Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 11, November 2019, pp. 2972-2981
DOI: 10.23940/ijpe.19.11.p16.29722981

Improving Extreme Learning Machine by a Level-based Learning Swarm Optimizer and its Application to Fault Diagnosis of 3D Printers

Jianyu Long[a], Ying Hong[a,*], Shaohui Zhang[a], Diego Cabrera[b], and Jingjing Zhong[a]

[a]*School of Mechanical Engineering, Dongguan University of Technology, Dongguan, 523808, China*
[b]*GIDTEC, Universidad Politécnica Salesiana, Ecuador*

Abstract

Fault diagnosis plays a significant role in the printing quality for 3D printers. In this paper, an extreme learning machine based on level-based learning swarm optimizer (LLSO-ELM) is proposed to diagnose faults of delta 3D printers. Extreme learning machine (ELM) achieves better performance in learning speed than traditional gradient descent algorithms. However, the random inputs weights and hidden biases are influential factors for the accuracy and generalization performance of ELM. LLSO has competitive performance in solution quality and computational efficiency for large scale optimization problems, and it is used to obtain the optimum configuration of the weights and biases for ELM. The proposed model is tested by using the attitude data of a delta 3D printer under different operating modes. The experimental results verify that the proposed approach performs better in generalization and stability than ELM.

Keywords: extreme learning machine; evolutionary extreme learning machine; level-based learning swarm particle; metaheuristic; fault diagnosis

(Submitted on April 12, 2019; Revised on October 11, 2019; Accepted on November 16, 2019)

1. Introduction

A 3D printer is a machine that shapes raw materials into objects of any shape based on three-dimensional model data [1]. 3D printing is an additive manufacturing technique and has been widely used in different industries, such as biomechanics, medicine, and prototyping [2]. The health condition of mechanical transmission systems is one of the major factors affecting printing quality. Therefore, fault diagnosis of mechanical transmission systems plays a significant role in ensuring the quality of printing. Our research object is a type of 3D printer with complex transmission structure, and it is called the delta 3D printer [3].

In the past few decades, using machine learning to diagnosis machinery faults has attracted many researchers' attention [4]. Extensive machine learning approaches have emerged in this field, including support vector machine (SVM) [5], convolution neural network (CNN) [6-7], deep belief network (DBN) [8], and sparse autoencoder (SAE) [9]. ELM, which was proposed by Huang et al. [10] in 2004, is an effective model to train a single hidden-layer feedforward neural network (SLFN) [11]. Unlike gradient descent algorithms, ELM calculates output weights using the Moore-Penrose generalized inverse after randomly selecting input weights and hidden biases. Compared with other machine learning algorithms, ELM has excellent performance in the speed of learning [11-13]. However, more hidden nodes are always used in ELM than classical gradient descent algorithms [14-15]. There are two main challenges in using ELM for fault diagnosis: (1) the random network parameters result in bad generalization performance, and (2) the model falls into overfitting because of redundant hidden nodes [16]. To overcome these weaknesses, many studies have already adopted evolutionary algorithms to improve the condition of ELM parameters. In this case, network structures obtain better performance and higher stability [17].

At present, evolutionary ELM using particle swarm optimization (PSO) [18], competitive swarm optimizer (CSO) [19], and differential evolution (DE) [20] have been studied widely. PSO aims to obtain the optimal solution by modifying the particles, which represent the candidate solutions, in the swarm over the process of iterations. Each particle has position and

* Corresponding author.
E-mail address: hongying@dgut.edu.cn

velocity in the search space. *pbest* and *gbest* are two key factors that change the position and velocity of particles in the iteration process. *pbest* represents the best experience of each particle, while *gbest* is the best experience of the swarm. Xu et al. [21] proposed an evolutionary ELM model based on PSO. The proposed model can achieve results similar to those of the back-propagation algorithm. Han et al. [13] proposed a PSO-ELM model and verified the superiority of the approach using three real benchmark classification problems. Subbulakshmi et al. [15] proposed a classification with PSO-ELM for medical datasets. CSO can be regarded as an improved PSO, and it performs better at solving high dimensional problems than PSO. Mohapatra et al. [22] proposed an improved CS-based ELM for binary medical datasets and evaluated the proposed model by using several evaluation measures. Eshtay et al. [23] proposed a CSO-ELM by using CSO to optimize the parameters of the classical ELM and the regularized ELM for medical classification problems. DE is a group-based heuristic search algorithm that is realized by generating initial individuals randomly and producing the next generation through a set of operators including mutation, recombination, and selection. Zhu et al. [24] proposed a novel ELM using DE to select the input weights. Experimental results demonstrated that the proposed method has good generalization performance. Nahvi et al. [25] proposed a soil temperature prediction approach by combining DE with ELM.

Many evolutionary algorithms have demonstrated potential at optimizing ELM parameters. However, they still encounter some problems, such as slow learning speed and low stability, for ELMs with complicate network structures. The fault diagnosis for 3D printers is a complicated problem, and many parameters are required to solve it. Accordingly, in this paper, we proposed a new approach that combines ELM with the level-based learning swarm optimizer algorithm (LLSO). LLSO was proposed by Yang et al. [26] for large scale optimization. It has been verified that LLSO has competitive performance in quality and computation speed for large-scale optimization.

The paper is organized as follows: Section 2 introduces the ELM and LLSO algorithm briefly. Section 3 describes the details of our proposed model. Section 4 introduces experimental setup and data preparation. Section 5 reports and discusses the experimental results. Finally, Section 6 concludes the paper.

2. Preliminaries

2.1. Extreme Learning Machine

ELM was first proposed by Huang et al. [10]. It is an advanced training algorithm based on a single hidden layer feedforward network (SLFN) [27]. In general, artificial neural network (ANN) trains the weights of networks using a gradient descent algorithm such as the back propagation algorithm [28]. ELM determines the values of input weights and hidden biases randomly, and it subsequently calculates the output weights using the Moore-Penrose generalized inverse. Therefore, compared to ANN, the training speed of ELM is faster.

The framework of ELM is shown in Figure 1, where the input weights refer to the weights between input nodes and hidden neurons, and the output weights refer to the weights between output nodes and hidden neurons. For a classification problem, assume that the n-dimensional dataset has N samples and can be divided into m categories. We use (x_i, t_i), i = 1, 2, $\cdots$, N to denote the dataset, where x_i ($x_i = [x_{i1}, x_{i2}, \cdots, x_{in}]^{\mathrm{T}}, i = 1,2, \cdots, N$) represents the i^{th} input data, and t_i ($t_i = [t_{i1}, t_{i2}, \cdots, t_{im}]^{\mathrm{T}}, i = 1,2, \cdots, N$) represents the i^{th} target data. L denotes the number of hidden neurons. The output layer of ELM can be formulated as follows:

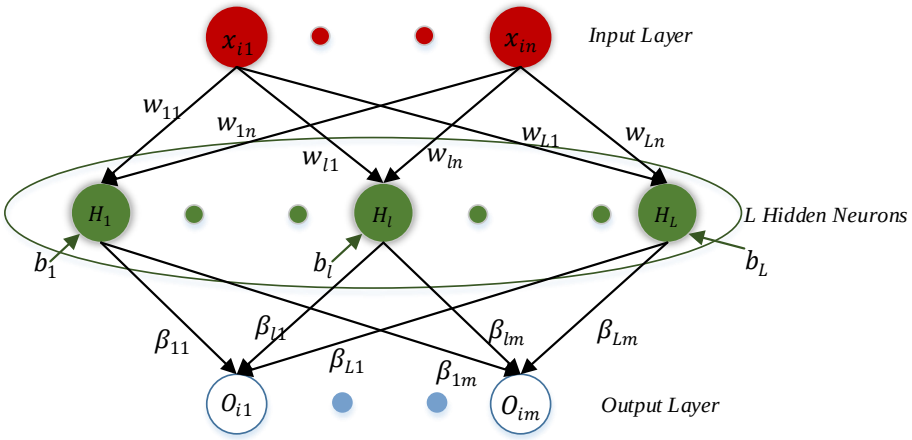

Figure 1. Framework of extreme learning machine

$$o_i = \sum_{l=1}^{L} \beta_l . g(w_l . x_i + b_l) , i = 1,2, \cdots, N \tag{1}$$

Where $w_l = [w_{l1}, w_{l2}, \cdots, w_{ln}]^{\mathrm{T}}$ represents the weight parameters between the input layer and the l^{th} hidden node, b_l denotes the l^{th} hidden node bias, and $\beta_l = [\beta_{l1}, \beta_{l2}, \cdots, \beta_{lm}]^{\mathrm{T}}$ represents the weight parameters between the l^{th} hidden node and the output layer. Function g is an activation function, which can be set as ReLU, a sigmoidal function, a radial basis function (RBF), and so on. For a high precision model, the following equation is valid:

$$\sum_{i=1}^{N} \|o_j - t_j\| = 0 \tag{2}$$

Therefore, Equation (1) can be written as follows:

$$\sum_{l=1}^{L} \beta_l . g(w_l . x_i + b_l) = t_i, i = 1,2, \cdots, N \tag{3}$$

Equation (3) can be expressed as $H\beta = T$ in matrix form, where

$$H = \begin{pmatrix} g(w_1 . x_1 + b_1) & \cdots & g(w_L . x_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(w_1 . x_N + b_1) & \cdots & g(w_L . x_N + b_L) \end{pmatrix}_{N \times L} \tag{4}$$

$$\beta = [\beta_1^{\mathrm{T}} \quad \cdots \quad \beta_L^{\mathrm{T}}]_{L \times m}^{\mathrm{T}} \tag{5}$$

$$T = [t_1^{\mathrm{T}} \quad \cdots \quad t_N^{\mathrm{T}}]_{N \times m}^{\mathrm{T}} \tag{6}$$

The loss function can be described as below:

$$\|H\beta' - T\| = min_{\beta} \|H\beta - T\| \tag{7}$$

Since the input weight parameters *w* and bias parameters *b* have been determined randomly, the parameter β' can be calculated by using the least square method. β' can be denoted as

$$\beta' = (H^{\mathrm{T}} H)^{-1} H^{\mathrm{T}} T \tag{8}$$

In summary, the procedure of ELM can be outlined as below:

Step 1 Determine the input weight matrix *w* and hidden layer bias vector *b* randomly.

Step 2 Calculate the output matrix of hidden layer *H*.

Step 3 Use Equation (8) to obtain the output weight matrix.

2.2. Level-based Learning Swarm Optimizer Algorithm

LLSO, which was proposed by Yang et al. [26] in 2017, is an improved version of PSO to address larger-scale optimization problems. As the scale of the problem increases, the space of feasible solutions and computation complexity also increase sharply. Therefore, optimization algorithms can easily fall into local optima and premature convergence when solving these problems. LLSO has been verified to be efficient in finding the global optima of large-scale problems.

The two main ideas of LLSO are the level-based learning (LL) strategy and exemplar selection. The LL strategy is employed to divide particles into the *NL* level after sorting them according to their fitness with social learning particle swarm optimization (SL-PSO) [29]. We stipulate that better particles belong to a higher level with a smaller level index.

Therefore, if L_i is used to represent the i^{th} level, L_1 is the highest level containing the best particles. Since more beneficial information may be held by the particles in higher levels, they should be used to guide those in lower levels to search the global optimum area. Assume that the size of particle swarm is *NP*, and the level size is *LS*, so the number of levels *NL* is $NL = NP/LS$. The procedure of the LL strategy is outlined as below:

Step 1 Sort the particle swarm in ascending order of fitness.

Step 2 Divide the particle swarm into *NL* levels.

Step 3 Update the particles of L_i ($2 \le i \le NL - 1$) by learning the particles in L_1 - L_{i-1}.

As mentioned above, the particles of the lower level need to learn from the particles of the higher level. Therefore, selecting two exemplars in the higher level is a key issue. The exemplar selection strategy provides an approach to choosing exemplars by taking the exploration and exploitation of particles into consideration. Exploration and exploitation are two key evaluation criteria in large-scale optimization [30].

For a particle in L_i, the procedure of exemplar selection is outlined as:

Step 1 Select rl_1 and rl_2 randomly, where $rl_1, rl_2 \in [1, i-1]$.

Step 2 If $rl_1 > rl_2$, exchange their values.

Step 3 Select k_1 and k_2 randomly, where $k_1, k_2 \in [1, LS]$.

Step 4 Return the k_1^{th} particle (X_{rl_1,k_1}) in level rl_1 $\mathrm{X}_{\mathrm{rl}_1,\mathrm{k}_1}$ and the k_2^{th} particle (X_{rl_2,k_2}) in level rl_2.

The two exemplars X_{rl_1,k_1} and X_{rl_2,k_2} will guide the evolution of particle in L_i. This strategy properly reserves the different exploration and exploitation in different levels.

The formulas of LLSO for updating particle $X_{i,j}$ are shown as follows:

$$v_{i,j} = r_1 \times v_{i,j} + r_2 \times \left(X_{rl_1,k_1} - X_{i,j}\right) + \emptyset \times r_3 \times (X_{rl_2,k_2} - X_{i,j}) \tag{9}$$

$$X_{i,j} = X_{i,j} + v_{i,j} \tag{10}$$

Where $X_{i,j}$ is the j^{th} particle in the L_i level, and $v_{i,j}$ is its velocity. X_{rl_1,k_1} and X_{rl_2,k_2} are determined by using the exemplar selection strategy. The parameters r_1, r_2, and r_3 are randomly selected within [0, 1]. $\emptyset$ is a control parameter to determine the influence of the second exemplar, and it is also within [0, 1]. While updating the particles in L_2, the two exemplars are both selected from L_1. The formulas for updating the particles in L_2 are described as follows:

$$v_{2,j} = r_1 \times v_{2,j} + r_2 \times \left(X_{1,k_1} - X_{2,j}\right) + \emptyset \times r_3 \times (X_{1,k_2} - X_{2,j}) \tag{11}$$

$$X_{2,j} = X_{2,j} + v_{2,j} \tag{12}$$

The particles in L_1 contain the best solutions in the swarm, so they directly leave into the swarm of the next generation. The iteration process of LLSO is illustrated in Figure 2.

3. The Proposed LLSO-ELM Model

As mentioned before, the input weight parameters and bias parameters are randomly selected in ELM. However, the performance of ELM with random weights and biases cannot achieve the best performance. The random parameters lead to more hidden neurons and worse generalization performance [13]. Therefore, we propose a hybrid approach name LLSO-ELM, which is realized by using LLSO to find the best parameters of ELM. Due to the full connection mode between the input layer and the hidden layer in ELM, the weights and biases optimization problem is a large-scale optimization problem. LLSO is an improved PSO algorithm that has been verified with competitive performance for solving such problems.

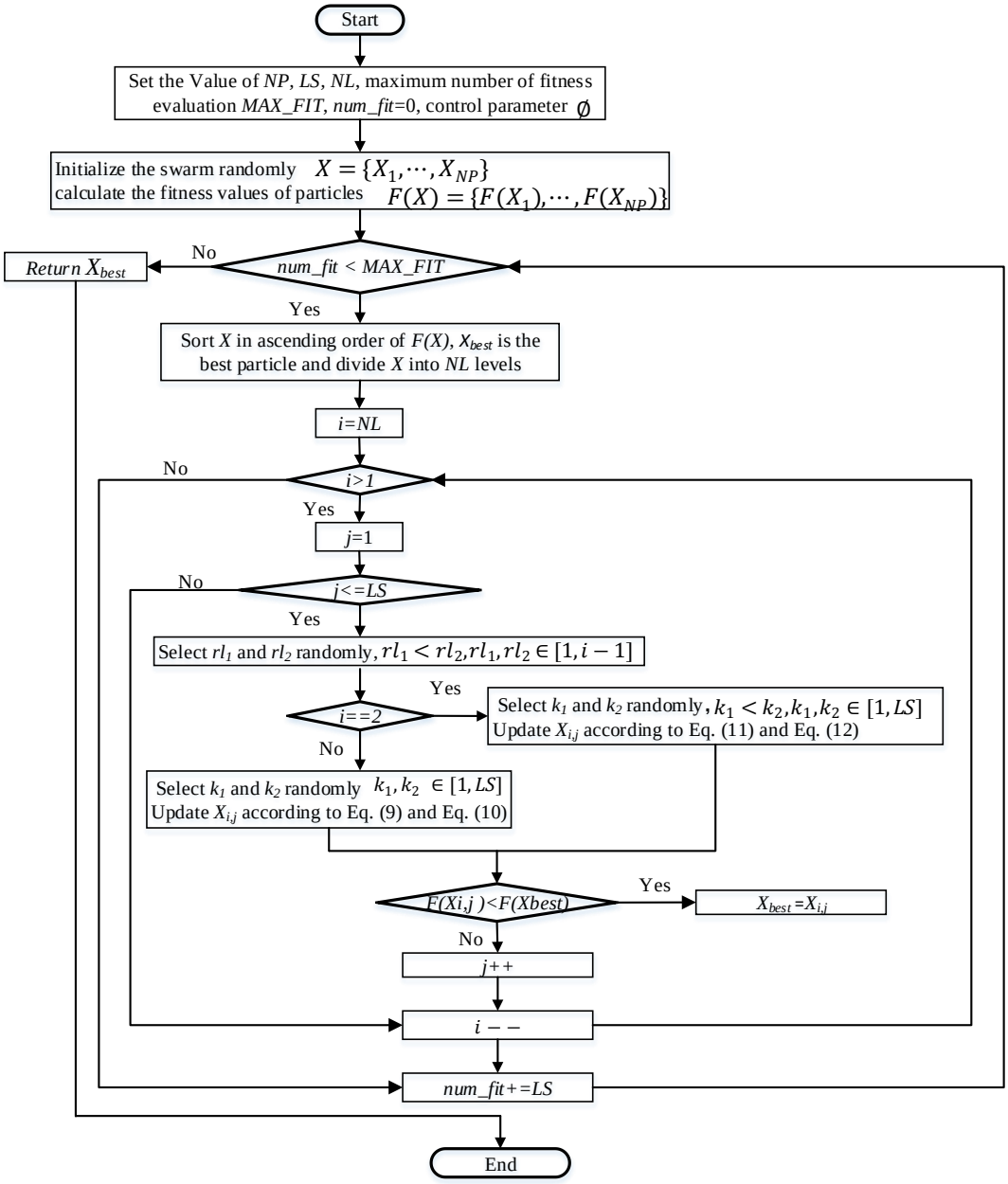

Figure 2. The iteration process of LLSO

Particle encoding and fitness function are two key problems in the procedure of using LLSO to optimize the parameters of ELM. Thus, we mainly introduce the design of the particle encoding and fitness function in the following. Subsequently, the procedure of LLSO-ELM is described.

3.1. Encoding and Fitness Function

Particles contain the solution information for an optimization problem in LLSO. The first step of LLSO is to initialize the particle swarm randomly and calculate the fitness value of each particle. Therefore, it is essential to address the particle encoding problem and fitness calculation method.

The particle of LLSO-ELM is composed of the weight vector of input layer and the bias vector of hidden neurons. A particle *P* can be expressed as follows:

$$P = [w_{11}, w_{12}, \cdots, w_{1L}, \cdots, w_{n1}, \cdots, w_{nL}, b_1, \cdots, b_L] \tag{13}$$

Where L is the number of hidden neurons, and n is the dimensions of the input dataset. The length of particle *LenOfParticle* can be calculated by Equation (14).

$$LenOfParticle = (n + 1) \times L \tag{14}$$

Fitness is used to evaluate the quality of particles. For classification problems, the equation to calculate the fitness value can be defined as

$$Fitness = 1 - Accuracy \tag{15}$$

Where *Accuracy* is the ratio of the number of samples correctly classified to the total number of samples.

3.2. LLSO-ELM Procedure

On the basis of the above discussion, the procedure of LLSO-ELM can be described as follows:

Firstly, initialize the swarm of LLSO randomly. Each particle is composed of input weights and biases, whose values are initialized within the range [-1, 1].

Secondly, calculate the fitness value of each particle using Equation (15). The accuracy can be obtained after calculating the output weights and constructing the ELM using the particle components.

Thirdly, update the lower level of swarm using Equations (9) to (12).

Fourthly, repeat the second and third step up to a predetermined number of iterations.

Finally, the corresponding ELM of the best particle produced from the above evolutionary process is used on the testing data to obtain the prediction accuracy of our approach.

The detailed procedure can be also illustrated as shown in Algorithm 1.

Algorithm 1 Procedure of LLSO-ELM

Input: Training data $D\ D = \{(x_i, t_i) | x_i \in R^n, t_i \epsilon R^m, i = 1, \cdots i, \cdots N\}$, *NP*, *LS*, *NL*, *MAX_FIT*, hidden neurons *L*, ∅
Output: Prediction accuracy *PA*β
//Get the best particle by LLSO
num_fit = 0;
Generate *NP* random particles based on the particle construction shown in Eq. (13);
Calculate the fitness value *F(X)* of each particle;
num_fit += *NP*;
X_{best} is used to represent the best particle;
while *num_fit* < *MAX_FIT* **do**
  Sort particles in ascending order of *F(X)* and divide them into *NL* levels;
  for *i* ={*NL*,…, *3*} **do**
    for *j* ={*1*,…, *LS*} **do**
      Select two levels rl_1 and rl_2 from the top $(i - 1)$ levels, ensuring $rl_1 < rl_2$;
      Select k_1 and k_2 randomly, where $k_1, k_2 \in [1, LS]$;
      Update $X_{i,j}$ according to Eq. (9) and Eq. (10) by using particles X_{rl_1,k_1} and X_{rl_2,k_2};
      Calculate the fitness value $F(X_{i,j})$;
      if ($F(X_{i,j}) < F(X_{best})$) **then**
        $X_{best} = X_{i,j}$;
      End if
    End for
    num_fit += *LS*;
  End for
  for *j* = {1, …, *LS*} **do**
    Select k_1 and k_2 randomly from [1, *LS*], $k_1, k_2 \in [1, LS]$ ensuring $k_1 < k_2$;
    Update $X_{2,j}$ according to Eq. (11) and Eq. (12) by using particles X_{1,k_1} and X_{1,k_2};
    Calculate the fitness value $F(X_{2,j})$
    if ($F(X_{2,j}) < F(X_{best})$) **then**
      $X_{best} = X_{2,j}$;
    End if
  End for
  num_fit += *LS*;

29:**End while**
//Calculate the prediction accuracy on the testing data
30: Get the input weights w and biases b from the particle X_{best};
31: Calculate the hidden layer output matrix H;
32: Calculate the output weights β by using Eq. (8);
33: Using the constructed ELM to calculate the prediction accuracy PA on the testing Data;
34: return PA;

4. Experimental Setup and Data Preparation

By assembling a delta 3D printer, an attitude sensor, and a workstation, the experimental setup shown in Figure 3 was built for data collection. The delta 3D printer was employed to offer a predefined movement under the normal operating mode or faulty operating modes. In this experiment, 15 faulty operating modes were generated by simulating the wear of different joint bearings and synchronous belts. More specifically, since there were 12 joint bearings between the sliders and the moving platform, 12 joint bearing faults were simulated by loosening a 1.75-turn for the screw of each joint bearing. For ease of description, these joint bearing faults are represented as modes A, B, C, D, E, F, G, H, I, J, K, and L. Moreover, three synchronous belt faults, represented as modes M, N, and O, were simulated by relaxing the length of two teeth for each synchronous belt. Note that the normal operating mode is denoted as mode P.

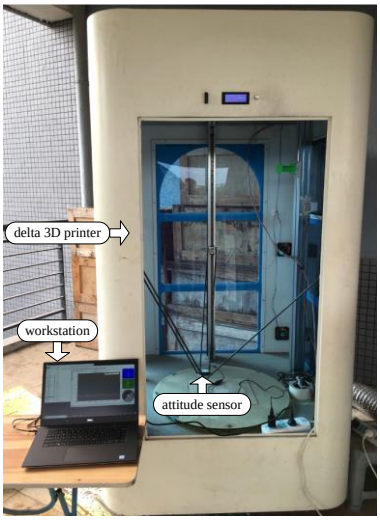

Figure 3. The experimental setup for collecting attitude data of delta 3D printer

The attitude data under different modes was collected using the attitude sensor mounted on the moving platform. The built-in module chip of the attitude sensor was JY901, and three-axial angular velocity signals, three-axial vibratory acceleration signals, three-axial magnetic field intensity signals and three-axial attitude angles could be collected by the attitude sensor.

In the data collection experiments, the 3D printer was used to print three cylindrical shell models with different radiuses of 75mm (denoted as R75), 90mm (denoted as R90), and 105mm (denoted as R105). For each cylindrical shell model, attitude data were collected from the circle movements with specific radius executed by the moving platform. In the experiment, the sampling frequency was set to 100Hz. Finally, 3,200 samples were collected for each cylindrical shell model, and each sample had 12-channel data. In the following fault diagnosis experiments, the dataset will be divided into two parts: a training dataset (80%) and a testing dataset (20%).

5. Experimental Results

5.1. Fault Diagnosis Results using the Proposed LLSO-ELM

The proposed LLSO-ELM was programmed with Python language, and it was trained and tested five times by using the collected dataset. Through conducting the parameter tuning experiment, the parameters of LLSO-ELM were set as follows: NP = 100, NL = 10, LS = 10, MAX_FIT = 4510, $\emptyset$ = 0.5, and L = 90. The prediction accuracies of each dataset are described in Table 1. As we can see, a high prediction accuracy can be obtained by our approach, and its performance is very stable.

Table 1. The prediction accuracies of the proposed LLSO-ELM for each dataset

Dataset	Order					Mean	Variance
	1st Run	2nd Run	3rd Run	4th Run	5th Run		
R75	0.950	0.934	0.950	0.953	0.942	0.946	0.0068
R90	0.955	0.964	0.960	0.966	0.963	0.961	0.0036
R105	0.959	0.955	0.966	0.962	0.964	0.961	0.0039

5.2. Comparison with ELM

To demonstrate the superiority of the proposed LLSO-ELM, it is compared with the ELM introduced by Huang et al [10]. As declared by Eshtay et al [23], there is no rule of thumb for setting the number of hidden neurons in ELM. Therefore, for a comprehensive comparison, an extensive experiment was conducted by changing the number of hidden neurons in LLSO-ELM and ELM. In the experiment, by increasing ten hidden neurons each time, both algorithms were sequentially performed starting with 50 hidden neurons and ending with 100 hidden neurons. All the algorithms were used to run the three datasets five times in the same operating environment described in Section 5.1. For each run, the training dataset and the testing dataset were randomly generated, and the prediction accuracies were calculated subsequently. Figure 4 shows the average prediction accuracies of LLSO-ELM and ELM with different hidden neurons for each dataset. From the results, we can find that the presented LLSO-ELM can achieve better performance than ELM on all the hidden nodes. In addition, the performance of LLSO-ELM is more stable because all the prediction accuracies of LLSO-ELM exceed 91%.

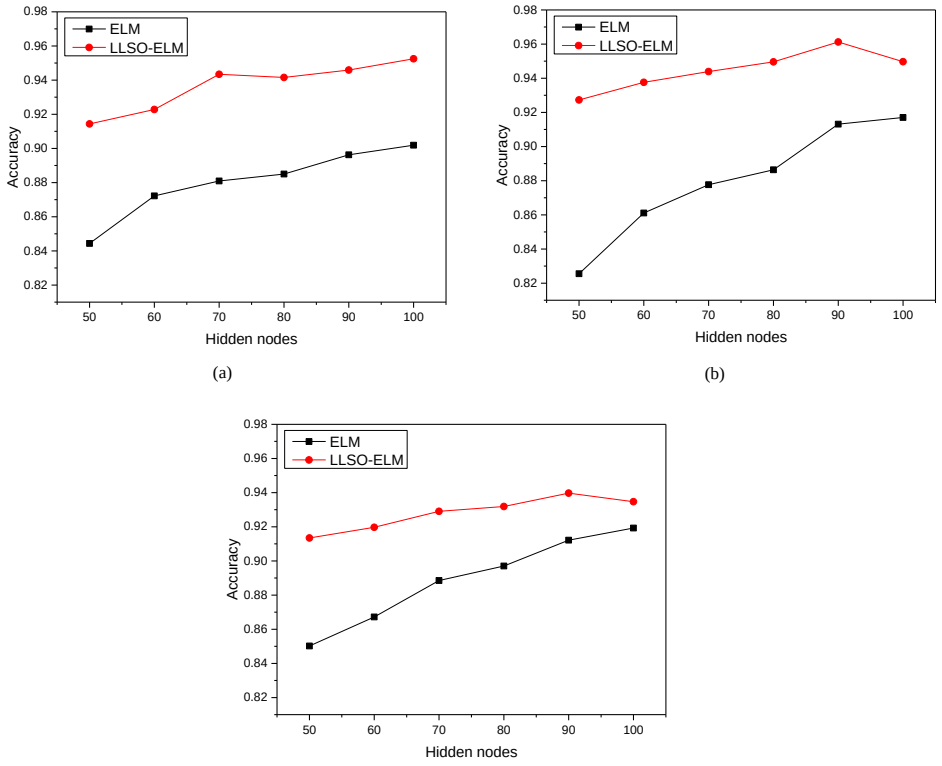

Figure 4. The average prediction accuracies of LLSO-ELM and ELM with different hidden neurons for each dataset: (a) R75; (b) R90; (c) R1059

6. Conclusions

An extreme learning machine based on level-based learning swarm optimizer (LLSO-ELM) was proposed in this paper to diagnose machinery faults of delta 3D printers. In order to further improve the generalization performance of ELM, LLSO-ELM was designed by using LLSO to optimize the input weights and hidden biases of ELM. An experimental setup consisting of a delta 3D printer, an attitude sensor, and a workstation was built for collecting data under different operating modes. By using these data, the performance of LLSO-ELM was evaluated. The experimental results reflect that the presented LLSO-ELM can obtain high prediction accuracy for the fault diagnosis of delta 3D printers. Moreover, its performance is very stable when setting different hidden neurons.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (No. 71801046, 51775112, and 51605406), Natural Science Foundation of Guangdong Province (No. 2018A030310029), Youth Innovative Talent Project (No. 2017KQNCX191) from the Department of Education of Guangdong Province, DGUT Research Project (KCYKYQD2017011), and Research Start-up Funds of DGUT (No. GC300501-12 and GC300501-26).

References

1. T. D. Ngo, A. Kashani, G. Imbalzano, K. T. Q. Nguyen, and D. Hui, "Additive Manufacturing (3D Printing): A Review of Materials, Methods, Applications and Challenges," *Composites Part B-Engineering*, Vol. 143, pp. 172-196, 2018
2. X. Song, Y. Pan, and Y. Chen, "Development of a Low-Cost Parallel Kinematic Machine for Multidirectional Additive Manufacturing," *Journal of Manufacturing Science and Engineering*, *Transactions of the ASME*, Vol. 137, No. 2, 2015
3. H. Kim, Y. R. Lin, and T. L. B. Tseng, "A Review on Quality Control in Additive Manufacturing," *Rapid Prototyping Journal*, Vol. 24, pp. 645-669, 2018
4. Z. Gao, C. Cecati, and S. X. Ding, "A Survey of Fault Diagnosis and Fault-Tolerant Techniques-Part II: Fault Diagnosis with Knowledge-based and Hybrid/Active Approaches," *IEEE Transactions on Industrial Electronics*, Vol. 62, pp. 3768-3774, 2015
5. A. Widodo and B. S. Yang, "Support Vector Machine in Machine Condition Monitoring and Fault Diagnosis," *Mechanical Systems and Signal Processing*, Vol. 21, No. 6, pp. 2560-2574, 2008
6. L. Wen, X. Y. Li, L. Gao, and Y. Y. Zhang, "A New Convolutional Neural Network-based Data-Driven Fault Diagnosis Method," *IEEE Transactions on Industrial Electronics*, Vol. 65, No. 7, pp. 5990-5998, 2017
7. O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, and D. J. Inman, "Realtime Vibration-based Structural Damage Detection using One-Dimensional Convolutional Neural Networks," *Journal of Sound and Vibration*, Vol. 388, pp. 154-170, 2016
8. Q. Gao, W. Yang, and Q. Li, "Research on Deep Belief Network Layer Tendency and Its Application into Identifying Fault Images of Aerial Images," *Chinese Journal of Scientific Instrument*, Vol. 36, No. 6, pp. 1267-1274, 2015
9. W. J. Sun, S. Y. Shao, R. Zhao, R. Q. Yan, X. W. Zhang, and X. F. Chen, "A Sparse Auto-Encoder-based Deep Neural Network Approach for Induction Motor Faults Classification," *Measurement*, Vol. 89, pp. 171-178, 2016
10. G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks," in *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, pp. 985-990, 2004
11. G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, Vol. 70, No. 1, pp. 489-501, 2005
12. S. Ding, X. Xu, and R. Nie, "Extreme Learning Machine and Its Application," *Neural Computing and Applications*, Vol. 25, No. 3-4, pp. 549-556, 2014
13. F. Han, H. F. Yao, and Q. H. Ling, "An Improved Evolutionary Extreme Learning Machine based on Particle Swarm Optimization," *Neurocomputing*, Vol. 116, pp. 87-93, 2011
14. L. H. Cho, D. J. Lee, and M. G. Chun, "Parameter Optimization of Extreme Learning Machine using Bacterial Foraging Algorithm," *Journal of Korean Institute of Intelligent Systems*, Vol. 17, No. 6, pp. 807-812, 2007
15. C. V. Subbulakshmi and S. N. Deepa, "Medical Dataset Classification: A Machine Learning Paradigm Integrating Particle Swarm Optimization with Extreme Learning Machine Classifier," *The Scientific World Journal*, DOI 10.1155/2015/418060, 2015
16. Y. Wu, Y. S. Zhang, X. B. Liu, Z. H. Cai, and Y. M. Cai, "A Multiobjective Optimization-based Sparse Extreme Learning Machine Algorithm," *Neurocomputing*, Vol. 317, pp. 88-100, 2018
17. P. Mohapatra, S. Chakravarty, and P. K. Dash, "An Improved Cuckoo Research based Extreme Learning Machine for Medical Data Classification," *Swarm and Evolutionary Computation*, Vol. 24, pp. 25-49, 2015
18. R. Eberhart and J. Kennedy, "A New Optimizer using Particle Swarm Theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39-43, 1995
19. I. Fister Jr., X. S. Yang, D. Fister, and I. Fister, "Cuckoo Search: A Brief Literature Review," *Cuckoo Search and Firefly Algorithm*, Vol. 516, pp. 49-62, 2013
20. K. Fleetwood, "An Introduction to Differential Evolution," in *Proceedings of the 2004 Mathematics and Statistics of Complex Systems* (*MASCOS*), 2004
21. Y. Xu and Y. Shu, "Evolutionary Extreme Learning Machine-based on Particle Swarm Optimization," in *Proceedings of the 2006 International Symposium on Neural Networks*, pp. 644-652, 2006
22. P. Mohapatra, S. Chakravarty, and P. K. Dash, "An Improved Cuckoo Search based Extreme Learning Machine for Medical Data Classification," *Swarm and Evolutionary Computation*, Vol. 24, pp. 25-49, 2015
23. M. Eshtay, H. Faris, and N. Obeid, "Improving Extreme Learning Machine by Competitive Swarm Optimization and Its Application for Medical Diagnosis Problems," *Expert Systems with Applications*, Vol. 104, pp. 134-152, 2018
24. Q. Y. Zhu, A. K. Qin, P. N. Suganthan, and G. B. Huang, "Evolutionary Extreme Learning Machine," *Pattern Recognition*, Vol. 38, No. 10, pp. 1759-1763, 2005
25. B. Nahvi, J. Habibi, K. Mohammadi, S. Shamshirband, and O. S. AI Razgan, "Using Self-Adaptive Evolutionary Algorithm to Improve the Performance of an Extreme Learning Machine for Estimating Soil Temperature," *Computers and Electronics in Agriculture*, Vol. 124, pp. 150-160, 2016
26. Q. Yang, W. N. Chen, J. D. Deng, Y. Li, T. L. Gu, and J. Zhang, "A Level-based Learning Swarm Optimizer for Large-Scale Optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 22, pp. 578-594, 2017

27. G. B. Huang, L. Chen, and C. K. Siew, "Universal Approximation using Incremental Constructive Feedforward Networks with Random Hidden Nodes," *IEEE Transactions on Neural Network*, Vol. 17, No. 4, pp. 879-892, 2006
28. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagation Errors," *Nature*, Vol. 323, pp. 533-536, 1986
29. R. Cheng and Y. C. Jin, "A Social Learning Particle Swarm Optimization Algorithm for Scalable Optimization," *Information Sciences*, Vol. 291, pp. 43-60, 2015
30. R. Cheng and Y. C. Jin, "A Competitive Swarm Optimizer for Large Scale Optimization," *IEEE Transactions on Cybernetics*, Vol. 45, No. 2, pp. 191-204, 2015