Neural Process Lett
DOI 10.1007/s11063-016-9552-8

State Preserving Extreme Learning Machine: A Monotonically Increasing Learning Approach

Md. Zahangir Alom[1] · Paheding Sidike[1] · Tarek M. Taha[1] · Vijayan K. Asari[1]

Abstract Extreme Learning Machines (ELM) has been introduced as a new algorithm for training single hidden layer feedforward neural networks instead of the classical gradient-based approaches. Based on the consistency property of data, which enforces similar samples to share similar properties, ELM is a biologically inspired learning algorithm that learns much faster with good generalization and performs well in classification tasks. However, the stochastic characteristics of hidden layer outputs from the random generation of the weight matrix in current ELMs leads to the possibility of unstable outputs in the learning and testing phases. This is detrimental to the overall performance when many repeated trials are conducted. To cope with this issue, we present a new ELM approach, named State Preserving Extreme Leaning Machine (SPELM). SPELM ensures the overall training and testing performance of the classical ELM while monotonically increases its accuracy by preserving state variables. For evaluation, experiments are performed on different benchmark datasets including applications in face recognition, pedestrian detection, and network intrusion detection for cyber security. Several popular feature extraction techniques, namely Gabor, pyramid histogram of oriented gradients, and local binary pattern are also incorporated with SPELM. Experimental results show that our SPELM algorithm yields the best performance on tested data over ELM and RELM.

Keywords Extreme learning machine (ELM) · Face recognition · Pedestrian detection · Intrusion detection · Feature extraction · State preserving ELM

✉ Md. Zahangir Alom
alomm1@udayton.edu

Paheding Sidike
pahedings1@udayton.edu

Tarek M. Taha
ttaha1@udayton.edu

Vijayan K. Asari
vasari1@udayton.edu

1 Department of Electrical and Computer Engineering, Universtiy of Dayton, 300 College Park, Dayton, OH 45469, USA

1 Introduction

Machine learning (ML) has apparently never been as serious and significant to real-life applications as they are in today's autonomous and big data era. In the last decade, several new prominent ML techniques have been developed and applied in different fields for various applications and have proven their potential successfully. However, the success of ML approaches depend on the existence of three necessary conditions: powerful computing environments, rich and large data, and efficient learning techniques. It is also observed that neural network based learning approaches became more popular and reliable again among the ML community. A standard neural network (NN) consists of many connected processors called neurons, each producing a sequence of real-valued activations. Input neurons are activated through sensors perceiving the environment, while other neurons are activated through weighted connections from previously active neurons. Some neurons may influence the environment by triggering actions. Learning or credit assignment is about finding weights that make NNs exhibit desired behavior, such as learning human faces, pedestrians, or cyber security threats. Depending on the problems and how the neurons are connected, such behavior may require more computational stages, where each stage transforms the aggregate activation of the network. Back-propagation (BP) is an efficient gradient descent method for teacher-based Supervised Learning (SL) in discrete, differentiable networks of arbitrary depth [39].

According to the directional structure of NN internal information transfer, NNs can be divided into two categories: feedforward neural networks and feedback neural networks. Due to the ability of approximate complex nonlinear mapping directly from the input samples, the feedforward neural network model has been extensively used in many fields. It has been proven that if the activation function is continuous, bounded, and non-constant, then continuous mappings can be approximated by NNs over compact input sets [16]. As shown by Leshno et al. [31], feedforward networks with a non-polynomial activation function can approximate continuous functions. Huang et al. [4] shows that for L hidden nodes, a nonlinear activation function, single hidden layer feedforward neural network (SLFN) can perfectly approximate L distinct instances. It is also shown that a single layer has a strong learning ability which can approximate a complex non-linear function and can solve problems that cannot be solved by parametric learning algorithms [17]. Traditionally, all parameters of feedforward neural networks need to be tuned and thus there exists the dependency between different layers of parameters. For the past decades, gradient descent-based methods have mainly been used in various learning algorithms of feedforward neural networks. However, it is clear that gradient descent-based learning methods are generally very slow and computationally expensive due to their incorrect convergence to local minima. Many iterative learning steps may be required by such learning algorithms in order to obtain better learning performance.

Huang et al. [23,24] recently proposed a new learning algorithm for the SLFN architecture called extreme learning machine (ELM) which overcomes the problems of traditional feedforward neural networks such as the presence of local minima, imprecise learning rate, and slow rate of convergence. ELM has significantly attracted machine learning community due to its higher regularization performance with a much faster speed [23–25]. The basic principle of ELM can be described as: when the input weights and biases are randomly allocated, the output weights are computed by the generalized inverse of the hidden layer output matrix (H). Generally, ELM can be viewed as a SLFN with L hidden neurons that can learn L distinct samples with zero error. The main advantage of ELM is that the hidden layer of SLFNs need not be tuned, even if the number of hidden neurons is less than the number of distinct samples. ELM can also assign random parameters to the hidden nodes and calculate the output weights using the pseudo-inverse of H. This gives only a small error. 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 [23–25]. ELM is a simple unified algorithm for regression and binary or multi-class classification problems, and has been successfully tested on benchmark problems of practical importance. Due to its remarkable efficiency, simplicity, and impressive generalization performance, ELM have been applied in a variety of domains, such as biomedical engineering, computer vision, system identification, controls, and robotics.

We introduced an improved version of the ELM named State Preserving Extreme Learning Machine (SPELM), to achieve better performance by preserving and updating state variables that are instrumental to system accuracy and stability [47]. In this paper, we present SPELM with different applications that justify its robustness. The main contributions of this work can be summarized as:

- Development of SPELM learning algorithm for stabilizing the performance by preserving state variables such as weights and biases.
- Evaluation of the SPELM on face recognition, pedestrian detection, and network intrusion detection for cyber security.
- Incorporation of different prominent feature extraction methods namely Gabor, LBP, and PHOG with SPELM.
- Performance comparison of SPELM with ELM and RELM for several benchmarks.

The remainder of this paper is organized as follows: related works are discussed in Sect. 2. The theoretical analysis of ELM and SPELM are described in Sect. 3. Several experimental results are shown in Sect. 4. Section 5 explains the property of SPELM. Finally, conclusions of this work are drawn in Sect. 6.

2 Related Works

Several studies have been conducted in the field of ELM from both theoretical and application aspects. Huang et al. [11,22] 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. There is much ongoing research regarding this algorithm, such as optimizing the number of hidden neurons, stabilizing the overall performance, producing better generalization, etc. Although ELM is a much faster learning machine than other learning algorithms, the stochastic characteristics of the hidden layer output matrix may lower its learning accuracy [7]. Further, it was observed that because of the random selection of input weights and hidden node biases, a large number of hidden nodes might be required to achieve an acceptable level of performance. This is computationally expensive for big data problems [28], and thus more compact networks having the ability to achieve good generalization performance [46] is desired. Another issue in ELM is to choose the optimal number of hidden nodes, which is usually done by trial and error methods.

Two heuristic approaches, namely constructive and pruning methods, have been proposed in the literature to address above-mentioned problems [36]. A sparse model based ELM is proposed by Bai et al. [2] which suggests that many components of the optimal solution vector will become zero and therefore the decision function can be determined using much fewer hidden nodes in comparison to classical ELM. The selection of weights for the hidden neurons in ELM is very fast. Hence, compared to some classical methods, the overall computation time for model structure selection and actual training of the model is often reduced significantly. Furthermore, the algorithm remains rather simple, which makes its implementation easy.

However, there are inherent limitations in ELM. For example, initial hidden layer parameters (connection weight, bias, the number of hidden neurons) of ELM have big impacts on classification accuracy [24,33,43]. Huang et al. proposed I-ELM which increases the number of hidden layer nodes in ELM one by one to improve the convergence rate [22]. An improved version of I-ELM is introduced in [21], which is named Enhanced Incremental ELM (EI-ELM) that learns faster with a more compact network structure. The above-mentioned ELMs do not take into account the structural risk that may lead to over fitting problems. Deng et al. [9] presented a Regularized Extreme Learning Machine (RELM) which incorporates the structural risk minimization theory and the weighted least squares methods into the ELM. RELM has better generalization performance that not only minimizes the training errors, but also makes the edge distance maximized. Further, it has certain anti-jumping capabilities for outliers. The input weights and hidden layer of RELM are randomly assigned and the changes of the hidden layer output matrix of RELM may be very large [24,34]. This, in turn, results in large change of the output weight matrix, which greatly increases both empirical risk and structural risk, and degenerates the robustness.

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 and RELM yields much better generalization performance with much faster learning speed and less human interventions than other conventional methods but still has inconsistent behavior in its output accuracy. There are however many issues that have not been solved yet, such as ensuring performance stabilization and fixing the optimum number of hidden neurons. From our point 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 [44,48]. Those computational problems occur when the hidden layer output matrix is ill-conditioned, typically caused by the random input weights and bias 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 [48]. Additionally, it is known that the size of the output layer weight (in terms of number of neurons and format of the activation function) is more related to the generalization competency than the configuration of the neural network [45]. 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 than the Ordinary Least Squares (OLS). Huynh et al. substituted the singular value decomposition method by the weighted least squares (similar to OLS). However, it creates penalties corresponding to the weighted training patterns that contributes to the final outputs [26]. Barros and Barreto [3] concentrated their efforts on robust classification problems with a proposal of an ELM that used Iteratively Reweighted Least Squares (IRLS), named ROB-ELM Another study by Horata et al. [15] provided solutions for solving two problems (i.e., computational robustness and outlier robustness) of ELM. They proposed to use the Extended Complete Orthogonal Decomposition (ECOD) to solve the computational problem, and another three methods, namely iteratively reweighted least squares, Multivariate Least-Trimmed Squares (MLTS), one-step reweighted MLTS, to solve the outlier robustness problem.

In our previous work [47], we proposed SPELM which is based on the Regularized ELM (RELM) [9,35] and state variables preserving strategy. SPELM yields monotonically incremental behavior on accuracy instead of stochastic characteristics of ELM. This ensures a better overall performance in training and testing phases. The following section provides theoretical analysis of ELM and SPELM.

3 Theoretical Analysis

The traditional neural network learning algorithms (e.g. BP algorithm) require to set up and adjust many parameters to obtain optimal solution. In contrast, ELM based approaches only need to set number of hidden layer nodes and does not need to adjust the network input weights and hidden biases which leads to fast learning speed and better generalization performance. In this section, we first review conventional ELM, and then introduce the proposed SPELM framework.

3.1 Extreme Learning Machine (ELM)

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 [9,23].

A typical architecture of ELM is shown in Fig. 1. The output function of ELM with L hidden nodes for generalized SLFNs is expressed as [23]

$$f_L(\boldsymbol{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i g_i(\boldsymbol{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d,\ \boldsymbol{\beta}_i \in \mathbb{R}^m \tag{1}$$

where $\boldsymbol{a}_i \in \mathbb{R}^d$ 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 $G(\boldsymbol{a}_i, b_i, \boldsymbol{x})$ of the ith hidden node, and $\boldsymbol{\beta}_i = [\beta_1, \beta_2, \ldots, \beta_L]^T$ is the weight vector linking the ith hidden node to the output nodes. For n arbitrary distinct samples $(\boldsymbol{x}_j, \boldsymbol{t}_j)$, where $\boldsymbol{x}_j \in \mathbb{R}^d$ and $\boldsymbol{t}_j \in \mathbb{R}^m$ the SLFNs with L hidden nodes can approximate these n samples with zero error, meaning $\sum_{j=1}^{n} \|\boldsymbol{f}_j - \boldsymbol{t}_j\| = 0$. Hence, there exists $(\boldsymbol{a}_i, b_i)$ and $\boldsymbol{\beta}_i$, such that

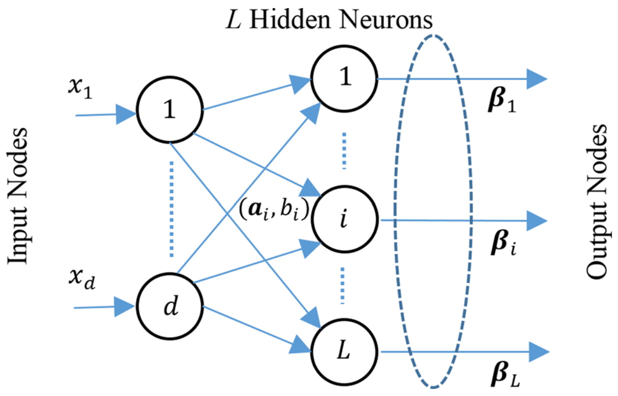


Fig. 1 A typical architecture of the ELM

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = \boldsymbol{t}_j, \quad j = 1, 2, \ldots, n \tag{2}$$

Equation (2) can be compactly written as

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T} \tag{3}$$

where

$$\boldsymbol{H} = \begin{bmatrix} h(\boldsymbol{x}_1) \\ \vdots \\ h(\boldsymbol{x}_n) \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_n) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_n) \end{bmatrix} \tag{4}$$

and

$$\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_m] \in \mathbb{R}^{L\times m}, \ \boldsymbol{T} = [\boldsymbol{t}_1^T, \boldsymbol{t}_2^T, \ldots, \boldsymbol{t}_n^T]^T \in \mathbb{R}^{n\times m}, \tag{5}$$

where $\boldsymbol{T}$ is the target matrix for training data, $\boldsymbol{H}$ is the hidden layer output matrix of the SLFN, and the ith column of $\boldsymbol{H}$ is the ith hidden node output with respect to inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, while the jth row, i.e., $h(\boldsymbol{x}_j)$, is the hidden layer feature mapping corresponding to the jth input x_j. As the hidden node parameters $(\boldsymbol{a}_i, b_i)$ can be randomly generated and remain unchanged, the only unknown parameters in ELM are the output weight vectors $_i$ 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 $\|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{T}\|$ and the norm of weights $\|\boldsymbol{\beta}\|$, the smallest norm least-squares solution of the above linear system is obtained by

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

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

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

Then the error matrix can be described as

$$\varepsilon = \|\boldsymbol{Y} - \boldsymbol{T}\|^2 = \|\boldsymbol{H}\boldsymbol{H}^{\dagger}\boldsymbol{T} - \boldsymbol{T}\|^2 \tag{8}$$

In order to increase the stability and generalization ability of the traditional ELM, the weight vector $\boldsymbol{H}^{\dagger}$ can be represented as [9]

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

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

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{H}\boldsymbol{H}^T + \lambda\boldsymbol{I})^{-1}\boldsymbol{H}^T\boldsymbol{T} \tag{10}$$

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

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

where $\|\boldsymbol{\beta}\|_2^2$ denotes the $l_2 - norm$ of $\boldsymbol{\beta}$. 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 ELM that may be summarized as in Algorithm 1 [5].

Algorithm 1: Extreme Learning Machine

```
  Input  : Number of training samples $n$ where
1 $\{(\boldsymbol{x}_j, \boldsymbol{t}_j) | \boldsymbol{x}_j \in \mathbb{R}^d, t_j \in \mathbb{R}^m, j = 1, 2, \ldots, n\}$.
2 Activation function $g_i(\boldsymbol{x})$, and number of hidden nodes $L$.
  Output:
3 Step 1: Input weight $\boldsymbol{a}_i$ and bias $b_i$ are initialized randomly, $i = 1, 2, \ldots, L$.
4 Step 2: Hidden layer outputs matrix $\boldsymbol{H}$ is calculated.
5 Step 3: Output weight matrix $\boldsymbol{\beta}$ is computed as follows:
6 $\hat{\boldsymbol{\beta}} = \boldsymbol{H}^{\dagger}\boldsymbol{T}$,
7 where $\boldsymbol{T} = [\boldsymbol{t}_1^T, \boldsymbol{t}_2^T, \cdots, \boldsymbol{t}_n^T]^T \in \mathbb{R}^{n \times m}$
```

3.2 Why SPELM?

The selection of the weight matrix in ELM for hidden neurons is very fast due to a non-iterative learning process. In contrast, some other classical learning methods require a number of trials to achieve the desired convergence in learning, which is computationally expensive. Studies in [23,24] showed that ELM initialization parameters (connection weights, the offset value, and the number of hidden neurons) have big impacts on classification accuracy. Moreover, I-ELM improves the classification rate in each trial by increasing the number of hidden layer neurons individually [21], whereas EI-ELM produces a more compact network structure with faster learning compared to I-ELM [21]. RELM incorporates the structural risk minimization theory and the weighted least squares methods into the ELM [9,35]. Although RELM shows better generalization performance compared to I-ELM and EI-ELM, the inconsistent behavior still exists in training and testing accuracy of RELM.

In case of ELM and RELM, it is not guaranteed that any given trial will provide better accuracy compared to its earlier trials. In addition, it is observed that accuracy deterioration occurs more than once during subsequent trials in ELM and RELM. These characteristics have a negative effect on overall performance. Consequently, we envisage that if weights and biases, which provide higher accuracy, are preserved, then the degradation in performance can be avoided in the successive trials. This leads to a monotonically increasing performance generally which ensures a better overall accuracy. Therefore, we propose SPELM to achieve such desired performance by preserving the optimum state variables seen such as weights and biases between trials. The mathematical details of SPELM are described in the following section.

3.3 Proposed State Preserving ELM

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 trial in case of batch learning; thirdly, the input samples are shuffled according to the output sequences of each trial. In contrast, in SPELM, training samples are randomly selected state variables such as weight, bias, train accuracy, and test accuracy are preserved for each trial. Then the highest accuracy with respect to relevant parameters are preserved until the following trials to provide a better accuracy. The same procedure will be continued until the end of the number of trials. The following section explains mathematical formulation of SPELM.

In SPELM, the state variables are the number of trials K, the state of the network s_k where $k = 1, 2, \ldots, K$, the accuracy of the state represented by A_{s_k}, the number of hidden nodes

L in state s_k denoted as $L_{s_k} \in \mathbb{Z}^+$, and the activation function $G(\boldsymbol{a}_{s_k}, b_{s_k}, \boldsymbol{x})$. The number of hidden nodes L_{s_k} for the state s_k is calculated based on the $d-$dimensional input features as [47]

$$L_{s_k} = \psi * d \tag{12}$$

where ψ is a constant which is empirically set to 10 in our experiments. The output function of SPELM with L_{s_k} hidden nodes for generalized SLFNs is expressed as:

$$f_{L_{s_k}}(\boldsymbol{x}) = \sum_{i=1}^{L_{s_k}} \boldsymbol{\beta}_{s_k} g_i(\boldsymbol{x}) \tag{13}$$

The weight matrix $\boldsymbol{a}_{s_k}$ and the bias b_{s_k} in the state of s_k are updated with respect to the present accuracy (A_{s_k}) and the immediate previous accuracy $(A_{s_{k-1}})$. These terms are defined by Eqs. (14) and (15), respectively.

$$\boldsymbol{a}_{s_k} = \begin{cases} \boldsymbol{a}_{s_k}, & if \ A_{s_k} > A_{s_{k-1}} \\ \boldsymbol{a}_{s_{k-1}}, & \text{otherwise} \end{cases} \tag{14}$$

$$b_{s_k} = \begin{cases} b_{s_k}, & if \ A_{s_k} > A_{s_{k-1}} \\ b_{s_{k-1}}, & \text{otherwise} \end{cases} \tag{15}$$

For n arbitrary distinct samples $(\boldsymbol{x}_j, \boldsymbol{t}_j)$, where $\boldsymbol{x}_j \in \mathbb{R}^d$ and $\boldsymbol{t}_j \in \mathbb{R}^m$ the SLFNs with Ls_i hidden nodes can approximate these n samples with zero error. Hence $\sum_{j=1}^{n} \|\boldsymbol{f}_j - \boldsymbol{t}_j\| = 0$, there exists $(\boldsymbol{a}_{s_k}, b_{s_k})$ and $\boldsymbol{\beta}_{s_k}$, such that

$$\sum_{i=1}^{L_{s_i}} \boldsymbol{\beta}_{s_k} G(\boldsymbol{a}_{s_k}, b_{s_k}, \boldsymbol{x}) = \boldsymbol{t}_j, \quad j = 1, 2, \ldots, n \tag{16}$$

Consequently, Eq. (10) can be rewritten as

$$\hat{\boldsymbol{\beta}}_{s_k} = (\boldsymbol{H}_{s_k} \boldsymbol{H}_{s_k}^T + \lambda \boldsymbol{I})^{-1} \boldsymbol{H}_{s_k}^T \boldsymbol{T} \tag{17}$$

The solution for the Eq. (17) can be obtained by solving the following optimization problem:

$$\min_{\boldsymbol{\beta}_{s_k}} \|\boldsymbol{\beta}_{s_k}^T \boldsymbol{H}_{s_k} - \boldsymbol{Y}\|_2^2 + \lambda \|\boldsymbol{\beta}_{s_k}\|_2^2 \tag{18}$$

where $\|\boldsymbol{\beta}_{s_k}\|_2^2$ is considered as $l_2 - norm$ of $\boldsymbol{\beta}_{s_k}$, and λ is a constant. In order to update the state accuracy A_{s_k} on test samples, the prediction value matrix $\boldsymbol{Y}_{s_k}$ is expressed by

$$\boldsymbol{Y}_{s_k} = \boldsymbol{H}_{s_k} \hat{\boldsymbol{\beta}}_{s_k} \tag{19}$$

Then the corresponding error can be described as

$$\varepsilon = \|\boldsymbol{Y}_{s_k} - \boldsymbol{T}\|^2 \tag{20}$$

Finally, the test accuracy of state A_{s_k} updates based on ε as follow

$$A_{s_k} = (1 - \varepsilon) \times 100 \tag{21}$$

The implementation of the above mentioned SPELM algorithm can be illustrated as in Algorithm 2.

Algorithm 2: State Preserving ELM

```
   Input  : Number of training samples $n$ where
1  $\{(\boldsymbol{x}_j, \boldsymbol{t}_j) | \boldsymbol{x}_j \in \mathbb{R}^d, t_j \in \mathbb{R}^m, j = 1, 2, \ldots, n\}$
2  Activation function $g(\boldsymbol{x})$, number of hidden nodes $L$, the $i$th state $s_k(k = 1, 2, \ldots, K)$, and $i$th state
   accuracy $A_{s_k}$
   Output:
3  Step 1: Input weight $\boldsymbol{a}_i$ and bias $b_i$ are initialized randomly, $i = 1, 2, \ldots, L$
4  while($k \leq K$)
5  if $k < 2$ then
6      Random initialization of input weight $\boldsymbol{a}_{s_k}$ and bias $b_{s_k}$ for first state;
7  else
8      if $A_{s_k} \geq A_{s_{k-1}}$ then
9          Update weight and bias according to the Eq. (14) and Eq. (15);
10     else
11         Random initialization of input weight $\boldsymbol{a}_{s_k}$ and bias $b_{s_k}$ current state.
12     end
13 end
14 end
15 Step 2: The output function of SPELM is computed according to Eq. (13).
16 Step 3: Output weight matrix $\hat{\boldsymbol{\beta}}_{s_k}$ with $l_2 - norm$ is obtained through Eq. (17).
17 Step 4: Preserve all state variables, i.e., weights and bias.
```

3.4 Universal Approximation of SPELM

Studies in [18–22] have shown that SLFNs with randomly generated additive nodes or radial basis function (RBF) hidden nodes has universal approximation capability. Given a sample input space X, the universal approximation of ELM is proved in an incremental way, known as incremental ELM (I-ELM) [22]. Let $e_q \equiv f - f_q$ where e_q is the error function for current network f_q with q hidden nodes, then we have the following Lemma.

Lemma ([20,22]). *Given any non-constant piecewise continuous function as the activation function $g : \mathbb{R} \rightarrow \mathbb{R}$ for additive hidden nodes or RBF nodes, for any continuous target function f, and any randomly generated sequence g_q based on any continuous sampling distribution, $lim_{q\rightarrow\infty} \|f - f_q\| = 0$ holds with probability 1 if*

$$\beta_q = \frac{\langle e_{q-1}, g_q \rangle}{\|g_q\|^2} \tag{22}$$

This Lemma shows that SLFNs can work as universal approximators by randomly choosing hidden neurons parameters and analytically calculating the output weights. Based on the similar proof of [22] we can further extend the universal approximation capability of ELM to SPELM with continuous number of trials. Difference from I-ELM, SPELM does not minimize the error through additive nodes, where it aims to decrease error through a number of trials while preserving the state variable such as weights and biases.

Theorem *Given any nonlinear piecewise continuous activation functions or their linear combinations as the activation function $g : \mathbb{R} \rightarrow \mathbb{R}$ for additive iterations (states) with adjustable hidden nodes, for any continuous target function f and any randomly generated sequence g_{s_k} for a specific state, $lim_{s\rightarrow\infty} \|f - f^L_{s_k}\| = 0$, where $f^L_{s_k}$ is the current state output function with L hidden nodes, holds with probability 1 if*

$$\beta_{s_k} = \frac{\langle e_{s_{k-1}}, g_{s_k}\rangle}{\|g_{s_k}\|^2} \tag{23}$$

Proof The proof strategy of I-ELM [22] can be adopted to prove above SPELM Theorem. Since $f_{s_k}^L$ is continuous and g_{s_k} is a nonconstant piecewise continuous function, we can prove that $\|e_{s_k}\| = \|f - (f_{s_{k-1}}^L + \beta_{s_k} g_{s_k})\|$ achieves its minimum if

$$\beta_{s_k} = \frac{\langle e_{s_{k-1}}, g_{s_k}\rangle}{\|g_{s_k}\|^2} \tag{24}$$

Consequently, the error sequence $\{\|e_{s_k}\|\}$ decreases and converges.

Let $\triangle = \|e_{s_{k-1}}\|^2 - \|e_{s_k}\|^2$. In SPELM, we consider the state function parameters are preserved until $e_{s_k} \leq e_{s_{k-1}}$ given sufficient number of trials of hidden nodes, thus $\triangle$ is always positive. Accordingly, similar to [22], we have

$$\begin{aligned}\triangle &= \|e_{s_{k-1}}\|^2 - \|f - (f_{s_{k-1}}^L + \beta_{s_k} g_{s_k})\|^2 \\ &= \|e_{s_{k-1}}\|^2 - \|e_{s_{k-1}} - \beta_{s_k} g_{s_k}\|^2 \\ &= \|g_s\|^2 \left(\frac{\langle e_{s-1}, g_{s_k}\rangle^2}{\|g_{s_k}\|^4} - \left(\beta_{s_k} - \frac{\langle e_{s_{k-1}}, g_{s_k}\rangle}{\|g_{s_k}\|^2} \right)^2 \right)\end{aligned} \tag{25}$$

From Eq. (25), $\triangle$ is maximized iff $\beta_{s_k} = \langle e_{s_{k-1}}, g_{s_k}\rangle / \|g_{s_k}\|^2$ and it is always equal or greater than zero, meaning that the error sequence $\{\|e_{s_k}\|\}$ is decreasing and thus it converges.

This completes the proof of this Theorem. □

4 Experimental Results

In following experiments, we evaluate ELM, RELM and SPELM on different benchmarks for face recognition, pedestrian detection and network intrusion detection for cyber security. For face recognition, three popular datasets namely Yale, CMU-AMP, and ORL are used. INRIA pedestrian dataset is used for pedestrian detection. Finally, KDD Cup 1999 dataset is used for network intrusion detection for cyber security. Details on these datasets are given in experimental results section. All of the experiments have been conducted on a desktop computer with an Intel @ Core$^{\text{TM}}$ 2 Duo CPU E86 @ 3.33 GHz processor and 4GB of RAM to evaluate the processing time in MATLAB (R2013a).

4.1 Experiments on Face Recognition

This section presents the experimental results of our proposed SPELM model for face recognition. To evaluate the performance of the approach, we test the SPELM on three popular face recognition databases, namely Yale, CMU-AMP, and ORL.

To show the effectiveness of the SPELM, we further evaluate its performance by incorporating local appearance descriptors for face recognition [49], such as Gabor wavelet [30], Local Binary Patterns (LBP) [1,38] and Pyramid Histogram of Orientated Gradients (PHOG) features [6], into SPELM for face recognition. A systematic scheme is depicted in Fig. 2. 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 [42]. PHOG features descriptor is mainly inspired by two sources: (i) the image pyramid representation [29], and (ii) the Histogram of Oriented Gradients (HOG) [8].

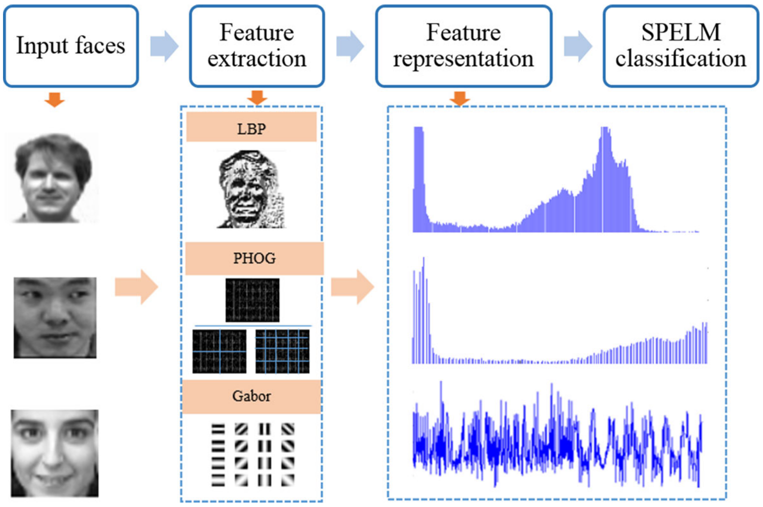


Fig. 2 Proposed implementation scheme for face recognition

All of above-mentioned three features (i.e., Gabor, LBP, PHOG) are rich in information content and computational efficiency. Thus, in this paper, we integrate these three feature extraction techniques with SPELM for evaluation. Test results show that feature based SPELM yields a better face recognition accuracy. 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.

4.1.1 Dataset and Test Setup

To evaluate the efficiency of the SPELM, we perform unconstrained face verification experiments on the Yale [4], CMU-AMP [32] and ORL [41] 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. For each of the three databases, all face images are cropped and resized to 32×32 and represented as a 1024 dimensional vector. Only six training samples per subject are randomly chosen for training and the rest date used for testing. 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. For Gabor feature, 16 filters were used with a size of 8×8.

4.1.2 Results and Comparison

In this experiment, we compare the SPELM model with ELM and RELM. The algorithm procedure is repeated 50 trials to produce a better estimation of recognition accuracy. Fig. 4 illustrates the recognition results on the Yale, CMU-AMP and ORL face databases without

Table 1 Statistics of three face dataset used in training and testing

Database	No. of samples	No. of classes	No. of samples per class
Yale	165	15	11
ORL	400	40	10
CMU-AMP	975	13	75

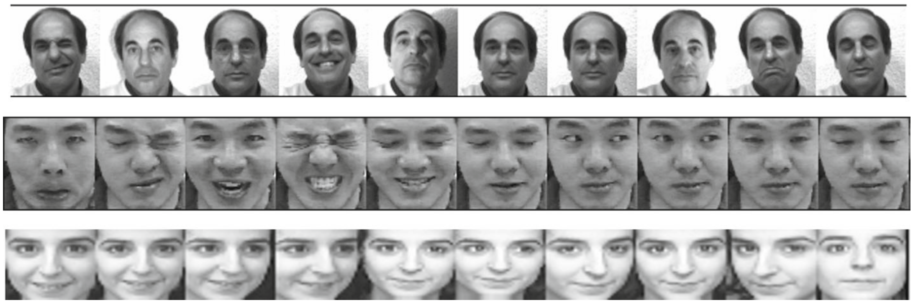


Fig. 3 The three rows show ten image samples from the Yale, CMU-AMP databases and ORL, respectively

applying any feature extraction. From these results, it is evident that the proposed SPELM model yields better performance on all the three datasets. Furthermore, In each trial, SPELM yields higher accuracy than ELM and RELM for the fixed number of hidden nodes generated automatically.

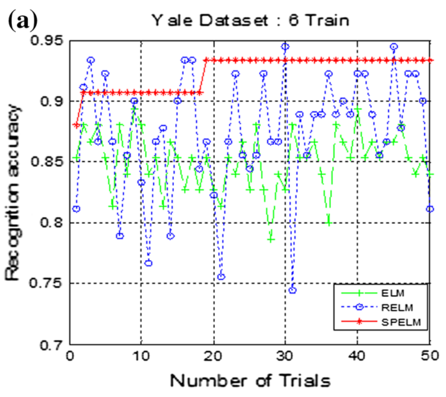
Table 2 shows the face recognition accuracy of ELM, RELM and SPELM using aforementioned features separately. From these results, it can be seen that SPELM provides the best performance in all three face datasets, thus demonstrating its robustness. To better visualize the test results, Figs. 5, 6, and 7, corresponding Table 2, show face recognition rate along with its standard deviation on the Yale, CMU-AMP, and ORL face databases, respectively.

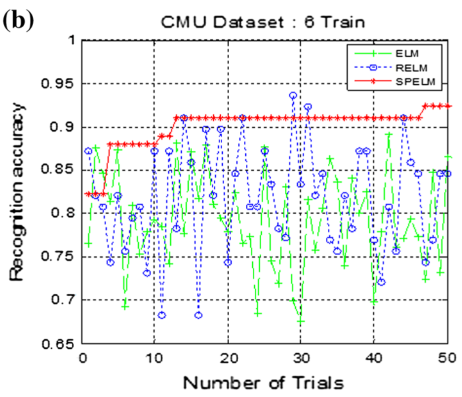
4.1.3 Time Efficiency

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. The evaluation is conducted on the three face databases using ELM, RELM, and SPELM. The average processing speed over 50 trials is shown in Table 3. From Table 3, it is clear that SPELM is computationally faster than the competitors.

4.2 Experiments on Multi-view Pedestrian Detection

Pedestrian detection and tracking is an important task in the field of computer vision. With the proliferation of high-powered computers, availability of high quality video cameras, and the increasing need for automated video analysis, human detection and tracking has generated a great deal of interest for many research fields [12,37]. In this paper, we investigate PHOG features with ELM, RELM and SPELM for pedestrian detection. Furthermore,the





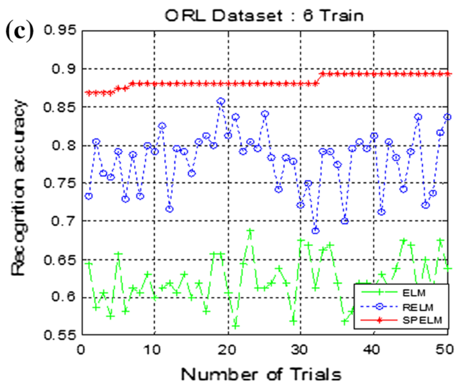


Fig. 4 Test results on face recognition using ELM, RELM and SPELM: **a** Yale. **b** CMU-AMP. **c** ORL datasets

Table 2 Recognition accuracy (mean SD %)

Methods	LBP	PHOG	Gabor
Yale database			
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			
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			
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

performance of SPELM is evaluated against ELM and RELM. The details on PHOG features are discussed in Section 4.1. Discussion on dataset and experimental results are given below.

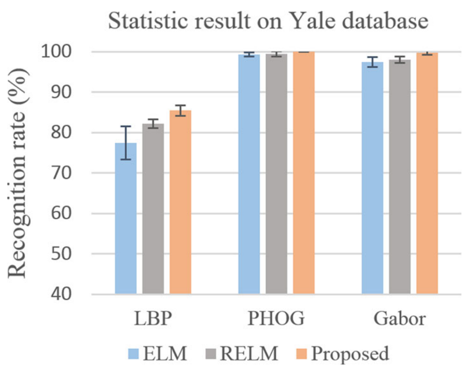


Fig. 5 Test result on Yale dataset with respect to LBP, PHOG, and Gabor features

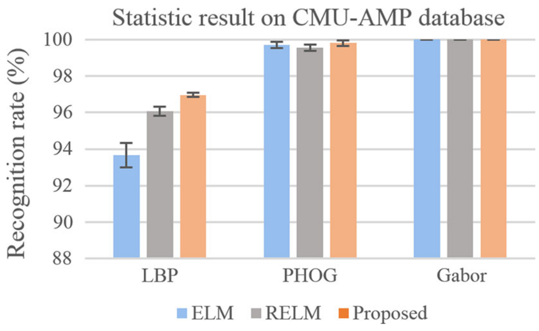


Fig. 6 Test result on CMU-AMP dataset with respect to LBP, PHOG, and Gabor features

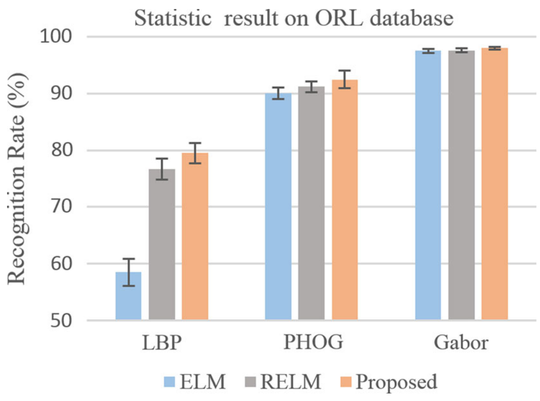


Fig. 7 Test result on ORL dataset with respect to LBP, PHOG, and Gabor features

Table 3 A comparison of computation time (mean: sec./trail)

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

4.2.1 Dataset and Test Setup

Multi-view INRIA pedestrian dataset [8] is used in this experiment.The image size in this database is 80 × 32. The proposed method is trained with 2000 multi-view positive and 3000

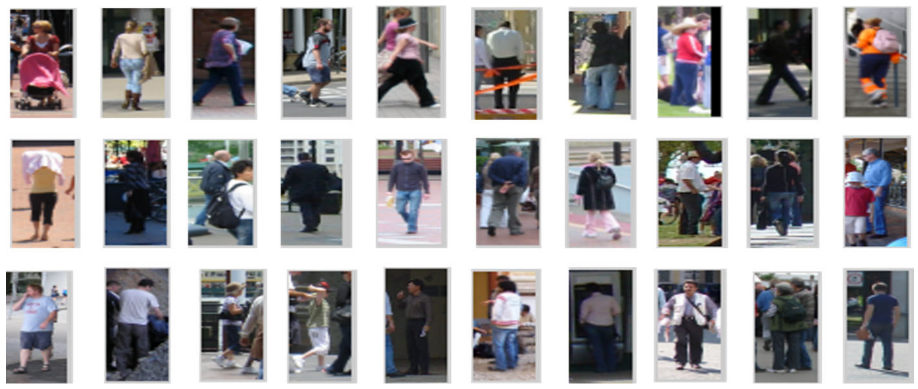


Fig. 8 Example positive images in multi-view pedestrian database

negative images. Some of the positive examples are shown in Fig. 8. In SPELM, 2560 nodes is set for the input layer, 1000 neurons for the hidden layer, and 2 nodes for the output layer. A total of 30 trials were conducted for training and testing.

4.2.2 Results and Analysis

Fig. 9 shows pedestrian detection accuracy on INRIA dataset. It can be observed that the proposed PHOG + SPELM outperforms PHOG + ELM and PHOG + RELM in most of the trials and yields higher average recognition accuracy. From Fig. 9, it can be seen that the accuracy of PHOG + SPELM rises as number of trial increases. This is because of the inherent character of SPELM, in which it always preserves better weights and biases for achieving a better or equal recognition accuracy in each trial. The overall accuracy of PHOG + ELM, PHOG + RELM, and PHOG + SPELM are presented in Table 4 and the highest accuracy is indicated in boldface.

4.3 Experiments on Network Intrusion Detection

With advances in digital technology, security threats for computer networks have increased dramatically over the last decade, thus there is a need to develop more accurate systems for cyber security [5,10,13,40]. In this paper, we explore the capabilities of SPELM on intrusion detection for cyber security using KDD Cup 1999 dataset [14]. In addition, robustness of SPELM is evaluated using dimensionality reduction technique such as Principal Component Analysis (PCA) [27].

4.3.1 Dataset Description

The KDD Cup 1999 intrusion detection dataset is based on the Defense Advanced Research Projects Agency (DARPA) initiative to provide designers of intrusion detection systems (IDS) with a benchmark on which to evaluate different methodologies. Each KDD Cup 1999 connection record contains 41 features (e.g., protocol type, service, and flag) and is labeled as either normal or a specific type of attack as provided in Fig. 10. The cyber attacks can be summarized into five categorizes:

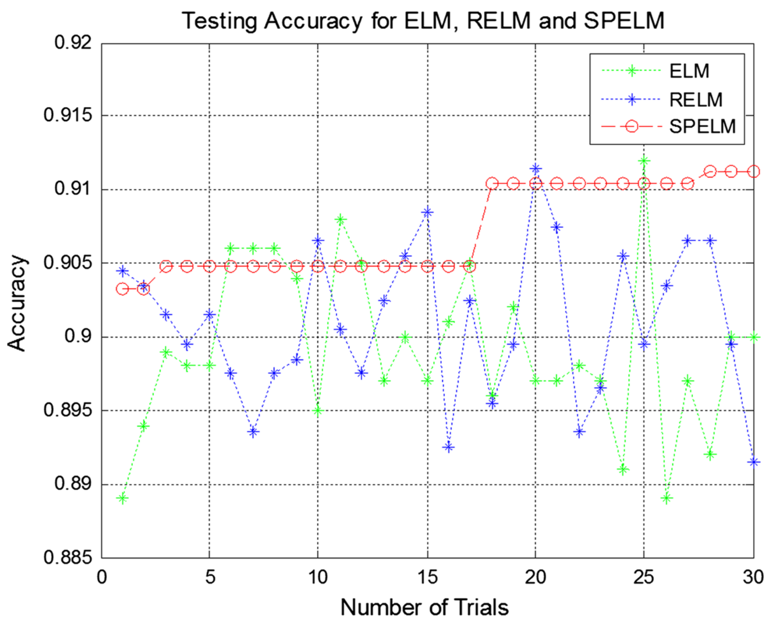


Fig. 9 Recognition accuracy for pedestrian classification

Table 4 Performance comparison and the best accuracy for each datasets is shown in boldface

Methods	Pedestrian detection accuracy
PHOG + ELM	89.92
PHOG + RELM	90.10
PHOG + SPELM	**90.70**

- Normal: Data with no attack.
- Denial of Service (DoS): Attacker tries to prevent legitimate users from using a service.
- Probe: Attacker tries to gain information about the target host.
- Remote to Local (R2L): Attacker does not have an account on the victim machine, hence tries to gain access.
- User to Root (U2R): Attacker has local access to the victim machine and tries to gain super user privileges.

In KDD Cup 1999, the training set contains a total of 5 categories of attack as given in Table 5. In this experiment, we use 25,000 numerical encoded and normalized data samples for training and testing.

4.3.2 Dataset Preprocessing and Results

The components within a packet (network protocols, services, flag, etc.) are represented as strings. In order to use these with SPELM, we encode each string as a single numerical value. Table 6 represents the corresponding values of individual strings for protocol names as an instance.

```
0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,
0.00,0.00,150,25,0.17,0.03,0.17,0.00,0.00,0.00,0.05,0.00,normal,20

0,udp,other,SF,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,1,0.00,0.00,0.00,0.00,0.08,
0.15,0.00,255,1,0.00,0.60,0.88,0.00,0.00,0.00,0.00,0.00,normal,15

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,123,6,1.00,1.00,0.00,0.00,0.05,0
.07,0.00,255,26,0.10,0.05,0.00,0.00,1.00,1.00,0.00,0.00,neptune,19

0,tcp,http,SF,232,8153,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.20,0.20,0.00,0.00,1.00,
0.00,0.00,30,255,1.00,0.00,0.03,0.04,0.03,0.01,0.00,0.01,normal,21

0,tcp,http,SF,199,420,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,30,32,0.00,0.00,0.00,0.00,1.00
,0.00,0.09,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal,21

0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0.00,0.00,1.00,1.00,0.1
6,0.06,0.00,255,19,0.07,0.07,0.00,0.00,0.00,0.00,1.00,1.00,neptune,21
```

Fig. 10 Sample data from the KDD Cup 1999 dataset

Table 5 Categorization of network attacks

Category	Example string	Class label
Normal	Normal network packet2	1
DoS	'Back', 'Land', 'Neptune', 'Pod', 'Smurf', 'Teardrop', 'Mailbomb', 'Processtable', 'Udpstorm', 'Apache2', 'Worm'	2
Probe	'Satan', 'IPsweep', 'Nmap', 'Portsweep', 'Mscan', 'Saint'	3
R2L	'Guess-password', 'Ftp-write', 'Imap', 'Phf', 'Multihop', 'Warez-master', 'Warezclient', 'Xlock', 'Xsnoop', 'Snmpguess', 'Snmpge-tattack', 'Httptunnel', 'Sendmail', 'Named'	4
U2R	'Buffer-overflow', 'Loadmodule', 'Rootkit', 'Perl', 'Sqlattack', 'Xterm', 'Ps'	5

Table 6 Protocol names with corresponding values (TCP: Transmission Control Protocol; UDP: User Datagram Protocol; ICMP: Internet Control Message Protocol)

Protocol name	TCP	UDP	ICMP	Otherwise
Value	1	2	3	0

All the remaining feature strings are encoded in a similar manner and the input numerical values are then normalized by a max-min normalization method maps all values into the range of [0 1]. After this normalization, some input data features end up being encoded as all zeros. These feature vectors are discarded from the input data set, thus reducing the dimension of the input feature from 41 to 39.

Figures 11, 12, and 13 show the classification results on KDD Cup 1999 dataset using 39-dimensional feature vector, whereas Figs. 14, 15 and 16 show the testing accuracy after reducing feature dimensions from 39 to 9 using PCA. From these results, it can be clearly seen that SPELM outperforms ELM and RELM in most of the trials, and yields higher average accuracy for both 39 and 9 dimensional feature vector as summarized in Tables 7 and 8, respectively.

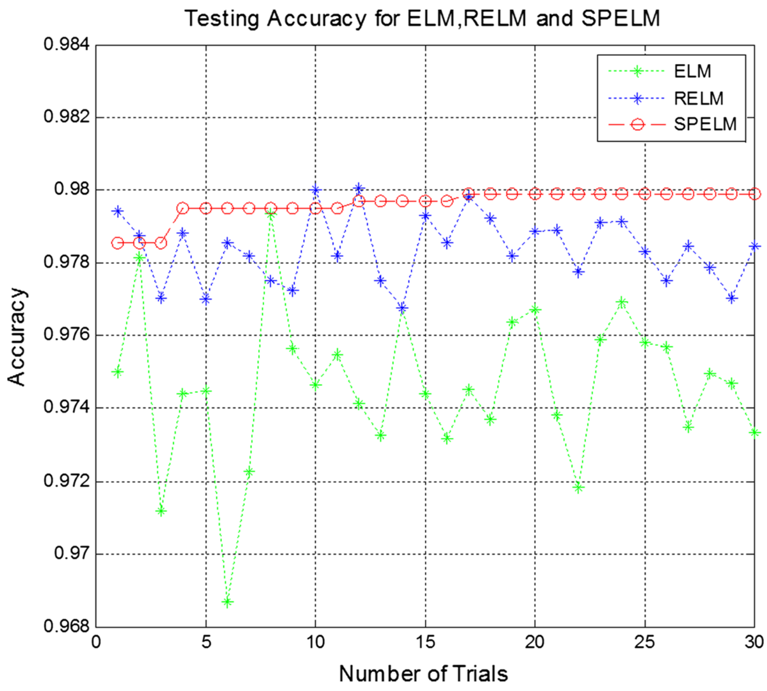


Fig. 11 Recognition accuracy of 20 % training dataset without feature dimensionality reduction

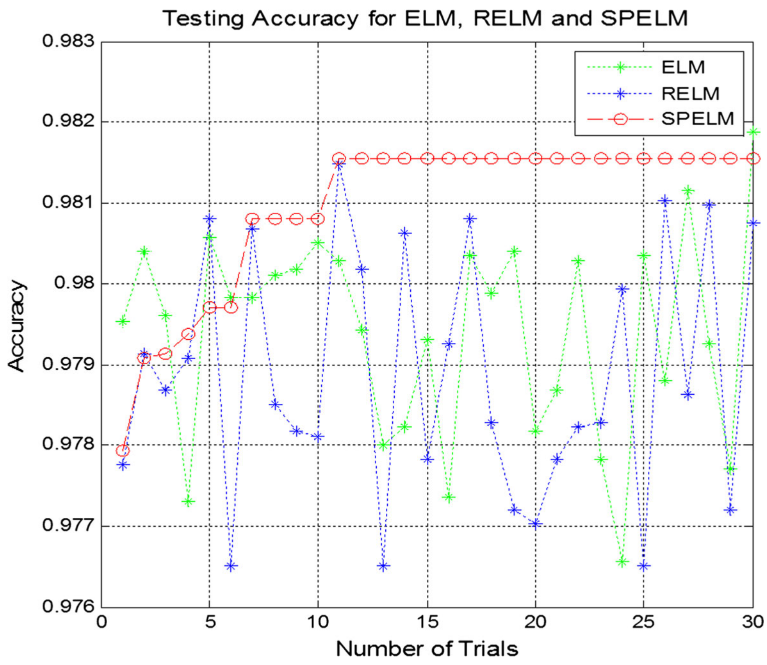


Fig. 12 Recognition accuracy of 30 % training dataset without feature dimensionality reduction

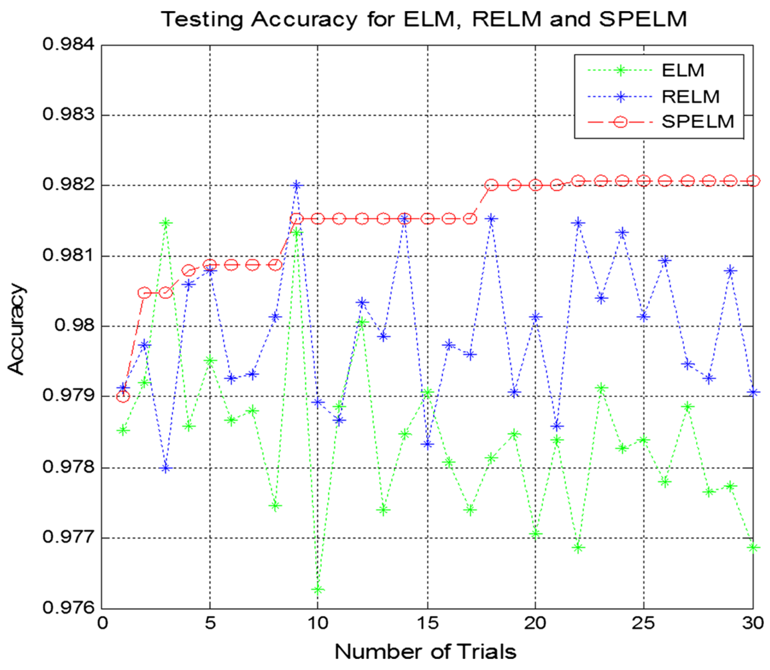


Fig. 13 Recognition accuracy of 40 % training dataset without feature dimensionality reduction

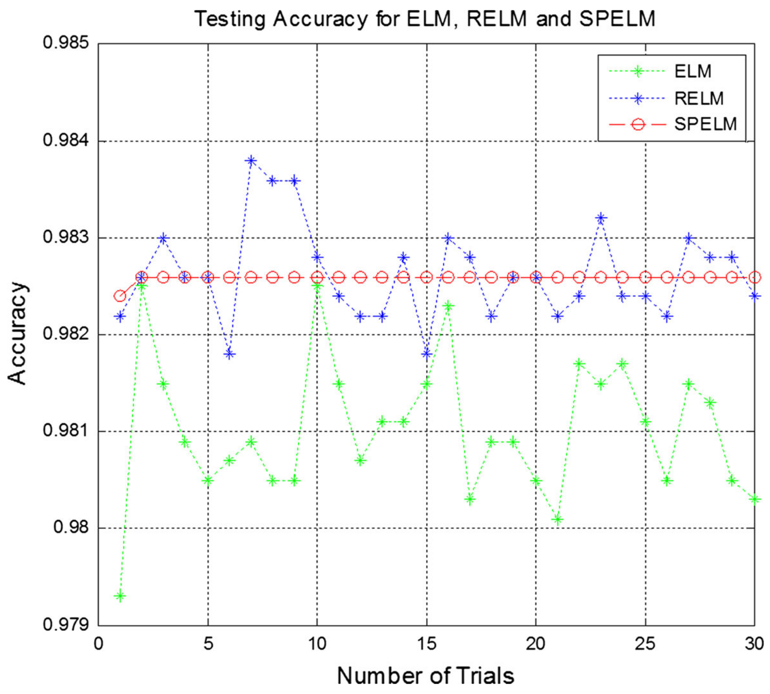


Fig. 14 Recognition accuracy for 20 % training set with 9 PCs

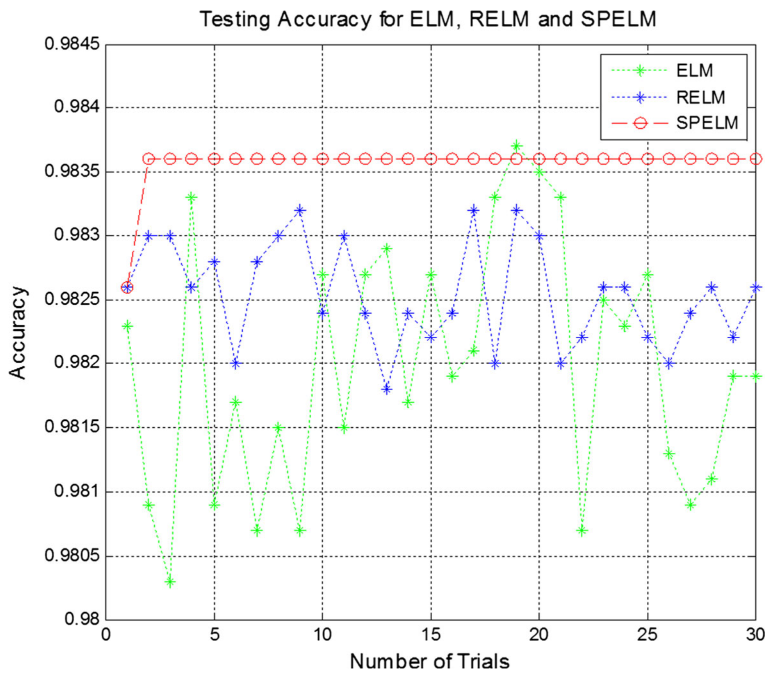


Fig. 15 Recognition accuracy of 30 % training set with 9 PCs

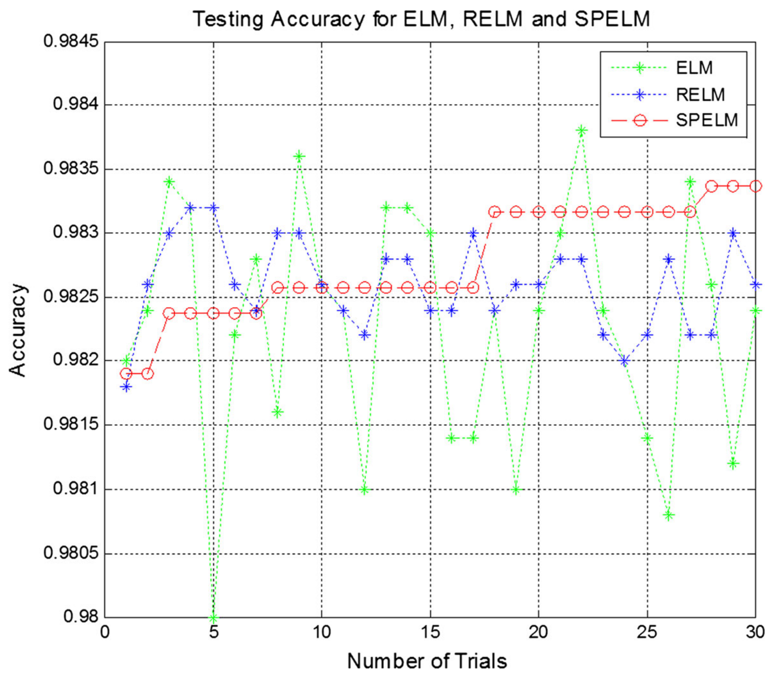


Fig. 16 Recognition accuracy of 40 % training set with 9 PCs

Table 7 Testing accuracy comparison with ELM and RELM without feature dimensionality reduction

# of training samples in %	ELM (%)	RELM (%)	Proposed SPELM (%)
20	97.41	97.78	97.96
30	97.75	97.87	98.10
40	97.84	97.99	98.15

Table 8 Accuracy comparison of SPELM with ELM and RELM using 9 PCs

# of training samples in %	ELM (%)	RELM (%)	Proposed SPELM (%)
20	98.10	98.26	98.26
30	98.20	98.25	98.36
40	98.20	98.26	98.28

5 Monotonically Increasing Learning in SPELM

Based on the state persevering property of SPELM as shown in previous results and discussion, the recognition accuracy is monotonically elevating as the number of trial increases. This is because SPELM preserves the state variables and adaptively updates them when superior weights are obtained. It is also observed that although the number of hidden neurons are fixed in each trial, the overall performance of ELM and RELM show stochastic behavior. This is due to the randomly generated weights and biases in each state without considering state parameters. In contrast, SPELM yields monotonically increasing output accuracy with respect to trial. Therefore, this adaptive learning property of SPELM would significantly boost the learning characteristics of ELM or ELM variants for achieving a better classification accuracy.

6 Conclusion

In this paper, we presented a new approach for computing state variables in ELM, termed as SPELM. SPELM provides a monotonically increasing learning characteristics by preserving state variables in each training and testing trial. It is observed that SPELM not only improves the inherent characteristics of the ELM based classification algorithms but also resolves the stochastic behavior of ELMs in training and testing phases. Experiments on face recognition, pedestrian detection, and network intrusion detection show the effectiveness and efficiency of SPELM.

References

1. Ahonen T, Hadid A, Pietikainen M (2006) Face description with local binary patterns: application to face recognition. IEEE Trans Pattern Anal Mach Intell 28(12):2037–2041
2. Bai Z, Huang GB, Wang D, Wang H, Westover MB (2014) Sparse extreme learning machine for classification. IEEE Trans Cybern 44(10):1858–1870
3. Barros ALB, Barreto GA (2013) Building a robust extreme learning machine for classification in the presence of outliers. In: Hybrid artificial intelligent systems. Springer, New York, pp 588–597

4. Belhumeur PN, Hespanha JP, Kriegman DJ (1997) Eigenfaces vs. fisherfaces: recognition using class specific linear projection. IEEE Trans Pattern Anal Mach Intell 19(7):711–720
5. Bontupalli V, Hasan R, Taha TM (2014) Power efficient architecture for network intrusion detection system. In: NAECON 2014-IEEE National aerospace and electronics conference, IEEE, pp 250–254
6. Bosch A, Zisserman A, Munoz X (2007) Representing shape with a spatial pyramid kernel. In: Proceedings of the 6th ACM international conference on Image and video retrieval, ACM, pp 401–408
7. Chen ZX, Zhu HY, Wang YG (2013) A modified extreme learning machine with sigmoidal activation functions. Neural Comput Appl 22(3–4):541–550
8. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. IEEE Computer Society Conference on computer vision and pattern recognition, IEEE vol 1, pp 886–893
9. Deng W, Zheng Q, Chen L (2009) Regularized extreme learning machine. In: IEEE symposium on computational intelligence and data mining, 2009. CIDM'09., IEEE, pp 389–395
10. Faraoun K, Boukelif A (2007) Neural networks learning improvement using the k-means clustering algorithm to detect network intrusions. World Acad Sci Eng Technol Int J Comput Electr Autom Control Inf Eng 1(10):3138–3145
11. Feng G, Huang GB, Lin Q, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20(8):1352–1357
12. Gavrila DM (1999) The visual analysis of human movement: a survey. Comput Vis Image Underst 73(1):82–98
13. Görnitz N, Kloft M, Rieck K, Brefeld U (2009) Active learning for network intrusion detection. In: Proceedings of the 2nd ACM workshop on security and artificial intelligence, ACM, pp 47–54
14. Hettich S, Bay SD (1999) The uci kdd archive. http://kdd.ics.uci.edu
15. Horata P, Chiewchanwattana S, Sunat K (2013) Robust extreme learning machine. Neurocomputing 102:31–44
16. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4(2):251–257
17. Huang GB (2003) Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans Neural Netw 14(2):274–281
18. Huang GB (2014) An insight into extreme learning machines: random neurons, random features and kernels. Cogn Comput 6(3):376–390
19. Huang GB (2015) What are extreme learning machines? Filling the gap between frank Rosenblatts dream and John von Neumanns puzzle. Cogn Comput 7(3):263–278
20. Huang GB, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70(16):3056–3062
21. Huang GB, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71(16):3460–3468
22. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
23. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501
24. Huang GB, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2(2):107–122
25. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B: Cybern 42(2):513–529
26. Huynh HT, Won Y (2008) Weighted least squares scheme for reducing effects of outliers in regression based on extreme learning machine. JDCTA 2(3):40–46
27. Jolliffe I (2002) Principal component analysis. Wiley Online Library
28. Lan Y, Soh YC, Huang GB (2010) Two-stage extreme learning machine for regression. Neurocomputing 73(16):3028–3038
29. Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. IEEE Computer Society Conference on computer vision and pattern recognition, IEEE vol 2, pp 2169–2178
30. Lee TS (1996) Image representation using 2D gabor wavelets. IEEE Trans Pattern Anal Mach Intell 18(10):959–971
31. Leshno M, Lin VY, Pinkus A, Schocken S (1993) Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw 6(6):861–867
32. Liu X, Chen T, Kumar BV (2003) Face authentication for multiple subjects using eigenflow. Pattern Recogn 36(2):313–328

33. Lu HJ, An CL, Ma XP, Zheng EH, Yang XB, Zhao CL, Li J, Zhang S, Zhang Z, Jin S et al (2013) Disagreement measure based ensemble of extreme learning machine for gene expression data classification. Jisuanji Xuebao (Chin J Comput) 36(2):341–348
34. Man Z, Lee K, Wang D, Cao Z, Khoo S (2012) Robust single-hidden layer feedforward network-based pattern classifier. IEEE Trans Neural Netw Learn Syst 23(12):1974–1986
35. MartıNez-MartıNez JM, Escandell-Montero P, Soria-Olivas E, MartıN-Guerrero JD, Magdalena-Benedito R, GóMez-Sanchis J (2011) Regularized extreme learning machine for regression problems. Neurocomputing 74(17):3716–3721
36. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) Op-elm: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21(1):158–162
37. Munder S, Gavrila DM (2006) An experimental study on pedestrian classification. IEEE Trans Pattern Anal Mach Intell 28(11):1863–1868
38. Ojala T, Pietikäinen M, Mäenpää T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans Pattern Anal Mach Intell 24(7):971–987
39. Rumelhart DE, Hinton GE, Williams RJ (1988) Learning representations by back-propagating errors. Cogn Model 5(3):1
40. Salama MA, Eid HF, Ramadan RA, Darwish A, Hassanien AE (2011) Hybrid intelligent intrusion detection scheme. In: Soft computing in industrial applications. Springer, pp 293–303
41. Samaria FS, Harter AC (1994) Parameterisation of a stochastic model for human face identification. In: Proceedings of the second IEEE workshop on applications of computer vision, IEEE, pp 138–142
42. Shen L, Bai L (2006) A review on gabor wavelets for face recognition. Pattern Anal Appl 9(2–3):273–292
43. Wang Y, Cao F, Yuan Y (2011a) A study on effectiveness of extreme learning machine. Neurocomputing 74(16):2483–2490
44. Wang Y, Cao F, Yuan Y (2011b) A study on effectiveness of extreme learning machine. Neurocomputing 74(16):2483–2490
45. Yu Q, Miche Y, Eirola E, Van Heeswijk M, Severin E, Lendasse A (2013) Regularized extreme learning machine for regression with missing data. Neurocomputing 102:45–51
46. Yuan Y, Wang Y, Cao F (2011) Optimization approximation solution for regression problem based on extreme learning machine. Neurocomputing 74(16):2475–2482
47. Zahangir Alom M, Sidike P, Asari VK, Taha TM (2015) State preserving extreme learning machine for face recognition. In: International joint conference on neural networks (IJCNN), IEEE, pp 1–7
48. Zhao G, Shen Z, Man Z (2011) Robust input weight selection for well-conditioned extreme learning machine. Int J Inf Technol 17(1):1–13
49. Zhao W, Chellappa R, Phillips PJ, Rosenfeld A (2003) Face recognition: a literature survey. ACM Comput Surv (CSUR) 35(4):399–458