Neurocomputing 268 (2017) 87–99

Contents lists available at ScienceDirect

Neurocomputing

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

Deep reservoir computing: A critical experimental analysis

Claudio Gallicchio*, Alessio Micheli, Luca Pedrelli

Department of Computer Science, University of Pisa, Largo Bruno Pontecorvo 3, Pisa 56127, Italy

ARTICLE INFO

Article history:
Received 8 July 2016
Revised 5 December 2016
Accepted 21 December 2016
Available online 29 April 2017

Keywords:
Reservoir computing
Echo State Networks
Deep Learning
Deep neural networks
Recurrent neural networks
Multiple time-scale dynamics

ABSTRACT

In this paper, we propose an empirical analysis of deep recurrent neural network (RNN) architectures with stacked layers. The main aim is to address some fundamental open research issues on the significance of creating deep layered architectures in RNN and to characterize the inherent hierarchical representation of time in such models, especially for efficient implementations. In particular, the analysis aims at the study and proposal of approaches to develop and enhance hierarchical dynamics in deep architectures within the efficient Reservoir Computing (RC) framework for RNN modeling. The effect of a deep layered organization of RC models is investigated in terms of both occurrence of multiple time-scale and increasing of richness of the dynamics. It turns out that a deep layering of recurrent models allows an effective diversification of temporal representations in the layers of the hierarchy, by amplifying the effects of the factors influencing the time-scales and the richness of the dynamics, measured as the entropy of recurrent units activations. The advantages of the proposed approach are also highlighted by measuring the increment of the short-term memory capacity of the RC models.

1. Introduction

The potential ability of Deep Learning models to learn data representations at different levels of abstraction [1–4] has progressively attracted the interest of the machine learning community, in particular for the possibility of disentangling the difficulties in modeling complex tasks by representing them in terms of simpler ones in a hierarchical fashion. In the neuro-computing area, deep neural networks are often composed by a feed-forward hierarchy of multiple hidden layers of non-linear units. Training of such architectures is particularly time-demanding, so that a fundamental aspect of the successful spread of deep neural networks is related to recent hardware advancements in the area of high performance computing, especially graphics processing units (GPUs), aiming at training deep models in affordable times.

The recent introduction of deep architectures in the class of recurrent neural networks (RNNs) is arousing a growing interest under both theoretical and applicative points of view [5–8], especially in regard to the possibility of developing a hierarchical processing of temporal data. Indeed, the ability to represent dynamical features at multiple levels of abstraction allows to capture more naturally the temporal structure of the data whenever it is intrinsically characterized by a multiple time-scales organization. Among the others, language [6], speech [9] and text processing [8] represent notable examples of application areas involving time-series data with this type of characterization. Besides, the capability of modeling multiple time-scales in recurrent networks dynamics has proved effective also as a mean to deal with long-term dependencies, as evidenced e.g. in [7,10] and, more recently, in [4,11].

Some works in literature aimed at obtaining multiple time-scales dynamics in a layered RNN architecture. One approach consists in progressively sub-sampling the input to the higher layers [7], forcing the different layers to operate at different frequencies [12]. Another approach consists in learning all the weights in the stack of recurrent layers, which is a difficult and extremely time consuming process even using GPUs and can require ad-hoc incremental training strategies in order to be effective [8].

The state-of-the-art in this respect is still in its infancy with many open challenges [4,13], and some intuitions present in literature deserve further research and critical assessments. In particular, the observation that stacking layers of recurrent units inherently creates different time-scales dynamics at different layers [6,8], and therefore a hierarchical representation of the temporal information *per se*, deserves to be investigated and analyzed.

A starting point for our analysis in this regard is represented by the observation that stacking recurrent layers can be actually interpreted as the application of a set of constraints to the architecture of a fully-connected RNN (with the same number of units). Such constraints involve the pattern of connectivity among the recurrent units (i.e. avoiding connections from higher layers to lower layers), which affects the flow of information and the dynamics of

* Corresponding author.
E-mail addresses: gallicch@di.unipi.it (C. Gallicchio), micheli@di.unipi.it (A. Micheli), luca.pedrelli@di.unipi.it (L. Pedrelli).

http://dx.doi.org/10.1016/j.neucom.2016.12.089
0925-2312/

sub-parts of the network state. Moreover, the architectural restrictions also concern the connectivity with the input layer (i.e. allowing only to the units in the first layer to be fed directly by the input), influencing the way in which the external input information is seen by recurrent units progressively more distant from the input layer (see an explanation related to the illustration of network architectures in Section 2.2).

This motivates us to a critical assessment of the possible and effective merits of a layered structure for recurrent architectures and to propose different approaches to achieve a hierarchy in temporal representation by efficient deep recurrent models.

To this aim, the modeling proposal of this paper is based on Reservoir Computing (RC) [14,15], which represents a state-of-the-art approach for extremely efficient RNN modeling. Moreover, and more importantly for the analysis purposes, the RC approach yields the possibility to investigate the architectural factors of deep models in a decoupled fashion with respect to the learning aspects of the dynamical part of the networks. This type of analysis on the one hand can provide insights on the true merits of learning of deep RNN dynamics, and on the other hand it allows to propose efficiently trained models for multiple time-scales processing of temporal data.

Previous works on hierarchical organizations of RC networks mainly focused on *ad-hoc* architectures of trained modules for temporal feature discovery [14], but still lack of a general view over the effective potentiality and emerging properties of deep architectures of layered reservoirs. In particular, since different parameters of RC models strongly affect their dynamical behavior and performance, their relationships with layering deserve a systematic investigation, still missing in literature. Such investigation allows us to study proposals on what aspects ruling the dynamics of reservoir models can amplify the potential benefits of a deep architecture (and vice versa), in particular for the timescale dynamics differentiation, as well as for the effect of known RC techniques, such as Intrinsic Plasticity (IP) [16–18], to enhance the richness of state dynamics (measured as the entropy of reservoir states).

Overall, the points at issue in this paper are how to obtain, enhance, and quantify the occurrence of different time-scales (or amplify the richness of the state dynamics) in deep recurrent architectures, thus contributing to address the following open issues: (i) Why stacking recurrent layers imposing constraints with respect to a full-connected RNN in favor of a layered deep organization? (ii) What is the inherent (independent from learning) architectural effect of layering on the hierarchical temporal dynamics developed by a deep RNN? (iii) Is it possible to keep the advantage of Deep Learning for RNN (e.g. in terms of multi time-scales representation of temporal data) by using an efficient approach such is RC? (iv) What is the role of the hyper-parameters that rule RC network dynamics within a layered organization of the reservoir?

This paper extends the preliminary investigation initiated in [19], by providing a deepened experimental analysis carried out through numerical simulations (referred to as experiments in the rest of this paper) both in terms of extension of model configurations and parameters, and of a new benchmark supporting the results. Furthermore, this paper introduces a novel quantitative analysis through measures of the synergistic effect of RC factors and layering for the ordering and separation of the developed time-scales, and for the richness of the state dynamics, reporting also the results of a complete experimental investigation on the short-term memory capacity (MC) task.

This paper is organized as follows. Section 2 provides an introduction to the basics of the models and architectures analyzed in the paper. Results of numerical simulations are presented and discussed in Section 3, by firstly taking into consideration the effect of RC hyper-parameters and IP on time-scales differentiation, then investigating the influence of layering in enhancing the impact of IP in terms of richness of reservoir dynamics, and then reporting the results achieved on the MC task under the different architectural conditions considered. Finally, Section 4 draws the conclusions.

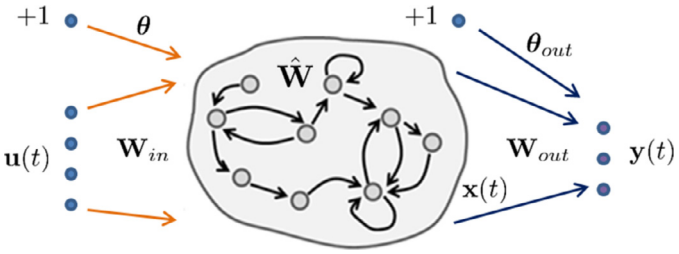


Fig. 1. The architecture of a shallow LI-ESN model.

2. Deep reservoir computing

In this section, we provide a description of the RC networks considered in this paper. First, in Section 2.1, we present a brief review of the standard shallow RC, then, in Section 2.2, we introduce and discuss the proposed deep RC architecture and baseline variants.

2.1. Shallow Echo State Networks

Within the RC framework [14,15], the Echo State Network (ESN) model [20] is a state-of-the-art approach for efficiently modeling RNNs. ESNs implement discrete-time dynamical systems by means of the computation carried out by an untrained recurrent reservoir layer, providing a suffix-based Markovian representation of the past input history [21,22], and by a trained linear readout. In this work we consider the Leaky Integrator ESN (LI-ESN) model [23], a variant of the basic ESN in which leaky integrator reservoir units are adopted. The basic LI-ESN architecture is graphically illustrated in Fig. 1.

In a LI-ESN with N_U input units and N_R reservoir units the state is updated according to the following state transition function:

$$\mathbf{x}(t) = (1-a)\mathbf{x}(t-1) + a\tanh(\mathbf{W}_{in}\mathbf{u}(t) + \boldsymbol{\theta} + \hat{\mathbf{W}}\mathbf{x}(t-1)), \quad (1)$$

where $\mathbf{u}(t) \in \mathbb{R}^{N_U}$ and $\mathbf{x}(t) \in \mathbb{R}^{N_R}$ denote respectively the input and the reservoir state at time t, $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix, $\boldsymbol{\theta} \in \mathbb{R}^{N_R}$ is the bias-to-reservoir weight vector (where we assume an input bias equal to 1 for the reservoir units), $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent reservoir weight matrix, tanh is the element-wise applied hyperbolic tangent activation function and $a \in [0, 1]$ is the leaky parameter. The reservoir parameters are initialized according to the constraints specified by the Echo State Property (ESP) [20,21] and then are left untrained. Accordingly, the weight values in $\mathbf{W}_{in}$ and in $\boldsymbol{\theta}$ are chosen from a uniform distribution over $[-scale_{in}, scale_{in}]$, where $scale_{in}$ represents an input-scaling parameter. The values in matrix $\hat{\mathbf{W}}$ are randomly selected from a uniform distribution and then re-scaled such that the spectral radius (i.e. the largest eigenvalue in absolute value) of matrix $\tilde{\mathbf{W}} = (1-a)\,\mathbf{I} + a\,\hat{\mathbf{W}}$, denoted by ρ, is smaller than 1, i.e. $\rho < 1$.

Note that the leaky parameter a and the spectral radius of the recurrent weight matrix ρ represent two relevant RC hyper-parameters. In particular, the value of a is related to the speed of reservoir dynamics in response to the input, with larger values of a resulting in reservoirs that react faster to the input [14,23]. The value of ρ, besides of being linked to the ESP for valid ESN initialization, is related to the variable memory length and the degree of contractivity of reservoir dynamics [21], with larger values of $\rho <$ 1 resulting in longer memory length.

The output of the LI-ESN is computed by the readout as a linear combination of the reservoir state activation, according to:

$$\mathbf{y}(t) = \mathbf{W}_{out}\mathbf{x}(t) + \boldsymbol{\theta}_{out} \quad (2)$$

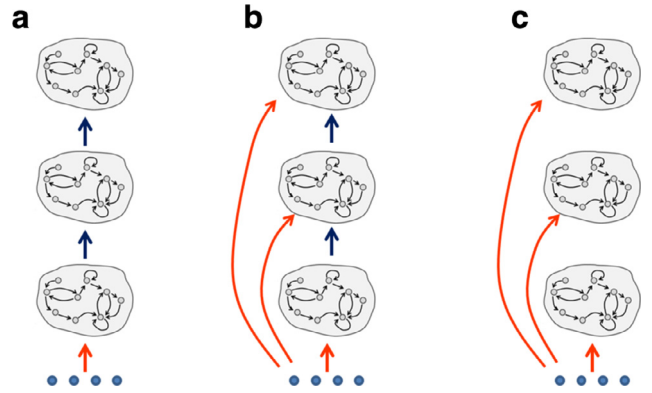


Fig. 2. Deep RC architectures: (a) deepESN, (b) deepESN-IA, (c) groupedESN. Aspects related to input bias and mathematical notation are not reported here for the ease of graphical representation (see text for details).

where $\mathbf{y}(t) \in \mathbb{R}^{N_Y}$ is the N_Y-dimensional output at time t, $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_R}$ is the reservoir-to-readout weight matrix and $\boldsymbol{\theta}_{out} \in \mathbb{R}^{N_Y}$ is the bias-to-readout weight vector (where we assume an input bias equal to 1 for the readout units). Training the readout involves solving a linear regression problem, typically approached by using direct methods such as Moore–Penrose pseudo-inversion. For further details on the properties of the RC approach the reader is referred to [14,20,21].

In the context of unsupervised reservoir adaptation techniques [14], a well known approach is represented by Intrinsic Plasticity (IP) [16–18]. With the goal of maximizing the entropy of the reservoir units output distribution, the IP rule implements a gradient descent algorithm that adapts the gain and bias parameters of the activation function locally to each reservoir unit. In particular, when using the hyperbolic tangent as activation function, as in our case, the IP rule aims at the minimization of the Kullback–Leibler divergence between the empirical output distribution and a Gaussian distribution [16].

In formulas, focusing the attention on a single reservoir unit, the *tanh* non-linearity can be expressed as $\tilde{x} = \tanh(g x_{net} + b)$, where x_{net} is the net input for the unit, $\tilde{x}$ is the output of the tanh non-linearity, whereas g and b respectively denote the gain and bias of the *tanh* function. Accordingly, the IP rule is described by the following equations [16]:

$$\begin{aligned} \Delta b &= -\eta(-(\mu/\sigma^2) + (\tilde{x}/\sigma^2)(2\sigma^2 + 1 - \tilde{x}^2 + \mu\tilde{x})), \\ \Delta g &= \eta/g + \Delta b\, x_{net}, \end{aligned} \tag{3}$$

where μ and σ denote the mean and standard deviation of the target Gaussian distribution, η is a learning rate, and the update Eq. (3) is applied individually to each unit in the reservoir. Further details on IP training can be found in literature works in [16–18].

2.2. Deep Echo State Networks

In this paper, we propose the study of deep RC architectures in which multiple reservoir layers are stacked one on top of each other. The main model that we take into consideration is a straight stack of reservoirs, called *deepESN* and shown in Fig. 2(a). In a deepESN, the first layer is fed by the external input and operates like the reservoir of a shallow ESN, whereas each successive layer is fed by the output of the previous one. The state transition function of deepESNs can be expressed as:

$$\begin{aligned} \mathbf{x}^{(l)}(t) = (1 - a^{(l)})\mathbf{x}^{(l)}(t-1) + a^{(l)} \tanh(\mathbf{W}_{in}^{(l)}\mathbf{i}^{(l)}(t) + \boldsymbol{\theta}^{(l)} \\ + \hat{\mathbf{W}}^{(l)}\mathbf{x}^{(l)}(t-1)), \end{aligned} \tag{4}$$

where the superscript (l) is used to refer the network's parameters and hyper-parameters at layer l. Assuming, for the sake of simplicity, that the same number of reservoir units N_R is present in each layer of the stack, $\mathbf{W}_{in}^{(l)}$ in Eq. (4) denotes the input weight matrix for layer l, $\boldsymbol{\theta}^{(l)} \in \mathbb{R}^{N_R}$ is the bias-to-reservoir weight vector for layer l and $\hat{\mathbf{W}}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ represents the recurrent weight matrix of layer l. In particular, for a deepESN we have $\mathbf{W}_{in}^{(1)} \in \mathbb{R}^{N_R \times N_U}$ for the first layer and $\mathbf{W}_{in}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ for successive layers, i.e. for $l > 1$. Moreover, note that $\mathbf{i}^{(l)}(t)$ in Eq. (4) is used to denote the input for lth layer of the deepESN architecture at time step t, i.e.:

$$\mathbf{i}^{(l)}(t) = \begin{cases} \mathbf{u}(t) & if \;\; l = 1 \\ \mathbf{x}^{(l-1)}(t) & if \;\; l > 1. \end{cases} \tag{5}$$

For what concerns the output computation in a deepESN, a readout component is used in order to linearly combine the outputs of all the reservoir units as in a standard ESN. Rewriting Eq. (2) by taking into account the hierarchical organization of the reservoir, and denoting the number of reservoir layers by N_L, the output of a deepESN at each time step t can be computed as:

$$\mathbf{y}(t) = \mathbf{W}_{out}\left[\mathbf{x}^{(1)}(t)\; \mathbf{x}^{(2)}(t)\; \ldots\; \mathbf{x}^{(N_L)}(t)\right]^T + \boldsymbol{\theta}_{out}, \tag{6}$$

where in this case $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_L N_R}$ represents the reservoir-to-readout weight matrix of the deepESN, connecting the reservoir units in all the layers to the units in the readout. Training a deepESN can therefore be accomplished by means of direct methods similarly to the case of a standard ESN. Although the main focus of our experimental analysis is on reservoir dynamics, readout training is also considered in this paper for the purposes of the MC task.

The importance of layering with respect to the construction of a progressively more abstract encoding of the input history, and the relevance of layering per se, are studied in the following by introducing some specific architectural cases as a baseline.

In particular, we consider a deepESN architectural variant in which the external input is provided to every layer, resulting in a model called deepESN Input to All (*deepESN-IA*), shown in Fig. 2(b). Similarly to the case of deepESN, the state transition function of deepESN-IA can be expressed by Eq. (4) in which the input for each layer $l > 1$ at step t is the concatenation of the external input and the state of the previous layer in the stack:

$$\mathbf{i}^{(l)}(t) = \begin{cases} \mathbf{u}(t) & if \;\; l = 1 \\ \left[\mathbf{u}(t)\; \mathbf{x}^{(l-1)}(t)\right]^T & if \;\; l > 1 \end{cases} \tag{7}$$

Accordingly, in a deepESN-IA for $l > 1$ we have that $\mathbf{W}_{in}^{(l)} \in \mathbb{R}^{N_R \times (N_R + N_U)}$. Note that while higher layers in a deepESN are at increasing distances from the (external) input, in a deepESN-IA the distance from the input is the same for every layer.

The relevance of layering in deepESN, with respect to the interplay among the reservoir dynamics at the different levels in the hierarchy, is investigated by considering an RC network containing sub-reservoirs that are all fed only by the input and are not organized in a stack. The resulting (shallow) architecture is called *groupedESN* and it is shown in Fig. 2(c). In this case, denoting by $\mathbf{x}^{(l)}$ the state of the lth sub-reservoir, the state transition function of a groupedESN can be formulated as follows:

$$\begin{aligned} \mathbf{x}^{(l)}(t) = (1 - a^{(l)})\mathbf{x}^{(l)}(t-1) + a^{(l)} \tanh(\mathbf{W}_{in}^{(l)}\mathbf{u}(t) + \boldsymbol{\theta}^{(l)} \\ + \hat{\mathbf{W}}^{(l)}\mathbf{x}^{(l)}(t-1)) \end{aligned} \tag{8}$$

where $\mathbf{W}_{in}^{(l)} \in \mathbb{R}^{N_R \times N_U}$ for every l, and it can be noticed that the dynamics of sub-reservoirs evolve independently of each other.

As a further architectural baseline, in the following we also take into consideration the case of a standard (shallow) fully connected ESN (whose dynamics are described by Eq. (1)), with the same number of reservoir units as in the whole architecture of a deep RC counterpart.

Note that, critically, a deep layered recurrent network adds architectural constraints to the recurrent connections of a fully connected RNN. A layered RNN can be re-interpreted as a fully connected RNN (with the same number of units) where some connections between groups of neurons are removed. In particular, a layered RNN architecture does not present connections from higher layers to lower layers and connections from the input layer to layers at a level greater than one, resulting in a constrained version of a fully connected RNN. Besides, we introduce (an implementation of) the staking by adding connections between groups (layers) of neurons without time-delays. Hence, states variables in Eq. (5) are all at the same time step t, thus imposing a serial order among the layers for computing the states at each time step. Of course, the same observations also apply when we take into consideration layered versus fully-connected ESN architectures. The study of the effects of the aforementioned constraints in the development of the hierarchical temporal dynamics is one of the main subject of this work. Moreover, again under a critical perspective, it is worth to note that an ESN with a shallow reservoir (i.e. without an explicit ordered layered structure) contains already by construction a rich pool of state dynamics (due to the random weight initialization). Hence, the same subject is studied in the following also with respect to the parameters that rule the ESN behavior, in order to investigate the possible enhancement due to a layered structure on their effect and on the state dynamics and temporal representation variety.

With this aim, in the following we investigate possible strategies aimed at driving the emergence of different time-scales dynamics through the different layers of a deep recurrent architecture.

Our first proposal consists in imposing by design a state dynamics differentiation among the layers, by setting different values of a and ρ at different layers (or sub-reservoirs in groupedESNs). Using different values of a implies a differentiation among the speed of state dynamics for the different layers of the deep architecture. Indeed, the use of leaky units results in the application of a running average on the state values [4], with the value of the leaky parameter at each layer determining the extent of the persistence of past information in the state dynamics at that layer (in our case, values of a closer to 1 imply that past information is more quickly discarded). Moreover, in this regard, it is worth to observe that the advantage of having RNN units with different leaky parameters in order to achieve multiple time-scales dynamics has been discussed in pioneering works already in the 1980s [24,25]. Varying the values of ρ implies a variation of contractivity [21,22] and memory length among the state dynamics of different layers.

Our second proposal consists in using an efficient unsupervised layer-wise adaptation of reservoir units by means of IP training. Specifically, we applied the IP rule as described by Eq. (3) indifferently for all the considered architectures.[1] In our experiments, for all the reservoir units we used the same values of the IP parameters μ, σ and η (see Eq. (3)), and we assessed the influence of layering also in terms of IP effect enhancement.

3. Experimental analysis

In this section, we present and discuss the results of our experimental analysis conducted by means of numerical simulations on deep RC networks, assessing the effectiveness of the methodologies introduced in Section 2.2. Specifically, in Section 3.1 we investigate the effect of architectural factors, RC hyper-parameters and IP learning on time-scales differentiation among the dynamics of different layers, in Section 3.2 we further inquire into the impact of IP on the richness of reservoir dynamics in deep RC architectures, and in Section 3.3 we evaluate the efficacy of the proposed approaches on the short-term memory capacity of the resulting models.

3.1. Time-scales differentiation

In order to assess the extent of the time-scales differentiation among the layers in the considered recurrent architectures, similarly to [8] we took into consideration an experimental setting comprising two input sequences, S_1 and S_2, both of length 5000 and identical to each other except for a typo (a perturbation) that is inserted in S_2 at time step 100. We ran the same RC network on both the unperturbed and the perturbed input sequences and collected the correspondingly obtained reservoir states, evaluating how long the effect of the input perturbation affects the dynamics of each layer by computing the distance between the states corresponding to S_1 and S_2 as a function of time. Specifically, a *qualitative* analysis is provided by plotting the Euclidean distance between the states of corresponding layers in the unperturbed and perturbed cases. In this concern, note that as S_1 and S_2 are identical until the typo at step $t = 100$, for all the layers the distances among the states are always zero *by construction* for all the time steps $t < 100$, and are therefore left out from the plots. Moreover, to complete the qualitative results provided by the plots described above, we also adopt *quantitative* measures of time-scales diversification. This is done by assessing the quality of the ordering among the time-scales in the different layers by resorting to known distances between rankings [26], i.e. Kendall's tau (KT) and Spearman's footrule (SF), and by introducing an index of time-scales separation (IS). Smaller values of KT and SF indicate a better ordering of time-scales across the layers, while higher values of IS denote a greater spacing among the duration of the perturbation effect across the layers. Details on these qualitative and quantitative means of investigation (including definitions of KT, SF and IS) are reported in Appendix A.

We instantiated this experimental approach by considering two datasets. The first dataset is an *Artificial* time-scales dataset, designed to avoid biases towards specific applications, in which each input element is drawn from a uniform distribution from an alphabet of 10 elements. The second dataset comes from an excerpt of the *Wikipedia* text corpus [27], used in [8] and adopted here to evaluate our results also for the case of a realistic task. Elements in the Wikipedia dataset represent characters, where in our setup we considered an alphabet comprising the 95 most common ones (the printable ASCII characters) and an unknown character (used to represent all the others), as in [8], for a total number of 96 characters. For both the datasets, we represented the input elements by using a one-hot encoding approach, thereby resulting in a one-of-10 encoding for the Artificial dataset and in a one-of-96 encoding for the Wikipedia dataset.

In our experiments, for the only scope of analysis and for the sake of its uniformity and simplicity, we considered deepESN (and deepESN-IA) stacked architectures with 10 layers of 10 fully connected units each with input scaling $scale_{in} = 1$. Analogously, for groupedESN, we used networks with 10 sub-reservoirs of 10 units. Moreover, for baseline comparison with the standard RC case, we also considered ESNs with 100 fully connected units, i.e. the total number of reservoir units used for the deep RC setup. We independently generated 10 guesses for each network hyper-parameterization, and averaged the results over such guesses.

For the sake of conciseness, we present the results of the qualitative analysis with the time-scales plots only for the Artificial dataset, while the quantitative results, i.e. values of KT, SF and IS, are reported also for the Wikipedia dataset. In the plots pre-

[1] The only practical aspect that changes in the different architectural cases is the way in which the net input is computed, i.e. the value of x_{net} in Eq. (3), which depends on the input received by each reservoir unit.

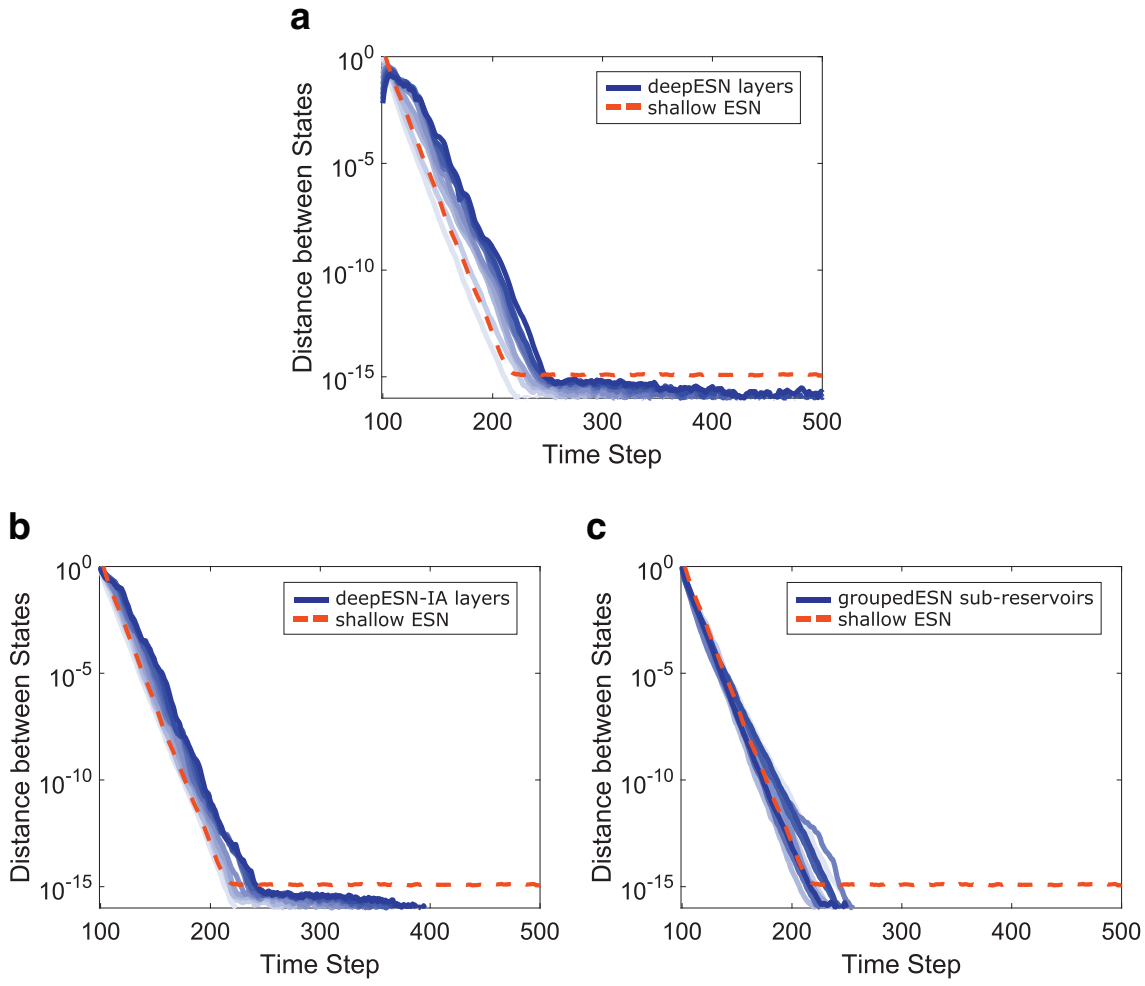


Fig. 3. Distance between perturbed and unperturbed states on the Artificial dataset for the considered RC architectures with $a = 0.55$ and $\rho = 0.9$ for every layer (or sub-reservoir). Continuous blue lines correspond to layers in deep RC networks (darker colors for higher layers) and to sub-reservoirs in groupedESN. Dashed red lines correspond to the shallow ESN with the same number of total reservoir units and hyper-parameterization. (a) deepESN, (b) deepESN-IA, (c) groupedESN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sented in the following, the curves for each layer (or sub-reservoir) are averaged over the 10 network guesses considered (in order to avoid influences of single instances on our analysis). Analogously, the values of KT, SF and IS were evaluated on the 10 guesses, reporting min-max ranges for KT and SF, and mean values for IS.

3.1.1. Intrinsic architectural differentiation

Our experimental analysis on the multiple time-scales differentiation is conducted by firstly considering fixed values of the leaky parameter $a = 0.55$ and of the spectral radius $\rho = 0.9$ among the layers (or sub-reservoirs), using values that are intentionally not optimized, as the purpose is not to achieve the best results, but to show the differences occurring among the different architectures under the same conditions. Fig. 3 graphically shows the results achieved on the Artificial dataset with deepESN, deepESN-IA and groupedESN. Continuous blue lines refer to the different layers of the deep architecture (different sub-reservoirs for groupedESN), with darker colors corresponding to higher layers. For the sake of comparison, the red dashed line refers to the shallow ESN baseline with $a = 0.55$ and $\rho = 0.9$, as in every layer of the deep networks. Table 1 reports the values of KT, SF and IS achieved by deepESN, deepESN-IA and groupedESN on both the Artificial and the Wikipedia datasets.

The intrinsic differentiation among the time-scales dynamics at the different layers of a deepESN is qualitatively analyzed through the plot shown in Fig. 3(a), from which it is possible to observe

Table 1
Values of Kendall's tau (KT), Spearman's footrule (SF) and index of separation (IS) achieved on the Artificial dataset and on the Wikipedia dataset by deepESN, deepESN-IA and groupedESN with $a = 0.55$ and $\rho = 0.9$ for every layer (or sub-reservoir). For KT and SF smaller values are better, for IS higher values are better.

Model	KT (min–max)	SF (min–max)	IS (mean)
Artificial dataset			
deepESN $a = 0.55$	0–2	0–2	203.90 (± 83.39)
deepESN-IA $a = 0.55$	0–2	0–2	134.10 (± 37.68)
groupedESN $a = 0.55$	8–10	26–42	18.70 (± 56.56)
Wikipedia dataset			
deepESN $a = 0.55$	0–2	0–2	175.70 (± 98.48)
deepESN-IA $a = 0.55$	0–4	0–4	123.90 (± 22.48)
groupedESN $a = 0.55$	7–10	22–44	−22.40 (± 58.16)

that the effects of the input perturbation last longer for higher layers in the stack. Such differentiation is indeed related to the layered deep architecture, as it is strongly attenuated when the external input is provided to each layer, as in the case of deepESN-IA (see Fig. 3(b)) or when layering is removed from the architectural design, as in the case of groupedESN (see Fig. 3(c)). These insights are quantitatively confirmed by the results in Table 1, showing that deepESN provides a preferable differentiation of time-scales at the different layers, in this regard generally presenting a more ordered organization of time-scales, i.e. smaller values of KT and SF, and better separability, i.e. larger values of IS.

In particular, a comparison between the behaviors of deepESN and deepESN-IA points out the relevance of having deeper layers at increasing distances from the external input as a key architectural factor for time-scales separation. Note that deepESN and deepESN-IA show a similar hierarchical organization of time-scales (similar values of KT and SF) and in both cases the higher layers of the architecture present longer time-scales than the corresponding standard ESN (as can be seen in the plots of Fig. 3(a) and (b)). However deepESN-IA shows a reduced separation of time-scales with respect to deepESN, as can be seen graphically through a comparison of Fig. 3(a) and (b), and also numerically in Table 1, with deepESN-IA leading to smaller IS values than deepESN.

The case of groupedESN, illustrated in Fig. 3(c), shows the intrinsic variability that can be already present in (sub-)reservoirs with the same hyper-parameterization when they are not organized in a stack. As can be seen, in this case the dynamics of all the sub-reservoirs do not present a particular ordering and have a similar behavior to the one of the shallow ESN with corresponding total number of units and values of the hyper-parameters. Quantitatively, a comparison between the results of deepESN and groupedESN in Table 1, shows the inherent impact in terms of ordering and separability among the time-scales dynamics (smaller values for KT and SF, larger values of IS) that are due to the hierarchical organization of the reservoir layers in deepESN. Notice that in this setting, in which there are no hierarchies among the sub-reservoirs of a groupedESN (they all have the same hyper-parameterization), the use of different grades of colors in Fig. 3(c) and the values of KT, SF and IS reported in Table 1 for groupedESN assume a different meaning than in the case of layered architectures. The results of groupedESN in this case is indeed representative of a completely un-ordered sub-reservoir organization and are therefore reported for the sake of completeness and scale comparison.

3.1.2. *Differentiation by variation of RC hyper-parameters*

In light of the results shown in Section 3.1.1, we can observe that the inherent diversification among the layers dynamics in a deepESN is quite narrow (Fig. 3(a)), with the range of emerging time-scales presenting a limited extent. Such differentiation can be emphasized within the efficient RC approach by resorting to the strategies proposed in Section 2.2.

We first take into consideration the effect due to a diversification of the value of the leaky parameter among the layers. Fig. 4 shows the results achieved by deepESN, deepESN-IA and groupedESN using a fixed value of $\rho = 0.9$ and decreasing values of the leaky parameter a for increasing layer depth, from 1 to 0.1, thus imposing a progressively slower speed of reservoir dynamics at higher layers in the architecture. Table 2 reports the KT, SF and IS values obtained by deepESN, deepESN-IA and groupedESN in the same conditions.

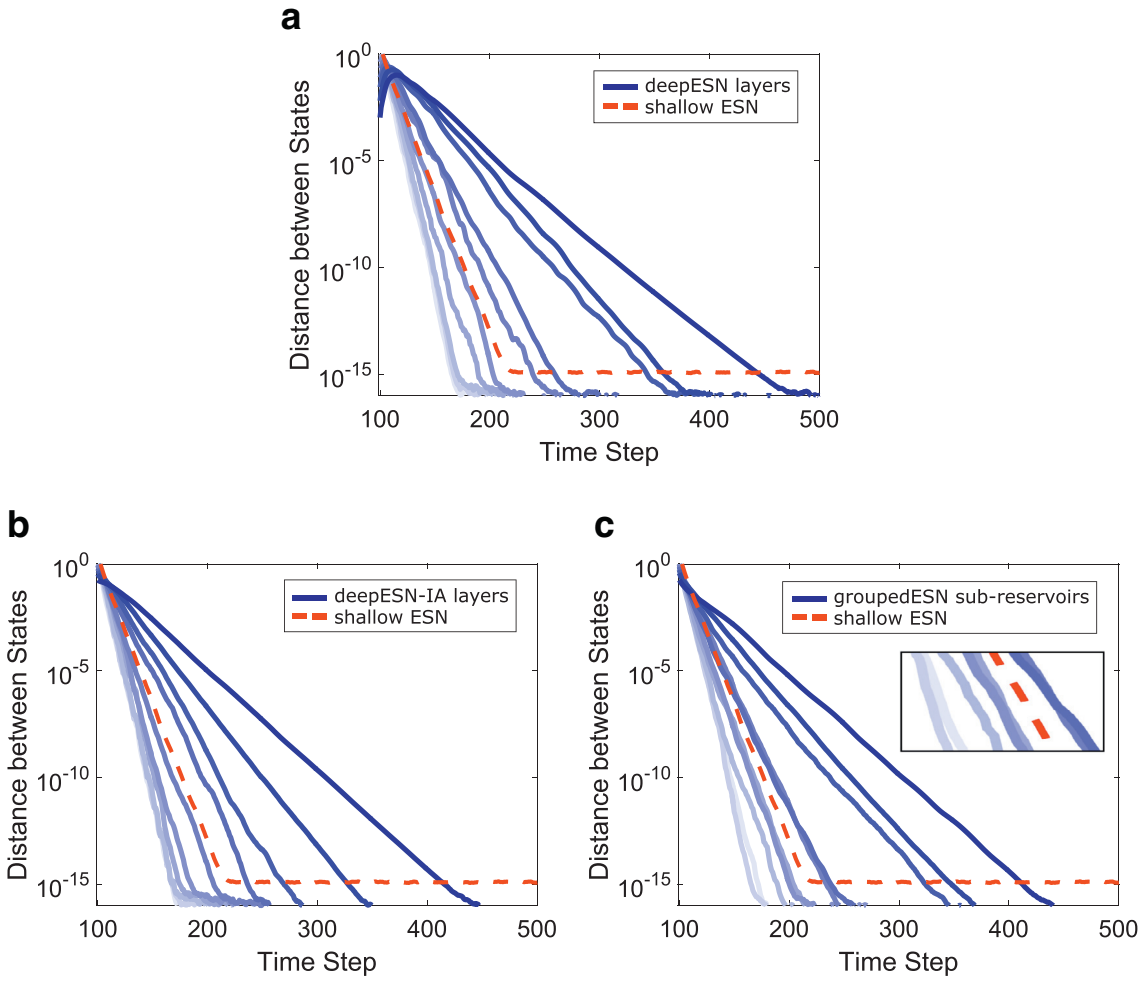


Fig. 4. Distance between perturbed and unperturbed states on the Artificial dataset for the considered RC architectures with $\rho = 0.9$ for every layer (or sub-reservoir) and a varying from 1 to 0.1 among the layers (or sub-reservoirs). Continuous blue lines correspond to layers in deep RC networks and to sub-reservoirs in groupedESN. Darker colors correspond to decreasing values of a, and to higher layers in deep RC networks. Dashed red lines correspond to the shallow ESN (for graphical reference with respect to Fig. 3, see text). (a) deepESN, (b) deepESN-IA, (c) groupedESN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Values of Kendall's tau (KT), Spearman's footrule (SF) and index of separation (IS) achieved on the Artificial dataset and on the Wikipedia dataset by deepESN, deepESN-IA and groupedESN with $\rho = 0.9$ for every layer (or sub-reservoir) and a varying from 1 to 0.1 among the layers (or sub-reservoirs). For KT and SF smaller values are better, for IS higher values are better.

Model	KT (min–max)	SF (min–max)	IS (mean)
Artificial dataset			
deepESN	0–0	0–0	367.80 (± 76.25)
deepESN-IA	0–2	0–2	294.30 (± 44.51)
groupedESN	2–9	4–18	285.00 (± 50.07)
Wikipedia dataset			
deepESN	0–2	0–2	335.50 (± 92.69)
deepESN-IA	0–2	0–2	295.00 (± 42.54)
groupedESN	4–9	4–18	298.10 (± 48.86)

For the sake of comparison, in each plot of Fig. 4 it is reported also the result obtained by standard shallow ESN with values of $\rho = 0.9$, as in every layer of the deep RC networks, and $a = 0.55$, i.e. the average value among the layers of the deep architectures. Such result is reported here (and also in the following analysis) as a summary for the values and comparisons already discussed with regard to Fig. 3, as indeed the aim is to assess the extent of the differentiation among the behaviors shown by the different layers, also in comparison to the average case.

As can be seen in Fig. 4(a), the variability of the leaky parameter has a great impact on the differentiation among the emerging time-scales dynamics, showing a much wider extent of the ordered diversification than deepESN with fixed values of a (Fig. 3(a)). As can be seen by comparing Tables 2 and 1, varying the value of the leaky parameter across the layers of a deepESN results in generally lower values of KT and SF and higher values of IS.

This characterization is a result of the interplay between layering and leaky integration variability, and also in this case it is strongly reduced when all the layers are at the same distance from the input, i.e. for deepESN-IA, or when non-stacked architectures are considered, i.e. for groupedESN. Specifically, also in this case, deepESN-IA leads to a reduced separation of time-scales across the layers, while groupedESN in addition to the reduced separation also results in a worse ordering with respect to the duration of the perturbation effect, which is graphically pointed out by the overlapping among the curves in Fig. 4(c) (highlighted in the zoom), and by the results in Table 2.

The effect due to the variation of the ρ hyper-parameter among the layers is similar to the case of variable a, though with a reduced extent of differentiation. For the sake of succinctness, results concerning the ρ hyper-parameter are omitted here and are reported in Appendix B.

3.1.3. Differentiation by IP training

The impact on the development of multiple time-scales due to the unsupervised IP training is shown in Fig. 5 and Table 3, considering the cases of deepESN, deepESN-IA and groupedESN with

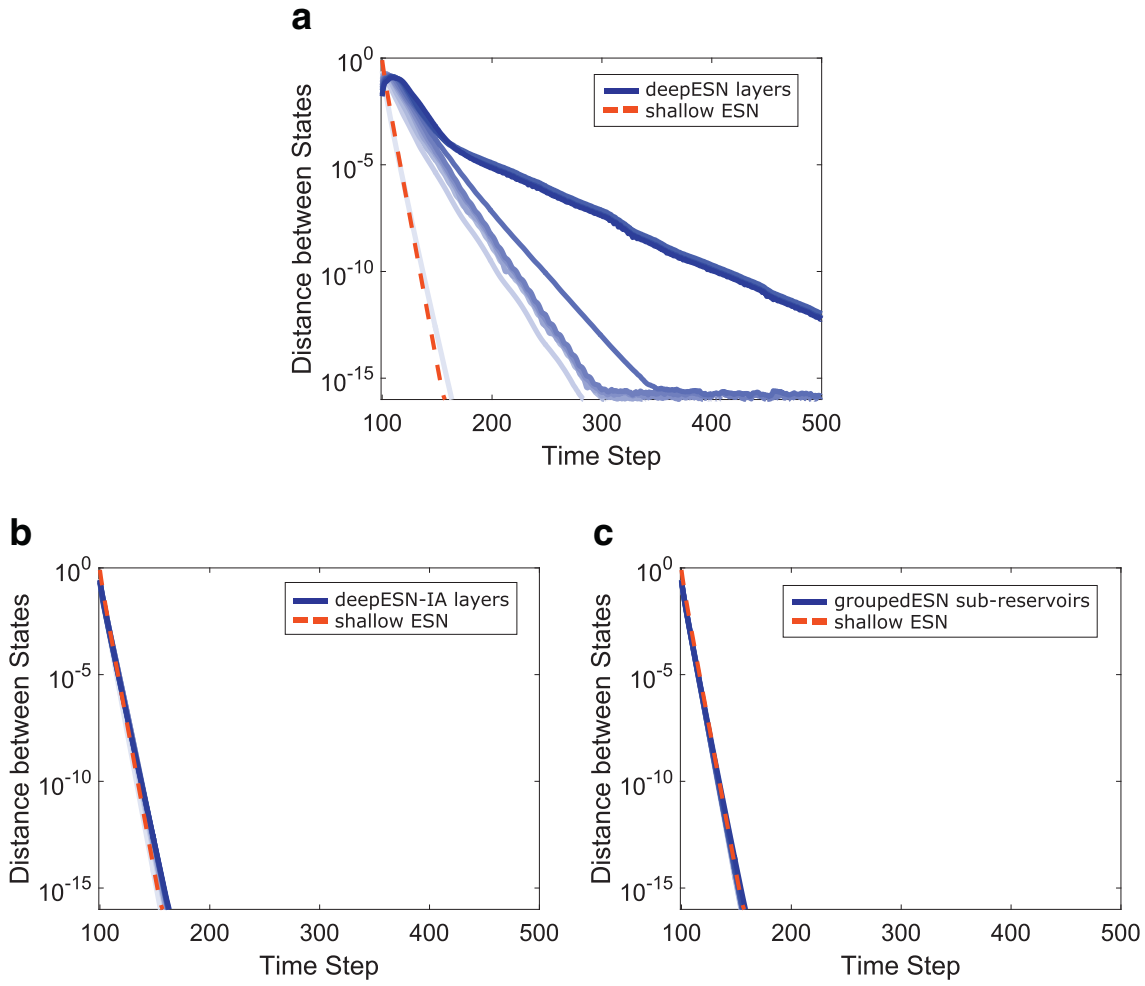


Fig. 5. Distance between perturbed and unperturbed states on the Artificial dataset for the considered RC architectures, with $a = 0.55$ and $\rho = 0.9$ for every layer (or sub-reservoir) and using IP learning. Continuous blue lines correspond to layers in deep RC networks (darker colors for higher layers) and to sub-reservoirs in groupedESN. Dashed red lines correspond to the shallow ESN with the same number of total reservoir units and hyper-parameterization. (a) deepESN, (b) deepESN-IA, (c) groupedESN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Values of Kendall's tau (KT), Spearman's footrule (SF) and index of separation (IS) achieved on the Artificial dataset and on the Wikipedia dataset by deepESN, deepESN-IA and groupedESN, with $a = 0.55$ and $\rho = 0.9$ for every layer (or sub-reservoir) and using IP learning. For KT and SF smaller values are better, for IS higher values are better.

Model	KT (min–max)	SF (min–max)	IS (mean)
Artificial dataset			
deepESN	0–2	0–2	785.10 (± 248.97)
deepESN-IA	0–7	0–14	24.00 (± 8.99)
groupedESN	6–10	20–44	−0.40 (± 3.23)
Wikipedia dataset			
deepESN	0–0	0–0	644.40 (± 183.61)
deepESN-IA	2–8	2–14	20.90 (± 4.66)
groupedESN	6–10	10–44	−1.80 (± 6.84)

constant values of $a = 0.55$ and $\rho = 0.9$ for all the layers (or sub-reservoirs), and using IP learning. In our experimental setting, we used values of $\mu = 0$, $\sigma = 0.1$ and $\eta = 0.00001$ for the IP parameters in Eq. (3). For comparison, in the plots of Fig. 5 we also show the result obtained by the corresponding standard shallow ESN architecture using IP with and the same hyper-parameterization.

The remarkable effect of IP on the time-scales differentiation in a layered architecture is pointed out by a comparison between the results of deepESN under the same settings of a and ρ, with IP learning (Fig. 5(a)) and without IP learning (Fig. 3(a)). It can be observed that after IP training the higher layers in the deepESN architecture tend to forget more slowly the past input history, and the effect of the typo perturbation has a much longer duration. Results in Table 3 show that deepESN with IP achieves better results in terms of time-scales ordering and separation among the layers, outperforming the results of the base deepESN case with corresponding settings (in Table 1).

In addition to the amplifying effect of IP on the time-scales differentiation observed on deepESN, it is also possible to notice the enhancement effect of layering on the IP efficacy. Indeed the hierarchical organization of reservoir layers in deepESN, with higher layers at increasing distance from the input, allows to trigger a process of increasing effectiveness of IP among the layers, as can be seen also by the fact that the curves representing the dynamics of the first deepESN layer and of the shallow ESN almost overlap in the plot in Fig. 5(a). On the other hand, when the deepESN architectural characterizations are lost, layer dynamics are made more uniform by IP learning, as can be seen for deepESN-IA (Fig. 5(b)) and groupedESN (Fig. 5(b)).

The strong effect of IP on the emerging of multiple time-scales differentiation in deepESN can be explained in terms of a diversification of the memory length in the different layers, similarly to the effect of the variation of the spectral radius. Indeed, by changing the gains of the reservoir units' activation functions, IP potentially act on the real value of the spectral radius at the different layers, as noticed also in [17] for standard RC networks.

3.2. Richness of reservoir dynamics: IP training and layering

The role of IP learning in relation to layering then deserves to be further investigated in the context in which it has been introduced, i.e. the information maximization of reservoir state dynamics [16–18]. To this aim, we evaluated the entropy of reservoir units activations over time, as a measure of the richness of state dynamics, assessing the effect of IP in conjunction with layering. We approximated the entropy of the output distribution of each reservoir unit i, by computing the integral estimate [28] H_i:

$$H_i = -\int f_i(x) \log f_i(x) dx \tag{9}$$

where f_i is the estimate of the probability density function of the ith reservoir unit output distribution over time, computed by means of kernel density estimation, and the integral in Eq. (9) is computed by numerical integration.[2] The layer-wise effect of IP on the entropy of reservoir units activations is graphically shown in Fig. 6 for both the Artificial (Fig. 6(a)) and the Wikipedia (Fig. 6(b)) datasets. The plots show the values of the entropy averaged on the units of each layer (or sub-reservoir) for deepESN, deepESN-IA and groupedESN, using the same experimental setting considered with regard to Fig. 5 and Table 3 (i.e. $a = 0.55$, $\rho = 0.9$, $\mu = 0$ and $\sigma = 0.1$). For the sake of comparison, Fig. 6 also shows the entropy achieved by a shallow ESN (with the same total number of reservoir units) under the same conditions. It can be seen that for both the datasets, the entropy of deepESN states clearly increases with increasing layer depth, showing the same incremental IP effect already observed in Fig. 5(a). On the other hand, in the cases of deepESN-IA and groupedESN, the entropy remains almost constant among the layers (or sub-reservoirs), and very close to the values corresponding to a standard shallow ESN.

The values of the state entropy averaged over all the reservoir units in deepESN, deepESN-IA, groupedESN and ESN in the same

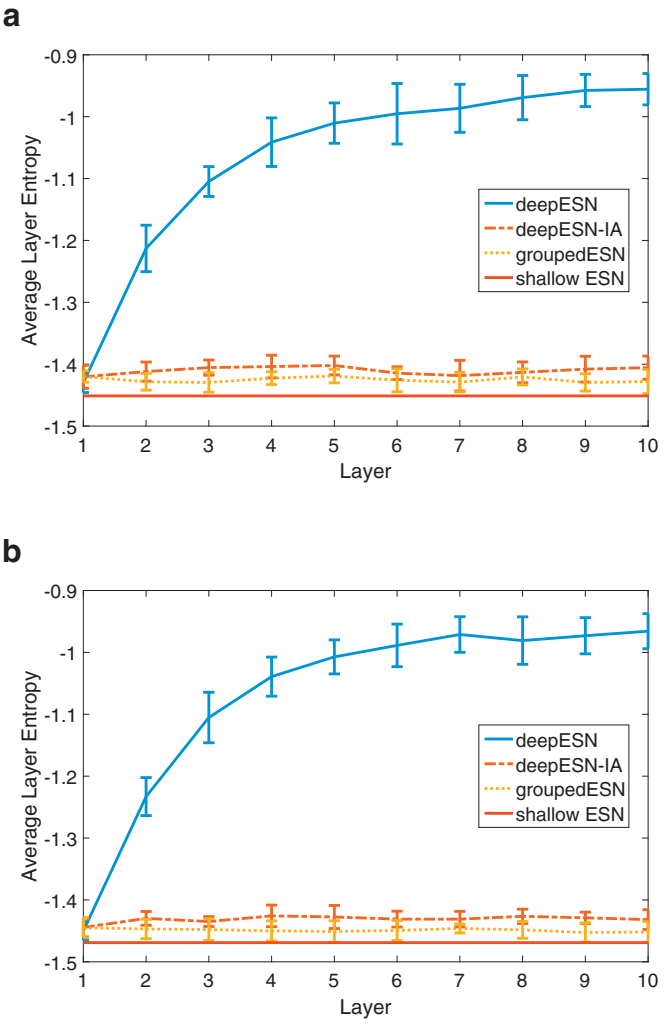


Fig. 6. Layer-wise averaged entropy of reservoir states on the Artificial dataset (a) and on the Wikipedia dataset (b), computed for deepESN, groupedESN and deepESN-IA, with $a = 0.55$ and $\rho = 0.9$ for every layer (or sub-reservoir) and IP learning. For groupedESN the results refer to sub-reservoirs. The average entropy of the shallow ESN counterpart is reported as a continuous red line across each plot.

[2] Note that the H_i values computed by means of Eq. (9) result in approximations of the Shannon's differential entropy, which can also assume negative values.

Table 4
Average entropy of reservoir states on the Artificial and on the Wikipedia datasets (higher values are better), obtained by deepESN, groupedESN and deepESN-IA,with $a = 0.55$ and $\rho = 0.9$ for every layer (or sub-reservoir) and IP learning. The average Entropy of corresponding shallow ESN is reported as well for the sake of reference comparison.

Model	Entropy
Artificial dataset	
deepESN	−1.066 ± 0.021
deepESN-IA	−1.410 ± 0.007
groupedESN	−1.425 ± 0.005
shallow ESN	−1.451 ± 0.003
Wikipedia dataset	
deepESN	−1.071 ± 0.019
deepESN-IA	−1.431 ± 0.006
groupedESN	−1.449 ± 0.004
shallow ESN	−1.469 ± 0.005

experimental settings are reported in Table 4. From such results it is possible to appreciate the overall strong impact of IP on the hierarchical architecture of deepESN, resulting in an average entropy improvement of ≈ 27% with respect to the shallow ESN for both the datasets, whereas deepESN-IA and groupedESN obtained results very close to those of shallow ESN. Results in Table 4 confirm that in this experimental setting the effectiveness of IP is enhanced only by using a hierarchical reservoir organization with layers at increasing distance from the input. When IP is applied to layers of reservoir units at the same distance from the input, or to non-stacked sub-groups of reservoir units, analogous results to the application of IP to a shallow ESN are achieved.

3.3. Short-term memory capacity

A last set of experiments has been considered to assess the effectiveness of the proposed approaches on the MC task [29]. This task provides a measure of short-term memory capacity of RC networks, by evaluating how well it is possible to recall delayed versions of the input based on reservoir activations. Input consists of a temporal signal whose elements $u(t)$ are drawn from a uniform distribution over $[-0.8, 0.8]$. The task requires to reconstruct the input stream with increasing delays, i.e. for each time step t we consider target values $\bar{y}_k(t) = u(t-k)$, for $k = 0, \ldots, \infty$. The overall MC is defined as:

$$MC = \sum_{k=0}^{\infty} r^2(u(t-k), y_k(t)) \tag{10}$$

where $r^2(u(t-k), y_k(t))$ is the squared correlation coefficient between the input with delay k and the corresponding re-constructed value $y_k(t)$. In practice, due to theoretical results on RC networks [29], the MC can be computed by considering only a finite number of delayed signals. In this paper we set up an MC task similarly to [16], by considering a number of delays equal to 200 (i.e. twice the number of total reservoir units considered). The input signal contained 6000 steps, 5000 of which used for training and the remaining 1000 for test. For this task we adopted similar settings to those already used for previous experiments. In particular, we considered deepESN architectures with 10 reservoir layers of 10 fully connected units and input scaling $scale_{in} = 0.1$, instantiating the networks with constant values of the leaky parameter $a \in \{0.1, 0.55, 1\}$ and of the spectral radius $\rho \in \{0.1, 0.5, 0.9\}$ among the layers. Moreover, we considered deepESN settings in which the value of a varies from 1 to 0.1 among the layers, with constant values for $\rho \in \{0.1, 0.5, 0.9\}$, and in which the value of ρ varies among the layers from 0.1 to 0.9, with constant values of $a \in \{0.1, 0.55, 1\}$. We also ran experiments using IP learning, which has a known improvement effect on the MC task [16], using values of $\eta = 0.00001$, $\mu = 0$ and $\sigma \in \{0.1, 0.01\}$ for all the RC settings mentioned above. Analogous experiments were conducted for deepESN-IA and groupedESN, as well as for standard ESN (for the sole scope of baseline reference and assessment). For each network hyper-parameterization, we independently generated 10 guesses, averaging the results over such guesses. In all the considered cases, the values of a, ρ and σ were chosen by model selection on a validation set (comprising 20% of the data in the training set).

Table 5
Memory capacity results (higher is better) achieved by deepESN, deepESN-IA and groupedESN and shallow ESN. Results for deepESN are reported also for the cases of layers with decreasing values of a (var. a) and increasing values of ρ (var. ρ) among the layers. The first group of results refers to RC models without the use of IP, the second group refers to the corresponding models with IP (denoted by +IP).

Model	Memory capacity
deepESN	42.45 ± 3.11
deepESN + var. a	37.15 ± 2.48
deepESN + var. ρ	30.79 ± 1.15
deepESN-IA	28.05 ± 1.87
groupedESN	28.02 ± 1.77
shallow ESN	27.50 ± 1.34
deepESN + IP	54.49 ± 3.82
deepESN + var. a + IP	52.03 ± 5.43
deepESN + var. ρ + IP	48.01 ± 3.36
deepESN-IA + IP	36.78 ± 2.69
groupedESN + IP	39.02 ± 2.25
shallow ESN + IP	37.06 ± 1.48

The MC values on the test set achieved by deepESN, deepESN-IA, groupedESN and shallow ESN are reported in Table 5. Results show that deepESN obtained the best MC both without IP and with IP, improving the results obtained by shallow ESNs (which are in line with literature results [16]). In particular, without IP, deepESN obtained an MC value of 42.45 that represents an improvement of ≈ 54% with respect to the value achieved by shallow ESN. The hierarchical organization of deepESN architecture also allowed to exalt the known effect of IP on the MC, leading to a value of 54.49, which improves the result achieved by shallow ESN with IP by ≈47%. In both the cases, without and with IP, the selected values of spectral radius and leaky parameter for deepESN were $a = 1$ and $\rho = 0.9$, while varying the values of these two parameters among the layers led to slightly lower MC results. Moreover, notice that deepESN-IA and groupedESN achieved MC values very close to the one of shallow ESN, both with and without IP.

A further comparison between the MC of deepESN and shallow ESN is presented in Fig. 7, which shows the results achieved in correspondence of different values of the leaky parameter a and of the spectral radius ρ (constant for all the layers) while the values of the other parameters were selected on the validation set, without using IP (Fig. 7(a) and (b)) and using IP (Fig. 7(c) and (d)). Results clearly show that deepESN improves the short-term MC of shallow ESN in all the cases.

4. Conclusions

In this paper we have proposed an experimental analysis of state dynamics in deep recurrent neural architectures, targeted at assessing the real effect of layering on the development of a hierarchical representation of the temporal information. In particular, the recourse to RC networks allowed us to conduct such analysis separately from learning aspects.

Despite the observations that stacking recurrent layers is just an architectural constraint to a fully connected RNN, and that a shallow reservoir already provides a rich pool of varied state dynamics by construction, the experimental evidences in this paper have shown that it is possible to exploit the same factors that influence the dynamics of shallow recurrent architectures to achieve a temporal data representation at multiple levels of abstractions through a layered network organization.

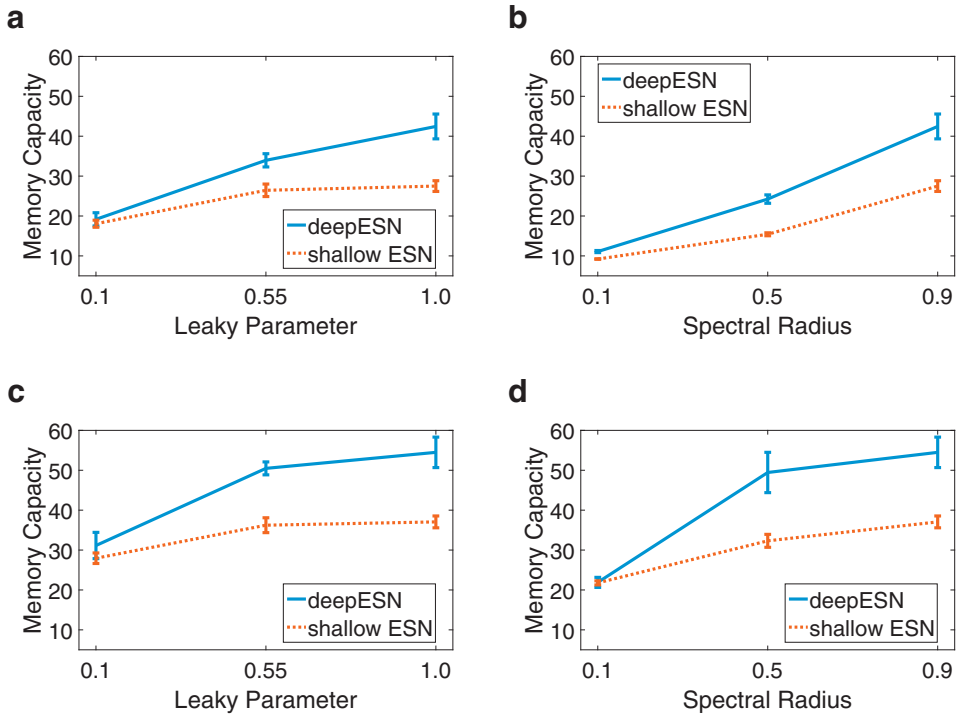


Fig. 7. MC results of deepESN and shallow ESN for different values of the spectral radius ρ and of the leaky parameter a. (a) Different values of a, without IP, (b): different values of ρ, without IP, (c) different values of a, with IP, (d) different values of ρ, with IP. For deepESN each reservoir layer has the same hyper-parameterization.

In particular, the introduction of the deepESN model allowed us to study the intrinsic properties of deep layered recurrent architectures in terms of time-scales differentiation, and highlight such properties in comparison to purposely introduced baseline models. This allowed us to evaluate the effect of architectural factors such as the progressive distance of higher layers from the external input (versus deepESN-IA) and the effective hierarchical interplay among layers of recurrent units (versus groupedESN).

Experiments on two benchmark datasets have shown the synergy between stacking reservoir layers and the role of already known RC parameters. On the one hand, such analysis provided insights on the amplification of the effect of these RC parameters in a deep architecture. On the other hand, it allowed us to propose effective strategies to enhance the time-scale differentiation among layers using different values of the leaky parameter and of the spectral radius, or by unsupervised IP learning focused only the parameters of the activation function. This allows us to preserve the efficiency of the RC approach, without resorting to a full RNN training (extended to all the units parameters).

More in detail, the variability of parameters of reservoir design ruling the speed of dynamics in response to the input, i.e. the leaky parameter, and the memory length, i.e. the spectral radius, could effectively amplify the emergence of multiple (separated) time-scales, hierarchically ordered across the layers of a deepESN.

The use of an efficient technique for unsupervised adaptation of (only) the parameters of the reservoir activation functions, i.e. IP learning, has shown a great impact on the development of multiple time-scales (enhanced in deep models). Such impact has been investigated also in terms of improved richness of reservoir dynamics by measuring the entropy of reservoir state activations, showing that the known effect of IP learning is actually progressively enhanced among the layers of a deepESN architecture. Furthermore, the advantages brought by the proposed approaches have been shown also on the MC task, showing that deepESN allows to improve the short term memory capacity with respect to the shallow case, and that the known effect of IP learning on the MC task is greatly exalted by the use of a layered architecture.

Overall, after assessing the intrinsic architectural properties of general deep layered RNN in representing different time-scale dynamics, more interestingly for RC modeling, the results of our experimental analysis pointed out the actual relevance of the interplay between layering and RC parameters aspects on the diversification of temporal representations. In particular, the proposed approaches allowed us to achieve a time-scale differentiation in deep models that is higher with respect to a standard ESNs without a layered structure, and led to explicitly address the concept of including temporal data representation at different level of abstraction within the RC paradigm.

As such, the analysis proposed in this paper paves the way to further studies on the design of novel deep neural network models for efficient representation learning on sequences. Future developments deserve to move from the current insights to the design and concrete set up of new learning models boosted by an enriched representation of the input dynamics, exploiting the time-scale differentiation developed through the layers to solve complex tasks that require/involve processing time-series data at different levels of time granularity. The opening of this line of research would contribute to achieving new findings that are demanded to result in a relevant breakthrough in the area of efficiently learning from sequential and temporal data.

Appendix A. Qualitative and quantitative measures of time-scales differentiation

In this section, we provide details on the measures used in Section 3.1 to evaluate the goodness of time-scales differentiation among the layers of a stacked RC architecture.

Taking into consideration a deepESN with N_L layers and 2 sequences, the unperturbed one S_1 and perturbed one S_2 (in which a typo is inserted with respect to S_1 at step $t = 100$), here we denote by $\mathbf{x}_u^{(l)}(t)$ and $\mathbf{x}_p^{(l)}(t)$ the state of layer l at step t for the unperturbed and the perturbed sequence, respectively. For each layer l, we evaluated the Euclidean distance between corresponding states $\mathbf{x}_u^{(l)}(t)$ and $\mathbf{x}_p^{(l)}(t)$ as a function of time, i.e. $D^{(l)}(t) =$

$\|\mathbf{x}_u^{(l)}(t) - \mathbf{x}_p^{(l)}(t)\|_2$. Then we plotted the distances $D^{(l)}(t)$ for $t \geq 100$ and for all the layers, in order to graphically investigate how long the effect of the input perturbation at step 100 affects the state dynamics of each layer, providing a *qualitative* analysis of the time-scales differentiation emerging in the architecture. Analogous plots can of course be obtained also for the cases of deepESN-IA and groupedESN.

The qualitative investigation described above is completed by adopting *quantitative* measures of time-scales diversification. To this aim, the maximum duration of the perturbation effect on layer l can be expressed as $P^{(l)} = \max_t(D^{(l)}(t) > 0)$. By ordering the set of values $\{P^{(l)}\}_{l=1}^{N_L}$, we can define a ranking on the layers based on the duration of the perturbation effect, denoted by $\{O^{(l)}\}_{l=1}^{N_L}$, and which represents a permutation of $\{1, 2, \ldots, N_L\}$. Specifically, if $O^{(l)} = n$ it means that the layer l is the nth one in terms of duration of the input perturbation effect. In this sense, the ideal case is represented by the identity permutation ranking $1, 2, \ldots, N_L$, i.e. $O^l = l$ for every $l = 1, \ldots, N_L$, corresponding to an increasing duration of the perturbation effect for higher layers. Based on these definitions, we can quantify the quality of the ordering among the time-scales in the network's layers by measuring the distance between the ranking $\{O^{(l)}\}_{l=1}^{N_L}$ and the identity permutation ranking. To do so, we adopt two known distances between rankings [26], i.e. Kendall's tau and Spearman's footrule distances, respectively denoted by *KT* and *SF*, and computed according to:

$$KT = |\{(l_1, l_2) : (1 \leq l_1 < l_2 \leq N_L) \wedge (O^{(l_1)} > O^{(l_2)})\}|$$
$$SF = \sum_{l=1}^{N_L} |l - O^{(l)}|, \tag{A.1}$$

where, with respect to the identity permutation, KT sums the number of required pairwise swaps, while SF sums the total amount of displacement of the elements in the ranking. Accordingly, smaller values of KT and SF denote better orderings of the times-scales.

Moreover, we can quantify the extent of time-scales separation by measuring the distances between the duration of the perturbation in consecutive layers, introducing an index of separation, denoted by IS, and computed as:

$$IS = \sum_{l=2}^{N_L} P^{(l)} - P^{(l-1)}, \tag{A.2}$$

where higher values of IS correspond to a greater spacing among the duration of the perturbation effect in the different layers.

Appendix B. Time-scales differentiation: variation of ρ

This section provides the results on time-scales differentiation due to the variability of the spectral radius ρ among the layers of a stacked RC network.

Fig. B.8 shows the results achieved by deepESN, deepESN-IA and groupedESN using a fixed value of $a = 0.55$ and increasing

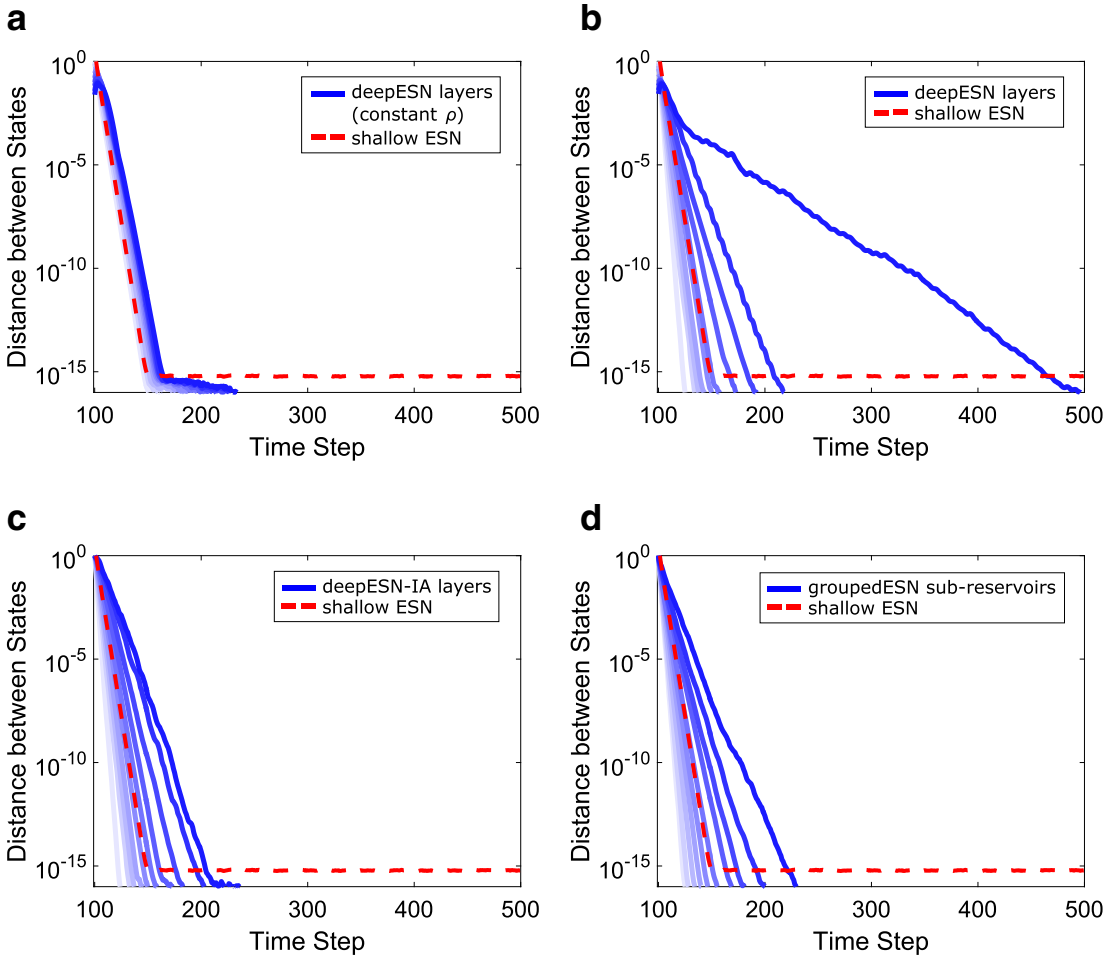


Fig. B.8. Distance between perturbed and unperturbed states on the Artificial dataset for the considered RC architectures with $a = 0.55$ for every layer (or sub-reservoir) and ρ varying from 0.1 to 0.9 among the layers (or sub-reservoirs). Continuous blue lines correspond to layers in deep RC networks and to sub-reservoirs in groupedESN. Darker colors correspond to increasing values of ρ, and to higher layers in deep RC networks. Dashed red lines correspond to the shallow ESN (for graphical reference with respect to Fig. 3, see text). For the sake of reference the results corresponding to deepESN with constant $\rho = 0.5$ for all the layers is reported as well. **a**: deepESN with constant ρ, (b) deepESN, (c) deepESN-IA, (d) groupedESN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table B.6
Values of Kendall's tau (KT), Spearman's footrule (SF) and index of separation (IS) achieved on the Artificial and on the Wikipedia datasets by deepESN, deepESN-IA and groupedESN with $a = 0.55$ for every layer (or sub-reservoir) and ρ varying from 0.1 to 0.9 among the layers (or sub-reservoirs). For the sake of reference comparison the results achieved for the case of deepESN with constant $\rho = 0.5$ for all the layers is reported as well. For KT and SF smaller values are better, for IS higher values are better.

Model	KT (min–max)	SF (min–max)	IS (mean)
Artificial dataset			
deepESN $\rho = 0.5$	0–3	0–4	68.20 (± 21.16)
deepESN var. ρ	0–2	0–2	161.90 (± 129.19)
deepESN-IA var. ρ	0–4	0–4	95.20 (± 24.02)
groupedESN var. ρ	0–6	0–6	95.40 (± 22.57)
Wikipedia dataset			
deepESN $\rho = 0.5$	0–6	0–6	54.90 (± 17.43)
deepESN var. ρ	0–2	0–2	168.00 (± 69.95)
deepESN-IA var. ρ	0–2	0–2	92.80 (± 18.75)
groupedESN var. ρ	0–4	0–6	100.70 (± 39.86)

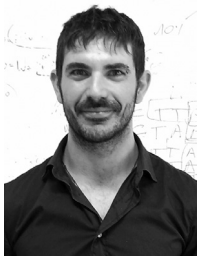
values of the spectral radius ρ for increasing layer depth, from 0.1 to 0.9, resulting in increasing memory length for higher layers. In Fig. B.8 we also show the result of standard shallow ESN with a and ρ equal to the corresponding averages among the layers of the deep architectures. Table B.6 reports the KT, SF and IS values obtained by deepESN, deepESN-IA and groupedESN in the same conditions. For the sake of reference comparison, Fig. B.8 and Table B.6 also report the results obtained by deepESN with constant value of $\rho = 0.5$, i.e. the average among the ρ values in the considered range of variability.

The effect of the spectral radius variation can be appreciated by comparing the results obtained for the cases of deepESN with constant ρ for every layer (Fig. B.8(a)) and of deepESN with ρ varying among the layers (Fig. B.8(b)). As can be seen, varying the value of ρ leads to a clear improvement of the hierarchical time-scales differentiation, as also reflected by the values in Table B.6, with deepESN using different values of ρ achieving better results than deepESN with constant ρ in terms of KT, SF and IS values.

A comparison among the considered cases with ρ varying among the layers (or sub-reservoirs) of the architecture, i.e. deepESN (Fig. B.8(b)), deepESN-IA (Fig. B.8(c)) and groupedESN (Fig. B.8(d)), confirmed also by the results on KF, SF and IS values in Table B.6, shows similar results to the case of variable a (analyzed in Section 3.1.2, in Fig. 4 and Table 2), though the effect of differentiation in this case is less significant.

References

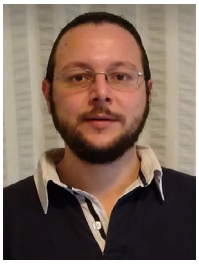
[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[2] L. Deng, D. Yu, Deep learning, Signal Process. 7 (2014) 3–4.
[3] Y. Bengio, Learning deep architectures for AI, Found. Trends® Mach. Learn. 2 (1) (2009) 1–127.
[4] I. Goodfellow, Y. Bengio, A. Courville, in: Deep Learning, MIT Press, 2016. Book in preparation http://www.deeplearningbook.org.
[5] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[6] R. Pascanu, Ç. Gülçehre, K. Cho, Y. Bengio, in: How to construct deep recurrent neural networks, 2014, pp. 1–13. arXiv preprint 1312.6026v5.
[7] S.E. Hihi, Y. Bengio, Hierarchical recurrent neural networks for long-term dependencies., in: Proceedings of the 1995 Conference on Neural Information Processing Systems (NIPS), 1995, pp. 493–499.
[8] M. Hermans, B. Schrauwen, Training and analysing deep recurrent neural networks, in: Proceedings of the 1995 Conference on Neural Information Processing Systems (NIPS), 2013, pp. 190–198.
[9] A. Graves, A.-R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 6645–6649.
[10] M. Mozer, Induction of multiscale temporal structure, Adv. Neural Inf. Process. Syst. 4 (1993) 275–282.
[11] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks., in: Proceedings of the 2013 International Conference on Machine Learning (ICML), 28, 2013, pp. 1310–1318.
[12] F. Triefenbach, A. Jalalvand, K. Demuynck, J.-P. Martens, Acoustic modeling with hierarchical reservoirs, IEEE Trans. Audio Speech Lang. Process. 21 (11) (2013) 2439–2450.
[13] P. Angelov, A. Sperduti, Challenges in deep learning, in: Proceedings of the Twenty-fourth European Symposium on Artificial Neural Networks (ESANN), 2016, pp. 489–495. i6doc.com
[14] M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comput. Sci. Rev. 3 (3) (2009) 127–149.
[15] D. Verstraeten, B. Schrauwen, M. d'Haene, D. Stroobandt, An experimental unification of reservoir computing methods, Neural Netw. 20 (3) (2007) 391–403.
[16] B. Schrauwen, M. Wardermann, D. Verstraeten, J. Steil, D. Stroobandt, Improving reservoirs using intrinsic plasticity, Neurocomputing 71 (7) (2008) 1159–1171.
[17] J.J. Steil, Online reservoir adaptation by intrinsic plasticity for backpropagation–decorrelation and echo state learning, Neural Netw. 20 (3) (2007) 353–364.
[18] J. Triesch, A gradient rule for the plasticity of a neurons intrinsic excitability, in: Proceedings of the 2005 International Conference on Artificial Neural Networks, Springer, 2005, pp. 65–70.
[19] C. Gallicchio, A. Micheli, Deep reservoir computing: A critical analysis, in: Proceedings of the Twenty-fourth European Symposium on Artificial Neural Networks (ESANN), 2016, pp. 497–502. i6doc.com
[20] H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication, Science 304 (5667) (2004) 78–80.
[21] C. Gallicchio, A. Micheli, Architectural and Markovian factors of echo state networks, Neural Netw. 24 (5) (2011) 440–456.
[22] P. Tiňo, B. Hammer, M. Bodén, Markovian bias of neural-based architectures with feedback connections, in: Perspectives of Neural-Symbolic Integration, Springer, 2007, pp. 95–133.
[23] H. Jaeger, M. Lukoševičius, D. Popovici, U. Siewert, Optimization and applications of echo state networks with leaky-integrator neurons, Neural Netw. 20 (3) (2007) 335–352.
[24] W. Stornetta, T. Hogg, B. Huberman, A dynamical approach to temporal pattern processing, in: Proceedings of the 1988 Conference on Neural Information Processing Systems (NIPS), 1988, pp. 750–759.
[25] S. Anderson, J. Merrill, R. Port, Dynamic Speech Categorization with Recurrent Networks, Indiana University, Computer Science Department, 1988.
[26] R. Kumar, S. Vassilvitskii, Generalized distances between rankings, in: Proceedings of the Nineteenth International Conference on World Wide Web, ACM, 2010, pp. 571–580.
[27] I. Sutskever, J. Martens, G.E. Hinton, Generating text with recurrent neural networks, in: Proceedings of the Twenty-eighth International Conference on Machine Learning (ICML-11), 2011, pp. 1017–1024.
[28] J. Beirlant, E.J. Dudewicz, L. Györfi, E.C. Van der Meulen, Nonparametric entropy estimation: An overview, Int. J. Math. Stat. Sci. 6 (1) (1997) 17–39.
[29] H. Jaeger, Short Term Memory in Echo State Networks Technical Report, German National Research Center for Information Technology, 2001.



Claudio Gallicchio received the Laurea degree in Computer Science from the University of Bari, Bari, Italy, in 2007 and the Ph.D. degree in Computer Science from the University of Pisa, Pisa, Italy, in 2011. Currently, he is a post-doc research fellow at the Department of Computer Science, University of Pisa, within the Computational Intelligence & Machine Learning Group (CIML). His research interests include machine learning, deep learning, recurrent and recursive neural networks, reservoir computing, sequence and structured domains learning, with applications to cheminformatics, wireless and intelligent sensor networks, robotics, human activity recognition, document processing, smart grid. Since 2011 he is involved in European projects in the fields of computational intelligence, robotics, ambient assisted living and human activity recognition. He is a member of IEEE Computational Intelligence Society.

Alessio Micheli received the Laurea degree and the Ph.D. degree in Computer Science from the University of Pisa, Pisa, Italy, in 1998 and 2003, respectively. Currently, he is Associate Professor at the Department of Computer Science, University of Pisa, where he is the coordinator of the Computational Intelligence & Machine Learning Group (CIML). He is the national coordinator of the "Italian Working group on Machine Learning and Data Mining" of the Italian Association for the Artificial Intelligence. He is member of the IEEE CIS Task Force on Deep Learning and member of several program committees of conferences and workshops in Machine Learning and Artificial Intelligence. He has been Visiting Fellow at Wollongong University, Australia, and at the Neural Computing Research Group (NCRG), Aston University, UK. His research interests include machine learning, neural networks, deep learning, sequence and structured domains learning, relational learning, recurrent and recursive neural networks, reservoir computing models, kernel-based learning for nonvectorial data, applications to bioinformatics, cheminformat-

ics, and intelligent sensor networks. He has been involved in several national and european projects in the fields of computational intelligence, robotics and cheminformatics. He has co-authored more than 100 papers published in international journals and conference proceedings. He is a member of the IEEE Computational Intelligence Society and of the Cheminformatics and QSAR Society.



Luca Pedrelli received the Laurea Degree in Computer Science from the University of Pisa, Pisa, Italy, in 2015. Currently, he is a Ph.D. student at the Department of Computer Science, University of Pisa, within the Computational Intelligence & Machine Learning Group (CIML). His research interests include machine learning, neural networks, deep learning and reservoir computing models.