Received 26 February 2024, accepted 16 March 2024, date of publication 27 March 2024, date of current version 3 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3382195

RESEARCH ARTICLE

Deep Neural Network Optimization Based on Binary Method for Handling Multi-Class Problems

YUQI LIU[1], SIBO YANG[2], AND YUAN BAO[3]
[1]Marine Engineering College, Dalian Maritime University, Dalian 116026, China
[2]School of Science, Dalian Maritime University, Dalian 116026, China
[3]School of Mathematics and Statistics, Xinyang Normal University, Xinyang 464000, China

Corresponding author: Yuan Bao (baoyuany@outlook.com)

This work was supported by the National Natural Science Foundation of China under Grant 61720106005.

ABSTRACT In this paper, we conceive a new kind of output layer design in deep neural networks for the multi-class problems. The traditional output layer is set by the one-to-one method. For the one-to-one method, the output layer neuron number is the same as the class number. And the ideal output for the j-th class sample is e_j, where e_j is j-th unit vector. However, one-to-one method requires too many output neurons, which will increase the number of weights connecting the last-hidden and the output layers. Furthermore, during the process of network training, computation time and cost will greatly increase. We design the binary method for the output layer: Let the class number be k ($k \geq 3$), and $2^{a-1} < k \leq 2^a$ ($a = \lceil log_2 k \rceil$), then the output layer neuron number is a and the ideal output is designed by binary method. Obviously, the binary method uses less output nodes than the traditional one-to-one method. On this foundation, the number of hidden-output weights will also decrease. On the other hand, while training the deep neural network, the learning efficiency will also be significantly improved. Numerical experiments show that binary method has better classification performance and calculation speed than one-to-one method on the datasets.

INDEX TERMS Deep neural networks, one-to-one method, binary method, multi-class problems.

I. INTRODUCTION

Recently, deep neural networks (DNNs) [1], [2], [3] are increasingly employed in various fields due to the strong self-learning ability. As two important research directions in DNNs, learning efficiency [4], [5] and structure optimization [6], [7] have attracted many scholars. Learning efficiency is mainly to select better algorithms to achieve better learning performance, such as classification accuracy; Structural optimization refers to using as few neurons and weights as possible in the network to achieve the same learning efficiency. Generally speaking, structural optimization can be achieved from three aspects: input, hidden and output layers. For the input layer, feature selection [8], [9], [10], [11] is applied to select some of the most effective features from the original features to reduce the dimension of the dataset. For the hidden layer, introducing regularization terms [12], [13], [14] into the learning procedure has been shown to be efficient to improve the generalization performance as well as decrease the number of the network weights.

For multi-class problems, in addition to the one-to-one(OTO) method, other commonly used methods include one-versus-one (OVO), one-versus-all (OVA), and error-correcting output coding (ECOC) [15]. In the OVO method, every two classes are alternately selected to perform a binary problem. Therefore, a total of $k(k-1)/2$ binary classification tasks are needed. For the OVA method, the k-class problem is transformed into k binary problems, requiring a total of k binary classification tasks. ECOC creates an encoding matrix M and transforms the k-class problem into several binary classification problems based on the matrix; Then calculate the hamming distance, and the class with the smallest distance is the predicted class. Researchers can control the dimension of bit encoding.

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

TABLE 1. Comparison of the number of binary classifiers required for five methods.

Algorithms	Number of binary classifiers
OVO	$k(k-1)/2$
OVA	k
ECOC	-
OTO	k
Binary	$\lceil log_2 k \rceil$

If the dimension of the bit encoding is greater than k, the prediction results of some classifiers can be corrected by other classifiers. That is to say, binary classifiers with more than k can achieve error correction ability. Looking at the three output layer methods introduced above, excessive binary classifiers are required, which will directly lead to high computational costs in the training and testing processes.

In this article, we mainly pay attention to the structure optimization of the output layer in DNNs. The customary output layer design is one-to-one method [16], [17], [18]: For the k-class problem, the output neuron number is k; The ideal output for the j-th class is e_j, where e_j is the j-th unit vector in $\mathbb{R}^k$. For instance, for a five-class problem, the ideal outputs of these five classes are $(1, 0, 0, 0, 0)$, $(0, 1, 0, 0, 0)$, $(0, 0, 1, 0, 0)$, $(0, 0, 0, 1, 0)$, $(0, 0, 0, 0, 1)$. Apparently, too many output neurons and hidden-output weights are required in this method. The root reason is that there is only one neuron with a value equal to 1. Can we jump out of this limit and set more neurons as value 1?

Binary method is proposed to optimize the output layer in the paper. For a k-class problem, suppose that $2^{a-1} < k \leq 2^a$ with $a \geq 2$. After that, a output neurons are employed, and the ideal outputs are set as the binary method. For example, for a five-class problem, the number of output neuron is 3, and the ideal outputs are $(0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 0)$, $(0, 1, 1)$, $(1, 0, 0)$. Thus, from this example, we can see that the binary method uses less output neurons than one-to-one method. If one-to-one and binary methods are applied for a sixteen-class problem, the ideal output neurons are 16 and 4, respectively. Therefore, the more the number of classes, the better the effect of the binary method. What's more, the weight number between the last hidden layer and output layer can be cut down on a large scale in the training process.

The remaining chapters of this article are as follows. In Section II, we introduce deep neural networks briefly. The binary method and some feasibility explanation are presented in Section III. Numerical experiments on ten datasets are presented to confirm the feasibility of the binary method in Section IV. Finally, some conclusions are given in Section V.

II. BRIEF INTRODUCTION OF DNN

A. FORWARD PROPAGATION OF DNN

Let us begin with the introduction of DNN. DNN is widely used because of its powerful performance: Since DNN can approximate any nonlinear function, it has good classification ability. DNN is mainly composed of input layer, several hidden layers and output layer. Each layer contains several neurons (see Fig. 1). Set the total number of layers as L, the input neuron number as p, the hidden neuron number as $m_1, m_2, \ldots, m_{L-2}$, and the output neuron number as q. The number of input neurons corresponds to the feature number of the training samples, and the input information propagates to the first hidden layer, then subsequently propagates to each hidden layer, finally reaches the output layer. The performance of DNN depends on the input characteristics, model structure and training algorithm.

In order to better display the DNN, a five-layer network is taken to introduce the calculation process. This network includes an input layer, three hidden layers, and an output layer. During the process from the input layer to the hidden layer, the output of the previous layer will be used as the input of the current layer. The calculation rule is usually: Calculate the sum of the product between the weight and the input, and then add the corresponding bias. w_{ij}^l denotes the weight from the j-th neuron in the $(l-1)$-th layer to the i-th neuron in the l-th layer. And W_i^l is the weight vector of i-th neuron in the l-th layer connecting $(l-1)$-th layer. b^l represents bias in l-th layer.

Specifically, for an input $X \in \mathbb{R}^p$, the output vector Y^2 of the first hidden layer is

$$\begin{aligned} y_i^2 &= \sigma(z_i^2) = \sigma(W_i^2 X + b_i^1) \\ &= \sigma\Big(\sum_{j=1}^{p} w_{ij}^2 \cdot x_j + b^1\Big), \; 1 \leq i \leq m_1; \end{aligned} \tag{1}$$

where $W_i^2 = (w_{i1}^2, w_{i2}^2, \ldots, w_{ip}^2)$ denotes the weight vector between the input and the first hidden layer. Analogously, the output vector Y^3 of the second hidden layer is

$$\begin{aligned} y_i^3 &= \sigma(z_i^3) = \sigma(W_i^3 Y^2 + b_i^2) \\ &= \sigma\Big(\sum_{j=1}^{m_1} w_{ij}^3 \cdot y_j^2 + b^2\Big), \; 1 \leq i \leq m_2; \end{aligned} \tag{2}$$

The output vector Y^4 of the third hidden layer is

$$\begin{aligned} y_i^4 &= \sigma(z_i^4) = \sigma(W_i^4 Y^3 + b_i^3) \\ &= \sigma\Big(\sum_{j=1}^{m_2} w_{ij}^4 \cdot y_j^3 + b^3\Big), \; 1 \leq i \leq m_3; \end{aligned} \tag{3}$$

The final output is

$$\begin{aligned} o_i = y_i^5 &= \sigma(z_i^5) = \sigma(W_i^5 Y^4 + b_i^4) \\ &= \sigma\Big(\sum_{j=1}^{m_3} w_{ij}^5 \cdot y_j^4 + b^4\Big), \; 1 \leq i \leq q. \end{aligned} \tag{4}$$

To simplify, we use matrix to represent the output. Suppose W^l be a $n \times m$ matrix connecting $l-1$-th layer to l-th layer, where in the l layer there are n neurons and m neuron in the $l-1$ layer. The output of layer l can be expressed by using the matrix method

$$Y^l = \sigma(z^l) = \sigma(W^l Y^{l-1} + b^{l-1}). \tag{5}$$

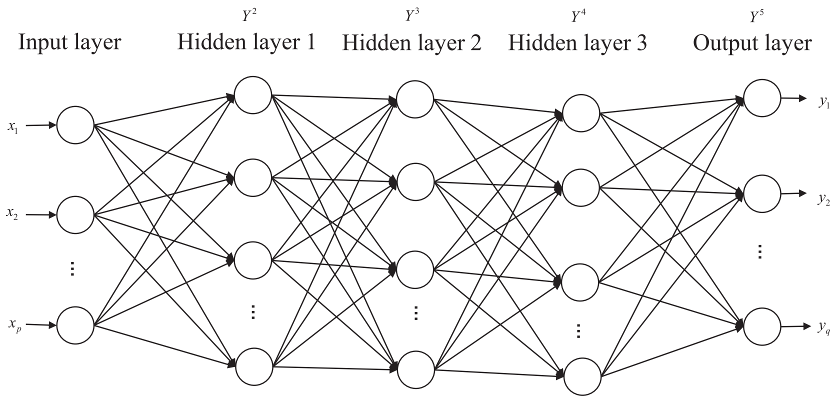


FIGURE 1. The framework of DNN.

B. BACK PROPAGATION OF DNN

The forward propagation is employed to calculate the actual output of training samples, and the error function is used to reflect the difference between calculated and real sample label. The core of back propagation (BP) algorithm [19], [20], [21] is to iteratively optimize the error function to find the minimum value according to the gradient descent method [22], [23], calculate the suitable weight matrices and biases to make the output calculated from all training sample inputs equal to or nearly equal to the sample label as much as possible.

Specifically, the BP algorithm in DNN is as follows:

Step 1: Input the total number of layers L, the neuron number of input, each hidden and output layers, activation function, training samples $(X^h, Z^h) \subset \mathbb{R}^p \times \mathbb{R}^q, h = 1, \ldots, N$. Here, X^h represents the features of the h-th sample, and Z^h represents the label of the h-th sample.

Step 2: Encode the classes of all the samples with one-to-one and binary methods, respectively.

Step 3: Define the error function [24], [25], [26]:

$$E(W^1 \cdots, W^{L-1}, b^1, \cdots, b^{L-1}) = \frac{1}{2}\sum_{h=1}^{N} ||O^h - Z^h||^2, \tag{6}$$

where O^h is the actual output calculated by Eqs. (1)-(4) or Eq. (5).

Step 4: Given the initial weight matrices $W^1 \cdots, W^{L-1}$ and biases $b^1, \cdots, b^{L-1}$.

Step 5: Update the weight matrices and biases iteratively according to the gradient method:

$$W^{l,\alpha+1} = W^{l,\alpha} - \eta \frac{\partial E(W^1 \cdots, W^{L-1}, b^1, \cdots, b^{L-1})}{\partial W^l};$$
$$b^{l,\alpha+1} = b^{l,\alpha} - \eta \frac{\partial E(W^1 \cdots, W^{L-1}, b^1, \cdots, b^{L-1})}{\partial b^l}, \tag{7}$$

where α stands for the iteration steps.

Step 6: Compute the actual output of the test samples according to the trained DNN, and then calculate the error.

Step 7: Compare the classification given by the network with the actual classification, and calculate the classification accuracies.

III. BINARY METHOD

A. INTRODUCTION TO BINARY METHOD

For a k-class problem, the traditional output layer setting is one-to-one method. If the class number is j, $(j \leq k)$, then the ideal output is set as j-th unit vector in $\mathbb{R}^k$. In detail, suppose the input vector X belongs to j-th class, the ideal output vector Y is

$$Y = (0, 0, \ldots, 0, 1, 0, \ldots, 0)^T \in \mathbb{R}^k, \tag{8}$$

where the j-th element is 1 and and the others are all 0. If the network actual output fulfillments

$$O \approx Y, \tag{9}$$

then the input $X \in \mathbb{R}^p$ is classified into j-th class. We claim that the one-to-one method has successfully solved the classification problem if it satisfies (9) for each input sample in the j-th class. Simultaneously, "failed classification" refers to classifying an i-th class sample into j-th class $(i \neq j)$. For a 5-class problem, take one-to-one method as an example. If the class label of a sample is 1, then the ideal output is set as $Y = (1, 0, 0, 0, 0)^T$. If the network actual output fulfillments $O \approx Y$, then this sample is recorded as "successful classification". However, when there is a significant difference between O and Y, it is called "failed classification".

It is not difficult to see that the above one-to-one method requires too many output neurons. The root reason is that there is only one neuron with the value equal to 1. Can we break this limit? For the same k-class problem, we will now introduce the binary method. If the value of each neuron can be 0 and 1, then we only need $\lceil log_2 k \rceil$ neurons. Let $a = \lceil log_2 k \rceil$, then the output neuron number can be cut down on a large scale. The ideal output can be designed in a binary system. In other words, the ideal output of j-th class is

$$t(j) = (t_1, t_2, \ldots, t_a)^T, \tag{10}$$

and

$$t_1 * 2^{(a-1)} + t_2 * 2^{(a-2)} + \ldots + t_a * 2^0 = j - 1, \tag{11}$$

where $t_i = 0$ or $1, \ i = 1, \ldots, a$.

For example, let the class number be $k = 8$, then we can get $a = 3$. And the ideal output vector of the eight classes can be listed as (0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1), respectively. Obviously, we only need three neurons in the output layer instead of eight in one-to-one method.

Analogously, we claim that this classification problem has been successfully solved by the binary method if it satisfies (12) for each input sample in the j-th class

$$O \approx t(j). \tag{12}$$

B. SOME FEASIBILITY EXPLANATION ABOUT BINARY METHOD

For a parity problem [27], [28] in neural networks, the ideal outputs of two classes are labeled as 1 and 0. For the two-class problem, we only use one output neuron. Essentially, this is binary method. Here, no one will use one-to-one method: For the two-class problem, these two classes are labeled as (1, 0) and (0, 1). Therefore, this example implies that binary method is more practical than one-to-one method at least in some cases.

For linear perceptron, an output neuron corresponds to a straight line in geometry. For a four-class problem, the one-to-one method is to find four straight lines l_i ($1 \leq i \leq 4$) so that l_i can just separates the i-th class from the other three classes (see Fig. 2). For the quadrilateral formed by the intersection of the above four lines, we take the lines l_a and l_b where the diagonal lies. These two lines are the two output neurons in the corresponding binary method. Therefore, suppose that a four-class problem can be successfully solved by one-to-one method, then it can also be solved by binary method.

However, when a four-class problem can be solved by binary method, the one-to-one method is not necessarily feasible. In Fig. 2(b), we give a special case. In this case, we cannot find four straight lines to separate each class of sample from the other three classes. But we can find two lines l_c and l_d to separate these four classes from each other. This example indicates the binary method is more universal and has wider application than the one-to-one method.

IV. NUMERICAL EXPERIMENTS

In this section, some numerical experiments are conducted to compare the performance of the proposed binary method with the conventional one-to-one method on several multi-class problems. In the first subsection, we give the experiment settings; Then the comparative experimental results and some analysis are given in the second subsection.

TABLE 2. Information of ten different datasets and output neuron number in two methods.

Dataset	Attribute	Class	Output neuron number	
			One-to-one	Binary
Mnist	784	10	10	**4**
Drive	48	8	8	**3**
Letter	16	26	26	**5**
CM	28	6	6	**3**
IS	20	7	7	**3**
wine	13	3	3	**2**
car	6	4	4	**2**
iris	4	3	3	**2**
LS	36	7	7	**3**
covertype	55	7	7	**3**

A. EXPERIMENT SETTINGS

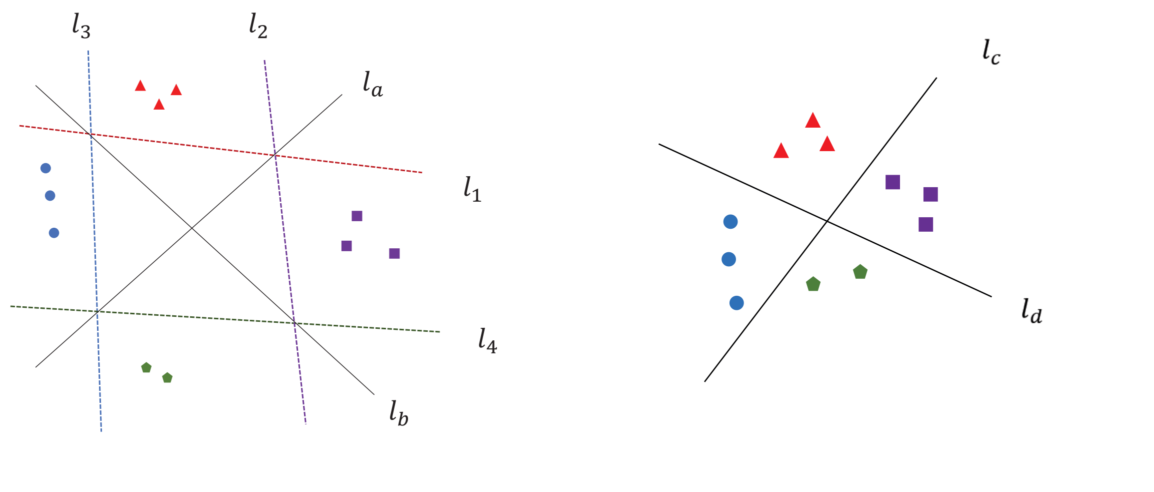
We mainly select ten multi-class problems, including Mnist, Drive, Letter, CM (crowdsourced mapping), IS (image segmentation), Wine, Car, Iris, LS (landsat satellite) and Covertype, which are mainly published in the UCI Machine Learning. The information of these ten specific datasets and the output neuron number in these two methods are shown in Tab. 2. Five-fold cross-validation technique is employed for both one-to-one and binary methods. In detail, the dataset is divided into five parts equally or almost equally, and then we can conduct five times of network learning. Then, each of the five parts is in turn as the test sample set, and the other four parts as the training sample set. We rearrange the samples and repeat these processes ten times. For each method data pair, a total of fifty classification results can be obtained. Each time, one of the five parts are in turn selected as the testing set, and the other four parts as the training set. In this way, we can get five sets of experimental results. Then we re-conduct the above process ten times. For each method data-pair, classification results can be obtained. The flow chart of the experimental process is shown in Fig. 3.

For both of the one-to-one and the binary methods, the ideal value for each output neuron is 1 or 0. According to the actual output, we can make a judgement which class the sample should belong to. Here, the following 40-20-40 criteria is employed: the actual output value less than or equal to 0.40 is regarded as 0, the actual output value bigger than or equal to 0.60 is regarded as 1; and when it is bigger than 0.4 and less than 0.6, we consider that the classification is failed. The activation function used in our experiments is sigmoid function

$$\sigma(x) = 1/(1 + \exp(-x)). \tag{13}$$

The details of the experiment is:

Step 1: Given the dataset $\mathbf{\Phi} = \{(\boldsymbol{x}_j, \boldsymbol{t}_j)|\boldsymbol{x}_j \in \mathbb{R}^p, \boldsymbol{t}_j \in \mathbb{R}^q, j = 1, \ldots, N\}$. For all the labels, encode the ideal outputs in binary and one-to-one methods.



(a) Dotted line represents one-to-one method; Solid line for binary method. (b) The binary method can solve this four-class problem, while the one-to-one method fails.

FIGURE 2. Illustration of one-to-one and binary methods in two special cases.

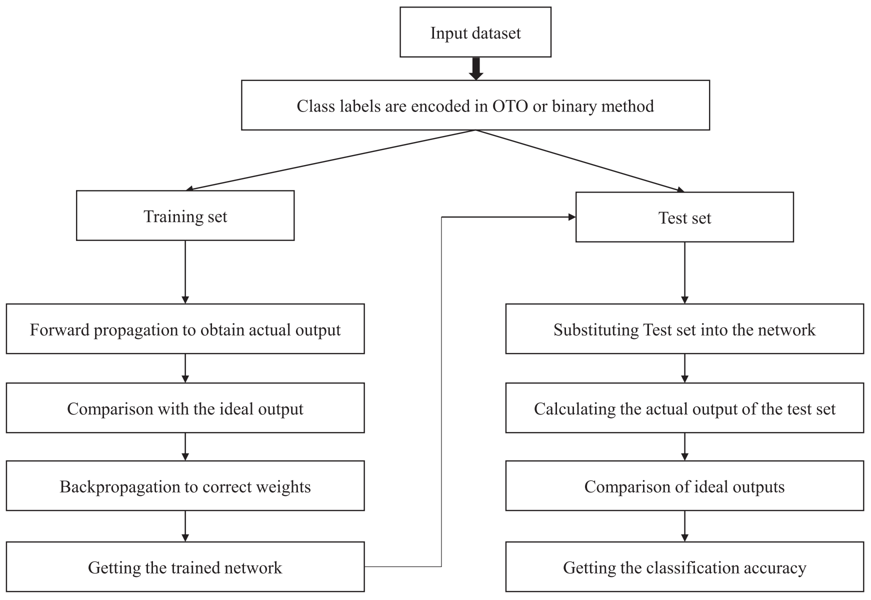

Input dataset
Class labels are encoded in OTO or binary method
Training set
Test set
Forward propagation to obtain actual output
Comparison with the ideal output
Backpropagation to correct weights
Getting the trained network
Substituting Test set into the network
Calculating the actual output of the test set
Comparison of ideal outputs
Getting the classification accuracy

FIGURE 3. The flow chart of the experiment.

Step 2: *Five*-fold cross validation technology: $\mathbf{\Phi} = \{(\boldsymbol{x}_j, \boldsymbol{t}_j)|\boldsymbol{x}_j \in \mathbb{R}^p, \boldsymbol{t}_j \in \mathbb{R}^q, j = 1, \ldots, N\}$ is equally divided into *five* parts: $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_5$.
Step 3: For $i = 1$ to $i = 5$, do Step 3 to Step 7. Let $\mathbf{\Phi}_i$ be the test samples, while $\mathbf{\Phi} \setminus \mathbf{\Phi}_i$ is the training samples.
Step 4: Calculate the corresponding output $\boldsymbol{o}_j$ based on Eqs. (1)-(4). Subsequently, give the error function as Eq. (6).
Step 5: Update the weights and biases of the network based on BP algorithm.
Step 6: For each test sample, calculate the the actual output according to the weights and biases computed in the previous step.
Step 7: For each dataset, calculate the classification accuracy and draw the error function curve.
Step 8: Repeat Steps 2 to 7 for ten times.
Step 9: Average the *fifty* experimental results and compare the binary method with the one-to-one method.

B. EXPERIMENTAL RESULTS

After the above algorithm calculation, we compare the classification accuracies of one-to-one and binary methods with the ten datasets mentioned previously (cf. Tab. 3). Since the error is very small, we take the logarithm of the error value to better reflect the difference between these two methods. In the experiment, the layer number of the deep neural

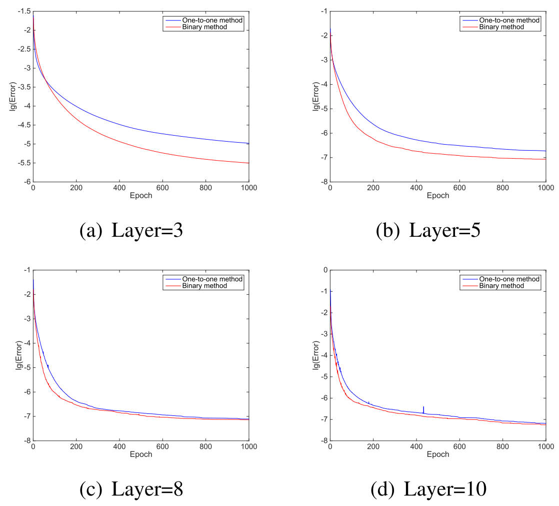


FIGURE 4. Error functions of Mnist dataset based on the one-to-one and binary methods.

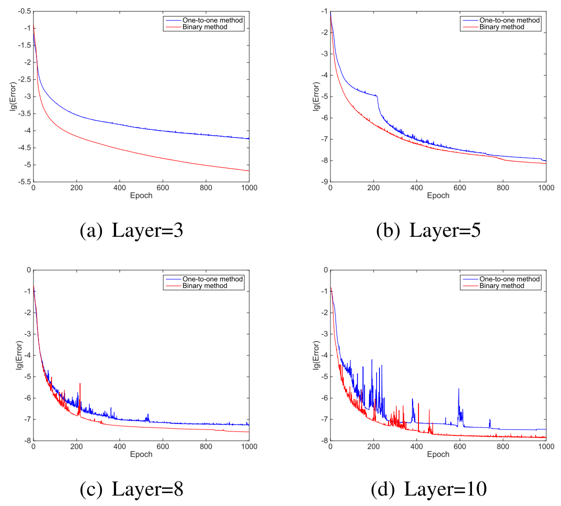


FIGURE 5. Error functions of Drive dataset based on the one-to-one and binary methods.

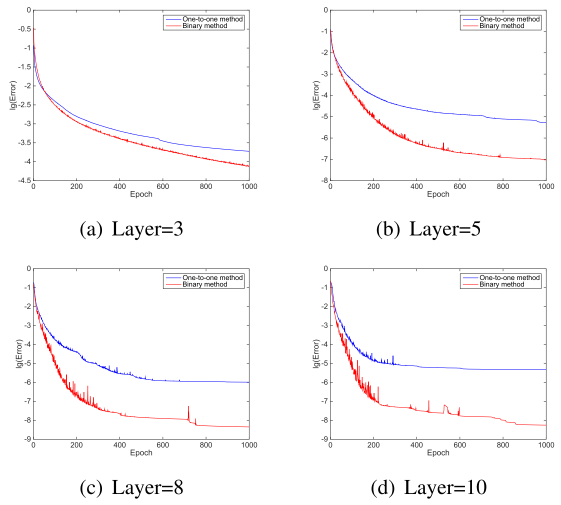


FIGURE 6. Error functions of letter dataset based on the one-to-one and binary methods.

network is set as 3, 5, 8 and 10, respectively. From Tab. 3, we can easily conclude that no matter how many the number of neural network layers is, the binary method outperforms the one-to-one method on all accuracies (including Minst, Drive, Letter, IS, Wine, Car and Covertype). In addition, in other three datasets (CM, Iris and LS), the binary method outperforms the one-to-one method on the whole. Besides the classification accuracies, we also compare the errors of these two methods and draw the error function curves (cf. Figs. 4-13). Taking Fig. 4(a) as an example, it displays the error functions of two methods on the Mnist dataset regarding the iteration steps. From this figure, it can be seen that as the number of iteration steps increases, the error will become smaller; In addition, the error in binary method is lower than that of the one-to-one method. Obviously, in all these ten cases, the error of the binary method is less than that of the one-to-one method. Specially, in half cases, the binary method is far superior to the one-to-one method; And in the remaining half, the binary method is slightly higher than the one-to-one method.

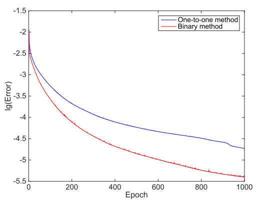
Besides, since the output neurons employed in binary are decreased, the number of neurons connecting last hidden to output layers will also decrease. And the more the class number, the more output neurons will be reduced. Therefore, the deep neural network with the binary method has a faster calculating speed. Taking 5 and 8 layer deep neural networks as examples, we give the average calculating time of each iteration in all datasets (cf. Tab. 4).

In previous experiments, we have compared the classification accuracies and error curves of one-to-one and binary methods. In order to better compare these two methods, we continue to calculate the following indicators: standard deviation(σ) [29], root mean square error(RMSE) [30], prediction rate(PR) [31], recall rate(RR) [32] and F1-measure [33]. Since the original PR and RR are aimed at the binary classification problems, we extend the definition of PR and RR to the multi-class classification problems in our experiment. For any k-class problem and any r ($1 \le r \le k$), we regard the rth-class samples as the positive samples and all the rest as the negative samples; Next, we can calculate the corresponding TP_r (the sample number that belong to r-th class and are correctly classified), FP_r (the sample number that do not belong to r-th class but are failed classified into class r), and FN_r (the sample number that belong to r-th class but are not classified into r-th class); Then we can calculate PR_r and RR_r; Finally, average $\{PR_1, PR_2, \ldots, PR_k\}$ and $\{RR_1, RR_2, \ldots, RR_k\}$, thus PR and RR of k-class problem can be obtained. The mathematical calculation formula of these indicators are as follows:

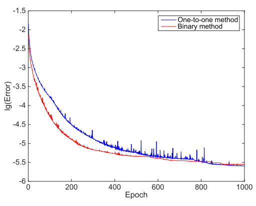
$$\sigma := \sqrt{\frac{1}{S-1}\sum_{i=1}^{S-1}\frac{1}{k}\sum_{j=1}^{k}(y_{ij}-\overline{y_j})^2}; \qquad (14)$$

TABLE 3. Comparison of classification accuracy between one-to-one (OTO) and binary methods.

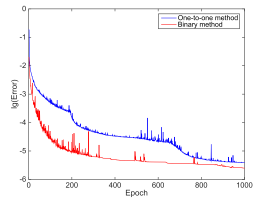
Dataset	Accuracy	Layer=3		Layer=5		Layer=8		Layer=10	
		OTO	Binary	OTO	Binary	OTO	Binary	OTO	Binary
Mnist	Training	99.05%	**99.38%**	99.79%	**99.86%**	99.85%	**99.87%**	99.89%	**99.90%**
	Test	90.51%	**91.69%**	94.10%	**94.97%**	94.47%	**95.60%**	94.72%	**95.78%**
Drive	Training	98.68%	**99.24%**	99.93%	**99.94%**	99.95%	**99.96%**	99.95%	**99.97%**
	Test	97.20%	**98.65%**	99.03%	**99.16%**	99.24%	**99.30%**	99.27%	**99.41%**
Letter	Training	95.93%	**98.23%**	99.09%	**99.47%**	99.51%	**99.55%**	99.64%	**99.71%**
	Test	86.50%	**91.18%**	92.98%	**95.06%**	94.72%	**95.82%**	99.53%	**99.54%**
CM	Training	98.78%	**99.19%**	99.05%	**99.46%**	99.43%	**99.48%**	99.64%	**99.71%**
	Test	92.51%	**92.72%**	92.93%	**93.22%**	93.81%	**94.35%**	**94.41%**	94.39%
IS	Training	99.93%	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**
	Test	99.01%	**99.38%**	99.51%	**99.63%**	99.63%	**99.75%**	**99.88%**	**99.88%**
wine	Training	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**
	Test	99.80%	**99.91%**	99.83%	**100.00%**	99.76%	**99.85%**	**100.00%**	**100.00%**
car	Training	96.92%	**97.83%**	97.92%	**98.58%**	99.51%	**99.67%**	99.58%	**99.83%**
	Test	95.64%	**96.40%**	96.21%	**96.59%**	98.10%	**98.63%**	98.48%	**99.24%**
iris	Training	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**
	Test	92.10%	**92.60%**	96.30%	**98.20%**	**98.40%**	98.30%	98.70%	**99.10%**
LS	Training	89.50%	**89.83%**	**96.81%**	95.57%	**98.00%**	97.23%	98.10%	**98.27%**
	Test	83.55%	**85.57%**	86.48%	**88.08%**	88.15%	**88.22%**	**89.06%**	89.03%
covertype	Training	99.11%	**99.52%**	99.79%	**99.87%**	99.92%	**99.97%**	99.84%	**99.89%**
	Test	98.44%	**99.26%**	99.30%	**99.46%**	99.57%	**99.65%**	99.35%	**99.47%**



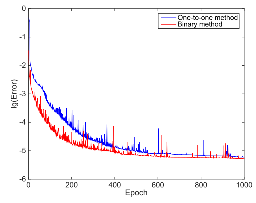
(a) Layer=3



(b) Layer=5



(c) Layer=8



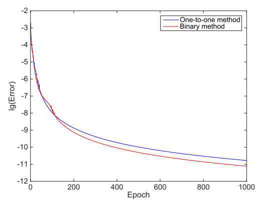
(d) Layer=10

FIGURE 7. Error functions of CM dataset based on the one-to-one and binary methods.

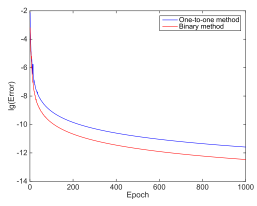
TABLE 4. Average calculating time of each iteration for one-to-one and binary methods.

Dataset	Layer=5		Layer=8	
	One-to-one	Binary	One-to-one	Binary
Mnist	178.96s	**162.27s**	221.67s	**206.79s**
Drive	0.41897s	**0.37126s**	0.67902s	**0.60704s**
Letter	9.2554s	**7.7081s**	14.261s	**12.582s**
CM	3.3058s	**3.0272s**	5.4529s	**5.3087s**
IS	0.68485s	**0.63675s**	1.0289s	**0.98447s**
wine	0.071679s	**0.069503s**	0.099583s	**0.097721s**
car	0.74017s	**0.069004s**	0.13724s	**0.12658s**
iris	0.069173s	**0.064775s**	0.10981s	**0.091008s**
LS	0.078631s	**0.067561s**	0.090684s	**0.083206s**
covertype	205.31s	**189.60s**	274.82s	**251.17s**

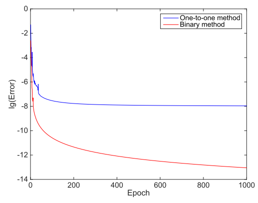
$$\mathrm{RMSE} := \sqrt{\frac{1}{S}\sum_{i=1}^{S}\frac{1}{k}\sum_{j=1}^{k}(y_{ij}-o_{ij})^2}; \tag{15}$$



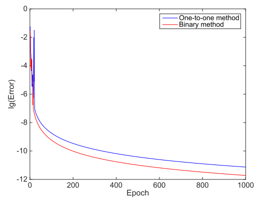
(a) Layer=3



(b) Layer=5



(c) Layer=8



(d) Layer=10

FIGURE 8. Error functions of IS dataset based on the one-to-one and binary methods.

$$\mathrm{PR} := \frac{1}{k}\sum_{r=1}^{k}\frac{\mathrm{TP}_r}{\mathrm{TP}_r+\mathrm{FP}_r}; \tag{16}$$

$$\mathrm{RR} := \frac{1}{k}\sum_{r=1}^{k}\frac{\mathrm{TP}_r}{\mathrm{TP}_r+\mathrm{FN}_r}; \tag{17}$$

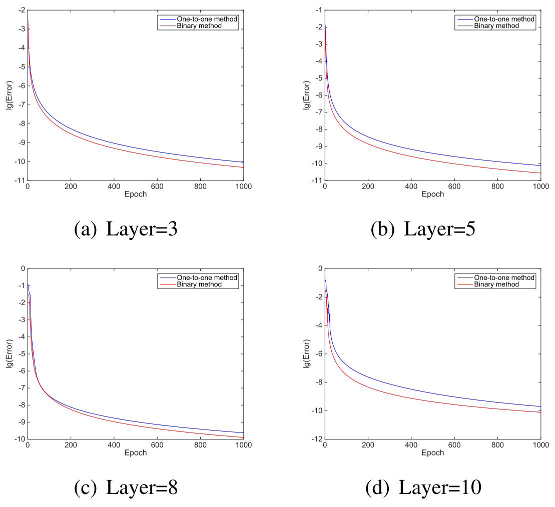
$$\mathrm{F1} := \frac{2\times\mathrm{PR}\times\mathrm{RR}}{\mathrm{PR}+\mathrm{RR}}. \tag{18}$$

Here, S is the number of samples in the training set, $(y_{i1}, y_{i2}, \ldots, y_{ik})$ denotes the actual output of the i-th sample, $(o_{i1}, o_{i2}, \ldots, o_{ik})$ represents the ideal output of the i-th sample. $\overline{y_j}$ is the average of the j-th component of the actual output of all samples, and the specific calculation formula is

$$\overline{y_j} = \frac{1}{S}\sum_{i=1}^{S} y_{ij}. \tag{19}$$

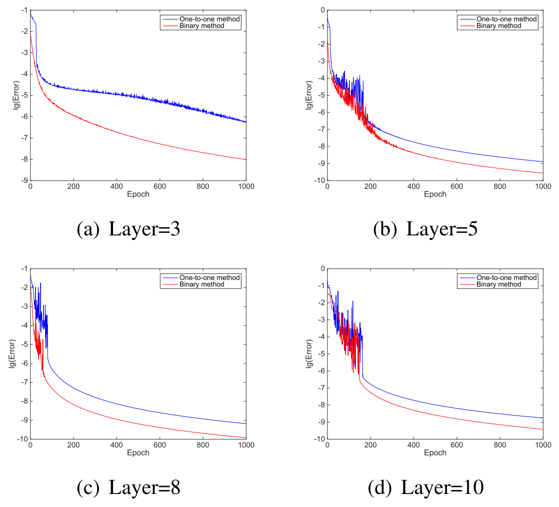
TABLE 5. Some indicators comparison for one-to-one and binary methods (Layer=5).

Dataset	One-to-one					Binary				
	PR	RR	F1	σ	RMSE	PR	RR	F1	σ	RMSE
Mnist	94.55%	92.18%	93.78%	0.0219	0.0186	**95.42%**	**93.03%**	**94.21%**	**0.0208**	**0.0150**
Drive	98.83%	**99.14%**	98.98%	0.0103	0.0036	**99.07%**	99.05%	**99.06%**	**0.0091**	**0.0028**
Letter	93.19%	78.35%	85.13%	0.0107	0.0034	**93.81%**	**81.03%**	**86.95%**	**0.0082**	**0.0020**
CM	**92.97%**	85.81%	89.25%	0.0164	**0.0135**	92.76%	**89.03%**	**90.86%**	**0.0153**	0.0139
IS	99.36%	99.04%	99.20%	0.0146	0.0057	**99.48%**	**99.43%**	**99.45%**	**0.0118**	**0.0039**
wine	**99.82%**	99.12%	99.47%	0.0204	0.0129	99.78%	**99.53%**	**99.65%**	**0.0155**	**0.0083**
car	**94.57%**	92.61%	**93.58%**	**0.0193**	0.0142	93.81%	**93.07%**	93.44%	0.0208	**0.0130**
iris	94.14%	94.18%	94.16%	0.0122	0.0120	**96.31%**	**96.06%**	**96.18%**	**0.0059**	**0.0081**
LS	87.56%	**81.08%**	84.20%	0.0278	0.0171	**91.15%**	79.63%	**85.00%**	**0.0242**	**0.0125**
covertype	98.32%	98.05%	98.18%	0.0286	0.0257	**98.66%**	**98.10%**	**98.38%**	**0.0226**	**0.0209**



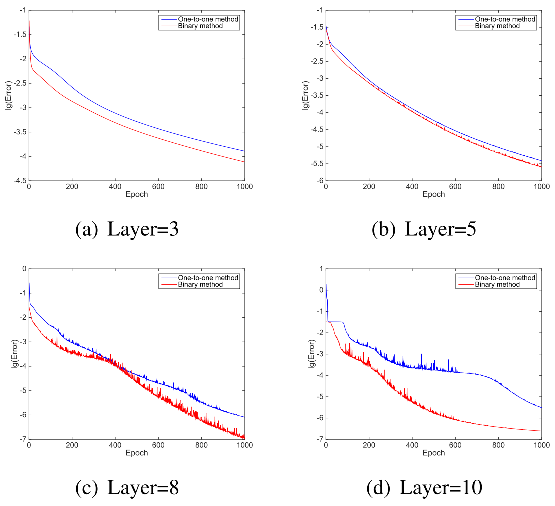
(a) Layer=3 (b) Layer=5
(c) Layer=8 (d) Layer=10

FIGURE 9. Error functions of wine dataset based on the one-to-one and binary methods.



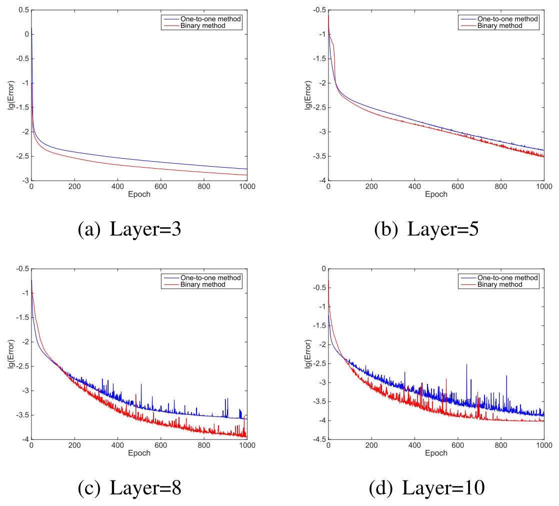
(a) Layer=3 (b) Layer=5
(c) Layer=8 (d) Layer=10

FIGURE 11. Error functions of iris dataset based on the one-to-one and binary methods.



(a) Layer=3 (b) Layer=5
(c) Layer=8 (d) Layer=10

FIGURE 10. Error functions of car dataset based on the one-to-one and binary methods.



(a) Layer=3 (b) Layer=5
(c) Layer=8 (d) Layer=10

FIGURE 12. Error functions of LS dataset based on the one-to-one and binary methods.

And Tab. 5 shows the above mentioned five indicators. In Mnist, Letter, IS, Iris and Covertype datasets, the binary method outperforms the one-to-one method under each indicator. And for the remaining five datasets, only on one or two criterias the one-to-one method outperforms our binary method. Based on the above analysis, the performance of binary method is significantly better than that of one-to-one method in these five indicators. Furthermore, when

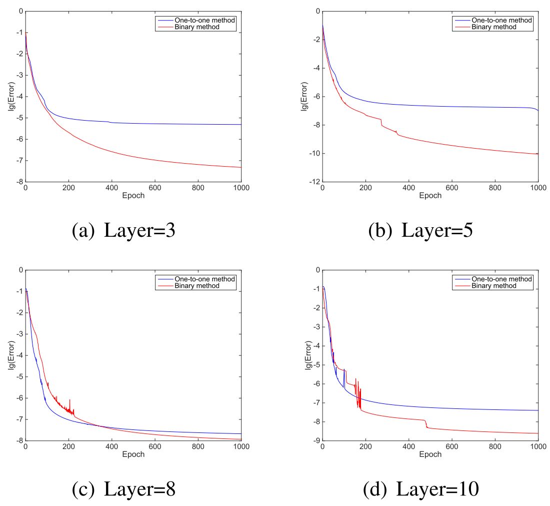


FIGURE 13. Error functions of covertype dataset based on the one-to-one and binary methods.

the dataset has a large number of samples or classes, the advantages of binary methods are more prominent. For the Mnist and covertype datasets, as well as the letter dataset with 26 classes, the binary method outperforms the one-to-one method in all indicators.

V. CONCLUSION

This paper mainly studies the optimization of output layer structure in deep neural networks. The common output layer design is the one-to-one method: For a k-class problem, k output neurons are requisitioned. However, the one-to-one method utilizes too many output neurons and weights, which will reduce the learning efficiency. In order to avoid these shortcomings, the binary method was proposed: Break the restriction that there is only one neuron in the one-to-one method and let any output neuron can be set as value 1. In this way, for the k-class problem, only $\lceil log_2 k \rceil$ output neurons are enough to solve the classification task. Moreover, the coding method adopts the binary manner. The experimental results show that the classification performance of the binary method is better than that of the one-to-one method regardless of the layer number in DNNs. Moreover, in terms of the learning efficiency, the binary method also has a faster learning speed.

REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[2] W. He, M. Wu, and S. Lam, "ACSL: Adaptive correlation-driven sparsity learning for deep neural network compression," *Neural Netw., Off. J. Int. Neural Netw. Soc.*, vol. 144, pp. 465–477, Dec. 2021.

[3] M. Xie, W. Sun, L. Huang, C. Xu, and H. Tan, "Accurate and efficient matrix completion using cascaded deep neural network," in *Proc. IEEE 6th Int. Conf. Comput. Commun. Syst. (ICCCS)*, Apr. 2021, pp. 174–179.

[4] L. Huang, Y. Zhou, L. Liu, F. Zhu, and L. Shao, "Group whitening: Balancing learning efficiency and representational capacity," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 9512–9521.

[5] M. An, T. Taura, and T. Shiose, "A study on acquiring underlying behavioral criteria for manipulator motion by focusing on learning efficiency," *IEEE Trans. Syst. Man, Cybern.—Part A, Syst. Humans*, vol. 37, no. 4, pp. 445–455, Jul. 2007.

[6] Y. Zheng, Y. Wang, and J. Liu, "Research on structure optimization and motion characteristics of wearable medical robotics based on improved particle swarm optimization algorithm," *Future Gener. Comput. Syst.*, vol. 129, pp. 187–198, Apr. 2022.

[7] R. Wang, Z. Zheng, W. Yu, Y. Shao, and S. Gao, "Structure-aware geometric optimization of hexahedral mesh," *Comput.-Aided Design*, vol. 138, Sep. 2021, Art. no. 103050.

[8] C. Manzano, C. Meneses, P. Leger, and H. Fukuda, "An empirical evaluation of supervised learning methods for network malware identification based on feature selection," *Complexity*, vol. 2022, pp. 1–18, Apr. 2022.

[9] R. Purushothaman, S. Selvakumar, and S. P. Rajagopalan, "Hybrid grasshopper and chameleon swarm optimization algorithm for text feature selection with density peaks clustering," *Int. J. Comput. Intell. Appl.*, vol. 21, no. 3, Sep. 2022, Art. no. 2250018.

[10] A. N. M. B. Rashid, M. Ahmed, L. F. Sikos, and P. Haskell-Dowland, "Anomaly detection in cybersecurity datasets via cooperative co-evolution-based feature selection," *ACM Trans. Manage. Inf. Syst.*, vol. 13, no. 3, pp. 1–39, Sep. 2022.

[11] P. Pudil, J. Novovič ová, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognit. Lett.*, vol. 15, no. 11, pp. 1119–1125, Nov. 1994.

[12] B. Bilgic, I. Chatnuntawech, A. P. Fan, K. Setsompop, S. F. Cauley, L. L. Wald, and E. Adalsteinsson, "Fast image reconstruction with L2-regularization," *J. Magn. Reson. Imag.*, vol. 40, no. 1, pp. 181–191, Jul. 2014.

[13] Y. Bao, Z. Liu, Z. Luo, and S. Yang, "Smooth group $L_{1/2}$ regularization for pruning convolutional neural networks," *Symmetry*, vol. 14, no. 1, p. 154, Jan. 2022.

[14] M. Lim and T. Hastie, "Learning interactions via hierarchical group-lasso regularization," *J. Comput. Graph. Statist.*, vol. 24, no. 3, pp. 627–654, Jul. 2015.

[15] A. Rocha and S. K. Goldenstein, "Multiclass from binary: Expanding one-versus-all, one-versus-one and ECOC-based approaches," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 289–302, Feb. 2014.

[16] A. C. H. Choong and N. K. Lee, "Evaluation of convolutionary neural networks modeling of DNA sequences using ordinal versus one-hot encoding method," in *Proc. Int. Conf. Comput. Drone Appl. (IConDA)*, Nov. 2017, pp. 60–65.

[17] M. Kumagai, K. Komatsu, F. Takano, T. Araki, M. Sato, and H. Kobayashi, "An external definition of the one-hot constraint and fast QUBO generation for high-performance combinatorial clustering," *Int. J. Netw. Comput.*, vol. 11, no. 2, pp. 463–491, 2021.

[18] W. Guo and S. Li, "Fast binary counters and compressors generated by sorting network," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 6, pp. 1220–1230, Jun. 2021.

[19] Y. Hirose, K. Yamashita, and S. Hijiya, "Back-propagation algorithm which varies the number of hidden units," *Neural Netw.*, vol. 4, no. 1, pp. 61–66, Jan. 1991.

[20] L. Wu, M. Zhou, Y. Wang, L. Wang, and X. Tian, "Online monitoring and early warning of subsynchronous oscillation using levenberg–marquardt and backpropagation algorithm combined with sensitivity analysis and principal component analysis," *Math. Problems Eng.*, vol. 2021, pp. 1–12, Jan. 2021.

[21] H.-W. Lu, X.-P. Yu, S.-Q. Wang, Y.-Y. Liu, Z.-Y. Huang, Z.-H. Lu, K.-S. Yeo, and J.-M. Chen, "A digital background calibration scheme for non-linearity of SAR ADC using back-propagation algorithm," *Microelectron. J.*, vol. 114, Aug. 2021, Art. no. 105113.

[22] Y. Wang, Y. He, and Z. Zhu, "Study on fast speed fractional order gradient descent method and its application in neural networks," *Neurocomputing*, vol. 489, pp. 366–376, Jun. 2022.

[23] J. Liu, R. Zhai, Y. Liu, W. Li, B. Wang, and L. Huang, "A quasi fractional order gradient descent method with adaptive stepsize and its application in system identification," *Appl. Math. Comput.*, vol. 393, Mar. 2021, Art. no. 125797.

[24] N. Baba, "A new approach for finding the global minimum of error function of neural networks," *Neural Netw.*, vol. 2, no. 5, pp. 367–373, Jan. 1989.

[25] A. Sardashti, H. Daniali, and S. Varedi-Koulaei, "Geometrical similarity error function-innovative adaptive algorithm methodology in path generation synthesis of the four-bar mechanism using metaheuristic algorithms," *Proc. Inst. Mech. Engineers, Part C, J. Mech. Eng. Sci.*, vol. 236, no. 3, pp. 1550–1570, Feb. 2022.

[26] L. Xiao, H. Tan, J. Dai, L. Jia, and W. Tang, "High-order error function designs to compute time-varying linear matrix equations," *Inf. Sci.*, vol. 576, pp. 173–186, Oct. 2021.

[27] E. M. Iyoda, H. Nobuhara, and K. Hirota, "A solution for the N-bit parity problem using a single translated multiplicative neuron," *Neural Process. Lett.*, vol. 18, no. 3, pp. 233–238, Dec. 2003.

[28] N. D. S. Ribeiro, M. A. M. Vieira, L. F. M. Vieira, and O. Gnawali, "SplitPath: High throughput using multipath routing in dual-radio wireless sensor networks," *Comput. Netw.*, vol. 207, Apr. 2022, Art. no. 108832.

[29] L. Shi, L. Shen, and B. Chen, "Complementary mean square deviation and stability analyses of the widely linear recursive least squares algorithm," *Digit. Signal Process.*, vol. 122, Apr. 2022, Art. no. 103357.

[30] D. S. K. Karunasingha, "Root mean square error or mean absolute error? Use their ratio as well," *Inf. Sci.*, vol. 585, pp. 609–629, Mar. 2022.

[31] J. Zhang, C. Ma, C. Zhong, P. Zhao, and X. Mu, "Multi-scale and multi-channel neural network for click-through rate prediction," *Neurocomputing*, vol. 480, pp. 157–168, Apr. 2022.

[32] P. Dai, Q. Zhang, Y. Wang, D. Jin, and Y. Gong, "Improving large-gap clone detection recall using multiple features," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 32, no. 7, pp. 1071–1099, Jul. 2022.

[33] M.-M. Cheng and D.-P. Fan, "Structure-measure: A new way to evaluate foreground maps," *Int. J. Comput. Vis.*, vol. 129, no. 9, pp. 2622–2638, Sep. 2021.

YUQI LIU is a Junior Student with the Marine Engineering College, Dalian Maritime University. His main research interests include algorithm improvement, image restoration in artificial intelligence, and neural networks.

SIBO YANG received the B.S. and Ph.D. degrees in computational mathematics from Dalian University of Technology, China, in 2013 and 2020, respectively. He is a Lecturer with the School of Science, Dalian Maritime University, Dalian, China. His research interests include extreme learning machine and the improvement of learning algorithms in neural networks.

YUAN BAO received the B.S. degree in mathematics and applied mathematics from Henan University, Kaifeng, China, in 2013, and the Ph.D. degree in computational mathematics from Dalian University of Technology, Dalian, China, in 2020. She is now a Teacher with the School of Mathematics and Statistics, Xinyang Normal University, Xinyang, China. Her research interests include finite element methods and computer networks.