7-2015

State Preserving Extreme Learning Machine for Face Recognition

Md. Zahangir Alom
University of Dayton

Paheding Sidike
University of Dayton

Vijayan K. Asari
University of Dayton, vasari1@udayton.edu

Tarek M. Taha
University of Dayton, ttaha1@udayton.edu

State Preserving Extreme Learning Machine for Face Recognition

Md. Zahangir Alom, Paheding Sidike, Vijayan K. Asari, Tarek M. Taha
Dept. of Electrical and Computer Engineering
University of Dayton, Dayton, OH 45469, USA
{alomm1, pahedings1, vasari1, tarek.taha}@udayton.edu

***Abstract*—Extreme Learning Machine (ELM) has been introduced as a new algorithm for training single hidden layer feed-forward neural networks (SLFNs) instead of the classical gradient-based algorithms. Based on the consistency property of data, which enforce similar samples to share similar properties, ELM is a biologically inspired learning algorithm with SLFNs that learns much faster with good generalization and performs well in classification applications. However, the random generation of the weight matrix in current ELM based techniques leads to the possibility of unstable outputs in the learning and testing phases. Therefore, we present a novel approach for computing the weight matrix in ELM which forms a State Preserving Extreme Leaning Machine (SPELM). The SPELM stabilizes ELM training and testing outputs while monotonically increases its accuracy by preserving state variables. Furthermore, three popular feature extraction techniques, namely Gabor, Pyramid Histogram of Oriented Gradients (PHOG) and Local Binary Pattern (LBP) are incorporated with the SPELM for performance evaluation. Experimental results show that our proposed algorithm yields the best performance on the widely used face datasets such as Yale, CMU and ORL compared to state-of-the-art ELM based classifiers.**

Keywords—Extreme learning machine; weight adaptive; neural network; feature extraction; face recognition

I. INTRODUCTION

Extreme Learning Machine (ELM) has attracted more and more attention of the community in the field of machine learning due to its higher regularization performance at a much faster speed [1-2]. The basic principle of ELM can be described as: when the input weight and bias are randomly allocated, the output weights are computed by the generalized inverse of the hidden layer outputs matrix(H). ELM can be viewed as a single hidden layer feed-forward neural network (SLFN) with L hidden neurons that can learn L distinct samples with zero error. Even if the number of hidden neurons is less than the number of distinct samples, ELM still can assign random parameters to the hidden nodes and calculate the output weights using the pseudo-inverse of H giving only a small error $\epsilon > 0$. The hidden node parameters, i.e., input weights and biases or centers and impact factors, do not need to be tuned during training and may simply be assigned with random values [1-5]. Many studies have been conducted in the field of ELM from both theoretical and application aspects. Huang *et al.* introduced an incremental constructive method to universally approximate the parameters in ELM where the number of hidden neurons have been generated randomly to SLFNs one by one or group by group [5]. ELM has several advantages, such as ease of use, faster learning speed, higher generalization performance, and being suitable for many nonlinear activation functions as well as kernel functions. It has also been shown that ELM yields much better generalization performance with much faster learning speed and less human interventions than other conventional methods.

From our points of view, there are two aspects that influence the robustness properties in ELM neural networks: 1) the computational robustness related to numerical stability, and 2) outliers robustness. The first aspect is generally ignored, since many efforts emphasize on the accuracy of applications [6]. Those computational problems occur when the hidden layer output matrix is ill-conditioned – typically caused by the random input weights and biases selection. This makes the linear system, used to train the output weights, result in a solution sensitive to data perturbation and become a poor estimation to the truth [6]. Additionally, it is known that the size of the output layer weight is more related for the generalization competency than the configuration of the neural network, in terms of number of neurons and format of activation function [7], [8]. Several studies [9-11] explore this issue.

The second aspect, related to outlier robustness, has been discovered in recent years in a few articles, using estimation methods that are known for being less sensitive to outliers then the Ordinary Least Squares (OLS). Studies such as Huynh and Wong [12] substitute the singular value decomposition method by the weighted least squares, which is similar to OLS, but creates penalties corresponding to training patterns to weight their contribution to the final solution. Barros *et al.* [13] concentrate their efforts on robust classification problems with a proposal of an ELM that used Iteratively Reweighted Least Squares (IRLS), named ROB-ELM. Horata *et al* [14] addresses both aspects by applying three estimation methods: IRLS, the Multivariate Least-Trimmed Squares (MLTS) estimator and the One-Step Reweighted MLTS (RMLTS) modified by Extended Complete Orthogonal Decomposition (ECOD), which acts over the computational problem.

In this paper, we consider both aspects to achieve the improved performance of ELM. Based on the regularized extreme learning machine (RELM) [4][9], which on the concept of similar samples should share similar properties, we propose a State Preserving Extreme Learning Machine (SPELM). This is achieved by preserving and updating state variables that are instrumental to system accuracy. The experimental results demonstrate that the SPELM can achieve much better performance in comparison with conventional ELM and RELM. To evaluate the performance of the approach, we test the SPELM on three popular face recognition databases, namely Yale, CMU-AMP, and ORL.

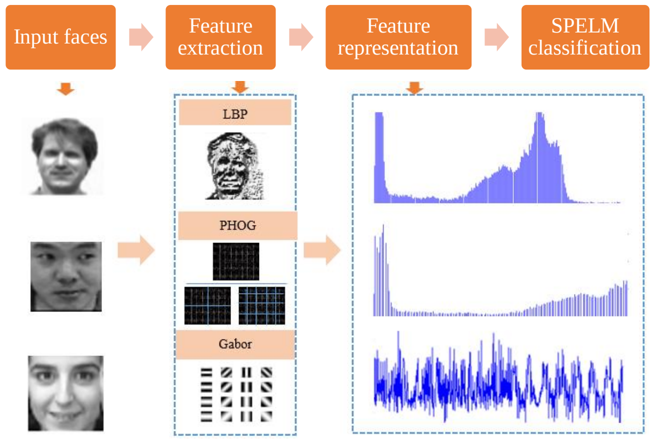


Fig. 1. Proposed implementation scheme.

To show the effectiveness of the SPELM, we further evaluate its performance by incorporating local appearance descriptors, such as Gabor wavelets [15-20], local binary patterns (LBP) [21-23], and pyramid histogram of orientated gradients (PHOG) features [24], into SPELM for face recognition. LBP feature is an efficient texture descriptor that extracts fine details of facial appearance and texture. In contrast, Gabor feature captures facial shape and appearance information over a range of coarser scales [25]. The PHOG feature is computed by creating a pyramid histograms over the entire image and appending the histograms for each level of the pyramid into a single vector. All of these three features are rich in information content and computational efficiency. Thus, in this paper, we integrate these three feature extraction techniques with the SPELM for evaluation. Test results show that feature based SPELM yields a better face recognition accuracy. Figure 1 depicts the overall test scheme of the proposed algorithm. Our main contributions in this work is summarized as follows:

- A new approach of controlling state weights of RELM which leads to the proposed SPELM for fixed number of hidden neurons generated automatically.
- Evaluation of the performance of the SPELM on face recognition by extracting facial features using three prominent feature extraction methods, namely Gabor, LBP, and PHOG.
- A comparison of the performance of SPELM with ELM and RELM.

II. Theoretical Analysis

In this section, we first review conventional and regularized ELM algorithms, and then introduce the proposed SPELM.

A. *Extreme Learning Machine*

ELM typically applies random computational nodes in the hidden layer and increases learning speed by means of randomly generated weights and biases for hidden nodes rather than iteratively adjusting network parameters, which is commonly adopted by gradient based methods. Different from traditional learning algorithms, ELM tends to reach not only the smallest training error but also the smallest norm of output weights [1-5][9].

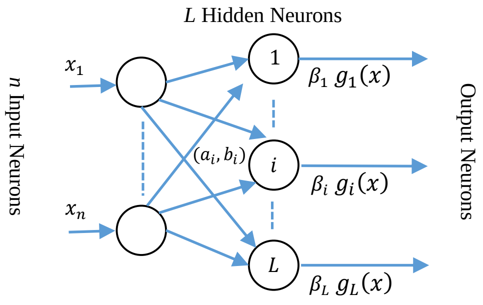


Fig. 2. A typical architecture of the ELM.

A typical architecture of ELM is shown in Fig. 2. The output function of ELM with L hidden nodes for generalized SLFNs is expressed as in [1]

$$f_L(x) = \sum_{i=1}^{L} \beta_i\ g_i(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x), x \in R^d, \beta_i \in R^m \qquad (1)$$

where $a_i = [a_{i1}, a_{i2}, \ldots \ldots, a_{in}]^T$ is the weight vector connecting the input nodes to the ith hidden node, b_i is the ith bias of the hidden node , g_i denotes the output function, i.e., activation function $G(a_i, b_i, x)$ of the ith hidden node, and $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots \ldots, \beta_{im}]^T$ is the weight vector linking the ith hidden node to the output nodes. For N arbitrary distinct samples $(x_j, t_j) \in R^d \times R^m$ the SLFNs with L hidden nodes can approximate these N samples with zero error, meaning $\sum_{j=1}^{L} \|f_j - t_j\| = 0$. Hence, there exists (a_i, b_i) and β_i such that

$$\sum_{i=1}^{L} \beta_i G(a_i, b_i, x_j) = t_j.\ j = 1,2, \ldots, N \qquad (2)$$

The above equations can be rewritten compactly as

$$H\beta = T \qquad (3)$$

where,

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_n) \end{bmatrix} = \begin{bmatrix} G(a_1, b_1, x_1) & \ldots & G(a_L, b_L, x_1,) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_N) & \ldots & G(a_L, b_L, x_N) \end{bmatrix}_{N\times L} \qquad (4)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L\times m}, \qquad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_L^T \end{bmatrix}_{N\times m}. \qquad (5)$$

H is the hidden layer output matrix of the SLFN, and the ith column of H is the ith hidden node output with respect to inputs $x_1, x_2, \cdots\cdots, x_N$, while the jth row, i.e., $h(x_j)$, is the hidden layer feature mapping corresponding to the jth input x_j. As the hidden node parameters (a_i, b_i) can be randomly generated and remain unchanged, the only unknown parameters in ELM are the output weight vectors β_i between the hidden layer and the output layer, which can be simply resolved by ordinary least-square error analysis. Since ELM aims to minimize the training error $\|H\beta - T\|$ and the norm of weights $\|\beta\|$, the smallest norm least-squares solution of the above linear system is

$$\hat{\beta} = H^{\dagger}T, \tag{6}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H [1]. Hence, the prediction value matrix Y is expressed by

$$Y = H\hat{\beta} = HH^{\dagger}T. \tag{7}$$

The error matrix can be described as

$$e = \|Y - T\|^2 = \left\|HH^{\dagger}T - T\right\|^2. \tag{8}$$

In order to increase the stability and generalization ability of the traditional EML, Huang *et al*. introduced the equality constrained optimization-based ELM [4]. According to the solution of the regularized ELM, the weight vector $\hat{\beta}$ can be represented as:

$$\hat{\beta} = \left(HH^T + \frac{I}{C}\right)^{-1} HT^T, \tag{9}$$

where C is a constant and I is an identity matrix. If $\lambda = 1/C$, Eq. (9) can be rewritten as

$$\hat{\beta} = (HH^T + \lambda I)^{-1} HT^T, \tag{10}$$

The solution of Eq. (10) can be obtained by solving the following optimization problem:

$$\min_{\beta} \|\beta^T H - T\|_2^2 + \lambda\|\beta\|_2^2, \tag{11}$$

where $\|\beta\|_2^2 = \sum_{j=1}^{K} \left\|\beta_j\right\|_2^2$ is a regularization factor and $\left\|\beta_j\right\|_2^2$ denotes the $\ell_2 - norm$ of the vector β_j. Furthermore, λ indicates the regularization parameter to balance the influence of the error term and the model complexity. As a result, a simple learning method for SLFNs is called extreme learning machine that may be summarized as in Algorithm I [1].

```
Algorithm I. Conventional Extreme Learning Machine

Inputs: Training set $\aleph$ where
    $\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \dots\dots, N\}$ ,
Activation function $g(x)$, and number of hidden nodes $\widetilde{N}$;
Output:
Step 1: Input weight $a_i$ and bias $b_i$ are initialized randomly,
$i = 1, \dots\dots\dots \widetilde{N}$,
Step 2: Hidden layer outputs matrix $H$ is calculated.
Step 3: Output weight matrix $\beta$ is computed as follows:
    $\beta = H^{\dagger}T$,
where $T = [t_1, \dots\dots t_N]^T$.
```

B. State Perserving ELM (SPELM)

In ELM and RELM, there are three key steps to process: firstly, weight and bias are computed randomly in each learning step; secondly, the input sequences of testing samples are generated randomly for each iteration in case of batch learning; thirdly, the input samples are shuffled according to the output sequences of each iteration. In contrast, in the SPELM, training samples are randomly selected with corresponding labels and state variables such as weight, bias, test sample sequences, and test accuracy are preserved for each iteration. Then the highest accuracy with relevant parameters are stored until the following iteration to provide a better accuracy. The same procedure will be continued until the end of the iteration. The following section explains the details of the SPELM.

In SPELM, the state variables are the number of iterations $\mathcal{K}$, the state of the network $\mathcal{S}_i$ where $i = 1 \dots\dots \mathcal{K}$, the accuracy of the state represented by $\mathcal{T}_{\mathcal{S}_i}$, the number of hidden nodes $\mathcal{H}_n$ of state $\mathcal{S}_i$ where$(\mathcal{H}_n)_{\mathcal{S}_i} \in \mathbb{Z}^+$, and the activation function $G(w_{\mathcal{S}_i}, b_{\mathcal{S}_i}, x)$. The number of hidden nodes $(\mathcal{H}_n)_{\mathcal{S}_i}$for the state $\mathcal{S}_i$ is calculated based on the dimension of input features (d) represents as

$$(\mathcal{H}_n)_{\mathcal{S}_i} = \psi * d \tag{12}$$

where ψ is a constant. Empirically we set $\psi = 10$. The output function of SPELM with $(\mathcal{H}_n)_{\mathcal{S}_i}$ hidden nodes for generalized SLFNs is expressed as:

$$f_{(\mathcal{H}_n)_{\mathcal{S}_i}}(x) = \sum_{i=1}^{\mathcal{H}_n} \beta_{\mathcal{S}_i}\, g_i(x)$$
$$= \sum_{i=1}^{\mathcal{H}_n} \beta_{\mathcal{S}_i}\, G(w_{\mathcal{S}_i}, b_{\mathcal{S}_i}, x). \tag{13}$$

where $x \in R^d$, $\beta_{\mathcal{S}_i} \in R^m$, $w_{\mathcal{S}_i}$ is the weight vector connecting the input nodes to the ith hidden node, $b_{\mathcal{S}_i}$ is the ith bias , and g_i denotes the output function. Hence the activation function $G(w_{\mathcal{S}_i}, b_{\mathcal{S}_i}, x)$ is for the ith hidden node of input x in state $\mathcal{S}_i$. The weight matrix $w_{\mathcal{S}_i}$ and the bias $b_{\mathcal{S}_i}$ in the state of $\mathcal{S}_i$ are updated with respect to the present accuracy $(\mathcal{T}_{\mathcal{S}_i})$ and the immediate previous accuracy $(\mathcal{T}_{\mathcal{S}_{i-1}})$. These terms are defined by the Eq. (14) and Eq. (15), respectively.

$$w_{\mathcal{S}_i} = \begin{cases} w_{\mathcal{S}_i}, & \mathcal{T}_{\mathcal{S}_i} > \mathcal{T}_{\mathcal{S}_{i-1}} \\ w_{\mathcal{S}_{i-1}}, & otherwise \end{cases} \tag{14}$$

and

$$b_{\mathcal{L}_i} = \begin{cases} b_{\mathcal{S}_i}, & \mathcal{T}_{\mathcal{S}_i} > \mathcal{T}_{\mathcal{S}_{i-1}} \\ b_{\mathcal{S}_{i-1}}, & otherwise \end{cases} \tag{15}$$

For N arbitrary distinct samples $(x_j, t_j) \in R^d \times R^m$ SLFNs with $\mathcal{H}_n$ hidden nodes can approximate these N samples with zero error. Hence $\sum_{j=1}^{\mathcal{H}_n} \left\| f_{\mathcal{H}_j} - t_j \right\| = 0$, and there exists $(\mathscr{w}_{\mathcal{S}_i}, \mathscr{b}_{\mathcal{S}_i})$ and $\beta_{\mathcal{S}_i}$ such that

$$\sum_{i=1}^{\mathcal{H}_n} \beta_{\mathcal{S}_i}\ G(\mathscr{w}_{\mathcal{L}_i}, \mathscr{b}_{\mathcal{L}_i}, x_j) = t_j;\ j = 1,2, \dots, N \tag{16}$$

Both equation written above can be expressed as

$$\hat{\beta}_{\mathcal{S}_i} = \left(H_{\mathcal{S}_i} H_{\mathcal{S}_i}{}^T + \lambda I\right)^{-1} H_{\mathcal{S}_i} T^T, \tag{17}$$

where C is a constant and I is an identity matrix. If $\lambda = 1/C$, the solution of the Eq. (17) can be obtained by solving the following optimization problem:

$$\min_{\beta} \left\| \beta_{\mathcal{S}_i}{}^T H_{\mathcal{S}_i} - T \right\|_2^2 + \lambda \left\| \beta_{\mathcal{S}_i} \right\|_2^2 \tag{18}$$

$\beta_{\mathcal{S}_i} = \left\| \beta_{\mathcal{S}_i} \right\|_2^2 = \sum_{j=1}^{K} \left\| \beta_{\mathcal{S}_j} \right\|_2^2$ is considered as the $\ell_2 - norm$ of the vector $\beta_{\mathcal{S}_j}$ mentioned in Eq. (18) and λ is the regularization parameter. In order to update the state accuracy $\mathcal{T}_{\mathcal{S}_i}$ on test examples, the prediction value matrix $Y_{\mathcal{S}_i}$ is expressed by

$$Y_{\mathcal{S}_i} = H_{\mathcal{S}_i} \hat{\beta}_{\mathcal{S}_i} \tag{19}$$

The error can be described as

$$\xi_{\mathcal{S}_i} = \left\| Y_{\mathcal{S}_i} - T \right\|^2 \tag{20}$$

Finally, the test accuracy of state $\mathcal{T}_{\mathcal{S}_i}$ updates based on $\xi_{\mathcal{S}_i}$ as follow

$$\mathcal{T}_{\mathcal{S}_i} = (1 - \xi_{\mathcal{S}_i}) * 100 \tag{21}$$

The implementation of the above mentioned SPELM algorithm can be expressed as in Algorithm II:

III. Experimental results

This section presents the experimental results of our proposed SPELM model for face recognition. The activation function of the hidden layer is set to a 'sigmoid' function and the number of hidden nodes is fixed to $10 \times numDim$ for all ELM, RELM and SPELM. We evaluate the performance of SPELM on face recognition from two aspects: (1) compare the SPELM model with the conventional ELM and RELM; (2) compare their performance by incorporating feature extraction techniques for face recognition.

Dataset: To evaluate the efficiency of the SPELM, we perform unconstrained face verification experiments on the Yale [26], CMU-AMP [27] and ORL [28] face recognition databases. The statistics of these datasets used in this experiment are summarized in Table 1. Fig. 3 shows sample images from these three datasets, in which one subject is randomly selected from each database and each one has 10 samples. As seen in Fig. 3, face images in these three databases contain various poses, illumination, and expressions.

```
Algorithm II. State Preserving ELM
Inputs: Training set $\aleph$, where
    $\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \dots\dots, N\}$,
$g_i(x)$, state $\mathcal{S}_i$ ( $i = 1 \dots\dots \mathcal{K}$ ), $\mathcal{T}_{\mathcal{S}_i}$, $(\mathcal{H}_n)_{\mathcal{S}_i}$ generated
according to the Eq. 12;
Output:
Step 1: while ($i \le \mathcal{K}$) { Start: $if\ (\mathcal{S}_i < 2)$ {
        Random initialization of input weight $\mathscr{w}_{\mathcal{S}_i}$ and
        bias $\mathscr{b}_{\mathcal{S}_i}$ for first state.}
        else { $if\ (\mathcal{T}_{\mathcal{S}_{i-1}} \ge \mathcal{T}_{\mathcal{S}_{i-2}})$
        {Update weight and bias according to the Eq. (14)
        and Eq. (15) }
        else { Random initialization of input weight $\mathscr{w}_{\mathcal{S}_i}$
        and bias $\mathscr{b}_{\mathcal{S}_i}$ for current state.}}
        end.
Step 2: Hidden layer outputs matrix $f_{(\mathcal{H}_n)_{\mathcal{S}_i}}$ is calculated
according to the Eq. (13)
Step3: Output weight matrix $\hat{\beta}_{\mathcal{S}_i}$ with $\ell_2 - norm$ is
computed according to Eq. (17)
Step 4: preserve all state variables
        $i = i + 1$;
} end while
```

For each of the three databases, all face images are cropped and resized to 32×32 and represented as a 1024 dimensional vector. Six training samples per subject are randomly chosen for training.

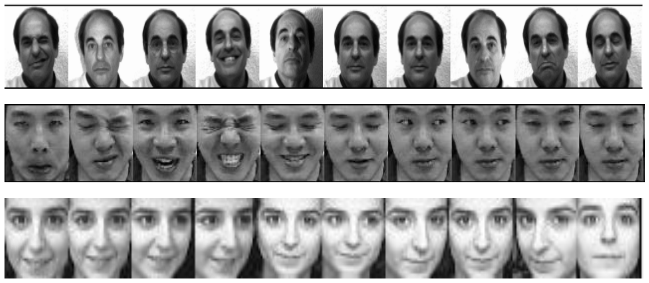
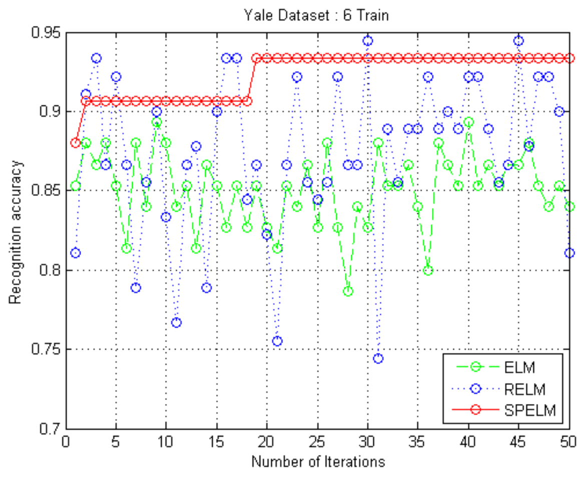


Fig. 3. The three rows show ten image samples from the Yale, CMU-AMP databases and ORL, respectively.

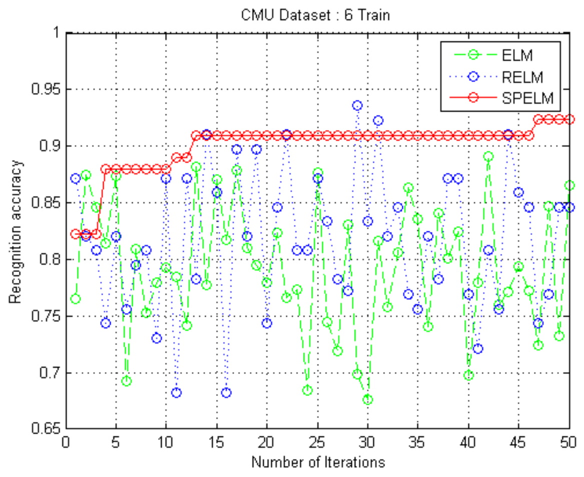
TABLE I. STATISTICS OF THREE FACE DATASET USED IN TEST

Database	#Samples	#Classes	#Sample/class
Yale	165	15	11
ORL	400	40	10
CMU-AMP	975	13	75

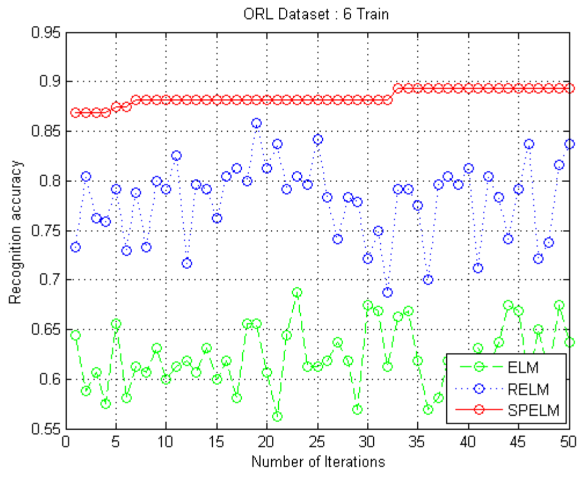
Results and Comparison: In this experiment, we compare the SPELM model with ELM and RELM. The algorithm procedure is repeated 50 times to produce a better estimation of recognition accuracy. Fig. 3 illustrates the recognition results on the Yale, CMU-AMP and ORL face databases without applying any feature extraction. From Fig. 4, it is evident that the proposed SPELM model yields better performance on all the three datasets. In each iteration stage SPELM gives a better recognition rate than conventional ELM and RELM for the fixed number of hidden nodes generated automatically.



(a)



(b)



(c)

Fig. 4. Test results on face recognition using ELM, RELM and SPELM: (a) Yale, (b) CMU, and (c) ORL datasets.

Monotonic increasing learning: Due to the state persevering properties of SPELM, the recognition accuracy is monotonically increasing during each iteration. In ELM and RELM the accuracy can decrease in any iteration as shown in Fig. 4. This is because SPELM preserves the output weight variables and adaptively updates them when superior weights are obtained. Fig. 4 shows that although the number of hidden neurons are fixed in each iteration, the overall performance of the ELM and RELM networks show scholastic behavior on the outputs. This is due to their random generation of weights and bias in each state. In contrast, SPELM yields monotonically increasing output accuracy with respect to iterations. This adaptive learning property would significantly boost the learning characteristics of ELM for achieving a better classification accuracy.

Feature embedding: To further demonstrate the efficiency of SPELM, we apply some popular feature extraction techniques, namely LBP, PHOG, and Gabor, on the raw inputs, and then perform the ELM based classification. In this experiment, the LBP feature vector is set to a length of 256. For PHOG we chose three pyramid levels with 9 bins histogram for each grid cell. In Gabor, 16 filters were used with a size of 8×8. Table 2 shows the face recognition accuracy of ELM, RELM and SPELM using these three features separately. These results show that SPELM provides the best performance in all three face datasets, thus demonstrating its robustness. To better visualize the test results, Figs. 5, 6, and 7 provide comparative histograms corresponding to Table 2 that show face recognition rate along with standard deviation on the Yale, CMU-AMP, and ORL face databases, respectively.

TABLE II. RECOGNITION ACCURACY (MEAN ± STD.-DEV. %)

Methods	Yale Database		
	LBP	PHOG	Gabor
ELM	77.47±4.06	99.33±0.51	97.47±1.22
RELM	82.23±1.12	99.53±0.65	98.07±0.81
SPELM	85.45±1.33	100.00±0.00	99.80±0.13
	CMU-AMP Database		
	LBP	PHOG	Gabor
ELM	93.66±0.68	99.70±0.18	100.00±0.00
RELM	96.08±0.24	99.55±0.16	100.00±0.00
SPELM	96.96±0.12	99.81±0.16	100.00±0.00
	ORL Database		
	LBP	PHOG	Gabor
ELM	58.50±2.39	90.06±1.03	97.50±0.35
RELM	76.63±1.85	91.19±0.95	97.56±0.35
SPELM	79.47±1.79	92.45±1.55	97.97±0.28

Time efficiency: The state preserving characteristics of the SPELM also contributes to computation speed. In ELM, the weights and bias are generated randomly, whereas the variables are only recomputed if a higher accuracy is found in SPELM. This saves a significant amount of memory and enhances the system processing speed. To experimentally show these merits, we used a desktop computer with a 1.7 GHz processor and 6GB of RAM to evaluate the processing time in MATLAB (R2014a). The evaluation is conducted on the three face

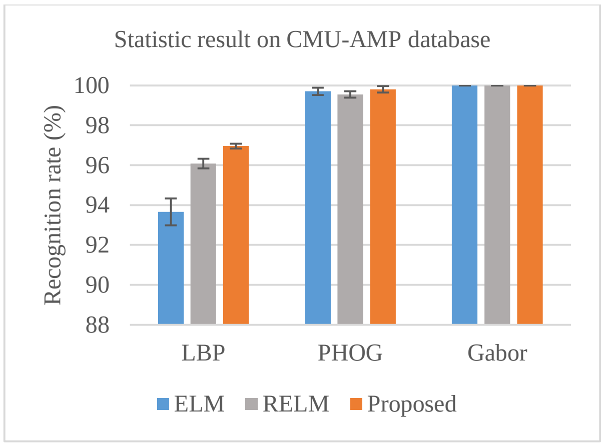


Fig. 5. Testing result on Yale Dataset with respect to LBP, PHOG, and Gabor features.

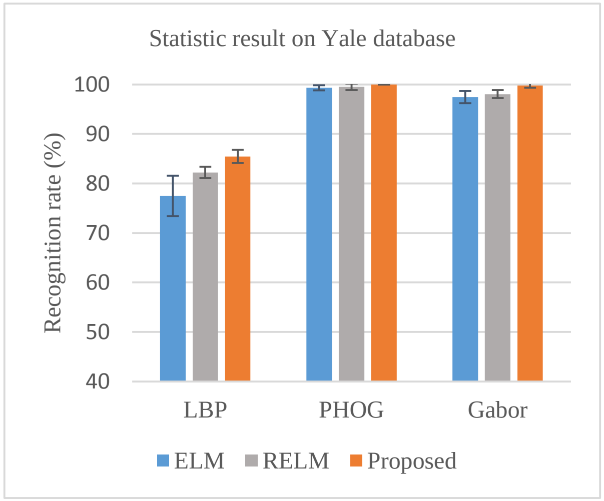


Fig. 6. Testing result on CMU-AMP dataset with respect to LBP, PHOG, and Gabor features.

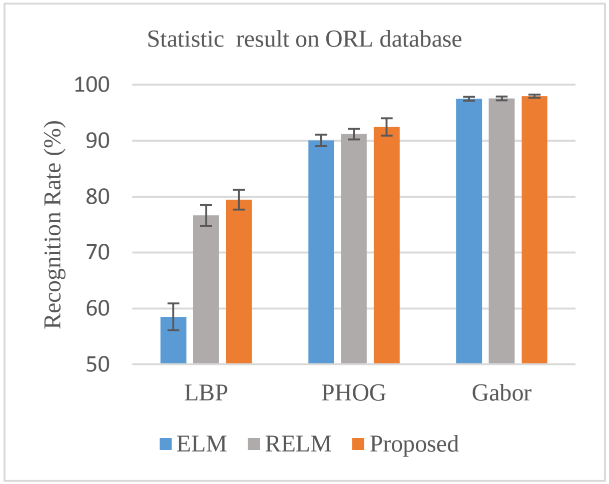


Fig. 7. Testing result on ORL dataset with respect to LBP, PHOG, and Gabor features.

databases using ELM, RELM, and SPELM. To avoid any bias, we repeat the experiments 50 times (iteration) and compute the average processing speed as shown in Table 3. From Table 3, it is clear that SPELM is the fastest.

TABLE III. A COMPARISON OF COMPUTATION TIME (MEAN: SEC./ITERATION)

Database	ELM	RELM	SPELM
Yale	0.406	0.385	0.278
ORL	0.230	0.155	0.140
CMU-AMP	0.244	0.165	0.150

IV. CONCLUSIONS

In this paper, we proposed a new approach for computing state variables in ELM, namely SPELM. We incorporated a monotonically increasing learning strategy by preserving state variables in each training and testing iteration. This improves the inherent characteristics of the ELM based classification algorithm. After evaluating SPELM on three different face databases, we observed that the proposed technique provides outstanding performance in comparison with conventional ELM and RELM. We are currently implementing SPELM in high performance computing systems using CUDA and MPI or OpenMP.

REFERENCES

[1] G.-B Huang, Q.Y. Zhu, C.-K Siew, “Extreme learning machine: theory and applications,” Neurocomputing, 70, pp. 489–501, 2006.

[2] G.-B Huang, D. H. Wang, and Y. Lan, “Extreme learning machines: a survey,” Int. J. Mach Learn Clybern., 2(2), pp. 107–122, 2011.

[3] G.-B Huang, L. Chen, C.-K Siew, “Universal approximation using incremental constructive feedforward networks with random hidden nodes,” IEEE Trans Neural Netw.,17(4), pp. 879–92,2006.

[4] G.-B Huang, H. Zhou, X. Ding, R. Zhang, “Extreme Learning machine for regression and multiclass classification,” IEEE Trans. Syst. Man Cybern. Part B Cybern. 42(2), pp. 513–529, 2012.

[5] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, “Error minimized extreme learning machine with growth of hidden nodes and incremental learning,” IEEE Transactions on Neural Networks, vol. 20, no. 8, pp. 1352–1357, 2009.

[6] G. Zhao, Z. Shen, and Z. Man, “Robust input weight selection for well-conditioned extreme learning machine,” International Journal of Information Technology, vol. 17, no. 1, 2011.

[7] A. C. P. Kulaif and F. J. V. Zuben, “Improved regularization in extreme learning machines,” in Annals of Congresso Brasileiro de Inteligłncia Computacional (CBIC), 2013.

[8] P. L. Bartlett, “The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network,” Information Theory, IEEE Transactions on, vol. 44, no. 2, pp. 525–536, 1998.

[9] W. Deng, Q. Zheng, and L. Chen, “Regularized extreme learning machine,” IEEE CIDM., pp. 389–395, 2009.

[10] Y. Wang, F. Cao, and Y. Yuan, “A study on effectiveness of extreme learning machine,” Neurocomputing, vol. 74, no. 16, pp. 2483–2490, 2011.

[11] J. M. Martnez-Mart´ınez, P. Escandell-Montero, E. Soria-Olivas, J. D. Mart´ın-Guerrero, R. Magdalena-Benedito, and J. G´omez-Sanchis, “Regularized extreme learning machine for regression problems,” Neurocomputing, vol. 74, no. 17, pp. 3716–3721, 2011.

[12] H. T. Huynh and Y. Won, “Weighted least squares scheme for reducing effects of outliers in regression based on extreme learning machine,” J. Digital Content Technol. Appl.(JDCTA), vol. 2, no. 3, pp. 40–46, 2008.

[13] A. L. B. Barros and G. A. Barreto, “Building a robust extreme learning machine for classification in the presence of outliers,” in Hybrid Artificial Intelligent Systems, ser. Lecture Notes in Computer Science, J.-S. Pan, M. Polycarpou, M. Woniak, A. C. Carvalho, H. Quintin, and E. Corchado, Eds. Springer Berlin Heidelberg, vol. 8073, pp. 588–597, 2013.

[14] P. Horata, S. Chiewchanwattana, and K. Sunat, “Robust extreme learning machine,” Neurocomputing, vol. 102, pp. 31–34, 2013.

[15] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, “Face recognition: A literature survey,” ACM Computing Survey, 34(4), pp. 399–485, 2003.

[16] Y. S. Huang and C. Y. Suen, “A method of combining multiple experts for the recognition of unconstrained handwritten numerals,” IEEE TPAMI, 17(1), pp. 90–94, 1995.

[17] C. Liu and H. Wechsler, “A shape- and texture-based enhanced fisher classifier for face recognition,” IEEE TIP, 10(4), pp. 598–608, 2001.

[18] M. Lades, J. C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Wurtz, and W. Konen, “Distortion invariant object recognition in the dynamic link architecture,” IEEE Trans. Comput., 42(3), pp. 300–311, 1993.

[19] L. Wiskott, J.-M. Fellous, N. Kruger, and C. von der Malsburg, “Face recognition by elastic bunch graph matching,”. IEEE TPAMI, 19(7), pp. 775–779, 1997.

[20] C. Liu, “Capitalize on dimensionality increasing techniques for improving face recognition grand challenge performance,” IEEE TPAMI, 28(5), pp. 725–737, 2006.

[21] T. Ojala, M. Pietikainen, and D. Harwood, “A comparative study of texture measures with classification based on feature distributions,” Pattern Recognition, 29(1), pp. 51-59, 1996.

[22] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” IEEE TPAMI, 24(7), pp. 971–987, 2002.

[23] T. Ahonen, A. Hadid, and M. Pietikainen, “Face description with local binary patterns: Application to face recognition,” IEEE TPAMI, 28(12), 2006.

[24] Anna Bosch, Andrew Zisserman and Xavier Munoz, “Representing shape with a spatial pyramid kernel,” International Conference on Image and Video Retrieval, pp. 401-408, 2007.

[25] L. Shen, and L. Bai, “A review on Gabor wavelets for face recognition,” Pattern Analysis and Application, 9, pp. 273–292, 2006.

[26] Yale face database, http://vision.ucsd.edu/content/yale-face-databas.

[27] X. Liu, T. Chen, and B. V. K. Vijaya Kumar, “Face authentication for multiple subjects using eigenflow,” Pattern. Recog. 36(2), pp. 313–328, 2003.

[28] F. S. Samaria, A. C. Harter, “Parameterisation of a stochastic model for human face identification,” Proceedings of the Second IEEE Workshop on Applications of Computer Vision, pp. 138-142, 1994.