Received 23 February 2023, accepted 23 March 2023, date of publication 29 March 2023, date of current version 4 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3263180

RESEARCH ARTICLE

Deep Learning-Based State-Dependent ARX Modeling and Predictive Control of Nonlinear Systems

TIAO KANG[1,2], **HUI PENG**[1], **WENQUAN XU**[1], **YAPENG SUN**[1], **AND XIAOYAN PENG**[3]

[1]School of Automation, Central South University, Changsha 410083, China
[2]Engineering Training Center, Hunan Institute of Engineering, Xiangtan 411101, China
[3]College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China

Corresponding author: Hui Peng (huipeng@csu.edu.cn)

ABSTRACT For many practical industrial objects with time-varying operating points, strong nonlinearity, and difficulty in obtaining analytical models, the data-driven identification method is usually used to model such nonlinear systems. However, it is difficult for traditional modeling algorithms to effectively extract the dynamic characteristics of nonlinear systems from data and obtain accurate mathematical models. In this paper, we consider using the deep learning network combined with the state-dependent exogenous variable autoregressive (SD-ARX) model framework to build the nonlinear system model, so as to effectively and accurately learn the space-time characteristics of the nonlinear system from the sample data. Based on the idea, the hybrid models, i.e., the RNN-ARX model, CNN-ARX model, and RNN-CNN-ARX model are built, which use recurrent neural networks (RNN), convolutional neural networks (CNN) and their combination to fit the function-type coefficients of SD-ARX model, respectively. SD-ARX model based on deep learning has the advantages of local linearity and global nonlinearity. Compared with the other two models, the RNN-CNN-ARX model has a stronger ability to extract the multidimensional spatiotemporal dynamic characteristics of nonlinear systems, because it combines the advantages of RNN in mining temporal features and CNN in extracting spatial features. According to the structural characteristics of these models, three model-based predictive control (MPC) strategies are designed, i.e., RNN-ARX-MPC, CNN-ARX-MPC, and RNN-CNN-ARX-MPC. The real-time control comparative experiment on an actual multi-water-tank object shows that the proposed modeling and MPC method is feasible and effective for the modeling and predictive control of the nonlinear system.

INDEX TERMS RNN-ARX model, CNN-ARX model, RNN-CNN-ARX model, model predictive control, deep learning.

I. INTRODUCTION

Model predictive control (MPC) is an optimal control algorithm developed for industrial process control. Because its control mechanism has good adaptability to complex industrial processes, it has attracted the extensive attention of scholars and has achieved a lot of research results [1], [2]. MPC algorithm uses a dynamic model to predict the future behavior of the system. Therefore, the key to realizing MPC is to establish a suitable model to accurately represent the dynamic characteristics of the system, and the model is easy to be used to design subsequent predictive controllers. Considering that the actual object to be controlled is usually a complex nonlinear system, using physical modeling methods, some important parameters may not be determined or difficult to obtain, which will lead to the failure to establish an accurate model [3]. Therefore, in this paper, input/output data-driven technology is used to build the model of a nonlinear system.

The associate editor coordinating the review of this manuscript and approving it for publication was Azwirman Gusrialdi.

Its remarkable advantage is that it does not need to accurately understand the complex variable relations in nonlinear systems in advance.

When the data identification model of the nonlinear system is used to design a predictive controller, the traditional piecewise linearization [4] or local linearization [5] modeling method is beneficial to the design of the predictive controller. However, these models have some limitations in describing the dynamic characteristics of complex nonlinear systems. In addition, directly using the nonlinear models that can well describe the nonlinear characteristics of the object, such as the bilinear model [6], Volterra series model [7], and neural network model [8], to design MPC needs to solve a high-order, constrained, non-convex nonlinear optimization problems online to obtain the control signal. Usually, it requires a lot of online computation, and sometimes even cannot obtain a feasible solution. Therefore, this method has strong limitations for complex industrial objects with high real-time requirements.

In the past few years, many researches and applications have tended to adopt composite models, such as Hammerstein model [9], Wiener model [10], and SD-ARX model [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]. The combined model can solve the aforementioned problems well, but the Hammerstein model and Wiener model can only describe the static nonlinear characteristics. SD-ARX model can well describe nonlinear dynamic characteristics because of its state-dependent function coefficients, and it may be easily used to design predictive controllers because of its quasi-linear ARX model structure, so it has broad application prospects. For example, the regression tree and random forest algorithm in machine learning technology were combined with the ARX model for complex system modeling and MPC design to improve prediction accuracy and control performance [11], [12]. RBF-ARX model based on radial basis function (RBF) neural network has been widely used in nonlinear time series modeling and predictive control [13], [14], [15], [16], [17], [18], [19]. In addition, wavelet neural network [20] and deep belief network [21] were also used to combine with the ARX model for nonlinear system modeling. However, these feedforward neural networks can only transfer data information from the input layer to the output layer, and there is no information feedback between nodes of each layer of the neural network, which cannot well extract the multi-dimensional spatiotemporal dynamic characteristics of the nonlinear system, so it is difficult to ensure the accuracy of modeling in some cases.

Recently, deep learning technology has performed well in various fields [22], [23]. Using a multi-layer network structure and a series of nonlinear functions, the deep learning model can effectively extract the dynamic characteristics of nonlinear systems, and convert complex process data information into more abstract mathematical expressions. One of the promising deep learning networks is RNN, which has achieved great success in machine translation [24], [25], sentiment analysis [26], [27], speech recognition [28], [29], fault diagnosis [30], [31], etc. The biggest characteristic of RNN is the feedback between neural network nodes at all levels. The output of the neural network node at one time can be transmitted to the neural network node again as the input at the next time, which can maintain the sequence dependency in the data, and has great advantages in nonlinear system modeling. For instance, Alhajeri et al. [32] studied an RNN-based output feedback model predictive controller, and the experiment in a continuous stirred tank reactor proved the effectiveness of this method. Zhang et al. [33] studied a predictive control scheme based on RNN, which achieved the formation flight of multiple unmanned quadcopters and trajectory tracking control through computer simulation. Wu et al. [34], [35] developed the MPC algorithm based on the RNN model set to predict the dynamic characteristics of a nonlinear system. These nonlinear system modeling methods based on RNN can better capture the time dimension features in the data, but some spatial features may not be well captured, which has a certain impact on modeling accuracy. In addition, these methods have only been tested and validated through digital simulation and have not been applied to actual systems.

CNN has a strong local spatial feature extraction ability, which is widely used in image processing [36], [37], [38], [39]. Therefore, the hybrid neural network composed of RNN and CNN can make up for the above shortcomings and better capture the spatiotemporal dynamic characteristics contained in the nonlinear system data. However, so far, the research on the hybrid neural network composed of RNN and CNN mainly focused on image recognition [40], [41], [42], emotion analysis [43], [44], [45], and text recognition [46], [47], but its research in MPC and nonlinear system modeling has not been found.

In this paper, the RNN-ARX model, CNN-ARX model, and RNN-CNN-ARX model are established to express the dynamic behavior of the nonlinear system by fitting the function coefficients of SD-ARX model with deep learning RNN or/and CNN. The SD-ARX model based on deep learning has the characteristics of local linearity and global nonlinearity, which is more conducive to system modeling and controller design. In addition, the three model-based predictive controllers, i.e., RNN-ARX-MPC, CNN-ARX-MPC, and RNN-CNN-ARX-MPC are designed, and real-time control experiments are carried out for the actual multi-tank object. The results show that the proposed model and model-based control strategy are feasible, especially RNN-CNN-ARX-MPC is superior to other control methods, because it combines the advantages of RNN in mining temporal features and CNN in extracting spatial features, and can effectively and accurately learn the temporal and spatial features of nonlinear systems from a large amount of data.

The main contributions of this paper are as follows: 1) Three state-dependent ARX models based on deep learning are proposed to describe the dynamic characteristics of

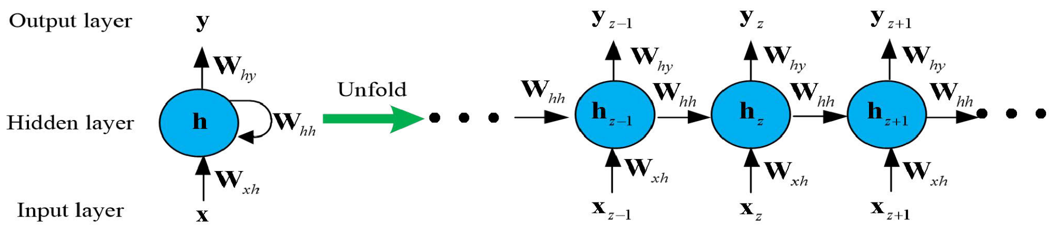


FIGURE 1. The schematic diagram for RNN.

a class of nonlinear systems, namely RNN-ARX model, CNN-ARX model, and RNN-CNN-ARX model; 2) Based on the proposed models, the predictive controllers are designed; 3) Using the proposed modeling and control method, the real-time control experiment is successfully carried out on a practical nonlinear multi-tank device. As far as we know, there is no report on the successful application of deep learning methods in nonlinear system modeling and MPC in practical nonlinear process control objects. This work shows how to build deep learning related models for existing nonlinear systems, design MPC strategies, and present actual control results, rather than just provide digital simulation results, as is done in relevant papers.

The structure of the rest of this paper is as follows. Section II introduces the associated work. The RNN-ARX, CNN-ARX, and RNN-CNN-ARX models are established in Section III. The above models-based predictive controllers are designed in Section IV. Section V shows the real-time control results on the multi-water tank plant using the proposed method. Section VI concludes this study.

II. ASSOCIATED WORK

An overview of the SD-ARX model, RNN, and CNN will be provided in this section.

A. SD-ARX MODEL

With regard to the actual industrial process, a class of nonlinear systems can be expressed as the nonlinear ARX model shown below [18]:

$$\begin{aligned} y(z) &= \lambda(\boldsymbol{\delta}(z-1)) + \gamma(z) \\ \boldsymbol{\delta}(z-1) &= [y(z-1)^{\mathrm{T}}, \ldots, y(z-n_y)^{\mathrm{T}}, \\ &\qquad u(z-1)^{\mathrm{T}}, \ldots, u(z-n_u)^{\mathrm{T}}]^{\mathrm{T}} \end{aligned} \tag{1}$$

where $y(z)$ denotes the output, $u(z)$ denotes the input, $\gamma(z)$ denotes the Gaussian white noise, n_y and n_u are the orders of the model. $\lambda(\bullet)$ can be extended using Taylor polynomials at a given operating point $\boldsymbol{\delta}_0$ as follows

$$\begin{aligned} \lambda(\boldsymbol{\delta}(z-1)) &= \lambda(\boldsymbol{\delta}_0) + \lambda'(\boldsymbol{\delta}_0)^{\mathrm{T}}(\boldsymbol{\delta}(z-1)-\boldsymbol{\delta}_0) \\ &\quad + \frac{1}{2}(\boldsymbol{\delta}(z-1)-\boldsymbol{\delta}_0)^{\mathrm{T}}\lambda''(\boldsymbol{\delta}_0)(\boldsymbol{\delta}(z-1)-\boldsymbol{\delta}_0) \\ &\quad + \ldots + r_n(\boldsymbol{\delta}(z-1)) \end{aligned} \tag{2}$$

where $r_n(\boldsymbol{\delta}(z-1))$ is the remainder after Taylor expansion, and then the equation (1) can be converted to

$$\begin{cases} y(z) = \psi_0\left(\boldsymbol{\delta}(z-1)\right) + \sum_{i=1}^{n_y} \psi_{y,i}\left(\boldsymbol{\delta}(z-1)\right) y(z-i) \\ + \sum_{j=1}^{n_u} \psi_{u,j}\left(\boldsymbol{\delta}(z-1)\right) u(z-j) + \gamma(z) \\ \eta_0 = \lambda(\boldsymbol{\delta}_0) - \lambda'(\boldsymbol{\delta}_0)^{\mathrm{T}}\boldsymbol{\delta}_0 + \frac{1}{2}\boldsymbol{\delta}_0^{\mathrm{T}}\lambda''(\boldsymbol{\delta}_0)\boldsymbol{\delta}_0 + \ldots \\ \eta_1(\boldsymbol{\delta}(z-1)) = r_n(\boldsymbol{\delta}(z-1)) \\ \Omega_0 = \lambda'(\boldsymbol{\delta}_0)^{\mathrm{T}} - \frac{1}{2}\boldsymbol{\delta}_0^{\mathrm{T}}\lambda''(\boldsymbol{\delta}_0) - \frac{1}{2}\boldsymbol{\delta}_0^{\mathrm{T}}\lambda''(\boldsymbol{\delta}_0)^{\mathrm{T}} + \ldots \\ \Omega_1(\boldsymbol{\delta}(z-1)) = \frac{1}{2}\boldsymbol{\delta}(z-1)^{\mathrm{T}}\lambda''(\boldsymbol{\delta}_0) + \ldots \\ \eta_0 + \eta_1(\boldsymbol{\delta}(z-1)) = \psi_0\left(\boldsymbol{\delta}(z-1)\right) \\ \Omega_0 + \Omega_1(\boldsymbol{\delta}(z-1)) \\ = \left[\psi_{y,1}\left(\boldsymbol{\delta}(z-1)\right), \ldots, \psi_{y,n_y}\left(\boldsymbol{\delta}(z-1)\right), \right. \\ \left. \psi_{u,1}\left(\boldsymbol{\delta}(z-1)\right), \ldots, \psi_{u,n_u}\left(\boldsymbol{\delta}(z-1)\right)\right] \end{cases} \tag{3}$$

where $\{\psi_{y,i}(\boldsymbol{\delta}(z-1)) | i = 1, \ldots, n_y\}$, $\{\psi_{u,j}(\boldsymbol{\delta}(z-1)) | j = 1, \ldots, n_u\}$ and $\psi_0(\boldsymbol{\delta}(z-1))$ are the state-dependent function coefficients of each regression variable after Taylor expansion, which change with the state working point $\boldsymbol{\delta}(z-1)$; $\boldsymbol{\delta}(z-1)$ is the state variable that causes the nonlinear change of the system. It can be the system output signal, control input signal, or a combination of these signals. When $\boldsymbol{\delta}(z-1)$ is fixed, model (3) is simplified to a linear ARX model, which is a local linearization of the model at a working point of the system, and when $\boldsymbol{\delta}(z-1)$ follows the system change, it can naturally switch to the next local linear ARX model. This feature decomposes the complexity of the model into the autoregressive parts of their respective variables and is helpful for the subsequent model-based predictive controller design. The coefficients $\psi_{y,i}$, $\psi_{u,j}$ and ψ_0 in (3) can be fitted by some neural networks, such as BRF neural network [13], [14], [15], [16], [17], [18], [19] and wavelet neural network [20]. In the next section, we will obtain a class of SD-ARX models by approximating the coefficients of the model (3) through deep learning networks, which can better represent the spatiotemporal dynamic properties of nonlinear systems.

B. RNN

RNN is a neural network with memory, which can maintain sequence correlation. In addition to the input information of the current time step, the input of each neural network node also includes the memory information generated in the previous time step. The architecture of RNN is shown in Fig. 1 [48], it is repeated according to the time series, and the weight coefficients are shared in each time step, which can significantly reduce the number of parameters in the model, thus reducing the complexity of calculation and saving calculation time. RNN can be expressed as follows

$$\begin{cases}\mathbf{h}_z=\sigma(\mathbf{x}_z\mathbf{W}_{xh}+\mathbf{h}_{z-1}\mathbf{W}_{hh}+\mathbf{b}_h)\\ \mathbf{y}_z=f(\mathbf{h}_z\mathbf{W}_{hy}+\mathbf{b}_y)\end{cases}\tag{4}$$

where $\mathbf{x}_z$ and $\mathbf{y}_z$ denotes the input and output, respectively; $\sigma(\bullet)$ and $f(\bullet)$ denotes the nonlinear activation functions; $\mathbf{h}_z$ represents the hidden layer unit at time z, $\mathbf{W}_{hh}$, $\mathbf{W}_{xh}$ and $\mathbf{W}_{hy}$ represent the weight coefficients, $\mathbf{b}_h$ and $\mathbf{b}_y$ represents the offset.

C. CNN

CNN has excellent nonlinear property extraction capabilities and can capture the local spatial characteristics in the data, and its fundamental composition is seen in Fig. 2. The connection of neurons between the traditional neural network layers is completely connected, which inevitably leads to the rapid increase of the number of parameters to be identified with the increase of the network layer. CNN uses a convolution kernel smaller than the input size as a filter to extract local features in the input and establishes local connections between neurons, thus reducing the number of network parameters. In addition, CNN uses weight sharing, and the convolution kernel can extract the same features at different input positions, which further reduces the number of recognition parameters, reduces the risk of over-fitting, and enhances the generalization ability of the model. The convolution operation is its core, and its calculation form is as follows

$$\mathbf{x}_{j_1}^{l_1}=\sigma_1\left(\sum_{i_1=1}^{g_1}\mathbf{x}_{i_1}^{l_1-1}\otimes\mathbf{W}_{i_1j_1}^{l_1}+\mathbf{b}_{j_1}^{l_1}\right)\tag{5}$$

where $\otimes$ denotes convolution operation; $\mathbf{x}_{j_1}^{l_1}$ is the j_1 -th feature map of the l_1 -th convolution layer; $\mathbf{W}_{i_1j_1}^{l_1}$ is the convolution kernel matrix, $\mathbf{b}_{j_1}^{l_1}$ is the offset, g_1 is the amount of input feature maps; $\sigma_1(\bullet)$ is the nonlinear activation function.

III. THE PROPOSED MODELS

This section introduces the three combined models proposed in this article, i.e., the RNN-ARX model, CNN-ARX model, and RNN-CNN-ARX model.

A. RNN-ARX MODEL

RNN uses time series data of nonlinear systems as input, which can overcome the shortcoming of no feedback between nodes in traditional forward neural networks (such as RBF neural network and wavelet neural network). The output of the RNN node at one time can be used as the input of the next time to be transmitted to the neural network node again so that the sequence correlation in the data can be maintained, so as to better process the time series data of the nonlinear system. Using RNN to approximate the functional coefficients of the model (3), the RNN-ARX model can be obtained, which combines the advantages of RNN in processing sequence data and the nonlinear description ability of SD-ARX model to better describe the nonlinear dynamic characteristics of nonlinear system. Fig. 3 depicts the structure of RNN-ARX model, and its formula is as follows.

$$\begin{cases}\mathbf{y}(z)=\boldsymbol{\psi}_0\left(\boldsymbol{\delta}(z-1)\right)+\displaystyle\sum_{i=1}^{n_y}\boldsymbol{\psi}_{y,i}\left(\boldsymbol{\delta}(z-1)\right)\mathbf{y}(z-i)+\\ \displaystyle\sum_{j=n_d}^{n_u+n_d-1}\boldsymbol{\psi}_{u,j}\left(\boldsymbol{\delta}(z-1)\right)\mathbf{u}(z-j)+\gamma(z)\\ \mathbf{W}\left(f(\mathbf{h}_d^n(z)\mathbf{W}_{hy}+\mathbf{b}_y)\right)+\mathbf{b}\\ =[\boldsymbol{\psi}_0(\boldsymbol{\delta}(z-1)),\boldsymbol{\psi}_{y,i}(\boldsymbol{\delta}(z-1)),\\ \boldsymbol{\psi}_{u,j}(\boldsymbol{\delta}(z-1))]\\ \mathbf{h}_l^r(z)=\sigma(\mathbf{h}_l^{r-1}(z)\mathbf{W}_{xh}^r+\mathbf{h}_{l-1}^r(z)\mathbf{W}_{hh}^r+\mathbf{b}_h^r),\\ r=2,3,\ldots,n\\ \mathbf{h}_l^1(z)=\sigma(\delta_l\mathbf{W}_{xh}^1+\mathbf{h}_{l-1}^1(z)\mathbf{W}_{hh}^1+\mathbf{b}_h^1),\ l=1,2,\ldots,d\\ \mathbf{h}_0^m(z)=\mathbf{0},\ m=1,2,\ldots,n\\ \boldsymbol{\delta}(z-1)=[\delta_1,\delta_2,\ldots,\delta_d]^T\end{cases}\tag{6}$$

where $\mathbf{y}(z)$ denotes the output; $\mathbf{u}(z)$ denotes the input; $\gamma(z)$ denotes white noise; n_yn_u, d are the orders of the model, and n_d is the time delay; $\boldsymbol{\psi}_0(\boldsymbol{\delta}(z-1))$, $\boldsymbol{\psi}_{y,i}(\boldsymbol{\delta}(z-1))$ and $\boldsymbol{\psi}_{u,j}(\boldsymbol{\delta}(z-1))$ are the functional coefficients of the model, which are determined by the state working point $\boldsymbol{\delta}(z-1)$, and they relate to the system input or/and output; n represents the number of hidden layers; $\mathbf{h}_l^r(z)$ denotes the state of the r -th hidden layer of time step l at time z; $\sigma(\bullet)$ and $f(\bullet)$ are the activation functions of the hidden and full connection layers, such as Sigmoid function, Relu function, and Tanh function, which can enhance the ability of the model to capture the nonlinearity of the complex system; $\{\mathbf{W}_{xh}^m,\mathbf{W}_{hh}^m|m=1,\ldots,n\}$ and $\{\mathbf{b}_h^m|m=1,\ldots,n\}$ are the weight coefficients and offsets of the hidden layer, respectively; $\left\{\mathbf{W}_{hy},\mathbf{W}\right\}$ and $\left\{\mathbf{b}_y,\mathbf{b}\right\}$ are the weight coefficients and offsets of the full connection layer, respectively.

B. CNN-ARX MODEL

The CNN-ARX model can be constructed by using CNN to fit the functional coefficients of the model (3), which combines the local feature extraction ability of CNN and the nonlinear description ability of the SD-ARX model. Fig. 4 depicts the structure of the CNN-ARX model. The convolution layer and pooling layer of CNN are the basic structures that distinguish CNN from other traditional neural networks. The convolution layer can extract the features of the data to be recognized. Using the pooling layer to remove redundant information extracted from the convolution layer can reduce the complexity of the network and improve the robustness of the model.

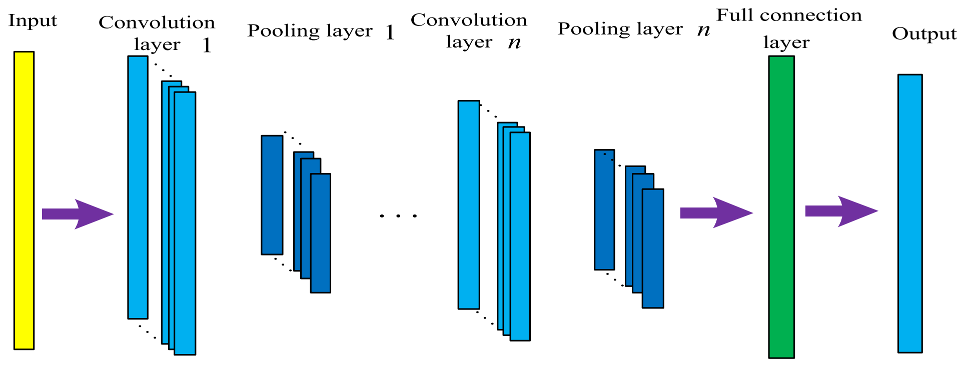


FIGURE 2. Framework for CNN.

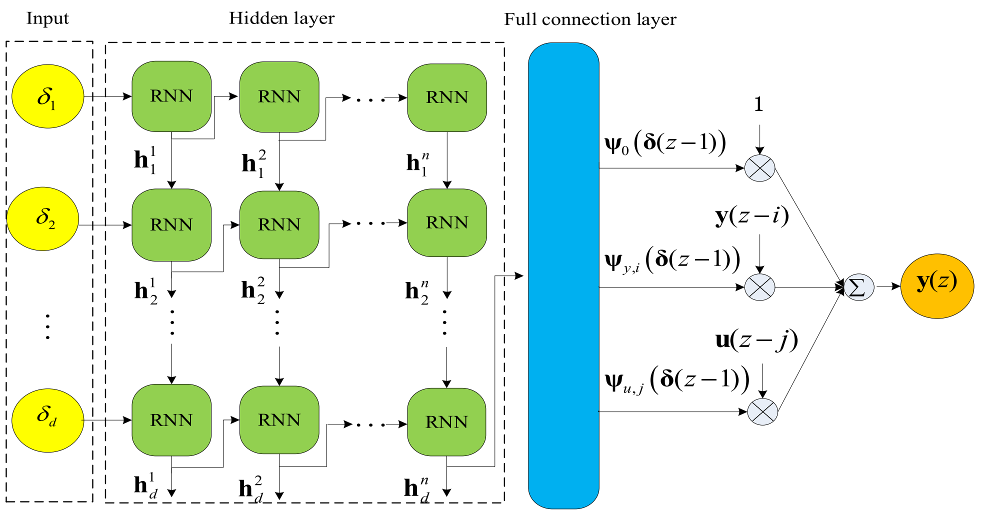


FIGURE 3. Structure of the RNN-ARX model.

The CNN-ARX model is represented as follows

$$\begin{cases} \mathbf{y}(z) = \boldsymbol{\psi}_0(\boldsymbol{\delta}(z-1)) + \sum_{i=1}^{n_y} \boldsymbol{\psi}_{y,i}(\boldsymbol{\delta}(z-1))\,\mathbf{y}(z-i) + \\ \sum_{j=n_d}^{n_u+n_d-1} \boldsymbol{\psi}_{u,j}(\boldsymbol{\delta}(z-1))\,\mathbf{u}(z-j)+\gamma(z) \\ \mathbf{W}_1\left(f_1(\tilde{\mathbf{x}}^{n_1}(z)\mathbf{W}_{xy}+\mathbf{b}_x)\right)+\mathbf{b}_1 \\ = [\boldsymbol{\psi}_0(\boldsymbol{\delta}(z-1))\,, \boldsymbol{\psi}_{y,i}(\boldsymbol{\delta}(z-1))\,, \\ \boldsymbol{\psi}_{u,j}(\boldsymbol{\delta}(z-1))] \\ \tilde{\mathbf{x}}^{n_1}(z) = \sigma_3\left(\hat{\mathbf{x}}_{j_1}^{n_1}(z)\right); \hat{\mathbf{x}}_{j_1}^{n_1}(z)=\sigma_2\left(\mathbf{x}_{j_1}^{n_1}(z)\right) \\ \mathbf{x}_{j_1}^{l_1}(z) = \sigma_1\left(\sum_{i_1=1}^{g_1}\mathbf{x}_{i_1}^{l_1-1}(z)\otimes \mathbf{W}_{i_1j_1}^{l_1}+\mathbf{b}_{j_1}^{l_1}\right), \ 1\le l_1\le n_1 \\ \mathbf{x}^0(k)=\boldsymbol{\delta}(z-1)=[\delta_1,\delta_2,\ldots,\delta_d]^T \end{cases} \tag{7}$$

where $\mathbf{x}^0(z)$ is the input of the CNN at time z; $\sigma_1(\bullet)$ and $f_1(\bullet)$ are the activation functions of the convolutional and full connection layer, such as the Sigmoid function, Relu function, and Tanh function; $\sigma_2(\bullet)$ is the pooling function to perform pooling operation on the input feature map; $\sigma_3(\bullet)$ indicates the flattening operation $\tilde{\mathbf{x}}^{n_1}(z)$ represents the one-dimensional vector obtained by flattening the output feature maps $\hat{\mathbf{x}}_{j_1}^{n_1}(z)$; $\mathbf{x}_{j_1}^{l_1}(z)$ represents the j_1 -th feature map of the l_1 -th convolution layer at time z; $\otimes$ indicates the convolution operation, and g_1 is the number of input feature maps; $\{\mathbf{W}_{i_1j_1}^{l_1}|l_1 = 1,\ldots,n_1\}$ and $\{\mathbf{b}_{j_1}^{l_1}|l_1 = 1,\ldots,n_1\}$ are the weight coefficients and offsets of the convolution layer, respectively; $\left\{\mathbf{W}_{xy}, \mathbf{W}_1\right\}$ and $\{\mathbf{b}_x, \mathbf{b}_1\}$ are the weight coefficients and offsets of the full connection layer, respectively.

C. RNN-CNN-ARX MODEL

Combining RNN and CNN to approximate the SD-ARX model coefficients, the RNN-CNN-ARX model can be established, which will have RNN's ability to mine temporal features of sequence data, CNN's ability to extract spatial features and the SD-ARX model's nonlinear description ability. It can effectively and accurately learn spatiotemporal characteristics from a large number of data to fully describe the dynamic behavior of nonlinear systems. Fig. 5 depicts the structure of the RNN-CNN-ARX model, and its

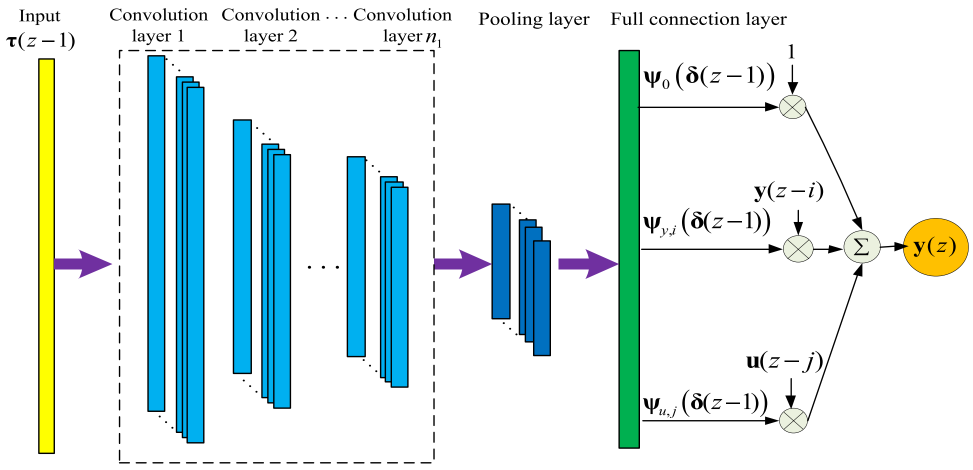


FIGURE 4. Framework of the CNN-ARX model.

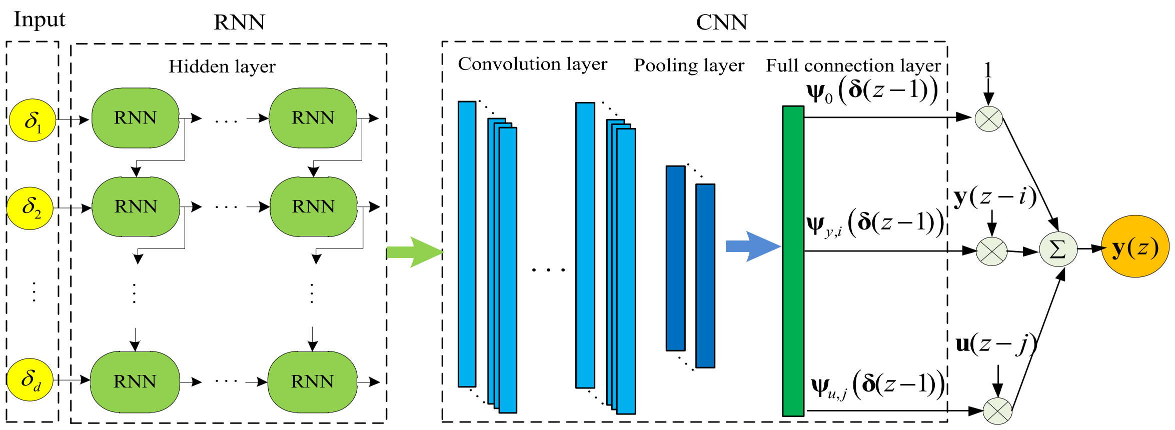


FIGURE 5. Framework of the RNN-CNN-ARX model.

mathematical expression is given below

$$\begin{cases}
\mathbf{y}(z) = \boldsymbol{\psi}_0(\boldsymbol{\delta}(z-1)) + \sum_{i=1}^{n_y} \boldsymbol{\psi}_{y,i}(\boldsymbol{\delta}(z-1))\,\mathbf{y}(z-i) + \\
\sum_{j=n_d}^{n_u+n_d-1} \boldsymbol{\psi}_{u,j}(\boldsymbol{\delta}(z-1))\,\mathbf{u}(z-j) + \gamma(z) \\
\mathbf{W}_1\left(f_1(\tilde{\mathbf{x}}^{n_1}(z)\mathbf{W}_{xy} + \mathbf{b}_x)\right) + \mathbf{b}_1 \\
= [\boldsymbol{\psi}_0(\boldsymbol{\delta}(z-1)), \boldsymbol{\psi}_{y,i}(\boldsymbol{\delta}(z-1)), \\
\boldsymbol{\psi}_{u,j}(\boldsymbol{\delta}(z-1))] \\
\tilde{\mathbf{x}}^{n_1}(z) = \sigma_3(\hat{\mathbf{x}}_{j_1}^{n_1}(k)); \hat{\mathbf{x}}_{j_1}^{n_1}(z) = \sigma_2\left(\mathbf{x}_{j_1}^{n_1}(z)\right) \\
\mathbf{x}_{j_1}^{l_1}(z) = \sigma_1\left(\sum_{i_1=1}^{g_1} \mathbf{x}_{i_1}^{l_1-1}(z) \otimes \mathbf{W}_{i_1 j_1}^{l_1} + \mathbf{b}_{j_1}^{l_1}\right), 1 \le l_1 \le n_1 \\
\mathbf{x}^0(z) = \mathbf{h}_l^n(z) \\
\mathbf{h}_l^r(z) = \sigma(\mathbf{h}_l^{r-1}(z)\mathbf{W}_{xh}^r + \mathbf{h}_{l-1}^r(z)\mathbf{W}_{hh}^r + \mathbf{b}_h^r), \\
r = 2, 3, \ldots, n \\
\mathbf{h}_l^1(z) = \sigma(\delta_l \mathbf{W}_{xh}^1 + \mathbf{h}_{l-1}^1(z)\mathbf{W}_{hh}^1 + \mathbf{b}_h^1), l = 1, 2, \ldots, d \\
\mathbf{h}_0^m(z) = \mathbf{0},\ m = 1, 2, \ldots, n;\ \boldsymbol{\delta}(z-1) = [\delta_1, \delta_2, \ldots, \delta_d]^T
\end{cases} \tag{8}$$

It can be seen from Fig. 5 and formula (8) that in the RNN-CNN-ARX model, the time characteristics of the input sequence data are first mined by RNN, and then the spatial characteristics are extracted by CNN. Finally, the state-dependent coefficients of the model are calculated through the full connection layer.

The estimation methods of the above three SD-ARX models based on the deep learning model are as follows. First, set the model structure parameters (such as model order, network layers, number of nodes) and super parameters (such as activation function, parameter optimization algorithm), and initialize the parameters to be estimated in the model. Then, calculate the loss function based on the predicted output sequence of the model, i.e., the mean square error (MSE) of predicted output and actual output, and update the parameters of the model through the back-propagation algorithm until the loss function is minimized. Finally, change the structure and super parameters of the model and repeat the above steps, and select the model with the minimum loss function value as the final RNN-ARX model, CNN-ARX model, or RNN-CNN-ARX model.

IV. MODEL PREDICTIVE CONTROLLER DESIGN

The proposed models-based predictive controllers are designed in this section. The goal of predictive controller design is to use the pseudo linear characteristics of local linearity and global nonlinearity of the model, calculate the predictive output of the nonlinear system by locally linearizing the model under the current state of the working point, and finally solve the quadratic programming problem (QP) online to obtain the MPC law.

First, in order to facilitate the design of the predictive controller, the model (6) - (8) is transformed into the following polynomial form

$$\begin{cases} \mathbf{y}(z) = \boldsymbol{\psi}_0(z-1) + \sum_{p=1}^{k_s} \widehat{\alpha}_{p,z-1}\mathbf{y}(z-p) + \\ \sum_{p=1}^{k_s} \widehat{\beta}_{p,z-1}\mathbf{u}(z-p) + \gamma(z) \\ \boldsymbol{\psi}_0(z-1) = \boldsymbol{\psi}_0(\boldsymbol{\delta}(z-1)) \\ k_s = \max\left(n_y, n_u + n_d - 1\right) \\ \widehat{\alpha}_{p,z-1} = \begin{cases} \boldsymbol{\psi}_{y,p}(\boldsymbol{\delta}(z-1)), (p \le n_y) \\ \mathbf{0} \ , (p > n_y) \end{cases} \\ \widehat{\beta}_{p,z-1} = \begin{cases} \boldsymbol{\psi}_{u,p}(\boldsymbol{\delta}(z-1)), (n_d \le p \le n_u + n_d - 1) \\ \mathbf{0} \ , \text{else} \end{cases} \end{cases} \tag{9}$$

In order to convert the above polynomial model into state space form, the following state vector is defined

$$\begin{cases} \mathbf{x}(z) = [\mathbf{x}_{1,z}^{\mathrm{T}}\mathbf{x}_{2,z}^{\mathrm{T}} \cdots \mathbf{x}_{k_s,z}^{\mathrm{T}}]^T, \\ \mathbf{x}_{1,z}^{\mathrm{T}} = \mathbf{y}(z), \\ \mathbf{x}_{p,z} = \sum_{i=1}^{k_s+1-p} \widehat{\alpha}_{i+p-1,z-1}\mathbf{y}(z-i) + \\ \sum_{i=1}^{k_s+1-p} \widehat{\beta}_{i+p-1,z-1}\mathbf{u}(z-i), \\ p = 2, 3, \ldots, k_s \end{cases} \tag{10}$$

Then, model (9) can be transformed into the subsequent state space model

$$\begin{cases} \mathbf{x}(z+1) = \mathbf{D}_z\mathbf{x}(z) + \mathbf{E}_z\mathbf{u}(z) + \boldsymbol{\psi}_z + \Omega(z+1) \\ \mathbf{y}(z) = \mathbf{F}\mathbf{x}(z) \end{cases} \tag{11}$$

where

$$\begin{cases} \mathbf{D}_z = \begin{bmatrix} \widehat{\alpha}_{1,z} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \widehat{\alpha}_{2,z} & \mathbf{0} & \mathbf{I} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \mathbf{0} \\ \widehat{\alpha}_{k_s-1,z} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \\ \widehat{\alpha}_{k_s,z} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}, \\ \mathbf{E}_z = [\widehat{\beta}_{1,z}\,\widehat{\beta}_{2,z} \cdots \widehat{\beta}_{k_s,z}]^{\mathrm{T}}, \\ \mathbf{F} = [\mathbf{I}\,\mathbf{0}\,\cdots\mathbf{0}], \\ \boldsymbol{\psi}_z = [\boldsymbol{\psi}_0\mathbf{0}\cdots\mathbf{0}]^{\mathrm{T}}, \\ \Omega(z+1) = [\gamma(z+1)\mathbf{0}\cdots\mathbf{0}]^{\mathrm{T}}, \end{cases} \tag{12}$$

The multi-step forward prediction output of the model (11) can be computed using equations (11-12). To further develop the MPC algorithm, the following vectors are defined

$$\begin{cases} \tilde{\mathbf{X}}(z) = [\tilde{\mathbf{x}}(z+1|z)^{\mathrm{T}}\tilde{\mathbf{x}}(z+2|z)^{\mathrm{T}} \cdots \tilde{\mathbf{x}}(z+N_y|z)^{\mathrm{T}}]^{\mathrm{T}} \\ \tilde{\mathbf{Y}}(z) = [\tilde{\mathbf{y}}(z+1|z)^{\mathrm{T}}\tilde{\mathbf{y}}(z+2|z)^{\mathrm{T}} \cdots \tilde{\mathbf{y}}(z+N_y|z)^{\mathrm{T}}]^{\mathrm{T}} \\ \tilde{\mathbf{U}}(z) = [\mathbf{u}(z)^{\mathrm{T}}\mathbf{u}(z+1)^{\mathrm{T}} \cdots \mathbf{u}(z+N_u-1)^{\mathrm{T}}]^{\mathrm{T}} \\ \tilde{\Psi}_z = [\boldsymbol{\psi}_z^{\mathrm{T}}\boldsymbol{\psi}_{z+1}^{\mathrm{T}} \cdots \boldsymbol{\psi}_{z+N_y-1}^{\mathrm{T}}]^{\mathrm{T}} \end{cases} \tag{13}$$

where $\tilde{\mathbf{X}}(z)$ and $\tilde{\mathbf{Y}}(z)$ represent the multi-step forward prediction vector of the state and output; $\left\{\tilde{\mathbf{x}}(z+i|z)^{\mathrm{T}}|i = 1, \ldots, N_y\right\}$ and $\left\{\tilde{\mathbf{y}}(z+i|z)^{\mathrm{T}}|i = 1, \ldots, N_y\right\}$ are the i -step forward prediction of the state and output based on the models (9) and (11) at time z, respectively; N_y is the prediction time domain, $N_u(N_u \le N_y)$ is the control time domain, and the control input after step N_u will not change, i.e., $\mathbf{u}(z+i) = \mathbf{u}(z+N_u-1)(i \ge N_u)$. Then at time z, the output expression of multi-step forward predictive control of the nonlinear system is as follows

$$\begin{cases} \tilde{\mathbf{X}}(z) = \tilde{\mathbf{D}}_z\mathbf{X}(z) + \tilde{\mathbf{E}}_z\tilde{\mathbf{U}}(z) + \tilde{\Sigma}_z\tilde{\Psi}_z, \\ \tilde{\mathbf{Y}}(z) = \tilde{\mathbf{F}}\tilde{\mathbf{X}}(z) \end{cases} \tag{14}$$

where (15) and (16), as shown at the bottom of the next page.

Note that $\tilde{\mathbf{D}}_z$, $\tilde{\mathbf{E}}_z$, $\tilde{\mathbf{F}}$, and $\tilde{\Sigma}_z$ are the coefficient matrices, and the calculation requires the state operation point $\boldsymbol{\delta}(z+i|z)(i = 1, 2, \ldots, N_y-1)$. However, in practical application, the information of the future state working point of the system is often unavailable at time z, and the current operating state $\boldsymbol{\delta}(z)$ has to be employed instead of $\boldsymbol{\delta}(z+i|z)$ for calculation. Based on model (11), a local linearization model can be obtained, and can be used to design the following MPC algorithm.

According to equations (10) - (16), the model prediction output can be obtained in the following form

$$\begin{cases} \tilde{\mathbf{Y}}(z) = \mathbf{W}_z\tilde{\mathbf{U}}(z) + \mathbf{Y}_{\mathbf{0}}(z) \\ \mathbf{W}_z = \tilde{\mathbf{F}}\tilde{\mathbf{E}}_z \\ \mathbf{Y}_{\mathbf{0}}(z) = \tilde{\mathbf{F}}\tilde{\mathbf{D}}_z\mathbf{X}(z) + \tilde{\mathbf{F}}\tilde{\Sigma}_z\tilde{\boldsymbol{\Psi}}_z \end{cases} \tag{17}$$

where $\tilde{\mathbf{Y}}(z)$ represents the model prediction output vector, $\tilde{\mathbf{U}}(z)$ represents the model prediction control vector, and $\mathbf{W}_z$ is the coefficient matrix. The desired output sequence $\tilde{\mathbf{Y}}_r(z)$

and control increment sequence $\Delta\tilde{\mathbf{U}}(z)$ are defined below

$$\begin{cases} \Delta\tilde{\mathbf{U}}(z) = [\Delta\mathbf{u}(z)^{\mathrm{T}}\Delta\mathbf{u}(z+1)^{\mathrm{T}}\cdots\Delta\mathbf{u}(z+N_u-1)^{\mathrm{T}}]^{\mathrm{T}} \\ \tilde{\mathbf{Y}}_r(z) = [\mathbf{y}_r(z+1)^{\mathrm{T}}\mathbf{y}_r(z+2)^{\mathrm{T}}\cdots\mathbf{y}_r(z+N_y)^{\mathrm{T}}]^{\mathrm{T}} \end{cases} \tag{18}$$

where $\Delta\mathbf{u}(z) = \mathbf{u}(z) - \mathbf{u}(z-1)$. The optimization function of the MPC is designed as follows

$$\begin{cases} \min_{\tilde{\mathbf{U}}(z)} J = \left\|\tilde{\mathbf{Y}}(z) - \tilde{\mathbf{Y}}_r(z)\right\|_{\mathrm{Q}_1}^2 + \left\|\tilde{\mathbf{U}}(z)\right\|_{\mathbf{S}_1}^2 + \left\|\Delta\tilde{\mathbf{U}}(z)\right\|_{\mathbf{S}_2}^2 \\ s.t.\mathbf{Y}_{\min} \le \tilde{\mathbf{Y}}(z) \le \mathbf{Y}_{\max}, \mathbf{U}_{\min} \le \tilde{\mathbf{U}}(z) \le \mathbf{U}_{\max}, \\ \Delta\mathbf{U}_{\min} \le \Delta\tilde{\mathbf{U}}(z) \le \Delta\mathbf{U}_{\max} \end{cases} \tag{19}$$

$$\tilde{\mathbf{D}}_z = \begin{bmatrix} \coprod_{i=0}^{0}\mathbf{D}_{z+i} \\ \coprod_{i=0}^{1}\mathbf{D}_{z+i} \\ \vdots \\ \coprod_{i=0}^{N_y-1}\mathbf{D}_{z+i} \end{bmatrix}, \quad \coprod_{i=j}^{p}\mathbf{D}_{z+i} = \begin{cases} \mathbf{D}_{z+p}\mathbf{D}_{z+p-1}\cdots\mathbf{D}_{z+j}, & j \le p \\ \mathbf{I}, & j > p \end{cases}$$

$$\tilde{\Sigma}_z = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \coprod_{i=1}^{1}\mathbf{D}_{z+i} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \coprod_{i=1}^{2}\mathbf{D}_{z+i} & \coprod_{i=2}^{2}\mathbf{D}_{z+i} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \coprod_{i=1}^{N_y-1}\mathbf{D}_{z+i} & \coprod_{i=2}^{N_y-1}\mathbf{D}_{z+i} & \cdots & \coprod_{i=N_y-1}^{N_y-1}\mathbf{D}_{z+i} & \mathbf{I} \end{bmatrix} \tag{15}$$

$$\tilde{\mathbf{E}}_z = \begin{bmatrix} \mathbf{E}_z & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \left(\coprod_{i=1}^{1}\mathbf{D}_{z+i}\right)\mathbf{E}_z & \mathbf{E}_{z+1} & \mathbf{0} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \left(\coprod_{i=1}^{N_u-1}\mathbf{D}_{z+i}\right)\mathbf{E}_z & \left(\coprod_{i=2}^{N_u-1}\mathbf{D}_{z+i}\right)\mathbf{E}_{z+1} & \cdots & \left(\coprod_{i=N_u-1}^{N_u-1}\mathbf{D}_{z+i}\right)\mathbf{E}_{z+N_u-2} & \mathbf{E}_{z+N_u-1} \\ \left(\coprod_{i=1}^{N_u}\mathbf{D}_{z+i}\right)\mathbf{E}_z & \left(\coprod_{i=2}^{N_u}\mathbf{D}_{z+i}\right)\mathbf{E}_{z+1} & \cdots & \left(\coprod_{i=N_u-1}^{N_u}\mathbf{D}_{z+i}\right)\mathbf{E}_{z+N_u-2} & \sum_{j=N_u-1}^{N_u}\left(\coprod_{i=j+1}^{N_u}\mathbf{D}_{z+i}\right)\mathbf{E}_{z+j} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \left(\coprod_{i=1}^{N_y-1}\mathbf{D}_{z+i}\right)\mathbf{E}_z & \left(\coprod_{i=2}^{N_y-1}\mathbf{D}_{z+i}\right)\mathbf{E}_{z+1} & \cdots & \left(\coprod_{i=N_u-1}^{N_y-1}\mathbf{D}_{z+i}\right)\mathbf{E}_{z+N_u-2} & \sum_{j=N_u-1}^{N_y-1}\left(\coprod_{i=j+1}^{N_y-1}\mathbf{D}_{z+i}\right)\mathbf{E}_{z+j} \end{bmatrix}$$

$$\tilde{\mathbf{F}} = \begin{bmatrix} \mathbf{F} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{F} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F} \end{bmatrix} \tag{16}$$

where $\|\mathbf{X}\|_{\lambda}^{2} = \mathbf{X}^{\mathrm{T}}\lambda\mathbf{X}$; Q_1, $\mathbf{S}_1$ and $\mathbf{S}_2$ are the weight coefficient matrices. By substituting Eq. (17) and Eq. (18) into Eq. (19) and removing the constant terms, the objective function (19) can be converted to the following QP problem, which can be easily solved online.

$$\begin{cases} \min_{\tilde{\mathbf{U}}(z)} \tilde{J} = \frac{1}{2}\tilde{\mathbf{U}}(z)^{\mathrm{T}}\left[\mathbf{W}_z^{\mathrm{T}}\mathbf{Q}_1\mathbf{W}_z + \mathbf{S}_1 + \mathbf{L}^{-T}\mathbf{S}_2\mathbf{L}^{-1}\right]\tilde{\mathbf{U}}(z) \\ + \left[\mathbf{Y_0}(z)^{\mathrm{T}}\mathbf{Q}_1\mathbf{W}_z - \mathbf{Y}_r(z)^{\mathrm{T}}\mathbf{Q}_1\mathbf{W}_z \right. \\ \qquad \left. -\mathbf{U}_0(z-1)^{\mathrm{T}}\mathbf{L}^{-T}\mathbf{S}_2\mathbf{L}^{-1}\right]\tilde{\mathbf{U}}(z) \\ s.t. \begin{bmatrix} \mathbf{W}_z \\ -\mathbf{W}_z \end{bmatrix}\tilde{\mathbf{U}}(z) \le \begin{bmatrix} \mathbf{Y}_{\max} - \mathbf{Y}_0(z) \\ -\mathbf{Y}_{\min} + \mathbf{Y}_0(z) \end{bmatrix}, \\ \mathbf{U}_{\min} \le \tilde{\mathbf{U}}(z) \le \mathbf{U}_{\max}, \\ \mathbf{U}_0(z-1) + \mathbf{L}\Delta\mathbf{U}_{\min} \le \tilde{\mathbf{U}}(z) \le \mathbf{U}_0(z-1) + \mathbf{L}\Delta\mathbf{U}_{\max}. \end{cases} \tag{20}$$

where

$$\begin{cases} \tilde{\mathbf{U}}(z) = \mathbf{U}_0(z-1) + \mathbf{L}\Delta\tilde{\mathbf{U}}(z), \\ \mathbf{U}_0(z-1) = [\,\mathbf{U}(z-1)^{\mathrm{T}}\ \mathbf{U}(z-1)^{\mathrm{T}} \cdots \mathbf{U}(z-1)^{\mathrm{T}}\,]^{\mathrm{T}} \\ \mathbf{L} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & \mathbf{I} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \end{bmatrix} \end{cases} \tag{21}$$

Using QP can solve the convex optimization problem (20) with constraints online. Additionally, if there is a reasonable solution to equation (20) in each sampling period, the stability of the MPC will be guaranteed [15], [17]. If there is no solution to the QP problem (20) in a certain control cycle, from a practical perspective, we can use the feasible solution obtained from the previous cycle to implement control.

In actual control, only the first element in the optimized optimal control sequence is used for control. The actual output at the next moment is observed, and the feedback correction is performed. Then the online rolling optimization is carried out again to effectively avoid environmental interference and the incompatibility between the model and the nonlinear system

According to the above predictive controller design process, the SD-ARX model based on deep learning has a specific structure that is easy to be locally linearized, so it is very convenient to develop a predictive control algorithm. This is different from the MPC based on the general nonlinear neural network model, which needs to calculate the higher-order derivative of the model online to generate the quadratic approximate local linearization model.

V. REAL-TIME CONTROL EXPERIMENTS

This section takes the water tank plant (WTP) commonly used in the process control equipment as the experimental object, uses the data-driven modeling method to build the object model, designs the predictive control algorithm, and carries out real-time control experiments on the WTP.

A. WATER TANK PLANT

The WTP is a two-input and two-output experimental equipment with strong coupling and nonlinearity, as shown in Fig. 6. Water from water storage tank 3 enters water tank 1 and 2 through the pump. The input flow of water tank 1 and 2 is controlled by adjusting the opening of electric valves EV1 and EV2, the output flow is controlled by proportional valves V1 and V2, and the liquid level height is measured by static pressure sensors LV1 and LV2, respectively. The modeling and real-time control experiments are carried out on the WTP, and its deep learning-based SD-ARX model is designed as follows.

$$\begin{cases} \mathbf{Y}(z) = \sum_{i=1}^{n_y} \mathbf{A}_{i,z-1}\mathbf{Y}(z-i) \\ + \sum_{j=n_d}^{n_u+n_d-1} \mathbf{B}_{j,z-1}\mathbf{U}(z-j) + \boldsymbol{\psi}_{0,z-1} + \Gamma(z) \\ \mathbf{A}_{i,z-1} = \begin{bmatrix} \alpha_{i,z-1}^{11}\alpha_{i,z-1}^{12} \\ a_{i,z-1}^{21}\alpha_{i,z-1}^{22} \end{bmatrix}, \mathbf{B}_{j,z-1} \\ = \begin{bmatrix} \beta_{j,z-1}^{11}\beta_{j,z-1}^{12} \\ \beta_{j,z-1}^{21}\beta_{j,z-1}^{22} \end{bmatrix}, \boldsymbol{\psi}_{0,z-1} = \begin{bmatrix} \psi_{0,z-1}^{1} \\ \psi_{0,z-1}^{2} \end{bmatrix} \end{cases} \tag{22}$$

where $\mathbf{Y}(z) = [y_1(z), y_2(z)]^{\mathrm{T}}$ are the fluid levels of tank 1 and 2; $\mathbf{U}(z) = [u_1(z), u_2(z)]^{\mathrm{T}}$ are the electric valve openings (0% $\sim$ 100%); $\{\mathbf{A}_{i,z-1}|i = 1, \ldots, n_y\}$, $\boldsymbol{\psi}_{0,z-1}$, and $\{\mathbf{B}_{j,z-1}|j = n_d, \ldots, n_u + n_d - 1\}$ are the state-dependent functional coefficients, which are from models (6)-(8). In this experiment, we set $\boldsymbol{\delta}(z-1) = [\mathbf{Y}(z-1)^{\mathrm{T}}, \ldots, \mathbf{Y}(z-d)^{\mathrm{T}}]^{\mathrm{T}}$, which indicates the state of the operating point, because the change of the liquid level is the main factor that causes the nonlinear change of the characteristics of the WTP.

B. MODEL ESTIMATION

In order to obtain the identification data including all dynamic modes of the object to estimate an accurate model, we first use the PID controllers to control the water levels of the water tanks, so that the input and output signals of the system can change within the full operating range as much as possible, and obtain the input and output data suitable for system identification. The system sampling period is 2 seconds, and the observation data are depicted in Fig. 7.

The sample data are divided into a training set and a test set. The first 3500 data of the sample data are used to estimate the model, and the last 1000 data are used to verify the modeling results. For the deep learning based SD-ARX model (22), we use the adaptive momentum (Adam) stochastic optimization algorithm to optimize the parameters of the model. It can dynamically change the learning rate of each parameter by calculating the moment estimation of the gradient to obtain a better modeling effect, and it has higher computational efficiency than other stochastic optimization

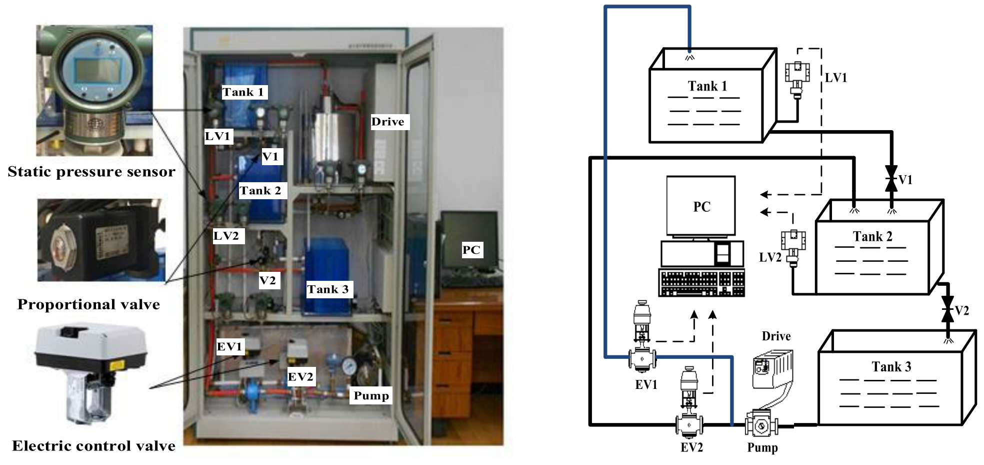


FIGURE 6. Water tank system.

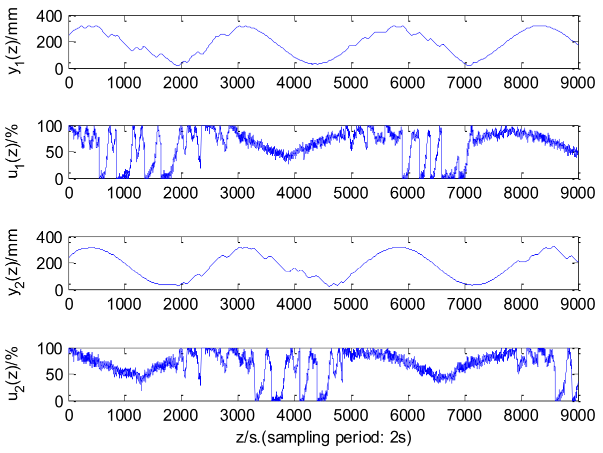


FIGURE 7. Identification data for WTP.

algorithms. The activation function of the full connection layer of the model is the Tanh function, which makes the model have faster convergence speed. For RNN in the model, the hidden layer activation function of RNN is the Sigmoid function to avoid divergence of the parameter training process. For CNN in the model, its convolution activation function is the Relu function, which can reduce the problem of gradient disappearing or exploding when training the model. The average pool is selected as the pool function, which is more conducive to transferring information to the next module for feature extraction and reducing dimensions. The kernel sizes of the convolution and pooling layers are 3 and 4, respectively. To estimate the model, first set the structure and super parameters of the model, calculate the prediction output sequence and mean square error (MSE) based on the model, and then update the parameters of the mode through the backpropagation algorithm until the MSE is minimized. Finally, compare the MSE under different model structures, and select the model with the minimum MSE, as shown in Table 1, for real-time control experiments.

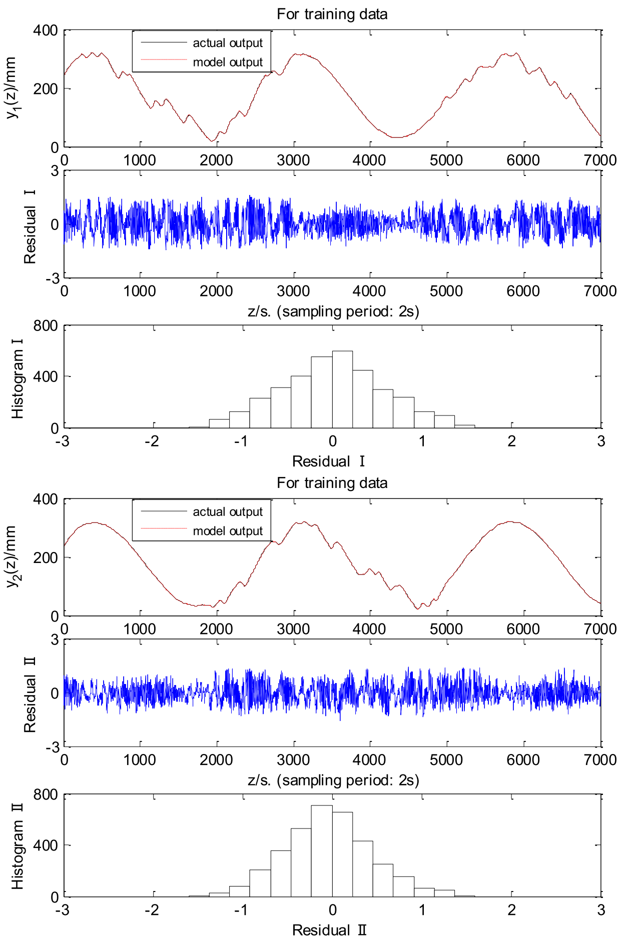


FIGURE 8. Training result of the RNN-CNN-ARX model.

In order to easily use the estimated model to design MPC, the collected raw data is directly used for modeling, and the measurement noise impact will be reflected in the modeling residual. Taking the RNN-CNN-ARX modeling as an example, the modeling results are shown in Figs. 8-9, from which

TABLE 1. The MSE of different models.

Model $(n_y, n_u, n_d, d, n, n_1)$	Number of nodes in each layer	MSE of training data $y_1(z)$	MSE of training data $y_2(z)$	MSE of test data $y_1(z)$	MSE of test data $y_2(z)$
ARX(18,18,6,/,/,/)	/	0.4910	0.3526	0.4176	0.4101
RBF-ARX(23,20,6,2,/,/)	2	0.4525	0.3158	0.3804	0.3712
CNN-ARX(19,22,6,2,0,3)	8,16,8	0.4277	0.2925	0.3562	0.3448
RNN-ARX(18,24,6,2,3,0)	16,32,16	0.4204	0.2865	0.3507	0.3389
RNN-CNN-ARX(18,21,6,2,3,3)	8,8,8,16,16,16	0.3903	0.2641	0.3216	0.3066

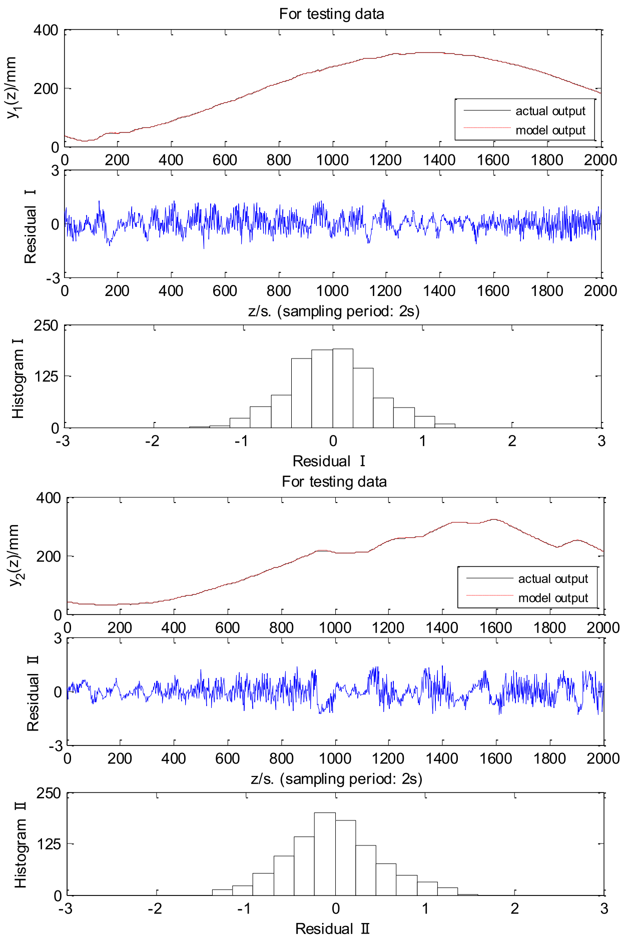


FIGURE 9. Test result of the RNN-CNN-ARX model.

one can see that the model output is very close to the actual output, and the residual for training and test data is very small. Table 1 gives the order and MSE of the estimated models. The results show that the MSE of the linear ARX model is the largest, because it cannot fully describe the dynamic characteristics of the nonlinear system. The MSE of the estimated RBF-ARX model is larger than that of the SD-ARX model based on deep learning, because the RBF neural network is a single-layer network and there is no feedback between the nodes of the neural network, so it cannot handle the sequence data of the nonlinear system well. The estimated RNN-ARX model has a slightly lower modeling error than the CNN-ARX model, which benefits from the memory function of RNN and thus maintains the advantage of dependence on sequence data. Among all models, the MSE of the estimated RNN-CNN-ARX model is the smallest, which can better capture the nonlinear dynamic characteristics of WTP. This is because the RNN-CNN-ARX model combines the ability of RNN to mine the temporal characteristics of sequence data and the ability of CNN to extract the spatial characteristics of data, and can skillfully handle the spatiotemporal correlation of WTP sequence data. By the way, in Table 1, one can see that the MSE of $y_1(z)$ for the test data is lower than that of $y_1(z)$ for the training data, this is because in the reference signal of $y_1(z)$ there are the sinusoidal signal and step signal in the training data, while in the reference signal of $y_1(z)$ there are not the step signal in the test data, as seen in Fig. 7, so the volatility of the test data is smaller than that of the training data.

C. REAL-TIME CONTROL EXPERIMENTS

In this section, we will show the real-time control results of using different model-based MPC algorithms on the WTP in Fig. 6. The control quantity calculation is implemented on MATLAB/Simulink platform. The nonlinear water tank system operating in different liquid level regions has obviously different dynamic characteristics. Therefore, the real-time control experiment is conducted in three different liquid level areas, namely, low liquid level area, middle liquid level area, and high liquid level area. Furthermore, for comparison, we also use the PID control, linear ARX model-based MPC (ARX-MPC), and the RBF-ARX model-based MPC (RBF-ARX-MPC) approaches to carry out the water level control experiments to show the advantages of the methods proposed in this paper.

In the real-time control experiments, the MPC parameters in equation (20) are selected as $\mathbf{S}_1 = \text{diag}[0.0001, 0.0001]$, $\mathbf{S}_2 = \text{diag}[0.80, 0.80]$, $\mathbf{Q}$=diag$[1, 1]$. The real-time control results of the water level of the WTP under different controllers are depicted in Figs 10-15, where $y_1(z)$ and $y_2(z)$ are the fluid level heights; $u_1(z)$ and $u_2(z)$ are the electric control valve openings $(0\% \sim 100\%)$; $y_r(z)$ is the reference trajectory, which is depicted by the pink dotted line. Additionally, Tables 2-4 give the control performance parameters of every controller at different liquid levels, including overshoot, peak time (PT), and settling time (ST), where $\uparrow$ represents the ascending step response and $\downarrow$ represents the descending step response.

TABLE 2. Control performance of the low fluid level area.

Control strategy	$y_1(z)$ PT (s) ↓/↑	$y_1(z)$ Overshoot (%) ↓/↑	$y_1(z)$ ST (s) ↓/↑	$y_2(z)$ PT (s) ↓/↑	$y_2(z)$ Overshoot (%) ↓/↑	$y_2(z)$ ST (s) ↓/↑
PID	280/162	39.4/13.3	692/616	258/170	39.5/14.2	814/782
ARX-MPC	278/194	36.2/8.6	478/256	256/194	34.2/9.6	732/376
RBF-ARX-MPC	272/208	30.2/5.6	420/242	242/186	29.6/7.2	534/234
RNN-ARX-MPC	262/244	23.4/4.2	350/172	238/254	24.4/5.1	368/244
CNN-ARX-MPC	264/218	25.0/4.6	352/172	240/268	23.2/5.8	340/248
RNN-CNN-ARX-MPC	258/238	18.4/3.4	348/170	230/230	17.0/3.8	334/162

TABLE 3. Control performance of the middle fluid level area.

Control strategy	$y_1(z)$ PT (s) ↑/↓	$y_1(z)$ Overshoot (%) ↑/↓	$y_1(z)$ ST (s) ↑/↓	$y_2(z)$ PT (s) ↑/↓	$y_2(z)$ Overshoot (%) ↑/↓	$y_2(z)$ ST (s) ↑/↓
PID	332/486	5.1/14.7	342/606	404/480	5.0/14.3	414/604
ARX-MPC	336/484	2.4/12.4	268/566	448/476	2.5/13.1	330/558
RBF-ARX-MPC	334/474	1.8/11.0	280/552	436/466	1.8/12.4	334/548
RNN-ARX-MPC	412/454	1.7/10.3	278/506	484/458	1.5/11.4	328/518
CNN-ARX-MPC	374/456	1.7/10.6	276/528	440/466	1.7/11.8	332/546
RNN-CNN-ARX-MPC	372/450	1.3/8.6	282/504	434/454	1.3/9.7	340/512

TABLE 4. Control performance of the high fluid level area.

Control strategy	$y_1(z)$ PT (s) ↑/↓	$y_1(z)$ Overshoot (%) ↑/↓	$y_1(z)$ ST (s) ↑/↓	$y_2(z)$ PT (s) ↑/↓	$y_2(z)$ Overshoot (%) ↑/↓	$y_2(z)$ ST (s) ↑/↓
PID	168/222	14.0/44.0	776/726	190/228	12.3/43.8	942/958
ARX-MPC	188/218	9.0/36.0	532/498	216/226	8.2/37.2	546/748
RBF-ARX-MPC	176/214	7.4/28.4	456/330	206/220	6.0/30.0	322/526
RNN-ARX-MPC	288/208	5.6/22.6	312/272	274/214	5.2/24.8	286/292
CNN-ARX-MPC	230/210	5.0/23.6	236/282	262/216	5.6/26.0	280/294
RNN-CNN-ARX-MPC	224/202	3.6/16.6	170/270	270/208	4.2/16.0	176/288

1) REAL-TIME CONTROL IN LOW FLUID LEVEL AREA

Figs. 10-11 and Table 2 show the real-time control results in the low fluid level area (50 ∼ 100mm), which illustrate that the control performance of RNN-CNN-ARX-MPC is better than that of other control algorithms. In the low liquid level area, for the liquid level $y_1(z)$ and $y_2(z)$ under the control of RNN-CNN-ARX-MPC, in the descending step response, its PT is 258s and 230s, the overshoot is 18.4% and 17%, the ST is 348s and 334s, and in the ascending step response, its overshoot is 3.4% and 3.8%, the ST is 170s and 162s, which are all better than the performance indicators of other controllers. In particular, the overshoot of RNN-CNN-ARX-MPC in the descending step response is much smaller than that of other controllers. This is because the RNN-CNN-ARX model combines the excellent temporal feature extraction ability of RNN and the spatial feature extraction ability of CNN, which can well represent the nonlinear dynamic behavior of WTP, thus achieving excellent control performance.

2) REAL-TIME CONTROL IN MIDDLE FLUID LEVEL AREA

Figs. 12-13 and Table 3 show the real-time control results in the middle fluid level area (100 ∼ 250 mm), from which it can be seen that the PID controller is inferior to other controllers in control performance, and its overshoot and ST are the largest. Compared with the ARX-MPC, the RBF-ARX-MPC has better control performance, and the latter has a smaller overshoot and PT. Nevertheless, the RBF-ARX-MPC is inferior to the RNN-ARX-MPC, CNN-ARX-MPC, and RNN-CNN-ARX-MPC in terms of overshoot and ST. This is because RBF belongs to the shallow neural network, and

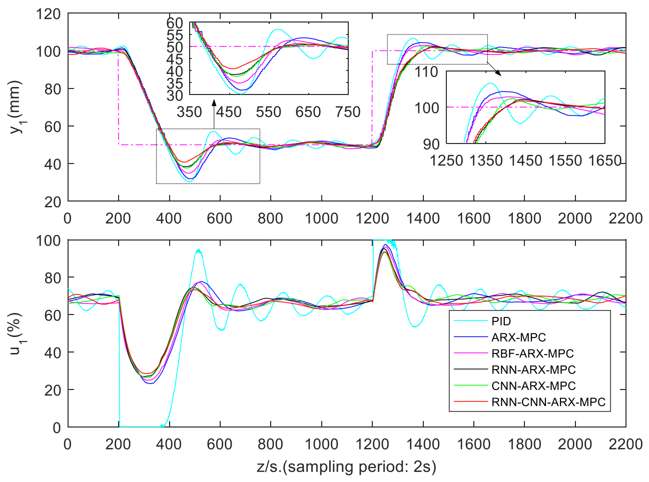


FIGURE 10. Step response results in low fluid level area (y_1 and u_1).

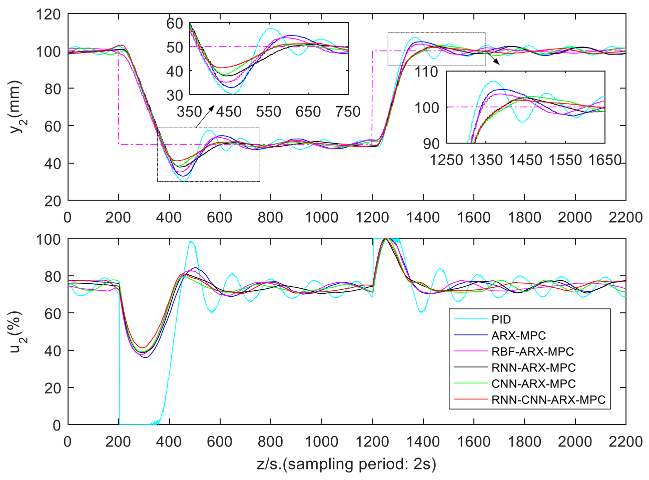


FIGURE 11. Step response results in low fluid level area (y_2 and u_2).

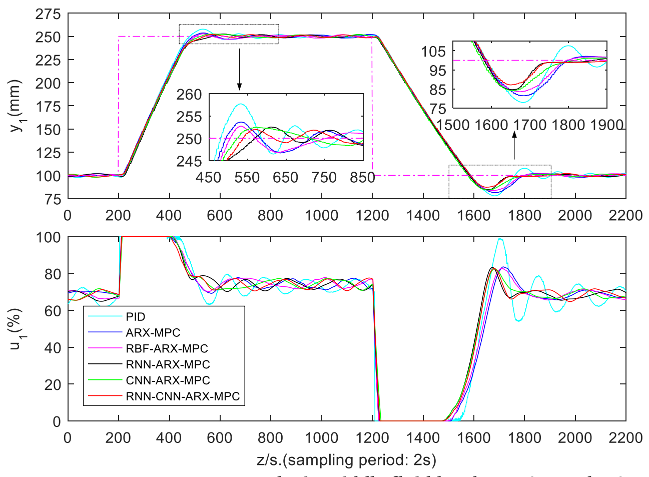


FIGURE 12. Step response results in middle fluid level area (y_1 and u_1).

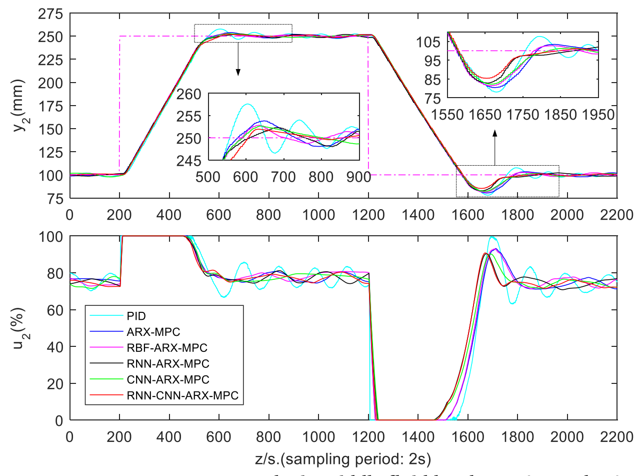


FIGURE 13. Step response results in middle fluid level area (y_2 and u_2).

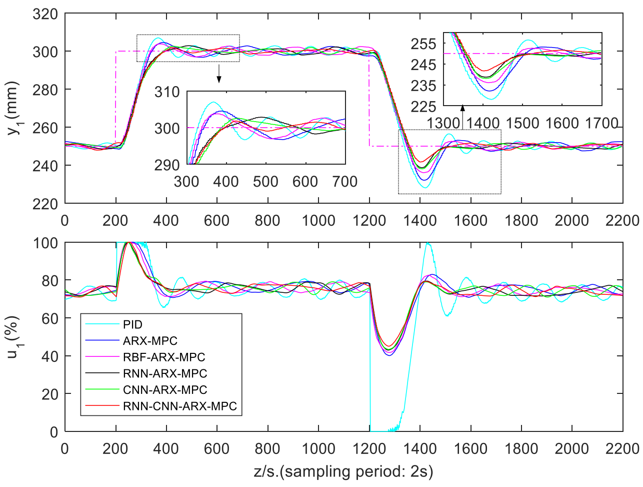


FIGURE 14. Step response results in high fluid level area (y_1 and u_1).

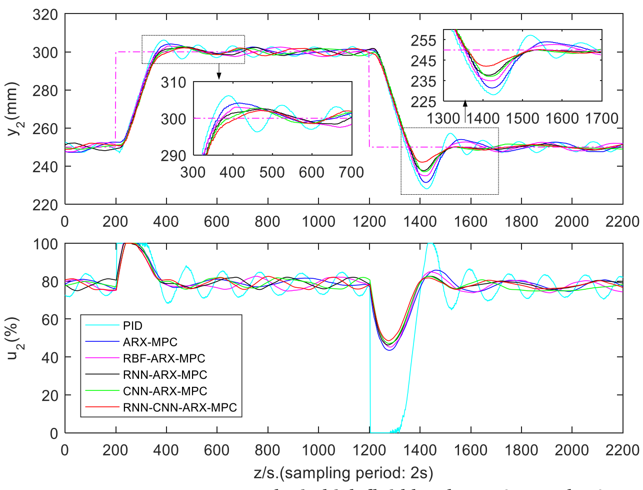


FIGURE 15. Step response results in high fluid level area (y_2 and u_2).

there is no information feedback between nodes of each layer, so the RBF-ARX modeling accuracy and the model-based predictive control effect are not as good as the deep learning-based SD-ARX. In addition, the overall control performance of RNN-ARX-MPC is slightly higher than that of the CNN-ARX-MPC, because RNN has a memory function and can maintain the correlation in sequence data, so it has advantages in sequence data modeling of nonlinear systems. Among all controllers, the RNN-CNN-ARX-MPC has the best control performance, with the lowest PT, ST, and overshoot in the descending step response. On the other hand, the overshoot of its rising step response is the smallest of all algorithms, which enables the liquid level of the WTP to follow the reference trajectory well.

3) REAL-TIME CONTROL IN HIGH FLUID LEVEL AREA

Figs. 14-15 and Table 4 shows the outcomes of the real-time control experiments under different controllers in the high fluid level area ($250 \sim 300$ mm). From the results, it is clear that the PID controller has the worst control performance among all controllers, and its overshoot and ST are the largest. Due to the limited ability of the linear ARX model to describe the nonlinear characteristics of WTP, the control performance of ARX-MPC is also poor. The comprehensive control performance of RNN-ARX-MPC is better than that of ARX-MPC and RBF-ARX-MPC, while the CNN-ARX-MPC is not as good as the RNN-CNN-ARX-MPC in the high fluid level area. The RNN-CNN-ARX-MPC has the best control performance, its overshoot, and ST are the smallest, especially its overshoot in the descending step response, which is greatly reduced compared with the overshoot under the control of other controllers. Because the RNN-CNN-ARX model has good spatiotemporal feature extraction ability, it can well describe the nonlinear dynamic characteristics of WTP.

Although the PID control and ARX-MPC can track the reference trajectory of liquid level, their control performance is quite poor due to the strong nonlinearity of WTP devices and the local adaptability of these controllers, which limits the scope of application of PID control and ARX-MPC. On the whole, the comprehensive control performance of the RBF-ARX-MPC is not as good as that of the RNN-ARX-MPC, CNN-ARX-MPC, and RNN-CNN-ARX-MPC, since the RBF neural network belongs to a single-layer network, its nonlinear description ability is not as good as that of the SD-ARX model based on multi-layer deep learning neural network. In addition, the RNN-ARX-MPC benefits from the memory function of RNN, which can maintain the dependency on sequence data, so its control performance is slightly better than that of the CNN-ARX-MPC. The RNN-CNN-ARX-MPC performs better in almost all cases, especially in the descending step response of low and high liquid level areas. In these areas, the strong coupling and nonlinearity of WTP lead to a large overshoot of all controllers, but the overshoot of RNN-CNN-ARX-MPC is far less than that of other controllers. This is mainly because the RNN-CNN-ARX model combines the advantages of RNN's strong temporal feature extraction ability and CNN's strong local spatial feature extraction ability, so it can well extract the nonlinear dynamic characteristics of WTP.

VI. CONCLUSION

In this paper, two types of deep learning models were used to fit the state-dependent functional regression coefficient of the SD-ARX model, RNN-ARX model, CNN-ARX model, and RNN-CNN-ARX model were constructed to describe the dynamic characteristics of nonlinear systems, and the three models based MPCs were designed. These models have the characteristics of local linearity and global nonlinearity, which is conducive to the design of predictive controllers for nonlinear systems. The real-time control experiment results on the WTP showed that the three predictive control algorithms proposed in this paper, namely RNN-ARX-MPC, CNN-ARX-MPC, and RNN-CNN-ARX-MPC, can better realize the tracking control of water tank level compared with other control strategies based on non-deep learning models, such as PID, ARX-MPC, and RBF-ARX-MPC. This is because the deep learning network has strong data feature extraction ability and can well express the nonlinear dynamic characteristics of WTP. The real-time control results also showed that the control performance of RNN-CNN-ARX-MPC is superior to the other two MPCs based on the deep learning model, namely RNN-ARX-MPC and CNN-ARX-MPC. Because the RNN-CNN-ARX model integrated the temporal feature extraction ability of RNN and the spatial feature extraction ability of CNN, it can better describe the multidimensional spatiotemporal dynamic characteristics of nonlinear systems.

However, the model built with RNN is prone to the problem of gradient vanishing or explosion when optimizing the model parameters, and may not be able to handle the long-term sequence correlation well. In future research, we will consider further optimizing the model structure, changing the RNN structure into long and short-term memory (LSTM) network, gated recurrent unit (GRU), or other relevant neural network structure, to improve the model's ability to describe the dynamic characteristics of nonlinear systems.

REFERENCES

[1] O. Palma-Flores and L. A. Ricardez-Sandoval, "Simultaneous design and nonlinear model predictive control under uncertainty: A back-off approach," *J. Process Control*, vol. 110, pp. 45–58, Feb. 2022.

[2] F. Grimm, P. Kolahian, Z. Zhang, and M. Baghdadi, "A sphere decoding algorithm for multistep sequential model-predictive control," *IEEE Trans. Ind. Appl.*, vol. 57, no. 3, pp. 2931–2940, May 2021.

[3] J. Kwapien and S. Drozdz, "Physical approach to complex systems," *Phys. Rep.*, vol. 515, nos. 3–4, pp. 115–226, Jun. 2012.

[4] P. Pareek and A. Verma, "Piecewise linearization of quadratic branch flow limits by irregular polygon," *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 7301–7304, Nov. 2018.

[5] M. Lawrynczuk, "Nonlinear predictive control of a boiler-turbine unit: A state-space approach with successive on-line model linearisation and quadratic optimisation," *ISA Trans.*, vol. 67, pp. 476–495, Mar. 2017.

[6] Z. Nie, F. Gao, and C. Yan, "A multi-timescale bilinear model for optimization and control of HVAC systems with consistency," *Energies*, vol. 14, no. 2, pp. 400–412, Jan. 2021.

[7] J. A. Becerra, "A bivariate Volterra series model for the design of power amplifier digital predistorters," *Sensors*, vol. 21, no. 17, pp. 5897–5909, Sep. 2021.

[8] T. A. Tutunji, "Parametric system identification using neural networks," *Appl. Soft Comput.*, vol. 47, pp. 251–261, Oct. 2016.

[9] S. Mete, S. Ozer, and H. Zorlu, "System identification using Hammerstein model optimized with differential evolution algorithm," *AEU-Int. J. Electron. Commun.*, vol. 70, no. 12, pp. 1667–1675, Dec. 2016.

[10] I. Aliskan, "Optimized inverse nonlinear function-based Wiener model predictive control for nonlinear systems," *Arabian J. Sci. Eng.*, vol. 46, no. 10, pp. 10217–10230, Oct. 2021.

[11] V. De Iuliis, G. Domenico Di Girolamo, F. Smarra, and A. D'Innocenzo, ''A comparison of classical identification and learning-based techniques for cyber-physical systems,'' in *Proc. 29th Medit. Conf. Control Autom. (MED)*, Jun. 2021, pp. 179–185. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9480333

[12] F. Smarra, G. D. Di Girolamo, V. De Iuliis, A. Jain, R. Mangharam, and A. D'Innocenzo, ''Data-driven switching modeling for MPC using regression trees and random forests,'' *Nonlinear Analy., Hybrid Syst.*, vol. 36, May 2020, Art. no. 100882.

[13] H. Peng, T. Ozaki, V. Haggan-Ozaki, and Y. Toyoda, ''A parameter optimization method for radial basis function type models,'' *IEEE Trans. Neural Netw.*, vol. 14, no. 2, pp. 432–438, Mar. 2003.

[14] F. Zhou, H. Peng, Y. Qin, X. Zeng, W. Xie, and J. Wu, ''RBF-ARX model-based MPC strategies with application to a water tank system,'' *J. Process Control*, vol. 34, pp. 97–116, Oct. 2015.

[15] H. Peng, K. Nakano, and H. Shioya, ''Nonlinear predictive control using neural nets-based local linearization ARX model—Stability and industrial application,'' *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 1, pp. 130–143, Jan. 2007.

[16] T. Kang, H. Peng, F. Zhou, X. Tian, and X. Peng, ''Robust predictive control of coupled water tank plant,'' *Int. J. Speech Technol.*, vol. 51, no. 8, pp. 5726–5744, Aug. 2021.

[17] X. Tian, H. Peng, X. Zeng, F. Zhou, W. Xu, and X. Peng, ''A modeling and predictive control approach to linear two-stage inverted pendulum based on RBF-ARX model,'' *Int. J. Control*, vol. 94, no. 2, pp. 357–369, Feb. 2021.

[18] H. Peng, J. Wu, G. Inoussa, Q. Deng, and K. Nakano, ''Nonlinear system modeling and predictive control using the RBF nets-based quasi-linear ARX model,'' *Control Eng. Pract.*, vol. 17, no. 1, pp. 59–66, Jan. 2009.

[19] M. Gan, H.-X. Li, and H. Peng, ''A variable projection approach for efficient estimation of RBF-ARX model,'' *IEEE Trans. Cybern.*, vol. 45, no. 3, pp. 476–485, Mar. 2015.

[20] G. Inoussa, H. Peng, and J. Wu, ''Nonlinear time series modeling and prediction using functional weights wavelet neural network-based state-dependent AR model,'' *Neurocomputing*, vol. 86, pp. 59–74, Jun. 2012.

[21] W. Xu, H. Peng, X. Tian, and X. Peng, ''DBN based SD-ARX model for nonlinear time series prediction and analysis,'' *Int. J. Speech Technol.*, vol. 50, no. 12, pp. 4586–4601, Dec. 2020.

[22] J. Schmidhuber, ''Deep learning in neural networks: An overview,'' *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[23] R. Mu and X. Zeng, ''A review of deep learning research,'' *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 4, pp. 1738–1764, 2019.

[24] S. K. Mahata, D. Das, and S. Bandyopadhyay, ''MTIL2017: Machine translation using recurrent neural network on statistical machine translation,'' *J. Intell. Syst.*, vol. 28, no. 3, pp. 447–453, Jul. 2019.

[25] A. Esan, J. Oladosu, C. Oyeleye, I. Adeyanju, O. Olaniyan, N. Okomba, B. Omodunbi, and O. Adanigbo, ''Development of a recurrent neural network model for English to Yorùbá machine translation,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 5, pp. 602–609, May 2020.

[26] C. R. Aydin and T. Gungor, ''Combination of recursive and recurrent neural networks for aspect-based sentiment analysis using inter-aspect relations,'' *IEEE Access*, vol. 8, pp. 77820–77832, 2020.

[27] J. A. Laura, G. O. Masi, and L. Argerich, ''From imitation to prediction, data compression vs recurrent neural networks for natural language processing,'' *Inteligencia Artif.*, vol. 21, no. 61, pp. 30–46, Mar. 2018.

[28] S. Hashemnia, L. Grasse, S. Soni, and M. S. Tata, ''Human EEG and recurrent neural networks exhibit common temporal dynamics during speech recognition,'' *Frontiers Syst. Neurosci.*, vol. 15, Jul. 2021, Art. no. 617605.

[29] J. Wang, ''Speech recognition in English cultural promotion via recurrent neural network,'' *Pers. Ubiquitous Comput.*, vol. 24, no. 2, pp. 237–246, Apr. 2020.

[30] J. Van Gompel, D. Spina, and C. Develder, ''Satellite based fault diagnosis of photovoltaic systems using recurrent neural networks,'' *Appl. Energy*, vol. 305, Jan. 2022, Art. no. 117874.

[31] H. Liu, J. Zhou, Y. Zheng, W. Jiang, and Y. Zhang, ''Fault diagnosis of rolling bearings with recurrent neural network-based autoencoders,'' *ISA Trans.*, vol. 77, pp. 167–178, Jun. 2018.

[32] M. S. Alhajeri, Z. Wu, D. Rincon, F. Albalawi, and P. D. Christofides, ''Machine-learning-based state estimation and predictive control of nonlinear processes,'' *Chem. Eng. Res. Des.*, vol. 167, pp. 268–280, Mar. 2021.

[33] B. Zhang, X. Sun, S. Liu, and X. Deng, ''Recurrent neural network-based model predictive control for multiple unmanned quadrotor formation flight,'' *Int. J. Aerosp. Eng.*, vol. 2019, pp. 1–18, Jun. 2019.

[34] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, ''Machine learning-based predictive control of nonlinear processes. Part I: Theory,'' *AIChE J.*, vol. 65, no. 11, p. 16729, Nov. 2019.

[35] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, ''Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation,'' *AIChE J.*, vol. 65, no. 11, p. 16734, Nov. 2019.

[36] Y. Chang, T. Tan, W. H. Lee, L. Chang, Y. Chen, K. Fan, and M. Alkhaleefah, ''Consolidated convolutional neural network for hyperspectral image classification,'' *Remote Sens.*, vol. 14, no. 7, pp. 1571–1586, Apr. 2022.

[37] A. Fakhrou, J. Kunhoth, and S. Al Maadeed, ''Smartphone-based food recognition system using multiple deep CNN models,'' *Multimedia Tools Appl.*, vol. 80, nos. 21–23, pp. 33011–33032, Sep. 2021.

[38] R. Dian, S. Li, and X. Kang, ''Regularizing hyperspectral and multispectral image fusion by CNN denoiser,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 1124–1135, Mar. 2021.

[39] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, ''Automatically designing CNN architectures using the genetic algorithm for image classification,'' *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020.

[40] Y. Guo, Y. Liu, E. M. Bakker, Y. Guo, and M. S. Lew, ''CNN-RNN: A large-scale hierarchical image classification framework,'' *Multimed. Tools Appl.*, vol. 77, no. 8, pp. 10251–10271, Apr. 2018.

[41] Y. You, C. Lu, W. Wang, and C.-K. Tang, ''Relative CNN-RNN: Learning relative atmospheric visibility from images,'' *IEEE Trans. Image Process.*, vol. 28, no. 1, pp. 45–55, Jan. 2019.

[42] B. Zhao, X. Li, X. Lu, and Z. Wang, ''A CNN–RNN architecture for multi-label weather recognition,'' *Neurocomputing*, vol. 322, pp. 47–57, Dec. 2018.

[43] M. E. Basiri, S. Nemati, M. Abdar, E. Cambria, and U. R. Acharya, ''ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis,'' *Future Gener. Comput. Syst.*, vol. 115, pp. 279–294, Feb. 2021.

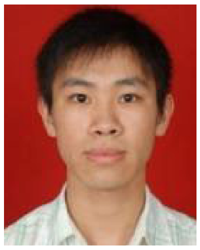
[44] D. Kollias and S. Zafeiriou, ''Exploiting multi-CNN features in CNN-RNN based dimensional emotion recognition on the OMG in-the-Wild dataset,'' *IEEE Trans. Affect. Comput.*, vol. 12, no. 3, pp. 595–606, Jul./Sep. 2021.

[45] L. Li, X. Zhu, Y. Hao, S. Wang, X. Gao, and Q. Huang, ''A hierarchical CNN-RNN approach for visual emotion classification,'' *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 15, no. 3, pp. 1–17, Nov. 2019.

[46] R. Geetha, T. Thilagam, and T. Padmavathy, ''Effective offline handwritten text recognition model based on a sequence-to-sequence approach with CNN–RNN networks,'' *Neural Comput. Appl.*, vol. 33, no. 17, pp. 10923–10934, Sep. 2021.

[47] H. Butt, M. R. Raza, M. J. Ramzan, M. J. Ali, and M. Haris, ''Attention-based CNN-RNN Arabic text recognition from natural scene images,'' *Forecasting*, vol. 3, no. 3, pp. 520–540, Jul. 2021.

[48] Y. Jiang, R. Qiao, Y. Zhu, and G. Wang, ''Data fusion of atmospheric ozone remote sensing LiDAR according to deep learning,'' *J. Supercomput.*, vol. 77, no. 7, pp. 6904–6919, Jul. 2021.



TIAO KANG received the M.Eng. degree in power engineering from the Hunan Institute of engineering, Xiangtan, China, in 2015. He is currently pursuing the Ph.D. degree in control science and engineering with Central South University, Changsha, China. His current research interests include complex systems modeling, optimization, and industrial process control.

HUI PENG received the B.Eng. and M.Eng. degrees in control science and engineering from Central South University, Changsha, China, in 1983 and 1986, respectively, and the Ph.D. degree in statistical science from The Graduate University for Advanced Studies, Japan, in 2003. He has been a Professor with Central South University, since 1998. His research interests include complex systems modeling, control and optimization, advanced control theory and intelligent automation systems, and industrial process control system development.

WENQUAN XU received the M.S. degree in control theory and control engineering from Hohai University, Nanjing, China, in 2011. He is currently pursuing the Ph.D. degree with the School of Automation, Central South University, Changsha, China. He is a Lecturer with the School of Electronic Engineering and Intelligent Manufacturing, Anqing Normal University, Anqing, China. His current research interests include deep learning, system identification, and nonlinear time series analysis.

YAPENG SUN received the bachelor's degree in computer science from Central South University, in 2003, and the master's degree in computer science from the Hunan University of Technology, in 2012. He is currently pursuing the Ph.D. degree in computer science with Central South University. After graduation, he was a Teacher with the School of Computer Science and Technology, Hunan University of Science and Technology. His current research interests include complex system modeling, optimization, and control.

XIAOYAN PENG received the B.S. and M.S. degrees in mechanical engineering and the Ph.D. degree in automatic control from Hunan University, Changsha, China, in 1986, 1989, and 2013, respectively. She is currently a Professor with the College of Mechanical and Vehicle Engineering, Hunan University. Her research interests include the control of mechatronic systems and the safety analysis of autonomous vehicles.