



Hybrid deep learning diagonal recurrent neural network controller for nonlinear systems

Ahmad M. El-Nagar¹ · Ahmad M. Zaki² · F. A. S. Soliman² · Mohammad El-Bardini¹

Received: 23 June 2021 / Accepted: 25 July 2022 / Published online: 23 August 2022
© The Author(s) 2022

Abstract

In the present paper, a hybrid deep learning diagonal recurrent neural network controller (HDL-DRNNC) is proposed for nonlinear systems. The proposed HDL-DRNNC structure consists of a diagonal recurrent neural network (DRNN), whose initial values can be obtained through deep learning (DL). The DL algorithm, which is used in this study, is a hybrid algorithm that is based on a self-organizing map of the Kohonen procedure and restricted Boltzmann machine. The updating weights of the DRNN of the proposed algorithm are developed using the Lyapunov stability criterion. In this concern, simulation tasks such as disturbance signals and parameter variations are performed on mathematical and physical systems to improve the performance and the robustness of the proposed controller. It is clear from the results that the performance of the proposed controller is better than other existent controllers.

Keywords Hybrid deep learning · Diagonal recurrent neural network · Nonlinear system · Lyapunov stability

1 Introduction

Since many real-time implementations of nonlinear systems involve nonlinearity, nonlinear systems play an important role in engineering research that are defined as systems whose manner is not proportional to their inputs [1]. These systems suffer inherent uncertainty, time-varying parameters and nonlinear dynamic behavioral. In this concern, non-optimal control suffers from some limitations due to the assumptions made for the control system such as linearity and time-invariance. Therefore, the non-optimal

control methods are not suitable for controlling nonlinear systems in practical applications [2, 3]. These problems can be overcome by optimal control techniques. Artificial intelligence (AI)-based controllers are one of these techniques, which have many advantages, such as [4, 5]: (1) it can lead to better performance when properly tuned. (2) It requires less tuning effort than non-optimal controllers. (3) It can be designed based on data from the real system or plant if an expert knowledge is not available. (4) It can be designed using a combination of linguistic and response-based information. [6–9]. The neural networks (NNs) are considered as one of AI, which are a series of algorithms to recognize underlying relationships in a set of data through a process, which like the operation of human brain [10].

In this concern, there are various structures of NNs such as recurrent neural networks (RNNs) [11–14] and multi-layer feed-forward neural networks (MLFFNNs) [15, 16]. MLFFNN is called static network, where there are not tapped delay lines. Several MLFFNN models from observational data were created for predicting the groundwater levels [17]. The NN controller was designed as a direct adaptive inverse control based on MLFFNN to control and estimate the model of nonlinear plants [18]. In [19], the researchers designed MLFFNN for classification of nonlinear mappings based on input and output samples.

✉ Ahmad M. El-Nagar
ahmed.elnagar@el-eng.menofia.edu.eg

Ahmad M. Zaki
ahmed_zaki889@yahoo.com

F. A. S. Soliman
fouad.saad.soliman@gmail.com

Mohammad El-Bardini
dralbardini@el-eng.menofia.edu.eg

¹ Department of Industrial Electronics and Control Engineering, Faculty of Electronic Engineering, Menoufia University, Menof 32852, Egypt

² Department of Electronics and Computers Engineering, Nuclear Materials Authority, El-Maadi, 530, Cairo, Egypt

However, there are tapped delay lines in RNN, and this is called a dynamic network. The RNN is more robust than MLFFNN because it contains the MLFNN framework with tapped delay lines [20]. Various RNN structures exist in this regard, including Elman NNs; the feedback connections from the output to the input of the hidden layer are performed via a context layer. Another RNN is Jordan NNs; the feedback connections from the output to the input layer are performed via a state layer. [21–23]. Recently, the fully connected RNN (FCRNN) was modified to provide a diagonal RNN (DRNN). In the hidden layer, the DRNN contains self-recurrent neurons that feed only their output back into themselves, not to other neurons in the same layer. [24–26]. In [27], in order to achieve high performance of the shunt active power filter, researchers designed a controller based on RNN. A self-organizing RNN for the nonlinear model predictive controller was designed to foresee the nonlinear systems behavior [28]. In [29], a flexible manipulator was designed with a DRNN controller to limit backward vibration, which is performed based on a shaking control signal generator and an online identification system. The DRNN was introduced as a controller and an observer for estimating the anonymous dynamics of the nonlinear system [30]. DRNN was developed to determine the optimal parameters of the PID controller for controlling induction motors [31]. In [32], based on the control inputs and current quadrotor states, researchers developed the PID controller using virtual sensing based on DRNNs and Kalman filters to predict the immeasurable cases of the quadrotor system.

Machine learning (ML) is one of the applications of AI that can automatically learn from experience without explicit programming. ML focuses on the development of programs that can access data and use it to learn them [33]. In this significance, the deep learning (DL) is a part of a wider family of ML based on ANN's that learn representations either supervised, unsupervised, or semi-supervised [34]. In this concern, there are various applications of DL exist as follows: (1) In automation systems, an approach for detecting and assessing food waste trays based on hierarchical DL algorithm was presented [35], (2) in medicine field, a DL algorithm was used to classify and predict mutations from non-small cell lung cancer histopathology images [36], (3) in agriculture field, a DL algorithm was introduced to locate paddy fields at the pixel level for a whole year long and for each temporal instance based on real imagery datasets of different landscapes from 2016 to 2018 [37], and (4) in recognition, a DL algorithm was used for real-time modeling of the human activity recognition with smartphones [38]. According to definitions [39, 40], the DL of NN's includes two steps: firstly is the unsupervised training and secondly is using the weights from the unsupervised training for initializing the multilayer NNs.

This is considered as the main advantage of DL because the initializing weights process is very critical issue.

1.1 Literature review

In [41], the parameters of the classical PID controller were tuned based on DL for controlling maglev train, which is a new type of the ground transports. The deep NNs (DNN) were introduced for dynamical systems modeling based on complex manner [42]. Three DNN structures are trained on successive data for studying validation of these networks in modeling of dynamical systems. In [43], DL was designed for analyzing the performance of a nonlinear continuous stirred tank reactor, which trained its weights tuned by hybrid algorithm. The DL was introduced as a hybrid algorithm with the fuzzy system for tuning the parameters of the PID controller [44], which was used for controlling the speed of brushless DC motor. In [45], DL controller was introduced, which is performed based on the MLFFNN and the RBM. It is used for initializing the weights values of a network for the nonlinear systems. In [46], DL was introduced for modeling the nonlinear systems based on Elman RNN and restricted Boltzmann machine (ERNN-RBM), which is considered as an unsupervised method for initializing only the first layer.

1.2 Motivation

It is evident from the literature review that DL applications are widely used for modeling systems and it does not cover the control research. Since nonlinear systems suffer from external disturbances and uncertainties, the main purpose of the present paper is to shed further light on the design of stable controllers for overcoming nonlinear system problems. In this concern, self-organizing map (SOM) is an unsupervised learning algorithm trained using dimensionality reduction (typically two-dimensional), discretized representation of input space of the training samples, and called a map. It differs from other ANN as it depends on the competitive learning and not the error-correction learning (like backpropagation with gradient descent). It uses a neighborhood function to preserve the topological properties of the input space to reduce data by creating a spatially organized representation, and also, it helps to discover the correlation between data [47–50]. On the other hand, the RBM is an unsupervised learning algorithm that makes inferences from input data without labeled responses. The controllers and models based on NNs are always stuck with the initialized weights. If the initialized weights are not appropriate, the network gets stuck in local minima and leads the training process to a wrong ending and the network becomes infeasible to train therefore. Therefore, RBM is used to overcome this problem [46, 51]. Hence, a

merge utilizing the features of SOM and RBM is proposed to improve the learning performance of the proposed network. To control the nonlinear systems, the hybrid deep learning DRNN controller (HDL-DRNNC) based on SOM and RBM is proposed. Initial weights for the DRNN are obtained using a hybrid deep learning (HDL) procedure, which is regarded as an unsupervised learning procedure. The HDL is performed based on RBM and self-organizing map (SOM) of the Kohonen procedure.

To ensure the stability of the adaptation parameters laws, the Lyapunov procedure is applied. The proposed HDL-DRNNC is trained quickly for keeping the trajectory and overcoming the system parameters variations and external disturbances. As shown in the simulation results, these features of the proposed HDL-DRNNC make it more robust than those of other controllers under the same conditions.

1.3 Novelties and contributions

The main contributions of the paper are summarized as:

- A new HDL for DRNN controller is proposed for nonlinear systems.
- Developing the updating weights law for the DRNN of the proposed algorithm using Lyapunov theory to achieve stability.
- Compared to other existing controllers, the HDL-DRNNC can handle problems of system uncertainties in both a mathematical system and a physical system.

The organization of the paper is as follows: the DRNN structure is exhibited in Sect. 2. The proposed HDL-DRNNC is introduced in Sect. 3. The weights updating based on Lyapunov stability criterion is introduced in Sect. 4. The HDL-DRNNC pseudocode is explained in Sect. 5. The simulation results for the mathematical and physical nonlinear systems are introduced in Sect. 6. At final, Sect. 7 exhibits the conclusion, which followed by the references.

2 Diagonal recurrent neural network structure

As shown in Fig. 1, the structure of DRNN consists of four layers, namely two hidden layers, an input layer, and an output layer.

Input layer: the external input vector is represented by $E(\kappa) = [e_{x1}(\kappa) \cdots e_{xn}(\kappa)]^T$ and $\mathfrak{R}^{(1)}(\kappa)$ is the input weight matrix, which links the input vector to the hidden layer (1) neurons and it is represented as:

$$\mathfrak{R}^{(1)}(\kappa) = \begin{bmatrix} \mathfrak{R}_{11}^{(1)}(\kappa) & \cdots & \mathfrak{R}_{1n}^{(1)}(\kappa) \\ \vdots & \ddots & \vdots \\ \mathfrak{R}_{j1}^{(1)}(\kappa) & \cdots & \mathfrak{R}_{jn}^{(1)}(\kappa) \end{bmatrix} \tag{1}$$

Generally, $\mathfrak{R}_{jn}^{(1)}(\kappa)$ denote the input weight between input neuron n and the hidden layer (1) neuron J .

Hidden layer (1): the output of each node is denoted by $\chi_j^{(1)}(\kappa)$, which is specified as:

$$\Lambda_j^{(1)}(\kappa) = \chi_j^{(1)}(\kappa - 1)\mathfrak{R}_j^{D1}(\kappa) + \sum_{i=1}^n \mathfrak{R}_{ji}^{(1)}(\kappa)e_{xi}(\kappa) + T_j(\kappa), \tag{2}$$

$j = 1, \dots, J$

$$\chi_j^{(1)}(\kappa) = f\left(\Lambda_j^{(1)}(\kappa)\right), \quad j = 1, \dots, J \tag{3}$$

where $\mathfrak{R}^{D1}(\kappa) = [\mathfrak{R}_1^{D1}(\kappa) \cdots \mathfrak{R}_J^{D1}(\kappa)]^T$ denotes the diagonal weight vector at the hidden layer (1), J is the nodes number, $T_j(\kappa)$ denotes the threshold value for every node, and $f()$ denotes hyperbolic tangent function, which is defined as:

$$f(v) = \tanh(v) \tag{4}$$

and its derivative can be defined by $f'(v) = 1 - f^2(v)$.

Hidden layer (2): the output of each node is denoted by $\chi_m^{(2)}(\kappa)$, which is specified as:

$$\Lambda_m^{(2)}(\kappa) = \chi_m^{(2)}(\kappa - 1)\mathfrak{R}_m^{D2}(\kappa) + \sum_{j=1}^J \mathfrak{R}_{mj}^{(2)}(\kappa)\chi_j^{(1)}(\kappa) + T_m(\kappa), \tag{5}$$

$m = 1, \dots, M$

$$\chi_m^{(2)}(\kappa) = f\left(\Lambda_m^{(2)}(\kappa)\right), \quad m = 1, \dots, M \tag{6}$$

where $\mathfrak{R}^{D2}(\kappa) = [\mathfrak{R}_1^{D2}(\kappa) \cdots \mathfrak{R}_M^{D2}(\kappa)]^T$ denotes the diagonal weight vector at the hidden layer (2), M denotes the nodes number, $T_m(\kappa)$ denotes the threshold value for each node and $\mathfrak{R}^{(2)}(\kappa)$ denotes the weights matrix between the hidden layer (1) and the hidden layer (2), which is defined as:

$$\mathfrak{R}^{(2)}(\kappa) = \begin{bmatrix} \mathfrak{R}_{11}^{(2)}(\kappa) & \cdots & \mathfrak{R}_{1J}^{(2)}(\kappa) \\ \vdots & \ddots & \vdots \\ \mathfrak{R}_{M1}^{(2)}(\kappa) & \cdots & \mathfrak{R}_{MJ}^{(2)}(\kappa) \end{bmatrix} \tag{7}$$

Generally, $\mathfrak{R}_{MJ}^{(2)}(\kappa)$ denote the weight between the hidden layer (1) neuron J and the hidden layer (2) neuron M .

Output layer: its output is denoted by $u(\kappa)$, which is calculated as:

$$u(\kappa) = \sum_{m=1}^M \mathfrak{R}_m^{(3)}(\kappa) \chi_m^{(2)}(\kappa) + T(\kappa) \tag{8}$$

where $T(\kappa)$ denotes the threshold value and $\mathfrak{R}^{(3)}(\kappa) = [\mathfrak{R}_1^{(3)}(\kappa) \cdots \mathfrak{R}_M^{(3)}(\kappa)]^T$ denotes the weight vector between the hidden layer (2) and the output layer.

3 Hybrid deep learning diagonal recurrent neural network controller

The initializing weights process for DRNN controllers is very critical issue. Where, if this process is zero or not suitable, then the DRNN will stumble in local minimum and it will lead to wrong network termination as learning due to the initial layers learning of a network will become impossible [46]. This issue may be leading the controller to become unstable. For this reason, DL is proposed. In this regard, any NN with more than one hidden layer is referred to as a deep network, which can be learned by DL. The proposed HDL-DRNNC consists of DRNN that can be trained by DL. The DL algorithm based on SOM and RBM is considered as an unsupervised learning for initializing the weights values of the DRNN. DRNN’s two hidden layers, which are described in the previous section, are trained using SOM algorithm. On the other hand, the RBM algorithm is used for training the DRNN output layer. The proposed HDL-DRNNC structure is shown in Fig. 2.

3.1 SOM of the Kohonen learning procedure

The initializing weights values for the hidden layers of the DRNN, which is introduced in Sect. 2, are the main purpose of this section. The NN, which is used to initialize the weights of the hidden layers of the DRNN, is shown in Fig. 3. The weights of the NN are trained based on SOM of the Kohonen process [52]. The training is performed based on the hypothesis that one of the layer neurons responds most to the input, which is called the winner neuron.

For training the hidden layer (1) of the DRNN, the number of neurons in the input layer of the NN (Fig. 3), which is used to initialize the weights of the hidden layers, is equal to the number of input neurons of the hidden layer (1) of the DRNN where their values are $I_f(\kappa) = e_{xi}(\kappa)$; ($f = i = 1, \dots, n$). f denotes the number of neurons in the input layer of the NN (Fig. 3) and i denotes the number of the input neurons of the hidden layer (1) of the DRNN. The number of neurons in the output layer of the NN (Fig. 3) is equal to the number of output neurons of the hidden layer (1) of the DRNN and the weights number for the hidden layer (1) of the DRNN are equal to the weights number for the NN. After the training of NN is completed, the values of the hidden layer (1) weights of the DRNN $\mathfrak{R}_{ji}^{(1)}(\kappa)$ will be equal to the values of NN weights, $\varpi_{lf}^{(1)}(\kappa)$; ($l = j = 1, \dots, J$).

For training the hidden layer (2) of the DRNN, the number of neurons in the input layer of NN (Fig. 3) are

Fig. 1 Structure of DRNN

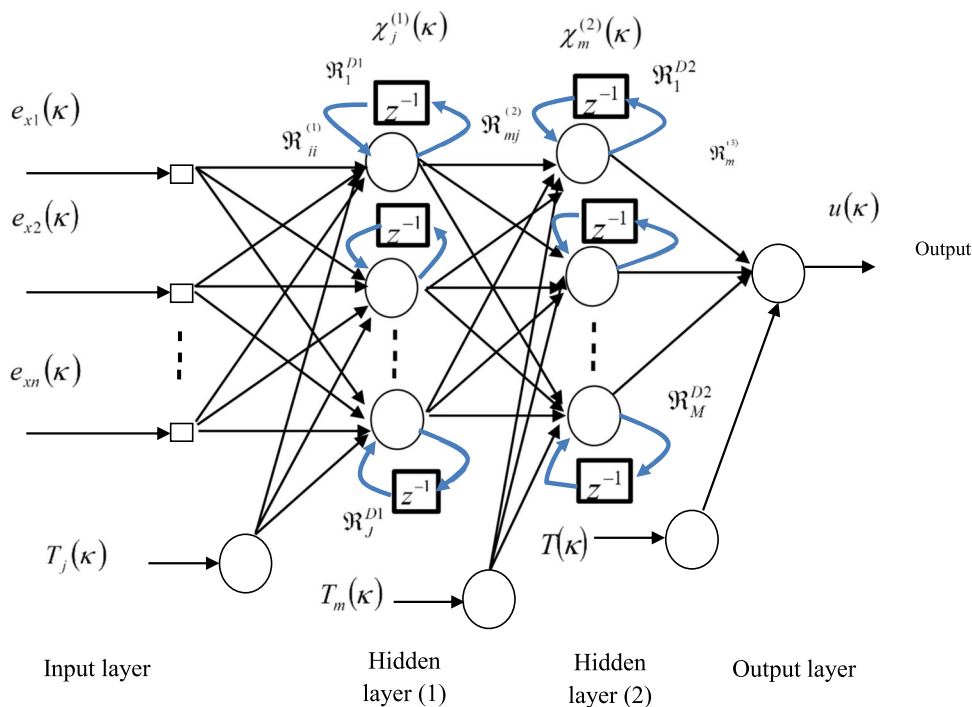


Fig. 2 Structure of HDL-DRNNC

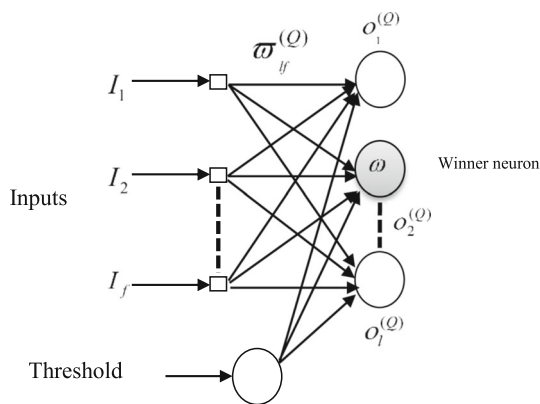
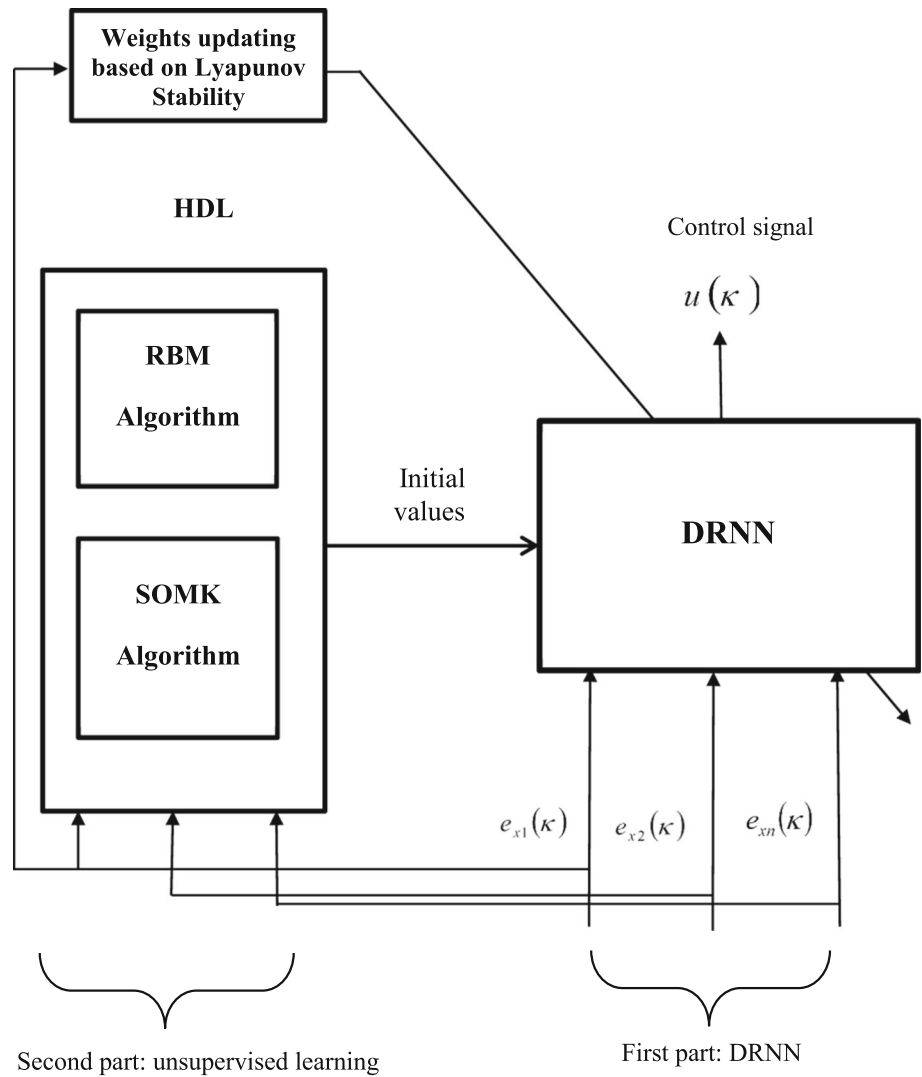


Fig. 3 NN based on SOMK unsupervised learning

equal to the number of input neurons of the hidden layer (2) of the DRNN where their values are $I_f(\kappa) = \chi_j^{(1)}(\kappa)$; ($f = j = 1, \dots, J$). The number of the input neurons of the hidden layer (2) are denoted by j , where f is defined

previously. The number of neurons in the output layer of the NN (Fig. 3) is equal to the number of output neurons of the hidden layer (2) of the DRNN and the number of weights for the hidden layer (2) of the DRNN are equal to the number of weights for the NN. After the training of NN is completed, the values of the hidden layer (2) weights of the DRNN $\mathfrak{R}_{mj}^{(2)}(\kappa)$ will be equal to the values of NN weights, $\varpi_{jf}^{(2)}(\kappa)$; ($l = m = 1, \dots, M$).

The SOM of the Kohonen (SOMK) procedure is summarized as:

- Step 1:** All the weights $\varpi_f^{(Q)}$; $Q = 1, 2$ of the NN, which is shown in Fig. 3, are initialized at zero values.
- Step 2:** Enter the values of $E(\kappa)$ to the NN.
- Step 3:** The winner neuron ω is selected using the Euclidean distance between the input and the neuron weights $\varpi_\omega^{(Q)}$ as:

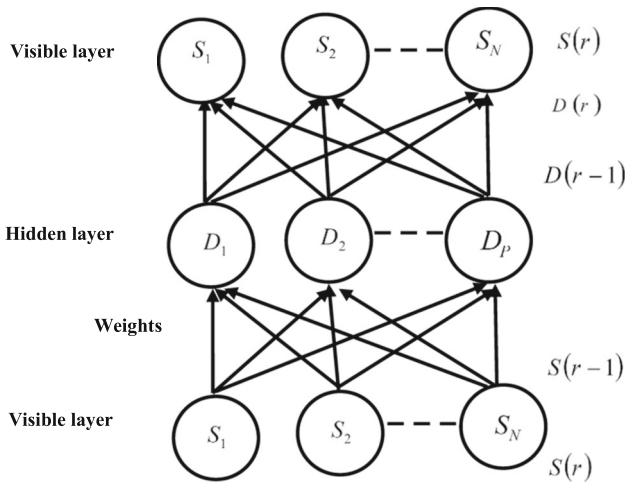


Fig. 4 Structure of RBM

$$\|e_x - \varpi_{\omega}^{(Q)}\| = \min_l \|e_x - \varpi_{\omega}^{(Q)}\| \quad (9)$$

where \$Q, l\$ and \$\omega\$ are the number of the NN layer, the index of any neuron and the index of the winner neuron, respectively.

Step 4: Calculate the Gaussian neighborhood function as:

$$\rho_{\omega l}^{(Q)}(\kappa) = \ell \exp\left(-\frac{(\omega - l)^2}{2\zeta}\right), \quad 0 < \rho_{\omega l}^{(Q)}(\kappa) \leq 1 \quad (10)$$

where \$\ell\$ and \$\zeta\$ are constants.

Step 5: The updating law of the weights of the NN \$\varpi_{lf}^{(Q)}(\kappa)\$, where \$Q = 1\$, is obtained as:

$$\Delta \varpi_{lf}^{(1)}(\kappa) = \rho_{\omega l}^{(1)}(\kappa)(e_x(\kappa) - \varpi_{lf}^{(1)}(\kappa)) \quad (11)$$

$$\varpi_{lf}^{(1)}(\kappa + 1) = \varpi_{lf}^{(1)}(\kappa) + \Delta \varpi_{lf}^{(1)}(\kappa), \quad (12)$$

(\$l = j = 1, \dots, J\$) and (\$f = i = 1, \dots, n\$)

The updating law of the weights of the NN \$\varpi_{lg}^{(Q)}(\kappa)\$, where \$Q = 2\$, is obtained as:

$$\varpi_{lf}^{(2)}(\kappa + 1) = \varpi_{lf}^{(2)}(\kappa) + \rho_{\omega l}^{(2)}(\kappa)(e_x(\kappa) - \varpi_{lf}^{(2)}(\kappa)), \quad (13)$$

(\$l = m = 1, \dots, M\$) and (\$f = j = 1, \dots, J\$)

3.2 Restricted Boltzmann machine

Initializing the weights values for the output layer of the DRNN is performed based on RBM [45, 53]. The RBM that is used in this section is illustrated in Fig. 4. Where all the weights of the output layer are equal to zero, RBM contains two main layers: firstly, the visible layer, which contains a visible nodes group \$S\$ and secondly, the hidden layers, which contains a hidden nodes group \$D\$ [46, 54].

For training the DRNN output layer, the number of neurons in the input layer of the RBM is equal to the number of input neurons of the output layer of the DRNN where their values are \$S_i(\kappa) = \chi_m^{(2)}(\kappa); (i = m = 1, \dots, M)\$. The number of neurons in the output layer of the RBM is equal to the number of neurons in the output layer of the DRNN and the weights number of the output layer of the DRNN equals to the weights number of the RBM. After the training of RBM is completed, the values of the output layer weights of the DRNN \$\mathfrak{R}_m^{(3)}(\kappa)\$ will be equal to the RBM weights values, \$X_{ji}(\kappa); (j = 1, \dots, P)\$ and (\$i = m = 1, \dots, M\$). In this paper, \$P = 1\$.

Based on the approach in [53, 55], Hinton introduced contrastive divergence (CD) for training RBM. The RBM input is \$S(r - 1)\$, which shifts to the visible layer at time (\$r - 1\$). Then, the hidden layer output is obtained as:

$$D_j(r - 1) = F\left(\sum_i X_{ji} S_i(r - 1) + B_j\right), \quad r = 1, \dots, \mathfrak{R}, \quad (14)$$

\$j = 1, \dots, P\$ and \$i = 1, \dots, N\$

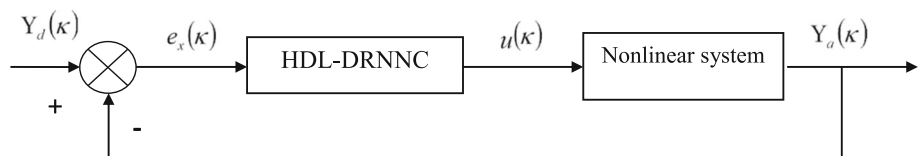
where \$X_{ji}\$ represents the weight between a visible node \$i\$ and a hidden node \$j\$ and \$S_i\$ represents the binary state of the visible node. \$P\$ and \$N\$ are the hidden nodes number and the visible nodes number, respectively. \$\mathbf{B} = [B_1 \dots B_P]^T\$ represents the hidden nodes biases and \$F\$ denotes sigmoid activation function \$F(z) = 1/(1 + \exp(-z))\$.

The inverse layer reconstructs the data from the hidden layer. As a result, \$S(r)\$ is obtained at \$r\$ as follows:

$$S_i(r) = F(Q_i(r)) = F\left(\sum_{j=1}^P X_{ij} D_j(r - 1) + A_i\right) \quad (15)$$

where \$X_{ij}\$ represents the weight between a hidden node \$j\$ and a visible node \$i\$, \$D_j\$ represents the binary state of a hidden node and \$\mathbf{A} = [A_1 \dots A_N]^T\$ represents the

Fig. 5 HDL-DRNNC block diagram with nonlinear system



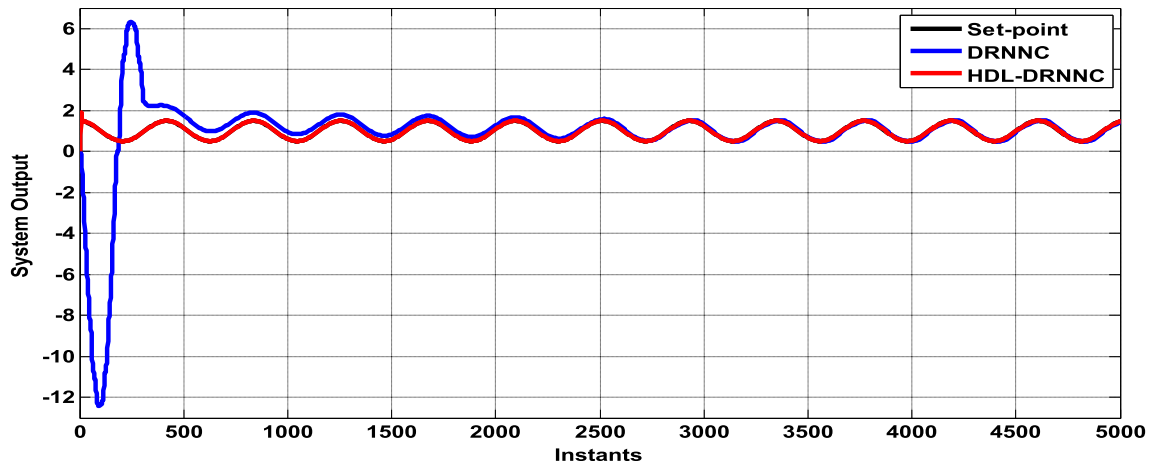


Fig. 6 Output response for the mathematical system (Task 1)

visible nodes biases. Subsequently, $S(r)$ transfers to the visible layer and the hidden layer output is obtained as:

$$D_j(r) = F(Q_j(r)) = F\left(\sum_{i=1}^N X_{ji}S_i(r) + B_j\right) \quad (16)$$

$CD - \Re$ case: The parameters learning rules for the weights and biases of nonlinear RBM are clarified as [55]:

$$X_{ji}(\kappa + 1) = X_{ji}(\kappa) + \varepsilon \left(\sum_{r=1}^{\Re} (D_j(r) - D_j(r - 1)) S_i(r) F'(Q_j(r)) + (S_i(r) - S_i(r - 1)) D_j(r - 1) F'(Q_i(r)) \right) \quad (17)$$

$$B_j(\kappa + 1) = B_j(\kappa) + \varepsilon \left(\sum_{r=1}^{\Re} (D_j(r) - D_j(r - 1)) F'(Q_j(r)) \right) \quad (18)$$

$$A_i(\kappa + 1) = A_i(\kappa) + \varepsilon \left(\sum_{r=1}^{\Re} (S_i(r) - S_i(r - 1)) F'(Q_i(r)) \right) \quad (19)$$

where ε is the RBM learning rate and κ is the iteration number. When the parameters of RBM are learned, hence the output layer of the DRNN can be initialized based on the weights of RBM $X_{ji}(\kappa + 1)$.

4 Weights updating based on Lyapunov stability

The performance function is denoted by $E_l(\kappa)$, which is defined as:

$$E_l(\kappa) = \frac{1}{2} (\Upsilon_d(\kappa) - \Upsilon_a(\kappa))^2 = \frac{1}{2} e_x^2(\kappa) \quad (20)$$

where $\Upsilon_d(\kappa)$ and $\Upsilon_a(\kappa)$ denote the reference input and the actual output, respectively. The DRNN is trained to minimize the error signal [56].

To achieve stability, the updating weights of DRNN of the proposed HDL-DRNNC are developed using Lyapunov stability criteria. Two conditions must be met in order to be asymptotically stable, as outlined in Eqs. (21 and 22)

$$R_x(\kappa) > 0 \text{ for all } \kappa \text{ except } \kappa = 0 \quad (21)$$

$$\Delta R_x(\kappa) = R_x(\kappa + 1) - R_x(\kappa) \leq 0 \quad (22)$$

where $R_x(\kappa)$ is a positive definite function. The updating weight equation can be expressed as a common form:

$$\Phi_l(\kappa + 1) = \Phi_l(\kappa) - \eta \Delta \Phi_l(\kappa) \quad (23)$$

where $\Phi_l(\kappa)$ and $\Delta \Phi_l(\kappa)$ denote a generalized weight vector and its desired modification and the learning rate is denoted by η .

Theorem 1 To achieve the stability of the controlled process, the updating equation for the DRNN weights of the proposed scheme is obtained as the following:

$$\Phi_l(\kappa + 1) = \Phi_l(\kappa) + \eta \frac{\beta \left(\Phi_l(\kappa) \left(1 + \frac{\sigma}{\beta} e_x^2(\kappa) \right) + \frac{\sigma}{\beta} e_x(\kappa) \Phi_l^2(\kappa) \frac{\partial e_x(\kappa)}{\partial \Phi_l(\kappa)} \right)}{2\lambda} \quad (24)$$

where β , σ and λ are positive constants.

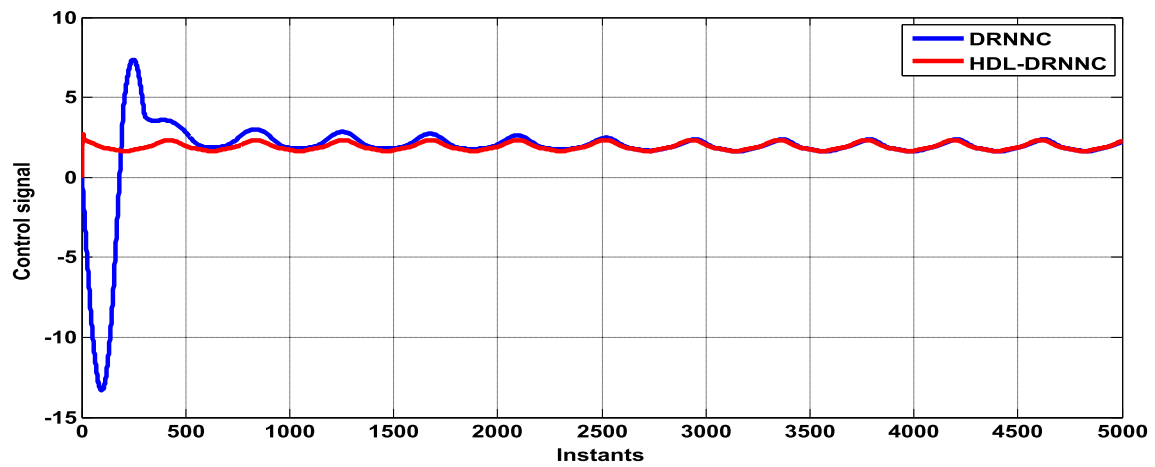


Fig. 7 Control signal (Task 1)

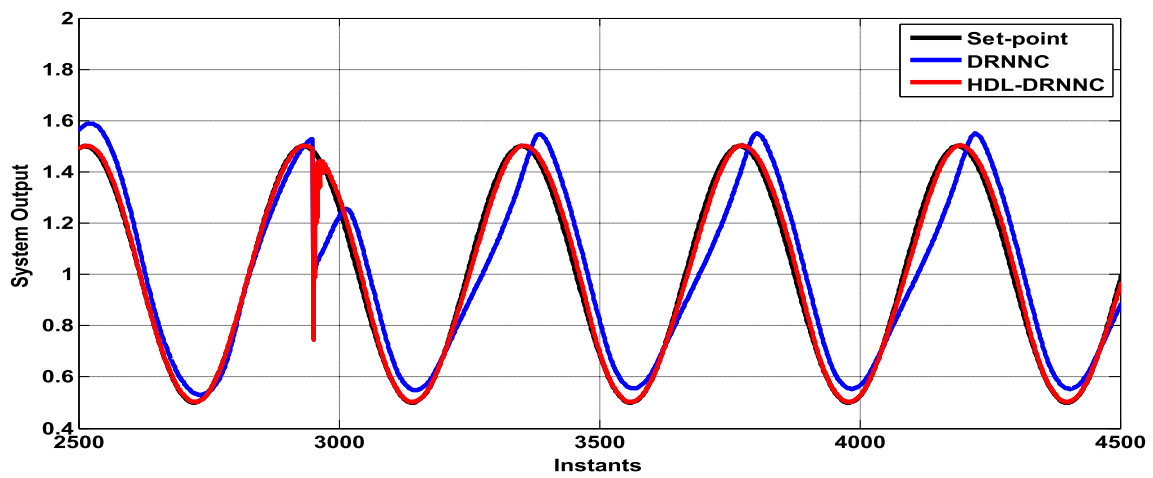


Fig. 8 Output response for the mathematical system (Task 2)

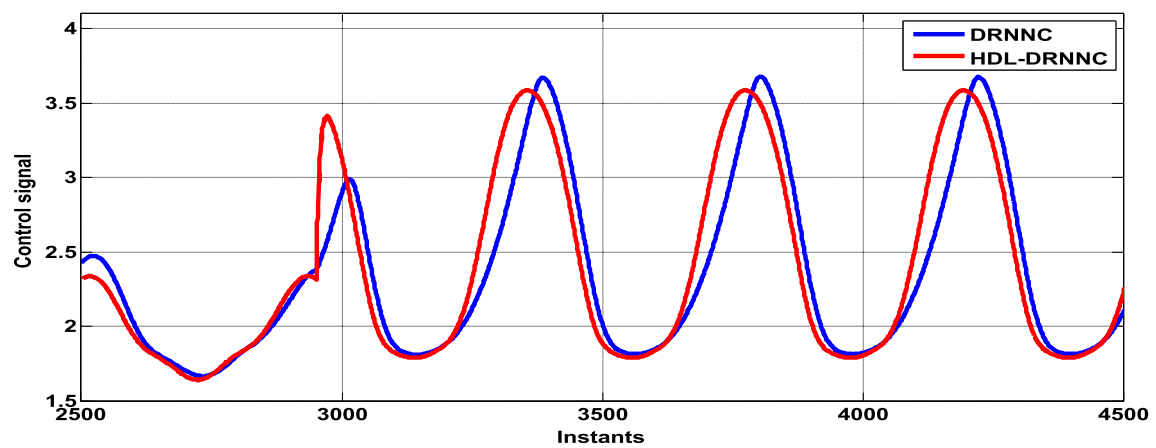


Fig. 9 Control signal (Task 2)

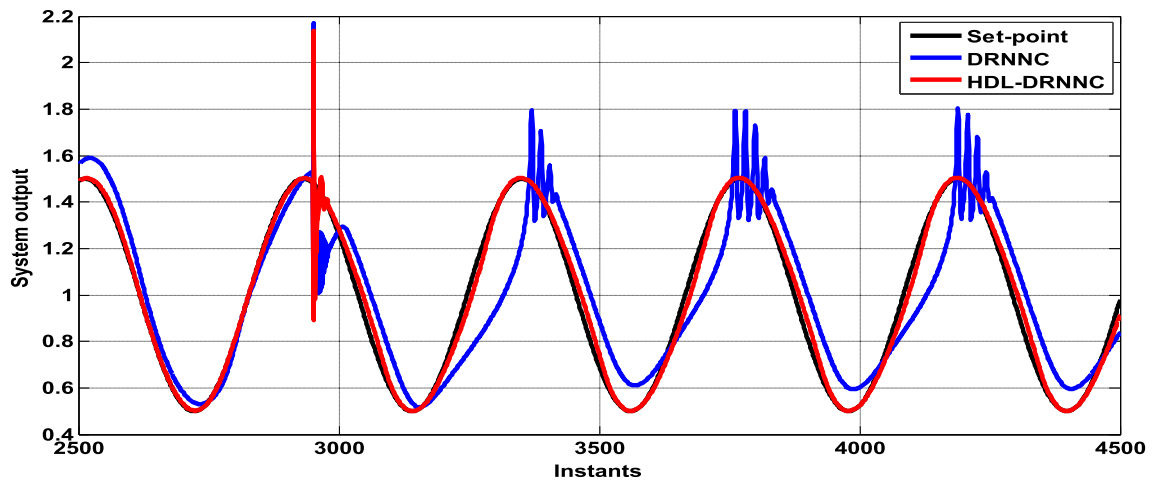


Fig. 10 Output response for the mathematical system (Task 3)

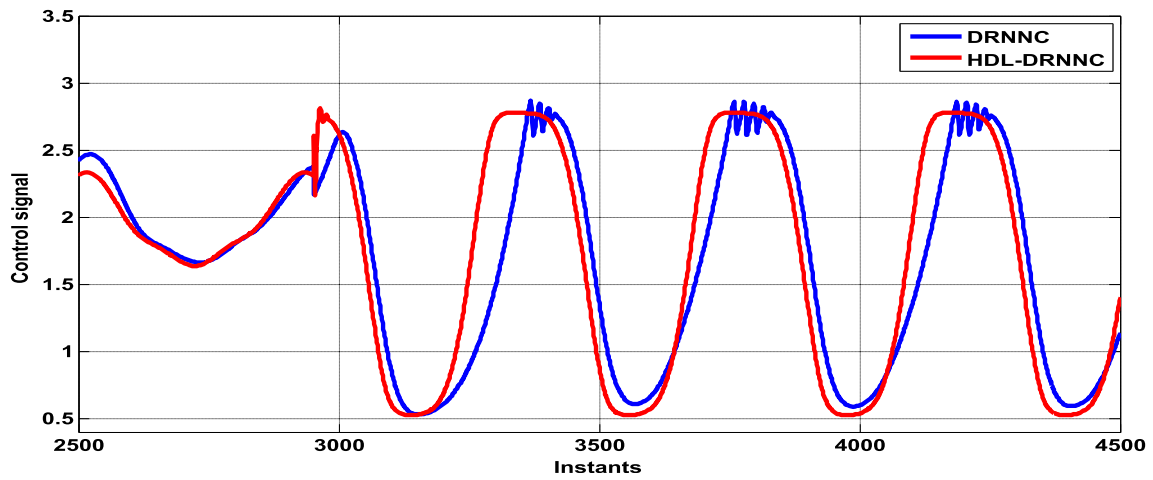


Fig. 11 Control signal (Task 3)

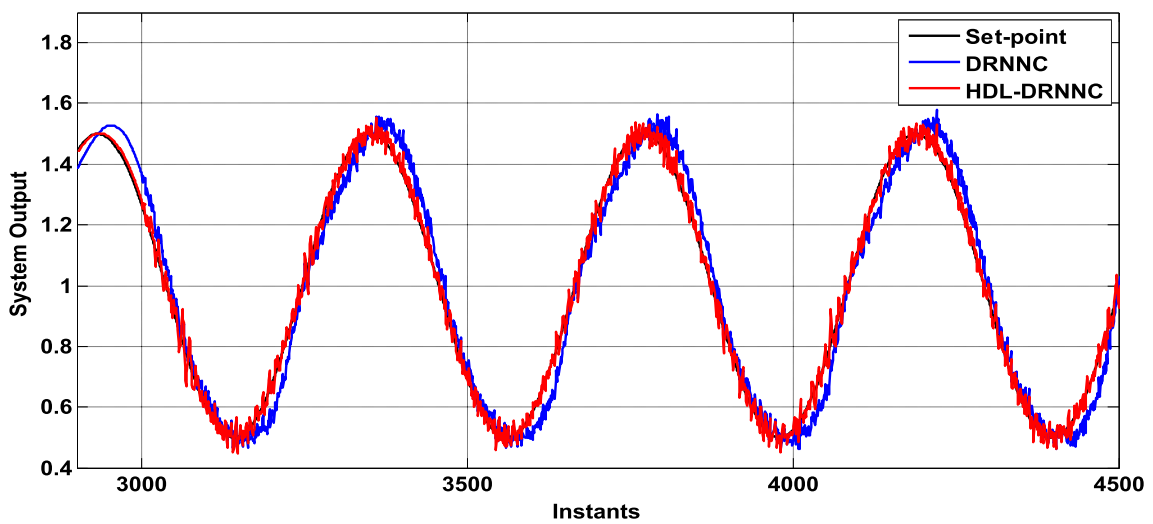


Fig. 12 Output response for the mathematical system (Task 4)

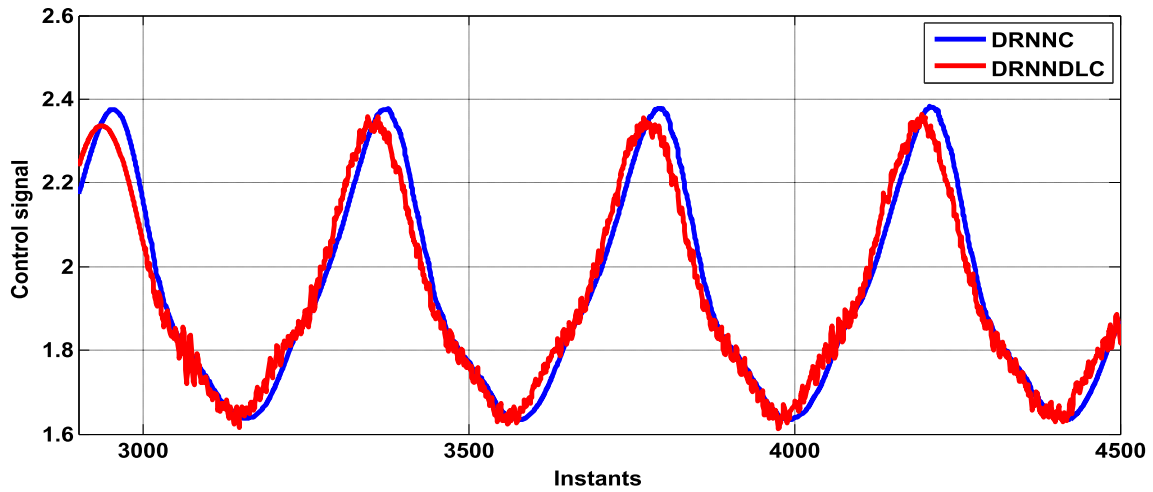


Fig. 13 Control signal (Task 4)

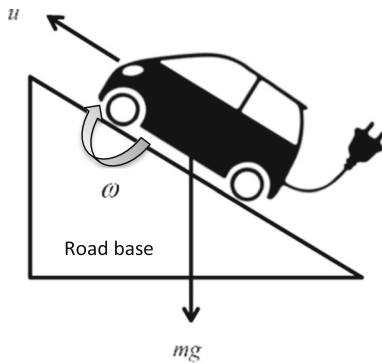


Fig. 14 EVS schematic diagram

Proof Suppose the next Lyapunov function:

$$R_x(\kappa) = R_a(\kappa) + R_b(\kappa) + R_c(\kappa) \tag{25}$$

where $R_a(\kappa) = \frac{\sigma}{2} (e_x(\kappa)\Phi_l(\kappa))^2$, $R_b(\kappa) = \frac{\beta}{2} (\Phi_l(\kappa))^2$, $R_c(\kappa) = \frac{\lambda}{2} (\Delta\Phi_l(\kappa))^2$, $\Delta R_a(\kappa)$, $\Delta R_b(\kappa)$ and $\Delta R_c(\kappa)$ are defined as:

$$\begin{aligned} \Delta R_a(\kappa) &= R_a(\kappa + 1) - R_a(\kappa) \\ &= \frac{\sigma}{2} (e_x(\kappa + 1)\Phi_l(\kappa + 1))^2 - \frac{\sigma}{2} (e_x(\kappa)\Phi_l(\kappa))^2 \end{aligned} \tag{26}$$

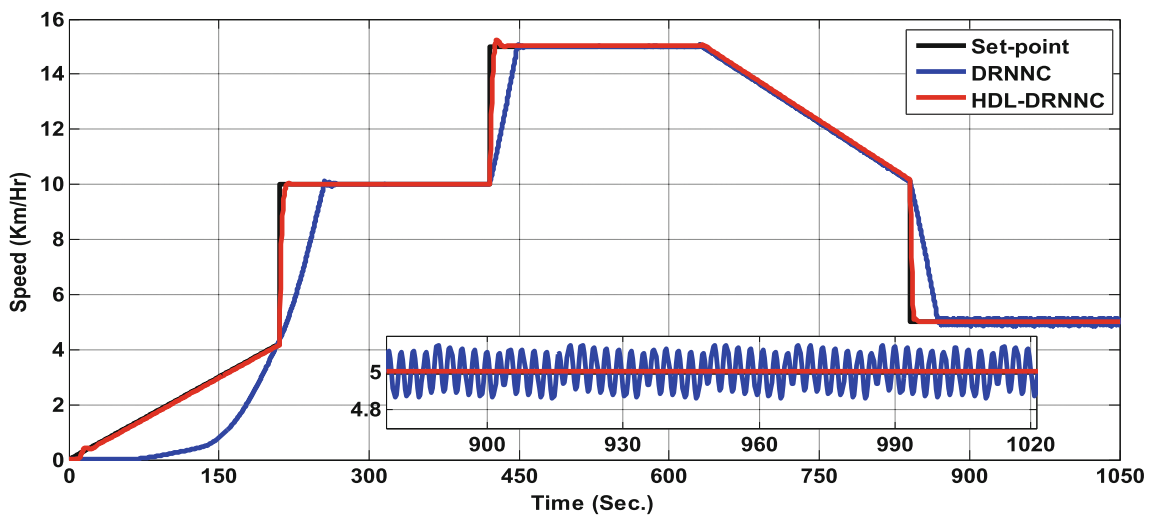


Fig. 15 Output response for the EVS (Task 1)

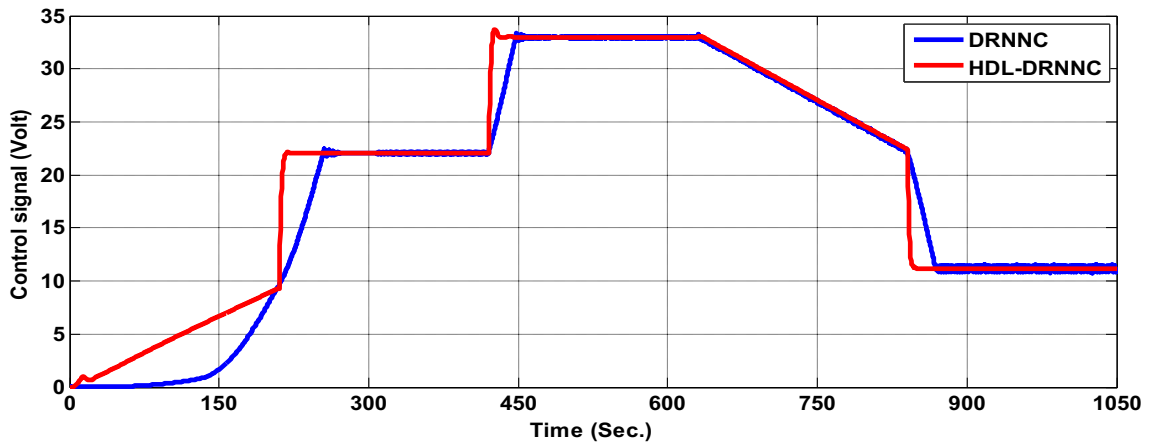


Fig. 16 The EVS control signal (Task 1)

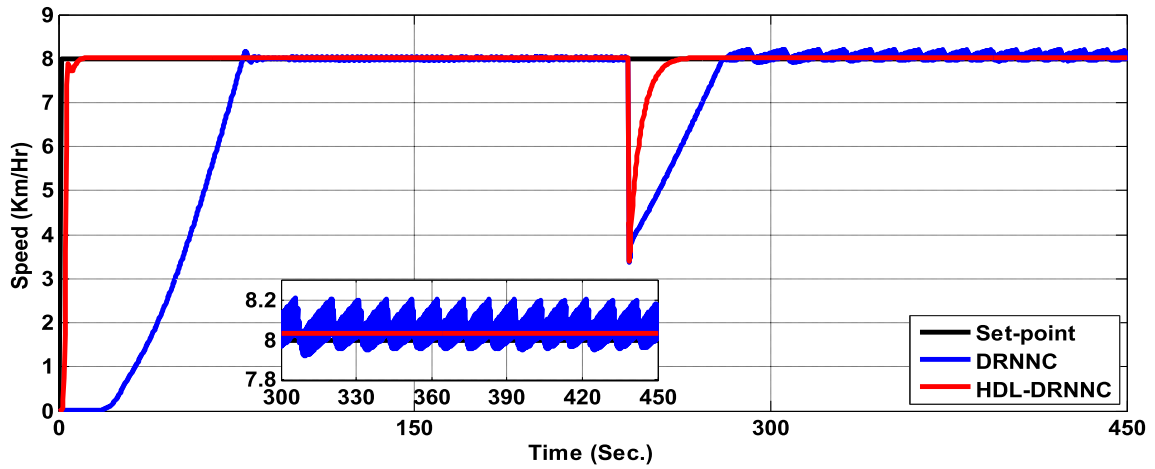


Fig. 17 Output response for the EVS (Task 2)

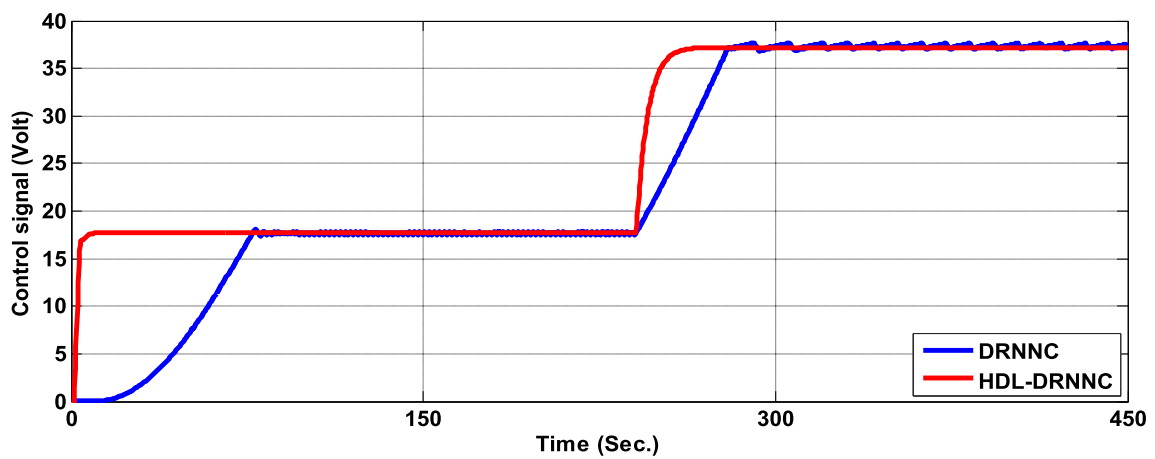


Fig. 18 The EVS control signal (Task 2)

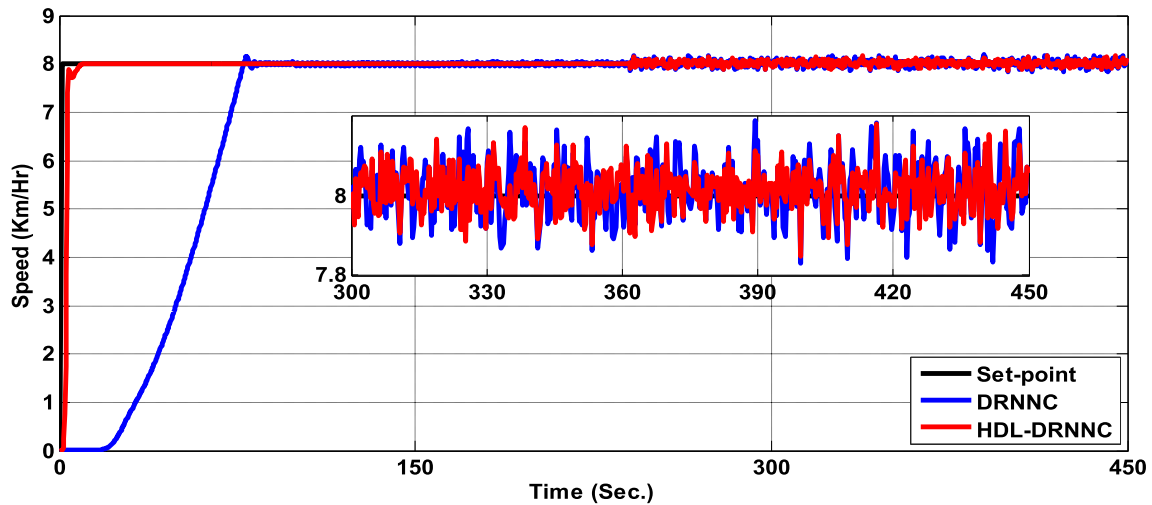


Fig. 19 Output response for the EVS (Task 3)

$$\begin{aligned} \Delta R_b(\kappa) &= R_b(\kappa + 1) - R_b(\kappa) \\ &= \frac{\beta}{2} (\Phi_I(\kappa + 1))^2 - \frac{\beta}{2} (\Phi_I(\kappa))^2 \end{aligned} \tag{27}$$

$$\begin{aligned} \Delta R_c(\kappa) &= R_c(\kappa + 1) - R_c(\kappa) \\ &= \frac{\lambda}{2} (\Delta \Phi_I(\kappa + 1))^2 - \frac{\lambda}{2} (\Delta \Phi_I(\kappa))^2 \end{aligned} \tag{28}$$

The term $\frac{\sigma}{2} (e_x(\kappa + 1) \Phi_I(\kappa + 1))^2$ can be defined based on Taylor series in the linear form as [24]:

$$\begin{aligned} \frac{\sigma}{2} (e_x(\kappa + 1) \Phi_I(\kappa + 1))^2 &= \frac{\sigma}{2} (e_x(\kappa) \Phi_I(\kappa))^2 \\ &\quad + \frac{\partial \left(\frac{\sigma}{2} (e_x(\kappa) \Phi_I(\kappa))^2 \right)}{\partial \Phi_I(\kappa)} \Delta \Phi_I(\kappa) \\ &\quad + \text{HOT} \end{aligned} \tag{29}$$

where HOT denotes to the higher order term, which can be ignored. Therefore, Eq. (29) can be rewritten as:

$$\begin{aligned} \frac{\sigma}{2} (e_x(\kappa + 1) \Phi_I(\kappa + 1))^2 - \frac{\sigma}{2} (e_x(\kappa) \Phi_I(\kappa))^2 \\ = \frac{\sigma}{2} \frac{\partial \left((e_x(\kappa) \Phi_I(\kappa))^2 \right)}{\partial \Phi_I(\kappa)} \Delta \Phi_I(\kappa) \end{aligned} \tag{30}$$

The right side of the previous equation is rewritten as follows:

$$\begin{aligned} \frac{\sigma}{2} \frac{\partial \left((e_x(\kappa) \Phi_I(\kappa))^2 \right)}{\partial \Phi_I(\kappa)} \Delta \Phi_I(\kappa) &= \sigma e_x(\kappa) \Phi_I^2(\kappa) \frac{\partial e_x(\kappa)}{\partial \Phi_I(\kappa)} \Delta \Phi_I(\kappa) \\ &\quad + \sigma e_x^2(\kappa) \Phi_I(\kappa) \Delta \Phi_I(\kappa) \end{aligned} \tag{31}$$

Similarity,

Table 1 The mathematical system MAE values

	Task 1	Task 2	Task 3	Task 4
DRNNC	0.5802	0.6603	0.6732	0.6385
FCRNNC [62]	0.5891	0.6686	0.6800	0.6471
ERNN-RBM [46]	0.2513	0.2869	0.2915	0.2813
DRNNC-SOM	0.0306	0.0518	0.0386	0.0386
FFNNHLC [61]	0.1057	0.1455	0.1537	0.1133
AIT2-TSK-FLC-RL [63]	0.07203	0.1038	0.1241	0.0728
FFNN-RBM [45]	0.04478	0.0655	0.0770	0.0475
HDL-DRNNC	0.0107	0.0164	0.0191	0.0154

$$e_x(\kappa + 1) = e_x(\kappa) + \frac{\partial e_x(\kappa)}{\partial \Phi_I(\kappa)} \Delta \Phi_I(\kappa) \tag{32}$$

Equation (32) can be rewritten as:

$$e_x(\kappa + 1) - e_x(\kappa) = \Delta e_x(\kappa) = \frac{\partial e_x(\kappa)}{\partial \Phi_I(\kappa)} \Delta \Phi_I(\kappa) \tag{33}$$

Then, by replacing the term $\frac{\partial e_x(\kappa)}{\partial \Phi_I(\kappa)} \Delta \Phi_I(\kappa)$ in Eq. (31), we obtain the following:

$$\begin{aligned} \Delta R_a(\kappa) &= \frac{\sigma}{2} (e_x(\kappa + 1) \Phi_I(\kappa + 1))^2 - \frac{\sigma}{2} (e_x(\kappa) \Phi_I(\kappa))^2 \\ &= \sigma e_x(\kappa) \Phi_I^2(\kappa) \Delta e_x(\kappa) + \sigma e_x^2(\kappa) \Phi_I(\kappa) \Delta \Phi_I(\kappa) \end{aligned} \tag{34}$$

Similarity,

$$\Delta R_b(\kappa) = \beta \Phi_I(\kappa) \Delta \Phi_I(\kappa) \text{ and } \Delta R_c = \lambda (\Delta \Phi_I(\kappa))^2.$$

The second condition for stability is determined as:

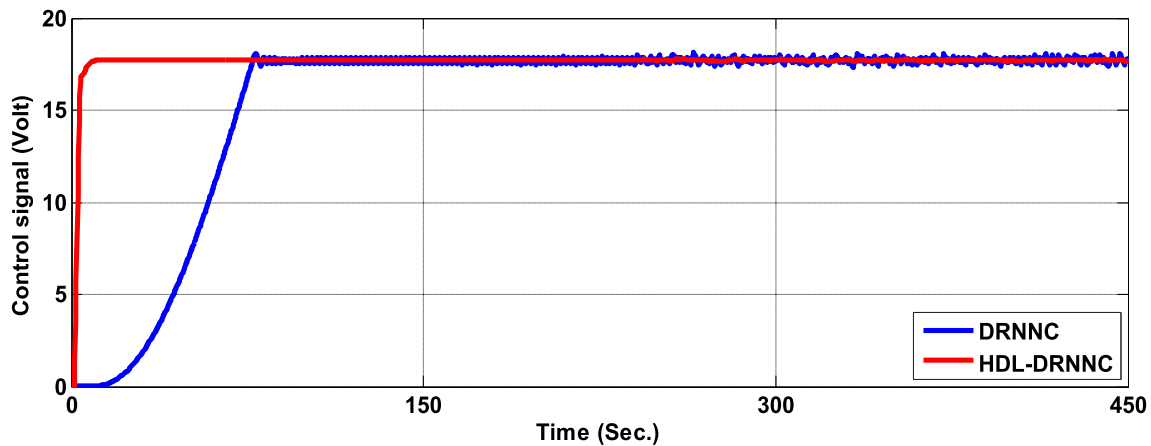


Fig. 20 The EVS control signal (Task 3)

Table 2 The mathematical system RMSE values

	Task 1	Task 2	Task 3	Task 4
DRNNC	1.94485	2.0513	2.0525	2.0500
FCRNNC [62]	1.94556	2.0521	2.0533	2.0506
ERNN-RBM [46]	0.90912	0.9589	0.9590	0.9584
DRNNC-SOM	0.17832	0.2117	0.1893	0.1885
FFNNHLC [61]	0.41366	0.4446	0.4480	0.4361
AIT2-TSK-FLC-RL [63]	0.12039	0.1595	0.1847	0.12489
FFNN-RBM [45]	0.13242	0.1528	0.1634	0.1397
HDL-DRNNC	0.03797	0.0461	0.0495	0.0427

$$\Delta R_x(\kappa) = \sigma e_x(\kappa)\Phi_l^2(\kappa)\Delta e_x(\kappa) + \sigma e_x^2(\kappa)\Phi_l(\kappa)\Delta\Phi_l(\kappa) + \beta\Phi_l(\kappa)\Delta\Phi_l(\kappa) + \lambda(\Delta\Phi_l(\kappa))^2 \leq 0 \tag{35}$$

Equation (35) can be rewritten as:

$$\Delta R_x(\kappa) = \sigma e_x(\kappa)\Phi_l^2(\kappa)\Delta e_x(\kappa) + \sigma e_x^2(\kappa)\Phi_l(\kappa)\Delta\Phi_l(\kappa) + \beta\Phi_l(\kappa)\Delta\Phi_l(\kappa) + \lambda(\Delta\Phi_l(\kappa))^2 = -\xi \tag{36}$$

where $\xi \geq 0$, so as to guarantee the condition, $\Delta R_x(\kappa) \leq 0$, the following equation is obtained:

$$\lambda(\Delta\Phi_l(\kappa))^2 + \Delta\Phi_l(\kappa) \left(\sigma e_x(\kappa)\Phi_l^2(\kappa) \frac{\Delta e_x(\kappa)}{\Delta\Phi_l(\kappa)} + \sigma e_x^2(\kappa)\Phi_l(\kappa) + \beta\Phi_l(\kappa) \right) + \xi = 0 \tag{37}$$

The general quadratic equation is determined as:

$$c\chi^2 + b\chi + a = 0 \tag{38}$$

The roots of Eq. (38) are calculated as:

$$\chi_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ca}}{2c} \tag{39}$$

From Eqs. (37) and (38), obviously, $\Delta\Phi_l(k)$ acts as χ in Eq. (38) and the values of c , b and a in Eq. (37) are obtained as:

$$c = \lambda, \quad b = \beta \left(\frac{\sigma}{\beta} e_x(\kappa)\Phi_l^2(\kappa) \frac{\Delta e_x(\kappa)}{\Delta\Phi_l(\kappa)} + \left(1 + \frac{\sigma}{\beta} e_x^2(\kappa) \right) \Phi_l(\kappa) \right) \text{ and } a = \xi \tag{40}$$

There is a single unique solution for Eq. (38), if $\sqrt{b^2 - 4ca} = 0$. Therefore,

$$\left(\beta \left(\frac{\sigma}{\beta} e_x(\kappa)\Phi_l^2(\kappa) \frac{\Delta e_x(\kappa)}{\Delta\Phi_l(\kappa)} + \left(1 + \frac{\sigma}{\beta} e_x^2(\kappa) \right) \Phi_l(\kappa) \right) \right)^2 - 4\lambda\xi = 0 \tag{41}$$

Table 3 EVS parameters

Symbol and abbreviation	Value	Symbol and abbreviation	Value
m (kg)	800	A (m ²)	1.8
Γ_{af} (mH)	1.776	ρ (kg/m ³)	1.25
J (kgm ²)	0.05	C_d	0.3
r_e (m)	0.25	μ_{rr}	0.015
$\Gamma_a + \Gamma_f$ (mH)	6.008	G	11
$\Re_a + \Re_f$ (Ω)	0.2	B (Nm/s)	0.0002

Table 4 EVS parameters variation values

Symbol and abbreviation	Value
m	900
C_d	0.2
J	0.04
r_e	0.3
$\Gamma_a + \Gamma_f$	4.008
$R_a + R_f$	0.25
μ_{rr}	0.025

and, ξ can be calculated as:

$$\xi = \frac{\left(\beta \left(\frac{\sigma}{\beta} e_x(\kappa) \Phi_l^2(\kappa) \frac{\Delta e_x(\kappa)}{\Delta \Phi_l(\kappa)} + \left(1 + \frac{\sigma}{\beta} e_x^2(\kappa) \right) \Phi_l(\kappa) \right) \right)^2}{4\lambda} \tag{42}$$

Since $\xi \geq 0$, which means

$$\frac{\left(\beta \left(\frac{\sigma}{\beta} e_x(\kappa) \Phi_l^2(\kappa) \frac{\Delta e_x(\kappa)}{\Delta \Phi_l(\kappa)} + \left(1 + \frac{\sigma}{\beta} e_x^2(\kappa) \right) \Phi_l(\kappa) \right) \right)^2}{4\lambda} \geq 0 \tag{43}$$

So, the unique root of Eq. (37) is $\chi_{1,2} = \frac{-b}{2c}$, similarly,

$$\Delta \Phi_l(\kappa) = - \frac{\beta \left(\Phi_l(\kappa) \left(1 + \frac{\sigma}{\beta} e_x^2(\kappa) \right) + \frac{\sigma}{\beta} e_x(\kappa) \Phi_l^2(\kappa) \frac{\partial e_x(\kappa)}{\partial \Phi_l(\kappa)} \right)}{2\lambda} \tag{44}$$

Equation (44) can be reformulated as:

$$\Delta \Phi_l(\kappa) = - \frac{\beta \left(\frac{\sigma}{\beta} e_x(\kappa) \Phi_l^2(\kappa) \frac{\partial e_x(\kappa)}{\partial \Phi_l(\kappa)} + \left(1 + \frac{\sigma}{\beta} e_x^2(\kappa) \right) \Phi_l(\kappa) \right)}{2\lambda} \tag{45}$$

So, by replacing $\Delta \Phi_l(\kappa)$ in Eq. (23), the updating equation for the parameters of the DRNN of the HDL-DRNNC can be given as in Eq. (24).

5 Steps of the proposed HDL-DRNNC

The system block diagram with the proposed HDL-DRNNC is shown in Fig. 5. The error signal $e_x(\kappa)$ is the difference between the reference input $Y_d(\kappa)$ and the actual output of the nonlinear system $Y_a(\kappa)$. The proposed controller input is $e_x(\kappa)$ and its output is the control signal $u(\kappa)$, which forward to the nonlinear system.

As shown in Figs. 1 and 2, the first layer of the HDL-DRNNC contains three inputs, which are the error signal $e_{x1}(\kappa) = e_x(\kappa)$, the change of error signal $e_{x2}(\kappa) = e_x(\kappa) - e_x(\kappa - 1)$ and the change of the change of error signal

Table 5 MAE values for the EVS

	Task 1	Task 2	Task 3
DRNNC	0.56730	1.17533	0.97630
FCRNNC [62]	0.50638	1.51069	1.31162
ERNN-RBM [46]	0.45990	0.95267	0.74690
DRNNC-SOM	0.43245	0.40114	0.16700
FFNNHLC [61]	0.47226	1.04951	0.86402
AIT2-TSK-FLC-RL [63]	0.28022	0.39834	0.29394
FFNN-RBM [45]	0.22735	0.21747	0.14997
HDL-DRNNC	0.07896	0.11124	0.06934

Table 6 RMSE values for the EVS

	Task 1	Task 2	Task 3
DRNNC	1.23253	2.61144	2.50004
FCRNNC [61]	1.16820	2.51704	2.40280
ERNN-RBM [46]	1.13236	2.05436	1.96370
DRNNC-SOM	1.08478	0.85328	0.6678
FFNNHLC [62]	1.14823	2.44867	2.33094
AIT2-TSK-FLC-RL [63]	0.73576	1.24810	1.12558
FFNN-RBM [45]	0.66215	1.03518	0.89828
HDL-DRNNC	0.39100	0.63858	0.53724

$e_{x3}(\kappa) = e_x(\kappa) - 2e_x(\kappa - 1) + e_x(\kappa - 2)$. The output layer contains one output $u(\kappa)$. Algorithm 1 summarizes the proposed HDL-DRNNC pseudocode for reader’s convenience.

6 Simulation results

A comparison of the proposed HDL-DRNNC and DRNNC is performed under the same conditions with zero initial weights to show the performance of the hybrid learning algorithm. In the present paper, assign $J = M = l = 10$, $n = 3$, $R = 10$, $N = 10$, $\Re = 1$ and $P = 1$. In order to evaluate the performance and demonstrate the robustness of the proposed controller, the mean absolute error (MAE) and the root-mean-square error (RMSE) are used. MAE and RMSE are clarified as [57, 58]:

$$MAE = \frac{1}{K_L} \sum_{\kappa=1}^{K_L} |e_x(\kappa)| \tag{46}$$

$$RMSE = \sqrt{\left(\frac{1}{K_L} \sum_{\kappa=1}^{K_L} (e_x(\kappa))^2 \right)} \tag{47}$$

where K_L denotes iterations number.

Algorithm 1: The proposed HDL-DRNNC Pseudo Code.

All weights of HDL-DRNNC = 0.

Initialization: $\kappa = 1$.

while $\kappa \leq K_L$ **do**

 Receive desired input; $Y_d(\kappa)$.

For $Q = 1 : 2$

If ($Q = 1$)

$I_f(\kappa) = e_{xi}(\kappa)$; ($f = i = 1, \dots, n$), ($l = j = 1, \dots, J$).

 Select the number of weights; $\varpi_{lf}^{(1)}(\kappa)$ = the number of weights; $\mathfrak{R}_{ji}^{(1)}(\kappa)$.

 Update the NN weights parameters; $\varpi_{lf}^{(1)}(\kappa)$ based on the SOMK as in Eq. (12).

 Set the weights values; $\mathfrak{R}_{ji}^{(1)}(\kappa) = \varpi_{lf}^{(1)}(\kappa)$.

 Compute the hidden layer (1) outputs; $\chi_j^{(1)}(\kappa)$; based on Eq. (3).

else

$I_f(\kappa) = \chi_j^{(1)}(\kappa)$; ($f = j = 1, \dots, J$) and ($l = m = 1, \dots, M$).

 Select the number of weights; $\varpi_{lf}^{(2)}(\kappa)$ = the number of weights; $\mathfrak{R}_{mj}^{(2)}(\kappa)$.

 Update the NN weights parameters; $\varpi_{lf}^{(2)}(\kappa)$ based on the SOMK as in Eq. (13).

 Set the weights values; $\mathfrak{R}_{mj}^{(2)}(\kappa) = \varpi_{lf}^{(2)}(\kappa)$.

 Compute the hidden layer (2) outputs; $\chi_m^{(2)}(\kappa)$ based on Eq. (6).

End If

End For

The RBM inputs are equal the inputs of the output layer of the DRNN where,

$S_i(\kappa) = \chi_m^{(2)}(\kappa)$; ($i = m = 1, \dots, M$).

Select the number of weights; $\mathfrak{R}_m^{(3)}(\kappa)$ = the number of weights; $X_{ji}(\kappa)$, ($j = 1, \dots, P$) and ($i = m = 1, \dots, M$).

Update the RBMs parameters based on Eqs. (17), (18) and (19).

Set the weights values; $\mathfrak{R}_m^{(3)}(\kappa) = X_i(\kappa)$.

Calculate; $u(\kappa)$.

Update the DRNN weights parameters; $\mathfrak{R}_{ji}^{(1)}(\kappa)$, $\mathfrak{R}_{mj}^{(2)}(\kappa)$ and $\mathfrak{R}_m^{(3)}(\kappa)$ according to the Lyapunov stability formula; Eq. (24).

Set the weights values; $\varpi_{lf}^{(1)}(\kappa) = \mathfrak{R}_{ji}^{(1)}(\kappa)$, $\varpi_{lf}^{(2)}(\kappa) = \mathfrak{R}_{mj}^{(2)}(\kappa)$, $X_i(\kappa) = \mathfrak{R}_m^{(3)}(\kappa)$

Measure; $Y_d(\kappa)$.

End while

6.1 Case1: mathematical system

A non-affine nonlinear system is used to test the performance of the proposed controller, which is given as [59, 60]:

$$G_1(\kappa + 1) = \alpha_1(\kappa) G_2(\kappa) + \alpha_2(\kappa) \sin(G_1(\kappa)) \quad (48)$$

$$G_2(\kappa + 1) = \alpha_3(\kappa) \cos(G_2(\kappa)) \sin(G_1(\kappa)) + \alpha_4(\kappa) u(\kappa) + \alpha_5(\kappa) \tanh(u(\kappa)) \quad (49)$$

$$Y_d(\kappa + 1) = G_1(\kappa + 1) \quad (50)$$

where $\alpha_1(\kappa) = 0.5$, $\alpha_2(\kappa) = -0.3$, $\alpha_3(\kappa) = -1$, $\alpha_4(\kappa) = 2$ and $\alpha_5(\kappa) = -2$.

6.1.1 Task 1: tracking the reference signal trajectory

The reference signal in this task is defined as:

$$Y_d(\kappa) = 1 + 0.5 \sin(0.05\kappa T) \quad (51)$$

where T denotes the sampling period.

Figures 6 and 7 exhibit the system response and the control signal for the proposed HDL-DRNNC and DRNNC. It is clear that there is an error between the set-point and the system output under using the DRNNC at the beginning of simulation task. However, the proposed controller using hybrid learning algorithm based on SOM and RBM is able to track the set-point without a steady-state error.

6.1.2 Task 2: uncertainty due to disturbance

To evaluate the robustness of the proposed HDL-DRNNC, a disturbance value of 50% of its desired output is added to the system output at $\kappa = 2950$. Figure 8 illustrates that the system output tracks the set-point without a steady-state error after adding a disturbance value to the measured output for the proposed HDL-DRNNC. After adding a disturbance value for the DRNNC, there is still a steady-state error. Figure 9 exhibits the control signal of the system for both controllers. In contrast with the DRNNC, the proposed HDL-DRNNC clearly responds to disturbance effects.

6.1.3 Task 3: Uncertainty due to disturbance with parameters variation

At $\kappa = 2950$ instant and after the system output reaches the reference input, the system parameters are varied as follows: $\alpha_1(\kappa) = 0.35$, $\alpha_2(\kappa) = -0.35$, $\alpha_3(\kappa) = -1.5$, $\alpha_4(\kappa) = 2.5$ and $\alpha_5(\kappa) = 1$ with an effect 40% disturbance. The output response and the control signal of the system are shown in Figs. 10 and 11, respectively, for both

controllers. It is evident that the proposed HDL-DRNNC is capable of responding to the uncertainty effects (the parameters variation and disturbance) as compared to the DRNNC.

6.1.4 Task 4: uncertainty due to noise

A random noise signal is added at $\kappa = 2950$ instant. The output response and the control signal of the system are shown in Figs. 12 and 13, respectively. In contrast to the output response based on DRNNC, the proposed HDL-DRNNC has the ability to recover from the impact of random noise more quickly. The proposed HDL-DRNNC is more robust than the DRNNC.

Tables 1 and 2 illustrate the values of MAE and RMSE for the proposed HDL-DRNNC, the DRNNC and other schemes, that are published previously such as feed-forward neural network based on RBM (FFNN-RBM) [45], ERNN-RBM [46], feed-forward neural network with hybrid learning controller (FFNNHLC) [61], FCRNNC [62] and adaptive interval type-2 Takagi–Sugeno–Kang fuzzy logic controller based on reinforcement learning (AIT2-TSK-FLC-RL) [63]. On the other hand, the proposed HDL-DRNNC is compared with DRNNC based on SOM (DRNNC-SOM) to show the benefits of hybrid learning.

The MAE and RMSE values for the proposed HDL-DRNNC scheme are clearly smaller than those obtained for other schemes. Due to use of DL to initialize the weight values of the proposed HDL-DRNNC, it can reduce the impact of the system uncertainties caused by external disturbance, parameter variations, and random noise compared with other schemes. The hybrid algorithm is used because it gave better results than the DRNNC-based SOM algorithm as shown in Tables 1 and 2.

6.2 Case 2: physical system

In this section, the proposed controller is used for controlling a physical system, which is the electrical vehicle system (EVS). Nowadays, EVSs are increasingly advancing because of the importance of environmental protection and lack of energy sources [64]. The control of EVSs is important role in order to determine a high-performance EVS with an optimal balance of travelling range per charge, maximum speed and acceleration performance [64]. EVSs are basically time-variant (e.g. the EVS parameters and the road condition are consistently varying) nonlinear system, which making the control of an EVS quite cumbersome [64]. Therefore, the control of EVS should be designed robustly and adaptively to improve the system in both dynamic and steady performance states.

Figure 14 shows the schematic diagram of an EVS and the mathematical model is given as [63–66]:

$$\begin{aligned} \dot{\bar{h}} &= V(\bar{h}) + q(\bar{h})u, \\ \Upsilon_a &= Z(\bar{h}) \end{aligned} \tag{52}$$

where $\bar{h} = \begin{bmatrix} i \\ \omega \end{bmatrix} = \begin{bmatrix} \bar{h}_1 \\ \bar{h}_2 \end{bmatrix}$, $V(\bar{h})$, $q(\bar{h})$ and $Z(\bar{h})$ are defined as:

$$V(\bar{h}) = \begin{bmatrix} \left(\frac{-1}{\Gamma_a + \Gamma_f}\right) C_1 \\ \left(\frac{1}{J + m \left(\frac{r_e}{G}\right)^2}\right) \left(\Gamma_{af} \bar{h}_1^2 - B \bar{h}_2 - \left(\frac{r_e}{G}\right) C_2\right) \end{bmatrix} \tag{53}$$

$$q(\bar{h}) = \begin{bmatrix} \left(\frac{1}{\Gamma_a + \Gamma_f}\right) \\ 0 \end{bmatrix} \text{ and } Z(\bar{h}) = \bar{h}_2 \tag{54}$$

$$C_1 = (\mathfrak{R}_a + \mathfrak{R}_f) \bar{h}_1 + \Gamma_{af} \bar{h}_1 \bar{h}_2 \tag{55}$$

$$C_2 = 0.5 \rho A C_d \left(\frac{r_e}{G}\right)^2 \bar{h}_2^2 + \mu_{rr} m g \tanh(\bar{h}_2) \tag{56}$$

where ω and i denote to angular speed and angular speed of the motor. Γ_f , Γ_a , \mathfrak{R}_f and \mathfrak{R}_a denote the field inductance, armature inductance, the field resistance and armature resistance, respectively. J denotes the inertia of the motor, r_e denotes the tire radius of the EVS, which includes the tires with gearing system, Γ_{af} is the mutual inductance between the armature and the field windings, and ρ , A and m denote the air density, the frontal area of the vehicle and the mass of the EVS, respectively. μ_{rr} , B , G and C_d denote the rolling resistance coefficient, the viscous coefficient, the gearing ratio and the drag coefficient, respectively. The values of EVS parameters are listed in Table 3.

6.2.1 Task 1: tracking the reference signal trajectory

Figures 15 and 16 exhibit the EVS response and its control signal when the desired input is given as:

$$\Upsilon_d = \begin{cases} 0.02t & 0 < t \leq 210 \\ 10 & 210 < t \leq 420 \\ 15 & 420 < t \leq 630 \\ -0.0237t + 30 & 630 < t \leq 840 \\ 2 & 840 < t \leq 1050 \end{cases} \tag{57}$$

In this task, the set-point changing is carried for testing the proposed HDL-DRNNC, which is compared with the DRNNC. It is clear that the EVS response using the proposed HDL-DRNNC reaches the set-point faster than the DRNNC.

6.2.2 Task 2: uncertainty due to parameters variation with disturbance

In this task, the EVS parameters are varied as in Table 4 with an effect 40% disturbance after the system output reaches the reference input at $t = 240$ sec. The system response and its control signal for both controllers are exhibited as in Figs. 17 and 18. It is clear that the robustness of the proposed HDL-DRNNC is better than the DRNNC due to its ability of reducing the effect of system uncertainties.

6.2.3 Task 3: uncertainty due to random noise

A random noise signal is added at $t = 240$ sec. The EVS response and its control signal are shown in Figs. 19 and 20. It is clear that the EVS response for the proposed HDL-DRNNC is quickly recovering from the impact of random noise as compared to the output response based on DRNNC. The robustness of the proposed HDL-DRNNC is better than that compared with DRNNC.

The analyses of the MAE and RMSE values for the proposed HDL-DRNNC, the DRNNC and other schemes are presented in Tables 5 and 6. It is clear that the MAE and RMSE values for the proposed HDL-DRNNC are smaller than those obtained for other schemes. Compared with other schemes, HDL-DRNNC has the ability to reduce the impact of system uncertainties.

The main features of the proposed HDL-DRNNC are gathered as follows: (1) It has a swift learning control due to its use of hybrid DL, which uses SOM and RBM to initialize the weights values, (2) the controller is stable as it uses the Lyapunov stability method to update the weight values and it guarantees the stability, and (3) it is successful for reducing the system uncertainties and tracking the performance output for both mathematical system and physical system.

7 Conclusion

In the present paper, the HDL-DRNNC is proposed for nonlinear systems. The HDL-DRNNC uses the DRNN, which can be learned from HDL. In order to guarantee the stability of the proposed controller, the updating weights of the DRNN are derived using the Lyapunov stability criterion. Two nonlinear systems, namely mathematical and physical, are used to estimate the performance of the proposed controller. According to the obtained results, the proposed HDL-DRNNC can overcome uncertainty and track the performance of the controlled systems. By comparing MAE and RMSE indicators, it is evident that the response of mathematical and physical systems based on

HDL-DRNNC is able to recover fast from the effects of uncertainties as compared with the response of mathematical and physical systems based on DRNNC and other existing controllers. As conclusion, HDL-DRNNC robustness has superior performance and a faster ability to recover from uncertainty as compared to DRNNC and other controllers. In the future work, the authors will try to implement practically the proposed algorithm using microcontrollers for controlling a real system.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations

Conflict of interest There is no conflict of interest between the authors to publish this manuscript.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Kang J, Meng W, Abraham A, Liu H (2014) An adaptive PID neural network for complex nonlinear system control. *Neurocomputing* 135:79–85. <https://doi.org/10.1016/j.neucom.2013.03.065>
- Zhao C, Guo L (2017) On the capability of PID control for nonlinear uncertain systems. *IFAC-Papers On Line* 50:1521–1526. <https://doi.org/10.1016/j.ifacol.2017.08.302>
- Cetin M, Iplikci S (2015) A novel auto-tuning PID control mechanism for nonlinear systems. *ISA Trans* 58:292–308. <https://doi.org/10.1016/j.isatra.2015.05.017>
- Roman RC, Precup RE, Petriu EM (2021) Hybrid data-driven fuzzy active disturbance rejection control for tower crane systems. *Eur J Control* 58:373–387. <https://doi.org/10.1016/j.ejcon.2020.08.001>
- Karanayil B, Rahman MF (2011) Artificial neural network applications in power electronics and electrical drives. *Power electronics handbook*. Elsevier, pp 1139–1154. <https://doi.org/10.1016/B978-0-12-382036-5.00038-0>
- Ma Y, Niu P, Zhang X, Li G (2017) Research and application of quantum-inspired double parallel feed-forward neural network. *Knowl-Based Syst* 136:140–149. <https://doi.org/10.1016/j.knsys.2017.09.013>
- Adhikari SP, Yang C, Slot K, Strzelecki M, Kim H (2018) Hybrid no-propagation learning for multilayer neural networks. *Neurocomputing* 321:28–35. <https://doi.org/10.1016/j.neucom.2018.08.034>
- Fourati F, Chtourou M (2007) A greenhouse control with feed-forward and recurrent neural networks. *Simul Model Pract Theory* 15:1016–1028. <https://doi.org/10.1016/j.simpat.2007.06.001>
- Kouziokas GN, Chatzigeorgiou A, Perakis K (2018) Multilayer feed forward models in groundwater level forecasting using meteorological data in public management. *Water Resour Manag* 32:5041–5052. <https://doi.org/10.1007/s11269-018-2126-y>
- Shafiq MA (2016) Direct adaptive inverse control of nonlinear plants using neural networks. In: *Future technologies conference (FTC)*. IEEE, pp 827–830
- Wu Z, Rincon D, Christofides PD (2020) Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. *J Process Control* 89:74–84. <https://doi.org/10.1016/j.jprocont.2020.03.013>
- Wang T, Gao H, Qiu J (2015) A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. *IEEE Trans Neural Netw Learn Syst* 27:416–425. <https://doi.org/10.1109/TNNLS.2015.2411671>
- Yan Z, Wang J (2012) Model predictive control of nonlinear systems with unmodeled dynamics based on feedforward and recurrent neural networks. *IEEE Trans Industr Inf* 8:746–756. <https://doi.org/10.1109/TII.2012.2205582>
- Hang S, Yingbai H, Karimi HR, Knoll A, Ferrigno G, De Momi E (2020) Improved recurrent neural network-based manipulator control with remote center of motion constraints: experimental results. *Neural Netw* 131:291–299. <https://doi.org/10.1016/j.neunet.2020.07.033>
- Li DJ, Li DP (2015) Adaptive controller design-based neural networks for output constraint continuous stirred tank reactor. *Neurocomputing* 153:159–163. <https://doi.org/10.1016/j.neucom.2014.11.041>
- Kumar R, Srivastava S, Gupta JRP, Mohindru A (2019) Comparative study of neural networks for dynamic nonlinear systems identification. *Soft Comput* 23:101–114. <https://doi.org/10.1007/s00500-018-3235-5>
- Wei Q, Liu D (2015) Neural-network-based adaptive optimal tracking control scheme for discrete-time nonlinear systems with approximation errors. *Neurocomputing* 149:106–115. <https://doi.org/10.1016/j.neucom.2013.09.069>
- Fei J, Wang H (2020) Recurrent neural network fractional-order sliding mode control of dynamic systems. *J Franklin Inst* 357:4574–4591. <https://doi.org/10.1016/j.jfranklin.2020.01.050>
- Li F, Zurada JM, Liu Y, Wei W (2017) Input Layer regularization of multilayer feedforward neural networks. *IEEE Access* 5:10979–10985. <https://doi.org/10.1109/ACCESS.2017.2713389>
- Akpan VA, Hassapis GD (2011) Nonlinear model identification and adaptive model predictive control using neural networks. *ISA Trans* 50:177–194. <https://doi.org/10.1016/j.isatra.2010.12.007>
- Kolen JF, Kremer SC (2001) *Gradient calculations for dynamic recurrent neural networks*. Wiley-IEEE Press, London. <https://doi.org/10.1109/9780470544037.ch11>
- Mikolov T, Zweig G (2012) Context dependent recurrent neural network language model. In: *2012 spoken language technology workshop (SLT)*. IEEE, pp 234–239. <https://doi.org/10.1109/SLT.2012.6424228>
- Song Q (2010) On the weight convergence of Elman networks. *IEEE Trans Neural Netw* 21(3):463–480. <https://doi.org/10.1109/TNN.2009.2039226>
- Asgharnia A, Jamali A, Shahnazi R, Maheri A (2020) Load mitigation of a class of 5-MW wind turbine with RBF neural network based fractional-order PID controller. *ISA Trans* 96:272–286. <https://doi.org/10.1016/j.isatra.2017.01.022>

25. Ku CC, Lee KY (1995) Diagonal recurrent neural networks for dynamic systems control. *IEEE Trans Neural Netw* 6:144–156. <https://doi.org/10.1109/72.363441>
26. Kumar R, Smriti Srivastava JRP, Gupta AM (2018) Diagonal recurrent neural network based identification of nonlinear dynamical systems with Lyapunov stability based adaptive learning rates. *Neurocomputing* 287:102–117. <https://doi.org/10.1016/j.neucom.2018.01.073>
27. Fei J, Wang H (2019) Experimental investigation of recurrent neural network fractional-order sliding mode control of active power filter. *IEEE Trans Circuits Syst II Express Briefs* 67:2522–2526. <https://doi.org/10.1109/TCSII.2019.2953223>
28. Han HG, Zhang L, Hou Y, Qiao JF (2015) Nonlinear model predictive control based on a self-organizing recurrent neural network. *IEEE Trans Neural Netw Learn Syst* 27:402–415. <https://doi.org/10.1109/TNNLS.2015.2465174>
29. Qiu ZC, Zhang WZ (2019) Trajectory planning and diagonal recurrent neural network vibration control of a flexible manipulator using structural light sensor. *Mech Syst Signal Process* 132:563–594. <https://doi.org/10.1016/j.ymssp.2019.07.014>
30. Elkenawy A, El-Nagar AM, El-Bardini M, El-Rabaie NM (2020) Diagonal recurrent neural network observer-based adaptive control for unknown nonlinear systems. *Trans Inst Meas Control* 42(15):2833–2856. <https://doi.org/10.1177/0142331220921259>
31. Chen C, Peng S, Yao Z, Wang Q (2016) Multi induction motor synchronous drive system based on diagonal recurrent neural network control. *Int J Control Autom* 9(10):257–274. <https://doi.org/10.14257/ijca.2016.9.10.25>
32. Nazaruddin YY, Andrini AD, Anditio B (2018) PSO based PID controller for quadrotor with virtual sensor. *IFAC-PapersOnLine* 51(4):358–363. <https://doi.org/10.1016/j.ifacol.2018.06.091>
33. Ertuğrul ÖF (2020) A novel randomized machine learning approach: reservoir computing extreme learning machine. *Appl Soft Comput* 94:106433. <https://doi.org/10.1016/j.asoc.2020.106433>
34. De la Rosa E, Yu W (2016) Randomized algorithms for nonlinear system identification with deep learning modification. *Inf Sci* 364:197–212. <https://doi.org/10.1016/j.ins.2015.09.048>
35. Sousa J, Rebelo A, Cardoso JS (2019) Automation of waste sorting with deep learning. In: 2019 XV Workshop de Visão Computacional (WVC). IEEE, pp 43–48. <https://doi.org/10.1109/WVC.2019.8876924>
36. Coudray N, Ocampo PS, Sakellaropoulos T, Narula N, Snuderl M, Fenyo D, Tsirigos A (2018) Classification and mutation prediction from non-small cell lung cancer histopathology images using deep learning. *Nat Med* 24:1559–1567. <https://doi.org/10.1038/s41591-018-0177-5>
37. Nguyen TT, Hoang TD, Pham MT, Tuyet Trinh V, Nguyen TH, Huynh Q-T, Jo J (2020) Monitoring agriculture areas with satellite images and deep learning. *Appl Soft Comput* 95:106565. <https://doi.org/10.1016/j.asoc.2020.106565>
38. Wan S, Qi L, Xiaolong X, Tong C, Zonghua G (2019) Deep learning models for real-time human activity recognition with smartphones. *Mob Netw Appl* 25(2):743–755. <https://doi.org/10.1007/s11036-019-01445-x>
39. Arnold L, Rebecchi S, Chevallier S, Paugam-Moisy H (2011) An introduction to deep learning. In: European symposium on artificial neural networks (ESANN) pp 477–488. <https://doi.org/10.1201/9780429096280-14>
40. Zhang WJ, Yang G, Lin Y, Ji C, Gupta MM (2018) On definition of deep learning. In: 2018 World automation congress (WAC). IEEE, pp 1–5. <https://doi.org/10.23919/WAC.2018.8430387>
41. Gao DG, Sun YG, Luo SH, Lin GB, Tong LS (2020) Deep learning controller design of embedded control system for maglev train via deep belief network algorithm. *Des Autom Embed Syst* 24:161–181. <https://doi.org/10.1007/s10617-020-09237-3>
42. Ogunmolu O, Gu X, Jiang S, Gans N (2016) Nonlinear systems identification using deep dynamic neural networks. arXiv preprint. <https://doi.org/10.48550/arXiv.1610.01439>
43. Deepa SN, Baranilingesan I (2018) Optimized deep learning neural network predictive controller for continuous stirred tank reactor. *Comput Electr Eng* 71:782–797. <https://doi.org/10.1016/j.compeleceng.2017.07.004>
44. Gobinath S, Madheswaran M (2019) Deep perceptron neural network with fuzzy PID controller for speed control and stability analysis of BLDC motor. *Soft Comput* 24(13):10161–10180. <https://doi.org/10.1007/s00500-019-04532-z>
45. Zaki AM, El-Nagar AM, Mohammad El-Bardini M, Soliman FAS (2020) Deep learning controller for nonlinear system based on Lyapunov stability criterion. *Neural Comput Appl* 33(5):1515–1531. <https://doi.org/10.1007/s00521-020-05077-1>
46. Jin X, Shao J, Zhang X, An W, Malekian R (2016) Modeling of nonlinear system based on deep learning framework. *Nonlinear Dyn* 84:1327–1340. <https://doi.org/10.1007/s11071-015-2571-6>
47. Fort JC, Letremy P, Cottrell M (2002) Advantages and drawbacks of the Batch Kohonen algorithm. *ESANN* 2:223–230
48. Qiao J, Wang G, Li X, Li W (2018) A self-organizing deep belief network for nonlinear system modeling. *Appl Soft Comput* 65:170–183. <https://doi.org/10.1016/j.asoc.2018.01.019>
49. Kalteh AM, Hjorth P, Berndtsson R (2008) Review of the self-organizing map (SOM) approach in water resources: analysis, modelling and application. *Environ Model Softw* 23:835–845. <https://doi.org/10.1016/j.envsoft.2007.10.001>
50. Qiao J, Guo X, Li W (2020) An online self-organizing modular neural network for nonlinear system modeling. *Appl Soft Comput* 97:106777. <https://doi.org/10.1016/j.asoc.2020.106777>
51. Zhang N, Ding S, Zhang J, Xue Y (2018) An overview on restricted Boltzmann machines. *Neurocomputing* 275:1186–1199. <https://doi.org/10.1016/j.neucom.2017.09.065>
52. Kohonen T (1990) The self-organizing map. *Proc IEEE* 78(9):1464–1480. <https://doi.org/10.1109/5.58325>
53. Golovko V, Kroschchanka A, Turchenko V, Jankowski S, Treadwell D (2015) A new technique for restricted Boltzmann machine learning. In: 2015 IEEE 8th international conference on intelligent data acquisition and advanced computing systems: technology and applications (IDAACS). IEEE, pp 182–186. <https://doi.org/10.1109/IDAACS.2015.7340725>
54. Hinton GE (2012) A practical guide to training restricted Boltzmann machines. In: Montavon G, Orr GB, Müller K-R (eds) *Neural Networks: tricks of the trade*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 599–619. https://doi.org/10.1007/978-3-642-35289-8_32
55. Golovko V, Kroschchanka A, Treadwell D (2016) The nature of unsupervised learning in deep neural networks: a new understanding and novel approach. *Opt Mem Neural Netw* 25:127–141. <https://doi.org/10.3103/S1060992X16030073>
56. Hwang CL, Jan C (2015) Recurrent-neural-network-based multivariable adaptive control for a class of nonlinear dynamic systems with time-varying delay. *IEEE Trans Neural Netw Learn Syst* 27:388–401. <https://doi.org/10.1109/TNNLS.2015.2442437>
57. El-Nagar AM, El-Bardini M (2014) Practical realization for the interval type-2 fuzzy PD+ I controller using a low-cost microcontroller. *Arab J Sci Eng* 39:6463–6476. <https://doi.org/10.1007/s13369-014-1153-0>
58. Khater AA, El-Nagar AM, El-Bardini M, El-Rabaie N (2019) A novel structure of actor-critic learning based on an interval type-2 TSK fuzzy neural network. *IEEE Trans Fuzzy Syst* 28:3047–3061. <https://doi.org/10.1109/TFUZZ.2019.2949554>
59. Zhang X, Zhang H, Sun Q, Luo Y (2012) Adaptive dynamic programming-based optimal control of unknown nonaffine nonlinear discrete-time systems with proof of convergence.

- Neurocomputing 91:48–55. <https://doi.org/10.1016/j.neucom.2012.01.025>
60. Khater AA, El-Nagar AM, El-Bardini M, El-Rabaie NM (2020) Online learning based on adaptive learning rate for a class of recurrent fuzzy neural network. *Neural Comput Appl* 32:8691–8710. <https://doi.org/10.1007/s00521-019-04372-w>
61. Nasr MB, Chtourou M (2014) Neural network control of non-linear dynamic systems using hybrid algorithm. *Appl Soft Comput* 24:423–431. <https://doi.org/10.1016/j.asoc.2014.07.023>
62. Maraqa M, Al-Zboun F, Dhyabat M, Zitar RABu (2012) Recognition of Arabic sign language (ArSL) using recurrent neural networks. *J Intell Learn Syst Appl* 04(01):41–52. <https://doi.org/10.4236/jilsa.2012.41004>
63. Khater AA, El-Nagar AM, El-Bardini M, El-Rabaie NM (2019) Online learning of an interval type-2 TSK fuzzy logic controller for nonlinear systems. *J Frankl Inst* 356:9254–9285. <https://doi.org/10.1016/j.jfranklin.2019.08.031>
64. Khooban MH, Niknam T, Blaabjerg F, Dehghani M (2016) Free chattering hybrid sliding mode control for a class of non-linear systems: electric vehicles as a case study. *IET Sci Meas Technol* 10:776–785. <https://doi.org/10.1049/iet-smt.2016.0091>
65. Khooban MH, Vafamand N, Niknam T (2016) T–S fuzzy model predictive speed control of electrical vehicles. *ISA Trans* 64:231–240. <https://doi.org/10.1016/j.isatra.2016.04.019>
66. Khooban MH, ShaSadeghi M, Niknam T, Blaabjerg F (2017) Analysis, control and design of speed control of electric vehicles delayed model: Multi-objective fuzzy fractional-order PI λ D μ controller. *IET Sci Meas Technol* 11:249–261. <https://doi.org/10.1049/iet-smt.2016.0277>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.