Noname manuscript No.
(will be inserted by the editor)

Driving reservoir models with oscillations: a solution to the extreme structural sensitivity of chaotic networks

Philippe Vincent-Lamarre [1] · Guillaume Lajoie [2] · Jean-Philippe Thivierge [1]

Received: May 8th 2016 / Accepted: June 28th 2016

Abstract A large body of experimental and theoretical work on neural coding suggests that the information stored in brain circuits is represented by time-varying patterns of neural activity. Reservoir computing, where the activity of a recurrently connected pool of neurons is read by one or more units that provide an output response, successfully exploits this type of neural activity. However, the question of system robustness to small structural perturbations, such as failing neurons and synapses, has been largely overlooked. This is in contrast to well-studied dynamical perturbations that lead to divergent network activity in the presence of chaos, as is the case for many reservoir networks. Here, we distinguish between two types of structural network perturbations, namely local (e.g. individual synaptic or neuronal death) and global (e.g. network-wide fluctuations). Surprisingly, we show that while global perturbations have a limited impact on the ability of reservoir models to perform various tasks, local perturbations can produce drastic effects. To address this limitation, we introduce a new architecture where the reservoir is driven by a layer of oscillators that generate stable and repeatable trajectories. This model outperforms previous implementations while being resistant to relatively large local and global perturbations. This finding has implications for the design of reservoir models that capture the capacity of brain circuits to perform cognitively and behaviorally relevant tasks while remaining robust to various forms of perturbations. Further, our work proposes a novel role for neuronal oscillations found in cortical circuits, where they may serve as a collection of inputs from which a network can robustly generate complex dynamics and implement rich computations.

Keywords reservoir computing · recurrent neural networks · chaotic networks · oscillations · perturbations · robustness

1. School of Psychology and Center for Neural Dynamics, University of Ottawa, Ottawa, Ontario, Canada
2. UW Institute for Neuroengineering, University of Washington, Seattle, Washington, US

1 Introduction

A fundamental building block of cognition and behavior is the ability of brain circuits to maintain information about objects beyond their time of appearance in the environment. Experimental evidence suggests that sustained patterns of neural activity may constitute a simple correlate of this ability. Indeed, patterns of sustained activity are well-known to guide behavior in tasks including motor control (Churchland and Shenoy 2007; De Zeeuw et al 2011), interval timing (Goel and Buonomano 2014) and delayed discrimination (Mazurek et al 2003; Romo et al 1999). Going further, sustained activity encompasses a broad range of neural phenomena and is not limited to cases where neurons maintain a fixed firing rate over time. In fact, patterns of sustained activity may take the form of precise fluctuations in firing rate in time (Buonomano and Maass 2009) or the timing of spike patterns (Mainen and Sejnowski 1995; Tiesinga et al 2008). However, it is unclear how neural circuits generate reliable patterns of neural activity given that the spike train produced by a neuron is highly dependent on its pre-stimulus history (Arieli et al 1996; Banerjee et al 2008) and that the addition of a single spike in a train of action potentials can cause significant postsynaptic variations (London et al 2010) (c.f. Lajoie et al (2013)).

One framework that has been successful at capturing patterns of temporal activity is reservoir computing (Jaeger 2002; Maass et al 2002), and has been proposed as a plausible theory of high-level cortical information processing. The core principle of reservoir computing consists in mapping the states of neurons through time (termed neural trajectory) in a recurrent circuit to a specific output function by adjusting the connection weights of one or many downstream units (termed readout units). A body of work suggests that reservoir computing can account for neural dynamics in different regions of the brain (Barak et al 2013; Toledo-Suárez et al 2014; Yamazaki and Tanaka 2007). Furthermore, there is increasing evidence suggesting that principles of reservoir computing could explain cognitive and behavioral processes (Barak et al 2013; Bernacchia et al 2011; Enel 2014; Joshi and Maass 2005; Laje and Buonomano 2013; Mante et al 2013; Sussillo and Abbott 2009; Sussillo et al 2015)

One interesting characteristic of some reservoir models is their ability to generate activity as autonomous systems (that is, systems whose ongoing dynamics are not driven by fluctuating inputs (c.f. Lajoie et al (2013, 2014)). This is achieved by adjusting the parameters of the reservoir to place it in a chaotic state where it generates rich spontaneous activity (see e.g. Sompolinsky et al (1988)). This regime has been proposed as a possible mechanism underlying sustained irregular activity observed in cortical areas (van Vreeswijk and Sompolinsky 1996). Such activity has been harnessed in some models to create stable and repeatable trajectories in the absence of ongoing external stimulation (Hoerzer et al 2014; Laje and Buonomano 2013; Sussillo and Abbott 2009; Sussillo et al 2015). In these models, chaos is suppressed either through strong feedback from the readout units in the reservoir or by stabilizing some neural trajectories by adjusting connection weights of the reservoir. These models are robust to the addition of noise (Laje & Buonomano, 2013) as well as small alterations in network connectivity (Sussillo and Barak 2013; Sussillo et al 2015). Therefore, these reservoirs models are attractive candidates to explain both the origin of stable patterns of neural activity and how such activity can be exploited by downstream neurons in a meaningful way.

However, despite the large body of work laying parallels between neural networks in the brain and reservoir computing systems, an important biological constraint has been overlooked: the resistance to small structural perturbations such as failures in synaptic transmission or neuronal death. Such perturbations occur frequently in healthy circuits in the brain without leading to drastic functional changes (Fu et al 2015; Pakkenberg et al 2003; Westlye et al 2009). This is what we investigate in this work.

We begin by testing the resistance of different reservoir models (Laje and Buonomano 2013; Sussillo and Abbott 2009) to post-training perturbations, while drawing a distinction between structural (parameters alteration, such as neuronal and synaptic loss) and state-space (alteration of the firing rate of the neurons) perturbations. Further, we examine the impact of both local and global forms of structural perturbations. On the one hand, global perturbations are noisy fluctuations in the activity of multiple neurons, or random variations of synaptic strengths across the network that are analogous to the changes that occur under synaptic plasticity (Froemke and Schreiner 2015). On the other hand, local perturbations consist of failure of individual synapses or neurons. Here, we show that local and global perturbations of similar magnitude have a markedly different impact on the dynamic of recurrent neural circuits. Going further, some perturbations are more devastating than previously reported (Laje and Buonomano 2013; Sussillo and Barak 2013; Sussillo et al 2015). In fact, we show that minimal perturbations can completely impair the capacity of the network to execute some previously learned tasks.

Several solutions may address the issue of instability to structural perturbations in chaotic networks. For instance, it is possible to suppress chaos in spontaneously active networks of recurrent firing rate units by the injection of white noise (Molgedey et al 1992) or time-varying input signals (Bertschinger and Natschläger 2004; Rajan et al 2010). However, these solutions assume that such input is available alongside the network during the execution of the task. As an alternative, we examined a solution where a layer of input units to the network behave as oscillators. We show that this model can generate robust trajectories and exhibits resilience to perturbations. This new framework is simpler, more efficient, and less training intensive than previous models.

2 Methods

2.1 Architectures

In this work, we employ networks of firing rate units connected according to five possible architectures. Architectures A,B and C have been proposed in (Sussillo and Abbott 2009) and we refer to their implementation as standard FORCE; architecture D is proposed in (Laje and Buonomano 2013) and we refer to its implementation as innate training. We introduce architecture E in this work and refer to its implementation as the

driven model. The various architectures (A to D, Fig. 1; E, Fig. 7(a)) can be summarized as follows:

(A) Only the synapses from the reservoir to the readout unit are modified during training and there is direct feedback from the readout unit to the reservoir.
(B) The feedback is provided by a control network instead of the readout unit. The synapses from the reservoir to the readout unit and to the control network are modified during training.
(C) No external feedback is provided to the reservoir. Rather, the reservoir is its only source of feedback and is modified during the training phase. This is achieved by applying the same learning rule to the readout unit and to the neurons in the reservoir.
(D) A subset of the neurons in the reservoir is adjusted to reinforce an innate trajectory of the network. This trajectory is triggered by kicking the network in a delimited state-space with an external input. Once this process is complete, the synapses of the readout unit are modified.
(E) A layer of oscillators controls the trajectory of the reservoir based on their relative phase. Only the synapses from the reservoir to the readout unit are modified during training.

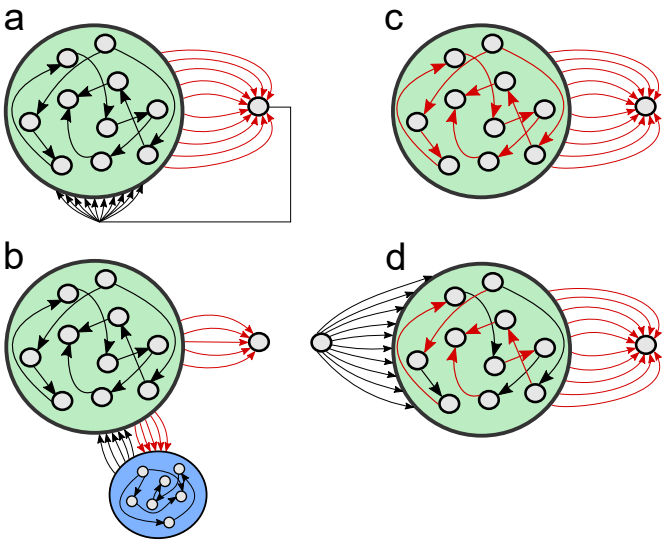

Fig. 1 Schematic representation of network architectures. Connections in black are randomly initialized and left untouched during training whereas connections in red are modified according to the training rule. **a** No external inputs are used to control the activity of the reservoir (green). Instead, the network relies on strong feedback from the readout unit to the reservoir to control the chaotic activity. **b** This architecture uses a control network (blue) instead of the readout unit to provide feedback to the reservoir. **c** Learning occurs in the reservoir as well as on the connections of the readout unit. The reservoir is its only source of feedback. **d** The reservoir displays chaotic spontaneous activity in the absence of input. After a brief pulse is injected in the network, the reservoir follows a trajectory that has been previously stabilized by training a subset of neurons.

2.2 Neural activity and parameters

We kept the same parameters used in most simulations of the original articles describing FORCE and innate training. Both models are described by the following equations:

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N_R} W_{ij}^R r_j + \sum_{\mu=1}^{N_I} W_{i\mu}^{RI} I_\mu, \tag{1}$$

$$z = \sum_{j=1}^{N} W_{ij}^{zR} r_j, \tag{2}$$

where $r_i = tanh(x_i)$ (i.e., hyperbolic tan function) is the firing rate of the recurrent units $x_i (i = 1, ..., N_R)$, $I_\mu (\mu = 1, ..., N_I)$ represents the input units, z represents the readout unit and the time constant $\tau = 10ms$. The reservoirs' weight matrices W^R are of size $N_R \times N_R$ where the non-zero values were drawn from a Gaussian distribution with a mean of zero and standard deviation equal to $g_R/\sqrt{p_R N_R}$, where g_R is the gain of the synaptic weights and p_R is the proportion of non-zero values in the matrices. Weight matrices connecting the input unit to the reservoir (W^{RI}) and connecting the reservoir to the readout units (W^{zR}) were initialized differently depending on the architecture.

Standard FORCE. The parameters used with architecture A, B and C can been found in Table 1. Readout weights W^{zR} were initially set to zero and when feedback from the readout unit z to the reservoir was used, the values of W^{Rz} were drawn from a uniform distribution with values between -1 and 1. No external inputs were used for our simulations so no weights W^{RI} were required.

Architectures A and B require the addition of some components to equation 1:

$$+W_i^{R_z} z, \tag{3}$$

$$+\sum_{a}^{N_F} W_{ia}^{RF} s_a, \tag{4}$$

$$\tau \frac{dy_a}{dt} = -y_a + \sum_{b=1}^{N_F} W_{ab}^F s_b + \sum_{i=1}^{N_G} W_{ai}^{FR} r_i, \tag{5}$$

The activity of architecture C is described with equation 1. The addition of equation 3 to equation 1 results in architecture A. The addition of equation 4 to equation 1 results in architecture B. Equation 5 provides the dynamics of the control network where $s_a = tanh(y_a)$

Table 1 Parameters used for the different architectures.

	Architectures			
	A	B	C	D
Reservoir size (N_R)	1000	2000	1000	1000
Number of input units (N_I)	0	0	0	1
Learning rate (α)	1	1	1	1
p_R	0.1	0.1 ($P_F = 1$)	1	0.1
p_{zR}	1	0.5 ($P_{FR} = 0.5$)	1	1
p_{Rz}	1	0 ($P_{RF} = 0.5$)	0	0
g_R	1.5	1.5 ($g_F = 1.2$)	1.5	1.5

is equal to the firing rate of $y_a(a = 1, ..., N_F)$. The control network's weight matrix W^F is of size $N_F \times N_F$ where the non-zero values were drawn from a Gaussian distribution with a mean of zero and standard deviation equal to $g_F/\sqrt{p_F N_F}$, where g_F is the gain of the synaptic weights and p_F is the proportion of non-zero values in the matrix. Connections from the reservoir to the control network W^{FR} were initially set to zero and the non-zero values of W^{RF} were drawn from a uniform distribution with values between -1 and 1.

Innate training. This architecture is shown on Fig. 1(d) and is described by equations 1 and 2. The default parameters used in architecture D can be found in Table 1. The input weights W^{RI} were drawn from a Gaussian distribution of mean zero and unit standard deviation and the values of W^{zR} were drawn from a Gaussian distribution with mean of zero and standard deviation of $1/\sqrt{N}$. The value of the active input unit for a given trial was held at zero except at the time of stimulation where it had an amplitude of 5 (arbitrary units) for 50 ms.

Driven model. The architecture E is shown in Fig. 7(a) and the parameters of this model are the same used for innate training (architecture D), except for a different input pattern and the absence of training of the reservoir's connections. The input layer is made of distinct oscillating units of different frequencies and a constant amplitude. The initial phase of the oscillators was chosen randomly for all simulations, and unless specified, their frequencies where randomly selected from a uniform distribution with a minimum of 1 Hz and a maximum of 5 Hz. Non-zero values of the input weights were drawn from a Gaussian distribution with a mean of zero and standard deviation equal to $g_{RI}/(N_I p_{RI})$, where g_{RI} is the gain of the input weights, N_I is the number of input units (oscillators) and p_{IR} is the proportion of non-zero values in the connection matrix. The default values of g_{RI} and p_{RI} are of 1.5 and 0.5, respectively.

2.3 Training tasks

Two different tasks were employed. In a first task, a signal composed of four sinusoidal waves was used as the target output. This signal was made of 12 cycles generated with sinusoids of same period but different amplitudes (Fig. 2(a)). We trained architectures A-C on this task, and assessed the performance of networks using the mean absolute error (MAE) between the output and the target values.

In a second task, termed the timing task, the readout target was determined by $f(t)$, set to a constant value of 0.2 until the start of the Gaussian curve with a peak centered at time t_{delay} of 1 second. The target trajectory was collected from the activity of the network from the end of the input pulse to the end of the Gaussian curve of the target function (1,150 ms after the end of the input pulse). Three different methods were used to assess the performance of the networks. The mean squared error (MSE) was used to compare the output and the target of the readout unit. We also computed the interval timing estimated by the network based on the time where it crossed a threshold and compared it to the actual peak of the target to obtain a timing lag. Finally, we computed the Pearson correlation coefficient between the target and the output. Even though the original model tolerates a rather large amount of external noise, we kept noise off across all of our simulations to isolate the effect of the perturbations under study. All error bars reported in the results below represent the standard error of the mean.

2.4 Learning algorithms

We used a powerful algorithm termed first-order reduced and controlled error (FORCE) learning (Sussillo and Abbott 2009) that can train networks to reproduce periodic patterns (e.g., a complex sinusoid) or chaotic signals for limited time intervals (e.g., a Lorenz attractor). This algorithm is based on the recursive least squares method (Haykin 2002), and was used to train all architectures, but the subset of synapses trained as well as the target function differed between architectures. All architectures had the following equation to adjust the readout weights:

$$W^{zR}(t) = W^{zR}(t - \Delta t) - e(t)P(t)r(t), \quad (6)$$

$$e(t) = W^{zR^T}(t)r(t) - f(t). \quad (7)$$

Where the error $e(t)$ was determined by the difference between the value of the readout unit obtained with the multiplication of the reservoir's activity with the weights W^{zR}, and the target function's value f at time

t. Each weight update was separated by a time interval Δt of 2 ms for all simulations. P is a running estimate of the inverse of the correlation matrix of the network rates r, modified according to equation 8 and initialized with equation 9.

$$P(t) = P(t - \Delta t) - \frac{P(t - \Delta t)r(t)r^T(t)P(t - \Delta t)}{1 + r^T(t)P(t - \Delta t)r(t)}, \quad (8)$$

$$P(0) = \frac{I}{\alpha}. \quad (9)$$

where I is the identity matrix and α is a learning rate constant. For architecture C, where the reservoir is subject to training, the whole connectivity matrix of the reservoir is modified with equation 6.

For innate training, the same version of FORCE learning described above was employed to train the reservoir as well as the readout unit. However, innate training requires an independent estimate of the inverse correlation matrix of the presynaptic inputs P for each plastic neuron in the reservoir:

$$W_i^R(t) = W_i^R(t - \Delta t) - e_i(t) \sum_{k \in B(i)} P_{jk}^i(t) r_k(t), \quad (10)$$

$$e_i = r_i(t) - R_i(t), \quad (11)$$

$$P_{jk}^i(t) = P_{jk}^i(t - \Delta t) - \frac{\sum_{m \in B(i)} \sum_{n \in B(i)} p_{jm}^i(t - \Delta t) r_m(t) r_n(t) P_{nk}^i(t - \Delta t)}{1 + \sum_{m \in B(i)} \sum_{n \in B(i)} r_m(t) P_{mn}^i(t - \Delta t) r_n(t)}. \quad (12)$$

where $B_i(i = 1, ..., N_{Plas})$ is the subset of presynaptic neurons in the reservoir (60% of the reservoir) and equation 11 represents their error on unit i. The term e_i is the error based on the discrepancy between $r_i(t)$ (activity of neuron i) and $R_i(t)$, the target activity based on the innate trajectory. Equation 12 is an altered version of equation 8 where each plastic unit (only plastic units have their synapses adjusted) has an independent matrix P, which are identity matrices initialized according to equation 9 (with a size equal to the number of presynaptic units k). The innate training of the recurrent weights was applied for 20 iterations of the reservoir trajectory, then the weights of the readout unit were trained for 10 iterations of the target output. The training started at the end of the input pulse until the end of the Gaussian peak of the target. As with FORCE learning, each weight update was separated by a time interval Δt of 2 ms for all simulations.

3 Results

We begin by showing the impact of the different types of perturbation on the performance of two training schemes described in Methods (standard FORCE and innate training). First, we show that local perturbations have a marked impact on the trained models. Then, we introduce an alternative model that can overcome the issue of robustness while increasing the performance of the network on a task of interval timing.

3.1 Structural perturbations in standard FORCE

As a starting point, we examined how small structural alterations can impact the behavior of a network with standard FORCE learning (Sussillo and Abbott 2009). The first type of perturbation that we investigated is the clamping of individual neurons. This was achieved by setting the activation as well as all afferent and efferent connections of a given subset of neurons to zero. We began by training a network with each architecture (Fig. 1(a-c)) to replicate a jagged sinusoidal signal. We used the MAE between the output and the target in order to evaluate training performance. Once the network was trained (a fixed number of cycles of the sinusoidal pattern, here 12 cycles), all architectures attained a similar level of error on a distinct testing phase of the same length (Fig. 2(a)) but without further modifications (for 100 trials, architecture A: average MAE of 0.06, SEM of 0.01, architecture B: average MAE of 0.06, SEM of 0.01 and architecture C: average MAE of 0.05, SEM of 0.01). However, only networks with a MAE below 0.01 were selected for further analysis. In Fig. 2(b), we clamped the activity of one random neuron from the reservoir (out of 1,000 neurons) to zero from architecture A, and left the readout neuron untouched. We then allowed the network to try and produce the jagged sinusoidal signal without further training. On most trials, this small perturbation produced catastrophic disruptions in network activity. In some rare cases, the networks were able to retain some of their performance (Fig. 2(c)), but were still unstable.

All three architectures were highly vulnerable to the clamping of a single, randomly selected neuron (out of a total of 1,000, representing 0.1% of all neurons in the reservoir). To further show this effect, we calculated the average MAE on multiple trials for each architecture (Fig. 2(d-f)). For each of the three architectures tested (A,B and C), five networks were generated (represented by different colors) from which the activity of a different neuron was clamped to 0 on each trial, for a total of 1,000 trials per network (one for each neuron).

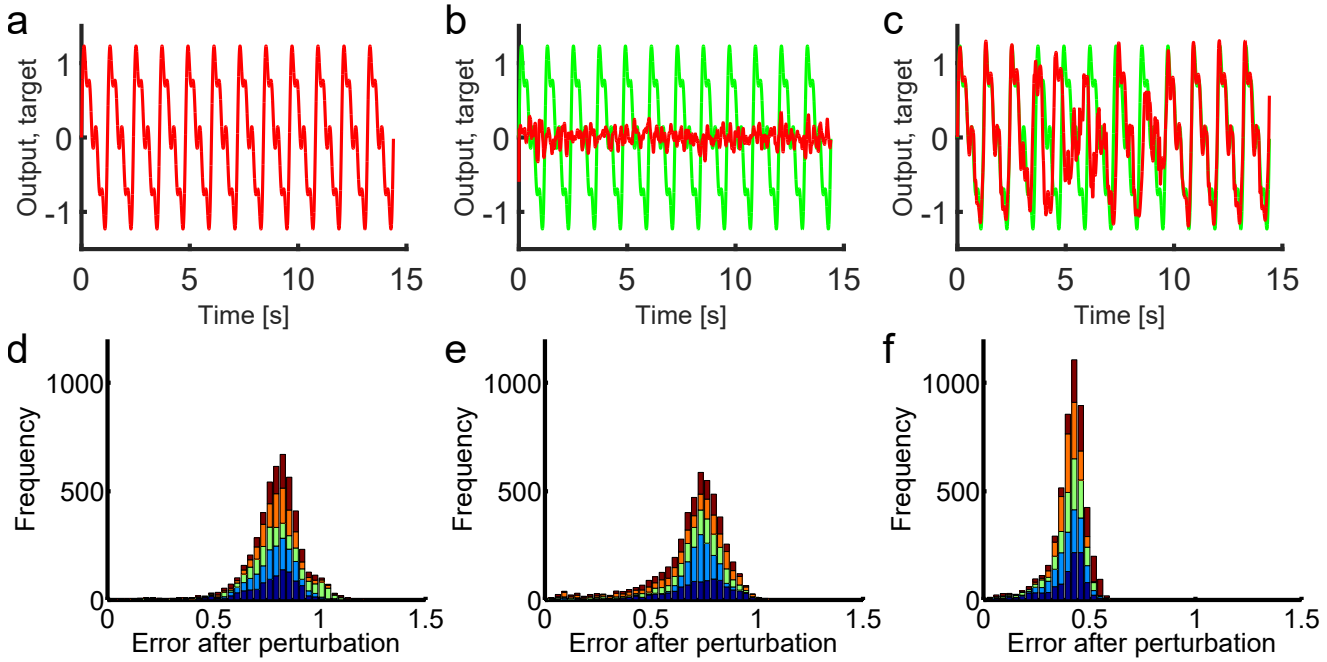

Fig. 2 Activity of intact and damaged networks. Top: the output signal of a network is in red and the target sinusoid is in green. **a** The output of a network once the training phase has ended is almost perfectly overlapping with the target. **b,c** represent the outputs of selected networks with N-1 neurons of architectures A (MAE: 0.62 and 0.40). **d,e,f** show the distribution of MAE obtained from 5,000 trials with architecture A, B and C, respectively. The average MAE are of 0.79, 0.69 and 0.41, respectively. When normalized with the amplitude of the output function, the averages are of 0.61, 0.53 and 0.56. Each color represents the distribution of a different network, from which each neuron of the reservoir was clamped in turn. Each network has an initial MAE below 0.01.

The average MAE over the 5,000 trials with architectures A, B and C were of 0.79, 0.69 and 0.41, respectively. The amplitude of the target function of architecture C was almost halved compared to the two others to stay consistent with (Sussillo and Abbott 2009) for fair comparison. That explains the shift of the distribution of Fig. 2(f) toward lower values. Because the MAE is dependent on the amplitude of the target function (greater amplitude leads to higher values of MAE), we normalized the average error based on this value (i.e., average MAE divided by the amplitude of the target). This yielded the following normalized average MAE: 0.61, 0.53 and 0.56 for architectures A-C, respectively. After normalization, the performance of architecture C was not significantly different from architecture A and B.

In summary, all architectures used with the standard FORCE learning procedure were highly vulnerable to the clamping of a single neuron. Next, we examined the sensitivity of networks with innate training. Even though this model is also trained with the FORCE algorithm, it relies on different principles to generate stable reservoir trajectories.

3.2 Structural perturbations in innate training

We tested innate training models on a timing task where the network was taught to remain quiescent and to peak after an interval of one second (Fig. 3(a-d)). Once the reservoir's trajectory was trained for 20 iterations and the readout unit was trained for 10 iterations, the network's output had an almost perfect fit with the target function (Fig. 3(a)). We then clamped different subsets of neurons within the reservoir and computed the MSE between the output and the target of the readout unit. The trained networks lost their ability to output a stable value between the input pulse and the target peak after a single neuron was clamped in the reservoir (Fig. 3(b)). Fig. 3(a-d) shows the output of a trained network on five different trials (red curves) where a different subset of neurons (Fig. 3(b-d), number of neurons of 1, 2 and 5, respectively) has been clamped. Nevertheless, some networks could produce a peak in their output after the appropriate interval despite the perturbation.

We then calculated the interval timing based on the time when the output crossed a threshold. In order to avoid penalizing a shift in the amplitude of the output value after the perturbation, we computed an independent threshold for every perturbation for each network. To find the most appropriate thresholds, we computed

the average delay for every trial of a condition with a threshold that was increased from 0 to 1 (maximal value of the target function) with increments of 1e-03. The threshold associated with the best performance was selected. We also computed the success rate, where trials were considered successful when they fell within a 40 ms window centered on the target's peak.

The performance degraded quickly as more neurons were clamped in the reservoir (Fig. 3(e,f)). The error values started to plateau when a little more than 1% of the neurons were clamped. The MSE between the output and the target increased monotonically as further neurons were eliminated (Fig. 3(f)). In addition to clamping entire subsets of neurons, we verified the impact of setting the weights of individual synapses to zero in the network (Fig. 4(a-c)). On the timing task, damaged networks were near the maximal error value when 1% of the synapses were clamped (Fig. 4(a)). The performance curve of networks with perturbed reservoirs with 0% to 1% of synapses set to zero was similar to the performance of networks from which 0% to 100% of the neurons connecting the reservoir to the readout unit were set to zero (Fig. 4(d-f)). This shows that in terms of proportion, the readout synapses are much less important for the performance of the network than the reservoir's synapses, as long as the trajectory in the reservoir remains intact.

We also tested the impact of Gaussian structural disruptions distributed across all the reservoir's synapses. A perturbation vector was obtained by multiplying the values of all non-zero synapses with a given proportion of disruption. This vector was then shuffled and added to the values of the non-zero synapses. After this manipulation, the performance was a lot less degraded than when comparable number of synapses or neurons were removed (Fig. 4(g-i)). The Δw (i.e., the total quantity of change in weights between the original and the perturbed network) is exactly the same for a given proportion of perturbation. For instance, when 1% of all synapses are removed or the value of all synapses are changed by 1% of their total value, the difference in total weight between the original and modified connectivity matrix (Δw) will be the same, on average. This greater structural robustness for global perturbations was also observed with standard FORCE (Fig. S1). This outlines an important difference between the two types of perturbation that is not captured by the mean amount of change but by the way it is distributed.

So far we designed perturbations on networks with a size of 1,000 neurons to match most of the results from (Laje and Buonomano 2013; Sussillo and Abbott 2009). When applied to networks of different size, perturbations were affected by the absolute number (rather

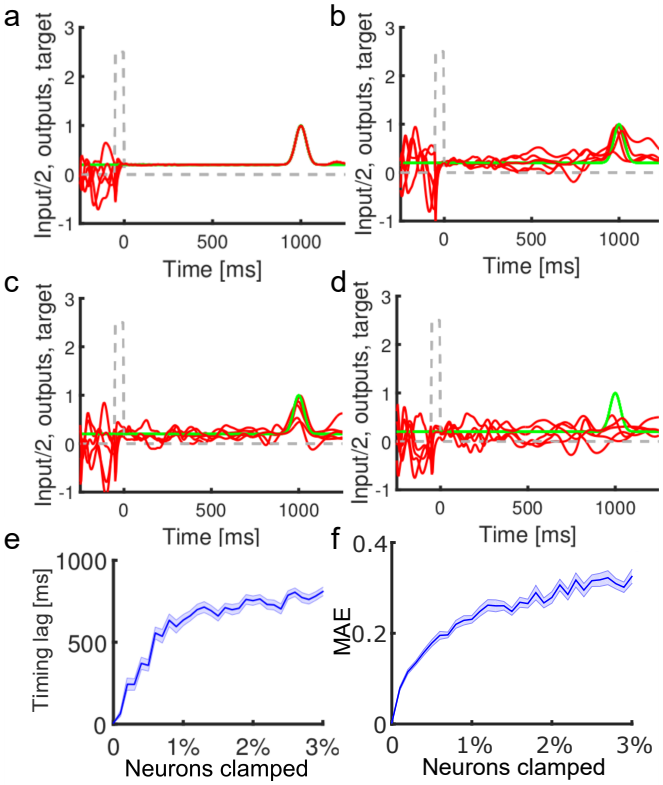

Fig. 3 Performance of innate training on the timing task after perturbations. Each reservoir has a size of 1,000 neurons. An external input (grey dashed line) act as a reset on the network's state, which triggers the trajectory previously stabilized in the reservoir. The network is trained to peak 1,000 ms after the end of the input signal. **a,b,c,d** show the outputs of networks with N, N-1, N-2 and N-5 neurons in the reservoir, respectively. **e** Lag between the time the output crossed the threshold and the peak of the target. **f** MAE between the output and the target signal. Shaded areas represent the standard error of the mean.

than the proportion) of neurons clamped. The average MAE for different number of neurons clamped decreased slightly as the total number of neurons in the reservoir was increased with architecture A (Fig. S2). But the smallest perturbation on the largest reservoir (1 neuron clamped out of 10,000) still led to catastrophic disruptions of the network's output (average of 0.5 MAE). With architecture D, the average error of a perturbation was not related to the size of the reservoir in the testable range (larger networks are harder to train). In other words, clamping a given number of neurons had a relatively similar impact on performance regardless of the size of the networks.

In summary, structural perturbations of the reservoir had a devastating impact on the task learned by the readout unit when applied in a local fashion (i.e. synapses set to zero, as opposed to the global Gaussian perturbations). In the following section, we address in

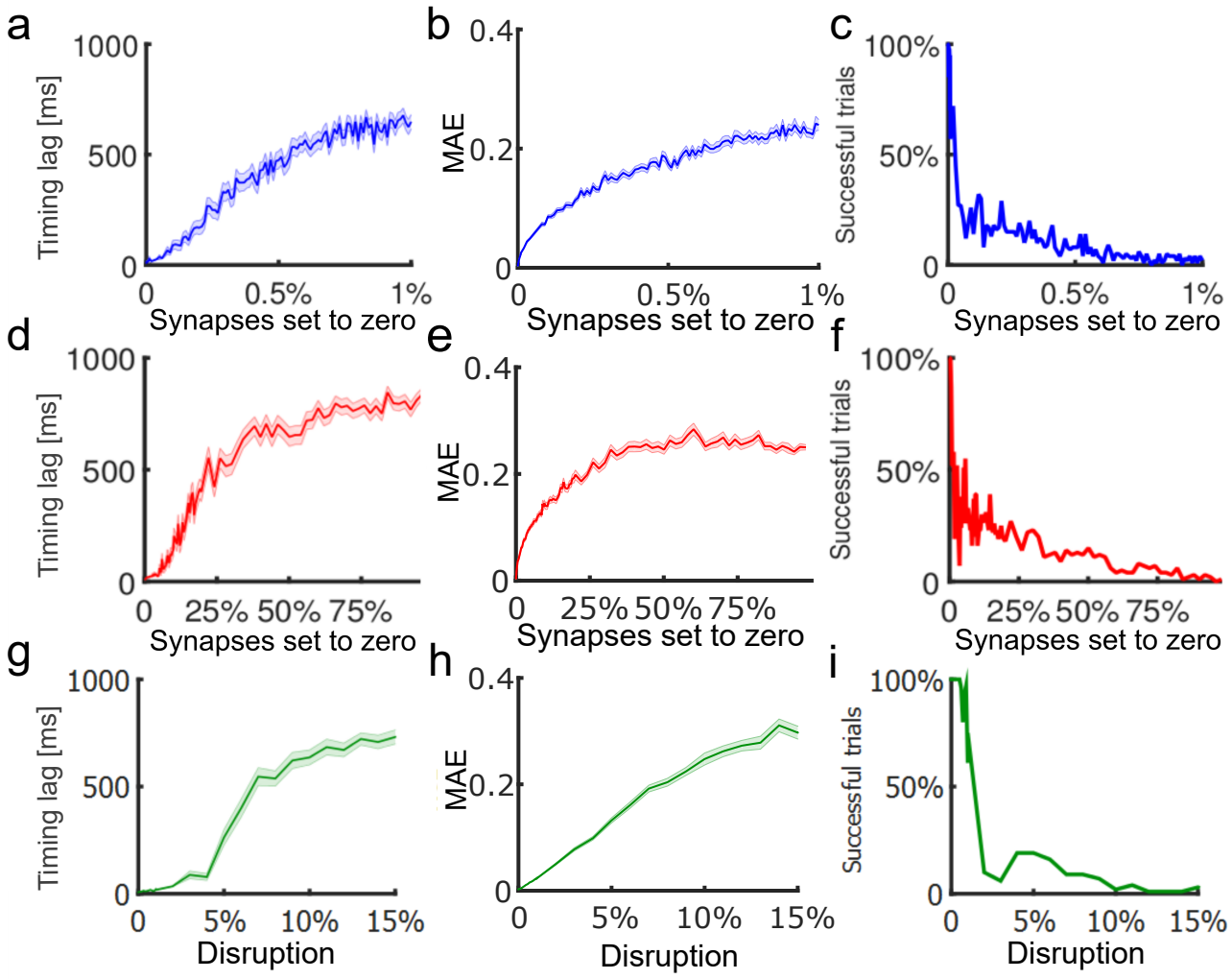

Fig. 4 Performance on the timing task after different types of perturbations. Impact of clamping individual synapses of the reservoir and of the readout unit, as well as a Gaussian perturbations on all synapses in the reservoir. **a,b,c**: disabling synapses from the reservoir. **d,e,f**: disabling synapses that connect the reservoir to the readout. **h,i,j**: disruption of synaptic weights. The leftmost column of the figure shows the lag between the estimated timing of the output and the target. The second column shows the MAE between the output and the target. The rightmost column shows the proportion of trials for which the timing estimate fell within a 40 ms window centered on the target's peak. Shaded areas represent the standard error of the mean.

detail the impact of such perturbations on the network's dynamics and activity features.

3.3 Structural perturbations and eigenvalues

We computed the eigenvalues of the random networks connectivity matrix to gain some insight on the effect of the perturbations. More specifically, we aimed to characterize the contrast between the dramatic impact of the structural perturbations that we investigated to the milder effect of similar perturbations reported in previous studies (Laje and Buonomano 2013; Sussillo and Barak 2013; Sussillo et al 2015), which used global perturbations (Gaussian perturbation to the reservoirs connectivity matrix) as opposed to the local perturbations that we introduced.

We computed the eigenvalues of random connectivity matrices (N = 1,000), where each non-zero connection is drawn from a Gaussian distribution ($\mu = 0$, σ (gain) $= 1.5$), and are equivalent to the connectivity matrices of untrained reservoirs. Therefore, we compared the impact of different types of structural perturbations outside of a training task. With this procedure, it is expected that the distribution of eigenvalues lies within a circle with a radius equal to the gain of the network's connections (see e.g. Bai (1997)). The deviation between the eigenvalues of the original and the damaged network (one neuron clamped) is quite large

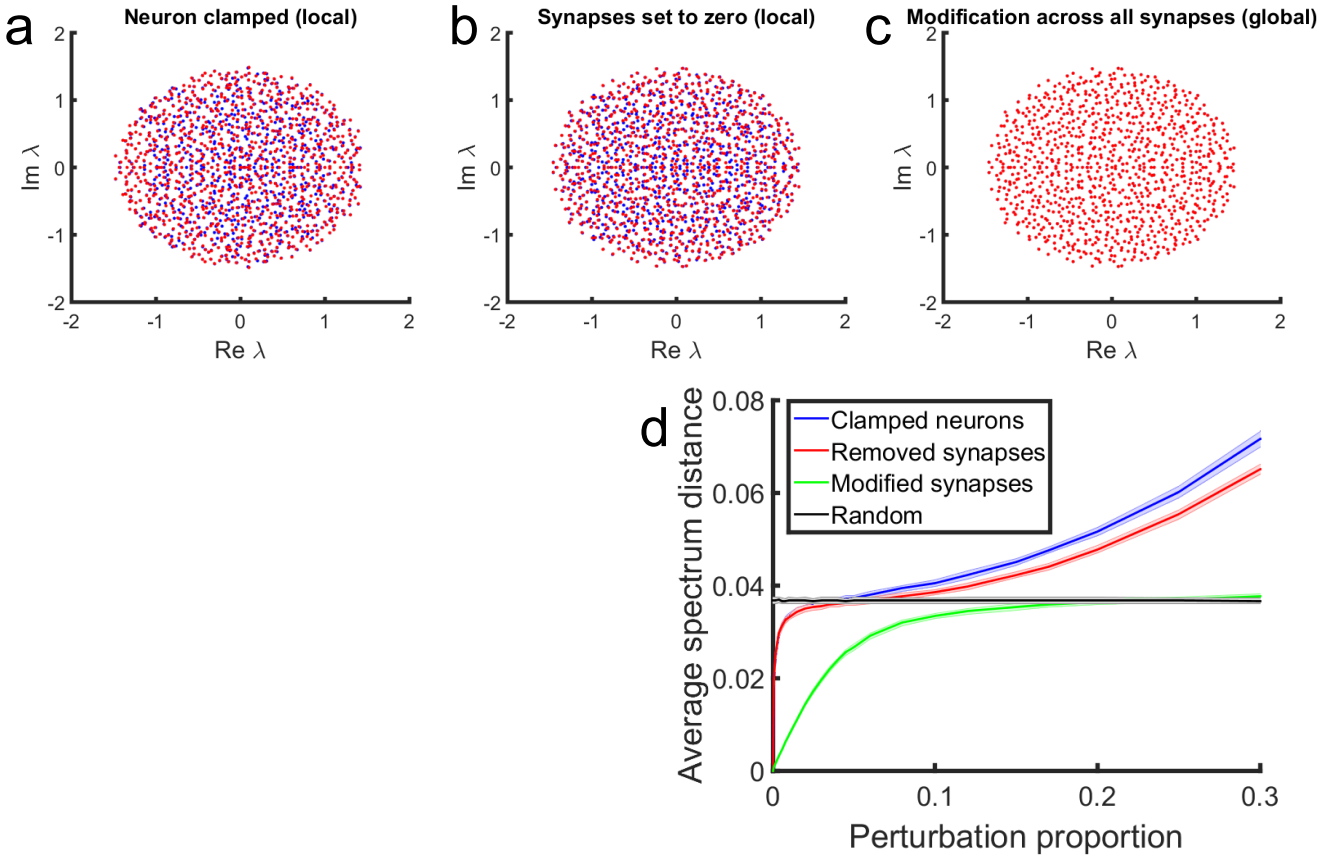

Fig. 5 Eigenvalues of perturbed networks. **a** Eigenvalues of a random network (blue) and of the same network with N-1 neurons (red) ($\Delta w = 21$). **b** Eigenvalues of a reservoir (blue) and of the same reservoir where 0.1% of the synapses have been set to zero ($\Delta w = 11.40$). **c** Eigenvalues of a reservoir (blue) and of the same reservoir where all non-zero synapses have been disrupted by values following a Gaussian distribution (red) for a total disruption equal to 0.1% of their total weight ($\Delta w = 11.87$). **d** Average distance between the eigenvalues of the original and the perturbed network as the size of the perturbation increased. Each proportion represents either the number of units removed or the weights changed. The random condition is constant across all levels of perturbation and shows the distance expected between two random networks. Shaded areas represent the standard error of the mean.

(Fig. 5(a)). This effect explains the discrepancy between the output and the target of the two conditions, as the eigenvalues govern the dynamics of the network.

To gain a better understanding of the different perturbations, we also tested the impact of removing synapses and of Gaussian perturbations to every synapses. We clamped 0.1% of the neurons (1/1,000 neurons, Fig. 5(a)), removed 0.1% of the synapses (100/100,000 synapses, Fig. 5(b)) and changed 0.1% of the synapses' weights (Fig. 5(c)). This results in the same amount of change (Δw) for the last two conditions ($\Delta w = 11$), and about double the amount of change when one neuron is clamped, because both the efferent and afferent connections are removed. In addition, clamping one neuron effectively removes one dimension from the eigenvalue spectrum (i.e., it reduces the rank of the connectivity matrix by one). When visually inspecting the eigenvalue spectrum, it is obvious that removing individual synapses and neurons has a much greater impact than splitting the same amount of change across all synapses.

We then formally compared the distance between the original and the perturbed eigenvalue spectrum. To do so, we computed the average distance between each eigenvalue of the original network and its closest eigenvalue in the perturbed network. For each level of perturbation and each type of perturbation, we generated 50 pairs of original/perturbed networks and computed their average distance (Fig. 5(d)). Different profiles emerged for the different types of perturbation. Local perturbations created large disruptions on the eigenvalue spectrum, where the distance between the original and perturbed networks reached the same value as for two independently generated random networks when about 5% of their units were removed. The curve for clamped neurons had a slightly larger distance because the number of dimensions decreased, whereas it remained constant when a proportion of synapses were

removed (until extreme values are reached). This contrasts with the impact of the change in synaptic weights, where between 15% and 20% of their value had to be changed to reach the same distance as would be expected between two random networks. It is interesting to note that even if the Δw is almost doubled when the same proportion of neurons as synapses was removed, their impact on the spectrum distance was similar until it reached the average distance between randomly generated networks, which is concordant with their similar impact on the timing task with architecture D (Fig. 3(e) and Fig. 4(a)). Finally, as neurons and synapses were removed, the radius of the eigenvalue spectrum shrank, which explains the increasing eigenvalue distances past the random condition. Because we added the equivalent of Gaussian perturbations with a mean of zero on the network's synapses, the radius of the eigenvalue spectrum remained the same.

Difference in eigenspectra should manifest in differences of network trajectories. With innate learning, the network is able to return to the trained trajectory after the injection of a substantial pulse in the reservoir (Laje and Buonomano 2013). The effect of this form of perturbation might be akin to distributing the structural perturbation across the whole reservoir (Fig. 4(g-i)). Based on the performance of the network on the trained task, the reservoir was no longer able to return to the original trajectory when a structural perturbation targets a specific portion of the network. Further analysis shows that despite the difference between the activity of the original and perturbed reservoirs, the trajectories of damaged networks appeared to be stable, because the new trajectory was still repeatable across trials (Fig. S3). In summary, the eigenvalues analysis confirms the large impact of local structural perturbations on the network independently of the training algorithms.

3.4 Impact of network connectivity

We explored alternatives to the standard patterns of connectivity in reservoirs that could improve the tolerance of these networks to structural perturbations. In previous simulations with architecture A, we used a sparse connectivity where only 10% of all possible connections were set to non-zero values (the standard value for most reservoirs models). However, the total number of connections that each neuron makes to its counterparts might have an impact on their importance on the overall network dynamics. We tested a lower range of sparseness that remained biologically relevant (1% to 10%), and found no correlation between sparseness and resistance of the networks to structural perturbations (r = -0.02, p = n.s.).

Next, we looked into different patterns of connectivity as potential solutions to the sensitivity of the model to structural perturbations. In typical reservoirs models, the number of connections of each neuron follows a normal distribution and each neuron has the same probability to connect to other neurons in the reservoirs. However, several studies suggest that the number of synapses per neuron in the cortex might follow either a power-law (slope of -1.3) (Bonifazi et al 2009) or exponential distribution (Perin et al 2011). This type of connectivity has been linked to higher resilience of networks to random failure (Rubinov and Sporns 2010). We tested with architecture A whether these types of connectivity could help improve the resistance of reservoirs to structural perturbations. We applied these distributions to the afferent and the efferent connections separately. This means that when one of these distributions was used for either the afferent or efferent connections, the other type was kept normally distributed.

Our results show that exponential and power-law (Fig. 6(a), left and middle) distributions slightly improve the resistance of the networks (larger left tail of the distributions compared to Fig. 2(d)), but only when applied to afferent connections. In this configuration, most neurons received a very small number of connections from other neurons whereas a few neurons received a high number of connections. However, the MAE of most trials was still high after clamping a single neuron.

3.5 Network modularity

There is a body of work that shows that the brain is modular on different scales (Meunier et al 2010). On the meso scale, the cortex is arranged in a multitude of cortical columns (Buxhoeveden and Casanova 2002; Mountcastle 1997). On a higher level, the cerebral cortex has been shown to be organized in different subregions that have non-symmetrical ways to communicate with each other (Keller et al 2014). Modular connectivity has been associated with neurobiologically realistic regimes of activity in simulated networks (Rubinov et al 2011). Additionally, clustered and modular reservoirs have been linked to increased stability in echo-state networks (Jarvis et al 2010; Li et al 2015). If cortical circuits were to work accordingly to reservoir computing principles, the reservoirs might have to be distributed across different sub-regions, or compartmentalized within each region to avoid the propagation of the error caused by the failure of one of its component.

We started by making networks made of modules without connections between them, then we progres-

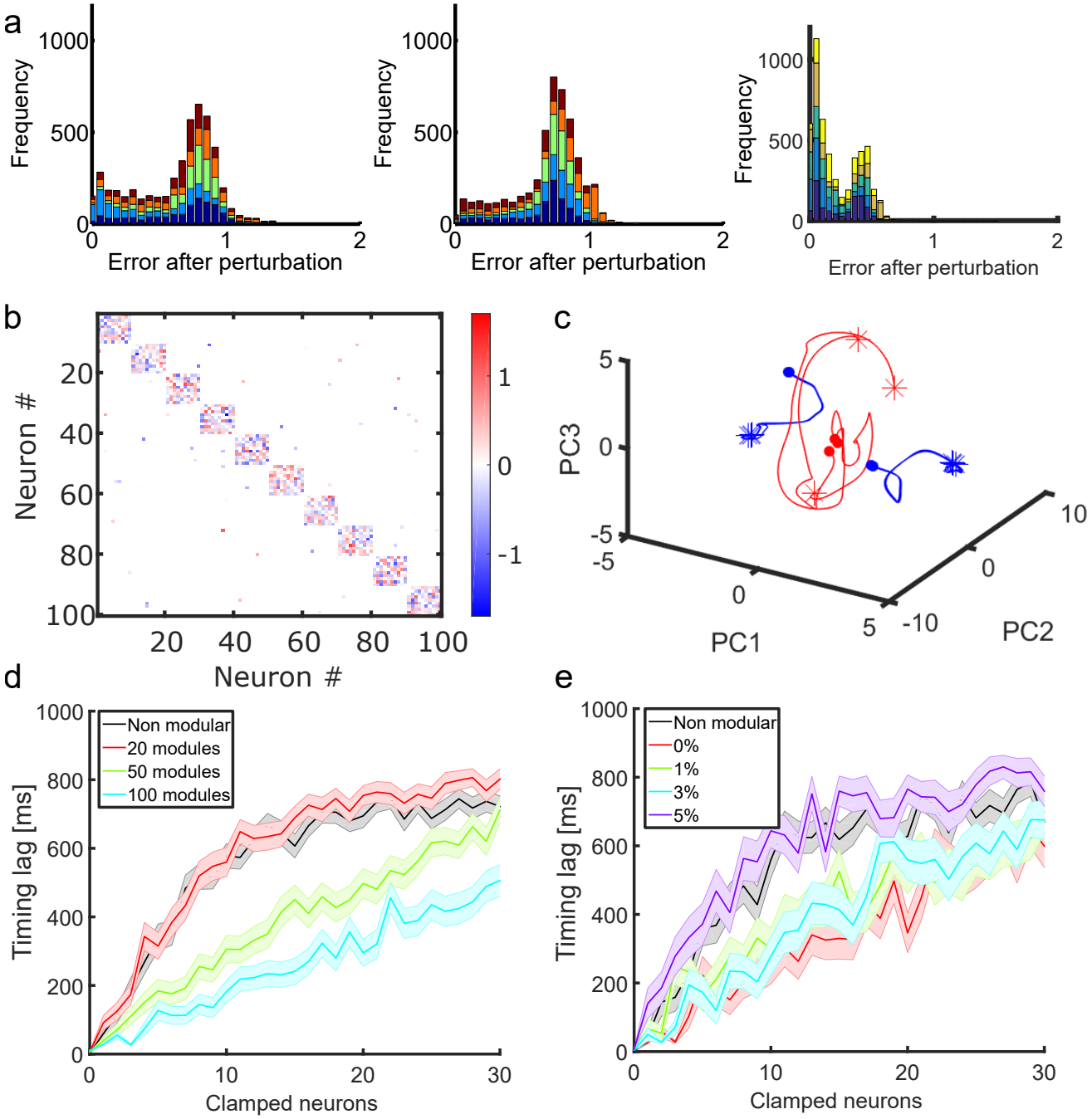

Fig. 6 Error distribution of networks with an exponential, power-law and modular connectivity. **a** All distributions were obtained from architecture A (5,000 trials each). Each color represents the MAE distribution from a different network, from which each neuron was clamped once. Error distribution of networks (N-1) with an exponential distribution (left) of afferent connections and a power-law distribution with a slope of -1.3 (middle) of afferent connections, and (right) a reservoir of 2,000 neurons with 100 modules (of 20 neurons each). **b** Connectivity matrix of a toy example for a network with 10 modules of 10 neurons each. For our analysis, the size (s) and number of modules (n) were the following (s by n): 1,000 by 1, 50 by 20, 20 by 50 and 10 by 100. Modules were originally fully connected without self-connections (diagonals set at zero). Then, 5% of these connections were rewired to connect modules to each other. **c** Toy example of the trajectories of different modules of a reservoir (100 by 10, probability of connectivity within modules = 1 (without self-connections), probability of connectivity between modules = 0.03). Red traces show the trajectories of the perturbed module (circle = start, star = end) for 300 ms. Blue traces show the trajectories of two modules that are not directly perturbed. **d** Timing lag of the output based on the time it crosses a threshold for reservoirs with different configurations of modules. **e** Timing lag of modular networks of size 10 by 100 where 0%, 1%, 3% and 5% of synapses that were originally connecting neurons within modules have been rewired between modules. Shaded areas represent the standard error of the mean.

sively added inter-module connections to assess the impact of modularity under more realistic conditions. Fig. 6(b) shows a toy model of a reservoir made of several modules that were initialized as smaller random networks. When a neuron was clamped in one module, only this module was affected by the perturbation and the other ones remained intact (Fig. 6(c)). Traces in blue show the trajectories (on three different trials) of two modules that were not directly affected by a perturbation. Traces in red show the trajectories of the module where one neuron was clamped. This illustrates the resistance of spared modules to noise in the perturbed modules, as long as the number of connections between modules remains low. When implemented with the models, there was indeed an improvement in the resistance to perturbations on the complex sinusoidal task with architecture A (Fig. 6(a), right). There is also an improvement with architecture D and the timing task, where Fig. 6(d) illustrates the impact of perturbations on different module size (reservoir sizes are kept constant). Of course, performance degraded as we added inter-module connections. Fig. 6(e) shows the impact of adding connections between modules, where 0%, 1%, 3% and 5% of the synapses within each modules were randomly rewired to neurons from different modules. While modularity reduces the vulnerability, it does not eliminate the problem and requires the network to implement strict parameters values to improve resilience.

3.6 Driving reservoirs with oscillations

The above findings show that recurrent networks operating in the chaotic regime are extremely sensitive to local structural perturbations. Based on these results, we conclude that it is unlikely that recurrent networks of firing rate neurons can produce reliable trajectories as autonomous systems (i.e., in absence of ongoing external drives) with the reservoir models tested in this study. This is in accordance with results from studies with recurrent networks of spiking neurons (Banerjee et al 2008; London et al 2010). In this section, we describe an alternative autonomous model where the reservoir is non-autonomous but is driven by an autonomous input layer. For the duration of a trial, this input layer is continuously stimulating the reservoir with an ongoing periodic drive where each input unit behave as an independent oscillator. Such oscillatory dynamics can be endogenously generated by small neural circuits with precise connectivity between their excitatory and inhibitory components (Yuste et al 2005). This framework allows the production of stable neural trajectories that are the result of the interaction between the periodic input, the state and the structure of the reservoir. This model is resilient to local and global perturbations and conserves the computational properties offered by chaotic neural networks (Bertschinger and Natschläger 2004).

Previous work showed that chaotic networks can be stabilized with an oscillatory input (Rajan et al 2010). However, an important limitation of this approach is that the duration of the resulting neural pattern can't exceed the period of the oscillation. Alternatively, in our model, combining a population of oscillatory units provides a major advantage: it increases the duration of the pattern that the input layer can produce. The mechanisms underlying this effect can be explained by the residue number system (Soderstrand et al 1986). In short, oscillators of different periods can repeat their combined initial state after a delay equal to the least common multiple of all their combined periods, as long as the ratio of their periods is a rational number. This creates a powerful mechanism by which the network creates meaningful and robust neural trajectories.

Fig. 7(a) illustrates the general mechanisms of this model. Each input unit oscillates at a constant frequency throughout the duration of the stimulation. The combination of oscillators with different periods (that are not multiples of each other) yields a much longer input period than the period of the slowest oscillator. For instance, the least common multiple of the two oscillators on Fig. 7(a) is 2,000 ms, which means the trajectory will repeat itself every two seconds. It is well-known mathematically that the combined activity of multiple oscillators can create very rich dynamics.

To visualize this, consider two linear oscillators whose phase evolutions are given by $\theta_1(t) = w_1 t (\%2\pi)$ and $\theta_2(t) = w_2 (\%2\pi)$ respectively, where θ represents an angle on the unit circle $[0, 2\pi]$. The combined state of the pair of oscillators, $(\theta_1(t), \theta_2(t))$ is a two-dimensional coordinate on the Cartesian product of two unit circles, or in other terms, a 2-Torus. Suppose, without loss of generality that the pair of oscillators both start at $\theta = 0$. The question is: will their combined trajectory ever wind back to (0,0)? If so, how many cycles of each oscillator will it take?

For both oscillators to come back to the origin at the same time means that there exists a single time t such that $w_1 t$ and $w_2 t$ are both exact multiples of 2π. Suppose then that m and n are integers such that $w_1 t = 2\pi m$ and $w_2 t = 2\pi n$, then it follows that $w_2/w_1 = n/m$, which is only possible if the fraction $w2/w1$ is a rational number. If this is the case, then the smallest valid integers m and n correspond to the respective number of cycles of each oscillator that combine to create a periodic trajectory on the torus. If the ratio of frequencies is not rational, it can be shown that the trajectory will

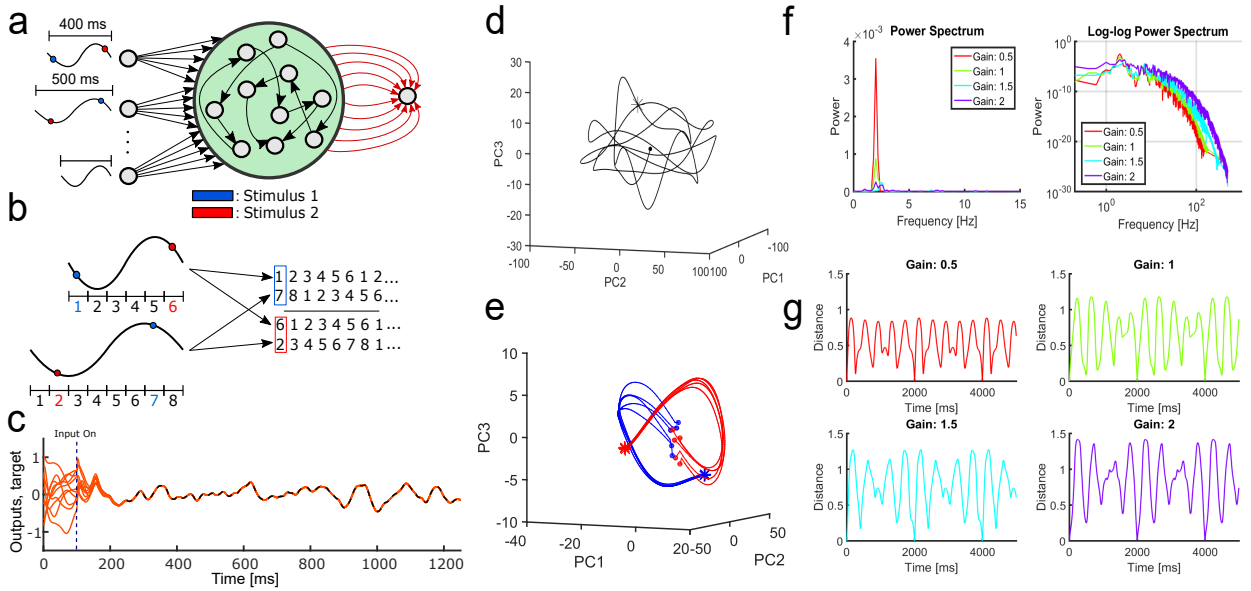

Fig. 7 Generation of stable trajectories with oscillators. **a** The activity of the oscillators is controlling the dynamics of the reservoir. Multiple oscillators can be used to increase the length of the trajectory they can generate. The two first oscillators in this example can create a trajectory that will not repeat itself until after 2,000 ms (least common multiple of 400 ms and 500 ms) of ongoing stimulation. The stimuli (e.g., a go cue) can be seen as triggers that align the phases of the oscillators in the same configuration on every trial. **b** The phase configuration can be created by an input that aligns the oscillations at some given phases corresponding to the segment of the trajectory that is required. **c** Outputs of the driven network in **a** that was trained on a randomly generated time-varying function with frequency components below 15 Hz.**d** Projection of the reservoir's activity on the principal components when driven by the two first oscillators. The trajectory repeat itself after 2,000 ms (green arrow). The initial segment before the green arrow represents the convergence of the reservoir's state to the dynamic attractor created by the input. **e** Five trials with random initial conditions (200 ms) created by the two stimuli. The filled circles represent the initial states and the stars mark the end of the 200 ms stimulation. **f** Left: power spectrum of the network activity. Right: power spectrum in log-log coordinates. **g** Distance between a random time step along the trajectory and all subsequent steps (average for all N neurons of the reservoir) for the different network gains. The trajectory is repeated every 2,000 ms, and the average distance between each point on the trajectory is increased with the network gain.

never repeat itself and that is will densely fill the entire torus over time (see e.g. Brin and Stuck (2002)). It is easy to see how this mechanism generalizes to the combination of multiple oscillators. The rich behaviour that emerges from the joint dynamics of uncoupled oscillators is a powerful mechanism that can be used in a variety of contexts (Buhusi and Meck 2005; Miall 1989).

The injection of such periodic activity in reservoirs requires no training of their recurrent connections, and allows the production of an unlimited number of trajectories where their total length is dependent on the input and not on the network (Fig. 7(a,b)). This is a major improvement compared to the innate training where only innate trajectories of the network could be stabilized, and those trajectories were greatly limited in duration (about 8 seconds) and quantity. The weights of the connections of the oscillators were drawn from a Gaussian distribution and were sparsely connected to the reservoir. Fig. 7C shows the output of a readout trained to produce a randomly generated time-varying output (white noise low pass filtered at 15 hz). Each trace shows a given trial with random initial conditions where the oscillators are turned on after 100 ms (blue dashed line) and the target output starts at 250 ms (black dashed line).

Fig. 7(d) shows one complete trajectory, based on the two first oscillating units of Fig. 7(a). After a transient segment where the network converges from its initial state to the dynamic attractor (green arrow) created by the injection of the oscillating input in the reservoir, the neural trajectory will repeat itself every two seconds as long as the oscillators are active and keep their relative phase shift. Some segments of this full trajectory can be evoked by stimuli that either reset or activate the oscillators at given phases (Fig. 7(b)), or that turn them on sequentially in order to match some predetermined phases shifts. Every time the oscillators are initialized at some predetermined phases, they can be used to evoke segments of the full trajectories (Fig. 7(e)), that can then be decoded by a readout unit.

The projections of the oscillators' activity in a recurrent network serves two principal roles. First, it creates a high-dimensional representation of the low-dimensional input (Häusler et al 2003). Secondly, it takes advan-

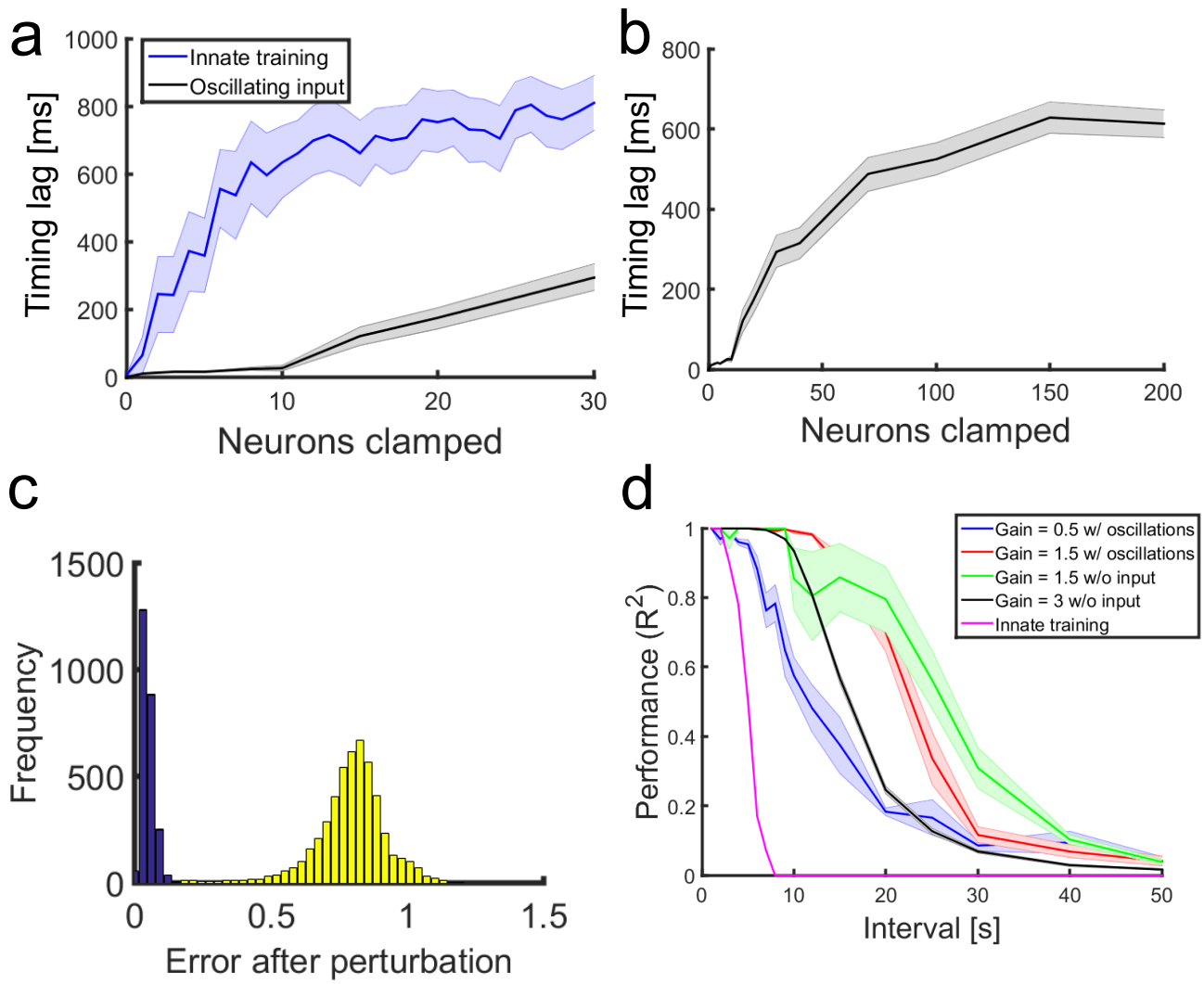

Fig. 8 Performance of the driven model. **a** Timing lag between the output and the target after clamping different numbers of neurons from the reservoir. **b** Timing lag for the driven model after greater proportions of neurons have been removed from the reservoir. The network reaches random performance (500 ms lag) after about 75 neurons out of 1,000 are clamped, compared to 8 neurons for innate training. **c** Error on the complex sinusoidal task after clamping a single neuron of the new model (blue, average MAE of 0.05) and architecture A (yellow, average MAE of 0.79). The frequency of the oscillators was set manually to match the length of a full cycle of the complex sinusoid (1.2 seconds). **d** Performance (R^2) on the timing task for different timing interval lengths. Blue/Red: the network is driven by multiple oscillators, with reservoir gains of 0.5 and 1.5 respectively. A higher gain increases the network performance. Green/Black: performance of architecture D with no recurrent training where the trajectory of the reservoir is repeated by resetting the initial state of the network to the same values on each trial. Magenta: performance of innate training. For **a**,**b** and **d**, we generated 10 networks (per conditions for **d**) where the frequencies of the oscillators were drawn for a uniform distribution with a minimum of 1 Hz and a maximum of 5 Hz. For **a** and **b**, we clamped 10 subsets of neurons per network, for a total of 100 tests per data point. Shaded areas represent the standard error of the mean.

tage of the recurrent connections of the network to decorrelate the highly correlated inputs that each neuron receives. Fig. 7(e) shows that as the network gain was increased, the peak in the power spectrum (2 Hz and 2.5 Hz based on the example of Fig. 7(a)) decreased, while the power of the other frequencies increased, which shows that the overall correlation of the network decreased. Fig. 7(g) shows the distance (average for all neurons of the reservoir) between a randomly selected time step of the network's trajectory and all subsequent time steps. The distance reaches 0 every 2,000 ms for all conditions, which shows that the network is back to its initial state. However, as the network gain increased, the overall distance between this point and the other states of the network increased, which suggest better discriminability between each step of the trajectory.

The resilience of the driven model was tested on the previous tasks (timing and complex sinusoid), although the activity of the reservoir driven with oscillations could be used with many other tasks where the output is a time-varying function. This model displayed considerably improved resistance to the structural perturbations that were completely shutting down the ability of the previous models to perform their task. Fig. 8(a,b) shows the performance after different numbers of neurons were clamped from the reservoir of the driven model and from reservoirs with innate training. With innate training, networks reached random performance when less than 10 neurons were clamped, whereas the driven model could sustain the loss of about 75 neurons before it reached the same disruption in performance. Fig. 8(c) shows the performance on 2,500 trials on the sinusoid production with the driven model where the period of the input oscillators matches the period of the target sinusoidal wave and 5,000 trials with architecture A (same as Fig. 2(d)). There is a large improvement with the new model where almost all networks were still successful after the perturbation compared to architecture A where no networks were able to sustain the loss of a single neuron.

The maximum length of the timing interval is significantly greater with the driven model than with innate training. To get a reliable estimate of the capacity of the new model, we generated the performance curve of different interval lengths with the spontaneous activity of reservoirs that was replayed multiple times with the exact same initial conditions and without noise. The performance of the driven model was close to the optimal performance obtained when using the spontaneous activity of chaotic networks (Fig. 8(d)). The lower performance of the spontaneous activity of the network with a gain of 3 was probably caused by the saturation of the reservoir's neurons (Fig. S4). This performance also appears to be dependent on the strength of the reservoir's connections for the driven model, where stronger connections (gain = 1.5) lead to better performances than lower gain values. There was a trade-off between the stability of the trajectory, the saturation and the correlation of the network's units (Fig. S5).

In summary, we showed that a weak periodic drive both helps a network to remain robust to structural perturbations and increase its performance compared to previous implementations.

4 Discussion

4.1 Summary of results

Our results show that reservoir models that were trained to produce precise patterns of activity were highly vulnerable to small structural perturbations. With every architecture that we tested, clamping a single neuron, or setting the weights of a very small subset of synapses to zero, had either a strong impact or completely disrupted the networks' performance on the selected task. This is at odds with the apparent resilience of the brain to minor functional and structural perturbations. For instance, the cerebellum shows considerable resistance to lesions on tasks related to interval timing (Perrett et al 1993). The impact of these lesions on the latency of behavioral responses can be absent.

In contrast, when the same amount of perturbation was distributed across all connections, the performance degradation was milder. We found little or no improvement in resistance to perturbations when increasing the dimensionality of the reservoirs. Thus, in the range of reservoir sizes that we tested, the absolute number of disrupted neurons, and not the proportion of neurons clamped, explained the impact on performance. Changing the distribution of connections in the reservoir had little benefit in terms of resistance, except that a small gain was obtained with an exponential and a power-law distribution of afferent connections. We found a marked improvement in resistance with modular networks, but even in this scenario networks were still vulnerable to local structural perturbations, and a very sparse connectivity between modules was required to benefit from this type of connectivity.

Finally, we introduced an alternative model, where the reservoir was driven by a layer of oscillators. This architecture could learn longer timing intervals, was simpler to train and was more resilient to local perturbations than other implementations. It is known that reservoir networks driven with inputs are more stable than non-driven network(Sussillo and Abbott 2009), and in this work, we showed that driving reservoirs also generate structural robustness. In models that are not driven by external inputs, neurons of the reservoir are only influenced by each other's activity. Conversely, in models that are driven by external inputs, neurons are also influenced by a stable source of external activation that is preserved. This external signal will therefore help to counteract the error created by the perturbation in the network. In other words, the external drive helps to guide the network to driven trajectories, which increase robustness to changes in connectivity.

The extreme sensitivity of the reservoir models that we tested might not be an issue in many tasks where a readout unit is used to discriminate between a few states of the reservoir. This is usually the case for classification tasks where as long as the resulting trajectory remains in the same subspace (e.g. basin of attraction of a fixed point attractor), the readout unit will retain its ability to generate the appropriate output. When tested on a 3-Bit Flip-Flop task (Sussillo and Barak 2013), the networks retained a large part of their ability to perform the task (Fig. S6), compared to when the same architecture was trained to reproduce a complex sinusoid. Issues arise when the readout unit is trained to produce complex output functions based on precise temporal activity in the reservoir. As shown in our results, very small perturbations to the structure of a network cause a very large impact on the network's dynamics, leading to very different trajectories.

This has an important impact on computations since the main source of disruption in the network's performance comes from the deviation of the reservoir's trajectory. This is highlighted by the fact that the networks can sustain a relatively large loss of synapses connecting the reservoir to the readout unit before being significantly impaired. Therefore, it is unlikely that the main factor explaining the impaired performance is the loss of dimensions to the solution found by the readout.

One mechanism that hasn't been extensively tested in the current work is the ability of a network to modify its synapses to compensate for perturbations. The training algorithms that we used are applied to every synapses of either the reservoir or the readout. Therefore, retraining the network after such perturbations generates a completely new solution equivalent to training a new network. A localized re-training (e.g., only a subset of neurons/synapses can be modified) could be used to compensate for synaptic failure in order to retain the configuration of the previous network. When only a subset of connections of the readout units was retrained, the performance was recovered when the size of this subset reached about 10% of all synapses (randomly selected) (Fig. S7). However, in the case of multiple readouts connecting to the reservoir, the total number of synapses to retrain would increase rapidly. Therefore, retraining the reservoir would appear to be more efficient, but this option is only available to architectures that allow reservoir plasticity. However, in the case of innate training, it was almost impossible to retrain the reservoir after a perturbation, possibly because it created different innate trajectories.

The division of reservoirs into compartmentalized pools of recurrent neurons appears to be an effective mechanism to improve their resistance to structural perturbations. Without connections between the modules, the error caused by a structural perturbation is dependent on the number of modules directly affected by the perturbation. Our results showed that modular networks were more resistant to structural perturbations than comparable random networks, due to their ability to restrain the perturbations in a cluster of neurons of the network. However, even if this configuration is beneficial for the network's resilience, this is likely not sufficient, due to the high number of modules required and the low tolerance to inter-module connections. The low number of inter-module connections required to increase to resilience isn't biologically plausible (Meunier et al 2010).

4.2 Biological relevance of the driven model

The driven model presented here relies on the injection of oscillations into neurons of the reservoir. Neuronal oscillations constitute an ubiquitous form of activity in various brain circuits, and transient increase in specific frequency bands is linked to many behavioral and cognitive outcomes (Churchland et al 2012; Kahana 2006; MacKay 2005). The dynamics of the oscillators used in our model are analogous to central pattern generators (CPGs) in the brain (Hooper 2001). These circuits have been shown to produce stable and repetitive activity in the absence of sustained sensory or central input. CPGs have mostly been used to explain stereotyped and repetitive motions and behaviors, such as breathing, walking or scratching (Hooper 2001). Although they are usually associated with lower level brain areas such as in the brainstem and spinal cord, the cerebral cortex could also implement similar dynamics (Yuste et al 2005). Additionally, the phase control of ongoing oscillations has been extensively studied, where reliable and systematic mechanisms for phase resetting have been shown in vitro with single cells (Farries and Wilson 2012), in populations of neurons (Akam et al 2012) and in model cells (Ermentrout 1996).

Therefore, based on the extensive reports of CPGs and phase-resetting mechanisms in the brain, we suggest that the neural mechanisms to start, sustain and end the dynamics of the oscillators proposed in our model are biologically plausible. However, we make no claims about the nature or the implementation of such mechanisms and leave this part for further studies.

4.3 Related models

The use of oscillators with different periods as a source of neural activity underlying interval timing has been

proposed in the past. The model proposed in (Miall 1989) relies on similar principles, where the maximum interval that can be encoded by a neural network is given by the least common multiple of multi-periodic oscillators. In this model, an output unit is trained to detect the coincident activation of some pacemaker cells. These mechanisms have been adapted in the striatal beat-frequency theory (Buhusi and Meck 2005; Lustig et al 2003), where the oscillators have origins distributed in the cortex, and where the basal ganglia act as a coincidence detector. However, these models are specific to interval timing tasks, and aren't applicable to other tasks that require a temporal component in their activity (Mauk and Buonomano 2004). On the other hand, (Fiete et al 2008) proposed a coding system based on similar principles for spatial navigation. In this theory, the different lengths of the lattices created by the grid cells are read by the place cells of the hippocampus based on a modular code, similar to the residue number system (Soderstrand et al 1986). The model that we propose is exploiting both the spatial and the temporal component of the modular code, which makes it more general and powerful.

Most reservoir models use supervised learning (Barak et al 2013; Laje and Buonomano 2013; Maass et al 2002; Sussillo and Abbott 2009), which has limited biological plausibility because it implies that the target function is accessible to the system. While we employed this learning algorithm in our model, some unsupervised learning rules have been successfully applied to similar models (Hoerzer et al 2014) and could potentially be used with our driven model.

4.4 Conclusion

In this work, we raised an important problem with the production of reliable patterns of activity by recurrent neural networks of firing rate units, which had already been highlighted in the past in spiking networks (Banerjee et al 2008; London et al 2010). Our study showed that chaotic neural networks that are resistant to noisy inputs and state-space perturbations (Laje and Buonomano 2013) are no longer resilient under minimal structural perturbations. A priori, chaos and structural perturbations are not necessarily related, and our work showed that it is indeed an important distinction to make with neural circuits.

The spontaneous activity of autonomous chaotic networks is a candidate mechanism to explain the brain dynamics required to execute intrinsically generated temporal processes. However, our work showed that this approach is too sensitive to structural perturbations. On the other hand, we showed that driving reservoirs with oscillating inputs is an attractive solution to generate activity that is resilient to structural perturbations. Future directions would involve the use of reservoirs made of spiking units to investigate their resilience to structural perturbations, as well as the possibility to generate useful spike patterns with oscillating inputs. Experimental studies could also be conducted to unravel the presence of trial-to-trial phase coherence of oscillatory activity related to the generation of spatiotemporal patterns.

Acknowledgements This work was supported by a Discovery grant from the Natural Sciences and Engineering Council of Canada (NSERC Grant No. 210977 and No. 210989), operating funds from the Canadian Institutes of Health Research (CIHR Grant No. 6105509), and the University of Ottawa Brain and Mind Institute (uOBMI), scholarships awarded to PVL from the Ontario Graduate Scholarship (OGS) and the Fonds de recherche Nature et technologies (FQRNT) as well as a FQRNT postdoctoral fellowship, a Bernstein Fellowship and an Innovation Fellowship from the Washington Research Foundation to GL. We would like to thank Eric S. Kuebler and Nareg Berberian for their helpful comments.

5 Compliance with Ethical Standards

The authors declare that they have no conflict of interest. No research involving human participants or animals was performed.

References

Akam T, Oren I, Mantoan L, Ferenczi E, Kullmann DM (2012) Oscillatory dynamics in the hippocampus support dentate gyrus-CA3 coupling. Nature neuroscience 15(5):763–768

Arieli A, Sterkin A, Grinvald A, Aertsen A (1996) Dynamics of ongoing activity: explanation of the large variability in evoked cortical responses. Science 273(5283):1868–1871

Bai Z (1997) Circular law. The Annals of Probability 25(1):494–529

Banerjee A, Seriès P, Pouget A (2008) Dynamical constraints on using precise spike timing to compute in recurrent cortical networks. Neural computation 20(4):974–993

Barak O, Sussillo D, Romo R, Tsodyks M, Abbott LF (2013) From fixed points to chaos: Three models of delayed discrimination. Progress in Neurobiology 103, DOI 10.1016/j.pneurobio.2013.02.002

Bernacchia A, Seo H, Lee D, Wang XJ (2011) A reservoir of time constants for memory traces in cortical neurons. Nat Neurosci 14(3):366–372, DOI 10.1038/nn.2752

Bertschinger N, Natschläger T (2004) Real-time computation at the edge of chaos in recurrent neural networks. Neural computation 16(7):1413–1436

Bonifazi P, Goldin M, Picardo MA, Jorquera I, Cattani A, Bianconi G, Represa A, Ben-Ari Y, Cossart R (2009) GABAergic hub neurons orchestrate synchrony in developing hippocampal networks. Science 326(5958):1419–1424

Brin M, Stuck G (2002) Introduction to dynamical systems. Cambridge University Press

Buhusi CV, Meck WH (2005) What makes us tick? Functional and neural mechanisms of interval timing. Nat Rev Neurosci 6(10):755–765, DOI 10.1038/nrn1764

Buonomano DV, Maass W (2009) State-dependent computations: spatiotemporal processing in cortical networks. Nature Reviews Neuroscience 10(2):113–125

Buxhoeveden DP, Casanova MF (2002) The minicolumn hypothesis in neuroscience. Brain 125(5):935–951

Churchland MM, Shenoy KV (2007) Temporal complexity and heterogeneity of single-neuron activity in premotor and motor cortex. Journal of neurophysiology 97(6):4235–4257

Churchland MM, Cunningham JP, Kaufman MT, Foster JD, Nuyujukian P, Ryu SI, Shenoy KV (2012) Neural population dynamics during reaching. Nature 487(7405):51–56, DOI 10.1038/nature11129

De Zeeuw CI, Hoebeek FE, Bosman LW, Schonewille M, Witter L, Koekkoek SK (2011) Spatiotemporal firing patterns in the cerebellum. Nature Reviews Neuroscience 12(6):327–344

Enel P (2014) Dynamic representation in the prefrontal cortex : insights from comparing reservoir computing and primate neurophysiology. Theses, Université Claude Bernard - Lyon I

Ermentrout B (1996) Type I membranes, phase resetting curves, and synchrony. Neural computation 8(5):979–1001

Farries MA, Wilson CJ (2012) Phase response curves of subthalamic neurons measured with synaptic input and current injection. Journal of Neurophysiology 108(7):1822–1837, DOI 10.1152/jn.00053.2012

Fiete IR, Burak Y, Brookings T (2008) What grid cells convey about rat location. The Journal of Neuroscience 28(27):6858–6871

Froemke RC, Schreiner CE (2015) Synaptic plasticity as a cortical coding scheme. Current opinion in neurobiology 35:185–199

Fu Y, Yu Y, Paxinos G, Watson C, Rusznák Z (2015) Aging-dependent changes in the cellular composition of the mouse brain and spinal cord. Neuroscience 290(0):406–420, DOI 10.1016/j.neuroscience.2015.01.039

Goel A, Buonomano DV (2014) Timing as an intrinsic property of neural networks: evidence from in vivo and in vitro experiments. Philosophical transactions of the Royal Society B: Biological sciences 369(1637):20120,460

Häusler S, Markram H, Maass W (2003) Perspectives of the high-dimensional dynamics of neural microcircuits from the point of view of low-dimensional readouts. Complexity 8(4):39–50

Haykin S (2002) Adaptive filter theory. Prentice Hall 2:478–481

Hoerzer GM, Legenstein R, Maass W (2014) Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. Cerebral cortex 24(3):677–690

Hooper SL (2001) Central Pattern Generators. In: eLS, John Wiley & Sons, Ltd

Jaeger H (2002) Adaptive nonlinear system identification with echo state networks. pp 593–600

Jarvis S, Rotter S, Egert U (2010) Extending stability through hierarchical clusters in echo state networks. Frontiers in neuroinformatics 4

Joshi P, Maass W (2005) Movement generation with circuits of spiking neurons. Neural Computation 17(8):1715–1738

Kahana MJ (2006) The cognitive correlates of human brain oscillations. The Journal of Neuroscience 26(6):1669–1672

Keller CJ, Honey CJ, Mégevand P, Entz L, Ulbert I, Mehta AD (2014) Mapping human brain networks with cortico-cortical evoked potentials. Philosophical Transactions of the Royal Society B: Biological Sciences 369(1653):20130,528

Laje R, Buonomano DV (2013) Robust timing and motor patterns by taming chaos in recurrent neural networks. Nat Neurosci 16, DOI 10.1038/nn.3405

Lajoie G, Lin KK, Shea-Brown E (2013) Chaos and reliability in balanced spiking networks with temporal drive. Physical Review E 87(5):052,901

Lajoie G, Thivierge JP, Shea-Brown E (2014) Structured chaos shapes spike-response noise entropy in balanced neural networks. Frontiers in Computational Neuroscience 8, DOI 10.3389/fncom.2014.00123

Li X, Zhong L, Xue F, Zhang A (2015) A Priori Data-Driven Multi-Clustered Reservoir Generation Algorithm for Echo State Network. PloS one 10(4)

London M, Roth A, Beeren L, Häusser M, Latham PE (2010) Sensitivity to perturbations in vivo implies high noise and suggests rate coding in cortex. Nature 466(7302):123–127

Lustig C, Matell MS, Meck WH (2003) Not "just" a coincidence: Frontal-striatal interactions in working memory and interval timing. Memory 13(3-4):441–448

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

MacKay WA (2005) Wheels of motion: oscillatory potentials in the motor cortex. Motor Cortex in Voluntary Movements: A Distributed System for Distributed Functions pp 181–211

Mainen ZF, Sejnowski TJ (1995) Reliability of spike timing in neocortical neurons. Science 268(5216):1503–1506

Mante V, Sussillo D, Shenoy KV, Newsome WT (2013) Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature 503(7474):78–84

Mauk MD, Buonomano DV (2004) The neural basis of temporal processing. Annu Rev Neurosci 27:307–340

Mazurek ME, Roitman JD, Ditterich J, Shadlen MN (2003) A role for neural integrators in perceptual decision making. Cerebral cortex 13(11):1257–1269

Meunier D, Lambiotte R, Bullmore ET (2010) Modular and hierarchically modular organization of brain networks. Frontiers in neuroscience 4

Miall C (1989) The storage of time intervals using oscillating neurons. Neural Computation 1(3):359–371

Molgedey L, Schuchhardt J, Schuster HG (1992) Suppressing chaos in neural networks by noise. Physical Review Letters 69(26):3717–3719

Mountcastle VB (1997) The columnar organization of the neocortex. Brain 120(4):701–722

Pakkenberg B, Pelvig D, Marner L, Bundgaard MJ, Gundersen HJG, Nyengaard JR, Regeur L (2003) Aging and the human neocortex. Proceedings of the 6th International Symposium on the Neurobiology and Neuroendocrinology of Aging 38(1–2):95–99, DOI 10.1016/S0531-5565(02)00151-1

Perin R, Berger TK, Markram H (2011) A synaptic organizing principle for cortical neuronal groups. Proceedings of the National Academy of Sciences 108(13):5419–5424

Perrett SP, Ruiz BP, Mauk MD (1993) Cerebellar cortex lesions disrupt learning-dependent timing of conditioned eyelid responses. The Journal of Neuroscience 13(4):1708–

1718
Rajan K, Abbott L, Sompolinsky H (2010) Stimulus-dependent suppression of chaos in recurrent neural networks. Physical Review E 82(1):011,903
Romo R, Brody CD, Hernandez A, Lemus L (1999) Neuronal correlates of parametric working memory in the prefrontal cortex. Nature 399(6735):470–473, DOI 10.1038/20939
Rubinov M, Sporns O (2010) Complex network measures of brain connectivity: uses and interpretations. Neuroimage 52(3):1059–1069
Rubinov M, Sporns O, Thivierge JP, Breakspear M (2011) Neurobiologically realistic determinants of self-organized criticality in networks of spiking neurons. PLoS computational biology 7(6):e1002,038
Soderstrand MA, Jenkins WK, Jullien GA, Taylor FJ (eds) (1986) Residue Number System Arithmetic: Modern Applications in Digital Signal Processing. IEEE Press, Piscataway, NJ, USA
Sompolinsky H, Crisanti A, Sommers H (1988) Chaos in random neural networks. Physical Review Letters 61(3):259
Sussillo D, Abbott LF (2009) Generating Coherent Patterns of Activity from Chaotic Neural Networks. Neuron 63, 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
Sussillo D, Churchland MM, Kaufman MT, Shenoy KV (2015) A neural network that finds a naturalistic solution for the production of muscle activity. Nature neuroscience 18(7):1025–1033
Tiesinga P, Fellous JM, Sejnowski TJ (2008) Regulation of spike timing in visual cortical circuits. Nature reviews neuroscience 9(2):97–107
Toledo-Suárez C, Duarte R, Morrison A (2014) Liquid computing on and off the edge of chaos with a striatal microcircuit. Frontiers in computational neuroscience 8
van Vreeswijk C, Sompolinsky H (1996) Chaos in neuronal networks with balanced excitatory and inhibitory activity. Science 274(5293):1724–1726
Westlye LT, Walhovd KB, Dale AM, Bjørnerud A, Due-Tønnessen P, Engvig A, Grydeland H, Tamnes CK, Østby Y, Fjell AM (2009) Life-span changes of the human brain white matter: diffusion tensor imaging (DTI) and volumetry. Cerebral cortex p bhp280
Yamazaki T, Tanaka S (2007) The cerebellum as a liquid state machine. Neural Networks 20(3):290–297
Yuste R, MacLean JN, Smith J, Lansner A (2005) The cortex as a central pattern generator. Nature Reviews Neuroscience 6(6):477–483