ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Simplifying Deep Reservoir Architectures

Claudio Gallicchio, Alessio Micheli, Antonio Sisbarra

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

Abstract. We study the impact of architectural simplifications to the design of deep Reservoir Computing (RC) models. To do so, we analyze the effects of shaping the structure of reservoir matrices, reducing the complexity of the deep recurrent network to a minimal setup. Experimental results point out the benefits of a particularly simple deep RC architecture with ring topology in each reservoir layer and deterministically constructed input and inter-reservoir connections.

1 Introduction

Reservoir Computing (RC) [1, 2] denotes a popular class of Recurrent Neural Network (RNN) models where the state-transition system - the *reservoir* - is randomly initialized under stability conditions and only the connections to a simple output layer - the readout - are adjusted based on a training set. The resulting efficiency of the training algorithms represents the major characteristic of the approach. A relevant line of research in RC focuses on studying the topology of the reservoir, i.e. how the recurrent neurons are connected among each other [3]. The idea is to operate on algebraic properties of the untrained weight matrices to ensure that the internal representations of the driving input signal are richer than what can be obtained by a standard random initialization [4, 5]. Besides this, constraining the reservoir topology has also the advantage of simplifying the network design, reducing the complexity of the resulting system and the impact of randomness in its initialization [6].

Recently, the class of RC models has been extended to include state transition systems computed by a stacked composition of multiple non-linear reservoir layers. Under the framework of discrete-time dynamical systems, we refer to this new class of RC models as Deep Echo State Network (DeepESN) [7]. The appeal of studying DeepESNs is twofold. Theoretically, they allow to study the intrinsic advantages of layering in RNN architectures in the absence of (or prior to) training of recurrent connections. From the applicative view-point, DeepESNs provide a great tool for capitalizing on the efficiency of RC training and on the advantages of deep RNNs at the same time. In particular, in [8] we showed how properly constructed deep reservoir systems are able to efficiently achieve state-of-the-art results in complex tasks related to speech and music processing.

In this paper, we go deeper into the analysis of the architectural design of DeepESN. We do so by studying progressive simplifications to the construction of deep reservoirs. In particular, taking inspiration from [6], we consider reservoirs constrained to a ring topology and non-randomized input and inter-layer connections. The general aim is to bring together the advantages coming from

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

the enriched dynamics of (globally) deep and (locally) ring-shaped reservoirs, at the same time exploiting the simplified architectural setup, which would enable a particularly effective search in the hyper-parameters space. We assess the impact of these network simplifications in terms of predictive performance on benchmarks tasks on time-series. Our experimental analysis is conducted also in comparison to both standard and simplified shallow ESNs.

2 Deep Echo State Networks

Deep Echo State Networks (DeepESNs) [7] are discrete-time input-driven recurrent neural models where the dynamical component is a stacked composition of multiple non-linear untrained reservoir layers, and the output is computed by a linear output readout layer. The crucial difference with respect to conventional RC models is that the reservoir is *deep*, i.e., the recurrent neurons are arranged in a layered, hierarchical architecture. In this way, the external input time-series drives the dynamics of the first layer, and the dynamics of each successive reservoir layer is driven by the state of the previous layer in the stack.

We denote the number of reservoir layers by L, and we consider DeepESNs where all reservoirs have the same number of neurons N. Considering leaky-integrator neurons (and dropping the bias terms for ease of notation), the deep reservoir state transition system is described by the following set of equations:

$$\begin{aligned}
&\mathbf{x}^{(1)}(t) = (1-\alpha)\,\mathbf{x}^{(1)}(t-1) + \alpha \tanh\left(\mathbf{U}\,\mathbf{u}(t) + \mathbf{W}^{(1)}\mathbf{x}^{(1)}(t-1)\right)\\
&\mathbf{x}^{(i)}(t) = (1-\alpha)\,\mathbf{x}^{(i)}(t-1) + \alpha \tanh\left(\mathbf{V}^{(i)}\mathbf{x}^{(i-1)}(t) + \mathbf{W}^{(i)}\mathbf{x}^{(i)}(t-1)\right) \quad (\text{for } i>1),
\end{aligned} \tag{1}$$

where $\mathbf{U}$ is the input weight matrix, $\mathbf{V}^{(i)}$ (for $i > 1$) and $\mathbf{W}^{(i)}$ (for $i \geq 1$) are respectively the inter-layer weight matrix and the recurrent reservoir weight matrix at layer i. Moreover, $\mathbf{u}(t)$ is the input at time-step t, and $\mathbf{x}^{(i)}(t)$ is the reservoir state computed at the i-th layer. The system in each layer is initialized with null state, i.e. $\mathbf{x}^{(i)}(0) = \mathbf{0}$ for all i. Finally, $\alpha \leq 1$ controls the leakage. Interestingly, compared to the case of shallow reservoirs, layering of the recurrent neurons in DeepESNs results in a sparser (simpler) architecture where some of the connections have been dropped (adding constraints as analyzed in [7]).

Initialization of DeepESNs is based on asymptotic stability of the set of nested dynamical systems in (1) (see [9]). This translates into a random initialization of the weight matrices, followed by a re-scaling of spectral properties. In practice, for initialization of $\mathbf{U}$ it is used a uniform distribution on $[-\omega_{in}, \omega_{in}]$, and for $\mathbf{V}^{(i)}$ a uniform distribution on $[-\omega_{il}, \omega_{il}]$. Recurrent weight matrices $\mathbf{W}^{(i)}$ are initialized from a uniform distribution over $[-1, 1]$ and then are re-scaled to have a desired effective spectral radius[1] value ρ. This procedure identifies the major hyper-parameters of the model, namely: input scaling ω_{in}, inter-layer scaling ω_{il} and spectral radius ρ. A further hyper-parameter is given by the leaking-rate α.

The output is computed by applying a dense readout linear layer to the concatenation of the reservoir activation in each layer. The output at time-step

[1]The largest among the eigenvalues in modulus.

t, i.e. $\mathbf{y}(t)$, is computed as $\mathbf{Z}\,[\mathbf{x}^{(1)}(t), \mathbf{x}^{(2)}(t), \ldots, \mathbf{x}^{(L)}(t)]$. The readout weights in $\mathbf{Z}$ are the only trained parameters, typically in closed form by pseudo-inversion.

3 Architectural Simplifications

In this paper we study progressive simplifications to the structure of the weight matrices involved in deep reservoir initialization. The aim is twofold: first of all simplifying the architectural setup, and, secondly, reducing (up to eliminating) the role of randomization in the deep network initialization. To this end, we consider the following DeepESN architectures.

Sparse – For each layer, the reservoir neurons are sparsely connected among each other, i.e. matrices $\mathbf{W}^{(i)}$ are sparse. Such sparse connectivity is randomized, and each reservoir neuron has incoming (recurrent) connections from a number of C other neurons at the same layer (we used $C = 10$). Input (i.e., $\mathbf{U}$) and inter-layer matrices (i.e., $\mathbf{V}^{(i)}$) are randomized as described in Section 2. This setup is used as baseline DeepESN reference in our experiments. Notice that to fully describe any instance of such neural architecture (i.e., the process of reservoir initialization) implies a number of "degrees of freedom" that scales quadratically with N, i.e. the number of neurons per reservoir layer.

Ring – At each layer, reservoir neurons are arranged in a ring, where each neuron receives input from the previous unit in the cycle, and projects its activation to the successive one. Accordingly, the topology of $\mathbf{W}^{(i)}$ matrices is constrained to a deterministic (structured) sparsity, where the only non-zero entries are those located in the sub-diagonal and in the top-right corner. All elements of $\mathbf{W}^{(i)}$ are set to the same value, which corresponds to the spectral radius ρ. Input and inter-layer weight matrices are randomized.

Ring + Simple Input – The deep reservoir architecture is shaped as in the Ring DeepESN, with the difference that the structure of input weights is simplified as follows. All the weights in $\mathbf{U}$ are set to the same value ω_{in}, while the sign of each entry is obtained in a deterministic fashion, by following the decimal expansion of the irrational number π, as in [6]. In particular, for decimal values < 5 the sign is set to $-$, otherwise it is set to $+$.

Ring + Simple Input & Inter – The deep reservoir is structured as in the previous case, and - in addition - the structure of all the inter-layer weight matrices is simplified. Specifically, all the elements of $\mathbf{V}^{(i)}$ have the same absolute value ω_{il}, and the sign of each entry is determined by following the decimal expansion of π as described in the previous case. This simplified deep reservoir architecture is shown in Fig. 1. Relevantly, in this setting the architecture of each DeepESN is fully described by just 3 numbers (the hyper-parameters ω_{in}, ω_{il} and ρ), which clearly highlights the design simplification with respect to the basic Sparse setting above. Moreover, it is worth noticing that in this setup all aspects of randomization in the network initialization have been eliminated.

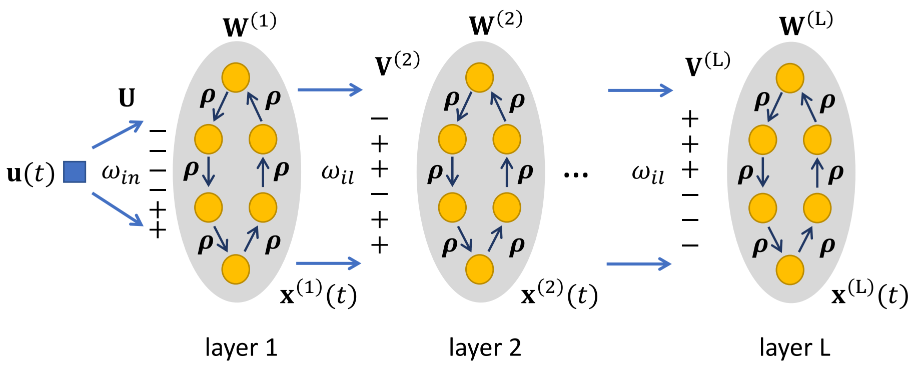

Fig. 1: Simplified deep reservoir architecture.

4 Experiments

In what follows, we use $d(t)$ to denote the target output at time-step t.
Datasets. We consider 4 tasks on time-series data. The first is LASER, where the data consists of sampled intensities from far-infrared laser in chaotic regime. The second and the third are achieved by discretizing the popular non-linear Mackey-Glass (MG) differential equation $\delta u(t)/\delta t = \big((0.2\,u(t-\tau))/(1+u(t-\tau)^{10}\big) - 0.1u(t)$. We consider two MG cases resulting in chaotic dynamics, i.e., MG17 corresponding to the choice of $\tau = 17$, and MG30 corresponding to the choice $\tau = 30$. For LASER, MG17 and MG30 the learning task consists in predicting the next time-step of the the chaotic time-series, i.e. $d(t) = u(t+1)$. The last task is based on a nonlinear auto-regressive moving average of order 10 (NARMA10), where at each time-step t the input comes from a uniform distribution over $[0, 0.5]$, and the target output is $d(t) = 0.3d(t)(t-1) + 0.05d(t)(t-1)\sum_{i=1}^{10} d(t)(t-i) + 1.5u(t-10)u(t-1) + 0.1$. The total number of time-steps is 10092 for LASER, and 10000 for MG17, MG30 and NARMA10. For all the cases, the first 5000 time-steps were used for training, and the remaining for test.
Settings. We considered DeepESNs with a number of layers L ranging in 2-5. All layers contained the same number of reservoir units, with the total number of recurrent neurons being fixed to 100. The same ρ and α were used in all layers, and the same inter-layer scaling ω_{il} was used for each layer $i > 1$. We explored values of $\rho \in [0.1, 1.5]$, $\alpha \in [0.1, 1]$, $\omega_{in} \in [10^{-5}, 1.5]$ and $\omega_{il} \in [10^{-5}, 1.5]$. Hyper-parameter values were selected by hold-out model selection on a validation set containing the last 1000 time-steps of the training data. To this end, we used random search and generated 30 random configurations. For each configuration, the performance was averaged (and corresponding std was computed) over a number of 10 repetitions (with the same hyper-parameters, but different random seeds for initialization). Averaging was not required in the Ring + Simple Input & Inter case, as the network is built in a deterministic fashion. In this

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

case, the hyper-parameter search considered 300 random configurations. The readout we trained by pseudo-inversion, with a washout of 100 steps.

Our analysis is conducted comparatively to shallow reservoir networks, in which case we limit ourselves to considering the Sparse, Ring and Ring + Simple Input settings. To this end, we ran experiments with ESNs under the same experimental settings delineated above for DeepESNs, and considering the same range for the total number of reservoir neurons (which in this case were arranged in a single layer). The hyper-parameters of all the deep and shallow RC variants were chosen individually by model selection.

Results. We assess the networks performance by computing the MSE (the lower the better). The achieved test set results are provided in Tab. 1.

LASER (MSE)		
Architecture	ESN	DeepESN
Sparse	$1.59\,e-3$ $_{(\pm 6.2e-4)}$	$3.09\,e-3$ $_{(\pm 1.4e-4)}$
Ring	$1.70\,e-3$ $_{(\pm 4.6e-4)}$	$3.87\,e-3$ $_{(\pm 2.4e-3)}$
Ring + Simple Input	$1.56\,e-3$	$2.10\,e-3$ $_{(\pm 4.0e-4)}$
Ring + Simple Input & Inter	-	$\mathbf{1.15\,e-3}$
MG17 (MSE)		
Architecture	ESN	DeepESN
Sparse	$1.42\,e-9$ $_{(\pm 8.2e-11)}$	$2.12\,e-9$ $_{(\pm 3.5e-10)}$
Ring	$1.53\,e-9$ $_{(\pm 1.9e-10)}$	$2.89\,e-9$ $_{(\pm 5.3e-10)}$
Ring + Simple Input	$1.57\,e-9$	$2.68\,e-9$ $_{(\pm 3.2e-10)}$
Ring + Simple Input & Inter	-	$\mathbf{1.27\,e-9}$
MG30 (MSE)		
Architecture	ESN	DeepESN
Sparse	$1.57\,e-8$ $_{(\pm 1.3e-9)}$	$6.86\,e-9$ $_{(\pm 4.2e-10)}$
Ring	$4.96\,e-9$ $_{(\pm 2.5e-10)}$	$5.26\,e-9$ $_{(\pm 4.4e-10)}$
Ring + Simple Input	$3.73\,e-9$	$4.69\,e-9$ $_{(\pm 3.1e-10)}$
Ring + Simple Input & Inter	-	$\mathbf{3.15\,e-9}$
NARMA10 (MSE)		
Architecture	ESN	DeepESN
Sparse	$1.71\,e-3$ $_{(\pm 3.7e-4)}$	$1.48\,e-3$ $_{(\pm 3.3e-4)}$
Ring	$1.32\,e-3$ $_{(\pm 5.3e-5)}$	$1.23\,e-3$ $_{(\pm 6.2e-5)}$
Ring + Simple Input	$1.02\,e-3$	$1.22\,e-3$ $_{(\pm 1.2e-4)}$
Ring + Simple Input & Inter	-	$\mathbf{9.75\,e-4}$

Table 1: Achieved test results. Best performance is highlighted in bold font.

We can note that simplified DeepESN architectures consistently outperform Sparse DeepESNs on all tasks. In most cases, the simpler is the deep reservoir architecture (and the more randomization aspects are eliminated) the better is the observed performance. Overall, the best result is obtained on all tasks by the simplest DeepESN, i.e., with Ring + Simple Input & Inter setup. Results in Tab. 1 confirm the benefit of simple deterministically constructed reservoirs also in shallow ESN, in line with [6]. In this case, however, the possible final performance enhancement is less pronounced than the one seen for deep reservoirs.

5 Conclusions

We have investigated the effects of architectural simplifications in the design of deep RC models. Remarkably, a very simple DeepESN with ring topology in each reservoir layer and deterministically constructed input and inter-layer connections showed the highest performance in comparison to more complex (both shallow and deep) RC setups. Overall, our analysis put forward a minimalistic deep RNN architecture, with few degrees of freedom and no randomization in its construction, as a very effective tool for learning tasks in the time-series domain.

Looking ahead, we believe that the advantages of minimal deep RC architectures can be exploited massively in real-world applications, e.g., in the context of Machine Learning embedded on edge devices. Moreover, the analysis conducted in this paper can be capitalized to extend the advantages of minimal deep RC architectures to the case of learning in graph domains [10].

References

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

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

[3] T. Strauss, W. Wustlich, and R. Labahn. Design strategies for weight matrices of echo state networks. *Neural computation*, 24(12):3246–3276, 2012.

[4] P. Tiňo. Dynamical systems as temporal feature spaces. *arXiv preprint arXiv:1907.06382*, 2019.

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

[6] A. Rodan and P. Tiňo. Minimum complexity echo state network. *IEEE transactions on neural networks*, 22(1):131–144, 2010.

[7] C. Gallicchio, A. Micheli, and L. Pedrelli. Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, 268:87–99, 2017.

[8] C. Gallicchio, A. Micheli, and L. Pedrelli. Design of deep echo state networks. *Neural Networks*, 108:33–47, 2018.

[9] C. Gallicchio and A. Micheli. Echo state property of deep reservoir computing networks. *Cognitive Computation*, 9(3):337–350, 2017.

[10] C. Gallicchio and A. Micheli. Fast and deep graph neural networks. In *Proceedings of AAAI*, 2020. arXiv preprint arXiv:1911.08941.