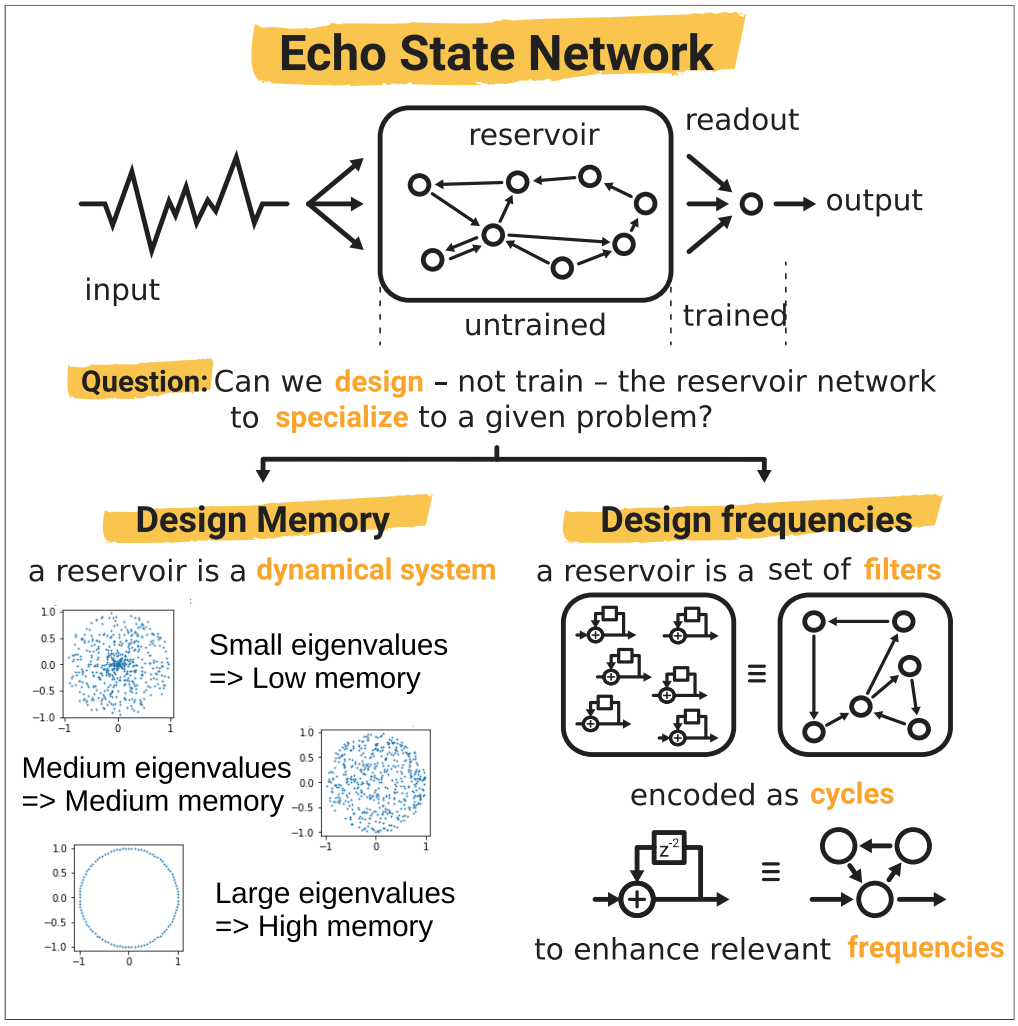
iScience

CellPress
OPEN ACCESS

Article

# Tailoring Echo State Networks for Optimal Learning




Pau Vilimelis Aceituno, Gang Yan, Yang-Yu Liu

yyl@channing.harvard.edu

## HIGHLIGHTS

Adapting the frequency response of a reservoir improves its performance

The frequency response of a reservoir can be tuned by adding or removing cycles

The memory of a reservoir network is controlled by the correlations between neurons

The correlations between neurons are controlled by the spectra of the network

Aceituno et al., iScience *23*, 101440
September 25, 2020 
https://doi.org/10.1016/j.isci.2020.101440

iScience

CellPress
OPEN ACCESS

Article

# Tailoring Echo State Networks for Optimal Learning

Pau Vilimelis Aceituno,[1,2] Gang Yan,[3] and Yang-Yu Liu[1,4,*]

## SUMMARY

**As one of the most important paradigms of recurrent neural networks, the echo state network (ESN) has been applied to a wide range of fields, from robotics to medicine, finance, and language processing. A key feature of the ESN paradigm is its reservoir—a directed and weighted network of neurons that projects the input time series into a high-dimensional space where linear regression or classification can be applied. By analyzing the dynamics of the reservoir we show that the ensemble of eigenvalues of the network contributes to the ESN memory capacity. Moreover, we find that adding short loops to the reservoir network can tailor ESN for specific tasks and optimize learning. We validate our findings by applying ESN to forecast both synthetic and real benchmark time series. Our results provide a simple way to design task-specific ESN and offer deep insights for other recurrent neural networks.**

## INTRODUCTION

As a promising paradigm of recurrent neural networks, echo state network (ESN) has a *reservoir* of neurons with randomly assigned and fixed synaptic connections (Jaeger 2001a, 2001b, 2002; Jaeger and Haas, 2004). For the ESN to model and predict specific temporal patterns, only the weights of output neurons need to be learned from training data. Owing to its simplicity, ESN and its variants have been applied to many different tasks such as electric load forecasting (Deihimi and Showkati, 2012), robotic control (Plöger et al., 2003), epilepsy forecasting (Buteneers et al., 2008), stock price prediction (Lin et al., 2009), grammar processing (Tong et al., 2007), and many others (Coulibaly, 2010; Newton and Smith, 2012; Pathak et al., 2018; Verplancke et al., 2010).

Over the last decade, a plethora of studies have focused on finding good reservoirs. Those studies fall broadly into two categories. *The first is systematical parameter search*. For specific tasks, this outperforms the classical Monte Carlo reservoir selection (Deng and Zhang, 2006; Ferreira and Ludermir, 2011; Jiang et al., 2008; Liebald, 2004; Rodriguez et al., 2019). Yet, this systematical parameter search is very time consuming and does not offer a significant performance improvement or better mechanistic understanding. *The second is particular reservoir characteristics*. Many studies have explored reservoirs with some particular characteristics that make them desirable, typically with long memory (Farkaš et al., 2016; Rodan and Tiňo, 2012; Strauss et al., 2012) or "rich" dynamics (Boedecker et al., 2012; Ozturk et al., 2007). However, the desirability of those reservoir characteristics is typically task-specific, rather than applicable to general tasks. Here, we propose a new strategy. We first focus on a general mechanistic understanding of the reservoir dynamics, which then helps us optimize or tailor reservoirs in a task-specific manner. We find that the idea of tailoring reservoirs is applicable to general tasks.

Formally, the discrete-time dynamics of the simplest ESN (as shown in Figure 1 and in Supplemental Information Section I) with $N$ neurons, one input, and one output is governed by

$$x(t) = f(W\,x(t-1) + w_{in}\,u(t) + w_{ofb}\,y(t-1)), \quad \text{(Equation 1)}$$

$$y(t) = w_{out}[x(t), u(t)]^{\top}. \quad \text{(Equation 2)}$$

Here the state vector $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_N(t)]^{\top} \in \mathbb{R}^N$ denotes the state of the $N$ neurons at time $t$, $u(t) \in \mathbb{R}$ is the input signal at time $t$, and $y(t) \in \mathbb{R}$ is the output at time $t$. The extended state vector $[\mathbf{x}(t), u(t)]^{\top} \in \mathbb{R}^{N+1}$ is just the concatenation of $\mathbf{x}(t)$ and $u(t)$. There are various possibilities for the nonlinear function $f$, and here we take the hyperbolic tangent, as it is often done in ESN literature (Jaeger, 2002; Jaeger and Haas, 2004). The matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the weighted adjacency matrix of the reservoir network describing

[1]Channing Division of Network Medicine, Brigham and Women's Hospital, Harvard Medical School, Boston, MA 02115, USA

[2]Max Planck Institute for Mathematics in the Sciences, 04103 Leipzig, Germany

[3]School of Physics Science and Engineering, Tongji University, 200092 Shanghai, China

[4]Lead Contact

*Correspondence: yyl@channing.harvard.edu

https://doi.org/10.1016/j.isci.2020.101440

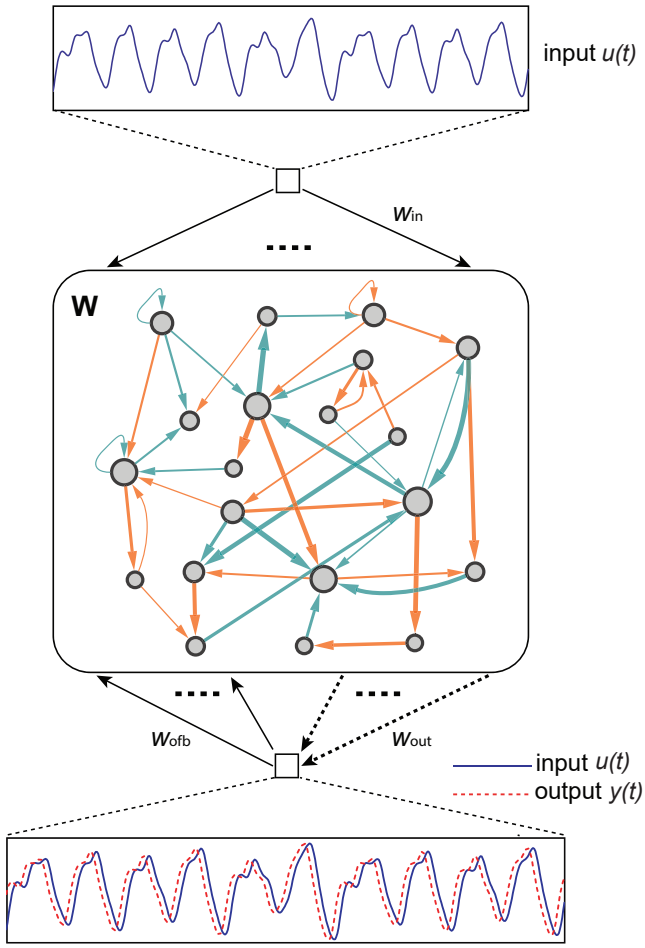



**Figure 1. The Basic Schema of an ESN**

The input signal $u(t)$ goes to each neuron in the reservoir with input weights $w_{in}$, the neurons send their states to their neighbors according to the matrix $\mathbf{W}$, and the contribution of each neuron to the output $y(t)$ is collected by $w_{out}$. The reservoir network may have self-loops, and can have both excitatory (yellow) and inhibitory (gray) synaptic connections.

the *fixed* wiring diagram of $N$ neurons in the reservoir. There is a rich literature on the conditions that the matrix $\mathbf{W}$ must fulfill (Buehner and Young, 2006; Gandhi et al., 2012; Jaeger, 2007; Yildiz et al., 2012). Here we adopt a conservative and simple condition that the reservoir must represent a stable dynamic system. The vector $\mathbf{w}_{in} \in \mathbb{R}^N$ captures the *fixed* weights of the input connections, which were drawn from a uniform distribution $\mathcal{U}[-1, 1]$. The vector $\mathbf{w}_{ofb} \in \mathbb{R}^N$ denotes the *fixed* weights of the feedback connections from the output to the $N$ neurons, which can induce instabilities if chosen carelessly and may be zero in some tasks (Jaeger, 2002). Finally, the row vector $\mathbf{w}_{out} \in \mathbb{R}^{1\times(N+1)}$ represents the *trainable* weights of the readout connections from the $N$ neurons and the input to the output.

A key feature of ESN is that $\mathbf{W}$, $\mathbf{w}_{in}$, and $\mathbf{w}_{ofb}$ are all fixed, and only $\mathbf{w}_{out}$ is trainable: $\mathbf{w}^*_{out} = \arg min_{\mathbf{w}_{out}} \sum_{t=t_0}^{t_0+T} (y(t) - \hat{y}(t))^2$, where $t_0$ is the starting time, $T$ is the training interval, and $\hat{y}(t)$ is the target output obtained from the training data (see Supplemental Information Section I for details). In other words, $\mathbf{w}^*_{out}$ is the linear regression weights approximating the desired output $\hat{y}(t)$ from the extended state vector $[\mathbf{x}(t), u(t)]^\top$, which can be easily solved. Hence, $\mathbf{w}^*_{out}$ captures the underlying mechanism of the dynamic system that produces the training data. Indeed, the right choice of $\mathbf{w}^*_{out}$ can be used to forecast, reconstruct, or filter nonlinear time series.

It is worth noticing that although ESN is easy to train and very flexible, it is often outperformed by more sophisticated methods requiring larger training dataset and longer training time (Hammami and Bedda,

2010), or *ad hoc* architectures (Wan, 1993). The reason why we focus on ESN in this work is because of its simplicity, which allows us to perform analytical calculations. Moreover, we want to explore how simple ideas from classical signal processing and network science can be applied to dissect ESN—a prototypical paradigm of recurrent neural networks.

## RESULTS

### ESN Performance Captured by Reservoir Spectrum

The success of ESN in tasks such as forecasting time series comes from the ability of its reservoir to retain memory of previous inputs (Cui et al., 2012). In ESN literature, this is quantified by the *memory capacity* (Jaeger, 2001a, 2001b):

$$M = \sum_{\tau=1}^{N} M_\tau, \qquad \text{(Equation 3)}$$

$$M_\tau = \max_{\mathbf{w}_{\text{out}}^\tau} \frac{\text{cov}^2(r(t-\tau), y_\tau(t))}{\text{var}(r(t-\tau))\text{var}(y_\tau(t))}$$

Here $r(t)$is a random variable drawn from a normal distribution $\mathcal{N}(0,1)$, serving as a random input; "cov" represents the covariance; $y_\tau(t)$ is the output as described in Equation 2; and $\mathbf{w}_{\text{out}}^\tau$ is obtained as a minimizer of the difference between $y_\tau(t)$ and $r(t-\tau)$.

To quantify the relationship between the reservoir dynamics and the memory capacity, we note that the extraction of information from the reservoir is made through a linear combination of the neurons' states. Hence, more linearly independent neurons would offer more variable states, and thus longer memory (Jaeger, 2005; Lukoševičius and Jaeger, 2009). To be more precise, we hypothesize that the memory capacity $M$ strongly depends on the average correlations among neuron states, which can be quantified as follows: $S = \langle P_{ij}^2 \rangle$with $P_{ij} = \frac{\text{cov}(x_i(t), x_j(t))}{\text{std}(x_i(t))\text{std}(x_j(t))}$, the Pearson correlation coefficient between the states of neurons $i$ and $j$, and $\text{std}(x_i(t))$ the standard deviation of the states of neuron $i$. Indeed, Figure 2A shows that for various network topologies there is a strong correlation between $S$ and $M$, which can also be justified analytically (see Supplemental Information Section VI). Thus, hereafter we only need to understand how the reservoir structure affects the neuron correlations.

For linear dynamical systems, the correlations between state variables depend on all the eigenvalues of the adjacency matrix (Boccaletti et al., 2006), with larger mean eigenvalue meaning lower correlations (see Supplemental Information Section VII for details). Our system (Equation 1) is nonlinear, but its first-order approximation is the identity function $f(z) = z$. Hence we can use the eigenvalues $\{\lambda_i\}$ of matrix $\mathbf{W}$ to approximately quantify how fast the input decays in the reservoir, and hence how poorly the ESN remembers. In other words, the eigenvalues of matrix $\mathbf{W}$ should be related to the memory capacity of the ESN. Indeed, we find that the average eigenvalue moduli:

$$\langle |\lambda| \rangle = 1/N \sum_{i=1}^{N} |\lambda_i|, \qquad \text{(Equation 4)}$$

strongly correlates with $S$ (Figure 2B) and therefore with $M$ as well.

Note that, as opposed to $M$ and $S$, $\langle |\lambda| \rangle$ is much easier to compute and is solely determined by the reservoir network. This offers us a simple measure to quantify the ESN memory capacity and hence its performance. For example, this explains two recent studies in which it was found that ring networks and orthogonalized networks have high memory capacities (Farkaš et al., 2016; Rodan and Tiňo, 2012), as both networks have large eigenvalues with respect to their spectral radii. Similarly, the modularity of the network can have an effect on its memory capacity (Rodriguez et al., 2019), which is explained by the effect of modularity in the eigenvalue distribution (Newman and Girvan, 2004). Moreover, $\langle |\lambda| \rangle$ is consistent with the effects of scaling the adjacency matrix to tune the spectral radius (Jaeger et al., 2007) and it extends to network topologies with a fixed spectral radius (see Supplemental Information Figures S1 and S2). Our result is also consistent with studies on Intrinsic Plasticit y, where the memory of the reservoir is increased by growing its entropy (Boedecker et al., 2009; Schrauwen et al., 2008), which is negatively correlated with its correlations. Finally, it suggests that $M$ can be maximized by using networks with large eigenvalues, the simplest one being a circulant network with degree 1 (Aceituno et al., 2019), which achieves $M = 20$, whereas the others go only up to $M = 17$ (see the example in Supplemental Information Section VIII).

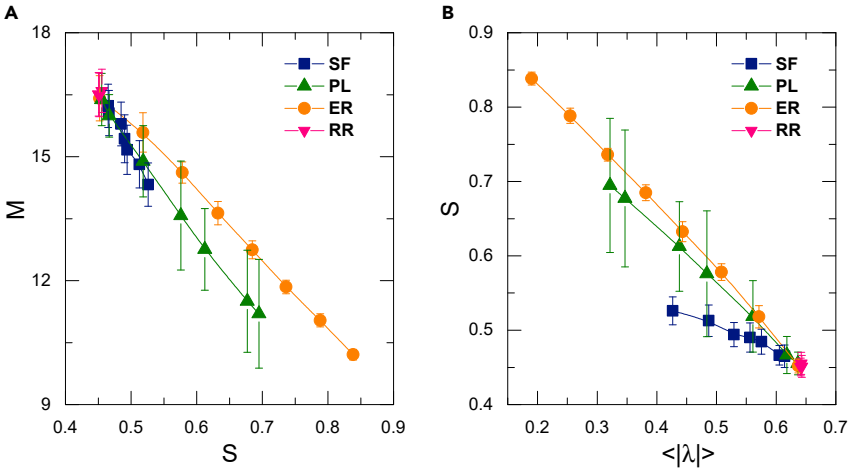



**Figure 2. Relations between Memory Capacity, Neuron Correlations, and Network Spectrum**

(A) Memory capacity $M$ versus average correlation of neuron states $S$.

(B) Average correlation of neuron states $S$ versus average eigenvalue modulus $\langle|\lambda|\rangle$. ESNs were created using reservoirs of $N = 400$ neurons and sequences of 4,000 random inputs chosen from $\mathcal{U}[-1, 1]$, and the error bars represent the standard deviation. ER represents reservoirs with structure generated by the classical Erdös-Rény (ER) random graphs, with edge weights drawn from a normal distribution and varying spectral radii. PL represents reservoirs with structures generated from Erdös-Rény random graphs, but the edge weights are drawn from a power-law (PL) distribution with varying exponent $\beta \in [2,5]$ and then normalized to have a spectral radius $\alpha = 1$. Lower $\beta$ renders lower $M$, higher $S$, and lower $\langle|\lambda|\rangle$. SF represents reservoirs with scale-free (SF) network structures with degree exponent $\gamma \in [2,6]$, and the edge weights are drawn from a normal distribution and then normalized to have a spectral radius $\alpha = 1$. More degree-heterogeneous networks render lower $M$, higher $S$, and lower $\langle|\lambda|\rangle$. RR represents reservoirs with structure generated by random regular (RR) graphs, with varying degrees and a spectral radius $\alpha = 1$. See Supplemental Information Sections III and V for more details. It is worth noticing that although the theoretical upper bound for the memory capacity of a reservoir is $M = N$ (Jaeger, 2001a, 2001b) and small input scalings (Farkaš et al., 2016) do achieve similar values, in our case the input scaling is large and thus the nonlinearity of the reservoir limits $M$ to be lower than 18. A more detailed numerical exploration of the dependency between the various network parameters and $M$, and between the network parameters and $\langle|\lambda|\rangle$, is presented in the Supplemental Information Section V. The network generation algorithms are presented in Supplemental Information Section III. All networks have a spectral radius $\alpha = 1$, except the ER random graphs where each point corresponds to a spectral radius in the range [0.2,1] to show the impact of spectral radius. It is worth noticing that although the theoretical upper bound for the memory capacity of a reservoir is $M = N$ (Jaeger, 2001a, 2001b) and small input scalings (Farkaš et al., 2016) do achieve similar values, in our case input scaling is large and thus the nonlinearity of the reservoir limits $M$ to be less than 18.

See also Figures S1–S3.

To further demonstrate the validity of $\langle|\lambda|\rangle$ as a proxy measure for the ESN performance, we consider the following tasks (Figures 3A–3C): (1) forecasting the chaotic Mackey-Glass time series (Mackey and Glass, 1977), which is a benchmark task to evaluate the performance of ESN (Jaeger and Haas, 2004; Lukoševicius and Jaeger, 2007); (2) forecasting the Laser Intensity time series (Hübner et al., 1989) downloaded from the Santa Fe Institute; and (3) classifying Spoken Arabic Digits (Hammami and Bedda, 2010) downloaded from the UCI Machine Learning Repository (Lichman, 2013). For each task, we consider ESNs with a wide range of reservoir topologies and parameters, and plot the ESN performance (in terms of forecasting error or failure rate) as a function of $\langle|\lambda|\rangle$ (Figures 3D–3F). We find that the optimal parameters for all reservoir networks are within a consistent range of $\langle|\lambda|\rangle$ (highlighted in pink). In other words, the performance of ESN is indeed captured by $\langle|\lambda|\rangle$, rather than by other reservoir characteristics networks.

## Adapting ESN in the Frequency Domain

Intuitively, a reservoir can be understood as a set of coupled filters that extract features from the input signal, and the readout simply selects the right combination of those features. Hence, the reservoir should be designed to extract the features that are relevant for the problem at hand, and these features can be expressed in the Fourier domain. This idea can be translated to machine learning terms through a geometric argument (Figure 4), which was made rigorous in Supplemental Information Section IX. In particular, we derived an upper bound for the forecasting error

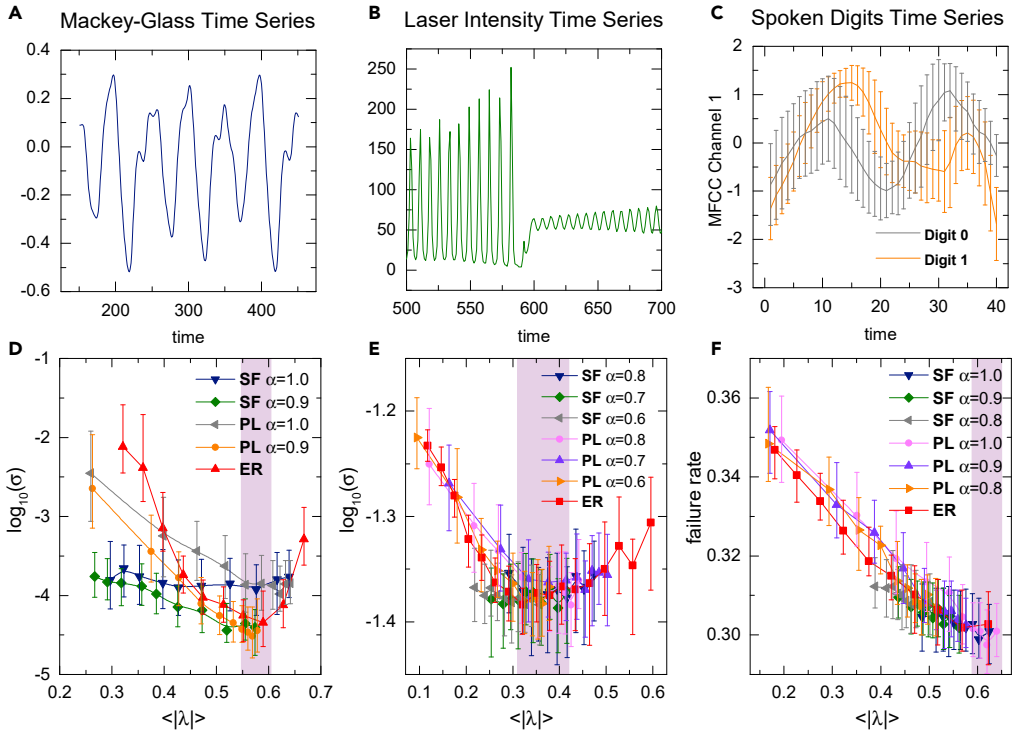



**Figure 3. Time Series Analyzed in This Work and the ESN Performance Explained by $\langle|\lambda|\rangle$**

(A) The classical Mackey-Glass time series (Mackey and Glass, 1977) with 500 data points.

(B) The Laser Intensity time series (Hübner et al., 1989) with 300 data points.

(C) The average value of the first mel-frequency cepstral coefficient (MFCC) Channel of the first Spoken Arabic Digit (Hammami and Bedda, 2010); the error bars represent standard deviations over the training dataset.

(D) The ESN forecasting performance for the Mackey-Glass time series.

(E) The ESN forecasting performance for Laser Intensity time series.

(F) The ESN failure classification rate for Spoken Arabic Digits. For each task, we use scale-free (SF) networks, Erdős-Rényi (ER) random graphs with homogeneous link weights, and ER random graphs with heterogeneous link weights following a power-law distribution (PL) as reservoirs (see Supplemental Information Section III for more details on reservoir generations). The SF and PL reservoirs have various spectral radii $\alpha$, chosen to be around the optimal value of $\alpha$ for the ER reservoirs. For each parameter set of each network type we created 200 ESN realizations, and then all the points obtained were grouped in 10 bins containing the same number of points. For each bin, we plotted the median $\langle|\lambda|\rangle$ against the median performance: $\log(\sigma)$ from Equation S5 for (D) and (E); and the failure rate for (F), with the error bars being the upper and lower quartiles, respectively. See Supplemental Information Section II for an expanded description of the three time series and the performance measurement.

$$\sigma \leq \frac{\sum_{i=1}^{N} |\mathcal{F}[x_i] \times \mathcal{F}[\hat{y}]|}{\sum_{i=1}^{N} |\langle \mathcal{F}[x_i], \mathcal{F}[\hat{y}] \rangle|}, \qquad \text{(Equation 5)}$$

where $\mathcal{F}[\bullet]$ stands for the Fourier Transform $\times$ and $\langle \bullet, \bullet \rangle$ are the cross and scalar products. In terms of signal processing, $|\mathcal{F}[x_i] \times \mathcal{F}[\hat{y}]|$ (or $|\langle \mathcal{F}[x_i], \mathcal{F}[\hat{y}] \rangle|$) can be expressed by how much the power spectral densities (PSD) of the neurons differ from (or resemble) $y$, respectively. This is quite a natural result, as it simply implies that if the time series of the variables $\mathbf{x}$ are similar to the target, then the readout will work better.

Thus, to achieve the optimal performance of ESN for any specific task, it is crucial to alter the PSD of the reservoir, which can be achieved by adding feedback loops with delay $L$ in our neurons, encoded as cycles of length $L$ in the reservoir network. We account for the strength of those cycles by using the following measure:

$$\rho_L = \frac{E_{L,s} - E_{L,-s}}{E}, \qquad \text{(Equation 6)}$$

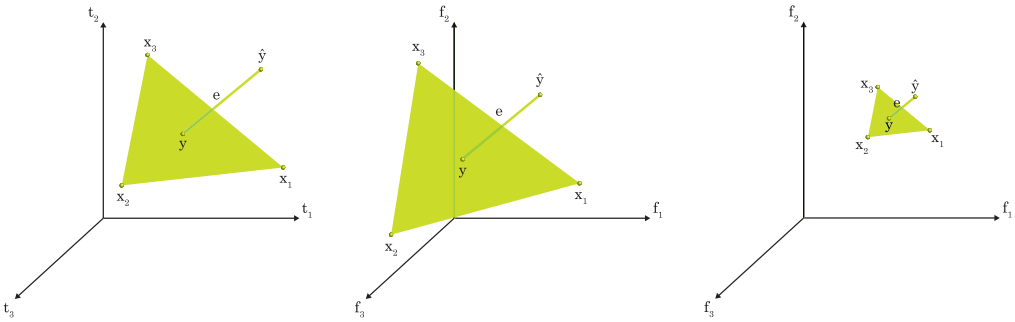



**Figure 4. Sketch of the Frequency Adaptation Argument**

(Left) The target output $\hat{y}$ is a time series of length $T$, represented by a vector in the corresponding space. A reservoir consists of $N$ nonlinear filters of the input time series, represented by $N$ points in that same space. The readout simply selects the point in the subspace spanned by the $N$ neurons that are closest to the target. (Center) Thanks to Parseval's theorem (Parseval, 1806), the distances between vectors do not change after Fourier transformation, hence the picture is still valid in the Fourier domain. (Right) However, in the Fourier domain it will be possible to alter the filters so that the $N$ points approach the target by making the reservoir resonate at appropriate frequencies, effectively reducing the forecasting error.

where $E_{L,s}$ is the number of edges in the reservoir network and $E_{L,s}$ (with $s\in\{+,-\}$) represents the number of edges embedded in cycles of length $L$ and sign $s = +$ or $-$, respectively. The value of $E$, the number of edges, depends on the specific ESN implementation (see Supplemental Information Section II). Note that standard ESNs (Jaeger and Haas, 2004; Pathak et al., 2018; Schrauwen et al., 2007) typically use fully random reservoirs, rendering $E_{L,s}-E_{L,-s}\sim 0$ regardless of the values of $E$, and hence $\rho_L\sim 0$ for all $L$.

As shown in Figure 5, a reservoir with different cycle lengths $L$ and different $\rho_L$ values as generated by an algorithm presented in Supplemental Information Section IV can enhance different families of frequencies. This holds true even though the dynamics of the neuron are nonlinear (see Supplemental Information Section X for an analytical explanation). For example, the Mackey-Glass time series and Spoken Arabic Digit time series, are dominated by low frequencies. Any reservoir with $\rho_L > 0$ will enhance the low frequencies, meaning that such a reservoir would enhance the frequencies relevant to those two time series. Similarly, for Laser Intensity, there are three peaks in the center of its PSD, which are enhanced in the cases of $\rho < 0$ for $L = 2$ and $L = 3$, but not for $L = 1$. This implies a strategy to tailor reservoir for any specific task.

To prove the concept of tailoring reservoir for specific tasks, we consider reservoirs with varying fractions of cycles and plot the ESN performance as a function of $\rho_L$ for $L = 1,2,3$. As shown in Figure 6, the ESN performance does change with $\rho_L$ for each $L$. To better understand this phenomenon, it is useful to consider the optimal $\rho_L$ value (highlighted in pink) and compare the corresponding average PSD of the neuron states for this reservoir with the PSD of the empirical time series.

For the Mackey-Glass (or the Spoken Arabic Digits) time series, the optimal $\rho_L$ is positive for $L = 1,2,3$ (see Figures 6A, 6D, and 6G; or 6C, 6F, and 6I). We also know that for $\rho_L>0$ (especially for $L = 2,3$), the reservoir's average PSD response is enhanced for the frequencies close to 0, which is exactly the regime where the spectrum of the Mackey-Glass (or the Spoken Arabic Digits) time series is concentrated (see Figures 5A, 5D, and 5G; or 5C, 5F, and 5I).

As for the Laser Intensity time series, the dominating frequencies of its PSD are around 0.13, 0.27, and 0.38, thus ESN is improved when the response of the reservoir enhances those frequencies. As shown in Figures 5E and 5H, this happens when $\rho_L<0$ for $L = 2,3$. Indeed, as shown in Figures 6E and 6H, negative $\rho_L$ (for $L = 2,3$) improves the ESN performance. For $L = 1$, we observed in Figure 5 that the three peaks cannot be all enhanced simultaneously by setting $\rho_1$ to be either positive or negative. Instead, setting $\rho_1 = 0$ would yield the optimal performance. This is exactly what we observed in Figure 6B.

Results shown in Figures 5 and 6 indicate that the reservoir should be tailored to resonate with the dominating frequencies present in the target signal. To achieve that, we designed a simple heuristic algorithm to find the optimal values of $\rho_L$ for cycles of different lengths (see Supplemental Information Section XI). This

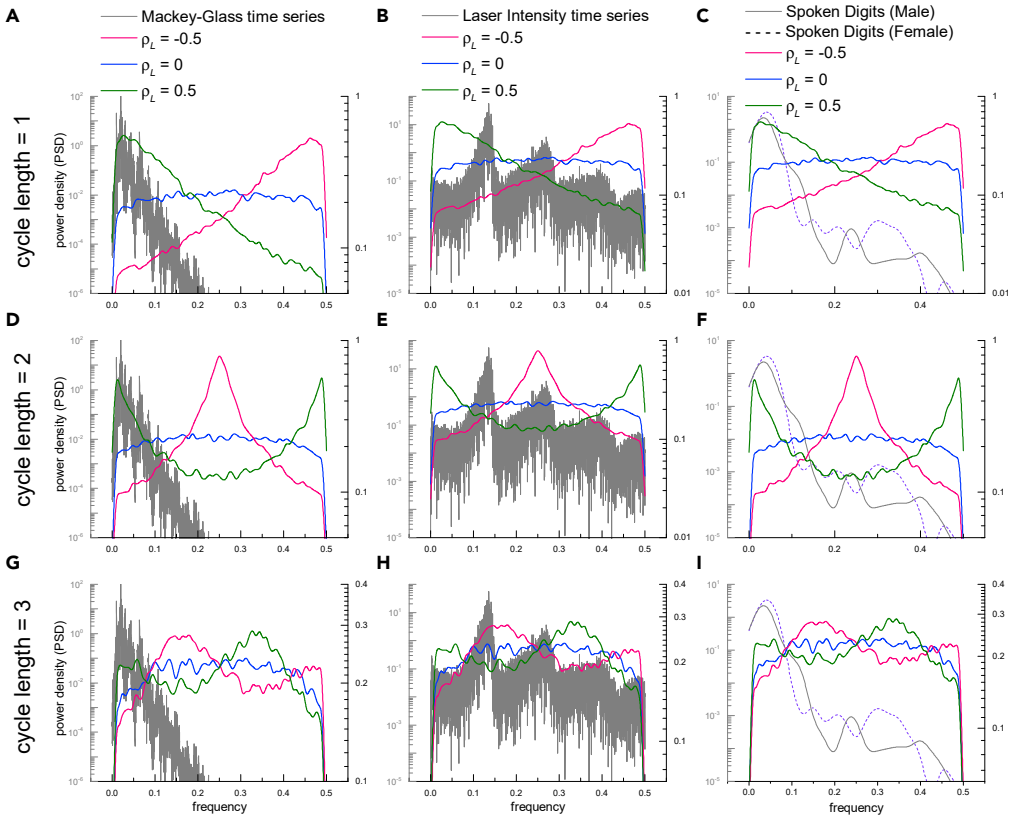



**Figure 5. Frequency Domain Analysis of Target Signals and Reservoir Frequencies**

(A–I) Left y axis: the power spectral density (PSD) of three empirical time series (Mackey-Glass in A, D, and G; Laser Intensity in B, E, and H; and Spoken Digits in C, F, and I). Right y axis: the average PSD of the neuron states for reservoirs with various $\rho_L$ when using a random Gaussian input from $\mathcal{N}(0,1)$. In each panel we plot the average PSD of 500 reservoirs with 400 neurons and connectivity 0.05. The length of cycles added into the reservoir is 1 (A–C), 2 (D–F), and 3 (G–I).

heuristic algorithm does offer much better ESN performance (dotted lines in Figure 6) than simply optimizing cycles of a fixed length $L$, and is definitely better than standard, state-of-the-art random reservoirs with random weights, where $\rho_L = 0$ (Jaeger and Haas, 2004; Pathak et al., 2018; Schrauwen et al., 2007).

## DISCUSSION

In this work, we start by showing how the correlations between neurons define the memory of ESN and demonstrate that those correlations are determined by the eigenvalues of the reservoir's adjacency matrix. This result allows us to easily assess the memory capacity of a particular reservoir network, unifying previous results (Farkaš et al., 2016; Jaeger, 2001a, 2001b; Rodan and Tiňo, 2012; Strauss et al., 2012). Then we go beyond the current ESN practice and reveal previously unexplored optimization strategies. In particular, we show that adding short loops to the reservoir network can create resonant frequencies and enhance ESN performance by adapting the reservoir to specific tasks. It is important to note that we are not advocating the hand-tuning of reservoir topologies for specific tasks of ESN, but rather raising the point that notions from classical signal processing can help us understand and improve recurrent neural networks, either through selection of appropriate initial topologies in a pre-training stage or by designing learning algorithms that account for the principles outlined here. Given that most current learning strategies such as back-propagation focus on adapting single weights, we are convinced that many new learning algorithms can be created by focusing on network-level features. Moreover, our approach goes beyond improving current techniques. By studying which properties of a recurrent neural network make it well suited for a particular problem, we are also addressing the converse question of how should a neural network be after it has been adapted to a specific task. Thus, we provide valuable insights into the training

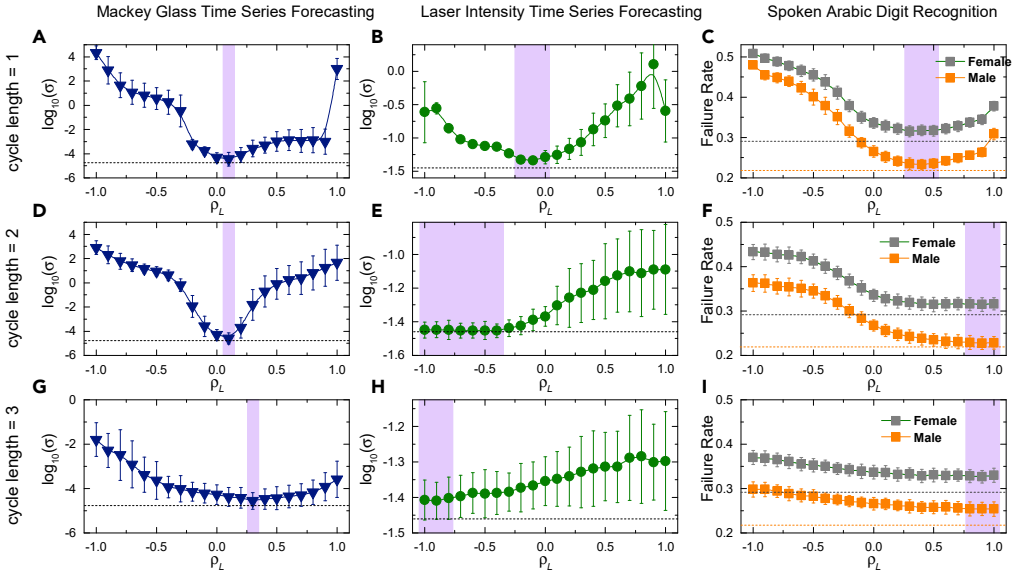



**Figure 6. Tailoring ESN through Frequency Adaptation**

(A–I) ESN performance as a function of $\rho_L$, for Mackey-Glass Forecasting (A, D, and G), Laser Intensity Forecasting (B, E, and H), and Spoken Arabic Digit Recognition (C, F, and I). Every point corresponds to the median performance, measured by forecasting error in (A, B, D, E, G, and H) and failure rate in (C, F, and I), over 200 realizations with error bars representing upper and lower quartiles. The length of cycles added into the reservoir is $L = 1$ in (A–C), 2 in (D–F), and 3 in (G–I). For each $L$, the optimal $\rho_L$ values are highlighted in pink. Dotted lines represent the best ESN performance obtained by combining cycles of different lengths.

process of general recurrent neural networks, as our theory highlights structural features that the training process would enhance or inhibit.

## Limitations of the Study

Finally, we would like to highlight some potential caveats of the current work. On the application side, although we demonstrate that adding short loops to the reservoir network can improve the ESN performance for the aforementioned tasks, ESN as a whole can be outperformed by other, more task-specialized approaches that often require larger training dataset and longer training time. For instance, with a standard MATLAB package we can obtain performances of $\log(\sigma) \sim -6$for Mackey-Glass Time Series Forecasting, although the algorithm takes much longer to train and requires more data. An *ad hoc* method for the Laser Intensity task with linear Finite Impulse Response Filters and a long memory buffer achieved performance in the interval $\log(\sigma) = [-1.7, -3]$ depending on the time interval (Wan, 1993). The Spoken Arabic Digit Recognition can be solved with a failure rate between 2.5% and 12% by using graphical models (Hammami and Sellam, 2009), although in the graphical models they used all the 13 Mel Frequency Cepstral Coefficient (MFCC) channels as opposed to the single channel that we used in this work. The reason why we focus on ESN in this work is just because of its simplicity, which allows us to perform analytical calculations. On the theoretical side, there are two caveats. First, the heuristic we used to find the optimal values of $\rho_L$ presented in Figure 6 and Supplemental Information Section XI does not have theoretical guarantees. We could create signals where the dominant frequency does not contain any relevant information, for instance, by adding a strong sinusoid to the time series to be processed. In this sense, we are counting on the domain knowledge of researchers who are interested in our method to filter out large non-informative components before feeding the time series to the ESN. Second, the relationship between eigenvalue moduli and neuron correlations assumes that linearization is a valid approximation. Although it is useful as an upper bound, the bound might become loose when the nonlinearity effects are strong. This could in principle be addressed by studying the Lyapunov spectra of the reservoir, which depends partially on the network structure and partially on the dynamics, accounting thus for the nonlinearities. A rigorous mathematical analysis is likely to be very challenging and is beyond the scope of the current study.

## Resource Availability

### Lead Contact

Further information and requests should be addressed to Yang-Yu Liu (yyl@channing.harvard.edu).

### Materials Availability

This study did not generate any new reagents.

### Data and Code Availability

The data and code used in this work are available at: https://github.com/pvili/EchoStateNetworks_NetworkAdaptation/tree/master.

## METHODS

All methods can be found in the accompanying Transparent Methods supplemental file.

## SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.isci.2020.101440.

## ACKNOWLEDGMENTS

We thank Professor Herbert Jäger and Benjamin Liebald for valuable discussions. This work was partially supported by the "la Caixa" Foundation and grants from the John Templeton Foundation (Award No. 51977) and National Institutes of Health (R01AI141529, R01HD093761, UH3OD023268, U19AI095219, and U01HL089856).

## AUTHOR CONTRIBUTIONS

Y.-Y.L. conceived and designed the project. P.V.A. performed all the analytical calculations and empirical data analysis. P.V.A. and G.Y. performed extensive numerical simulations. All authors analyzed the results. P.V.A. and Y.-Y.L. wrote the manuscript. G.Y. edited the manuscript.

## DECLARATION OF INTERESTS

The authors declare that they have no competing financial interests.

Received: May 14, 2020
Revised: July 17, 2020
Accepted: August 3, 2020
Published: September 25, 2020

## REFERENCES

Aceituno, P.V., Rogers, T., Schomerus, H., et al. (2019). Universal hypotrochoidic law for random matrices with cyclic correlations. Physical Review E. *100*, 010302, In press. https://doi.org/10.1103/PhysRevE.100.010302.

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D.-U. (2006). Complex networks: structure and dynamics. Phys. Rep. *424*, 175–308.

Boedecker, J., Obst, O., Lizier, J.T., Mayer, N.M., and Asada, M. (2012). Information processing in echo state networks at the edge of chaos. Theor. Biosci. *131*, 205–213.

Boedecker, J., Obst, O., Mayer, N.M., and Asada, M. (2009). Studies on reservoir initialization and dynamics shaping in echo state networks. In European Symposium on Artificial Neural Networks (Bruges (Belgium): European neural network society), ISBN 2-930307-09-9. https://www.elen.ucl.ac.be/esann/proceedings/papers.php?ann=2009 In press.

Buehner, M., and Young, P. (2006). A tighter bound for the echo state property. IEEE Trans. Neural Network *17*, 820–824.

Buteneers, P., Schrauwen, B., Verstraeten, D. and Stroobandt, D. (2008), Real-time epileptic seizure detection on intra-cranial rat data using reservoir computing, in International Conference on Neural Information Processing, Springer, pp. 56–63.

Coulibaly, P. (2010). Reservoir computing approach to great lakes water level forecasting. J. Hydrol. *381*, 76–88.

Cui, H., Liu, X., and Li, L. (2012). The architecture of dynamic reservoir in the echo state network, *Chaos*. Interdiscip. J. Nonlinear Sci. *22*, 033127.

Deihimi, A., and Showkati, H. (2012). Application of echo state networks in short-term electric load forecasting. Energy *39*, 327–340.

Deng, Z. and Zhang, Y. (2006), Complex systems modeling using scale-free highly-clustered echo state network, in International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 3128–3135.

Farkaš, I., Bosák, R., and Gergel, P. (2016). Computational analysis of memory capacity in echo state networks. Neural Networks *83*, 109–120.

Ferreira, A.A. and Ludermir, T.B. (2011), Comparing evolutionary methods for reservoir computing pre-training, in International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 283–290.

Gandhi, M., Tiňo, P. and Jaeger, H. (2012), Theory of input driven dynamical systems, in European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 25–27.

Hammami, N. and Bedda, M. (2010), Improved tree model for arabic speech recognition, in International Conference on Computer Science and Information Technology (ICCSIT), Vol. 5, IEEE, pp. 521–526.

Hammami, N. and Sellam, M. (2009), Tree distribution classifier for automatic spoken arabic digit recognition, in 2009 International Conference for Internet Technology and Secured Transactions,(ICITST), IEEE, pp. 1–4.

Hübner, U., Abraham, N., and Weiss, C. (1989). Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared nh 3 laser. Phys. Rev. A *40*, 6354.

Jaeger, H. (2001a). Short Term Memory in Echo State Networks (GMD-Forschungszentrum Informationstechnik).

Jaeger, H. (2001b). The "Echo State" Approach to Analysing and Training Recurrent Neural Networks -with an Erratum Note*148* (German National Research Center for Information Technology GMD), p. 34, Technical Report.

Jaeger, H. (2002). Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the "Echo State Network" Approach (GMD-Forschungszentrum Informationstechnik).

Jaeger, H. (2005). Reservoir riddles: suggestions for echo state network research. Proc. Int. Joint Conf. Neural Networks *3*, 1460–1462.

Jaeger, H. (2007). Discovering Multiscale Dynamical Features with Hierarchical Echo State Networks (Jacobs University Bremen), Technical Reports.

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

Jaeger, H., Lukoševičius, M., Popovici, D., and Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. Neural Networks *20*, 335–352.

Jiang, F., Berry, H., and Schoenauer, M. (2008). Supervised and evolutionary learning of echo state networks. Parallel Problem Solving from Nature–PPSN X (Springer), pp. 215–224.

Lichman, M. (2013). UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

Liebald, B. (2004). Exploration of Effects of Different Network Topologies on the ESN Signal Crosscorrelation Matrix Spectrum, PhD Thesis (University Bremen).

Lin, X., Yang, Z., and Song, Y. (2009). Short-term stock price prediction based on echo state networks. Expert Syst. Appl. *36*, 7313–7317.

Lukoševicius, M., and Jaeger, H. (2007). Overview of Reservoir Recipes (Technical Reports: Jacobs University Bremen).

Lukoševičius, M., and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. *3*, 127–149.

Mackey, M.C., and Glass, L. (1977). Oscillation and chaos in physiological control systems. Science *197*, 287–289.

Newman, M.E., and Girvan, M. (2004). Finding and evaluating community structure in networks. Phys. Rev. E *69*, 026113.

Newton, M.J., and Smith, L.S. (2012). A neurally inspired musical instrument classification system based upon the sound onset. J. Acoust. Soc. Am. *131*, 4785–4798.

Ozturk, M.C., Xu, D., and Príncipe, J.C. (2007). Analysis and design of echo state networks. Neural Comput. *19*, 111–138.

Parseval, M.-A. (1806). Mémoire sur les Séries et sur lâ€™Intégration Complète dâ€™une Équation Aux Différences Partielles Linéaires du Second Ordre, à Coefficients Constants, *1* (Mémoires Présentés Par Divers. Savants, Academie Des Sciences), pp. 638–648.

Pathak, J., Hunt, B., Girvan, M., Lu, Z., and Ott, E. (2018). Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. Phys. Rev. Lett. *120*, 024102.

Plöger, P.G., Arghir, A., Günther, T., and Hosseiny, R. (2003). Echo state networks for mobile robot modeling and control. In Robot Soccer World Cup, D. Polani, B. Browning, A. Bonarini, and K. Yoshida, eds. (Springer), pp. 157–168.

Rodan, A., and Tiňo, P. (2012). Simple deterministically constructed cycle reservoirs with regular jumps. Neural Comput. *24*, 1822–1852.

Rodriguez, N., Izquierdo, E., and Ahn, Y.-Y. (2019). Optimal modularity and memory capacity of neural reservoirs. Netw. Neurosci. *3*, 551–566.

Schrauwen, B., Verstraeten, D. and Van Campenhout, J. (2007), An overview of reservoir computing: theory, applications and implementations, in Proceedings of the European Symposium on Artificial Neural Networks, pp. 471–482.

Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J.J., and Stroobandt, D. (2008). Improving reservoirs using intrinsic plasticity. Neurocomputing *71*, 1159–1171.

Strauss, T., Wustlich, W., and Labahn, R. (2012). Design strategies for weight matrices of echo state networks. Neural Comput. *24*, 3246–3276.

Tong, M.H., Bickett, A.D., Christiansen, E.M., and Cottrell, G.W. (2007). Learning grammatical structure with echo state networks. Neural Networks *20*, 424–432.

Verplancke, T., Looy, S., Steurbaut, K., Benoit, D., Turck, F., Moor, G., and Decruyenaere, J. (2010). A novel time series analysis approach for prediction of dialysis in critically ill patients using echo-state networks. BMC Med. Inform. Decis. Making *10*, 1.

Wan, E.A. (1993). Time series prediction by using a connectionist network with internal delay lines. Santa Fe Institute Studies in the Sciences of Complexity-Proceedings, *Vol. 15* (Addison-Wesley Publishing Co), p. 195.

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

**iScience, Volume *23***

## Supplemental Information

# Tailoring Echo State Networks for Optimal Learning

**Pau Vilimelis Aceituno, Gang Yan, and Yang-Yu Liu**

CONTENTS

# Part I
# Transparent Methods

## I. AN INTRODUCTION OF ESN

Echo state network (ESN) is a promising paradigm of recurrent neural networks (RNNs) that can be used to model and predict the temporal behavior of nonlinear dynamic systems Jaeger and Haas (2004). As a special form of RNNs, ESN has feedback loops in the randomly assigned and fixed synaptic connections and trains only a linear combination of the neurons' states. This fundamentally differs from the traditional feed-forward neural networks, which have multiple layers but no cycles Christopher (2006) and simplifies other RNN architectures that suffer from the difficulty in training synaptic connections Pascanu *et al.* (2013).

An ESN can be viewed as a dynamic system from which the information of input signals is extracted Dambre *et al.* (2012). Mathematically, the discrete-time dynamics of the simplest ESN with $N$ neurons $x_i(t)$, one input $u(t)$ and one output $y(t)$ is given by

$$\mathbf{x}(t) = f(\mathbf{W}\,\mathbf{x}(t-1) + \mathbf{w}_{\text{in}}\,u(t) + \mathbf{w}_{\text{ofb}}\,y(t-1)), \tag{S1}$$

$$y(t) = \mathbf{w}_{\text{out}}\,[\mathbf{x}(t), u(t)]^{\top}. \tag{S2}$$

The nonlinear function $f$ can have different forms, e.g., the logistic sigmoid or the hyperbolic tangent Christopher (2006). The matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ is the *fixed* weighted adjacency matrix of the reservoir network. The vector $\mathbf{w}_{\text{in}} \in \mathbb{R}^{N}$ captures the *fixed* weights of the input connections. The vector $\mathbf{w}_{\text{ofb}} \in \mathbb{R}^{N}$ denotes the *fixed* weights of the feedback connections from the output to the $N$ neurons. Finally, the row vector $\mathbf{w}_{\text{out}} \in \mathbb{R}^{1\times(N+1)}$ represents the *trainable* weights of the readout connections from the $N$ neurons and the input to the output.

It has been shown that the information processing capacity of a dynamic system, in theory, depends only on the number of linearly independent variables or, in our case, neurons Dambre *et al.* (2012); Duport *et al.* (2012); Maass *et al.* (2002). Yet, the theoretical capacity does not imply that all implementations are practical Whitley and Watson (2005); Büsing *et al.* (2010), nor does it mean that any reservoir is equally desirable for a given task. A clear example is the effect of the reservoir's spectral radius (i.e., the largest eigenvalue in modulus): an ESN with a larger spectral radius has longer-lasting memory, indicating that it can better process information from past inputs Jaeger (2002).

### A. Basic schema

Note that there is a rich literature on methods to improve the ESN performance such as using regularization in the computation of $\mathbf{w}^{*}_{\text{out}}$ Jaeger (2002), controlling the input weights Strauss *et al.* (2012) or changing the dynamics of the neurons Lukoševičius and Jaeger (2009). Those results, while relevant and important for applications, are tangential to our study. Therefore in this work we will use the simplest version of the ESN as depicted in Fig.1 of the main text.

### B. Training the readout

The training of an ESN aims to find the output weights of each neuron state such that the output $y(t)$ can best approximate the target variable $\hat{y}(t)$ Jaeger (2001a). Here the output is a linear combination of the neurons' states $\mathbf{x}(t)$ and the input $u(t)$, as shown in Eq.S2.

This aim can be achieved by minimizing the squared training errors $\sum_{t=1}^{T}(\hat{y}(t) - y(t))^2$. Hence it becomes a classical linear regression problem: Given $T$ vectors $\mathbf{x}(t)$ of dimension $N$ and the target variable $\hat{y}(t)$, calculate the vector $\mathbf{w}_{\text{out}}$ that satisfies

$$\mathbf{w}_{\text{out}} = \arg\min_{\mathbf{w}_{\text{out}}} \sum_{t=0}^{T} \left( \hat{y}(t) - \mathbf{w}_{\text{out}} \begin{pmatrix} \mathbf{x}(t) \\ u(t) \end{pmatrix} \right)^2.$$

We rewrite Eq.S2 as a matrix equation $Y = \mathbf{w}_{\text{out}} \cdot X$, where $Y$ is the column vector containing all $y(t)$ for $t = 1, ..., T$ values and $X$ a matrix where each row contains the values of the corresponding vector $\begin{pmatrix} \mathbf{x}(t) \\ u(t) \end{pmatrix}$. Thus, $\mathbf{w}_{\text{out}}$ can be

solved through the Moore-Penrose pseudo-inverse $X^+ = X^* \left(XX^*\right)^{-1}$ where $X^*$ is the Hermitian transpose (also known as conjugate transpose of $X$). In the case of real matrices, the Hermitian transpose is just the transpose. Thus we can write $X^+ = X^\top \left(XX^\top\right)^{-1}$, and

$$\mathbf{w}_{\text{out}} = Y \cdot X^+.$$

The calculation of $X^+$ is implemented with the command pinv in Matlab.

Depending on the task Jaeger (2002), the output $y(t)$ can be fed back to the reservoir through $\mathbf{w}_{\text{ofb}}$. During the training phase, we do not have the actual $y(t)$, since $\mathbf{w}_{\text{ofb}}$ has not been trained. If the reservoir's output is expected to be close to the target variable, we can instead feed the target variable $\hat{y}(t+1)$. To have a valid comparison with the original benchmark problem Jaeger and Haas (2004), we use this process for the task of forecasting Mackey-Glass time series, and for the rest of tasks the values of $\mathbf{w}_{\text{ofb}}$ are set to be 0.

### C. Selecting reservoir parameters

Besides the training of the readout, the reservoir has other parameters that must be optimized. Typically, those are the scaling of $\mathbf{w}^{\text{in}}$, $\mathbf{w}_{\text{ofb}}$ and $\mathbf{W}$. In this work, however, we take a very simple version of ESN where $\mathbf{w}^{\text{in}}$ and $\mathbf{w}_{\text{ofb}}$ are fixed and only the scaling of $\mathbf{W}$ is trained. This is often trained through the spectral radius $|\lambda_{\max}|$, which grows linearly with the weights of $\mathbf{W}$ – just as our metric $\langle|\lambda|\rangle$ does.

In the examples used in this work we trained this in the simplest possible way: by trying a range of scalings – from a spectral radius of $0.2$ to $1$ with intervals of $0.05$. This is done with classical Erdős-Rényi reservoirs, and in the last experiments, we evaluated the $\langle|\lambda|\rangle$. By noticing that the Erdős-Rényi reservoirs have a uniform eigenvalue distribution within a circle in the complex plane centered at zero and with radius $|\lambda_{\max}|$, then we only need to integrate the radius over the uniform distribution and divide by the area,

$$\langle|\lambda|\rangle = \frac{1}{\pi|\lambda_{\max}|^2} \int_0^{|\lambda_{\max}|} r 2\pi r dr = \frac{2}{3}\lambda_{\max}. \tag{S3}$$

### D. Performance Measurement

In the literature of ESN it is common to forecast time series Jaeger and Haas (2004). To be consistent with the previous literature we use the normalized root mean squared error (NRMSE), as a metric of forecasting error

$$\sigma = \sqrt{\frac{\sum_{t=t_0}^{t_0+T}(y(t) - \hat{y}(t))^2}{T \cdot \text{var}(\hat{y}(t))}}. \tag{S4}$$

This metric is a normalization of the classical root mean squared error. The parameter $t_0$ is used to describe when we start to count the performance, since it is also common to ignore the inputs during the initialization phase Jaeger (2002), which is taken here as the full initialization steps given for each task (see details in the subsequent sections). The parameter $T$ is simply the number of time-steps considered, which we take here as the full count of all points except the initialization phase in each testing time series.

The NRMSE is obviously not a good metric for classification tasks where the target variable is discrete. In order to have a comparable metric for ESN performance, we use the failure rate in classification tasks such as the Spoken Arabic Digit Recognition. Note that having 10 digits implies that the failure rate with random guesses is 0.9, therefore a failure rate of 0.3 is well below it.

## II. Benchmark Tasks

### A. Forecasting Mackey-Glass time series

Forecasting Mackey-Glass time series is a benchmark task to test the performance of ESN Jaeger and Haas (2004). The Mackey-Glass time series follows the ordinary differential equation Jaeger and Haas (2004):

$$\frac{ds(t)}{dt} = \beta \frac{s(t-\tau)}{1+s(t-\tau)^n} - \gamma s(t),$$

where $\beta$, $\gamma$, $\tau$, $n$ are real positive numbers. We used the parameters $\beta = 0.2$, $\gamma = 0.1$, $\tau = 17$, $n = 10$ in our simulations. The discrete version of the equation uses a time step of length $h = 0.1$. For each time series we generated $\frac{\tau}{h} = 170$ uniformly distributed random values between $1.1$ and $1.3$ and then followed the equations. The first 1000 points were considered as initialization steps, which did not fully capture the time series dynamics and were thus discarded. For training and testing we used time series of 10,000 points, but in both cases the first 1000 states of the reservoir were considered as initialization steps and were thus ignored for training and testing. For an ESN with 1000 neurons and an optimized memory, the forecasting performance for this setting is close to its maximum value, thus the addition of short cycles will have a small effect. In order to show the interest of our contribution, we normalized the signal to have mean zero and variance of one and we added Gaussian white noise with $\sigma = 0.05$, and the forecasting was done using reservoirs of 1000 neurons, average degree $\langle k \rangle = 20$, hence $E = 20000$ and spectral radius of $\alpha = 0.85$, and the output was feed back to the reservoir through the vector $\mathbf{w}_{\text{ofb}}$ where every entry is independently drawn from a uniform distribution on the interval $[-1, 1]$. The ESN was trained to forecast one time-step, and then we used this readout to forecast 84 time-steps in the future by recursively feeding the one-step prediction of $s(t+1)$ into the ESN as the new input.

### B. Forecasting Laser Intensity time series

The Laser Intensity time series Hübner *et al.* (1989); Huebner *et al.* (1989) was obtained from the Santa Fe Institute time series Forecasting Competition Data. It consists of 10,093 points, which we normalized to have an average of zero and an standard deviation of one, and were filtered with a Gaussian filter of length three and standard deviation of one. The forecasting was done using reservoirs of 100 neurons, average degree $\langle k \rangle = 10$, hence $E = 1000$ and spectral radius of $\alpha = 0.9$, without feedback so $\mathbf{w}_{\text{ofb}} = 0$. Here we forecasted one time-step. We used 1,000 points of the time series for initialization, 4,547 for training and 4,546 for testing.

### C. Spoken Arabic Digit Recognition

The Spoken Arabic Digits Hammami and Bedda (2010) dataset was downloaded from the Hammami and Bedda (2010) from the UCI Machine Learning Repository Lichman (2013). This dataset consists of 660 recordings (330 from men and 330 from women) for each of the ten digits and 110 recordings for testing. Each recording is a time series of varying length encoded with MCCF Mermelstein (1976) with 13 channels. While using the first three channels gave a better performance, here we use only the first channel, which is akin to a very lossy compression. We normalized this time series to have average of zero and a standard deviation of one, and a length of 40. Since in most cases we had less than 40 points, we computed the missing values by interpolation. The classification procedure was done using the forecasting framework. We collected the reservoir states from all the training examples of each digit and computed $\mathbf{w}^{\text{out}}$ as did in the previous forecasting tasks. In the testing we collected the states and computed the forecasting performance $\sigma$ for each of the 20 cases. We classified the time series as the digit that yielded the lowest forecasting error. Then we calculate the failure rate as the number of misclassified recordings divided by the total number of recordings in the training set. We used reservoirs of 100 neurons, average degree $\langle k \rangle = 10$, hence $E = 1000$ and spectral radius of $\alpha = 1$, without feedback ($\mathbf{w}_{\text{ofb}} = 0$).

## III. Network Generation

In this paper we systematically studied the impact of network topology on the performance of ESN. In this section we described the methods of generating various model networks with different topological properties.

### A. Scale-free networks

In scale-free (SF) networks the probability distribution of node degrees follows a power law, i.e., $P(k) \sim k^{-\gamma}$, and the exponent $\gamma$ usually varies between 2 and 3.

In this paper we adopted the so-called static model Goh *et al.* (2001), to generate scale-free networks with tunable degree exponent $\gamma$ and mean degree $\langle k \rangle$. Since we use directed networks, we have outgoing and incoming edges. For simplicity we use $\gamma_{in} = \gamma_{out}$, meaning that for every node the expected outgoing and incoming degrees are the same. The static model can be described as follows:

**Step-1**: We start with $N$ isolated nodes, labeled from $1$ to $N$. Each node is assigned a weight $w_i \sim i^{-a}$, where $a = 1/(\gamma - 1)$.

**Step-2**: We independently pick up two nodes according to their assigned weights, and add a link between these two nodes if they have not been connected before. Self-links and double-links are forbidden.

**Step-3**: Repeat Step-2 until $M = \langle k \rangle N/2$ links have been added into the network.

### B. Random regular networks

In a random regular (RR) network, all nodes have the same degree and the edges randomly connect node pairs. Random regular networks can be generated by rewiring, as described by the following algorithm, which follows the same logic as Wormald's algorithm Wormald (1984) but is designed for directed networks:

The algorithm takes the number of nodes $N$ and the connectivity $c$ as parameters.

**Step-1**: Create a list $L_1$ where, for each of the $N$ nodes, there are $cN/2$ outgoing edges.

**Step-2**: Copy the previous list and make a random permutation, creating list $L_2$. The $n$th edge is the edge that goes from the node $L_1(n)$ to the node $L_2(n)$.

**Step-3**: If there are repeated edges, randomly swap the destinations, until there are no more repeated edges.

**Step-4**: If Step-3 was repeated $M$ times, jump to Step-2.

The limited number of iterations given by step 4 was set to avoid infinite iterations. In this work we set $M = 500$.

For each node there are $k$ outgoing links and the $k$ links can go to $k$ of the $N$ nodes. Thus we have $\frac{N!}{(N-k)!}$ combinations of destinations that do not include a repetition against $N^k$ possible combinations, yielding a probability of $p_r = 1 - \frac{N!}{N^k(N-k-1)!}$ of repeating an edge. Thus on the first iteration we will have $p_r cN^2$ repetitions, on the second one $p_r^2 cN^2$ and therefore the complexity is $O(\frac{1}{1-p_r} cN^2)$.

### C. Erdős-Rényi networks

In an Erdős-Rényi (ER) random network, each pair of nodes is connected with probability $p$. We can use different probability distributions to assign link weights in the ER random networks, including binary, uniform, normal and power law distributions. Networks used in ESN typically have no preference for positive or negative weights, and we modified the link weight distributions to keep the mean zero.

- For binary distribution, each link was assigned a weight $-a$ or $a$, with $a > 0$.
- For uniform distribution, the link weights were randomly drawn from the interval $[-a, a]$ where $a > 0$.
- For normal distribution, the link weights were drawn from a normal distribution with zero mean and standard deviation $a$.
- For the power law (PL) distribution we use the inverse transform sampling method to convert a uniform distribution in $[0, 1]$ into a power law Deák (1990). The power law distribution provides only positive numbers, thus we randomly inversed the sign of each link with probability $0.5$.

### D. Spectral radius and the variance of the weight distribution

In the main text and in the previous network generation algorithms we do not mention the variance of the probability distribution from which we draw the weights. Here we show that the variance of the link weight distribution is actually irrelevant.

The networks referred in the main text have an adjacency matrix $W = (W)_{ij}$ the weights are drawn from a normal distribution with zero mean and some variance $v$. As we consider the case where the dimension of the adjacency matrix is very high, the variance converges to its expected value. We recall the definition of the variance,

$$v = \mathsf{Var}\left[W_{ij}\right] = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}^2.$$

If we multiply the matrix by a scalar $a \geq 0$, when the variance becomes

$$\mathsf{Var}\left[aW_{ij}\right] = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} a^2 W_{ij}^2 = a^2 v. \tag{S5}$$

Note that the entries of $aW$ are still drawn from a normal distribution, it is simply that the variance has changed. Consider now the spectral radius $\alpha$ of $W$,

$$\alpha = \max_{\mathbf{v}} \frac{\|W\mathbf{v}\|}{\|\mathbf{v}\|}$$

where $\|\cdot\|$ is the Euclidean norm. If we scale the matrix by $a$,

$$\max_{\mathbf{v}} \frac{\|aW\mathbf{v}\|}{\|\mathbf{v}\|} = \max_{\mathbf{v}} a\frac{\|W\mathbf{v}\|}{\|\mathbf{v}\|} = a\alpha. \tag{S6}$$

Putting together Eqs.S5 and S6, we see that both values can be set though $\alpha$, meaning that there is a one-to-one correspondence between variance and spectral radius. Therefore, when we fix the spectral radius, we also fix the variance. Thus, the original variance of the distribution is irrelevant.

## IV. GENERATING RESERVOIRS WITH CYCLES

In order to generate networks with desired $\rho_L = \frac{E_{L,s} - E_{L,-s}}{E}$, we designed the following algorithm, which takes as parameters the number of neurons $N$; the connectivity $c$; $|\rho_L| \in [0, 1]$, which is the portion of edges that are dedicated to cycles of length $L$ –the other ones being random–; and $s \in \{-1, 1\}$, which corresponds to the feedback sign.

**If $L = 1$:** Create a random sparse matrix $\mathbf{W}_\mathsf{r}$ with $cN(N-1)$ non-zero entries. Normalize the spectral radius to 1 and then $\mathbf{W} \leftarrow \alpha\left((1 - |\rho_1|)\mathbf{W}_\mathsf{r} + sr_1 I\right)$, where $I$ is the identity matrix.

**Else:**

**Step-1:** Create $\frac{|\rho_L| cN^2}{2L}$ permutations of $L$ numbers randomly picked from 1 to $N$ without replacement. Each permutation corresponds to $L$ nodes that will be connected form a cycle.

**Step-2:** For each cycle, draw the edge weights from $\mathcal{N}(0, 1)$.

**Step-3:** For each cycle, if the sign of the product of the edge weights is not the same as $s$, multiply the last edge by $-1$. This gives the adjacency matrix $\mathbf{W}_\mathsf{c}$.

**Step-4:** Create a random sparse matrix $\mathbf{W}_\mathsf{r}$ with $\frac{(1-|\rho_L|)cN^2}{2}$ entries and weights drawn from $\mathcal{N}(0, 1)$.

**Step-5:** If the edges do not overlap in $\mathbf{W}_\mathsf{r}$ and $\mathbf{W}_\mathsf{c}$, $\mathbf{W} = (\mathbf{W}_\mathsf{r} + \mathbf{W}_\mathsf{c})$. Otherwise, move the edge from $\mathbf{W}_\mathsf{r}$ into a new random node pair.

**Step-6:** Normalize to the desired average eigenvalue moduli $\langle|\lambda|\rangle$ by $\mathbf{W} \leftarrow \frac{\mathbf{W}}{\frac{1}{N}\sum_{i=1}^{N}|\lambda_i(\mathbf{W})|}\langle|\lambda|\rangle$, where $\lambda_i(\mathbf{W})$ are the eigenvalues of $\mathbf{W}$

The special treatment of the case $L = 1$ is due to the fact that with length of 1 if all edges are self loops the network is completely disconnected and the number of edges is at most $N$, meaning that for some values of $\rho_1$ the number of edges would be lower than the number required by the connectivity parameter.

## V. NUMERICAL STUDY OF EIGENVALUE DENSITY AND MEMORY CAPACITY

One of the motivations of our study on Memory Capacity is that the spectral radius is not enough to capture how $M$ changes, particularly for different network structures. In Fig.S1, we show that for some families of networks the memory is also affected by other parameters.

Specifically, we study the following architectures:

- Erdös-Rény (ER) random graphs with weights drawn from a Gaussian distribution and varying spectral radii.
- Erdös-Rény random graphs with weights drawn from a power law distribution (PL) with $\beta \in [2, 5]$ but normalized to have a spectral radius $\alpha = 1$.
- Scale-Free (SF) networks where the degree heterogeneity is given by the degree exponent $\gamma \in [2, 6]$, and the weights are drawn from a Gaussian distribution, also normalized to have a spectral radius $\alpha = 1$.
- Random Regular (RR) graphs with varying degrees and a spectral radius $\alpha = 1$.

The results from Fig.1 can be contrasted with the eigenvalue densities of the aforementioned network families presented in Fig.2. We observe that the networks where the eigenvalues are concentrated in the center, either through the spectral radius $\alpha$, the power law exponent $\beta$ or the degree heterogeneity $\gamma$, have lower memory, while those with eigenvalues uniformly spread have more memory.

## VI. MEMORY CAPACITY AND THE CORRELATIONS BETWEEN NEURONS

Here we derive an upper bound which connects the variance of the reservoir across different directions with the memory capacity. The gist of our argument goes as follows: the memory capacity reflects the precision with which previous inputs can be recovered; the nonlinearity of the reservoir and other, far-in-the-past inputs induce noise that complicates recovery, so the variance of the linear part of the reservoir must be placed in such a way as to maximize how much information can be recovered. First we argue that the inputs should be projected into orthogonal directions of the reservoir state space, so that they do not add noise to each other; within those orthogonal directions, the variance should be such that it is as evenly spread across the different dimensions –inputs – as possible, as concentrating the variance into few inputs makes the rest loose precision. This is quantified by the covariance of the neurons.

We start by noticing that the linear nature of the projection vector $\mathbf{w}_{\text{out}}$ implies that we are treating the system as

$$\mathbf{x}(t) = \sum_{k=0}^{\infty} \mathbf{a}_k r(t-k) + \boldsymbol{\varepsilon}_r(t) \tag{S7}$$

where the vectors $\mathbf{a}_k \in \mathbb{R}^N$ are correspond to the linearly extractable effect of $r(t-k)$ onto $\mathbf{x}(t)$ and $\boldsymbol{\varepsilon}_r(t)$ is the nonlinear contribution of all the inputs onto the state of $\mathbf{x}(t)$.

Notice that there are two perspectives here: on one side, the readout extracts the best linear approximation of past inputs with a noise-like term, and on the other it can be interpreted as a Taylor expansion around an undefined point where the first order corresponds to the first term and the non-linear behavior to the other expansion terms. This separation between linear and non-linear behavior has been thoroughly studied Dambre *et al.* (2012); Ganguli *et al.* (2008) and the general understanding is that linear reservoirs have longer memory, but nonlinearity is needed to perform interesting computations. Here we will not try to leverage or bypass this trade-off, but rather we will show that for a fixed ratio of the non-linearity, the more decorrelated the neurons are the higher the memory.

To maintain this trade-off between linear and non-linear behavior, we will assume that the distribution of the linear and non-linear strengths are fixed. This can be achieved if we impose the probabilities of the neuron states do not change, meaning that the mean, variance and other moments of the neuron outputs are unchanged and hence the strength of the nonlinear effects is unchanged.

A first constraint can also be obtained from the maintained strength of the linear side of Eq.S7

$$\text{var}\left(\sum_{\tau=1}^{\infty} \mathbf{a}_\tau r(t-\tau)\right) = c \tag{S8}$$

where $c$ is a constant.

If we are allowed to shift the linear side of Eq.S7, the natural choice of $\mathbf{a}_\tau$ to maximize the memory capacity would be to impose $a_\tau > 0 \iff \tau > N$, and also make the vectors $\mathbf{a}_\tau$ orthogonal to each other, so that the input at time $t-\tau_1$ does not interfere with the effect of an input at time $t-\tau_2$. This can be introduced into Eq.S8 and we obtain

$$\text{var}\left(\sum_{\tau=1}^{\infty} \mathbf{a}_\tau r(t-\tau)\right) = \sum_{\tau=1}^{N} \text{var}(r(t-\tau))\|\mathbf{a}_\tau\|^2 = \sum_{\tau=1}^{N} \|\mathbf{a}_\tau\|^2 = c \tag{S9}$$

This leaves us with a straightforward choice for the readout vector, namely

$$\mathbf{w}_{\text{out}}^{\tau} = \mathbf{a}_\tau, \tag{S10}$$

which we can plug into the memory capacity to obtain

$$\begin{aligned} M_\tau^* &= \frac{\text{cov}^2(r(t-\tau), \mathbf{a}_\tau \mathbf{x}(t))}{\text{var}(\mathbf{a}_\tau \mathbf{x}(t))} \\ &= \frac{\text{cov}^2(r(t-\tau), \|\mathbf{a}_\tau\|^2 r(t-\tau) + \langle \mathbf{a}_\tau, \boldsymbol{\varepsilon}_r(t)\rangle)}{\text{var}(\|\mathbf{a}_\tau\|^2 r(t) + \langle \mathbf{a}_\tau, \boldsymbol{\varepsilon}_r(t))} \end{aligned} \tag{S11}$$

where $M_\tau^*$ is the maximum memory that can be achieved by shifting the linear side of Eq.S7 but maintaining the linear-nonlinear ratio of variance. We must recall the definition of $\boldsymbol{\varepsilon}_r(t)$ in Eq.S7, which implies that any correlation between its projection on $\mathbf{a}_\tau$ and $r(t)$ would imply that part of $r(t-\tau)$ is projected linearly onto the non-linear

component. This necessarily means that cov$(\langle r(t-\tau), \langle \mathbf{a}_\tau, \boldsymbol{\varepsilon}_r(t)\rangle) = 0$, hence our previous equation becomes

$$M_\tau^* = \frac{\|\mathbf{a}_\tau\|^2}{\|\mathbf{a}_\tau\|^2 + \text{var}\left(\langle \frac{\mathbf{a}_\tau}{\|\mathbf{a}_\tau\|}, \boldsymbol{\varepsilon}_r(t)\rangle\right)}, \tag{S12}$$

and for the sake of simplicity we will name $\|\mathbf{a}_\tau\|^2 = a_\tau$, and var $\left(\langle \frac{\mathbf{a}_\tau}{\|\mathbf{a}_\tau\|}, \boldsymbol{\varepsilon}_r(t)\rangle\right) = \varepsilon_\tau$, leaving us with

$$M_\tau^* = \frac{a_\tau}{a_\tau + \varepsilon_\tau}, \tag{S13}$$

where $a_\tau$ is the squared modulus of the linear projection of the input $r(t-\tau)$ on the reservoir state and $\varepsilon_\tau$ is the variance in that direction induced by the non-linear terms in that same direction.

Hence the new problem that we must solve is to maximize

$$\sum_{\tau=1}^{N} M_\tau^* = \sum_{\tau=1}^{N} \frac{a_\tau}{a_\tau + \varepsilon_\tau} \tag{S14}$$

subject to the constraint

$$\sum_{\tau=1}^{N} a_\tau = c, \tag{S15}$$

with an order on the coefficients arising from the contractiveness or the reservoir

$$a_{\tau+1} < a_\tau \tag{S16}$$

and under the assumption that we do not change the nonlinear effects on any direction, hence $\varepsilon_\tau$ will be fixed.

Finally, we shall also note that we will assume that $M_\tau^*$ is always a monotonically decreasing variable that goes from $M_1^* \approx 1$ to $M_N^* \approx 0$, as we observed in the plots on Fig. 1. By noting that

$$M_\tau^* = \frac{1}{1 + \frac{\varepsilon_\tau}{a_\tau}}, \tag{S17}$$

this assumption implies that $q_\tau = \frac{\varepsilon_\tau}{a_\tau}$ goes from $r_1 \approx 0$ to $r_N \to \infty$, hence $\varepsilon_\tau$ starts being much smaller than $a_\tau$ and decreases much slower than $a_\tau$.

### *1. Optimizing the linear projections*

Now the problem is to find how a change in the distribution of $[a_1, a_2, ..., a_N]$ would affect the value of $M^*$. Our approach will be to show that the fastest $a_\tau$ decreases, the lower $M^*$.

We will show that there is one case, namely $a_\tau \propto \varepsilon_\tau$ which has higher $M^*$ than an other setting where $a_\tau$ decreases faster. Since we know that the values of $a_\tau$ must decrease faster than $\varepsilon_\tau$, our best arrangement of the strength of the linear projection of $r(t)$ is to try to make $a_\tau$ decrease as slowly as possible.

With a fixed ratio $a_\tau = \chi\varepsilon_\tau$, the upper bound on the memory is

$$M^* = \sum_{\tau=1}^{N} \frac{a_\tau}{a_\tau + \varepsilon_\tau} = \sum_{\tau=1}^{N} \frac{1}{1 + \frac{\varepsilon_\tau}{a_\tau}} = N\frac{1}{1+\chi}. \tag{S18}$$

Now we will split the sequence of $M_\tau^*$ into two subsequences, one with $\tau < k$ where the values of $a_\tau$ will be increased by a factor $\beta_+$ and another one with $\tau > k+1$ where the values will be decreased by $\beta_-$. This leads us to the new memory capacity bound,

$$M^* = k\frac{\beta_+}{\beta_+ + \chi} + (N-k)\sum_{\tau=k+1}^{N} \frac{\beta_-}{\beta_- + \chi} \tag{S19}$$

which for simplicity we will normalize to obtain

$$m^* = \frac{M^*}{N} = \phi\frac{\beta_+}{\beta_+ + \chi} + (1-\phi)\frac{\beta_-}{\beta_- + \chi}. \tag{S20}$$

where $\phi = \frac{k}{N}$, and $m^*$ is just a normalized memory capacity bound.

Note that the function $m^*$ is still subject to the constraints presented in Eq.S15 and Eq.S16. By introducing $\gamma = \frac{1}{c}\sum_{\tau=1}^{k} a_\tau$ we obtain

$$\beta_+\gamma + \beta_-(1-\gamma) = 1. \tag{S21}$$

Now we can introduce another variable $q = \frac{\beta_+}{\beta_-}$ which determines the difference between $\beta_+$ and $\beta_-$. As our main point is to show that a faster decrease of $a_\tau$ would decrease $m^*$, which in this particular case translates to

$$\frac{\partial m^*}{\partial q} < 0, \tag{S22}$$

which would imply that $\beta_+$ is larger than $\beta_-$ and hence $a_\tau$ decreases faster than $\varepsilon_\tau$ at $\tau = k$ and equally at any other $\tau$.

We can now compute the derivative

$$\frac{\partial m^*}{\partial q} = \phi\frac{\chi}{(\beta_+ + \chi)^2}\frac{\partial \beta_+}{\partial q} + (1-\phi)\frac{\chi}{(\beta_- + \chi)^2}\frac{\partial \beta_-}{\partial q} \tag{S23}$$

where the first term is positive and the second is negative – because the memory capacity for $\tau < k$ will increase when $\beta_+$ grows and vice-versa. Since $\beta_+ > 1 > \beta_-$, $\frac{\chi}{(\beta_+ + \chi)^2} < \frac{\chi}{(\beta_- + \chi)^2}$ we only need to prove that

$$\phi\frac{\partial \beta_+}{\partial q} < -(1-\phi)\frac{\partial \beta_-}{\partial q}. \tag{S24}$$

To do so we will use the constraint from Eq.S21. By setting $\beta_+ = \beta_- q$ we can solve both equations and obtain

$$\begin{aligned} \beta_+ &= \frac{1}{\gamma}\frac{q}{q+\frac{1-\gamma}{\gamma}} \Rightarrow \frac{\partial \beta_+}{\partial q} &&= \frac{\frac{1-\gamma}{\gamma}}{(q+\frac{1-\gamma}{\gamma})^2} \\ \beta_- &= \frac{1}{\gamma}\frac{1}{q+\frac{1-\gamma}{\gamma}} \Rightarrow \frac{\partial \beta_-}{\partial q} &&= -\frac{1}{(q+\frac{1-\gamma}{\gamma})^2}. \end{aligned} \tag{S25}$$

We can plug this into Eq.S24, which simplifies to

$$\begin{aligned} &\frac{\phi}{\gamma}\frac{\frac{1-\gamma}{\gamma}}{(q+\frac{1-\gamma}{\gamma})^2} < \frac{1-\phi}{\gamma_+}\frac{1}{(q+\frac{1-\gamma}{\gamma})^2} \\ &\iff \phi(1-\gamma) < (1-\phi)\gamma \iff \phi < \gamma, \end{aligned} \tag{S26}$$

which is true by the definitions of $\gamma$ and $\phi$, and the fact that $a_\tau$ is decreasing.

Now we have proven that whenever we can take an index $k$ and increase $a_\tau$ for $\tau > k$ and decrease $a_\tau$ for $\tau < k$, the memory capacity bound $M^*$ decreases. We shall now use this result to prove that whenever

$$\frac{\epsilon_{\tau+i}}{\epsilon_\tau} < \frac{a_{\tau+i}}{a_\tau} \quad \forall i, \tau > 0, \tag{S27}$$

the memory capacity bound $M^*$ is lower than when $\varepsilon_\tau \propto a_\tau$.

We do so by starting with the case $\varepsilon_\tau \propto a_\tau$ and we will change it step by step to a series of $a'_\tau$ that fulfills Eq.S27. This is a simple iterative procedure. We start with $k = 1$, then select $q = \frac{\beta_+}{\beta_-}$ such that $a_k\beta_+ = a'_k$ under the constraint from Eq.S21. Then we fix $a_1$ and we have the same problem for $a_\tau$ starting at $\tau \geq 2$. This process can be repeated until $k = N$, at which point the memory bound $m^*$ will be lower because every modification with index $k$ lowered it.

Note that we have used $a_\tau$ in our argument, hence we obtained that the distribution of $a_\tau$ should be as homogeneous as possible. However, our result can also be stated for $a_\tau + \varepsilon_\tau$, because the non-linear effects of the reservoir

on every direction are unchanged and the series $\varepsilon_\tau$ is also a decreasing one.

### 2. Correlations as constraints on the variance

From the previous section we know that the memory bound increases when the variance along the projections of the input into the reservoir state become more homogeneous. This can be expressed in terms of the state space of the reservoir. Intuitively, the variances at directions $\mathbf{a}_\tau$ must fit into the variances of the state space, and since we already established orthogonality of the projections, those variances must be along orthogonal directions. Since our goal is to have a variance as homogeneous as possible along the directions of $\mathbf{a}_\tau$, we need variance that is as homogeneous along orthogonal directions. Finally, this homogeneity is reduced when we add correlations between neurons.

We shall recall that we started our discussion by assuming that the probability distribution of neuron states is unchanged, which would ensure that the strength of the nonlinearity is not altered. This implies that the distribution of variances of the neurons is fixed. If we start by having zero correlations, we can start by setting

$$a_\tau + \varepsilon_\tau = \text{var}\left(x_{\text{sort}(\tau)}(t)\right) \tag{S28}$$

where sort$(\tau)$ is the operation that finds the neuron with the $\tau$th largest variance in the reservoir. In other words, we associate every neuron to one direction of $\mathbf{a}_\tau$, with the constraint that the variances along those directions are ordered, hence we associate $\mathbf{a}_1$ to the neuron with highest variance, $\mathbf{a}_2$ to the second and so on.

The distribution of $a_\tau + \varepsilon_\tau$ in that particular case is then given by the distribution of var $(x_n(t))$. If the correlations are not zero, however, we need a new family of vectors which preserves orthogonality across the covariance matrix $\mathbf{C}$. This is given by the eigenvectors of $\mathbf{C}$. In that framework, the new variances are given by the eigenvalues of the covariance matrix, $\lambda_n(\mathbf{C})$. Naturally, when the correlations are zero, the eigenvalues correspond to the entries of the diagonal, which in our case are the variances as in the previous case.

Hence we have to now work on the distribution of the eigenvalues of the covariance matrix. Specifically, we would want to show that increasing the correlations between neurons increases the inhomogeneity of the eigenvalues, which would decrease our memory bound $M^*$. A simple way to quantify this inhomogeneity is the mean with respect to the square root of the raw variance, which is given by

$$\nu = \frac{\sum_{n=1}^{N} \lambda_n^2(\mathbf{C})}{\left(\sum_{n=1}^{N} \lambda_n(\mathbf{C})\right)^2}, \tag{S29}$$

where $\lambda_n(\mathbf{C})$ is the $n$th eigenvalue of $\mathbf{C}$. To get an intuition of how this metric reflects the inhomogeneity, consider the case of two eigenvalues $\lambda_1,\ \lambda_2$; when $\lambda_1 = \lambda_2$ –very homogeneous – then $\nu = \frac{1}{2}$, but when $\lambda_1 > 0, \lambda_2 = 0$ – very inhomogeneous–, then $\nu = 1$. For $N \gg 1$ the perfectly homogeneous case approach zero but the perfectly inhomogeneous one is still one.

We can compute $\left(\sum_{n=1}^{N} \lambda_n(\mathbf{C})\right)^2$ by using th relationship between trace and eigenvalues,

$$\sum_{n=1}^{N} \lambda_n(\mathbf{C}) = \text{tr}\left[\mathbf{C}\right] = \sum_{n=1}^{n} \text{var}\left(x_n(t)\right) \tag{S30}$$

which is constant by the assumption that the probability distributions of the neuron activities are fixed. Hence we can focus on the value of $\sum_{n=1}^{N} \lambda_n^2(\mathbf{C})$. This is easily done by noting that

$$\mathbf{C}^k v_n(\mathbf{C}) = \lambda_n(\mathbf{C})\mathbf{C}^{k-1} v_n(\mathbf{C}) = \lambda_n^k(\mathbf{C}) v_n(\mathbf{C}) \tag{S31}$$

where $v_n(\mathbf{C})$ and $\lambda_n(\mathbf{C})$ are, respectively the $n$th eigenvector and eigenvalue of $\mathbf{C}$. If we plug this into the relationship between trace and eigenvalues we obtain

$$\sum_{n=1}^{N} \lambda_n^2(\mathbf{C}) = \text{tr}\left[\mathbf{C}^2\right], \tag{S32}$$

which we can compute by decomposing the square of covariance matrix and obtain

$$\sum_{n=1}^{N} \lambda_n^2(\mathbf{C}) = \sum_{n=1}^{N}\sum_{m=1}^{N} \mathbf{C}_{nm}\mathbf{C}_{mn} = \sum_{n=1}^{N} \mathsf{cov}\left(x_n(t), x_m(t)\right)^2 . \tag{S33}$$

This obviously grows when the neurons become correlated. Hence, the inhomogeneity measured by $\nu$ grows.

### *3. Example*

The bound developed in previous sections might seem a bit artificial and far from the standard practice of reservoir computing, particularly the notion that we can adapt the linear part of the dynamics as we want. To make it more understandable and to show that the bound is indeed sharp we will present a simple example where all our assumptions are easily verified and the bounds are sharp.

For this we consider a line of neurons with the input on the first one. That is,

$$\mathbf{W}_{ij} = \begin{cases} w & \iff j = i+1 \\ 0 & \text{otherwise}, \end{cases} \tag{S34}$$

with $w < 1$ to keep the contractivity and the input is only sent to the first neuron, meaning that $\mathbf{w}_{\mathsf{in}} = [w_{\mathsf{in}}, 0, 0, ...]$. If we let $w_{\mathsf{in}} \ll 1$, then the network is effectively linear, because the hyperbolic tangent is almost an identity around $0$. Then the reservoir state becomes

$$\mathbf{x}(t) \approx w_{\mathsf{in}} \left[u(t), wu(t-1), ..., w^N u(t-N)\right] \tag{S35}$$

which corresponds to the case where $\varepsilon_\tau \approx 0$, and the previous input can be easily recovered, and this gives us $M = M^* = N$, which is the memory maximum Dambre *et al.* (2012); Jaeger (2001b). If we increase the value of $w_{\mathsf{in}}$, the reservoir is described as

$$\mathbf{x}(t) = \left[\tanh\left(w_{\mathsf{in}} u(t)\right), \tanh\left(w \tanh\left(w_{\mathsf{in}} u(t-1)\right)\right), ...\right] \tag{S36}$$

where a readout can be obtained for each delay but the nonlinearity of repeatedly applying the hyperbolic tangent makes the memory harder and harder to recover, so the factor $\frac{\varepsilon_\tau}{a_\tau}$ grows. Naturally, here $M = M^*$.

In either case, the covariance matrix is diagonal because the inputs $r(t),\ r(t-\tau)$ are uncorrelated. Notice that, if we add connections between neurons or if we feed inputs with some alternative $\mathbf{w}_{\mathsf{in}}$, then we would be increasing the correlations because there would be more neurons fed by the same input. This can only decrease the memory, because a reservoir with a line of neurons achieves its maximum memory capacity Dambre *et al.* (2012); White *et al.* (2004).

## VII. CORRELATIONS AND EIGENVALUES IN DYNAMICAL SYSTEMS

We will show that the larger the eigenvalues of $\mathbf{W}$, the lower the correlations.

To do so we will first linearize the system presented in Eq.S1, which gives us

$$\mathbf{x}(t) = \mathbf{W}\mathbf{x}(t-1) + \mathbf{w}^{\text{in}}u(t). \tag{S37}$$

This linearizion might seem unjustified, as a key requirement of a reservoir is that it must be non-linear Jaeger (2002) to provide the necessary diversity of computations that a practical ESN requires. However, here we are interested in the memory capacity, which is maximized for linear reservoirs Jaeger (2002); White *et al.* (2004). That is, by studying a linear system we are implicitly deriving an upper bound on the memory, similarly to the approach taken in the control-theoretical study of the effect of the spectral radius Jaeger (2001b). Finally, note that this linearizion is within the parameters of the ESN from Eq.S1, as it would suffice to set $\|\mathbf{w}^{\text{in}}\| \ll 1$.

Given that our system is linear, we can formulate the state of a single neuron $x_i(t)$ as

$$x_i(t) = \sum_{k=0}^{\infty} \left(W^k \mathbf{w}^{\text{in}}\right)_i u(t-k) = \sum_{k=0}^{\infty} a_{i,k} u(t-k) = \langle \mathbf{a}_i, \mathbf{u}_t \rangle \tag{S38}$$

where the vector $\mathbf{a}_i = [a_{i,0}, a_{i,1}, ...]$ represents the coefficients that the previous inputs $\mathbf{u}_t = [u(t), u(t-1), ...]$ have on $x_i(t)$. We can then plug this into the covariance between two neurons,

$$\text{cov}\left(x_i, x_j\right) = \lim_{T\to\infty} \frac{1}{T} \sum_{q=t}^{t+T} \langle \mathbf{a}_i, \mathbf{u}_q \rangle \langle \mathbf{a}_j, \mathbf{u}_q \rangle = \langle \mathbf{a}_i, \mathbf{a}_j \rangle \lim_{T\to\infty} \frac{1}{T} \sum_{q_i=0}^{T} \sum_{q_j=0}^{T} \langle \mathbf{u}_{q_i}, \mathbf{u}_{q_j} \rangle, \tag{S39}$$

and given that $u(t)$ is a random time series with zero autocorrelation and variance of one,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{q_i=0}^{T} \sum_{q_j=0}^{T} \langle \mathbf{u}_{q_i}, \mathbf{u}_{q_j} \rangle = \lim_{T\to\infty} \frac{1}{T} \sum_{q=0}^{T} \langle \mathbf{u}_q, \mathbf{u}_q \rangle = \mathbb{E}\left[u^2(t)\right] = 1. \tag{S40}$$

This gives us

$$\text{cov}\left(x_i, x_j\right) = \langle \mathbf{a}_i, \mathbf{a}_j \rangle, \tag{S41}$$

and similarly, we can compute the variance of $x_i$,

$$\text{var}\left(x_i\right) = \text{cov}\left(x_i, x_i\right) = \langle \mathbf{a}_i, \mathbf{a}_i \rangle = \|\mathbf{a}_i\|^2. \tag{S42}$$

We can plug the previous two formulas into the Pearson's correlation coefficient between two nodes $i, j$ as:

$$P_{ij} = \frac{\langle \mathbf{a}_i, \mathbf{a}_j \rangle}{\|\mathbf{a}_i\| \|\mathbf{a}_j\|} = \cos(\mathbf{a}_i, \mathbf{a}_j) \tag{S43}$$

which is the same as the cosine distance between vectors $\mathbf{a}_i$ and $\mathbf{a}_j$.

The next step is thus to write $\mathbf{a}_i$ as a function of the eigenvalues of $\mathbf{W}$. To do so, we note that the state of a neuron can be written as

$$\mathbf{x}(t) = \sum_{k=0}^{\infty} \mathbf{W}^k \mathbf{w}^{\text{in}} u(t-k) = \sum_{k=0}^{\infty} \left(\mathbf{V}\mathbf{\Lambda}^k \mathbf{V}^{-1}\right) \mathbf{w}^{\text{in}} u(t-k) \tag{S44}$$

where $\mathbf{V}$ is the matrix eigenvectors of $\mathbf{W}$ and $\mathbf{\Lambda}$ the diagonal matrix containing the eigenvalues of $\mathbf{W}$. When we obtain

$$x_i(t) = \sum_{k=0}^{\infty} \sum_{n=1}^{N} \lambda_n^k \langle v_n^{-1}, \mathbf{w}^{\text{in}} \rangle (v_n)_i u(t-k), \tag{S45}$$

where $\mathbf{v}_n$ and $\mathbf{v}_n^{-1}$ are, respectively, the left and right eigenvectors of $\mathbf{W}$. Notice that as long as the network given by $\mathbf{W}$ is drawn from an edge-symmetric probability distribution – meaning that $\text{Pr}\left[\mathbf{W}_{ij} = a\right] = \text{Pr}\left[\mathbf{W}_{ji} = a\right]\ \forall a \in \mathbb{R}$–

and is self averaging then $\mathbf{v}_n$ and $\mathbf{v}_n^{-1}$ are vectors drawn from the same distribution.

The $\lambda_n^k$ terms present in the previous equation can be used as a new vector basis,

$$x_i(t) = \sum_{n=1}^{N} \langle \mathbf{v}_n^{-1}, \mathbf{w}^{\text{in}} \rangle (\mathbf{v}_n)_i \langle \boldsymbol{\lambda}_n, \mathbf{u}_t \rangle = \sum_{k=0}^{\infty} \sum_{n=1}^{N} \lambda_n^k b_{i,n} u(t-k), \tag{S46}$$

where $\boldsymbol{\lambda}_n = \left[1, \lambda_n, \lambda_n^2, ...\right]$ and $b_{i,n} = \langle \mathbf{v}_n^{-1}, \mathbf{w}^{\text{in}} \rangle (\mathbf{v}_n)_i$. By simple identification from Eq.S38, we find that

$$(\mathbf{a}_i)_k = \sum_{n=1}^{N} \lambda_n^k b_{i,n}. \tag{S47}$$

Thus every coefficient of $\mathbf{a}_i$ is a sum of many terms. Specifically, every term is a multiplication of $b_{i,n}$, which are all independent as they refer to the projections of $v_{n,i}$ into $\mathbf{w}_{\text{in}}$ and the values of $\lambda_n$, whose phase – which we assume to be uniformly distributed on $[0, 2\pi]$– ensures that $(\mathbf{a}_i)_k$ is uncorrelated with $(\mathbf{a}_i)_{k+1}$.

We will now proceed to cast the distribution of $\mathbf{a}_i$ as a uniform distribution of points defining an ellipsoid. By the central limit theorem, the values of $\mathbf{a}_i$ are independent random variables drawn from a normal distribution with zero mean and whose variance decreases with the index $k$. Therefore the distribution of $\mathbf{a}_i$ is given by

$$\prod_{k=0}^{\infty} \exp\left(-\frac{(\mathbf{a}_i)_k^2}{s_k^2}\right) \tag{S48}$$

where $s_k$ is a decreasing function of $k$. Thus all the points with probability $\exp\left(-r^2\right)$ are given by the surface

$$\sum_{k=0}^{\infty} \frac{a_k^2}{s_k^2} - r^2 = 0 \tag{S49}$$

which are ellipsoids of infinite dimension and axis $\frac{s_k}{r}$. Furthermore, as we are only interested in the angular coordinates of the points in the ellipsoid, not on their distance to the origin, we can project every one of those surfaces into an ellipsoid with axis

$$\boldsymbol{s} = \left[1, \frac{s_2}{s_1}, \frac{s_3}{s_1}, ...\right]. \tag{S50}$$

Note that, even though the ellipsoid has infinite dimensions, the length of the axes decreases exponentially due to the factor $\lambda_n^k$. Therefore, it has finite surface and it can be approximated by an ellipsoid with finite dimensions.

Now we have that the vectors $\mathbf{a}_i$ are, ignoring their length, uniformly distributed on an ellipsoid with axis $\boldsymbol{\sigma}$. Then

$$\lim_{N\to\infty} S = \frac{1}{2} \int_{E_{\boldsymbol{s}}} \int_{E_{\boldsymbol{s}}} \cos^2(\angle(p,q)) dp dq \tag{S51}$$

where the integrals are taken over $E_{\boldsymbol{s}}$, the ellipsoid with axes $\boldsymbol{s}$, and the half factor comes from counting every pair only once.

If we now change to spherical coordinates we will find that the two vectors can be expressed as

$$p = [r_p, \phi_1^p, \phi_2^p, ...]$$
$$q = [r_q, \phi_1^q, \phi_2^q, ...],$$

where $\phi_1^p$ is the angle of $p$ on the plane given by the first and second axis, $\phi_2^p$ the plane by the first and third axis and so on. The cosine between the two vectors is then

$$\cos(\angle(p,q)) = \prod_{k=1}^{\infty} \cos(\phi_k^p - \phi_k^q). \tag{S52}$$

Thus we can write the integral from Eq.S51 as

$$\lim_{N\to\infty} S = \frac{1}{2}\prod_{k=1}^{\infty}\int_0^{2\pi} \cos^2\left(\phi_k^p - \phi_k^q\right)\mu_{\phi_k}(\phi_k^p)\mu_{\phi_k}(\phi_k^p)d\phi_k^p d\phi_k^q \tag{S53}$$

where $\mu_{\phi_k}(\phi)$ is the probability density function of the difference angle $\phi_k^p - \phi_k^q$.

We will show that this integral decreases when the values $s_k$ increase. To do so, it is helpful to consider the extreme cases to get an intuition: when the semi-minor axis is zero, then we have a line, and all the points in the line have an angle between them either of zero or $\pi$, and thus a squared cosine of one. Conversely, when the semi-minor axis is maximal it equals the semi-major one and we have a circle, and the average squared cosine becomes $1/2$. Those are the two extreme values and thus the squared cosine decreases as the ellipse becomes more similar to a circle.

To make this argument more precise, we start by finding the density $\mu_{\phi_k}(\phi_k^p)$. This density is found by taking a segment of differential length $dl_S$ on the sphere of radius one and then compare it with the length covered in the ellipse $dl_E$. This gives us

$$\begin{aligned}\mu_{\phi_k}(\phi) \propto \frac{dl_E}{dl_S} &= \frac{\|\cos(\phi - d\phi) - \cos(\phi), s_k^2(\sin(\phi - d\phi) - \sin(\phi))\|}{\phi + d\phi - \phi} \\ &= \sqrt{\sin^2(\phi) + a_k^2\cos^2(\phi)} = \sqrt{1 - (1 - s_k^2)\cos^2(\phi)}.\end{aligned}$$

To fully evaluate the previous integral we would need to normalize $\mu_{\phi_k}$ and then evaluate the integral as a function of $s_k$. However, we would take a simpler approach and note that $s_k$ controls the homogeneity of $\mu_{\phi_k}$: the larger $s_k$ is (within the interval $[0, 1]$), the more the mass of probability is concentrated on the area around $\phi \sim 0$ and $\phi \sim \pi$.

Furthermore, we note that the squared cosine has the following periodicities

$$\cos^2(\theta) = \cos^2(\pi + \theta) = \cos^2(\pi - \theta) = \cos^2(2\pi - \theta),$$

thus, when we integrate over the angle $\phi$ we can take advantage of the four-fold symmetry and integrate only on the interval $[0, \pi/2]$. Thus we only need to study the integral

$$\int_0^{\frac{\pi}{2}} \cos^2\left(\phi_k^p - \phi_k^q\right)\mu_{\phi_k}(\phi_k^p)\mu_{\phi_k}(\phi_k^p)d\phi_k^p d\phi_k^q, \tag{S54}$$

and by using $\sin^2(\theta) + \cos^2(\theta) = 1$, we can recast the previous integral as

$$1 - \int_0^{\frac{\pi}{2}} \sin^2\left(\phi_k^p - \phi_k^q\right)\mu_{\phi_k}(\phi_k^p)\mu_{\phi_k}(\phi_k^p)d\phi_k^p d\phi_k^q. \tag{S55}$$

In the interval $\phi \in [0, \pi/2]$ the squared sine can be seen a metric between two angles. Therefore the term

$$\int_0^{\frac{\pi}{2}} \sin^2\left(\phi_k^p - \phi_k^q\right)\mu_{\phi_k}(\phi_k^p)\mu_{\phi_k}(\phi_k^p)d\phi_k^p d\phi_k^q \tag{S56}$$

is nothing else than an average distance between the points which have a density given by $\mu_{\phi_k}$. Thus, the more homogeneous the density, the larger the distance, and vice-versa. Putting it all together, $s_k$ controls the homogeneity of $\mu_{\phi_k}$, and the homogeneity of $\mu_{\phi_k}$ controls the terms on Eq.S53. Specifically, increasing $s_k$ decreases $S$.

The last thing to mention is that the values of $|\lambda_n|$ control $s_k$, as they give the variance to $(\mathbf{a}_i)_k$ in Eq.S47. Thus, the larger $|\lambda_n|$ the higher $s_k$ and the lower $S$. By the negative correlation between $M$ and $S$, increasing the values of $|\lambda_n|$ should increase the memory.

## VIII. MAXIMIZING THE MEMORY CAPACITY

Given that the memory capacity is one of the main characteristics of a reservoir, it is natural to ask whether our set-up offers some insight into how to maximize it. Obviously, the direct answer is to maximize $\langle\lambda\rangle$, so making every eigenvalue such that $|\lambda_n| = 1$.

There are many methods to create matrices with a given spectra, but they tend to be dense, thus we resort to directed circulant networks. Those networks have all nodes aligned in a circle and every node has a connection to the $d$ subsequent clockwise neighbors. For such a network with degree $d$, the corresponding is matrix such that

$$\mathbf{W}_{nm} = \begin{cases} \mathcal{N}(0,1) \iff n - m \mod N \in [1,..,d] \\ 0 \text{ otherwise} \end{cases} \tag{S57}$$

which have eigenvalues distributed in $\lfloor \frac{d+1}{2} \rfloor$ concentric circles centered at the origin (see Aceituno (2018) for details). In any case, the eigenvalues for $d = 1$ are the $N$th roots of the product of the weights, thus after rescaling we obtain $|\lambda_n| = 1$. The resulting memory curve is presented in Fig. 3.

It is worth noticing that this particular architecture is very similar to having a long line of neurons, each one receiving inputs form its predecessor –the only difference being that the last neuron is coupled to the first. This line architecture can obviously provide the maximum memory $M = N$ is the input is scaled appropriately, but it is beyond the traditional rules of reservoir computing: its eigenvalues are aways zero, and its input weights need to be set as $\mathbf{w}_{\text{in}} = [s, 0, 0, ..., 0]$ with $s$ being almost zero. It is remarkable, however, that we obtain a similar architecture just by looking at $\langle\lambda\rangle$.

## IX. RESERVOIR DESIGN IN THE FOURIER DOMAIN

The intuition that we will use here is that every neuron can be seen as a filter that extracts some features from the input time series. If the reservoir extract the right features, the ESN performance would improve.

Training the readout through a linear regression is a minimization of the distance between the linear subspace spanned by the neurons' outputs and the target output, given by

$$\|\mathbf{e}\|^2 = \sum_{t=t_0}^{T+t_0} \left(\hat{y}(t) - \mathbf{w}_{\text{out}}\mathbf{x}(t)\right)^2 \quad = \left\| \hat{y} - \sum_{n=1}^{N} r_n x_n \right\|^2 \tag{S58}$$

where $\|\cdot\|$ is the euclidean norm and $\mathbf{e}$ is the vector of errors, which inhabits the space of the time series. In this space, $\hat{y}$ is the target point, where every value of $\hat{y}(t)$ corresponds to the coordinate of $\hat{y}$ at dimension $t$. The time series of the neurons $x_n$ are also points in that space with $x_n(t)$ being their corresponding coordinates. Then, $\mathbf{w}_{\text{out}}\mathbf{x}$ is the linear combination of neuron points that is closest to $\hat{y}$. Having this geometrical interpretation of the ESN training it is clear that we should set $x_n$ to be as close as possible to the target $\hat{y}$, then the error should also be reduced.

To make this statement more precise, we develop a bound that shows how the error in the training error changes as the neurons become more similar to the readout. We do so by decomposing every neuron time series

$$x_n = x_n^{||y} + x_n^{\perp y} \tag{S59}$$

where $x_n^{||y}$ is the projection of $x_n$ onto the direction of $\hat{y}$ and $x_n^{\perp\hat{y}}$ the part that is orthogonal to it. We propose a very simple readout vector,

$$\left(\mathbf{w}^*_{\text{out}}\right)_n = s_n \gamma \tag{S60}$$

where

$$\gamma = \frac{\|\hat{y}\|}{\sum_{n=1}^{N} \|x_n^{||\hat{y}}\|}, \quad s_n = \begin{cases} 1 & \iff x_n^{||\hat{y}} > 0 \\ -1 & \iff x_n^{||\hat{y}} \leq 0. \end{cases} \tag{S61}$$

Thus the readout simply adds the vectors $x_n$ in the direction of $\hat{y}$ and scales them equally so that

$$\sum_{n=1}^{N} \left(\mathbf{w}^*_{\text{out}}\right)_n x_n^{||\hat{y}} = \hat{y}. \tag{S62}$$

Notice that our readout is obviously not the optimal, one, so the error that we get is *harder*,

$$\|e\| \leq \|e^*\| = \left\| \hat{y} - \sum_{n=1}^{N} \left(\mathbf{w}^*_{\text{out}}\right)_n x_n \right\|. \tag{S63}$$

We can now decompose $x_n$, obtaining

$$\left\| \hat{y} - \sum_{n=1}^{N} \left(\mathbf{w}^*_{\text{out}}\right)_n x_n \right\|^2 = \left\| \left( \hat{y} - \sum_{n=1}^{N} \left(\mathbf{w}^*_{\text{out}}\right)_n x_n^{||\hat{y}} \right) + \sum_{n=1}^{N} r_n^* x_n^{\perp\hat{y}} \right\|^2, \tag{S64}$$

which, given Eq.S62, becomes

$$\|e^*\|^2 = \left\| \sum_{n=1}^{N} s_n x_n^{\perp\hat{y}} \right\|^2 = \|\hat{y}\|^2 \frac{\left\| \sum_{n=1}^{N} s_n x_n^{\perp\hat{y}} \right\|^2}{\left( \sum_{n=1}^{N} \|x_n^{||\hat{y}}\| \right)^2}. \tag{S65}$$

In the worst case, $s_n x_n^{\perp\hat{y}}$ are all aligned on and on the same direction, so $\| \sum_{n=1}^{N} s_n x_n^{\perp y} \| = \sum_{n=1}^{N} \|x_n^{\perp\hat{y}}\|$. This yields

the bound

$$\sigma^2 = \frac{\|e\|^2}{\|\hat{y}\|^2} \leq \frac{\|e^*\|^2}{\|\hat{y}\|^2} = \frac{\left(\sum_{n=1}^{N} \|x_n^{\perp\hat{y}}\|\right)^2}{\left(\sum_{n=1}^{N} \|x_n^{||\hat{y}}\|\right)^2}. \tag{S66}$$

The final step in this geometric bound is to notice that the basis that we used for our space is not unique; when we talk about $x_n^{\perp\hat{y}},\ x_n^{||\hat{y}}, \hat{y}$, we do not need to use the basis where every entry corresponds to one time entry. We can instead choose another orthonormal basis and we will still keep the same values and distances. Our choice here is the Fourier basis, which is a natural choice when thinking about time series or signals Elliott (2013). At this point it us useful to drop the geometric interpretation and come back to the time series or frequency formulation of $\hat{y}$ and $e^*$, giving

$$\sigma^2 \leq \frac{\|e^*\|^2}{\|\hat{y}\|^2} = \frac{\left(\sum_{n=1}^{N} \|\mathfrak{F}\left[x_n^{\perp\hat{y}}\right]\|\right)^2}{\left(\sum_{n=1}^{N} \|\mathfrak{F}\left[x_n^{||\hat{y}}\right]\|\right)^2}, \tag{S67}$$

where $\mathfrak{F}[x_n]$ is the same as $x_n$ in the Fourier basis. By noting that

$$\begin{aligned} \|\mathfrak{F}\left[x_n^{\perp\hat{y}}\right]\| &= |\mathfrak{F}[x_n] \times \mathfrak{F}[\hat{y}]|\ \|\hat{y}\| \\ \|\mathfrak{F}\left[x_n^{||\hat{y}}\right]\| &= |\langle \mathfrak{F}[x_n], \mathfrak{F}[\hat{y}]\rangle|\ \|\hat{y}\| \end{aligned} \tag{S68}$$

we obtain the bound

$$\sigma \leq \frac{\sum_{i=1}^{N} |\mathfrak{F}[x_i] \times \mathfrak{F}[\hat{y}]|}{\sum_{i=1}^{N} |\langle \mathfrak{F}[x_i], \mathfrak{F}[\hat{y}]\rangle|}, \tag{S69}$$

as shown in the main text.

## X. ADAPTING THE POWER SPECTRAL DENSITY IN NON-LINEAR RESERVOIRS

While the relationship between cycles and frequencies is very natural in linear systems Elliott (2013), we are dealing with a non-linear reservoir, meaning that we must prove that adding cycles does indeed modify the frequency response of the reservoir. This is a simple consequence of the monotonicity of the nonlinearity, which implies that adding a cycle of length $L$ will increase the autocorrelation with delay $L$ of the neurons embedded in the cycle, and this in turn increases the PSD of the neuron.

To make this more precise, we start by applying the Wiener-Khinshin theorem Khintchine (1934); Wiener *et al.* (1930) to a neuron indexed by $n$,

$$\mathsf{PSD}_{x_n}(f) = |\mathscr{F}[x_n](f)|^2 = \mathbb{E}_t\left[x_n(t)x_n(t-\tau)\right] = C_{x_n}(\tau), \tag{S70}$$

where $\frac{\tau}{T} = f$ and $C_{x_n}(\tau)$ is the autocorrelation function. Hence, to increase the PSD of neuron $n$ at frequency $f$ we need to increase the autocorrelation of neuron $n$ with itself at delay $fT$. To show that this can be done by rewiring the network so that there are many cycles of length $L$ with positive feedback, we start by writing the equation describing a neuron state,

$$x_n(t) = \tanh\left(v_n u(t) + \sum_{m=1}^{N} w_{mn} x_m(t-1)\right). \tag{S71}$$

By applying the mean value theorem

$$\begin{aligned} x_n(t) &= g\left(x_n, t\right)\left[v_n u(t) + \sum_{m\in\mathcal{S}_n} w_{mn} x_m(t-1)\right] \\ g\left(x_n, t\right) &= \tanh'\left(c\left[v_n u(t) + \sum_{m\in\mathcal{S}_n} w_{mn} x_m(t-1)\right]\right) \end{aligned} \tag{S72}$$

where $c_{t,n} \in [0,1]$. This can be expanded recursively to obtain

$$\begin{aligned} x_n(t) = &\sum_{l=1}^{\tau-1}\sum_{m=1}^{N}\sum_{\mathbf{p}\in\mathscr{P}_l(n,m)} G(\mathbf{p},t) w_{\mathbf{p}} v_m u(t-l) \\ &+ \sum_{m=1}^{N}\sum_{\mathbf{p}\in\mathscr{P}_\tau(n,m)} G(\mathbf{p},t) w_{\mathbf{p}} x_m(t-\tau) \end{aligned} \tag{S73}$$

where $\mathscr{P}_l(n,m)$ are the paths from neuron $n$ to neuron $m$ with length $l$ such that each vector $\mathbf{p} = [n, p_1, p_2, ..., p_{l-1}, m]$ contains the indexes of the neurons with $p_0 = n$ and $p_{\dim(\mathbf{p})} = m$. The function $G(p,t) = \prod_{k=1}^{\dim(\mathbf{p})} g(p_k, t-k)$ corresponds to the attenuation given by the nonlinearity along each path, and finally, $w_p = \prod_{k=1}^{\dim(\mathbf{p})} w_{nm}$ is the cumulative weight of path $\mathbf{p}$.

Sparse random large graphs such as the ones used for ESN reservoirs are locally tree-likeWormald *et al.* (1999); Bollobás *et al.* (2010), meaning that for $\tau \ll N$, almost all the neuron states $x_m(t-\tau)$ in the last term of Eq.S73 are distinct and $n \neq m$. However, when we rewire our graph such that it has many cycles with length $\tau$, we obtain

$$x_n(t) = Q(n,t) + x_n(t-\tau)\sum_{\mathbf{c}\in\mathscr{P}_\tau(n,n)} G(\mathbf{c},t) w_{\mathbf{c}}, \tag{S74}$$

where $Q(n,t)$ is the term that includes the first summand of Eq.S73 as well as all the paths from the second summand that are not cyclic. By putting this into the autocorrelation,

$$\mathbb{E}_t\left[x_n(t)x_n(t-\tau)\right] = \overline{Q}(n) + \mathbb{E}_t\left[x_n^2(t)\right]\sum_{\mathbf{c}\in\mathscr{P}_\tau(n,n)} w_{\mathbf{c}}\overline{G}(\mathbf{c}), \tag{S75}$$

where $\overline{Q}(n) = \mathbb{E}_t\left[Q(n,t)\right]$ and $\overline{G}(\mathbf{c}) = \mathbb{E}_t\left[G(\mathbf{c},t)\right]$. Given that all the entries of the weight matrix and the input vector are randomly sampled from probability distributions with mean zero, we would expect $Q(n,t)$ to also have mean zero

across all neurons, so

$$\mathbb{E}_n\left[C_{x_n}(\tau)\right] = \mathbb{E}_n\left[\text{var}\left[x_n(t)\right] \sum_{\mathbf{c}\in\mathcal{P}_\tau(n,n)} w_{\mathbf{c}}\overline{G}(\mathbf{c})\right] \tag{S76}$$

Here it is worth noticing that $G(\mathbf{c},t)$ is a multiplication of $g(m,t) > 0$, hence it is always positive just as the variance of $x_n(t)$. Thus the strength of the average PSD of the neuron the signs of $w_{\mathbf{c}}$.

An important insight from this derivation presented here is that by using the mean value theorem we loose information about the actual value of the autocorrelation. That is, while in linear systems we know *how much* the frequencies will be modified, in non-linear systems we do not, having to conform ourselves with knowing which frequencies will be enhanced or dampened. Yet, the mean value theorem implies that we do not need to know the value of the derivative of the nonlinear function applied to the neurons. Therefore any positive monotonic function will fulfill the requirements of our derivation, hence we can use other nonlinearities such as a sigmoid of a piecewise linear function to obtain the same result.

## XI. ALGORITHM TO ADAPT RESERVOIRS

In the tasks described in the main text, different tasks require different reservoir parameters. Specifically, for a given maximum cycle length value $L$ there is one combination of $\rho_l,\ \forall l \leq L$ and a value of $\langle|\lambda|\rangle$, which optimizes the ESN performance. In this section we present the heuristic that we use to find those parameters.

The first step is to tune the memory for the current task. We take a very simple approach, where we simply try many spectral radii on the interval $[0, 1]$ and pick the one that gives the best performance on a classical ER network. Once found, we calculate the corresponding $\langle|\lambda|\rangle$ using S3 and we will use that for the rest of the process.

Since the reservoirs that we use are not linear, we need to characterize their frequency response for various values of $\rho_l$ for all the $l \leq L$ considered. This response, that we denote $R(\rho_l, L)$ is computed by generating Gaussian noise with the same variance and mean as the original signal and use it as an input for the reservoir. Then apply the Fast Fourier Transform to the neurons' states and average over all neurons. As the reservoirs are generated randomly, it is necessary to average those responses over multiple reservoir instances. For a given $L$ we use the following heuristic to find the optimal combination of cycles with size no greater than $L$.

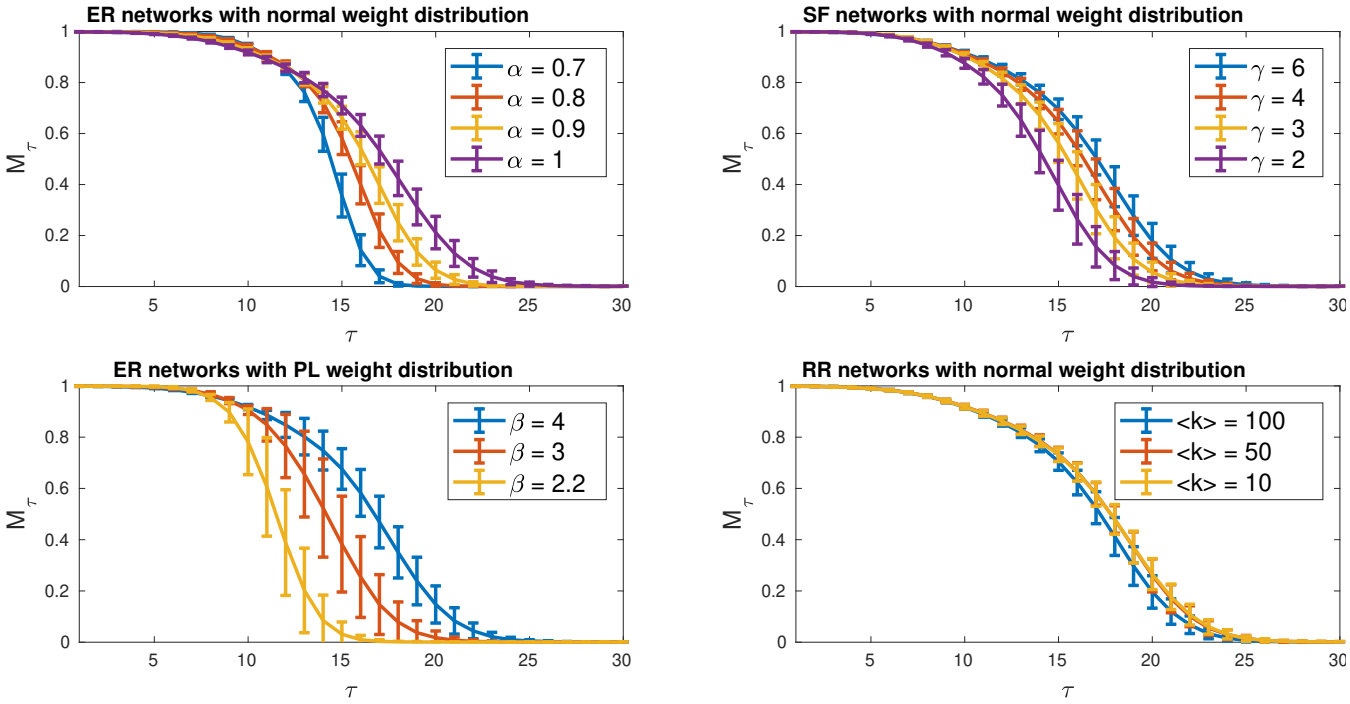
**Step-1:** Compute the Fourier transform of the input signal and keep the vector of absolute values $\hat{s} = [\hat{s}(0), \hat{s}(2), ...\hat{s}(f_S)]$, where $f_S$ is half the sampling frequency.

**Step-2:** Compute the scalar product $\langle\hat{s}, R(\rho_l, l)\rangle$ for all $\rho_l, l$, and select the $\rho_l$ that maximizes it for each $l$.

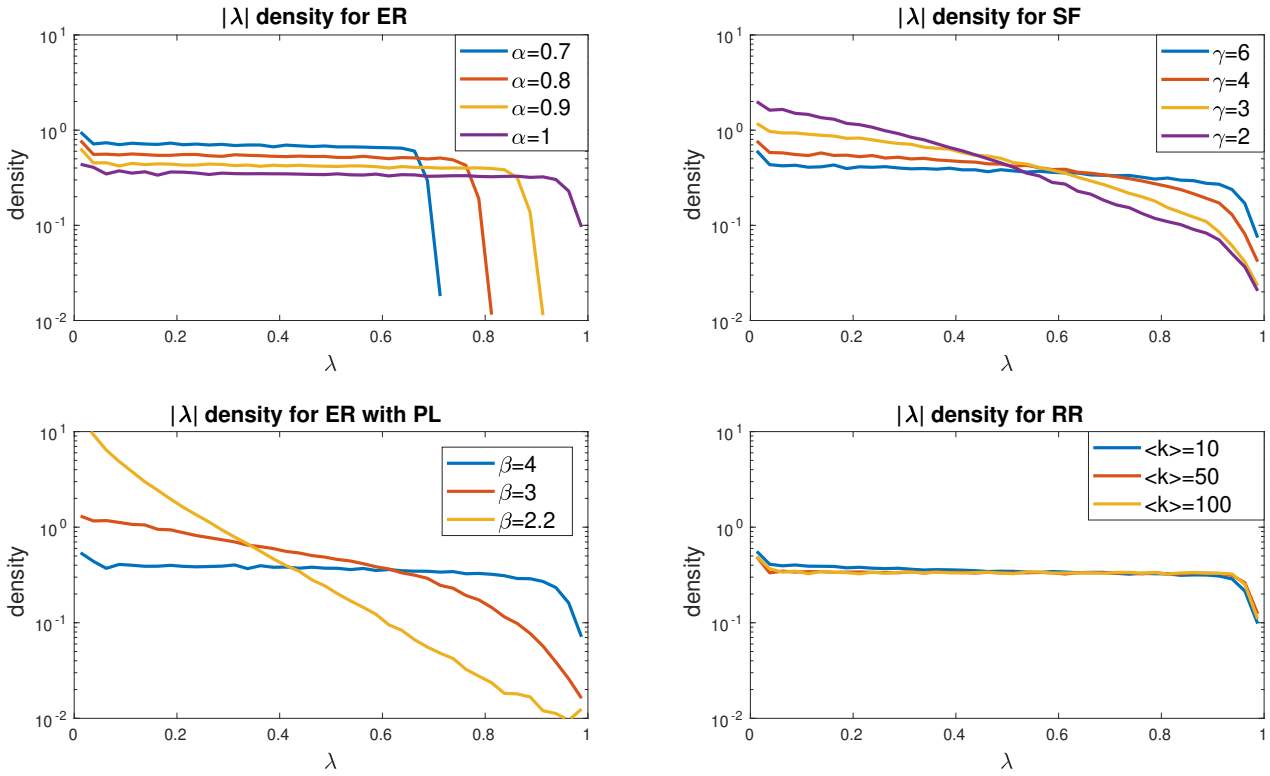
**Step-3:** Test the performance of an ESN with the values of $\rho_l$ found in the previous step. If the performance is lower than in the default case of $\rho_l = 0$, do not optimize with regard to that length.

**Step-4:** For all values of $\rho_l$ where the cycle length is allowed and which fill the condition $\|\rho\|_1 \leq 1$, select the one that maximizes $\sum_l \langle\hat{s}, R(\rho_l, l)\rangle$.

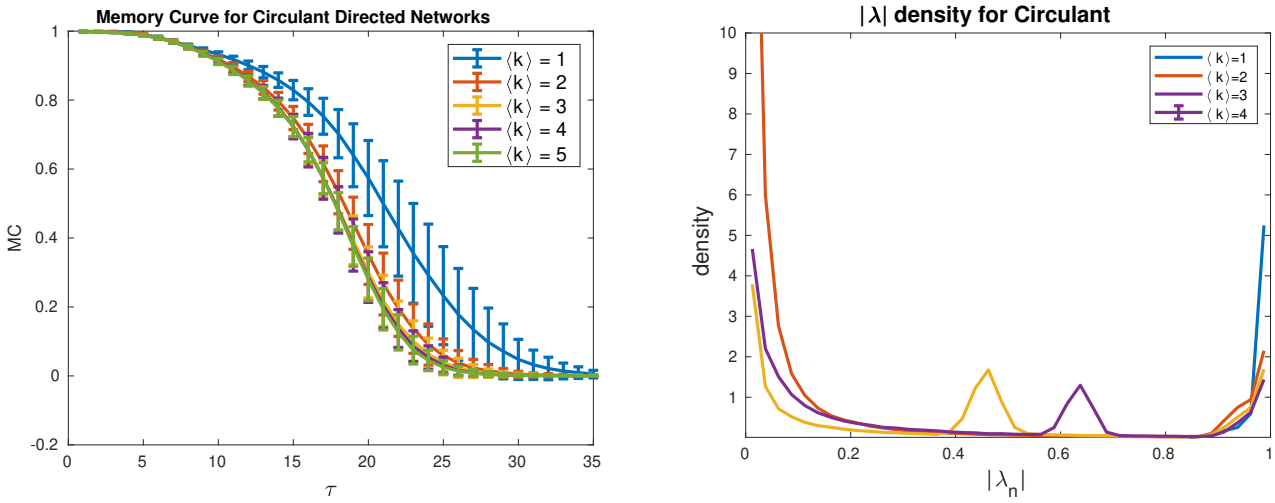
# Part II
# Supplemental Figures




Supplementary Figure 1: **Memory Decay of ESN, Related to Figure.2.** The ability of the reservoir to retrieve previous inputs decays as with $\tau$, the delay with which we try to retrieve them. Each network has 400 neurons and average degree is $\langle k \rangle = 20$. Each curve shows the average result over 100 trials and the error bars are the standard deviation. The spectral radius $\alpha = 1$, except in ER networks where it varies as marked in the legend.




Supplementary Figure 2: **Eigenvalues of various network topologies, Related to Figure.2.** Eigenvalue densities of the random networks studied in the main text. Each line shows the density of eigenvalues at a distance $\lambda$ from the origin. $200$ realizations of a network. The network size is $1000$, the edge weights are drawn from a normal distribution except when in the ER networks with PL. Unless explicitly stated in the sub-figure legend, $\langle k \rangle = 50$ and the spectral radius is $\alpha = 1$.




Supplementary Figure 3: **Memory decay (left) and eigenvalue densities (right) for circulant networks with various degrees, Related to Figure.2.** The memory decays significantly slower for $\langle k \rangle = 1$, corresponding to the eigenvalue distribution for $N = 400$, which is all concentrated at $|\lambda| = 1$ for $\langle k \rangle = 1$, but for $\langle k \rangle > 1$ more and more eigenvalues concentrate around lower $|\lambda|$.

# Part III
# Supplemental References


Aceituno, P. V. (2018), 'Eigenvalues of random graphs with cycles', *arXiv preprint arXiv:1804.04978* .

Bollobás, B., Kozma, R. and Miklos, D. (2010), *Handbook of large-scale random networks*, Vol. 18, Springer Science & Business Media.

Büsing, L., Schrauwen, B. and Legenstein, R. (2010), 'Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons', *Neural Computation* **22**(5), 1272–1311.

Christopher, M. B. (2006), 'Pattern recognition and machine learning', *Company New York* **16**(4), 049901.

Dambre, J., Verstraeten, D., Schrauwen, B. and Massar, S. (2012), 'Information processing capacity of dynamical systems', *Scientific reports* **2**.

Deák, I. (1990), *Random number generators and simulation*, Akadémiai Kiadó.

Duport, F., Schneider, B., Smerieri, A., Haelterman, M. and Massar, S. (2012), 'All-optical reservoir computing', *Optics express* **20**(20), 22783–22795.

Elliott, D. F. (2013), *Handbook of digital signal processing: engineering applications*, Academic Press.

Ganguli, S., Huh, D. and Sompolinsky, H. (2008), 'Memory traces in dynamical systems', *Proceedings of the National Academy of Sciences* **105**(48), 18970–18975.

Goh, K.-I., Kahng, B. and Kim, D. (2001), 'Universal behavior of load distribution in scale-free networks', *Physical Review Letters* **87**(27), 278701.

Hammami, N. and Bedda, M. (2010), Improved tree model for arabic speech recognition, *in* 'Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on', Vol. 5, IEEE, pp. 521–526.

Hübner, U., Abraham, N. and Weiss, C. (1989), 'Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared NH 3 laser', *Physical Review A* **40**(11), 6354.

Huebner, U., Klische, W., Abraham, N. and Weiss, C. (1989), On problems encountered with dimension calculations, *in* 'Measures of Complexity and Chaos', Springer, pp. 133–136.

Jaeger, H. (2001*a*), *Short term memory in echo state networks*, GMD-Forschungszentrum Informationstechnik.

Jaeger, H. (2001*b*), 'The "echo state" approach to analysing and training recurrent neural networks-with an erratum note', *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* **148**, 34.

Jaeger, H. (2002), *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach*, GMD-Forschungszentrum Informationstechnik.

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

Khintchine, A. (1934), 'Korrelationstheorie der stationären stochastischen prozesse', *Mathematische Annalen* **109**(1), 604–615.

Lichman, M. (2013), 'UCI machine learning repository'.
**URL:** *http://archive.ics.uci.edu/ml*

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

Maass, W., Natschläger, T. and Markram, H. (2002), 'Real-time computing without stable states: A new framework for neural computation based on perturbations', *Neural Computation* **14**(11), 2531–2560.

Mermelstein, P. (1976), 'Distance measures for speech recognition, psychological and instrumental', *Pattern recognition and artificial intelligence* **116**, 374–388.

Pascanu, R., Mikolov, T. and Bengio, Y. (2013), On the difficulty of training recurrent neural networks, *in* 'International conference on machine learning', pp. 1310–1318.

Strauss, T., Wustlich, W. and Labahn, R. (2012), 'Design strategies for weight matrices of echo state networks', *Neural Computation* **24**(12), 3246–3276.

White, O. L., Lee, D. D. and Sompolinsky, H. (2004), 'Short-term memory in orthogonal neural networks', *Physical Review Letters* **92**(14), 148102.

Whitley, D. and Watson, J. P. (2005), Complexity theory and the no free lunch theorem, *in* 'Search Methodologies', Springer, pp. 317–339.

Wiener, N. et al. (1930), 'Generalized harmonic analysis', *Acta mathematica* **55**, 117–258.

Wormald, N. C. (1984), 'Generating random regular graphs', *Journal of algorithms* **5**(2), 247–280.

Wormald, N. C. et al. (1999), 'Models of random regular graphs', *London Mathematical Society Lecture Note Series* pp. 239–298.