Deep Reservoir Neural Networks for Trees

Claudio Gallicchio[a], Alessio Micheli[a]

[a]*Department of Computer Science, University of Pisa*

Abstract

Tree structured data are a flexible tool to properly express many forms of hierarchical information. However, learning of such data through deep recursive models is particularly demanding. We will show through the introduction of the Deep Tree Echo State Network model (DeepTESN) that the randomized Neural Networks framework offers a formidable approach to allow an efficient treatment of learning in tree structured domains by deep architectures. Theoretical properties, for the Reservoir Computing setup constraints, and empirical behavior of the proposed approach are analyzed, showing its feasibility and accuracy.

Keywords: Reservoir Computing, Deep Echo State Networks, Recursive Neural Networks, Learning in Structured Domains, Echo State Property

1. Introduction

Representing data by abstract structures is a mainstream in the development of Computer Science and Artificial Intelligence from their origins, opening solutions both in terms of efficacy and in terms of efficiency in many applicative fields. In particular, the treatment of trees offers a natural way to represent hierarchical information in may real-world domains, whereas the aim in terms of Machine Learning is to learn a map directly from a domain of trees to an output space representing some properties associated to the input structure. At the same time, the fruitful exploitation of deep architectures moves the focus of recent research toward complex architectures for the entire class of Neural Networks, while often entailing an increasingly high computation cost.

Hence, the main problem addressed in this paper can be expressed by a short key question concerning the possibility to make efficient and deep learning for structured data. To further explain the motivations, in the following we gently introduce the three main ingredients that provide a ground for this aim and to the proposed approach, namely the *Recursive Neural Networks*, the *deep architectures* and the *Reservoir Computing (randomized)* approach.

The *Recursive Neural Networks* (RecNN) are an extension of Recurrent Neural Networks (RNN) used for the treatment of hierarchical/recursive data in

Email addresses: gallicch@di.unipi.it (Claudio Gallicchio), micheli@di.unipi.it (Alessio Micheli)

form of tree data structures. This approach has a long history, which has its roots in the first proposals of models and applications (about 20 years ago) [9, 14, 24, 43] until composing a framework that includes different paradigms of learning and applicative areas. The common idea underlying all these variants is to deal with hierarchical structured data using a recursive computation implemented by a state-transitions system with free-parameters. This approach is naturally suited to treat the variable size and variable structure of trees (typically in the form of rooted k-ary trees), and to realize adaptive transductions from structured input domain to an output domain, in the form of scalar or structured values. The main concepts founding the approach are the following [14, 38]: (i) a *structured encoding*, whereas a representation of a structure is obtained hierarchically as composition of the representations of its substructures, (ii) the *adaptivity*, which is gained by implementing the transduction in the form of different classes of neural networks or probabilistic approaches and hence through the use of free-parameters tuned by a learning algorithm, and finally, (iii) the parsimony, based on the sharing of the free-parameters (*weight sharing*) associated to the nodes of the input trees in the structured encoding. From a different point of view, starting from the advantages of dynamical systems (i.e. of RNNs) in modeling non-stationary data (with respect to vector-based feed-forward neural networks), RecNNs extend them from sequential to tree/hierarchical structures. The recursive learning framework nowadays includes approaches for unsupervised learning (see [25, 27] and references therein), generative versions in the form of hidden tree Markov models (HTMM) [3, 14, 23] and long short-term memory networks [44], up to extensions to directed acyclic graphs [8, 39] and to the development of kernels from families of HTMM [5]. The theoretical ground for the universal approximation capability of RecNN over tree domains has been provided in [26, 28, 29]. On the applications side it is worth to mention successful results developed in fields such as document processing [1, 3, 12, 13, 15], cheminformatics [7, 9, 38], bioinformatics [6], and the recent applications to sentence parse trees (see, e.g., [44]), which highly popularized the use of RecNNs in the Natural Language Processing (NLP) area.

At the same time (as a second component of our ingredients), recent developments of Machine Learning models push toward the use of sophisticated and large neural networks architectures, as in the case of the *deep learning* framework, that typically require a growing computation demand. The combination with the treatment of structured data requires even more computational effort, as the models are typically "deep and wide" to face the challenges brought about by the increasing complexity of the data domain from flat to the structured one. An example of a recent framework (from the point of view of non-Euclidean spaces) including different deep approaches for structured data (though often extended to graphs) can be the "geometric deep learning" [10].

In this context, the *randomized* learning techniques [19, 36, 42, 49] offer a redoubled approach for the design of very efficient neural networks models able to address such complexity (and they are the third ingredient of our background). In fact, a modular solution, typical of Echo State Networks (ESNs) [32, 33] and in general of Reservoir Computing (RC) [37, 47], permits to separate the

treatment of the recursive input structures through a dynamical system corresponding to a tree-automata state machine with randomized fixed parameters called *reservoir* (so to avoid the training of the recursive part, which is the most costly part), and an output component, realized through simple linear *readout* model trained by a regularized closed-form solution. Exploiting these ideas, the Tree ESN (TESN) model [15] has been introduced for the design of RecNN in the RC framework (combining the RecNN and RC ingredients).

Randomized approaches have been also exploited to implement multi-layered RNNs (combining other two of the proposed ingredients, namely RC and deep architectures). Specifically, the DeepESN [20, 21] has shown to be able to develop a hierarchical multiple time-scales representation of the sequential input data while preserving the efficiency of the RC paradigm, and to outperform the shallow counterpart (with the same network dimension) for time-series predictions. Further support to analysis of the intrinsic properties of a deep RNN architectural construction has been introduced under a dynamical system perspective [17] and in relation to stability analysis through local Lyapunov exponents [22] (a summary of the recent results can be found in [16]).

Through these preliminaries naturally emerges the demand for a two-fold exploitation of the randomized neural networks recursive processing of *hierarchical data* and of the hierarchical abstract representations allowed by the *deep* architectural construction. Indeed, current DeepESN models are restricted to sequential data, and, at the same time, the TESN does not exploit the ideas introduced with the deep learning paradigm, in particular the opportunity to introduce different (stacked) layers of recursive units in the model's architecture. For instance, preliminary results for a deep fully trained RecNN [31] witness the current interest and potentiality in exploring such possibilities, while leaving open the efficiency issues waiving the use of randomized approaches in the model design. Finally, the lack of literature results on the theoretical properties concerning the stability conditions for the model setup, and on the analysis of the empirical behavior of the combination of randomized versions of RecNN with deep architectures further motivates our study.

On the basis of such concepts, rooted in the RecNN and TESN approaches, and on the recent studies on the DeepESN advantages [16], we propose the study of a randomized neural network model for trees with a deep recursive architecture, called in the following Deep Tree ESN (DeepTESN). Besides the efficiency inherited by the RC approach, following the development of the DeepESN idea, in DeepTESN the architecture of the reservoir is constrained by imposing a layered structure that introduces, for the first time at the best of our knowledge, the properties of deep recurrent models in the class of recursive randomized neural networks. It is also important to stress that the deep part in DeepTESN is not added as a deep feed-forward component placed on top or within the model architecture but it is formed through a layered construction of the recursive part. Moreover, such deep layering is explicitly built in the reservoir architecture and it is not just implicitly considered by looking at the deep "unfolding network" derived by unrolling the model through the tree input structure. The proposed hierarchical construction of recursive layers allows us to exploit the

effect of composition typical of the hierarchical abstraction in deep learning for the internal tree recursive representation. In details, we will show the use of different stacked (reservoir) layers of recursive units, progressively more distant from the node input labels and organized such that the weighted connections among them run only from the lower to the higher layers (i.e. with a constrained architecture with respect to a fully recursively connected shallow RecNN).

Combining the ESN randomized neural networks approach, the recursive concept, and the deep construction we aim to show the feasibility of a "deep and efficient" learning way for structured data. To support such an aim, we will discuss the efficiency of the new DeepTESN approach, and we will show that the original TESN is made even more efficient by the proposed hierarchical layered deep construction. Moreover, we will show how the stability condition for the processing in the deep reservoirs can be extended to treat tree structured information. In particular, besides the efficiency issues, a mathematical description and a strong theoretical characterization are needed (and pursued in this paper) to properly place the method within the umbrella of RC methods (generalizing the Echo State Property [32, 37, 50] to tree processing with deep architectures), for the extension of the input domain to structures and to derive general properties and insights. Finally, we will focus on the empirical advantages of DeepTESN in learning tasks, with respect to the shallow counterpart (TESN) and literature results (including comparison with HTMMs and kernel-based approaches) in different conditions and according to both efficiency and predictive performance[1].

The rest of this paper is structured as follows. In Section 2 we recall the main characteristics of the TESN model, also introducing the notation adopted for learning in tree domains. We introduce the novel DeepTESN model in Section 3, describing the deep recursive reservoir architecture and the readout computation, as well as discussing the advantages in terms of computational complexity[2]. In Section 4 we introduce the Tree Echo State Property and we give two conditions, one sufficient and one necessary, for a DeepTESN to satisfy it[3]. The outcomes of our experimental analysis on real-world datasets is reported in Section 5. Finally, in Section 6 we draw the conclusions.

2. Tree Echo State Networks

In our notation, we use $\mathbf{t}$ to denote a tree and $\mathcal{N}(\mathbf{t})$ to refer to the set of nodes of $\mathbf{t}$. We use $n \in \mathcal{N}(\mathbf{t})$ to denote a generic node of $\mathbf{t}$ and $ch_i(n)$ to indicate

[1] A very preliminary version of a limited part of the content of this manuscript has been introduced in a conference paper [18]. However, this manuscript significantly extends the content, the description and the results by adding the mathematical ground, the theoretical analysis and discussion, an extended experimental assessment both in terms of explored configurations and of reported results and analysis.

[2] The DeepESN model is recalled here as a sub-case of DeepTESN for the sake of brevity.

[3] The proofs of the theorems involved by the conditions for the Tree Echo State Property are provided in Appendix A and Appendix B for the sake of conciseness of theoretical results presentation.

the i-th children of node n, with i ranging from 1 to k, where k is the maximum degree of the set of tree structures under consideration. We use the notation $\mathbf{t}(n)$ to indicate the (sub-)tree rooted in n and induced by the descendants of n. Moreover, by considering a recursive definition of trees, we can say that a k-ary tree $\mathbf{t}$ is either the empty tree, indicated as nil, or it is composed by a root node n and the sub-trees rooted in the children of n, indicated as $n(\mathbf{t}(ch_1(n)), \mathbf{t}(ch_2(n)), \ldots, \mathbf{t}(ch_k(n)))$. In such definition, if any of the children of n is absent, the corresponding sub-tree is the empty tree. The height of a tree $\mathbf{t}$ is the maximum length of a path from its root to any of its nodes. To specify that a tree has height h we use the notation $\mathbf{t}_h$. In what follows, we assume to deal with non-positional trees, i.e. trees in which for each node n it is possible to enumerate its children, but their positions cannot be distinguished.

We focus our attention on labeled trees, i.e. trees whose nodes have attached a label (which can be both of numerical or categorical nature). We use $\mathcal{U}$ to indicate the input label space, and $\mathcal{U}^{\#k}$ to denote the set of k-ary input trees with node labels in $\mathcal{U}$. For our purposes, we are interested in learning functions over domains of trees, such as occurring, e.g., in tasks involving the classification of trees or regression over trees. Functions of this kind can be expressed, e.g., as $f : \mathcal{U}^{\#k} \rightarrow \mathcal{Y}$, where $\mathcal{Y}$ denotes the unstructured output domain.

The Tree ESN (TESN) model [15] has been recently proposed as an efficient randomized neural networks methodology [19, 42] for learning in domains of trees. TESN belongs to the class of RecNNs [9, 14, 24, 43], as it is based on the idea of computing functions over trees by exploiting the internal encoding realized by a hidden layer of (typically non-linear) recurrent units, recursively applied to the nodes of the input tree structure. The output component of the network then utilizes such internally developed representations for the final output computation. The TESN model also belongs to the class of RC methods [37, 47], and, as such, it is characterized by the fact that the hidden layer of recurrent units, called *reservoir*, is left untrained after initialization, leaving the need for training only to the output component, called *readout*.

The recursive reservoir component of TESN encodes the input information by computing a state for each node of the input tree. In our notation, we use $\mathbf{x}(n) \in \mathcal{R}$ to indicate the state computed for node n, where $\mathcal{R}$ denotes the reservoir state space of the TESN. In general, in this paper the domains of interest are represented as vectorial spaces, where in particular $\mathcal{U} \subseteq \mathbb{R}^{N_U}$ and $\mathcal{R} \subseteq \mathbb{R}^{N_R}$. Under the architectural viewpoint, this translates into an input layer with N_U units and a reservoir layer with N_R recurrent units. The reservoir architecture of a TESN is graphically illustrated in Figure 1, in its application to a node n of an input tree. As it can be seen, the state for node n, i.e. $\mathbf{x}(n)$, is computed as a function of the input label attached to n, i.e. $\mathbf{u}(n)$, and of the states computed for the children of n, i.e. $\mathbf{x}(ch_1(n)), \mathbf{x}(ch_2(n)), \ldots, \mathbf{x}(ch_k(n))$. More formally, the reservoir of a TESN implements a state transition function

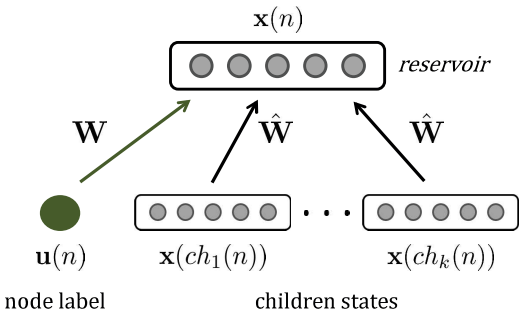


Figure 1: Reservoir architecture of a TESN.

H defined as follows[4]:

$$
\begin{aligned}
H &: \mathcal{U} \times \underbrace{\mathcal{R} \times \ldots \times \mathcal{R}}_{k} \to \mathcal{R}, \\
\mathbf{x}(n) &= H(\mathbf{u}(n), \mathbf{x}(ch_1(n)), \ldots, \mathbf{x}(ch_k(n))) \\
&= \tanh\left(\mathbf{W}\,\mathbf{u}(n) + \textstyle\sum_{i=1}^{k} \hat{\mathbf{W}}\,\mathbf{x}(ch_i(n))\right),
\end{aligned}
\tag{1}
$$

where $\mathbf{W} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix, which expresses the dependence from the input label information, and $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the recursive reservoir weight matrix, which expresses the dependencies from the states of children nodes. In the computation of the state transition function implemented by the reservoir, i.e. in the last row of equation 1, whenever it occurs that a child node of n is absent, then the corresponding state in the summation is set to an initial state vector value. Besides, in the last row of equation 1, we use tanh to indicate the element-wise application of the hyperbolic tangent non-linearity, used as activation function of the reservoir units.

Following the parsimony (weight sharing) assumption typical of RecNNs, the same reservoir architecture illustrated in Figure 1, i.e. the same state transition function described by equation 1, is applied to all the nodes of any given input tree $\mathbf{t} \in \mathcal{U}^{\#k}$. In particular, the recursive definition of the state transition function in equation 1, such that the state for each node n is computed as a function of the states of its children, practically corresponds to a visiting process on the nodes of $\mathbf{t}$ that proceeds in a bottom-up fashion, from the leaves up to the root. The bottom-up encoding process carried out by the reservoir of TESN, illustrated through an example in Figure 2, generalizes the operation of standard reservoirs of ESNs from processing of sequences to processing of tree structures. Figure 2 also highlights the aspect of *weight sharing*, typical of RecNN models,

[4]Here, and in the rest of this paper, bias terms are omitted in equations for the ease of notation.

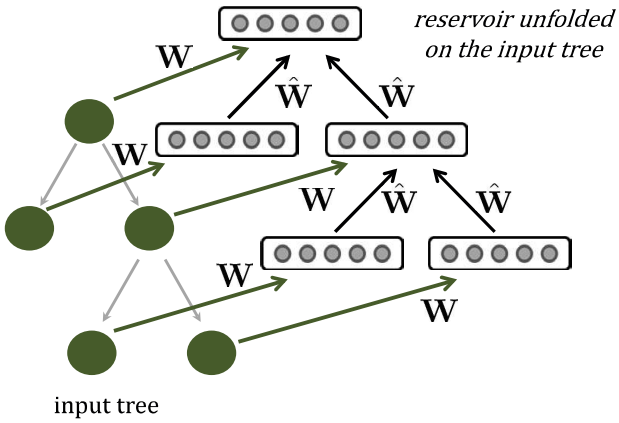


Figure 2: Example of the bottom-up encoding process implemented by the reservoir of a TESN.

when the reservoir architecture is unfolded on the input tree. Note that when the reservoir is applied to a node n, the states that are propagated from deeper nodes encode all the information pertaining to the descendants of n, contextualizing the input label attached to n. In this sense, the reservoir component of a TESN provides a sort of structural memory over the "previous" inputs, and the state computed for each node n gives a representation of the sub-tree rooted in n.

The output of the TESN model is computed by a readout layer, that combines the states computed in the nodes of the input tree for the final output computation. As common in the RC paradigm, the readout is the only part of the network's architecture that undergoes a training process, typically performed by using direct methods such as ridge-regression. The weight values in $\mathbf{W}$ and $\hat{\mathbf{W}}$ are instead left untrained after initialization. For the sake of conciseness, the mathematical descriptions of the readout operation and of the reservoir initialization conditions are postponed to Section 3 and Section 4, respectively, which give a comprehensive description of the DeepTESN methodology where the TESN is treated as a special sub-case within a unified description.

3. Deep Tree Echo State Networks

The DeepTESN model extends the reservoir architecture of TESN, described in Section 2, in the direction of deep neural networks. This is done by introducing a *deep recursive reservoir*, consisting in a stacked composition of multiple recursive hidden layers. In this way, the information associated to each node is processed by a hierarchy of (non-linear) reservoir modules, hence enabling a deep processing of the tree structured information. From the architectural viewpoint, the reservoir of a DeepTESN contains a number of N_L layers that constitute a state encoding pipeline, such that, when applied to any given node

n, the input label attached to n, i.e. $\mathbf{u}(n)$, is propagated only to the first layer. The input to successive layers in the deep reservoir pipeline consists in the output of the previous layer in the stack. The modular architecture of the deep reservoir of DeepTESN is graphically illustrated in Figure 3.

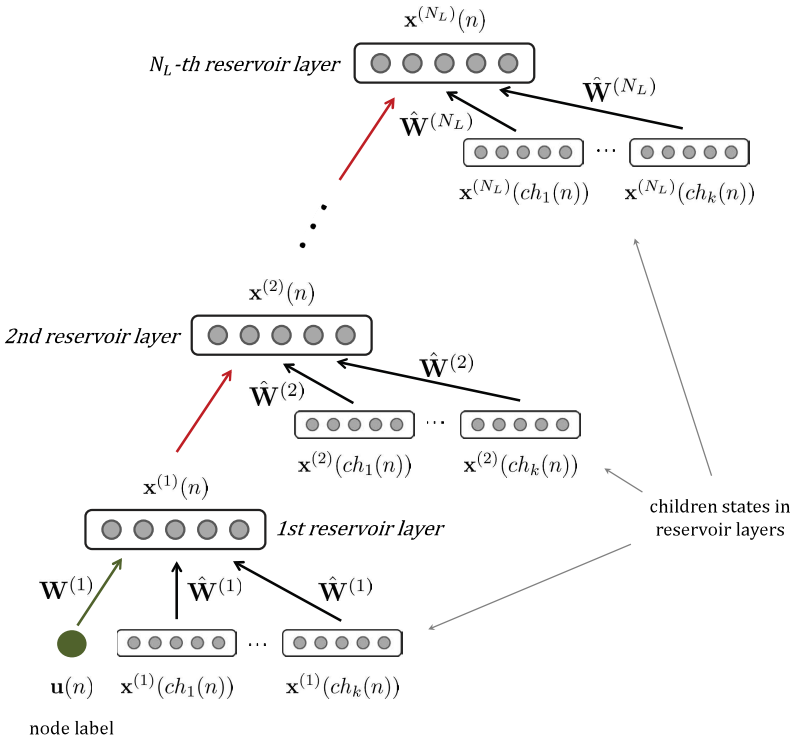


Figure 3: Deep reservoir architecture of a DeepTESN.

In order to develop a mathematical description of the DeepTESN approach, we extend the notation introduced in Section 2 as follows. The state computed for node n by the l-th reservoir layer, for $l = 1, \ldots, N_L$, is denoted by $\mathbf{x}^{(l)}(n)$. Analogously, the reservoir state space at layer l is referred to as $\mathcal{R}^{(l)}$, i.e. $\mathbf{x}^{(l)}(n) \in \mathcal{R}^{(l)}$. Here we use $\mathbf{x}(n) = \left(\mathbf{x}^{(1)}(n), \mathbf{x}^{(2)}(n), \ldots, \mathbf{x}^{(N_L)}(n)\right)$ to indicate the global state of the entire DeepTESN reservoir (i.e. considering all the layers), whereas $\mathcal{R}^*$ denotes the product space of all the reservoir state spaces in the DeepTESN architecture, i.e. $\mathcal{R}^* = \mathcal{R}^{(1)} \times \mathcal{R}^{(2)} \times \ldots \times \mathcal{R}^{(N_L)}$, with $\mathbf{x}(n) \in \mathcal{R}^*$. For the sake of convenience, we also introduce the notation $\mathbf{x}^{(1,\ldots,l)}(n) = \left(\mathbf{x}^{(1)}(n), \mathbf{x}^{(2)}(n), \ldots, \mathbf{x}^{(l)}(n)\right)$ to indicate the state of the DeepTESN aggregated from layer 1 to layer l (with $l \leq N_L$). Moreover, we use $\mathcal{R}^{(1,\ldots,l)}$ to denote the product space of the reservoir state spaces up to layer l (with $l \leq N_L$), i.e. $\mathcal{R}^{(1,\ldots,l)} = \mathcal{R}^{(1)} \times \mathcal{R}^{(2)} \times \ldots \times \mathcal{R}^{(l)}$, with $\mathbf{x}^{(1,\ldots,l)}(n) \in \mathcal{R}^{(1,\ldots,l)}$. In the following, we make the assumption that all the layers in the deep reser-

voir of DeepTESN contain the same number of N_R recursive units, i.e. for all $l = 1, \dots N_L$ we set $\mathcal{R}^{(l)} \subseteq \mathbb{R}^{N_R}$. This assumption is made here for the only sake of easing the model's description, while, of course, the DeepTESN framework generally allows the flexibility of having different reservoir layers with different number of units.

Exploiting the above introduced notation, the function that rules the global state transition computation in DeepTESN can be defined as:

$$\begin{aligned} &F : \mathcal{U} \times \underbrace{\mathcal{R}^* \times \ldots \times \mathcal{R}^*}_{k} \to \mathcal{R}^* \\ &\mathbf{x}(n) = F(\mathbf{u}(n), \mathbf{x}(ch_1(n)), \mathbf{x}(ch_2(n)), \ldots, \mathbf{x}(ch_k(n))). \end{aligned} \tag{2}$$

Considering the hierarchical layer-wise organization of the reservoir in DeepTESN, we can write F also as $F = \big(F^{(1)}, F^{(2)}, \ldots, F^{(N_L)}\big)$, where each $F^{(l)}$ denotes the state transition function of the l-th reservoir layer, for $l = 1, \ldots, N_L$. In particular, for the 1-st layer we have that the state computed for any given node n is a function of the input label attached to n, i.e. $\mathbf{u}(n)$, and of the states computed for the children of n at the same reservoir layer, i.e. $\mathbf{x}^{(1)}(ch_1(n))$, $\mathbf{x}^{(1)}(ch_2(n))$, $\ldots$, $\mathbf{x}^{(1)}(ch_k(n))$. Function $F^{(1)}$ can thus be defined as follows:

$$\begin{aligned} &F^{(1)} : \mathcal{U} \times \underbrace{\mathcal{R}^{(1)} \times \ldots \times \mathcal{R}^{(1)}}_{k} \to \mathcal{R}^{(1)} \\ &\mathbf{x}^{(1)}(n) \;\; = F^{(1)}\big(\mathbf{u}(n), \mathbf{x}^{(1)}(ch_1(n)), \mathbf{x}^{(1)}(ch_2(n)), \ldots, \mathbf{x}^{(1)}(ch_k(n))\big) \\ &\qquad\quad = \tanh\big(\mathbf{W}^{(1)}\,\mathbf{u}(n) + \textstyle\sum_{i=1}^{k} \hat{\mathbf{W}}^{(1)}\,\mathbf{x}^{(1)}(ch_i(n))\big), \end{aligned} \tag{3}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{N_R \times N_U}$ and $\hat{\mathbf{W}}^{(1)} \in \mathbb{R}^{N_R \times N_R}$ respectively denote the input and the recursive reservoir weight matrices for the first reservoir layer. For the second layer, the state computed for node n, i.e. $\mathbf{x}^{(2)}(n)$ depends on the state computed for the same node n in the previous layer, i.e. it is a function of $\mathbf{x}^{(1)}(n) = F^{(1)}\big(\mathbf{u}(n), \mathbf{x}^{(1)}(ch_1(n)), \mathbf{x}^{(1)}(ch_2(n)), \ldots, \mathbf{x}^{(1)}(ch_k(n))\big)$. In addition, the state $\mathbf{x}^{(2)}(n)$ also depends on the states of the children of n computed at layer 2. Accordingly, the state update at layer 2 can be described as $\mathbf{x}^{(2)}(n) = F^{(2)}\big(\mathbf{u}(n), \mathbf{x}^{(1,2)}(ch_1(n)), \mathbf{x}^{(1,2)}(ch_2(n)), \ldots, \mathbf{x}^{(1,2)}(ch_k(n))\big)$. More in general, the state update at layer $l > 1$ of the deep reservoir is ruled by following

state transition function $F^{(l)}$:

$$F^{(l)} : \mathcal{U} \times \underbrace{\mathcal{R}^{(1,\ldots,l)} \times \ldots \times \mathcal{R}^{(1,\ldots,l)}}_{k} \rightarrow \mathcal{R}^{(l)}$$

$$\begin{aligned}
\mathbf{x}^{(l)}(n) &= F^{(l)}\big(\mathbf{u}(n), \mathbf{x}^{(1,\ldots,l)}(ch_1(n)), \mathbf{x}^{(1,\ldots,l)}(ch_2(n)), \ldots, \mathbf{x}^{(1,\ldots,l)}(ch_k(n))\big) \\
&= \tanh\big(\mathbf{W}^{(l)}\,\mathbf{x}^{(l-1)}(n) + \textstyle\sum_{i=1}^{k} \hat{\mathbf{W}}^{(l)}\,\mathbf{x}^{(l)}(ch_i(n))\big) \\
&= \tanh\big(\mathbf{W}^{(l)}\,F^{(l-1)}(\mathbf{u}(n), \mathbf{x}^{(1,\ldots,l-1)}(ch_1(n)), \ldots, \mathbf{x}^{(1,\ldots,l-1)}(ch_k(n))) + \\
&\qquad \textstyle\sum_{i=1}^{k} \hat{\mathbf{W}}^{(l)}\,\mathbf{x}^{(l)}(ch_i(n))\big),
\end{aligned} \tag{4}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ is the weight matrix for the inter-layer connections from layer $l-1$ to layer l, and $\hat{\mathbf{W}}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ is the recursive reservoir weight matrix for layer l. In both equations 3 and 4, whenever it occurs that a child node $ch_i(n)$ is absent, the corresponding state is set to an initial vector value. In other words, if $ch_i(n)$ is absent, for all $l = 1, \ldots, N_L$ we set $\mathbf{x}^{(l)}(ch_i(n)) = \mathbf{x}_0^{(l)}$, where $\mathbf{x}_0 = \big(\mathbf{x}_0^{(1)}, \mathbf{x}_0^{(2)}, \ldots, \mathbf{x}_0^{(N_L)}\big)$ plays the role of initial condition for the reservoir of DeepTESN. In analogy to the case of standard ESNs, null vectors can be used as initial states for all the reservoir layers in the architecture (which corresponds to setting $\mathbf{x}_0^{(l)} = \mathbf{0} \in \mathcal{R}^{(l)}$ for all $l = 1, \ldots, N_L$).

On the architectural side, it is interesting to observe that the hierarchical organization of the hidden recursive reservoir of a deep RecNN/DeepTESN can be regarded as obtained from the one of a fully connected RecNN/TESN through the application of constraints that graphically correspond to layering. For the case of recurrent neural architectures, this aspect is in depth analyzed in [16], to which the reader is referred for further details. It is also worth to note that the architectural "simplification" implied by layering has also a deeper impact on the computational cost of the state computation process, resulting in an advantageous network setting, as discussed in Section 3.2.

As in the case of TESNs, given any input tree $\mathbf{t} \in \mathcal{U}^{\#k}$, the same deep recursive reservoir architecture shown in Figure 3, i.e. the same global state transition function in equation 2, is applied to all the nodes $n \in \mathcal{N}(\mathbf{t})$, resulting in a bottom-up visiting process of $\mathbf{t}$. The process of state computation carried out by the deep recursive reservoir of DeepTESN is graphically illustrated through an example in Figure 4. As in the case of TESN, the state computed for each node n of the input tree provides a representation of the sub-tree rooted in n. Crucially, in the case of DeepTESN, the process of state computation for each node proceeds in a hierarchical fashion. As such, for each node n, the input for higher layers in the reservoir stack already provides structural features computed by means of the operation of lower layers. Overall, the introduction of depth into the recursive reservoir architecture enables a diversification in the representations of the input tree developed in successive layers. The different layers in the recursive reservoir are naturally driven towards developing their own qualitatively different dynamics, hence giving, as a whole, a richer set of

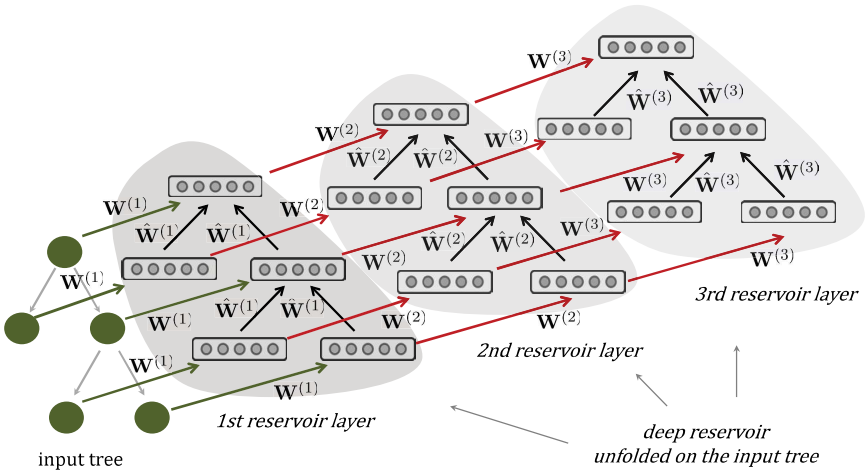


Figure 4: Example of the bottom-up encoding process implemented by the deep recursive reservoir of DeepTESN. A 3-layered reservoir architecture is considered for the sake of graphical illustration.

state features than it would be possible within a shallow recursive counterpart.

Remark 1. *The TESN model [15] described in Section 2 can be seen a particular sub-case of DeepTESN, obtained when the recursive reservoir units are organized into a single layer within a shallow architecture, i.e. when* $N_L = 1$. *In this case, the global DeepTESN reservoir state space* $\mathcal{R}^*$ *reduces to* $\mathcal{R}^{(1)} = \mathcal{R}$, *and the reservoir state transition function* F *in equation 2 reduces to the state transition function* H *for shallow TESN described by equation 1.*

Remark 2. *The DeepESN model, recently introduced in [20, 21], represents a special sub-case of DeepTESN, achieved when the model is applied to sequential/temporal (instead of tree) data processing, i.e. when* $k = 1$. *Specifically, the reservoir computation in DeepESN is described through equations 3 and 4, applying a couple of modifications in the interpretation of the notation. The first is that the node symbol* n *assumes the meaning of time-step index, and the second is that states of children nodes are replaced by the state at the previous time-step, i.e.* $\mathbf{x}^{(l)}(ch_1(n)) = \mathbf{x}^{(l)}(n-1)$.

Remark 3. *The standard ESN model [32, 33] itself can be seen as a special sub-case of DeepTESN, obtained when the reservoir units are organized in a single layer, i.e.* $N_L = 1$, *and when the model is applied to sequential/temporal data, i.e. when* $k = 1$. *Specifically, the reservoir operation of standard ESNs is obtained from that of DeepTESN in the same way as described in Remark 2 for DeepESNs, but limiting the consideration to just one layer in the recurrent architecture.*
Finally, in this regard it is also interesting to observe that the concept of feedback

connection in the recurrent architecture of standard RC networks is generalized in the case of DeepTESN. In this latter case, indeed, the feedback connections in the recursive part of the architecture are linked to the children of the node to which the reservoir is applied (rather than being linked to the input at the previous time-step).

The parameters of the deep recursive reservoir, i.e. the weight values in $\mathbf{W}^{(l)}$ and $\hat{\mathbf{W}}^{(l)}$ for $l = 1, \ldots, N_L$, are left untrained after initialization subject to stability constraints, as common in the RC paradigm. For the case of DeepTESN (and for TESN as a particular sub-case), the reservoir initialization conditions are expressed by a generalization of the Echo State Property (ESP) for standard ESNs [32, 37, 50], as described in detail in Section 4. This essentially translates into an initialization process in which the elements in $\mathbf{W}^{(l)}$ and $\hat{\mathbf{W}}^{(l)}$ are first randomly chosen (e.g. from a uniform distribution) and then re-scaled in order to control their spectral properties. In particular, in this paper we control the 2-norm of matrices $\mathbf{W}^{(l)}$ and the spectral radius of matrices $\hat{\mathbf{W}}^{(l)}$ (see further details in Section 4).

3.1. Readout Computation

As in standard RC models, the output in DeepTESN is computed by means of a trained readout component, which essentially combines the state representations provided by the reservoir part. Specifically, given a input tree $\mathbf{t} \in \mathcal{U}^{\#k}$, the readout computes the network's output $\mathbf{y}(\mathbf{t}) \in \mathcal{Y}$ as a function of the reservoir states computed for the nodes of $\mathbf{t}$, i.e. $\{\mathbf{x}(n) : n \in \mathcal{N}(\mathbf{t})\}$. Here we assume that $\mathcal{Y} \subseteq \mathbb{R}^{N_Y}$, i.e. the readout is featured by N_Y output units, and that the readout operation is described by the following equation:

$$\mathbf{y}(\mathbf{t}) = \mathbf{W}_{out} \ \tanh\big(\sum_{n \in \mathcal{N}(\mathbf{t})} \mathbf{W}_{\varphi} \ \mathbf{x}(n)\big), \tag{5}$$

where $\mathbf{W}_{\varphi} \in \mathbb{R}^{N_{\varphi} \times N_L N_R}$ represents a projection matrix, and $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_{\varphi}}$ is the readout weight matrix. The readout is the only part of the network's architecture that is trained, i.e. only the readout weights in $\mathbf{W}_{out}$ are learned from a training set, typically in closed-form by using direct methods such as ridge-regression, as common in RC practice [37].

In line with the approach of randomized neural networks, the elements in $\mathbf{W}_{\varphi}$ are randomly initialized from a uniform distribution (e.g. over a symmetric interval) and then are re-scaled in order to have a unitary 2-norm. Note that, differently from previous investigations on the TESN approach in [15], the formulation of the readout operation that is given in equation 5 allows a non-linear combination of the DeepTESN reservoir states computed for all the nodes in the input tree. Moreover, although different alternatives for the choice of reservoir-readout connectivity are possible (as described in the context of deep RNNs, e.g., in [30]), here we consider the case in which the readout is fed by all the layers in the deep reservoir. Such a choice enables the readout learner to potentially give a different weight to the structural representations developed by the different levels in the reservoir hierarchy.

3.2. Computational Complexity

The layered architectural construction of the deep recursive reservoir implies a strong saving in terms of reservoir weight connections. Interestingly, this also corresponds to a reduction of the computational complexity, leading to an overall network setup that is even more advantageous than the already efficient TESN approach.

Following the architectural description adopted so far, we consider DeepTESN whose deep recursive reservoir is featured by N_L layers, each of which contains N_R fully connected units. Focusing on the aspect of time complexity for the process of state computation, for each node n of an input tree, the cost of the reservoir operation at the first layer (equation 3) is given by $\mathcal{O}(N_R\, N_U + k\, N_R^2)$. For each successive layer $l > 1$ (equation 4) the cost is $\mathcal{O}(N_R^2 + k\, N_R^2)$. Hence, under the assumption that $N_U < N_R$, the cost of state computation per layer can be expressed as $\mathcal{O}(k\, N_R^2)$, and the cost of the whole state computation process (i.e. considering all the N_L layers) per node is given by $\mathcal{O}(k\, N_L\, N_R^2)$. Overall, for an input tree $\mathbf{t}$, the cost of the state computation process performed by DeepTESN if given by:

$$\mathcal{O}(|\mathcal{N}(\mathbf{t})|\, k\, N_L\, N_R^2), \tag{6}$$

where $|\mathcal{N}(\mathbf{t})|$ is used to denote the number of nodes in the input tree.

In order to provide a comparison with the shallow counterpart, we take into account the case in which the same total number of recursive reservoir units considered in the case of DeepTESN, i.e. $N_L\, N_R$, are arranged into a single-layer TESN architecture. In this case (see Remark 1), the process of state computation has a cost per node that is given by $\mathcal{O}(k(N_L N_R)^2) = \mathcal{O}(kN_L^2 N_R^2)$. The cost per tree is then given by the following equation:

$$\mathcal{O}(|\mathcal{N}(\mathbf{t})|\, k\, N_L^2\, N_R^2). \tag{7}$$

A comparison between equations 6 and 7 reveals that the layered architectural reservoir design of DeepTESN leads to a clear advantage with respect to the shallow TESN case, with a time complexity reduction for the state computation (rate between equations 7 and 6) that increases linearly with the number of layers in which the reservoir units are organized.

Moreover, as already noticed for the case of TESN in [15], the recursive reservoir connections in DeepTESN are left untrained after initialization, and thus the cost of state encoding (equation 6) is the same for both training and test. This common RC feature, already advantageous in comparison to state-of-the-art kernel methods for trees and fully-trained RecNN models in the TESN formulation [15], is here further enhanced in DeepTESN by exploiting a layered RecNN design.

Finally, note that the training cost for DeepTESN is restricted to the cost of adjusting the elements in the $\mathbf{W}_{out}$ matrix in equation 5. Of course, this training cost depends on the specific algorithm that is employed (with SVD-based approaches representing common choices in this regard), and generally it

is lower than the cost of training more complex "readout" implementations, such as Multi-Layer Perceptrons or Support Vector Machines, commonly adopted in trained RecNNs and kernels for trees.

4. Tree Echo State Property

In this section we provide a mathematical background to the study of DeepTESNs dynamics. Framed in the context of RC literature, our analysis allows us to generalize the fundamental ESP and the related elementary conditions [17, 32, 37], extending them to the case of reservoir models that operate over discrete tree structures in a deep fashion. We found such analysis by introducing the Tree Echo State Property (in Definition 1), and by establishing two conditions for it to hold in the case of DeepTESN, one sufficient and one necessary (respectively in Theorems 1 and 2). In light of the characterizations of deep recursive reservoir models discussed in Section 3 (see Remarks 1, 2 and 3), the results of our investigation also apply to TESN, DeepESN and ESN as special cases. Under a dynamical system view-point, our study can be seen as aimed at characterizing the (Lyapunov) stability of DeepTESN state transition systems[5]. Moreover, from a historical perspective, it is also interesting to see that the mathematical analysis presented here is related to (and rooted in) studies in the areas of convergent dynamics (see [41] for a review), input-to-state stability and small-gain theorems (see e.g. [2] and references therein), iterated function systems (see, e.g., [45]) and Markovian architectural bias of RNN architectures (see, e.g., [46]).

For the sake of clarity we recall from Section 3 that the reservoir of DeepTESN implements a hierarchically structured state transition system over discrete tree structures, where the computation takes place by means of a stacked composition of recursive (untrained) hidden layers. Input data for this system hence comes in the form of k-ary rooted labeled trees, where $\mathcal{U}^{\#k}$ is used to specify the set of k-ary input trees under consideration, i.e. those with node labels in the input label space $\mathcal{U}$. While a generic input tree in $\mathcal{U}^{\#k}$ is denoted by $\mathbf{t}$, we use $\mathbf{t}_h$ whenever we also want to specify that the tree has height h. The deep recursive reservoir of DeepTESN is ruled by a global state transition function F (equation 2), which is applied to all the nodes of the input tree in a bottom-up fashion (see Figure 4). In this way, for each node n of an input tree, the reservoir computes a state $\mathbf{x}(n)$ in a global state space denoted by $\mathcal{R}^*$. Whenever we restrict our attention to a specific layer l in the deep reservoir architecture, the state space under consideration is denoted by $\mathcal{R}^{(l)}$. In what follows, we shall assume that the input label space and the reservoir state spaces for all the layers of DeepTESN are compact sets. We shall further assume that the considered reservoir state spaces are metric spaces equipped with the distance induced by the L_2-norm (i.e. the Euclidean distance) for the individual layers' spaces, and

[5] In what follows, unless otherwise stated, the term stability is used to indicate asymptotic stability in the sense of Lyapunov.

with the max-product metric for the involved products of reservoir state spaces (similarly to the case of DeepESN, as described in [17]).

To study the behavior of deep recursive reservoirs we find it useful to introduce an iterated version of the state transition function F in equation 2, denoted by $\hat{F}$ and defined as follows:

$$\hat{F} : \mathcal{U}^{\#k} \times \mathcal{R}^* \to \mathcal{R}^*$$

For every input tree $\mathbf{t} \in \mathcal{U}^{\#k}$ and initial state $\mathbf{x}_0 \in \mathcal{R}^*$:

$$\hat{F}(\mathbf{t}, \mathbf{x}_0) = \begin{cases} \mathbf{x}_0 & \text{if } \mathbf{t} = nil \\ F\big(\mathbf{u}(n), \hat{F}(\mathbf{t}(ch_1(n)), \mathbf{x}_0), \\ \hat{F}(\mathbf{t}(ch_2(n)), \mathbf{x}_0), \ldots, \hat{F}(\mathbf{t}(ch_k(n)), \mathbf{x}_0)\big) & \text{if } \mathbf{t} = n\big(\mathbf{t}(ch_1(n)), \\ & \mathbf{t}(ch_2(n)), \ldots, \mathbf{t}(ch_k(n))\big). \end{cases} \tag{8}$$

In other words, $\hat{F}$ takes in input an input tree $\mathbf{t}$ and an initial state $\mathbf{x}_0$ and returns in output the state computed by the reservoir of DeepTESN for the root node of $\mathbf{t}$, starting from the initial conditions given by $\mathbf{x}_0$.

With this background we can define the Tree Echo State Property (TESP) as follows:

Definition 1 (Tree Echo State Property). *Assume a DeepTESN whose state transition dynamics is ruled by function F as in equation 2, instantiated through layer-wise equations 3 and 4, and whose iterated extension is defined by equation 8. The network fulfills the TESP if for any input tree $\mathbf{t}_h \in \mathcal{U}^{\#k}$ of height h, and for any two initial conditions $\mathbf{x}_0, \mathbf{z}_0 \in \mathcal{R}^*$ it holds that:*

$$\|\hat{F}(\mathbf{t}_h, \mathbf{x}_0) - \hat{F}(\mathbf{t}_h, \mathbf{z}_0)\| \to 0 \text{ as } h \to \infty. \tag{9}$$

The TESP definition in equation 9 essentially says that the process of state encoding performed by the reservoir of a DeepTESN should asymptotically depend only on the (tree-structured) input information, while the influence of the initial conditions should progressively vanish with the height of the input tree[6]. In other words, the effects on the reservoir state due to differences in deeper nodes of the input should progressively become less relevant. This ensures that input differences in certain nodes do not affect the encoding of the whole input structure unboundedly, with similar sub-structures being encoded by similar reservoir states. As such, given the untrained characterization of the reservoir weights, in complete analogy to the case of the ESP for standard ESNs, the TESP can be regarded as a global asymptotic stability property on the network's states developed under the influence of the "driving" input information.

[6]The concept of trees of arbitrary (infinite) height, i.e. considering $h \to \infty$ in equation 9, is used here as a mathematical construction to enable the study of the asymptotic behavior of DeepTESN states. This is in analogy to the role played by sequences of infinite length in the analysis of the ESP for standard ESNs [32].

Remark 4. *It is worth noticing that when the DeepTESN model reduces to a DeepESN (i.e. when it is applied to input of sequential/temporal nature, see Remark 2), the TESP expressed by equation 9 reduces to the ESP for deep RC models, recently introduced in [17]. If, in addition, the reservoir architecture comprises only one layer (see Remark 3), i.e. when $N_L = 1$, then equation 9 reduces to the case of standard ESP for (shallow) ESNs [32, 37].*

The reservoir of a DeepTESN is initialized in order to satisfy the TESP in Definition 1. In the following we give two conditions for the TESP to hold, one sufficient (expressed by Theorem 1) and one necessary (expressed by Theorem 2). Note that we use the notation $\rho(\cdot)$ to indicate the spectral radius operator, returning the maximum among the eigenvalues in modulus of its matrix argument.

Theorem 1 (Sufficient Condition for the Tree Echo State Property). *Assume a DeepTESN whose state transition dynamics is ruled by function F as in equation 2, instantiated through layer-wise equations 3 and 4. Then, a sufficient condition for the TESP to hold is given by the following equation:*

$$\sigma_T = \Big(k \max_{l=1,\dots,N_L} \big(\sum_{i=1,\dots,l} \|\hat{\mathbf{W}}^{(i)}\| \prod_{j=i+1,\dots,l} \|\mathbf{W}^{(j)}\| \big)\Big) < 1. \tag{10}$$

The proof is given in Appendix A.

Theorem 2 (Necessary Condition for the Tree Echo State Property). *Assume a DeepTESN whose state transition dynamics is ruled by function F as in equation 2, instantiated through layer-wise equations 3 and 4. Assume that perfect k-ary trees with zero labels[7] and of arbitrary height[8] represent an admissible input for the system. Then, a necessary condition for the TESP to hold is given by the following equation:*

$$\rho_T = \Big(k \max_{l=1,\dots,N_L} \rho(\hat{\mathbf{W}}^{(l)})\Big) < 1. \tag{11}$$

The proof is given in Appendix B.

Remark 5. *The conditions for the TESP expressed by Theorems 1 and 2 can be applied also to the case of shallow TESN, obtained when the recursive reservoir comprises a single layer, i.e. when $N_L = 1$ (see Remark 1). Moreover, the conditions in Theorems 1 and 2 generalize the corresponding conditions for the ESP of deep and shallow ESNs for sequential/temporal data processing already reported in literature. In particular, when DeepTESN reduces to DeepESN, i.e.*

[7] A tree with zero input labels for each node, in which all the interior nodes have k children, and in which the leaves are all at the same depth.

[8] Note that the assumption on admissible "null" perfect k-ary input trees of arbitrary height generalizes the analogous assumption on admissible null input sequences of infinite length, used in the case of the standard ESP [32].

$k = 1$ and the model is applied to input of sequential/temporal nature (see Remark 2), the sufficient and the necessary conditions given in equations 10 and 11 reduce to the corresponding conditions for the ESP of deep RC models, as recently introduced in [17]. When, in addition, the DeepTESN model reduces to a standard ESN, i.e. $k = 1$ and $N_L = 1$ (see Remark 3), then the conditions for the TESP formulated in equations 10 and 11 reduce to the corresponding conditions for the ESP of standard ESNs, as commonly reported in RC literature [32, 37].

Although an in-depth analysis of contractive and stability properties of DeepTESN states is out of the scopes of this paper, we can draw some insights on their roles on the intrinsic diversification of state representations developed in the different layers of DeepTESN. Specifically, we can note that the left-hand side of equation 10 represents the maximum among the contraction coefficients of the recursive reservoir layers. The contraction coefficient characterizing the state transition function implemented by the l-th layer is given by isolating the corresponding term in the inner maximum operator (see Appendix A for details). We can thus characterize the degree of contractivity of layer l by $\sigma_T^{(l)} = k \left(\sum_{i=1,\dots,l} \|\hat{\mathbf{W}}^{(i)}\| \prod_{j=i+1,\dots,l} \|\mathbf{W}^{(j)}\| \right)$. Equation 10 indicates that the contraction coefficient of the global DeepTESN system is ruled by the highest contraction coefficient among the different reservoir layers. Thereby, if any of the $\sigma_T^{(l)}$ coefficients does not fulfill the contractive property (i.e., if there exists $l = 1, \dots, N_L$ such that $\sigma_T^{(l)} \geq 1$) then it is not possible to guarantee that the TESP holds. Moreover, it is easy to see that simple network settings, e.g. involving a uniform scaling of the reservoir matrices with $\|\mathbf{W}^{(1)}\| = \dots = \|\mathbf{W}^{(N_L)}\| = \omega$, and $\|\hat{\mathbf{W}}^{(1)}\| = \dots = \|\hat{\mathbf{W}}^{(N_L)}\| = \hat{\omega}$, can determine an increasing ordering of the contraction coefficients for increasing layers' level in the architecture, i.e. $\sigma_T^{(1)} \leq \sigma_T^{(2)} \leq \dots \leq \sigma_T^{(N_L)}$. This basically suggests that (under simple reservoir matrices initialization schemes) higher layers in DeepTESN tend to be featured by less contractive dynamics. Analogous considerations can be done in relation to the necessary condition in equation 11. In this case, the degree of stability of the DeepTESN reservoir up to the l-th layer (see Appendix B for details) can be represented in terms of $\rho_T^{(l)} = k \max_{j=1,\dots,l} \rho(\hat{\mathbf{W}}^{(j)})$. This indicates that when we consider a DeepTESN with increasingly more layers, the resulting degree of stability can never decrease as the number of layers increases, and can thus be considered in this sense as a monotonic non-decreasing function of the network's depth.

Overall, these observations provide an interesting insight on the intrinsic role of layering in deep recursive architectures. Essentially, deeper RecNNs naturally tend to be featured by progressively less contractive and less stable behaviors of the state transition functions, leading to an effective diversification of the quality of reservoir state dynamics in the different levels of the deep recursive architecture even in the absence (or before) training of the recursive connections. Finally, note that this insight is also in line with what already observed in the case of deep RC models for sequential/temporal data processing in [17].

The sufficient condition for the TESP expressed by Theorem 1 (equation 10) stems from the study of contractive properties of the state transition functions in the DeepTESN reservoir layers. Essentially, it says that if all the layers in the deep reservoir architecture implement a contraction (i.e. a function that is Lipschitz continuous with constant smaller than 1), then the DeepTESN has the TESP for all the input trees in the considered domain. Analogously, the study of the necessary condition for the TESP, expressed by Theorem 2 (equation 11), is based on the analysis of the stability of the state transition functions implemented by the deep recursive reservoir of DeepTESN. Basically, such condition requires the reservoir dynamics developed at all the layers of the deep reservoir system to be stable under the influence of zero input. Globally, the two conditions for the TESP characterize the behavior of the DeepTESN model by means of the two quantities σ_T (equation 10) and ρ_T (equation 11). The former expresses the degree of contractivity of the DeepTESN global state transition function (larger values of σ_T indicate less a less contractive system), while the latter provides an indicator of how much the system is sensible to initial conditions (when driven by a constant zero input). Although an in-depth analysis of the relation between the sufficient and the necessary conditions for the TESP is not in the scopes of this paper, we can easily see that, as in general $\rho_T \leq \sigma_T$, the sufficient condition in Theorem 1 clearly represents a more restrictive constraint than the necessary one in Theorem 2. In other words, as already observed in standard RC literature [32, 37], the sufficient condition for the TESP in equation 10 is more conservative than the necessary one expressed in equation 10. Moreover, assuming the same simple network setting as considered above, i.e. with $\|\mathbf{W}^{(1)}\| = \ldots = \|\mathbf{W}^{(N_L)}\| = \omega$, and $\|\hat{\mathbf{W}}^{(1)}\| = \ldots = \|\hat{\mathbf{W}}^{(N_L)}\| = \hat{\omega}$ for all $l = 1, \ldots, N_L$, we can go a step further in our analysis and observe that in this case[9] $\sigma_T = k\,\hat{\omega}\,(\omega^{N_L-1} - \omega^{-1})/(1-\omega^{-1})$ $= k\,\hat{\omega}\,(1-\omega^{N_L})/(1-\omega)$. Here we can also note that the quantity $k\,\hat{\omega}$ represents an upper bound of ρ_T. Accordingly, the gap between ρ_T and σ_T increases for increasing values of ω. If, in addition, $\omega > 1$ then we can see that the gap increases exponentially fast with the number of layers N_L.

For application purposes, as common in the ESN literature [32, 37], we shall refer to the necessary condition (equation 11) for scaling the reservoir weight matrices during DeepTESN initialization. Accordingly, the weights in matrices $\hat{\mathbf{W}}^{(l)}$ (for all $l = 1, \ldots, N_L$) are initially randomly chosen from a uniform distribution over a symmetric interval (e.g. $[-1, 1]$), and then are re-scaled in order to control the value of ρ_T in equation 11, after which they can be left untrained. As regards the input and the inter-layer reservoir matrices $\mathbf{W}^{(l)}$, with $l = 1, \ldots, N_L$, these are initialized with weights from uniform distributions over $[-1, 1]$, and then their magnitudes are controlled by scaling their 2-norms to desired factors for the input scaling $scale_{in}$ (for the first layer), and for the inter-layer scaling $scale_{il}^{(l)}$ (for layers $l > 1$).

[9] For simplicity we assume $\omega \neq 1$, while for $\omega = 1$ we have $\sigma_T = k\,\hat{\omega}\,N_L$.

As already pointed out for the case of deep [22] and shallow [32, 37] ESN models in the case of sequence processing, under a practical perspective the TESP strongly depends on the actual input tree to which the reservoir system is applied. As such, in practical cases, the TESP can hold even if $\rho_T \geq 1$. Following this line of reasoning, in our applications of DeepTESN, we slightly relax the constraints of the necessary condition for the TESP as follows. Firstly, the value of k in equation 11 (used for reservoir initialization), i.e. the maximum degree of the set of trees under consideration, is replaced by the average degree of the nodes in the considered set of trees[10]. This is done to avoid excessive shrinking of reservoir weights in cases in which the degree is highly variable among the nodes in the trees. In such cases, indeed, using the value of the maximum degree would penalize the local gain of state transmission for nodes whose degree is much smaller than the maximum, but close to the average. Secondly, we allow exploring also values of ρ_T larger than 1.

5. Experiments

In this section we report the results of our experimental assessment of DeepTESN on three real-world tasks in the area of computational biology and document processing. In order to assess the impact on the predictive performance due to the layered architectural design of recursive networks, we performed our experimental analysis comparatively with single-layered (shallow) TESN settings, under the condition of equal number of total reservoir units. From a more general perspective, the performance obtained by DeepTESN (and TESN) is also compared with state-of-the-art results from literature.

The adopted tasks and the considered experimental settings are described respectively in Sections 5.1 and 5.2. Experimental results are then reported in Section 5.3.

5.1. Tasks

The first two tasks adopted in our experiments are from the area of computational biology. Specifically, we considered two collections of glycan molecules taken from the KEGG/Glycan database [34] and respectively known as Cystic and Leukemia datasets. Glycan molecules have complex structures that can be represented as trees, where mono-saccharides are represented as nodes and sugar bonds are represented as edges between nodes. The input label attached to each node is a one-hot encoding representation of the corresponding mono-saccharide description, resulting in an input label space of size $N_U = 29$ for the Cystic dataset, and of size $N_U = 57$ for the Leukemia dataset. Two binary classification tasks on trees are defined on the basis of the Cystic and Leukemia datasets, by assigning a target value $+1$ to glycans that are annotated as cystic fibrosis or leukemic (in the two cases, respectively), and a target value -1

[10]Only inner nodes are considered in the average, in order to preserve the generalization of the standard ESN approach with $k = 1$.

to all other molecules. As such, in our experiments on the glycans tasks we used 1-dimensional output spaces, i.e. $N_Y = 1$. The Cystic and the Leukemia datasets respectively contain 160 and 442 molecules, that are split according to a stratified 10-folds cross validation scheme. The performance on both the glycans tasks is computed in terms of area under the curve (AUC) values.

The third task that we took into consideration is the INEX 2006 task, a challenging multi-classification task in the area of document processing coming from the 2006 INEX competition [12]. In particular, in our experiments we referred to the IEEE structure only corpus, which comprises XML-formatted documents belonging to 18 different journals. Each XML document is represented by a tree that is built following its hierarchical XML organization, where the input label attached to each node is a 65-dimensional one-hot vector encoding the corresponding XML tag. For each document, the target is an 18-dimensional vector representing a one-hot encoding of the corresponding journal, where all the entries are set to a -1 value, except the one that identifies the correct document classification, which is set to a value of 1. In our experiments on the INEX 2006 task, we used input label and output spaces for DeepTESN with $N_U = 65$ and $N_Y = 18$. Moreover, for each N_Y-dimensional output vector, the corresponding class label was assigned as the index of the output unit with the highest activation. The dataset comprises a number of 12107 documents, 6053 of which are used for training, leaving the remaining 6054 for test, according to a rough $50\% - 50\%$ split [12]. Results on the INEX 2006 task are assessed in terms of 18-class accuracy.

5.2. Experimental Settings

In order to demonstrate the potentiality of the proposed DeepTESN model in real-world applications, we adopted an architectural setup in which the reservoir hyper-parameters assume the same values in all the network's layers. Accordingly, we scaled the spectral radius of all the recursive reservoir weight matrices to the same value, i.e. to $\rho = \rho(\hat{\mathbf{W}}^{(l)})$ for all $l = 1, \dots, N_L$. Moreover, we used the same inter-layer scaling for all layers, i.e. $scale_{il} = scale_{il}^{(l)}$ for all $l > 1$, We used DeepTESN networks with a number of $N_L = 10$ layers, each of which containing the same number of N_R reservoir units. Note that this design choice is made for the only sake of simplicity in the experimental setting, while the DeepTESN approach clearly allows for a higher flexibility in the choice of the hyper-parameters and reservoir dimensions in the different layers of the architecture.

For all the tasks, the values of the reservoir hyper-parameters were chosen by model selection, using a random search approach with 20 random reservoir configurations. The value for the spectral radius ρ was sampled from a uniform distribution over $(0, 1.5]$, while the values $scale_{in}$ and $scale_{il}$ were sampled from a uniform distribution over $(1, 5]$. For each reservoir configuration the results were averaged over 10 network guesses for the INEX 2006 task and over 20 network guesses for the Cystic and the Leukemia tasks. Such guesses were independently generated with the same reservoir hyper-parameters, but using

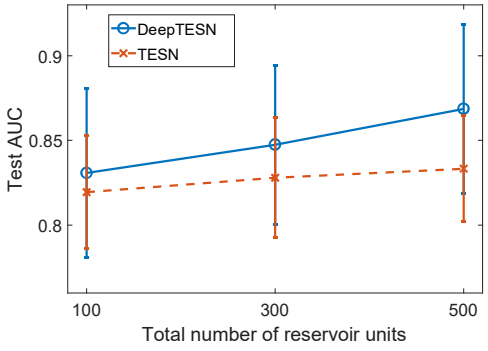
different random seeds for the initialization. As concerns the readout training, for each reservoir configuration we used ridge regression with a regularization coefficient λ_r chosen by using a grid search over $\{10^{-i} : i = -5, -4, \ldots, 4, 5\}$. The model selection scheme adopted in this paper followed the main literature choices for the three tasks. In particular, for the Cystic and the Leukemia tasks, we used a double cross-fold validation approach, such that for each of the 10 folds in the external level of cross-validation, the hyper-parameters' values were selected based on a nested level of 3-fold cross-validation. Moreover, on these tasks, we also computed the optimal performance on the 10-fold cross-validation, as common in literature works in this case. For the INEX 2006 task we used a hold-out cross-validation scheme in which the training samples were split into an inner level of training and validation sets, containing respectively 4237 and 1816 samples. In consideration of the different characteristics and number of samples in the datasets, we used different values of the reservoirs size N_R for the three tasks. For the Cystic and the Leukemia tasks we used values of N_R in the range 10-50, while for the INEX 2006 task we explored values of N_R in the range 100-300. In all cases, for the readout projection matrix we used a value of N_φ equal to the double of the total number of reservoir units, i.e. $N_\varphi = 2\,N_L\,N_R$.

In order to assess the actual impact of layering on the results, we also ran experiments on the same tasks using TESN, adopting the same experimental settings described above (ranges of hyper-parameters values and model selection approaches). In this case, we considered single-layered (shallow) recursive reservoirs with the same total number of units as used in the DeepTESN experiments, i.e. in the range 100-500 for Cystic and Leukemia, and in the range 1000-3000 for INEX 2006. Note that this choice enabled a direct comparison between the cases of DeepTESN and TESN, leading to readouts settings with the same total number of output weights, i.e. in a condition of fairness on the number of free-parameters of the learner.

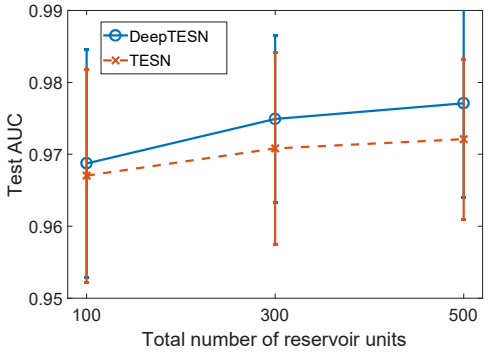
5.3. Results

The test performance obtained by DeepTESN and TESN for increasing number of total reservoir units is shown in Figures 5(a), 5(b) and 5(c), respectively for the Cystic, Leukemia and INEX 2006 tasks. Results in Figure 5 collectively show that DeepTESNs systematically perform better than TESNs on all the tasks, with a performance advantage (i.e. a gap between the performance of DeepTESN and TESN) that tends to increase as larger reservoirs are considered. We could verify that the performance improvement of DeepTESN with respect of TESN is statistically significant with a two-sided Wilcoxon signed rank test ($p < 0.05$) for a total number of reservoir units > 100 for the Cystic and Leukemia tasks, and for all the considered reservoir sizes in the case of INEX 2006. Moreover, as regards the qualitative trends at the increase of the total reservoir dimension, we can also see from Figure 5 that while in the case of TESN it can be observed a saturation effect, a better scaling behavior can be appreciated in the case of DeepTESN.

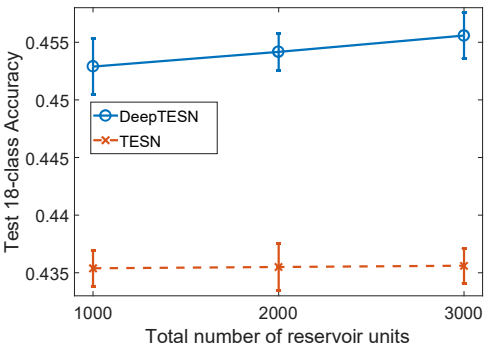
The test set performances obtained by DeepTESN and TESN in correspondence of the largest reservoir settings considered are reported in Table 1 for the



(a) Cystic.



(b) Leukemia.



(c) INEX 2006.

Figure 5: Test set performance (the higher the better) obtained by DeepTESN and TESN on the considered tasks, for increasing total number of recursive reservoir units (vertical bars indicate standard deviation among reservoir guesses). Note that reservoir units in DeepTESN are organized in 10 layers.

Cystic and Leukemia tasks, and in Table 2 for the INEX 2006 task. Details on the selected hyper-parameters are given in Appendix C. For the sake of comparison with the state-of-the-art, in the Tables 1 and 2 we also reported literature results that correspond to the learning models that, at the best of our knowledge, achieved the highest performances in the respective cases. To this aim, we took into consideration a wide range of kernel models including the sub-tree (ST) [48], the subset tree (SST) [11], the Partial Tree (PT) [40], and Subpath [35] kernels. Another group of considered learning models comes from the class of activation mask kernels for recursive models, based on generative topographic mappings, as for the AM-GTM [5] kernel, or on self-organizing maps for structures, in the cases of the AM-SOM and AMπ-SOM [1] kernels. This latter approach has been also investigated in combination with the structural encoding provided by TESN, yielding the KTESN model, described in [4]. Finally, an approach that has been recently developed within the class of HTMMs is given by the generative tree kernel (GTK) family, introduced in [5].

Results in Tables 1 and 2 and indicate that, while the TESN model already reaches top-rank performances in comparison to literature approaches, the proposed DeepTESN model is able to outperform the state-of-the-art results in all the considered tasks. Moreover, a further performance boost is achieved by considering a simple ensemble of DeepTESN classifiers. Table 3 reports the performance achieved by the ensemble models obtained by averaging the output of the networks guesses selected through the model selection for each task, individually computed for both DeepTESN and TESN. Results in Table 3 show that such a simple ensemble strategy is able to improve the performance of DeepTESN even further, with a favorable comparison with respect to TESN also in this setting.

Model	*AUC on Cystic*	*AUC on Leukemia*
DeepTESN	**0.869** (±**0.050**)	**0.977** (±**0.013**)
max	*0.916 (± 0.039)*	*0.982 (± 0.010)*
TESN	0.833 (±0.031)	0.972 (±0.011)
max	*0.910 (± 0.019)*	*0.981 (± 0.008)*
Subpath	0.850 –	0.971 –
GTK	0.826 (±0.104)	0.971 (±0.028)
PT	0.823 –	0.967 –
ST	0.798 –	0.961 –
SST	0.696 –	0.933 –

Table 1: Test AUC (and std on the reservoir guesses) achieved by DeepTESN and TESN on the Cystic and Leukemia tasks. For DeepTESN and TESN, we report the results achieved by the models selected through double cross-fold validation, as well as those corresponding to the best setting on the external 10-fold cross-validation (indicated in the corresponding *max* rows). State-of-the-art results from literature are also reported (with std indicated when available). Best results are highlighted in bold font.

Finally, for the analysis of the computational cost, Table 4 reports the times needed for the operations of training and test DeepTESN and TESN on the considered tasks, in correspondence of the selected configurations with the largest

Model	*Accuracy on INEX 2006*
DeepTESN	**0.4556** (±**0.002**)
TESN	0.4350 (±0.001)
GTK	0.4506 (±0.002)
AM-GTM	0.4371 –
KTESN	0.4341 (±0.001)
PT	0.4113 –
SST	0.4041 –
AMπ-SOM	0.4033 –
AM-SOM	0.4007 –
Subpath	0.3988 –
ST	0.3202 –

Table 2: Test 18-class accuracy (and std on the reservoir guesses) achieved by DeepTESN and TESN on the INEX 2006 task. State-of-the-art results from literature are also reported (with std indicated when available). Best results are highlighted in bold font.

Model	*Cystic*	*Leukemia*	*INEX 2006*
DeepTESN	0.887	0.986	0.4564
TESN	0.843	0.974	0.4349

Table 3: Test performance achieved by the ensemble of DeepTESNs and TESNs on the considered tasks. Performances are expressed in terms of AUC for Cystic and Leukemia, and in terms of 18-class accuracy for INEX 2006.

reservoirs. Times in Table 4 were averaged over the considered reservoir guesses and, for the Cystic and Leukemia tasks, also on the external folds of the cross-validation. Times are referred to experiments with MATLAB implementations of DeepTESNs and TESNs, run on an system equipped with Intel Xeon Processor E5-2630L v3 and with 128 Gb of RAM. To provide unbiased estimation of the computational times, the values reported in Table 4 were obtained in single-core mode, without using any high-performance computing support for GPU or multi-core computing. The most relevant aspect emerging from the results in Table 4 is the striking advantage in terms of computational efficiency of the proposed DeepTESN approach. Compared to the already efficient TESN model, DeepTESN leads to a strong reduction of the required times for both training and test. Interestingly, this observation nicely matches the outcomes of the mathematical analysis of computational complexity reported in Section 3.2. Such mathematical analysis indeed pointed out that the cost of state computation in DeepTESN is asymptotically reduced (i.e the efficiency is boosted), with respect to the TESN case, by a factor that increases with the number of layers in which the same number of reservoir units are organized. This mathematical characterization is actually reflected by the results in Table 4, which show that, in particular for the case of INEX 2006 task for which the largest reservoirs were considered, DeepTESN with 10 layers is ≈ 10 times faster than the shallow TESN counterpart (under the same experimental settings).

Model	*Cystic*		*Leukemia*		*INEX 2006*	
	TR	TS	TR	TS	TR	TS
DeepTESN	0.38”	0.03”	1.47”	0.15”	3.65’	3.56’
TESN	1.54”	0.16”	7.29”	0.80”	39.75’	40.31’

Table 4: Time needed for training and test on the considered tasks by DeepTESN and TESN.

6. Conclusions

The DeepTESN model has been introduced to concretely show how is it possible to conjugate deep learning and adaptive processing of structured data by a randomized neural networks approach, within the class of RC methodologies. To this aim, a solid theoretical ground has been developed to study the conditions for the initialization of deep reservoirs applied to trees, which also gave some insight on the properties of the model.

In particular, our analysis allowed us to generalize the fundamental characterization of ESN dynamics from the standard case of sequential/temporal data to the case of tree structured information, introducing the so called Tree Echo State Property. We have provided two conditions for such property, one sufficient and one necessary, rooted respectively in the study of contractive properties and sensitivity to initial conditions of the DeepTESN state transition functions. Besides giving a mathematically grounded and practical approach to the initialization of untrained recursive deep reservoirs for trees, our mathematical analysis also gave interesting insights on the natural diversification of recursive dynamics developed in successive layers of stacked RecNNs. Essentially, higher layers in a deep recursive architecture (i.e. those more distant from the input) tend to be featured by progressively less contractive and less stable behaviors, hence developing a rich pool of internal representations that the readout component can fruitfully combine at training stage. These aspects, in summary, contribute to support the potentiality to combine the extension of the input domain to trees by a multi-layered (deep) recursive architecture.

The introduced DeepTESN model also showed relevant practical advantages, namely for from the viewpoints of efficiency and predictive performance in comparison to the shallow counterpart. Our experimental analysis showed that such efficiency advantage is not paid in terms of accuracy, as the predictive performance is competitive with respect to comparable learning models. In particular, in comparison to literature approaches, DeepTESN establishes new state-of-the-art results for all the considered tasks, i.e. Cystic and Leukemia (in the domain of computational biology), and INEX 2006 (in the document processing area).

Looking to future developments, some insights can be hypothesized from the analysis provided in this paper, looking at the model's behavior with respect to the scaling of the number of hidden recursive units. In particular, the factor of regularization has an effect that apparently matches well the hierarchical domain and the tasks at hand. Another factor is related to the capability of the models to treat multiple views of the contextual information over trees, which is enabled through the explicit layering of the recursive units of the reservoir. Under

a theoretical perspective, the mathematical analysis introduced in relation to the Tree Echo State Property paves the way to important advancements on the characterization of the behavior of deep recursive neural architectures. Instances in this sense include the study of system stability in presence of driving input and the analysis of the approximation capabilities of the model. In particular, as pertains to the latter aspect, a major research line involves the analysis of the initialization scope of the untrained internal weights, which has been shown to play a very important role in the construction of powerful neural network models within the randomized framework [49]. Besides, a further interesting aspect of model development is toward the extension of the class of data domains, e.g. by pursuing the research line of contextual processing of structured data [39] and thus extending the computational capabilities of the approach to directed acyclic graphs as expressed in [28].

The final aim addressed by the introduction of the DeepTESN model is to stimulate further analyses of the characteristics of deep models for efficiently learning in structured domains, and at the same time to concretely provide a very efficient yet effective model, in the class of randomized neural networks and with a solid basis, to cope with the advancements in application areas characterized by large-scale and complex (structured) data.

References

[1] F. Aiolli, G. Da San Martino, and A. Sperduti. A New Tree Kernel Based on SOM-SD. *Artificial Neural Networks–ICANN 2010*, pages 49–58, 2010.

[2] B.D.O Anderson. The small-gain theorem, the passivity theorem and their equivalence. *Journal of the Franklin Institute*, 293(2):105 – 115, 1972. ISSN 0016-0032. doi: https://doi.org/10.1016/0016-0032(72)90150-0.

[3] D. Bacciu, A. Micheli, and A. Sperduti. Compositional generative mapping for tree-structured data - part i: Bottom-up probabilistic modeling of trees. *IEEE Transactions on Neural Networks and Learning Systems*, 23 (12):1987–2002, 2012.

[4] D. Bacciu, C. Gallicchio, and A. Micheli. A reservoir activation kernel for trees. In *Proceedings of the 24th European Symposium on Artificial Neural Networks (ESANN)*, pages 29–34. i6doc.com, 2016.

[5] D. Bacciu, A. Micheli, and A. Sperduti. Generative kernels for tree-structured data. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4932–4946, 2018. doi: 10.1109/TNNLS.2017.2785292.

[6] P. Baldi and G. Pollastri. The Principled Design of Large-Scale Recursive Neural Network Architectures–DAG-RNNs and the Protein Structure Prediction Problem. *Journal of Machine Learning Research*, 4(Sep):575–602, 2003.

[7] L. Bernazzani, C. Duce, A. Micheli, V. Mollica, A. Sperduti, A. Starita, and M.R. Tiné. Predicting physical- chemical properties of compounds from molecular structures by recursive neural networks. *Journal of Chemical Information and Modeling*, 46(5):2030–2042, 2006.

[8] M. Bianchini, M. Gori, and F. Scarselli. Processing directed acyclic graphs with recursive neural networks. *IEEE Transactions on Neural Networks*, 12(6):1464–1470, 2001.

[9] A.M. Bianucci, A. Micheli, A. Sperduti, and A. Starita. Application of cascade correlation networks for structures to chemistry. *Applied Intelligence*, 12(1):117–147, 2000.

[10] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[11] M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics, 2002.

[12] L. Denoyer and P. Gallinari. Report on the xml mining track at inex 2005 and inex 2006: categorization and clustering of xml documents. In *ACM SIGIR Forum*, volume 41, pages 79–90. ACM, 2007.

[13] M. Diligenti, P. Frasconi, and M. Gori. Hidden tree markov models for document image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):519–523, 2003.

[14] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9 (5):768–786, 1998.

[15] C. Gallicchio and A. Micheli. Tree Echo State Networks. *Neurocomputing*, 101:319–337, 2013.

[16] C. Gallicchio and A. Micheli. Deep Echo State Network (DeepESN): A brief survey. *arXiv preprint arXiv:1712.04323*, 2017.

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

[18] C. Gallicchio and A. Micheli. Deep Tree Echo State Networks. In *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)*, pages 499–506. IEEE, 2018.

[19] C. Gallicchio, J. D. Martin-Guerrero, A. Micheli, and E. Soria-Olivas. Randomized Machine Learning Approaches: Recent Developments and Challenges. In *Proceedings of the 25th European Symposium on Artificial Neural Networks (ESANN)*, pages 77–86. i6doc.com, 2017.

[20] C. Gallicchio, A. Micheli, and L. Pedrelli. Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, 268:87–99, 2017. ISSN 0925-2312.

[21] C. Gallicchio, A. Micheli, and L. Pedrelli. Design of Deep Echo State Networks. *Neural Networks*, 108:33–47, 2018.

[22] C. Gallicchio, A. Micheli, and L. Silvestri. Local Lyapunov exponents of Deep Echo State Networks. *Neurocomputing*, 298:34–45, 2018.

[23] N. Gianniotis and P. Tiňo. Visualization of tree-structured data through generative topographic mapping. *IEEE Transactions on Neural Networks*, 19(8):1468–1493, 2008.

[24] C. Goller and A. Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN)*, volume 1, pages 347–352, 1996.

[25] M. Hagenbuchner, A. Sperduti, and A.C. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3):491–505, 2003.

[26] B. Hammer. *Learning with recurrent neural networks*, volume 254. Springer, 2007.

[27] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert. Recursive self-organizing network models. *Neural Networks*, 17(8):1061–1085, 2004.

[28] B. Hammer, A. Micheli, and A. Sperduti. Universal approximation capability of cascade correlation for structures. *Neural Computation*, 17(5): 1109–1159, 2005.

[29] B. Hammer, A. Micheli, and A. Sperduti. Adaptive contextual processing of structured data by recursive neural networks: A survey of computational properties. *Perspectives of Neural-Symbolic Integration*, 77:67–94, 2007.

[30] M. Hermans and B. Schrauwen. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 190–198, 2013.

[31] O. Irsoy and C. Cardie. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2096–2104, 2014.

[32] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks - with an erratum note. Technical report, GMD - German National Research Institute for Computer Science, 2001.

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

[34] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The kegg resource for deciphering the genome. *Nucleic acids research*, 32(suppl_1): D277–D280, 2004.

[35] D. Kimura, T. Kuboyama, T. Shibuya, and H. Kashima. A subpath kernel for rooted unordered trees. *Advances in Knowledge Discovery and Data Mining*, pages 62–74, 2011.

[36] M. Li and D. Wang. Insights into randomized algorithms for neural networks: Practical issues and common pitfalls. *Information Sciences*, 382: 170–178, 2017.

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

[38] A. Micheli. *Recursive processing of structured domains in machine learning.* PhD thesis, Dept. of Comput. Sci., Univ. of Pisa, Pisa, Italy, 2003. TD-13/03.

[39] A. Micheli, D. Sona, and A. Sperduti. Contextual processing of structured data by recursive cascade correlation. *IEEE Transactions on Neural Networks*, 15(6):1396–1410, 2004.

[40] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European conference on Machine Learning (ECML)*, pages 318–329. Springer, 2006.

[41] A. Pavlov, A. Pogromsky, N. van de Wouw, and H. Nijmeijer. Convergent dynamics, a tribute to boris pavlovich demidovich. *Systems & Control Letters*, 52(3-4):257–261, 2004.

[42] S. Scardapane and D. Wang. Randomness in neural networks: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2), 2017.

[43] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[44] K.S. Tai, R. Socher, and C.D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, page 1556–1566. Association for Computational Linguistics, 2015.

[45] P. Tiňo, G. Dorffner, and C. Schittenkopf. Understanding state space organization in recurrent neural networks with iterative function systems dynamics. In S. Wermter and R. Sun, editors, *Hybrid Neural Systems, Lecture Notes in Computer Science*, volume 1778, pages 255–269. Springer, 2000.

[46] P. Tiňo, M. Cernansky, and L. Benuskova. Markovian architectural bias of recurrent neural networks. *IEEE Transactions on Neural Networks*, 15(1): 6–15, 2004.

[47] D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20 (3):391–403, 2007.

[48] S.V.N. Vishwanathan and A.J. Smola. Fast kernels for string and tree matching. In *Advances in Neural Information Processing Systems (NIPS)*, pages 585–592. MIT Press, 2003.

[49] D. Wang and M. Li. Stochastic configuration networks: Fundamentals and algorithms. *IEEE Transactions on Cybernetics*, 47(10):3466–3479, 2017.

[50] I.B. Yildiz, H. Jaeger, and S.J. Kiebel. Re-visiting the echo state property. *Neural Networks*, 35:1–9, 2012.

Appendix A. Proof of Theorem 1

In order to develop a sufficient condition for the TESP we focus on the analysis of contractive properties of DeepTESN state transition functions. For the sake of convenience, here we recall that we are assuming that input and reservoir state spaces are compact metric spaces, where individual layers' spaces are equipped with the (L_2-norm induced) Euclidean distance, and the considered product spaces are endowed with the resulting max-product metric (as in the case of DeepESN, described in [17], to which the reader is referred for further information). Accordingly, the distance between two any global DeepTESN states $\mathbf{x}, \mathbf{z} \in \mathcal{R}^*$, where $\mathbf{x} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N_L)})$ and $\mathbf{z} = (\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N_L)})$, is given by $\|\mathbf{x} - \mathbf{z}\| = \max_{l=1,\dots,N_L} \|\mathbf{x}^{(l)} - \mathbf{z}^{(l)}\|$. Moreover, for any couple of sets of k global DeepTESN states (occurring, e.g., in the definitions of state transition functions in equations 3 and 4) $\mathbf{x}_1, \dots, \mathbf{x}_k$, $\mathbf{z}_1, \dots, \mathbf{z}_k \in \mathcal{R}^*$, we have that $\|(\mathbf{x}_1, \dots, \mathbf{x}_k) - (\mathbf{z}_1, \dots, \mathbf{z}_k)\| = \max_{i=1,\dots,k} \|\mathbf{x}_i - \mathbf{z}_i\|$.

With this background, we say that the global state transition function F of DeepTESN (equation 2) implements a contraction if it is Lipschitz continuous with Lipschitz constant smaller than 1. In formulas, F is contractive if there exists a real constant σ_T, with $0 \leq \sigma_T < 1$, such that for any $\mathbf{u} \in \mathcal{U}$ and $\mathbf{x}_1, \dots, \mathbf{x}_k, \mathbf{z}_1, \dots, \mathbf{z}_k \in \mathcal{R}^*$, it results that:

$$\|F(\mathbf{u}, \mathbf{x}_1, \dots, \mathbf{x}_k) - F(\mathbf{u}, \mathbf{z}_1, \dots, \mathbf{z}_k)\| \leq \sigma_T \|(\mathbf{x}_1, \dots, \mathbf{x}_k) - (\mathbf{z}_1, \dots, \mathbf{z}_k)\|, \quad \text{(A.1)}$$

where the Lipschitz constant σ_T represents the degree of contractivity of the global state transition function. Analogously, we say the l-th reservoir layer in the DeepTESN (with $l = 1, \dots, N_L$) implements a contraction if there exists a real constant $\sigma_T^{(l)}$, with $0 \leq \sigma_T^{(l)} < 1$, such that for any $\mathbf{u} \in \mathcal{U}$ and

$\mathbf{x}_1^{(1,\dots,l)}, \dots, \mathbf{x}_k^{(1,\dots,l)}, \mathbf{z}_1^{(1,\dots,l)}, \dots, \mathbf{z}_k^{(1,\dots,l)} \in \mathcal{R}^{(1,\dots,l)}$, it results that:

$$\begin{aligned} \|F(\mathbf{u}, \mathbf{x}_1^{(1,\dots,l)}, \dots, \mathbf{x}_k^{(1,\dots,l)}) - F(\mathbf{u}, \mathbf{z}_1^{(1,\dots,l)}, \dots, \mathbf{z}_k^{(1,\dots,l)})\| \leq \\ \sigma_T^{(l)} \|(\mathbf{x}_1^{(1,\dots,l)}, \dots, \mathbf{x}_k^{(1,\dots,l)}) - (\mathbf{z}_1^{(1,\dots,l)}, \dots, \mathbf{z}_k^{(1,\dots,l)})\|, \end{aligned} \tag{A.2}$$

with the Lipschitz constant $\sigma_T^{(l)}$ playing the role of degree of contractivity of the state transition function of the l-th DeepTESN layer.

We start by showing that if the global state transition function F implements a contraction then the DeepTESN satisfies the TESP. Considering any input tree of height h, denoted as $\mathbf{t}_h$ and defined as $n(\mathbf{t}(ch_1(n)), \mathbf{t}(ch_2(n)), \dots, \mathbf{t}(ch_k(n)))$, two any initial DeepTESN states $\mathbf{x}_0$, $\mathbf{z}_0 \in \mathcal{R}^*$, and assuming that equation A.1 holds true, we can provide an upper bound to $\|\hat{F}(\mathbf{t}_h, \mathbf{x}_0) - \hat{F}(\mathbf{t}_h, \mathbf{z}_0)\|$, as follows:

$$\begin{aligned} &\|\hat{F}(\mathbf{t}_h, \mathbf{x}_0) - \hat{F}(\mathbf{t}_h, \mathbf{z}_0)\| = \\ &\\ &\|F\big(\mathbf{t}_h, \hat{F}(\mathbf{t}(ch_1(n)), \mathbf{x}_0), \dots, \hat{F}(\mathbf{t}(ch_k(n)), \mathbf{x}_0)\big) - \\ &\quad F\big(\mathbf{t}_h, \hat{F}(\mathbf{t}(ch_1(n)), \mathbf{z}_0), \dots, \hat{F}(\mathbf{t}(ch_k(n)), \mathbf{z}_0)\big)\| \leq \\ &\sigma_T \max_{i=1\dots,k} \|\hat{F}(\mathbf{t}(ch_i(n)), \mathbf{x}_0) - \hat{F}(\mathbf{t}(ch_i(n)), \mathbf{z}_0)\| \leq \\ &\dots \\ &\sigma_T^h \|\mathbf{x}_0 - \mathbf{z}_0\| \leq \sigma_T^h \, d, \end{aligned} \tag{A.3}$$

where d denotes the diameter of $\mathcal{R}^*$. As σ_T is smaller than 1 (by assumption of contractivity) from the last line of equation A.3 it is straightforward to see that as $\|\hat{F}(\mathbf{t}_h, \mathbf{x}_0) - \hat{F}(\mathbf{t}_h, \mathbf{z}_0)\|$ converges to 0 as h goes to ∞, and thus the TESP in equation 9 is satisfied.

Our next step consists in showing that if all the reservoir layers implement a contractive state transition function, then the resulting global dynamics of the DeepTESN are contractive as well. Assuming that each layer l is featured by a contractive state transition function $F^{(l)}$ with degree of contractivity $\sigma_T^{(l)} < 1$, we can show that F is a contraction with degree of contractivity $\sigma_T = \max_{l=1,\dots,N_L} \sigma_T^{(l)}$. Indeed, for any $\mathbf{u} \in \mathcal{U}$, and $\mathbf{x}_1, \dots, \mathbf{x}_k$, $\mathbf{z}_1, \dots, \mathbf{z}_k \in \mathcal{R}^*$,

we have that:

$$
\begin{aligned}
&\|F(\mathbf{u},\mathbf{x}_1,\ldots,\mathbf{x}_k) - F(\mathbf{u},\mathbf{z}_1,\ldots,\mathbf{z}_k)\| = \\
&\|\big(F^{(1)}(\mathbf{u},\mathbf{x}_1^{(1)},\ldots,\mathbf{x}_k^{(1)}),\ldots,F^{(N_L)}(\mathbf{u},\mathbf{x}_1^{(1,\ldots,N_L)},\ldots,\mathbf{x}_k^{(1,\ldots,N_L)})\big) - \\
&\quad\big(F^{(1)}(\mathbf{u},\mathbf{z}_1^{(1)},\ldots,\mathbf{z}_k^{(1)}),\ldots,F^{(N_L)}(\mathbf{u},\mathbf{z}_1^{(1,\ldots,N_L)},\ldots,\mathbf{z}_k^{(1,\ldots,N_L)})\big)\| = \\
&\max_{l=1,\ldots,N_L} \|F^{(l)}(\mathbf{u},\mathbf{x}_1^{(1,\ldots,l)},\ldots,\mathbf{x}_k^{(1,\ldots,l)}) - F^{(l)}(\mathbf{u},\mathbf{z}_1^{(1,\ldots,l)},\ldots,\mathbf{z}_k^{(1,\ldots,l)})\| \leq \\
&\max_{l=1,\ldots,N_L} \big(\sigma_T^{(l)}\|(\mathbf{x}_1^{(1,\ldots,l)},\ldots,\mathbf{x}_k^{(1,\ldots,l)}) - (\mathbf{z}_1^{(1,\ldots,l)},\ldots,\mathbf{z}_k^{(1,\ldots,l)})\|\big) \leq \\
&\max_{l=1,\ldots,N_L} \sigma_T^{(l)}\ \|(\mathbf{x}_1,\ldots,\mathbf{x}_k) - (\mathbf{z}_1,\ldots,\mathbf{z}_k)\|,
\end{aligned}
\tag{A.4}
$$

and hence F is a contraction with $\sigma_T = \max\limits_{l=1,\ldots,N_L} \sigma_T^{(l)} < 1$.

Our last step is to find conditions under which the state transition functions of the individual reservoir layers implement contractions. For the first layer, for all $\mathbf{u} \in \mathcal{U}$ and for all $\mathbf{x}_1^{(1)},\ldots,\mathbf{x}_k^{(1)}$, $\mathbf{z}_1^{(1)},\ldots,\mathbf{z}_k^{(1)} \in \mathcal{R}^{(1)}$ we have that:

$$
\begin{aligned}
&\|F^{(1)}(\mathbf{u},\mathbf{x}_1^{(1)},\ldots,\mathbf{x}_k^{(1)}) - F^{(1)}(\mathbf{u},\mathbf{z}_1^{(1)},\ldots,\mathbf{z}_k^{(1)})\| = \\
&\|\tanh\left(\mathbf{W}^{(1)}\mathbf{u} + \textstyle\sum_{i=1}^k \hat{\mathbf{W}}^{(1)}\mathbf{x}_i^{(1)}\right) - \tanh\left(\mathbf{W}^{(1)}\mathbf{u} + \textstyle\sum_{i=1}^k \hat{\mathbf{W}}^{(1)}\mathbf{z}_i^{(1)}\right)\| \leq \\
&\|\textstyle\sum_{i=1}^k \hat{\mathbf{W}}^{(1)}(\mathbf{x}_i^{(1)} - \mathbf{z}_i^{(1)})\| \leq \\
&\|\hat{\mathbf{W}}^{(1)}\|\ \|\textstyle\sum_{i=1}^k (\mathbf{x}_i^{(1)} - \mathbf{z}_i^{(1)})\| \leq \\
&\|\hat{\mathbf{W}}^{(1)}\|\ \textstyle\sum_{i=1}^k \|(\mathbf{x}_i^{(1)} - \mathbf{z}_i^{(1)})\| \leq \\
&\|\hat{\mathbf{W}}^{(1)}\|\ k\ \max_{i=1}^k \|(\mathbf{x}_i^{(1)} - \mathbf{z}_i^{(1)})\| = \\
&(k\ \|\hat{\mathbf{W}}^{(1)}\|)\ \|(\mathbf{x}_1^{(1)},\ldots,\mathbf{x}_k^{(1)}) - (\mathbf{z}_1^{(1)},\ldots,\mathbf{z}_k^{(1)})\|,
\end{aligned}
\tag{A.5}
$$

from which it is easy to see that $\sigma_T^{(1)} = k\ \|\hat{\mathbf{W}}^{(1)}\|$ is a Lipschitz constant of $F^{(1)}$, and the first reservoir layer implements a contraction if $\sigma_T^{(1)} < 1$. For the l-th layer, with $l > 1$, assuming that the $(l-1)$-th layer implements a Lipschitz continuous state transition function with Lipschitz constant $\sigma_T^{(l-1)}$, we have that for all $\mathbf{u} \in \mathcal{U}$ and for all $\mathbf{x}_1^{(1,\ldots,l)},\ldots,\mathbf{x}_k^{(1,\ldots,l)}$, $\mathbf{z}_1^{(1,\ldots,l)},\ldots,\mathbf{z}_k^{(1,\ldots,l)} \in \mathcal{R}^{(1,\ldots,l)}$ it

results:

$$
\begin{aligned}
&\|F^{(l)}(\mathbf{u}, \mathbf{x}_1^{(1,\dots,l)}, \dots, \mathbf{x}_k^{(1,\dots,l)}) - F^{(l)}(\mathbf{u}, \mathbf{z}_1^{(1,\dots,l)}, \dots, \mathbf{z}_k^{(1,\dots,l)})\| = \\
&\| \tanh\big(\mathbf{W}^{(l)} F^{(l-1)}(\mathbf{u}, \mathbf{x}_1^{(1,\dots,l-1)}, \dots, \mathbf{x}_k^{(1,\dots,l-1)}) + \textstyle\sum_{i=1}^{k} \hat{\mathbf{W}}^{(l)} \mathbf{x}_i^{(l)}\big) - \\
&\quad \tanh\big(\mathbf{W}^{(l)} F^{(l-1)}(\mathbf{u}, \mathbf{z}_1^{(1,\dots,l-1)}, \dots, \mathbf{z}_k^{(1,\dots,l-1)}) + \textstyle\sum_{i=1}^{k} \hat{\mathbf{W}}^{(l)} \mathbf{z}_i^{(l)}\big)\| \leq \\
&\|\mathbf{W}^{(l)}\big(F^{(l-1)}(\mathbf{u}, \mathbf{x}_1^{(1,\dots,l-1)}, \dots, \mathbf{x}_k^{(1,\dots,l-1)}) - F^{(l-1)}(\mathbf{u}, \mathbf{z}_1^{(1,\dots,l-1)}, \dots, \mathbf{z}_k^{(1,\dots,l-1)})\big) + \\
&\textstyle\sum_{i=1}^{k}\big(\hat{\mathbf{W}}^{(l)}(\mathbf{x}_i^{(l)} - \mathbf{z}_i^{(l)})\big)\| \leq \\
&\|\mathbf{W}^{(l)}\|\, \|F^{(l-1)}(\mathbf{u}, \mathbf{x}_1^{(1,\dots,l-1)}, \dots, \mathbf{x}_k^{(1,\dots,l-1)}) - F^{(l-1)}(\mathbf{u}, \mathbf{z}_1^{(1,\dots,l-1)}, \dots, \mathbf{z}_k^{(1,\dots,l-1)})\| + \\
&\|\hat{\mathbf{W}}^{(l)}\|\, \textstyle\sum_{i=1}^{k} \|\mathbf{x}_i^{(l)} - \mathbf{z}_i^{(l)}\| \leq \\
&\|\mathbf{W}^{(l)}\|\, \sigma_T^{(l-1)}\, \|(\mathbf{x}_1^{(1,\dots,l-1)}, \dots, \mathbf{x}_k^{(1,\dots,l-1)}) - (\mathbf{z}_1^{(1,\dots,l-1)}, \dots, \mathbf{z}_k^{(1,\dots,l-1)})\| + \\
&\|\hat{\mathbf{W}}^{(l)}\|\, k\, \max_{i=1}^{k} \|\mathbf{x}_i^{(l)} - \mathbf{z}_i^{(l)}\| \leq \\
&\|\mathbf{W}^{(l)}\|\, \sigma_T^{(l-1)}\, \|(\mathbf{x}_1, \dots, \mathbf{x}_k) - (\mathbf{z}_1, \dots, \mathbf{z}_k)\| + \\
&\|\hat{\mathbf{W}}^{(l)}\|\, k\, \|(\mathbf{x}_1, \dots, \mathbf{x}_k) - (\mathbf{z}_1, \dots, \mathbf{z}_k)\| = \\
&\big(\|\mathbf{W}^{(l)}\|\, \sigma_T^{(l-1)} + k\, \|\hat{\mathbf{W}}^{(l)}\|\big)\, \|(\mathbf{x}_1, \dots, \mathbf{x}_k) - (\mathbf{z}_1, \dots, \mathbf{z}_k)\|,
\end{aligned}
\tag{A.6}
$$

from which we can see that $\sigma_T^{(l)} = \|\mathbf{W}^{(l)}\|\sigma_T^{(l-1)} + k\|\hat{\mathbf{W}}^{(l)}\|$ is a Lipschitz constant of $F^{(l)}$, and the l-th reservoir layer implements a contraction if $\sigma_T^{(l)} < 1$.

Solving the recursion in the definition of $\sigma_T^{(l)}$, we express the Lipschitz constant of the state transition function implemented by the l-th layer, for $l = 1, \dots, N_L$, as follows:

$$
\sigma_T^{(l)} = k \sum_{i=1}^{l} \|\hat{\mathbf{W}}^{(i)}\| \Big(\prod_{j=i+1}^{l} \|\mathbf{W}^{(j)}\| \Big). \tag{A.7}
$$

In light of the above discussion, if $\sigma_T^{(l)} < 1$ for all $l = 1, \dots, N_L$ than the DeepTESN global state transition function F is a contraction with degree of contractivity $\sigma_T = \max_{l=1,\dots,N_L} \sigma_T^{(l)}$. Hence a sufficient condition for the contractivity of F is that

$$
\sigma_T = \Big(k \max_{l=1,\dots,N_L} \Big(\sum_{i=1,\dots,l} \|\hat{\mathbf{W}}^{(i)}\| \prod_{j=i+1,\dots,l} \|\mathbf{W}^{(j)}\| \Big) \Big) < 1. \tag{A.8}
$$

Finally, in light of the previous result in equation A.3, if the DeepTESN satisfies the condition expressed in equation 10, then the TESP holds true, and Theorem 1 is proved. ■

Appendix B. Proof of Theorem 2

We study the sensitivity to initial conditions of the iterated version of the global state transition function of DeepTESN, i.e. of function $\hat{F}$ in equation 8. Specifically, we develop a necessary condition for the TESP by identifying cases in which equation 9 is certainly not satisfied. To this aim, we consider an input tree of height h, with a null (i.e. zero) label attached to each node. We denote such input tree as $\mathbf{0}_h$, where $\mathbf{u}(n) = \mathbf{0}_u \in \mathcal{U}$ for each $n \in \mathcal{N}(\mathbf{0}_h)$. We also assume that the input tree $\mathbf{0}_h$ is a perfect k-ary tree, i.e. all of its interior nodes have k children and all of its leaves are at the same depth h.

We are interested in studying the evolution of the DeepTESN reservoir system that receives in input $\mathbf{0}_h$ starting from different initial states $\mathbf{x}_0$ and $\mathbf{z}_0$, i.e. the behavior of the quantity

$$\|\hat{F}(\mathbf{0}_h, \mathbf{x}_0) - \hat{F}(\mathbf{0}_h, \mathbf{z}_0)\|. \tag{B.1}$$

As initial states, we consider a null (i.e. zero) state $\mathbf{x}_0 = \mathbf{0} \in \mathcal{R}^*$ and its perturbation $\mathbf{z}_0 = \mathbf{x}_0 + \delta\mathbf{x}_0 = \delta\mathbf{x}_0$, so that the quantity in equation B.1 reduces to

$$\|\hat{F}(\mathbf{0}_h, \delta\mathbf{x}_0)\|, \tag{B.2}$$

the behavior of which we are interested in characterizing. For now we shall not apply any particular restriction to $\delta\mathbf{x}_0$, while later in the proof we will identify a specific value of interest.

For the sake of mathematical convenience, we find it useful to introduce a variant of function F (equation 2) in which we assume that the children states are all the same. Such a function is denoted by G and is defined as follows:

$$\begin{aligned} &G : \mathcal{U} \times \mathcal{R}^* \to \mathcal{R}^* \\ &\text{For any } \mathbf{u} \in \mathcal{U}, \mathbf{x} \in \mathcal{R}^* : \quad G(\mathbf{u}, \mathbf{x}) = F(\mathbf{u}, \mathbf{x}, \ldots, \mathbf{x}). \end{aligned} \tag{B.3}$$

As for function F, considering the layer-wise DeepTESN reservoir organization, we can write $G = (G^{(1)}, G^{(2)}, \ldots, G^{(N_L)})$, where, for every input label $\mathbf{u} \in \mathcal{U}$ and state $\mathbf{x} \in \mathcal{R}^*$ we define:

$$\begin{aligned} &\text{- for the first layer} \\ &\quad G^{(1)}(\mathbf{u}, \mathbf{x}) = \tanh(\mathbf{W}^{(1)}\mathbf{u} + k\, \hat{\mathbf{W}}^{(1)}\mathbf{x}^{(1)}) \\ &\text{- for successive layers } l > 1 \\ &\quad G^{(l)}(\mathbf{u}, \mathbf{x}) = \tanh(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + k\, \hat{\mathbf{W}}^{(l)}\mathbf{x}^{(l)}) \end{aligned} \tag{B.4}$$

The iterated version of function G, indicated as $\hat{G}$, can be defined in the same way as function $\hat{F}$ in equation 8. Given these definitions, it is straightforward to see that the quantity in equation B.2 can be equivalently written as $\|\hat{G}(\mathbf{0}_h, \delta\mathbf{x}_0)\|$. Applying one step of the recursive definition of $\hat{G}$ we have that

$$\|\hat{G}(\mathbf{0}_h, \delta\mathbf{x}_0)\| = \|G(\mathbf{0}_u, \hat{G}(\mathbf{0}_{h-1}, \delta\mathbf{x}_0))\| = \|G(\mathbf{0}_u, \delta\mathbf{x}_1)\|, \tag{B.5}$$

where $\mathbf{0}_{h-1}$ is defined as $\mathbf{0}_h$, but with height $h-1$, and we set $\delta\mathbf{x}_1 = \hat{G}(\mathbf{0}_{h-1}, \delta\mathbf{x}_0)$, i.e. the DeepTESN state for the nodes at depth 1 in the input tree. By linearizing G around the null state $\mathbf{0}$, we can use the following approximation

$$G(\mathbf{0}_u, \delta\mathbf{x}1) \approx J_G(\mathbf{0}_u, \mathbf{0})\, \delta\mathbf{x}_1 + G(\mathbf{0}_u, \mathbf{0}) = J_G(\mathbf{0}_u, \mathbf{0})\, \delta\mathbf{x}_1, \tag{B.6}$$

where $J_G(\mathbf{0}_u, \mathbf{0})$ is the Jacobian matrix of G evaluated in the null state and for null input. Hence, we can write the left-hand side of equation B.5 as follows

$$\|\hat{G}(\mathbf{0}_h, \delta\mathbf{x}_0)\| \approx \|J_G(\mathbf{0}_u, \mathbf{0})\, \delta\mathbf{x}_1\| = \|J_G(\mathbf{0}_u, \mathbf{0})\, \hat{G}(\mathbf{0}_{h-1}, \delta\mathbf{x}_0)\|, \tag{B.7}$$

where in the right-hand side we have expanded the $\delta\mathbf{x}_1$ term. By applying again the steps of recursive definition of $\hat{G}$ and linearization of G around the null state, we have that

$$\|\hat{G}(\mathbf{0}_h, \delta\mathbf{x}_0)\| \approx \|J_G(\mathbf{0}_u, \mathbf{0})^2\, \delta\mathbf{x}_2\|, \tag{B.8}$$

where $\delta\mathbf{x}_2 = \hat{G}(\mathbf{0}_{h-2}, \delta\mathbf{x}_0)$ is the DeepTESN state for the nodes at depth 2 in the input tree. Iterating the same process h times we find that

$$\|\hat{G}(\mathbf{0}_h, \delta\mathbf{x}_0)\| \approx \|J_G(\mathbf{0}_u, \mathbf{0})^h\, \delta\mathbf{x}_0\|. \tag{B.9}$$

At this stage, by exploiting the definition of (induced) matrix norm, we can identify $\delta\mathbf{x}_0$ as the vector such that $\|J_G(\mathbf{0}_u, \mathbf{0})^h\, \delta\mathbf{x}_0\| = \|J_G(\mathbf{0}_u, \mathbf{0})^h\|$, which, in the limit of $h \to \infty$, approaches the quantity $\rho\big(J_G(\mathbf{0}_u, \mathbf{0})\big)^h$. Hence, we have found an input tree $\mathbf{t}_h = \mathbf{0}_h$ and two initial states $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{z}_0 = \delta\mathbf{x}_0$ such that:

$$\|\hat{F}(\mathbf{t}_h, \mathbf{x}_0) - \hat{F}(\mathbf{t}_h, \mathbf{z}_0)\| \approx \rho\big(J_G(\mathbf{0}_u, \mathbf{0})\big)^h. \tag{B.10}$$

Considering the layer-wise components of G, the Jacobian in the right-hand side of equation B.10 can be written as the following block matrix:

$$J_G(\mathbf{0}_u, \mathbf{0}) = \begin{pmatrix} J_{G^{(1)},\mathbf{x}^{(1)}} & J_{G^{(1)},\mathbf{x}^{(2)}} & \dots & J_{G^{(1)},\mathbf{x}^{(N_L)}} \\ J_{G^{(2)},\mathbf{x}^{(1)}} & J_{G^{(2)},\mathbf{x}^{(2)}} & \dots & J_{G^{(2)},\mathbf{x}^{(N_L)}} \\ \vdots & \vdots & \ddots & \vdots \\ J_{G^{(N_L)},\mathbf{x}^{(1)}} & J_{G^{(N_L)},\mathbf{x}^{(2)}} & \dots & J_{G^{(N_L)},\mathbf{x}^{(N_L)}} \end{pmatrix} \tag{B.11}$$

where for every $i, j = 1, \dots, N_L$, $J_{G^{(i)},\mathbf{x}^{(j)}}$ denotes the partial derivative of $G^{(i)}$ with respect to $\mathbf{x}^{(j)}$, evaluated at $(\mathbf{0}_u, \mathbf{0})$. Taking into account the hierarchical organization of the deep recursive reservoir of DeepTESN, we can notice that $J_G(\mathbf{0}_u, \mathbf{0})$ in equation B.11 has the structure of a lower-triangular block matrix, as for every $j > i$ it results that $J_{G^{(i)},\mathbf{x}^{(j)}}$ is a null matrix (lower layers are independent of higher layers). Moreover, with the purpose of computing the spectral radius of $J_G(\mathbf{0}_u, \mathbf{0})$ we can limit our attention to the blocks $J_{G^{(l)},\mathbf{x}^{(l)}}$, for $l = 1, \dots, N_L$, as the set of eigenvalues of a triangular block matrix is given by the set of eigenvalues of its diagonal blocks (see e.g. proof of Lemma 2 in [22]). Exploiting equation B.4, we find that $J_{G^{(l)},\mathbf{x}^{(l)}} = k\, \hat{\mathbf{W}}^{(l)}$, and hence the spectral radius of $J_G(\mathbf{0}_u, \mathbf{0})$ takes the following form:

$$\rho\big(J_G(\mathbf{0}_u, \mathbf{0})\big) = \max_{l=1,\dots,N_L} \rho(k\, \hat{\mathbf{W}}^{(l)}) = k \max_{l=1,\dots,N_L} \rho(\hat{\mathbf{W}}^{(l)}). \tag{B.12}$$

Finally, using equation B.12 in equation B.10 we have that

$$\|\hat{F}(\mathbf{t}_h, \mathbf{x}_0) - \hat{F}(\mathbf{t}_h, \mathbf{z}_0)\| \approx \big(k \max_{l=1,\dots,N_L} \rho(\hat{\mathbf{W}}^{(l)})\big)^h. \tag{B.13}$$

By setting $\rho_T = k \max_{l=1,\dots,N_L} \rho(\hat{\mathbf{W}}^{(l)})$, we can conclude that if $\rho_T \geq 1$ then the TESP in equation 9 is not satisfied, as we have found an example of a case in which $\hat{F}(\mathbf{t}_h, \mathbf{x}_0)$ and $\hat{F}(\mathbf{t}_h, \mathbf{z}_0)$ will never converge to each other in the limit of $h \to \infty$. Hence, a necessary condition for the TESP to hold is that

$$\rho_T = \big(k \max_{l=1,\dots,N_L} \rho(\hat{\mathbf{W}}^{(l)})\big) < 1$$

and Theorem 2 is proved.
As a final remark, we can note that, in the example considered here, the quantity ρ_T played a crucial role in characterizing the degree of stability of DeepTESN (in terms of sensitivity to initial conditions). Analogously, in light of equation B.12, if we focus on considering only the first l layers in the deep recursive architecture, then the stability of the resulting system is characterized by the quantity $\rho_T^{(l)} = (k \max_{j=1,\dots,l} \rho(\hat{\mathbf{W}}^{(j)}))$. ■

Appendix C. Selected Hyper-parameters

In this section we report the values of DeepTESN and TESN hyper-parameters, selected through model selection for the tasks considered in the experimental analysis in Section 5. In light of the mathematical investigations provided in Section 4, a specific focus is given to the spectral radius of the recursive reservoir weight matrices.

Tables C.5, C.6 and C.7 show the values of the spectral radius, corresponding degree of stability of DeepTESN dynamics (equation. 11), input scaling, inter-layer scaling and readout regularization coefficient (for ridge-regression training), selected respectively for the Cystic, Leukemia and INEX 2006 tasks. Results correspond to the largest reservoir settings considered, namely 500 units for the glycans tasks (Tables C.5, C.6), and 3000 units for the INEX 2006 task (Table C.7). For DeepTESN, the reservoir units are organized in 10 layers.

Cystic

Hyper-parameter	DeepTESN	TESN
Spectral radius ρ	0.21	0.62
Degree of stability ρ_T (eq.11)	0.26	0.75
Input scaling $scale_{in}$	2.70	3.36
Inter-layer scaling $scale_{il}$	3.88	-
Log regularization coefficient $log(\lambda_r)$	-0.4	-1.2

Table C.5: Selected DeepTESN and TESN hyper-parameters on the Cystic task (averaged on the external folds of the cross-validation). Results correspond to reservoirs with 500 units, organized in 10 layers in the case of DeepTESN.

Leukemia

Hyper-parameter	DeepTESN	TESN
Spectral radius ρ	0.42	0.91
Degree of stability ρ_T (eq.11)	0.49	1.06
Input scaling $scale_{in}$	3.05	4.36
Inter-layer scaling $scale_{il}$	4.15	-
Log regularization coefficient $log(\lambda_r)$	0.3	-0.7

Table C.6: Selected DeepTESN and TESN hyper-parameters on the Leukemia task (averaged on the external folds of the cross-validation). Results correspond to reservoirs with 500 units, organized in 10 layers in the case of DeepTESN.

INEX 2006

Hyper-parameter	DeepTESN	TESN
Spectral radius ρ	1.48	1.27
Degree of stability ρ_T (eq.11)	3.21	2.76
Input scaling $scale_{in}$	4.50	4.32
Inter-layer scaling $scale_{il}$	4.86	-
Log regularization coefficient $log(\lambda_r)$	2.0	-2.0

Table C.7: Selected DeepTESN and TESN hyper-parameters on the INEX 2006 task. Results correspond to reservoirs with 3000 units, organized in 10 layers in the case of DeepTESN.

From Tables C.5 and C.6 we observe that the selected degree of networks stability is smaller than 1 for both DeepTESN and TESN (with a value very close to 1 in the case of TESN for the Leukemia task), and the chosen hyper-parametrizations fall within the constraint expressed by the necessary condition for the TESP (Theorem 2). Interestingly, Table C.7 shows that in the case of INEX 2006, for both DeepTESN and TESN the best networks configurations are obtained in correspondence of a spectral radius (and corresponding degree of stability) exceeding 1. On the one hand this confirms similar results obtained for standard ESNs in real-world applications (e.g. in [47]), and on the other hand it highlights even further the necessary nature of the constraint on the spectral radius of reservoir matrices also in the case of DeepTESN, as already discussed in Section 4. Finally, we can note that, for all the the considered tasks, the selected DeepTESN configurations are generally characterized by large inter-layer scalings and by regularization coefficients that are larger than those selected in the corresponding TESN cases.