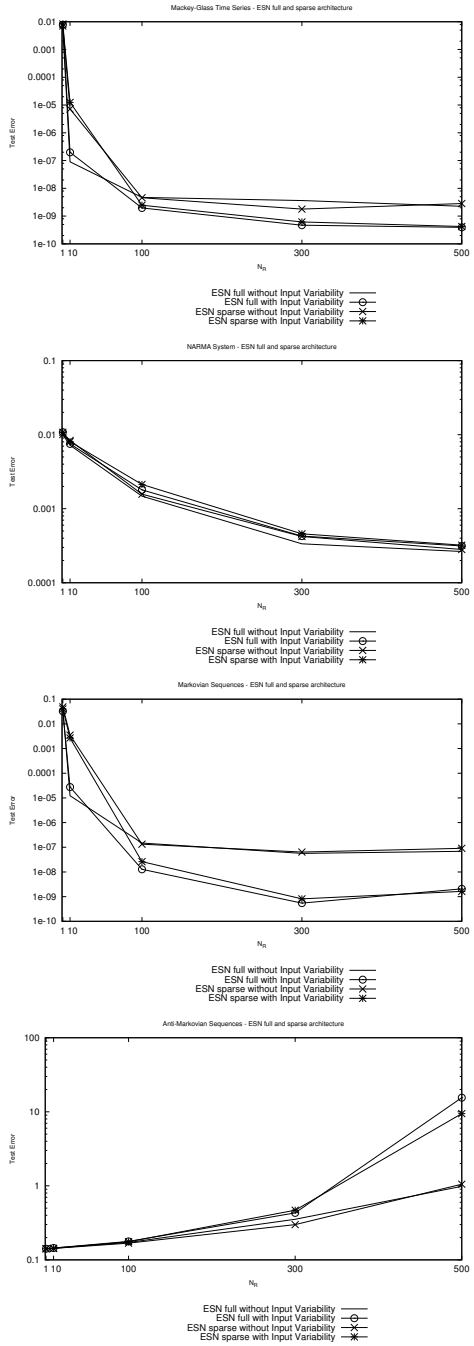


**Figure 13:** Averaged squared test errors for ESN *full* and ESN *sparse* architectures on the four tasks. Errors are reported for increasing reservoir dimension and a constant value of the contraction coefficient ($\sigma = 0.9$ for Mackey-Glass and NARMA System tasks, $\sigma = 0.5$ for Markovian and anti-Markovian sequences task).

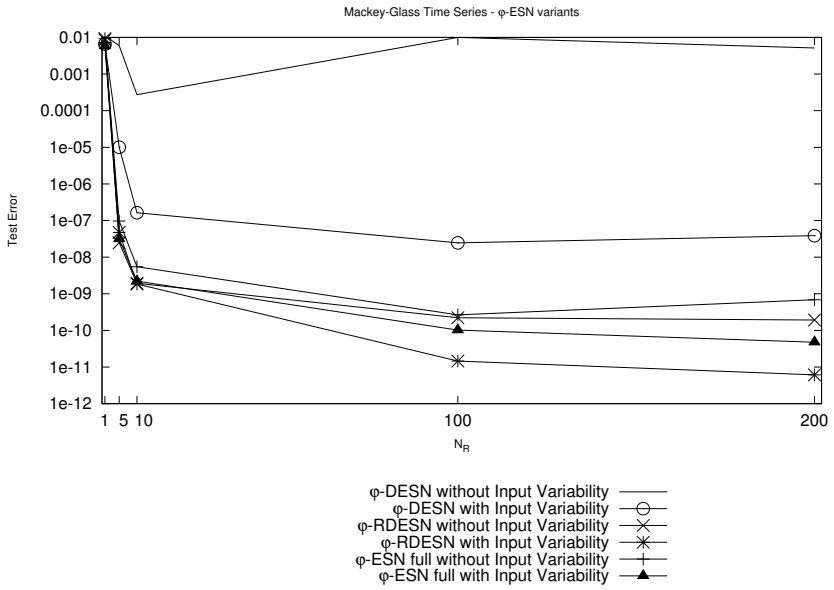


**Figure 14:** Averaged squared test errors for $\varphi$-ESN variants on the Mackey-Glass task. Errors are reported for increasing recurrent reservoir dimension, a constant dimension $N_\varphi = 500$ of the static part of the reservoir, and for a constant value of the contraction coefficient $\sigma = 0.9$.

effective enrichment of reservoir state dynamics is due to the presence of architectural factors which may produce diversification among units, as resulted by the presented experiments. The sparse setting of the reservoir weight matrix is only an efficient way to include such factors in the design of ESNs, but it is not responsible by itself for this enrichment.

### 6.3.4 Regression in a High Dimensional Feature Space

To investigate the benefit of having a high dimensional reservoir space for regression, we tested the $\varphi$-ESN architectures varying the model used for the recurrent part of the reservoir. Figures 14, 15, 16 and 17 provide the results for the four tasks.

The predictive performance of $\varphi$-ESN variants resulted to be sensible to the three other architectural factors described so far. The relevance of singular factors and of composition of factors on $\varphi$-ESN performance roughly reflected what was found for the ESNs variants in Sections 6.3.1, 6.3.2 and 6.3.3, with a few differences. For the Mackey-Glass dataset we found that RDESNs outperformed ESNs *full* for both the cases of presence and absence of input variability (Figure 14). Also observe that the $\varphi$-DESN variant with input variability overfitted training data for the NARMA system task (Figure 15).

What is probably more interesting is the comparison between the class of $\varphi$-ESN variants and the class of ESN variants (representable in the following by the variant providing the best result). As shown in Figures 18, 19, 20 and 21, and Tables 6 and 7, $\varphi$-ESN architectures compare well with standard ESNs. In fact, for the Mackey-Glass task, $\varphi$-ESN *full* and $\varphi$-RDESN with only 10 re-

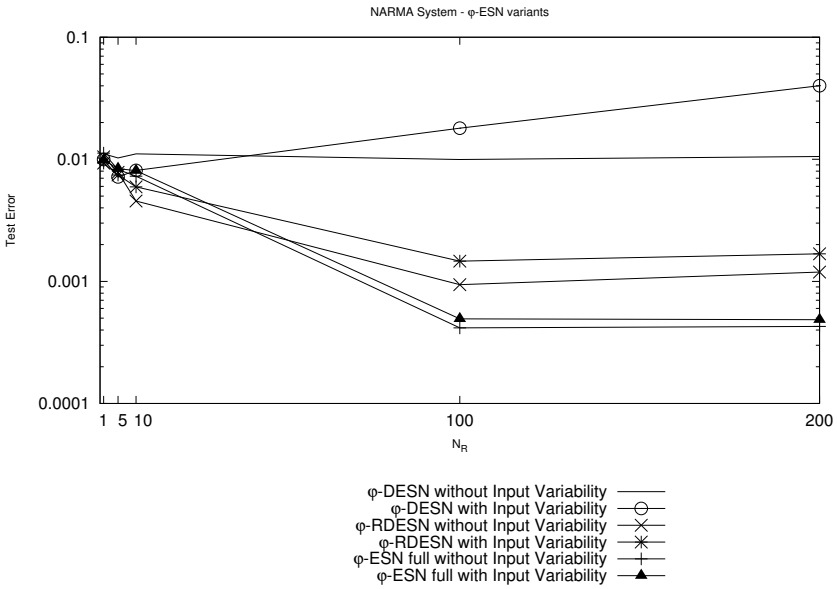


**Figure 15:** Averaged squared test errors for $\varphi$-ESN variants on the NARMA system task. Errors are reported for increasing recurrent reservoir dimension, a constant dimension $N_\varphi = 500$ of the static part of the reservoir, and for a constant value of the contraction coefficient $\sigma = 0.9$.

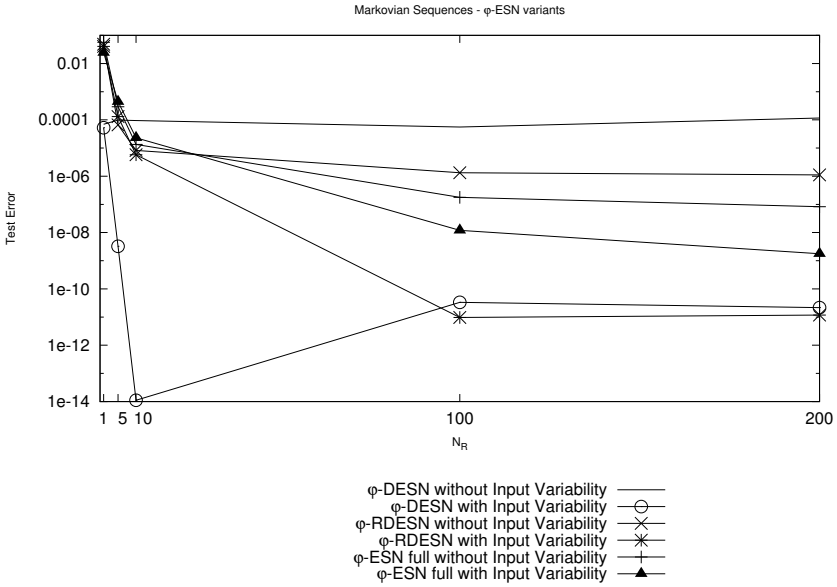


**Figure 16:** Averaged squared test errors for $\varphi$-ESN variants on the Markovian sequences task. Errors are reported for increasing recurrent reservoir dimension, a constant dimension $N_\varphi = 500$ of the static part of the reservoir, and for a constant value of the contraction coefficient $\sigma = 0.5$.

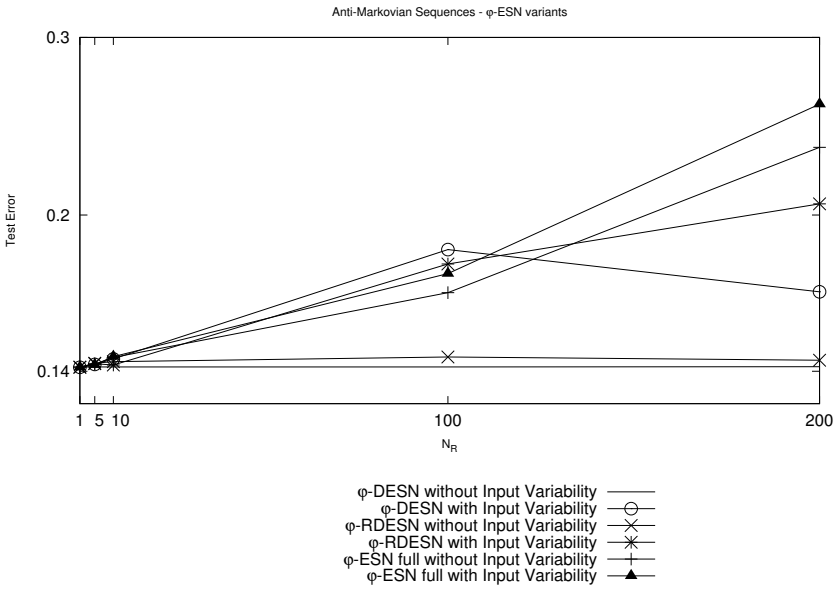


**Figure 17:** Averaged squared test errors for $\varphi$-ESN variants on the anti-Markovian sequences task. Errors are reported for increasing recurrent reservoir dimension, a constant dimension $N_\varphi = 500$ of the static part of the reservoir, and for a constant value of the contraction coefficient $\sigma = 0.5$.

current neurons achieved comparable results with ESNs *full* with 100 recurrent neurons, while $\varphi$-ESNs and $\varphi$-RDESNs with 100 recurrent neurons were better than ESNs *full* with 500 recurrent neurons (see Table 6). The best result on this task was obtained by $\varphi$-RDESN with input variability and $N_R = 200$, which outperformed the best standard ESNs (i.e. ESN *full* with input variability) and $N_R = 500$ of almost two orders of magnitude. This remarks the fact that a sufficient diversification of state dynamics can be produced even by reservoirs with a simple organization and a limited number of recurrent units. For the Mackey-Glass task, a number of 100 recurrent units turned out to be sufficient for producing this diversification. The extra reservoir dimensions represent a facility for the readout, but are not strictly necessary for the enrichment of the state dynamics. Analogous considerations can be done for the NARMA system task, for which we found that $\varphi$-ESNs and $\varphi$-RDESNs with input variability beat the best ESNs *full* for a number of recurrent neurons of $N_R = 100$. For larger recurrent reservoirs the ESN *full* model achieved a performance which is only slightly better than the best one obtained with $\varphi$-ESNs (Table 7). For the Markovian sequences task the effect of the augmented reservoir dimensionality is much less evident (see Figure 20). In general, we found no any relevant difference in the performance of standard ESNs and $\varphi$-ESN variants. It is apparent that $\varphi$-DESN and $\varphi$-RDESN with input variability produced nearly the same results as the best DESN model with input variability, while $\varphi$-ESN (both fully and sparsely connected) lowered the performance of two orders of magnitude. However, we can note that these results are again mainly due to the peculiarity of the task, which is of a pronounced Markovian nature and, what is arguably more important, of linear dynamics. Therefore it seems quite likely that a non-

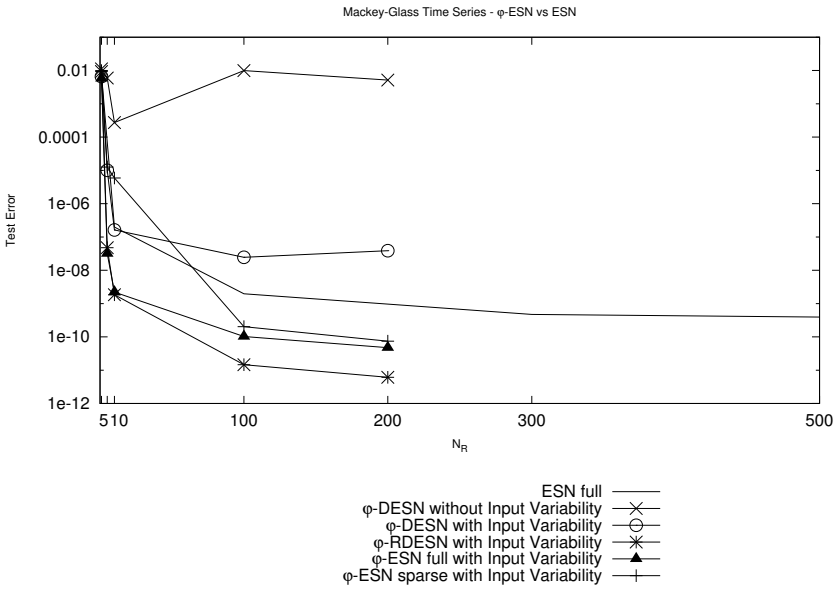


**Figure 18:** Averaged squared test errors for $\varphi$-ESN variants and the best ESN architecture for the Mackey-Glass task. Errors are reported for increasing recurrent reservoir dimension, a constant dimension $N_\varphi = 500$ of the static part of the reservoir, and for a constant value of the contraction coefficient $\sigma = 0.9$.

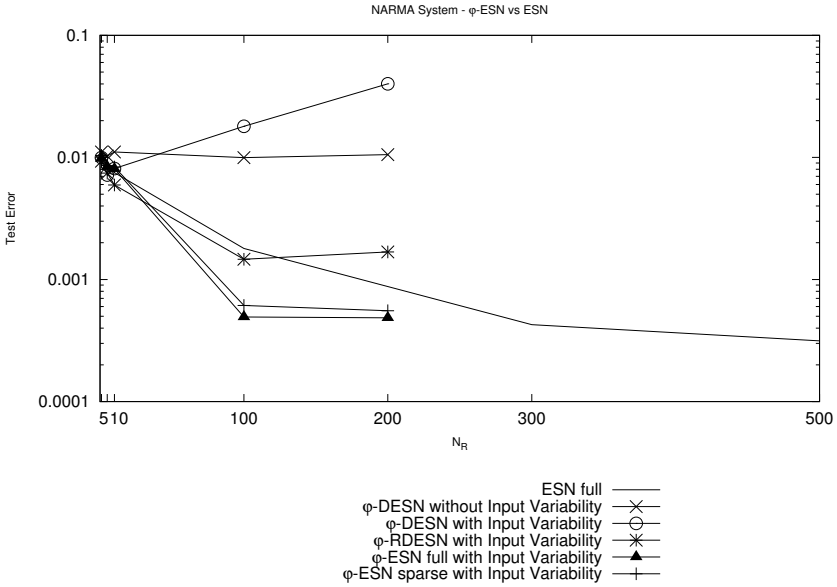


**Figure 19:** Averaged squared test errors for $\varphi$-ESN variants and the best ESN architecture for the NARMA system task. Errors are reported for increasing recurrent reservoir dimension, a constant dimension $N_\varphi = 500$ of the static part of the reservoir, and for a constant value of the contraction coefficient $\sigma = 0.9$.

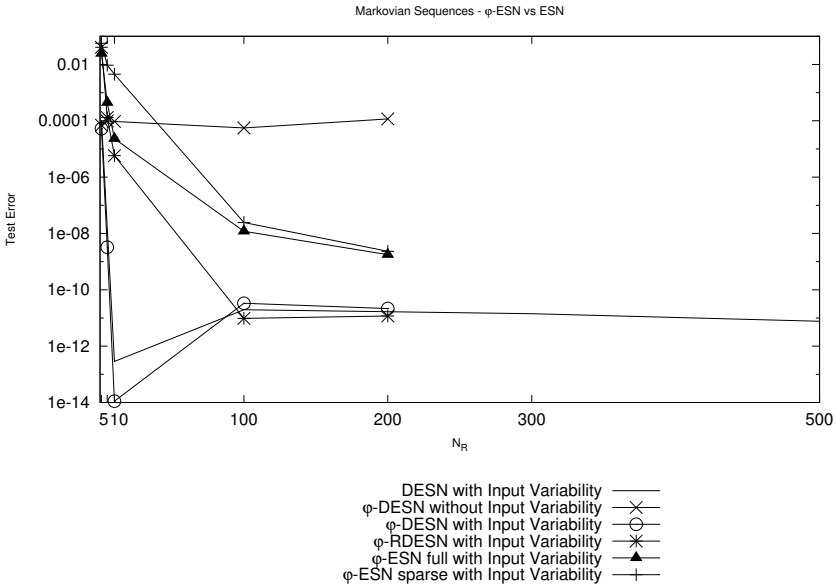


**Figure 20:** Averaged squared test errors for $\varphi$-ESN variants and the best ESN architecture for the Markovian sequences task. Errors are reported for increasing recurrent reservoir dimension, a constant dimension $N_\varphi = 500$ of the static part of the reservoir, and for a constant value of the contraction coefficient $\sigma = 0.9$.

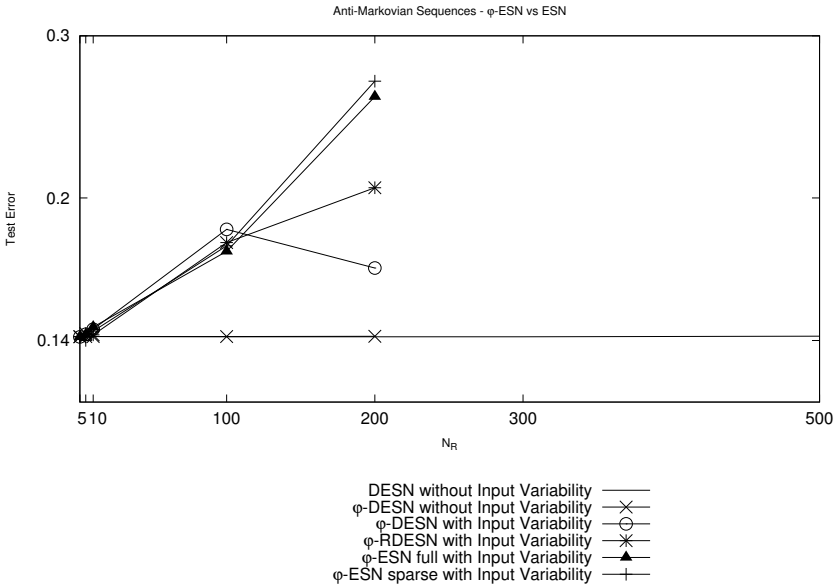


**Figure 21:** Averaged squared test errors for $\varphi$-ESN variants and the best ESN architecture for the anti-Markovian sequences task. Errors are reported for increasing recurrent reservoir dimension, a constant dimension $N_\varphi = 500$ of the static part of the reservoir, and for a constant value of the contraction coefficient $\sigma = 0.9$.

| | Test Error |
|---|---|
| **ESN** ***full*** ($N_R = 100$) | $1.9690 \times 10^{-9}$ |
| **ESN** ***full*** ($N_R = 500$) | $3.9408 \times 10^{-10}$ |
| $\varphi$**-ESN** ***full*** ($N_R = 10$) | $2.2203e \times 10^{-9}$ |
| $\varphi$**-RDESN** ($N_R = 10$) | $1.8430 \times 10^{-9}$ |
| $\varphi$**-ESN** ***full*** ($N_R = 100$) | $1.0218 \times 10^{-10}$ |
| $\varphi$**-RDESN** ($N_R = 100$) | $1.4548 \times 10^{-11}$ |
| $\varphi$**-ESN** ***full*** ($N_R = 200$) | $4.7595 \times 10^{-11}$ |
| $\varphi$**-RDESN** ($N_R = 200$) | $6.1120 \times 10^{-12}$ |

**Table 6:** Mean squared test errors on the Mackey-Glass task for ESNs and $\varphi$-ESN variants. $N_R = 100, 500$ for ESN with full connectivity. $N_R = 10, 100, 200$ and $N_\varphi = 500$ for $\varphi$-ESN variants. Contraction coefficient $\sigma = 0.9$ for every model.

| | Test Error |
|---|---|
| **ESN** ***full*** ($N_R = 100$) | $1.7967 \times 10^{-3}$ |
| **ESN** ***full*** ($N_R = 500$) | $3.1413 \times 10^{-4}$ |
| $\varphi$**-ESN** ***full*** ($N_R = 100$) | $4.9291 \times 10^{-4}$ |
| $\varphi$**-RDESN** ($N_R = 100$) | $1.4637 \times 10^{-3}$ |
| $\varphi$**-ESN** ***full*** ($N_R = 200$) | $4.8569 \times 10^{-4}$ |
| $\varphi$**-RDESN** ($N_R = 200$) | $1.6815 \times 10^{-3}$ |

**Table 7:** Mean squared test errors on the NARMA system task for ESNs and $\varphi$-ESN variants. $N_R = 100, 500$ for ESN with full connectivity. $N_R = 100, 200$ and $N_\varphi = 500$ for $\varphi$-ESN variants. Contraction coefficient $\sigma = 0.9$ for every model.

linear augment of the reservoir state space is not the architectural factor which can determine a critical improvement in the predictive performance.
If we turn now to the anti-Markovian task, we can note from Figure 21 that the overfitting problem is contained by augmenting the reservoir dimensionality. However, for recurrent reservoirs organized in a more complex way than simple DESN with no input variability (which is functionally equivalent to one single unit) we obtained worse results. $\varphi$-ESN variants were not able to solve this anti-Markovian task just as the ESN counterparts were.

### A Note on the Predictive Performance of $\varphi$-ESNs

Performances of $\varphi$-ESN variants are also consistent with the best results reported in literature. For instance, we tested the $\varphi$-ESN model on a variant of the Mackey-Glass task considered in [21]. For each repetition of the experiment, $N_{train} = 3000$ time steps of the series were generated for training, $N_{transient} = 1000$ of which were used as initial transient. After the training process, the network was driven by a number of $N_{teacher} = 3000$ time steps of the correct continuation of the training series. The network was then left run freely, driven by its own output for a number of $N_{freerun} = 84$ time steps, after which the discrepancy between the network output and the correct continuation of the time series was evaluated. A number of $N_{trials} = 100$ independent repetitions of the experiment has been carried out, in order to compute the normalized root-mean squared error after 84 passes ($NRMSE_{84}$), as in [21]:

$$NRMSE_{84} = \sqrt{\frac{\sum_{n=1}^{N_{trials}} (y(84) - \hat{y}(84))^2}{N_{trials}\, Var_{signal}}} \tag{27}$$

where $y(84)$ and $\hat{y}(84)$ are respectively the network output and the correct value of the time series after the free-run period, while $Var_{signal} \approx 0.046$ is the variance of the Mackey-Glass time series signal.
ESNs achieved the best result known in literature on this task, with $NRMSE_{84} \approx 10^{-4.2} (= 6.3096 \times 10^{-5})$ for a reservoir dimensionality equal to $N_R = 1000$, a sparse connectivity of 1% and a fixed value of the spectral radius $\rho = 0.8$, as reported in [21]. In our experiments, network settings similar to those specified in [21] were adopted. Input-to-output connections were added to the basic model of equation (9), a constant input bias for reservoir units equal to 0.2 was used, and normal noise of size $10^{-10}$ was added to the input for the readout before training. Reservoir weight matrices were scaled to have a fixed spectral radius of value $\rho = 0.9$ and input-to-reservoir weight values were randomly selected from a uniform distribution over $[-1, 1]$. For ESNs *sparse* with $N_R = 1000$ reservoir units and 1% of connectivity, we obtained an error value[6] of $NRMSE_{84} = 3.9034 \times 10^{-5}$. For $\varphi$-ESNs we considered reservoirs with a number of $N_R = 200$ units in the recurrent part, with a projection into a feature space of size $N_\varphi = 5000$. Weight values in matrix $\mathbf{W}_\varphi$ were randomly chosen according to a uniform distribution over $[-0.1, 0.1]$. For a sparse connectivity of 10% we obtained an error value of $NRMSE_{84} = 3.73066 \times 10^{-5}$, while for a connectivity equal to 25% the error was $NRMSE_{84} = 2.7589 \times 10^{-5}$.

Note that both these results outperformed the performance of the standard ESN model by using only one fifth of the recurrent reservoir units. Moreover, compared to the standard ESN tested in this experiment, the $\varphi$-ESN model with 10% of sparse connectivity had on average a smaller number of recurrent reservoir connections, while for the $\varphi$-ESN model with 25% of connectivity the averaged number of recurrent connections was the same [7].

# 7 Conclusions

Markovianity and high dimensionality of the reservoir state space representation have revealed a relevant influence on the behavior and performance of the ESN model.

First, we have observed that the contractivity of the state transition function leads ESN states to suffix-based Markovian representation of input sequences. This intrinsic Markovian organization of the reservoir state space of ESNs has been called the Markovian factor in this paper. The role of the Markovian factor (and of the "fading memory" property of ESNs [16]) is not restricted to delineate the basic ESN property, allowing the state to be independent of the initial conditions, but it has much deeper implications. In particular, we have shown that it is possible to define very easy or very hard tasks, by respecting or contrasting the Markovian assumption, respectively. The Markovian factor defines a major inherent limitation of the ESN approach whenever an anti-Markovian condition characterizes the task at hand. On the other hand, we have shown that on tasks with a distinct Markovian nature, the desired target behavior is synthesized by

[6] The fact that the $NRMSE_{84}$ we found is smaller than what reported in [21] might depend on the different value of the spectral radius adopted and on the different method used for the Mackey-Glass time series discretization (see Appendix A).

[7] The averaged number of recurrent connections for the standard ESN model and the $\varphi$-ESN model with 25% of connectivity was 10000, while for the $\varphi$-ESN model with 10% of connectivity it was 4000.

the value of the contraction coefficient of the network. Thus one single reservoir unit may result in the best performance, if its parameters are properly set. Our subsequent investigations have risen from this base case, showing how different architectural factors of ESN design can be used to introduce a diversification in the model dynamics that reveal to be useful for common tasks that do not belong to these two extreme classes.

The second general result of our investigation concerns the effect of high dimensionality of the reservoir state space in the ESN design. We have identified and evaluated architectural factors for which a high number of reservoir units is effective. In fact, there are tasks for which the influence of the Markovian factor can be characterized as in the middle between the outlined classes of easy and hard tasks. For such tasks the architectural design has revealed a major role. In particular, Markovianity is not sufficient to completely characterize the behavior of the ESN model for tasks for which the best performance has been obtained for high dimensional reservoirs. This situation is typical in tasks presented in the ESN literature, such as the Mackey-Glass chaotic time series and the 10-th order NARMA system, which have been used in this paper as test bed.

The effect of dimensionality has been experimentally analyzed firstly by considering the richness of dynamics introduced by differentiating the reservoir units activations through few basic architectural factors of ESN design, namely variability on the input, variability on the contractivity of units (multiple time-scales dynamics) and variability on the interaction among units, on reservoirs with the same degree of contractivity. Accordingly, we have presented two variants on the basic ESN model: the DESN model, which only supports input variability, and the RDESN model, in which multiple time-scales may be supported as well. The identified architectural factors have individually shown an influence in progressively improving ESN performance (with increasing reservoir dimension). Moreover, their combination can sum the individual effects generally resulting in a further improvement of the performance. For what concerns the assessment of the relative importance of those three factors, we found that the different dynamics ruled by the multiple time-scales dynamics and non-linear interactions factors have shown the major empirical effects. Nevertheless, input variability among reservoir units can show by itself a significant, thought inferior, impact on performance, as for the Mackey-Glass task. Interactions among reservoir units has been observed to be more influent in the case of the NARMA system task. Moreover, this last factor has shown to express a clear influence on model performance even in presence of a small number of units interactions, corresponding to a sparse reservoir connectivity.

As a global result, the general dependence of the ESN performance on the reservoir dimensionality has thus been decomposed and traced back to the dependence on the architectural factors that differentiate the reservoir dynamics. Increasing the reservoir dimensionality is then a way to allow a better expression of the units diversification due to the presence of such factors.

The other aspect is related to the effect of high reservoir dimensionality in making the readout regression easier. High dimensional representations of the input sequences have been constructed by a non-linear random projection of the reservoir activation into a higher dimensional feature space. This corresponds to the introduced $\varphi$-ESN model. The effects of this state dimensionality amplification have been compared with standard ESN models in terms of predictive performance, showing that actually a modest number of recurrent units

is sufficient to produce the necessary diversification in the reservoir state. For instance, we have shown that for the Mackey-Glass time series prediction, the set of diverse state dynamics obtained with 100 units reservoirs may result in a higher probability of discriminating among different input patterns when the state is randomly and non-linearly mapped into a 500 units space. Extra reservoir dimensions have thus turned out to be only partially responsible for the enrichment of ESN dynamics. The other part of the effect of the increased reservoir dimensionality has found to be just a facility for the linear readout tool. The possibility of regressing a high dimensional reservoir state space has revealed a relevant role in determining the performance of ESNs. For this reason it has been identified as a distinct architectural factor.

Although the aim of this paper has been focused on analyzing and assessing the properties of Markovianity and of other relevant reservoir architectural factors to give insights for future research, a number of simple and useful outcomes can be derived for practical use of ESNs.

First, Markovianity can greatly help in characterizing a successful or unsuccessful use of ESNs. A simple architectural design is convenient when the target task has a strongly Markovian nature, while poor results are to be expected (independently of the architectural design and of the number of reservoir units) for tasks with anti-Markovian properties. ESNs are not suitable for anti-Markovian processing.

Second, despite ESN literature claims from the very beginning [16, 21], sparse connectivity did not show to have a major role in defining the richness of reservoir dynamics, as a sparse configuration of the reservoir weight matrix did not affect the behavior of the different architectures. However, provided that the identified architectural factors are included in the ESN design, a sparse connectivity among reservoir units provides an efficient solution.

Third, both variability and dimensionality of the reservoir state dynamics are essential features in searching the best architectural configuration. The provided architectural factors define some possible architectural variants with different values. According to the suitable trade-off between efficiency and performance for the task at hand, different combinations can be exploited.
For ESNs with a fixed global contractive parameter, a better performance can be achieved whenever reservoir units can activate multiple time-scales dynamics by differentiating their self recurrent weights. If efficiency is important, then the simple diagonal reservoir architecture (i.e. RDESN), with number of recurrent weights that is linear (instead of quadratic) in the number of units, should be tried. Moreover, this factor, together with a $\varphi$-ESN architecture, may lead to a high performance model.

Finally, although further studies are needed, the authors hope that introduction of simple key factors analysis can contribute to the critical positioning of the ESN model in the area of the machine learning for sequence processing.

# A Appendix: Implementation Details

Our code was written in C++. For mathematical classes and functions we used the IT++ 4.0.6 library[8].
To train the linear readout in an off-line fashion, as explained in Section 3.3, we

[8]available at http://itpp.sourceforge.net/current/index.html.

used pseudo-inversion. This was implemented through singular value decomposition, by using the IT++ function `svd`.

In the case of sparse reservoirs, the recurrent weight matrix $\hat{\mathbf{W}}$ could result in a null matrix with zero norm for small reservoir dimensions. In such cases the null matrices were discarded until a valid one (with the correct desired norm) was generated.

To obtain the discrete version of the Mackey-Glass time series (see Section 6.1.2) we used a 4-th order Runge-Kutta method, with a sampling rate of 1.0 and integration step of 0.01. The implemented functions were adapted to C++ from functions available at http://www.bme.ogi.edu/∼ericwan/data.html. The initial conditions for the series were randomly chosen according to a uniform distribution over $[0, 1]$. Parameter $\alpha$ of equation (25) was set to 17.0 for every experiment on this benchmark. The first 1000 elements of each generated series were discarded as in [21]. Elements of each discrete sequence obtained were shifted by $-1$ and passed through the hyperbolic tangent function, as in [21].

For the NARMA system task we implemented input-target sequences according to what specified in Section 6.1.3. Instances which resulted in a divergent target sequence were rejected.

Markovian and anti-Markovian input sequences and target values were obtained using equations (23) and (24). Each target value was then scaled into the range $[-1, 1]$. This was done by firstly computing the minimum ($minvalue$) and maximum ($maxvalue$) possible target values for a given range of possible input sequence lengths. Then values in $[minvalue, maxvalue]$ were rescaled into $[-1, 1]$.

The header file used to generate all benchmark data is available online at *http://www.di.unipi.it/∼gallicch/sources/*.

# References


[1] A. F. Atiya and A. G. Parlos. New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE Transactions on Neural Networks*, 11(3):697–709, May 2000.

[2] M. Buehner and P. Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820–824, May 2006.

[3] M. Cernanský and M. Makula. Feed-forward echo state networks. In *IJCNN 2005. Proceedings of the IEEE International Joint Conference on Neural Networks, 2005*, volume 3, pages 1479–1482, 2005.

[4] M. Cernanský and P. Tiňo. Predictive modeling with echo state networks. In *Artificial Neural Networks - ICANN 2008*, volume 5163/2008, pages 778–787. Springer Berlin / Heidelberg, 2008.

[5] P. F. Dominey, M. Hoen, J.M. Blanc, and T. Lelekov-Boissard. Neurological basis of language and sequential cognition: Evidence from simulation, aphasia, and erp studies. *Brain and Language*, 86(2):207 – 225, 2003.

[6] P.F. Dominey, M. Hoen, and T. Inui. A neurolinguistic model of grammatical construction processing. *Journal of Cognitive Neuroscience*, 18(12):2088–2107, 2006.

[7] G. Fette and J. Eggert. Short term memory and pattern matching with simple echo state networks. In *ICANN (1)*, volume 3696 of *Lecture Notes in Computer Science*, pages 13–18. Springer, 2005.

[8] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9:768–786, 1998.

[9] S. Haeusler and W. Maass. A Statistical Analysis of Information-Processing Properties of Lamina-Specific Cortical Microcircuit Models. *Cerebral Cortex*, 17(1):149–162, 2007.

[10] M.A. Hajnal and A. Lorincz. Critical echo state networks. In *Artificial Neural Networks ICANN 2006*, pages 658–667, 2006.

[11] B. Hammer, A. Micheli, and A. Sperduti. Adaptive contextual processing of structured data by recursive neural networks: A survey of computational properties. In *Perspectives of Neural-Symbolic Integration*, volume 77/2007 of *Studies in Computational Intelligence*, pages 67–94. Springer Berlin / Heidelberg, 2007.

[12] B. Hammer and P. Tiňo. Recurrent neural networks with small weights implement definite memory machines. *Neural Computation*, 15(8):1897–1929, 2003.

[13] S. Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 1999.

[14] J. Hertzberg, H. Jaeger, and F. Schönherr. Learning to ground fact symbols in behavior-based robots. In *ECAI. Proceedings of the 15th Eureopean Conference on Artificial Intelligence*, pages 708–712. IOS Press, 2002.

[15] K. Ishu, T. van der Zant, V. Becanovic, and P. Ploger. Identification of motion with echo state network. In *OCEANS 2004. MTTS/IEEE TECHNO-OCEAN 2004*, volume 3, pages 1205–1210 Vol.3, November 2004.

[16] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical report, GMD - German National Research Institute for Computer Science, 2001.

[17] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In *NIPS. Advances in Neural Information Processing Systems 15*, pages 593–600. MIT Press, 2002.

[18] H. Jaeger. Short term memory in echo state networks. GMD-Report 152, GMD - German National Research Institute for Computer Science, 2002.

[19] H. Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach. Technical report, GMD - German National Research Institute for Computer Science, 2002.

[20] H. Jaeger. Reservoir riddles: suggestions for echo state network research. *IJCNN 2005. Proceedings of the IEEE International Joint Conference on Neural Networks, 2005*, 3:1460–1462 vol. 3, July-4 Aug. 2005.

[21] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, April 2004.

[22] J.F. Kolen and S.C. Kremer, editors. *A Field Guide to Dynamical Recurrent Networks*. IEEE Press, Inc., New York, 2001.

[23] M. Lukosevicius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127 – 149, 2009.

[24] W. Maass, T. Natschlager, and H. Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–60, 2002.

[25] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.

[26] M. Makula, M. Cernanský, and L. Benusková. Approaches based on markovian architectural bias in recurrent neural networks. In *SOFSEM 2004: Theory and Practice of Computer Science*, volume 2932 of *Lecture Notes in Computer Science*, pages 257–264. Springer, 2004.

[27] D. Nikolic, S. Häusler, W. Singer, and W. Maass. Temporal dynamics of information content carried by neurons in the primary visual cortex. In *NIPS. Advances in Neural Information Processing Systems 19*, pages 1041–1048. MIT Press, 2006.

[28] M.C. Ozturk, D. Xu, and J.C. Principe. Analysis and design of echo state networks. *Neural Computation*, 19(1):111–138, 2007.

[29] P. Plöger, A. Arghir, T. Günther, and R. Hosseiny. Echo state networks for mobile robot modeling and control. In *RoboCup*, pages 157–168, 2003.

[30] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez. Training recurrent networks by evolino. *Neural Computation*, 19(3):757–779, 2007.

[31] B. Schrauwen, M. Wardermann, D. Verstraeten, J.J. Steil, and D. Stroobandt. Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71(7-9):1159 – 1171, 2008.

[32] H. T. Siegelmann and E. D. Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4:77–80, 1991.

[33] M.D. Skowronski and J.G. Harris. Minimum mean squared error time series classification using an echo state network prediction model. In *ISCAS 2006. Proceedings of the IEEE International Symposium on Circuits and Systems, 2006*, pages 4 pp.–3156, 0-0 2006.

[34] J. J. Steil. Backpropagation-decorrelation: online recurrent learning with o(n) complexity. In *IJCNN 2004. Proceedings of the IEEE International Joint Conference on Neural Networks, 2004*, volume 2, pages 843–848, July 2004.

[35] J. J. Steil. Online stability of backpropagation-decorrelation recurrent learning. *Neurocomputing*, 69(7-9):642–650, 2006.

[36] P. Tiňo, M. Cernanský, and L. Benusková. Markovian architectural bias of recurrent neural networks. *IEEE Transactions on Neural Networks*, 15(1):6–15, Jan. 2004.

[37] P. Tiňo, B. Hammer, and M. Bodén. Markovian bias of neural-based architectures with feedback connections. In *Perspectives of Neural-Symbolic Integration*, pages 95–133. Springer-Verlag, 2007.

[38] G. K. Venayagamoorthy and B. Shishir. Effects of spectral radius and settling time in the performance of echo state networks. *Neural Networks*, 22(7):861 – 863, 2009.

[39] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391 – 403, 2007.

[40] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[41] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[42] T. Yamazaki and S. Tanaka. The cerebellum as a liquid state machine. *Neural Networks*, 20(3):290 – 297, 2007.

[43] X. Yanbo, Y. Le, and S. Haykin. Decoupled echo state networks with lateral inhibition. *Neural Networks*, 20(3):365–376, 2007.

[44] Y.Bengio, P. Frasconi, and P. Simard. The problem of learning long-term dependencies in recurrent networks. *IEEE International Conference on Neural Networks, 1993*, pages 1183–1188 vol.3, 1993.