Information processing using a single dynamical node as complex system

Supplementary Information

L. Appeltant[1], M.C. Soriano[2], G. Van der Sande[1], J. Danckaert[1] , S. Massar[3], J. Dambre[4], B. Schrauwen[4], C.R. Mirasso[2], I. Fischer[2,*]

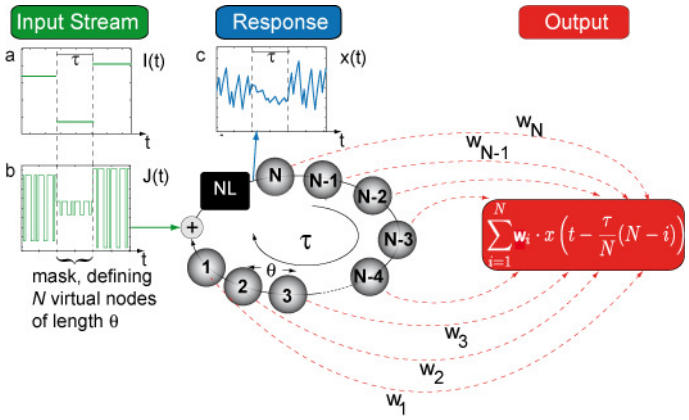
1 Applied Physics Research Group (APHY),Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussel, Belgium

2 Instituto de Física Interdisciplinar y Sistemas Complejos, IFISC (UIB-CSIC), Campus Universitat de les Illes Balears, E-07122 Palma de Mallorca, Spain

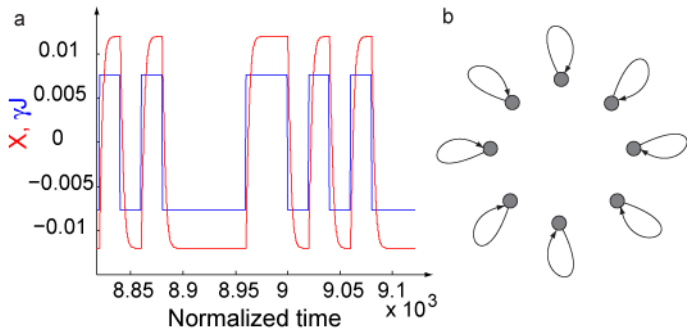
3 Laboratoire d'Information Quantique, CP 225, Université Libre de Bruxelles, Boulevard du Triomphe, B-1050 Bruxelles, Belgium

4 Department of Electronics and Information Systems, Ghent University, St. Pietersnieuwstraat 41, B-9000 Ghent, Belgium

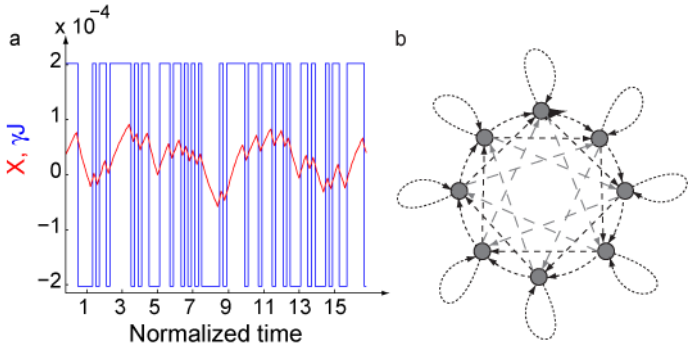
Supplementary Figures



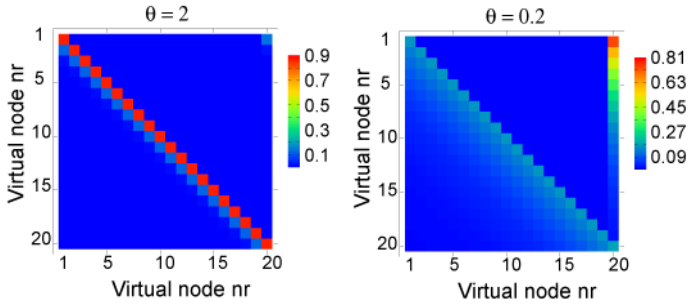
Supplementary Figure S1: Schematic representation of reservoir computing with a delayed feedback system and a single nonlinear node. The discrete input stream $\boldsymbol{u}(k)$ is transformed into a piecewise continuous function $I(t)$ using a sample and hold procedure. Three time steps of length τ are represented in panel (a). This input is multiplied by the mask function $M(t)$. The mask function is piecewise constant over interval θ and periodic over period τ. In the present case the mask function is taken to be a binary function. The resulting function $J(t)=M(t)I(t)$ is represented in panel (b) for 3 time steps. The sum $x(t-\tau)+\gamma J(t)$ drives the nonlinear node, where γ is an adjustable parameter. The response of the nonlinear node is a complex function x(t) depicted in panel (c). Finally, a linear combination of the values of the N virtual nodes is taken to obtain the output. The only parameters which are optimised in this procedure are the input gain γ, the degree of nonlinearity of the nonlinear node (typically depending on a single parameter), the separation θ of the nodes, and the output weights w_i .



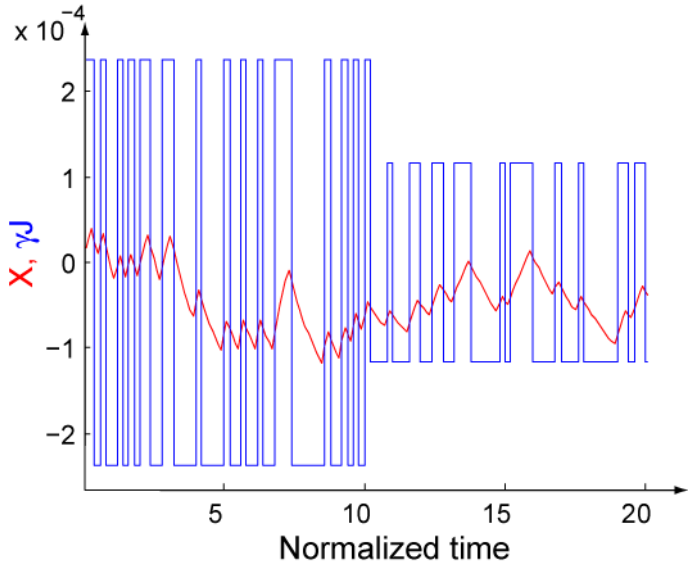
Supplementary Figure S2: Input time trace for large θ and corresponding interaction graph (a) Input time trace $\gamma \cdot J(t)$ (blue) and oscillator output *x(t)* (red) of our system when the time scale T of the Mackey-Glass system is much smaller than the separation θ of the virtual nodes T<<θ. Here we choose T/θ=0.05. The values on both the x- and y-axis are dimensionless. The mask M(t) takes two possible values. For this choice of parameters, the system rapidly reaches a state that is independent of previous inputs. In this regime the system behaves like *N* independent nodes, each of which is coupled only to itself at the previous time step, as schematised by the interaction graph in panel (b).



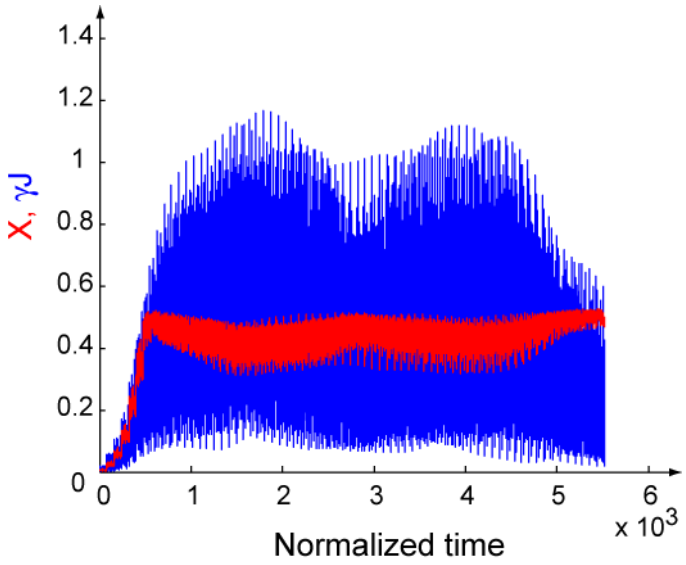
Supplementary Figure S3: Input time trace for small θ and corresponding interaction structure (a) Time trace of input γJ(t) (blue) and oscillator output *x(t)* (red) of our system when the time scale *T* of the Mackey-Glass system and the separation *θ* of the virtual nodes satisfy *T/θ=5 (This is the same plot as in Fig. 5)*. The values on both the x- and y-axis are dimensionless. The mask *M(t)* takes two possible values. In this case the system does not have the time to reach an asymptotic value. Therefore, the dynamics of the nonlinear node couples neighboring virtual nodes, as schematised by the interaction structure in panel (b).



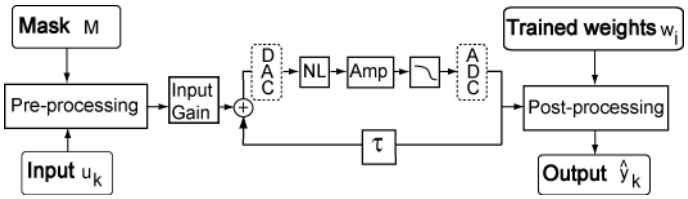
Supplementary Figure S4: Interaction graphs for large and small θ Interaction graphs for different virtual node separation where we plot the coefficients Ω_{ni} and Δ_{ji} of eq. S18 as a matrix using colour coding. For large values of θ (left), the diagonal elements are significantly larger than all others, but when θ decreases (right), the exponential tail of the off-diagonal elements and the also the connection to the last virtual node of the previous input step become dominant.



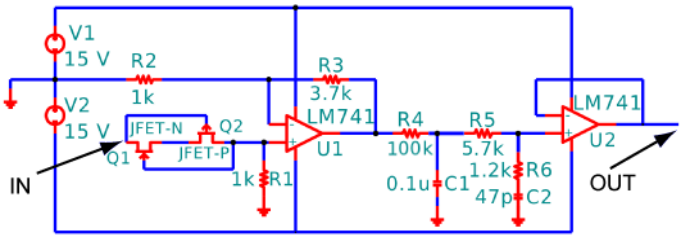
Supplementary Figure S5: Time trace NARMA10 task Time trace of the input stream and the corresponding response of the Mackey-Glass node for the optimal parameters for the NARMA10 task (η = 0.5, γ = 0.01, p = 1 and τ = 80 (400 nodes of 0.2 time units separation)). The blue line represents the input $J(t)$ with imprinted input mask, multiplied by the input scaling γ. The red line shows the Mackey-Glass output $x(t)$. The values on both the x- and y-axis are dimensionless. As input mask, a random series of amplitudes of 0.1 and -0.1 are used.



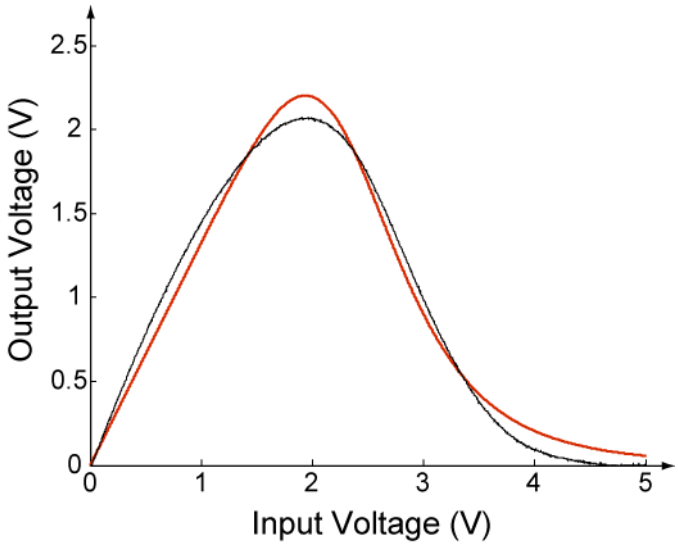
Supplementary Figure S6: Time-trace of the input stream and the corresponding response of the nonlinear node in the case of spoken digit recognition, represented over the duration of one spoken digit The input with imprinted input mask is multiplied by the input scaling γ and is represented by the blue line. The red line denotes the oscillator output. Parameters: $\eta = 0.8$, $\gamma = 0.5$, $p = 7$ and $\tau = 80$ with 400 nodes of 0.2 time units separation. The values on both the x- and y-axis are dimensionless.



Supplementary Figure S7: Schematic representation of the experiment The Mackey Glass system is realised by a nonlinearity (NL), and amplifier (Amp), a low pass filter, and a delay. DAC and ADC, Digital to Analog Converter and Analog to Digital Converter respectively. The input gain block corresponds the parameter γ, while the amplification block corresponds to η.



Supplementary Figure S8: Schematic representation of the hardware node Two FET transistors, one n-channel (Fairchild 2N5457) and one p-channel (Fairchild 2N5460) generate the nonlinear function itself. An amplifier (LM741) provides the desired magnification factor and two RC-circuits allow to set the time constant of the circuit. R1 = 507Ω, R2 = 1kΩ, R3 = 3.7kΩ, R4 = 100kΩ, R5 = 5.7kΩ, R6 = 1.2kΩ, C1 = 0.1µF, C2 = 47pF.



Supplementary Figure S9: Fitting of experimental transfer function Experimental transfer function (black) compared to a fit using the Mackey-Glass equation (red). Fit parameters correspond to C = 1.33, b = 0.4 and p = 6.88 (eq. S20).

Supplementary Discussion

Different perspectives on Reservoir Computing

Reservoir computing (RC) is a powerful method recently introduced in the field of machine learning. In hard classification or prediction tasks, it often outperforms other state-of-the-art approaches. For instance, RC improves prediction of chaotic dynamics by three orders of magnitude over previous methods [7]. Even more it won an international financial time series forecasting competition [27]. Concerning speech recognition, (isolated digit recognition), using the benchmark described in section 'Specific aspects of the benchmark tests', the word error rate was decreased from the previous best of 0.6% to 0.2% [12], while for the Japanese vowel benchmark the test error rate was brought to zero (previous best 1.8%) [28].

RC can be approached from different points of view and can therefore be linked to various fields of research.

From the viewpoint of machine learning, the techniques used in RC are related to those implemented in support vector machines, originally introduced by Vapnik [29]. Support vector machines have proven to be able to attain state-of-the-art performance on a number of tasks. They also rely on a mapping of a low-dimensional input onto high-dimensional states. The main difference to RC lies in the exact realisation of the high-dimensional mapping. While in RC the mapping is explicit - the dynamical response resulting in the reservoir states - , in support vector machines a technique called the kernel trick is employed [30]. The hyperplanes that separate two input classes in the high dimensional feature space are calculated by defining cross products in terms of a kernel function. A second difference is that in reservoir computing the mapping onto feature space is explicitly temporal. This is implemented by reservoirs exhibiting fading memory.

From the viewpoint of neuroscience, RC aims at mimicking, in a reductionist scheme, how our brain does information processing. In this context, RC assumes that the neurons are embedded in a randomly-connected complex network whose intrinsic activity is modified by external stimuli. The persistent neuronal network activity makes the information processing of a given stimulus occur in the context of the response to previous excitations. The generated network activity is then projected into other cortical areas that interpret or classify the outputs. It was this bio-inspired view that motivated one of the original RC concept (Liquid State Machine) [6].

From a dynamical systems point of view, the reservoir can be regarded as a complex dynamical system that operates optimally in a certain dynamical regime. Three basic properties, linked to the dynamical properties of the network [6], should be fulfilled for a network to perform as reservoir. Firstly, different inputs should be mapped onto different reservoir states. This is generally referred to as the *separation property*. Secondly, reservoir states that are only slightly different should be mapped onto identical targets. If not, noise would suffice to map identical inputs onto different target values. This is called the *approximation property*. Finally, *fading memory* is desired. In many tasks, the information is stored in the temporal behaviour of the input (e.g. speech recognition). It does not suffice to process the present input values, also previous values have to be taken into account. Usually, only recent inputs are relevant while those from the far past do not need to be taken into account. These three properties can be realised by the dynamical system, provided that the system resides in a proper dynamical regime. When the system operates in a chaotic regime, it is highly sensitive to small input variations and therefore has very good separation properties. The separation might, however, become so high that the approximation property no longer holds. In the reservoir community it is often claimed that the edge of chaos is an optimal operating point, since it offers a compromise between a stable system, with good approximation properties and fading memory, and a chaotic system, with excellent separation capability. More generally, the edge of stability, where the system goes from an ordered regime to another regime (oscillatory or chaotic), has been identified as appropriate operating point. This can be understood by noting that when a constant input is fed into the reservoir, most likely the target function will also be a constant. In case the system does not reside in a fixed point, but operates, e.g., in the oscillatory regime, a fluctuating reservoir state would need to be mapped onto a constant target, which is difficult to achieve with the linear training algorithm used in RC.

This viewpoint, relating RC to complex dynamics, suggests that RC can be implemented in a wide variety of physical systems, provided that separation, approximation and fading memory properties are fulfilled. This has led to a few proof-of-principle demonstrations using different systems such as a bucket of water [31], the cerebral cortex of a cat [32], a VLSI chip [33], or an array of semiconductor optical amplifiers [34] (the latter only in simulation). However, in all these implementations the tasks performed have been rather simple and the performances did not reach those of digital implementations.

Reservoir computing is not only of conceptual interest. Indeed because of its flexibility, RC represents an alternative approach to building information processing machines. It could find applications in regimes, such as ultra-low energy or ultra-fast computing, which are inaccessible with standard electronics. However the first step before starting to address these issues is to show that analog RC can be efficiently implemented and can reach performances comparable to digital realisations. It is this important milestone which is reached in the present work.

Reservoir computing: general concepts

Different variants of RC have been investigated. All of these variants comprise two layers. The first layer is called *'the reservoir'* or *'the liquid*'. This layer consists of a randomly interconnected network of nonlinear nodes (sometimes referred to as neurons). The nodes are driven by random linear combinations of Q input signals, projecting the original input signal onto a high dimensional state space. The emerging reservoir state is given by the combined states of all the individual nodes. Unlike with traditional recurrent neural networks, the coupling weights in the reservoir are not trained. They are usually chosen in a random way, globally scaled in order for the network to operate in a certain dynamical regime. The second layer performs the readout. Due to the projection of the low-dimensional input data onto a high dimensional space, the readout can be efficiently done by linearly combining the states of the system. The training algorithm can thus be simplified drastically to a linear classifier.

The RC implementation proposed in this paper is closely related to echo state networks [7]. In echo state networks the node states at time step k are computed according to the following equation:

$$\boldsymbol{x}(k) = f\left(W_{res}^{res} \cdot \boldsymbol{x}(k-1) + W_{in}^{res} \cdot \boldsymbol{u}(k) + W_{out}^{res} \cdot \hat{\boldsymbol{y}}(k-1) + W_{bias}^{res}\right) \qquad \text{S1}$$

In this equation, $\boldsymbol{x}(k)$ is the $(Nx1)$-dimensional vector of node states, $\boldsymbol{u}(k)$ is the $(Qx1)$-dimensional input matrix and $\hat{\boldsymbol{y}}(k)$ the reservoir output value(s), all at time step k. The matrices W_{xxx}^{res} contain the (generally random) reservoir, input and feedback connection weights and random bias values. The weight matrices W_{res}^{res} are scaled by multiplicative factors $W_{xxx}^{res} \rightarrow \alpha_{xxx} W_{xxx}^{res}$ in order to get good performance. For the nonlinear function f, often a sigmoidal function, such as $f(x) = tanh(x)$ is chosen. In some cases, feedback from the output to the reservoir nodes is also included. This is not used in our approach, we nevertheless include it here for completeness.

In the most general formulation, the output is a weighted linear combination of the node states, a constant bias value and the input signals themselves.

$$\hat{\boldsymbol{y}}(k) = W_{res}^{out} \cdot \boldsymbol{x}(k) + W_{in}^{out} \cdot \boldsymbol{u}(k-1) + W_{out}^{out} \cdot \hat{\boldsymbol{y}}(k-1) + W_{bias}^{out} \qquad \text{S2}$$

Sometimes the previous value of the output is also taken into account, but in our approach we set W_{out}^{out} to zero.

In RC only the matrices W_{xxx}^{out} in equation S2 are optimised (trained) to minimize the mean square error between the calculated output values $\hat{\boldsymbol{y}}(k)$ and the required output values **y**(k). During the whole process, all weight matrices in equation S1 remain unchanged.

The determination of optimal weight values W_{xxx}^{out}, the process referred to as training, can be performed either in one-shot (offline) learning or by gradually adapting the weights (online learning). The former approach has been applied in our work. It consists of driving the reservoir with a sufficient number of input samples (either a single time trace, as for the NARMA task, or multiple short time traces as for the Spoken Digit Recognition task, see 'Specific aspects of the benchmark tests') and recording the node states for each time step. For N nodes and M time steps, the result is a (NxM)-dimensional reservoir state matrix. To this matrix, we add a constant signal to generate the correct first moment of the required output signal. We will refer to the resulting ((N+1) x M) matrix as S, and to the concatenation of all readout weight matrices as W, being a R x (N+1) matrix, where R is the number of outputs. $\boldsymbol{y}$ designates the R x M matrix corresponding to the desired output. The aim is to minimize the mean square error $\|WS-\boldsymbol{y}\|^2$. This can be obtained by choosing

$$W=\left(\boldsymbol{y}S^{\dagger}\right)^{T} \qquad \text{S3}$$

Here † denotes the Moore-Penrose pseudo-inverse, which allows to avoid problems with ill-conditioned matrices.

After the training stage, the performance of the system is evaluated by applying previously unseen input signals to the reservoir (the testing stage).

In order to avoid overfitting to the training data, regularisation is commonly used, either by adding some Gaussian noise to the node states during training, or by using so-called *Tikhonov regularisation* or *ridge regression*, which minimizes $\|WS-\boldsymbol{y}\|^2+\|\lambda W\|^2$ instead. The second term serves the purpose of keeping the weights as small as possible, while still minimizing the error. Regularisation complicates the training because the parameter λ needs to be optimized first, using yet another data set than the ones used in training and testing. In our paper, ridge regression was used.

For the reader interested in a more in-depth presentation of reservoir computing, we refer to the recent review articles [35,36,37].

Delayed feedback systems as reservoirs

Delayed feedback systems have been extensively studied in the nonlinear dynamics community, see e.g. [1]. The typical evolution equation for a delayed feedback system, such as used in the present work, is

$$\dot{\boldsymbol{x}}(t)=F\left(\boldsymbol{x}(t),\boldsymbol{x}(t-\tau)\right) \qquad \text{S4}$$

F describes a dynamical system, therefore an intrinsic time scale T is present in addition to the delay time τ. The role of these timescales will be discussed later.

Delayed feedback can have a significant impact on the dynamical behaviour of systems. It can e.g. lead to characteristic instabilities, induce synchronisation between subsystems, or also lead to stabilisation. Delayed feedback also implies that the phase space of the system becomes mathematically infinite dimensional, because its state is defined by the continuous function $\boldsymbol{x}(s)$ in the interval $t-\tau < s \leq t$. The delay induces many degrees of freedom, providing an explicit mapping from temporal to spatial information in a high-dimensional space [38].

In the present work we show how, in the context of RC, one can use systems with delayed feedback to drastically increase the available dimensions, even when only a single nonlinear node is used. This is done by exploiting both the present system's state and those of the past as *computing states*. Thus we replace the spatial multiplexing of usual RC (wherein multiple nonlinear nodes act in parallel) by time multiplexing in which a single nonlinear node processes the computing states sequentially.

To this end the delay interval is divided into N pieces of length $\theta=\tau/N$, their endpoints representing nodes. We refer to these nodes as virtual, because they are simply a delayed version of the output of the hardware node.

Contrary to the case of classical reservoirs, where the input vector at a certain time is injected in parallel to all nodes, in our system the input vector is fed to the nodes in a serial manner. We now outline this *masking procedure*, schematised in Supplementary Figure S1, see also Figure 1 (c) in the main text.

First, we sample and hold the input for a duration of τ (see Supplementary Figure S1(a)). The resulting function $I(t)$ is related to the continuous input signal $\boldsymbol{u}(k)$ by

$$I(t)=\boldsymbol{u}(k) \text{ for } \tau k \le t < \tau(k+1) \qquad \text{S5}$$

Second, in order to break the symmetry between the N nodes, we multiply $I(t)$ by a mask function $M(t)$. This mask function is a piecewise constant function, constant over an interval of θ and periodic, with period τ. The values of the mask function during each interval of length θ are chosen independently at random from some probability distribution: $M(t)=W_{in,i}^{res}$ for $(i-1)\theta \le t \le i\theta$ and $M(t+\tau)=M(t)$, with $W_{in,i}^{res}$ random values. In terms of a 'classical' reservoir setup, the values of the mask function $M(t)$ correspond to the weights of the connection between the input layer and the reservoir layer. In equation S1 these weights were referred to as W_{in}^{res}.

When the input signal consists of a single channel, the values to be injected are given by

$$J(t)=I(t)\cdot M(t). \qquad \text{S6}$$

The function $J(t)$ is the product of the input and the mask function (Supplementary Figure S1(b)). When the input consists of Q values $I^j(t)$, we generate a separate mask $M^j(t)$ for each input j and subsequently they are all summed together. The value to be injected is then given by:

$$J(t)=\sum_{j=1}^{Q} I^j(t)\times M^j(t) \qquad \text{S7}$$

The resulting evolution equations are thus

$$\dot{x}(t)=F\left(x(t),x(t-\tau)+\gamma J(t)\right) \qquad \text{S8}$$

where γ is an adjustable parameter (usually referred to as input gain).

The final step when using the delay system as a reservoir computer is to construct the output using a (linear) perceptron so that every discrete input step $\boldsymbol{u}(k)$ is mapped onto a discrete target value $\hat{\boldsymbol{y}}(k)$ and this for every k. The reservoir state comprises the virtual node states, i.e. the values at the end of each interval θ. For the i^{th} virtual node the k^{th} discrete reservoir state is given by

$$x_k^i = x\left(k\tau-(N-i)\theta-\varepsilon\right) \qquad \text{S9}$$

with ε being a very small value compared to θ, which takes into account that the last simulated time step or the last experimental sample is taken. Thereafter a set of trained weights α_i for $i = 1...N$ is used to calculate $\hat{\boldsymbol{y}}_{\boldsymbol{k}}=\sum_{i=1}^{N} w_i\, \boldsymbol{x}_k^i$, where $\hat{\boldsymbol{y}}(k)$ is the calculated approximation of the target function $\boldsymbol{y}(k)$

with $w_i = W^{out}_{res,i}$. The α_i are determined in such a way that the Normalised Root Mean Square Error, defined as

$$NRMSE = \sqrt{\frac{1}{M}\frac{\sum_{k=1}^{M}\left(\hat{y}_k - y_k\right)^2}{\sigma^2\left(y_k\right)}} \qquad \text{S10}$$

is minimized.

Notes on the Mackey-Glass model

Our starting point is the Mackey-Glass model introduced as a model of blood cell regulation [16], modified by an additional input *J(t')*

$$\dot{x}(t') = \frac{1}{T}\left[-x(t') + \frac{C\cdot\left[\alpha\cdot x(t'-\tau') + \beta\cdot J(t')\right]}{1 + b^{p\cdot}\left[\alpha\cdot x(t'-\tau') + \beta\cdot J(t')\right]^p}\right], \qquad \text{S11}$$

with *C* being the coupling factor, *p* the exponent, *b* a nonlinearity coefficient, *T* the intrinsic timescale and τ the delay time. The factor α determines how much of the feedback signal is mixed with the input, while the factor β scales the magnitude of the input signal. The mixing of input and feedback signal happens just before the reinjection into the nonlinear node.

We have rescaled the variables and parameters in the previous equation to obtain the minimum number of significant parameters, as follows: $\eta = C\alpha$, $\gamma = b\beta$, X=bαx and $t = t'/T$, yielding

$$\dot{X}(t) = -X(t) + \frac{\eta\cdot\left[X(t-\tau) + \gamma\cdot J(t)\right]}{1 + \left[X(t-\tau) + \gamma\cdot J(t)\right]^p}. \qquad \text{S12}$$

which is eq. (1) of the main text.

Time Scales

In the above delayed feedback system with external input we can identify three time scales: the separation of the virtual nodes θ, the delay time τ, and the timescale T of the Mackey-Glass system. We find that good performance occurs when the time scales are related by $\theta < \approx T << \tau$.

If $T << \theta$, the Mackey-Glass system reaches its steady state for each virtual node. In this case the reservoir state *x(t)* is only determined by the instantaneous value of the input *J(t)* and the delayed reservoir state $x(t-\tau)$. There is no coupling between virtual nodes, and the dynamics of the system is too simple to perform well as reservoir. The behaviour in this case is illustrated in Supplementary Figure S2.

When $\theta < T$, the state *x(t)* of the system at time t depends on the states of the previous virtual nodes. The strength of this dependency is an exponentially decaying function of the separation of the virtual nodes. However, when T/θ is too large, the Mackey-Glass system is essentially not responding to the instantaneous value of the feedback and input, but only to the average taken over many previous nodes, which is not a good regime of operation either. Empirically we have found that for N=400 virtual nodes, the best choice is T/θ=5. This leads to significant coupling between virtual nodes, but without too much averaging. This regime is illustrated in **Supplementary Figure S3**.

Relation to Traditional Reservoir Computing

Here we establish a more formal link between the traditional formulation of RC given in 'Reservoir computing: general concepts' and the interconnection graphs presented in 'Time Scales'. In contrast to traditional reservoirs where all communication between nodes takes place from one discrete time step to another, in our concept interaction between nodes occurs through inertia of the nonlinear system and through the feedback line. For this reason, the interaction graphs shown in Supplementary Figure S2(b) and Supplementary Figure S3(b) do not quite correspond to the interconnection matrix W_{res}^{res} used in traditional reservoirs (see S1). In what follows, we will derive an approximate interconnection matrix W_{res}^{res} describing the coupling between virtual nodes processing information from different input time steps.

For simplicity of notation, in the following we normalise all times with respect to the intrinsic time scale of the nonlinear system T, that is we work in units where T=1.
Let us consider again the nonlinear equation for the Mackey-Glass node:

$$\dot{x}(t) = -x(t) + f\left(x(t-\tau), J(t)\right) \tag{S13}$$

with

$$f\left(x(t-\tau), J(t)\right) = \frac{\eta\left[x(t-\tau) + \gamma J(t)\right]}{1+\left[x(t-\tau) + \gamma J(t)\right]^p} \tag{S14}$$

where $J(t)=M(t)I(t)$, with $M(t)$ the mask function. We recall that $I(t)$ is constant over each segment with duration τ and equals $u(k)$ over this segment.

In a linear approximation, assuming a constant value of $f(x(t-\tau),J(t))$ during the duration θ, solving the equation yields:

$$x(t) = x_0 e^{-t} + \left(1-e^{-t}\right) f\left(x(t-\tau), J(t)\right) \tag{S15}$$

where x_0 is the initial value at the beginning of each interval θ, i.e., the value for the previous virtual node. In particular, the values of the virtual nodes are given by S15 with t replaced by θ.

We now return to the discrete time of input signal $\boldsymbol{u}(k)$ The state of the ith virtual node ($i \in [1,N]$) is

reached after a time θ , denoted by $\boldsymbol{x}_{i,k}$. The input to virtual node i at time step k equals $w_{in,i}u_k$. (S15) can be written as:

$$\begin{aligned}
x_{1,k} &= x_{n,k-1}e^{-\theta} + \left(1-e^{-\theta}\right) f\left(x_{1,k-1}, w_{in,1}u_k\right) \\
&\cdots \\
x_{i,k} &= x_{i-1,k}e^{-\theta} + \left(1-e^{-\theta}\right) f\left(x_{i,k-1}, w_{in,i}u_k\right) \\
&\cdots \\
x_{n,k} &= x_{n-1,k}e^{-\theta} + \left(1-e^{-\theta}\right) f\left(x_{n,k-1}, w_{in,n}u_{n,k}\right)
\end{aligned} \tag{S16}$$

where θ is the separation of the virtual nodes. This equation allows us to recursively compute each virtual node state at time step k only as a function of the input at the same time step k and virtual node states at time step $k-1$:

$$x_{i,k} = \Omega_{ni} x_{n,k-1} + \sum_{j=1}^{i} \Delta_{ji} f\left(x_{j,k-1}, w_{in,j} u_k\right) \quad \text{S17}$$

with

$$\Omega_{ni} = e^{-i\theta},$$

$$\Delta_{ji} = \left(1 - e^{-\theta}\right) e^{-(i-j)\theta}. \quad \text{S18}$$

This equation is our analogue of equation (S1), representing classical reservoirs and it explicitly describes the state coupling between consecutive time steps. However, it differs from traditional reservoirs because the nonlinear functions are applied to the states before the summation is taken. The interaction topology encoded in S17 is similar to that in the recently proposed cycle reservoir [26]. Supplementary Figure S4 illustrates this interaction topology by showing interaction strength matrices for two values of θ. The coefficients Ω_{ni} correspond to the values found in the last column, while the diagonal and off-diagonal elements are given by Δ_{ji}. In terms of traditional reservoirs, this can be related to W_{res}^{res}.

Specific aspects of the benchmark tests

In the following we provide more details on the procedures used to perform the two benchmark tests: NARMA10 and spoken digit recognition. As noted before, we work in units where the time constant of the nonlinear node is normalised to T=1. This corresponds to the experimental situation where the nonlinear node is fixed, and the delay τ and duration of virtual nodes θ can be adjusted.

NARMA10

The NARMA10 task is one of the most widely used benchmarks in reservoir computing. It was introduced in [23], and used in many other publications in the context of RC, for instance in [21] and [26].

For the NARMA10 task, the input *u(k)* of the system consists of scalar random numbers, drawn from a uniform distribution in the interval [0, 0.5] and the target *y(k+1)* is given by the recursion

$$y_{k+1} = 0.3 y_k + 0.05 y_k \left[\sum_{i=0}^{9} y_{k-i}\right] + 1.5 u_k u_{k-9} + 0.1. \quad \text{S19}$$

In [21], for a reservoir of size N=100, the best performance reported is NRMSE=0.18 (eq. S10). If the reservoir is replaced by a shift register that contains the input, the minimal NRMSE is 0.4. NRMSE values below this level require a nonlinear reservoir.

We illustrate the input procedure, along with the response of the dynamical node, in Supplementary Figure S5. The time trace representing the input stream γJ(t) for the NARMA10 test is plotted in blue and the response of the nonlinear node is shown in red. The input mask consists of a random series of amplitudes of 0.1 and -0.1.The input signal, multiplied with the mask and the input scaling factor γ, is depicted together with the output of the Mackey-Glass node. Note how for this value of θ the Mackey-Glass system is in the transient regime for every node.

The importance of the parameters γ, η and θ is already discussed in the main paper. Note that the optimal parameters for the NARMA10 task are quite particular: the nonlinearity is weak (p=1), the input scaling is small (γ = 0.01), and the input mask is also small (random values of $\pm$ 0.1). This results in a better linear memory, which is crucial for this specific benchmark. Although, because of the

memory requirements the node is close to linear, it still can significantly outperform a shift register that contains the input (for which NRMSE ≥ 0.4) and thus necessarily exploits the weak nonlinearities that are present. We have checked that this requires the inputs to be calculated with sufficiently high precision. These parameters and high precision are presently not accessible in our experiments.

Spoken digit recognition

The spoken digit recognition task, as introduced by Doddington and Schalk [19], is generally accepted as a basic speech recognition task in the RC community [11,12,19]. Also other approaches have used this test as a benchmark, one of them being the Sphinx 4 engine [20] by Sun Microsystems.

The input dataset for the spoken digit recognition consists of a subset of the NIST TI-46 corpus[1] with ten spoken digits (0...9), each one recorded ten times by five different female speakers. Hence, we have 500 spoken words, all sampled at 12.5 kHz. The input for the reservoir is in this case a set of 86-dimensional state vectors with up to 130 time steps. Each of these inputs represents a spoken digit, preprocessed using a standard cochlear ear model [25]. To construct an appropriate target function, ten linear classifiers are trained, each representing another digit of the dataset. The target function is -1 if it does not correspond to the targeted digit and +1 if it does. For every target the time trace is averaged in time and a winner-takes-all approach is applied to select the actual digit. To indicate the performance of the reservoir on this benchmark, the error is expressed both as the word error rate and as the margin. The margin expresses 'the distance' between the reservoir's best guess and the second best guess. When the 10 linear classifiers are trained to be +1 or -1, the resulting output series are averaged over the entire sample and the classifier with the highest mean value is selected to be the best approximation of the +1. If the margin is very high, this implies that the classification is very clear and that no confusion is possible. When the margin is very low, there is almost no difference between the best and the second best guess. Note that even with a very low margin, in theory the word error rate could still go down to 0%. However, both in simulation and experiment, we observe a clear correlation between margin and word error rate.

In the case of speech recognition, to eliminate the impact of the specific division of the available data samples between regularisation, training and testing, we use n-fold cross validation. This means that the entire process of regularisation, training and testing is repeated n times on the same data, but each time with a different assignment of data samples to each of the three stages. The reported performances are the mean across these n runs. For experiment and modelling we used n=20, thus a 20-fold cross validation. In the modelling we additionally averaged over 6 of such runs.

In **Supplementary Figure S6** we illustrate input and response of the Mackey-Glass node for the spoken digit recognition task. Similarly to the NARMA10 test, the nonlinear oscillator exhibits a transient behaviour (red line) because of the fast alternating input signal (blue line). The mask consists of a random assignment of three values: 0.59, 0.41 and 0. The first two values have equal probability of being selected, while the third one is more likely to be selected. Using a zero mask value implies that some nodes are insensitive to certain channels, thus avoiding averaging of all the channels. The degree of nonlinearity is now much higher: p = 7. For the spoken digit recognition task memory is of less importance and more emphasis is put on nonlinear transformation. Hence the node can be much more nonlinear and it becomes possible to implement this experimentally.

This spoken digit benchmark was introduced in the Reservoir Computing community in [11] where a WER of 4.3% was reported for a reservoir of size 1232. In [12] a winner-takes-all approach was introduced, and WER of 0.2% was obtained for a reservoir of size 308. In [26] a WER rate of 1.3%, both for traditional and cycle reservoirs of size 200 is reported. Our experimental system has a WER of 0.2%. For comparison, using Hidden Markov Models, the Sphinx-4 system [20] reported a WER of 0.55% on the same data set.

[1] Texas Instruments-Developed 46-Word Speaker-Dependent Isolated Word Corpus (TI46), September 1991, NIST Speech Disc 7-1.1 (1 disc).

Notes on the electronic Mackey-Glass implementation

Our experimental implementation of reservoir computing, as used in the spoken digit recognition experiments, is shown in Supplementary Figure S7.

Following [17], the Mackey-Glass system is constructed according to the scheme depicted in Supplementary Figure S8. The circuit consists of four parts: the nonlinearity, an amplifier, a RC-filter and a buffer. The nonlinearity was constructed using two field effect transistors, one p-channel, and one n-channel. Both of them are coupled with the gate of each transistor connected to the source of the other, resulting in a transfer function that can be fitted to the Mackey-Glass equation. In Supplementary Figure S both the experimentally observed transfer curve and the fit to the Mackey-Glass equation are depicted. To fit the nonlinearity to the Mackey-Glass equation we use

$$X_{out} = \frac{C \cdot X_{in}}{1 + b^p \left(X_{in} \right)^p}. \qquad \text{S20}$$

he RC filter is used to determine the time constant of the system, which is 10 ms (R4·C1). Connected to the circuit of Supplementary Figure S8, we added a PC controlled A/D D/A converter (National Instruments 6025E, 200 kSamples/s, 12-Bit A/D conversion). The delay line and the combination with the external input are both implemented digitally in the PC via LabView code. The continuously acquired data are delayed for a time corresponding to the feedback time. The input stream $\boldsymbol{u}(k)$ is converted into the function $J(t)$ by imprinting the mask. Finally, the sum of external input $J(t)$ and delayed output of the circuit are fed into the nonlinearity (FET transistors).

Supplementary References

27. http://www.neural-forecasting-competition.com/NN3/index.htm
28. Jaeger, H., Lukosevicius, M, Popovici, D., and Siewert, U.. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3), 335–352 (2007).
29. Vapnik, V.N., *The nature of statistical learning theory*. Statistics for engineering and information science. Springer, second edition (1999).
30. Aizerman, M., Braverman, E. And Rozonoer, L., Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, **25**(6), 821-837 (1964).
31. Fernando, C., and Sojakka, S., Pattern Recognition in a Bucket, *Lecture Notes in Computer Science*, Volume 2801/2003, 588-597, DOI: 10.1007/978-3-540-39432-7_63 (2003).
32. Nikolic, D., Haeusler, S., Singer, W., and Maass, W., Temporal dynamics of information content carried by neurons in the primary visual cortex. *In Proc. of NIPS*, Advances in Neural Information Processing Systems, MIT Press **19,** 1041-1048 (2007).
33. Schürmann, F., Meier, K., Schemmel, J., Edge of Chaos Computation in Mixed-Mode VLSI - A Hard Liquid, *in proceedings of Advances in Neural Information Processing Systems* **17** [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]
34. Vandoorne, K., Dierckx, W., Schrauwen, B., Verstraeten, D., Baets, R., Bienstman, P., and Van Campenhout, J., "Toward optical signal processing using Photonic Reservoir Computing," *Opt. Express* 16, 11182-11192 (2008).
35. Legenstein, R.A. and Maass, W., Edge of chaos and prediction of computational performance for neural microcircuit models. *Neural Networks*, 323-333 (2007).
36. Lukoševičius, M., Jaeger, H., Reservoir computing approaches to recurrent neural network training, *Computer Science Review*, **3,** No. 3, 127-149 (2009).
37. Hammer, B., Schrauwen, B., Steil, J., Recent advances in efficient learning of recurrent networks; Brugge: d-facto; 2009. p. 213-226. In Proceedings of European Symposium on Artificial Neural Networks (ESANN 2009).

38. Giacomelli, G., Meucci, R., Politi, A., Arecchi, F.T., Defects and Spacelike Properties of Delayed Dynamical Systems, *Phys. Rev. Lett.* **73**, 1099 (1994).