LETTER Communicated by Herbert Jaeger

A Robust Model of Gated Working Memory

Anthony Strock
Anthony.Strock@inria.fr
Xavier Hinaut
Xavier.Hinaut@inria.fr
Nicolas P. Rougier
Nicolas.Rougier@inria.fr
Inria Bordeaux Sud-Ouest, 33405 Talence Cedex, France; LaBRI, Université de Bordeaux, Institut Polytechnique de Bordeaux, Centre National de la Recherche Scientifique, 33405 Talence Cedex, France; and Institut des Maladies Neurodégénératives, Université de Bordeaux, Centre National de la Recherche Scientifique, 33076 Cedex, Bordeaux, France

Gated working memory is defined as the capacity of holding arbitrary information at any time in order to be used at a later time. Based on electrophysiological recordings, several computational models have tackled the problem using dedicated and explicit mechanisms. We propose instead to consider an implicit mechanism based on a random recurrent neural network. We introduce a robust yet simple reservoir model of gated working memory with instantaneous updates. The model is able to store an arbitrary real value at random time over an extended period of time. The dynamics of the model is a line attractor that learns to exploit reentry and a nonlinearity during the training phase using only a few representative values. A deeper study of the model shows that there is actually a large range of hyperparameters for which the results hold (e.g., number of neurons, sparsity, global weight scaling) such that any large enough population, mixing excitatory and inhibitory neurons, can quickly learn to realize such gated working memory. In a nutshell, with a minimal set of hypotheses, we show that we can have a robust model of working memory. This suggests this property could be an implicit property of any random population, that can be acquired through learning. Furthermore, considering working memory to be a physically open but functionally closed system, we give account on some counterintuitive electrophysiological recordings.

X.H. and N.R. contributed equally.

Neural Computation **32, 153–181** (2020)
https://doi.org/10.1162/neco_a_01249

1 Introduction

The prefrontal cortex (PFC), noteworthy for its highly recurrent connections (Goldman-Rakic, 1987), is involved in many high-level capabilities, such as decision making (Bechara, Damasio, Tranel, & Anderson, 1998), working memory (Goldman-Rakic, 1987), goal-directed behavior (Miller & Cohen, 2001), and temporal organization and reasoning (Fuster, 2001). In this letter, we are more specifically interested in gated working memory (O'Reilly & Frank, 2006), defined as the capacity of holding arbitrary information at a given random time t_0 so as to be accessible at a later random time t_1 (see Figure 1). Between times t_0 and t_1, we make no assumption about the inner mechanisms of the working memory. The only measures we are interested in are the precision of the output (compared to the initial information) and the maximal delay during which this information can be accessed within a given precision range. One obvious and immediate solution to the task is to make an explicit copy (inside the memory) of the information at time t_0 and hold it unchanged until it is read at time t_1, much like a computer program variable that is first assigned a value in order to be read later. Such a solution can be easily characterized by a fixed pattern of sustained activities inside the memory. This is precisely what led researchers to search for such sustained activity inside the frontal cortex (Funahashi, 2017; Constantinidis et al., 2018), where an important part of our working memory capacities is believed to be located. Romo, Brody, Hernández, and Lemus (1999) have shown that PFC neurons of nonhuman primates can maintain information about a stimulus for several seconds. Their firing rate was correlated with the coding of a specific dimension (frequency) of the stimulus maintained in memory. However, when Machens, Romo, and Brody (2010) later reanalyzed the data of this experiment, they showed that the stimulus was actually encoded over a subpopulation using a distributed representation. Similarly, when Rigotti et al. (2013) analyzed single-neuron activity recorded in the lateral PFC of monkeys performing complex cognitive tasks, they found several neurons displaying task-related activity. Once they discarded all the neurons that were displaying task-related activity, they were still able to decode task information with a linear decoder. They proposed that the PFC hosts high-dimensional linear and nonlinear mixed-selectivity activity. The question is thus, if working memory is not encoded in the sustained activity, what can the alternatives be?

Before answering that question, we first characterize the type and the properties of information we consider before defining what it means to access the information.

The type of information that can be stored inside working memory has been characterized using different cognitive tasks—for example, delayed matching-to-sample (DMTS), the N-back task, or the Wisconsin Card Sorting Task (WCST). From these different tasks, we can assume that virtually any kind of information, be it an auditory or visual stimulus, textual or

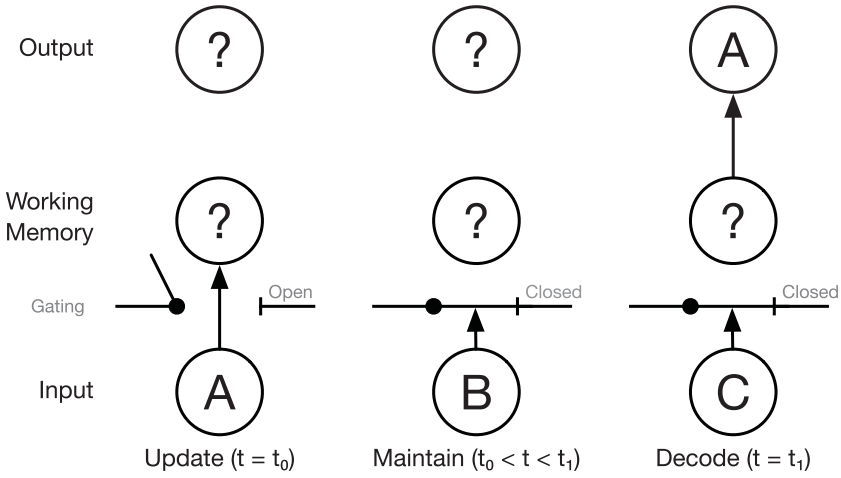


Figure 1: Gated working memory is defined as the capacity of holding arbitrary information at a given random time t_0 so as to be accessible at a later random time t_1. Between times t_0 and t_1, we make no assumption about the activity inside the working memory. Note that a closed gate does not mean external activities cannot enter the memory.

verbal instruction, implicit or explicit cue, can be memorized and processed inside the working memory. From a computational point of view, this can be abstracted into a set of categorical, discrete, or continuous values. In this work, we are interested only in the most general case, the continuous one, which can be reduced to a single scalar value. The question we want to address is how a neural population can gate and maintain an arbitrary (e.g., random) value, in spite of noise and distractors, such that it can be decoded nonambiguously at a later time.

To answer this question, we can search the extensive literature on computational models of working memory that have been extensively reviewed by Durstewitz, Seamans, and Sejnowski (2000), Compte (2006) and, more recently, Barak and Tsodyks (2014). More specifically, Compte (2006) explains that the retention of information is often associated with attractors. In the simplest case, a continuous scalar information can be identified with a position on a line attractor. However, this kind of memory exhibits stability issues. Unlike a point attractor (i.e., a stable fixed point), a line attractor is marginally stable such that it cannot be robust against all small perturbations. Such a line attractor can be stable against orthogonal perturbations but not against colinear perturbations (i.e., perturbations along the line). Furthermore, the design (and numerical implementation) of a line attractor is tricky because even small imperfections (e.g., numerical errors) can lead to instability. Nevertheless, several models can overcome these limitations.

This is the case for the theoretical model by Amari (1977), who proved that a local excitation could persist in the absence of stimuli, in the form of a localized bump of activity in an homogeneous and isotropic neural field model, using long-range inhibition and short-range excitation. This model represents de facto a spatial attractor formed by a collection of localized bumps of activity over the plane. A few decades later, Compte (2000) showed that the same lasting bump property can also be achieved using leaky integrate-and-fire neurons arranged on a ring, with short-range excitation and constant inhibition (ring bump attractor). This model has since then been extended (Edin et al., 2009; Wei, Wang, & Wang, 2012) with the characterization of the conditions allowing simultaneous bumps of activity. This would explain multi-item memorization, where each bump represents different information that is maintained simultaneously with the other bumps. Similarly, Bouchacourt and Buschman (2019) proposed handling multi-item memorization by duplicating the bump attractor model. They explicitly limited the number of items to be maintained in memory through the interaction of the different bump attractor models (using a random layer of neurons).

If all of these models can cope with the memorization of graded information, this information is precisely localized in the bumps of activity and corresponds to sustained activity. Such patterns of activity have been identified in several cerebral structures—for example, head direction cells (Zhang, 1996) in mammals and superior colliculus (Gandhi & Katnani, 2011) in primates—but it is not yet clear to what extent this can give an account of a general working memory mechanism. Such sustained activity is also present in the model of Koulakov, Raghavachari, Kepecs, and Lisman (2002), who consider a population of bistable units that encodes (quasi) graded information using distributed encoding (percentage of units in high state). This solves both the robustness and stability issue of the line attractor by virtue of discretization. Finally, some authors (Zipser, Kehoe, Littlewort, & Fuster, 1993; Lim & Goldman, 2013) consider the encoding of the value to be correlated with the firing rate of a neuron or of a group of neurons. This is the case for the model proposed by Lim and Goldman (2013), who obtain stability of the firing rate by adding negative derivative self-feedback (hence, artificially increasing the time constant of neurons). They show how such a mechanism can be implemented by the interaction of two populations of excitatory and inhibitory neurons evolving at different timescales. However, independent of the encoding of the graded value, most of model authors are interested in characterizing the mechanism responsible for the maintenance property. They tend to consider the memory as an isolated system, not prone to external perturbations, with the noticeable exception of the model by Zipser et al. (1993), which is constantly fed by an input.

In this work, we consider working memory to be an open system under the constant influence of external activities (even when the gate is closed). Thus, we cannot rely on a dynamical system that hosts a line attractor in the absence of inputs. We have to design an input-dependent dynamical system

that is robust against all kinds of perturbations (input, internal, output feedback). First, we formalize a set of tasks that we used to study the features and performance of the different models we have considered. Then, we introduce a minimal model that will help us explain the mechanism needed for a more general model. For this general one, we consider a particular instance of reservoir: an echo state network (ESN; Jaeger, 2001). The analysis of this model will allow us to show that reservoir activity is characterized by a combination of both sustained and transient activities. Moreover, we show that none of these activities are critical for the decoding of the correct output. Finally, we show that in the absence of input, the dynamics of the model implement a segment attractor (i.e., a line attractor with bounding values).

2 Methods

In this section, we formalize and extend the gated working memory task that we described in section 1 (see Figure 1). We consider four tasks that will illustrate the generic capacity of the reservoir model to store continuous values or a discrete set of values. The first three are variations of the working memory (WM) task for continuous values with various numbers of input values (n-value) and gating WM units (n-gate). The last task includes nonlinear computation (digit recognition from pixels) in addition to the gating task for discrete values.

2.1 The n-Value p-Gate Scalar Working Memory Tasks. In the one-value, one-gate scalar version of the task, we consider at time $t \in \mathbb{N}$ an input signal $V(t)$ in $[-1, +1]$, an input trigger $T(t)$ in $\{0, 1\}$, and an output $M(t)$ that is defined as the value of $V(t^*)$ where t^* is the most recent time such that $T(t^*) = 1$ (see Figure 2A). This can be written as

$$M(t) = V(t^*) \text{ with } t^* = \max_{0 \leq t' \leq t} (t' \mid T(t') = 1). \tag{2.1}$$

Said differently, the output is the value of the input when the gate was open for the last time. Note that this can also be rewritten as a simple select operator between the input V and the output M:

$$M(t) = T(t)V(t) + (1 - T(t))M(t - 1) \tag{2.2}$$

In the one-value, three-gate version of the task, we consider at time $t \in \mathbb{N}$ an input signal $V(t)$ in $[-1, +1]$, three input triggers $T_{\{1,2,3\}}(t)$ in $\{0, 1\}$, and three outputs $M_{\{1,2,3\}}(t)$ that are respectively defined as the value of $V(t^*)$ where t^* is the most recent time such that $T_{\{1,2,3\}}(t^*) = 1$ (see Figure 2B). This can be written as

$$M_i(t) = M_i(t^*) \text{ with } t^* = \max_{0 \leq t' \leq t} (t' \mid T_i(t') = 1), i \in \{1, 2, 3\}. \tag{2.3}$$

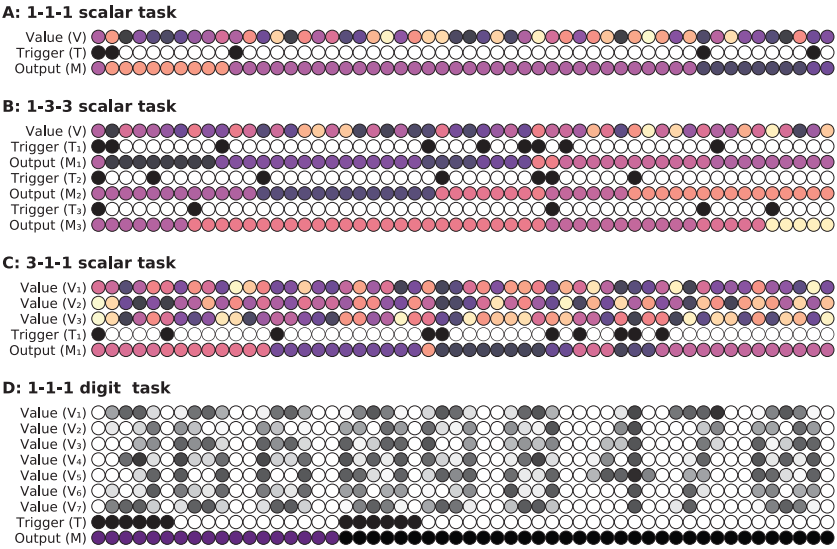


Figure 2: Working memory tasks. Each column represents a time step (time increases from left to right). Colored discs represent activity in the input (V or V_i and T or T_i) and the output (M or M_i). (A) Gated working memory task with one gate. (B) Gated working memory task with three gates. (C) Gated working memory task with one gate but three inputs. To solve the task, it is necessary to ignore the two irrelevant inputs. (D) Gated working memory task with one gate where the scalar input V has been replaced by a streamed input of bitmap digits. To solve the task, it is necessary to recognize the digits and transform them into a normalized value.

In the three-value, one-gate scalar version of the task, we consider at time $t \in \mathbb{N}$ three input signals $V_{\{1,2,3\}}(t)$ in $[-1, +1]$, an input trigger $T(t)$ in $\{0, 1\}$, and an output $M(t)$ that is defined as the value of $V_1(t^*)$ where t^* is the most recent time such that $T(t^*) = 1$ (see Figure 2C). This can be written as

$$M(t) = V_1(t^*) \text{ with } t^* = \max_{0 \leq t' \leq t} (t' \mid T(t') = 1). \tag{2.4}$$

This can be easily generalized to an n-value, p-gate scalar task with n input signals, p input triggers, and p outputs. Only the first input signal and the t input triggers determine the p outputs. The other $n - 1$ input signals are additional inputs irrelevant to the task.

2.2 The Digit One-Value, One-Gate Working Memory Task. In the digit version of the one-value, one-gate working memory task, we define a slightly more complex version of the input V by considering a bitmap

representation of it (see Figure 2D). Using an 11 points monotype font (Inconsolata Regular),[1] we draw a gray-scaled bitmap representation of a sequence of random digits (0 to 9), each digit being of size 6×7 pixels (after having cropped top and bottom empty lines) and the trigger signal being expanded to the width of a glyph. The output is defined as a discrete and normalized value. There is no possible linear interpolation between the different inputs, as it was the case for the scalar version. Formally, we can define the output as

$$\begin{aligned} M(t) &= f(V_{i\in[1,7]}(t^*), V_{i\in[1,7]}(t^*-1), \ldots, V_{i\in[1,7]}(t^*-5)) \text{ with } t^* \\ &= \max_{0\leq t'\leq t}(t' \mid T(t') = 1), \end{aligned} \tag{2.5}$$

with f a function that maps the representation of a digit to a normalized value. Since there are 10 values, the digit n is associated with $\frac{n}{10}$. This function has to be learned by the model in order to solve the task. It would have been possible to use the MNIST database (Schaetti, Salomon, & Couturier, 2016) instead of a regular font, but this would also have made the task more complex and the training period much longer, because a digit is processed only when a trigger is present. If we consider, for example, a sequence of 25,000 digits and a trigger probability of 0.01, this represents (in average) 250 triggers for the entire sequence and, consequently, only 25 presentations per digit. In comparison, the MNIST training data set has 60,000 digits, which would have required as many triggers and a hundred times more digits.

2.3 The Minimal Model. It is possible to define a minimal degenerated reservoir model (if we consider X_1, X_2, and X_3 to be the reservoir) that takes explicit advantage of the nonlinearity to perform the gating mechanism, as shown in Figure 3. There is no learning in this model. It is parameterized by two values, a (large enough) and b (small enough), and it is fully defined by the following set of equations:

$$\begin{aligned} X_1[n+1] &= \tanh(b\,V[n]), \\ X_2[n+1] &= \tanh(b\,V[n] + a\,T[n]), \\ X_3[n+1] &= \tanh(b\,M[n] + a\,T[n]), \\ M[n+1] &= (X_1[n+1] - X_2[n+1] + X_3[n+1])/b. \end{aligned} \tag{2.6}$$

In order to understand how this minimal model works, we can write the output $M[n]$ relative to the value of the trigger $T[n]$, which lies in $\{0, 1\}$:

[1] The Inconsolata font is available from https://www.levien.com/type/myfonts/inconsolata.html.

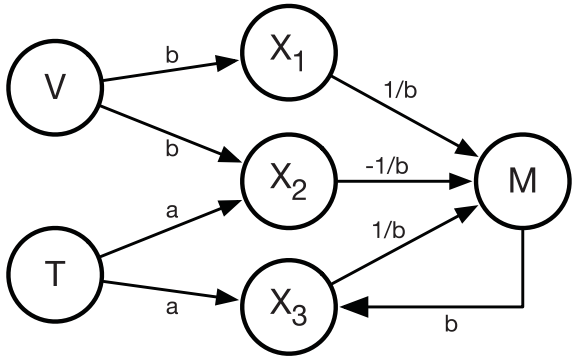


Figure 3: A minimal gated working memory model is able to solve the one-value, one-gate working memory task by taking advantage of the nonlinearity and asymptotic behavior of tanh units. Performance is controlled with parameters a and b. This architecture can be generalized to the n-value, p-gate task by adding more than one reservoir unit (corresponding to X_3) for each supplementary couple of trigger and output.

$$\text{If } T[n] = 0 \begin{cases} X_1[n+1] = \tanh(b\,V[n]) \\ X_2[n+1] = \tanh(b\,V[n]) \\ X_3[n+1] = \tanh(b\,M[n]) \\ M[n+1] = \tanh(b\,M[n])/b \end{cases} . \tag{2.7}$$

When $T[n] = 0$, if we assign b a small value (e.g., $b = 10^{-3}$) and consider that $\tanh(x) \underset{x\to 0}{\sim} x$, from equation 2.7, we have $M[n+1] \approx M[n]$:

$$\text{If } T[n] = 1 \begin{cases} X_1[n+1] = \tanh(b\,V[n]) \\ X_2[n+1] = \tanh(b\,V[n] + a) \\ X_3[n+1] = \tanh(b\,M[n] + a) \\ M[n+1] = \tanh(b\,V[n])/b - (\tanh(b\,V[n] + a) \\ \qquad\qquad - \tanh(b\,M[n] + a))/b \end{cases} . \tag{2.8}$$

When $T[n] = 1$, if we assign a to a large value (e.g., $a = 1000$) and considering that b is small and that $\lim_{x\to\infty} \tanh(x) = 1$, we have $\tanh(b\,V[n] + a) \approx \tanh(b\,M[n] + a) \approx 1$. From equation 2.8, we then have $M[n+1] \approx \tanh(b\,V[n])/b \approx V[n]$.

Consequently, the trigger $T[n]$ fully dictates the output. When $T[n] = 0$, the output $M[n+1]$ is unchanged and corresponds to the current memory ($M[n]$). When $T[n] = 1$, the output $M[n+1]$ is assigned the value $V[n]$ of the input. We think this model represents the minimal model that is able to solve the gating working memory task using such simple neurons (with

tanh activation function). By taking advantage of the linear regime around 0 and the asymptotic behavior around infinity, this model gracefully solves the task using only two parameters (a and b). However, we have no formal proof that a model having only two neurons in the reservoir part cannot solve the task.

Note that this minimal model turns out to be similar to the memory cell of a gated recurrent unit (Cho et al., 2014) (GRU) without its reset gate,[2] but using only simple tanh neurons, in comparison to handcrafted LSTM-like cells. Without the reset gate, the dynamics of a GRU cell can be simplified to

$$h[n+1] = (1 - z[n+1])h[n] + z[n+1]x[n+1], \tag{2.9}$$

where h is the state of the memory cell, z is the update gate neuron, and x is an input neuron. By using functional equations of hyperbolic functions and classical order 1 Taylor expansions and for the sake of simplicity, by replacing all the $o_{b\to 0}(b)$ by $o(b)$, we can obtain a similar equation in the minimal model:

$$\begin{aligned}
&M[n+1] \\
&= \frac{\tanh(b\,M[n] + a\,T[n]) - \tanh(b\,V[n] + a\,T[n]) + \tanh(b\,V[n])}{b} \\
&= \frac{\tanh(b\,M[n] + a\,T[n]) - \tanh(b\,V[n] + a\,T[n])}{b} + \frac{\tanh(b\,V[n])}{b} \\
&= \frac{\sinh(b\,(M[n] - V[n]))}{b(\cosh(b\,M[n] + a\,T[n])\cosh(b\,V[n] + a\,T[n]))} + \frac{\tanh(b\,V[n])}{b} \\
&= \frac{\sinh(b\,(M[n] - V[n]))}{b}\frac{1}{\cosh(b\,M[n] + a\,T[n])\cosh(b\,V[n] + a\,T[n])} \\
&\quad + \frac{\tanh(b\,V[n])}{b} \\
&= \frac{b\,(M[n] - V[n]) + o(b)}{b}\left(\frac{1}{\cosh(a\,T[n])^2} + o(1)\right) + \frac{b\,V[n] + o(b)}{b} \\
&= \frac{(M[n] - V[n])}{\cosh(a\,T[n])^2} + V[n] + o(1) \\
&= (M[n] - V[n])(1 - \tanh(a\,T[n])^2) + V[n] + o(1) \\
&= (1 - \tanh(a\,T[n])^2)M[n] + \tanh(a\,T[n])^2 V[n] + o(1).
\end{aligned}$$

[2] We can find a similar LSTM variant in Greff, Srivastava, Koutnik, Steunebrink, and Schmidhuber (2017), the coupled input and forget gate.

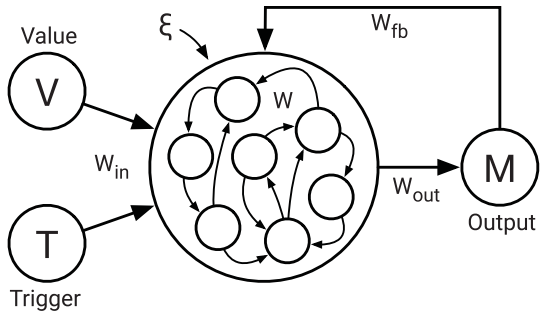


Figure 4: The reservoir model receives a random signal V in $[-1, +1]$ and a trigger signal T in $\{0, 1\}$. The reservoir is made of nonlinear units (tanh), and the output M is fed back to the reservoir at each iteration. For n-gate, p-value task, we use n triggers, p values, and n outputs. For a generic approach, we use the notation u for the input, x for the reservoir, and y for the output, independently of the task.

If b is small enough, we have $M[n+1] \approx (1 - \tanh(a\,T[n])^2)M[n] + \tanh(a\,T[n])^2 V[n]$; consequently, T acts as an update gate for M. The final form that is equivalent to the GRU without reset gate (as in equation 2.9) is given by

$$M[n+1] = (1 - z[n+1])M[n] + z[n+1]V[n+1], \tag{2.10}$$

with $z[n] \approx (1 - \tanh(a\,T[n])^2)$.

2.4 The Reservoir Model. We consider an echo state network (Jaeger, 2001; see Figure 4) with leaky neurons, and feedback from readout units to the reservoir, which is described by the following update equations,

$$\begin{aligned} x[n] &= (1-\alpha)x[n-1] \\ &\quad + \alpha \tanh\left(W_{in}u[n] + W(x[n-1] + \xi) + W_{fb}(y[n-1])\right) \\ y[n] &= W_{out}x[n], \end{aligned} \tag{2.11}$$

where $u[n]$, $x[n]$, and $y[n]$ are, respectively, the input, the reservoir, and the output at time n. W, W_{in}, W_{fb}, and W_{out} are, respectively, the recurrent, the input, the feedback, and the output weight matrix, and α is the leaky rate. ξ is a uniform white noise term added to the reservoir: ξ is independent for each neuron. During initialization, matrices W_{in} and W_{fb} are sampled randomly and uniformly between -1 and $+1$ and multiplied by the input scaling factor and the feedback scaling factor, respectively. Matrix W is sampled randomly between -1 and $+1$, and we ensure that the matrix enforces

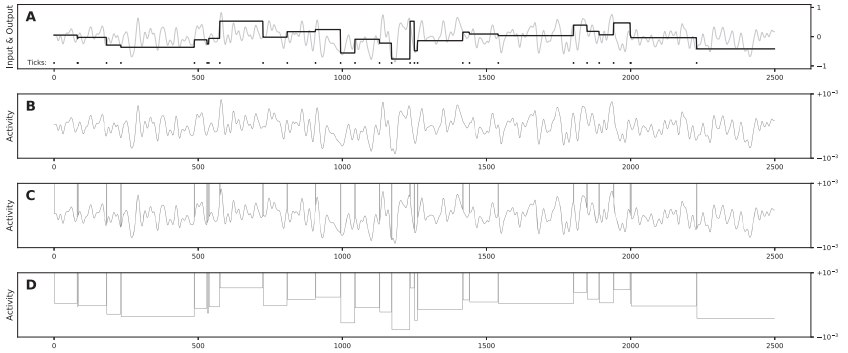


Figure 5: Performance of the reduced model on the one-gate task. (A) The light gray line is the input signal, and the thick black one is the output of the model. Black dots at the bottom represent the trigger signal (when to memorize a new value). (B–D) Respective activity of X_1, X_2, and X_3 units.

the defined sparsity (ratio of non-null values). The resulting matrix is then divided by its largest absolute eigenvalue and multiplied by the specified spectral radius. The matrix W_{out} is initially undefined and is learned through teacher forcing (Lukoševičius, 2012) and a linear regression using Moore-Penrose pseudoinverse,

$$W_{out} = YX^T(XX^T)^{-1}, \tag{2.12}$$

where Y and X correspond to the respective concatenation of all the desired outputs and reservoir states during a run, each row corresponding to a time step.

3 Results

3.1 The Reduced Model. The reduced model displays a quasi-perfect performance (RMSE = 2e-6 with $a = 10$ and $b = 10^{-3}$) as shown in Figure 5, and the three neurons $X1$, $X2$, and $X3$ behave as expected. $X1$ is strongly correlated with V (see Figure 5B), $X2$ is strongly correlated with V and saturates in the presence of a tick in T (see Figure 5C), and $X3$ is strongly correlated with M and saturates in the presence of a tick in T (see Figure 5D). This reduced model is actually a very good approximation of a line attractor (i.e., a line of points with very slow dynamics) even though we can prove that due to the tanh nonlinearity, in the absence of inputs, the model will converge to a null state (possibly after a very long time), independent of parameters a and b and the initial state. Nonetheless, Figure 5 clearly shows that information can be maintained provided b is small enough. There is a

Table 1: Default Parameters.

Parameter	Value
Spectral radius	0.1
Sparsity	0.5
Leak	1.0 (no leak)
Input scaling	1.0
Feedback scaling	1.0
Number of units	1000
Noise	0.0001
Training time steps	25,000
Testing time steps	2,500
Trigger probability	0.01

Note: Unless specified otherwise, these are the parameters used in all the simulations.

drift, but this drift is so small that it can be considered negligible relative to the system time constants: these slow points can be considered as a line or segment attractor (Seung, 1996; Sussillo & Barak, 2013). As Seung (1998) explained, "The reader should be cautioned that the term 'continuous attractor' is an idealization and should not be taken too literally. In real networks, a continuous attractor is only approximated by a manifold in state space along which drift is very slow." Nevertheless, it is worth mentioning that in order to have a true line attractor, one can replace the tanh activity function with a linear function saturated to 1 and, -1.

3.2 The Reservoir Model. Unless specified otherwise, all the reservoir models were parameterized using values given in Table 1. These values were chosen to be simple and do not have an impact on the performance of the model, as we explain in section 3.3. All simulations and figures were produced using the Python scientific stack: SciPy (Jones, Oliphant, & Peterson, 2001), Matplotlib (Hunter, 2007), and NumPy (van der Walt, Colbert, & Varoquaux, 2011). (Sources are available at github.com/rougier/ESN-WM.)

Results for the reservoir model show a very good generalization performance with a precision on the order of 10^{-3} for the level of noise considered (10^{-4}). Better precision can be obtained for lower noise levels, as shown in Figure 7. Surprisingly, this generalization property stands with as few as only four random training values, where we can achieve a 10^{-3} level of precision.

3.2.1 One-Value, One-Gate Scalar Task. The model has been trained using the parameters given in Table 1. The V signal is made of 25,000 random values sampled from a pseudo-random uniform distribution between

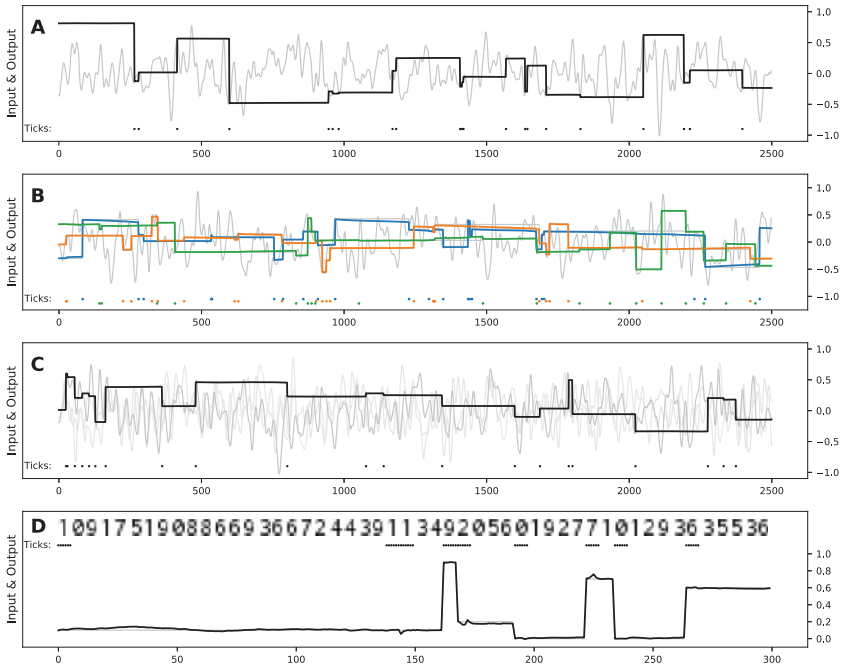


Figure 6: Performance of the reservoir model on working memory tasks. The light gray line is the input signal, and the thick black (or colored) one is the output of the model. Dots at the bottom represent the trigger signal (when to memorize a new value). For the digit task, the input containing the value to maintain is shown as a picture instead. (A) One-value, one-gate scalar task. (B) One-value, three-gate scalar task. (C) Three-value, one-gate scalar task. (D) One-value, one-gate digit task.

-1 and $+1$. The T signal is built from 25,000 random binary values with probability 0.01 of having $T = 1$ and $T = 0$ otherwise. During training, each of the inputs is presented to the model, and the output is forced with the last triggered input. All the input (u) and internal (x) states are collected, and the matrix W_{out} is computed according to equation 2.12. The model has been tested using a V signal made of 2500 random values sampled from a pseudo-random uniform distribution between -1 and $+1$. For readability of the figure (see Figure 6A), this signal has been smoothed using a fixed-size Hann window filter. The corresponding T signal has been generated following the same procedure as during the training stage. Figure 6A displays an illustrative test run of the model (for a more thorough analysis of the performance, see the section 3.3) where the error in the output is always kept below 10^{-2} and the RMSE is about 3×10^{-3}.

3.2.2 One-Value, Three-Gate Scalar Task. We trained the model on the one-value, three-gate task using the same protocol as for the one-value, one-gate task, using a single-value input, three input triggers, and three corresponding outputs. Since there are now three feedbacks, we divided the respective feedback scaling by 3. Figure 6B shows that maintaining information simultaneously has an impact on the performance of the model (illustrative test run). There is no catastrophic effect but performances are clearly degraded when compared to the one-value, one-gate task. The RMSE on this test run increased by one order of magnitude and is about 2×10^{-2}. Nevertheless, in the majority of the cases we tested, the error does stay below 10^{-2}. However in a few cases, one memory (and not necessary all) degrades faster.

3.2.3 Three-Value, One-Gate Scalar Task. We used the same protocol as for the one-value, one-gate scalar task, but there are now two additional inputs not related to the task and that can be considered as noise. Adding such irrelevant inputs had no effect on solving the task, as illustrated in Figure 6C, which shows an illustrative test run of the model. The error in the output is also always kept below 10^{-2}, and the RMSE is about the same (3×10^{-3}). This is an interesting result because it means the network is not only able to deal with background noise at 10^{-4}, but it is also able to deal with noise that has the same amplitude as the input. This is an important property to be considered for the modeling of the prefrontal cortex: being an integrative area, the PFC is typically dealing with multimodal information, much of it not relevant for the working memory task at hand (Mante, Sussillo, Shenoy, & Newsome, 2013).

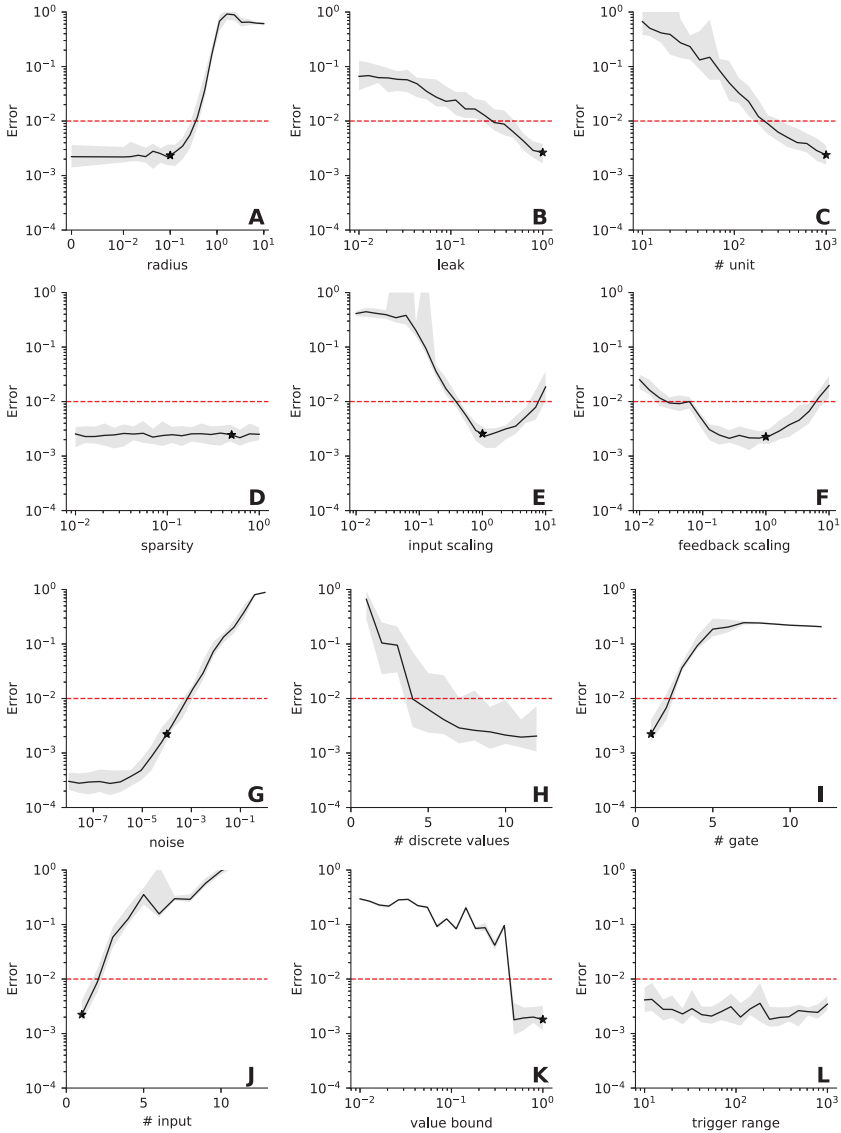
3.2.4 One-Value, One-Gate Digit Task. The model has been trained using 25,000 random integer values between 0 and 9 sampled from an uniform distribution (these values are then divided by 10 to fit in [0, 1]). Each of these values is drawn one after the other onto an image, each character covering six columns of the image. The input V consists then of the rows of this image. The T signal is sampled similarly as in the one-value, one-gate task and then expanded six times to fit the transformation of the value to the picture of the values. This means that the trigger lasts six time steps. Interestingly, as we show in Figure 6D, even if the value to maintain is no longer explicit, it can still be extracted and maintained. On the test run, we show that the RMSE is about 4×10^{-2}. It is to be noted that the recognition of a digit is not straightforward and may require a few time steps before the digit is actually identified. However, when a good value is maintained, it seems to last; the absolute error stays below 0.05, the threshold from which we can distinguish between two values. The reservoir parameters that we found are robust enough to enable not only a pure memory task (i.e., gating), but also a discrimination task (i.e., digit recognition).

Dambre, Verstraeten, Schrauwen, and Massar (2012) demonstrated the existence of a universal trade-off between the nonlinearity of the

computation and the short-term memory in the information processing capacity of any dynamical systems, including echo state networks. In other words, the hyperparameters used to generate an optimal reservoir for solving a given memory task would not be optimal for a nonlinear computation task. Here we see that even if the reservoir is made to memorize values for a while, it is still able to do a nonlinear task such as discriminating a stream of digits. Pascanu and Jaeger (2011) initiated the concept of such working memory (WM) units for reservoirs. They processed streams of characters using six binary WM units to record the deepness of curly brackets that appeared. We made such reservoir WM units coupling more general: from binary to continuous values. Instead of relying on a collection of N binary WM units to encode N values or on the maximum encoding of 2^N values, we have shown that a reservoir can use only one WM unit to encode a continuous value with good precision.

3.3 Robustness. We analyzed the robustness of the model first by measuring its sensitivity for each of the hyperparameters (see Figures 7A–7F): input scaling, feedback scaling, spectral radius, sparsity of the reservoir weight matrix, leak term (α) in equation 2.11, and number of units in the reservoir. We also measured its sensitivity to task hyperparameters (see Figures 7G–7L): the noise level (ξ), the number of discrete values used during training (when there is a trigger, V is uniformly sampled between -1 and 1 discrete values), the temporal delay between successive gating signals in training (T is built sampling its interval between triggers uniformly between 0 and a bound), the bound used to sample the input value during training (V is uniformly sampled between $-b$ and b instead, where b is the bound), the total number of input gates (with a corresponding number of outputs), and the number of input values. For most hyperparameters, we set a minimum and maximum value and pick 20 values logarithmically spread inside this range. For each task and model hyperparameter, we ran 20 simulation instances for 25,000 time steps and record the mean performance using 2500 values. Results are shown in Figure 7.

First, we can see a nonsensitivity to the sparsity (i.e., minor differences in performances when these parameters vary). Similarly, we can see a nonsensitivity to the leak term, input scaling, and feedback scaling as long as they are not too small. Note that input and feedback scaling should also not be too big. As expected, performance increases with the number of neurons. Surprisingly, we note that the performance decreases with the spectral radius. In fact, in supplementary Figure S5 we analyzed the behavior of the reservoir model with various spectral radii. We show that even with a bigger spectral radius, the reservoir continues to maintain something relevant but less precise (the segment attractor is slowly degenerating). Globally, the reservoir model is very robust against model hyperparameter changes (as long as it is trained in each condition).



Concerning the task hyperparameters, one can see in Figure 7 that only the trigger range has no impact. This means that whatever the time elapsed between two triggers, it does not affect the performance. Performance naturally decreases with the increase of the noise level (see Figure 7G), the number of input gates (I), or the number of input values (J). We note that

the number of discrete values used during training affects performance in a very specific way (H). Using between four and seven training values, the performance is already good and does not improve further with supplementary training values. This means that even if the reservoir model has been trained only to build a few stable points, it is able to interpolate and maintain the other points on the segment attractor. Interestingly, in Figure 7K, we can see a similar case of interpolation relative to the input value bound x (i.e., the interval $[-x, x]$ on which the output is trained). The performance reached a plateau when the bound values reached 0.5, while the interval used for testing performance is always $[-1, 1]$.

3.4 Dynamics

3.4.1 A Segment Attractor. Figure 8 shows how the model evolves after having been trained for the one-value, one-gate task using different starting positions and receiving no input. This results in the formation of a segment attractor even though the model was trained only to gate and memorize continuous values. If we compute the principal component analysis (PCA) on the reservoir states and look at the principal components (PCs) in the absence of inputs, we can observe (see supplementary Figure S4) that all the reservoir states are organized along a straight line on the first component (the one that explains most of the variance) and each point of this line can be associated with a corresponding memory value. Interestingly enough, there are points on this line that correspond to values outside the $[-1; 1]$ range,

Figure 7: Robustness of the model to hyperparameters. The performance of the model has been measured when a single hyperparameter varies while all others are kept constant and equal to the value given in Table 1. For each plot, we ran 20 iterations of the model using different random seeds for initialization. Most of the time (A–G, K–L), the varying parameter has been split into 20 log-uniformly spread values between the minimum and maximum values. (A–F) Model hyperparameters. (G–L) Task hyperparameters. (A) Spectral radius ranging from 0.01 to 10. with an additional 0. (B) Leak term ranging from 0.01 to 1 (no leak). (C) Number of units in the reservoir (from 1 to 1000). (D) Sparsity level ranging from 0.01 to 1.0 (fully connected). (E) Input scaling ranging from 0.01 to 10. (F) Feedback scaling ranging from 0.01 to 10. (G) Noise level inside the reservoir, ranging from 10^{-8} to 1. (H) Number of discrete values used to train the model, ranging from 1 to 12. (I) Number of input gates and output channels, ranging from 1 to 12. (One-value, n-gate scalar task. RMSE has been divided by the square root of the number of gates.) (J) Number of input values, ranging from 1 to 12. (n-value, one-gate scalar task.) (K) Bound for the input value during training, ranging from 0.01 to 1. (L) Maximal interval (number of steps) between consecutive ticks during training, ranging from 10 to 1000. The stars show the performance for default parameters (see Table 1).

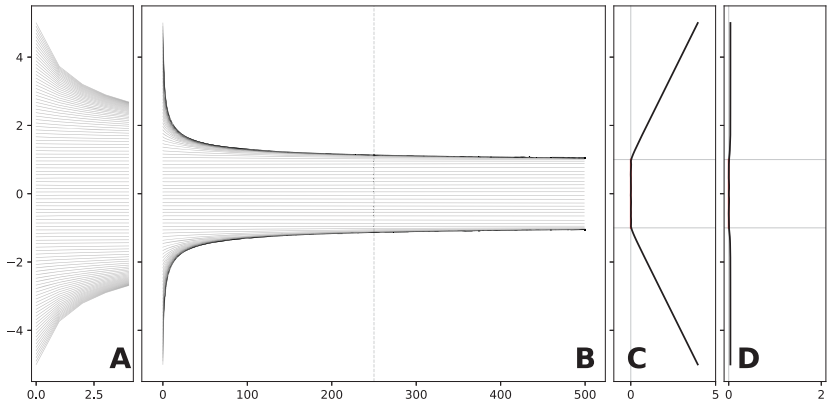


Figure 8: An approximate segment attractor. The same model trained on the one-value, one-gate was tested for 500 iterations without inputs, starting with an initial trigger along with values linearly distributed between -5 and $+5$. Panels A and B are output trajectories. Each line corresponds to one trajectory (output value) of the model. Panel A is a zoom of panel B on the first few time steps. (C) Measure of how well the initial triggered value is maintained—the absolute difference in the output between initial time and final time. (D) Measure of how stable the states are reached by the reservoir: The maximal absolute difference between states at intermediate time (dashed line) and final time.

that is, values for which the model has not been trained for. However, these points are not stable, and any dynamics starting from these points converge toward the points associated with the values 1 or -1 (see Figure 8).

3.4.2 V-Like and M-Like Neurons. Similar to the minimal model, in the absence of input, the inner dynamic of the reservoir model is a combination of both sustained and highly variable activities. More precisely, in the one-value, one-gate task, we notice two types of neurons that are similar to neurons $X1$ and $X3$ in the reduced model: (1) neurons that solely follow the input V (i.e., V-like neurons) and (2) neurons that mostly follow the output M (i.e., M-like neurons), respectively. We also notice that M-like neurons' average activity is linearly linked with the M value and fluctuate around this mean activity according to the input V. In Figure 9, we show M-like neurons for the different tasks. These neurons were found by taking the most correlated neurons with the output M[3]. From Figures 9A to 9D, we see that the M-like neurons' link with M output gets weaker and weaker: the average sustained activity goes from nearly flat to highly perturbed.

[3] Because a trigger input has substantial influence on the reservoir states, we made the categorization by ignoring the time steps when there is a trigger.

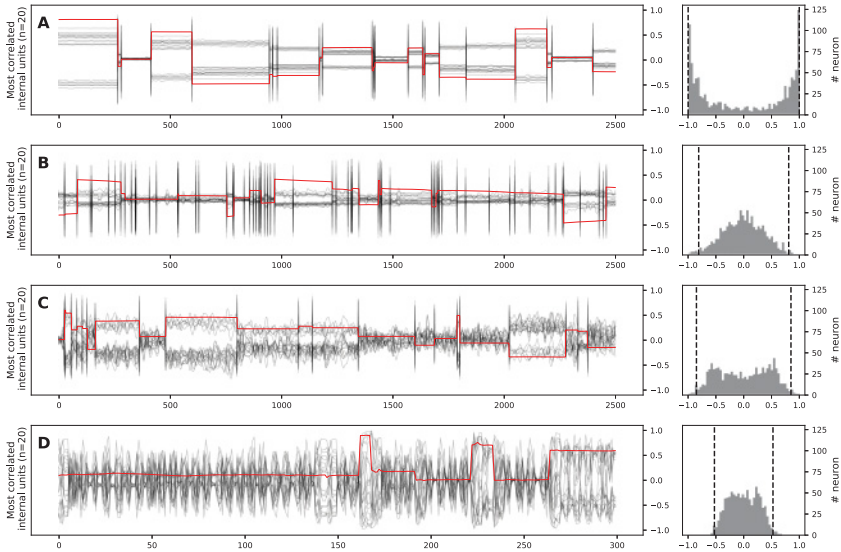


Figure 9: Most correlated reservoir units displaying various degrees of maintained activity. (Left) In black are the activities of the 20 neurons that are the most correlated with the output, and in red are the outputs of the model. (Right) Histogram of the correlation between the neurons and the output. The y-axis represents the number of neurons correlated with the output produced by the model within a correlation bin. The dashed lines indicate the 20th most correlated and most anticorrelated neurons with the output. (A) One-value, one-gate scalar task. (B) One-value, three-gate scalar task. (C) Three-value, one-gate scalar task. (D) One-value, one-gate digit task. Most correlated reservoir units do not necessarily show clear, sustained activity; we can see a degradation of sustained activities from panels A to D according to the different tasks. Thus, sustained activity is not mandatory to perform working memory tasks.

This "degradation" of sustained activity is explained by the change in the distribution of correlations of the entire reservoir population with M output. In Figure 9 (right) we see that the correlation with M output is quickly shrinking from panels A to D. For the one-value, one-gate task (see Figure 9A), almost all neurons stay at the same value while maintaining the memory. However, when more values have to be maintained (see Figure 9B) or when more inputs are received (see Figure 9C), most of the activities no longer stay at the same value while maintaining the memory. In fact, in the one-value, one-gate digit task, neurons do not display a sustained activity at all (see Figure 9D). Interestingly, similar behavior (no sustained activity) can be obtained by lowering the feedback scaling (see supplementary Figure S6) or by enforcing the input weights to be big

enough (see supplementary Figure S8). More formally, when there is no trigger, the activity of the neurons can be rewritten as $\tanh(aX + bM)$. The two proposed modifications make the ratio $\frac{a}{b}$ bigger, and eventually, when $a \gg b$, $\tanh(aX + bM) \approx \tanh(aX)$. Consequently, $\tanh(aX)$ is highly correlated with X as aX stays bounded between -1 and 1 and does not depend of M. Similarly, when $a \ll b$, $\tanh(aX + bM) \approx \tanh(bM)$ which is in turn highly correlated with M for the same reasons.

3.4.3 Linear Decoder. To go further in understanding the role played by sustained activities, we wanted to know how much of these sustained activities were necessary to decode the output memory M. For the one-value, one-gate task, we trained a separate decoder based on a subsample of the reservoir population. We increasingly subsampled neurons based on three conditions: choosing the most correlated one first, choosing the least correlated one first, or just randomly selecting them. In Figure 10, we can see two interesting facts. First, there is no need for the whole reservoir population to decode well enough the memory M: taking 100 neurons among 1000 is sufficient. Second, if we take enough neurons, there is no advantage in taking the most correlated one first; random ones are enough. Surprisingly, it seems better to rely on randomly selected units than most correlated ones. This suggests that randomly distributed activities contained more information than just the most correlated unit activities and offers a complementary perspective when comparing it to Rigotti et al. (2013) decoding of neural population recorded in monkeys: when task-related neurons are not kept for decoding, information (but less accurate) can still be decoded.

3.5 Equivalence between the Minimal and the Reservoir Model. In order to understand the equivalence between the minimal and the reservoir model, it is important to note that there are actually two different regimes, as shown in Figure 11. One regime corresponds to the absence of a trigger ($T = 0$), and the other regime corresponds to the presence of a trigger ($T = 1$). When there is no trigger ($T = 0$), the activities of X_1 and X_2 compensate each other because they are in a quasi-linear regime (b being very small), and their summed contribution to the output is nil.

In the reservoir model, we can identify an equivalent population by discriminating neurons inside the reservoir based on the strength of their input weight relative to V. More formally, we define R_{12} as the group of neurons whose input weight from V (absolute value) is greater than 0.1. Figure 12H shows that the summed output of these neurons is quasi nil, while the complementary population, R_3, is fully correlated with the output and is thus equivalent to the X_3 unit in the minimal model.

Symmetrically, in the presence of a trigger ($T = 1$), the activities of X_2 and X_3 compensate each other because they are in a saturated regime (a being very large) and their summed contribution to the output is nil. We can again identify an equivalent population in the reservoir model by

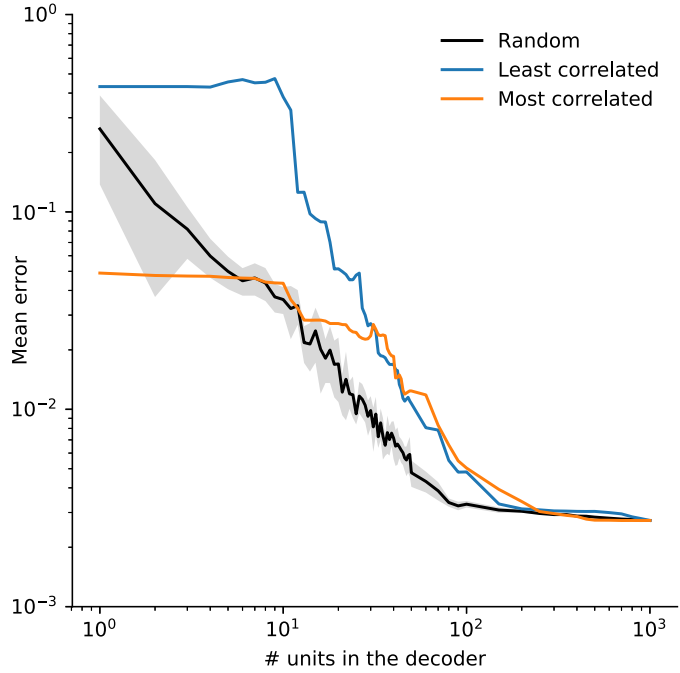


Figure 10: Performance of variable size linear decoders. After training, we ran the model on the training set and recorded all the internal states. We selected a subset (from 1 to 1000 units) of these internal states to find the associated best linear decoder on the training data set. Finally, we measured the performance of these decoders on the training sets. Units composing a subset have been sampled using either the least or the most correlated units (with the output) or simply randomly. Performance for subsets of size 1000 is equal to the performance of the original model.

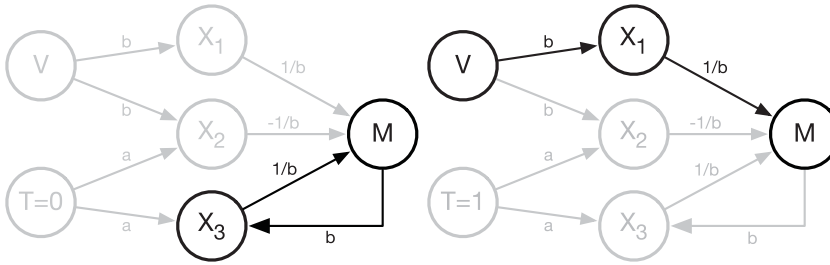


Figure 11: The two regimes of the minimal model depending on the absence (left) or the presence (right) of a trigger T (0 or 1).

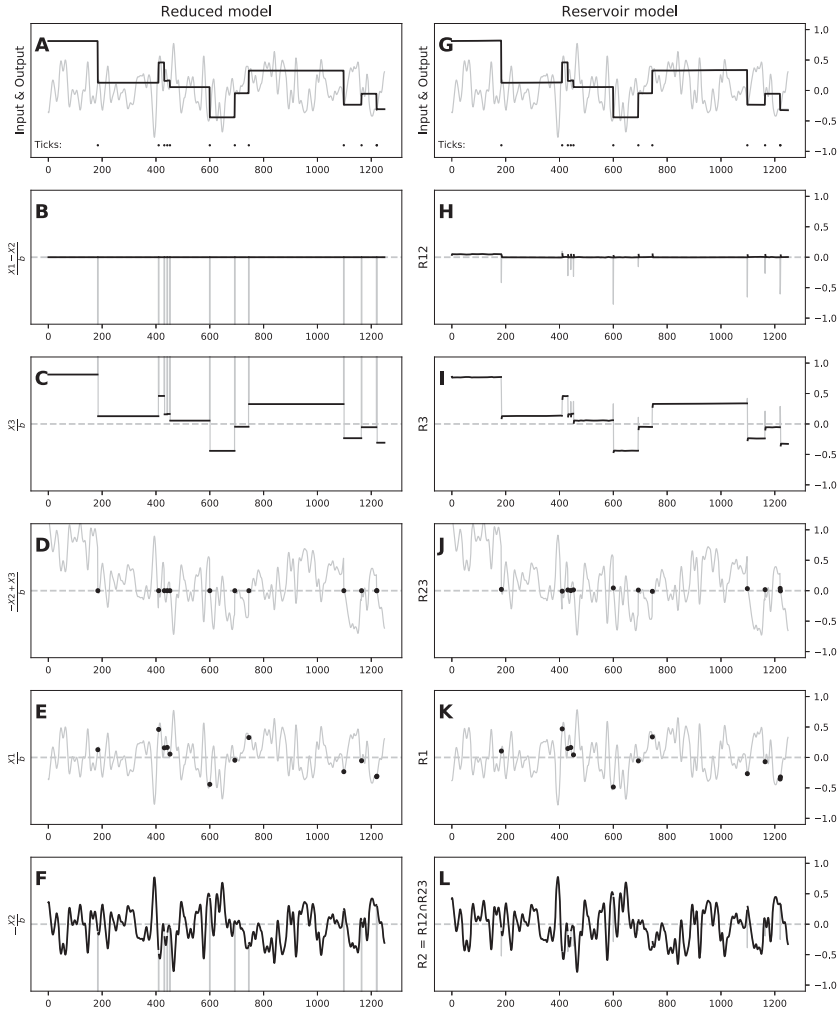


Figure 12: Side-by-side comparison of the minimal (left) and full (right) models. (A, G) Output of the two models, respectively. (B, H) Group of neurons whose sum of activity is nil between triggers. (C, I) Group of neurons whose sum of activity is equal to the output between triggers. (D, J) Group of neurons whose sum of activity is nil during triggers. (E, K) Group of neurons whose sum of activity is equal to the input during triggers. (F, L) Group of neurons whose sum of activity is equal to the input between triggers.

Table 2: Correlation between Respective Subpopulations in the Reduced and Reservoir Models.

Populations	Output Correlation
X1 / R1 ($T = 1$)	0.9996
X2 / R2 ($T = 0$)	0.9997
X3 / R3 ($T = 0$)	0.9997

Notes: Output is restricted to the contribution of the considered subpopulation. Correlations are computed only at time relevant for the subpopulation ($T = 0$ or $T = 1$).

discriminating neurons inside the reservoir based on the strength of their input weights relative to both T and V. More formally, we define R_{23} as the group of neurons whose input weight from T (absolute value) is greater than 0.05 and whose input weight from V (absolute value) is smaller than 0.1. Figure 12J shows that the summed output of these neurons is quasi nil, while the complementary population R_1 is fully correlated with the input V and is thus equivalent to the X_1 unit in the minimal model.

We can identify R_2 by taking the intersection of R_{12} and R_{23}, whose activity is similar to X_2 (see Figure 12L). Consequently, we have identified in the reservoir disjoint subpopulations that are respectively and collectively equivalent to activity of X_1, X_2, and X_3 in the minimal model. Table 2 quantifies this equivalence using simple correlations. To explore further this equivalence, we also conducted comparative lesion studies between the two models (see the supplementary materials). However, with the original set of parameters for the reduced model ($a = 1000$, $b = 0.001$), lesioning X_2 or X_3 makes the reduced model behave in a degraded and extreme mode: outputs range from 0 to ± 1000 whenever there is a trigger, and it makes the comparison with the reservoir difficult (see supplementary Figure S9). By choosing an alternative set of parameters ($a = 1000$, $b = 1$, which, incidently, makes the reduced model unable to sustain memory), we can show a strong correlation with the reservoir when X_1/R_1 (resp. X_2/R_2, X_3/R_3) are silenced (see supplementary Figure S10) and this further tightens the relationship between the two models.

4 Discussion

In computational neuroscience, the reservoir computing paradigm (RC) (Jaeger, 2001; Maass, Natschläger, & Markram, 2002; Verstraeten, Schrauwen, D'Haene, & Stroobandt, 2007), originally proposed independently by Dominey (1995) and Buonomano and Merzenich (1995),[4] is

[4] Earlier formulations of very similar concepts can be found in Jaeger (2007).

often used as a model of canonical microcircuits (Maass et al., 2002; Hoerzer, Legenstein, & Maass, 2012; Sussillo, 2014). It is composed of a random recurrent neural network (i.e., a reservoir) from which readout units (i.e., outputs) are trained to linearly extract information from the high-dimensional, nonlinear dynamics of the reservoir. Several authors have taken advantage of this paradigm to model cortical areas such as PFC (Hinaut & Dominey, 2013; Mannella & Baldassarre, 2015; Hinaut et al., 2015; Enel, Procyk, Quilodran, & Dominey, 2016) because most of the connections are not trained, especially the recurrrent ones. Another reason to use the reservoir computing paradigm for PFC modelling is that PFC also hosts high-dimensional, nonlinear dynamics (Rigotti et al., 2013). RC offers a neuroanatomically plausible view of how cortex-to-basal-ganglia (i.e., cortico-basal) connections could be trained with dopamine: the reservoir plays the role of a cortical area (e.g., trained with unsupervised learning), and the read-out units play the role of basal ganglia input (i.e., striatum).

However, in many dynamical systems, reservoirs included, there exists a trade-off between memory capacity and nonlinear computation (Dambre et al., 2012).[5] This is why some studies have focused on reservoirs with dedicated readout units acting as WM units (Hoerzer et al., 2012; Pascanu & Jaeger, 2011; Nachstedt & Tetzlaff, 2017). These WM units have feedback connections projecting to the reservoir and are trained to store binary values that are input dependent. This somehow simplifies the task and enables the reservoir to access and use such long-time dependency information to perform a more complex task, freeing the system from constraining reservoir short-term dynamics. Such ideas already had some theoretical support; for instance Maass, Joshi, and Sontag (2007) showed that with an appropriate readout and feedback functions, readout units could be used to approximate any k-order differential equation. Pascanu and Jaeger (2011) used up to six binary WM units to store information in order to solve a nested bracketing-levels task. Using principal component analysis, they showed that these binary WM units constrain the reservoir in lower-dimensional "attractors." In addition, Hoerzer et al. (2012) showed that analog WM units (encoding binary information) also drive the reservoir into a lower-dimensional space (i.e., 99% of the variability of the reservoir activities are explained by fewer principal components). More recently, Strock, Rougier, and Hinaut (2018) and Beer and Barak (2019) used such WM units in order to store analog values (as opposed to binary ones) in order to build a line attractor (Seung, 1996; Sussillo & Barak, 2013). In particular, Beer and Barak (2019) explored how a line attractor can be built online, by comparing FORCE (Sussillo & Abbott, 2009) and LMS algorithms, using a WM unit to maintain a continuous value in the absence of input perturbations.

[5] For reservoirs, this trade-off depends on the hyperparameters (HP) chosen. Some HP sets give more memory, others more computational capacity (Legenstein & Maass, 2007).

In that context, the minimal model of three neurons we have proposed helps to understand the mechanisms that allow a reservoir model to gate and maintain scalar values in the presence of input perturbations instead of just binary values. As explained previously, this minimal model exploits the nonlinearity and the asymptotic behavior of the three tanh units and mimics a select operator between the input signal and the output. In the case of the reservoir model, there is no precise architecture or crafted weights, but this is compensated by the size of the population inside the reservoir, along with the training of the output weights. More precisely, we have shown that virtually any population of randomly connected units is able to maintain an analog value at any time and for an arbitrary delay. Taking advantage of the nonlinearity of the neuron transfer function, we have shown how such a population can learn a set of weights during the training phase using only a few representative values. Given the random nature of the architecture and the large set of hyperparameters, for which the precision of the output remains acceptable, this suggests that this property could be a structural property of any population that could be acquired through learning. To achieve such property, we mainly used offline training in our analyses (for efficiency reasons), but we have shown that it also works with online FORCE learning (see the supplementary materials and Figures S2 and S3).

We have shown that the reservoir model behavior is similar to the minimal model with the presence of two "macrostates" that are implemented by compensatory clusters. In a nutshell, this working memory uses two distinct mechanisms: a selection mechanism (i.e., a switch) and a line attractor. Such mechanisms have been also reported in a fully trained recurrent neural network with backpropagation (Mante et al., 2013). The authors proposed a context-dependent selective integration task and showed that "the rich dynamics of PFC responses during selection and integration of inputs can be characterized and understood with just two features of a dynamical system—the line attractor and the selection vector, which are defined only at the level of the neural population." However in our case, not only did we rely on the analysis of the dynamical system to understand the behavior of the system, but we were able to design a minimal model implementing these mechanisms and show that these same mechanisms are also present in the reservoir model but in a distributed way.

Finally, one important feature of the model is that it is actually an open system and, as such, is continuously under the direct influence of external activities. More precisely, the model is able to retain information when the gate is closed, but this closed gate corresponds to a functional state rather than a physical state where input signals would have been blocked and information continues to enter the system. This is illustrated quite clearly when one looks at the internal activities inside the reservoir: a large number of neurons are directly (and only) correlated with the input signals. This has consequences for the analysis of the dynamics of the population: this population is partly driven by the (working memory) task and partly driven by

uncorrelated external activities. If we go back to biology, this makes perfect sense: a population of neurons is never isolated from the rest of the brain. When studying electrophysiological recordings, we have to keep in mind that such activities can be fully uncorrelated with the task we are observing. This might be one of the reasons for the variability of hypotheses about working memory encoding mechanisms.

References

Amari, S.-I. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, *27*(2), 77–87. doi:10.1007/bf00337259

Barak, O., & Tsodyks, M. (2014). Working models of working memory. *Current Opinion in Neurobiology*, *25*, pp. 20–24. doi:10.1016/j.conb.2013.10.008

Bechara, A., Damasio, H., Tranel, D., & Anderson, S. W. (1998). Dissociation of working memory from decision making within the human prefrontal cortex. *Journal of Neuroscience*, *18*(1), 428–437. doi:10.1523/jneurosci.18-01-00428.1998

Beer, C., & Barak, O. (2019). *One step back, two steps forward: Interference and learning in recurrent neural networks.* arXiv:1805.09603v2.

Bouchacourt, F., & Buschman, T. J. (2019). A flexible model of working memory. *Neuron*, *103*(1), 147–160. doi:10.1016/j.neuron.2019.04.020

Buonomano, D. V., & Merzenich, M. M. (1995). Temporal information transformed into a spatial code by a neural network with realistic properties. *Science*, *267*(5200), 1028–1030.

Cho, K., van Merriënboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (pp. 1724–1734). Stroudsburg, PA: Association for Computational Linguistics. http://www.aclweb.org/anthology/D14-1179

Compte, A. (2000). Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model. *Cerebral Cortex*, *10*(9), 910–923. doi:10.1093/cercor/10.9.910

Compte, A. (2006). Computational and in vitro studies of persistent activity: Edging towards cellular and synaptic mechanisms of working memory. *Neuroscience*, *139*(1), 135–151. doi:10.1016/j.neuroscience.2005.06.011

Constantinidis, C., Funahashi, S., Lee, D., Murray, J. D., Qi, X.-L., Wang, M., & Arnsten, A. F. (2018). Persistent spiking activity underlies working memory. *Journal of Neuroscience*, *38*(32), 7020–7028. doi:10.1523/jneurosci.2486-17.2018

Dambre, J., Verstraeten, D., Schrauwen, B., & Massar, S. (2012). Information processing capacity of dynamical systems. *Scientific Reports*, *2*(1). doi:10.1038/srep00514

Dominey, P. F. (1995). Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biological Cybernetics*, *73*(3), 265–274. doi:10.1007/bf00201428

Durstewitz, D., Seamans, J. K., & Sejnowski, T. J. (2000). Neurocomputational models of working memory. *Nature Neuroscience*, *3*, pp. 1184–1191. doi:10.1038/81460

Edin, F., Klingberg, T., Johansson, P., McNab, F., Tegner, J., & Compte, A. (2009). Mechanism for top-down control of working memory capacity. *Proceedings of the National Academy of Sciences*, *106*(16), 6802–6807. doi:10.1073/pnas.0901894106

Enel, P., Procyk, E., Quilodran, R., & Dominey, P. F. (2016). Reservoir computing properties of neural dynamics in prefrontal cortex. *PLOS Computational Biology*, *12*(6), e1004967. doi:10.1371/journal.pcbi.1004967

Funahashi, S. (2017). Working memory in the prefrontal cortex. *Brain Sciences*, *7*(12), 49. doi:10.3390/brainsci7050049

Fuster, J. M. (2001). The prefrontal cortex—an update. *Neuron*, *30*(2), 319–333. doi:10.1016/s0896-6273(01)00285-9

Gandhi, N. J., & Katnani, H. A. (2011). Motor functions of the superior colliculus. *Annual Review of Neuroscience*, *34*(1), 205–231. doi:10.1146/annurev-neuro-061010-113728

Goldman-Rakic, P. S. (1987). Circuitry of primate prefrontal cortex and regulation of behavior by representational memory. *Comprehensive Physiology*. doi:10.1002/cphy.cp010509

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. doi:10.1109/tnnls.2016.2582924

Hinaut, X., & Dominey, P. F. (2013). Real-time parallel processing of grammatical structure in the fronto-striatal system: A recurrent network simulation study using reservoir computing. *PLoS One*, *8*(2), e52946. doi:10.1371/journal.pone.0052946

Hinaut, X., Lance, F., Droin, C., Petit, M., Pointeau, G., & Dominey, P. F. (2015). Corticostriatal response selection in sentence production: Insights from neural network simulation with reservoir computing. *Brain and Language*, *150*, pp. 54–68. doi:10.1016/j.bandl.2015.08.002

Hoerzer, G. M., Legenstein, R., & Maass, W. (2012). Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. *Cerebral Cortex*, *24*(3), 677–690.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*, *9*(3), 90–95. doi:10.1109/MCSE.2007.55

Jaeger, H. (2001). *The "echo state" approach to analysing and training recurrent neural networks* (Tech. Rep. No. 148). Bonn: German National Research Center for Information Technology GMD. http://publica.fraunhofer.de/starweb/servlet.starweb?path=urn.web&search=urn:nbn:de:0011-b-731351

Jaeger, H. (2007). Echo state network. *Scholarpedia*, *2*(9), 2330. doi:10.4249/scholarpedia.2330

Jones, E., Oliphant, T., & Peterson, P. (2001). *SciPy: Open source scientific tools for Python*. http://www.scipy.org

Koulakov, A. A., Raghavachari, S., Kepecs, A., & Lisman, J. E. (2002). Model for a robust neural integrator. *Nature Neuroscience*, *5*(8), 775–782. doi:10.1038/nn893

Legenstein, R., & Maass, W. (2007). Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, *20*(3), 323–334. doi:10.1016/j.neunet.2007.04.017

Lim, S., & Goldman, M. S. (2013). Balanced cortical microcircuitry for maintaining information in working memory. *Nature Neuroscience*, *16*(9), 1306–1314. doi:10.1038/nn.3492

Lukoševičius, M. (2012). A practical guide to applying echo state networks. In G. Montavon, G. B. Orr, & K.-R. Müller, (Eds.), *Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science*, vol. *7700* (pp. 659–686). Berlin: Springer. doi:10.1007/978-3-642-35289-8_36

Maass, W., Joshi, P., & Sontag, E. D. (2007). Computational aspects of feedback in neural circuits. *PLOS Computational Biology*, *3*(1), e165. doi:10.1371/journal.pcbi.0020165

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560. doi:10.1162/089976602760407955

Machens, C. K., Romo, R., & Brody, C. D. (2010). Functional, but not anatomical, separation of "what" and "when" in prefrontal cortex. *Journal of Neuroscience*, *30*(1), 350–360. doi:10.1523/jneurosci.3276-09.2010

Mannella, F., & Baldassarre, G. (2015). Selection of cortical dynamics for motor behaviour by the basal ganglia. *Biological Cybernetics*, *109*(6), 575–595. doi:10.1007/s00422-015-0662-6

Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, *503*(7474), 78–84. doi:10.1038/nature12742

Miller, E. K., & Cohen, J. D. (2001). An integrative theory of prefrontal cortex function. *Annual Review of Neuroscience*, *24*(1), 167–202. doi:10.1146/annurev.neuro.24.1.167

Nachstedt, T., & Tetzlaff, C. (2017). Working memory requires a combination of transient and attractor-dominated dynamics to process unreliably timed inputs. *Scientific Reports*, *7*(1). doi:10.1038/s41598-017-02471-z

O'Reilly, R. C., & Frank, M. J. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, *18*(2), 283–328. doi:10.1162/089976606775093909

Pascanu, R., & Jaeger, H. (2011). A neurodynamical model for working memory. *Neural Networks*, *24*(2), 199–207. doi:10.1016/j.neunet.2010.10.003

Rigotti, M., Barak, O., Warden, M. R., Wang, X.-J., Daw, N. D., Miller, E. K., & Fusi, S. (2013). The importance of mixed selectivity in complex cognitive tasks. *Nature*, *497*(7451), 585–590. doi:10.1038/nature12160

Romo, R., Brody, C. D., Hernández, A., & Lemus, L. (1999). *Nature*, *399*(6735), 470–473. doi:10.1038/20939

Schaetti, N., Salomon, M., & Couturier, R. (2016). Echo state networks–based reservoir computing for MNIST handwritten digits recognition. In *Proceedings of the 2016 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering*. Piscataway, NJ: IEEE. doi:10.1109/cse-euc-dcabes.2016.229

Seung, H. S. (1996). How the brain keeps the eyes still. *Proceedings of the National Academy of Sciences*, *93*(23), 13339–13344. doi:10.1073/pnas.93.23.13339

Seung, H. S. (1998). Learning continuous attractors in recurrent networks. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems*, *10* (pp. 654–660). Cambridge, MA: MIT Press.

Strock, A., Rougier, N. P., & Hinaut, X. (2018). A simple reservoir model of working memory with real values. In *Proceedings of the 2018 International Joint Conference on Neural Networks*. Piscataway, NJ: IEEE. doi:10.1109/ijcnn.2018.8489262

Sussillo, D. (2014). Neural circuits as computational dynamical systems. *Current Opinion in Neurobiology*, *25*, 156–163.

Sussillo, D., & Abbott, L. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, *63*(4), 544–557. doi:10.1016/j.neuron.2009.07.018

Sussillo, D., & Barak, O. (2013). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, *25*(3), 626–649. doi:10.1162/neco a 00409

van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, *13*(2), 22–30. doi:10.1109/mcse.2011.37

Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, *20*(3), 391–403. doi:10.1016/j.neunet.2007.04.003

Wei, Z., Wang, X.-J., & Wang, D.-H. (2012). From distributed resources to limited slots in multiple-item working memory: A spiking network model with normalization. *Journal of Neuroscience*, *32*(33), 11228–11240. doi:10.1523/jneurosci .0735-12.2012

Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory. *Journal of Neuroscience*, *16*(6), 2112–2126. doi:10.1523/jneurosci.16-06-02112.1996

Zipser, D., Kehoe, B., Littlewort, G., & Fuster, J. (1993). A spiking network model of short-term active memory. *Journal of Neuroscience*, *13*(8), 3406–3420. doi:10.1523/jneurosci.13-08-03406

Received May 14, 2019; accepted August 31, 2019.