Research in Astronomy and Astrophysics, 24:065024 (13pp), 2024 June https://doi.org/10.1088/1674-4527/ad4963
© 2024. National Astronomical Observatories, CAS and IOP Publishing Ltd. Printed in China and the U.K.

The Adjustment Analysis Method of the Active Surface Antenna Based on Convolutional Neural Network

You Ban[1,2], Shang Shi[1], Na Wang[2], Qian Xu[2], and Shufei Feng[3]
[1] School of Mechanical Engineering, Xinjiang University, Urumqi 830017, China; banyou_xd@163.com
[2] Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi 830011, China; na.wang@xao.ac.cn
[3] School of Mechanical Engineering, Dongguan University of Technology, Dongguan 523808, China
Received 2024 February 26; revised 2024 April 27; accepted 2024 May 8; published 2024 June 10

Abstract

Active surface technique is one of the key technologies to ensure the reflector accuracy of the millimeter/submillimeter wave large reflector antenna. The antenna is complex, large-scale, and high-precision equipment, and its active surfaces are affected by various factors that are difficult to comprehensively deal with. In this paper, based on the advantage of the deep learning method that can be improved through data learning, we propose the active adjustment value analysis method of large reflector antenna based on deep learning. This method constructs a neural network model for antenna active adjustment analysis in view of the fact that a large reflector antenna consists of multiple panels spliced together. Based on the constraint that a single actuator has to support multiple panels (usually 4), an autonomously learned neural network emphasis layer module is designed to enhance the adaptability of the active adjustment neural network model. The classical 8-meter antenna is used as a case study, the actuators have a mean adjustment error of 0.00252 mm, and the corresponding antenna surface error is 0.00523 mm. This active adjustment result shows the effectiveness of the method in this paper.

Key words: techniques: radar astronomy – telescopes – methods: analytical – methods: numerical

1. Introduction

Large reflector antennas are important observation equipment in the field of deep space exploration and radio astronomy. With the increasing demand for high-frequency observation, large reflector antennas are constantly developing in the direction of large aperture, high frequency band, and high precision (Baars & Kärcher 2018). The surface accuracy of large reflector antenna directly affects the electrical performance of the antenna, but its reflector is usually made of a large number of panels spliced together, which will inevitably produce deformation under the action of gravity, temperature and wind and other external loads, which will directly lead to the antenna to appear gain degradation, pointing error and other problems (Ban et al. 2020). Therefore, ensuring the accuracy of the antenna reflector surface is a key link to guarantee that the antenna realizes the desired performance.

At present, the active surface technique is the most effective and direct method to realize the surface accuracy of large full panel high-frequency reflector antennas. It has been widely used, for example, the 100 m Green Bank Telescope (GBT) in the U.S. (White et al. 2022), the 64 m Sardinia Radio Telescope (SRT) in Italy (Bolli et al. 2015), the 50-meter Large Millimeter Telescope (LMT) in Mexico (Hughes et al. 2010), and the 65 m Tianma Radio Telescope (TMT) in Shanghai (Dong et al. 2016), etc. Currently, the world's largest full panel and steerable large reflector antennas, QTT (Xu & Wang 2016; Wang et al. 2023), is being built in Qitai, Xinjiang, which has a 110 m aperture of the main reflector with 2242 panels and can operate at 115 GHz, and will also use the active surface technique to ensure the required surface accuracy. The active surface technique mainly realizes the regulation of antenna surface accuracy through the actuator supporting the panel (which adjusts the position of the panel by extending or shortening). Among them, calculating the adjustment value of the actuator is one of the key elements of the active surface technique to realize the antenna reflector with high accuracy requirements.

Currently, many scholars have studied the calculation method of the actuator adjustment value, and the main idea is to take the normal distance between the ideal reflector node and the actual reflector node as the adjustment value. Usually, 2-parameter, 5-parameter and 6-parameter methods can be used to determine the optimal coincident reflector (Fu et al. 2015), and some scholars have used microwave holography to measure the antenna surface error to establish the calculation method of the actuator adjustment value (Ban et al. 2024b). These methods are simple and effective, and the resulting adjustment value improves the accuracy of the antenna surface to some extent. However, these adjustment methods ignore the effect of the elastic deformation of the panel itself when the actuator adjusts the panel position. Therefore, some scholars have approximated the linear relationship matrix between the

elastic deformation of the panel and the adjustment value of the actuator (Lian et al. 2021), and the adjustment value of the antenna actuator is calculated by this optimization method. Ban et al. directly simplified the actuator mechanism and then directly established the structural model of the active reflector antenna using the simplified actuator (Ban et al. 2023, 2024a), which further improved the method of the analysis of the active surface and the calculation of the adjustment value. These methods center around the structural error for the calculation of the actuator adjustment value, and do not take into account the influence of the change of the adjustment amount brought about by different complex environments, the aging of the regulation equipment during the actual operation and maintenance process of the project. For this reason, this paper proposes an extensible method for analyzing the adjustment value, i.e., the neural network-based adjustment analysis method, which can be intelligently learned to improve the adjustment amount through the adjustment data in the future engineering practice.

In recent years, with the rapid development of computer hardware, artificial intelligence, especially deep learning technology, has gained great progress in the application field. It has been widely applied in the fields of materials (Fang et al. 2022), physics (Thuerey et al. 2021), chemistry (Goh et al. 2017), medicine (Zhou et al. 2021) and other fields with remarkable results. Its intelligent autonomous learning through data also opens up new ideas for the calculation method of the active adjustment value of large reflector antennas, but there is still no research on deep learning applied to antenna active surface technique. Therefore, this paper for large reflector antenna active surface technique urgently needs intelligent learning to improve the demand for adjustment value calculation method, innovatively put forward the active surface adjustment value calculation method based on neural network, In order to train and fine-tune the parameters of the established active adjustment neural network model according to the use data of the active surface technique in the process of engineering application, so that the active adjustment value calculation method has the intelligent property of self-learning. Due to the large number of large reflector antenna panels and actuators, and for such a large number of feature input-output problems, the neural network using only a multilayer perceptron will produce training difficulties and slow arithmetic (Zhou et al. 2017); on the other hand, the reflector panel nodes contain prior knowledge such as structural and contextual information, which can help the neural network to understand and analyze the data better, and help to improve the convergence accuracy. Combining the above characteristics, there are two major types of neural network models available at present, namely Convolutional Neural Networks (CNNs) (Hadji & Wildes 2018) and Vision Transformer (Liu et al. 2023). After the measurement of the antenna active adjustment neural network model established in this paper, under the same conditions, the CNNs with the same accuracy level are obtained with higher efficiency of the active adjustment mode, so this paper finally adopts CNNs to establish the antenna active surface adjustment deep learning model.

The overall flow of this paper is shown in Figure 1. The convolutional neural network model for antenna active adjustment is constructed in Section 2. The data preprocessing method for the model is described in Section 3.1. The initial parameter settings, loss function definitions, and the model training strategy of the constructed model are also described in Section 3.2. The classical 8-meter antenna is used as a case in Section 4, and the built active tuning deep learning model is trained to obtain feasible and effective application results. Finally, all the conclusions are described in Section 5.

2. Modeling of Convolutional Neural Network for Antenna Active Adjustment

Since the reflector panel nodes have a priori knowledge such as structural information and contextual information, in order to use this a priori knowledge to improve the convergence accuracy, and at the same time for the large reflector antenna panels and actuators are more in number, i.e., the neural network will be faced with a larger number of inputs and outputs, the convolutional neural networks (CNNs) are more efficient compared to other neural network structures, and therefore, this paper finally adopted CNNs to establish the antenna active adjustment model.

As shown in Figure 2, the input features of this neural network are the coordinate information of the nodes of the reflector panels, and the output is the actuator adjustment value corresponding to the input panels position. In order for the neural network to learn the changing features of the panels at different levels, i.e., to perceive the information of the panel surface at different scales, and to help the neural network to learn the regularity of the data set, multi-scale convolution is required. For example, in the first layer of the model in Figure 2, different sizes of convolution kernels are used for convolution respectively. In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output (N, C_{out}, H_{out}, W_{out}) can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \otimes \text{input}(N_i, k) \quad (1)$$

where $\otimes$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

In order to save arithmetic power at the same time to ensure that the neural network has a large range of perceptual ability, combined with the use of large size convolutional kernel and a larger step size, such as the first layer of Figure 2, the use of 5×5 size convolutional kernel and 7×7 size convolutional kernel, and

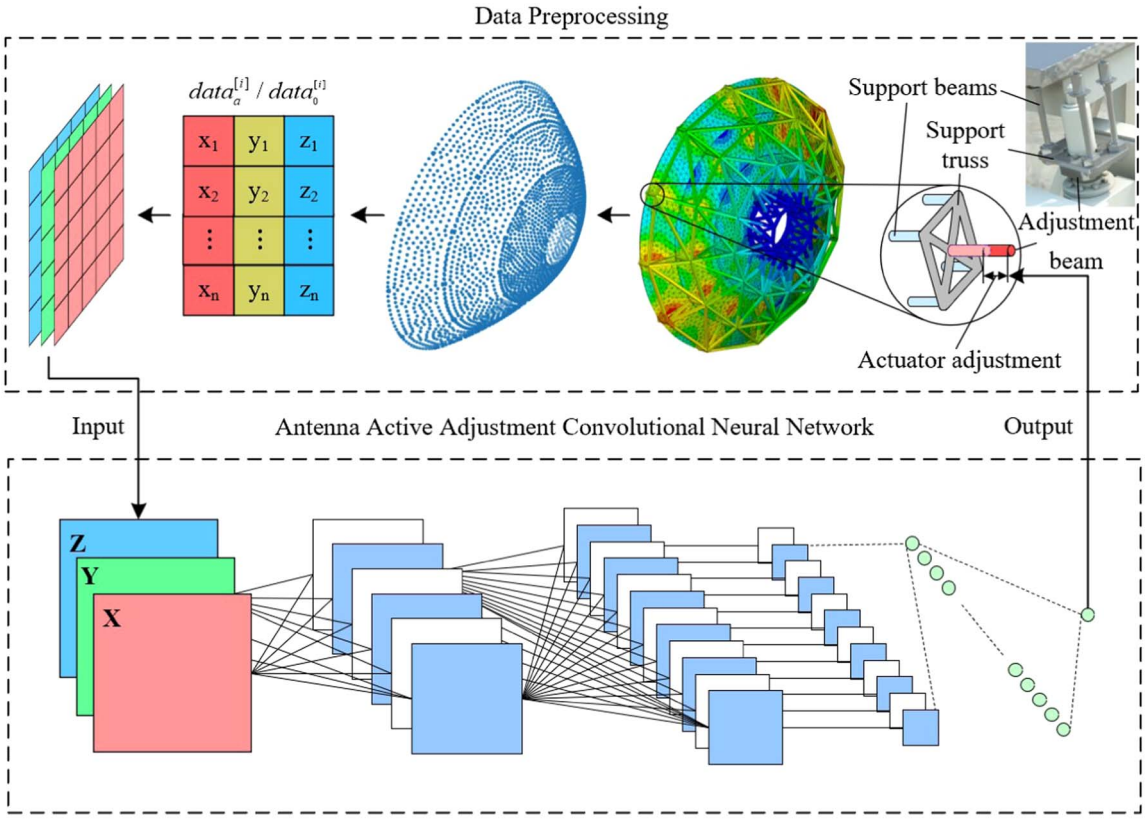

Figure 1. The overall flow of convolutional neural network model construction and training method for active antenna adjustment.

so that the two convolutions of the step size of 2. At this point, if the direct boundary filling makes the output of the four convolutions of the first layer of the same size, there will be a decline in training efficiency, so it can be supplemented with the use of jump connections, the output of the 5×5 convolution and 7×7 convolution is connected to the output of the third layer of the convolution to improve the training efficiency.

In order to improve the stability of neural network convergence during the training process and make smooth convergence, a normalization layer is usually used between the connected convolutional layers. However, since batch normalization (Ioffe & Szegedy 2015) leads to differences in the features of different batches of data, in order to reduce the impact of data changes and make the convolutional neural network have a better generalization effect on unseen data, and then more accurately predict the adjustment value of the actuator, the normalization layer of this model uses a layer normalization layer (Ba et al. 2016), and the computational formula is:

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \varepsilon}} \times \gamma + \beta \tag{2}$$

where x is the set of inputs to the layer, y is the set of outputs from the layer, E represents the mean, Var represents the variance, ε in the denominator is a very small number that serves to prevent the numerical computation from being unstable, and γ and β are the parameters of the learnable affine transformation.

For a single panel controlled by an actuator, a neural network model structure can be built as shown in Figure 2 (without the emphasis layer), which has five logical layers, with every two layers outputting a convolution of the same size connected in the depth direction, with an additional layer of normalization after each convolution, and using a Gaussian Error Linear Unit (GELU) activation function as shown in Equation (3) (Hendrycks & Gimpel 2016). Specifically, the first layer uses four convolutional kernels of different sizes, where the 1×1 convolutional kernel and the 3×3 convolutional kernel leave the feature map size unchanged and concatenate the outputs of both. the 5×5 convolutional kernel and the 7×7 convolutional kernel shrunk the feature map's by half, concatenating their outputs by jumping to the outputs of the third layer. The second layer used a 1×1 convolutional kernel and a 3×3 convolutional kernel that concatenated their outputs, leaving the feature map size unchanged. The third layer used a 3×3 convolution kernel with the feature map size halved and jump-connected its output to the output of the convolution kernel

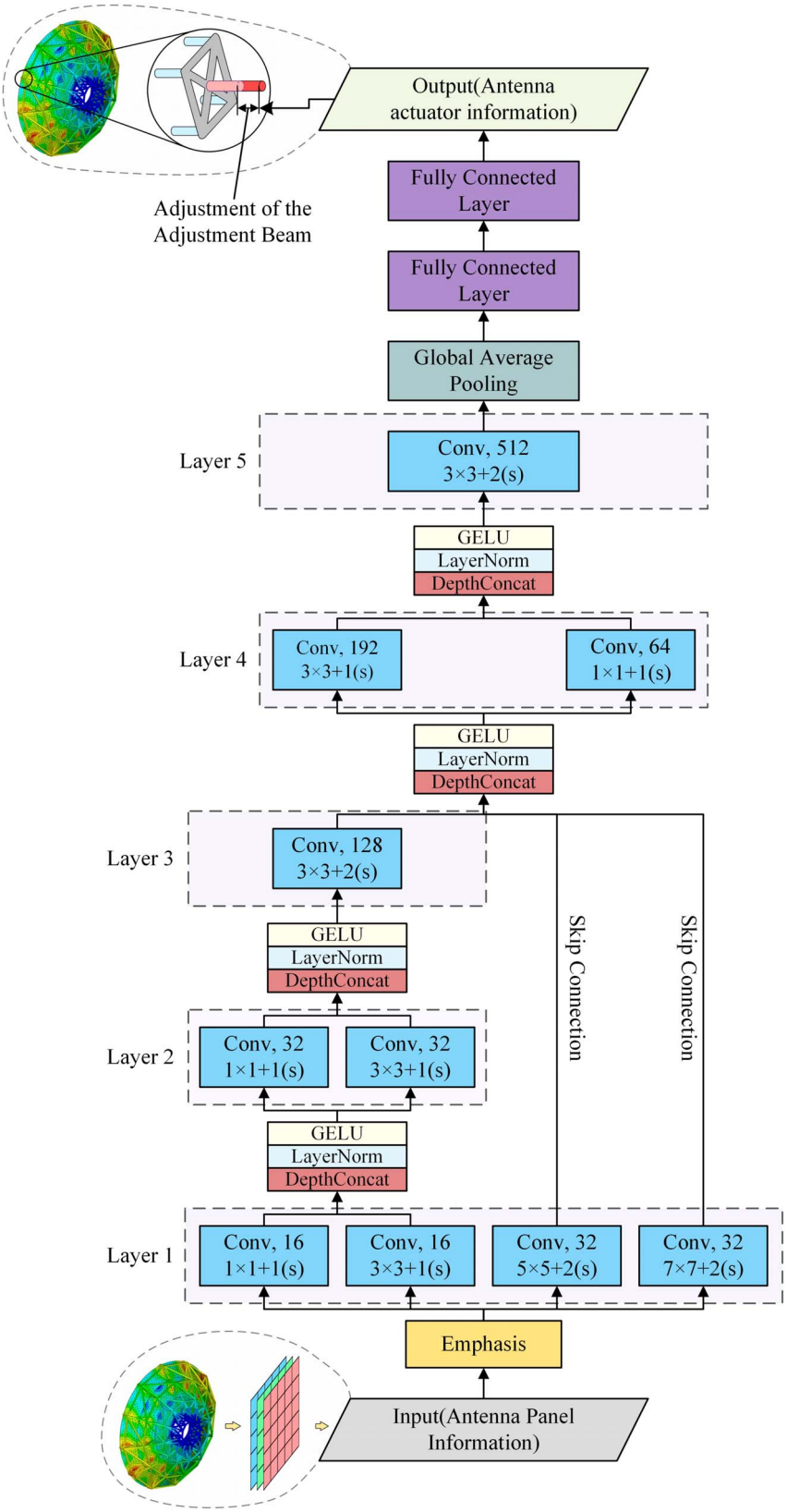

Figure 2. Antenna active adjustment convolutional neural network model.

with the halved feature map of the first layer. The fourth layer used a 1×1 convolution kernel and a 3×3 convolution kernel, concatenating their outputs with the feature map size unchanged. The fifth layer used only 3×3 convolution kernels, but halved the feature maps and increased the number of channels to compress the information for final information extraction using global average pooling.

$$\mathrm{GELU}(x) = 0.5 \times x \times (1 + \mathrm{Tanh}(\sqrt{2/\pi} \times (x + 0.044\,715 \times x^3))) \tag{3}$$

where x represents the input and Tanh is the hyperbolic tangent function. The above method is more applicable to the case of an actuator controlling a single panel. For the overall reflector of a multi-panel splice, i.e., a single actuator controlling multiple panels (usually one actuator controlling multiple panels), the method will take the influence of panels that are not directly connected to the actuator into account with the same weight, which affects the training results. Therefore, when using the panel position to predict the actuator adjustment value, the four directly connected panels should be given higher weights to attract the neural network's "attention", so this paper designs an autonomously learnable "emphasis layer.". This layer allows the neural network to autonomously choose the importance of each panel node to the connected/unconnected actuators. At the same time, in order to keep the three-dimensional coordinates of the same node scaled equally to prevent positional distortion, the emphasis layer uses a single weight to scale the *xyz* values of the nodes by the same size. This module is placed at the very bottom of the model structure in Figure 2 (near the input end), i.e., the first layer after the input of antenna panel data, and is used to redistribute the weights of the different panel nodes of the input. This emphasis layer has the same dimensions as the input data, and the computational formula for this layer is:

$$\mathbf{O} = \mathbf{I} \odot \mathbf{W} \tag{4}$$

Where $\boldsymbol{I}$ is the input of the layer, $\boldsymbol{W}$ is the parameter matrix of the layer and $\boldsymbol{O}$ is the output of the layer, i.e.,

$$\mathbf{I} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{bmatrix} \tag{5}$$

$$\mathbf{X} = \begin{bmatrix} \mathrm{x}_{11} & \mathrm{x}_{12} & \cdots & \mathrm{x}_{1n} \\ \mathrm{x}_{21} & \ddots & & \\ \vdots & & \ddots & \vdots \\ \mathrm{x}_{n1} & & & \mathrm{x}_{nn} \end{bmatrix} \tag{6}$$

$$\mathbf{Y} = \begin{bmatrix} \mathrm{y}_{11} & \mathrm{y}_{12} & \cdots & \mathrm{y}_{1n} \\ \mathrm{y}_{21} & \ddots & & \\ \vdots & & \ddots & \\ \mathrm{y}_{n1} & & & \mathrm{y}_{nn} \end{bmatrix} \tag{7}$$

$$\mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ \mathrm{y}_{21} & \ddots & & \\ \vdots & & \ddots & \\ z_{n1} & & & z_{nn} \end{bmatrix} \tag{8}$$

$$\mathbf{W} = [\mathbf{w}] = \begin{bmatrix} \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & \ddots & & \\ \vdots & & \ddots & \\ w_{n1} & & & w_{nn} \end{bmatrix} \end{bmatrix} \tag{9}$$

$$\mathbf{O} = \begin{bmatrix} \mathbf{X} \odot \mathbf{w} \\ \mathbf{Y} \odot \mathbf{w} \\ \mathbf{Z} \odot \mathbf{w} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \mathrm{x}_{11} \cdot w_{11} & \mathrm{x}_{12} \cdot w_{12} & \cdots & \mathrm{x}_{1\mathrm{n}} \cdot w_{1n} \\ \mathrm{x}_{21} \cdot w_{21} & \ddots & & \\ \vdots & & \ddots & \\ \mathrm{x}_{n1} \cdot w_{n1} & & & \mathrm{x}_{nn} \cdot w_{nn} \end{bmatrix} \\ \begin{bmatrix} \mathrm{y}_{11} \cdot w_{11} & \mathrm{y}_{12} \cdot w_{12} & \cdots & \mathrm{y}_{1n} \cdot w_{1n} \\ \mathrm{y}_{21} \cdot w_{21} & \ddots & & \\ \vdots & & \ddots & \\ \mathrm{y}_{n1} \cdot w_{n1} & & & \mathrm{y}_{nn} \cdot w_{nn} \end{bmatrix} \\ \begin{bmatrix} z_{11} \cdot w_{11} & z_{12} \cdot w_{12} & \cdots & z_{1n} \cdot w_{1n} \\ z_{21} \cdot w_{21} & \ddots & & \\ \vdots & & \ddots & \\ z_{n1} \cdot w_{n1} & & & z_{nn} \cdot w_{nn} \end{bmatrix} \end{bmatrix} \tag{10}$$

3. Convolutional Neural Network Training for Antenna Active Adjustment

In order to transform the data into a form that can be input into a convolutional neural network as well as to preserve the a priori knowledge of the reflector panel nodes, data preprocessing is required to transform the raw data into the desired form of an ordered 3D array. The initial parameters of the model need to be specified and the loss function defined before the model is trained. The computational optimizer and dynamic learning rate methods during training are also described in this section.

3.1. Data Preprocessing

Since the panel nodes contain a priori knowledge similar to that of a picture, but the nodes of the generated data set are not ordered. If the input to the network is only reorganized without ordering, it will result in the loss of this a priori knowledge, causing a decrease in prediction accuracy. In order to retain this a priori knowledge and improve the stability of convergence, this section performs sorting, reorganization, and feature scaling on the source data, and the general process is shown in Figure 1(a). Sorting and reorganization transforms the two-dimensional array containing the coordinates of the panel nodes before and after adjustment into a three-dimensional array of coordinate changes that approximate the real node layout order. At this point the change value corresponds to the node coordinates, in this three-dimensional array from left to right

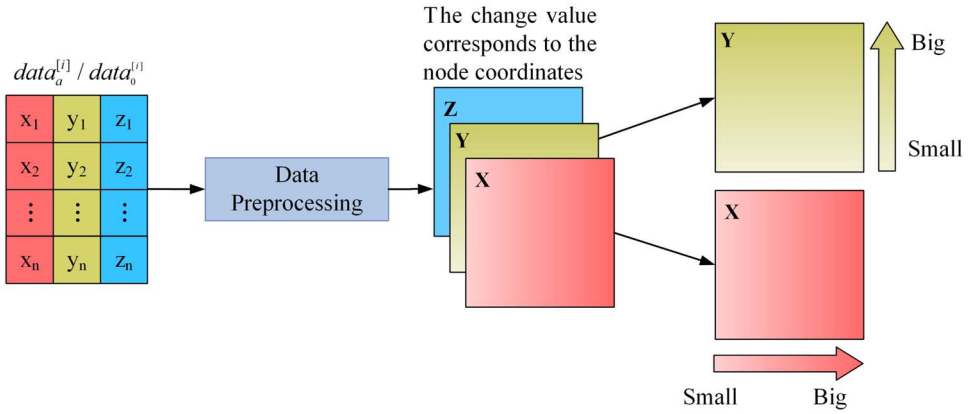

Figure 3. Purpose of data preprocessing.

x-coordinate gradually increases, from bottom to top y-coordinate gradually increases, as shown in Figure 3. Finally this coordinate change value and the actuator adjustment value are feature scaled. The preprocessed panel node coordinate change value is used as the feature of the neural network described in Section 2, and the actuator adjustment value matching the node change value is used as its label.

The reason for using the node coordinate changes here instead of directly using the adjusted position coordinates as the neural network inputs is that the large reflector surface of a large reflector antenna has a huge reflector size, and if normalization is performed directly, it will result in the amount of coordinate changes and their minuteness such that the neural network will not be able to perceive the changes in the node positions, resulting in the failure of convergence in training the neural network.

Feature scaling means limiting the distribution of the input data to $[-1, 1]$ in order to better train the deep learning model. Specifically, data normalization has two main effects: (a). Help accelerate training: data normalization can make the gradient descent algorithm converge faster, thus accelerating the training process. (b). Avoid gradient vanishing or exploding: when the distribution of the input data is too large or too small, it will lead to gradient vanishing or exploding in the process of back-propagation, which can be avoided by data normalization. Since for different actuator adjustment values, this may result in uncertainty about the maximum value of the panel nodes, the practice of dividing the feature and label by the maximum adjustment value was adopted. At this point, some of the data will be slightly out of the range of $[-1, 1]$, but it has little effect on the stability of the training, which we refer to as feature scaling.

Since only feature scaling is performed for label, it is not repeated here. The processing flow for the feature is shown in Figure 4, where each set of data of the feature contains the pre-adjusted ideal/designed panel node coordinates $data_0^{[i]}$ and the post-adjusted panel node coordinates $data_a^{[i]}$ (In this paper, the designed paraboloid is used as the example). The processing flow can be divided into two parts. The first part is on the left side of Figure 4, which is used to determine the sorting order, the number of groups, and the number of nodes in each group after grouping. The reason for using i instead of j for processing here is that the positions of the deformed panel nodes will be shifted, potentially causing the sorted positions to be misaligned and affecting the processing results. The second part is on the right side of Figure 4, which serves to process the data set sequentially using the sorting order and grouping results of the first part.

In Figure 4, $sort\_sb(A_0, 0)$ denotes that column 0 of A_0 is sorted in order from smallest to largest and extended to other columns. $split(B)$ denotes that C is grouped, assuming that the group with the most elements contains h elements, which will be described in the next paragraph due to the complexity of the grouping method. $sort\_bs(C_j, 1)$ denotes that the 1st column of C is sorted in descending order and extended to the other columns. $sort\_o(O, OX)$ denotes that the rows of O are sorted by OX. $split\_o(P, OY)$ indicates that P is grouped according to the number of elements in each group in OY. The purpose of the grouping operation is to increase the dimension of the array and produce the effect of increasing the x-coordinate sequentially in that dimension. After grouping, if the number of elements in the group is less than h, it is necessary to add element 0 to both ends of the array, so that the number of elements in each group is m. $R.add(Q)$ denotes the addition of the array Q to the end of the array R. $transpose(R)$ denotes that the order of the $1/2/3$ axis is adjusted without affecting the order in which the data are arranged, i.e., the four-dimensional array with the original shape of $t \times w \times h \times 3$ is transformed into the shape of $t \times 3 \times w \times h$. It should be reminded that for the convenience of writing the model code, 0 elements can be added at both ends to make w equal to h. $pseudo\_norm(S)$ denotes the feature scaling operation of S.

As a reminder, since feature scaling is also applied to the labels. Therefore, the final neural network prediction is not the

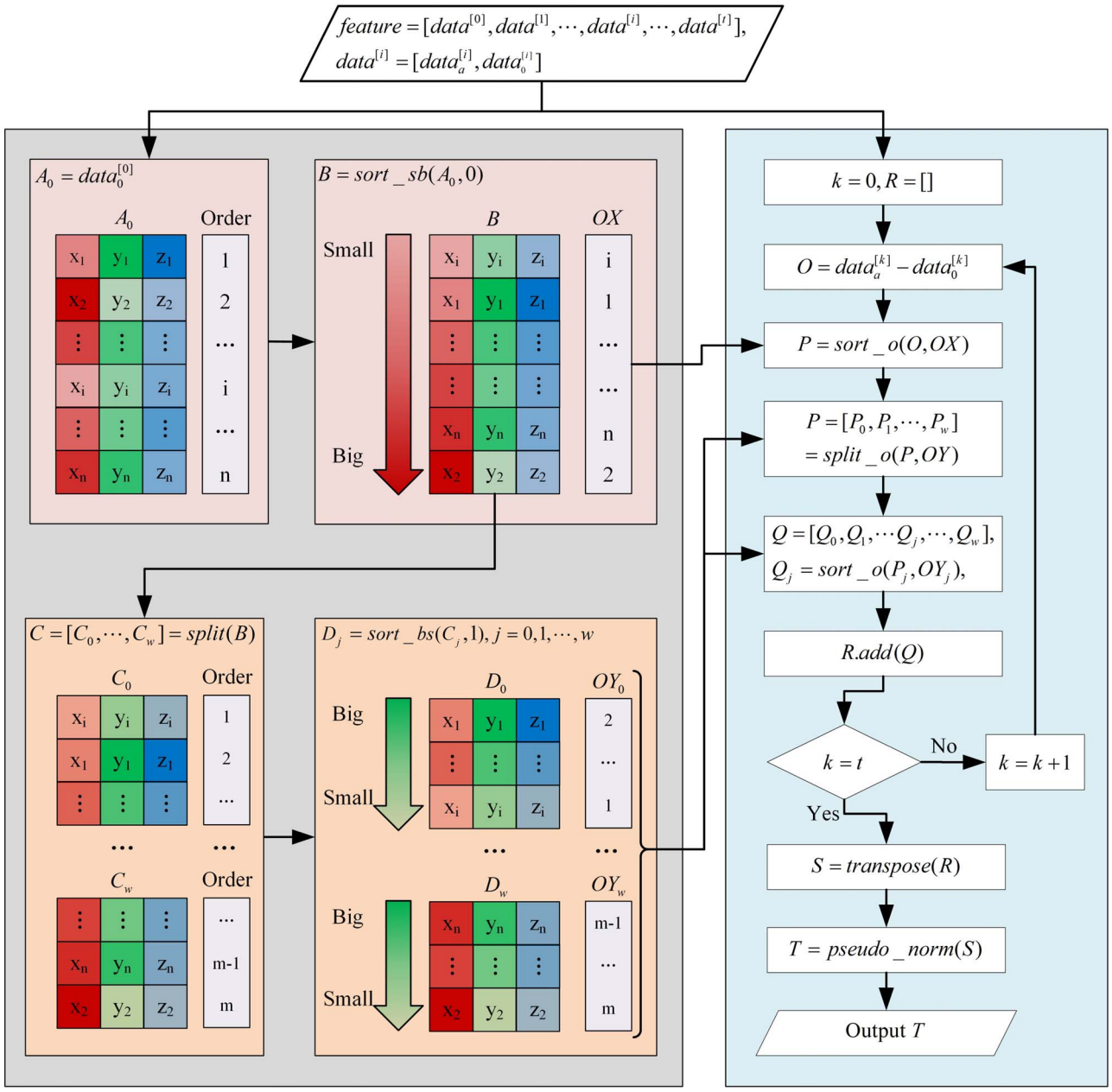

Figure 4. Data preprocessing process. A_0, B and other matrices in the figure, the darker the color indicates a larger value.

true value, which is the inverse of the predicted value multiplied by the multiplier that was shrunk when pseudo-normalized.

3.2. Training Strategies

3.2.1. Model Parameter Initialization

Model initialization is the process of assigning initial values to the parameters (weights and biases) of a neural network before it is trained. Proper initialization helps prevent problems such as vanishing or exploding gradients, speeds up convergence, and helps the neural network learn more efficiently.

In order to avoid any impact on the node weights of the input, the initialization of the parameters of the emphasis layer is carried out using constant initialization. This ensures that the relative positions of the node coordinates of the input neural network are unchanged and have not been misaligned. Due to the use of the GELU activation function, the parameters of the neural network other than the emphasis layer are initialized with "kaiming normal" for numerical stability (He et al. 2015).

3.2.2. Loss Function

The coordinates of the panel nodes that have been preprocessed by 3.1 are used as the convolutional neural network inputs, and the mean absolute error (MAE) between the corresponding actuator adjustment value and the neural network outputs is used as the loss function, i.e., the L1 loss, which is computed as shown in Equation (11):

$$\mathrm{L1Loss} = \frac{l_1 + l_2 + \cdots + l_i + \cdots + l_p}{p},\ l_i(a_i, a_i') = |a_i - a_i'| \tag{11}$$

Here a_i is the adjustment value of actuator i, a_i is the predicted value of the neural network output of actuator i, and p is the number of actuators.

However, subsequent experiments revealed that it is difficult for a single model to achieve accurate prediction of the adjustment value of each actuator, so in this paper, we designed the emphasis layer described in Section 2 and changed the output to a single actuator, i.e., p in the loss function Equation (11) is 1.

The same model is used for training, and p actuators are trained sequentially for a total of p times, and p sets of neural network parameters can be obtained in the end. For prediction, it is only necessary to replace the p sets of neural network parameters obtained during the above training to realize the prediction of the adjustment value of these actuators.

3.2.3. Computational Optimizer

Adaptive Moment Estimation (ADAM) (Kingma & Ba 2014) and Stochastic Gradient Descent (SGD) are two optimization algorithms commonly used for training neural networks. Both algorithms aim at minimizing the loss function. However, they differ in the way they update the model parameters during training, and there is no superiority between the two. It is usually necessary to empirically test the algorithm for a specific problem and choose the one that works better. For the model in this paper, learning rate and weight decay are the main hyper-parameters that need to be tuned for testing, and the rest of the parameters use the recommended values from the literature.

3.2.4. Dynamic Learning Rate

When using a gradient descent algorithm to optimize the loss function, as it gets closer to the local minimum of the loss, the learning rate should become smaller to bring the model as close as possible to this minimum point. The cosine annealing learning rate (Loshchilov & Hutter 2016) is an effective dynamic learning rate strategy, where the learning rate value first decreases slowly as the epoch increases, then accelerates, and finally decreases slowly again. This learning rate adjustment strategy makes the learning rate transition smoothly between high and low learning rates, avoids sudden jumps in the learning rate, and makes the training process more stable. At the same time, this learning rate adjustment strategy helps to improve the convergence and make the model parameters easier to approach the optimal solution.

4. Example

In this section, the classical 8-meter reflective surface antenna will be used as an example to validate the active surface antenna tuning method described in this paper, using the neural network model constructed in Section 2 and applying the data preprocessing method and training strategy described in Section 3.

4.1. Dataset Generation

This data set is generated by modeling the proposed method in literature (Ban et al. 2023), which is based on the active adjustment of the antenna active main reflector finite element model of the simplified actuator. This model consists of three parts: the backup structure, the simplified actuator and the panel, where the backstay and the simplified actuator are composed of spatial beam units and the panel is composed of triangular elastic shell units. The antenna panel profile is adjusted by giving the actuator an adjustment value.

In this paper, we apply the 8 m antenna model in the literature, which the actuator pointing will change with the backstay, the adjustment direction will change according to the real pointing after deformation, and the nodes of the panels can also be displaced along different directions. As shown in Figure 5, the model consists of 36 panels with about 99 nodes each, and the panels are supported and adjusted by 48 actuators. These 48 actuators are numbered for ease of description below. When generating the data set, the model is given a vertically downward force to simulate the elevation condition. When generating the data set, the model was given a downward force to simulate the work conditions in the elevated sky under the influence of gravity. Due to the lack of comprehensiveness of the working conditions and influencing factors, a randomized adjustment scheme was adopted in order to increase the complexity of the data. That is, 48 actuators on the model were randomly adjusted within the adjustment range of [0, 10] mm to obtain the adjusted nodal displacements of the antenna panel under gravity. Its actuator adjustment value and its corresponding panel node coordinates are a set of data set.

A total of 20480 sets of data sets are generated cyclically with the above method, and the data generated in present section are preprocessed using the data preprocessing strategy in Section 3.1 to transform the 20480 sets of 3564×6 arrays of source nodes into a pseudo-normalized ordered array of $20480 \times 3 \times 64 \times 64$. The corresponding actuator adjustment value of these 20480 groups of panel nodes are pseudo-normalized and transformed into a $20\,480 \times 48$ array. For

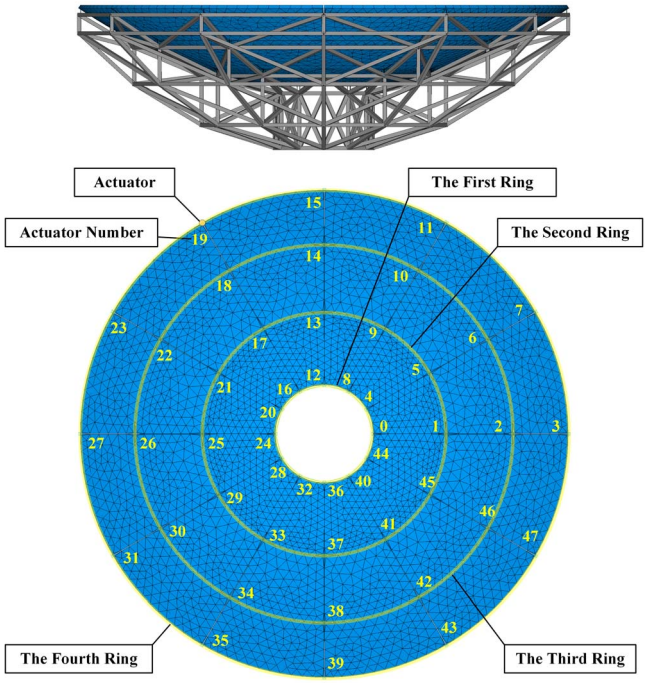

Figure 5. 8 m antenna model based on the simplified actuators.

training, the data set is divided into a training set and a validation set, with 18432 groups in the training set and 2048 groups in the validation set.

4.2. Training and Results

Build the convolutional neural network model in Section 2, set the number of outputs to 1, i.e., train only one actuator per training. Use the method in Section 3.2.1 to initialize the model parameters, which was tested to work best when the emphasis layer was initialized to 0.0001. Set the loss function to L1 loss in Section 3.2.2. It is tested that for this 8 m antenna data set, the ADAM algorithm mentioned in Section 3.2.3 converges more stable and has higher accuracy, and it is also tested that the hyper-parameter learning rate of 0.0001 and the weight decay of 3×10^{-6} are more appropriate values. The dynamic learning rate method without using restart strategy in Section 3.2.4 is adopted. The model training adopts the Mini-Batch Gradient Descent method, and the batch size is set to 128, and the number of training epoch of the model is 150. It has been measured that more epoch of training will give a tiny improvement in accuracy, but it is not significant, and 150 is a compromise between accuracy and efficiency.

Since the antenna model in Section 4.1 is a symmetric structure, it is representative to select one actuator per ring for performance testing. Therefore, in order to prove the effect of emphasis layer on the improvement of accuracy, the actuators 8, 9, 10 and 11 are selected for training in this paper. Comparing their training with and without the emphasis layer, the loss changes during the training process are shown in Figure 6 (since the actuator adjustment value are pseudo-normalized, the loss has been transformed into the real error here for the sake of easy observation and description of the specific accuracy). The left column of the figure shows the training process of the model without the emphasis layer and the right column shows the training process of the model with the emphasis layer. It can be seen that the model containing the emphasis layer converges faster than the model without the emphasis layer and has a high convergence accuracy, which indicates that the emphasis layer of the model does help to improve the overall performance and generalization ability of the model.

Extracting the emphasis layer weights to validate the role of the emphasis layer in the model, in order to make it more representative, it is necessary to select four actuators located in different rings, and all four actuators have different orientations. The actuators selected here as representative actuators are numbered 9, 20, 35, and 46. As shown in Figure 7, the right column is a visualization of the weight matrix of the emphasis layer of the model when the actuator is trained. Since the size of the weights is only related to the size of the absolute value, which has nothing to do with the positive and negative values, in order to make it easier to observe here, all the weights are taken as their absolute values, and the left column is a dot matrix of the right weights mapped on the nodes of the panel. It can be seen that the coordinates with larger weights have the same orientation as the antenna panel nodes in the simulation model, and the weights of the nodes around the actuator are significantly larger than the other positions. It can thus be demonstrated that sorting and reorganizing the data using the data preprocessing method in Section 3.1 results in the node coordinates being located at the real locations overall, and it can also be demonstrated that the emphasis layer is able to find the location of the corresponding panel node through learning, and is interested in increasing the weight on the location of that node in line with the expectation of emphasizing the directly connected panel nodes.

All the actuators are trained sequentially and the trained model parameters of each actuator are saved for use in the following tests. The training results are shown in Tables 1–2, which compares the convergence accuracies of all the actuators with and without emphasis layers, and the table shows that the training accuracies differ depending on the ring where the actuators is located. As shown in Figure 8, the model with emphasis layer for the first ring actuators have a mean error reduction of 0.00715 mm compared the model without emphasis layer, which is a relative error reduction of 79.0%. The second ring actuators mean error reduced by 0.00514 mm, relative error reduced by 62.8%. The third ring actuators mean error reduced by 0.00436 mm, relative error reduced by 59.6%. The fourth ring actuators mean error reduced by 0.00389 mm

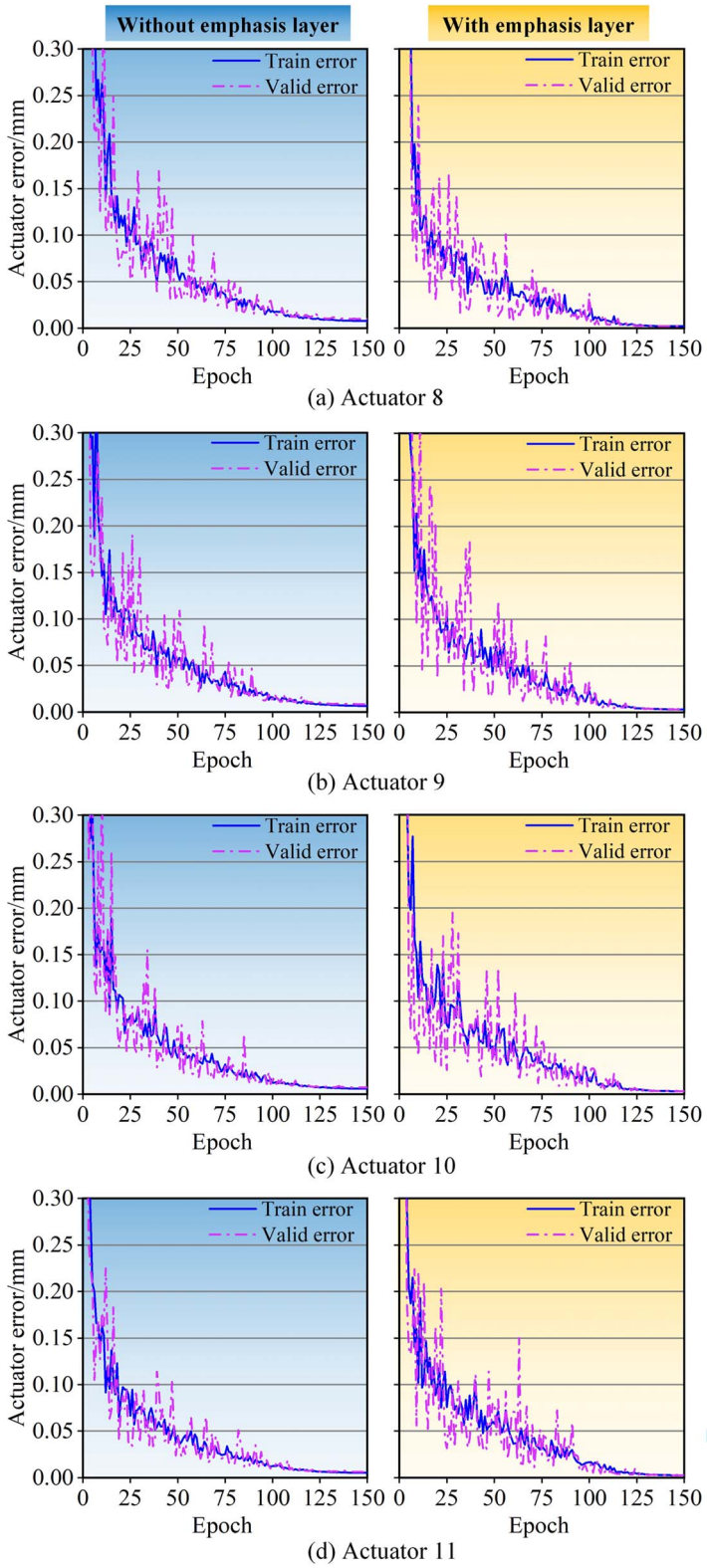

Figure 6. Error changes of the training process of the actuators 8, 9, 10 and 11 when the model contains an emphasis layer or not. The left side shows the change in error of the model without the emphasis layer and the right side shows the change in error with the emphasis layer.

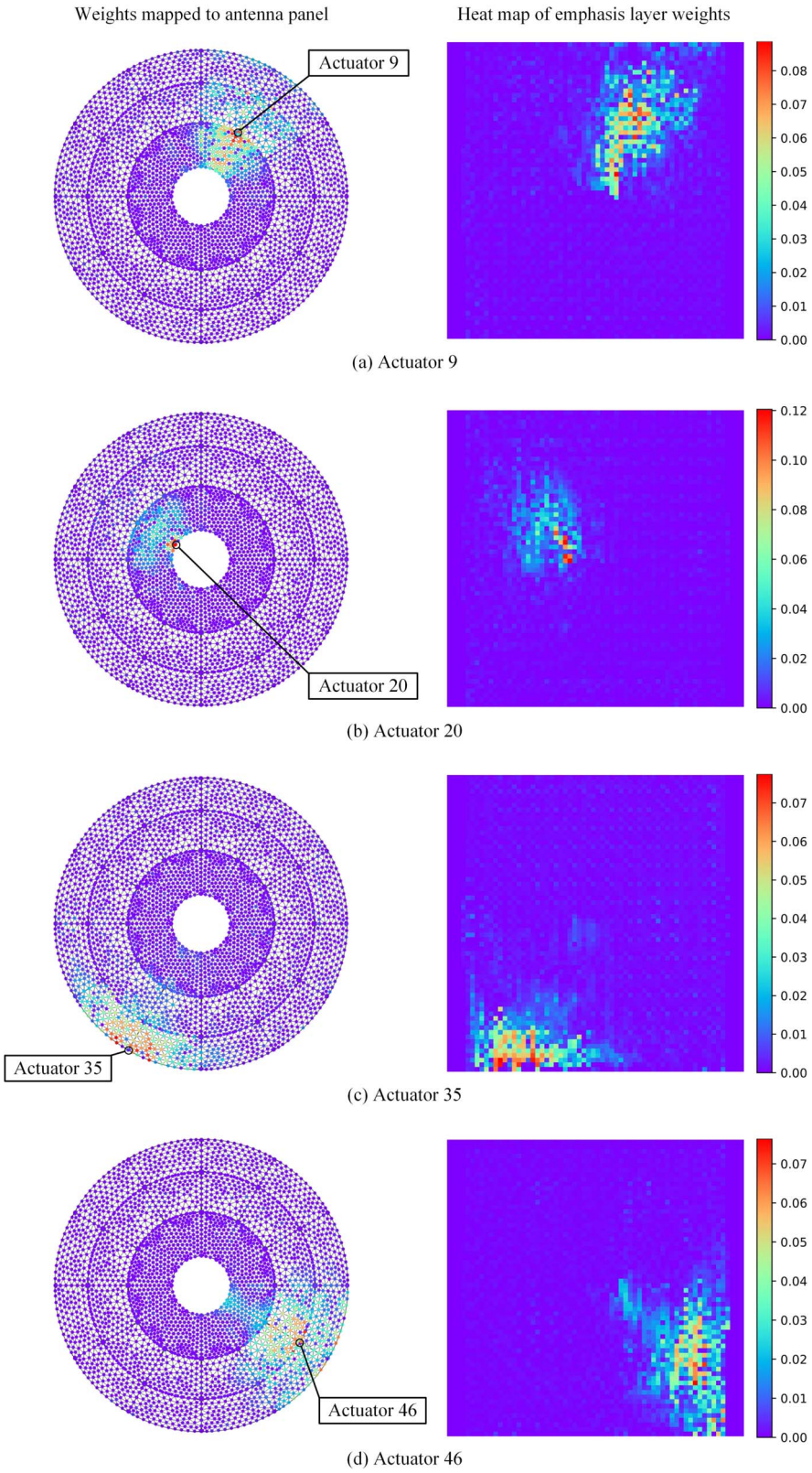

Figure 7. Actuators 9, 20, 35, and 46 emphasis layer weight heat map (right) and weight mapping to antenna panel (left)

Table 1
Comparison of the Error of First Ring/Second Ring Actuators with and Without Emphasis Layer

Actuator Number on the First Ring	Validation Set Error without Emphasis Layer (mm)	Validation Set Error with Emphasis Layer (mm)	Actuator Number on the Second Ring	Validation Set Error without Emphasis Layer (mm)	Validation Set Error with Emphasis Layer (mm)
0	0.00891	0.00161	1	0.00887	0.00294
4	0.00889	0.00204	5	0.00758	0.00277
8	0.00999	0.00199	9	0.00848	0.00331
12	0.00896	0.00171	13	0.00896	0.00304
16	0.00962	0.00206	17	0.00805	0.00282
20	0.00864	0.00169	21	0.00842	0.00329
24	0.00819	0.00176	25	0.00922	0.00279
28	0.00904	0.00190	29	0.00809	0.00330
32	0.00834	0.00222	33	0.00814	0.00267
36	0.00903	0.00206	37	0.00806	0.00336
40	0.00977	0.00195	41	0.00732	0.00331
44	0.00927	0.00185	45	0.00709	0.00302
Mean	0.00905	0.00190	Mean	0.00819	0.00305

Table 2
Comparison of Error of Third/Fourth Ring Actuators with and Without Emphasis Layer

Actuator Number on the Third Ring	Validation Set Error without Emphasis Layer (mm)	Validation Set Error with Emphasis Layer (mm)	Actuator Number on the Fourth Ring	Validation Set Error without Emphasis Layer (mm)	Validation Set Error with Emphasis Layer (mm)
2	0.00768	0.00296	3	0.00610	0.00237
6	0.00754	0.00305	7	0.00570	0.00240
10	0.00722	0.00321	11	0.00618	0.00263
14	0.00780	0.00295	15	0.00596	0.00231
18	0.00731	0.00236	19	0.00747	0.00236
22	0.00725	0.00316	23	0.00642	0.00259
26	0.00682	0.00263	27	0.00637	0.00181
30	0.00699	0.00313	31	0.00671	0.00234
34	0.00754	0.00287	35	0.00666	0.00298
38	0.00818	0.00299	39	0.00633	0.00231
42	0.00713	0.00312	43	0.00627	0.00247
46	0.00641	0.00313	47	0.00591	0.00287
Mean	0.00732	0.00296	Mean	0.00634	0.00245

and relative error reduced by 61.4%. This proves that the law that models containing emphasis layers have higher convergence accuracy than models without emphasis layers holds true for all actuators. It can be seen that the neural network containing the emphasis layer has a greater improvement in the prediction accuracy of the reflector antenna actuator adjustment value, which is extremely important for the adjustment value of the active surface of the reflector antenna.

The final panel node accuracy test is performed. Using the method of Section 4.1, 128 sets of data were generated as a test set for final accuracy assessment. The test starts with building the convolutional neural network model in Section 2, after which the corresponding model parameters of different actuators trained above are loaded to predict different actuator adjustment value. In this paper, we use a graphics card model TUF-RTX4080-O16G to predict 128 sets of data at the same time, with a predicting time of 1.8 s, which finally results in the mean error of the actuators adjustment values in this test set of 0.00252 mm. The actuator positions predicted from the neural network were input into the model of Section 4.1 to derive the corresponding panel node coordinates, which were then used to calculate the root mean square (rms) with the panel node coordinates of this test set, resulting in a final surface error of 0.00523 mm.

5. Conclusions

In this paper, we propose a convolutional neural network-based antenna active surface adjustment method for actuator adjustment value calculation, as well as a data preprocessing method adapted to the input-output form of this convolutional neural network model, and we summarize the strategy used to train the model. The convolutional neural network model adopts a multi-scale convolutional approach, and uses jump connections to connect the feature maps output from the large convolutional kernel at the bottom layer of the model to the

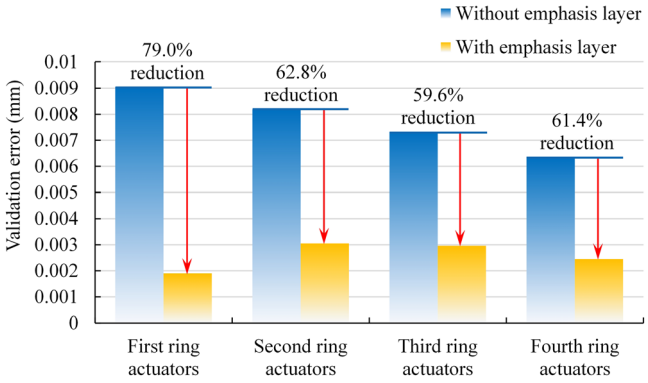

Figure 8. Comparison of the error of each ring actuator with and without emphasis layer.

output at the top layer of the model. An autonomously learnable emphasis layer is designed to realize that the model can autonomously learn which are the relatively important panel nodes when training different actuators, which makes the model accuracy significantly improved. Meanwhile, due to the retrainable nature of the neural network model, the model parameters can be fine-tuned by using the usage data as a training set during subsequent use to adapt to the errors caused by the aging of the components and other conditions. Compared with the traditional method of calculating the actuator adjustment value for active surface adjustment, the model and method have wide application prospects and sufficient development potential, and are also smarter, which can provide a reference for other large aperture high-precision antenna active surface adjustment schemes under other working conditions and complex environmental conditions.

Acknowledgments

This work was supported by the National Key R&D Program of China No. 2021YFC220350, the National Natural Science Foundation of China Nos. 12303094 & 52165053, the Natural Science Foundation of Xinjiang Uygur Autonomous Region Nos. 2022D01C683, the China Postdoctoral Science Foundation Nos. 2023T160549 & 2021M702751, and in part by Guangdong Basic and Applied Basic Research Foundation Nos. 2020A1515111043 & 2023A1515010703.

ORCID iDs

You Ban https://orcid.org/0000-0001-8461-7603

References

Ba, J. L., Kiros, J. R., & Hinton, G. E. 2016, arXiv:1607.06450
Baars, J. W., & Kärcher, H. J. 2018, Radio Telescope Reflectors: Historical Development of Design and Construction (Astrophysics and Space Science Library), Vol. 447 (Berlin: Springer)
Ban, Y., Chai, P., Xu, Q., & Feng, S. 2023, RAA, 23, 075014
Ban, Y., Feng, S., Vandenbosch, G. A. E., et al. 2020, IEEE Trans. Antennas Propag., 68, 5855
Ban, Y., Gao, T., Wang, N., et al. 2024, SSPMA, 54, 219507
Ban, Y., Liu, Y., Wang, N., Guljaina, K., & Feng, S. 2024, SSPMA, 54, 219503
Bolli, P., Orlati, A., Stringhetti, L., et al. 2015, JAI, 4, 1550008
Dong, J., Jin, H., Ye, Q., et al. 2016, Proc. SPIE, 9913, 49
Fang, J., Xie, M., He, X., et al. 2022, Mater. Today Commun., 33, 104900
Fu, L., Zhong, W., Qiao, H., Liu, G., & Qian, H. 2015, Acta Astronomica Sinica, 56, 378
Goh, G. B., Hodas, N. O., & Vishnu, A. 2017, J. Comput. Chem., 38, 1291
Hadji, I., & Wildes, R. P. 2018, arXiv:1803.08834
He, K., Zhang, X., Ren, S., & Sun, J. 2015, in 2015 IEEE International Conference on Computer Vision (ICCV) (Santiago: IEEE), 1026
Hendrycks, D., & Gimpel, K. 2016, arXiv:1606.08415
Hughes, D., Correa, J., Schloerb, F., et al. 2010, Proc. SPIE, 7733, 773312
Ioffe, S., & Szegedy, C. 2015, in Proc. 32nd Int. Conf. Mach. Learn., 37 (Lille: PMLR), 448
Kingma, D. P., & Ba, J. L. 2014, arXiv:1412.6980
Lian, P., Wang, C., Xue, S., et al. 2021, IEEE Trans. Antennas Propag., 69, 6351
Liu, Y., Zhang, Y., Wang, Y., et al. 2023, IEEE Transactions on Neural Networks and Learning Systems, 1, 1
Loshchilov, I., & Hutter, F. 2016, arXiv:1608.03983
Thuerey, N., Holl, P., Mueller, M., et al. 2021, arXiv:2109.05237
Wang, N., Xu, Q., Ma, J., et al. 2023, SCPMA, 66, 289512
White, E., Ghigo, F. D., Prestage, R. M., et al. 2022, A&A, 659, A113
Xu, Q., & Wang, N. 2016, Proc. SPIE, 9906, 1936
Zhou, F., Jin, L., & Dong, J. 2017, Chinese J. Comput., 40, 1229
Zhou, S. K., Greenspan, H., Davatzikos, C., et al. 2021, Proc. IEEE, 109, 820