Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 3049632, 10 pages
http://dx.doi.org/10.1155/2016/3049632

Research Article

Deep Convolutional Extreme Learning Machine and Its Application in Handwritten Digit Classification

Shan Pang[1] and Xinyi Yang[2]

[1]*College of Information and Electrical Engineering, Ludong University, Yantai 264025, China*
[2]*Department of Aircraft Engineering, Naval Aeronautical and Astronautical University, Yantai 264001, China*

Correspondence should be addressed to Shan Pang; pangshanpp@163.com

Received 27 April 2016; Revised 7 July 2016; Accepted 19 July 2016

Academic Editor: Stefano Squartini

In recent years, some deep learning methods have been developed and applied to image classification applications, such as convolutional neuron network (CNN) and deep belief network (DBN). However they are suffering from some problems like local minima, slow convergence rate, and intensive human intervention. In this paper, we propose a rapid learning method, namely, deep convolutional extreme learning machine (DC-ELM), which combines the power of CNN and fast training of ELM. It uses multiple alternate convolution layers and pooling layers to effectively abstract high level features from input images. Then the abstracted features are fed to an ELM classifier, which leads to better generalization performance with faster learning speed. DC-ELM also introduces stochastic pooling in the last hidden layer to reduce dimensionality of features greatly, thus saving much training time and computation resources. We systematically evaluated the performance of DC-ELM on two handwritten digit data sets: MNIST and USPS. Experimental results show that our method achieved better testing accuracy with significantly shorter training time in comparison with deep learning methods and other ELM methods.

1. Introduction

Extreme learning machine is a novel learning algorithm for general single-hidden-layer neural networks proposed by Huang et al. [1]. In ELM, the input weights and hidden biases are randomly generated, and the output weights are analytically determined by regularized least square method, providing a simple deterministic solution. There are no iterations and parameters tuning as in back propagation (BP) based neural networks (NNs). Furthermore, solving the regularized least squares in ELM is also faster than solving the quadratic programming problem in standard support vector machine (SVM) method. Studies have proved that ELM learns much faster with higher generalization performance than NNs or SVM [2].

Due to its extreme fast training and good generalization performance, ELM has been becoming a significant research topic for pattern recognition and machine learning. ELM and its variant methods present competitive accuracy with superb efficiency in many pattern recognition applications such as face recognition [3, 4], engine fault diagnosis [5], hyperspectral images classification [6], and human action recognition [7, 8]. However, due to their shallow architectures, feature learning using ELM methods may not be effective for some image classification applications, even with a large number of hidden nodes.

In recent years, some deep learning methods have been highlighted and show promising results and significantly outperform shallow neural networks in the field of image classification [9–12]. Composed of many layers, deep learning methods gradually extract more complicated and invariant features from the raw input images than shallow neural networks [13]. The emergence of many large-scale data sets and more powerful computing environments has made the training of deep neural networks possible, leading to a widespread application of deep learning methods.

Among these methods, convolutional neural network (CNN) has gained incredible popularity in many different

domains. It is even becoming the default option for difficult tasks on large image data sets. With local receptive field (LRF) and shared weights, CNN is able to take advantage of the 2D structure of input images and has fewer parameters than fully connected deep networks with the same number of hidden nodes; thus it is easier to train. As all the hidden nodes in CNN need to be tuned with BP learning method, CNN learning faces the problems inherited from BP algorithm such as local minima and time-consuming and intensive human intervention.

On the contrary, ELM does not need tuning of parameters and is extremely fast to implement. Therefore Huang combines the concept of LRF with ELM and proposed a local receptive field based extreme learning machine (LRF-ELM) [14] in order to learn local correlations of input images. The input layer and hidden convolution layer in LRF-ELM are locally connected which allows the network to consider local structures of images. Results on NORB data set show that it has better performance than standard CNN and DBN.

Since LRF-ELM has only one convolution layer followed by a pooling layer, the performance is restricted by its shallow architecture. Another problem is that many feature maps are required in its convolution and pooling layer to attain good performance. Therefore, LRF-ELM consumes much computer memory in implementation. To solve these problems, in this paper, we propose a deep convolutional extreme learning machine (DC-ELM). It adopts multiple alternate convolution layers and pooling layers to obtain more abstract and meaningful feature representations than LRF-ELM. Different from CNN, the local receptive weights are randomly generated without tuning and the output weights are analytically calculated. In order to save computer memory and training complexity, it adopts stochastic pooling [15] in the last hidden layer to reduce dimensionality of feature vector.

To verify the effectiveness of the proposed algorithm, we applied it to some handwritten digits classification tasks and compared it with other state-of-the-art methods. Handwritten digits' recognition has its real world application, such as the postal mail sorting or form data processing [16]. Several methods based on neural networks [17–19], machine learning [20, 21], and other techniques [22, 23] have been studied. Recently some ELM based methods have also been applied to handwritten digits recognition and show good performance on MNIST data set. The ML ELM proposed in [24] achieved 99.03% correct classification. And a test accuracy of 99.19% by deep ELM is achieved in [25]. In [26] a RF-C-ELM was proposed and attained a test accuracy of 99.43%, very close to 99.61%, obtained by Deep Conv. Net [27].

The rest of the paper is organized as follows. Section 2 gives a brief review of ELM and LRF-ELM. Section 3 describes the proposed DC-ELM. In Sections 4 and 5, our method is applied to MNIST and USPS data sets and compared with other state-of-the-art methods. Section 6 analyzes the effect of stochastic pooling and some other parameters. Finally, Section 7 draws the conclusions and points out the future work.

2. Reviews of ELM and LRF-ELM

2.1. Extreme Learning Machine (ELM). ELM was proposed for single-hidden layer feedforward neural networks (SLFNs). It is very different from conventional neural network learning algorithms. It randomly chooses the parameters of hidden nodes and analytically determines the output weights. Thus the training is extremely fast and efficiently completed without time-consuming iterations.

The input data is mapped to an L-dimensional ELM random feature space, and the network output is

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i g_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\,\boldsymbol{\beta}, \tag{1}$$

where $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_L]^T$ is the matrix of output weights and $\mathbf{h}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_L(\mathbf{x})]^T$ are the hidden node outputs for input $\mathbf{x}$. $g_i(\mathbf{x})$ is the output of ith hidden node. Given N training samples $(\mathbf{x}_i, \mathbf{t}_i)$, the ELM can approximate these N samples with zero error which means that

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{2}$$

$\mathbf{H} = [\mathbf{h}^T(\mathbf{x}_1), \ldots, \mathbf{h}^T(\mathbf{x}_N)]^T$, and $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N]^T$ are the matrix of desired output. The output weights can then be calculated using regularized least squares method as follows:

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{T}, \tag{3}$$

where C is the regularization parameter, which is used to obtain better generalization performance.

2.2. Local Receptive Fields Based Extreme Learning Machine (LRF-ELM). As the name suggests, LRF-ELM introduces local receptive field to the input layer, thus obtaining a locally connected ELM. The hidden layers of LRF-ELM consists of a convolution layer and a pooling layer. They are composed of several feature maps. The input weights between input and convolution layers are first randomly generated according to some continuous probability distribution and then orthogonalized in order to obtain a more complete set of features. The square root pooling is used to formulate the combinatorial node in pooling layer. The square and summation operations introduce rectification nonlinearity and translational invariance, respectively, into the network, which is very important for successful image processing tasks. The pooling layer is in full connection with the output layer. The output weights are analytically calculated as in the unified ELM using regularized least squares.

LRF-ELM with local randomly connected hidden nodes can be regarded as a specific type of ELM. Huang has proved that the universal approximation/classification capability of such LRF-ELM can still be preserved.

3. Deep Convolutional Extreme Learning Machine

In this section, we propose a new deep convolutional extreme learning machine designed to solve image classification tasks.

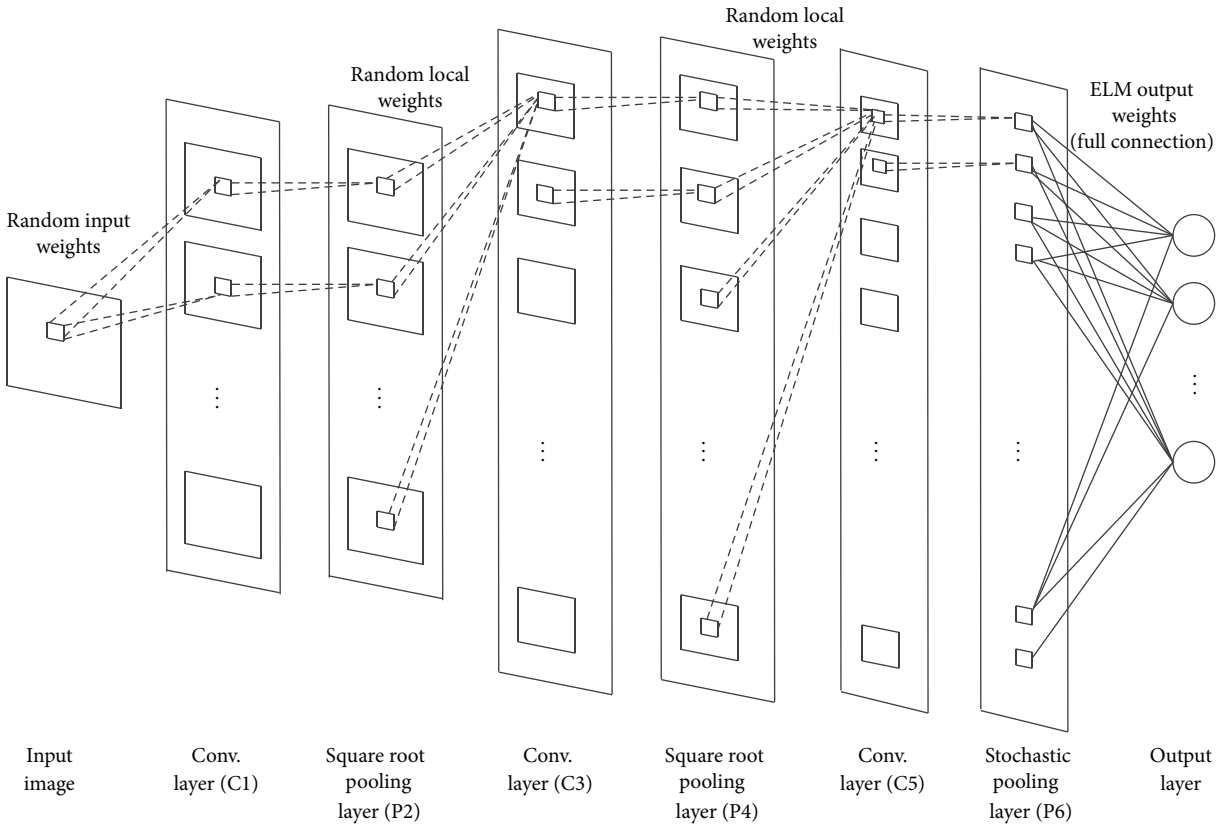


Figure 1: An example of DC-ELM network (with three convolution layers).

DC-ELM combines the feature abstracting performance of convolutional neuron network and fast training of extreme learning machine.

As shown in Figure 1, the structure of DC-ELM consists of an input layer, an output layer, and several hidden layers which are arranged alternately as one convolution layer followed by one pooling layer. The convolution layer consists of several feature maps which are grouped by convolution nodes. The input weights of the same feature map are shared while being distinct among different maps. The square root pooling layer is used to introduce translational invariance to the network. It has the same number of feature maps with the same size as the previous convolution layer. The node in any feature map of a convolution layer is connected to all the feature maps in its previous pooling layer, while the node on a feature map in pooling layer is connected to only one corresponding feature map in its previous convolution layer as shown in Figure 1. The last pooling layer adopts stochastic pooling strategy, thus reducing the size of its feature maps. It is in full connection with the output layer.

This network is designed under three considerations. First, the multiple hidden convolution and pooling layers can extract high level features effectively from input images which is key to image classification tasks. Secondly the shared local receptive weights enable our method to learn local correlations in images and handle image rotation invariance properly. Thirdly the batch training of ELM makes our method run much faster than deep learning methods.

The training procedure for DC-ELM can be described in two phases: (1) convolution feature abstracting, where DC-ELM generates high level feature layer by layer; (2) ELM classifier calculation, where all features generated by convolution and pooling layers are combined into a vector. And the output weights are analytically calculated using regularized least squares method.

3.1. Feature Abstraction

3.1.1. Generation and Orthogonalization of Local Weights. DC-ELM randomly generates input weights between input layer and the first convolution layer (also the local weights between the pooling layer and the following convolution layer) according to some continuous probability distribution. In our paper, we choose Gaussian probability function as the sampling distribution for input weights.

The input/local weights are calculated as follows.

Generate the initial weight matrix $\widehat{\mathbf{A}}_{\text{ini}}$ using Gaussian probability distribution for each feature map in the convolution layer:

$$\widehat{\mathbf{A}}_{\text{ini}} \in R^{r^2 \times K} = \left[\widehat{\mathbf{a}}_{\text{ini}}^1, \widehat{\mathbf{a}}_{\text{ini}}^2, \ldots, \widehat{\mathbf{a}}_{\text{ini}}^K\right] \quad \widehat{\mathbf{a}}_{\text{ini}}^k \in R^{r^2},\ k = 1, \ldots, K, \tag{4}$$

where $r \times r$ is the size of local receptive field; if the previous layer size is $d \times d$, the size of the feature map would be

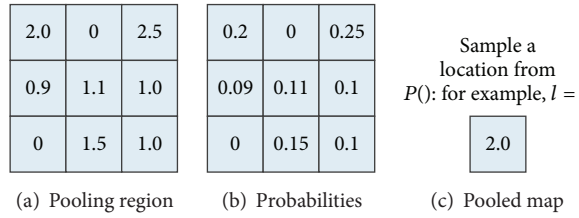


Figure 2: Illustration of stochastic pooling.

$(d - r + 1) \times (d - r + 1)$. Then the initial weight matrix is orthogonalized using singular value decomposition method. Each column $\hat{\mathbf{a}}^k$ of the orthogonalized weight matrix $\widehat{\mathbf{A}}$ is then transformed to $\mathbf{a}^k \in R^{r \times r}$, the input weight to the kth feature map.

3.1.2. Convolution. The convolution layer extracts features using the convolution operation on the input image or the feature maps in the previous pooling layer. For the convolutional node at coordinate (i, j) on the kth feature map in the first convolution layer, it is calculated as

$$c_{i,j,k}(\mathbf{x}) = \sum_{m=1}^{r} \sum_{n=1}^{r} x_{i+m-1,j+n-1} \cdot a_{m,n}^{k} \qquad i, j = 1, \ldots, (d - r + 1), \tag{5}$$

where $\mathbf{x}$ is the input image. Unlike CNN, the feature maps are not applied to nonlinear function. While for the map in higher level convolution layer, as it is connected to all the feature maps in previous pooling layer, we first calculate the convolution with each feature map using its local weights as (5); then we add them up, and we obtain the pooled feature map.

3.1.3. Pooling. In DC-ELM, we adopt two pooling strategies: square root pooling and stochastic pooling. The effectiveness of square root pooling has been testified by some research [28, 29]. In [28], it has been shown that convolutional square pooling architectures can be inherently frequency selective and translation invariant, even when initialized with random weights. Square root pooling includes the square operation and summation operation. These two operations, respectively, introduce rectification nonlinearity and translational invariance into the network, which have been discussed in [29]. And experimental results show that square root pooling helps the network outperform max or average pooling.

The square root pooling is applied to all pooling layers except the last pooling layer. The node at coordinate (p, q) on the kth pooling map is calculated as

$$h_{p,q,k} = \sqrt{\sum_{i=(p-e)}^{p+e} \sum_{j=(q-e)}^{q+e} c_{i,j,k}^{2}} \qquad p, q = 1, \ldots, (d - r + 1); \text{ if } (i, j) \text{ is out of bound: } c_{i,j,k} = 0, \tag{6}$$

where e is the pooling size.

As the size of square root pooling map maintains the same as the previous convolution layer, to reduce the dimensionality of the last hidden layer, there are two conventional choices: mean pooling and max pooling. The former takes the arithmetic mean of the nodes in each pooling region while the max pooling selects the largest node. But these two types of pooling have their own drawbacks when training deep convolutional networks. Average pooling has the effect of downweighting strong activations and leads to small pooled responses in some cases while max pooling tends to overfit the training set and affect the generalization performance. To overcome these drawbacks, a stochastic pooling scheme was proposed in [15]. It has the advantages of max pooling, but its stochastic nature helps to prevent overfitting problem. Experimental results show that stochastic pooling has better performance on image classification applications than mean and max pooling. In this work, we use the stochastic pooling scheme for the last pooling layer.

In stochastic pooling, the pooled maps are determined by sampling from each pooling region using a multinomial distribution. We first calculate the probabilities for each pooling region R_j by normalizing the values of the nodes within the region in previous convolution layer as follows:

$$p_i = \frac{c_i}{\sum_{n \in R_j} c_n}. \tag{7}$$

Then we sample from the multinomial distribution based on p to pick a location l within the region. The pooled value s_j is then simply the value of node at location l (c_l):

$$s_j = c_l \quad \text{where } l \sim P\left(p_1, \ldots, p_{|R_j|}\right). \tag{8}$$

The procedure of stochastic pooling can be illustrated by Figure 2. Figure 2(a) shows a pooling region with 3×3 elements. Figure 2(b) is the calculated probabilities for this region. We sample from the multinomial distribution to pick a location l from $[1, 2, \ldots, 9]$; then the pooled value is the element in this location. For example, if $l = 1$, the pooled value would be 2.0, the value in the first grid of the pooled region.

It should be noted that the selected node for the pooling region may not be the largest one. Stochastic pooling can thus represent multimodal distributions of convolution nodes within a region. Furthermore, by stochastic pooling, we can obtain a pooled feature map with much smaller size than square root pooling

All the alternate convolution and pooling layers form a hierarchical feature extractor that maps the original input

images into high level features which makes the classification more efficient. And the use of stochastic pooling helps DC-ELM to reduce computation burden greatly, thus leading to shorter training time.

3.2. ELM Output Weights Calculation. After feature generation, the values of all nodes in the last stochastic pooling layer are concatenated into a row vector. And putting the rows of N input samples together, we obtain the hidden layers output matrix $\mathbf{H} \in R^{N \cdot K \cdot r_s^2}$, where r_s is the size of feature map in stochastic pooling layer.

Then the output weights are analytically determined using regularized least squares method as

$$\boldsymbol{\beta} = \begin{cases} \mathbf{H}^T \left(\dfrac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} & \text{if } N \le K \cdot r_s^2, \\ \left(\dfrac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H} \right)^{-1} \mathbf{H}^T\mathbf{T} & \text{if } N > K \cdot r_s^2, \end{cases} \tag{9}$$

where T is the labels of the input sample images and regularization parameter C is used to control the trade-off between the norm of output weights and training error term. The suitable setting of this parameter helps to improve the algorithm's generalization performance.

4. Evaluation on MNIST Data Set

In this section, we evaluate the performance of DC-ELM on MNIST [30] data set. The performance of DC-ELM is compared with ELM, LRF-ELM, and two deep learning algorithms: CNN and DBN. The MNIST handwriting data set contains 60 000 training images and 10 000 testing images of handwriting digits 0–9. As different digits have their unique shapes and different people write the number in their own ways, the MNIST is an ideal data set and commonly used to test deep learning algorithms.

4.1. Parameter Setting. For basic ELM, we adopt a single hidden layer with 3000 hidden nodes and sigmoid function. The parameter setting for LRF-ELM, DC-ELM, and CNN is listed in Table 1. For LRF-ELM, we employ 30 feature maps. The kernel size is 3×3 and the pooling size is 5×5. While for DC-ELM, we adopt the architecture of three convolution layers and three pooling layers. The number of feature maps in convolution layers is 5, 10, and 15 separately. And the kernel size is 3×3 for all convolution layers. The pooling size is 5×5 for the first two pooling layers and the last one adopts stochastic pooling with a pooling size of 2×2. The regularization parameter C is chosen as 0.01 by experiments for all ELM methods.

To compare fairly, CNN employs the same number of hidden layers and feature maps as DC-ELM. But unlike DC-ELM, whose feature maps size does not decrease after square root pooling, CNN adopts mean pooling and the size of feature maps will decrease after pooling. Thus the kernel or pooling size setting in CNN is different from that in DC-ELM. The kernel size is 5×5 for the first two convolution layers and 3×3 for the last one. The pooling size is 2×2 for all mean pooling layers. The learning epochs are set as 100. For DBN, the hidden layer structure is 500-500-1000; the learning rate is set as 0.1. The unsupervised pretraining epochs are set as 50 and supervised fine-tuning epochs are set as 100. The training data set is divided into minibatches, each containing 100 samples.

Table 1: Parameter setting for LRF-ELM, DC-ELM, and CNN.

Method	K	LRF size Kernel size	Pooling size
DC-ELM	5	3×3	5×5
	10	3×3	5×5
	15	3×3	2×2
LRF-ELM	30	3×3	5×5
CNN	5	5×5	2×2
	10	5×5	2×2
	15	3×3	2×2

All simulations have been made in MATLAB R2008a environment running on a PC with 3.4 GHz CPU with 2 cores and 4 GB RAM.

4.2. Experimental Results. As our computer has only 4 GB RAM, more training samples may cause "run out of memory" error; we did not use all 60000 training samples. Instead, the training samples are randomly selected from original training set. We trained all methods with 10000 and 15000 training samples separately.

All the methods are run 10 times separately for each case. Tables 2 and 3 list the results of these methods on two cases. These results include training time, mean, and standard deviation of training/testing accuracy.

It can be seen from the tables that our method achieved the highest mean testing accuracy among all the methods on both cases with different training samples. Because we did not use all the 60000 samples for training, the testing accuracy is less than that in some literature introduced in Section 1, but it is still satisfied considering that the number of training samples is relatively small. In general, LRF-ELM achieved better testing performance than deep learning methods. Although ELM attained the highest training accuracy, its testing accuracy is not satisfying compared to other methods. It can be also found that our method consumed fewer training time on two cases than ELM or LRF-ELM. This is mainly attributed to the application of stochastic pooling which reduces the dimensionality greatly of the feature vector. On the contrary, ELM and LRF-ELM need a large amount of hidden nodes to attain good generalization performance, thus consuming more computational resource and training time. The training times of CNN and DBN are much longer than ELM methods due to the iteration nature of deep learning.

To give insight into how DC-ELM abstract features layer by layer, Figure 3 uses one image sample (digit 8) for illustration. It displays all the feature maps generated in each convolution layer. It can be observed that the five feature maps in the first convolution layer looks similar, as they

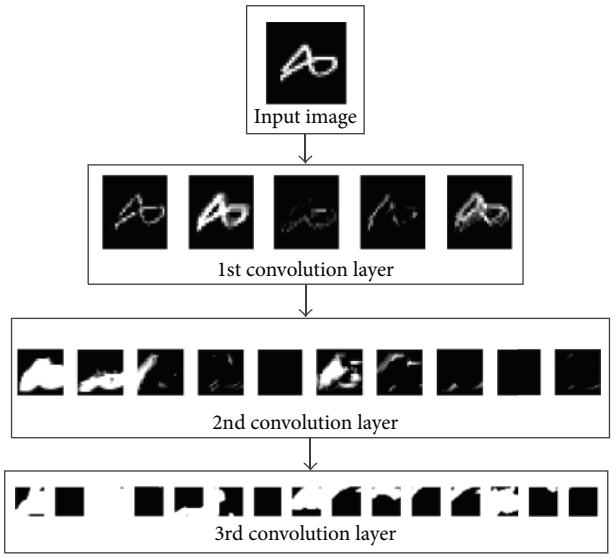


Figure 3: Input image and corresponding feature maps in different convolution layers.

Table 2: Performance on MNIST (with only 10k training samples).

Algorithms	Training time (s)	Training accuracy		Testing accuracy	
		Mean	Std	Mean	Std
ELM	102.6	0.9965	0.0013	0.9457	0.0017
LRF-ELM	179.8	0.9882	0.0058	0.9763	0.0034
DC-ELM	**78.15**	0.9802	0.0027	**0.9803**	0.0015
CNN	2031.5	0.9703	0.0017	0.9724	0.0013
DBN	2968.4	0.9872	0.0022	0.9632	0.0021

Table 3: Performance on MNIST (with only 15k training samples).

Algorithms	Training time (s)	Training accuracy		Testing accuracy	
		Mean	Std	Mean	Std
ELM	169.3	0.9976	0.0012	0.9522	0.0009
LRF-ELM	264.9	0.9890	0.0039	0.9779	0.0021
DC-ELM	**120.4**	0.9887	0.0020	**0.9841**	0.0012
CNN	2960.9	0.9873	0.0014	0.9781	0.0011
DBN	4791.3	0.9905	0.0009	0.9759	0.0015

all are generated from the same input digit. However, each feature map has its unique highlighted part; thus we obtain the diverse representations of the original image. Then these feature maps are processed by pooling and passed to the next level convolution layer. After several convolution and pooling operations we can obtain the high level features which make the subsequent ELM classification more accurate and efficient.

5. Evaluation on USPS Data Set

We also demonstrate the performance of DC-ELM on USPS [31] data set. The USPS handwriting data set consists of 11 000 samples of handwriting digits 0–9 which are collected from different writers.

5.1. Parameter Setting. The parameter setting in this section is different from that on MNIST data set. We have found that DC-ELM attains better performance with fewer hidden layers for USPS than for MNIST, partly because the size of sample image in USPS data set is much smaller.

The parameter setting for LRF-ELM, DC-ELM, and CNN is listed in Table 4. Basic ELM and LRF-ELM have the same parameter setting as in Section 4 while DC-ELM adopts two hidden convolution layers with 10 and 15 feature maps separately. And the kernel size is 3×3 for both convolution layers. The pooling size is 5×5 for the first pooling layer and 2×2 for the second one. The regularization parameter C is 0.01 for all ELM methods. To compare fairly, CNN also adopts two convolution layers. The kernel size is 3×3 for the first convolution layer and 2×2 for the second. The pooling size is 2×2 for both mean pooling layers. The learning epochs is set

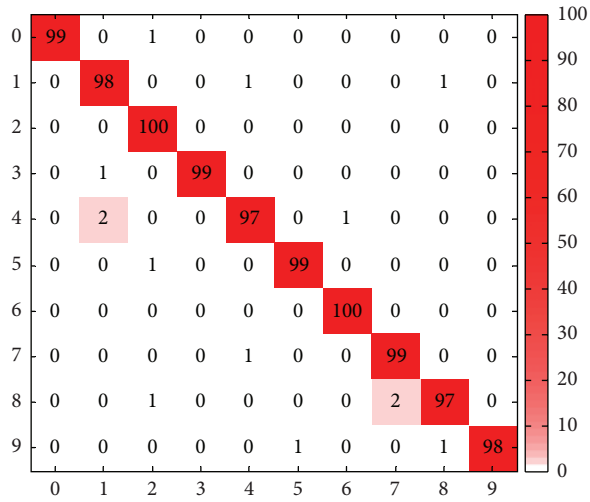


Figure 4: Confusion matrix obtained by DC-ELM on USPS data set.

Table 4: Parameter setting for LRF-ELM, DC-ELM, and CNN.

Method	K	LRF size Kernel size	Pooling size
DC-ELM	10	3×3	5×5
	20	3×3	2×2
LRF-ELM	30	3×3	5×5
CNN	10	3×3	2×2
	20	2×2	2×2

as 100. For DBN, the hidden layer structure is 500-1000 and other parameters are the same as that for MNIST data set.

5.2. Experimental Results. We test all the methods on two different cases with 7000 and 10000 training samples separately. The training samples are randomly selected from USPS data set. We trained all methods with 10000 and 15000 training samples separately. All the methods are run 10 times separately for each case. The results are listed in Tables 5 and 6.

It can be seen from Tables 5 and 6, like on MNIST data set, that our method achieved the best testing accuracy and the shortest training time on both cases on USPS data set. ELM achieved very high training accuracy, but its performance on test data is inferior to DC-ELM. This suggests that ELM is suffering from overfitting in the training set. Also it can be concluded that deep architecture would be helpful to improve ELM's generalization ability. We can also see that training our method is much faster than LRF-ELM. And the training time of LRF-ELM is shorter than ELM. It can be concluded that the introduction of local receptive field and stochastic pooling would help to accelerate the training of ELM. The testing performance of deep learning methods is better than ELM but is inferior to LRF-ELM and DC-ELM and is much time-consuming.

Figure 4 shows a confusion matrix obtained by our method in a typical run; it can be seen that DC-ELM obtained a satisfying classification result since the confusion matrix is very close to dominantly diagonal. And two digits are classified with 100 percent accuracy.

6. Parameter Analysis

6.1. Effect of Network Depth. The network structure has great impact on method. To study how the depth of the network affects the performance of DC-ELM, we conducted experiments on MNIST (with 15k training samples) and USPS (with 10k training samples) data sets with varying number of convolution layers, and each layer contains 10 feature maps. Figure 5 depicts how the mean test error of ten trials varies with number of convolution layers.

Interestingly, we found that it is not "the more the better" for the hidden convolution layers. And the best depth of network is not the same for different data sets. As it can be seen in Figure 5, DC-ELM obtains best test error with three convolution layers on MNIST data set, while it performs best with only two convolution layers on USPS data set. It can be concluded that the best network structure for DC-ELM is related to applications. There is no unified optimal network structure for all data sets.

6.2. Effect of Regularization Parameter C. We have also tested the effect of regularization parameter C in (9). It is used to control the trade-off between the norm of output weights and training error term. Figure 6 shows how the mean test error on two data sets varies with parameter C. It can be seen from the figure that the test error decreases firstly and then increases greatly with parameter C. It reaches the lowest value when $\log C = -2$. Thus, in Sections 4 and 5, we set this parameter equal to 0.01.

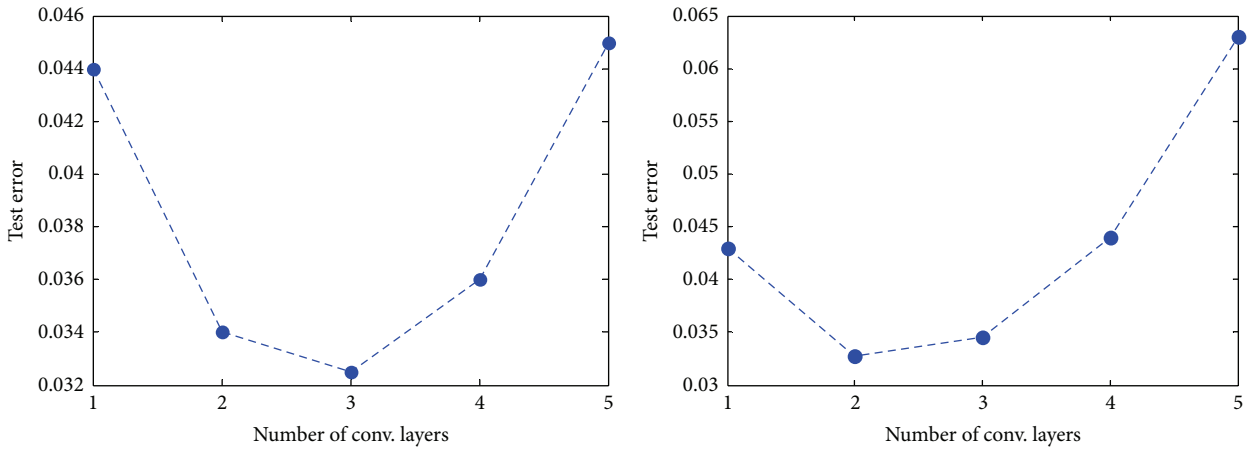
6.3. Effect of Stochastic Pooling. We have replaced stochastic pooling in the last pooling layer with square root pooling and tested it on MNIST and USPS. Figure 7 shows the mean training time and test error of ten trials by square root pooling

Table 5: Performance on USPS (with 7000 training samples).

Algorithms	Training time (s)	Training accuracy		Testing accuracy	
		Mean	Std	Mean	Std
ELM	90.1	0.9974	0.0016	0.9323	0.0021
LRF-ELM	37.5	0.9849	0.0045	0.9651	0.0027
DC-ELM	**20.6**	0.9783	0.0034	**0.9693**	0.0028
CNN	2590.3	0.9750	0.0022	0.9614	0.0021
DBN	3861.4	0.9696	0.0033	0.9586	0.0026

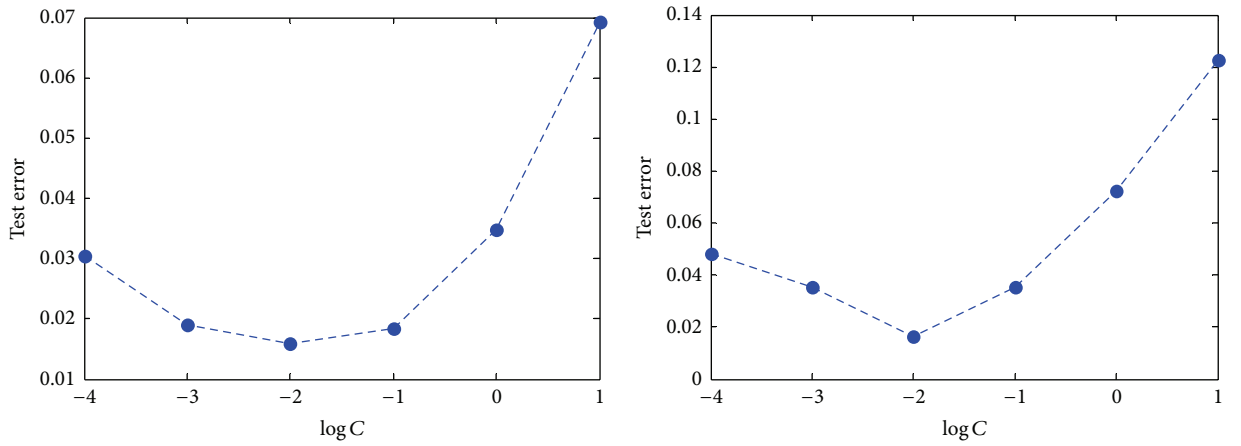
Table 6: Performance on USPS (with 10000 training samples).

Algorithms	Training time (s)	Training accuracy		Testing accuracy	
		Mean	Std	Mean	Std
ELM	109.5	0.9992	0.0011	0.9650	0.0013
LRF-ELM	55.2	0.9948	0.0039	0.9803	0.0022
DC-ELM	**27.8**	0.9889	0.0025	**0.9843**	0.0014
CNN	3731.5	0.9815	0.0019	0.9736	0.0020
DBN	5284.1	0.9877	0.0024	0.9704	0.0015



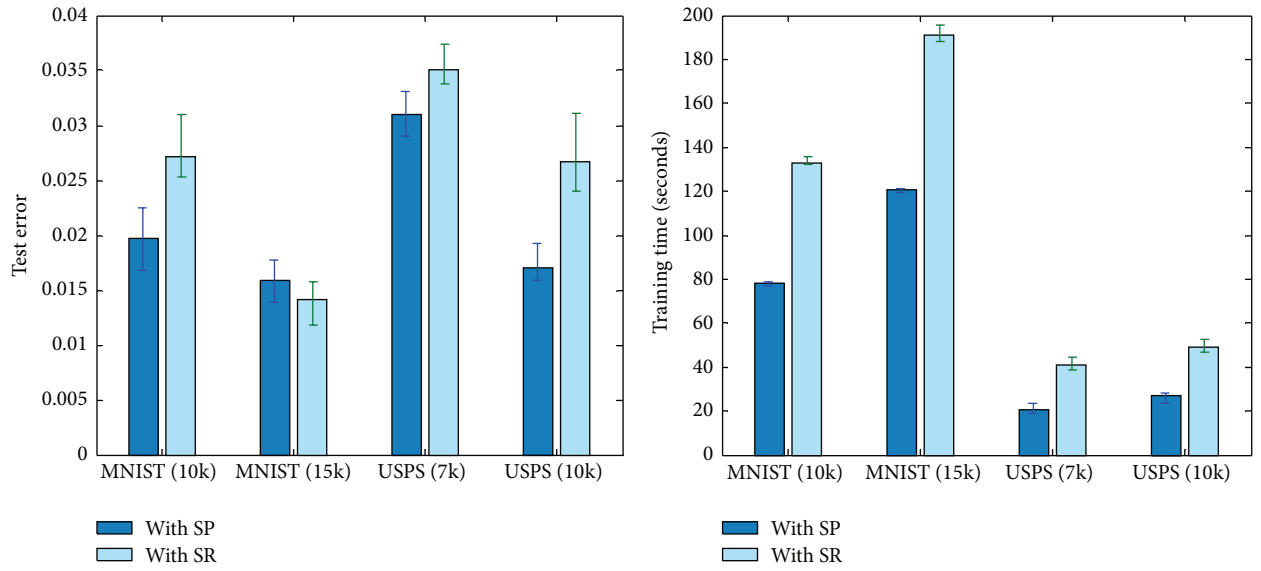
(a) Mean test error on MNIST (b) Mean test error on USPS

Figure 5: Effect of network depth on the mean test error.



(a) Mean test error on MNIST (b) Mean test error on USPS

Figure 6: Effect of regularization parameter C on the mean test error.



(a) Mean test error

(b) Mean training time

Figure 7: Comparison of stochastic pooling with square root pooling.

and that by stochastic pooling on MNIST and USPS data sets. The tests were conducted with different number of training samples.

It can be seen that the mean test errors with stochastic pooling are smaller or very close to that with square root pooling. This suggests that stochastic pooling is effective in improving ELM on its testing accuracy. Meanwhile the training time of DC-ELM reduces greatly in all cases. This is because of the fact that the stochastic pooling reduces the dimensionality in the last pooling layer, thus speeding up the calculation of output weights.

7. Conclusions

In this paper, a deep convolutional extreme learning machine is presented. It employs multiple alternate convolution layers and pooling layers that gradually extract more sophisticated and robust features from the raw input images than LRF-ELM. Based on these high level features, ELM classifier on top of the hidden layers provides a deterministic solution of the output weights. There is no parameter tuning or iterations in the training process of DC-ELM; thus the training of our method is very faster than deep learning methods.

In the implementation, we use Gaussian probability function to sample the local connections. The square root pooling is used after convolution to further introduce translational invariance into the network. We apply a simple and effective stochastic pooling strategy in the last pooling layer in order to reduce the size of feature vector, thus saving much computational resource.

Experiments conducted on handwritten digit recognition tasks show that the proposed DC-ELM presents better test accuracy on different cases than ELM, LRF-ELM, and state-of-the-art deep leaning methods. This suggests that the deep convolutional feature abstraction is more efficient than the shallow one for ELM classifier. Furthermore, training our method is much faster than other compared methods thanks to stochastic pooling and the appropriate LRF size. Therefore, we argue that our method provides an effective and useful tool which may benefit handwritten recognition and other image classification tasks.

The parameter analysis also indicates that the best network structure of DC-ELM is application-dependent. Thus it may be worth investigating method which determines the optimal network structure for different image classification applications in the future.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

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

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

[3] A. A. Mohammed, R. Minhas, Q. M. Jonathan Wu, and M. A. Sid-Ahmed, "Human face recognition based on multidimensional PCA and extreme learning machine," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2588–2597, 2011.

[4] W. W. Zong and G.-B. Huang, "Face recognition based on extreme learning machine," *Neurocomputing*, vol. 74, no. 16, pp. 2541–2551, 2011.

[5] X. Yang, S. Pang, W. Shen, X. Lin, K. Jiang, and Y. Wang, "Aero engine fault diagnosis using an optimized extreme learning

machine," *International Journal of Aerospace Engineering*, vol. 2016, Article ID 7892875, 10 pages, 2016.

[6] Y. Zhou, J. Peng, and C. L. P. Chen, "Extreme learning machine with composite kernels for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2351–2360, 2015.

[7] R. Minhas, A. Baradarani, S. Seifzadeh, and Q. M. Jonathan Wu, "Human action recognition using extreme learning machine based on visual vocabularies," *Neurocomputing*, vol. 73, no. 10–12, pp. 1906–1917, 2010.

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

[9] N. Le Roux and Y. Bengio, "Representational power of restricted Boltzmann machines and deep belief networks," *Neural Computation*, vol. 20, no. 6, pp. 1631–1649, 2008.

[10] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009.

[11] D. Yu, L. Deng, I. Jang, P. Kudumakis, M. Sandler, and K. Kang, "Deep learning and its applications to signal and information processing," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 145–154, 2011.

[12] H. A. Perlin and H. S. Lopes, "Extracting human attributes using a convolutional neural network approach," *Pattern Recognition Letters*, vol. 68, pp. 250–259, 2015.

[13] N. Le Roux and Y. Bengio, "Deep belief networks are compact universal approximators," *Neural Computation*, vol. 22, no. 8, pp. 2192–2207, 2010.

[14] G.-B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 18–29, 2015.

[15] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," in *Proceedings of the Learning Representations*, Scottsdale, Ariz, USA, May 2013.

[16] E. Mohebi and A. Bagirov, "A convolutional recursive modified Self Organizing Map for handwritten digits recognition," *Neural Networks*, vol. 60, pp. 104–118, 2014.

[17] A. Goltsev and V. Gritsenko, "Investigation of efficient features for image recognition by neural networks," *Neural Networks*, vol. 28, pp. 15–23, 2012.

[18] L. Pape, F. Gomez, M. Ring, and J. Schmidhuber, "Modular deep belief networks that do not forget," in *Proceedings of the 2011 International Joint Conference on Neural Network (IJCNN '11)*, pp. 1191–1198, IEEE, San Jose, Calif, USA, August 2011.

[19] D. Yu and L. Deng, "Efficient and effective algorithms for training single-hidden-layer neural networks," *Pattern Recognition Letters*, vol. 33, no. 5, pp. 554–558, 2012.

[20] X.-X. Niu and C. Y. Suen, "A novel hybrid CNN-SVM classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, 2012.

[21] Z. Man, K. Lee, D. Wang, Z. Cao, and S. Khoo, "An optimal weight learning machine for handwritten digit image recognition," *Signal Processing*, vol. 93, no. 6, pp. 1624–1638, 2013.

[22] N. Das, J. M. Reddy, R. Sarkar et al., "A statistical-topological feature combination for recognition of handwritten numerals," *Applied Soft Computing*, vol. 12, no. 8, pp. 2486–2495, 2012.

[23] C. D. Stefano, F. Fontanella, C. Marrocco, and A. S. di Freca, "A GA-based feature selection approach with an application to handwritten character recognition," *Pattern Recognition Letters*, vol. 35, no. 1, pp. 130–141, 2014.

[24] 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.

[25] M. D. Tissera and M. D. McDonnell, "Deep extreme learning machines: supervised autoencoding architecture for classification," *Neurocomputing*, vol. 174, pp. 42–49, 2016.

[26] M. D. McDonnell, M. D. Tissera, T. Vladusich, A. Van Schaik, J. Tapson, and F. Schwenker, "Fast, simple and accurate handwritten digit classification by training shallow neural network classifiers with the 'extreme learning machine' algorithm," *PLoS ONE*, vol. 10, no. 8, Article ID 0134254, 2015.

[27] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," *Journal of Machine Learning Research*, vol. 38, pp. 562–570, 2015.

[28] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On random weights and unsupervised feature learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML '11)*, pp. 1089–1096, Bellevue, Wash, USA, July 2011.

[29] Y.-L. Boureau, J. Ponce, and Y. Lecun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th International Conference on Machine Learning (ICML '10)*, pp. 111–118, Haifa, Israel, June 2010.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[31] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in Neural Information Processing Systems*, vol. 16, no. 3, pp. 21–328, 2004.