IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 9, 2021, accepted May 24, 2021, date of publication May 28, 2021, date of current version June 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3084610

Greedy Broad Learning System

WEITONG DING[1], YUBO TIAN[1,2], SHUDAN HAN[1], AND HUINING YUAN[1]
[1]School of Electronics and Information, Jiangsu University of Science and Technology, Zhenjiang 212003, China
[2]School of Information and Communication Engineering, Guangzhou Maritime University, Guangzhou 510725, China

Corresponding author: Yubo Tian (tianyubo@just.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61771225, and in part by the Qinglan Project of Jiangsu Higher Education.

ABSTRACT In order to overcome the extremely time-consuming drawback of deep learning (DL), broad learning system (BLS) was proposed as an alternative method. This model is simple, fast, and easy to update. To ensure the fitting and generalization ability of BLS, the hidden layer neurons are often set too many, in fact, a lot of neurons are not needed. Greedy BLS (GBLS) is proposed in this paper to deal with the redundancy of the hidden layer in BLS from another perspective. Different from BLS, the structure of GBLS can be seen as a combination of unsupervised multi-layer feature representation and supervised classification or regression. It trains with a greedy learning scheme, performs principal component analysis (PCA) on the previous hidden layer to form a set of compressed nodes, which are transformed into enhancement nodes and then activated by nonlinear functions. The new hidden layer is composed of all newly generated compressed nodes and enhancement nodes, and so on. The last hidden layer of the network contains the higher-order and abstract essential features of the original data, which is connected to the output layer. Each time a new layer is added to the model, and there is no need to retrain from the beginning, only the previous layer is trained. Experimental results demonstrate that the proposed GBLS model outperforms BLS both in classification and regression.

INDEX TERMS Deep learning, broad learning system, principal component analysis, greedy learning.

I. INTRODUCTION

In recent years, neural networks (NNs) have been widely used in a series of challenging fields such as image recognition [1], computer vision [2], [3] and large-scale data processing [4], [5]. Among them, deep learning (DL) has performed particularly well. However, due to the complex structure of deep neural networks (DNNs), it is difficult to analyze them. So far, the theory of DNNs is still scarce, and most NNs are used as black boxes [6]. Although DNNs are so powerful in their approximation and feature extraction capabilities, they are limited by complexity of the structure and the large number of hyperparameters. In order to effectively build a model, it is necessary to continuously adjust the number of NN layers and the number of nodes required for each layer. The iterative method is used to determine the connection weights between each layer, and the training process is extremely time-consuming and computationally expensive when the amount of data is huge. The learning efficiency and speed of traditional deep structures are far lower than the requirements, becoming an important bottleneck for many applications.

The associate editor coordinating the review of this manuscript and approving it for publication was Varuna De Silva.

To solve this problem, Chen proposed broad learning system (BLS) [7] in 2017 and proved that it has universal approximation capabilities [8]. Some subsequent experimental results also proved that BLS is easy to extend to other NNs. For example, Chen *et al.* [9] applied broad learning algorithm and incremental learning algorithm to the radial basis function (RBF) network [10] and hierarchical extreme learning machines (H-ELM) [11]. BLS is a random vector single-layer NN learning system that uses the random vector functional-link NN (RVFLNN) [12]–[14] as its carrier and realizes the horizontal expansion of the network through the increase of neural nodes in the hidden layer. Unlike RVFLNN, which directly brings the original data into the network, BLS first maps the data to feature nodes, in which the input weight matrix is not randomly generated, but uses the sparse autoencoder to obtain the optimal input weights. Compared with the deep structure, BLS only contains one hidden layer, which is composed of the feature layer and the enhancement layer. Considering the high time cost of the classic error back propagation (BP) algorithm [15] and the shortcomings of being easily trapped in local minimums, the performance of the network is often affected by the initialization area. Therefore, the output weights of BLS are solved by ridge regression

generalized inverse [16] by default, which is consistent with the approach of RVFLNN. BLS has a simple structure, and only the output weights need to be solved, which can achieve satisfactory accuracy in a short training time. In addition, BLS uses the rapid incremental learning algorithm proposed by Chen *et al.* in 1999 [17] to solve the problems caused by the increase in data volume and data dimensions. New inputs, feature nodes, and enhancement nodes can be added according to actual needs, so the network can be quickly remodeled without a complete retraining process. For traditional NNs, the structure of the network is fixed, and the parameters need to be optimized to obtain better performance. On the contrary, the parameters of BLS are randomly selected and fixed, and hidden layer nodes are added through horizontal expansion to improve the fitting ability of the model.

As soon as BLS was proposed, it has received extensive attention from the academia, and many scholars have conducted researches on it. In order to obtain acceptable performance on more complex data sets, Liu *et al.* [18] used the K-means clustering algorithm as an improved feature extraction method to improve the fitting ability of the model. Chen *et al.* [19] proposed a robust BLS based on regularization to model uncertain data, and has better generalization ability for data with noise and outliers. Xu *et al.* [20] added recursive connections to the enhancement nodes to make the network have the ability to remember historical information, and proposed recurrent BLS for time series prediction. Zhao *et al.* [21] extended the BLS based on the popular regularization framework and proposed a semi-supervised BLS, which can use a large number of unlabeled samples and a small number of labeled samples to achieve semi-supervised classification. Until very recently, various BLS methods are summarized by Chen *et al.* [22] from the aspects of algorithm, theory, application and future research problems. In practical application, in order to fully learn the information of input data and ensure the function approximation and generalization ability of the system, BLS often sets too many nodes in its hidden layer, and partial nodes are actually unnecessary. The extra nodes increase the amount of computation, which is time-consuming and occupies the storage space, which is not conducive to the practical application of the model. Our study breaks away from the inertial thinking of reducing nodes, and finds another way to reduce redundant information without reducing the number of output hidden layer nodes. This paper studies how to express features to the maximum, and proposes a structure that combines width and depth. The model deeply extracts higher-order and abstract essential features of the original data in a greedy, layer-by-layer manner, and obtains a more compact and meaningful feature representation under the same number of nodes. The construction method proposed in this paper is as follows. The first hidden layer of the network is generated just as the standard BLS. In order to obtain a sparse representation of the input data, sparse autoencoder is used to fine-tune the initial weights to obtain optimal feature nodes, which are transformed into enhancement nodes and then activated by nonlinear functions. The first hidden layer is composed of all feature nodes and enhancement nodes. Taking a page from CNN-BLS [23], Yang used PCA to reduce the dimensions of the features resulting from convolution and max pooling. In this paper, PCA is used as a mapping method to extract the principal components of hidden layer, and they are called compressed nodes. This approach removes some redundant nodes that contribute little to the variance. In the same way as RVFLNN and BLS, the compressed nodes are then transformed into enhancement nodes. Considering the training time and complexity of the model, the number of enhancement nodes is set to the number of nodes deleted by PCA, so that the number of nodes in each hidden layer of GBLS is the same. The second hidden layer is composed of all newly generated compressed nodes and enhancement nodes, and so on. The last hidden layer is connected to the output layer, and the output weights are also obtained by the ridge regression algorithm. This paper refers to this kind of layer-by-layer training scheme as greedy learning, and the proposed method is named greedy BLS (GBLS).

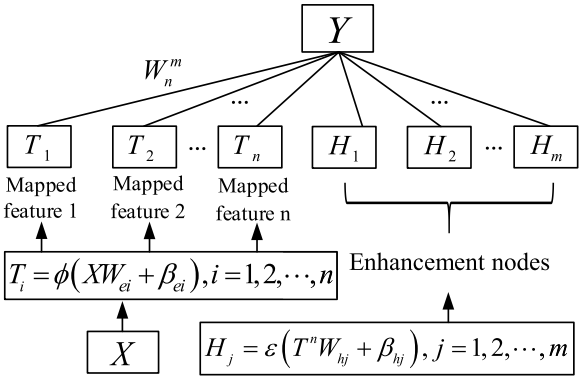


FIGURE 1. Structure of standard BLS.

The paper is organized as follows. In Section II, BLS is briefly reviewed and the specific algorithm of PCA is also shown here. Section III presents the structure and training method of the proposed GBLS. In Section IV, we conduct experiments on MNIST and 10 UCI data sets to verify the proposed method and present the results and analysis. At last, Section V draws the conclusion.

II. PRELIMINARY WORK

In this section we briefly introduce several basic concepts, including standard BLS and the theory of principal component analysis.

A. STANDARD BROAD LEARNING SYSTEM

BLS is a typical forward NN, and its structure draws on the idea of RVFLNN. It has three layers: input layer, hidden layer, and output layer. The structure of standard BLS is shown in Figure 1, it can be quickly updated by expanding the width of the hidden layer.

For supervised learning task, when given the training data set $\{X, Y\}$, BLS first maps the input to n sets of feature

nodes through n feature mapping ϕ, which is the feature mapping function. ϕ_i and ϕ_p can be different functions for $i \neq p$. Without loss of generality, the subscripts of the ith are omitted in this paper. The ith group of feature nodes can be represented as

$$T_i = \phi\left(XW_{ei} + \beta_{ei}\right), i = 1, 2, \cdots, n \tag{1}$$

where $X \in \mathbb{R}^{N\times R}$, $Y \in \mathbb{R}^{N\times M}$, it means that the input data X equips with N samples, each with R dimensions, and M is the dimension of corresponding outputs. Weights W_{ei} and bias terms β_{ei} are randomly generated matrices with the proper dimensions. BLS uses linear transformation by default and set feature map $\phi(x) = x$ to reduce computational complexity. Denote all the feature nodes as

$$T^n \equiv [T_1, T_2, \cdots, T_n] \tag{2}$$

In order to obtain more appropriate and sparse feature nodes, BLS uses a sparse autoencoder to fine-tune the initial random weight matrix W_{ei}, and then use T^n to form m groups of enhancement nodes and perform nonlinear transformations on them. Suppose ε is a nonlinear activation function, and ε_j and ε_q can be different functions for $j \neq q$. Without loss of generality, the subscripts of the jth are omitted in this paper. So, the jth group of enhancement nodes can be represented as

$$H_j = \varepsilon\left(T^n W_{hj} + \beta_{hj}\right), j = 1, 2, \cdots, m \tag{3}$$

where W_{hj} and β_{hj} are randomly generated weights and bias terms connecting the outputs of feature layer to enhancement nodes. BLS uses ε=$tanh(\cdot)$ by default. Denote all the enhancement nodes as

$$H^m \equiv [H_1, H_2, \cdots, H_m] \tag{4}$$

Hence, the hidden layer of the broad model can be expressed as

$$A = [T^n | H^m] \in \mathbb{R}^{N\times L} \tag{5}$$

where L is the total number of nodes in the hidden layer.

The output matrix of BLS can be represented as

$$Y = AW_n^m \tag{6}$$

where W_n^m are the weights connecting the hidden layer and the output layer, and it can be calculated rapidly by the ridge regression approximation of pseudoinverse A^+ using (7). This is a classic convex optimization problem, also known as ridge regression problem.

$$\arg\min_{W_n^m} : \left\|AW_n^m - Y\right\|_2^2 + \frac{\lambda}{2}\left\|W_n^m\right\|_2^2 \tag{7}$$

where Y is given output, and value λ is regularization coefficient, which further restricts the sum of squared weights. The addition of regular l_2 norm regularization can effectively suppress the overfitting of the network. The solution of the optimal problem is formulated as

$$W_n^m = \left(A^T A + \lambda I\right)^{-1} A^T Y \in \mathbb{R}^{L\times M} \tag{8}$$

where I is identity matrix, A^T is the transposed matrix of A. If $\lambda \to \infty$, the solution is strictly limited and tends to 0. If $\lambda = 0$, the inverse problem degenerates into the least square problem, and it is easy to derive the solution of the original pseudo-inverse.

At last, we have

$$W_n^m = A^+ Y \tag{9}$$

where

$$A^+ = \lim_{\lambda\to 0}\left(A^T A + \lambda I\right)^{-1} A^T \tag{10}$$

The above scheme is the default method of BLS to solve the output weights. Especially for some ill-conditioned problems, it is helpful to improve generalization ability of the system. This approach provides a unified solution for solving optimal weights of regression and classification problems.

B. PRINCIPAL COMPONENT ANALYSIS

The goal of PCA is to map high-dimensional data to low-dimensional data through a certain linear projection, and expect to maximize variance of the data in the projected dimension, so as to use fewer dimensions while retaining more dimensions of original data. The solution process of PCA is as follows.

1) Standardize the original data. This step can solve the problem that generalization ability of the model is reduced due to different dimensions between variables. The Z-Score method is used to standardize the mean and standard deviation of the original data, and conversion function is shown in equation (11), the processed data conforms to the standard normal distribution. Then, we can calculate the covariance matrix between the variables of the data.

$$x^* = \frac{x - \bar{x}}{\sigma} \tag{11}$$

where $\bar{x}$ and σ are the mean and standard deviation of the original data, respectively.

2) Solve the eigenvalues and eigenvectors of the covariance matrix. Assuming that the dimension of the original data X is n, define the eigenvalues of the covariance matrix as λ_i, $i = 1, 2, \ldots, n$, arrange them from the largest to the smallest, we get $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. The corresponding eigenvectors are $V_1, V_2, \ldots, V_n$. According to the variance contribution rates, select first k ($k < n$) principal components to obtain the conversion matrix $V = [V_1, V_2, \ldots, V_k]$. So the output matrix is Y=XV, which means each principal component is a linear combination of the original variables. What's more, the variance of ith principal component Y_i is equal to ith eigenvalue of the covariance matrix.
3) Calculate the variance contribution rate. Extract the first k principal components and the cumulative

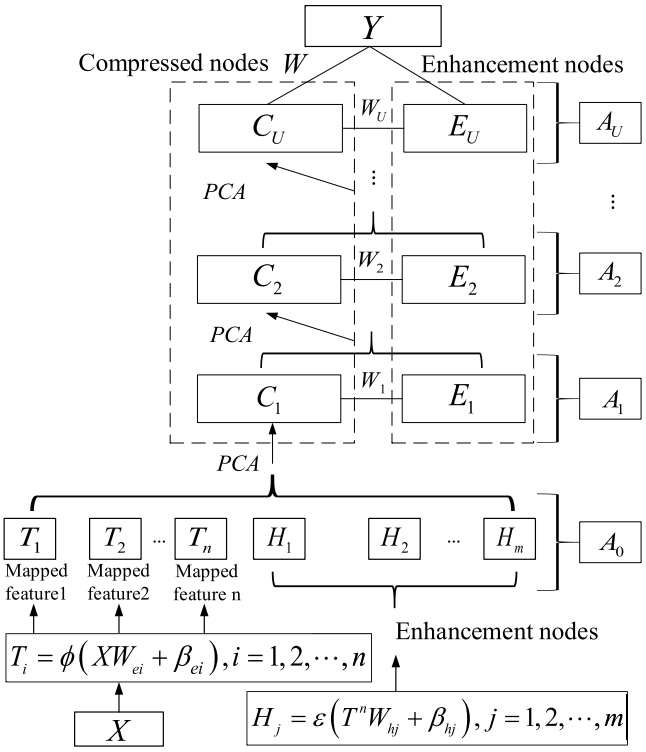


FIGURE 2. Structure of the proposed GBLS.

contribution rate is

$$p = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \tag{12}$$

4) Select the number of principal components. Generally, take the top k principal components whose cumulative variance contribution rate is more than 95% to reasonably explain the original data. The variance of the first principal component is the largest, and the subsequent ones gradually decrease. The corresponding information of the first k principal components contains most of information of the original data.

PCA refers to the formation of new feature sets after orthogonal transformation, then select more important parts of the sub-feature sets to achieve dimensionality reduction. This way is not to choose among the original features, so the linear dimensionality reduction method of PCA retains features of the original data to the greatest extent. Furthermore, these new features are spatially orthogonal, which means that they are uncorrelated.

III. PROPOSED ALGORITHM MODEL

Broadening and deepening are two ways to improve the generalization ability of NNs, and this study attempts to combine the structure of width and depth to explore deep characteristics of data and reduce the redundancy of the model. The horizontal expansion in broad learning can directly change the size of the hidden layer and extract more features of the original data, while increasing the number of layers will increase the degree of feature transformation.

The principle of GBLS is to build a layered model, extract features from the data at the bottom and pass them layer by layer to the upper layer in order to express the input data hierarchically and extract essential features from the data efficiently. Each time a new layer is added, there is no need to train from the beginning, but to continue from the previous layer. In addition, due to avoiding gradient descent, it has extremely fast calculation speed. GBLS inherits characteristics of rapidity of standard BLS, that is, in addition to the output weights, all other weights and biases involved are randomly generated, so GBLS can quickly complete tasks such as classification and regression. The structure of the proposed GBLS is shown in Figure 2.

The first hidden layer, which is the hidden layer of the standard BLS, is expressed as

$$A_0 = [T^n | H^m] \in \mathbb{R}^{N \times L} \tag{13}$$

where generation methods of T^n and H^m have been described in Section II.A and will not be repeated here.

The newly added hidden layer is represented by A_u, where u represents the number of newly added hidden layers, $u = 1, 2, \ldots, U$. All new hidden layers are generated as follows.

Firstly, perform PCA on previous hidden layer A_{u_1} to obtain its conversion matrix U_{u-1}. Standardize the features

of the hidden layer A_0 with equation (11) and its covariance matrix can be expressed as

$$C = A_0 A_0^T \tag{14}$$

Calculate the eigenvalues λ_i of the covariance matrix C, $i = 1, 2, \ldots, n$, sort them from largest to smallest, we have $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, the corresponding eigenvectors are V_1, $V_2, \ldots, V_n$. Calculate the cumulative contribution rate p with equation (12) to extract the first k principal components to obtain the conversion matrix U_{u-1}

$$U_{u-1} = [V_1, V_2, \cdots, V_k] \tag{15}$$

Then the compressed nodes C_u in the latter hidden layer A_u is defined as

$$C_u = A_{u-1} U_{u-1} \in \mathbb{R}^{N \times K} \tag{16}$$

where K ($K < L$) is the number of compressed nodes.

Secondly, transform the compressed nodes into enhancement nodes and activate them with a nonlinear function. For the latter hidden layer A_u, the new enhancement nodes are

$$E_u = \varepsilon (C_u W_u + \beta_u) \in \mathbb{R}^{N \times (L-K)} \tag{17}$$

where ε is nonlinear activation function, W_u and β_u are randomly generated. So we have the latter hidden layer

$$A_u = [C_u | E_u] \in \mathbb{R}^{N \times L} \tag{18}$$

Consequently, the last hidden layer, which is output hidden layer, that is

$$A_U = [C_U | E_U] \in \mathbb{R}^{N \times L} \tag{19}$$

Hence, we have the output of the model

$$Y = A_U W \tag{20}$$

where $W \in R^{L \times M}$ is connecting weight of the model and can be calculated easily by ridge regression algorithm:

$$W = \left(A_U^T A_U + \lambda I \right)^{-1} A_U^T Y \tag{21}$$

where I is identity matrix.

The training steps of the proposed GBLS are shown in detail in algorithm 1. Take two-layer GBLS as an example, the flow chart of the algorithm is illustrated in Figure 3. It is worth noting that only the training set can be dimensionalized and the current conversion matrix is saved for dimensionality reduction of testing set, which is because the latter is unknowable to us until the model is generated, so we cannot use any information about the testing set.

IV. CASES STUDY

In this section, experiments of classification and regression are conducted to demonstrate the proposed model, and comparisons are presented between the mainstream algorithms and the proposed methodology. Table 1 gives a brief overview of the main differences between the proposed GBLS and BLS. We hope that these descriptions will give readers a better understanding of the proposed GBLS, which may be of some help in future research.

Algorithm 1 Greedy Learning - Increment of U hidden layers

Input: training samples X;
Output: W
for $i = 0; i \leq n$ **do**
  Random W_{ei}, β_{ei}
  Calculate $T_i = \phi (X W_{ei} + \beta_{ei}), i = 1, 2, \cdots, n$
end
Set the feature group $T^n \equiv [T_1, T_2, \cdots, T_n]$
for $j = 1; j \leq m$ **do**
  Random W_{hj}, β_{hj}
  Calculate $H_j = \varepsilon \left(T^n W_{hj} + \beta_{hj} \right), j = 1, 2, \cdots, m$
end
Set the enhancement group $H^m \equiv [H_1, H_2, \cdots, H_m]$
Set the first hidden layer $A_0 = [T^n | H^m]$
for $u = 1; u \leq U$ **do**
  Calculate $C = A_0 A_0^T$
  Calculate λ_i and V_i of $C, i = 1, 2, \cdots, n$
  Sort λ_i and its V_i from largest to the smallest
  Calculate $U_{u-1} = [V_1, V_2, \cdots, V_k]$
  Calculate $C_u = A_{u-1} U_{u-1}$
  Random W_u, β_u
  Calculate $E_u = \varepsilon (C_u W_u + \beta_u)$
  Calculate the u_{th} hidden layer $A_u = [C_u | E_u]$
end
Calculate W by Eq.(21)

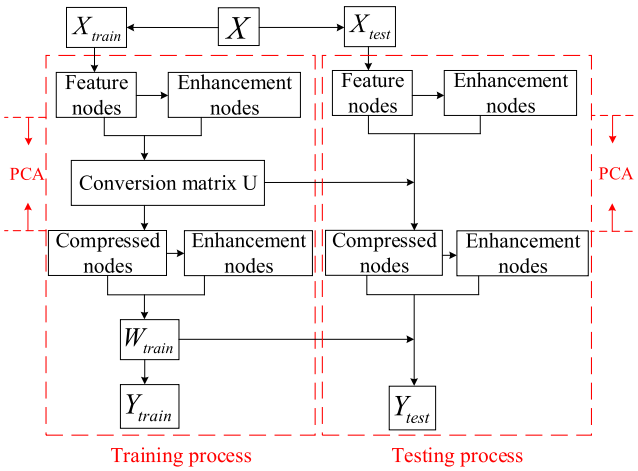


FIGURE 3. Flow chart of the algorithm.

A. CLASSIFICATION

To verify the effectiveness of the proposed GBLS, classification experiments are performed on the MNIST data [24], its typical image is shown in Figure 4.

This dataset consists of black and white handwritten digital images from 0 to 9, and the size of each digital image is 28×28. The training set has 60, 000 images and the testing set has 10, 000 images. In addition, in order to prove the efficiency, the proposed model is compared with some mainstream methods.

For all the experiments, the structure of GBLS in the first hidden layer consists of 10×10 feature nodes and $1 \times N$

TABLE 1. Brief descriptions of BLS and GBLS.

Characteristic	BLS	GBLS
Training method	Broad learning	Greedy learning
Structure	One hidden layer	Multiple hidden layers
Weights(except output weights) and biases	All randomly generated	All randomly generated
Manually adjusted parameters	λ	λ, p
Redundant nodes	More	Less
Training time	Less	More but acceptable
Size of each hidden layer	$\vdots^{N\times L}$	$\vdots^{N\times L}$
Generalization performance	Good (in most cases)	Better

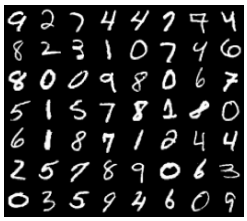


FIGURE 4. MNIST data set.

TABLE 2. Classification accuracies.

Enhancement nodes	Standard BLS Accuracy/%	Two-layer GBLS Accuracy/%	Three-layer GBLS Accuracy/%
50	90.95	91.57	91.60
400	95.67	96.01	96.09
2000	97.7	97.99	98.04
6000	98.47	98.61	98.63
9500	98. 59	98.75	98.77
10500	98. 59	**98. 81**	**98.82**
11000	**98. 74**	98.76	98.77
12000	98.67	98.76	98.77

enhancement nodes, where N is the number of enhancement nodes. What's more, the last 5 groups are replicated experiments in [7] and the corresponding classification results of the original BLS are cited for fairness of comparison. In GBLS, all weights and biases corresponding to enhancement nodes are randomly generated, which are drawn from standard uniform distributions on interval [-1,1]. The value of regularization parameter λ in ridge regression algorithm is set as 10^{-8}. In this paper, our real goal is to remove the redundant nodes in the hidden layer and retain more information of the original data as much as possible. Dimensionality reduction is only an incidental result. If a low contribution rate is set, large numbers of nodes will be deleted, which may contain useful information for the model, so the interval of variance contribution rate is set to (0.99,1). The searching step is a user-set parameter according to the actual problem. Eight groups of different structures are selected, the optimal test accuracies corresponding to different models under different structures are shown in Table 2 and the best results corresponding to each model are expressed in bold.

Results show that features of the original BLS in its hidden layer contain a lot of redundancy, the proposed GBLS, as an improvement plan of the standard BLS, greatly exploits the potential of the nodes. The upper limit of the layer number selected in this paper is three, for MNIST, three-layer GBLS has almost approached the limit of the recognition rate. Let F_0, M_0, C_u and E_u represent feature nodes, mapping groups, compressed nodes and enhancement nodes respectively, where u is the number of hidden layers. Table 3 shows the structure corresponding to the optimal result 98.82% and Table 4 presents the training times and results of this structure for each greedy learning.

TABLE 3. Structure of best model.

Data	F_0	M_0	E_0	C_1	E_1	C_2	E_2
MNIST	10	10	10500	10433	167	10587	13

Under the same feature nodes, when only 9500 enhancement nodes are used, GBLS has obtained better generalization ability than standard BLS using 11000 enhancement nodes. The training time used for the optimal result 98.82% is 237.26s, compared with 59.87s for the best result 98.74% of BLS, the GBLS model increases some acceptable training time, which to a certain extent improves the performance of the model while dealing with the structural redundancy.

In addition, we also compare GBLS with existing mainstream methods, including BLS, deep Boltzmann machines (DBM) [25], deep belief nets (DBN) [26], stacked auto encoders (SAE) [27], another version of stacked autoencoder (SDA) [28], multilayer perceptron-based methods (MLP) [29], fuzzy restricted boltzmann machine (FRBM) [30]. The classification results of above comparative experiments are quoted from [7] and all these experiments are tested using MATLAB software platform on a laptop equipped with Intel-i7 2.4GHz CPU, 16GB memory. All classification accuracies are given in Table 5. It is worth noting that GBLS and the duplicate experiment of BLS are done on a computer equipped with Intel(R) Xeon(R) CPU E5-2678 v3 @ 2.50GHz 2.50 GHz (2 processors), 32GB memory, and these two experiments are indicated by a special superscript *.

Except for BLS, all the other comparison methods mentioned above are deep structures, hyperparameters involved are adjusted by BP algorithm, where the initial learning rate is set to 0.1, the decay rate of each learning iteration is set to 0.95, and the remaining more detailed parameters can be verified in [11]. In BLS and GBLS, the hyperparameter

TABLE 4. Snapshot results of the best structure.

Model	Testing Accuracy/%	Each Additional Training Time/s	Accumulative Training Time/s	Each Additional Testing Time/s	Accumulative Testing Time/s
Standard BLS	98.59	48.59	48.59	1.58	1.58
Two-layer GBLS	98.81	93.86	142.45	11.94	13.52
Three-layer GBLS	98.82	94.81	237.26	11.79	25.31

TABLE 5. Accuracy corresponding to different methods.

Method	Accuracy/%	Training time/s
DBM	99.05	121455.69
DBN	98.87	53219.77
SAE	98.60	36448.4
SDA	98.72	37786.03
MLP	97.39	21468.12
FRBM	97.44	577.8
GBLS*	98.82	237.26
BLS	98.74	78.68
BLS*	98.74	59.87

corresponding to ridge regression is set to 10^{-8}. The hyperbolic tangent function is chosen to activate the enhancement nodes. Furthermore, except that the output weights are calculated by ridge regression, all other weights and biases used are randomly generated, which are drawn from the standard uniform distributions on interval [-1,1]. In particular, the input weights are first randomly generated in this way, then use the advantages of the sparse encoder to fine-tune initial weights to obtain better feature nodes.

As shown in Table 5, although 98.82% is not the best one, in fact, in terms of network complexity and training time, the performance of GBLS is much better than other networks. Compared with the use of high-performance computers in deep structures that take hours or even days to go through hundreds of iterations, GBLS can be easily constructed in a few minutes.

B. REGRESSION

In the following experiments, 10 real-world regression data sets are chosen from the University of California, Irvine (UCI) database [31] in the categories of small size, medium size and large size. The details of these data sets are put up in Table 6. We selected the optimal results for each data set from 10 trials. All relevant parameters are provided in Table 7.

Table 8 gives the best results for different models. In addition, the optimal testing results for each dataset are indicated in bold.

Typical models like SVM [32], LSSVM and ELM are compared to GBLS. The corresponding parameters and results of the three typical models and BLS can be checked in [8]. For fair comparison, we also perform a grid search from [1,10]×[1,30]×[1,200] to determine the numbers of feature nodes, mapping groups and enhancement nodes, and the searching step is set to 1. Root mean square error (RMSE) [33] is selected as the performance evaluation indice to measure the prediction errors of different models.

TABLE 6. Details of data sets.

Data	Train sample	Test sample	Input features
Abalone	2784	1393	8
Basketball	64	32	4
Bodyfat	168	84	14
Cleveland	202	101	13
Housing	337	169	13
Mortgage	699	350	15
Pyrim	49	25	27
Quake	1452	726	3
Strike	416	209	6
Weather Izmir	974	487	9

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_i^*)^2} \tag{22}$$

where y_i is the actual value, y_i^* is the predicted value and N is the number of samples.

Experiments on 10 function approximation data sets show that BLS outperforms SVM, LSSVM and ELM in data prediction. However, the fitting results of some data sets are not satisfactory, after calculation, for housing, mortgage, cleveland and pyrim, the optimal testing results of GBLS are 5.33%, 6.98%, 11.51% and 34.08% higher than those of the original BLS, and the improvement effect of the remaining data sets is less than 5%.

On the one hand, although the improvement effects of some data sets are not obvious, the fitting result of GBLS for these data sets is already the best conclusion in the existing literatures. On the other hand, as can be seen from Table 7, the original hidden layer of BLS contains many redundant nodes, which is better improved by the greedy learning proposed in this paper and this effect is more obvious in the third layer.

From the above experimental results, including the classification problem, it can be seen that the performance of three-layer GBLS is better than that of two-layer GBLS, except for the dataset of quake. The possible reason is that the conversion matrix extracted by PCA is extracted from the training set, and those features that contribute less to the variance may contain important information that may be useful for the testing set.

TABLE 7. Structures of different models.

Data	BLS			Two-layer GBLS					Three-layer GBLS						
	F_0	M_0	E_0	F_0	M_0	E_0	C_1	E_1	F_0	M_0	E_0	C_1	E_1	C_2	E_2
Abalone	5	6	41	10	3	44	54	20	8	10	67	75	72	76	71
Basketball	6	7	4	2	26	4	13	43	7	1	6	11	2	12	1
Bodyfat	6	5	13	2	23	35	79	2	4	9	18	50	4	53	1
Cleveland	1	10	9	1	9	1	9	1	1	9	1	9	1	9	1
Housing	5	29	50	6	9	14	28	40	8	8	17	32	49	80	1
Mortgage	9	4	135	8	21	48	64	152	8	21	48	64	152	215	1
Pyrim	3	7	2	2	7	4	11	7	2	20	55	48	47	49	46
Quake	10	2	6	1	15	1	5	11	1	8	14	8	14	21	1
Strike	9	11	30	5	16	1	11	70	5	16	1	11	70	79	2
Weather	4	3	87	5	16	34	38	76	7	12	64	74	74	147	1

TABLE 8. RMSE results on data sets.

Data	SVM		LSSVM		ELM		BLS		Two-layer GBLS		Three-layer GBLS	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Abalone	0.0748	0.0773	0.0717	0.0756	0.0756	0.0777	0.0737	0.0754	0.0722	0.0749	0.0720	**0.0747**
Basketball	0.0804	0.0767	0.0826	0.0744	0.0801	0.0719	0.0834	0.0659	0.0795	0.0636	0.0803	**0.0628**
Bodyfat	0.0038	0.0049	0.0027	0.0038	0.0042	0.0033	0.0060	0.0030	0.0021	0.0029	0.0054	**0.0027**
Cleveland	0.1032	0.1514	0.1039	0.1256	0.1038	0.1281	0.1040	0.1199	0.1075	0.1061	0.1074	**0.1061**
Housing	0.0286	0.0792	0.0282	0.0780	0.0371	0.0762	0.0571	0.0751	0.0612	0.0729	0.0535	**0.0711**
Mortgage	0.0019	0.0046	0.0026	0.0053	0.0042	0.0057	0.0031	0.0043	0.0025	0.0041	0.0025	**0.0040**
Pyrim	0.0130	0.1549	0.0255	0.1133	0.0767	0.1376	0.0420	0.0578	0.0441	0.0422	0.0093	**0.0381**
Quake	0.1603	0.2029	0.1576	0.1733	0.1716	0.1729	0.1703	0.1718	0.1708	**0.1712**	0.1700	0.1713
Strike	0.0584	0.1053	0.0463	0.1007	0.0592	0.1043	0.0503	0.1001	0.0461	0.0967	0.0457	**0.0964**
Weather	0.0166	0.0190	0.0161	0.0191	0.0158	0.0191	0.0165	0.0188	0.0163	0.0187	0.0157	**0.0186**

V. CONCLUSION

The proposed GBLS in this paper uses a greedy layer-by-layer scheme for training, constructs a structure that combines width and depth, removes most of the redundancy and extract deep features to supplement it, transforms low-level features into higher-level abstract features, which significantly improves the model's feature expression ability and generalization performance. Since all weights and biases involved are generated randomly, so the construction of the model is extremely simple and fast. Actually, the establishment of the system is based on the ideas of RVFLNN and BLS. The RVFLNN proposed by Pao and Takefuji has confirmed that weights from input layer to enhancement layer can be randomly generated. The construction of BLS is based on theory of RVFLNN, which once again verified the correctness of this theory and laid a theoretical foundation for GBLS. The performance on MNIST and 10 UCI regression data sets confirms the effectiveness and efficiency of GBLS.

Taking into account the training speed and generalization ability, the structure of GBLS, is significantly better than the existing structure that simply increases the width or depth. Although some training time is increased while enhancing the fitting ability, it is insignificant compared with hardly tolerable repeated parameter adjustment and lengthy training process in deep learning. It is worth mentioning that each hidden layer sets the same number of nodes, and the output hidden layer of GBLS is obtained through greedy learning, so almost all nodes are effective nodes. Therefore, the biggest advantage of GBLS is that it makes the best use of computing resources, rather than wasting them on plenty of redundant nodes. It is foreseeable that this kind of greedy training method could extend to the processing of hidden layers in other neural networks, and this is also our next research work.

ACKNOWLEDGMENT

(Yubo Tian is co-first author.)

REFERENCES

[1] K. Yue, F. Xu, and J. Yu, "Shallow and wide fractional max-pooling network for image classification," *Neural Comput. Appl.*, vol. 31, no. 2, pp. 409–419, Feb. 2019.

[2] X. He and L. Deng, "Deep learning for image-to-text generation: A technical overview," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 109–116, Nov. 2017, doi: 10.1109/MSP.2017.2741510.

[3] A. Ruiz-Garcia, M. Elshaw, A. Altahhan, and V. Palade, "A hybrid deep learning neural approach for emotion recognition from facial expressions for socially assistive robots," *Neural Comput. Appl.*, vol. 29, no. 7, pp. 359–373, Apr. 2018.

[4] W. Hou, X. Gao, D. Tao, and X. Li, "Blind image quality assessment via deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1275–1286, Jun. 2015, doi: 10.1109/TNNLS.2014.2336852.

[5] S. Ning, Y. He, L. Yuan, Y. Huang, S. Wang, T. Cheng, and Y. Sui, "Wireless channel scene recognition method based on an autocorrelation function and deep learning," *IEEE Access*, vol. 8, pp. 226324–226336, 2020, doi: 10.1109/ACCESS.2020.3044167.

[6] G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," 2016, *arXiv:1610.01644*. [Online]. Available: https://arxiv.org/abs/1610.01644

[7] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018, doi: 10.1109/TNNLS.2017.2716952.

[8] C. L. P. Chen, Z. Liu, and S. Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1191–1204, Apr. 2019, doi: 10.1109/TNNLS.2018.2866622.

[9] Z. Liu and C. L. P. Chen, "Broad learning system: Structural extensions on single-layer and multi-layer neural networks," in *Proc. Int. Conf. Secur., Pattern Anal., Cybern. (SPAC)*, Shenzhen, China, Dec. 2017, pp. 136–141, doi: 10.1109/SPAC.2017.8304264.

[10] P. Strumiłło and W. Kamiński, "Radial basis function neural networks: Theory and applications," in *Neural Networks and Soft Computing*. Heidelberg, Germany: Physica, 2003, pp. 107–119.

[11] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016, doi: 10.1109/TNNLS.2015.2424995.

[12] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: Theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, May 1992, doi: 10.1109/2.144401.

[13] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995, doi: 10.1109/72.471375.

[14] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, Apr. 1994.

[15] Y. L. Cun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2, 1989, pp. 396–404.

[16] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, Feb. 1970.

[17] C. L. P. Chen and J. Z. Wan, "A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction," *IEEE Trans. Syst., Man Cybern., B, Cybern.*, vol. 29, no. 1, pp. 62–72, Feb. 1999, doi: 10.1109/3477.740166.

[18] Z. Liu, J. Zhou, and C. L. P. Chen, "Broad learning system: Feature extraction based on K-means clustering algorithm," in *Proc. 4th Int. Conf. Inf., Cybern. Comput. Social Syst. (ICCSS)*, Dalian, China, Jul. 2017, pp. 683–687, doi: 10.1109/ICCSS.2017.8091501.

[19] J. Jin, C. L. P. Chen, and Y. Li, "Robust broad learning system for uncertain data modeling," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2018, pp. 3524–3529, doi: 10.1109/SMC.2018.00596.

[20] M. Xu, M. Han, C. L. P. Chen, and T. Qiu, "Recurrent broad learning systems for time series prediction," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1405–1417, Apr. 2020, doi: 10.1109/TCYB.2018.2863020.

[21] H. Zhao, J. Zheng, W. Deng, and Y. Song, "Semi-supervised broad learning system based on manifold regularization and broad network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 3, pp. 983–994, Mar. 2020, doi: 10.1109/TCSI.2019.2959886.

[22] X. Gong, T. Zhang, C. L. P. Chen, and Z. Liu, "Research review for broad learning system: Algorithms, theory, and applications," *IEEE Trans. Cybern.*, early access, Mar. 17, 2021, doi: 10.1109/TCYB.2021.3061094.

[23] F. Yang, "A CNN-based broad learning system," in *Proc. IEEE 4th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2018, pp. 2105–2109, doi: 10.1109/CompComm.2018.8780984.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[25] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, Clearwater Beach, FL, USA, Jul. 2009, pp. 448–455.

[26] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[27] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[28] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 1096–1103.

[29] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). Secaucus, NJ, USA: Springer-Verlag, 2006.

[30] S. Feng and C. L. P. Chen, "A fuzzy restricted Boltzmann machine: Novel learning algorithms based on the crisp possibilistic mean value of fuzzy numbers," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 1, pp. 117–130, Feb. 2018, doi: 10.1109/TFUZZ.2016.2639064.

[31] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, Tech. Rep., 1998. [Online]. Available: https://archive.ics.uci.edu/ml/datasets.php

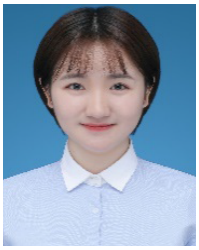
[32] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.

[33] J. S. Armstrong and F. Collopy, "Error measures for generalizing about forecasting methods: Empirical comparisons," *Int. J. Forecasting*, vol. 8, no. 1, pp. 69–80, Jun. 1992.

WEITONG DING was born in Suzhou, China, in 1997. He received the B.Tech. degree from the North China Institute of Science and Technology. He is currently pursuing the master's degree with the Jiangsu University of Science and Technology. His research interests include machine learning and its application in antenna optimization.

YUBO TIAN was born in Tieling, Liaoning, China, in 1971. He received the Ph.D. degree in radio physics from the Department of Electronic Science and Engineering, Nanjing University, Nanjing, China. From 1997 to 2004, he was with the Department of Information Engineering, Shenyang University, Shenyang, China. From 2005 to 2020, he was with the School of Electronics and Information, Jiangsu University of Science and Technology, Zhenjiang, China, where he has been a Full Professor and the Vice Dean, since 2011. He was a Visiting Scholar with the University of California Los Angeles, in 2009, and Griffith University, in 2015. He is currently with the School of Information and Communication Engineering, Guangzhou Maritime University, Guangzhou, China. He has authored and coauthored more than 100 journal articles and three books. He holds more than 20 filed/granted China patents. His current research interests include machine learning methods and their applications in electronics and electromagnetics.

SHUDAN HAN was born in Yancheng, China, in 1997. She received the B.Tech. degree from the Suzhou University of Science and Technology. She is currently pursuing the master's degree with the Jiangsu University of Science and Technology. Her research interests include applications of intelligent optimization algorithms and deep learning methods to the design of electromagnetic microwave devices.



HUINING YUAN was born in Anhui, China, in 1997. She received the B.Tech. degree from West Anhui University. She is currently pursuing the master's degree with the Jiangsu University of Science and Technology. Her research interests include machine learning and biological signal detection.