Multi-layer Online Sequential Extreme Learning Machine for Image Classification

Bilal Mirza, Stanley Kok and Fei Dong

Abstract In this paper, a multi-layer online sequential extreme learning machine (ML-OSELM) is proposed for image classification. ML-OSELM is an online sequential version of a recently proposed multi-layer extreme learning machine (ML-ELM) method for batch learning. Existing ELM-based sequential learning methods, such as state-of-the-art online sequential extreme learning machine (OS-ELM), were proposed only for single-hidden-layer networks. A distinctive feature of the new method is that it can sequentially train a multi-hidden-layer ELM network. Auto-encoders are used to perform layer-by-layer unsupervised sequential learning in ML-OSELM. We used four image classification datasets in our experiments and ML-OSELM performs better than the OS-ELM method on all of them.

Keywords Deep learning · Extreme learning machine · Feature learning · Image classification · Sequential learning

1 Introduction

Deep learning has attracted much attention in the past decade or so due to its successes in various research domains such as pattern recognition, computer vision and automatic speech recognition [1, 2]. Neural-network-based deep architectures or deep neural networks, convolutional neural networks, and deep belief networks are some of the commonly used deep learning architectures. Multiple layers in deep architectures provide multiple non-linear transformations of the original raw

B. Mirza (✉) · S. Kok · F. Dong
Information Systems Technology & Design Pillar, Singapore University of Technology & Design, Singapore, Singapore
e-mail: bilal2@e.ntu.edu.sg

S. Kok
e-mail: stanleykok@sutd.edu.sg

F. Dong
e-mail: fei_dong@mymail.sutd.edu.sg

J. Cao et al. (eds.), *Proceedings of ELM-2015 Volume 1*,
Proceedings in Adaptation, Learning and Optimization 6,
DOI 10.1007/978-3-319-28397-5_4

input data for better representation learning. Deep networks can capture higher level abstractions which most single layer networks are unable to achieve. Every layer in deep networks may learn a different representation of the input by performing feature or representation learning using efficient unsupervised or semi-supervised learning algorithms.

Restricted Boltzmann machine (RBM) [3] and auto-encoders (AE) [4] have been successfully applied for feature learning in deep networks. Examples of RBM-based deep networks include deep belief networks (DBNs) [3] and deep Boltzmann machines (DBMs) [5] while examples of auto-encoder-based deep networks are stacked auto-encoders (SAEs) [4] and stacked denoising auto-encoders (SDAEs) [5]. Multiple RBMs are stacked to create DBN and DBM networks, whereas multiple AEs are stacked to create SAE and SDAE networks. These methods have outperformed support vector machines, single-hidden-layer feed-forward neural networks and traditional multi-layer neural networks on image classification, automatic speech recognition, and other tasks. However, the time taken for learning deep networks on these big dataset applications is generally long. Recently, multi-layer extreme learning machine (ML-ELM) [6], which is based on extreme learning machine theory [7, 8], has been proposed as a computationally efficient alternative to existing state-of-the-art deep networks. ML-ELM learns significantly faster than existing deep networks, and obtains better or similar generalization performance to DBNs, SAEs, SDAEs and DBMs. In ML-ELM, extreme learning machine auto-encoders (ELM-AEs) are used for unsupervised layer-by-layer feature learning in the hidden layers.

The training of ELM-AEs is similar to that of regular ELMs except that the output in ELM-AEs is the same as the input. The extreme learning machine (ELM) algorithm [7, 8] is becoming popular in large datasets and online learning applications due to its fast learning speed. ELM provides a single step least square estimation (LSE) method for training single-hidden-layer feedforward network (SLFN) instead of using iterative gradient descent methods such as backpropagation.

ML-ELM has been proposed for image classification and pattern recognition applications, and the dimensions of the corresponding datasets are generally very high. When constrained with limited memory, both single layer ELM and ML-ELM are not suitable for big datasets. Therefore, batch learning or learning with complete datasets in one step becomes challenging. Also, graphical processing units (GPUs), widely used in scientific computing, generally have limited memory that cannot accommodate big datasets. To overcome this problem, online sequential extreme learning machine (OS-ELM) has been proposed for sequential learning from big data streams [9]. OS-ELM has become one of the standard algorithms for sequential learning due to its fast learning speed. Sequential or incremental learning algorithms store the previously learned information and update themselves only with a chunk of new data [10–13]. Once the chunk of data has been used for training, it may be discarded. OS-ELM is preferable over batch ELM not only for their reduced computational time on large datasets, but also for their ability to adapt to online learning applications. Note that OS-ELM is an online sequential version of the original single-hidden-layer extreme learning machine.

In this paper, to imbue ML-ELM with the online sequential learning advantages of OS-ELM, we propose a sequential version of ML-ELM, which we term multi-layer online sequential extreme learning machine (ML-OSELM). To our knowledge, we are the first to develop an ELM that is both multi-layered and can learn from data in an online sequential manner. In order to perform layer-by-layer unsupervised sequential learning in ML-OSELM, an online sequential extreme learning machine auto-encoder (OS-ELM-AE) is also proposed. ML-OSELM resembles deep networks since it stacks on top of OS-ELM-AE to create a multi-layer neural network. This paper is organized as follows. Section 2 discusses the preliminaries. Section 3 presents the details of the new multi-layer online sequential learning method. This is followed by experiments for validating the performance of the proposed framework in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Preliminaries

2.1 Extreme Learning Machine (ELM) and Online Sequential ELM (OS-ELM) Methods

Extreme learning machine [7] provides a single step least squares error (LSE) estimate solution for a single-hidden-layer feedforward network (SLFN). With ELM there is no need to tune the hidden layer of the SLFN as in traditional gradient-based algorithms. ELM randomly assigns weights and biases in the hidden layer while the output weights connecting the hidden layer and the output layer are determined using the LSE method.

Consider a q class training dataset $\{x_i, y_i\}$, $i = 1, \dots, N$ and $y_i \in R^q$. $x_i \in R^d$ is a d-dimensional data point. The SLFN output with L hidden nodes is given by

$$o_i = \Sigma_{j=1}^{L} \beta_j G(a_j, b_j, x_i), i = 1, \dots, N \tag{1}$$

where a_j and b_j, $j = 1, \dots, L$ are the jth hidden node's weights and biases respectively, and they are assigned randomly, independent of the training data. $\beta_j \in R^q$ is the output weight vector connecting the jth hidden node to the output nodes and $G(x)$ can be any infinitely differentiable activation function such as the sigmoidal function or radial basis function in the hidden layer.

The N equations in (1) can be written in a compact form as below.

$$O = H\beta \tag{2}$$

where $H = \begin{bmatrix} G(a_1, b_1, x_1) & \dots & G(a_L, b_L, x_1) \\ \vdots & \dots & \vdots \\ G(a_1, b_1, x_N) & \dots & G(a_L, b_L, x_N) \end{bmatrix}_{N \times L}$

is called the hidden layer output matrix, H_{ij} represents the jth hidden node output corresponding to the input x_i, $\beta = [\beta_1, \beta_2, \dots, \beta_L]^T$ and $O = [o_1, o_2, \dots, o_N]^T$.

In order to find the output weight matrix β which minimizes the cost function $\|O - Y\|$, a LSE solution of (2) is obtained as below

$$\beta = H^{\dagger} Y = (H^T H)^{-1} H^T Y \tag{3}$$

where $Y = [y_1, y_2, \dots, y_N]^T$ and $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H. This closed-form single step LSE solution is referred to as an extreme learning machine [7].

Online sequential extreme learning machine (OS-ELM) [9] has been proposed as an incremental version of the batch ELM. OS-ELM achieves better generalization performance than the previous algorithms proposed for SLFN and at a much faster learning speed. It can learn from data one example at a time as well as chunk-by-chunk (with a fixed or varying chunk size). Only the newly arrived samples are used at any given time so the examples that have already been used in the learning procedure can be discarded. The learning in OS-ELM consists of two phases.

Step 1: Initialization
A small portion of training data $n_0 = \{x_i, y_i\}$, $i = 1, \dots, N_0$ with $N_0 \in N$ is considered for initializing the network. The initial output weight matrix is calculated according to the ELM algorithm by randomly assigning weights a_j and bias b_j, $j = 1, \dots, L$ as follows

$$\beta^{(0)} = P_0 H_0^T Y_0 \tag{4}$$

where $P_0 = (H_0^T H_0)^{-1}$ and H_0 is the initial hidden layer output matrix.

It is recommended that the number of initial training samples should be greater than or equal to the number of hidden neurons. With this setting the generalization performance of online sequential ELM reaches that of the batch ELM.

Step 2: Sequential Learning
Upon the arrival of a new set of observations $n_{k+1} = \{x_i, y_i\}$, $i = (\Sigma_{l=0}^{k} N_l) + 1, \dots, \Sigma_{l=0}^{k+1} N_l$, *i.e.*, the $(k+1)$th chunk of data, we first compute the partial hidden layer output matrix H_{k+1}. N_{k+1} is the number of samples in the $(k+1)$th chunk. Then by using the output weight update equation shown below, we calculate the output weight matrix β^{k+1} with $Y_{k+1} = [y_{(\Sigma_{l=0}^{k} N_l)+1}, \dots, y_{\Sigma_{l=0}^{k+1} N_l}]^T$.

$$\begin{aligned} P_{k+1} &= P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k \\ \beta^{(k+1)} &= \beta^{(k)} + P_{k+1} H_{k+1}^T (Y_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned} \tag{5}$$

Each time a new chunk of data arrives, the output weight matrix is updated according to (5). Note the one-by-one learning can be considered a special case of chunk-by-chunk learning when the chunk size is set to 1 and the matrices in (5) become vectors.

2.2 Multi-layer Extreme Learning Machine (ML-ELM)

In multi-layer neural networks (ML-NN), the hidden layer weights are initialized by layer-by-layer unsupervised learning and then the whole network is fine-tuned using backpropagation. ML-NN performs better with layer-by-layer unsupervised learning as compared to only using backpropagation. However, fine-tuning is avoided in such deep networks using the recently proposed multi-layer extreme learning machine (ML-ELM) method.

ML-ELM hidden layer weights are initialized randomly using extreme learning machine auto-encoders (ELM-AEs). ELM-AE performs layer-by-layer unsupervised learning. ELM-AE is trained differently from ELM in that the output is set to be equal to the input, *i.e.*, $Y = X$ in (3) (see Fig. 1), and hidden layer weights and biases are chosen to be orthogonal to the random weights in ELM. Orthogonalization of these weights tends to result in better generalization performance. Note that ELM-AE output weights are obtained analytically unlike RBMs and other auto-encoders, which require iterative algorithms.

ELM-AE's main objective is to transform features from input data space to lower or higher dimensional feature space. Since ELM is a universal approximator [8], ELM-AE is also a universal approximator.

Once the hidden layer weights are learnt using ELM-AE, the output weights connecting the last hidden layer to the output layer of ML-ELM are determined analytically using (3).

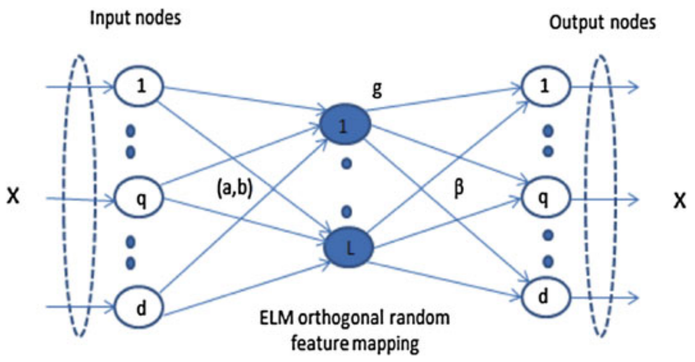

Fig. 1 ELM-AE uses the same architecture as original ELM with the exception that the target output '**x**' is the same as input. Here 'g' is the activation function, (a, b) represents random weights and biases, and β represents output weights

3 Proposed Method

In this section, we propose an online sequential version of the multi-layer extreme learning machine, which we term multi-layer online sequential extreme learning machine (ML-OSELM). In Sect. 3.1, we propose an online sequential extreme learning machine auto-encoder (OS-ELM-AE) for feature or representation learning from sequential data streams. In Sect. 3.2, the proposed OS-ELM-AE is used to perform layer-by-layer unsupervised sequential learning in multi-layer extreme learning machines. The deep sequential learning in ML-OSELM is performed by stacking several OS-ELM-AEs. Note that the network architectures of OS-ELM-AE and ML-OSELM are identical to ELM-AE (Fig. 1) and ML-ELM (Fig. 2) respectively. In the following, we will discuss how these architectures are used for sequential learning.

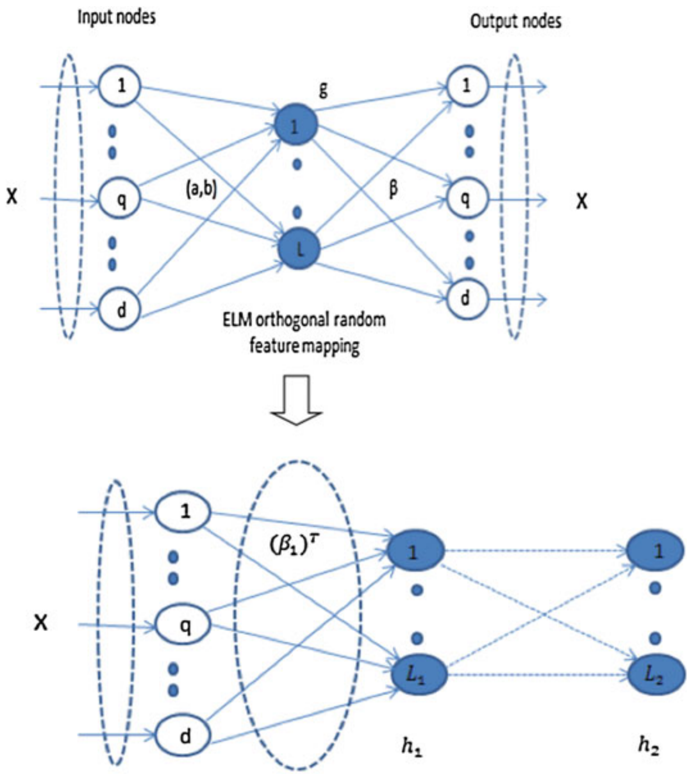

Fig. 2 The weights of each hidden layer in ML-ELM are determined using ELM-AE. $(\beta_1)^T$ is responsible for feature learning at the first hidden layer of ML-ELM

3.1 Online Sequential Extreme Learning Machine Auto-Encoder (OS-ELM-AE)

OS-ELM-AE is a special case of OS-ELM where output is the same as the input at every time step. The hidden layer weights are randomly generated, as in OS-ELM, but in OS-ELM-AE orthogonal of random weights and biases are used. Learning in OS-ELM-AE is done in two phases as described below.
Step 1: Initialization

The OS-ELM-AE is initialized with a portion of training data $n_0 = \{x_i, y_i\}$, $i = 1, \dots, N_0$ with $N_0 \in N$. The initial output weights $\beta^{(0)}$ of OS-ELM-AE is given as

$$\beta^{(0)} = P_0 H_0^T X_0 \tag{6}$$

where $X_0 = [x_1, x_2, \dots, x_{N_0}]^T$, $P_0 = (H_0^T H_0)^{-1}$ and H_0 is the hidden layer output matrix.

Step 2: Sequential Learning
With the arrival of a new chunk of training data, the recursive least square equation in (5) is used, but with input set equal to output as below

$$\begin{aligned} P_{k+1} &= P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k \\ \beta^{(k+1)} &= \beta^{(k)} + P_{k+1} H_{k+1}^T (X_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned} \tag{7}$$

The output weight β in OS-ELM-AE is responsible of learning the transformation from input space to feature space. Note that (7) is different from (5) since Y_{k+1} is replaced with X_{k+1}.

3.2 Multi-layer Online Sequential Extreme Learning Machine (ML-OSELM)

The OS-ELM-AE method proposed in the previous subsection is now applied to layer-by-layer unsupervised sequential learning in a multi-layer online sequential extreme learning machine. All the hidden layer are initialized using (6) and then sequentially trained with the arrival of new data using (7). The hidden layer output matrix corresponding to hidden layer m at time step k is given as

$$H_k^m = g((\beta_m^{(k)})^T H_k^{m-1})$$

where g can be any activation function that can be used with ELMs [7, 8] and $\beta_m^{(k)}$ is the output weight matrix obtained using OS-ELM-AE for layer m at time step k.

The input or data layer can be considered as the *zero*th hidden layer where $m = 0$. Assuming a total of p hidden layers in the network, the output weight matrix in ML-OSELM connecting the last hidden layer to the output layer is obtained as follows,

Step 1: Initialization

$$\beta^{(0)} = P_0 H_0^T Y_0 \tag{8}$$

where $H_0 = H_0^p = g((\beta_p^{(0)})^T H_0^{p-1})$.

Step 2: Sequential Learning

$$\begin{aligned} P_{k+1} &= P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k \\ \beta^{(k+1)} &= \beta^{(k)} + P_{k+1} H_{k+1}^T (Y_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned} \tag{9}$$

where $H_{k+1} = H_{k+1}^p = g((\beta_p^{(k+1)})^T H_{k+1}^{p-1})$.
The training algorithm of ML-OSELM is summarized in Algorithm 1.

Algorithm 1 Multi-layer online sequential extreme learning machine (ML-OSELM) algorithm

```
Input: ${x_i, y_i}, x_i \in R^d$ and $y_i \in R^q, i = 1, \dots, N$.
Output: $\beta^{(k+1)}$

Initialization:
Initial data set: $n_o = \{x_i, y_i\}, x_i \in R^d$ and $y_i \in R^q, i = 1, \dots, N_0$ with $N_0 \in N$.

for $m = 1 \rightarrow p$ do                                   % number of hidden layers %
    $H_0^m = g((\beta_m^{(0)})^T H_0^{m-1})$
    where $\beta_m^{(0)}$ is obtained using (6)
end for

     $\beta^{(0)} = P_0 H_0^T Y_0$
     where $H_0 = H_0^p$

Sequential Learning:
for $k = 0 \rightarrow K$ do                                    % number of time steps %
    for $m = 1$ to $p$ do
        $H_{k+1}^m = g((\beta_m^{(k+1)})^T H_{k+1}^{m-1})$
        where $\beta_m^{(k+1)}$ is obtained using (7)
    end for

    $\beta^{(k+1)} = \beta^{(k)} + P_{k+1} H_{k+1}^T (Y_{k+1} - H_{k+1} \beta^{(k)})$
    where $H_{k+1} = H_{k+1}^p$
end for
```

4 Experiments

In this section, we use four image classification datasets for the performance evaluation of ML-OSELM. The first dataset is CIFAR10 [14], the second CALTECH [15], the third Olivetti faces [16] and the last UMIST [17]. CIFAR10 consists of 50,000 training and 10,000 testing samples belonging to 10 different categories. We use a feature extraction pipeline as described in [18, 19]. The pipeline extracts 6×6 pixel patches from the training set images, performs ZCA whitening of those patches, runs K-means for 50 rounds, and then normalizes its dictionary to have zero mean and unit variance. The final feature vector is of dimension $G \times G \times K$, where K is the dictionary size and $G \times G$ represents the size of the pooling grid. We set $K = 400$ and $G = 4$ in our experiments. We compare the results obtained by multi-layer OS-ELM with that of single layer OS-ELM and also ML-ELM (batch mode) under identical settings.

For the CALTECH dataset, we used 5 classes representing faces, butterfly, crocodile, camera and cell phone. For each class, we set aside one third of the images (up to 50) for testing and used the rest for training. The pixel values represent gray scale intensities which are normalized to have zero mean and unit variance. Similar to [20], we used fixed-size square images in all the categories even though the original sizes may vary in different categories. The original images are rescaled so that the longer side is of length 100, and then we used the inner 64×64 portions for our experiments.

Out of 400 samples in Olivetti faces, we used 300 for training and 100 for testing. Each sample is a 64×64 gray scale image. It represents 40 unique people with 10 images each, all frontal and with a slight tilt of the head.

The UMIST database has 565 total images of 20 different people with 19–36 images per person. 400 samples are used for training while 175 are used for testing. Each sample is a 112×92 gray scale image. The subjects differ in race, gender and appearances. The dataset covers various angles from left profile to right profile.

We conducted all the experiments on a high performance computer with Xeon-E7-4870 2.4 GHz processors, 256 GBytes of RAM, and running Matlab 2013b. For consistency, we used a three-hidden-layer (3000-4000-5000) network structure for multi-layer ELM in all the datasets. For OS-ELM, the number of hidden neurons is set to 5000, *i.e.*, the same as the last layer in the multi-layer network. The sizes of each initialization set and chunk of samples at each time step in sequential learning are respectively set to 15000 and 500 for CIFAR10, 200 and 50 for CALTECH, 100 and 50 for Olivetti faces, and 150 and 50 for UMIST faces. Sigmoid is used as the activation function throughout the experiments. The classification accuracy comparison between ML-OSELM and OS-ELM on the four datasets is given in Table 1. The results represent average accuracy over 20 runs with different initialization, training and testing sets.

We use the two-sample t-test to see whether the results obtained by ML-OSELM are statistically significantly better than those obtained by single layer OS-ELM. The null hypothesis is rejected at $\alpha = 0.01$ for UMIST and CIFAR10 with p values of

Table 1 Accuracy comparisons between multi-layer OS-ELM (ML-OSELM) and single-layer OS-ELM

Dataset	ML-OSELM	OS-ELM
UMIST	**98.03**	96.51
Olivetti	**92.81**	90.23
CIFAR10	**64.96**	64.01
CALTECH	**59.22**	59.07

Table 2 Accuracy comparisons between multi-layer OS-ELM (ML-OSELM) and batch multi-layer ML-ELM

Dataset	ML-OSELM	ML-ELM
UMIST	98.03	**98.83**
Olivetti	92.81	**94.91**
CIFAR10	64.96	**66.41**
CALTECH	59.22	**59.57**

0.0054 and 1.35E-5 respectively. For Olivetti faces, the null hypothesis is rejected at $\alpha = 0.05$ with a p value of 0.043. For CALTECH, p value is 0.735 and the null hypothesis is not rejected.

With these statistical significances test, ML-OSELM is found to be superior to OS-ELM on the UMIST, CIFAR10, CALTECH datasets.

The aim of sequential learning methods is to achieve a performance similar to that of batch learning methods when constrained with limited memory. It can be observed from Table 2 that ML-OSELM results are competitive against those obtained by ML-ELM (batch mode).

5 Summary and Future Work

We propose the multi-layer online sequential extreme learning machine (ML-OSELM). Our empirical results show that by using multiple layers, our proposed ML-OSELM outperforms the state-of-the-art single-layer online sequential ELM. Further, our ML-OSELM achieves competitive results against a batch multi-layer ELM that has the advantage of having the full dataset available for training.

As future work, we want to come up with a method to find the optimal number of hidden layers in ML-OSELM, incorporate concept drift learning into ML-OSELM, and compare it against the recently proposed concept drift learning method for single-hidden-layer ELMs [21].

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013)
2. Bengio, Y.: Learning deep architectures for AI. Found. Trends Mach. Learn. **2**(1), 1–127 (2009)
3. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
4. Vincent, P., et al.: Stacked Denoising Autoencoders, Learning Useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
5. Salakhutdinov, R., Larochelle, H.: Efficient learning of deep Boltzmann machines. J. Mach. Learn. Res. **9**, 693–700 (2010)
6. Kasun, L.L.C., Zhou, H., Huang, G.B., Vong, C.M.: Representational learning with ELMs for big data. IEEE intell. Syst. **28**(6), 31–34 (2013)
7. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
8. Huang, G.-B., et al.: Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. **42**(2), 513–529 (2012)
9. Lian, N.Y., Huang, G.B., Saratchandran, P., Sundararajan, N.: A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans. Neural Netw. **17**(6), 1411–1423 (2006)
10. Mirza, B., Lin, Z., Toh, K.A.: Weighted online sequential extreme learning machine for class imbalance learning. Neural Process. Lett. **38**(3), 465–486 (2013)
11. Kim, Y., Toh, K.A., Teoh, A.B.J., Eng, H.L., Yau, W.Y.: An online learning network for biometric scores fusion. Neurocomputing **102**, 65–77 (2013)
12. Gu, Y., Liu, J., Chen, Y., Jiang, X., Yu, H.: TOSELM: timeliness online sequential extreme learning machine. Neurocomputing **128**, 119–127 (2014)
13. Zhao, J., Wang, Z., Park, D.S.: Online sequential extreme learning machine with forgetting mechanism. Neurocomputing **87**, 79–89 (2012)
14. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Computer Science Department, University of Toronto, Techinical Report, vol. 1, no. 4, 2009
15. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categorie, IEEE. CVPR 2004, Workshop on Generative-Model Based Vision, 2004
16. Samaria, F., Harter, A.: Parameterisation of a stochastic model for human face identification. In: Proceedings of 2nd IEEE Workshop on Applications of Computer Vision, Sarasota FL, Dec 1994
17. Graham, D.B., Allinson, N.M.: Characterizing Virtual Eigensignatures for General Purpose Face Recognition. Face Recognition: From Theory to Applications, NATO ASI Series F, Computer and Systems Sciences, vol. 163, pp. 446–456 (1998)
18. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: International Conference on Artificial Intelligence and Statistics, 2011
19. Jia, Y., Huang, C., Darrell, T.: Beyond spatial pyramids: receptive field learning for pooled image features. CVPR, 2012
20. Poon, H., Domingos, P.: Sum-product networks: a new deep architecture. In: IEEE Computer Vision Workshops (ICCV Workshops), pp. 689–690 (2011)
21. Mirza, B., Lin, Z., Nan, L.: Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. Neurocomputing **149**(A), 316–329 (2015)