Neural Networks 140 (2021) 237–246

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Bidirectional stochastic configuration network for regression problems☆

Weipeng Cao, Zhongwu Xie, Jianqiang Li, Zhiwu Xu, Zhong Ming, Xizhao Wang*

College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

ARTICLE INFO

Article history:
Received 15 July 2020
Received in revised form 29 January 2021
Accepted 5 March 2021
Available online 18 March 2021

Keywords:
Stochastic configuration network
Random vector functional link network
Randomized algorithms
Neural networks with random weights
Constructive neural networks

ABSTRACT

To adapt to the reality of limited computing resources of various terminal devices in industrial applications, a randomized neural network called stochastic configuration network (SCN), which can conduct effective training without GPU, was proposed. SCN uses a supervisory random mechanism to assign its input weights and hidden biases, which makes it more stable than other randomized algorithms but also leads to time-consuming model training. To alleviate this problem, we propose a novel bidirectional SCN algorithm (BSCN) in this paper, which divides the way of adding hidden nodes into two modes: forward learning and backward learning. In the forward learning mode, BSCN still uses the supervisory mechanism to configure the parameters of the newly added nodes, which is the same as SCN. In the backward learning mode, BSCN calculates the parameters at one time based on the residual error feedback of the current model. The two learning modes are performed iteratively until the prediction error of the model reaches an acceptable level or the number of hidden nodes reaches its maximum value. This semi-random learning mechanism greatly speeds up the training efficiency of the BSCN model and significantly improves the quality of the hidden nodes. Extensive experiments on ten benchmark regression problems, two real-life air pollution prediction problems, and a classical image processing problem show that BSCN can achieve faster training speed, higher stability, and better generalization ability than SCN.

1. Introduction

Randomized neural networks are a special type of feed-forward neural networks and its representative algorithms include the random vector functional link network (RVFL) (Pao & Takefuji, 1992), the neural network with random weights (NNRW) (Schmidt, Kraaijveld, & Duin, 1992), etc. The most notable feature of this type of neural networks is that their input weights (i.e., the weights between the input layer and the hidden layer) and hidden biases (i.e., the thresholds of hidden nodes) are assigned randomly according to certain rules and remain unchanged throughout the training process of the model, while the output weights are obtained analytically. This non-iterative training mechanism enables them to train faster than traditional neural networks such as the back-propagation algorithm (BP) (Rumelhart, Durbin, Golden, & Chauvin, 1995) and work better on many platforms with limited hardware resources (e.g., various IoT terminals Xu, Zhang, Gao, Xue, Qi, & Dou, 2020), so it has been widely concerned and applied in many scenarios in recent years (Zhang, Li, Li, & Wang, 2020).

However, most of the existing randomized neural networks suffer from two notorious weaknesses, that is, (a) the quality of the random parameters (i.e., the input weights and hidden biases) is hard to be guaranteed and (b) the number of hidden nodes is difficult to be determined before modeling. For the former problem, several empirical guidelines are given in Li and Wang (2017), but they can only work in some specific scenarios. There are also some existing solutions targeting at the latter problem, which can be divided into two categories: constructive strategy and pruning strategy. The basic idea of the constructive strategy is to start the model with a simple network structure and then gradually increase the hidden nodes until the performance of the model reaches the preset conditions. Incremental RVFL (I-RVFL) is one of the representative algorithms using this strategy (Li & Wang, 2017). The pruning strategy starts the model with a very large network structure and then deletes the unimportant hidden nodes according to certain criteria. For example, in Ertuğrul (2018), the author sorted the importance of hidden nodes according to the output weights and the coefficient of variation of the hidden matrix, and then removed the relatively unimportant nodes. These two strategies have effectively reduced the labor of parameter tuning, but rarely consider the quality of

☆ This work was supported by National Natural Science Foundation of China (61976141, 61732011, and 61836005) and Guangdong Science and Technology Department, China (2018B010107004).

* Corresponding author.
E-mail address: xizhaowang@ieee.org (X. Wang).

https://doi.org/10.1016/j.neunet.2021.03.016
0893-6080/

the input parameters, which cannot guarantee the generalization ability of the corresponding models. Therefore, the above two problems still hinder the extensive application of randomized models in practice.

To alleviate the above problems, Wang and Li (2017b) proposed a constructive randomized algorithm called stochastic configuration network (SCN) in 2017. Compared with other randomized neural networks, SCN uses a supervisory random mechanism to assign the input weights and hidden biases of hidden nodes, which enables it to have better stability and generalization ability. Moreover, SCN can automatically search for the number of hidden nodes that can make the model achieve an expected accuracy, which greatly reduces the workload of parameter tuning. These advantages have made SCN quickly attracted extensive attention, and various variants have been proposed. Some notable work based on SCN include: In Wang and Li (2017b), Wang DH et al. theoretically proved that the method of generating random parameters using the supervisory mechanism can guarantee the universal approximation ability of the randomized algorithms, which lays a theoretical foundation for SCN. Later, they proposed a hybrid method by combining SCN with kernel density estimation (KDE) (Wang & Li, 2017a) and applied it to solve the uncertain data modeling problems. Moreover, they proposed an ensemble learning algorithm based on SCN from the perspective of heterogeneous features fusion, and adopted the negative correlation learning strategy (NCL) to evaluate its output weights (Wang & Cui, 2017). The new algorithm effectively improves the robustness of the model. To further improve the modeling performance of SCN, Wang and Li (2018) proposed a deep SCN with a multi-hidden layer network structure. Some interesting properties and improved modeling performance can be observed from Deep SCN. Li and Wang (2019) extended the original SCN framework with two dimensional inputs for image data informatics, showing good potential for fast image processing. In Pratama and Wang (2019), the authors improved the original SCN to make it have the ability to deal with data stream learning problems. On this basis, they used the stacking strategy to expand it to a deep architecture to handle complex and non-stationary data stream scenarios. In Huang, Huang, and Wang (2019), Huang CQ et al. designed an adaptive power storage replica management system based on SCN to evaluate and analyze the traffic state of power data networks. In addition, SCN is applied to the data modeling in process industries (Dai, Li, Zhou, & Chai, 2019), workload forecasting in geo-distributed cloud data centers (Bi, Yuan, Zhang, & Zhang, 2019), carbon residual prediction of crude oil (Lu & Ding, 2019a), prediction of key variables in industrial process (Lu & Ding, 2019b), component concentrations forecasting in sodium aluminate liquor (Wang & Wang, 2020), and the interval prediction in the industrial process (Lu, Ding, Dai, & Chai, 2020).

Although SCN and its variants have played significant roles in many applications, they still suffer from a common weakness, that is, they spend too much time searching for candidate input parameters during the model training process. Specifically, when adding a new hidden node, they need to prepare multiple candidates that meet the preset conditions through the above-mentioned supervisory mechanism, and then select the one that can reduce the current residual error greatest as the new node. This training mechanism ensures that the error of the model is monotonically decreasing, but causes the training process to be very time-consuming, especially when the number of candidates is large or the residual error becomes small. Note that if the number of candidates were set too small, the quality of some hidden nodes may be poor, which would reduce the convergence rate of the model and could not get a compact architecture with good generalization ability.

To solve the above problem, we have optimized the process of adding hidden nodes in SCN and proposed a novel semi-random constructive algorithm called bidirectional stochastic configuration network (BSCN), which includes two learning modes: forward learning for the odd nodes and backward learning for the even nodes. Specifically, when a new hidden node is ready to be added, if its order is odd (e.g., the first node), BSCN uses a supervisory mechanism to find appropriate input parameters for it (called forward learning), which is exactly the same as SCN; otherwise (i.e., the order is even, such as the second hidden node), BSCN calculates its input parameters at one time according to the current residual error feedback (called backward learning). During training, these two learning modes proceed in turn. The forward learning naturally inherits the advantages of the original SCN, that is, the supervisory method can improve the quality of hidden nodes to a certain extent and guarantee the universal approximation ability of the model (Wang & Li, 2017b); and the backward learning can avoid the problem of excessive time consumption caused by finding a large number of candidates, and the hidden nodes obtained in each step can minimize the residual error at that time. Therefore, this bidirectional learning mechanism enables BSCN to have the following advantages:

(1) Naturally inherits the universal approximation ability possessed by the SCN model;
(2) Greatly accelerates the training efficiency of the model;
(3) The quality of hidden nodes is effectively improved, which in turn makes the trained model better in generalization ability and more compact in network structure.

We verified the effectiveness of BSCN on ten benchmark regression problems, two real-life air pollution prediction problems, and a classical problem of age estimation from a single face image. Experimental results show that, compared with SCN, BSCN has not only much faster training speed but also better generalization ability and stability. Moreover, compared with other typical constructive neural networks such as I-RVFL and constructive BP (C-BP), the experimental results show that the generalization ability and stability of the BSCN model are significantly better than them.

The remainder of this paper is organized as follows. In Section 2, we briefly review the training mechanism of SCN, I-RVFL, and C-BP. The details of our proposed BSCN algorithm and its pseudocode are given in Section 3. In Section 4, we introduce the experimental data, parameter settings, and experimental results. In Section 5, we conclude this paper.

2. Preliminaries

In this section, we briefly review the training mechanism of SCN. SCN is a constructive feed-forward neural network with a single hidden layer. Take the SCN model for regression problems as an example, whose network structure is shown in Fig. 1, where w refers to the input weights between the input layer and the hidden layer, b refers to the thresholds of hidden nodes (a.k.a., hidden biases), β refers to the output weights between the hidden layer and the output layer, d is the dimension of the input data, L_{max} is the maximum number of hidden nodes.

Given a training data set $\{X, Y\} \in R^{(d+m)\times N}$, where d is the dimension of the input data, N is the number of samples, and m is the class number for classification problems and the constant

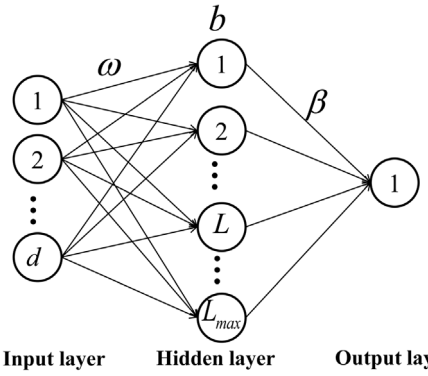

Fig. 1. The network structure of the SCN model.

one for regression problems. The SCN model with L hidden nodes and an activation function $g(\cdot)$ can be represented as

$$f_L(X) = \sum_{i=1}^{L} \beta_i g(w_i \cdot X + b_i) \tag{1}$$

where $w_i = [w_{i1}, w_{i2}, \ldots, w_{id}]$ and $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]$.

Suppose that the target function is $f : R^d \rightarrow R^m$, the current residual error can be represented as

$$e_L = f - f_L = [e_{L1}, e_{L2}, \ldots, e_{Lm}] \tag{2}$$

If the residual error has not reached to an acceptable level, then add a new hidden node, and the current SCN model is

$$f_{L+1}(X) = f_L + \beta_{L+1} g(w_{L+1} \cdot X + b_{L+1}) \tag{3}$$

where w_{L+1}, b_{L+1} and β_{L+1} are the input weights, the hidden bias, and the output weights of the newly added node, respectively.

For SCN, the values of w_{L+1} and b_{L+1} are randomly assigned but required to meet the following criteria (i.e., the so-called supervisory random mechanism)

$$\sum_{j=1}^{m} \langle e_{Lj}, g_{L+1} \rangle^2 \geq \|g_{L+1}\|^2 \delta_{L+1} \tag{4}$$

where

$g_{L+1} = g(w_{L+1} \cdot X + b_{L+1})$,
$\delta_{L+1} = (1 - r - \mu_{L+1}) \|e_L\|^2$,
and $\mu_{L+1} = \frac{1-r}{L+1}$, $0 < r < 1$.

The output weights of the newly added node are calculated by using the following equation:

$$\beta_{L+1,j} = \frac{\langle e_{Lj}, g_{L+1} \rangle}{\|g_{L+1}\|^2}, j = 1, 2, \ldots, m \tag{5}$$

Then update the current residual error: $e_{L+1} = e_L - \beta_{L+1} g(w_{L+1} \cdot X + b_{L+1})$. Repeat (2)–(5) until the prediction error of the model reaches the threshold or the number of hidden nodes reaches the maximum L_{max}.

Strictly speaking, the above training process belongs to the SC-I algorithm in the SCN family, which is characterized by only solving the output weights of the newly added node each time and the output weights of the existing nodes remain unchanged. To further improve the convergence rate of the SC-I, Wang DH et al. proposed the SC-II and SC-III algorithms (Wang & Li, 2017b). The difference between them lies in the way of solving the output weights of the newly added node. Specifically, SC-II updates the output weights of the latest hidden nodes within a given time window each time; while SC-III updates the output weights of all existing hidden nodes each time. The universal approximation property of the SC-I, SC-II, and SC-III models has been theoretically proved in Wang and Li (2017b). Generally, the convergence rate of the SC-III is faster than that of the SC-II and SC-I, and the generalization ability and stability of the SC-III model are also better than the other two algorithms. The BSCN algorithm proposed in this paper is designed based on the SC-III. For simplicity and without loss of generality, the SCN mentioned below refers to SC-III.

I-RVFL (Li & Wang, 2017) is also a constructive randomized neural network. Unlike SCN, I-RVFL uses a completely random way to assign the input parameters for the hidden nodes. Specifically, the input weights and hidden bias of the newly added node in I-RVFL are randomly generated from $[-1, 1]$ according to a uniform distribution, while the corresponding output weights are obtained analytically like SCN. C-BP (Kwok & Yeung, 1997) is a traditional constructive neural network, which uses the gradient descent-based method to calculate the parameters of the model iteratively. The termination conditions of the model training are the same as those of SCN and I-RVFL.

3. Bidirectional stochastic configuration network (BSCN)

In this section, we introduce the details of the proposed BSCN algorithm and present its pseudo-code.

3.1. Algorithm description

Similar to SCN, BSCN employs a constructive randomized neural network, that is, the number of hidden nodes gradually increases until that the model accuracy reaches the expected threshold. To alleviate the low training efficiency of SCN that is mentioned in Section 1, BSCN divides the process of adding hidden nodes into two categories: forward learning and backward learning, depending on the orders of the hidden nodes to be added. Let L represent the current number of hidden nodes, that is to say, the newly added node is the Lth one. Specifically, considering a newly added node, if its order is odd, that is, $L \in \{2n + 1, n \in Z\}$, then its input parameters are generated by the same supervisory mechanism as SCN (referred to as forward learning); otherwise (i.e., $L \in \{2n, n \in Z\}$), its input parameters are calculated according to the residual error feedback of the current model (referred to as backward learning).

Given a training data set $\{X, Y\} \in R^{(d+m)\times N}$, the expected model error ε, the maximum number of hidden nodes L_{max}, the maximum times of random configurations T_{max}, and a BSCN model with $L - 1$ hidden nodes ($L - 1 < L_{max}$) and the activation function $g(\cdot)$. The current residual error is represented as $e_{L-1}(X) = [e_{L-1,1}(X), e_{L-1,2}(X), \ldots, e_{L-1,m}(X)] \in R^{N\times m}$, and suppose it does not satisfy $\|e_{L-1}(X)\| < \varepsilon$. So in this situation, we need to add the Lth hidden node.

If $L \in \{2n + 1, n \in Z\}$, we perform forward learning on the newly added node. In detail, we randomly generate T_{max} pairs of input weights and hidden bias (w_i, b_i), $i = 1, 2, \ldots, T_{max}$ from a symmetric interval $[-\lambda, \lambda]$, $\lambda > 0$ according to the inequality (4) and calculate the corresponding output weights based on (5). Then, we pick out the pair that can reduce the current residual error greatest as the input parameters of the newly added node. After that, we will update the current residual error: $e_L(X) = e_{L-1}(X) - \beta_L g(w_L \cdot X + b_L)$. If $\|e_L(x)\| < \varepsilon$ or $L \geq L_{max}$, we complete the model training; otherwise, we continue to add new hidden nodes.

Otherwise (i.e., $L \in \{2n, n \in Z\}$), we perform backward learning instead. Specifically, we first calculate the input weights and

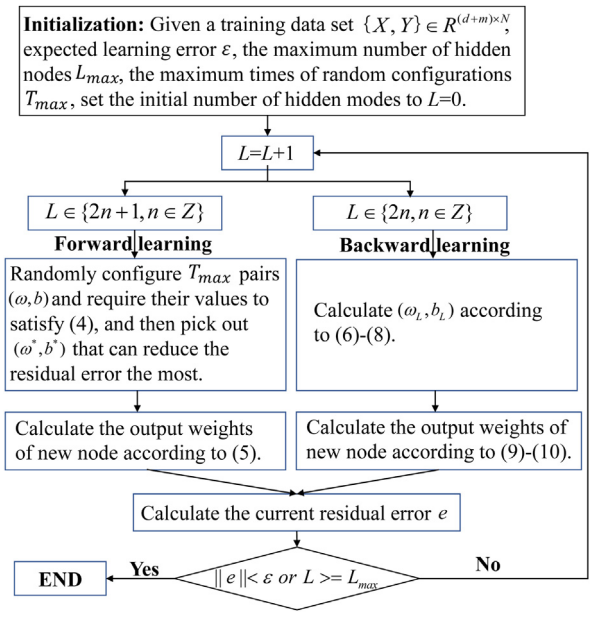

Fig. 2. The learning pipeline of BSCN.

hidden bias of the newly added node according to the following equations:

$$H_L^e = e_{L-1} \cdot (\beta_{L-1})^{-1} \tag{6}$$

$$w_L = g^{-1}(u(H_L^e)) \cdot X^{-1} \tag{7}$$

$$b_L = \sqrt{mse(g^{-1}(u(H_L^e)) - w_L \cdot X^{-1})} \tag{8}$$

where H_L^e is the current error feedback function sequence, $g^{-1}(\cdot)$ represents the reverse function of the activation function, and $u(\cdot)$ is a normalized function $u : R \to (0, 1]$.

Next, we compute the output weights of the new node using the following equations:

$$H_L = u^{-1}(g(w_L \cdot X + b_L)) \tag{9}$$

$$\beta_{L,j} = \frac{\langle e_{L-1,j}, H_L \rangle}{\|H_L\|^2}, j = 1, 2, \ldots, m \tag{10}$$

where $u^{-1}(\cdot)$ is the reverse function of $u(\cdot)$.

Finally, the same as forward learning, we update the residual error and determine the termination of model training.

The above learning process is shown in Fig. 2.

Theorem 1. *Given a constructive randomized neural network with a single hidden layer, and a bounded non-constant piecewise continuous function as its activation function. If the input parameters (w, b) of the odd hidden nodes (i.e., $L \in \{2n+1, n \in Z\}$) are assigned according to (4) and their output weights β are calculated by (5); while the input parameters (w, b) of the even hidden nodes (i.e., $L \in \{2n, n \in Z\}$) are calculated by (6)–(8) and their output weights β are calculated by (9)–(10); then for any continuous target function f, we have*

$$\lim_{n\to+\infty} \|f - (f_{2n-1} + H_{2n} \cdot \beta_{2n} + H_{2n+1} \cdot \beta_{2n+1})\| = 0,$$

where H_{2n} and H_{2n+1} are the outputs of the $(2n)$th and the $(2n+1)$th hidden nodes, respectively.

Proof. Since $e_{2n+1} = e_{2n} - H_{2n+1} \cdot \beta_{2n+1}$, let $\Delta = \|e_{2n}\|^2 - \|e_{2n+1}\|^2$, then we have

$$\begin{aligned}\Delta &= \|e_{2n}\|^2 - \|e_{2n} - H_{2n+1} \cdot \beta_{2n+1}\|^2 \\ &= 2\beta_{2n+1} \langle e_{2n}, H_{2n+1} \rangle - \|H_{2n+1}\|^2 \cdot \beta_{2n+1}^2 \\ &= \|H_{2n+1}\|^2 \left(\frac{2\beta_{2n+1}\langle e_{2n}, H_{2n+1}\rangle}{\|H_{2n+1}\|^2} - \beta_{2n+1}^2\right) \\ &= \|H_{2n+1}\|^2 (2\beta_{2n+1} \cdot \beta_{2n+1} - \beta_{2n+1}^2) \\ &= \|H_{2n+1}\|^2 \cdot \beta_{2n+1}^2 \geq 0\end{aligned} \tag{11}$$

Thus, we have $\|e_{2n}\|^2 \geq \|e_{2n+1}\|^2$. According to Theorem 6 in Wang and Li (2017b), Wang DH et al. have proved that

$$\|e_{2n}\|^2 - (r + \mu_{2n}) \|e_{2n-1}\|^2 \leq 0 \tag{12}$$

where $0 < r < 1$ and $\mu_{2n} \leq 1 - r$.

From (12), we can get $\|e_{2n}\|^2 \leq \|e_{2n-1}\|^2$. Thus, we have $\|e_{2n+1}\|^2 \leq \|e_{2n}\|^2 \leq \|e_{2n-1}\|^2$. So far we have proved that as the number of hidden nodes increases, the residual error of the model will decrease monotonically.

Wang and Li (2017b) have proved that as long as the input parameters (w, b) of hidden nodes are assigned according to (4) and their output weights β are calculated by (5), then $\lim_{L\to+\infty} \|f - f_L\| = 0$. In BSCN, the input parameters and output weights of hidden nodes in the forward learning mode are assigned in this way, and the above proof also shows that the residual error of the model is decreasing monotonically, therefore, it is easy to verify that $\lim_{L\to+\infty} \|e_L\| = 0$ and $\lim_{n\to+\infty} \|f - (f_{2n-1} + H_{2n} \cdot \beta_{2n} + H_{2n+1} \cdot \beta_{2n+1})\| = 0$.

3.2. Pseudo-code for BSCN

The pseudo-code for BSCN is given in algorithm 1.

4. Experimental setting and results

In this section, we evaluate the performance of the proposed BSCN on ten benchmark regression problems from the UCI machine learning repository,[1] two real-world air pollution prediction problems, and a classical problem of age estimation from a single face image. We also compare the performance of BSCN with SCN (Wang & Li, 2017b), I-RVFL (Li & Wang, 2017), and C-BP (Kwok & Yeung, 1997) on these problems.

4.1. The details of experimental setting and benchmarks

The details of the ten regression data sets and the division of the training set and testing set are shown in Table 1. Note that we used ten-fold cross validation scheme to select the optimal model for each algorithm.

In our experiments, we chose the most commonly used root mean square error (RMSE), testing standard deviation (SD), and training time in regression problems as indicators to evaluate the performance of different algorithms. Among them, RMSE and SD can be calculated as follows:

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(t_i - y_i)^2}{N}}, \tag{13}$$

$$SD = \sqrt{\frac{\sum_{j=1}^{S} (e_j - \bar{e})^2}{S - 1}}, \tag{14}$$

where

t_i is the prediction value of the ith instance,

[1] UCI Repository: http://archive.ics.uci.edu/ml/index.php.

Algorithm 1 BSCN algorithm

Input: Given a training data set $\{X, Y\} \in R^{(d+m)\times N}$, the expected learning error ε, the maximum number of hidden nodes L_{max}, the maximum times of random configurations T_{max}, the number of hidden modes L, the activation function $g(\cdot)$, and the residual error e.

Output: The model parameters.

1: **Initialization**: Set $L = 0$ and $e = Y$.
2: **While** $\|e\| > \varepsilon$ & $L < L_{max}$ **do**
3: Increase the number of hidden nodes: $L = L + 1$.
4: **If** $L \in \{2n + 1, n \in Z\}$ **then**
5: Under the premise of satisfying (4), randomly generate T_{max} pairs $(w_i, b_i), i = 1, 2, \cdots, T_{max}$ as candidate input weights and hidden biases for the new hidden node;
6: Select the pair (w^*, b^*) that can reduce the current residual error the most from the above candidates as the (w_L, b_L) of the new hidden node;
7: Calculate the output weights of the new node according to (5).
end if
8: **If** $L \in \{2n, n \in Z\}$ **then**
9: Calculate the (w_L, b_L) of the new node according to (6)–(8);
10: Calculate the output weights of the new node according to (9)–(10);
11: **end if**
12: Update the current residual error: $e = e - \beta_L g(w_L \cdot X + b_L)$.
13: **end while**
Return the parameters of the model, including the input weights and hidden biases of hidden nodes, and the corresponding output weights.

Table 1
Details of regression data sets.

Data-set	No. of training samples	No. of testing samples	No. of attributes
GT_turbine	6000	5934	16
Airfoil Self-noise	750	753	5
GT_compressor	6000	5934	16
Housing	250	256	13
parkinsons_motor	3000	2875	16
Concrete compressive strength	500	530	8
Yacht	150	158	6
White wine quality	2000	2898	11
Solar_C	700	689	10
Red wine quality	800	799	11

y_i is the real value of the ith instance,
N is the number of samples,
S is the number of independent experiments for each case,
e_j is the prediction error of the model in the jth experiment,
and $\bar{e}$ is the average prediction error of S experiments.

Note that RMSE can effectively reflect the prediction ability of the model, the smaller the value, the better the prediction ability of the model; SD can reflect the stability of the model, the smaller the value, the better the stability of the model.

For SCN, I-RVFL, and BSCN, the initial number and the maximum number of hidden nodes (i.e., L and L_{max}) were set to 0 and 30, respectively. For SCN and BSCN, the maximum times of random configurations T_{max} was set to 100, the expected learning error ε was set to 0.001, the boundary value λ of the symmetric interval was set to [0.5, 1, 5, 10, 30, 50, 100, 150, 200, 250] respectively, and the range of the regularization parameter r was set to [0.9, 0.99, 0.999, 0.9999, 0.99999, 0.999999] respectively. For I-RVFL, the input weights and hidden biases of hidden nodes were assigned from $[-1, 1]$ under a uniform distribution. For C-BP, the number of epochs and the learning rate were set to 100 and 0.01, respectively. The Sigmoid function (i.e., $g(w, X, b) = \frac{1}{1+\exp^{-(w\cdot X+b)}}$) was selected as the activation function of all algorithms. All experiments were conducted under Windows 10 X64 OS with Intel Core i5-5300U CPU 2.29 GHz, and our simulation software is MATLAB with the R2014a version.

4.2. Experimental results and analysis on the benchmarks

Each experiment was independently conducted twenty times, and the experimental results we show here are the average of the results corresponding to these twenty independent experiments. The training time, RMSE, and SD of SCN, I-RVFL, C-BP, and BSCN on these ten regression problems are shown in Table 2. Note that the best results are in bold.

It can be observed from Table 2 that the testing RMSE of the BSCN model is smaller than that of the SCN, I-RVFL, and C-BP models on all the data sets, which means that the BSCN model has better generalization ability than the SCN, I-RVFL, and C-BP models. Moreover, one can observe that the testing standard deviation of the BSCN model is always smaller than that of the SCN, I-RVFL, and C-BP models, which implies that the BSCN model has better stability than them.

In terms of training time, one can observe that the training times of the randomized algorithms (i.e., SCN, I-RVFL, and BSCN) are much shorter than that of the C-BP on all the data sets, which verifies that the non-iterative training mechanism adopted by the randomized algorithms has higher training efficiency than that of the gradient descent-based training mechanism adopted by the C-BP algorithm.

Moreover, the training times of the BSCN on some data sets such as GT_turbine and GT_compressor are shorter than that of the SCN, I-RVFL, and C-BP; but on other data sets such as Airfoil self-noise, the I-RVFL is the fastest one. For this phenomenon, here we give an empirical explanation: when modeling the GT_turbine and GT_compressor problems, BSCN and SCN complete the training process before the number of hidden nodes grows to the preset maximum, that is, when the prediction error of the model is lower than the expected error, L is still smaller than L_{max}; but for I-RVFL, it cannot terminate the training process until the number of hidden nodes reaches L_{max}. As for C-BP, although it can also complete model training before the number of hidden nodes increases to L_{max}, its training speed is slower because it uses the iterative training mechanism. Therefore, the training times of the I-RVFL and C-BP models on these two problems are longer than that of the BSCN and SCN models. For other data sets, none of the three constructive algorithms can make their model errors lower than the threshold before the number of hidden nodes reaches L_{max}, so they all must increase L_{max} hidden nodes to terminate their training process. When adding hidden nodes, I-RVFL uses a one-time completely random method to assign input parameters for the new nodes, while SCN and BSCN both generate the input parameters through a supervisory mechanism, which is more time-consuming than the former method, so the training times of the SCN and BSCN are longer than that of the I-RVFL. For C-BP, the iterative training mechanism makes its training speed much slower than the randomized algorithms using the non-iterative training mechanism.

To compare the training efficiency of BSCN and SCN more intuitively, we visually represent their model training times on the above regression problems in Fig. 3:

It can be observed from Fig. 3 that the training time of BSCN is much shorter than that of SCN on all problems, which means that the training efficiency of BSCN is significantly higher than that

Table 2
The training time, standard deviation, training RMSE, and testing RMSE of the SCN, I-RVFL, C-BP, and BSCN on the benchmark data sets.

Data-set	Algorithm	Training time (s)	Standard deviation	Training RMSE	Testing RMSE
GT_turbine	C-BP	2.6586	0.0003	0.0007	0.0009
	SCN	0.2461	0.0003	0.0005	0.0005
	I-RVFL	1.0930	0.0733	0.0718	0.0735
	BSCN	**0.1148**	**0.0002**	**0.0003**	**0.0003**
Airfoil self-noise	C-BP	14.3766	0.0597	0.2224	0.2379
	SCN	6.2906	0.1102	0.0517	0.1463
	I-RVFL	**0.8938**	0.0305	0.2923	0.2769
	BSCN	3.4023	**0.0039**	**0.0458**	**0.0652**
GT_compressor	C-BP	2.5883	0.0003	0.0004	0.0005
	SCN	0.2359	0.0002	0.0006	0.0006
	I-RVFL	1.0852	0.0511	0.0764	0.0766
	BSCN	**0.1148**	**0.0002**	**0.0003**	**0.0003**
Housing	C-BP	12.3391	0.0930	0.1699	0.1716
	SCN	2.6234	0.0020	0.0175	0.0208
	I-RVFL	**0.8672**	0.0423	0.0884	0.0904
	BSCN	1.7852	**0.0016**	**0.0159**	**0.0181**
Parkinsons motor	C-BP	17.0148	0.1404	0.4601	0.4616
	SCN	8.1742	0.4136	**0.2918**	0.5050
	I-RVFL	**0.9203**	0.2272	0.4161	0.4287
	BSCN	4.1547	**0.1127**	0.2956	**0.3493**
Concrete compressive strength	C-BP	12.4609	0.0797	0.1424	0.1430
	SCN	2.6570	0.0034	**0.0141**	0.0219
	I-RVFL	**0.8945**	0.0396	0.0893	0.0898
	BSCN	1.7453	**0.0021**	0.0149	**0.0190**
Yacht	C-BP	11.4938	0.0581	0.4471	0.4975
	SCN	3.0070	0.0199	0.1051	0.1420
	I-RVFL	**0.3531**	0.0258	0.3910	0.5028
	BSCN	1.8125	**0.0104**	**0.0981**	**0.1281**
White wine quality	C-BP	15.5469	0.0247	0.0648	0.0657
	SCN	5.3578	0.0739	**0.0123**	0.0878
	I-RVFL	**0.9406**	0.0443	0.0699	0.0724
	BSCN	3.0563	**0.0230**	0.0124	**0.0321**
Solar_C	C-BP	13.7445	0.0603	0.4989	0.5276
	SCN	1.7078	0.0053	**0.3215**	0.3574
	I-RVFL	**0.8008**	0.0118	0.3897	0.4388
	BSCN	1.2773	**0.0049**	0.3274	**0.3566**
Red wine quality	C-BP	13.1758	0.0902	0.2267	0.2322
	SCN	2.7242	0.1546	**0.0533**	0.1504
	I-RVFL	**0.8563**	0.0756	0.1374	0.1461
	BSCN	1.7805	**0.0721**	0.0546	**0.0849**

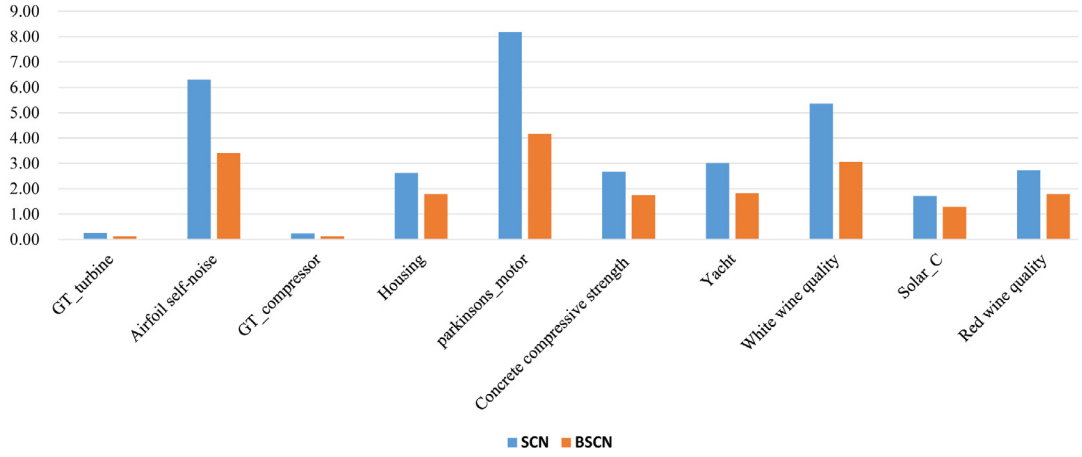

Fig. 3. The training time (s) comparison of BSCN and SCN.

of SCN. This experimental phenomenon implies that using the bidirectional learning mechanism to add hidden nodes is effective and efficient.

In addition, we recorded the learning error changes of the SCN, I-RVFL, C-BP, and BSCN models during their training process. The experimental results show that the trends of the learning errors of these models are consistent on the above regression problems. Therefore, here we take the Airfoil self-noise data set as an example to analyze the corresponding experimental phenomenon (as shown in Fig. 4).

It can be observed from Fig. 4 that as the number of hidden nodes increases, the learning errors of the SCN, I-RVFL, C-BP, and

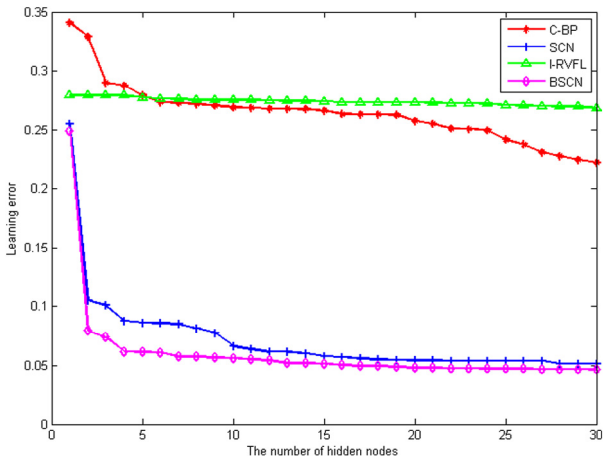

Fig. 4. The learning error changes of the SCN, I-RVFL, C-BP, and BSCN models.

BSCN models all become smaller, but the error reduction speed of the BSCN and SCN models is significantly faster than that of the I-RVFL and C-BP models. This phenomenon implies that the quality of newly added nodes generated by the BSCN and SCN is much better than that of the I-RVFL and C-BP. Furthermore, one can observe that the error reduction speed of the BSCN is faster than that of the SCN, which verifies the efficiency of the BSCN again.

Moreover, one can observe that when the first hidden node is added, the error gap between the BSCN and SCN models is not obvious, this is because they use the same method to generate the parameters of the first hidden node. After the second hidden node is added, there is a clear gap between the errors of the two models. Specifically, the learning error of the BSCN model is significantly lower than that of the SCN model. This phenomenon implies that the quality of the second hidden node generated by the BSCN is higher than that of the SCN. As the number of hidden nodes increases, the error gap between the BSCN model and the SCN model begins to slow down, which indicates that the two models are gradually close to the learning error boundary of the problem.

From the experimental phenomenon in Fig. 4, one can infer that BSCN can achieve a faster convergence rate than SCN, I-RVFL, and C-BP. Given the same expected error, BSCN can approach the threshold faster than the others, so the network structure of the BSCN model is expected to be more compact.

4.3. Application of BSCN in real-life air pollution prediction

Particulate matter 2.5 (PM 2.5) and particulate matter 10 (PM10) consist of airborne particles with aerodynamic diameters of less than 2.5 μm and 10 μm, respectively. The concentration of PM 2.5 and PM 10 in the air is an important part of the Air Quality Index (AQI) and is currently the most important reference index used by many countries to evaluate their air pollution status. Accurately predicting the concentration of PM 2.5 and PM 10 in the air is crucial for governments to issue air pollution warnings in time. In this section, we evaluate the performance of BSCN on PM 2.5 and PM 10 concentration prediction problems.

We collected the air quality monitoring data published by Beijing and Oslo governments: Beijing PM2.5 (Liang, et al., 2015) and Oslo PM10 data sets.[2] For the Beijing PM2.5 data set, there are 43824 samples with twelve attributes, and eleven of which

[2] Oslo PM10: http://lib.stat.cmu.edu/datasets/.

Table 3
The details of air pollution data sets.

Data-set	Training data	Testing data	Attributes
Beijing PM 2.5	21757	20000	11
Oslo PM 10	250	250	7

are key factors affecting PM2.5 concentration in the air, such as temperature, pressure, dew point, and accumulated hours of rain. The remaining dimension is row number, which has nothing to do with PM2.5 concentration, so we deleted this column in the experiment. Because the original data set contains some missing values and symbolic values, we first preprocessed the data set. Specifically, we deleted the samples containing missing values and replaced the symbolic values with Arabic numerals. For example, for the attribute "Combined wind direction", which contains four symbolic values: CV, NE, NW, and SE, we replaced them with 1, 2, 3, and 4, respectively. The Oslo PM10 data set was originally collected by the Norwegian Public Roads Administration, which includes 500 samples and each sample contains seven attribute values, such as the logarithm of the number of cars per hour, wind speed, and temperature. These attributes are important factors that affect the concentration of PM10 in the air. We also used the above method to perform a simple preprocessing on this data set.

After data preprocessing, we divided them into training set and testing set in the manner shown in Table 3.

The environment of this experiment and the parameter settings of BSCN, SCN, I-RVFL, and C-BP algorithms are the same as those in the above benchmark experiments. We evaluated the performance of these three algorithms on Beijing PM2.5 and Oslo PM10 data sets. The details of the experimental results are shown in Table 4.

From Table 4, one can observe that the testing RMSE and testing standard deviation of the BSCN model are smaller than those of the SCN, I-RVFL, and C-BP models, which means that the BSCN model can predict the concentration of PM2.5 and PM10 more accurately and stably. At the same time, it can be seen from Table 4 that the training time of BSCN is also significantly shorter than that of SCN, which implies that the training efficiency of BSCN is higher than that of SCN.

Moreover, one can observe that I-RVFL can achieve the least training time on these two problems. The reason for this phenomenon is the same as the explanation in the benchmark experiments. That is, for the above air pollution prediction problems, these three algorithms can only terminate their training process by increasing the number of hidden nodes to L_{max}. In the process of adding hidden nodes, I-RVFL uses a one-time completely random method to assign input parameters for the new nodes, while SCN and BSCN both generate the input parameters through a supervisory mechanism, which is more time-consuming than the former method, so the training times of the SCN and BSCN are longer than that of the I-RVFL. As for C-BP, the iterative training mechanism causes its training speed to be much slower than these randomized algorithms. Moreover, since the input parameters of new nodes in the backward learning mode of BSCN are calculated based on the residual error feedback of the model, the training efficiency of this method is much higher than the supervisory mechanism, so the training time of the BSCN model is much less than that of the SCN model.

4.4. Application of BSCN in the age estimation problem

In this section, we evaluate the performance of BSCN on the age estimation from a single face image without the use of facial landmarks. Different from the existing works such as

Table 4
The training time, standard deviation, training RMSE, and testing RMSE of the SCN, I-RVFL, C-BP, and BSCN on the air pollution data sets.

Data-set	Algorithm	Training time (s)	Standard deviation	Training RMSE	Testing RMSE
Beijing PM 2.5	C-BP	54.3820	0.0421	0.1904	0.1916
	SCN	37.7930	0.0005	0.0746	0.0761
	I-RVFL	**1.9711**	0.0024	0.0891	0.0893
	BSCN	21.0047	**0.0003**	**0.0742**	**0.0755**
Oslo PM 10	C-BP	13.2422	0.1049	0.1613	0.1629
	SCN	2.8063	0.0017	0.0081	0.0125
	I-RVFL	**0.7383**	0.0433	0.0999	0.1030
	BSCN	1.8055	**0.0010**	**0.0079**	**0.0114**

Fig. 5. Examples of the MORPH2 data set (Ricanek & Tesafaye, 2006).

Rothe, Timofte, and Van Gool (2018), which poses the age estimation problem as a classification problem, we regard this problem as a regression problem. We chose MORPH2 (Ricanek & Tesafaye, 2006) as the experimental data set, which is one of the most popular data sets in age estimation studies. MORPH2 contains a large number of images from diverse subjects over many years. Taking Fig. 5 as an example, it shows a group of facial images of a man at his 45, 46, and 47 years old, respectively. In this study, we expect that the model can accurately predict the corresponding age based on the input facial image.

Data preprocessing: we first scaled the RGB facial images in MORPH2 to a size of 224 * 224 uniformly, and then randomly selected 10000, 20000, and 30000 samples from them to form the corresponding three data sets. We divided each data set into the corresponding training set and testing set according to 8: 2.

In this experiment, we used the ResNet-101 model (He, Zhang, Ren, & Sun, 2016) pre-trained on MORPH2 as the feature extractor, and SCN, I-RVFL, C-BP, and BSCN as learners to mine the mapping relationship between visual features and age labels. The maximum of the hidden nodes in these learners was set to 300. The setting of other parameters is the same as the benchmark experiment. The experimental results on the above three data sets are shown in Table 5.

It can be observed from Table 5 that our BSCN model surpasses other algorithms in terms of prediction accuracy and stability, which means that giving BSCN the same data features as other algorithms, it can better mine the intrinsic relationship between these features and the corresponding labels. Moreover, one can observe that the model training time of randomized neural networks (i.e., BSCN, SCN, and I-RVFL) is significantly shorter than that of traditional neural networks (i.e., C-BP).

This experimental phenomenon inspires us that if a high-quality pre-trained model can be obtained in advance as the feature extractor of the model, it may be a promising approach to choose a randomized neural network, especially BSCN, to replace the traditional fully connected network as the final decision layers. This method is expected to not only exert the feature extraction ability of the deep model, but also the fast learning ability and good generalization ability of the randomized algorithms.

Remark 1. In the backward learning mode of BSCN, it can find the most suitable parameters for the newly added nodes according to the residual errors. Further, the improvement of the quality of hidden nodes makes the generalization ability and stability of the BSCN model outperform the SCN and I-RVFL models. Moreover, this method is done in one go, so it can achieve much higher training efficiency than the supervisory random search method used in the existing SCN algorithms.

Therefore, one can briefly summarize the advantages of BSCN: it possesses the universal approximation ability in theory and is superior to other randomized algorithms such as SCN and I-RVFL in generalization and stability. Moreover, the training mechanism of BSCN can achieve much higher learning efficiency than the original SCN both theoretically and practically. However, one of the disadvantages of BSCN is its relatively shallow network architecture, which leads to many difficulties in dealing with large data sets like ImageNet. In the future, one may consider combining traditional deep learning architectures such as ResNet to design deep BSCN algorithms to alleviate this issue.

Remark 2. Randomized neural networks vs traditional neural networks. The former assumes that not all parameters need to be fine-tuned, so these algorithms randomly assign some parameters (e.g., the input weights and hidden biases) and keep them unchanged throughout the model training process, and the remaining parameters are obtained analytically (e.g., least square method). This training mechanism has been empirically proved in many cases that it can make the model have extremely fast training speed and good generalization ability. However, these successful cases have a common feature, that is, the complexity of the data feature is not very high. For this phenomenon, our speculative explanation is that in these cases, the original data can be linearly separated in a high-dimensional space with a high probability after the non-linear random feature mapping, and then using analytical methods to calculate the output weights can make up for the instability caused by random parameters and guarantee the generalization ability of the model. In fact, some researchers have explained the rationality of this phenomenon from the theory of learning with similarity functions (Balcan, Blum, & Srebro, 2008). The disadvantage of randomized neural networks is that the complexity of the model is relatively low and it is difficult to use them directly to deal with complex problems (e.g., natural language processing). Moreover, the theoretical analysis of the rationality of random feature mapping is still an open problem.

For traditional neural networks such as convolution neural networks (Wu, Li, Lin, & Pirouz, 2019) and recurrent neural networks (Chou, Huang, Huang, & Tseng, 2019), all parameters in the network need to be fine-tuned. These neural networks usually fine-tune their parameters iteratively by the gradient descent technique according to the residual error of the model until the model converges. This training method is more time-consuming

Table 5
The training time, standard deviation, training RMSE, and testing RMSE of the SCN, I-RVFL, C-BP, and BSCN on the age estimation data sets.

Data-set	Algorithm	Training time (s)	Standard deviation	Training RMSE	Testing RMSE
MORPH2-10000	C-BP	2646.3323	49.9809	134.3410	134.9013
	SCN	141.1305	0.1359	3.1156	4.1095
	I-RVFL	**56.5266**	2.9360	12.3566	12.5773
	BSCN	652.8563	**0.0680**	**2.6903**	**3.6295**
MORPH2-20000	C-BP	4915.4995	75.1322	179.4321	179.6134
	SCN	263.1422	0.1251	3.2275	3.6646
	I-RVFL	**102.4727**	2.9913	12.8385	12.8940
	BSCN	1141.2523	**0.0446**	**3.0008**	**3.4875**
MORPH2-30000	C-BP	6964.9189	105.3616	154.9312	155.0126
	SCN	389.3328	0.7392	3.2663	3.6514
	I-RVFL	**152.0188**	3.5044	11.8227	12.1760
	BSCN	1805.6484	**0.0155**	**3.0964**	**3.5287**

than the non-iterative training mechanism adopted by the randomized neural networks and involves many hyper-parameters, whose values are usually not easy to determine. Therefore, when the data features of modeling problems are relatively simple, the performance of traditional neural networks is sometimes inferior to that of randomized neural networks. However, as the complexity of modeling problems increases, traditional neural networks can increase their model complexity by freely increasing the number of hidden layers and nodes, which can better handle the complex problems, while randomized neural networks are difficult to do this. How to better combine the advantages of these two types of neural networks is a problem worthy of study.

Remark 3. Since the 1990s, the bidirectional training mechanism of neural networks has attracted the attention of researchers (Gedeon, 1998; Schuster & Paliwal, 1997), and it has become a hot spot in recent years. Relevant representative algorithms include dynamic Boltzmann machine with bidirectional training scheme (Osogami, Kajino, & Sekiyama, 2017), bidirectional feature pyramid network (BFPN) (Zhu, Deng, Hu, Fu, Xu, Qin, et al., 2018), gated bidirectional network (Sun, Li, Zheng, & Lu, 2019), etc. The common point of these algorithms is to improve the learner's ability to perceive and fuse the spatial and temporal information of data features by constructing information transmission channels between different layers (or modules). Different from these algorithms, the bidirectional learning mechanism used in this study refers to the generation of hidden node parameters, which is divided into the forward learning and the backward learning. The details of the forward learning and the backward learning have been given in Section 3 and will not be repeated here.

5. Conclusions

To improve the training efficiency of SCN, this paper optimized the constructive process of its hidden nodes and proposed the bidirectional SCN (BSCN), which uses two learning modes (i.e., the forward learning and the backward learning) to add the hidden nodes. Specifically, the forward learning uses the same supervisory mechanism as SCN to assign the input weights and hidden biases for the odd nodes, which can guarantee the universal approximation ability of the model; while the backward learning calculates the above parameters for the even nodes according to the residual error feedback of the current model, which is able to greatly speed up the model training.

We have theoretically proved the convergence of BSCN and experimentally demonstrated that the training efficiency, generalization ability, and stability of BSCN are much higher than SCN on ten benchmark regression problems. Compared with I-RVFL and C-BP, BSCN can also achieve much better generalization ability and stability. Moreover, we verified the effectiveness of BSCN on two real-world air pollution prediction problems. The experimental results show that the BSCN model can achieve higher accuracy and stability than the SCN, I-RVFL and C-BP models in predicting the concentration of PM2.5 and PM10 in the air, which provides a new solution for real-life air quality assessment problems. We also evaluated the prediction ability of BSCN and other algorithms in recognizing the age of a person from a single face image. The experimental results confirm the above conclusion again, that is, BSCN exceeds SCN in both prediction accuracy and training efficiency.

BSCN provides a stable and fast modeling solution for platforms with limited computing resources, such as various IoT terminals, and is expected to be widely deployed in various industrial scenarios. In the future, we will expand BSCN into a multi-hidden layer architecture to enhance its feature extraction capability and apply it to solve more real-life problems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Balcan, M. F., Blum, A., & Srebro, N. (2008). A theory of learning with similarity functions. *Machine Learning*, *72*(1–2), 89–112.

Bi, J., Yuan, H., Zhang, L., & Zhang, J. (2019). SGW-SCN: An integrated machine learning approach for workload forecasting in geo-distributed cloud data centers. *Information Sciences*, *481*, 57–68.

Chou, C. H., Huang, Y., Huang, C. Y., & Tseng, V. S. (2019). Long-term traffic time prediction using deep learning with integration of weather effect. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 123–135). Springer.

Dai, W., Li, D., Zhou, P., & Chai, T. (2019). Stochastic configuration networks with block increments for data modeling in process industries. *Information Sciences*, *484*, 367–386.

Ertuğrul, Ö. F. (2018). Two novel versions of randomized feed forward artificial neural networks: stochastic and pruned stochastic. *Neural Processing Letters*, *48*(1), 481–516.

Gedeon, T. (1998). Stochastic bidirectional training. In *SMC'98 conference proceedings. 1998 IEEE international conference on systems, man, and cybernetics (Cat. No. 98CH36218) (vol. 2)* (pp. 1968–1971). IEEE.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Huang, C., Huang, Q., & Wang, D. (2019). Stochastic configuration networks based adaptive storage replica management for power big data processing. *IEEE Transactions on Industrial Informatics*, *16*(1), 373–383.

Kwok, T. Y., & Yeung, D. Y. (1997). Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Transactions on Neural Networks*, *8*(3), 630–645.

Li, M., & Wang, D. (2017). Insights into randomized algorithms for neural networks: Practical issues and common pitfalls. *Information Sciences*, *382*, 170–178.

Li, M., & Wang, D. (2019). 2-D stochastic configuration networks for image data analytics. *IEEE Transactions on Cybernetics*, 1–14.

Liang, X., Zou, T., Guo, B., Li, S., Zhang, H., Zhang, S., et al. (2015). Assessing Beijing's PM2. 5 pollution: severity, weather impact, APEC and winter heating. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *471*(2182), Article 20150257.

Lu, J., & Ding, J. (2019a). Construction of prediction intervals for carbon residual of crude oil based on deep stochastic configuration networks. *Information Sciences*, *486*, 119–132.

Lu, J., & Ding, J. (2019b). Mixed-distribution-based robust stochastic configuration networks for prediction interval construction. *IEEE Transactions on Industrial Informatics*, *16*(8), 5099–5109.

Lu, J., Ding, J., Dai, X., & Chai, T. (2020). Ensemble stochastic configuration networks for estimating prediction intervals: A simultaneous robust training algorithm and its application. *IEEE Transactions on Neural Networks and Learning Systems*.

Osogami, T., Kajino, H., & Sekiyama, T. (2017). Bidirectional learning for time-series models with hidden units. In *International conference on machine learning* (pp. 2711–2720). PMLR.

Pao, Y. H., & Takefuji, Y. (1992). Functional-link net computing: theory, system architecture, and functionalities. *Computer*, *25*(5), 76–79.

Pratama, M., & Wang, D. (2019). Deep stacked stochastic configuration networks for lifelong learning of non-stationary data streams. *Information Sciences*, *495*, 150–174.

Ricanek, K., & Tesafaye, T. (2006). Morph: A longitudinal image database of normal adult age-progression. In *7th International conference on automatic face and gesture recognition* (pp. 341–345). IEEE.

Rothe, R., Timofte, R., & Van Gool, L. (2018). Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, *126*(2–4), 144–157.

Rumelhart, D. E., Durbin, R., Golden, R., & Chauvin, Y. (1995). Backpropagation: The basic theory. In *Backpropagation: Theory, architectures and applications* (pp. 1–34).

Schmidt, W. F., Kraaijveld, M. A., & Duin, R. P. W. (1992). Feedforward neural networks with random weights. In *Proceedings., 11th IAPR international conference on pattern recognition. vol. II. conference B: pattern recognition methodology and systems* (pp. 1–4).

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Sun, H., Li, S., Zheng, X., & Lu, X. (2019). Remote sensing scene classification by gated bidirectional network. *IEEE Transactions on Geoscience and Remote Sensing*, *58*(1), 82–96.

Wang, D., & Cui, C. (2017). Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics. *Information Sciences*, *417*, 55–71.

Wang, D. H., & Li, M. (2017a). Robust stochastic configuration networks with kernel density estimation for uncertain data regression. *Information Sciences*, *412*, 210–222.

Wang, D., & Li, M. (2017b). Stochastic configuration networks: Fundamentals and algorithms. *IEEE Transactions on Cybernetics*, *47*(10), 3466–3479.

Wang, D., & Li, M. (2018). Deep stochastic configuration networks with universal approximation property. In *2018 International joint conference on neural networks* (pp. 1–8). IEEE.

Wang, W., & Wang, D. (2020). Prediction of component concentrations in sodium aluminate liquor using stochastic configuration networks. *Neural Computing and Applications*, 1–14.

Wu, J. M. T., Li, Z., Lin, J. C. W., & Pirouz, M. (2019). A new convolution neural network model for stock price prediction. In *International conference on genetic and evolutionary computing* (pp. 581–585). Springer.

Xu, X., Zhang, X., Gao, H., Xue, Y., Qi, L., & Dou, W. (2020). BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing. *IEEE Transactions on Industrial Informatics*, *16*(6), 4187–4195.

Zhang, Q., Li, W., Li, H., & Wang, J. (2020). Self-blast state detection of glass insulators based on stochastic configuration networks and a feedback transfer learning mechanism. *Information Sciences*, *522*, 259–274. http://dx.doi.org/10.1016/j.ins.2020.02.058.

Zhu, L., Deng, Z., Hu, X., Fu, C. W., Xu, X., Qin, J., et al. (2018). Bidirectional feature pyramid network with recurrent attention residual modules for shadow detection. In *Proceedings of the European conference on computer vision* (pp. 121–136).