Reservoir computing with higher-order interactive coupled pendulums

Xueqi Li [1], Michael Small [2,3] and Youming Lei [1,4,*]

[1] *School of Mathematics and Statistics, Northwestern Polytechnical University, Xi'an 710072, China*

[2] *Complex Systems Group, Department of Mathematics and Statistics, The University of Western Australia, Crawley WA 6009, Australia*

[3] *Mineral Resources, CSIRO, Kensington WA 6151, Australia*

[4] *Ministry of Industry and Information Technology Key Laboratory of Dynamics and Control of Complex Systems, Northwestern Polytechnical University, Xi'an 710072, China*

(Received 28 June 2023; accepted 28 November 2023; published 21 December 2023)

The reservoir computing approach utilizes a time series of measurements as input to a high-dimensional dynamical system known as a reservoir. However, the approach relies on sampling a random matrix to define its underlying reservoir layer, which leads to numerous hyperparameters that need to be optimized. Here, we propose a nonlocally coupled pendulum model with higher-order interactions as a novel reservoir, which requires no random underlying matrices and fewer hyperparameters. We use Bayesian optimization to explore the hyperparameter space within a minimal number of iterations and train the coupled pendulums model to reproduce the chaotic attractors, which simplifies complicated hyperparameter optimization. We illustrate the effectiveness of our technique with the Lorenz system and the Hindmarsh-Rose neuronal model, and we calculate the Pearson correlation coefficients between time series and the Hausdorff metrics in the phase space. We demonstrate the contribution of higher-order interactions by analyzing the interaction between different reservoir configurations and prediction performance, as well as computations of the largest Lyapunov exponents. The chimera state is found as the most effective dynamical regime for prediction. The findings, where we present a new reservoir structure, offer potential applications in the design of high-performance modeling of dynamics in physical systems.

DOI: 10.1103/PhysRevE.108.064304

I. INTRODUCTION

Reservoir computing (RC) [1], commonly described as a generalization of echo-state networks [2] or liquid state machines [3], is gaining popularity as a powerful and effective machine learning tool for the reconstruction and even prediction of many complex systems, including low-dimensional chaotic systems [4–6], neuronal models [5,6], and spatiotemporal chaos [4–7]. A total of three weight layers are involved in RC's architecture [8], as shown in Fig. 1. The input weight matrix and a reservoir network matrix are randomly generated and fixed, while only one output weight matrix is amenable to training. For the task of predicting the state variables of a dynamical system, the reservoir computer is utilized in two distinct phases. We define one configuration as the training phase and the other configuration as the prediction phase [8]. RC has been used successfully in tasks such as accurate speech transcription [9], prediction in time series data [6], control of dynamic processes [10], reconstruction of system dynamics [11], and computations on peripheral devices [12].

The technique of reservoir computing [5,13] significantly simplifies the training of a recurrent neural network. However, to maximize performance, one needs to carefully construct the network's topology, position it in the appropriate dynamical regime (typically task dependent), and ensure that the input signal scaling is optimal. These factors generate a list of multiple hyperparameters that must be simultaneously tuned for every benchmark assignment. Hyperparameter optimization is the process of finding the right combination of hyperparameter values to achieve the maximum performance on training data in a reasonable amount of time. Technically, there are two distinct approaches. The first is manual hyperparameter tuning, which may involve many tests, and can be computationally expensive. The second is automatic hyperparameter tuning, which uses existing algorithms to automate the process. The algorithm runs experiments to extract the best set of hyperparameters to achieve the best results. For example, grid search [6] works by trying all possible combinations of parameters in the model, it performs an exhaustive search, which is computationally expensive. A more scalable approach is to use a random search [14], where random combinations of hyperparameter values are used to find the best solution for the built model. However, this may miss important values in the search space, and does not take advantage of the structure of the search space. Alternatively, the Bayesian optimization [15,16] method minimizes the loss function by changing the model parameters, which can find the point of the minimum loss function in fewer steps. This optimization is less susceptible to local minima than gradient descent techniques [17]. The Bayesian optimization [15,16] uses the acquisition function [18] to directly sample areas that are likely to be better than the current

*Corresponding author: leiyouming@nwpu.edu.cn

best observation results, that is, there is both exploitation and exploration here. The Bayesian optimization combines the advantages of both random search and grid search strategies, including higher efficiency, adaptability, effective handling of high-dimensional problems, and robustness to noise and uncertainty [6,14–18].

In general, the reservoir layer is constructed as a network of interacting entities with a random topology. Numerous topological structures have been proposed and studied, including Erdos-Rényi networks [19,20], small-world networks [21,22], as well as cycle linear networks [15]. The performance of reservoir computing for a specific task is optimized by modifying hyperparameters, or large-scale network properties, while constraining others. To successfully serve as a substrate for RC, a dynamical system needs to possess only a few basic properties [23], such as the separation property [24] (different inputs result in distinct outputs), the approximation property [25] (similar inputs result in similar outputs), the fading memory property [26] (information about recent inputs is retained), and the consistency property (outputs are a function of input). Those translate to the requirement that the dynamical system must generate a variety of nonlinear functions pertinent to a given task, and this complexity can be manifested in mechanical systems [5].

Mechanical systems [5] are composed of an array of elements, most of which exhibit nonlinear dynamics, leading to the inherent complexity. However, the complexity can be exploited advantageously to generate the rich nonlinear dynamics required for RC, making them viable options for physical reservoirs [5]. RC with a mechanical system represents a contemporary approach [5] that leverages the inherent dynamics of the physical structure for computational purposes. The reservoir of a mass-spring network [27,28] can be used for pattern generation tasks [28], the structure of a tensegrity robot is designed for developing planetary rovers [1,10,29] as a reservoir, the body of a soft machine [29–31] serves as a computational reservoir, facilitating the processing of control actions and the transmission of computations to remote locations, and an origami structure [32] can function as a dynamic reservoir, possessing the requisite computational capacity to model nonlinear systems. In addition, spin-torque oscillator arrays [33] exhibit optimal reservoir computing performance at the synchronization-disorder boundary. Utilizing spin-wave dynamics, magnetic skyrmion crystals [34] offer a simplified, yet highly effective method for spintronics reservoir computation. Other classifications of high-dimensional systems, including swarms [23], cellular automata [35,36], and coupled phase oscillators [37,38], have also been utilized as reservoirs. Overall, physical implementations of the RC principle that have the potential for rapid and efficient computation [5].

Coupled dynamical networks, which are essential to the development of physical models, provide a comprehensive framework for interpreting phenomena in the fields of computing, information, biology, chemistry, engineering, and the social sciences [39]. Especially, the coupled chain of pendulums serves as a model for various physical processes [40], encompassing coupled Josephson junctions, charge density wave conductors, metal crystal dislocations, and this chain replicates spike and burst behaviors observed in neurons [40]. Currently, no reservoirs use the coupled pendulums model, although the model [20,39,40] has been extensively studied in other settings. Coupled networks draw inspiration from the structure of the brain, which needs to transform from only considering paired interactions to capturing higher-order relationships. Pairwise interactions are insufficient to capture the complex interactions among dynamical elements. Higher-order interactions, which stand out by creation of complex patterns as a result of interactions among multiple bodies, become of utmost significance. Richer dynamics are a general phenomenon emerging from higher-order interactions that has been proven [41]. Incorporating high-order interactions into a coupled pendulum system enables richer dynamical behaviors [40], making it an attractive option for physical reservoirs.

In this work, we show that a system of coupled pendulums with higher-order interactions can serve as a reservoir. The coupled pendulum model is a complex system made up of numerous single oscillators interacting. The power of this model lies in the fact that we consider higher-order interactions. We introduce higher-order interactions through one node being affected by a nonlinear combination of the states of several other nodes. The RC structure proposed in this work differs from previous structures in that it is not randomly generated, and we show that a rule structure with high-order interactions can also act as an observer in the reservoir.

The rest of the paper is organized as follows. In Sec. II, we outline the reservoir computing configuration utilizing coupled pendulums with higher-order interactions as a reservoir. We describe the Bayesian optimization approach to choose the best hyperparameters of the reservoir. In Sec. III, we illustrate the ability of our reservoir to replicate the (long-term) attractor dynamics using the well-known Lorenz-63 chaotic system and Hindmarsh-Rose neuronal model. In Sec. IV, we evaluate the results of our simulations with two metrics, the Pearson correlation coefficient and Hausdorff metrics, and we analyze the influence of reservoir sizes and its behavior on our reservoir. We compare the task performance without the terms of higher-order interactions and no nonlinearity when injecting inputs. To do this, we estimate the largest Lyapunov exponent to describe the contribution of our reservoir with the higher-order interactions and nonlinearity when inputting information, and we find the chimera state to be the most effective dynamical regime for enhanced prediction. Finally, we conclude with a summary in Sec. V.

II. RESERVOIR COMPUTING CONFIGURATION

Reservoir computing is an approach to transforming one time-varying signal (the input to reservoir computing) into another time-varying signal (the output of reservoir computing) by using the dynamics of an internal system known as the reservoir—the reservoir acts as a dynamic nonlinear filter with memory. We define our reservoir as the coupled nonlinear oscillator network where each isolated oscillator is forced, damped, and subjected to a periodic substrate po-

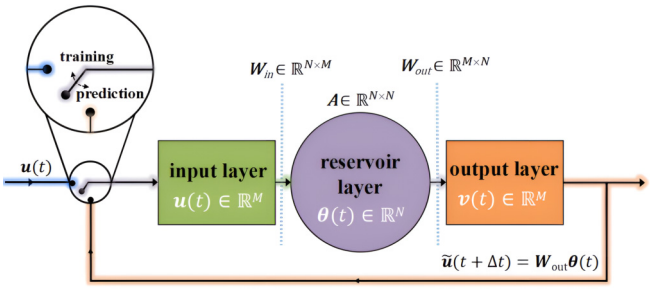

FIG. 1. Diagrammatic representation of the reservoir computing training and prediction structures. In the form of reservoir computing, there are three layers: the input layer, the reservoir layer, and the output layer. Throughout the prediction process, this structure will develop into a closed loop.

tential [37], and the whole network includes higher-order interactions [40].

A. Coupled pendulums with higher-order interactions

Regarding the most basic simplicial complexes of higher order in a network, we study not only pairwise connections, such as links (one-simplex), but also nonpairwise connections, such as the three-body interaction in the network known as triangles (two-simplex) [40]. We provide an example to facilitate intuitive comprehension of the network topology. Figure 2 depicts the network of oscillator interactions when $N = 8$, where the red and green lines represent the interactions between two and three oscillators, respectively. It is visible how pairwise interactions progress into higher-order interactions [40].

The mathematical form of coupled pendulums with higher-order interactions is given by

$$\begin{aligned}\ddot{\theta}_i(t) &= \gamma\dot{\theta}_i(t) + \omega_i - \sin\theta_i(t) \\ &+ \frac{k_1}{N}\sum_{j=1}^{N} A_{ij}\sin[\theta_j(t) - \theta_i(t)] \\ &+ \frac{k_2}{2N}\sum_{j=1}^{N}\sum_{k=1}^{N} B_{ijk}\sin[\theta_j(t) + \theta_k(t) - 2\theta_i(t)], \end{aligned} \quad (1)$$

where $i = 1, 2, \ldots, N$, θ_i is the angular position of the ith oscillator agent, and N is the number of oscillator agents. $\dot{\theta}_i$ and $\ddot{\theta}_i$ are the angular velocity and angular acceleration

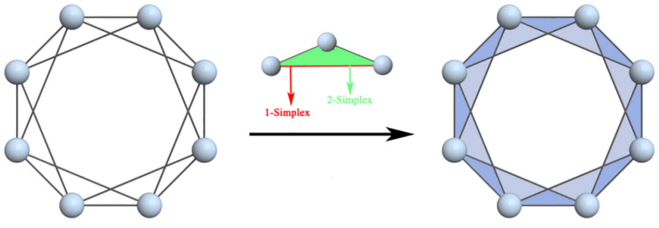

FIG. 2. Schematic of pairwise and three-body interactions in a network with eight nonlocally coupled nodes. Examples of one-simplex (red line) and two-simplex (green triangle).

of the ith unit in the phase space, respectively. We set the underdamping coefficient $\gamma = -0.32$ [40], a constant force $\omega_i = \omega = 0.7155$ [40], and periodic boundary conditions with $\theta_0 = \theta_N$ and $\theta_1 = \theta_{N+1}$. The matrix $\mathbf{A}$ denotes the pairwise adjacency matrix of the reservoir layer, and the tensor $\mathbf{B}$ is the nonpairwise adjacency tensor of the reservoir layer. The parameters k_1 and k_2 are the coupling strengths of low-order (pairwise) and higher-order (nonpairwise), respectively.

B. Higher-order interactive coupled pendulums as a reservoir

In general, the reservoir is a complex dynamical system. Here, the coupled pendulums model is a typical nonlinear dynamic system with complex dynamic behavior, especially when considering high-order interactions. We seek to utilize this model as a reservoir for a reservoir computing system.

1. Internal reservoir construction

We need to determine how the signal can be injected into the reservoir, the external fields or mechanical forces operating on oscillators are fixed by the input signal on which computation is to be performed. Since the oscillator agents are driven by external fields or other mechanical forces, their motion is determined not only by the coupling between all oscillator agents but also by the location and recent history of the external fields or forces operating, and thus by the signal. Therefore, we add an additional force to Eq. (1) and rewrite it as:

$$\begin{aligned}\ddot{\theta}_i(t) &= (1-\varepsilon)[\gamma\dot{\theta}_i(t) + \omega_i - \sin\theta_t(t)] \\ &+ \varepsilon\Bigg\{\frac{k_1}{N}\sum_{j=1}^{N} A_{ij}\sin[\theta_j(t) - \theta_i(t)] \\ &+ \frac{k_2}{2N}\sum_{j=1}^{N}\sum_{k=1}^{N} B_{ijk}\sin[\theta_j(t) + \theta_k(t) - 2\theta_i(t)] \\ &+ \alpha\tanh[W_{in}u(t) + \boldsymbol{\beta}]_i\Bigg\}, \end{aligned} \quad (2)$$

where the M-dimensional input $u(t)$ is fed into the N reservoir nodes via a linear input weight matrix denoted by $W_{in} \in \mathbb{R}^{N\times M}$. The parameter $0 < \varepsilon \leqslant 1$ represents the leakage rate, modulating the reservoir's temporal dynamics by balancing the influence between past and present inputs. As $\varepsilon \to 0$, the evolution of the reservoir slows down [19]. The parameter α denotes the coupling strength of the input data, and β is the input bias vector [28], whose elements are randomly chosen at an interval $[-0.989, 0.989]$. Each dimension of the vector $\boldsymbol{u}(t) = [u_1(t), u_2(t), \ldots, u_M(t)]^T$ represents the input data[1] and the vector $\boldsymbol{\theta}(t) = [\theta_1(t), \theta_2(t), \ldots, \theta_N(t)]^T$ signifies the phase of a single network node. The $\tanh(\cdots)$ is a typical

[1]We perform preprocessing on all the components of $\boldsymbol{u}(t)$. After normalization, the value of each element is in the range of $[-1, 1]$. The purpose of this normalization process is to map the data to a fixed range, eliminate dimension effects, and avoid numerical instability, thereby improving the performance of the reservoir.

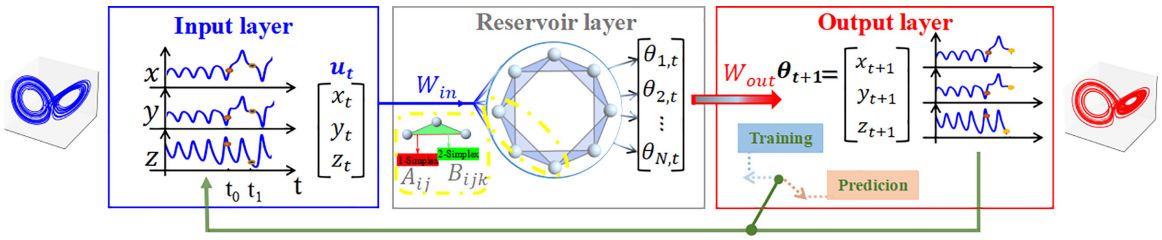

FIG. 3. Reservoir computing with higher-order interactions pendulums reproduces the original attractor obtained using regular networks. The framework of the configuration of our coupled pendulums reservoir in the corresponding training and prediction phases. A reservoir computer consists of three parts, an input layer, a reservoir layer, and an output layer. The matrices $\boldsymbol{W}_{\mathbf{in}}$ and $\boldsymbol{W}_{\mathbf{out}}$ denote the input-to-reservoir and reservoir-to-output weights, respectively.

activation function of the reservoir nodes that operates componentwise on vectors [15]: $\tanh(x)_i = \tanh(x_i)$.

2. Training step

Figure 3 depicts our reservoir configuration using the Lorenz-63 system [11] as input. A normalized input $u(t) = [x(t), y(t), z(t)]^T$ signal is coupled to the reservoir via $\mathbf{W}_{in}$ and combined with the reservoir state $\boldsymbol{\theta}(t) = [\theta_1(t), \theta_2(t), \ldots, \theta_N(t)]^T$ to produce $\boldsymbol{\theta}(t + \Delta t) = [\theta_1(t + \Delta t), \theta_2(t + \Delta t), \ldots, \theta_N(t + \Delta t)]^T$. The reservoir state vector is given by Eq. (2), and the reservoir nodes contain both pairwise and higher-order couplings, as depicted in Fig. 3 by the red lines and green triangles, respectively. The final trained output $\boldsymbol{v}(t)$ is obtained by combining the reservoir states $\boldsymbol{\theta}(t)$ with the trained readout weight matrix $\boldsymbol{W}_{\text{out}}$:

$$\boldsymbol{v}(t) = \boldsymbol{W}_{\text{out}}\boldsymbol{\theta}(t). \tag{3}$$

The key aspect of reducing the complexity that makes reservoir computing consequently simple and computationally efficient is the linearization of the training-to-output process [6]. The least-squares solution [6,23] is readily handled by matrix computations. We choose $\boldsymbol{W}_{\text{out}}$ by minimizing the difference between the true $\hat{\boldsymbol{v}}(t)$ and the trained output $\boldsymbol{v}(t)$. To do this, a loss function is designed as:

$$\sum_{t=0}^{T} \|\hat{\boldsymbol{v}}(t) - \boldsymbol{W}_{\text{out}}\boldsymbol{\theta}(t)\|^2 + \phi\|\boldsymbol{W}_{\text{out}}\|^2. \tag{4}$$

In a bid to prevent overfitting, the second term in the error function modifies ordinary linear least-squares regression; this modification is often referred to as ridge regression or Tikhonov regularization [6]. We present results using the ridge regularization parameter $\phi = 10^{-7}$. If the training phase is successful for $0 \leqslant t \leqslant T$, the readout of the reservoir output should provide a decent approximation $\boldsymbol{W}^*_{\text{out}}\boldsymbol{\theta}(t)$ of the unmeasured quantity $\hat{\boldsymbol{v}}(t)$ for $t > T$. Equation (3) can be reformulated as:

$$\hat{\boldsymbol{v}}(t) = \boldsymbol{W}^*_{\text{out}}\boldsymbol{\theta}(t), \tag{5}$$

where $\boldsymbol{W}^*_{\text{out}}$ denotes the solutions for the minimizers of Eq. (4), as follows,

$$\boldsymbol{W}^*_{\text{out}} = \delta\boldsymbol{U}\delta\boldsymbol{R}^T(\delta\boldsymbol{R}\delta\boldsymbol{R}^T + \phi\boldsymbol{I})^{-1}, \tag{6}$$

where $\delta\boldsymbol{U} \in \mathbb{R}^{M\times T}$ is the matrix whose kth column is $\boldsymbol{u}(k\Delta t)$, $\delta\boldsymbol{R} \in \mathbb{R}^{N\times T}$ is the matrix whose kth column is $\boldsymbol{\theta}(k\Delta t)$, and $\boldsymbol{I} \in \mathbb{R}^{M\times T}$ is the identity matrix. Now once $\boldsymbol{W}^*_{\text{out}}$ is determined, the reservoir computer is trained.

In the prediction phase, we substitute the output for the input to the reservoir, and the reservoir system works autonomously based on the equation

$$\begin{aligned}\ddot{\theta}_i(t) = (1-\varepsilon)[\gamma\dot{\theta}_i(t) + \omega_i - \sin\theta_t(t)] \\ + \varepsilon\left\{\frac{k_1}{N}\sum_{j=1}^{N} A_{ij}\sin[\theta_j(t) - \theta_i(t)]\right. \\ + \frac{k_2}{2N}\sum_{j=1}^{N}\sum_{k=1}^{N} B_{ijk}\sin[\theta_j(t) + \theta_k(t) - 2\theta_i(t)] \\ \left. + \alpha\tanh[\boldsymbol{W}_{in}\boldsymbol{W}^*_{out}\boldsymbol{\theta}(t) + \beta]_i\right\},\end{aligned} \tag{7}$$

which is independent of the input $\boldsymbol{u}(t)$ and runs autonomously.

3. Bayesian optimization on reservoir computing

To maximize reservoir performance, we aim to discover the global optimal hyperparameters in the fewest possible steps [42]. Here, we apply Bayesian optimization to a situation in experimental large-scale reservoir computing. The implementation of the Bayesian optimization [16,18] requires the definition of surrogate and acquisition functions [18,19]. The surrogate function is used to fit the relationship between hyperparameter and performance, and the acquisition function is used to determine which hyperparameter to try in the next iteration.

Bayesian optimization is an iterative process, as shown in Algorithm 1. First, randomly try a set of points (hyperparameters) to get the corresponding performance, then use the surrogate function to fit the results of Step 1, and determine the next time according to the acquisition function to iterate over points to try until the stop condition is satisfied.

Algorithm 1: Bayesian optimization to find the optimal hyperparameters

Presteps: Set the search space for hyperparameter and acquisition function
Select surrogate function
1: Randomly try some points (hyperparameters) in the parameter space to get their performance (call it Box)
2: Use the surrogate function to fit the results of step 1
3: Repeat
4: Add an additional set of hyperparameters to Box based on acquisition function
5: Reevaluate the surrogate function
6: **Until**
7: Maximum surrogate function remains unchanged
8: Or reached a preset number of iterations
Output: Best accuracy and a set of hyperparameters

(1) Acquisition function: setting a specific required objective function. During the training process of the algorithm, the goal is to make the objective function value lower. We use root mean square error (*RMSE*) over a certain period of time as the objective function and $\max(0, RMSE_{\text{best}} - RMSE)$ as an acquisition function.

(2) Surrogate function: performing the Bayesian hyperparameter optimization process. The objective function is minimized by the surrogate function after iterating through each model and the search space. There are four different optimization functions provided:[2]

(1) dummy minimize: using a random search within the given boundaries via uniform sampling;

(2) forest minimize: using decision trees for sequential optimization;

(3) gbrt minimize: using gradient boosted trees for sequential optimization; and

(4) gp minimize: using Gaussian processes for Bayesian optimization.

In addition, the objective function accepts a vector of reservoir hyperparameters and returns a real value. The smaller the real value, the better the performance of the reservoir. The search space refers to the possible value range of the reservoir hyperparameters. Usually, the search space is a hypercube, and we choose the hyperparameter ranges in order to explore the space without having to run the algorithm for an unreasonably lengthy time, as well as to include the values that existing heuristics would pick. Based on this approach, we learn through numerical simulations that our reservoir computing can be used to predict the time series and reproduce climate in the chaotic system.

[2]The SCIKIT-OPTIMIZE package offers a variety of surrogate functions in PYTHON. The following PYTHON version and packages are necessary for SCIKIT-OPTIMIZE. PYTHON >= 3.6, NUMPY (>= 1.13.3), SCIPY (>= 0.19.1), JOBLIB (>= 0.11), SCIKIT-LEARN >= 0.20, MATPLOTLIB >= 2.0.0, and then import crucial packages including SCIKIT-OPTIMIZE.

TABLE I. The range of hyperparameters sought via Bayesian optimization.

Hyperparameters	Min	Max
ε	0.01	0.90
γ	−2.30	−0.01
α	0.01	10.00

III. NUMERICAL RESULTS

In the following section, we show how our technique can accurately predict the short-term state and long-term dynamical behavior (the so-called climate) of two distinct systems: Lorenz-63 chaotic system, which serves as an ideal and well-known benchmark for evaluating the computational of RC, and the Hindmarsh-Rose neural model, a typical representative of a multiscale model. The choice of these systems as benchmarks is aimed to elucidate the predictive capabilities of our technique.

A. Lorenz-63 chaotic system

According to Lorenz's chaotic system of 1963 [43],

$$\begin{aligned} \dot{x} &= 10(y - x) \\ \dot{y} &= x(28 - z) - y \\ \dot{z} &= xy - \frac{8}{3}z. \end{aligned} \tag{8}$$

Based on Bayesian optimization on reservoir computing, the following are steps of our reservoir computer's Bayesian optimization for the Lorenz-63 chaotic system:

(1) *Defining the objective function.* Consider RMSE as an evaluation indicator and minimizing it as the objective function;

(2) *Defining the search space.* Select the hyperparameter space includes three hyperparameters: ε, γ, and α, as shown in Table I.;

(3) *Initializing the optimizer.* Choose the Bayesian optimizer and initializing the optimizer according to the search space;

(4) *Performing optimization.* Use an optimizer to iteratively optimize the objective function until a certain stopping condition is reached, such as reaching the maximum number of iterations or achieving convergence of the objective function;

(5) *Obtaining optimal parameters.* Obtain the final vector of reservoir computing hyperparameters from the optimization is the optimal solution of the objective function.

The optimizer builds a reservoir with the selected hyperparameters for each iteration, trains it using the steps outlined in Algorithm 1, and evaluates its performance using $\max(0,\ RMSE_{\text{best}} - RMSE)$. It selects a new set of hyperparameters to test based on this measurement that may be closer to the desired values. Before producing an optimized reservoir, we restrict 200 iterations to a maximum of reservoir performance. Once optimized, we evaluate the convergence results of different surrogate functions and see how our model improves its optimal performance in each iteration. Four

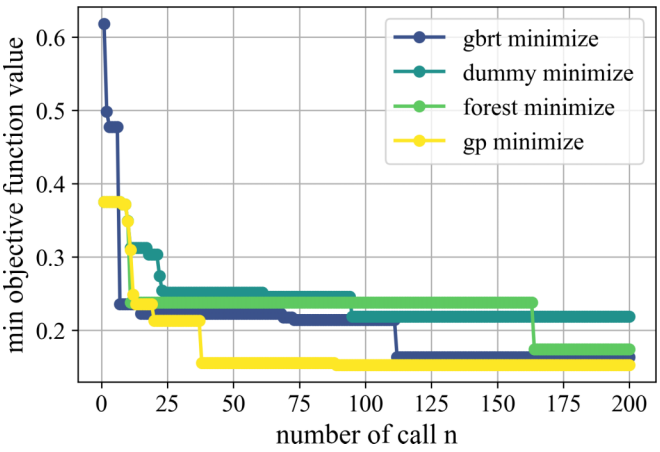

FIG. 4. Convergence results with four different surrogate functions to see how the best performance of the model can be improved in each iteration. The minimum *RMSE* value found (y axis) as a function of the number of iterations (x axis) executed so far.

different surrogate functions are gbrt, dummy, forest, and gp minimize approaches, as shown in Fig. 4. The four approaches converge as the number of iterations increases, especially the gp minimize function, which has the fastest convergence speed and is the optimal convergence value. Thus, gp minimize is used as the surrogate function in the following simulations.

Figure 5 shows the evolution of the gp minimize surrogate function. For each set of hyperparameters, a histogram is obtained. For each set of hyperparameters, the scatter plot of sampled values is represented by color. A two-dimensional scatter plot of every point is displayed in the lower triangle.

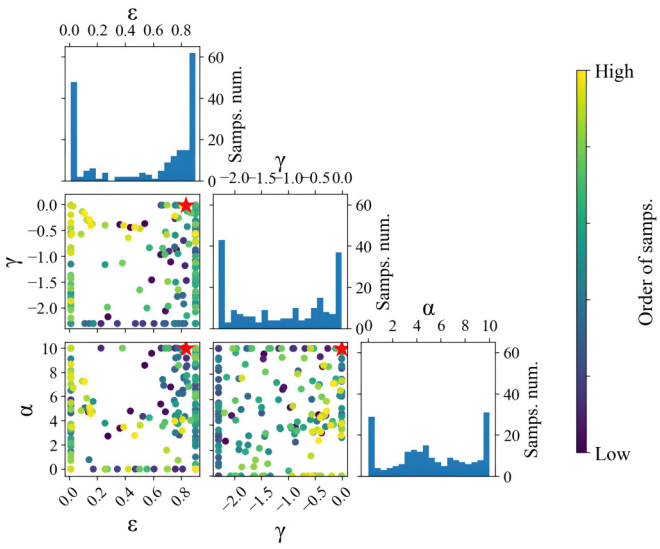

FIG. 5. Visualize the sampling order of the points used for optimization on the three-by-three grid. The diagonal plots are histograms that illustrate the distribution of samples for each search-space dimension, and the plots below the diagonal are scatter plots of the samples for all combinations of search-space dimensions. The order of sample evaluations is transmitted in the color of each point. A red star denotes the best-found hyperparameters.

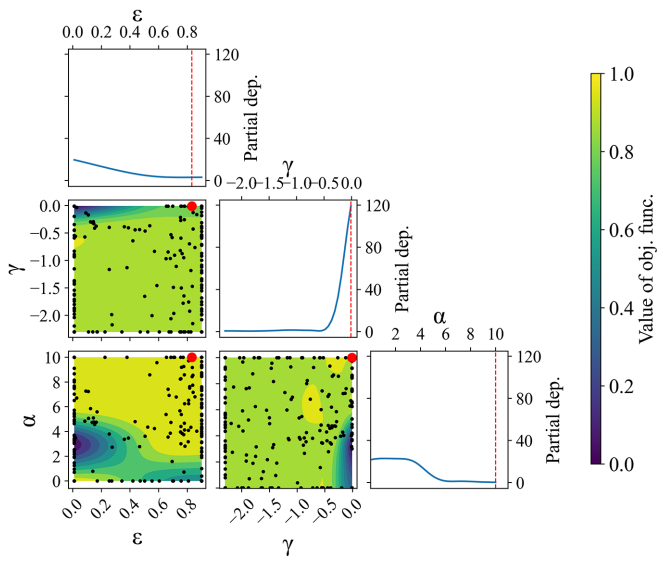

FIG. 6. The distribution of the objective function value when each hyperparameter changes separately. The graph on the diagonal shows the degree of influence of the hyperparameter on the objective function. The horizontal and vertical axes represent the value range of the hyperparameter, the color represents the corresponding objective function value, and the brighter color represents the higher objective function value.

The order in which points are evaluated is encoded in the color of each point. Darker (purple) colors correspond to earlier samples, and lighter (yellow) colors correspond to later samples. A red star point shows the location of the minimum found by the optimization process. The points begin to gather around the true minimum position. Each of the dimensions' histograms is displayed on the diagonal. The histogram shows that targets are evaluated more frequently near one of the three minimum values.

Partial dependence plots were proposed by Friedman [44] as a method for interpreting the importance of input features used in gradient boosting machines. The advantage lies in the intuition of fractional sensitivities associated with hyperparameters. It is possible to decide which parts of the space require a more fine-grained search, and to check out which hyperparameters barely affect the score, and may be removed from the search. The plot on the diagonal is the distribution graph of the objective function value when each hyperparameter is used as an axis alone in Fig. 6. We can visualize the one-dimensional partial dependence of the surrogate model. If the shape in the graph appears as a flat line, the objective function is not affected by this hyperparameter. If the plot on the diagonal shows a certain shape, it indicates that the objective function is very sensitive to the value of the hyperparameter or has a very high optimal value in a certain range of the hyperparameter. The horizontal and vertical axes in the lower triangle represent the value range of each hyperparameter, respectively. The color represents the corresponding objective function value, the brighter the color, the higher the objective function value.

Figures 4–6 indicate that the optimal hyperparameters are determined to be $\epsilon = 0.81$, $\gamma = -0.01$, and $\alpha = 9.42$,

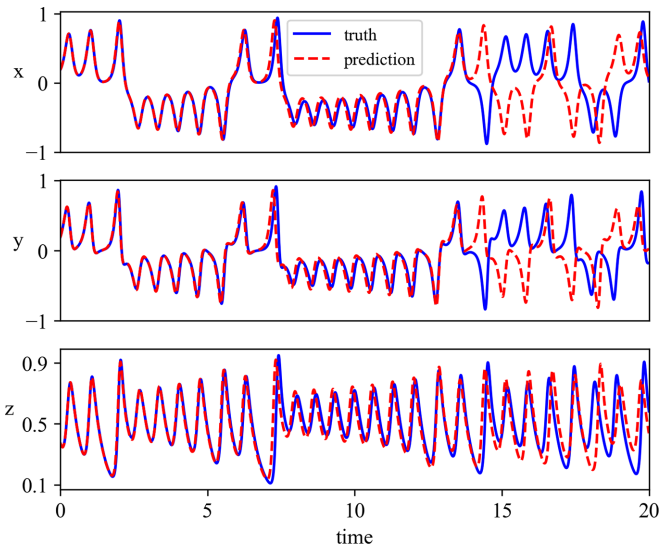

FIG. 7. The predicted state (red) of the optimized reservoir and the actual trajectories (blue) of the Lorenz-63 system for $0 < t \leqslant 20$. The values of the hyperparameters are $\varepsilon = 0.81$, $\gamma = -0.01$, and $\alpha = 9.42$.

respectively. We produce a prediction of the Lorenz-63 chaotic system using the best performing optimized hyperparameters RC and use these predictions to show the time series and the attractor.

Figure 7 depicts the predicted time series for $0 < t \leqslant 20$ for the trained and optimized reservoir. The reservoir (R1 system) generates good short-term forecasts for approximately 14 time units. It then tends to deviate from the actual Lorenz trajectories. The long-term dynamics resemble the features of the original Lorenz system in Fig. 8(a), and the attractor is reproduced. A more detailed evaluation is necessary to verify the accuracy of the climate. We plot the return map of successive maxima of $z(t)$ in Fig. 8(b). Initially, we obtain $z(t)$ for $0 < t \leqslant 50$ in both the actual and predicted time series. Then, we identify all local maxima of the actual and predicted $z(t)$ in time order and label them $[z_1, z_2, \ldots, z_m]$. Following which, consecutive pairs of these maxima $[z_i^{\max}, z_{i+1}^{\max}]$ for $i = 1, 2, \ldots, m-1$ are plotted as points in Fig. 8(b), and the blue (truth) dots remain covered with red (prediction) dots, the

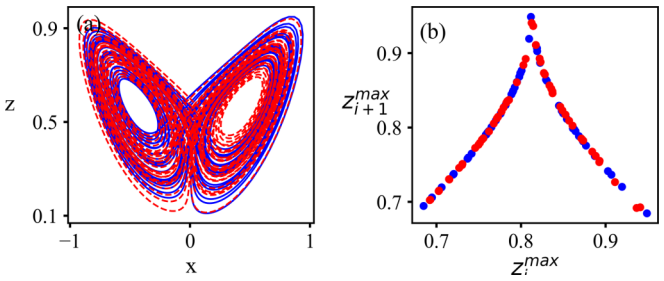

FIG. 8. (a) Motion in phase space for the actual (blue) and predicted (red) trajectories for $0 < t \leqslant 50$. (b) Poincaré return map of successive local maxima of $z(t)$ for the actual (blue) and predicted (red) trajectories for $0 < t \leqslant 50$. Both (a) and (b) utilize the same conditions as Fig. 7.

TABLE II. Best reservoir computers using hyperparameter settings of various reservoir sizes following 200 iterations of the Bayesian optimization algorithm using the Lorenz-63 chaotic system as input. The algorithm-selected hyperparameters are displayed on the right.

parameters	Lorenz-63 chaotic system		
	ε	γ	α
$N = 100$	0.89	−1.44	9.06
$N = 200$	0.76	−0.01	9.14
$N = 300$	0.85	−0.01	7.40
$N = 400$	0.03	−1.48	1.78
$N = 500$	0.91	−0.01	5.96
$N = 600$	0.90	−0.01	6.92
$N = 700$	0.81	−0.01	9.33
$N = 800$	0.81	−0.01	9.42
$N = 900$	0.90	−0.09	5.96
$N = 1000$	0.90	−0.01	8.40

local maximal value for the truth and the prediction are at the same point. Consequently, the reservoir appears to reproduce the long-term climate of the attractor.

As shown in Table II, we use Bayesian optimization to find the optimal hyperparameter with different number of oscillators. It can be seen from the table that no matter the number of the oscillators, the optimal hyperparameter has preferential characteristics, and there is always a part of the number that is always selected as the optimal value. The first hyperparameter ε frequently appears around 0.8 across various reservoir sizes. The most common value of the second hyperparameter γ is approximately -0.01. Interestingly, the optimal values of the first two parameters occur near the search boundary. These tendencies, as identified by the Bayesian optimization approach, suggest the existence of a possible underlying structure in the space and emphasize the significance of these hyperparameters in achieving optimal performance.

B. Hindmarsh-Rose neural model

Neurons can exhibit chaotic behavior, characterized by irregular quantities of spikes per burst and irregular time intervals between bursts. These behaviors can be imitated by the Hindmarsh-Rose (HR) neuronal model. The system consists of a set of three first-order differential equations with dimensionless variables representing the voltage variable x, the spiking variable y, and the bursting variable z. The state evolution of the HR neural model is given by [45]

$$\begin{aligned} \dot{x} &= y - x^3 + 3x^2 - z + 3.25 \\ \dot{y} &= 1 - 5x^2 - y \\ \dot{z} &= 0.005[4(x + 1.6) - z]. \end{aligned} \tag{9}$$

The model Eq. (9) can be segregated into a fast subsystem made up of the first two equations and a slow subsystem made up of the third equation. For the neuron with parameters fixed, the neuronal dynamic displays chaotic behavior in Fig. 9(a), with irregular numbers (four or five) of spikes per burst as well as irregular time intervals between bursts in Fig. 9(b). Since

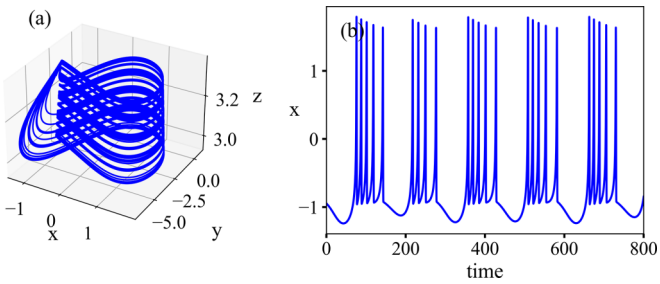

FIG. 9. (a) Chaotic motion of Hindmarsh-Rose neural model. (b) The time series of variable x in the Hindmarsh-Rose neural model.

the HR neuron model is a fast-slow system, the reservoir layer is required to learn both the dynamical characteristics of the slow variable and the fast variable after training, that is, the reservoir has to learn the timescale of the input (a fast-slow system), which increases the difficulty of prediction.

In this case, we use a vector $\boldsymbol{u}(t) = [x(t), y(t), z(t)]^T$ of HR neural model states for which we extract 60000 training data points as input signal for training the reservoir with time step 0.01, and the following 35000 data points as the desired output. The input and output data points are carried on data preprocessing. We show the results of training a reservoir computer to predict the future behavior of the HR neural model in Fig. 10. In the short term, the reservoir generates accurate predictions, but as time evolves, it diverges from the actual neuronal trajectory. After the predicted trajectory starts to diverge from the actual trajectory at about time 249, as shown in Fig. 11, the two trajectories continue to exhibit the same qualitative behavior.

The long-term dynamics resemble the characteristics of the Hindmarsh-Rose neural model illustrated in Fig. 11. We obtain $x(t)$ for a period $0 < t \leqslant 350$ for both the actual and

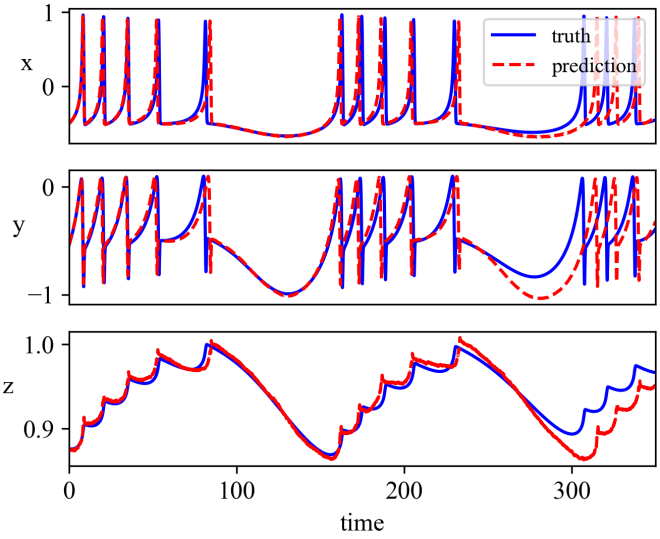

FIG. 10. The predicted state (red) of the optimized reservoir and the actual trajectories (blue) of the Hindmarsh-Rose neural model for $0 < t \leqslant 350$. The value of the hyperparameters are $\varepsilon = 0.90$, $\gamma = -0.01$, and $\alpha = 8.64$.

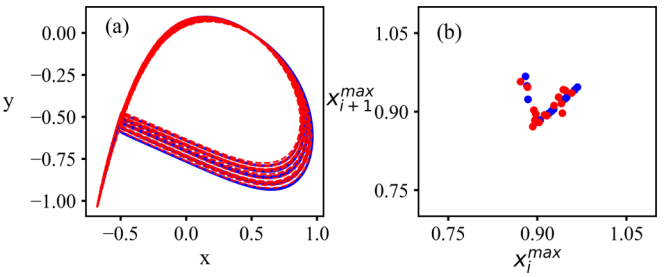

FIG. 11. (a) Motion in phase space for the actual (blue) and predicted (red) trajectories for $0 < t \leqslant 350$. (b) Poincaré return map of successive local maxima of $x(t)$ for the actual (blue) and predicted (red) trajectories for $0 < t \leqslant 350$. Both (a) and (b) utilize the same conditions as Fig. 10.

predicted time series. Then, all local maxima of the actual and predicted $x(t)$ are identified in time order and labeled as $[x_1, x_2, \ldots, x_m]$. Consequently, successive pairs of these maxima $[x_i^{\max}, x_{i+1}^{\max}]$ for $i = 1, 2, \ldots, m-1$ are depicted as points in Fig. 11(a), and the blue (truth) points remain covered with red (prediction) points in Fig. 11(b), indicating that the local maximal value for the truth and the prediction are similar. Therefore, the reservoir appears to replicate the long-term climate of the attractor.

Next, we consider a network of reservoirs with different sizes N ranging from 100–1000. The input weight matrix $\boldsymbol{W}_{in} \in \mathbb{R}^{N\times M}$ elements are obtained from an uniform distribution with values between -0.98 and 0.98, with a regularization factor of 10^{-7}. Refer to Table I for the selection of hyperparameters, and the iterations of the Bayesian optimization algorithm is 200. The above settings are the same as Lorenz. The advantage is that we do not need to change the structure of the reservoir itself for specific tasks. The selection of hyperparameters appears with various reservoir sizes, as summarized in Table III. The three hyperparameters ε, γ, and α, vary with different reservoir sizes without a noticeable trend.

TABLE III. Best reservoir computers using hyperparameter settings of various reservoir sizes following 200 iterations of the Bayesian optimization algorithm using the Hindmarsh-Rose neural model as input. The algorithm-selected hyperparameters are displayed on the right.

parameters	Hindmarsh-Rose neural model		
	ε	γ	α
$N = 100$	0.01	−1.26	10.00
$N = 200$	0.01	−2.30	8.54
$N = 300$	0.14	−0.31	4.74
$N = 400$	0.01	−2.30	0.96
$N = 500$	0.90	−0.01	8.46
$N = 600$	0.89	−0.02	9.78
$N = 700$	0.90	−0.01	10.00
$N = 800$	0.60	−0.60	7.93
$N = 900$	0.85	−0.87	6.13
$N = 1000$	0.90	−0.01	10.00

IV. EVALUATION

A. Tool for evaluating production ability of reservoir

To calculate our reservoir's capacity, we utilize the Hausdroff metric and the Pearson correlation coefficient. Assuming predicted state $\boldsymbol{v}(t) = [x_v(t), y_v(t), z_v(t)]^T$ in the prediction phase $T \leqslant t \leqslant T_1$, the corresponding true state at time t is $\boldsymbol{u}(t) = [x_u(t), y_u(t), z_u(t)]^T$.

The Pearson correlation coefficient is defined as the quotient of covariance and standard deviation between two variables, and we use the variable $x(t)$ to calculate its Pearson correlation coefficient,

$$\rho_{X_v,X_u} = \frac{\mathrm{cov}(X_v, X_u)}{\sigma_{X_v}\sigma_{X_u}} = \frac{E[(X_v - \mu_{X_v})(X_u - \mu_{X_u})]}{\sigma_{X_v}\sigma_{X_u}} = \frac{\sum_{t=T}^{T_1}(X_v(t) - \bar{X}_v(t))(X_u(t) - \bar{X}_u(t))}{\sqrt{\sum_{t=T}^{T_1}(X_v(t) - \bar{X}_v(t))^2}\sqrt{\sum_{t=T}^{T_1}(X_u(t) - \bar{X}_u(t))^2}}, \tag{10}$$

where X_v is a time series of x_v, and X_u is a time series of x_u. The Pearson correlation coefficient ρ_{X_v,X_u} varies from -1 to 1. The values of $\rho_{X_v,X_u} = 1$ or $\rho_{X_v,X_u} = -1$ mean that X_v and X_u are well described by linear equations, and all data points fall nicely on a straight line, and $\rho_{X_v,X_u} = 0$ means that there is no linear relationship between the two variables. For the intuitiveness of the calculation results, let $r_\rho = |\rho_{X_v,X_u}|$, r_ρ is greater than 0 and less than or equal to 1, and the larger r_ρ is, the more correlated the two variables are.

The Hausdorff metric is a measure that describes the similarity between two sets of point sets, and it is defined as:

$$d(X_v, X_u) = \max\left\{\sup_{x_v \in X_v}\inf_{x_u \in X_u} d(x_v, x_u), \sup_{x_u \in X_u}\inf_{x_v \in X_v} d(x_v, x_u)\right\}, \tag{11}$$

where sup and inf represent the least upper bound and greatest lower bound, respectively. The quantity $d(X_v, X_u)$ is the bidirectional distance between the prediction set X_v and truth set X_u. The smaller $d(X_v, X_u)$, the more accurate the prediction.

B. Influence of reservoir size on RC capability

Investigating the effect of various reservoir sizes on computational capabilities may be of interest: while the memory capacity increases with the number of reservoir units, the prediction of time series requires only a finite amount of memory. The Pearson correlation coefficient r_ρ and the Hausdorff metric $d(X_v, X_u)$ are plotted against the total reservoir size in Figs. 12(a) and 12(b), respectively. The predictive performance shows an upward trend as the size of the reservoir increases, until the size of the reservoir equals 800. Although the size of the reservoir continues increase to 1000, it does not affect the prediction. However, there is no improvement in predictive accuracy when using the same training data set as the reservoir size exceeds 1000 in Fig. 12. An explanation could be the advent of overfitting as a result of a reservoir that is disproportionately large to the available training data. In the field of RC, the complexity of the dynamics it can capture increases with reservoir size. Without a proportional increase in the training data, this could result in the model memorizing the training data rather than generalizing, which is a sign of overfitting. Therefore, our reservoir only requires a small storage capacity for time series prediction.

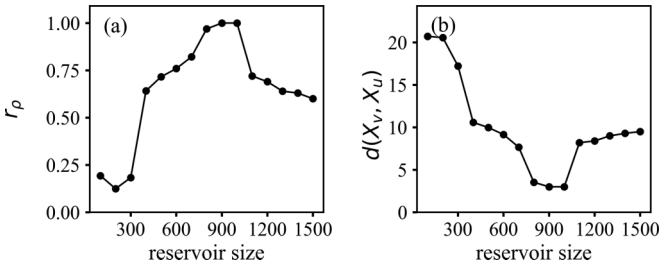

FIG. 12. Our reservoir requires only a modest amount of storage space for time series forecasting. (a) Variation of the Pearson correlation coefficient r_ρ as a function of reservoir size. (b) Variation of the Hausdorff metric $d(X_v, X_u)$ as a function of reservoir size.

C. Influence of reservoir behavior on RC capability

Coupled pendulums with higher-order interactions have been shown to exhibit numerous types of behaviors, from coherent motion to chimera states and even incoherent motion. The purpose of this investigation is to determine what types of collective behavior, if any, are best able to listen the input signals. By analyzing the global order parameter R [38], the collective behaviors observed in this study can be classified into two types: non-chimera and chimera states with spatial features. The measurement R [38] does not perfectly depict the complex dynamics of the coupled pendulums with higher-order interactions as it is a global metric for stationary state, however, for the purposes of this part of the exploration, it is an adequate metric.

The relationship between the reservoir's behavior and the task performance of RC is illustrated in Figs. 13 and 14. With hyperparameters $\varepsilon = 0.10$, $\gamma = -2.30$, and $\alpha = 0$, the RC dynamic is coherent, as depicted in Fig. 13(a), and the snapshot confirms that all oscillators operate in synchronization in Fig. 13(b), which represents a unique coherent state and its global order parameter $R = 1$. Under hyperparameters $\varepsilon = 0.81$, $\gamma = -0.01$, and $\alpha = 0$, the emergence of a chimera state is observed, as reflected in both the spatiotemporal plot Fig. 13(c) and the snapshot representation where the green and blue nodes represent an incoherent state, while the red nodes indicate a coherent one in Fig. 13(d). A detailed examination of the first 50 oscillators showcases an amplified coherent segment, indicating the presence of a multiheaded stationary chimera state, underscoring the intricate coexistence of coherent and incoherent dynamics in the system. Moreover, the global order parameter R for this configuration is 0.67, which is consistent with the established criteria [38] for a chimera state. For $\varepsilon = 0.16$, $\gamma = -0.01$, and $\alpha = 0$, the distinctiveness of this state is characterized by the absence of any synchronized regions in Fig. 13(f), with oscillators demonstrating an entirely desynchronized spatial behavior across the entire system in Fig. 13(e), which is a hallmark of the incoherent state.

We now compare results for three simulations using reservoir configurations with coherent, chimera, and incoherent

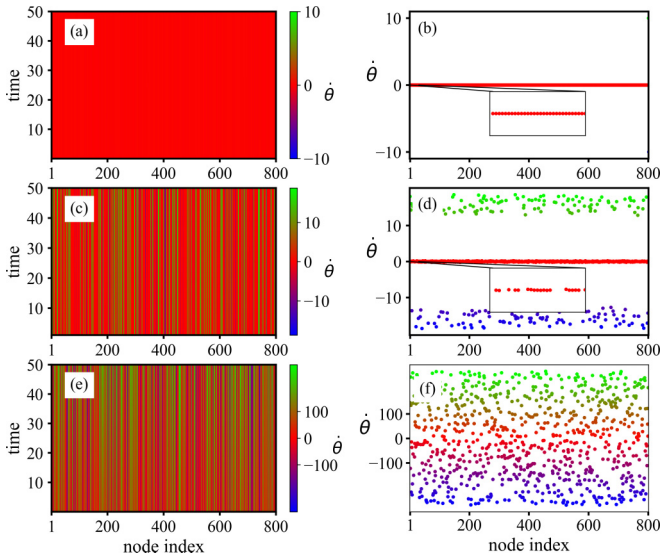

FIG. 13. The chimera state favors maximum information storage and transfer. Left column: Spatiotemporal plots of the variable $\dot{\theta}$ for node $i = 1, 2, \ldots, N$. Right column: Snapshots of the variable $\dot{\theta}$ at $t = 50$. All simulations under [(a) and (b)] coherent state for $\varepsilon = 0.10$, $\gamma = -2.30$, and $\alpha = 0$, [(c) and (d)] chimera state for $\varepsilon = 0.81$, $\gamma = -0.01$, and $\alpha = 0$, [(e) and (f)] incoherent state for $\varepsilon = 1.60$, $\gamma = -0.01$, and $\alpha = 0$.

states. The prediction x variable for $0 < t \leqslant 20$ for trained reservoirs is shown by pink dashed line (coherent state of RC), red dashed line (chimera state of RC), and green dashed line (incoherent state of RC) in Fig. 14. All those patterns of RC generate correct short term predictions and then deviate from the actual Lorenz-63 trajectories. However, after the failure

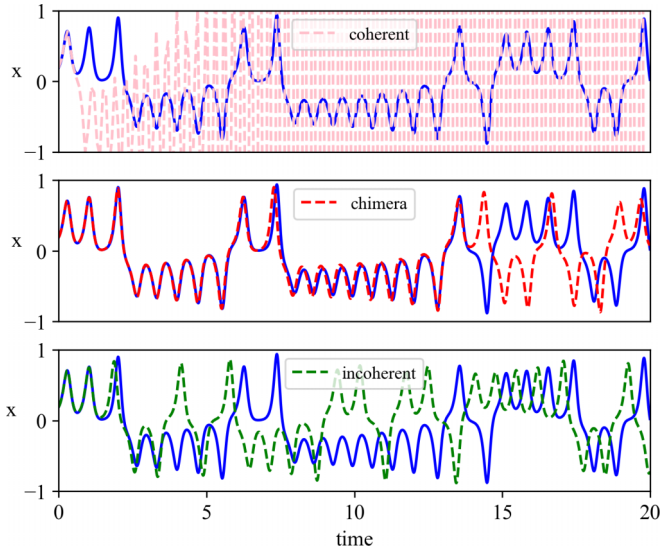

FIG. 14. Predictive performance across three different dynamical states. The pink dashed line denotes predictions in the coherent state, the red dashed line shows predictions in the chimera state, and the green dashed line represents predictions in the incoherent state. These results underscore the enhanced predictive potential associated with the chimera state.

of the short-term prediction, the different patterns of reservoir show substantially different predictions. With the incoherent state of RC, although the prediction deviates from the actual trajectory, the long-term behaviors appear to resemble that of the original Lorenz system. In contrast, with the coherent state of RC, which is not the case for reproducing the long-term behavior of the Lorenz-63 system.

When comparing the predictive outcomes of RC with the chimera state and the non-chimera state, the best performance does come when the observer is in the chimera state, while there are observers in the non-chimera state that cause the testing and training errors to become larger as the reservoir computing approaches coherent and incoherent states. We emphasize that our finding is that information storage and transfer are maximized in proximity to spatially characterized locations, as depicted in Figs. 13(c) and 13(d). The result highlights the essential role of the chimera state in forecasting tasks, it is a prerequisite for the reservoir to produce good predictions.

Regarding the observed tendency in Lorenz-63 system in Table II, we delve into the interplay between these parameters and system dynamics. Systems with hyperparameters that display the described tendencies invariably manifest chimera states. Taking the reservoir system with $N = 800$ as an example, mirroring the behavior in other systems with similar hyperparameter tendencies. As showcased in Figs. 13(c) and 13(d), the pattern of reservoir is the coexistence of coherent and incoherent states. Contrastingly, when we deviate from these hyperparameter tendencies, even if the parameters are optimal, the dynamical behavior of the system might not necessarily be that of a chimera. A case in point is the system with $N = 100$ does exhibit non-chimera dynamic. This divergence in dynamics, which is driven by hyperparameter selection, reveals an inherent relationship: finding the proper balance of hyperparameters may not be only about optimization in the traditional sense, but rather about creating the precise dynamical environment favourable to chimera formation. This supports the idea that for certain computing outcomes, not just any optimal state will suffice, the distinct dynamics of a chimera state may be required.

D. Influence of higher-order interactions on RC capability

In this study, we focus on the analysis of the interactive dynamics shown by the coupled pendulums model, specifically we consider the higher-order interactions. The incorporation of high-order interactions significantly enhances the dynamical pattern of a reservoir [41], resulting in increased complexity. This allows for a more accurate representation of intricate nonlinear relationships and a more comprehensive modeling of real-world phenomena. The process of enrichment not only enhances the representation of complex systems, but also enhances the reservoir's ability to handle advanced tasks. To show that the higher-order interactions are important for the system to act as a reservoir, we keep the other parameters of the RC unchanged but set the strength of higher-order interactions $k_2 = 0$, the coupled pendulums model without higher-order interactions (R2 system), and then train the reservoir using the same inputs.

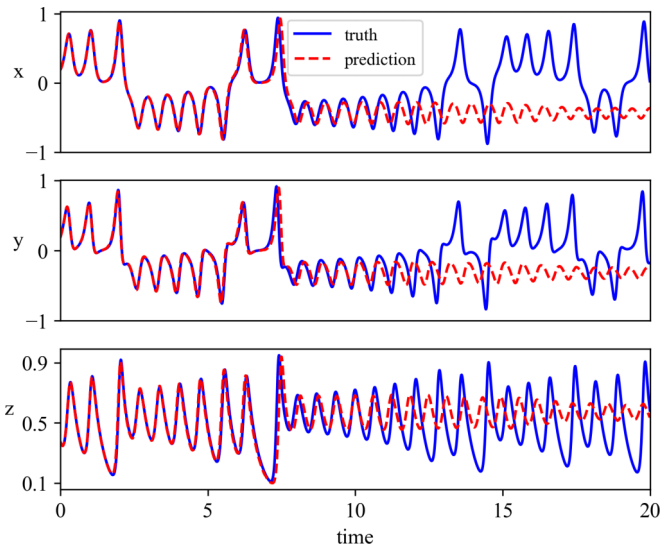

FIG. 15. The absence of higher-order interactions in the coupled pendulums model leads to inadequate prediction performance in the trained machines. The predicted state (red) of the optimized reservoir without higher-order interactions, and the actual trajectories (blue) of the Lorenz-63 system for $0 < t \leqslant 20$. The results within the other parameters of the RC are unchanged but set $k_2 = 0$ in Eq. (2).

Figure 15 depicts the state evolution predicted by R2 system. We observe that the R2 system is capable of predicting the future for approximately nine time units, which is less than R1 system. Figure 16(a) shows the predictive system ultimately fails to reconstruct the dynamic characteristics by the R2 system. The return map constructed from the predicted $z(t)$ in Fig. 16(b) is different from the actual one. In the case of no higher-order interactions, we regard the reservoir as being poorly trained. Thus, we clarify that the contribution of higher-order interactions to task performance is essential.

E. Influence of external nonlinearity when injecting inputs on RC capability

In traditional machine learning architecture, tanh is widely used as the nonlinear activation function among layers.

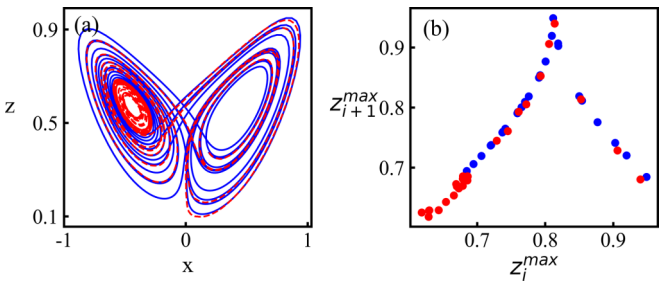

FIG. 16. In the absence of higher-order interactions in the coupled pendulums model, the prediction performance and pattern properties of the trained machines are poor. (a) Motion in phase space for the actual (blue) and predicted (red) trajectories for $0 < t \leqslant 50$. (b) Poincaré return map of successive local maxima of $z(t)$ for the actual (blue) and predicted (red) trajectories for $0 < t \leqslant 50$. Both (a) and (b) utilize the identical Lorenz trajectory and reservoir as Fig. 8.

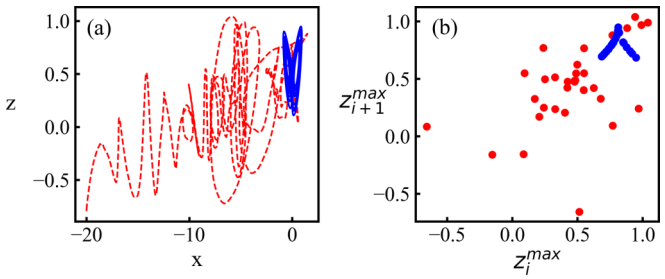

FIG. 17. In the absence of activation function tanh when injecting the inputs in the coupled pendulums model, the prediction performance and pattern properties of the trained machines are poor. (a) Motion in phase space for the actual (blue) and predicted (red) trajectories for $0 < t \leqslant 50$. (b) Poincaré return map of successive local maxima of $z(t)$ for the actual (blue) and predicted (red) trajectories for $0 < t \leqslant 50$.

Nonlinearity is well acknowledged to be a key aspect in comprehending information processing in reservoir computing. The standard strategy, commonly utilized in the RC community [46], is to select a nonlinear activation function tanh, we introduce an external nonlinearity tanh when injecting the linear sum of the inputs to the reservoir system in Eq. (2).

Choice of activation function tanh is important, we discuss how the performance varies when inputs are injected without tanh (R3 system). As shown in Fig. 17, the reservoir is completely incapable of producing accurate forecasts and fails to replicate the dynamics' long-term attractor. Due to the lack of the nonlinear compression effect (zero-center characteristic) provided by tanh, it cannot ensure bounded activations and capture the intricate nonlinear correlations present in the data. Without such nonlinearity, the system is unable to adequately represent and anticipate complex behaviors.

F. Largest Lyapunov exponent in different RC configurations.

It is critical to precisely capture the complex climate and long-term behavior in the domain of dynamical systems. The fundamental strategy is the calculation of the largest Lyapunov exponent (LLE) Λ_1 [47]. The LLE is acknowledged for its ability to quantitatively assess the divergence or convergence of trajectories inside a system. It serves as a robust indicator of a system's sensitivity to its initial conditions. In this section, we illustrate the capability of our technique with different configurations through LLE to replicate the climate of the Lorenz-63 system and Hindmarsh-Rose neural model, as shown in Table IV.

TABLE IV. The largest Lyapunov exponents Λ_1 for the Lorenz-63 system and Hindmarsh-Rose neural model. The reservoir set up in three configurations of coupled pendulums model. R1 system: coupled pendulums model with higher-order interactions; R2 system: coupled pendulums model without higher-order interactions; R3 system: coupled pendulums model with higher-order interactions, but no tanh.

	Actual system	R1 system	R2 system	R3 system
Λ_1^{L}	0.91	0.92	0.47	0.32
Λ_1^{HR}	0.18	0.18	0.02	0.14

For the Lorenz-63 system, while the actual system showcases LLE $\Lambda_1^L = 0.91$, our approach (R1 system) closely emulates this with 0.92. This near-perfect matching highlights the efficacy of the R1 system in mirroring intricate dynamics. In contrast, the absence of high-order interactions in the R2 system leads to a marked reduction to 0.47. When nonlinear activation is omitted during the input in the R3 system, the Λ_1^L further deteriorates to 0.32. For the Hindmarsh-Rose neural model, the actual HR system presents $\Lambda_1^{\mathrm{HR}} = 0.18$. Our R1 system parallels this value, reflecting its capability in accurate system representation. However, when high-order dynamics are excluded, the R2 system exhibits a raised $\Lambda_1^{\mathrm{HR}} = 0.02$, suggesting an underestimation of system chaos. The R3 system, bereft of nonlinear activation at the input, yields $\Lambda_1^{\mathrm{HR}} = 0.14$, which is a slight underestimation compared to the actual system. The R1 system exhibits a good capacity to accurately replicate the dynamic characteristics of both the Lorenz-63 and HR systems. The significance of the discrepancy revealed in the R2 and R3 systems highlights the crucial need of high-order dynamics and nonlinear activations in achieving precise chaos prediction and system representation. The considerable disparity observed in Λ_1 upon the exclusion of these pivotal components serves to underscore their importance in the modeling of complex dynamical systems.

V. CONCLUSIONS

To summarize, we propose a coupled pendulums model with higher-order interactions as a suitable candidate for reservoir computation within the fundamental context of dynamical systems considered here. We have studied coupled pendulums with higher-order interactions as dynamical systems for reservoir, the spatiotemporal properties of this reservoir are explored with respect to chaotic short-time prediction and long-time climate reproduction tasks. We have numerically investigated the Bayesian optimization algorithm to tune the best hyperparameters for the reservoir. We tested the Lorenz-63 chaotic system and the Hindmarsh-Rose neural model under the same internal composition of the reservoir, and we showed that the nonlocal coupling network has good predictive effects as a reservoir. We explored the predictive capabilities of our reservoir, emphasizing the role of higher-order interactions. By analyzing the interplay between different reservoir configurations and prediction performance, combined with calculations of largest Lyapunov exponents, we highlighted the significance of higher-order interactions. The chimera state is identified as the ideal dynamic for prediction. We illustrated the greater predictability of this state by analyzing the interplay between system features and hyperparameter setups. The underlying mechanisms that contribute to this heightened predictability warrant further exploration. These results help to understand the computational benefits of coupled pendulums as reservoirs.

The data that support the findings of this study are available in the manuscript.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Grant No. 12072262), and by the Shaanxi Computer Society and Xi'an Xiangteng Microelectronics Technology Co., Ltd. We express our gratitude to the anonymous reviewers for thoughtful suggestions. The authors declare no potential conflicts of interest with respect to the research, authorship, and publication of this manuscript.

[1] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).

[2] 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 Report No. 148, 2001, p. 13.

[3] 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 (2002).

[4] D. J. Gauthier, Reservoir computing: Harnessing a universal dynamical system, SIAM News **51**, 12 (2018).

[5] G. Tanaka, T. Yamane, J. B. Hćroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Recent advances in physical reservoir computing: A review, Neural Netw. **115**, 100 (2019).

[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, Mach. Learn. Appl. **8**, 100300 (2022).

[7] J. A. Platt, S. G. Penny, T. A. Smith, T.- C. Chen, and H. D. I. Abarbanel, A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics, Neural Netw. **153**, 530 (2022).

[8] 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).

[9] F. Triefenbach, K. Demuynck, and J.- P. Martens, Improving large vocabulary continuous speech recognition by combining GMM-based and reservoir-based acoustic modeling, 2012 (IEEE Spoken Language Technology Workshop (SLT), Miami, 2012), pp. 107–112.

[10] K. Caluwaerts, M. D'Haene, D. Verstraeten, and B. Schrauwen, Locomotion without a brain: Physical reservoir computing in tensegrity structures, Artif. Life **19**, 35 (2013).

[11] Y. Chen, Y. Qian, and X. Cui, Time series reconstructing using calibrated reservoir computing. Sci. Rep. **12**, 16318 (2022).

[12] Y. Zhong, J. Tang, X. Li, B. Gao, H. Qian, and H. Wu, Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing, Nature Commun. **12**, 1 (2021).

[13] B. Schrauwen, D. Verstraeten, and J. Van Campenhout, An overview of reservoir computing: Theory, applications and implementations, in *Proceedings of the 15th European Symposium*

on Artificial Neural Networks (ESANN), Bruges, Belgium, 2007, pp. 471–482.

[14] P. Kaelo and M. Ali, Some variants of the controlled random search algorithm for global optimization, J. Optim. Theory Appl. **130**, 253 (2006).

[15] A. Griffith, A. Pomerance, and D. J. Gauthier, Forecasting chaotic systems with very low connectivity reservoir computers, Chaos **29**, 123108 (2019).

[16] P. Antonik, N. Marsal, D. Brunner, and D. Rontani, Bayesian optimization of large-scale photonic reservoir computers, Cogn. Comput. **13**, 1 (2021).

[17] L. A. Thiede, and U. Parlitz, Gradient based hyperparameter optimization in echo state networks, Neural Netw. **115**, 23 (2019).

[18] J. T. Wilson, F. Hutter, and M. P. Deisenroth, Maximizing acquisition functions for Bayesian optimization, in: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, (NeurIPS), Montréal, Canada, 2018, pp. 9906–9917.

[19] A. Haluszczynski and C. Rth, Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing, Chaos **29**, 103143 (2019).

[20] J. Pathak, B. Hunt, M. Girvan, Z. X. Lu, and E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, Phys. Rev. Lett. **120**, 024102 (2018).

[21] K. I. Kitayama, Guiding principle of reservoir computing based on small-world network, Sci. Rep. **12**, 16697 (2022).

[22] I. Matsumoto, S. Nobukawa, N. Wagatsuma, and T. Kurikawa, Functionality of neural dynamics induced by long-tailed synaptic distribution in reservoir computing, Nonlinear theory appl. IEICE **14**, 342 (2013).

[23] T. Lymburn, S. D. Algar, M. Small, and T. Jüngling, Reservoir computing with swarms, Chaos **31**, 033121 (2021).

[24] N. Bertschinger, T. Natschläger, Real-time computation at the edge of chaos in recurrent neural networks, Neural Comput. **16**, 1413 (2004).

[25] R. Martínez-Peña, and J. P. Ortega, Quantum reservoir computing in finite dimensions, Phys. Rev. E **107**, 035306 (2023).

[26] D. Snyder, A. Goudarzi, and C. Teuscher, Computational capabilities of random automata networks for reservoir computing, Phys. Rev. E **87**, 042808 (2013).

[27] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass, Towards a theoretical foundation for morphological computation with compliant bodies, Biol. Cybern. **105**, 355 (2011).

[28] K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, A soft body as a reservoir: Case studies in a dynamic model of octopus-inspired soft robotic arm, Front. Comput. Neurosci. **7**, 91 (2013).

[29] K. Caluwaerts, J. Despraz, A. Işçen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral, J. R. Soc., Design and control of compliant tensegrity robots through simulation and hardware validation, Interface **11**, 20140520 (2014).

[30] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, Information processing via physical soft body, Sci. Rep. **5**, 10487 (2015).

[31] K. Tanaka, Y. Tokudome, Y. Minami, S. Honda, T. Naka-jima, K. Takei, and K. Nakajima, Self-organization of remote reservoirs: Transferring computation to spatially distant locations, Adv. Intell. Syst. **4**, 2100166 (2022).

[32] P. Bhovad and S. Li, Physical reservoir computing with origami and its application to robotic crawling, Sci. Rep. **11**, 13002 (2021).

[33] T. Kanao, H. Suto, K. Mizushima, H. Goto, T. Tanamoto, and T. Nagasawa, Reservoir Computing on Spin-Torque Oscillator Array, Phys. Rev. Appl. **12**, 024052 (2019).

[34] M.-K. Lee and M. Mochizuki, Reservoir Computing with Spin Waves in a Skyrmion Crystal, Phys. Rev. Appl. **18**, 014074 (2022).

[35] N. Babson and C. Teuscher, Reservoir computing with complex cellular automata, Complex Syst. **28**, 433 (2019).

[36] G. Milano, G. Pedretti, K. Montano, S. Ricci, S. Hashemkhani, L. Boarino, D. Ielmini, and C. Ricciardi, Reservoir computing with complex cellular automata, Nature Mater. **21**, 195 (2022).

[37] L. Wang, H. Fan, J. Xiao, Y. Lan, and X. Wang, Criticality in reservoir computer of coupled phase oscillators, Phys. Rev. E **105**, L052201 (2022).

[38] T. Kapitaniak, P. Kuzma, J. Wojewoda, K. Czolczynski, and Y. Maistrenko, Imperfect chimera states for coupled pendula, Sci. Rep. **4**, 6379 (2014).

[39] D. Dudkowski, K. Czołczyński, and T. Kapitaniak, Multistability and synchronization: The co-existence of synchronous patterns in coupled pendula, Mech. Syst. Signal Process **166**, 108446 (2022).

[40] X. Li, D. Ghosh, and Y. Lei, Chimera states in coupled pendulum with higher-order interaction, Chaos Solitons Fractals **170**, 113325 (2023).

[41] P. Cisneros-Velarde, F. Bullo, Multi-group SIS epidemics with simplicial and higher-order interactions, IEEE Trans. Control Netw. Syst. **9**, 695 (2021).

[42] H. Jaeger, M. Lukoáevicius, D. Popovici, and U. Siewert, Optimization and applications of echo state networks with leaky-integrator neurons, Neural Netw. **20**, 335 (2007).

[43] E. N. Lorenz, Deterministic nonperiodic flow, J Atmos Sci **20**, 130 (1963).

[44] J. H. Friedman, Greedy function approximation: A gradient boosting machine, Ann. Stat. **29**, 1189 (2001).

[45] N. Corson, M. Aziz-Alaoui, Asymptotic dynamics of Hindmarsh-Rose neuronal system, Dyn. Cont. Dis. Ser. **B4**, 535 (2009).

[46] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. Barbosa, Next generation reservoir computing, Nature Commun. **12**, 5564 (2021).

[47] M. T. Rosenstein, J. J. Collins, and C. J. D. Luca, A practical-method for calculating largest lyapunov exponents from small data sets, Physica D **65**, 117 (1993).