

Connectivity, Dynamics, and Memory in Reservoir Computing with Binary and Analog Neurons

Lars Büsing

lars@igi.tugraz.at

*Institute for Theoretical Computer Science, Graz University of Technology,
A-8010 Graz, Austria*

Benjamin Schrauwen

benjamin.schrauwen@ugent.be

*Electronics and Information Systems Department, Ghent University,
B-9000 Ghent, Belgium*

Robert Legenstein

legi@tugraz.at

*Institute for Theoretical Computer Science, Graz University of Technology,
A-8010 Graz, Austria*

Reservoir computing (RC) systems are powerful models for online computations on input sequences. They consist of a memoryless readout neuron that is trained on top of a randomly connected recurrent neural network. RC systems are commonly used in two flavors: with analog or binary (spiking) neurons in the recurrent circuits. Previous work indicated a fundamental difference in the behavior of these two implementations of the RC idea. The performance of an RC system built from binary neurons seems to depend strongly on the network connectivity structure. In networks of analog neurons, such clear dependency has not been observed. In this letter, we address this apparent dichotomy by investigating the influence of the network connectivity (parameterized by the neuron in-degree) on a family of network models that interpolates between analog and binary networks. Our analyses are based on a novel estimation of the Lyapunov exponent of the network dynamics with the help of branching process theory, rank measures that estimate the kernel quality and generalization capabilities of recurrent networks, and a novel mean field predictor for computational performance. These analyses reveal that the phase transition between ordered and chaotic network behavior of binary circuits qualitatively differs from the one in analog circuits, leading to differences in the integration of information over short and long timescales. This explains the decreased computational performance observed in binary circuits that are densely connected. The mean

field predictor is also used to bound the memory function of recurrent circuits of binary neurons.

1 Introduction

The idea of using a randomly connected recurrent neural network for on-line computations on an input sequence was independently introduced in Jaeger (2001) and Maass, Natschläger, and Markram (2002). In these papers the network activity is regarded as an “echo” of the recent inputs, and a memoryless readout device is then trained in order to approximate from this echo a given time-invariant target operator with fading memory (see Maass et al., 2002), whereas the network itself remains untrained. Jaeger used analog sigmoidal neurons as network units and named the model echo state network (ESN). Maass termed the idea liquid state machine (LSM), and most of the related literature focuses on networks of spiking neurons or threshold units. Both ESNs and LSMs are special implementations of a concept now generally called reservoir computing (RC), which subsumes the idea of using general dynamical systems—for example, a network of interacting optical amplifiers (Vandoorne et al., 2008) or an analog VLSI cellular neural network chip (Verstraeten et al., 2008), the so-called reservoirs—in conjunction with trained memoryless readout functions as computational devices. These RC systems have been used in a broad range of applications (often outperforming other state-of-the-art methods) such as chaotic time series prediction and nonlinear wireless channel equalization (Jaeger & Haas, 2004), speech recognition (Verstraeten, Schrauwen, Stroobandt, & Campenhout, 2005; Jaeger, Lukoševičius, Popovici, & Siewert, 2007), movement analysis (Legenstein, Markram, & Maass, 2003), and robot control (Joshi & Maass, 2005).

Although ESNs and LSMs are based on very similar ideas—and in applications it seems possible to switch between both approaches without loss of performance (Verstraeten, Schrauwen, D’Haene, & Stroobandt, 2007), an apparent dichotomy exists regarding the influence of the reservoir’s connectivity on its computational performance. The performance of an ESN using analog, rate-based neurons is largely independent of the sparsity of the network (Jaeger, 2007) or the exact network topology such as small-world or scale-free connectivity graphs.¹ For LSMs, which consist of spiking or binary units, a profoundly different effect has been observed: introducing small-world or biologically measured lamina-specific cortical interconnection statistics (Häusler & Maass, 2007) clearly leads to an increase

¹Shown by results of unpublished experiments that have been reported by the lab of Jaeger through personal communication. Of course, drastic topological changes such as disconnecting all network units influence performance. Further, a specific class of (non-normal) connectivity matrices with advantageous memory properties for linear systems was characterized in Ganguli, Huh, and Sompolinsky (2008).

in performance. Further, in the results of Bertschinger and Natschläger (2004), it can be observed (although not specifically stated there) that for networks of threshold gates with a simple connectivity topology of fixed in-degree per neuron, an increase in performance can be found for decreasing in-degree. None of these effects can be reproduced using ESNs.

In order to systematically study this fundamental difference between binary (spiking) LSMs and analog ESNs in a unified framework, we close the gap between these models by introducing in section 2 a class of models termed quantized ESNs (qESNs). The reservoir of a quantized ESN is defined as a network of discrete-valued units, where the number of admissible states of a single unit is controlled by a parameter called state resolution, which is measured in bits. A binary network (where the units have binary outputs) has a state resolution of 1, whereas high-resolution networks (where the units provide high-resolution outputs) have high state resolutions. LSMs and ESNs can thus be interpreted as the two limiting cases of quantized ESNs for low and high state resolution, respectively. We briefly describe the dynamics of qESNs, which exhibit ordered and chaotic behavior separated by a phase transition. Further, the concept of Lyapunov exponents is discussed in the context of qESNs, and an approach to approximately compute them for qESNs is introduced based on branching process theory.

In section 3 we numerically study the influence of the network connectivity parameterized by the in-degree of the network units on the computational performance of quantized ESNs for different state resolutions. This generalizes and systemizes previous results obtained for binary LSMs and analog ESNs.

In section 4 the empirical results are analyzed by studying the Lyapunov exponent of qESNs, which exhibits a clear relation to the computational performance (Legenstein & Maass, 2007a). We show that for binary qESNs, the chaos-order phase transition is significantly more gradual when the networks are sparsely connected. It is exactly in this transition regime that the computational power of an RC system is found to be optimal (Legenstein & Maass, 2007a). This effect disappears for high-resolution ESNs.

A clear explanation of the influence of the network connectivity on the computational performance can be found by investigating the rank measure presented in Legenstein and Maass (2007a). This measure characterizes the computational capabilities of a network as a trade-off between the so-called kernel quality and the generalization ability. We show that for highly connected binary reservoirs, the region of an efficient trade-off implying high performance is narrow. For sparser networks, this region is shown to broaden. Consistently, for high-resolution networks, the region is found to be independent of the interconnection degree.

In section 5 we present a novel mean field predictor for computational power that is able to reproduce the influence of the connectivity on the qESN model. This predictor is based on an estimation of the input separation

property of a network, and it is a generalization of the predictor introduced in Bertschinger and Natschläger (2004) as it can be calculated for general (not just binary) qESNs and it can be determined with a significantly reduced computation time. This enables us to study computational power based on state separation for recurrent networks of nearly analog nonlinear neurons. The novel theoretical measure matches the experimental and rank measure findings closely. It describes high performance as an interplay between input separation on different timescales revealing important properties of RC systems necessary for good computational capabilities.

We investigate the relation of the mean field predictor to the memory function of qESNs in section 6 and show that the latter is bounded by a function of the input separation property. To our best knowledge, this represents the first computationally feasible bound on the memory function of nonlinear networks, since such bounds were previously derived only for linear networks (White, Lee, & Sompolinsky, 2004; Jaeger, 2002; Ganguli et al., 2008). Further, we investigate the scaling of the memory capacity and the temporal capacity of binary qESNs with the network size yielding a logarithmic dependence.

The numerical and theoretical finding that for binary qESNs the optimal performance is observed for sparse connectivity is compared to experimental results regarding the connectivity of neocortical microcircuits in section 7 revealing an apparent discrepancy between the optimal parameter values of the binary qESN model and experimental observations. Sparse network activity, which is ubiquitous in biological spiking networks but absent in the qESN model, is proposed as a possible mechanism that can resolve this discrepancy.

2 Quantized ESNs and Their Dynamics

In this section the network model is defined that will later serve as the dynamic reservoir for qESNs. The networks are reminiscent of random Boolean networks (see Shmulevich, Dougherty, Kim, & Zhang, 2002) and Kauffman networks (see Kauffman, 1969; Derrida & Stauffer, 1986) and exhibit just like the latter two distinct dynamical regimes, the chaotic and the ordered regime, depending on the choice of parameters. These two “phases” are separated by an order-chaos (also termed order-disorder) phase transition (for general information on phase transitions, see Zinn-Justin, 2003).

2.1 Definition of the Network Model. We consider networks of N units with the state variable $\mathbf{x}(t) = (x_1(t), \dots, x_N(t)) \in [-1, +1]^N$ in discrete time $t \in \mathbb{Z}$. All units have an in-degree of K , that is, every unit i receives input from K other randomly chosen units with independently and identically distributed (i.i.d.) weights w_{ij} drawn from a normal distribution $\mathcal{N}(0, \sigma^2)$ with zero mean and standard deviation (STD) σ . All remaining weights are defined as zero. Sample weight matrices $(w_{ij})_{i,j \in \{1, \dots, N\}}$ (constituting

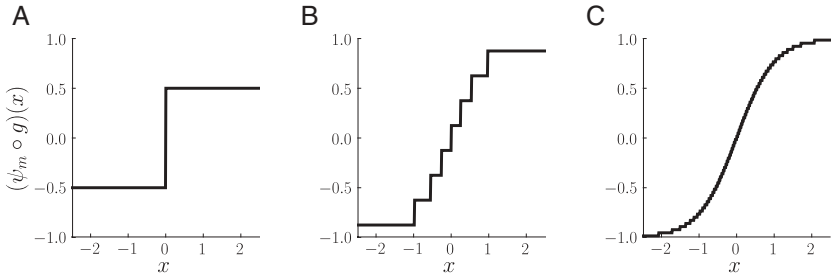


Figure 1: The quantized activation function $\psi_m \circ g$ for state resolutions $m = 1$ (A), $m = 3$ (B), and $m = 6$ (C). Networks with the $m = 1$ activation function, which just assumes two different values, are termed binary reservoirs, whereas networks with large m (here $m = 6$) are termed high-resolution reservoirs as their units possess a large state space of 2^m states. The discretization schema was chosen such that states are equidistant and $\sum_{i=1}^{2^m} s_i p(s_i) = 0$ as well as $\sum_{i=1}^{2^m} |s_i| p(s_i) = 1/2$ independent of the state resolution m assuming a uniform distribution over the single unit state space $p(s_i) = 2^{-m}$.

concrete networks or circuits) from this distribution will be denoted by C . The network state is updated according to

$$x_i(t+1) = (\psi_m \circ g) \left(\sum_{j=1}^N w_{ij} x_j(t) + u(t) \right), \quad (2.1)$$

where $g = \tanh$ is the usual hyperbolic tangent nonlinearity and u denotes the input sequence common to all units. At every time step t , the input $u(t)$ is drawn uniformly from $\{-1, 1\}$. The function ψ_m is called quantization function for m bits as it maps from $(-1, 1)$ to its discrete range \mathcal{S}_m of cardinality 2^m :

$$\psi_m : (-1, 1) \rightarrow \mathcal{S}_m, \quad \psi_m(x) := \frac{2 \lfloor 2^{m-1}(x+1) \rfloor + 1}{2^m} - 1.$$

Here $\lfloor x \rfloor$ denotes the integer part of x . Due to ψ_m , the variables $x_i(t)$ are discrete (quantized) and assume values in the state space $\mathcal{S}_m = \{s_1, \dots, s_{2^m}\} \subset (-1, 1)$ with $s_k := (2k - 1)/2^m - 1$. Three examples of the quantized activation function $\psi_m \circ g$ for state resolutions $m = 1, 3, 6$ are shown in Figure 1.

Depending on the parameters m , K , and σ , the system defined by equation 2.1 shows two qualitatively different behaviors, ordered and chaotic dynamics, which are separated in the parameter space by a sharp boundary, often called the critical line, where a phase transition takes place. The ordered (also called “frozen”) and chaotic regimes are defined via the

average damage spreading properties (sensitivity to perturbations of initial conditions) of the networks (Derrida & Pomeau, 1986; Derrida & Stauffer, 1986; Bertschinger & Natschläger, 2004). One considers the temporal evolution of the average distance $H(t) = \langle \|\mathbf{x}^1(t) - \mathbf{x}^2(t)\| \rangle_{C,u}$ between two states $\mathbf{x}^1(t)$ and $\mathbf{x}^2(t)$ at time t evolving from initial conditions that differ in a single unit at time 0. Here $\|\cdot\|$ denotes some norm in \mathbb{R}^N ; for example, the p-1 norm $\|\cdot\|_1$ and $\langle \cdot \rangle_{C,u}$ denotes the average over networks C (with the same parameters m, σ, K), initial perturbations and input sequences u . A set of parameters m, K, σ is in the ordered phase if $\lim_{t \rightarrow \infty} H(t) =: H^* = 0$, that is, if perturbations in the initial conditions $H(0)$ eventually die out. Parameter sets with $H^* > 0$ are in the chaotic regime where the initial “damage” $H(0)$ persists and influences the network state for all later times. Hence H^* can be considered an order parameter of the phase transition as it is zero in one phase and larger than zero in the other. Examples for the behavior of H^* for state resolutions $m = 1, 3, 6$ and varying parameters σ and K are shown in Figure 2, exhibiting the characteristic behavior of a phase transition.

The state distance $H(t)$ also allows the introduction of a heuristic measure for the speed of divergence of trajectories in discrete time and discrete state-space systems. We term this measure Lyapunov exponent λ as it is reminiscent of the maximal Lyapunov exponent in systems with continuous (see Katok & Hasselblatt, 1995) and binary state space (see Luque & Solé, 2000). It is defined by

$$\lambda = \lim_{T \rightarrow \infty} \frac{1}{T} \ln \left(\frac{H(T)}{H(0)} \right). \quad (2.2)$$

In the ordered regime, we have $\lambda < 0$, while $\lambda > 0$ holds for chaotic dynamics. The above definition of λ makes sense only for infinitely large systems.

In finite size systems, we characterize the speed of divergence of nearby trajectories by a numerical estimation λ_{exp} of the Lyapunov exponent that was determined in the following way. After 20 initial simulation steps, the smallest admissible (for m) state difference $\delta_0(m) = 2^{1-m}$ was introduced in a single network unit, and the resulting state difference δ after one time step was measured averaged over 10^5 trials with randomly generated networks and initial states. The initial states of all neurons were i.i.d. uniformly over \mathcal{S}_m . The estimation λ_{exp} was then determined by $\lambda_{\text{exp}} = \ln(\delta/\delta_0(m))$. This one-step approximation of λ is expected to produce accurate results for large networks with sparse connectivity where all “damages” spread independently through the network. It is shown below that even at a network size of $N = 150$, λ and λ_{exp} agree well for a large regime of network parameters.

2.2 Lyapunov Exponents via Branching Processes. Besides estimating the Lyapunov exponent defined in equation 2.2 using the scheme for finite size systems described above, it can be calculated for infinitely large systems

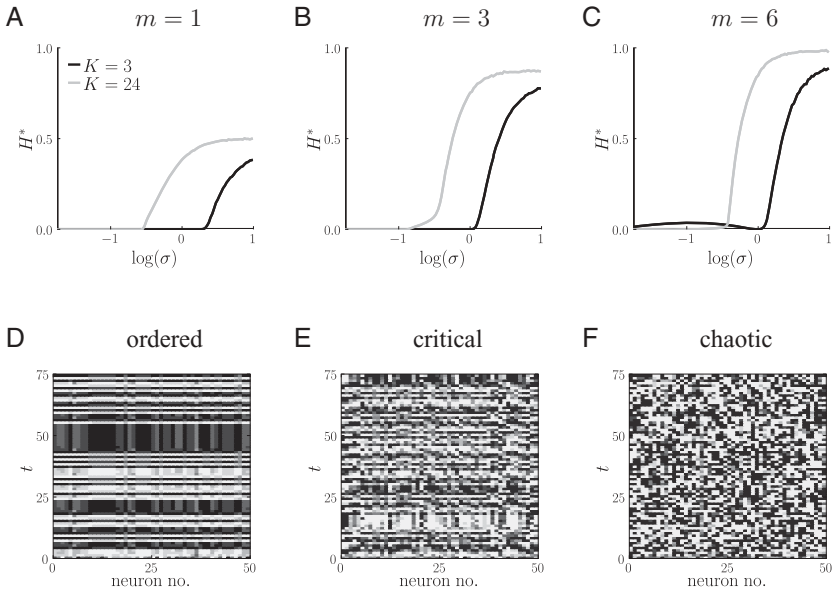


Figure 2: Phase transitions in randomly connected networks with dynamics defined by equation 2.1. (A–C) Shown is the fixed point H^*/N of the normalized distance $H(t)/N$ between two states evolving from different initial conditions in networks with $N = 500$ units for three state resolutions $m = 1, 3, 6$ and varying in-degree K and weight standard deviation σ . The abrupt change in the values of H^* for different parameters is characteristic for a phase transition. Shown results are averages over 500 circuits (with a single random input sequence each). The initial perturbation $H(0)$ was chosen as the smallest admissible perturbation (for the specific m) in a single unit, and H^* was measured after 100 update steps. (D–F) Evolution of the state $x_i(t)$ of 75 out of $N = 500$ units from a network with $m = 3$ and $K = 3$ for $\log(\sigma) = -0.5$ (D), $\log(\sigma) = 0$ (E), and $\log(\sigma) = 0.5$ (F) showing ordered, critical, and chaotic dynamics respectively.

$N \rightarrow \infty$ under the annealed approximation (AA) using results from the theory of multitype branching processes (see Athreya & Ney, 1972). In the AA that was introduced in Derrida and Pomeau (1986), one assumes that the circuit connectivity and the corresponding weights are drawn i.i.d. at every time step. Although being a drastic simplification, the AA has been shown in various studies (see Derrida & Pomeau, 1986; Bertschinger & Natschläger, 2004; White et al., 2004) to be a powerful tool for investigating network dynamics yielding accurate results for large system sizes N ; hence, its application is well justified in the limit $N \rightarrow \infty$ considered here. Branching process theory has already been applied in theoretical neuroscience to describe the temporal and spatial dynamics of neural

activity (see Beggs & Plenz, 2003; Vogels, Rajan, & Abbott, 2005). Here we propose a novel approach by applying branching process theory to study the evolution of perturbations of network states, allowing the approximate evaluation of the Lyapunov spectrum of the network model defined above.

Let $\mathcal{S}_m = \{s_1, \dots, s_{2^m}\}$ denote the single unit state space with a state resolution m . Consider two states $\mathbf{x}^1(t)$ and $\mathbf{x}^2(t)$ of the same infinitely large network (that evolved from different initial conditions). We say that there is a perturbation of type $s_i \rightarrow s_j$ at time t (of $\mathbf{x}^1(t)$ relative to $\mathbf{x}^2(t)$) if there is a network unit, with some index $l \in \mathbb{N}$, which is in the state s_i in $\mathbf{x}^1(t)$ and in the state s_j in $\mathbf{x}^2(t)$, that is, $x_l^1(t) = s_i$ and $x_l^2(t) = s_j$. Assuming the AA, the neuron indices can be permuted arbitrarily (as the weight matrix is regenerated at every time step), and hence the difference between the two states $\mathbf{x}^1(t)$ and $\mathbf{x}^2(t)$ is fully described by counting the perturbations of all types. Now let $\mathbf{x}^1(t)$ and $\mathbf{x}^2(t)$ differ in n coordinates, which can, without loss of generality, be assumed to be the first n coordinates, that is, $x_i^1(t) = s_{a_i} \neq x_i^2(t) = s_{b_i}$ with $a_i, b_i \in \{1, \dots, 2^m\}$ for $i = 1, \dots, n$ and $x_i^1(t) = x_i^2(t) = s_i$ for $i > n$. These n perturbations of types $s_{a_1} \rightarrow s_{b_1}, \dots, s_{a_n} \rightarrow s_{b_n}$ at time t cause perturbations in the next time step $t + 1$. Because of the finite in-degree K and the infinite system size $N = \infty$, these n perturbations give rise to descendant perturbations at $t + 1$ independently. Therefore this system is equivalent to a multitype branching process with $2^m \cdot (2^m - 1)$ types (diagonal perturbations $s_a \rightarrow s_a$ do not contribute), a mathematical model that has been extensively studied (see Athreya & Ney, 1972). The multitype branching process describing the perturbation spreading in the considered network is fully specified by $p_{i,j}^{\alpha,\beta}$ for $\alpha, \beta, i, j = 1, \dots, 2^m$ denoting the probability of a $s_\alpha \rightarrow s_\beta$ perturbation to cause a $s_i \rightarrow s_j$ perturbation per outgoing link in the next time step which can be explicitly calculated using the AA (see appendix A). When the results from branching theory (see Athreya & Ney, 1972) are applied, the maximal Lyapunov exponent λ defined in equation 2.2 is given by the logarithm of the largest eigenvalue (being the Perron root) of the matrix M , whose entries $M_{\alpha+2^m\beta, i+2^mj} = K \cdot p_{i,j}^{\alpha,\beta}$ denote the mean number of descendants of type $s_i \rightarrow s_j$ caused by a $s_\alpha \rightarrow s_\beta$ perturbation. Branching processes with $\lambda < 0$ are called subcritical, corresponding to ordered dynamics, implying that all perturbations eventually die out with probability 1, whereas the case $\lambda > 0$ is termed supercritical, corresponding to chaotic dynamics, implying exponential growth of the number of perturbations on average. For $m > 1$, there is more than one eigenvalue of M giving rise to a Lyapunov spectrum λ_i for $i = 1, \dots, 2^{m-1}(2^m - 1)$ with $\lambda_i \geq \lambda_{i+1}$ and $\lambda_1 = \lambda$.

In Figure 3 the numerically determined Lyapunov exponent λ_{exp} (for $N = 150$), the largest Lyapunov exponent λ as well as the second largest one λ_2 (for $m \neq 1$) obtained by branching process theory are plotted as a function of the weight STD σ for $K = 24$ and three different state resolutions $m = 1, 3, 6$. It can be observed that as long as $\lambda_2 < 0$, the branching process exponent λ predicts well the numerically determined exponent λ_{exp} . For

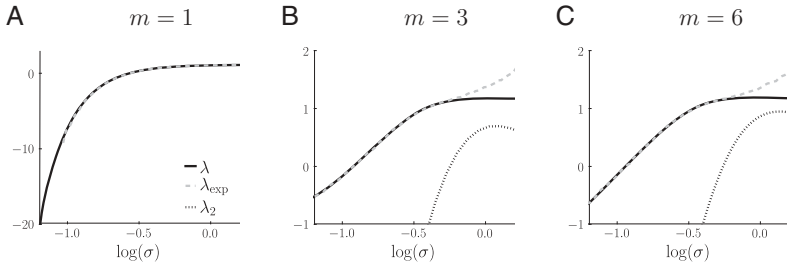


Figure 3: The numerically determined Lyapunov exponent λ_{exp} (for $N = 150$), the largest and second-largest Lyapunov exponents λ and λ_2 (for $m \neq 1$) obtained by branching process theory, are plotted as a function of the weight scale σ for $K = 24$ and three different state resolutions: $m = 1, 3, 6$. If $\lambda_2 < 0$, λ and λ_{exp} agree well, whereas λ_{exp} is larger than λ for weight scales σ with $\lambda_2 > 0$. This can be explained by the fact that λ_{exp} measures the total rate of perturbation growth, which is governed by all positive Lyapunov exponents, in particular by λ and λ_2 .

$\lambda_2 > 0$, λ_{exp} is larger than λ , as in this case, λ_2 also contributes to the total growth rate of the number of perturbations, which is numerically estimated by λ_{exp} . Hence it can be concluded that the Lyapunov spectrum determined by branching process theory using the AA characterizes the perturbation dynamics in the case of finite-size systems quite accurately.

3 Online Computations with Quantized ESNs

In this section we numerically investigate the capabilities of the networks defined in section 2 for online computations on the binary input sequence u using the RC approach; the networks are augmented by a trainable readout device, which in our context is a simple linear classifier. In this letter we consider tasks where the binary target output at time t depends solely on n input bits in the recent past, that is, on the n input bits $u(t - \tau - 1), \dots, u(t - \tau - n)$ for given $n \geq 1$ and delay parameter $\tau \geq 0$. More precisely, the target output is given by $f_T(u(t - \tau - 1), \dots, u(t - \tau - n))$ for a function $f_T \in \{f | f : \{-1, 1\}^n \rightarrow \{-1, 1\}\}$. In order to approximate the target output at time t , a linear classifier with the output $\text{sign}(\sum_{i=1}^N \alpha_i x_i(t) + b)$ at time t is applied to the instantaneous network state $\mathbf{x}(t)$. The coefficients α_i and the bias b were trained via a pseudo-inverse regression method (see Jaeger, 2001). The RC system consisting of a network defined by equation 2.1 with parameters m, K, σ and a linear classifier is called a quantized ESN (qESN) of state resolution m in the remainder of this letter.

We assessed the computational capabilities of a given network C based on the numerically determined performance on an example task, which was chosen to be the τ -delayed parity function of n bits $\text{PAR}_{n,\tau}$, that is, the

desired output at time t is $\text{PAR}_{n,\tau}(u, t) = \prod_{i=1}^n u(t - \tau - i)$ for a delay $\tau \geq 0$ and $n \geq 1$. A separate readout classifier is trained for each combination of n and τ , all using the same network C as reservoir. We define p_{exp} , which quantifies the experimentally determined computational performance of a given circuit C on the PAR_n task, as

$$p_{\text{exp}}(C, \text{PAR}_n) := \sum_{\tau=0}^{\infty} \kappa(C, \text{PAR}_{n,\tau}). \quad (3.1)$$

Here $\kappa(C, \text{PAR}_{n,\tau}) \in [0, 1]$ denotes the performance of circuit C on the $\text{PAR}_{n,\tau}$ task measured in terms of Cohen's kappa coefficient (where 0 corresponds to chance and 1 to optimal performance).² To facilitate intuition, in Figure 13 in appendix B, the dependence of Cohen's kappa coefficient on the delay τ and the network size N is illustrated for two different parameter settings. According to its definition, the performance measure p_{exp} sums up the kappa coefficients for all delays τ . For example, a network C that operates optimally for delays $\tau = 0, \dots, 2$ on a given TASK_n and at chance level for other delays will have $p_{\text{exp}}(C, \text{TASK}_n) = 3$. Extensive numerical experiments indicate that the performance results for PAR_n can be considered quite representative of the general computational capabilities of a circuit C of the considered type, as qualitatively very similar results were obtained for numerous classification tasks with two classes as well as for $p_{\text{exp}}(C, \text{RAND}_n)$ denoting the performance averaged over 50 randomly chosen functions f_T of n bits.

In Figure 4 the performances $p_{\text{exp}}(C, \text{PAR}_3)$, $p_{\text{exp}}(C, \text{PAR}_5)$ averaged over 20 circuits C , and $p_{\text{exp}}(C, \text{RAND}_5)$ averaged over 50 circuits C , and random tasks for three different state resolutions $m = 1, 3, 6$ are shown. The results are plotted as functions of the network in-degree K and the logarithm of the weight STD σ .³ Qualitatively very similar results were obtained for network graphs with binomial or scale-free distributed in-degree with average K (results not shown). The critical line (i.e., the location of the order-chaos phase transition) is also shown (dashed black line), which was determined by the root of the largest Lyapunov exponent λ given by the branching process approach outlined in the previous section. Further, the root of the second largest Lyapunov exponent λ_2 is plotted (dashed white line). The critical line predicts the zone of optimal performance well for $m = 1$ but is less accurate for ESNs with $m = 3, 6$. The root of λ_2 gives a quite reliable upper bound for the weight STD σ , that is, all networks with a σ for which $\lambda_2 > 0$ are too chaotic to exhibit good performance measures p_{exp} . One can

² κ is defined as $(c - c_l)/(1 - c_l)$ where c is the fraction of correct trials and c_l is the chance level. The sum in equation 3.1 was truncated at $\tau = 15$, as the performance was negligible for higher delays $\tau > 15$ for the network size $N = 150$.

³All logarithms are taken to the basis 10 ($\log = \log_{10}$) if not stated otherwise.

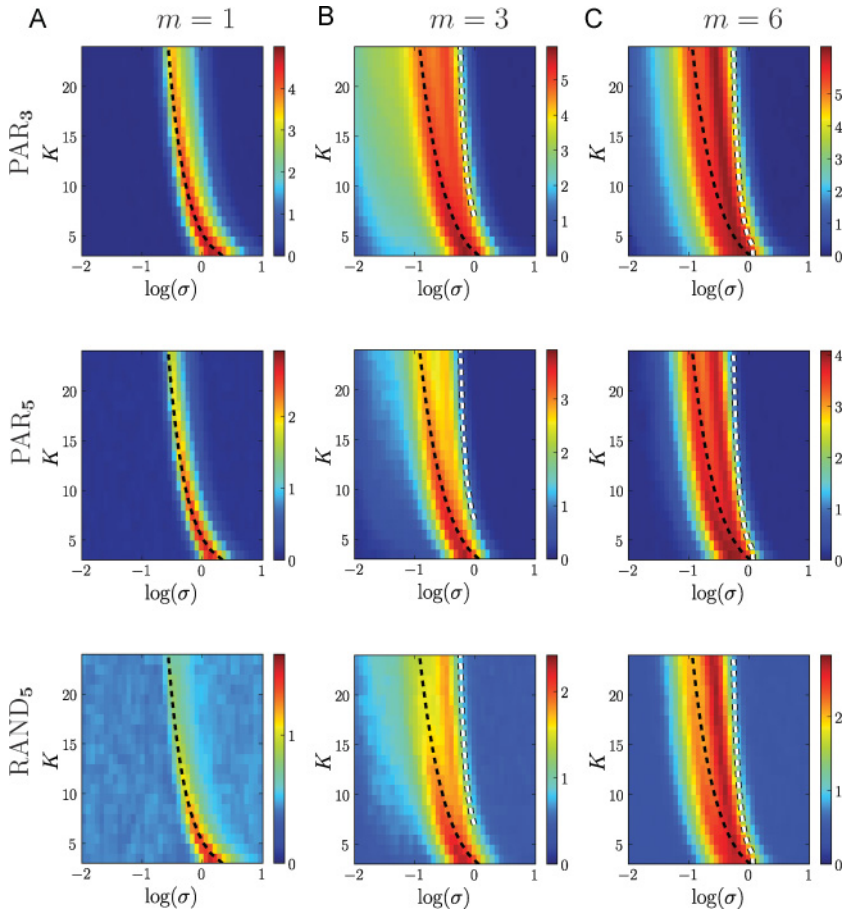


Figure 4: The performance $p_{\text{exp}}(C, \text{PAR}_3)$ (top row), $p_{\text{exp}}(C, \text{PAR}_5)$ (middle row), and $p_{\text{exp}}(C, \text{RAND}_5)$ (bottom row) for three different state resolutions $m = 1$ (column A), $m = 3$ (column B), and $m = 6$ (column C) is plotted as a function of the network in-degree K and the weight STD σ . Further, the zero crossing of the largest Lyapunov exponent λ (the critical line, black dashed line) as well as of the second largest one λ_2 (for $m \neq 1$, white dashed line) are shown. The network size is $N = 150$, the results $p_{\text{exp}}(C, \text{PAR}_5)$ have been averaged over 20 circuits C , initial conditions, and randomly drawn input time series of length 10^4 time steps. For $p_{\text{exp}}(C, \text{RAND}_5)$, results have been averaged over 50 random tasks of 5 bit, circuits C , initial conditions, and randomly drawn input time series of length 10^4 time steps.

see that for ESNs with low state resolutions ($m = 1, 3$), networks with a small in-degree K reach a significantly better peak performance than those with high in-degree. The effect disappears for a high state resolution ($m = 6$). This phenomenon is consistent with the observation that the network connectivity structure is in general an important issue if the reservoir is composed of binary or spiking neurons but less important if analog neurons are employed. The performance landscapes for analog networks ($m = \infty$) exhibit qualitatively the same key feature as the ones of high-resolution networks (high performance can be found for all in-degrees by scaling σ) but differ quantitatively from the latter (the region of high performance is generally broader; results not shown). Note that for $m = 3, 6$, we see a bifurcation in the zones of optimal performance that is not observed for the limiting cases of ESNs and LSMs.

4 Phase Transitions in Quantized ESNs

In this section, we examine the phase transition between ordered and chaotic dynamics in quantized ESNs in order to explain the difference between binary and high-resolution reservoirs shown in Figure 4. It was often hypothesized that systems with high computational power in recurrent networks are located in a parameter regime near the critical line, that is, near the phase transition between ordered and chaotic behavior (see, e.g., Legenstein & Maass, 2007b, for a review; compare also the performance with the critical line in Figure 4). Starting from this hypothesis, we investigated whether the network dynamics of binary networks near this transition differs qualitatively from the one of high-resolution networks. We analyzed the network properties by considering the Lyapunov exponent λ approximated by the branching process approach introduced above. However, we not only determined the critical line (i.e., the parameter values where the estimated Lyapunov exponent crosses zero), but also considered its values nearby. For a given in-degree K , λ can then be plotted as a function of the STD of weights σ (centered at the critical value σ_0 of the STD for that K). This was done for binary ($m = 1$; see Figure 5A) and high-resolution networks ($m = 6$; see Figure 5B) with in-degrees $K = 3, 12$, and 24. Interestingly, the dependence of λ on the STD σ near the critical line is qualitatively quite different between the two types of networks. For binary networks, the transition becomes much sharper with increasing in-degree K , which is not the case for high-resolution networks. These observations are confirmed by investigation of the numerically determined Lyapunov exponent λ_{exp} (plotted as dotted lines in Figure 5), which agrees accurately with λ in the considered parameter regime.

How can this sharp transition between ordered and chaotic dynamics of binary ESNs with high in-degree K explain their reduced computational performance? The tasks considered in this letter require some limited amount of memory, which has to be provided by the reservoir. Hence,

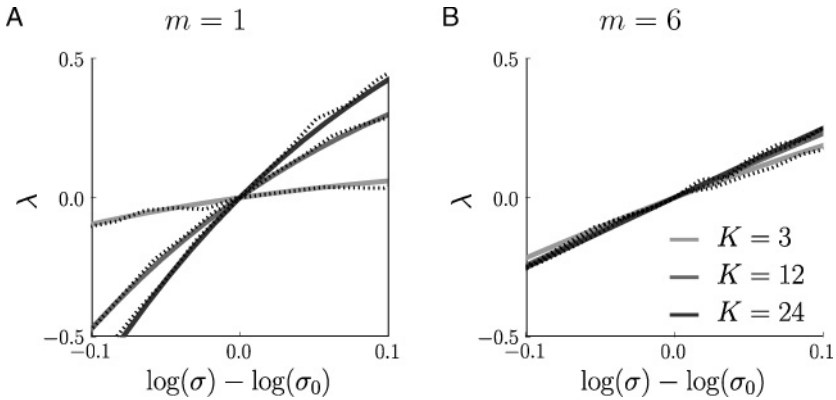


Figure 5: Phase transitions in binary networks ($m = 1$) differ from phase transitions in high-resolution networks ($m = 6$). The branching process approximation λ of the largest Lyapunov exponent is plotted as a function of the STD of weights σ for in-degrees $K = 3$ (light gray), $K = 12$ (gray), and $K = 24$ (dark gray). Further, the corresponding finite-size estimations λ_{exp} (evaluated for $N = 150$) are shown (dotted black). In order to facilitate comparison, the plot for each K is centered around $\log(\sigma_0)$, where σ_0 is the STD of weights for which λ is zero (i.e., σ_0 is the estimated critical σ value for that K). The transition sharpens with increasing K for binary reservoirs (A), whereas it is virtually independent of K for high-resolution reservoirs (B).

the network dynamics has to be located in a regime where memory about recent inputs is available and past input bits do not interfere with that memory. Intuitively, an effect of the sharper phase transition could be stated in the following way. For low σ (i.e., in the ordered regime), the memory needed for the task is not provided by the reservoir. With increasing σ , the memory capacity increases, but older memories interfere with recent ones, making it more difficult for the readout to extract the relevant information. This intuition is confirmed by an analysis introduced in Legenstein and Maass (2007a) that we applied to our setup. We estimated two measures of the reservoir, the so-called kernel quality and the generalization rank, both being the rank of a matrix consisting of certain state vectors of the reservoir. These two measures quantify two complementary properties of a reservoir with respect to the target function to be learned by the readout. For both measures, one defines N different input streams $u_1(\cdot), \dots, u_N(\cdot)$ and computes the rank of the $N \times N$ matrix, the state matrix, whose columns are the circuit states resulting from these input streams. The difference between the kernel quality and the generalization rank arises from the choice of the input streams. For the kernel quality, one chooses input streams that differ strongly with respect to the target function (e.g., streams that belong to different target classes). Since different input streams can be separated by

a readout only if they are represented by the reservoir in a diverse manner, it is desirable for the rank of the state matrix to be high in this case. For the generalization rank, one chooses similar input streams (again with respect to the target function). The rank of this state matrix should be small. The generalization rank can be related via the VC-dimension (Vapnik, 1998) to the ability of the reservoir-readout system to generalize from the training data to test data of the system (see Legenstein & Maass, 2007a, for details). In general, a high kernel quality and a low generalization rank (corresponding to a high ability of the network to generalize) are desirable. A network in the ordered regime will, however, have low values on both measures, while a chaotic network will have high values on both measures. By the use of these two separate measures, one can gain some insight into the different factors that determine the computational power of a reservoir system, as we will see below.

To evaluate the kernel quality of the reservoir, we randomly drew $N = 150$ input streams $u_1(\cdot), \dots, u_N(\cdot)$ and computed the rank of the $N \times N$ matrix whose columns were the circuit states resulting from these input streams.⁴ Intuitively, this rank measures how well the reservoir represents different input streams. The generalization rank was evaluated as follows. We randomly drew N input streams $\tilde{u}_1(\cdot), \dots, \tilde{u}_N(\cdot)$ such that the last three input bits in all of these input streams were identical.⁵ The generalization rank is then given by the rank of the $N \times N$ matrix whose columns are the circuit states resulting from these input streams. Intuitively, the generalization rank with this input distribution measures how strongly the reservoir state at time t is sensitive to inputs older than three time steps. The rank measures calculated in this way thus have predictive power for computations, which require memory of the last three time steps.

Figures 6A and 6D show the difference between the two measures as a function of $\log(\sigma)$ and K for binary networks and high-resolution networks, respectively. The plots show that the peak value of this difference is decreasing with K in binary networks, whereas it is independent of K in high-resolution reservoirs, reproducing the observations in the plots for the computational performance in Figure 4. A closer look at the binary circuit at $K = 3$ and $K = 24$ is given in Figures 6B and 6C. A comparison of these plots shows that the transition of both measures is much steeper for $K = 24$ than for $K = 3$, in agreement with the observed sharper phase transition illustrated in Figure 5, which leads to a smaller difference between

⁴The initial states of all neurons were i.i.d. uniformly over \mathcal{S}_m . The rank of the matrix was estimated by singular value decomposition on the network states after 15 time steps of simulation.

⁵First, we drew each of the last three bits $\tilde{u}(13), \dots, \tilde{u}(15)$ independently from a uniform distribution over $\{-1, 1\}$. For each input stream $\tilde{u}_i(1), \dots, \tilde{u}_i(15)$, we drew $\tilde{u}_i(1), \dots, \tilde{u}_i(12)$ independently from a uniform distribution over $\{-1, 1\}$ and set $\tilde{u}_i(t) = \tilde{u}(t)$ for $t = 13, \dots, 15$.

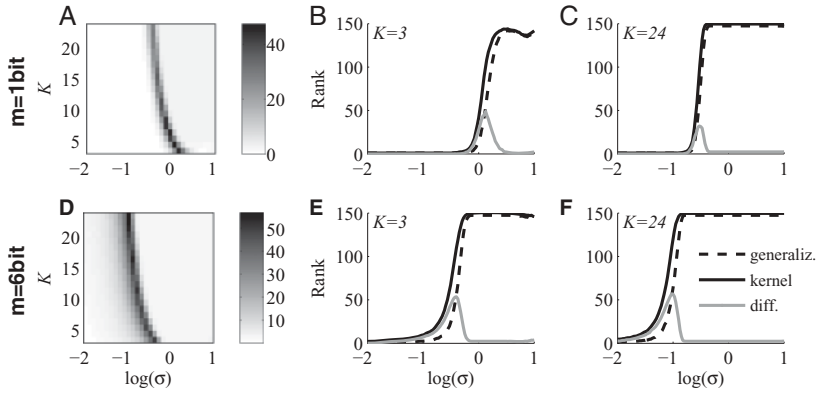


Figure 6: Kernel-quality and generalization rank of quantized ESNs of size $N = 150$. Upper plots are for binary reservoirs ($m = 1$), lower plots for high-resolution reservoirs ($m = 6$). (A) The difference between the kernel quality and the generalization rank as a function of the log STD of weights and the in-degree K . (B) The kernel quality (red), the generalization rank (blue), and the difference between both (black) for $K = 3$ as a function of $\log(\sigma)$. (C) Same as B, but for an in-degree of $K = 24$. In comparison to B, the transition of both measures is much steeper. (D, E, F) Same as A, B, and C, respectively, but for a high-resolution reservoir. All plotted values are means over 100 independent runs with randomly drawn networks, initial states, and input streams.

the measures. We interpret this finding in the following way. For $K = 24$, the reservoir increases its separation power very fast as $\log(\sigma)$ increases. However, the separation of past input differences increases likewise, and thus early input differences cannot be distinguished from later ones. This reduces the computational power of binary ESNs with large K on the considered tasks. In comparison, the corresponding plots for high-resolution reservoirs (see Figures 6E and 6F) show that the transition shifts to lower-weight STDs σ for larger K , but apart from this fact, the transitions are virtually identical for low and high K values. Comparing Figure 6D with Figure 4C, one sees that the rank measure does not accurately predict the whole region of good performance for high-resolution reservoirs. It also does not predict the observed bifurcation in the zones of optimal performance, a phenomenon reproduced by the mean field predictor introduced in the following section.

5 Mean Field Predictor for Computational Performance

The question why and to what degree certain nonautonomous dynamical systems are useful devices for online computations has been addressed theoretically, among others, in Bertschinger and Natschläger (2004). There, the

computational performance of networks of randomly connected threshold gates was linked to their separation property (for a formal definition, see Maass et al., 2002). It was shown that only networks that exhibit sufficiently different network states for different instances of the input stream (i.e., networks that separate the input), can compute complex functions of the input stream. Furthermore, the authors introduced an accurate predictor of the computational capabilities for the considered type of binary networks based on the separation capability. The latter was numerically evaluated by a simple mean field approximation of the Hamming distance between different network states evolving from different input sequences.⁶

Here we aim at constructing a mean field predictor of computational performance for qESNs, extending the approach of Bertschinger and Natschläger (2004), which was viable only for binary networks. We use the term *predictor* of computational power to indicate a quantity that strongly correlates with experimental measures of computational power (e.g., p_{exp}) for varying parameters K and σ at fixed m . A predictor in the above sense can be used to efficiently identify the dependence of the computational power on the network parameters and gain theoretical insight into this dependence.

Instead of a straightforward generalization of the predictor presented in Bertschinger and Natschläger (2004), we make a somewhat different ansatz for two reasons. First, we wish to simplify the form of the mean field predictor. Second, a straightforward generalization turned out to be computationally too expensive for quantized ESNs with a state resolution $m > 1$. Therefore we introduce a modified mean field predictor, which can be computed more efficiently and still has all desirable properties of the one introduced in Bertschinger and Natschläger.

Suppose the target output of the network at time t is a function $f_T \in F = \{f | f : \{-1, 1\}^n \rightarrow \{-1, 1\}\}$ of the n bits $u(t - \tau - 1), \dots, u(t - \tau - n)$ of the input sequence u with delay τ as described in section 3. In order to exhibit good performance on an arbitrary $f_T \in F$, pairs of inputs that differ in at least one of the n bits have to be mapped by the network to different states at time t . Only then will the linear classifier be able to assign the inputs to different classes (function values). In order to quantify this so-called separation property of a given network, we introduce the normalized distance $d(k)$. It measures the average distance between two network states $\mathbf{x}^1(t) = (x_1^1(t), \dots, x_N^1(t))$ and $\mathbf{x}^2(t) = (x_1^2(t), \dots, x_N^2(t))$ arising from applying to the same network two input sequences u^1 and u^2 that differ only in the single bit at time $t - k$, that is, $u^1(t - k) = -u^2(t - k)$ and $u^1(\tau) = u^2(\tau)$ for

⁶The theoretical approach presented in Bertschinger and Natschläger (2004) is not a mean field theory in the strict sense of physics literature. However, due to the AA used in Bertschinger and Natschläger, all network units receive recurrent inputs that are drawn (independently) from the *same* distribution. This is why we adopt the term *mean field* and apply it to our theoretical considerations.

all $\tau \neq t - k$. Formally we define the k -step input separation $d(k)$:⁷

$$d(k) = \frac{1}{N} \langle \|\mathbf{x}^1(t) - \mathbf{x}^2(t)\|_1 \rangle_{C, u^1}. \quad (5.1)$$

The average $\langle \cdot \rangle_{C, u^1}$ is taken over all inputs u^1 (the input u^2 is simply a function of u^1), all initial conditions of the network, and all circuits C with given network parameters N, m, σ, K . However, a good separation of the n relevant bits— $d(k) \gg 0$ for $\tau < k \leq n + \tau$ —is a necessary but not sufficient condition for the ability of the network to calculate the target function. Beyond this, it is desired that the network “forgets” all (for the target function) irrelevant bits $u(t - k)$, $k > n + \tau$ of the input sufficiently quickly: $d(k) \approx 0$ for $k > n + \tau$. We use the limit $d(\infty) = \lim_{k \rightarrow \infty} d(k)$ to quantify this irrelevant separation, which can be considered as noise with regard to the target function f_T . Hence, we propose the quantity p_∞ as a heuristic predictor for computational power:

$$p_\infty = \max\{d(2) - d(\infty), 0\}. \quad (5.2)$$

As the first contribution to p_∞ , we chose $d(2)$ as it reflects the ability of the network to perform a combination of two mechanisms. In order to exhibit a high value for $d(2)$, the network has to separate the inputs at time step $t - 2$ and sustain the resulting state distance via its recurrent dynamics in the next time step, $t - 1$. We therefore consider $d(2)$ to be a measure for input separation on short timescales relevant for the target function.

The quantity p_∞ is calculated using a mean field model similar to the one presented in Bertschinger and Natschläger (2004), which itself is rooted in the AA; the latter was described briefly in section 2. In the AA and with $N \rightarrow \infty$, all components of the difference $\mathbf{x}^1(t) - \mathbf{x}^2(t)$ appearing in equation 5.1 are i.i.d. Hence it is sufficient to determine the joint probability of a single network unit to be in state $s_i \in \mathcal{S}_m$ in the network receiving input u^1 and being in state $s_j \in \mathcal{S}_m$ in the network receiving input u^2 . This probability is denoted as $q_{ij}(t, u^1, u^2)$. The diagonal elements $i = j$ of $q_{ij}(t, u^1, u^2)$ quantify the probability that the state of a unit is not affected by the difference in inputs u^1 and u^2 , whereas the off-diagonal elements $i \neq j$ quantify the probability that the state s_i of a unit is “flipped” to state s_j due to the different inputs u^1 and u^2 . The separation $d(k)$ can be computed as

$$d(k) = \sum_{i, j=0}^{2^m - 1} q_{ij}(k, u^1, u^2) |s_j - s_i|. \quad (5.3)$$

The probabilities $q_{ij}(k, u^1, u^2)$ can be calculated iteratively using $q_{ij}(k - 1, u^1, u^2)$ as a distribution of the recurrent inputs at time step k , analogous

⁷For vectors $\mathbf{x} = (x_1, x_2, \dots) \in \mathbb{R}^N$ we use the Manhattan norm $\|\mathbf{x}\|_1 := \sum_{i=1}^N |x_i|$.

to the method presented in Bertschinger and Natschläger (2004). For high-resolution networks, however, there are 2^{2m} different elements $q_{ij}(t, u^1, u^2)$. In order to avoid this “combinatorial explosion,” we introduce an approximation that we term separation approximation (SA). Consider a network unit in the state s_i . The state s_i can be uniquely represented by the binary tuple $(B_0(s_i), \dots, B_{m-1}(s_i)) \in \{0, 1\}^m$, where $B_0(s_i)$ is the most and $B_{m-1}(s_i)$ the least significant bit of this binary representation. We can then define the probability $q_{\alpha, \beta}^l(k, u^1, u^2)$ for $\alpha, \beta \in \{0, 1\}$ as the probability of the l th bit B_l to be in state α in the network receiving input u^1 and $B_l = \beta$ in the network receiving input u^2 . The SA consists in assuming the probability of $B_l(s_i)$ to be independent from the ones of $B_n(s_i)$ for $n \neq l$:

$$q_{ij}(k, u^1, u^2) \approx \prod_{l=0}^{m-1} q_{B_l(s_i), B_l(s_j)}^l(k, u^1, u^2).$$

This approximation neglects the statistical dependencies between bits $B_l(s_i)$ and $B_n(s_i)$ for $n \neq l$ for the sake of computational efficiency. Trivially the SA is exact for binary reservoirs. Numerical results suggest that for small and intermediate state resolutions $m \leq 5$, the SA is still quite accurate, whereas for large $m \geq 5$, deviations from full simulations of the network are clearly visible. All further details of the calculation of p_∞ can be found in appendix C.

It is worth noticing that in the AA, as the weights w_{ij} are symmetric the following relation holds:⁸

$$q_{ij}(t, u^1, u^2) = q_{ij}(t, \hat{u}^1, \hat{u}^2),$$

where \hat{u}^1 and \hat{u}^2 are sequences with $\hat{u}^1(\tau)\hat{u}^2(\tau) = u^1(\tau)u^2(\tau)$ for all $\tau \in \mathbb{Z}$. Hence, flipping input bits in both input sequences u^1 and u^2 at the same time steps leaves the input separation $d(k)$ unaltered in the AA. Therefore, in the AA, $d(k)$ can be determined with a single sample sequence u^1 without the need for averaging over different inputs as indicated in equation 5.3, where the average $\langle \cdot \rangle_{u^1}$ does not appear. This constitutes the main advantages of p_∞ over the the predictor defined in Bertschinger and Natschläger (2004), the so-called NM separation. The NM separation requires averaging the mean field separation measure over input sequences, resulting in a significantly larger computation time. Furthermore, the NM separation is determined by three contributions, whereas p_∞ contains only two terms, $d(2)$ and $d(\infty)$, making the latter more simple and intuitive.

In Figure 7 two examples of the evolution of $d(k)$ for state resolutions $m = 1, 3, 6$ with the parameters $\log(\sigma) = -0.45$ and $K = 3, 24$ are shown. With this choice of σ , the network with in-degree $K = 3$ is in the ordered regime for all state resolutions $m \in \{1, 3, 6\}$, and the network with in-degree

⁸We call a random variable z symmetric if and only if $p(z) = p(-z)$.

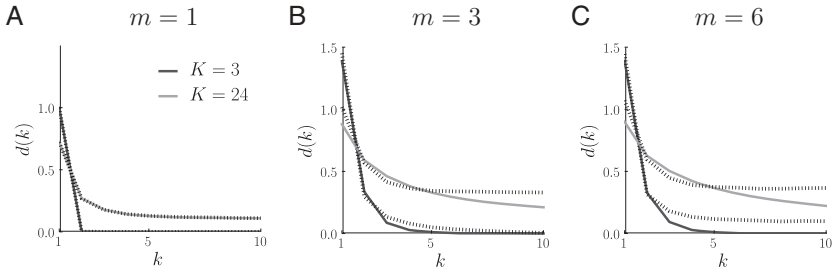


Figure 7: The k -step input separation measure $d(k)$ defined in equation 5.1 for networks with $N = 150$ units and $\log \sigma = -0.45$ with in-degree $K = 3$ (dark gray) corresponding to ordered dynamics and $K = 24$ (light gray) corresponding to chaotic dynamics determined by explicit simulations of the networks. The mean field approximation of $d(k)$ is plotted as a dotted line showing good agreement for low state resolutions $m = 1, 3$ (A,B) and larger deviations for high state resolutions ($m = 6$, C).

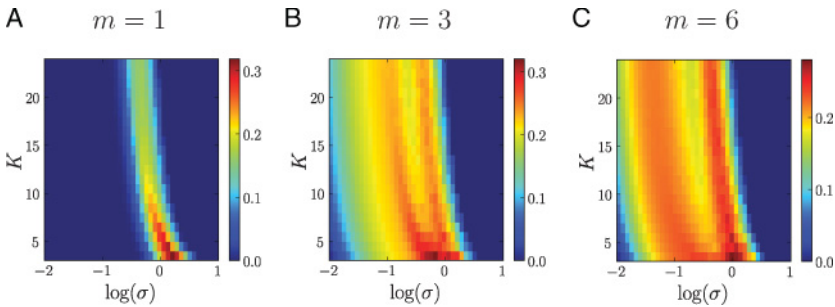


Figure 8: Mean field predictor p_∞ for computational power for different state resolutions $m = 1$ (A), $m = 3$ (B), and $m = 6$ (C) as a function of the STD σ of the weights and in-degree K . Compare this result to the numerically determined performance p_{exp} plotted in Figure 4.

$K = 24$ is in the chaotic regime. The mean field approximations of $d(k)$ are in quite good agreement for $m = 1, 3$, with the values for $d(k)$ determined by explicitly simulating the networks. For high-resolution networks (here $m = 6$), visible errors occur due to the AA and the SA.

In Figure 8 the predictor p_∞ is plotted as a function of the weight STD σ and the in-degree K for three different values of the state resolution $m \in \{1, 3, 6\}$. When comparing these results with the actual network performance p_{exp} plotted in Figure 4, one can see that p_∞ serves as a reliable predictor for p_{exp} of a network for sufficiently small m . For larger values of m , the predictor p_∞ starts to deviate from the true performance while still capturing the interesting features of the performance landscape

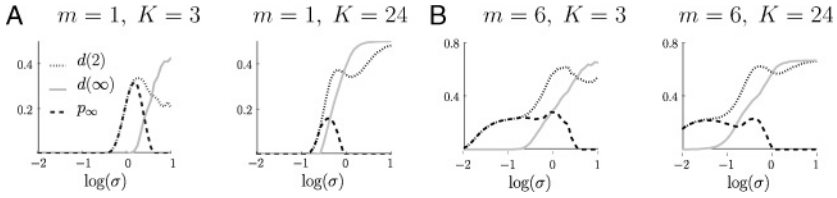


Figure 9: The contributions $d(2)$ (dotted line), $d(\infty)$ (light gray) of the mean field predictor p_∞ and their difference (dashed black line) for different state resolutions $m = 1, 6$ as a function σ . The plots show slices of the 2D plots in Figures 8A and 8C for constant $K = 3, 24$. (A) For $m = 1$ the region in $\log(\sigma)$ -space with high $d(2)$ and low $d(\infty)$ is significantly larger for $K = 3$ than for $K = 24$. (B) For $m = 6$, this region is roughly independent of K except a shift toward lower σ -values.

qualitatively. The dominant effect of the state resolution m on the performance discussed in section 3 is well reproduced by p_∞ . For $m = 1$ the in-degree K has a considerable impact: for large K , maximum performance drops significantly. For high state resolutions, however, there exists a region in the parameter space exhibiting high performance for all values of K .

The interplay between the two contributions $d(2)$ and $d(\infty)$ of p_∞ delivers insight into the dependence of the computational performance on the network parameters. A high value of $d(2)$ corresponds to a good separation of inputs on short timescales relevant for the target task, a property found predominantly in networks that are not strongly input driven—networks with relatively strong recurrent connection (large weight STD σ). A small value of $d(\infty)$ is a necessary condition for different inputs on which the target function assumes the same value to be mapped to nearby network states. Only then is a linear readout able to assign them to the same class regardless of their irrelevant remote history. This condition is met for small weight STD σ . For $m = 1$, as can be seen in Figure 9, the region in $\log(\sigma)$ space where both conditions for good performance are present, the region of intermediate σ , decreases for growing K . In contrast, for $m > 2$, a reverse effect is observed: for increasing K , the parameter range for σ fulfilling the two opposing conditions for good performance grows moderately, resulting in a large region of high p_∞ for high in-degree K . This observation is in close analogy to the behavior of the rank measure discussed in section 4. Also note that p_∞ predicts the novel bifurcation effect also observed in Figure 4.

6 An Annealed Approximation of the Memory Function for Binary qESNs

A quantity that has extensively been studied (White et al., 2004; Mayor & Gerstner, 2005) in order to characterize the ability of a given network to

store information is the memory function defined in Jaeger, 2002; see also Ganguli et al., 2008, for a novel alternative definition. In this section we show that the memory function for qESNs is tightly linked to the k -step separation $d(k)$ defined in equation 5.1. More precisely, the separation $d(k)$ can be used to formulate an upper bound on the memory function. For binary ($m = 1$) qESNs in the ordered regime, this upper bound turns out to be very close to the true memory function while being computationally cheap to evaluate, especially for networks with large system size N .

The memory function $m(k)$ defined in Jaeger (2002), which assumes values in $[0, 1]$, measures the ability of a network in conjunction with a linear readout to reconstruct the input signal $u(t - k)$ that was presented k time steps ago, where $m(k) = 1$ corresponds to a perfect reconstruction and $m(k) = 0$ to a readout output that is uncorrelated with the input $u(t - k)$. More precisely, the memory function is defined as

$$m(k) := \frac{\text{cov}(y(t), y_T(t))^2}{\text{var}(y(t))\text{var}(y_T(t))}. \quad (6.1)$$

Here $\text{var}(\cdot)$ denotes the variance, and $\text{cov}(\cdot, \cdot)$ denotes the covariance of the arguments. The quantity $y(t) = (\sum_{i=1}^N \alpha_i x_i(t) + b)$ is the output of the linear readout at time t with weights $\alpha = (\alpha_1, \dots, \alpha_N)$ and bias b and $y_T(t) = u(t - k)$ is the target output. The weights and the bias are learned by linear regression. According to definition 6.1, the memory function measures the overlap between the readout output $y(t)$ and the target output $y_T(t)$. The memory function $m(k)$ is often numerically evaluated from the identity (see Jaeger, 2002; White et al., 2004)

$$m(k) = p_k^T A^{-1} p_k. \quad (6.2)$$

The matrix A with elements $A_{ij} = \text{cov}(x_i(t), x_j(t))$ denotes the covariance matrix of the network state, and $p_k = \text{cov}(x(t), y_T(t))$ is the covariance vector between the network state and the target output. For networks with linear (thus analog) units, many properties of the memory function can be characterized explicitly in terms of the connectivity matrix (see Jaeger, 2002; White et al., 2004). However, for networks of nonlinear units, little is known about the memory function. In general it has to be determined numerically by evaluating equation 6.2, which requires simulating the full network in order to estimate A and p_k .

For the special case of a binary input sequence u with $p(u(t) = +1) = p(u(t) = -1)$, as assumed in this letter, the memory function can be bounded by using the k -step separation $d(k)$. The following relation is derived in appendix D:

$$m(k) \leq \min \left\{ \frac{N^2}{4} \|A^{-1}\|_2 d(k)^2, 1 \right\}, \quad (6.3)$$

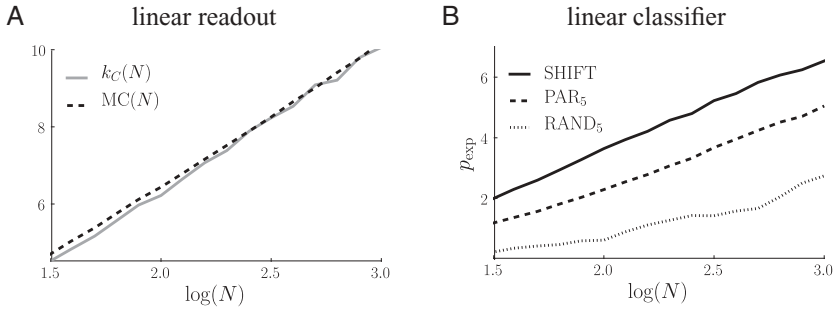


Figure 10: The computational capabilities of binary qESNs scale logarithmically with the network size N . The networks have the parameters $K = 3$ and $\log(\sigma) = 0.2$ and are thus close to the critical line. (A) Scaling of the temporal capacity $k_C(N)$ (light gray) and the memory capacity $MC(N)$ (dashed). According to the definition of the memory function, a linear readout was used to reconstruct the target signal. (B) Scaling of the experimental performance measures $p_{\text{exp}}(C, \text{PAR}_5)$ (dashed) and $p_{\text{exp}}(C, \text{RAND}_5)$ (dotted). Further, the performance $p_{\text{exp}}(C, \text{SHIFT})$ for the SHIFT task (solid line) is shown that consists of reconstructing the past input with a linear classifier. The experimental performance measures were determined as described in the caption of Figure 4.

where $\|\cdot\|_2$ is the operator norm induced by the standard Euclidean norm. The upper bound presented in the equation is striking, as it links the memory function $m(k)$ with the dynamical property of k -step input separation $d(k)$, allowing us to draw conclusions about $m(k)$ from the behavior of $d(k)$. As observed in numerical simulations, $d(k)$ approximately decays exponentially⁹ in the ordered regime, implying also that $m(k)$ decays (at least) exponentially with a time constant that is half as large as the one for $d(k)$. This consideration also results in an upper bound for the scaling of the temporal capacity $k_C(N)$ with the network size N . In White et al. (2004), $k_C(N)$ is defined as the smallest $k_0 \in \mathbb{N}$ such that for all $k > k_0$, the memory function of a network of size N is smaller than $1/2$, that is, $m(k) < 1/2$. As can be easily seen from equation 6.3, $k_C(N) = \mathcal{O}(\log(N))$ given that $d(k)$ decays exponentially. Hence, the temporal capacity of all qESN networks in the ordered regime grows only logarithmically in contrast to, for example, linear networks with orthogonal connectivity matrices that exhibit an extensive growth $k_C(N) \propto N$ as shown in White et al. (2004). Another quantity of interest characterizing the capabilities of a circuit to store information is the memory capacity $MC(N)$, which is defined as $MC(N) := \sum_{k=1}^{\infty} m(k)$ (see Jaeger, 2002). In Figure 10A, $k_C(N)$ and $MC(N)$ for $m = 1$ networks at

⁹It is intuitive to conjecture that $d(k)$ should decay like $\exp(\lambda k)$, where λ is the Lyapunov exponent. However, this is not trivial to show. Further investigation is required that addresses this interesting point.

the critical line are shown to exhibit a clearly logarithmic growth with N over 1.5 decades. In Figure 10B, the performance measure p_{exp} is plotted for the three different tasks PAR₅, RAND₅, and SHIFT as a function of N , which also show logarithmic scaling with the system size N .¹⁰ In the chaotic parameter regime, $d(k)$ decays toward a nonzero baseline, and therefore inequality 6.3 will not yield a useful upper bound on $m(k)$ as the latter always decays toward zero as numerically observed.

Inequality 6.3 can also be used to numerically estimate an upper bound for the memory function using the AA. This upper bound is computationally very efficient, as its complexity is independent of the system size N , and it turns out to be close to the true memory function. In the AA, one can easily derive an expression for the term $\|A^{-1}\|_2$ as a function of the network connectivity parameters K and σ (see section D.2):

$$\|A^{-1}\|_2 = 4 \left(1 - \left(\Phi \left(\frac{2}{K^{1/2}\sigma} \right) - \Phi \left(-\frac{2}{K^{1/2}\sigma} \right) \right)^2 \right)^{-1}. \quad (6.4)$$

Here $\Phi(x)$ denotes the cumulative probability distribution of a random variable distributed normally with unit variance. Combining equations 6.3 and 6.4, one can evaluate an upper bound for the memory function assuming that the AA is valid. However the accuracy of the mean field approximation for $d(k)$ is of sufficient accuracy only for $m = 1$; hence, the memory bound 6.3 is of practical use only for binary qESNs in the ordered regime. Three examples for $m(k)$ and for the upper bound 6.3 of binary qESNs with $N = 1000$ units with different parameter settings in the ordered regime are shown in Figure 11. As can be seen, the distance between the memory function $m(k)$ and the upper bound (see equation 6.3) is varying with the weight STD σ and the in-degree K . It was numerically observed that the upper bound was in general closer to $m(k)$ for networks whose parameters σ , K are “deeper” in the ordered dynamic regime. As mentioned above, in the chaotic regime, the inequality 6.3 does not provide any sensible information on the memory function.

7 Sparse Network Activity and Computational Power _____

In the neocortex, the spiking (hence binary) neurons usually exhibit a high in-degree around 10^3 up to 10^4 (see DeFelipe & Fariñas, 1992; Destexhe, Rudolph, & Paré, 2003). Assuming the hypothesis that cortical circuits can be regarded (at least partially) as RC devices, the high in-degrees observed experimentally are in stark contrast to the findings described in the previous

¹⁰The target output for the SHIFT _{τ} task is defined as SHIFT _{τ} (u, t) = $u(t - \tau - 1)$, and the performance is defined as $p_{\text{exp}}(\text{C}, \text{SHIFT}) = \sum_{\tau=0}^{\infty} \kappa(\text{C}, \text{SHIFT}_{\tau})$. In contrast to the memory function, a linear classifier is used to reconstruct the target signal.

A $\log(\sigma) = 0.0, K = 3$ **B** $\log(\sigma) = -0.5, K = 10$ **C** $\log(\sigma) = -0.6, K = 20$

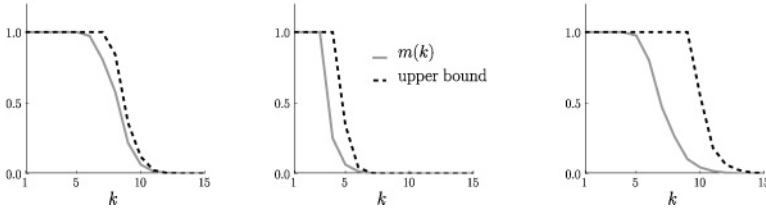


Figure 11: The memory function for three binary qESNs with $N = 1000$ units evaluated according to equation 6.2 is shown (light gray). Further, the upper bound given by equation 6.3 with the AA expression (see equation 6.4) for $\|A^{-1}\|_2$ is plotted (dashed). The qESNs were generated with the parameters $\log(\sigma) = 0.0, K = 3$ (A), $\log(\sigma) = -0.5, K = 10$ (B) and $\log(\sigma) = -0.6, K = 20$ (C). As can be seen, the distance between $m(k)$ and the upper bound varies depending on the network parameters. In general, the more “ordered” the network dynamics are, the “tighter” the upper bound (see equation 6.3) gets.

sections. As discussed above, we would expect reservoirs consisting of binary units to be of low average in-degree as computational performance in the RC sense is best in this parameter regime. In the following section, we show that this apparent inconsistency can be resolved by introducing sparse network activity into the binary qESN model.

A characteristic property of the neural activity in the neocortex is that spikes are scarce events in time. Assuming that the refractory period of cortical neurons is in the millisecond range, they could in principle emit spikes with a frequency of 100 Hz and more. However, the observed average firing rate of a cortical neuron is well below 10 Hz. It has often been suggested that this sparse activity is due to metabolic cost and energy constrains (see Levy & Baxter, 1996; Laughlin, de Ruyter van Steveninck, & Anderson, 1998; Lennie, 2003). The binary qESNs introduced above, however, are symmetric in the states $+1/2$ and $-1/2$, yielding equal probabilities to be in these two states. In order to mimic sparse network activity, we augment the input $u(t)$ in the state update equation, 2.1, with a bias $b < 0$; we replace $u(t)$ by $u(t) + b$, which leads to a preferred “resting” state $-1/2$ and a less frequent “spiking” state $+1/2$. The probability for a unit to assume the value $1/2$ (to emit a “spike”) can be evaluated in the AA to

$$p_+ = p(x_i(t) = 1/2) = 1 - \frac{1}{2} \left(\Phi \left(\frac{2(-b-1)}{K^{1/2}\sigma} \right) + \Phi \left(\frac{2(-b+1)}{K^{1/2}\sigma} \right) \right). \quad (7.1)$$

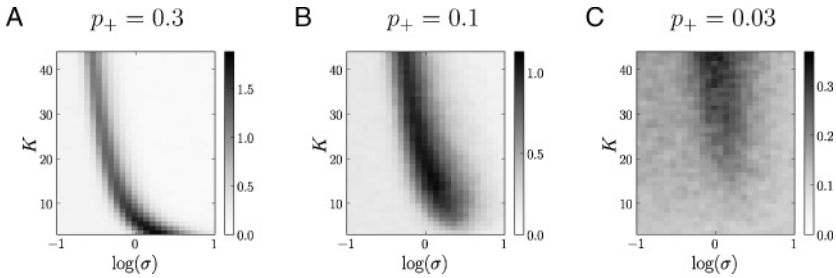


Figure 12: The peak computational performance of networks with sparser network activity is observed at increasing connectivity. Shown is the computational performance $p_{\text{exp}}(C, \text{PAR}_5)$ averaged over 20 random circuits C as a function of weight standard deviation σ and in-degree K for binary qESNs with sparse network activity $p_+ = 0.3$ (A), $p_+ = 0.1$ (B), and $p_+ = 0.03$ (C) caused by a bias.

In order to illustrate the impact of sparse activity on the computational performance, we compare the measure p_{exp} of networks with different parameters σ and K at a given constant sparse activity p_+ . To do so, equation 7.1 is numerically solved for b , yielding the required bias to achieve the given activity p_+ for a network with parameters σ and K . In Figure 12 the resulting computational performance measure $p_{\text{exp}}(C, \text{PAR}_5)$ of binary qESNs for the five-bit parity task PAR_5 is shown as a function of K and σ for three sparsity levels: $p_+ = 0.3$ (panel A), $p_+ = 0.1$ (panel B) and $p_+ = 0.03$ (panel C). By comparison with Figure 4 (where the activity is $p_+ = 0.5$), it can be observed that introducing a sufficiently sparse activity has a drastic effect on the computational performance landscape. In general computational performance decreases with increasing sparsity of the network activity. The most striking effect, however, is that the parameter region of maximum performance is shifting toward higher connectivity values with increasing sparsity of the network activity. Hence, networks with sparser activity require a higher in-degree in order to exhibit good computational capabilities.

Given the above result (sparsely connected networks need higher activity to “work well”), one might arrive at the intuitive hypothesis that for varying levels of connectivity and activity, the average input to a single network unit is constant for reservoirs with high performance: higher network activity can compensate for fewer input connections and vice versa. More precisely, one might argue that for different mean activities p_+ , the quantity $p_+ \cdot K_{\text{max}}(p_+)$ is constant, where $K_{\text{max}}(p_+) = \arg \max_K p_{\text{exp}}$ is the in-degree yielding the largest performance p_{exp} for the given activity p_+ . However, the results of the numerical experiments we performed (data not shown) clearly indicate that this is not the case. The product $p_+ \cdot K_{\text{max}}(p_+)$ varies strongly (by a factor of 1.6) for activities in the range $p_+ \in [0.1, 0.3]$.

Networks with sparse activity need a higher in-degree than one would expect from the above hypothesis. Hence the optimal working point (in the RC sense) of a network cannot be determined by solely considering the average input to a single network unit. This insight is of relevance for the simulation of cortical microcircuit models, where (due to limited computer resources) usually only low levels of connectivity are taken into account, compensated by a higher network activity. Our analysis for the qESN model suggests that this approach might considerably change the computational properties of the simulated circuits, indicating that the working point of these circuits needs to be carefully tuned, possibly by adapting further parameters.

Unfortunately, the numerical results for qESNs with sparse network activity outlined above cannot easily be reproduced by the mean field predictor p_∞ . Its numerical evaluation for networks with a nonvanishing bias is quite complex, as the bias destroys the symmetry of the update equation, 2.1, which is explicitly taken advantage of for the calculation of p_∞ . Without this symmetry, the evaluation of p_∞ requires averaging over input sequences $u(\cdot)$, which renders the computation very time-consuming.

Recapitulating, a possible explanation in the RC framework for the high in-degrees experimentally found in cortical microcircuits is sparse network activity, a ubiquitous feature observed in cortex.

8 Discussion

In this letter, we introduced the qESN model that interpolates between the binary LSM and the continuous ESN. This nonautonomous network model exhibits ordered or chaotic dynamics, separated by a phase transition, depending on the weight and connectivity parameters. These dynamical regimes are reminiscent of the ones observed in binary (e.g., Derrida & Stauffer, 1986) and in multistate (see Solé, Luque, & Kauffman, 2000) Kauffman networks. In agreement with previous results (Wolfram, 1984; Langton, 1990; Packard, 1988; Legenstein & Maass, 2007a; Bertschinger & Natschläger, 2004; but see Mitchell, Hraber, & Crutchfield, 1993) qENSs show optimal computational performance near the critical line: the order-chaos phase transition.

The qESN model for RC computations allowed the systematic investigation of a fundamental difference between LSMs and ESNs arising from the different nature of its reservoir units: the difference in the influence of the network connectivity onto the computational capabilities. Our results clearly show that for binary and low-resolution reservoirs, the network connectivity, parameterized here by the in-degree K , has a profound impact on the phase transition and hence also on the maximal computational performance. For sparse connectivity (small K), a gradual phase transition is found characterized by a small gradient of the Lyapunov exponent around its root, resulting in a high peak performance for a large region of the parameter space. For densely connected binary and low-resolution networks,

the phase transition becomes steep, indicating a reduced parameter region that exhibits desirable critical dynamics. This effect results in a significantly reduced computational performance of densely connected binary and low-resolution networks. The performance of high-resolution networks, however, does not exhibit this drastic dependence on the in-degree K . These numerical observations can be understood by analyzing rank measures, assessing the kernel and generalization properties, as well as by analyzing an annealed approximation of the input separation of a given network. In the light of these two theoretical approaches, the observed influence of connectivity on computational performance can be successfully explained in terms of separation of the input on short, task-relevant timescales versus separation on long, task-irrelevant timescales.

We emphasize that the experimental and numerical results indicating little influence of connectivity on computational performance for high-resolution qESNs are based on the assumption that there is no inherent spatial structure of the networks and of the input, a situation often occurring in typical machine learning applications of RC systems. In particular, we assumed in this study that all units receive the one-dimensional input $u(t)$, and the readout also gets input from all network units. If, however, the network exhibits a spatial structure (e.g., the set of network units receiving input and the set of units projecting to the readout are disjoint and “distant”), different connectivity schemes may well influence the performance of high-resolution qESNs and analog ESNs.

It should be noted that the effect of quantization cannot be emulated by additive or multiplicative i.i.d. or correlated gaussian noise on the output of analog neurons. The noise just degrades performance homogeneously, and the differences in the influence of the in-degree observed for varying state resolutions cannot be reproduced. Thus, this type of noise is qualitatively different from the so-called quantization noise introduced by discretizing the state space of the network units.

The results presented in this letter that emphasize the importance of a gradual order-chaos phase transition offer a possible explanation of why ESNs are more often used in engineering applications than LSMs. Although these two RC systems can have a very similar performance on a given task (Verstraeten et al., 2007), it is significantly harder to create an LSM consisting of spiking neurons operating in a desirable dynamic regime (i.e., at the edge of chaos). In order to archive this, the excitation and inhibition in the network need to be finely balanced to avoid quiescent or epileptic activity. For ESNs, this is not the case; there is usually a broad parameter range in which the ESN performs well. This difference reveals the need especially regarding LSMs for homeostatic control mechanisms or unsupervised learning rules that bring and keep a dynamic reservoir in a close-to-optimal working regime, replacing (possibly suboptimal) ad hoc parameter settings. This kind of unsupervised dynamic reservoir optimization has become quite standard for ESNs (Triesch, 2007; Steil, 2007; Schrauwen,

Wardermann, Verstraeten, Steil, & Stroobandt, 2008; see Lukoševičius & Jaeger, 2007, for a review); for LSMs, interesting steps in this direction have been undertaken, among others, in Natschläger, Bertschinger, and Legenstein (2005), Lazar, Pippa, and Triesch (2007), Joshi and Triesch (2008).

We have shown that the k -step separation $d(k)$ can also be used to efficiently compute an upper bound for the memory function of binary networks with ordered dynamics. Previously only the memory function of linear networks was studied in detail (White et al., 2004; Jaeger, 2002; Ganguli et al., 2008), whereas theoretical results for nonlinear networks were missing. Given the numerical observation that $d(k)$ decays exponentially in the ordered regime, one can infer from the presented upper bound on the memory function that the information storage capabilities of qESNs scale like $\mathcal{O}(\log(N))$ with the system size N . It was also shown numerically that this scaling holds for the performance on the representative classification tasks ($p_{\text{exp}}(C, \text{PAR}_5)$ and $p_{\text{exp}}(C, \text{RAND}_5)$) as well. These findings might indicate a trade-off for RC systems between memory capacity and kernel quality. Two extreme examples can illustrate this consideration. On the one hand, delay line networks as well as another special class of linear networks (the so-called orthogonal networks) exhibit a very good memory performance: their memory capacities scale like $\mathcal{O}(N)$ (White et al., 2004), while failing on classification tasks like PAR_n , AND_n (as they are not linearly separable) and with high probability (for large n) also failing on RAND_n . Hence their kernel quality can be considered poor. On the other hand, the nonlinear qESNs exhibit a comparably homogeneous performance over all tasks that were studied in this letter, indicating a good kernel quality but showing only logarithmic memory scaling. Formalizing and investigating this apparent trade-off might reveal deep insights into the art of RC system design.

We informally equated in this letter RC systems with binary units with LSMs. Most work about LSMs is concerned with modeling cortical circuits, and the reservoir consequently often consists of biologically inspired spiking neuron models such as integrate-and-fire type units. We did not explicitly simulate such biologically more realistic reservoirs; however, our results for reservoirs with binary units show characteristic properties also observed in biological modeling. For example, the performance of spiking reservoirs (commonly termed liquids) also strongly depends on the in-degree distribution of the network (Häusler & Maass, 2007). This indicates that the binary nature of spikes is an important factor in the network dynamics of cortical circuits, a feature included in binary qESNs but not present in mean field or rate-coded models of biological circuits.

The finding that binary reservoirs have high performance exclusively for low in-degrees stands in stark contrast to the fact that cortical neurons feature high in-degrees of over 10^4 . This raises the interesting question as to which properties and mechanism of cortical circuits not accounted for in the qESN model may cause this discrepancy. We have shown that

sparse network activity as observed in cortical networks is a suitable candidate mechanism, as it indeed shifts the region of optimal performance to higher in-degree values in binary networks. Interestingly, the sparse activity regime has also been proposed as a good computational regime for ESNs (Steil, 2007; Schrauwen et al., 2008) and can be easily attained using the unsupervised dynamic reservoir optimization techniques mentioned above.

Although sparse activity is a prominent property of neocortical circuits, it is not the only possible explanation for the topological discrepancy between cortical circuits and the optimal circuits identified by our analysis. For example, the transmission of information between neurons via synapses is known to be error prone as, for example, vesicle release into the synaptic cleft is a stochastic process with little reliability (Tsodyks & Markram, 1997). This stochastic aspect of biological connectivity might well result in a considerably smaller “true” or effective in-degree that is closer to the parameter regime found to be optimal for binary qESNs.

Appendix A: Lyapunov Exponents via Branching Processes

In this appendix, the probabilities $p_{ij}^{\alpha\beta}$ defining the multitype branching process described in section 2 are calculated. The network is assumed to be of infinite size ($N = \infty$), and the weight matrix W is assumed to be regenerated independently at every time step (AA approximation). In the AA, the recurrent input $\sum_{j=1}^{\infty} w_{ij}x_j(t)$ to unit i at time t is distributed symmetrically with regard to 0 as the weights w_{ij} are distributed symmetrically with regard to 0. It is straightforward to see that all dynamics are invariant whether the input $u(t) = 1$ or $u(t) = -1$. Hence, without loss of generality (wlog), we can assume $u(t) = 1$ in the rest of the analysis, a fact that makes the use of branching theory possible.

In order to calculate the probabilities $p_{ij}^{\alpha\beta}$, it is necessary to calculate the steady-state probabilities $p_{S_m}(s)$ of a network unit to assume the state $s \in S_m$. Let $z_{ij}(t)$ denote the product $w_{ij}x_j(t)$ of the weight w_{ij} (with $w_{ij} \sim p_w = \mathcal{N}(0, \sigma^2)$) and the activation $x_j(t)$ of unit j at time t . All $z_{ij}(t)$ for $i, j \in \mathbb{N}$ are i.i.d. according to p_z . The recurrent input $Z_i(t) = \sum_{j=1}^{\infty} z_{ij}(t)$ to unit i is the sum of K independent contributions (corresponding to the K recurrent connections for each unit). The following equations hold for the distributions:

$$\begin{aligned}
 p_z(z) &= \int ds \frac{1}{|s|} p_{S_m}(s) p_w(z/s) \\
 p_z &= *^K p_z = \underbrace{p_z * \dots * p_z}_{K\text{-times}} \\
 p_{S_m}(s) &= \int dZ p_Z(Z) \chi_{I_s}(Z + 1),
 \end{aligned} \tag{A.1}$$

where $*^K$ denotes the K -fold convolution and $\chi_I(\cdot)$ is the indicator function of the interval I . The interval $I_s = (\psi_m \circ g)^{-1}(s)$ is the inverse image of s under the quantized activation function $\psi_m \circ g$. Equations A.1 are solved numerically in an iterated fashion for the steady-state distributions p_{S_m} and p_z .

$p_{ij}^{\alpha\beta}$, denoting the probability per output link that a perturbation $s_\alpha \rightarrow s_\beta$ causes a perturbation $s_i \rightarrow s_j$ (where $S_m = \{s_1, \dots, s_{2^m}\}$ is the single unit state space), can easily be calculated using the steady-state distributions p_{S_m} and p_z from equation A.1. It is given by

$$p_{ij}^{\alpha\beta} = \int dw \int dZ' p_w(w) p_{Z'}(Z') \chi_{I_{s_i}}(Z' + s_\alpha w + 1) \chi_{I_{s_j}}(Z' + s_\beta w + 1)$$

$$p_{Z'} = *^{K-1} p_z,$$

where w denotes the weight of the input that is perturbed by the $s_\alpha \rightarrow s_\beta$ perturbation and Z' denotes the recurrent input to the unit from the remaining $K - 1$ presynaptic units, which are unaffected by the $s_\alpha \rightarrow s_\beta$ perturbation.

In an infinite-size network with neuron in-degree K generated as described in section 2, the neuron out-degree is Poisson distributed with mean K . Therefore, the mean descendant matrix M whose element $M_{\alpha+2^m(\beta-1), i+2^m(j-1)}$ denotes the average number of descendants of type $s_i \rightarrow s_j$ caused by a $s_\alpha \rightarrow s_\beta$ perturbation for $\alpha, \beta, i, j = 1, \dots, m$ is given by

$$M_{\alpha+2^m(\beta-1), i+2^m(j-1)} = K \cdot p_{ij}^{\alpha\beta}.$$

According to equation 2.2, the largest Lyapunov exponent is defined as

$$\lambda = \lim_{n \rightarrow \infty} \frac{1}{n} \ln \left(\frac{H(n)}{H(0)} \right)$$

$$H(n) = d(\mathbf{x}^1(n), \mathbf{x}^2(n)) = \|\mathbf{x}^1(n) - \mathbf{x}^2(n)\|_1.$$

Here we restricted ourselves wlog to the use of the $p - 1$ norm (as all norm are equivalent on \mathbb{R}^N). Following Athreya and Ney (1972), the expected number of perturbations after n time steps is given by $Z^T M^n$. Here Z^T denotes the transposed of $Z \in \mathbb{N}^{2^{2m}}$ whose i th element Z_i denotes the initial number of perturbations of type $s_\alpha \rightarrow s_\beta$ with $i = \alpha + 2^m(\beta - 1)$ at time step 0. Hence $H(n)$ can be cast into the following form:

$$H(n) = Z^T M^n D,$$

where the elements of $D \in \mathbb{R}^{2^{2m}}$ are given by $D_{\alpha+2^m(\beta-1)} = |s_\alpha - s_\beta|$. It is then straightforward to see that

$$\lambda = \lim_{n \rightarrow \infty} \frac{1}{n} \ln \left(\frac{Z^T M^n D}{Z^T D} \right) = \ln(\rho),$$

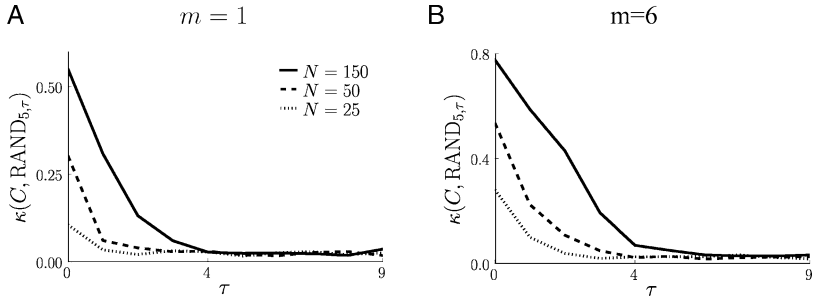


Figure 13: Cohen’s kappa coefficient $\kappa(C, \text{RAND}_{5,\tau})$ plotted as a function of the delay parameter τ for binary networks with $m = 1$ ($\log \sigma = 0.2$, $K = 3$; A) and for high-resolution networks with $m = 6$ ($\log \sigma = 0$, $K = 3$; B). The plots show results for different networks with size $N = 25$ (dotted), $N = 50$ (dashed), and $N = 150$ (solid). Results for each parameter pair (τ, N) were averaged over 50 different circuits C . The performance measure $p_{\text{exp}}(C, \text{TASK}_n)$ can be interpreted as the area under the plotted curves (corresponding to a summation over τ).

where ρ is the largest eigenvalue of M , which is guaranteed to be nonnegative by the Perron-Frobenius theorem.

Although the dimension of M is $2^{2m} \times 2^{2m}$, we cannot expect to obtain 2^{2m} meaningful eigenvalues of M as the matrix also contains 2^m rows describing the “diagonal” perturbations $s_\alpha \rightarrow s_\alpha$, which are only trivial perturbations. Furthermore, as the network dynamics of a qESN defined in equation 2.2 are symmetric with regard to 0, the perturbations $s_\alpha \rightarrow s_\beta$ are equivalent to the perturbation $s_{2^m+1-\alpha} \rightarrow s_{2^m+1-\beta}$. Hence, M only has $2^{m-1}(2^m - 1)$ meaningful eigenvalues of which the logarithm represents the Lyapunov spectrum.

Appendix B: Dependence of Computational Performance on Task Delay and Network Size

The performance measure $p_{\text{exp}}(C, \text{TASK}_n)$ introduced in equation 3.1 for a task TASK_n of n bits (e.g., PAR_5) and a circuit C is given by summation of the performances $\kappa(C, \text{TASK}_{n,\tau})$ for the τ -delayed task $\text{TASK}_{n,\tau}$ over all delays $\tau \geq 0$. The quantity $p_{\text{exp}}(C, \text{TASK}_n)$ is hence not easy to interpret, as it is not bounded for all system sizes $N \in \mathbb{N}$. To give some intuition about the performance of the qESN, the kappa coefficient $\kappa(C, \text{RAND}_{n,\tau})$ (which is in $[0, 1]$) is explicitly plotted as a function of τ in Figure 13A for binary ($m = 1$) and in Figure 13B for high-resolution networks ($m = 6$) with different network sizes $N = 25, 50, 150$ for the task RAND_5 . It can be observed that for fixed N , the kappa coefficient κ decreases monotonically with τ , and for fixed τ , it increases monotonically with N (in agreement with the results

presented in Figure 10). The performance measure $p_{\text{exp}}(C, \text{TASK}_n)$ is the area under the curves shown in Figure 13.

Appendix C: Input Separation $d(k)$ in the Annealed Approximation —

Here we calculate the distance $d(k)$ defined in equation 5.1 for large networks $N \rightarrow \infty$ using the annealed approximation (AA) introduced in Derrida and Pomeau (1986). We denote networks receiving the the input $u^1(\cdot)$ and $u^2(\cdot)$ with N1 and N2, respectively. First, we notice that in the AA and the limit $N \rightarrow \infty$, the following holds:

$$\begin{aligned}
 d(k) &= \lim_{N \rightarrow \infty} \left\langle \frac{1}{N} \sum_{b=1}^N |x_b^1(0) - x_b^2(0)| \right\rangle_C = \langle |x_a^1(0) - x_a^2(0)| \rangle_C \quad \forall a \in \mathbb{N} \\
 &= \sum_{i,j=0}^{2^m-1} q_{ij}(k, u^1, u^2) |s_i - s_j|. \tag{C.1}
 \end{aligned}$$

Here $q_{ij}(k, u^1, u^2)$ denotes the joint probability of finding $x_a^1(k)$ in the state s_i and $x_a^2(k)$ in the state s_j given the input $u^1(\cdot), u^2(\cdot)$. Due to the AA, this probability is independent of the node index a . Hence, in the AA and for $N \rightarrow \infty$, the state of the network is completely described by the joint distribution of a pair of states $x_a^1(k)$ and $x_a^2(k)$ in the state space $\mathcal{S}_m \times \mathcal{S}_m$. Moreover, $q_{ij}(k, u^1, u^2)$ determines the distribution of the recurrent feedback for the next time step $k + 1$. We define the matrix $q(k, u^1, u^2)$ with the entries $q_{ij}(k, u^1, u^2)$. We denote the mapping from $q_{ij}(k, u^1, u^2)$ to $q_{ij}(k + 1, u^1, u^2)$ as S representing the transition from time step k to $k + 1$ by applying the input pair $u^1(k)$ and $u^2(k)$:

$$q(k + 1, u^1, u^2) = S(q(k, u^1, u^2)).$$

C.1 Separation Approximation. Since the state $x_a^i(k)$ of neuron a is quantized with m bits, we can write it in the following way:

$$\begin{aligned}
 x_a^i(k) &= \sum_{l=0}^{m-1} 2^{-l} (b_l^i - 1/2), \quad b_l^i \in \{0, 1\} \\
 B_l(x_a^i(k)) &:= b_l^i. \tag{C.2}
 \end{aligned}$$

According to equation C.2, there is a unique binary representation of $x_a^i(k)$ given by $(b_0^i, \dots, b_{m-1}^i)$; the mapping $B_l(\cdot)$ maps a state to its l th bit. Equation C.2 can be interpreted as effectively replacing unit a with m binary units of states b_l^i whose outputs are reduced by $1/2$ and multiplied

by 2^{-l} and finally summed up; this is still exact. Now we assume that each of these m units receives input drawn independently from the input distribution and has different weights drawn independently from $N(0, \sigma^2)$ every time step. For given presynaptic input, the b_l^1 and b_l^2 are independent for all $l = 0, \dots, m-1$ under this approximation:

$$q_{ij}(k, u^1, u^2) = \prod_{l=0}^{m-1} q_{B_l^1(s_i), B_l^2(s_j)}^l(k, u^1, u^2);$$

$q^l(k, u^1, u^2)$ denotes the 2×2 matrix, whose entry $q_{b_l^1, b_l^2}^l(k, u^1, u^2)$ is the joint probability of finding the bit number l of unit $x_a^1(k)$ in state $b_l^1 \in \{0, 1\}$ and of unit $x_a^2(k)$ in state $b_l^2 \in \{0, 1\}$. Under this approximation, we only have to calculate $4m$ matrix entries instead of the 2^{2m} entries, a considerable reduction of complexity.

We denote the update mapping for q^l by S^l :

$$q^l(k+1, u^1, u^2) = S^l(q(k, u^1, u^2)).$$

An explicit form for S^l can be derived in the following way. First, we condition the probability $q^l(k+1, u^1, u^2)$ on the presynaptic input $h^1 = (h_1^1, \dots, h_K^1) \in \mathcal{S}_m^K$ for network N1 and $h^2 = (h_1^2, \dots, h_K^2) \in \mathcal{S}_m^K$ for network N2 and on the weight matrix W defining the circuit C for this time step $k+1$ (which is the same for N1 and N2). The pairs (h_i^1, h_i^2) , $i \in 1, \dots, K$ are i.i.d. according to $q(k, u^1, u^2)$. This yields

$$\begin{aligned} q^l(k+1, u^1, u^2) &= \langle (q^l(k+1, u^1, u^2 \mid h^1, h^2, W))_C \rangle_{h^1, h^2} \\ &= \sum_{h^1, h^2} p(h^1, h^2) \int dW p(W) q^l(k+1, u^1, u^2 \mid h^1, h^2, W). \end{aligned} \tag{C.3}$$

Here $p(W)$ denotes the probability of the weight matrix W , which is multinormal for all the nonvanishing K weights per row. Conditioned on the input and the weights, the network realizations N1 and N2 are independent, and the K -fold integral over the weights appearing in equation C.3 can be explicitly integrated to a single integral:

$$\int dW p(W) q_{ij}^l(k+1, u^1, u^2 \mid h^1, h^2, W) = F_{ij}^l(h^1, h^2, u^1(k), u^2(k)).$$

F^l is defined as

$$\begin{aligned}
 F_{ij}^l(h^1, h^2, u^1(k), u^2(k)) &= \frac{1}{(8\pi \det(C) \Sigma_{22})^{1/2}} \\
 &\times \sum_{\alpha=0}^{2^l-1} \int_{I_{\alpha,i}-u^1(k)} dv \exp\left(\frac{v^2}{2} \left(\frac{\Sigma_{12}^2}{\Sigma_{22}} - \Sigma_{11}\right)\right) \\
 &\times \sum_{\beta=0}^{2^l-1} \left[\operatorname{erf}\left(\frac{(I_{\beta,j}^+ - u^2(k))\Sigma_{22} + \Sigma_{21}v}{(2\Sigma_{22})^{1/2}}\right) \right. \\
 &\quad \left. - \operatorname{erf}\left(\frac{(I_{\beta,j}^- - u^2(k))\Sigma_{22} + \Sigma_{21}v}{(2\Sigma_{22})^{1/2}}\right) \right].
 \end{aligned}$$

Here we use the following definitions:

$$R = \sigma^2 \begin{pmatrix} (h^1)^T h^1 & (h^1)^T h^2 \\ (h^2)^T h^1 & (h^2)^T h^2 \end{pmatrix}$$

$$\Sigma_{ij} = (R^{-1})_{ij}$$

$$I_{\alpha,i} = [I_{\alpha,i}^-, I_{\alpha,i}^+] := [\tanh^{-1}(2^{-l+1}\alpha - 1), \tanh^{-1}(2^{-l+1}(\alpha + 1) - 1)],$$

where $(h^a)^T h^b = \sum_{i=1}^N h_i^a h_i^b$ denotes the standard dot product. Using the above expression for F_{ij}^l , the update can finally be written as

$$q_{ij}^l(k+1, u^1, u^2) = \langle F_{ij}^l(h^1, h^2, u^1(k), u^2(k)) \rangle_{h^1, h^2}.$$

For the sake of computational tractability for larger m and K , we do not evaluate the $2K$ sums explicitly involved in the average over the presynaptic input $\langle \cdot \rangle_{h^1, h^2}$. Instead we determine this expectation value by a finite number of samples from the joint distribution $p(h^1, h^2)$. This sampling is easily realizable since $p(h^1, h^2)$ is of the product form given in equation C.3. The sample size for all experiments was chosen to be 150.

Appendix D: Bound for the Memory Function

D.1 Upper Bound for the Memory Function. Here we derive the upper bound (see equation 6.3) for the memory function $m(k)$. The target output at time t for the memory task of k time steps is given by $y_T(t) = u(t-k)$. The linear readout with weights α_i has the output $y(t) = \alpha^T x(t) = \sum_{i=1}^N \alpha_i x_i(t)$. α is the learned by linear regression yielding $\alpha = A^{-1} p(k)$, where $A = \langle x(t)x(t)^T \rangle$ is the covariance matrix of the network state $x(t)$ and $p_k = (p_{k1}, \dots, p_{kN}) = \langle x(t)y_T(t) \rangle$. Here $\langle \cdot \rangle$ denotes the

average over the input $u(\cdot)$ for a given circuit A . Using the expression for the memory function given in White et al. (2004) results in

$$m(k) = p_k^T A^{-1} p_k \leq \|A^{-1}\|_2 \|p_k\|^2,$$

where $\|\cdot\|$ is the standard Euclidean norm and $\|\cdot\|_2$ is the corresponding induced operator norm. Without loss of generality we can set the readout time $t = 0$ as the input sequence $u(\cdot)$ is stationary. Now look at a single component of p_k :

$$p_{ki} = \langle x_i[u](0)u_k \rangle.$$

Here $x[u](0) = (x_1[u](0), \dots, x_N[u](0))$ is the state vector that results from applying the left-infinite input sequence $u = (u_1, u_2, \dots)$, where $u_k := u(-k)$. Further, we define

$$u^i := (u_{i+1}, u_{i+2}, \dots)$$

$$u^{i,j} := (u_{i+1}, \dots, u_j).$$

The \circ is used for concatenating sequences such that $u = u^{1,k} \circ u^k = u^{1,k-1} \circ u_k \circ u^k \forall k \in \mathbb{N}$. Using this notation, the component p_{ki} can be written as

$$\begin{aligned} p_{ki} &= \langle x_i[u](0)u_k \rangle = \sum_u p(u)x_i[u](0)u_k \\ &= \sum_{u^{0,k-1}} p(u^{0,k-1}) \sum_{u_k} p(u_k) \sum_{u^k} p(u^k) \langle x_i[u^{0,k-1} \circ u_k \circ u^k](0)u_k \rangle. \end{aligned}$$

Since all inputs are independent, we can carry out explicitly the sum over u_k with $p(u_k) = 1/2$:

$$\begin{aligned} p_{ki} &= \frac{1}{2} \sum_{u^{0,k-1}} p(u^{0,k-1}) \sum_{u^k} p(u^k) (x_i[u^{0,k-1} \circ (+1) \circ u^k](0) \\ &\quad - x_i[u^{0,k-1} \circ (-1) \circ u^k](0)) \\ &= \frac{1}{2} \langle x_i[u^{0,k-1} \circ (+1) \circ u^k](0) - x_i[u^{0,k-1} \circ (-1) \circ u^k](0) \rangle_{u^{0,k-1}, u^k}. \end{aligned}$$

Now the vector $h = (h_1, \dots, h_N)$ is defined in the following way:

$$h_i := x_i[u^{0,k-1} \circ (+1) \circ u^k](0) - x_i[u^{0,k-1} \circ (-1) \circ u^k](0).$$

It can be seen from equation 5.1 of $d(k)$ that the following identity holds:

$$d(k) = \frac{1}{N} \langle \|h\|_1 \rangle.$$

The squared Euclidean norm of p_k can be expressed as

$$\begin{aligned} \|p_k\|^2 &= \frac{1}{4} \sum_i^N \langle h_i \rangle^2 \frac{1}{4} \leq \frac{1}{4} \sum_i^N \langle |h_i| \rangle^2 \\ &\leq \frac{1}{4} \left(\sum_i^N \langle |h_i| \rangle \right)^2 = \frac{1}{4} \left(\left\langle \sum_i^N |h_i| \right\rangle \right)^2 = \frac{1}{4} \langle \|h\|_1 \rangle^2 = \frac{N^2}{4} d(k)^2. \end{aligned}$$

Together with the fact that $m(k) \leq 1$ (which is simply the Cauchy-Schwarz inequality), this finally results in

$$m(k) \leq \min \left\{ \frac{N^2}{4} \|A^{-1}\|_2 d(k)^2, 1 \right\}.$$

D.2 An Annealed Approximation for $\|A^{-1}\|_2$. Here we calculate $\|A^{-1}\|_2$ for $m = 1$ (binary) qESNs using the annealed approximation (AA). We start by explicitly calculating the components $A_{ij} = \langle x_i(t)x_j(t) \rangle$. The diagonal elements simply evaluate to

$$A_{ii} = \langle x_i(t)x_i(t) \rangle = \langle x_i(t)^2 \rangle = 0.25 =: a.$$

The network state update equations are rewritten in the following form:

$$\begin{aligned} x_i(t) &= \Theta \left(\frac{1}{2} z_i(t-1) + u(t-1) \right) \\ z_i(t-1) &:= 2 \sum_{j=1}^K w_{ij} x_j(t-1), \end{aligned}$$

where $\Theta(x) = 1/2$ for $x > 0$ and $\Theta(x) = -1/2$ otherwise. Independent of the input, the quantity $z_i(t-1)$ is normally distributed according to $z_i(t) \sim \mathcal{N}(0, K\sigma^2)$, and $z_i(t-1)$ and $z_j(t-1)$ are i.i.d. for $i \neq j$. Further, as $\langle z_i(t-1) \rangle = 0$, we may assume wlog $u(t-1) = 1$. The probability p_+ for $x_i(t) = +0.5$ and p_- evaluate to

$$p(x_i(t) = +0.5) = p_+ = \Phi \left(\frac{2}{K^{1/2}\sigma} \right)$$

$$p(x_i(t) = -0.5) = p_- = \Phi\left(-\frac{2}{K^{1/2}\sigma}\right)$$

$$\Phi(x) = \frac{1}{2} \left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right).$$

Now the off-diagonal elements of A can easily be computed ($i \neq j$):

$$A_{ij} = \langle x_i(t)x_j(t) \rangle = 0.25(p_+ - p_-)^2 := b.$$

Due to the simple form of A (all diagonal elements are equal to a and the off-diagonal elements are equal to b), its eigenvalues can easily be determined. There are two different eigenvalues $\lambda_{\min} < \lambda_{\max}$:

$$\lambda_{\max} = a + (N - 1)b$$

$$\lambda_{\min} = a - b = 0.25(1 - (p_+ - p_-)^2)$$

$$= 0.25 \left(1 - \left(\Phi\left(\frac{2}{K^{1/2}\sigma}\right) - \Phi\left(-\frac{2}{K^{1/2}\sigma}\right)\right)^2\right).$$

Now the following relation holds for the matrix norm $\|\cdot\|_2$:

$$\|A^{-1}\|_2 = \lambda_{\min}^{-1} = 4 \left(1 - \left(\Phi\left(\frac{2}{K^{1/2}\sigma}\right) - \Phi\left(-\frac{2}{K^{1/2}\sigma}\right)\right)^2\right)^{-1}.$$

Acknowledgments

This letter was written under partial support by the FWO Flanders project G.0088.09; the Photonics@be Interuniversity Attraction Poles program (IAP 6/10); the Austrian Science Fund FWF projects P17229-N04 and S9102-N13; and projects FP6-015879 (FACETS), FP7-216593 (SECO) of the European Union.

References

- Athreya, K. B., & Ney, P. E. (1972). *Branching processes*. Berlin: Springer.
- Beggs, J. M., & Plenz, D. (2003). Neuronal avalanches in neocortical circuits. *Journal of Neuroscience*, 23(35), 11167–11177.
- Bertschinger, N., & Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7), 1413–1436.
- DeFelipe, J., & Fariñas, I. (1992). The pyramidal neuron of the cerebral cortex: Morphological and chemical characteristics of the synaptic inputs. *Prog. Neurobiol.*, 39(6), 563–607.

- Derrida, B., & Pomeau, Y. (1986). Random networks of automata: A simple annealed approximation. *Europhysics Letters*, 1(2), 45–49.
- Derrida, B., & Stauffer, D. (1986). Phase transitions in two-dimensional Kauffman cellular automata. *Europhys. Lett.*, 2, 739–745.
- Destexhe, A., Rudolph, M., & Paré, D. (2003). The high-conductance state of neocortical neurons in vivo. *Nat. Rev. Neurosci.*, 4(9), 739–751.
- Ganguli, S., Huh, D., & Sompolinsky, H. (2008). Memory traces in dynamical systems. *PNAS*, 15(48), 18970–18975.
- Häusler, S., & Maass, W. (2007). A statistical analysis of information processing properties of lamina-specific cortical microcircuit models. *Cerebral Cortex*, 17(1), 149–162.
- Jaeger, H. (2001). *The “echo state” approach to analyzing and training recurrent neural networks* (GMD Rep. 148). St. Augustin: German National Research Center for Information Technology.
- Jaeger, H. (2002). *Short term memory in echo state networks* (GMD Rep. 152). St. Augustin: German National Research Center for Information Technology.
- Jaeger, H. (2007). Echo state networks. *Scholarpedia*, 2(9), 2330.
- Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304, 78–80.
- Jaeger, H., Lukoševičius, M., Popovici, D., & Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3), 335–352.
- Joshi, P., & Maass, W. (2005). Movement generation with circuits of spiking neurons. *Neural Computation*, 17(8), 1715–1738.
- Joshi, P., & Triesch, J. (2008). A globally asymptotically stable plasticity rule for firing rate homeostasis. In *Proceedings of the International Conference on Neural Networks (ICANN)*. Berlin: Springer.
- Katok, A., & Hasselblatt, B. (1995). *Introduction to the modern theory of dynamical systems*. Cambridge: Cambridge University Press.
- Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly connected nets. *J. Theoret. Biol.*, 22, 437–467.
- Langton, C. G. (1990). Computation at the edge of chaos. *Physica D*, 42, 12–37.
- Laughlin, S., de Ruyter van Steveninck, R., & Anderson, J. (1998). The metabolic cost of neural information. *Nature Neuroscience*, 1, 36–41.
- Lazar, A., Pippa, G., & Triesch, J. (2007). Fading memory and time series prediction in recurrent networks with different forms of plasticity. *Neural Networks*, 20, 312–322.
- Legenstein, R., & Maass, W. (2007a). Edge of chaos and prediction of computational performance for neural microcircuit models. *Neural Networks*, 20, 323–334.
- Legenstein, R., & Maass, W. (2007b). What makes a dynamical system computationally powerful? In S. Haykin, J. C. Principe, T. Sejnowski, & J. McWhirter (Eds.), *New directions in statistical signal processing: From systems to brain* (pp. 127–154). Cambridge, MA: MIT Press.
- Legenstein, R., Markram, H., & Maass, W. (2003). Input prediction and autonomous movement analysis in recurrent circuits of spiking neurons. *Reviews in the Neurosciences*, 14(1–2), 5–19.
- Lennie, P. (2003). The cost of cortical computation. *Current Biology*, 13, 493–497.

- Levy, W. B., & Baxter, R. A. (1996). Energy efficient neural codes. *Neural Computation*, 8(3), 531–543.
- Lukoševičius, M., & Jaeger, H. (2007). *Overview of reservoir recipes* (Tech. Rep. 11). Bremen: Jacobs University Bremen.
- Luque, B., & Solé, R. V. (2000). Lyapunov exponents in random Boolean networks. *Physica A*, 284, 33–45.
- Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11), 2531–2560.
- Mayor, J., & Gerstner, W. (2005). Signal buffering in random networks of spiking neurons: Microscopic versus macroscopic phenomena. *Physical Review E*, 72, 051906.
- Mitchell, M., Hraber, P. T., & Crutchfield, J. P. (1993). Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7, 89–130.
- Natschläger, T., Bertschinger, N., & Legenstein, R. (2005). At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems*, 17 (pp. 145–152). Cambridge, MA: MIT Press.
- Packard, N. (1988). Adaption towards the edge of chaos. In J. A. S. Kelso, A. J. Mandell, & M. F. Shlesinger (Eds.), *Dynamic patterns in complex systems* (pp. 293–301). Singapore: World Scientific.
- Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J. J., & Stroobandt, D. (2008). Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71, 1159–1171.
- Shmulevich, I., Dougherty, E., Kim, S., & Zhang, W. (2002). Probabilistic Boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2), 261–274.
- Solé, R., Luque, B., & Kauffman, S. (2000). *Phase transitions in random networks with multiple states* (Tech. Rep. 00-02-011). Santa Fe: Santa Fe Institute.
- Steil, J. J. (2007). Online reservoir adaptation by intrinsic plasticity for back-propagation-decorrelation and echo state learning. *Neural Networks*, 20(3), 353–364.
- Triesch, J. (2007). Synergies between intrinsic and synaptic plasticity mechanisms. *Neural Computation*, 19(4), 885–909.
- Tsodyks, M., & Markram, H. (1997). The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proc. Nat. Acad. Sci. USA*, 94, 719–723.
- Vandoorne, K., Dierckx, W., Schrauwen, B., Verstraeten, D., Baets, R., Bienstman, P., et al. (2008). Toward optical signal processing using photonic reservoir computing. *Optics Express*, 16(15), 11182–11192.
- Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.
- Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). A unifying comparison of reservoir computing methods. *Neural Networks*, 20, 391–403.
- Verstraeten, D., Schrauwen, B., Stroobandt, D., & Campenhout, J. V. (2005). Isolated word recognition with the liquid state machine: A case study. *Information Processing Letters*, 95(6), 521–528.
- Verstraeten, D., Xavier-de Souza, S., Schrauwen, B., Suykens, J., Stroobandt, D., & Vandewalle, J. (2008). Pattern classification with cnns as reservoirs. In *Proceedings*

- of the International Symposium on Nonlinear Theory and Its Applications (NOLTA)*. Tokyo: Institute of Electronics, Information and Communication Engineers.
- Vogels, T. P., Rajan, K., & Abbott, L. (2005). Neural networks dynamics. *Annual Review of Neuroscience*, 28, 357–376.
- White, O. L., Lee, D. D., & Sompolinsky, H. (2004). Short-term memory in orthogonal neural networks. *Phys. Rev. Letters*, 92(14), 148102.
- Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica D*, 10, 1–35.
- Zinn-Justin, J. (2003). *Phase transitions and renormalization group: From theory to numbers*. New York: Oxford University Press.

Received January 19, 2009; accepted August 10, 2009.