

Fast Handwritten Digit Recognition with Multilayer Ensemble Extreme Learning Machine

Hossein Ghodrati Noushahr, Samad Ahmadi and Arlene Casey

Abstract Conventional artificial neural networks and convolutional neural networks perform well on the task of automatic handwriting recognition. But, they suffer from long training times and their complex nature. An alternative learning algorithm called Extreme Learning Machine overcomes these shortcomings by determining the weights of a neural network analytically. In this paper, a novel classifier based on Extreme Learning Machine is proposed that achieves competitive accuracy results while keeping training times low. This classifier is called multilayer ensemble Extreme Learning Machine. The novel classifier is evaluated against traditional backpropagation and Extreme Learning Machine on the well-known MNIST dataset. Possible future work on parallel Extreme Learning Machine is shown up.

1 Introduction

Artificial Neural Networks (ANN) have been successfully applied for the difficult task of handwritten digit recognition. However, ANN that train their weights with the traditional backpropagation (BP) algorithm suffer from slow learning speed. This has been a major bottleneck for ANN applications in the past.

Recently, Extreme Learning Machine (ELM) has been proposed as an alternative to BP for the task of training ANN [8]. ELM follows an approach that aims to reduce human invention, increase learning accuracy, and to reduce the time to train an ANN. This is done by randomly initiating the weights, then fixing the weights of the hidden layer nodes and subsequently determining the weights of the output layer analytically.

H. Ghodrati Noushahr (✉) · S. Ahmadi · A. Casey
Centre for Computational Intelligence, School of Computer Science and Informatics,
De Montfort University, The Gateway, Leicester LE1 9BH, UK
e-mail: hossein@ghodrati.net

S. Ahmadi
e-mail: sahmadi@dmu.ac.uk

A. Casey
e-mail: arlene.casey@myemail.dmu.ac.uk

Human invention is reduced as hyper-parameters, such like the learning rate and the momentum of traditional BP do not have to be determined manually.

ELM was applied successfully on a variety of classification and function approximation tasks [7]. In this paper, a novel classifier based on ELM will be presented that achieves competitive accuracy results while keeping the training time low and limiting human invention.

The remainder of this paper is structured as follows. In Sect. 2 the handwriting recognition problem and the corresponding dataset will be explained in detail. In Sect. 3 recent classifier for this task will be reviewed. This includes conventional ANN, but also very successful variations of ANN called convolutional neural networks (CNN) that fall into the research area of *Deep Learning*. Section 4 introduces ELM and ELM ensemble models. Furthermore, the novel classifier will be presented in the same section. The results of the experimental work conducted will be shown in Sect. 5, and Sect. 6 concludes the paper.

2 Problem Definition

Automatic handwriting recognition is a challenging problem that caught academic and commercial interest. Some commercial applications are: letter sorting in post offices, personal check reading in banks, or large-scale digitization of manuscripts [5]. The Mixed National Institute of Standards and Technology Database (MNIST) is the most widely used benchmark for isolated handwritten digit recognition [10]. It consists of 70,000 images from approx. 250 writers. 60,000 images represent the training sample, and the remaining 10,000 images the test sample for evaluation. The images have $28 \times 28 = 784$ gray-scale pixels (0: background—255: maximum foreground intensity). Figure 1 shows examples of the ten digits in the MNIST database.

3 Related Work

3.1 Single Hidden Layer Feedforward Neural Networks

ANN are massively parallel distributed processors made up of simple processing units. ANN are inspired by the human brain and the way it processes information.

Fig. 1 Examples from the MNIST database



One of the main benefits of ANN is their ability to detect nonlinear relations and patterns. Single Hidden Layer Feedforward Neural Networks (SLFN) are ANN with only one hidden layer. Conventional SLFN train the weights of the ANN with the BP algorithm. BP is a gradient-based learning algorithm that tunes iteratively all parameters of the SLFN.

LeCun et al. [9] evaluated SLFN against the MNIST database. A SLFN with 300 hidden layer nodes had an error rate of 4.7 % on the test set. A SLFN with 1,000 hidden layer nodes achieved a slightly better error rate of 4.5 % on the test set.

3.2 Multiple Hidden Layer Feedforward Neural Networks

Multiple Hidden Layer Feedforward Neural Networks (MLFN) are identical to SLFN, but with the main difference that they have more than one hidden layer. Although it is proven that SLFN are universal approximators [6], MLFN were used in the past for the handwritten digit recognition problem. In [9] error rates as low as 3.05 and 2.95 % were achieved with a 300-100-10 and a 500-150-10 MLFN.

3.3 Convolutional Neural Networks

LeCun et al. [9] proposed CNN with a focus on automatic learning and higher order feature selection. CNN combine three architectural ideas to ensure shift, scale, and distortion invariance: local receptive fields, shared weights, and spatial subsampling.

A node in the hidden layer is not connected to all inputs from the previous layer, but only to a subregion. The advantage of local receptive fields is that they reduce dramatically the number of weights compared to fully connected hidden layer. Furthermore, this approach is computationally less expensive.

Hidden layer nodes are organized in so called feature maps that share the same weights. As each hidden layer node within a feature map has a different local receptive field, the same pattern can be detected across the whole receptive field. Each feature map is specialized to recognize a different pattern by having different weights. The architectural idea of weight sharing reduces even more the number of weights.

The idea of spatial subsampling refers to the reduction of the receptive field resolution. In the case of LeNet-5 [9] a non-overlapping 2x2 neighborhood in the previous layer is aggregated to a single output. The aggregation could be either the maximum, or the average of the 2x2 neighborhood. The subsampling layer reduces the number of inputs by the factor 4. Spatial subsampling provides invariance to local translations.

Convolutional layer implement the local receptive field concept and also the weight sharing. Subsampling layer realize the idea of spatial subsampling.

Figure 2 illustrates the architecture of LeNet-5 [9]. It consists of 6 layers and convolutional layers are labeled Cx, subsampling layers Sx, and fully connected layers

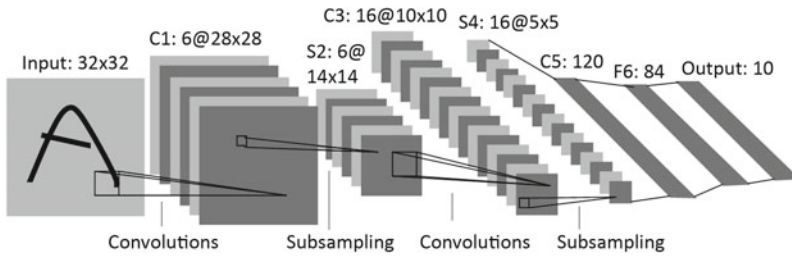


Fig. 2 Architecture of the CNN LeNet-5 [9]

F_x , where x is the layer index. The first convolutional layer $C1$ has 6 28×28 feature maps followed by the subsampling layer $S2$ with also 6 feature maps that reduce the size to 14×14 . Layer $C3$ is a 10×10 convolutional layer having 16 feature maps. Layer $S4$ is again a subsampling layer that reduces the size further down to 5×5 . Layer $C5$ is a convolutional layer with 120 1×1 feature maps. Layer $F6$ is a fully connected layer that computes a dot product between the input vector and the weight vector and a bias like in traditional ANN.

It can be summarized that CNN scan automatically the input image for higher order features. The exact position of these higher order features is less relevant, only the relative position to other higher order feature is relevant. In the case of the number 7 a CNN would look for the endpoint of a horizontal element in the upper left area, a corner in the upper right area, and an endpoint of a roughly vertical segment in the lower portion of the image.

LeNet-5 could reach a test error rate of 0.95 % [9] on the MNIST dataset, more recent classifier based on CNN could reach test error rates as low as 0.23 % [4]. These error rates are comparable to error rates of humans performing the same task [11].

3.4 Other Approaches

Image deformation is a technique that was applied in some studies for the problem discussed in this paper. Ciresan et al. [5] deformed the 60,000 training images to get more training samples. They combined rotation, scaling, horizontal shearing, and elastic deformations in order to emulate uncontrolled oscillations of hand muscles and trained with the original and additional training samples several MLFN. A test error of 0.32 % was reached. Although, 12.1 M weights had to be trained with a total training time of 114.5 h (Table 1).

Alonso-Weber et al. [1] followed a similar approach. In addition to the above mentioned deformations, noise was fed into the MLFN by wiping out a proportion of pixels and also adding pixels randomly. The MLFN had a topology of 300-200-10. A test error rate of 0.43 % was achieved. No statistics about the training times were provided (Fig. 3).

Table 1 Error rates on MNIST database in [5]

Model	Topology	Best test error (%)	Training time	Weights (M)
1	1000-500-10	0.44	23.4h	1.34
2	1500-1000-500-10	0.40	44.2h	3.26
3	2000-1500-1000-500-10	0.39	66.7h	6.69
4	2500-2000-1500-1000-500-10	0.32	114.5h	12.11

Fig. 3 Image deformation examples applied in [1]

4 Extreme Learning Machines

4.1 Review of Extreme Learning Machine

Huang et al. [8] proposed ELM as a new learning algorithm to train SLFN. The original design objectives and key advantages of ELM compared to conventional BP are: least human invention, high learning accuracy, and fast learning speed [7].

Due to the slow learning speed, BP has been a major bottleneck for SLFN applications in the past decades [8]. ELM follows a very different approach: hidden layer weights are chosen randomly and the output layer weights determined analytically by solving a linear system using the least square method. Hence, no hyper-parameters such like the learning rate or the momentum need to be determined manually compared to traditional BP.

For N arbitrary distinct training samples $\{(x_i, t_i)\}_{i=1}^N$, where $x_i \in R^d$ and $t_i \in R^m$, the output of a SLFN with L hidden nodes is:

$$f_L(x) = \sum_{i=1}^L \beta_i h_i(x) = h(x)\beta \quad (1)$$

where $\beta = [\beta_1, \dots, \beta_L]^T$ is the output weight vector between L hidden layer nodes and $m \geq 1$ output nodes. $h_i(x)$ is the output in form of a nonlinear activation function of the i th hidden node for the input x . Table 2 lists the most common activation functions.

Table 2 Activation functions in ELM

Sigmoid function	$h(x) = \frac{1}{1+\exp(-\omega x+b)}$
Hyperbolic tangent function	$h(x) = \frac{1-\exp(-\omega x+b)}{1+\exp(-\omega x+b)}$
Gaussian function	$h(x) = \exp(-b \ x - \omega\)$

For all N training samples, Eq. 1 can be written in an equivalent compact form as:

$$T = H\beta \quad (2)$$

where H is the hidden layer output matrix:

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \dots & h_L(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \dots & h_L(x_N) \end{bmatrix} \quad (3)$$

and T is the training sample target matrix:

$$T = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix} = \begin{bmatrix} t_{11} & \dots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{N1} & \dots & t_{Nm} \end{bmatrix} \quad (4)$$

The output layer weights β are determined by minimizing the squared approximation error:

$$\min_{\beta \in \mathbb{R}^{L \times m}} \|H\beta - T\|^2, \quad (5)$$

The optimal solution to 5 is given by

$$\hat{\beta} = H^+T, \quad (6)$$

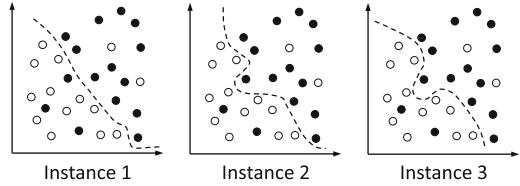
where H^+ denotes the Moore-Penrose generalized inverse of matrix H . Algorithm 1 summarizes the ELM learning algorithm.

Algorithm 1 ELM

Input: N arbitrary training samples $\{(x_i, t_i)\}_{i=1}^N$

- 1: Assign randomly hidden layer weights
 - 2: Calculate the hidden layer output matrix H
 - 3: Calculate the output layer weights β : $\hat{\beta} = H^+T$
-

Fig. 4 ELM instances with different random parameters



4.2 Ensemble ELM

The combination of multiple classifiers reduces the risk of overfitting and leads to better accuracy compared to single model classifiers. Such combined classifiers are referred to as ensemble classification models. Based on promising results of ensemble models for ELM presented in [3], an ensemble model called Ensemble ELM (EELM) will be built and evaluated against the MNIST database.

ELM constructs a nonlinear separation boundary in classification problems. Samples that are located near the classification boundary might be misclassified by one single ELM model. Figure 4 illustrates on the left hand side such a misclassification near the boundary. The classification boundary depends on the randomly initiated weights of the hidden layer neurons. As these weights are not changed during the training phase, the classification boundary remains as initialized. The majority vote of several ELM that are initialized with independent random weights reduces the misclassification of samples near the classification boundary. Algorithm 2 summarizes the EELM algorithm.

Algorithm 2 EELM

Input:

- N arbitrary training samples $\{(x_i, t_i)\}_{i=1}^N$;
- N^{test} test samples $\{(x_i^{test}, t_i^{test})\}_{i=1}^{N^{test}}$;
- Odd number of independent ELM: K

Training phase:

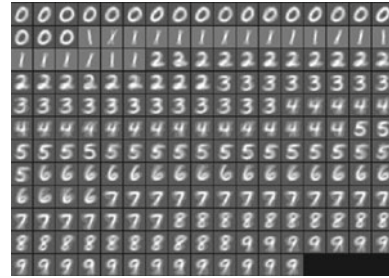
- 1: Create K ELM classifier with independent weights
- 2: Train all K independent ELM classifier with the training samples

Test phase:

- 3: For each test sample predict the class as the majority vote of all K independent ELM classifier
-

Figure 4 illustrates for a $K = 3$ EELM model the correct classification of a sample near the classification boundary due to a majority vote of two ELM models. Additional to [3], further ELM ensembles are mentioned in [7].

Fig. 5 ELM autoencoder output weight visualization



4.3 Multilayer ELM

Kasun et al. [2] proposed a multilayer ELM (ML-ELM) based on the idea of autoencoding. That is, extract higher order features by reducing the high dimensional input data to a lower dimensional feature space similar to CNN. This is done as follows: unsupervised learning is performed by setting the input data as the output data $t = x$. The random weights are chosen to be orthogonal as it tends to improve the generalization performance. Figure 5 visualizes the output weights of an 784-20-784 ELM autoencoder. It can be seen that the autoencoder is able to extract digit specific patterns. ML-ELM consists of several stacked ELM autoencoder. In [2] the presented model has a topology of 784-700-700-15000-10 achieving a test error rate of 0.97 % while it took only 7.4 m to train the model on a system with an Intel i7-3740QM CPU at 2.7 GHz and 32 GB RAM.

4.4 Multilayer Ensemble ELM

The novel classifier called Multilayer Ensemble ELM (ML-EELM) presented in this paper combines concepts of CNN, ensemble models, and the ELM training algorithm. The architectural idea of shift invariance is realized by spatial subsampling. In order to classify correctly a digit, it is not necessary to know the grey-scale intensity of each pixel. Instead, it is sufficient to know the approximate position of high intensity activations on the receptive field. Hence, a convolution layer reduces the image size from 28x28 down to 26x26 by applying an aggregation function to overlapping 3x3 regions. A subsampling layer then halves the image resolution further down to 13x13 by applying an aggregation function to non-overlapping 2x2 regions. The image is then fed into an EELM model with K instances. The topology of a single instance is illustrated on Fig. 6.

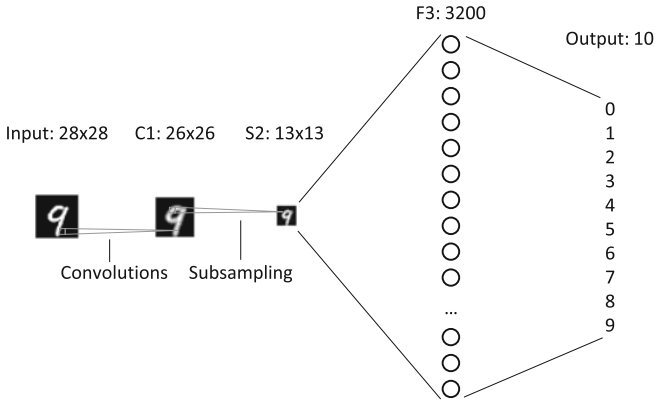


Fig. 6 ML-EELM single instance architecture

5 Experimental Setup and Evaluation

Experimental work is focused on ELM, EELM, and the novel classifier: Multilayer EELM.

Four ELM models with different number of nodes in the hidden layer were created and tested. The number of hidden layer nodes ranges from 800 to 3,200. Each model was trained and tested ten times in order to validate the robustness of the model. The training and testing time, the error rate on the training and test sample, as well as the standard deviations (SD) for all ten iterations were measured. Table 3 lists all statistics for the ELM model evaluation. A model with 3,200 hidden nodes could

Table 3 ELM evaluation results for MNIST database

Model	Hidden Layer Nodes	Iterations	Training Time (min)	SD Training Time (min)	Training Error Rate (%)	SD Training Error Rate (%)
1	800	10	0.5743	0.0289	8.9837	0.1396
2	1,600	10	1.6786	0.0262	6.2192	0.0832
3	2,400	10	3.4801	0.0326	4.7735	0.0647
4	3,200	10	6.3509	0.1024	3.8955	0.0543
Model	Hidden Layer Nodes	Iterations	Test Time (min)	SD Test Time (min)	Test Error Rate (%)	SD Test Error Rate (%)
1	800	10	0.0218	0.0019	8.9410	0.1449
2	1,600	10	0.0400	0.0033	6.6860	0.1786
3	2,400	10	0.0576	0.0043	5.6630	0.1189
4	3,200	10	0.1035	0.0079	5.0350	0.0920

Table 4 EELM evaluation results for MNIST database

Model	Hidden Layer Nodes	K	Training Time (min)	Training Error Rate (%)	Test Time (min)	Test Error Rate (%)
1	800	3	1.6222	7.8967	0.0694	7.8900
2	800	9	4.8298	6.9233	0.2174	6.7400
3	800	15	8.0541	6.6283	0.3502	6.5700
4	1600	3	4.9866	5.2933	0.1292	5.9500
5	1600	9	14.9735	4.7617	0.3843	5.2900
6	1600	15	25.2254	4.6367	0.6596	5.2400
7	2400	3	10.4370	4.1533	0.1747	5.0800
8	2400	9	31.2249	3.7583	0.5169	4.6300
9	2400	15	52.2731	3.6500	0.9365	4.5300
10	3200	3	19.5807	3.3900	0.2915	4.5900
11	3200	9	57.7446	3.0017	1.2527	4.1600
12	3200	15	97.1288	2.9550	1.5457	4.0700

achieve a test error rate of 5.04 % (SD: 0.09 %). It took on average 6.4 min (SD: 0.1 min) to train the ELM.

Twelve EELM models were trained and evaluated. The hidden layers of the models have 800 to 3,200 nodes. The parameter K for the number of independent models per EELM was set to 3, 9, and 15. Table 4 lists the training and testing times, as well as the training and test error rates for all twelve EELM models. The EELM model #12 (3,200 nodes, $K = 15$) was trained in 97 min achieving a test error of 4.07%. The model #11 (3,200 nodes, $K = 9$) however, could be trained after only 58 min having only a slightly higher test error of 4.16%.

The novel classifier ML-EELM was built and evaluated after having determined the aggregation functions applied in the convolution and subsampling layers. This was done by training a simple ELM model with one additional 2x2 subsampling layer. The average, standard deviation, maximum, and minimum value of all 4 pixels were evaluated as aggregation functions. Table 5 lists the results. The average function achieved the lowest test error rate and was subsequently used as the aggregation function in the convolution and subsampling layers of the ML-EELM model. A ML-EELM model with $K = 15$ and 3,200 hidden layer nodes achieved a test error rate as low as 2.73 % and was trained in only 96 min.

It was observed that with increasing number of hidden layer nodes, and in the case of EELM with increasing K , the test error rate decreases. The training time grows linearly with K , and exponentially with the number of hidden layer neurons.

Although the accuracy on the training data set becomes very high with more hidden layer nodes, the test error rate does not increase. The effect of improving accuracy on the training data and decreasing accuracy on the test data, known as overfitting could not be observed which speaks for the good generalization performance of ELM.

Table 5 ELM evaluation results for MNIST database with 2x2 subsampling

Aggregation	Hidden Layer Nodes	Iterations	Training Time (min)	SD Training Time (min)	Training Error Rate (%)	SD Training Error Rate (%)
Avg.	3,200	10	7.1067	0.4953	3.0330	0.0567
Std.dev.	3,200	10	6.9656	0.4880	9.5905	0.1104
Max.	3,200	10	6.7650	0.6209	3.5038	0.0284
Min.	3,200	10	7.1850	0.3130	8.7338	0.1090
Aggregation	Hidden Layer Nodes	Iterations	Training Time (min)	SD Test Time (min)	Training Error Rate (%)	SD Test Error Rate (%)
Avg.	3,200	10	0.0677	0.0052	3.9120	0.1125
Std.dev.	3,200	10	0.0766	0.0061	12.1140	0.1393
Max.	3,200	10	0.0693	0.0072	4.5010	0.1718
Min.	3,200	10	0.0704	0.0056	10.8430	0.1059

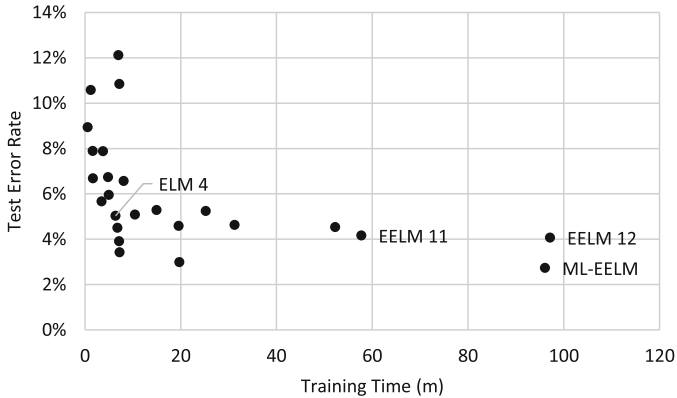


Fig. 7 Graphical evaluation of the results

Other models presented in the literature outperform with regards to the test error rate the SLFN ELM models presented in this paper. However, the experimental results confirm the initial design objectives of ELM: least human invention, high learning accuracy and fast learning speed. No training times were provided in most of the papers mentioned previously. The training time of 114 h in [5] acts as a guiding value for the training time of the other models identified in the literature.

ML-EELM, first introduced in this paper, achieves competitive test error rates on the MNIST database while requiring only fractions of training time on commodity hardware. The results confirm the conceptual ideas of CNN. Due to the convolution and subsampling layers in ML-EELM, the feature space could be reduced from 784 down to 169 leading to further improved accuracy rates (Fig. 7).

Table 6 Model comparison (NA = Not available)

Source	Model	Accuracy (%)	Training Time
[4]	CNN	0.23	NA
[5]	MLFN	0.32	114 h
[1]	MLFN	0.43	NA
[9]	CNN	0.95	NA
[2]	ML-ELM	0.97	7.4 m
This paper	ML-EELM	2.73	96 m
This paper	EELM	4.07	97 m
[9]	SLFN	4.70	NA
This paper	ELM	5.04	6.4 m

ML-ELM presented in [2] achieves slightly higher test error rates than CNN, but training the model in less time. In general, models trained with ELM outperform all other models with regards to training time. This is inline with experimental results from Huang et al. [8]. Table 6 lists a comparison of the models.

Matlab R2014a (8.3.0.532) was used for the computation of the ELM, EELM, and ML-EELM models on a Windows 7 64 bit system with 8 GB RAM and an Intel Core i5-2310 CPU at 2.90 GHz.

6 Conclusion

ELM could successfully be applied for the task of handwritten digit recognition based on the MNIST dataset. Competitive results were achieved with a test error rate of only 2.73 % with a novel multilayer ensemble ELM model presented first in this paper. While these results cannot beat the results of CNN, classifier based on ELM are relatively easy to create, have good generalization performance and most important, have fast learning speed. An ELM model with 3,200 hidden layer nodes can be trained in just about 6 min on a standard commodity desktop PC.

Moreover, Huang et al. [8] applied ELM to a variety of classification and function approximation problems and found that ELM learns up to hundreds times faster than BP. Furthermore, they observed that ELM does not face BP specific issues like local minima, improper learning rates and overfitting. The results presented in this paper confirm the initial design objectives of ELM.

ELM has great potential for applications where the training data changes frequently and hence the models need to be re-trained often. This could be the case when writing styles of different individuals have to be learned in high frequencies.

Moreover, the parallel computation possibility of EELM and ML-EELM models with a large number of K individual instances could further improve the accuracy of ELM based classifier while keeping the training time low. Wang et al. [12] have made

first efforts to implement parallel ELM using MapReduce on some classification problems. A parallel implementation of EELM or ML-EELM is recommended as possible future research.

References

1. Alonso-Weber, J., Sesmero, M., Sanchis, A.: Combining additive input noise annealing and pattern transformations for improved handwritten character recognition. *Expert Syst. Appl.* **41**(18), 8180–8188 (2014)
2. Cambria, E., Huang, G.B., Kasun, L.L.C., Zhou, H., Vong, C.M., Lin, J., Yin, J., Cai, Z., Liu, Q., Li, K., Leung, V.C., Feng, L., Ong, Y.S., Lim, M.H., Akusok, A., Lendasse, A., Corona, F., Nian, R., Miche, Y., Gastaldo, P., Zunino, R., Decherchi, S., Yang, X., Mao, K., Oh, B.S., Jeon, J., Toh, K.A., Teoh, A.B.J., Kim, J., Yu, H., Chen, Y., Liu, J.: Extreme learning machines [Trends & Controversies]. *IEEE Intell. Syst.* **28**(6), 30–59 (2013)
3. Cao, J., Lin, Z., Huang, G.B., Liu, N.: Voting based extreme learning machine. *Inf. Sci.* **185**(1), 66–77 (2012)
4. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3642–3649 (2012)
5. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep, big, simple neural nets for handwritten digit recognition. *Neural Comput.* **22**(12), 3207–3220 (2010)
6. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**(5), 359–366 (1989)
7. Huang, G., Huang, G.B., Song, S., You, K.: Trends in extreme learning machines: a review. *Neural Netw.* **61**, 32–48 (2015)
8. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. *Neurocomputing* **70**(1–3), 489–501 (2006)
9. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2323 (1998)
10. LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U., Säckinger, E., Simard, P., Vapnik, V.: Comparison of learning algorithms for handwritten digit recognition. In: International Conference on Artificial Neural Networks, pp. 53–60 (1995)
11. Schmidhuber, J.: Deep learning in neural networks: an overview. *Neural Netw.* **61**, 85–117 (2015)
12. Wang, B., Huang, S., Qiu, J., Liu, Y., Wang, G.: Parallel online sequential extreme learning machine based on MapReduce. *Neurocomputing* **149**, 224–232 (2015)