

Article

Forecasting Network Interface Flow Using a Broad Learning System Based on the Sparrow Search Algorithm

Xiaoyu Li ¹, Shaobo Li ^{2,*} , Peng Zhou ³ and Guanglin Chen ³

¹ College of Computer Science and Technology, Guizhou University, Guiyang 550025, China; gs.xiaoyuli20@gzu.edu.cn

² State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China

³ College of Mechanical Engineering, Guizhou University, Guiyang 550025, China; pzhou@gzu.edu.cn (P.Z.); gs.glchen21@gzu.edu.cn (G.C.)

* Correspondence: lishaobo@gzu.edu.cn

Abstract: In this paper, we propose a broad learning system based on the sparrow search algorithm. Firstly, in order to avoid the complicated manual parameter tuning process and obtain the best combination of hyperparameters, the sparrow search algorithm is used to optimize the shrinkage coefficient (r) and regularization coefficient (λ) in the broad learning system to improve the prediction accuracy of the model. Second, using the broad learning system to build a network interface flow forecasting model. The flow values in the time period $[T - 11, T]$ are used as the characteristic values of the traffic at the moment $T + 1$. The hyperparameters outputted in the previous step are fed into the network to train the broad learning system network traffic prediction model. Finally, to verify the model performance, this paper trains the prediction model on two public network flow datasets and real traffic data of an enterprise cloud platform switch interface and compares the proposed model with the broad learning system, long short-term memory, and other methods. The experiments show that the prediction accuracy of this method is higher than other methods, and the moving average reaches 97%, 98%, and 99% on each dataset, respectively.

Keywords: hyperparameter optimization; network traffic; prediction



Citation: Li, X.; Li, S.; Zhou, P.; Chen, G. Forecasting Network Interface Flow Using a Broad Learning System Based on the Sparrow Search Algorithm. *Entropy* **2022**, *24*, 478. <https://doi.org/10.3390/e24040478>

Academic Editors: Jaroslaw Krzywanski, Yunfei Gao, Marcin Sosnowski, Karolina Grabowska, Dorian Skrobek, Ghulam Moeen Uddin, Anna Kulakowska, Anna Zylka and Bachil El Fil

Received: 27 February 2022

Accepted: 26 March 2022

Published: 29 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The number of cloud platform users has increased in tandem with the development of internet technologies. In the context of high concurrency and limited cloud platform resources, how to reasonably allocate resources is one of the problems studied by cloud platform managers [1]. Forecasting the traffic of the cloud platform network interfaces is an effective way to achieve reasonable resource allocation: by predicting the traffic of each interface in the future, judging its resource demand, and accordingly allocating resources and planning the network to achieve a dynamic allocation of resources with the number of requests and achieve load balancing [2]. However, with the rapid increase of cloud platform access, scholars extracting internet traffic features for network traffic modeling and prediction not only have to consider its complex characteristics such as nonlinearity and multi-scale but also face the problems of decreasing prediction accuracy and increasing resource consumption caused by the increasing data scale. Therefore, the research of high-speed, high-efficiency, and high-precision network traffic prediction methods can not only further optimize and improve network resource provisioning, planning, and network security but also be extremely significant for the development of the internet and its good integration with other industries.

Network traffic forecasting belongs to the field of time series forecasting. The types of flow prediction methods include traditional statistical analysis [3,4] and machine learning. Traditional statistical analysis uses statistical and mathematical methods to make speculations and estimates on the development trend of internet traffic in the future period, and

common models include the autoregressive integrated moving average (ARIMA) model [5] and the generalized autoregressive conditional heteroskedasticity (GARCH) model [6,7]. The method of machine learning can be divided into deep learning and classical machine learning methods such as support vector machines (SVM) [8,9]. In recent years, deep neural networks have been widely used in several fields due to their good feature extraction ability [10–12]. Meanwhile, it has also become a common method in network traffic prediction. Miguel [13] used artificial neural network swarms to predict long-term internet flow. They presented four network flow prediction models that are based on the ensemble of time-lagged feedforward networks (TLFNS), demonstrating the superiority of the proposed models by comparing them to the classical Holt–Winters approach. Nie [14] fused deep belief networks and Gaussian models, extracted the low-pass components of network traffic that can describe its own long-distance dependence using discrete wavelet transform and then learned deep belief networks from the low-pass components to build prediction models. Fang [15] used graph convolutional neural networks and long short-term memory (LSTM) to capture the temporal and spatial aspects of the cellular network of a single cell and build a prediction model, respectively. Zhang [16] proposed a spatio-temporal graph convolutional gated recurrent unit (GC-GRU) model to capture the spatial features of network traffic using graph convolutional neural network (GCN) and further process the spatio-temporal characteristics features using gated units (GRU) to improve the prediction performance of network traffic.

Deep models usually have complex structures and large parameters, so these models require repeated iterations to train the network, spending long training time and computational resources. Chen [17] proposed a new shallow neural network based on random vector function-link neural networks (RVFLNN) [18–20], named as broad learning system (BLS). It contains only one hidden layer, consisting of feature mapping nodes and enhancement nodes, which reduces the complexity of the neural network and has some feature capturing ability. Meanwhile, Chen proposed an incremental learning algorithm to calculate the output weights of the newly added hidden layer nodes in the BLS, so it can complete the training of the model in a shorter time while obtaining better accuracy [17,21,22]. Since its proposal, BLS has received a lot of attention and gained rapid development [23,24]. The experiment of Chen using BLS to predict short-term wind power has demonstrated it has good performance in time-series prediction [25]. However, as a kind of neural network, the hyperparameters of the BLS have a large impact on the network accuracy, and researchers usually need to train the model repeatedly to adjust the network hyperparameters to improve the model precision. This manual tuning method not only consumes a lot of time and energy but also wastes resources such as electricity for repeated training. Therefore, automatic hyperparameter optimization methods represented by population intelligence optimization algorithms such as ant colony algorithms [26] and particle swarm algorithms [27] have been developed as a result. In recent years, there have been numerous studies on the optimization of hyperparameters using population intelligence optimization algorithms. Zhou [28] improved the gray wolf algorithm and used this optimization algorithm to optimize hyperparameters such as kernel parameters in support vector machines. Xu [29] used the whale optimization algorithm to optimize the learning rate, training time, and the number of nodes in two hidden layers of the BiLSTM_Attention model to maximize the performance of the model. The ant colony algorithm and particle swarm algorithm have problems such as being easy to fall into local optimum and unsuitable for convergence. Sparrow search algorithm (SSA) [30] is a new type of swarm intelligence optimization algorithm with the advantages of good stability, strong global search ability, and fast convergence, so it has attracted extensive attention and research from scholars at home and abroad [31,32]. Tian [33] used SSA to optimize the hyperparameters of LSTM networks. Gai [34] used SSA to compute the best learning rate and batch size of deep confidence networks. Song [35] used SSA to optimize the penalty parameters and kernel function parameters of least squares support vector machines to improve the prediction accuracy and generalization ability of LSSVM.

In order to establish a fast and accurate network traffic prediction model, BLS is applied to network traffic prediction in this paper. At the same time, in order to quickly select the optimal hyperparameters to reduce their influence on the accuracy of the BLS, this paper combines the SSA with BLS, uses SSA to filter out the optimal combination of hyperparameters, and then uses the optimal hyperparameters to train BLS to build the network traffic prediction model.

The remaining part of this paper is organized as follows. Section 2 introduces the relevant methods used, including the BLS and the SSA. Section 3 introduces the proposed broad learning system based on sparrow search algorithm (SSA-BLS). Section 4 presents our experiments: SSA-BLS model is trained using two public datasets and real traffic data collected from the switch interface of an enterprise cloud platform, and its performance is compared with other models to verify the performance of the model. Section 5 summarizes our work, presents the limitations of the current approach, and briefly describes future work.

2. Related Work

2.1. Broad Learning System (BLS)

Broad learning system (BLS) is a new kind of shallow neural network based on the random vector functional-link neural network that is primarily used to tackle the problems of large computation and long training time for deep learning [17]. As shown in Figure 1, the hidden layer of the BLS is a single-layer structure, consisting of a feature mapping layer and an enhancement node layer.

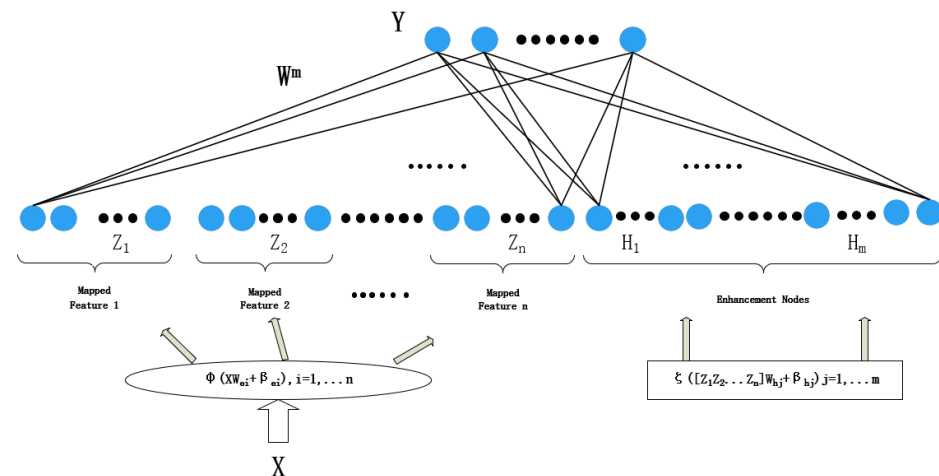


Figure 1. Broad learning system structure.

The input training data $X \in R^{M \times N}$ of BLS contains M samples, each with N dimensions, and the corresponding label is $Y \in R^{M \times C}$. The feature mapping layer maps the input data into n sets of feature mappings with K_i nodes by the feature mapping functions φ_i ($i = 1, \dots, n$):

$$Z_i = \varphi_i(XW_{ei} + \beta_{ei}), i = 1, 2, \dots, n \quad (1)$$

where Z_i means i -th group feature mapping; W_{ei} and β_{ei} is the randomly generated optimal feature mapping weight matrix and bias matrix, determined by the sparse self-encoder. In practical applications, φ_i is often a nonlinear mapping function, such as Relu, Tanh. The groups of feature nodes are spliced to obtain the feature node matrix $Z_{in} = [Z_1Z_2 \dots Z_n]$. After that, the enhancement nodes are generated by the following equations:

$$E_j = Z_{in}W_{hj} + \beta_{hj} \quad (2)$$

$$H_j = \zeta_j(s \cdot E_j / \max E_j), j = 1, 2, \dots, m \quad (3)$$

The enhancement layer contains m groups and each consists of q nodes; H_j denotes the j -th group of enhancement nodes; W_{hj} and β_{hj} are random weights and biases. ζ_j is a nonlinear activation function. Each group in the enhancement layer can choose a different ζ_j to fully extract the feature. In addition, s is the hyperparameter shrinkage coefficient.

Similarly, the m -group H_j in the enhancement node layer is denoted as $H_{jm} = [H_1 H_2 \dots H_m]$. Combine Z_{in} and H_j to obtain the hidden layer A :

$$A = (Z_{in} | H_{jm}) \quad (4)$$

Then, the label of training data can be represented as:

$$Y = (Z_{in} | H_{jm})W = AW \quad (5)$$

where W is the weight of the output layer connected to the hidden layer, that is the network parameters to be learned, which can be calculated by the matrix pseudo-inverse:

$$W = A^+ Y \quad (6)$$

$$A^+ = \lim_{\lambda \rightarrow 0} \left(\lambda I + A^T A \right)^{-1} A^T \quad (7)$$

where A^+ is the pseudo-inverse matrix of matrix A ; I is a unit matrix, and $\lambda > 0$ is a hyperparameter, regularization coefficient.

BLS has two key characteristics compared with the deep neural networks. First, to better represent the input data and enhance computing efficiency, it employs sparse self-encoders to filter the random features of the input data into sparse and compact feature sets, then mines the key features using sparse feature learning models. Secondly, it addresses the problem that if the network model cannot reach the required accuracy in the deep learning system, it needs to add network layers or retrain the network after changing the structure. BLS employs the incremental learning algorithm to dynamically adjust the model by adding hidden layer nodes, which can obtain great accuracy in a short time. Figure 2 shows the algorithm flow chart of BLS.

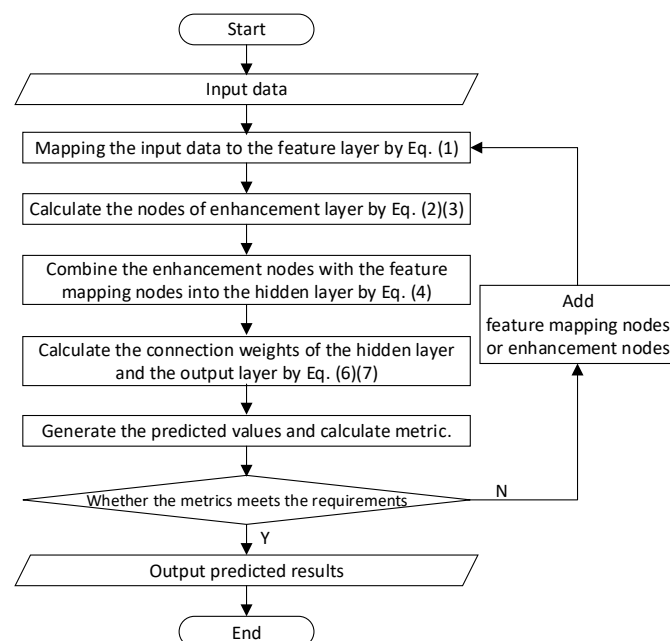


Figure 2. The algorithm flow chart of broad learning system.

2.2. Sparrow Search Algorithm (SSA)

Sparrow search algorithm is a new method of swarm intelligence optimization that is relied on sparrows' predatory and anti-predatory behavior [30]. It divides sparrows into explorers and followers and designs the following rules according to the sparrow's movement patterns.

- (1) Similar to the slap swarm algorithm [36], sparrows in the population are divided into explorers and followers according to their fitness. The fitness is the objective function of optimization, which reflects the quality of the sparrow's position.
- (2) Sparrows with good fitness are explorers, and others act as followers. The explorer is responsible for investigating food-rich locations and guiding the followers to foraging locations and directions. The followers were able to search for the explorer with the best feeding position and then foraged around it.
- (3) The fitness of a sparrow is dynamic, so the identity of explorers and followers can change with each other, but the proportion of explorers remains the same.
- (4) The bad fitness of the followers, the worse their foraging position is indicated. These followers may randomly fly to other places to forage.
- (5) A certain percentage of individuals in the sparrow population was selected as scouter, responsible for monitoring the safety of their surroundings. When a predator is detected, the scouter will sound an alarm, and when the alarm value is bigger than the safety value, the explorer will lead the followers to a safer area to forage.
- (6) When danger is recognized, sparrows located at the edge of the group will quickly move to a safe area to get a better position, while sparrows located in the center will move randomly.

The steps of SSA are as follows, and its algorithm flow chart is shown in Figure 3.

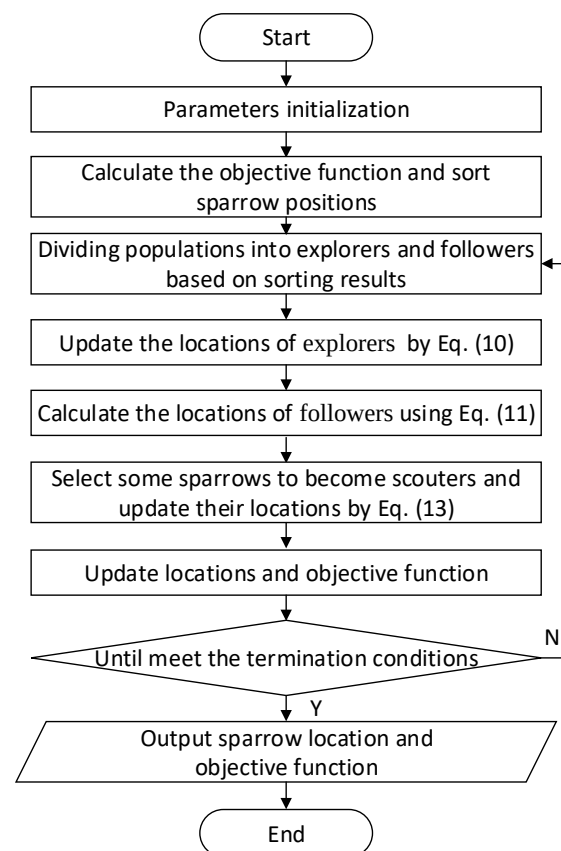


Figure 3. The algorithm flow chart of sparrow search algorithm.

Step 1: Parameter initialization, which mainly includes setting the number of sparrow population, the proportion of explorers, the location of sparrows, and the number of iterations. The population containing n sparrows can be expressed as:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^d \\ x_2^1 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^d \end{bmatrix} \quad (8)$$

where x_n^d denotes the position of the n -th sparrow in dimension d ; n is the population size, and d is the dimension of the variable to be optimized.

Step 2: Calculate the objective function and sort the sparrow positions. The objective function of the i -th sparrow can be indicated as:

$$F_i = f([x_i^1 \ x_i^2 \ \dots \ x_i^d]) \quad (9)$$

where f denotes the objective function.

Step 3: Determine whether the current position of the population is safe and update the explorers' position:

$$X_{ij}^{t+1} = \begin{cases} X_{ij}^t \cdot \exp\left(\frac{-i}{a \cdot b}\right), & R_2 < ST \\ X_{ij}^t + QL, & R_2 \geq ST \end{cases} \quad (10)$$

where X_{ij}^t denotes the value of the i -th sparrow in the t -th iteration in the j -th dimension; b is a constant whose value is the maximum number of iterations; R_2 denotes the warning value in the range of $[0, 1]$, which is a uniformly distributed random number; ST denotes the safety threshold and takes the value range of $[0.5, 1.0]$; L is a $1 \times d$ dimensional matrix; Q is a random number subjecting to normal distribution. When $R_2 < ST$ means that the current location is safe and the sparrow flock goes to look for food. Conversely, the current location is threatened and the explorer needs to guide the sparrow flock to look for a new place to find food.

Step 4: Determine the state of the follower and update its position. The location is updated as follows:

$$X_{ij}^{t+1} = \begin{cases} Q \cdot \exp\left(\frac{X_w^t - X_{ij}^t}{i^2}\right), & i > n/2 \\ X_p^{t+1} + |X_{ij}^t - X_p^{t+1}| \cdot A^+ \cdot L, & i \leq n/2 \end{cases} \quad (11)$$

$$A^+ = A^T (AA^T)^{-1} \quad (12)$$

where X_w denotes the worst position in the sparrow population; X_p is the position of the optimal explorer; A is a $1 \times d$ dimensional matrix with each dimensional value randomly generated from $[-1, 1]$. When $i > n/2$, it means that the follower is poorly positioned and does not get food, it needs to go to other places where it can get more food. Conversely, it continues to search for food near the explorer.

Step 5: Some sparrows find the danger and become scouts, updating the location of the scouts as follows:

$$X_{ij}^{t+1} = \begin{cases} X_b^t + \beta \cdot |X_{ij}^t - X_b^t|, & f_i > f_g \\ X_{ij}^t + K \cdot \left(\frac{|X_{ij}^t - X_w^t|}{(f_i - f_w) + \varepsilon}\right), & f_i = f_g \end{cases} \quad (13)$$

where X_b denotes the best position in the population; f_i is the objective function of i -th sparrow, f_g is the best value of the objective function, and f_w is the worst value of the

objective function; β is a standard normally distributed random number; K is a uniform random number with values in the range of $[-1, 1]$; ε is a smaller value to prevent the denominator from being zero.

Step 6: Update the objective function.

Step 7: Determine whether it satisfies the iteration stop condition and, if not, repeat steps 3 to 6.

3. Broad Learning System Based on the Sparrow Search Algorithm (SSA-BLS)

To minimize the impact of network hyperparameters and improve the accuracy of network flow forecasting, this paper employs the sparrow search algorithm to optimize hyperparameters of the broad learning system, shrinkage coefficient (r), and regularization coefficient (λ), to obtain the optimal hyperparameters and use them to build the training model. We named this method Sparrow Search Algorithm-Broad Learning System (SSA-BLS), and the algorithm is broken down into five steps.

Step 1: Parameter initialization. Determine the parameters of SSA, for example, explorer proportion and population size. Determine the range of shrinkage coefficient (r) and regularization coefficient (λ), respectively, and generate p (p is the population size) groups of initial hyperparameters as the initial position of the sparrow. The sparrow population is expressed as:

$$X = \begin{bmatrix} r_1 & \lambda_1 \\ r_2 & \lambda_2 \\ \vdots & \vdots \\ r_p & \lambda_p \end{bmatrix} \quad (14)$$

where r and λ are randomly generated. They are the hyperparameters to be optimized.

Step 2: Choosing the root mean square error (RMSE) of the BLS's prediction as to the objective function. Using the p sets of initial hyperparameters generated in the first step trains BLS to obtain the initial objective function. The objective function of the i -th sparrow is calculated as follows:

$$f_i = \sqrt{\frac{\sum_{j=1}^n (\hat{y}_j^i - y_j)^2}{n}}, i = 1, 2, \dots, p \quad (15)$$

where $\hat{y}_j^i$ is the predicted value of the j -th sample of the BLS trained with the i -th set of hyperparameters; y_j is the true value of the j -th sample; n is the number of training samples. The smaller the f_i , the better.

Step 3: The objective function is input into SSA, and execute the algorithm. According to the algorithm to update the sparrow population and objective function to achieve optimization of the hyperparameters of BLS.

Step 4: If the predefined number of iterations is reached, the optimization is completed and output the minimum value of the objective function:

$$f_m = \min([f_1, f_2, \dots, f_p]) \quad (16)$$

where m is the subscript of the minimum objective function. Then the hyperparameters obtained by SSA are:

$$r, \lambda = x_m = [r_m \quad \lambda_m] \quad (17)$$

Step 5: Put the hyperparameters r and λ obtained in the previous step into the BLS, train and build the network flow prediction model.

The SSA-BLS flow chart is given in Figure 4.

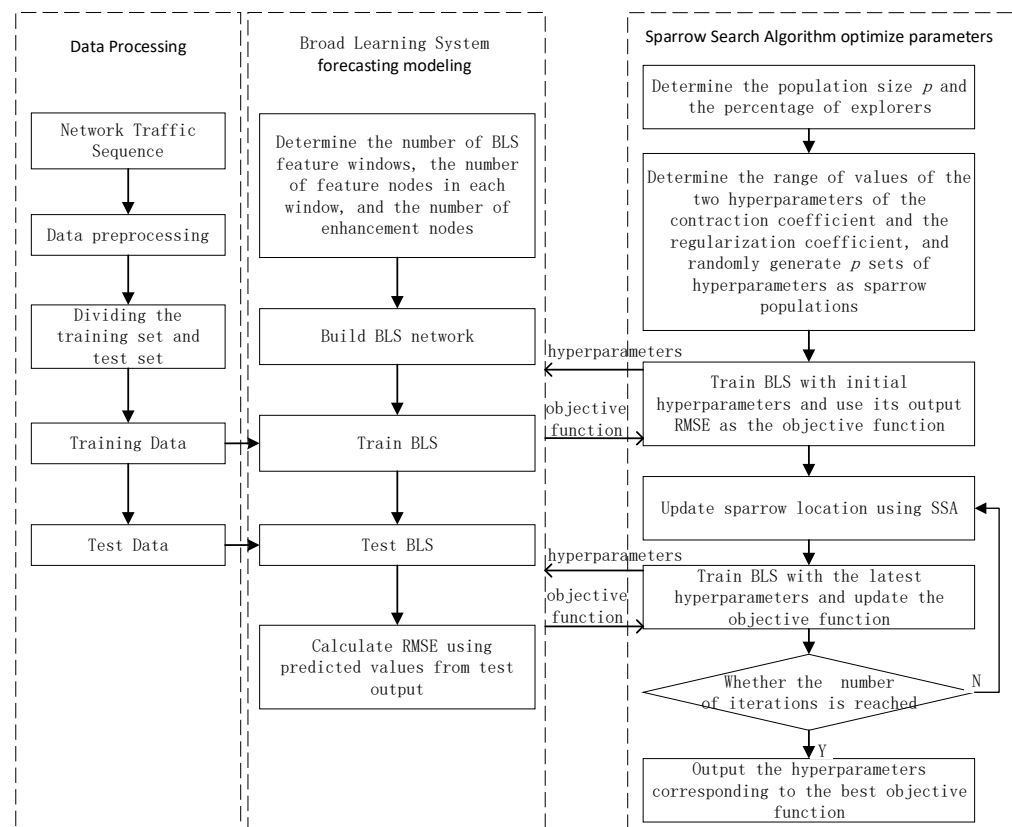


Figure 4. The algorithm flow chart of SSA-BLS.

4. Experimentation

All the experimental programs are developed based on python 3.8, the main packages used include numpy1.21 and pandas1.3, and the deep learning related models are implemented using pytorch3.8. The experimental environment is Windows 10OS, Intel (R) Core (TM) i5-1135G7 2.40GHz CPU, and 16.0GB RAM.

4.1. Datasets

The experimental dataset uses the core network traffic dataset of European cities and the academic backbone network traffic dataset of the UK.

The core network traffic dataset of European cities: the traffic of 11 European cities in the private ISPs. The traffic in bits on a transatlantic link from 7 June 2005, at 06:57 to 31 July 2005, at 11:17 are collected with a sampling interval of five minutes.

UK academic backbone traffic dataset: the dataset collects gathering flow from UK academic network backbone from 09:30 on 19 November 2004, to 11:11 on 27 January 2005, with a sampling interval of five minutes.

We use the data from 1 July to 25 July 2005, in the core network traffic dataset of European cities as the training set and from 26 July to 28 July as the test set; the data from 1 January to 24 January 2005, in the UK academic network backbone traffic dataset as the training set and the test set is from 25 January to 27 January.

4.2. Parameters and Evaluation Indicators

The SSA-BLS parameters are chosen as follows. Setting population size as 50, the explorer proportion is 20%, and the maximum number of iterations is 5; the number of windows in the mapping layer is 10, the number of nodes within each window in the feature mapping layer is 10, the enhancement nodes' number is 50, and the values of shrinkage coefficient (r) and regularization coefficient (λ) are taken in the ranges of $[0.09, 0.999999]$ and $[2^{-30}, 2^{-35}]$, respectively.

Mean squared error (MSE), root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and moving average (MA) are used as evaluation indicators. These indicators are calculated as follows:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (18)$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad (19)$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad (20)$$

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \right| \quad (21)$$

$$\text{MA} = 1 - \text{MAPE} = \left(1 - \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \right| \right) * 100\% \quad (22)$$

where samples number is n , y_i is the true value and $\hat{y}_i$ is the output. The smaller the MSE, RMSE, MAE, and MAPE, the better, while MA is closer to 100% indicating better model prediction performance.

4.3. Results and Discussion

The flow values of $[T - 11, T]$ were input into the SSA-BLS to predict the flow at the moment of $T + 1$ in the experiment. We compare SSA-BLS with the similarly structured BLS, Extreme Learning Machine (ELM) [37], Stochastic Configuration Networks (SCN) [38], RVFLNN, dRVFL (deep RVFL) (the variant of RVFL) [39], and the LSTM [40] that is commonly used in network traffic prediction and used to evaluate the quality of the SSA-BLS. Each model is run 100 times independently, and the prediction metrics of different models are evaluated individually each time, taking the average metrics as the final result of each model. The following are the parameters of each model: the values of r and λ for BLS are automatically obtained from $\{0.1, 0.5, 0.9, 0.99, 0.9999, 0.99999\}$ and $\{2^{-30}, 2^{-20}, 2^{-10}, 0.5, 1, 5, 10\}$, respectively, and the remaining parameters are the same as those of the SSA-BLS; the SCN can have a maximum of 250 hidden layer nodes, the training tolerance is 0.001, and candidate nodes maximum allowed 100; the regularization factor of RVFL is 1×10^{-3} , and hidden layer has 100 nodes; dRVFL parameters are the same as RVFL; the hidden layer of ELM contains 200 nodes, and the maxing coefficient for distance and dot product input activations is 1.0; the LSTM contains 3 hidden layers, each with 12 blocks, and is trained with a learning rate of 1×10^{-2} , batch size of 64 and epoch is 15. Tables 1 and 2 show the prediction performance of two datasets on the different models.

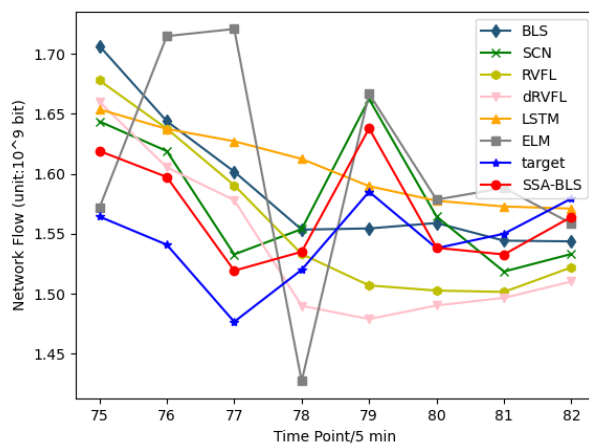
Table 1. Experimental results of a core network traffic dataset in European cities.

	MSE	RMSE	MAE	MAPE	MA
SSA-BLS	0.0159047	0.1261069	0.0937315	0.0294284	97.057155%
BLS	0.0781227	0.2551322	0.1878021	0.0571019	94.289801%
SCN	0.0154907	0.1244372	0.0934485	0.0295662	97.043378%
RVFL	0.0254023	0.1593589	0.1208186	0.0388347	96.116525%
dRVFL	0.0227553	0.1507691	0.1135728	0.0367191	96.328085%
ELM	0.1394488	0.3686439	0.2710739	0.0780252	92.197470%
LSTM	0.0781441	0.2502372	0.1884968	0.0517535	94.824642%

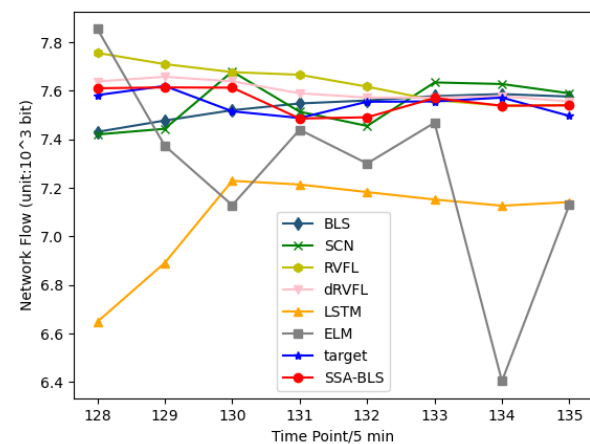
Table 2. Experimental results of UK academic backbone network traffic dataset.

	MSE	RMSE	MAE	MAPE	MA
SSA-BLS	0.0071345	0.0844570	0.0618339	0.0136258	98.637411%
BLS	0.0913392	0.2639297	0.1873060	0.0440354	95.596458%
SCN	0.0097822	0.0982373	0.0668127	0.0143890	98.561099%
RVFL	0.0289572	0.1701013	0.1179645	0.0264767	97.352324%
dRVFL	0.0234114	0.1527494	0.1058797	0.0232484	97.675151%
ELM	0.1051171	0.3157292	0.2166875	0.0426738	95.732612%
LSTM	0.3192739	0.3290859	0.2518907	0.0595686	94.043138%

On the test set of the public dataset, Figure 5 shows the predicted versus true values of the SSA-BLS model versus the other models. Moreover, to better validate the prediction accuracy of the SSA-BLS model, the model is applied to a private traffic dataset. The private traffic dataset is derived from the real incoming traffic data of switch interfaces of an enterprise from 5 October to 18 October 2021. We employ the data from 5 October to 16 October 2021, in the private dataset as training data, using the data from 17 October to 18 October as test data.



(a)



(b)

Figure 5. (a) Prediction results for a core network traffic dataset in European cities; (b) UK academic backbone network traffic dataset forecast results.

Since the sampling interval of the enterprise switch interface traffic data is unequal, the resampling is first performed: the average value of the interface traffic within 5 min is calculated, and if there is no traffic data within 5 min, the previous value is used to fill in. Meanwhile, there are great abnormal traffic values in the original data, and to lessen the influence of abnormal values, the data are smoothed using spectral smoothing (spectral smoother). Table 3 shows the experimental results.

Table 3. Enterprise cloud platform switch interface traffic data set experimental results.

	MSE	RMSE	MAE	MAPE	MA
SSA-BLS	0.0000734	0.0082991	0.0063628	0.0021407	99.785924%
BLS	0.0103714	0.0811761	0.0563080	0.0176119	98.238804%
SCN	0.0001742	0.0130396	0.0067544	0.0021857	99.781427%
RVFL	0.0361230	0.1899801	0.1288009	0.0400804	95.991952%
dRVFL	0.0327578	0.1807739	0.1277397	0.0403579	95.964208%
ELM	0.0579519	0.2382928	0.1327614	0.0400340	95.996599%
LSTM	0.0283041	0.1057008	0.0759722	0.0238097	97.619024%

Figure 6 shows the predicted versus true values of the SSA-BLS compared to others on the test data in the private dataset. It is clear from Tables 1–3 that the hyperparameters have a strong influence on BLS. If the hyperparameters are bad, the prediction performance of BLS will be degraded. The results show that the SSA-BLS model has better prediction accuracy than the other models on both the UK academic backbone network traffic dataset and the enterprise cloud platform switch interface traffic dataset, and its prediction performance on the European urban core network traffic dataset is only slightly below SCN. It can be seen that the SSA-BLS model, which is obtained after optimizing BLS using SSA, provides optimal hyperparameters for BLS through SSA, so that the SSA-BLS model can choose to capture the time characteristics of traffic better, and its network traffic prediction capability gains a large improvement compared with the original BLS model.

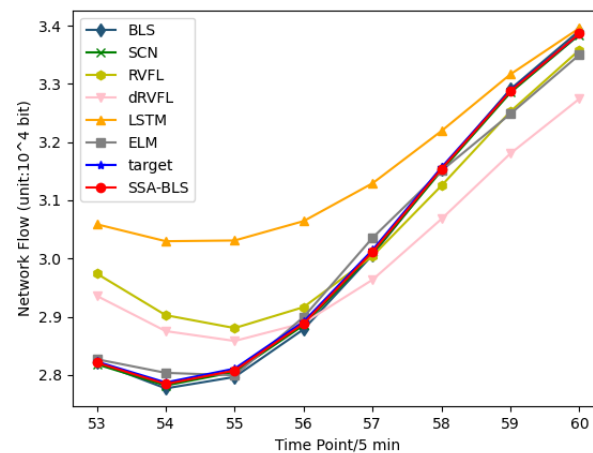


Figure 6. Enterprise cloud platform switch interface traffic dataset prediction results.

Meanwhile, this paper uses BLS for network traffic prediction based on the advantage of less training time due to its “expanding landscape” network structure. The running time of BLS in SSA-BLS is the main factor affecting the time of SSA-BLS. In order to verify the advantage of SSA-BLS model in time consumption, we compare the running time of BLS and the running time of LSTM for one epoch on three datasets, and the experimental results are shown in Figure 7. In Figure 7, dataset 1, dataset 2, and dataset 3 are the UK academic backbone network traffic dataset, European urban core network traffic dataset, and enterprise cloud platform switch interface traffic dataset, respectively. The experimental results show that BLS can complete the training in a shorter time, and the larger the data volume, the greater the advantage of BLS.

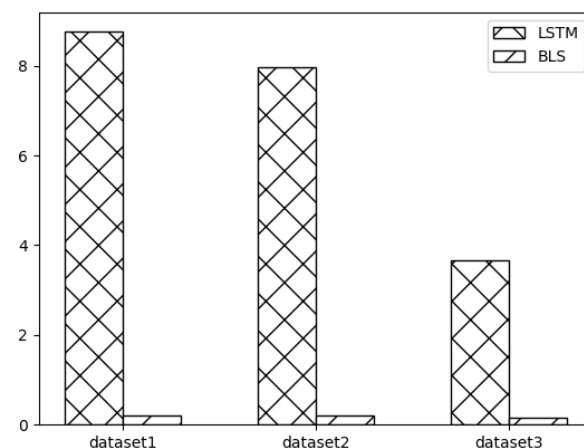


Figure 7. BLS and LSTM runtime.

5. Conclusions, Limitations, and Future Research

Predicting future traffic on the cloud platform interface can be used to assist the cloud platform in provisioning resources and planning the network, and it is an effective way to help achieve dynamic resource allocation and load balancing with the volume of requests. In this paper, we propose a model named SSA-BLS to predict the network interface traffic. The model uses SSA to optimize two hyperparameters in BLS to obtain the optimal combination of hyperparameters quickly and enhance the performance of BLS. At the same time, the model uses BLS to capture the traffic timing features and reduce the training time of the prediction model. Finally, we apply SSA-BLS to the short-term network traffic prediction, selecting two public datasets of network traffic and a real dataset of network switch interface traffic of an enterprise cloud platform for experiments. Finally, we compare the SSA-BLS with other models, and the experiments show that the SSA-BLS can select better hyperparameters to make the network traffic prediction accuracy above 97%.

Currently, most network traffic prediction models have a strict sampling interval for training data, requiring the data to be equally spaced. Sometimes, frequent sampling is required to obtain more fine-grained data. However, frequent sampling for a long time will increase resource consumption, and it is difficult to present the data with equal spacing due to the inevitable packet loss during the network transmission. Therefore, future research will be conducted for the prediction modeling of non-equally spaced sampled data to reduce the requirement of data spacing and improve the generalizability of the model.

Author Contributions: Project administration, S.L.; Writing—original draft, X.L.; Writing—review & editing, P.Z. and G.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the National Key R&D Program under grant no.2020YFB1713300, and by Higher Education Project of Guizhou Province under grants no. [2020]005 and [2020]009.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Singh, S.; Chana, I. A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges. *J. Grid Comput.* **2016**, *14*, 217–264. [\[CrossRef\]](#)
2. Katris, C.; Daskalaki, S. Comparing Forecasting Approaches for Internet Traffic. *Expert Syst. Appl.* **2015**, *42*, 8172–8183. [\[CrossRef\]](#)
3. Yang, J.; Xiao, X.; Mao, S.; Rao, C.; Wen, J. Grey Coupled Prediction Model for Traffic Flow with Panel Data Characteristics. *Entropy* **2016**, *18*, 454. [\[CrossRef\]](#)
4. Vo, N.; Ślepaczuk, R. Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index. *Entropy* **2022**, *24*, 158. [\[CrossRef\]](#)
5. Zhong-Da, T.; Shu-Jiang, L.; Yan-Hong, W.; Xiang-Dong, W. Network Traffic Prediction Based on ARIMA with Gaussian Process Regression Compensation. *J. Beijing Univ. Posts Telecommun.* **2017**, *40*, 65.
6. Kim, S. Forecasting Internet Traffic by Using Seasonal GARCH Models. *J. Commun. Netw.* **2011**, *13*, 621–624. [\[CrossRef\]](#)
7. Kim, M. Network Traffic Prediction Based on INGARCH Model. *Wirel. Netw.* **2020**, *26*, 6189–6202. [\[CrossRef\]](#)
8. Alekseeva, D.; Stepanov, N.; Veprev, A.; Sharapova, A.; Lohan, E.S.; Ometov, A. Comparison of Machine Learning Techniques Applied to Traffic Prediction of Real Wireless Network. *IEEE Access* **2021**, *9*, 159495–159514. [\[CrossRef\]](#)
9. Wang, Q.; Fan, A.; Shi, H. Network Traffic Prediction Based on Improved Support Vector Machine. *Int. J. Syst. Assur. Eng. Manag.* **2017**, *8*, 1976–1980. [\[CrossRef\]](#)
10. Nabipour, M.; Nayyeri, P.; Jabani, H.; Mosavi, A.; Salwana, E.; Shahab, S. Deep Learning for Stock Market Prediction. *Entropy* **2020**, *22*, 840. [\[CrossRef\]](#)
11. Liu, F.; Liu, B.; Sun, C.; Liu, M.; Wang, X. Deep Belief Network-Based Approaches for Link Prediction in Signed Social Networks. *Entropy* **2015**, *17*, 2140–2169. [\[CrossRef\]](#)
12. Huang, Z.; Xia, J.; Li, F.; Li, Z.; Li, Q. A Peak Traffic Congestion Prediction Method Based on Bus Driving Time. *Entropy* **2019**, *21*, 709. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Miguel, M.L.F.; Penna, M.C.; Nievola, J.C.; Pellenz, M.E. New Models for Long-Term Internet Traffic Forecasting Using Artificial Neural Networks and Flow Based Information. In Proceedings of the 2012 IEEE Network Operations and Management Symposium, Maui, HI, USA, 16–20 April 2012; pp. 1082–1088. [\[CrossRef\]](#)
14. Nie, L.; Jiang, D.; Yu, S.; Song, H. Network Traffic Prediction Based on Deep Belief Network in Wireless Mesh Backbone Networks. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017; pp. 1–5.

15. Fang, L.; Cheng, X.; Wang, H.; Yang, L. Mobile Demand Forecasting via Deep Graph-Sequence Spatiotemporal Modeling in Cellular Networks. *IEEE Internet Things J.* **2018**, *5*, 3091–3101. [\[CrossRef\]](#)
16. Zhang, K.; Zhao, X.; Li, X.; You, X.; Zhu, Y. Network Traffic Prediction via Deep Graph-Sequence Spatiotemporal Modeling Based on Mobile Virtual Reality Technology. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 2353875. [\[CrossRef\]](#)
17. Chen, C.P.; Liu, Z. Broad Learning System: An Effective and Efficient Incremental Learning System without the Need for Deep Architecture. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 10–24. [\[CrossRef\]](#)
18. Pao, Y.-H.; Takefuji, Y. Functional-Link Net Computing: Theory, System Architecture, and Functionalities. *Computer* **1992**, *25*, 76–79. [\[CrossRef\]](#)
19. Pao, Y.-H.; Park, G.-H.; Sobajic, D.J. Learning and Generalization Characteristics of the Random Vector Functional-Link Net. *Neurocomputing* **1994**, *6*, 163–180. [\[CrossRef\]](#)
20. Igel'nik, B.; Pao, Y.-H. Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-Link Net. *IEEE Trans. Neural Netw.* **1995**, *6*, 1320–1329. [\[CrossRef\]](#)
21. Chen, C.P.; Wan, J.Z. A Rapid Learning and Dynamic Stepwise Updating Algorithm for Flat Neural Networks and the Application to Time-Series Prediction. *IEEE Trans. Syst. Man Cybern. Part B* **1999**, *29*, 62–72. [\[CrossRef\]](#)
22. Gong, X.; Zhang, T.; Chen, C.P.; Liu, Z. Research Review for Broad Learning System: Algorithms, Theory, and Applications. *IEEE Trans. Cybern.* **2021**, 1–29. [\[CrossRef\]](#)
23. Jin, J.-W.; Chen, C.P. Regularized Robust Broad Learning System for Uncertain Data Modeling. *Neurocomputing* **2018**, *322*, 58–69. [\[CrossRef\]](#)
24. Chen, C.P. Broad Learning System and Its Structural Variations. In Proceedings of the 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 13–15 September 2018; pp. 000011–000012.
25. Chen, C.P.; Liu, Z.; Feng, S. Universal Approximation Capability of Broad Learning System and Its Structural Variations. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *30*, 1191–1204. [\[CrossRef\]](#) [\[PubMed\]](#)
26. Gambardella, M.; Martinoli, M.B.A.; Stützle, R.P.T. Ant Colony Optimization and Swarm Intelligence. In *5th International Workshop*; Springer: Berlin/Heidelberg, Germany, 2006.
27. Figueiredo, E.M.; Luder'mir, T.B.; Bastos-Filho, C.J. Many Objective Particle Swarm Optimization. *Inf. Sci.* **2016**, *374*, 115–134. [\[CrossRef\]](#)
28. Zhou, Z.; Zhang, R.; Wang, Y.; Zhu, Z.; Zhang, J. Color Difference Classification Based on Optimization Support Vector Machine of Improved Grey Wolf Algorithm. *Optik* **2018**, *170*, 17–29. [\[CrossRef\]](#)
29. Xu, X.; Liu, C.; Zhao, Y.; Lv, X. Short-Term Traffic Flow Prediction Based on Whale Optimization Algorithm Optimized BiLSTM_Attention. *Concurr. Comput. Pract. Exp.* **2022**, e6782. [\[CrossRef\]](#)
30. Xue, J.; Shen, B. A Novel Swarm Intelligence Optimization Approach: Sparrow Search Algorithm. *Syst. Sci. Control. Eng.* **2020**, *8*, 22–34. [\[CrossRef\]](#)
31. Zhang, C.; Ding, S. A Stochastic Configuration Network Based on Chaotic Sparrow Search Algorithm. *Knowl.-Based Syst.* **2021**, *220*, 106924. [\[CrossRef\]](#)
32. Zhang, F.; Sun, W.; Wang, H.; Xu, T. Fault Diagnosis of a Wind Turbine Gearbox Based on Improved Variational Mode Algorithm and Information Entropy. *Entropy* **2021**, *23*, 794. [\[CrossRef\]](#)
33. Tian, Z.; Chen, H. A Novel Decomposition-Ensemble Prediction Model for Ultra-Short-Term Wind Speed. *Energy Convers. Manag.* **2021**, *248*, 114775. [\[CrossRef\]](#)
34. Gai, J.; Zhong, K.; Du, X.; Yan, K.; Shen, J. Detection of Gear Fault Severity Based on Parameter-Optimized Deep Belief Network Using Sparrow Search Algorithm. *Measurement* **2021**, *185*, 110079. [\[CrossRef\]](#)
35. Song, C.; Yao, L.; Hua, C.; Ni, Q. A Water Quality Prediction Model Based on Variational Mode Decomposition and the Least Squares Support Vector Machine Optimized by the Sparrow Search Algorithm (VMD-SSA-LSSVM) of the Yangtze River, China. *Environ. Monit. Assess* **2021**, *193*, 363. [\[CrossRef\]](#) [\[PubMed\]](#)
36. Devarapalli, R.; Sinha, N.K.; Rao, B.V.; Knypinski, L.; Lakshmi, N.J.N.; Márquez, F.P.G. Allocation of Real Power Generation Based on Computing over All Generation Cost: An Approach of Salp Swarm Algorithm. *Arch. Electr. Eng.* **2021**, *70*, 337–349.
37. Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme Learning Machine: Theory and Applications. *Neurocomputing* **2006**, *70*, 489–501. [\[CrossRef\]](#)
38. Wang, D.; Li, M. Stochastic Configuration Networks: Fundamentals and Algorithms. *IEEE Trans. Cybern.* **2017**, *47*, 3466–3479. [\[CrossRef\]](#)
39. Shi, Q.; Katuwal, R.; Suganthan, P.N.; Tanveer, M. Random Vector Functional Link Neural Network Based Ensemble Deep Learning. *Pattern Recognit.* **2021**, *117*, 107978. [\[CrossRef\]](#)
40. Zhuo, Q.; Li, Q.; Yan, H.; Qi, Y. Long Short-Term Memory Neural Network for Network Traffic Prediction. In Proceedings of the 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Nanjing, China, 24–26 November 2017; pp. 1–6. [\[CrossRef\]](#)