entropy

Article

Reservoir Computing with Delayed Input for Fast and Easy Optimisation

Lina Jaurigue [1,*], Elizabeth Robertson [2,3], Janik Wolters [2,3] and Kathy Lüdge [4]

[1] Institute of Theoretical Physics, Technische Universität Berlin, Hardenbergstr. 36, 10623 Berlin, Germany
[2] Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Institut fur Optische Sensorsysteme, Rutherfordstr. 2, 12489 Berlin, Germany; Elizabeth.Robertson@dlr.de (E.R.); janik.wolters@tu-berlin.de (J.W.)
[3] Institut für Optik und Atomare Physik, Technische Universität Berlin, 10623 Berlin, Germany
[4] Institute of Physics, Technische Universität Ilmenau, Weimarer Str. 25, 98693 Ilmenau, Germany; kathy.luedge@tu-ilmenau.de
* Correspondence: linajaurigue@tu-berlin.de

Abstract: Reservoir computing is a machine learning method that solves tasks using the response of a dynamical system to a certain input. As the training scheme only involves optimising the weights of the responses of the dynamical system, this method is particularly suited for hardware implementation. Furthermore, the inherent memory of dynamical systems which are suitable for use as reservoirs mean that this method has the potential to perform well on time series prediction tasks, as well as other tasks with time dependence. However, reservoir computing still requires extensive task-dependent parameter optimisation in order to achieve good performance. We demonstrate that by including a time-delayed version of the input for various time series prediction tasks, good performance can be achieved with an unoptimised reservoir. Furthermore, we show that by including the appropriate time-delayed input, one unaltered reservoir can perform well on six different time series prediction tasks at a very low computational expense. Our approach is of particular relevance to hardware implemented reservoirs, as one does not necessarily have access to pertinent optimisation parameters in physical systems but the inclusion of an additional input is generally possible.

Keywords: reservoir computing; time series prediction; performance optimisation

Citation: Jaurigue, L.; Robertson, E.J.; Wolters, J.; Lüdge, K. Reservoir Computing with Delayed Input for Fast and Easy Optimisation. *Entropy* **2021**, *23*, 1560. https://doi.org/10.3390/e23121560

Academic Editors: Serge Massar, Guy Van der Sande and Piotr Antonik

Received: 31 October 2021
Accepted: 21 November 2021
Published: 23 November 2021

1. Introduction

Reservoir computing (RC) is a machine learning method that is particularly suited to solving dynamical tasks [1]. It was introduced as a way of using recurrent networks for machine learning but circumventing the costly training of the network weights [2]. The main principle underpinning reservoir computing is that the reservoir projects the inputs into a sufficiently high dimensional phase space such that it suffices to linearly sample the response of the reservoir in order to approximate the desired target for a given task. For this to work, the reservoir must fulfil certain criteria: the response to sufficiently different inputs must be linearly separable, the reservoir must be capable of performing nonlinear transforms, and the reservoir must have the fading memory property [2]. However, even when these criteria are fulfilled, the performance depends greatly on the dynamics of the reservoir. Hence, in the past two decades a lot of research in the reservoir computing community has focused on the optimisation of the reservoir parameters [3–9]. Furthermore, the optimisation of the reservoir is a task-specific problem [1,10–12] and a universal reservoir, which performs well in a range of tasks, remains elusive.

In a recent paper [13], the authors aim to eliminate the issue of hyperparameter optimisation altogether by removing the reservoir. Their approach essentially takes the well-known nonlinear vector autoregression (NVAR) method, uses a less parsimonious approach to filling the feature vector, and adds Tikhonov regularisation. However, the method of [13] trades the optimisation of the reservoir hyperparameters for the optimisation of the feature vector elements and it cannot be asserted that the latter is generally less costly. Furthermore, one of the main factors driving research into reservoir computing forward is the possibility for hardware implementation [14–19], which is impractical when the reservoir is absent.

In this contribution we demonstrate a new approach that reduces the need for hyperparameter optimisation and is well suited to boosting the performance of physically implemented reservoir computers. Specifically, we show that, by adding a time-delayed version of the input for a given task, the performance of an unoptimised reservoir can be greatly improved. We demonstrate this by using one unaltered reservoir to perform six different time series prediction tasks. In each case the only optimisation parameters are the delay and input strength of the additional delayed input. The aim of this work is not to achieve the best possible performance, but rather to demonstrate that reasonable performance can be achieved for various tasks using the same reservoir and at a very low computational cost.

Using time-delayed input is a common approach for adding memory to feedforward networks [20–23] and is the basis of statistical forecasting methods [21,24]. However, despite the simplicity of this idea, to the best of our knowledge, time-delayed inputs have not been widely used to optimise the performance of reservoir computers. This may be because the focus has been on constructing reservoirs that have the necessary memory to perform a given task [1]. One study in which time-delayed inputs have been used to improve the performance of a time series prediction task is [25]. However, in [25], the manner in which the time-delayed input was constructed assumed that the memory requirements of the task monotonically decrease with increasing steps into the past and did not allow for the input scaling of the delayed input to be varied as a free parameter.

Our results are of particular relevance to the hardware implementation of reservoir computing, because in physical systems one does not always have access to the relevant hyperparameters necessary for optimisation of the task-dependent performance but it should always be possible to add an additional input.

2. Methods

In the following, we describe the reservoir computing concept, the model for the reservoir that we use, our proposed time-delayed input method, and the benchmarking tasks that are used to test our approach.

2.1. Reservoir Computing

In reservoir computing, the reservoir, which at this point can be treated as a black box, is fed an input and the response of the system is sampled a number of times. The responses are then linearly combined to approximate the desired output (see Figure 1a). The linear output weights are trained via linear regression, typically using Tikhonov regularisation or regularisation by noise [1]. A variant of reservoir computing, that is of particular relevance for hardware implementation, is time-multiplexed reservoir computing using only one nonlinear element [26]. In this scheme both the injection of the data into the reservoir and the filling of state matrix $\underline{\underline{\mathbf{S}}}$ occur sequentially. Typically, a mask is applied to the input data in order to diversify the response of the reservoir to the input. In the training phase, the reservoir is fed a sequence of training data of length K_{tr}. A mask of length N_v is applied to each element of the training data, where N_v corresponds to the readout dimension (i.e., the number virtual nodes). Hence, there are $N_v K_{\text{tr}}$ time-multiplexed inputs that are sequentially fed into the reservoir. The corresponding state matrix, which has the dimensions $K_{\text{tr}} \times (N_v + 1)$, is filled row by row with an additional bias term of 1 at the end of each row. The training step is then to find the $(N_v + 1)$ dimensional weight vector $\underline{\mathbf{W}}$ that best approximates

$$\underline{\hat{\mathbf{o}}} \approx \underline{\underline{\mathbf{S}}} \cdot \underline{\mathbf{W}}, \tag{1}$$

where $\underline{\hat{\mathbf{o}}}$ is the vector of K_{tr} target outputs. The solution to this linear problem is given by

$$\underline{\mathbf{W}} = \left(\underline{\underline{\mathbf{S}}}^{\mathbf{T}}\underline{\underline{\mathbf{S}}} + \lambda \underline{\underline{\mathbf{I}}}\right)^{-1} \underline{\underline{\mathbf{S}}}^{\mathbf{T}}\underline{\hat{\mathbf{o}}}, \tag{2}$$

where λ is the Tikhonov regularisation parameter and $\underline{\underline{\mathbf{I}}}$ is the identity matrix.

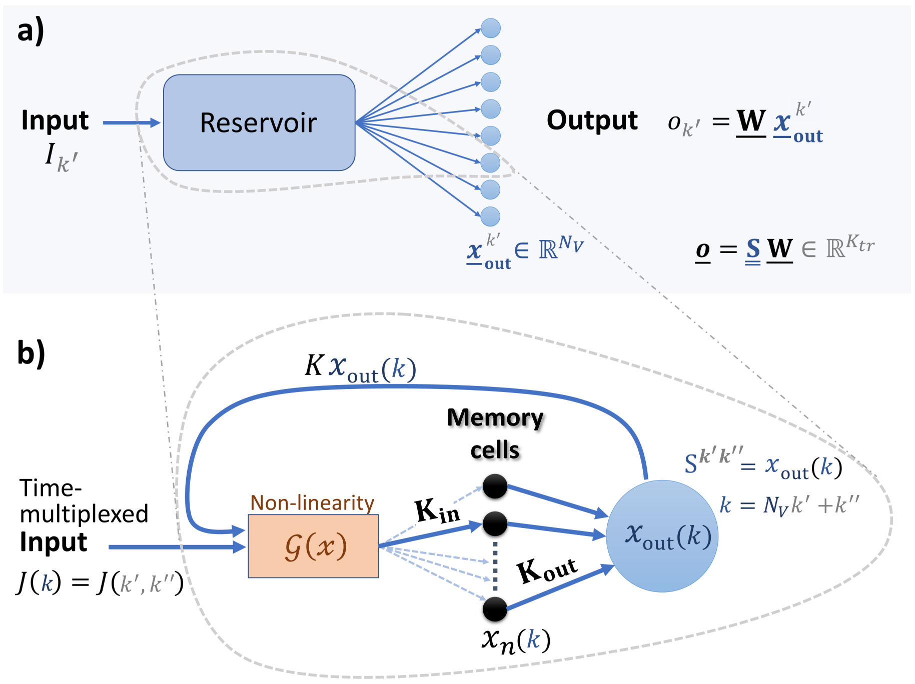
a)
Input
$I_{k'}$
Reservoir
Output $o_{k'} = \underline{\mathbf{W}}\,\underline{x}^{k'}_{\text{out}}$
$\underline{x}^{k'}_{\text{out}} \in \mathbb{R}^{N_V}$
$\underline{o} = \underline{\underline{S}}\,\underline{\mathbf{W}} \in \mathbb{R}^{K_{tr}}$
b)
$K x_{\text{out}}(k)$
Memory cells
Time-multiplexed Input
$J(k) = J(k', k'')$
Non-linearity
$\mathcal{G}(x)$
$\mathbf{K}_{\text{in}}$
$\mathbf{K}_{\text{out}}$
$x_n(k)$
$x_{\text{out}}(k)$
$S^{k'k''} = x_{\text{out}}(k)$
$k = N_V k' + k''$

Figure 1. (**a**) Sketch of the reservoir computing concept. The vector $\underline{\mathbf{x}}^{k'}_{\text{out}}$ is the responses of the reservoir to an input $I_{k'}$ and the corresponding output $o_{k'}$ is generated by a weighted sum of these responses. The read-out weights $\underline{\mathbf{W}}$ are trained. (**b**) Sketch of the memory cell reservoir described in Section 2.2, where a time-multiplexed input $J(k)$ is fed into the reservoir ($k = N_v k' + k''$, please see Figure 2 for the construction of the time-multiplexed input). The index n labels the memory cells (in total N) that are addressed via the coupling matrices $\mathbf{K_{in}}$ and $\mathbf{K_{out}}$. K labels the feedback strength at which the output of the memory cells $x_{\text{out}}(k)$ (given by Equation (5)) is fed back into the nonlinearity $\mathcal{G}$. The elements of the state matrix $\underline{\underline{\mathbf{S}}}$ are given by $S^{k',k''} = x_{\text{out}}(N_v k' + k'')$, with $k' \in (1 \ldots K_{tr})$ and $k'' \in (1 \ldots N_v)$, i.e., one row of $\underline{\underline{\mathbf{S}}}$ corresponds to the vector of responses $\underline{\mathbf{x}}^{k'}_{\text{out}}$ in (**a**).

Error Measure

To quantify the performance of the reservoir computer we use the normalised root mean squared error (NRMSE), defined as

$$\text{NRMSE} = \sqrt{\frac{\sum_{k'=1}^{K_o} (\hat{o}_{k'} - o_{k'})^2}{K_o \text{var}(\underline{\hat{\mathbf{o}}})}}, \tag{3}$$

where $\hat{o}_{k'}$ are the target values, $o_{k'}$ are the outputs produced by the reservoir computer, K_o is the length of the vector $\underline{\hat{\mathbf{o}}}$, and $\text{var}(\underline{\hat{\mathbf{o}}})$ is the variance of the target sequence.

2.2. Reservoir Model

To investigate the effect of delayed input on a physically implemented reservoir computer, we model a physical system that is inspired by optical delay line reservoirs [27,28]. Delay line implementations have shown promise due to high throughput speeds [29]. However, complex network connectivity, achieved via the introduction of multiple delays, represents a significant experimental hurdle, or requires opto-electrical conversion of the signal for electronic storage, thereby forgoing the advantages of an all-optical implementation. Recent developments in optical quantum memories with high bandwidth [30] and high capacity [31] allow for the on-demand storage and retrieval of optical pulses and thus the implementation of delays of arbitrary length, limited only by the coherence time of the optical memory, which can reach up to one second [32]. The reservoir model described below models a physical optical system including the optical memory for the reconfigurable and arbitrary coupling of the injected information (modeled as memory cells with input and output coupling), a nonlinear element (modeled as a semiconductor optical amplifier), and a short delay line, whose purpose is not for introducing delay, but to recouple existing information back into the optical system. A sketch of the envisaged setup is shown in Figure 1b. A time-multiplexed input is fed through a nonlinear element and then stored in the memory cells (described with the index n) with a certain input topology $\mathrm{K}_{\mathrm{in}}^{n}$. Combinations of the memory cells are partially read out with the output topology $\mathrm{K}_{\mathrm{out}}^{n}$ and finally coupled back into the nonlinear element. Since the write and read-out process repeats in time, it is possible to realise time-varying read and write topologies. We describe this by adding the index $m = k(\mathrm{mod}\ M)$, where M is the period within which the coupling sequence repeats, giving the coupling matrix elements $\mathrm{K}_{\mathrm{in}}^{mn}$ and $\mathrm{K}_{\mathrm{out}}^{mn}$. The map describing this process is given by the following. Let $x_n(k)$ be the state of the n^{th} memory cell at time step k. The next time step is then given by

$$x_n(k+1) = \mathrm{K}_{\mathrm{in}}^{mn}\mathcal{G}(Kx_{out}(k) + J(k+1)) + \mathrm{K}_{\mathrm{nin}}^{mn}(1 - \mathrm{K}_{\mathrm{out}}^{mn})x_n(k), \tag{4}$$

where

$$x_{out}(k) = \sum_{n=1}^{N} \mathrm{K}_{\mathrm{out}}^{mn} x_n(k). \tag{5}$$

$\mathcal{G}(x)$ is the function describing the nonlinear element, and the matrices $\mathbf{K_{in}}$, $\mathbf{K_{nin}}$, and $\mathbf{K_{out}}$ describe the (possibly time-varying) coupling into and out of the memory cells. The value of K describes the percentage of the output $x_{out}(k)$ that is coupled back into the nonlinear element and $J(k)$ is the input. The coupling matrices have the dimensions MxN, where N is the number of memory cells. Note that the index $m = k(\mathrm{mod}\ M)$ depends on the time step k. For each iteration one row of the coupling matrices determines which memory cells are written into and which are read out of. $\mathbf{K_{in}}$ gives the write sequence and $\mathbf{K_{out}}$ the out-coupling sequence. These two matrices contain values from zero to one. For $\mathbf{K_{in}}$, the row sum must be one. The entries of the matrix $\mathbf{K_{nin}}$ are

$$\mathrm{K}_{\mathrm{nin}}^{mn} = \begin{cases} 0 \text{ if } \mathrm{K}_{\mathrm{in}}^{mn} \neq 0 \\ 1 \text{ if } \mathrm{K}_{\mathrm{in}}^{mn} = 0 \end{cases}.$$

This allows the memory cells with new input to be overwritten and those without to be updated according to how much was read out.

The model described above allows for arbitrary coupling between the memory cells. For this study, we choose $\mathbf{K_{in}} = \mathbf{K_{out}} = \underline{\underline{\mathbf{I}}}$, and $M = N$. This means, at every input cycle, one memory cell is overwritten and one is read out. For this choice of coupling, Equations (4) and (5) can be rewritten as

$$x_{out}(k+1) = \mathcal{G}(Kx_{out}(k - N + 1) + J(k+1)). \tag{6}$$

We then choose $N = N_v + 1$ where N_v is the number of virtual nodes that will be used for the reservoir computing tasks. This coupling describes a type of ring coupling akin to delay-based reservoir computers with the feedback delay time $\tau = T + \theta$, where T is the input clock-cycle and θ is the virtual node separation [28,33]. Comparing the continuous and discrete cases gives $\tau \to N$, $T \to N_v$, and $\theta \to 1$. We choose such a simple coupling scheme as it has been demonstrated that such coupling topologies perform similarly to random coupling topologies [4]. Using Equation (6), the rows of the state matrix $\underline{\underline{\mathbf{S}}}$ are filled with N_v sequential $x_{out}(k)$, i.e., $S^{k',k''} = x_{out}(N_v k' + k'')$ and the bias $S^{k',(N_v+1)} = 1$ (see Figure 1 for an illustration).

For the nonlinearity, we choose

$$\mathcal{G}(x) = \frac{g_0 x}{1 + x/P_{\text{sat}}}, \tag{7}$$

which describes the input response of a semiconductor optical amplifier [34,35].

2.3. Input and Mask

The reservoir input is given by a task-dependent time series and a time-delayed version of this time series. Before the data are fed into the reservoir, masks are applied to both input series. The masks consist of N_v values drawn from a uniform distribution between 0 and 1. The final input is then given by

$$J(k) = G_1 I(k') M_1(k'') + G_2 I(k' - d) M_2(k'') + J_0, \tag{8}$$

where G_1 and G_2 are the input scaling factors, $I(k')$ is the input time series, d is the input delay, $M_1(k'')$ and $M_2(k'')$ are k''th entries of the N_v dimensional masking vectors, and J_0 is a constant offset. A sketch of the masked input sequence is shown in Figure 2.

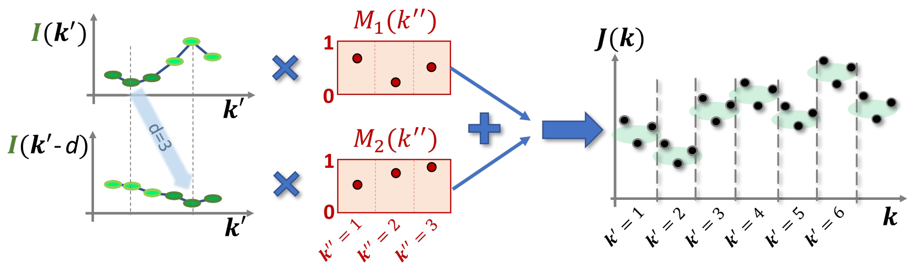

Figure 2. Sketch of the generation of the final time-multiplexed input sequence $J(k)$ using the task-dependent input $I(k')$, a delayed version of this input $I(k' - d)$, and the masks $M_1(k'')$ and $M_2(k'')$, as described by Equation (8).

2.4. Time Series Prediction Tasks

2.4.1. Mackey–Glass

The Mackey–Glass equation is a delay differential equation which exhibits chaotic dynamics. The reservoir computing benchmarking task is to predict the time series s number of steps ahead in the chaotic regime. The Mackey–Glass equation is [36]:

$$\frac{dx}{dt} = \beta \frac{x(t-\tau)}{1 + x(t-\tau)^n} - \gamma x. \tag{9}$$

We use the standard parameters: $\tau = 17$, $n = 10$, $\beta = 0.2$, and $\gamma = 0.1$. To create the input sequence $I(k')$, the time series generated by Equation (9) is sampled with a time step of $dt = 1$. The corresponding target sequence is then given by $I(k' + s)$.

2.4.2. NARMA10

NARMA10 is a commonly used benchmarking task that is defined by the iterative formula

$$A_{n+1} = 0.3A_n + 0.05A_n \left(\sum_{i=0}^{9} A_{n-i} \right) + 1.5u_{n-9}u_n + 0.1, \tag{10}$$

where u_n are identically and independently drawn random numbers from a uniform distribution in the interval $[0, 0.5]$ [37]. The reservoir input sequence $I(k')$ is given by the sequence of u_n and the target sequence is given by the corresponding A_n.

2.4.3. Lorenz

The Lorenz system [38] is given by

$$\frac{dx}{dt} = c_1 y - c_1 x, \quad \frac{dy}{dt} = x(c_2 - z) - y, \quad \text{and} \quad \frac{dz}{dt} = xy - c_3 z. \tag{11}$$

With $c_1 = 10$, $c_2 = 28$ and $c_3 = 8/3$, this system exhibits chaotic dynamics. We use the x variable, sampled with a step size of $dt = 0.02$, as the input $I(k')$ for two time series prediction tasks. The first is one step ahead ($s = 1$) prediction of the x variable. The second is one step ahead ($s = 1$) cross-prediction of the z variable.

2.5. Simulation Conditions

For all tasks, the reservoir is initialised with an input sequence $I(k')$ of length 10,000. The system is then trained on $K_{\text{tr}} = 10{,}000$ inputs. This is followed by another buffer of 10,000 inputs, before the performance is tested on a sequence of $K_{\text{te}} = 5000$ inputs, unless stated otherwise. For each task, the reservoir parameters are kept identical and are as given in Table 1. The input scaling of the primary input G_1 (nondelayed input) and the offset J_0 are scaled such that the input range for each task is approximately [0.4, 1.3]. The scaling of the delayed input G_2 and the input delay d are used as the optimisation parameters. For each task, the performance is averaged over 100 realisations of the random masks and also, in the case of NARMA10, the random inputs.

Table 1. Reservoir and input parameter values.

Parameter	Value	Parameter	Value
g_0	40	P_{sat}	1
K	0.02	N	31
N_v	30	λ	5×10^{-6}
G_1 (Mackey–Glass)	1	J_0 (Mackey–Glass)	0
G_1 (Lorenz)	0.03	J_0 (Lorenz)	0.85
G_1 (NARMA10)	1.8	J_0 (NARMA10)	0.4

3. Results

The performance of the reservoir with additional delayed input is tested on six tasks; we first consider Mackey–Glass time series prediction for one, three, and ten steps into the future. The results of the Mackey–Glass tasks, and their relation to the delayed input parameters, are depicted in Figure 3a–c. By inspecting the evolution of the performance error as a function of d and G_2, an optimal performance and thus an optimal value for d can be identified (brightest light yellow region). This value, however, depends on the task and thus changes in between the panels. In order to quantify the impact of the delayed-input strength on the performance, we present scans of the delayed-input strength G_2 for the optimal input delay d, for each task, in Figure 4a–c. $G_2 = 0$ corresponds to the system without delayed input and should be used as the reference to quantify the performance boost due to the delayed input. For each of the three cases, the delayed input leads to a reduction in the NRMSE, ranging from 20% for $s = 1$ to over a factor three for $s = 10$. The optimal values for the delay and the input scaling G_2 vary depending on the number of steps s predicted into the future. In agreement with the results presented in [39], larger input scaling is required as s increases, indicating that nonlinear transforms become increasingly important. In terms of the absolute performance, similar results are achieved compared with other studies [39,40], despite the number of virtual nodes used in this study being significantly lower.

The results for the NARMA10 task are shown in Figures 3d and 4d. Without the delayed input ($G_2 = 0$) the performance of the reservoir is very poor. This is in contrast to the Mackey–Glass $s = 1$ for which the performance without delayed input (Figure 3a with $G_2 = 0$) is reasonable. Moreover, this finding supports the general observation that reservoir computers have to be optimised to individual tasks and perform poorly as universal approximators [1,10,11]. The inclusion of delayed input significantly reduced the NARMA10 error, reaching an NRMSE of about 0.3 for the input delay $d = 9$. In absolute terms, an NRMSE of 0.3 is within the range of typically quoted best values (NRMSE = 0.15–0.4) [4,10,41–44], however, it is usually achieved with a much higher output dimension than the $N_v = 30$ used here. The performance achieved in this study came at a very low computational cost. As a comparison, in [44] the authors investigate the influence of combining echo state networks with different timescales and achieve a best performance of just under 0.4 for the NRMSE, at a greater computational expense.

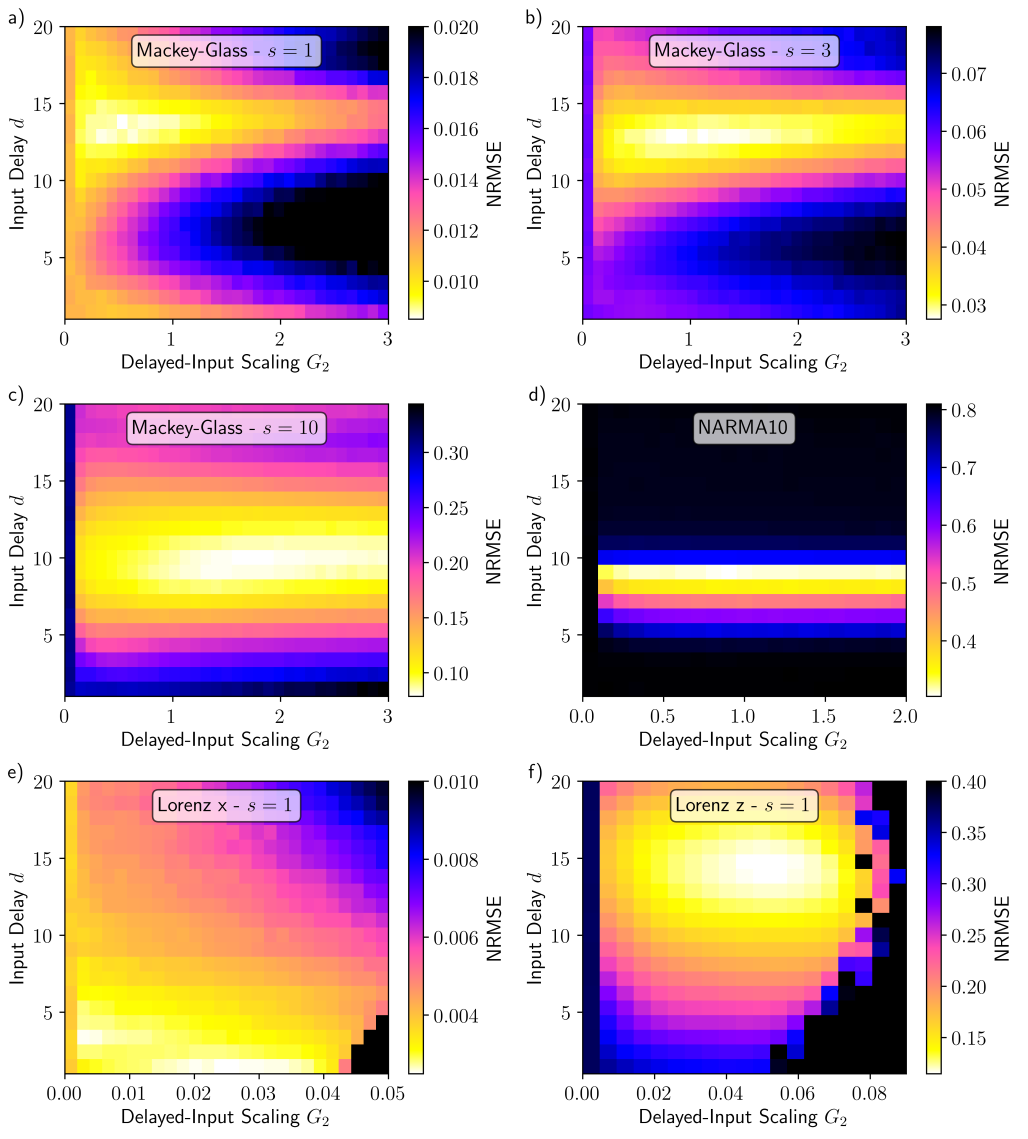

Figure 3. NRMSE as a function of the delayed input parameters d and G_2 for Mackey–Glass (**a**) one, (**b**) three, and (**c**) ten steps ahead prediction, (**d**) NARMA10, (**c**) Lorenz x one step ahead prediction, and (**f**) Lorenz z one step ahead cross-prediction. Parameters are as stated in Section 2.5, except for (**a**,**e**) where $K_{\mathrm{te}} = 30,000$.

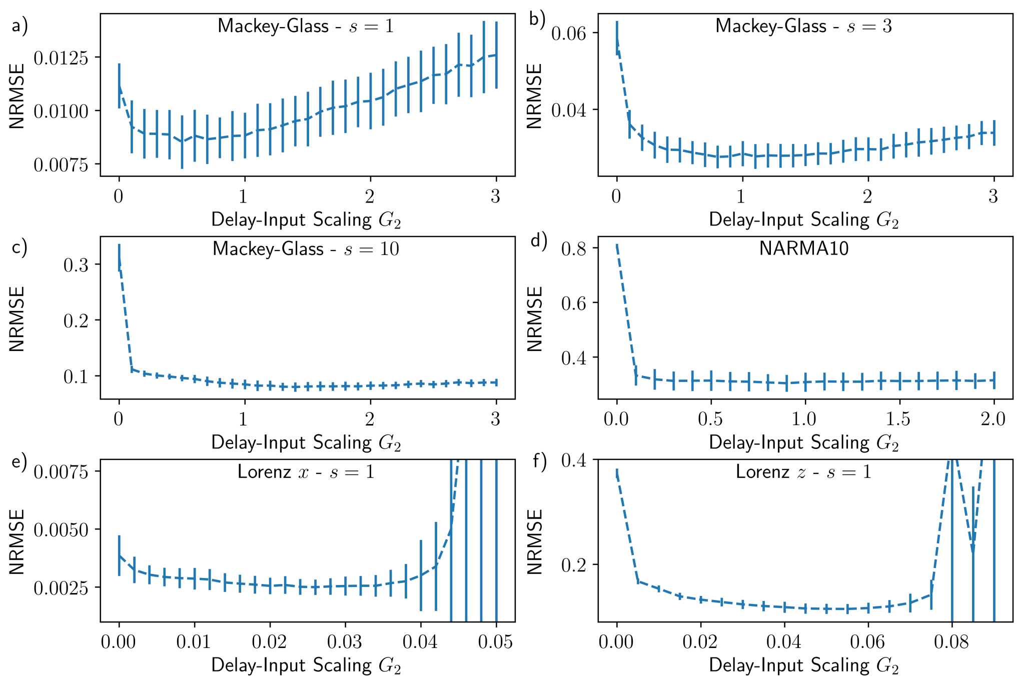

Figure 4. NRMSE for optimised input delay d, as a function of the delayed-input scaling G_2 for Mackey–Glass (**a**) one, (**b**) three, and (**c**) ten steps ahead prediction, (**d**) NARMA10, (**c**) Lorenz x one step ahead prediction, and (**f**) Lorenz z one step ahead cross-prediction. The error bars indicate the standard deviation. The optimal input delays are (**a**) $d = 14$, (**b**) $d = 13$, (**c**) $d = 9$, (**d**) $d = 9$, (**e**) $d = 1$, and (**f**) $d = 15$. The remaining parameters are as stated in Section 2.5, except for (**a**,**e**) where $K_{\text{te}} = 30,000$.

The remaining two tasks are one step ahead Lorenz x prediction and one step head Lorenz z cross-prediction, the results of which are shown in Figures 3e,f and 4e,f. In both cases there is an improvement in the performance with the correct choice of the delayed input. It is has been demonstrated that the Lorenz x one step ahead prediction requires only the very recent history of the x-variable time series [13], and we find the optimal input delay of $d = 1$ to be consistent with this prior knowledge. For the Lorenz z cross-prediction task, on the other hand, there is a strong dependence on the history of the Lorenz x variable. In this case, the best performance is achieved when the second input is delayed by $d = 14$ time steps. The optimal delayed-input scaling G_2 is larger for the Lorenz z task than the Lorenz x task (as seen by comparing the positions of the minima in Figure 4e,f), indicating that the cross-prediction task requires a greater degree of nonlinearity as well as a longer memory.

In order to demonstrate that the improvement in the performance with delayed input is not specific to the reservoir parameters used for Figure 3, in Figure 5 we show the NRMSE for the Mackey–Glass $s = 10$ task as a function of (a) the virtual node coupling strength K and (b) the coupling delay N (i.e., the number of memory cells). These parameters have a strong influence on the properties of the reservoir. In both cases the NRMSE without delayed input (orange dotted line) shows a large variation over the respective parameter ranges and is always larger than the error with optimised delayed input (blue dashed line). With optimised delayed input the variation in the error is comparatively small, demonstrating that the inclusion of the delayed input works well independent of the reservoir properties. The peak in the NRMSE at $N = 30$ in Figure 5b is a well-known resonance effect that occurs at resonances between the number of virtual nodes and the coupling range N, equivalent to clock time and delay resonances in time continuous systems [45].

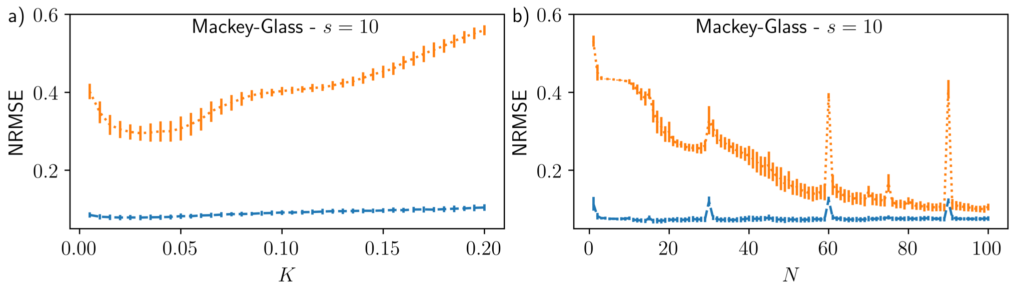

Figure 5. NRMSE for Mackey–Glass 10 step ahead prediction as a function of (**a**) the virtual node coupling strength K and (**b**) the coupling delay N. The orange dotted (blue dashed) lines show the results without (with) delayed input. Along the blue curve the delayed input parameters d and G_2 have been optimised (see Figure A1 in Appendix A for their values). The error bars indicate the standard deviation. All remaining parameters are as stated in Section 2.5.

To further demonstrate the universality of this method, we show the NARMA10 error with delayed input for a time continuous reservoir in Figure 6. In this case the reservoir is given by the Stuart–Landau equation with time-delayed feedback (see Appendix B). The reservoir parameters have not been optimised for the NARMA10 task, resulting in very poor performance without delayed input ($G_2 = 0$). With optimised delayed-input parameters reasonable performance is achieved, similar to the optimal results for the memory cell reservoir in Figure 3d. For the Stuart–Landau reservoir, optimal performance is achieved for the input delay $d = 10$, whereas, for the memory cell reservoir, the optimal input delay is $d = 9$. This is because the required input delay depends both on the dynamics of the reservoir as well as the memory requirements of the particular task.

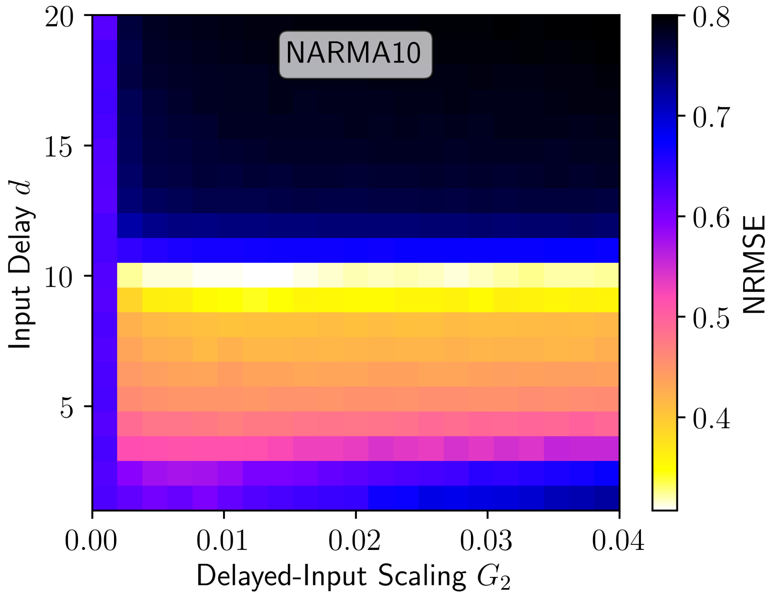

Figure 6. NRMSE for the NARMA10 task as a function of the delayed input parameters d and G_2 using the Stuart–Landau delay-based reservoir computer described in Appendix B.

4. Discussion

We have shown that, for various time series prediction tasks, including a delayed version of the input can lead to a substantial improvement in the performance of a reservoir. We have demonstrated this using a simple map describing a semiconductor optical amplifier nonlinearity and a ring-like coupling realised via memory cells. With this approach we were able to use one unaltered reservoir to perform well on six different tasks, each with different memory and nonlinear transform requirements. The performance boost due to the delayed input is achieved over a wide range of the reservoir parameters and was also demonstrated for a time continuous system, indicating that our approach is applicable to a wide range of reservoirs.

Our results are significant for a number of reasons. Firstly, we have demonstrated that computationally expensive hyperparameter optimisation can be circumvented by tuning only two input parameters. By including an additional delayed input, reasonable performance can be achieved using an unoptimised reservoir. Nevertheless, we note that, depending on the requirements for a given task, additional hyperparameter optimisation may be necessary. Secondly, to the best of our knowledge, this is the first demonstration of an identical reservoir performing well on such a large range of tasks. Thirdly, the simplicity of our approach means that it is well suited to be applied on physical reservoirs.

This study has raised several questions surrounding delay-based reservoir optimisation that require further investigation. For example, the optimal delayed-input parameters are task dependent and how these relate to a given task is not fully understood. The NARMA10 results presented in this study indicate that the optimal delayed-input parameters are related both to the reservoir and requirements of the task. This means that it may be possible to not only use reservoir computing for real-world time series prediction tasks, but also to gain insights into the dynamical systems being investigated. For example, in tasks such as El Niño prediction where the underlying dynamical system is very complex and the relevant physical processes are not fully understood [46]. Here, investigations surrounding delay-based input could provide critical insight into the involved timescales. Furthermore, the minimum requirements for a reservoir to yield good performance on a range of tasks by only tuning the delayed input parameters remain to be determined.

A natural extension of our proposed approach is to include multiple delayed input terms. This would bring the reservoir computing approach closer to classical statistical forecasting methods such as NVAR and could lead to a further improved performance, especially for tasks involving multiple disparate timescales. However, possible performance improvement with added input terms must be weighed against the associated increase in the computational cost as each added input adds two new optimisation parameters.

Author Contributions: Conceptualisation, L.J., J.W. and K.L.; methodology, L.J.; software, L.J.; validation, L.J.; formal analysis, L.J.; investigation, L.J. and E.R.; writing—original draft preparation, L.J.; writing—review and editing, L.J., E.R., J.W. and K.L.; visualisation, L.J. and K.L.; funding acquisition, L.J., J.W. and K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deutsche Forschungsgemeinschaft (DFG), grant number LU 1729/3-1. E.R. acknowledges funding through the Helmholtz Einstein International Berlin Research School in Data Science (HEIBRiDS).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable

Data Availability Statement: The datasets generated and analysed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RC	Reservoir computing
NVAR	Nonlinear vector autoregression
NRMSE	Normalised root mean squared error

Appendix A. Optimised Input Parameters

The optimised values of the delayed-input scaling d and the input delay d corresponding to Figure 5 are given in Figure A1.

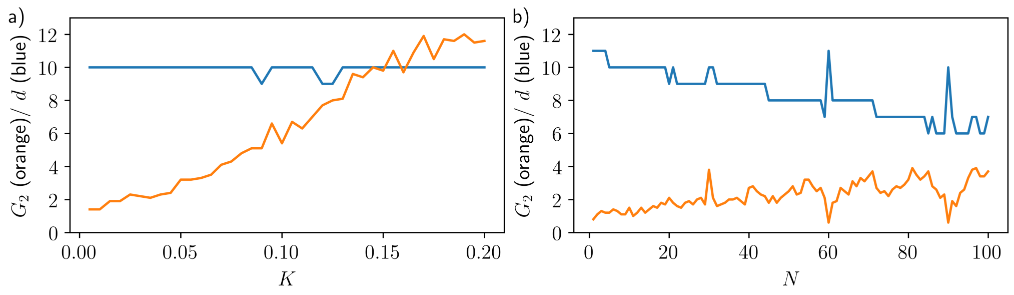

Figure A1. Values for the optimised input parameters G_2 (orange) and d (blue), corresponding to the Mackey–Glass $s = 10$ results depicted in Figure 5, as a function of (**a**) the virtual node coupling strength K and (**b**) the coupling delay N. All remaining parameters are as stated in Section 2.5.

Appendix B. Stuart-Landau Delay-Based Reservoir Computer

The Stuart–Landau system with time-delayed feedback is given by

$$\frac{dZ}{dt} = \left(\lambda_{SL} + J(t) + i\omega + \gamma|Z|^2\right)Z + Ke^{i\phi}Z(t-\tau). \tag{A1}$$

The parameter values are given in Table A1. The input sequence $J(t)$ for this system comprises piece-wise constant steps of length $\theta = T/N_v$, T is the clock time [43]. For this system, we used regularisation by noise, meaning that we added Gaussian white noise of strength R_{noise} to the state matrix **S** entries.

Table A1. Reservoir and input parameter values for the Stuart–Landau RC.

Parameter	Value	Parameter	Value
λ_{SL}	-0.02	ω	0
γ	-0.1	K	0.1
τ	105	ϕ	0
N_v	30	R_{noise}	1×10^{-7}
G_1	0.01	J_0	0
T	80		

References

1. Nakajima, K.; Fischer, I. *Reservoir Computing: Theory, Physical Implementations, and Applications*; Springer: New York, NY, USA, 2021.
2. Jaeger, H. *The 'Echo State' Approach to Analysing and Training Recurrent Neural Networks*; GMD Report 148; GMD—German National Research Institute for Computer Science: Darmstadt, Germany, 2001.
3. Dutoit, X.; Schrauwen, B.; Van Campenhout, J.; Stroobandt, D.; Van Brussel, H.; Nuttin, M. Pruning and regularization in reservoir computing. *Neurocomputing* **2009**, *72*, 1534–1546. [CrossRef]
4. Rodan, A.; Tiňo, P. Minimum Complexity Echo State Network. *IEEE Trans. Neural Netw.* **2011**, *22*, 131–144. [CrossRef] [PubMed]
5. Grigoryeva, L.; Henriques, J.; Larger, L.; Ortega, J.P. Stochastic nonlinear time series forecasting using time-delay reservoir computers: Performance and universality. *Neural Netw.* **2014**, *55*, 59. [CrossRef] [PubMed]
6. Nguimdo, R.M.; Verschaffelt, G.; Danckaert, J.; Van der Sande, G. Simultaneous Computation of Two Independent Tasks Using Reservoir Computing Based on a Single Photonic Nonlinear Node With Optical Feedback. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 3301–3307. [CrossRef] [PubMed]
7. Griffith, A.; Pomerance, A.; Gauthier, D.J. Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos* **2019**, *29*, 123108. [CrossRef] [PubMed]
8. Carroll, T.L. Path length statistics in reservoir computers. *Chaos* **2020**, *30*, 083130. [CrossRef]
9. Zheng, T.Y.; Yang, W.H.; Sun, J.; Xiong, X.Y.; Li, Z.T.; Zou, X.D. Parameters optimization method for the time-delayed reservoir computing with a nonlinear duffing mechanical oscillator. *Sci. Rep.* **2021**, *11*, 997. [CrossRef]
10. Ortín, S.; Pesquera, L. Reservoir Computing with an Ensemble of Time-Delay Reservoirs. *Cogn. Comput.* **2017**, *9*, 327–336. [CrossRef]

11. Röhm, A.; Lüdge, K. Multiplexed networks: Reservoir computing with virtual and real nodes. *J. Phys. Commun.* **2018**, *2*, 085007. [CrossRef]
12. Brunner, D. *Photonic Reservoir Computing, Optical Recurrent Neural Networks*; De Gruyter: Berlin, Germany, 2019.
13. Gauthier, D.J.; Bollt, E.M.; Griffith, A.; Barbosa, W.A.S. Next generation reservoir computing. *Nat. Commun.* **2021**, *12*, 5564. [CrossRef] [PubMed]
14. Vandoorne, K.; Dambre, J.; Verstraeten, D.; Schrauwen, B.; Bienstman, P. Parallel reservoir computing using optical amplifiers. *IEEE Trans. Neural Netw.* **2011**, *22*, 1469–1481. [CrossRef] [PubMed]
15. Duport, F.; Schneider, B.; Smerieri, A.; Haelterman, M.; Massar, S. All-optical reservoir computing. *Opt. Express* **2012**, *20*, 22783–22795. [CrossRef]
16. Tanaka, G.; Yamane, T.; Héroux, J.B.; Nakane, R.; Kanazawa, N.; Takeda, S.; Numata, H.; Nakano, D.; Hirose, A. Recent advances in physical reservoir computing: A review. *Neural Netw.* **2019**, *115*, 100–123. [CrossRef] [PubMed]
17. Canaday, D.; Griffith, A.; Gauthier, D.J. Rapid time series prediction with a hardware-based reservoir computer. *Chaos* **2018**, *28*, 123119. [CrossRef]
18. Harkhoe, K.; Verschaffelt, G.; Katumba, A.; Bienstman, P.; Van der Sande, G. Demonstrating delay-based reservoir computing using a compact photonic integrated chip. *Opt. Express* **2020**, *28*, 3086. [CrossRef]
19. Freiberger, M.; Sackesyn, S.; Ma, C.; Katumba, A.; Bienstman, P.; Dambre, J. Improving Time Series Recognition and Prediction With Networks and Ensembles of Passive Photonic Reservoirs. *IEEE J. Sel. Top. Quantum Electron.* **2020**, *26*, 7700611. [CrossRef]
20. Waibel, A.; Hanazawa, T.; Hinton, G.E.; Shikano, K.; Lang, K.J. Phoneme recognition using time-delay neural networks. *IEEE Trans. Signal Process.* **1989**, *37*, 328–339. [CrossRef]
21. Karamouz, M.; Razavi, S.; Araghinejad, S. Long-lead seasonal rainfall forecasting using time-delay recurrent neural networks: A case study. *Hydrol. Process.* **2008**, *22*, 229–241. [CrossRef]
22. Han, B.; Han, M. An Adaptive Algorithm of Universal Learning Network for Time Delay System. In Proceedings of the 2005 International Conference on Neural Networks and Brain, Beijing, China, 13–15 October 2005; Volume 3, pp. 1739–1744.[CrossRef]
23. Ranzini, S.M.; Da Ros, F.; Bülow, H.; Zibar, D. Tunable Optoelectronic Chromatic Dispersion Compensation Based on Machine Learning for Short-Reach Transmission. *Appl. Sci.* **2019**, *9*, 4332. [CrossRef]
24. Bardella, P.; Drzewietzki, L.; Krakowski, M.; Krestnikov, I.; Breuer, S. Mode locking in a tapered two-section quantum dot laser: Design and experiment. *Opt. Lett.* **2018**, *43*, 2827–2830. [CrossRef] [PubMed]
25. Takano, K.; Sugano, C.; Inubushi, M.; Yoshimura, K.; Sunada, S.; Kanno, K.; Uchida, A. Compact reservoir computing with a photonic integrated circuit. *Opt. Express* **2018**, *26*, 29424–29439. [CrossRef] [PubMed]
26. Appeltant, L.; Soriano, M.C.; Van der Sande, G.; Danckaert, J.; Massar, S.; Dambre, J.; Schrauwen, B.; Mirasso, C.R.; Fischer, I. Information processing using a single dynamical node as complex system. *Nat. Commun.* **2011**, *2*, 468. [CrossRef]
27. Paquot, Y.; Duport, F.; Smerieri, A.; Dambre, J.; Schrauwen, B.; Haelterman, M.; Massar, S. Optoelectronic Reservoir Computing. *Sci. Rep.* **2012**, *2*, 1–6. [CrossRef] [PubMed]
28. Brunner, D.; Penkovsky, B.; Marquez, B.A.; Jacquot, M.; Fischer, I.; Larger, L. Tutorial: Photonic neural networks in delay systems. *J. Appl. Phys.* **2018**, *124*, 152004. [CrossRef]
29. Brunner, D.; Soriano, M.C.; Mirasso, C.R.; Fischer, I. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nat. Commun.* **2013**, *4*, 1364. [CrossRef] [PubMed]
30. Wolters, J.; Buser, G.; Horsley, A.; Béguin, L.; Jöckel, A.; Jahn, J.P.; Warburton, R.J.; Treutlein, P. Simple Atomic Quantum Memory Suitable for Semiconductor Quantum Dot Single Photons. *Phys. Rev. Lett.* **2017**, *119*, 060502. [CrossRef]
31. Jiang, N.; Pu, Y.F.; Chang, W.; Li, C.; Zhang, S.; Duan, L.M. Experimental realization of 105-qubit random access quantum memory. *NPJ Quantum Inf.* **2019**, *5*, 28. [CrossRef]
32. Katz, O.; Firstenberg, O. Light storage for one second in room-temperature alkali vapor. *Nat. Commun.* **2018**, *9*, 2074. [CrossRef]
33. Arecchi, F.T.; Giacomelli, G.; Lapucci, A.; Meucci, R. Two-dimensional representation of a delayed dynamical system. *Phys. Rev. A* **1992**, *45*, R4225. [CrossRef] [PubMed]
34. Zajnulina, M.; Lingnau, B.; Lüdge, K. Four-wave Mixing in Quantum Dot Semiconductor Optical Amplifiers: A Detailed Analysis of the Nonlinear Effects. *IEEE J. Sel. Top. Quantum Electron.* **2017**, *23*, 3000112. [CrossRef]
35. Lingnau, B.; Lüdge, K. Quantum-Dot Semiconductor Optical Amplifiers. In *Handbook of Optoelectronic Device Modeling and Simulation*; Series in Optics and Optoelectronics; Piprek, J., Ed.; CRC Press: Boca Raton, FL, USA, 2017; Volume 1, Chapter 23. [CrossRef]
36. Mackey, M.C.; Glass, L. Oscillation and chaos in physiological control systems. *Science* **1977**, *197*, 287. [CrossRef]
37. Atiya, A.F.; Parlos, A.G. New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE Trans. Neural Netw.* **2000**, *11*, 697–709. [CrossRef]
38. Lorenz, E.N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **1963**, *20*, 130. [CrossRef]
39. Goldmann, M.; Mirasso, C.R.; Fischer, I.; Soriano, M.C. Exploiting transient dynamics of a time-multiplexed reservoir to boost the system performance. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–8. [CrossRef]
40. Ortín, S.; Soriano, M.C.; Pesquera, L.; Brunner, D.; San-Martín, D.; Fischer, I.; Mirasso, C.R.; Gutierrez, J.M. A Unified Framework for Reservoir Computing and Extreme Learning Machines based on a Single Time-delayed Neuron. *Sci. Rep.* **2015**, *5*, 14945. [CrossRef] [PubMed]

41. Köster, F.; Yanchuk, S.; Lüdge, K. Insight into delay based reservoir computing via eigenvalue analysis. *J. Phys. Photonics* **2021**, *3*, 024011. [CrossRef]
42. Köster, F.; Ehlert, D.; Lüdge, K. Limitations of the recall capabilities in delay based reservoir computing systems. *Cogn. Comput.* **2020**, *2020*, 1–8. [CrossRef]
43. Röhm, A.; Jaurigue, L.C.; Lüdge, K. Reservoir Computing Using Laser Networks. *IEEE J. Sel. Top. Quantum Electron.* **2019**, *26*, 7700108. [CrossRef]
44. Manneschi, L.; Ellis, M.O.A.; Gigante, G.; Lin, A.C.; Del Giudice, P.; Vasilaki, E. Exploiting Multiple Timescales in Hierarchical Echo State Networks. *Front. Appl. Math. Stat.* **2021**, *6*, 76. [CrossRef]
45. Stelzer, F.; Röhm, A.; Lüdge, K.; Yanchuk, S. Performance boost of time-delay reservoir computing by non-resonant clock cycle. *Neural Netw.* **2020**, *124*, 158–169. [CrossRef]
46. Nooteboom, P.D.; Feng, Q.Y.; López, C.; Hernández-García, E.; Dijkstra, H.A. Using network theory and machine learning to predict El Niño. *Earth Syst. Dyn.* **2018**, *9*, 969–983. [CrossRef]