Chaos
An Interdisciplinary Journal of Nonlinear Science

RESEARCH ARTICLE | MARCH 05 2024

Deep learning-based state prediction of the Lorenz system with control parameters

Xiaolong Wang ; Jing Feng ; Yong Xu ; Jürgen Kurths

Chaos 34, 033108 (2024)
https://doi.org/10.1063/5.0187866

Deep learning-based state prediction of the Lorenz system with control parameters

Cite as: Chaos 34, 033108 (2024); doi: 10.1063/5.0187866
Submitted: 16 November 2023 · Accepted: 15 February 2024 ·
Published Online: 5 March 2024

Xiaolong Wang,[1,2] Jing Feng,[3] Yong Xu,[2,4,a)] and Jürgen Kurths[5,6]

AFFILIATIONS

[1]School of Mathematics and Statistics, Shaanxi Normal University, Xi'an 710119, China
[2]School of Mathematics and Statistics, Northwestern Polytechnical University, Xi'an 710072, China
[3]School of Science, Xi'an University of Posts & Telecommunications, Xi'an 710121, China
[4]MOE Key Laboratory for Complexity Science in Aerospace, Northwestern Polytechnical University, Xi'an 710072, China
[5]Potsdam Institute for Climate Impact Research, Potsdam 14412, Germany
[6]Department of Physics, Humboldt University Berlin, Berlin 12489, Germany

[a)]**Author to whom correspondence should be addressed:** hsux3@nwpu.edu.cn

ABSTRACT

Nonlinear dynamical systems with control parameters may not be well modeled by shallow neural networks. In this paper, the stable fixed-point solutions, periodic and chaotic solutions of the parameter-dependent Lorenz system are learned simultaneously via a very deep neural network. The proposed deep learning model consists of a large number of identical linear layers, which provide excellent nonlinear mapping capability. Residual connections are applied to ease the flow of information and a large training dataset is further utilized. Extensive numerical results show that the chaotic solutions can be accurately forecasted for several Lyapunov times and long-term predictions are achieved for periodic solutions. Additionally, the dynamical characteristics such as bifurcation diagrams and largest Lyapunov exponents can be well recovered from the learned solutions. Finally, the principal factors contributing to the high prediction accuracy are discussed.

Deep neural networks have been recognized as innovative and powerful tools to develop predictive models for complex nonlinear systems. The approaches are able to learn sophisticated nonlinear relationships by stacking simple affine mappings, bypassing the involved domain knowledge and mathematical treatment. In this study, a neural network-based modeling approach is proposed to predict parameter-dependent chaotic dynamics with high accuracy, which is an appropriate modeling for various real-world systems. We verify its efficiency by considering the classical chaotic Lorenz system with three control parameters. As it will be shown below, our method gives an impressive prediction accuracy. The key factors which may affect the capability of the deep learning-based method are also explored via simulations. We discuss the potential, but also the limitations of the proposed approach, which can provide guidance for future research on the prediction of highly complex systems and nonlinear processes.

I. INTRODUCTION

Nonlinear dynamical systems have been utilized in many fields of natural and social sciences, such as epidemiology, aerodynamics, neurodynamics, and finance.[1–5] In real applications, it is often difficult to obtain *a priori* mathematical model. The law of dynamics has to be inferred from data, recently often resorting to machine learning methods.[6–8] Among these methods, deep learning-based approaches[9–14] have shown outstanding performance, where deep neural networks with rather simple building blocks are constructed to model complex mappings.

Not surprisingly, the interplay between deep learning and nonlinear dynamics has become a research hotspot. Different deep neural networks have been successfully applied to classify periodic and chaotic time series,[15–19] to recognize time series of different dynamical systems[20,21] as well as identifying dynamical changes of chaotic systems.[22] In this interdisciplinary research field, the state prediction of dynamical systems from data is an important yet challenging task, especially for chaotic systems. An accurate state prediction is not only practically useful for foreseeing time series and calculating dynamical characteristics such as the largest Lyapunov exponent (LLE),[23,24] but also a basis for a better control or synchronization of the considered system.[25]

Till now, several popular deep learning architectures have been applied in modeling chaotic dynamical systems. A fully connected neural network (FCNN) with four hidden layers[26] has

been applied to predict the states of discrete maps and continuous systems from several recent states. Reservoir computers,[27–29] recurrent neural networks,[30] long short-term memory neural networks,[3,31–33] and autoencoders[34] have been adopted to perform one-step prediction of nonlinear dynamical systems. Multi-step prediction has been achieved by recursively using the obtained neural networks. These studies reveal the power of deep learning in modeling complex systems.

However, the previous studies of state prediction or trajectory generation often assume that the control parameters of the dynamical systems are fixed.[3,24,27–29,31,32] Separate neural networks have to be trained to learn parameter-dependent systems, which is inconvenient and computation-intensive. Though the existing neural networks[26] can handle one to three control parameters, the prediction accuracy is limited. When the control parameters of a nonlinear system vary, the dynamics of the solutions may change dramatically. For instance, the well-known Lorenz system[35] has stable fixed-point, chaotic or multi-periodic solutions.[36] The complex and rich phenomena of a parameter-dependent system often make it difficult to achieve a highly accurate model by using shallow neural networks with limited nonlinear mapping abilities. Compared to shallow neural networks, deep neural networks can represent abstract functions more efficiently[37] and implement functions of higher complexity with the same amount of resources.[38] Therefore, it is instructive to investigate a general deep neural network for the prediction of parameter-dependent dynamical systems as well as exploring possible guidelines to build capable methods.

In order to forecast parameter-dependent systems such as the Lorenz system, the learning model should not only be capable of encoding the highly complex dynamics, but also be properly trained. The main contribution of our work is to numerically verify that one can obtain a promising neural network by simply stacking a large number of identical linear layers and training it on a massive dataset. The residual connections used by the ResNet[18,20,39] are also indispensable to alleviate the inefficiency of information transfer across shallow and deep layers. These connections directly link every few layers of the neural network, serving as highways for the flow of gradient information. They have made the ResNet achieve the best performance in classifying time series.[40] Extensive experiments will be conducted to show the capability of our method. The key factors for an accurate state prediction will be scrutinized and the limitations will also be discussed.

The rest of the paper is organized as follows. Section II introduces the deep learning model for state prediction. Section III evaluates the accuracy of one-step and multi-step predictions. Ablation studies of the network architectures and the training datasets are presented in Sec. IV, and Sec. V concludes the work.

II. THE METHODOLOGY

A. Problem definition

We consider the autonomous dynamical system described by the ordinary differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t); \boldsymbol{\Theta}), \mathbf{x}(0) = \mathbf{x}_0, \tag{1}$$

where $\mathbf{x}(t)$ is the vector or scalar state at time t and $\mathbf{F}$ is a vector field parameterized by the control parameters $\boldsymbol{\Theta}$. The dot over variables denotes differentiation with respect to time t. The solution of the system equation (1) at time t with the initial state $\mathbf{x}_0$ is denoted by $\phi_t(\mathbf{x}_0; \boldsymbol{\Theta})$. It satisfies the following group property:

$$\phi_{t+s}(\mathbf{x}_0; \boldsymbol{\Theta}) = \phi_t(\phi_s(\mathbf{x}_0; \boldsymbol{\Theta}); \boldsymbol{\Theta}), \tag{2}$$

which implies that a long-term state prediction could be partitioned into several consecutive short-term predictions.

For the initial state $\mathbf{x}_0$ and parameters $\boldsymbol{\Theta}$, if the system equation (1) is explicitly known, the map $\phi_{\Delta t}(\mathbf{x}_0; \boldsymbol{\Theta})$ that predicts the state in Δt later can be obtained traditionally by finite difference methods such as the fourth-order Runge–Kutta (RK4) method. A high accuracy could be achieved by using a low enough time increment h. By using Eq. (2), the k-step prediction of the state in $k\Delta t$ later can be calculated by recursively utilizing the map $\phi_{\Delta t}$, i.e.,

$$\phi_{k\Delta t}(\mathbf{x}_0; \boldsymbol{\Theta}) = \phi^k_{\Delta t}(\mathbf{x}_0; \boldsymbol{\Theta}).$$

Therefore, learning the solutions of Eq. (1) can be simplified to learn the function $\phi_{\Delta t}(\mathbf{x}; \boldsymbol{\Theta})$ for all possible parameters $\boldsymbol{\Theta}$ and states $\mathbf{x} \in \{\phi_t(\mathbf{x}_0; \boldsymbol{\Theta}) | t \geq 0\}$ with a fixed small time step Δt.

In this paper, we assume that the system equation (1) is unknown. The one-step prediction function $\phi_{\Delta t}(\cdot; \cdot)$ is modeled by a neural network $\phi_{\mathrm{NN}}(\cdot; \cdot)$, which is learned from samples on solution trajectories with different control parameters. A sample is defined as a triple $(\mathbf{x}, \boldsymbol{\Theta}, \mathbf{y})$, which includes the state $\mathbf{x}$ at the previous time step, the control parameters $\boldsymbol{\Theta}$ and the one-step true prediction $\mathbf{y} = \phi_{\Delta t}(\mathbf{x}; \boldsymbol{\Theta})$. The input of ϕ_{NN} is the vector $[\mathbf{x}^{\mathrm{T}}, \boldsymbol{\Theta}^{\mathrm{T}}]^{\mathrm{T}}$ and the output is the learned prediction $\hat{\mathbf{y}} = \phi_{\mathrm{NN}}(\mathbf{x}; \boldsymbol{\Theta})$ expected to be close to $\mathbf{y}$. Once the neural network ϕ_{NN} is obtained, the long-term prediction of $\phi_t(\mathbf{x}_0; \boldsymbol{\Theta})$ initialized at $\mathbf{x}_0$ is approximated by recursively applying the neural network $\lfloor t/\Delta t \rfloor$ times, i.e., $\phi^{\lfloor t/\Delta t \rfloor}_{\mathrm{NN}}(\mathbf{x}_0; \boldsymbol{\Theta})$, where $\lfloor \cdot \rfloor$ is the round down function.

B. The Lorenz system

In the following, we apply the state prediction neural network ϕ_{NN} on the paradigmatic Lorenz system, which is an autonomous dynamical system with the vector field

$$\mathbf{F}(\mathbf{x}; \boldsymbol{\Theta}) = \begin{bmatrix} a(y - x) \\ bx - xz - y \\ xy + cz \end{bmatrix},$$

where $\mathbf{x} = [x, y, z]^{\mathrm{T}}$ is the three-dimensional state and $\boldsymbol{\Theta} = [a, b, c]^{\mathrm{T}}$ includes the three control parameters. Accordingly, the input of the neural network ϕ_{NN} is six dimensional and the output is three dimensional. The time step Δt is fixed[27] to 0.02.

The solutions of the Lorenz system may reach asymptotically to stable fixed points, stable limit cycles or chaotic attractors,[41] depending on the parameters $\boldsymbol{\Theta}$. The dynamical behaviors of the solutions are indicated concisely by the LLE. The LLE is defined in terms of the length of the longest ellipsoidal principal axis $p_1(t)$ of the ellipsoid evolved from an infinitesimal sphere of initial states,[42]

$$\lambda_1 = \lim_{t \to \infty} \frac{1}{t} \ln \frac{p_1(t)}{p_1(0)}. \tag{3}$$

It measures the average rate of exponential divergence between adjacent solution trajectories.[43] The LLE will be utilized in Sec. III as a

quantitative metric to compare the learned solutions generated from ϕ_{NN} and the true solutions.

C. Neural network architecture

The Δt-time prediction $\phi_{\Delta t}(\mathbf{x};\boldsymbol{\Theta})$ is a highly nonlinear and complex function, requiring a capable neural network to learn. In general, a neural network is a multivariable function that cascades several layers of neurons. The nonlinear mapping capability of a neural network increases with the number of layers and the number of neurons in each layer.[44] We mainly explore the first more effective idea and briefly discuss the second idea in Sec. IV.

As shown in Fig. 1, the proposed deep neural network ϕ_{NN} cascades L_{RB} identical sub-networks named residual blocks (RBs),[39] which are mainly small traditional neural networks with extra connections named residual connections. If the input vector of the ith residual block is $\mathbf{I}_i$, then the output $\mathbf{O}_i$ is the vector addition of the main part $\mathbf{M}_i$ and the residual connection $\mathbf{R}_i$,

$$\begin{cases} \mathbf{O}_i = \mathbf{M}_i(\mathbf{I}_i) + \mathbf{R}_i(\mathbf{I}_i), & i = 1,\ldots,L_{\mathrm{RB}}, \\ \mathbf{I}_i = \mathbf{O}_{i-1}, & i = 2,\ldots,L_{\mathrm{RB}}, \\ \mathbf{I}_i = [\mathbf{x}^{\mathrm{T}}, \boldsymbol{\Theta}^{\mathrm{T}}]^{\mathrm{T}}, & i = 1. \end{cases} \tag{4}$$

The main part $\mathbf{M}_i$ of the ith residual block is defined by

$$\mathbf{M}_i(\mathbf{I}_i) = f_{\mathrm{ELU}}(f_{\mathrm{LN},i,3}(f_{\mathrm{ELU}}(f_{\mathrm{LN},i,2}(f_{\mathrm{ELU}}(f_{\mathrm{LN},i,1}(\mathbf{I}_i)))))).$$

It stacks a linear (LN) layer followed by the exponential linear unit (ELU)[45] activation function for three times to extract deep features from the input vector. The linear layer $f_{\mathrm{LN},i,j}$ is defined by the affine function

$$f_{\mathrm{LN},i,j}(\mathbf{x}) = \mathbf{W}_{ij}\mathbf{x} + \mathbf{b}_{ij}, \quad j = 1,2,3,$$

with the weight matrix $\mathbf{W}_{ij}$ and the bias vector $\mathbf{b}_{ij}$. f_{ELU} is the elementwise activation function

$$f_{\mathrm{ELU}}(x) = \begin{cases} x, & x > 0, \\ e^x - 1, & x \le 0, \end{cases}$$

which provides nonlinearity to the neural network.[46] In Fig. 1, an (m,n)-linear layer represents that the lengths of its input and output vectors are m and n, respectively. For simplicity, we set all linear layers in the main parts $\{\mathbf{M}_i\}_{i=1}^{L_{\mathrm{RB}}}$ to the type $(128,128)$.

The residual connection $\mathbf{R}_i$ of the i-th residual block is defined by

$$\mathbf{R}_i(\mathbf{I}_i) = \begin{cases} \mathbf{W}_1\mathbf{I}_1 + \mathbf{b}_1, & i = 1, \\ \mathbf{I}_i, & i = 2,\ldots,L_{\mathrm{RB}}, \end{cases} \tag{5}$$

serving as a shortcut to send the input feature directly to the output side. Hence the main part of the residual block is only responsible for fine-tuning the increment between the input $\mathbf{I}_i$ and output $\mathbf{O}_i$. The only exception is for the first residual connection, where a $(6,128)$-linear layer with the weighting matrix $\mathbf{W}_1$ and the bias vector $\mathbf{b}_1$ is required to adjust the vector shape. Therefore, the two parts $\mathbf{M}_1$ and $\mathbf{R}_1$ have the same length 128 and the vector addition in Eq. (4) can be carried out.

Finally, a $(128,3)$-linear layer with the weighting matrix $\mathbf{W}_2$ and the bias vector $\mathbf{b}_2$ is required after the last residual block to map the obtained deep features to the one-step state prediction $\hat{\mathbf{y}}$. As a result, the tunable parameters of the neural network are all distributed in the $3L_{\mathrm{RB}}+2$ linear layers, namely, $\{\mathbf{W}_1,\mathbf{b}_1,\mathbf{W}_2,\mathbf{b}_2\} \cup \{\mathbf{W}_{ij},\mathbf{b}_{ij}\}_{i=1,\ldots,L_{\mathrm{RB}},j=1,2,3}$. We choose $L_{\mathrm{RB}} = 17$ such that there are more than 50 linear layers. It is worth noting that though the classic design of three linear layers is used in each residual block,[39] a preliminary experiment shows that cascading two linear layers in each residual block achieves a comparable prediction accuracy.

Compared with the traditional deep neural networks,[26] the main improvement of our method is the adoption of the residual connections $\{\mathbf{R}_i\}_{i=1}^{L_{\mathrm{RB}}}$ such that very deep neural networks can be effectively trained. The residual connections can greatly accelerate the learning speed of the neural network as well as considerably lowering the training loss,[47] as the gradient information can flow backward easily without passing through the nonlinear activation functions. Compared with the previous shallow ResNets[18,20] which include less than 10 layers, a key improvement of our method is to utilize much more residual blocks to provide outstanding nonlinear mapping capability. The residual connections ensure that our very deep neural networks with excellent nonlinear mapping capability can be brought into full play.

D. Data acquisition

A deep neural network with huge tunable parameters often requires a large number of data in the training stage. Therefore, we intend to generate a large training dataset that consists of one-step predictions of the Lorenz system with random control parameters and initial states. The ranges of the control parameters are chosen in the 3D cuboid

$$\mathscr{P} = \{(a,b,c) | a \in [10,26], b \in [35,55], c \in [-6.5,-0.5]\}, \tag{6}$$

which includes chaotic and multiple periodic solutions as well as stable fixed-point solutions. The training dataset includes 200 million samples and the validation set includes another 2 million samples.

To generate a sample $(\mathbf{x},\boldsymbol{\Theta},\mathbf{y})$ for training and test, the control parameters $\boldsymbol{\Theta}$ are first uniformly sampled in $\mathscr{P}$. The states $\mathbf{x}$ and $\mathbf{y}$ apart by $\Delta t = 0.02$ are then randomly selected near the corresponding attractor. In detail, an initial state $\mathbf{x}_0 = [\xi_1,\xi_2,|\xi_3|]^{\mathrm{T}}$ is randomly sampled where $\xi_i, i = 1,2,3$ are independently drawn from the centered Gaussian distribution $\mathscr{N}(0,10\,000)$ with variance 10 000. Then, a solution trajectory in $t \in [0,300]$ governed by $\boldsymbol{\Theta}$ is simulated by the RK4 method with the time increment $h = 0.001$ and the initial state $\mathbf{x}_0$. The sample points in $t \in [0,75)$ are not considered, since they are likely to be transient or distant from the attractor. $\mathbf{x} = \phi_t(\mathbf{x}_0;\boldsymbol{\Theta})$ and $\mathbf{y} = \phi_{t+\Delta t}(\mathbf{x}_0;\boldsymbol{\Theta})$ are chosen from the solution trajectory where t is uniformly sampled from the time range $[75, 300-\Delta t]$.

This sampling strategy of one point per trajectory (OPPT) maximizes the variability of the training dataset, because each sample has uniformly generated unique parameters $\boldsymbol{\Theta}$ as well as a state $\mathbf{x}$ randomly distributed across the attractor. The samples are generated by CUDA parallel programming such that the generation could be finished in a day. If only a limited number of solution trajectories is available, sampling multiple points on each trajectory also works well, which will be shown in Sec. IV.

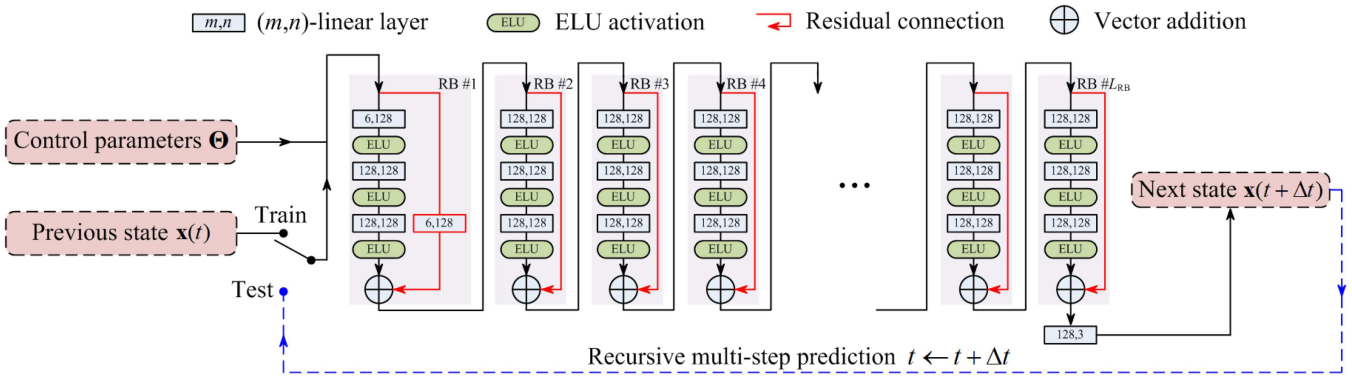


FIG. 1. The architecture of the proposed neural network that cascades L_{RB} residual blocks. The training stage uses samples of one-step predictions while in the test stage multi-step predictions are achieved by recursively using the neural network.

E. Network training and validation

Given a batch of n samples $\{(\mathbf{x}_i, \mathbf{\Theta}_i, \mathbf{y}_i)\}_{i=1}^n$, the loss function for training the neural network is the mean squared error (MSE) between the true values $\mathbf{y}_i = \phi_{\Delta t}(\mathbf{x}_i; \mathbf{\Theta}_i)$ and the learned predictions $\hat{\mathbf{y}}_i = \phi_{\mathrm{NN}}(\mathbf{x}_i; \mathbf{\Theta}_i)$ by the neural network for $i = 1, \ldots, n$, i.e.,

$$\mathscr{L}_{\mathrm{train}}\left(\{(\mathbf{x}_i, \mathbf{\Theta}_i, \mathbf{y}_i)\}_{i=1}^n\right) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 = \frac{1}{n}\sum_{i=1}^{n} \|\phi_{\Delta t}(\mathbf{x}_i; \mathbf{\Theta}_i) - \phi_{\mathrm{NN}}(\mathbf{x}_i; \mathbf{\Theta}_i)\|^2, \quad (7)$$

where $\|\mathbf{x}\| = \sqrt{x^2 + y^2 + z^2}$ is the l^2 norm for $\mathbf{x} = [x, y, z]^{\mathrm{T}}$. The validation loss for the model comparison in Sec. IV is defined as the root mean squared error (RMSE),

$$\mathscr{L}_{\mathrm{val}}(\{(\mathbf{x}_i, \mathbf{\Theta}_i, \mathbf{y}_i)\}_{i=1}^n) = \sqrt{\mathscr{L}_{\mathrm{train}}\left(\{(\mathbf{x}_i, \mathbf{\Theta}_i, \mathbf{y}_i)\}_{i=1}^n\right)}.$$

The neural network is implemented in PyTorch[48] and optimized by the ADAM algorithm[49] with a small learning rate 0.0002. It is trained for 600 epochs with the batch size being 0.2 million. Each epoch includes 1000 batches such that all the training samples could be used once. All the simulations are carried out on a desktop with an Intel Core i9-13900K CPU and a NVIDIA RTX 4090 GPU with 24 GB memory.

III. NUMERICAL RESULTS

A. Prediction accuracies of the solutions

In order to demonstrate the prediction accuracies of ϕ_{NN} across the parameter space $\mathscr{P}$, six ac-slices with $b = 35, 39, 43, 47, 51, 55$ are considered in Fig. 2. The bifurcation diagrams in Fig. 2(a) show the categories of the solutions in these slices. The classification of each parameter choice is obtained by generating a solution trajectory by the RK4 method and counting the number of different local maxima of $z(t)$, i.e., the number of loops.[50] The solutions with more than 20 loops are labeled as chaotic solutions. As shown in Fig. 3, when c is small, the solutions are stable fixed points. Increasing c leads to chaotic behaviors and finally when c is near -0.5, multiple periodic solutions occur and the period doubling bifurcation happens.

Given the state $\mathbf{x}$ at the previous time step and the parameters $\mathbf{\Theta}$, the error between the one-step prediction $\hat{\mathbf{y}} = \phi_{\mathrm{NN}}(\mathbf{x}; \mathbf{\Theta})$ and the true value $\mathbf{y} = \phi_{\Delta t}(\mathbf{x}; \mathbf{\Theta})$ is evaluated by the l^2 norm $\|\mathbf{y} - \hat{\mathbf{y}}\|$. The errors on the six ac-slices in Fig. 2(a) are detailed in Fig. 2(b). Each subfigure is summarized on 1 million random samples generated by the method described in Sec. II D with the fixed b. Each ac-slice is partitioned into a 100×100 grid and each pixel of the grid represents the average error of around 100 random samples with a and c falling in it. It can be seen that the prediction errors are low, medium, and high for the parameters in the regimes of stable fixed points, periodic solutions and chaotic solutions, respectively. The most inaccurate parameter regions are around the left, bottom, and top boundaries of Fig. 2(b6), corresponding to the chaotic regimes near the upper boundary $b = 55$ of the parameter space $\mathscr{P}$, where the training data are insufficient for an accurate prediction. Similarly, the prediction errors in Fig. 2(b1) are higher than those in Fig. 2(b2), as Fig. 2(b1) corresponds to the lower boundary $b = 35$ of $\mathscr{P}$.

The accuracy of the k-step prediction of ϕ_{NN} for the parameter choice $\mathbf{\Theta}$ and initial state $\mathbf{x}_0$ is measured by the normalized error

$$E(k) = \frac{\|\phi_{k\cdot\Delta t}(\mathbf{x}_0; \mathbf{\Theta}) - \phi_{\mathrm{NN}}^k(\mathbf{x}_0; \mathbf{\Theta})\|}{\sqrt{\langle \|\phi_t(\mathbf{x}_0; \mathbf{\Theta})\|^2 \rangle}}, \quad (8)$$

where the mean operator $\langle\cdot\rangle$ is the average of a long trajectory generated by the RK4 method. The accuracy of multi-step prediction of ϕ_{NN} is concisely evaluated by the valid prediction time (VPT) t_v, defined as[28]

$$t_v = \Delta t \cdot \inf_k\{k | E(k) > \delta_{\mathrm{VPT}}, k \in \mathbb{N}\}, \quad (9)$$

where $\mathbb{N}$ is the set of natural numbers, inf is the infimum operator and δ_{VPT} is a threshold. In other words, the VPT is the time when the deviation between the trajectory generated by ϕ_{NN} and the true solution first exceeds the relative ratio δ_{VPT}, which is fixed to[28] 0.4 in the follow experiments. It is worth noting that the predictability of a chaotic system is often measured by the normalized VPT

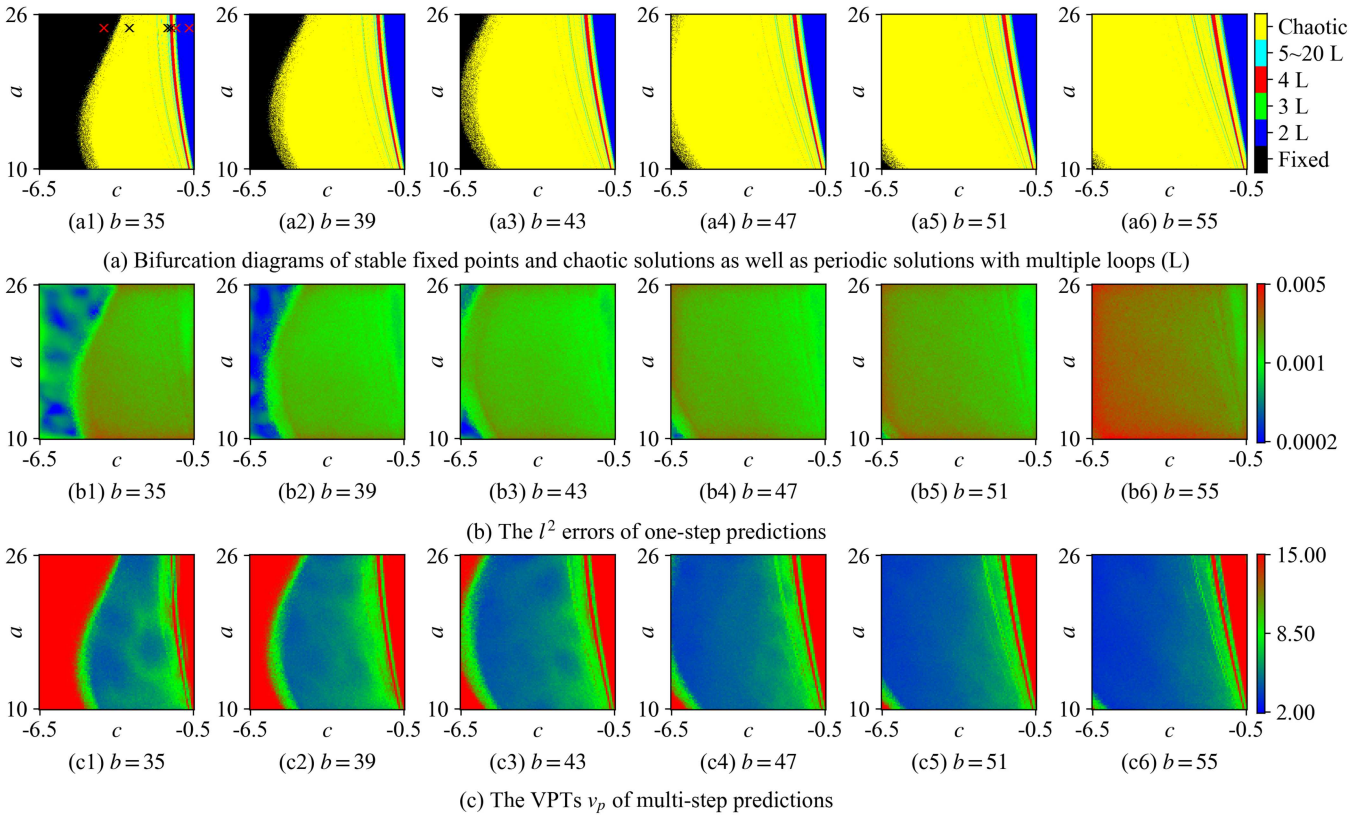


FIG. 2. The categories of the true solutions of Lorenz system and the prediction errors of the neural network on several *ac*-slices. The true solutions corresponding to the six crosses in (a1) are plotted in Fig. 3.

$t_v \cdot \lambda_1$ where the normalization factor is the Lyapunov time $1/\lambda_1$, i.e., the inverse of the LLE.[43] We do not normalize the VPTs in terms of the Lyapunov time for that the calculation of millions of LLEs is time-consuming and the periodic solutions have no Lyapunov times. As expected, the unnormalized VPTs t_v are often lower for chaotic solutions with larger LLEs.

The VPTs on 0.5 million random samples are drawn in each subfigure of Fig. 2(c), where each pixel represents the average VPT of around 50 samples. For each sample $(\mathbf{x}, \boldsymbol{\Theta})$, the maximal prediction step is set to 750 in the simulation for convenience of computation, corresponding to the time horizon $T = 750\Delta t = 15$. One can see that the fixed-point solutions on the left sides and the periodic solutions on the right sides of Figs. 2(c1)–2(c6) can be well predicted for a long time, as the VPTs reach the maximum $T = 15$. In contrast, the VPTs of chaotic solutions are short, especially when the parameters deepen into the chaotic regime. Even though, the minimal average

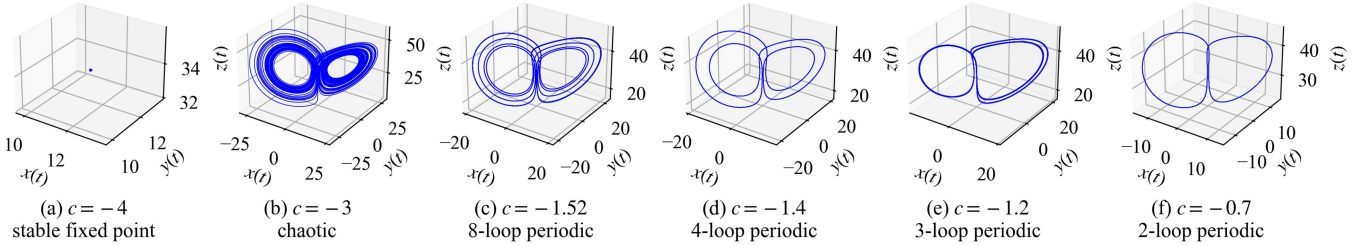


FIG. 3. Six solutions of the Lorenz system for various c and fixed $a = 24.5$ and $b = 35$. The cases correspond to the crosses in Fig. 2(a1).

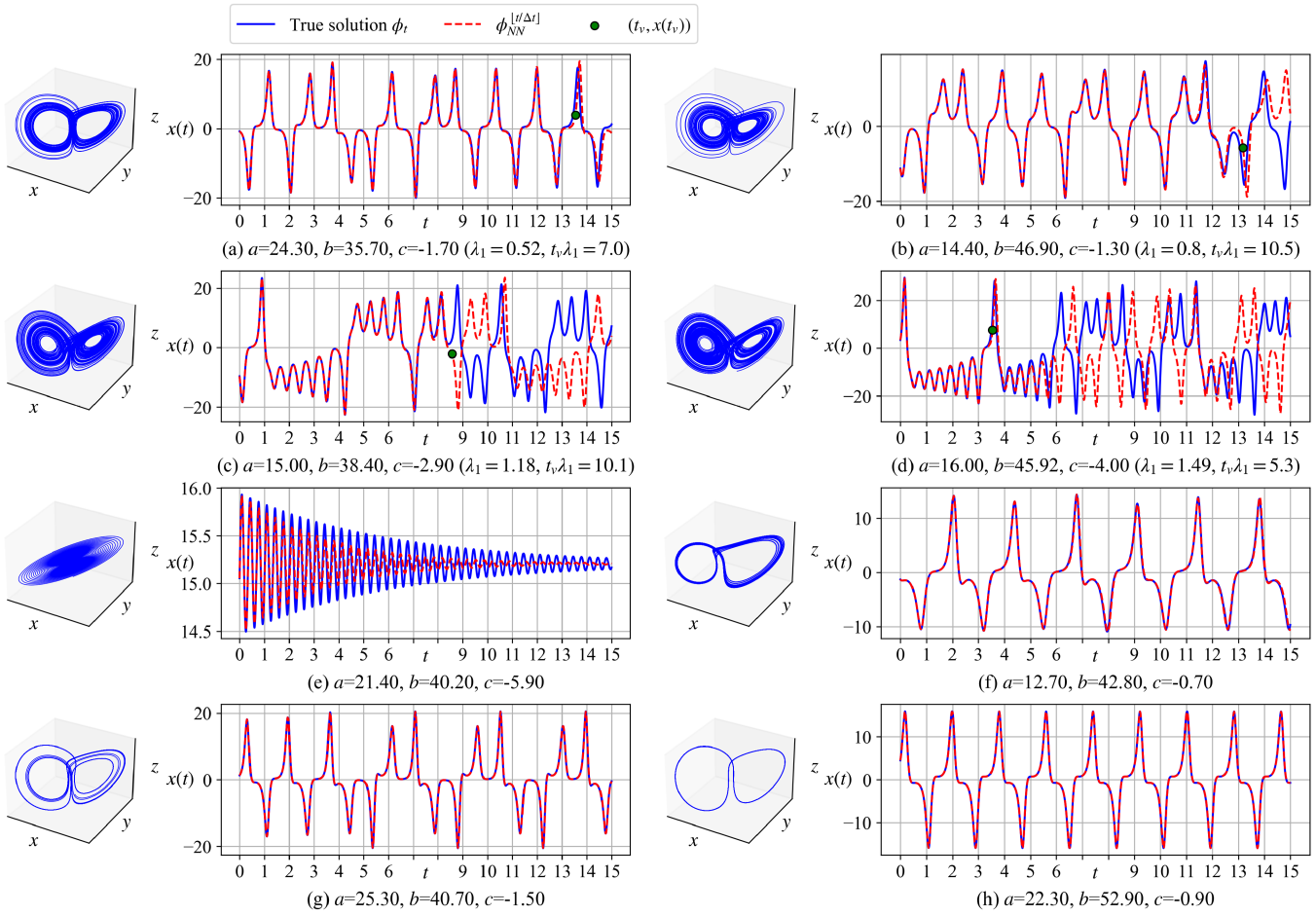


FIG. 4. Comparisons between the multi-step predictions by the proposed neural network ϕ_{NN} and the true solutions generated by the RK4 method. The green dots indicate the states at the VPTs t_v. The LLEs and the normalized VPTs $t_v\lambda_1$ are presented in the four chaotic cases (a)–(d).

VPTs in each pixel of Figs. 2(c1)–2(c6) are 3.73, 3.97, 3.50, 3.32, 3.11 and 2.77, respectively.

Several groups of multi-step predictions of $x(t)$ by ϕ_{NN} are detailed in Fig. 4. Figures 4(a) and 4(b) show that ϕ_{NN} can accurately predict more than $t = 10$ for two chaotic solutions. In (c) and (d), the VPTs are lower since these two cases are more chaotic, i.e., they have larger LLEs λ_1. The VPTs of all the four chaotic cases reach five multiples of the Lyapunov time, i.e., $t_v\lambda_1 > 5$. Though the learned solutions deviate from the true solutions in less than $t = 10$, they may be still visually similar for large t. For example, when $t \in [13, 15]$ in (c) and (d), the x components of $\phi_{\text{NN}}^{\lfloor t/\Delta t \rfloor}$ approximate the opposite of the true solutions. It indicates that the neural network occasionally switches to the symmetric solution

$$\tilde{\mathbf{x}}(t) = [-x(t), -y(t), z(t)]^{\text{T}} \tag{10}$$

of the true solution $\mathbf{x}(t) = [x(t), y(t), z(t)]^{\text{T}}$. Not surprisingly, the correct information of $\mathbf{x}(t)$ such as the extreme values of $x(t)$ and the LLE can still be extracted from the learned solution that incorrectly mixes $\tilde{\mathbf{x}}(t)$ and $\mathbf{x}(t)$. The parameters in (e) correspond to a stable fixed point. The true solution trajectory is a spiral as the initial state is near the fixed point. The convergent speed of ϕ_{NN} to the fixed point is slightly faster than the true solution. The last three examples in (f)–(h) show that ϕ_{NN} can excellently predict weakly chaotic solutions as well as periodic solutions.

The accuracy of multi-step prediction $\phi_{\text{NN}}^k(\mathbf{x}_0; \boldsymbol{\Theta})$ depends not only on the control parameter $\boldsymbol{\Theta}$, but also on the initial state $\mathbf{x}_0$ and the training progress. To illustrate the latter observations, the control parameters $\boldsymbol{\Theta} = [15, 38.4, -2.9]^{\text{T}}$ are fixed and the neural networks right after the 1, 5, 10, 50, 200, and 600 training epochs are selected to perform multi-step predictions. The VPTs of 1 million initial states $\mathbf{x}_0$ randomly sampled on the chaotic attractor are visualized in Fig. 5 in terms of the xz-projections. Figure 5(a) shows that after just one epoch of training, the neural network can accurately predict more than $t = 0.5$ or 25 Δt-steps in the worst case. Figure 5(c) illustrates that 50 epochs of training makes the VPTs reach around $t = 5$ for almost all initial states. When the

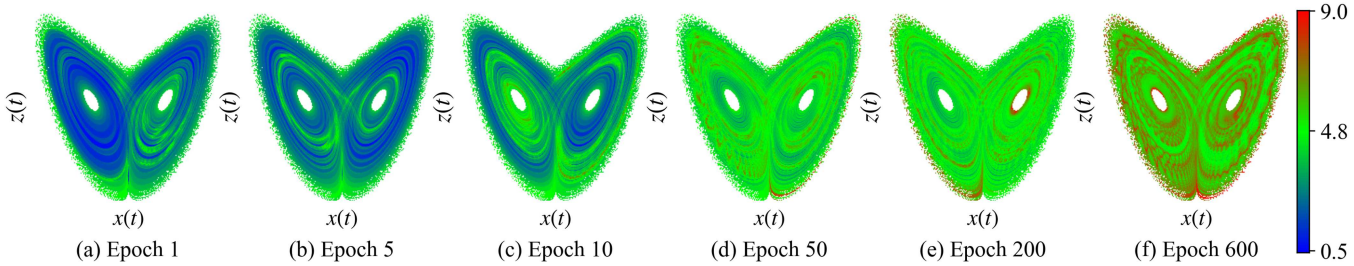


FIG. 5. The VPTs of solutions with different initial states scattered on the attractor for $\Theta = [15, 38.4, -2.9]^{\mathsf{T}}$ ($\lambda_1 = 1.18$). The accuracy of multi-step prediction depends on the initial state as well as the training process.

training process continues, i.e., the number of training epochs increases, the VPTs increase in general. The improvements are locally non-uniform. For example, in Fig. 5(a) the solutions starting near the right hole of the attractor achieve higher VPTs than those starting near the left hole, while in Fig. 5(c) the opposite happens.

B. Statistics of the solutions

We next consider the statistics of the learned solutions. Several bifurcation diagrams of the local maxima of $z(t)$ from the true solutions by the RK4 method and the recursively generated solutions by ϕ_{NN} are compared in Fig. 6. In each of the three columns, one of the parameters a, b, and c is varied as the bifurcation parameter and the other two are fixed. To produce the data of every parameter choice $\mathbf{\Theta}$, the true solution $\{\phi_t(\mathbf{x}_0; \mathbf{\Theta}) | t \in [0, 800]\}$ and the learned solution $\{\phi^k_{\mathrm{NN}}(\mathbf{x}_0; \mathbf{\Theta})\}_{k=1}^{40000}$ are generated from the same random initial state $\mathbf{x}_0$ on the respective attractor, corresponding to the same time horizon $T = 800$. It is clear that the bifurcation diagrams in the second row of Fig. 6 are largely overlapped with the counterparts in the first row. In Fig. 6(a), the main inconsistency takes place at $a = 22$, which is a bifurcation point in that the symmetric solutions defined in Eq. (10) coincide with the true solutions only when $a > 22$. Near this bifurcation point, the learned solutions fluctuate slightly around the true periodic solutions. In general, the long-term statistics of the solutions are learned well by the proposed neural network even though the VPTs are short.

The LLE defined in Eq. (3) is widely applied as a quantitatively metric for evaluating deep learning-based predication methods.[3,27] Figure 7 plots the LLEs of the true solutions by the RK4 method and the learned solutions by ϕ_{NN} with the parameters on six line segments in the parameter space $\mathscr{P}$. For each parameter choice $\mathbf{\Theta}$, the true solution $\{\phi_t(\mathbf{x}_0; \mathbf{\Theta}) | t \in [0, 2000]\}$ is generated with a random initial state $\mathbf{x}_0$ on the attractor and the 100 000 states equally spaced by $\Delta t = 0.02$ are retained. The corresponding learned solution $\{\phi^k_{\mathrm{NN}}(\mathbf{x}_0; \mathbf{\Theta})\}_{k=0}^{100\,000}$ is recursively calculated. The LLEs are calculated by Wolf algorithm[42] from the two trajectories. It can be seen that the two groups of LLEs in the six subfigures are well aligned, especially near the sharp changes in Figs. 7(a), 7(c), and 7(f), demonstrating that in general ϕ_{NN} indeed recovers the correct signatures of chaos.

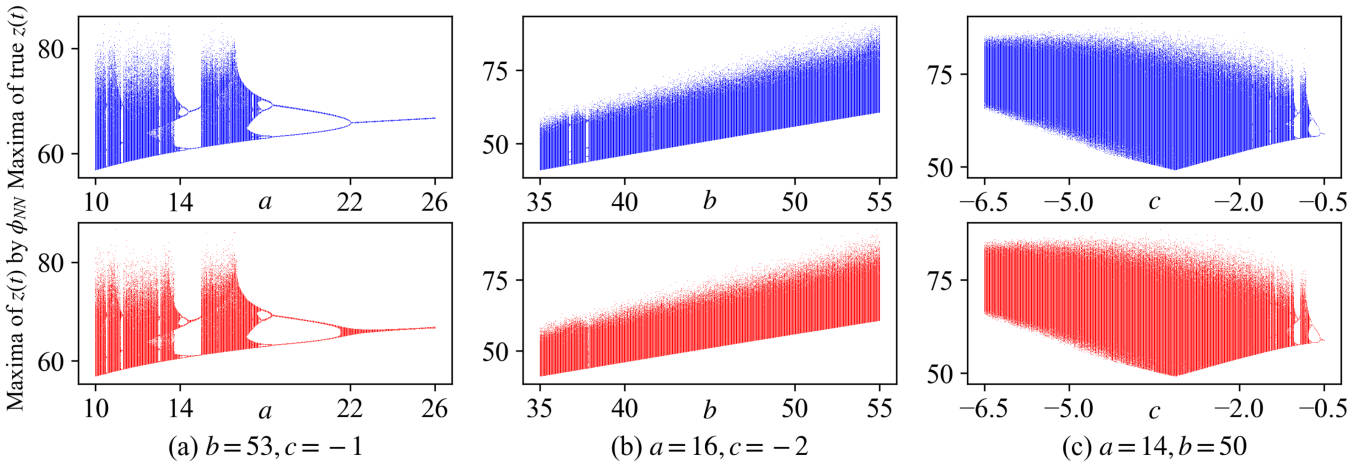


FIG. 6. Three groups of bifurcation diagrams of the local maxima of $z(t)$ calculated from the solutions by the RK4 method (top) and the proposed neural network (bottom). The corresponding LLEs are plotted in Figs. 7(a)–7(c).

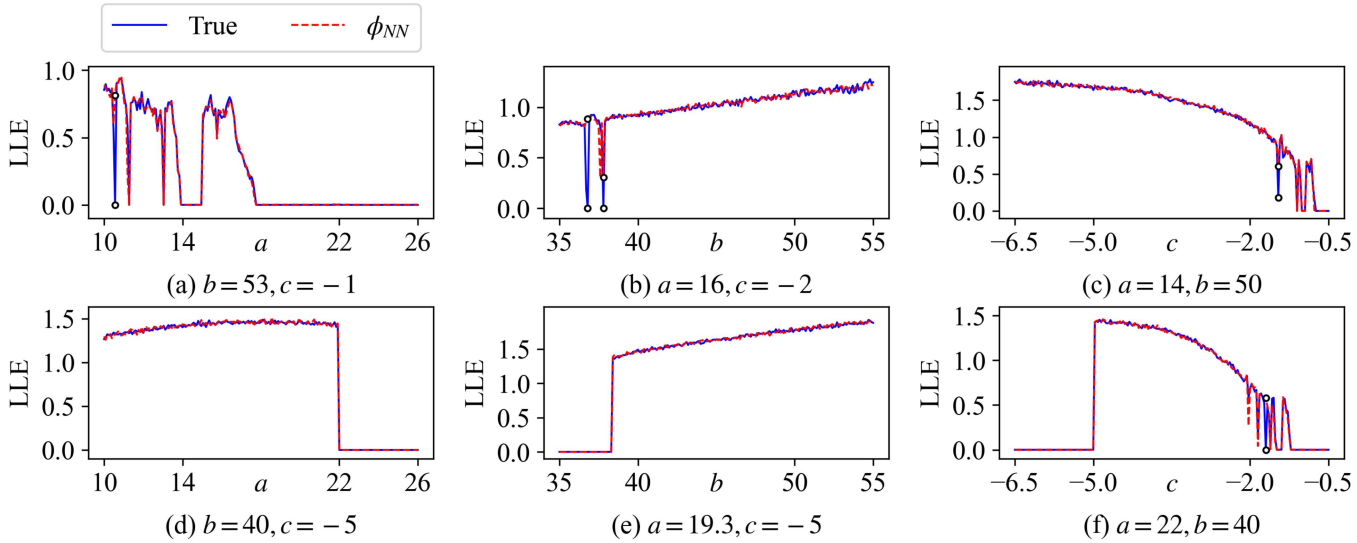


FIG. 7. The LLEs calculated by Wolf algorithm on the true solutions and the solutions recursively generated by the neural network ϕ_{NN}. Cases (a)–(c) correspond to Figs. 6(a)–6(c), respectively. The apparently inconsistent LLEs are marked by black circles.

There are five pairs of apparently inconsistent LLEs in Figs. 7(a)–7(c) and 7(f), respectively, which are marked by small black circles. Though the true solutions in these cases are periodic, the corresponding control parameters are located in small windows of periodic solutions interspersed in the chaotic regime[41] of the parameter space $\mathscr{P}$. In contrast, the learned solutions by ϕ_{NN} may be chaotic, leading to overestimated LLE values. By using the parameters $\mathbf{\Theta} = [16, 37.802, -2]^{\mathrm{T}}$ of the inconsistent case in Fig. 7(b), two true solution trajectories $\phi_t(\mathbf{x}_i; \mathbf{\Theta}), i = 1, 2$ initialized at $\mathbf{x}_1 = [-8.82, -7.87, 7.42]^{\mathrm{T}}$ and $\mathbf{x}_2 = [-8.82, -7.87, 7.43]^{\mathrm{T}}$ are shown in Figs. 8(a) and 8(b) for $t \in [0, 200]$ and $t \in [900, 1000]$, respectively. Figure 8(a) shows that the two solutions are transiently chaotic or metastable chaotic[51,52] for $t < 200$. Figure 8(b) shows that for large t the true solutions converge to two coexisted stable limit cycles, even though the initial states are very close. For $t \in [0, 200]$, the learned solutions in Fig. 8(c) deviate from the true solutions in (a). By comparing Figs. 8(d) and 8(b) for large t values, one can see that the long-term predictions of both solutions by ϕ_{NN} are wrong. The periodic solution initialized at $\mathbf{x}_1$ becomes chaotic in Fig. 8(d), leading to the overestimation of the LLE in Fig. 7(b). Though the learned solution initialized at $\mathbf{x}_2$ is still periodic in Fig. 8(d), it converges to the wrong limit cycle.

By confining the initial states on the xz-plane with $y = -7.87$, the basins of attraction of the two limit cycles in Fig. 8(b) are detailed in Fig. 8(e). Except for a few narrow open sets converging entirely to one of the two limit cycles, the initial states converging to either limit cycle are densely distributed across the majority of the plane. It indicates that the basins of attraction could be fractal.[53,54] The values

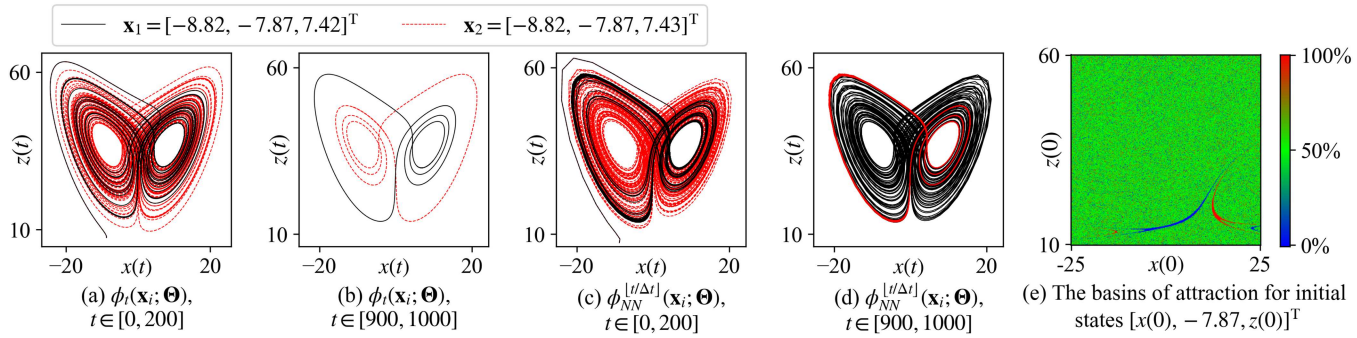
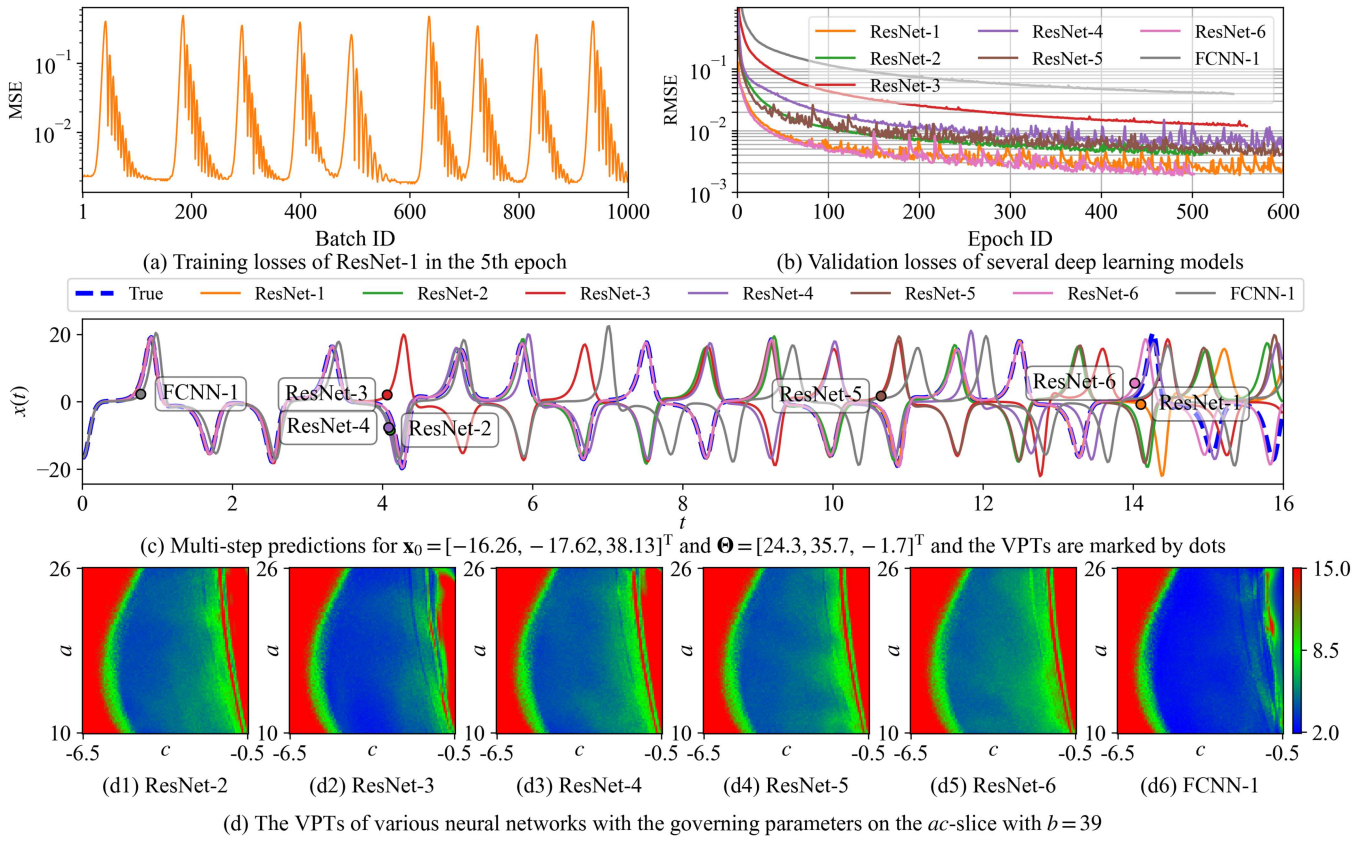


FIG. 8. The true solutions ϕ_t and the solutions predicted by $\phi_{\mathrm{NN}}^{\lfloor t/\Delta t \rfloor}$ from two close initial states $\mathbf{x}_1$ and $\mathbf{x}_2$ with the same control parameters $\Theta = [16, 37.802, -2]^{\mathrm{T}}$. The basins of attraction of the two limit cycles in (b) are detailed in (e), where the colors indicate the proportions of nearby initial states being attracted by the red dashed limit cycle in (b).



(a) Training losses of ResNet-1 in the 5th epoch

(b) Validation losses of several deep learning models

(c) Multi-step predictions for $\mathbf{x}_0 = [-16.26, -17.62, 38.13]^{\mathrm{T}}$ and $\mathbf{\Theta} = [24.3, 35.7, -1.7]^{\mathrm{T}}$ and the VPTs are marked by dots

(d1) ResNet-2 (d2) ResNet-3 (d3) ResNet-4 (d4) ResNet-5 (d5) ResNet-6 (d6) FCNN-1

(d) The VPTs of various neural networks with the governing parameters on the ac-slice with $b = 39$

FIG. 9. Comparison of the training processes and the prediction accuracies among the neural networks listed in Table I.

and types of the long-term map $\phi_t(\cdot; \mathbf{\Theta})$ for large t have significant initial value sensitivity. As a result, the misleading transient chaos in the long-term predictions of the neural network and the convergence to the wrong limit cycles may not be solved by using deeper neural networks or denser training datasets.

IV. MODEL DISCUSSION AND COMPARISON

In building a capable deep learning model, it is vital to choose a proper size of the neural network, including the number of layers and the number of neurons in each layer. In addition, the training process and dataset also affect the accuracy of the resulting neural networks. In this section, we discuss possible guidelines by investigating several neural networks.

In Table I, the neural network utilized in the previous sections is named ResNet-1, consisting of 17 residual blocks. As shown in Fig. 1, each residual block includes three (128, 128)-linear layers. In the first residual block and after the last residual block, a (6, 128) and a (128, 3)-linear layers are required, respectively, to align the shapes of vectors. Therefore there are $17 \times 3 + 2 = 53$ linear layers in total. Since the number of tunable parameters of a (m, n)-linear layer is $mn + n$. ResNet-1 has 827 779 tunable parameters.

In each training epoch, the training losses Eq. (7) of ResNet-1 may fluctuate significantly after every few training batches, instead of decreasing monotonously, as shown in Fig. 9(a). The reason is that each batch of data is insufficient to guide the best optimization direction, such that overshooting and undershooting often occur. In order to conduct a more stable evaluation of the training process, in each epoch, we propose to only record the best model among the 1000 batches when the lowest training loss is attained. Then the validation loss on the independent validation set is calculated on the best model as the validation loss of the whole epoch. The curve of ResNet-1 in Fig. 9(b) demonstrates that the validation losses of these best models decrease more steadily when the training process continues.

By varying the network architectures and the sizes of the training datasets using OPPT sampling, we find that the neural networks having more layers and more training data achieve higher prediction accuracies. By decreasing the number of residual blocks, ResNet-1, ResNet-2, and ResNet-3 in Table I have less and less tunable parameters. The average one-step errors in Table I and the validation losses in Fig. 9(b) increase accordingly. Though the architecture of ResNet-4 is identical to ResNet-1, ResNet-4 uses 40 millions training samples, which is only 1/5 of the data size used for training ResNet-1. Figure 9(b) shows that ResNet-4 is considerably inferior

TABLE I. A comparison of neural networks with different architectures and training data.

ID	No. Residual blocks L_{RB}	Shapes of residual blocks	No. Linear layers	No. Tunable parameters	No. Training samples in millions	One-step error[c] MEAN/SD $\times 10^3$	Training time per epoch in second
ResNet-1[a]	17	(128, 128, 128)[b]	53	827 779	200	1.50/1.05	1121
ResNet-2	9	(128, 128, 128)	29	431 491	200	3.21/2.28	1139
ResNet-3	3	(128, 128, 128)	11	134 275	200	9.42/7.06	1109
ResNet-4	17	(128, 128, 128)	53	827 779	40	4.07/2.93	222
ResNet-5	6	(512, 128, 512)	20	829 699	200	3.17/2.24	1154
ResNet-6	40	(128, 64, 128)	122	822 915	200	1.56/1.07	1319
FCNN-1	0	NA	10	133 379	200	39.90/28.74	1109

[a]The neural network is utilized throughout Sec. III.
[b](m, n, k) stands for that the three linear layers in one residual block have the input and output lengths being (m, n), (n, n), and (n, k), respectively. The first residual block in the neural network is different as an additional linear layer described in Eq. (5) is required.
[c]The mean and standard deviation (SD) of the one-step prediction errors (l^2 norm) are calculated on 100 million samples with the parameters uniformly distributed in $\mathscr{P}$.

to ResNet-1. Nevertheless, the last column of Table I indicates that the training of ResNet-4 is the fastest since the time of each training epoch is roughly proportional to the size of the training dataset.

The aforementioned neural networks ResNet-1 to ResNet-4 mainly utilize the same residual blocks cascading three $(128, 128)$-linear layers with 49 536 parameters. We now increase the numbers of neurons in the linear layers such that the sizes of the three linear layers in each residual block are $(512, 128)$, $(128, 128)$, and $(128, 512)$ and there are 148 224 parameters in each residual block. ResNet-5 cascades 6 such wider residual blocks such that the number of tunable parameters is comparable to ResNet-1. Figure 9(b) shows that ResNet-5 is inferior than ResNet-1 but comparable to ResNet-2.

In contrast, we can readily increase the number of residual blocks and decrease the number of neurons in each residual block. For example, the ResNet-6 in Table I has 40 residual blocks. Each block consists of three linear layers with the smaller sizes $(128, 64)$, $(64, 64)$, and $(64, 128)$, such that the total number of parameters is comparable to ResNet-1. Though there are 130% more linear layers in ResNet-6, its training speed is only 18% slower than ResNet-1. Table I and Fig. 9(b) show that the performance of ResNet-6 is comparable to ResNet-1. In summary, if the tunable parameters and the training dataset are comparable, the deeper neural network with less neurons in each layer may achieve a better prediction accuracy than the shallower neural network with more neurons in each layer. If we keep increasing the depth of the neural network, the improvement of prediction accuracy may reach a limit.

If we remove all the residual connections in ResNet-3, it becomes a FCNN that stacks 10 linear layers separated by activation functions. The simplified neural network is named FCNN-1. As listed in Table I, the number of tunable parameters of FCNN-1 is comparable to ResNet-3, except that there is an additional linear layer in the first residual block of ResNet-3 to adjust the output shape of the residual connection. However, Fig. 9(b) shows that the learning speed of FCNN-1 in terms of the validation loss is significantly slower than ResNet-3, verifying the vital role of the residual connections in speeding up the training process.[47,55] Table I further shows that the average one-step error of FCNN-1 is four times higher than ResNet-3. It demonstrates clearly that the residual connections can greatly improve the nonlinear mapping capability of the neural network.

The multi-step predictions of $x(t)$ by the neural networks listed in Table I with the same random initial state and control parameters are compared in Fig. 9(c). ResNet-1 and ResNet-6 perform the best as they are the deepest and they are trained on a large dataset. The shallow ResNet-2, ResNet-3 and the deep ResNet-4 with limited training data achieve inferior performances. The performance of the shallowest FCNN-1 without residual connections is the poorest. A more thorough comparison of the VPTs by these neural networks on the ac-slice with fixed $b = 39$ is detailed in Fig. 9(d) using the similar setting for producing Fig. 2(c2) of ResNet-1. It can be seen that all the neural networks are effective to some extent. The shallow neural networks ResNet-2, ResNet-3, and FCNN-1 as well as the ResNet-4 with limited training data may not achieve accurate long-term predictions for the periodic solutions when c values are near -0.5. ResNet-5 and ResNet-6 achieve comparable performances to ResNet-1, since they have comparable sizes of tunable parameters and training dataset.

Though the proposed method requires a large training dataset, the OPPT sampling is not necessary. When only a limited number of solution trajectories with different control parameters is available, one can construct a large dataset by sampling multiple points on each trajectory. The neural network still can learn comprehensively the parameter-dependent system. For example, the ResNet-4 in Table I utilizes 40 million samples on 40 million trajectories. Instead, here we train ResNet-4 from another three training datasets that include only 4,000, 400 and 200 trajectories. Each trajectory is generated using the method described in Sec. II D with the control parameters uniformly sampled in Eq. (6). In the three new cases, 10,000, 100,000 and 200,000 training samples are uniformly sampled on each trajectory with $t \in [75, 300]$, respectively, such that all the training datasets include 40 million samples. The validation losses in Fig. 10(a) and the VPTs in Figs. 9(d3) and 10(b2) demonstrate that the prediction accuracies of the models trained on 40 million

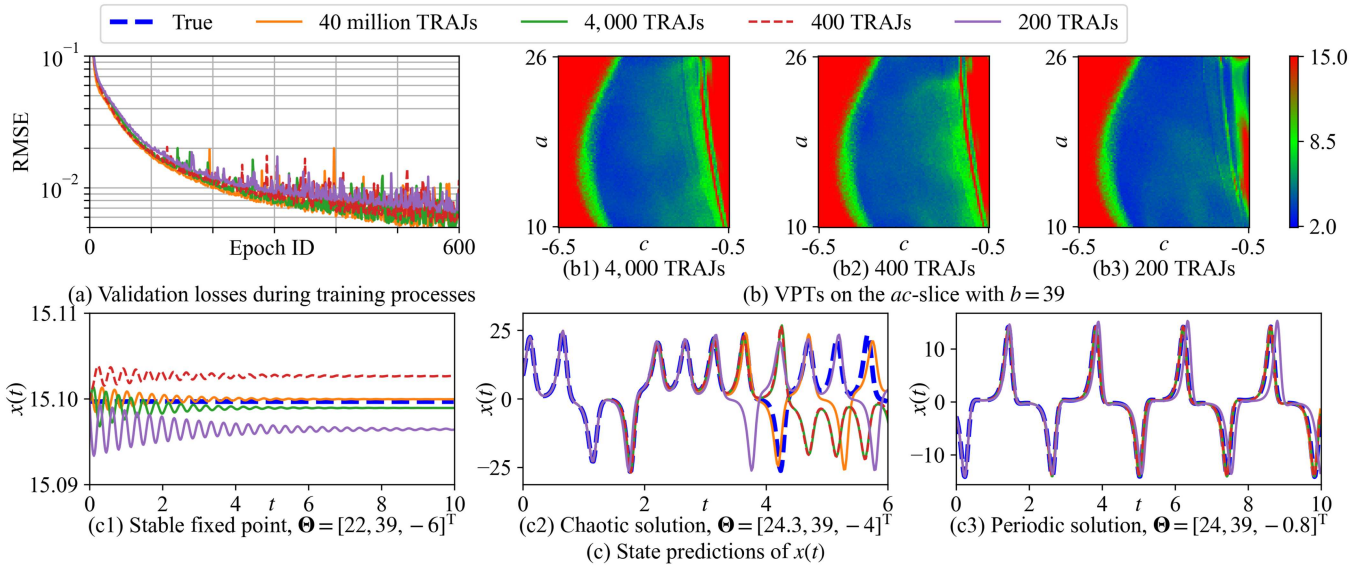


FIG. 10. Comparison of four implementations of the ResNet-4 trained on datasets with 40 million samples, which are sampled from 40 million, 4000, 400, and 200 trajectories (TRAJs), respectively. The VPTs of the implementation using 40 million trajectories are shown in Fig. 9(d3).

trajectories and 400 trajectories have no significant difference. Only when the number of training trajectories decreases to 200, Fig. 10(b3) indicates that the long-term predictions of the periodic solutions near $c = 0.5$ fail early.

Figure 10(c) further details the predictions of three solutions using the four implementations of ResNet-4. Figure 10(c1) shows that the deviations between the x components of the stable fixed points of the learned solutions and the true fixed point increase when the numbers of training trajectories decrease. Remarkably, even though there are only 200 training trajectories, the deviation is less than 0.05. Figure 10(c2) implies that 200 training trajectories are sufficient for accurately predicting a chaotic solution for $t = 3$. The periodic solution in (c3) is effectively learned from 200 training trajectories, though the period of the learned solution is a slightly longer than the true solution. Overall, a large dataset sampled from hundreds of solution trajectories is enough for training a neural network capable of generating visually similar solutions for different control parameters.

V. CONCLUSION

In this work, a very deep neural network is proposed to perform recursive prediction of the chaotic Lorenz system with three control parameters. The neural network cascades a large number of simple linear layers with residual connections. A large dataset of one-step predictions is used to sufficiently train it. The results show that the proposed method can simultaneously perform long-term predictions for the periodic solutions and accurately predict chaotic solutions for several multiples of the Lyapunov time. The long-term predictions of the neural network well approximate the dynamical characteristics of the true solutions such as the bifurcation diagrams and the LLEs. Nevertheless, the method may learn the misleading transient chaos captured by the training dataset and fail to predict the asymptotic behaviors of the solutions due to the initial value sensitivity of the Lorenz system.

An ablation study of different network architectures further shows that the residual connections are indispensable for improving the nonlinear mapping capability of very deep neural networks. When the numbers of tunable weights are comparable, the neural networks consisting of more layers or being trained on larger datasets perform better. A dataset sampled from a very limited number of solution trajectories is enough for an effective learning. The training speed is roughly proportional to the size of training dataset and insensitive to the increase of the network layers. These observations suggest that constructing massive training data and building very deep neural networks with residual connections are promising modeling strategies for parameter-dependent dynamical systems. Nevertheless, shallow neural networks trained on limited data are also effective with inferior prediction accuracies.

ACKNOWLEDGMENTS

This study was partly supported by the National Natural Science Foundation of China (NNSFC) (Grant Nos. 12202255, 12102341 and 12072264), Natural Science Basic Research Program of Shaanxi, China (Nos. 2022JQ-020, 2022JG-716, and 2022JM-007), and Scientific Research Program Funded by Shaanxi Provincial Education Department (Program No. 21JK0904). Y. Xu acknowledges the support from the Key International (Regional) Joint Research Program of the NSF of China (No. 12120101002). J. Kurths acknowledges the support from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), Project No. 411803875.

AUTHOR DECLARATIONS

Conflict of Interest

The authors declare that they have no conflict of interest.

Author Contributions

Xiaolong Wang: Funding acquisition (equal); Methodology (equal); Software (lead); Visualization (lead); Writing – original draft (equal); Writing – review & editing (equal). **Jing Feng:** Conceptualization (equal); Funding acquisition (equal); Methodology (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Yong Xu:** Conceptualization (equal); Funding acquisition (equal); Methodology (equal); Supervision (lead); Validation (equal); Writing – review & editing (equal). **Jürgen Kurths:** Conceptualization (equal); Methodology (equal); Validation (equal); Writing – review & editing (equal).

DATA AVAILABILITY

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

REFERENCES

[1] C. You, Y. Deng, W. Hu, J. Sun, Q. Lin, F. Zhou, C. H. Pang, Y. Zhang, Z. Chen, and X. Zhou, "Estimation of the time-varying reproduction number of COVID-19 outbreak in China," Int. J. Hyg. Environ. Health **228**, 113555 (2020).
[2] Q. Liu, Y. Xu, J. Kurths, and X. Liu, "Complex nonlinear dynamics and vibration suppression of conceptual airfoil models: A state-of-the-art overview," Chaos **32**, 062101 (2022).
[3] R. Cestnik and M. Abel, "Inferring the dynamics of oscillatory systems using recurrent neural networks," Chaos **29**, 063128 (2019).
[4] J. Wang, T. Sun, B. Liu, Y. Cao, and H. Zhu, "CLVSA: A convolutional LSTM based variational sequence-to-sequence model with attention for predicting trends of financial markets," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19 (AAAI Press, 2019), pp. 3705–3711.
[5] W. Zan, Y. Xu, and J. Kurths, "Path integral solutions for n-dimensional stochastic differential equations under α-stable Lévy excitation," Theor. Appl. Mech. Lett. **13**, 100430 (2023).
[6] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2019).
[7] Q. Liu, Y. Xu, Y. Li, J. Kurths, and X. Liu, "Fixed-interval smoothing of an aeroelastic airfoil model with cubic or free-play nonlinearity in incompressible flow," Acta Mech. Sin. **37**, 1168–1182 (2021).
[8] D. Liu, S. Xu, and J. Ma, "Bayesian system identification and chaotic prediction from data for stochastic Mathieu-van der Pol-Duffing energy harvester," Theor. Appl. Mech. Lett. **13**, 100412 (2023).
[9] P. Rajendra and V. Brahmajirao, "Modeling of dynamical systems through deep learning," Biophys. Rev. **12**, 1311–1320 (2020).
[10] X. Wang, J. Feng, Q. Liu, Y. Li, and Y. Xu, "Neural network-based parameter estimation of stochastic differential equations driven by Lévy noise," Phys. A **606**, 128146 (2022).
[11] J. Feng, X. Wang, Q. Liu, Y. Li, and Y. Xu, "Deep learning-based parameter estimation of stochastic differential equations driven by fractional Brownian motions with measurement noise," Commun. Nonlinear Sci. Numer. Simul. **127**, 107589 (2023).
[12] H. Zhang, Y. Xu, Q. Liu, X. Wang, and Y. Li, "Solving Fokker–Planck equations using deep KD-tree with a small amount of data," Nonlinear Dyn. **108**, 4029–4043 (2022).
[13] Y. Xu, H. Zhang, Y. Li, K. Zhou, Q. Liu, and J. Kurths, "Solving Fokker-Planck equation using deep learning," Chaos **30**, 013133 (2020).
[14] H. Zhang, Y. Xu, Q. Liu, and Y. Li, "Deep learning framework for solving Fokker-Planck equations with low-rank separation representation," Eng. Appl. Artif. Intell. **121**, 106036 (2023).
[15] S. Hassona, W. Marszalek, and J. Sadecki, "Time series classification and creation of 2D bifurcation diagrams in nonlinear dynamical systems using supervised machine learning methods," Appl. Soft Comput. **113**, 107874 (2021).
[16] W. S. Lee and S. Flach, "Deep learning of chaos classification," Mach. Learn.: Sci. Technol. **1**, 045019 (2020).
[17] D. W. Liedji, J. H. Talla Mbé, and G. Kenné, "Chaos recognition using a single nonlinear node delay-based reservoir computer," Eur. Phys. J. B **95**, 18 (2022).
[18] N. Boullé, V. Dallas, Y. Nakatsukasa, and D. Samaddar, "Classification of chaotic time series with deep learning," Physica D **403**, 132261 (2020).
[19] J. Nam and J. Kang, "Classification of chaotic squeak and rattle vibrations by CNN using recurrence pattern," Sensors **21**, 8054 (2021).
[20] M. Zanin, "Can deep learning distinguish chaos from noise? Numerical experiments and general considerations," Commun. Nonlinear Sci. Numer. Simul. **114**, 106708 (2022).
[21] B. Aricioglu, S. Uzun, and S. Kacar, "Deep learning based classification of time series of Chen and Rössler chaotic systems over their graphic images," Physica D **435**, 133306 (2022).
[22] G. Kavuran, "When machine learning meets fractional-order chaotic signals: Detecting dynamical variations," Chaos Solitons Fractals **157**, 111908 (2022).
[23] H. Rappeport, I. Levin Reisman, N. Tishby, and N. Q. Balaban, "Detecting chaos in lineage-trees: A deep learning approach," Phys. Rev. Res. **4**, 013223 (2022).
[24] D. Ayers, J. Lau, J. Amezcua, A. Carrassi, and V. K. Ojha, "Supervised machine learning to estimate instabilities in chaotic systems: Estimation of local Lyapunov exponents," Q. J. R. Meteorol. Soc. **149**, 1236–1262 (2022).
[25] Y. Tang, J. Kurths, W. Lin, E. Ott, and L. Kocarev, "Introduction to focus issue: When machine learning meets complex systems: Networks, chaos, and nonlinear dynamics," Chaos **30**, 063151 (2020).
[26] R. Falahian, M. M. Dastjerdi, M. Molaie, S. Jafari, and S. Gharibzadeh, "Artificial neural network-based modeling of brain response to flicker light," Nonlinear Dyn. **81**, 1951–1967 (2015).
[27] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data," Chaos **27**, 121102 (2017).
[28] 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).
[29] H. Fan, J. Jiang, C. Zhang, X. Wang, and Y.-C. Lai, "Long-term prediction of chaotic systems with machine learning," Phys. Rev. Res. **2**, 012080 (2020).
[30] J. de Jesús Serrano-Pérez, G. Fernández-Anaya, S. Carrillo-Moreno, and W. Yu, "New results for prediction of chaotic systems using deep recurrent neural networks," Neural Process. Lett. **53**, 1579–1596 (2021).
[31] A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian, "Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: Reservoir computing, artificial neural network, and long short-term memory network," Nonlinear Processes Geophys. **27**, 373–389 (2020).
[32] P. Dubois, T. Gomez, L. Planckaert, and L. Perret, "Data-driven predictions of the Lorenz system," Phys. D **408**, 132495 (2020).
[33] Z. Lin, Y. Liang, J. Zhao, and J. Li, "Predicting solutions of the Lotka–Volterra equation using hybrid deep network," Theor. Appl. Mech. Lett. **12**, 100384 (2022).
[34] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," Nat. Commun. **9**, 4950 (2018).
[35] E. N. Lorenz, "Deterministic nonperiodic flow," J. Atmos. Sci. **20**, 130–141 (1963).
[36] C. Sparrow, *The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors* (Springer, 1982).
[37] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, "Scaling learning algorithms toward AI," in *Large-Scale Kernel Machines* (MIT Press, 2007), pp. 321–359.
[38] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," IEEE Trans. Neural Netw. Learn. Syst. **25**, 1553–1565 (2014).
[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Las Vegas, NV, 2016), pp. 770–778.

[40]H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: A review," Data Min. Knowl. Discov. **33**, 917–963 (2019).
[41]S. H. Strogatz, "Nonlinear dynamics and chaos: With applications to physics, biology, chemistry and engineering," Phys. Today **48**(3), 93–94 (1994).
[42]A. Wolf, J. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," Phys. D **16**, 285–317 (1985).
[43]A. Pikovsky and A. Politi, *Lyapunov Exponents: A Tool to Explore Complex Dynamics* (Cambridge University Press, 2016).
[44]T. Nguyen, M. Raghu, and S. Kornblith, "Do wide and deep networks learn the same things? Uncovering how neural network representations vary with width and depth," arXiv **2010.15327** (2020).
[45]D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," arXiv **1511.07289** (2016).
[46]A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Commun. ACM **60**, 84–90 (2012).
[47]C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *2017 AAAI Conference on Artificial Intelligence*, AAAI'17 (AAAI Press, 2017), pp. 4278–4284.
[48]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY, 2019), pp. 8026–8037.
[49]D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," arXiv **1412.6980** (2017).
[50]G. C. Layek, *An Introduction to Dynamical Systems and Chaos* (Springer, 2015).
[51]J. A. Yorke and E. D. Yorke, "Metastable chaos: The transition to sustained chaotic behavior in the Lorenz model," J. Stat. Phys. **21**, 263–277 (1979).
[52]A. M. Wojtusiak, A. G. Balanov, and S. E. Savel'ev, "Intermittent and metastable chaos in a memristive artificial neuron with inertia," Chaos Solitons Fractals **142**, 110383 (2021).
[53]I. Djellit, J. C. Sprott, and M. R. Ferchichi, "Fractal basins in the Lorenz model," Chinese Phys. Lett. **28**, 060501 (2011).
[54]W. Huang, Y. Zhang, and X. Yang, "Complicated boundaries of the attraction basin in a class of three-dimensional polynomial systems," Int. J. Bifurc. Chaos **32**, 2250235 (2022).
[55]H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18 (Curran Associates Inc., Red Hook, NY, 2018), pp. 6391–6401.