PLOS ONE

RESEARCH ARTICLE

Persistent Memory in Single Node Delay-Coupled Reservoir Computing

André David Kovac[1☯], Maximilian Koall[1☯], Gordon Pipa[1‡], Hazem Toutounji[2‡*]

1 Neuroinformatics Department, Institute of Cognitive Science, University of Osnabrück, Osnabrück, Lower Saxony, Germany, **2** Department of Theoretical Neuroscience, Central Institute of Mental Health, Medical Faculty Mannheim of Heidelberg University, Mannheim, Baden-Württemberg, Germany

☯ These authors contributed equally to this work.
‡ These authors also contributed equally to this work.
* hazem.toutounji@zi-mannheim.de

Citation: Kovac AD, Koall M, Pipa G, Toutounji H (2016) Persistent Memory in Single Node Delay-Coupled Reservoir Computing. PLoS ONE 11(10): e0165170. doi:10.1371/journal.pone.0165170

Editor: Eleni Vasilaki, University of Sheffield, UNITED KINGDOM

Received: March 2, 2016

Accepted: October 7, 2016

Published: October 26, 2016

Data Availability Statement: The software code necessary to replicate the findings is available at: https://github.com/Andruschenko/reservoir-persistent-memory.

Funding: The authors acknowledge the financial support by Deutsche Forschungsgemeinschaft (DFG) and Open Access Publishing Fund of Osnabrück University, in addition to the European project PHOCUS in the Framework 'Information and Communication Technologies' (FP7-ICT-2009-C / Proposal Nr. 240763).

Competing Interests: The authors have declared that no competing interests exist.

Abstract

Delays are ubiquitous in biological systems, ranging from genetic regulatory networks and synaptic conductances, to predator/pray population interactions. The evidence is mounting, not only to the presence of delays as physical constraints in signal propagation speed, but also to their functional role in providing dynamical diversity to the systems that comprise them. The latter observation in biological systems inspired the recent development of a computational architecture that harnesses this dynamical diversity, by delay-coupling a single nonlinear element to itself. This architecture is a particular realization of Reservoir Computing, where stimuli are injected into the system in time rather than in space as is the case with classical recurrent neural network realizations. This architecture also exhibits an internal memory which fades in time, an important prerequisite to the functioning of any reservoir computing device. However, fading memory is also a limitation to any computation that requires persistent storage. In order to overcome this limitation, the current work introduces an extended version to the single node Delay-Coupled Reservoir, that is based on trained linear feedback. We show by numerical simulations that adding task-specific linear feedback to the single node Delay-Coupled Reservoir extends the class of solvable tasks to those that require nonfading memory. We demonstrate, through several case studies, the ability of the extended system to carry out complex nonlinear computations that depend on past information, whereas the computational power of the system with fading memory alone quickly deteriorates. Our findings provide the theoretical basis for future physical realizations of a biologically-inspired ultrafast computing device with extended functionality.

Introduction

Some neuron types are endowed with extensive dendritic trees. Each dendrite is characterized by its spatial location within the tree, and the *delay* required for a postsynaptic action potential to propagate to the soma. While several studies investigate the computational role of the dendrites' spatial distribution [1–3], the functionality of dendritic propagation delays is scarcely

probed. One suggestion is that propagation delays enrich the dynamics of recurrent neural networks by turning them into *infinite-dimensional dynamical systems* [4]. The latter observation was the basis of a neurally-inspired computational paradigm, the *single node Delay-Coupled Reservoir* (DCR), where a single nonlinear neuron is delay-coupled to itself [5, 6]. Inputs are multiplexed in time across the delay, and are nonlinearly processed at the neural site. The DCR provides a promising testing bed to theories of neural computations with delays. For instance, some of the authors have shown that applying homeostatic plasticity [7] directly to the delays dramatically improved the computational capabilities of the DCR [8].

In this article, we demonstrate that adding a trained feedback, or *teacher forcing*, to the DCR endow the latter with the ability to store stable memories. The DCR is a specific realization of *Reservoir Computing* (RC) [9–12], which is a flexible framework for capturing temporal dependencies in time series of complex natural systems. This ability is a necessary ingredient in neural information processing [13–16], in addition to spawning a wide range of applications, including time series forecasting [17], signal generation [18] and robot navigation [19].

RC models are large, driven, nonlinear dynamical systems, such as a recurrent neural network, which map their input to a high-dimensional space. On the one hand, the recurrency allows input to travel within the dynamical system, or *reservoir*, for a certain period of time, resulting in a form of *short-term memory*. On the other hand, random nonlinear motifs within the reservoir *nonlinearly* mix past and present inputs. Together, memory and the nonlinear mixing allow a desired output to be *linearly* combined from the activity of the reservoir by output units, using linear regression.

Teacher forcing is a technique originally used in training recurrent neural networks to approximate trajectories of dynamical systems [20, 21]. Output units are clamped to their target value during training, which assures a low amount of training error that, otherwise, would limit the neural network's ability to learn target trajectories. The same principle was also applied successfully to RC with sigmoidal recurrent neural networks and has shown to improve the ability of predicting chaotic trajectories by several orders of magnitude [17]. An alternative training mechanism avoids large training error and the resulting instability of learning by enforcing a rapid decrease in output error at early stages of learning [22, 23]. This procedure, called FORCE, requires chaotic dynamics in the spontaneous activity of the recurrent neural network. The current paper, however, only considers fixed point spontaneous DCR dynamics in concord with previous studies [5, 8, 24], and exploring a setup similar to FORCE will be carried out elsewhere. In addition, in its original form, a DCR only retains temporal dependencies on a short time scale. The goal of the current paper is to extend the DCR's applicability to cases that require *stable memories*, while enhancing its ability to encode and predict dynamical systems by teacher forcing or FORCE is beyond its scope. In classical RC based on sigmoidal and spiking recurrent neural networks, stable memories were shown to be realizable by the aid of teacher forcing and dedicated output units [25, 26], which is the approach we follow to endow the DCR with this capability.

As in classical RC, a DCR receives an input stream that perturbs its autonomous dynamics. Instead of the spatial distribution of input across many neurons, computation in the DCR is implemented by using *time-multiplexing* at several dendritic arbors across the delay line [27]. This leads to *virtual nodes*, at which the dynamics of the single delay-coupled neuron is sampled. Despite these differences, a DCR is approximately equivalent to a recurrent neural network with constrained connectivity [5, 6, 8]. This translates to comparable performance [5], as well as similar principles of self-organization based on plasticity [28–30] that can improve the DCR in an unsupervised fashion [8]. At the same time, the simple architecture of the DCR allows for a largely reduced complexity in physical implementation, which has already been demonstrated on electronic [5], optoelectronic [31, 32] (with the input either in synchrony

with [31] or asynchronous to [32] the delay line), and ultrafast all-optical hardware [33]. Hence, DCRs have the potential for dramatic changes in the field of biologically-inspired ultrafast computation, based on new physical realizations, which is reflected in the fast growing attention paid to this field of research [6].

A standard RC architecture, including the DCR, undergoes rapid washout of previous inputs, a property that is termed *fading memory* [34]. This property assures the execution of computations that demand input retrieval for several time steps in the immediate past. However, systems with fading memory fail at computations that require stable storage of relevant features for arbitrary length of time. Luckily, as pointed out above, this limitation can be overcome by teacher forcing [25]. The latter leads to a stabilization of a finite number of memorized states, and therefore extends the class of executable computations. Here, we demonstrate that such feedback can also be employed to stabilize memory in DCRs. Based on simulations, we show that DCRs which incorporate trained feedback are able to have memory traces of an arbitrary length.

The article is structured as follows. We start with describing the RC architecture that is based on a single nonlinear node with delayed feedback, i.e., the DCR. We then present how the DCR can be extended by linear feedback. This is followed by numerical simulations, which demonstrate the role of teacher forcing in stabilizing memory, while preserving the system's ability to perform nonlinear computations. These simulations consist of three experiments, showing that the enhanced system is able to learn complex nonlinear tasks requiring long-term memory that cannot be learned by classical DCRs. At last, we demonstrate that this memory can be maintained for practically infinite time.

Materials and Methods

Delay-based Reservoir Computing

In a DCR, the recurrent neural network in classical RC is replaced by a single nonlinear node with delayed feedback. Past and present inputs $u \in \mathbb{R}^m$ undergo nonlinear mixing via injection into the nonlinear node. Formally, the dynamics can be modeled by a forced (or driven) *Delay Differential Equation* (DDE) of the form

$$\dot{x}(t) = -x(t) + f(x(t-\tau), u(t)), \tag{1}$$

where τ is the delay time, and $x(t), x(t-\tau) \in \mathbb{R}$ are the current and delayed DCR activities. Fig 1 illustrates the full DCR setup with trained feedback.

DCRs were successfully implemented both virtually and physically. Despite variable performance of different implementations, the principal computational properties remain invariant. Here, we restrict our simulations to the Mackey-Glass system [35]. This choice of nonlinearity is motivated by its superior performance, and the possibility of approximating it by electronic circuits [5]. After a proper transformation M of the input (see below), the input-driven Mackey-Glass DCR is modeled by:

$$\dot{x}(t) = -x(t) + \frac{\eta(x(t-\tau) + \gamma M u(t))}{1 + (x(t-\tau) + \gamma M u(t))^{\rho}} \tag{2}$$

where γ, η and ρ are model parameters, the latter regulating the chaoticity of the system.

Solving the system (1) for $t \geq 0$ requires specifying an appropriate *initial value function* $\phi_0 : [-\tau, 0] \to \mathbb{R}$. This suggests that the phase space in which the solution resides is a *Banach space* $C_{1,\tau} = C([-\tau, 0], \mathbb{R})$ which is *infinite dimensional* [36]. This entails that using a DDE as a reservoir provides the high-dimensional expansion of input, usually achieved by using a large number of neurons.

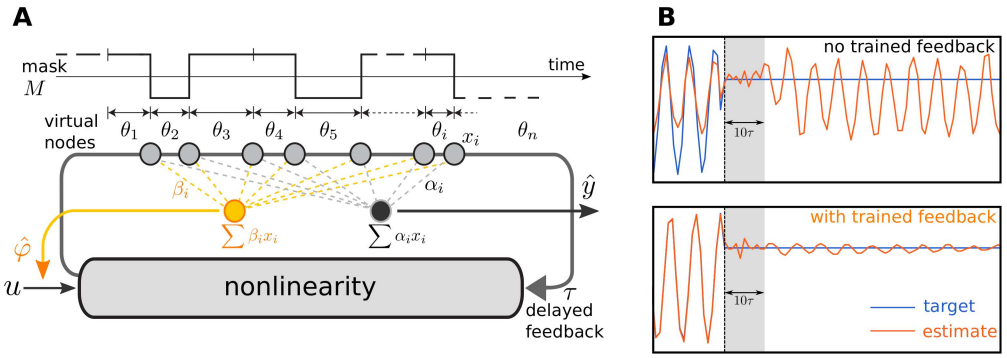

Fig 1. DCR with trained feedback (fDCR) and its extended functionality. (A) In a fDCR, the input u and trained feedback signals φ are *temporally multiplexed* across a delay line of length τ, by using random binary masks M of n bits each (only one mask shown). Each mask bit M_i is held constant for a short delay θ_i, such that $\sum\theta_i = \tau$. The masked input is then nonlinearly transformed and mixed with past input by the nonlinear node with delayed feedback. At the end of each θ_i, resides virtual nodes with activity x_i. Feedback coefficients β_i are estimated through *teacher forcing* to allow storing stable memories, and feedforward coefficients α_i are then trained to perform computations through linear regression. (B) With no trained feedback (top), the DCR is not capable of retaining cue information (dashed vertical line) beyond the bounds of fading memory ($\sim 10\tau$). With trained feedback (bottom), memory of the cue is stabilized, allowing the fDCR's output (orange) to track the change in the target function (blue) beyond the bounds of fading memory.

doi:10.1371/journal.pone.0165170.g001

Instead of distributing an m-dimensional input spatially across neurons, input to the DCR is *time-multiplexed*, which is carried out as follows:

The DCR receives a constant input $u(\bar{t}) \in \mathbb{R}^m$ in each reservoir time step $\bar{t} = \lceil \frac{t}{\tau} \rceil$, corresponding to one τ-cycle of the system. The input is then linearly transformed by a mask M that is piecewise constant for short periods θ_i. These represent the delays between sampling points of $i = 1, \ldots, n$ *virtual nodes* along the delay line. Accordingly, the delays between the virtual nodes satisfy $\sum_{i=1}^{n} \theta_i = \tau$, where $n \gg m$ is the effective dimensionality of the DCR. Here, the mask M is binary with random mask bits $M_i \in \{-\mu, +\mu\}^m$, so that the virtual node i receives a weighted input $M_i u(\bar{t})$. In order to assure that the DCR possesses fading memory of the input, the system (1) is set to operate, when unforced, in a single fixed point regime. Thus, the masking procedure effectively prevents the driven dynamics of the underlying system from saturating to the fixed point.

Following the time-multiplexing of input, a sample of the DCR's response is read out at the end of each θ_i. This yields n predictors x_i per time step $\bar{t}$, corresponding to the virtual nodes' activity. Computations are performed on the predictors using a linear regression model for some scalar target time series y, given by $\hat{y}(\bar{t}) = \sum_{i=1}^{n} \alpha_i x_i(\bar{t})$. The coefficients α_i are determined by using the *least squares solution*, minimizing the sum of squared errors $\sum_{\bar{t}} (y(\bar{t}) - \hat{y}(\bar{t}))^2$. These linear readouts are called *feedforward readouts* to distinguish them from *feedback readouts* of the extended DCR.

Feedback readouts for stabilizing memory

In an RC architecture, parameters need to be tuned such that it possesses fading memory. This is achieved in DCRs by setting the nonlinearity to operate in a fixed point regime, in addition to masking the input as outlined above. Similar to classical RC, possessing fading memory alone restricts the class of computations a DCR can carry out to those that depend on relatively recent inputs only. In order to overcome this restriction, we rely on an important theoretical result for conventional RC [25]. This result states that under certain conditions, augmenting

the system with trained feedback allows it to store nonfading memory. The same extension can be applied to the DCR, leading to similar boost in its computational capability.

More precisely, input is extended by additional channels $\varphi(\bar{t}) \in \mathbb{R}^q$, which are the fed back outputs $\hat{z}(\bar{t}-1) = \sum_{i=1}^{n} \beta_i x_i(\bar{t}-1)$ at the previous reservoir time step $\bar{t}-1$ of a subset of linear readouts. The resulting DCR with trained feedback (fDCR) is shown in Fig 1. The regression coefficients β_i of these feedback readouts are estimated offline at the end of initial teacher forcing phase that precedes the training of feedforward readouts: The reservoir is fed with training data $(u(\bar{t}), \tilde{\varphi}(\bar{t})) \in \mathbb{R}^{m+q}$, where training feedback signals are replaced by a noisy version of their target values $\tilde{\varphi}(\bar{t}) = \tilde{z}(\bar{t}-1) = z(\bar{t}-1) + \epsilon(\bar{t}-1)$. Adding noise ϵ assures that at later phases, the feedback readouts are robust to noise, i.e., prediction errors in the trained feedback are not amplified due to overfitting [25]. The feedback coefficients β_i are determined by using the least squares solution, minimizing the sum of squared errors $\sum_{\bar{t}} (\tilde{z}(\bar{t}) - \hat{z}(\bar{t}))^2$.

Following teacher forcing, feedforward coefficients α_i are estimated offline at the end of the training phase, outlined in the previous section. The model is then validated on new input and feedback time series. Feedback signals in both training and validation phases are computed by $\hat{\varphi}(\bar{t}) = \sum_{i=1}^{n} \beta_i x_i(\bar{t}-1)$. The full procedure is shown in Fig 2.

Computational tasks

In order to examine the ability of the fDCR to retain memory traces for time spans which exceed fading memory, we designed three tasks whose proper execution requires the presence of long-term memory. The exact experimental setups and typical results are presented in the Results section.

In the first two tasks the target function switches between two computations on one or more input streams where each of these computations on its own does not require long-term memory. The switch between the two distinct computations is triggered by short cues, which are received alternately via two additional input streams. A cue in the first cue channel triggers a switch from one task to the other, while the second cue triggers the opposite switch. In order to learn the described tasks, the fDCR has to preserve a memory trace of the last cue at every point in time. This is an easy task if the duration between cues is within the bounds of fading memory, i.e., smaller than around 10 reservoir time steps. If, instead, time gaps between two successive cue signals exceed this limit, the resulting long-term memory dependent task cannot be learned by the standard DCR. This in mind, experiments here are designed with gaps between cues which exceed the fading memory trace by at least tenfold. In order to control for the possibility that feedback readouts are only learning to generate periodic signals, independent of cue time, cues are irregularly spaced.

In the third experiment, more complex feedback signals are trained to encode the time since last cue. This task is designed to demonstrate that the fDCR is not only capable of registering the binary information of a cue's presence or absence, but also the time since the last cue has

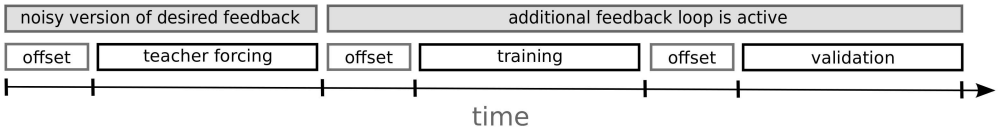

Fig 2. Training and validation of fDCR. Input-output pairs are split into three consecutive phases. In the *teacher forcing* phase, the feedback readouts are trained. In the *training phase*, feedforward readouts are trained to solve computational tasks that require both fading and nonfading memory. in the *validation phase*, the model performance is assessed with unseen data. Each phase is preceded by a brief offset for the fading memory of the fDCR to wash out.

doi:10.1371/journal.pone.0165170.g002

Table 1. Parameter values of the Mackey-Glass system and the fDCR.

Parameter	Value	Description
γ	0.01	scaling factor of input
η	0.5	scaling factor of delayed nonlinearity
ρ	1[a]	regulating parameter of chaoticity
$\pm\mu$	±0.1	values of mask bits
n	300	number of virtual nodes
τ	600	delay time

The first group of parameters corresponds to the parameters of the Mackey-Glass system 2, while the second corresponds to the parameters of the fDCR.

[a]This choice of ρ sets the unforced Mackey-Glass system to operate in a single fixed point regime.

doi:10.1371/journal.pone.0165170.t001

been shown, and to use this information in computation. This information is stored in the value of a ramping feedback signal. The time scale at which time is stored is defined by slope of the ramp length. Longer ramps corresponds to higher sensitivity to the time of older cues.

We follow these three experiments with a longitudinal simulation, that serves to demonstrate how stable cue storage is.

Model parameters

Mackey-Glass parameters as they appear in Eq (2) in addition to all other fDCR parameters are fixed across all simulations, and are summarized in Table 1.

Consecutive cue onsets are separated by gaps that are uniformly drawn from the ranges [50, 400] and [100, 800] reservoir time steps for teacher forcing and for training and validation, respectively. The gaps during teacher forcing are shorter to assure that the regression sees more cues. Otherwise, the sparsity of cues would lead the regression for feedback weights to minimize square errors by assuming that no cues exist. Cues have a duration of 5 reservoir time steps.

Simulation

The DDE Eq 1 was numerically solved using the recursive *method of steps* for handling delays, and *Heun's method* for numerical integration. Heun's method assures quadratic decay of errors with respect to discretization time step. The numerical solution is evaluated at 600 simulation points across the overall delay τ. The latter contains 300 virtual nodes, distributed randomly over the simulation points, such that $\sum_i \theta_i = \tau$.

Each experiment follows the training and testing procedure outlined in Fig 2. Table 2 shows the number of reservoir time steps in each simulation phase.

Results

Experiment 1: Switching between a sine function and a constant value

The setup of the fDCR for this experiment is depicted in Fig 3A. The target output d is either the oscillatory $u^{\sin}$ (a sine wave $100 \sin(\bar{t}) + 1/3$, filtered by a Gaussian kernel with $std = 5\tau$) after the cue onset in the input channel u^{+}, or is constant at 5 after cue onset in input channel u^{-}. The trained feedback signal d^{tf} represents the fourth input stream.

A closer look at the output signal immediately following cue u^{-} onset in the top panel of Fig 3B demonstrates that the standard DCR produces output that is fairly close to the desired constant value. However, only after a few reservoir time steps the cue's fading memory vanishes, and the input-driven system returns to oscillate in synchrony with its input. Thus, the desired

Table 2. Number of reservoir time steps within each phase of simulation.

Phase	Duration (in reservoir time steps)	
	Experiments 1 and 2	Experiment 3
offset	50	50
teacher forcing	50000	50000
training	10000	20000
validation	10000	20000

doi:10.1371/journal.pone.0165170.t002

signal could not be retained over the entire time span between two consecutive cues. This instability is due to the fact that the readout neuron has to transform transient information (caused by permanent input) into a stable output. Yet, approximating constant output is very difficult for reservoir computing [10]. The mismatch between the desired and observed output signals is also demonstrated by the divergence from diagonal of their corresponding scatter plots (upper panel in Fig 3C), particularly when the desired output is at the constant value.

The fDCR, on the other hand, overcomes this limitation, as shown in the bottom panel of Fig 3B. The close match between desired and observed signals demonstrates that trained feedback is successfully utilized in order to stabilize the memory of the last cue (also see lower

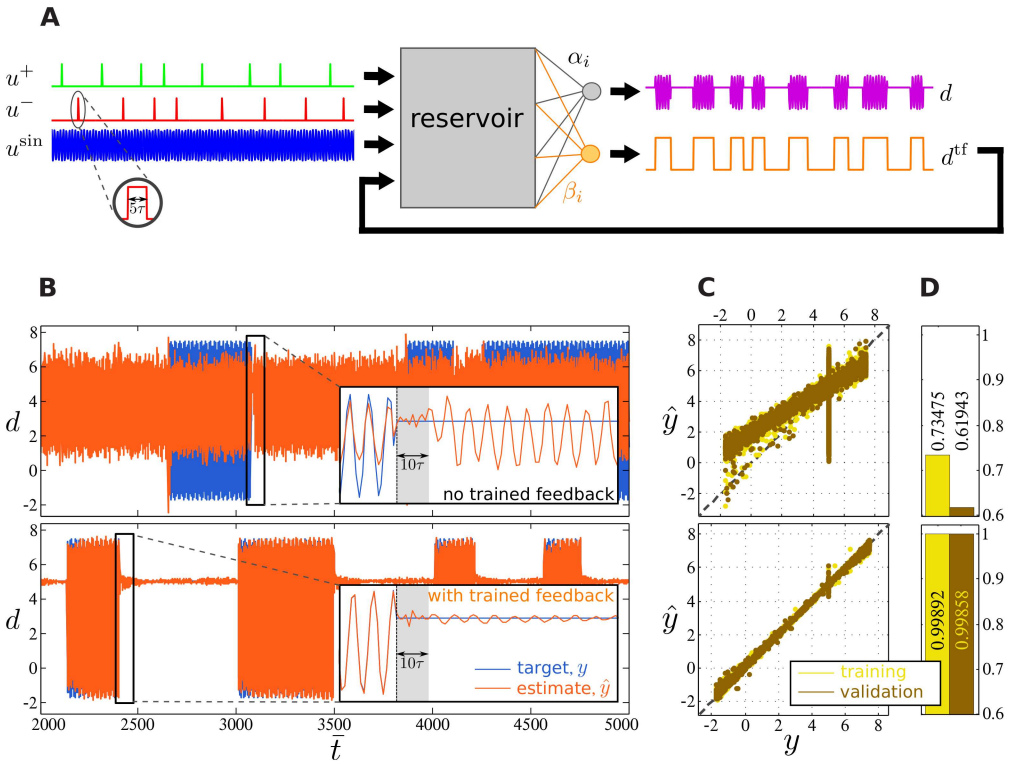

Fig 3. Experiment 1: Switching between a sine wave and the constant value 5. (A) Schematic of the experiment. Input consists of four streams: two cue channels u^+ and u^-, a sine wave $u^{\sin}$, and the additional trained feedback signal d^{tf}. First, feedback weights (orange) are learned by teacher forcing, followed by the feedforward weights (gray). Successful learning is achieved when the output signal d (magenta) matches its target value. (B) Comparison between the output signal (orange) and target value (blue) with (bottom) and without (top) trained feedback. (C) Scatter plots of the target signal y verses observed output $\hat{y}$ with (bottom) and without (top) trained feedback, and (D) their correlation coefficient for training (brown) and validation data sets (yellow).

doi:10.1371/journal.pone.0165170.g003

panel in Fig 3C). We also note that the fDCR approximates the desired output with increasing accuracy over time.

Finally, to quantify these observations, we compute correlation coefficients between the desired and observed signals for both training and validation sets (Fig 3D). We particularly note that, in addition to significantly higher correlation in the case of fDCR, the correlations for training and validation data are very similar (see lower panel in Fig 3D). This demonstrates that the fDCR generalizes well and better than standard DCR (upper panel in Fig 3D).

Experiment 2: Concurrent linear and nonlinear tasks

We demonstrated that including trained feedback stabilizes memories of cue signals. We now show that the computational resources of the fDCR are not fully depleted by the demands of learning these signals. Mainly, we show that the fDCR is still capable of performing several (potentially nonlinear) computations concurrently, some of which are cue-independent.

Experiment 2 is designed with this goal in mind, as shown in Fig 4A. Input to the fDCR consists of two cue channels u^+ and u^-, two streams of bounded and filtered uniformly distributed noise u_1^{arb} and u_2^{arb}, and the additional trained feedback signal d^{tf}. Input signals u_1^{arb} and u_2^{arb} are filtered with a Gaussian kernel (std = 5τ) to improve performance after being drawn uniformly from the range $[-5, 15]$. Nevertheless, the fDCR is still capable of learning computations on uniform white noise as well. The fDCR is trained to perform three computations $d_i(u_1^{\text{arb}}, u_2^{\text{arb}})$ for $i = 1, 2, 3$. These computations are given by:

$$d_1(\bar{t}) = \begin{cases} u_1^{\text{arb}}(\bar{t}) & \text{when } u^+ \\ 2u_2^{\text{arb}}(\bar{t}) & \text{when } u^- \end{cases} \tag{3}$$

$$d_2(\bar{t}) = \begin{cases} u_1^{\text{arb}}(\bar{t}) + u_2^{\text{arb}}(\bar{t}) & \text{when } u^+ \\ |u_1^{\text{arb}}(\bar{t}) - u_2^{\text{arb}}(\bar{t})| & \text{when } u^- \end{cases} \tag{4}$$

$$d_3(\bar{t}) = 0.1\left[u_1^{\text{arb}}(\bar{t})\right]^3 + 0.2u_1^{\text{arb}}(\bar{t}) \cdot u_2^{\text{arb}}(\bar{t}) \tag{5}$$

The first computation d_1 (Eq 3) is cued and linear. The second computation d_2 (Eq 4) is also cued but is nonlinear, due to the absolute value computation upon the onset of u^-. These two computations are performed concurrently using the same cue signals. The third computation d_3 (Eq 5) is a nonlinear function of the two random input signals, and is independent of the cues.

As Fig 4B shows, feedforward linear readouts (orange) are able to closely track the desired signals. Only in the highly nonlinear u^--cued d_2 computation, the scatter plot between the desired and observed signals diverges slightly from the diagonal (see middle panel in Fig 4C). This divergence results, however, in a minuscule reduction in correlation between the two signals, as shown in the middle panel of Fig 4D. The bottom panels of Fig 4B–4D show that, despite its independence from the trained feedback signal, the nonlinear d_3 computation is performed with remarkable precision. This suggests that the fDCR dynamics is rich enough to support concurrent nonlinear computations, with no detectable interference between a trained feedback signal and those computations not dependent on it.

Experiment 3: Feedback depending on time since last cue

Experiment 3 assesses different aspects which go beyond experiments 1 and 2. Here, the fDCR is simultaneously required to learn and maintain two feedback loops. Each feedback loop stores

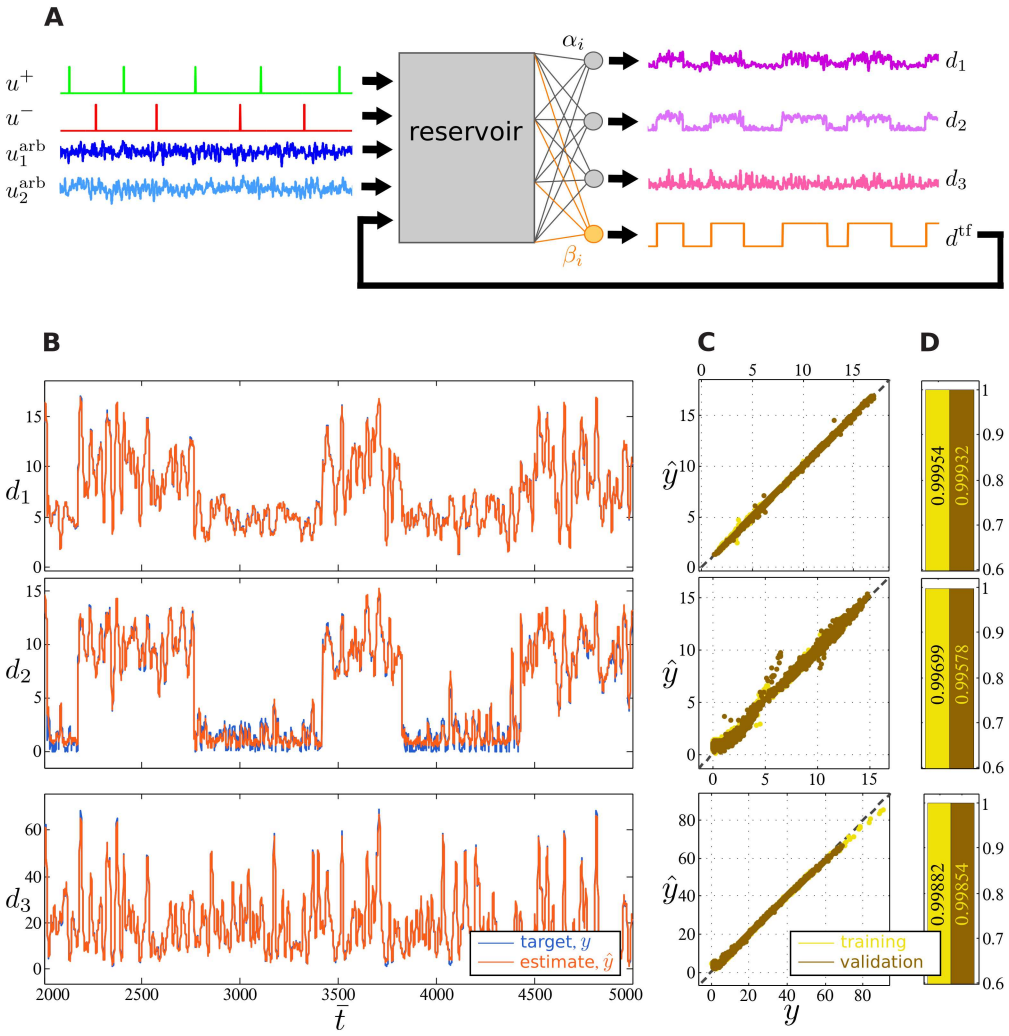

Fig 4. Experiment 2: Concurrent linear and nonlinear tasks. (A) Schematic of the experiment. Input consists of five streams: two cue channels u^+ and u^-, two streams of bounded, uniformly distributed noise u_1^{arb} and u_2^{arb} that are filtered by a Gaussian kernel, and the additional trained feedback signal d^{tf}. In addition to the trained feedback signal, the fDCR computes three desired outputs d_1, d_2, and d_3, corresponding to a linear cued task, nonlinear cued task, and nonlinear cue-independent task, respectively. (B) Comparison between desired (blue) and observed (orange) fDCR output signal d_1 (top), d_2 (middle), and d_3 (bottom). (C) Scatter plots of the target verses observed output for both training (yellow) and validation (brown) data sets, when the target is d_1 (top), d_2 (middle), and d_3 (bottom). (D) Correlation coefficient between desired and observed fDCR outputs for both training (brown) and validation (yellow), when the target is d_1 (top), d_2 (middle), and d_3 (bottom).

doi:10.1371/journal.pone.0165170.g004

the time of last cue up to a certain threshold, one corresponding to fast d_f^{tf} and the other to slow d_s^{tf} forgetting of the time since last cue. This allows for computations that are not only a function of cue onset, but of its time, such as delayed response tasks.

This effect is implemented as follows. A cue triggers a sudden downward shift from amplitude 9 to 5 in each of the feedback signals. Instead of a sudden upward shift, feedback signals linearly increase back to a threshold value; i.e., a cue triggers a rising ramp. The two ramps are of different time scales, corresponding to 300 and 600 reservoir time steps for the short and a long ramp, respectively, as shown in Fig 5A. These time scales are larger than the fading

memory capacity of the standard DCR in order to assess long-term stable memory of the time since last cue onset.

In contrast to the previous two experiments, the feedback signal itself is manipulated by cue onset in Experiment 3. Particularly, the target function d_3 nonlinearly combines the random input stream u^{arb} with the fast ramp d_f^{tf}:

$$d_3(\bar{t}) = |u^{\mathrm{arb}}(\bar{t})|^{-2.5 \cdot d_f^{\mathrm{tf}}(\bar{t})} \tag{6}$$

As shown in Fig 5B shows, the output d_3 is learned with high precision. A little mismatch occurs when the target signal $y < 1$, as the scatter plot Fig 5C demonstrates. However, the effect of this mismatch on the correlation between the target and desired signals is very little, as shown in Fig 5D, and does not result in overfitting the training data. In fact, the inability of the readout to track the target signal when $y < 1$ indicates that the readout mechanism is robust to outliers, since events where $y < 1$ are only sparsely present in the target time series (see S1 Fig). Further simulations demonstrate, as shown in S2 Fig, that the ability to carry out computations that depend on the time of cue onsets are not restricted to the function d_3.

Experiment 4: How stable is cue memory?

In order to answer this question, we ran a longitudinal simulation of a final experiment. It assesses how long the information of latest cue (as in Experiments 1 and 2) can be stored stably in the fDCR via the trained feedback loop.

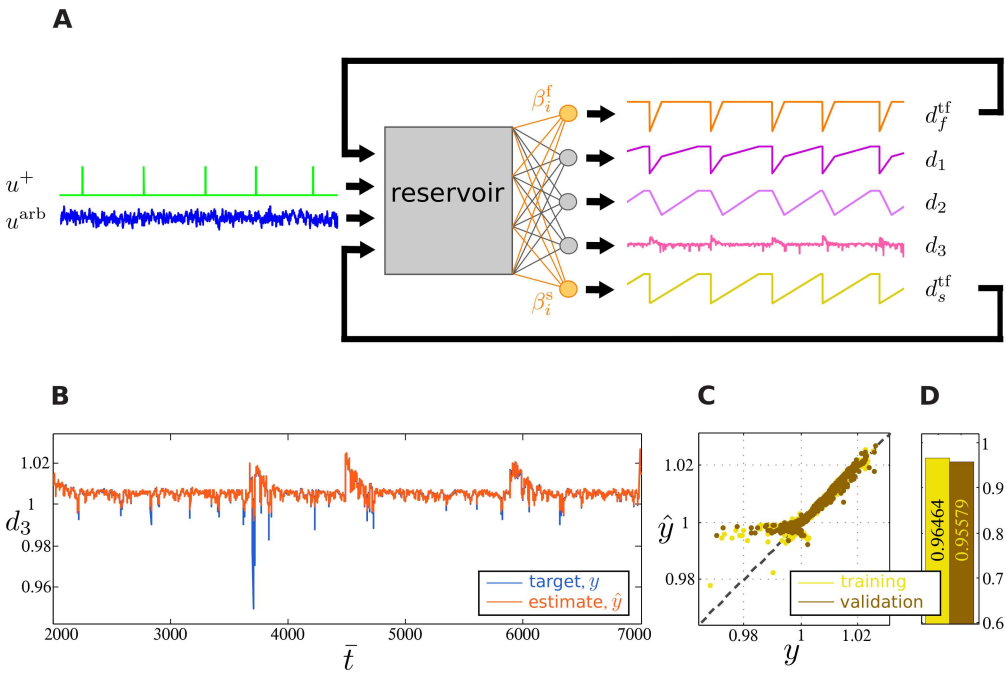

Fig 5. Experiment 3: Feedback depending on time since last cue. (A) Schematic of the experiment. Input consists of four streams: one cue channel u^+, a stream of bounded, uniformly distributed noise u^{arb} that is filtered by a Gaussian kernel, and the additional fast d_f^{tf} and slow d_s^{tf} trained feedback signals that are ramps of duration 600 and 300 Reservoir time steps, respectively. In addition to the trained feedback signals, the fDCR computes three desired outputs d_1, d_2, and d_3, corresponding to the sum of the feedback signals, their difference, and nonlinear function of the fast ramp and the random input, respectively. (B) Comparison between desired (blue) and observed (orange) fDCR output signal d_3. (C) Scatter plots of the target verses observed output for both training (yellow) and validation (brown) data sets, when the target is d_3. (D) Correlation coefficient between desired and observed fDCR output for both training (brown) and validation (yellow), when the target is d_3.

doi:10.1371/journal.pone.0165170.g005

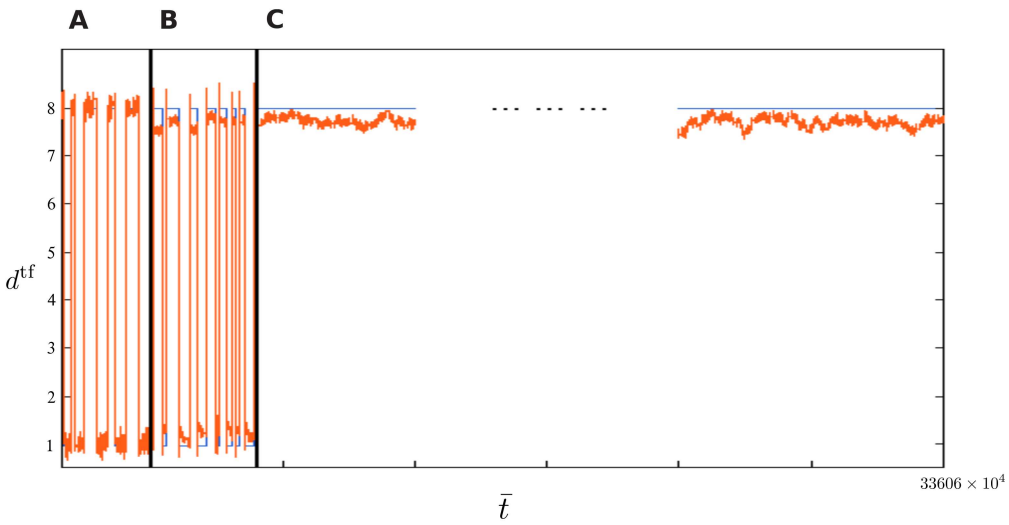

Fig 6. Experiment 4: stable storage of the cue through a longitudinal simulation. (A) A noisy version of the desired feedback signal (teacher forcing). (B) Trained feedback operating on input data with randomly timed cues. (C) Stability test phase where no more cues are input to the fDCR.

doi:10.1371/journal.pone.0165170.g006

The input to the fDCR for this experiment is similar to that of Experiment 1 (see Fig 3A), but with the sinusoidal input u^{sin} replaced by bounded random noise u^{arb} as in Experiment 2. This is to assure that long-term memory is robust to noise coming from that input channel.

Only one readout is trained by teacher forcing to generate the feedback signal. The ability of the fDCR is then tested for a number of time steps on generating the feedback signal in response to the two cues. Eventually, no cues were shown anymore in order to test how long the last cue may be maintained stably by the trained feedback. We term this phase of simulation the *stability test phase*. Simulation is canceled automatically if the feedback signal deviated beyond a certain error margin from its desired value.

Fig 6 illustrates the results of this experiment. Following the teacher forcing and testing of trained feedback, the stability test was run for a day of physical time. Afterwards, simulation was interrupted manually, because the error margin was never exceeded. At this point, the fDCR has maintained its memory of the cue for $\sim 336M$ reservoir time steps. The feedback signal shows slight downward and upward shift due to modulation by the random input, but it never shows overall drift away from the desired value. Instead, the feedback signal seems to maintain a constant average value with no time limit.

Discussion

As proven theoretically and confirmed through simulation by Maass and colleagues [25], trained feedback can overcome the limitations of fading memory in conventional reservoir computing. While the latter is modeled by a system of ordinary differential equations, here we show through simulation that this applies to single node Delay-Coupled Reservoirs, which are modeled by a single delay differential equation. The resulting fDCR (Fig 1A) successfully learns nonlinear long-term-memory-dependent tasks concurrently, and with high accuracy (Figs 3–5). We also show that memory storage is not only extended by an order of magnitude beyond fading memory, but is practically infinite (Fig 6). These simulations serve to demonstrate that with the added trained feedback, the fDCR combines sensitivity and stability. That is, the fDCR's high-dimensional dynamics consists of transient input-sensitive representations, and attractor states where stable memories are stored.

A few issues remain to be addressed in the future. First, all tasks (Figs 3–5) require long-term memory and knowledge of the current input value only, with no demands for fading memory (in the exception of the feedback signal, which requires computing the highly-nonlinear exclusive-nor operation between the current and previous cue values [30]). The performance for more complex tasks, for which the target function depends on both long-term memory and fading memory, remain to be explored. We expect, as in the case of conventional reservoirs, that more complex computations require more complex systems. However, while complexity in conventional reservoirs can be controlled by the number of neurons, simply increasing the number of virtual nodes in a DCR does not immediately lead to improvement. This is because increasing the number of virtual nodes within constant delay τ also increases cross-correlations, since the delays between virtual nodes become shorter. The complexity of a DCR can only be controlled by understanding the tight interplay between the number of virtual nodes and their location, the total delay, the mask structure, and the nonlinearity responsible of mixing past and current inputs [8].

Furthermore, the error margin was exceeded between the desired and target feedback signals in some longitudinal simulations (as in Fig 6). The feedback signal does approach the desired feedback value shortly after a cue. However, it directly starts drifting towards the second desired feedback value within a relatively short timespan of about 2000 reservoir time steps, which is much higher than the limits of fading memory. This drift may be due to suboptimal choice of reservoir parameters or to the mask structure, which may result in insufficient fDCR effective dimensionality to support both stable storage of cue signals and fading memory of input. No parameter optimization was carried out here, since the main goal of demonstrating the potentials of trained feedback was met. Parameter optimization methods are currently under development, and a technique for improving mask structure through plasticity is now available [8]. These tools could provide the way to circumvent the above issue of feedback signal drift.

The current results are only based on numerical simulations, while a rigorous proof of the universal computational power of fDCR remains to be shown. Following the same line of proof as in ODE based reservoirs [25] is not feasible, since sufficient analytical tools to deal with nonlinear delay differential equations are still unavailable [36]. A direct benefit to such analytical tools is providing a theoretical basis to the generalizability of the current findings to other nonlinearities. This, in its turn, is highly relevant to successful physical realizations of fDCRs with naturally occurring nonlinearities [33].

Finally, it is tempting to relate delay-based computational architectures such as the DCR to computational biology, especially that delays are abundant in nature. One noted similarity is that both the DCR and single neurons function on multiple time scales. A neuron receives, at different delays, hundreds of signals in the form of postsynaptic potentials (PSPs) from its afferents and integrates these subthreshold PSPs nonlinearly as action potentials emitted at a slower time scale. Similarly, the DCR nonlinearly integrates faster time scale activity of its virtual nodes to generate its output at a slower time scale. The correspondence, however, is not one-to-one, since the DCR, following a high level of activity, does not undergo a reset of its output. Given the current choice of saturating nonlinearity, the DCR acts more as a mean-field [37] or a firing rate model [38], rather than as a single spiking neuron, and DCRs with resetting nonlinearities similar to spiking neurons remain to be tested. In addition, computations at even slower time scales, corresponding to neural networks can be envisaged in a DCR setting by adding extra delay lines [39, 40] or coupling multiple DCRs to one another. In summary, the DCR architecture provides a fertile ground for studying neural computations based on delays, the harvest of which will occupy research for years to come.

Supporting Information

S1 Fig. The readout training procedure avoids overfitting the target. (A) Comparison between desired (blue) and observed (orange) fDCR output signal d_3 in Experiment 3 (zoomed-in, different run from Fig 5). The training procedure results in a readout that is both robust against outliers ($y < 1$) and is capable of tracking the desired target accurately. (B) Scatter plots of the target verses observed output for both training (yellow) and validation (brown) data sets. (C) Correlation coefficient between desired and observed fDCR output for both training (brown) and validation (yellow).
(TIF)

S2 Fig. Computing the product of random input and ramping feedback signal. (A) Comparison between desired (blue) and observed (orange) fDCR output signal $d(\bar{t}) = d_f^{\mathrm{tf}} \cdot |u^{\mathrm{arb}}(\bar{t})|$. (B) Scatter plots of the target verses observed output for both training (yellow) and validation (brown) data sets. (C) Correlation coefficient between desired and observed fDCR output for both training (brown) and validation (yellow).
(TIF)

Acknowledgments

The contributions of Johannes Schumacher in discussing this work and for providing the basic DCR simulation code are gratefully acknowledged, so are the fruitful discussions with the members of the PHOCUS consortium.

Author Contributions

Conceptualization: GP HT.

Data curation: ADK.

Formal analysis: ADK MK.

Funding acquisition: GP.

Investigation: ADK MK GP.

Methodology: ADK MK GP HT.

Project administration: GP HT.

Resources: GP.

Software: ADK MK.

Supervision: GP HT.

Validation: ADK MK.

Visualization: ADK MK HT.

Writing – original draft: ADK MK HT.

Writing – review & editing: GP HT.

References

1. Rumsey CC, Abbott LF. Synaptic democracy in active dendrites. J Neurophysiol. 2006; 96(5):2307–2318. doi: 10.1152/jn.00149.2006 PMID: 16837665

2. Gollo LL, Kinouchi O, Copelli M. Active dendrites enhance neuronal dynamic range. PLoS Comput Biol. 2009; 5(6):e1000402. doi: 10.1371/journal.pcbi.1000402 PMID: 19521531
3. Graupner M, Brunel N. Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. Proc Natl Acad Sci U S A. 2012; 109(10):3991–3996. doi: 10.1073/pnas.1109359109 PMID: 22357758
4. Izhikevich EM. Polychronization: computation with spikes. Neural Comput. 2006; 18(2):245–282. doi: 10.1162/089976606775093882 PMID: 16378515
5. Appeltant L, Soriano MC, Van der Sande G, Danckaert J, Massar S, Dambre J, et al. Information processing using a single dynamical node as complex system. Nat Commun. 2011; 2:468. doi: 10.1038/ncomms1476 PMID: 21915110
6. Schumacher J, Toutounji H, Pipa G. An Introduction to Delay-Coupled Reservoir Computing. In: Koprinkova-Hristova P, Mladenov V, Kasabov NK, editors. Artificial Neural Networks. vol. 4 of Springer Series in Bio-/Neuroinformatics. Springer International Publishing; 2015. p. 63–90. doi: 10.1007/978-3-319-09903-3_4
7. Turrigiano GG, Nelson SB. Homeostatic plasticity in the developing nervous system. Nat Rev Neurosci. 2004; 5(2):97–107. doi: 10.1038/nrn1327 PMID: 14735113
8. Toutounji H, Schumacher J, Pipa G. Homeostatic Plasticity for Single Node Delay-Coupled Reservoir Computing. Neural Comput. 2015; 27(6):1159–1185. doi: 10.1162/NECO_a_00737 PMID: 25826022
9. Jaeger H. The "echo state" approach to analysing and training recurrent neural networks. Bremen: German National Research Center for Information Technology; 2001.
10. Maass W, Natschläger T, Markram H. Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Comput. 2002; 14(11):2531–2560. doi: 10.1162/089976602760407955 PMID: 12433288
11. Buonomano DV, Maass W. State-dependent computations: spatiotemporal processing in cortical networks. Nat Rev Neurosci. 2009; 10(2):113–125. doi: 10.1038/nrn2558 PMID: 19145235
12. Lukoševičius M, Jaeger H. Reservoir computing approaches to recurrent neural network training. Computer Science Review. 2009; 3(3):127–149. doi: 10.1016/j.cosrev.2009.03.005
13. Häusler S, Maass W. A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. Cereb Cortex. 2007; 17(1):149–162. doi: 10.1093/cercor/bhj132 PMID: 16481565
14. Karmarkar UR, Buonomano DV. Timing in the absence of clocks: encoding time in neural network states. Neuron. 2007; 53(3):427–438. doi: 10.1016/j.neuron.2007.01.006 PMID: 17270738
15. Yamazaki T, Tanaka S. The cerebellum as a liquid state machine. Neural Netw. 2007; 20(3):290–297. doi: 10.1016/j.neunet.2007.04.004 PMID: 17517494
16. Nikolić D, Häusler S, Singer W, Maass W. Distributed fading memory for stimulus properties in the primary visual cortex. PLoS Biol. 2009; 7(12):e1000260. doi: 10.1371/journal.pbio.1000260 PMID: 20027205
17. Jaeger H, Haas H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science. 2004; 304(5667):78–80. doi: 10.1126/science.1091277 PMID: 15064413
18. Jaeger H, Lukoševičius M, Popovici D, Siewert U. Optimization and applications of echo state networks with leaky-integrator neurons. Neural Netw. 2007; 20(3):335–352. doi: 10.1016/j.neunet.2007.04.016 PMID: 17517495
19. Dasgupta S, Wörgötter F, Manoonpong P. Information dynamics based self-adaptive reservoir for delay temporal memory tasks. Evolving Systems. 2013; 4(4):235–249. doi: 10.1007/s12530-013-9080-y
20. Pearlmutter BA. Learning State Space Trajectories in Recurrent Neural Networks. Neural Comput. 1989; 1(2):263–269. doi: 10.1162/neco.1989.1.2.263
21. Williams RJ, Zipser D. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. Neural Comput. 1989; 1(2):270–280. doi: 10.1162/neco.1989.1.2.270
22. Sussillo D, Abbott LF. Generating Coherent Patterns of Activity from Chaotic Neural Networks. Neuron. 2009; 63(4):544–557. doi: 10.1016/j.neuron.2009.07.018 PMID: 19709635
23. Sussillo D, Abbott LF. Transferring learning from external to internal weights in echo-state networks with sparse connectivity. PLoS One. 2012; 7(5):e37372. doi: 10.1371/journal.pone.0037372 PMID: 22655041
24. Schumacher J, Toutounji H, Pipa G. An Analytical Approach to Single Node Delay-Coupled Reservoir Computing. In: Mladenov P, Koprinkova-Hristova V, Palm G, Villa AEP, Appollini B, Kasabov N, editors. Artificial Neural Networks and Machine Learning–ICANN 2013. vol. 8131 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2013. p. 26–33. doi: 10.1007/978-3-642-40728-4_4

25. Maass W, Joshi P, Sontag ED. Computational aspects of feedback in neural circuits. PLoS Comput Biol. 2007; 3(1):e165. doi: 10.1371/journal.pcbi.0020165 PMID: 17238280

26. Pascanu R, Jaeger H. A neurodynamical model for working memory. Neural Netw. 2011; 24(2):199–207. doi: 10.1016/j.neunet.2010.10.003 PMID: 21036537

27. Rodan A, Tiňo P. Minimum complexity echo state network. IEEE Trans Neural Netw. 2011; 22(1):131–144. doi: 10.1109/TNN.2010.2089641 PMID: 21075721

28. Lazar A, Pipa G, Triesch J. Fading memory and time series prediction in recurrent networks with different forms of plasticity. Neural Netw. 2007; 20(3):312–322. doi: 10.1016/j.neunet.2007.04.020 PMID: 17556114

29. Lazar A, Pipa G, Triesch J. SORN: a self-organizing recurrent neural network. Front Comput Neurosci. 2009; 3(23). doi: 10.3389/neuro.10.023.2009 PMID: 19893759

30. Toutounji H, Pipa G. Spatiotemporal Computations of an Excitable and Plastic Brain: neuronal plasticity Leads to Noise-Robust and Noise-Constructive Computations. PLoS Comput Biol. 2014; 10(3): e1003512. doi: 10.1371/journal.pcbi.1003512 PMID: 24651447

31. Larger L, Soriano M, Brunner D, Appeltant L, Gutiérrez JM, Pesquera L, et al. Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. Opt Express. 2012; 20(3):3241–3249. doi: 10.1364/OE.20.003241 PMID: 22330562

32. Paquot Y, Duport F, Smerieri A, Dambre J, Schrauwen B, Haelterman M, et al. Optoelectronic reservoir computing. Sci Rep. 2012; 2. doi: 10.1038/srep00287 PMID: 22371825

33. Brunner D, Soriano MC, Mirasso CR, Fischer I. Parallel photonic information processing at gigabyte per second data rates using transient states. Nature Commun. 2013; 4:1364. doi: 10.1038/ncomms2368 PMID: 23322052

34. Boyd S, Chua LO. Fading memory and the problem of approximating nonlinear operators with Volterra series. IEEE Trans Circuits Syst. 1985; 32(11):1150–1161. doi: 10.1109/TCS.1985.1085649

35. Glass L, Mackey M. Mackey-Glass equation. Scholarpedia. 2010; 5(3):6908. doi: 10.4249/scholarpedia.6908

36. Guo S, Wu J. Bifurcation theory of functional differential equations. Springer New York; 2013. doi: 10.1007/978-1-4614-6992-6

37. Brunel N. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. J Comput Neurosci. 2000; 8(3):183–208. doi: 10.1023/A:1008925309027 PMID: 10809012

38. Pasemann F. Dynamics of a single model neuron. Int J Bifurcat Chaos. 1993; 03(02):271–278. doi: 10.1142/S0218127493000210

39. Martinenghi R, Rybalko S, Jacquot M, Chembo YK, Larger L. Photonic nonlinear transient computing with multiple-delay wavelength dynamics. Phys Rev Lett. 2012; 108(24):244101. doi: 10.1103/PhysRevLett.108.244101 PMID: 23004274

40. Soriano MC, Brunner D, Escalona-Morán M, Mirasso CR, Fischer I. Minimal approach to neuro-inspired information processing. Front Comput Neurosci. 2015; 9(68). doi: 10.3389/fncom.2015.00068 PMID: 26082714