Extended Hierarchical Extreme Learning Machine with Multilayer Perceptron

Khanittha Phumrattanaprapin [1] and **Punyaphol Horata**[2], Non-members

ABSTRACT

The Deep Learning approach provides a high performance of classification, especially when invoking image classification problems. However, a shortcomming of the traditional Deep Learning method is the large time scale of training. The hierarchical extreme learning machine (H-ELM) framework was based on the hierarchical learning architecture of multilayer perceptron to address the problem. H-ELM is composed of two parts; the first entails unsupervised multilayer encoding, and the second is the supervised feature classification. H-ELM can give a higher accuracy rate than the traditional ELM. However, there still remains room to enhance its classification performance. This paper therefore proposes a new method termed the extending hierarchical extreme learning machine (EH-ELM), which extends the number of layers in the supervised portion of the H-ELM from a single layer to multiple layers. To evaluate the performance of the EH-ELM, the various classification datasets were studied and compared with the H-ELM and the multilayer ELM, as well as various state-of-the-art such deep architecture methods. The experimental results show that the EH-ELM improved the accuracy rates over most other methods.

Keywords: Hierarchical Extreme Learning Machine, Hierarchical learning, Multilayer Perceptron

1. INTRODUCTION

The Extreme Leaning Machine (ELM) is one of the most popular algorithms for regression and classification. Guang-Bin Huang, et al., 2012 [1] proposed an ELM based on the least squares approach. The ELM is capable of learning from a simple structure called the single layer feedforward neuron networks (SLFNs) [2, 3]. Within the ELM learning process, input weights and biases are randomly generated in hidden layers which calculate and compute the output weights. The ELM offers better performance and faster training time scales than traditional algorithms; such as the back propagation (BP) [4] and the support vector machine (SVM) [5]. Many researchers and application developers have applied ELM in various applications, such as face classification [6], image segmentation [7], human action recognition [8], and power-load forecasting [9, 10].

Although ELM is a fast learning method, many researchers have developed numerous ways to improve the ELM. Kasun, et al., 2003 [11] developed an adapted version of the ELM, referred to as the Multilayer Extreme Learning Machine (ML-ELM), which increases the number of layers in each network. ML-ELM learning is achieved through a building block approach using an ELM-based autoencoder for classification problems. Each layer of the training network is represented as hierarchical learning, stacked layer by layer, suitable for learning within large sized datasets. In ML-ELM training, input weights and biases are orthogonal and randomly generated within hidden parameters. Through this method, the ML-ELM demonstrated superior performance over the traditional ELM within large datasets. The ML-ELM differs from other multilayer schemes, such as the Deep Belief Network (DBN) [12] and the Deep Boltzmann Machine (DBM) [13], because it does not need to fine-tune its parameters. In the decision making process of the ML-ELM, prior to the supervised least mean square optimiaztion, the encoding outputs are directly fed into the last layer, which calculates output weights, without random feature mapping. While the ML-ELM might not use all of the learning advantages of the ELM [1], the universal approximation capability [14] of the ELM requires the feature mapping of inputs in order to calculate the final results.

Meanwhile, the hierarchical learning schemes are designed to be usable with large datasets. Example algorithms include the deep learning (DL) approaches [15,16], in which the deep architectural [17] learning schemes are represented as multiple levels. DL is used for complicated, multi-feature extraction, in order to learn how to automatically produce features in the higher layers from the lower layers. The extraction process is based on an unsupervised learning approach, in which the output of the unsupervised parts provide the inputs of the supervised learniing parts. The DL approach based on the BP algorithm requires multiple fine-tuning of the network parameters. The complicated DL structure affects the learning speed, which is very slow in large sized datasets.

The hierarchical ELM (H-ELM) [18,19] was proposed to address this problem, which combined the ELM with the hierarchical learning approach. The

Manuscript received on October 4, 2016 ; revised on November 24, 2016.

Final manuscript received on February 8, 2017.

[1,2] The authors are with the Department of Computer Science, Faculty of Science, Khon kaen University, 123 Mitraphab Road, Nai-Meuang, Meuang, Khon kaen, 40002 Thailand, E-mail: p.khanittha@kkumail.com and punhor1@kku.ac.th

H-ELM consists of two parts: 1) the unsupervised feature extraction part: the endoding of the input dataset via the sparse autoencoder (based on the ELM autoencoder), and 2) the supervised feature classification part: the single layer used to compute the final output weights, similar to the original ELM. However, the H-ELM may modify the second part to improve classification performance.

The limitations of the supervised classification part of the single layered H-ELM have inspired the use of such deep architecture concepts, such as the two-hidden-layer extreme learning machine (T-ELM),. Qu, et al., 2016 [20], which increases the the single layer of the original ELM to two layers. The performance and testing accuracy of the T-ELM were higher than the those of the batch ELM. Therefore, in order to improve the performance of the H-ELM, we propose herein a modified version of the H-ELM, referred to as the *extended* hierarchical extreme learning machine (EH-ELM), in which the second part of the proposed method (supervised classification training) extends to two or more layers.

The rest of this paper is organized as follow: the next section introduces preliminary related works, consisting of general information and concepts of the ELM. The third section describes the proposed EH-ELM framework, and the fourth section contains the performance evaluation with which to verify the efficiency of the EH-ELM with the various classification datasets and the experimental results of other MLP learning algorithms. Eventually, we draw conclusions in the last section.

2. RELATED WORKS

In this section, we will describe the related works to prepare some essential background knowledge. There are comprised of the conventional ELM [1], H-ELM framework [18] and T-ELM [20].

2.1 Extreme Learning Machine (ELM)

ELM is a learning algorithm, which represented for SLFN with L dimension random feature space. The hidden neurons are randomly generated the parameters (input weights and biases). Let $(\boldsymbol{x_j}, \boldsymbol{t_j})$, $j = 1, \ldots, N$ be a sample of the N distinct samples where $\boldsymbol{x_j}$ is the j-th input sample, $\boldsymbol{t_j}$ is the target vector of $\boldsymbol{x_j}$. The input samples are mapped to L dimension ELM feature spaces, and the output of network is defined as follows

$$f_L(\mathbf{x}) = \sum\nolimits_{j=1}^{L} g(\boldsymbol{w_j}\boldsymbol{x} + b_j) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{1}$$

where $\boldsymbol{w_j}$ denotes the input weight vector between the input layer and the output layer, $b_j$is the bias of jth layer, $\boldsymbol{\beta} = [\boldsymbol{\beta_1}, \ldots, \boldsymbol{\beta_L}]^{\boldsymbol{T}}$ is the output weight matrix, g$(\cdot)$ is the activation function, $\mathbf{x}$ is the input sample and pmbh(x) $= [g_1(\mathbf{x}), \ldots, g_L(\mathbf{x})]^T$ is the vector of the output matrix $\mathbf{H}$ input $\mathbf{x}$.

ELM can resolve the learning problem is as follows

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

where $\mathbf{H}$ is the hidden layer output matrix

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\boldsymbol{x_1}) \\ \vdots \\ \mathbf{h}(\boldsymbol{x_N}) \end{bmatrix} = \begin{bmatrix} h_1(\boldsymbol{x_1}) & \cdots & h_L(\boldsymbol{x_1}) \\ \vdots & \ddots & \vdots \\ h_1(\boldsymbol{x_N}) & \cdots & h_L(\boldsymbol{x_N}) \end{bmatrix}, \tag{3}$$

and $\mathbf{T}$ is a target matrix or the desired matrix as

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{bmatrix}. \tag{4}$$

where m is the dimension of matrix $\mathbf{T}$ for each $1, \ldots, N$ data.

ELM computes the output weights as follows

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{5}$$

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

Learning of ELM is summarized in three steps as the following:

Step 1: Randomly generated parameters consist of input weights $\mathbf{w}_{ij}$ and bias b_j for hidden layer.

Step 2: Calculate the hidden layer output matrix $\mathbf{H}$.

Step 3: Compute the output weights $\boldsymbol{\beta}$.

2.2 Hierarchical Extreme Learning Machine (H-ELM)

Hierarchical Extreme Learning Machine (H-ELM) proposed by Tang et al. [18]. H-ELM is a new multilayer perceptron (MLP) algorithm for training data. The data before learning should be transformed to the ELM random feature space, and the H-ELM algorithm has two parts as shown in Fig.1:

1) The first part is the unsupervised learning to extract features of the input samples as the hierarchical representation. This part consists of N-layers to transform the input samples to sparse high-level features, and the output of each layer is computed as

$$\mathbf{H}_i = g(\mathbf{H}_{i-1} \cdot \boldsymbol{\beta}) \tag{6}$$

where $\mathbf{H}_i$ is the random mapping output of the i-th layer, $\mathbf{H}_{i-1}$ is a previous random mapping output of $\mathbf{H}_i$ layer, g$(\cdot)$ is an activation function, $\boldsymbol{\beta}$ is the hidden layer weights. Moreover, each layer extracts features of the input samples using the ELM sparse autoencoder, which it is defined the objective function as

$$O_{\beta} = \operatorname{argmin}_{\beta} \left\{ \|\mathbf{H}\boldsymbol{\beta} - \mathbf{X}\|^2 + \|\boldsymbol{\beta}\|_{\ell_1} \right\} \tag{7}$$

where $\mathbf{X}$ is the input data, $\mathbf{H}$ is the random mapping output matrix, $\boldsymbol{\beta}$ is the hidden layer weight. To ob-

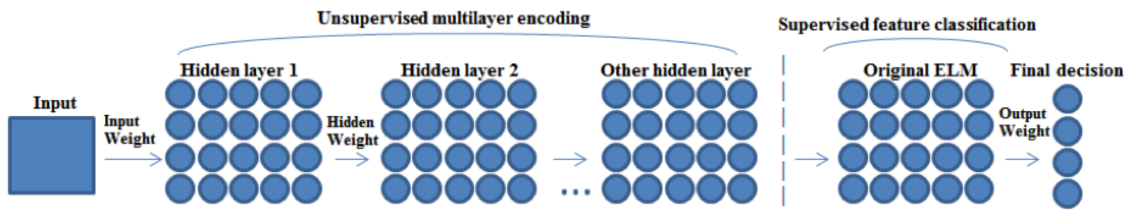


Fig.1: *The overall H-ELM two-part structure [9].*

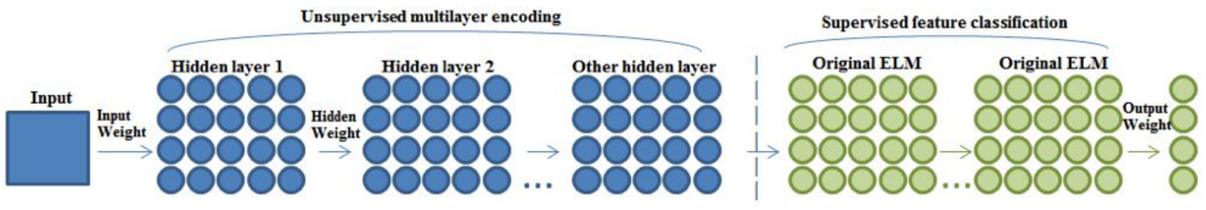


Fig.2: *The two parts of the proposed learning algorithm; the unsupervised multilayer encoding, and the supervised feature classification, respectively.*

tain $\boldsymbol{\beta}$, the ℓ_1 optimization was used in Eg. (8) [18]. Thus, the aim of the ELM sparse autoencoder is to extract features of $\mathbf{X}$ that can be reconstructed $\mathbf{X}$ from the output features without tuned parameters.

2) After the unsupervised part, this part is the supervisor learning part to compute the output weights using the traditional ELM. That is the input weights and biases are randomly generated and then they are used to compute the final output weights.

2.3 Two-hidden-layer extreme learning machine (T-ELM)

The two-hidden-layer feedforward networks (TLFNs) approach was originally proposed by Tamura and Tateishi [21] and Huang [22] to improve the TLFNs framework. T-ELM proposed the ideas of ELM into TLFNs by Qu, et al., 2016 [20], utillizing the ELM algorithm for regression and classification, but having a two-hidden-layer structure. Assume that the first and second layers have the same number of hidden nodes.

The T-ELM Learning process is summarized in the following seven steps:

Step 1: Randomly generate the input weight matrix $\mathbf{w}$ and bias $\mathbf{b}$ of each layer.

Step 2: Calculate $\mathbf{H}=g(\mathbf{w}_{IE}\mathbf{x}_E)$, where $\mathbf{w}_{IE}$ is $[\mathbf{b}\ \mathbf{w}]$ and $\mathbf{x}_E$ is $\boldsymbol{\beta}[1\ \mathbf{x}]^T$.

Step 3: Obtain the hidden weight of the first layer and the output layer, $\boldsymbol{\beta}=\mathbf{H}^{\dagger}\mathbf{T}$.

Step 4: Calculate the output of the first layer, $\mathbf{H}_1=\mathbf{T}\boldsymbol{\beta}^{\dagger}$

Step 5: Assign the hidden weight and bias of the second layer.

Step 6: Obtain the output of the second layer, $\mathbf{H}_2=g(\mathbf{w}_{HE}\mathbf{H}_E)$.

Step 7: Calculate the output weight in the output layer, $\boldsymbol{\beta}=\mathbf{H}_2^{\dagger}\mathbf{T}$.

Note, that while our proposed learning algorithm extends the layers within the supervised learning part, we have not considered the pseudo-inverse of the output weights, written as $\boldsymbol{\beta}^{\dagger}$.

3. PROPOSED LEARNING ALGORITHM

In this section, we outline our proposed framework, which extends the supervised classification part of the H-ELM, referred to as the EH-ELM. Unlike traditional DL frameworks [12, 17] which consist of two training parts, we focused on how to modify the single layer into multilayer supervised classification of the H-ELM. The EH-ELM therefore out performed and proved more efficient than the H-ELM and MLP algorithms.

In the supervised part of the proposed method, the single layer is extended into multiple layers. The output weights of each supervised layer are computed similarly to the traditional ELM, where the approximated target matrix of the previous layer is the input of the next supervised layer. The final output weights are computed from the last supervised layer of the supervised part.

According to the EH-ELM, shown in Fig 2, the input will create and feed samples into the unsupervised part. The resulting high-level features will then be extracted from the input data. The results of the next layer are further input in the multilayer supervised classification part. In the unsupervised part, the input results are computed by M-layers of the ELM based regression, and the subsequent input data is extracted and converted into hierarchical features. The output of each layer in the supervised part is calculated as:

$$\mathbf{H}_j = g(\mathbf{H}_{j-1} \cdot \boldsymbol{\beta}) \quad (8)$$

where $\mathbf{H}_j$ denotes the output of the j-th layer of the supervised part ($j \in [i, N]$), $\mathbf{H}_{j-1}$ is the output of the (j-1)-th layer of the supervised part, g($\cdot$) represents an activation function for each hidden layer, and $\boldsymbol{\beta}$ = $[\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_L]^T$ is the output weight matrix of each layer. However, the parameters of each layer do not need to be fine-tuned, and each layer of the EH-ELM is an independent section.

The learning process of the EH-ELM is summarized as follows:

The unsupervised encoding part:

Input: N training sample $(\boldsymbol{x_k}, \boldsymbol{t_k})$, k=1, …, N of N distinct samples, $\boldsymbol{x_k}$ is the k-th sample, and $\boldsymbol{t_k}$ denotes the target vector of $\boldsymbol{x_k}$.

Step 1: Randomly generate the input weights $\mathbf{w}_i$ of each layer, where i is $(i \in [1, M])$.

Step 2: Insert the α for the input data, $\mathbf{H}_i$= $[x_i \ \alpha]$.

Step 3: Calculate $\mathbf{H}_i$=($\mathbf{H}_{i-1}$·$\mathbf{w}_{i-1}$).

Step 4: $\mathbf{A}_i$= $\mathbf{H}_i$ is the adjusted scale in [-1,1].

Step 5: Calculate the output weights of each layer, $\boldsymbol{\beta}_i$ = autoencoder ($\mathbf{A}_i$, $\mathbf{H}_i$) [18].

Step 6: Calculate the approximated target of each layer, $\mathbf{T}_i$=($\mathbf{H}_i\boldsymbol{\beta}_i$).

Step 7: Adjust $\mathbf{T}_i$ scale in [0,1].

Return to Step 2 until all of a layers of the unsupervised encoding part are completed.

Output: Finish training unsupervised encoding part. The result is **T**.

The supervised classification part:

Input: **T** of the previous part is the input of this part, where S represents a scale factor, and C is a ℓ_2-penalty value.

Step 8: Randomly generate the input weights $\mathbf{w}_j$ of each layer, where j is $(j \in (M, P])$.

Step 9: Insert α for the input data, $\mathbf{H}_{j-1}$= $[\mathbf{T}_{j-1}\alpha]$.

Step 10: Calculate $\mathbf{T}_j$=$g(\mathbf{H}_{j-1} \cdot \mathbf{w}_j)$.

Return to Step 9 until all layers of the supervised classification part are completed.

Step 11: Calculate the expected output weights, β=$\mathbf{H}^{\dagger}\mathbf{T}_j$.

The difference between the proposed EH-ELM and ML-ELM are as follows: 1) ML-ELM utilizes unsupervised learning, using a sample, layer by layer. In contrast, H-ELM is modified into the EH-ELM, which consists of two parts: the unsupervised encoding part and the supervised classification part. 2) The ML-ELM employs orthogonal random hidden parameters, in which to construct the weight matric of each unsupervised layer; which is not necessary in the EH-ELM, due to the various outputs of the random number of the input nodes. 3) ML-ELM uses the ℓ_2-norm ELM autoencoder, whereas the EH-ELM uses an ℓ_1 penalty autoencoder to obtain more sparsely hidden input data.

4. PERFORMANCE EVALUATION

In this section, we verify the performance of proposed EH-ELM in three ways: first, we compared the accuracy rate and the standard deviation values of the EH-ELM and H-ELM, using the various benchmark binary and multiple class classification datasets. Next, we compared the last number of hidden neurons of both the EH-ELM and H-ELM algorithms; and lastly, we compared the accuracy rates of the proposed EH-ELM with various state-of-the-art MLP algorithms. In both the second and third methods, two images classification datasets were used, shown in Tables 1 and 2.

All experiments were conducted on an Intel Core i7 3.50 GHz processor CPU, 8G ram, Windows 7, and Matlab R2014a.

4.1 Benchmark Datasets

In order to extensively verify the performance of the two comparative methods, the proposed EH-ELM and the traditional framework H-ELM, four binary classification cases and eight multiple classification case datasets were selected. Most of the datasets were obtained through the UCIdatabase (https://archive.ics.uci.edu/ml/datasets.html) [23], represented in Table 1

Table 1: *Specification of binary and multiple classes classification datasets.*

Datasets	♯train	♯test	♯ feature	♯classes
Liver	242	104	6	2
Diabetes	538	231	8	2
Leukemia	38	34	7129	2
Phoneme	3800	1622	5	2
Glass	150	65	9	6
Led7digit	350	150	7	10
Vehicle	593	254	18	4
Pendigits	537	231	256	10
Semeion	956	638	256	10
Optdigits	3823	1767	64	10
Thyroid	5040	2160	21	3
Shuttle	43500	14500	9	6

From Table 1, within the 12 classification datasets; the four binary class datasets (Liver, Diabetes, Leukemia [24], and Phoneme) and the multiple class datasets (Glass, Led7digit, Vehicle, Pendigits, Semeion, Optdigits, Thyroid, and Shuttle) can be classified in to six groups of data, as follows:

1) Data sets of relatively small size and low dimensions: Liver, Diabetes, Glass, Led7digit, and Vehicle.
2) Data sets of relatively small size and medium dimensions: Pengigits and Semeion.
3) Data sets of relatively small size and high dimensions: Leukemia.
4) Data sets of relatively medium size and low dimensions: Phoneme and Thyroid.
5) Data sets of relatively medium size and medium dimensions: Optdigits.
6) Data sets of relatively large size and low dimensions: Shuttle.

Table 2: *Specification classification datasets for comparison of the proposed EH-ELM and the state-of-the art MLP algorithms.*

Datasets	♯train	♯test	♯feature	♯classes
MNIST	50000	1000	784	10
NORB	24300	24300	2048	5

Table 2 demonstrates multiple classes within two datasets: MNIST and NORB, detailed as follows:

1) MNIST: the MNIST dataset [25] is the image classification dataset, which contains handwritten digits, containing 60,000 training and 10,000 testing sample composts. The digital images (28 × 28 pixels) have been sized and fixed for normalization. The original imagess are placed into a training algorithm without preprocessing. Each MNIST sample has 784 features. The number of the output layers are then assigned as the number of its like features.
2) NORB: the NORB dataset [26] represents the 3D images classification problem. It contains pictures of 50 toys, separated into five classes: four animals, human figures, airplanes, trucks, and car. We reconstructed the input pixels, as N = 2 x 1024 =2048. In this paper, sizes of NORB samples for training and testing were divided into equal groups of 24,300 units.

4.2 Comparison between the Proposed EH-ELM and H-ELM

1) Selecting the Number of Hidden Neurons: This section is the criteria for selecting the number of hidden node of each layer that can obtain from the empirical study. It described the effects of selecting hidden nodes in unsupervised autoencoder and supervised classification part of EH-ELM for each dataset problem. A case study for the Glass problem will be illustrated in this section to describe how to select the number of hidden properly. Let $L1$, $L2$ be the number of hidden nodes in the first layer and second layer of the unsupervised autoencoder part, and L3, L4 be the number of hidden nodes in the first and the second in the supervised classification part of EH-ELM. $L1$ and $L2$ were set in the range of $[10, 20, \ldots, 100]$, while $L3$ and $L4$ were set in the range of $[100, 200, \ldots, 1500]$. Each network of EH-ELM in this study, we set $L1$ is equal to $L2$, and also $L3$ is equal $L4$.

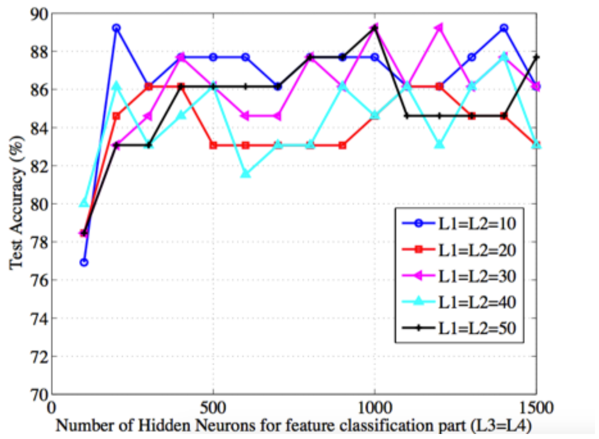


Fig.3: *The Testing Accuracy of EH-ELM in terms of a Number of Hidden Neurons L3 and L4 Feature Classification Part.*

The Fig. 3 shows the testing accuracy rate when EH-ELMs were trained by a various number of hidden nodes in each layer in range of 10 to 100. Each line is illustrated as the testing accuracy rate when $L1$ and $L2$ were fixed. $L3$ and $L4$ are various within the setting condition, $L1 = L2$, and $L3 = L4$. The graph in this figure is shown that the neural network of EH-ELM giving the highest testing accuracy rate for the Glass dataset is the network building the structure as $L1 - L2 - L3 - L4$ to be set as 10-10-200-200, respectively.

2) Performance of Learning Accuracy: We are set the number of hidden neurons in the unsupervised autoencoder and supervised classification parts of the EH-ELM equivalent to the H-ELM for fair comparison. In the EH-ELM, two-layer feature encoding is utilzed as the input for two-layer feature classification. Within the H-ELM, two-layer feature encoding is then utilzed as the input in one-layer feature classification. Performance verification of each method was made through an extensive number of datasets in both the binary class and multiple class datasets.

The training and testing experiments were repeated fifty times for validity, and the average values of the accuracy rates and standard deviations are reported in Table 3.

Table 3 illustrates the EH-ELM classification of four binary datasets (Liver, Diabetes, Leukimia, and Phoneme) and their respective accuracy rates against the H-ELM (86.22, 83.85, 84.76, and 91.92, respectively). Within the Leukemia case, the EH-ELM im-

Table 3: *Performance comparisons of the proposed EH-ELM and H-ELM with binary and multiple class benchmarks.*

Datasets	EH-ELM			H-ELM		
	Number of Hidden neurons	Accuracy (%)	Standard Deviation	Standard Deviation	Accuracy (%)	Number of Hidden neurons
Binary Classes Datasets						
Liver	20,20,200,200	**86.22**	**2.45**	3.05	85.19	20,20,200
Diabetes	50,50,400,400	**83.85**	1.73	1.61	82.96	50,50,400
Leukemia	200,200,5000,5000	**84.76**	**9.10**	10.05	82.41	200,200,5000
Phoneme	100,100,5000,5000	**91.92**	**0.36**	0.40	90.66	100,100,5000
Multiple Classes Datasets						
Glass	10,10,200,200	**84.86**	**2.56**	2.66	81.16	10,10,200
Led7digit	50,50,500,500	**75.10**	0.80	0.82	74.66	50,50,500
Vehicle	50,50,300,300	**82.54**	**1.63**	1.93	81.99	50,50,300
Pendigits	50,50,300,300	97.18	0.76	0.74	97.14	50,50,300
Semeion	300,300,500,500	**97.49**	**0.32**	0.44	97.41	300,300,500
Optdigits	50,50,490,490	97.77	0.19	0.19	97.82	50,50,490
Thyroid	200,200,5000,5000	96.00	0.17	0.17	95.93	200,200,5000
Shuttle	100,100,2000,2000	99.47	0.08	0.06	99.47	100,100,2000

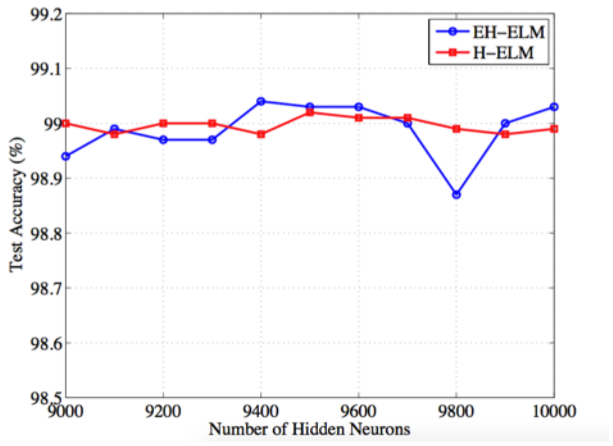
proved accuracy up to 2.3 percent of the H-ELM. The EH-ELM and H-ELM maintained comparable stanrdard derivation accuracy rates.

In the multiple class datasets (Glass, Led7digit, Vehicle, Pendigits, Semeion, Optdigits, Thyroid, and Shuttle), the EH-ELM produced accuracy rates of 84.86, 75.10, 82.54, 97.18, 97.49, 97.77, 96.00, and 99.47, respectively. Remarkably, the EH-ELM accuracy rates out performed the H-ELM in four of eight datasets (Glass, Led7digit, Vehicle, and Semeion cases. Accuracy rates and stand deviations were comparable in the Pendigets, Optdigit, Thyroid, and Shuttle datasets. Summarily, most datasets produced improved accuracy rates using the proposed H-ELM.

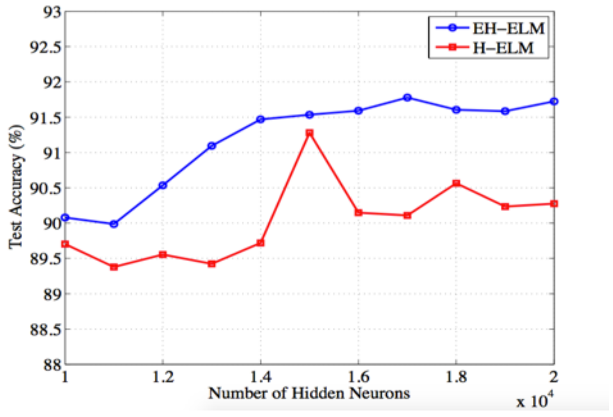
4.3 Comparison of the last number of hidden neurons between the EH-ELM and H-ELM

Comparisons were made between the proposed EH-ELM and H-ELM when the number of hidden neurons in the last supervised layer varied. The number of hidden neurons in the unsupervised autoencoder for both methods were set as equal. In the supervised feature classification of the EH-ELM, comparisons were made prior to the equal setting of the hidden neurons of the H-ELM. The results of the classification performance for testing samples are shown in Fig 4.

In the MNIST dataset, the fluctuation of the testing accuracy rates of the H-ELM were less than those of the EH-ELM, shown in Fig 4(a). The number of hidden neurons varied in range, from 9,000 to 10,000 nodes. The EH-ELM yielded the lowest accuracy rates at 9,800 hidden neuron nodes, while the highest accuracy rate was found at 9,400 hidden neuron



(a)



(b)

Fig.4: *Comparison of testing rates between EH-ELM and H-ELM: (a) MNIST (b) NORB.*

Table 4: *Learning Accuracy performance comparisons in the MNIST Dataset.*

Algorithm	Accuracy (%)	Training time (s)
SAE [27]	98.60	>17 hours
SDA [28]	98.72	>17 hours
DBN [12]	98.87	66849
MLP-BP [4]	97.39	214268.10
ML-ELM [11]	99.04	2316.19
H-ELM [18]: two unsupervised layers	98.92	281.37
H-ELM [18]: three unsupervised layers	97.81	4120.79
EH-ELM: two unsupervised layers, two supervised layers	**99.05**	1632.45
EH-ELM: two layers of unsupervised, three layer of supervised.	98.92	3552.62

Table 5: *Learning Accuracy performance comparisons in the NORB Dataset.*

Algorithm	Accuracy (%)	Training time (s)
SAE [27]	86.28	> 2 days
SDA [28]	87.62	> 2 days
DBN [12]	88.62	327960
MLP-BP [4]	84.20	34005.47
ML-ELM [11]	88.91	2344.83
H-ELM [18]: two unsupervised layers	91.28	881.43
H-ELM [18]: three unsupervised layers	90.28	8279.67
EH-ELM: two unsupervised layers, two supervised layers	**91.78**	1341.67
EH-ELM: two layers of unsupervised, three layer of supervised.	90.72	6585.57

nodes. In all cases, the accuracy rates of the EH-ELM were higher than those of the H-ELM.

In the NORB dataset, Fig 4(b), the accuracy of the EH-ELM was greater than the H-ELM in all selected number of hidden neurons, ranging from 10,000 to 20,000. Within this range, the best accuracy rate of the H-ELM at 15,000 nodes was less than the best accuracy rate of EH-ELM at 17,000 nodes. The accuracy rates of EH-ELM were higher than those of the H-ELM in every quantity of hidden nodes.

4.4 Comparision with State-of-the-Art MLP Algorithms

In this section, we apply two image classification datasets for comparison of the EH-ELM with the comparative multilayer algorithms; such as the Stacked AutoEncoder (SAE) [27], Staked Denoising Autoencoder (SDA) [28], Deep Belief Networks (DBN) [12], MLP-BP [4], ML-ELM [11], and H-ELM [18]. Within the backpropagation algorithm based MLP training algorithm (SAE, SDA, DBN, and MLP-BP), the learning rate was set at 0.1, and the decay rate at 0.95 for each learning epoch. In the case of SDA, the input corruption rate was 0.5, with a dropout rate of 0.2. The ℓ_2 penalty parameters in the ML-ELM, there were set at 10^{-1}, 10^3, and 10^8; for three layers, respectively. Within the H-ELM algorithm, the number of layers of the unsupervised encoding part was set at two and three layers. Within the proposed EH-ELM, the number of supervised layers was specified at two or three layers, in order to compare the learning performances when changing the number of supervised layers

As shown in Table 4, the testing accuracy of the proposed EH-ELM on MNIST achieves 99.05 %; similar to the ML-ELM, but with faster training times. The EH-ELM proved superior to the SAE, SDA, DBN, and MLP-BP in term of accuracy rates and the training times. Furthermore, the similar architecture of the EH-ELM's supervised part yielded accuracy rates of 99.05 and 98.90 in two and three supervised layers, which were superior to those of the H-ELM, at 98.90 and 97.81, respectively. Table 5 shows that the EH-ELM outperforms than basic H-ELM and other MLP algorithms, including the SAE, SDA, DBN, and ML-ELM. The classification performance test of the EH-ELM having two supervised layers obtained up to 91.78% accuracy, which was greater than the best accuracy rate of the H-ELM, at 91.28 [18]. Although the unsupervised layer of the H-ELM was increased to three layers, its classification rate performance of 90.20 was still less than that of the best accuracy rate of the EH-ELM.

From Tables 4 and 5, the experiment's results demonstrated that the proposed EH-ELM framework (based on H-ELM approach, having two supervised layers) significantly improved classification rates over the original H-ELM, having two or three unsupervised parts, as well as the other multi-layer algorithms in the MNIST and NORB datasets (due to the overfitting of input data). Furthermore, the training times of the EH-ELM in both the MNIST and NORB datasets (1,632.45, and 1,341.67 seconds, respectively) were slower than that of the H-ELM having two unsupervised layers (due to the greater number of supervised layers of the proposed algorithm than that of the supervised layers of the H-ELM.

5. CONCLUSIONS

In this paper, we propose a new method of high performance classification by extending the supervised classification part of the H-ELM, from one supervised layer to multiple supervised layers, referred to as the Extended H-ELM (EH-ELM). Within our experimental setting, we divided the datasets into two categories: twelve datasets consisting of four binary classes, and eight datasets consisting of multiple classes. Datasets also varied in size and dimensions. The results found that within our experimental datasets, the EH-ELM yielded greater average accuracy rates than those of the traditional H-ELM in eight of twelve datasets; with significant statistical difference ($p<0.05$) four of all eight problems consist of Phoneme, Glass, Led7digit and Thyroid datasets. Additional tests were conducted on MNIST and NORB datasets, which contained a varied number of hidden neurons in the last layer of the supervised classification parts. Within the MNIST data sets, the hidden neurons of the EH-ELM and H-ELM totaled 9400 and 9600 nodes, respectively. While both results were sufficiently high, the accuracy rates of EH-ELM were slightly superior to the H-ELM, in which the number of hidden neurons varied. Within the NORB dataset, the accuracy rates of the EH-ELM were higher than those of H-ELM, in every variable of hidden nodes, 17,000 total; of which the EH-ELM produced the best classification accuracy for this problem. The classification accuracy rates of the EH-ELM also proved superior to the state-of-the-art methods (SAE, SDA, DBN, MLP-BP, ML-ELM, and H-ELM) in both the MNIST and NORB datasets. Therefore, the EH-ELM was conclusively deemed the best classification method among all the comparative algorithms. Note that in the MNIST dataset, the accuracy of EH-ELM and H-ELM yielded 99.05% and 98.92% accuracy, with training times of 281.37 and 1632.45 seconds, respectively. In the NORB dataset, the classification accuracy of EH-ELM was 91.78% with a training time of 1341.67 seconds, while the original H-ELM yielded 90.20% with a training time of 881.43 seconds. Additionally, in comparing the effects of the various number of layers, the experimental results indicate that the EH-ELM with two-supervised layers out performed the H-ELM having three or four layers, due to the overfiting of the input data. Regarding training times, the EH-ELM proved slightly slower than the learning time scale of the H-ELM, due to its greater number of supervised layers.

ACKNOWLEDGEMENT

This research was partially supported by the Department of Computer Science, Faculty of Science, Khon Kaen University, Thailand.

References

[1] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme learning machine for regression and multiclass classification,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513-529, Apr. 2012.

[2] K. Hornik, M. Stinchcombe, H. White, “Multilayer feedforward networks are universal approximators,” *Neural Netw.*, vol. 2, no. 5, pp.359-366, 1989.

[3] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Netw.*, vol. 4, no. 2, pp.251-257, 1991.

[4] C. Bishop, *Pattern Recognition and Machine Learning*, New York, NY, USA: Springer-Verlag, 2006.

[5] Suykens, J. A., and Vandewalle, J., “Least squares support vector machine classifiers,” *Neural processing letters*, vol.9, no. 3,pp. 293-300, 1999.

[6] A. A. Mohammed, R. Minhas, Q. M. J. Wu, and M. A. Sid-Ahmed,“Human face recognition based on multidimensional PCA and extreme learning machine,” *Pattern Recognit.*, vol. 44, nos. 10-11, pp. 2588-2597, 2011.

[7] C. Pan, D. S. Park, Y. Yang, and H. M. Yoo, “Leukocyte image segmentation by visual attention and extreme learning machine,” *Neural Comput. Appl.*, vol. 21, no. 6, pp. 1217-1227, 2012.

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

[9] S. Cheng, J. Yan, D. Zhao, et al., “Short-term load forecasting method based on ensemble improved extreme learning machine,” *J. Xi’an Jiaotong University.* 2:029, 2009.

[10] L. Mao, Y. Wang, X. Liu, et al., “Short-term power load forecasting method based on improved extreme learning machine,” *Power Syst. Prot. Control*, vol. 40, no. 20, pp.140-144, 2012.

[11] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, “Representational learning with extreme learning machine for big data,” *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 31-34, Nov. 2013.

[12] G. Hinton, S. Osindero, and Y. Teh, “A fast learning algorithm for deep belief nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527-1554, Jul. 2006.

[13] R. Salakhutdinov and G. Hinton, “Deep Boltzmann machines,” in *Proc. 12th Int. Conf. Artif.*

Intell. Statist., Clearwater Beach, FL, USA, pp. 448-455, Jul. 2009.

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

[15] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1-127, 2009.

[16] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798-1828, Aug. 2013.

[17] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1-127, 2009.

[18] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang, "Extreme Learning Machine for Multilayer Perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 5, no. 4, pp. 809-821, 2015.

[19] H.-G. Han, L.-D. Wang, J.-F. Qiao, "Hierarchical extreme learning machine for feedforward neural network," *Neurocomputing*, pp. 128-135. 2014.

[20] Qu, B. Y., et al. "Two-hidden-layer extreme learning machine for regression and classification," *Neurocomputing*, vol. 175, pp. 826-834, Jan. 2016.

[21] S. I. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layers versus three," *IEEE Transactions on Neural Networks*, vol.8, no. 2, pp.251-255, 1997.

[22] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Transactions on Neural Networks*, vol.14 no.2, pp.274-281, 2003.

[23] C. L. Blake and C. J. Merz, "UCI Repository of Machine Learning Databases," Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, 1998. [Online]. Available:http://www.ics.uci.edu/ mlearn/MLRepository.html

[24] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *science*, pp.531-537,1999.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[26] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on.*, Vol. 2, 2004.

[27] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504-507,2006.

[28] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 1096-1103.

Khanittha Phumrattanaprapin received the B.Sc. degree in Computer Science from Khon Kaen University, Khon Kaen, Thailand, in 2014, where she currently pursuing the M.S. degree with the Department of Computer Science. She current research interests include machine learning, hierarchical learning, deep neural networks and big data analytics.

Punyaphol Horata received the B.Sc. degree in Mathematics from Khon Kaen University, M.S. degree in Computer Science from Chulalongkorn University, and graduated Ph.D. in Computer at Khon Kaen University, Thailand. He is currently an assistant professor at computer science department, faculty of science, Khon kaen University, Thailand. His research interests include machine learning, soft computing, software engineering and pattern recognition.