

Predicting chaotic system behavior using machine learning techniques

Huaiyuan Rao^{1*} , Yichen Zhao¹, and Hsuan-Pin Chen¹

¹*School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia, United States of America*

Article History:

Received: July 18, 2025

Revised: August 7, 2025

Accepted: August 12, 2025

Published online: August 26, 2025

ABSTRACT

Recently, the superior performance of machine learning approaches over classical forecasting models for complex time series analysis has been demonstrated in various domains. However, predicting chaotic time series continues to be a major challenge due to their inherent complexity. This paper investigates the accuracy, efficiency, and robustness of three recurrent neural network architectures: (i) next generation reservoir computing, (ii) reservoir computing, (iii) long short-term memory for the task of chaotic systems prediction. Four canonical chaotic systems, namely the Lorenz, Rössler, Chen, and Qi systems, are used for comparing these three methods. Numerical results demonstrate that the next generation reservoir computing is more computationally efficient and offers greater potential for predicting long-term chaotic time series.

Keywords: Time series prediction; Chaotic time series; Recurrent neural networks



1. Introduction

The analysis of time series data is fundamental to numerous fields within the natural and social sciences,¹⁻³ as these datasets are essential for understanding complex system behavior and developing predictive models. Recently, machine learning techniques have emerged as powerful approaches for analyzing and predicting time series data.⁴⁻⁹ Nevertheless, predicting highly complex time series remains a significant challenge due to their non-periodic behavior. Chaotic time series are among the most complex because even small perturbations in initial values lead to extremely large variations in their behavior, rendering long-term prediction of chaotic time series exceptionally difficult.

For predicting chaotic time series, recurrent neural networks (RNNs) significantly outperform conventional feed-forward (FF) neural networks¹⁰⁻¹⁴ because their recurrent structure is inherently designed to capture temporal dependencies. Among RNNs, two specific architectures, namely long short-term memory (LSTM) networks¹⁵⁻¹⁸ and reservoir computing (RC),¹⁹⁻²² have excelled at predicting chaotic time series. By incorporating gating mechanisms, LSTM networks overcome the vanishing gradient problem that impairs the ability of traditional RNNs to learn long-range dependencies. On the other hand, a RC model utilizes a type of RNN with a large, fixed pool of interconnected neurons called the reservoir. To simplify training and avoid the vanishing gradient problem, the connection weights to and within the reservoir are initialized randomly and

***Corresponding author(s):**

Huaiyuan Rao (hrao43@gatech.edu).

Citation:

Rao H, Zhao Y, Chen H-P. Predicting chaotic system behavior using machine learning techniques. *Nonlinear Sci Cont Eng*. doi: 10.36922/NSCE025290003

Copyright: © 2025 The Author(s). This is an Open Access article distributed under the terms of the Creative Commons Attribution License, permitting distribution, and reproduction in any medium, provided the original work is properly cited.

remain fixed. Consequently, only the output weights are trained, typically using a regularized linear least-squares optimization. A comparison of these two methods for predicting chaotic time series is presented in Vlachas et al.²³

The effectiveness of randomly constructed RCs in accurately forecasting chaotic time series has motivated a deeper investigation into their operating principles. A key insight from Bollt²⁴ showed that RCs with linear and quadratic readouts can be interpreted as vector auto-regressive (VAR) and non-linear VAR (NVAR) models, respectively. Inspired by this, next-generation reservoir computing (NG-RC) was proposed in Gauthier et al.²⁵ Compared to traditional RC, NG-RC replaces the random reservoir with a feature vector composed directly of the time-delayed system observations and their nonlinear functions, effectively creating a data-driven NVAR model. Additionally, NG-RC requires less training time than conventional RC. Due to these advantages, NG-RC can reproduce long-term chaotic time series and infer the behavior of unseen data from a chaotic system.

In spite of achieving high performance on one-step-ahead predictions for chaotic time series,^{26,27} multi-step forecasting remains a major challenge and a critical benchmark for machine learning. An LSTM network that achieves strong performance on multi-step chaotic time series forecasting was designed in Sangiorgio and Dercole.¹⁸ Other LSTM networks have proven to be an effective method for multi-step time series prediction.^{28–32} In recent years, RC has been extensively investigated for the task of multi-step forecasting of chaotic time series.^{33–37} Na et al.³⁸ proposed a modified RNN architecture named the hierarchical delay-memory echo state network (HDESNE), which has multiple reservoirs with delay-memory capabilities to better extract information from chaotic time series. To overcome the antagonistic trade-off between memory capacity and nonlinear transformation for multi-step chaotic time series prediction, the hierarchical echo state network with augmented random features (HESNE-ARF) was proposed in Na et al.,³⁹ which leverages a hierarchical structure and nonlinear kernel transformations to enhance both properties simultaneously while maintaining computational efficiency. As a computationally efficient method, NG-RC has demonstrated strong performance in multi-step chaotic time series prediction.^{40–42} Furthermore, the potential of NG-RC remains underexplored, leaving significant avenues for future investigation.

The main contributions of this work are threefold. Firstly, we compare the capability of three ML methods for forecasting chaotic time series obtained from four representative dynamical systems. Secondly, we further compare the abilities of RC and NG-RC on predicting chaotic time series using a smaller observable dataset and extend their prediction horizons. Finally, we test the robustness of RC and NG-RC in predicting chaotic time series derived from noisy system ODEs.

The remainder of this paper is organized as follows. In Section 2, we formulate the benchmarking prediction problems. Section 3 summarizes the modeling approaches used for forecasting chaotic time series, providing details on each model's implementation and the evaluation metrics employed. In Section 4, we apply these methods to address the different forecasting tasks and discuss each method's strengths and weaknesses. Finally, Section 5 presents our conclusions.

2. Problem formulation

The evolution of chaotic system is governed by an ordinary differential equation (ODE):

$$\dot{x}(t) = f(x(t)), \quad (1)$$

where $f(\cdot)$ represents the system dynamics, $x(t) \in \mathbb{R}^n$ is the state vector, and t denotes time.

Problem 1 (Multi-steps Forecasting Chaotic Time Series): Suppose in a finite time horizon $t \in [0, T]$, the state vector $x(t)$ is only observable in a limited time period $[0, T_1]$, where $0 < T_1 < T$. This means we have a discrete time series of the system's state, sampled at a uniform interval t_0 over the period $[0, T_1]$. The data is represented by the sequence $x(0), x(t_0), x(2t_0), \dots, x(T_1)$. Our aim is to use these time-limited observable state data to predict the system's true state in the time horizon $t \in [T_1, T]$.

Problem 2 (Predicting Noisy Chaotic System Behavior): Chaotic system process is easily affected by unknown noise in the real world. Usually, the observed state of a system is a combination of real system state $x(t) \in \mathbb{R}^n$ and noise $\varepsilon \in \mathbb{R}^n$. Using noisy input data represents a more realistic implementation in practical applications for chaotic time series predictions. In short, predicting noise chaotic time series also test algorithm's robustness when encounter some unknown perturbations.

3. Models and methods

3.1. Long short-term memory network

An LSTM network processes a variable-length sequence $x = (x_1, x_2, \dots, x_n)$ using a recurring module with a central cell state. This cell state is updated at each time step, and the flow of information is governed by the following three primary gates:

- (i) The forget gate determines the proportion of the previous state's information that needs to be discarded
- (ii) The input gate regulates how much of the new information from the current input is stored in the cell state.
- (iii) The output gate controls which portion of the updated cell state is passed on as the hidden state and exposed to the next layer. The forms of the equations for an LSTM cell are

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f), \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i), \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o), \\ \tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \\ h_t &= o_t \odot \sigma_h(c_t), \end{aligned} \quad (2)$$

where $x_t \in \mathbb{R}^d$ is the input vector, f_t, i_t, o_t represent the gate activation vectors, $(h_t, \tilde{c}_t)$ are the output vector and input activation vector, respectively, and $c_t \in \mathbb{R}^h$ is the cell state vector. The learnable parameters of the network are given by $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$, and $b \in \mathbb{R}^h$, which correspond to the input weights, recurrent weights, and bias vector, respectively. The structure of LSTM is shown in Figure 1.

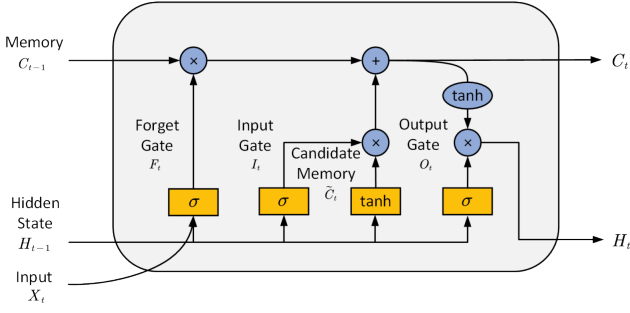


Figure 1. The LSTM cell's structure

LSTMs are useful to handle sequence prediction problems with input data that has long-range temporal dependencies. It is designed to retain information from past states and specifically addresses long-term dependency issues, making it perfectly suited for modeling a chaotic system. This is because the future state of the system predominantly relies on a limited number of preceding states.

3.2. Reservoir computing (RC)

The most common realization of the RC approach is echo state networks (ESNs), which have the advantage of low training cost, which is superior in terms of low training cost, comparing to other ML approaches. A key feature of it is that the training process is confined to the output layer, namely the readout layer. While the internal network parameters are randomly initialized and remain fixed thereafter, only the readout weights are learned. Consequently, the optimization task simplifies from a nonlinear problem to a standard linear regression, which can be solved efficiently.

RC is an effective method for modeling and predicting nonlinear dynamics (e.g., chaotic dynamics). Figure 2 shows a typical type of RC, which consists of an input layer, a reservoir layer (also known as the hidden layer), and an output layer. The input layer and output layer sizes are specified by a nonlinear system's input and output sizes. The raw inputs are transformed and linked to the internal part of the RC, which consists of a coupled dynamical system with multiple nodes forming a recurrent network. The internal states of this system are calculated according to predetermined computational rules, and the outputs are generated by combining these internal states. In a standard RC setup, the connections from input to internal states and within the internal network are fixed, while only the connection from internal states to output is trained. This contrasts with general RNNs, where the entire network is trainable. RC can generally be mathematically described as the follows:

$$\begin{aligned} x(t+1) &= (1-\gamma)x(t) + \gamma f(Wx(t) \\ &\quad + W^{(\text{in})}u(t) + b) \\ y(t) &= W^{(\text{out})}x(t) \end{aligned} \quad (3)$$

where f is a nonlinear activation function (e.g., $\tanh(\cdot)$). The inputs and outputs are mapped by weights $W^{(\text{in})}$ and $W^{(\text{out})}$, respectively. On the other hand, the internal network weight is W . The constant parameters b and γ are introduced to ensure that the states $x(t)$ remain bounded, non-diminishing, and ideally exhibit rich patterns, facilitating subsequent extraction. The output weight is derived from minimizing a loss function. Given a

time series data set $z(t)$, which is typically to be normalized for ensuring the model fairness and stability. Once the RC system is established by fixing the choice of all other parameters f , $W^{(\text{in})}$, W , γ , and b during the training process, the output weight can be obtained as follows:

$$W^{(\text{out})} = \arg \min_w (\|Xw - z\|^2 + \beta\|w\|^2), \quad (4)$$

where $x = [x(1)^\top, x(2)^\top, \dots, x(T)^\top]^\top$, $z = [z(1)^\top, z(2)^\top, \dots, z(T)^\top]^\top$, and $\beta \in [0, 1]$. is a penalization term that prevents w from becoming too large. This problem is a form of Tikhonov regularization which yields an explicit solution $(W^{(\text{out})})^\top = (X^\top X + \beta I)^{-1} X^\top z$.

3.3. Next generation reservoir computing

NG-RC requires shorter training datasets and training time compared to RC.²⁵ The NG-RC does not require a neural network; instead, it constructs a feature vector directly from the discretely sampled input data. As shown in Figure 3, $O_{\text{total}} = c \oplus O_{\text{lin}} \oplus O_{\text{nonlin}}$, where $\oplus$ represents the vector concatenation operation. Here, c is a constant bias term, O_{lin} , O_{nonlin} represent linear part and nonlinear parts of feature vector, respectively. Furthermore, the output weight matrix has the same form with RC.

$$W^{(\text{out})} = z O_{\text{total}}^\top (O_{\text{total}} O_{\text{total}}^\top + \beta I)^{-1}. \quad (5)$$

The linear features $O_{\text{lin},i}$ at time i are constructed from observations from input vector X , where $X_i = [x_{1,i}, x_{2,i}, \dots, x_{d,i}]$, d denotes its dimension. These observations are taken at the current time step and at k_1 previous time steps, with each time step spaced by s . Here, s_1 represents the number of skipped steps between consecutive observations. Thus, O_{lin} has dk components and is given by

$$O_{\text{lin}} = X_i \oplus X_{i-s} \oplus X_{i-2s} \oplus \dots \oplus X_{i-(k-1)s}. \quad (6)$$

The nonlinear part O_{nonlin} of the feature vector is composed of nonlinear polynomial features of O_{lin} , where a p^{th} -order polynomial feature vector can be formulated as:

$$O_{\text{nonlin}} = O_{\text{lin}} \otimes O_{\text{lin}} \otimes \dots \otimes O_{\text{lin}_p}, \quad (7)$$

with O_{lin} appearing p times, where $\otimes$ denotes the tensor product, from which only the unique monomials are collected into a vector. Another advantage of NG-RC over regular RC is that it requires a shorter warm-up period of sk time steps, while traditional RC needs a longer warm-up period to ensure that the reservoir state is independent of the initial conditions. Reducing the warm-up time can be advantageous in scenarios where data are scarce or when it is hard to collect data.

4. Simulation and results

4.1. Dataset generation

To obtain real system datasets, we select four representative chaotic systems,^{43,44} shown in Figure 4, which consists of 5000 data points for each state with time step of $dt = 0.01s$ using the RK23 method in Python.

1) *Lorenz system* : The following equation describes the Lorenz system:

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= x(\rho - z) - y, \\ \dot{z} &= xy - \beta z, \end{aligned} \quad (8)$$

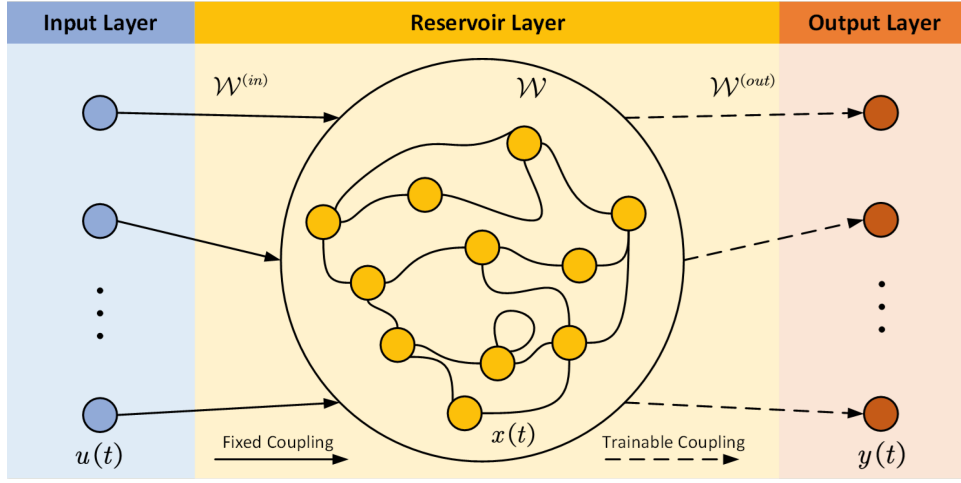


Figure 2. The RC structure, illustrating the input layer, a fixed reservoir layer, and a trainable readout (output) layer

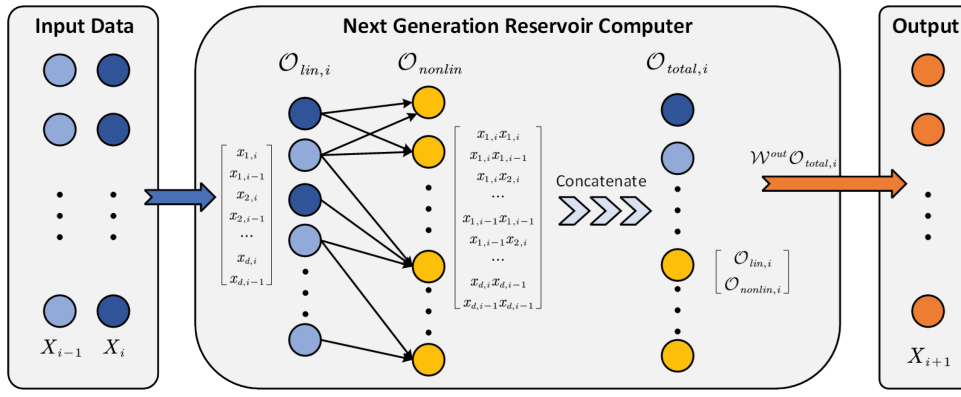


Figure 3. Schematic of the NG-RC framework

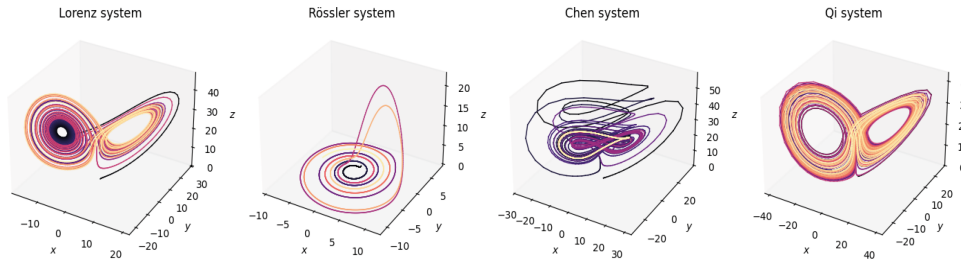


Figure 4. Four representative chaotic systems (a) Lorenz system, (b) Rössler system, (c) Chen system, and (d) Qi system

where constants σ , ρ , β are system parameters. The system exhibits chaotic behavior for these (and nearby) values $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$.

2) *Rössler system* : The following equation describes the Rössler system:

$$\begin{aligned}\dot{x} &= -y - z, \\ \dot{y} &= x + ay, \\ \dot{z} &= (x - c)z + b.\end{aligned}\quad (9)$$

we use the parameters $a = 0.2, b = 0.2, c = 5.7$ for studying the chaotic behavior of the Rössler system.

3) *Chen system* : The Chen system is a double scroll chaotic attractor defined as follows:

$$\begin{aligned}\dot{x} &= a(y - x), \\ \dot{y} &= (c - a)x - xz + cy, \\ \dot{z} &= xy - bz,\end{aligned}\quad (10)$$

we use the parameters $a = 40, b = 28, c = 3$ for studying the chaotic behavior of the Chen system.

4) *Qi system* : The following equation describes the Qi system:

$$\begin{aligned}\dot{x} &= a(y - x) + yz, \\ \dot{y} &= (a - c)x - y - xz, \\ \dot{z} &= xy - bz,\end{aligned}\quad (11)$$

we use the parameters $a = 35, b = 7, c = 10$ for studying the chaotic behavior of the Qi system.

¹ Data used for warming-up RC and NG-RC are contained in train dataset.

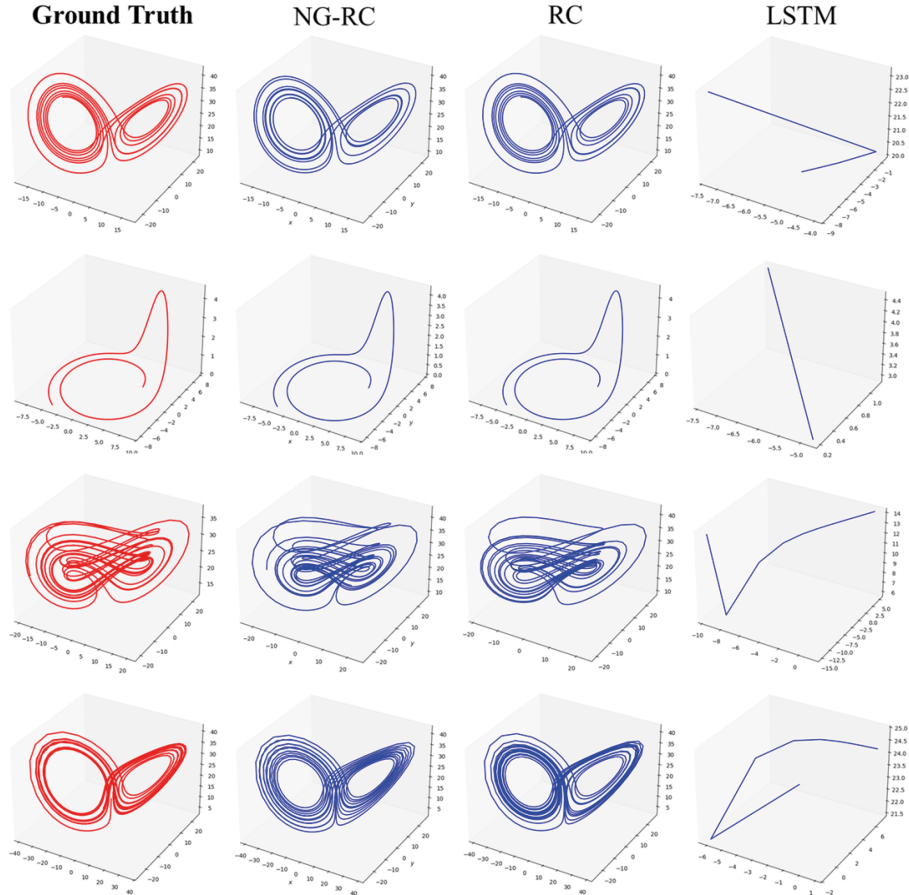


Figure 5. Task 1: Comparison of prediction results from the three methods

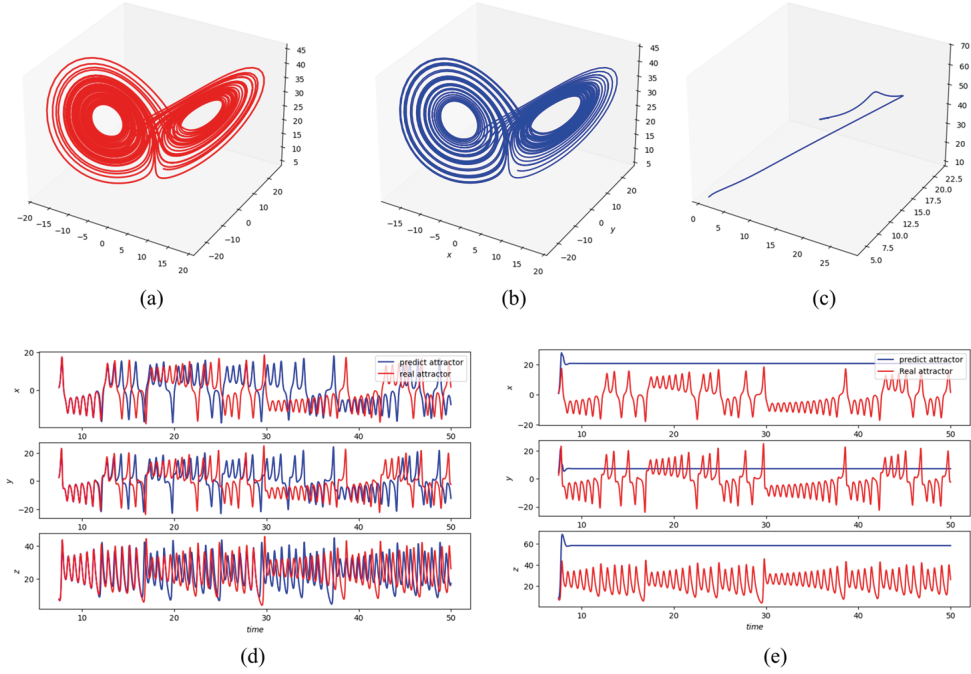


Figure 6. Comparison results of training RC and NG-RC with a smaller dataset. (a) Real attractor, (b) NG-RC predicted attractor, (c) RC predicted attractor, (d) each state of NG-RC predicted attractor (blue), (e) each state of RC predicted attractor (blue).

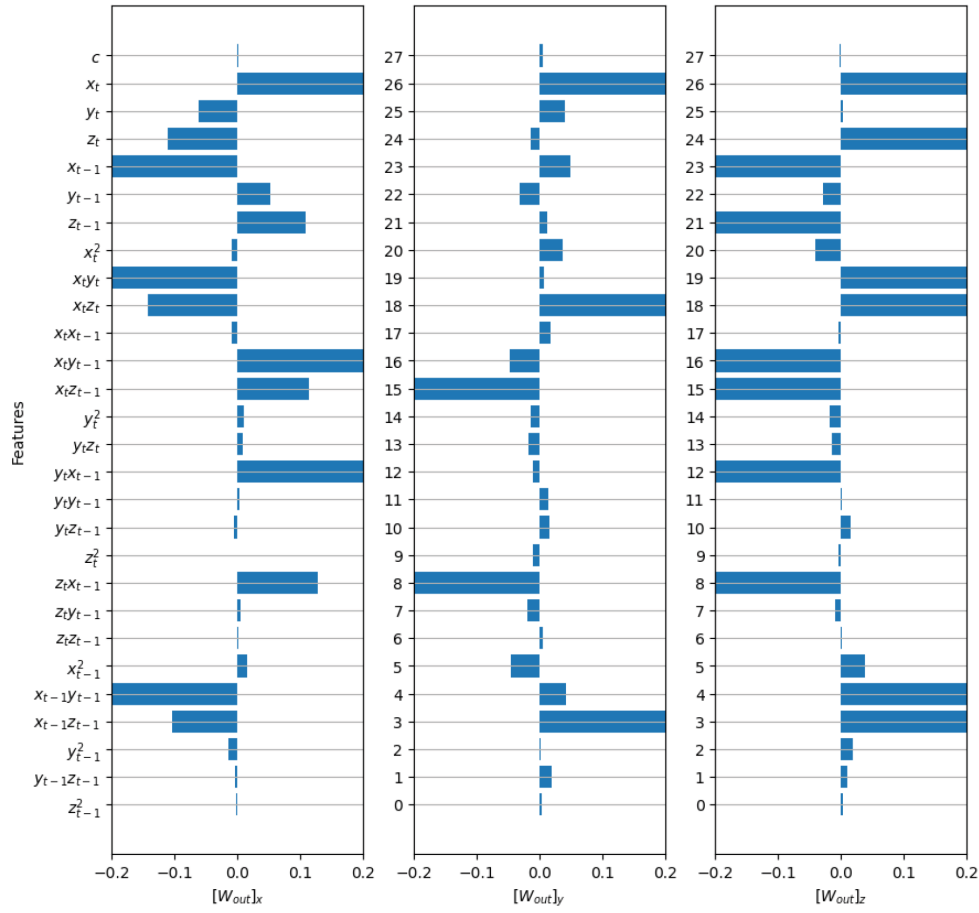


Figure 7. The output weights distribution of the NG-RC in Task 1

To evaluate the performance of these approaches in multi-step forecasting of chaotic time series, the aforementioned methods are applied to predict the four chaotic benchmarks with the same initial condition $[x_0, y_0, z_0] = [17.6771581, 12.9313791, 43.9140433]$.

4.2. Task 1: Multi-steps forecasting of chaotic time series

For Task 1, we divide the dataset into a train and a test dataset¹ with a ratio of 8:2. For the LSTM and RC approaches, each dimension¹ is scaled into $[0, 1]$ using `MinMaxScaler` to ensure training stability and accelerate training convergence. The LSTM network's input and output dimensions are set to 3 with a hidden layer of size 16 and it is trained for 300 epochs. The RC's parameters are set as shown in Table 1. The NG-RC's delay and stride parameters are set to $k = 2$ and $s = 1$ with order $p = 2$. The regularization term $\beta = 2.5 \times 10^{-6}$. After setting up all methods, we run them in Python on a single-core desktop processor. As shown in Figure 5, a qualitative comparison between the NG-RC, RC's predicted results, and the actual system reveals a high degree of similarity, indicating that NG-RC and RC can reproduce the long-term chaotic dynamics process with an adequate training dataset. On the other hand, the results from the LSTM show that it cannot reproduce the chaotic system dynamics even if it is well-trained for single-step forecasting. This makes sense because in a one-step forecasting task, LSTM may perform well since it only needs to input past data. But in this task, the LSTM network needs to use its own output as new input to generate the entire chaotic behavior recursively. Therefore, we conclude that the difference between the

LSTM and the other two methods is that the latter two learn the intrinsic dynamics of the chaotic system, while the former only learns the trend of the data.

Table 1. Hyperparameters of RC

Params	Value
Number of Units	4000
Leak rate	0.3
Spectral radius	1.25
Connectivity	0.1
Regularization	1e-8

Compared with RC, NG-RC shows almost the same predictive ability as RC in this task but is significantly faster, as shown in Table 2. Therefore, we conducted another comparison experiment with RC and NG-RC which uses only a small training dataset to train and predict the rest of the dataset. In this experiment, the first 500 and 250 data points are used for training and warming-up, respectively. The remaining 4250 data points are testing data points used for comparing with the generated attractor. The RC and NG-RC's parameter settings are the same as above. As shown in Figure 6, RC cannot predict the Lorenz chaotic behavior over the entire prediction length due to the lack of a sufficient training dataset. However, it can be seen that the NG-RC can still perform well in a scenario with a small training dataset. Even though a gap exists between the real attractor and the predicted attractor, they are similar in shape, which indicates that NG-RC learns the topology of the Lorenz system and that its model has a

clear interpretability. Other comparison results for Rössler, Chen, Qi system are shown in Appendix (Figures A1–3). The output weights distribution of the NG-RC readout layer is shown in Figure 7.

Table 2. Training time comparison

Time	NG-RC	RC	LSTM
Lorenz	0.46s	0.72s	13.41s
Rössler	0.49s	0.76s	14.05s
Chen	0.50s	0.84s	14.37s
Qi	0.46s	0.75s	13.92s

Also, to better present NG-RC’s superior performance over a relatively long time horizon, we also extended the dynamical system process time to 100 s. In other words, we use a total of 10,000 data points to train and test NG-RC. In this setting, we use 2000 and 3000 points for warm-up and training, respectively. The remaining 5000 points are used for testing. The results are shown in Figure 8.

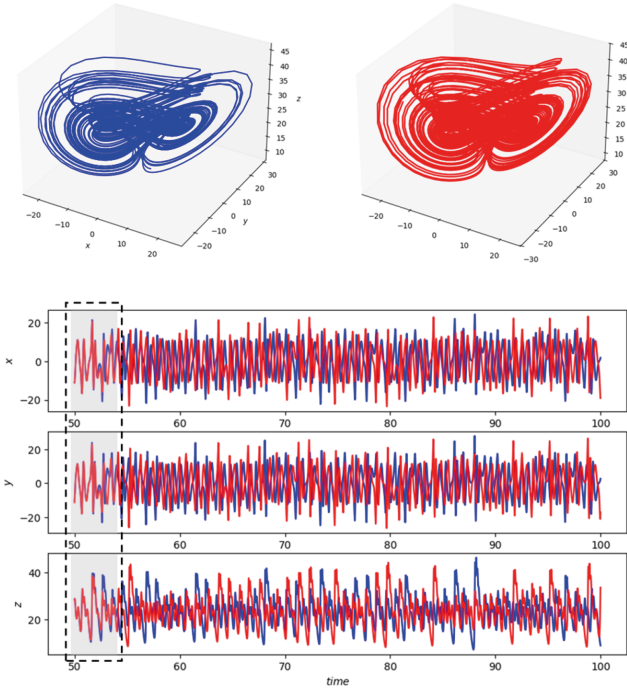


Figure 8. The NG-RC model’s prediction of the Chen system over a long horizon, shown in the gray area, exhibits strong initial performance and demonstrates the model’s forecasting ability

It can be seen that the prediction on the state vector deviates from the ground truth in the plots from each x, y, z dimension, yet from the 3-D state trajectory plot, the prediction captures the overall system behavior adequately. Overall, NG-RC and RC outperform LSTM in prediction when trained with a sufficient training dataset. NG-RC also showed better prediction results in the experiment where RC and NG-RC were trained only with a small dataset, which shows the data efficiency of NG-RC. It is worth mentioning that NG-RC also has an explicit interpretability compared to other methods and shows good prediction results in the long-horizon prediction task. Therefore, problem 1 has been solved.

4.3. Task 2: Predicting system state with noisy input data

To benchmark NG-RC and RC’s robustness on noisy systems, we use the following noisy Lorenz system to generate a dataset with process noise.

$$\begin{aligned}\dot{x} &= 10(y - x) + \sigma\varepsilon_1, \\ \dot{y} &= x(28 - z) - y + \sigma\varepsilon_2, \\ \dot{z} &= xy - \frac{8}{3}z + \sigma\varepsilon_3,\end{aligned}\tag{12}$$

where Gaussian noise is $\varepsilon_i \sim \mathcal{N}(0, 1), i = 1, 2, 3$. σ is the magnitude of the noise. Except for these modifications, the RC and NG-RC’s parameters are set the same as in Task 1. Data points totaling 3750 and 250 are used for training and warm-up, respectively. Prediction results of RC and NG-RC given in Figure 9 show good performance when encountering different noise levels. Even for the Lorenz system with a high level of noise, both RC and NG-RC show similar predictive results that reproduce the Lorenz system. It indicates RC and NG-RC have nearly the same robustness to noise. More experiments with different noise levels are presented in the Appendix (Figures A4–6). So we solved problem 2.

5. Discussion

In this study, three popular machine learning approaches for time-series forecasting—NG-RC, RC, and LSTM—were evaluated on their ability to predict four representative chaotic systems: Lorenz, Rössler, Chen, and Qi.

Our results from the primary forecasting task (Task 1) show that the NG-RC method consistently outperforms the other models. Its advantages are multifaceted, including faster computational speed and fewer adjustable parameters. Unlike RC and LSTM, whose parameter counts can explode dramatically with increasing data dimensions, NG-RC maintains a lightweight network structure. This efficiency does not come at the cost of performance; in fact, NG-RC provides clear model interpretability by revealing the weights of linear and nonlinear components, thereby clarifying the impact of each on the final model. Furthermore, the experiments demonstrated NG-RC’s superior data efficiency. When the training dataset size was significantly reduced, the standard RC model failed to accurately predict the chaotic time series, whereas the performance of NG-RC was largely unaffected.

We also tested NG-RC’s performance over a long time horizon, where it produced satisfactory results in reproducing the system’s overall chaotic behavior. Although the generated attractor was not a perfect one-to-one match with the ground truth, NG-RC successfully learned the fundamental topology of the real attractor. This finding further demonstrates NG-RC’s powerful ability to model complex system dynamics from only a small sample of observable data points.

Moreover, a robustness test was conducted by adding random Gaussian noise to the dynamical systems. In contrast to traditional models that can be highly sensitive to polluted data, both NG-RC and RC proved to be robust. Both methods were able to successfully reproduce the underlying chaotic dynamics even when trained on time series with high levels of noise.

The primary limitation of the standard RC approach is its lack of interpretability; it functions as a “black-box” model with numerous hyperparameters requiring careful adjustment. While a tuned RC model can represent a time series, it offers little insight into the mathematical structure of the data. NG-RC overcomes this by providing a more transparent and efficient framework. Its demonstrated capabilities suggest strong potential for various engineering applications, such as creating digital twins for physical systems from observed data

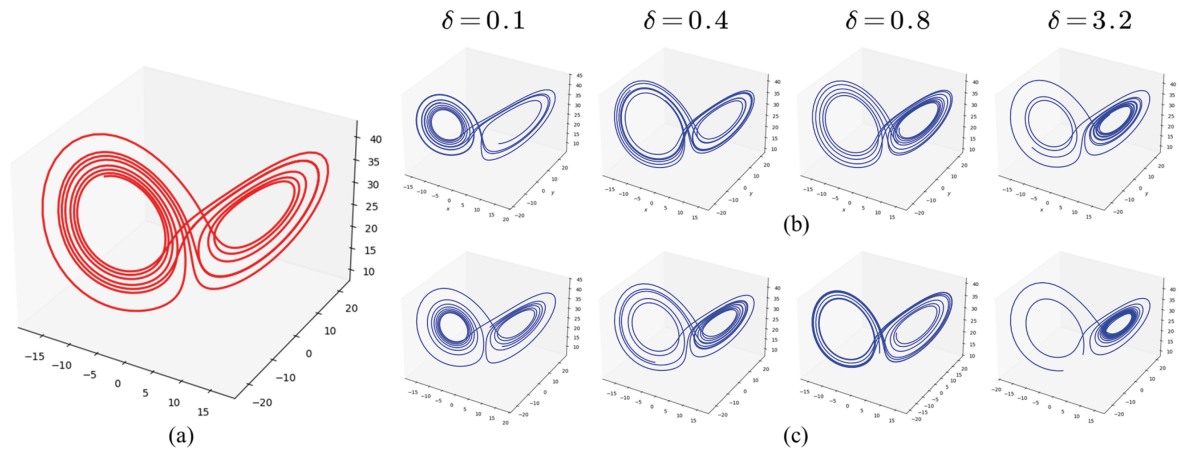


Figure 9. Prediction performance of the noisy Lorenz system for RC and NG-RC with different noise level. (a) Original real attractor, (b) NG-RC's prediction results, (c) RC's prediction results.

or integrating approximate models with real-time observations for data assimilation. When deriving a system's governing equations from first principles is challenging, NG-RC presents a more effective solution than other data-driven methods.

6. Conclusion

This paper demonstrated the superior performance of the NG-RC model for forecasting chaotic time series compared to standard RC and LSTM networks. Our future work will focus on developing efficient methods to predict chaotic systems affected by different types of noise. We also plan to benchmark the ability of various machine learning models to reconstruct unseen system states from partially observable data, a challenge with significant implications for control systems engineering.

Acknowledgments

None.

Funding

None.

Conflict of interest

The authors declare they have no competing interests.

Author contributions

Conceptualization: Huaiyuan Rao, Yichen Zhao

Methodology: Huaiyuan Rao, Yichen Zhao

Writing—original draft: All authors

Writing—review & editing: All authors

Availability of data

Not applicable.

AI tools statement

All authors confirm that no AI tools were used in the preparation of this manuscript.

References

- Roondiwala M, Patel H, Varma S, et al. Predicting stock prices using LSTM. *Int J Sci Res.* 2017;6(4):1754-1756.
- Sheth D, Shah M. Predicting stock market using machine learning: best and accurate way to know future stock prices. *Int J Syst Assur Eng Manag.* 2023;14(1):1-18.
- Price I, Sanchez-Gonzalez A, Alet F, et al. Probabilistic weather forecasting with machine learning. *Nature.* 2025;637(8044):84-90.
- Fan J, Zhang K, Huang Y, Zhu Y, Chen B. Parallel spatiotemporal attention-based TCN for multivariate time series prediction. *Neural Comput Appl.* 2023;35(18):13109-13118.
- Alsharef A, Aggarwal K, Sonia, Kumar M, Mishra A. Review of ML and AutoML solutions to forecast time-series data. *Arch Comput Methods Eng.* 2022;29(7):5297-5311.
- Li G, Jung JJ. Deep learning for anomaly detection in multivariate time series: approaches, applications, and challenges. *Inf Fus.* 2023;91:93-102.
- Torres JF, Hadjout D, Sebaa A, Martínez-Álvarez F, Troncoso A. Deep learning for time series forecasting: a survey. *Big Data.* 2021;9(1):3-21.
- Kong X, Chen Z, Liu W, et al. Deep learning for time series forecasting: a survey. *Int J Mach Learn Cybern.* 2025:1-34.
- Salman D, Direkoglu C, Kusaf M, Fahrioglu M. Hybrid deep learning models for time series forecasting of solar power. *Neural Comput Appl.* 2024;36(16):9095-9112.
- Han M, Xi J, Xu S, Yin FL. Prediction of chaotic time series based on the recurrent predictor neural network. *IEEE Trans Signal Process.* 2004;52(12):3409-3416.
- Chandra R, Zhang M. Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction. *Neurocomputing.* 2012;86:116-123.
- Dudukcu HV, Taskiran M, Taskiran ZGC, Yildirim T. Temporal convolutional networks with RNN approach for chaotic time series prediction. *Appl Soft Comput.* 2023;133:109945.
- Shi H, Xu M, Li R. Deep learning for household load forecasting—a novel pooling deep RNN. *IEEE Trans Smart Grid.* 2017;9(5):5271-5280.
- Ahmed NK, Atiya AF, Gayar NE, El-Shishiny H. An empirical comparison of machine learning models for time series forecasting. *Econometric Rev.* 2010;29(5-6):594-621.
- Wang L, Dai L. Chaotic time series prediction of multi-dimensional nonlinear system based on bidirectional LSTM model. *Adv Theory Simul.* 2023;6(8):2300148.

16. Yanan G, Xiaoqun C, Bainian L, Kecheng P. Chaotic time series prediction using LSTM with CEEMDAN. *J Phys Conf Ser.* 2020;1617:012094.
17. Langeroudi MK, Yamaghani MR, Khodaparast S. FD-LSTM: a fuzzy LSTM model for chaotic time-series prediction. *IEEE Intell Syst.* 2022;37(4):70-78.
18. Sangiorgio M, Dercole F. Robustness of LSTM neural networks for multi-step forecasting of chaotic time series. *Chaos Solitons Fractals.* 2020;139:110045.
19. Pathak J, Hunt B, Girvan M, Lu Z, Ott E. Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. *Phys Rev Lett.* 2018;120(2):024102.
20. Rafayelyan M, Dong J, Tan Y, Krzakala F, Gigan S. Large-scale optical reservoir computing for spatiotemporal chaotic systems prediction. *Phys Rev X.* 2020;10(4):041037.
21. Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L, Zdeborová L. Machine learning and the physical sciences. *Rev Mod Phys.* 2019;91(4):045002.
22. Yan M, Huang C, Bienstman P, Tino P, Lin W, Sun J. Emerging opportunities and challenges for the future of reservoir computing. *Nat Commun.* 2024;15(1):2056.
23. Vlachas PR, Pathak J, Hunt BR, et al. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Netw.* 2020;126:191-217.
24. Bollt E. On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD. *Chaos.* 2021;31(1):013123.
25. Gauthier DJ, Bollt E, Griffith A, Barbosa WA. Next generation reservoir computing. *Nat Commun.* 2021;12(1):1-8.
26. Madondo M, Gibbons T. Learning and modeling chaos using LSTM recurrent neural networks. Proceedings of the Midwest Instruction and Computing Symposium, Duluth, Minnesota; 2018:6-7.
27. Li Q, Lin R-C. A new approach for chaotic time series prediction using recurrent neural network. *Math Probl Eng.* 2016;2016:3542898.
28. Wang R, Peng C, Gao J, Gao Z, Jiang H. A dilated convolution network-based LSTM model for multi-step prediction of chaotic time series. *Comput Appl Math.* 2020;39(1):30.
29. Sangiorgio M. Deep learning in multi-step forecasting of chaotic dynamics. In: Special Topics in Information Technology. Cham: Springer International Publishing; 2022:3-14.
30. Sun Y, Zhang L, Yao M. Chaotic time series prediction of nonlinear systems based on various neural network models. *Chaos Solitons Fractals.* 2023;175:113971.
31. Lu X-Q, Tian J, Liao Q, et al. CNN-LSTM based incremental attention mechanism enabled phase-space reconstruction for chaotic time series prediction. *J Electron Sci Technol.* 2024;22:100256.
32. Serrano-Perez JDJ, Fernández-Anaya G, Carrillo-Moreno S, Yu W. New results for prediction of chaotic systems using deep recurrent neural networks. *Neural Process Lett.* 2021;53(2):1579-1596.
33. Shi L, Wang H, Wang S, Du R, Qu S-X. Predicting nonsmooth chaotic dynamics by reservoir computing. *Phys Rev E.* 2024;109(1):014214.
34. Luo H, Du Y, Fan H, Wang X, Guo J, Wang X. Reconstructing bifurcation diagrams of chaotic circuits with reservoir computing. *Phys Rev E.* 2024;109(2):024210.
35. Akiyama T, Tanaka G. Computational efficiency of multi-step learning echo state networks for nonlinear time series prediction. *IEEE Access.* 2022;10:28535-28544.
36. Tavakoli SK, Longtin A. Boosting reservoir computer performance with multiple delays. *Phys Rev E.* 2024;109(5):054203.
37. Bo YC, Wang P, Zhang X. An asynchronously deep reservoir computing for predicting chaotic time series. *Appl Soft Comput.* 2020;95:106530.
38. Na X, Ren W, Xu X. Hierarchical delay-memory echo state network: a model designed for multi-step chaotic time series prediction. *Eng Appl Artif Intell.* 2021;102:104229.
39. Na X, Zhang M, Ren W, Han M. Multistep-ahead chaotic time series prediction based on hierarchical echo state network with augmented random features. *IEEE Trans Cogn Dev Syst.* 2022;15(2):700-711.
40. Cheng X, Liu Z, Fang H. Deep fusion of discrete-time and continuous-time models for long-term prediction of chaotic dynamical systems. *Nonlinear Dyn.* 2025;113(11):12545-12563.
41. Chepuri R, Amzalag D, Antonsen T, Girvan M. Hybridizing traditional and next-generation reservoir computing to accurately and efficiently forecast dynamical systems. *Chaos.* 2024;34(6):063123.
42. Yang L, Pang S, Zhang Y, et al. Improved next generation reservoir computing with time decay factor and kernel function. *Chaos Solitons Fractals.* 2025;198:116514.
43. Lü J, Chen G, Cheng D, Celikovskiy S. Bridge the gap between the Lorenz system and the Chen system. *Int J Bifurc Chaos.* 2002;12(12):2917-2926.
44. Qi G, Chen G. Analysis and circuit implementation of a new 4d chaotic system. *Phys Lett A.* 2006;352(4-5):386-397.

Appendix

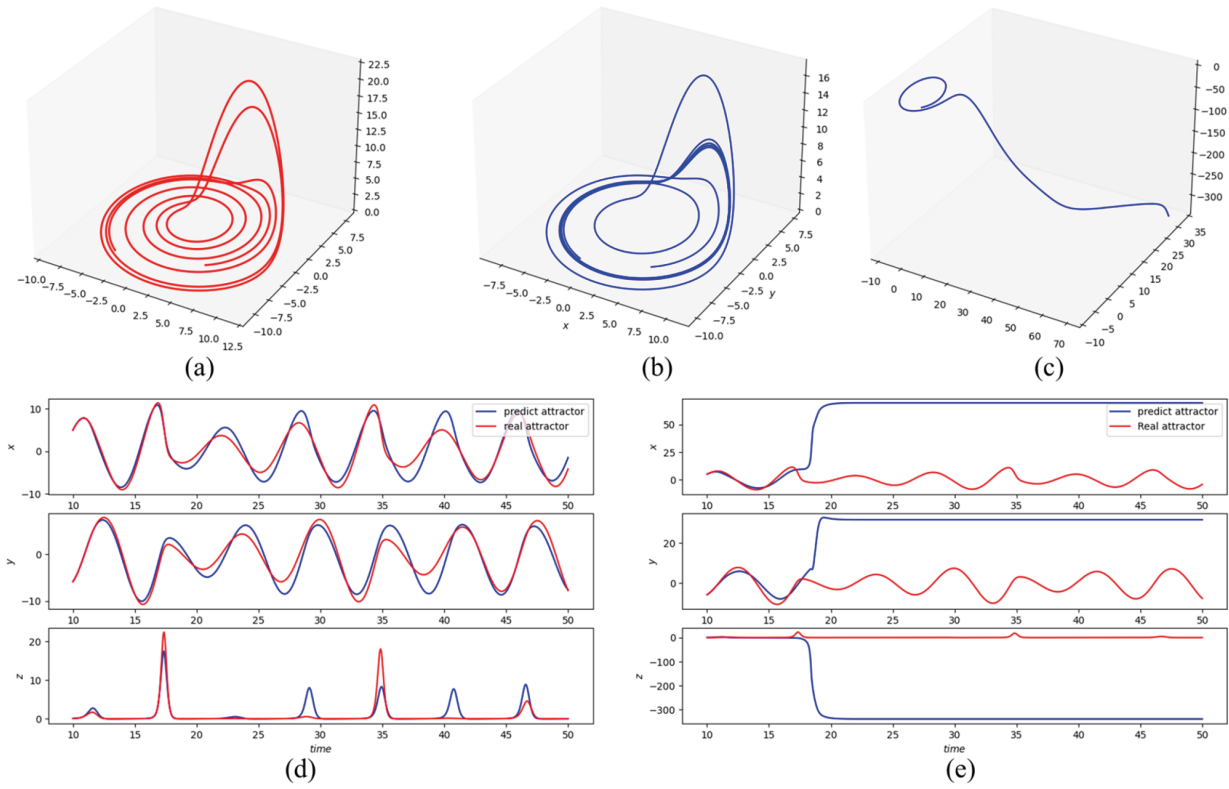


Figure A1. Comparison results of training RC and NG-RC with a smaller dataset of the Rössler system (750 data points for training and 250 data points for warming-up). (a) Real attractor. (b) NG-RC predicted attractor. (c) RC predicted attractor. (d) Each state of NG-RC predicted attractor (blue). (e) Each state of RC predicted attractor (blue).

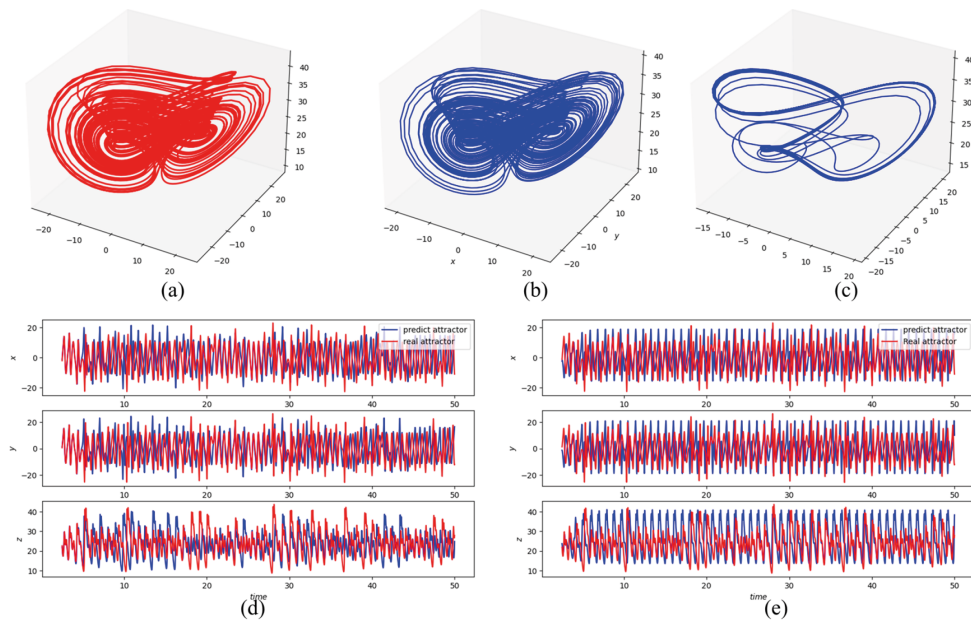


Figure A2. Comparison results of training RC and NG-RC with a smaller dataset of the Chen system (150 data points for training and 100 data points for warming-up). (a) Real attractor, (b) NG-RC predicted attractor, (c) RC predicted attractor, (d) each state of NG-RC predicted attractor (blue), (e) each state of RC predicted attractor (blue).

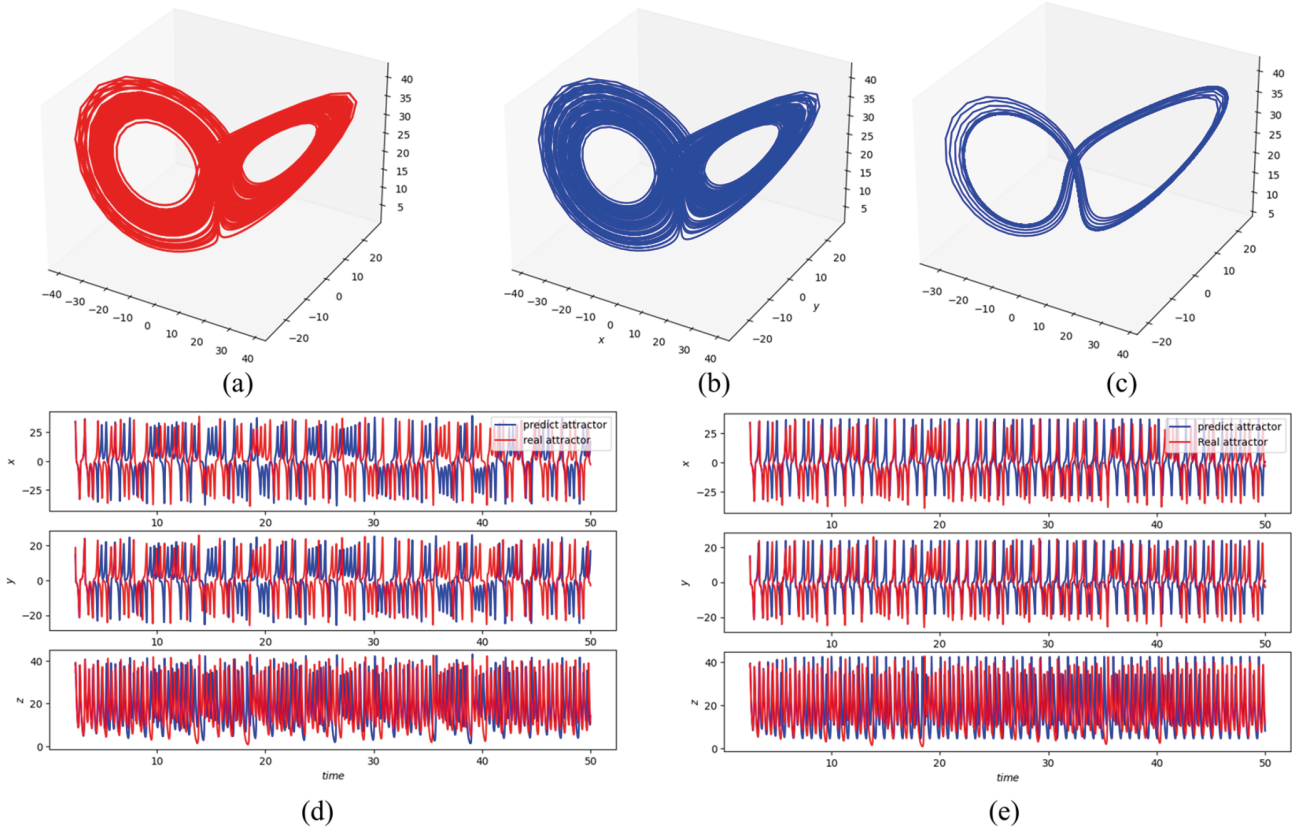


Figure A3. Comparison results of training RC and NG-RC with a smaller dataset of the Qi system (150 data points for training and 100 data points for warming-up). (a) Real attractor, (b) NG-RC predicted attractor, (c) NG-RC's prediction, (d) each state of NG-RC predicted attractor (blue), (e) each state of RC predicted attractor (blue).

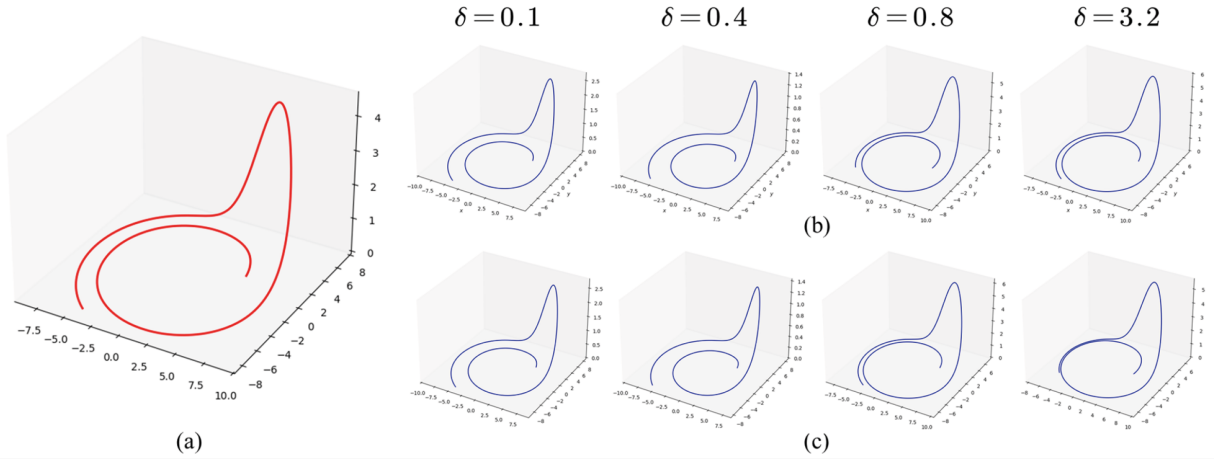


Figure A4. Prediction performance of the noisy Chen system for RC and NG-RC with different noise levels. (a) Original real attractor, (b) NG-RC's prediction, (c) RC's prediction results.

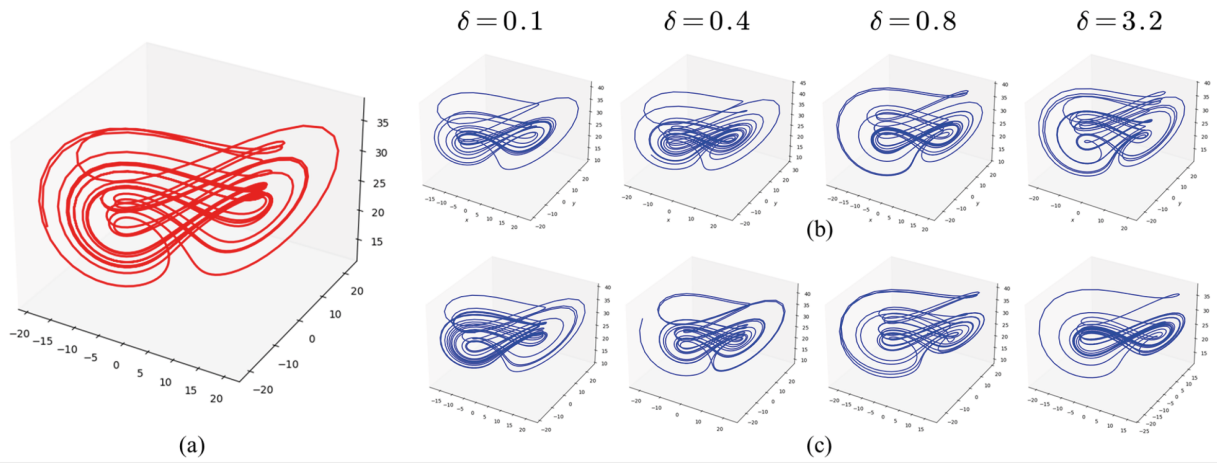


Figure A5. Prediction performance of the noisy Qi system for RC and NG-RC with different noise levels. (a) Original real attractor, (b) NG-RC's prediction results, (c) RC's prediction results.

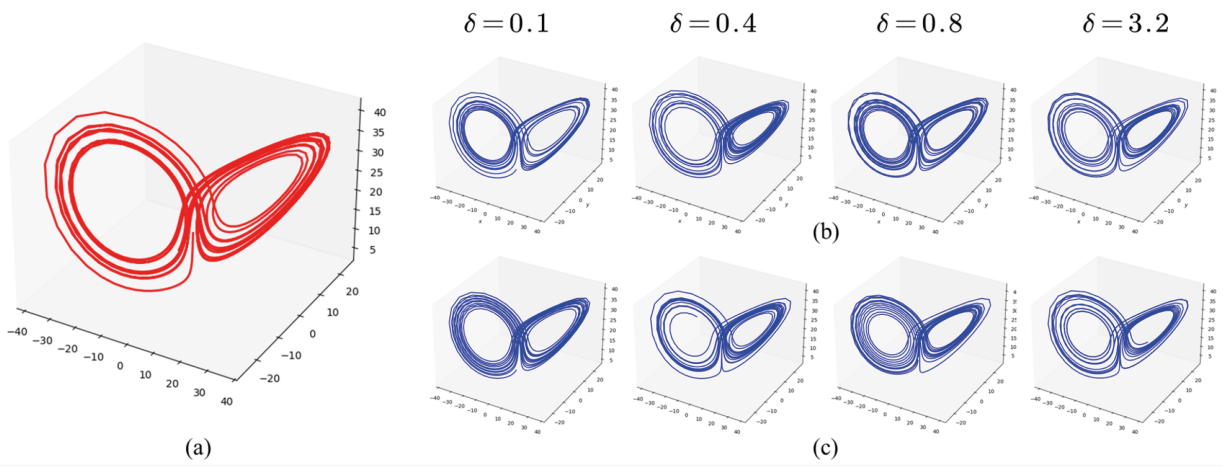


Figure A6. Prediction performance of the noisy Qi system for RC and NG-RC with different noise levels. (a) Original real attractor, (b) NG-RC's prediction results, (c) RC's prediction results.