Fast Spectral Radius Initialization for Recurrent Neural Networks

Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli

Department of Computer Science, University of Pisa
Largo B. Pontecorvo 3, Pisa, Italy
{gallicch, micheli, luca.pedrelli}@di.unipi.it

Abstract. In this paper we address the problem of grounded weights initialization for Recurrent Neural Networks. Specifically, we propose a method, rooted in the field of Random Matrix theory, to perform a fast initialization of recurrent weight matrices that meet specific constraints on their spectral radius. Focusing on the Reservoir Computing (RC) framework, the proposed approach allows us to overcome the typical computational bottleneck related to the eigendecomposition of large matrices, enabling to efficiently design large reservoir networks and hence to address time-series tasks characterized by medium/big datasets. Experimental results show that the proposed method enables an accurate control of the spectral radius of randomly initialized recurrent matrices, providing an initialization approach that is extremely more efficient compared to common RC practice.

Keywords: Recurrent Neural Networks · Echo State Networks · Spectral Radius of Random Matrices · Time-series Prediction

1 Introduction

Recurrent Neural Networks (RNNs) represent a powerful class of learning models suitable for time-series prediction and sequence classification. In particular, the Reservoir Computing (RC) framework [9] provides an efficient approach for the implementation of RNN architectures in which the recurrent part of the network is left untrained after initialization. Thereby, studies in the field of RC allow focusing the research investigations towards the analysis of the intrinsic characterizations of RNN models, independently from learning aspects. A major issue in this context is related to stability of recurrent network dynamics in presence of driving input signals, which is typically addressed by requiring the network to fulfill a stability property known as the Echo State Property (ESP) [9]. Literature conditions for the ESP link the stability behavior of the recurrent system to spectral properties, and in particular to the spectral radius, of the recurrent weight matrix in the RNN architecture, resulting in a network initialization procedure that requires to perform an eigendecomposition algorithm. Hence, despite their simplicity, commonly used RC initialization procedures can become impractical and lead to a computational bottleneck in the case of real-world tasks for which a large number of recurrent units (in the order of thousands

[6]) is needed. Besides, from a broader perspective, studies on RC-like initialization are of interest also for fully-trained RNNs [11], where a proper control of spectral properties of the recurrent weight matrices can be important to alleviate vanishing and exploding gradients [10].

In this paper, we propose an efficient approach to initialize the recurrent weights of an RNN in order to obtain an approximation of the desired spectral radius, avoiding to perform expensive algorithms. In particular, we propose two theoretically grounded algorithms stemming from Random Matrix theory and circular law [1][12][2] for the initialization of fully-connected and sparse matrices from a uniform distribution. The proposed approach is experimentally assessed in terms of both accuracy of spectral radius estimation and computational advantage with respect to common RC initialization practice. Moreover, we show how to exploit the proposed approach for implementing large RC networks (up to 10000 units) in challenging time-series tasks, considering both RC benchmarks and real-world datasets in the area of music processing.

2 Echo State Networks

The Echo State Network (ESN) [8] model is an efficient implementation of the RNN approach within the RC framework. The architecture is composed of two parts, a non-linear recurrent layer called reservoir and a linear output layer (the readout). The reservoir is randomly initialized and left untrained, while the readout computes the output of the network exploiting the temporal state representation developed by the reservoir part. Figure 1 shows the ESN architecture.

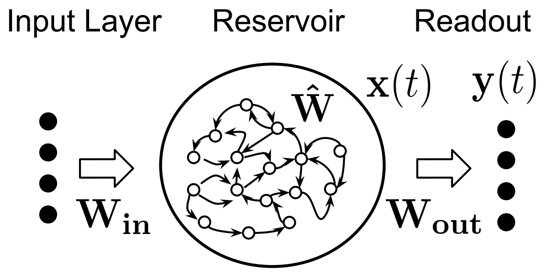

Fig. 1. The ESN architecture.

Omitting the bias terms in the following formula for the ease of notation, the state transition function of ESN is defined as follows:

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

where $\mathbf{u}(t) \in \mathbb{R}^{N_U}$ and $\mathbf{x}(t) \in \mathbb{R}^{N_R}$ are respectively the *input* and the reservoir *state* at time t, $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ is the matrix of the input weights, $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the matrix of the recurrent weights, and **tanh** represents the element-wise application of the hyperbolic tangent activation function.

Reservoir parameters are initialized according to the Echo State Property (ESP) [8,4]. Typically, following the necessary condition for the ESP, elements in $\hat{\mathbf{W}}$ are randomly initialized from a uniform distribution and then re-scaled such that $\rho(\hat{\mathbf{W}})$, which denotes the *spectral radius* of $\hat{\mathbf{W}}$ (i.e. its maximum absolute eigenvalue), is smaller than 1:

$$\rho(\hat{\mathbf{W}}) < 1. \tag{2}$$

The typical ESN initialization (ESNinit) is shown in Algorithm 1. The cost of the ESNinit procedure is dominated by the eigendecomposition algorithm computed in line 3 with an asymptotic cost of $\mathcal{O}({N_R}^3)$. Concerning the input weights, the values in matrix $\mathbf{W}_{in}$ are randomly selected from a uniform distribution over $[-scale_{in}, scale_{in}]$, where $scale_{in}$ is the *input scaling* parameter.

Algorithm 1 ESNinit

```
1: procedure ESNINIT($N_R$, $\hat{\rho}$)
2:     $\hat{\mathbf{W}}$ = uniform(($N_R, N_R$), $-1, +1$)    ▷ generate matrix $\in \mathbb{R}^{N_R \times N_R}$ from u.d.
3:     $\rho$ = eig($\hat{\mathbf{W}}$)    ▷ eigendecomposition of $\hat{\mathbf{W}}$
4:     $\hat{\mathbf{W}}$ = ($\hat{\mathbf{W}}/\rho$)*$\hat{\rho}$    ▷ re-scale spectral radius of $\hat{\mathbf{W}}$ to $\hat{\rho}$
       return $\hat{\mathbf{W}}$
```

The output of the network at time t is computed by the readout as a linear combination of the reservoir units activations, i.e. $\mathbf{y}(t) = \mathbf{W}_{out}\mathbf{x}(t)$, where $\mathbf{y}(t) \in \mathbb{R}^{N_Y}$ is the *output* at time t, and $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_R}$ is the matrix of output weights. The readout layer is the only part of the network that is trained, typically in closed form by means of pseudo-inversion or ridge regression [9].

3 Fast Spectral Initialization

In this section, we introduce two efficient methods that we call Fast Spectral Initialization (FSI) and Sparse FSI (S-FSI). Assuming that the aim is to build a reservoir weight matrix with a desired spectral radius $\hat{\rho}$, FSI and S-FSI algorithms are defined in order to obtain a matrix $\hat{\mathbf{W}}$ such that $\rho(\hat{\mathbf{W}})$ is an approximation of the desired spectral radius $\hat{\rho}$. In the following, we propose a theoretical support rooted in the field of Random Matrix theory.

Theorem 1. *(FSI) Let* $\hat{\boldsymbol{W}} \in \mathbb{R}^{N_R \times N_R}$ *be the matrix of recurrent weights whose entries are i.i.d. copies of a random variable* w *with uniform distribution in* $\left[-\frac{\hat{\rho}}{\sqrt{N_R}}\frac{6}{\sqrt{12}}, +\frac{\hat{\rho}}{\sqrt{N_R}}\frac{6}{\sqrt{12}}\right]$*, then*

$$\rho(\hat{\boldsymbol{W}}) \to \hat{\rho} \;\; for \;\; N_R \to \infty. \tag{3}$$

Proof. Let $\overline{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ be the matrix whose entries are i.i.d. copies of a random variable $\overline{w}$ with uniform distribution in $\left[-\frac{6}{\sqrt{12}}, +\frac{6}{\sqrt{12}}\right]$, hence $\overline{w}$ has mean 0 and variance 1. Then, from results in [1] and [12] (see [2] for details) we obtain that

$$\frac{\rho(\overline{\mathbf{W}})}{\sqrt{N_R}} \to 1 \text{ for } N_R \to \infty. \tag{4}$$

Let $\hat{\mathbf{W}} = \frac{\hat{\rho}}{\sqrt{N_R}}\overline{\mathbf{W}}$, hence random variables in $\hat{\mathbf{W}}$ are in $\left[-\frac{\hat{\rho}}{\sqrt{N_R}}\frac{6}{\sqrt{12}}, +\frac{\hat{\rho}}{\sqrt{N_R}}\frac{6}{\sqrt{12}}\right]$. Moreover, $\overline{\mathbf{W}} = \frac{\sqrt{N_R}}{\hat{\rho}}\hat{\mathbf{W}}$; by Property in Equation (4) and linearity of spectral radius:

$$\frac{\rho(\frac{\sqrt{N_R}}{\hat{\rho}}\hat{\mathbf{W}})}{\sqrt{N_R}} = \frac{\sqrt{N_R}}{\hat{\rho}}\frac{\rho(\hat{\mathbf{W}})}{\sqrt{N_R}} \xrightarrow{N_R\to\infty} 1,$$

then by $\rho(\hat{\mathbf{W}}) \to \hat{\rho}$ for $N_R \to \infty$. □

From Theorem 1 we can note that w has mean 0 and variance $\frac{\hat{\rho}^2}{N_R}$. Overall, Theorem 1 means that initializing the values of $\hat{\mathbf{W}}$ from a uniform distribution defined in $\left[-\frac{\hat{\rho}}{\sqrt{N_R}}\frac{6}{\sqrt{12}}, +\frac{\hat{\rho}}{\sqrt{N_R}}\frac{6}{\sqrt{12}}\right]$ (with mean 0 and variance $\frac{\hat{\rho}^2}{N_R}$) we obtain $\rho(\hat{\mathbf{W}}) \approx \hat{\rho}$ with an approximation accuracy that increases with the increasing of N_R. The FSI initialization is defined in Algorithm 2.

Algorithm 2 FSI initialization

```
1: procedure FSI($N_R$, $\hat{\rho}$)
2:     $a = -\frac{\hat{\rho}}{\sqrt{N_R}}\frac{6}{\sqrt{12}}$, $b = +\frac{\hat{\rho}}{\sqrt{N_R}}\frac{6}{\sqrt{12}}$
3:     $\hat{\mathbf{W}} = \texttt{uniform}((N_R, N_R), a, b)$     ▷ generate matrix $\in \mathbb{R}^{N_R \times N_R}$ from u.d. [a,b]
       return $\hat{\mathbf{W}}$
```

Since we are interested to propose efficient approaches for medium/big data, we also take a step forward considering typical approaches for ESN based on sparse reservoirs. Accordingly, we introduce an extension of Theorem 1 for sparse matrices $\hat{\mathbf{W}}_C$, where C (called also connectivity) is the percentage of the incoming connections of each neuron in the recurrent layer.

Conjecture 1. (S-FSI) Let $\hat{\mathbf{W}}_C \in \mathbb{R}^{N_R \times N_R}$ be the matrix of recurrent weights with connectivity C whose non-null entries are i.i.d. copies of a random variable w with uniform distribution in $\left[-\frac{\hat{\rho}}{\sqrt{CN_R}}\frac{6}{\sqrt{12}}, +\frac{\hat{\rho}}{\sqrt{CN_R}}\frac{6}{\sqrt{12}}\right]$, then

$$\rho(\hat{\mathbf{W}}_C) \to \hat{\rho} \text{ for } N_R \to \infty. \tag{5}$$

The Conjecture 1 states that if we initialize a sparse recurrent matrix $\hat{\mathbf{W}}_C$ from a uniform distribution in $\left[-\frac{\hat{\rho}}{\sqrt{CN_R}}\frac{6}{\sqrt{12}}, +\frac{\hat{\rho}}{\sqrt{CN_R}}\frac{6}{\sqrt{12}}\right]$ (with mean 0 and variance $\frac{\hat{\rho}^2}{CN_R}$) we obtain $\rho(\hat{\mathbf{W}}) \approx \hat{\rho}$ with an accuracy of the approximation that increases with the increasing of N_R.

The insight that allowed us to formulate this conjecture is relatively simple. We considered a sparse matrix with connectivity C initialized from a uniform distribution as a matrix with non-null values of dimension $\lfloor CN_R \rfloor \times \lfloor CN_R \rfloor$. However, since the theoretical aspects of this case are not trivial, we decided to propose this enunciate as a conjecture delegating the theoretical proof to further works. The S-FSI initialization is defined in Algorithm 3. It is worth to note that in Algorithms 2 and 3 we avoid the cubic cost spent for eigendecomposition computation in Algorithm 1.

Algorithm 3 S-FSI initialization

```
1: procedure S-FSI($N_R$, $\hat{\rho}$, $C$)
2:     $\hat{\mathbf{W}} = zeros((N_R, N_R))$                ▷ generate null matrix $\in \mathbb{R}^{N_R \times N_R}$
3:     $a = -\frac{\hat{\rho}}{\sqrt{CN_R}}\frac{6}{\sqrt{12}}$, $b = +\frac{\hat{\rho}}{\sqrt{CN_R}}\frac{6}{\sqrt{12}}$
4:     $N = \lfloor C{*}N_R \rfloor$                         ▷ integer part of $C{*}N_R$
5:     for $irow$ in 1, ..., $N_R$ do
6:         $P = \texttt{randperm}(N_R, N)$      ▷ select only P indices to initialize (sparsity)
7:         $\hat{\mathbf{W}}(irow, P) = \texttt{uniform}((1, N_R), a, b)$  ▷ generate vec. $\in \mathbb{R}^{N_R}$ from u.d. [a,b]
       return $\hat{\mathbf{W}}$
```

4 Experimental Assessments

In this section, we experimentally assess the FSI and S-FSI approaches introduced in Section 3 to efficiently exploit large RNNs and to improve models accuracy on medium/big time-series tasks. First, we empirically evaluate Theorem 1 and Conjecture 1. Then, we compare the computational times between our approaches and ESNinit. Finally, we show the benefits of the proposed method on RC benchmarks and real-world tasks characterized by medium/big dataset for time-series prediction.

4.1 Empirical Assessment

In this section, we perform an empirical evaluation of Theorem 1 and Conjecture 1 measuring the difference between the spectral radius of the recurrent matrices initialized by our proposed methods, i.e. $\rho(\texttt{FSI}(N_R, \hat{\rho}))$ and $\rho(\texttt{S-FSI}(N_R, \hat{\rho}, C))$, and the desired value of the spectral radius $\hat{\rho}$. Accordingly, we measure the error by the following equation:

$$ERR(\hat{\rho}) = \rho(\texttt{FSI}(N_R, \hat{\rho})) - \hat{\rho}. \tag{6}$$

Analogously, we calculate the error for the S-FSI case as $\rho(\texttt{S-FSI}(N_R, \hat{\rho}, C)) - \hat{\rho}$.

In the following experiments, for each reservoir hyper-parameterization, we independently generated 10 reservoir guesses, and the results has been averaged over such guesses. Figure 2**a)** shows ERRs obtained at the increasing of N_R by FSI and S-FSI approaches. As expected from Theorem 1 we can note that the ERR obtained by FSI algorithm decreases with the increase of N_R, remaining below 0.03. Moreover, S-FSI obtained less that 0.02 ERR for all considered number of units. Note that these ERR results are very good for practical cases. Interestingly, the results achieved by FSI and S-FSI are very similar despite they work on different matrix structures. Overall, although Conjecture 1 (at the best of our knowledge) is not theoretically proved we think that these results represent a good empirical support in favor of the validity of the Conjecture 1.

Here, we assess the benefit in terms of computational time (on a CPU Intel Xeon E5, 1.80GHz, 16 cores) offered by proposed methods. Accordingly, we measured the difference between the times (in seconds) required by ESNinit and our approaches. Figure 2**b**) shows the time saved by FSI and S-FSI with respect to ESNinit at the increase of N_R.

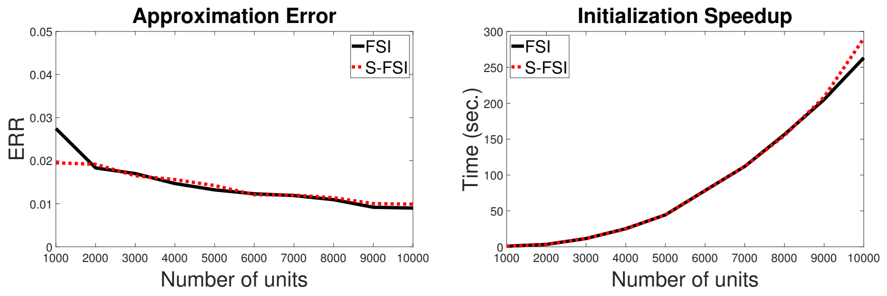

Fig. 2. Numerical results obtained for increasing number of units. **Left**: ERRs (with $\hat{\rho} = 1$). **Right**: Initialization speedup (in sec.) w.r.t. ESNinit.

In both fully and sparse reservoirs cases, the trend of curves shown in Figure 2**b**) confirms the cubic advantage in terms of computation time achieved by our algorithms as described in Section 3. In particular, in the case of fully-connected reservoirs composed by 10000 units, ESNinit spent 265 seconds while FSI required only 2.15 seconds. The best result is achieved by S-FSI that spent only 0.43 seconds to initialize a sparse matrix of size 10000×10000 with respect to ESNinit that required 290 seconds.

4.2 Experimental Assessment on Time-series Prediction Tasks

In this section, we experimentally show how the proposed approaches are able to improve the model accuracy on medium/large datasets characterized by time-series prediction exploiting large reservoirs in an efficient way.

Experimental Setup. In order to show that very large reservoirs can be effectively exploited to obtain better performance, we performed a model selection on the validation set on a number of units N_R in $\{1000, 2000, 3000, 4000, 5000\}$ for RC benchmarks and N_R in $\{4000, 6000, 8000, 10000\}$ with a reservoir connectivity $C = 0.01$ for real-world tasks. Moreover, in the model selection we considered spectral radius ρ values in $\{0.5, 0.6, 0.7, 0.8, 0.9\}$, and $scale_{in} = 1$ for RC benchmarks and $scale_{in}$ values in $\{0.01, 0.1\}$ for real-world tasks. The training was performed through ridge regression [8,9] with regularization coefficient λ_r in $\{10^{-16}, 10^{-15}, ..., 10^{-1}\}$ for RC benchmarks and λ_r in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ for real-world tasks.

RC Benchmarks. First, we assessed the FSI approach on 3 RC benchmarks: Mackey-Glass time-series, 10th order NARMA system and Santa Fe Laser time-series, that we call in the following MG, NARMA and LASER tasks, respectively. For the RC benchmarks we measured the mean squared error (MSE).

The MG task is a standard benchmark [8] for chaotic time-series next-step prediction. The input sequence is obtained by a discretization of the following equation:

$$\frac{\partial u(t)}{\partial t} = \frac{0.2u(t-17)}{1+u(t-17)} - 0.1u(t). \tag{7}$$

The input sequence was shifted by -1 and fed to **tanh** function as performed in [8]. We generated a time-series with 10000 time-steps, first $N_{train} = 5000$ time-steps were used for training, last $N_{validation} = 1000$ of the training set were used for validation and final $N_{test} = 5000$ for test. The first $N_{transient} = 1000$ time-steps was discarded before the readout training.

The LASER task [4] consists in a next-step prediction of a sequence obtained by sampling the intensity of a far-infrared laser in a chaotic regime. The input sequence consists in 10093 time-steps, first $N_{train} = 5000$ time-steps were used for training, last $N_{validation} = 1000$ of the training set were used for validation and final $N_{test} = 5093$ for test. The first $N_{transient} = 1000$ time-steps were used as initial transient.

The NARMA task [4] consists in the output prediction of a 10-th order non-linear autoregressive moving average system. The input sequence $u(t)$ is generated on a uniform distribution in $[0, 5]$. The target sequence is computed by the following equation:

$$\hat{y}(t) = 0.3\hat{y}(t-1) + 0.05\hat{y}(t-1)(\sum_{i=1}^{10} \hat{y}(t-i)) + 1.5u(t)(t-10)u(t-1) + 0.1. \tag{8}$$

Given $u(t)$ the aim of the task is to predict $\hat{y}(t)$. The input sequence consists in 4200 time-steps, first $N_{train} = 2200$ time-steps were used for training, last $N_{validation} = 1000$ of training set were used for validation and final $N_{test} = 2000$ for test. The initial transient was set at $N_{transient} = 200$.

The results obtained by FSI and ESN initialization through eigendecomposition (ESNinit) are shown in Table 1. The models obtained the best results with $N_R = 5000$ recurrent units in all tasks. Therefore, in this cases, a fast approach to approximate the spectral radius of large recurrent matrices is very important to improve the performance in an efficient way. Moreover, we can see that FSI approach obtained very similar test MSEs to those obtained by ESNinit highlighting the effectiveness of the spectral radius approximation. Interestingly, FSI required only 0.51 seconds to initialize the recurrent layer with respect to 45 seconds required by ESNinit for each hyper-parameters configuration. In this concern, note that the computational advantage of the proposed approach is further amplified when considering a grid search for the hyper-parametrization. In the following, we take a step forward considering larger recurrent matrices and more challenging and bigger datasets.

Approach	Test MSE	$ERR(\hat{\rho})$	Init. time	Training time	Units
MG					
FSI	8.32e-15 (1.06e-14)	0.012 (0.002)	0.51 sec.	222 sec.	5000
ESNinit	8.97e-15 (1.44e-14)	-	45 sec.		
LASER					
FSI	3.90e-3 (1.83e-04)	0.007 (0.001)	0.51 sec.	207 sec.	5000
ESNinit	4.14e-3 (2.96e-04)	-	45 sec.		
NARMA					
FSI	2.80e-4 (2.02e-05)	0.012 (0.002)	0.51 sec.	83 sec.	5000
ESNinit	2.78e-4 (2.30e-05)	-	45 sec.		

Table 1. Test MSE and initialization time (Init. time column). $ERR(\hat{\rho})$ column: ERR obtained by FSI where $\hat{\rho}$ is the selected spectral radius by model selection. Training time column: time required for readout training, averaged over the FSI and ESNinit reservoir initialization methods (it does not depend on the initialization approach). Units column: number of recurrent units selected by model selection.

Poliphonic Music Tasks. Here we consider S-FSI for sparse reservoirs on challenging real-world tasks characterized by medium/big datasets. The sparse connections implemented in large recurrent layers allow models to perform efficient computations despite the big quantity of units. The approach is assessed on polyphonic music tasks defined in [3]. 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. The datasets comprise a large number of timesteps (in the order of hundreds of thousands, see further details in [6,3]). Training, validation and test sets of piano-rolls used in [3] are publicly available[2]. Further details concerning the polyphonic music datasets (omitted here for the sake of conciseness) are reported in [6].

The prediction accuracy of the models is measured in expected frame-level accuracy (ACC) adopted in polyphonic music tasks [3], and computed as follows:

$$\mathrm{ACC} = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + \sum_{t=1}^{T} FP(t) + \sum_{t=1}^{T} FN(t)}, \tag{9}$$

[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`).

[2] `www-etud.iro.umontreal.ca/~boulanni/icml2012`

Approach	test ACC	$ERR(\hat{\rho})$	Init. time	Training time	units
Piano-midi.de					
S-FSI	28.80 (0.01) %	0.009 (0.003)	0.43 sec.	124 sec.	10000
ESNinit	28.80 (0.03) %	-	290 sec.		
MuseData					
S-FSI	33.79 (0.01) %	0.005 (0.002)	0.43 sec.	101 sec.	10000
ESNinit	33.80 (0.03) %	-	290 sec.		
JSBchorales					
S-FSI	32.18 (0.26) %	0.005 (0.002)	0.43 sec.	29 sec.	10000
ESNinit	32.14 (0.44) %	-	290 sec.		
Nottingham					
S-FSI	70.28 (0.02) %	0.009 (0.003)	0.43 sec.	280 sec.	10000
ESNinit	70.25 (0.04) %	-	290 sec.		

Table 2. Test MSE and initialization time (Init. time column). $ERR(\hat{\rho})$ column: ERR obtained by S-FSI where $\hat{\rho}$ is the selected spectral radius by model selection. Training time column: time required for readout training, averaged over the S-FSI and ESNinit reservoir initialization methods (it does not depend on the initialization approach). Units column: number of recurrent units selected by model selection.

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.

Table 2 shows the results obtained by S-FSI and ESNinit. Similarly to results obtained in RC benchmarks, here the models obtained the best results with $N_R = 10000$ units. Moreover, the test ACCs obtained by S-FSI and ESNinit are very similar. These results show that there is no need to compute the exact spectral radius to obtain good performance. Moreover, the use of sparsity operation enables to further improve both initialization and training times for the models. Indeed, S-FSI resulted significantly more efficient than FSI. Note that the order of magnitudes of times required by ESNinit is comparable with the ones required by readout training. Overall, S-FSI is extremely more efficient than ESNinit requiring only 0.43 seconds to initialize a recurrent matrix of size 10000×10000 with respect to 290 seconds required by ESNinit for each hyper-parameters configuration. In conclusion, the proposed methods allowed us to design extremely efficient ESN models able to reach a performance that is close to state-of-the-art results present in literature on polyphonic music tasks [6,7].

5 Conclusions

In this paper, we introduced a fast initialization approach for RNNs able to accurately control the spectral radius of recurrent matrices. First, we provided theoretical support and then we defined FSI and S-FSI algorithms for fully-connected and sparse matrices. Moreover, we empirically verified the validity of the theoretical tools as the number of recurrent units increases. Finally, we evaluated the proposed approaches in large RC networks on RC benchmarks and real-world tasks. The results show that the proposed approaches are extremely more efficient than typical spectral radius initialization methods allowing reservoir models to exploit many units and to improve accuracy. Overall, the proposed procedures overcome the computational bottleneck typical of ESNs initialization providing practical tools for the rapid design of such models.

The methodologies proposed in this paper open the way to further works based on the exploitation of RC in Big Data tasks especially for what regards the synergy with the deep learning paradigm such as in Deep Echo State Network [5] models. More in general, future studies can regard the assessment of our initialization approaches for fully-trained deep RNNs.

References

1. Bai, Z., Yin, Y.: Limiting behavior of the norm of products of random matrices and two problems of geman-hwang. Probability theory and related fields **73**(4), 555–569 (1986)
2. Bordenave, C., Caputo, P., Chafaï, D., Tikhomirov, K.: On the spectral radius of a random matrix. arXiv preprint arXiv:1607.05484 (2016)
3. Boulanger-Lewandowski, N., Bengio, Y., Vincent, P.: Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In: Proceedings of the 29th International Conference on Machine Learning (2012)
4. Gallicchio, C., Micheli, A.: Architectural and markovian factors of echo state networks. Neural Networks **24**(5), 440–456 (2011)
5. Gallicchio, C., Micheli, A., Pedrelli, L.: Deep reservoir computing: a critical experimental analysis. Neurocomputing **268**, 87–99 (2017)
6. Gallicchio, C., Micheli, A., Pedrelli, L.: Design of deep echo state networks. Neural Networks **108**, 33–47 (2018)
7. Gallicchio, C., Micheli, A., Pedrelli, L.: Comparison between DeepESNs and gated RNNs on multivariate time-series prediction. Proceedings of the 27th European Symposium on Artificial Neural Networks (ESANN) (in press)
8. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
9. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review **3**(3), 127–149 (2009)
10. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: Proceedings of International Conference on Machine Learning (ICML). vol. 28, pp. 1310–1318 (2013)
11. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International conference on machine learning. pp. 1139–1147 (2013)
12. Tao, T., Vu, V., Krishnapur, M., et al.: Random matrices: Universality of esds and the circular law. The Annals of Probability **38**(5), 2023–2065 (2010)