nature neuroscience

Robust timing and motor patterns by taming chaos in recurrent neural networks

Rodrigo Laje[1,5] & Dean V Buonomano[1–4]

The brain's ability to tell time and produce complex spatiotemporal motor patterns is critical for anticipating the next ring of a telephone or playing a musical instrument. One class of models proposes that these abilities emerge from dynamically changing patterns of neural activity generated in recurrent neural networks. However, the relevant dynamic regimes of recurrent networks are highly sensitive to noise; that is, chaotic. We developed a firing rate model that tells time on the order of seconds and generates complex spatiotemporal patterns in the presence of high levels of noise. This is achieved through the tuning of the recurrent connections. The network operates in a dynamic regime that exhibits coexisting chaotic and locally stable trajectories. These stable patterns function as 'dynamic attractors' and provide a feature that is characteristic of biological systems: the ability to 'return' to the pattern being generated in the face of perturbations.

Timing is a fundamental component of sensory and motor processing, learning, and cognition; however, the neural mechanisms underlying temporal processing remain unknown[1–3]. On the scale of milliseconds and seconds, a number of different mechanisms have been proposed to underlie sensory and motor forms of timing, including internal clocks that rely on a pacemaker and counter[4], ramping firing rates[5,6], multiple oscillator models that rely on detecting the beats between oscillators running with different periods[7,8], and the stochasticity of neural dynamics[9]. Although these models are not necessarily mutually exclusive, many of them focus primarily on simple temporal tasks. For example, internal clock and ramping models are generally proposed as mechanisms underlying the timing of single intervals and are unlikely to contribute to complex temporal or spatiotemporal motor processing such as tapping Morse code or generating cursive handwriting. We focused on a more general framework that could account for a wide range of temporal and spatiotemporal tasks in the range of tens of milliseconds to a few seconds. Specifically, the idea that motor timing relies on the dynamic changes in the pattern of activity of neurons in recurrent neural networks[1,10,11].

The first models to propose that time might be encoded in the dynamic changes in the patterns of active neurons were developed in the context of the cerebellum[11,12]. Subsequent models emphasized the importance of dynamic patterns of activity in a population of neurons for neural computations in general[13–16]. In this framework, the state of a network at any given time can be represented by a point in a high-dimensional space where each dimension corresponds to the activity level of a neuron. The concatenation of these points over time forms a 'neural trajectory'. In contrast with conventional attractor models, temporal and spatiotemporal computations in this 'population clock' framework arise from the voyage through state space, as opposed to the arrival at any one given location. The advantage of computing with neural trajectories is particularly obvious for tasks that require timing, as time is implicitly encoded in the trajectory and can be read out by downstream neurons. This framework is quite general because it can account for both temporal and spatiotemporal processing, that is, the generation of complex motor patterns. Furthermore, experimental studies in different brain areas have identified time-varying populations of active neurons that encode time[17–20].

At a theoretical level, the hypothesis that neural networks can autonomously generate continuously changing patterns of activity in a flexible and robust manner has been controversial. The main challenge has been that recurrent neural networks operating in 'high-gain' regimes in which recurrent connections are strong enough to generate self-sustained patterns of activity are highly sensitive to noise and are often formally chaotic[21–27]. Thus, although the dynamics in these networks is potentially computationally powerful, the fact that minute levels of noise can produce vastly different neural trajectories effectively abolishes their computational power because a given pattern cannot be reliably reproduced across trials.

Building on two previous firing rate models[28,29], we developed a recurrent network model that produces complex, high-dimensional trajectories that are highly resistant to noise. This robustness was achieved by tuning the recurrent connections of the network. A powerful computational consequence of this approach is that a previously chaotic trajectory becomes a locally stable channel or 'dynamic attractor' (meaning that even if the network is perturbed it can return to its trained trajectory). We found that these stable neural trajectories can markedly improve the ability of random recurrent networks (RRNs) to tell time and generate complex motor patterns in the presence of high levels of noise. Because our model is based on firing-rate units, the problem of chaotic behavior in spiking neural networks remains unsolved. However, we found that it is possible to tame chaos in firing-rate

[1]Department of Neurobiology, University of California, Los Angeles, California, USA. [2]Department of Psychology, University of California, Los Angeles, California, USA. [3]Brain Research Institute, University of California, Los Angeles, California, USA. [4]Integrative Center for Learning and Memory, University of California, Los Angeles, California, USA. [5]Present addresses: Departamento de Ciencia y Tecnología, Universidad Nacional de Quilmes, Bernal, Argentina, and Consejo Nacional de Investigaciones Científicas y Técnicas, Buenos Aires, Argentina. Correspondence should be addressed to D.V.B. (dbuono@ucla.edu).

Received 23 December 2012; accepted 20 April 2013; published online 26 May 2013; doi:10.1038/nn.3405

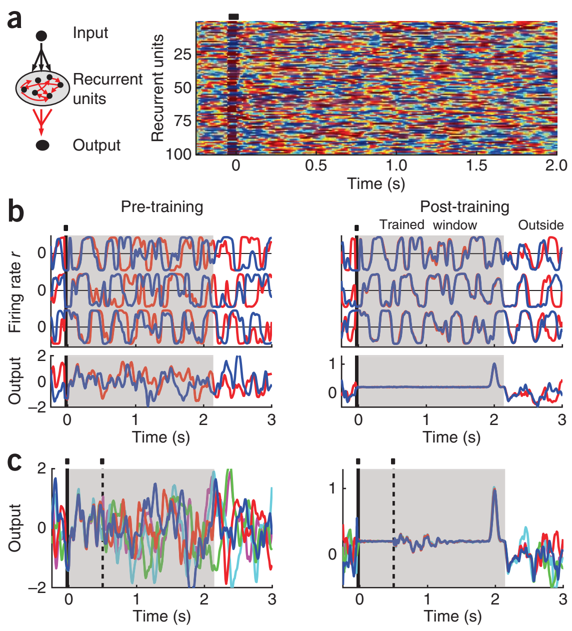

Figure 1 Complexity without chaos. (**a**) A random recurrent network (left) in the chaotic regime is stimulated by a brief input pulse (small black rectangle at $t = 0$, right) to produce a complex pattern of activity in the absence of noise. Right, color-coded raster plot of the activity of 100 of 800 recurrent units. Color-coded activity ranges from −1 (blue) to 1 (red). (**b**) Time series of three sample recurrent units (top) and the output unit (bottom). In the pre-training (left), the blue traces comprised the innate trajectory subsequently used for training. The divergence of the blue and red lines demonstrates that two different initial conditions (before the input) lead to diverging trajectories before training, even in the absence of ongoing noise. After training (right), however, the time series are reproducible during the trained window (2.25 s, shaded area). That is, despite different initial conditions, the blue and red lines trace very similar paths while still diverging outside of the trained window. The output unit was trained to pulse after 2 s. (**c**) Five different runs of the network above, perturbed with a 10-ms pulse at $t = 0.5$ s (dashed line) from an additional input unit randomly connected to the recurrent network. The trained network (right) robustly reproduces the trained trajectory, recovering from the perturbation resulting in the timed response of the output unit at $t = 2$ s.

recurrent networks and that the resulting dynamics offers a new neurocomputational framework based on dynamic attractors.

RESULTS

Innate training

The network model that we studied consists of randomly connected nonlinear firing-rate units[26,28,29]. In this network, the connectivity is represented by a recurrent weight matrix, $\mathbf{W}^{\text{Rec}}$, drawn from a normal distribution with a mean of zero and a s.d. scaled by a gain parameter, g. For large networks, values of $g > 1$ generate increasingly complex and chaotic patterns of self-sustained activity[26]. In all of our simulations, the networks are in this high-gain chaotic regime ($g \geq 1.5$)[26,30]. **Figure 1a** provides an example of such an RRN (see Online Methods). By adjusting the synaptic weights onto an output unit, we could train the network to produce some desirable computation, such as a timed response or a complex motor output[10–12,28] (see below). The network is spontaneously active (that is, it has self-sustaining activity) and an external input at $t = 0$ ms (50-ms duration) temporarily kicks the network into a delimited volume of state space, which can be defined as the starting point of a neural trajectory. Across trials, even in the absence of continuous noise, different initial conditions resulted in a divergence of the trajectories (**Fig. 1b**). This divergence renders the network useless from a computational perspective because the patterns cannot be reproduced across trials. One approach to overcome this problem has been to use tuned feedback to control the dynamics of the network[28,29]. An alternate approach would be to alter the weights of the RRN proper to decrease the sensitivity to noise; this approach, however, has been limited by the challenges inherent in changing the weights in recurrent networks. Specifically, given that all weights are being used throughout the trajectory, plasticity tends to markedly alter network dynamics, produce bifurcations and not converge[31].

It is important to note that, in the current ‘reservoir’ framework, the precise pattern produced by the recurrent network is largely irrelevant; what matters is that it is complex and that these patterns can be used by downstream units[13,16,32]. This means that, independent of the ultimate desired output, there is really no specific desired target activity pattern in the recurrent network. Thus, we reasoned that noise sensitivity could be reduced by training the units in the network to reproduce their ‘innate’ pattern of activity, rather than some trajectory determined by the desired output. We define an innate trajectory as one triggered by a given input in an untrained network (using an arbitrary initial condition); we chose the innate trajectories in the absence of noise, but they can also be chosen in the presence of noise (see Online Methods). The approach is to tune the recurrent units to do what they can already do. Toward this end, we trained recurrent units to reproduce their innate activity profile using a supervised learning rule to rapidly minimize the errors during a training trial (see Online Methods)[28,29]. By training the RRN to reproduce its innate trajectory over a 2.25-s period, it was possible to create a locally stable transient channel (**Fig. 1b**), largely preserving the shape of the original trajectory while turning it into an attracting one in the 2.25-s window. Outside the training window, however, the trajectory rapidly diverges. Once the RRN generates stable trajectories, the output can be trained to produce a timed response at 2 s (**Fig. 1b**). This timed response is now robust to differences in initial conditions, noise and large perturbations in the recurrent network (see below for a more detailed analysis). **Figure 1c** shows an example in which the pretraining and trained RRN are perturbed with a 10-ms pulse from a second input unit. Despite this perturbation, the trained network can recover and return to the innate trajectory and generate a timed response at approximately 2 s.

In the above example, the timing that generated the late response is encoded in the neural dynamics of the network. This same high-dimensional dynamics can be used to generate arbitrary spatiotemporal patterns that are highly resistant to noise and perturbations. To illustrate this, we first trained the RRN to robustly reproduce two different innate activity patterns in the same manner described above and then trained two output units to generate handwritten words. Two distinct brief inputs (50-ms duration) were used to stimulate an RRN in the absence of noise to generate the two innate trajectories for training the RRN. After training the RRN on both trajectories, two output units (representing x and y axes) were trained and then tested (in the presence of continuous noise) to produce the words “chaos” and “neuron” in response to inputs 1 and 2, respectively (**Fig. 2a** and **Supplementary Matlab Routines**). One notable feature of creating locally stable trajectories is that they function as dynamic attractors: even relatively large perturbations to the RRN can be self-corrected. This feature can be seen by perturbing the network activity after the trajectory has already been initiated (**Fig. 2b**). We perturbed the network using a 10-ms pulse of an additional input

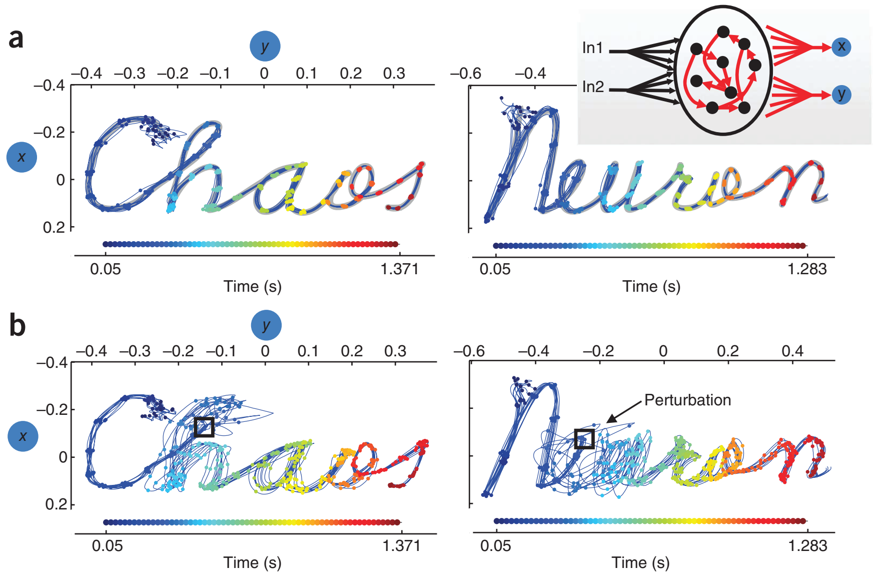

Figure 2 Generation and stability of complex spatiotemporal motor patterns. (**a**) Blue traces represent ten test trials in response to input 1 (In1, left) or input 2 (In2, right) after training; the background gray line shows the output target. These test trials were run over different initial conditions in the presence of continuous noise (0.001) in all of the 800 recurrent units. Time is represented by uniformly placed colored circles ($\Delta t \cong 18$ ms). (**b**) Test trials run under the same initial condition in the presence of continuous noise, but with the addition of a perturbation at 300 ms (open square). The perturbation was produced by an additional 10-ms input pulse (not diagrammed) with an amplitude of 0.2.

unit randomly connected to all units in the RRN with an input amplitude of 0.2, injected at $t = 300$ ms (approximately the time of the “h” and “e” during the “chaos” and “neuron”, respectively). Despite the obvious effect of the perturbation on the state of the recurrent network (as evidenced by the altered output), the network returned to the original trajectory over the course of a few hundred milliseconds, resulting in increasingly clear writing.

Computational power of innate training

To characterize the computational power of the innate training, we quantified the timing capacity of the network by determining the maximal delay after the input that the network could produce (**Fig. 3**). The target output function was flat (nonzero) with a simple pulsed response at different delays after the 50-ms input. A network of 800 neurons ($g = 1.5$) reliably learned a 5,000-ms delay (note that estimates of timing capacity must be interpreted in the context of the time constant of the units, 10 ms), but not a 6,000-ms delay, reflecting the finite memory of such networks[28,33] (**Fig. 3a** and **Supplementary Matlab Routines**). To quantify this, we parametrically varied the delay and compared the performance of the innate training approach to two additional architectures (**Fig. 3b**) using the same set of ten initial networks for all architectures. Together, the three architectures were the current approach (innate training), in which recurrent plasticity in the RRN was directed at the innate trajectory, an echo-state/FORCE approach (echo state), in which the output feeds back onto the RRN and only the connections from the recurrent to output units were plastic[28,29], and an RRN with recurrent plasticity (fair recurrent plasticity), which provided a control for the amount of plastic connections involved in the training; thus, as in the innate training architecture, the weights of 60% of recurrent units were adjusted according to the error in the output unit[29]. Both training and testing in this task occurred with random initial conditions and in the presence of continuous noise (noise s.d., $I_0 = 0.001$). The innate training of the recurrent connections markedly improved the maximal time delay of the network (defined as the time delay at which performance decays to 0.5), producing, on average, a fivefold improvement (**Fig. 3b**).

All of the networks were trained for 30 training trials of the RRN (**Fig. 3**). To examine the effects of the number of training trials on performance, we also carried out the same analysis over 10 and 20 training trials. We found that there was a trade-off between the duration of the training window, the number of training loops, and performance; shorter windows required fewer training trials to achieve maximal performance (**Supplementary Fig. 1**).

The observed timing capacity of approximately 5 s (for a network of 800 neurons) raises the question of what determines this limit. There are a number of factors contributing to this capacity, including the intrinsic richness of the RRN patterns (related to g), noise levels and ability of the output unit to readout these patterns. However, it

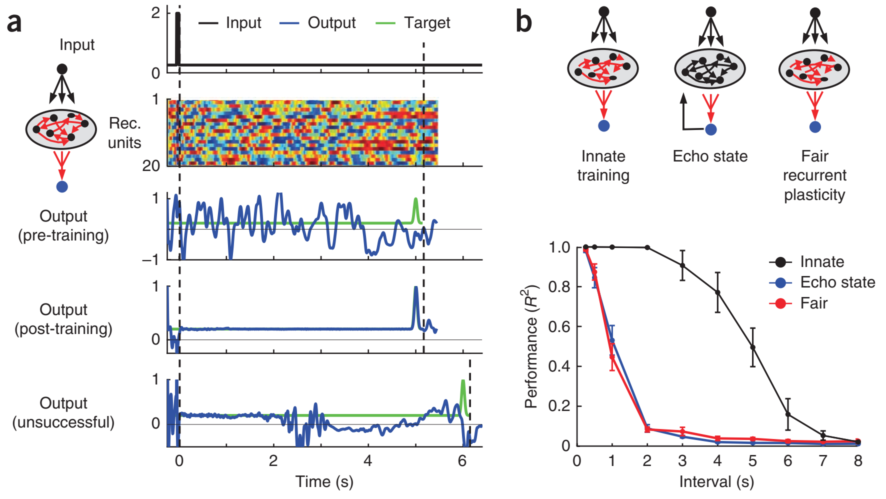

Figure 3 Improved timing capacity. (**a**) An input pulse (black trace) triggers a chaotic innate neural trajectory, displayed as a color-coded raster plot (only 20 of 800 units shown). The linear readout unit receives input from all the recurrent units (blue trace), showing irregular pre-training activity. After the RRN is trained to the innate trajectory (training window defined by dashed lines), the readout unit is trained to reproduce a flat target with a pulse at a given interval (green trace, 5-s duration in this example). An unsuccessful simulation from a 6-s interval training is also included as an example. (**b**) Performance across different architectures. Ten RRNs were trained in each of the three displayed architectures, parametrically varying the delay. The performance (goodness of reproduction) is quantified by the Pearson correlation coefficient R^2 between target and actual output (green and blue traces in **a**); mean ± s.e.m. across networks.

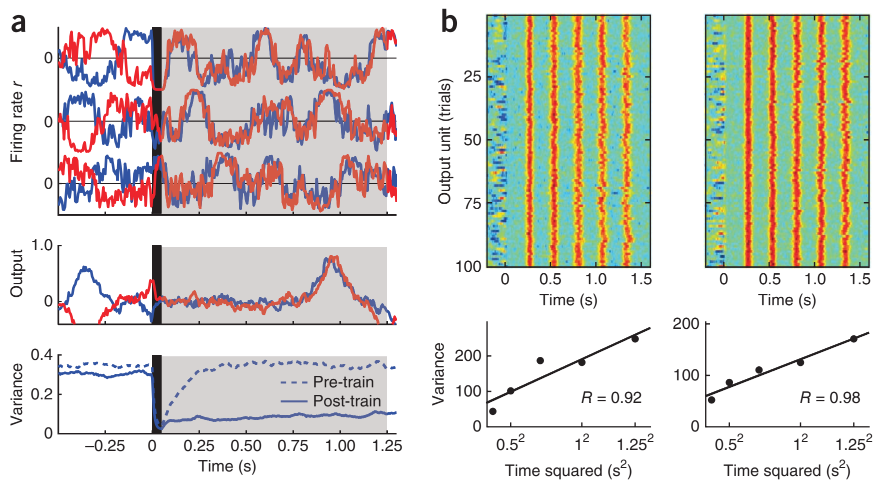

Figure 4 Innate training decreases the neural variance and results in Weber-like timing. (**a**) Top, time traces of three sample units over two different trials (blue and red; $N = 800$, $g = 1.5$, $p_c = 0.25$, 1.3-s training window). Gaussian noise with a s.d. of $I_0 = 1.5$ was continuously injected into all recurrent units. As in **Figures 1** and **3**, the output unit was trained to generate a timed pulse (1,000 ms after the onset of the 50-ms input pulse, middle). Bottom, neural variance. The variance of each unit was calculated over eight trials, and then averaged over all 800 units. There was a sharp decrease in variance produced by the onset of the stimulus, which persisted over many seconds before gradually ramping back up to baseline (data not shown). The dashed line shows the neural variance before training: because the input clamps network activity, stimulus onset also produced a decrease in the variance, but it rapidly increased after stimulus offset. The mean s.d. across units at the input of the input pulse were 0.037 and 0.024, before and after training, respectively. (**b**) Example of two simulations in which the output unit was trained to produce events at 250, 500, 750, 1,000 and 1,250 ms (top). Variance across trials was estimated by calculating the time of the peak of each response. The relationship between variance and t^2 was well fit by a linear function (bottom). $I_0 = 1.0$.

is possible to obtain an empirical upper bound on the 'raw' encoding capacity of the network by performing the same analysis shown in **Figure 3b** in the absence of any noise and with an untrained recurrent network (**Supplementary Fig. 2**). These results reveal an upper bound of approximately 20 s. This upper bound is, of course, essentially useless from a computational perspective because the network is chaotic. But it does provide an empirical ceiling for the temporal encoding capacity of the model.

Cross-trial variability and timing precision

Implicit in the findings described above is that, after training, there are different types of dynamics in the same network: while ongoing activity (or trajectories triggered by untrained inputs) continues to produce chaotic trajectories, the trained trajectories exhibit locally stable patterns of activity. Recent experimental studies have also revealed different types of dynamics in the same network. For example, it has been shown that cross-trial variability of neural activity is quenched in response to stimulus onset[34]; that is, the variability of neural ongoing or background activity is substantially larger than that observed after a stimulus or during a behavioral task. We therefore quantified the cross-trial variance before and after the brief 50-ms input in the trained and untrained networks. In addition, to push the envelope in terms of how much noise the network can handle, we increased the noise levels during training and testing (as well as the number of training trials). The variance was calculated over eight test trials for each of the 800 units over a time period starting 500 ms before the stimulus. The target delay was 1,000 ms (and the training window was 1,300 ms). In the presence of continuous very high levels of noise ($I_0 = 1.5$), each of the recurrent units exhibited substantial jitter, reminiscent of the membrane voltage fluctuations observed *in vivo*, resulting in a high cross-trial variance before stimulation ($t < 0$; **Fig. 4a**). Nevertheless, in response to the input, the trained network was still able to robustly generate an appropriately timed output, and, as expected, this robustness reflected a marked decrease in the variance of the activity after the stimulus onset (**Fig. 4a**).

Psychophysical studies have carefully characterized the precision of timed motor responses (see Discussion). The variance of timed motor responses in the range of up to a few seconds is generally well captured by a linear relationship with t^2, known as the generalized Weber's law. To characterize the variance signature of the model, we trained the output units to generate several consecutive responses at intervals of 250 ms. The relationship between variance of the peak response and t^2 was well fit by a linear function ($R > 0.9$ in each of five networks tested; **Fig. 4b**). These results establish that stable RRNs can account for Weber's law. We stress, however, that, depending on the noise levels and intervals being trained, nonlinear relationships are also observed (see Discussion).

Noise analysis, suppression of chaos and stimulus specificity

We next examined two critical issues relating to the stability and dynamics of the trained recurrent networks. First, we performed a parametric noise analysis to quantitatively characterize the response of the trained networks in the presence of high levels of noise. To this end, we continuously injected different levels of noise into all 800 units of the recurrent network. Second, we examined whether training specifically altered the noise sensitivity of the trajectory elicited by the trained input or whether training produced global changes of all network trajectories. This question can be seen as addressing whether learning (creating locally stable trajectories) is stimulus specific. Each of ten different networks ($N = 800$, $g = 1.8$) was stimulated with two different 50-ms long inputs. The neural trajectory produced by input 1 served as the innate training target (duration of 2 s) for recurrent plasticity, whereas the trajectory triggered by input 2 served as a control to determine the effect of training on untrained trajectories. After training, performance was quantified by examining the correlation in the 2-s window between the trajectories elicited in the presence of noise in relation to the trajectory in the absence of noise (reproducibility; see Online Methods). After training, the activity patterns in the recurrent units were very similar in the absence and in the presence of continuous noise at levels of $I_0 = 0.001$ and 0.1, but not 1.0 (**Fig. 5a**). The average data indicate that in the presence of noise amplitudes of up to 0.1 performance in response to input 1 was essentially perfect (**Fig. 5b**). In these simulations, the RRNs were trained for 20 trials (noise amplitude during training $I_0 = 0.001$). The reproducibility was not substantially better with 30 training trials (**Supplementary Fig. 3**). However, the sensitivity to noise could be even further decreased by training in the presence of more noise for more trials (for example, **Fig. 4** and **Supplementary Figs. 1** and **3**).

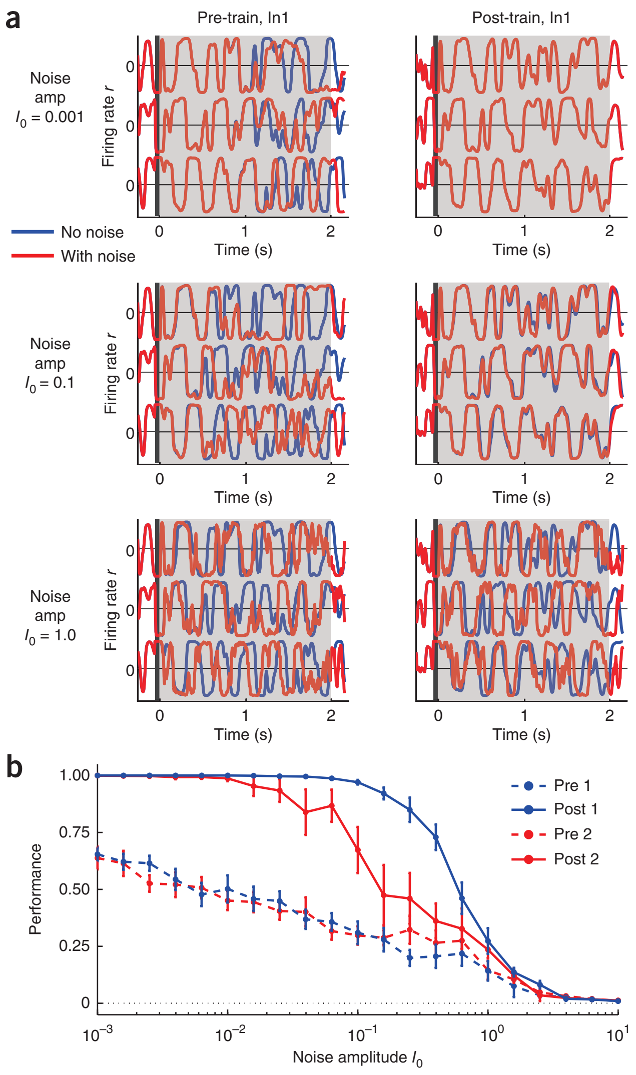

Figure 5 Robustness against noise. (**a**) Activity of three sample units in the recurrent network at three different levels of noise. Blue indicates template trajectory (no noise) and red indicates test trajectory (continuous noise in each unit). The s.d. of the noise current I_0 was 0.001, 0.1 and 1.0 (top to bottom; noise amplitudes as a fraction of total absolute incoming synaptic weights to each unit averaged across units are 0.007%, 0.7% and 7%, respectively). (**b**) Average data from ten different networks. Performance was measured as the averaged Pearson correlation coefficient between template (blue) and test trajectories (red) for each condition (after Fisher transformation), mean ± s.e.m. across networks.

Training to the input 1 trajectory also improved the reproducibility of input 2, but, in addition to the magnitude of the improvement, there was a fundamental difference between the trained and untrained trajectories. The increased reproducibility of both the trained and untrained patterns does not imply that either of them was no longer chaotic, but rather provides an estimate of how much two trajectories overlap in a 2-s window in response to different levels of noise. Thus, to formally characterize the behavior of the networks before and after training, we quantified the divergence of trajectories by estimating the largest Lyapunov exponent (λ), which provides a measure of the rate of separation of two nearby points in state space, a standard way to determine whether a dynamical system is chaotic. For each of the ten networks, λ was numerically estimated for the trajectories elicited by input 1 and input 2, both before and after training (**Fig. 6**) and both inside and outside of the training window. Before training, both trajectories exhibited positive exponents, indicative of exponential divergence and, thus, chaotic dynamics. After training, the mean λ across networks for input 1 was not significantly different from zero ($\lambda = 0.05 \pm 0.45$, $P = 0.90$), suggestive of local stability. The mean λ for input 2 also decreased, but remained above zero (10 of 10 networks). The dynamics in response to both inputs outside the training window (between $t = 8$ s and $t = 10$ s) exhibited chaotic dynamics (8 of 10 networks) or entered stable limit cycles (2 of 10). Which of these regimes occurred was dependent in part on the initial structure of the network and the extent of the training: lower initial values of λ and/or more training loops were more likely to lead to a limit cycle (data not shown). Notably, a 2 × 3 two-way ANOVA with repeated measures (factors input and training) revealed a significant interaction effect ($F_{2,18} = 20.7$, $P = 2 \times 10^{-5}$), meaning that λ post-training was differentially affected by input 1.

These results indicate that the original innate trajectory was transformed into a locally attracting trajectory, best described as a stable transient channel to the chaotic attractor (see **Supplementary Modeling** for a discussion of other relevant chaotic phenomena). Thus, in a local sense, the chaotic behavior of the trained trajectory was 'tamed' by training. In contrast, the untrained trajectories remained chaotic.

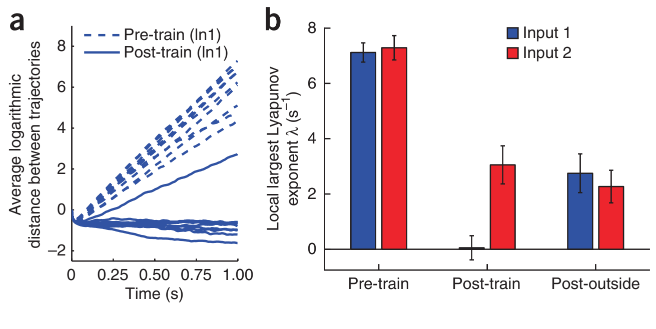

Figure 6 Suppression of chaos. (**a**) Average logarithmic distance between original and perturbed trajectories for each of ten networks for the trajectories triggered by input 1 (the trained input) before and after training. A straight portion with a positive slope indicates chaotic dynamics; the value of the slope is the estimate for the largest Lyapunov exponent (λ). After training, the original and perturbed trajectories no longer diverged (except for one network). (**b**) The pre-training trajectories triggered by both inputs displayed positive λ, indicative of chaotic dynamics (input 1: $\lambda = 7.12 \pm 0.35$, mean ± s.e.m. across the ten networks, values significantly different from zero, t test $P = 10^{-8}$; input 2: $\lambda = 7.29 \pm 0.45$, $P = 4 \times 10^{-8}$; all reported λ values have units of s^{-1}). After training, the trajectory triggered by input 1 was locally stable, as indicated by a near zero mean λ ($\lambda = 0.05 \pm 0.45$, $P = 0.90$); input 2, however, still produced diverging trajectories as evidence by λ significantly above zero ($\lambda = 3.05 \pm 0.70$, $P = 0.0016$). After training, the trajectories outside the trained window had a positive mean λ in response to both inputs (input 1: $\lambda = 2.75 \pm 0.70$, $P = 0.0035$; input 2: $\lambda = 2.27 \pm 0.60$, $P = 0.0039$), with some networks displaying chaotic activity (8 of 10) and some entering limit cycles (2 of 10). The interaction effect was significant ($F_{2,18} = 20.7$, $P = 2 \times 10^{-5}$, a 2 × 3 two-way ANOVA with repeated measures, factors input and training). In addition to this stimulus-specific effect of training, there was a global nonspecific effect of decreased divergence of trajectories after training, represented by a lower, although still positive, λ for post-train input 2 and post-outside inputs 1 and 2.

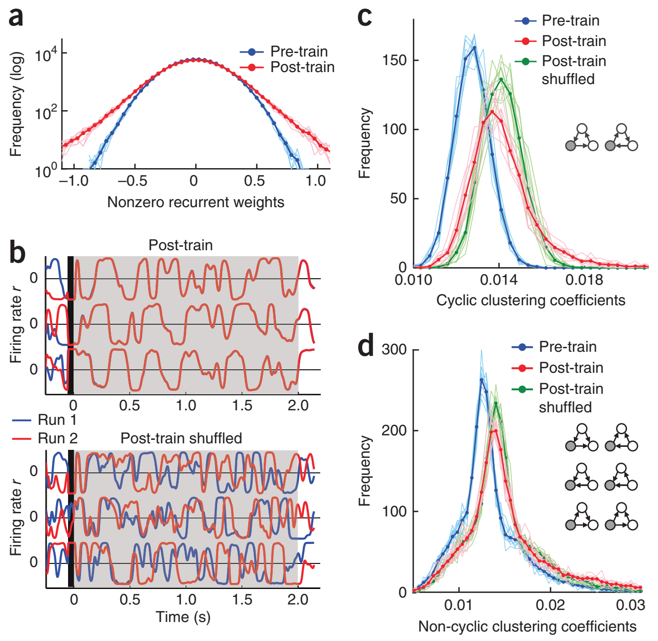

Figure 7 Effects of training on network structure. (**a**) Distribution of the nonzero recurrent weights. Thin lines represent the distributions of the weights of ten networks before (blue) and after (red) training. Thick lines represent the averages across the ten networks. Pre-training: networks are Gaussian by construction. Post-training: all networks are non-Gaussian (Lilliefors test, $P < 0.001$ for each of the ten networks). Median absolute synaptic weights significantly increased after training (pre-train median ± MAD across ten networks, 0.1358 ± 0.0004; post-train, 0.147 ± 0.001; paired Wilcoxon sign-rank test, $P = 0.002$). (**b**) Numerical simulation of one trained network before and after shuffling the weights of its recurrent matrix $\mathbf{W}^{\mathrm{Rec}}$ (two runs each, without noise), showing that the stability properties of the shuffled network are lost despite having the same weight distribution and the same connectivity. (**c**) Distribution of local weighted cyclic clustering coefficients. Training leads to an increase in the cyclic clustering coefficients. Shuffling (green) of the weights of the post-train recurrent matrix $\mathbf{W}^{\mathrm{Rec}}$ significantly changed the cyclic clustering distribution (two-sample Kolmogorov-Smirnov test between post-train and post-train shuffled for every network, $P < 0.002$ for all cyclic distributions). Insets reflect the possible circuit motifs in relation to a reference unit shown in gray. (**d**) Distribution of local weighted non-cyclic clustering coefficients. Training also increased the median non-cyclic clustering coefficients.

In summary, while **Figure 5b** demonstrates an improvement in the reproducibility of the untrained trajectory, **Figure 6** establishes that the untrained trajectory is still chaotic—that is, in response to a perturbation the trajectories will still diverge at an exponential rate. The practical meaning of these results is that, in response to fairly large perturbations, the trained trajectory exhibits the desirable feature of being able to 'find its way back' after being perturbed, whereas, in response to small or large perturbations, the untrained trajectory will continue to diverge off on some new path (albeit at a slower rate than before training), even though it can stay on track for a few seconds in response to tiny perturbations (**Supplementary Fig. 4**).

Mechanisms: network structure after training

To characterize the relationship between the observed behavior and the structure of the trained recurrent networks, we examined the distribution of weights and the connectivity patterns before and after training. The distribution of the nonzero recurrent weights changed very consistently (**Fig. 7a**). Innate training led to a non-Gaussian distribution with long tails (note that the number of nonzero weights does not change because training does not alter which units are connected), meaning that the median absolute synaptic weight increased (pre-train median ± mean absolute deviation from the median (MAD) across 10 networks: 0.1358 ± 0.0004; post-train: 0.147 ± 0.001; paired Wilcoxon sign-rank test, $P = 0.002$). Shuffling the weights (but not the connections) of the recurrent matrix $\mathbf{W}^{\mathrm{Rec}}$ after training left the weight distribution untouched, but the stability properties of the network were destroyed (**Fig. 7b**). Thus, it's not simply the statistics of the synaptic weights or the binary connectivity what defines the network behavior. As an example of the importance of precise wiring rather than the distribution, we found that post-training weights from bidirectional connections were significantly stronger on average than those from unidirectional connections (in absolute value; unidirectional median ± MAD across networks: 0.145 ± 0.001; bidirectional: 0.161 ± 0.003; paired Wilcoxon sign-rank test, $P = 0.002$; **Supplementary Fig. 5**). Notably, both the long-tailed weight distribution and the bidirectional versus unidirectional connectivity features that we observed have been reported in the rat visual cortex[35].

To explore the role of the connectivity structure of the trained networks, we computed the distribution of local clustering coefficients that are associated with recurrency and self-sustained activity (see Online Methods)[36]. The cyclic clustering coefficients provide a measure of the number of neuron triplets connected in a circular fashion, weighted by their synaptic strengths. Innate training increased the median cyclic clustering coefficients (pre-train median ± MAD across networks: 0.01270 ± 0.00005; post-train: 0.0139 ± 0.0001; paired Wilcoxon sign-rank test, $P = 0.002$) and made the distribution of the clustering coefficients have a longer right tail and a non-Gaussian distribution (**Fig. 7c**). Notably, innate training also resulted in an increase in the non-cyclic clustering coefficients (Pre-train median ± MAD across networks: 0.01280 ± 0.00005; Post-train: 0.0142 ± 0.0002; paired Wilcoxon sign-rank test $P = 0.002$; **Fig. 7d**), leading to a stronger short-range feedforward structure.

To determine whether the observed dynamics reflected the specific wiring signature of the trained networks, we calculated both cyclic and non-cyclic clustering distributions after shuffling the weights of the trained networks (**Fig. 7c,d**). Shuffling significantly altered the distribution of the cyclic distribution more than that of the non-cyclic coefficients (two-sample Kolmogorov-Smirnov test between post-train and post-train shuffled for every network, $P < 0.002$ for all cyclic distributions; P values of non-cyclic distributions ranged from 0.002 to 0.11), suggesting that the presence of cyclic clusters may have be important for the ability of an RRN to generate complex, yet stable, neural trajectories. However, as noted above, an untrained input can produce a chaotic trajectory after training; thus, it is clear that some interaction between the input and the structure of the recurrent network is involved in the resulting dynamics.

As an initial attempt to correlate the stable activity pattern with the structure of the network, we examined the correlation between all of the plastic recurrent weights W_{ij} of a network and the correlation in firing rates of the pre- and postsynaptic units r_i and r_j during the trained innate trajectory (**Supplementary Fig. 6**). There was a moderate correlation ($R = 0.355 \pm 0.005$, $P < 10^{-16}$ for each of five networks examined) between the initial weights and the presynaptic-postsynaptic correlations. Contrary to our expectations, there was actually a small decrease in this correlation after training ($R = 0.322 \pm 0.005$; $P < 10^{-16}$ for each of the five networks). Overall, these results indicate that the stability of the trained trajectory is not a simple optimization of the weights on the basis of the mean correlation of presynaptic-postsynaptic activity.

DISCUSSION

Here we describe a robust and general mechanism by which recurrent neural networks could encode time and generate complex spatiotemporal patterns. The model builds on a number of previous studies[10,11,16,28,29], but is unique in the extent to which it behaves as a dynamic attractor—that is, the network can return to and complete a trained pattern even when the entire recurrent network is substantially perturbed. Indeed, in the sense that previously chaotic trajectories are turned into stable ones, it can be said that this approach tames chaos. In addition to the locally stable nonperiodic trajectories, the network exhibited coexisting chaotic trajectories. These features are absent from previous models operating in the high-gain regime, including those that used controlled feedback or that incorporated recurrent plasticity driven by the output error (echo state and fair recurrent plasticity architectures; **Fig. 3**)[29]. Here local stability was achieved by tuning the weights of the recurrent network to reproduce an innate trajectory, effectively teaching the network to robustly reproduce one of the arbitrary trajectories it can already generate. The advantage of training on an innate trajectory is that it guarantees that the network is attempting to learn an attainable trajectory. The outcome of training is that the learned trajectories are locally stable over many seconds, despite the fact that all of the units in the network have a 10-ms time constant.

Implications for the neural mechanisms of timing

A long-standing and ongoing debate on the neural basis of timing relates to where in the brain temporal computations occur and whether timing is a result of centralized (dedicated) or general (intrinsic) mechanisms[3]. Our view is that, precisely because timing is critical to so many forms of processing, it is a general computation performed by recurrent neural networks. For this reason, our model is presented as a general computational framework of recurrent networks that may be engaged in a number of different areas depending on the task at hand. Indeed, this view is supported by a growing experimental literature that suggests that a large number of different brain areas are involved in timing. These areas include, but are not limited to, the cerebellum, basal ganglia, hippocampus, and motor, frontal and parietal cortex[1,2,37–43].

Traditionally, the experimentally observed variance signature of timed responses has been used as an important criterion to evaluate models of timing. In a given task, timing variability is often well described by Weber's law, meaning that there is a constant ratio between the s.d. of the response and the interval being timed[2]. For motor timing on the scale of up to a few seconds, it is established that variability is best accounted for by Weber's generalized law, in which the variance of the response is linearly related to time squared (plus an additional variance term). The timing described here is well captured by the generalized Weber function, but we emphasize that this result is dependent on parameters (**Fig. 4b**). Specifically, variance can become either sub- or supralinear depending on the overall level of noise and the timescale being examined; with very low noise levels, the relationship tends to be sublinear, and, over time spans that exceed the timing capacity of the network, the relationship becomes supralinear. Nevertheless, it is relevant that the model can capture the experimentally observed linear relationship between variance and time squared.

When considering the neural mechanisms of timing, it is useful to distinguish between sensory and motor timing tasks. In contrast with sensory timing, motor timing requires the active internal generation of events. For this reason, we propose that sensory and motor timing may rely on networks operating in low-gain (no self-sustaining activity) and high-gain regimes, respectively. Previous studies have demonstrated that randomly connected recurrent networks in low-gain regimes can discriminate temporal stimuli on the basis of hidden states (for example, short-term synaptic plasticity)[13,16,44]. In our model, timing arises entirely from the active state of neural networks. For this reason, our framework is particularly relevant to motor tasks, which require the active generation of temporal or spatiotemporal patterns rather than the discrimination of the temporal features of sensory stimuli.

Biological plausibility

Our results provide an existence proof that recurrent plasticity can suppress the chaotic behavior of specific trajectories of recurrent networks. Nevertheless, it remains to be determined whether recurrent neural networks in the brain operate in similar regimes. And if so, how such regimes are achieved, given that the learning rule that we used here is not biologically plausible.

Our work was inspired by a study that used the recursive least square algorithm to tune the weights of the recurrent units onto the output units[29]. Whether applied to the output or recurrent units, the approach relies on a supervised 'online' tuning of the weights to minimize the error between the actual firing rate of a unit and its target rate. Although the approach is 'delta rule like' in that it minimizes an error, it is computationally sophisticated and, as applied here, operates on a unrealistically fast timescale; however, as previously noted, there may be conditions under which more plausible rules can be used[29].

In addition, there is a separate target pattern that guides plasticity for each unit in the network in our implementation: a highly implausible biological scenario. Nevertheless, in one sense, the rule is more biologically plausible than traditional supervised learning rules: the rule does not require an external teacher to figure out the correct target pattern because the target trajectory is the innate internally generated trajectory. Thus, more realistic versions of this approach may be viable because which trajectories are 'burned in' is largely irrelevant, what matters is that networks settle on one (or a few) of the immense set of possible innate trajectories.

It is important to stress that our work was based on simple firing-rate units, as opposed to spiking units. Chaos control and chaos suppression in spike-based models present a more complex problem[21,23–25] and our work does not directly speak to solving the problem of chaos in spiking networks. An initial step toward translating the current work to spiking models will be to first create spiking networks that exhibit the complex balanced dynamic regimes similar to the untrained firing-rate networks studied here. Although this has not yet been achieved, recent advances in understanding the dynamics of recurrent spiking networks[45] and the generation of simple trajectories in spiking recurrent networks[46] have taken steps in this direction.

Structure and mechanisms underlying stable trajectories

The presence of stable trajectories in an otherwise chaotic state space raises important questions in neuroscience and nonlinear systems as to why some network architectures exhibit this dynamic regime. In linear recurrent networks, the structure of the network, as analyzed through a number of techniques, including eigen and Schur decompositions, provides valuable keys to understand the dynamics of such systems[47]. However, predicting the behavior of a continuous-time nonlinear network from its connectivity matrix is still not possible in the general case. In addition, a key observation is that the interaction between the input connectivity and recurrent weights is important for the manner in which the network responds to external stimuli;

as shown here, the same network can respond very differently to different inputs after training (**Figs. 2**–**6**). Steps toward understanding this interaction and the dynamics in response to external inputs have been taken for both discrete-time linear networks[33] and continuous-time nonlinear networks[30], but it remains impossible in continuous nonlinear networks to predict the modes of activity or describe why some trajectories are locally stable and others are not.

Despite the limitations in mathematically analyzing and predicting the dynamics of nonlinear networks, it is interesting that analysis of the connectivity patterns and network structure revealed highly reproducible, non-random signatures in the recurrent weight matrices. For example, innate training produced a robust increase in the median absolute weight, resulting in a non-Gaussian long-tailed weight distribution (**Fig. 7a**).

After training, there were global changes in the entire family of trajectories generated by the RRN. The untrained trajectories diverged at a slower rate, but were still not stable in the sense that they could not return to their original path after a perturbation. The changes captured by the statistics and structure of the connectivity matrix likely contribute to the global changes in the untrained trajectories, but not the trajectory specific training effects, as these are specific to a small subset of trajectories and, at some level, must rely on the creation of specific basins of attraction around the trained trajectories.

Conclusions and experimental predictions

In our model, recurrent cortical circuits would exhibit preferred or learned neural trajectories. That is, although spontaneously active networks exhibit complex, but unstable, trajectories, trained stimuli elicit preferred stable trajectories that can last many seconds and are highly robust to noise. The presence of these two modes of activity is consistent with experimental evidence[34], and we found (**Fig. 4**) that the networks studied here reproduce the experimentally observed decrease in neural variance in response to stimulus onset. An experimentally testable prediction is that the magnitude of the variance drop and its duration is stimulus specific and dependent on learning. That is, the decrease in variance in response to overtrained stimuli will be larger and longer lasting than that to novel or irrelevant stimuli.

Our results demonstrate that, in principle, recurrent plasticity can locally suppress chaos and substantially enhance the computational power of recurrent networks. A phenomenon observed here is that of dynamic attractors (locally stable transient channels), which account for the ability of a network to not only generate complex patterns (for example, the hand-writing patterns of **Fig. 2**), but to be able to return to the pattern in response to large perturbations. To the best of our knowledge, this is the first description of a high-dimensional nonlinear system capable of this level of robustness. The demonstration of how to create stable trajectories suggests a previously unknown neural computational framework. Specifically, an influential theory in neuroscience has been that some computations are instantiated by the activity of neural networks converging to steady-state patterns that represent fixed-point attractors in neural state space[48,49]. In contrast with these standard attractor models, in our framework (dynamic attractor computing), computations arise from the voyage through stable channels in state space rather than the arrival at any one given location.

It has often been suggested that neural circuits may operate on the edge of chaos, referring to a dynamic regime that offers desirable computational features while avoiding chaotic behavior. But theoretical and experimental studies have shown that the brain does exhibit full-blown chaotic regimes[21–25,50], and experimental evidence and common sense also tell us that neural circuits can generate reproducible neural trajectories that are critical for sensory and motor processing. Our results reconcile these observations and suggest that, rather than operating on the edge of chaos, the same network can produce both locally stable and chaotic trajectories.

METHODS

Methods and any associated references are available in the online version of the paper.

Note: Supplementary information is available in the online version of the paper.

ACKNOWLEDGMENTS
We thank A. Garfinkel and R. Huerta for helpful discussions and comments on the manuscript. This work was supported by the US National Institutes of Health (NS077340), the National Science Foundation (II-1114833), the Pew Charitable Trusts, and Consejo Nacional de Investigaciones Científicas y Técnicas (Argentina).

AUTHOR CONTRIBUTIONS
R.L. and D.V.B. designed the experiments and wrote the code, and R.L. performed most of the simulations and data analysis. R.L. designed and performed the stability and structural experiments. D.V.B. conceived of the approach, and R.L. and D.V.B. wrote the paper.

COMPETING FINANCIAL INTERESTS
The authors declare no competing financial interests.

1. Mauk, M.D. & Buonomano, D.V. The neural basis of temporal processing. *Annu. Rev. Neurosci.* **27**, 307–340 (2004).
2. Buhusi, C.V. & Meck, W.H. What makes us tick? Functional and neural mechanisms of interval timing. *Nat. Rev. Neurosci.* **6**, 755–765 (2005).
3. Ivry, R.B. & Schlerf, J.E. Dedicated and intrinsic models of time perception. *Trends Cogn. Sci.* **12**, 273–280 (2008).
4. Church, R.M., Meck, W.H. & Gibbon, J. Application of scalar timing theory to individual trials. *J. Exp. Psychol. Anim. Behav. Process.* **20**, 135–155 (1994).
5. Durstewitz, D. Self-organizing neural integrator predicts interval times through climbing activity. *J. Neurosci.* **23**, 5342–5353 (2003).
6. Simen, P., Balci, F., de Souza, L., Cohen, J.D. & Holmes, P. A model of interval timing by neural integration. *J. Neurosci.* **31**, 9238–9253 (2011).
7. Miall, C. The storage of time intervals using oscillating neurons. *Neural Comput.* **1**, 359–371 (1989).
8. Matell, M.S., Meck, W.H. & Nicolelis, M.A. Interval timing and the encoding of signal duration by ensembles of cortical and striatal neurons. *Behav. Neurosci.* **117**, 760–773 (2003).
9. Ahrens, M.B. & Sahani, M. Observers exploit stochastic models of sensory change to help judge the passage of time. *Curr. Biol.* **21**, 200–206 (2011).
10. Buonomano, D.V. & Laje, R. Population clocks: motor timing with neural dynamics. *Trends Cogn. Sci.* **14**, 520–527 (2010).
11. Medina, J.F. & Mauk, M.D. Computer simulation of cerebellar information processing. *Nat. Neurosci.* **3** (suppl.), 1205–1211 (2000).
12. Buonomano, D.V. & Mauk, M.D. Neural network model of the cerebellum: temporal discrimination and the timing of motor responses. *Neural Comput.* **6**, 38–55 (1994).
13. Buonomano, D.V. & Merzenich, M.M. Temporal information transformed into a spatial code by a neural network with realistic properties. *Science* **267**, 1028–1030 (1995).
14. Durstewitz, D. & Deco, G. Computational significance of transient dynamics in cortical networks. *Eur. J. Neurosci.* **27**, 217–227 (2008).
15. Rabinovich, M., Huerta, R. & Laurent, G. Transient dynamics for neural processing. *Science* **321**, 48–50 (2008).
16. Buonomano, D.V. & Maass, W. State-dependent computations: spatiotemporal processing in cortical networks. *Nat. Rev. Neurosci.* **10**, 113–125 (2009).
17. Hahnloser, R.H.R., Kozhevnikov, A.A. & Fee, M.S. An ultra-sparse code underlies the generation of neural sequence in a songbird. *Nature* **419**, 65–70 (2002).
18. Long, M.A., Jin, D.Z. & Fee, M.S. Support for a synaptic chain model of neuronal sequence generation. *Nature* **468**, 394–399 (2010).
19. Crowe, D.A., Averbeck, B.B. & Chafee, M.V. Rapid sequences of population activity patterns dynamically encode task-critical spatial information in parietal cortex. *J. Neurosci.* **30**, 11640–11653 (2010).
20. Li, J.X. & Lisberger, S.G. Learned timing of motor behavior in the smooth eye movement region of the frontal eye fields. *Neuron* **69**, 159–169 (2011).
21. London, M., Roth, A., Beeren, L., Hausser, M. & Latham, P.E. Sensitivity to perturbations *in vivo* implies high noise and suggests rate coding in cortex. *Nature* **466**, 123–127 (2010).

22. Izhikevich, E.M. & Edelman, G.M. Large-scale model of mammalian thalamocortical systems. *Proc. Natl. Acad. Sci. USA* **105**, 3593–3598 (2008).
23. van Vreeswijk, C. & Sompolinsky, H. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* **274**, 1724–1726 (1996).
24. Brunel, N. Dynamics of networks of randomly connected excitatory and inhibitory spiking neurons. *J. Physiol. Paris* **94**, 445–463 (2000).
25. Banerjee, A., Series, P. & Pouget, A. Dynamical constraints on using precise spike timing to compute in recurrent cortical networks. *Neural Comput.* **20**, 974–993 (2008).
26. Sompolinsky, H., Crisanti, A. & Sommers, H.J. Chaos in random neural networks. *Phys. Rev. Lett.* **61**, 259–262 (1988).
27. Monteforte, M. & Wolf, F. Dynamic flux tubes form reservoirs of stability in neuronal circuits. *Phys. Rev. X* **2**, 041007 (2012).
28. Jaeger, H. & Haas, H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78–80 (2004).
29. Sussillo, D. & Abbott, L.F. Generating coherent patterns of activity from chaotic neural networks. *Neuron* **63**, 544–557 (2009).
30. Rajan, K., Abbott, L.F. & Sompolinsky, H. Stimulus-dependent suppression of chaos in recurrent neural networks. *Physical Rev. E Stat. Nonlin. Soft Matter Phys.* **82**, 011903 (2010).
31. Doya, K. in *Proc. IEEE Int. Symp. Circuits and Syst.* 2777–2780 (1992).
32. Jaeger, H., Maass, W. & Principe, J. Special issue on echo state networks and liquid state machines. *Neural Netw.* **20**, 287–289 (2007).
33. Ganguli, S., Huh, D. & Sompolinsky, H. Memory traces in dynamical systems. *Proc. Natl. Acad. Sci. USA* **105**, 18970–18975 (2008).
34. Churchland, M.M. *et al.* Stimulus onset quenches neural variability: a widespread cortical phenomenon. *Nat. Neurosci.* **13**, 369–378 (2010).
35. Song, S., Sjostrom, P.J., Reigl, M., Nelson, S. & Chklovskii, D.B. Highly nonrandom feature of synaptic connectivity in local cortical circuits. *PLoS Biol.* **3**, e66 (2005).
36. Watts, D.J. & Strogatz, S.H. Collective dynamics of 'small-world' networks. *Nature* **393**, 440–442 (1998).
37. Janssen, P. & Shadlen, M.N. A representation of the hazard rate of elapsed time in the macaque area LIP. *Nat. Neurosci.* **8**, 234–241 (2005).
38. Bueti, D., Lasaponara, S., Cercignani, M. & Macaluso, E. Learning about time: plastic changes and interindividual brain differences. *Neuron* **75**, 725–737 (2012).
39. Coull, J. & Nobre, A. Dissociating explicit timing from temporal expectation with fMRI. *Curr. Opin. Neurobiol.* **18**, 137–144 (2008).
40. Merchant, H., Zarco, W., Pérez, O., Prado, L. & Bartolo, R. Measuring time with different neural chronometers during a synchronization-continuation task. *Proc. Natl. Acad. Sci. USA* **108**, 19784–19789 (2011).
41. Pastalkova, E., Itskov, V., Amarasingham, A. & Buzsaki, G. Internally generated cell assembly sequences in the rat hippocampus. *Science* **321**, 1322–1327 (2008).
42. Ivry, R.B., Keele, S.W. & Diener, H.C. Dissociation of the lateral and medial cerebellum in movement timing and movement execution. *Exp. Brain Res.* **73**, 167–180 (1988).
43. Medina, J.F., Garcia, K.S., Nores, W.L., Taylor, N.M. & Mauk, M.D. Timing mechanisms in the cerebellum: testing predictions of a large-scale computer simulation. *J. Neurosci.* **20**, 5516–5525 (2000).
44. Buonomano, D.V. Decoding temporal information: a model based on short-term synaptic plasticity. *J. Neurosci.* **20**, 1129–1141 (2000).
45. Litwin-Kumar, A. & Doiron, B. Slow dynamics and high variability in balanced cortical networks with clustered connections. *Nat. Neurosci.* **15**, 1498–1505 (2012).
46. Liu, J.K. & Buonomano, D.V. Embedding multiple trajectories in simulated recurrent neural networks in a self-organizing manner. *J. Neurosci.* **29**, 13172–13181 (2009).
47. Goldman, M.S. Memory without feedback in a neural network. *Neuron* **61**, 621–634 (2009).
48. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **79**, 2554–2558 (1982).
49. Wang, X.J. Synaptic reverberation underlying mnemonic persistent activity. *Trends Neurosci.* **24**, 455–463 (2001).
50. Skarda, C.A. & Freeman, W.J. How brains make chaos in order to make sense of the world. *Behav. Brain Sci.* **10**, 161–173 (1987).

ONLINE METHODS

Network equations. The network dynamics of the model is described by[26,28]

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} W_{ij}^{\text{Rec}} r_j + \sum_{j=1}^{2} W_{ij}^{\text{In}} y_j + I_i^{\text{noise}} \qquad (1)$$
$$z = \sum_{j=1}^{N} W_{ij}^{\text{Out}} r_j$$

where $r_i = \tanh(x_i)$ represents the activity level, often referred to as the firing rate (even though it takes on negative values) of recurrent unit x_i ($i = 1, \ldots, N$). The variable y represents the input units ($i = 1{,}2$), and z is the output. $N = 800$ is the network size (number of recurrent units) and $\tau = 10$ ms is the unit time constant. The recurrent connectivity is represented by the sparse $N \times N$ matrix $\mathbf{W}^{\text{Rec}}$, with nonzero initial values randomly chosen from a Gaussian distribution with zero mean and s.d., $g/\sqrt{p_c N}$, where g is the synaptic strength scaling coefficient, and $p_c = 0.1$ is the connection probability between units ($p_c = 0.25$ was used because it enhanced resistance to noise; **Fig. 4**). As with previous studies, in the high-gain regime, we used g values in the range of 1.5 (**Figs. 2** and **4**) to 1.8 (**Figs. 1**, **3** and **5–7**)[29,30]; performance was very similar across a wide range of g values, whereas higher values tended to generate more complex trajectories and require more training. The activity of the network was readout by z through the $1 \times N$ output connectivity matrix $\mathbf{W}^{\text{Out}}$, with initial values drawn from a Gaussian distribution with zero mean and s.d., $1/\sqrt{N}$. The input weight vector, $\mathbf{W}^{\text{In}}$, is drawn from a Gaussian distribution with zero mean and unit variance. The values of y are 0, except during an input pulse comprising a step with 50-ms duration and amplitude of 5 (except for **Fig. 2**, where the amplitude was 2). In the case of multiple inputs, values of both inputs were zero, except during the time window in which one input was briefly turned on in a given trial. A noise current is included as a $N \times 1$ random vector $\mathbf{I}^{\text{noise}}$ drawn from a Gaussian distribution with a s.d. of I_0 and a zero mean. I_0 was constant during a trial and equal to 0.001 unless stated otherwise (training was successful with s.d. as high as 0.1, as in **Fig. 4**; however, many more training trials are need to converge).

In the control simulations when feedback from the output unit was present (echo-state architecture), an additional feedback term was included in equation (1), leading to

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} W_{ij}^{\text{Rec}} r_j + \sum_{j=1}^{2} W_{ij}^{\text{In}} y_j + W_i^{Fb} z + I_i^{\text{noise}} \qquad (2)$$
$$z = \sum_{j=1}^{N} W_{ij}^{\text{Out}} r_j$$

This equation represents the traditional echo-state architecture[28,51]. It is the same as equation (1), except for the presence of the feedback term $\mathbf{W}^{\text{Fb}} \cdot z$, where $\mathbf{W}^{\text{Fb}}$ is a length N vector with elements drawn from a uniform distribution between −1 and 1. In this architecture, only the $\mathbf{W}^{\text{Out}}$ weights were plastic, and obeyed the same learning rule for all architectures (see below)[29]. It is possible that the poor performance of the echo state architecture (**Fig. 3**) was poor because the feedback was flat throughout most of the trial. Thus, we ran additional simulations with a separate feedback unit that learned its innate output pattern. Performance was still substantially worse than that of innate training architecture.

Recurrent learning rule, innate training. The key step to the approach defined here is to train the recurrent network dynamics to generate a pattern that the RRN is already capable of producing, as opposed to the conventional strategy of training it to produce a pattern based on the desired output. We refer to the target pattern as the innate trajectory and define it as the trajectory generated in the absence of noise and starting from an arbitrary initial condition; there are of course a vast number of potential innate trajectories, and there is nothing special about any of them except that they clearly fall in the domain of trajectories that the network can generate. The innate trajectory could be chosen in the presence of noise; here, the innate trajectory was generated in the absence of noise by a specific input simply because it allows for a fairly standardized definition. In all results presented here, the target (innate) trajectory of the recurrent network was chosen in the absence of noise. Choosing the innate trajectories in the presence of noise yielded similar results (under the conditions tested; **Supplementary Fig. 7**). However, choosing a foreign trajectory (that is, training a network on the innate trajectory of another network) did not produce effective training (at least over the 30 training trials examined). We did not address the problem of what constitutes an achievable or an optimal target trajectory. Rather, given that the precise structure of the stable target trajectory is largely irrelevant, we provide a practical choice of a recurrent target by using an innate pattern; this approach guarantees that the target falls in the domain of possible trajectories for the network.

The learning rule used to train the plastic recurrent units is based on the recursive least squares (RLS) rule[52], which we implemented according to the FORCE (first-order reduced and controlled error) algorithm[29]. The rule was applied to a subset of the recurrent synapses in the network; similar performance was observed in the range of 60–95% (most simulations presented here imposed the condition that all the synapses onto 60% of the units were plastic). The weight update for the synapses (plastic) onto recurrent unit i was

$$W_{ij}^{\text{Rec}}(t) = W_{ij}^{\text{Rec}}(t - \Delta t) - e_i(t) \sum_{k \in \mathrm{B}(i)} P_{jk}^{i}(t) r_k(t) \qquad (3)$$

where $\mathbf{B}(i)$ is the subset of recurrent units presynaptic to unit i, and e_i represents the individual error of unit i defined by

$$e_i(t) = r_i(t) - R_i(t) \qquad (4)$$

where $r_i(t)$ is the activity of unit i before the weight update, and $R_i(t)$ is the innate target activity of that unit. The innate activity $R_i(t)$ is recorded before the training begins by letting the network evolve in time under the same conditions that will be used during training (same input brief input, but in the absence of noise). $\mathbf{P}^i$ (one for each recurrent plastic unit i) is a square matrix that estimates the inverse of the correlation matrix of the presynaptic inputs to element i ($\mathbf{B}(i)$), and is updated by

$$P_{jk}^{i}(t) = P_{jk}^{i}(t - \Delta t) - \frac{\sum_{m \in \mathrm{B}(i)} \sum_{n \in \mathrm{B}(i)} P_{jm}^{i}(t - \Delta t) r_m(t) r_n(t) P_{nk}^{i}(t - \Delta t)}{1 + \sum_{m \in \mathrm{B}(i)} \sum_{n \in \mathrm{B}(i)} r_m(t) P_{mn}^{i}(t - \Delta t) r_n(t)} \qquad (5)$$

Training procedure. Harvest an innate trajectory by letting the network evolve according to equation (1) in response to the input in the absence of noise; record the activity $R_i(t)$ for all recurrent units in the training window defined by $[t_{\text{off}}{:}t_{\text{end}}]$, where t_{off} is the offset time of the input pulse and t_{end} is the end point of the training window. Train the recurrent plastic weights with the innate algorithm as defined by equations (3–5) (the network evolves according to equation (1)) in the presence of noise and with random initial conditions for a number of training loops (training generally converges in between a few loops and a few dozen loops, depending on the duration of the training window); I_{noise} is Gaussian with zero mean and s.d. I_0. After training the recurrent units, the readout unit can also be trained using previously proposed algorithms in the interval $[t_{\text{off}}{:}t_{\text{end}}]$. Test (run) the network with the trained (fixed) set of recurrent and readout weights, again in the presence of noise and random initial conditions.

Over the course of training the total change in synaptic weights converges to an asymptote, but generally does not reach zero because plasticity is driven by noise. In the simulations shown here, a fixed number of training loops was used. In the simulations in **Figures 1**–**3** and **5**–**7**, there were between 20 and 30 training trials, which led to fairly asymptotic performance in these networks (**Supplementary Fig. 3**). In the high-noise simulations of **Figure 4**, hundreds of training trials were used.

Recurrent weights learning rule for control architecture. The learning rule for the recurrent units in the control simulations of fair recurrent plasticity architecture (**Fig. 3**) was implemented as described previously[29]. The implementation of this rule is fairly similar to the one used. The critical difference is that in fair recurrent plasticity architecture, the error for each of the recurrent units was the same backpropagated error from the output unit, as opposed to the local error of each recurrent unit. A consequence of this difference is that in fair recurrent plasticity architecture, training of the recurrent and output weights takes place simultaneously. As in innate training architecture, the presynaptic weights to

60% of recurrent units were plastic. As for the innate training (equation (3)), the weight update postsynaptic recurrent unit i is

$$W_{ij}^{\text{Rec}}(t) = W_{ij}^{\text{Rec}}(t - \Delta t) - e(t) \sum_{k \in \mathrm{B}(i)} P_{jk}^{i}(t) r_k(t) \tag{6}$$

but here, in contrast with equation (3), the error signal is the same for all recurrent plastic units, and was equal to the error signal for the readout unit (see also equation (9))

$$e(t) = \sum_j W_j^{\text{out}}(t - \Delta t) r_j(t) - f(t) \tag{7}$$

where $f(t)$ is the target function of the output unit. $\mathbf{P}^i$ is a square matrix (one for each recurrent plastic unit i, with each dimension equal to the number of units in $\mathbf{B}(i)$[29]. Its update rule is the same as in equation (5).

Readout weights learning rule (all architectures). The learning rule for the readout unit in all the results shown here was the same RLS/FORCE algorithm used in previous studies[28,29,52]. The readout weight update is defined as

$$W_i^{\text{Out}}(t) = W_i^{\text{Out}}(t - \Delta t) - e(t) \sum_j P_{ij}(t) r_j(t) \tag{8}$$

where the error, $e(t)$, is defined as in equation (7).

The weight update occurs at times separated by the small time interval Δt, which may be larger than the time step δt for the numerical integration of equation (1) (or equation (2) in the case of echo state architecture) (we used $\delta t = 1$ ms and $\Delta t = 2$ ms). Note that $\mathbf{W}^{\text{Out}}$ enters equation (9) with non-updated value, that is, it is evaluated at the earlier time $t - \Delta t$ rather than at t. $\mathbf{P}$ is a running estimate of the inverse of the correlation matrix of the network rates $\mathbf{r}$ plus a regularization term

$$P_{ij}(t) = P_{ij}(t - \Delta t) - \frac{\sum_m \sum_n P_{jm}(t - \Delta t) r_m(t) r_n(t) P_{nj}(t - \Delta t)}{1 + \sum_m \sum_n r_m(t) P_{mn}(t - \Delta t) r_n(t)} \tag{9}$$

The only difference between equations (5) and 9 is that in equation (5), each matrix $\mathbf{P}^{\mathbf{i}}$ is specific to a subset of presynaptic recurrent units to recurrent postsynaptic unit i, whereas in equation (9) a single matrix $\mathbf{P}$ refers to all presynaptic recurrent units (which all contact the output unit).

Output target functions. The handwritten targets used in **Figure 2** were obtained using custom Matlab code and a Wacom Bamboo Pen Tablet. x and y pen positions were originally sampled at approximately 50 Hz, then low-pass filtered and resampled with interpolation to 1 kHz (corresponding to the time step of 1 ms used for all simulations). In **Figure 3**, the readout target function, $f(t)$, is defined by a constant value with a Gaussian pulse at time t_{delay} (0.25, 0.5, 1, 2, 3, …, 8 s), where $t = 0$ corresponds to the offset of the input.

Noise analysis. Network performance in the presence of noise (**Fig. 5**) was quantified by estimating the correlation between two trajectories from two different runs for each network within each condition: a template trajectory without noise and a test trajectory with noise. A high correlation indicates a high degree of reproducibility. We first calculated the Pearson correlation coefficient between the two trajectories for each recurrent unit, then averaged across units (after Fisher transformation), and then averaged across networks. Correlation was calculated for the duration of the trained window only (0 – 2 s). Model and test trajectories had the same pre-stimulus initial conditions. Noise was continuously injected into all units in the recurrent network only after input was off, with a Gaussian distribution with zero mean and s.d. I_0 (equation (1)). Noise amplitude is to be compared to total absolute incoming synaptic weights to a unit (averaged across units); i.e., comparing the average sizes of the second and fourth terms on the right-hand side of equation (1). A noise amplitude $I_0 = 1.0$ in **Figure 5** corresponds to 7% of the average total absolute incoming synaptic weight.

Largest Lyapunov exponent (LLE) estimates. We estimated the local LLE (λ, also known as finite-time LLE) in a manner similar to that described previously[28] with some improvements described elsewhere[53,54] (see **Supplementary Fig. 8**). We computed the distance between perturbed and unperturbed trajectories in state space as a function of time, then found an interval in which the log(distance) versus time plot is linear with a well-defined slope. To this end, the network was first run with random initial conditions and no noise to get a fiducial trajectory $\mathbf{x}(t)$ in 800-dimensional state space. Ten segments of 1,000-ms duration were extracted from the fiducial trajectory, which served as the unperturbed trajectories. The first segment $\mathbf{x}_1(t)$ started 100 ms after the input went off, at state $\mathbf{x}(100)$; the second $\mathbf{x}_2(t)$, third $\mathbf{x}_3(t)$, etc. segments started 100 ms after the previous, at states $\mathbf{x}(200)$, $\mathbf{x}(300)$, etc. The perturbed trajectories were obtained as follows. Each segment was perturbed at its initial time point (for instance, for the first segment, the state $\mathbf{x}(100)$ was perturbed) by a uniform-noise vector of size 10^{-7} and then the network was run for 1,000 ms; the perturbation was performed ten times, leading to ten perturbed trajectories $\mathbf{y}_{ij}(t)$ for each unperturbed segment $\mathbf{x}_i(t)$. We then computed the average $d_i(t)$ of the Euclidean distances between each of the ten perturbed trajectories and the unperturbed trajectory (for each of the ten segments) in 800-dimensional state space as a function of time (0–1,000 ms):

$$d_i(t) = \frac{1}{10}(\|\mathbf{x}_i(t) - \mathbf{y}_{i1}(t)\| + \|\mathbf{x}_i(t) - \mathbf{y}_{i2}(t)\| + \ldots + \|\mathbf{x}_i(t) - \mathbf{y}_{i10}(t)\|)$$

We then normalized it to the initial distance (0 ms), that is, the size of the perturbation, computed the logarithm and averaged across all segments: $h(t) = \frac{1}{10}\sum_{i=1}^{10} \log[d_i(t)/d_i(0)]$. This procedure was repeated ten times for each of the ten networks. We visually searched for a portion in which all ten repetitions have a linear shape with a well-defined slope of at least 300-ms duration in 100–900-ms range; the average slope of the linear fits was the local LLE estimate for each network at each condition (pre-training, post-training, post-training outside the trained interval). In **Figure 6a**, for visualization purposes, we plot the average of the ten repetitions for each network: $\frac{1}{10}\sum_{\text{repetitions}} h(t)$. In the condition post-training outside the trained interval, the fiducial trajectory started 8 s after the input went off (6 s after the end of the training window). Note that any trajectory not converging to a fixed point will have at least one zero Lyapunov exponent (in the direction of the flow); if the trajectory is stable, then it will be the largest Lyapunov exponent, all other exponents being negative.

Network structure: clustering coefficients. Local clustering coefficients were calculated using the generalization to directed, weighted networks proposed previously[55]. Cyclic clustering coefficients correspond previous findings (c.f. Table 1 of ref. 55); non-cyclic clustering coefficients pool all 'middlemans', 'ins' and 'outs'. Cyclic and non-cyclic clusters are mutually exclusive: they sum up to the total number of undirected clusters. The values of the (weighted) clustering coefficients, however, are not restricted. As these definitions assume a positive-definite weight matrix $\mathbf{W}^{\text{Rec}}$, all clustering coefficients were calculated based on the absolute values of the weights. All self-connections were excluded from the calculation. Shuffling of the weights was performed by keeping the binary connectivity and randomly reassigning all nonzero weights; thus, for each network, all three conditions (pre-train, post-train, and post-train shuffled) have the same binary connectivity.

Statistics. All error bars and measures of dispersion represent s.e.m unless otherwise noted (for example, MAD). All P values are two tailed.

Parameter values for Figures. Figures 1 and **5–7**, and **Supplementary Figures 3–7**: $N = 800$, $g = 1.8$, $p_c = 0.1$; 20 training loops; noise amplitude $I_0 = 0.001$ unless noise was parametrically varied (**Fig. 5**). Perturbation in **Figure 1c**: input amplitude = 5, input duration 50 ms. For these figures, a set of ten networks working as seeds was defined; all training and testing procedures were applied to this same set of networks.

Figure 3, and **Supplementary Figures 1** and **2**: $N = 800$, $g = 1.5$, $p_c = 0.1$; 30 training loops; noise amplitude $I_0 = 0.001$. Input amplitude = 5, input duration 50 ms. To make a controlled comparison across architectures, a second set of ten seed networks was defined; all training and testing procedures were applied to this same set of networks. **Figure 2**: $N = 800$, $g = 1.5$, $p_c = 0.1$; 30 training loops. Input amplitude = 2, input duration 50 ms. **Figure 4a**: $N = 800$, $g = 1.5$,

$p_c = 0.25$; 500 training loops; noise amplitude $I_0 = 1.5$. Input amplitude = 5, input duration 50 ms.

51. Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks. *GMD Report No. 148 (German National Research Center for Computer Science)* (2001).
52. Haykin, S. *Adaptive Filter Theory* (Prentice Hall, 2002).
53. Kantz, H. A robust method to estimate the maximal Lyapunov exponent of a time series. *Phys. Lett. A* **185**, 77–87 (1994).
54. Boffetta, G., Lacorata, G., Radaelli, G. & Vulpiani, A. Detecting barriers to transport: a review of different techniques. *Physica D* **159**, 58–70 (2001).
55. Fagiolo, G. Clustering in complex directed networks. *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* **76**, 026107 (2007).