ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

Comparison between DeepESNs and gated RNNs on multivariate time-series prediction

Claudio Gallicchio and Alessio Micheli and Luca Pedrelli

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

Abstract. We propose an experimental comparison between Deep Echo State Networks (DeepESNs) and gated Recurrent Neural Networks (RNNs) on multivariate time-series prediction tasks. In particular, we compare reservoir and fully-trained RNNs able to represent signals featured by multiple time-scales dynamics. The analysis is performed in terms of efficiency and prediction accuracy on 4 polyphonic music tasks. Our results show that DeepESN is able to outperform ESN in terms of prediction accuracy and efficiency. Whereas, between fully-trained approaches, Gated Recurrent Units (GRU) outperforms Long Short-Term Memory (LSTM) and simple RNN models in most cases. Overall, DeepESN turned out to be extremely more efficient than others RNN approaches and the best solution in terms of prediction accuracy on 3 out of 4 tasks.

1 Introduction

Recurrent Neural Networks (RNNs) are a class of neural networks suitable for time-series processing. In particular, gated RNNs [1, 2], such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), are fully-trained recurrent models that implement adaptive gates able to address signals characterized by multiple time-scales dynamics. Recently, within the Reservoir Computing (RC) [3] framework, the Deep Echo State Network (DeepESN) model has been proposed as extremely efficient way to design and training of deep neural networks for temporal data, with the intrinsic ability to represent hierarchical and distributed temporal features [4, 5, 6].

In this paper, we investigate different approaches to RNN modeling (i.e., untrained stacked layers and fully-trained gated architectures), through an experimental comparison between RC and fully-trained RNNs on challenging real-world prediction tasks characterized by multivariate time-series. In particular, we perform a comparison between DeepESN, LSTM and GRU models on 4 polyphonic music tasks [7]. Since these datasets are characterized by sequences with high-dimensionality and complex temporal sequences, these challenging tasks are particularly suitable for RNNs evaluation [8]. Moreover, we consider ESN and simple RNN (Simple Recurrent Network - SRN) as baseline approaches for DeepESN and gated RNNs, respectively. The models are evaluated in terms of predictive accuracy and computation efficiency.

In a context in which the model design is difficult, especially for fully-trained RNNs, this paper would provide a first glimpse in the experimental comparison between different state-of-the-art recurrent models on multivariate time-series prediction tasks which still lacks in literature.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

2 Deep Echo State Networks

DeepESNs [4] extend Echo State Network (ESN) [9] models to the deep learning paradigm. Fig. 1 shows an example of a DeepESN architecture composed by a hierarchy of N_L reservoirs, coupled by a readout output layer.

In the following equations, $\mathbf{u}(t) \in \mathbb{R}^{N_U}$ and $\mathbf{x}^{(l)}(t) \in \mathbb{R}^{N_R}$ represent the external input and state of the l-th reservoir layer at step t, respectively. Omitting bias terms for the ease of notation, and using leaking-rate reservoir units, the state transition of the first recurrent layer is described as follows:

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

while for each layer $l > 1$ the state computation is performed as follows:

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

In eq. 1 and 2, $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ represents the matrix of input weights, $\hat{\mathbf{W}}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ is the matrix of the recurrent weights of layer l, $\mathbf{W}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ is the matrix that collects the inter-layer weights from layer $l-1$ to layer l, $a^{(l)}$ is the leaky parameter at layer l and $\mathbf{f}$ is the activation function of recurrent units implemented by a hyperbolic tangent ($\mathbf{f} \equiv \mathbf{tanh}$). Finally, the (global) state of the DeepESN is given by the concatenation of all the states encoded in the recurrent layers of the architecture $\mathbf{x}(t) = (\mathbf{x}^{(1)}(t), \ldots, \mathbf{x}^{(N_L)}(t)) \in \mathbb{R}^{N_L N_R}$.

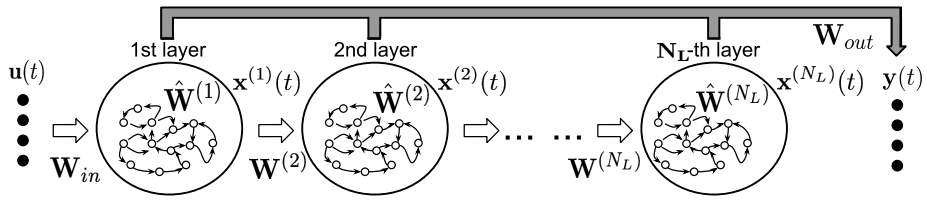

Fig. 1: Hierarchical architecture of DeepESN.

The weights in matrices $\mathbf{W}_{in}$ and $\{\mathbf{W}^{(l)}\}_{l=2}^{N_L}$ are randomly initialized from a uniform distribution and re-scaled such that $\|\mathbf{W}_{in}\|_2 = \sigma$ and $\|\mathbf{W}^{(l)}\|_2 = \sigma$ respectively, where σ is an input scaling parameter. Recurrent layers are initialized in order to satisfy the necessary condition for the Echo State Property of DeepESNs [10]. Accordingly, values in $\{\hat{\mathbf{W}}^{(l)}\}_{l=1}^{N_L}$ are randomly initialized from uniform distribution and re-scaled such that $\max_{1 \leq l \leq N_L} \rho\left((1 - a^{(l)})\mathbf{I} + a^{(l)}\hat{\mathbf{W}}^{(l)}\right) < 1$, where ρ is the spectral radius of its matrix argument, i.e. the maximum among its eigenvalues in modulus. The standard ESN case is obtained considering DeepESN with 1 single layer, i.e. when $N_L = 1$.

The output of the network at time-step t is computed by the readout as a linear combination of the activation of reservoir units, as follows: $\mathbf{y}(\mathbf{t}) = \mathbf{W}_{out}\mathbf{x}(t)$,

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

where $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_L N_R}$ is the matrix of output weights. This combination allows to differently weight the contributions of the multiple dynamics developed in the network's state. The training of the network is performed only on the readout layer by means of direct numerical methods. Finally, as pre-training technique we use the Intrinsic Plasticity (IP) adaptation for deep recurrent architectures, particularly effective for DeepESN and ESN architectures [4, 6].

3 Experimental Comparison

In this section we present the results of the experimental comparison performed between randomized and fully-trained RNNs. The approaches are assessed on polyphonic music tasks defined in [7]. In particular, we consider the following 4 datasets[1]: Piano-midi.de, MuseData , JSBchorales and Nottingham. A polyphonic music task is defined as a next-step prediction on 88-, 82-, 52- and 58- dimensional sequences for Piano-midi.de, MuseData, JSBchorales and Nottingham datasets, respectively. Each dimension of the input sequences corresponds to the input and the output dimension of the models. The tasks consist of classifying if the next note is played (value 1) or not (value 0) for each output dimension with a threshold value of 0.5. As the datasets consist in high-dimensional time-series characterized by heterogeneous sequences, sparse vector representations and complex temporal dependencies involved at different time-scales, they are considered challenging real-world benchmarks for RNNs [8].

Models' performance is measured by using the expected frame-level accuracy (ACC), commonly adopted as prediction accuracy in polyphonic music tasks [7], and computed as follows:

$$\text{ACC} = \sum_{t=1}^{T} TP(t) \Big/ \Big(\sum_{t=1}^{T} TP(t) + \sum_{t=1}^{T} FP(t) + \sum_{t=1}^{T} FN(t) \Big), \tag{3}$$

where T is the total number of time-steps, while $TP(t)$, $FP(t)$ and $FN(t)$ respectively denote the numbers of true positive, false positive and false negative notes predicted at time-step t.

Concerning DeepESN and ESN approaches, we considered reservoirs initialized with 1% of connectivity. Moreover, we performed a model selection on the major hyper-parameters considering spectral radius ρ and leaky integrator a values in $\{0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$, and input scaling σ values in $\{0.5, 1.5, 2.5\}$. Training of the readout was performed through ridge regression [9, 3] with regularization coefficient λ_r in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Moreover, based on the results of the design analysis in [6] on polyphonic music tasks, we set up DeepESN with $N_L = 30$ layers composed by $N_R = 200$ units, and ESN with $N_R = 6000$ recurrent units. We used an IP adaptation configured as in [4, 6] with a standard deviation of $\sigma_{IP} = 0.1$.

For what regards fully trained RNNs, we used the Adam learning algorithm [11] with a maximum of 2000 epochs. In order to regularize the learning process, we applied dropout methods, a clipping gradient with a value of 5 and an early stopping with a

[1] Piano-midi.de (`www.piano-midi.de`); MuseData (`www.musedata.org`); JSBchorales (chorales by J. S. Bach); Nottingham (`ifdo.ca/~seymour/nottingham/nottingham.html`).

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Model	total recurrent units	free-parameters	test ACC	computation time
Piano-midi.de				
DeepESN	6000	540088	**33.33 (0.11) %**	**386**
ESN	6000	540088	30.43 (0.06) %	748
SRN	652	540596	29.48 (0.35) %	3185
LSTM	316	539816	28.98 (2.93) %	2333
GRU	369	539566	31.38 (0.21) %	2821
MuseData				
DeepESN	6000	504082	**36.32 (0.06) %**	**789**
ESN	6000	504082	35.95 (0.04) %	997
SRN	632	503786	34.02 (0.28) %	8825
LSTM	307	504176	34.71 (1.17) %	18274
GRU	358	503072	35.89 (0.17) %	18104
JSBchorales				
DeepESN	6000	324052	**30.82 (0.12) %**	**83**
ESN	6000	324052	29.14 (0.09) %	140
SRN	519	323908	29.68 (0.17) %	341
LSTM	254	325172	29.80 (0.38) %	532
GRU	295	323372	29.63 (0.64) %	230
Nottingham				
DeepESN	6000	360058	69.43 (0.05) %	**677**
ESN	6000	360058	69.12 (0.08) %	1473
SRN	545	360848	65.89 (0.49) %	2252
LSTM	266	361286	70.00 (0.24) %	26175
GRU	309	359116	**71.50 (0.77)** %	11844

Table 1: Free-parameters and test ACC achieved by DeepESN, SRN, LSTM and GRU. Computation time represents the seconds to complete training and test.

patience value of 30. Then, we performed a model selection considering learning rate values in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and dropout values in $\{0.1, 0.2,\ 0.3, 0.4, 0.5\}$.

Since randomized and fully-trained RNNs implement different learning approaches, it is difficult to set up a fair experimental comparison between them. However, we faced these difficulties by considering a comparable number of free-parameters for all the models. The number of recurrent units and free-parameters considered in the models is shown in the second and third columns of Tab. 1. Each model is individually selected on the validation sets through a grid search on hyper-parameters ranges. We independently generated 5 guesses for each network hyper-parametrization (for random initialization), and averaged the results over such guesses.

In accordance with the different characteristics of the considered training approaches (direct methods for RC and iterative methods for fully-trained models) we preferred the most efficient method in all the considered cases. Accordingly, we used a MATLAB implementation for DeepESN and ESN models, and a Keras implementation for fully-trained RNNs. We measured the time in seconds spent by models in training and test procedures, performing experiments on a CPU "Intel Xeon E5, 1.80GHz, 16 cores" in the case of RC approaches, and on a GPU "Tesla P100 PCIe 16GB" in the case of fully-trained RNNs, with the same aim to give the best resource to each of them.

Tab. 1 shows the number of recurrent units, the number of free-parameters, the predictive accuracy and the computation time (in seconds) achieved by DeepESN, ESN, SRN, LSTM and GRU models. For what regards the comparison between RC approaches in terms of predictive performance, results indicate that DeepESN out-

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

performed ESN with an accuracy improvement of 2.90%, 0.37%, 1.68% and 0.31% on Piano-midi.de, MuseData, JSBchorales and Nottingha tasks, respectively. Concerning the comparison between fully-trained RNNs, GRU obtained a similar accuracy to SRN and LSTM models on JSBchorales task and it outperformed them on Piano-midi.de, MuseData and Nottingham tasks.

Further experiments confirm (though not fully reported here) that the number of units used for the comparison resulted appropriate for each model in terms of rate between performance and computational time. Indeed, the models are subject to a rapid performance deterioration if we decrease the current number of units, and vice versa, there is not a significant accuracy improving w.r.t. the computational cost if we increase the number of units.

The efficiency assessments show that DeepESN requires about less that one order of magnitude of computation time with respect to fully-trained RNNs, boosting the already striking efficiency of standard ESN models. Moreover, while ESN benefits in terms of efficiency only by exploiting the sparsity of reservoirs (with 1% of connectivity), in the case of DeepESN the benefit is intrinsically due to the architectural constraints involved by layering [6] (and are obtained also with fully-connected layers).

Overall, the DeepESN model outperformed all the other approaches on 3 out of 4 tasks, resulting extremely more efficient with respect to fully-trained RNNs.

4 Conclusions

In this paper, we performed an experimental comparison between radomized and fully-trained RNNs on challenging real-world tasks characterized by multivariate time-series. This kind of comparisons in complex temporal tasks, that is practically absent in literature especially for what regards efficiency aspects, offered the opportunity to assess efficient alternative models (ESN and DeepESN in particular) to typical RNN approaches (LSTM and GRU). Moreover, we assessed also the effectiveness of layering in deep recurrent architectures with a large number of layers (i.e., 30).

Concerning fully-trained RNNs, GRU outperformed the other gated RNNs on 3 out of 4 tasks and it was more efficient than LSTM in most cases. The effectiveness of GRU approaches found in our experiments is in line with the literature that deals with the design of adaptive gates in recurrent architectures.

For what regards randomized RNNs, the results show that DeepESN is able to outperform ESN in terms of prediction accuracy and efficiency on all tasks. Interestingly, this highlights that the layering aspect allows us to improve the effectiveness of RC approaches on multiple time-scales processing. Overall, the DeepESN model outperformed other approaches in terms of prediction accuracy on 3 out of 4 tasks. Finally, DeepESN required much less time in computation time with respect to the others models resulting in an extremely efficient model able to compete with the state-of-the-art on challenging time-series tasks.

More in general, it is interesting to highlight the gain in the prediction accuracy showed by the multiple time-scales processing capability obtained by layering in deep RC models and by using adaptive gates in fully-trained RNNs in comparison to the respective baselines (ESN and SRN, respectively). Also, it is particularly interesting to note the comparison between models with the capability to learn multiple time-scales dynamics (LSTM and GRU) and models showing an intrinsic capability to develop such kind of hierarchical temporal representations (DeepESN), which was completely lacking in literature.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

In addition to provide insights on such general issues, this paper would contribute to show a practical way to efficiently approach the design of learning models in the scenario of deep RNN, extending the set of tools available to the users for complex time-series tasks. Indeed, the first empirical results provided in this paper seem to indicate that some classes of models are sometimes uncritically adopted, i.e. despite their cost, guided by the natural popularity due to their software availability (GRU, LSTM). The same diffusion of software tools deserve more effort on the side of the other models (DeepESN class), although the first instances are already available[2].

References

[1] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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

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

[5] C. Gallicchio, A. Micheli, and L. Pedrelli. Hierarchical Temporal Representation in Linear Reservoir Computing. In *Neural Advances in Processing Nonlinear Dynamic Signals*, pages 119–129, Cham, 2019. Springer International Publishing. WIRN 2017.

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

[7] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[8] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. In *ICASSP 2013*, pages 8624–8628. IEEE, 2013.

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

[10] C. Gallicchio and A. Micheli. Echo State Property of Deep Reservoir Computing Networks. *Cognitive Computation*, 9(3):337–350, 2017.

[11] D. Kinga and J. B. Adam. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, volume 5, 2015.

[2]DeepESN implementations are made publicly available for download both in MATLAB (see `https://it.mathworks.com/matlabcentral/fileexchange/69402-deepesn`) and in Python (see `https://github.com/lucapedrelli/DeepESN`).