International Journal of Control, Automation and Systems 15(3) (2017) 1466-1477
http://dx.doi.org/10.1007/s12555-015-0463-7
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

Elongation Prediction of Steel-strips in Annealing Furnace with Deep Learning via Improved Incremental Extreme Learning Machine

Chao Wang*, Jian-Hui Wang, Shu-Sheng Gu, Xiao Wang, and Yu-Xian Zhang

Abstract: The elongation of steel-strips in annealing furnace is an important factor that affects the position of welding line and safety of air-knife since there is no extra space to install welding line detector in field conditions. Therefore, predicting the elongation of steel-strips in the annealing process is important to fulfill the requirements of eliminating security risks and improving economic performance. In this paper, we propose a deep architectures called I-ELM/MLCSA autoencoders with the concept of stacked generalization philosophy to solve large and complex data mining problems. The comparison results of the case studies indicate that D-ELMs-AE/MLCSA is a promising prediction algorithm and can be employed for steel-strips elongation predictions with excellent performance.

Keywords: Baldwinian learning, Clone selection algorithm, deep learning, elongation prediction, incremental extreme learning machine, Lamarckian learning.

1. INTRODUCTION

In recent years, deep learning techniques have received a great deal of attention for its advantages on capturing relevant high-level abstractions and characterizing the data representations. Deep learning is a learning algorithm based on artificial networks which has multilayer perceptions [1–5]. Deep learning is capable to approximate complex functions and alleviate optimization difficulties associated with the deep models. Motivated by the remarkable success of deep learning techniques, recently, many algorithms based on deep learning have been implemented in a variety of practical applications and shown a state-of-the-art performance in pattern recognition, computer vision, wind speed predictions, moving object detection, wireless localization, and so forth [6–10]. The main interest of this work is to integrate multi-learning clonal selection theories into deep learning architecture to predict the position of welding line to improve the safety of air-knife in critical situations and the rate of capacity utilization.

In this paper, we design a new stacked deep learning architecture through multi-learning clonal selection theories and incremental extreme learning machine, which incorporates the clonal selection theories, including Baldwinian learning and Lamarckian learning, with I-ELM and extends the model into deep learning architecture. The proposed method performs excellent effects when solve large and complex data problems, which can be applied in the elongation prediction of steel-strips in annealing furnace. We implemented I-ELM/MLCSA autoencoder in each iteration of deep incremental extreme machine. The multi-learning clonal selection learning (D-ELMs/MLCSA) is applied to reconstruct the input data and estimate the errors of the prediction functions with t layer-by-layer architectures. Both of the supervised and unsupervised data can be regarded as the pertaining input of the proposed deep network. Moreover, the I-ELM/MLCSA autoencoder-based deep network (D-ELMs-AE/MLCSA) is capable to obtain the improvement for generalization performance and outperforms the deep models compared in the paper, including DNN, ML-ELM, DSVDD, AE-S-ELMs and PWDNN.

2. PRELIMINARIES

In this section, the main concepts and theories of clonal selection algorithm (CSA) are briefly reviewed. Then, a new multi-learning clonal selection algorithm combined with Baldwinian learning and Lamarckian learning is proposed in the rest of this subsection.

2.1. Clonal selection algorithm (CSA)

Clonal selection theory proposed by Burnet [13, 14] is

Manuscript received December 10, 2015; revised March 23, 2016, June 26, 2016, and July 19, 2016; accepted August 1, 2016. Recommended by Associate Editor Huaping Liu under the direction of Editor Euntai Kim. This work was supported by the National Natural Science Foundation of China (61102124), Liaoning Key Industry Programme(JH2/101).

Chao Wang, Jian-Hui Wang, Shu-Sheng Gu, and Xiao Wang are with the College of Information Science and Engineering, Northeastern University, Shenyang, 110819, China (e-mails: supper_king1018@163.com, wangjianhui@mail.neu.edu.cn, gushusheng@mail.neu.edu.cn, wangxiao.owl@gmail.com). Yu-Xian Zhang is with the bSchool of Electrical Engineering, Shenyang University of Technology, Shenyang Liaoning 110870, China (e-mail: yuxian524524@163.com).
* Corresponding author.

the fundamental of artificial immune system (AIS), which introduces the mechanism biological immune system and creates a new antibody population with the affinity calculation to expand the search scope. Besides, it recombines the population by updating the low affinity antibodies to increase the diversity of antibody population. Subsequently, Castro claimed the Clonal Selection Algorithm (CSA) [15] based on the clonal selection theory for optimization problems, which established the framework of the searching paradigm through immune clonal selection and affinity maturation with the concept of hypermutation. According to CSA, the following implementations are conducted to antibody population $A(t)$ after the selection operator is defined: Clonal Proliferation T^C, Hypermutation T^M, Clonal Selection T^S. Thus, the state transfer of antibody population can be expressed as the following evolution process.

2.2. Multi-learning clonal selection algorithm (ML CSA)

With the principle of Baldwinian Learning and Lamarckian Learning, the multi-learning clonal selection strategy is established, which expands the search space of antibodies information and enhances the reproduction ability of antibody population with high affinity. The antibody parameters with best fitness values are obtained . The state transfer of antibody population generated by multi-learning clonal selection strategy can be expressed as follows:

$$A(t)\xrightarrow{\textbf{clone}(H^C)}X(t)\xrightarrow{\textbf{B-Learning}(H_L^B)}$$
$$Y(t)\xrightarrow{\textbf{L-Learning}(H_L^L)}Z(t)\xrightarrow{\textbf{recombination}(Z(t)\cup A(t))}$$
$$D(t)\xrightarrow{\textbf{selection}(H^S)}A(t+1)$$

The framework of multi-learning clonal selection algorithm is summarized in Algorithm 1. The four main operators, Clonal Proliferation, Baldwinian Learning, Lamarckian Learning and Clonal Selection, are explained as follows:

2.2.1 Clonal proliferation

Define the antibody population $A(t)=\{A_1(t),A_2(t),\ldots,A_n(t)\})$, thus the clonal proliferation is implemented as follows: $X(t)=H^C(A(t)=\{H^C(A_1(t),H^C(A_2(t),\ldots,H^C(A_n(t))\})$,where, $x_i(t)=H^C(A_i(t)=\{X_{i1}(t),X_{i2}(t),\ldots,X_{iq_i}(t)\})$, $X_{ij}(t)=A_i(t)$, i=1,2,...,n , $j=1,2,\ldots,q_i$, q_i is the clonal scale of antibody population.

2.2.2 Baldwinian learning

Baldwinian Learning can acquire the best solutions with the better evolutionary method via transforming the shape of search space, which is based on Baldwin effect. Define $X(t)=\{X_1(t),X_2(t),\ldots,X_n(t)\}$, $X_i(t)=\{x_{i1}(t), x_{i2}(t),\ldots,x_{iq_i}(t)\}$, the specific implementation of Baldwinian Learning is as follows:

$$Y_{ij}(t)=H_L^B(x_{ij}(t))$$

```
Algorithm 1 Multi-learning Clonal Selection Algorithm (MLCSA)
Step 1 (Initialization): Randomly generate the initial
    antibody population $A(t)$, set $N=0$ and initialize the
    termination criterion;
Step 2 (Clonal Proliferation $H^C$): According to the affinity,
    generate amplificatory $X(t)$ from $A(t)$ with clonal
    proliferation operator, and scale of clone is a monotone
    increasing function of antibody affinity;
Step 3 (Multi-learning operation)
Step 3.1 (Baldwinian Learning $H_L^B$): Apply Baldwinian
    learning strategy on every antibody in $X(t)$, and
    generate the antibody population $Y(t)$, according to (1)
    and (2);
Step 3.2 (Lamarckian Learning $H_L^L$): Apply Lamarckian
    learning strategy on every clonal antibody in $Y(t)$, and
    generate the population $Z(t)$, according to (3);
Step 4 (Evaluation): Calculate the affinity of each antibody
    in $Z(t)$;
Step 5 (Recombination operation): Implement recombina-
    tion operation with $Z(t)$ and $A(T)$ to form antibody
    population $D(t)$;
Step 6 (Clonal Selection $H^S$): Generate new antibody popu-
    lation $A(t+1)$ by applying clonal selection operator to
    $D(t)$;
Step 7 (Termination): If termination criterion is satisfied,
    stop the algorithm and obtain antibody with the highest
    affinity in $A(t+1)$ as the output; otherwise, $t=t+1$,
    go to Step 2.
Endwhile
```

$$=\begin{cases} x_{ij}(t)+s\cdot(x_l(t)-x_m(t)), & \text{if rand} \le p_l \\ x_{ij}(t), & \text{else,} \end{cases} \tag{1}$$

where $l,m\in\{1,2,\ldots,n\}$, $l\neq m\neq i$, $x_l(k)$ and $x_m(t)$ denotes the antibodies randomly selected from $X_l(k)$ and $X_m(t)$, respectively, and $F(x_l(k))>F(x_m(k))$; $s>0$ is the strength of Baldwinian learning, $p_l\in(0,1]$ is the probability of Baldwinian learning, and *rand* denotes a random number chosen from a uniform distribution on the interval $[0,1]$. With Baldwinian learning, the population becomes:

$$Y(t)=\{Y_1(t),Y_2(t),\ldots,Y_n(t)\}, \tag{2}$$

where, $Y_i(t)=\{y_{i1}(t),\ldots,y_{iq_i}(t)\}$,$y_{ij}(k)=H_L^B(x_{ij}(t))$, $j=1,2,\ldots,q$, $i=1,2,\ldots,n.$

2.2.3 Lamarckian learning H_L^L

Randomly generate q_i stochastic direction vectors, and set the initial directions for the local searching and individual $\overrightarrow{a_i}$ as the initial point, search the local optimal solution in the neighborhood of $\overrightarrow{a_i}$ with local search method. The process can be expressed as follows:

$$\begin{aligned}&H_L^L(Y(t)) = Z(t)\\&= \{H_L^L(\overrightarrow{Y}_1^1(t), \overrightarrow{d}_1^1) + \ldots + H_L^L(\overrightarrow{Y}_1^{q_1}(t), \overrightarrow{d}_1^{q_1})\}\\&+ \ldots + \{H_L^L(\overrightarrow{Y}_n^1(t), \overrightarrow{d}_n^1) + \ldots + H_L^L(\overrightarrow{Y}_n^{q_1}(t), \overrightarrow{d}_n^{q_1})\},\end{aligned} \tag{3}$$

where $d_i^j(i = 1,2,...,n, j = 1,2,...,q_i)$ denote direction vectors.

2.2.4 Clonal selection H^S

Define $z^*(t) \in Z_i(t)$, $i = 1,2,...,n$, $z^*(t)$is the antibody with the highest affinity in $Z_i(t)$,furthermore, Implement the recombination operation with population and to form , which populations have applied with multi-learning operation. Generate the new antibody population by applying clonal selection operator to . The process is implemented as follows:

$$\begin{aligned}a(t+1) &= H^C(Z_i(t) \cup a_i(t))\\&= \begin{cases} z_i^*(t), & \text{if } F(z_i^*(t)) > F(a_i(t)) \\ a_i(t), & \text{else,} \end{cases}\end{aligned} \tag{4}$$

where $A_i(t+1) = H^C(Z_i(t) \cup A_i(t))$, $i = 1,2,...,n$.

The major aim is to enhance the antibody information including other antibodies' information to alert the search space with the help of Baldwinian learning and utilize the information of each individual to reinforce the exploitation with the help of Lamarckian learning.

2.3. Experimental studies on function optimization problems

In this section, experiments are carried out to evaluate the performance of MLCSA by solving 5 commonly used global optimization problems. MLCSA is compared with MLCSA, as well as other state-of-the-art evolutionary computing models. The parameters of BCSA are set as follows: the population size n = 50, the total clonal scale n_c =200, the probability of Baldwinian learning p_l=0.8, and the mutation probability p_m = 1/D, where D is the number of variables. Furthermore, the performance of MLCSA is analyzed. Table 1 gives the test functions in the experiments, which can be categorized into two types: f_1 is unimodal functions, $f_2 \sim f_5$ are unrotated multi-modal functions.

The experiments contain two parts in this subsection. (1) We compare MLCSA with the traditional CSA on the 5 test problems with 2 dimensions, 10 dimensions and 30 dimensions mentioned above, respectively ; (2) experiments are conducted for the proposed MLCSA model and other six evolutionary algorithms on the same optimization problems. The six representative evolutionary algorithms are listed as follows:

- Baldwinian Clonal Selection Algorithm (BCSA) [16];
- Lamarckian Clonal Selection Algorithm (LCSA) [17];
- Improved Chaotic Particle Swarm Optimization (ICPSO) [18];
- Improved MOEA/D with Baldwinian Learning (MOEA/
D/BL) [19];
- Differential Evolutionary Algorithm (DEA) [20];
- Hybrid Learning Clonal Selection Algorithm (HLCSA) [21];

Table 2 show the statistical results of traditional CSA and MLCSA. To evade possible biased comparisons, the experiments are designed to optimize the 5 test functions with 2, 10 and 30 dimensions (D=2, D=10 and D=30) respectively, based on 50 independent trials the maximum, minimum, mean and standard deviation are calculated. The best mean values and standard deviation values are shown in bold face. As the results show, MLCSA performs much better than traditional CSA. From the results, we can observe that HLCSA performs much better than CLONALG for all these test instances. MLCSA can find the exact global optima of functions f_1 (D=2), f_5 (D=10) and f_5(D=30) with probability 1 and the approximate global optima of functionsf_3 (D=10) and f_4 (D=10). From the statistical results, we can obtain the conclusion that the solutions of different optimization problems solved by MLCSA are outstanding and stable, due to the learning mechanism based on multi-learning with Baldwinian learning and Lamarckian learning strategies. Table 3 shows the results for BCSA, MOEA/D/BL, HLCSA, DMDE and MLCSA with 2, 10 and 30 dimensions (D=2, D=10 and D=30) respectively. Taking functions f_3, f_4 and f_5 as example, when D=10, Mean$\pm$Std values are 3.0295E-015$\pm$2.6618E-016, 4.3811E-016$\pm$7.2761E- 017, 0$\pm$0 respectively, which indicates that MLCSA scales well in dealing with high-dime-nsional optimization problems.

In this section, we propose MLCSA by incorporating two learning mechanisms, Baldwinian learning and Lamarckian learning, into CSA to guide the immune response process. The Baldwinian learning works for exploration (global search) by using the phenotype to guide the evolutionary search for good genotypes, and the Lamarckian learning works for reinforce the exploitation (local search) by replacing the locally improved individual back into the population to compete for reproductive opportunities. The Baldwinian learning operator allows learning antibodies to evolve much faster than the non-learning ones. The Lamarckian learning is applied to all the clones, which substitutes for the mutation operator in CSA to enhance the performance of local convergence and reduce the blindness of mutation. The performance of MLCSA outperforms other state-of-the-art evolutionary computing models with no degeneration for all the test problems.

Table 1. Benchmark functions used in our experimental study.

Name	Test function	D	S
Goldstein-Price	$f_1(x) = (1+(x_1+x_2+1)^2\cdot(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2))\cdot(30+(2x_1-3x_2)^2\cdot(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2))$	2	$x_1,x_2\in[-2,2]$
Schaffer F6	$f_2(x)=0.5+[(\sin\sqrt{x_1^2+x_2^2})^2-0.5]/[1.0+0.001(x_1^2+x_2^2)]^2$	2	$x_1,x_2\in[-100,100]$
Ackley	$f_3(x) = -20*exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-exp(-\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i))+20+e$	10/30	[-5,5]
Rastrigrin	$f_4(x)=\sum_{i=1}^{D-1}[x_i^2-10\cdot\cos(2\pi x_i)+10]$	10/30	[-5.12,5.12]
Weierstrass	$f_5(x)=\sum_{i=1}^{D-1}\{\sum_{k=1}^{20}[0.5^k\cos(2\pi\cdot 3^k(x_i+0.5))]\}\}-D\sum_{i=1}^{20}\{0.5^k\cos(2\pi\cdot 3^k\cdot 0.5)\}$	10/30	[-0.5,0.5]

Table 2. Results of traditional clonal selection algorithm and MLCSA when D=30.

Funs	D	CSA				MLCSA			
		Max	Min	Mean	Std	Max	Min	Mean	Std
f_1	2	9.1862E+004	1.4730E+003	1.3203E+004	2.8781E+001	0	0	**0**	**0**
f_2	2	9.3722E-001	5.5207E-001	8.0023E-001	1.6563E-002	4.6211E-001	3.8905E-001	**3.4407E-001**	**1.2152E-021**
f_3	10	4.4102E-002	5.2736E-005	2.8554E-003	8.3013E-003	3.2017E-015	0	**3.1025E-015**	**2.6993E-016**
	30	9.1202E-002	1.0704E-004	1.0762E-002	2.9982E-002	9.6213E-011	4.4205E-013	**7.8201E-015**	**3.5831E-015**
f_4	10	3.4038E+001	8.2219E+000	3.2883E+001	4.4651E+000	5.4847E-002	0	**4.6925E-016**	**7.2879E-017**
	30	1.2950E+002	2.1981E+001	7.5816E+001	3.8504E+000	9.1024E-002	4.0101E-002	**8.6322E-003**	**1.1363E-004**
f_5	10	5.2318E-002	4.3801E-003	4.3676E-002	8.2209E-003	0	0	**0**	**0**
	30	6.7837e-001	4.0722e-001	5.4479e-001	5.8425e-002	0	0	**0**	**0**

Table 3. Results (mean ± std) of MLCSA and other state-of-the-art evolutionary algorithms when D=30.

Funs	D	BCSA	MOEA/D/BL	HLCSA	DMDE	MLCSA
		Mean±Std	Mean±Std	Mean±Std	Mean±Std	Mean±Std
f_1	2	2.8182E+000±3.4013E+000	4.3122E+003±3.5796E+001	**0±0**	1.7276E+000±2.8823E+000	**0±0**
f_2	2	4.1845E-001±2.5478E-016	5.1041E-001±7.4544E-004	7.4982E-001±1.6283E-018	**0±0**	3.8732E-001±1.2086E-021
f_3	10	4.9882E-015± **0**	5.1314E-014±3.5846E-016	2.9514E-013±2.7069E-015	7.5102E-009±7.8325E-011	**3.0295E-015**±2.6618E-016
	30	1.1640E-011±3.5996E-013	5.8612E-010±6.7021E-013	8.4078E-012± **0**	7.0048E-013±4.8912E-015	**8.0576E-015**±3.2762E-015
f_4	10	4.3070E-015±2.6925E-014	4.6602E-015±6.4486E-014	3.0524E-013±9.3926E-015	3.5127E-014±8.2676E-015	**4.3811E-016±7.2761E-017**
	30	**0±0**	8.1793E-001±2.4890E-004	**0±0**	4.0305E-005±1.6402E-007	4.4046E-002±7.2603E-005
f_5	10	**0±0**	**0±0**	**0±0**	**0±0**	**0±0**
	30	**0±0**	3.5018E-002±3.5737E-001	7.4027E-002±2.4759E-001	6.2408E-001±9.9214E-001	**0±0**

Utilizing the multi-learning mechanism, MLCSA successfully avoids antibodies falling into deep local optimal solutions and effectively guide the evolutionary process towards the global optima.

3. DEEP NETWORK BASED ON STACKED I-ELM/MLCSA AUTOENCODERS(D-ELMS-AE/MLCSA)

3.1. Incremental extreme learning machine (I-ELM)

Extreme learning machine (ELM) proposed by Huang et al. [22] is a special type of single-hidden layer feedforward networks (SLFNs) with randomly generated additive or RBF hidden nodes and hidden node parameters, which has recently been extensively studied by many researchers in various areas of scientific research and engineering due to the excellent approximation capability [23]. Incremental extreme learning machine, termed as I-ELM, is proposed by Huang et al. [24], which randomly adds nodes to the hidden layer one by one and freezes the output weights of the existing hidden nodes when a new hidden node is added. The output weights of the new added node are calculated by a simple formula analytically. I-ELM is fully automatic in that there is no need to intervene the learning process by tuning control parameters manually for users except for target errors and the allowed maximum number of hidden nodes. But, there still exists some issues to be tackled.

The motivation for the work in this section comes from the selection of $\Omega_N(a_L,b_L)$ that there are some input weights a_L and biases b_L at each learning step which make residual error reduce more. Therefore, our aim is to propose a simple improved implementation of ELM in order to find $\Omega_N(a_L,b_L)$, meanwhile, to achieve a more compact network architecture. In this section, we propose an improved I-ELM algorithm (I-ELM/MLCSA) based on Multi-learning CSA, and prove the I-ELM/MLCSA algorithm in theory.

3.2. Proposed I-ELM based on multi-learning CSA (I-ELM/MLCSA)

In this section, we propose an improved I-ELM algorithm (I-ELM/MLCSA) based on Multi-learning CSA, and prove the I-ELM/MLCSA algorithm in theory. Moreover, we construct the deep architectures with I-ELM/MLCSA, and obtain the better performance than other state-of-the-art deep models.

Given N arbitrary distinct samples $\{(x_i,t_i) \mid x_i \in R^n, t_i \in R^m, 1 \leq i \leq N\}$, activation function $G(x)$, number L of hidden nodes, expected learning accuracy η,the maximum number of hidden nodes L_{max}.The details of I-ELM/MLCSA are described in Algorithm 2.

We design the I-ELM/MLCSA with the parameters a_i and b_i as the variables which need to be optimized, where $i = 1,2,..,n$. The objective function is the expected residual error of network η. The population size is N, and antibody population is randomly generated. Meanwhile, we conduct the encoding operation to all the antibodies within population in binary format. And then, we can calculate the best hidden parameters a_i and b_i according to the multi-learning strategy. Increase the number of hidden nodes by one or more: $L = L + \lambda$, generate the antibody population with the population size M. Obtain the approximate optimum solution by cross-validation, until the residual error of network is equal to the termination criterion, the process of multi-learning clonal selection is stop. The encoding principle of network parameters based on MLCSA is shown in Fig. 1.

3.3. Implementation of stacked I-ELM /MLCSA autoencoders in deep network

As a branch of machine learning based on a set of algorithms, deep learning can obtain the high level cognition of distributed data structure and abstractions for the things, which stimulates multiple processing layers with complex structures of human brain to make decisions and acquire the deep-level mode of thinking. The Autoencoder is a kind of unsupervised neural networks, which is frequently applied in the deep learning building blocks as an effective method, using the qualification of the input of network is equal to the output, then fine-tune the parameters of the whole network [25, 26]. The ELM autoencoder (ELM-AE) proposed by Kasun et al. [27] is constituted of input layer, hidden layer and output layer. The input data is projected to a different or equal dimension space [28], the expressions are following:

$$h = \sigma(a \cdot x + b), a^T a = 1, b^T b = 1, \tag{5}$$

$$h(x_i)V = x_i^T, i = 1,2,...,N, \tag{6}$$

Algorithm 2 I-ELM based on Multi-learning CSA (I-ELM/MLCSA)

```
Stage 1: Calculate the best hidden nodes $L_{best}$
Step 1: Initialization: Randomly generate the initial antibody
    population $A(t)$, set $N = 0$ and initialize the termination
    criterion. Let $L = 0$, $\Omega_N(a_i,b_i) = A$ and residual error
    $E = t$, where $t = [t_1,t_2,...,t_N]^T$;
Step 2: While $L < L_{max}$, $\|E\| < \eta$, $L = L+1$;
Step 3: According to the affinity, generate amplificatory $X(t)$
    from $A(t)$ with clonal proliferation operator, and
    scale of clone is a monotone increasing function of
    antibody affinity;
Step 4: Multi-learning operation:
Step 4.1: Apply Baldwinian learning strategy on every
    antibody in $X(t)$, generate the antibody population $Y(t)$;
Step 4.2: Apply Lamarckian learning strategy on every
    clonal antibody in $Y(t)$, generate the population $Z(t)$;
Step 5: Calculate the affinity of each antibody in $Z(t)$;
Step 6: Generate the new antibody population $A(t+1)$ by
    applying clonal selection operator to $D(t)$;
Step 7: If termination criterion is satisfied, stop the algorithm
    and obtain antibody with the highest affinity in $A(t+1)$
    as the output; Otherwise, $t = t+1$, go to Step 2;
Step 8: Calculate the output weight $a_L^*$ and residual error for
    the new hidden nodes:
    $\beta = \frac{E_L \cdot H_L^T(\Omega_L^*)}{H_L(\Omega_L^*) \cdot H_L^T(\Omega_L^*)}$, $E_L = E_{L-1} - \beta_L^* \parallel H_L(\Omega_L^*), x \parallel$;
Step 9: If criteria is satisfied $E_L < \eta$, the process of multi-
    learning is terminated. Otherwise, go to Step 3;
Stage 2: Online calculate the output of predictions $\hat{Y}_{test}$
Step 10: Given $G = [K_{ELM} + \frac{1}{C}]$ , for time $t+1$: $G_{t+1}$;
Step 11: The incremental form can be used to update the
    matrix inverse process $G_{t+1}^{-1}$ ;
Step 12: Update and output the $\hat{Y}_{test}$:
    $\hat{Y}_{test} = \begin{bmatrix} K(x_{test1},x_1) & \cdots & K(x_{test1},x_M) \\ \vdots & \ddots & \vdots \\ K(x_{testN},x_1) & \cdots & K(x_{testN},x_M) \end{bmatrix} \begin{bmatrix} A_M^{-1}y_1 \\ \vdots \\ A_M^{-1}y_M \end{bmatrix}.$
Endwhile
```

where $a = [a_1,...,a_L]^T$ are the weights generated orthogonally randomly, and $b = [b_1,...,b_L]^T$ are the biases generated orthogonally randomly between the input and hidden nodes. There are three calculation approaches to obtain the output weight β of ELM-AE:

1) For sparse ELM-AE representations, output weights β can be calculated as follows:

$$\beta = (\frac{I}{C} + H^T H)^{-1} H^T X. \tag{7}$$

2) For compressed ELM-AE representations, output weights β can be calculated as follows:

$$\beta = H^T(\frac{I}{C} + H^T H)^{-1} X. \tag{8}$$

3) For equal dimension ELM-AE representations, output weights β can be calculated as follows:

$$\beta = H^{-1}X. \tag{9}$$

In this section, we would like to use the idea of autoencoder to constitute deep learning architecture with improved I-ELM, which incorporated with I-ELM autoencoder based on multi-learning clonal selection algorithm (MLCSA) and deep architecture. The stop criterions of autoencoder process is residual error E is equal to the expected learning accuracy ε or the number of hidden nodes L achieves L_{max} are met. The model structure of I-ELM-AE /MLCSA can randomly control the number of the nodes without the computation accuracy.

Given a training set $N = \{(x_i,t_i)|x_i \in R^n, t_i \in R^m, i = 1,2,...,N\}$, where $x_i = (x_{i1},x_{i2},...,x_{in})$,$t_i = (t_{i1},t_{i2},...,t_{im})$, activation function $\sigma(x)$, maximum number of hidden nodes in single layer L_{max}, the input data is reconstructed at the output layer through the following function: $\sum_{i=1}^{l} \beta_i \sigma(a_i \cdot x_j + b_j) = x_j, j = 1,2,\dots,L$, the output weight can be obtained with the following:

$$\beta = \frac{E_L \cdot H_L^T(\Omega_L^*)}{H_L(\Omega_L^*) \cdot H_L^T(\Omega_L^*)} \tag{10}$$

$$E_L = E_{L-1} - \beta_L^* \parallel H_L(\Omega_L^*), x \parallel \tag{11}$$

where $a_i = [a_{i1},a_{i2},...,a_{id}]^T$ is the randomly generated input weight, Ω_L^*denotes the best hidden parameters $\Omega_L^*(a_i,b_i)$ and $x_i = (x_{i1},x_{i2},...,x_{in})^T$ are the input and output of the I-ELM-AE /MLCSA.

The D-ELMs-AE/MLCSA proposed in this paper utilize the methodology of deep learning to construct the deep architecture. Different from the deep model in ML-ELM [25], D-ELMs-AE/MLCSA uses the individual improved I-ELM (I-ELM/MLCSA) as the block, which can be regarded as one layer of the multi-hidden layers. Instead of "randomly" generating only some parameters of the hidden nodes based on the distribution of the training data or some probability space determined by the target functions f in each layer of ML-ELM, the D-ELMs-AE/MLCSA inherits the advantages of incremental constructive feedforward networks model, which does not recalculate the output weights of the existing hidden nodes after a new hidden node is added. In other words, once the output weights of hidden nodes are calculated they will remain frozen and will not be changed any more. Meanwhile, the algorithm proposed retain the abilities of deep learning algorithms on exactly capturing higher-level abstractions and characterizing the data representations mapped to I-ELM /MLCSA feature space , in each layer, I-ELM-AE/MLCSA output weights with respect to input data are the weight of the first layer, for the same reason, the output weights of I-ELM-AE/MLCSA, with respect to hidden layer output are the layer weights of D-ELMs-AE/MLCSA.

Algorithm 3 I-ELM-AE/MLCSA in deep network (D-ELMs-AE/MLCSA)

```
Stage 1: I-ELM-AE/MLCSA on layer 1
Step 1: Initialization: Randomly generate the initial antibody
    population $A(t)$, set $N = 0$ and initialize the termination
    criterion. Let $L = 0$, $\Omega_N(a_i,b_i) = A$ and residual error
    $E = t$, where $t = [t_1,t_2,...,t_N]^T$;
Step 2: While $L < L_{max}$, $\|E\| < \eta$, $L = L+1$;
Step 3: According to the affinity, generate amplificatory $X(t)$
    from $A(t)$ with clonal proliferation operator, and scale
    of clone is a monotone increasing function of antibody
    affinity;
Step 4: Multi-learning operation:
Step 4.1: Apply Baldwinian learning strategy on every anti-
    body in $X(t)$, and generate antibody population $Y(t)$;
Step 4.2: Apply Lamarckian learning strategy on every
    clonal antibody in $Y(t)$, and generate the population
    $Z(t)$;
Step 5: Calculate the affinity of each antibody in $Z(t)$;
Step 6: Generate the new antibody population $A(t+1)$ by
    applying clonal selection operator to $D(t)$;
Step 7: If termination criterion is satisfied, stop algorithm
    and obtain antibody with the highest affinity in $A(t+1)$
    as the output; Otherwise, $t = t+1$, go to Step 2;
Step 8: Calculate the output weight $a_L^*$ and residual error for
    the new hidden nodes with (10), (11);
Step 9: If criteria is satisfied $E_L < \eta$, the process of multi-
    learning is terminated. Otherwise, go to Step 3;
Step 10: Given $G = [K_{ELM} + \frac{1}{C}]$ , For time $t+1$:$G_{t+1}$;
Step 11: The incremental form can be used to update the
    matrix inverse process $G_{t+1}^{-1}$ ;
Step 12: Update and output the $\hat{Y}_{test} = [y_1,y_2,...,y_M]^H$
Stage 2: I-ELM-AE/MLCSA on layer $2 \to N$
Step 13: While $L_{2\to N} < L_{max}$, $\| E \| > \varepsilon$;
Step 14: Calculate the output weight $\beta^*_{(2\to N)L}$ for newly
    added hidden node with the hidden layer output matrix:
    $H_{(2\to N)L}(\Omega_L^*)$: $\beta^*_{(2\to N)L} = \frac{E_L \cdot H^T_{(2\to N)L}(\Omega_L^*)}{H_{(2\to N)L}(\Omega_L^*) \cdot H^T_{(2\to N)L}(\Omega_L^*)}$;
Step 15: Calculate the residual error after adding the new
    hidden node $L_{2\to N}$:
    $E_L = E_{L-1} - \beta^*_{(2\to N)L} \parallel H_{(2\to N)L}(\Omega_L^*), x \parallel$.
Endwhile
```

As shown in Fig. 2, the interesting input data distributions in the reconstruction matrix β are then retained for data pretraining in the new I-ELM, instead of randomly generating input weights, β^T is used as the input weights. As demonstrated in [25], the output weight β can learn to represent the input data via singular values and perform better than manually calculated SVD basis. Therefore, β^T can be used for unsupervised pretraining of the data, and will likely result in better generalization performance when solving large unstructured data problems.

Given N arbitrary distinct samples $\{(x_i,t_i) \mid x_i \in R^n, t_i \in$

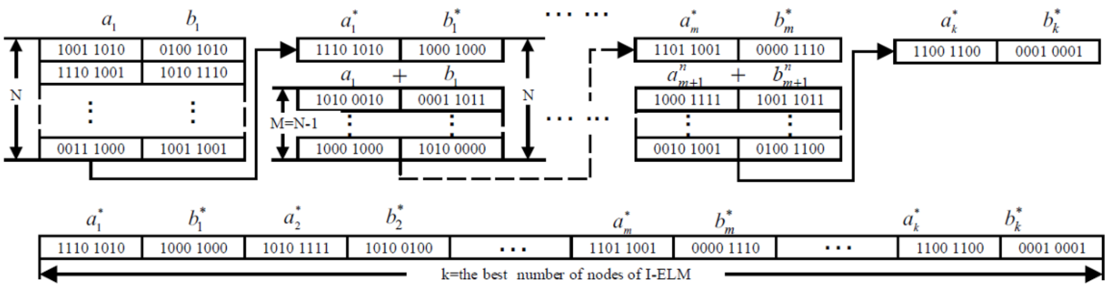


Fig. 1. The encoding principle of network weights based on MLCSA.

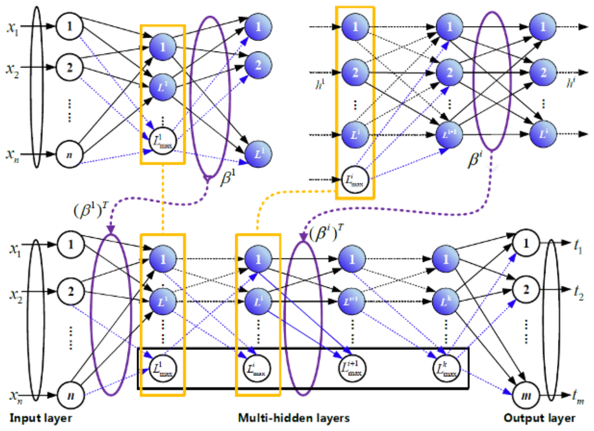


Fig. 2. The model structure of D-ELMs-AE/MLCSA.

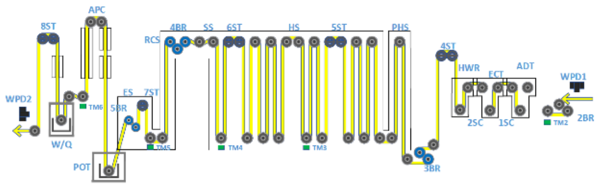


Fig. 3. General scheme of strip steel in annealing process.

Table 4. Specification of 22 input variables.

No.	Variables	Description
x_1	Speed	Unit Speed
x_2	Measure	Steel-Strip Width
x_3	Measure	Steel-Strip Speed
x_{4-8}	Temp	SS, RCS, ES, TOP Temperature
x_{9-11}	Tension	SS, HS, RCS Section Tension
x_{12-16}	Speed	$2-6BR_1$ Roller Speed
x_{17-21}	Speed	$2-6BR_2$ Roller Speed
x_{22}	Measure	Elongation Value of Steel-Strip

$R^m, 1 \le i \le N\}$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$, activation function $G(x)$, number L of hidden nodes, the maximum number of hidden nodes in single layer L_{max}, expected learning accuracy η. The detailed algorithm of D-ELMs-AE/MLCSA is shown in Algorithm 3.

4. CASE STUDY ON ELONGATION PREDICTION OF STRIPS

4.1. Simulations

The process of continuous annealing as shown in Fig. 3. The annealing treatment is considered the most important process to steel-strips, in which the steel-strips can eliminate the cold working hardening and internal stress, meanwhile, reduce the hardness of steel-strips. Furthermore, the process also can improve the ability of plastic deformation, stamping and technique mechanical. However, during the annealing process, the dual physical and chemical changes can make the steel-strips extend or shorten, which pass through the preheating section (PHS), radiation heating section (HS), slow cooling section (SCS), rapid cooling section (RCS) and other temperature sections with the tension action of rollers in the furnace [9, 10]. Moreover, the surface friction coefficient and the rotational speed of the tension rolls also affect the elongation of strips, which cause the weld position unable to be tracked inaccurately. It have been demonstrated that the accurate degree of computational position has a great influence on the rate of finished product and the safety of air-knife.

Based on the recent surveys, more than 3.2% steel-strips are squandered on avoiding the welding line in every year, because of the restricted condition which can afford no space to install any equipment to detect the operation position of welding line. And, the whole cost resulting from all the annealing furnace units is also significant economic losses. Therefore, in order to keep high efficiency and ensure security of equipment during the operation, an accurate elongation of steel-strips predicting and monitoring is quite desired.

In this section, all of the experimental results for the elongation of strips prediction are presented. In the processing of steel-strips annealing, there are many variables

Table 5. Results of D-ELMs-AE/MLCSA and other algorithms on different datasets.

Datasets	Algorithms	Training Accuracy (%)				
		1-day data	5-days data	10-days data	30-days data	50-days data
5-months	DNN	84.6153	83.4428	80.3884	78.0972	78.1212
	ML-ELM	85.9782	85.9347	82.4276	80.1652	79.1304
	DSVDD	85.5604	85.1796	83.3368	81.9831	79.8136
	AE-S-ELMs	87.7848	87.7788	86.7805	85.8688	85.2764
	PWDNN	87.6377	86.7561	86.5472	84.3189	84.1935
	D-ELMs-AE/MLCSA	**90.8869**	**90.7612**	**89.5693**	**88.1478**	**87.6385**
10-months	DNN	92.9401	92.4049	90.2953	89.8073	89.3317
	ML-ELM	93.9519	93.8371	93.4739	92.5638	91.1230
	DSVDD	92.5011	92.1914	91.9283	91.0261	88.2732
	AE-S-ELMs	94.2066	94.0551	93.3980	92.6732	92.1247
	PWDNN	94.2950	94.2464	93.8203	93.3506	93.1745
	D-ELMs-AE/MLCSA	**96.1126**	**96.0171**	**95.9739**	**95.3195**	**94.8167**
20-months	DNN	93.6779	93.2975	93.1823	92.5348	92.3274
	ML-ELM	94.3827	94.3039	94.2673	93.7995	93.2501
	DSVDD	94.5590	94.4180	94.2365	94.1173	94.0035
	AE-S-ELMs	99.0397	98.9829	98.5190	**98.4339**	97.4772
	PWDNN	97.2004	97.1311	96.8446	96.7177	95.9238
	D-ELMs-AE/MLCSA	**99.5161**	**99.3272**	**99.1560**	98.4240	**98.3975**

that can be used in D-ELMs-AE/MLCSA as the inputs. As shown in Table 4, the dominant factors include four kinds of actual measured values (22 input variables containing the speed of unit and roller, the dimensions of steel-strips, the temperature values of every section and the tension values of every section) which are recorded once by one second, the specific descriptions are listed in Table 4. To verify the availability of algorithm we presented, we adopt other six algorithm based on deep architecture to compare the performance with D-ELMs-AE/MLCSA. The historical data which are used in simulations contain the last 22 months data, which can affect the position of the welding seam. Moreover, because of the continuity of the annealing process, we must establish the model which can meet the computation requirement for different categories of steel-strips specifications. The specific results are shown in Table 5, and the simulations are obtained by the average of 30 trails. The entire following simulations are conducted by MATLAB 2013a environment running on the Windows 7 standard desktop with at 128 GB of memory and Intel Xeon E5 2620V2 (2.1GHZ) processor.

The specific analyses on the results of testing accuracy are shown in Table 5. It can be seen that D-ELMs-AE/ML CSA perform better than ML-EM and AE-S-ELMs, which build the deep architecture with ELM algorithm, while the performances of deep models (i.e., DNN, DSVDD, PWDNN) are comparable. Using the 2 months data as training dataset, the D-ELMs-AE/MLCSA can obtain the testing accuracy with 81.5385%(1-day), 79.4898% (5-day), 74.8813% (10-day), 71.5536%(30-day), 65.0590% (50-day). Actually, there are some difficulties to generate the prediction functions when on a training dataset with limitations. Although features representations are difficult to be yielded with the insufficient input, the experimental results still show that D-ELMs-AE/MLCSA outperforms other deep learning algorithms, which appear to demonstrate that D-ELMs-AE/MLCSA suitable for prediction tasks of steel-strips elongation. Compared with DNN, ML-ELM, DSVDD, AE-S-ELMs and PWDNN on 5-months data, 10-months data and 20-months data described in Table 5, the deep learning architecture we proposed can consistently show the better performances than others.

4.2. Simulations in faulty conditions

Generally speaking, the efficacy of simulations depends on the fruitfulness of the datasets and, by using more comprehensive datasets. However, in practical engineering, the lack of production data or fault data which are generated in process operating faults, can cause big error of prediction results which highly lead to serious consequences. Hence, to evade possible biased comparisons with consistently sufficient production data, in this experiment, we use the datasets which lack some dominant input variables in various degrees to training data, and different sized datasets (5/10/20-months datasets) are employed for the different experiments to verify the feasibility and performance of the algorithms to insufficient input variables. Details of the three training datasets and the descriptions are listed in Table 6, where the 20-days training data and

Table 6. Specification of PG-1, PG-2 and PG-3 datasets.

Datasets	Training Sample	Testing Sample	Attributes	Description
PG-1	94420/ 188230/ 389722 (5 / 10 / 20 months)	8722	21	training dataset without $2BR_2$ roller speed and 20-days testing dataset
PG-2		8406	21	training dataset without RCS section temperature and 20-days testing dataset
PG-3		18406	20	training dataset without $3BR_1$ roller speed and SS section tension, 40-days testing dataset

Table 7. Results of D-ELMs-AE/MLCSA and other algorithms on PG-1, PG-2 and PG-3 datasets.

Datasets	Algorithms	PG-1		PG-2		PG-3	
		Accuracy (%)	Deviation(%)	Accuracy (%)	Deviation(%)	Accuracy (%)	Deviation(%)
5-months	DNN	77.1291	0.1798	72.4977	0.8825	69.2022	1.2744
	ML-ELM	80.4208	1.0151	72.7867	0.2337	69.5689	0.5075
	DSVDD	81.6426	0.7485	80.3125	0.3745	78.6223	1.0375
	AE-S-ELMs	83.6967	0.0566	81.6897	**0.2089**	81.3137	0.4232
	PWDNN	84.8939	0.2209	83.5701	0.5785	82.4312	0.8479
	D-ELMs-AE/MLCSA	**87.0107**	**0.0382**	**86.9569**	0.2398	**86.5346**	**0.2269**
10-months	DNN	82.9069	0.2442	81.9169	1.0553	81.5971	0.7106
	ML-ELM	84.5934	0.4544	83.7914	1.1194	83.7739	**0.3426**
	DSVDD	85.3329	0.2985	84.1361	1.0824	83.4454	0.4838
	AE-S-ELMs	88.6904	0.2454	87.7503	0.8056	86.7931	0.9642
	PWDNN	87.0120	0.3197	85.7699	0.7496	84.5309	1.2367
	D-ELMs-AE/MLCSA	**93.9306**	**0.0112**	**92.8877**	**0.4109**	**90.9484**	0.4303
20-months	DNN	92.1317	0.2346	91.1278	0.3910	90.6629	0.6789
	ML-ELM	92.5362	0.4909	92.0638	1.1446	91.2070	1.1887
	DSVDD	93.8825	0.2907	92.9413	0.4932	92.7363	0.7495
	AE-S-ELMs	97.7902	0.1016	95.5267	0.2837	95.0160	0.3623
	PWDNN	96.1107	0.6215	94.7743	0.7045	94.2840	1.2460
	D-ELMs-AE/MLCSA	**97.9759**	**0.0734**	**97.9032**	**0.2330**	**96.87686**	**0.3107**

testing data in PG-1 and PG-2 respectively are different.

All experiments are performed on the DNN, ML-ELM, DSVDD, AE-S-ELMs, PWDNN and D-ELMs-AE/MLC SA shown in Table 7. The best results are shown in boldface. The results obtained with 5-months Training Data, 10-months Training Data and 20-months Training Data, respectively, show that the simulation results demonstrate the D-ELMs-AE/MLCSA is superior to DNN, ML-ELM, DSVDD, AE-S-ELMs and PWDNN on testing accuracy and standard deviation with 50 repetitions. From the overall results, the testing accuracy of D-ELMs-AE/MLCSA are 87.0107%, 93.9306%, 97.9759% (PG-1), 86.9569%, 92.8877%, 97.9032% (PG-2), 86.5346%, 90.9484% and 96.87686%(PG-3), better than those of DNN, ML-ELM, DSVDD, AE-S-ELMs and PWDNN, respectively. Its deviation with 50 runs is also smaller than those of DNN, ML-ELM, DSVDD, AE-S-ELMs and PWDNN in almost all cases, showing that the performance of D-ELMs-AE /MLCSA is quite stable.

For further investigation on the prediction capabilities of the potential D-ELMs-AE/MLCSA, in Fig. 4, the performances of algorithms are evaluated in terms of four criteria, i.e., mean absolute error (MAE), mean square error (MSE), root mean square error (RMSE) and mean absolute percentage error (MAPE). MAE in Fig. 4(a) evaluates the disparity between the real elongation of steel-strips and the predicted values. This function is more robust to the large errors than the other functions, meanwhile, MAPE and MSE in Fig. 4(b) and in Fig. 4(c), respectively, also reflect the dispersion of models. However, they are sensitive to the large errors compared with MAE because the errors are squared and the large errors are amplified further. RMSE in Fig. 4(d) is the ratio between errors and real elongation of steel-strips. It can be considered as a relative error function. These four functions can be used to measure the performances of the prediction algorithms

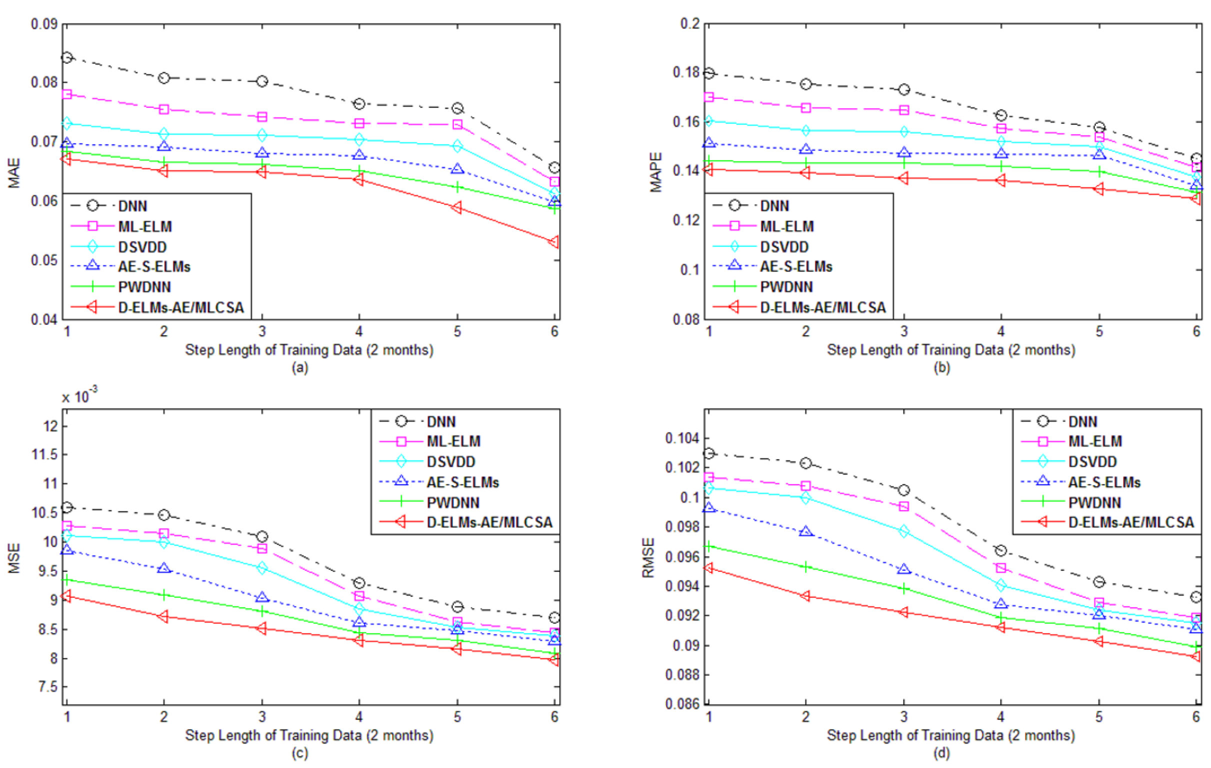


Fig. 4. Different prediction errors of six models on 10-months data.

from different viewpoints. By analyzing the results, it can be seen apparently that the comparisons for four criteria based on 12 months training data which are selected randomly from 20-months dataset (PG-3), indicate that the prediction performances of D-ELMs-AE/MLCSA outperform other algorithms, therefore, the algorithm we proposed can be applied effectively in practical engineering.

Fig. 5 depicts the prediction values and correlation of the deep architecture methods with 6 modules, respectively. The three figures in Fig. 5 describe simulations with different training and testing datasets which are randomly selected from PG-1, PG-2 and PG-3, furthermore, the sizes of testing dataset are 917, 883 and 937, respectively. Those figures help us to clearly evaluate the performance of each module as well as the overall output of the entire network. As it can be observed from figures, the D-ELMs-AE/MLCSA can show more excellent prediction results than algorithms.

5. CONCLUSIONS

In this paper, we proposed a deep architecture based on I-ELM-AE/MLCSA algorithm, called D-ELMs-AE/MLC SA, and demonstrated the efficacy of D-ELMs-AE/MLC SA with the elongation predictions of steel-strips in annealing furnace with different actual production conditions. The simulations were conducted to discern the prediction accuracy of the proposed algorithm. The identification capability of the resulting method together with its regression performance were compared with different state-of-the-art methods. In the case of insufficient inputs, the D-ELMs-AE/MLCSA proposed also can show the more outstanding performance than DNN, ML-ELM, DSVDD, AE-S-ELMs and PWDNN. In this work, the proposed algorithm was employed in several months in practical engineering with different production conditions to make sure that it can guarantee the safety of air-knife. The numerical experiments indicated that D-ELMs-AE/MLCSA can reliably focus on both the security of equipment and economy objectives, and also, make a deliberate trade-off between accuracy of prediction and operation time. In summary, the results indicate the performance advantages of D-ELMs-AE/MLCSA prone it to be reliably used for real-time implementations in annealing process. In the future, we will expand further research on the stability and arithmetic speed of D-ELMs-AE/MLCSA to ensure the better performance in other engineering.

REFERENCES

[1] X. Lu, Z. Lin, H. Jin, and J. Yang, "Rating pictorial aesthetics using deep learning," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1-1, 2015.

[2] Y. J. Park and M. Kellis, "Deep learning for regulatory genomics," *Nature Biotechnology*, vol. 3, no. 8, pp. 825-826,

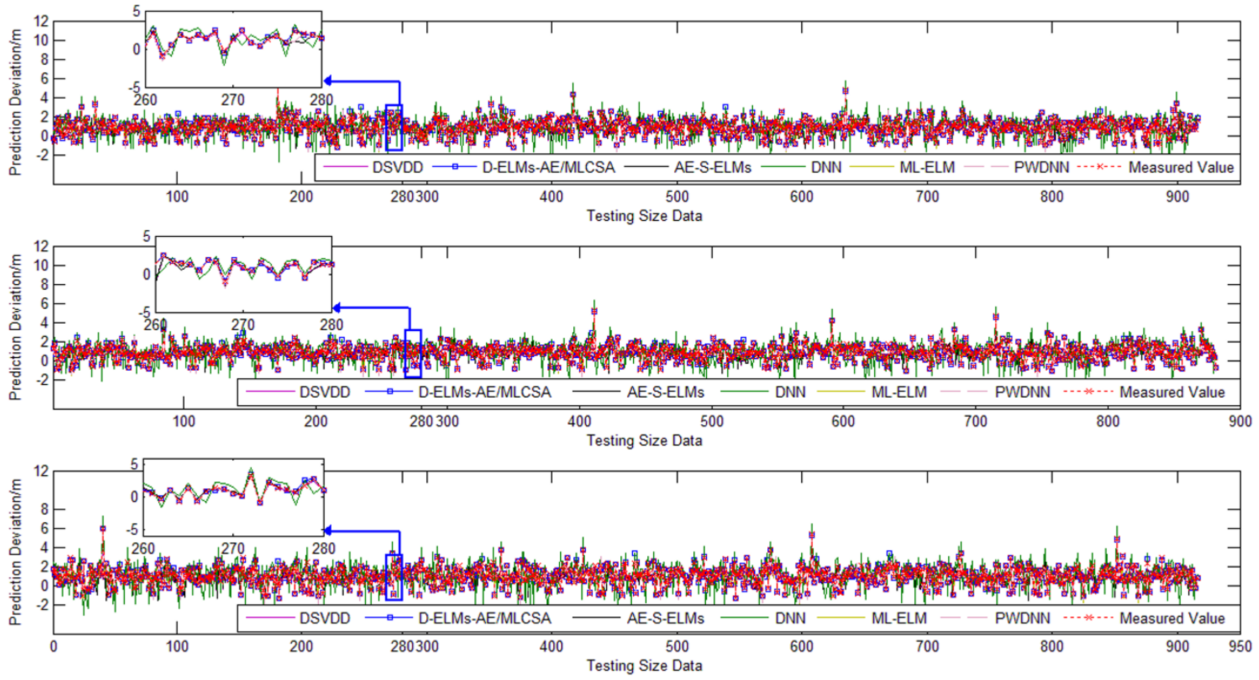


Fig. 5. The comparisons of prediction on PG-1, PG-2 and PG-3 datasets.

2015.

[3] H. I. Suk, C. Y. Wee, S. W. Lee, and D. Shen, "State-space model with deep learning for functional dynamics estimation in resting-state fMRI," *Neuroimage*, vol. 129, no. 2016, pp. 292-307, 2016. [click]

[4] H. M. Zhou, G. B. Huang, Z. P. Lin, H. Wang, and Y. C. Soh, "Stacked extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp.1, 2014.

[5] J. Tang, C. Deng, and G. B. Huang. "Extreme learning machine for multilayer perceptron," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 27, no. 4, pp. 809-821, 2015.

[6] H. Liu, Y. Liu, and F. Sun. "Robust exemplar extraction using structured sparse coding," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 26, no. 8, pp. 1816-1821, 2015.

[7] Y. Zhang, X. Li, Z. Zhang, F. Wu, and L. Zhao, "Deep learning driven blockwise moving object detection with binary scene modeling," *Neurocomputing*, vol. 168, no. 2015, pp. 454-463, 2015. [click]

[8] Y. Gu, Y. Chen, J. Liu, and X. Jiang, "Semi-supervised deep extreme learning machine for Wi-Fi based localization," *Neurocomputing*, vol. 166, no. 2015, pp. 282-293, 2015. [click]

[9] C. L. Wen, D. X. Wu, H. S. Hu, and W. Pan, "Pose estimation-dependent identification method for field moth images using deep learning architecture," *Biosystems Engineering*, vol. 136, no. 2015, pp. 117-128, 2015. [click]

[10] Q. H. Hu, R. J. Zhang, and Y. C. Zhou, "Transfer learning for short-term wind speed prediction with deep neural networks," *Renewable Energy*, vol. 85, no. 2016, pp. 83-95, 2016. [click]

[11] C. Wang, J. H. Wang, S. S. Gu, and Y. X. Zhang, "A soft sensor based on optimized LSSVM for elongation prediction of strip steel," *Journal of Northeastern University (Natural Science)*, vol. 36, no. 8, pp. 1084-1088, 2015. [click]

[12] Y. J. Choi and M. C. Lee, "PID sliding mode control for steering of lateral moving strip in hot strip rolling," *International Journal of Control, Automation, and Systems*, vol. 7, no. 3, pp. 399-407, 2009. [click]

[13] F. M. Burnet, *The Clonal Selection Theory of Acquired Immunity*, Cambridge University Press, 1959.

[14] F. M. Burnet, *Clonal Selection and After*, Theoretical immunology, Marcel Dekker, New York, pp. 63-85, 1978.

[15] L. N. D. Castro and F. J. V. Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239-251, 2002.

[16] M. Gong, L. Jiao, and L. Zhang, "Baldwinian learning in clonal selection algorithm for optimization," *Inform. Sci*, vol. 180 no. 8, pp. 1218-1236, 2010. [click]

[17] M. G. Gong, L. C. Jiao, J. Yang, and L. Fang "Lamarckian learning in clonal selection algorithm for numerical optimization," *International Journal of Artificial Intelligence Tools*, vol. 19, no. 1, pp. 19-37, 2010.

[18] Y. Wang, J. Z. Zhou, H. Qin, and Y. L. Lu, "Improved chaotic particle swarm optimization algorithm for dynamic economic dispatch problem with valve-point effects," *Energy Conversion and Management*, vol. 51, no. 12, pp. 2893-2900, 2010. [click]

[19] X. L. Ma, F. Liu, Y. T. Qi, and L. L. Li, "MOEA/D with Baldwinian learning inspired by the regularity property of

continuous multiobjective problem," *Neurocomputing*, vol. 145, no. 18, pp. 336-352, 2014.

[20] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341-359, 1997. [click]

[21] Y. Peng and B. L. Lu, "Hybrid learning clonal selection algorithm," *Information Sciences*, vol. 296, no. 19, pp. 128-146, 2015. [click]

[22] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol.70, no. s 1-3, pp. 489-501, 2006. [click]

[23] X. Luo and X. H. Chang, "A novel data fusion scheme using grey model and extreme learning machine in wireless sensor networks," *International Journal of Control, Automation, and Systems*, vol. 13, no. 3, pp. 539-546, 2015. [click]

[24] G. B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans Neural Netw*, vol. 17, no. 4, pp. 879-892, 2006.

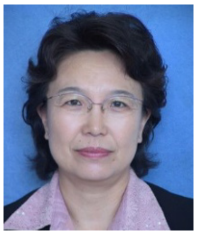
[25] P. Li and G. H. Yang, "An adaptive fuzzy design for fault-tolerant control of MIMO nonlinear uncertain systems," *Journal of Control Theory & Applications*, vol. 9, no. 2, pp. 244-250, 2011. [click]

[26] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1-127, 2009. [click]

[27] L. L. C. Kasun, H. Zhou, G.B. Huang, and C. M. Vong, "Representational Learning with ELMs for Big Data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31-34, 2013.

[28] W. Johnson and J. Lindenstrauss, "Extensions of Lipschitz maps into a Hilbert space," *Contemporary Mathematics*, vol. 26, pp. 189-206, 1984.

Chao Wang received his B.S. and M.S. degrees in Electrical Engineering from Shenyang Jianzhu University, China, in 2008 and 2011, respectively, where he is currently pursuing a Ph.D. degree in Northeastern University, China. His research interests include computational intelligence, intelligent control, and machine learning.


Jian-Hui Wang received her B.S., M.S., and Ph.D. degrees in Electrical Engineering from Northeastern University, China, in 1982, 1986, and 1999, respectively. Her research interests include intelligent control theory and its application.

Shu-Sheng Gu received his B.S. in Automation from Northeastern University, China, in 1969. His research interests include intelligent control theory and its application.

Xiao Wang received the B.S in automation from the college of information science and engineering, Northeastern University of China in 2013, where he is pursuing the Ph.D. degree in control theory and engineering. His research interests include the applications of advanced controls in wind turbine system, and the transient stability of the power system related to large-scale wind power integrations.

Zhang Yuxian received his Ph.D degree in 2007 from Northeastern University, and finished his postdoctoral study in 2009 from Tsinghua University. Now, he is an associate professor in Shenyang University of Technology. His main research interests include intelligent control, optimal control and data mining for hybrid data.