Emergence of transient chaos and intermittency in machine learning

Journal of Physics: Complexity

OPEN ACCESS

PAPER

Emergence of transient chaos and intermittency in machine learning

Ling-Wei Kong[1], Huawei Fan[2], Celso Grebogi[3] and Ying-Cheng Lai[1,4,*]

[1] School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287, United States of America
[2] School of Physics and Information Technology, Shaanxi Normal University, Xi'an 710062, People's Republic of China
[3] Institute for Complex Systems and Mathematical Biology, King's College, University of Aberdeen, Aberdeen AB24 3UE, United Kingdom
[4] Department of Physics, Arizona State University, Tempe, AZ 85287, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: Ying-Cheng.Lai@asu.edu

Keywords: transient chaos, machine learning, reservoir computing, intermittency, scaling law

RECEIVED
4 February 2021

REVISED
8 June 2021

ACCEPTED FOR PUBLICATION
14 June 2021

PUBLISHED
2 July 2021

Abstract

An emerging paradigm for predicting the state evolution of chaotic systems is machine learning with reservoir computing, the core of which is a dynamical network of artificial neurons. Through training with measured time series, a reservoir machine can be harnessed to replicate the evolution of the target chaotic system for some amount of time, typically about half dozen Lyapunov times. Recently, we developed a reservoir computing framework with an additional parameter channel for predicting system collapse and chaotic transients associated with crisis. It was found that the crisis point after which transient chaos emerges can be accurately predicted. The idea of adding a parameter channel to reservoir computing has also been used by others to predict bifurcation points and distinct asymptotic behaviors. In this paper, we address three issues associated with machine-generated transient chaos. First, we report the results from a detailed study of the statistical behaviors of transient chaos generated by our parameter-aware reservoir computing machine. When multiple time series from a small number of distinct values of the bifurcation parameter, all in the regime of attracting chaos, are deployed to train the reservoir machine, it can generate the correct dynamical behavior in the regime of transient chaos of the target system in the sense that the basic statistical features of the machine generated transient chaos agree with those of the real system. Second, we demonstrate that our machine learning framework can reproduce intermittency of the target system. Third, we consider a system for which the known methods of sparse optimization fail to predict crisis and demonstrate that our reservoir computing scheme can solve this problem. These findings have potential applications in anticipating system collapse as induced by, e.g., a parameter drift that places the system in a transient regime.

1. Introduction

Recent years have witnessed a growing interest in exploiting machine learning for model-free prediction of the state evolution of chaotic dynamical systems [1–18], with a focus on reservoir computing (a type of recurrent neural networks (RNNs)) [19–22]. The core of a reservoir computing machine is a nonlinear dynamical network of artificial neurons, typically of complex topology. With proper training based on time series data from the target chaotic system of interest, the network becomes a self-evolving dynamical system that supposedly represents a replica of the target system to some reasonable accuracy. From the same initial condition, temporal synchronization can be achieved between the reservoir machine and the target system [16], enabling prediction for a finite duration of time. In most existing studies, the attention has been to the parameter regime of the target system, where there is a chaotic attractor, so training is done using time series data from the attractor with the goal to predict the state evolution as determined by this underlying attractor. Because of these fea-

tures, the corresponding reservoir computing machine itself is trained into a dynamical system that generates a chaotic attractor.

In this paper, we build on our recent work [23] to address the issue of machine-generated transient chaos. In principle, if the machine is trained with an ensemble of transient chaotic time series, it should be able to generate transient chaos insofar as the amount of training data is sufficient. However, in nonlinear dynamical systems, the occurrence of transient chaos implies the inevitable collapse of the system to an undesired state. We thus assume that transient chaotic time series are not available and the available training data are collected while the system is in a parameter regime of chaotic attractors. Our recent work [23] has demonstrated that it is possible to train the neural machine with attracting chaotic time series to predict the critical transition from sustained to transient chaos. There is also initial evidence that a properly trained reservoir computing machine is able to generate transient chaos. *The present paper has three main points that go beyond our recent work* [23]. First, we carry out a more systematic analysis of the statistical behaviors of the machine-generated transient chaos, using different model chaotic systems than those in reference [23], thereby widening the scope of the finding that a reservoir computing machine can be trained to faithfully generate transient chaos that agrees statistically with the ground truth. Second, we demonstrate that a properly trained machine can generate intermittency. (To the best of our knowledge, there were no previous works on predicting intermittency with reservoir computing.) Third, we focus on a paradigmatic chaotic system for which the existing sparse optimization methods fail to predict crisis and transient chaos, and demonstrate that our parameter-cognizant machine learning approach can solve this problem. This result was not reported in reference [23]. It should be emphasized that, for the neural machine to generate statistically meaningful transient chaos, training should be done with attracting chaotic time series because normal functioning of the system is often associated with a chaotic attractor, while transient chaos leads to system collapse. In such a situation, it is feasible to collect the time series only when the system still functions normally, for if the system is in a transient chaotic regime, it will collapse after a finite duration of time, rendering infeasible to obtain sufficient amount of training data.

The main idea behind predicting crisis [23] and training a reservoir computing machine to generate the correct transient chaotic dynamics is the following. Let p be the bifurcation parameter of the target chaotic system, with p_c being the critical point, where the system exhibits a chaotic attractor for $p < p_c$ and transient chaos for $p > p_c$. We train the reservoir machine with time series collected from a small number of parameter values in the attractor regime. For each parameter value, we stipulate that the machine is well trained in the sense that it is capable of predicting correctly and accurately the chaotic evolution at the same parameter value for a reasonable amount of time. As training is done at multiple parameter values, it is imperative that the machine is made to be cognizant of the parameter value, which can be accomplished by designating a special input channel for the values of the bifurcation parameter [23]. It has been demonstrated [23] that, insofar as the machine is well trained for a small number of parameter values in the attractor regime, it can predict the crisis transition point. Here, we shall demonstrate that a parameter change that pushes the value of p beyond p_c will make the machine to generate the 'correct' transient chaos in the sense that, statistically, the transient chaotic behaviors generated by the neural machine agree with those of the target system.

These results have practical applications in providing early warnings for a possible system collapse. For example, if the target chaotic system is in the attractor regime close to the critical point and is regarded as functioning normally, a direct examination of the time series would give absolutely no indication that the system could collapse upon a small parameter drift. A well trained reservoir computing machine—a faithful replica of the original system, can predict possible drift of the system into the transient chaos regime and the subsequent collapse [23].

It is important to place the main idea behind our recent work [23] and the present work in a proper perspective with respect to previous works. The idea of a parameter-aware RNN was proposed in an early work [24], where the authors trained an RNN using time series from different systems for different parameter values, and demonstrated that with a "fixed weight neural network," changing input alone can make the RNN produce various dynamical behaviors of the target systems with one-time-step predictions. More recently, reservoir computing with a parameter channel has been studied independently in references [25–27]. In reference [25], the approach was used to predict the occurrence of periodic windows and other regime transitions in non-stationary chaotic systems with or without dynamical noise. In reference [26], the approach was applied to the Lorenz system to predict Hopf, saddle-node, and pitchfork bifurcations, where training was carried out based on the normal forms of the bifurcations. In reference [27], a relevant yet different approach of dynamical learning with reservoir computing was articulated, where an error feedback loop and a context feedback loop were added to the standard reservoir structure. It was shown that, after training, the modified reservoir system has the ability to learn dynamics that were different from those of the training set with a small amount of data. The process was named as 'dynamical learning', which may be understood as an automatic adaptation of the fixed weight neural network with the error and context feedback loops. In the supplemental material

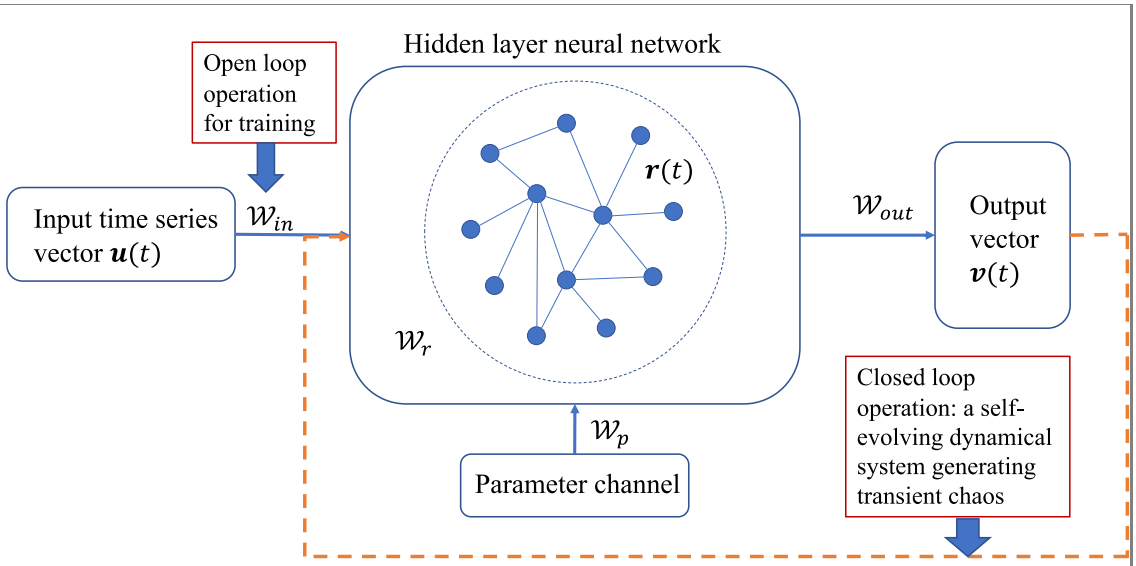

Figure 1. Reservoir computing machine for generating transient chaos. The neural network consists of three layers: the input layer, the hidden layer and the output layer. The three vectors $\mathbf{u}(t)$, $\mathbf{r}(t)$, and $\mathbf{v}(t)$ denote the input signals, the dynamical states of the hidden layer, and the output signals, respectively. The value of the bifurcation parameter of the target system is fed into the hidden layer neural network through the parameter channel. During the training, the reservoir system is open as it takes in external time series at a small number of parameter values in the regime in which the target system has a chaotic attractor. After training, the output variables $\mathbf{v}(t)$ is connected to the input, closing the system and making it a self-evolving dynamical system that can generate transient chaos for input parameter values beyond the attractor regime.

of reference [27], it was demonstrated that the framework is capable of predicting the Hopf bifurcation in the Lorenz system. Regarding transient chaos, in reference [26], it was demonstrated that the reservoir's predicted trajectory of the Lorenz attractor behaves chaotically for a while, and then begins to fall into a fixed point. Recent works [23, 26, 27] have thus demonstrated the ability of reservoir computing machines to learn and predict transient chaos. Our present work focuses on the following three aspects: (1) statistical properties of machine-generated transient chaos, (2) intermittency, and (3) transient chaos in nonlinear dynamical systems for which the previous sparse-optimization based prediction methods failed.

2. Parameter cognizant reservoir computing machine for generating transient chaos

Here we briefly describe our recent parameter-cognizant reservoir computing framework for predicting crisis and transient chaos [23]. The codes of this work are available from GitHub [28].

A reservoir computing machine is a RNN, where the elements of the input-layer matrix and of the connection matrix of the dynamical network in the hidden layer are randomly chosen and held fixed. The only entity to be determined through training is the elements of the output-layer matrix. Figure 1 shows that, the input time-series data constitute a $D_{\rm in}$-dimensional input vector $\mathbf{u}(t)$, and the input matrix $\mathcal{W}_{\rm in}$ of dimension $D_{\rm r} \times D_{\rm in}$ projects $\mathbf{u}(t)$ into a high-dimensional state vector in the the hidden layer. The bifurcation parameter p of the target system is fed into the hidden layer through the parameter channel defined by matrix $\mathcal{W}_{\rm p}$ of dimension $D_{\rm r} \times D_{\rm p}$. Matrix $\mathcal{W}_{\rm r}$ of dimension $D_{\rm r} \times D_{\rm r}$ is the connection matrix of the hidden layer, which typically has a random topology. The dynamical evolution of the nodal state in the hidden layer is governed by a nonlinear activation function, such as the hyperbolic tangent function. The nodal states of the network in the hidden layer at time step t is represented by the $D_{\rm r}$-dimensional vector $\mathbf{r}(t)$. The output layer matrix $\mathcal{W}_{\rm out}$ of dimension $D_{\rm out} \times D_{\rm r}$ is a readout matrix from the hidden state vector $\mathbf{r}(t)$ to the output vector $\mathbf{v}(t)$ of dimension $D_{\rm out} = D_{\rm in}$.

Prior to training, the weights (matrix elements) in $\mathcal{W}_{\rm in}$, $\mathcal{W}_{\rm p}$ and $\mathcal{W}_{\rm r}$ are generated randomly. The matrices are then fixed afterward. Specifically, the weights of $\mathcal{W}_{\rm in}$ are generated from a uniform distribution in the interval $[-k_{\rm in}, k_{\rm in}]$ and the weights of $\mathcal{W}_{\rm p}$ are drawn from another uniform distribution in the interval $[-k_{\rm p}, k_{\rm p}]$. Both $\mathcal{W}_{\rm in}$ and $\mathcal{W}_{\rm p}$ are dense matrices so that each node in the input layer is connected to all the nodes in the hidden layer. The matrix $\mathcal{W}_{\rm r}$ defines a random network of size $D_{\rm r}$ and average degree d, which is undirected and weighted with the weights drawn from a standard normal distribution and rescaled such that the spectral radius of the network is λ (a hyperparameter). Here, the average degree d of a network is the average number of links that a node has. The spectral radius λ of a network is the largest absolute value of the eigenvalues of its adjacency (connection) matrix.

During training, the time series and the associated value of the bifurcation parameter are fed into the machine in a step-by-step manner. The dynamical evolution of the reservoir hidden state $\mathbf{r}(t)$ is governed by the following rule:

$$\mathbf{r}(t+\Delta t) = (1-\alpha)\mathbf{r}(t) + \alpha \tanh[\mathcal{W}_{\rm r}\cdot\mathbf{r}(t) + \mathcal{W}_{\rm in}\cdot\mathbf{u}(t) + \mathcal{W}_{\rm p}(p+p_0)], \tag{1}$$

where Δt is the time step, $\tanh(\mathbf{q}) \equiv [\tanh(q_1), \tanh(q_2), \ldots]^{\rm T}$ for $\mathbf{q} = [q_1, q_2, \ldots]^{\rm T}$, α is the leakage factor, p_0 is the bias of p. A reservoir computing machine is thus a discrete-time dynamical system. To ensure that it can accurately represent a target dynamical system, we use time steps that are two orders of magnitude smaller than the typical time scale of the target system. The initial condition for the hidden state can be conveniently set to be $\mathbf{r}(t=0) = \mathbf{0}$. The process is repeated for each value of the training bifurcation parameter. (The effect of the relative locations of training points will be discussed in section 4.1.) The state vectors $\mathbf{r}(t)$ at all the time steps are recorded, which allows the output matrix $\mathcal{W}_{\rm out}$ to be calculated through a standard regression process between the true data vector $\mathbf{u}(t)$ and the hidden state vector $\mathbf{r}(t)$. Because of the need of training at a number of distinct values of the bifurcation parameter, there are multiple pairs of $\mathbf{u}(t)$ and $\mathbf{r}(t)$. We stack these pairs together in the temporal dimension to form a pair of vectors $\mathbf{r}_{\rm all}(t)$ and $\mathbf{u}_{\rm all}(t)$ that extend a longer temporal domain. To remove the undesired transient behaviors of the reservoir dynamical network, the first 10 time steps for each pairs of $\mathbf{u}(t)$ and $\mathbf{r}(t)$ are disregarded before they are stacked. The regression method in reference [4] is used to deal with the issue of symmetries in the reservoir system, where $\mathbf{r}_{\rm all}(t)$ is replaced by $\mathbf{r}'_{\rm all}(t)$ with $\mathbf{r}'_{\rm all}(t)_i = \mathbf{r}_{\rm all}(t)_i^2$ for even rows (corresponding to nodes in the hidden layer with even indices) and the other elements in odd rows being the same as in $\mathbf{r}_{\rm all}(t)$. A standard linear regression between $\mathbf{u}_{\rm all}(t)$ and $\mathbf{r}'_{\rm all}(t)$ can then be carried out through minimizing the loss function

$$\mathcal{L} = \sum_t \|\mathbf{u}_{\rm all}(t) - \mathcal{W}_{\rm out}\cdot\mathbf{r}'_{\rm all}(t)\|^2 + \beta\|\mathcal{W}_{\rm out}\|^2, \tag{2}$$

where $\beta > 0$ is the l_2-regularization coefficient. The regularized regression can be achieved through

$$\mathcal{W}_{\rm out} = \mathbf{U}\cdot\mathbf{R}'^{\rm T}\cdot(\mathbf{R}'\cdot\mathbf{R}'^{\rm T} + \beta\mathbf{I})^{-1}, \tag{3}$$

where $\mathbf{I}$ is an identity matrix of dimension $D_{\rm r}$, $\mathbf{U}$ and $\mathbf{R}'$ are the matrix forms of $\mathbf{u}_{\rm all}(t)$ and $\mathbf{r}'_{\rm all}(t)$, respectively, with different columns representing different time steps and different rows corresponding to different dimensions.

Validation is achieved by letting the trained reservoir machine make short-time predictions of the target system for each training value of the bifurcation parameter. In particular, the prediction is obtained through

$$\mathbf{v}(t) = \mathcal{W}_{\rm out}\cdot\mathbf{r}'(t). \tag{4}$$

Comparing between the predicted and real time series leads to the validation error. Since the real time series are not available during prediction, the input vector $\mathbf{u}(t)$ in equation (1) is replaced by the output vector $\mathbf{v}(t)$ from the last time step. In the validation and prediction phases, equation (1) becomes

$$\mathbf{r}(t+\Delta t) = (1-\alpha)\mathbf{r}(t) + \alpha \tanh[\mathcal{W}_{\rm r}\cdot\mathbf{r}(t) + \mathcal{W}_{\rm in}\cdot\mathcal{W}_{\rm out}\cdot\mathbf{r}'(t) + \mathcal{W}_{\rm p}(p+p_0)], \tag{5}$$

which effectively defines the reservoir-computing machine as a self-evolving dynamical system under external parameter input p. As the time series are validated immediately after the training phase, the initial condition of the reservoir hidden state can be set as the state from the last time step of training. The typical validation length is about 4–6 Lyapunov times of the target system. (For a chaotic system, the Lyapunov time is a characteristic time scale, which is defined as the inverse of the largest Lyapunov exponent.) The prediction error is the average root mean square error.

Taken together, the so-trained and validated reservoir machine is now a stand-alone, self-evolving dynamical system. When the input parameter channel takes on any value in the vicinity of the training parameter values, the system generates sustained chaotic behaviors. The system can generate transient chaos if the input parameter value is in the regime of transient chaos of the original target system. We emphasize that the reservoir machine has never been trained in this regime of transient chaos, i.e., the parameter values in this regime are completely 'new' to the reservoir machine. As the reservoir machine has a high-dimensional hidden states, it is necessary to set not only the initial input vector but also the initial hidden states appropriate for prediction. We use a short period of the real time series from the target system (e.g., several oscillation cycles), taking from the training parameter regime to 'warm up' the neural network.

While the trained reservoir system can generate transient chaos, do the characteristics of the chaotic transients agree with those of the target system in the same parameter regime? To ensure a reasonable agreement, it is necessary to optimize the reservoir system in its ability to acquire the 'dynamical climate' of the target

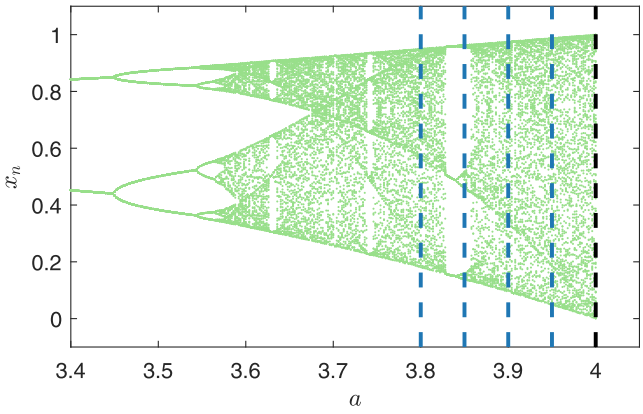

Figure 2. Bifurcation diagram of the classic logistic map. The vertical black dashed line denotes the crisis at $a = a_c = 4$. The vertical blue dashed lines denote the training points of the reservoir machine: $a = 3.8, 3.85, 3.9$ and 3.95, all in the attractor regime, where a chaotic or a periodic attractor lives in the unit interval.

system. We take the following steps. For choosing the values of the seven hyperparameters ($k_{\rm in}$, $k_{\rm p}$, p_0, d, λ, α and β), we repeat the training 800 times with different values of the hyperparameters for optimization. We use the function 'surrogateopt' in Matlab for choosing the values of the hyperparameters from 800 iterations. However, even when the values of the hyperparameters have been optimized, the randomness in the matrices $\mathcal{W}_{\rm in}$, $\mathcal{W}_{\rm p}$ and $\mathcal{W}_{\rm r}$ will cause fluctuations and errors in the prediction results, as the optimized hyperparameter values determine only a few statistical properties of them and there is still a great degree of freedom in choosing the matrix elements. Inevitably, some realizations of these matrices can make the reservoirs unable in learning the dynamics of the target system. A more detailed discussion and simulation results about the effects of different realizations of the random system matrices can be found in reference [23]. In the present work, we use a simple method to reduce these errors: we conduct the training using five different realizations of these matrices and pick the one with the smallest validation error. The result is that, in the validation phase, with the optimized hyperparameter values and one out of five random realizations chosen, the reservoir computing machine can replicate the true dynamical evolution with small errors (relative errors less than 5%) for at least four or five Lyapunov times.

3. Results

3.1. Machine generated transient chaos in the logistic map

We first use the classic logistic map, $x_{n+1} = ax_n(1 - x_n)$, to demonstrate the emergence of transient chaos in machine learning, where a is the bifurcation parameter. A bifurcation diagram is shown in figure 2, where a critical transition occurs at $a_c = 4.0$ as denoted by the vertical black dashed line. For $3.0 < a < a_c$, the unit interval $(0, 1)$ is invariant, which contains an attractor together with coexisting non-attracting invariant sets. At $a = a_c = 4$, a boundary crisis [29] occurs, which converts a chaotic attractor into a non-attracting chaotic invariant set. For $a > a_c$, there is transient chaos within the interval $(0, 1)$ that eventually leads to escape to the infinity. Figures 3(a1) and (a2) show the typical dynamical behaviors of the logistic map for $a < a_c$ and $a > a_c$, respectively.

We train the reservoir machine at four values of the bifurcation parameter: $a = 3.8, 3.85, 3.9$ and 3.95, all in the attractor regime where there is a chaotic or a periodic attractor, as indicated in figure 2 by the vertical blue dashed lines. During the validation for each a value trained, the reservoir machine is able to accurately generate the state evolution of the target system for more than 4–5 Lyapunov times. More importantly, the machine generated trajectories for an arbitrarily long stretch of time land on the chaotic attractor. The training length is 1000 iterations for each value of a, and the validation length is 15 iterations. The 'warming up' length is 30 iterations. The values of the hyperparameters are optimized to be $D_{\rm r} = 400$, $d = 210$, $\lambda = 1.47$, $k_{\rm in} = 2.13$, $k_{\rm p} = 1.15$, $p_0 = -0.30$, $\alpha = 0.86$, and $\beta = 1 \times 10^{-6}$.

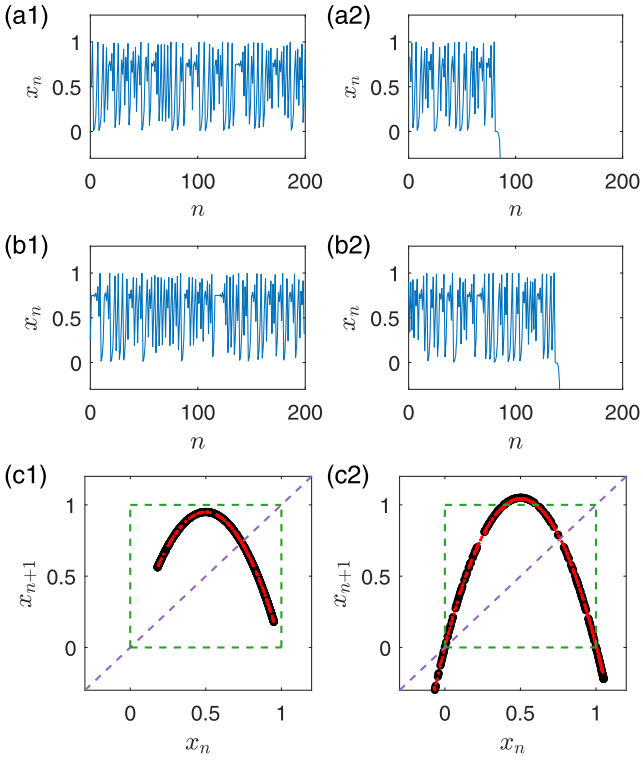

Figure 3. Machine generated transient chaos in the logistic map. (a1) and (a2) Typical time series of x_n of the logistic map in the normal and transient regimes, respectively, for $a = 3.8 < a_c$ and $a = 4.2 > a_c$. (b1) and (b2) Time series of x_n generated by a reservoir machine for the same values of a as in (a1) and (a2), respectively. The machine gives correctly that there is transient chaos for $a > a_c$. (c1) and (c2) Real logistic map (black dots) and reservoir-computing generated return maps (red dots). The green dashed square defines an interval: a chaotic attractor for $a = 3.8 < a_c$ (c1) or a nonattracting chaotic set for $a = 4.2 > a_c$ due to the escaping region at the top of the panel leading to transient chaos (c2).

After training, we impose systematic changes in a and test if the reservoir machine generates transient chaos. An exemplary pair of the machine generated time series for $a = 3.8 < a_c$ and $a = 4.2 > a_c$ are shown in figures 3(b1) and (b2), respectively, where the reservoir machine correctly generates transient chaos in the latter case. To assess whether the machine generated transient behavior is 'correct' in the sense that it matches with the ground truth, we calculate the return map from the machine trajectories and compare it with the true map. The results are shown in figures 3(c1) and (c2) for $a = 3.8 < a_c$ and $a = 4.2 > a_c$, respectively, where the red and black dots represent the machine generated and the true maps. The agreement is remarkable. In particular, for $a = 3.99 < a_c$, there is a green dashed square defining an interval in which a chaotic attractor lies. For $a = 4.01 > a_c$, the invariant set in the unit interval becomes nonattracting and is a fractal, where there is an escaping region outside the green square, leading to transient chaos occurring on the unit interval.

A well trained reservoir machine is capable of generating transient chaos with statistical characteristics matching those of the real system. We examine a fundamental characteristic of transient chaos: the lifetime distribution. In the transient chaotic regime, the true distribution is exponential. As shown in figure 4(a) for $a = a_c + 0.01$, the distribution of the length of the machine generated transiently chaotic trajectories is indeed exponential, where 1500 stochastic realizations of the reservoir system and 1500 random initial conditions for each realization are used. The average transient lifetime from the fitted slope of the data points in figure 4(a) is $\langle\tau\rangle \approx 28$, while that of the real system is about 33. The scaling law of the average transient lifetime $\langle\tau\rangle$ with $(a - a_c)$ produced by the machine is shown in figure 4(b), which is algebraic: $\langle\tau\rangle \sim (a - a_c)^{-\gamma}$ for $a > a_c$, where $\gamma \approx 0.62$. This agrees with the real scaling law with $\gamma \approx 0.58$. The distribution and expected value of the lifetime of the machine generated chaotic transients, as well as the scaling relation of the average transient lifetime with parameter variation beyond the critical transition point, all agree sufficiently well with the corresponding behaviors of the real system, attesting to the trained capability of the reservoir machine to generate authentic transient chaos, expected from the real system.

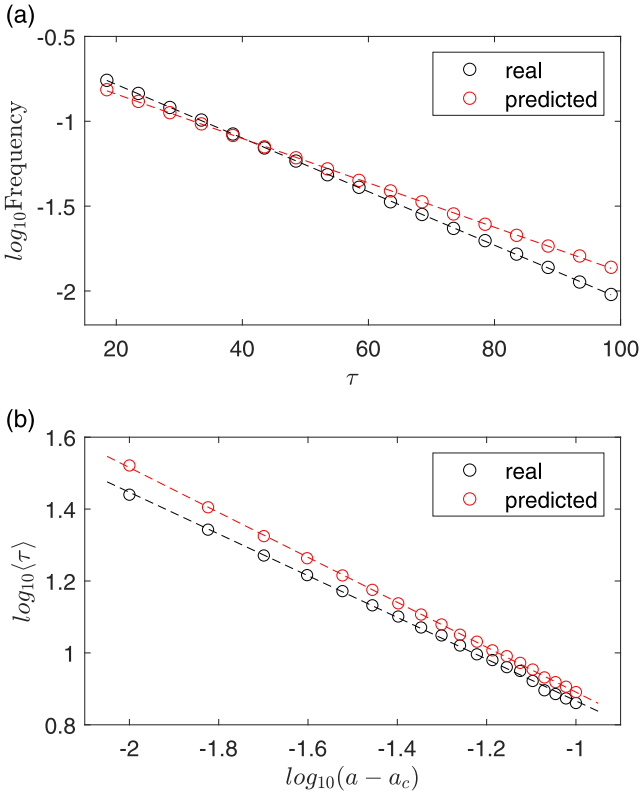

Figure 4. Characteristics of machine generated transient chaos for the logistic map in comparison with the ground truth. (a) Transient lifetime distributions for $a = a_c + 0.01$ from the real system (black) and from the machine (red). (b) Scaling law of the average transient lifetime with the parameter difference $(a - a_c)$ by the reservoir (red) in comparison with the true scaling (black). The statistical characteristics of the machine generated chaotic transients in (a) and (b) are obtained from 1500 random reservoir machine realizations, each realization with 1500 different random 'warming up' sequences of initial conditions.

3.2. Machine generated transient chaos in the classic Lorenz system

We next demonstrate that our parameter-cognizant reservoir computing machine can faithfully generate transient chaos from the classic Lorenz system [30]:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= x(\rho - z) - y, \\ \frac{dz}{dt} &= xy - \eta z, \end{aligned} \tag{6}$$

where $\sigma = 10$, $\eta = 8/3$, and ρ is the bifurcation parameter. Figure 5 shows a representative bifurcation diagram, which indicates that a crisis occurs at $\rho_c = 24.06$ (denoted by the vertical black dashed line). An exemplary pair of time series $x(t)$ for $\rho > \rho_c$ and $\rho < \rho_c$ are shown in figures 6(a1) and (a2), respectively, where there is sustained chaos for $\rho > \rho_c$ and transient chaos for $\rho < \rho_c$. We use a fourth order Runge–Kutta method with time step $\Delta t = 0.003$ for numerical integration of the Lorenz system.

Time series from four values of the bifurcation parameter: $\rho = 24.56$, 26.06, 27.56 and 29.06, all in the attractor regime, are used to train the reservoir machine, as indicated by the four vertical blue dashed lines in figure 5. The time step of the reservoir system is chosen to be $\Delta t = 0.015$ (quite arbitrarily insofar as it is small as compared with the average length of the oscillation cycles). The length of the training time series is $t = 400$ for each ρ value, and the length of the time interval for validation is $t = 8$. The 'warming up' length for the machine is $t = 0.1$. The optimal values of the hyperparameters are determined to be $D_r = 800$, $d = 490$, $\lambda = 1.78$, $k_{in} = 0.029$, $k_p = 0.052$, $p_0 = 2.99$, $\alpha = 0.40$, and $\beta = 6 \times 10^{-4}$.

Figure 6 presents typical results of transient chaos generated by the reservoir machine in comparison with those from the actual Lorenz system. In particular, figures 6(a1) and (a2) show two representative time series $x(t)$ of the actual Lorenz system in the attractor and transient regimes, respectively, for $\rho = 25.0 > \rho_c$ and $\rho = 23.0 < \rho_c$. The corresponding machine generated time series are shown in figures 6(b1) and (b2).

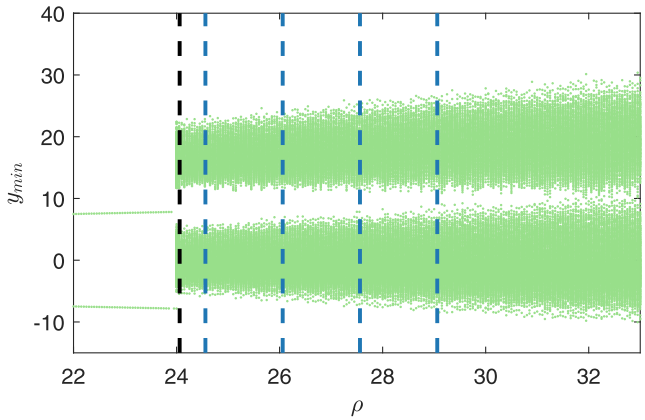

Figure 5. Bifurcation diagram of the Lorenz system. The vertical black dashed line denotes the crisis at $\rho = \rho_c = 24.06$. The vertical blue dashed lines indicate the training points of the reservoir computing machine: $\rho = 24.56, 26.06, 27.56$ and 29.06.

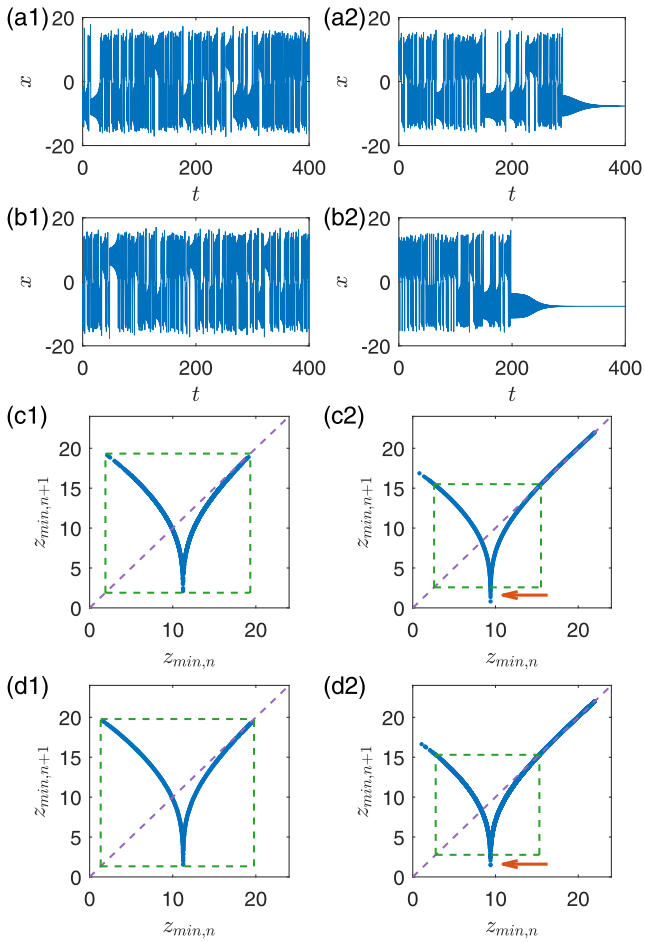

Figure 6. Emergence of transient chaos of the classic Lorenz system in machine learning. (a1) and (a2) Representative time series of the x component of the actual Lorenz system in the attractor and transient regimes, respectively, for $\rho = 25.0 > \rho_c$ and $\rho = 23.0 < \rho_c$. (b1) and (b2) Time series $x(t)$ generated by a reservoir machine for the same values of ρ as in (a1) and (a2), respectively. The machine generates the 'correct' transient chaos for $\rho < \rho_c$ in the sense that it is statistically indistinguishable from the true transient behavior. (c1) and (c2) Return maps constructed from the local minima of z from the true time series, where the green dashed square defines an interval that contains a chaotic attractor at $\rho = 25.0 > \rho_c$ (c1) or a nonattracting chaotic set for $\rho = 23.0 < \rho_c$ due to the escaping region (marked by the red arrow) leading to transient chaos (c2). (d1) and (d2) Machine generated return maps for the same values of ρ as in (c1) and (c2), respectively, which agree with the actual return maps.

It can be seen that, not only is the machine able to correctly generate the characteristically distinct behaviors in the pre-critical and post-critical regimes, but the statistical characteristics of the time series are also indistinguishable from the real ones. Figures 6(c1), (c2), (d1) and (d2) show the real return maps and those extracted from the machine generated time series, respectively, in the pre-critical and post-critical regimes,

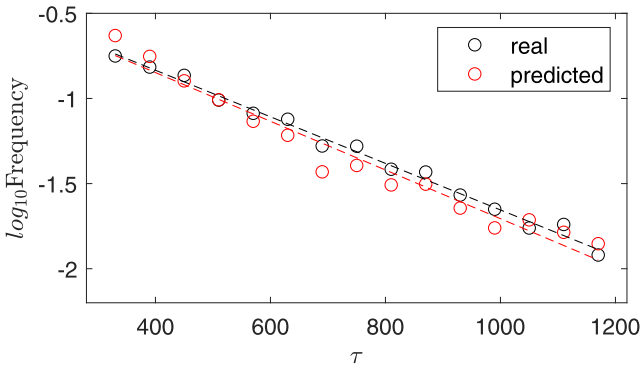

Figure 7. Comparison of a key statistical characteristic between the real and machine generated transient chaos. Shown is the transient lifetime distributions for $\rho = 23$ of the real transient (black) and reservoir computing generated transient chaos (red) obtained from 100 random reservoir machine realizations (200 different 'warming up' sequences for each realization).

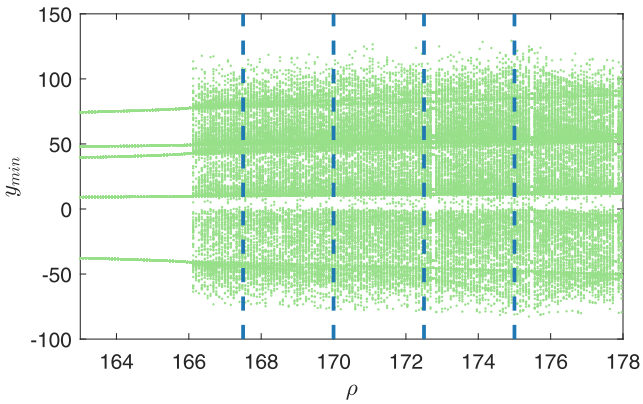

Figure 8. Bifurcation diagram of the classic Lorenz system in an intermittency regime. The vertical blue dashed lines indicate the training points: $\rho = 167.5, 170, 172.5$ and 175.

with reasonable agreement. Especially, in the pre-critical regime, the return map is fully contained in an invariant region in the phase space, whereas in the post-critical regime, a small escaping cusp emerges in the return map. The remarkable result is that the reservoir machine is able to generate these features faithfully.

The reservoir computing machine is also able to generate the correct exponential distribution of the transient lifetime, as shown in figure 7. The average transient lifetime is determined to be $\langle\tau\rangle \approx 3.0 \times 10^2$. Comparing with the true average lifetime $\langle\tau\rangle \approx 3.2 \times 10^2$, we see that the agreement is rather remarkable. The small error is the result of the small size of the escaping region in the return map, which is sensitive to random factors such as the accuracy of reservoir training.

3.3. Machine generated intermittency in the classic Lorenz system

Intermittency [31] can be regarded as a special type of transient behavior, where the system switches between two distinct states, spending a finite amount of time in each. We now demonstrate that our parameter-cognizant reservoir machine can generate intermittency in the chaotic Lorenz system in a parameter region different from the one studied in section 3.2. For $\sigma = 10$ and $\eta = 8/3$, an intermittency regime arises for larger values of ρ: about $\rho = 166$. As shown in figures 8 and 9, as ρ increases through a critical point (about 166), there is a transition from a periodic attractor to a chaotic attractor, where intermittency arises after the transition. Here we use a fourth order Runge–Kutta method with time step $\delta t = 0.002$ for numerical integration of the Lorenz system.

We use time series from four values of the bifurcation parameter: $\rho = 167.5$, 170, 172.5 and 175, all in a chaotic attractor regime that is relatively far from the intermittency regime, to train the reservoir machine, as indicated by the four vertical blue dashed lines in figure 8. The time step of the reservoir system is $\Delta t = 0.01$. For each ρ value, the length of the training time series is $t = 400$ and the length of the time interval for validation is $t = 2$. (Note that the validation length is shorter than that used in section 3.2 because the maximum Lyapunov exponent of the target system is larger in the present parameter region.) The 'warming up'

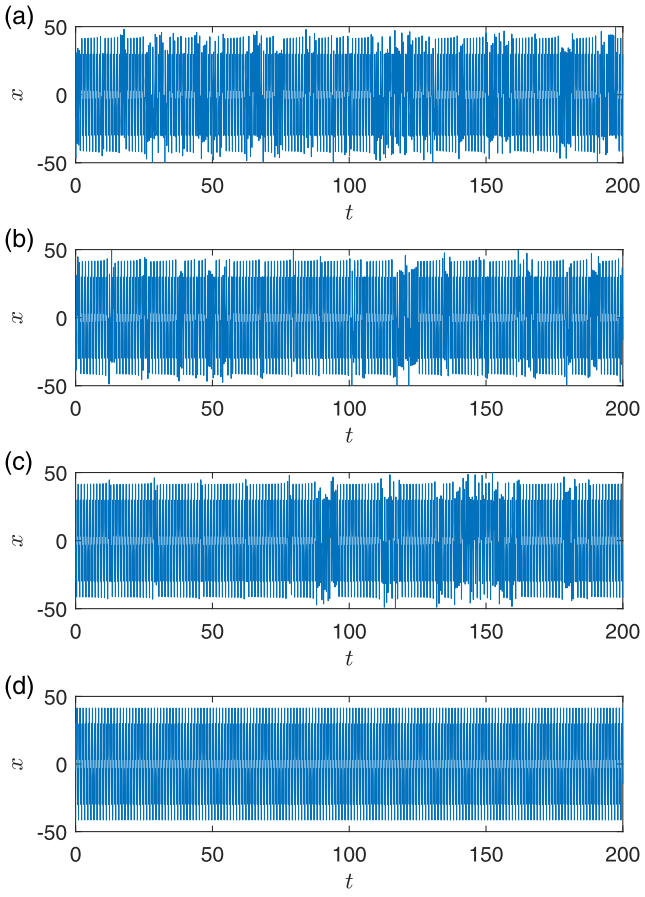

Figure 9. Intermittency in the classic Lorenz system. As the value of the bifurcation parameter ρ increases through a critical point $\rho_c \approx 166.05$, there is a transition from a periodic attractor to chaos, after which intermittency arises. (a)–(d) Representative intermittent time series for $\rho = 166.35$, $\rho = 166.25$, $\rho = 166.15$, and $\rho = 166.05$, respectively.

length for the machine is $t = 0.1$. The optimal values of the hyperparameters are $D_r = 800$, $d = 501$, $\lambda = 1.93$, $k_{in} = 0.0131$, $k_p = 0.0136$, $p_0 = -0.86$, $\alpha = 0.96$, and $\beta = 3.6 \times 10^{-4}$. Figure 10 shows that the reservoir computing machine is able to generate the intermittent behavior.

3.4. Machine generated transient chaos in the Ikeda map model

To demonstrate that our machine learning approach represents an advance over the traditional sparse optimization methods in certain scenarios, we present an example of the Ikeda map model for which the sparse optimization methods fail. In particular, the basic requirement of any sparse optimization technique for finding the system equations is *sparsity*: when the system equations are expanded into a power series or a Fourier series, it must be that only a few terms are present so that the coefficient vectors to be determined from data are sparse [32, 33].

The Ikeda map describes the dynamics of a laser pulse propagating in a nonlinear cavity, which is given by [34–36]:

$$z_{n+1} = \mu + \gamma z_n \exp\left(\mathrm{i}\kappa - \frac{\mathrm{i}\nu}{1 + |z_n|^2}\right), \tag{7}$$

where z is a complex dynamical variable, the bifurcation parameter μ is the dimensionless laser input amplitude, γ is the coefficient of the reflectivity of the partially reflecting mirrors of the cavity, κ is the laser empty cavity detuning, and ν measures the detuning due to the presence of a nonlinear medium in the cavity. It is very difficult, if not infeasible, to find a sparse representation of equation (7) with purely observational time series of the system. Thus the sparse optimization methods cannot deal with this system. As we will demonstrate below, our parameter-cognizant reservoir computing scheme can solve this problem.

We choose the values of the parameters in equation (7) to be $\gamma = 0.9$, $\kappa = 0.4$, and $\nu = 6.0$. The parameter μ is the bifurcation parameter. The system exhibits a boundary crisis [37] at $\mu = 1.0027$, as shown by the black vertical dashed line in figure 11. The dynamical behaviors for $\mu < \mu_c$ and $\mu > \mu_c$ are shown in figures 12(a1) and (a2), respectively. There is a chaotic attractor for $\mu < \mu_c$, and transient chaos leading to an escape of the system out of the previous operation region for $\mu > \mu_c$.

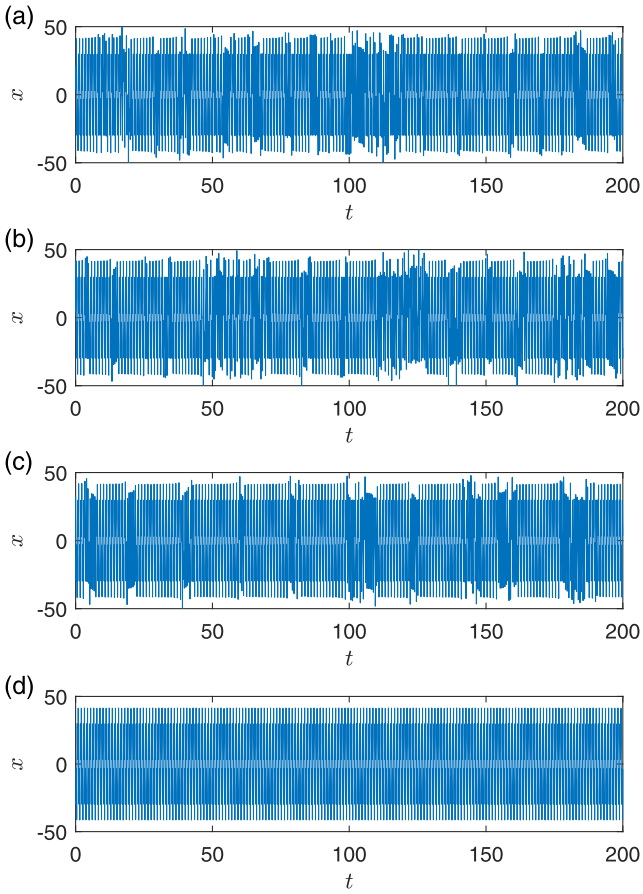

Figure 10. Machine generated intermittency in the classic Lorenz system. (a)–(d) Representative intermittent time series generated by the reservoir machine for $\rho = 166.30$, $\rho = 166.20$, $\rho = 166.10$, and $\rho = 166.00$, respectively. The machine gives the value of the transition point to be about $\rho_c = 166.00$, which agrees with the true point.

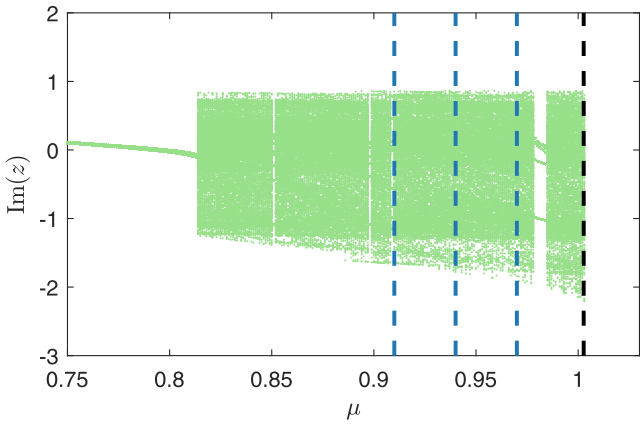

Figure 11. Bifurcation diagram of the Ikeda map system. The vertical black dashed line indicates the critical point $\mu_c = 1.0027$. The three vertical blue dashed lines specify the three values of the bifurcation parameter μ used for training the reservoir machine: $\mu = 0.91, 0.94, 0.97$.

The optimized hyperparameter values are $D_r = 400$, $d = 283$, $\lambda = 0.17$, $k_{in} = 2.6$, $k_p = 0.35$, $p_0 = 0.47$, $\alpha = 1.0$ and $\beta = 1 \times 10^{-6}$. For each selected value of μ, the training and validation lengths are $t_{train} = 800$ steps and $t_{validating} = 15$ steps, respectively. During validation, the reservoir system is able to predict the system evolution for more than 5 Lyapunov times with small relative error.

We train the reservoir machine at $\mu = 0.91, 0.94, 0.97$. Their values are shown in figure 11 by the three vertical blue dashed lines, and they are all in the chaotic attractor regime. After training, we apply parameter change $\Delta\mu$ and test if the reservoir computing machine generates transient chaos. As shown by an exemplary pair of machine generated time series, in figures 12(b1) and (b2), our reservoir approach can successfully

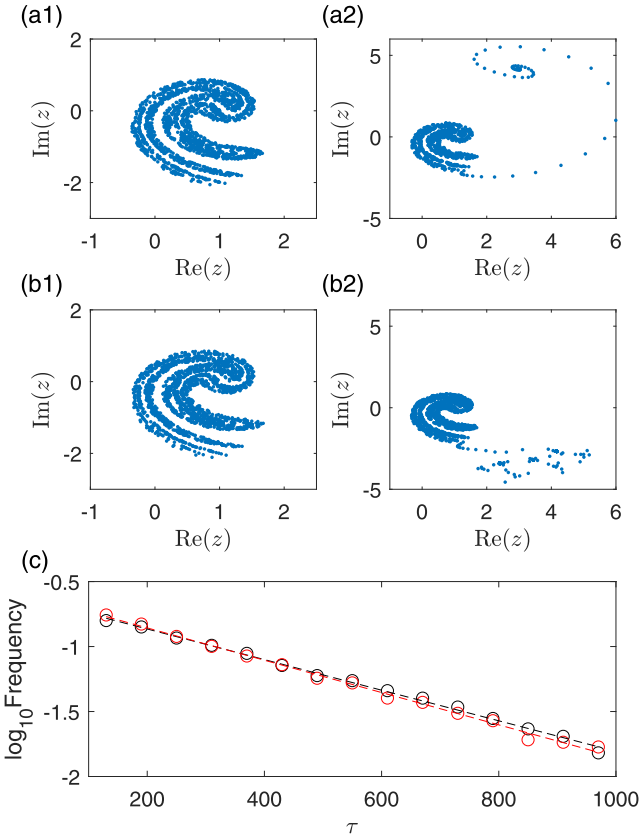

Figure 12. Predicting transient chaos and collapse of dynamics in a nonlinear optical cavity as described by the Ikeda map. (a1) and (a2) Typical behaviors of the system in the sustained and transient chaos regimes, respectively, for $\mu = 0.99 < \mu_c$ and $\mu = 1.01 > \mu_c$. (b1) and (b2) Predicted behaviors by the reservoir machine for the same values of μ as in (a1) and (a2), respectively. The machine predicts correctly the system collapse for $\mu > \mu_c$. (c) Actual (black) and predicted (red) exponential transient lifetime distributions for $\mu = \mu_c + 0.02$.

generate transient chaos in the transient regime, even though it was trained only with data from a chaotic attractor.

To demonstrate the statistical characteristics of the generated transient dynamics, we compare the transient lifetime distribution of both the reservoir generated time series and the time series calculated from equation (7). We set the control parameter as $\mu = \mu_c^* + 0.02$ for the reservoirs, where μ_c^* is the critical point calculated from each realization of reservoir machine. In total, 50 stochastic realizations of the reservoir system and 400 random initial conditions for each realization are used, and we record the transient lifetimes of these 20 000 trials. The result is shown in figure 12(c), where the distribution of the reservoir computing (marked in red) is very close to the distribution of the real system with $\mu = \mu_c + 0.02$ (marked in black), demonstrating the power of our reservoir approach for generating transient chaos.

4. Limited performance analysis of reservoir computing with dynamic climate control

A comprehensive performance analysis of our reservoir computing scheme in terms of its ability to generate transient chaos is infeasible, as there are a large number of 'free' parameters in the system. Here we carry out a limited analysis based of the logistic map, focusing on two issues: the selection of the training parameter points and the effect of noise.

4.1. Dependency on training points

We study the dependency of reservoir machine's performance on the values of the training bifurcation parameters. For convenience, we use the parameter difference L between the training parameter value and the critical point to measure how 'far' the former is from the latter. For the logistic map, we use $\max\{a_{\text{train}}\}$ to denote the largest value of the bifurcation parameter a used for training (so $L = a_c - \max\{a_{\text{train}}\}$) and fix the relative positions of the training points. This way, when there is a small change in the value of L the training values of the bifurcation parameter are shifted by the same amount. During this process, all values of the hyperparameters of the reservoir machine are kept fixed.

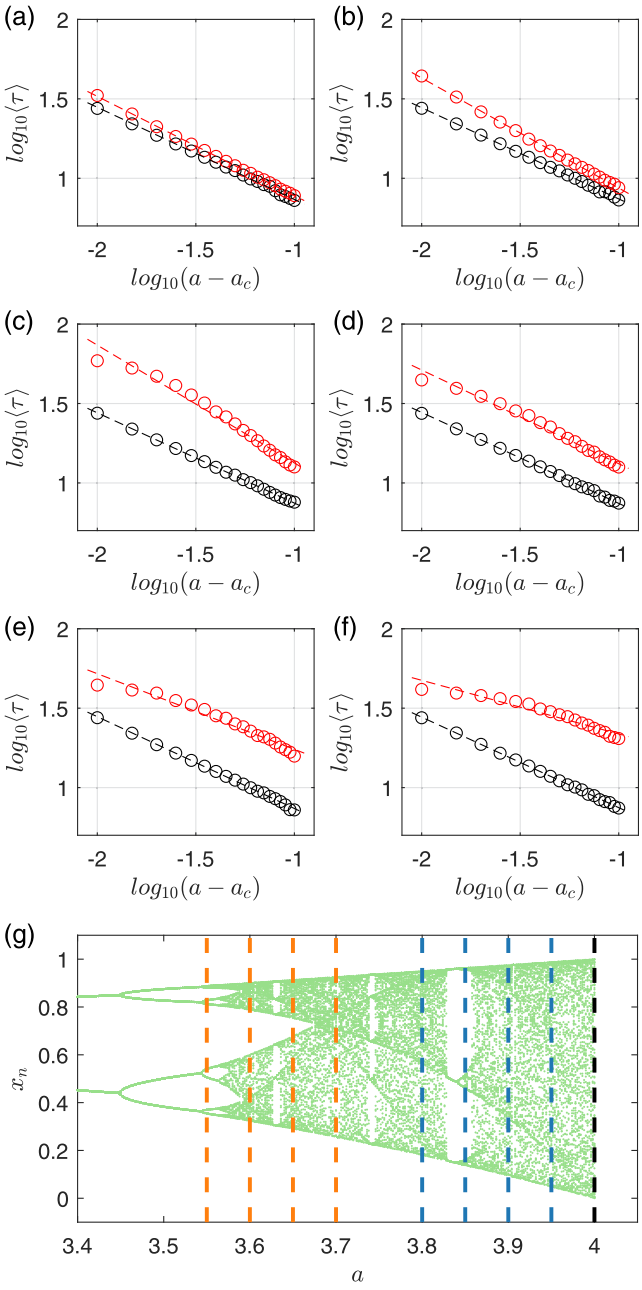

Figure 13. Effect of the distances between the training parameter values and the critical point. (a)–(f) Machine generated (red) scaling law of the average transient lifetime with $(a - a_c)$ on a log-log scale for $L = 0.05, 0.1, 0.15, 0.2, 0.25$ and 0.3, respectively. The corresponding actual scaling laws are shown in black. The discrepancy increases with L as expected, but the average transient lifetime generated by the machine agrees with the true value within the same order of magnitude. (g) Positions of the training points in (a) with $L = 0.05$ (the blue dashed lines) and training points in (f) with $L = 0.3$ (the orange dashed lines). All other training points fall in-between these two sets of values. The black dashed line denotes the position of the critical point $a_c = 4.0$.

The results are shown in figure 13, where panels (a)–(f) show the machine generated scaling law of the average transient lifetime $\langle\tau\rangle$ with $(a - a_c)$ for $L = 0.05, 0.1, 0.15, 0.2, 0.25$ and 0.3. The positions of the training points with largest and smallest L values are shown in figure 13(g). It can be seen that, the error in the machine generated scaling law grows with L, with the estimated average transient lifetime slightly longer than the actual value. As the training set moves further leftward from the critical point, the error in the machine estimated critical point a_c increases, leading to an apparent deviation of the scaling law from being algebraic. However, the reservoirs are still able to generate transient chaos even when the training points are away from the transient region. And there is no order of magnitude difference in the machine generated and the actual average transient lifetime.

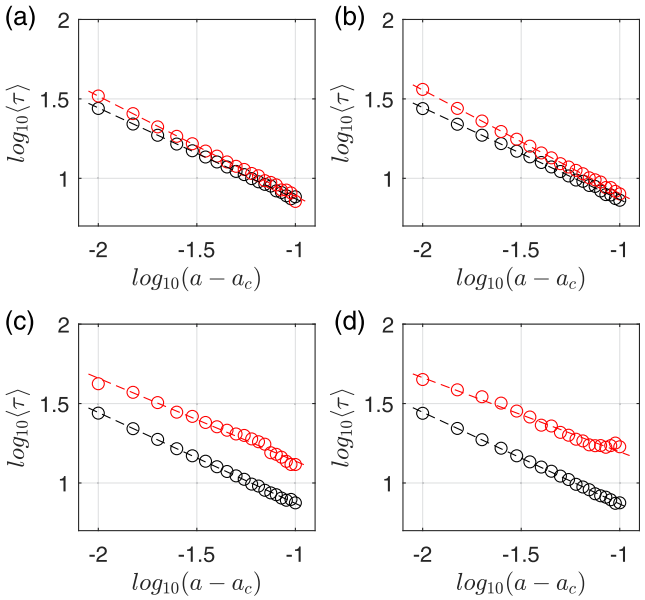

Figure 14. Impact of observational noise on reservoir machine's ability to generate accurate transient chaos. Shown are the machine generated (red) and real (black) scaling law of the average transient lifetime $\langle\tau\rangle$ with $(a - a_c)$. The training data for the reservoir machines in each panel are contaminated by different levels of noise: (a) $\sigma = 0$, (b) $\sigma = 10^{-3.5}$, (c) $\sigma = 10^{-2.5}$, and (d) $\sigma = 10^{-1.5}$. All the distributions are obtained from 1000 independent random reservoir machine realizations, each with 1000 different random 'warming up' sequences.

4.2. Effect of noise

We investigate the impact of observational noise on reservoir machine's ability to replicate transient chaos. The noises are of the Gaussian type of amplitude σ and they are added directly to the training data set. Figures 14(a)–(d) show the scaling of the average lifetime $\langle\tau\rangle$ with $(a - a_c)$ for four different values of σ: 0, $10^{-3.5}$, $10^{-2.5}$, and $10^{-1.5}$, respectively. As the noise become stronger, the machine generated scaling law begins to deviate from the actual one, as expected. The remarkable feature is that the scaling exponent (the slope of the linear fitting on the log–log plot) stays close to the real one, even for relatively large noise amplitude, providing support for the robustness of the performance of the reservoir machine in generating the 'correct' transient chaotic behaviors.

5. Discussion

Transient chaos is ubiquitous in nonlinear dynamical systems [38]. Here we demonstrate the emergence of transient chaos in machine learning. In particular, by focusing on reservoir computing, a class of RNNs with simple structure, we find that it can generate transient chaos with statistical behaviors that match those of the target system. The training (supervised learning) process makes use of time series data taken from a small number of distinct values of the bifurcation parameter of the target system, during which the machine is an open dynamical system. These training parameter values are 'recorded' by the machine through a particular input channel to the reservoir network. Training is deemed successful when the reservoir machine is able to generate trajectories that stay close to the true trajectories for some reasonable amount of time for each of the training parameter values. After the training, the open loop in the reservoir system is closed and it becomes a self-evolving nonlinear dynamical system. Our main point is that this system can generate transient chaos whose statistical behaviors mimic those of the target system for the same value of the bifurcation parameter. It is worth emphasizing that training is done completely in the attractor regime, and the reservoir system has never been exposed to any transient chaotic behavior. Yet the training has instilled the dynamical 'climate' of the target system into the machine and it gains the ability to generate different dynamical behaviors for different values of the bifurcation parameter, even in the regime of transient chaos, where the dynamics are characteristically distinct from those in the attractor regime.

A basic statistical characteristic of transient chaos is the distribution of the transient lifetime, which is exponential in dissipative dynamical systems. We have demonstrated that an adequately trained machine can faithfully generate this exponential distribution, with the average transient lifetime (the inverse of the exponential rate) agreeing with the actual value but only to within the same order of magnitude. To reduce this error remains a challenge, as it often depends sensitively on the details of the dynamical structure responsible

for transient chaos such as the returned map. Especially, in the regime of attracting chaos, there is a region in the return map that is invariant. However, in the regime of transient chaos, an escaping gap emerges in the region, which leads to chaotic transients. The statistical characteristics of the resulting transient chaos depend sensitively on the details of the escaping region, such as its size. While the reservoir machine is able to generate the return map that agrees qualitatively with the true map, there can often be small discrepancies in the detail, especially those around the escaping gap, which can lead to a sizable difference in the average lifetime of the machine generated transient chaos from the actual lifetime.

Another characteristic of transient is the scaling relation between the average lifetime and the parameter difference from the critical point. We have demonstrated that our properly trained reservoir machine can faithfully generate this scaling law even in the presence of observational noise, which is defined entirely in the regime of transient chaos to which the machine has never been exposed.

Taken together, for a nonlinear dynamical system of interest, to develop a machine learning system that is capable of generating transient chaos beyond the attractor regime, where training takes place, has implications to predicting the future dynamical state of the target system [23].

Acknowledgments

This work was supported by ONR under Grant No. N00014-21-1-2323.

Data availability statement

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

ORCID iDs

Ling-Wei Kong https://orcid.org/0000-0002-8921-1642
Huawei Fan https://orcid.org/0000-0001-7703-0185
Celso Grebogi https://orcid.org/0000-0002-9811-4617
Ying-Cheng Lai https://orcid.org/0000-0002-0723-733X

References

[1] Haynes N D, Soriano M C, Rosin D P, Fischer I and Gauthier D J 2015 *Phys. Rev.* E **91** 020801
[2] Larger L, Baylón-Fuentes A, Martinenghi R, Udaltsov V S, Chembo Y K and Jacquot M 2017 *Phys. Rev.* X **7** 011015
[3] Pathak J, Lu Z, Hunt B R, Girvan M and Ott E 2017 *Chaos* **27** 121102
[4] Lu Z, Pathak J, Hunt B, Girvan M, Brockett R and Ott E 2017 *Chaos* **27** 041102
[5] Duriez T, Brunton S L and Noack B R 2017 *Machine Learning Control–Taming Nonlinear Dynamics and Turbulence* (Berlin: Springer)
[6] Lu Z, Hunt B R and Ott E 2018 *Chaos* **28** 061104
[7] Pathak J, Wikner A, Fussell R, Chandra S, Hunt B R, Girvan M and Ott E 2018 *Chaos* **28** 041101
[8] Pathak J, Hunt B, Girvan M, Lu Z and Ott E 2018 *Phys. Rev. Lett.* **120** 024102
[9] Carroll T L 2018 *Phys. Rev.* E **98** 052209
[10] Nakai K and Saiki Y 2018 *Phys. Rev.* E **98** 023111
[11] Roland Z S and Parlitz U 2018 *Chaos* **28** 043118
[12] Weng T, Yang H, Gu C, Zhang J and Small M 2019 *Phys. Rev.* E **99** 042203
[13] Griffith A, Pomerance A and Gauthier D J 2019 *Chaos* **29** 123108
[14] Jiang J and Lai Y C 2019 *Phys. Rev. Res.* **1** 033056
[15] Vlachas P R, Pathak J, Hunt B R, Sapsis T P, Girvan M, Ott E and Koumoutsakos P 2019 arXiv:1910.05266
[16] Fan H, Jiang J, Zhang C, Wang X and Lai Y C 2020 *Phys. Rev. Res.* **2** 012080
[17] Zhang C, Jiang J, Qu S-X and Lai Y-C 2020 *Chaos* **30** 083114
[18] Kuptsov P V, Kuptsova A V and Stankevich N V 2021 Artificial neural network as a universal model of nonlinear dynamical systems *Russ. J. Nonlinear Dyn.* **17** 5–21
[19] Jaeger H 2001 *Technical Report* vol 148 German National Research Center for Information Technology GMD p 13
[20] Mass W, Nachtschlaeger T and Markram H 2002 *Neural Comput.* **14** 2531–60
[21] Jaeger H and Haas H 2004 *Science* **304** 78–80
[22] Manjunath G and Jaeger H 2013 *Neural Comput.* **25** 671–96
[23] Kong L W, Fan H W, Grebogi C and Lai Y C 2021 *Phys. Rev. Res.* **3** 013090
[24] Feldkamp L A, Puskorius G V and Moore P C 1997 *Inf. Sci.* **98** 217–35
[25] Patel D, Canaday D, Girvan M, Pomerance A and Ott E 2021 *Chaos* **31** 033149
[26] Kim J Z, Lu Z, Nozari E, Pappas G J and Bassett D S 2021 *Nat. Mach. Intell.* **3** 316–23
[27] Klos C, Kossio Y F K, Goedeke S, Gilra A and Memmesheimer R M 2020 *Phys. Rev. Lett.* **125** 088103
[28] The codes of this work are shared at github.com/lw-kong/Reservoir_with_a_Parameter_Channel_JPC2021
[29] Grebogi C, Ott E and Yorke J A 1982 *Phys. Rev. Lett.* **48** 1507

[30] Lorenz E N 1963 *J. Atmos. Sci.* **20** 130–41
[31] Pomeau Y and Manneville P 1980 *Commun. Math. Phys.* **74** 189–97
[32] Wang W-X, Yang R, Lai Y-C, Kovanis V and Grebogi C 2011 *Phys. Rev. Lett.* **106** 154101
[33] Wang W-X, Lai Y-C and Grebogi C 2016 *Phys. Rep.* **644** 1–76
[34] Ikeda K 1979 *Opt. Commun.* **30** 257–61
[35] Ikeda K, Daido H and Akimoto O 1980 *Phys. Rev. Lett.* **45** 709–12
[36] Hammel S M, Moloney J V and Jones C K R T 1985 *J. Opt. Soc. Am.* B **2** 552–64
[37] In V, Spano M L and Ding M 1998 *Phys. Rev. Lett.* **80** 700
[38] Lai Y C and Tél T 2011 *Transient Chaos: Complex Dynamics on Finite Time Scales* (New York: Springer)