Mathematics Interdisciplinary Research **8** (2) (2023) 71 − 92

Original Scientific Paper

Chaotic Time Series Prediction Using Rough-Neural Networks

Ghasem Ahmadi⋆ *and Mohammad Dehghandar*

Abstract

Artificial neural networks with amazing properties, such as universal approximation, have been utilized to approximate the nonlinear processes in many fields of applied sciences. This work proposes the rough-neural networks (R-NNs) for the one-step ahead prediction of chaotic time series. We adjust the parameters of R-NNs using a continuous-time Lyapunov-based training algorithm, and prove its stability using the continuous form of Lyapunov stability theory. Then, we utilize the R-NNs to predict the well-known Mackey-Glass time series, and Henon map, and compare the simulation results with some well-known neural models.

Keywords: Artificial neural network, Rough-neural network, Time series prediction, Lyapunov-based learning algorithm, Lyapunov stability theory.

2020 Mathematics Subject Classification: 68T05, 62M10, 93D05.

How to cite this article
G. Ahmadi and M. Dehghandar, Chaotic time series prediction using rough-neural networks, *Math. Interdisc. Res.* **8** (2) (2023) 71-92.

1. Introduction

Time series prediction is one of the most important problems in many fields, such as finance, economics, medical sciences, social sciences, and engineering [1]. The nature of real processes and the effects of uncertainties make time series prediction challenging in the literature. The researchers try to solve this problem using the

⋆Corresponding author (E-mail: g.ahmadi@pnu.ac.ir)
Academic Editor: Seyed Morteza Babamir
Received 4 June 2021, Accepted 14 April 2023
DOI: 10.22052/MIR.2023.242878.1290

new excellent approaches recently presented in computer science, such as artificial neural networks (ANNs).

ANNs are motivated from the brain and nervous system of human beings [2]. Neurons are the basic units in the brain and nervous system, and the information is processed by them. In ANNs, a neuron is a mathematical model of biological neuron. It consists of the summation of weighted inputs, biases, and nonlinear activation functions. The neurons are connected, and the parameters are adjusted using some learning algorithms such that they can approximate the nonlinear functions. Recently, neural networks have been utilized to solve many problems in the applied sciences and engineering.

The effectiveness of ANNs in time series prediction has been shown in many works. In 2001, Frank et al. utilized the neural networks for time series prediction [3]. In 2006, Chen et al. used the local linear wavelet neural networks to forecast the time series [4]. Gholipour et al. applied the neurofuzzy models for the chaotic time series forecasting [5]. Faruk utilized the neural networks to predict the water quality time series [6]. Deep learning models have recently been used for multi-step ahead time series forecasting [7]. Tealab considered the most recent works that are relevant to the usage of ANNs for nonlinear time series forecasting [8].

Due to the effects of uncertainties in the natural time series, their predictions with the conventional ANNs is difficult. To deal with the uncertainties in prediction of time series, this work proposes the rough-neural networks (R-NNs) for the prediction of nonlinear chaotic time series. For the first time, Lingras propose the R-NNs on the basis of rough set theory (RST) [9]. There are some rough neurons (RNs) in R-NNs that help them to handle the uncertainties.

Recently, R-NNs have been applied to solve some important problems. In 1996, Lingras has been used the R-NNs for traffic volume prediction [9]. Yamaguchi et al. have been used them for medical diagnostic support system [10]. Nowicki has been used them for data classification [11]. Sasirekha and Thangavel have been used them for biometric face classification [12].

Recently, R-NNs have been used for the nonlinear system identification. Alehasher and Teshnehlab implemented an special structure and learning algorithm of R-NNs for system identification [13]. Ahmadi and Teshnehlab designed a sinusoidal rough-neural identifier for discrete-time nonlinear systems with an interval-based structure and a Lyapunov-based learning algorithm [14]. In [14], the discrete form of Lyapunov stability theory is used for stability analysis with the long computations. In 2018, Ahmadi et al. employed the paradigm of emotional learning to accelerate the performance of R-NNs in the system identification [15]. In 2020, R-NNs with a stochastic gradient-based learning algorithm used for the identification of multi input-multi output cement rotary kiln [16]. On the basis of R-NNs, the rough extreme learning machines (RELMs) has been proposed for the identification of continuous-time nonlinear systems [17]. RELMs are some types of R-NNs that their parameters in first layer are selected randomly and never updated. Therefore, only the parameters between the hidden layer and outputs are updated. In [17], the adjustable parameters of RELMs have been trained using a

Lyapunov-based learning algorithm, and a continuous form of Lyapunov stability theory is used for its stability analysis.

In the literature, the researchers have been utilized the R-NNs for forecasting some time series. Lingras proposed and applied the R-NNs to predict the traffic volume time series [9, 18]. Khodayar et al. have been proposed the deep R-NNs for short-term wind speed forecasting [19]. On the basis of R-NNs, they have been developed some rough extensions of auto-encoders and denoising auto-encoders to cope with the uncertainties in wind speed prediction. Jahangir et al. utilized the R-NNs for electricity price forecasting [20]. In other study, on the basis of R-NNs, some feedforward and recurrent neural architectures have been developed to forecast travel behavior in plug-in electric vehicles [21]. Cao et al. used the fuzzy R-NNs with a multiobjective optimization algorithm for stock prediction [22]. Sheikhoushaghi et al. applied the R-NNs to forecast the oil production rate of an oil field [23]. They used the capabilities of R-NNs in handling the noisy and nonlinear real data, and compared them with some well-known architectures such as long-short-term memory.

In this work, extending the presented approach in [17], a Lyapunov-based learning algorithm is proposed to adjust the parameters of R-NNs, and a continuous form of Lyapunov stability theory is used for its stability analysis. The proof of the relevant theorem in this work, is very shorter and stronger than the presented proof in [14]. Then, R-NNs are utilized for time series prediction. Two well-known benchmarks Mackey-Glass time series (M-GTS) and Henon map are predicted using the R-NNs. The simulation results are compared with some well-known neural predictors.

In contrast to our recent work in this context [24], we extend the literature review, consider the relevant published papers, and compare the present work with them. We give more explanations about the R-NNs, and state the main result as Theorem 4.1, and prove it. We state the M-GTS prediction with more details, predict the Henon map, and compare the results with some well-known neural networks.

This paper is continued as follows. A short introduction about the time series prediction is given in Section 2. Then, R-NNs are described in Section 3. Time series prediction using R-NNs is presented in Section 4, and a Lyapunov-based training algorithm is suggested for R-NNs. The M-GTS and Henon map is predicted using R-NNs in Section 5. The conclusion is drawn in Section 6.

2. Time series prediction

A time series is a set of data $\{x(t)\}_{t=0}^{\infty}$ where t denotes the time index. We can suppose that $x(t)$ is a continuous function with the variable t [3]. In the natural processes, to achieve a discrete dataset, the sampled data are utilized. In the prediction of time series using the neural networks, the future values are forecasted

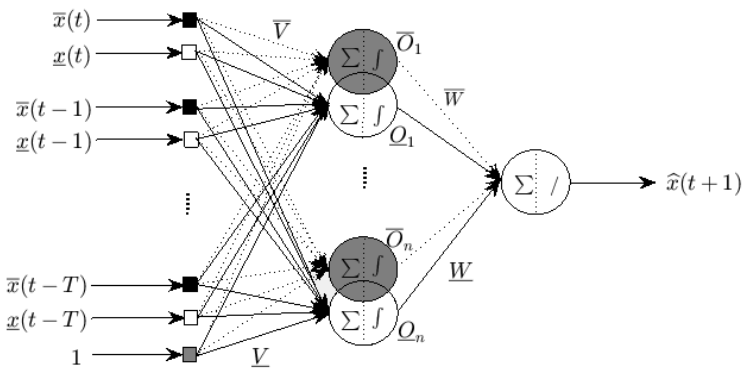

Figure 1: Structure of R-NN in time series prediction.

using the past values. In fact, we try to find the function g such that

$$x(t+d) = g(x(t), x(t-1), \cdots, x(t-T)), \tag{1}$$

where T is the number of time steps. For $d = 1$, the one-step ahead prediction is done, and for $d > 1$, the multi-step ahead prediction is done.

3. Rough-neural networks

R-NNs are introduced on the basis of RST. RST is a different approach for managing the uncertain and imperfect knowledge that has been proposed by Pawlak in 1982 [25]. In this theory, the uncertain sets are defined using the upper and lower approximations. Therefore, a clear relation exists between this theory and interval analysis. Due to the wonderful properties of ANNs, the researchers try to use them for modeling of nonlinear systems in the presence of uncertain and imperfect data. Therefore, the idea of combining RST with ANNs has been appeared and then, the R-NNs are proposed by Lingras in 1996 [9]. A RN contains two usual neurons that are named the upper bound neurons (UBNs) and the lower bound neurons (LBNs). The output of UBN is the maximum of their outputs, and the output of LBN is the minimum of their outputs.

In other words, R-NNs have been proposed to give the neural networks the ability of working with interval knowledge. The flexibility of computations in R-NNs are more than interval neural networks and their computations are simpler than interval computations.

This section explains the structure of R-NNs in time series prediction. Consider the R-NN with hidden RNs (HRNs) and the usual output neurons, as shown in Figure 1. We denote the outputs of R-NN with $\widehat{x}(t+1)$ and the inputs of R-NN with

$$\mathbf{x}(t) = [\overline{x}(t), \underline{x}(t), \overline{x}(t-1), \underline{x}(t-1), \cdots, \overline{x}(t-T), \underline{x}(t-T), 1]^T,$$

where $\underline{x}$ is the lower bound, and $\overline{x}$ is the upper bound of x, and T is the number of time steps. In the vector $\mathbf{x}$, the input of biases is shown with the component 1. Let $\underline{V}$ and $\overline{V}$ be the weights of connections between the inputs and hidden LBNs and the weights of connections between the inputs and hidden UBNs, respectively. Suppose that $\underline{W}$ and $\overline{W}$ be the weights of connections between the hidden LBNs and output neurons and the weights of connections between the hidden UBNs and output neurons, respectively.

Further, let $\underline{O}$, and $\overline{O}$ be the outputs of LBNs and the outputs of UBNs in the hidden layer, respectively. Besides, ϕ shows the activation function of hidden neurons. Then, according to the definition of RNs we have [9, 14]:

$$\underline{O} = \min\left(\underline{\phi}, \overline{\phi}\right), \quad \overline{O} = \max\left(\underline{\phi}, \overline{\phi}\right),$$

where $\underline{\phi} = \phi(\underline{V}\mathbf{x}(t))$, $\overline{\phi} = \phi(\overline{V}\mathbf{x}(t))$. The output $\widehat{x}(t+1)$ of R-NN is given by

$$\widehat{x}(t+1) = \underline{WO} + \overline{WO} = \underline{W}\min\left(\underline{\phi}, \overline{\phi}\right) + \overline{W}\max\left(\underline{\phi}, \overline{\phi}\right). \tag{2}$$

To simplify the relations in the next section, we try to substitute the min and max operations with some algebraic equations. To this end, we define the n-vectors $\underline{\delta} = (\underline{\delta}^1, \underline{\delta}^2, \cdots, \underline{\delta}^n)$ and $\overline{\delta} = (\overline{\delta}^1, \overline{\delta}^2, \cdots, \overline{\delta}^n)$ such that

$$\underline{\delta}^j, \overline{\delta}^j = 0 \ or \ 1, \quad \underline{\delta}^j + \overline{\delta}^j = 1, \quad j = 1, 2, \cdots, n. \tag{3}$$

$$\underline{\delta}^j \underline{\phi}^j + \overline{\delta}^j \overline{\phi}^j \leq \underline{\phi}^j, \overline{\phi}^j \leq \overline{\delta}^j \underline{\phi}^j + \underline{\delta}^j \overline{\phi}^j, \tag{4}$$

where $\underline{\phi}^j$ and $\overline{\phi}^j$ denote the jth components of $\underline{\phi}$ and $\overline{\phi}$, respectively. Then, we have

$$\min\left(\underline{\phi}^j, \overline{\phi}^j\right) = \underline{\delta}^j \underline{\phi}^j + \overline{\delta}^j \overline{\phi}^j, \tag{5}$$

$$\max\left(\underline{\phi}^j, \overline{\phi}^j\right) = \overline{\delta}^j \underline{\phi}^j + \underline{\delta}^j \overline{\phi}^j. \tag{6}$$

As a result,

$$\min\left(\underline{\phi}, \overline{\phi}\right) = \mathrm{diag}(\underline{\delta})\underline{\phi} + \mathrm{diag}(\overline{\delta})\overline{\phi}, \tag{7}$$

$$\max\left(\underline{\phi}, \overline{\phi}\right) = \mathrm{diag}(\overline{\delta})\underline{\phi} + \mathrm{diag}(\underline{\delta})\overline{\phi}. \tag{8}$$

Then, we introduce $\mathcal{C} = \underline{W}\mathrm{diag}(\underline{\delta}) + \overline{W}\mathrm{diag}(\overline{\delta})$, and $\mathcal{D} = \underline{W}\mathrm{diag}(\overline{\delta}) + \overline{W}\mathrm{diag}(\underline{\delta})$. Therefore, using (2), (7) and (8), we have

$$\begin{aligned}
\widehat{x}(t+1) &= \underline{W}\min\left(\underline{\phi}, \overline{\phi}\right) + \overline{W}\max\left(\underline{\phi}, \overline{\phi}\right) \\
&= \underline{W}\left(\mathrm{diag}(\underline{\delta})\underline{\phi} + \mathrm{diag}(\overline{\delta})\overline{\phi}\right) + \overline{W}\left(\mathrm{diag}(\overline{\delta})\underline{\phi} + \mathrm{diag}(\underline{\delta})\overline{\phi}\right) \\
&= \left(\underline{W}\mathrm{diag}(\underline{\delta}) + \overline{W}\mathrm{diag}(\overline{\delta})\right)\underline{\phi} + \left(\mathrm{diag}(\overline{\delta}) + \mathrm{diag}(\underline{\delta})\right)\overline{\phi} \\
&= \mathcal{C}\underline{\phi} + \mathcal{D}\overline{\phi}.
\end{aligned} \tag{9}$$

4. Time series prediction using R-NNs

The time evolution of time series can be defined in some state space model in dynamical systems [26, 27]. The time series observations $\{x(t)\}_{t=0}^{\infty}$ can be transformed to the state vectors $\{\mathbf{z}(t)\}_{t=0}^{\infty}$ where $\mathbf{z}(t) \in \mathbf{R}^n$ shows the system state. The dynamics of these states can be defined as follows:

$$\dot{\mathbf{z}}(t) = f(\mathbf{z}(t)), \tag{10}$$

where f is an unknown function. Suppose that A is a Hurwitz matrix and

$$f(\mathbf{z}(t)) = A\mathbf{z}(t) + g(\mathbf{z}(t)), \tag{11}$$

where g is the nonlinear part of f. To simplify the relations, in continue, we remove the argument t. Suppose that R-NN can model the nonlinear function g using the ideal weights $\mathcal{C}_\star, \mathcal{D}_\star, \underline{V}_\star$, and $\overline{V}_\star$. Then, according to the Equation (9), we have

$$\dot{\mathbf{z}} = A\mathbf{z} + \mathcal{C}_\star\phi(\underline{V}_\star\mathbf{x}) + \mathcal{D}_\star\phi(\overline{V}_\star\mathbf{x}). \tag{12}$$

We construct the parametric model of (14) as follow:

$$\begin{aligned} \dot{\widehat{\mathbf{z}}} &= A\widehat{\mathbf{z}} + \widehat{\mathcal{C}}\phi(\widehat{\underline{V}}\mathbf{x}) + \widehat{\mathcal{D}}\phi(\widehat{\overline{V}}\mathbf{x}) \\ &= A\widehat{\mathbf{z}} + \widehat{\mathcal{C}}\underline{\phi} + \widehat{\mathcal{D}}\overline{\phi}, \end{aligned} \tag{13}$$

where $\underline{\phi} = \phi(\widehat{\underline{V}}\mathbf{x})$, $\overline{\phi} = \phi(\widehat{\overline{V}}\mathbf{x})$, and $\widehat{\mathcal{C}}$, $\widehat{\mathcal{D}}$, $\widehat{\underline{V}}$, and $\widehat{\overline{V}}$ denote the estimates of $\mathcal{C}_\star$, $\mathcal{D}_\star$, $\underline{V}_\star$, and $\overline{V}_\star$ respectively. We use the Taylor's expansion for the nonlinear terms in (12), and achieve the following relation:

$$\dot{\mathbf{z}} = A\mathbf{z} + \widehat{\mathcal{C}}\underline{\phi} + \widetilde{\mathcal{C}}\underline{\phi} + \widehat{\mathcal{C}}\underline{\phi}'\widetilde{\underline{V}}\mathbf{x} + \underline{R}_2 + \widehat{\mathcal{D}}\overline{\phi} + \widetilde{\mathcal{D}}\overline{\phi} + \widehat{\mathcal{D}}\overline{\phi}'\widetilde{\overline{V}}\mathbf{x} + \overline{R}_2, \tag{14}$$

where $\underline{R}_2$ and $\overline{R}_2$ are the Taylor's series reminders, $\underline{\phi}'$ and $\overline{\phi}'$ are the derivatives of $\underline{\phi}$ and $\overline{\phi}$, and

$$\widetilde{\mathcal{C}} = \mathcal{C}_\star - \widehat{\mathcal{C}}, \quad \widetilde{\mathcal{D}} = \mathcal{D}_\star - \widehat{\mathcal{D}}, \quad \widetilde{\underline{V}} = \underline{V}_\star - \widehat{\underline{V}}, \quad \widetilde{\overline{V}} = \overline{V}_\star - \widehat{\overline{V}}. \tag{15}$$

More details about Equation (14) can be found in [14]. Now, we can compute the state error as follows:

$$\begin{aligned} \dot{\mathbf{e}} &= \dot{\mathbf{z}} - \dot{\widehat{\mathbf{z}}} \\ &= A\mathbf{z} + \widehat{\mathcal{C}}\underline{\phi} + \widetilde{\mathcal{C}}\underline{\phi} + \widehat{\mathcal{C}}\underline{\phi}'\widetilde{\underline{V}}\mathbf{x} + \widehat{\mathcal{D}}\overline{\phi} + \widetilde{\mathcal{D}}\overline{\phi} + \widehat{\mathcal{D}}\overline{\phi}'\widetilde{\overline{V}}\mathbf{x} + \zeta \\ &- A\widehat{\mathbf{z}} - \widehat{\mathcal{C}}\underline{\phi} - \widehat{\mathcal{D}}\overline{\phi} \\ &= A\mathbf{e} + \widetilde{\mathcal{C}}\underline{\phi} + \widehat{\mathcal{C}}\underline{\phi}'\widetilde{\underline{V}}\mathbf{x} + \widetilde{\mathcal{D}}\overline{\phi} + \widehat{\mathcal{D}}\overline{\phi}'\widetilde{\overline{V}}\mathbf{x} + \zeta, \end{aligned} \tag{16}$$

where

$$\mathbf{e} = \mathbf{z} - \widehat{\mathbf{z}}, \quad \widetilde{\underline{V}} = \underline{V}_\star - \widehat{\underline{V}}, \quad \widetilde{\overline{V}} = \overline{V}_\star - \widehat{\overline{V}}, \tag{17}$$

$$\widetilde{\mathcal{D}} = \mathcal{D}_\star - \widehat{\mathcal{D}}, \quad \widetilde{\mathcal{C}} = \mathcal{C}_\star - \widehat{\mathcal{C}}, \quad \zeta = \underline{R}_2 + \overline{R}_2. \tag{18}$$

In [14], a Lyapunov-based algorithm is presented for R-NNs in the identification of nonlinear systems where in the present work, a continuous-time Lyapunov-based learning algorithm is derived. Recently, a Lyapunov-based algorithm is presented for the RELMs in continuous-time systems identification [17]. In RELMs, the weights of connections between the inputs and hidden layer are not adjusted where in the R-NNs, all the weights are adjusted.

Theorem 4.1. *Suppose that R-NN can model the time series (10) where the following laws are used to train it:*

$$\dot{\widehat{\underline{W}}} = \boldsymbol{e}\left[\min(\underline{\phi}, \overline{\phi})\right]\Gamma_1^{-1}, \tag{19}$$

$$\dot{\widehat{\overline{W}}} = \boldsymbol{e}\left[\max(\underline{\phi}, \overline{\phi})\right]\Gamma_2^{-1}, \tag{20}$$

$$\dot{\widehat{\underline{V}}} = \Gamma_3^{-1}(\underline{\phi}^{'})^T\widehat{\mathcal{C}}^T\boldsymbol{e}x^T, \tag{21}$$

$$\dot{\widehat{\overline{V}}} = \Gamma_4^{-1}(\overline{\phi}^{'})^T\widehat{\mathcal{D}}^T\boldsymbol{e}x^T, \tag{22}$$

where the positive definite matrices Γ_1, Γ_2, Γ_3 and Γ_4 are the learning gains. Suppose that

$$\|\boldsymbol{e}\| \geq \frac{\|\zeta\|}{|\lambda_{min}(A)|}, \tag{23}$$

where $\lambda_{\min}(A)$ is smallest eigenvalue of A. Then, the prediction error $\boldsymbol{e}$ tends to zero.

Proof. Let

$$v = \frac{1}{2}\mathbf{e}^T\mathbf{e} + \frac{1}{2}\mathrm{tr}\left(\widetilde{\underline{W}}\Gamma_1\widetilde{\underline{W}}^T\right) + \frac{1}{2}\mathrm{tr}\left(\widetilde{\overline{W}}\Gamma_2\widetilde{\overline{W}}^T\right) + \frac{1}{2}\mathrm{tr}\left(\widetilde{\underline{V}}^T\Gamma_3\widetilde{\underline{V}}\right)$$
$$+\frac{1}{2}\mathrm{tr}\left(\widetilde{\overline{V}}^T\Gamma_4\widetilde{\overline{V}}\right).$$

Then, using Equation (16), we have

$$\begin{aligned}
\dot{v} &= \mathbf{e}^T\dot{\mathbf{e}} + \mathrm{tr}\left(\dot{\widetilde{\underline{W}}}\Gamma_1\widetilde{\underline{W}}^T\right) + \mathrm{tr}\left(\dot{\widetilde{\overline{W}}}\Gamma_2\widetilde{\overline{W}}^T\right) + \mathrm{tr}\left(\dot{\widetilde{\underline{V}}}^T\Gamma_3\widetilde{\underline{V}}\right) + \mathrm{tr}\left(\dot{\widetilde{\overline{V}}}^T\Gamma_4\widetilde{\overline{V}}\right) \\
&= \mathbf{e}^TA\mathbf{e} + \mathbf{e}^T\widetilde{\mathcal{C}}\underline{\phi} + \mathbf{e}^T\widehat{\mathcal{C}}\underline{\phi}^{'}\widetilde{\underline{V}}x + \mathbf{e}^T\widetilde{\mathcal{D}}\overline{\phi} + \mathbf{e}^T\widehat{\mathcal{D}}\overline{\phi}^{'}\widetilde{\overline{V}}x + \mathbf{e}^T\zeta + \mathrm{tr}\left(\dot{\widetilde{\underline{W}}}\Gamma_1\widetilde{\underline{W}}^T\right) \\
&+ \mathrm{tr}\left(\dot{\widetilde{\overline{W}}}\Gamma_2\widetilde{\overline{W}}^T\right) + \mathrm{tr}\left(\dot{\widetilde{\underline{V}}}^T\Gamma_3\widetilde{\underline{V}}\right) + \mathrm{tr}\left(\dot{\widetilde{\overline{V}}}^T\Gamma_4\widetilde{\overline{V}}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \mathbf{e}^T A\mathbf{e} + \mathbf{e}^T\zeta + \mathrm{tr}\left(\mathbf{e}\underline{\phi}^T\widetilde{\mathcal{C}}^T\right) + \mathrm{tr}\left(\mathbf{e}\overline{\phi}^T\widetilde{\mathcal{D}}^T\right) + \mathrm{tr}\left(x\mathbf{e}^T\widehat{\mathcal{C}}\underline{\phi}'\underline{\widetilde{V}} + \dot{\underline{\widetilde{V}}}^T\Gamma_3\underline{\widetilde{V}}\right)\\
&+ \mathrm{tr}\left(\mathbf{x}\mathbf{e}^T\widehat{\mathcal{D}}\overline{\phi}'\overline{\widetilde{V}} + \dot{\overline{\widetilde{V}}}^T\Gamma_4\overline{\widetilde{V}}\right) + \mathrm{tr}\left(\dot{\underline{\widetilde{W}}}\Gamma_1\underline{\widetilde{W}}^T\right) + \mathrm{tr}\left(\dot{\overline{\widetilde{W}}}\Gamma_2\overline{\widetilde{W}}^T\right)\\
&= \mathbf{e}^T A\mathbf{e} + \mathbf{e}^T\zeta + \mathrm{tr}\left(\mathbf{e}\underline{\phi}^T(\underline{\widetilde{W}}diag(\underline{\delta}) + +\overline{\widetilde{W}}diag(\overline{\delta}))^T\right)\\
&+ \mathrm{tr}\left(\mathbf{e}\overline{\phi}^T(\underline{\widetilde{W}}diag(\overline{\delta}) + \overline{\widetilde{W}}diag(\underline{\delta}))^T\right) + \mathrm{tr}\left(\mathbf{x}\mathbf{e}^T\widehat{\mathcal{C}}\underline{\phi}'\underline{\widetilde{V}} + \dot{\underline{\widetilde{V}}}^T\Gamma_3\underline{\widetilde{V}}\right)\\
&+ \mathrm{tr}\left(\mathbf{x}\mathbf{e}^T\widehat{\mathcal{D}}\overline{\phi}'\overline{\widetilde{V}} + \dot{\overline{\widetilde{V}}}^T\Gamma_4\overline{\widetilde{V}}\right) + \mathrm{tr}\left(\dot{\underline{\widetilde{W}}}\Gamma_1\underline{\widetilde{W}}^T\right) + \mathrm{tr}\left(\dot{\overline{\widetilde{W}}}\Gamma_2\overline{\widetilde{W}}^T\right)\\
&= \mathbf{e}^T A\mathbf{e} + \mathbf{e}^T\zeta + \mathrm{tr}\left(\mathbf{e}\left[\underline{\phi}^T diag(\underline{\delta}) + \overline{\phi}^T diag(\overline{\delta})\right]\underline{\widetilde{W}}^T + \dot{\underline{\widetilde{W}}}\Gamma_1\underline{\widetilde{W}}^T\right)\\
&+ \mathrm{tr}\left(\mathbf{e}\left[\underline{\phi}^T diag(\overline{\delta}) + \overline{\phi}^T diag(\underline{\delta})\right]\overline{\widetilde{W}}^T + \dot{\overline{\widetilde{W}}}\Gamma_2\overline{\widetilde{W}}^T\right)\\
&+ \mathrm{tr}\left(xe^T\widehat{\mathcal{C}}\underline{\phi}'\underline{\widetilde{V}} + \dot{\underline{\widetilde{V}}}^T\Gamma_3\underline{\widetilde{V}}\right) + \mathrm{tr}\left(\mathbf{x}\mathbf{e}^T\widehat{\mathcal{D}}\overline{\phi}'\overline{\widetilde{V}} + \dot{\overline{\widetilde{V}}}^T\Gamma_4\overline{\widetilde{V}}\right)\\
&= \mathbf{e}^T A\mathbf{e} + \mathbf{e}^T\zeta + \mathrm{tr}\left(\mathbf{e}\min(\underline{\phi},\overline{\phi})\underline{\widetilde{W}}^T + \dot{\underline{\widetilde{W}}}\Gamma_1\underline{\widetilde{W}}^T\right)\\
&+ \mathrm{tr}\left(\mathbf{e}\max(\underline{\phi},\overline{\phi})\overline{\widetilde{W}}^T + \dot{\overline{\widetilde{W}}}\Gamma_2\overline{\widetilde{W}}^T\right)\\
&+ \mathrm{tr}\left(\mathbf{x}\mathbf{e}^T\widehat{\mathcal{C}}\underline{\phi}'\underline{\widetilde{V}} + \dot{\underline{\widetilde{V}}}^T\Gamma_3\underline{\widetilde{V}}\right) + \mathrm{tr}\left(\mathbf{x}\mathbf{e}^T\widehat{\mathcal{D}}\overline{\phi}'\overline{\widetilde{V}} + \dot{\overline{\widetilde{V}}}^T\Gamma_4\overline{\widetilde{V}}\right).
\end{aligned}
$$

From the fact that

$$
\underline{\widetilde{W}} = \underline{W}_\star - \underline{\widehat{W}},\quad \overline{\widetilde{W}} = \overline{W}_\star - \overline{\widehat{W}},\quad \underline{\widetilde{V}} = \underline{V}_\star - \underline{\widehat{V}},\quad \overline{\widetilde{V}} = \overline{V}_\star - \overline{\widehat{V}}, \tag{24}
$$

we have

$$
\dot{\underline{\widetilde{W}}} = -\dot{\underline{\widehat{W}}},\quad \dot{\overline{\widetilde{W}}} = -\dot{\overline{\widehat{W}}}, \dot{\underline{\widetilde{V}}} = -\dot{\underline{\widehat{V}}},\quad \dot{\overline{\widetilde{V}}} = -\dot{\overline{\widehat{V}}}. \tag{25}
$$

Therefore, using the assumptions (19)-(22), we conclude that

$$
\begin{aligned}
\dot{v} &= \mathbf{e}^T A\mathbf{e} + \mathbf{e}^T\zeta\\
&\leq -\|\mathbf{e}\|^2|\lambda_{\min}(A)| + \|\mathbf{e}\|\|\zeta\|.
\end{aligned} \tag{26}
$$

The expression (26) is a polynomial of degree two with the variable $\|\mathbf{e}\|$. Using Equation (23), $\dot{v} < 0$. Thus, v is a decreasing function. We conclude that for every $t > 0,\quad v < v(0)$. Then, for every $t > 0$, we conclude that $0 < v < v(0)$, $v \in \mathbf{L}_\infty$,

and $\mathbf{e} \in \mathbf{L}_\infty$. using the relation (26), we have

$$\begin{aligned} 0 &< \int_0^\infty \|\mathbf{e}\|^2 \lambda_{\min}(A) dt - \int_0^\infty \|\mathbf{e}\| \|\zeta\| dt \\ &\leq -\int_0^\infty \dot{v} dt = v(0) - v(\infty) < \infty, \end{aligned} \tag{27}$$

that implies $\mathbf{e} \in \mathbf{L}_2$. At result, $\mathbf{e} \in \mathbf{L}_\infty \cap \mathbf{L}_2$, and according to the Barbalat's lemma [28], we conclude that $\mathbf{e} \longrightarrow 0$ as $t \longrightarrow \infty$. □

Table 1: MSEs of different models in the prediction of M-GTS.

Model	n_h	Para.	Epochs	A	Γ_i	SNR	Testing MSE
ANN+PSO[29]	-	-	1500	-	-	-	0.00019
ANN+PSO[29]	-	-	1500	-	-	10	0.01664
ANN+PSO[29]	-	-	1500	-	-	20	0.00026
WNN[4]	10	90	3000	-	-	-	0.00005
RBF	10	-	10	-	-	20	0.00270
RBF	20	-	10	-	-	20	0.00200
ANFIS	34	50	10	-	-	-	0.00062
ANFIS	34	50	10	-	-	15	0.00506
ANFIS	158	292	10	-	-	15	0.00464
MLP	64	320	-	-25	$100I_{64}$	15	0.00097
MLP	96	480	-	-25	$100I_{96}$	15	0.00090
MLP	128	640	-	-25	$100I_{128}$	15	0.00082
LeNN	44	44	-	-25	$10I_{44}$	15	0.00087
LeNN	124	124	-	-25	$10I_{124}$	15	0.00071
R-NN	20	320	-	-30	$30I_{20}$	-	0.00018
R-NN	30	480	-	-30	$30I_{20}$	20	0.00023
R-NN	10	160	-	-25	$100I_{10}$	15	0.00091
R-NN	20	320	-	-25	$100I_{20}$	15	0.00068
R-NN	40	640	-	-25	$100I_{40}$	15	0.00052
R-NN	30	480	-	-40	$20I_{30}$	15	0.00041
R-NN	30	480	-	-30	$20I_{30}$	15	0.00036
R-NN	30	480	-	-20	$20I_{30}$	15	0.00053
R-NN	30	480	-	-40	$50I_{30}$	15	0.00044
R-NN	30	480	-	-30	$50I_{30}$	15	0.00035
R-NN	30	480	-	-20	$50I_{30}$	15	0.00034
R-NN	30	480	-	-40	$100I_{30}$	15	0.00047
R-NN	30	480	-	-30	$100I_{30}$	15	0.00054

Remark 1. In Theorem 4.1, the learning laws are stated using the differential equations (continuous-time form) where in [14], the learning laws are stated using the difference equations (discrete-time form). For this reason, we used the continuous form of the Lyapunov stability theory for stability analysis. The presented proof in this work is much shorter and stronger than the proof presented in [14].

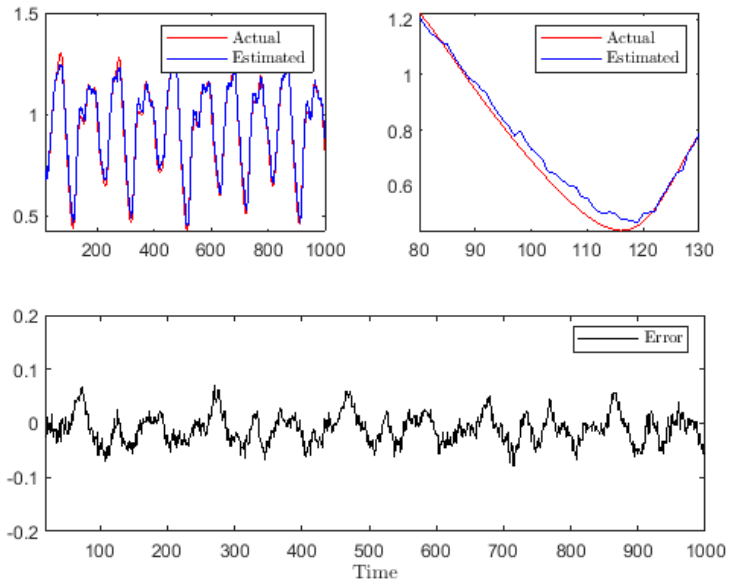

Figure 2: The M-GTS, its prediction and the error in testing of MLP with 128 hidden neurons, in the presence of noises (SNR=20).

Remark 2. Theorem 4.1 is an extension of the Theorem 1 in [17] for RELMs. In RELMs, only the parameters of hidden layer have been updated where in R-NNs, all the parameters are updated. Therefore, the situation for R-NNs are more complex than RELMs.

Remark 3. Time series usually contain big data, and a deep network may be useful for forecasting. In this work, due to the shallowness of the proposed structure for R-NNs, we concentrate on training them with a Lyapunov-based learning algorithm with an elegant and short stability proof in the continuous-time description. However, utilizing the deep R-NNs for time series prediction is a very attractive field to track by the researchers in the future [19].

5. Simulation results

In this section, two benchmark chaotic time series are simulated using the proposed approach, and the results are compared with multi-layer perceptron (MLP), Legendre neural network (LeNN) [30], radial basis function (RBF), fuzzy neural networks, and some other neural predictors in the literature. The M-GTS and the Henon map are selected for this purpose. The M-GTS refer to a delayed differ-

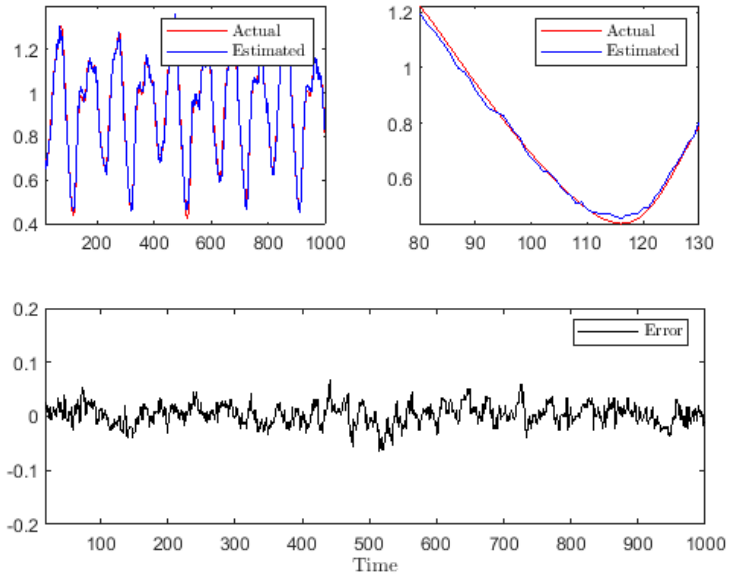

Figure 3: The M-GTS, its prediction and the errors in testing of LeNN with 124 neurons, in the presence of noises (SNR=20).

ential equation that governs some behaviors in the biological systems [31]. The Henon map is a simple model of a certain part of Lorenz model [32].

To compare the proposed predictor R-NN with some of the well-known neural predictors, the testing mean squared errors (MSEs) are shown in the Tables 1 and 2.In these tables, n_h shows the number of hidden (rough) neurons. To show the capabilities of R-NNs, some noises with different signal to noise ratio (SNR) are added to the datasets. The following formula is used for SNR:

$$SNR = 10 \log_{10} \left(\frac{\sigma_s^2}{\sigma_n^2} \right), \tag{28}$$

where σ_s^2 and σ_n^2 denote the variance of signal and the variance of noise, respectively. To implement the R-NNs, the upper and lower bounds of data are needed.

Remark 4. In the presented learning algorithm (Equations (19)-(22)), and therefore, in the Tables 1 and 2, $\Gamma_i (i = 1, 2, 3, 4)$ is a positive definite matrix of learning rates. For the simulation in this work, we choose some diagonal matrices for learning rates Γ_i, such as $100 I_n$ where I_n shows the identity matrix of order n. It must be mentioned that in the structure of learning laws, Γ_i^{-1} is used.

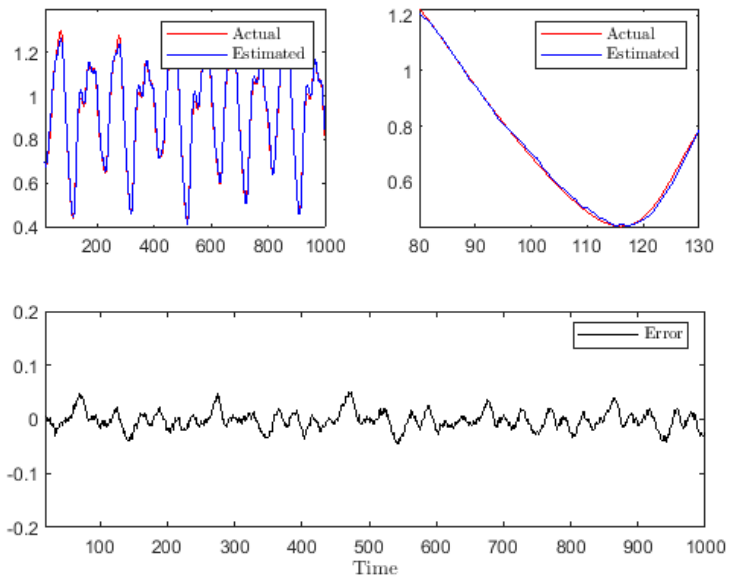

Figure 4: The M-GTS, its prediction and the errors in testing of R-NN with 40 HRNs, in the presence of noises (SNR=20).

5.1 The Mackey-Glass time series prediction

The M-GTS is generated using the equation

$$\dot{z}(t) = \frac{0.2z(t-\tau)}{1+z^{10}(t-\tau)} - 0.1z(t), \tag{29}$$

where $\tau = 17$. The one-step ahead prediction of (29) is done by MLP, LeNN, RBF, adaptive neuro fuzzy inference system (ANFIS), and R-NN where the activation functions of hidden neurons in MLP and R-NN are sinusoidal. Besides, the results are compared with some other works in the literature such as wavelet neural network (WNN) [4], and ANN with particle swarm optimization (ANN+PSO) [29].

The initial values of the weights in models are random numbers in the interval $[-0.5, 0.5]$. The input vector of MLP and LeNN is $\mathbf{x}(t) = [z(t), z(t-6), z(t-12), 1]^T$, the input vector of ANFIS is $\mathbf{x}(t) = [z(t), z(t-6), z(t-12), z(t-18), 1]^T$, and the input vector of R-NN is

$$\mathbf{x}(t) = [\overline{z}(t), \underline{z}(t), \overline{z}(t-6), \underline{z}(t-6), \overline{z}(t-12), \underline{z}(t-12), 1]^T, \tag{30}$$

where $\overline{z}$ is the upper bound of z, and $\underline{z}$ is the lower bound of z. The MSEs of one-step ahead prediction of M-GTS with different models are listed in Table 1.

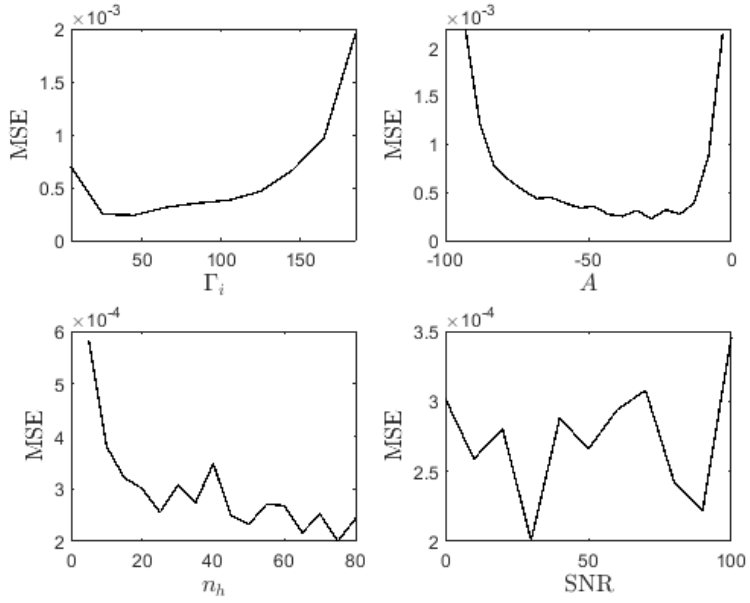

Figure 5: The testing MSEs of R-NN in the prediction of M-GTS, obtained by changing the learning rates $\Gamma_i (i = 1, 2, 3, 4)$, parameter A, the number of HRNs (n_h), and SNR.

In this table, Γ_1, Γ_2, Γ_3 and Γ_4 denote the learning rates, and I_{n_h} is the identity matrix of size n_h. The hyper-parameters for LeNN are chosen as

$$n_h = 24,\ 44,\ 84,\ 124,\quad \Gamma_1,\ \Gamma_2 = 10 I_{n_h}. \tag{31}$$

Besides, the simulation results for different hyper-parameters A and Γ_i $(i = 1, 2, 3, 4)$ in the usage of R-NN (with 30 HRNs) are stated in Table 1. According to these results, we can conclude that the optimal value for A is near to -30 and the optimal value for Γ_i $(i = 1, 2, 3, 4)$ is near to $50 I_{30}$, empirically.

The M-GTS, its predictions and the errors in testing of MLP with 128 hidden neurons, LeNN with 124 neurons, and R-NN with 40 HRNs (in the presence of noises (SNR=20)) are shown in Figures 2 to 4.

Figure 5 shows the testing MSEs of R-NN in the prediction of M-GTS (SNR=20), obtained by changing the different parameters in the algorithm design. These parameters include the learning rates Γ_i $(i = 1, 2, 3, 4)$, number of HRNs n_h, SNR, and the parameter A, where $A\mathbf{x}$ models the linear part of the time series as described in (11). According to the Figure 5, we can conclude that the increasing of the learning rates Γ_i $(i = 1, 2, 3, 4)$ results in the increasing of testing MSEs. Besides, due to the numerical errors, very small value for Γ_i $(i = 1, 2, 3, 4)$ does

not lead to the best results.

Due to the quadratic form of the MSEs, the curve obtained with changing the parameter A in Figure 5, is similar to a parabolic curve and its minimum occurs near the $A = -30$. Therefore, we can conclude that in the prediction of the M-GTS, to approximate the linear part, the best value for A belongs to the interval $[-35, -20]$. It seems that the nonsmoothness of this curve, is related to the numerical errors and the continuity of M-GTS. According to the Figure 5, increasing the number of HRNs results in decreasing the testing MSEs. In this figure, there is no regularity in the curve obtained by changing the SNR for M-GTS dataset, and the dataset is not very affected from the noises. The Remark 5 in Section 5.2 gives more details about this graph.

From the Table 1 and Figures 2 to 5, we can conclude the following results in the prediction of M-GTS:

- In the presence of noises, the performance of R-NN is better than the other models.
- The performance of LeNN is better than MLP, RBF, and ANFIS.
- The performance of WNN (without noises) is better than R-NN.
- The performance of R-NN (with or without noises) is a bit better than ANN+PSO.
- Increasing the number of HRNs results in decreasing the testing MSE.
- The dataset of M-GTS is robust against the noises.

5.2 Henon map prediction

The Henon map is generated using the equation

$$z(k+1) = 1 - \alpha z(k)^2 + \beta z(k-1). \tag{32}$$

In this work, we suppose that $\alpha = 1.4$ and $\beta = 0.3$. It is possible to write the equation (32) with two variables z_1 and z_2 as follow:

$$\begin{cases} z_1(k+1) &= 1 - \alpha z_1(k)^2 + \beta z_2(k), \\ z_2(k+1) &= z_1(k). \end{cases} \tag{33}$$

The one-step ahead prediction of (32) is done by MLP, LeNN, and R-NN where the activation function of hidden neurons in MLP and R-NN is the hyperbolic tangent. Besides, the results are compared with some other models such as interval type 2 fuzzy neural networks (IT2FNN) [33], RBF [34], deep belief net (DBN) [35].

Table 2: MSEs of different models in the prediction of Henon map time series.

Model	n_h	Para.	Epochs	A	Γ_i	SNR	MSE
IT2FNN[33]	8 rules	128	100	-	-	-	0.00094
RBF[34]	16	-	1000	-	-	-	0.00220
RBF[34]	25	-	1000	-	-	-	0.00012
RBF[34]	101	-	1000	-	-	-	0.00007
DBN[35]	-	-	-	-	-	-	0.00004
MLP	20	140	-	0.05	$10I_{20}$	40	0.00023
MLP	40	280	-	0.05	$10I_{40}$	40	0.00014
MLP	63	441	-	0.05	$10I_{63}$	40	0.00011
LeNN	24	24	-	0.05	$10I_{24}$	40	0.00010
LeNN	44	44	-	0.05	$10I_{44}$	40	0.00009
LeNN	64	64	-	0.05	$10I_{64}$	40	0.00007
R-NN	20	440	-	0.05	$10I_{20}$	-	0.00001
R-NN	6	140	-	0.05	$10I_{6}$	40	0.00006
R-NN	12	280	-	0.05	$10I_{12}$	40	0.00004
R-NN	20	440	-	0.05	$10I_{20}$	40	0.00002
R-NN	12	280	-	0	$10I_{12}$	50	0.00071
R-NN	12	280	-	0.05	$10I_{12}$	50	0.00005
R-NN	12	280	-	0.1	$10I_{12}$	50	0.00053
R-NN	12	280	-	0	$20I_{12}$	50	0.00057
R-NN	12	280	-	0.05	$20I_{12}$	50	0.00002
R-NN	12	280	-	0.1	$20I_{12}$	50	0.00053
R-NN	12	280	-	0	$30I_{12}$	50	0.00064
R-NN	12	280	-	0.05	$30I_{12}$	50	0.00009
R-NN	12	280	-	0.1	$30I_{12}$	50	0.00058

The initial values of the weights in models are random numbers in the interval $[-0.05, 0.05]$. The input vector of MLP and LeNN is $\mathbf{x} = [z_1(k), z_2(k), z_1(k-1), z_2(k-2)]^T$, and the input vector of R-NN is

$$\mathbf{x} = [\overline{z}_1(k), \underline{z}_1(k), \overline{z}_2(k), \underline{z}_2(k), \overline{z}_1(k-1), \underline{z}_1(k-1), \underline{z}_2(k-1), \overline{z}_2(k-1), 1]^T . \quad (34)$$

The MSEs of one-step ahead prediction of Henon map with the models are listed in Table 2. In this table, the hyper-parameters for MLP are chosen as

$$n_h = 20,\ 40,\ 63,\quad \Gamma_1, \Gamma_2 = 10I_{n_h}, \quad (35)$$

and the hyper-parameters for LeNN are chosen as

$$n_h = 24,\ 44,\ 64,\quad \Gamma_1, \Gamma_2 = 10I_{n_h}, \quad (36)$$

where Γ_1, and Γ_2 denote the learning rates, and I_{n_h} shows the identity matrix of size n_h. The hyper-parameters for R-NN are chosen as

$$n_h = 6,\ 12,\ 20,\quad \Gamma_1,\ \Gamma_2,\ \Gamma_3,\ \Gamma_4 = 10I_{n_h},\ 20I_{n_h},\ 30I_{n_h}. \quad (37)$$

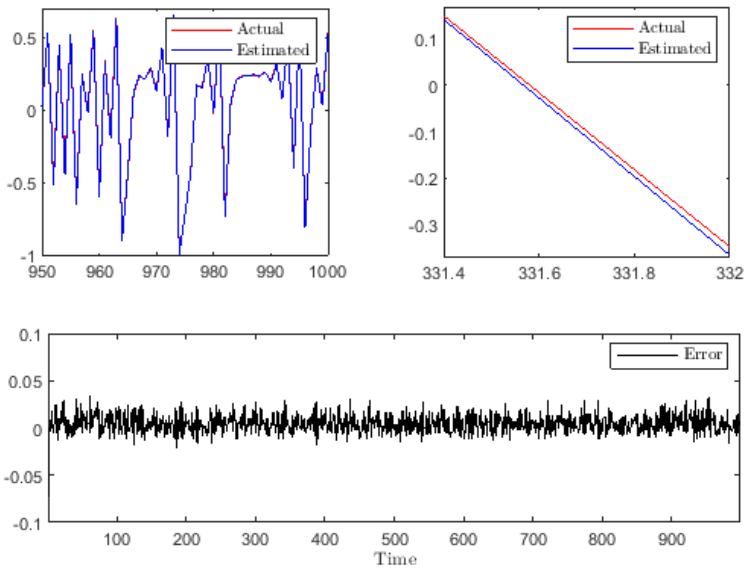

Figure 6: The Henon map, its prediction, and the error in the testing of MLP with 40 hidden neurons, in the presence of noises (SNR=40).

where Γ_1, Γ_2, Γ_3, and Γ_4 denote the learning rates.

Besides, the simulation results for different hyper-parameters A and Γ_i $(i = 1, 2, 3, 4)$ in the usage of R-NN (with 12 HRNs) are stated in Table 2. According to these results, we can conclude that the optimal value for A is near to 0.05, and the optimal value for Γ_i $(i = 1, 2, 3, 4)$ is near to $20I_{12}$, empirically.

The Henon map, its prediction, and the error in the testing of MLP with 40 hidden neurons, LeNN with 64 neurons, and R-NN with 12 HRNs are shown in Figures 6 to 8, in the presence of noises (SNR=40).

Figure 9 shows the testing MSEs of R-NN in the prediction of Henon map time series (SNR=50), obtained by changing the different parameters in the algorithm design. These parameters include the learning rates Γ_i $(i = 1, 2, 3, 4)$, number of HRNs n_h, SNR, and the parameter A, where $A\mathbf{x}$ models the linear part of the time series as described in (11). According to the Figure 9, we can conclude that the increasing of the learning rates Γ_i $(i = 1, 2, 3, 4)$ results in the increasing of testing MSEs.

Besides, this figure shows the testing MSEs of R-NN with 12 HRNs in the prediction of Henon map time series (SNR=50), obtained by changing the parameter A. Due to the quadratic form of the MSEs, this graph is a parabolic curve and its minimum occurs in $A = 0.05$. Therefore, we can conclude that in the predic-

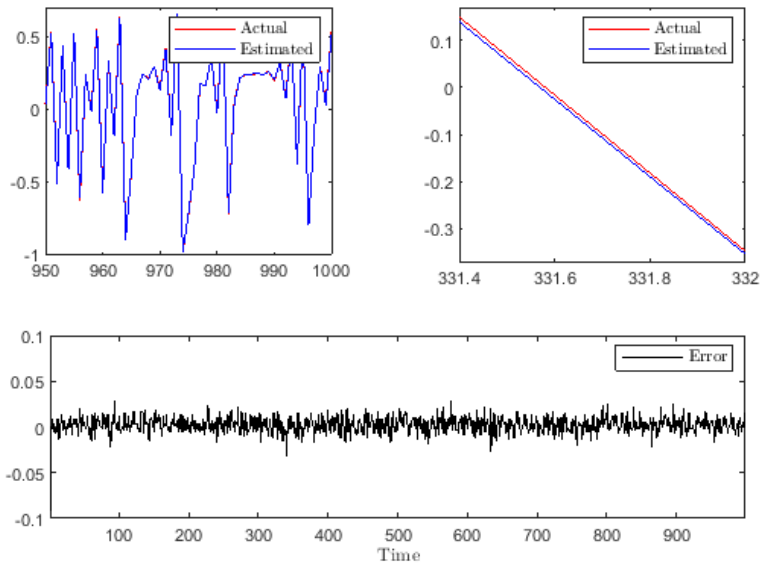

Figure 7: The Henon map, its prediction, and the error in testing of LeNN with 64 neurons, in the presence of noises (SNR=40).

tion of Henon map time series, to approximate the linear part, the best value for A is 0.05. According to the Figure 9, increasing the number of HRNs results in decreasing the testing MSEs. In this figure, increasing the SNR for Henon map dataset results in decreasing the testing MSE.

Remark 5. In Figure 5, there was no regularity in the curve obtained by changing the SNR for M-GTS, where in Figure 9, increasing the SNR for Henon map dataset results in decreasing the testing MSE. The reason of this event is concerned with the nature of these datasets. The M-GTS dataset arises from a continuous-time formula, where the Henon map dataset arises from a discrete-time formula. The discrete-time systems are very vulnerable against the noises, where the continuous-time systems are robust against the noises [17]. Therefore, the continuous-time models are more reliable than the discrete-time models.

From Table 2, and Figures 6 to 9, we can conclude the following results in the prediction of Henon map time series:

- The performance of R-NN is better than the other predictors.
- The performance of LeNN is better than MLP, and IT2FNN.
- The performance of DBN (without noises) is better than IT2FNN, RBF, and LeNN.

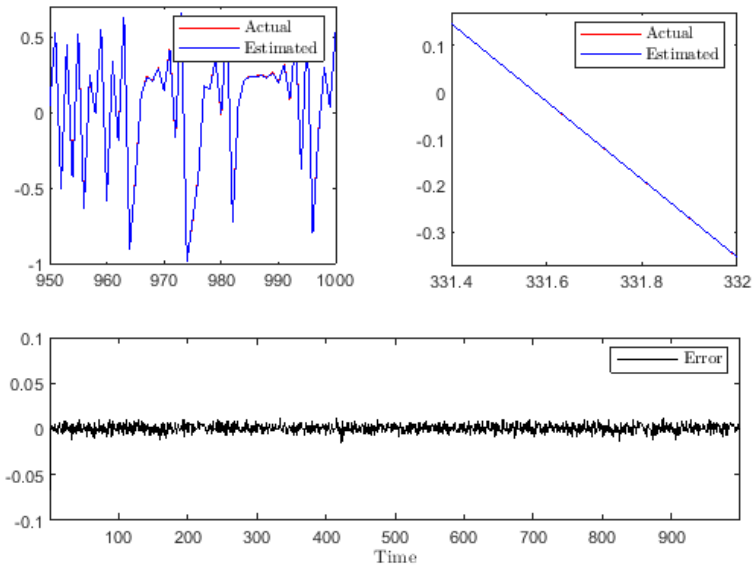

Figure 8: The Henon map, its prediction, and the error in the testing of R-NN with 12 hidden rough neurons, in the presence of noises (SNR=40).

- Increasing the number of HRNs in R-NN results in decreasing the testing MSE.

- The dataset of Henon map time series is very vulnerable against the noises.

6. Conclusion

This work proposed the R-NNs for the prediction of chaotic time series where they are trained with a continuous-time Lyapunov-based learning algorithm, and its stability is proved. Simulation results show the efficiencies of R-NNs in the prediction of time series. Time series usually contain big data, and a deep network may be useful for forecasting. Therefore, future works focus on the usage of deep R-NNs for time series prediction. Besides, we try to use the R-NNs to solve the regression problems in the other applied sciences.

Conflicts of Interest. The authors declare that they have no conflicts of interest regarding the publication of this article.

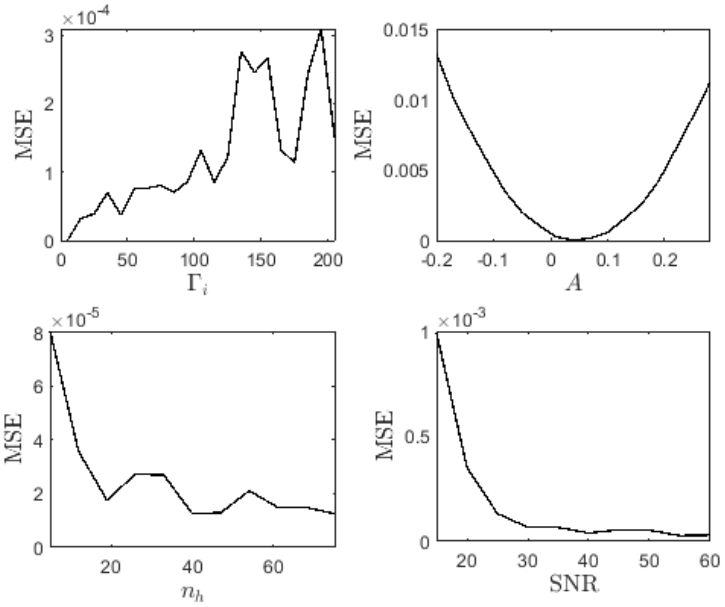

Figure 9: The testing MSEs of R-NN in the prediction of Henon map time series, obtained by changing the learning rates $\Gamma_i (i = 1, 2, 3, 4)$, parameter A, the number of HRNs (n_h), and SNR.

References

[1] D. C. Montgomery, C. L. Jennings and M. Kulahci, *Introduction to Time Series Analysis and Forecasting*, John Wiley & Sons, Hoboken, New Jersey, 2015.

[2] M. M. Gupta, L. Jin and N. Homma, *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*, John Wiley & Sons, Inc. 2003.

[3] R. J. Frank, N. Davey and S. P. Hunt, Time series prediction and neural networks, *J. Intell. Robot. Syst.* **31** (2001) 91 − 103, https://doi.org/10.1023/A:1012074215150.

[4] Y. Chen, B. Yang and J. Dong, Time-series prediction using a local linear wavelet neural network, *Neurocomputing*, **69** (2006) 449 − 465, https://doi.org/10.1016/j.neucom.2005.02.006.

[5] A. Gholipour, B. N. Araabi and C. Lucas, Predicting chaotic time series using neural and neurofuzzy models: A comparative study, *Neural Process Lett.* **24** (2006) 217 − 239, https://doi.org/10.1007/s11063-006-9021-x.

[6] D. O. Faruk, A hybrid neural network and ARIMA model for water quality time series prediction, *Eng. Appl. Artif. Intell.* **23** (2010) 586 − 594, https://doi.org/10.1016/j.engappai.2009.09.015.

[7] R. Chandra, S. Goyal and R. Gupta, Evaluation of deep learning models for multi-step ahead time series prediction, *IEEE Access* **9** (2021) 83105 − 83123, https://doi.org/10.1109/ACCESS.2021.3085085.

[8] A. Tealab, Time series forecasting using artificial neural networks methodologies: A systematic review, *Future Computing Inform. J.* **3** (2) (2018) 334−340, https://doi.org/10.1016/j.fcij.2018.10.003.

[9] P. Lingras, Rough neural networks, *Proceedings of the 6th international conference on information processing and management of uncertainty (IPMU)* Granada, Spain (1996) 1445 − 1450.

[10] D. Yamaguchi, F. Katayama, M. Takahashi, M. Arai and K. J. Mackin, The medical diagnostic support system using extended rough neural network and multiagent, *Artif Life Robot.* **13** (2008) 184 − 187, https://doi.org/10.1007/s10015-008-0543-3.

[11] R. K. Nowicki, *Rough Set–Based Classification Systems*, Springer, 2019.

[12] K. Sasirekha and K. Thangavel, Biometric face classification with the hybridised rough neural network, *Int. J. Biom.* **12** (2) (2020) 193 − 217, https://doi.org/10.1504/IJBM.2020.107717.

[13] S. M. J. Alehasher and M. Teshnehlab, Implementation of rough neural networks with probabilistic learning for nonlinear system identification, *J. Control*, **6** (1) (2012) 41 − 50.

[14] G. Ahmadi and M. Teshnehlab, Designing and implementation of stable sinusoidal rough-neural identifier, *IEEE Trans. Neural Netw. Learn. Syst.* **28** (8) (2017) 1774 − 1786, https://doi.org/10.1109/TNNLS.2016.2551303.

[15] G. Ahmadi, M. Teshnehlab and F. Soltanian, A higher order online Lyapunov-based emotional learning for rough-neural identifiers, *Control Optim. Appl. Math.* **3** (1) (2018) 87 − 108, https://doi.org/10.30473/coam.2019.40779.1083.

[16] G. Ahmadi and M. Teshnehlab, Identification of multiple input-multiple output non-linear system cement rotary kiln using stochastic gradient-based rough-neural network, *J. AI Data Min.* **8** (3) (2020) 417 − 425, https://doi.org/10.22044/jadm.2020.8865.2021.

[17] G. Ahmadi, Stable rough extreme learning machines for the identification of uncertain continuous-time nonlinear systems, *Control Optim. Appl. Math.* **4** (1) (2019) 83 − 101, https://doi.org/10.30473/coam.2020.51511.1137.

[18] P. Lingras, Comparison of neofuzzy and rough neural networks, *Inf. Sci.* **110** (3-4) (1998) 207 − 215, https://doi.org/10.1016/S0020-0255(97)10045-7.

[19] M. Khodayar, O. Kaynak and M. E. Khodayar, Rough deep neural architecture for short-term wind speed forecasting, *IEEE Trans. Industr. Inform.* **13** (2017) 2770 − 2779, https://doi.org/10.1109/TII.2017.2730846.

[20] H. Jahangir, H. Tayarani, S. Baghali, A. Ahmadian, A. Elkamel, M. A. Golkar and M. Castilla, A novel electricity price forecasting approach based on dimension reduction strategy and rough artificial neural networks, *IEEE Trans. Industr. Inform.* **16** (4) (2020) 2369 − 2381, https://doi.org/10.1109/TII.2019.2933009.

[21] H. Jahangir, H. Tayarani, A. Ahmadian, M. A. Golkar, J. Miret, M. Tayarani and H. Oliver Gao, Charging demand of plug-in electric vehicles: forecasting travel behavior based on a novel rough artificial neural network approach, *J. Clean. Prod.* **229** (2019) 1029-1044, https://doi.org/10.1016/j.jclepro.2019.04.345.

[22] B. Cao, J. Zhao, Z. Lv, Y. Gu, P. Yang and S. K. Halgamuge, Multiobjective evolution of fuzzy rough neural network via distributed parallelism for stock prediction, *IEEE Transactions on Fuzzy Systems*, **28** (2020) 939 − 952, https://doi.org/10.1109/TFUZZ.2020.2972207.

[23] A. Sheikhoushaghi, N. Yarahmadi Gharaei and A. Nikoofard, Application of rough neural network to forecast oil production rate of an oil field in a comparative study, *J. Pet. Sci. Eng.* **209** (2022) p. 109935, https://doi.org/10.1016/j.petrol.2021.109935.

[24] G. Ahmadi and M. Dehghandar, Mackey-glass time series prediction using rough-neural networks, *Proceedings of the 51th annual iranian mathematics conference*, Kashan, (2021) 1243 − 1248.

[25] Z. Pawlak, Rough sets, *Int. J. Comput. Inform. Sci.* **11** (1982) 341 − 356, https://doi.org/10.1007/BF01001956.

[26] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, Cambridge University Press, 2004.

[27] A. Basharat and M. Shah, Time series prediction by chaotic modeling of nonlinear dynamical systems, *IEEE 12th International Conference on Computer Vision, Kyoto, Japan* (2009) 1941 − 1948, https://doi.org/10.1109/ICCV.2009.5459429.

[28] P. A. Ioannou and J. Sun, *Robust adaptive control*, Prentice Hall, 1996.

[29] C. H. López-Caraballo, I. Salfate, J. A. Lazzús, P. Rojas, M. Rivera and L. Palma-Chilla, Mackey-glass noisy chaotic time series prediction by a swarm-optimized neural network, *J. Phys.: Conf. Ser.* **720** (2016) p. 012002, https://doi.org/10.1088/1742-6596/720/1/012002.

[30] J. C. Patra and C. Bornand, Nonlinear dynamic system identification using Legendre neural network, *The 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain,* (2010) 1 – 7, https://doi.org/10.1109/IJCNN.2010.5596904.

[31] M. C. Mackey and L. Glass, Oscillation and chaos in physiological control systems, *Science*, **197** (1977) 287 – 289, https://doi.org/10.1126/science.267326.

[32] M. Hénon, A two-dimensional mapping with a strange attractor, *Commun. Math. Phys.* **50** (1976) 69 – 77, https://doi.org/10.1007/BF01608556.

[33] C. H. Lee, F. Y. Chang and C. M. Lin, An efficient interval type-2 fuzzy CMAC for chaos time-series prediction and synchronization, *IEEE Trans. Cybern.* **44** (3) (2013) 329 – 341, https://doi.org/10.1109/TCYB.2013.2254113.

[34] A. E. Omidvar, Configuring radial basis function network using fractal scaling process with application to chaotic time series prediction, *Chaos, Solit. & Fractals*, **22** (4) (2004) 757 – 766, https://doi.org/10.1016/j.chaos.2004.03.008.

[35] T. Kuremoto, M. Obayashi, K. Kobayashi, T. Hirata and S. Mabu, Forecast chaotic time series data by DBNs, *7th International Congress on Image and Signal Processing*, Dalian, China (2014) 1130 – 1135, https://doi.org/10.1109/CISP.2014.7003950.

Ghasem Ahmadi
Department of Mathematics,
Payame Noor University,
Tehran, Iran
e-mail: g.ahmadi@pnu.ac.ir

Mohammad Dehghandar
Department of Mathematics,
Payame Noor University,
Tehran, Iran
e-mail: m_dehghandar@pnu.ac.ir