MULTI LAYER MULTI OBJECTIVE EXTREME LEARNING MACHINE

Chamara Kasun Liyanaarachchi Lekamalage[1], *Kang Song*[1], *Guang-Bin Huang*[1], *Dongshun Cui*[1] *and Ken Liang*[2]

[1]Nanyang Technological University
[2]Delta Electronics
Emails: {chamarakasun,songkang,egbhuang,dcui002}@ntu.edu.sg and ken.liang@deltaww.com

ABSTRACT

Fully connected multi layer neural networks such as Deep Boltzmann Machines (DBM) performs better than fully connected single layer neural networks in image classification tasks and has a smaller number of hidden layer neurons than Extreme Learning Machine (ELM) based fully connected multi layer neural networks such as Multi Layer ELM (ML-ELM) and Hierarchical ELM (H-ELM) However, ML-ELM and H-ELM has a smaller training time than DBM. This paper introduces a fully connected multi layer neural network referred to as Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM) which uses a multi objective formulation to pass the label and non-linear information in order to learn a network model which has a similar number of hidden layer parameters as DBM and smaller training time than DBM. The experimental results show that MLMO-ELM outperforms DBM, ML-ELM and H-ELM on OCR and NORB datasets.

***Index Terms*—** Extreme Learning Machine (ELM), fully connected multi layer neural networks, Deep Boltzmann Machine (DBM), image classification.

1. INTRODUCTION

Traditional fully connected multi layer learning algorithm such as Deep Boltzmann Machines (DBM) [1, 2, 3] consists of two learning steps: 1) learn the hidden layer parameters using an unsupervised feature learning algorithm such as Restricted Bolztmann Machine (RBM) [4]; 2) fine tune the multi layer neural network using Back-Propagation (BP) [5]. As DBM and RBM learn non-linear features in the first step and pass the label information using BP in the second step, DBM and RBM learn a network with small number of hidden layer parameters than Extreme Learning Machine (ELM) based fully connected multi layer neural networks such as Multi Layer Extreme Learning Machine (ML-ELM) [6] and Hierarchical Extreme Learning Machine (H-ELM) [7]. ML-ELM and ELM-AE consist of one learning step, which is to learn the hidden parameters using Extreme Learning Machine Auto-Encoder (ELM-AE) [6, 8] and does not fine tune the network using BP. As ELM-AE is unsupervised feature learning algorithm which learns the hidden layer parameters using linear regression, ML-ELM and H-ELM has a network architecture with a large number of hidden layer parameters and small training time. ELM-AE require a large amount of hidden layer parameters due to the following: 1) linear feature learning algorithm; 2) is an unsupervised feature learning algorithm.

Representing non-linear features using a linear feature learning algorithm requires a large amount of parameters. Hence introducing a non-linear feature learning algorithm will reduce the number of parameters required. Furthermore supervised feature learning algorithms can use label information to remove unwanted features and reduce the number of hidden layer parameters.

This paper introduces Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM) fully connected multi layer neural network which uses a one step learning process to learn the hidden layer parameters. Multi Objective Extreme Learning Machine Auto-Encoder (MO-ELMAE) learns the hidden layer parameters of MLMO-ELM. In contrast to ELM-AE which is a single objective algorithm, MO-ELMAE has three objectives. These three objectives of MO-ELMAE are: 1) traditional ELM-AE objective; 2) objective to learn non-linear features; 3) objective to learn label information. Hence MO-ELMAE allows MLMO-ELM to learn a small number of hidden layer parameters at a lower training time.

2. LITERATURE REVIEW

2.1. Extreme Learning Machine (ELM)

Extreme Learning Machine (ELM) theories [9, 10, 11, 12, 13, 14, 15] shows that the hidden layer parameters of a Single Layer Feed-forward Neural-network (SLFN) can be randomly chosen from any distribution and not tuned. Hence only the output weights of ELM must be calculated. For N training samples $\{(\mathbf{x}_j, \mathbf{t}_j)\}_{j=1}^{N}$, ELM output is calculated as:

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

where L are the number of hidden neurons of ELM, $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_L]^T$ is the output weights, $h_i(\mathbf{x}_j)$ is the output of the i-th hidden neuron and $\mathbf{x}_j = [x_1, \cdots, x_d]$ is the j-th input data sample with d number of input neurons. ELM feature mapping $h_i(\mathbf{x}_j) = g(\mathbf{a}_i, \mathbf{x}, \mathbf{b}_i)$ is calculated using $g(.)$ piecewise non-linear activation function, $\mathbf{a}_i$ and $\mathbf{b}_i$ fixed random parameters. ELM solves the following learning problem [14, 15]:

$$\text{Minimize: } ||\boldsymbol{\beta}||_p^{\sigma_1} + C||\mathbf{H}\boldsymbol{\beta} - \mathbf{T}||_q^{\sigma_2} \tag{2}$$

where $\sigma_1 > 0$, $\sigma_2 > 0$, $p, q = 0, \frac{1}{2}, 1, 2, \cdots, +\infty$, C is a user given parameter, $\mathbf{T} = [\mathbf{t}_1, \cdots, \mathbf{t}_N]^{\mathrm{T}}$ is the target data $\mathbf{t} \in \mathbf{R}^m$ with m number of output neurons and $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \cdots, \mathbf{h}(\mathbf{x}_N)]^{\mathrm{T}}$ is the ELM feature mapping. When C is infinitely large and $\sigma_1 = \sigma_2 = p = q = 2$, the solution to Equation (2) is:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{3}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$.

2.2. Extreme Learning Machine Auto-Encoder (ELM-AE))

Extreme Learning Machine Auto-Encoder (ELM-AE) is used to create ELM based fully connected multi layer networks and can learn features in three different representations: 1) Compressed representation: in this representation ELM-AE has less number of hidden neurons than input neurons and performs dimension reduction $d > L$; 2) Equal dimension representation: in this representation ELM-AE has the same number of hidden neurons as the input neurons $d = L$; 3) Sparse representation: in this representation ELM-AE has larger number of hidden neurons than input neurons $d < L$.

For compressed ($d > L$) and equal dimension ($d = L$) representation representation the ELM feature mapping is calculated as: $\mathbf{h}(\mathbf{x}_j) = g(\mathbf{x}_j\mathbf{A} + \mathbf{b})$ where hidden layer parameters are orthogonal random $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ and $\mathbf{b}^T\mathbf{b} = 1$. For sparse ($d < L$) representation ELM feature napping is calculated as: $\mathbf{h}(\mathbf{x}_j) = g(\mathbf{x}_j\mathbf{A}+\mathbf{b})$ where hidden layer parameters are orthogonal random $\mathbf{A}\mathbf{A}^T = \mathbf{I}$ and $\mathbf{b}\mathbf{b}^T = 1$.

For compressed ($d > L$) and sparse ($d < L$) representation ELM-AE solves the following learning problems:

$$\text{Minimize: } ||\boldsymbol{\beta}||_2^2 + C||\mathbf{H}\boldsymbol{\beta} - \mathbf{X}||_2^2 \tag{4}$$

For equal dimension ($d = L$) representation ELM-AE solves the following learning problems:

$$\begin{aligned} &\text{Minimize: } ||\mathbf{H}\boldsymbol{\beta} - \mathbf{X}||_2^2 \\ &\text{Subject to: } \boldsymbol{\beta}^T\boldsymbol{\beta} = \mathbf{I} \end{aligned} \tag{5}$$

2.3. Multi Layer Extreme Learning Machine (ML-ELM)

ELM-AE learns the hidden layer parameters of Multi Layer Extreme Learning Machine (ML-ELM). Figure 1 shows that ELM-AE uses input data $\mathbf{X}$ to learn the first hidden layer parameters of ML-ELM. ELM-AE uses the first hidden layer output $\mathbf{H}^1$ of ML-ELM to learn the second hidden layer parameters of ML-ELM. ELM-AE uses the p-th hidden layer output $\mathbf{H}^p$ of ML-ELM to learn the p + 1-th hidden later parameters of ML-ELM. Ridge regression learns the output layer parameters of ML-ELM.

3. MULTI LAYER MULTI OBJECTIVE EXTREME LEARNING MACHINE (MLMO-ELM)

Multi Objective Extreme Learning Machine Auto-Encoder (MO-ELMAE) learns the hidden layer parameters of Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM). Ridge regression learns the output layer weights of MLMO-ELM.

3.1. Multi Objective Extreme Learning Machine Auto-Encoder (MO-ELMAE)

Figure 2 shows that MO-ELMAE consists of three networks with two set of learn-able weights: 1) $\boldsymbol{\beta}_X$; 2) $\boldsymbol{\beta}_T$. Hence MO-ELMAE performs the following tasks: 1) learn weights which can capture important information of input data; 2) learn non-linear weights by using the euclidean distance information of input data; 3) learn weights with label information. These tasks can be formulated as a multi objective function as:

$$\begin{aligned} E = min_{\boldsymbol{\beta}_X, \boldsymbol{\beta}_T} &\frac{1}{2}||\mathbf{H}\boldsymbol{\beta}_X - \mathbf{X}||_2^2 + \frac{1}{2}||g(\mathbf{X}\boldsymbol{\beta}_X^T) - \mathbf{X}\mathbf{A}||_2^2 \\ &+ \frac{1}{2}||g(\mathbf{X}\boldsymbol{\beta}_X^T) - \mathbf{T}\boldsymbol{\beta}_T||_2^2 \\ &+ \frac{C_X}{2}||\boldsymbol{\beta}_X||_2^2 + \frac{C_T}{2}||\boldsymbol{\beta}_T||_2^2 \end{aligned} \tag{6}$$

$\mathbf{H}$ is calculated as:

$$\begin{aligned} \mathbf{h}(\mathbf{x}_j) &= g(\mathbf{x}_j\mathbf{A} + \mathbf{b}) = [h_1(\mathbf{x}_j), \cdots, h_L(\mathbf{x}_j)] \\ &= [\mathbf{a}_1 \cdot \mathbf{x}_j + b_1, \cdots, \mathbf{a}_L \cdot \mathbf{x}_j + b_L] \end{aligned} \tag{7}$$

$$\begin{array}{cc} \text{if } d >= L & \text{if } d < L \\ \mathbf{A}^T\mathbf{A} = \mathbf{I} & \mathbf{A}\mathbf{A}^T = \mathbf{I} \\ \mathbf{b}^T\mathbf{b} = 1 & \mathbf{b}\mathbf{b}^T = 1 \end{array}$$

where $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_L]$ is the orthogonal random weight matrix between the input nodes and hidden nodes, $\mathbf{b} = [b_1, \cdots, b_L]$ is the orthogonal bias vector of the hidden nodes, $g(.)$ is a non-linear activation function (in this paper we used the sigmoid activation function), $\boldsymbol{\beta}_X = [\boldsymbol{\beta}_{X1}, \cdots, \boldsymbol{\beta}_{XL}]^T$ is MO-ELMAE weights with respect to input data $\mathbf{X}$ and $\boldsymbol{\beta}_T = [\boldsymbol{\beta}_{T1}, \cdots, \boldsymbol{\beta}_{TL}]^T$ is MO-ELMAE weights with respect to targets $\mathbf{T}$. MO-ELMAE weights $\boldsymbol{\beta}_T$ $\boldsymbol{\beta}_X$ can be calculated using alternative optimization as shown in Algorithm 1.

The output weights $\boldsymbol{\beta}_T$ can be calculated as:

$$\boldsymbol{\beta}_T = \left(C_T + \mathbf{H}_X^T\mathbf{H}_X\right)^{-1}\mathbf{H}_X^T\mathbf{T} \tag{8}$$

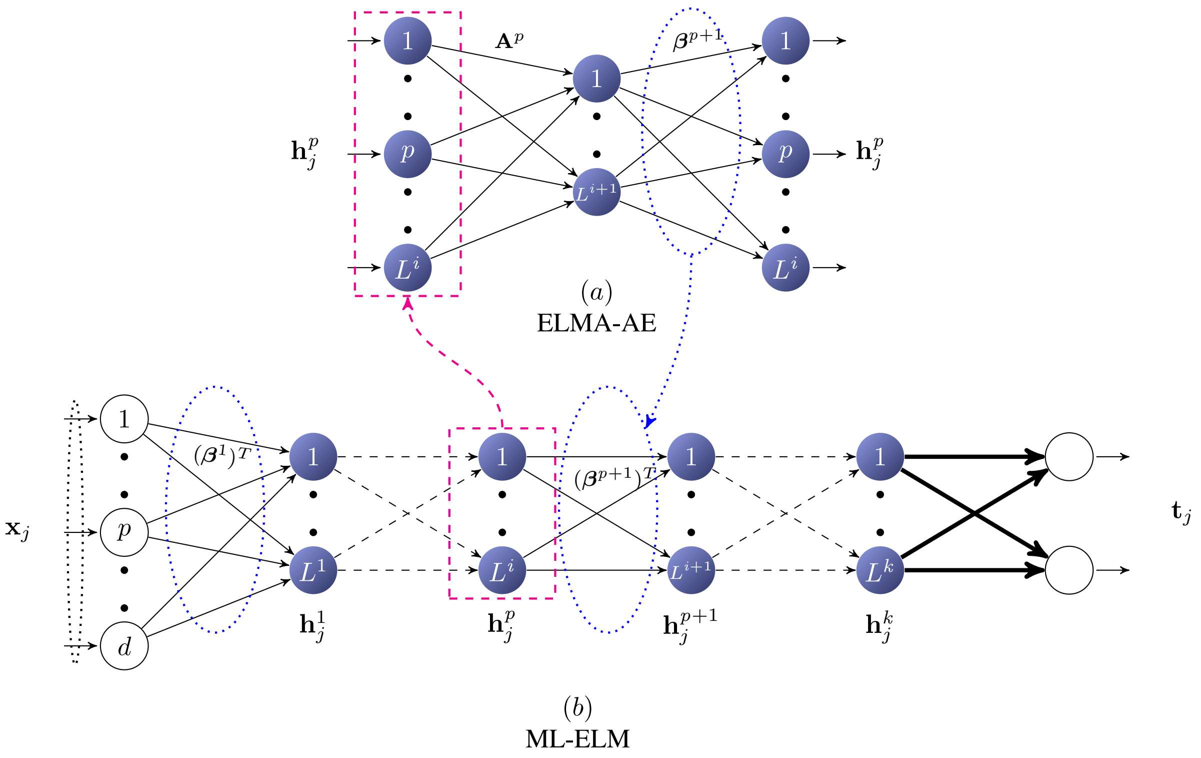

Fig. 1. ML-ELM network structure. (a) ELM-AE learns the p-th layer weights $\boldsymbol{\beta}^{p+1}$ of ML-ELM using p-th hidden layer output $\mathbf{h}_j^p$. (b) Ridge regression learns the ML-ELM output layer weights.

Algorithm 1 MO-ELMAE learning algorithm

```
1: while $iter < max\_iter$ do
2:     Calculate $\boldsymbol{\beta}_T$
3:     Calculate $\boldsymbol{\beta}_X$
4: end while
```

where $\mathbf{H}_X = g(\mathbf{X}\boldsymbol{\beta}_X^T)$.

As $\boldsymbol{\beta}_X$ cannot be calculated analytically, E is differentiated with respect to $\boldsymbol{\beta}_X$ to calculate $\frac{\delta E}{\delta \boldsymbol{\beta}_X}$ as:

$$\begin{aligned}\frac{\delta E}{\delta \boldsymbol{\beta}_X} &= \mathbf{H}^T(\mathbf{H}\boldsymbol{\beta}_X - \mathbf{X}) \\ &+ \mathbf{H}_X.(1-\mathbf{H}_X).(\mathbf{H}_X\mathbf{A}\mathbf{A}^T - \mathbf{X}\mathbf{A}) \\ &+ \mathbf{H}_X.(1-\mathbf{H}_X).(\mathbf{H}_X\boldsymbol{\beta}_T{\boldsymbol{\beta}_T}^T - \mathbf{T}\boldsymbol{\beta}_T) \\ &+ C_X\end{aligned} \tag{9}$$

$\boldsymbol{\beta}_X$ is calculated iteratively as:

$$\boldsymbol{\beta}_X = \boldsymbol{\beta}_X - \lambda\frac{\delta E}{\delta \boldsymbol{\beta}_X} \tag{10}$$

where λ is the learning rate. We use the off the shelf solver *minfunc* [16] to calculate MO-ELMAE weights $\boldsymbol{\beta}_X$ using Limited memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm. L-BFGS algorithm finds learning rate λ and must not be provided by the user.

3.2. Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM)

MO-ELMAE learns the hidden layer parameters of Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM). Figure 2 shows that MO-ELMAE uses input data $\mathbf{X}$ to learn the first hidden layer parameters of MLMO-ELM. MO-ELMAE uses the first hidden layer output $\mathbf{H}^1$ of MLMO-ELM to learn the second hidden layer parameters of MLMO-ELM. MO-ELMAE uses the p-th hidden layer output $\mathbf{H}^p$ of MLMO-ELM to learn the $p+1$-th hidden later parameters of MLMO-ELM. Ridge regression learns the output layer parameters of MLMO-ELM.

4. EXPERIMENTS

We perform the experiments on two benchmark datasets: 1) NORB object recognition dataset [17]; 2) Optical Character Recognition (OCR) dataset [18]. NORB dataset contains stereo images of size $2 \times 96 \times 96$ and for the experiments, NORB images were down-sampled to $2 \times 32 \times 32$. NORB dataset contains 24300 training samples and 24300 testing samples representing 5 object classes. OCR dataset contains 42152 training samples and 10000 testing samples representing 26 classes from a-z characters. OCR data are binary pixel images of 16×8. The experiments were carried out in a

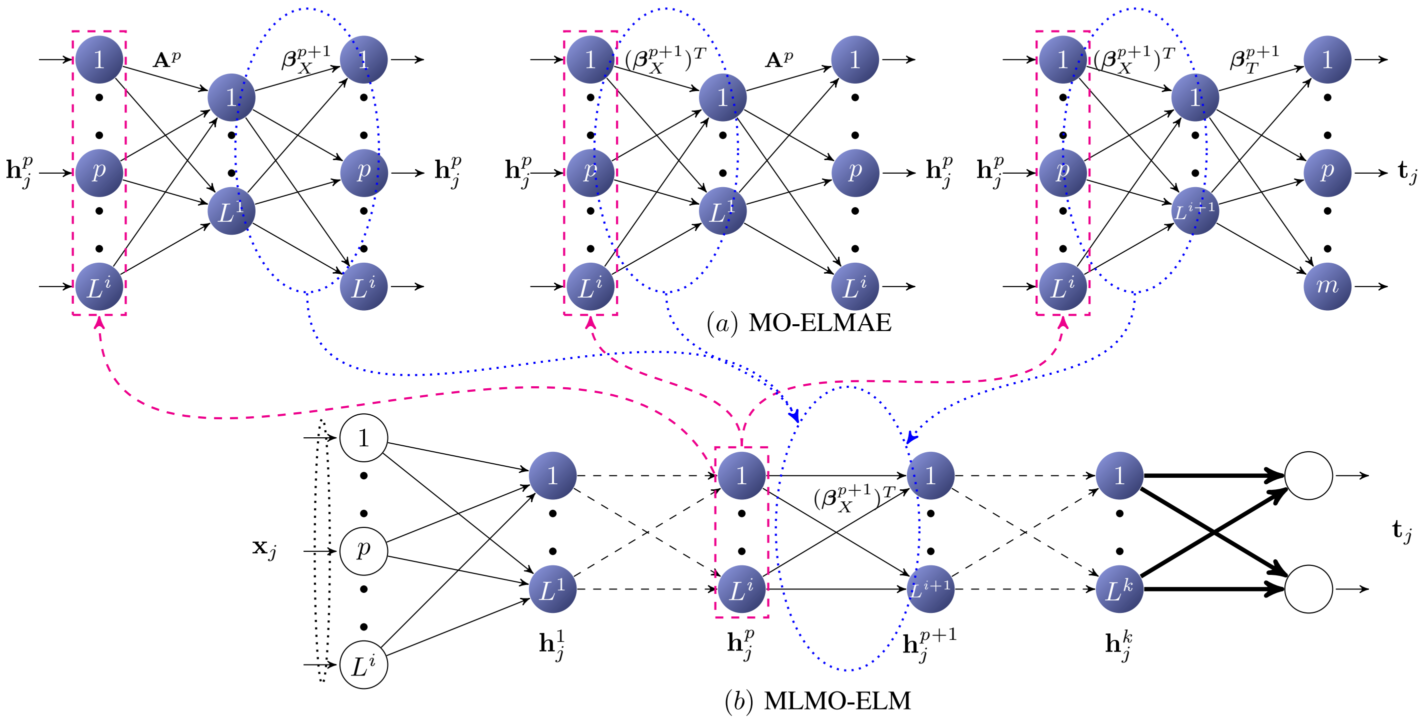

Fig. 2. MLMO-ELM network structure. MO-ELMAE consist of three networks which shares $\boldsymbol{\beta}_X^{p+1}$ weights. While ELM-AE consist of only one neural network. (a) MO-ELMAE learns the p-th layer weights $\boldsymbol{\beta}_X^{p+1}$ of MLMO-ELM using p-th hidden layer output $\mathbf{h}_j^p$. (b) Ridge regression learns the MLMO-ELM output layer weights.

workstation with a 2.6 Ghz Xeon E5-2630 v2 processor and 512 GB ram running matlab 2016a. Average testing accuracy and average training time of ten trials are reported for the MLMO-ELM algorithm. Table 1 shows that MLMO-ELM achieves better testing results than DBM, ML-ELM and H-ELM for OCR and NORB datasets. Furthermore, the number hidden layer parameters required by MLMO-ELM is lower than DBM, ML-ELM and H-ELM.

Algorithm	Network Architecture	Testing Accuracy	Training Time
OCR dataset			
DBM [2]	128-2000-2000-26	91.56%	>24h
ML-ELM [6]	128-100-100-15000-26	90.31%(± 0.13)	0.06h
H-ELM [7]	128-200-200-15000-26	90.16%	0.02h
MLMO-ELM	128-200-3000-26	**91.99%(± 0.06)**	1.7h
NORB dataset			
DBM [2]	2048-4000-4000-4000-5	92.77%	>48h
ML-ELM [6]	2048-2000-2000-4000-5	89.54%(± 0.17)	0.02h
H-ELM [7]	2048-3000-3000-15000-5	91.28%	0.05h
MLMO-ELM	2048-3000-4000-5	**92.98%(± 0.26)**	5.78h

Table 1. Experimental results of DBM, ML-ELM, H-ELM and MLMO-ELM on OCR and NORB datasets.

5. CONCLUSIONS

This paper introduces multi objective auto-encoder referred to as MO-ELMAE to learn features of input data. In contrast to ELM-AE, MO-ELMAE has three objective functions and two learn-able weights. MO-ELMAE objective functions are: 1) traditional ELM-AE objective; 2) objective to learn non-linear features; 3) objective to learn label information. MO-ELMAE learns hidden layer parameters of MLMO-ELM fully connected multi layer neural network.

6. ACKNOWLEDGMENTS

The authors would like to acknowledge Delta Electronics Inc.

7. REFERENCES

[1] Ruslan Salakhutdinov and Geoffrey Hinton, "Deep Boltzmann machines," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009, vol. 5, pp. 448–455.

[2] Ruslan Salakhutdinov and Hugo Larochelle, "Efficient learning of deep boltzmann machines.," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010, vol. 9, pp. 693–700.

[3] R. Salakhutdinov and G. Hinton, "An efficient learning procedure for deep boltzmann machines," *Neural Computation*, vol. 24, no. 8, pp. 1967–2006, Aug 2012.

[4] G E Hinton and R R Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[5] D.E. Rumelhart, G.E. Hintont, and R.J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[6] Liyanaarachchi Lekamalage Chamara Kasun, Hongming Zhou, Guang-Bin Huang, and Chi Man Vong, "Representational Learning with Extreme Learning Machines for Big Data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.

[7] 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, 2016.

[8] L. L. C. Kasun, Y. Yang, G. B. Huang, and Z. Zhang, "Dimension reduction with extreme learning machine," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3906–3918, Aug 2016.

[9] Guang-Bin Huang, Lei Chen, and Chee-Kheong 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.

[10] Guang-Bin Huang and Lei Chen, "Convex Incremental Extreme Learning Machine," *Neurocomputing*, vol. 70, pp. 3056–3062, 2007.

[11] Rui Zhang, Yuan Lan, Guang-Bin Huang, and Zong-Ben Xu, "Universal Approximation of Extreme Learning Machine With Adaptive Growth of Hidden Nodes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 2, pp. 365–371, 2012.

[12] Guang-Bin Huang and Lei Chen, "Enhanced Random Search Based Incremental Extreme Learning Machine," *Neurocomputing*, vol. 71, no. 16-18, pp. 3460–3468, 2008.

[13] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[14] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 42, no. 2, pp. 513–529, 2012.

[15] Guang-Bin Huang, "An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 1–15, 2014.

[16] M. Schmidt, " minFunc: unconstrained differentiable multivariate optimization in Matlab http://www.cs.ubc.ca/ schmidtm/Software/minFunc.htm," 2005.

[17] Y. LeCun, Fu Jie Huang, and L. Bottou, "Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2004, vol. 2, pp. 97–104.

[18] B. Taskar, C. Guestrin, and D. Koller, "Max-Margin Markov Networks," in *Advances in Neural Information Processing Systems*, 2004.