Published in final edited form as:
Mach Learn Appl. 2022 June 15; 8: . doi:10.1016/j.mlwa.2022.100300.

Prediction of chaotic time series using recurrent neural networks and reservoir computing techniques: A comparative study

Shahrokh Shahi[a,*],
Flavio H. Fenton[b],
Elizabeth M. Cherry[a]

[a]School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA 30332, United States of America

[b]School of Physics, Georgia Institute of Technology, Atlanta, GA 30332, United States of America

Abstract

In recent years, machine-learning techniques, particularly deep learning, have outperformed traditional time-series forecasting approaches in many contexts, including univariate and multivariate predictions. This study aims to investigate the capability of (i) gated recurrent neural networks, including long short-term memory (LSTM) and gated recurrent unit (GRU) networks, (ii) reservoir computing (RC) techniques, such as echo state networks (ESNs) and hybrid physics-informed ESNs, and (iii) the nonlinear vector autoregression (NVAR) approach, which has recently been introduced as the next generation RC, for the prediction of chaotic time series and to compare their performance in terms of accuracy, efficiency, and robustness. We apply the methods to predict time series obtained from two widely used chaotic benchmarks, the Mackey–Glass and Lorenz-63 models, as well as two other chaotic datasets representing a bursting neuron and the dynamics of the El Niño Southern Oscillation, and to one experimental dataset representing a time series of cardiac voltage with complex dynamics. We find that even though gated RNN techniques have been successful in forecasting time series generally, they can fall short in predicting chaotic time series for the methods, datasets, and ranges of hyperparameter values considered here. In contrast, for the chaotic datasets studied, we found that reservoir computing and NVAR techniques are more computationally efficient and offer more promise in long-term prediction of chaotic time series.

[*]Corresponding author: shahi@gatech.edu (S. Shahi).

CRediT authorship contribution statement
Shahrokh Shahi: Conceptualization, Methodology, Software, Visualization, Writing – original draft. **Flavio H. Fenton:** Resources, Data curation, Supervision, Investigation, Funding acquisition. **Elizabeth M. Cherry:** Conceptualization, Methodology, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Keywords

Recurrent neural networks; Reservoir computing; Echo state networks; Deep learning; Chaotic time series; Nonlinear vector autoregression

1. Introduction

Time series are important in many real-world applications, such as biology (Bar-Joseph et al., 2003), finance (Dingli & Fournier, 2017; Plagianakos & Tzanaki, 2001; Takahashi et al., 2019; Tsay, 2005; Zhao, 2009), climate science (Ghil & Vautard, 1991), anomaly detection in computer networks (Limthong, 2013) and social networks (Gong et al., 2018), and energy (Billinton et al., 1996; Bunn, 2000; Deihimi & Showkati, 2012). Accordingly, the analysis and prediction of time series data are of great importance and have been the focus of much research in the past few decades. In general, a time series represents a record of observations of a dynamical system at specific time intervals. Therefore, time series prediction involves determining the future evolution of a dynamical system, which can be especially challenging for chaotic dynamical systems. The states of such systems can be represented by chaotic time series, which are recognized by the orbital instability characteristic, where infinitesimal differences in the initial values bring about large differences in the time series behavior. Consequently, prediction of a chaotic time series is only feasible for a relatively short time before the appearance of orbital instability. For this reason, forecasting chaotic time series has remained a difficult task for the last few decades.

Data-driven approaches, and machine-learning (ML) techniques in particular, have recently become the main approaches used for time-series forecasting (Ahmed et al., 2010; Ben Taieb et al., 2012; Chandra et al., 2021; Chattopadhyay et al., 2020; Cheng et al., 2015; De Gooijer & Hyndman, 2006; Dubois et al., 2020; Kutz, 2013; Li et al., 2005; Tealab, 2018). In particular, recurrent neural networks (RNNs) are the mainstream architecture for analyzing sequential data, owing to their ability in interpreting temporal dependencies in the input time series (Chandra et al., 2021; Elman, 1990; Elman & Zipser, 1988; Schmidhuber, 2015). The recurrent connections in such networks serve as a notion of memory, allowing them to embed temporal information. Despite the success of RNNs in modeling short-term temporal data and non-chaotic dynamical systems, the high computational cost of back-propagation through time and their vulnerability to the vanishing or exploding gradient problems have limited their applications. Gated RNN architectures were introduced to address some of these problems. More precisely, the memory cell architecture and the gating mechanism enable these networks to be more selective over the information that needs be remembered or forgotten, thereby enabling them to learn long-term dependencies in temporal sequences. Long short-term memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) and gated recurrent units (GRUs) (Chung et al., 2014) are among the most widely used gated RNNs.

An alternative approach to deal with time-series forecasting and modeling dynamical systems is reservoir computing (RC), a learning paradigm mostly implemented as echo state networks (ESNs) (Jaeger, 2002; Lukoševičius & Jaeger, 2009; Sun et al., 2020). The

RC paradigm is fundamentally derived from RNN concepts offering a streamlined training process, which remains limited to obtaining the output layer weights, while the rest of the parameter values are set randomly and remain untrained. Notwithstanding such a major simplification, ESNs have successfully been employed for multi-step-ahead prediction of nonlinear time series and modeling chaotic dynamical systems at low computational cost (Bianchi et al., 2017; Han et al., 2021), triggering the development of several network topologies in recent years. For instance, clustered ESNs (CESNs) (Deng & Zhang, 2006; Junior et al., 2020), where multiple sub-graphs of sparsely connected hidden units form the reservoir, and deep ESNs, where the reservoir consists of multiple sub-reservoir layers stacked hierarchically (Gallicchio & Micheli, 2017; Gallicchio et al., 2017), are two widely used architectures. Hybrid ESNs (HESNs) are another category of RC techniques introduced in a physics-informed ML framework (Oh, 2020; Willard et al., 2020), where additional inputs from physics-based mathematical models integrate corresponding domain knowledge into data-driven models (Doan et al., 2019; Pathak, Hunt, et al., 2018).

The successful application of ESNs, despite their random construction, in forecasting complex dynamical systems using time-series data triggered a series of recent research providing an interpretation of how RC techniques function. Recently, Bollt demonstrated how the RC with linear activation functions and linear readout layer shares similarities with the well-studied vector autoregressive (VAR) concept, while using a quadratic readout can be interpreted as nonlinear VAR (NVAR) (Bollt, 2021). Later, Gauthier et al. further studied this similarity and introduced the next generation RC, where instead of explicitly generating a reservoir of randomly connected neurons, an NVAR machine is formed in which the feature vector consists of time-delayed observations of the dynamical system and is augmented by nonlinear functions of these observations. Accordingly, with this approach there are fewer hyperparameters to tune and the intrinsic random nature of ESNs is effectively avoided. This approach was employed for one-step-ahead forecasting of benchmark chaotic time series for both reconstruction and cross-prediction tasks (Gauthier et al., 2021).

In this work, we assess the capability of the mainstream gated RNN techniques; ESN architectures, including the clustered architecture and the physics-informed hybrid approach; and the NVAR approach for multi-step-ahead prediction of nonlinear time series describing chaotic dynamical systems. In particular, we compare the performance of these models for forecasting two frequently used benchmark chaotic time series, derived from the Mackey–Glass and Lorenz dynamical systems, two additional chaotic times series derived from a bursting Morris–Lecar neuron model and the Vallis El Niño Southern Oscillation (ENSO) system, and one real-world dataset consisting of a time series of irregular cardiac voltage traces obtained in ex-vivo experiments in terms of the prediction error and computational efficiency. Moreover, this experimental dataset is further used to evaluate the performance of NVAR against traditional RC approaches in more detail.

This paper is structured as follows. Section 2 presents a summary of the modeling approaches used for forecasting chaotic time series in this research and provides details about the implementation of each model and the evaluation metrics employed in this study.

These methods are applied to datasets whose characteristics are described in Section 3. The results are presented and discussed in Section 4, and Section 5 presents concluding remarks.

2. Time-series prediction methods

This section presents an overview of the computational methods used for chaotic time series prediction in this work.

2.1. Gated recurrent neural networks

RNNs are one of the most common approaches for handling temporal data; the recurrent connections in the network provide a native way to learn the internal dependencies within the time-dependent data. However, their vulnerability to vanishing and exploding gradients limits their application to only learning short-term relations in the observations. Gated RNNs, such as LSTMs, were introduced as a remedy for such a limitation. The gating mechanism provided by the memory cell architecture enables them to select which information should be kept and which forgotten, making them more robust to irrelevant perturbations. Therefore, gated RNNs can model the temporal dependencies for longer time horizons. Fig. 1a schematically depicts the flow of information in an LSTM cell in which a hidden state h_t is calculated using the following equations:

$$\begin{aligned} i_t &= \sigma(\boldsymbol{W_i} x_t + \boldsymbol{U_i} h_{t-1} + b_i), \\ f_t &= \sigma\left(\boldsymbol{W_f} x_t + \boldsymbol{U_f} h_{t-1} + b_f\right), \\ o_t &= \sigma(\boldsymbol{W_o} x_t + \boldsymbol{U_o} h_{t-1} + b_o), \\ \tilde{c}_t &= \tanh(\boldsymbol{W_c} x_t + \boldsymbol{U_c} h_{t-1} + b_c), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \\ h_t &= \tanh(c_t) \odot o_t, \end{aligned} \tag{1}$$

where i_t, o_t, and f_t denote the input, output, and forget gates at time t; x_t is the input vector; $\boldsymbol{W}_*$ and $\boldsymbol{U}_*$ are the weight matrices that along with the biases b_* are the trainable parameters and are adjusted during the learning process; c_t denotes the internal memory of the LSTM unit known as the cell state; and $\tilde{c}_t$ is the cell input activation vector. In these equations, each $\boldsymbol{\sigma}$ designates a sigmoidal function and $\odot$ denotes Hadamard element-wise multiplication.

A similar desire to avoid vanishing and exploding gradient problems led to the development of GRUs, which share many similarities in architecture and thus performance with LSTMs. As illustrated in Fig. 1b, a GRU memory cell can be considered as a simplified version of an LSTM unit, where the tasks of input and forget gates are handled by a single gate known as the update gate. This simplification improves the overall efficiency as fewer parameters are required to be trained, while the prediction accuracy is minimally affected in most cases and in some applications improvements are even reported (Bianchi et al., 2017). The evolution of hidden states in GRUs is given by the following equations:

$$\begin{aligned} z_t &= \sigma(\boldsymbol{W_z} x_t + \boldsymbol{U_z} h_{t-1} + b_z), \\ r_t &= \sigma(\boldsymbol{W_r} x_t + \boldsymbol{U_r} h_{t-1} + b_r), \\ \tilde{h}_t &= \tanh(\boldsymbol{W_h} x_t + \boldsymbol{U_h}(r_t \odot h_{t-1}) + b_h), \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \end{aligned} \tag{2}$$

where z_t and r_t represent update and reset gates and determine which information should be kept through time and which is irrelevant and can be forgotten. The candidate state is denoted by $\tilde{h}_t$. The weight matrices $\boldsymbol{W}_*$ and $\boldsymbol{U}_*$ and the bias vectors b_* are adjusted in the training process, thereby enabling the update and reset gates to select which information should be passed along to the future, and which information is irrelevant and thus should be forgotten.

2.2. Echo state networks

ESNs are the most common realizations of the RC approach and utilize a low-cost training process in which only the weights of the output layer, known as the readout layer, are adjusted, and the rest of parameters are initialized randomly and remain untrained. Despite this considerable simplification, which turn the training problem into a linear regression task, ESNs provide an effective approach to model and predict complex dynamics, including chaotic time series. Fig. 2a illustrates the main components of a typical ESN, which include an input layer, a hidden layer of randomly connected neurons known as the reservoir, and a readout layer. The number of input and output variables specifies the size of the input and output layers, respectively. In this work, we employ an extension of the standard ESN in which leaky integrator neurons (Jaeger et al., 2007) are employed as the hidden units. The evolution of the reservoir state h_t is described by

$$h_t = (1 - \alpha)h_{t-1} + \alpha \tanh\left(\boldsymbol{W}^{\boldsymbol{in}} x_t + \boldsymbol{W} h_{t-1}\right), \tag{3}$$

where $\boldsymbol{W}^{\boldsymbol{in}}$ and $\boldsymbol{W}$ denote the input weight and reservoir weight matrices, respectively. The input signal is denoted by x_t and $\alpha \in [0, 1]$ is a constant parameter known as the leaking rate. The output of the network is obtained by the following equation:

$$y_t = f^{out}\left(\boldsymbol{W}^{\boldsymbol{out}}[x_t; h_t]\right), \tag{4}$$

where $\boldsymbol{W}^{\boldsymbol{out}}$ denotes the readout weights and is obtained by least-square regression with Tikhonov regularization to prevent overfitting. The activation function of the output layer is given by f^{out} and is chosen here as a unity function.

Once the readout weights are calculated, the future values of the time series can then be obtained using a recursive strategy in which the results of predictions at each time step will be fed to the network as the input for the next time step (see Fig. 2).

The initial success of RC techniques and ESNs motivated further research on the structure of ESNs (Carroll & Pecora, 2019) and new reservoir topologies, such as clustered reservoirs, where the reservoir consists of a set of sub-reservoirs sparsely connected to each other. Fig. 2b demonstrates a CESN, where the reservoir is constructed as three sub-reservoirs. The update equation and training process remain the same as for the baseline ESN.

The physics-informed version of an ESN, known as a hybrid ESN (HESN), has been successfully employed in a number of application domains (Doan et al., 2020; Oh, 2020; Pathak, Wikner, et al., 2018; Shahi et al., 2021) where domain knowledge is integrated into an ESN by feeding the network an additional input from a knowledge-based mathematical model that approximates the behavior of the dynamical system. Fig. 2c illustrates the schematic architecture of a HESN. In this work, in the case of generated time series, an "imperfect" version of the mathematical equations is used to generate the knowledge-based approximation, where the imperfect mathematical equations are obtained by multiplying one of the original model parameters by (1+ϵ), where ϵ represents a dimensionless unknown error (Pathak, Wikner, et al., 2018). In the case of experimental time series, a mathematical model that provides an approximation of the dynamical system is employed.

2.3. Nonlinear vector autoregressive model

It is demonstrated that a reservoir computer with linear activation functions whose feature vector also includes weighted sums of nonlinear functions of the reservoir output values is mathematically comparable to an NVAR model, which consequently offers a powerful universal approximator of dynamical systems (Bollt, 2021; Gauthier et al., 2021). In such an NVAR model, the state matrix is constructed by concatenating a linear part, including k time-delay embeddings of the d-dimensional input time series, and a nonlinear part, which is generated by applying a nonlinear functional (in practice, a polynomial) to the linear part. Therefore, the state vector at step t has the following form:

$$h_t = \left[h_{lin,\,t}; h_{nonlin,\,t}\right], \tag{5}$$

where the linear part $h_{lin,t}$ includes the input signal at time step t and the $k-1$ previous time steps spaced by a parameter s and is given by

$$h_{lin,t} = \left[x_t, x_{t-s}, x_{t-2s}, \ldots, x_{t-(k-1)s}\right]^T. \tag{6}$$

Therefore, s–1 steps are skipped between each two consecutive entries of this vector. The nonlinear part of the hidden vector $h_{nonlin,t}$ is obtained by applying a polynomial functional to the linear part $h_{lin,t}$. For instance, in the case of choosing a quadratic polynomial, the entries of $h_{nonlin,t}$ include the $kd(kd+1)/2$ unique monomials obtained by the cross product of $h_{lin,t}$ with itself and are given by

$$h_{nonlin,\,t} = \left[x_t^2, x_t x_{t-s}, x_t x_{t-2s}, \ldots, x_{t-(k-1)s}^2\right]^T. \tag{7}$$

Then, the rest of the calculation, including finding the readout weights $\boldsymbol{W^{out}}$ and prediction, is identical to what is used for ESNs. Accordingly, sometimes a bias can also be added to the state vector in Eq. (5), i.e. $h_t = [1; h_{lin,t}; h_{nonlin,t}]$. The output of the NVAR method at time step t then is obtained by the following equation:

$$y_t = \boldsymbol{W^{out}}\left[1; h_{lin,\,t}; h_{nonlin,\,t}\right] \tag{8}$$

Thus, this method circumvents the requirement of constructing an explicit reservoir of randomly connected neurons, which increases the randomness and sensitivity of ESNs to the hyperparameter values and initial parameters. In fact, in comparison to ESN, NVAR has fewer hyperparameters to tune whose optimal values can be determined by a grid search or some other optimization technique with less computational effort.

3. Datasets

To evaluate the performance of these approaches in multi-step forecasting of chaotic time series, the methods are applied to predict four chaotic benchmarks and one experimental dataset representing cardiac voltage time series with highly nonlinear dynamics. Below we describe these five datasets.

3.1. Mackey-Glass

As the first example, we use a time series obtained by solving the Mackey–Glass (MG) equation (Mackey & Glass, 1977), which is one of the most commonly studied benchmarks to evaluate chaotic time-series forecasting approaches. The following equation describes the MG time-delay differential system:

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1 + x^c(t-\tau)} - bx(t), \tag{9}$$

where $a = 0.2$, $b = 0.1$, and $c = 10$ are constant parameters. The nonlinearity of the system increases as the time delay parameter τ increases. The system demonstrates chaotic behavior when $\tau \geq 17$. To generate the time series used here, τ is set to 17 and the numerical integration step size is set to $\Delta t = 0.1$ using a fourth-order Runge–Kutta method implemented in MATLAB to solve delay differential equations at discrete equally spaced times. Then, the data is sampled by $10\Delta t$ to form a time series with 15,000 data points split into the training set (80%), where the first 1000 steps are considered the pre-training warm-up period required in RC approaches (Lukoševičius, 2012), and testing set (20%). Fig. 3a illustrates the generated MG dataset; panel (c) shows a blowup of the shaded regions in panel (a) within the training data.

The knowledge-based time series, which is required to evaluate the HESN approach, is generated by an imperfect mathematical model obtained by changing the constant b to $(1 + \epsilon)b$ in Eq. (9), where the error parameter ϵ is set to 0.1 to demonstrate a noticeable difference in the time series values (see Fig. 3c).

3.2. Lorenz

The second chaotic time series benchmark is derived from the 1963 Lorenz system (Lorenz, 1963), which is given by the following differential equations:

$$\begin{cases} \dfrac{dx}{dt} = a(y - x), \\ \dfrac{dy}{dt} = x(b - z) - y, \\ \dfrac{dz}{dt} = xy - cz, \end{cases} \tag{10}$$

where $a = 10$, $b = 28$, and $c = 8/3$ are the constant parameters. The time series is obtained by integrating the equation numerically using ode45, the fourth-order Runge–Kutta solver in MATLAB, where the solution is evaluated at times spaced $\Delta t = 0.01$ apart to obtain a set of 10,000 data points. Then, the time series is scaled to lie in the interval $[-1, 1]$ and divided into training and testing datasets using an 80–20 split. Similar to the MG dataset, the training set includes a 500-step pre-training period required in RC approaches. Fig. 4a illustrates the generated time series.

Similar to the MG dataset, the knowledge-based time series is obtained by replacing the constant b with $(1+\epsilon)b$. The error parameter is set to $\epsilon = 0.05$, which generates an observable difference in the time series values. Fig. 4c shows the difference between the true time series (solid) and the imperfect knowledge-based model time series (dashed).

3.3. Bursting Morris–Lecar

To obtain a third chaotic dataset, we use a busting Morris–Lecar (BML) model of a neuron as described by Izhikevich (Izhikevich, 2012):

$$\begin{cases} \frac{dV}{dt} = -g_l(V - E_l) - g_k w(V - E_k) - g_{Ca} m_\infty(V)(V - E_{Ca}) - u, \\ \frac{dw}{dt} = \lambda(V)(w_\infty(V) - w), \\ \frac{du}{dt} = \mu(V_0 + V), \end{cases} \tag{11}$$

where

$$\begin{aligned} m_{inf}(V) &= (1 + \tanh((V - V_1)/V_2))/2, \\ w_{inf}(V) &= (1 + \tanh((V - V_3)/V_4))/2, \\ \lambda(V) &= \cosh((V - V_3)/(2V_4))/3. \end{aligned}$$

Parameter values were selected to achieve chaotic bursting dynamics as follows: $g_l = 0.5$, $E_l = -0.5$, $g_k = 2$, $E_k = -410$, $g_{C_a} = 1.2$, $E_{C_a} = 1$, $\mu = 0.1$, $V_0 = 0.2$, $V_1 = -0.01$, $V_2 = 0.15$, $V_3 = 0.1$, $V_4 = 0.05$.

The BML time series is generated by solving the differential equations numerically using the forward Euler method implemented in MATLAB, where the time step is set to $\Delta t = 0.01$. The data is then sampled by $10\Delta t$ to form a time series with 15,000 data points divided into training and testing datasets using an 80–20 split, where the first 1000 steps are considered as the pre-training period required in RC techniques. Fig. 5 illustrates the generated dataset. The BML time series is then linearly scaled to lie in the interval [−1, 1].

The knowledge-based time series corresponding to the BML model is obtained by perturbing the constant g_k by a factor of $(1+\epsilon)$ in Eq. (11), where the error parameter ϵ is set to 0.05 to demonstrate a noticeable difference in the time series values (see Fig. 5c).

3.4. El Niño-Southern Oscillation

Another example of a chaotic model is the simple three-variable El Niño-Southern Oscillation (ENSO) model by Vallis (Vallis, 1986). The model represents the ocean as a box with east and west temperatures T_e and T_w, respectively, along with a third variable representing the surface wind (or current) u between the two sides. The ENSO model equations are as follows:

$$\begin{cases} \dfrac{du}{dt} &= \dfrac{B}{2\Delta x}(T_e - T_w) - C(u - u^*), \\ \dfrac{dT_w}{dt} &= \dfrac{u}{2\Delta x}\left(\bar{T} - T_e\right) - A(T_w - T^*), \\ \dfrac{dT_e}{dt} &= \dfrac{u}{2\Delta x}\left(T_w - \bar{T}\right) - A(T_e - T^*), \end{cases} \tag{12}$$

where B describes the rate of flow of current due to the difference in temperatures, Δx is half the width of the ocean, C denotes the frictional flow resistance, u^* is an average current, $\bar{T}$ represents the deep ocean temperature, A scales the rate of heat loss, and T^* functions as an average temperature the ocean aims to maintain. We set the parameter values as $B = 940$, $\Delta x = 7.5$, $C = 3$, $u^* = -14.2$, $\bar{T} = 16$, $A = 1$, and $T^* = 28$. The model has the same structure as the Lorenz model and thus it is capable of producing chaotic dynamics for certain parameter regimes, including the parameter values we chose. In addition, the structural similarity of the ENSO model will allow for an interesting comparison of results with those from the Lorenz model.

The ENSO time series is constructed by applying a forward Euler method implemented in MATLAB to solve the corresponding differential equations (Eq. (12)), where the step size Δt is set to 5×10^{-4} for overall $t = 150$. The final time series, including 30000 data points, is obtained after sampling the data by $10\Delta t$. The first 80% of the time series is assigned to the training dataset, where the first 2000 steps marked as the pre-training warm-up period. The last 20% of the time series forms the testing set (see Fig. 6). Note the values of the ENSO time series are then linearly scaled to be within $[-1, 1]$.

To generate the time series representing the imperfect knowledge-based model, the constant parameter C is perturbed by a factor of $(1+\epsilon)$ in Eq. (12), where the error parameter ϵ is set to 0.05. Fig. 6c demonstrates the knowledge-based ENSO time series (dashed) as opposed to the true ENSO time series (solid).

3.5. Experimental cardiac voltage recordings

Many real-world dynamical systems can show chaotic behavior, including the time evolution of the electrical potential of a cardiac cell (also known as an action potential). To evaluate the capability of these approaches in forecasting real-world chaotic time series, the final dataset we considered represents action potentials recorded from a zebrafish heart as described in Shahi et al. (2021).

For the knowledge-based model, a mathematical model approximating the voltage dynamics of a cardiac cell can be employed. For instance, the two-variable Mitchell–Schaeffer (Mitchell & Schaeffer, 2003) and the three-variable Fenton–Karma (Fenton & Karma, 1998) models are two knowledge-based model candidates. Here we use the Corrado–Niederer modification of the Mitchell–Schaeffer model (Corrado & Niederer, 2016) with $\tau_{in} = 0.3$ ms, $\tau_{out} = 6$ ms, $\tau_{open} = 40$ ms, $\tau_{close} = 20$ ms, and $v_{gate} = 0.13$.

Cardiac cells like those considered here are not natural pacemakers and thus require exogenous stimulation, typically through the direct application of current for a brief time (typically 1–2 ms), to elicit each activation. Thus, information about the pacing stimulus timing must be introduced as an additional input to the network along with the cardiac voltage time series (Shahi et al., 2021). Furthermore, the knowledge-based model also must be stimulated at the same times as the experimental time series. Because the timing of applied stimuli can be variable and is not directly available for the experimental data, a pre-processing step is applied to detect the timestamp at the beginning of each action potential. Then, the pacing stimulus time series is generated by assigning a stimulus voltage with magnitude 0.2 and duration 2 ms at the onset of each beat in time. Fig. 7c exhibits the experimental time series and the corresponding knowledge-based model. Note that the voltage is rescaled to be between zero and one.

4. Implementation

All methods were implemented in MATLAB (R2021a) and were run on the same computer equipped with an Apple M1 processor and 8 GB of RAM, operating with macOS Big Sur (Version 11.5.2).

4.1. Hyperparameter selection

Hyperparameter values used to construct each model play a pivotal role in the performance of each model. Thus, to have a fair comparison between the prediction ability of these techniques, finding a good set of hyperparameters is an inevitable initial step. Here the optimum hyperparameter values were determined by an extensive grid search, with the admissible ranges and the size of the hyperparameter grids informed by initial experiments on a validation set.

In general, gated RNNs, i.e. LSTMs and GRUs, need a large set of tunable hyperparameters, including the number of hidden layers and hidden units, the optimization technique to train the network and the hyperparameters corresponding to the chosen optimization solver, e.g., learning rate, the learning rate drop factor, maximum number of epochs, and regularization factor. Therefore, running an exhaustive search on a wide grid of hyperparameter values is practically infeasible in most cases. In this work, after an initial round of experiments, some of these hyperparameters are set while the values of hyperparameters with more influence on the performance of the networks are optimized by a grid search. In this regard, we use the Adam optimizer (Kingma & Ba, 2014) for training the networks with its default configurations in MATLAB, while the grid search determines the optimum values of number of layers, the dropout probabilities, initial learning rate, maximum number of epochs, and regularization factor (see Table 1).

In comparison to the gated RNNs, ESN approaches are more sensitive to hyperparameter values (Lara-Benítez et al., 2021), while they demand significantly less computational effort for training. Therefore, wider intervals and finer grids were chosen for running the grid search for the ESN techniques. Table 1 presents a summary of the hyperparameters and the parameter grid values used to obtain the best performance for each technique. Note that the number of hyperparameters required to be tuned in the NVAR approach is considerably smaller than for the other techniques.

4.2. Gated RNN implementations

The LSTM and GRU networks were implemented utilizing the Deep Learning toolbox in MATLAB, where the network architectures are defined as a layerGraph object that consists of an array of network layers forming a directed acyclic graph structure. First, a sequential input layer was required to feed the input time series into the network. Then, depending on the architecture, one or multiple LSTM or GRU layers were required to learn the long-term dependencies in the temporal data. Afterward, a fully connected layer connected the last gated RNN layer to a regression output layer. To avoid overfitting issues, dropout layers with various dropout probability were added to the architecture. The number of gated layers, dropout layers, and the corresponding dropout probabilities were hyperparameters and their optimum values were determined in the grid search (see Table 1).

Once the network was trained, the recursive approach was employed to perform a multi-step-ahead prediction, where the predicted response at each time step was provided as input for prediction of the response in the next time step, and the network state was updated correspondingly.

4.3. Echo state network implementations

The implementation of the baseline ESN was based on Jaeger's tutorial introducing ESNs (Jaeger, 2002) and employing the practical remarks suggested by Lukoševičius (Lukoševičius, 2012). To generate the initial reservoir graph in the baseline ESN, the Erdős–Rényi algorithm (Bollobás & Béla, 2001) was used. Then, the reservoir weight matrix was adjusted to satisfy the echo state property of the network (Yildiz et al., 2012) to guarantee that the network was state-forgetting, i.e., the effect of initial conditions should vanish over time to ensure that the reservoir state asymptotically depends solely on the input signal.

The structure of the CESN was very similar to the baseline ESN except for generating the reservoir graph, where the sub-reservoir clusters were generated first and then were connected to each other randomly with an inter-cluster connection probability chosen to be smaller than the intra-cluster connection probability.

Constructing the reservoir in the HESN was identical to the baseline ESN and followed the same Erdős–Rényi approach; the difference was in the input layer, where the sequential input data was augmented with an additional time series from a knowledge-based model synchronized with the original time series (see Fig. 2c).

Note that during the training for the ESNs, the initial states of the network were discarded to ensure that the network dynamics were fully developed and the training was not affected by the initial transient dynamics. This transient phase is exhibited in gray, as the pre-train data, in Figs. 3–7.

Furthermore, by construction, ESNs may suffer from excessive sensitivity to the hyperparameters and the initial values of the parameters, which remain untrained; this property is an outcome of the randomized construction of the networks in RC approaches. To minimize the effect of the random nature of ESNs on the results of our study, the results of each method for each network size were averaged over 10 experiments with different seed values for the random number generator in MATLAB program.

On account of such sensitivity, finding a set of optimum values for hyperparameters is of great importance for network performance. The number of hidden units in the reservoir; the connection probability used in the Erdős–Rényi graph generation process, which determines the sparsity of the reservoir connections; and the spectral radius of the reservoir graph showed the most influence on the performance of the network. Other hyperparameters including the leaking rate, the input weight scale, and the ridge regression regularization factor were also crucial to obtain good prediction results. Moreover, the number of clusters in CESNs along with parameters of the knowledge-based model in the HESNs were additional hyperparameters to be determined. Therefore, in comparison to the gated RNNs, the number of hyperparameters for ESN approaches was considerably higher (see Table 1), leading to a more involved search because the size of the grid search grows exponentially as the parameter ranges grows. Nonetheless, due to the substantially lower computational costs of ESNs, we obtained the grid search results in almost the same wall-clock times as for the RNNs.

4.4. Nonlinear vector autoregressive implementation

The implementation of the NVAR approach is straightforward and shares many similarities with ESNs. There are only a few main considerations in implementing NVAR technique.

First, the degree of the polynomial functional to construct the nonlinear part of the state vectors must be determined; it was shown that employing a low-order polynomial can lead to high prediction accuracy (Gauthier et al., 2021). Accordingly, in this work, we used the simplest case, which is a second-degree polynomial. Therefore, at each time step, the state vector has $1+kd+kd(kd+1)/2$ entries including the bias, linear, and nonlinear parts, respectively.

The second consideration is the pre-training period required in NVAR, which only needs to be ks time steps to have all k-delayed input values to form the linear part of the state vector. Therefore, in practice, the pre-training period in an NVAR can be less than that of an equivalent ESN. Nonetheless, to make these two technique more comparable, in this work, the pre-training periods were chosen to be identical in both techniques.

In contrast to the gated RNNs and ESNs, there are no explicit hidden units in NVAR approach. This makes the comparison of NVAR to the other approaches more challenging. The reason that we include NVAR in this comparative study is that it can perform equally well as optimized ESNs in some applications (Gauthier et al., 2021). For a more fair comparison, the hyperparameters of NVAR were chosen such that the size of the state matrix was almost identical for NVAR and the corresponding ESNs. The number of columns of this matrix is equal to the training length. Therefore, we chose the same length of pre-training periods to ensure the number of columns of the state matrix was the same in both approaches. The number of rows in the state matrix is equal to the number of entries in the state vector at each time step (i.e., $1 + |h_{lin,t}| + |h_{nonlin,t}|$), which is a function of the delay (k) and dimension (d); the latter is fixed in each problem. Thus, for each network size in gated RNNs and ESNs, the number of delays in NVAR was chosen such that the size of the state matrix was almost equal to that of the ESN approaches.

Fixing the delay value k in each NVAR model left only two tunable hyperparameters: the skipping step s and the ridge regression regulation value (see Table 1). Compared to the gated RNN and ESN approaches, the considerably smaller number of hyperparameters together with the reduced computational effort required in each run of training and prediction using NVAR facilitated finding the optimum values of the hyperparameters in a grid search.

Finally, to conduct a multi-step prediction into the future, the same recursive approach was also employed in the case of NVAR method, which means that at each time step, to construct the linear part of the state vector, the prediction of the previous step was introduced as the new input recursively, and the delayed values were fetched from the predictions in the previous steps. Then, the nonlinear terms were computed accordingly.

4.5. Univariate versus multivariate time series

The forecasting approaches for the MG and Lorenz datasets share many similarities; for instance, in the case of applying ESNs to both problems, the constructed networks mirror the schematic architectures shown in Fig. 2. The only main difference is in the number of input and output variables in each time series. The MG dataset is a univariate time series where only one variable needs to be predicted over time. In contrast, the Lorenz dataset is an example of a multivariate time series in which the input layer must accept three input signals for x, y, and z, respectively, i.e. $\mathbf{x_t} = [x(t); y(t); z(t)]$. The same consideration is necessary for the BML and ENSO datasets, where the input time series consist of three input signals. Note that in the HESN approach, the number of inputs for both univariate and multivariate datasets is multiplied by a factor of two due to the additional inputs from the knowledge-based models in each case.

Additional considerations are required to predict cardiac action potential time series, where the dynamics relies on an external stimulus. In particular, the pacing stimulus must also be introduced to the network along with the cardiac voltage signals. In this case, although forecasting the action potentials entails predicting one variable over time, the input signal is a multivariate time series consisting of the action potentials (voltage signal) and the pacing stimulus time series. Accordingly, the network architectures are adjusted to accommodate the additional input from the pacing stimulus time series. This adjustment for the ESNs is portrayed in Fig. 8, where the baseline ESN (Fig. 8a) and CESN (Fig. 8b) are driven with two signals and the HESN (Fig. 8c) is driven with three signals, including one additional input for the knowledge-based time series.

4.6. Evaluation metrics

The prediction accuracy is evaluated using the root mean square error (RMSE) metric, given by

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{d}\sum_{i=1}^{n}(\hat{Y}_{ji} - Y_{ji})^2}, \tag{13}$$

where Y_{ji} and $\hat{Y}_{ji}$ are the target and predicted outputs, respectively, while d and n denote the number of input dimensions and the length of the dataset, respectively. Therefore, for the MG and the experimental cardiac action potential datasets, d is equal to 1, and for the other datasets, d is set to 3.

In the case of the cardiac dataset, action potential duration (APD) provides another meaningful evaluation measure that is used widely in the field. By definition, an APD is the time interval over which the voltage during an action potential is continuously larger than a specified threshold value, which here is chosen as 0.35.

5. Results

5.1. Mackey-Glass dataset

Fig. 9 presents the prediction results of the MG time series by the six described methods (LSTM, GRU, ESN, CESN, HESN, and NVAR) for a fixed network size of 100 hidden units in the first five methods and an equivalent delayed state matrix in NVAR. The results show that despite the general success of gated RNNs in time-series forecasting tasks, they are not capable of capturing the dynamics of this chaotic time series. Both the LSTM and GRU networks show poor results in the prediction phases. The absolute error diagram at the bottom of Fig. 9 further demonstrates the poor performance of the gated RNNs. The best prediction accuracy is obtained by the NVAR approach, where the prediction values exhibit a perfect match for the entire 2000-step testing span. This result is significant considering the fact that NVAR requires considerably less computational effort for constructing and tuning the model. The ESN approaches are in the middle of the prediction performance spectrum, and all three ESN variants can capture the dynamics of the chaotic system and predict the future dynamics for a considerably longer time than the gated RNN approaches. In particular, the baseline ESN can predict the time series accurately, with the least overall absolute error for 1400 steps into the future. The prediction results obtained by the CESN and HESN are accurate for around 200 steps and the discrepancies started after 3 periods, as can be seen in the absolute error diagram, where the relative error exceeds 10%. As a result, the prediction error using the baseline ESN is nearly half that obtained using the other two ESNs for almost all network sizes tested. However, compared to the HESN, the CESN generally provides a better overall match with less overall absolute error in the predicted values.

Varying the network size demonstrates that increasing the model complexity by increasing the number of hidden units in the first five methods and the number of embedded delays in the NVAR approach has a limited effect on the overall accuracy measured by RMSE. Fig. 10a shows that the prediction error remains roughly constant in the LSTM, GRU, and CESN approaches for a range of network sizes; the ESN and HESN show a noticeable improvement with more hidden units. In the NVAR case, by increasing the delay, which for purposes of comparison we are considering as roughly equivalent computationally to increasing the network size in the other approaches, the accuracy remains almost constant with very minor improvements. However, the largest ESN (500 hidden units) shows better accuracy (nearly 8% less prediction error) compared to the corresponding NVAR model.

According to Fig. 10b, which illustrates the elapsed time for the combination of training and prediction tasks, the NVAR approach is the most efficient technique among all methods tested owing to avoiding the explicit construction of the networks. In particular, NVAR is faster than the ESN and gated RNN techniques by more than one and two orders of magnitude, respectively. This plot also reveals that the elapsed time increases with respect to network sizes in all methods including the NVAR approach, where the model complexity is increased by adding more embedded delays. This trend is expected because in each method, by increasing the model complexity, more computational effort is required for training and prediction.

5.2. Lorenz dataset

The results of applying the six described methods to the Lorenz dataset using the same fixed network size are presented in Fig. 11. The LSTM and GRU models can accurately predict for only a short period of time (around 50 time steps) and the prediction results quickly deviate from the true values, whereas the ESNs can capture the dynamics of the system and provide accurate forecasting for more than 400 time steps. Although the prediction accuracy is roughly the same across all three ESNs for the entire test set, the prediction errors illustrate that the CESN achieved slightly better performance. Nonetheless, similar to the MG dataset, the best overall performance measured by the RMSE metric is obtained by the NVAR approach, which demonstrates that this technique can successfully forecast multivariate time series for multiple steps into the future. The last panel in Fig. 11 shows the mean absolute error of the predicted values obtained by each method. This plot reveals that even though the overall prediction error obtained by NVAR is the minimum across all applied techniques, the prediction obtained by ESN approaches are more accurate at the beginning of the prediction before discrepancies start to appear later.

Increasing the network size for the ESNs can slightly improve their performance in predicting the Lorenz time series, as shown in Fig. 12. In contrast, the LSTM and GRU models do not display this trend; for some intermediate network sizes, the models performed poorly even for the first few time steps. As opposed to the gated RNNs and ESNs, increasing the embedded delays in the NVAR approach significantly improved the prediction accuracy of the model. Accordingly, the discrepancies that currently can be seen in the NVAR prediction results in Fig. 12 do not persist with higher delay values. We study this more using the experimental dataset, as discussed in the next section.

The lower plot in Fig. 12 shows that the NVAR approach is the most efficient with the smallest overall computational times. All three ESN techniques show similar computational times for various reservoir sizes. Although the ESN computational times are larger than for the NVAR approach, they still remain within a factor of 2–5 times more than the corresponding NVAR models. However, the LSTM and GRU models are more than 3 orders of magnitude slower than the NVAR and ESN models in most cases. Another expected trend was the increase in computational times when increasing the model size, which is demonstrated in Fig. 12.

5.3. Bursting Morris–Lecar dataset

Fig. 13 shows the results of using the BML dataset of bursting neural activity with the six different prediction methods. The LSTM and GRU approaches have little if any predictive power and quickly predict constant values. In contrast, NVAR is a particularly good choice with very low error throughout the entire prediction time. The performance of the various ESN approaches fails to match the accuracy of NVAR but provides good predictions for more than half of the prediction time, especially for the ESN and CESN methods. The bottom panel indicates absolute error over time, with the highest values associated with the LSTM and GRU approaches; over time, the error grows appreciably for the ESN methods as well.

For the BML dataset, prediction accuracy as measured by RMSE is relatively insensitive to the network size (or number of embedded delays, for NVAR), as shown in the top panel of Fig. 14 Because of the decrease in prediction quality for the second half of the time series, the ESN approaches have RMSE values almost as high as those of the LSTM and GRU methods. For NVAR, increasing the embedded delays may lead to a decrease in accuracy. For all methods, the computational time needed increases with the network size. The LSTM and GRU methods require the most time, nearly two orders of magnitude more than the ESN methods, which require similar times. NVAR is the most efficient, but the ESN methods remain competitive in terms of computational requirements.

5.4. El Niño-Southern Oscillation Dataset

In Fig. 15, the results of predicting the ENSO dataset can be seen for the different methods. The LSTM method performs particularly poorly, with only a single oscillation before remaining at a constant value. GRU does better but essentially misses all the large-amplitude oscillations in the middle of the testing data and also does not predict the amplitudes and phases of the smaller oscillations consistently. The baseline ESN predicts accurately for more than half of the test data but fails to predict the later large-amplitude oscillations and predicts the last portion of the dataset with proper amplitude but incorrect phase. The HESN method performs similarly, except that the last portion of the dataset has less severe discrepancies in phase and slightly higher amplitude discrepancies. Very good performance is obtained by the CESN method, with almost no difference from the testing data, narrowly beating out NVAR, which includes some slight discrepancies. In the bottom panel of Fig. 15, the absolute error measurements over time confirm the poor predictions using LSTM and GRU throughout the time series and the consistently small errors achieved only for CESN and NVAR.

Fig. 16 compares the performance for the ENSO dataset across different network sizes. RMSE values for the LSTM and GRU methods do not vary meaningfully with network size; however, the ESN methods show a general downward trend in RMSE as the network size is increased, although the effect is not monotonic. For NVAR, RMSE decreases with the number of embedded delays. Note that despite the similarities between the ENSO and

Lorenz models, there are some differences in performance for the corresponding datasets, particularly with improvements for ENSO by the CESN, baseline ESN, and even GRU approaches. Both datasets experience similarly poor performance by the LSTM and good performance by NVAR.

As for computational efficiency, all methods show a trend toward increasing time required for increasing network sizes with the ENSO dataset, as shown in the lower panel of Fig. 16. The LSTM and GRU methods require similar times, which are 2–3 orders of magnitude longer than required for the ESN and NVAR approaches. The NVAR method requires the least time, but the ESN methods remain competitive.

5.5. Experimental cardiac voltage dataset

For the experimental dataset, the gated RNNs and the ESN models can reconstruct the main features of the voltage time series, with the exception of the GRU, as shown in Fig. 17. However, in contrast to the other two datasets, the NVAR model shows the maximum prediction error and very poor prediction results. According to the absolute error values, the results obtained by the HESN appear closest in reconstructing the full voltage trace. Similar behavior can be seen in the APD plots, where the best predictions are obtained by the CESN and HESN models; see Fig. 18. The gated RNNs and the baseline ESN not only have high error but show little of the observed variation in APD. This shortcoming gets worse in the case of NVAR, where the predicted APD values show the poorest agreement with the target values.

As with the other two datasets, the same experiment was repeated with various model complexities to study the influence of network size (or the embedded delays for the NVAR approach) on the prediction abilities of the applied methods and the corresponding computational times (Fig. 19). The variation in the computational times is almost identical to the previous cases, where by increasing the model complexities, the computational time increases. In contrast to the other two datasets, where the NVAR technique was significantly faster than ESNs, for this dataset, the ESNs and NVAR approach exhibit the same range of efficiency except the baseline ESN, where the elapsed time is almost half of the NVAR model in most cases. Nevertheless, the ESNs and the NVAR model are considerably faster than the LSTM and GRU by more than two orders of magnitude.

Fig. 19a indicates that the prediction accuracy of each ESN model remains almost constant for all network sizes, whereas the RMSE values obtained by the ESN and CESN are almost equal and roughly two times more than the HESN, but are still 50% less than for the gated RNN methods. The most interesting trend can be observed for the NVAR approach, where increasing the delay values significantly reduces the prediction errors as measured by RMSE. This finding suggests that it might be possible to improve meaningfully upon the results demonstrated in Fig. 17 by increasing the number of embedded delays and steps skipped between each two consecutive delays. More specifically, the NVAR results presented in Fig. 17 are obtained using only 7 delays ($k = 7$) to have the same number of rows in the corresponding state matrix as for the ESN with 100 hidden units, to improve the fairness of the comparative study. However, the error obtained by 16 embedded delays (same matrix size as the ESN with 500 neurons) is less than 30% of the case with 7 delays.

These observations suggest that although the NVAR model relies on fewer hyperparameters and needs less tuning effort, more embedded delays and consequently more computational effort may be required to obtain a proper prediction accuracy. To examine this hypothesis, the NVAR technique was applied to the same experimental dataset with more delay values. Fig. 20 illustrates the prediction results obtained by NVAR with $k = \{20, 30\}$ and $s = \{5, 10\}$ and shows that increasing the number of embedded delays yields better prediction results. In particular, by choosing $k = 30$ and $s = 10$, the predicted values are nearly identical to those in the test dataset. The same result can be observed in the APD diagrams exhibited in Fig. 21, where agreement is improved and even those cases with discernible error display similar changes in APD. As a comparison of the matrix sizes and the required computational effort, to accommodate the bias, linear and nonlinear parts of the state vectors, the corresponding state matrix includes 1891 rows, which is roughly equal to a state matrix of an ESN with 944 hidden units for the same dataset. However, the NVAR approach is still faster and provides a more accurate prediction as measured by RMSE.

6. Conclusion

In this paper, six different ML time-series forecasting approaches, including two gated RNN techniques, three variants of ESNs, and the NVAR approach, were tested to predict five chaotic time series, including the Mackey–Glass, Lorenz-63, bursting Morris–Lecar, Vallis ENSO, and experimental cardiac action potential time series. Although we considered relatively large but still limited numbers of datasets and methods, we found that the LSTM and GRU approaches, despite their high computational costs and in contrast to the ESN and NVAR methods, were incapable of forecasting the Mackey–Glass, Lorenz, and bursting Morris–Lecar time series more than a few steps into the future, and that increasing the network size did not significantly improve their performance. For the ENSO model, the GRU method could predict somewhat longer, but it did not compare favorably with the ESN and NVAR approaches.

Three variants of ESNs were employed including the baseline ESN, the clustered ESN (CESN), and the hybrid physics-informed ESN (HESN). For the five datasets we used in this work, only one (ENSO) showed improvement by using a more complicated ESN architecture such as the clustered reservoir. In all the other cases, the baseline ESN demonstrated similar or better performance compared to CESN. In contrast, whereas the HESN provided the same level of prediction accuracy for the four synthetic time series, within the tested network sizes, it was the most successful approach for forecasting the experimental dataset, where it delivered more accurate predictions as measured by RMSE. Thus, incorporating the domain knowledge of a dynamical system if available may improve the prediction ability of the ESN technique and may help with obtaining good predictions using smaller network sizes.

For the tested network sizes and datasets, the best prediction performance in the case of the Mackey–Glass, Lorenz, and bursting Morris–Lecar datasets was obtained by the NVAR method, which was recently introduced as the next generation of RC techniques and has been demonstrated to be as successful as optimized ESNs. For the ENSO dataset, NVAR's prediction accuracy was only slightly lower than that of the most accurate method, CESN. A noticeable advantage of the NVAR technique over conventional ESNs is avoiding the explicit construction of randomly connected neurons and circumventing the intrinsic randomness that increases the sensitivity of the network to the hyperparameter values and initial parameters that remain untrained. Moreover, the number of hyperparameters is much smaller than for ESNs, which makes NVAR easier to tune. Such advantages may initially suggest that the amount of data required to train the NVAR model could be less than that needed for the conventional ESNs. However, our experiments showed in the case of the experimental cardiac voltage dataset, better performance was only obtained by embedding more delays and at the cost of more computational time and effort. Nevertheless, in general, this approach shows promise for efficient prediction of chaotic time series. To the best of our knowledge, this work is one of the first applications of this newly introduced technique to real-world experimental time series. Further studies in this area may reveal more of the potential of this approach. For instance, in this work, we used a quadratic polynomial functional to construct the nonlinear portion of the state vectors at each time step; however, other nonlinear functions such as higher-order polynomials could also be employed and studied.

It should also be noted that our conclusions are based on a limited number of datasets and employed methods. Moreover, in each case, the optimum hyperparameters were obtained in a finite grid search process. Accordingly, it is possible that the same approaches could provide different results when applied to other datasets or when hyperparameters are determined using more extensive grids or different optimization techniques.

Acknowledgments

This study was supported by NSF, United States of America grants CMMI-2011280 and CMMI-1762553. We thank Conner J. Herndon for assistance in providing the experimental dataset.

References

Ahmed N, Atiya A, El Gayar N, & El-Shishiny H (2010). An empirical comparison of machine learning models for time series forecasting. Econometric Reviews, 29(5), 594–621. 10.1080/07474938.2010.481556.

Bar-Joseph Z, Gerber GK, Gifford DK, Jaakkola TS, & Simon I (2003). Continuous representations of time-series gene expression data. Journal of Computational Biology, 10(3–4), 341–356. [PubMed: 12935332]

Ben Taieb S, Bontempi G, Atiya A, & Sorjamaa A (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. Expert Systems with Applications, 39(8), 7067–7083. 10.1016/j.eswa.2012.01.039.

Bianchi FM, Maiorino E, Kampffmeyer MC, Rizzi A, & Jenssen R (2017). Other recurrent neural networks models. In Recurrent neural networks for short-term load forecasting (pp. 31–39). Springer.

Billinton R, Chen H, & Ghajar R (1996). Time-series models for reliability evaluation of power systems including wind energy. Microelectronics Reliability, 36(9), 1253–1261.

Bollobás B, & Béla B (2001). Random graphs, no. 73. Cambridge University Press.

Bollt E (2021). On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD. Chaos. An Interdisciplinary Journal of Nonlinear Science, 31(1), Article 013108.

Bunn DW (2000). Forecasting loads and prices in competitive power markets. Proceedings of the IEEE, 88(2), 163–169.

Carroll TL, & Pecora LM (2019). Network structure effects in reservoir computers. Chaos. An Interdisciplinary Journal of Nonlinear Science, 29(8), Article 083130. 10.1063/1.5097686.

Chandra R, Goyal S, & Gupta R (2021). Evaluation of deep learning models for multi-step ahead time series prediction. IEEE Access, 9, 83105–83123. 10.1109/ACCESS.2021.3085085.

Chattopadhyay A, Hassanzadeh P, & Subramanian D (2020). 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 in Geophysics, 27(3), 373–389.

Cheng C, Sa-Ngasoongsong A, Beyca O, Le T, Yang H, Kong Z, & Bukkapatnam S (2015). Time series forecasting for nonlinear and non-stationary processes: A review and comparative study. IIE Transactions (Institute of Industrial Engineers), 47(10), 1053–1071. 10.1080/0740817X.2014.999180.

Chung J, Gulcehre C, Cho K, & Bengio Y (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Corrado C, & Niederer SA (2016). A two-variable model robust to pacemaker behaviour for the dynamics of the cardiac action potential. Mathematical Biosciences, 281, 46–54. 10.1016/j.mbs.2016.08.010. [PubMed: 27590776]

De Gooijer J, & Hyndman R (2006). 25 Years of time series forecasting. International Journal of Forecasting, 22(3), 443–473. 10.1016/j.ijforecast.2006.01.001.

Deihimi A, & Showkati H (2012). Application of echo state networks in short-term electric load forecasting. Energy, 39(1), 327–340.

Deng Z, & Zhang Y (2006). Complex systems modeling using scale-free highly-clustered echo state network. In The 2006 IEEE international joint conference on neural network proceedings (pp. 3128–3135). IEEE.

Dingli A, & Fournier KS (2017). Financial time series forecasting–a deep learning approach. International Journal of Machine Learning and Computing, 7(5), 118–122.

Doan NAK, Polifke W, & Magri L (2019). Physics-informed echo state networks for chaotic systems forecasting. In International conference on computational science (pp. 192–198). Springer.

Doan NAK, Polifke W, & Magri L (2020). Physics-informed echo state networks. Journal of Computer Science, 47, Article 101237.

Dubois P, Gomez T, Planckaert L, & Perret L (2020). Data-driven predictions of the lorenz system. Physica D: Nonlinear Phenomena, 408, Article 132495.

Elman J (1990). Finding structure in time. Cognitive Science, 14(2), 179–211. 10.1016/0364-0213(90)90002-E.

Elman J, & Zipser D (1988). Learning the hidden structure of speech. Journal of the Acoustical Society of America, 83(4), 1615–1626. 10.1121/1.395916. [PubMed: 3372872]

Fenton F, & Karma A (1998). Vortex dynamics in three-dimensional continuous myocardium with fiber rotation: Filament instability and fibrillation. Chaos. An Interdisciplinary Journal of Nonlinear Science, 8(1), 20–47.

Gallicchio C, & Micheli A (2017). Deep echo state network (deepesn): A brief survey. arXiv preprint arXiv:1712.04323.

Gallicchio C, Micheli A, & Pedrelli L (2017). Deep reservoir computing: A critical experimental analysis. Neurocomputing, 268, 87–99.

Gauthier DJ, Bollt E, Griffith A, & Barbosa WAS (2021). Next generation reservoir computing. Nature Communications, 12(1), 5564. 10.1038/s41467-021-25801-2.

Ghil M, & Vautard R (1991). Interdecadal oscillations and the warming trend in global temperature time series. Nature, 350(6316), 324–327.

Gong Q, Chen Y, He X, Zhuang Z, Wang T, Huang H, Wang X, & Fu X (2018). DeepScan: Exploiting deep learning for malicious account detection in location-based social networks. IEEE Communications Magazine, 56(11), 21–27.

Han Z, Zhao J, Leung H, Ma KF, & Wang W (2021). A review of deep learning models for time series prediction. IEEE Sensors Journal, 21(6), 7833–7848. 10.1109/JSEN.2019.2923982.

Hochreiter S, & Schmidhuber J (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. [PubMed: 9377276]

Izhikevich EM (2012). Neural excitability, spiking and burstin. International Journal of Bifurcation and Chaos, 10(6), 1171–1266. 10.1142/S0218127400000840,

Jaeger H (2002). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" Echo State Network" approach, vol. 5. GMD-Forschungszentrum Informationstechnik Bonn.

Jaeger H, Lukoševičius M, Popovici D, & Siewert U (2007). Optimization and applications of echo state networks with leaky- integrator neurons. Neural Networks, 20(3), 335–352. 10.1016/j.neunet.200A7.04.016. [PubMed: 17517495]

Junior LO, Stelzer F, & Zhao L (2020). Clustered echo state networks for signal observation and frequency filtering. In Anais do VIII symposium on knowledge discovery, mining and learning (pp. 25–32). SBC.

Kingma DP, & Ba J (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kutz JN (2013). Data-driven modeling & scientific computation: Methods for complex systems & big data. Oxford University Press.

Lara-Benítez P, Carranza-García M, & Riquelme JC (2021). An experimental review on deep learning architectures for time series forecasting. arXiv preprint arXiv:2103.12057.

Li G, Song H, & Witt S (2005). Recent developments in econometric modeling and forecasting. Journal of Travel Research, 44(1), 82–99. 10.1177/0047287505276594.

Limthong K (2013). Real-time computer network anomaly detection using machine learning techniques. Journal of Advances in Computer Networks, 1(1), 126–133.

Lorenz EN (1963). Deterministic nonperiodic flow. Journal of Atmospheric Sciences, 20(2), 130–141.

Lukoševičius M (2012). A practical guide to applying echo state networks. In Neural networks: Tricks of the trade (pp. 659–686). Springer.

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

Mackey MC, & Glass L (1977). Oscillation and chaos in physiological control systems. Science, 197(4300), 287–289. [PubMed: 267326]

Mitchell CC, & Schaeffer DG (2003). A two-current model for the dynamics of cardiac membrane. Bulletin of Mathematical Biology, 65(5), 767–793. 10.1016/S0092-8240(03)00041-7. [PubMed: 12909250]

Oh DK (2020). Toward the fully physics-informed echo state network–an ode approximator based on recurrent artificial neurons. arXiv preprint arXiv:2011.06769.

Pathak J, Hunt B, Girvan M, Lu Z, & Ott E (2018). Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. Physical Review Letters, 120(2), Article 024102. [PubMed: 29376715]

Pathak J, Wikner A, Fussell R, Chandra S, Hunt BR, Girvan M, & Ott E (2018). Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. Chaos, 28(4), Article 041101. 10.1063/1.5028373. [PubMed: 31906641]

Plagianakos V, & Tzanaki E (2001). Chaotic analysis of seismic time series and short term forecasting using neural networks. In IJCNN'01. International joint conference on neural networks. Proceedings (Cat. No. 01CH37222), vol. 3 (pp. 1598–1602). IEEE.

Schmidhuber J (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117. 10.1016/j.neunet.2014.09.003. [PubMed: 25462637]

Shahi S, Marcotte CD, Herndon CJ, Fenton FH, Shiferaw Y, & Cherry EM (2021). Long-time prediction of arrhythmic cardiac action potentials using recurrent neural networks and reservoir computing. Frontiers in Physiology, 12, 10.3389/fphys.2021.734178.

Sun C, Song M, Hong S, & Li H (2020). A review of designs and applications of echo state networks. arXiv preprint arXiv:2012.02974.

Takahashi S, Chen Y, & Tanaka-Ishii K (2019). Modeling financial time-series with generative adversarial networks. Physica A: Statistical Mechanics and its Applications, 527, Article 121261.

Tealab A (2018). Time series forecasting using artificial neural networks methodologies: A systematic review. Future Computing and Informatics Journal, 3(2), 334–340.

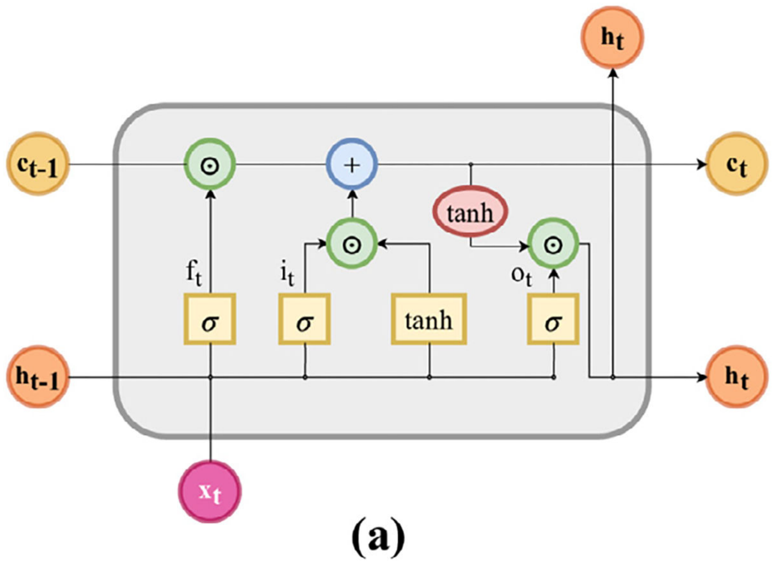
Tsay RS (2005). Analysis of financial time series, vol. 543. John wiley & sons.

Vallis GK (1986). El Niño: A chaotic dynamical system? Science, 232(4747), 243–245, URL https://www.jstor.org/stable/1696890. [PubMed: 17780809]

Willard J, Jia X, Xu S, Steinbach M, & Kumar V (2020). Integrating physics-based modeling with machine learning: A survey. arXiv preprint arXiv:2003.04919.

Yildiz IB, Jaeger H, & Kiebel SJ (2012). Re-visiting the echo state property. Neural Networks, 35, 1–9. [PubMed: 22885243]

Zhao H (2009). A chaotic time series prediction based on neural network: Evidence from the shanghai composite index in china. In 2009 International conference on test and measurement, vol. 2 (pp. 382–385). IEEE.



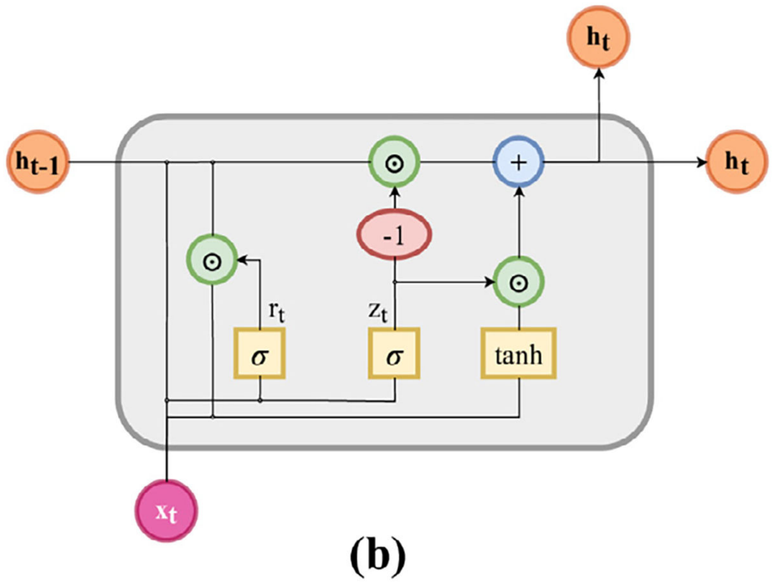


Fig. 1.
Architectures of memory cells in gated recurrent neural networks. (a) Long short-term memory. (b) Gated recurrent unit.

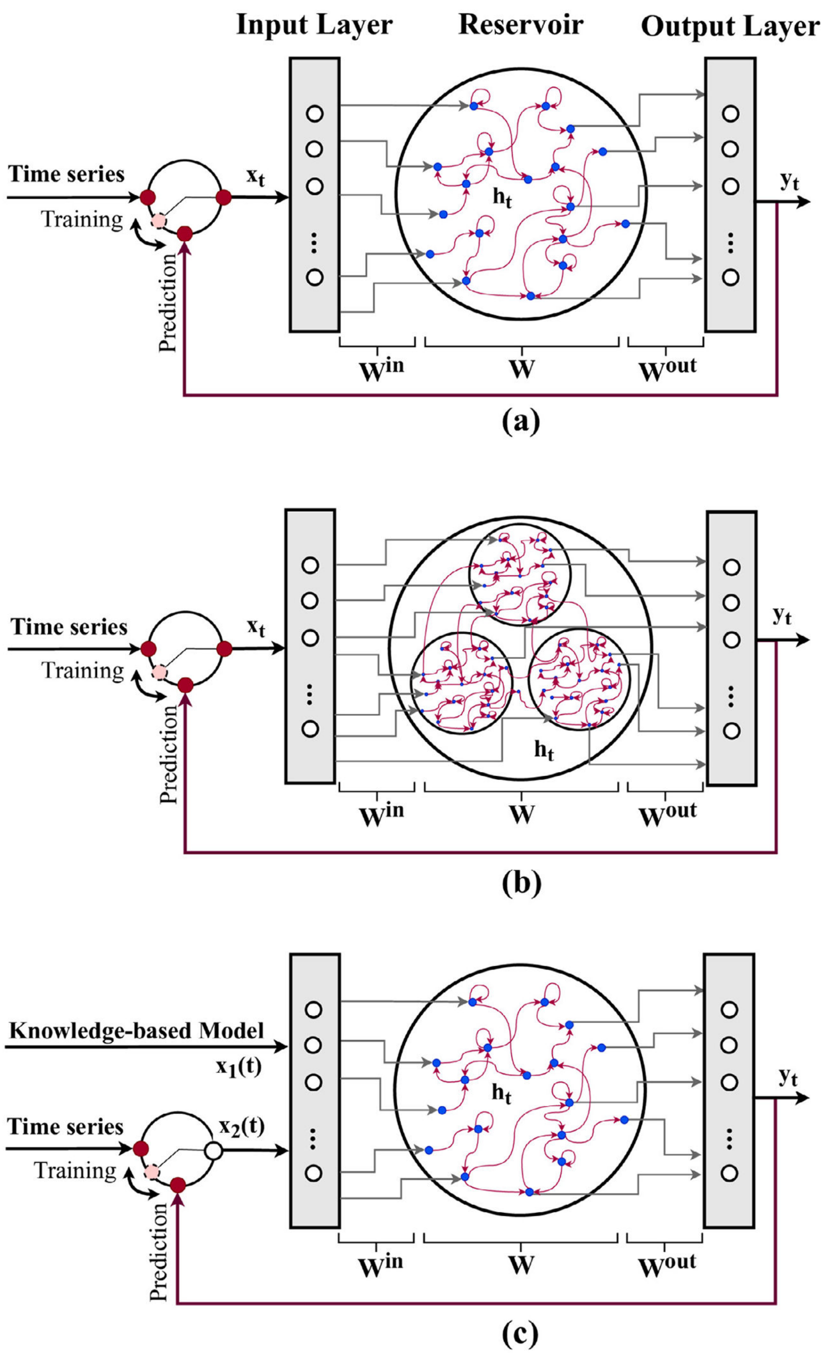


Fig. 2.

Components of reservoir computing approaches, including (a) the baseline ESN, (b) CESN, and (c) HESN. In these architectures, the input and output signals can be either univariate or multivariate time series.

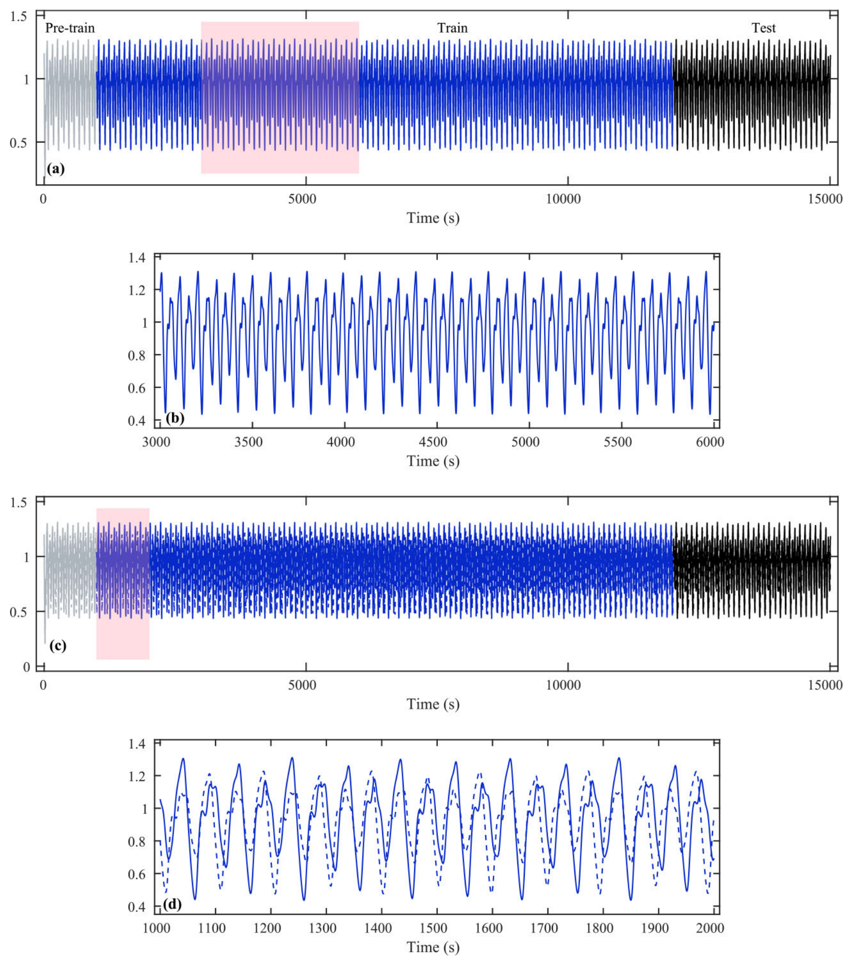


Fig. 3.

The Mackey–Glass time series. (a) Generated time series including unused pre-training data (gray), training data (blue), and testing (prediction) data (black). (b) Zoomed-in section corresponding to the shaded region in panel (a). (c) Mackey–Glass time series (solid) and the imperfect knowledge-based model (dashed). (d) Zoomed-in section corresponding to the shaded region in panel (c) demonstrating the difference between the generated time series and the imperfect knowledge-based model.

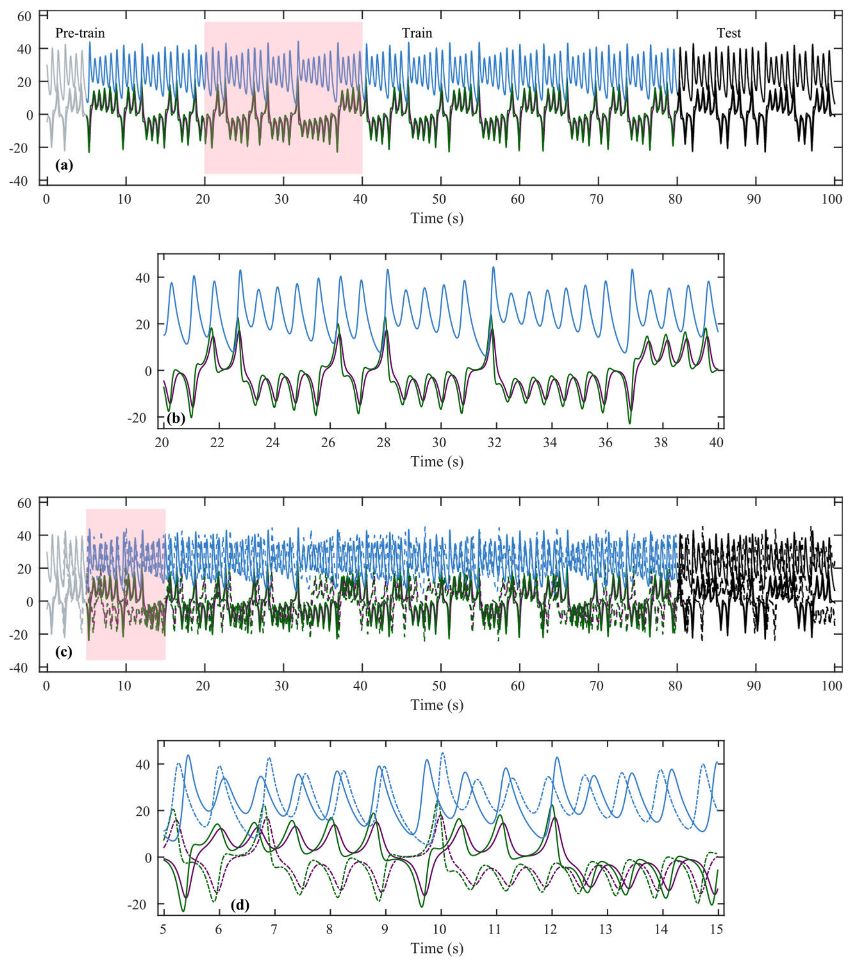


Fig. 4.

The Lorenz system time series. (a) The unused pre-training data is shown in gray, and the training data are in purple, green, and blue colors indicating the *x*, *y*, and *z* variables, respectively. The testing data is in black. (b) Zoomed-in section corresponding to the shaded region in panel (a). (c) Lorenz system time series (solid) and the imperfect knowledge-based model (dashed). (d) Zoomed-in section corresponding to the shaded region in panel (c) demonstrating the difference between the generated time series and the imperfect knowledge-based model.

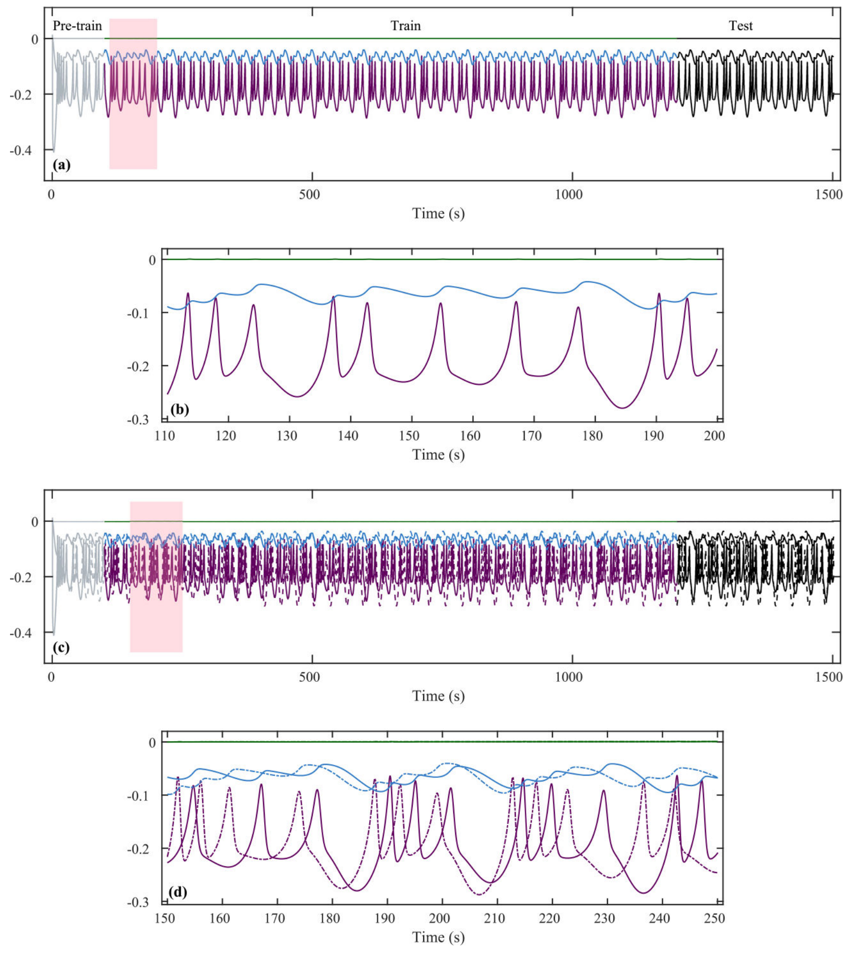


Fig. 5.

The bursting Morris–Lecar time series. (a) The unused pre-training data is shown in gray, and the training data are in purple, green, and blue colors indicating the V, w, and u variables, respectively. The testing data is in black. (b) Zoomed-in section corresponding to the shaded region in panel (a). (c) The bursting Morris–Lecar time series (solid) and the imperfect knowledge-based model (dashed). (d) Zoomed-in section corresponding to the shaded region in panel (c) demonstrating the difference between the generated time series and the imperfect knowledge-based model.

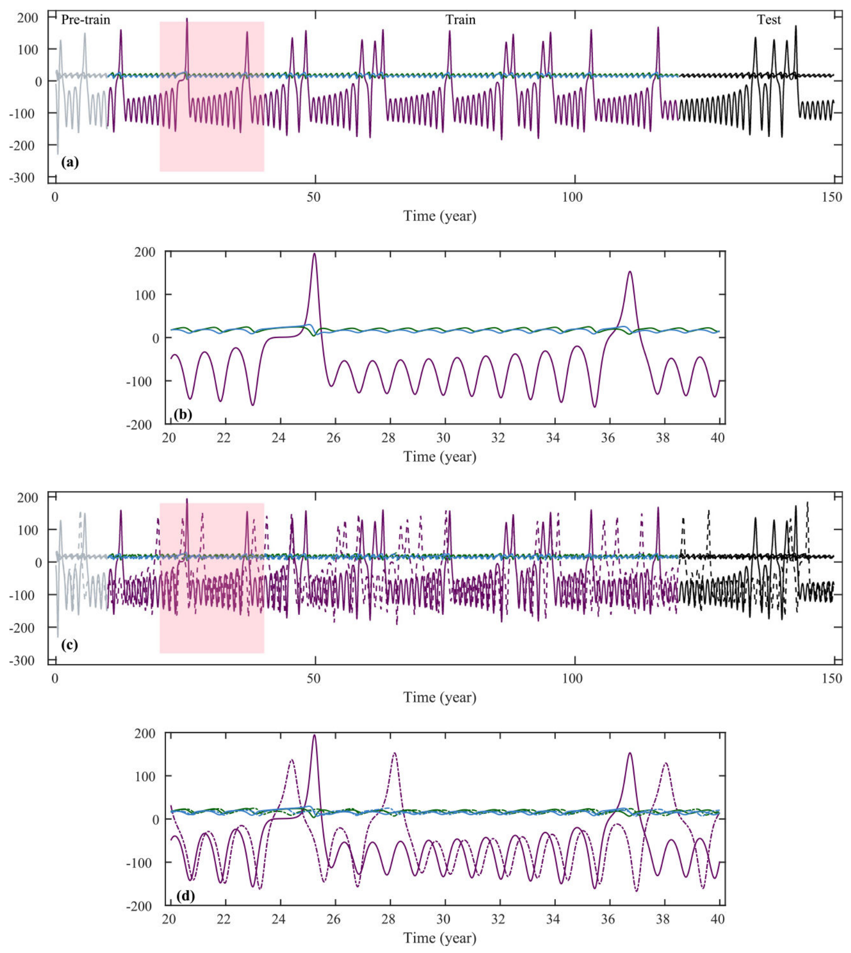


Fig. 6.

The ENSO time series. (a) The unused pre-training data is shown in gray, and the training data are in purple, green, and blue colors indicating the u, T_w, and T_e variables, respectively. The testing data is in black. (b) Zoomed-in section corresponding to the shaded region in panel (a). (c) The ENSO time series (solid) and the imperfect knowledge-based model (dashed). (d) Zoomed-in section corresponding to the shaded region in panel (c) demonstrating the difference between the generated time series and the imperfect knowledge-based model.

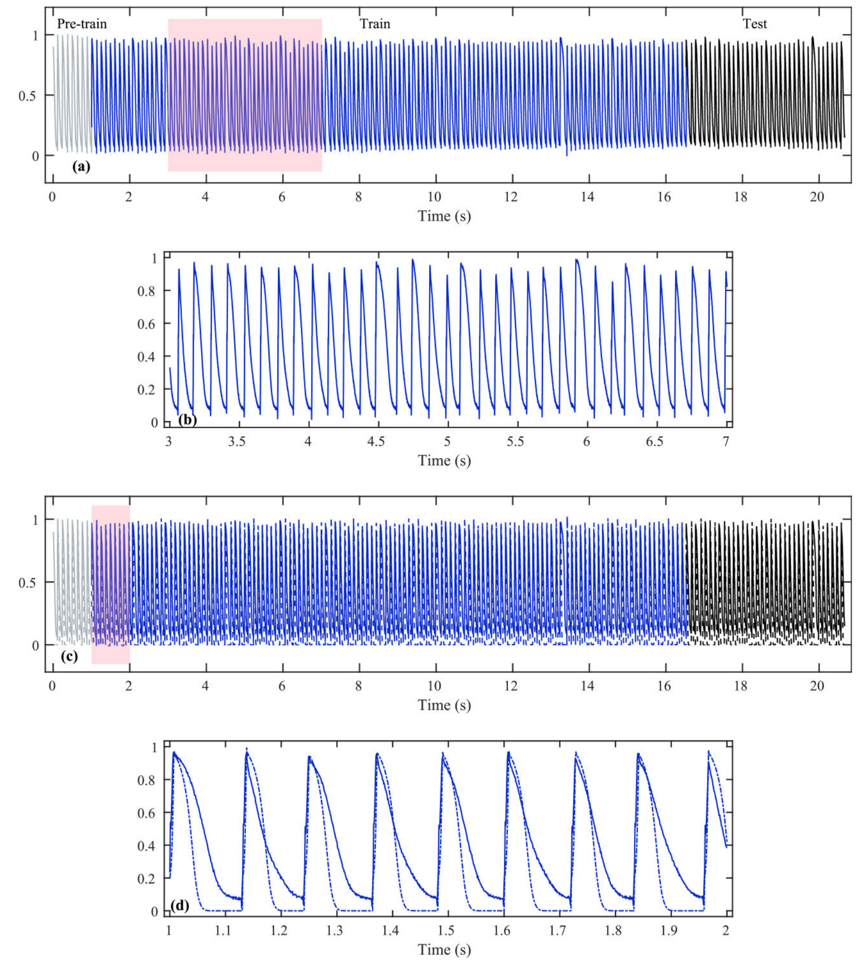


Fig. 7.

Experimental cardiac voltage time series featuring irregular action potentials. (a) Voltage time series including unused pre-training data (gray), training data (blue), and testing data (black). (b) Zoomed-in section corresponding to the shaded region in panel (a). (c) Experimental cardiac action potential time series (solid) and the imperfect knowledge-based model (dashed)line. (d) Zoomed-in section corresponding to the shaded region in panel (c), where the difference between the generated time series and the imperfect knowledge-based model can be observed.

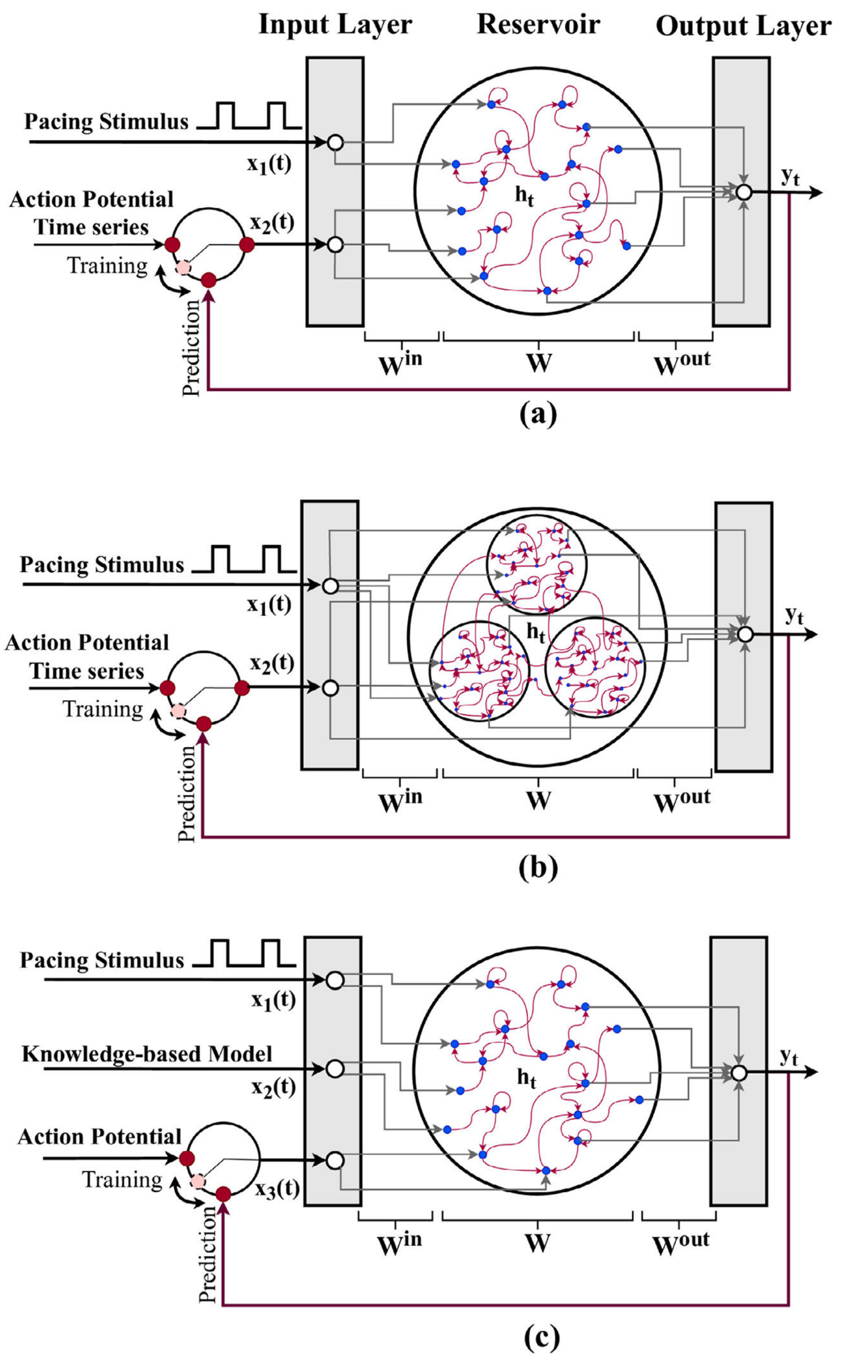


Fig. 8.

Components of reservoir computing approaches for modeling cardiac action potential time series, including (a) the baseline ESN, (b) CESN, and (c) HESN.

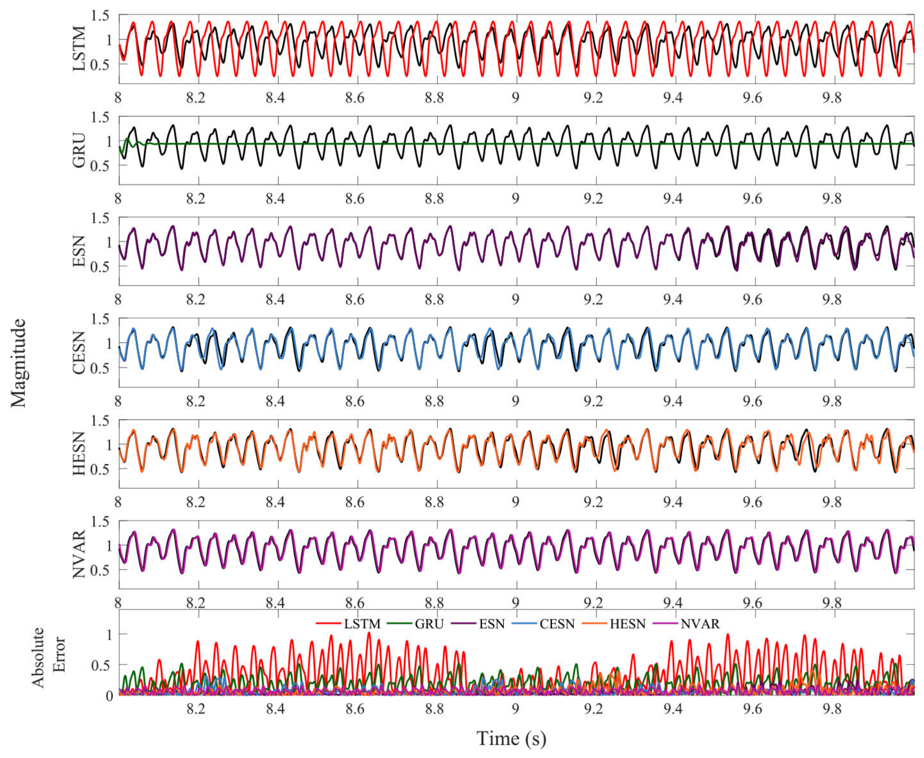


Fig. 9.

Mackey–Glass dataset forecasting results obtained by the six methods using a fixed network size of 100 neurons for the gated RNNs and ESN models and a computationally equivalent delay size for NVAR approach. The reference test data are shown in black and the predictions in color. Absolute errors of the predictions are presented in the bottom subplot, with color corresponding to each prediction method.

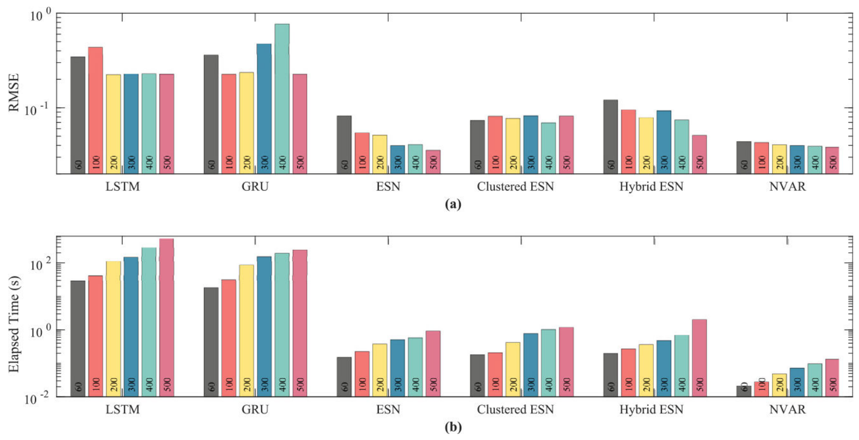


Fig. 10.

Comparison of RMSE (top) and computational time (bottom) for each method and network size tested for the Mackey–Glass dataset.

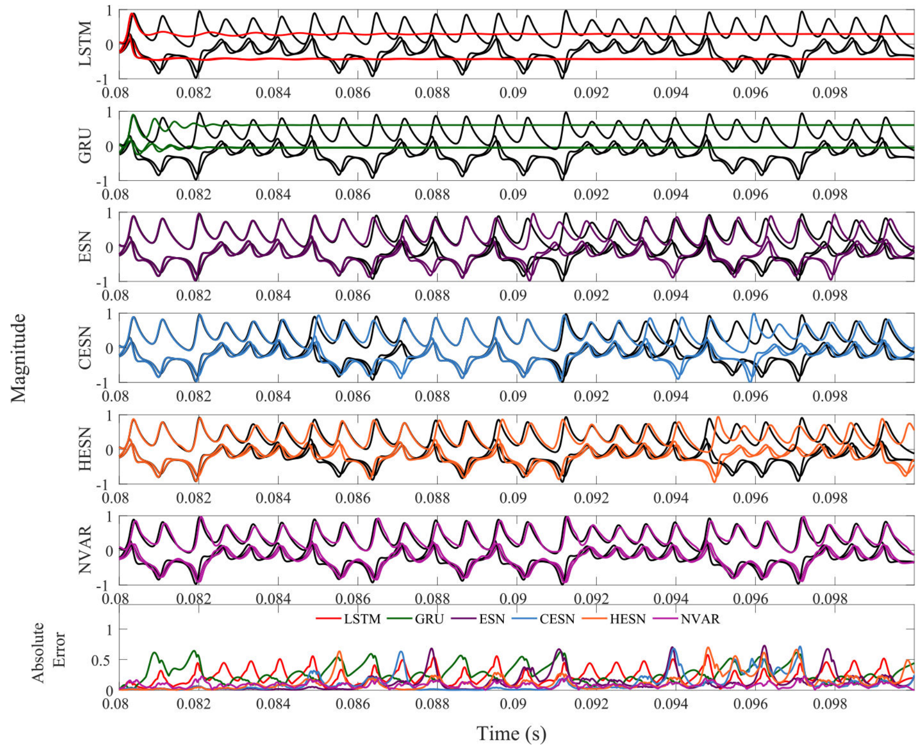


Fig. 11.

Lorenz system dataset forecasting results obtained by the six methods using a fixed network size of 100 neurons for the gated RNNs and ESN models and an equivalent delay size for the NVAR approach. The reference test data are shown in black and the predictions in color. Absolute errors of the predictions are presented in the bottom subplot, with color corresponding to each prediction method. Note that the reported error is the mean absolute error over the three input time series.

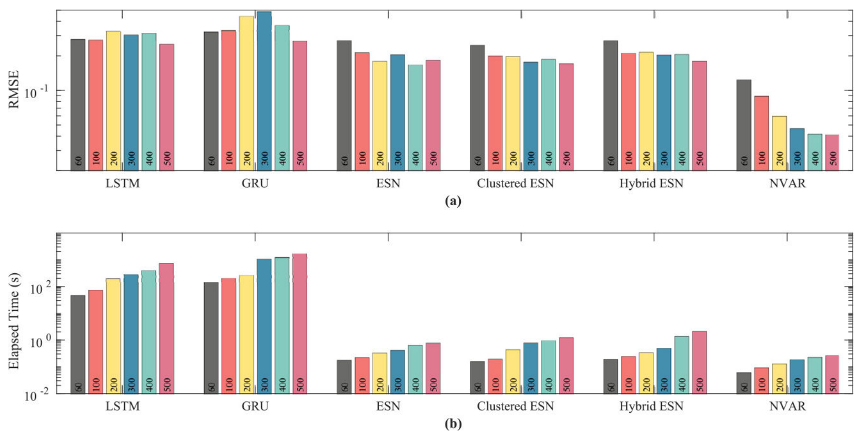


Fig. 12.

Comparison of RMSE (top) and computational time (bottom) for each method and network size tested for the Lorenz system dataset.

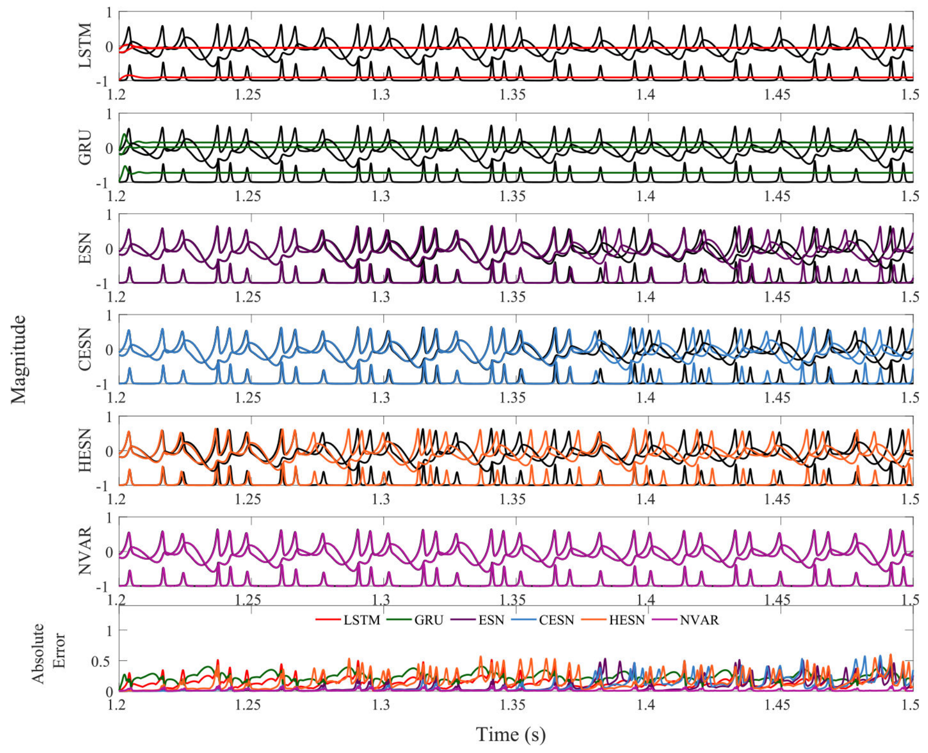


Fig. 13.

Morris–Lecar dataset forecasting results obtained by the six methods using a fixed network size of 100 neurons for the gated RNNs and ESN models and an equivalent delay size for the NVAR approach. The reference test data are shown in black and the predictions in color. Absolute errors of the predictions are presented in the bottom subplot, with color corresponding to each prediction method. Note that the reported error is the mean absolute error over the three input time series.

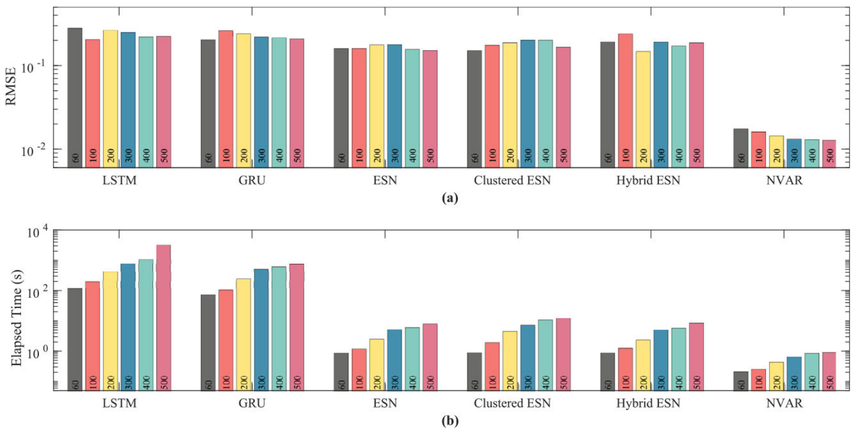


Fig. 14.

Comparison of RMSE (top) and computational time (bottom) for each method and network size tested for the Morris–Lecar dataset.

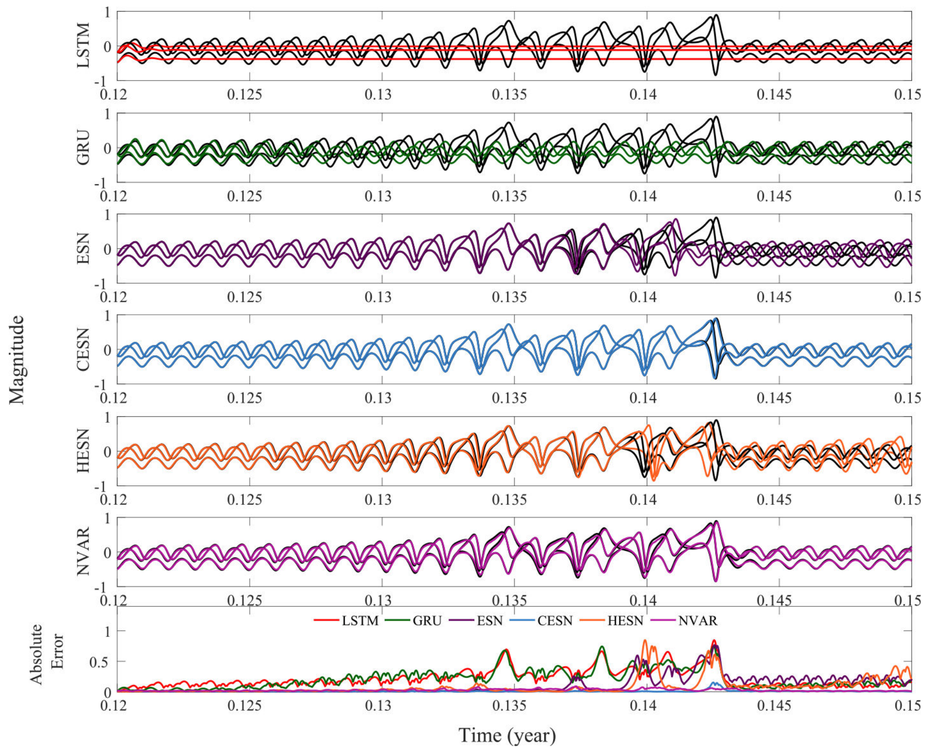


Fig. 15.
ENSO dataset forecasting results obtained by the six methods using a fixed network size of 100 neurons for the gated RNNs and ESN models and an equivalent delay size for the NVAR approach. The reference test data are shown in black and the predictions in color. Absolute errors of the predictions are presented in the bottom subplot, with color corresponding to each prediction method. Note that the reported error is the mean absolute error over the three input time series.

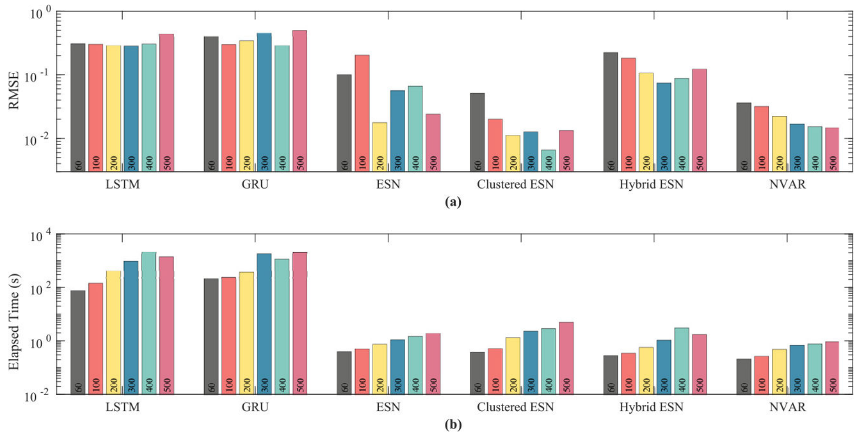


Fig. 16.

Comparison of RMSE (top) and computational time (bottom) for each method and network size tested for the ENSO dataset.

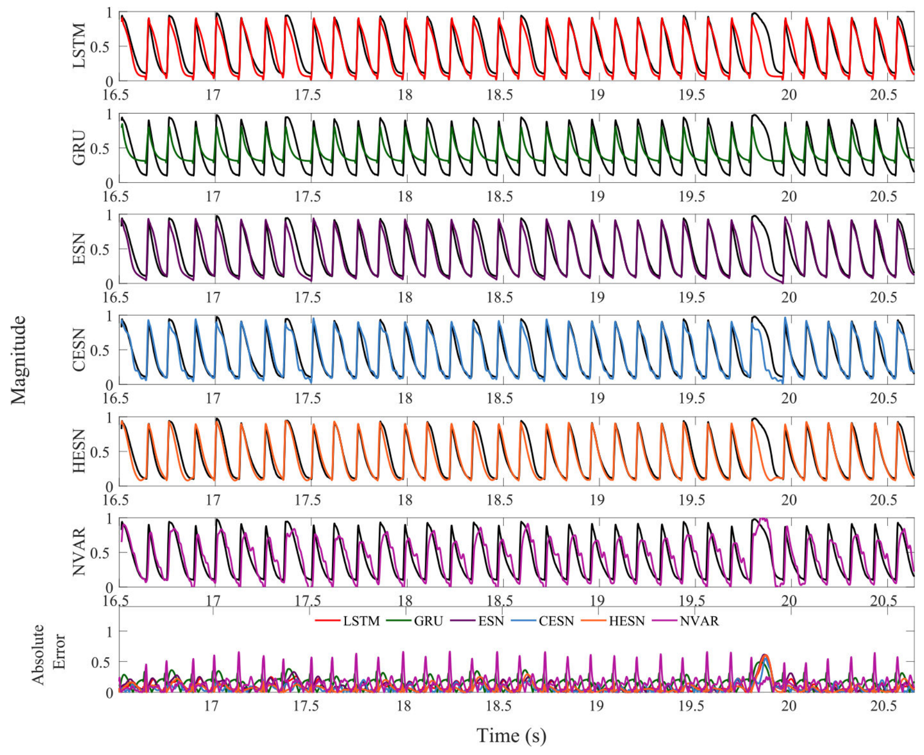


Fig. 17.
Experimental dataset forecasting results obtained by the six methods using a fixed network size of 100 neurons for the gated RNNs and ESN models and an equivalent delay size for NVAR approach. The reference test data are shown in black and the predictions in color. Absolute errors of the predictions are presented in the bottom subplot, with color corresponding to each prediction method.

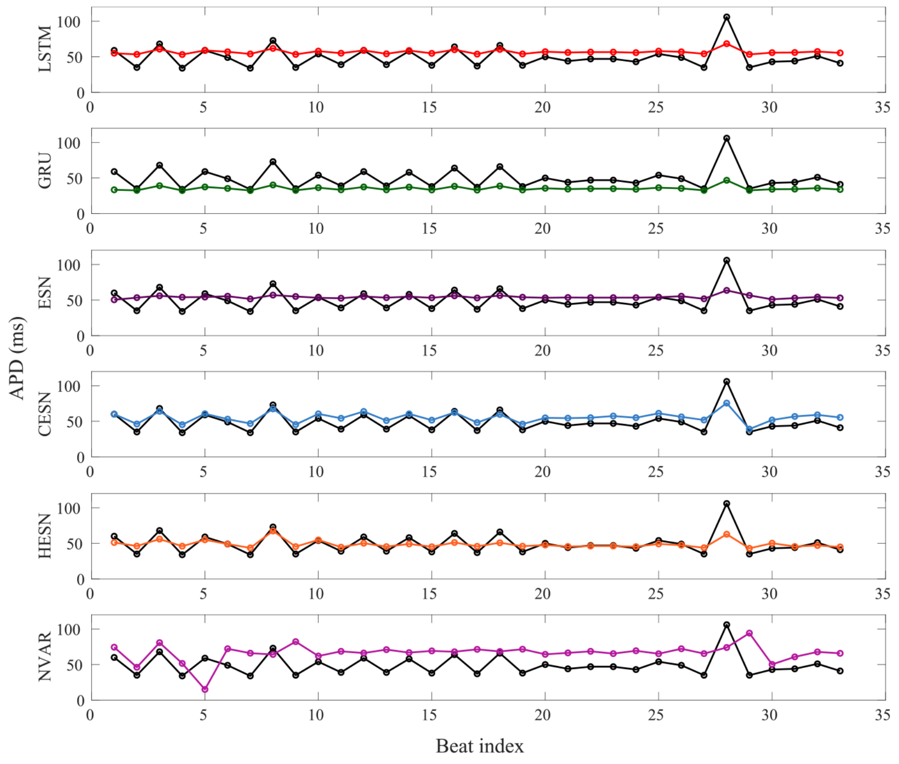


Fig. 18.

Experimental dataset APD forecasting results obtained by the six methods using a fixed network size of 100 neurons for the gated RNNs and ESN models and an equivalent delay size for NVAR approach. The reference APD values are shown in black and the predicted values in color.

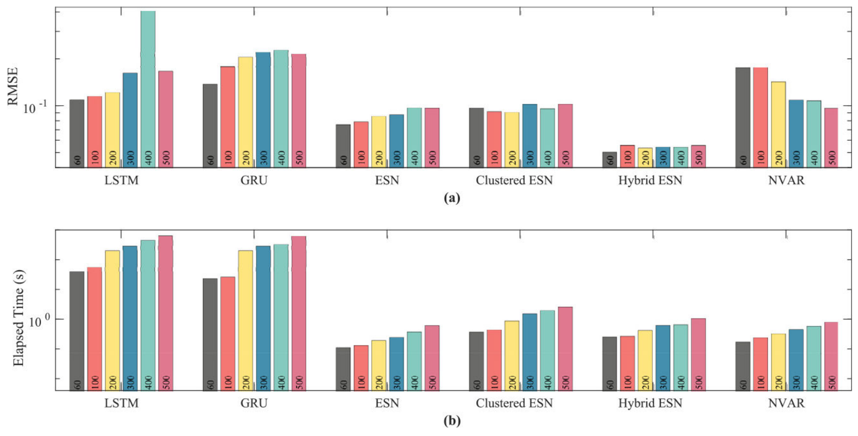


Fig. 19.

Comparison of RMSE (top) and computational time (bottom) for each method and network size tested for the experimental dataset.

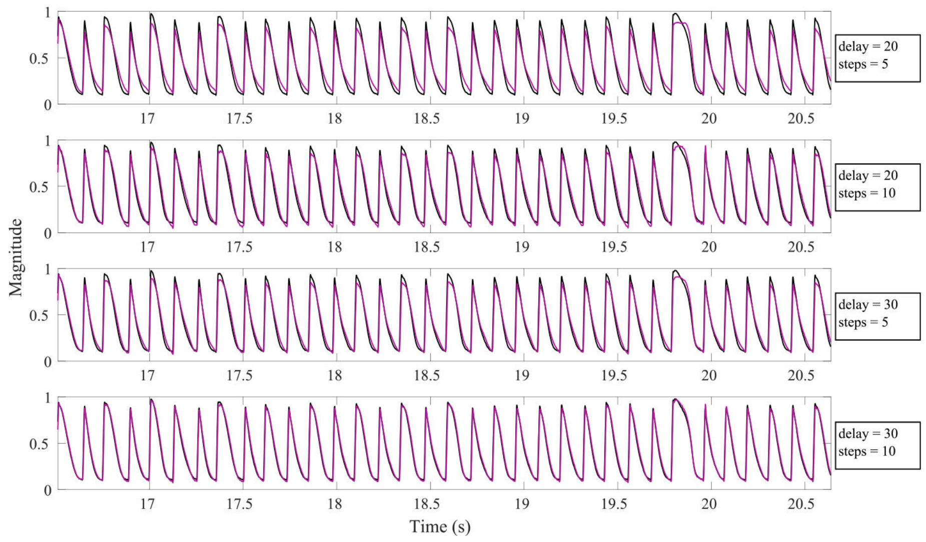


Fig. 20.

Experimental dataset action potential forecasting results obtained by the NVAR method using larger values for delays and skipped steps showing the reference test data (black) and the prediction results (red).

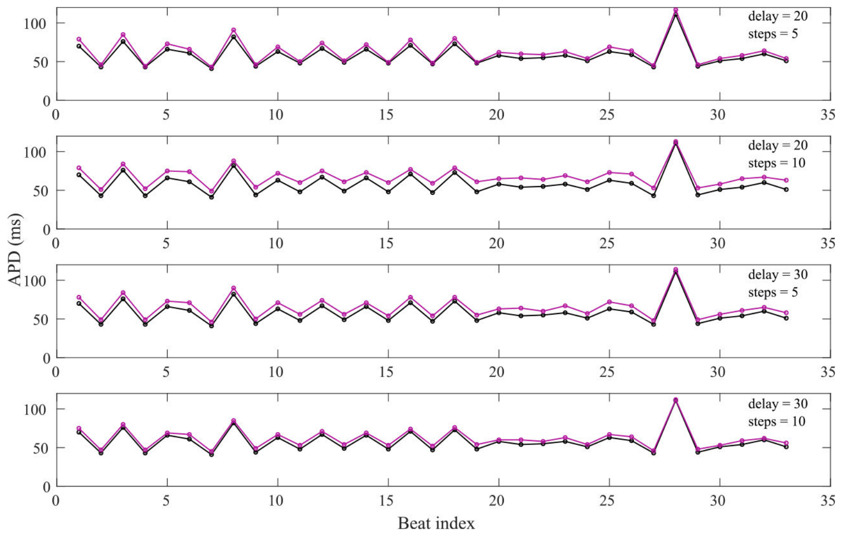


Fig. 21.

Experimental dataset APD forecasting results obtained by the NVAR method using larger values for delays and skipped steps. The reference APD values are shown in black and the predicted values in red.

Table 1

Hyperparameter values used for the grid search optimization for each prediction method. In the case of gated RNNs, the initial learning rate, maximum number of epochs, and regularization factor are the effective hyperparameters for training the networks by the Adam optimizer, while the number of layers determines the architecture of each network for a given number of hidden units. The dropout probability controls the dropping rate of dropout layers used as a regularization technique preventing overfitting. In RC techniques, i.e., ESN, CESN, and HESN, the input weight scale (σ_{in}^{j}) represents the scalar value multiplied by the jth input time series ($1 \leq j \leq d$) to adjust the magnitude of the input signal. Therefore, for univariate time series, e.g., the MG dataset, d is equal to 1 and the input weight scale is only one scalar, while for multivariate time series, e.g., the Lorenz dataset, σ_{in} consists of three scalar values that need to be tuned. Similarly, the HESN approach requires an additional set of weight scales for the knowledge-based model input(s), denoted by σ_{kb}^{j}. The selected spectral radius ρ is used to scale the reservoir weight matrix in ESNs. The amount of excitation discarded by the leaky integrator neurons is specified by the leaking rate α. The sparsity of the reservoir graph is controlled by the connection probability pr, defined as the probability of connection between any two hidden units in the reservoir. Furthermore, the CESN approach requires an additional hyperparameter pr_c, which is the inter-cluster probability indicating the sparsity of the connections between each pair of sub-reservoirs. The regularization parameter λ determines the ridge regression regularization factor used to obtain the readout weights. In the NVAR approach, the hyperparameter skip (s) controls the number of steps skipped between each two entries of delay embedding vectors.

Methods	Parameters	Values
	Number of layers	{1, 2, 4}
	Initial learning rate	{0.001, 0.002, 0.005, 0.010, 0.050, 0.100}
LSTM	Dropout probability	{0.00, 0.05, 0.10, 0.20, 0.50}
	Max number of epochs	{5, 10, 15, 20, 30, 50, 100}
	L_2 regularization	$\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}0\}$
	Number of layers	{1, 2, 4}
	Initial learning rate	{0.001, 0.002, 0.005, 0.010, 0.050, 0.100}
GRU	Dropout probability	{0.00, 0.05, 0.10, 0.20, 0.50}
	Max number of epochs	{5, 10, 15, 20, 30, 50, 100}
	L_2 regularization	$\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}0\}$
	Input weight scale (σ_{in}^{j}, $1 \leq j \leq d$)	{0.02, 0.05, 0.10, 0.20, 0.50, 0.80}
	Spectral radius (ρ)	{0.80, 0.85, 0.90, 0.99, 1.05, 1.15, 1.25, 1.55}
ESN	Leaking rate (α)	{0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00}
	Regularization (λ)	$\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$

Methods	Parameters	Values
	Connection probability (*pr*)	{0.01, 0.02, 0.05, 0.10, 0.15, 0.20}
	Number of clusters (n_c)	{2, 3, 4, 5}
	Input weight scale (σ_{in}^{j}, $1 \le j \le d$)	{0.02, 0.05, 0.10, 0.20, 0.50, 0.80}
	Spectral radius (ρ)	{0.80, 0.85, 0.90, 0.99, 1.05, 1.15, 1.25, 1.55}
CESN	Leaking rate (α)	{0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00}
	Regularization (λ)	{10^{-7}, 10^{6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}}
	Intra-cluster connection probability (*pr*)	{0.60, 0.7, 0.80, 0.85, 0.90, 0.95, 0.98}
	Inter-cluster connection probability (pr_c)	{0.01, 0.02, 0.05, 0.10, 0.15, 0.20}
	Input weight scale (σ_{in}^{j}, $1 \le j \le d$)	{0.02, 0.05, 0.10, 0.20, 0.50, 0.80}
	Knowledge based input weight scale (σ_{kb}^{j}, $1 \le j \le d$)	{0.02, 0.05, 0.10, 0.20, 0.50, 0.80}
HESN	Spectral radius (ρ)	{0.80, 0.85, 0.90, 0.99, 1.05, 1.15, 1.25, 1.55}
	Leaking rate (α)	{0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00}
	Regularization (λ)	{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}}
	Connection probability (*pr*)	{0.01, 0.02, 0.05, 0.10, 0.15, 0.20}
NVAR	Skip (*s*)	{2, 3, 4, 5, 6, 7, 8, 10, 15, 20, 25, 30}
	Regularization (λ)	{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}}