Double Pruning Structure Design for Deep Stochastic Configuration Networks Based on Mutual Information and Relevance

YAN Aijun[1, 2, 3], LI Jiale[1, 2], TANG Jian[1]

(1. *Faculty of Information Technology*, *Beijing University of Technology*, *Beijing* 100124, *China*;
2. *Engineering Research Center of Digital Community*, *Ministry of Education*, *Beijing* 100124, *China*;
3. *Beijing Laboratory for Urban Mass Transit*, *Beijing* 100124, *China*)

Abstract: Deep stochastic configuration networks (DSCNs) produce redundant hidden nodes and connections during training, which complicates their model structures. Aiming at the above problems, this paper proposes a double pruning structure design algorithm for DSCNs based on mutual information and relevance. During the training process, the mutual information algorithm is used to calculate and sort the importance scores of the nodes in each hidden layer in a layer-by-layer manner, the node pruning rate of each layer is set according to the depth of the DSCN at the current time, the nodes that contribute little to the model are deleted, and the network-related parameters are updated. When the model completes the configuration procedure, the correlation evaluation strategy is used to sort the global connection weights and delete insignificance connections; then, the network parameters are updated after pruning is completed. The experimental results show that the proposed structure design method can effectively compress the scale of a DSCN model and improve its modeling speed; the model accuracy loss is small, and fine-tuning for accuracy restoration is not needed. The obtained DSCN model has certain application value in the field of regression analysis.

Keywords: Deep Stochastic Configuration Networks, Mutual Information, Relevance, Hidden Node, Double Pruning

1 Introduction

In recent years, deep neural networks (DNNs) have been widely used in image recognition, data modeling, and predictive control for industrial systems[1-3]. The performance of a deep learning model is constrained by the network depth and network width[4,5]. The larger the network depth is, the stronger the nonlinear expression ability of the corresponding DNN. The larger the network width is, the more features can be learned at each layer of the network[6]. When faced with complex high-dimensional data, the performance of a deep model is generally guaranteed by increasing the number of network nodes or hidden layers, but these approaches lead to increased algorithm complexity and overfitting problems. Finding the appropriate network width and depth while ensuring the accuracy of the deep model and improving the modeling speed is an important research direction with respect to optimizing DNNs.

At present, the commonly used methods for DNN compression include quantization, low-rank decomposition, knowledge distillation, and pruning. Quantization usually refers to replacing an original high bit width with a low bit width to achieve the goals of improving the parallel processing capability of the system and reducing the memory bandwidth. However, low-bit quantization, with its discrete nature, often leads to instability in the deep learning model training process and a serious accuracy loss[7,8]. A larger bit width forms a quantized deep learning model without significant speedup and scale changes[9]. Low-rank decomposition is often used to accelerate neural networks[10], but problems such as low compression rates and reduced accuracy rates occur[11]. Knowledge distillation[12] refers to distilling relevant knowledge from a trained teacher model to a student model to obtain higher accuracy, but this method has difficulty optimizing deeper DNNs, and a single static teacher model may introduce noise information when training the student model, causing the student model to be limited by its teacher model[13]. Compared with the above three compression strategies, the pruning method is more intuitive and can directly remove irrelevant parameters from a network to achieve sparseness in the deep learning model structure. According to different pruning objects, this method can be divided into unstructured pruning and structured pruning. Unstructured pruning generally removes redundant connections and is a type of fine-grained pruning. For example, Han et al.[9] proposed a hybrid deep compression strategy based on pruning, trained quantization and Huffman coding, thereby maximizing the compression of convolutional neural networks while considering the accuracy and hardware storage issues. Migue[15] proposed an amplitude-based pruning strategy, which describes pruning as an optimization problem and automatically learns the optimal number of weights for pruning in each hidden layer of the given neural network. In [16], an evaluation criterion based on the importance levels of activation weights was proposed to screen out weights with low contribution rates; this technique can effectively prune densely connected network parameters. Unstructured pruning methods induce small accuracy losses in deep learning models, and the accuracy can be recovered through fine-tuning. Structured pruning methods change the learning model's structure and belong to the coarse-grained pruning category. In [17], through Fisher's linear discriminant analysis approach, the paper pointed out that starting from the last convolutional layer, the activation values between neurons are highly irrelevant, so it is preferable to delete hidden nodes with less information. Luo[18] regarded filter pruning as an optimization problem and decided whether to prune according to the information contained in the next layer of a convolutional network. When a structured pruning method is adopted and the pruning object setting is unreasonable, the accuracy of the model will greatly decrease. In addition, a proportional pruning rate is set for the hidden layers during pruning, but it needs to be considered that the amounts of information contained in different hidden layers are generally different. This type of fixed pruning rate may lead to the incorrect deletion of some important hidden nodes in the network, which has irreversible effects on the perform- ance of a DNN.

Deep stochastic configuration networks (DSCNs)[19] were proposed by Wang and Li. A DSCN randomly generates hidden nodes through a supervision mechan- ism, and it easily produces nodes that contribute little to modeling, increasing the structural complexity of the network and the difficulty of modeling. Therefore, this paper proposes a double pruning structure design method based on mutual information and relevance for DSCNs, and this method is called mutual information-relevance-DSCNs (MI-R-DSCNs). The pruning opera-tion based on mutual information is aimed at the hidden network nodes. It calculates the mutual information values between the activation values of the hidden nodes and the output variable in a layer-by-layer manner. According to the mutual information values, the hidden nodes with low contributions are pruned, and the DSCN structures are pruned. The pruning

operation based on relevance is aimed at the connection weights between the hidden nodes. It analyzes the importance of each weight, filters out the unimportant weights and sets them to zero, and finally realizes structural sparseness. Through function approximation and regression experi- ments, it is proven that the method proposed in this paper can effectively compress the scale of a DSCN and improve its modeling speed.

The rest of this paper is arranged as follows. Section 2 introduces DSCNs and their existing problems, Section 3 presents the MI-R-DSCN algorithm, and Section 4 conducts an experimental evaluation of the proposed method. Finally, Section 5 is the conclusion.

2 Reviewing DSCNs and Analyzing Existing Problems

In this section, the basic theory, structure and existing problems of DSCNs are introduced.

2.1 Theory and Structure of DSCNs

Stochastic learning algorithms can effectively avoid the problem of falling into local optima due to gradient descent, and they have great potential in fast dynamic modeling and processing real-time data[20]. The stochastic configuration network (SCN)[21] proposed by Wang and Li is a typical stochastic learning algorithm. On this basis, Wang and Li proposed the DSCN algorithm, which has been applied in the field of industrial big data analysis and data flow learning[22-24]. Fig.1 shows the structure of a DSCN with two hidden layers, and we can see from the figure that this DSCN is a typical fully connected neural network. However, the obvious difference between a standard fully connected feedforward neural network and a DSCN is that the former only requires the output result to be represented by the fully connected learning between the last hidden layer and the output layer, while a DSCN requires the full connections between each hidden layer and the output layer, and such learned representations can provide greater flexibility in terms of regression and classification[19].

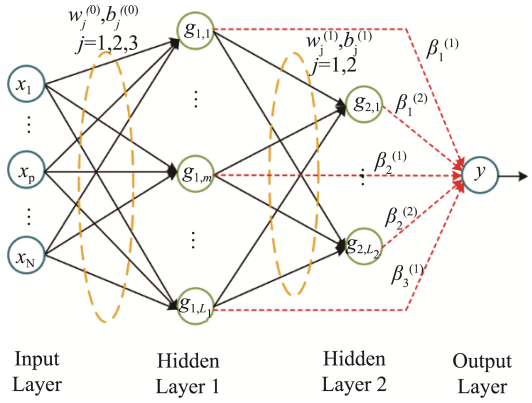

Fig.1 N-L_1-L_2-1 DSCN Model

The specific implementation process of a DSCN is as follows. Given a target function F: $R^d \rightarrow R$ and a training dataset $\{X, Y\}$ which the number of samples is N, the outputs of the DSCN can be denoted by:

$$F_{S_n}^{(n)} = \sum_{k=1}^{n}\sum_{j=1}^{L_k} \beta_j^{(k)} g_{k,j}\left(x^{(k-1)}; w_j^{(k-1)}, b_j^{(k-1)}\right) \quad (1)$$

where $S_n = \{L_1, L_2, \ldots, L_n\}$ represents the set of hidden node numbers in each layer, $g_{k,j}\left(x^{(k-1)}; w_j^{(k-1)}, b_j^{(k-1)}\right)$ and $\beta_j^{(k)}$ respectively represent the random basis functions and output weights of the j-th node within the k-th hidden layer, $w_j^{(k-1)}$ and $b_j^{(k-1)}$ are the input weights and biases of the j-th node within the k-th hidden layer, respectively, and $x(k) = [g_{k,1}, g_{k,2}, \ldots, g_{k,Lk}]$ and $x(0) = x$ represent the inputs of each hidden layer. When the n-th hidden layer already has L_n-1 nodes, the error is expressed as:

$$\begin{aligned} \varepsilon_{L_n-1}^{(n)} &= \varepsilon_{L_n-1}^{(n)}(X) = Y - F_{L_{n-1}}^{(n)} \\ &= \left[\varepsilon_{L_n-1}^{(n)}(x_1), \ldots, \varepsilon_{L_n-1}^{(n)}(x_N)\right]^{\mathrm{T}} \end{aligned} \quad (2)$$

Suppose that the span (Γ) is dense in the L_2 space, given $0 < r < 1$ and a nonnegative decreasing sequence $\{u_l\}$, where $u_l \leqslant (1 - r)$ and $\lim u_l = 0$ when n tends to ∞, the L_n-th node within n-th hidden layer must satisfy:

$$\theta_{L_n}^{(n)} = \frac{\left\langle \varepsilon_{L_n}^{(n)}, h_{L_n}^{(n)} \right\rangle^2}{\left\langle h_{L_n}^{(n)}, h_{L_n}^{(n)} \right\rangle^2} - (1-r-u)\left\| \varepsilon_{L_n-1}^{(n)} \right\|^2 \geqslant 0 \quad (3)$$

where $h_{Ln}^{(n)}$ represents the output value of this hidden node.

Under the constraint of the supervision mechanism shown in (3), a DSCN randomly configures its hidden nodes layer by layer in an incremental manner and then uses the global least-squares algorithm to update the readout weights as in (4). When the condition for stopping the training of the hidden layers is satisfied, the training process of the current hidden layer is completed. Then, the above configuration process is repeated for the hidden nodes until the termination condition of model training is reached.

$$\beta^* = \arg\min_{\beta} \left\| \sum_{k=1}^{n} \sum_{j=1}^{L_k} \beta_j^{(k)} g_{k.j} - Y \right\|^2 \tag{4}$$

2.2 Issues Faced by DSCNs

Although DSCNs can generate high-quality hidden nodes through their supervision mechanisms and have great advantages in terms of fast modeling, the following problems remain.

(1) The numbers of network layers and hidden nodes may be set unreasonably. When the number of network layers or hidden nodes is set too large, a large number of insignificance connections and hidden nodes which are irrelevant or redundant may be present in the DSCN, which will increase the computational and structural complexity of the network and cause overfitting. In contrast, the generalization ability of DSCNs is insufficient.

(2) The hidden node parameters are random. Since the weights and biases of the hidden nodes of a DSCN are randomly configured under the supervision mechanism, the nodes that are generated during the modeling process of the DSCN may contribute very little to the output. For example, the hidden nodes that cause a small decrease in the expected error after configuration, the nodes that are similar, and the connections between adjacent nodes that are insignificance also complicate the model structure.

3 MI-R-DSCN

To solve the problems of DSCNs, because deleting redundant hidden nodes can directly reduce the size of a DSCN and its parameters directly affect the subsequent parameter configuration process, mutual information is used to evaluate the relationships between the hidden nodes and target outputs in a layer wise manner to achieve structured pruning. In addition, considering the mutual influence between adjacent hidden layers, a pruning strategy based on weight relevance is proposed to realize unstructured pruning. Finally, the MI-R-DSCN algorithm is obtained.

3.1 Structured Pruning Based on Mutual Information

Pruning algorithms generally require pruning the hidden nodes that have little impact on performance from deep learning models[17], maintaining the original network performance as much as possible, and building a lightweight deep model. How to evaluate the importance of hidden nodes is the focus of pruning research. There is a difference between DSCNs and traditional neural networks, such as backpropagation neural networks. A DSCN only iterates once to determine the optimal network parameters, and the training of its next hidden layer is affected by the dataflow of the previous layer. If a DSCN is directly pruned from the perspective of global pruning after the training process is completed, thereby ignoring the influence of the data on the model during the training process[24], the generalization ability of the DSCN deteriorates.

Mutual information is often used to measure the degree of association between two random variables. Therefore, this section proposes a mutual information-DSCN (MI-DSCN) algorithm for layer-by-layer dynamic node pruning based on mutual information; this approach can evaluate the degree of association between the hidden nodes of each layer and the expected output in a layer-by-layer manner during the DSCN training process, calculate the level of correlation between the hidden nodes and the expected output, and then use the evaluation results as the judgment basis for pruning to realize local DSCN pruning. The MI-DSCN algorithm judges the correlation between the hidden nodes and the output by calculating the mutual information value between the

activation value of the nodes and the output variable, uses the correlation as the basis for judging the importance of the hidden nodes, and filters out the nodes in the hidden layer whose output values in the hidden are most suitable relative to the expected output value. Assuming that the number of training samples is N and the output variable is $Y = \{y_1, y_2, ..., y_N\}$, the activation matrix of the i-th hidden node within the n-th hidden layer can be expressed as:

$$h_i^{(n)} = \left[g_{n,i}\left(x_1^{(n-1)}\right), ..., g_{n,i}\left(x_N^{(n-1)}\right) \right]^{\mathrm{T}} \tag{5}$$

$H(h_i^{(n)})$ and $H(Y)$ are used to denote the activation entropy of the i-th hidden node within the n-th hidden layer and the information entropy of the output variable Y, respectively. The larger the information entropy is, the more information it contains. The mutual information $I_i^{(n)}$ between $h_I^{(n)}$ and variable Y, that is, the importance score $s_{n,j}$ of the hidden node, can be expressed as:

$$s_{n,i} = I_i^{(n)} \tag{6}$$

$$I_i^{(n)} = H\left(h_i^{(n)}\right) + H(Y) - H\left(h_i^{(n)}, Y\right) \tag{7}$$

where $H(h_i^{(n)}, Y)$ represents the joint entropy of vector $h_i^{(n)}$ and variable Y. When the mutual information value $I_i^{(n)}$ is 0, it means that the hidden node and the expected output are uncorrelated or independent of each other, having little effect on the construction of the DSCN. In contrast, the larger $I_i^{(n)}$ is, the higher the correlation between this node and the expected output, the more intersecting information, and the more important role this node plays in the model. The value of the vector $h_i^{(n)}$ is divided into D_h intervals, and the value of the vector Y is divided into D_y intervals. Then, $p_{i,m}^{(n)}$ and p_j represent the probabilities that the values of vectors $h_i^{(n)}$ and Y are distributed in intervals m and j, respectively. The calculation formulas are expressed as:

$$H\left(h_i^{(n)}\right) = -\sum_{m=1}^{D_h} p_{i,m}^{(n)} \log_2\left(p_{i,m}^{(n)}\right) \tag{8}$$

$$H(Y) = -\sum_{j=1}^{D_y} p_j \log_2\left(p_j\right) \tag{9}$$

$$H\left(h_i^{(n)}, Y\right) = -\sum_{g \in D_h} \sum_{y \in D_y} p(g,y) \log_2 p(g,y) \tag{10}$$

where $p_{i,m}^{(n)} \geqslant 0$ and $p_j \geqslant 0$. The larger $s_{n,j}$ is, the greater the contribution of the corresponding hidden node in the node set of the n-th hidden layer. Therefore, after sorting the nodes of the n-th layer according to their node importance scores, we obtain:

$$s_{n,1}' \geqslant s_{n,2}' \geqslant \cdots \geqslant s_{n,L_n}' \tag{11}$$

Generally, pruning algorithms require a large degree of pruning at the beginning of the iterative process and a lower pruning rate in the later stage of iteration. This is because a neural network based on the gradient descent algorithm searches for the global optimal solution through iterations. In the later stage, the performance of the learning model tends to be stable and is no longer suitable for large-scale pruning. However, the DSCN does not need to iteratively find the optimal solution, so it needs to maintain a low pruning rate at the beginning of training to ensure that the DSCN model is built quickly. In the middle of training, the DSCN can delete redundant nodes from the model through large-scale pruning. Although large-scale pruning causes the accuracy of the DSCN to decrease, the accuracy will be restored when the hidden layers are constructed later. At the end of training, the structure of the DSCN has been basically determined, so it is necessary to set a low pruning rate. The node pruning rate threshold for the hidden layers is defined as a, and the pruning rate within the k-th hidden layer of the DSCN is $pr(k)$ by (12). When the learning coefficient c is $2a/(M\text{-}1)^2$, the curve of the function $pr(k)$ is shown in Fig.2. It can be seen that the node pruning rate of the hidden layers increases first and then decreases with the depth of the network so that the network can achieve low pruning in the early and late stages and high pruning in the middle.

$$pr(k) = -c \cdot \left(k - \frac{M+1}{2}\right)^2 + a, k = 1, 2, ...M \tag{12}$$

where $0 \leqslant a < 1$, $0 \leqslant c \leqslant 4a/(M-1)^2$, and M is the maximum number of hidden layers. When a=0 and c=0 are satisfied, the MI-DSCN algorithm degenerates to the DSCN algorithm. After sorting and pruning the hidden nodes within the n-th hidden layer by (11) and (12), the number of nodes remaining in this layer can

be expressed as:

$$num(n) = round\left(L_n' * (1 - pr(n))\right) \tag{13}$$

where L_n' represents the number of hidden nodes in the n-th hidden layer after the configuration process is completed, and $pr(n)$ represents the hidden node pruning rate within the n-th hidden layer. When the dynamic layer-by-layer DSCN pruning procedure is completed, the global hidden node pruning rate Δp of the DSCN is calculated:

$$\Delta p = \frac{\sum_{i=1}^{n} L_i' - \sum_{j=1}^{n} num(j)}{\sum_{i=1}^{n} L_i} \times 100\% \tag{14}$$

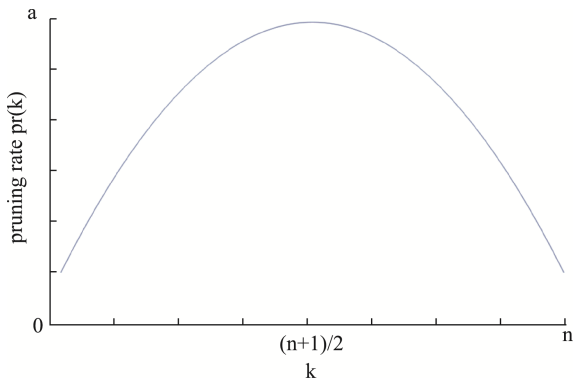

Fig.2 Curve of the function *pr*(*k*)

In summary, the process of the pruning part of the MI-DSCN algorithm is as follows.

Step 1: Evaluate the importance of the hidden nodes. Calculate the mutual information values between the activation values of the hidden nodes within the k-th hidden layer and the expected output by (6) and (7), and use these values as the evaluation standard of hidden node importance.

Step 2: Sort the hidden nodes. Sort the hidden nodes within the k-th layer from large to small by (11).

Step 3: Delete the unimportant hidden nodes and their connection weights. Sort the nodes within the k-th layer according to the importance scores calculated in *Step* 1, only keep the first $num(k)$ nodes after sorting, prune the related parameters of the remaining nodes from the original DSCN structure, and update the DSCN parameters.

Step 4: Determine whether the model training process terminates. If the conditions for terminating the training of the DSCN are not met after pruning, construct nodes within the (k+1)-th layer, and repeat *Step* 1-3 for the (k+1)-th layer. Otherwise, the pruning operation is completed.

3.2 Unstructured Pruning Based on Weight Relevance

Unstructured pruning algorithms usually regard a weight whose absolute value is greater than the weight pruning threshold as an important weight. If the norm value of the weight is used as the only criterion for weight pruning and the influence of the weights on the connections between adjacent hidden layer nodes is ignored, this may cause the important weights that have direct impacts on the network parameters of the next hidden layer to be misjudged as unimportant weights and incorrectly deleted. Therefore, we propose a method for judging the importance of connections based on the relevance strategy, where a node considers its own weight and its impact on the weights of the next hidden layer of connections. The connection importance scores of the weights can be expressed as:

$$\begin{cases} \vartheta_{i,j}^{(k)} = \left|w_{i,j}^{(k)}\right| + \dfrac{1}{L_{k+1}''} \sum_{m=1}^{L_{k+1}} \left|w_{i,m}^{(k+1)}\right|, \ k = 0,\ 1,\ ...,\ n-2 \\ \vartheta_{i,j}^{(k)} = \left|w_{i,j}^{(k)}\right|, \ k = n-1 \end{cases} \tag{15}$$

where $w_{i,j}^{(k)}$ represents the connection weights between the i-th node within the k-th hidden layer and the j-th node within the (k+1)-th hidden layer, $w_{i,j}^{(0)}$ represents the connection weights between the input layer and the first hidden layer, and L_{k+1}'' indicates the number of existing hidden nodes in the (k+1)-th hidden layer. The smaller $\vartheta_{i,j}^{(k)}$ is, the less important the connection weight is, and the associated node is listed as the priority pruning object. Therefore, after sorting the global connection weights according to their connection importance scores, we obtain:

$$\vartheta_{1,1}^{(0)\prime} \geqslant \cdots \geqslant \vartheta_{i,j}^{(k)\prime} \geqslant \cdots \geqslant \vartheta_{L_{n-1},L_n}^{(n-1)\prime} \tag{16}$$

In summary, the pruning strategy based on weight relevance includes the following steps.

Step 1: Evaluate the importance of the connection weights between the hidden nodes. Calculate the importance score of the connection weight between each pair of hidden nodes by (15).

Step 2: Sort the connection weights. Sort the global connection weights from large to small by (16).

Step 3: Prune the weights. Set the global weight pruning ratio b, reset the weights to 0, convert the high-dimensional sparse weight matrix into a compressed sparse matrix, update the parameters of the DSCN, and complete the pruning operation.

3.3 MI-R-DSCN Algorithm

A structural pruning strategy can reduce the required modeling time but may result in a loss of accuracy. The unstructured pruning strategy is the opposite. Based on the above analysis, this section further proposes a double pruning MI-R-DSCN algorithm on the basis of the two pruning methods proposed in the previous sections to achieve maximum DSCN compression. The pseudo code of the MI-R-DSCN algor- ithm is as follows.

Algorithm 1: Mutual information-relevance-deep stochastic configuration networks

Input: Dataset $\{X, Y\}$; The maximum number of hidden layers, M; The maximum number of hidden nodes within the n-th hidden layer,$L_{max}^{(n)}$; The expected error tolerance, ϵ; The maximum times of random configuration, T_{max}; The hidden node parameter configuration range, $[-\lambda, \lambda]$

Output: The model of MI-R-DSCN

Initialization: $\epsilon_0^{(1)} = [t_1, ..., t_N]$, $\mathcal{H}$,Ω,$W := [\,]$;
while $n \leq M$ *and* $\|\epsilon_0^{(1)}\|_F > \epsilon$ **do**
  while $L_n \leq L_{max}^{(n)}$ *and* $\|\epsilon_0^{(1)}\|_F > \epsilon$ **do**
    Randomly configure the group candidate hidden nodes parameters from $[-\lambda, \lambda]^T$ and $[-\lambda, \lambda]$;
    Calculate the activation matrix h of each candidate node by (5);
    Save the candidate node parameters $w_{L_n}^{(n-1)}$ and $b_{L_n}^{(n-1)}$ satisfying (3) in W, and $\theta_{L_n}^{(n-1)}$ in Ω;
    if W *is not empty* **then**
      Find $w_{L_n}^{(n-1)*}$ and $b_{L_n}^{(n-1)*}$ maximizing $\theta_{L_n}^{(n-1)}$, and update $H_{L_n}^{(n)} = [h_1^{(n)*}, ..., h_N^{(n)*}]$;
    else
      Continue: go to step 4;
    end
    Set $\mathcal{H} = [\mathcal{H}, H_{L_n}^{(n)}]$, calculate $\beta^* = \mathcal{H}^\dagger Y$, $\epsilon_{L_n}^{(n)} = \mathcal{H}\beta^* - Y$,$\epsilon^{(n)} = \epsilon_{L_n}^{(n)}$, and update $\epsilon_0^{(1)} = \epsilon_{L_n}^{(n)}$, $L_n := L_n + 1$;
  end
  Pruning 1: Layer-by-layer dynamic hidden node pruning based on mutual information (Step 15 - Step 17)
  Calculate the important score $s_i^{(n)}$ of the hidden nodes by (6) and (7), and then sort the hidden nodes by (11);
  Calculate the pruning rate $pr(n)$ and $num(n)$ by (12) and (13), and only keep the first $num(n)$ hidden nodes parameters;
  Update $\mathcal{H}$, W, β^*, $\epsilon_{L_n}^{(n)}$ and $\epsilon_0^{(1)}$;
end
Pruning 2: Global weight pruning based on relevance (Step 20 - Step 23)
Calculate the important score ϑ of the connection weights by (15) and sort the connection weights (16);
Set the global weight pruning ratio b, and set the connection weights listed as the pruning objects to 0;
Update W and convert it to a sparse matrix;
Update $\mathcal{H}$, β^*, and $\epsilon_{L_n}^{(n)}$;
Return: A MI-R-DSCN model

4 Experiment and Result Analysis

To verify the effectiveness of the MI-R-DSCN algorithm, function approximation and standard regression datasets are selected as the experimental contents. For convenience, the DSCN algorithm based on mutual information and layer-by-layer dynamic single pruning is abbreviated as MI-DSCN, and the layer-by-layer DSCN pruning algorithm based on mutual information with a fixed pruning rate is abbreviated as MDSCN. The experiments are all completed in an environment with Windows 10 and MATLAB R2016a. The hardware configuration is an Intel(R) Core (TM) i5-9500 CPU @ 3.00 GHZ, and the memory is 8 GB.

4.1 Function Approximation

The following function[25] is selected to compare the approximation abilities of the improved DSCN algorithm and the typical algorithm.

$$\begin{cases} f(x)=0.2e^{-(10x-4)^2}+0.5e^{-(80x-40)^2}+0.3e^{-(80x-20)^2} \\ 0\leqslant x\leqslant 1 \end{cases} \tag{17}$$

Starting with a uniform distribution, 600 points are randomly selected as the training dataset and test training dataset. The parameter settings of the MI-R-DSCN, DSCN, MI-DSCN and MDSCN algorithms are shown in Table 2. These models select the sigmoid function as their activation functions, and all parameters are obtained after conducting a validation set test. Commonly used sparse matrix storage formats include coordinate format (COO), compressed sparse row format (CSR), compre- ssed sparse column format (CSC), etc. Because CSR can quickly perform matrix-vector product operations and has efficient sparse matrix computing capabilities, this experiment uses the CSR format to store sparse matrices. The mean absolute error (MAE) and root mean square error (RMSE) are selected as the evaluation indicators of the algorithms.

$$g(x)=sigmoid(x)=\frac{1}{1+e^{-x}} \tag{18}$$

This experiment is mainly used to compare the effects of different pruning strategies and hidden node pruning rates on the performance of a DSCN. Under different hidden node pruning rate thresholds, the global node pruning rates Δp of the MI-R-DSCN model with double pruning are shown in Table 2, and they are correspondingly used as the values of the fixed pruning rates for the MDSCN. The regression results of the double-pruned MI-R-DSCN, DSCN, single-pruned MI-DSCN, and fixed-pruned MDSCN algorithms, as well as the comparison results of these models in terms of their global hidden node pruning rates Δp, are shown in Fig.3. The threshold a of the hidden node pruning rate in Fig.3 is 0, which means that the deep learning model is not pruned at this time. When the values of the node pruning rate and weight pruning rate are both 0, the MI-DSCN degenerates to the DSCN algorithm. Each model is independently run 50 times, and the mean of the evaluation indicators is used as a standard for evaluating the performance of these deep models.

In Fig.3, as the threshold a of the hidden node pruning rate becomes larger, the RMSEs and MAEs of the MI-R-DSCN, MI-DSCN and MDSCN become larger, and the test times change obviously first and then slowly, which shows that pruning can speed up the DSCN, but over pruning leads to poor performance and a poor generalization ability for the DSCN. When the node pruning rate threshold a is at a low value, the RMSEs and MAEs of the MI-R-DSCN, MI-DSCN and DSCN are similar. The RMSE and MAE of MDSCN are the largest, which shows that although the layer-by-layer pruning strategy with a fixed pruning rate can effectively compress the DSCN, it seriously affects the fitting accuracy of the DSCN. However, when the threshold a is set reasonably, the layer-by-layer dynamic node pruning strategy based on mutual information can also compress the DSCN and has a lesser effect on the fitting accuracy of the DSCN.

Table 1 Parameter Settings for the Function Approximation Experiment

Parameter Settings	MI-R-DSCN	DSCN	MI-DSCN	MDSCN
Maximum Number of Hidden Layers M	3	3	3	3
Maximum Number of Hidden Nodes L_{max}	50	50	50	50
Maximum Times of Random Configuration T_{max}	200	200	200	200
Threshold for Node Pruning Rate a	{0, 0.05, ...,0.95}	—	{0, 0.05, ...,0.95}	—
Proportion of Global Weight Pruning b	30%	—	30%	—
Learning Coefficient c	0.75a	—	0.75a	—
Expected Error of Training ε	10^{-4}	10^{-4}	10^{-4}	10^{-4}

Table 2 Global Node Pruning Rates Δp of the MI-R-DSCN Under Different Hidden Node Pruning Rate Thresholds

Threshold a	MI-R-DSCN Global Node Pruning Rate Δp	Threshold a	MI-R-DSCN Global Node Pruning Rate Δp
0	0	0.50	35.21%
0.05	2.45%	0.55	36.54%
0.10	5.67%	0.60	40.97%
0.15	8.97%	0.65	44.79%
0.20	13.69%	0.70	48.70%
0.25	16.80%	0.75	52.52%
0.30	20.42%	0.80	57.80%
0.35	23.74%	0.85	60.18%
0.40	27.12%	0.90	65.43%
0.45	30.86%	0.95	69.52%

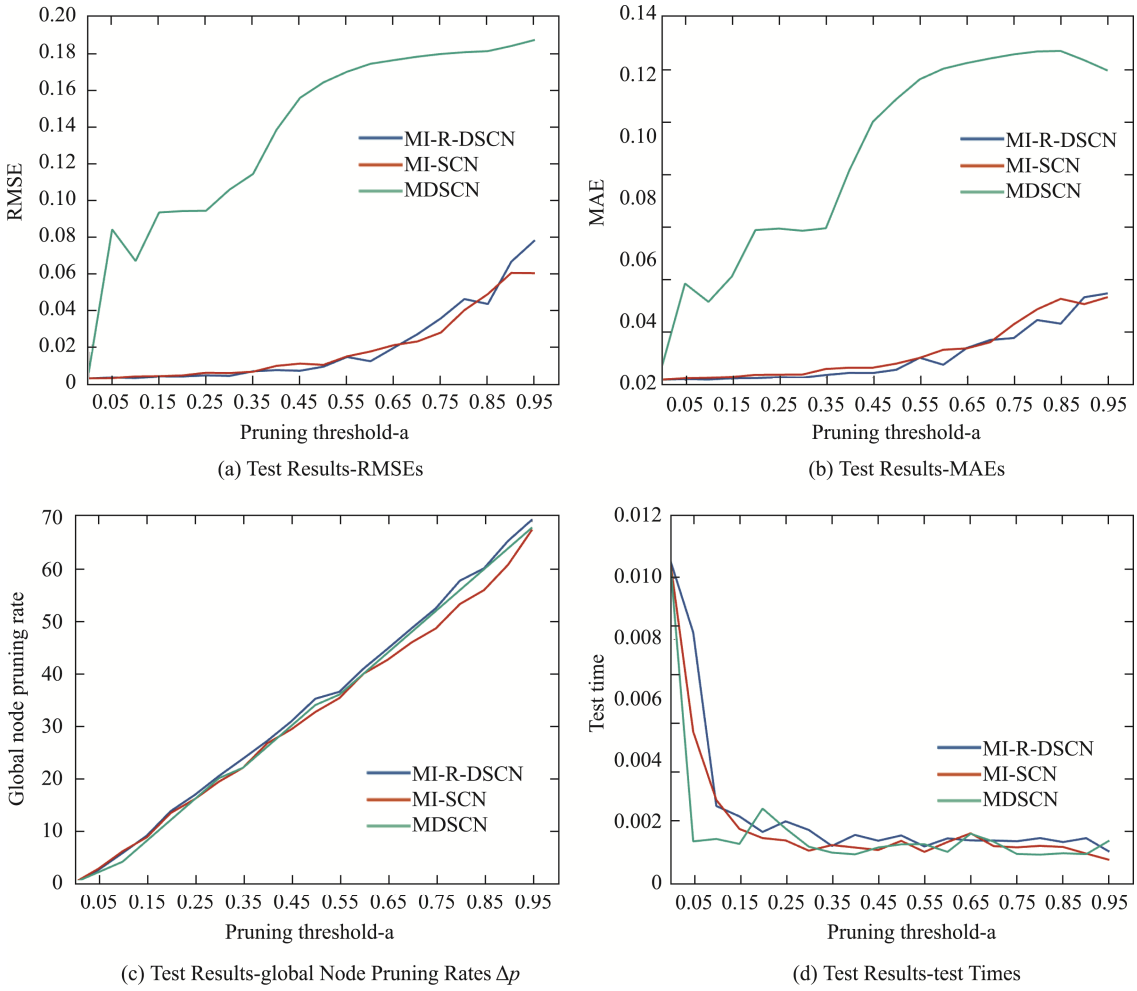

Fig.3 Comparison of the Results Obtained by the MI-R-DSCN, DSCN, MI-DSCN and MDSCN Under Different Thresholds (The Three Improved Algorithms Based on the DSCN Degenerate to the DSCN Algorithm When the Threshold a = 0)

Table 3 and Fig.4 show the comparison results regarding the fitting conditions of the nonlinear function produced by each method when the threshold a of the hidden node pruning rate is 0.35 so that the fitting conditions of the four methods can be observed more intuitively. From them, when $a = 0.35$, all three pruning methods based on the DSCN can speed up the modeling process, but the RMSE and MAE of the MDSCN are the largest. The RMSEs and MAEs of the MI-R-DSCN and MI-DSCN are similar to those of the DSCN. The RMSE and MAE of the MI-R-DSCN are only 0.0006 and 0.0019 higher than those of the DSCN and 0.0001 and 0.0003 higher than those of the MI-DSCN, respectively. The global node pruning rates Δp of the MI-R-DSCN and MI-DSCN differ by only 1.73%, but 30% of the connection weights in the MI-R-DSCN model are assigned 0, which indicates that the MI-R-DSCN node parameters are fewer than those of the MI-DSCN. In the case where the MI-R-DSCN has a larger compression scale than the MI-DSCN, their accuracy losses are similar, which further leads to the conclusion that the MI-R-DSCN performs better in terms of DSCN compression.

Table 3 Comparison of the Results of the MI-R-DSCN, DSCN, MI-DSCN and MDSCN When the Threshold $a = 0.35$

Evaluation indicator	MI-R-DSCN	DSCN	MI-DSCN	MDSCN
RMSE	0.0024 ± 0.0005	**0.0018 ± 0.0001**	0.0023 ± 0.0008	0.0558 ± 0.0014
MAE	0.0030 ± 0.0009	**0.0011 ± 0.0002**	0.0027 ± 0.0011	0.0252 ± 0.0017
Test Time	**0.0012 ± 0.0003**	0.0106 ± 0.0011	0.0032 ± 0.0001	0.0036 ± 0.0000
Global Node Pruning Rate Δp	**23.74% ± 0.0008**	0.00% ± 0.0011	22.01% ± 0.0007	22.02% ± 0.0013

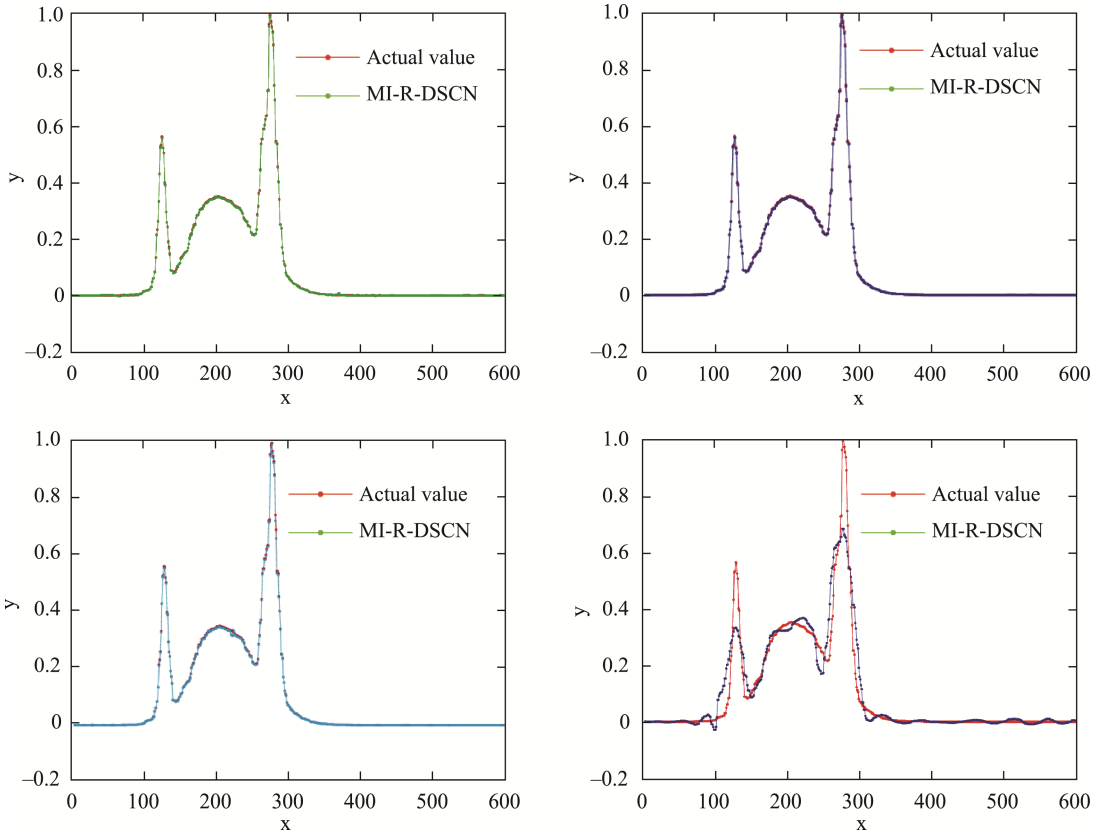

Fig.4 Comparison Among the MI-R-DSCN, DSCN, MI-DSCN and MDSCN Algorithms When $a = 0.35$

4.2 Standard Datasets

4.2.1 Dataset Preparation and Parameter Setting

Dataset preparation stage: The experimental datasets are selected from the Elevators, Computer Activity, Treasury and California Housing regression datasets in the "Knowledge extraction based on evolutionary learning" standard datasets. For the convenience of the subsequent descriptions, the four aforementioned datasets are marked as D1-D4, and the basic information about the datasets is shown in Table 4. A total of 1000 samples are randomly selected from each dataset, 80% of which are used as a training dataset and 20% of which are used as a test dataset.

Algorithm parameter setting stage: The parameter settings of the DSCN, MI-DSCN, MDSCN, BP and MI-R-DSCN algorithms are shown in Table 5. All deep learning models select the sigmoid function as the activation function. The BP neural network selects the Levenberg-Marquardt function as its training function, and its number of iterations is set to 1000. Similarly, the experiments use the CSR format to store sparse matrices.

The compression ratio comp of the pruned model is expressed as:

$$comp = \Delta p + (1 - \Delta p) \times b \tag{20}$$

where the DSCN, MDSCN, and BP network cannot contain the weight pruning part, so they all use 0 as the parameter b when calculating $comp$. The physical dimensionality of each feature variable is different, so it is necessary to normalize the input variables and output variables to eliminate the influence of the dimensionality on these deep learning models. The RMSE, MAE, determination coefficient (R^2), test time and compress- ion ratio $comp$ are selected as performance evaluation indicators.

4.2.2 Analysis

To further verify the effectiveness of the MI-R-DSCN algorithm proposed in this paper in the regression subfield, we compare the MI-R-DSCN with the SCN, MI-DSCN, MDSCN and BP network. Each model is independently run 50 times, and the average values of the evaluation metrics are used to evaluate these models' performance.

When the node pruning rate threshold a of the MI-R-DSCN and MI-DSCN is set to 0.45, the global weight pruning rate b of the MI-R-DSCN is 0.5, and the global hidden node pruning rate Δp of the MI-R-DSCN model is the same as the value of the fixed pruning rate of the MDSCN, the regression comparison results obtained by each algorithm on the standard regression datasets are shown in Fig.5 and Table 6. Comparing the

Table 4 Basic Information of the Standard Regression Datasets

Label	Dataset Name	Number of Samples	Number of Features
D1	Elevators	16599	18
D2	Computer Activity	8192	21
D3	Treasury	1049	15
D4	California Housing	20460	8

Table 5 Parameter Settings for Regression Experiments

Parameter settings	DSCN	MI-DSCN	MDSCN	BP	MI-R-DSCN
Maximum Number of Hidden Layers M	6	6	6	6	6
Maximum Number of Hidden Nodes L_{max}	100	100	100	100	100
Maximum Times of Random Configuration T_{max}	200	200	—	—	200
Threshold for Node Pruning Rate a	—	0.45	—	—	0.45
Proportion of Global Weight Pruning b	—	50%	—	—	50%
Learning Coefficient c	—	0.054	—	—	0.054
Expected Error of Training ε	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}

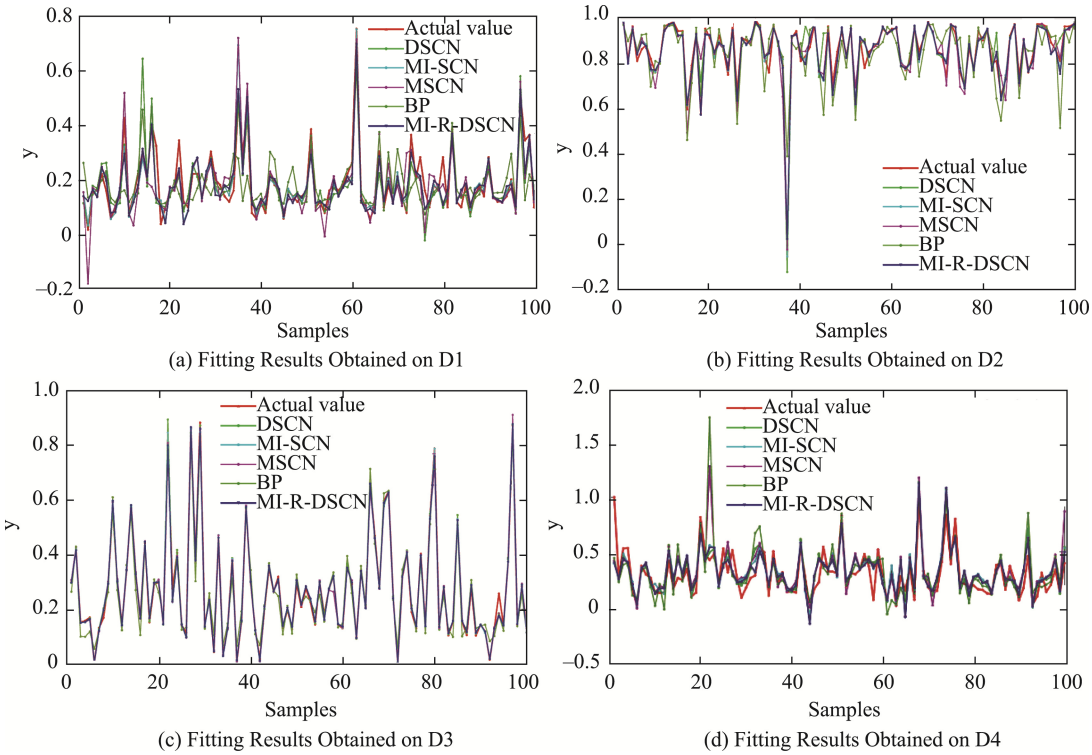

Fig.5 Comparison of the Fitting Results of Algorithms on Standard Datasets

Table 6 Comparison Among the Evaluation Criteria Yielded by the DSCN, MI-DSCN, MDSCN, BP, and MI-R-DSCN Algorithms

Dataset Label	Evaluation Indicators	Algorithm				
		DSCN	MI-DSCN	MDSCN	BP	MI-R-DSCN
D1	RMSE	0.0582	**0.0506**	0.0676	0.0870	0.0513
	MAE	0.0380	**0.0302**	0.0447	0.0634	0.0312
	R^2	0.7490	**0.8227**	0.6613	0.4379	0.8118
	Test Time *t*/s	0.0139	0.0065	0.0042	0.0730	**0.0018**
	Compression Ratio *comp*	0.00%	32.17%	34.75%	0.00%	**67.38%**
D2	RMSE	0.0423	**0.0385**	0.0438	0.0816	0.0396
	MAE	0.0181	**0.0176**	0.0179	0.0547	0.0177
	R^2	0.9602	**0.9620**	0.8835	0.5948	0.9571
	Test Time *t*/s	0.0129	0.0052	0.0046	0.0524	**0.0024**
	Compression Ratio *comp*	0.00%	33.84%	31.61%	0.00%	**67.37%**
D3	RMSE	0.0153	0.0143	0.0147	0.0342	**0.0126**
	MAE	0.0076	0.0087	0.0093	0.0289	**0.0072**
	R^2	0.9939	0.9949	0.9933	0.9692	**0.9958**
	Test Time *t*/s	0.0125	0.0043	0.0056	0.0416	**0.0029**
	Compression Ratio *comp*	0.00%	30.00%	33.96%	0.00%	**65.3%**
D4	RMSE	0.1187	0.1340	0.1581	0.1355	**0.1181**
	MAE	0.0899	0.0978	0.1062	0.1080	**0.0896**
	R2	0.6644	0.6588	0.4048	0.5632	**0.6679**
	Test Time *t*/s	0.0116	0.0015	0.0074	0.0210	**0.0010**
	Compression Ratio *comp*	0.00%	31.04%	34.13%	0.00%	**67.20%**

regression results of D1 and D2, it can be seen that the fitting RMSE, MAE, and R^2 of the MI-DSCN are the smallest, while the RMSE, MAE, and R^2 of the MI-R-DSCN are similar to those of the MI-R-DSCN; the compression ratios *comp* of the MI-R-DSCN are 34.67% and 33.53% higher than those of the MI-DSCN, and its test times are 72.31% and 53.85% shorter than those of the MI-DSCN on D1 and D2, respectively, indicating that the performance of the double pruning strategy is better than that of the single-pruning strategy. Comparing the fitting results obtained on datasets D3 and D4, the evaluation indicators of the MI-R-DSCN are the best, and the evaluation indicators of BP are the worst. In summary, when the parameters are set reasonably, the MI-R-DSCN can fit standard regression datasets well, and it does not cause a significant decrease in the fitting ability of the DSCN. Moreover, the MI-R-DSCN can also alleviate the overfitting phenomenon of the DSCN caused by unreasonable settings of the numbers of hidden layers and hidden nodes.

5 Conclusion

To solve the problem that DSCNs generate redundant hidden nodes and insignificance connections that can increase the network complexity, this paper proposes a double pruning MI-R-DSCN algorithm based on mutual information and correlation. The main contributions of this paper are as follows.

First, a global weight pruning strategy based on relevance is proposed. The network weights and the influences between implicit nodes in adjacent layers are used as conditions for judging the importance levels of the connection weights, and the unimportant weights are reset to zero to realize the sparseness of DSCNs.

Second, a double pruning method named the MI-R-DSCN is proposed based on mutual information and relevance. By dynamically obtaining the node pruning rate in a layer-by-layer manner, this method solves the problems of setting the same pruning rate for each hidden layer and ignoring the different amounts of information contained in each layer. In addition, the double pruning strategy can accurately screen and delete unimportant parameters in the DSCN, and the loss of model accuracy is small. This strategy solves the problem of overfitting caused by unreasonable hyperparameter settings and realizes compression and acceleration for the DSCN model.

Although the MI-R-DSCN can reduce the required computation time, this method causes an excessive model accuracy loss when the pruning ratio is set unreasonably. In future research, we will consider incorporating the improved swarm intelligence algorithm into the MI-R-DSCN algorithm so that the population can find the best pruning ratio during the optimization process and further realize a lighter DSCN structure.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62073006) and the Beijing Natural Science Foundation of China (4212032).

References

[1] Daniya T, Vigneshwari S (2022). Deep neural network for diseased detection in rice plant using the texture and deep Features. *The Computer Journal.* 65(7). pp.1812-1825.

[2] Jiang W, Wang Z, Jin J S, et al (2019). Speech emotion recognition with heterogeneous feature unification of deep neural network. *Sensors*. 19(20). pp.2730-2744.

[3] Wen X, Xu Z (2021). Wind turbine fault diagnosis based on ReliefF-PCA and DNN. *Expert Systems with Applications*. 178(11). pp.115016.

[4] Eldan R, Shamir O (2016). The power of depth for feedforward neural networks. *The Proceedings of 29th Conference on Learning Theory*, June 23-26, New York, USA.

[5] Telgarsky M (2016). Benefits of depth in neural networks. *Conference on Learning Theory*. 49(6).pp. 1517-1539.

[6] Nguyen Q, Hein M (2018). Optimization landscape and expressivity of deep CNNs. *The Proceedings of 35th International Conference on Machine Learning*, July 10-15, Stockholmsmässan, Stockholm Sweden.

[7] Gong R H, Liu X L, Jiang S H, et al (2019). Differentiable soft quantization: Bridging full- precision and low-bit neural networks. *The Proceedings of IEEE/CVF International Conference on Computer Vision*, October 27 to November 02, Seoul, Korea (South).

[8] Rastegari M, Ordonez V, Redmon J, et al (2016). XNOR-net: ImageNet classification using binary convolutional neural networks. *The Proceedings of 14th European Conference on Computer Vision*, October 11–14, Amsterdam, the Netherlands.

[9] Han S, Mao H, Dally W J (2016). Deep compression: Compressing deep neural networks with pruning, trained

quantization and huffman coding. *The Proceedings of 4th International Conference on Learning Representations*, May 2-4, San Juan, Puerto Rico.

[10] Tai C, Xiao T, Zhang Y, et al (2016). Convolutional neural networks with low-rank regularization. *The Proceedings of International Conference on Learn- ing Representations*, May 2-4, San Juan, Puerto Rico.

[11] Yu X Y, Liu T L, Wang X C, Tao D C(2017). On compressing deep models by low rank and sparse decomposition. *The Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, July 21-26, Honolulu, USA.

[12] Hinton G, Vinyals O, Dean J (2015). Distilling the Knowledge in a Neural Network. *Computer Scienc.* 14(7). pp.38-39.

[13] Yim J, Joo D, Bae J, Kim J (2017). A gift from knowledge distillation: fast Optimization, network minimization and transfer learning. *The Proceed- ings of IEEE Conference on Computer Vision and Pattern Recognition*, July 21-26, Honolulu, USA.

[14] Li X, Fan D P, Yang F, et al (2021). Probabilistic model distillation for semantic correspondence. *The Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 20- 25, Nashville, USA.

[15] Carreira-Perpinan M A, Idelbayev Y (2018). "Learning-Compression" algorithms for neural net pruning. *The Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 18-23, Salt Lake City, USA.

[16] Dong J, Zheng H, Lian L (2017). Activation-based weight significance criterion for pruning deep neural networks. *The Proceedings of 9th International Conference on Image and Graphics*, September 13-15, Shanghai, China.

[17] Qing T, Tal A, James J C (2017). Deep LDA- pruned nets for efficient facial gender classification. *The Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, July 21-26, Honolulu, USA.

[18] Luo J H, Wu J, Lin W (2017). ThiNet: A filter level pruning method for deep neural network compression. *The Proceedings of IEEE Interna- tional Conference on Computer Vision*, October 22-29, Venice, Italy.

[19] Wang D, Li M (2018). Deep stochastic configur- ation networks with universal approximation property. *The Proceedings of 2018 International Joint Conference on Neural Networks*, July 8-13, Rio de Janeiro, Brazil.

[20] Igelnik B, Pao Y H (1995). Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*. 6(6). pp.1320-1329.

[21] Wang D, Ming L (2017). Stochastic configuration networks: fundamentals and algorithms. *IEEE Transactions on Cybernetics*, 47(10). pp.3466-3479.

[22] Lu J, Ding J L (2019). Construction of prediction intervals for carbon residual of crude oil based on deep stochastic configuration networks. *Information Sciences*. 486(17). pp.119-132.

[23] Pratama M, Wang D H (2019). Deep stacked stochastic configuration networks for lifelong learning of non-stationary data streams. *Information Sciences*. 495(25). pp.150-174.

[24] Guo J C, Yan A J (2021). Robust deep stochastic configuration network modeling method based on kernel density estimation. *The Proceedings of 2021 33rd Chinese Control and Decision Conference*, May 22-24, Kunming, China.

[25] Tyukin I Y, Prokhorov D V (2009). Feasibility of random basis function approximators for modeling and control. *The Proceedings of IEEE Control Applications & Intelligent Control*, July 08-10, St. Petersburg, Russia.

Author Biographies

YAN Aijun received his Ph.D. degree from North- eastern University in 2006. He is currently a professor in the Beijing University of Technology. His research interest includes artificial intelligence, intelligence optimization algorithm, process modeling and control.

E-mail: yanaijun@bjut.edu.cn

LI Jiale is currently a M.Sc. candidate at the Faculty of Information Technology, Beijing University of Technology. Her research interest includes intelligence optimization algorithm and its applications.

E-mail: LiJiaL@emails.bjut.edu.cn

TANG Jian received the Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China in 2012. He is currently a Professor with the Faculty of Information Technology, Beijing University of Technology. His main research interests include machine learning based on small sample data, intelligent modeling and control of complex industrial process, and digital twin system of municipal solid waste incineration process.

E-mail: LiJiaL@emails.bjut.edu.cn