Analysis and Variants of Broad Learning System

Liang Zhang, *Member, IEEE*, Jiahao Li, Guoqing Lu, Peiyi Shen, Mohammed Bennamoun, *Senior Member, IEEE*, Syed Afaq Ali Shah, *Member, IEEE*, Qiguang Miao, *Senior Member, IEEE*, Guangming Zhu, Ping Li, and Xiaoyuan Lu

***Abstract*—The broad learning system (BLS) is designed based on the technology of compressed sensing and pseudo-inverse theory, and consists of feature nodes and enhancement nodes, has been proposed recently. Compared with the popular deep learning structures, such as deep neural networks, BLS has the ability of rapid incremental learning and can remodel the system without the usual tedious retraining process. However, given that BLS is still in its infancy, it still needs analysis, improvements, and verification. In this article, we first analyze the principle of fast incremental learning ability of BLS in depth. Second, in order to provide an in-depth analysis of the BLS structure, according to the novel structure design concept of deep neural networks, we present four brand-new BLS variant networks and their incremental realizations. Third, based on our analysis of the effect of feature nodes and enhancement nodes, a new BLS structure with a semantic feature extraction layer has been proposed, which is called SFEBLS. The experimental results show that SFEBLS and its variants can increase the accuracy rate on the NORB dataset 6.18%, Fashion-MNIST dataset by 3.15%, ORL data by 5.00%, street view house number dataset by 12.88%, and CIFAR-10 dataset by 18.42%, respectively, and the four brand-new BLS variant networks also obviously outperform the original BLS.**

***Index Terms*—Broad learning system (BLS), deep neural network, four brand-new BLS variant networks, incremental realizations, semantic feature extraction layer.**

I. Introduction

Deep learning models, such as the deep Boltzmann machines (DBMs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) have achieved a prominent performance in computer vision and natural language processing applications. However, CNN or RNN based on deep structures require a computationally expensive training process to achieve highly accurate results. In addition, these architectures demand high-performance hardwares and complex hyper-parameter tuning, which is another bottleneck.

Recently, a new learning model called a broad learning system (BLS), which is based on the principle of compressed sensing and pseudo-inverse theory has been developed. BLS can learn extremely fast. For instance, on some datasets, such as NYU NORB, the speed of BLS is 8600 times of DBM, with approximately the same accuracy. In addition, BLS has the ability to perform incremental learning, which means it can directly add nodes or input data to the original model without reconstructing. Therefore, BLS can simplify the process of parameter tuning and remodeling.

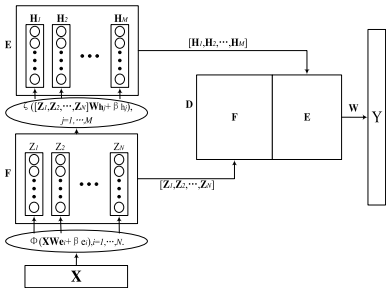


Fig. 1. Structure of BLS. The inputs and outputs are defined as $\mathbf{X} \in \mathbb{R}^{N\times M}$ and $\mathbf{Y} \in \mathbb{R}^{N\times C}$, respectively. $\mathbf{F}$ contains feature nodes $\mathbf{Z}_i$, which are generated by a sparse autoencoder function ϕ based on lasso regression. $\mathbf{E}$ contains enhancement nodes $\mathbf{H}_j$, which are generated by the activation function ζ. Circles in $\mathbf{Z}_i$ and $\mathbf{H}_j$ denote row vectors. $\mathbf{W_e}$, β_e can be computed through ϕ. $\mathbf{W_h}$ and β_h are generated randomly. $\mathbf{E}$ and $\mathbf{F}$ are merged into $\mathbf{D}$, and the $\mathbf{W}$ is trained to output $\mathbf{Y}$.

Fig. 1 shows the original BLS structure, which contains the Feature nodes matrix ($\mathbf{F}$) and the Enhancement nodes matrix ($\mathbf{E}$). Feature nodes are obtained by the sparse autoencoder with lasso regression of the input data, while the enhancement nodes are generated by the nonlinear activation functions. The essential characteristic of BLS is that the limit formula of Moore inverse is adopted and the pseudo-inverse incremental formula is utilized, which can ensure the training accuracy and a fast incremental learning ability.

In fact, as the development of BLS is still in its initial stages, the theoretical basis of this model needs to be further analyzed and consolidated, and no effort has been made to use the popular deep learning feature representation structures (such as pyramid feature representation, dropout mechanism, etc.) to the BLS. In this article, we thoroughly analyze the principle of fast incremental learning of BLS. Several network structures are described in [1], such as the cascade of feature nodes BLS (CFBLS) and the cascade of enhanced node BLS (CEBLS). Although these network variants have achieved better results compared with the baseline BLS, they fail to provide an intuitive explanation of the semantic feature extraction and strategies to prevent overfitting. Inspired by the mechanism used in the popular neural networks, such as DeepLab [2]–[4] for image segmentation, ResNet [5], and DenseNet [6] for image classification, we propose variant network structures for BLS and their incremental algorithm realizations. Our contributions are summarized as follows.

1) An analysis of the principle of fast incremental learning ability.
2) Four new BLS variant networks are proposed based on the Pyramid feature representation and the dropout mechanism and

Manuscript received August 22, 2019; revised December 5, 2019; accepted May 6, 2020. Date of publication June 2, 2020; date of current version December 17, 2021. This work was supported by the Ningbo 2025 Key Project of Science and Technology Innovation under Grant 2018B10071, and in part by the National Key Research and Development Program of China under Grant 2019YFB1311600. This article was recommended by Associate Editor H.-X. Li. *(Corresponding author: Liang Zhang.)*

Liang Zhang, Jiahao Li, Guoqing Lu, Peiyi Shen, Qiguang Miao, and Guangming Zhu are with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China (e-mail: liangzhang@xidian.edu.cn; ljh98214@gmail.com; 345502542@163.com; pyshen@xidian.edu.cn; qgmiao@xidian.edu.cn; gmzhu@xidian.edu.cn).

Mohammed Bennamoun is with the School of Computer Science and Software Engineering, University of Western Australia, Crawley, WA 6009, Australia (e-mail: mohammed.bennamoun@uwa.edu.au).

Syed Afaq Ali Shah is with the Discipline of IT, Statistics, and Mathematics, Murdoch University, Murdoch, WA 6150, Australia (e-mail: afaq.shah@murdoch.edu.au).

Ping Li and Xiaoyuan Lu are with the Laboratory for Data and Virtual, ShangHai BNC, Shanghai 200336, China (e-mail: pli@bnc.org.cn; xylu@bnc.org.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2020.2995205.

Digital Object Identifier 10.1109/TSMC.2020.2995205

their incremental realizations are designed. Our experimental results show the superior performance compared to baseline. This provides further research directions for BLS.

3) Furthermore, we integrate the semantic feature extraction layer with BLS, which leads to some remarkable results on several datasets.

II. Related Work

Over the past decade, deep learning has undergone tremendous technological advancements. The back propagation algorithm proposed by Geoffrey Hinton in 1986 is regarded as the starting point for the rapid development of deep learning. Convolution neural network [7] and LSTM [8] have also broken new ground in Computer Vision and Natural Language Processing research. Some novel network structures, such as ResNet [5] and DenseNet [6] have performed surprisingly well. Some applications based on deep learning [9]–[11] have also achieved excellent results.

Although deep learning structures can achieve a remarkable accuracy and have a great generalization ability, these structures require time-consuming computations and high-performance hardware. Some other models have also made remarkable progress. For instance, Huang *et al.* [12] presented the extreme learning machine (ELM), which has a high learning speed. Then, Chen and Liu [13] proposed the BLS based on the concept of compressed sensing and pseudo-inverse theory. BLS has the ability to perform fast incremental learning. Recently, Chen *et al.* [1] proved the universal approximation capability of BLS, and proposed some simple variations. Han *et al.* [14] illustrated the application of BLS in time series analysis and prediction subsequently. Transductive transfer learning on BLS is illustrated in [15]. Zhang *et al.* [16] demonstrated the efficacy of BLS in facial expression recognition. Song *et al.* [17] combined BLS with dynamical graph CNNs to perform emotion recognition. Feng and Chen [18] proposed the fuzzy BLS, which is used for data regression and classification. Despite all these related works, BLS is still in its infancy and requires a more in-depth analysis and further research.

III. Methodology

A. Demonstration

1) Introduction and Analysis of BLS: The structure of BLS is intuitively shown in Fig. 1. In terms of the training process, BLS employs ridge regression to calculate the pseudo-inverse matrix of $\mathbf{D}$, which is convex and has a good generalization ability. Equations (1)–(4) show the process of training a BLS

$$\mathbf{Y} = \mathbf{DW} \tag{1}$$

$$\mathbf{W} = \mathbf{D}^{+}\mathbf{Y} \tag{2}$$

where $\mathbf{D}$, $\mathbf{W}$, and $\mathbf{Y}$ are the matrices shown in Fig. 1, $\mathbf{D}^{+}$ is the pseudo-inverse matrix of $\mathbf{D}$.

$\mathbf{W}$ can be solved based on ridge regression

$$\arg\min_{\mathbf{W}} : \|\mathbf{DW} - \mathbf{Y}\|_2^2 + \lambda\|\mathbf{W}\|_2^2 \tag{3}$$

Equation (4) can be derived from (3)

$$\mathbf{W} = \left(\lambda\mathbf{I} + \mathbf{DD}^T\right)^{-1}\mathbf{D}^T\mathbf{Y} \tag{4}$$

Comparing (2) and (4), we get

$$\mathbf{D}^{+} = \lim_{\lambda \to 0}\left(\lambda\mathbf{I} + \mathbf{DD}^T\right)^{-1}\mathbf{D}^T \tag{5}$$

where λ is a parameter that stabilizes the system as it approaches 0.

In (4), $\mathbf{W}$ is a matrix learned during the training process which can be computed when $\mathbf{D}$ and $\mathbf{Y}$ are known. In BLS as shown in Fig. 1, $\mathbf{D}$ contains two parts

$$\mathbf{D} = \left[\phi(\mathbf{XW}_{\mathbf{e}_i} + \beta_{e_i})|\zeta\left([\mathbf{Z}_1, \ldots, \mathbf{Z}_N]\mathbf{W}_{\mathbf{h}_j} + \beta_{h_j}\right)\right] \quad i = 1, \ldots, N, j = 1, \ldots, M \tag{6}$$

$\mathbf{W_e}$ is a matrix which is learned using the sparse autoencoder based on lasso regression, and $\mathbf{W_h}$ is the randomly generated matrix weight. $\mathbf{W_e}$ and $\mathbf{W_h}$ are kept constant during the training process. Now if $\mathbf{W_e}$ and $\mathbf{W_h}$ are learned through (1)–(4) instead of autoencoder, then

$$\left[\left(\mathbf{XW}_{\mathbf{e}_i} + \beta_{e_i}\right)|\zeta\left([\mathbf{Z}'_1, \ldots, \mathbf{Z}'_N]\mathbf{W}_{\mathbf{h}_j} + \beta_{h_j}\right)\right]\mathbf{W} = \mathbf{Y} \tag{7}$$

where $\mathbf{W_e}$, $\mathbf{W_h}$, and $\mathbf{W}$ are three independent variables. $\mathbf{Z}'_i$ is the new feature node.

Obviously, it is impossible to solve these three matrices ($\mathbf{W_e}$, $\mathbf{W_h}$, $\mathbf{W}$) simultaneously with only one equation. Therefore, BLS requires $\mathbf{D}$ to be a constant matrix before training.

2) Analysis and Consolidation of the Dynamic Incremental Learning Ability of BLS: The dynamic incremental learning ability of BLS is based on the pseudo-inverse theory, and the implementation details are summarized as follows.

We define matrix $\mathbf{B} \in M_{p\times q}(\mathbb{C})$, vector $a \in \mathbb{C}^P$, and matrix $\mathbf{A} = (\mathbf{B}, a) \in M_{p\times(q+1)}(\mathbb{C})$. $\mathbf{B}$ represents the matrix $\mathbf{D}$ before incremental process, a denotes the incremental part and $\mathbf{A}$ denotes the matrix $\mathbf{D}$ after the incremental process. d, c, and b are intermediate variables, where $d = \mathbf{B}^{+}a$, $c = a - \mathbf{B}d$. $\mathbb{C}$ means the complex set, so $\mathbf{B}$ is a complex matrix and the size is $p \times q$. a is a complex vector

$$b = \begin{cases} c^{+} & \text{if } c \neq 0 \\ \left(1 + |d|^2\right)^{-1} d^{*}\mathbf{B}^{+} & \text{if } c = 0 \end{cases}$$

where $\mathbf{B}^{+}$ is the pseudo-inverse matrix of $\mathbf{B}$.

Based on the above definitions [19], In BLS they claim that the pseudo-inverse matrix of $\mathbf{A}$ can be represented as

$$\mathbf{A}^{+} = \begin{pmatrix} \mathbf{B}^{+} - db \\ b \end{pmatrix}. \tag{8}$$

If (8) is the pseudo-inverse of $\mathbf{A}$, then $\mathbf{A}^{+}$ can be computed iteratively from $\mathbf{B}^{+}$ and a without pseudo-inverse computation through (5), which means that $\mathbf{W}$ can be easily updated according to (2) based on $\mathbf{A}^{+}$. Using this procedure, BLS can realize incremental learning. Unfortunately, the detailed proof of (8) for the pseudo-inverse of $\mathbf{A}$ is not provided in the original BLS [13]. To ensure the incremental learning ability of the proposed variants (discussed in Section III-B), we list four lemmas that are helpful to prove (8), i.e., the pseudo-inverse of $\mathbf{A}$.

Lemma 1: If $\mathbf{A}^{+}$ is the pseudo-inverse of matrix $\mathbf{A}$, they must satisfy

$$\mathbf{AA}^{+}\mathbf{A} = \mathbf{A} \tag{9}$$

$$\mathbf{A}^{+}\mathbf{AA}^{+} = \mathbf{A}^{+} \tag{10}$$

$$\mathbf{AA}^{+} \in \mathbf{H} \tag{11}$$

$$\mathbf{A}^{+}\mathbf{A} \in \mathbf{H} \tag{12}$$

where $\mathbf{H}$ means Hermitian Matrix set.

Lemma 2: Given $a \in \mathbb{C}^P$, if a has pseudo-inverse vector, which is a^{+}, then we have $a^{+} = \|a\|_2^{-2}a^{*}$, where $a^{*} = \bar{a}^T$.

Lemma 3 [20]: Given $\mathbf{B} \in M_{p\times q}(\mathbb{C})$ and $\mathbf{B} \neq 0$, $\mathbf{B}$ has pseudo-inverse matrix $\mathbf{B}^{+}$. Let $\mathbf{B} = \mathbf{XY}$, where $\mathbf{X}$ and $\mathbf{Y}$ are, respectively, the column full matrix and row full rank matrix, then we have

$$\mathbf{B}^{+} = \mathbf{Y}^{*}\left(\mathbf{YY}^{*}\right)^{-1}\left(\mathbf{X}^{*}\mathbf{X}\right)^{-1}\mathbf{X}^{*}. \tag{13}$$

Lemma 4: Given $\mathbf{B} \in M_{p*q}(\mathbb{C})$ and $\mathbf{B} \neq 0$, $\mathbf{B}$ has pseudo-inverse matrix $\mathbf{B}^+$, then we have

$$\left(\mathbf{B}^+\right)^*\mathbf{B}^+\mathbf{B} = \left(\mathbf{B}^+\right)^*. \tag{14}$$

Based on the above lemmas, we can prove that (8) is the pseudo-inverse matrix of A. On this basis, we propose new variants of BLS.

B. Novel Network Variants of BLS

As a newly proposed network model, BLS has room for improvement, especially in its network variants. There are only a few cascades models presented in [1]. However, their experimental results show that the research on network variants is feasible and valuable. In order to enrich the network variants of BLS, we propose four network variants and demonstrate their effects on accuracy. These variants are CFBLS-pyramid, CEBLS-dropout, CFBLS-dense, and CEBLS-dense. When we design our network variants, we first consider the basic structure of BLS. The feature nodes of BLS are mainly used to extract sparse features, we propose CFBLS-pyramid and CFBLS-dense to extract hierarchical features and iterative features, respectively. The enhancement nodes of BLS are used to increase the number of parameters in the model to improve the regression accuracy, which may however cause overfitting. We therefore propose CEBLS-dropout and CEBLS-dense to prevent overfitting. Each network variant has its own characteristics.

1) Cascade of Feature Nodes in Pyramid Structure (CFBLS-Pyramid): Features extracted from different depths have a unique semantic information, and more comprehensive results can be obtained by combining diverse features.

The CFBLS mentioned in [1] can extract hierarchical features. Unfortunately, CFBLS only performs a random feature extraction on the input data, and the features at different levels are based on this feature extraction. Therefore, CFBLS aggravates the impact of random errors. To address this, we propose a pyramid network structure, which not only extracts features at different levels but also introduces multiple feature extraction of input data to reduce the random errors.

In CFBLS-pyramid, as illustrated in Fig. 2, the structure of the feature nodes is just like a pyramid. In the first layer of the structure, only one group of feature nodes is employed. Then, in each subsequent layer, the number of feature nodes are increased. Each layer of this structure cascades a group of feature nodes one after the other.

The feature node $\mathbf{Z}_{ij}$ can be denoted as

$$\mathbf{Z}_{ij} = \begin{cases} \phi\left(\mathbf{X}\mathbf{W}_{\mathbf{e}_{ij}} + \beta_{e_{ij}}\right) & i = 1, \ldots, N \; j = 1 \\ \phi\left(\mathbf{Z}_{i(j-1)}\mathbf{W}_{\mathbf{e}_{ij}} + \beta_{e_{ij}}\right) & i = 1, \ldots, N \; j = 2, \ldots, i \end{cases} \tag{15}$$

where the "i" denotes the ith layer of the pyramid and the "j" denotes the jth feature node.

Next, all feature nodes: $\mathbf{Z}^{N'} = [\mathbf{Z}_{11}, \mathbf{Z}_{21}, \mathbf{Z}_{22}, \ldots, \mathbf{Z}_{NN}]$, where $N' = N(N+1)/2$ can be fed into an activation function ζ to generate enhancement nodes $\mathbf{H}^{M} = [\mathbf{H}_1, \ldots, \mathbf{H}_M]$, where

$$\mathbf{H}_k = \zeta\left(\mathbf{Z}^{N'}\mathbf{W}_{\mathbf{h}_k} + \beta_{h_k}\right), \; k = 1, \ldots, M. \tag{16}$$

Finally, the system model of this cascade of feature nodes in a pyramid structure can be summarized as

$$\mathbf{Y} = \left[\mathbf{Z}^{N'}|\mathbf{H}^{M}\right]\mathbf{W}_{N'}^{M}, \tag{17}$$

where $\mathbf{W}_{N'}^{M}$, is computed through (2).

2) Cascade of Enhancement Nodes With Dropout (CEBLS-Dropout): Overfitting is a challenging problem, which overcomplicates the training process and leads to poor results when insufficient data is available or too much noise. Inevitably, BLS may exacerbate overfitting due to the redundancy in the linear space due to expansion. Fortunately, [21] has proposed the technique called dropout, which can effectively alleviate overfitting and has been widely used in deep learning research. CEBLS mentioned in [1] performs better than the baseline BLS network. As CEBLS is a deep cascade structure network, overfitting is inevitable in this structure. Therefore, we introduce dropout in CEBLS to alleviate the adverse impact caused by overfitting.

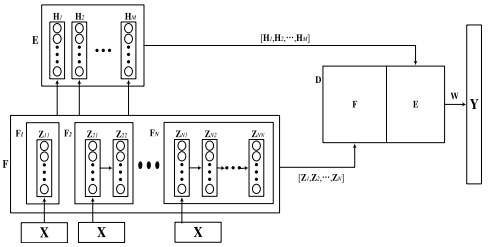


Fig. 2. CFBLS-pyramid: cascade of feature nodes in pyramid ways.

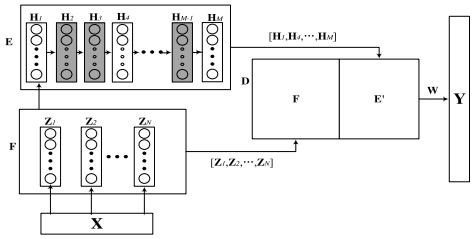


Fig. 3. CEBLS-dropout: cascade of enhancement nodes with dropout, and gray enhancement nodes represent inactive nodes which are not connected with output.

As in Fig. 3, not all the enhancement nodes are connected with the output, we set the value of a few enhancement nodes to zero. Therefore, the network is formulated as follows:

$$\mathbf{Y} = \left[\mathbf{Z}^{N}|\mathbf{H}^{M'}\right]\mathbf{W}_{N}^{M'} \tag{18}$$

where $\mathbf{Z}^{N} = [\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_N]$, and $\mathbf{H}^{M} = [\mathbf{H}_1, \ldots, \mathbf{H}_M]$. We set some nodes in $\mathbf{H}^{M}$ to zero according to a certain probability to form $\mathbf{H}^{M'}$. $\mathbf{W}_{N}^{M'}$ is computed through (2).

3) Cascade of Feature Nodes With Dense Connection (CFBLS-Dense): Enhancing feature propagation and feature reusability is an effective way to obtain better results. Huang *et al.* [6] proposed a new network structure, DenseNet, which has a dense connections between all the front and back layers. Inspired by DenseNet and CFBLS [1], we propose a new variant architecture of BLS. In this architecture, the adoption of dense connections can feed each feature node with all feature representations that are extracted by the former nodes. The structure of the network is shown in Fig. 4.

The first feature nodes are connected with the input

$$\mathbf{Z}_1 = \phi\left(\mathbf{X}\mathbf{W}_{\mathbf{e}_1} + \beta_{e_1}\right). \tag{19}$$

Then, the second feature nodes are connected by input data $\mathbf{X}$ and the first feature nodes $\mathbf{Z}_1$. They are generated by the following equation:

$$\mathbf{Z}_2 = \phi\left(\left[\mathbf{X}, \mathbf{Z}_1\right]\mathbf{W}_{\mathbf{e}_2} + \beta_{e_2}\right). \tag{20}$$

The other feature nodes are given by

$$\mathbf{Z}_i = \phi\left(\left[\mathbf{X}, \mathbf{Z}_1, \ldots, \mathbf{Z}_{i-1}\right]\mathbf{W}_{\mathbf{e}_i} + \beta_{e_i}\right), \; 2 \leq i \leq N. \tag{21}$$

Next, we have the general representation of the enhancement nodes

$$\mathbf{H}_j = \zeta\left(\mathbf{Z}^{N}\mathbf{W}_{\mathbf{h}_j} + \beta_{h_j}\right), \; j = 1, \ldots, M. \tag{22}$$

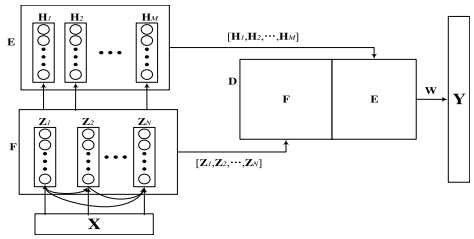


Fig. 4. CFBLS-dense: cascade of feature nodes with dense connections.

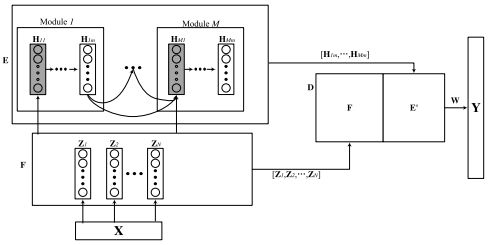


Fig. 5. CEBLS-dense: cascade of enhancement nodes with dense connections and gray enhancement nodes are not connected with output.

In the above equation, $\mathbf{W_h}$ and β_h are randomly generated. Consequently, the nodes $\mathbf{Z}^N = [\mathbf{Z}_1, \ldots, \mathbf{Z}_N]$ and $\mathbf{H}^M = [\mathbf{H}_1, \ldots, \mathbf{H}_M]$ are connected with the output.

4) Cascade of Enhancement Nodes With Dense Connection (CEBLS-Dense): Like CFBLS-dense, the enhancement nodes in CEBLS-dense also adopt dense connections. We divide the enhancement nodes in CEBLS-dense into M modules, and each module contains m enhancement nodes. The input of the first enhancement node of each module is composed of all the feature nodes and the last enhancement node in the previous modules, and this structure cascades the module of enhancement nodes one after the other. In addition, only the last node for each module is fed into the output. Obviously, redundancy typically occurs in the enhancement part of the BLS, thus CEBLS-dense can avoid overfitting effectively.

As seen in Fig. 5, the feature nodes are connected with input $\mathbf{X}$, they are generated by the following equations:

$$\mathbf{Z}_i = \phi\left(\mathbf{X}\mathbf{W}_{\mathbf{e}_i} + \beta_{e_i}\right), \ i = 1, \ldots, N. \tag{23}$$

For enhancement nodes, $\mathbf{H}_{ij}$ can be denoted as

$$\mathbf{H}_{jk} = \begin{cases} \zeta\left(\mathbf{Z}^N\mathbf{W}_{\mathbf{h}_{jk}} + \beta_{h_{jk}}\right) & j = 1, \ k = 1 \\ \zeta\left(\left[\mathbf{Z}^N, \mathbf{H}_{1m}, \ldots, \mathbf{H}_{(j-1)m}\right]\mathbf{W}_{\mathbf{h}_{jk}} + \beta_{h_{jk}}\right) & j = 2, \ldots, M \ k = 1 \\ \zeta\left(\mathbf{H}_{j(k-1)}\mathbf{W}_{\mathbf{h}_{jk}} + \beta_{h_{jk}}\right) & j = 1, \ldots, M \ k = 2, \ldots, m \end{cases} \tag{24}$$

where $\mathbf{W_h}$ and β_h are randomly generated.

To avoid overfitting, not all the enhancement nodes are connected with the output. In this model, we select one enhancement node for each module. Therefore, the output data is connected with the feature nodes and M enhancement nodes. The enhancement nodes which are connected with the output can be represented as $\mathbf{H}^M = [\mathbf{H}_{1m}, \mathbf{H}_{2m}, \ldots, \mathbf{H}_{Mm}]$.

C. Semantic Feature Extraction

Obviously, using a complex deep network with bulky GPUs is not cost-effective for simple tasks. BLS can run faster on CPUs and is ideal under these circumstances.

However, the original BLS does not explore the details of the effects of the feature nodes. Feature nodes in BLS use compressed sensing technology to reduce the dimension of complex signals, and the results achieve orthogonality, thus making the classification more accurate and ensuring rapid increments. Nevertheless, it is known that an image with semantic information is not a pure data matrix. Feature nodes in BLS are not able to extract semantic information from images, such as texture information, edge information, etc.

Image semantic feature extraction has always been a major focus in computer vision and image processing research. Some renowned methods of feature extraction, such as histogram of oriented gradient (HOG) [22], local binary patterns (LBPs) [23], have achieved a remarkable performance for object and face recognition. Based on the structure of our proposed BLS network variants, which are inspired by deep learning neural networks (such as DenseNet), our hypothesis suggested that if the data is processed by the semantic feature extraction, and then combined with the feature nodes to extract sparse features of the data, the results can improve compared to the baseline. Therefore, we add a semantic feature extraction layer, and the new model is named as SFEBLS. Semantic feature extraction layer and the BLS network structure are separate, i.e., the former is used for extracting semantic features, and the latter is used for processing and analyzing the semantic features.

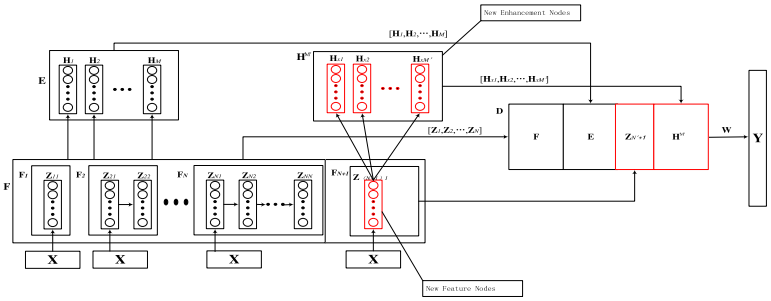
Algorithm 1 CFBLS-Pyramid: Increment of N+1 Feature Nodes

Input: training samples $\mathbf{X}$
Output: $\mathbf{W}$

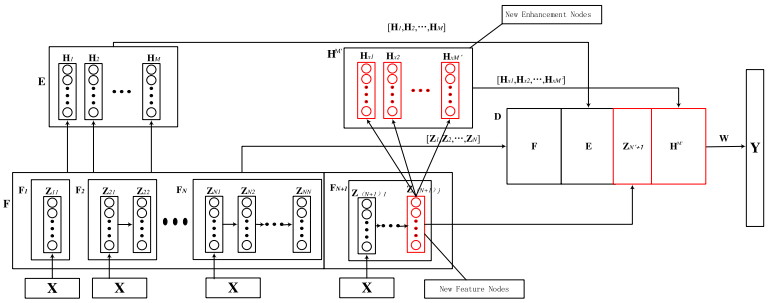
for $i = 1; i \leq N$ **do**
  Random $\mathbf{W}_{\mathbf{e}_{i1}}, \beta_{e_{i1}}$;
  Calculate $\mathbf{Z}_{i1} = \phi(\mathbf{X}\mathbf{W}_{\mathbf{e},ij} + \beta_{e,ij})$;
  for $j = 2; j \leq i$ **do**
    Random $\mathbf{W}_{\mathbf{e}_{ij}}, \beta_{e_{ij}}$;
    Calculate $\mathbf{Z}_{ij} = \phi(\mathbf{Z}_{ij-1}\mathbf{W}_{\mathbf{e}_{ij}} + \beta_{e_{ij}})$;
  end for
end for
All feature nodes:$\mathbf{Z}^{N'} = [\mathbf{Z}_{11}, \mathbf{Z}_{21}, \mathbf{Z}_{22}, \ldots, \mathbf{Z}_{NN}]$;
for $k = 1; k \leq M$ **do**
  Random $\mathbf{W}_{\mathbf{h}_k}, \beta_{h_k}$;
  Calculate $\mathbf{H}_k = \zeta(\mathbf{Z}^{N'}W_{\mathbf{h}_k} + \beta_{h_k})$
end for
All enhancement nodes: $\mathbf{H}^M = [\mathbf{H}_1, \ldots, \mathbf{H}_M]$
Set $\mathbf{D}^M_{N'} = [\mathbf{Z}^{N'}|\mathbf{H}^M]$
Calculate $(\mathbf{D}^M_{N'})^+$ and $\mathbf{W}^M_{N'}$
while *The number of feature nodes increased is less than the set number* **do**
  Random $\mathbf{W}_{\mathbf{e}_{N'+1}}, \beta_{e_{N'+1}}$;
  if $\mathbf{Z}_{N'+1}$ is the first feature node of the layer **then**
    Calculate $\mathbf{Z}_{N'+1} = \phi(\mathbf{X}\mathbf{W}_{\mathbf{e}_{N'+1}} + \beta_{e_{N'+1}})$;
  else
    Calculate $\mathbf{Z}_{N'+1} = \phi(\mathbf{Z}_{N'}\mathbf{W}_{\mathbf{e}_{N'+1}} + \beta_{e_{N'+1}})$;
  end if
  Random $\mathbf{W}_{\mathbf{h}_{xp}}, \beta_{e_{xp}}, p = 1, \ldots, M'$;
  Calculate $\mathbf{H}^{M'} = [\zeta(\mathbf{Z}_{N'+1}\mathbf{W}_{h_{x1}} + \beta_{h_{x1}}), \ldots, \zeta(\mathbf{Z}_{N'+1}\mathbf{W}_{h_{xM'}} + \beta_{h_{xM'}})]$
  Update $\mathbf{D}^M_{N'+1} = [\mathbf{D}^M_{N'}|\mathbf{Z}_{N'+1}|\mathbf{H}^{M'}]$
  Calculate $(\mathbf{D}^M_{N'+1})^+$ by Equation(28)$\sim$(29)
  Calculate $\mathbf{W}^M_{N'+1}$ by Equation(30)
  N'=N'+1;
end while
return($\mathbf{W}^M_{N'+1}$)

IV. Incremental Learning

Incremental learning is an important characteristic of BLS. In this section, the incremental algorithm of BLS network variants will be presented. We propose four network variants of BLS, including CFBLS-pyramid, CEBLS-dense, CFBLS-dense, and



(a)



(b)

Fig. 6. Incremental learning for CFBLS-pyramid: increment of additional feature nodes. (a) New feature node is the first node of the last layer. (b) New feature node is not the first node of the last layer.

Algorithm 2 CFBLS-Dense: Increment of $N+1$ Feature Nodes

Input: training samples $\mathbf{X}$
Output: $\mathbf{W}$

Random $\mathbf{W}_{\mathbf{e}_1}, \beta_{e_1}$;
Calculate $\mathbf{Z_1} = \phi(\mathbf{X}\mathbf{W}_{\mathbf{e}_1} + \beta_{e_1})$;
for $i = 2; i \leq N$ **do**
  Random $\mathbf{W}_{\mathbf{e}_i}, \beta_{e_i}$;
  Calculate $\mathbf{Z_i} = \phi([\mathbf{X}, \mathbf{Z_1}, \ldots, \mathbf{Z_{i-1}}]\mathbf{W}_{\mathbf{e}_i} + \beta_{e_i})$;
end for
All feature nodes:$\mathbf{Z}^N = [\mathbf{Z_1}, \ldots, \mathbf{Z_N}]$;
for $j = 1; j \leq M$ **do**
  Random $\mathbf{W}_{\mathbf{h}_j}, \beta_{h_j}$;
  Calculate $\mathbf{H}_j = \zeta(\mathbf{Z}^N\mathbf{W}_{\mathbf{h}_j} + \beta_{h_j})$
end for
All enhancement nodes: $\mathbf{H}^M = [\mathbf{H}_1, \ldots, \mathbf{H}_M]$
Set $\mathbf{D}_N^M = [\mathbf{Z}^N|\mathbf{H}^M]$
Calculate $(\mathbf{D}_N^M)^+$ and $\mathbf{W}_N^M$
while *The number of feature nodes increased is less than the set number* **do**
  Random $\mathbf{W}_{\mathbf{e}_{N+1}}, \beta_{e_{N+1}}$;
  Calculate$\mathbf{Z_{N+1}} = \phi([\mathbf{X}, \mathbf{Z}^N]\mathbf{W}_{\mathbf{e}_{N+1}} + \beta_{e_{N+1}})$
  Random $\mathbf{W}_{\mathbf{h}_{xp}}, \beta_{e_{xp}}, p = 1, \ldots, M'$;
  Calculate $\mathbf{H}^{M'} = [\zeta(\mathbf{Z}_{N'+1}\mathbf{W}_{h_{x1}} + \beta_{h_{x1}}), \ldots, \zeta(\mathbf{Z}_{N'+1}\mathbf{W}_{h_{xM'}} + \beta_{h_{xM'}})]$
  Update $\mathbf{D}_{N+1}^M = [\mathbf{D}_N^M|\mathbf{Z}_{N+1}|\mathbf{H}^M]$
  Calculate $(\mathbf{D}_{N+1}^M)^+$ by Equation(28)$\sim$(29)
  Calculate $\mathbf{W}_{N+1}^M$ by Equation(30)
  N=N+1;
end while
return($\mathbf{W}_{N+1}^M$)

CEBLS-dropout. It should be noted that the incremental part can only be placed after the original matrix. $A = (\mathbf{B}, a)$ in (8) shows that the incremental part a must be placed after the original matrix $\mathbf{B}$.

There are three types of incremental algorithms: 1) the increment of additional enhancement nodes; 2) the increment of additional feature nodes; and 3) the increment of additional input data. Chen and Liu [13] has introduced the basic implementation of the incremental algorithm. In this article, we introduce incremental algorithms of the four proposed network variants.

A. Incremental Learning for CFBLS-Pyramid: Increment of Additional Feature Nodes

CFBLS-pyramid, as the network variant of BLS, inherits the incremental learning perfectly. Because the incremental learning on the enhancement nodes and the input data is similar to the BLS, Compared with BLS, CFBLS-pyramid only makes changes to the acquisition mode of the feature nodes, so its input data and increment of enhancement nodes do not change much. Same for CFBLS-dense and CEBLS-dense. Therefore, we only focus on the incremental learning of feature nodes in CFBLS-pyramid.

Assume that the initial structure consists of N' feature nodes which are divided into N layers and M enhancement nodes. Considering that the $(N'+1)$th feature nodes are added. There are two possible scenarios, if the new feature node is the first feature node of the layer, as shown in Fig. 6(a), it can be denoted as

$$\mathbf{Z}_{N'+1} = \phi\left(\mathbf{X}\mathbf{W}_{\mathbf{e}_{N'+1}} + \beta_{e_{N'+1}}\right). \tag{25}$$

In other cases, as shown in Fig. 6(b), it can be denoted as

$$\mathbf{Z}_{N'+1} = \phi\left(\mathbf{Z}_{N'}\mathbf{W}_{\mathbf{e}_{N'+1}} + \beta_{e_{N'+1}}\right) \tag{26}$$

where $\mathbf{Z}_{N'}$ is the last feature node in initial structure.

When the number of feature nodes increases, the number of enhancement nodes also increases by M', which is a predetermined constant. The additional enhancement nodes are randomly generated as follows:

$$\mathbf{H}^{M'} = \left[\zeta\left(\mathbf{Z}_{N'+1}\mathbf{W}_{h_{x1}} + \beta_{h_{x1}}\right), \ldots, \zeta\left(\mathbf{Z}_{N'+1}\mathbf{W}_{h_{xM'}} + \beta_{h_{xM'}}\right)\right] \tag{27}$$

Algorithm 3 CEBLS-Dense: Increment of N+1 Enhancenment Nodes

Input: training samples $\mathbf{X}$
Output: $\mathbf{W}$

for $i = 1; i \leq N$ **do**
  Random $\mathbf{W_{e,}}_i$, $\beta_{e,i}$;
  Calculate $\mathbf{Z_i} = \phi(\mathbf{X}\mathbf{W_{e,i}} + \beta_{e,i})$;
end for
All feature nodes:$\mathbf{Z^N} = [\mathbf{Z_1}, \ldots, \mathbf{Z_N}]$;
for $j = 1; j \leq M$ **do**
  for $k = 1; k \leq m$ **do**
    if $j = 1$ AND $k = 1$ **then**
      Random $\mathbf{W_{h,}}_1$, $\beta_{h,1}$;
      Calculate $\mathbf{H_{11}} = \zeta(\mathbf{Z^N}\mathbf{W_{h,11}} + \beta_{h,11})$;
    else if $j \neq 1$ AND $k = 1$ **then**
      Random $\mathbf{W_{h,}}_{jk}$, $\beta_{h,jk}$;
      Calculate $\mathbf{H_{jk}} = \zeta([\mathbf{Z^N}, \mathbf{H_{1m}}, \ldots, \mathbf{H_{(j-1)m}}]\mathbf{W_{h,jk}} + \beta_{h,jk})$;
    else
      Random $\mathbf{W_{h,}}_{jk}$, $\beta_{h,jk}$;
      Calculate $\mathbf{H_{jk}} = \zeta(\mathbf{H_{j(k-1)}}\mathbf{W_{h,jk}} + \beta_{h,jk})$;
    end if
  end for
end for
All enhancement nodes: $\mathbf{H}^M = [\mathbf{H}_{1m}, \ldots, \mathbf{H}_{Mm}]$
Set $\mathbf{D}_N^M = [\mathbf{Z}^N|\mathbf{H}^M]$
Calculate $(\mathbf{D}_N^M)^+$ and $\mathbf{W}_N^M$
while *The number of enhancement nodes increased is less than the set number* **do**
  for $k = 1; k \leq m$ **do**
    if $k = 1$ **then**
      Random $\mathbf{W_{h,}}_{(M+1)1}$, $\beta_{h,(M+1)1}$;
      Calculate $\mathbf{H_{(M+1)1}} = \zeta([\mathbf{Z^N}, \mathbf{H^M}]\mathbf{W_{h,(M+1)1}} + \beta_{h,(M+1)1})$;
    else
      Random $\mathbf{W_{h,}}_{(M+1)k}$, $\beta_{h,(M+1)k}$;
      Calculate $\mathbf{H_{(M+1)k}} = \zeta(\mathbf{H_{(M+1)(k-1)}}\mathbf{W_{h,(M+1)k}} + \beta_{h,(M+1)k})$;
    end if
  end for
  Update $\mathbf{D}_N^{M+1} = [\mathbf{D}_N^M|\mathbf{H}_{(M+1)m}]$,
  Calculate $(\mathbf{D}_N^{M+1})^+$ by Equation(33)$\sim$(34)
  Calculate $\mathbf{W}_N^{M+1}$ by Equation(35)
  M=M+1;
end while
return($\mathbf{W}_{N+1}^M$)

where $\mathbf{W_h}$ and β_e are randomly generated. Then, we can get $\mathbf{D}_{N'+1}^M = [\mathbf{D}_{N'}^M|\mathbf{Z}_{N'+1}|\mathbf{H}^{M'}]$, and the upgraded pseudoinverse matrix of it can be computed by the following equation:

$$\left(\mathbf{D}_{N'+1}^M\right)^+ = \begin{bmatrix} \left(\mathbf{D}_{N'}^M\right)^+ - \mathbf{Q}\mathbf{B}^T \\ \mathbf{B}^T \end{bmatrix} \tag{28}$$

where $\mathbf{Q} = (\mathbf{D}_{N'}^M)^+[\mathbf{Z}_{N'+1}|\mathbf{H}^{M'}]$

$$\mathbf{B}^T = \begin{cases} \mathbf{C}^+ & \text{if } \mathbf{C} \neq 0 \\ \left(1 + \mathbf{Q}^T\mathbf{Q}\right)^{-1}\mathbf{Q}^T\left(\mathbf{D}_{N'}^M\right)^+ & \text{if } \mathbf{C} = 0 \end{cases} \tag{29}$$

and $\mathbf{C} = [\mathbf{Z}_{N'+1}|\mathbf{H}^{M'}] - \mathbf{D}_{N'}^M\mathbf{Q}$

So, the new weights are

$$\mathbf{W}_{N'+1}^M = \begin{bmatrix} \mathbf{W}_{N'}^M - \mathbf{Q}\mathbf{B}^T\mathbf{Y} \\ \mathbf{B}^T\mathbf{Y} \end{bmatrix}. \tag{30}$$

The incremental algorithm of the increment feature node is shown in Algorithm 1.

Algorithm 4 CEBLS-Dropout: Increment of N+1 Enhancenment Nodes

Input: training samples $\mathbf{X}$
Output: $\mathbf{W}$

for $i = 1; i \leq N$ **do**
  Random $\mathbf{W_{e,}}_i$, $\beta_{e,i}$;
  Calculate $\mathbf{Z_i} = \phi(\mathbf{X}\mathbf{W_{e,i}} + \beta_{e,i})$;
end for
All feature nodes:$\mathbf{Z^N} = [\mathbf{Z_1}, \ldots, \mathbf{Z_N}]$;
for $j = 1; j \leq M$ **do**
  Random $\mathbf{W_{h_j}}$, β_{h_j};
  Calculate $\mathbf{H}_j = \zeta(\mathbf{Z}^N\mathbf{W_{h_j}} + \beta_{h_j})$
  Calculate $r = p\{1, 0\}$, r is 1 or 0 generated randomly with probability p.
  Calculate $\mathbf{H}_{(j)}^{'} = r\mathbf{H_{(j)}}$
end for
All enhancement nodes: $\mathbf{H}^M = [\mathbf{H}_1^{'}, \ldots, \mathbf{H}_M^{'}]$
Set $\mathbf{D}_N^M = [\mathbf{Z}^N|\mathbf{H}^M]$
Calculate $(\mathbf{D}_N^M)^+$ and $\mathbf{W}_N^M$
while *The number of enhancement nodes increased is less than the set number* **do**
  Random $\mathbf{W_{h,}}_{(M+1)}$, $\beta_{h,(M+1)}$;
  Calculate $\mathbf{H_{(M+1)}} = \zeta([\mathbf{Z^N}, \mathbf{H^M}]\mathbf{W_{h,(M+1)}} + \beta_{h,(M+1)})$;
  Calculate $\mathbf{H}_{(M+1)}^{'} = r\mathbf{H_{(M+1)}}$
  Update $\mathbf{D}_N^{M+1} = [\mathbf{D}_N^M|\mathbf{H}_{(M+1)}^{'}]$,
  Calculate $(\mathbf{D}_N^{M+1})^+$ by Equation(33)$\sim$(34)
  Calculate $\mathbf{W}_N^{M+1}$ by Equation(35)
  M=M+1;
end while
return($\mathbf{W}_{N+1}^M$)

B. Incremental Learning for CFBLS-Dense: Increment of Additional Feature Nodes

The incremental algorithm of CFBLS-dense is similar to the CFBLS-pyramid. Next, we will describe the incremental algorithm of CFBLS-dense.

Assume that the initial structure consists of N feature nodes and M broad enhancement nodes. Considering that the $(N+1)$th feature nodes are added, unlike CFBLS-pyramid, there is only one situation for CFBLS-dense (see Fig. 7), and it can be denoted as

$$\mathbf{Z_{N+1}} = \phi\left(\left[\mathbf{X}, \mathbf{Z}^N\right]\mathbf{W}_{\mathbf{e}_{N+1}} + \beta_{e_{N+1}}\right) \tag{31}$$

where $\mathbf{Z}^N$ is a set of feature nodes of the initial structure.

Similar to CFBLS-pyramid, the new feature nodes in CFBLS-dense also bring new enhancement nodes. The number of new enhancement nodes is M' which is also determined, the corresponding enhancement nodes are randomly generated as follows:

$$\mathbf{H}^{M'} = \left[\zeta\left(\mathbf{Z}_{N'+1}\mathbf{W}_{h_{x1}} + \beta_{h_{x1}}\right), \ldots, \zeta\left(\mathbf{Z}_{N'+1}\mathbf{W}_{h_{xM'}} + \beta_{h_{xM'}}\right)\right] \tag{32}$$

where $\mathbf{W_h}$ and β_e are randomly generated. Then, we can get $\mathbf{D}_{N+1}^M = [\mathbf{D}_N^M|\mathbf{Z}_{N+1}|\mathbf{H}_P]$, and the upgraded pseudo-inverse matrix $(\mathbf{D}_{N+1}^M)^+$ can be computed by (28) and (29) The new weights can be obtained using (30).

The incremental algorithm of the increment feature node is shown in Algorithm 2.

C. Incremental Learning for CEBLS-Dense: Increment of Additional Enhancement Nodes

Enhancement nodes are very important in BLS, CEBLS-dense is a new structure whose enhancement nodes are connected in a dense

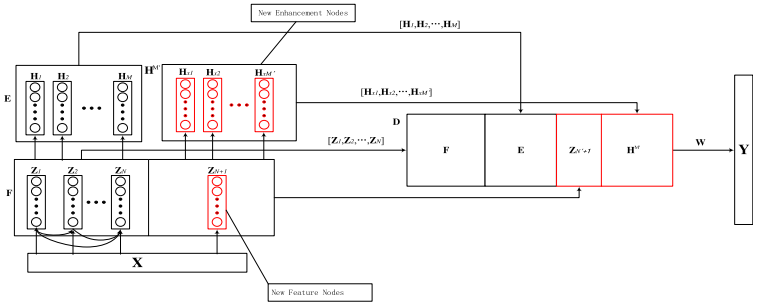


Fig. 7. Incremental learning for CFBLS-dense: increment of additional feature nodes.

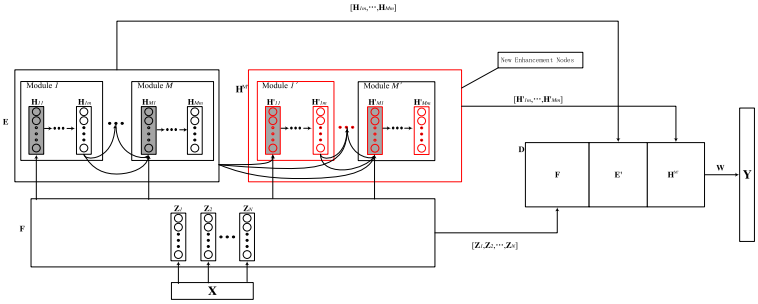


Fig. 8. Incremental learning for CEBLS-dense: increment of additional enhancement nodes.

way. Therefore, in this part, we introduce the incremental algorithm of additional enhancement nodes. The process is shown in Fig. 8.

We will detail the broad expansion method for adding p enhancement nodes. We assume $\mathbf{D}_N^{M+1} = [\mathbf{D}_N^M | \mathbf{H}_{(M+1)m}]$, where $\mathbf{H}_{(M+1)m}$ is the additional enhancement nodes, and $\mathbf{H}_{(M+1)m}$ can be computed by the following:

$$\mathbf{H}_{(M+1)k} = \begin{cases} \zeta\left(\mathbf{D}_N^M \mathbf{W}_{\mathbf{h}_{(M+1)k}} + \beta_{h_{(M+1)k}}\right) & k = 1 \\ \zeta\left(\mathbf{H}_{(M+1)(k-1)} \mathbf{W}_{\mathbf{h}_{(M+1)k}} + \beta_{h_{(M+1)k}}\right) & k = 2, \ldots, m. \end{cases}$$

Then, we could deduce the pseudo-inverse of the new matrix as

$$\left(\mathbf{D}_N^{M+1}\right)^+ = \begin{bmatrix} \left(\mathbf{D}_N^M\right)^+ - \mathbf{Q}\mathbf{B}^T \\ \mathbf{B}^T \end{bmatrix} \tag{33}$$

where $\mathbf{Q} = (\mathbf{D}_N^M)^+ \mathbf{H}_{(M+1)m}$

$$\mathbf{B}^T = \begin{cases} \mathbf{C}^+ & \text{if } \mathbf{C} \neq 0 \\ \left(1 + \mathbf{Q}^T\mathbf{Q}\right)^{-1} \mathbf{Q}^T \left(\mathbf{D}_N^M\right)^+ & \text{if } \mathbf{C} = 0 \end{cases} \tag{34}$$

and $\mathbf{C} = \mathbf{H}_{(M+1)m} - \mathbf{D}_N^M \mathbf{Q}$.

So, the new weights are

$$\mathbf{W}_N^{M+1} = \begin{bmatrix} \mathbf{W}_N^M - \mathbf{Q}\mathbf{B}^T\mathbf{Y} \\ \mathbf{B}^T\mathbf{Y} \end{bmatrix}. \tag{35}$$

The incremental algorithm of the increment enhancement node is shown in Algorithm 3.

D. Incremental Learning for CEBLS-Dropout: Increment of Additional Enhancement Nodes

The incremental algorithm of CEBLS-dropout is similar to CEBLS [1]. The difference is that when we calculate the enhancement node $\mathbf{H}_{(j)}$, we should first generate r, which is 0 or 1, with probability p to randomly select the enhancement nodes. The selected enhancement node is denoted as $\mathbf{H}'_{(j)}$

$$\mathbf{H}'_{(j)} = r\mathbf{H}_{(j)} \tag{36}$$

In the incremental learning part, we also need to select the incremental enhancement nodes $\mathbf{H}_{(M+1)}$. The selected enhancement node is denoted as $\mathbf{H}'_{(M+1)}$.

$$\mathbf{H}'_{(M+1)} = r\mathbf{H}_{(M+1)}. \tag{37}$$

The incremental algorithm is shown in Algorithm 4. Since the incremental structure of CEBLS-dropout is graphically identical to that of CEBLS [1], the schematic diagram of the incremental structure of CEBLS-dropout has been omitted.

V. Experiments

In this section, four BLS network variants and SFEBLS are evaluated. The experiments are conducted on the MacOS 10.14 operating system, and the CPU is Intel Core i7.

A. Dataset

The experiments were conducted on six datasets: 1) MNIST; 2) NORB; 3) street view house number (SVHN); 4) fashion-MNIST; 5) ORL; and 6) CIFAR-10.

The MNIST dataset, which is a famous handwritten digital dataset [25], includes 60 000 samples for training and 10 000 samples for test. Each image is 28×28 gray-scaled pixels.

The NORB dataset [26] is an image dataset that contains five different 3-D toy species. There are 48 600 images, and each image has $2 \times 32 \times 32$ pixels.

The SVHN dataset [27] is from the Google Street View number. There are 73 257 digits for training, 26 032 digits for testing, and 531 131 additional digits for the expansion experiment. The size of each image is $32 \times 32 \times 3$.

The Fashion-MNIST dataset [28], whose data size and format are exactly the same as MNIST, but the images are more complex than those of MNIST. The Fashion-MNIST dataset is mainly composed of images of clothes and shoes.

The ORL dataset [29] consists of 400 gray-level facial images of 40 different people, and each image has $2 \times 92 \times 112$ pixels.

TABLE I
EXPERIMENTAL RESULTS ON MNIST DATASET

Algorithms	No. of Nodes or Groups				No. of Prra.	Training time	Accuracy
	N_f	N_m	N_e	N_t			
CNN	-	-	-	-	-	-	95.63% *
MLP	-	-	-	-	-	-	97.39% *
FRBM	-	-	-	-	-	-	97.44% *
BLS	15	15	13000	1	132k	111.9s	98.58±0.18%
CFBLS-pyramid	15	15	13000	1	132k	105.2s	**98.65±0.14%**
CEBLS-dropout	15	15	1300	10	132k	88.21s	98.59±0.09%
CFBLS-dense	15	15	13000	1	132k	106.2s	98.59±0.20%
CEBLS-dense	15	15	1300	10	132k	**63.5s**	98.40±0.05%

* Results of CNN, MLP and Fuzzy Restrict Boltzmann Machine(FRBM) are obtained from Table I in [13].

TABLE II
EXPERIMENTAL RESULTS ON NORB DATASET

Algorithms	No. of Nodes or Groups				No. of Prra.	Training time	Accuracy
	N_f	N_m	N_e	N_t			
SAE	-	-	-	-	-	-	86.28% *
MLP	-	-	-	-	-	-	84.20% *
BLS	21	100	8000	1	50k	34.5s	87.20±0.15%
CFBLS-pyramid	21	100	8000	1	50k	33.4s	87.55±0.19%
CEBLS-dropout	21	100	800	10	50k	**19.8s**	88.35±0.39%
CFBLS-dense	21	100	8000	1	50k	80.4s	87.45±0.36%
CEBLS-dense	21	100	800	10	50k	23.5s	**88.40±0.29%**

* Results of MLP and Stacked Auto Encoders (SAE) are obtained from Table VII in [13].

TABLE III
EXPERIMENTAL RESULTS ON FASHION-MNIST DATASET

Algorithms	No. of Nodes or Groups				No. of Prra.	Training time	Accuracy
	N_f	N_m	N_e	N_t			
MLP	-	-	-	-	-	425s	83.89±0.03%
KNN	-	-	-	-	-	4927s	84.70±0.00%
BLS	15	15	13000	1	132k	125s	89.20±0.20%
CFBLS-pyramid	15	15	13000	1	132k	138s	**89.88±0.15%**
CEBLS-dropout	15	15	1300	10	132k	90s	89.71±0.17%
CFBLS-dense	15	15	13000	1	132k	136s	89.74±0.16%
CEBLS-dense	15	15	1300	10	132k	**65s**	89.45±0.18%

The CIFAR-10 dataset [30] consists of 60 000 images, and each image has $3 \times 32 \times 32$ pixels. The dataset consists of ten categories, each containing 6000 images.

B. Performance of the Network Variants

In this section, we demonstrate the advantages of the four proposed network structures, and compare the classification abilities of these structures with BLS. These experiments are performed on the MNIST, NORB, Fashion-MNIST, and ORL datasets, respectively.

We choose the same total number of parameters to eliminate the influence of the parameter selection in our experiment, and the parameters include the following: 1) N_f represents the number of column vectors in each feature node; 2) N_m represents the number of feature nodes; and 3) N_e represents the number of enhancement nodes, and N_t represents the group of enhancements. It should be noted that under the condition that the total number of parameters is consistent, we use the grid search method to determine the parameters of the variant model to achieve the best results. We ran each experiment ten times and calculated the average accuracy. We also calculated the standard deviation to make sure that the results are stable. Our analysis of the results is summarized as follows.

1) Experimental Results on MNIST Dataset: Our results are reported in Table I. For the MNIST dataset, the accuracy of BLS is high enough, and the four network variants achieve in par performance with BLS. Among the four network structures, CFBLS-pyramid has the highest test accuracy of 98.65%, which is due to the fact that the pyramid structures can capture multilayered features. The enhancement nodes in CEBLS-dense are divided into M modules, and the computation time of the enhancement node in each module is much less than that in BLS. Therefore, CEBLS-dense has the shortest training time, saving nearly 50% of the time.

2) Experimental Results on NORB Dataset: The results are reported in Table II. In this case, sufficient feature extraction can easily lead to overfitting of the enhancement nodes. As Table II shows, the structures which can avoid overfitting have a good result, like CEBLS-dropout and CEBLS-dense, the result of the latter is 1.20% higher than the baseline.

3) Experimental Results on Fashion-MNIST Dataset: The results are reported in Table III. The Fashion-MNIST dataset is more complex than the MNIST, which means the data has richer features. Therefore, CFBLS-pyramid that can extract hierarchical features and CFBLS-dense that can extract iterative features perform better, the former outperforms the baseline with 0.68%.

4) Experimental Results on ORL Dataset: The results are reported in Table IV. In this dataset, each category has ten images. We randomly selected seven as the training set and the rest as the test set.

TABLE IV
EXPERIMENTAL RESULTS ON ORL DATASET

Algorithms	No. of Nodes or Groups				No. of Prra.	Training time	Accuracy
	N_f	N_m	N_e	N_t			
CNN	-	-	-	-	-	-	91.67%*
SVM	-	-	-	-	-	-	82.5%*
KNN.	-	-	-	-	-	-	83.3%*
BLS	15	15	13000	1	132k	**7s**	95.00±0.00%
CFBLS-pyramid	15	15	13000	1	132k	7s	96.67±0.00%
CEBLS-dropout	15	15	1300	10	132k	9s	96.67±0.00%
CFBLS-dense	15	15	13000	1	132k	9s	**97.50±0.00%**
CEBLS-dense	15	15	1300	10	132k	9s	97.50±0.00%

* Results of CNN, SVM and KNN are obtained from Table 2 in [24].

TABLE V
INCREMENTAL EXPERIMENTAL RESULTS OF CFBLS-PYRAMID ON MNIST: INCREMENT OF N + 1 FEATURES NODES

Number of Feature Nodes	Number of Enhancement Nodes	Testing Accuracy	Additional Training Times	Accumulative Training Time
45	600	95.15%	2.95s	2.95s
45 → 60	600 → 1500	96.85%	4.71s	7.66s
50 → 75	1500 → 2400	97.56%	6.38s	14.04s
75 → 90	2400 → 3300	97.89%	10.72s	24.76s
90 → 105	3300 → 4200	98.12%	13.14s	37.90s

TABLE VI
INCREMENTAL EXPERIMENTAL RESULTS OF CFBLS-DENSE ON MNIST: INCREMENT OF N + 1 FEATURES NODES

Number of Feature Nodes	Number of Enhancement Nodes	Testing Accuracy	Additional Training Times	Accumulative Training Time
30	600	95.2%	2.87s	2.87s
30 → 45	600 → 1500	97.24%	4.34s	7.21s
45 → 60	1500 → 2400	97.75%	6.27s	13.48s
50 → 75	2400 → 3300	97.94%	8.38s	21.86s
75 → 90	3300 → 4200	98.17%	10.53s	32.39s

TABLE VII
INCREMENTAL EXPERIMENTAL RESULTS OF CEBLS-DENSE ON MNIST: INCREMENT OF N+1 ENHANCEMENT NODES

Number of Feature Nodes	Number of Enhancement Nodes	Testing Accuracy	Additional Training Times	Accumulative Training Time
45	900	96.47%	4.93s	4.93s
45	900 → 1300	96.84%	2.49s	7.42s
45	1300 → 1700	97.23%	2.98s	10.40s
45	1700 → 2100	97.33%	3.50s	13.90s
45	2100 → 2500	97.55%	3.99s	17.89s

The results are reported in Table IV. As Table IV shows, the dense model CFBLS-dense and CEBLS-dense have better results, which is 2.50% higher than the baseline.

The results also indicate that our models outperform the standard CNN.

C. Incremental Experiments

In order to illustrate the incremental ability of our proposed models, we did four additional experiments shown in Tables V–VIII. The dataset we used is MNIST [25].

We did two kind of incremental experiments: 1) the increment of feature nodes and 2) increment of enhancement nodes. Incremental experiments of feature nodes are implemented on CFBLS-pyramid and CFBLS-dense, Tables V and VI show the results. The incremental experiment of the enhancement nodes is implemented on CEBLS-dense and CEBLS-dropout, Tables VII and VIII lists the results.

In Table V, we tested the feature node incremental algorithm on CFBLS-pyramid. We selected 45 feature nodes and 600 enhancement nodes as initial data. Each incremental step adds 15 feature nodes and 900 enhancement nodes. From Table V, we can see that with the increase number of nodes, the accuracy has improved gradually. The increase in the training time is also in the acceptable range.

In Table VI, we tested the feature node incremental algorithm on CFBLS-dense. The initial number of feature nodes and enhancement nodes is 30 and 600, respectively. Each incremental step adds 15 feature nodes and 900 enhancement nodes. From Table VI, we can see the effectiveness of the incremental algorithm on CFBLS-dense.

In Table VII, we tested the enhancement node incremental algorithm on CEBLS-dense. We did an incremental experiment

TABLE VIII
INCREMENTAL EXPERIMENTAL RESULTS OF CEBLS-DROPOUT ON MNIST: INCREMENT OF N + 1 ENHANCEMENT NODES

Number of Feature Nodes	Number of Enhancement Nodes	Testing Accuracy	Additional Training Times	Accumulative Training Time
45	900	96.80%	5.22s	5.22s
45	900 $\rightarrow$ 1300	97.63%	2.45s	7.67s
45	1300 $\rightarrow$ 1700	98.12%	4.04s	11.71s
45	1700 $\rightarrow$ 2100	98.20%	7.80s	19.51s
45	2100 $\rightarrow$ 2500	98.27%	10.42s	29.93s

TABLE IX
EXPERIMENTAL RESULTS ON SIX DATASETS WITH FEATURE EXTRACTION COMPARISON FOR DIFFERENT NETWORK VARIANTS

Algorithms	NORB	Fashion-MNIST	ORL	SVHN	CIFAR-10
BLS	89.27%*	89.20±0.20%	95.00±0.00%	77.56±0.28%	49.30±0.35%
SFEBLS(HOG)	92.83±0.41%	91.8±0.45%	100.00±0.00%	90.05±0.29%	64.55±0.21%
SFEBLS(LBP)	93.75±0.36%	90.48±0.36%	100.00±0.00%	88.16±0.40%	67.15±0.18%
SFE(HOG)-CFBLS-pyramid	93.18±0.33%	92.11±0.39%	100.00±0.00%	90.10±0.39%	64.45±0.25%
SFE(LBP)-CFBLS-pyramid	94.67±0.36%	90.86±0.41%	100.00±0.00%	88.16±0.40%	67.13±0.16%
SFE(HOG)-CEBLS-dropout	92.84±0.39%	92.33±0.29%	100.00±0.00%	90.09±0.24%	65.30±0.30%
SFE(LBP)-CEBLS-dropout	**95.45±0.40%**	90.76±0.31%	100.00±0.00%	88.36±0.28%	67.46±0.32%
SFE(HOG)-CFBLS-dense	93.34±0.21%	92.34±0.20%	100.00±0.00%	90.16±0.29%	64.75±0.16%
SFE(LBP)-CFBLS-dense	94.76±0.26%	90.88±0.28%	100.00±0.00%	88.35±0.27%	67.25±0.28%
SFE(HOG)-CEBLS-dense	93.19±0.30%	**92.35±0.30%**	**100.00±0.00%**	**90.44±0.30%**	64.82±0.36%
SFE(LBP)-CEBLS-dense	95.19±0.30%	90.58±0.33%	100.00±0.00%	88.43±0.33%	**67.72±0.34%**

* In this experiment, we opt to use the best result of each model for comparison. Therefore, we use the best results of NORB from [13] instead of our results in Table II.

with enhancement nodes. At first, it has 45 feature nodes and 900 enhancement nodes, and each incremental step adds 400 enhancement nodes. Because of the disadvantage of the enhancement nodes increment, the results in Table VII are poorer than those in Tables V and VI. Nevertheless, the experimental results show that the increment of CEBLS-dense is still effective.

In Table VIII, we tested the enhancement node incremental algorithm on CEBLS-dropout. The number of original enhancement nodes and incremental enhancement nodes for each step are the same as in CEBLS-dense. The results show that the incremental algorithm on CFBLS-dropout is effective and efficient.

D. Feature Extraction

In this section, we compare the object recognition ability of BLS, SFEBLS, and the variants based on semantic feature extraction. The methods we select for the semantic feature extraction layer are HOG and LBPs. It should be noted that unlike the previous variant structures validation, we no longer maintain the same number of model parameters. We use the grid search method and achieve the best results, because our goal is not to verify the superiority of the structural variants, but to demonstrate the effect of feature extraction. We ran each experiment ten times and calculated the average. We also calculated the standard deviation to make sure that the results are stable.

Table IX shows that SFEBLS and its variants can increase the accuracy rate on the NORB dataset by 6.18% [SFE(LBP)-CEBLS-dropout], Fashion-MNIST dataset by 3.15% [SFE(HOG)-CEBLS-dense], ORL data by 5.00% (all of our SFE models), SVHN dataset by 12.88% [SFE(HOG)-CEBLS-dense], and CIFAR-10 dataset by 18.42% [SFE(LBP)-CEBLS-dense], respectively. Results indicate that SFEBLS has better feature representation by extracting the semantic features of images first and then combining them with sparse feature extraction. Moreover, CFE-CEBLS-dense has better results than other models, combined with the results of Tables I–IV, we see that CEBLS-dense has a fast computing speed. Therefore, we know that CEBLS-dense is the most effective and efficient model.

VI. CONCLUSION

In this article, we analyze and consolidate the theoretical basis of BLS. Moreover, we propose four novel variant networks. In order to apply BLS to object recognition, a BLS structure called SFEBLS with a semantic feature extraction layer has been proposed. Our results show that the four proposed networks and SFEBLS can outperform the baseline BLS. Our analysis shows that BLS and the generalized inverse theory are inseparable, which indicates that a large number of generalized inverse theories can be applied to BLS. Performance evaluation of SFEBLS indicates that semantic feature extraction is essential when BLS is applied to object recognition tasks. In future research, more BLS network variants can be explored based on the generalized inverse theory. A semantic feature layer can use CNN and other deep networks to obtain more distinctive semantic features.

REFERENCES

[1] C. 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.

[2] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," *Comput. Sci.*, vol. 4, pp. 357–361, Dec. 2014.

[3] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[4] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017. [Online]. Available: arXiv:1706.05587.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 770–778.

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 4700–4708.
[7] S. Haykin and B. Kosko, *Gradientbased Learning Applied to Document Recognition*, Wiley-IEEE Press. 2009, pp. 306–351.
[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[9] L. Zhang, G. Zhu, L. Mei, P. Shen, S. A. A. Shah, and M. Bennamoun, "Attention in convolutional LSTM for gesture recognition," in *Proc. 32nd Int. Conf. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1953–1962.
[10] L. Zhang *et al.*, "Semantic scene completion with dense CRF from a single depth image," *Neurocomputing*, vol. 318, pp. 182–195, Nov. 2018.
[11] L. Zhang, S. Zhang, P. Shen, G. Zhu, S. A. A. Shah, and M. Bennamoun, "Relationship detection based on object semantic inference and attention mechanisms," in *Proc. Int. Conf. Multimedia Retrieval*, 2019, pp. 68–72.
[12] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Budapest, Hungary, 2005, pp. 985–990.
[13] C. 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.
[14] M. Han, S. Feng, C. P. Chen, M. Xu, and T. Qiu, "Structured manifold broad learning system: A manifold perspective for large-scale chaotic time series analysis and prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 9, pp. 1809–1821, Sep. 2019.
[15] L. Yang, S. Song, and C. P. Chen, "Transductive transfer learning based on broad learning system," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Miyazaki, Japan, 2018, pp. 912–917.
[16] T. Zhang, Z. Liu, X.-H. Wang, X.-F. Xing, C. P. Chen, and E. Chen, "Facial expression recognition via broad learning system," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Miyazaki, Japan, 2018, pp. 1898–1902.
[17] T. Song, W. Zheng, P. Song, and Z. Cui, "EEG emotion recognition using dynamical graph convolutional neural networks," *IEEE Trans. Affective Comput.*, early access, Mar. 21, 2018, doi: 10.1109/TAFFC.2018.2817622.
[18] S. Feng and C. 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.
[19] D. Serre, *Matrices*, vol. 216. New York, NY, USA: Springer, 2010.
[20] A. Ben-Israel and T. N. Greville, *Generalized Inverses: Theory and Applications*, vol. 15. New York, NY, USA: Springer, 2003.
[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit. (CVPR'05)*, vol. 1. San Diego, CA, USA, 2005, pp. 886–893.
[23] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.
[24] T. L. Zhang, R. Chen, X. Yang, and S. Guo, "Rich feature combination for cost-based broad learning system," *IEEE Access*, vol. 7, pp. 160–172, 2018.
[25] 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.
[26] 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. Compu. Vis. Pattern Recognit. (CVPR)*, vol. 2. Washington, DC, USA, 2004, pp. 97–104.
[27] 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, p. 9.
[28] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017. [Online]. Available: arXiv:1708.07747.
[29] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. IEEE Workshop Appl. Comput.Vis.*, vol. 22. Sarasota, FL, USA, 1994, pp. 138–142.
[30] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Rep. 1, vol. 1, Jan. 2009.
[31] L. Wen, L. Gao, and X. Li, "A new deep transfer learning based on sparse auto-encoder for fault diagnosis," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 1, pp. 136–144, Jan. 2019.
[32] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.
[33] J. Deutsch and D. He, "Using deep learning-based approach to predict remaining useful life of rotating components," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 1, pp. 11–20, Jan. 2018.
[34] K. Muhammad, J. Ahmad, Z. Lv, P. Bellavista, P. Yang, and S. W. Baik, "Efficient deep CNN-based fire detection and localization in video surveillance applications," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 7, pp. 1419–1434, Jul. 2019.
[35] B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani, "Deep convolutional neural networks and learning ECG features for screening paroxysmal atrial fibrillation patients," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 12, pp. 2095–2104, Dec. 2018.
[36] H. Laga, Y. Guo, H. Tabia, R. B. Fisher, and M. Bennamoun, *3D Shape Analysis: Fundamentals, Theory, and Applications*. Newark, NJ, USA: Wiley, 2018.
[37] T. Zhang, X. Wang, X. Xu, and C. 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.
[38] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A guide to convolutional neural networks for computer vision," *Synth. Lectures Comput. Vis.*, vol. 8, no. 1, pp. 1–207, 2018.