Chaos
An Interdisciplinary Journal of Nonlinear Science

RESEARCH ARTICLE | MAY 16 2025

Enhancing reservoir predictions of chaotic time series by incorporating delayed values of input and reservoir variables

Special Collection: Nonlinear Dynamics of Reservoir Computing: Theory, Realization and Application

Luk Fleddermann ; Sebastian Herzog ; Ulrich Parlitz

Chaos 35, 053147 (2025)
https://doi.org/10.1063/5.0258250

Enhancing reservoir predictions of chaotic time series by incorporating delayed values of input and reservoir variables

Cite as: Chaos 35, 053147 (2025); doi: 10.1063/5.0258250
Submitted: 15 January 2025 · Accepted: 25 April 2025 ·
Published Online: 16 May 2025

Luk Fleddermann,[1,2] Sebastian Herzog,[3] and Ulrich Parlitz[1,2,a)]

AFFILIATIONS

[1]Biomedical Physics Group, Max Planck Institute for Dynamics and Self-Organization, Am Faßberg 17, 37077 Göttingen, Germany
[2]Institute for the Dynamics of Complex Systems, University of Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany
[3]Third Institute of Physics and Bernstein Center for Computational Neuroscience, University of Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany

Note: This paper is part of the Special Topic on Nonlinear Dynamics of Reservoir Computing: Theory, Application and Realization.
[a)]**Author to whom correspondence should be addressed:** ulrich.parlitz@ds.mpg.de

ABSTRACT

Time series generated by chaotic dynamical systems can be effectively predicted using readouts from driven reservoir dynamics. In practical scenarios, however, only time series measurements with partial knowledge of the chaotic system's state are usually available. To address this aspect, we evaluate and compare the performance of reservoir computing in predicting time series under both conditions of complete and partial knowledge of the state. Our results show that memory improves the prediction accuracy only when the system state is partially known. For cases with partial state knowledge, we extend the mean prediction horizon by including delayed values of both the input and reservoir variables. To ensure the robustness of this result, we test it in systems with varying degrees of complexity. Finally, we show that the inclusion of delayed values can also facilitate the optimization of hyperparameters for predictions based on full knowledge of the system state.

Reservoir computing is a rapidly growing machine learning approach that is rooted in dynamical system theory and offers fast training and low computational costs. In this study, a recent modification of the classical reservoir computing method is investigated, which reuses time-delayed values to improve the prediction capabilities by effectively utilizing delay embedding. Our research not only shows that delayed values can improve time series predictions in the presence of partial system knowledge but also confirms its ability to facilitate the often complex task of hyperparameter optimization. These results broaden the knowledge about time-delayed values in reservoir computing, demonstrating their applicability and benefits for tackling challenging time series prediction problems.

I. INTRODUCTION

Reservoir computing[1–4] is a machine learning method that uses readouts of the dynamics of a (high dimensional) dynamical system—the reservoir—to perform predictions. The reservoir itself is driven by an input signal that contains the information that enables predictions. The key characteristic of reservoir computing is that only a linear readout of the reservoir's activation is trained,[5] usually by means of linear regression. Therefore, the training process is computationally efficient[6] and provides an optimal solution, offering significant advantages over other machine learning approaches. In addition, it performs well across various tasks that are typically considered challenging. In particular, good performance is achieved on tasks with strong sequential components such as modeling of complex temporal dependencies and time series predictions.[6–10] However, a more overarching task is the optimization of the reservoir itself, which is typically achieved by tuning the hyperparameters. Hyperparameters are generally predefined parameters that govern the behavior and performance of a machine learning method. In the case of reservoir computing, these parameters define the inner structure and, hence, the dynamical features of the reservoir. While determining the optimal readout is straightforward,

optimizing the hyperparameters presents a challenge in itself. This task significantly increases computational effort or requires prior knowledge of hyperparameters that are well-suited for the task at hand.

When trying to simplify this optimization task, many articles are concerned with building theoretical foundations to find parameters that work well[11–14] and are trying to ease the task of hyperparameter optimization by augmenting the system[15] or by finding computationally cheap hyperparameter optimization schemes.[16–18] On the contrary, articles propose new methods of augmenting the common reservoir computing approach.[19–21] In doing so, they introduce additional hyperparameters. When new methods are proposed, their performance is typically evaluated using a subset of the pre-existing hyperparameters. As a result, it often remains unclear whether the new method truly outperforms the existing structural modifications or if its superiority is limited to the specific subset of reservoir structures and hyperparameters selected for testing.

One of these newly introduced methods of augmenting the classical reservoir computing approach is the use of delayed values.[15,20,22–26] This method saves time series of input or reservoir states and reuses them in a later prediction step, i.e., it reuses the time-delayed values. In recent studies, the use of delayed values has shown to reduce the dependence on hyperparameters,[15] enable a reduction of the needed reservoir size,[20,23] or minimize the error of linear readouts.[22,26] We aim to further investigate these findings, focusing on whether the assumed improvements fall within the range of the delayed values, or if they result from the interaction of various factors, such as hyperparameters.

The use of delayed values has been predominantly evaluated on one-step prediction tasks.[15,20,22,26] Here, the reservoir is driven by a given input time series and the error is measured between the readout of the reservoir system and the true evaluation data set. However, the prediction of time series and the replication of chaotic attractors itself[27] (i.e., the reproduction of the climate) can be performed by iteratively feeding the reservoir's output back as a new input (closed-loop). Thereby, the reservoir is turned into an autonomous system, where the one-step-ahead prediction is trained to match the behavior of the training data. The influences of the delayed reservoir states on iterative predictions is so far only investigated by evaluating the prediction errors after a limited number of iterative applications.[23] Whether the feedback of the delayed values is beneficial to precisely predict a time series for as long as possible by iterative applications has not been tested to the best of our knowledge.

This iterative approach opens up the possibility to not only predict the trajectories of a dynamical system, but iteratively predict the temporal evolution of only a part of the variables of the dynamical system at hand. In the first case, the whole system state needs to be known, such that time series of it can be used in training and eventually be predicted. However, the second case is the most common real-world application of time series predictions, for instance, for predicting the future evolution of weather, climate, or ecological-systems.[28–32] Here, only partial knowledge of the underlying dynamical system is available due to incomplete measurement capabilities or even incomplete knowledge of the full dynamical system state. Even though, a single time step does not uniquely determine the state of the dynamical system, theoretically, such time series can be predicted with proper knowledge of the past.[33,34] To the best of our knowledge, the benefits of delayed values for the iterative predictions of time series with only partial state knowledge available are previously not explored.

To contribute to the research on delayed values, in this article, we test the extension of delayed values on the task of iterative time series prediction. We evaluate its performance using the valid time as a short term prediction measure and compare between predictions with full or partial state knowledge available. By using and optimizing a large number of (classical) hyperparameters, we show that memory of the reservoir improves performances if only partial state knowledge is available. For these memory requiring tasks, we demonstrate that reservoir computing particularly benefits from introducing delayed values and show the superiority of delayed values reservoir computing over the optimized classical approach. In addition, we demonstrate that delayed values enhance time series predictions with full state knowledge if classical hyperparameters are chosen non-optimal. To this end, in Sec. II A, we introduce the method of reservoir computing using echo state networks in detail. We introduce the non-linear task of iterative time series prediction and the chosen measure of the quality of a prediction in Sec. II B, discuss the method of delayed values in Sec. II D, and highlight how we compared the performance in Sec. II E. The results of this research are presented in Sec. III. In Sec. III A and Sec. III B, we outline the differences for classical reservoir computing with and without memory, and delayed values reservoir computing in time series predictions with either full or partial state knowledge. Subsequently, we present the benefits of delayed values in reservoir computing with non-optimal hyperparameter sets in Sec. III C. Finally, we contextualize our results in previous research by discussing it and presenting an outlook.

II. METHODS

A. Reservoir/Echo state networks

An Echo State Network (ESN) that follows the design of Jaeger[1] is used as a reservoir. An ESN is a randomly drawn recurred neural network, which is specified only by the choice of a few structural hyperparameters. The ESN consists of a reservoir of N inner nodes $\vec{s}_m$, which are driven by an input time series $\vec{u}_m = \vec{u}(t_m)$ at discrete time instances $t_m = t_0 + m\Delta t$, with $m \in \mathbb{N}$. The weights of the recurrent connections of the network are summarized in the adjacency matrix $\boldsymbol{W}$. Similarly, the input matrix $\boldsymbol{W}_{\text{in}}$ describes the coupling of the input $\vec{z}_m$ to the reservoir nodes, where $\vec{z}_m = [b_{\text{in}}, \vec{u}_m]$ is a concatenation of a constant input bias $b_{\text{in}} = 1$ and the driving system state $\vec{u}_m$. For both matrices $\boldsymbol{W}$ and $\boldsymbol{W}_{\text{in}}$, the entries are independently drawn from a uniform distribution of values in $[-1, 1]$. The adjacency matrix $\boldsymbol{W}$ is turned into a sparse matrix by setting random entries to zero with the exception of a small fraction ε. Furthermore, the adjacency matrix is rescaled to have a spectral radius of unity.

At time step m, the inner nodes of reservoir $\vec{s}_m$ are updated from the previous state $\vec{s}_{m-1}$ with the driving signal $\vec{u}_m$ using a leaky integrator[19] by

$$\vec{s}_m = (1-\alpha)\vec{s}_{m-1} + \alpha \tanh\left(\nu \boldsymbol{W}_{\text{in}}\vec{z}_m + \rho \boldsymbol{W}\vec{s}_{m-1}\right), \qquad (1)$$

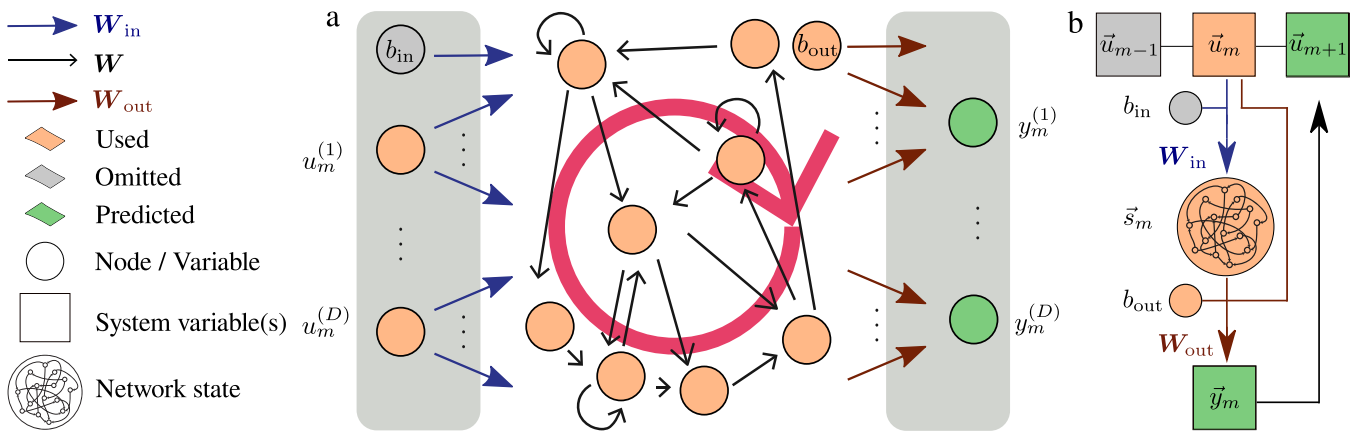

FIG. 1. Schematic illustration of the structure and use of an ESN in an iterative application. In (a), the inner structure of the reservoir is shown (inspired by Zimmermann and Parlitz[43]). Panel (b) shows a schematic illustration of one step of an iterative application of an ESN.

where ν, ρ, and α are three hyperparameters, which specify the input, the spectral radius of the inner coupling, and the leaking rate of the reservoir dynamics, respectively.

Predictions $\vec{y}_m$ of a reservoir consist of a trained linear superposition $\boldsymbol{W}_{\text{out}}$ of the driving signal, the reservoir states, and a constant output bias $b_{\text{out}} = 1$, which are summarized in the extended state vector $\vec{x}_m = [b_{\text{out}}, \vec{s}_m, \vec{u}_m]$, that is $\vec{y}_m = \boldsymbol{W}_{\text{out}}\vec{x}_m$. The readout function of reservoir computing can be extended by using superpositions of the nonlinear basis functions of the extended state vector, such as polynomials,[27,35] which not only improve nonlinear approximation but can also break unwanted symmetries.[36] Predictions of the reservoir are trained by recording the input, the reservoir's response, and the desired output over time—collected in an extended state matrix $\boldsymbol{X}$ and a target matrix $\boldsymbol{Y}$—and choosing $\boldsymbol{W}_{\text{out}}$ using Tichonov regularization,[37]

$$\boldsymbol{W}_{\text{out}} = \boldsymbol{Y}\boldsymbol{X}^T\left(\boldsymbol{X}\boldsymbol{X}^T + \beta\boldsymbol{I}\right)^{-1}, \text{which minimizes}$$
$$\sum_m \|\vec{y}_m^{\text{true}} - \boldsymbol{W}_{\text{out}}\vec{x}_m\|^2 + \beta\|\boldsymbol{W}_{\text{out}}\|_2^2. \tag{2}$$

A schematic visualization of the structure of the ESN is shown in Fig. 1(a).

Due to memory of the reservoir (added by recurrent connections of $\boldsymbol{W}$ and leaky integration), the uniformly, randomly initialized reservoir state $\vec{s}_0 \in [0,1)^N$ has an influence on the subsequent reservoir states $\vec{s}_m$ for $m \geq 1$ and, hence, its predictions $\vec{y}_m$. If the echo-state-property[1,38] is fulfilled, this influence decays in magnitude over time, s.t. the reservoir's response is asymptotically uniquely determined by the given input time series and different initializations of $\vec{s}_0$ converge to identical reservoir states $\vec{s}_m$ for m sufficiently large. However, the memory of the reservoir requires a sufficiently long transient time t_{trans} in which the reservoir is driven by a ground truth input signal. During the transient time, reservoir states are not recorded in the training process and predicted values are not used in the evaluation.

If the input signal is generated by a dynamical system, the echo-state-property is closely related to generalized synchronization[26] and the reservoir system may effectively provide a delay embedding of the attractor from which the input signal is obtained.[39–42]

B. Task: Iterative time series prediction and valid time

In this work, reservoir performance is evaluated on the task of iterative time series prediction, which is a commonly studied problem.[27,44] Here, the reservoir is trained and used to predict one time step ahead, i.e., $\vec{y}_m = \vec{u}_{m+1}$, and is iteratively applied to its own output. Figure 1(b) visualizes one step of the iterative application. Note that this task differs greatly from considering training errors (as in Ref. 22), one-step-ahead or cross predictions (as considered in Refs. 15, 20, 22, and 26). In iterative predictions, not only the readout of the reservoir might deviate from the truth, but the reservoir dynamics is driven by a predicted—and, hence, error-prone—driving signal.

The quality of the prediction is measured in valid times t_{valid}, i.e., the time length of predictions for which the normalized error $E(t)$ does not exceed an error tolerance e,

$$E(t) = \frac{\|\vec{u}(t) - \vec{u}^{\,\text{true}}(t)\|}{\langle\|\vec{u}^{\,\text{true}}(t)\|^2\rangle_t^{1/2}} \quad \text{and} \quad t_{\text{valid}} = \max_{E(t)<e} t, \tag{3}$$

where $\langle\cdot\rangle_t$ denotes the temporal mean. The valid time is a short term prediction measure, which does not measure the correct prediction of the attractor climate. An example of the measurement of valid times is shown in Fig. 2 for the Lorenz-63 system (see Sec. II C), with complete knowledge of the system state (see Panel **a**) and sole knowledge of the univariate y-time series (see Panel **b**).

C. Test systems: With full or partial state knowledge

We evaluate the performance of different reservoir computing approaches using three test systems. For each system, two

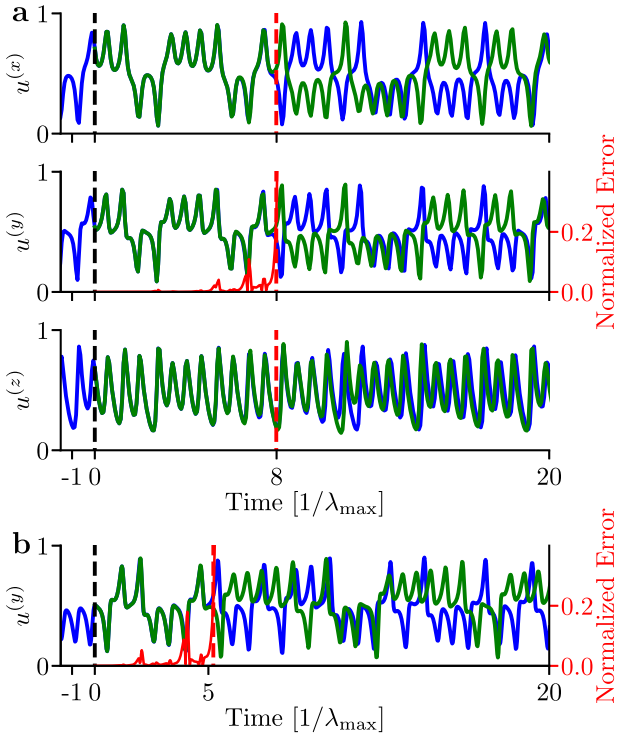

FIG. 2. Performance is measured using the valid time. The figure visualizes how the valid time of a prediction is computed by showing the true time series (*blue*) and the iterative prediction (*green*). The normalized error E (*red*) is shown for $E(t) \leq e$. In (a), the iterative prediction of the Lorenz-63 system with full state knowledge is shown. The normalized error is calculated using the y-coordinate only. It exceeds the error threshold $e = 0.2$ at $t_{\text{valid}} \approx 8$ Lyapunov times. Panel (b) shows an iterative prediction of the Lorenz–63 system with partial state knowledge. Here, the valid time is $t_{\text{valid}} \approx 5$ Lyapunov times.

distinct cases are considered: Predictions with full state knowledge and predictions with partial state knowledge. That means, either a time series of the full system state is used as the training input and target time series $\vec{u}(t)$ of the reservoir prediction or only a subset of the system variables is used. The latter is a measurement of the system state and, hence, represents the most common real-world prediction tasks. The focus on the distinction between these two cases goes back to Storm *et al.*,[14] who show that the need of careful hyperparameter tuning is restricted to the case of partial state knowledge. To ensure a precise comparison between the two cases, we consistently evaluate the error, on which the valid time is based [see Eq. (3)], using only the variables relevant in the context of partial state knowledge. Furthermore, for all systems considered, the system's variables are rescaled, such that their values are in $[0, 1]$ using the minimum and maximum values of each variable across all training and testing trajectories. The three test systems are briefly introduced in this section.

The Lorenz-63 system[45] with differential equations

$$\begin{aligned} \frac{\mathrm{d}u^{(x)}}{\mathrm{d}t} &= a\left(u^{(y)} - u^{(x)}\right), \\ \frac{\mathrm{d}u^{(y)}}{\mathrm{d}t} &= u^{(x)}\left(b - u^{(z)}\right) - u^{(y)}, \\ \frac{\mathrm{d}u^{(z)}}{\mathrm{d}t} &= u^{(x)}u^{(y)} - cu^{(z)}, \end{aligned} \tag{4}$$

and parameters $a = 10$, $b = 28$, and $c = 8/3$ exhibits chaotic dynamics with the largest Lyapunov exponent $\lambda_{\max} = 0.9056 \pm 0.0001$. In time series predictions, either full state knowledge $\vec{u}(t)$ or (for partial state knowledge), the $u^{(y)}$-variable is used. For training, we use time series of length $T = 500$ time units, with a step size of the dynamical system $\Delta t_{\text{ds}} = 0.01$. A transient time for testing and evaluating predictions of $t_{\text{trans}} = 100$ time units is used. Exemplarily, time series of the system, along with iterative predictions, are shown in Fig. 2 for full state knowledge [Fig. 2(a)] and partial state knowledge [Fig. 2(b)].

The Hindmarsh–Rose system[46] in simplified form follows the differential equations[47]

$$\begin{aligned} \frac{\mathrm{d}u^{(x)}}{\mathrm{d}t} &= -\left(u^{(x)}\right)^3 + 3\left(u^{(x)}\right)^2 + u^{(y)} - u^{(z)}, \\ \frac{\mathrm{d}u^{(y)}}{\mathrm{d}t} &= 1 - 5\left(u^{(x)}\right)^2 - u^{(y)}, \\ \frac{\mathrm{d}u^{(z)}}{\mathrm{d}t} &= \xi\left(u^{(x)} - \frac{u^{(z)} - z_0}{4}\right), \end{aligned} \tag{5}$$

with $\xi = 0.004$ and $z_0 = 3.19$. The maximal Lyapunov exponent $\lambda_{\max} = 0.0032 \pm 0.0001$ is calculated using linearized system equations. In time series of partial state knowledge, a time series of the $u^{(x)}$-variable is used, which represents the membrane potential of a single neuron. This variable exhibits complex spiking and bursting behavior characteristic of neuronal dynamics. For training, we use a time series of length $T = 10\,000$ time units, with a step size of the dynamical system $\Delta t_{\text{ds}} = 0.2$, and a transient time of $t_{\text{trans}} = 2000$ time units.

The Lorenz-96 system[48] follows the differential equations:

$$\begin{aligned} \frac{\mathrm{d}u^{(j)}}{\mathrm{d}t} &= \left(u^{(j+1)} - u^{(j-2)}\right)u^{(j-1)} - u^{(j)} + F, \\ \text{with} \quad u^{(j)} &= u^{(j+D)} \quad \forall j \in \mathbb{N}. \end{aligned} \tag{6}$$

In the case considered here, forcing equals $F = 8$ and the system dimension D ranges from 5 to 15. The corresponding largest Lyapunov exponents $\lambda_{\max}$ are calculated in Julia using the library DynamicalSystems.jl[49] and the results are summarized in Table I. Similarly, the Kaplan–Yorke dimension D_{KY} is calculated and depicted in Table I. The results show a nearly linear increase of the Kaplan–Yorke dimension with increasing system dimensions. For the Lorenz-96 system, multiple experiments with partial system state knowledge are tested. For different system dimensions, the measurement spacing $k \in \{1, 2, 3, 4\}$ specifies that partial state knowledge is restricted to each kth variable of the system state.[50] For training, we use a time series of length $T = 500$ time units, with a step size of the

TABLE I. The table shows computational results of the largest Lyapunov exponents λ_{max} and the Kaplan–Yorke dimension D_{KY} for the Lorenz-96 system with different system dimensions D.

D	5	6	7	8	9	10	11	12	13	14	15
λ_{max}	0.47	0.95	1.25	1.61	1.23	1.18	1.37	1.47	1.48	1.46	1.48
D_{KY}	2.92	4.03	4.55	5.38	5.66	6.51	7.42	8.19	8.75	9.39	10.09

dynamical system of $\Delta t_{ds} = 0.01$ time units. Here, the transient time equals $t_{trans} = 100$ time units.

D. Delayed values

The extension of delayed values aims to improve the prediction capabilities of the classical reservoir scheme as discussed above by reusing historical values in a time-delayed manner.[24,26] In the current study, (time-) delayed values of the input signal $\vec{u}$—delayed input—and/or the reservoir state vector $\vec{s}$—delayed reservoir—are reused. Figure 3 schematically illustrates a single time step of the iterative prediction process for different methods of incorporating the delayed values. In contrast to Fig. 1(b), this diagram depicts solely the dynamic variables. The color coding highlights whether components are a part of the extended state vector (used—orange) or not (omitted—gray).

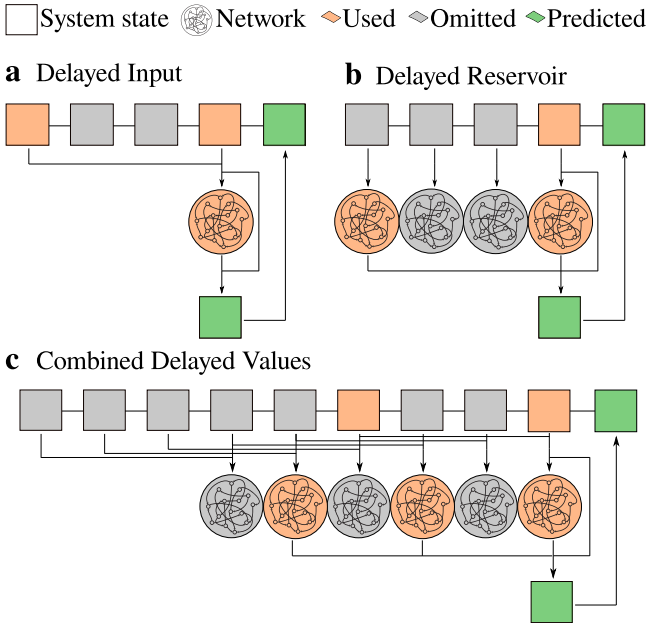

FIG. 3. The proposed extensions introduce delayed values to enhance reservoir predictions. In (a), the inclusion of delayed input signals, extending the reservoir's input $\vec{z}_m$ and extended state vector $\vec{x}_m$, is shown. The incorporation of delayed reservoir states by extending the extended state vector $\vec{x}_m$ is shown in (b). Panel (c) shows the combination of the previous two extensions with more than two delayed reservoir states. Only dynamic variables are depicted. Coloring marks the (dynamic) elements used (*orange*) or not used (*gray*) by the output matrix $\boldsymbol{W}_{out}$.

The use of time-delayed input signals[15] is depicted in Fig. 3(a). Here, the driving input is supplemented with delayed input values. In general, this extension entails two new hyperparameters: the number of delayed inputs N_I and the time step of the delayed input Δt_I. The latter is a multiple of the prediction time step Δt, i.e., $\Delta t_I := i_{IN} \Delta t$, with $i_{IN} \in \mathbb{N}$. In Fig. 3(a), the extension is shown for $N_I = 1$ delayed input with a time step $\Delta t_I = 3\Delta t$. Explicitly, the input vector $\vec{z}_m$ and the extended state vector $\vec{x}_m$ are extended by the delayed values, such that

$$\vec{z}_m = [b_{in}, \vec{u}_m, \vec{u}_{m-i_{IN}}, \ldots, \vec{u}_{m-N_I i_{IN}}]$$

and

$$\vec{x}_m = [b_{in}, \vec{s}_m, \vec{u}_m, \vec{u}_{m-i_{IN}}, \ldots, \vec{u}_{m-N_I i_{IN}}].$$

This approach effectively uses a delay-embedded vector as the input, following the principles of Takens' theorem,[33,34] while predicting only the next time step $\vec{u}_{m+1}$ as the output. A similar method was investigated by Jaurigue *et al.*,[15] who report a decreased need of hyperparameter optimization in linear prediction tasks.

The use of delayed reservoir states[20,22,23] is depicted in Fig. 3(b). In this case, the extended state vector is supplemented with delayed values from the reservoir's internal dynamics. Hence,

$$\vec{x}_m = [b_{in}, \vec{s}_{m-i_R}, \ldots, \vec{s}_{m-N_R i_R}, \vec{u}_m],$$

where N_R, the number of delayed reservoir states, and $\Delta t_R = i_R \Delta t$, the time step of the delayed reservoir state, are two new hyperparameters. Figure 3(b) depicts the use of delayed reservoir states with $N_R = 2$ reservoir states with a time step $\Delta t_R = 3\Delta t$. Prior work by Sakemi *et al.*[20] has shown for single-step prediction tasks that by using delayed reservoir states, the number of inner nodes N can be reduced significantly without a substantial deterioration of the reservoir's performance.

Combining both methods yields a combined use of the delayed values.[26] A schematic representation of the prediction of one time step using delayed values is shown in Fig. 3(c). For Illustration, $N_I = 2$ input states with time steps $\Delta t_I = 3\Delta t$ and $N_R = 3$ reservoir states with delayed time steps $\Delta t_R = 2\Delta t$ are depicted.

E. Evaluation of hyperparameters and delayed values

As sketched in the introduction, the problem of proper evaluation of the newly proposed method—delayed values—is encountered in this article. We evaluate the use of delayed values, i.e., the introduction of four hyperparameters, with respect to the seven classical hyperparameters. The hyperparameters are summarized in Table II.

For each hyperparameter set, we train reservoirs on a single-step prediction task using a sample of M random reservoirs, each

TABLE II. The table shows all classical and delay related hyperparameters tested.

Classical hyperparameter		Delay hyperparameters	
N	Node number	N_I	Delayed input states
ε	Sparseness	Δt_I	Delayed input step size
ρ	Spectral radius	N_R	Delayed reservoir states
ν	Input scaling	Δt_R	Delayed reservoir step size
β	Regularization constant		
α	Leaking rate		
Δt	Step size		

trained on different trajectories of a given test system. We then evaluate these reservoirs across K distinct evaluation trajectories. The performance is assessed iteratively using the valid time [see Eq. (3)] for each trajectory. From the resulting $M \times K$ valid times, we compute the mean valid time, which serves as the quality metric for the corresponding hyperparameter set. The random structure of the reservoirs is not fixed across different hyperparameter sets. To neglect performance fluctuations based on different evaluation data sets, we use the standard deviation between the mean (over K evaluation trajectories) reservoir performances as the uncertainty of the measurement. It is important to note that performance variability is more pronounced across different evaluation data sets (i.e., initial conditions of the prediction) than between randomly drawn reservoirs. To ensure consistency and comparability across experiments, we use the same starting points of predictions within the evaluation time series.

We first optimize the classical hyperparameters based on the mean valid time using a grid search method. For the numerical experiments in Sec. III A, we subsequently start a local optimization scheme (downhill simple/Nelder Mead[51]) using the SciPy[52] library, from the best set of hyperparameters found in the grid search method. Here, we fix the seeds of the random networks to avoid fluctuations. The found maximum mean valid time serves as a benchmark of the classical reservoir computing scheme.

In the second step, we compare the best classical performance to the performance using delayed values. Therefore, we fix the classical hyperparameters to the best performing set and vary the numbers of delayed input N_I and reservoir frames N_R, their time steps Δt_{IN}, Δt_R, and the regularization constant β.

Within the scope of the considered hyperparameters and hyperparameter ranges, we fully optimize the classical method. In difference to that, we do not explicitly adjust classical hyperparameters to the perturbed task of time series prediction with delayed values. We, therefore, compare a highly optimized classical method with a partially optimized, delayed values method.

III. RESULTS

A. Delayed values using full or partial state knowledge

Within this section, we compare the mean valid time of the best performing classical reservoir computing method with reservoirs using delayed values for three different dynamical systems (compare Sec. II C): The Lorenz-63 system [Eq. (4)], the Lorenz-96 system with $D = 5$ dimensions [Eq. (6)], and the Hindmarsh–Rose system [Eq. (5)]. Doing so, we distinguish between predictions with full state knowledge, i.e., the full system state vector is used as input, and predictions where only partial state knowledge is available, i.e., only a subset of the system variables are used as input. This distinction is motivated in Fig. 4 by comparing reservoir performances with and without reservoir memory for the classical reservoir computing method (without delayed values).

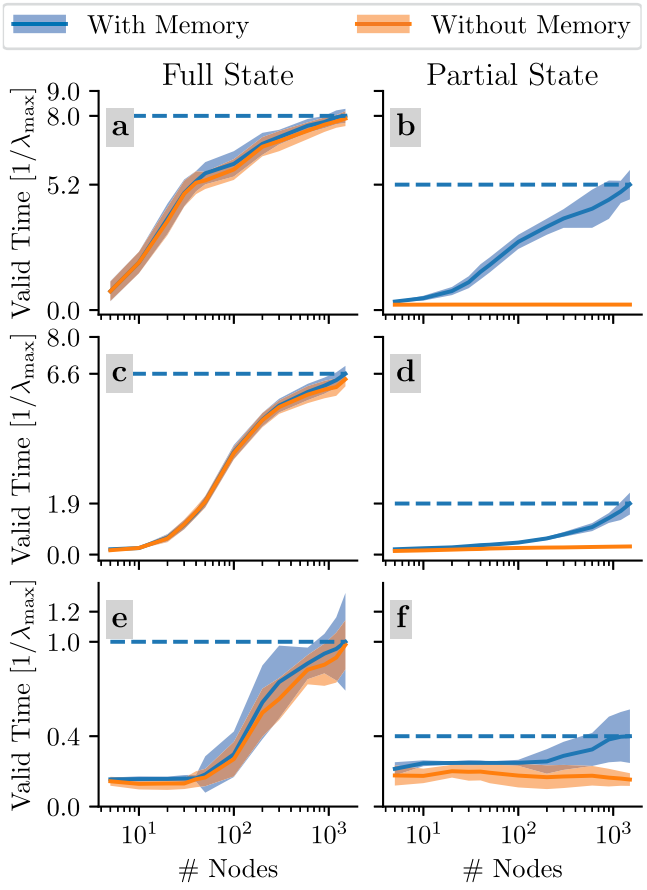

FIG. 4. Reservoir memory increases prediction performance only for partial state knowledge. The mean valid times of time series predictions with full (*left*) or partial (*right*) system state knowledge is compared using reservoir computing with or without memory. For full state knowledge, valid times do not increase by using reservoir memory. For partial state knowledge, reservoir memory significantly increases mean valid times of time series predictions. The results are shown for different dynamical systems: (a) and (b) show the results of time series predictions of the Lorenz-63 system, (c) and (d) for the Lorenz-96 system, and (e) and (f) for the Hindmarsh–Rose system, respectively.

In Fig. 4(a), the mean valid prediction times for the Lorenz-63 system using classical reservoirs with full state knowledge are displayed. The figure compares the performance of reservoirs with memory (*blue*) and without memory (*orange*), where the latter corresponds to a spectral radius of $\rho = 0$ and a leaking rate of $\alpha = 1$.[53] Other hyperparameters were optimized for each network size individually.

The results consistently indicate that the reservoir's performance is not influenced by its memory if the step size is optimized. Thus, optimizing hyperparameters related to memory, such as spectral radius and leaking rate, does not significantly impact performance in this case. Similarly, this implies that tuning parameters governing the reservoir's adjacency matrix, $\boldsymbol{W}$, is largely unnecessary.

Independence on memory in case of full state knowledge has also been discussed by Jaurigue[54] who studied very small networks and the influence of topology. Other prior studies also noted this observation,[14,24,55–57] but they do not explicitly emphasize that reservoir memory does not improve the prediction accuracy in these particular tasks. The lack of dependency on memory further underscores the limited importance of optimizing memory-related hyperparameters in these systems.

Figure 4(b) shows the results for predictions with partial state knowledge, i.e., only a time series of the $u^{(y)}$-coordinate is iteratively predicted. In this case, the memory of the reservoir plays a central role for qualitative predictions, because it provides an internal delay reconstruction of the input dynamics.[39–42] Figures 4(c) and 4(d) show similar results for the Lorenz-96 system. In the case of partial state knowledge [see Fig. 4(d)], only a time series of variables $u^{(0)}, u^{(2)}$, and $u^{(4)}$ is iteratively predicted. Figures 4(e) and 4(d) show similar results for the Hindmarsh–Rose system, where only the $u^{(x)}$-time series is used in the case of partial state knowledge.

Figure 5 shows the deviation in performance from the use of delayed values to the classical approach by varying the number of delayed inputs and reservoir states.

The classical hyperparameters are fixed to the best performing set of the classical case. The regularization and the step size of delayed inputs and reservoirs are optimized for each combination of the numbers of delayed input and reservoirs. Figure 5(a) shows the results for predictions using full state knowledge for the Lorenz-63 system. The use of delayed values worsens the performance for all variations of numbers of delayed input and reservoir frames. Figure 5(b) depicts the results of the same experiment with partial state knowledge. In particular, the combination of delayed inputs and delayed reservoirs increases the mean valid time of time series predictions. Figures 5(c) and 5(d) show similar, but less pronounced results for time series predictions of the Lorenz-96 system and Figs. 5(e) and 5(f) for time series predictions of the Hindmarsh–Rose system with full (*left*)—or partial state knowledge (*right*), respectively. In contrast to the results of the Lorenz-63 system, achieving improved performance requires a careful selection of delayed inputs and reservoirs. Notably, significant performance gains are observed primarily in the case of partial state knowledge, where the reservoir memory plays a crucial role in achieving accurate predictions (see Fig. 4). However, classical hyperparameters are optimized based on time series predictions without the use of delayed values. Therefore, the depicted performances of the delayed values can also be further increased by adjusting the classical hyperparameters.

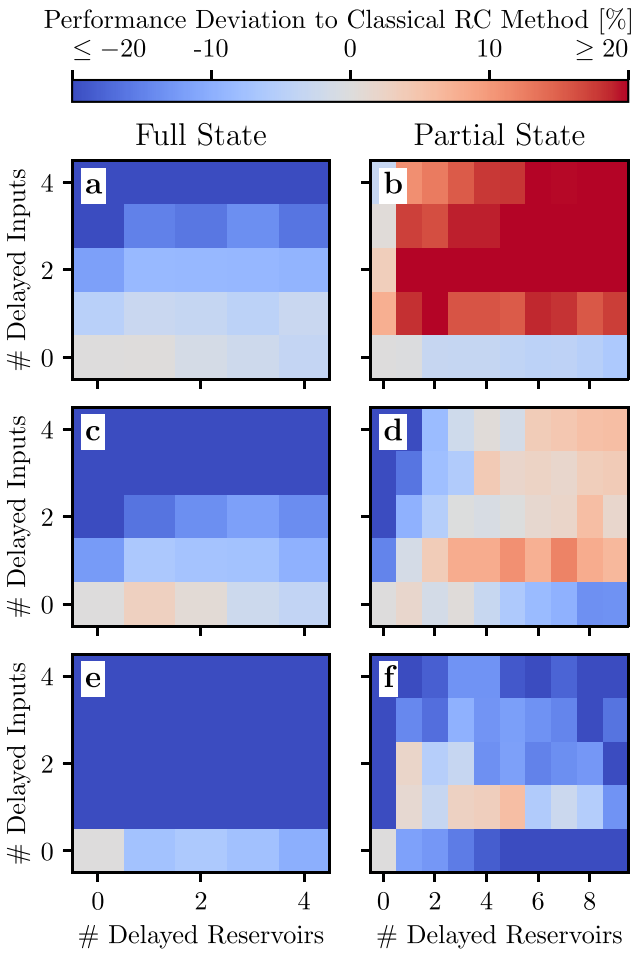

FIG. 5. Delayed values increase prediction performance for time series predictions with partial state knowledge. The deviation of mean valid times using delayed values to mean valid time of the classical reservoir are shown. For time series predictions with full state knowledge (*left*), the use of delayed values shortens the mean valid time of predictions. For time series predictions with partial state knowledge, significant performance improvements can be achieved. The results are depicted for different systems. (a) and (b) show the results of time series predictions of the Lorenz-63 system, (c) and (d) of the Lorenz-96 system, and (e) and (f) of the Hindmarsh–Rose system, respectively.

B. Increasing system and attractor dimensionality

While larger system and attractor dimensions are common in real applications, the systems considered so far have only a small system and attractor dimension. To get a first glance on the performance of delayed values using larger system dimensionality, we gradually increase the dimension of the Lorenz-96 system from $D = 5$ to $D = 15$. For the Lorenz-96 system, the Kaplan–Yorke dimension—and, hence, the complexity of the prediction task—increases approximately linearly with increasing input dimension (see Table I and Refs. 58 and 59). Similar to Sec. III A, we distinguish between full and partial state knowledge. To enlarge the analysis of partial system state observations, we use the measurement spacing $k \in \{1, 2, 3, 4\}$, i.e., we restrict the input/predicted time series to each kth variable of the dynamical system state. Similar to Sec. III A, we evaluate the performance increment using delayed values.

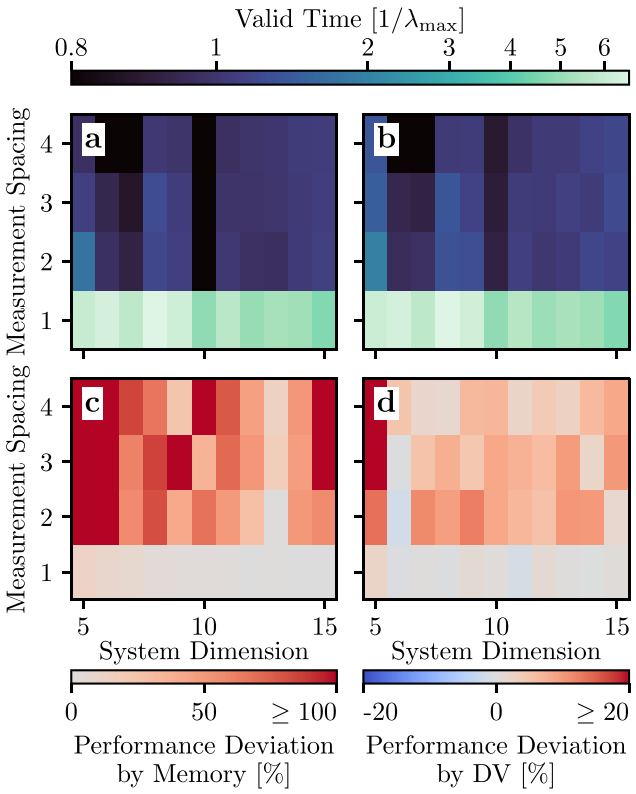

FIG. 6. Predictions of different system dimensions of the Lorenz-96 system show results similar to those in Secs. II A and II B: Reservoir memory enhances predictions if only partial state knowledge exists (measurement spacing $k > 1$) and time series predictions with partial state knowledge are improved by using delayed values. (a) shows best mean valid times for the Lorenz-96 system with different system dimension and measurement spacing using classical reservoir computing with memory. (b) shows similar results using delayed values. (c) shows the deviation in valid time of reservoirs with memory, shown in (a), to reservoirs without memory. (d) shows the deviation in valid time of reservoirs using delayed values, see (b), to classical reservoirs with memory, see (a).

Figure 6 shows that the results presented so far generalize for higher dimensional systems. Figure 6(a) shows mean valid times for different system dimensions D and measurement spacings k using the best set of classical hyperparameters. Similar to the results shown in Fig. 4, we find that valid predictions using partial system state knowledge are shorter than valid predictions obtained with full system knowledge. In fact, mean valid times of iterative predictions with partial system knowledge are mostly below the value of 2. Figure 6(c) shows the relative improvement of the prediction quality due to memory of the reservoir, i.e., comparing the best mean valid time for $\alpha = 1$ and $\rho = 0$ with the best mean valid time for all hyperparameters. Similar to Fig. 4, large improvements in valid prediction length are found for partial system state knowledge (measurement spacing $k > 1$). Only minor improvements exist for full system state knowledge (measurement spacing $k = 1$).

Figures 6(b) and 6(d) show the performance of the reservoir computing approach using delayed values; (b) in absolute valid times and (d) comparing to results without the use of delayed values. In agreement to the previous results (compare Fig. 5), the use of delayed values increases the mean performance of time series prediction in the case of partial state knowledge.

C. Missing hyperparameter optimization

As our results seem to differ from previous experiments[20,23,26] and delayed values are also discussed in terms of easing the hyperparameter optimization task,[15] we test the method of utilizing delayed values on non-optimal classical hyperparameter sets. Therefore, we deliberately choose not the best performing set of classical hyperparameters, vary the number of delayed reservoir and input states, and compare the performance to the classical approach with similar hyperparameters. We choose the time step of the driving system Δt and the leaking rate α as non-optimal hyperparameters and exemplary restrict our experiments to full state knowledge predictions of the Lorenz-63 system.

Figure 7(a) shows the best mean valid times, predicting the Lorenz-63 system with full state knowledge, for a given set of Δt and α, where all other classical hyperparameters are optimized. The parameters are positively correlated. Therefore, densely sampled trajectories, i.e., small system time steps ($\Delta t \leq 0.04$), require leaky integration and, hence, reservoir memory, for good performances. Increasing the system time steps increases the best performing leaking rate. For non-leaky reservoirs, i.e., $\alpha = 1$, good performance is only achieved within a range of system time steps, which agrees with the findings of Tsuchiyama *et al.*[21] The best performing sets of α and Δt, with mean valid time $t_{valid} = 8.0 \pm 0.3$, are marked with a cross. In addition, three non-optimal hyperparameter sets, with valid time $t_{valid} \in [6.6, 7.1]$, are marked with a dot.

Each panel of Figs. 7(a)–7(c) shows the deviation of performance to the classical reservoir computing approach when utilizing a given number of delayed input or reservoir states. The classical hyperparameters in each panel correspond to the hyperparameter sets marked and labeled in Fig. 7(a). Similar to Secs. III A and III B, the step sizes of the delayed values are optimized for each set of delayed reservoir and input states. For the classical hyperparameters of **b**, the system time step is larger than optimal, and decreasing the leaking rate worsens classical predictions. In this case, the reservoir memory, in the form of the leaking rate, degrades the overall performance. Furthermore, using larger time steps also exacerbates the prediction errors, leading to worse outcomes. Here, the use of delayed input values, i.e., $N_I \geq 1$, greatly deteriorates prediction performances. Exclusively re-using delayed reservoirs, i.e., $N_I = 0$ and $N_R \geq 1$, leads to small performance increments. However, the best performing hyperparameter set of delayed values results in a valid time which is 7.3 ± 0.4 and, hence, smaller than the best performing classical hyperparameter set with a mean valid time of $t_{valid} = 8.0 \pm 0.3$. In Fig. 7(c), the system time step is chosen optimal, while increases in the leaking rate could further increase performances. The use of delayed reservoirs or combined delayed values can greatly improve prediction performances. Here, the best classical mean valid time can be recovered by the selection of suitable numbers (N_R and N_I) and time steps (Δt_R and Δt_I) of delayed

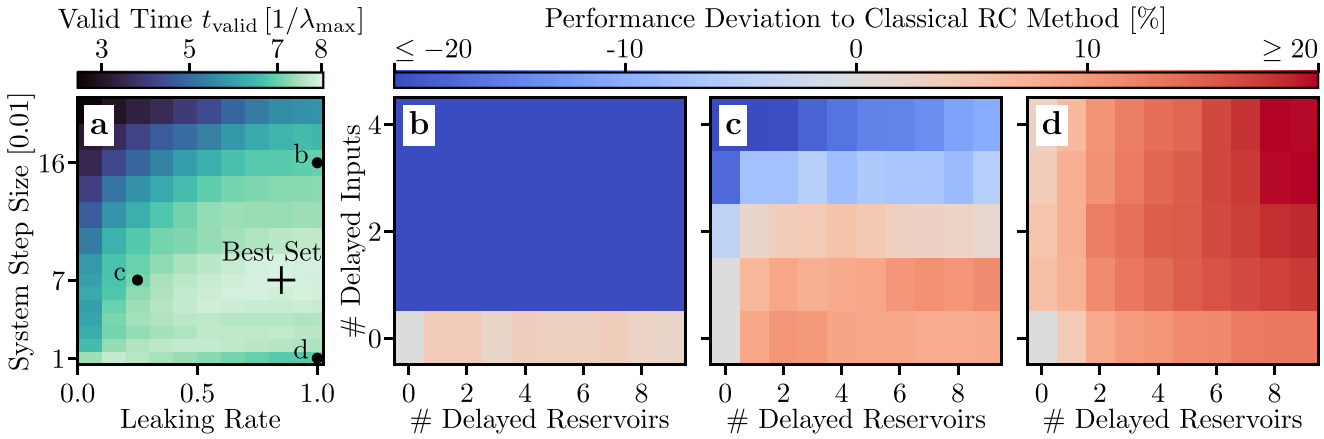

FIG. 7. Prediction performance of reservoirs with non-optimal classical hyperparameters is enhanced with delayed values. Panel (a) shows absolute valid times for different combinations of system step sizes Δt and leaking rates α and marks the chosen sets of non-optimal classical hyperparameters. The relative deviation in the performance of reservoirs using delayed values to classical reservoir computing with non-optimal hyperparameters is shown in panels (b)–(d), where the tested classical hyperparameter sets are marked in panel (a).

values, retaining non-optimal classical hyperparameters. The best mean valid time of Fig. 7(c) is 8.0 ± 0.4. In Fig. 7(d), the system time step is smaller than optimal. Despite the iterative predictions with full system state knowledge for $\Delta t = 0.01$, predictions with reservoir memory ($\alpha < 1$) perform better than the chosen set of hyperparameters. The use of delayed values similarly enhances reservoir predictions, which is in agreement with the results in Sec. III A. Figure 7(d) shows growing performance gains with increasing numbers of delayed inputs and delayed reservoirs. In this case, the best set of delayed values ($N_R = 8$ and $N_I = 4$) with non-optimal classical hyperparameters exceeds the classical optimum with a valid time of 8.2 ± 0.4. In summary, it is shown that predictions with full state knowledge of reservoirs with non-optimal classical hyperparameters can be improved by utilizing delayed values.

IV. DISCUSSION AND OUTLOOK

We conducted a comprehensive analysis of augmenting classical reservoir computing with time-delayed input and reservoir values. The method's performance was evaluated on the task of iterative time series prediction across multiple dynamical systems of varying complexity. By testing and comparing the mean performance of ideal reservoirs optimized over a wide range of classical hyperparameters (see Table II), we identified the critical impact of the difference between predictions based on full vs partial knowledge of the chaotic system's state. Our findings show that the reservoir computer's memory and the incorporation of delayed values significantly improve the prediction accuracy if only partial state knowledge is available. In addition, we confirmed that delayed values can enhance predictions even under non-optimal hyperparameter configurations, where full system state knowledge is accessible.

While reservoir memory is a mandatory component for good iterative predictions with only partial state knowledge available, the performance of iterative predictions with full system state knowledge does not depend on reservoir memory [compare Sec. III A, Fig. 4 and Sec. III B, Fig. 6(c)]. Here, reservoir computing without memory refers to a simplified version of the reservoir computing framework where the internal reservoir dynamics, typically characterized by recurrent connections and memory capabilities, are absent. In this setup, the system operates primarily as a linear projection (input matrix) with a non-linear transformation (activation function) followed by a linear readout. The absence of recurrent dynamics is advantageous in those scenarios where memory is not required, simplifying the architecture (reducing the complex hyperparameter optimization task[14]) and reducing computational complexity. The system relies solely on the transformation capabilities of the non-linear layer and the readout's ability to linearly combine these transformed features to produce accurate outputs. This class of static data driven modeling approaches includes extreme learning machines[60] and next generation reservoir computing,[53,57] which have been demonstrated to be very efficient, if full state knowledge is available.

For predictions using only partial system state knowledge, the method of reusing optimized time-delayed values outperforms the strongly optimized classical reservoir computers across different dynamical systems with varying complexity. In particular, the combination of delayed input and reservoir variables enhances prediction capabilities. In contrast, when the full system state is known, incorporating delayed values offers little to no performance improvement in iterative predictions. In fact, using time-delayed values often deteriorates the prediction accuracy. Here, it needs to be stressed that mean valid times of the delayed values method can be further increased by re-optimizing the classical hyperparameters. A simultaneous global optimization of 11 hyperparameters (classical and delay related) would achieve a better comparison between different methods but is beyond our computational feasibility. This hints at the problem of using time-delayed values: They increase the count of hyperparameters, aggravating the often complex hyperparameter optimization task. Furthermore, especially the utilization of time-delayed input values might even worsen

the prediction capabilities. But the use of delayed reservoir states sometimes also decreases the prediction performance in iterative predictions, which might be an effect of training on single-step predictions and evaluating performance on iterative applications.[61] This explains the differences to previous utilizations of delayed values through the literature,[20,22,26] which generally found increased performances.

Reservoir memory and delayed values are similar in the sense that time-delayed information is reused. In this work, we find similar cases of beneficial applications. These are prediction tasks with only partial system state knowledge, where from a theoretical standpoint, historic values are required to reconstruct the full dynamical system state.[33,34]

For a less thorough optimization of classical hyperparameters, the prediction performance can also be improved in predictions with full system state knowledge using delayed values. We tested this by deliberately choosing non-optimal values for two hyperparameters—the reservoir leaking rate α and time step of the prediction task Δt—which are commonly not considered (for instance, in Refs. 23 and 20). The improvement is the greatest in Fig. 7(d), where also the classical reservoir computing method is improved by reservoir memory (compare Sec. III C, Fig. 7(a), $\Delta t = 0.01$). However, delayed values also increase the prediction performances for the other tested hyperparameter sets, where classical reservoir memory does not increase the prediction performances. These results agree with the previous findings[15] that highlight the ease of hyperparameter optimization by introducing delayed input values and offer a second explanation to deviating prior findings[20,22,23,26] of improved predictions using full state knowledge.

At this point, it remains unclear why incorporating time-delayed input values for predictions, when only partial system state knowledge is available, fails to match the performance of predictions based on full state knowledge. The optimized use of time-delayed inputs is analogous to a delay embedding, which is diffeomorphic to the full dynamical system. However, for open-loop predictions, it has been observed that the best performing delayed step sizes are non-uniform[62]—a possibility which is not tested in the current study. Whether these results are also valid for iterative predictions and whether non-uniform delayed time steps can achieve similar performances to predictions with full state knowledge available require further investigation that might hold great potential, as it may unlock even greater improvements when time-delayed values are used effectively.

A comprehensive approach to optimizing all 11 hyperparameters simultaneously would be another promising step in this direction. Moreover, this simultaneous optimization could help clarify whether real performance gains, beyond those achievable through hyperparameter optimization, are limited to predictions based on partial system state knowledge.

Although the use of delayed values comes with additional hyperparameters and more research is required on time-delayed values, we showed the capability of the approach to enhance iterative predictions for memory requiring tasks. In practical applications, this is the most common case of iterative time series predictions, as predicting time series of only some variables or measured quantities of dynamical systems, i.e., predictions with only partial knowledge of the dynamical system state, are the most common. In addition, we confirmed and extended the use of delayed values as a method of easing the classical hyperparameter task.

ACKNOWLEDGMENTS

The authors thank Stefan Luther and all members of the Biomedical Research Group for their support and interesting scientific discussion.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Luk Fleddermann: Conceptualization (supporting); Investigation (lead); Software (lead); Writing – original draft (lead); Writing – review & editing (equal). **Sebastian Herzog:** Conceptualization (supporting); Investigation (supporting); Supervision (supporting); Writing – review & editing (equal). **Ulrich Parlitz:** Conceptualization (lead); Project administration (lead); Supervision (lead); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

APPENDIX: HYPERPARAMETER OPTIMIZATION

Grid search for the optimization in Sec. III A is performed using the same hyperparameters for all tested systems. Due to computational reasons, we fixed some hyperparameters in the grid search performed for the results of Sec. III B. The tested ranges and chosen values are summarized in Table III.

TABLE III. Tested ranges and chosen values of hyperparameters. In optimizations of the classical method, the parameters above horizontal partitioning are optimized. In optimizations of delayed values, the hyperparameters below horizontal partitioning are optimized, while the other hyperparameters are set to the best performing found in the optimization of the classical method.

	Hyperparameter	Sec. III A	Sec. III B
N	Node number	[5, 1500]	1500
ε	Sparseness	$[10^{-4}, 1]$	$[10^{-4}, 1]$
ρ	Spectral radius	[0, 13]	[0, 13]
ν	Input scaling	[1, 13]	[0.01, 13]
β	Regularization constant	$[10^{-6}, 10^{-2}]$	10^{-4}
α	Leaking rate	[0.05, 1]	[0.1, 1]
Δt	Step size	$[1, 8] \cdot \Delta t_{\mathrm{ds}}$	Δt_{ds}
N_{I}	Delayed input states	[0, 9]	[0, 9]
Δt_{I}	Delayed input step size	$[1, 10] \cdot \Delta t$	$[1, 10] \cdot \Delta t$
N_{R}	Delayed reservoir states	[0, 9]	[0, 9]
Δt_{R}	Delayed reservoir step size	$[1, 10] \cdot \Delta t$	$[1, 5] \cdot \Delta t$
β	Regularization constant	$[10^{-6}, 10^{-2}]$	10^{-4}

We do not constrain the hyperparameters to regions that are commonly explored. For instance, the suggestion of using $\rho \leq 1$ is ignored, which goes back to $\rho = 1$ being the parameter where we are at the "edge of chaos,"[63,64] which is commonly discussed as the best choice for hyperparameters in echo state networks.[11] We do so because of three different reasons:

1. We find the best performing hyperparameters to be far outside of the suggested ranges (for instance, $\rho \geq 10$) (which is in agreement with Ref. 16).
2. The edge of chaos (or more exactly: stability) depends on more than the spectral radius.
3. It is not clear whether this actually optimizes the prediction in terms of valid times.

The mean valid times of the experiments in Sec. III A and Sec. III C are sampled using $M = 50$ different reservoir initializations, each evaluated on $K = 50$ testing time series. The results of Sec. III B are based on smaller sample sizes using $M = 20$ randomly drawn reservoirs, each evaluated on $K = 50$ testing time series.

REFERENCES

[1]H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," German National Research Center for Information Technology GMD Technical Report (2001), Vol. 148, p. 13.
[2]W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," Neural Comput. **14**, 2531–2560 (2002).
[3]H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," Science **304**, 78–80 (2004).
[4]D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," Neural Netw. **20**, 391–403 (2007).
[5]M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," Comput. Sci. Rev. **3**, 127–149 (2009).
[6]S. Shahi, F. H. Fenton, and E. M. Cherry, "Prediction of chaotic time series using recurrent neural networks and reservoir computing techniques: A comparative study," Machine Learn. Appl. **8**, 100300 (2022).
[7]F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "Other recurrent neural networks models," in *Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis*, edited by F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen (Springer International Publishing, Cham, 2017), pp. 31–39.
[8]A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian, "Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: Reservoir computing, artificial neural network, and long short-term memory network," Nonlinear Process. Geophys. **27**, 373–389 (2020).
[9]Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, "A review of deep learning models for time series prediction," IEEE Sens. J. **21**, 7833–7848 (2021).
[10]E. Bollt, "On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD," Chaos **31**, 013108 (2021).
[11]N. Bertschinger and T. Natschläger, "Real-time computation at the edge of chaos in recurrent neural networks," Neural Comput. **16**, 1413–1436 (2004).
[12]R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," Neural Netw. Echo State Netw. Liquid State Mach. **20**, 323–334 (2007).
[13]J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, "Information processing capacity of dynamical systems," Sci. Rep. **2**, 514 (2012).
[14]L. Storm, K. Gustavsson, and B. Mehlig, "Constraints on parameter choices for successful time-series prediction with echo-state networks," Mach. Learn.: Sci. Technol. **3**, 045021 (2022).
[15]L. Jaurigue and K. Lüdge, "Reducing reservoir computer hyperparameter dependence by external timescale tailoring," Neuromorphic Comput. Eng. **4**, 014001 (2024).
[16]J. J. Steil, "Online reservoir adaptation by intrinsic plasticity for backpropagation–decorrelation and echo state learning," Neural Netw. Echo State Netw. Liquid State Mach. **20**, 353–364 (2007).
[17]J. Yperman and T. Becker, "Bayesian optimization of hyper-parameters in reservoir computing," arXiv:1611.05193[cs] (2017).
[18]L. A. Thiede and U. Parlitz, "Gradient based hyperparameter optimization in echo state networks," Neural Netw. **115**, 23–29 (2019).
[19]H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky- integrator neurons," Neural Netw. Echo State Netw. Liquid State Mach. **20**, 335–352 (2007).
[20]Y. Sakemi, K. Morino, T. Leleu, and K. Aihara, "Model-size reduction for reservoir computing by concatenating internal states through time," Sci. Rep. **10**, 21794 (2020).
[21]K. Tsuchiyama, A. Röhm, T. Mihana, R. Horisaki, and M. Naruse, "Effect of temporal resolution on the reproduction of chaotic dynamics via reservoir computing," Chaos **33**, 063145 (2023).
[22]E. Del Frate, A. Shirin, and F. Sorrentino, "Reservoir computing with random and optimized time-shifts," Chaos **31**, 121103 (2021).
[23]B. A. Marquez, J. Suarez-Vargas, and B. J. Shastri, "Takens-inspired neuromorphic processor: A downsizing tool for random recurrent neural networks via feature extraction," Phys. Rev. Res. **1**, 033030 (2019).
[24]L. Jaurigue, E. Robertson, J. Wolters, and K. Lüdge, "Reservoir computing with delayed input for fast and easy optimisation," Entropy **23**, 1560 (2021).
[25]L. Jaurigue, E. Robertson, J. Wolters, and K. Lüdge, "Photonic reservoir computing with non-linear memory cells: Interplay between topology, delay and delayed input," SPIE Proc. **12204**, 61–67 (2022).
[26]U. Parlitz, "Learning from the past: Reservoir computing using delayed variables," Front. Appl. Math. Stat. **10**, 1221051 (2024).
[27]Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," Chaos **28**, 061104 (2018).
[28]B. K. Williams and E. D. Brown, "Partial observability and management of ecological systems," Ecol. Evol. **12**, e9197 (2022).
[29]L. Dabush, A. Kroizer, and T. Routtenberg, "State estimation in partially observable power systems via graph signal processing tools," Sensors **23**, 1387 (2023).
[30]J. Gong and S. Liu, "Partially observable collaborative model for optimizing personalized treatment selection," Eur. J. Oper. Res. **309**, 1409–1419 (2023).
[31]V. Gupta, L. K. B. Li, S. Chen, and M. Wan, "Model-free forecasting of partially observable spatiotemporally chaotic systems," Neural Netw. **160**, 297–305 (2023).
[32]R. Thundathil, F. Zus, G. Dick, and J. Wickert, "Assimilation of GNSS tropospheric gradients into the Weather Research and Forecasting (WRF) model version 4.4.1," Geosci. Model. Dev. **17**, 3599–3616 (2024).
[33]F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*, Lecture Notes in Mathematics Vol. 898, edited by D. Rand and L.-S. Young (Springer, Berlin, 1981), pp. 366–381.
[34]T. Sauer, J. A. Yorke, and M. Casdagli, "Embedology," J. Stat. Phys. **65**, 579–616 (1991).
[35]E. Bollt, "On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD," Chaos **31**, 013108 (2021).
[36]J. Herteux and C. Räth, "Breaking symmetries of the reservoir equations in echo state networks," Chaos **30**, 123142 (2020).
[37]A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," Technometrics **12**, 55–67 (1970).
[38]I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," Neural Netw. **35**, 1–9 (2012).
[39]A. Hart, J. Hook, and J. Dawes, "Embedding and approximation theorems for echo state networks," Neural Netw. **128**, 234–247 (2020).
[40]L. Grigoryeva, A. Hart, and J.-P. Ortega, "Chaos on compact manifolds: Differentiable synchronizations beyond the Takens theorem," Phys. Rev. E **103**, 062204 (2021).

[41] X.-Y. Duan, X. Ying, S.-Y. Leng, J. Kurths, W. Lin, and H.-F. Ma, "Embedding theory of reservoir computing and reducing reservoir network using time delays," Phys. Rev. Res. **5**, L022041 (2023).
[42] L. Pecora and T. Carroll, "Statistics for differential topological properties between data sets with an application to reservoir computers," arXiv:2409.04571[nlin.CD] (2024).
[43] R. S. Zimmermann and U. Parlitz, "Observing spatio-temporal dynamics of excitable media using reservoir computing," Chaos **28**, 043118 (2018).
[44] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," Chaos **28**, 041101 (2018).
[45] E. N. Lorenz, "Deterministic nonperiodic flow," J. Atmos. Sci. **20**, 130–141 (1963).
[46] J. L. Hindmarsh, R. M. Rose, and A. F. Huxley, "A model of neuronal bursting using three coupled first order differential equations," Proc. R. Soc. London Ser. B **221**, 87–102 (1984).
[47] X.-J. Wang, "Genesis of bursting oscillations in the Hindmarsh-Rose model and homoclinicity to a chaotic saddle," Physica D **62**, 263–274 (1993).
[48] E. N. Lorenz, "Predictability—A problem partly solved," in *Predictability of Weather and Climate*, edited by R. Hagedorn and T. Palmer (Cambridge University Press, Cambridge, 2006), pp. 40–58.
[49] G. Datseris, "Dynamicalsystems.jl: A Julia software library for chaos and nonlinear dynamics," J. Open Source Soft. **3**, 598 (2018).
[50] If k does not divide D in $\mathbb{N}$, the measurement spacing between the first and the last system variable is shorter than k.
[51] J. A. Nelder and R. Mead, "A simplex method for function minimization," Comput. J. **7**, 308–313 (1965).
[52] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," Nat. Methods **17**, 261–272 (2020).
[53] For the case $\rho = 0$ and $\alpha = 1$, the internal dynamics of the reservoir system is deactivated and only a static non-linear transformation is performed.[57] One could, therefore, question whether this case should still be regarded as reservoir computing whose constitutive characteristic is to be based on a dynamic process.
[54] L. Jaurigue, "Chaotic attractor reconstruction using small reservoirs—The influence of topology," Mach. Learn.: Sci. Technol. **5**, 035058 (2024).
[55] A. Griffith, A. Pomerance, and D. J. Gauthier, "Forecasting chaotic systems with very low connectivity reservoir computers," Chaos **29**, 123108 (2019).
[56] R. Pyle, N. Jovanovic, D. Subramanian, K. V. Palem, and A. B. Patel, "Domain-driven models yield better predictions at lower cost than reservoir computers in Lorenz systems," Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. **379**, 20200246 (2021).
[57] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. S. Barbosa, "Next generation reservoir computing," Nat. Commun. **12**, 5564 (2021).
[58] A. Karimi and M. Paul, "Extensive chaos in the Lorenz-96 model," Chaos **20**, 043105 (2010).
[59] I. Kottlarz and U. Parlitz, "Ordinal pattern-based complexity analysis of high-dimensional chaotic time series," Chaos **33**, 053105 (2023).
[60] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)* (IEEE, Budapest, 2004), Vol. 2, pp. 985–990.
[61] The single step prediction cannot get worse by reusing time-delayed reservoir states, since delayed reservoir state variables, which increase the normalized error can simply be neglected in the trained superposition.
[62] J. Jaurigue, J. Robertson, A. Hurtado, L. Jaurigue, and K. Lüdge, "Post-processing methods for delay embedding and feature scaling of reservoir computers," Commun. Eng. **4**, 1–13 (2025).
[63] C. G. Langton, "Computation at the edge of chaos: Phase transitions and emergent computation," Physica D **42**, 12–37 (1990).
[64] J. Boedecker, O. Obst, J. T. Lizier, N. M. Mayer, and M. Asada, "Information processing in echo state networks at the edge of chaos," Theory Biosci. **131**, 205–213 (2012).