

Reservoir Topology in Deep Echo State Networks

Claudio Gallicchio and Alessio Micheli

Department of Computer Science, University of Pisa,
Largo B. Pontecorvo, 3, 56127 Pisa, Italy
gallicch@di.unipi.it, micheli@di.unipi.it

Abstract. Deep Echo State Networks (DeepESNs) recently extended the applicability of Reservoir Computing (RC) methods towards the field of deep learning. In this paper we study the impact of constrained reservoir topologies in the architectural design of deep reservoirs, through numerical experiments on several RC benchmarks. The major outcome of our investigation is to show the remarkable effect, in terms of predictive performance gain, achieved by the synergy between a deep reservoir construction and a structured organization of the recurrent units in each layer. Our results also indicate that a particularly advantageous architectural setting is obtained in correspondence of DeepESNs where reservoir units are structured according to a permutation recurrent matrix.

Keywords: Deep Echo State Networks, Deep Reservoir Computing, Reservoir Topology

1 Introduction

Reservoir Computing (RC) [20,27] delineates a class of Recurrent Neural Network (RNN) models based on the idea of separating the non-linear dynamical component of the network, i.e. the recurrent hidden reservoir layer, from the feed-forward linear readout layer. The reservoir is initialized randomly under stability constraints and then is left untrained, leaving the burden of training to fall only on the readout part of the architecture, hence resulting in a strikingly efficient approach to RNN design. In this context, the Echo State Network (ESN) model [17,15] is a popular realization of the RC paradigm based on implementing the reservoir in terms of a discrete-time non-linear dynamical system. Being featured by untrained dynamics, ESNs represent an important tool to understand and characterize the operation and potentialities of recurrent neural models. Shaping the reservoir architecture in order to achieve desired properties and optimized performance in applications, even in the absence of training of the recurrent connections, is one of the key goals of RC research [8].

In this paper we bring together two major trends in the area of ESN architectural studies. The first one focuses on the pattern of connectivity among the recurrent units. In this case, the aim is to constrain the random reservoir initialization process towards topologies that determine specific algebraic properties

of the resulting recurrent weight matrices. A relevant class of reservoir variants in this regard is given by ESNs with orthogonal recurrent matrices [29,14], which were shown to lead to improved performance with respect to random reservoirs both in terms of memorization skills and in terms of predictive performance on non-linear tasks. In particular, reservoirs whose structure is based on permutation matrices represent particularly appealing instances of orthogonal ESNs [14,25], entailing a simple and very sparse pattern of connectivity among the recurrent units. Other relevant architectural variants are given by reservoirs structured according to a ring topology or to form a chain of units [23,25]. The second major line of research that we consider regards the construction of hierarchically structured reservoir models. While initial studies in this context focused on composing multiple ESN modules to form ad-hoc architectures [18,26], recent works started analyzing the effects of stacking multiple untrained reservoir layers with the introduction of the DeepESN model [7]. On the one hand, the analysis of DeepESN dynamics contributes to uncover the intrinsic computational properties of deep neural networks in the temporal domain [7,12]. On the other hand, a proper architectural design of deep reservoirs might have a huge impact in real-world applications [11], enabling effective multiple time-scales processing and at the same time preserving the training efficiency typical of RC models.

In this paper we analyze the impact on the predictive performance given by a constrained reservoir topology in DeepESNs. Specifically, we consider deep architectures in which the individual reservoir layers are implemented based on permutation matrices, as well as on ring and on chain topologies. Our study is conducted in comparison to shallow ESN counterparts through numerical experiments on several benchmarks in the RC area.

The rest of this paper is structured as follows. The DeepESN model is introduced in Section 2, while the investigated reservoir topologies are described in Section 3. The experimental analysis is reported in Section 4. Finally, Section 5 draws conclusions and delineates future research directions.

2 Deep Echo State Networks

A DeepESN is an RC model in which the reservoir part is organized into a stacked composition of multiple untrained recurrent hidden layers. The external input is propagated only to the first reservoir layer, while each successive level in the deep architecture is fed by the output of the previous one, as graphically illustrated in Figure 1.

To fix our notation, we use L to indicate the number of layers in the deep reservoir, while N_U and N_Y respectively denote the sizes of input and output spaces. For the sake of simplicity in the presentation of the DeepESN model, here we make the assumption that all the reservoir layers are featured by the same number of units, indicated by N_R . The operation of each reservoir layer can be described in terms of a discrete-time non-linear dynamical system, whose state update equation is given in the form of an iterated mapping. In particular, at time-step t , the state of the first layer, i.e. $\mathbf{x}^{(1)}(t) \in \mathbb{R}^{N_R}$, is computed as

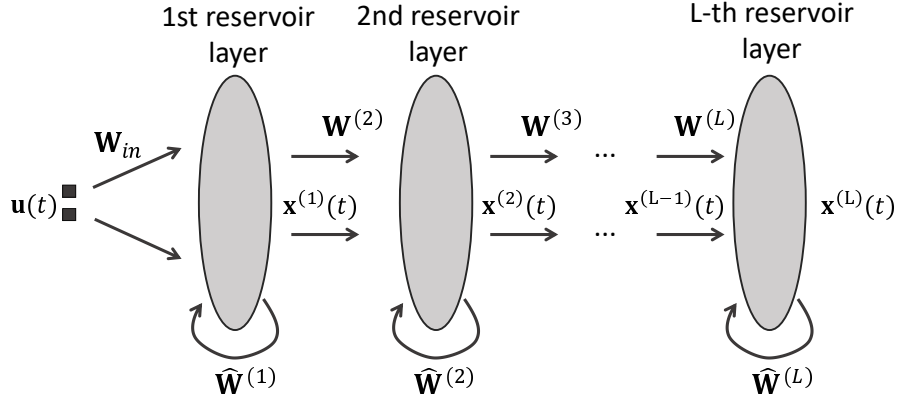


Fig. 1. Hierarchical reservoir architecture in a DeepESN.

follows:

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

while the state of each successive layer $l > 1$, i.e. $\mathbf{x}^{(l)}(t) \in \mathbb{R}^{N_R}$, is given by:

$$\mathbf{x}^{(l)}(t) = \tanh(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)}(t) + \hat{\mathbf{W}}^{(l)}\mathbf{x}^{(l)}(t-1)). \quad (2)$$

Here, \tanh indicates the element-wise application of the hyperbolic tangent non-linearity, $\mathbf{u}(t) \in \mathbb{R}^{N_U}$ represents the external input at time-step t , while \mathbf{W}_{in} , $\mathbf{W}^{(l)}$ and $\hat{\mathbf{W}}^{(l)}$ respectively denote the input weight matrix (that modulates the external input stimulation to the first layer), the inter-layer weight matrix for layer l (that modulates the strength of the connections from layer $l-1$ to layer l), and the recurrent reservoir weight matrix for layer l . In both the above equations 1 and 2 we omitted the bias terms for the ease of notation. The interested reader can find in [7] a more detailed description of the deep reservoir equations, framed in the more general context of leaky integrator reservoir units. In order to set up initial conditions for the state update equations 1 and 2, at time-step 0 all reservoir layers are set to a null state, i.e. $\mathbf{x}^{(l)}(0) = \mathbf{0}$ for all $l = 1, \dots, L$. Given this framework, it is worth noticing that a standard shallow ESN model can be seen as a special case of DeepESN in which a single reservoir layer is considered, i.e. $L = 1$.

As in standard RC approaches, the parameters of the entire reservoir component, i.e. the elements in all the weight matrices in equations 1 and 2, are left untrained after initialization subject to stability constraints. These are required to avoid the system dynamics to fall into unstable regimes, which would make them unsuitable for robust processing of time-series data. In the context of ESNs, the analysis of asymptotic stability is usually described in terms of the Echo State Property (ESP) [15,20], providing simple algebraic conditions for the initialization of reservoir weight matrices that have been recently extended to

cope with the case of deep reservoirs in [5]. Under a practical view-point, the analysis in [5] suggests to carefully control the spectral radius of all the reservoir weight matrices in the deep reservoir. In this paper, we use $\rho^{(l)}$ to denote the spectral radius in layer l , i.e. the largest among the absolute values of the eigenvalues of $\hat{\mathbf{W}}^{(l)}$. A simple initialization procedure for the reservoir of a DeepESN then consists in choosing the elements in $\hat{\mathbf{W}}^{(l)}$ randomly from a uniform distribution on $[-1, 1]$, subsequently re-scaling them to achieve desired values of $\rho^{(l)}$, typically not above unity. Similarly, the elements in \mathbf{W}_{in} and those in $\mathbf{W}^{(l)}$ (for $l > 1$) are initialized randomly from a uniform distribution on $[-1, 1]$, and then are re-scaled to control the input scaling hyper-parameter $\omega_{in} = \|\mathbf{W}_{in}\|_2$, and the set of inter-layer scaling hyper-parameters $\omega_{il}^{(l)} = \|\mathbf{W}^{(l)}\|_2$.

The output of the DeepESN is computed by a simple readout tool, which linearly combines the reservoir representations developed in all the layers of the deep architecture. In formulas, the output at time-step t , denoted as $\mathbf{y}(t) \in \mathbb{R}^{N_Y}$, is computed by the following equation:

$$\mathbf{y}(t) = \mathbf{W}_{out} [\mathbf{x}^{(1)}(t); \dots; \mathbf{x}^{(L)}(t)], \quad (3)$$

where \mathbf{W}_{out} is the output weight matrix, and $[\mathbf{x}^{(1)}(t); \dots; \mathbf{x}^{(L)}(t)]$ represents the global deep reservoir state at time-step t , expressed as the concatenation of all the states in the architecture. The elements in \mathbf{W}_{out} represent the only learnable weights of the DeepESN, and are typically adjusted to fit a training set by exploiting non-iterative training algorithms as in the case of standard RC models [20]. Notice that, although different patterns of reservoir-to-readout connectivity are possible [22], the one employed here, where all reservoir layers are used to feed the readout, has the advantage to allow the training algorithms to modulate and exploit differently the variety of representations provided by the different levels in the deep architecture.

A more comprehensive description of the characteristics and advantages of the DeepESN approach can be found in [6], while a constantly updated overview on the advancements achieved in this research field is given in [9]. To date, software implementations of the DeepESN model are made publicly available as libraries for Python¹, Matlab² and Octave³.

3 Reservoir Topology

We consider DeepESN architectural variants where the recurrent weight matrix in each layer l , i.e. $\hat{\mathbf{W}}^{(l)}$, is characterized by a specific structure, according to the topologies described in the following. The resulting patterns of reservoir connectivity are graphically exemplified in Figure 2.

¹ <https://github.com/luapedrelli/DeepESN>

² <https://it.mathworks.com/matlabcentral/fileexchange/69402-deepesn>

³ https://github.com/gallicch/DeepESN_octave

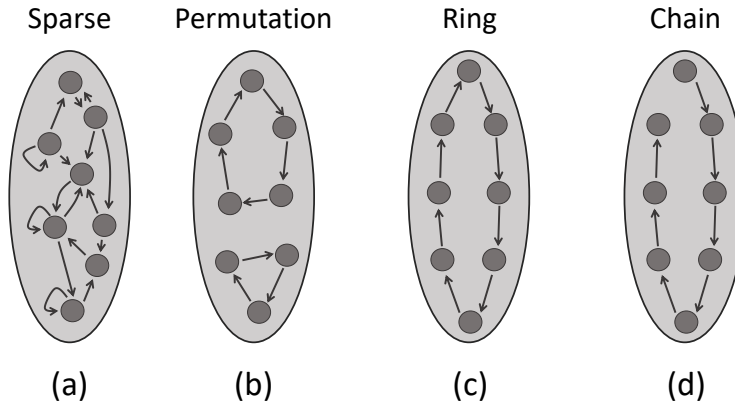


Fig. 2. Reservoir topologies of DeepESN layers.

Sparse Each reservoir unit is randomly connected to a subset of the others, determining a *sparse* recurrent matrix $\hat{\mathbf{W}}^{(l)}$ (see Figure 2(a)). This corresponds to a common setting used in RC practice and serves here as baseline for our analysis.

Permutation The structure of the recurrence matrix $\hat{\mathbf{W}}^{(l)}$ is given by a *permutation* matrix \mathbf{P} , i.e. we have:

$$\hat{\mathbf{W}}^{(l)} = \lambda \mathbf{P}, \quad (4)$$

where \mathbf{P} is obtained by randomly permuting the columns of the identity matrix, and λ is a multiplicative constant that specifies the value of the non-zero recurrent weights. In this case, the spectral radius of $\hat{\mathbf{W}}^{(l)}$ is determined by the value of λ , i.e. $\rho^{(l)} = \lambda$. The permutation topology implies that each row and each column of the recurrence matrix have exactly one non-zero element, resulting into a reservoir architecture that presents a variable number of disjoint cyclic structures, as graphically exemplified in Figure 2(b). The levels in the deep reservoir architecture are allowed to employ different permutations, i.e. the number of cycles in each reservoir layer can be different.

In the context of shallow ESNs, this kind of topology has been empirically studied in [2], where it was shown to achieve good memorization skills at the same time improving the performance of randomly initialized reservoirs in tasks involving non-linear mappings. Interestingly, the permutation topology has been investigated in [14] as a way to implement orthogonal reservoir matrix structures, under the name of Critical ESNs.

Ring The reservoir units are organized to form a single *ring*, as shown in Figure 2(c). Accordingly, the recurrent weight matrix $\hat{\mathbf{W}}^{(l)}$ is expressed as:

$$\hat{\mathbf{W}}^{(l)} = \lambda \begin{bmatrix} 0 & 0 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 0 \end{bmatrix}, \quad (5)$$

where λ is the value of non-zero recurrent weights, and determines the spectral radius of $\hat{\mathbf{W}}^{(l)}$, i.e. $\rho^{(l)} = \lambda$. The ring topology can be easily seen as a special case of the permutation topology, where the pattern of reservoir connectivity is ruled by the specific permutation matrix in equation 5, and the reservoir units are all part of the same cyclic structure.

Reservoirs following this architectural organization have been subject of several studies in literature on shallow RC. Notable instances in this regard are given by the work in [25], in which the ring topology is studied in the context of orthogonal reservoir structures, and by the work in [23], where the study is carried out under the perspective of architectural design simplification for minimum complexity ESN construction. One interesting outcome of previous analysis on the ring topology is that, compared to randomly initialized reservoirs, it shows superior memory capacity that, at least in the linear case, approaches the optimal value [23]. While this optimal memory characterization has been extensively analyzed in literature for the more general class of orthogonal recurrent weight matrices (see e.g. [29,16,3]), the ring topology presents the advantage of a strikingly simple (and sparse) dynamical network construction.

Chain The recurrent units are arranged in a pipeline, where each unit - except for the first one - receives in input the activation of the previous one, forming a *chain* as in the example in Figure 2(d). The only non-zero elements in $\hat{\mathbf{W}}^{(l)}$ are located in the lower sub-diagonal, i.e. we have:

$$\hat{\mathbf{W}}^{(l)} = \lambda \begin{bmatrix} 0 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 0 \end{bmatrix}, \quad (6)$$

where as in previous cases λ identifies the value of non-zero weights. Although in this case $\hat{\mathbf{W}}^{(l)}$ is nilpotent and hence its spectral radius is always 0, we still operate on λ to control the magnitude of recurrent weights. As such, with a slight abuse of notation, also in this case we set $\rho^{(l)} = \lambda$. Overall, the chain topology results in a particularly simple design strategy that, from the architectural perspective, applies a further simplification to the ring topology by removing one of the connections between the internal units.

Literature works on shallow ESN models pointed out the merits of reservoir organizations based on a chain topology (also called delay-line reservoirs), as a very simple approach to the architectural design of the network, resulting

in a model that is easier to analyze [29] and that leads to comparable or even better performance than standard ESNs [23,25].

4 Experiments

In this section we illustrate the experimental analysis conducted in this paper. Specifically, in Section 4.1 we detail the datasets considered and the experimental settings adopted in our work, whereas in Section 4.2 we report and discuss the achieved numerical results.

4.1 Datasets and Experimental Settings

In our experiments, we considered benchmark datasets featured by univariate time-series (i.e., $N_U = N_Y = 1$).

The first dataset is obtained from a non-linear auto-regressive moving average system of the 10-th order (NARMA10). At each time-step, the input $u(t)$ comes from a uniform distribution over $[0, 0.5]$, whereas the corresponding target output $y_{tg}(t)$ is given by the following relation:

$$y_{tg}(t) = 0.3 y_{tg}(t-1) + 0.05 y_{tg}(t-1) \sum_{i=1}^{10} y_{tg}(t-i) + 1.5 u(t-10) u(t-1) + 0.1. \quad (7)$$

The second dataset that we considered is the Santa Fe Laser time-series [28], where the input values $u(t)$ are sampled intensities from a far-infrared laser in chaotic regime, re-scaled by a factor of 0.01. We used the Laser dataset to define a next-step prediction task, where $y_{tg}(t) = u(t+1)$ for each time-step t .

The last two datasets are instances of the Mackey-Glass (MG) [21,4] time-series, obtained by discretizing the following non-linear differential equation:

$$\frac{\delta u(t)}{\delta t} = \frac{0.2 u(t-\tau)}{1 + u(t-\tau)^{10}} - 0.1 u(t), \quad (8)$$

where τ is a parameter of the system influencing its dynamical behavior. We generated two MG time-series using $\tau = 17$ (MG17) and $\tau = 30$ (MG30), representing cases with increasingly complex chaotic behavior. In both cases, the elements of the time-series were shifted by -1 and then passed through the tanh squashing function as in [17,15]. The two MG time-series allowed us to set up two next-step prediction tasks, where $y_{tg}(t) = u(t+1)$ for each time-step t .

For NARMA10, MG17 and MG30 we generated datasets with 10000 time-steps, while the Laser dataset contained a number of 10092 samples. In all the cases, the available data was split into a training set, comprising the first 5000 time-steps, and a test set, comprising the remaining samples. The first 100 time-steps were used as transient to wash out the initial conditions. The performance of the considered RC models was evaluated in terms of mean squared error (MSE) in all the tasks.

In our experiments, we considered DeepESNs with a total number of 500 recurrent reservoir units, distributed evenly across the layers of the deep architecture, varying the number of layers L from 2 to 5⁴. All the reservoir layers in the deep architecture shared the same values for the scaling hyper-parameters ρ and ω_{il} , i.e. $\rho = \rho^{(1)} = \dots = \rho^{(L)}$ and $\omega_{il} = \omega_{il}^{(2)} = \dots = \omega_{il}^{(L)}$. To account for sparsity, each reservoir unit was randomly connected to 5 units in the previous layer and to 5 units in the same layer. Of course, when considering permutation, ring and chain reservoir topologies, the connectivity of the reservoir units in each layer followed the corresponding specific structure described in Section 3. In all the cases, we used fully-connected input weight matrices. For every task and choice of the reservoir topology, the DeepESN hyper-parametrization was chosen by model selection on a validation set comprising the last 1000 time-steps of the training split. To this end, we performed a random search with 50 networks configurations for each number of layers, sampling the value of ρ from a uniform distribution in $(0.1, 1]$, and the values of ω_{in} and ω_{il} from uniform distributions in $(0.1, 2]$. The achieved results were averaged on 10 network guesses for each hyper-parametrization explored, and readout training was performed by using pseudo-inversion. Finally, our experimental analysis was conducted in comparison with shallow ESN setups, considering the same reservoir topologies investigated in the DeepESN case, and using the same experimental setting described above, with the only crucial exception that all the available reservoir units were organized into a single layer (i.e. $L = 1$).

4.2 Results

The test MSE values obtained by DeepESNs in correspondence of all the considered types of layer-wise reservoir topology are reported in Table 1. For the sake of comparison, the same table shows the results achieved by shallow ESNs under the same architectural conditions examined in the deep case. In all the cases, the sparse reservoir topology is considered as a baseline setup for our analysis.

The performance values reported in Table 1 allow us to draw several lines of observations. First of all, our results confirm the goodness of the considered reservoir architectural variants already in the shallow setup, showing improved performance (i.e., a smaller MSE) with respect to the sparse baseline in all the cases (with the sole exception of permutation shallow reservoirs on Laser). Second, we observe that the performance of DeepESN with constrained topology (i.e. permutation, ring and chain) enhances that one of sparse DeepESN in all the considered tasks (with the only exception of deep reservoirs with chain architecture on Laser). Moreover, we can see that DeepESN improves the results of shallow ESN in all the tasks and for all the choices of reservoir topology, both in the constrained architectural cases and for the base sparse reservoir setup. Taken together, results in Table 1 clearly indicate the performance advantage arising from the synergy between deep organization and constrained topology as

⁴ With the only exception of the case $L = 3$, where the first two layers contained 167 units and the last one contained 166 units.

NARMA10			
Topology	ESN	DeepESN	Layers
Sparse	1.658 10^{-4} (3.367 10^{-5})	1.647 10^{-4} (3.415 10^{-5})	2
Permutation	1.354 10^{-4} (1.589 10^{-5})	<u>1.243 10^{-4}</u> (1.464 10^{-5})	2
Ring	1.494 10^{-4} (1.547 10^{-5})	1.482 10^{-4} (1.713 10^{-5})	2
Chain	1.571 10^{-4} (1.780 10^{-5})	1.569 10^{-4} (2.594 10^{-5})	2

Laser			
Topology	ESN	DeepESN	Layers
Sparse	1.226 10^{-3} (1.037 10^{-4})	8.228 10^{-4} (2.309 10^{-4})	5
Permutation	1.312 10^{-3} (1.385 10^{-4})	<u>6.633 10^{-4}</u> (8.861 10^{-5})	5
Ring	1.161 10^{-3} (7.541 10^{-5})	7.640 10^{-4} (4.331 10^{-5})	4
Chain	9.496 10^{-4} (1.183 10^{-4})	8.555 10^{-4} (8.302 10^{-5})	4

MG17			
Topology	ESN	DeepESN	Layers
Sparse	3.739 10^{-9} (1.387 10^{-9})	2.328 10^{-9} (8.299 10^{-10})	2
Permutation	3.093 10^{-9} (3.241 10^{-10})	<u>4.576 10^{-10}</u> (6.280 10^{-10})	5
Ring	1.585 10^{-9} (2.989 10^{-10})	5.043 10^{-10} (3.891 10^{-10})	5
Chain	1.950 10^{-9} (3.745 10^{-10})	4.913 10^{-10} (2.535 10^{-10})	3

MG30			
Topology	ESN	DeepESN	Layers
Sparse	1.476 10^{-8} (1.781 10^{-9})	1.172 10^{-8} (1.406 10^{-9})	2
Permutation	1.027 10^{-8} (5.412 10^{-10})	<u>8.618 10^{-9}</u> (1.457 10^{-9})	3
Ring	1.197 10^{-8} (1.549 10^{-9})	1.078 10^{-8} (2.066 10^{-9})	5
Chain	1.086 10^{-8} (9.519 10^{-10})	9.096 10^{-9} (1.803 10^{-9})	3

Table 1. Test MSE (and std) achieved by shallow ESN and DeepESN settings for different choices of the reservoir topology. The last column reports the number of layers selected for DeepESN. Best results for each task are underlined.

factors of architectural design of reservoirs. Giving a structure to the architecture of reservoirs both at a coarser level, i.e. organizing the recurrent units into layers, and at a finer level, i.e. organizing individual layers’ units into cyclic or chain structures, amplifies the benefits brought by the two factors individually. Finally, we notice that the best performing architecture in our experiments is the DeepESN with permutation reservoir topology, which obtained the smallest errors on all the tasks, and is put forward here as a particularly effective (yet sparse and efficient) approach to the architectural design of reservoir models.

5 Conclusions

In this paper we have investigated the role of reservoir topology in the architectural design of DeepESNs. Specifically, we focused on analyzing the effects

of constraining the recurrent weight matrix of each layer according to permutation, ring and chain topologies. Numerical results on several RC benchmarks pointed out a striking beneficial effect arising from the combination of a deep reservoir construction with a structured organization of the recurrent units in each layer. Our results indicate that DeepESN with reservoir units arranged to obey a permutation scheme (i.e., forming multiple rings) provides a particularly advantageous design strategy for reservoirs, leading to the best performance in all the explored tasks.

While already giving interesting empirical evidences on the potentialities of deep RC architectures, the study presented in this paper opens the way to several directions for further research. First of all, the experimental analysis described here suggests that the use of simplified deep RC models has a great potential that can be exploited massively in real-world applications. Leveraging the parsimonious design approach resulting from structured sparsity of reservoir units, the class of deep neural models studied in this work seems an ideal candidate e.g. for embedding advanced learning capabilities on resource-constrained computing devices. On the methodological side, a natural extension of the work in this paper is to analyze the effect of a broader pool of reservoir architectural variants, including e.g. small-world [19], cycles with regular jumps [24] and concentric [1] reservoirs. Moreover, future research could pursue even further the simplification of architectural construction in deep RC models, reducing the impact of randomness in the network initialization in the same vein as the works on minimum complexity ESNs [23,24]. Simplifying the reservoir structure locally to each layer can also be exploited from a more theoretically-oriented perspective, easing the mathematical analysis of dynamical properties naturally emerging in deep RNNs. In this concern, it is certainly interesting to extend fundamental mathematical results, e.g. pertaining to short-term memory capacity [23,29,16], or to approximation properties [13] of shallow reservoirs to the case of DeepESN. In addition to this, we believe that the role of orthogonality in deep reservoirs, studied in this paper in relation to the individual layers of the architecture, is an intriguing concept that deserves to be investigated also at the level of global (instead of local) DeepESN dynamics. Finally, the advantages of constrained DeepESN architectures delineated in this paper can be extended to larger classes of models, including e.g. deep RC for complex data structures [10], as well as fully trained deep RNNs.

References

1. Bacciu, D., Bongiorno, A.: Concentric esn: Assessing the effect of modularity in cycle reservoirs. In: 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2018)
2. Boedecker, J., Obst, O., Mayer, N.M., Asada, M.: Studies on reservoir initialization and dynamics shaping in echo state networks. In: Proc. of the 17th European Symposium on Artificial Neural Networks (ESANN). pp. 227–232. d-side publi. (2009)

3. Farkaš, I., Bosák, R., Gergel', P.: Computational analysis of memory capacity in echo state networks. *Neural Networks* **83**, 109–120 (2016)
4. Farmer, J.D.: Chaotic attractors of an infinite-dimensional dynamical system. *Physica D: Nonlinear Phenomena* **4**(3), 366–393 (1982)
5. Gallicchio, C., Micheli, A.: Echo state property of deep reservoir computing networks. *Cognitive Computation* **9**(3), 337–350 (2017)
6. Gallicchio, C., Micheli, A.: Why Layering in RNN? A DeepESN Survey. In: *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)*. pp. 1–8. IEEE (2018)
7. Gallicchio, C., Micheli, A., Pedrelli, L.: Deep reservoir computing: A critical experimental analysis. *Neurocomputing* **268**, 87–99 (2017). <https://doi.org/https://doi.org/10.1016/j.neucom.2016.12.089>
8. Gallicchio, C., Micheli, A., Tiño, P.: Randomized recurrent neural networks. In: *26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2018)*. pp. 415–424. i6doc. com publication (2018)
9. Gallicchio, C., Micheli, A.: Deep echo state network (deepesn): A brief survey. arXiv preprint arXiv:1712.04323 (2017)
10. Gallicchio, C., Micheli, A.: Deep reservoir neural networks for trees. *Information Sciences* **480**, 174–193 (2019)
11. Gallicchio, C., Micheli, A., Pedrelli, L.: Design of deep echo state networks. *Neural Networks* **108**, 33–47 (2018)
12. Gallicchio, C., Micheli, A., Silvestri, L.: Local lyapunov exponents of deep echo state networks. *Neurocomputing* **298**, 34–45 (2018)
13. Grigoryeva, L., Ortega, J.P.: Echo state networks are universal. *Neural Networks* **108**, 495–508 (2018)
14. Hajnal, M.A., Lőrincz, A.: Critical echo state networks. In: *Proc. of the International Conference on Artificial Neural Networks (ICANN)*. pp. 658–667. Springer (2006)
15. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks - with an erratum note. Tech. rep., GMD - German National Research Institute for Computer Science (2001)
16. Jaeger, H.: Short term memory in echo state networks. Tech. rep., German National Research Center for Information Technology (2001)
17. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* **304**(5667), 78–80 (2004)
18. Jaeger, H.: Discovering multiscale dynamical features with hierarchical echo state networks. Tech. rep., Jacobs University Bremen (2007)
19. Kawai, Y., Park, J., Asada, M.: A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks* (2019)
20. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. *Computer Science Review* **3**(3), 127–149 (2009)
21. Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. *Science* **197**(4300), 287–289 (1977)
22. Pascanu, R., Gulcehre, C., Cho, K., Bengio, Y.: How to construct deep recurrent neural networks. arXiv preprint arXiv:1312.6026v5 (2014)
23. Rodan, A., Tino, P.: Minimum complexity echo state network. *IEEE transactions on neural networks* **22**(1), 131–144 (2011)
24. Rodan, A., Tiño, P.: Simple deterministically constructed cycle reservoirs with regular jumps. *Neural computation* **24**(7), 1822–1852 (2012)

25. Strauss, T., Wustlich, W., Labahn, R.: Design strategies for weight matrices of echo state networks. *Neural computation* **24**(12), 3246–3276 (2012)
26. Triefenbach, F., Jalalvand, A., Schrauwen, B., Martens, J.P.: Phoneme recognition with large hierarchical reservoirs. In: *Advances in neural information processing systems*. pp. 2307–2315 (2010)
27. Verstraeten, D., Schrauwen, B., dHaene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. *Neural networks* **20**(3), 391–403 (2007)
28. Weigend, A.S.: *Time series prediction: forecasting the future and understanding the past*. Routledge (2018)
29. White, O.L., Lee, D.D., Sompolinsky, H.: Short-term memory in orthogonal neural networks. *Physical review letters* **92**(14), 148102 (2004)