Novel Efficient RNN and LSTM-Like Architectures: Recurrent and Gated Broad Learning Systems and Their Applications for Text Classification

Jie Du, Chi-Man Vong, *Senior Member, IEEE*, and C. L. Philip Chen, *Fellow, IEEE*

***Abstract*—High accuracy of text classification can be achieved through simultaneous learning of multiple information, such as *sequence information* and *word importance*. In this article, a kind of flat neural networks called the broad learning system (BLS) is employed to derive two novel learning methods for text classification, including recurrent BLS (R-BLS) and long short-term memory (LSTM)-like architecture: gated BLS (G-BLS). The proposed two methods possess three advantages: 1) higher accuracy due to the simultaneous learning of multiple information, even compared to deep LSTM that extracts deeper but single information only; 2) significantly faster training time due to the noniterative learning in BLS, compared to LSTM; and 3) easy integration with other discriminant information for further improvement. The proposed methods have been evaluated over 13 real-world datasets from various types of text classification. From the experimental results, the proposed methods achieve higher accuracies than LSTM while taking significantly less training time on most evaluated datasets, especially when the LSTM is in deep architecture. Compared to R-BLS, G-BLS has an extra forget gate to control the flow of information (similar to LSTM) to further improve the accuracy on text classification so that G-BLS is more effective while R-BLS is more efficient.**

***Index Terms*—Broad learning system (BLS), sequence information, simultaneous learning, text classification, word importance.**

Manuscript received August 6, 2019; revised November 1, 2019; accepted January 14, 2020. Date of publication February 20, 2020; date of current version February 17, 2021. This work was supported in part by the University of Macau Research under Grant MYRG2018-00138-FST and Grant MYRG2016-00134-FST, in part by the Science and Technology Development Fund, Macau, China, under Grant 273/2017/A, Grant 079/2017/A2, Grant 024/2015/AMJ, and Grant 0119/2018/A3, in part by the Natural Science Foundation of Shenzhen University under Grant 860-000002110121 and Grant 860-000002110122, and in part by the National Natural Science Foundation of China under Grant 61871274, Grant 61801305, Grant 81571758, Grant 61751202, Grant 61751205, Grant 61572540, Grant U1813203, and Grant U1801262. This article was recommended by Associate Editor S. Ozawa. *(Corresponding author: Chi-Man Vong.)*

Jie Du is with the National-Regional Key Technology Engineering Laboratory for Medical Ultrasound, Guangdong Key Laboratory for Biomedical Measurements and Ultrasound Imaging, School of Biomedical Engineering, Health Science Center, Shenzhen University, Shenzhen 518060, China (e-mail: dujie@szu.edu.cn).

Chi-Man Vong is with the Department of Computer and Information Science, University of Macau, Macau, China (e-mail: cmvong@umac.mo).

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).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2020.2969705

I. Introduction

TEXT classification is a typical and important task resorting to natural language processing (NLP) techniques [1]–[3]. Currently, the most effective techniques in NLP usually employ a deep neural network for its high accuracy and ability to handle the temporal properties in natural languages [4]–[6]. Recurrent neural network (RNN) [7], [8] is a kind of deep neural network that can deal with natural languages for its *memory set* remembering the inputs that have been received so far [9]. In other words, RNN makes use of *sequence information* to effectively handle natural languages. Specifically, "sequence information" $\mathbf{Z}_p$ is the information learned based on both current input word $\mathbf{X}_p$ and the previously learned information $\mathbf{Z}_{p-1}$.

In RNN, in order to optimize the network parameters, such as weights and bias, the gradient-based technique is used for training. Different from feedforward neural networks [10], [11], the gradient in RNN training depends on the calculations of input over both current and previous time steps. Such gradient can be computed by an iterative procedure called backpropagation through time (BPTT) [12]. However, RNN trained with BPTT has difficulties in learning long-term dependence (i.e., dependence between the steps that are far apart) due to the well-known *vanishing* and *exploding* gradient problems [13], [14]. Hence, long short-term memory (LSTM) [13] is then specifically designed to get around these two gradient problems.

LSTM is based on RNN and learns *sequence information* as well. In every LSTM-layer (or LSTM memory block), three gates are designed and are separately controlling the following components [15]: 1) input (input gate); 2) memory cell (forget gate); and 3) output (output gate). Then multiple LSTM-layers are stacked on top of each other to construct a deep LSTM [16].

In fact, except *sequence information*, *word importance* also plays a significant role in many NLP tasks, such as text classification. For instance, to identify if an article is relevant to some interesting topics (e.g., economy) or not [17], some relevant words (e.g., company, market, economic, etc.) are very helpful and they are deemed more important than other words for discrimination. Therefore, for more effective text classification, *word importance* should be simultaneously considered along with *sequence information*.

In order to learn both sequence information and word importance in one network, there are three possible strategies.

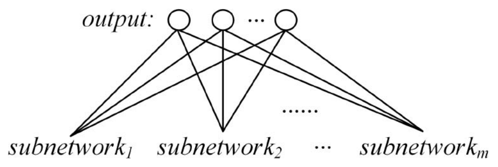

Fig. 1. Simultaneous learning of multiple information by a flat neural network.

1) *Layer-by-layer learning of multiple information* using a deep architecture, in which each layer learns a different kind of information. However, deep architecture is a black box [18] in which each layer cannot exactly learn the user-specified knowledge (i.e., *sequence information* or *word importance*). Currently, the user can only specify the number of layers and the amount of hidden nodes in each layer.
2) *Independent learning of multiple information* using an ensemble of models [19]–[21], in which each model learns a different kind of information for classification and then their results are combined. However, the way of combining results is usually heuristic and difficult to optimally determine [22]. Furthermore, in this way, the *sequence information* and *word importance* are learned independently to determine the classification, and this independent learning way seems unpromising.
3) *Simultaneous learning of multiple information* using a flat neural network as illustrated in Fig. 1, in which each tiled subnetwork learns a different kind of information, and then the outputs of all subnetworks are connected to the output layer simultaneously. This way seems most promising because *sequence information* and *word importance* are learned simultaneously in the flat neural network.

However, the traditional RNN or LSTM mechanism does not provide the way of simultaneous learning so that it becomes highly nontrivial to implement strategy 3) over RNN or LSTM mechanism. To our best effort, we cannot find such a simultaneous learning mechanism of multiple information for RNN or LSTM.

Recently, a kind of noniterative learning called the broad learning system (BLS) [23] was proposed. Based on the random projection of initial weights and bias, BLS can run up to hundred times faster than traditional gradient-based iterative learning while achieving even higher accuracy. In fact, BLS is a kind of flat neural network in which the original inputs are transferred and placed as mapped features in *feature nodes* (one subnetwork) and the structure is expanded in the wide sense in the *enhancement nodes* (another subnetwork). Then both feature and enhancement nodes are connected to output nodes. Therefore, BLS provides the necessary mechanism of simultaneous learning in Fig. 1. To our best knowledge, the attractive advantage that a flat neural network can simultaneously learn multiple information is first discovered in this article and practiced in real-world applications. As a remark, another popular noniterative learning method is an extreme learning machine (ELM) [24], [25], but it does not have an enhancement layer.

Unfortunately, both the subnetworks in BLS only learn the same kind of feature information but not multiple kinds of

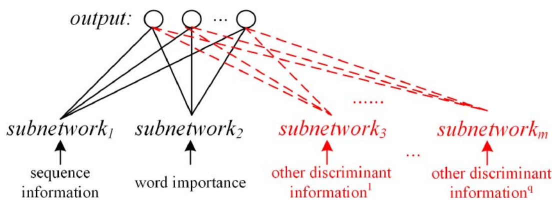

Fig. 2. Integrating other discriminant information (to be excavated) into BLSs.

information. Moreover, BLS assumes that all inputs are independent of each other, and simply sets the whole data matrix $\mathbf{X}$ as input [23]. Since the words in input sentences in most NLP tasks, such as text classification, are highly correlated [26], it is impossible to assume the independence of all inputs so that the application of BLS to text classification becomes nontrivial. This point is further discussed in Section III-A.

In this article, two novel structures for BLS are proposed to simultaneously learn *sequence information* and *word importance* for text classification, namely, recurrent BLS (R-BLS) and gated BLS (G-BLS), whose significance are briefly shown as follows.

1) R-BLS compensates the weaknesses of BLS, RNN, and LSTM on text classification by simultaneously learning the *sequence information* and *word importance* of text. Inherited from BLS, R-BLS has a closed-form solution so that BPTT is no longer required for the optimization of network parameters, such as weights and bias. As a result, R-BLS does not suffer from the issues of *vanishing* and *exploding* gradients.
2) In R-BLS, all previous *sequence information* from step 0th to $(p-1)$th are captured into current *memory set* $\mathbf{Z}_p$ (detailed in Section III-A1). However, in practice, not all previous information is necessarily captured into $\mathbf{Z}_p$, because the current processed word may not be correlated with the previous words that are far ahead. In order to only remember the words that are highly correlated and forget the words that are uncorrelated, G-BLS is proposed. G-BLS is based on R-BLS but has *forget gates* (similar to LSTM) to forget the uncorrelated information and only delivers the correlated ones to $\mathbf{Z}_p$ for further computation.
3) Both R-BLS and G-BLS are very effective for text classification, especially when compared to deep LSTM. Even though a deep and complicated structure is constructed to extract the deeper representation of *sequence information*, it still cannot be compared with simple BLSs (i.e., R-BLS and G-BLS) with additional information of *word importance* (detailed in Section IV-E1).
4) Deep neural networks suffer from a time-consuming training process due to iterative layerwise tuning of billions of network parameters [23]. Meanwhile, the proposed R-BLS and G-BLS are all noniterative learning methods and hence they are much more efficient than deep neural networks (detailed in Section IV-D1).
5) Other discriminant information (to be excavated) can be easily integrated into the proposed BLSs for further improvement, as illustrated in Fig. 2.

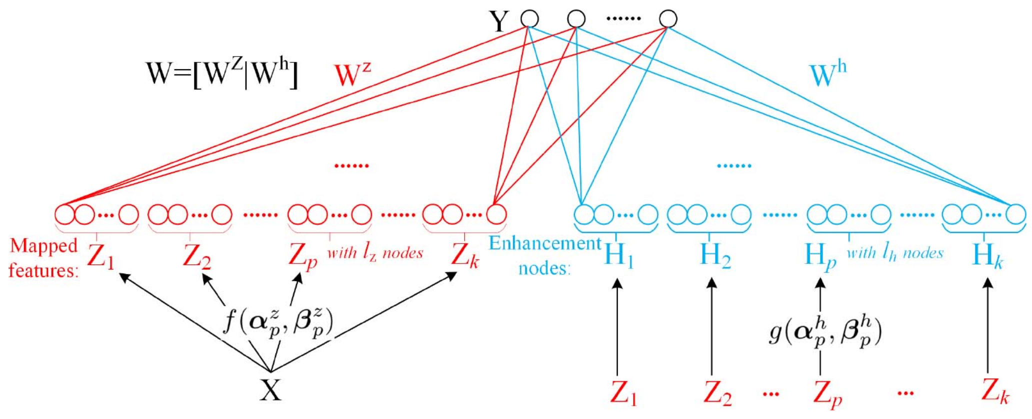

Fig. 3. Structure of the original BLS.

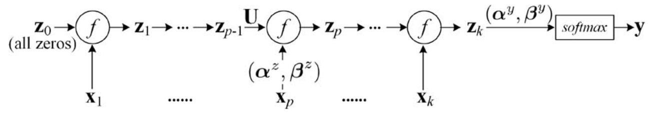

Fig. 4. Structure of RNN for text classification.

The organization of this article is as follows. Section II provides a short review on BLS, RNN, and LSTM. Section III details our proposed methods: R-BLS and G-BLS. Section IV shows the experimental results with analysis and discussion. Finally, a conclusion is drawn in Section V.

II. Preliminaries

A. Broad Learning System

Given N training data $\{\mathbf{x}^i, \mathbf{y}^i\}$, $i = 1$ to N, where $\mathbf{x}^i \in \mathbb{R}^D$ represents the ith sample with the corresponding output $\mathbf{y}^i$. In matrix form, $\mathbf{X} = [\mathbf{x}^i] \in \mathbb{R}^{N\times D}$ is the input matrix and $\mathbf{Y} = [\mathbf{y}^i] \in \mathbb{R}^{N\times m}$ is the output matrix, where D is the dimension of input vector $\mathbf{x}^i$ and m is the number of class labels. For k features $\mathbf{Z}_p$ mapped from $\mathbf{X}$, $p = 1$ to k, each $\mathbf{Z}_p$ is with l_z hidden nodes and can be represented as

$$\mathbf{Z}_p = f\left(\mathbf{X}\boldsymbol{\alpha}_p^z + \boldsymbol{\beta}_p^z\right) \tag{1}$$

where k and l_z are user-specified parameters, and f is an activation function such as *sigmoid*. $\boldsymbol{\alpha}_p^z \in \mathbb{R}^{D\times l_z}$ and $\boldsymbol{\beta}_p^z \in \mathbb{R}^{N\times l_z}$ are, respectively, the random weights and bias for input $\mathbf{X}$. Under this way, every $\mathbf{Z}_p$ is obtained with dimension of $N \times l_z$.

Similarly, the *enhancement nodes* $\mathbf{H}_p$ with l_h hidden nodes, $p = 1$ to k, are obtained by

$$\mathbf{H}_p = g\left(\mathbf{Z}_p\boldsymbol{\alpha}_p^h + \boldsymbol{\beta}_p^h\right) \tag{2}$$

where g is also an activation function, which can be the same with f. $\boldsymbol{\alpha}_p^h \in \mathbb{R}^{l_z\times l_h}$ and $\boldsymbol{\beta}_p^h \in \mathbb{R}^{N\times l_h}$ are, respectively, randomly generated weights and bias for the mapped features $\mathbf{Z}_p$. Hence, $\mathbf{H}_p$ is obtained with the dimension of $N \times l_h$.

Then, the output nodes $\mathbf{Y}$ can be represented as a wide or broad structure by

$$\mathbf{Y} = [\mathbf{Z}_1, \ldots, \mathbf{Z}_k|\mathbf{H}_1, \ldots, \mathbf{H}_k]\mathbf{W}. \tag{3}$$

According to [23], the connecting weights for this broad structure are $\mathbf{W} = \mathbf{A}^\dagger\mathbf{Y}$, which can be easily computed through ridge regression approximation as follows:

$$\mathbf{A}^\dagger = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T \tag{4}$$

where $\mathbf{A} = [\mathbf{Z}_1, \ldots, \mathbf{Z}_k|\mathbf{H}_1, \ldots, \mathbf{H}_k]$. Fig. 3 details the broad network structure. Moreover, three incremental learning algorithms of BLS are also given in [23] for feature nodes, enhancement nodes, and new incoming inputs, respectively. Since incremental learning is not the objective in this article, these three incremental algorithms are not introduced here. For details, kindly refer to [23].

B. Recurrent Neural Network

RNN is a popular model that has shown great promise in many NLP tasks [9], [27], [28]. Fig. 4 shows the structure of RNN for text classification. The detailed training procedure is shown below.

Different from image and numeric datasets, a vocabulary $\mathbb{V}$ of words is required to construct from the text dataset, which includes only the words with occurrence frequency larger than a user-specified count. Then, each sample or sentence $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_p, \ldots, \mathbf{s}_k)$ in the text dataset for classification includes k words from this vocabulary only, that is, $\mathbf{s}_p \in \mathbb{V}$ for $p = 1$ to k. For the processing in neural networks, all of the words $\mathbf{s}_p$ are first mapped to numeric vector representation using a trained word embedding. After the mapping, every $\mathbf{s}_p \in \mathbb{V}$ is mapped to $\mathbf{x}_p \in \mathbb{R}^d$ where d is user specified. For every $\mathbf{x}_p$ ($p = 1$ to k), its *memory set* $\mathbf{z}_p$ is calculated based on not only the current input $\mathbf{x}_p$ but also the previous memory $\mathbf{z}_{p-1}$ ($\mathbf{z}_0$ is typically set as all zeros)

$$\mathbf{z}_p = f\left(\boldsymbol{\alpha}^z\mathbf{x}_p + \mathbf{U}\mathbf{z}_{p-1} + \boldsymbol{\beta}^z\right) \tag{5}$$

where f is an activation function; $\boldsymbol{\alpha}^z$ and $\mathbf{U}$ are the weights associated with $\mathbf{x}_p$ and $\mathbf{z}_{p-1}$, respectively; and $\boldsymbol{\beta}^z$ is the bias. Note that $\mathbf{z}_p$ has the same dimension as $\mathbf{x}_p$.

For text classification, only the last $\mathbf{z}_k$ is conveyed to the *softmax* layer for output [29], that is

$$\mathbf{y} = \text{softmax}\left(\boldsymbol{\alpha}^y\mathbf{z}_k + \boldsymbol{\beta}^y\right) \tag{6}$$

where $\boldsymbol{\alpha}^y$ and $\boldsymbol{\beta}^y$ are the weights and bias for final $\mathbf{z}_k$, and $\mathbf{y}$ is the corresponding output label.

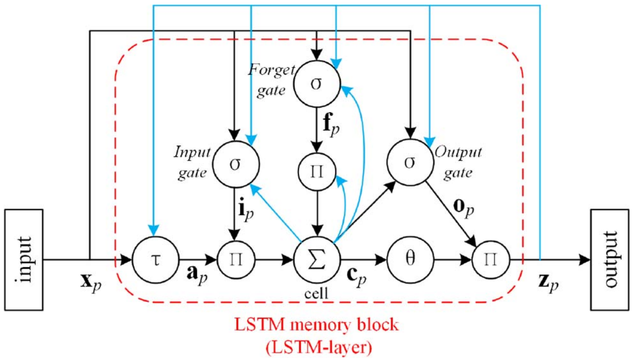

Fig. 5. Structure of an LSTM network with one memory block, that is, one LSTM-layer.

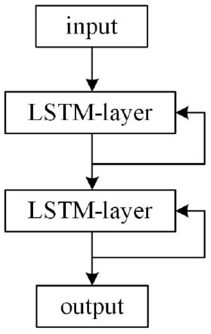

Fig. 6. Structure of a deep LSTM with two LSTM-layers.

Then, (5) and (6) are iteratively applied over different sentences in a dataset until all sentences are learned. The parameters $\boldsymbol{\alpha}^z$, $\mathbf{U}$, $\boldsymbol{\beta}^z$, $\boldsymbol{\alpha}^y$, and $\boldsymbol{\beta}^y$ are all optimized by BPTT. However, RNN cannot process long temporal sequences of data [13], [14].

C. Long Short-Term Memory

LSTM is designed based on RNN to process long temporal sequence of data [13]. LSTM has *input gate*, *output gate*, and *forget gate* to control the flow of information, as illustrated in Fig. 5. In detail, input gate determines which information can be sent into the memory cell; forget gate determines which information should be removed from the memory cell; and output gate determines which information can be output.

Given an input sequence $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ with the corresponding output label $\mathbf{y}$, the output $\mathbf{z}_p$ of an LSTM-layer for time $p = 1$ to k is as follows:

$$\mathbf{i}_p = \sigma\left(\mathbf{W}_{xi}\mathbf{x}_p + \mathbf{W}_{hi}\mathbf{z}_{p-1} + \mathbf{W}_{ci}\mathbf{c}_{p-1} + \mathbf{b}_i\right) \quad (7)$$
$$\mathbf{f}_p = \sigma\left(\mathbf{W}_{xf}\mathbf{x}_p + \mathbf{W}_{hf}\mathbf{z}_{p-1} + \mathbf{W}_{cf}\mathbf{c}_{p-1} + \mathbf{b}_f\right) \quad (8)$$
$$\mathbf{a}_p = \tau\left(\mathbf{W}_{xc}\mathbf{x}_p + \mathbf{W}_{hc}\mathbf{z}_{p-1} + \mathbf{b}_c\right) \quad (9)$$
$$\mathbf{c}_p = \mathbf{f}_p\mathbf{c}_{p-1} + \mathbf{i}_p\mathbf{a}_p \quad (10)$$
$$\mathbf{o}_p = \sigma\left(\mathbf{W}_{xo}\mathbf{x}_p + \mathbf{W}_{ho}\mathbf{z}_{p-1} + \mathbf{W}_{co}\mathbf{c}_p + \mathbf{b}_o\right) \quad (11)$$
$$\mathbf{z}_p = \mathbf{o}_p\theta\left(\mathbf{c}_p\right) \quad (12)$$

where σ is a sigmoid activation function; $\mathbf{i}_p$, $\mathbf{f}_p$, $\mathbf{o}_p$, $\mathbf{a}_p$, and $\mathbf{c}_p$ are, respectively, the outputs of input gate, forget gate, output gate, cell input activation, and cell state vectors at time p, and all of which are the same size with $\mathbf{z}_p$; $\mathbf{W}_{ci}$, $\mathbf{W}_{cf}$, and $\mathbf{W}_{co}$ are diagonal weight matrices for peephole connections; and τ and θ are activation functions, generally tanh [16].

For text classification

$$\mathbf{y} = \mathrm{softmax}\left(\boldsymbol{\alpha}^y\mathbf{z}_k + \boldsymbol{\beta}^y\right) \quad (13)$$

where $\boldsymbol{\alpha}^y$ and $\boldsymbol{\beta}^y$ are, respectively, the weights and bias for the softmax function. Similar to RNN, (7)–(13) are iteratively applied to learn all sentences in a dataset. The parameters in LSTM are also optimized by BPTT.

Moreover, LSTM can be in deep structure by stacking multiple LSTM-layers on top of each other [16], as illustrated in Fig. 6. In detail, the output $\mathbf{z}_p$ from the lower LSTM-layer is considered as the input $\mathbf{x}_p$ of the upper LSTM-layer. This deep LSTM network can flexibly learn long-range context, and significant performance improvement was shown on some NLP datasets [9], [16], compared to the shallow one shown in Fig. 5.

III. Proposed Method

In this section, the proposed R-BLS and G-BLS are introduced in detail.

A. Recurrent Broad Learning System

In text classification, better prediction can always result from knowing both the word sequence and the importance of words. Therefore, the intuitive idea for R-BLS is to make use of both *sequence information* and *word importance*.

For instance, in the task of identifying the sentiments of film reviews, there are two such reviews: 1) it is *not good* at all and 2) it is *good* and *not* bad. Although "not" and "good" occur in both reviews, they have totally different meanings. Obviously, review *a* is *negative* while *b* indicates *positive*. The key issue is that these two words are in different *orders* or *sequence* shown in the two reviews. Hence, sequence information is very important for text classification. Moreover, for review *a*, the words "not" and "good" are obviously more important than "it", "is", "at", and "all" for identification. In review *b*, the words good, not, and "bad" are also more important than other words. As a result, both *sequence information* and *word importance* must be simultaneously considered for more accurate text classification.

In the following two sections, the limitations of the original BLS framework for learning *sequence information* and *word importance* are, respectively, explained followed by the detailed procedures of R-BLS to learn these two types of information.

1) Learning Sequence Information: Given N training data $\{\mathbf{x}^i, \mathbf{y}^i\}$, $i = 1$ to N, where $\mathbf{x}^i = (\mathbf{x}_1^i, \mathbf{x}_2^i, \ldots, \mathbf{x}_p^i, \ldots, \mathbf{x}_k^i)$ and $\mathbf{y}^i = (y_1^i, y_2^i, \ldots, y_j^i, \ldots, y_m^i)$ (m is the number of class labels) is the associated label. Assume every $\mathbf{x}_p^i \in \mathbb{R}^d$ ($p = 1$ to k) is a mapped vector representation under *word2vec*. Every $y_j^i \in \{1, -1\}$, and $y_j^i = 1$ for $j = u \in [1, \ldots, m]$ (u is the associated label index), and $y_j^i = -1$ otherwise. In matrix form $\mathbf{X} = [\mathbf{x}^i] \in \mathbb{R}^{N\times(k\times d)}$, while the matrix representation of the pth word in all N samples is $\mathbf{X}_p = [\mathbf{x}_p^1, \mathbf{x}_p^2, \ldots, \mathbf{x}_p^i, \ldots, \mathbf{x}_p^N]^T \in \mathbb{R}^{N\times d}$. $\mathbf{Y} = [\mathbf{y}^i] \in \mathbb{R}^{N\times m}$ is the label matrix for all N samples.

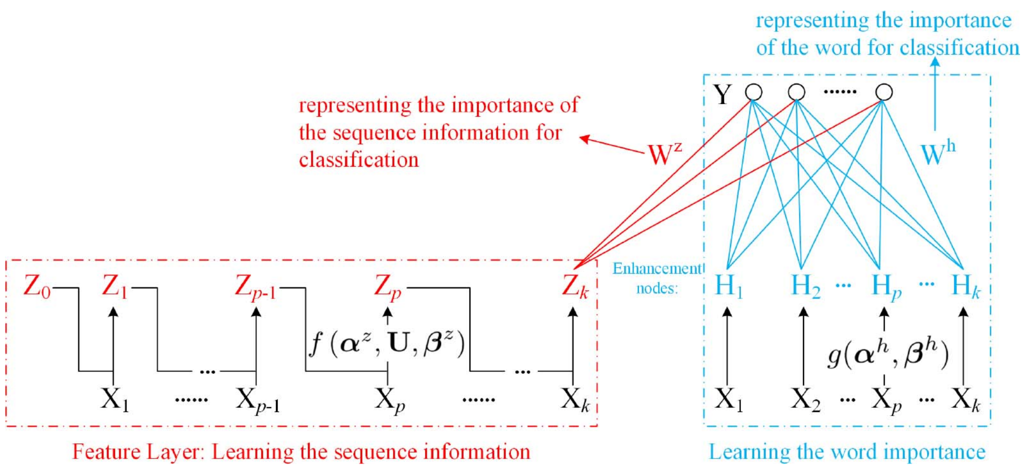
representing the importance of the word for classification
representing the importance of the sequence information for classification
W^z
W^h
Y
Enhancement nodes:
H_1 H_2 ··· H_p ··· H_k
g(α^h, β^h)
X_1 X_2 ··· X_p ··· X_k
Z_0 Z_1 Z_{p-1} Z_p Z_k
f(α^z, U, β^z)
X_1 ······ X_{p-1} X_p ······ X_k
Feature Layer: Learning the sequence information
Learning the word importance

Fig. 7. Structure of the proposed R-BLS.

In the original BLS framework shown in Fig. 3, every attribute $x_j \in \mathbf{x} \in \mathbb{R}^D$ ($j = 1$ to D) is required to be independent of all other attributes. Under this independence, each mapped feature matrix $\mathbf{Z}_p$ can be learned using a different set of weights $\boldsymbol{\alpha}_p^z$ and bias $\boldsymbol{\beta}_p^z$ over the whole input matrix $\mathbf{X}$ [See (1)]. However, for NLP tasks, such as text classification, it is impossible to impose such independence over the input matrix $\mathbf{X}$ because every attribute or word in a sentence is *not independent* of each other [26]. If simply following the framework of constructing mapped features in original BLS for text classification, the *sequence information* of text data cannot be learned.

On the contrary, our proposed R-BLS learns the *sequence information* of text data by computing every $\mathbf{Z}_p$ from not only the current input $\mathbf{X}_p$ but also the previous *memory set* $\mathbf{Z}_{p-1}$ as follows:

$$\mathbf{Z}_p = f\left(\mathbf{X}_p\boldsymbol{\alpha}^z + \mathbf{Z}_{p-1}\mathbf{U} + \boldsymbol{\beta}^z\right) \tag{14}$$

where f is an activation function, such as *sigmoid*, and $\mathbf{Z}_0$ is typically set as all zeros. Moreover, as in RNN, only one set of weights and bias is necessary in the proposed R-BLS rather than multiple sets of weights and biases in original BLS.

Inherited from the original BLS, the weights $\boldsymbol{\alpha}^z$ and $\mathbf{U}$, and the bias $\boldsymbol{\beta}^z$ are randomly generated [23]. Note that $\mathbf{Z}_p$ has the same dimension of $\mathbf{X}_p$, that is, $N \times d$.

In fact, $\mathbf{Z}_p$ is the *memory set* of the network, which captures the information about what happened in all previous time steps. Therefore, the final $\mathbf{Z}_k$ is the input features that represents the *sequence information* learned from all words, that is, $\mathbf{X}_1$–$\mathbf{X}_k$. Moreover, in Fig. 7, the weight $\mathbf{W}^z$ connecting $\mathbf{Z}_k$ to the output nodes $\mathbf{Y}$ represents the importance of *sequence information* for final classification. After k iterations, the final $\mathbf{Z}_k$ is obtained by

$$\mathbf{Z}_k = f\left(\mathbf{X}_k\boldsymbol{\alpha}^z + \mathbf{Z}_{k-1}\mathbf{U} + \boldsymbol{\beta}^z\right). \tag{15}$$

2) Learning Word Importance: The enhancement nodes in the original BLS framework shown in Fig. 3 are computed based on the mapped features $\mathbf{Z}_1$–$\mathbf{Z}_k$. However, each $\mathbf{Z}_p$ in R-BLS is a *memory set* capturing *sequence information* only, rather than *word importance*. Following the framework of original BLS, the enhancement nodes cannot capture the information of *word importance*. For this reason, R-BLS takes the original words $\mathbf{X}_p$ as input rather than the mapped features or *memory set* $\mathbf{Z}_p$, for $p = 1$ to k. Under this way, R-BLS can capture the importance of each word by the weight $\mathbf{W}^h$, which connects the enhancement nodes $\mathbf{H}_p$ ($p = 1$ to k) to the output nodes $\mathbf{Y}$. Each enhancement node $\mathbf{H}_p$ with l hidden nodes is calculated by

$$\mathbf{H}_p = g\left(\mathbf{X}_p\boldsymbol{\alpha}^h + \boldsymbol{\beta}^h\right) \tag{16}$$

where g is an activation function as well, and $\boldsymbol{\alpha}^h$ and $\boldsymbol{\beta}^h$ are also randomly generated as in original BLS. Hence, $\mathbf{H}_p$ is obtained with the dimension of $N \times l$.

3) Connecting Weights for R-BLS: In Fig. 7, $\mathbf{W}^z$ and $\mathbf{W}^h$, respectively, represent the importance of *sequence information* and *word importance* for final classification. The connecting weight $\mathbf{W}$ is then the concatenation of $\mathbf{W}^z$ and $\mathbf{W}^h$, that is, $\mathbf{W} = [\mathbf{W}_z|\mathbf{W}_h]$. As in original BLS, $\mathbf{W}^z$ and $\mathbf{W}^h$ are not separately computed but $\mathbf{W} = [\mathbf{W}_z|\mathbf{W}_h]$ is directly calculated through ridge regression approximation [30] of the pseudoinverse of $[\mathbf{Z}_k|\mathbf{H}_1, \ldots, \mathbf{H}_k]$. Therefore, the *sequence information* and the *word importance* are simultaneously learned to determine the final classification result. In detail, the connecting weight $\mathbf{W}$ is obtained by

$$\begin{aligned}\mathbf{W} &= \mathbf{A}^{\dagger\mathbf{Y}}\\ &= \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{Y}\end{aligned} \tag{17}$$

where

$$\mathbf{A} = [\mathbf{Z}_k|\mathbf{H}_1, \ldots, \mathbf{H}_k]. \tag{18}$$

By substituting (18) into (17)

$$\begin{aligned}\mathbf{W} &= [\mathbf{Z}_k|\mathbf{H}_1, \ldots, \mathbf{H}_k]^\dagger\mathbf{Y}\\ &= \left([\mathbf{Z}_k|\mathbf{H}_1, \ldots, \mathbf{H}_k]^T[\mathbf{Z}_k|\mathbf{H}_1, \ldots, \mathbf{H}_k]\right)^{-1}\\ &\quad\times [\mathbf{Z}_k|\mathbf{H}_1, \ldots, \mathbf{H}_k]^T\mathbf{Y}.\end{aligned} \tag{19}$$

Hence, the proposed R-BLS has closed-form solution (19) for connecting weight $\mathbf{W}$ so that the gradient descent-based optimization approach BPTT is no longer needed. In other

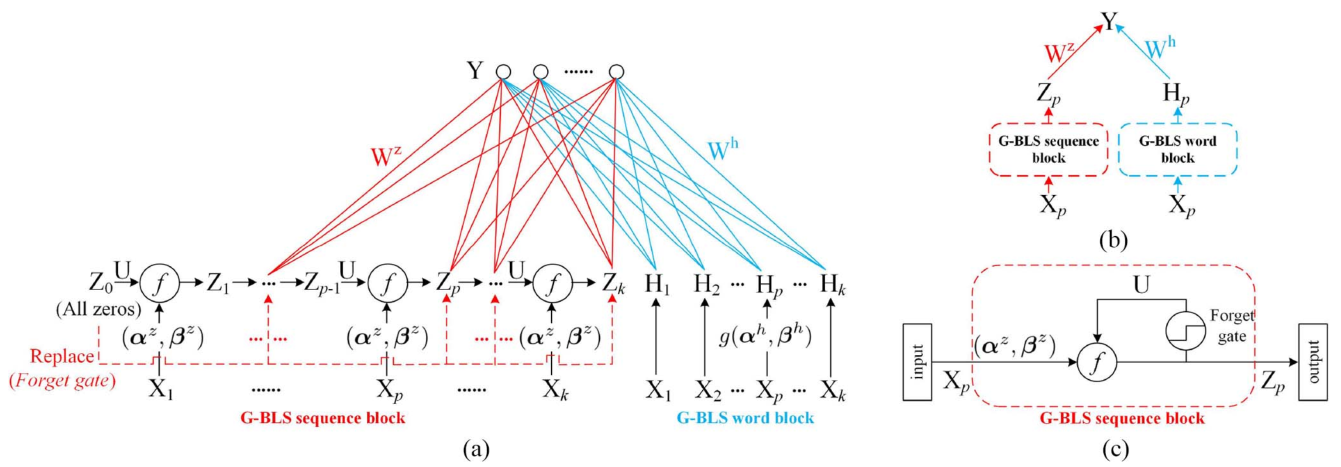

Fig. 8. Structure of G-BLS. (a) Detailed structure. (b) Thumbnail of G-BLS. (c) Structure of the *G-BLS sequence block* referring to the *LSTM memory block*.

Algorithm 1 R-BLS

```
**Input**: The matrix representation of $p$th word in all $N$ training samples: $\mathbf{X}_p, p = 1$ to $k$; and the label matrix for all $N$ training samples: $\mathbf{Y}$.
**Output**: $\mathbf{W}$
**Steps of learning sequence information**:
1. Random $\boldsymbol{\alpha}^z$, $\mathbf{U}$ and $\boldsymbol{\beta}^z$;
2. Set $\mathbf{Z}_0$ to all zeros;
3.
    for $p = 0$; $p \le k$ do
        Calculate $\mathbf{Z}_p = f(\mathbf{X}_p\boldsymbol{\alpha}^z + \mathbf{Z}_{p-1}\mathbf{U} + \boldsymbol{\beta}^z)$
    end
4. Obtain the final $\mathbf{Z}_k$.
**Steps of learning word importance**:
1. Artificially set $l$ for enhancement node $\mathbf{H}_p$;
2. Random $\boldsymbol{\alpha}^h$ and $\boldsymbol{\beta}^h$;
3.
    for $p = 1$; $p \le k$ do
        Calculate $\mathbf{H}_p = g(\mathbf{X}_p\boldsymbol{\alpha}^h + \boldsymbol{\beta}^h)$
    end
**Calculate connecting weights**:
    $\mathbf{W} = ([\mathbf{Z}_k|\mathbf{H}_1, \cdots, \mathbf{H}_k]^{\mathrm{T}}[\mathbf{Z}_k|\mathbf{H}_1, \cdots, \mathbf{H}_k])^{-1} [\mathbf{Z}_k|\mathbf{H}_1, \cdots, \mathbf{H}_k]^{\mathrm{T}}\mathbf{Y}$
```

words, the issues of vanishing and exploding gradient do not influence the performance of R-BLS. The pseudocode of the proposed R-BLS is illustrated in Algorithm 1.

4) Execution: For unseen samples in matrix form $\mathbf{X}^{\text{test}}$, in which the words have been mapped to numeric vector representation by the same trained embedding as in training procedure, the corresponding output labels can be calculated by

$$\mathbf{Y}^{\text{test}} = [\mathbf{Z}_k^{\text{test}}|\mathbf{H}_1^{\text{test}}, \ldots, \mathbf{H}_k^{\text{test}}]\mathbf{W} \tag{20}$$

where $\mathbf{Z}_k^{\text{test}}$ and $\mathbf{H}_p^{\text{test}}$ ($p = 1$ to k) are, respectively, calculated by (14)–(16) but over the unseen samples $\mathbf{X}^{\text{test}}$.

B. Gated Broad Learning System

Similar to R-BLS, the proposed G-BLS learns both *sequence information* and *word importance*, except that *an extra forget gate* is designed to control the learned *sequence information*. In R-BLS, all (even far ahead) previous words are correlated with the current processed word so that all previous sequence information are captured into current *memory set* $\mathbf{Z}_p$. However, in practice, some (far ahead) previous words may not necessarily correlate with the current word. For this reason, forget gate is specifically designed in G-BLS to forget the uncorrelated information and only deliver the correlated ones to $\mathbf{Z}_p$ for further processing.

1) Forget Gate: In Fig. 8, our proposed *forget gate* is illustrated in the G-BLS sequence block, which determines when the information in the *memory set* $\mathbf{Z}_p$, $p = 1$ to k, will be removed, that is, $\mathbf{Z}_p$ is reset to all zeros. In order to enable forget gate in G-BLS, a certain step size c is first determined, which specifies after how many steps the information in the *memory set* $\mathbf{Z}_p$ is removed. Note that, before removal, the original information in $\mathbf{Z}_p$ is kept connecting to the output nodes. To remove $\mathbf{Z}_p$, the following computation of sequence information (e.g., $\mathbf{Z}_{p+1}$) is simply based on $\mathbf{Z}_0$ rather than $\mathbf{Z}_p$. In detail

$$\mathbf{Z}_p = f(\mathbf{X}_p\boldsymbol{\alpha}^z + \mathbf{Z}_{p-1}\mathbf{U} + \boldsymbol{\beta}^z). \tag{21}$$

Then, for $\mathbf{Z}_{p+1}$

$$\mathbf{Z}_{p+1} = \begin{cases} f(\mathbf{X}_{p+1}\boldsymbol{\alpha}^z + \mathbf{Z}_0\mathbf{U} + \boldsymbol{\beta}^z), & \text{if } p \bmod c = 0 \\ f(\mathbf{X}_{p+1}\boldsymbol{\alpha}^z + \mathbf{Z}_p\mathbf{U} + \boldsymbol{\beta}^z), & \text{else} \end{cases} \tag{22}$$

where f is an activation function. $\boldsymbol{\alpha}^z$, $\mathbf{U}$, and $\boldsymbol{\beta}^z$ are randomly generated weights and bias.

In order to simplify the tuning procedure of parameter c, only one c is set for the input sentence. Simply speaking, the input sentence is separated into several short sentences of fixed length of c words. However, tuning only one c cannot guarantee the words in one short sentence are all correlated to each other. Fortunately, the analytically determined weights $\mathbf{W}^z$ can help reduce the influence of improperly tuned c. In detail, if c is not suitable for $\mathbf{Z}_p$ ($p = 1$ to k), the weight $\mathbf{W}_p^z$ connecting $\mathbf{Z}_p$ with $\mathbf{Y}$ will automatically become smaller or

Algorithm 2 G-BLS

```
Input: The matrix representation of pth word in all $N$ training samples: $\mathbf{X}_p, p = 1$ to $k$; and the label matrix for all $N$ training samples: $\mathbf{Y}$.
Output: $\mathbf{W}$
Steps of learning sequence information:
1. Random $\boldsymbol{\alpha}^z$, $\mathbf{U}$ and $\boldsymbol{\beta}^z$;
2. Set $\mathbf{Z}_0$ to all zeros;
3. Artificially set a certain step size $c$;
4.
    for $p = 0$; $p \le k$ do
        Calculate $\mathbf{Z}_p = f(\mathbf{X}_p\boldsymbol{\alpha}^z + \mathbf{Z}_{p-1}\mathbf{U} + \boldsymbol{\beta}^z)$;
        Calculate $\mathbf{Z}_{p+1} = \begin{cases} f(\mathbf{X}_{p+1}\boldsymbol{\alpha}^z + \mathbf{Z}_0\mathbf{U} + \boldsymbol{\beta}^z), & \text{if } p \bmod c = 0 \\ f(\mathbf{X}_{p+1}\boldsymbol{\alpha}^z + \mathbf{Z}_p\mathbf{U} + \boldsymbol{\beta}^z), & \text{else} \end{cases}$
    end
4. Obtain $\mathbf{Q} = [\mathbf{Z}_p]$, where $p \bmod c = 0$.

Steps of learning word importance:
1. Artificially set $l$ for enhancement node $\mathbf{H}_p$;
2. Random $\boldsymbol{\alpha}^h$ and $\boldsymbol{\beta}^h$;
3.
    for $p = 1$; $p \le k$ do
        Calculate $\mathbf{H}_p = g(\mathbf{X}_p\boldsymbol{\alpha}^h + \boldsymbol{\beta}^h)$
    end
Calculate connecting weights:
    $\mathbf{W} = ([\mathbf{Q}|\mathbf{H}_1, \cdots, \mathbf{H}_k]^{\mathrm{T}}[\mathbf{Q}|\mathbf{H}_1, \cdots, \mathbf{H}_k])^{-1} [\mathbf{Q}|\mathbf{H}_1, \cdots, \mathbf{H}_k]^{\mathrm{T}}\mathbf{Y}$
```

less important than other weights $\mathbf{W}_q^z$ ($q = 1$ to k and $q \neq p$), in which c is suitable for $\mathbf{Z}_q$. This is because, compared to $\mathbf{Z}_p$ with improper c, $\mathbf{Z}_q$ with proper c is more effective for prediction. Hence, the parameter c combined with the weights $\mathbf{W}^z$ can practically control the flow of information to improve its effectiveness, which is similar to LSTM using several gates to control the flow of information. Moreover, comparing Fig. 5 with Fig. 8, LSTM only has *LSTM memory block* to learn sequence information, whereas in the proposed G-BLS, *G-BLS sequence block* and *G-BLS word block* are simultaneously used to learn both sequence information and word importance.

Consequently, the sequence information for G-BLS is represented by $\mathbf{Q} = [\mathbf{Z}_p]$, where $p \bmod c = 0$. For example, given $c = 3$ and $k = 15$, $\mathbf{Q} = [\mathbf{Z}_3, \mathbf{Z}_6, \mathbf{Z}_9, \mathbf{Z}_{12}, \mathbf{Z}_{15}]$.

2) Connecting Weights for G-BLS: Similar to R-BLS, the connecting weight $\mathbf{W}$ for G-BLS is

$$\begin{aligned}\mathbf{W} &= \mathbf{A}^{\dagger}\mathbf{Y} \\ &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}\end{aligned} \tag{23}$$

where $\mathbf{A} = [\mathbf{Q}|\mathbf{H}_1, \ldots, \mathbf{H}_k]$ and $\mathbf{H}_p = g(\mathbf{X}_p\boldsymbol{\alpha}^h + \boldsymbol{\beta}^h)$ as in R-BLS. The pseudocode of the proposed G-BLS is illustrated in Algorithm 2. Moreover, the execution procedure of G-BLS is similar to that of R-BLS.

IV. Experiments and Evaluations

In this section, the proposed R-BLS and G-BLS are compared with LSTM. From [31], if the forget gate bias of LSTM is set to 1, LSTM performs almost the same as another popular method gated recurrent unit (GRU) [32], [33]. For this reason, GRU is not compared in this article. In our experiments, we tested both the "original LSTM" and the "LSTM with bias 1." For simplicity, in tables of the following sections, only the better experimental results between these two methods are selected and shown.

A. Experimental Setup

The comparison was conducted over 13 real-world datasets for text classification from CrowdFlower[1] and DL4J[2] about assigning subject categories, topics, or genres, sentiment analysis, etc. Before experiments, all raw data (in text form) in each dataset are preprocessed with the following steps: 1) erasing the punctuation, conversion to lower case, and then tokenization [34]; 2) since the samples (sentences) are of different lengths, every sentence is practically truncated or padded to a fixed length of L words [35]; and 3) using the *trainWordEmbedding* function in the MATLAB environment (Text Analytics Toolbox) to train a word embedding. Then with the trained word embedding, *word2vec* function is employed to map the words in the vocabulary to numeric vector representations. In *trainWordEmbedding*, two parameters are manually defined: 1) *Mincount* and 2) *Dimension*. *MinCount* is the minimum word count to be included in the embedding. For the construction of vocabulary, the words in a text dataset with occurrence frequency less than *MinCount* are discarded from the vocabulary. *Dimension* is the dimension of the resulting embedding, that is, the dimension of each word vector. As a result, the parameters *Mincount* and *Dimension* determine the trained word embedding. These two parameters with optimal values along with the data properties are shown in Table I.

For each dataset, 80% of the data are randomly selected for training and the remaining 20% for testing. Moreover, the results are collected after ten runs to show the stability of the compared methods.

The overall performance is measured and compared in both *test accuracy* and *training time*, followed by the *analysis of word importance*.

All the experiments were conducted under MATLAB on a PC of 3.60 GHz with 32-GB RAM.

B. Additional Notes for Experimental Setup

1) Stop words are not removed from the vocabulary. That is because, stop words include not and "no," etc. Removing these stop words (e.g., not and no) may lower the accuracy. For instance, removing "not" from sentence "I am not happy," the sentiment of this sentence is totally different.

[1]https://www.crowdflower.com/data-for-everyone/
[2]https://deeplearning4j.org/cn/opendata

TABLE I
PROPERTIES OF DATASETS

Dataset	#Samples	max(#words)[1]	#Classes	*MinCount*	*Dimension*
Apple-Twitter-Sentiment	3886	29	4	15	50
claritin-october-twitter	4900	30	3	5	30
Twitter-sentiment	7156	30	6	15	50
objective-sentence	8344	10	5	5	10
Airline-Sentiment	14640	35	3	15	50
New-years-resolutions	5011	31	10	15	50
progressive-tweet	1159	29	4	5	30
electronic-sentiment	2000	2216	2	15	50
tweet-global-warming	6090	31	3	5	50
corporate-messaging	3118	29	4	15	30
books-sentiment	2000	6719	2	15	50
political-media	5000	2624	9	5	50
weather-sentiment	1000	31	4	5	20

[1] max(#words): means the number of words in the longest sample (or sentence).

TABLE II
EXPERIMENTAL RESULTS ALONG WITH DIFFERENT PAIRS OF MINCOUNT AND DIMENSION FOR DATASET AIRLINE-SENTIMENT

Mincount, Dimension	G-BLS[1]	LSTM [2]
15, 50	***75.26±0.23***	***73.34±2.26***
5, 50	73.08±0.43	69.59±1.59
20, 50	74.89±0.40	73.10±2.32
30, 50	73.61±0.36	71.22±2.53
15, 10	71.03±0.52	69.01±0.83
15, 100	73.49±0.41	71.42±2.01

[1] For fair comparison, other parameters, such as "#hidden nodes l" and "fixed length L", are fixed in both G-BLS and LSTM. $l = 90$ and $L = 30$ for G-BLS;
[2] $l = 200$ and $L = 30$ for LSTM;

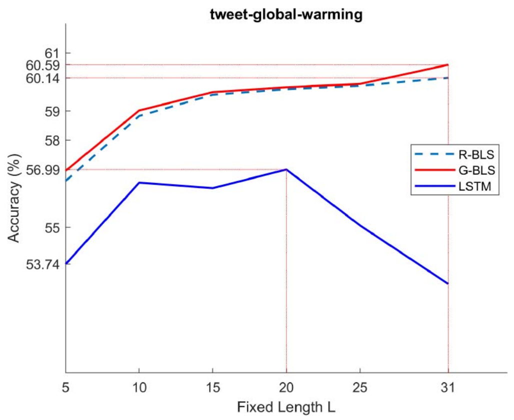

Fig. 9. Test accuracy for dataset *tweet-global-warming* along with increasing fixed length of L words.

2) The parameters *Mincount* and *Dimension* determine the trained word embedding. Noteworthy, good word embedding enables any method to achieve its best result. Hence, for one dataset, it is necessary to only show one pair of *Mincount* and *Dimension* (Table I). In Table II for dataset *Airline-Sentiment*, when *Mincount* < 15 and *Mincount* > 15, poor performances are simultaneously observed in both this article and compared methods. Similar results (but not shown) are also observed on other datasets in the experiments. Therefore, the range of *Mincount* $= 5$–15 is to be fine-tuned in our experiments. Similarly, the parameter *Dimension* $= 10$–50 is to be fine-tuned as well.
3) In all experiments, the activation functions f and g for BLSs (i.e., R-BLS and G-BLS) are *sigmoid*.
4) Deep LSTM is also considered in the experiments, in which both *hidden nodes* and *LSTM-layers* are tuned to achieve the optimal results. For instance, in *corporate-messaging* in Table III, there are two LSTM-layers and each layer has 3000 hidden nodes in order to achieve its optimal results for this dataset. For other datasets, such as *objective-sentence* in Table III, one LSTM-layer with 100 hidden nodes already enables LSTM to achieve its optimal result.
5) As detailed in Section IV-A, truncating or padding every sentence to a fixed length L is required [35] before training. In this article, the sentences in all datasets are first padded with zeros to the length of their respective max(#words). Then every sentence is truncated to a tuned length L that enables the method to achieve its optimal result. In detail, for dataset *tweet-global-warming*, BLSs achieve their optimal results when L =max(#words) $= 31$ and LSTM gets its best result when $L = 20$, as illustrated in Fig. 9. Hence, in this article, L is also tuned for each compared method and dataset, and the optimal L are also shown in Table III.
6) The parameter c in G-BLS is an integer that is tuned from 2 to L.

C. Test Accuracy

1) BLSs Versus LSTM: Table III shows the test accuracies of the compared methods overall datasets. From the results, BLSs (R-BLS and G-BLS) totally outperform LSTM up to 13.30% (*weather-sentiment*). It is because, according to (19) and (23), BLSs have closed-form solutions and hence avoid the suffering of vanishing and exploding gradient problems. Moreover, benefited from the flat neural-network mechanism of BLS, *sequence information* along with *word importance* are simultaneously learned for higher effectiveness in text classification. G-BLS performs the best overall dataset because of the extra forget gate to abandon the unnecessary information (i.e., equivalent to only retain the necessary one). Since LSTM is derived from RNN, it only learns *sequence information* for text classification [13] while *word importance*

TABLE III
TEST ACCURACY AND TRAINING TIME

Dataset	Methods	#Hidden nodes l[1]	Fixed Length L[2]	Accuracy (%)	Improved[3] (%)	Training Time (s)	Time Ratio
Apple-Twitter-Sentiment	R-BLS	10	29	67.34±1.00	0.53	***0.11***	1/744
	G-BLS	10	29	***67.76±0.67*** (c=15)[4]	0.95	0.15	1/546
	LSTM	2000	29	66.81±0.73	\	81.89	1
claritin-october-twitter	R-BLS	110	7	***98.35±0.06***	1.04	***0.20***	1/409
	G-BLS	110	7	***98.43±0.07*** (c=5)	1.04	***0.20***	1/409
	LSTM	2000	7	97.31±0.15	\	81.72	1
Twitter-sentiment	R-BLS	80	10	58.85±0.55	1.20	***0.36***	1/138
	G-BLS	90	10	***59.11±0.38*** (c=3)	1.46	0.55	1/90
	LSTM	1000	30	57.65±0.00	\	49.76	1
objective-sentence	R-BLS	50	10	35.27±0.44	1.39	***0.18***	1/63
	G-BLS	50	10	***35.39±0.42*** (c=4)	1.51	0.19	1/59
	LSTM	100	10	33.88±0.16	\	11.27	1
Airline-Sentiment	R-BLS	90	30	75.08±0.40	1.74	***6.45***	1/5
	G-BLS	90	30	***75.26±0.23*** (c=3)	1.92	9.5	1/3
	LSTM	200	30	73.34±2.26	\	30.99	1
New-years-resolutions	R-BLS	90	8	38.20±1.04	2.21	***0.23***	1/121
	G-BLS	90	8	***38.59±1.01*** (c=3)	2.60	0.25	1/111
	LSTM	1000	10	35.99±0.21	\	27.73	1
progressive-tweet	R-BLS	200	29	47.03±1.03	2.68	***0.49***	1/105
	G-BLS	200	29	***48.45±1.15*** (c=15)	4.10	0.52	1/99
	LSTM	3000	29	44.35±0.82	\	51.28	1
electronic-sentiment	R-BLS	150	100	66.10±1.47	2.80	***4.39***	1/2
	G-BLS	150	100	***66.35±0.79*** (c=70)	3.05	4.45	1/2
	LSTM	500	50	63.30±3.67	\	10.75	1
tweet-global-warming	R-BLS	45	31	60.14±0.71	3.15	***0.89***	1/43
	G-BLS	60	31	***60.59±0.91*** (c=3)	3.60	3.02	1/13
	LSTM	1000	20	56.99±1.48	\	38.20	1
corporate-messaging	R-BLS	50	29	81.35±0.81	3.88	***0.52***	1/770
	G-BLS	50	29	***81.87±1.13*** (c=7)	4.40	0.63	1/636
	LSTM	(3000, 3000)[5]	29	77.47±1.29	\	400.60	1
books-sentiment	R-BLS	100	35	61.18±0.71	5.13	***0.93***	1/29
	G-BLS	100	35	***62.25±1.36*** (c=8)	6.20	0.99	1/28
	LSTM	300	250	56.05±0.61	\	27.30	1
political-media	R-BLS	30	400	38.03±0.71	10.23	***31.45***	1/3
	G-BLS	30	400	***38.62±0.52*** (c=200)	10.82	32.56	1/2
	LSTM	200	400	27.80±0.00	\	79.03	1
weather-sentiment	R-BLS	100	31	45.15±1.68	11.75	***0.23***	***1/867****
	G-BLS	100	31	***46.70±1.84*** (c=4)	***13.30****	0.25	1/798
	LSTM	(3000, 3000, 3000)	20	33.40±2.12	\	199.39	1

[1] l is for BLSs, indicating the number of hidden nodes in each enhancement node $\mathbf{H}_p$. Note that, the number of hidden nodes for $\mathbf{Z}_p$ in the feature layer is fixed and equals to d, where d is the dimension of each word vector;
[2] Fixed length L for each sentence (or sample) in the dataset;
[3] The accuracy increased by BLSs, compared to LSTM;
[4] c is the parameter in G-BLS, 'c=15' here means the optimized c for *Apple-Twitter-Sentiment* is 15;
[5] In LSTM, two LSTM-layers are used and each layer has 3000 hidden nodes for *corporate-messaging*.

for final discrimination is ignored or unable to learn. Illustrated from the experimental results, the higher effectiveness of BLSs simultaneously using *sequence information* and *word importance* for text classification is verified.

2) R-BLS Versus G-BLS: In Table III, G-BLS achieves better performance than R-BLS for almost all compared datasets due to the extra forget gate in G-BLS (determined by both parameter c and weights $\mathbf{W}^z$) to abandon the unnecessary information (or equivalent to only retain the necessary one). Obviously, G-BLS can achieve at least the same accuracy of R-BLS by forgetting nothing, that is, c is set to fixed length L, as illustrated by *claritin-october-twitter* in Table III. Although the effectiveness of R-BLS is slightly worse than that of G-BLS, R-BLS is easier to implement than G-BLS because R-BLS does not need to tune the extra parameter c.

D. Training Time

1) BLSs Versus LSTM: As shown in Table III, LSTM takes several hundred times of training time longer than BLSs over half of the compared datasets. Over *weather-sentiment*, R-BLS takes only 1/867 of the training time of LSTM because LSTM adopts iterative gradient descent strategy to optimize all the parameters [13], including the weights and bias in every gate. On the contrary, BLSs only need to analytically calculate the connecting weight $\mathbf{W}$ while other weights and bias (such as $\boldsymbol{\alpha}^z$ and $\boldsymbol{\beta}^z$) are all randomly generated. Hence, BLSs are significantly more efficient than LSTM over most compared datasets.

However, over *Airline-Sentiment*, *electronic-sentiment*, and *political-media*, BLSs can be only several times faster than LSTM in terms of training time. From the experimental settings of these three datasets, it can be observed that BLSs take relatively more hidden nodes l or larger fixed length L compared to the settings of other datasets for optimal results. In other words, BLSs are sensitive to l and L because it needs to compute an inverse matrix for the calculation of connecting weight $\mathbf{W}$, and the dimension of the inverse matrix is determined by both l and L as shown in (19) and (23). Noteworthy,

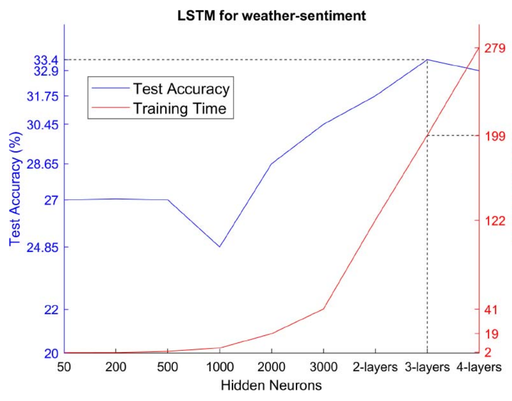

Fig. 10. Test accuracy and training time of LSTM for *weather-sentiment* along the number of hidden nodes. *Remark:* In the x-axis, 50–3000 hidden nodes are under 1 LSTM-layer. For 2–4 LSTM-layers, 3000 hidden nodes for each LSTM-layer are utilized.

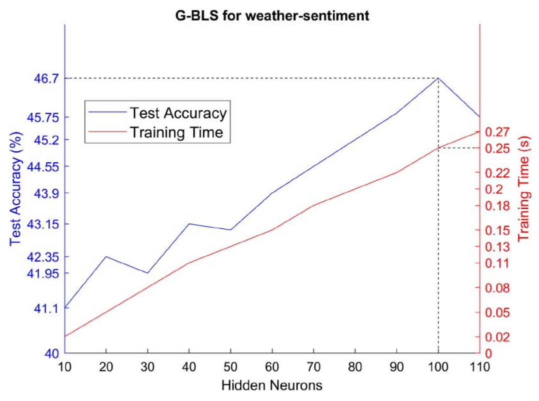

Fig. 11. Test accuracy and training time of G-BLS for *weather-sentiment* along the number hidden nodes in each enhancement node $\mathbf{H}_p$.

although taking more training time on these three datasets, BLSs are still much faster than LSTM with higher accuracies (up to 10.82% in *political-media*) as shown in Table III. In the future, we would try to further improve the efficiency of BLSs to make them not so sensitive to l and L.

2) R-BLS Versus G-BLS: In Table III, G-BLS is slightly slower than R-BLS because its inverse matrix is slightly larger than that of R-BLS, as shown in (19) and (23). In detail, the dimension of the inverse matrix of R-BLS is $(d + L \times l)$ and that of G-BLS is $(\lceil L/c \rceil \times d + L \times l)$, where d is the dimension of the words (shown in Table I), and $L/c \geq 1$. As a result, the time complexities to calculate the inverse matrices in R-BLS and G-BLS are, respectively, represented by $O((d + L \times l)^3)$ and $O((\lceil L/c \rceil \times d + L \times l)^3)$. Hence, R-BLS is slightly more efficient than G-BLS because of its simpler structure.

E. Analysis of Word Importance

1) Significance of Word Importance in BLSs When Compared to Deep LSTM: For some complicated datasets, such as *corporate-messaging* and *weather-sentiment* in Table III, deep architecture of LSTM is required to achieve optimal accuracies. In Fig. 10 of LSTM for *weather-sentiment*, when three LSTM-layers are adopted, LSTM achieves its optimal accuracy of 33.40%. As illustrated in Fig. 11, BLSs with 100 hidden nodes in each $\mathbf{H}_p$ already achieves its best accuracy of 46.70% (G-BLS is used as an example of BLSs). Even LSTM constructs a deep and complicated structure to extract deeper representation of *sequence information*, it still cannot be compared with simple BLSs (100 hidden nodes only) with additional information about *word importance*. Note that it is nontrivial to incorporate *word importance* and *sequence information* simultaneously in RNN or LSTM. To our best knowledge, we cannot find such simultaneous learning mechanism of multiple information for RNN or LSTM. Moreover, in Fig. 10, compared to a single LSTM-layer, more LSTM-layers require much more training time. In summary, BLSs are much more effective and efficient than deep LSTM in the current study.

2) Significance of Word Importance in Boosting Performance in BLSs: Since the weights for learning *sequence information* are randomly generated in the feature layer of our proposed BLSs, the extracted/learned *sequence information* is not as diverse as that in LSTM. Hence, only using *sequence information* in BLSs usually cannot obtain as high accuracies as that in LSTM.

On the other hand, higher accuracies than LSTM are observed when considering both *sequence information* and *word importance* in our proposed BLSs, as shown in Table III. Moreover, the *word importance* in the proposed BLSs are analytically determined by weight $\mathbf{W}_h$, as shown in Fig. 7. Hence, in our proposed BLSs, *word importance* plays a more important role than *sequence information* in boosting classification performances.

3) Limitation of Word Importance: For datasets, such as *Apple-Twitter-Sentiment*, *claritin-october-twitter*, *Twitter-sentiment*, and *objective-sentence* in Table III, BLSs can only earn minor improvements ($\approx 1\%$) over LSTM, whose reason is detailed below.

LSTM considers *sequence information* only. For *claritin-october-twitter*, LSTM already achieves an accuracy of 97.31%. In other words, for this dataset, *sequence information* (rather than other discriminant information, such as *word importance*) is the most significant information to discriminate the different classes in the dataset. Hence, even with the additional information of *word importance*, BLSs do not have significant improvement. However, BLSs provide a signification reduction (about 1/409) of the training time of LSTM. This is already a significant contribution.

For the remaining three datasets with minor improvements, there may be other unidentified discriminant information (in addition to *word importance*). It is because even with the additional information of *word importance*, BLSs cannot earn as much improvement (>2%) as in other datasets.

In the future, we will continue the excavation of this discriminant information. Once the information is identified, our proposed BLSs could easily and simultaneously learn this information for further improvement on the effectiveness of text classification or other NLP tasks. For instance, two recent works [36], [37] give a new viewpoint that language could also be thought of as signal, that is, "characters for texts" is equivalent to "pixels for images". In the future, *character importance* or *prefix-suffix importance* can be considered as other discriminant information other than *word importance*.

Moreover, recently, a few language representation models, such as BERT [38] and ELMo [39], are proposed, which apply pretrained language representations to downstream tasks. Motivated by these two models, in the future, the proposed R-BLS and G-BLS can be first used to train word embedding using large corpus and then the model parameters (e.g., output weights) are updated by the specific downstream tasks data using recursive least squares [40].

F. Summary

1) BLSs Versus LSTM: Considering both test accuracy and training time, BLSs achieve much better performance than LSTM in almost all datasets, especially when compared to deep LSTM.

2) R-BLS Versus G-BLS: For effectiveness: G-BLS>R-BLS, while for efficiency: R-BLS>G-BLS. Moreover, compared to G-BLS, R-BLS is simpler and does not need to tune the parameter c. In practice, the user can first adopt R-BLS to obtain a base model. Then for further improvement on effectiveness, G-BLS can be used.

V. Conclusion

In this article, an attractive advantage of flat neural networks (e.g., BLS) that can simultaneously learn multiple information is discovered and applied in real-world applications. In detail, two novel efficient RNN and LSTM-like architectures: R-BLS and G-BLS are proposed to learn *sequence information* and *word importance* for text classification, while traditional RNN or LSTM can only learn *sequence information*. Benefited from the random projection mechanism of BLS, the training time of the proposed BLSs is up to hundred times faster than RNN or LSTM. Moreover, BLSs have closed-form solutions so that the vanishing and exploding gradient problems in RNN and LSTM do not influence the performance of BLSs. Furthermore, the proposed G-BLS has an extra forget gate (similar to LSTM) to abandon the unnecessary information (equivalent to only retain the necessary one).

The proposed BLSs have been compared with the popular LSTM over 13 real-world datasets for text classification about assigning subject categories, topics, or genres, sentiment analysis, etc. The experimental results show that our proposed BLSs can improve the accuracy and also significantly the training time. The contributions of these works are summarized as follows.

1) Different from deep neural networks in which user cannot specify the learned information in each layer, flat neural networks can specify the information learned in each subnetwork.
2) BLSs totally outperform LSTM in terms of test accuracy up to 13.30%.
3) BLSs take significantly less training time than LSTM on most compared datasets, especially for *weather-sentiment*, in which R-BLS took only 1/867 of the training time of LSTM.
4) Benefited from *word importance*, simple BLSs achieve much better performance (including both test accuracy and training time) than LSTM using deep and complicated structure.
5) Other discriminant information (to be excavated) can be easily and simultaneously learned using our proposed BLSs for further improvement.
6) The proposed G-BLS is more effective than the proposed R-BLS while R-BLS is more efficient than G-BLS. In practice, the user can first adopt R-BLS to obtain a base model. Then G-BLS can be used for further improvement on effectiveness.

In a nutshell, the proposed R-BLS and G-BLS are highly effective and efficient than (deep) LSTM for text classification. In the near future, we would apply R-BLS and G-BLS to resolve other NLP tasks such as machine translation.

References

[1] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.

[2] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Aug. 2011.

[3] F. Zhuang, P. Luo, C. Du, Q. He, Z. Shi, and H. Xiong, "Triplex transfer learning: Exploiting both shared and distinct concepts for text classification," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1191–1203, Jul. 2014.

[4] C.-H. Wang, C.-Y. Chen, and K.-N. Hung, "Toward a new task assignment and path evolution (TAPE) for missile defense system (MDS) using intelligent adaptive SOM with recurrent neural networks (RNNs)," *IEEE Trans. Cybern.*, vol. 45, no. 6, pp. 1134–1145, Jun. 2015.

[5] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. ACM 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[7] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 263–269, 1989.

[8] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[9] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE ICASSP*, 2013, pp. 6645–6649.

[10] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *IEEE Potentials*, vol. 13, no. 4, pp. 27–31, Oct./Nov. 1994.

[11] G. P. Zhang, "Neural networks for classification: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 30, no. 4, pp. 451–462, Nov. 2000.

[12] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertainty Fuzziness Knowl. Based Syst.*, vol. 6, no. 2, pp. 107–116, 1998.

[15] D. Tao, X. Lin, L. Jin, and X. Li, "Principal component 2-D long short-term memory for font recognition on single Chinese characters," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 756–765, Mar. 2016.

[16] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *Proc. IEEE ICASSP*, 2015, pp. 4520–4524.

[17] M. Ghiassi, M. Olschimke, B. Moon, and P. Arnaudo, "Automated text classification using a dynamic artificial neural network model," *Expert Syst. Appl.*, vol. 39, no. 12, pp. 10967–10976, 2012.

[18] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. ICLR*, 2015, pp. 1–16.

[19] D. Huang, C.-D. Wang, and J.-H. Lai, "Locally weighted ensemble clustering," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1460–1473, May 2018.
[20] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, no. 2, pp. 139–157, 2000.
[21] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. New York, NY, USA: Springer, 2012.
[22] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann, "Review of classifier combination methods," in *Machine Learning in Document Analysis and Recognition*. Berlin, Germany: Springer-Verlag, 2008, pp. 361–386.
[23] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.
[24] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, Dec. 2006.
[25] 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.
[26] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Int. Speech Commun. Assoc.*, Chiba, Japan, Sep. 2010, pp. 1045–1048.
[27] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, "Translating videos to natural language using deep recurrent neural networks," 2014. [Online]. Available: arXiv:1412.4729.
[28] X.-Y. Zhang, F. Yin, Y.-M. Zhang, C.-L. Liu, and Y. Bengio, "Drawing and recognizing Chinese characters with recurrent neural network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 849–862, Mar. 2018.
[29] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," 2016. [Online]. Available: arXiv:1605.05101.
[30] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
[31] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 2342–2350.
[32] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014. [Online]. Available: arXiv:1406.1078.
[33] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014. [Online]. Available: arXiv:1412.3555.
[34] S. Bird, E. Klein, and E. Loper, *Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit*. Sebastopol, CA, USA: O'Reilly Media, 2009.
[35] J. Reyes, D. Ramírez, and J. Paciello, "Automatic classification of source code archives by programming language: A deep learning approach," in *Proc. IEEE CSCI*, 2016, pp. 514–519.
[36] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.
[37] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," 2016. [Online]. Available: arXiv:1606.01781.
[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018. [Online]. Available: arXiv:1810.04805.
[39] M. E. Peters *et al.*, "Deep contextualized word representations," 2018. [Online]. Available: arXiv:1802.05365.
[40] S. Haykin, "Adaptive filter theory," in *Information and System Sciences Series*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.

Jie Du received the B.S. degree in computer science and technology from the University of Shihezi, Shihezi, China, in 2013, and the Ph.D. degree from the Department of Computer and Information Science, University of Macau, Macau, China, in 2019.

She is currently an Assistant Professor with the School of Biomedical Engineering, Health Science Center, Shenzhen University, Shenzhen, China. Her research interests include machine learning methods and medical image processing.

Chi-Man Vong (Senior Member, IEEE) received the M.S. and Ph.D. degrees in software engineering from the University of Macau, Macau, China, in 2000 and 2005, respectively.

He is currently an Associate Professor with the Department of Computer and Information Science, University of Macau. His current research interests include machine learning methods and intelligent systems.

C. L. Philip Chen (Fellow, IEEE) received the graduation degree from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1985.

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, 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 systems, cybernetics, and computational intelligence.

Dr. Chen was a recipient of the 2016 Outstanding Electrical and Computer Engineers Award from his alma mater, Purdue University, and 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 and 2019. He was the IEEE Systems, Man, and Cybernetics Society President from 2012 to 2013. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS from 2014 to 2019 and has been an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS and the IEEE TRANSACTIONS ON CYBERNETICS. He was the Chair of TC 9.1 Economic and Business Systems of International Federation of Automatic Control from 2015 to 2017. He is currently a Vice President of the Chinese Association of Automation (CAA). He is a fellow of AAAS, IAPR, CAA, and HKIE and a member of Academia Europaea, European Academy of Sciences and Arts, and International Academy of Systems and Cybernetics Science.