Reservoir Computing for Time-Series Prediction

Swati Chauhan[1,2], Pratibha Verma[2] and Manish Dev Shrimali[2*]

[1]Graduate School of Engineering, Nagoya Institute of Technology, Nagoya 466 8555, Japan
[2]Department of Physics, Central University of Rajasthan, Ajmer 305 817, India
[*]*shrimali@curaj.ac.in*

(Submitted 14-03-2026,
Accepted 09-06-2026)

Abstract

Reservoir computing (RC) is a machine learning framework that utilizes a nonlinear system (often called a reservoir) to process sequential data. In this framework, the reservoir is driven by an input signal, and readouts from the reservoir are trained to generate a desired output signal. Here, we focus on the implementation of the echo state network (ESN) of RC and demonstrate its application for predicting chaotic time series of two representative systems: the Lorenz system and the logistic map. We observe that the ESN framework accurately learns the system's short-term time series while also capturing its long-term dynamical characteristics. We provide a brief overview of the RC framework and illustrate its potential application for time-series forecasting

1 Introduction

Today, Artificial Intelligence (AI) is an important field of research, permeating every scientific and technological discipline, while also playing a significant role in our daily lives. Remarkable progress has been made in the field in the last few decades. Notably, machine learning (ML), a key subset of AI, which combines principles from computer science and mathematics, has transformed the idea of how we approach complex problems and harness the power of data effectively. Today, ML is integrated across all areas of technology, affecting any technology user with its applications. For example, advances in ML algorithms have significantly enhanced natural language processing, facilitating the development of powerful language models that are capable of generating human-like text [1], improving web search [2] and powering facial recognition software [3].

Many modern ML applications employ the artificial neural networks (ANNs) framework to process information. Inspired by biological neural networks (BNNs), which represent neural connections in the

human brain, ANNs can learn complex relationships across large datasets as they evolve. Due to the availability of ANNs to uncover hidden patterns in data without relying on manually curated features, they are applied over a wide range of applications, ranging from healthcare to agriculture, including speech and image recognition, and time series forecasting [4].

Numerous architectures of ANN have been proposed, which are broadly classified as feedforward neural networks (FNNs) and recurrent neural networks (RNNs). Early applications of NNs often assume that input data samples are independent. While this assumption is valid in many scenarios, these networks restrict results to static, time-independent systems and cannot effectively process sequential data, such as time series or texts. To address these limitations, RNNs were used to capture the temporal dependencies in the data. The feedback mechanism in RNNs enables them to capture information from earlier inputs and maintain it within a hidden state, which can then be utilized for future predictions. Their loop-based architecture allows learning sequential patterns. Despite the remarkable computational power of RNNs for modeling sequential data, the RNNs training process suffers from exploding and vanishing gradients during backpropagation. This limits the network's ability to capture long-term relationships within data. In the early 2000s, fundamentally new approaches to RNNs design were introduced independently by Herbert Jaeger as echo state network (ESN) [5], architecture, which consists of sparsely connected sigmoidal response neurons arranged in a random network architecture. Similarly in the field of computational neuroscience Maass introduced liquid state machines (LSM) [6]. These approaches are collectively referred to as Reservoir Computing (RC) [7]. This study explores the RC paradigm, with particular emphasis on the ESN framework, and evaluates its performance in forecasting nonlinear time-series generated by two prototypical chaotic dynamical systems: the Lorenz system and the logistic map.

Time-series forecasting has been addressed using a wide range of statistical, machine learning, and deep learning approaches. Traditional statistical methods, such as the autoregressive integrated moving average (ARIMA) model, are effective for linear and stationary time series but often struggle to capture the nonlinear characteristics of complex dynamical systems [8]. More recently, deep learning architectures such as long short-term memory (LSTM) networks, gated recurrent units (GRUs), recurrent neural networks (RNNs), and transformer-based models have demonstrated remarkable performance in modeling temporal dependencies and long-range correlations in sequential data [9, 10]. However, these methods typically require extensive training, large datasets, and significant computational resources. In contrast, RC offers an efficient alternative by training

only the readout layer while maintaining competitive predictive performance, particularly for nonlinear and chaotic dynamical systems.

2 Reservoir Computing

RC was introduced as a conceptually simple and computationally efficient alternative for training RNNs. RC is specially designed for processing temporal and sequential data generated by dynamical systems. An RC architecture typically consists of three main components: an input layer, a reservoir layer, and an output layer, also called the readout layer. The RC framework uses a fixed, randomly initialized RNN as a reservoir to produce the complex, high-dimensional representations of low-dimensional input data. In general, a reservoir comprises a large network of recurrently coupled units, such as artificial neurons or even a physical dynamical system. The central idea of RC is to simplify the training of RNNs by keeping the input-to-reservoir and reservoir internal connections fixed after random initialization, while only the readout layer is optimized using a regression method. This significantly simplifies the overall training procedure and considerably reduces the computational cost of using the RC method. The simplicity of the training process also reduces the risk of overfitting and mitigates the vanishing and exploding gradient problem present in RNN. During the training phase, the reservoir internal states are recorded at each time step as the input signal drives them. These states collectively form a dynamical features space that encodes the input's history. During prediction phase, a well-trained RC framework no longer requires the external input data. It operates as a dynamical system whose trajectory from a given initial condition represents the predicted temporal evolution of the target system, initialized with the same initial conditions.

Since its development, RC has emerged as a powerful and versatile tool for time-series prediction, often achieving state-of-the-art performance on benchmark datasets. Recently, interest in RC has increased significantly, driven by advances in big data and the growing need for real-time processing [7]. Beyond forecasting, RC frameworks have been applied to hidden variable estimation [11], signal classification [12], learning coupling among oscillators [13, 14], predicting multistability [15], and control engineering [16, 17]. In the context of nonlinear dynamical systems, RC provides an effective approach for analyzing complex temporal behavior, bridging machine learning techniques with the mathematical foundations of dynamical systems theory.

3 Echo state Network

This section introduces the ESN architecture and its training process for time series prediction.

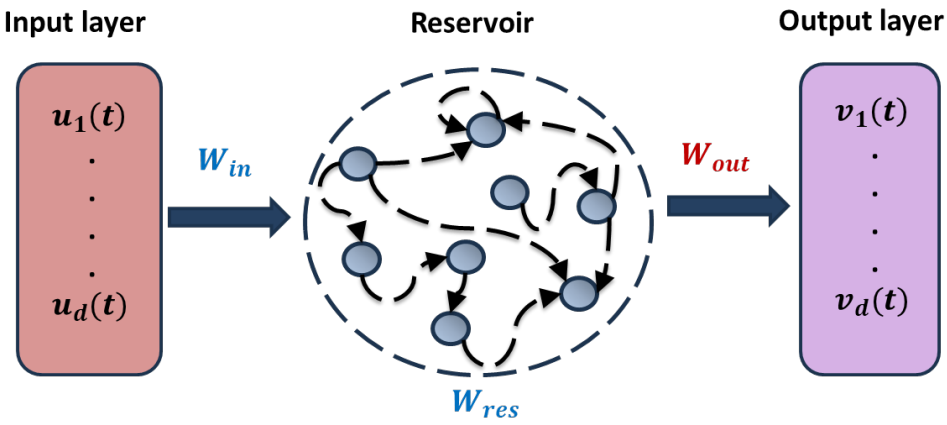

Figure 1: Schematic diagram of echo state network (ESN).

3.1 Architecture

ESN, a prominent RC method, has recently attracted attention for its strong ability to process temporal data [18]. An ESN architecture is composed of three components: an input layer, a reservoir layer, and an output layer. The schematic diagram of the ESN framework is shown in Fig. 1. The (low) d-dimensional input signal is mapped into a (high) N_{res}-dimensional space through the weighted $N_{res} \times d$ matrix $\mathbf{W_{in}}$, often referred to as the input connection matrix. The reservoir layer is composed of a network of N_{res} neurons, characterized by $\mathbf{W_{res}}$, an adjacency matrix of dimension $N_{res} \times N_{res}$. The N_{res}-dimensional signal from the reservoir layer is mapped back into the lower-dimensional output signal via the output weighted matrix $\mathbf{W_{out}}$.

The internal dynamics of the reservoir evolve according to the nonlinear state update equation:

$$\mathbf{r}(t+1) = (1-\alpha)\mathbf{r}(t) + \alpha\mathbf{F}\left(\mathbf{W_{res}}\mathbf{r}(t) + \mathbf{W_{in}}\mathbf{u}(t)\right) \quad (1)$$

Here, $\mathbf{r}(t) \in \mathbb{R}^{N_{\text{res}}}$ is a column vector that represents the reservoir states at time t, and $\mathbf{u}(t) \in \mathbb{R}^d$ is the input data. Here $\mathbf{F}$ is the activation function that introduces nonlinearity, allowing the reservoir to capture complex temporal patterns.

Training phase : During the training phase, the reservoir layer is excited with input data $\mathbf{u}(t)$ for a time length from t= 0 to $t = T_{train}$, where T_{train} is the number of data points in the training set. The internal states of the reservoir $\mathbf{r}(t)$ are calculated at each time step using Eq. 1. The column vector $\mathbf{r}(t)$ is collected at each time step and is stored in a reservoir matrix $\mathbf{R}$ after discarding a few transient time steps T_{trans}. Thus $\mathbf{R}$ is a matrix of dimension $N_{res} \times (T_{train} - T_{trans})$. Notably, the initial state of column vector $\mathbf{r}(t)$ is set as 0. The target data corresponding to the input signal is also collected

in a $\mathbf{T_{teach}}$ over the same training length after discarding the transient time steps. Now the objective is to determine the weights in the readout layer $\mathbf{W_{out}}$ such that

$$\mathbf{v}(t) = \mathbf{W_{out}}\mathbf{r}(\mathbf{t}) \tag{2}$$

here $\mathbf{v}(t) \in \mathbb{R}^d$ is the predicted output vector. The weights of $\mathbf{W_{out}}$ are obtained to minimize the difference between $\mathbf{T_{teach}}$-$\mathbf{W_{out}}\mathbf{r}(t) \approx 0$. We apply ridge regression and calculate the weights of the readout layer using the expression:

$$\mathbf{W}_{out} = \mathbf{T}_{teach}\mathbf{R}\left[\mathbf{R}^T\mathbf{R} + \beta I\right]^{-1} \tag{3}$$

where β is the regularization parameter and I is the identity matrix.

Prediction phase : During prediction phase the reservoir states $\mathbf{r}(t)$ are updated using Eq. 1. The reservoir states are collected after discarding a warmup time steps and are used to calculate output $\mathbf{v}(\mathbf{t})$ using Eq. 2. The only difference during prediction from training is that now the network operates autonomously such that the predicted output is used as the input in Eq. 1, i.e., $\mathbf{u}(\mathbf{t}+\mathbf{1}) = \mathbf{v}(\mathbf{t})$.

3.2 Generation of Matrices

As seen in the previous section, during the ESN training process, the input and reservoir weight matrices, $\mathbf{W_{in}}$ and $\mathbf{W_{res}}$, are randomly initialized. We first discuss the generation of random matrix $\mathbf{W}_{res} \in \mathbb{R}^{N_{res}\times N_{res}}$.

- The reservoir weight matrix $\mathbf{W_{res}}$ is constructed by first generating an initial matrix $\mathbf{W_0} \in \mathbb{R}^{N_{res}\times N_{res}}$ that represents the connectivity between the nodes of the reservoir.
- The entries of $\mathbf{W}_0$ are assigned random weights with a specified connection probability p.
- Next, the maximum absolute eigenvalue of $\mathbf{W_0}$, denoted as ρ_0, is computed.
- Finally, the matrix is rescaled as $\mathbf{W_{res}} = \frac{\rho}{\rho_0}\mathbf{W_0}$, ensuring that the largest absolute eigenvalue of the matrix $\mathbf{W_{res}}$ is ρ.

Next, the input weight matrix $\mathbf{W_{in}} \in \mathbb{R}^{N_{res}\times d}$ is generated such that the nonzero elements of $\mathbf{W_{in}}$ are sampled from a uniform distribution in the interval $[-\sigma, \sigma]$. In each column of $\mathbf{W_{in}}$, N_{res}/d elements are chosen to be nonzero in such a way that each reservoir node receives input from only one variable of the input vector.

3.3 Hyperparameters Optimization

The ESN architecture involves several hyperparameters, including the spectral radius ρ, the link probability p governing the reservoir network's connectivity, the scaling factor of the input weight matrix σ, the leakage parameter α, and the regularization coefficient β. These hyperparameters are selected before training and remain fixed throughout the learning process. The values of these hyperparameters significantly affect the predicted output; therefore, optimizing them

before making predictions is crucial. For instance, the reservoir size, or the number of neurons in the reservoir layer N_{res}, can be viewed as a proxy for the computational resources allocated to the task. In general, increasing N_{res} tends to improve the performance of the ESN scheme. Similarly, α, the leakage parameter, defines the reservoir's characteristic time scale. The spectral radius ρ affects the memory of the reservoir. There are several optimization strategies, such as grid search, random search, Bayesian optimization [19], and so on.

4 Time Series Prediction

To demonstrate the effectiveness of ESN for processing temporal data, we consider time-series prediction of the Lorenz system and the logistic map as representative examples. We aim to predict the time series of both the Lorenz system and the logistic map in their chaotic regimes. The sensitive dependence of these systems on initial conditions makes it difficult to accurately predict their trajectories for long time intervals in a chaotic regime [20]. Here, we focus on predicting time series at short time steps while reproducing the attractor over longer times.

4.1 Lorenz system

The Lorenz system is one of the most well-known chaotic systems in nonlinear dynamics. It was originally introduced by Edward Lorenz while studying atmospheric convection [21]. The extreme sensitivity of the system to its initial condition gives rise to the well-known concept of the butterfly effect, where even small differences at the start can lead to significantly different trajectories over time. The Lorenz system is governed by the following set of ordinary differential equations

$$\begin{aligned} \frac{dx}{dt} &= \sigma_l(y - x), \\ \frac{dy}{dt} &= x(\rho_l - z) - y, \\ \frac{dz}{dt} &= xy - \beta_l z, \end{aligned} \tag{4}$$

where σ_l, ρ_l, and β_l are system parameters. In a chaotic regime, the commonly used parameter values in the Lorenz system are $\sigma_l = 10$, $\rho_l = 28$, and $\beta_l = 8/3$. The Lorenz system in Eq. (4) is numerically integrated using the fourth-order Runge–Kutta method with time step of $dt = 0.01$. The original time series of the variables (x, y, z) of the Lorenz system are shown in Fig. 2 as a solid line (blue). The corresponding phase space in (x, z) plane is also plotted in Fig. 3 using a solid line (blue).

In order to predict the time series of the Lorenz system, we use the ESN architecture. The input data, consisting of the Lorenz system variables $u(t) = [(x(t), y(t), z(t))]$, is sequentially fed into the reservoir for $T_{train} = 50000$ time steps. At each training step, the reservoir state is updated using Eq. 1, and the resulting reservoir states are collected in the reservoir matrix $\mathbf{R}$ after discarding a few transient steps ($T_{trans} = 100$). The complete training procedure of the ESN is described in Sec. 3. The corresponding target

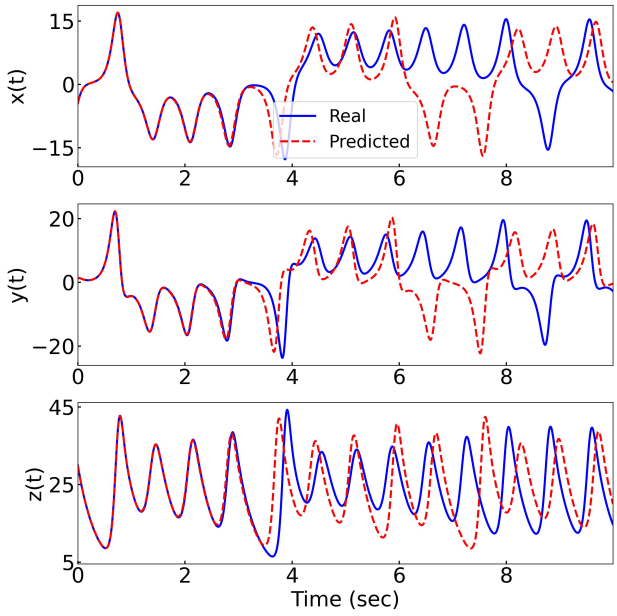

Figure 2: Time series of real (solid blue line) and predicted (dashed red line) Lorenz system variables.

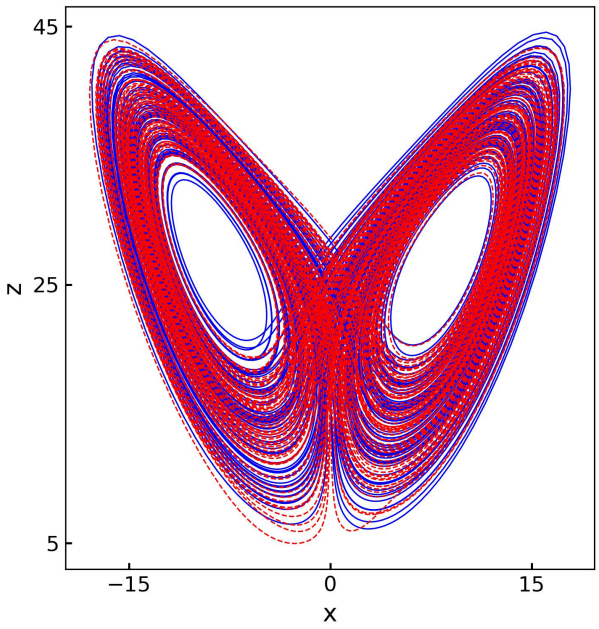

Figure 3: Phase portrait of real (solid blue line) and predicted (dashed red line) Lorenz attractor.

data is also collected in the matrix $\mathbf{T_{teach}}$ over the same training interval after discarding the transient. The readout weights $\mathbf{W}_{out}$ are then optimized. Once the training is completed, the trained reservoir is used to predict the target output $\mathbf{v}(\mathbf{t})$ using Eq. 2. This predicted output vector $\mathbf{v}(\mathbf{t})$, consisting of the predicted state variables of the Lorenz system, is then fed back into the reservoir as the next input, enabling the reservoir to generate future time-series values autonomously. The predicted and original time series of the Lorenz system are presented in Fig. 2. Fig. 3 shows the predicted and original phase space of the chaotic Lorenz attractor. We observe that the short-term predictions are highly accurate, while the system's long-term behavior remains well-preserved. The hyperparameters for this case are [$N_{res} = 500$, $\alpha = 0.75$, $\rho = 0.8$, $p = 50$, $\sigma = 0.27$, $\beta = 0.01$].

4.2 Logistic map

The Logistic map is one of the simplest and most widely studied discrete-time chaotic systems in nonlinear dynamics. It was originally introduced as a mathematical model for population growth [22]. The logistic map is defined by the following nonlinear difference equation

$$x(t+1) = ax(t)(1 - x(t)), \qquad (5)$$

where $x(t)$ represents the state variable at discrete time step t, and a is the control parameter that determines the dynamical be-

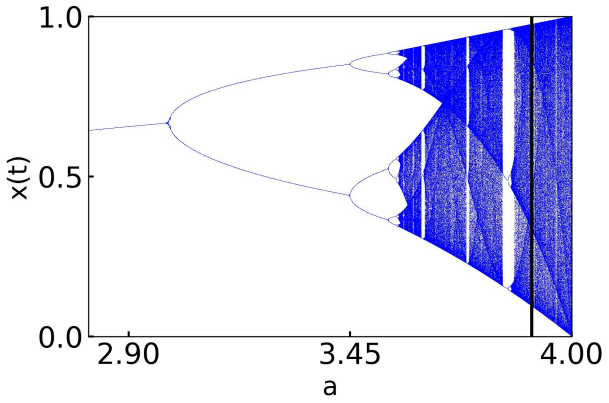

Figure 4: The bifurcation diagram of the logistic map. The solid black vertical line indicates the parameter value a = 3.9.

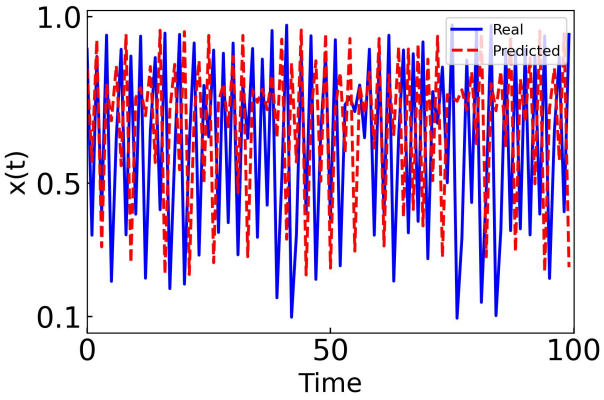

Figure 5: Time series of real (solid blue line) and predicted (dashed red line) logistic map.

havior of the system. The time series of the logistic map is generated using Eq. 5 starting from an initial condition $x(0)$. The original bifurcation diagram for parameter values $a \in [2.8, 4.0]$ is plotted in Fig. 4, illustrating the well-known period-doubling behavior. For certain parameter values, such as $a = 3.9$, the system exhibits chaotic dynamics, as indicated by the vertical solid black line in Fig. 4. The corresponding time series $(x(1), x(2), x(3), \ldots, x(t))$ at $a = 3.9$ generated from an initial condition $x(0)$ is plotted in Fig. 5 using a solid line (blue).

We now employ the ESN architecture to predict the time series of the logistic map. The input data consists of the state variable of the logistic map $\mathbf{u(t)} = [x(t)]$, is sequentially fed into the reservoir for T_{train} = 50000 times steps. At each training step, the reservoir state is updated using Eq. 1, and the resulting reservoir states are collected after discarding $T_{trans} = 100$ time steps. The corresponding target data is also collected in the matrix $\mathbf{T_{teach}}$ over the same training interval after discarding the transient. Once the training process is completed, the optimized readout weights $\mathbf{W_{out}}$ are used for prediction. During the prediction phase, the predicted output vector $\mathbf{v(t)}$, consisting of the state variable of the logistic map $(x(t))$, is fed back into the reservoir as the next input, allowing the reservoir to autonomously generate future time-series values of the logistic map. The predicted and original time series of the logistic map are presented in Fig. 5. The hyperparameters for this case are $[N_{res} = 200, \alpha = 1.0, \rho = 0.8, p = 50, \sigma = 0.27, \beta = 0.01]$.

5 Conclusion

This study presents a pedagogical introduction to ML where the fundamental concepts of RC are introduced, an ML framework, with particular emphasis on the ESN architecture. The work explains the structure of

ESN and the training procedure. To demonstrate the approach's effectiveness in predicting sequential data, a time-series prediction task is performed on two chaotic systems: the Lorenz system and the logistic map. We show that ESN can accurately predict short-term behavior while preserving the long-term dynamics of the systems. The ESN algorithm for time series prediction is also provided in the appendix Sec. 6.

Recently, there has been an increasing interest in incorporating physical knowledge of the system into ML architectures to improve prediction accuracy. Such frameworks are commonly referred to as physics-informed neural networks (PINNs). Motivated by this concept, the prediction capabilities of the standard RC approach can also be enhanced by incorporating a physics-informed model. This approach is referred to as physics-informed RC (PIRC) [23, 24]. In addition, another branch of RC has recently gained significant attention, as it leverages the physical properties and dynamics of a system as a computational substrate. This approach is known as physical RC (PRC) [25, 26]. Given the rapid growth of ML, this work offers a clear, accessible introduction to applying ML techniques to time series prediction.

6 Appendix

ESN algorithm for time-series prediction.

1: **Input data:** Collect training data $\mathbf{u}(\mathbf{t})$ for $t = 1 - T_{train}$ from model system.
2: Generate $\mathbf{W_{in}} \in \mathbb{R}^{N_{res} \times d}$
3: Generate $\mathbf{W_{res}} \in \mathbb{R}^{N_{res} \times N_{res}}$

4: **Training phase**
for $t = 1$ to T_{train}
$\mathbf{r}(t) = (1 - \alpha)\mathbf{r}(t - 1) + \alpha \tanh(\mathbf{W_{res}}\mathbf{r}(t-1) + \mathbf{W_{in}}\mathbf{u}(\mathbf{t}))$
store $\mathbf{r}(t)$ in matrix $\mathbf{R}$
end for
5: Construct teacher matrix $\mathbf{T_{teach}}$ that contains target signal

6: **Learning readout layer weights** : $\mathbf{W_{out}} = \mathbf{T_{teach}}\mathbf{R}(\mathbf{R}^T\mathbf{R} + \beta I)^{-1}$

7: **Prediction phase**
Put $\mathbf{u}(\mathbf{t}) = \mathbf{u}(\mathbf{T_{train}})$ and run reservoir
for $t = 1$ to T_{pred}
$\mathbf{r}(t) = (1 - \alpha)\mathbf{r}(t - 1) + \alpha \tanh(\mathbf{W_{res}}\mathbf{r}(t-1) + \mathbf{W_{in}}\mathbf{v}(\mathbf{t}))$
where $\mathbf{v}(\mathbf{t}) = \mathbf{W_{out}}\mathbf{r}(t)$
end for

Code Availability

The Python implementation of the ESN framework presented in this work is openly available at `https://github.com/2024phdphy004-bot/Reservoir-Computing-for-Time-Series-Prediction.git` to facilitate reproducibility and further research.

References

[1] Devlin, J., Chang, M., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings Of The 2019 Conference Of The North American Chapter Of The Association For Computational Linguistics: Human Language Technologies, Volume 1 (long And Short Papers)*. pp. 4171-4186 (2019)

[2] Guo, X., Singh, S., Lee, H., Lewis, R. & Wang, X. Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. *Advances In Neural Information Processing Systems*. **27**, pp. 3338 - 3346 (2014)

[3] Brunelli, R. & Poggio, T. Face recognition: Features versus templates. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **15**, 1042-1052 (1993)

[4] Lim, B. & Zohren, S. Time-series forecasting with deep learning: a survey. *Philosophical Transactions Of The Royal Society A: Mathematical, Physical And Engineering Sciences*. **379**, 2194 (2021)

[5] Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center For Information Technology Gmd Technical Report*. **148**, 13 (2001)

[6] Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*. **14**, 2531-2560 (2002)

[7] Tanaka, G., Yamane, T., Héroux, J., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D. & Hirose, A. Recent advances in physical reservoir computing: A review. *Neural Networks*. **115**, 100–123 (2019).

[8] Tunnicliffe Wilson, G. Time Series Analysis: Forecasting and Control, 5th Edition, by George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel and Greta M. Ljung. *Journal of Time Series Analysis*. **37**, 286–287 (2016).

[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. Attention Is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008 (2017).

[10] Lu, G., Ou, Y., Wang, Z., Qu, Y., Xia, Y., Tang, D., Kotenko, I. & Li, W. A Survey of Deep Learning for Time Series Forecasting: Theories, Datasets, and State-of-the-Art Techniques. *Computers, Materials and Continua*. **85**(2), 2403–2441 (2025).

[11] Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R. & Ott, E. Reservoir

observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal Of Nonlinear Science*. **27**, 041102 (2017)

[12] Carroll, T. Using reservoir computers to distinguish chaotic signals. *Physical Review E*. **98**, 052209 (2018)

[13] Mandal, S. & Shrimali, M. Learning unidirectional coupling using an echo-state network. *Physical Review E*. **107**, 064205 (2023)

[14] Chauhan, S., Mandal, S., Dixit, S. & Dev Shrimali, M. Predicting collective states of a star network using reservoir computing. *Chaos: An Interdisciplinary Journal Of Nonlinear Science*. **35**, 123112 (2025)

[15] Roy, M., Mandal, S., Hens, C., Prasad, A., Kuznetsov, N. & Dev Shrimali, M. Model-free prediction of multistability using echo state network. *Chaos: An Interdisciplinary Journal Of Nonlinear Science*. **32**, 101104 (2022)

[16] Waegeman, T., Schrauwen, B. & Others Feedback control by online learning an inverse model. *IEEE Transactions On Neural Networks And Learning Systems*. **23**, 1637-1648 (2012)

[17] Mandal, S., Chauhan, S., Verma, U., Shrimali, M. & Aihara, K. Adaptive control of dynamical systems using reservoir computing. *Chaos: An Interdisciplinary Journal Of Nonlinear Science*. **35**, 091102 (2025)

[18] Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*. **3**, 127–149 (2009).

[19] Yperman, J. & Becker, T. Bayesian optimization of hyper-parameters in reservoir computing. *ArXiv Preprint ArXiv:1611.05193*. (2016)

[20] Pathak, J., Wikner, A., Fussell, R., Chandra, S., Hunt, B., Girvan, M. & Ott, E. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal Of Nonlinear Science*. **28**, 041101 (2018).

[21] Lorenz, Edward Norton "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, **20**(2), pp. 130–141, 1963.

[22] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, **261**(5560), pp. 459–467, 1976.

[23] Mammedov, Y., Olugu, E. & Farah, G. Weather forecasting based on data-driven and physics-informed reservoir computing models. *Environmental Science And Pollution Research*. **29**, 24131-24144 (2022)

[24] Perrusquía, A. & Guo, W. Reservoir computing for drone trajectory intent prediction: A physics informed approach. *IEEE Transactions On Cybernetics*. **54**, 4939-4948 (2024)

[25] Nakajima, K. Physical reservoir computing—an introductory perspective. *Japanese Journal Of Applied Physics*. **59**, 060501 (2020)

[26] Mandal, S., Sinha, S. & Shrimali, M. Machine-learning potential of a single pendulum. *Physical Review E*. **105**, 054203 (2022).