Computer Science and Information Systems 19(1):379–396 https://doi.org/10.2298/CSIS210218063D

On the effectiveness of Gated Echo State Networks for data exhibiting long-term dependencies

Daniele Di Sarli, Claudio Gallicchio, and Alessio Micheli

Department of Computer Science
University of Pisa
Pisa, Italy
daniele.disarli@phd.unipi.it
{gallicch,micheli}@di.unipi.it

Abstract. In the context of recurrent neural networks, gated architectures such as the GRU have contributed to the development of highly accurate machine learning models that can tackle long-term dependencies in the data. However, the training of such networks is performed by the expensive algorithm of gradient descent with backpropagation through time. On the other hand, reservoir computing approaches such as Echo State Networks (ESNs) can produce models that can be trained efficiently thanks to the use of fixed random parameters, but are not ideal for dealing with data presenting long-term dependencies. We explore the problem of employing gated architectures in ESNs from both theoretical and empirical perspectives. We do so by deriving and evaluating a necessary condition for the non-contractivity of the state transition function, which is important to overcome the fading-memory characterization of conventional ESNs. We find that using pure reservoir computing methodologies is not sufficient for effective gating mechanisms, while instead training even only the gates is highly effective in terms of predictive accuracy.

Keywords: echo state networks, gated recurrent neural networks, reservoir computing.

1. Introduction

Several approaches are possible for learning functions over temporal domains. Recurrent Neural Networks (RNNs) represent an effective neural architecture for temporal tasks, with applications in many different domains [20]. When it comes to the training algorithm for RNNs, we distinguish two major approaches.

The first approach is *reservoir computing* [21,31], in which the neural network is studied as a dynamical system and involves the encoding of the input sequence and its temporal features into a fixed size vector in the state space. The peculiar characteristic of the reservoir computing approach is that the weights of the RNN are not trained: instead, only the weights associated to a simple stateless readout layer get trained. This allows for very fast and efficient trainings. A widely known instance of reservoir computing model is represented by Echo State Networks (ESN) [17,16]. ESNs and other reservoir computing models are tightly connected to the concepts of Markovian bias [30,11], fading-memory [12], and contractivity [16], which are fundamental characteristics of their state dynamics.

On the other hand, the second and most popular approach involves training *all* network weights by gradient methods, namely gradient descent. The flexibility of this approach allowed the emergence of architectural variants which, while maintaining the computational

power of simple RNNs, can make training easier over data exhibiting long-term dependencies [5]. Instances of such variants are LSTM [14] and GRU [7], which thanks to the introduction of so-called gating mechanisms allow the network to remember or discard from the internal state selected information about the input sequences (or, from the point of view of the gradient computation, can alleviate the problem of gradient vanishing [5]). While definitely effective in terms of predictive accuracy, gated network models still require gradient descent for training, which is often much more computationally expensive than the reservoir computing approaches.

In recent years, there has been an increasing research interest regarding alternative solutions for maintaining information over long time spans in recurrent models. An example is the application of the Learning-to-Learn paradigm and neuronal adaptation to spiking neural networks in the context of reservoir computing methodologies [29,2]. In this paper we focus on the following question: *can typical gating mechanisms be embedded within efficient reservoir computing networks, and within what degree of effectiveness?*

In ESNs, whose recurrent dynamics are completely untrained, it is not immediate to extend the architecture with gate-like mechanisms. In this work we investigate whether it is possible to extend the architecture of an ESN by introducing gating mechanisms, and we analyze the results in terms of training efficiency and predictive performance. The goal is to make progress towards efficient neural models which improve the predictive performance with respect to the current reservoir computing state-of-the-art, while maintaining most of the efficiency.

In short, we summarize here the key contributions of this work which are:

- the extension of the ESN architecture with the use of gating mechanisms, which we denote as *Gated ESN*;
- the theoretical analysis of the conditions associated to the fading memory of the network dynamics of the Gated ESN;
- a critical experimental analysis of the Gated ESN; and
- the suggestion of likely paths towards effective gated reservoir computing models.

Regarding the organization of the manuscript, it is laid out as follows: we start by discussing a few related works in Section 2 before introducing, in Section 3, the two main approaches from the state-of-the-art literature regarding RNN training, namely ESNs and GRUs. In Section 4 we describe the model that we study (Gated ESN), and we provide a theoretical analysis for the conditions related to memory and stability in Section 5. Then, in Section 6 we report the methodology and the results regarding our experimental analysis, whose implications are discussed in Section 7. Finally, in Section 8 we draw the conclusions.

2. Related Works

The process of extending the architecture of ESNs with gating mechanisms has been first investigated in our previous work [9] and, concurrently and independently, in [32]. The two works share a similar idea, which consists of extending the state transition function of an ESN to include the same gating mechanisms of a GRU, while keeping all the parameters in the gates untrained. Both works investigate the behavior of such network

when trained by conventional reservoir computing techniques (as opposed to the typical approach of training GRUs by using gradient descent).

While the underlying idea from [9] and [32] is similar, in [9] and in the current work we also take care to (1) consider the fundamental details regarding the initialization strategy, (2) perform an experimental evaluation over a dataset that makes it possible to evaluate more clearly the actual influence of the gates, and (3) perform a more extensive evaluation of the competing reservoir computing models, which led to important insights about the feasibility of the approach. Moreover, the current paper further extends our previous work [9] to include (a) the development of a theoretical analysis of the state dynamics in the proposed models, (b) the analysis and discussion of the agreement between the theoretical results and the experimental measurements, (c) the expansion of the hyperparameter search for the experiments, (d) the collection and discussion of additional measurements for the activation of the gates, and (e) the reporting and discussion of additional measurements regarding the weight matrices.

Finally, in the *Gating ESN* model [1] from earlier literature, it is employed a combination of many parallel instances of ESNs, each initialized with different hyperparameters, and each trained separately on the same task. A gating network then learns to select which instance to use for any given time-step. While the name of our proposed model may bear similarity with the *Gating ESN*, the approaches are radically different: instead of instantiating many different sub-models, we aim to enrich the dynamics of the state of a single ESN by explicitly introducing gated units (cells) in the spirit of architectures such as GRU [7] and LSTM [14].

3. Background

In this section we briefly describe the models that serve as the basis of our study: in Section 3.1 we describe the Echo State Networks (ESNs), a reservoir computing approach for modeling sequences, while in Section 3.2 we describe the popular approach of Gated Recurrent Units (GRUs), representing one of several existing variants of gated RNNs.

3.1. Echo State Networks

Among the different instances of the general framework of RNNs, ESNs [16,17] represent an efficient approach for sequence modeling thanks to the use of a distinctive perspective for the study of the recurrent network. As reservoir computing models, in ESNs there is a sharp distinction between the recurrent part of the network, which is referred to as *reservoir*, and the output layer, which is called *readout*. The *reservoir* is studied as a dynamical system and is responsible for embedding the input sequence into a high dimensional state space of fixed size. The key characteristic is that the reservoir does not get trained. Instead, the weights in the reservoir are initialized from a random distribution that allows to meet a mathematical property for stability. We will discuss this property in the final part of this section. The *readout* is typically implemented as a linear regression model. Since it is the only part of the model that gets trained, it is possible to obtain closed-form solution to the regression problem.

The architecture of an ESN has three main dimensions: the number of input units (N_U), the number of units in the reservoir (N_R) and the number of output units in the

readout (N_Y). Then, the ESN can be applied to sequences $\mathbf{u}(1), \ldots, \mathbf{u}(T) \in \mathbb{R}^{N_U}$ of any length T.

The state of the reservoir of the network at each time step t, which is denoted as $\mathbf{x}(t) \in \mathbb{R}^{N_R}$, is computed as

$$\begin{aligned} \mathbf{x}(0) &= \mathbf{0}, \\ \mathbf{x}(t) &= \tanh\left(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t-1)\right), \end{aligned} \tag{1}$$

where $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix, and $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent reservoir-to-reservoir weight matrix.

Instead of being tuned by a training process, the matrices $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$ are fixed after being randomly initialized. As part of the random initialization process, $\mathbf{W}_{in}$ is multiplied by a real scaling hyperparameter. The matrix $\hat{\mathbf{W}}$, instead, is initialized so that its norm $\|\hat{\mathbf{W}}\|$ or spectral radius $\rho(\hat{\mathbf{W}})$ (the largest eigenvalue in absolute value) meets the conditions for stability given in [16]. We report a sufficient condition for stability at the end of this section.

After the states for the input sequence have been collected, the output is computed as

$$\mathbf{y}(t) = \mathbf{W}_{out}\mathbf{x}(t), \tag{2}$$

with $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_R}$.

Leaky ESN – A simple but effective variant of the basic ESN is denoted as *leaky ESN*. In the leaky ESN, the neurons in the reservoir are leaky-integrators [18] which act as lowpass filters. Their leaking rate is considered a hyperparameter of the model, and as such is chosen and fixed at model selection time. For a leaky ESN, the state transition function of the reservoir is defined as

$$\begin{aligned} \mathbf{x}(0) &= \mathbf{0}, \\ \mathbf{x}(t) &= (1-a)\,\mathbf{x}(t-1) + a\tanh\left(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t-1)\right), \end{aligned} \tag{3}$$

where $a \in \mathbb{R}$ is the leaking rate, under the constraint that $0 < a \leq 1$.

Training – The characteristic of ESNs and its variants such as leaky ESNs is that the weights in the reservoir are not trained. As such, the only parameters that need to be optimized are those in the readout, i.e., the matrix $\mathbf{W}_{out}$. Since the readout is limited to a linear computation, a closed-form solution to the convex minimization of the error can be obtained by algorithms such as ridge regression. In practice, first the input data is fed to the reservoir and the N_{train} states that need to be classified are collected column-wise into a matrix $\mathbf{X} \in \mathbb{R}^{N_R \times N_{train}}$. Then, the readout is trained by finding a solution to the following least squares problem:

$$\min_{\mathbf{W}_{out}} \|\mathbf{W}_{out}\mathbf{X} - \mathbf{Y}_{tg}\|_2^2. \tag{4}$$

In Equation 4, $\mathbf{Y}_{tg} \in \mathbb{R}^{N_Y \times N_{train}}$ indicates the column-wise concatenation of the target vectors. A regularized solution to Equation 4 can be computed in closed-form as follows:

$$\mathbf{W}_{out} = \mathbf{Y}_{tg}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda_r\mathbf{I})^{-1}, \tag{5}$$

where $\mathbf{I}$ is the identity matrix, and $\lambda_r \in \mathbb{R}^+$ is the regularization parameter which can be chosen by model selection.

Echo State Property – To guarantee the stability conditions that allow the untrained state dynamics to encode meaningful representations of the data, the reservoir must meet the so-called Echo State Property (ESP) [16,33]. The ESP is related to asymptotic properties of the reservoir and intuitively states that, for sufficiently long input sequences, the state in which the network ends up should only depend upon the input itself. In other words, the initial conditions of the network should not influence its long-term dynamics. For ESNs with hyperbolic tangent activation functions, a sufficient condition for the ESP is $\|\hat{\mathbf{W}}\|_2 < 1$ (see [16]).

3.2. Gated Recurrent Units

The problems associated to gradient descent training with backpropagation through time over long input sequences [5] led to the development of gated RNN models such as LSTM [14] and GRU [7]. The gates are simple mechanisms that are able to dynamically squash to zero the individual components of the possibly multidimensional signal to which they are applied. In the case of GRU, the state transition function contains two gates which are called *reset gate* and *update gate*. The activations of such gates at time t are denoted as respectively $\mathbf{r}(t) \in \mathbb{R}^{N_R}$ and $\mathbf{z}(t) \in \mathbb{R}^{N_R}$. Informally, a gate is said to *open* or *close* depending on the values of its activations, which vary in $(0, 1)$.

Intuitively, the purpose of the gates in the GRU is to open and close to regulate the flow of information within the state: the reset gate can zero out unnecessary information from the previous state $\mathbf{x}(t-1)$, while the update gate can merge information from the previous state and the current candidate state $\mathbf{h}(t)$ into the new state $\mathbf{x}(t)$. More in detail, the recurrent state $\mathbf{x}(t)$ of a GRU at a given time step t is computed as:

$$
\begin{aligned}
&\mathbf{x}(0) = \mathbf{0}, \\
&\mathbf{r}(t) = \sigma(\mathbf{W}_{in}^{r}\mathbf{u}(t) + \hat{\mathbf{W}}^{r}\mathbf{x}(t-1)) \\
&\mathbf{z}(t) = \sigma(\mathbf{W}_{in}^{z}\mathbf{u}(t) + \hat{\mathbf{W}}^{z}\mathbf{x}(t-1)) \\
&\mathbf{h}(t) = \tanh(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}(\mathbf{r}(t) \odot \mathbf{x}(t-1))) \\
&\mathbf{x}(t) = \mathbf{z}(t) \odot \mathbf{x}(t-1) \; + \; (1 - \mathbf{z}(t)) \odot \mathbf{h}(t).
\end{aligned}
\tag{6}
$$

Here, we have $\mathbf{r}(t), \mathbf{z}(t), \mathbf{h}(t), \mathbf{x}(t) \in \mathbb{R}^{N_R}$, and in particular $\mathbf{r}(t), \mathbf{z}(t) \in (0, 1)$ due to the sigmoidal activation function.

From a given state of the GRU, a prediction can be obtained by means of any kind of differentiable readout layer, such as a linear one. The whole model (including the behavior of the gates) can be trained end-to-end by backpropagation through time.

4. Gated ESN

The capability of ESNs to discriminate between different input sequences depends on the guarantees given by the ESP [16] described in Section 3.1. The ESP is related to the *fading-memory* property of the ESN, i.e., in a properly initialized ESN any information from the initial conditions of the reservoir will be asymptotically washed out with time. It is easy to see how the fading memory property, if not properly controlled, can

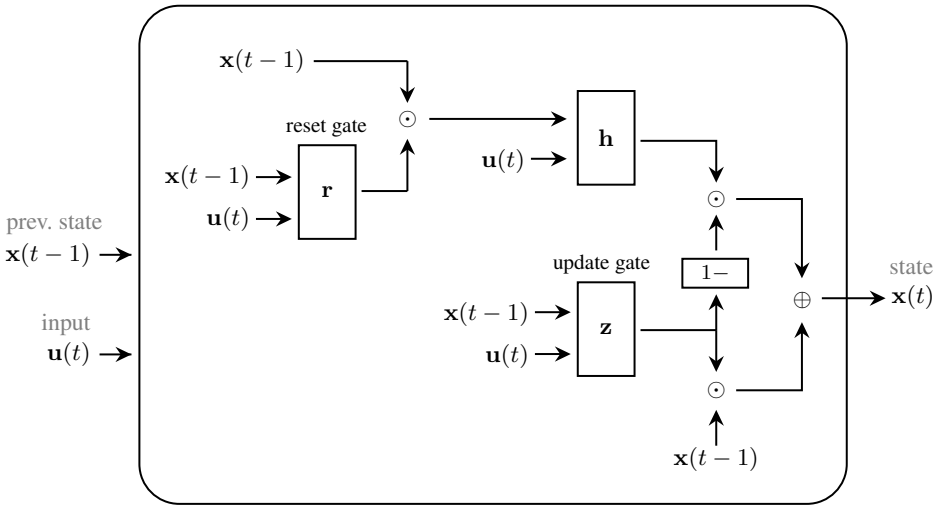


Fig. 1. Depiction of the recurrent cell of *Gated ESN* (and *Gated ESN RZ*) for a generic time step t. Symbols $\odot$ and $\oplus$ respectively denote the elementwise product and the sum of two vectors. While the architecture is identical to the one of a GRU, in *Gated ESN* the parameters controlling the activations of $\mathbf{r}(t), \mathbf{z}(t)$, and $\mathbf{h}(t)$ are not trained (hence only the readout is trained, which is not depicted here). For *Gated ESN RZ*, instead, in addition to the readout also the parameters controlling $\mathbf{r}(t)$ and $\mathbf{z}(t)$ are trained while the dynamics of $\mathbf{h}(t)$ remain untrained

also represent a dramatic limitation by introducing a Markovian bias [11] in the model. For example, imagine a task characterized by long input sequences in which the key information for performing correct predictions is often located near the beginning of the sequences. In such cases, the Markovian bias may prevent the readout to access such information, thus reducing the learning capability of the model. One may argue that it is always possible to optimize the amount of memory of the model (e.g., by increasing the number of recurrent units) so that distant information will not be lost, however this strategy does not allow to generalize to different sequence lengths. What is needed is a way for ESNs to dynamically and selectively remember or forget parts of a sequence while preserving their generalization capabilities.

We investigate whether it is possible to employ gating mechanisms in order to improve the predictive performance of ESNs in such contexts. To this aim, we make use of a gated architecture paired with a reservoir computing training methodology. In particular, we define two models which only differ for their training method. In fact, both models borrow the same state transition function of the GRU (Equation 6) as illustrated in Fig. 1. However, in the first model the recurrent part is fully untrained, while in the second one we make partial use of training for the gates. The whole model is illustrated in Fig. 2 by showing its unfolding in time. The details of the recurrent cells for Gated ESN and Gated ESN RZ are described in Sections 4.1 and 4.2.

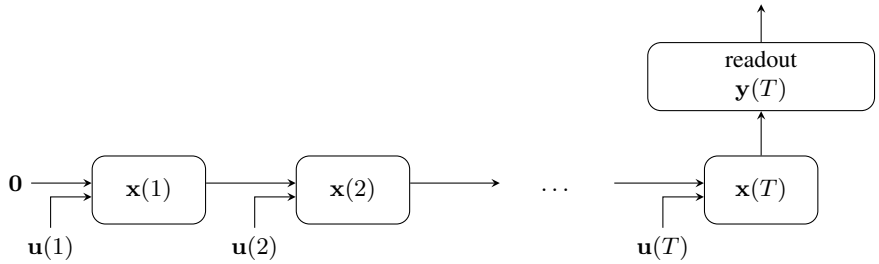


Fig. 2. Unfolding in time of the Gated ESN and Gated ESN RZ architecture for a sequence of length T. Each state $\mathbf{x}(t)$ is computed by the recurrent cell illustrated in Fig. 1. Since in this work we are concerned with the classification of a sequence given the last time step, we only show that configuration.

Table 1. Schematic view of the different matrices involved in Gated ESN, Gated ESN RZ and GRU. On the right we report whether a given matrix is tuned as part of the training procedure

Matrix	Shape	Description	Trained?		
			Gated ESN	Gated ESN RZ	GRU
$\mathbf{W}_{in}^{r}$	$N_R \times N_U$	input to reset gate	–	✓	✓
$\hat{\mathbf{W}}^{r}$	$N_R \times N_R$	reservoir to reset gate	–	✓	✓
$\mathbf{W}_{in}^{z}$	$N_R \times N_U$	input to update gate	–	✓	✓
$\hat{\mathbf{W}}^{z}$	$N_R \times N_R$	reservoir to update gate	–	✓	✓
$\mathbf{W}_{in}$	$N_R \times N_U$	input to reservoir	–	–	✓
$\hat{\mathbf{W}}$	$N_R \times N_R$	reservoir to reservoir	–	–	✓
$\mathbf{W}_{out}$	$N_Y \times N_R$	readout	✓	✓	✓

4.1. Gated ESN

We denote the first variant as *Gated ESN*. This can be considered a pure reservoir computing model in the sense that its reservoir is fully untrained and its inner mechanics (which resemble those of a GRU) are completely irrelevant for the training algorithm of the readout. In particular, all matrices in the reservoir (including those in the gates) are randomly initialized and then rescaled according to the value of corresponding hyperparameters, just like what happens in a standard ESN. The only parameters that get trained are those in the linear readout, as highlighted in Table 1, with no difference with respect to what happens in a standard ESN. In fact, the parameters of the readout can be obtained by ridge regression.

4.2. Gated ESN RZ

We denote the second variant as *Gated ESN RZ*. The name comes from the fact that in this case, the behavior of the reset and update gates (*R* and *Z*) is learned from the data. Due to the multiple nonlinearities separating the output of the model with the parameters of the gates, the training algorithm of choice is gradient descent with backpropagation through time for both the parameters of the gates ($\mathbf{W}_{in}^{r}$, $\mathbf{W}_{in}^{z}$, $\hat{\mathbf{W}}^{r}$, and $\hat{\mathbf{W}}^{z}$) and, jointly,

those of the readout ($\mathbf{W}_{out}$). The other matrices of the reservoir, namely $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$, are randomly initialized, rescaled and then kept fixed as in *Gated ESN*. A summary of which are the matrices that get trained is available in Table 1.

5. Contractivity conditions of the gated reservoir

In this section we provide an analysis of the state dynamics for a gated reservoir. In particular, we seek conditions to escape from the inherent fading-memory behavior of conventional reservoir computing approaches. To this end, we derive a bound that the reservoir matrices must satisfy when the state transition function is non-contractive. The results are insightful for the initialization and the analysis of such systems, as they enable us to characterise the contractivity (and the resulting fading-memory regime) of such gated architectures. We focus our analysis over the architecture of GRU, regardless of whether it is trained (as, precisely, in GRU), untrained (as in Gated ESN) or partially trained (as in Gated ESN RZ). Thus, the results apply to all these models after initialization and, where applicable, training.

For ease of notation, in this section let us hide the explicit time dependency and denote with $\mathbf{u}$ and $\mathbf{x}$ respectively a generic input vector and state vector. The state transition function of the reservoir will be represented by the function $\tau : \mathbb{R}^{N_U} \times \mathbb{R}^{N_R} \to \mathbb{R}^{N_R}$. Moreover, we denote with $\left(\frac{\partial y(x)}{\partial x}\right)_{f(x)}$ the partial derivative of y with respect to x while considering $f(x)$ as a constant.

For the ESP to hold, it can be shown that it is sufficient for the state transition function to be contractive. Contractivity is defined as:

$$\begin{gathered}\exists C \in \mathbb{R}, \quad 0 \leq C < 1, \quad \forall \mathbf{u} \in \mathbb{R}^{N_U}, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R} : \\ \left\|\tau(\mathbf{u}, \mathbf{x}) - \tau(\mathbf{u}, \mathbf{x}')\right\| \leq C \left\|\mathbf{x} - \mathbf{x}'\right\|,\end{gathered} \tag{7}$$

i.e. τ must be Lipschitz continuous with constant $C < 1$. Equation 7 can also be formulated in terms of the derivative of τ, in fact:

$$\begin{gathered}\forall C \in \mathbb{R}, \quad C \geq 0, \quad \forall \mathbf{u} \in \mathbb{R}^{N_U}, \quad \forall \mathbf{x} \in \mathbb{R}^{N_R} : \\ \tau \text{ is } C\text{-Lipschitz w.r.t. } \mathbf{x} \quad \iff \quad \left\| \frac{\partial \tau(\mathbf{u}, \mathbf{x})}{\partial \mathbf{x}} \right\| \leq C.\end{gathered} \tag{8}$$

We now study the characterization of the asymptotic stability of the state dynamics for a GRU. First, in Lemma 1 we show that for a GRU whose weights are within a given bound, the contractivity of the state transition function is guaranteed. Then we use the aforementioned bound to derive the main result of Proposition 1, which gives a necessary condition for a state transition function that is non-contractive.

Lemma 1. *Let τ be the state transition function of a GRU as defined in Equation 6, and let $z_{max} = \max_t \left\|\mathbf{z}(t)\right\|_\infty$. A sufficient condition for the contractivity of τ is:*

$$\|\hat{\mathbf{W}}\|_2 \left(1 + \|\hat{\mathbf{W}}^r\|_2\right) + 2\|\hat{\mathbf{W}}^z\|_2 + z_{max} < 1. \tag{9}$$

Proof. We write the state transition function $\tau(\mathbf{u}, \mathbf{x})$ as

$$\begin{aligned} \mathbf{r} &= \sigma(\mathbf{W}_{in}^{r}\mathbf{u} + \hat{\mathbf{W}}^{r}\mathbf{x}) \\ \mathbf{z} &= \sigma(\mathbf{W}_{in}^{z}\mathbf{u} + \hat{\mathbf{W}}^{z}\mathbf{x}) \\ \mathbf{h} &= \tanh(\mathbf{W}_{in}\mathbf{u} + \hat{\mathbf{W}}(\mathbf{r} \odot \mathbf{x})) \\ \tau(\mathbf{u}, \mathbf{x}) &= \mathbf{z} \odot \mathbf{x} \ + \ (1 - \mathbf{z}) \odot \mathbf{h}, \end{aligned} \tag{10}$$

Then we can compute the 2-norm of the partial derivative of τ with respect to x as follows:

$$\begin{aligned} \left\| \frac{\partial \tau(\mathbf{u}, \mathbf{x})}{\partial \mathbf{x}} \right\|_2 &= \left\| \frac{\partial \tau(\mathbf{u}, \mathbf{x})}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{x}} + \frac{\partial \tau(\mathbf{u}, \mathbf{x})}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{x}} + \left(\frac{\partial \tau(\mathbf{u}, \mathbf{x})}{\partial \mathbf{x}} \right)_{\substack{\mathbf{h} \\ \mathbf{z}}} \right\|_2 \\ &= \Big\| \operatorname{diag}(1 - \mathbf{z}) \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \\ &\quad + \operatorname{diag}\big((\mathbf{x} - \mathbf{h}) \odot \mathbf{z} \odot (1 - \mathbf{z})\big) \hat{\mathbf{W}}^{z} \\ &\quad + \operatorname{diag}(\mathbf{z}) \Big\|_2 \\ &\leq \|1 - \mathbf{z}\|_\infty \left\| \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right\|_2 \\ &\quad + \|(\mathbf{x} - \mathbf{h}) \odot \mathbf{z} \odot (1 - \mathbf{z})\|_\infty \|\hat{\mathbf{W}}^{z}\|_2 \\ &\quad + \|\mathbf{z}\|_\infty \\ &\leq \|1 - \mathbf{z}\|_\infty \|\hat{\mathbf{W}}\|_2 \left(1 + \|\hat{\mathbf{W}}^{r}\|_2\right) \\ &\quad + \|(\mathbf{x} - \mathbf{h}) \odot \mathbf{z} \odot (1 - \mathbf{z})\|_\infty \|\hat{\mathbf{W}}^{z}\|_2 \\ &\quad + \|\mathbf{z}\|_\infty \\ &\leq \|\hat{\mathbf{W}}\|_2 \left(1 + \|\hat{\mathbf{W}}^{r}\|_2\right) + 2\|\hat{\mathbf{W}}^{z}\|_2 + \|\mathbf{z}\|_\infty \\ &\leq \|\hat{\mathbf{W}}\|_2 \left(1 + \|\hat{\mathbf{W}}^{r}\|_2\right) + 2\|\hat{\mathbf{W}}^{z}\|_2 + z_{max}. \end{aligned} \tag{11}$$

where $\operatorname{diag}(\mathbf{v})$ (for a generic vector $\mathbf{v}$) is the diagonal matrix with the entries of $\mathbf{v}$ on the main diagonal, and $\|\mathbf{v}\|_\infty = \max_i |v_i|$ is the infinity norm of vector $\mathbf{v}$.

Equation 11 provides an upper bound to the derivative of τ. From Equation 8 it follows that when this bound is less than unity, τ is contractive. □

While the contractivity of the state transition function ensures that the ESP is satisfied, the whole idea of a gated architecture is for the state dynamics to not be restricted to contractive trajectories. This would allow the network to relax the strong Markovian bias discussed at the beginning of Section 4 and escape from the strict fading memory behaviour. Then, the initialization strategy must ensure that the network is outside of a strictly contracting regime. From the bound of Lemma 1 we can obtain a necessary condition for having a non-contractive state transition function, as reported in Proposition 1.

Proposition 1. *Let τ be the state transition function of a GRU as defined in Equation 6, and let $z_{max} = \max_t \|\mathbf{z}(t)\|_\infty$. If τ is non-contractive, then it holds:*

$$\|\hat{\mathbf{W}}\|_2 \left(1 + \|\hat{\mathbf{W}}^{r}\|_2\right) + 2\|\hat{\mathbf{W}}^{z}\|_2 + z_{max} > 1. \tag{12}$$

Proof. The statement follows straightforwardly from the result in Lemma 1 (by negation). □

In other words, the result in Proposition 1 states that if the GRU is outside of the fading memory regime, then Equation 12 must be satisfied. We can use this bound as a means to verify the contractivity conditions of the different models under consideration, and possibly as a strategy for the initialization of the weights. Note that the presence of the term $z_{max} \in (0, 1)$ in Equation 12 suggests that a network on the edge of a strictly contractive regime could be able to dynamically enter and exit such regime by means of the activations of the update gate.

6. Experimental analysis

In this section we describe in detail the experimental evaluation of the gated models introduced in Sections 4.1 and 4.2. In particular, we test our hypothesis (i.e., can gates provide advantages to reservoir computing models?) on a Natural Language Processing task which has been specifically chosen for the presence of long-term dependencies, and thus for its potential to clearly highlight the effect of the gating mechanisms. Note that the application of ESNs to Natural Language Processing tasks, whose data by their nature can often include long-term dependencies, has been quite limited: to the best of our knowledge there are only a few of such works [26,25,28,10].

6.1. TREC Dataset

We have chosen to empirically assess our model over real-world data exhibiting clear long-term dependencies. A good fit for a dataset exhibiting these characteristics is the TREC dataset for the Question Classification task[1] [19], which is a commonly used benchmark for evaluating Natural Language Processing systems. The TREC dataset deals with the task of classifying a number of input sentences, written in English, into one of 6 classes that indicate their broad topic (i.e. whether they ask about a person, a location, a number, a human being, a description or an entity). The output classes are represented in our models as one-hot encoded vectors. While the dataset also contains more detailed fine-grained classes, here we only focus on the 6 commonly used coarse-grained classes.

To support our model validation methodology, the dataset has been split in three folds: training, validation and test. The test fold is directly provided by the authors of the dataset [19] and contains 500 labeled questions. The other fold provided by the authors of the dataset, composed of 5452 labeled questions, was partially used for training and partially for validation. In fact, we have split this fold by the commonly used "80/20 rule", where 80% of the instances (chosen at random) are used for training and the other 20% for validation. This yields a training set of 4362 questions and a validation set of 1090 questions, with similar class distributions between the two sets (we did not perform an explicit stratification).

We have performed tokenization of the input questions, so that we could assign a word embedding to each token. In particular, we represented each token by a pretrained FastText embedding vector for the English language, with 300 dimensions [13]. Whenever

[1] `http://cogcomp.org/Data/QA/QC/`

Table 2. The total number of trainable parameters across the models is kept constant by controlling the size N_R of the state

	ESN	Gated ESN	Leaky ESN	Gated ESN RZ	GRU
Trainable params	19386	19386	19386	19386	19386
N_R	3230	3230	3230	29	20

a word that does not have a corresponding embedding in FastText is encountered, we use a random vector of the same shape instead. This vector is different for each missing word.

6.2. Experimental methodology

All models have been selected after a randomized hyperparameter search of 60 iterations by employing a hold-out validation set. Then, the selected models are retrained over the union of the training and validation sets, and their performance on the test set is measured and averaged across 10 trials, each with different random initializations of the parameters. Where needed (i.e., when employing gradient descent), the data in the training set has been shuffled.

To provide a fair and rigorous comparison, we made sure to keep the total number of trainable parameters uniform between all models by controlling the number of recurrent units, as shown in Table 2. In the literature this is a commonly used strategy for comparing RNNs, since forcing the same number of units for all models would lead to misleading results [8].

We point out that several architectural modifications can be introduced to significantly boost the predictive performance of an ESN on this task, as shown in our previous work [10]. For example, a bidirectional architecture [6,27] can easily help to capture most of the important information in this task [10]. However, in this work we deliberately consider only the simplest architectures in order to focus on the improvements brought by the gates.

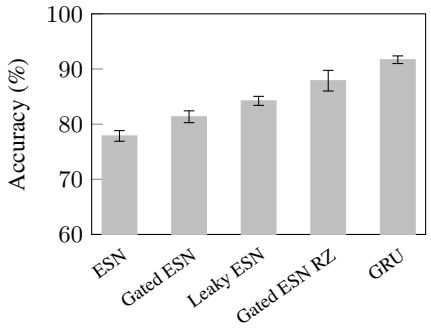
All experiments have been carried out on a NVIDIA Tesla V100 GPU.

Initialization – All reservoir matrices for the ESN, Gated ESN and Leaky ESN have been randomly initialized by sampling from $\mathcal{U}(-1, 1)$ and then rescaled according to the hyperparameters. The same initialization scheme has been used for Gated ESN RZ; here and in GRU, however, the entries for the matrices which are tuned by gradient descent are directly sampled from $\mathcal{U}\left(-1/N_R, 1/N_R\right)$ without further rescaling.

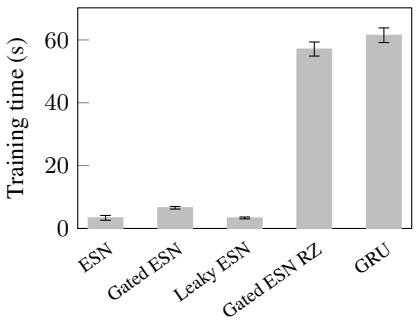
Training algorithm – The pure reservoir computing models (i.e., ESN, Gated ESN, and Leaky ESN) only involve the training of a linear output layer. For this reason, in such cases we employ ridge regression. The Gated ESN RZ and the GRU are trained by gradient descent. In Gated ESN RZ, we avoid the computation of the gradients associated to the matrices that are kept fixed ($\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$).

6.3. Results

In Fig. 3 we have reported the predictive accuracy and training time for the investigated variants of ESNs and for a fully trained GRU. Here we start our analysis with some



(a) Predictive performance. The highest accuracy is reached by the fully trained model, while the basic ESN is the baseline over which the improvements are built



(b) Training time. As soon as backpropagation is used (last two bars), even if only partially, the very fast training times of the pure reservoir computing models grow significantly.

Fig. 3. Test set results for the experimental comparison on the Question Classification task, with standard deviations. *Left:* Test set accuracy *Right:* Training times. The gates in Gated ESN are able to improve the predictive performance of the ESN, but training their parameters (Gated ESN RZ) seems necessary. Unfortunately, using backpropagation for this training process drastically increases the training time.

general considerations, leaving to a later moment more specific considerations on the results obtained by architectures with untrained or trained gates.The first three models in Fig. 3a have a completely untrained reservoir and as such they all exhibit the same number of recurrent units *and* trainable parameters. Nonetheless, a distinct difference in predictive accuracy can be observed between these models, which hints at the importance of more advanced reservoir state transition functions. The importance of reservoir computing models is clearly highlighted when comparing the time required for training the parameters (see Fig. 3b): the pure reservoir computing models (the first three, i.e., ESN, Gated ESN, and Leaky ESN) provide a remarkable efficiency advantage with respect to the other two (Gated ESN RZ and GRU).

As expected, the best performing model is the GRU, in which all the parameters are trained by gradient descent and backpropagation through time. Also as expected, the basic ESN displays the lowest level of accuracy. While the number of trainable parameters is the same in these two models, their striking difference in accuracy is due to their different biases with respect to the data. In the task under consideration, the most important input words for producing a correct prediction are often located at the beginning of the sentences: due to the fading memory property of the ESN, their contribution has a very low influence on the final states of the network, which are those observed by the classifier [10].

Untrained gates dynamics – For what concerns the improvements in predictive performance brought by the Gated ESN architecture with untrained gates, it can be observed from Fig. 3 that there is a significant increase in accuracy with respect to an ESN. However, the simpler model which uses leaky-integrator neurons (Leaky ESN) produces better results. To better understand this trend, we take a step further in our analysis and in Ta-

Table 3. Statistics about the activations of the gates for Gated ESN. Mean, standard deviation and maximum value are computed by aggregating across both the unit dimension and the time dimension

$\mathbf{r}(t)$			$\mathbf{z}(t)$		
mean	std. dev	max	mean	std. dev	max
0.4997	0.0761	0.9993	0.5000	0.0108	0.6488

Table 4. Average spectral radius and norm of the reservoir matrices after initialization and training (where applicable). For different models, the matrices can have different size (see Table 2). Also note that the value of $a = 0.04$ has been chosen by model selection in $\mathcal{U}(0, 1)$

	State		Reset gate		Update gate	
Model	$\rho(\hat{\mathbf{W}})$	$\|\hat{\mathbf{W}}\|$	$\rho(\hat{\mathbf{W}}^r)$	$\|\hat{\mathbf{W}}^r\|$	$\rho(\hat{\mathbf{W}}^z)$	$\|\hat{\mathbf{W}}^z\|$
ESN	1.35	76.38	–	–	–	–
Gated ESN	5.30	297.92	0.28	15.70	0.06	3.37
Leaky ESN ($a = 0.04$)	0.02	1.16	–	–	–	–
Gated ESN RZ	7.62	9.18	8.16	14.57	7.59	25.31
GRU	4.52	10.06	1.69	7.86	2.78	11.91

ble 3 we report the main statistics about the activations of the gates over time. According to the measurements in Table 3 the gates (and especially the *update* gate) are not being fully exploited. In fact, from Table 3 it can be inferred that the matrices in the update gate have been rescaled by the procedure of model selection so that it behaves roughly like a constant, in this case $\mathbf{z}(t) \approx 0.5 \cdot \mathbf{1}\ \forall t$. This can be deduced by observing the low standard deviations for the gate activations. Thus, the behavior of the Gated ESN is actually approximating on average the one of a Leaky ESN. In the upper part of Table 4 we have reported the average norm of the recurrent matrices for the different models. It can be observed that Gated ESN tends to values of $\|\hat{\mathbf{W}}\|$ that are significantly higher than the other untrained models (ESN and Leaky ESN). This by itself brings Gated ESN well within the bound provided by Proposition 1, which gives a necessary condition for the non-contractivity of the state transition function and thus allows state dynamics that are outside of the fading memory regime which is typical of ESNs. The values of $\|\hat{\mathbf{W}}^r\|$ and $\|\hat{\mathbf{W}}^z\|$ are made irrelevant for the bound by the higher-than-unity value of $\|\hat{\mathbf{W}}\|$.

For completeness, in Table 4 we also report the average spectral radius of the matrices, as it represents a reference parameter for the initialization of the recurrent matrices in reservoir computing-based networks. Most networks (all except the Leaky ESN) exhibit a value of $\rho(\hat{\mathbf{W}}) > 1$, which is noteworthy because for basic ESNs, $\rho(\hat{\mathbf{W}}) < 1$ represents a traditionally used bound for the initialization of ESNs in practical applications (even though it is not a sufficient condition for the ESP). This hints to the fact that the networks are trying to escape from the fading memory regime implied by their contractive state transition function, but in the case of the non-gated models it is impossible to dynamically do so.

Trained gates dynamics – While using gates with untrained parameters appears to be effective only to a limited extent, applying learning as in *Gated ESN RZ* drastically im-

proves the predictive performance (Fig. 3a). In particular, even though the training is only applied to the parameters in the gates and the dynamics are still mainly determined by random weights, the accuracy of *Gated ESN RZ* dominates over all pure reservoir computing models. Moreover, in this case a relatively small size of the reservoir is sufficient to obtain good performance (29 units instead of 3230, as indicated in Table 2). However, from the measurements reported in Fig. 3b it is clear that the introduction of backpropagation through time also causes a severe increase in the training time. This makes the approach of training the gates via the specific algorithm of backpropagation through time unappealing in practice, as the efficiency advantages coming from the reservoir computing approach are vanishing.

Regarding the average matrix norms reported in the bottom part of Table 4, notice how even though matrix $\hat{\mathbf{W}}$ for the GRU is trained, it ends up having a very similar norm to the one of Gated ESN RZ, in which $\hat{\mathbf{W}}$ is untrained (10.06 for GRU versus 9.18 for Gated ESN RZ). In addition, we can observe how both models respect the bound from Proposition 1, i.e. the necessary condition for the non-contractivity of the state transition function. This does not mean that the state transition function is never contractive. On the contrary, it is likely that the recurrent dynamics are mainly contractive, which is needed in order to provide meaningful data representations to the readout. However, in the case of Gated ESN RZ, the untrained dynamics of the reservoir are able to occasionally exit the contractive (i.e., fading memory) regime thanks to the activations of the gates, thus allowing the network to relax its Markovian bias and to increase its memory capacity. The same happens in GRU, even though by means of a fully adapted state transition function.

7. Discussion

We have shown how there exist tasks with particular characteristics such that simple ESNs, despite their exceptional efficiency, do not compete in accuracy with the more popular and expensive alternatives such as GRU. In such cases, a state transition function that is able to give different weights to different parts of the input can have an important impact on the predictive performance of the model.

Equipping the reservoir of the ESN with gating mechanisms while maintaining its weights untrained does not appear to be sufficient for a meaningful increase in predictive performance. What seems to be effective, instead, is maintaining a mostly untrained dynamics but injecting a training signal into the gates. The benefits of such approach are twofold. On one hand, training only the gates allows to employ much smaller reservoirs than what would be necessary in an ESN, and a reservoir of a given size can also generalize better with respect to longer sequences.[2] On the other hand, the approach has the potential of reducing the training time compared to what would be required for a GRU.

Currently, due to its relatively low efficiency the algorithm of backpropagation through time is not suited to train the parameters of the gates. However, the constrained model that we propose under the name of Gated ESN RZ has the potential for allowing the use of less conventional training algorithms that may be more efficient in this case. One of the problems with the use of backpropagation through time for Gated ESN RZ is that, for all time steps, the gradients need to flow through $\mathbf{h}(t)$ and $\mathbf{x}(t)$ anyway, regardless of the

[2] For example, in principle the network is allowed to discard any irrelevant time step in the input sequence without alterations of the reservoir state.

fact that the parameters $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$ are *not* trained. This adds a significant cost to the computation of the gradients, especially on larger reservoirs.

Considering the potential impact of gated reservoir computing, innovative methods for training the gates are needed. One likely candidate is represented by the class of local algorithms which could *bypass* the chain of gradients by injecting a training signal directly into the gates, a concept similar to the direct feedback alignment in feedforward networks [24]. An instance of such algorithms is the biologically inspired Hebbian learning, which must be modulated by an error signal to allow supervised learning. However, replacing backpropagation through time requires to employ alternative techniques for addressing the problem of *credit assignment*, or *distal reward* [23]: how did each individual synapse contribute to the final prediction of the model, especially in case of long delays before the error signal is available? The impact of this problem is clearly evident in the case of classification problems, in which the error signal is only available at the end of each sequence. For addressing the credit assignment problem, variants of biologically motivated mechanisms of *eligibility traces* [15] can be used, which can also be compatible with approximations of the gradients that would be computed by backpropagation through time [4,3].

In the literature, reward-modulated Hebbian learning has already been successfully employed for training all parameters of a recurrent neural network [22]. However, in order to exploit the untrained discrimination capabilities of the reservoir computing approach we see as a promising method that of not training the whole network, but instead only steering the trajectories of the state of the reservoir through the use of gate or gate-like mechanisms trained by variants of the above-mentioned approach.

Efficient alternatives for training RNNs are needed: the suggested approach of employing both untrained and trained dynamics could represent a tool for allowing the already efficient ESNs to effectively tackle problems which today represent a hurdle due to the presence of long-term dependencies.

8. Conclusion

In this paper we have discussed the introduction of gated mechanisms in the architecture of reservoir computing neural networks. We have started by presenting the limitations of reservoir computing models such as ESNs. Their fading memory characteristic, which is a strength in certain situations, can become a weakness when dealing with data presenting long-term dependencies. We have thus proposed a reservoir computing model, Gated ESN, and its variant Gated ESN RZ, for overcoming those limitations by the use of gating mechanisms.

To allow the Gated ESN to escape a strict fading memory regime, we have derived a general bound that links the weight matrices of all GRU-based gated models (GRU, Gated ESN, and Gated ESN RZ) to their ability to escape such regime. This gives a means of initializing or verifying these networks for the desired behaviour.

We have performed an experimental comparison between the different models under consideration by testing the generalization performance on a Question Classification task that was chosen for its suitedness to highlight the effect of the gates. We have discovered that gates *can* indeed provide advantages even to reservoir computing models. In addition, we have shown that the use of backpropagation through time drastically increases the

time required for training. We have then verified that the experimental results match the theoretical bound that we have derived.

While the results of this work provide insights about gated reservoir computing, we have also critically discussed the reasons why we believe a pure gated reservoir computing model to be ineffective in practice. We have then suggested directions to produce efficient gated models that join reservoir computing techniques with trained gate dynamics, with an important focus on local training algorithms.

Looking ahead, we believe that the key for efficient and effective RNN models is to be found in the form of an interplay between a suitable gated architecture and a suitable local training algorithm.

Acknowledgments. This work has been partially supported by the European Union's Horizon 2020 Research and Innovation program, under project TEACHING (Grant agreement ID: 871385), URL `https://www.teaching-h2020.eu`, and by the project BrAID under the Bando Ricerca Salute 2018 – Regional public call for research and development projects aimed at supporting clinical and organisational innovation processes of the Regional Health Service – Regione Toscana.

References

1. Babinec, S., Pospichal, J.: Gating echo state neural networks for time series forecasting. In: ICONIP (1). Lecture Notes in Computer Science, vol. 5506, pp. 200–207. Springer (2008)
2. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R.A., Maass, W.: Long short-term memory and learning-to-learn in networks of spiking neurons. In: NeurIPS. pp. 795–805 (2018)
3. Bellec, G., Scherr, F., Hajek, E., Salaj, D., Subramoney, A., Legenstein, R.A., Maass, W.: Eligibility traces provide a data-inspired alternative to backpropagation through time. In: Neuro AI Workshop, NeurIPS (2019)
4. Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., Maass, W.: A solution to the learning dilemma for recurrent networks of spiking neurons. bioRxiv p. 738385 (2019)
5. Bengio, Y., Simard, P.Y., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Networks 5(2), 157–166 (1994)
6. Bianchi, F.M., Scardapane, S., Løkse, S., Jenssen, R.: Bidirectional deep-readout echo state networks. In: ESANN (2018)
7. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014. pp. 1724–1734 (2014)
8. Collins, J., Sohl-Dickstein, J., Sussillo, D.: Capacity and trainability in recurrent neural networks. In: ICLR (Poster). OpenReview.net (2017)
9. Di Sarli, D., Gallicchio, C., Micheli, A.: Gated echo state networks: a preliminary study. In: INISTA. pp. 1–5. IEEE (2020)
10. Di Sarli, D., Gallicchio, C., Micheli, A.: Text classification by untrained sentence embeddings. Intelligenza Artificiale 14(2), 245–259 (2020)
11. Gallicchio, C., Micheli, A.: Architectural and Markovian factors of echo state networks. Neural Networks 24(5), 440–456 (2011)
12. Gonon, L., Ortega, J.P.: Fading memory echo state networks are universal. Neural Networks (2021)
13. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)

14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)
15. Izhikevich, E.M.: Solving the distal reward problem through linkage of stdp and dopamine signaling. Cerebral cortex 17(10), 2443–2452 (2007)
16. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks – with an erratum note'. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report (2001)
17. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science 304(5667), 78–80 (2004)
18. Jaeger, H., Lukosevicius, M., Popovici, D., Siewert, U.: Optimization and applications of echo state networks with leaky-integrator neurons. Neural Networks 20(3), 335–352 (2007)
19. Li, X., Roth, D.: Learning question classifiers. In: 19th International Conference on Computational Linguistics, COLING 2002 (2002)
20. Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. CoRR 1506.00019 (2015)
21. Lukosevicius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. 3(3), 127–149 (2009)
22. Miconi, T.: Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. Elife 6, e20899 (2017)
23. Minsky, M.: Steps toward artificial intelligence. Proceedings of the IRE 49(1), 8–30 (1961)
24. Nøkland, A.: Direct feedback alignment provides learning in deep neural networks. In: NIPS. pp. 1037–1045 (2016)
25. Popov, A., Koprinkova-Hristova, P., Simov, K., Osenova, P.: Echo state vs. lstm networks for word sense disambiguation. In: International Conference on Artificial Neural Networks. pp. 94–109. Springer (2019)
26. Ramamurthy, R., Stenzel, R., Sifa, R., Ladi, A., Bauckhage, C.: Echo state networks for named entity recognition. In: ICANN (Workshop). Lecture Notes in Computer Science, vol. 11731, pp. 110–120. Springer (2019)
27. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. 45(11), 2673–2681 (1997)
28. Simov, K.I., Koprinkova-Hristova, P.D., Popov, A., Osenova, P.: Word embeddings improvement via echo state networks. In: INISTA. pp. 1–6. IEEE (2019)
29. Subramoney, A., Scherr, F., Maass, W.: Reservoirs learn to learn. CoRR abs/1909.07486 (2019)
30. Tiňo, P., Hammer, B., Bodén, M.: Markovian bias of neural-based architectures with feedback connections. In: Perspectives of Neural-Symbolic Integration, Studies in Computational Intelligence, vol. 77, pp. 95–133. Springer (2007)
31. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Networks 20(3), 391–403 (2007)
32. Wang, X., Jin, Y., Hao, K.: A gated recurrent unit based echo state network. In: IJCNN. pp. 1–7. IEEE (2020)
33. Yildiz, I.B., Jaeger, H., Kiebel, S.J.: Re-visiting the echo state property. Neural Networks 35, 1–9 (2012)

Daniele Di Sarli has conducted research on Recurrent Neural Networks and Reservoir Computing at the Department of Computer Science, University of Pisa, Italy. The main focus of his research concerns the study of the effectiveness of Reservoir Computing approaches.

Claudio Gallicchio is an Assistant Professor of Machine Learning at the Department of Computer Science of the University of Pisa, Italy. His research is based on the fusion

of concepts from Deep Learning, Recurrent Neural Networks, and Randomized Neural Systems.

Alessio Micheli is an Associate Professor at the Department of Computer Science, University of Pisa, Italy. His main research lines are in the field of Machine Learning and Neural Networks, with a pioneering research activity since the end of 90's for learning in structured domains (sequence, tree and graph data).

Received: February 18, 2021; Accepted: August 31, 2021.