Stacked Broad Learning System: From Incremental Flatted Structure to Deep Model

Zhulin Liu, C. L. Philip Chen, *Fellow, IEEE*, Shuang Feng, Qiying Feng, and Tong Zhang, *Member, IEEE*,

***Abstract*—The broad learning system (BLS) has been proved to be effective and efficient lately. In this article, several deep variants of BLS are reviewed, and a new adaptive incremental structure, Stacked BLS, is proposed. The proposed model is a novel incremental stacking of BLS. This invariant inherits the efficiency and effectiveness of BLS that the structure and weights of lower layers of BLS are fixed when the new blocks are added. The incremental stacking algorithm computes not only the connection weights between the newly stacking blocks but also the connection weights of the enhancement nodes within the BLS block. The Stacked BLS is considered as the increment of "layers" and "neurons" dynamically during the training for multilayer neural networks. The proposed architecture along with the training algorithms that utilizes the residual characteristic is very versatile in comparison with traditional fixed architecture. Finally, experimental results on UCI datasets, MNIST dataset, NORB dataset, CIFAR-10 dataset, SVHN dataset, and CIFAR-100 dataset indicate that the proposed method outperforms the selected state-of-the-art methods on both accuracy and training speed, such as deep residual networks. The results also imply that the proposed structure could highly reduce the number of nodes and the training time of the original BLS in the classification task of some datasets.**

***Index Terms*—Broad learning system (BLS), deep learning, functional link neural networks, nonlinear function approximation, universal approximation.**

Manuscript received November 26, 2020; accepted December 1, 2020. Date of publication December 25, 2020; date of current version January 12, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1703600 and Grant 2019YFA0706200; in part by the National Natural Science Foundation of China under Grant 62006079, Grant 61702195, Grant 61751202, Grant U1813203, Grant U1801262, and Grant 61751205; in part by the China Postdoctoral Science Foundation under Grant 2020TQ0105; in part by the Science and Technology Major Project of Guangzhou under Grant 202007030006; in part by the Science and Technology Development Fund, Macau SAR, under Grant 0119/2018/A3; in part by the Multiyear Research Grants of University of Macau; in part by the Guangdong–Hong Kong–Macao Greater Bay Area Center for Brain Science and Brain-Inspired Intelligence Fund under Grant 2019016; and in part by the Foundation for Distinguished Young Talents in Higher Education of Guangdong, China, under Grant 2018KQNCX324. This article was recommended by Associate Editor J. Sarangapani. *(Corresponding author: C. L. Philip Chen.)*

Zhulin Liu and Qiying Feng are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, China (e-mail: zhulinlau@gmail.com; qiyingvon@gmail.com).

C. L. Philip Chen is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, China, and also with the Faculty of Science and Technology, University of Macau, Macau, China (e-mail: philip.chen@ieee.org).

Shuang Feng is with the School of Applied Mathematics, Beijing Normal University, Zhuhai 519087, China (e-mail: fengshuang@bnuz.edu.cn).

Tong Zhang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: tony@scut.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2020.3043147.

Digital Object Identifier 10.1109/TSMC.2020.3043147

I. Introduction

WITH the development of intelligence technology, a great number of machine learning algorithms are proposed for adapting the huge requirement of large-scale data processing [1]–[4]. Many of them have already achieved breakthroughs in various applications, for example, the deep learning algorithms in generative learning and in discriminative learning. For the discriminative learning algorithms and models, the typical one is the convolution neural networks (CNNs) [5] and its deep model and its variations. These include deep residual networks (ResNet-34) [6]–[8] and dense networks [9], [10]. For the generative learning, the representative learning methods are the deep belief networks (DBNs) [11], [12], and the deep Boltzmann machines (DBMs) [13]. All these deep methods and their variants have become significant components of machine learning. They have also succeeded in pattern recognition, image recognition, speech recognition, and video processing applications, and exhibited outstanding performance.

Despite the powerful capacity of the deep structures, their training processing is time consuming due to the huge number of hyperparameters or deep structures. Furthermore, this complexity greatly increases the difficulty of analyzing a deep structure theoretically, especially when the higher accuracy is required such that deeper structure and more neural units are forced to increase.

Due to the high volume, high velocity, and/or high variety [14] of the large-scale dataset, single block with fixed learning algorithms could not meet the requirements of tasks in practice. Therefore, multiple alternative learning algorithms associated with a number of parameters are chosen to establish a valid model.

Lately, an efficient and effective discriminative learning algorithm named the broad learning system (BLS) and its variants are proposed in [15] and [16]. BLS takes the inputs and generates feature mapping nodes and enhancement nodes to connect with the output layer. Besides, the BLS can update the system (or relearn) in an incremental way when necessary. The designed BLS is also able to expand the network by the increment of the feature nodes, enhancement nodes, and input patterns efficiently. Consequently, the structure of the BLS is extremely suitable for remodeling and learning various applications [17]–[23]. The mathematical proof of the universal approximation properties is also offered in [16], which implies that the BLS is a good nonlinear function approximator.

Although the BLS has been accomplishing in various applications, the more accurate blocks algorithm is preferred in applications. Among them, its combinations with deep structures perform obvious advantages from the past few years. In this article, we offer a comprehensive review on such kind of models. Furturemore, a novel expansion is followed by combining several BLS blocks with the *Stacking* rule. Specifically, the new system is built in two repeated steps, i.e., generating the BLS blocks, and then stacking the next BLS blocks on top of the previous one. From this point of view, the designed method is expanded not only in broad sense but also stacked in deep if needed. Generally, the computational cost of the new variant of BLS is still low since the lower bottomed BLS blocks are fixed when stacking new block.

In this proposed variant of BLS, the residual of the lower blocks is set as the desired output data for the new block. In fact, the concept of residual we use here is deeply embedded into the original BLS, which could degenerate to an approximator for the residual of the input and the desired output.

Finally, the proposed deeply stacked BLS is evaluated by several popular datasets for regression and classification. The experimental results report that the stacked BLS could achieve the highest testing accuracy in most of the cases.

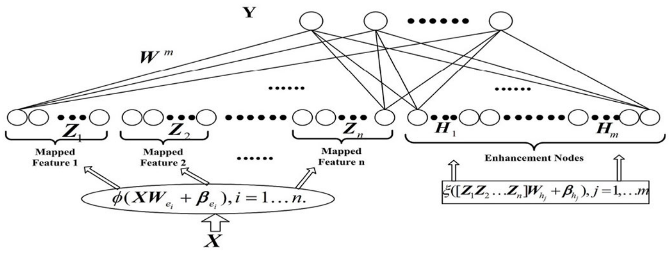
The remainder of this article is organized as follows. First, the BLS with its incremental algorithms, and the fast review of its variants with deep structures are summarized. Second, the proposed expansion (Stacked BLS) is introduced, as well as its incremental algorithms. Third, discussions and analysis about the concept of residual within the BLS are offered. In Section V, experiments are carried out to compare the performance of the Stacked BLS with other state-of-the-art methods. At last, conclusions and further discussions on Stacked BLS are given.

II. Classical Broad Learning System and Its Deep Variants

A. Brief Introduction of BLS

In [15], the BLS offers an alternative frame for constructing deep networks based on the idea of Random Vector Functional Link Neural Networks (RVFLNN) [24] and incremental learning paradigm in [25]. In [26], an algorithm derived by the gradient descent of a cost function with several regularization is introduced. In [16], a typical structural variant of the BLS, which is characterized with the cascade of convolution feature mapping nodes (CCFBLS) is proposed. Detailed structures of the original BLS could be checked in Fig. 1. Next, mathematical formulations are offered.

Suppose that the training samples are $\{\boldsymbol{x}_i | \boldsymbol{x}_i \in \mathbb{R}^M\}$, and $\{\boldsymbol{y}_i | \boldsymbol{y}_i \in \mathbb{R}^C, i = 1, \dots, N\}$ is the associated labels. If the designed BLS consists of n groups of feature nodes and m groups of enhancement nodes, the typical structure could be constructed as follows.



Y
W^m
Z_1
Z_2
Z_n
H_1
H_m
Mapped Feature 1
Mapped Feature 2
Mapped Feature n
Enhancement Nodes
$\phi(XW_{e_i} + \beta_{e_i}), i = 1 \dots n.$
$\xi([Z_1 Z_2 \dots Z_n]W_{h_j} + \beta_{h_j}), j = 1, \dots m$
X

Fig. 1. Illustration of a typical BLS.

First, the ith ($i = 1, \dots, n$) group of feature nodes generated by the feature mapping ϕ_i is obtained by

$$\boldsymbol{Z}_i \triangleq \phi_i(\boldsymbol{X}\boldsymbol{W}_{e_i} + \boldsymbol{\beta}_{e_i}), i = 1, \dots, n \tag{1}$$

where the weights within the feature nodes, such as $\boldsymbol{W}_{e_i}$ and $\boldsymbol{\beta}_{e_i}$, are random samples of some given distributions. Collection of all the n groups is denoted as $\boldsymbol{Z}^n \triangleq [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \dots, \boldsymbol{Z}_n]$.

Second, the jth ($j = 1, \dots, m$) group of enhancement nodes is generated by the nonlinear activation function ξ_j, which is

$$\boldsymbol{H}_j \triangleq \xi_j(\boldsymbol{Z}^n \boldsymbol{W}_{h_j} + \boldsymbol{\beta}_{h_j}), j = 1, 2, \dots, m. \tag{2}$$

Similarly, the weights $\boldsymbol{W}_{h_i}$ and $\boldsymbol{\beta}_{h_i}$ within the enhancements are random samples drawn from some certain distribution. The total enhancement nodes are denoted as $\boldsymbol{H}^m \triangleq [\boldsymbol{H}_1, \boldsymbol{H}_2, \dots, \boldsymbol{H}_m]$.

Finally, the collections of n groups of feature nodes and m groups of enhancement nodes are fed into the output layer to construct a desired result $\boldsymbol{Y}$. The consequent equation is

$$\boldsymbol{Y} \triangleq [\boldsymbol{Z}^n, \boldsymbol{H}^m]\boldsymbol{W}^m \tag{3}$$

where $\boldsymbol{W}^m$ is the weights of the output layer.

In [15], the above weights are obtained by the pseudo inverse of matrix $[\boldsymbol{Z}^n, \boldsymbol{H}^m]$, i.e., $\boldsymbol{W}^m = [\boldsymbol{Z}^n, \boldsymbol{H}^m]^+ \boldsymbol{Y}$. This solution could also be guaranteed by the theory of ridge regression approximation, whose cost function is modified with the l_2-norm regularization.

Another advantage of BLS is that extra feature nodes and enhancement nodes could be added to the system dynamically if necessary. Most of all, the retrain procedure of the whole system is avoided [15]. The structure of all increments is shown in Fig. 2.

The additional p enhancement nodes, which form the $(m+1)$th group of enhancement nodes, are denoted as $\xi_{m+1}(\boldsymbol{Z}^n \boldsymbol{W}_{h_{m+1}} + \boldsymbol{\beta}_{h_{m+1}})$. Similarly, the random weights $\boldsymbol{W}_{h_{m+1}}$ and $\boldsymbol{\beta}_{h_{m+1}}$ are samples of some given distributions. To update the output weights $\boldsymbol{W}^{m+1}$ from $\boldsymbol{W}^m$, the dynamic solution could be calculated by

$$\boldsymbol{W}^{m+1} \triangleq (\boldsymbol{A}^{m+1})^+ \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{W}^m - \boldsymbol{D}\boldsymbol{B}^T\boldsymbol{Y} \\ \boldsymbol{B}^T\boldsymbol{Y} \end{bmatrix} \tag{4}$$

where

$$\begin{aligned} \boldsymbol{A}^m &\triangleq [\boldsymbol{Z}^n, \boldsymbol{H}^m] \\ \boldsymbol{A}^{m+1} &\triangleq [\boldsymbol{A}^m, \xi_{m+1}(\boldsymbol{Z}^n \boldsymbol{W}_{h_{m+1}} + \boldsymbol{\beta}_{h_{m+1}})] \\ (\boldsymbol{A}^{m+1})^+ &= \begin{bmatrix} (\boldsymbol{A}^m)^+ - \boldsymbol{D}\boldsymbol{B}^T \\ \boldsymbol{B}^T \end{bmatrix} \end{aligned} \tag{5}$$

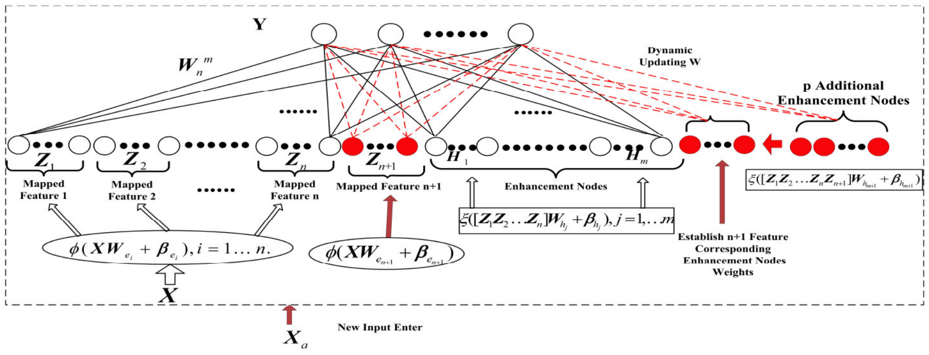

Y
W_n^m
Dynamic Updating W
p Additional Enhancement Nodes
Z_1 Mapped Feature 1
Z_2 Mapped Feature 2
Z_n Mapped Feature n
Z_{n+1} Mapped Feature n+1
H_1 H_m Enhancement Nodes
$\xi([Z_1Z_2\ldots Z_nZ_{n+1}]W_{h_{m+1}}+\beta_{h_{m+1}})$
$\xi([Z_1Z_2\ldots Z_n]W_{h_j}+\beta_{h_j}), j=1,\ldots m$
$\phi(XW_{e_i}+\beta_{e_i}), i=1\ldots n.$
$\phi(XW_{e_{n+1}}+\beta_{e_{n+1}})$
X
Establish n+1 Feature Corresponding Enhancement Nodes Weights
X_a
New Input Enter

Fig. 2. BLS with increment of input variables, feature nodes, and enhancement nodes.

and

$$\begin{aligned}\boldsymbol{B}^T &= \begin{cases}\boldsymbol{C}^+, & \text{if } \boldsymbol{C}\neq 0\\ (\mathbf{1}+\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{D}^T(\boldsymbol{A}^m)^+, & \text{if } \boldsymbol{C}=0\end{cases}\\ \boldsymbol{C} &= \xi_{m+1}(\boldsymbol{Z}^n\boldsymbol{W}_{h_{m+1}}+\boldsymbol{\beta}_{h_{m+1}})-\boldsymbol{A}^m\boldsymbol{D}\\ \boldsymbol{D} &= (\boldsymbol{A}^m)^+\xi_{m+1}(\boldsymbol{Z}^n\boldsymbol{W}_{h_{m+1}}+\boldsymbol{\beta}_{h_{m+1}}).\end{aligned} \quad (6)$$

Therefore, the BLS is incrementally updated without the retraining of the whole system.

B. Deep Variants for BLS

So far, BLS and its deep structure variants have been widely used in various applications. We only display a limited number of variants that is characterized with deep structures in this section.

1) Cascaded BLS: Due to its novel structure, BLS usually enhances the model performance by increasing the number of hidden layer nodes. Here are some representative variants. In recent years, many studies have focused on improving the compactness of the BLS framework, which provide reliable suggestions for reducing the number of BLS parameters and calculation time.

Typically, in [16], the framework of BLS variants that characterizes with the cascade of feature mapping nodes or enhancement nodes (such as CFEBLS) is introduced. Generally speaking, such variants can improve their performance on large datasets by adding functional nodes or enhancing the cascading degree of nodes.

In addition, Zhang *et al.* [26] proposed a novel modified BLS with cascaded enhancement nodes with dense connections (CEBLS dense). The input of the first enhanced node of each module consists of all function nodes and the last enhanced node of the previous module. The last node of each module is sent to the outputs in order to effectively avoid redundant information and overfitting problem. Compared to the original BLS, CEBLS-dense can lower training time in the MNIST and Fashion-MNIST datasets by 43% and 48%, respectively.

2) Recurrent and Gated BLS for Sequential Data: In [16], two possible changes for embedding the recurrent neural network into the BLS are considered. This makes the model have the ability to handle sequential signals. First, two cascaded of feature map nodes can be modified to a structure like the recursive system, called recursive feature nodes. It can learn sequential knowledge from data and expressed as

$$\mathbf{Z}_k = \phi(\mathbf{Z}_{k-1}\mathbf{W}_{e_k}+\mathbf{X}\mathbf{W}_{z_k}+\boldsymbol{\beta}_{e_i}),\ \ k=1,2,\ldots,n. \quad (7)$$

Here, the matrices $\mathbf{W}_{z_k}$, $\mathbf{W}_{e_k}$, and $\boldsymbol{\beta}_{e_k}$ are randomly generated. More specifically, the value of each $\mathbf{Z}_k$ influences the previous value $\mathbf{Z}_{k-1}$ and the input sample $\mathbf{X}$ at the same time with recursive feature mapping nodes.

To be continued, Recurrent-BLS and Gated-BLS are proposed by Du *et al.* [27] and they are applied to text classification in the field of natural language processing in actual experiments. Experiments reveal that a well-designed framework can learn the importance of information and text in the same time series, thereby obtaining better results in terms of accuracy and training time.

Another variant of [16] is that the cascading enhancement node is transformed into a recursive so that it is able to capture the dynamic characteristics of the data and is called a recursive enhancement node. The formula is expressed as

$$\mathbf{H}_k = \xi(\mathbf{H}_{j-1}\mathbf{W}_{h_j}+\mathbf{Z}^n\mathbf{W}_{z_j}+\boldsymbol{\beta}_{h_i}),\ \ j=1,\ldots,m \quad (8)$$

where $\mathbf{W}_{h_j}$ and $\mathbf{W}_{z_j}$ are randomly generated for the last state of enhancement nodes $\mathbf{H}_{j-1}$ and the collected features $\mathbf{Z}^n$.

Xu *et al.* [18] connected the enhanced nodes in each group cyclically, called R-BLS. This recurrence structure can capture the dynamic characteristics of the time series at the enhanced node. Use two typical chaotic system and real air quality datasets [28] to estimate the performance of R-BLS, which illustrates the efficient time-series forecasting ability.

3) Convolutional BLS: In [16], the BLS and convolution kernels are suggested to be constructed, providing prior knowledge for subsequence research. First, randomly sample the weights of the convolution filter based on the given distribution. The feature mapping node can be defined as

$$\begin{aligned}\mathbf{Z}_k &= \phi\big(\mathbf{Z}_{k-1};\{\mathbf{W}_{ek},\boldsymbol{\beta}_{e_k}\}\big)\\ &\triangleq \theta\big(P\big(\mathbf{Z}_{k-1}\otimes\mathbf{W}_{e_k}+\boldsymbol{\beta}_{e_k}\big)\big),\quad \text{for } k=1,\ldots,n \end{aligned}\quad (9)$$

where $\otimes$ is the convolutional operator. The functions $P(\cdot)$ and $\theta(\cdot)$ are the pooling operator and activation function, respectively. Then, the enhancement nodes can be generated by $\mathbf{H}_j = \xi_j(\mathbf{Z}\mathbf{W}_{h_j}+\boldsymbol{\beta}_{h_j})$, $j=1,2,3,\ldots,m$, where $\mathbf{Z}^n=[\mathbf{Z}_1,\ldots,\mathbf{Z}_n]$. Finally, $\mathbf{Z}^n$ and $\mathbf{H}^m$ are connected directly to calculate the output weight. It achieved a better testing accuracy and less time in face recognition images selected from MS-Celeb-1M [29] database compared with ResNet-34.

Li *et al.* [30] took advantages of this version of convolutional BLS and Adam algorithm for image classification. Yang [31] proposed, based on CNN-BLS, which maps the input image to PCA function node, and considers the hidden state and collection layer after convolution as enhancement features. It is obvious that it naturally connects BLS and CNN to process some complex image data.

Moreover, Yu and Zhao [32] used a similar structure to construct a novel convolutional BLS, called board convolutional neural network (BCNN), for fault diagnosis in industrial processes. BCNN can use lagging samples to understand the interactive information between each sample to represent existing samples. It have useful function that can feed back new faults in time without a repeated step on retraining the model.

4) Deep Cascade BLS: The adaptive cascaded deep BLS (DCBLS) designed by Ye *et al.* [33], its feature modification node and mapping node are respectively, converted into n and m deep cascade structure.

III. Adaptive Deep Variant of Broad Learning System: Stacked Structure

In the rest of this article, to construct the novel expansions of the BLS, a simplification description of BLS is provided as follows. Again for the BLS with n groups of feature nodes and m groups of enhancement nodes, in (3), the approximation results of the network should be

$$\begin{aligned} \boldsymbol{Y} &= [\boldsymbol{Z}^n, \boldsymbol{H}^m]\boldsymbol{W}^m \\ &= [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_n, \boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_m]\boldsymbol{W}^m \\ &= [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_n]\boldsymbol{W_E} + [\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_m]\boldsymbol{W_H} \end{aligned} \quad (10)$$

where $\boldsymbol{W}^m = \begin{bmatrix} \boldsymbol{W_E} \\ \boldsymbol{W_H} \end{bmatrix}$. Let $\mathcal{P}$ denote the generalized function for n feature mappings, e.g., a gathering of n groups of feature nodes in [15]. Similarly, $\mathcal{Q}$ denotes the generalized function to generate the enhancement nodes, e.g., a gathering of m groups of enhancement nodes with the feature nodes as its inputs in [15]. Therefore, the structure of BLS could be rewritten as

$$\begin{aligned} \boldsymbol{y} &= \mathcal{P}(\boldsymbol{x}, \mathcal{W}_e)\boldsymbol{W_E} + \mathcal{Q}(\mathcal{P}(\boldsymbol{x}, \mathcal{W}_e), \mathcal{W}_h)\boldsymbol{W_H} \\ &= \mathcal{P}(\boldsymbol{x}, \mathcal{W}_e)\boldsymbol{W_E} + \mathcal{Q}_{\mathcal{P}}(\boldsymbol{x}, \mathcal{W}_e, \mathcal{W}_h)\boldsymbol{W_H} \end{aligned} \quad (11)$$

where $\boldsymbol{y}$ and $\boldsymbol{x}$ are the output and input vectors, $\mathcal{Q}_{\mathcal{P}}(\cdot)$ denotes the composite mappings for $\mathcal{P}(\cdot)$ and $\mathcal{Q}(\cdot)$, and $\mathcal{W}_e = \{\boldsymbol{W}_{e_i} | 1 \le i \le n\}$ and $\mathcal{W}_h = \{\boldsymbol{W}_{h_j} | 1 \le j \le m\}$ are sets of random weights associated with the feature nodes and the enhancement nodes. The redrawn structure of BLS is given in Fig. 3.

In [16], it is concluded that plenty of linear or nonlinear information connected with the output layer could further enhance the performance of BLS. Unexpectedly, such modification may increase the redundancy of the network, which is against the efficient and effective nature of BLS. Here, we try to alleviate the above issue by constructing alternative deep variant of BLS networks.

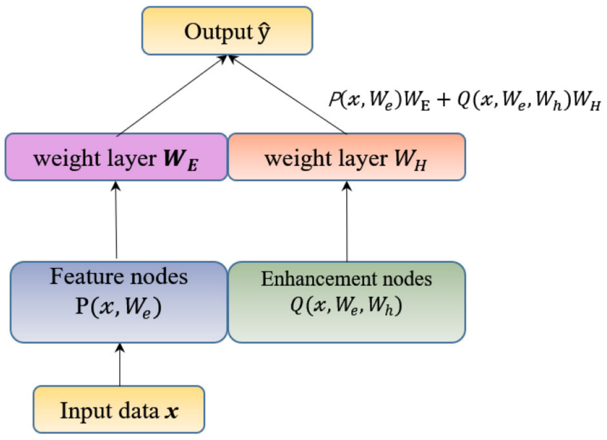

Output ŷ
$P(x, W_e)W_E + Q(x, W_e, W_h)W_H$
weight layer W_E
weight layer W_H
Feature nodes $P(x, W_e)$
Enhancement nodes $Q(x, W_e, W_h)$
Input data x

Fig. 3. BLS: redrawn representation.

However, one crucial problem is that how to find a proper combination method to take advantages of several BLS blocks. Stacking several normal BLS blocks to form a deeper structure is a straightforward choice. Here, we proposed Stacked BLS networks where the added BLS are trained to approximate the residual of the previous BLS. In the end of this article, we also conclude that residual is a typical kind of functional link designed by identity mapping.

In this section, first, a general scheme for Stacked BLS networks will be given. After that, the incremental algorithms of Stacked BLS are shown. It is obvious that the new deep model is still efficient and effective since no gradient methods are involved. Finally, another incremental algorithm that also considers the increment of enhancement nodes within the Stacked BLS is proposed.

A. Stacked BLS

For the Stacked BLS, mathematically, we have the following formulations.

Denote the input dataset as $\boldsymbol{x}$ and the output data as $\boldsymbol{y}$, the proposed structure with n Stacked BLS blocks $\boldsymbol{u}_i, i = 1, \ldots, n$ is described below.

First, recall that the original BLS network could be represented equivalently by (11). In order to separate the individual BLS blocks from the original BLS, we reformulate the block as follows. For the ith block, we have

$$\begin{aligned} \boldsymbol{u}_i &= \mathcal{P}_i(\boldsymbol{v}_i, \mathcal{W}_{e_i})\boldsymbol{W}_{\boldsymbol{E}_i} + \mathcal{Q}_i(\mathcal{P}_i(\boldsymbol{v}_i, \mathcal{W}_{e_i}), \mathcal{W}_{h_i})\boldsymbol{W}_{\boldsymbol{H}_i} \\ &= \mathcal{P}_i(\boldsymbol{v}_i, \mathcal{W}_{e_i})\boldsymbol{W}_{\boldsymbol{E}_i} + \mathcal{Q}_{\mathcal{P}_i}(\boldsymbol{v}_i, \mathcal{W}_{e_i}, \mathcal{W}_{h_i})\boldsymbol{W}_{\boldsymbol{H}_i} \end{aligned} \quad (12)$$

where $i = 1, \ldots, n$, the weights $\mathcal{W}_{e_i}$ and $\mathcal{W}_{h_i}$ are randomly generated, and $\boldsymbol{v}_i = g(\boldsymbol{u}_{i-1})$. In this section, function g is selected as the identity function, i.e., $\boldsymbol{v}_i = \boldsymbol{u}_{i-1}$. Theoretically, $\mathcal{P}_i(\cdot)$ and $\mathcal{P}_j(\cdot)$, for $i \neq j$, can be chosen differently depending upon the complexity of the modeling tasks. Without loss of generality, the subscripts of the ith random mappings $\mathcal{P}_i(\cdot)$ and the jth feature mappings $\mathcal{P}_j(\cdot)$ are omitted in this article. Similarly, the subscripts of the ith mappings $\mathcal{Q}_i(\cdot)$ and the jth random mappings $\mathcal{Q}_j(\cdot)$ are also omitted.

In (12), the weights $\boldsymbol{W}_{\boldsymbol{E}_i}$ and $\boldsymbol{W}_{\boldsymbol{H}_i}$ are usually obtained by solving the following proximal problem:

$$\underset{\boldsymbol{W}_{\boldsymbol{E}_i}, \boldsymbol{W}_{\boldsymbol{H}_i}}{\arg\min}: \ \|\boldsymbol{u}_i - \boldsymbol{y}_i\|_2^2 + \lambda \|[\boldsymbol{W}_{\boldsymbol{E}_i}, \boldsymbol{W}_{\boldsymbol{H}_i}]\|_2^2 \tag{13}$$

where $\boldsymbol{y}_i$ is the desired output for the training data $\boldsymbol{v}_i$ in the ith blocks.

Theoretically, the optimal solution could be approximated by the ridge regression, which is

$$\boldsymbol{W}_i = \begin{bmatrix} \boldsymbol{W}_{\boldsymbol{E}} \\ \boldsymbol{W}_{\boldsymbol{H}} \end{bmatrix} = \boldsymbol{A}^{+}\boldsymbol{y}_i = \left(\lambda \boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{y}_i \tag{14}$$

where $\boldsymbol{A} = [\mathcal{P}(\boldsymbol{v}_i, \mathcal{W}_{e_i}) | \mathcal{Q}_{\mathcal{P}}(\boldsymbol{v}_i, \mathcal{W}_{e_i}, \mathcal{W}_{h_i})]$. Specifically, we have that

$$\boldsymbol{A}^{+} = \lim_{\lambda \to 0}\left(\lambda \boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T. \tag{15}$$

Second, to take full advantage of the output of each BLS blocks, all the outputs $\boldsymbol{u}_i (i = 1, \ldots, n)$ are used to approximate the desired output layer $\boldsymbol{y}$. The adjacent blocks are combined by the residual, i.e., the desired output of each block could be

$$\boldsymbol{y}_i = \boldsymbol{y} - \sum_{k=1}^{i-1} \boldsymbol{u}_k. \tag{16}$$

For any given training data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i) | \boldsymbol{x}_i \in \mathbb{R}^M, \boldsymbol{y}_i \in \mathbb{R}^C, i = 1, \ldots, N\}$, the first BLS block is generated by setting $\boldsymbol{y}_1 = \boldsymbol{y}$ and $\boldsymbol{v}_1 = \boldsymbol{x}$, which is

$$\boldsymbol{u}_1 = \mathcal{P}(\boldsymbol{x}, \mathcal{W}_{e_1})\boldsymbol{W}_{\boldsymbol{E}_1} + \mathcal{Q}_{\mathcal{P}}(\boldsymbol{x}, \mathcal{W}_{e_1}, \mathcal{W}_{h_1})\boldsymbol{W}_{\boldsymbol{H}_1} \tag{17}$$

where weights $\mathcal{W}_{e_1}$ and $\mathcal{W}_{h_1}$ are randomly generalized and the weights $\boldsymbol{W}_{\boldsymbol{E}_1}$ and $\boldsymbol{W}_{\boldsymbol{H}_1}$ are calculated through the pseudo inverse of matrix $[\mathcal{P}(\boldsymbol{x}, \mathcal{W}_{e_1}), \mathcal{Q}_{\mathcal{P}}(\boldsymbol{x}, \mathcal{W}_{e_1}, \mathcal{W}_{h_1})]$ by (15), which is

$$\begin{bmatrix} \boldsymbol{W}_{\boldsymbol{E}_1} \\ \boldsymbol{W}_{\boldsymbol{H}_1} \end{bmatrix} = [\mathcal{P}(\boldsymbol{x}, \mathcal{W}_{e_1}), \mathcal{Q}_{\mathcal{P}}(\boldsymbol{x}, \mathcal{W}_{e_1}, \mathcal{W}_{h_1})]^{+}\boldsymbol{y}. \tag{18}$$

In fact, the above procedure generate a standard network of the original BLS.

For the subsequent BLS blocks, we follow (16) to generate new dataset to train them. In other words, the transformed output of the first BLS network $g(\boldsymbol{u}_1)$ is fed into the next individual block as the training data, which is denoted as $\boldsymbol{v}_2 = g(\boldsymbol{u}_1)$, and the desired output of the second BLS block $\boldsymbol{y}_2$ is the residual of the first BLS block, $\boldsymbol{y}_2 = \boldsymbol{y} - \boldsymbol{u}_1$.

Specifically, we have

$$\begin{aligned} \boldsymbol{u}_2 = \ &\mathcal{P}(g(\boldsymbol{u}_1), \mathcal{W}_{e_2})\boldsymbol{W}_{\boldsymbol{E}_2} \\ &+ \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_1), \mathcal{W}_{e_2}, \mathcal{W}_{h_2})\boldsymbol{W}_{\boldsymbol{H}_2} \end{aligned} \tag{19}$$

where weights $\mathcal{W}_{e_2}$ and $\mathcal{W}_{h_2}$ are randomly generalized. Meanwhile, weights $\boldsymbol{W}_{E_2}$ and $\boldsymbol{W}_{H_2}$ are the solutions of the proximal problem (13), which could be deduced as

$$\begin{aligned} \begin{bmatrix} \boldsymbol{W}_{\boldsymbol{E}_2} \\ \boldsymbol{W}_{\boldsymbol{H}_2} \end{bmatrix} = \ &[\mathcal{P}(g(\boldsymbol{u}_1), \mathcal{W}_{e_2}), \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_1), \mathcal{W}_{e_2}, \mathcal{W}_{h_2})]^{+} \\ &\times (\boldsymbol{y} - \boldsymbol{u}_1). \end{aligned} \tag{20}$$

Respectively, for the increment of the ith BLS block, $i = 2, \ldots, n$, we have that

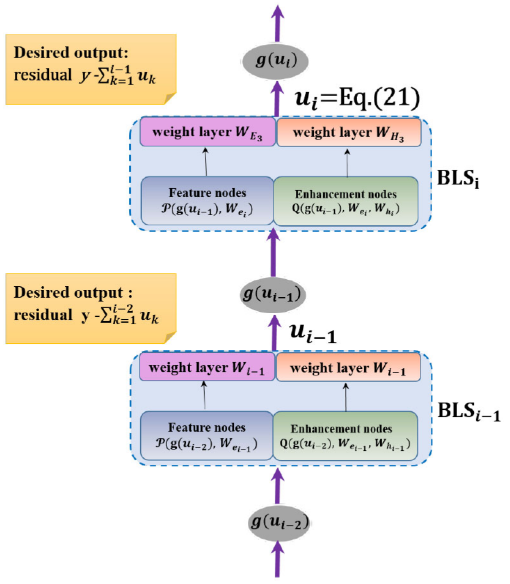

Desired output:
residual $y - \sum_{k=1}^{i-1} u_k$
$g(u_i)$
u_i=Eq.(21)
weight layer W_{E_3}
weight layer W_{H_3}
Feature nodes $\mathcal{P}(g(u_{i-1}), W_{e_i})$
Enhancement nodes $Q(g(u_{i-1}), W_{e_i}, W_{h_i})$
$\mathrm{BLS_i}$
Desired output :
residual $y - \sum_{k=1}^{i-2} u_k$
$g(u_{i-1})$
u_{i-1}
weight layer W_{i-1}
weight layer W_{i-1}
Feature nodes $\mathcal{P}(g(u_{i-2}), W_{e_{i-1}})$
Enhancement nodes $Q(g(u_{i-2}), W_{e_{i-1}}, W_{h_{i-1}})$
BLS_{i-1}
$g(u_{i-2})$

Fig. 4. Stacked BLS: the *i*th Stacked BLS block.

$$\begin{aligned} \boldsymbol{u}_i = \ &\mathcal{P}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i})\boldsymbol{W}_{\boldsymbol{E}_i} \\ &+ \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{h_i})\boldsymbol{W}_{\boldsymbol{H}_i} \end{aligned} \tag{21}$$

where weights $\mathcal{W}_{e_i}$ and $\mathcal{W}_{h_i}$ are randomly generalized, and

$$\begin{aligned} \begin{bmatrix} \boldsymbol{W}_{\boldsymbol{E}_i} \\ \boldsymbol{W}_{\boldsymbol{H}_i} \end{bmatrix} = \ &[\mathcal{P}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}), \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{h_i})]^{+} \\ &\times \left(\boldsymbol{y} - \sum_{k=1}^{i-1} \boldsymbol{u}_k\right). \end{aligned} \tag{22}$$

Fig. 4 illustrates the generating procedure of the ith BLS network. Moreover, for the Stacked BLS with n blocks, the last block is trained to approximate $\boldsymbol{y}_n$, which is

$$\boldsymbol{u}_n \approx \boldsymbol{y}_n = \boldsymbol{y} - \sum_{k=1}^{n-1} \boldsymbol{u}_k.$$

Equivalently, we have that

$$\boldsymbol{y} \approx \sum_{k=1}^{n} \boldsymbol{u}_k. \tag{23}$$

Consequently, we could deduce that the desired output $\boldsymbol{y}$ is finally approximated by the summation of all the outputs of n BLS blocks. The whole structure of the proposed method is illustrated in Fig. 5.

Through the above steps, the Stacked BLS that consists of n individual BLS blocks is constructed. Algorithm 1 offers an intuitive representation for the above procedure. It should be mentioned that in our proposed Stacked method, the former blocks are fixed when a new BLS network is added. In other words, the BLS blocks avoid the retrain procedure when the new block is set up. From this point of view, the new model is equivalent to an incremental algorithm for the increment of additional "layer." Recall that the incremental learning models based on the BLS are discussed in [16], which includes: 1) the increment of the feature mapping nodes; 2) the increment of additional enhancement nodes; and 3) the increment

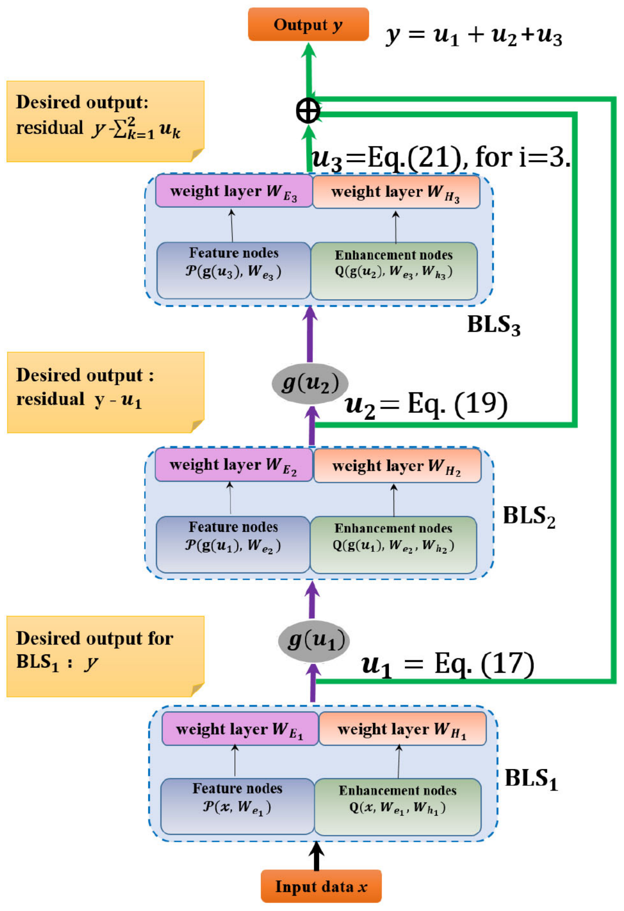

Output y
$y = u_1 + u_2 + u_3$
Desired output: residual $y - \sum_{k=1}^{2} u_k$
u_3=Eq.(21), for i=3.
weight layer W_{E_3}
weight layer W_{H_3}
Feature nodes $\mathcal{P}(g(u_3), W_{e_3})$
Enhancement nodes $Q(g(u_2), W_{e_3}, W_{h_3})$
BLS$_3$
Desired output : residual y - u_1
$g(u_2)$
u_2= Eq. (19)
weight layer W_{E_2}
weight layer W_{H_2}
Feature nodes $\mathcal{P}(g(u_1), W_{e_2})$
Enhancement nodes $Q(g(u_1), W_{e_2}, W_{h_2})$
BLS$_2$
Desired output for BLS$_1$: y
$g(u_1)$
u_1 = Eq. (17)
weight layer W_{E_1}
weight layer W_{H_1}
Feature nodes $\mathcal{P}(x, W_{e_1})$
Enhancement nodes $Q(x, W_{e_1}, W_{h_1})$
BLS$_1$
Input data x

Fig. 5. Stacked BLS: the whole network of the Stacked BLS model when $n = 3$, i.e., there are three blocks stacked in the proposed model.

of additional input patterns. Here, the Stacked BLS could be regarded as an important extension for the incremental expansion of BLS.

B. Incremental Learning of Stacked BLS

In real applications, the scale of the stacked individual blocks could not be set in advance, especially when the network is aimed to achieve a desired accuracy. In this case, it is preferred that the network and the blocks are able to insert additional nodes to obtain better performance.

Fortunately, BLS is a suitable system that is able to add the enhancement nodes or feature mapping nodes dynamically. The superiority of the incremental algorithms proposed in [16] has been demonstrated in various datasets. Next, we will detail the Stacked BLS with the increment of additional enhancement nodes in each blocks.

Assume that for the ith block, a group of p enhancement nodes is added. Consequently, the formulation of the ith blocks could be updated by

$$\begin{aligned} \boldsymbol{u}_i = &\ \mathcal{P}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i})\boldsymbol{W}_{\boldsymbol{E}_i} \\ &+ \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{h_i})\boldsymbol{W}_{\boldsymbol{H}_i} \\ &+ \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{ah_i})\boldsymbol{W}_{a\boldsymbol{H}_i} \end{aligned} \quad (24)$$

where $\mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{ah_i})$ is the new enhancement nodes and the weights $\mathcal{W}_{ah_i}$ are randomly generalized.

Algorithm 1: Algorithm for Stacked BLS

```
Input: sample $x$; number of increment blocks of
       Stacked BLS $n$,threshold $\epsilon$
Output: the output of Stacked BLS $\hat{y}$
1  for $i = 1$ do
2      Randomly initialize $\mathcal{W}_{e_1}, \mathcal{W}_{h_1}$;
3      Calculate $[\mathcal{P}(x, \mathcal{W}_{e_1}), \mathcal{Q}_{\mathcal{P}}(x, \mathcal{W}_{e_1}, \mathcal{W}_{h_1})]$;
4      Calculate $W_{E_1}$, $W_{H_1}$ by (18);
5      Calculate $u_1$ by (17);
6  end
7  for all the i,$i = 2, \ldots, n$ do
8      Set $y_i = y - \sum_{k=1}^{i-1} u_k$;
9      Randomly initialize $\mathcal{W}_{e_i}, \mathcal{W}_{h_i}$;
10     Calculate $[\mathcal{P}(g(u_{i-1}), \mathcal{W}_{e_i}), \mathcal{Q}_{\mathcal{P}}(g(u_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{h_i})]$;
11     Calculate $W_{E_i}$, $W_{H_i}$ by (22);
12     Calculate $u_i$ by (21);
13     if $y - \sum_{k=1}^{i-1} u_k < \epsilon$ then
14         Break;
15     end
16 end
17 Calculate $\hat{y} = \sum_{k=1}^{n} u_k$.
```

Following the discussions in [15], we could dynamically update the weights matrix $\boldsymbol{W}_{\boldsymbol{E}_i}, \boldsymbol{W}_{\boldsymbol{H}_i}$, and $\boldsymbol{W}_{a\boldsymbol{H}_i}$ by the following formula:

$$[A, \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{ah_i})]^{+} = \begin{bmatrix} (\boldsymbol{A})^{+} - \boldsymbol{D}\boldsymbol{B}^{T} \\ \boldsymbol{B}^{T} \end{bmatrix} \quad (25)$$

where $\boldsymbol{A} = [\mathcal{P}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}), \mathcal{Q}_{\mathcal{P}}(\boldsymbol{u}_{i-1}, \mathcal{W}_{e_i}, \mathcal{W}_{h_i})]$, $\boldsymbol{D} = (\boldsymbol{A})^{+}\mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{ah_i})$

$$\boldsymbol{B}^{T} = \begin{cases} (\boldsymbol{C})^{+}, & \text{if } \boldsymbol{C} \neq 0 \\ (\boldsymbol{1} + \boldsymbol{D}^{T}\boldsymbol{D})^{-1}\boldsymbol{D}^{T}(\boldsymbol{A})^{+}, & \text{if } \boldsymbol{C} = 0 \end{cases} \quad (26)$$

and $\boldsymbol{C} = \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{ah_i}) - \boldsymbol{A}\boldsymbol{D}$.

Hence, the new weights are

$$\begin{pmatrix} \boldsymbol{W}_i \\ \boldsymbol{W}_{a\boldsymbol{H}_i} \end{pmatrix} = \begin{bmatrix} \boldsymbol{W}_i - \boldsymbol{D}\boldsymbol{B}^{T}\boldsymbol{y} \\ \boldsymbol{B}^{T}\boldsymbol{y} \end{bmatrix} \quad (27)$$

where $\boldsymbol{W}_i = \begin{bmatrix} \boldsymbol{W}_{\boldsymbol{E}_i} \\ \boldsymbol{W}_{\boldsymbol{H}_i} \end{bmatrix}$.

Consequently, output $\boldsymbol{u}_i$ is updated by (24). The whole procedure of the increment of enhancement nodes is illustrated in Fig. 6. Meanwhile, the detailed algorithm is offered in Algorithm 2. Note that with this designed architecture, the weights of the lower remains unchanged; only the weights of the newly added BLS blocks are calculated. The calculation of the weights of the newly added BLS blocks is the same as the training of the BLS that was mentioned in literature. This makes the Stacked BLS very versatile and efficient.

IV. Residual Within the Structure of Broad Learning System

In this section, discussions about the residual within the structures of BLS are given. Residual has been widely used in various learning methods, such as the greedy algorithms [34] developed in 1930s. In the other hand, if the feature function

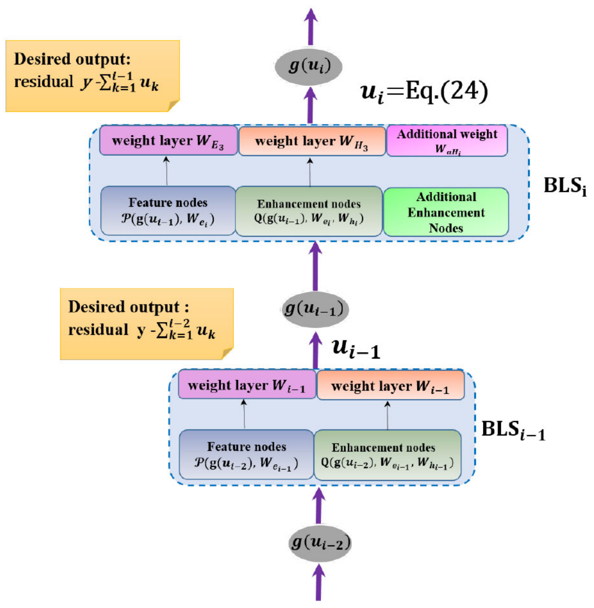

Desired output:
residual $y-\sum_{k=1}^{i-1} u_k$
$g(u_i)$
u_i=Eq.(24)
weight layer W_{E_3}
weight layer W_{H_3}
Additional weight W_{aH_i}
Feature nodes $\mathcal{P}(g(u_{i-1}), W_{e_i})$
Enhancement nodes $Q(g(u_{i-1}), W_{e_i}, W_{h_i})$
Additional Enhancement Nodes
BLS_i
Desired output :
residual $y-\sum_{k=1}^{i-2} u_k$
$g(u_{i-1})$
u_{i-1}
weight layer W_{i-1}
weight layer W_{i-1}
Feature nodes $\mathcal{P}(g(u_{i-2}), W_{e_{i-1}})$
Enhancement nodes $Q(g(u_{i-2}), W_{e_{i-1}}, W_{h_{i-1}})$
BLS_{i-1}
$g(u_{i-2})$

Fig. 6. Stacked BLS: the *i*th Stacked BLS blocks with the increment of additional enhancement nodes.

$\mathcal{P}$ is degenerated to identity mapping in the model of BLS, and the weight $\boldsymbol{W_E}$ is identity, the resulted structure is equivalent to the form of residual.

Recall that for any given training data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)|\boldsymbol{x}_i \in \mathbb{R}^M, \boldsymbol{y}_i \in \mathbb{R}^C, i = 1, \ldots, N\}$, the output approximation of the BLS should be

$$\begin{aligned} \boldsymbol{y} &= [\boldsymbol{x}, \xi(\boldsymbol{x}\boldsymbol{W}_h + \boldsymbol{\beta})]\boldsymbol{W} \\ &= \boldsymbol{x}\boldsymbol{W_E} + \xi(\boldsymbol{x}\boldsymbol{W}_h + \boldsymbol{\beta})\boldsymbol{W}_H \end{aligned} \tag{28}$$

where the weights $\boldsymbol{W}_h$ and $\boldsymbol{\beta}$ are randomly generated, and $\boldsymbol{W_E}$ and $\boldsymbol{W_H}$ are divisions of $\boldsymbol{W}$. Next, let function $\mathcal{F}$ represent the generalized enhancement mapping to be trained, then (28) could be rewritten as

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{x}\boldsymbol{W_E} + \mathcal{F}(\boldsymbol{x}, \{\boldsymbol{W}_h, \boldsymbol{W}_H\}) \\ &= \boldsymbol{x}\boldsymbol{W_E} + \mathcal{F}(\boldsymbol{x}, \boldsymbol{W}). \end{aligned} \tag{29}$$

Without loss of generalization, suppose that the linear projection $\boldsymbol{W_E}$ is identity, the BLS could be rebuilt as

$$\boldsymbol{y} = \boldsymbol{x} + \mathcal{F}(\boldsymbol{x}, \boldsymbol{W}). \tag{30}$$

Next, suppose that the desired output $\boldsymbol{y}$ is generated by the hidden mapping $\boldsymbol{y} = \mathcal{H}(\boldsymbol{x})$, we have that

$$\mathcal{F}(\boldsymbol{x}, \boldsymbol{W}) = \mathcal{H}(\boldsymbol{x}) - \boldsymbol{x}. \tag{31}$$

In this case, the BLS network is solved by approximating the residual of the desired output and input. However, the above formulation is compatible with the model proposed in [6], which tries to solve given approximation problem by a residual involved mapping instead of an original hidden feature mapping.

In the theory of residual learning, the above structure is not trivial. Generally, it is motivated by a classical feedforward neural network embedding with "shortcut connections" [35]–[37], where the shortcut connections are used to compress the structure of multiple layer networks. If the shortcut connections are selected as identify mappings, the output should be $\mathcal{F}(\boldsymbol{x}) + \boldsymbol{x}$. The above structure could be checked in Fig. 7 [38]. Although the identity mapping of the ResNets adds no additional parameters of the structure, the first choice of training the entire ResNet is still a gradient descent method or statistic gradient-descent method based on backpropagation. Such solutions are easy to be trapped in the local minimum or suffered from the time-consuming training process.

In conclusion, the residual is deeply embedded in the BLS and the Stacked BLS.

Remark: Although the structure of the BLS implies the construction of residual, the performance on specific applications may be different with the residual-based deep networks [38]. The most important reason is that their training methods are basically distinguish. For example, the ResNet in [38] is a typically gradient-based network, and iterations on the fine-tuned

Algorithm 2: Incremental Algorithm of Stacked BLS

```
Input: sample $\boldsymbol{x}$; number of increment blocks of Stacked BLS $n$, number of increment of additional enhancement nodes in per BLS blocks $p$, number of groups of additional enhancement nodes added in per BLS blocks $m$, threshold $\epsilon$
Output: the output of Stacked BLS $\hat{\boldsymbol{y}}$
1  for $i = 1$ do
2      Randomly initialize $\mathcal{W}_{e_1}, \mathcal{W}_{h_1}$;
3      Calculate $[\mathcal{P}(\boldsymbol{x}, \mathcal{W}_{e_1}), \mathcal{Q}_{\mathcal{P}}(\boldsymbol{x}, \mathcal{W}_{e_1}, \mathcal{W}_{h_1})]$;
4      Calculate $\boldsymbol{W}_{\boldsymbol{E}_1}, \boldsymbol{W}_{\boldsymbol{H}_1}$ by (18);
5      Calculate $\boldsymbol{u}_1$ by (17);
6  end
7  for all the i,$i = 2, \ldots, n$ do
8      Set $\boldsymbol{y}_i = \boldsymbol{y} - \sum_{k=1}^{i-1} \boldsymbol{u}_k$;
9      Randomly initialize $\mathcal{W}_{e_i}, \mathcal{W}_{h_i}$;
10     Calculate $[\mathcal{P}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}), \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{h_i})]$;
11     Calculate $\boldsymbol{W}_{\boldsymbol{E}_i}, \boldsymbol{W}_{\boldsymbol{H}_i}$ by (22);
12     Calculate $\boldsymbol{u}_i$ by (21), set
         $\boldsymbol{A} = [\mathcal{P}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}), \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{h_i})]$, and
         $\boldsymbol{W}_i = \begin{bmatrix} \boldsymbol{W}_{\boldsymbol{E}_i} \\ \boldsymbol{W}_{\boldsymbol{H}_i} \end{bmatrix}$;
13     for all the j,$j = 1, \ldots, m$ do
14         Randomly initialize $\mathcal{W}_{ah_i}$;
15         Calculate $\begin{bmatrix} \boldsymbol{W}_i \\ \boldsymbol{W}_{\boldsymbol{a}\boldsymbol{H}_i} \end{bmatrix}$ by (25, 26, 27);
16         Calculate $\boldsymbol{u}_i$ by (24);
17         Update $\boldsymbol{W}_{\boldsymbol{H}_i} = \begin{bmatrix} \boldsymbol{W}_{\boldsymbol{H}_i} \\ \boldsymbol{W}_{\boldsymbol{a}\boldsymbol{H}_i} \end{bmatrix}$, and $\mathcal{W}_{\boldsymbol{h}_i} = \mathcal{W}_{\boldsymbol{h}_i} \cup \mathcal{W}_{ah_i}$;
18         set
             $\boldsymbol{A} = [\mathcal{P}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}), \mathcal{Q}_{\mathcal{P}}(g(\boldsymbol{u}_{i-1}), \mathcal{W}_{e_i}, \mathcal{W}_{h_i})]$,
             and $\boldsymbol{W}_i = \begin{bmatrix} \boldsymbol{W}_{\boldsymbol{E}_i} \\ \boldsymbol{W}_{\boldsymbol{H}_i} \end{bmatrix}$;
19     end
20 end
21 Calculate $\hat{\boldsymbol{y}} = \sum_{k=1}^{n} \boldsymbol{u}_k$.
```

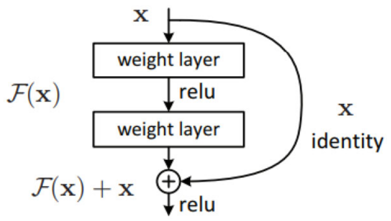


Fig. 7. Illustration of a typical ResNet [8].

TABLE I
DETAILS OF DATASETS FOR REGRESSION

Data set	No. of samples		Input variables
	Training	Testing	
Abalone	2784	1393	8
Basketball	64	32	4
Cleveland	202	101	13
Pyrim	49	25	27
Strike	416	209	6

parameters are necessary. For the BLS, only the pesudo-inverse of one matrix (generated by the feature nodes and enhancement nodes) is needed.

V. Performance Evaluation of Stacked BLS

In this section, experiments of the proposed models are presented. Various methods are also compared in several benchmark datasets. Additionally, the incremental algorithms of Stacked BLS are tested. Algorithms for the proposed method tested in all the experiments are available in https://www.broadlearning.ai.

A. Regression on UCI Datasets

Five regression datasets of the UCI datasets [39] are selected to test the performance of the proposed method. Generally, the selected datasets are in medium size and dimensions, which are the typical benchmarks for the regression tasks. Information of the datasets is given in Table I. Here, the Stacked BLS is compared with several methods, such as support vector machine (SVM), least square SVM (LSSVM) [40], extreme learning machine (ELM) [41], BLS, and Fuzzy BLS [42]. Among them, the so-called Fuzzy BLS is a typically variant of BLS, which merges the Takagi–Sugeno (TS) fuzzy system into the BLS, while the other methods are the typical benchmarks in experiments tested on UCI dataset.

For a fair comparison, all the parameters involved in the tested methods are selected by a grid search. For the SVM, LSSSVM and ELM, the penalty parameter C that balances the regularization and the squared errors, and the Gaussian kernel parameter γ that decides the specific kernel adopted in the approximations need to be tuned. Hence, the parameters (C, γ) are carefully selected from the set of $\{2^{-24}, 2^{-23}, \dots, 2^{24}, 2^{25}\}$. For the BLS, the number of feature nodes per group N_f, the number of group of feature nodes N_m, and the number of enhancement nodes N_e need to be searched from the sets of $[1, 10]$, $[1, 30]$, and $[1\,200]$, respectively. For the Fuzzy BLS, the number of fuzzy rules N_r, the number of fuzzy subsystem N_t, and the number of enhancement nodes N_e are chosen from the sets of $[1, 20]$, $[1, 20]$, and $[1\,100]$, respectively.

For the Stacked BLS, the structure of each BLS block could be totally different. For simplicity, settings of the proposed algorithm are divided into two parts. For the first BLS blocks, the number of feature nodes per group N_f, the number of groups of feature nodes N_m, and the number of enhancement nodes N_e are searched from the sets $[1, 10]$, $[1, 20]$, and $[1, 50]$, respectively. For the additional BLS blocks in the system, the number of feature nodes per group N_{fs}, the number of group of feature nodes N_{ms}, and the number of enhancement nodes N_{es} are search from the sets $[1, 10]$, $[1, 10]$, and $[1, 20]$, respectively. Meanwhile, the number of the Stacked BLS blocks N_n is searched from 2 to 10. All the searching step is set as 1. The selected parameters for the various methods are shown in Table II.

Experiments on the five datasets are repeated 50 times and the average of the testing root mean-squared errors (testing RMSE) are listed in Table III. To give a more straightforward comparison between various methods, the standard deviation for the results of 50 trials are also listed. Although the testing RMSEs of BLS and fuzzy BLS outperform the other methods, the Stacked BLS achieves the best approximation among all the six methods. Results indicate that the Stacked BLS has more potential in regression-based applications.

B. Classification on MNIST and NORB Datasets

Experiments on classification are tested on the MNIST [43] and NYU Object Recognition Benchmark (NORB) [44] datasets in this section, including the stacked model in Section III-A and the incremental model in Section III-B.

Several methods with deep structure are compared with BLS in the tests on MNIST and NORB datasets. Among them, various methods, such as DBNs [11], DBM [13], two kinds of stacked auto encoders (SAEs) [12], [45], multilayer perceptron (MLP) [46], and the standard ResNet with 34 (ResNet-34) [6] without special feature tricks are based on the gradient. Therefore, there are parameters need to be tuned carefully. The detailed settings are listed below. The initial learning rate of the gradient decent method is set as 0.1; the decay learning rate for each learning epoch is 0.95 (except for the ResNet 34 which is set as 5^{-6}). There are other methods based on ELM, including the multilayer ELM (MLELM) [47] and the modified hierarchical ELM (HELM) [48]. Regularization parameters for the 3-layer MLELM are set as 10^{-1}, 10^{3}, and 10^{8}, respectively, and for the HELM algorithms, the penalty parameter of the regularization item is set as 10^{8}. All the experiments of the above methods are tested on a laptop that equips with Intel-i7 2.4-GHz CPU. In addition, other details of the experimental setting could be checked in [48]. Except for the original BLS algorithms, we also compare the results on the MNIST and NORB datasets with the structural variants proposed in [16], including the cascade of the feature mapping nodes (CFBLS), the limited connection between the groups of cascaded feature maps and the enhancement nodes

TABLE II
PARAMETER SETTINGS OF SVM, LSSVM, ELM, AND BLS FOR THE UCI DATASETS

Data set	SVM		LSSVM		ELM		BLS			Fuzzy BLS			Stacked BLS						
	C	γ	C	γ	C	γ	N_f	N_m	N_e	N_r	N_t	N_e	N_f	N_m	N_e	N_{fs}	N_{ms}	N_{es}	N_n
Abalone	2^2	2^{-1}	2.8932	3.0774	2^0	2^0	5	6	41	4	6	37	5	4	25	3	2	10	2
Basketball	2^0	2^0	6.0001	27.3089	2^{25}	2^{11}	6	7	4	5	1	26	7	3	6	6	2	5	3
Cleveland	2^2	2^2	0.7527	45.2507	2^{13}	2^{15}	1	10	9	16	1	44	6	5	4	2	3	13	2
Pyrim	2^{10}	2^8	52.5877	3.2463	2^2	2^6	3	7	2	1	7	48	3	2	5	2	4	3	2
Strike	2^0	2^{-4}	0.3167	0.7383	2^{-1}	2^5	9	11	30	3	2	23	1	7	37	9	8	19	2

TABLE III
PERFORMANCE COMPARISON (TESTING RMSE) OF SVM, LSSVM, ELM, BLS, FUZZY BLS, AND STACKED BLS FOR THE REGRESSION DATASETS

Datasets	SVM	LSSVM	ELM	BLS	Fuzzy BLS	Stacked BLS
	Aver±Std	Aver±Std	Aver±Std	Aver±Std	Aver±Std	Aver±Std
Abalone	0.0757±0.0017	0.0748±0.0014	0.0754±0.0012	0.0746±0.0011	0.0745±0.0011	**0.0739±0.0008**
Basketball	0.0831±0.0055	0.0815±0.0067	0.0824±0.0064	0.0810±0.0069	0.0808±0.0052	**0.0753±0.0070**
Cleveland	0.1252±0.0058	0.1169±0.0060	0.1165±0.0118	0.1100±0.0045	0.1112±0.0041	**0.1054±0.0039**
Pyrim	0.1069±0.0080	0.0887±0.0077	0.0824±0.0081	0.0929±0.0117	0.0767±0.0075	**0.0640±0.0077**
Strike	0.0736±0.0142	0.0725±0.0096	0.0713±0.0169	0.0682±0.0132	0.0665±0.0103	**0.0553±0.0141**

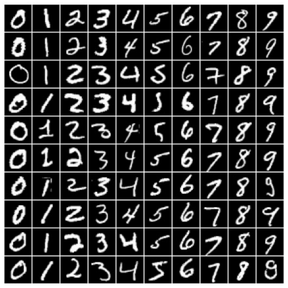


Fig. 8. MINIST dataset.

TABLE IV
CLASSIFICATION ACCURACY ON THE MNIST DATASET

Method	Accuracy (%)	Traing time (s)
SAE	98.60	36448.40
SDA	98.72	37786.03
DBN	98.87	53219.77
DBM	99.05	121455.69
MLP	97.39	21468.12
MLELM	99.04	475.83
HELM	**99.13**	281.37
FRBM	97.44	577.8
ResNet34	98.96	16583
BLS [15]	98.96	**29.9157***
CFEBLS [16]	98.83	62.6869*
Stacked BLS	**99.12**	30.1916*

(LCFBLS), the cascade of the enhancement nodes (CEBLS), and cascade of feature mapping nodes and enhancement nodes (CFEBLS). The details of the experimental setting could be checked in [16]. Since the CFEBLS outperforms other structural variants, only the results of CFEBLS are considered in this section. The BLS-related algorithms are carried out on a 3.40-GHz Intel i7-6700 CPU processor PC with MATLAB platform. Several duplicate experiments are also tested in the server computer equips with 2.30-GHz Intel Xeon E5-2650 CPU processor. Since the tests on the server are dramatically rapid compared with a plain computer, the results on consuming time are marked with a special superscript $*$.

The details of the datasets and experimental results are listed as follows.

1) MNIST: The MNIST dataset consists of 60 000 training images and 10 000 testing images of ten classes. Fig. 8 represents typical examples with the dimension of 28×28 per digit image.

To test the proposed method in the MNIST dataset, the structure and results reported in [15] are used to evaluate our Stacked BLS. Recall that the BLS network for experiments on MNIST is constructed by total 10×10 feature nodes and 11 000 enhancement nodes. To take the full advantage of this prior knowledge, the network setting for the first BLS block is the same as the structure reported in [15]. For the other stacked blocks, parameters N_f, N_w, and N_e are selected from the sets [1, 10], [1, 10], and [1, 500], respectively. The block stacking is stopped when the testing accuracy decreases. Detailed results for the mentioned methods are listed in Table IV. The testing accuracy of the Stacked BLS is greatly increased compared with the original BLS and its structural variants, while the total time for training is almost the same with the original BLS.

The corresponding snapshot results of each Stacked BLS blocks are shown in Table V. Despite the huge improvement in accuracy, the accumulative training time and testing time are slightly increased. It is concluded that the Stacked BLS could be implemented easily upon the pretrained BLS networks, with small extra cost.

2) NORB: NORB dataset consists of 48 600 images with $2 \times 32 \times 32$ pixels each. The objects in the dataset are belong to 5 distinct classes, which are: 1) animals; 2) humans; 3) airplanes; 4) trucks; and 5) cars. Among them, 24 300 images are selected for training and the other 24 300 images are for testing, as shown in Figs. 9 and 10.

TABLE V
SNAPSHOT RESULTS OF MNIST CLASSIFICATION USING STACKED BLS

No. of Stacked BLS blocks	Accuracy %	No. Para.			Time(s)		Accumulative Time(s)	
	Test	N_f	N_m	N_e	Train	Test	Train	Test
1	98.74%	10	10	11000	29.9157*	1.0783*	29.9157*	1.0783*
2	99.03%	5	5	470	0.0796*	0.1829*	29.9953*	1.2612*
3	99.09%	2	7	6	0.1035*	0.0963*	30.0988*	1.3575*
4	99.11%	1	3	7	0.0372*	0.0716*	30.1360*	1.4291*
5	99.12%	1	1	19	0.0556*	0.1045*	30.1916*	1.5337*

Fig. 9. Examples for training figures.

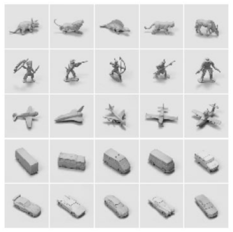


Fig. 10. Examples for testing figures.

To evaluate the performance of the Stacked BLS, we carry out an alternative scheme of the model structure with no prior knowledge. Specifically, for the first block, to control the number of parameters of the whole system, parameters (N_f, N_w, N_e) are selected as $(100, 10, 3000)$. Meanwhile, the size of the stacked blocks is determined by the grid search from $[1, 10] \times [1, 10] \times [1, 500]$, with the step size as 1. Consequently, the network is constructed by three Stacked BLS blocks whose structures are $(100, 10, 3000)$, $(1, 1, 400)$, and $(1, 1, 400)$, respectively.

Table VI represents the final accuracy of the Stacked BLS while Table VII shows the snapshot results of the algorithms. It is surprised that the training time of Stacked BLS is shortened to 5.1718 s, while the testing accuracy on NORB dataset is 91.90%. Moreover, it should be noticed that the total number of nodes is reduced to 4802, which is much less than the reported regular BLS stricture in [15] that has 10 000 nodes. This indicates that the supremacy of the Stacked BLS structure.

Additional experiments are designed to test Algorithm 2. The initial network begins with the output of the first Stacked BLS block, which is set as 100×10 feature nodes and 3000 enhancement nodes. Similar to the previous design in [15], the enhancement nodes of each stacked block are dynamically increased from 200 to 400 at the step of 100 in each update. The accuracy results and the time costs of each update

TABLE VI
CLASSIFICATION ACCURACY ON THE NORB DATASET

Method	Accuracy (%)	Traing time (s)
SAE	86.28	60504.34
SDA	87.62	65747.69
DBN	88.47	87280.42
DBM	89.65	182183.53
MLP	84.20	34005.470
MLELM	88.91	775.2850
HELM	91.28	432.19
BLS [15]	89.27	21.2546*
CFEBLS [16]	90.02	55.6718*
Stacked BLS	**91.90**	**5.1718***

are shown in Table VIII. The results are competitive to that we present in Table VI when the networks finally reach the structure of $(100, 10, 3000) - (1, 1, 400) - (1, 1, 400)$. This proves that the incremental learning algorithm for the proposed Stacked BLS is promising and effective.

C. Classification on Fashion-MNIST Dataset

Lately, Fashion-MNIST [49] dataset is developed to overcome the following serious shortcomings in MNIST dataset. MNIST dataset is lack of challenges. A number of deep models, such as CNN networks, could distinguish the test images of MNIST perfectly (the testing accuracy is up to 99.7%). MNIST is exhausted in classification tasks. MNIST cannot serve as a typical benchmark especially in modern Computer Vision tasks. Therefore, Fashion-MNIST is more preferred by the researchers to demonstrate their algorithms.

Similar to the MNIST dataset, there are 60 000 training samples and 10 000 testing samples in Fashion-MNIST which are labeled by the following ten tags: 1) T-shirt/top; 2) trouser; 3) pullover; 4) dress; 5) coat; 6) sandal; 7) shirt; 8) sneaker; 9) bag; and 10) ankle boot. Each of the samples is equipped with 28×28 pixels.

In our experiments, technical preprocessing is not involved before the implementation of our Stacked BLS. Consequently, to validate the proposed algorithms, we compared the performance in Fashion-MNIST with several selected methods, such as the MLP $(256 - 128 - 100)$, 3-layer CNN $(16 - 32 - 64)$, Random Forest Classifier (RF [50]), Xgboost method with triplet loss [51], Dyra-Net [52], and BLS $(100 \times 10 - 9000)$. The Stacked BLS is performed upon the residual

TABLE VII
SNAPSHOT RESULTS OF NORB CLASSIFICATION USING STACKED BLS

No. of Stacked BLS blocks	Accuracy %	No. Para.			Time(s)		Accumulative Time(s)	
	Test	N_f	N_m	N_e	Train	Test	Train	Test
1	89.86%	100	10	3000	4.7491*	1.433*	4.7491*	1.4330*
2	90.58%	1	1	400	0.2044*	0.2234*	4.9536*	1.6564*
3	91.90%	1	1	400	0.2182*	0.2194*	5.1718*	1.8758*

TABLE VIII
SNAPSHOT RESULTS OF NORB CLASSIFICATION USING STACKED BLS: INCREMENT OF ENHANCEMENT NODES

No. of Stacked BLS blocks	Accuracy %	No. Enh.	Time(s)		Accumulative Time(s)	
	Test	N_e	Train	Test	Train	Test
2	90.94%	200	0.9732*	0.5492*	5.7223*	1.9882*
	91.01%	300	0.8057*	0.4887*	6.5280*	2.4769*
	91.36%	400	1.0622*	0.7689*	7.2969*	3.2458*
3	91.41%	200	1.1164*	0.7171*	8.4133*	3.9629*
	91.42%	300	0.8662*	0.5030*	9.2795*	4.4659*
	91.48%	400	1.2793*	0.6098*	10.5588*	5.0757*

TABLE IX
CLASSIFICATION ACCURACY ON THE FASHION-MNIST DATASET

Method	Accuracy (%)
MLP	88.33
CNN	90.33
RF	87.3
Xgboost	89.82
AlexNet	89.9
Dyra-Net	90.6
BLS [15]	91.39
Stacked BLS	**91.53**

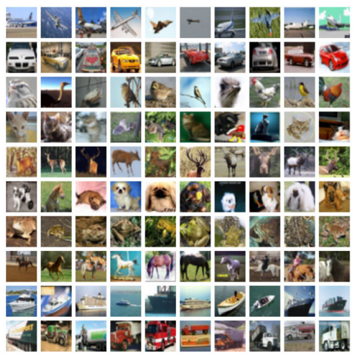


Fig. 11. Example images selected from the CIFAR-10 dataset.

of the original BLS networks and the structure of each BLS-blocks is search from $[1, 10] \times [1, 10] \times [1, 500]$ with the step size as 1. As a result, the final structure of the 3-layer Stacked BLS is $(100, 10, 9000)$, $(8, 9, 380)$, and $(8, 9, 380)$. Detailed results are shown in Table IX and the snapshot results are shown in Table X, where the benefits of the original BLS and the Stacked BLS are fully demonstrated.

Remark: To take advantages of the features generated by the former BLS-blocks, a richer combination of the information is more suitable in some cases. Therefore, instead of sending the output of the former BLS-blocks, experiments are also designed to dispatch its assembly of feature nodes and enhancement nodes to the new BLS-blocks. The resulted structure of modified Stacked BLS is $(100, 10, 9000)$, $(6, 10, 2100)$, and $(8, 9, 2100)$ and it performs well on Fashion-MNIST with comparative results whose accuracy is 91.60%.

D. Classification on CIFAR-10, SVHN, and CIFAR-100 Datasets

We also develop and perform the Stacked CCFBLS model on two benchmarks for image classification: 1) CIFAR-10 Dataset [53] and CIFAR-100 Dataset [53], whose training examples are shown in Fig. 11; and 2) the street-view house numbers (SVHNs) Dataset [54], whose training examples are illustrated in Fig 12. Here, the discussed Stacked CCFBLS model consists of one CCFBLS block and several BLS blocks. Similarly, other variants of BLS, such as the multiple kernel embedded BLS [55], could also be expanded into the stacked structure for better performance. Associated experiments are tested here to ensure its superiority over the ResNet [6].

Details of the datasets are listed as follows.

1) CIFAR-10: The CIFAR-10 dataset is a subset of the 80 million tiny images dataset which contains 50 000 training images ladled equally with 10 classes and 10 000 testing images of the corresponding ten classes. It is one of the most popular image datasets in machine learning. The image size of the samples is 32×32.

2) SVHN: The SVHNs dataset includes 73 257 images of ten classes for training, and 26 032 images for testing. It is captured in the Google Street View images and is developed for machine learning and object recognition algorithms. The samples of this dataset are similar to the MNIST dataset with 32×32 pixels.

3) CIFAR-100: The CIFAR-100 dataset is similar to the CIFAR-10 Dataset, which is also a subset of the 80 million tiny images dataset. It contains 50 000 training images labeled equally with 100 classes and 10 000 testing images of the

TABLE X
SNAPSHOT RESULTS OF FASHION-MNIST CLASSIFICATION USING STACKED BLS

No. of Stacked BLS blocks	Accuracy %	No. Para.			Time(s)		Accumulative Time(s)	
	Test	N_f	N_m	N_e	Train	Test	Train	Test
1	91.39%	100	10	9000	135.676*	7.644*	135.676*	7.644*
2	91.47%	8	9	380	2.958*	0.206*	138.634*	7.850*
3	91.53%	8	9	380	3.095*	0.189*	141.729*	8.039*

TABLE XI
PERFORMANCE COMPARISON BETWEEN STACKED CCFBLS AND RESNET IN CIFAR-10

Algorithms	ResNet-32	CCFBLS	Stacked CCFBLS
Testing Accuracy	92.49%	94.29%	**94.78%**
No. of blocks	-	-	3
Accumulative No. of layers	32	26	30
No. of Param.	11.174M	2.933M	2.951M

Fig. 12. Example images selected from the SVHN dataset.

corresponding 100 classes. The image size of the samples is 32×32.

A standard CCFBLS in [16] with 26 layers is trained first. It is constructed with the basic convolutional modules and a fundamental soft-max loss function. Experiments in [16] shows that its performance would significantly improved if additional technical skills and regularizations are embedded into its structures. Therefore, only the basic model of ResNet is compared with our proposed method. For the CIFAR-10 dataset, additional two BLS blocks are trained to approach the residual of the actual output and the CCFBLS model output. Detail results of these experiments are shown in Table XI. We compare our model with ResNet-32, which is one of the most competing methods. It is noted that the proposed Stacked CCFBLS outperforms the ResNet-32 by a large margin in the classification accuracy with huge reduction on the number of parameters. Respectively, for the SVHN dataset, additional three BLS blocks are stacked to build the Stacked CCFBLS network. In Table XII, the comparison between standard ResNet-50 network with our proposed model is reported. A modified structure of CCFBLS with additional group regularization [16] is used to test the capability of Stacked BLS. The modified CCFBLS has 28 convolutional layers and two BLS blocks are further stacked. The results for the CIFAR-100 dataset are shown in Table XIII, with similar conclusions compared with the other tables. Therefore, it can be concluded that our methods achieve a significant performance gain by learning Stacked BLS blocks.

TABLE XII
PERFORMANCE COMPARISON BETWEEN STACKED CCFBLS AND RESNET IN SVHN

Algorithms	ResNet-50	CCFBLS	Stacked CCFBLS
Testing Accuracy	92.06%	96.69%	**97.12%**
No. of blocks	-	-	4
Accumulative No. of layers	50	26	32
No. of Param.	23.521M	2.934M	2.959M

TABLE XIII
PERFORMANCE COMPARISON BETWEEN STACKED CCFBLS AND RESNET IN CIFAR100

Algorithms	ResNet-50	CCFBLS	Stacked CCFBLS
Testing Accuracy	78.51%	81.89%	**88.56%**
No. of blocks	-	-	3
Accumulative No. of layers	50	28	32
No. of Param.	23.706M	3.658M	3.803M

TABLE XIV
PERFORMANCE COMPARISON WITH SELECTED STATE-OF-THE-ART DEEP METHODS

Algorithms	CIFAR-10	CIFAR-100	SVHN
Capsule Net [56]	89.3%	-	95.7%
GDCNN [57]	89.23%	66.7%	-
Meta Net [58]	88%	-	-
CGap [59]	93.59%	73%	96.25%
SReluDCNN [60]	93.02%	70.9%	-
Stacked CCFBLS	**94.78%**	**88.56%**	**97.12%**

In addition, Table XIV shows that, when compared with other deep networks with convolution layers, such as Capsule Net [56], genetic DCNN designer (GDCNN) [57], Boosting-Inspired Deep Learning Ensemble method (Meta Net) [58], an efficient Deep Learning method with pruning (CGap) [59], and a deep learning method with designed rectified linear activation functions (SReluDCNN) [60], the model we proposed could achieve the highest accuracy in all the datasets. It is indicated that our methods outperforms such kind of deep structures.

VI. CONCLUSION

The BLS has been proved to be effective and efficient in [15]. In this article, we reviewed several deep variants of BLS. They have shown significant improvements on various applications. A newly deep variant of the BLS model named Stacked BLS was proposed by stacking several BLS blocks and the training was implemented to approach the residual outputs of each BLS block.

The Stacked BLS model is efficient since the weights and the architecture of the lower Stacked BLS blocks are fixed when the new BLS blocks are added. From this point of view, the Stacked BLS could also be regarded as the increment of layers, and very versatile in comparison with the traditionally fixed architecture. This algorithm fills the gaps in the incremental framework of the BLS. In addition, an incremental algorithm was introduced to increase the enhancement nodes of the newly added BLS block dynamically.

It was illustrated that the ResNet structure is a simple version of the BLS network by setting the feature function to identify mapping.

The performance of Stacked BLS was evaluated and compared with other deep networks, including MLP, SAD, RBM, DBM, ResNet-34, Capsule Net, GDCNN, Meta Net, CGap, SReluDCNN, and other variants of BLS. Experiments on UCI dataset for regression and tests on MNIST, NORB, CIFAR-10, SVHN, and CIFAR-100 datasets for classification are presented and discussed. The performance of Stacked BLS outperforms all other methods.

References

[1] M. Gong, J. Zhao, J. Liu, Q. Miao, and L. Jiao, "Change detection in synthetic aperture radar images based on deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 1, pp. 125–138, Jan. 2016.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2012, pp. 1097–1105.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[4] Z. Liu, G. Lai, Y. Zhang, and C. L. P. Chen, "Adaptive neural output feedback control of output-constrained nonlinear systems with unknown output nonlinearity," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1789–1802, Aug. 2015.

[5] Y. LeCun *et al.*, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*. San Mateo, CA, USA: Morgan Kaufman, 1990, pp. 396–404.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385.

[7] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," 2016. [Online]. Available: https://arxiv.org/abs/1603.08029.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," 2016. [Online]. Available: https://arxiv.org/abs/1603.05027.

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, 2017, pp. 2261–2269.

[10] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger, "Convolutional networks with dense connectivity," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, May 23, 2019, doi: 10.1109/TPAMI.2019.2918284.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, Jul. 2006.

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

[13] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proc. 12th Int. Conf. Artif. Intell. Stat.*, 2009, pp. 448–455.

[14] C. L. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Inf. Sci.*, vol. 275, pp. 314–347, Aug. 2014.

[15] C. L. P. Chen and Z. L. 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.

[16] 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. 1194–1204, Apr. 2019.

[17] J. Du, C.-M. Vong, and C. L. P. Chen, "Novel efficient RNN and LSTM-like architectures: Recurrent and gated broad learning systems and their applications for text classification," *IEEE Trans. Cybern.*, early access, Feb. 20, 2020, doi: 10.1109/TCYB.2020.2969705.

[18] M. L. 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.

[19] J. W. Jin, Z. L. Liu, and C. L. P. Chen, "Discriminative graph regularized broad learning system for image recognition," *Sci. China Inf. Sci.*, vol. 61, Oct. 2018, Art. no. 112209, doi: 10.1007/s11432-017-9421-3.

[20] T. Zhang, X. Wang, X. Xu, and C. L. P. Chen, "GCB-Net: Graph convolutional broad network and its application in emotion recognition," *IEEE Trans. Affective Comput.*, early access, Aug. 27, 2019, doi: 10.1109/TAFFC.2019.2937768.

[21] H. Tang, P. Dong, and Y. Shi, "A construction of robust representations for small data sets using broad learning system," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Dec. 24, 2019, doi: 10.1109/TSMC.2019.2957818.

[22] Y. Yang, Z. Gao, Y. Li, Q. Cai, N. Marwan, and J. Kurths, "A complex network-based broad learning system for detecting driver fatigue from EEG signals," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Dec. 6, 2019, doi: 10.1109/TSMC.2019.2956022.

[23] Z. Gao, W. Dang, M. Liu, W. Guo, K. Ma, and G. Chen, "Classification of EEG signals on VEP-based BCI systems with broad learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Feb. 4, 2020, doi: 10.1109/TSMC.2020.2964684.

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

[25] C. L. P. Chen and J. Z. Wan, "A rapid learning and dynamic stepwise updating algorithm for at 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: doi: 10.1109/3477.740166.

[26] L. Zhang *et al.*, "Analysis and variants of broad learning system," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Jun. 2, 2020, doi: 10.1109/TSMC.2020.2995205.

[27] J. Du, C.-M. Vong, and C. P. Chen, "Novel efficient RNN and LSTM-like architectures: Recurrent and gated broad learning systems and their applications for text classification," *IEEE Trans. Cybern.*, early access, Feb. 20, 2020, doi: 10.1109/TCYB.2020.2969705.

[28] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," *Sens. Actuat. B, Chem.*, vol. 129, no. 2, pp. 750–757, 2008.

[29] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "MS-CELEB-1M: A dataset and benchmark for large-scale face recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 87–102.

[30] T. Li, B. Fang, J. Qian, and X. Wu, "CNN-based broad learning system," in *Proc. IEEE 4th Int. Conf. Signal Image Process. (ICSIP)*, Wuxi, China, 2019, pp. 132–136.

[31] F. Yang, "A CNN-based broad learning system," in *Proc. IEEE 4th Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, 2018, pp. 2105–2109.

[32] W. Yu and C. Zhao, "Broad convolutional neural network based industrial process fault diagnosis with incremental learning capability," *IEEE Trans. Ind. Electron.*, vol. 67, no. 6, pp. 5081–5091, Jun. 2020.

[33] H. Ye, H. Li, and C. P. Chen, "Adaptive deep cascade broad learning system and its application in image denoising," *IEEE Trans. Cybern.*, early access, Mar. 23, 2020, doi: 10.1109/TCYB.2020.2978500.

[34] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, 1994.

[35] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[36] W. N. Venables and B. D. Ripley, *Modern Applied Statistics With S-PLUS*. New York, NY, USA: Springer, 2013.

[37] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[38] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[39] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," in *Department of Information and Computer Science*,

vol. 55. Irvine, CA, USA: Univ. California, 1998. [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html

[40] B. De Mor and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Sci. Pub. Co., 2002.

[41] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[42] S. Feng and C. L. P. Chen, "Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 414–424, Feb. 2020, doi: 10.1109/TCYB.2018.2857815.

[43] 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.

[44] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. Washington, DC, USA, Jun. 2004, pp. 97–104.

[45] 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.*, 2008, pp. 1096–1103.

[46] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). New York, NY, USA: Springer, 2006.

[47] E. Cambria *et al.*, "Extreme learning machines [trends & controversies]," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 30–59, Nov./Dec. 2013.

[48] 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.

[49] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1708.07747.

[50] M. Pal, "Random forest classifier for remote sensing classification," *Int. J. Remote Sens.*, vol. 26, no. 1, pp. 217–222, 2005.

[51] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016. [Online]. Available: arXiv:1602.07360.

[52] D. Schäfer and E. Hüllermeier, "Dyad ranking using plackett–luce models based on joint feature representations," *Mach. Learn.*, vol. 107, no. 5, pp. 903–941, 2018.

[53] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, USA, Rep. TR-2009, 2009.

[54] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011.

[55] Z. Liu, C. L. P. Chen, T. Zhang, and J. Zhou, "Multi-Kernel broad learning systems based on random features: A novel expansion for nonlinear feature nodes," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Bari, Italy, 2019, pp. 193–197, doi: 10.1109/SMC.2019.8914328.

[56] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, vol. 30, I. Guyon *et al.*, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2017, pp. 3856–3866.

[57] B. Ma, X. Li, Y. Xia, and Y. Zhang, "Autonomous deep learning: A genetic DCNN designer for image classification," *Neurocomputing*, vol. 379, pp. 152–161, Feb. 2020.

[58] J. He, I. Pedroza, and Q. Liu, "MetaNet: A boosting-inspired deep learning image classification ensemble technique," in *Proc. Int. Conf. Image Process. Comput. Vis. Pattern Recognit. (IPCV)*, 2019, pp. 51–54.

[59] X. Du, Z. Li, and Y. Cao, "CGaP: Continuous growth and pruning for efficient deep learning," 2019. [Online]. Available: arXiv:1905.11533.

[60] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep learning with S-shaped rectified linear activation units," 2015. [Online]. Available: arXiv:1512.07030.

Zhulin Liu received the bachelor's degree in mathematics from Shandong University, Jinan, China, in 2009, the M.S. degree in mathematics and the Ph.D. degree in software engineering from the University of Macau, Macau, China, in 2011 and 2019, respectively.

She is currently a Postdoctoral Fellow with the South China University of Technology, Guangzhou. Her research area is in broad learning system, computational intelligence, machine learning, and function approximation.

C. L. Philip Chen (Fellow, IEEE) received the M.S. degree in electrical and computer science from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1985, and the Ph.D. degree in electrical and computer science from Purdue University, West Lafayette, IN, USA, in 1988.

He is the Chair Professor and the Dean of the College of Computer Science and Engineering, South China University of Technology, Guangzhou, China. Being a Program Evaluator of the Accreditation Board of Engineering and Technology Education in the U.S., for computer engineering, electrical engineering, and software engineering programs, he successfully architects the University of Macau's Engineering and Computer Science programs receiving accreditations from Washington/Seoul Accord through Hong Kong Institute of Engineers (HKIE), Hong Kong, of which is considered as his utmost contribution in engineering/computer science education for Macau as the former Dean of the Faculty of Science and Technology. His current research interests include cybernetics, systems, and computational intelligence.

Dr. Chen was a recipient of the 2016 Outstanding Electrical and Computer Engineers Award from his alma mater, Purdue University in 1988. He received the IEEE Norbert Wiener Award in 2018 for his contribution in systems and cybernetics, and machine learning. He is also a highly cited researcher by Clarivate Analytics in 2018, 2019, and 2020. He is currently the Editor-in-Chief of the IEEE TRANSACTIONS ON CYBERNETICS, an Associate Editor of the IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE, and IEEE TRANSACTIONS ON FUZZY SYSTEMS. He was the IEEE Systems, Man, and Cybernetics Society President from 2012 to 2013, the Editor-in-Chief of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS from 2014 to 2019. He was the Chair of TC 9.1 Economic and Business Systems of International Federation of Automatic Control from 2015 to 2017. He is a Fellow of AAAS, IAPR, CAA, and HKIE; a member of Academia Europaea, European Academy of Sciences and Arts, and International Academy of Systems and Cybernetics Science.

Shuang Feng received the B.S. degree in mathematics and M.S. degree in applied mathematics from Beijing Normal University, Beijing, China, in 2005 and 2008 respectively, and the Ph.D. degree in computer science from the University of Macau, Macau, China, in January 2019.

From May 2019 to May 2020, he was working as a Postdoctoral Fellow with the Faculty of Science and Technology, University of Macau. He is currently working as an Associate Professor with the School of Applied Mathematics, Beijing Normal University. His research interests focus on fuzzy systems, fuzzy neural networks, and their applications in computational intelligence.

Qiying Feng received the B.S. degree in electronic information engineering from South China Agricultural University, Guangzhou, China, in 2014, and the Ph.D. degree in computer and information science from the University of Macau, Macau, China, in 2020.

She is currently a Postdoctoral Fellow with the South China University of Technology, Guangzhou, China. Her research interests include soft computing and machine learning.

Tong Zhang (Member, IEEE) received the B.S. degree in software engineering from Sun Yat-sen University, Guangzhou, China, in 2009, and the M.S. degree in applied mathematics and the Ph.D. degree in software engineering from the University of Macau, Macau, China, in 2011 and 2016, respectively.

He is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His research interests include affective computing, evolutionary computation, neural network, and other machine learning techniques and their applications.

Dr. Zhang has been working in publication matters for many IEEE conferences.