Reservoir Computing Using Laser Networks

André Röhm, Lina Jaurigue, and Kathy Lüdge

(Invited Paper)

***Abstract*—Reservoir computing is a neuromorphic computing scheme inspired by the human brain. It has found great success as a versatile hardware-compatible application of machine learning concepts. In this paper, we highlight the fundamental working principles and important characteristics of reservoir computing with a particular focus on photonic systems and networks. These systems can further be enhanced by the inclusion of delayed variables to produce complex spatiotemporally mixed "time-multiplexed" networks. We use a simple nonlinear oscillator model, that is not only applicable to lasers, but can also describe a variety of other oscillating systems.**

***Index Terms*—Networks, reservoir computing, time delay, laser dynamics.**

I. Introduction

MACHINE learning is a quickly expanding field. It deals with concepts inspired by the human brain and features abstracted versions of neurons and synapses, which are trained to fulfil a specific task. Advancements are being made by ever more sophisticated computer programs running on ever larger server farms. However, brain-inspired concepts are not necessarily dependent on binary transistor-based logic, i.e. CPUs and GPUs. Reservoir computing is one particular set of machine learning rules that is compatible with non-standard hardware.

A basic reservoir computer works with a sufficiently high dimensional and driven dynamical system [1], [2]. Under the perturbation of the input, the system will react, i.e. its state will change. Different transformations can now be generated by a simple linear combination of accessible degrees of freedom of the reservoir, allowing for the construction of a wide array of functions, where finding the optimal combination is the task of the learning process. From the point of view of the existing artificial neural network framework, reservoir computers can be seen as a subclass of recurrent neural networks, where only the final output layer is trained (see Fig. 1a). The use of networks was originally envisioned [1]. However, because the reservoir is not touched upon, it can be replaced by any system that provides a nontrivial transformation of the input - essentially allowing the use of many dynamical systems as the reservoir (see Fig. 1b and c).

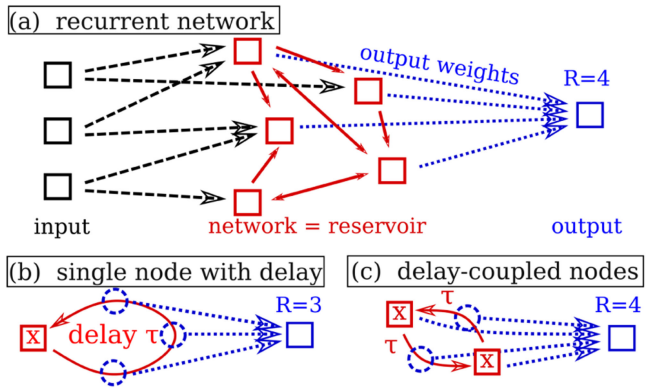

Fig. 1. Sketch of the basic components of the system discussed in this paper: (a) A recurrent artificial neural network, consisting of an input layer, a recurrent network and an output layer. If only the output connections (blue) are trained, this is one example of a reservoir computer system. (b) The delay-based reservoir approach consists of a single dynamical node with time-delayed feedback. (c) Delay-coupling of nodes allows to combine delay-based and network-based reservoir computing concepts into a time-multiplexed network. R denotes the read out dimension.

Reservoirs need to fulfil certain basic conditions: A reservoir should exhibit the same response if the same input is repeated, to allow for a reproducible set of base transformations. Slightly different inputs should still lead to mostly similar responses, so that the learning procedure can generalize. However, responses for sufficiently different inputs must be linearly separable in at least one dimension of the read-out degrees, to allow for a separation of characteristically different input signals. Furthermore, a reservoir should have the fading memory property, i.e. in the absence of any input it should return to a pre-defined fixed state. This corresponds to a physical system that operates around a fixed point attractor and is never pushed out of the attractor basin by the input. Finally, the reservoir should also contain nonlinearities, as otherwise the system will not be able to perform nonlinear transformations. This last condition can likely be refined with more quantitative measures of task and reservoir nonlinearity [3].

Contemporary reservoir computing prototypes typically posses a lot fewer nodes than state-of-the-art deep learning neural networks. This is partially based on the experimental nature, where read-out procedures often represent a bottle-neck, but is also due to the relative novelty of this subfield. While

Manuscript received April 7, 2019; revised June 21, 2019; accepted July 2, 2019. Date of publication July 9, 2019; date of current version August 1, 2019. This work was supported by the Deutsche Forschungsgemeinschaft within the framework of the collaborative research center SFB910. *(Corresponding author: André Röhm.)*

A. Röhm is with IFISC, Palma de Mallorca 07122, Spain (e-mail: andre@ifisc.uib-csic.es).

L. Jaurigue and K. Lüdge are with the Institut für Theoretische Physik, Technische Universität Berlin, Berlin 10623, Germany (e-mail: linajaurigue@gmail.com; kathy.luedge@tu-berlin.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTQE.2019.2927578

computer science has studied machine learning algorithms for decades, the photonics and theoretical physics communities are only now exploring the potential of future neuromorphic photonic computers. Additionally, as Ref. [4] argues, many of the current record breaking architectures in artificial neural networks also profit from the increased availability of powerful GPUs, allowing for the efficient simulation of large networks. Furthermore, by their nature reservoir computers will make less efficient use of every node than a fully trained recurrent network. At this point it is not clear, how powerful reservoir computing systems could become, if they were given similar amounts of resources.

This paper focuses on reservoir computing in laser networks. In particular, we will show important features of network and delay-based reservoirs. Section II gives an overview of the photonic reservoir computing literature. Section III provides the mathematical background. Section IV shows the importance of the timescales in delay-based reservoir computing using a Stuart-Landau oscillator model. Section V then introduces the theory behind time-multiplexed networks, with concluding remarks in Section VI.

II. Photonic Reservoir Computing

Reservoir computing was first introduced by Jaeger [1] and Maass *et al.* [2] at the beginning of the century and has since seen several additional conceptual extensions, especially the addition of delay-based systems [5]. While initially a purely theoretical concept, the power for experimental implementations was soon realized [6]. Photonic systems make excellent reservoirs as they are typically fast (add citation and numbers here). Photonic implementations of reservoir computing can also rely on a vast array of mature telecommunication technologies, allowing for rapid prototyping with off-the-shelves components. Reference [7] gives a good overview about photonic reservoir computing and shows the variety of experiments that have already been performed. Reference [4] gives an excellent introduction to the topic, with a focus on the connection to conventional artificial neural networks and different delay-based approaches for reservoirs.

There exist many different implementations of photonic reservoir computing, and both experiments and numerical simulations have investigated the capabilities of different reservoirs. In most cases, the tasks that are used are simple benchmark tasks, such as the Santa Fe chaotic timeseries prediction [8], the nonlinear auto-regressive moving average (NARMA) [9] (see App. A) and the TI spoken digit recognition tasks. Recently, practical applications are gaining attention, such as the reconstruction of distorted bit patterns [10]. For characterizing reservoirs, the linear memory capacity [11] (see App. B) is an important measure of any reservoir. Reference [3] outlines the possibility to extend this to nonlinear memory capacities. As of yet, no accepted standard for a holistic classification of reservoirs exists, so that results are not always comparable. Good performance on one tasks does not guarantee good results on a different benchmark task. Reservoir computing is also a powerful tool for the exploration of dynamical systems in general, as has been recently shown for the study of Lyapunov exponents [12], however, these results have yet to be reproduced for photonic systems.

There are different ways in which photonic systems can serve as the reservoir. The traditional approach relies on the brain-inspired concept of using networks of coupled optical elements. For example, semiconductor optical amplifiers can be coupled via on-chip wave-guides as has been demonstrated with numerical simulations [13], [14]. Later, this design was transformed to a network of passive waveguides and implemented experimentally [15]. Alternatively, an array of VCSELs can be coupled via a diffractive optical setup to create a network of lasers [16]. In this setup, a completely hardware based boolean reinforcement learning of the readout weights has also been realized [17]. Photonic network reservoirs generally allow for a fast computation, as the clock cycle of the computation is essentially bound to the time scale of the individual active element. However, these networks have the disadvantage of having many different interacting elements. This can lead to control problems. For example, achieving synchrony or other forms of ordered behaviour in large laser networks is difficult, with Ref. [18] showing the extensive work that is needed to control even networks of just 16 lasers.

Alternative approaches exist: Arguably the most important architecture for photonic reservoir computing is the delay-line based approach. A single nonlinear node is used in conjunction with a long delayed-feedback (see Fig. 1b). Delay-differential equations have an infinite dimensional phase space, which makes them excellent reservoirs. This was first demonstrated in an electrical setup [5], but quickly extended to electro-optical [19], [20] and fully optical reservoirs [6]. Delay-based reservoirs have the intrinsic advantage of only needing a single active component, which drastically reduces implementation complexity. Typically, a delay-based photonic reservoir relies only on a single laser source that provides the base light signal. Data injection is either obtained via direct modulation of the laser pump current, or via additional optical elements, such as Mach-Zehnder modulators. In some setups, the laser is entirely decoupled from the feedback loop and injection mechanism, acting only as a stable source of continuous wave light [20], [21]. The high-dimensional phase-space is then exploited with the help of a so-called masking procedure (see Section IV), which provokes the exploration of the phase space.

In general, delay-based reservoirs can be divided into two different categories: Optical and electro-optical reservoirs. This nomenclature denotes the nature of the delay. In an electro-optical setup, the delay-signal is electrical, typically a time-delayed version of the signal recorded by a photodiode. This signal is then fed back into the system, closing the loop. This can be either a bias modulation, e.g. that of a Mach-Zehnder modulator [21], or a direct current modulation of the laser, depending on which kind of nonlinearity is used. On the other hand, a mirror or a long optical fibre can be used to create a purely optical delay-based reservoir [6]. Here, the delayed signal is fed back into a system that is sensitive to optical feedback. However, care must be taken not to create chaotic or self-oscillating dynamics. In principle, optical and electro-optical delays could

also be combined, but a single delay already offers a robust reservoir. It is important to note, that there is a price to be paid for the simplicity of delay-based reservoirs: they can not be easily scaled up. In particular, due to the way that the system's used dimensionality scales with the length of the delay, any increase of the reservoir size typically results in an increase of the clock cycle T, i.e. a reduction of the computation speed.

One way to overcome this unfavourable scaling is to include multiple independent time-delayed reservoirs [22]. Alternatively, we have proposed a mixture of both the delay- and network based approach in Ref. [23]. These 'time-multiplexed networks' (see Fig. 1c) combine elements of both delay-based and network-based reservoirs. As highly complex laser networks are hard to control in large numbers, we expect the future large-scale reservoirs to be of this mixed type.

III. Reservoir Computer Theory

There are many ways to introduce the mathematical details of reservoir computing. A good introduction to the simplest forms of reservoirs, so called echo state networks, is given in Ref. [24]. Delay-based reservoirs in particular are well outlined in Ref. [25]. As our goal is explicitly to combine network and delay-based approaches, we will focus on a general but brief description of the main features, that is applicable to both.

First, it is important to realize, that reservoir computing is optimal for dynamical systems that evolve in time. Recall, that recurrent connections are an essential ingredient in a network-based reservoir. They are necessary to produce memory. The reservoir in general is assumed to be a dynamical system that can be described by a system state $\mathbf{X}(t)$ that evolves in time t according to:

$$\frac{d\mathbf{X}(t)}{dt} = F\left(\mathbf{X}(t), \mathbf{X}(t-\tau), \mathbf{J}(t)\right) \tag{1}$$

where F is the local evolution function that depends on the driving term $\mathbf{J}(t)$. We explicitly allow the evolution function F to depend on delayed states with time delay τ to account for delay-based reservoirs. This could easily be extended to multi-delay, state-dependent delay or distributed delays.

Because the reservoir is a dynamical system that evolves in time, reservoir computers are best suited for tasks with an inherent time dimension, such as timeseries prediction, nontrivial on-the-fly convolutions and audio or video classification. We want to make explicit use of the fact, that the reservoir does not reset to a neutral state inbetween input steps. The system state $\mathbf{X}(t)$ is kept transient at all times, and its position relative to its neutral resting state encodes information of the recent inputs.

The tasks possess a sampled input sequence $\{u\}$ with elements $u_k \in \mathbb{R}^U$ for index $k \in \mathbb{N}^+$, that represents the raw data to be processed. The reservoir computer is now supposed to perform a transformation G of this input sequence $\{u\}$ to produce the output sequence $\{o\}$ with elements $o_k \in \mathbb{R}^O$:

$$o_k = G(\{u^k\}) \tag{2}$$

where $\{u^k\}$ indicates the sequence of inputs up to element k, i.e. the nonlinear transformation G can in principle depend on any previous element, but not on future inputs. In general, the dimensionality of the input U and of the output O does not have to be the same. Nevertheless, many benchmark tasks restrict themselves to the case of both one-dimensional input and output $U = O = 1$. The input sequence has to be transformed into a time-continuous function $\mathbf{J}(t)$ to be fed into the dynamical system of Eq. (1). We want to treat all inputs u_k equally and thus feed them in the correct order at a constant rate, one at a time. Thus, for each time t there will be a k such that u_k is the newest input that has entered the reservoir. A typical choice for $\mathbf{J}(t)$ is to use a piece-wise constant function of heights u_k and lengths T. We will refer to this time T as the clock cycle, and it is one of the most important parameters of the reservoir computer. *A priori* it could be chosen as any value, however, practical limitations will result in a certain optimal clock cycle as will later be shown in Section IV.

The system state $\mathbf{X}(t)$ will start to evolve due to the forcing induced by the input $\mathbf{J}(t)$ according to Eq. (1). During each clock cycle T we will record the state of the reservoir. However, realistically and typically, not all degrees of freedom of the reservoir are accessible. Let $\mathbf{X}_k \in \mathbb{R}^R$ be the vector of reservoir degrees that is measured every clock cycle T. Here, R is the effective dimensionality of the read-out, and will be an important metric of the resulting reservoir performance (see Section V), if its individual entries are linearly independent. It is from this recorded state $\mathbf{X}_k$ that we will try to construct the output via a simple linear combination:

$$\hat{o}_k = \mathbf{X}_k \cdot W_{out} + c., \tag{3}$$

where $\hat{o}_k \in \mathbb{R}^O$ is the predicted output. The output weight matrix $W_{out} \in \mathbb{R}^{R\times O}$ and bias c are the only malleable parts of this approach and need to be found appropriately to produce optimal predictions. In the one-dimensional output case $O = 1$ the output weights W_{out} become a vector.

As reservoir computing is a *supervised* machine learning paradigm, the output weight matrix W_{out} and bias term c in Eq. (3) are found in a training phase for which the outputs $\{o\}$ are known, where the reservoir is driven with the corresponding inputs $\{u\}$. We will have to discard the reaction of the reservoir to the first K_{buffer} input elements, due to the system needing time to converge to its operating regime. Then, we will record the output degrees $\mathbf{X}_k$ under the next K_{training} elements, which form the state matrix S:

$$S = \begin{pmatrix} x_1^1 & \dots & x_1^R & 1 \\ \vdots & & \vdots & \vdots \\ x_{K_{\text{training}}}^1 & \dots & x_{K_{\text{training}}}^R & 1 \end{pmatrix} \tag{4}$$

where we have included 1 as a bias term and x_k^r is the rth element of the state vector $\mathbf{X}_k$. To quantify the error between prediction $\hat{o}_k$ and known target o_k we will define the normalized mean-squared error (NMSE):

$$NMSE(\hat{o}_k) = \frac{1}{K}\frac{\sum_{k=1}^{K}\left(o_k - \hat{o}_k\right)^2}{\sigma^2(o)} \tag{5}$$

which is minimized [1] by finding the best linear regression:

$$o \approx S\tilde{W}_{out}, \tag{6}$$

where the last element of $\tilde{W}_{out} \in \mathbb{R}^{R+1}$ is the bias weight c and o is the vector of the corresponding K_{training} known outputs. The linear regression of Eq. (6) corresponds to finding the best hyperplane through a cloud of data points in a high-dimensional space. It can be solved via matrix-inversion or with many numerical linear algebra libraries. We have used the armadillo wrapper for OpenBLAS in this paper [26].

The absolute freedom that is allowed with this paradigm is fascinating: We have not specified at all, what the nature of the reservoir should be, apart from a few very general properties. We have not put any restrictions on the evolution function F in Eq. (1) in particular. Thus, a very large class of systems is useable. Furthermore, we have also not specified how the data injection needs to occur, or how the read-out is supposed to happen. For the basic scheme it is only important that the reservoir be driven by some input $J(t)$ at a constant clock cycle T, and on the read-out we only require that some degrees of freedom can be recorded. This freedom makes the reservoir computing paradigm so powerful for use in analogue computing hardware. There is very little restriction on the architectures that are allowed. As opposed to spike-timing based computing in other neuromorphic research areas, the reservoir computer can work with any hardware, as long as said hardware allows for stable operation, has a sufficiently nonlinear response to input and has fading memory.

Delay-based reservoirs are a special case. Here, the dimensionality of the reservoir is not spatial, but temporal. Thus, to make full use of these reservoirs, the input and read-out is more complicated than the basic setup presented in previous paragraphs. In delay-based reservoirs, only a few degrees of freedoms are directly accessed, as for example only a single active element is neccessary. However, these degrees are measured *multiple times* per same clock cycle T. This is called time-multiplexing. Because the temporal dynamics of delay-systems are so complex, measuring the same node at multiple times per clock cycle T will effectively act like independent degrees of freedom. The effectiveness of delay-based reservoirs can further be increased by a more sophisticated pre-processing called masking [5] (see Section IV). The nature of the mask can play an important role [27], [28], but we will restrict ourselves to the case of equidistant binary masks in this paper for simplicity.

The most basic delay-based reservoirs rely on a single element, which is measured hundreds of times per clock cycle T to obtain a large effective read-out dimension R. When masking and multiple readouts per clock cycles are employed, one often refers to the degrees of freedom that are measured this way as 'virtual nodes', as they do not correspond to physically separate systems. The next section will highlight the complex interplay of the different timescales that can be found in delay-based reservoirs.

IV. Timescales of Delay-Based Reservoir Computers

An important feature of time-delay based reservoir computing systems is the interplay of the different time scales. First, we have the time delay τ. This is a variable that can typically be adjusted in experiments, for example by varying the lengths of the optical fibre or increasing the digitally controlled delay on the electronic feedback. Second, we have the clock cycle T that represents the speed of computation and is defined by the pre- and post-processing associated with data injection and state recording. And lastly we have the timescales of the physical system without feedback.

To illustrate the interplay of these timescales, we will first use a simple model and rely on a solitary Stuart-Landau oscillator with feedback. This model, which can be seen as a simple class-A laser model close to threshold, is the normal form of the Andronov-Hopf bifurcation. There are many more complex models to describe Laser dynamics, but we have chosen for now to focus on the simplest case, as it also allows for faster numerical simulations. The time it takes to simulate a large enough amount of data is one of our main bottlenecks. In the Stuart-Landau oscillator model, the complex variable Z evolves in time according to:

$$\frac{dZ}{dt} = \left(\lambda + gJ(t) + \mathrm{i}\omega + \gamma|Z|^2\right) Z + \kappa e^{\mathrm{i}\phi} Z(t-\tau) \qquad (7)$$

where $\lambda \in \mathbb{R}$ is the bifurcation parameter of the Stuart-Landau system and acts like a pump current, $\omega \in \mathbb{R}$ is the solitary frequency and $\gamma \in \mathbb{R}$ controls the nonlinearity. The feedback term $Z(t-\tau)$ is delayed by the delay time τ and is similar in nature to the usual feedback term in lasers, such as in the Lang-Kobayashi system [29], [30]. The feedback strength is κ and the feedback phase is ϕ. The input gain g scales how strong the system is driven by the time-varying input $J(t)$. We use the supercritical case and keep $\omega = 1, \gamma = -0.1, \kappa = 0.1$ and $\phi = 0$, which were determined to be suitable parameters by previous simulations [23]. We use the intensity $|Z|^2$ as the read-out variable emulating a photo-diode and will thus disregard the fast phase-dynamics of Z. We will only consider tasks with $O = U = 1$.

Consider Eq. (7): Let there now be N_V 'virtual nodes'. We will readout the system state N_V times per clock cycle T equidistantly, i.e. the effective reservoir dimension is $R = N_V$. Then we can denote the temporal separation of these virtual nodes as $\theta = T/N_V$, also called the time per virtual node. To increase the likelihood that these measurements are linearly independent, we use a masking that is illustrated in Fig. 2. Sophisticated masking procedures have been proposed [27], which can increase the performance. The focus here will be on general trends and so we restrict ourselves to the simplest case of a binary mask. The masked input $J(t)$ (Fig. 2c) is created from the raw input data u_k (Fig. 2a) via multiplication of a T-periodic binary mask (Fig. 2b). The readout is performed at the end of each θ segment. The system response then is dynamically rich (Fig. 2d). The first restriction on the choice of timescales will now be the following: To make effective use of the time-multiplexing, θ should be on the timescale of the typical local system dynamics. If θ is orders of magnitudes smaller than the local dynamics, successive measurements of the system state will not be sufficiently different, and thus the virtual nodes will not be linearly independent. Conversely, if θ is too large, a lot of potential information will have been lost by the too sparse measurement of the temporal dynamics. Restricting the time per virtual node θ to the order of the local system time scales

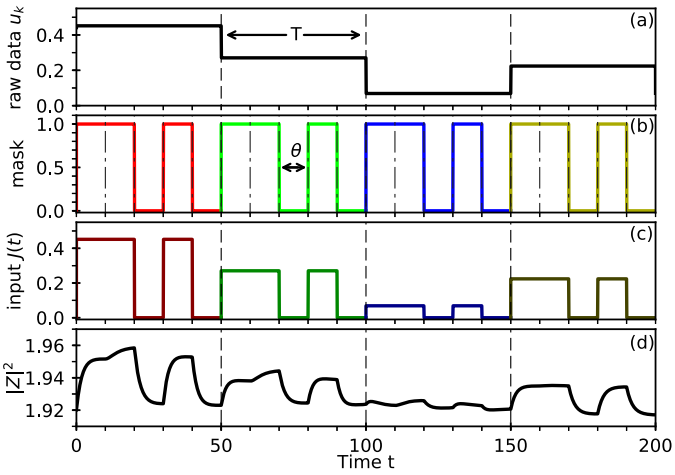

Fig. 2. The timescales and masking procedure for a reservoir computer based on single node with delayed feedback as in Eq. (7). The top row shows the raw input data u_k, which is fed in as a piece-wise constant function. Every clock cycle T a new input is taken. The second row shows the binary mask, which is periodic with T and consists of N_V segments of length θ. The mask and raw input data are multiplied to obtain the masked input $J(t)$ (third panel). This is fed into the system and creates the response as given by the bottom panel. Parameters: $\lambda = 0.2$, $\omega = 1$, $g = 0.01$, $\gamma = -0.1$, $\kappa = 0.1$, $\phi = 0$, $T = 50$, $\tau = 60$, $N_V = R = 5$, $T = 50$, $\theta = 10$.

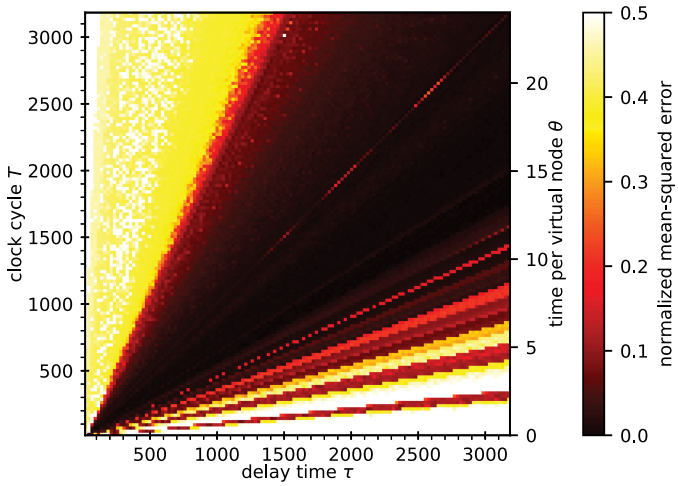

Fig. 3. Reservoir computing benchmark results (NARMA10) for the 'virtual network' approach of delay-based reservoir computing using the Stuart-Landau oscillator model of Eq. (7) as a function of clock cycle T and delay time τ. The colour code indicates the NARMA10 error as measured by NMSE of Eq. (5) in the testing phase. Parameters: $\lambda = -0.02$, $\omega = 0$, $g = 0.01$, $\gamma = -0.1$, $\kappa = 0.1$, $\phi = 0$, $N_V = 128$, $K_{\text{buffer}} = 20000$, $K_{\text{training}} = 20000$, $K_{\text{testing}} = 5000$.

then directly translates into a clock cycle T for any given N_V by $T = N_V\theta$. In the case of the Stuart-Landau oscillator with parameters as given, this means θ should be on the order of 10. This is determined by both γ and λ as can be calculated by evaluating the right hand side of Eq. (7) for typical values of Z.

It is now important to note, that the delay-time τ can be chosen completely independently of both θ and T. It is not clear *a priori* what an optimal value would be. At no point in the introduction of time-multiplexing do we require any particular τ. The only role of the delay is to introduce long-term correlations of the dynamics. Historically, the resonant case of $T = \tau$ was often used [5], [6], [20], although experiments likely only approximated that case. We will now explore the whole range of τ and T ratios.

Figure 3 shows the importance of the system time scales for a single Stuart-Landau oscillator according to Eq. (7) with delayed feedback on the reservoir computing benchmark NARMA10 task (see App. A). A binary (0,1) masking procedure was used to generate a virtual network. Figure 3 is a two-parameter plot varying both the delay time τ as well as the clock cycle T. The resulting performance is measured by the normalized mean-squared error of Eq. (5) as indicated by the colour code. All other parameters are kept constant and are given in the caption.

As can be seen in Fig. 3, the system performs better (dark) in some regions than others. First, extremely low values of either delay time τ or clock cycle T are detrimental to the reservoir computing. This reduction in the performance is likely caused by either a drastically too short information retention span (short delay) or a reduction of the effective reservoir dimension for too low θ, if successive measurements of the system state are not linearly independent. This underlines that the right choice of τ and θ is important.

More interestingly, one can also see a clear resonance effect between τ and T. In particular, for integer values p and q that roughly fulfil:

$$p\tau \approx qT, \tag{8}$$

the error is clearly increased. In Fig. 3 the line for $\tau = T$ is the diagonal and visibly stands out in some places, but other smaller resonances can also be seen. Additional simulations have shown that these resonances become more prominent for smaller virtual node numbers N_V and with regularization. Even apart from the pure resonances, the whole structure of Fig. 3 is clearly organized by lines given by $T = c\tau$ with different c. These numerical finding indicates that the historical choice of $T = \tau$ could be actively detrimental.

Figure 4 shows additional information. Here, the total linear memory capacity (see App. B), a measure of the systems ability to recall past inputs, is shown as a function of τ and T. The linear memory of the system is also clearly organized around lines given by Eq. (8). Our investigations show that along the resonances, a non-trivial redistribution of memory occurs. This often leads to a loss of short-term memory, but an increase in the memory for the extremely long past ($m > 100$). One cause for the increase of the NARMA10 error seen in Fig. 3 therefore lies in a loss of linear memory along the resonances. Further investigation of these resonance features and experimental verification thereof will be necessary in the future. It is important to note, that the results of this section also apply to the time-multiplexing of networks in the next section. The same interplay of timescales can be observed there.

V. Time-Multiplexed Networks

We will now turn towards networks consisting of multiple different oscillators. As opposed to the 'virtual network' scenario,

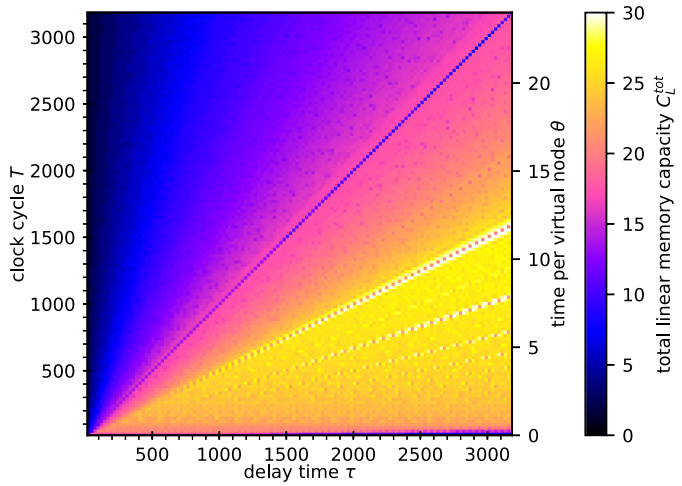

Fig. 4. Total linear memory capacity C_L^{tot} for the 'virtual network' approach of delay-based reservoir computing using the Stuart-Landau oscillator model as a function of clock cycle T and delay time τ. The colour code indicates the total memory capacity as measured by NMSE in the Testing run. Parameters: $\lambda = -0.02$, $\omega = 0$, $g = 0.01$, $\gamma = -0.1$, $\kappa = 0.1$, $\phi = 0$, $N_V = 128$, $K_{\text{buffer}} = 20000$, $K_{\text{training}} = 20000$, $K_{\text{testing}} = 5000$.

these different oscillators represent physically separate systems, and not temporally separated measurements. We will keep using the same local dynamics and thus arrive at a network of coupled Stuart-Landau oscillators:

$$\dot{Z}_l = (\lambda + gJ_l(t) + \mathrm{i}\omega + \gamma\,|Z_l|^2)Z_l + \kappa e^{\mathrm{i}\phi}\sum_{l=1}^{N_R} A_{lm}Z_m(t-\tau) \tag{9}$$

where A_{lm} is the entry of the coupling matrix A describing the coupling from node m to l and N_R is the number of real physical nodes (as opposed to the number of virtual nodes N_V). We have chosen to use a delay-coupled network here, because this is the realistic case for photonic systems, as transmission delays can typically not be ignored. It is now relatively straightforward to use such systems directly for reservoir computing, where each node is recorded individually. It is important to apply a different preprocessing to the inputs $J_l(t)$ of the different nodes l. This will promote a complex trajectory in the phase space of the system. If an identical input was chosen for all nodes, this could restrict the evolution of the system state to the in-phase synchronized manifold and thus drastically reduce the effective dimensionality.

Recall that we constructed the virtual networks of Eq. (7) by recording the intensity of the sole oscillator $|Z|^2$ multiple times per clock cycle T to create additional 'virtual nodes'. This was possible, because the nature of the delayed feedback dynamics meant that there were rich temporal dynamics that we would otherwise have left underutilized. There is now nothing conceptually different in the network case of Eq. (9) that prevents us from doing the same here. If the coupling between the different nodes is long, then they will have rich temporal responses to being driven. Hence it is easy to apply time-multiplexing to networks as well [23].

The simplest implementation of time-multiplexing for Eq. (9) is to record the intensity $|Z_l|^2$ of every oscillator l every θ. This will create $N_V = T/\theta$ virtual nodes for every real node in the system. We will thus refer to N_V as the virtualisation factor in this case. Time-multiplexing in the network case will lead to a multiplicative scaling of the total effective read-out dimension $R = N_V N_R$. It can be an efficient way of increasing the dimensionality of a network that already possesses delayed coupling. Ref. [23] investigated these networks in more details, where we have called them time-multiplexed networks. We shall focus on a general aspect of reservoirs here, which is particularly easy to illustrate with time-multiplexed networks: The effect of the total read-out dimension.

A. *Importance of Total Read-Out Dimension*

One can conjecture that the limiting factor of the total computational power of a given reservoir computer is linked to the total read-out dimension R. This is strongly implied by the results of Dambre *et al.* [3]. There are no small perfect reservoirs. And a perfect reservoir will act like an imperfect one, if only very few degrees of freedom are tapped. In particular, adding additional nodes to a network-based reservoir should not be expected to increase the performance, if these nodes are not recorded, because the read-out dimension R stays constant. Contrast this with artificial neural network systems, where all internal links are trained. The training process and the structure of the 'hidden layers' is crucial and enlarging the hidden layers does usually increase the computational power.

This is not to say, that the internal dynamics of the reservoir, even of the parts that are not measured, are irrelevant - quite the contrary. But due to the learning process being restricted to be purely linear on the output weights for reservoir computers, the error on any task can only be minimized if sufficiently many degrees of freedom of the reservoir are recorded. Therefore accessing additional read-out degrees, and thus increasing the read-out dimension R, is typically more important than increasing the phase-space dimension of the reservoir. Naturally, any new degree of freedom is only useful, if it has a new orthogonal component to the already present readouts.

Time-multiplexed networks now offer an easy way to study the two main ways of increasing the read-out dimension: First, by increasing the number of real nodes N_R, or, second, by increasing the virtualisation factor N_V. Figure 5a shows the results for the NARMA task in colour code for time-multiplexed networks of different sizes. Both the number of real nodes N_R and the virtualisation factor N_V have been varied independently.

In Figure 5a there is a clear correlation between total read-out dimension $R = N_V N_R$ and the resulting reservoir computing performance, with larger read-out degrees generally performing better (top right corner). For very small reservoirs (left and bottom borders of Fig. 5), the benchmark NARMA task has high errors (Fig. 5a) and the reservoir computer has overall low memory capacities (Fig. 5b). In Fig. 5 we have kept constant the time per virtual node at $\theta = 12$ and also the ratio of delay and clock time at $12\tau = 17.7T$. Thus, the delay time τ is scaling linearly with the virtualisation factor N_V.

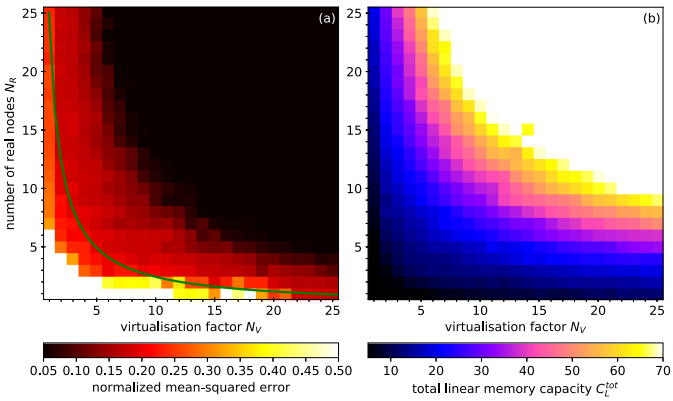

Fig. 5. NARMA task normalized mean-squared error (a) and linear memory capacity (b) for time multiplexed networks. The Stuart-Landau oscillator model of Eq. (9) was used with different network sizes N_R and virtualisation factors N_V, which influence the total readout dimension R given by $R = N_V N_R$. The green line in (a) shows constant $R = 25$. Note the different scaling on the colour bars to highlight features. The delay and clock cycle scale with N_V. Parameters: $\lambda = -0.02, \omega = 0, g = 0.01, \gamma = -0.1, \kappa = 0.1, \phi = 1.0, T = 12\ N_V, \tau = 17.7\ N_V, K_{\text{buffer}} = 20000, K_{\text{training}} = 20000, K_{\text{testing}} = 5000$.

B. Advantages of Time-Multiplexed Networks

It is important to note, that the combination of time-multiplexing and networks offers multiple advantages: First, as opposed to the purely delay-based approach, the system size can be increased without necessarily increasing the clock cycle T, which would reduce the computation speed. Furthermore, the topology of the network of real nodes can be customized to optimize the performance [23]. Moreover, due to the multiplicative scaling of the read-out dimension $R = N_V N_R$, large reservoir computers can be also constructed with relatively few elements, keeping them in the realm of current technological limits. Previous simulations [23] have also indicated that a mixture of both real and virtual nodes can offer an advantage over reservoirs using just one of the approaches, even if reservoirs of identical sizes are compared.

VI. Conclusion

We have given a brief review of the state of the art in photonic reservoir computing. Reservoir computing is a promising approach to hardware based machine-learning and can be implemented in a multitude of substrates and ways. Photonic systems are especially suited as reservoirs, as their dynamics has been extensively explored in the past and can be combined with mature telecommunication hardware for rapid prototyping.

The two most promising ways for photonic reservoirs are the delay-based approach where a 'virtual networks' is created via time-multiplexing and the coupling of multiple optical elements to create a classical network. However, both approaches suffer from scaling problems in different ways: The time-multiplexed systems generally become slower if additional virtual nodes are included, as the size of the phase space scales in time. Additionally, special care must be taken when choosing the defining system time scales. Conversely, coupling a large number of optical elements in a controlled way is difficult when using the classical network approach.

We have demonstrated a simple way of how these two approaches can be combined to produce realistic yet powerful network-based reservoir computers. We use a delay-coupled network and apply a time-multiplexing procedure to generate a time-multiplexed network. These networks then show a multiplicative scaling of the read-out dimension $R = N_V N_R$ with the virtualisation factor N_V and the number of nodes N_R. This measure R is an important predictor of reservoir performance.

Appendix A
NARMA Task

The Normalized Auto-Regressive Moving Average (NARMA) task is a benchmark reservoir computing task. It combines nonlinearity and memory [9]. However, it is arguable somewhat artificial and is used purely for historical reasons. It is defined for different input lengths. We use the length 10. The input for the NARMA10 is a series of uniformly distributed u_k in the interval [0,0.5]. The output o_k is defined by an iterative formula as given by:

$$o_{k+1} = 0.3o_k + 0.05o_k \left(\sum_{i=0}^{9} o_{k-i} \right) + 1.5u_{k-9}u_k + 0.1 \tag{10}$$

where it is assumed for the initialization, that all $u_{k<0}$, $a_{k<0} = 0$. Unfortunately, the NARMA10 task of Eq. (10) is not well defined, and in rare cases if a stretch of random numbers u_k happen to all be rather large, the sequence will diverge. We therefore cap $o_k \leq 1$, which prevents it from diverging.

Appendix B
Memory Capacity

The memory capacity was first quantified for echo state networks in Ref. [11]. The input for the memory capacity task u_k is a random number. The target is

$$o_k^m = u_{k-m}, \tag{11}$$

where o_k^m is called the m-step *linear recall*. The corresponding m-step memory capacity C_L^m is then given by [11]:

$$C_L^m = 1 - NMSE(\hat{o}_k^m) \tag{12}$$

with the the normalized mean squared error (NMSE) as given by Eq. (5). The m-step memory capacity C_L^m is 0 if the system is unable to recall any information m steps in the past, and correspondingly $C^m = 1$ for a perfect linear recall. Here, the memory was calculated simultaneously with the NARMA10 task and thus for an input sequence drawn from a uniform distribution between 0 and 0.5. The total linear memory capacity of a system $C_{L,tot}$ is given by the sum of all C_L^m:

$$C_{L,tot} = \sum_{m=0}^{\infty} C^m \tag{13}$$

where we have summed the first 400 elements. We have set all $C^m < 0.05$ to 0 to exclude errors caused by the finite testing length.

REFERENCES

[1] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks," German National Research Institute for Computer Science, GMD Rep. 148, 2001. [Online]. Available: https://www.researchgate.net/publication/215385037

[2] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comp.*, vol. 14, pp. 2531–2560, 2002.

[3] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, "Information processing capacity of dynamical systems," *Sci. Rep.*, vol. 2, 2012, Art. no. 514.

[4] D. Brunner *et al.*, "Tutorial: Photonic neural networks in delay systems," *J. Appl. Phys.*, vol. 124, no. 15, 2018, Art. no. 152004.

[5] L. Appeltant *et al.*, "Information processing using a single dynamical node as complex system," *Nature Commun.*, vol. 2, 2011, Art. no. 468.

[6] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature Commun.*, vol. 4, 2013, Art. no. 1364.

[7] G. Van der Sande, D. Brunner, and M. C. Soriano, "Advances in photonic reservoir computing," *Nanophotonics*, vol. 6, no. 3, 2017, Art. no. 561.

[8] A. S. Weigend and N. A. Gershenfeld, "Results of the time series prediction competition at the Santa Fe Institute," in *Proc. IEEE Int. Conf. Neural Netw.*, Mar./Apr. 1993, pp. 1786–1793.

[9] A. F. Atiya and A. G. Parlos, "New results on recurrent network training: Unifying the algorithms and accelerating convergence," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 697–709, May 2000.

[10] A. Argyris, J. Bueno, and I. Fischer, "Photonic machine learning implementation for signal recovery in optical communications," *Sci. Rep.*, vol. 8, no. 8487, pp. 1–13, 2018.

[11] H. Jaeger, "Short term memory in echo state networks," German National Research Institute for Computer Science, GMD Rep. 152, 2002. [Online]. Available: https://www.researchgate.net/publication/247514367

[12] J. Pathak, Z. Lu, B. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data," *Chaos*, vol. 27, no. 12, 2017, Art. no. 121102.

[13] K. Vandoorne *et al.*, "Toward optical signal processing using photonic reservoir computing," *Opt. Express*, vol. 16, no. 15, 2008, Art. no. 11182.

[14] K. Vandoorne, J. Dambre, D. Verstraeten, B. Schrauwen, and P. Bienstman, "Parallel reservoir computing using optical amplifiers," *IEEE Trans. Neural Netw.*, vol. 22, no. 9, pp. 1469–1481, Sep. 2011.

[15] K. Vandoorne *et al.*, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature Commun.*, vol. 5, 2014, Art. no. 3541.

[16] D. Brunner and I. Fischer, "Reconfigurable semiconductor laser networks based on diffractive coupling," *Opt. Lett.*, vol. 40, no. 16, pp. 3854–3857, 2015.

[17] J. Bueno *et al.*, "Reinforcement learning in a large-scale photonic recurrent neural network," *Optica*, vol. 5, no. 6, pp. 756–760, 2018.

[18] A. Argyris, M. Bourmpos, and D. Syvridis, "Experimental synchrony of semiconductor lasers in coupled networks," *Opt. Express*, vol. 24, no. 5, pp. 5600–5614, 2016.

[19] Y. Paquot *et al.*, "Optoelectronic reservoir computing," *Sci. Rep.*, vol. 2, 2012, Art. no. 287.

[20] L. Larger *et al.*, "Photonic information processing beyond turing: An optoelectronic implementation of reservoir computing," *Opt. Express*, vol. 20, no. 3, pp. 3241–3249, 2012.

[21] L. Larger *et al.*, "High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification," *Phys. Rev. X*, vol. 7, 2017, Art. no. 011015.

[22] S. Ortin and L. Pesquera, "Reservoir computing with an ensemble of time-delay reservoirs," *Cogn. Comput.*, vol. 9, no. 3, pp. 327–336, 2017.

[23] A. Röhm and K. Lüdge, "Multiplexed networks: reservoir computing with virtual and real nodes," *J. Phys. Commun.*, vol. 2, 2018, Art. no. 085007.

[24] A. Rodan and P. Tiňo, "Simple deterministically constructed cycle reservoirs with regular jumps," *Neural Comput.*, vol. 24, no. 7, pp. 1822–1852, 2012.

[25] M. Hermans, M. C. Soriano, J. Dambre, P. Bienstman, and I. Fischer, "Photonic delay systems as machine learning implementations," *J. Mach. Learn. Res.*, vol. 16, 2015, Art. no. 2081.

[26] C. Sanderson and R. Curtin, "Armadillo: A template-based C library for linear algebra," *J. Open Source Softw.*, vol. 1, p. 26, 2016.

[27] J. Nakayama, K. Kanno, and A. Uchida, "Laser dynamical reservoir computing with consistency: An approach of a chaos mask signal," *Opt. Express*, vol. 24, no. 8, pp. 8679–8692, 2016.

[28] Y. Kuriki, J. Nakayama, K. Takano, and A. Uchida, "Impact of input mask signals on delay-based photonic reservoir computing with semiconductor lasers," *Opt. Express*, vol. 26, no. 5, pp. 5777–5788, 2018.

[29] R. Lang and K. Kobayashi, "External optical feedback effects on semiconductor injection laser properties," *IEEE J. Quantum Electron.*, vol. QE-16, no. 3, pp. 347–355, Mar. 1980.

[30] P. M. Alsing, V. Kovanis, A. Gavrielides, and T. Erneux, "Lang and Kobayashi phase equation," *Phys. Rev. A*, vol. 53, no. 6, pp. 4429–4434, 1996.

André Röhm received the doctoral degree in physics from Technische Universität Berlin, Berlin, Germany, in 2018. He has worked on the modeling of quantum-dot based laser devices, laser networks, reservoir computing, and symmetry-breaking in coupled oscillators.

Lina Jaurigue received the doctoral degree in physics from Technische Universität Berlin, Berlin, Germany, in 2016. She has worked on the modeling of stochastic properties in mode-locked laser devices, on bifurcation analysis in delay systems and on reservoir computing with optical networks.

Kathy Lüdge was born in Berlin, Germany, in 1976. She received the Diploma and the Dr. rer. nat degrees in physics and the Habiltation (Venia Lengendi) degree from Technische University Berlin (TU Berlin), Berlin, Germany, in 2000, 2003, and 2011, respectively. Since 2016, she has been a Professor of theoretical physics with TU Berlin. She held a one-year Humboldt Fellowship with the Math Department, University of Auckland, Auckland, New Zealand, in 2017. In 2015, she was a Visiting Professor with the Freie Universität Berlin, and from 2011 to 2014, she was a Privatdozentin with the Institute of Theoretical Physics, TU Berlin. From 2001–2002, she was a Visiting Scholar with the Department of Material Science, University of Minnesota, Minneapolis, MN, USA. She is an Editor of the book entitled *Nonlinear Laser Dynamics-From Quantum Dots to Cryptography* (Wiley, 2012). Her research interests include modeling of semiconductor quantum-dot lasers, nonlinear laser dynamics, and reservoir computing with optical networks and delay.