agronomy

MDPI

Article

A Reversible Automatic Selection Normalization (RASN) Deep Network for Predicting in the Smart Agriculture System

Xuebo Jin [1,2], Jiashuai Zhang [1], Jianlei Kong [1,2,*], Tingli Su [1] and Yuting Bai [1]

[1] Artificial Intelligence College, Beijing Technology and Business University, Beijing 100048, China; jinxuebo@btbu.edu.cn (X.J.); zhangjiashuai@st.btbu.edu.cn (J.Z.); sutingli@btbu.edu.cn (T.S.); baiyuting@btbu.edu.cn (Y.B.)

[2] National Engineering Laboratory for Agri-Product Quality Traceability, Beijing 100048, China

* Correspondence: kongjianlei@btbu.edu.cn

Abstract: Due to the nonlinear modeling capabilities, deep learning prediction networks have become widely used for smart agriculture. Because the sensing data has noise and complex nonlinearity, it is still an open topic to improve its performance. This paper proposes a Reversible Automatic Selection Normalization (RASN) network, integrating the normalization and renormalization layer to evaluate and select the normalization module of the prediction model. The prediction accuracy has been improved effectively by scaling and translating the input with learnable parameters. The application results of the prediction show that the model has good prediction ability and adaptability for the greenhouse in the smart agriculture system.

Keywords: normalization; time series prediction; reversible normalization; deep learning; automatic normalization; smart agriculture system

Citation: Jin, X.; Zhang, J.; Kong, J.; Su, T.; Bai, Y. A Reversible Automatic Selection Normalization (RASN) Deep Network for Predicting in the Smart Agriculture System. *Agronomy* **2022**, *12*, 591. https://doi.org/10.3390/agronomy12030591

Academic Editor: Andrea Sciarretta

Received: 22 January 2022
Accepted: 25 February 2022
Published: 27 February 2022

1. Introduction

The smart agricultural system combines mobile Internet, cloud computing, and Internet of Things (IoT) technologies. Thanks to various sensor nodes and wireless communication networks such as environmental temperature and humidity, soil moisture, CO_2, and images, massive time series can be obtained [1,2], shown in Figure 1. Based on the sensing time series data, prediction and analytics have been used in agronomy for precision agriculture tools and as parts of decision support systems for farm management [3–5]. Accurate and reliable prediction can provide information for production planning and control.

Currently, time series prediction methods can be broadly classified into three categories: traditional statistical models, machine learning, and deep learning methods [6,7]. Traditional statistical models are highly interpretable, but their modeling requires high a priori knowledge. It is difficult to determine the model's predefined parameters, and the applicability and flexibility of model prediction are limited. The time series models are based on statistical data, and the model parameters can be estimated through identification methods [8–17], such as the hierarchical algorithms [18–22]. Machine learning methods, such as shallow neural networks, do not require a priori physical knowledge or assumptions and can handle data with nonlinear characteristics. Still, their simple structures suffer from slow learning speed and overfitting. Deep learning methods are currently the most widely used methods for time series prediction, which have a strong fitting capability, are big data-driven, and yield more advanced results than previous methods [23–25].

However, the current deep learning prediction methods also face difficulties in the practical smart agriculture system because the data collected by sensors have noise and outliers, which affect the fitting of the model, even leading to overfitting or underfitting of the model [26,27]. Further, the input data for time series prediction usually have different magnitudes, i.e., the large differences between the values, which will cause a slow model

training speed and a large model training error. Therefore, it is necessary to use the preprocessing stage before training the deep learning networks.

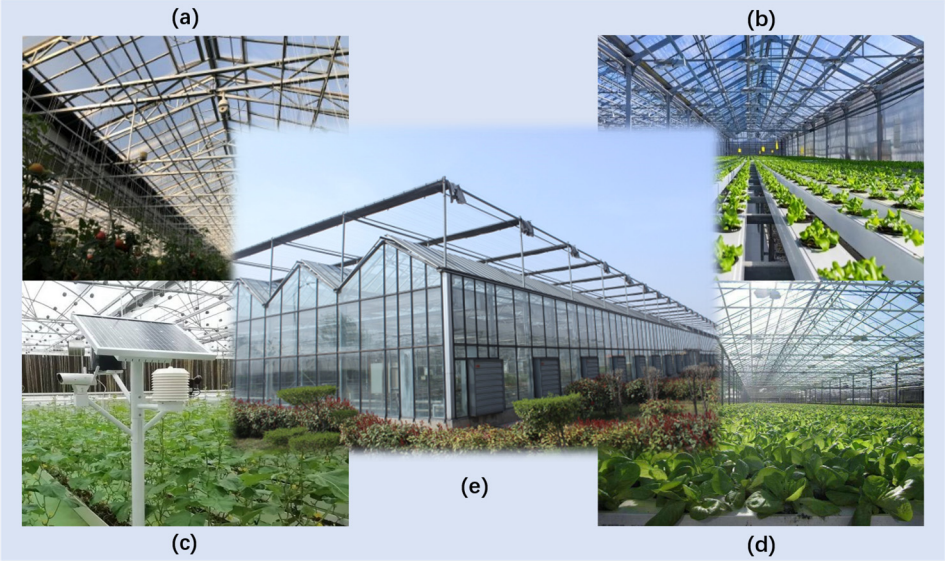

Figure 1. Schematic diagram of data collection environment. (**a**–**d**) show the sensors used to collect data on temperature, humidity, light, CO_2, and light intensity at the top, ground, front, and back of the greenhouse, respectively; (**e**) shows a panoramic view of the greenhouse.

As one of the preprocessing techniques to remove the magnitudes difference, normalization methods have been widely used for neural networks. The classical normalization methods include Min-Max normalization [28], z-score normalization [29], etc. The purpose of normalization is to make the preprocessed data limited to a certain range (e.g., [0, 1] or [–1, 1]) to eliminate the adverse effects caused by data magnitudes and outliers. Normalizing deep learning can improve the model convergence speed [30] and accuracy [31]. Zou et al. [32] compared four methods, including Min-Max, z-score, logarithmic, and standard normalization, proving that different normalization in the preprocessing stage has a large impact on the prediction results; therefore, data normalization methods should be chosen carefully.

To avoid the drawbacks of a single normalization method, Dalwinder Singh et al. [33] proposed the feature-wise normalization method, in which each input data is normalized independently by using different methods, and a normalization unit is developed to combine the multiple methods so that the data can be better normalized. Additionally, Sukirty Jain et al. [34] used normalization selection techniques. In the above study, the normalization is in the data preprocessing stage and was not a component of the deep learning network. Moreover, the normalized data is fixed and cannot be adjusted through learning.

The researchers believed that if the normalization method is set as a layer of the deep neural network, the normalization will be learned and more adaptive to the characteristics of the input data. Sergey Ioffe et al. [35] added the batch normalization layer to the deep network, which uses the z-score method to learn how to perform normalization on each layer. Passalis et al. [36] designed an adaptive input normalization layer that was trained using an end-to-end backpropagation approach and learned how to perform normalization, with a significant performance improvement compared with the fixed normalization. Tomar et al. [37] used a deep convolutional generator network, which can achieve cross-modal image transformation results by intermediately activated self-attention space normalization. Park et al. [38] designed a spatially adaptive normalization layer to implement synthetic images based on the semantic layout of a given input. The experimental results show that the introduction of the normalization layer is effective. Still, the network mentioned above [35–38] only applies an image-based normalization method,

and the network cannot be used for prediction problems in smart agricultural systems directly [39,40].

In order to improve the adaptability to the input data characteristics, the researchers designed switchable normalization layers (SN). Luo et al. [41] combined three normalization methods, batch, channel, and layer normalization, to calculate the mean and variance of the input data. The performance of the SN layer was verified on benchmark datasets such as ImageNet, COCO, CityScapes, ADE20K, Kinetics, etc. Shao et al. [42] designed the sparse SN to reduce the computational redundancy. Yang et al. [43] applied an SN block in the DeepLabV3 segmentation model to mitigate feature divergence in RGB and NIR images. From the above research results, we can see that the SN method can adapt the deep network to complex data features and improve the modeling ability of the network. However, these networks are all applied to image classification [44]. Unlike image recognition and general classification tasks, the time series prediction task in smart agriculture applications is a regression modeling problem. The output is not a classification probability but requires inverse normalization. Inverse normalization requires mean and variance in normalization, which limits the switching and fusion operations of the normalization layer.

Therefore, this paper proposes a Reversible Automatic Selection Normalization (RASN) prediction network for the smart agriculture system with the following innovation.

(1) Design adaptive and inverse normalization layers based on four normalization methods and backpropagation principles. Compared with traditional techniques, it can improve the adaptive ability of normalization methods and retain the data expression ability;

(2) Design the automatic selection module of normalization methods, comprehensively compare the prediction effects of four normalization methods, and output the most suitable normalization method and prediction results.

The other sections of this paper are organized as follows. Section 2 introduces the model framework designed in this paper and explains the operation of the model. Section 3 uses the method proposed in this paper to predict temperature, humidity, CO_2, light intensity, etc., for the smart agricultural system in Weifang, Shandong, China. The experimental results show that the proposed method can improve the accuracy of environmental information prediction of smart agricultural systems, and finally, Section 4 draws conclusions.

2. Materials and Methods

2.1. Networks Structure

This paper uses four normalization calculation methods, including Min-Max, interval, decimal calibration, and z-score normalization. All four normalization methods have their advantages and disadvantages. They can achieve different normalization purposes while maintaining the correlation between the input and output data. The four normalization methods can lead to different changes in data and the loss of some hidden information, which may lead to deviations in the model fitting to the normalized data.

The comparison of the data obtained by the four normalization methods with the source data is shown in Figure 2. Min-Max normalization completely normalizes the data to the range of (0–1), but it loses the negative information of the data. In contrast, interval and decimal calibration normalize the data between (−0.8–0.8) and (−0.1–0.1), preserving the negative number information of the source data. The z-score method maintains the 0-mean case of the values, but the values are in the range of (−2–2), which is not friendly for the fitting process of the normalized neural network.

The analysis of the effect of different normalizations using the correlation analysis method [45,46] shows that the data processed by the four methods have a different correlation between the input data and the output data of the networks.

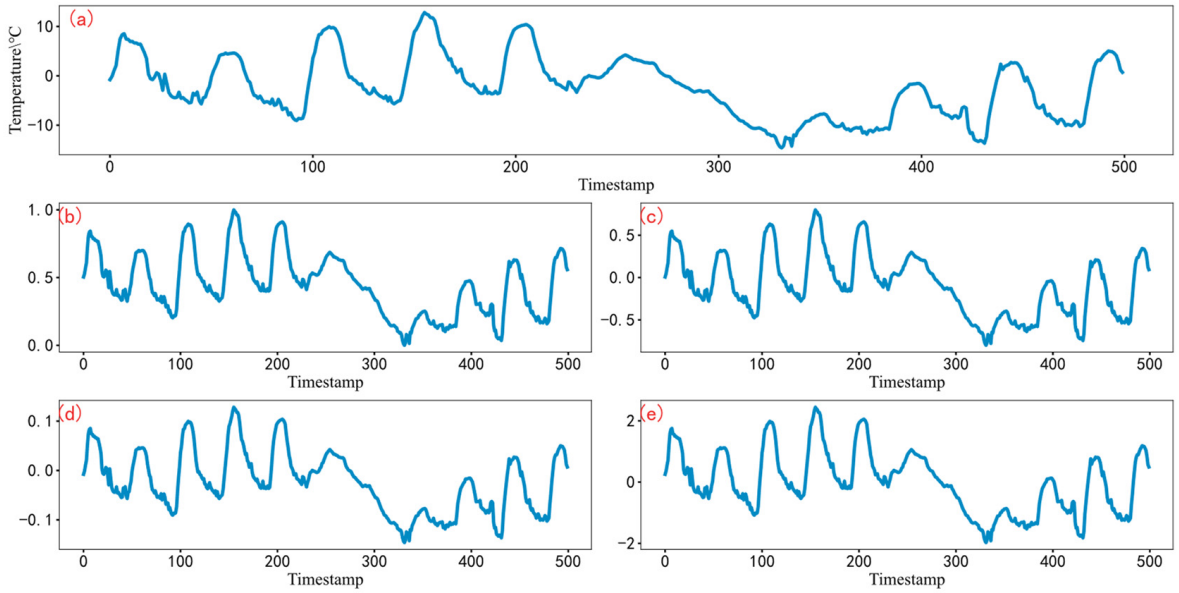

Figure 2. Comparison of the effects of four normalization methods; (**a**) the input data; (**b**) the effect after using Min-Max normalization, it normalizes the data to the range of (0–1); (**c**) the effect after using interval normalization, it normalizes the data to the range of (−0.8–0.8); (**d**) the effect after decimal calibration normalization, it normalizes the data to the range of (−0.1–0.1), and (**e**) the effect after z-score normalization, it normalizes the data to the range of (−2–2).

The schematic diagram of the correlation analysis method is shown in Figure 3. For example, assume that we want to predict indoor temperature (Y) by outdoor temperature (X), so for the prediction networks, X is input data, and Y is output data. Let us examine the changes in correlation after applying different normalization methods. As shown in Table 1, the correlation between X and Y is 0.660778. Use four normalization methods, including Min-Max, Interval, Decimal calibration, and z-score, to normalize X. The obtained results are represented by X_M, X_I, X_D, and Xz, respectively. Then, let us consider the correlation between X_M, X_I, X_D, Xz, and Y, respectively. It can be seen that Min-Max and Interval normalization enhance the correlation from 0.660778 to 0.670713 and 0.670713, respectively, and z-score has little effect (the correlation is 0.660724, almost the same as the correlation between X and Y 0.660778).

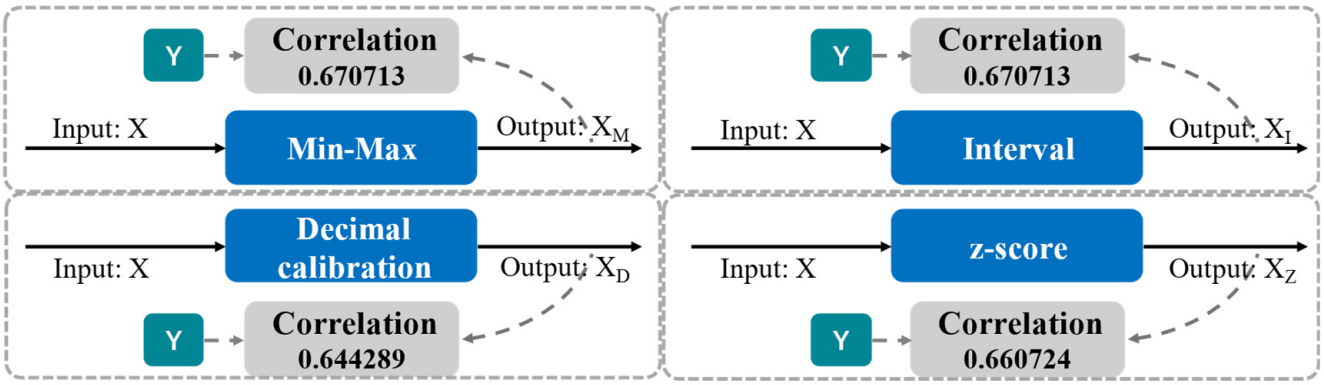

Figure 3. Schematic diagram of correlation analysis method.

In contrast, the Decimal calibration method reduces the correlation to 0.644289. The highly correlated input and output data can make the modeling performance of the prediction network better, and more accurate prediction performance can be obtained. Therefore, in a smart agricultural system, when we use X as input data and Y as output data to train the prediction network, we need to choose the Min-Max normalization or Interval normalization, and Decimal calibration should be avoided. So, we can find that it is necessary to select the normalization method and find the most suitable method to make the normalized model predict better.

Table 1. Correlation by four normalization methods.

Methods	Between X and Y	Between X_M and Y (Min-Max Normalization)	Between X_I and Y (Interval Normalization)	Between X_D and Y (Decimal Calibration)	Between X_Z and Y (z-Score Normalization)
Correlation	0.660778	0.670713	0.670713	0.644289	0.660724

The proposed model in this paper contains four main parts. The first part is the adaptive normalization layer (The details are mentioned in Section 2.2). The input data are divided into minibatches to enter the model. Then, the fundamental values such as mean and variance necessary for the normalization are calculated and updated. The updating methods are all exponentially weighted averages, which can memorize the past data and gradually fit the global distribution. Then, the normalization layer contains four normalization calculation methods: Min-Max, Interval, Decimal calibration, and z-score. Min-Max normalizes the data to between (0–1), while z-score treats the data to a state with a mean of 0 and a variance of 1. The Decimal calibration normalization method can quickly scale the data and better affect negative numbers without removing the positive and negative data. The interval normalization is similar to the Decimal calibration, which can retain the negative characteristics of the data and normalize the data to the intense action interval of the activation function. The results of the four normalization methods need to be calculated with the parameters of the normalization layer to output the normalized results. The parameters, the scaling, and translation factors, can realize the scaling and translation of the normalized values. The values can be trained by backpropagation so that the adaptive normalization can be realized.

The second part is a deep learning model based on Gated Recurrent Unit (GRU), a variant of recurrent neural network (RNN) with a simple structure and the ability to retain long-term information of time series, which has proven its effectiveness in many applications [47,48]. The model contains three GRU layers and two Dropout layers [49], and the predicted values are output by a fully connected layer.

The third part is the adaptive renormalization layer (The details are mentioned in Section 2.3). It also corresponds to each of the four calculations in the normalization layer, using the inverse operation corresponding to the normalization. In order to perform the inverse operation, we use the renormalization layer to obtain the min and max of the batch data for the normalization calculation and obtain the renormalization result accordingly. This layer also sets trainable parameters similar to the normalization layer, used to scale and translate the renormalization results. The renormalization can also be trained by backpropagation so that adaptive renormalization can be achieved.

The fourth part is the normalization method selection module (details in Section 2.4). Since different normalization methods have different effects on data processing, we need to select the appropriate normalization method according to the model prediction performance. This module compares the root mean square error (RMSE) and mean absolute error (MAE) of the prediction results given by the four normalization methods and finally selects the normalization method with the slightest error and outputs the prediction results of that method. The model structure diagram is shown in Figure 4.

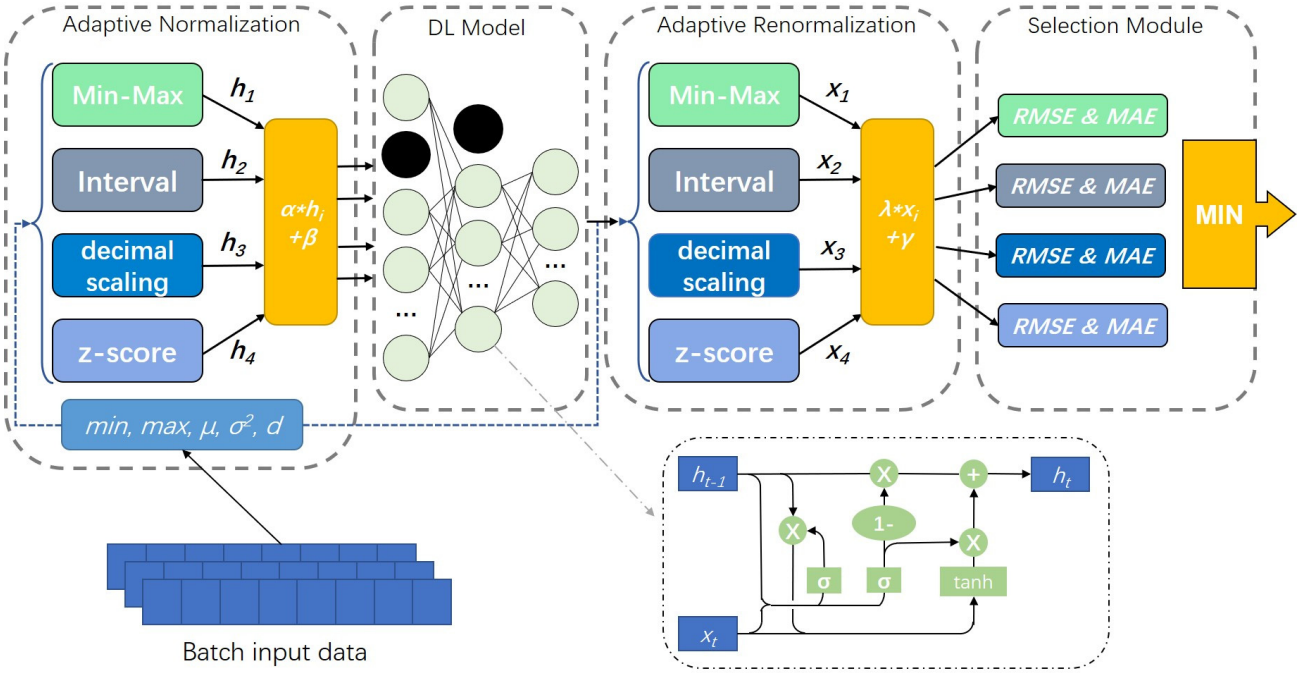

Figure 4. The RASN structure. The networks take over the input of the minibatch and later unify the data into a two-dimensional format; *min*, *max*, μ, σ^2 and d represent the calculated minimum, maximum, mean, variance, and fractional displacement values of the batch. These key computational parameters are shared between the normalization and denormalization layers (indicated by the blue dashed line); h_i and x_i represent the output of the normalization and renormalization layers, both processed by the scaling and translation factors. The activation functions used in the deep learning (DL) networks model are all tanh activation functions to reduce the influence on the model results.

2.2. Adaptive Normalization Layer

This layer includes the four normalization calculation methods in Equations (1)–(4).

$$\hat{x} = \frac{x - min}{max - min} \tag{1}$$

$$\hat{x} = a + \frac{(b-a)(x - min)}{max - min} \tag{2}$$

$$\hat{x} = \frac{x}{10^{\lceil \log_{10}|x|_{max} \rceil}} \tag{3}$$

$$\hat{x} = \frac{x - mean}{\sqrt{var + 1 \times 10^{-5}}} \tag{4}$$

where x represents the original data, $\hat{x}$ represents the normalized data, *min*, *max*, *mean*, *var* represents the maximum, minimum, mean, and variance of the original data, respectively. a, b represent the intervals on both sides that need to be normalized.

The data can be directly input into the deep learning networks. After processing by the adaptive normalization layer, the normalized output results of the four methods can be obtained, which is convenient for the subsequent comparison of normalization effects.

The input data of time series for prediction are generally two-dimensional or three-dimensional. In order to facilitate the normalization calculation, the normalization layer first unifies the format of the input data into two dimensions. Subsequently, the input data's key values (minimum *min*, maximum *max*, mean μ, variance σ^2, and decimal shift d) are calculated and saved. They are necessary to complete the normalization and renormalization calculations, so they must be shared between the normalization and renormalization layers. Since the normalization layer designed in this paper is combined inside the deep

learning model and takes over the data of the batch, in the training phase of the model, the key values of the batch are calculated each time, and the size of the values will change with the batch. The model parameters have been fixed in the testing phase, and the fundamental values such as mean and variance are matched. The exponentially weighted average method [50] is used to record the mean and variance of each batch so that the values used in the testing phase are close to the distribution of the total sample, and the calculation formulae are shown in Equations (5)–(9).

$$running\_mean_t = m * running\_mean_{t-1} + (1-m)\mu_t \tag{5}$$

$$running\_var_t = m * running\_var_{t-1} + (1-m){\sigma^2}_t \tag{6}$$

$$running\_max_t = m * running\_max_{t-1} + (1-m) * max_t \tag{7}$$

$$running\_min_t = m * running\_min_{t-1} + (1-m) * min_t \tag{8}$$

$$running\_d_t = m * running\_d_{t-1} + (1-m) * d_t \tag{9}$$

where $running\_mean_t$ and $running\_mean_{t-1}$ represent updating the value of mean μ at moments t and $t-1$, $running\_var_t$, $running\_var_{t-1}$ represent updating the moment values of variance σ^2, $running\_max_t$, $running\_max_{t-1}$ represent updating the moment values of max, $running\_min_t$, $running\_min_{t-1}$ represent updating the moment values of min, $running\_d_t$, $running\_d_{t-1}$ represent updating the moment values of d. m represents the weight of retaining information from previous moments, set to 0.6.

In addition, to make the normalization layer better adaptable to complex data, we added learnable parameters and scaling and translation factors, respectively. The two parameters can be back-propagated to be updated in an end-to-end training manner during training. The fixed normalized output can be scaled and panned at the output of the normalization layer to make the data more expressive and improve the model fit to the input data. The expression at the output of the normalization layer can be expressed as:

$$Y_i = \alpha h_i + \beta \tag{10}$$

where Y_i denotes the output of the batch's normalization layer, denotes the batch's value after the normalization calculation, α is the scaling factor, and β is the translation factor.

Finally, to make the normalization layer's output acceptable to the GRU-based deep learning network, the final output needs to be converted to the data format. If the source data is three-dimensional, then it only needs to be reduced to the source data format. Furthermore, if the input data is two-dimensional, then the output needs to be extended to three-dimensional. In this way, the output values of the normalization layer can be directly fed into the deep learning model for training. Algorithm 1 shows the algorithm for the normalization layer.

Algorithm 1 Normalization Layer Algorithm Framework

Input: minibatch : $R = \{x_1, \ldots, x_m\}$, Interval : a, b, Forgetting weight: m, Selection parameter: mode; Learning parameters : α, β

Output: $\left\{ y_i = \mathrm{RASNLayer}_{\alpha,\beta}(x_i) \right\}$

$min \leftarrow x_{min}, max \leftarrow x_{max}$, $\mu_R \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$, $\sigma_R^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_R)^2$, $d = 10^{\lceil \log_{10}|x|_{max} \rceil}$

$running\_max_t \leftarrow m * running\_max_{t-1} + (1-m) * max$ // Update *max* parameters

$running\_min_t \leftarrow m * running\_min_{t-1} + (1-m) * min$ // Update *min* parameters

$running\_d_t \leftarrow m * running\_d_{t-1} + (1-m) * d$ // Update *d* parameters

if mode = 'minmax' **then** //Min-Max normalization

$output_i \leftarrow \frac{x_i - running\_min}{running\_max - running\_min}$

if mode = 'Interval' **then** //Interval normalization

$output_i \leftarrow a + \frac{(b-a)(x_i - running\_min)}{running\_max - running\_min}$

if mode = 'calibrate' **then** // Decimal calibration normalization

$output_i \leftarrow \frac{x_i}{running\_d}$

if mode = 'z-score' **then** //z-score normalization

if flag = 'train' **then**

$output_i \leftarrow \frac{x_i - \mu_R}{\sqrt{\sigma_R^2 + 1 \times 10^{-5}}}$

$running\_mean_t \leftarrow m * running\_mean_{t-1} + (1-m)\mu_R$

$running\_var_t \leftarrow m * running\_var_{t-1} + (1-m)\sigma_R^2$ // Update *var* parameters

else

$output_i \leftarrow \frac{x_i - running\_mean_t}{\sqrt{running\_\mathrm{var}_t + 1 \times 10^{-5}}}$

$y_i \leftarrow output_i * \alpha + \beta \equiv \mathrm{RASNLayer}_{\alpha,\beta}(x_i)$

2.3. Adaptive Inverse Normalization Layer

The renormalization layer is also integrated into the model. It contains four renormalization methods: Min-Max, Interval, decimal calibration, and z-score renormalization, which correspond to the four methods in the normalization layer and can be used to renormalize the output values of each normalization operation. The corresponding renormalization formulas are:

$$x = \hat{x} * (max - min) + min \tag{11}$$

$$x = \frac{(max - min)(\hat{x} - a)}{b - a} + min \tag{12}$$

$$x = \hat{x} * 10\hat{}\lceil log_{10}|x|_{max} \rceil \tag{13}$$

$$x = \hat{x} * \sqrt{var + 1 \times 10^{-5}} + mean \tag{14}$$

where x represents the data after renormalization, $\hat{x}$ represents the data without renormalization, and min, max, $mean$, var represents the four parameters of maximum value, minimum value, mean value, and variance in the original data. Then, represents the Interval to which both sides need to be normalized.

To perform the inverse operation, we must know the key values for the normalization calculation, including min, max, μ, σ^2, and the decimal shift value d. Therefore, the renormalization layer first obtains these five parameters as known quantities, and when performing the renormalization operation, the corresponding renormalization calculation method is invoked using the values entered into the layer, and the parameters in the formula use the key values obtained during the normalization calculation.

Furthermore, we add learnable parameters λ and ν to the renormalization layer as scaling and translation factors, respectively. At the output of the inverse normalization layer, the two parameters can scale and translate the fixed renormalization output to

improve the adaptability of the model to nonstationary data. Thus, the expression at the output of the renormalization layer can be expressed as:

$$y_i = \lambda x_i + \nu \tag{15}$$

where y_i denotes the actual value of the output of the inverse normalization layer of the batch, x_i denotes the value of the batch after the renormalization calculation, λ is the scaling factor, ν is the translation factor.

After adding the renormalization layer, the obtained result needs to be transformed into the desired data format by the scaling factor and translation factor to perform the renormalization operation, so the data format of both λ and ν is needed to be carefully considered. Algorithm 2 shows the algorithm for the renormalization layer.

Algorithm 2 Renormalization Layer Algorithm Framework

Input: DL Model's output : $\hat{R}= \{\hat{x}_1, \ldots, \hat{x}_m\}$, Interval : a, b,
Selection parameter : mode, Learning parameters : λ, ν
Output: $\left\{\hat{y}_i = \text{RASNLayer2}_{\lambda,\nu}(\hat{x}_i)\right\}$
if mode = 'minmax' **then** //Min-Max renormalization
$output_i \leftarrow \hat{x}_i * (running\_max - running\_min) + running\_min$
if mode = 'Interval' **then** // Interval renormalization
$output_i \leftarrow \frac{(running\_max - running\_min)(\hat{x}_i - a)}{b-a} + running\_min$
if mode = 'calibrate' **then** // Decimal calibration renormalization
$output_i \leftarrow \hat{x} * running\_d$
if mode = 'z-score' **then** // z-score renormalization
if flag = 'train' **then**
$output_i \leftarrow \hat{x} * \sqrt{{\sigma_R}^2 + 1 \times 10^{-5}} + \mu_R$
else
$output_i \leftarrow \hat{x} * \sqrt{running\_var + 1 \times 10^{-5}} + running\_mean$
$\hat{y}_i \leftarrow output_i * \lambda + \nu \equiv \text{RASNLayer2}_{\lambda,\nu}(\hat{x}_i)$

2.4. Normalization Method Selection Module

Different normalization methods have very different effects on data processing and must be selected reasonably for the characteristics of the data. In this paper, after the adaptive normalization layer and the inverse normalization layer, a normalization method selection module is added to enable the selection of normalization methods.

This module is mainly implemented by comparing the prediction performance of the four normalization methods. Since the adaptive normalization layer and the inverse normalization layer designed in this paper contain four normalization methods for different types of data, during the training and testing of the model, the four normalization methods process the source data separately and input the results into the GRU-based deep learning model to finally obtain the prediction results of the four methods. At this time, the normalization method selection module obtains the prediction results of the four methods, evaluates the prediction performance using root mean square error (RMSE) and mean absolute error (MAE), and judges the optimal normalization method by comparing the values of evaluation indexes. The process within the module is:

$$\begin{aligned} \text{RMSE}_{\text{mode}} &= \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \\ \text{MAE}_{\text{mode}} &= \frac{1}{m}\sum_{i=1}^{m}|(y_i - \hat{y}_i)| \\ \text{mode} &\leftarrow Min(\text{RMSE}_{\text{mode}}, \text{MAE}_{\text{mode}}) \end{aligned} \tag{16}$$

where mode is the normalization method, m is the total number of values, $\hat{y}$ represents the true value of the data, y is the prediction result given by the model, $\text{RMSE}_{\text{mode}}$ is

the RMSE value of each normalization method, and MAE_{mode} is the MAE value of each normalization method. The module finally outputs the best normalization method and its prediction value.

3. Experiments and Analysis

3.1. Experimental Dataset

The indoor data from the smart greenhouses in Weifang, Shandong, China, were used in this experiment and collected by each environmental monitoring station and IoT sensors developed by the Beijing Agricultural Intelligent Equipment Technology Research Center itself [51,52]. Each greenhouse is equipped with an intelligent management system to collect, analyze, and process massive environmental information inside and outside the entire greenhouse. The measurement of various parameters of the greenhouse has been recorded and transmitted to the cloud platform via serial port and stored in the database of the management system. Then, these data are automatically transmitted to the backend cloud server through CAN/4G/WiFi and other forms of communication at regular intervals and stored in the database of the management system through the serial port at any time [53].

Accurate prediction of temperature, humidity, CO_2, and light intensity can plan the growth state of indoor crops and intelligently control them to improve greenhouse production [54]. In this experiment, temperature, humidity, CO_2, and light intensity data inside the greenhouse from 1 August 2020 to 20 January 2021 is used. The selected data is shown in Figure 5**.**

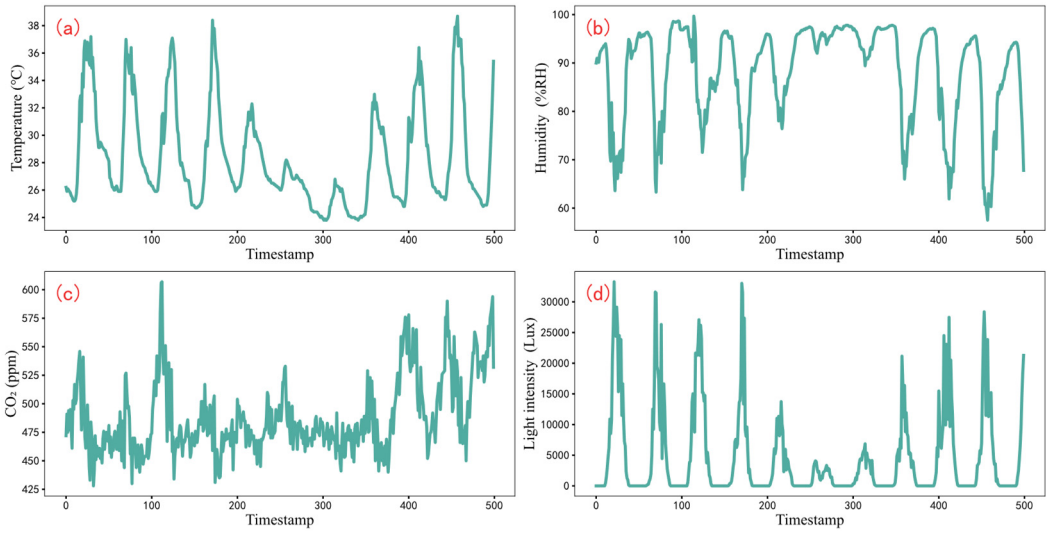

Figure 5. Display of data used in the experiment. (**a**) the temperature data, (**b**) the humidity data, (**c**) the CO_2 content data, and (**d**) the light intensity data.

3.2. Evaluation Indexes

In order to evaluate the reliability of the predicted values given by the model and quantify the performance of the model, this paper uses five evaluation indexes: root mean square error RMSE, mean absolute error MAE, mean square error MSE, mean absolute percentage error MAPE, and Pearson correlation coefficient R. Among them, the smaller the value of the four evaluation indexes RMSE, MAE, MSE, and MAPE, the more the predicted values given by the model are. The smaller the value of RMSE, MAE, MSE, and MAPE, the closer the prediction value given by the model is to the true value, and the larger the value of R, the better the fitting ability of the model. The calculation Equations (17)–(21) for the five evaluation indicators are shown below.

$$\mathrm{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{17}$$

$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m}|(y_i - \hat{y}_i)| \tag{18}$$

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{19}$$

$$\mathrm{MAPE} = \frac{100\%}{m}\sum_{i=1}^{m}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{20}$$

$$\mathrm{R} = \frac{\sum_{i=1}^{m}(y_i - \overline{y}_i)(\hat{y}_i - \overline{\hat{y}}_i)}{\sqrt{\sum_{i=1}^{m}(y_i - \overline{y}_i)^2}\sqrt{\sum_{i=1}^{m}(\hat{y}_i - \overline{\hat{y}}_i)^2}} \tag{21}$$

where m is the total number of data sets, $\hat{y}$ denotes the true value of the data, y is the prediction result given by the model, $\overline{\hat{y}}$ is the average of the true value, and $\overline{y}$ denotes the average of the prediction result.

3.3. Prediction in the Greenhouse in Smart Agriculture

To validate the model performance, we compared the RASN Model with other deep learning models such as BP, LSTM, GRU, BiLSTM, and Attention_LSTM, which all use the traditional normalization method, i.e., normalizing the data in the preprocessing stage. The number of network layers for BP, LSTM, GRU, and BiLSTM is set to three, the number of neurons in each layer is 100, 48, and 24, respectively, the batch_size of the model is set to 24, the number of training iterations is 100, the optimizer is ADAM, and the learning rate is 0.001.

There are 9276 sets of data collected in the smart agriculture house with an interval of 30 minutes, divided into 80% for training and 20% for testing. This experiment predicts the 12 hours of temperature, humidity, CO_2, and light intensity, respectively.

The model prediction performance was evaluated using the evaluation metrics proposed in Section 3.2, and the experiment was repeated 10 times for each model independently. The specific performance metrics are shown in Table 2.

Table 2. Comparison of model prediction performance based on temperature data.

MODELS	RMSE	MAE	MSE	MAPE	R
BP [55]	2.2326	1.6305	4.9848	1.0068	0.8358
LSTM [56]	2.3901	1.8321	5.7132	1.0049	0.8034
GRU [57]	2.4916	1.8748	5.7296	1.0034	0.7814
BiLSTM [58]	2.3745	1.8002	5.6022	0.9991	0.8124
BiGRU [59]	2.3603	1.7999	5.5714	0.9873	0.8207
Attention_LSTM [6]	2.2372	1.7033	5.3406	1.0112	0.8341
Attention_GRU [60]	2.1542	1.6613	4.6404	1.0008	0.8426
RASN (our)	1.9032	1.4609	3.4609	0.7001	0.8928

Figure 6 visualizes the differences between the evaluation indexes of the model proposed in this paper and the other models. As can be seen from the Figure, the three evaluation indexes of RMSE, MAE, and MAPE of the proposed model are lower than those of other models, indicating that the difference between the predicted value and the true value given by the model is smaller. At the same time, the R index of the model is larger than that of other models, which indicates that the model proposed in this paper has a better fit. Compared with the best-performing Attention_GRU model in this dataset, the

RMSE of the proposed model is reduced by 11.6%, MAE is reduced by 12.1%, and R is improved by 6%.

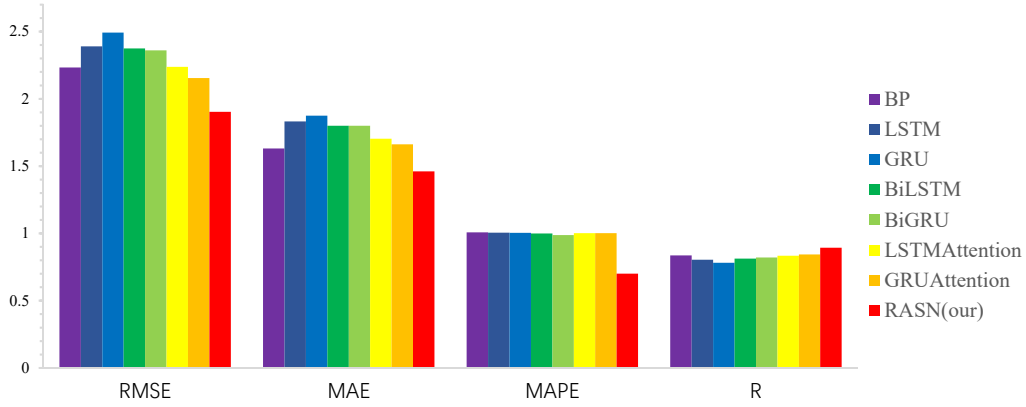

Figure 6. Comparison of experimental evaluation indicators for temperature prediction.

Figure 7 shows the temperature prediction values given by the model used in the comparison test. From the Figure, it can be seen that the prediction results given by the model proposed in this paper have a good fit with the real values and can better reflect the trend of the real temperature data.

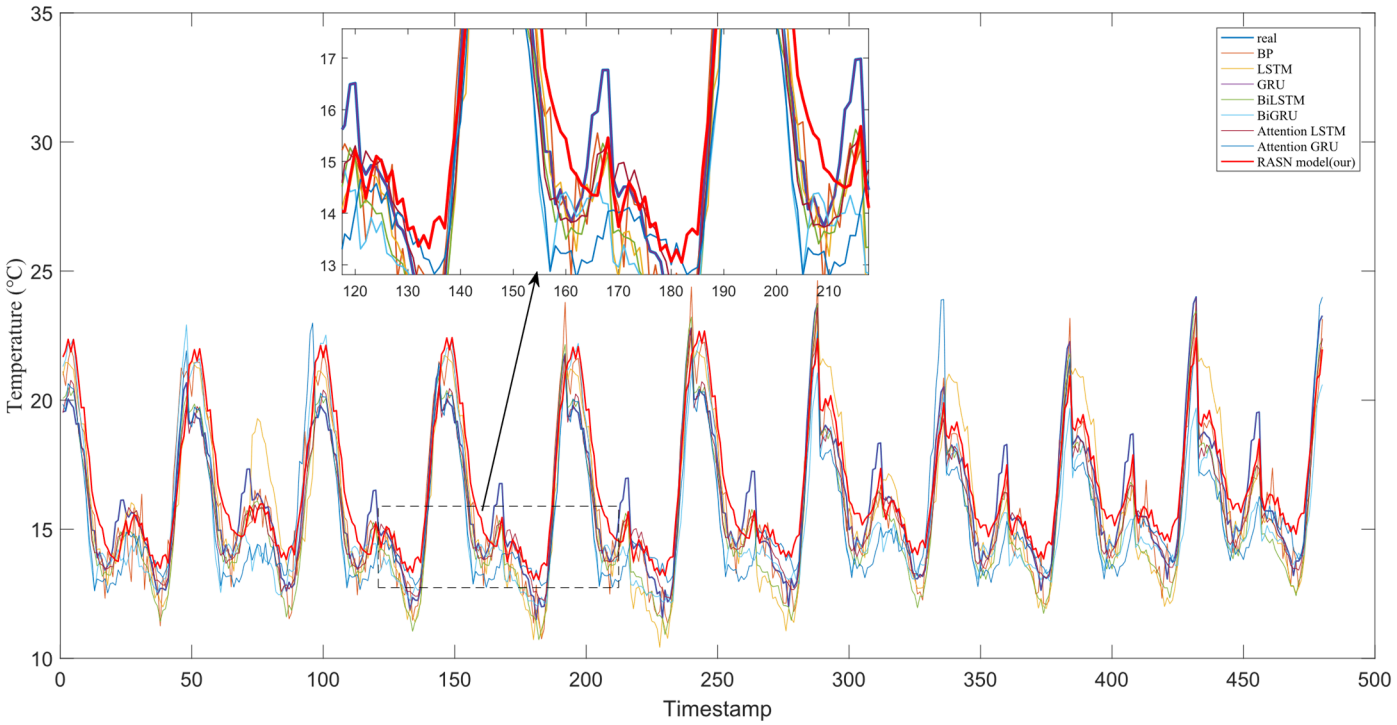

Figure 7. Comparison of model predictions based on temperature data.

Table 3 shows the comparison of evaluation metrics based on humidity data. Figure 8 shows the comparison of evaluation metrics in humidity prediction. The best performing benchmark model in the humidity prediction task is the Attention_GRU model with an RMSE of 2.2025, while the model proposed in this paper has a 4.6% reduction in the RMSE metric, a 12.2% reduction in the MAE metric, and a 5.9% improvement in R on this basis. The humidity prediction curves given by each model are shown in Figure 9. As shown in the Figure, the proposed model in this paper can give the most realistic humidity prediction values.

Table 3. Comparison of model prediction performance based on humidity data.

MODELS	RMSE	MAE	MSE	MAPE	R
BP [55]	2.6413	2.1036	6.7658	1.9965	0.7161
LSTM [56]	2.4681	1.8895	6.2634	1.8759	0.7379
GRU [57]	2.5015	1.9033	6.3631	1.8996	0.7321
BiLSTM [58]	2.3946	1.8112	6.0986	1.8769	0.7568
BiGRU [59]	2.3711	1.8012	6.0124	1.8586	0.7621
Attention_LSTM [6]	2.2172	1.6863	5.8735	1.7121	0.7989
Attention_GRU [60]	2.2025	1.6036	5.8632	1.6876	0.8068
RASN (our)	2.1017	1.4077	5.6007	1.4553	0.8572

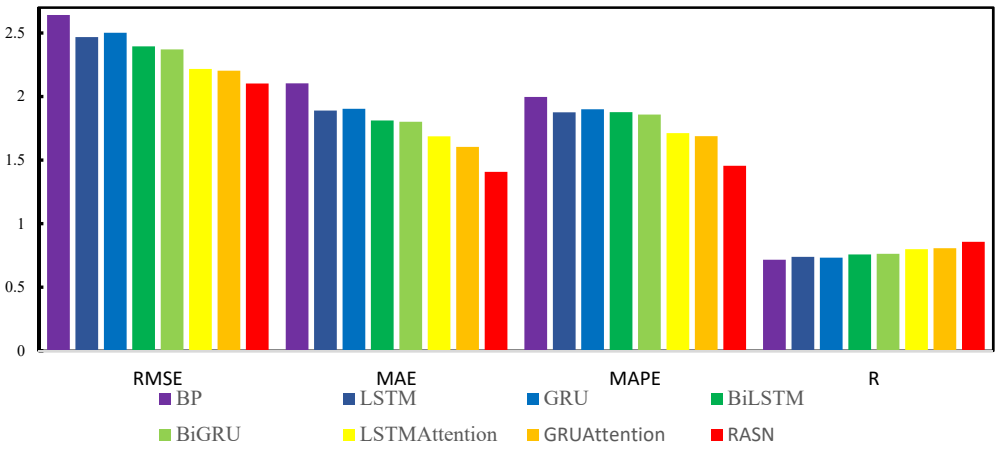

Figure 8. Comparison of experimental evaluation indexes for humidity prediction.

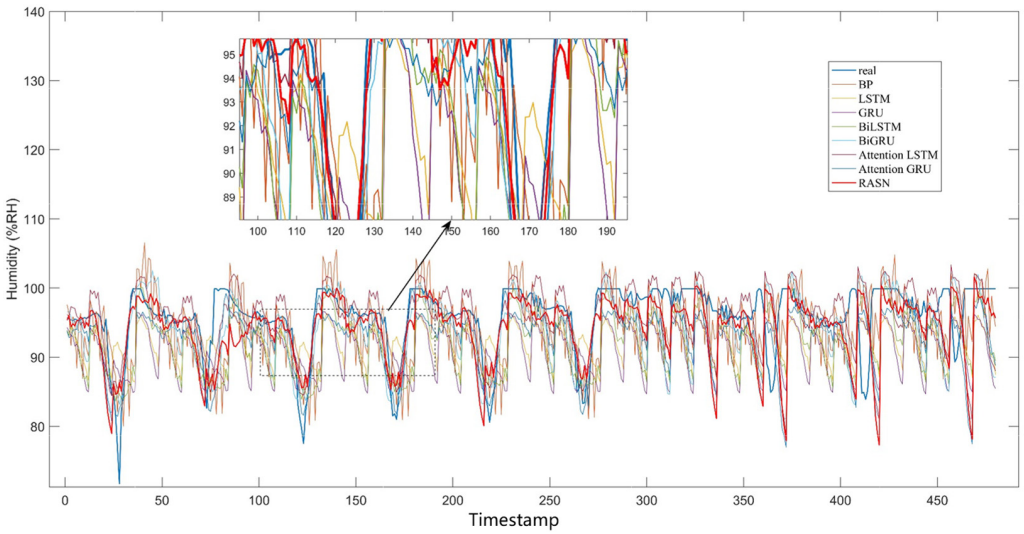

Figure 9. Comparison of model predictions based on humidity data.

The model evaluation metrics based on CO_2 data are shown in Table 4. From the evaluation indexes, it can be seen that the proposed model has the lowest RMSE, MAE, and MAPE indexes and the highest R index, compared with the other seven comparison models. This result shows that the prediction results given by the model are more in line with the real values compared with the other benchmark models, demonstrating the excellent prediction ability of the model for the CO_2 content data in the actual greenhouse. Figure 10 shows the comparison graph of the evaluation indicators.

Table 4. Comparison of model prediction performance based on CO_2 data.

MODELS	RMSE	MAE	MAPE	R
BP [55]	102.2564	79.2658	30.1542	0.3026
LSTM [56]	92.6584	69.2546	22.2165	0.4996
GRU [57]	89.6359	65.2597	20.6541	0.5248
BiLSTM [58]	77.6521	54.9854	15.5896	0.7316
BiGRU [59]	75.3654	53.6218	14.3584	0.7568
Attention_LSTM [6]	69.7441	49.1254	10.9651	0.8368
Attention_GRU [60]	69.7258	48.9856	10.4852	0.8436
RASN (our)	67.7661	47.0659	9.2122	0.8778

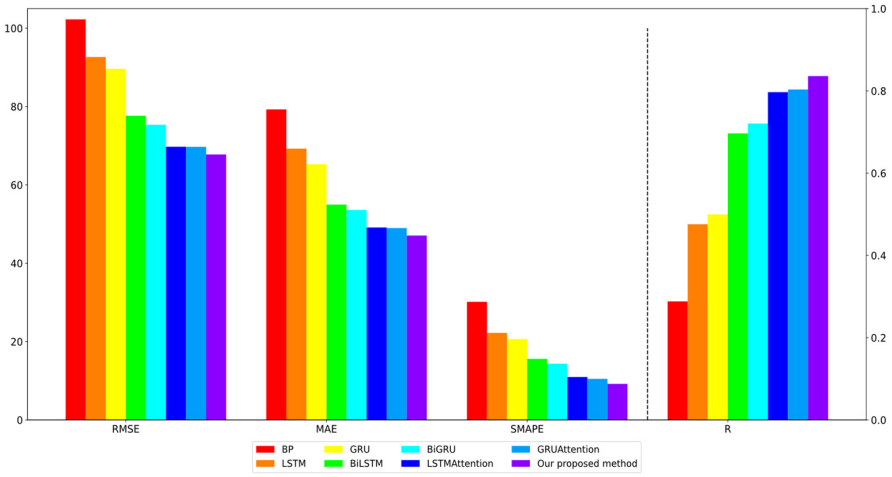

Figure 10. Comparison of evaluation indexes of the model based on CO_2 data.

Light intensity has a strong nonlinearity, and its data is highly volatile, often being zero for 12 consecutive data and then suddenly changing to a larger value. Hence, the prediction of light intensity is a greater test of model performance. Figure 11 shows the evaluation index of the prediction performance based on light intensity. Table 5 compares the evaluation indexes of the model prediction results. By comparing the evaluation indexes of the models, it can be seen that the model proposed in this paper achieves the best results among all the compared models. The prediction of light intensity is difficult, and the other comparative models poorly fit the light intensity. The better model is the Attention_GRU model. Still, the model proposed in this paper improves 15.9% in the R index, demonstrating a better fitting ability for the strong nonlinear data.

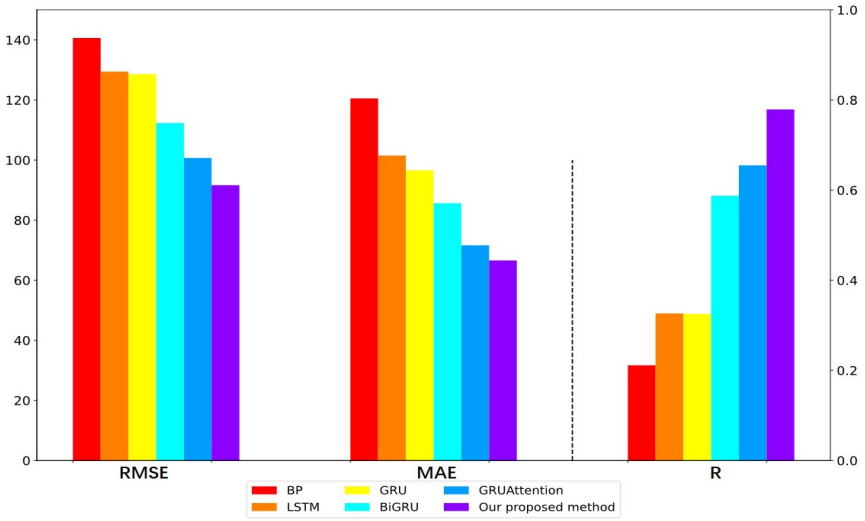

Figure 11. Model performance comparison based on light intensity data.

Table 5. Comparison of model prediction performance based on light intensity data.

MODELS	RMSE	MAE	R
BP [55]	140.6614	120.5418	0.2114
LSTM [56]	129.4581	101.5268	0.3265
GRU [57]	128.6524	96.5924	0.3254
BiGRU [59]	112.3584	85.6521	0.5876
Attention_GRU [60]	100.7216	71.6594	0.6551
RASN (our)	91.6691	66.6154	0.7791

In order to consider the influence of more complex greenhouse external conditions and verify the predictive ability of the model for multi-input variables, three variables of external greenhouse temperature, external wind speed, and internal greenhouse temperature are used as inputs to verify the performance of the model in outputting 24-step and 36-step indoor temperature prediction. The results of the evaluation indicators are shown in Table 6.

Table 6. Prediction performance of the model with multiple input variables.

STEP	RMSE	MAE	MSE	MAPE	R
36-step	2.2364	1.5242	5.0014	1.6478	0.8358
24-step	2.0289	1.4467	4.1165	0.5279	0.8673

Figure 12 shows the model's prediction performance with different step sizes under multivariable input. From the verification results of multiple input variables, the proposed model can still maintain good prediction accuracy under external complex input. Still, the long output step length will affect the prediction effect of the model. When the model considers the long prediction output step, the prediction accuracy decreases, while the prediction accuracy is higher in the short term.

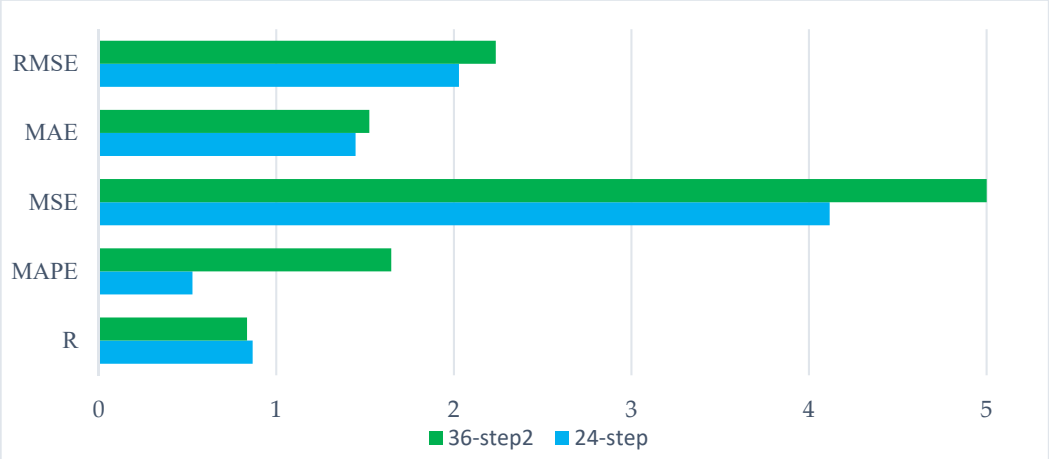

Figure 12. Comparison of prediction results with different step sizes in multivariable input.

The model proposed in this paper can fit temperature and humidity, CO_2, and light intensity data in greenhouses and output more accurate future prediction values. The accurate prediction can provide data support for greenhouse management, significantly improve the intelligent management of greenhouses, and help the development of smart agriculture.

4. Conclusions

To solve the problems of switching normalization methods and poor adaptability to predicting in the greenhouse for a smart agriculture system, we proposed a deep prediction model that can achieve automatic normalization in this paper.

The model includes an adaptive normalization layer, a GRU-based deep network model, an adaptive inverse normalization layer, and a normalization method selection module. The data are normalized by each of the four normalization methods in the normalization layer, and the normalized data are scaled and panned by learnable parameters. Second, the normalized data are fed into the GRU deep learning network to derive the prediction results. Again, the four normalized prediction results are transformed into actual data by the corresponding inverse normalization methods in the inverse normalization layer, and the learnable parameters optimize this data. The optimized actual data is output to the model. Finally, the normalization selection module evaluates the prediction data obtained by different normalization methods to obtain the prediction result that best matches the true value.

The model performance is validated using the data sets sensing from the practical greenhouse and compared with other benchmark models. The experimental results show that the proposed method can better adapt to nonstationary data and better predict greenhouse meteorological data. Good prediction accuracy can play a positive role in promoting greenhouse control and the development of smart agriculture. The proposed prediction approaches of time series models in the paper can combine other parameter estimation algorithms [61–68] with studying the parameter identification problems of linear and nonlinear systems with different disturbances [69–77] for building the soft sensor models and prediction models based on the time series data. It can be applied to other fields [78–81], such as signal processing, engineering application systems [82–93], and others.

In future work, we will further validate the portability of our proposed RASN model. At the same time, we will explore more normalization methods to comprehensively consider and verify the influence of various normalization methods on the model effect.

Author Contributions: Conceptualization, X.J. and J.Z.; methodology, X.J. and J.Z.; software, J.Z.; validation, X.J. and J.Z.; formal analysis, J.K.; resources, X.J. and J.K.; data curation, J.Z., Y.B. and T.S.; writing—original draft preparation, X.J. and J.Z.; writing—review and editing, X.J. and J.Z.; visualization, J.Z.; supervision, J.K., Y.B. and T.S.; funding acquisition, X.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (No. 62006008, 62173007, 61903009), the National Key Research and Development Program of China (No. 2021YFD2100605).

Data Availability Statement: Not applicable.

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

References

1. Zheng, Y.Y.; Kong, J.L.; Jin, X.B.; Wang, X.Y.; Zuo, M. Crop Deep: The crop vision dataset for deep-learning-based classification and detection in precision agriculture. *Sensors* **2019**, *19*, 1058. [CrossRef] [PubMed]
2. Zheng, Y.Y.; Kong, J.L.; Jin, X.B.; Wang, X.Y.; Su, T.L.; Wang, J.L. Probability fusion decision framework of multiple deep neural networks for fine-grained visual classification. *IEEE Access* **2019**, *7*, 122740–122757. [CrossRef]
3. Idrees, S.M.; Alam, M.A.; Agarwal, P. A prediction approach for stock market volatility based on time series data. *IEEE Access* **2019**, *7*, 7287–17298. [CrossRef]
4. Jin, X.B.; Yu, X.H.; Su, T.L.; Yang, D.N.; Bai, Y.T.; Kong, J.L.; Wang, L. Distributed deep fusion predictor for a multi-sensor system based on causality entropy. *Entropy* **2021**, *23*, 219. [CrossRef]
5. Faruk, S.; Yigit, A.; Adnan, K. Hybrid time series forecasting methods for travel time prediction. *Phys. A* **2021**, *579*, 126134.
6. Jin, X.B.; Zheng, W.Z.; Kong, J.L.; Wang, X.Y.; Bai, Y.T.; Su, T.L.; Lin, S. Deep-learning forecasting method for electric power load via attention-based encoder-decoder with bayesian optimization. *Energies* **2021**, *14*, 1596. [CrossRef]
7. Jin, X.B.; Wang, H.X.; Wang, X.Y.; Bai, Y.T.; Su, T.L.; Kong, J.L. Deep-Learning prediction model with serial two-level decomposition based on bayesian optimization. *Complexity* **2020**, *2020*, 14. [CrossRef]
8. Ding, F.; Chen, T. Combined parameter and output estimation of dual-rate systems using an auxiliary model. *Automatica* **2004**, *40*, 1739–1748. [CrossRef]
9. Ding, F.; Chen, T. Parameter estimation of dual-rate stochastic systems by using an output error method. *IEEE Trans. Autom. Control* **2005**, *50*, 1436–1441. [CrossRef]

10. Ding, F.; Shi, Y.; Chen, T. Auxiliary model-based least-squares identification methods for Hammerstein output-error systems. *Syst. Control Lett.* **2007**, *56*, 373–380. [CrossRef]
11. Zhou, Y. Modeling nonlinear processes using the radial basis function-based state-dependent autoregressive models. *IEEE Signal Processing Lett.* **2020**, *27*, 1600–1604. [CrossRef]
12. Zhou, Y.H.; Zhang, X. Partially-coupled nonlinear parameter optimization algorithm for a class of multivariate hybrid models. *Appl. Math. Comput.* **2022**, *414*, 126663. [CrossRef]
13. Zhou, Y.H.; Zhang, X. Hierarchical estimation approach for RBF-AR models with regression weights based on the increasing data length. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 3597–3601. [CrossRef]
14. Ding, F.; Zhang, X.; Xu, L. The innovation algorithms for multivariable state-space models. *Int. J. Adapt. Control Signal Processing* **2019**, *33*, 1601–1608. [CrossRef]
15. Ding, F.; Liu, G.; Liu, X.P. Parameter estimation with scarce measurements. *Automatica* **2011**, *47*, 1646–1655. [CrossRef]
16. Liu, Y.J.; Shi, Y. An efficient hierarchical identification method for general dual-rate sampled-data systems. *Automatica* **2014**, *50*, 962–970. [CrossRef]
17. Zhang, X. Optimal adaptive filtering algorithm by using the fractional-order derivative. *IEEE Signal Processing Lett.* **2022**, *29*, 399–403. [CrossRef]
18. Li, M.H.; Liu, X.M. The filtering-based maximum likelihood iterative estimation algorithms for a special class of nonlinear systems with autoregressive moving average noise using the hierarchical identification principle. *Int. J. Adapt. Control Signal Processing* **2019**, *33*, 1189–1211. [CrossRef]
19. Ding, J.; Liu, X.P.; Liu, G. Hierarchical least squares identification for linear SISO systems with dual-rate sampled-data. *IEEE Trans. Autom. Control* **2011**, *56*, 2677–2683. [CrossRef]
20. Ding, F.; Liu, Y.J.; Bao, B. Gradient-based and least squares based iterative estimation algorithms for multi-input multi-output systems. *Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng.* **2012**, *226*, 43–55. [CrossRef]
21. Xu, L.; Chen, F.Y.; Hayat, T. Hierarchical recursive signal modeling for multi-frequency signals based on discrete measured data. *Int. J. Adapt. Control Signal Processing* **2021**, *35*, 676–693. [CrossRef]
22. Wang, Y.J. Novel data filtering based parameter identification for multiple-input multiple-output systems using the auxiliary model. *Automatica* **2016**, *71*, 308–313. [CrossRef]
23. Jin, X.B.; Yu, X.H.; Wang, X.Y.; Bai, Y.T.; Su, T.L.; Kong, J.L. Deep learning predictor for sustainable precision agriculture based on internet of things system. *Sustainability* **2020**, *12*, 1433. [CrossRef]
24. Jin, X.B.; Yang, N.X.; Wang, X.Y.; Bai, Y.T.; Su, T.L.; Kong, J.L. Deep hybrid model based on EMD with classification by frequency characteristics for long-term air quality prediction. *Mathematics* **2020**, *8*, 214. [CrossRef]
25. Zhen, T.; Kong, J.; Yan, L. Hybrid Deep-Learning framework based on gaussian fusion of multiple spatiotemporal networks for walking gait phase recognition. *Complexity* **2020**, *2020*, 1–17. [CrossRef]
26. Jin, X.B.; Zheng, W.Z.; Kong, J.L.; Wang, X.Y.; Zuo, M.; Zhang, Q.C.; Lin, S. Deep-Learning temporal predictor via bi-directional self-attentive Encoder-Decoder framework for IoT-based environmental sensing in intelligent greenhouse. *Agriculture* **2021**, *11*, 802. [CrossRef]
27. Jin, X.B.; Yang, N.X.; Wang, X.Y.; Bai, Y.T.; Su, T.L.; Kong, J.L. Integrated predictor based on decomposition mechanism for PM2.5 long-term prediction. *Appl. Sci.* **2019**, *9*, 4533. [CrossRef]
28. Kim, H.J.; Baek, J.W.; Chung, K. Associative knowledge graph using fuzzy clustering and Min-Max normalization in video contents. *IEEE Access* **2021**, *9*, 74802–74816. [CrossRef]
29. Jain, A.; Nandakumar, K.; Ross, A. Score normalization in multimodal biometric systems. *Pattern Recognit.* **2013**, *38*, 2270–2285. [CrossRef]
30. Summers, C.; Dinneen, M.J. Four things everyone should know to improve Batch Normalization. *ICLR* **2020**, 1–18.
31. Pan, H.; Niu, X.; Li, R.; Dou, Y.; Jiang, H. Annealed gradient descent for Deep Learning. *Neurocomputing* **2019**, *380*, 201–211. [CrossRef]
32. Zou, Z.X.; Yang, Y.M.; Fan, Z.Q.; Tang, H.M.; Zou, M.; Hu, X.L.; Xiong, C.R.; Ma, J.W. Suitability of data preprocessing methods for landslide displacement forecasting. *Stoch. Environ. Res. Risk Assess.* **2020**, *34*, 1105–1119. [CrossRef]
33. Singh, D.; Singh, B. Feature wise normalization: An effective way of normalizing data. *Pattern Recognit.* **2022**, *122*, 108307. [CrossRef]
34. Jain, S.; Shukla, S.; Wadhvani, R. Dynamic selection of normalization techniques using data complexity measures. *Expert Syst. Appl.* **2018**, *106*, 252–262. [CrossRef]
35. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv* **2015**, arXiv:1502.03167.
36. Passalis, N.; Tefas, A.; Kanniainen, J.; Gabbouj, M.; Iosifidis, A. Deep adaptive input normalization for time series forecasting. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *99*, 1–6. [CrossRef] [PubMed]
37. Tomar, D.; Lortkipanidze, M.; Vray, G.; Bozorgtabar, B.; Thiran, J.P. Self-Attentive spatial adaptive normalization for cross-modality domain adaptation. *IEEE Trans. Med. Imaging* **2021**, *40*, 2926–2938. [CrossRef] [PubMed]
38. Park, T.; Liu, M.Y.; Wang, T.C.; Zhu, J.Y. Semantic image synthesis with spatially-adaptive normalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 1–10.

39. Yan, J.J.; Wan, R.S.; Zhang, X.Y.; Zhang, W.; Wei, Y.C.; Sun, J. Towards stabilizing batch statistics in backward propagation of Batch Normalization. *arXiv* **2020**, arXiv:2001.06838.
40. Du, Y.J.; Zhen, X.T.; Shao, L.; Snoek, C.G.M. MetaNorm: Learning to normalize few-shot batches across domains. In Proceedings of the International Conference on Learning Representations, online, 3–7 May 2021; pp. 1–23.
41. Luo, P.; Ren, J.M.; Peng, Z.L.; Zhang, R.M.; Li, J.Y. Differentiable learning-to-normalize via switchable normalization. *arXiv* **2019**, arXiv:1806.10779.
42. Shao, W.Q.; Meng, T.J.; Li, J.Y.; Zhang, R.M.; Wang, X.G.; Luo, P. SSN: Learning sparse switchable normalization via SparsestMax. *Int. J. Comput. Vis.* **2020**, *128*, 2107–2125. [CrossRef]
43. Yang, S.; Yu, S.; Zhao, B.; Wang, Y. Reducing the feature divergence of RGB and near-infrared images using Switchable Normalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Seattle, WA, USA, 13–19 June 2020; pp. 206–211.
44. Giraldo, L.G.S.; Schwartz, O. Integrating flexible normalization into midlevel representations of deep convolutional neural Networks. *Neural Comput.* **2019**, *31*, 2138–2176. [CrossRef] [PubMed]
45. Bermudez-Edo, M.; Barnaghi, P.; Moessner, K. Analysing real world data streams with spatio-temporal correlations: Entropy vs. Pearson correlation. *Autom. Constr.* **2018**, *88*, 87–100. [CrossRef]
46. Wang, W.; Li, W. Research on trend analysis method of multi-series economic data based on correlation enhancement of deep learning. *Neural Comput. Appl.* **2021**, *33*, 4815–4831. [CrossRef]
47. Khodabandelou, G.; Jung, P.G.; Amirat, Y.; Mohammed, S. Attention-Based gated recurrent unit for gesture recognition. *IEEE Trans. Autom.* **2020**, *18*, 495–507. [CrossRef]
48. Tian, L.; Li, X.; Ye, Y.; Xie, P.; Li, Y. A generative adversarial gated recurrent unit model for precipitation nowcasting. *IEEE Geosci. Remote Sens. Lett.* **2020**, *17*, 601–605. [CrossRef]
49. Shen, X.; Tian, X.; Liu, T.; Xu, F.; Tao, D.C. Continuous dropout. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 3926–3937. [CrossRef] [PubMed]
50. Yu, J.; Kim, S.B.; Bai, J.; Han, S.W. Comparative study on exponentially weighted moving average approaches for the self-starting forecasting. *Appl. Sci.* **2020**, *10*, 7351. [CrossRef]
51. Kong, J.L.; Wang, H.X.; Jin, X.B.; Wang, X.Y.; Fang, X.; Lin, S. Multi-stream hybrid architecture based on cross-level fusion strategy for fine-grained crop species recognition in precision agriculture. *Comput. Electron. Agric.* **2021**, *185*, 106134. [CrossRef]
52. Idoje, G.; Dagiuklas, T.; Iqbal, M. Survey for smart farming technologies: Challenges and issues. *Comput. Electr. Eng.* **2021**, *92*, 107104. [CrossRef]
53. Kong, J.L.; Yang, C.C.; Wang, J.L.; Zuo, M.; Jin, X.B.; Lin, S. Deep-stacking network approach by multisource data mining for hazardous risk identification in IoT-based intelligent food management systems. *Comput. Intell. Neurosci.* **2021**, *2021*, 1194565. [CrossRef] [PubMed]
54. Chakrabarty, A.; Mansoor, N.; Uddin, M.I.; Al-adaileh, M.H.; Alsharif, N.; Alsaade, F.W. Prediction approaches for Smart cultivation: A comparative study. *Complexity* **2021**, *2021*, 1–16. [CrossRef]
55. Yang, S.T.; Luo, L.N.; Tan, B.H. Research on sports performance prediction based on BP neural network. *Mob. Inf. Syst.* **2021**, *2021*, 8. [CrossRef]
56. Van Houdt, G.; Mosquera, C.; Nápoles, G. A review on the long short-term memory model. *Artif. Intell. Rev.* **2020**, *53*, 5929–5955. [CrossRef]
57. Jin, X.B.; Gong, W.T.; Kong, J.L.; Bai, Y.T.; Su, T.L. Variational Bayesian deep network with data self-screening layer for massive time-series data forecasting. *Entropy* **2022**, *24*, 335. [CrossRef]
58. Petroanu, D.M.; Prjan, A. Electricity consumption forecasting based on a bidirectional long-short-term memory artificial neural network. *Sustainability* **2020**, *13*, 104. [CrossRef]
59. Meng, F.; Song, T.; Xu, D.Y.; Xie, P.F.; Li, Y. Forecasting tropical cyclones wave height using bidirectional gated recurrent unit. *Ocean Eng.* **2021**, *234*, 108795. [CrossRef]
60. Niu, Z.; Yu, Z.; Tang, W.; Wu, Q.; Reformat, M. Wind power forecasting using attention-based gated recurrent unit network. *Energy* **2020**, *196*, 117081. [CrossRef]
61. Ding, F.; Lv, L.; Pan, J.; Wang, X.K.; Jin, X.B. Two-stage gradient-based iterative estimation methods for controlled autoregressive systems using the measurement data. *Int. J. Control Autom. Syst.* **2020**, *18*, 886–896. [CrossRef]
62. Ding, F.; Wang, F.F.; Wu, M.H. Decomposition based least squares iterative identification algorithm for multivariate pseudo-linear ARMA systems using the data filtering. *J. Frankl. Inst.* **2017**, *354*, 1321–1339. [CrossRef]
63. Zhang, X. Hierarchical parameter and state estimation for bilinear systems. *Int. J. Syst. Sci.* **2020**, *51*, 275–290. [CrossRef]
64. Xu, L.; Zhu, Q.M. Decomposition strategy-based hierarchical least mean square algorithm for control systems from the impulse responses. *Int. J. Syst. Sci.* **2021**, *52*, 1806–1821. [CrossRef]
65. Zhang, X.; Xu, L.; Hayat, T. Combined state and parameter estimation for a bilinear state space system with moving average noise. *J. Frankl. Inst.* **2018**, *355*, 3079–3103. [CrossRef]
66. Pan, J.; Jiang, X.; Ding, W. A filtering based multi-innovation extended stochastic gradient algorithm for multivariable control systems. *Int. J. Control Autom. Syst.* **2017**, *15*, 1189–1197. [CrossRef]
67. Zhang, X.; Yang, E.F. State estimation for bilinear systems through minimizing the covariance matrix of the state estimation errors. *Int. J. Adapt Control Signal Processing* **2019**, *33*, 1157–1173. [CrossRef]

68. Pan, J.; Ma, H.; Liu, Q.Y. Recursive coupled projection algorithms for multivariable output-error-like systems with coloured noises. *IET Signal Processing* **2020**, *14*, 455–466. [CrossRef]
69. Ding, F.; Liu, G.; Liu, X.P. Partially coupled stochastic gradient identification methods for non-uniformly sampled systems. *IEEE Trans. Autom. Control* **2010**, *55*, 1976–1981. [CrossRef]
70. Ding, F.; Shi, Y.; Chen, T. Performance analysis of estimation algorithms of non-stationary ARMA processes. *IEEE Trans. Signal Processing* **2006**, *54*, 1041–1053. [CrossRef]
71. Wang, Y.J.; Wu, M.H. Recursive parameter estimation algorithm for multivariate output-error systems. *J. Frankl. Inst.* **2018**, *355*, 5163–5181. [CrossRef]
72. Xu, L. Separable multi-innovation Newton iterative modeling algorithm for multi-frequency signals based on the sliding measurement window. *Circuits Syst. Signal Processing* **2022**, *41*, 805–830. [CrossRef]
73. Xu, L. Separable Newton recursive estimation method through system responses based on dynamically discrete measurements with increasing data length. *Int. J. Control Autom. Syst.* **2022**, *20*, 432–443. [CrossRef]
74. Xu, L.; Yang, E.F. Auxiliary model multiinnovation stochastic gradient parameter estimation methods for nonlinear sandwich systems. *Int. J. Robust Nonlinear Control* **2021**, *31*, 148–165. [CrossRef]
75. Zhang, X. Adaptive parameter estimation for a general dynamical system with unknown states. *Int. J. Robust Nonlinear Control* **2020**, *30*, 1351–1372. [CrossRef]
76. Zhang, X. Recursive parameter estimation methods and convergence analysis for a special class of nonlinear systems. *Int. J. Robust Nonlinear Control* **2020**, *30*, 1373–1393. [CrossRef]
77. Zhang, X. Recursive parameter estimation and its convergence for bilinear systems. *IET Control Theory Appl.* **2020**, *14*, 677–688. [CrossRef]
78. Liu, S.Y.; Hayat, T. Hierarchical principle-based iterative parameter estimation algorithm for dual-frequency signals. *Circuits Syst. Signal Processing* **2019**, *38*, 3251–3268. [CrossRef]
79. Wan, L.J. Decomposition- and gradient-based iterative identification algorithms for multivariable systems using the multi-innovation theory. *Circuits Syst. Signal Processing* **2019**, *38*, 2971–2991. [CrossRef]
80. Jin, X.B.; Yang, N.X.; Wang, X.Y.; Bai, Y.T. Hybrid deep learning predictor for smart agriculture sensing based on empirical mode decomposition and gated recurrent unit group model. *Sensors* **2020**, *20*, 1334. [CrossRef]
81. Wang, X.H. Modified particle filtering-based robust estimation for a networked control system corrupted by impulsive noise. *Int. J. Robust Nonlinear Control* **2022**, *32*, 830–850. [CrossRef]
82. Pan, J.; Li, W.; Zhang, H.P. Control algorithms of magnetic suspension systems based on the improved double exponential reaching law of sliding mode control. *Int. J. Control Autom. Syst.* **2018**, *16*, 2878–2887. [CrossRef]
83. Ma, H.; Pan, J.; Ding, W. Partially-coupled least squares based iterative parameter estimation for multi-variable output-error-like autoregressive moving average systems. *IET Control Theory Appl.* **2019**, *13*, 3040–3051. [CrossRef]
84. Ding, F.; Liu, X.P.; Yang, H.Z. Parameter identification and intersample output estimation for dual-rate systems. *IEEE Trans. Syst Man Cybern. Part A Syst. Hum.* **2008**, *38*, 966–975. [CrossRef]
85. Ding, F.; Liu, X.P.; Liu, G. Multiinnovation least squares identification for linear and pseudo-linear regression models. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2010**, *40*, 767–778. [CrossRef]
86. Xu, L.; Song, G.L. A recursive parameter estimation algorithm for modeling signals with multi-frequencies. *Circuits Syst. Signal Processing* **2020**, *39*, 4198–4224. [CrossRef]
87. Jin, X.B.; Gong, W.T.; Kong, J.L.; Bai, Y.T.; Su, T.L. PFVAE: A planar flow-based variational auto-encoder prediction model for time-series data. *Mathematics* **2022**, *10*, 610. [CrossRef]
88. Xu, L.; Sheng, J. Separable multi-innovation stochastic gradient estimation algorithm for the nonlinear dynamic responses of systems. *Int. J. Adapt. Control Signal Processing* **2020**, *34*, 937–954. [CrossRef]
89. Hou, J.; Chen, F.W.; Li, P.H.; Zhu, Z.Q. Gray-box parsimonious subspace identification of Hammerstein-type systems. *IEEE Trans. Ind. Electron.* **2021**, *68*, 9941–9951. [CrossRef]
90. Zhao, Z.Y.; Zhou, Y.Q.; Wang, X.Y.; Wang, Z.Y.; Bai, Y.T. Water quality evolution mechanism modeling and health risk assessment based on stochastic hybrid dynamic systems. *Expert Syst. Appl.* **2022**, *193*, 116404. [CrossRef]
91. Chen, Q.; Zhao, Z.Y.; Wang, X.Y.; Xiong, K. Microbiological predictive modeling and risk analysis based on the one-step kinetic integrated Wiener process. *Innovat. Food Sci. Emerg. Technol.* **2022**, *75*, 102912. [CrossRef]
92. Shu, J.; He, J.C.; Li, L. MSIS: Multispectral instance segmentation method for power equipment. *Comput. Intell. Neurosci.* **2022**, *2022*, 2864717. [CrossRef]
93. Peng, H.X.; He, W.J.; Zhang, Y.L.; Li, X.W.; Ding, Y.; Menon, V.G.; Verma, S. Covert non-orthogonal multiple access communication assisted by multi-antenna jamming. *Phys. Comm.* **2022**, *2022*, 101598. [CrossRef]