actuators

MDPI

*Article*

# LSTM-CNN Network-Based State-Dependent ARX Modeling and Predictive Control with Application to Water Tank System

**Tiao Kang [1,2], Hui Peng [1,*] and Xiaoyan Peng [3]**

[1] School of Automation, Central South University, Changsha 410083, China; 40102@hnie.edu.cn
[2] Engineering Training Center, Hunan Institute of Engineering, Xiangtan 411101, China
[3] College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China; xypeng@hnu.edu.cn
* Correspondence: huipeng@csu.edu.cn

**Abstract:** Industrial process control systems commonly exhibit features of time-varying behavior, strong coupling, and strong nonlinearity. Obtaining accurate mathematical models of these nonlinear systems and achieving satisfactory control performance is still a challenging task. In this paper, data-driven modeling techniques and deep learning methods are used to accurately capture a category of a smooth nonlinear system's spatiotemporal features. The operating point of these systems may change over time, and their nonlinear characteristics can be locally linearized. We use a fusion of the long short-term memory (LSTM) network and convolutional neural network (CNN) to fit the coefficients of the state-dependent AutoRegressive with the eXogenous variable (ARX) model to establish the LSTM-CNN-ARX model. Compared to other models, the hybrid LSTM-CNN-ARX model is more effective in capturing the nonlinear system's spatiotemporal characteristics due to its incorporation of the strengths of LSTM for learning temporal characteristics and CNN for capturing spatial characteristics. The model-based predictive control (MPC) strategy, namely LSTM-CNN-ARX-MPC, is developed by utilizing the model's local linear and global nonlinear features. The control comparison experiments conducted on a water tank system show the effectiveness of the developed models and MPC methods.

**Keywords:** LSTM-ARX model; MPC; water tank system; LSTM-CNN-ARX model



**Citation:** Kang, T.; Peng, H.; Peng, X. LSTM-CNN Network-Based State-Dependent ARX Modeling and Predictive Control with Application to Water Tank System. *Actuators* **2023**, *12*, 274. https://doi.org/10.3390/act12070274

Academic Editors: Guangtao Ran, Jian Liu, Yongbao Wu and Rathinasamy Sakthivel

Received: 1 June 2023
Revised: 5 July 2023
Accepted: 5 July 2023
Published: 6 July 2023



## 1. Introduction

MPC is an effective control method developed in industrial practice. It can adequately deal with the problems of multivariable, multi-constraint, strong coupling, and strong nonlinearity existing in actual industrial process objects; it has been a widespread in academia and industry and resulted in numerous theoretical research and application results [1,2]. The MPC method predicts the system's future behavior based on the controlled object's dynamic model to achieve the goal of optimal control. Therefore, the predictive model's capacity to describe the system has a substantial effect on the MPC controller's control performance. Due to the fact that some key parameters of complex nonlinear systems cannot be determined or obtained, it is difficult to obtain a physical model that accurately represents the system's nonlinear dynamic characteristics by establishing differential or difference equations via Newtonian mechanics analysis or the Lagrange energy method [3]. Therefore, data-driven identification is a highly effective modeling approach for realizing the MPC of nonlinear systems. This method uses input and output data to construct the system model and does not need to analyze the complex interrelationship between the various physical variables of the system [4,5].

When designing predictive control algorithms based on the identified nonlinear system models, using segmented linearization [6] or local linearization [7] models can simplify the controller's design but not adequately represent the complex system's nonlinear dynamic

characteristics, which will affect the controller's control performance. To address the accuracy loss issue caused by linearization models, direct use of nonlinear models such as bilinear models [8,9], Volterra series models [10,11], and neural network models [12,13] can provide a better description of the system's nonlinear dynamic characteristics. However, predictive control algorithms based on these nonlinear models need to solve non-convex optimization problems online, which may increase the computational burden and not guarantee feasible solutions. To address these issues, in recent years, many scholars have started studying combined models, including the Hammerstein model [14], Wiener model [15], Hammerstein-Wiener model [16,17], and SD-ARX model [18–25]. Among them, the SD-ARX model outperforms the Hammerstein and Wiener models in capturing the nonlinear system's dynamic properties. The SD-ARX model uses state-dependent coefficients to represent the system's nonlinear dynamic properties, and its local linear and global nonlinear structural characteristics make it easy to design MPC controllers. In recent years, SD-ARX models obtained by fitting model coefficients using RBF neural networks [18–23], deep belief networks [24], and wavelet neural networks [25] have been widely employed for complex industrial system modeling and control. However, these neural networks belong to feedforward neural networks, and their data information can only be transmitted forward. The connection between network nodes at each layer does not form a cycle, so their ability to describe nonlinear systems is limited.

With the fast advancement of AI technology, deep learning has attained success in numerous fields [26]. Its core idea is to learn the high-dimensional features of data via multi-level nonlinear transformation and a back-propagation algorithm. LSTM and CNN are popular deep learning neural networks, but their design objectives and application scenarios are different. LSTM is primarily employed for time series modeling, has strong memory ability and long-term dependence, and has achieved great success in natural language processing [27–29], speech recognition [30–32], time series prediction [33–36], etc. Because the LSTM neural network introduces a gating mechanism, it successfully solves the issue of gradient vanishing and exploding in a standard recurrent neural network (RNN), allowing it to process long sequence data more efficiently, which is conducive to nonlinear systems modeling. For example, Wu et al. [37] developed a dropout LSTM and co-teaching LSTM strategy for nonlinear systems. Terzi et al. [38] developed an MPC method based on the LSTM network and carried out numerical tests on the pH reactor simulator. Zarzycki et al. [39] developed a model prediction algorithm based on the LSTM network that has achieved good modeling accuracy and control effects by using online advanced tracking linearization. Although these modeling methods based on the LSTM neural network can properly learn the time dimension features in nonlinear system data, their ability to learn the spatial dimension features is limited, which affects the accuracy of system modeling. In addition, the validation of these methods has been proved by numerical simulation only and has not been applied to real objects.

The CNN can autonomously learn the spatial features of input data via convolution and pooling operations and is mainly used in computer vision [40–43]. Therefore, the composite neural network consisting of LSTM and CNN can effectively learn the spatiotemporal features of data and improve the accuracy of complex nonlinear modeling. Its research mainly focuses on emotion analysis [44–46], text classification [47–49], and electricity forecasting [50,51], but has not been found to be used for modeling and MPC of real industrial objects.

This article established the LSTM-ARX and LSTM-CNN-ARX models to represent a category of smooth nonlinear systems whose operating point may change over time and whose nonlinear characteristics can be locally linearized by using LSTM or/and CNN to fit the SD-ARX model's coefficients. Furthermore, according to the pseudo-linear structure of the SD-ARX model, which exhibits both local linearization and global nonlinearity, two model predictive controllers have been designed, namely LSTM-ARX-MPC and LSTM-CNN-ARX-MPC. To evaluate the performance of these models and control algorithms, a real-time control comparative study was conducted on the commonly used water tank

system in industrial process control. The outcomes showed that the developed models and MPC algorithms are effective and feasible. Particularly, the LSTM-CNN-ARX-MPC demonstrates excellent comprehensive control performance by leveraging the strengths of LSTM for learning temporal dimension characteristics and CNN for learning spatial dimension characteristics, enabling it to efficiently and accurately learn the nonlinear system's spatiotemporal characteristics from large volumes of data. This article's major contributions are summarized below.

(1) The LSTM-ARX and LSTM-CNN-ARX models are proposed to describe the system's nonlinear features.
(2) The predictive controller was developed using the model's pseudo-linear structure features.
(3) Control comparison experiments were conducted on the water tank system, which is a commonly used industrial process control device, to validate the efficiency of the developed models and control algorithms. To our knowledge, there are currently no reports on using deep learning algorithms for nonlinear system modeling and the real-time control of actual industrial equipment. This study demonstrates how to establish deep learning-related models for nonlinear systems, design the MPC algorithms, and achieve real-time control rather than only doing a numerical simulation as in the relevant literature.

The article's structure is as follows: Section 2 describes the related work. Section 3 studies three combination models. Section 4 designs the model-based predictive controllers. Section 5 presents the results of real-time control comparative experiments on the water tank system. Section 6 summarizes the research content.

## 2. Related Work

The SD-ARX, LSTM, and CNN models are summarized in this part.

### *2.1. SD-ARX Model*

Taylor expansion and local linearization models are often used to deal with nonlinear systems [52,53]. Using these ideas, Priestley [54] proposed the SD-AR model structure of a class of nonlinear time series and pointed out that the SD-AR model can fit nonlinear systems without any prior conditions. Peng et al. [22] extended the SD-AR model to the SD-ARX model. The nonlinear ARX model is used to represent a category of smooth nonlinear systems, as shown below:

$$\begin{cases} y(a) = \eta(\varepsilon(a-1)) + \theta(a) \\ \varepsilon(a-1) = [y(a-1)^{\mathrm{T}}, \ldots, y(a-s_y)^{\mathrm{T}}, u(a-1)^{\mathrm{T}}, \ldots, u(a-s_u)^{\mathrm{T}}]^{\mathrm{T}} \end{cases} \tag{1}$$

where $y(a)$ represents output, $u(a)$ represents input, $\theta(a)$ represents modeling error, $s_y$ and $s_u$ represent model orders. At an operating point $\varepsilon_0$, $\eta(\bullet)$ is expanded into the following Taylor polynomial

$$\begin{aligned} \eta(\varepsilon(a-1)) = \eta(\varepsilon_0) + \eta'(\varepsilon_0)^{\mathrm{T}}(\varepsilon(a-1) - \varepsilon_0) + \\ \tfrac{1}{2}(\varepsilon(a-1) - \varepsilon_0)^{\mathrm{T}}\eta''(\varepsilon_0)(\varepsilon(a-1) - \varepsilon_0) + \ldots + \gamma(\varepsilon(a-1)) \end{aligned} \tag{2}$$

Then, substituting Equation (2) into Equation (1) yields the SD-ARX model as follows.

$$\begin{cases} y(a) = \rho_0(\varepsilon(a-1)) + \sum\limits_{i=1}^{s_y} \rho_{y,i}(\varepsilon(a-1))y(a-i) + \sum\limits_{j=1}^{s_u} \rho_{u,j}(\varepsilon(a-1))u(a-j) + \theta(a) \\ \ell_0 = \eta(\varepsilon_0) - \eta'(\varepsilon_0)^{\mathrm{T}}\varepsilon_0 + \frac{1}{2}\varepsilon_0^{\mathrm{T}}\eta''(\varepsilon_0)\varepsilon_0 + \ldots \\ \ell_1(\varepsilon(a-1)) = \gamma(\varepsilon(a-1)) \\ \mathrm{T}_0 = \eta'(\varepsilon_0)^{\mathrm{T}} - \frac{1}{2}\varepsilon_0^{\mathrm{T}}\eta''(\varepsilon_0) - \frac{1}{2}\varepsilon_0^{\mathrm{T}}\eta''(\varepsilon_0)^{\mathrm{T}} + \ldots \\ \mathrm{T}_1(\varepsilon(a-1)) = \frac{1}{2}\varepsilon(a-1)^{\mathrm{T}}\eta''(\varepsilon_0) + \ldots \\ \ell_0 + \ell_1(\varepsilon(a-1)) = \rho_0(\varepsilon(a-1)) \\ \mathrm{T}_0 + \mathrm{T}_1(\varepsilon(a-1)) = \left[\rho_{y,1}(\varepsilon(a-1)), \ldots, \rho_{y,s_y}(\varepsilon(a-1)),\ \rho_{u,1}(\varepsilon(a-1)), \ldots, \rho_{u,s_u}(\varepsilon(a-1))\right] \end{cases} \tag{3}$$

where $\{\rho_{y,i}(\varepsilon(a-1))|i=1,\ldots,s_y\}$, $\{\rho_{u,j}(\varepsilon(a-1))|j=1,\ldots,s_u\}$ and $\rho_0(\varepsilon(a-1))$ represent the regression coefficients that can be obtained by fitting using neural networks, including BRF neural networks [18–23] and wavelet neural networks [24]. $\varepsilon(a-1)$ represents the operating state at time $a$, which may correspond to the system's input or/and output. When fixing $\varepsilon(a-1)$, Equation (3) is considered a locally linearized model. When $\varepsilon(a-1)$ varies with the system's operating point, Equation (3) represents a model capable of globally describing the system's nonlinear properties. This model has local linear and global nonlinear features, making it advantageous for MPC design. In this article, we use LSTM or/and CNN to approximate the model's regression coefficients and obtain a category of SD-ARX models that can effectively represent the system's nonlinear properties.

### *2.2. LSTM*

LSTM and GRU are commonly used RNN variants for handling time series data. In comparison to the GRU network, LSTM possesses stronger memory and can capture longer dependencies. which may be more effective in handling long data sequences and modeling intricate systems. LSTM networks can not only solve the problem that traditional RNNs cannot handle the dependence of long sequence data but also the issue that gradient exploding or vanishing is easy to occur with increasing training time and network layers of the neural networks. The LSTM network consists of multiple LSTM units, and its internal structure and chain structure are shown in Figure 1, with its core being the cell state **C**, represented by a horizontal line running through the entire cell at the top layer. It controls the addition or removal of information via gates. LSTM mainly consists of an input gate, a forget gate, and an output gate. The input gate controls which input information needs to be added to the current time step's storage unit. The forget gate controls which information needs to be forgotten in the storage unit of the previous time step. The output gate controls which information in the current time step's storage unit needs to be output to the next time step. LSTM introduces a gate mechanism to control information transmission, remember content that needs to be memorized for a long time, and forget unimportant information, making it particularly effective in handling and forecasting critical events with long intervals and temporal delays in sequential data. The formula for an LSTM unit is as follows.

$$\begin{cases} \mathbf{c}_l = \mathbf{f}_l \odot \mathbf{c}_{l-1} + \mathbf{i}_l \odot \sigma_5(\mathbf{x}_l\mathbf{W}_{cx} + \mathbf{h}_{l-1}\mathbf{W}_{ch} + \mathbf{b}_c) \\ \mathbf{h}_l = \mathbf{o}_l \odot \sigma_5(\mathbf{c}_l) \\ \mathbf{f}_l = \sigma_4\left(\mathbf{x}_l\mathbf{W}_{fx} + \mathbf{h}_{l-1}\mathbf{W}_{fh} + \mathbf{b}_f\right) \\ \mathbf{i}_l = \sigma_4(\mathbf{x}_l\mathbf{W}_{ix} + \mathbf{h}_{l-1}\mathbf{W}_{ih} + \mathbf{b}_i) \\ \mathbf{o}_l = \sigma_4(\mathbf{x}_l\mathbf{W}_{ox} + \mathbf{h}_{l-1}\mathbf{W}_{oh} + \mathbf{b}_o) \end{cases} \tag{4}$$

where $l$ represents the time step of the input sequence, referred to as the number of rounds; $\odot$ represents matrix dot multiplication; $\mathbf{c}_l$ and $\mathbf{h}_l$ denote cell state vector and hidden state vector, respectively; $\sigma_4(\bullet)$ and $\sigma_5(\bullet)$ denote the nonlinear activation functions; $\mathbf{x}_l$ represents input information; $\mathbf{f}_l$ represents the forget gate's output, which decides the amount of information from the preceding state $\mathbf{c}_{l-1}$ needs to be forgotten. The value of the output ranges from 0 to 1, with a smaller value indicating more information to be forgotten, 0 meaning total forgetting, and 1 meaning total retention; $\mathbf{i}_l$ indicates the input gate's output, which controls the current output state. Its output value is usually mapped to a range of 0 to 1 using the $\sigma_4(\bullet)$ function, indicating the degree of activation of the output state, with 1 indicating full activation and 0 indicating no activation; $\mathbf{o}_l$ represents the output gate's output, which decides the information to be output in this round;$\mathbf{W}_{cx}$, $\mathbf{W}_{ch}$, $\mathbf{W}_{fx}$, $\mathbf{W}_{fh}$, $\mathbf{W}_{ix}$, $\mathbf{W}_{ih}$, $\mathbf{W}_{ox}$, and $\mathbf{W}_{oh}$ denote weight coefficients; $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_o$, and $\mathbf{b}_c$ denote offsets.

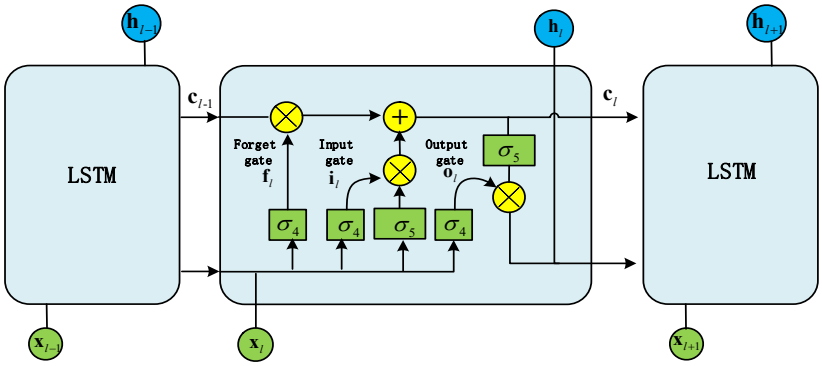



**Figure 1.** The schematic diagram of LSTM [32].

### *2.3. CNN*

CNN has outstanding spatial feature extraction capabilities. CNN can automatically learn the spatial characteristics of liquid level distribution and liquid pressure in water tank systems. For example, CNN can learn the liquid level distribution at different positions, that is, the changes in water surface height. Meanwhile, by learning the pressure changes hidden in the input and output data, CNN can learn the liquid pressure information at different positions. CNN is mainly composed of the input layer, convolutional layer, pooling layer, fully connected layer, and output layer, as demonstrated in Figure 2. The input layer is employed for receiving raw data. The convolutional layer is utilized for extracting features from the input data. The pooling layer downsamples the convolution layer's output to lower the data's dimension. The fully connected layer turns the pooling layer's output into a one-dimensional vector and outputs it through the output layer. Compared with the full connection mode of the traditional neural network, CNN uses a convolution kernel of appropriate size to extract the local features of this layer's input, that is, this CNN convolution layer's neurons are only connected to some of the upper layer's neurons. This local connectivity method reduces the neural network's parameter count and overfitting risk. At the same time, CNN employs weight sharing to enable convolutional kernels to capture identical features at varying places, resulting in reduced model training complexity and imparting the model with the characteristics of translation invariance. The convolution operation formula is below:

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

where $\otimes$ represents convolution calculation; $\mathbf{x}_{j_1}^{l_1}$ denotes the $j_1$-th feature map of the $l_1$-th convolution layer; $\mathbf{b}_{j_1}^{l_1}$ represents offset; $\mathbf{w}_{i_1 j_1}^{l_1}$ represents convolution kernel matrix; $\sigma_1(\bullet)$ represents the nonlinear activation functions; $g_1$ indicates the number of input feature maps.

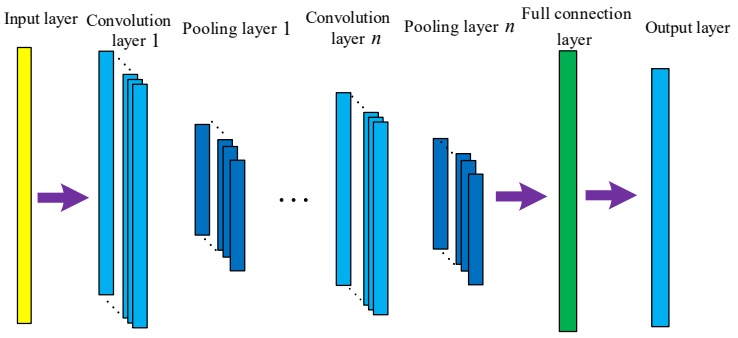



**Figure 2.** The CNN's framework.

## 3. Hybrid Models

This section describes three combination models, namely the LSTM-ARX model, CNN-ARX model, and LSTM-CNN-ARX model.

### *3.1. LSTM-ARX Model*

LSTM neural networks utilize recurrent connections that allow information to flow in a directed cycle, enabling them to effectively capture the temporal dependencies in data during modeling. Unlike feedforward neural networks, which only propagate information forward, LSTM can retain and utilize information from previous time steps, enabling them to learn and represent long-term dependencies in the input sequences. This property makes LSTM particularly well-suited for building sequential models of nonlinear systems. Simultaneously, a gating mechanism is introduced in LSTM networks to control the flow and discarding of historical information and input characteristics, which solves the long-term dependence problem in simple RNN networks. In the actual modeling process, single-layer LSTM is usually unable to express complex temporal nonlinear characteristics, so a series approach is often adopted to stack multiple LSTM layers to deepen the network and bolster modeling capabilities. The LSTM-ARX model can be obtained by approximating the function coefficient of the model (3) with LSTM. It combines the benefits of LSTM for handling long sequence data with the SD-ARX model's nonlinear expression capabilities to fully represent the nonlinear system's dynamic features. Figure 3 describes the LSTM-ARX model's architecture and its expression below:

$$
\begin{cases}
\mathbf{y}(a) = \boldsymbol{\rho}_0(\boldsymbol{\varepsilon}(a-1)) + \sum\limits_{i=1}^{s_y} \boldsymbol{\rho}_{y,i}(\boldsymbol{\varepsilon}(a-1))\mathbf{y}(a-i) + \sum\limits_{j=s_d}^{s_u+s_d-1} \boldsymbol{\rho}_{u,j}(\boldsymbol{\varepsilon}(a-1))\mathbf{u}(a-j) + \theta(a) \\
[\boldsymbol{\rho}_0(\boldsymbol{\varepsilon}(a-1)), \boldsymbol{\rho}_{y,i}(\boldsymbol{\varepsilon}(a-1)), \boldsymbol{\rho}_{u,j}(\boldsymbol{\varepsilon}(a-1))] = \mathbf{W}\Big(f(\mathbf{h}_d^n(a)\mathbf{W}_{hy} + \mathbf{b}_y)\Big) + \mathbf{b} \\
\mathbf{h}_l^r(a) = \sigma_4\Big(\mathbf{h}_l^{r-1}(a)\mathbf{W}_{ox}^r + \mathbf{h}_{l-1}^r(a)\mathbf{W}_{oh}^r + \mathbf{b}_o^r\Big) \odot \sigma_5\big(\mathbf{c}_l^r(a)\big), \mathbf{r} = 2, 3, \ldots, n \\
\mathbf{c}_l^r(a) = \sigma_4\Big(\mathbf{h}_l^{r-1}(a)\mathbf{W}_{fx}^r + \mathbf{h}_{l-1}^r(a)\mathbf{W}_{fh}^r + \mathbf{b}_f^r\Big) \odot \mathbf{c}_{l-1}^r(a) \\
\qquad + \sigma_4\Big(\mathbf{h}_l^{r-1}(a)\mathbf{W}_{ix}^r + \mathbf{h}_{l-1}^r(a)\mathbf{W}_{ih}^r + \mathbf{b}_i^r\Big) \odot \sigma_5\Big(\mathbf{h}_l^{r-1}(a)\mathbf{W}_{cx}^r + \mathbf{h}_{l-1}^r(a)\mathbf{W}_{ch}^r + \mathbf{b}_c^r\Big) \\
\mathbf{h}_l^1(a) = \sigma_4\Big(\varepsilon_l\mathbf{W}_{ox}^1 + \mathbf{h}_{l-1}^1(a)\mathbf{W}_{oh}^1 + \mathbf{b}_o^1\Big) \odot \sigma_5\big(\mathbf{c}_l^1(a)\big),\ l = 1, 2, \ldots, d \\
\mathbf{c}_l^1(a) = \sigma_4\Big(\varepsilon_l\mathbf{W}_{fx}^1 + \mathbf{h}_{l-1}^1(a)\mathbf{W}_{fh}^1 + \mathbf{b}_f^1\Big) \odot \mathbf{c}_{l-1}^1(a) \\
\qquad + \sigma_4\Big(\varepsilon_l\mathbf{W}_{ix}^1 + \mathbf{h}_{l-1}^1(a)\mathbf{W}_{ih}^1 + \mathbf{b}_i^1\Big) \odot \sigma_5\Big(\varepsilon_l\mathbf{W}_{cx}^1 + \mathbf{h}_{l-1}^1(a)\mathbf{W}_{ch}^1 + \mathbf{b}_c^1\Big) \\
\mathbf{h}_0^k(a) = \mathbf{0},\ \mathbf{c}_0^k(a) = \mathbf{0}, k = 1, 2, \ldots, n; \boldsymbol{\varepsilon}(a-1) = [\varepsilon_1, \ldots, \varepsilon_d]^T
\end{cases}
\tag{6}
$$

where $\mathbf{y}(a)$ represents output; $\mathbf{u}(a)$ represents input; $\theta(a)$ represents white noise; $s_y s_u$, $d$ are model's orders; $s_d$ denotes time delay; $\boldsymbol{\varepsilon}(a-1)$ represents the input data at time $a$, which may correspond to the system's input or/and output; $\boldsymbol{\rho}_0(\boldsymbol{\varepsilon}(a-1))$, $\boldsymbol{\rho}_{y,i}(\boldsymbol{\varepsilon}(a-1))$, and $\boldsymbol{\rho}_{u,j}(\boldsymbol{\varepsilon}(a-1))$ represent the model's function coefficient; $n$ denotes the number of hidden layers; $\mathbf{h}_l^r(a)$ and $\mathbf{c}_l^r(a)$ represent the hidden layer state and cell state of the $r$-th hidden layer of time step $l$ at time $a$; $\sigma_4(\bullet)$, $\sigma_5(\bullet)$, and $f(\bullet)$ represent activation functions (e.g., Hard Sigmoid and Tanh) that are employed to improve the model's capability in capturing the system's nonlinearity; $\left\{\mathbf{b}_f^k, \mathbf{b}_i^k, \mathbf{b}_o^k, \mathbf{b}_c^k \middle| k = 1, \ldots, n\right\}$,$\mathbf{b}_y$, and $\mathbf{b}$ represent offsets; $\left\{\mathbf{W}_{fx}^k, \mathbf{W}_{fh}^k, \mathbf{W}_{ix}^k, \mathbf{W}_{ih}^k, \mathbf{W}_{ox}^k, \mathbf{W}_{oh}^k, \mathbf{W}_{cx}^k, \mathbf{W}_{ch}^k \middle| k = 1, \ldots, n\right\}$, $\mathbf{W}_{hy}$, and $\mathbf{W}$ represent weight coefficients.

### *3.2. CNN-ARX Model*

CNN has strong nonlinear feature extraction capabilities. The CNN-ARX model is derived by fitting the SD-ARX model's coefficients to CNN. This model combines the ability of CNN's local feature extraction and the SD-ARX model's expression. Compared to traditional neural networks, CNN has the characteristics of local connections, weight sharing, pooling operations, etc., which can further decrease the complexity and overfitting

risks of the model and enable the model to possess a certain level of robustness and fault tolerance. Figure 4 describes the CNN-ARX model's architecture and its expression below:

$$\begin{cases} \mathbf{y}(a) = \boldsymbol{\rho}_0(\boldsymbol{\varepsilon}(a-1)) + \sum\limits_{i=1}^{s_y} \boldsymbol{\rho}_{y,i}(\boldsymbol{\varepsilon}(a-1))\mathbf{y}(a-i) + \sum\limits_{j=s_d}^{s_u+s_d-1} \boldsymbol{\rho}_{u,j}(\boldsymbol{\varepsilon}(a-1))\mathbf{u}(a-j) + \theta(a) \\ [\boldsymbol{\rho}_0(\boldsymbol{\varepsilon}(a-1)), \boldsymbol{\rho}_{y,i}(\boldsymbol{\varepsilon}(a-1)), \boldsymbol{\rho}_{u,j}(\boldsymbol{\varepsilon}(a-1))] = \mathbf{W}_1\left(f_1(\tilde{\mathbf{x}}^{n_1}(a)\mathbf{W}_{xy} + \mathbf{b}_x)\right) + \mathbf{b}_1 \\ \tilde{\mathbf{x}}^{n_1}(a) = \sigma_3\left(\hat{\mathbf{x}}_{j_1}^{n_1}(a)\right); \hat{\mathbf{x}}_{j_1}^{n_1}(a) = \sigma_2\left(\mathbf{x}_{j_1}^{n_1}(a)\right) \\ \mathbf{x}_{j_1}^{l_1}(a) = \sigma_1\left(\sum\limits_{i_1=1}^{g_1} \mathbf{x}_{i_1}^{l_1-1}(a) \otimes \mathbf{W}_{i_1 j_1}^{l_1} + \mathbf{b}_{j_1}^{l_1}\right), 1 \le l_1 \le n_1 \\ \mathbf{x}^0(a) = \boldsymbol{\varepsilon}(a-1) = [\varepsilon_1, \ldots, \varepsilon_d]^T \end{cases} \tag{7}$$

where $\mathbf{x}^0(a)$ represents input; $\sigma_1(\bullet)$ and $f_1(\bullet)$ represent activation functions; $\sigma_2(\bullet)$ and $\sigma_3(\bullet)$ indicate pooling operation and flattening operation; $\tilde{\mathbf{x}}^{n_1}(a)$ denotes the one-dimensional vector obtained by flattening the output feature maps $\hat{\mathbf{x}}_{j_1}^{n_1}(a)$; $\left\{\mathbf{W}_{i_1 j_1}^{l_1} \middle| l_1 = 1, \ldots, n_1\right\}$, $\mathbf{W}_{xy}$, and $\mathbf{W}_1$ represent weight coefficients; $\left\{\mathbf{b}_{j_1}^{l_1} \middle| l_1 = 1, \ldots, n_1\right\}$, $\mathbf{b}_x$, and $\mathbf{b}_1$ represent the offsets.

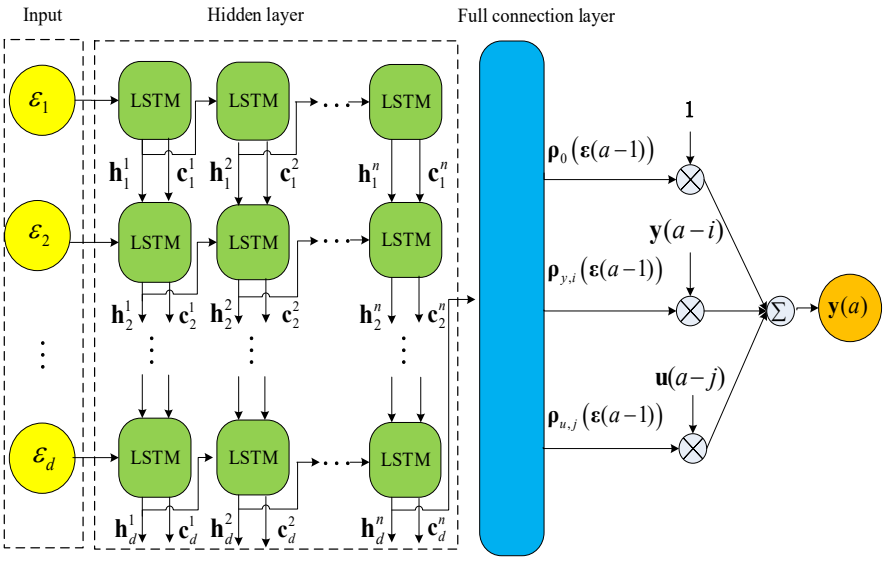



**Figure 3.** The LSTM-ARX model's architecture.

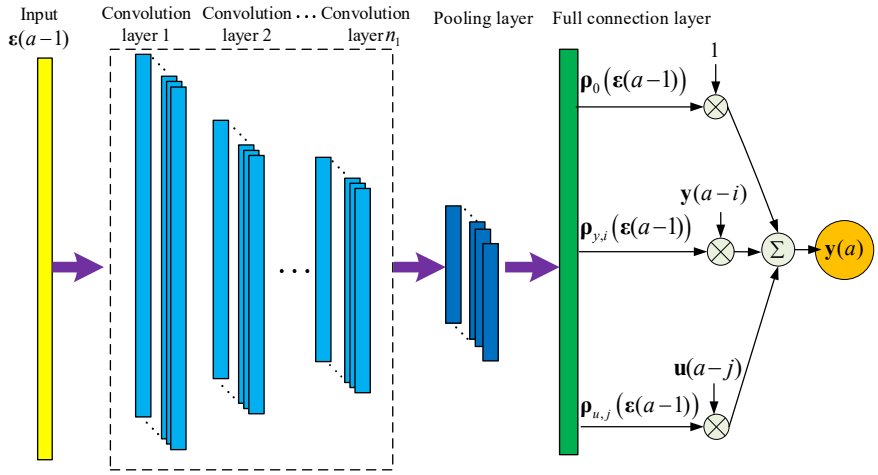



**Figure 4.** The CNN-ARX model's architecture.

### 3.3. LSTM-CNN-ARX Model

The LSTM-CNN-ARX model is obtained by combining LSTM and CNN to approximate SD-ARX model coefficients, as shown in Figure 5. First, employ LSTM for learning the temporal characteristics of the input data, followed by the utilization of CNN to capture the spatial characteristics, and finally calculate the SD-ARX model's correlation coefficients through a full connection layer to obtain the LSTM-CNN-ARX model. This model integrates the spatiotemporal characteristic extraction ability of LSTM-CNN with the expressive power of the SD-ARX model; this approach effectively captures the nonlinear system's dynamic features. The LSTM-CNN-ARX model's mathematical expression is obtained from Formulas (6) and (7), as follows:

$$
\begin{cases}
\mathbf{y}(a) = \boldsymbol{\rho}_0(\boldsymbol{\varepsilon}(a-1)) + \sum\limits_{i=1}^{s_y} \boldsymbol{\rho}_{y,i}(\boldsymbol{\varepsilon}(a-1))\mathbf{y}(a-i) + \sum\limits_{j=s_d}^{s_u+s_d-1} \boldsymbol{\rho}_{u,j}(\boldsymbol{\varepsilon}(a-1))\mathbf{u}(a-j) + \theta(a) \\
[\boldsymbol{\rho}_0(\boldsymbol{\varepsilon}(a-1)), \boldsymbol{\rho}_{y,i}(\boldsymbol{\varepsilon}(a-1)), \boldsymbol{\rho}_{u,j}(\boldsymbol{\varepsilon}(a-1))] = \mathbf{W}_1\left(f_1(\tilde{\mathbf{x}}^{n_1}(a)\mathbf{W}_{xy} + \mathbf{b}_x)\right) + \mathbf{b}_1 \\
\tilde{\mathbf{x}}^{n_1}(a) = \sigma_3(\hat{\mathbf{x}}_{j_1}^{n_1}(a)); \hat{\mathbf{x}}_{j_1}^{n_1}(a) = \sigma_2\left(\mathbf{x}_{j_1}^{n_1}(a)\right) \\
\mathbf{x}_{j_1}^{l_1}(a) = \sigma_1\left(\sum\limits_{i_1=1}^{g_1} \mathbf{x}_{i_1}^{l_1-1}(a) \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(a) = \mathbf{h}_l^n(a) \\
\mathbf{h}_l^r(a) = \sigma_4\left(\mathbf{h}_l^{r-1}(a)\mathbf{W}_{ox}^r + \mathbf{h}_{l-1}^r(a)\mathbf{W}_{oh}^r + \mathbf{b}_o^r\right) \odot \sigma_5(\mathbf{c}_l^r(a)), \mathrm{r} = 2,3,\ldots,n \\
\mathbf{c}_l^r(a) = \sigma_4\left(\mathbf{h}_l^{r-1}(a)\mathbf{W}_{fx}^r + \mathbf{h}_{l-1}^r(a)\mathbf{W}_{fh}^r + \mathbf{b}_f^r\right) \odot \mathbf{c}_{l-1}^r(a) \\
\qquad + \sigma_4\left(\mathbf{h}_l^{r-1}(a)\mathbf{W}_{ix}^r + \mathbf{h}_{l-1}^r(a)\mathbf{W}_{ih}^r + \mathbf{b}_i^r\right) \odot \sigma_5\left(\mathbf{h}_l^{r-1}(a)\mathbf{W}_{cx}^r + \mathbf{h}_{l-1}^r(a)\mathbf{W}_{ch}^r + \mathbf{b}_c^r\right) \\
\mathbf{h}_l^1(a) = \sigma_4\left(\varepsilon_l\mathbf{W}_{ox}^1 + \mathbf{h}_{l-1}^1(a)\mathbf{W}_{oh}^1 + \mathbf{b}_o^1\right) \odot \sigma_5(\mathbf{c}_l^1(a)), \ l = 1,2,\ldots,d \\
\mathbf{c}_l^1(a) = \sigma_4\left(\varepsilon_l\mathbf{W}_{fx}^1 + \mathbf{h}_{l-1}^1(a)\mathbf{W}_{fh}^1 + \mathbf{b}_f^1\right) \odot \mathbf{c}_{l-1}^1(a) \\
\qquad + \sigma_4\left(\varepsilon_l\mathbf{W}_{ix}^1 + \mathbf{h}_{l-1}^1(a)\mathbf{W}_{ih}^1 + \mathbf{b}_i^1\right) \odot \sigma_5\left(\varepsilon_l\mathbf{W}_{cx}^1 + \mathbf{h}_{l-1}^1(a)\mathbf{W}_{ch}^1 + \mathbf{b}_c^1\right) \\
\mathbf{h}_0^k(a) = \mathbf{0}, \ \mathbf{c}_0^k(a) = \mathbf{0}, k = 1,2,\ldots,n; \boldsymbol{\varepsilon}(a-1) = [\varepsilon_1,\ldots,\varepsilon_d]^T
\end{cases}
\tag{8}
$$

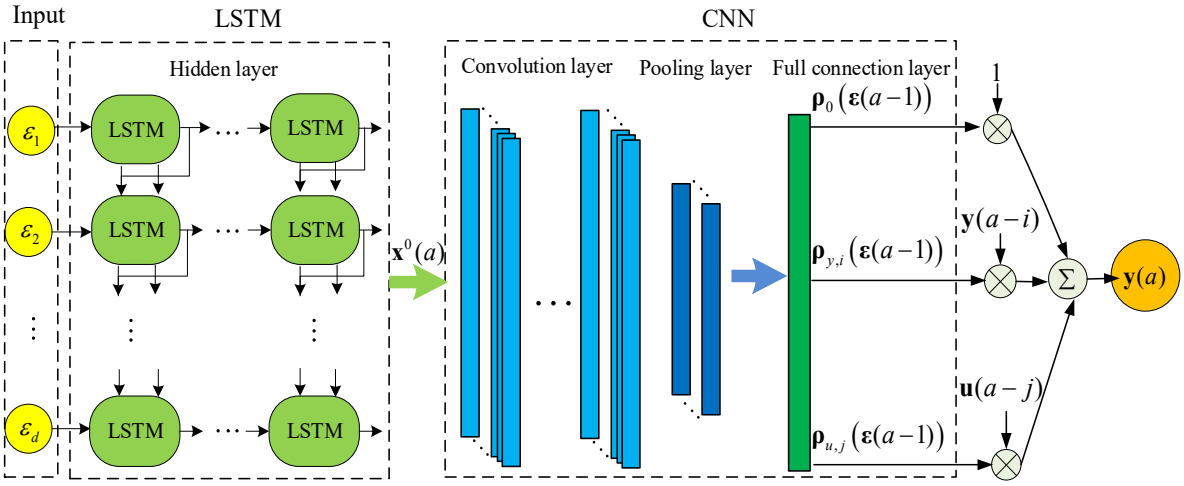



**Figure 5.** The LSTM-CNN-ARX model's architecture.

The aforementioned deep learning-based SD-ARX models adopt data-driven techniques for modeling. First, select appropriate and effective input and output data as identification data for modeling. Second, based on the complexity of the identified system, select the model's initial orders, the network's layer count, and the node count per layer. Third, choose activation functions (such as Sigmoid, Tanh, and Relu) and optimizers (such as Adam and SGD) and configure other hyperparameters of the model. Fourth, train the model and evaluate its performance using a test set, computing the mean square error

(MSE). Fifthly, adjust the orders while keeping the other structures of the model unchanged to train the model with new orders again, and select the orders of the model with the minimum MSE as the optimal orders. Next, keep the other structures of the model unchanged, adjust the network's layer count, and select the network's layer count with the smallest MSE as the optimal number of layers. Using the same method, determine the other model structures, i.e., the node count, activation function, and hyperparameters, and ultimately choose the model structure with the smallest MSE as the optimal model.

## 4. MPC Algorithm Design

This section designs the predictive controller based on models (6)–(8). By utilizing the model's local linear and global nonlinear features, the predictive controller uses current state operating point information to locally linearize the model, calculates the nonlinear system's predicted output, and obtains the control law by solving a quadratic programming (QP) problem.

For the convenience of MPC algorithm development, models (6)–(8) can be converted to the following forms:

$$\begin{cases} \mathbf{y}(a) = \boldsymbol{\rho}_0(a-1) + \sum\limits_{q=1}^{k_g} \widehat{\boldsymbol{\gamma}}_{q,a-1}\mathbf{y}(a-q) + \sum\limits_{q=1}^{k_g} \widehat{\boldsymbol{\lambda}}_{q,a-1}\mathbf{u}(a-q) + \boldsymbol{\theta}(a) \\ \boldsymbol{\rho}_0(a-1) = \boldsymbol{\rho}_0(\boldsymbol{\varepsilon}(a-1)) \\ k_g = \max\left(s_y, s_u + s_d - 1\right) \\ \widehat{\boldsymbol{\gamma}}_{q,a-1} = \begin{cases} \boldsymbol{\rho}_{y,q}(\boldsymbol{\varepsilon}(a-1)), (q \le s_y) \\ \mathbf{0}\ , (q > s_y) \end{cases} , \widehat{\boldsymbol{\lambda}}_{q,a-1} = \begin{cases} \boldsymbol{\rho}_{u,q}(\boldsymbol{\varepsilon}(a-1)), (s_d \le q \le s_u + s_d - 1) \\ \mathbf{0}\ , \text{else} \end{cases} \end{cases} \tag{9}$$

where $\boldsymbol{\rho}_0(a-1)$, $\widehat{\boldsymbol{\gamma}}_{q,a-1}$, and $\widehat{\boldsymbol{\lambda}}_{q,a-1}$ indicate regression coefficients; $k_g$ represents the maximum value of $s_u + s_d - 1$ and $s_y$; $\boldsymbol{\theta}(a)$ denotes the white noise.

To transform the above equation into the state-space form, design the vectors below:

$$\begin{cases} \mathbf{X}(a) = [\mathbf{x}_{1,a}^{\mathrm{T}}, \mathbf{x}_{2,a}^{\mathrm{T}} \cdots \mathbf{x}_{k_g,a}^{\mathrm{T}}]\mathrm{T}, \\ \mathbf{x}_{1,a} = \mathbf{y}(a) \\ \mathbf{x}_{p,a} = \sum\limits_{i=1}^{k_g+1-p} \widehat{\boldsymbol{\gamma}}_{i+p-1,a-1}\mathbf{y}(a-i) + \sum\limits_{j=1}^{k_g+1-p} \widehat{\boldsymbol{\lambda}}_{j+p-1,a-1}\mathbf{u}(a-j) \\ p = 2, 3, \cdots, k_g \end{cases} \tag{10}$$

Therefore, Equation (9) becomes the following form:

$$\begin{cases} \mathbf{X}(a+1) = \mathbf{G}_a\mathbf{X}(a) + \mathbf{H}_a\mathbf{u}(a) + \boldsymbol{\rho}_a + \boldsymbol{\Lambda}(a+1) \\ \mathbf{y}(a) = \mathbf{Q}\mathbf{X}(a) \end{cases} \tag{11}$$

where

$$\begin{cases} \mathbf{G}_a = \begin{bmatrix} \widehat{\boldsymbol{\gamma}}_{1,a} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \widehat{\boldsymbol{\gamma}}_{2,a} & \mathbf{0} & \mathbf{I} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \mathbf{0} \\ \widehat{\boldsymbol{\gamma}}_{k_g-1,a} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \\ \widehat{\boldsymbol{\gamma}}_{k_g,a} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}_{k_g \times k_g}, \\ \mathbf{H}_a = [\widehat{\boldsymbol{\lambda}}_{1,a}\ \widehat{\boldsymbol{\lambda}}_{2,a} \cdots \widehat{\boldsymbol{\lambda}}_{k_g,a}]^{\mathrm{T}}, \mathbf{Q} = [\mathbf{I}\ \mathbf{0} \cdots \mathbf{0}], \\ \boldsymbol{\rho}_a = [\boldsymbol{\rho}_0\ \mathbf{0} \cdots \mathbf{0}]^{\mathrm{T}}, \boldsymbol{\Lambda}(a+1) = [\boldsymbol{\theta}(a+1)\ \mathbf{0} \cdots \mathbf{0}]^{\mathrm{T}} \end{cases} \tag{12}$$

Note that the coefficient matrix above is related to the working state $\boldsymbol{\varepsilon}(a)$ at time $a$. For different state-dependent ARX models, the values of matrices $\mathbf{G}_a$, $\mathbf{H}_a$, and $\boldsymbol{\rho}_a$ in

Equation (12) are different and can be calculated using Equation (9). To further develop the model predictive controller, define the vectors below:

$$\begin{cases} \widehat{\mathbf{X}}(a) = [\widehat{\mathbf{X}}(a+1|a)^{\mathrm{T}}\widehat{\mathbf{X}}(a+2|a)^{\mathrm{T}}\cdots\widehat{\mathbf{X}}(a+M_y|a)^{T}]^{\mathrm{T}} \\ \widehat{\mathbf{Y}}(a) = [\widehat{\mathbf{y}}(a+1|a)^{\mathrm{T}}\widehat{\mathbf{y}}(a+2|a)^{\mathrm{T}}\cdots\widehat{\mathbf{y}}(a+M_y|a)^{\mathrm{T}}]^{\mathrm{T}} \\ \widehat{\mathbf{U}}(a) = [\mathbf{u}(a)^{\mathrm{T}}\mathbf{u}(a+1)^{\mathrm{T}}\cdots\mathbf{u}(a+M_u-1)^{\mathrm{T}}]^{\mathrm{T}} \\ \widehat{\boldsymbol{\rho}}_a = [\boldsymbol{\rho}_a^{\mathrm{T}}\boldsymbol{\rho}_{a+1}^{\mathrm{T}}\cdots\boldsymbol{\rho}_{a+M_y-1}^{\mathrm{T}}]^{\mathrm{T}} \end{cases} \tag{13}$$

where $\widehat{\mathbf{X}}(a)$, $\widehat{\mathbf{Y}}(a)$, $\widehat{\mathbf{U}}(a)$, and $\widehat{\boldsymbol{\rho}}_a$ represent the multi-step forward prediction state vector, multi-step forward prediction output vector, multi-step forward prediction control vector, and multi-step forward prediction offset, respectively; $\left\{\widehat{\mathbf{X}}(a+q|a)^{\mathrm{T}}\middle|q=1,\ldots,M_y\right\}$ and $\left\{\widehat{\mathbf{y}}(a+q|a)^{\mathrm{T}}\middle|q=1,\ldots,M_y\right\}$ represent the $q$-step forward prediction of the state and output; $M_y$ and $M_u$ represent the prediction and control time domains, respectively. The model's multi-step forward prediction control is presented below:

$$\begin{cases} \widehat{\mathbf{X}}(a) = \widehat{\mathbf{G}}_a\mathbf{X}(a) + \widehat{\mathbf{H}}_a\widehat{\mathbf{U}}(a) + \widehat{\boldsymbol{\Xi}}_a\widehat{\boldsymbol{\rho}}_a, \\ \widehat{\mathbf{Y}}(a) = \widehat{\mathbf{Q}}\widehat{X}(a) \end{cases} \tag{14}$$

where

$$\widehat{\mathbf{G}}_a = \begin{bmatrix} \coprod_{i=0}^{0}\mathbf{G}_{a+i} \\ \coprod_{i=0}^{1}\mathbf{G}_{a+i} \\ \vdots \\ \coprod_{i=0}^{M_y-1}\mathbf{G}_{a+i} \end{bmatrix}, \quad \coprod_{i=j}^{q}\mathbf{G}_{a+i} = \begin{cases} \mathbf{G}_{a+q}\mathbf{G}_{a+q-1}\cdots\mathbf{G}_{a+j}, & j \le q \\ \mathbf{I}, & j > q \end{cases} \tag{15}$$

$$\widehat{\boldsymbol{\Xi}}_a = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \coprod_{i=1}^{1}\mathbf{G}_{a+i} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \coprod_{i=1}^{2}\mathbf{G}_{a+i} & \coprod_{i=2}^{2}\mathbf{G}_{a+i} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \coprod_{i=1}^{M_y-1}\mathbf{G}_{a+i} & \coprod_{i=2}^{M_y-1}\mathbf{G}_{a+i} & \cdots & \coprod_{i=M_y-1}^{M_y-1}\mathbf{G}_{a+i} & \mathbf{I} \end{bmatrix} \tag{16}$$

$$\widehat{\mathbf{H}}_a = \begin{bmatrix} \mathbf{H}_a & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \left(\coprod_{i=1}^{1}\mathbf{G}_{a+i}\right)\mathbf{H}_a & \mathbf{H}_{a+1} & \mathbf{0} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \left(\coprod_{i=1}^{M_u-1}\mathbf{G}_{a+i}\right)\mathbf{H}_a & \left(\coprod_{i=2}^{M_u-1}\mathbf{G}_{a+i}\right)\mathbf{H}_{a+1} & \cdots & \left(\coprod_{i=M_u-1}^{M_u-1}\mathbf{G}_{a+i}\right)\mathbf{H}_{a+M_u-2} & \mathbf{H}_{a+M_u-1} \\ \left(\coprod_{i=1}^{M_u}\mathbf{G}_{a+i}\right)\mathbf{H}_a & \left(\coprod_{i=2}^{M_u}\mathbf{G}_{a+i}\right)\mathbf{H}_{a+1} & \cdots & \left(\coprod_{i=M_u-1}^{M_u}\mathbf{G}_{a+i}\right)\mathbf{H}_{a+M_u-2} & \sum_{j=M_u-1}^{M_u}\left(\coprod_{i=j+1}^{M_u}\mathbf{G}_{a+i}\right)\mathbf{H}_{a+j} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \left(\coprod_{i=1}^{M_y-1}\mathbf{G}_{a+i}\right)\mathbf{H}_a & \left(\coprod_{i=2}^{M_y-1}\mathbf{G}_{a+i}\right)\mathbf{H}_{a+1} & \cdots & \left(\coprod_{i=M_u-1}^{M_y-1}\mathbf{G}_{a+i}\right)\mathbf{H}_{a+M_u-2} & \sum_{j=M_u-1}^{M_y-1}\left(\coprod_{i=j+1}^{M_y-1}\mathbf{G}_{a+i}\right)\mathbf{H}_{a+j} \end{bmatrix} \tag{17}$$

$$\widehat{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q} \end{bmatrix} \tag{18}$$

In Formula (14), the coefficient matrices $\widehat{\mathbf{G}}_a$, $\widehat{\mathbf{\Xi}}_a$, $\widehat{\mathbf{H}}_a$, and $\widehat{\mathbf{Q}}$ change with the system state and can be obtained by solving the system's future working point $\boldsymbol{\varepsilon}(a+q|a)$ $(q = 1, 2, \cdots, M_y - 1)$. Nevertheless, in the actual control process, it may be challenging to obtain information on the future state operating point. Therefore, compute using the current operating point $\boldsymbol{\varepsilon}(a)$ instead of $\boldsymbol{\varepsilon}(a+q|a)$ for the computer. Obtain the locally linearized model according to Equation (11) and use it to develop the following MPC method.

To facilitate the design of the objective function, Equation (14) is converted into the following forms:

$$\begin{cases} \widehat{\mathbf{Y}}(a) = \mathbf{R}_a \widehat{\mathbf{U}}(a) + \mathbf{Y_0}(a) \\ \mathbf{R}_a = \widehat{\mathbf{Q}}\widehat{\mathbf{H}}_a \\ \mathbf{Y_0}(a) = \widehat{\mathbf{Q}}\widehat{\mathbf{G}}_a \mathbf{X}(a) + \widehat{\mathbf{Q}}\widehat{\mathbf{\Xi}}_a \widehat{\boldsymbol{\rho}}_a \end{cases} \tag{19}$$

Then, the desired output $\widehat{\mathbf{Y}}_r(a)$ and control increment $\Delta\widehat{\mathbf{U}}(a)$ are defined:

$$\begin{cases} \Delta\widehat{\mathbf{U}}(a) = [\Delta\mathbf{u}(a)^T \Delta\mathbf{u}(a+1)^T \cdots \Delta\mathbf{u}(a+M_u-1)^{\mathrm{T}}]^{\mathrm{T}} \\ \widehat{\mathbf{Y}}_r(a) = [\mathbf{y}_r(a+1)^{\mathrm{T}}\, \mathbf{y}_r(a+2)^T \cdots \mathbf{y}_r(a+M_y)^{\mathrm{T}}]^T \end{cases} \tag{20}$$

where $\Delta\mathbf{u}(a) = \mathbf{u}(a) - \mathbf{u}(a-1)$ . The optimization objective function design for MPC as below:

$$\begin{cases} \min\limits_{\widehat{\mathbf{U}}(a)} J = \|\widehat{\mathbf{Y}}(a) - \widehat{\mathbf{Y}}_r(a)\|^2_{\mathbf{A}_1} + \|\widehat{\mathbf{U}}(a)\|^2_{\mathbf{B}_1} + \|\Delta\widehat{\mathbf{U}}(a)\|^2_{\mathbf{B}_2} \\ s.t.\ \mathbf{Y}_{\min} \le \widehat{\mathbf{Y}}(a) \le \mathbf{Y}_{\max}, \mathbf{U}_{\min} \le \widehat{\mathbf{U}}(a) \le \mathbf{U}_{\max}, \Delta\mathbf{U}_{\min} \le \Delta\widehat{\mathbf{U}}(a) \le \Delta\mathbf{U}_{\max} \end{cases} \tag{21}$$

where $\|\mathbf{X}\|^2_{\Delta} = \mathbf{X}^T\mathbf{\Delta}\mathbf{X}$; $\mathbf{A}_1$, $\mathbf{B}_1$, and $\mathbf{B}_2$ represent weight coefficient matrices. Substitute Equations (19) and (20) into Equation (21) to eliminate the constant term and convert it into the following QP problem:

$$\begin{cases} \min\limits_{\widehat{\mathbf{U}}(a)} \widehat{J} = \frac{1}{2}\widehat{\mathbf{U}}(a)^T\left[\mathbf{R}_a^{\mathrm{T}}\mathbf{A}_1\mathbf{R}_a + \mathbf{B}_1 + \mathbf{N}^{-\mathrm{T}}\mathbf{B}_2\mathbf{N}^{-1}\right]\widehat{\mathbf{U}}(a) \\ \qquad + \left[\mathbf{Y_0}(a)^{\mathrm{T}}\mathbf{A}_1\mathbf{R}_a - \mathbf{Y}_r(a)^{\mathrm{T}}\mathbf{A}_1\mathbf{R}_a - \mathbf{U}_0(a-1)^{\mathrm{T}}\mathbf{N}^{-\mathrm{T}}\mathbf{B}_2\mathbf{N}^{-1}\right]\widehat{\mathbf{U}}(a) \\ s.t. \begin{bmatrix} \mathbf{R}_a \\ -\mathbf{R}_a \end{bmatrix}\widehat{\mathbf{U}}(a) \le \begin{bmatrix} \mathbf{Y}_{\max} - \mathbf{Y}_0(a) \\ -\mathbf{Y}_{\min} + \mathbf{Y}_0(a) \end{bmatrix}, \mathbf{U}_{\min} \le \widehat{\mathbf{U}}(a) \le \mathbf{U}_{\max}, \\ \qquad \mathbf{U}_0(a-1) + \mathbf{N}\Delta\mathbf{U}_{\min} \le \widehat{\mathbf{U}}(a) \le \mathbf{U}_0(a-1) + \mathbf{N}\Delta\mathbf{U}_{\max}. \end{cases} \tag{22}$$

where

$$\begin{cases} \widehat{\mathbf{U}}(a) = \mathbf{U}_0(a-1) + \mathbf{N}\Delta\widehat{\mathbf{U}}(a), \\ \mathbf{U}_0(a-1) = [\ \mathbf{U}(a-1)^{\mathrm{T}}\ \ \mathbf{U}(a-1)^T\ \ \cdots\ \ \mathbf{U}(a-1)^{\mathrm{T}}\ ]^{\mathrm{T}} \\ \mathbf{N} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & \mathbf{I} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \end{bmatrix} \end{cases} \tag{23}$$

Equation (22) represents a convex QP problem with constraints. If the feasible set defined by the constraint condition is not empty and the objective function has a lower bound within the feasible set, then the QP problem has a globally optimal solution. If Equation (22) has no feasible solution in a certain control period, the feasible solution obtained in the previous period is used for control in the practical control. In addition, in practical control, in order to avoid control sequence deviations caused by environmental or system inherent biases, the optimal control sequence's initial value is employed as the control input, and the system output is observed at the next moment for feedback correction, thus realizing online rolling optimization.

## 5. Control Experiments

This part uses the water tank system shown in Figure 6 as the experimental object, which is a commonly used process control device in industrial production. The deep learning models are established using the data-driven modeling approach, and the predictive controllers are designed based on the model's distinctive structure for real-time control comparative experiments on water tank systems.

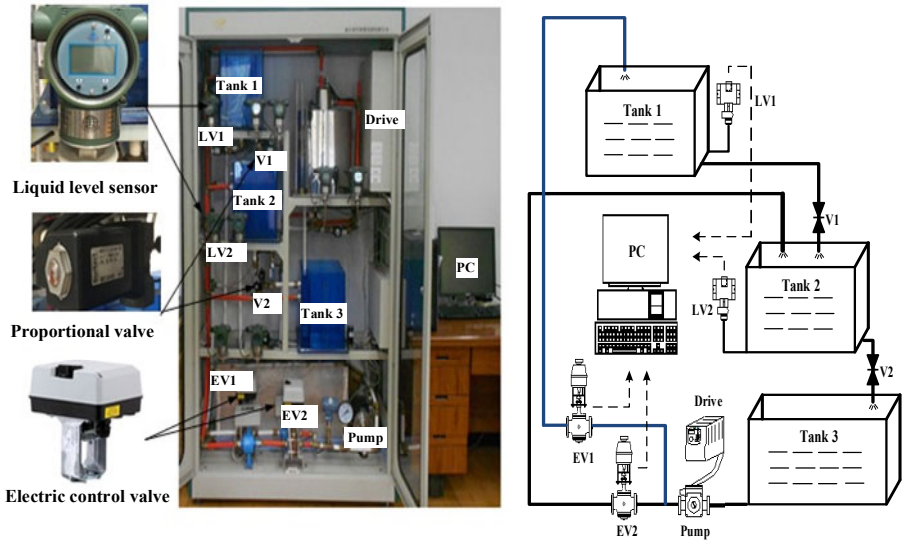



**Figure 6.** Water tank system [21].

### *5.1. Water Tank System*

The water tank system is a representative dual-input, dual-output process control piece of experimental equipment with strong coupling, large time delay, and strong nonlinearity. The water inflows of water tanks 1 and 2 are controlled by electric control valves EV1 and EV2, the water outflows are regulated by proportional valves V1 and V2, and the water level heights are detected by the liquid level sensors LV1 and LV2. Control experiments were performed on the water tank system, and models (6)–(8) were designed and are shown below:

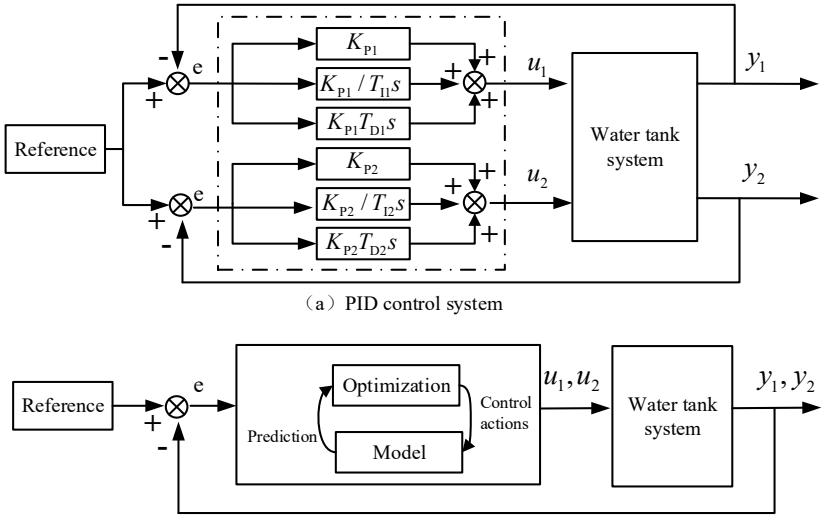
$$\begin{cases} \mathbf{Y}(a) = \sum\limits_{i=1}^{s_y} \mathbf{G}_{i,a-1}\mathbf{Y}(a-i) + \sum\limits_{j=s_d}^{s_u+s_d-1} \mathbf{H}_{j,a-1}\mathbf{U}(a-j) + \boldsymbol{\rho}_{0,a-1} + \boldsymbol{\theta}(a) \\ \mathbf{G}_{i,a-1} = \begin{bmatrix} \gamma_{i,a-1}^{11}\gamma_{i,a-1}^{12} \\ \gamma_{i,a-1}^{21}\gamma_{i,a-1}^{22} \end{bmatrix}, \mathbf{H}_{j,a-1} = \begin{bmatrix} \lambda_{j,a-1}^{11}\lambda_{j,a-1}^{12} \\ \lambda_{j,a-1}^{21}\lambda_{j,a-1}^{22} \end{bmatrix}, \boldsymbol{\rho}_{0,a-1} = \begin{bmatrix} \rho_{0,a-1}^{1} \\ \rho_{0,a-1}^{2} \end{bmatrix} \end{cases} \tag{24}$$

where $\mathbf{Y}(a) = [y_1(a), y_2(a)]^{\mathrm{T}}$ represents the liquid level heights of tank 1 and tank 2; $\mathbf{U}(a) = [u_1(a), u_2(a)]^{\mathrm{T}}$ represents the openings (0%~100%) of electric control valves EV1 and EV2; $\{\mathbf{G}_{i,a-1} | i = 1, \ldots, s_y\}$, $\{\mathbf{H}_{j,a-1} | j = s_d, \ldots, s_u + s_d - 1\}$, and $\boldsymbol{\rho}_{0,a-1}$ are the weight coefficients, which can be obtained by models (6)–(8); $\boldsymbol{\theta}(a)$ indicates the white noise

signal. Among them, we have set the working point $\boldsymbol{\varepsilon}(a-1) = [\mathbf{Y}(a-1)^{\mathrm{T}}, \cdots, \mathbf{Y}(a-d)^{\mathrm{T}}]^{\mathrm{T}}$ because the change in water level is an important factor leading to the strongly nonlinear dynamic characteristics of the water tank system.

### *5.2. Estimation of Model*

In order to make the collected modeling data contain the various nonlinear dynamic characteristics of the water tank system to the maximum extent, this article uses the expected signal, including sine and step, to control the large fluctuation of the water tank level in the normal working range via the PID controller to obtain effective input/output sampling data. The time is 2 s. Figure 7 presents the structure of the PID controller used, which is mainly used to generate the system identification data. The PID controller output is limited between 0 and 100 (%).




(a) PID control system

(b) MPC control system

**Figure 7.** The structure of MPC and PID control systems.

The sampling data given in Figure 8 are segregated into different proportions for model training and testing. The best proportion of the final modeling results is that the first 3500 data points (sampling time 2 s) are utilized for estimating the model, while the subsequent 1000 data points are utilized for verifying the model.

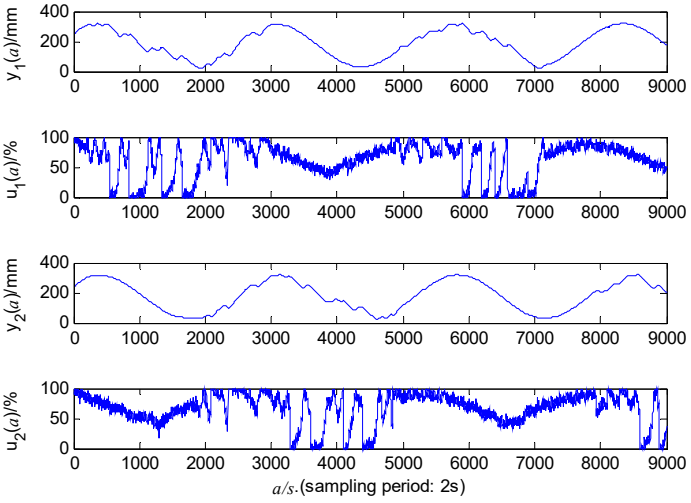



**Figure 8.** Identification data of water tank system.

The parameter optimization algorithms of the deep learning-based SD-ARX models are all the sampling adaptive motion estimation (Adam) algorithms, which integrate the advantages of momentum optimization and adaptive learning rate with high computational efficiency and can accelerate parameter convergence. The Tanh function is the full connecting layer activation function for all models, which gives the models a faster convergence rate and a stronger nonlinear expression ability. The hidden layer activation functions $\sigma_4(\bullet)$ and $\sigma_5(\bullet)$ of the LSTM are the Hard Sigmaid function and Tanh function, respectively. Compared with the standard Sigmoid function, the application of the Hard Sigmaid function in the neural network is more stable. It can reduce the gradient vanishing problem and has a faster calculation speed. The CNN employs the Relu as its convolutional activation function, which assists in reducing redundant calculation and parameter quantity between neurons, thus enhancing the model's efficiency and generalization performance. The CNN pool function selects the average pool, which is beneficial for reducing the model's complexity, minimizing the overfitting risk, and improving the feature's robustness. Its convolutional and pooling layers have kernel sizes of 3 and 4. The MSE of different model structures is calculated and compared. When the model complexity meets the requirements, select the model structure and parameters with the lowest MSE for real-time control experiments. In addition, the search range for hyperparameters of all deep learning models includes the learning rate between 0.0001 and 0.1, the batch size between 4 and 128, and the epochs between 100 and 600. The hyperparameters finally selected include the Learning rate, batch size, and epochs of 0.001, 16, and 400, respectively. The parameter settings of the LSTM-CNN-ARX model proposed in this article are shown in Table 1.

**Table 1.** Parameter setting of the proposed model.

| **Proposed Model** | **Configuration** | | | |
|---|---|---|---|---|
| LSTM-CNN-ARX | LSTM | Units 1 | Units = 8 | Epochs = 400, Batch size = 16; Optimizer = 'Adam'; Learning rate = 0.001. |
| | | Units 2 | Units = 8 | |
| | | Units 3 | Units = 8 | |
| | CNN | Convolution 1 | Filter = 16; Stride = 1; Kernel size = 3 | |
| | | Convolution 2 | Filter = 16; Stride = 1; Kernel size = 3 | |
| | | Convolution 3 | Filter = 16; Stride = 1; Kernel size = 3 | |
| | | Average-pooling | Stride = 1; Kernel size = 4 | |

The model training in this study was implemented using the Keras [55] framework in the Jupyter Notebook 5.5.0 software development environment on a PC (Intel i9-11900k 3.5 GHz, 16 GB RAM). The supervisory control software used for the real-time control experiments was Kingview 6.55. Data communication between Matlab/Simulink 2022b software and Kingview 6.55 was achieved via the Dynamic Data Exchange (DDE) protocol, enabling real-time control of the water tank system. Taking the LSTM-CNN-ARX model as an example, Figures 9 and 10 show that the model's training and testing residuals are very small and the models have high modeling accuracy. Table 2 illustrates that different models have varying structures because they have distinct characteristics and nonlinear description capabilities. At the same time, the model with more parameters and deeper network layers has stronger nonlinear expression ability, but this also increases the computational burden, resulting in a longer training time. The findings suggest that the ARX model exhibits the maximum MSE and cannot capture the nonlinear system's dynamic behavior well. The deep learning-based ARX model has a lower MSE than the RBF-ARX model because the multi-layer deep learning network has stronger nonlinear expression capabilities than the single-layer RBF neural network. The LSTM-ARX model has less MSE than the CNN-ARX model because LSTM can maintain long-term dependencies on sequence data. In addition, because the LSTM-CNN-ARX model incorporates the strengths of LSTM for learning temporal characteristics and CNN for capturing spatial characteristics, its MSE is smaller than the LSTM-ARX and CNN-ARX models, which can accurately capture the water tank system's nonlinear behavior.

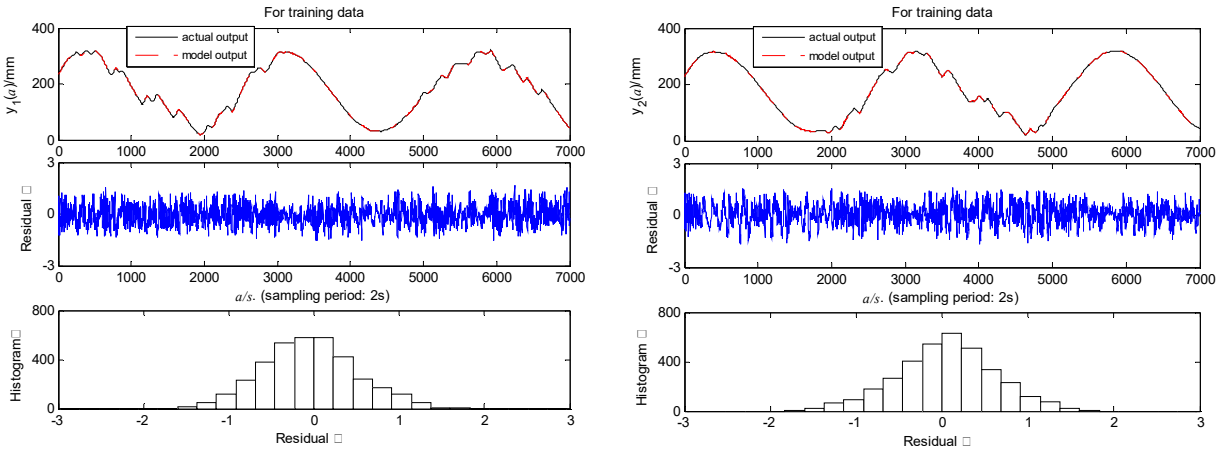



**Figure 9.** LSTM-CNN-ARX model's training result.

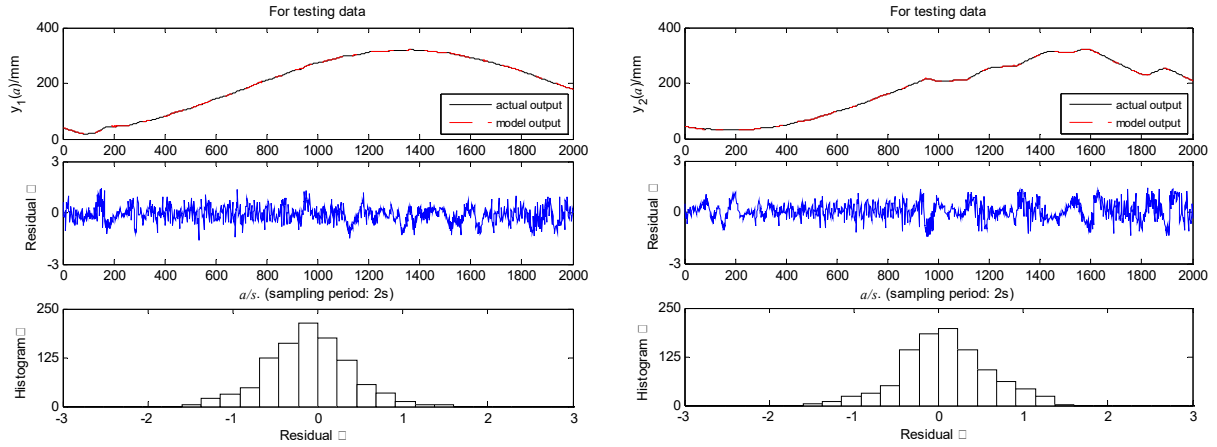



**Figure 10.** LSTM-CNN-ARX model's testing result.

**Table 2.** Comparison of modeling results.

| Models ($s_y$,$s_u$,$s_d$,$d$,$n$,$n_1$) | Number of Nodes in Each Layer | Number of Parameters | Training Time | MSE of Training Data | | MSE of Testing Data | |
|---|---|---|---|---|---|---|---|
| | | | | $y_1(a)$ | $y_2(a)$ | $y_1(a)$ | $y_2(a)$ |
| ARX (18,18,6,2,/,/) [19] | / | 146 | 18 s | 0.4910 | 0.3526 | 0.4176 | 0.4101 |
| RBF-ARX (23,20,6,2,/,/) [19] | 2 | 562 | 169 s | 0.4525 | 0.3158 | 0.3804 | 0.3712 |
| CNN-ARX (19,22,6,2,0,3) | 8,16,8 | 3142 | 205 s | 0.4277 | 0.2925 | 0.3562 | 0.3448 |
| LSTM-ARX (18,20,6,2,3,0) | 16,32,16 | 23,738 | 1152 s | 0.4013 | 0.2634 | 0.3353 | 0.3163 |
| LSTM-CNN-ARX (22,21,6,2,3,3) | 8,8,8,16,16,16 | 9710 | 1160 s | 0.3732 | 0.2437 | 0.3076 | 0.2866 |

### *5.3. Real-Time Control Experiments*

This study was conducted in a laboratory environment using the water tank system experimental equipment shown in Figure 1, which uses the deep learning ARX model-based MPC method to compare with PID, ARX-MPC, and RBF-ARX-MPC methods, demonstrating the developed method's superiority. The PID parameters were tuned for good performance at the given state of the water tank system. Figure 7 shows the control system's structure, where the PID controller's parameters are $K_{P1} = 18$, $K_{I1} = 1.2$, $K_{D1} = 0.2$ and $K_{P2} = 20$, $K_{I2} = 1.1$, $K_{D2} = 0.2$. Because the water tank system operates in different liquid levels, it exhibits varying dynamic behaviors. Thus, control experiments are performed in different water level regions, i.e., low, medium, and high water level zones. The parameter setting of the objective optimization function (22) for the predictive control method is $\mathbf{B}_1 = \text{diag}[0.0001, 0.0001]$, $\mathbf{B}_2 = \text{diag}[0.80, 0.80]$, A1= diag[1, 1]. The control experimental results of the water tank system are shown in Figure 11, where $y_1(a)$ and

$y_2(a)$ represent the liquid level heights, $u_1(a)$ and $u_2(a)$ represent the control valve opening (ranging from 0% to 100%), and $y_r(a)$ indicates that the expected output is a pink dashed line. In addition, Tables 3–5 display each algorithm's control outcomes, encompassing overshoot (O), peak time (PT), and adjustment time (AT); ⇑ and ⇓ represent the rising and falling step response, respectively.

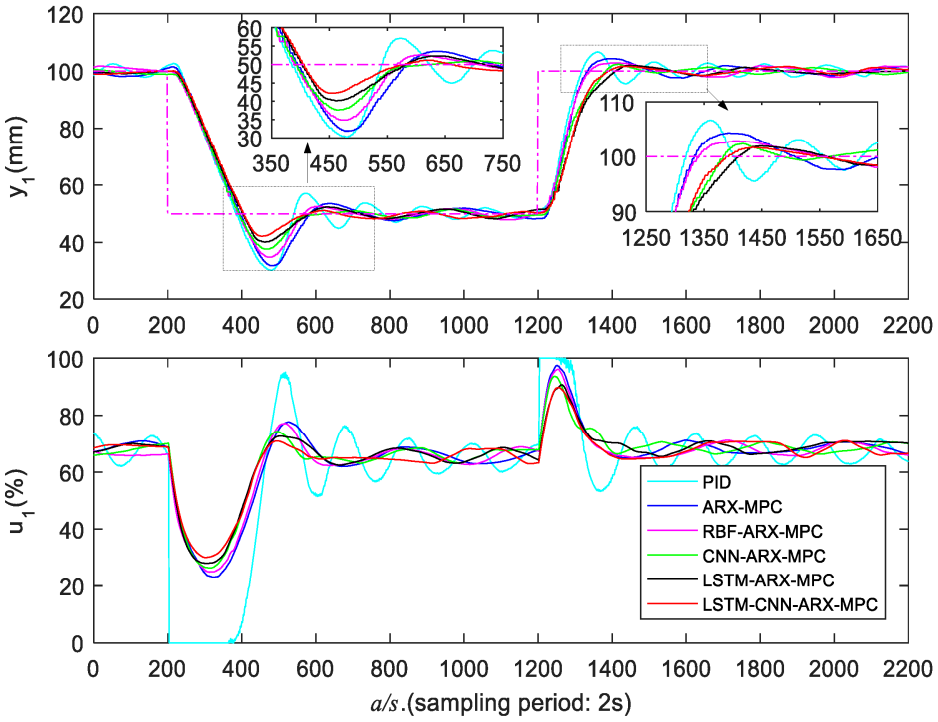



**Figure 11.** Low liquid level zone's experimental results ($y_1/u_1$).

**Table 3.** Low liquid level zone's control performance.

| Control Strategy | $y_1$ | | | $y_2$ | | |
|---|---|---|---|---|---|---|
| | PT (s) | O (%) | AT (s) | PT (s) | O (%) | AT (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 |
| CNN-ARX-MPC | 264/218 | 25.0/4.6 | 352/172 | 240/268 | 23.2/5.8 | 340/248 |
| LSTM-ARX-MPC | 262/248 | 19.6/3.8 | 348/186 | 236/238 | 19.0/4.0 | 326/184 |
| LSTM-CNN-ARX-MPC | 252/234 | 15.6/3.2 | 336/166 | 224/222 | 14.0/3.6 | 320/160 |

**Table 4.** Medium liquid level zone's control performance.

| Control Strategy | $y_1$ | | | $y_2$ | | |
|---|---|---|---|---|---|---|
| | PT (s) | O (%) | AT (s) | PT (s) | O (%) | AT (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 |
| CNN-ARX-MPC | 374/456 | 1.7/10.6 | 276/528 | 440/466 | 1.7/11.8 | 332/546 |
| LSTM-ARX-MPC | 372/450 | 1.3/9.1 | 278/502 | 454/454 | 1.4/10.3 | 328/512 |
| LSTM-CNN-ARX-MPC | 366/444 | 1.1/7.5 | 274/490 | 430/448 | 1.1/8.5 | 326/502 |

**Table 5.** High liquid level zone's control performance.

| Control Strategy | $y_1$ | | | $y_2$ | | |
|---|---|---|---|---|---|---|
| | PT (s) | O (%) | AT (s) | PT (s) | O (%) | AT (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 |
| CNN-ARX-MPC | 230/210 | 5.0/23.6 | 236/282 | 262/216 | 5.6/26.0 | 280/294 |
| LSTM-ARX-MPC | 258/214 | 4.0/16.8 | 184/272 | 272/208 | 4.2/18.0 | 206/278 |
| LSTM-CNN-ARX-MPC | 216/196 | 3.2/13.8 | 162/260 | 266/202 | 3.6/12.6 | 164/272 |

#### 5.3.1. Low Liquid Level Zone Control Experiments

The control outcomes of distinct algorithms in a low liquid level zone (50–100 mm) are presented in Figures 11 and 12 and Table 3, and the LSTM-CNN-ARX-MPC approach shows significantly superior control performance compared to other algorithms. In the falling step response of LSTM-CNN-ARX-MPC, the PTs of liquid levels $y_1(a)$ and $y_2(a)$ are 252 s and 224 s, the Os are 15.6% and 14.0%, and the ATs are 336 s and 320 s; in the rising step response, the Os are 3.2% and 3.6%, and the ATs are 166 s and 160 s, respectively, which are superior to other methods. This is due to the LSTM-CNN-ARX model incorporating the strengths of LSTM for learning temporal characteristics and CNN for capturing spatial characteristics, which can efficiently capture the water tank system's nonlinear characteristics and achieve outstanding control performance. It should be noted that the PT of LSTM-CNN-ARX-MPC may not always outperform other methods. However, the evaluation of control performance is not just about looking at PT; we also need to see other indicators, such as O and AT. In fact, the O and AT of LSTM-CNN-ARX-MPC are much smaller than those of other methods, so overall, LSTM-CNN-ARX-MPC outperforms other methods. Usually, for controllers, their smaller PT may result in a larger O and AT, and there may also be significant oscillations at the beginning of the control process.

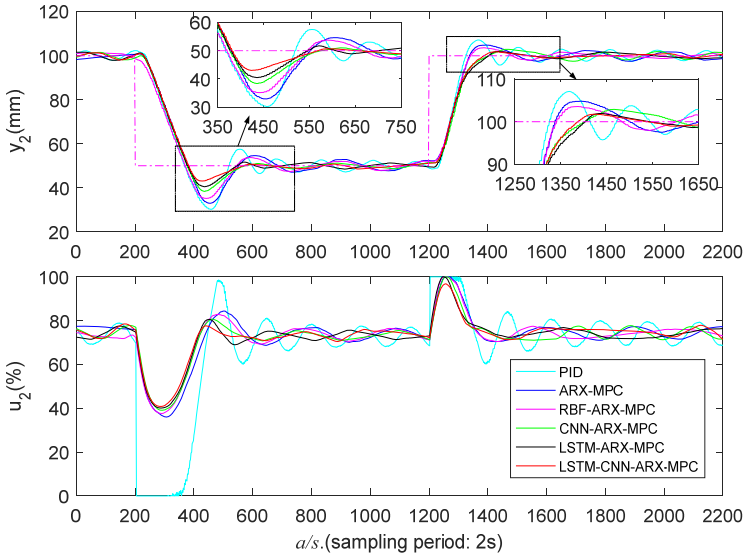



**Figure 12.** Low liquid level zone's experimental results ($y_2/u_2$).

#### 5.3.2. Medium Liquid Level Zone Control Experiments

Figures 13 and 14 and Table 4 present the experimental results of various controllers in a medium liquid level zone (100–250 mm), indicating that the PID's control performance is not as good as other methods; it has the maximum overshoot and AT. The control effectiveness of RBF-ARX-MPC outperforms ARX-MPC, exhibiting lower O and PT. However,

RBF-ARX-MPC demonstrates inferior O and AT compared to the three deep learning-based MPC algorithms. Multi-layer deep learning networks have stronger nonlinear expression and generalization capabilities than single-layer RBF neural networks and can adaptively learn the water tank system's nonlinear features. Therefore, the predictive control effect and modeling accuracy based on the deep learning ARX models are better than those of the RBF-ARX models. Additionally, LSTM-ARX-MPC's comprehensive control effect is marginally superior to CNN-ARX-MPC because LSTM belongs to a recurrent neural network, which has storage function and advantages in time series data modeling, so it can maintain long-term correlation in sequence data. LSTM-CNN-ARX-MPC outperforms all other algorithms in the falling step response, exhibiting the lowest PT, AT, and O. Additionally, it has the smallest O in the rising step response, which enables the water tank system's liquid level to effectively follow the reference trajectory.

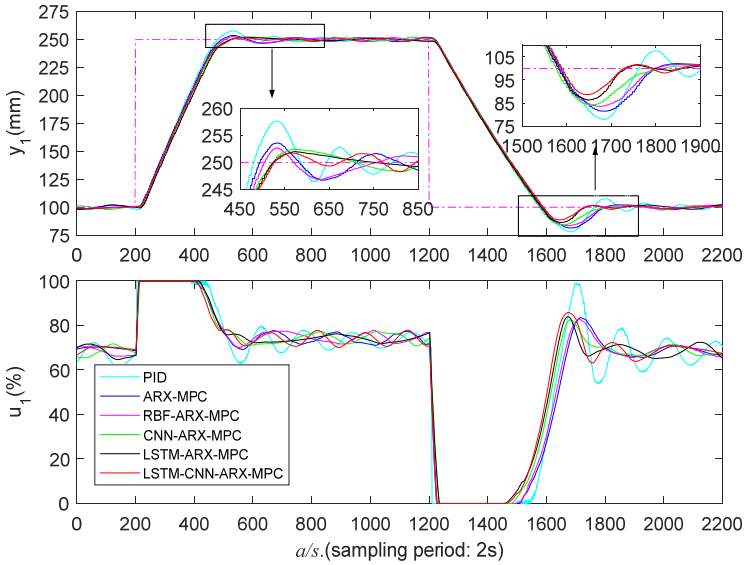



**Figure 13.** Medium liquid level zone's experimental results ($y_1/u_1$).

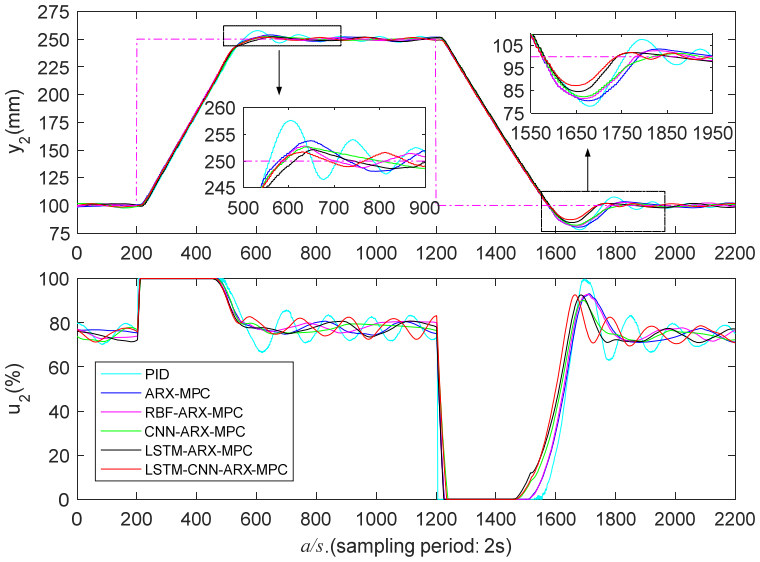



**Figure 14.** Medium liquid level zone's experimental results ($y_2/u_2$).

### 5.3.3. High Liquid Level Zone Control Experiments

Figures 15 and 16 and Table 5 present the experimental results in a high liquid level zone (250–300 mm). It is apparent that the PID controller exhibits the poorest control perfor-

mance with the largest O and AT. ARX-MPC's control performance is also poor because the ARX model has limited capability to capture nonlinear features. LSTM-ARX-MPC demonstrates superior overall control performance compared to ARX-MPC, RBF-ARX-MPC, and CNN-ARX-MPC due to the advantage of LSTM in maintaining long-term dependence on sequence data in time series modeling. LSTM-CNN-ARX-MPC demonstrates optimal comprehensive control effectiveness, particularly exhibiting a considerably lower O in the falling step. This is attributable to the LSTM-CNN-ARX model's powerful spatiotemporal feature capture capability, which enable it to effectively capture the water tank system's nonlinear properties.

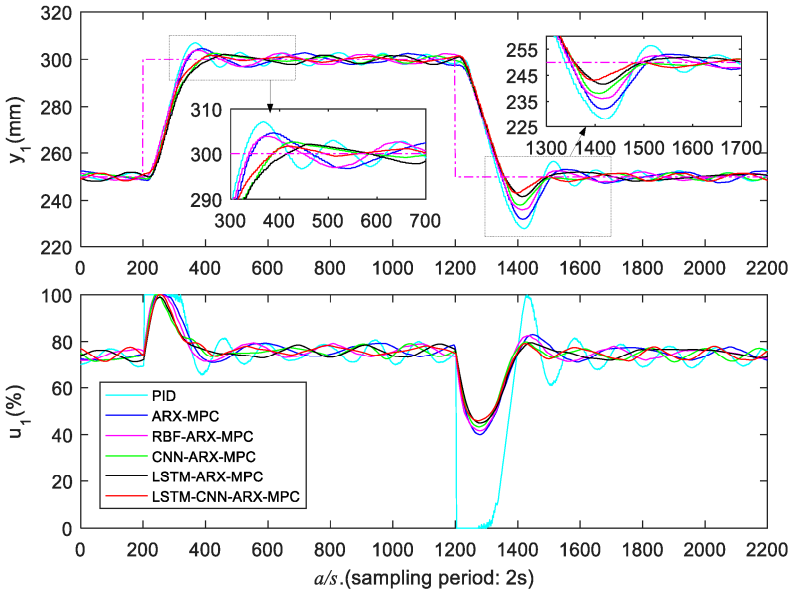



**Figure 15.** High liquid level zone's experimental results ($y_1/u_1$).

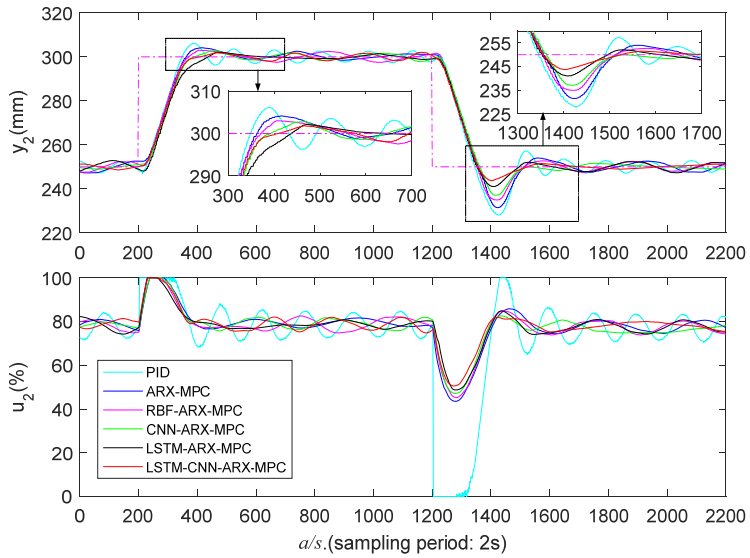



**Figure 16.** High liquid level zone's experimental results ($y_2/u_2$).

In conclusion, PID and ARX-MPC exhibit the poorest control performance, failing to effectively capture the nonlinear characteristics of the water tank system, thereby limiting their applicability. Additionally, the overall control effectiveness of RBF-ARX-MPC is not as good as deep learning-based MPC. This is because RBF is a single-layer network with lesser nonlinear descriptive capabilities compared to multi-layer deep learning networks. Further-

more, LSTM-ARX-MPC's superior control performance compared to CNN-ARX-MPC can be attributed to LSTM's memory function, which preserves the long-term dependence on sequential data. In almost all cases, LSTM-CNN-ARX-MPC demonstrates the best control effectiveness, especially in the low and high water level regions with strong nonlinearity. This is attributed to the powerful spatiotemporal feature learning capability of the LSTM-CNN-ARX model, which enable it to capture all the nonlinear features of the water tank system.

## 6. Conclusions

This study aims to solve the modeling and predictive control problems of a category of smooth nonlinear systems. The studied system's operating point may change over time, and its dynamic characteristics can be locally linearized. We utilized the spatiotemporal feature extraction capabilities of the LSTM and CNN, as well as the SD-ARX model's pseudo-linear ARX structural feature, to establish the LSTM-CNN-ARX model to describe the studied system's nonlinear properties. The developed modeling and MPC algorithms were validated in real-time control rather than digital simulation on an actual multivariable water tank system, confirming their effectiveness. The proposed methods can be useful and can generate better results for a category of smooth nonlinear plants than some well-established, simpler, but proven efficiency approaches.

This article used the deep learning model to fit the state-dependent ARX model's autoregressive coefficients to obtain the LSTM-ARX model and the LSTM-CNN-ARX model that accurately describe the system's nonlinear properties. These models have local linear and global nonlinear characteristics, which decompose the model's complexity into the autoregressive part, making it easier to design the MPC controllers. Control comparative experiments demonstrate that compared with the PID, ARX-MPC, and RBF-ARX-MPC algorithms, the three deep learning-based MPC algorithms can achieve more precise trajectory tracking control of the water tank system's liquid level. This is because of the stronger nonlinear expression and generalization ability of deep learning networks, which can adaptively learn system features and model parameters and better represent the nonlinear behavior of a water tank system. Furthermore, the comprehensive control performance of LSTM-CNN-ARX-MPC is superior to LSTM-ARX-MPC and CNN-ARX-MPC due to its incorporation of the strengths of LSTM for learning temporal characteristics and CNN for capturing spatial characteristics, which enables it to capture the multi-dimensional spatiotemporal nonlinear features of water tank systems more effectively.

While LSTM neural networks may effectively address the gradient vanishing or exploding problem in traditional RNN modeling and better handle long sequence data, the complex structure of the LSTM model with numerous unidentified parameters results in significant computational demands during model training, convergence difficulties, and the risk of overfitting. Consequently, future studies will concentrate on improving the model structure by replacing the LSTM structure with gated recurrent units or related improved structures to enhance modeling efficiency and expand the model's applicability in scenarios with limited computational resources.

**Author Contributions:** Conceptualization, T.K. and H.P.; methodology, T.K.; software, T.K.; validation, H.P.; data curation, X.P.; writing—original draft preparation, T.K.; writing—review and editing, H.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the Project of the National Natural Science Foundation of China (under Grant No. 61773402 and No. 52275105).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** All codes included in this study are available and can be obtained by contacting the author via email.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Mayne, D.Q. Model predictive control: Recent developments and future promise. *Automatica* **2014**, *50*, 2967–2986. [CrossRef]
2. Elnawawi, S.; Siang, L.C.; O'Connor, D.L.; Gopaluni, R.B. Interactive visualization for diagnosis of industrial Model Predictive Controllers with steady-state optimizers. *Control Eng. Pract.* **2022**, *121*, 105056. [CrossRef]
3. Kwapień, J.; Drożdż, S. Physical approach to complex systems. *Phys. Rep.* **2012**, *515*, 115–226. [CrossRef]
4. Chen, H.; Jiang, B.; Ding, S.X.; Huang, B. Data-driven fault diagnosis for traction systems in high-speed trains: A survey, challenges, and perspectives. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 1700–1716. [CrossRef]
5. Yu, W.; Wu, M.; Huang, B.; Lu, C. A generalized probabilistic monitoring model with both random and sequential data. *Automatica* **2022**, *144*, 110468. [CrossRef]
6. Pareek, P.; Verma, A. Piecewise Linearization of Quadratic Branch Flow Limits by Irregular Polygon. *IEEE Trans. Power Syst.* **2018**, *33*, 7301–7304. [CrossRef]
7. Ławryńczuk, M. Nonlinear predictive control of a boiler-turbine unit: A state-space approach with successive on-line model linearisation and quadratic optimisation. *ISA Trans.* **2017**, *67*, 476–495. [CrossRef] [PubMed]
8. Qian, K.; Zhang, Y. Bilinear model predictive control of plasma keyhole pipe welding process. *J. Manuf. Sci. Eng.* **2014**, *136*, 31002. [CrossRef]
9. Nie, Z.; Gao, F.; Yan, C.B. A multi-timescale bilinear model for optimization and control of HVAC systems with consistency. *Energies* **2021**, *14*, 400. [CrossRef]
10. Shi, Y.; Yu, D.L.; Tian, Y.; Shi, Y. Air-fuel ratio prediction and NMPC for SI engines with modified Volterra model and RBF network. *Eng. Appl. Artif. Intell.* **2015**, *45*, 313–324. [CrossRef]
11. Gruber, J.K.; Ramirez, D.R.; Limon, D.; Alamo, T. A convex approach for NMPC based on second order Volterra series models. *Int. J. Robust Nonlinear Control* **2015**, *25*, 3546–3571. [CrossRef]
12. Chen, H.; Chen, Z.; Chai, Z.; Jiang, B.; Huang, B. A single-side neural network-aided canonical correlation analysis with applications to fault diagnosis. *IEEE Trans. Cybern.* **2021**, *52*, 9454–9466. [CrossRef]
13. Chen, H.; Liu, Z.; Alippi, C.; Huang, B.; Liu, D. Explainable intelligent fault diagnosis for nonlinear dynamic systems: From unsupervised to supervised learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**. [CrossRef]
14. Ding, B.; Wang, J.; Su, B. Output feedback model predictive control for Hammerstein model with bounded disturbance. *IET Control. Theory Appl.* **2022**, *16*, 1032–1041. [CrossRef]
15. Raninga, D.; TK, R.; Velswamy, K. Explicit nonlinear predictive control algorithms for Laguerre filter and sparse least square support vector machine-based Wiener model. *Trans. Inst. Meas. Control* **2021**, *43*, 812–831. [CrossRef]
16. Wang, Z.; Georgakis, C. Identification of Hammerstein-Weiner models for nonlinear MPC from infrequent measurements in batch processes. *J. Process Control* **2019**, *82*, 58–69. [CrossRef]
17. Du, J.; Zhang, L.; Chen, J.; Li, J.; Zhu, C. Multi-model predictive control of Hammerstein-Wiener systems based on balanced multi-model partition. *Math. Comput. Model. Dyn. Syst.* **2019**, *25*, 333–353. [CrossRef]
18. Peng, H.; Ozaki, T.; Haggan-Ozaki, V.; Toyoda, Y. A parameter optimization method for radial basis function type models. *IEEE Trans. Neural Netw.* **2003**, *14*, 432–438. [CrossRef]
19. Zhou, F.; Peng, H.; Qin, Y.; Zeng, X.; Xie, W.; Wu, J. RBF-ARX model-based MPC strategies with application to a water tank system. *J. Process Contr.* **2015**, *34*, 97–116. [CrossRef]
20. Peng, H.; Nakano, K.; Shioya, H. Nonlinear predictive control using neural nets-based local linearization ARX model—Stability and industrial application. *IEEE Trans. Control Syst. Technol.* **2006**, *15*, 130–143. [CrossRef]
21. Kang, T.; Peng, H.; Zhou, F.; Tian, X.; Peng, X. Robust predictive control of coupled water tank plant. *Appl. Intell.* **2021**, *51*, 5726–5744. [CrossRef]
22. Peng, H.; Ozaki, T.; Toyoda, Y.; Shioya, H.; Nakano, K.; Haggan-Ozaki, V.; Mori, M. RBF-ARX model-based nonlinear system modeling and predictive control with application to a NOx decomposition process. *Control Eng. Pract.* **2004**, *12*, 191–203. [CrossRef]
23. Peng, H.; Wu, J.; Inoussa, G.; Deng, Q.; Nakano, K. Nonlinear system modeling and predictive control using the RBF nets-based quasi-linear ARX model. *Control Eng. Pract.* **2009**, *17*, 59–66. [CrossRef]
24. Xu, W.; Peng, H.; Tian, X.; Peng, X. DBN based SD-ARX model for nonlinear time series prediction and analysis. *Appl. Intell.* **2020**, *50*, 4586–4601. [CrossRef]
25. Inoussa, G.; Peng, H.; Wu, J. Nonlinear time series modeling and prediction using functional weights wavelet neural network-based state-dependent AR model. *Neurocomputing* **2012**, *86*, 59–74. [CrossRef]
26. Mu, R.; Zeng, X. A review of deep learning research. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 1738–1764. [CrossRef]
27. Lippi, M.; Montemurro, M.A.; Degli Esposti, M.; Cristadoro, G. Natural language statistical features of LSTM-generated texts. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3326–3337. [CrossRef]
28. Shuang, K.; Tan, Y.; Cai, Z.; Sun, Y. Natural language modeling with syntactic structure dependency. *Inf. Sci.* **2020**, *523*, 220–233. [CrossRef]
29. Shuang, K.; Li, R.; Gu, M.; Loo, J.; Su, S. Major-minor long short-term memory for word-level language model. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 3932–3946. [CrossRef]
30. Ying, W.; Zhang, L.; Deng, H. Sichuan dialect speech recognition with deep LSTM network. *Front. Comput. Sci.* **2020**, *14*, 378–387. [CrossRef]

31. Jo, J.; Kung, J.; Lee, Y. Approximate LSTM computing for energy-efficient speech recognition. *Electronics* **2020**, *9*, 2004. [CrossRef]
32. Oruh, J.; Viriri, S.; Adegun, A. Long short-term Memory Recurrent neural network for Automatic speech recognition. *IEEE Access* **2022**, *10*, 30069–30079. [CrossRef]
33. Yang, B.; Yin, K.; Lacasse, S.; Liu, Z. Time series analysis and long short-term memory neural network to predict landslide displacement. *Landslides* **2019**, *16*, 677–694. [CrossRef]
34. Hu, J.; Wang, X.; Zhang, Y.; Zhang, D.; Zhang, M.; Xue, J. Time series prediction method based on variant LSTM recurrent neural network. *Neural Process. Lett.* **2020**, *52*, 1485–1500. [CrossRef]
35. Peng, L.; Zhu, Q.; Lv, S.X.; Wang, L. Effective long short-term memory with fruit fly optimization algorithm for time series forecasting. *Soft Comput.* **2020**, *24*, 15059–15079. [CrossRef]
36. Langeroudi, M.K.; Yamaghani, M.R.; Khodaparast, S. FD-LSTM: A Fuzzy LSTM Model for Chaotic Time-Series Prediction. *IEEE Intell. Syst.* **2022**, *37*, 70–78. [CrossRef]
37. Wu, Z.; Rincon, D.; Luo, J.; Christofides, P.D. Machine learning modeling and predictive control of nonlinear processes using noisy data. *AICHE J.* **2021**, *67*, e17164. [CrossRef]
38. Terzi, E.; Bonassi, F.; Farina, M.; Scattolini, R. Learning model predictive control with long short-term memory networks. *Int. J. Robust Nonlinear Control* **2021**, *31*, 8877–8896. [CrossRef]
39. Zarzycki, K.; Ławryńczuk, M. Advanced predictive control for GRU and LSTM networks. *Inf. Sci.* **2022**, *616*, 229–254. [CrossRef]
40. Yu, W.; Zhao, C.; Huang, B. MoniNet with concurrent analytics of temporal and spatial information for fault detection in industrial processes. *IEEE Trans. Cybern.* **2021**, *52*, 8340–8351. [CrossRef]
41. Li, G.; Huang, Y.; Chen, Z.; Chesser, G.D.; Purswell, J.L.; Linhoss, J.; Zhao, Y. Practices and applications of convolutional neural network-based computer vision systems in animal farming: A review. *Sensors* **2021**, *21*, 1492. [CrossRef] [PubMed]
42. Yang, X.; Ye, Y.; Li, X.; Lau, R.Y.; Zhang, X.; Huang, X. Hyperspectral image classification with deep learning models. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 5408–5423. [CrossRef]
43. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Chen, T. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
44. Jiang, M.; Zhang, W.; Zhang, M.; Wu, J.; Wen, T. An LSTM-CNN attention approach for aspect-level sentiment classification. *J. Comput. Methods Sci. Eng.* **2019**, *19*, 859–868. [CrossRef]
45. Kumar, B.S.; Malarvizhi, N. Bi-directional LSTM-CNN combined method for sentiment analysis in part of speech tagging (PoS). *Int. J. Speech Technol.* **2020**, *23*, 373–380. [CrossRef]
46. Priyadarshini, I.; Cotton, C. A novel LSTM-CNN-grid search-based deep neural network for sentiment analysis. *J. Supercomput.* **2021**, *77*, 13911–13932. [CrossRef]
47. Jang, B.; Kim, M.; Harerimana, G.; Kang, S.U.; Kim, J.W. Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism. *Appl. Sci.* **2020**, *10*, 5841. [CrossRef]
48. Shrivastava, G.K.; Pateriya, R.K.; Kaushik, P. An efficient focused crawler using LSTM-CNN based deep learning. *Int. J. Syst. Assur. Eng. Manag.* **2022**, *14*, 391–407. [CrossRef]
49. Zhu, Y.; Gao, X.; Zhang, W.; Liu, S.; Zhang, Y. A bi-directional LSTM-CNN model with attention for aspect-level text classification. *Future Internet* **2018**, *10*, 116. [CrossRef]
50. Zhen, H.; Niu, D.; Yu, M.; Wang, K.; Liang, Y.; Xu, X. A hybrid deep learning model and comparison for wind power forecasting considering temporal-spatial feature extraction. *Sustainability* **2020**, *12*, 9490. [CrossRef]
51. Farsi, B.; Amayri, M.; Bouguila, N.; Eicker, U. On short-term load forecasting using machine learning techniques and a novel parallel deep LSTM-CNN approach. *IEEE Access* **2021**, *9*, 31191–31212. [CrossRef]
52. Hoiberg, J.A.; Lyche, B.C.; Foss, A.S. Experimental evaluation of dynamic models for a fixed-bed catalytic reactor. *AICHE J.* **1971**, *17*, 1434–1447. [CrossRef]
53. O'Hagan, A. Curve fitting and optimal design for prediction. *J. R. Stat. Soc. Ser. B Methodol.* **1978**, *40*, 1–24. [CrossRef]
54. Priestley, M.B. State-dependent models: A general approach to non-linear time series analysis. *J. Time Ser. Anal.* **1980**, *1*, 47–71. [CrossRef]
55. Ketkar, N. Introduction to Keras. In *Deep Learning with Python*; Apress: Berkeley, CA, USA, 2017; pp. 97–111. [CrossRef]