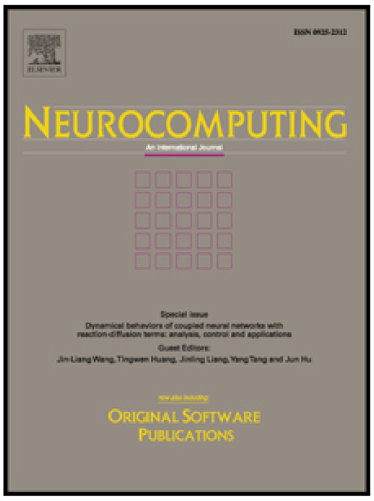

Sparse Pseudoinverse Incremental Extreme Learning Machine

Peyman Hosseinzadeh Kassani, Andrew Beng Jin Teoh, Euntai Kim

PII: S0925-2312(18)30128-0
DOI: 10.1016/j.neucom.2018.01.087
Reference: NEUCOM 19289

To appear in: *Neurocomputing*

Received date: 13 June 2017
Revised date: 18 January 2018
Accepted date: 31 January 2018

Highlights

- Proposed a new sparse extreme learning machine based on the condition number.
- Proposed a fast iterative matrix decomposition algorithm.
- Found a lemma to relate the condition number and number of hidden neurons.
- Proposed model is compact and fast; suitable for practical applications.
- Verified the model reliability using experiments on several data sets.

Sparse Pseudoinverse Incremental Extreme Learning Machine

Peyman Hosseinzadeh Kassani[a], Andrew Beng Jin Teoh[a], Euntai Kim[a,*]

[a]*Electrical and Electronic Engineering Department, Yonsei University, Seoul, South Korea*

Abstract

An extreme learning machine (ELM) is a popular analytic single hidden layer feedforward neural network because of its rapid learning capacity. However, vanilla dense ELMs are affected by the overfitting problem when the number of hidden neurons is high. Further direct consequences of the density are decreases in both the training and prediction speeds. In this study, we propose an incremental method for sparsifying the ELM using a newly devised indicator driven by the condition number in the ELM design matrix, in which we call sparse pseudoinverse incremental-ELM (SPI-ELM). SPI-ELM exhibits better generalization performance and lower run-time complexity compared with ELM. However, the sparsification process may negatively affect the learning speed of SPI-ELM; thus, we introduce an iterative matrix decomposition algorithm to address this issue. We also demonstrate that there is a useful relationship between the condition number in the ELM design matrix and the number of hidden neurons. This relationship helps to understand the random weights and nonlinear activation functions in ELMs. We evaluated the SPI-ELM method based on 20 benchmark data sets from the University of California Irvine repository and three real-world databases from the computer vision domain.

Keywords: Extreme learning machine, Hidden neuron, Least squares estimation, Matrix decomposition, Sparsity

*Corresponding author
Email address: `etkim@yonsei.ac.kr` (Euntai Kim)

1. Introduction

Extreme learning machine (ELM) is a special type of single hidden layer feedforward network that provides an analytic solution for learning [1, 2, 3, 4, 5]. Huang et al. [6] proved that hidden neurons in single hidden layer feedforward networks can be randomly generated and iterative learning like in backpropagation algorithm can be avoided, thereby significantly reducing the training time.

However, ELMs have two major drawbacks [7]: (1) in ELMs, learning employs the pseudoinverse of the design matrix, $\mathbf{H}^{\dagger}$, to search for the output weights, which is essentially based on the least squares error (LSE) minimization principle, and thus, this method may be vulnerable to overfitting owing to an ill-conditioned $\mathbf{H}$ [8]; and (2) the run-time (prediction) performance of ELM is highly dependent on the number of random hidden neurons. For real-world problems, a huge number of random hidden neurons may be required to achieve satisfactory performance [6, 9, 10]. However, the use of a vast number of hidden neurons makes ELMs cumbersome and the predictions obtained may be inefficient. Moreover, a large network is more likely to overlearn due to the high complexity of the model employed. Thus, an adequate number of random hidden neurons is preferable for learning various prediction tasks with better performance, where the model should not be overly complex because this could make the generalization performance poor and the predictions would be obtained slowly[6].

Section 2 describes research related to the problem outlined above. In Section 3, we present preliminary details of the ELM method. In Section 4, we explain our proposed SPI-ELM model. In Section 5, we present a comprehensive evaluation of the SPI-ELM method. Finally, we give our conclusions in Section 6.

2. Related Work

In this section, we discuss previous studies of the two problems highlighted above. These studies can be categorized according to four types, i.e., incremental, decomposition, sparse, and collinearity based methods.

Several incremental-based methods have been proposed. In contrast to ELM, error-minimized ELM (EM-ELM) introduces the hidden neurons either individually or chunk by chunk until a predefined error is satisfied; thus, an excessive number of random neurons are not included in the network [11]. However, EM-ELM does not outperform ELM in terms of accuracy, although EM-ELM learns faster. The most recent example of an incremental-based model is QR factorization-based incremental ELM [12], which is also capable of adding random neurons incrementally and its computational complexity is significantly lower than that of ELM, but without any reduction in accuracy.

The decomposition-based fast ELM (DF-ELM) [13] decomposes the design matrix **H** into two parts and applies the block matrix inversion lemma to reduce the computational load, although with a massive number of hidden neurons. DF-ELM has a shorter training time but still maintains similar accuracy to ELM. Another decomposition-based model is stacked ELMs (S-ELMs), which splits a single large ELM network into multiple stacked small ELMs. The S-ELMs can approximate a large ELM network with significantly lower memory requirements [14].

The well-established sparsity notion in machine learning, which is often yield compact with good generalizability models, has also been utilized successfully in several ELM variants [1, 7, 15, 16]. Luo et al. [7] proposed a sparse Bayesian ELM (SB-ELM), which was inspired by the relevance vector machine (RVM) [17], and showed that SB-ELM can achieve good performance while remains compact. The key strength of SB-ELM is that there is no need to supply sparsity degree because it is determined automatically by a mechanism called automatic relevance determination [18, 19]. Unfortunately, it inherits the disadvantage of RVM, where the SB-ELM may yield poor predictions if a test instance is distant from the relevance vectors [20, 21].

Another example of a sparse ELM was proposed based on quadratic programming [15], where the quadratic programming problem is split into a series of small subproblems that can be solved easily. This model reduces both the storage and time complexities. However, the learning algorithm is still based on iterative computation. Therefore, it is unavoidably slower than the ELM. In addition, Tikhonov regularized optimally pruned ELM (TROP-ELM) was reported, which utilizes a cascade of L1- and L2-type regularizations [4]. First, using L1-regularization, TROP-ELM ranks a group of the best neurons to minimize the training error. Next, the L2 penalty on the regression weights is applied to facilitate the efficient pruning of neurons. TROP-ELM was used as a benchmark for comparison with some sparse ELM variants in previous studies [7, 22].

Few methods have been proposed in the final category of collinearity-based methods but this type of method is closest to our proposed model. Zhao et al. [23] constructed an ELM where the condition number is forced to one to make the design matrix **H** well-conditioned. In particular, the random weights associated with the first hidden neuron are assigned first, before the subsequent neuron weights (random bases) are made mutually orthogonal to ensure that the matrix is well conditioned. However, the number of hidden neurons is limited to $d + 1$ where d is the number of input neurons, which can severely restrict the expressive power of ELMs. Another study [24] proposed a solution to the ill-posed inversion problem of **H** by using recursive partial least-squares (RPLS) to estimate the output weights, and an online adaptive algorithm was also introduced.

Recently, Principe and Chen [25] stated that the difficulty caused by the correlated bases in ELM, which manifest as the column vectors in **H**, may make **H** rank-deficient and this problem is fairly common. Correlated bases may lead to poor performance with both the training data and unseen data, i.e., poor generalizability. The level of correlation among random bases in ELMs can be measured using the max-min eigenvalue ratio of the design matrix. Thus, collinearity-based methods focus mainly on solving the ill-conditioned design matrix problem in ELMs, but they do no attempt to address the efficiency issues considered by the other three categories. Table 1 summarizes the advantages and limitations of the methods in each category.

Table 1. Summary of the advantages and limitations of the methods in each category

Method	Advantages	Disadvantages
Incremental-based [9, 11, 12, 26, 27, 28, 29, 30];	Relatively faster training and prediction time for ELMs. The number of random neurons can be set deterministically based on a predefined error.	No performance gain in terms of accuracy compared with ELMs.
Decomposition-based [13, 14, 28, 31, 32, 33, 34]	Relatively faster training time due to the efficient design matrix decomposition algorithm.	No attempt to control the network size; thus, the dense network remains and there is no gain in accuracy compared with ELMs.
Sparsity-based [1, 4, 7, 15, 16, 17, 35, 36, 37, 38, 39]	Better performance at generalization. Faster prediction time and low storage complexity.	The sparsification process may slow down the training speed.
Collinearity-based [23, 24, 25]	Generates distinct and less correlated hidden neurons to avoid the ill-posed inversion problem.	No attempt to control the network size; thus, the dense network remains.

2.1. Contributions

In the following, we propose a new ELM variant by considering the sparsity-, decomposition-, and collinearity-based approaches. Our main aim is to enhance both the training and prediction speeds, as well as the generalizability performance of ELMs by applying sparse pseudoinverse matrix decomposition in an incremental manner. Our proposed method called sparse pseudoinverse decomposition-based incremental ELM (SPI-ELM) exploits the advantages of the three main approaches as well as overcoming their limitations. To make an ELM sparse, we detect and prune the highly collinear (correlated) random neurons, which are often the main cause of the ill-conditioned LSE model [40] upon which the ELM is based. To identify the collinear random neurons, we evaluate the condition numbers in the design matrix in an ELM. Using our new indicator based on the condition number, a systematic method can be designed to prune the unwanted neurons and sparsify the network. The degree of sparseness can be adjusted by a parameter called the sparsity factor.

The main contributions of this study are summarized as follows.

1. We propose a novel incremental method for sparsifying an ELM by using an indicator based on the condition number in the ELM design matrix, which is useful for establishing a well-conditioned ELM. Another direct advantage of a sparse ELM is that its run-time complexity is significantly lower than that of vanilla ELM. This method is illustrated for SPI-ELM in the present study but all LSE minimization-based machine learning models could benefit from this approach.
2. We prove that adding hidden neurons monotonically to the ELM increases the condition number in the design matrix, where a high value indicates an ill-conditioned model. We note that the rate of growth in the condition number has a direct relationship with the range values for the input-hidden weights and the biases, as well as the types of nonlinear activation functions. This observation is useful for understanding ELMs.
3. Extensive experiments were conducted based on several University of California Irvine (UCI) data sets and three real-world databases to evaluate the proposed model in terms of its generalizability performance, and the training and testing time required.

3. Preliminary Details of ELM

The ELM was proposed by Huang et al. [6, 41] as a type of single hidden layer network without an iterative learning algorithm [42]. ELM was proved to be a universal approximator by Huang et al. [26]. In particular, let N be the number of training samples and $\mathbf{x}\in\mathbb{R}^{d\times 1}$ is a feature vector in the training set, $D=\{(\mathbf{x}_i, \mathbf{t}_i) \mid i=1,\ldots,N\}$, where $\mathbf{t}_i\in\mathbb{R}^{m\times 1}$ is a target vector with m classes. ELM comprises L hidden neurons in a parameterized nonlinear activation function, $G(\mathbf{a},b,\mathbf{x})$**,** where $\mathbf{a},b$ are the parameters of the activation function. The output of ELM can be written as $f_L(\mathbf{x}_j) = \sum_{i=1}^{L} G(\mathbf{a_i},b_\mathbf{i},\mathbf{x}_j)$. A linear equation system, $f_L(\mathbf{x}_j) = \mathbf{t}_j$, can be established as follows:

$$\sum_{i=1}^{L}\beta_i G(\mathbf{a_i},b_{\mathbf{i}},\mathbf{x}_j)=\mathbf{t}_j\ ,\qquad j=1,\ldots,N \tag{1}$$

where $\mathbf{a_i},b_{\mathbf{i}}$ are the uniformly distributed random weights and bias, respectively, which connect the input layer to the hidden layer, and β_i is the output weight from the hidden layer to the output layer for hidden neuron i. Two commonly used types for $G()$ are the sigmoid function,

$$G(\mathbf{a},b,\mathbf{x})=\frac{1}{1+e^{-(\mathbf{a}.\mathbf{x}+b)}} \tag{2}$$

and the radial basis function (RBF),

$$G(\mathbf{a},b,\mathbf{x})=e^{\frac{-\|(\mathbf{x}-\mathbf{a}\|^2}{b}} \tag{3}$$

Eq. 1 can be rewritten in matrix form as

$$\mathbf{HB}=\mathbf{T} \tag{4}$$

where $\mathbf{H}=[G(\mathbf{a}_i,b_i,\mathbf{x}_j)]\in\mathbb{R}^{N\times L},\ \ i=1,\ldots,N,\ j=1,\ldots,L$ is called the *design matrix*. Note that the column vectors in **H,** $\{\mathbf{h}_k\in\mathbb{R}^N|k\text{=}1,\ldots,L\}$ are called *bases* because the projected feature space induced by $G(\mathbf{a_i},b_{\mathbf{i}},\mathbf{x}_j)$ is spanned by these vectors. $\mathbf{T}\in\mathbb{R}^{N\times m}$ is the target matrix and $\mathbf{B}\in\mathbb{R}^{L\times m}$ is the set of output weights, β_i, which can be solved in a least-squares sense as follows:

$$\mathbf{B}=\mathbf{H}^{\dagger}\mathbf{T}=\left(\mathbf{H^TH}\right)^{-1}\mathbf{H}^T\mathbf{T} \tag{5}$$

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

The main training steps in the ELM algorithm are described in Algorithm 1.

```
Algorithm 1. ELM
Input:
1) Training data $D=(\mathbf{x}_i, \mathbf{t}_i)$, $i=1,\ldots,N$, the maximum number of hidden
   neurons $L_{max}$ and the target $\mathbf{T}$:
2) Randomly generate the input weights $\mathbf{a}$ and bias $b$.
Procedure:
3) Compute the design matrix $\mathbf{H}$ and form a linear equation system as in
   Eq. 4.
Output:
4) Given $\mathbf{T}$ and $\mathbf{H}$, compute the output weight matrix $\mathbf{B}$ as in Eq.5
```

Given $\widehat{\mathbf{B}}$, the prediction output $\widehat{\mathbf{T}}$ is computed as $\widehat{\mathbf{T}}= H\widehat{\mathbf{B}}$.

4. Sparse Pseudoinverse Incremental-based ELM (SPI-ELM)

145 ### *4.1. Overview*

The proposed SPI-ELM comprises three distinct sequential stages, as shown in Fig. 1. The first stage computes the condition numbers for the current $\mathbf{H}$ [43],

$$Cond(\mathbf{H}) = \sqrt{\frac{\lambda_{max}(\mathbf{H}^{\mathbf{T}}\mathbf{H})}{\lambda_{min}(\mathbf{H}^{\mathbf{T}}\mathbf{H})}} \tag{6}$$

after adding a hidden neuron, where $\lambda_{max}(\bullet)$ and $\lambda_{min}(\bullet)$ are the maximum and minimum eigenvalues of the given matrix, respectively. This involves a repetitive computation process on Cond($\mathbf{H}$) as the network grows incrementally. The second stage involves network sparsification by pruning the collinear random bases, which can be identified via Cond($\mathbf{H}$) (Section 4.3). The last stage involves computing the pseudoinverse of **H,** i.e., $\mathbf{H}^{\dagger}$, using an iterative matrix decomposition approach, as described in Section 4.4.

155 ### *4.2. Computing the Condition Number*

The sparsification mechanism proposed for the ELM is driven by the collinearity and condition number. Hence, we first give a brief explanation of the relationship between collinearity and the condition number in the following section.

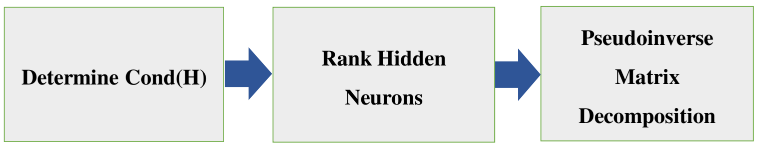

Figure 1: Flowchart illustrating the proposed SPI-ELM method

4.2.1. Collinearity and Condition Number

160 It is well known that solving a least squares-based optimization problem via $\mathbf{B}=\mathbf{H}^{\dagger}\mathbf{T}$ can be an ill-conditioned problem subject to certain conditions [40]. It has been shown that **H** cannot be of full column rank if there is collinearity among the bases [44]. However, even with full column rank **H**, a few bases might be highly collinear [45]. ELM is a typical LSE-based learner so it suffers from the same problem without exceptions 165 [40]. Indeed, collinearity in **H** is inevitable for ELM because the bases generated by the random parameterized $G(\mathbf{a_i}, b_\mathbf{i}, \mathbf{x}_j)$ are not orthogonal. These bases can be made orthogonal artificially [23], but this is limited to $d+1$ bases, where d is the number of input neurons. If more than $d+1$ is used, then collinearity among the bases is unavoidable.

170 The severity of the collinearity in a matrix can be determined from its eigenvalues and condition numbers, as shown in Tables 2 and 3 [46].

Table 2. Interpreting collinearity based on eigenvalues

Degree of Collinearity	**Form of Matrix**	**Magnitude of Eigenvalues**
No collinearity	Nonsingular	Not equal to zero
Near perfect collinearity	Near singular	Close to zero
Perfect collinearity	Singular (not positive definite)	Equal to zero

Table 3. Interpreting collinearity based on condition numbers (CN)

Condition Number	**Degree of Collinearity**
If CN < 100	Weak
If $100 <$ CN < 1000	Moderate
If CN > 1000	Severe

It is conceivable that the ill-conditioned problem in ELM can be analyzed based on the collinearity in the design matrix $\mathbf{H}$. In *Lemma 1*, we show that adding only one hidden neuron to the ELM increases the condition number.

Lemma 1. Let $\mathbf{H}_{k-1} = [G(\mathbf{a_i}, b_\mathbf{i}, \mathbf{x}_j)]$, $i = 1, \ldots, N$, $j = 1, \ldots, L$ be the positive definite design matrix of the ELM with L hidden neurons. When a hidden neuron is added, $\mathbf{H}_{k-1}$ is updated to $\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{k-1} & \mathbf{h}_k \end{bmatrix}. = [G(\mathbf{a_i}, b_\mathbf{i}, \mathbf{x}_j)], (i = 1, \ldots, N,\ j = 1, \ldots, L+1)$. Let $Cond(\mathbf{H}_{k-1})$ be the condition number of $\mathbf{H}_{k-1}$ with spectrum $\{\lambda_i | i = 1, \ldots L\}$ with L hidden neurons and $Cond(\mathbf{H}_k)$ be the condition number of $\mathbf{H}_k$ with spectrum $\{\beta_i | i = 1, \ldots L+1\}$ with $L+1$ hidden neurons wherein λ_i and β_i are the eigenvalues of $\mathbf{H}_{k-1}^T \mathbf{H}_{k-1}$ and $\mathbf{H}_k^\mathbf{T} \mathbf{H}_k$ respectively. Hence [43],

$$Cond(\mathbf{H}_k) = \sqrt{\frac{\beta_{max}\left(\mathbf{H}_k^T \mathbf{H_k}\right)}{\beta_{min}\left(\mathbf{H}_k^T \mathbf{H_k}\right)}} \geqslant Cond(\mathbf{H}_{k-1}) = \sqrt{\frac{\lambda_{max}\left(\mathbf{H}_{k-1}^T \mathbf{H_{k-1}}\right)}{\lambda_{min}\left(\mathbf{H}_{k-1}^T \mathbf{H_{k-1}}\right)}} \tag{7}$$

Proof.

Knowing that $\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{k-1} & \mathbf{h}_k \end{bmatrix}$, where $\mathbf{H}_{k-1} \in \mathbb{R}^{N \times L}$ and $\mathbf{h}_k \in \mathbb{R}^{N \times 1}$, we can write $\mathbf{H}_k^\mathbf{T} \mathbf{H}_k$ as,

$$\mathbf{H}_k^\mathbf{T} \mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{k-1}^\mathbf{T} \mathbf{H}_{k-1} & \mathbf{c} \\ \mathbf{c}^T & m_{L+1} \end{bmatrix} \tag{8}$$

where $\mathbf{c} = \mathbf{H}_{k-1}^\mathbf{T} \mathbf{h}_k \in \mathbb{R}^{L \times 1}$ and,

$$m_{L+1} = \mathbf{h}_k^\mathbf{T} \mathbf{h}_k \tag{9}$$

Let say that the matrix $\mathbf{H}_{k-1}^\mathbf{T} \mathbf{H}_{k-1}$ has ascending sorted eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_L$ and the ascending sorted eigenvalues for $\mathbf{H}_k^\mathbf{T} \mathbf{H}_k$ is $\beta_1 \geqslant \beta_2 \geqslant \cdots > \beta_L \geqslant \beta_{L+1}$. And, all eigenvalues are real and positive because $\mathbf{H}_{k-1}^\mathbf{T} \mathbf{H}_{k-1}$ and $\mathbf{H}_k^\mathbf{T} \mathbf{H}_k$ are positive definite symmetric matrices. Using *Courant-Fischer Theorem,* the eigenvalues of $\mathbf{H}_{k-1}^{\mathbf{T}}\mathbf{H}_{k-1}$ interlaced with eigenvalues of $\mathbf{H}_k^{\mathbf{T}}\mathbf{H}_k$ in this way (The example 7.5.3 of [47]):

$$\beta_1 \geq \lambda_1 \geq \beta_2 \geq \lambda_2 \geq \ldots \geq \beta_L \geq \lambda_L \geq \beta_{L+1} \tag{10}$$

This implies that the largest eigenvalue for $\mathbf{H}_k^{\mathbf{T}}\mathbf{H}_k$ is equal or larger than the largest eigenvalue for $\mathbf{H}_{k-1}^{\mathbf{T}}\mathbf{H}_{k-1}$ (i.e. $\beta_1 \geq \lambda_1$). Similarly, the smallest eigenvalue for $\mathbf{H}_k^{\mathbf{T}}\mathbf{H}_k$ is equal or smaller than the smallest eigenvalue for $\mathbf{H}_{k-1}^{\mathbf{T}}\mathbf{H}_{k-1}$ (i.e. $\lambda_L \geq \beta_{L+1}$). Hence, $Cond\left(\mathbf{H}_k\right) \geqslant Cond\left(\mathbf{H}_{k-1}\right)$ and the proof is met.

We further are interested to see the rate of change in the condition number when a new hidden neuron is added to the network. Note that the trace of a matrix is sum of its eigenvalues. Hence, relying on Eq. 8,

$$Trace\left(\mathbf{H}_k^{\mathbf{T}}\mathbf{H}_k\right) = Trace\left(\mathbf{H}_{k-1}^{\mathbf{T}}\mathbf{H}_{k-1}\right) + m_{L+1} \tag{11}$$

Where *Trace* shows the trace of a matrix. As the trace of a matrix is sum of its eigenvalues, we can write,

$$\beta_1 + \beta_2 + \cdots + \beta_L + \beta_{L+1} = \lambda_1 + \lambda_2 + \cdots + \lambda_L + m_{L+1} \tag{12}$$

As it can be seen, every time a new hidden neuron is added to the network, the positive term $m_{L+1}=\mathbf{h}_k^{\mathbf{T}}\mathbf{h}_k$ in Eq. 9, is added to the sum of eigenvalues and this is the reason why the change in condition number depends on the term $m_{L+1}=\mathbf{h}_k^{\mathbf{T}}\mathbf{h}_k$. Another evidence is for Eq. 12 which can be written as follows:

$$\beta_1 + \cdots + \beta_L + \beta_{L+1} = (\lambda_1 + M_1) + \cdots + (\lambda_L + M_L) + M_{L+1} \tag{13}$$

where $M_1 + \cdots + M_{L+1} = m_{L+1}$ and $M_i \geqslant 0,\ i = 1, 2, \ldots, L+1$. This can be easily seen via Eq. 10. So, every eigenvalue β_i of $Cond\left(\mathbf{H}_k\right)$ is equal to the eigenvalue λ_i of $Cond\left(\mathbf{H}_{k-1}\right)$ plus a positive term M_i , $i = 1,\ 2, \ldots, L+1$. This means the changes in new condition number depends on the term m_{L+1}. Since $m_{L+1}=\mathbf{h}_k^{\mathbf{T}}\mathbf{h}_k$, *Lemma 1* suggests that the random weights $\{\mathbf{a},\ b\}$ as well as the nonlinear activation functions G in the newly introduced basis $\mathbf{h}_k=G(\mathbf{a}, b, \mathbf{x})$ directly affect the change in the condition number. Since there is no control over $\{\mathbf{a},\ b\}$, the new condition number will be unavoidably higher than the previous condition number, thereby making the ELM ill-conditioned. This assertion is indeed consistent with the observations of Principe and Chen [25], who explored the correlation issue for bases in ELMs.

Figure 2 illustrates the difference of two consecutive condition numbers, e.g. $Cond(\mathbf{H}_k) - Cond(\mathbf{H}_{k-1})$ with respect to the network size for RBF and sigmoid functions using the Credit Bank data set with 620 training samples [48]. Figure 3 shows the training and testing accuracies with respect to the network size. For each specific network size, we ran the simulation five times and the average result was recorded. The values of $\{\mathbf{a}, b\}$ were drawn from a uniform distribution [–1 1]. As shown in Fig. 2, with the same random weights, the sigmoid function contributed more to the condition number than the RBF function.

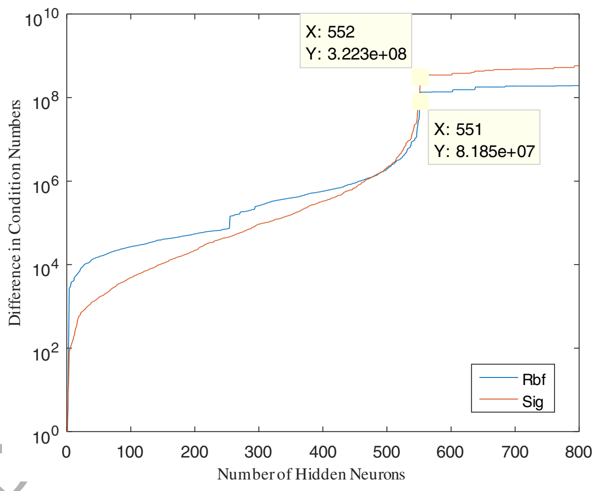

Figure 2: Monotonic difference in the condition number (log scale) with respect to the network size.

We analyzed this behavior as follows. First, for networks with 1–50 hidden neurons with condition numbers less than 1000, the training and testing accuracies both improved monotonically, as shown in Fig. 3, thereby supporting our assertion that small and medium-sized ELMs are well-conditioned, where this was confirmed by the behavior of the accuracy curves. Second, for networks with 51 to 552 hidden neurons, the condition number increased abruptly with both sigmoid- and RBF-based ELM. This suggests that the networks were highly likely to be ill-conditioned, and this was

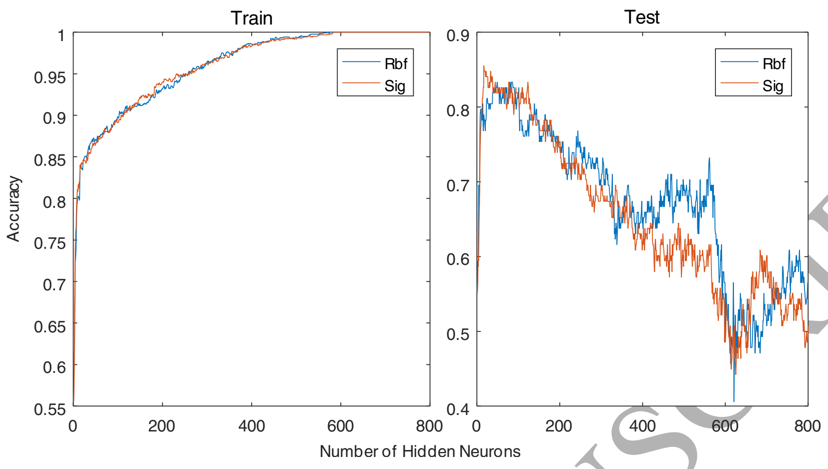

Figure 3: Changes in the testing and training accuracies with respect to the network size.

supported by the continual decline in the test accuracies shown in Fig. 3.

235 This interesting observation can be justified from a geometric perspective. Figure 4 shows the randomly projected feature space characterized by the basis vectors, $\mathbf{h}_i$.

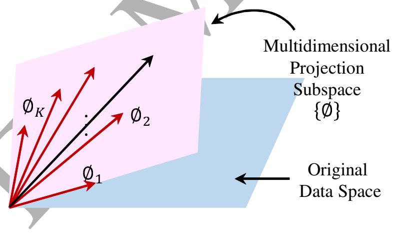

Figure 4: Visualization of the projected feature space

The correlations between the basis vectors can be measured according to the angles of the pairwise basis vectors [49], where the correlation is higher when the angle is more acute. As the number of hidden neurons in the ELM increases, more basis 240 vectors are introduced into the feature space and the angles between the basis vectors become even smaller. The total correlation between the basis vectors will become larger, thereby negatively affecting the discriminative capacity and thus the accuracy of the ELM.

Another interesting observation is that when networks become very large, and thus ill-conditioned due to the extremely high condition number, the RBF function exhibits better generalizability performance than the sigmoid activation function, as depicted in Fig. 2.

We may summarize these findings as follows.

1. Networks with a small condition number can learn better than those with a large condition number.
2. The sigmoid activation function is more sensitive to the rate of change in the condition number than the RBF function; thus, it is likely to have better test accuracy with fewer hidden neurons.

4.3. Network Sparsification

To alleviate the ill-conditioned problem, the problematic hidden neurons that lead to a high condition number should be detected and pruned by using the difference in the condition numbers vector $\mathbf{v}$ established in the previous section. Thus, the values of $\mathbf{v}$ are sorted in *ascending* order. The first $S_p\%$ among the sorted $\mathbf{v}$ values is chosen and the rest are discarded. According to the selected hidden neurons, the corresponding column vectors of $\mathbf{H}$ are selected to obtain $\mathbf{H}_{Sp}$.

4.4. Iterative Matrix Decomposition Stage

Given $\mathbf{H}_{Sp}$, the SPI-ELM proceeds to solve $\mathbf{H}^{\dagger}_{Sp}$ and eventually we have $\mathbf{B}_{Sp}{=}\mathbf{H}^{\dagger}_{Sp}T$. To avoid notational clutter, the subscript *sp* for $\mathbf{H}_{sp}$ is omitted in the following.

Inspired by a similar procedure outlined in previous studies [28, 50], we propose our rapid iterative matrix decomposition method. Let $\mathbf{H}_k{\in}\mathbb{R}^{N\times L}$ and $\mathbf{H}^{\dagger}_k{\in}\mathbb{R}^{L\times N}$ be the design matrix and its pseudoinverse counterpart in iteration k, respectively, where N and L are the training data size and the number of hidden neurons. We also define the partitioned forms of $\mathbf{H}_k$ and $\mathbf{H}^{\dagger}_k$ as follows:

$$\mathbf{H}_k{=}\begin{bmatrix} \mathbf{H}_{k-1}{\in}\mathbb{R}^{N\times(L-1)} & \mathbf{h}_k{\in}\mathbb{R}^{N\times 1} \end{bmatrix} \quad , \quad \mathbf{H}^{\dagger}_k{=}\begin{bmatrix} \mathbf{G}_k{\in}\mathbb{R}^{(L-1)\times N} \\ \mathbf{g}_k{\in}\mathbb{R}^{1\times N} \end{bmatrix} \tag{14}$$

$\mathbf{H}_k$ and $\mathbf{H}_k^{\dagger}$ are multiplied to obtain

$$\mathbf{H}_k\mathbf{H}_k^{\dagger}=\mathbf{H}_{k-1}\mathbf{G}_k+\mathbf{h}_k\mathbf{g}_k \tag{15}$$

If Eq. 15 is further multiplied from the left with $\mathbf{H}_{k-1}^{\dagger}$, we have

$$\mathbf{H}_{k-1}^{\dagger}\mathbf{H}_k\mathbf{H}_k^{\dagger}=\mathbf{H}_{k-1}^{\dagger}\mathbf{H}_{k-1}\mathbf{G}_k+\mathbf{H}_{k-1}^{\dagger}\mathbf{h}_k\mathbf{g}_k \tag{16}$$

For the right side of the above equation, $\mathbf{H}_{k-1}^{\dagger}\mathbf{H}_k\mathbf{H}_k^{\dagger}$, as $\mathbf{H}_k\mathbf{H}_k^{\dagger}$ is symmetric, and when is used as a left multiplier, any matrix with columns in the column space of $\mathbf{H}_k$ is leaved unchanged [50]. Hence, if $\mathbf{H}_k\mathbf{H}_k^{\dagger}$ is used as a right multiplier, it leaves unchanged any matrix with rows in the transposed column space of $\mathbf{H}_k$. The matrix $\mathbf{H}_{k-1}^{\dagger}$ in $\mathbf{H}_{k-1}^{\dagger}\mathbf{H}_k\mathbf{H}_k^{\dagger}$ has rows in the transposed column space of $\mathbf{H}_{k-1}$, thus $\mathbf{H}_{k-1}^{\dagger}\mathbf{H}_k\mathbf{H}_k^{\dagger}=\mathbf{H}_{k-1}^{\dagger}$. By the same reasoning, $\mathbf{H}_{k-1}^{\dagger}\mathbf{H}_{k-1}\mathbf{G}_k=\mathbf{G}_k$ [50]. Thus, eq. 16 can be simplified to

$$\mathbf{H}_{k-1}^{\dagger}=\mathbf{G}_k+\mathbf{H}_{k-1}^{\dagger}\mathbf{h}_k\mathbf{g}_k \tag{17}$$

and rewritten as

$$\mathbf{G}_k=\mathbf{H}_{k-1}^{\dagger}-\mathbf{d}_k\mathbf{g}_k \tag{18}$$

where

$$\mathbf{d}_k=\mathbf{H}_{k-1}^{\dagger}\mathbf{h}_k \tag{19}$$

and thus $\mathbf{H}_k^{\dagger}$ in Eq. 14 can be reformulated as

$$\mathbf{H}_k^{\dagger}=\begin{bmatrix}\mathbf{H}_{k-1}^{\dagger}-\mathbf{d}_k\mathbf{g}_k\\ \mathbf{g}_k\end{bmatrix} \tag{20}$$

Hence, to compute $\mathbf{H}_k^{\dagger}$, we only need to find $\mathbf{g}_k$. In addition, multiplying $\mathbf{H}_k$ in Eq. 14 by $\mathbf{H}_\mathbf{k}^{\dagger}$ in Eq. 20 gives

$$\mathbf{H}_k\mathbf{H}_k^{\dagger}=\mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger}-\mathbf{H}_{k-1}\mathbf{d}_k\mathbf{g}_k+\mathbf{h}_k\mathbf{g}_k=\mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger}+\mathbf{c}_k\mathbf{g}_k \tag{21}$$

where

$$\mathbf{c}_k=\mathbf{h}_k-\mathbf{H}_{k-1}\mathbf{d}_k \tag{22}$$

and multiplying Eq. 22 from the left by $\mathbf{H}_{k-1}^{\dagger}$ while considering Eq. 19 yields

$$\mathbf{H}_{k-1}^{\dagger}\mathbf{c}_k=\mathbf{0} \tag{23}$$

Eq. 23 implies that $\mathbf{c}_k \in \mathbb{R}^{N\times 1}$ is orthogonal to the column space of $\mathbf{H}_{k-1}$. If N training data are distinguished, then $\mathbf{H}_k$ is of full column rank when $L \leq N$ [50]. As $\left(\mathbf{H}_{k-1}^{\dagger}\mathbf{c}_k\right)^{\dagger} = \mathbf{c}_k^{\dagger}\mathbf{H}_{k-1}$, then from Eq. 23, $\mathbf{c}_k^{\dagger}\mathbf{H}_{k-1} = \mathbf{0}$.

Given Eq. 23, let $\mathbf{c}_k \neq \mathbf{0}$ or $\mathbf{c}_k = \mathbf{0}$. First $\mathbf{c}_k \neq \mathbf{0}$ and $\mathbf{c}_k^{\dagger}\mathbf{c}_k = 1$. If Eq. 22 is substituted into $\mathbf{c}_k$ in $\mathbf{c}_k^{\dagger}\mathbf{c}_k = 1$, we have $\mathbf{c}_k^{\dagger}(\mathbf{h}_k - \mathbf{H}_{k-1}\mathbf{d}_k) = 1$ and after expansion, we obtain $\mathbf{c}_k^{\dagger}\mathbf{h}_k - \mathbf{c}_k^{\dagger}\mathbf{H}_{k-1}\mathbf{d}_k = 1$. As $\mathbf{c}_k^{\dagger}\mathbf{H}_{k-1} = 0$, then

$$\mathbf{c}_k^{\dagger}\mathbf{h}_k = 1 \tag{24}$$

The aim is to obtain $\mathbf{g}_k$ which is the only vector required to deduce $\mathbf{H}_k^{\dagger}$. By observing that $\mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{g}_k$ in Eq. 21, we need this to equal $\mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{c}_k^{\dagger}$. If this is the case, then we can eliminate the unnecessary terms and solve $\mathbf{g}_k$. Hence, let

$$\mathbf{P}_k = \mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{c}_k^{\dagger} \tag{25}$$

Multiplying Eq. 25 from the right to $\mathbf{h}_k$ yields $\mathbf{P}_k\mathbf{h}_k = \mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger}\mathbf{h}_k + \mathbf{c}_k\mathbf{c}_k^{\dagger}\mathbf{h}_k$., with $\mathbf{d}_k = \mathbf{H}_{k-1}^{\dagger}\mathbf{h}_k$, $1 = \mathbf{c}_k^{\dagger}\mathbf{h}_k$, and $\mathbf{c}_k = \mathbf{h}_k - \mathbf{H}_{k-1}\mathbf{d}_k$ from Eq. 17, Eq. 24, and Eq. 22, respectively, $\mathbf{P}_k\mathbf{h}_k = \mathbf{h}_k$. Similarly, multiplying Eq. 25 from the right to $\mathbf{H}_{k-1}$ yields $\mathbf{P}_k\mathbf{H}_{k-1} = \mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger}\mathbf{H}_{k-1} + \mathbf{c}_k\mathbf{c}_k^{\dagger}\mathbf{H}_{k-1}$. As $\mathbf{H}_{k-1}^{\dagger}\mathbf{H}_{k-1} = \mathbf{I}$ and $\mathbf{c}_k^{\dagger}\mathbf{H}_{k-1} = 0$, then $\mathbf{P}_k\mathbf{H}_{k-1} = \mathbf{H}_{k-1}$. According to similar reasoning, by multiplying $\mathbf{P}_k$ to with $\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{k-1} & \mathbf{h}_k \end{bmatrix}$ in Eq. 14, we have $\mathbf{P}_k\mathbf{H}_k = \begin{bmatrix} \mathbf{P}_k\mathbf{H}_{k-1} & \mathbf{P}_k\mathbf{h}_k \end{bmatrix}$. As $\mathbf{P}_k\mathbf{h}_k = \mathbf{h}_k$ and $\mathbf{P}_k\mathbf{H}_{k-1} = \mathbf{H}_{k-1}$, as shown above, then

$$\mathbf{P}_k\mathbf{H}_k = \mathbf{H}_k \tag{26}$$

Since $\mathbf{H}_k\mathbf{H}_k^{\dagger}\mathbf{H}_k = \mathbf{H}_k$ (page 28 of [51]), $\mathbf{P}_k$ in Eq. 26 is hence equal to $\mathbf{H}_k\mathbf{H}_k^{\dagger}$. Moreover, Eq. 21 can be rewritten as $\mathbf{P}_k = \mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{g}_k$. Therefore, $\mathbf{P}_k = \mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{c}_k^{\dagger}$ in Eq. 25, and $\mathbf{P}_k = \mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{g}_k$ are equal; thus, we have

$$\mathbf{g}_k = \mathbf{c}_k^{\dagger} \tag{27}$$

For the case $\mathbf{c_k} = \mathbf{0}$, with similar reasoning,

$$\mathbf{g}_k = \left(1 + \mathbf{d}_k^T\mathbf{d}_k\right)^{-1}\mathbf{d}_k^{\mathbf{T}}\mathbf{H}_{k-1}^{\dagger}. \tag{28}$$

305 Readers may refer to [50] for more details. Finally, $\mathbf{H}_k^{\dagger}$ can be estimated by substituting Eq. 19 and Eq. 27 or Eq. 28 ($\mathbf{c}_k^{\dagger} \in \mathbb{R}^{1\times N}$ can be solved using Eq. 22) to obtain $\mathbf{H}_k^{\dagger} = \begin{bmatrix} \mathbf{H}_{k-1}^{\dagger} - \mathbf{d}_k \mathbf{g}_k \\ \mathbf{g}_k \end{bmatrix}$. Given these three essential components of SPI-ELM, we describe the overall procedure for SPI-ELM in Algorithm 2.

Algorithm 2. SPI-ELM

```
Input:
1) Given design matrix $\mathbf{H}_k$, maximum number of hidden neurons $L_{max}$, and sparsity factor $S_p$.
Procedure:
% Condition number computation
2) Compute $\mathbf{h}_1$ and initiate $Cond\left(\mathbf{h}_1^{\mathbf{T}}\mathbf{h}_1\right) = \omega_1 = 1$.
For $k = 2$ to $L_{max}$
    3) Select the next basis vector $\mathbf{h}_k$.
    4) Compute the new term $\omega_k = Cond\left(\mathbf{H}_k^{\mathbf{T}}\mathbf{H}_k\right) - Cond\left(\mathbf{H}_{k-1}^{\mathbf{T}}\mathbf{H}_{k-1}\right)$ where $\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{k-1} \\ \mathbf{h}_k \end{bmatrix}$.
    5) Store $\omega_k$ in vector .
End For.
% Network sparsification
6) Sort $\omega$ and its indices ($k$) in ascending order.
7) Choose only the first $S_p$% of the indices.
8) Select columns from $\mathbf{H}_k$ in terms of these indices to create $\mathbf{H}_{Sp}$.
% Iterative Matrix Decomposition to create $\mathbf{H}_{Sp}^{\dagger}$
While $L_k \leq L_{max}$
    $\mathbf{d}_k = \mathbf{H}_{Sp,k-1}^{\dagger}\mathbf{h}_{Sp,k}$
    $\mathbf{c}_k = \mathbf{h}_{Sp,k} - \mathbf{H}_{Sp,k-1}\mathbf{d}_k$
    If $\mathbf{c}_k \neq \mathbf{0}$
        $\mathbf{g}_k = \mathbf{c}_k^{\dagger}$
    else $\mathbf{g}_k = \left(1 + \mathbf{d}_k^T\mathbf{d}_k\right)^{-1}\mathbf{d}_k^T\mathbf{H}_{k-1}^{\dagger}$
    End if
    $\mathbf{H}_{Sp,k}^{\dagger} = \begin{pmatrix} \mathbf{H}_{Sp,k-1}^{\dagger} - \mathbf{d}_k\mathbf{g}_k \\ \mathbf{g}_k \end{pmatrix}$
    $\mathbf{B}_{Sp} = \mathbf{H}_{Sp}^{\dagger}T$
End while.
```

We store the difference of condition numbers after adding any hidden neuron to the network. In this manner, we can identify which neuron contributes higher condition number. In the last stage of the algorithm above, $\mathbf{H}_{Sp,k-1}$ comprises the first $(k-1)$ columns chosen from $\mathbf{H}_{Sp}$ and $\mathbf{h}_{Sp,k}$ is a vector among all the vectors that belong to $\mathbf{H}_{Sp}$. Given the estimated $\mathbf{B}_{Sp}$, the network output $\widehat{\mathbf{T}}$ is computed via $\widehat{\mathbf{T}} = \mathbf{H}_{Sp}\widehat{\mathbf{B}}_{Sp}$.

4.5. Discussion

In this section, we consider the theoretical analysis of the training and prediction time complexity, as well as the generalizability performance of both ELM and SPI-ELM.

Note that the training time complexity for $\mathbf{H}^{\dagger}=\left(\mathbf{H}^{T}\mathbf{H}\right)^{-1}\mathbf{H}^{T}$ in ELM is $O\left(2L^{2}N+L^{3}\right)$ [12]. However, with recursive matrix decomposition formulation above, the time complexity of ELM reduces to $O((4L+2)N)$ for the case $\mathbf{c}_{k}\neq\mathbf{0}$ or $O(L(5N+1))$ for the case $\mathbf{c}_{k}=\mathbf{0}$ because the time complexities for $\mathbf{d}_{k}=\mathbf{H}_{k-1}^{\dagger}\mathbf{h}_{k}$, $\mathbf{c}_{k}=\mathbf{h}_{k}-\mathbf{H}_{k-1}\mathbf{d}_{k}$, $\mathbf{g}_{k}=\mathbf{c}_{k}^{\dagger}$ or $\mathbf{g}_{\mathrm{k}}=\left(1+\mathbf{d}_{k}^{T}\mathbf{d}_{k}\right)^{-1}\mathbf{d}_{k}^{T}\mathbf{H}_{k-1}^{\dagger}$, $\mathbf{d}_{k}\mathbf{g}_{k}$ in $\mathbf{H}_{k}^{\dagger}=\begin{bmatrix}\mathbf{H}_{k-1}^{\dagger}-\mathbf{d}_{k}\mathbf{g}_{k}\\ \mathbf{g}_{k}\end{bmatrix}$, and $\mathbf{B}=\mathbf{H}_{k}^{\dagger}T$ are LN, LN, $2N$ or $LN+L$, LN, and LN, respectively. It should be noted that we have to compute $\mathbf{H}_{k}^{\dagger}$ only for the first hidden neuron and computations are not required for the following iterations because $\mathbf{H}_{k}^{\dagger}$ can be solved via $\mathbf{H}_{k-1}^{\dagger}$. It is also worth noting that in practice the case $\mathbf{c}_{k}\neq\mathbf{0}$ often happens than the case $\mathbf{c}_{k}=\mathbf{0}$.

The prediction time complexity, i.e., $\widehat{\mathbf{T}}=\mathbf{H}\widehat{\mathbf{B}}$**,** *is* $O(NL)$ for vanilla ELM, whereas it is $O(Nl)$ for SPI-ELM, where l is the total number of *effective* hidden neurons (non-zero activation) and $l<<L$. Therefore, SPI-ELM appears to be much more efficient at prediction.

5. Experiments

5.1. Data Sets and Setup

To verify the reliability of the proposed method, we used 20 benchmark data sets from the UCI data repository [48] and three large databases from computer vision (CV) domains. Detailed information regarding the 20 UCI data sets is given in Table 4.

Table 4. Descriptions of the data sets*

#	Data set name	# Samples	# attributes (R/I/N)*	# Classes
Binary class problems				
1	Wbcd	683	9 (0/9/0)	2
2	Australia	690	33 (32/1/0)	2
3	Credit-app	690	15 (6/0/9)	2

4	Heart	270	13 (1/12/0)	2
5	Tic-Tac-Toe	958	9 (9/0/0)	2
6	Ionosphere	351	33 (32/1/0)	2
7	Pima	768	8 (8/0/0)	2
8	Bupa	345	6 (1/5/0)	2
9	Loan	700	8 (8/0/0)	2
10	Parkinson	195	23 (23/0/0)	2
11	Sonar	208	60 (60/0/0)	2
Multi-class problems				
12	Iris	150	4 (4/0/0)	3
13	Wine	178	13 (13/0/0)	3
14	Balance	625	4 (4/0/0)	3
15	Nursery	12960	8 (8/0/0)	5
16	Zoo	101	16 (0/0/16)	7
17	Segment	2310	19 (19/0/0)	7
18	Ecoli	336	7 (7/0/0)	8
19	Pendigit	10992	16 (0/16/0)	10
20	Optdigit	5620	64 (0/64/0)	10

* R: real, I: integer, N: nominal

For CV datasets, the first database comprised the German Traffic Sign Recognition Benchmark (GTSRB), which was first introduced by Stallkamp et al. [52]. The GTSRB images were collected from recorded video sequences and they capture great variations in the appearance of traffic signs due to differences in illumination, sunlight overexposure or underexposure, partial occlusions, and rotations. This large database contains more than 50,000 images with 39,209 training images and 12,630 testing images, which are divided into 43 classes with unbalanced class distribution. In the original database, the size of the traffic sign images varies between 15×15 and 250×250 pixels. In this study, all of the images were down-sampled or up-sampled to 40×40 via bilinear interpolation.

The Mixed National Institute of Standards and Technology (MNIST) handwritten digit recognition database [53] contains 70,000 images with 60,000 training images and 12,630 testing images, which are divided into 10 classes. While preserving the aspect ratio of the original black and white images, they were normalized to 20×20

pixels. Next, the images were centered in a 28×28 image by computing the center of mass of the pixels [54].

The last database is KUL Belgium Traffic Sign Classification benchmark (BTSC) built for traffic sign classification [55]. BTSC contains 7095 cropped traffic signs images divided to 4575 training images and 2520 images in 62 different classes. Similar to GTSRB, we also rescale all images of the BTSC to 40×40 pixels. Figure 5 shows some samples from the GTSRB and MNIST databases.

Figure 5: Two examples from real-world data sets: MNIST (left) and GTSRB (right).

The experimental setup involved the following.

1. For UCI data sets, each data set was divided into training and testing sets using 10 different random permutations without replacement. For each permutation, 80% of the data were used for training and the remainder for testing, where the mean and maximum of the results obtained were reported as the final results. The training and testing sets were normalized to zero-mean and unit variance. For the CV databases, we ran the models only five times.
2. For the GTSRB and BTSC traffic sign databases, we employed histogram of oriented gradient (HOG) feature descriptors [54]. The cell size was fixed at 5×5. Each four cells (2×2 cells) formed a block and 50% overlapping of blocks was used, where the number of bins was set to eight. With these settings, the HOG feature size was 1568. For the MNIST database, every image with a size of 28×28 was directly vectorized to a feature vector without applying any specific feature extractor.

3. Performance metrics were determined based on the training time, testing time, test accuracy (generalizability performance) and number of effective hidden neurons.
4. The results were compared with ELM and two sparse variants of ELM called TROP-ELM [4] and SB-ELM [7]. For the UCI data sets and all the ELM-based models except SB-ELM, the number of hidden neurons L was in the range of $round\,(\{0.2,\ 0.3, \ldots,\ 0.8\} \times number\ of\ training\ samples)$. Moreover, the sparsity factors for SPI-ELM were S_p=$\{5\%,\ 10\%,\ 15\%, \ldots,\ 40\%\}$. For the MNIST and GTSRB databases, all the ELM-based models except SB-ELM, to avoid "out of memory" errors, L=$\{15k,\ 20k, ...,\ 40k\}$ was considered and for BTSC database, L=$\{1500,\ 2000, ...,\ 4000\}$. The sparsity factors were S_p=$\{30\%,\ 40\%, \ldots,\ 80\%\}$.

   For SB-ELM, training depends on the cubic number of hidden neurons so it is very slow, and a previous study [7] suggested using a small number of hidden neurons to increase the classification accuracy, where the *seed* used for generating uniformly random $\{\mathbf{a},\ b\}$ must also be tuned. Thus, in the training stage for SB-ELM, a fivefold cross-validation was performed to determine the number of hidden neurons L and the seed s for generating uniformly random hidden weights with respect to the range $[20,\ 40,\ 60, ...,\ 200]$, $[1,\ 2,\ 3, ...,\ 10]$ for L and s, respectively.
5. Sigmoid and RBF functions were used for ELM, SPI-ELM, TROP-ELM, and SB-ELM. The random weights associated with both activation functions, $\{\mathbf{a},\ b\}$ were drawn from a uniform distribution of [–1, 1].
6. For multiclass problems, all of the ELM-based models except SB-ELM used multiple output neurons, where the number of output neurons was equal to the number of classes. Hence, a winner-takes-all approach was used to determine the class of the samples. SB-ELM employs the idea of pairwise coupling [56, 57] to solve multiclass problems.
7. The models were implemented on a personal computer with an Intel(R), i5 core, 3.4 GHz processor and 24 GB of installed RAM. MATLAB was used to perform the experiments.

5.2. Results Based on UCI Data Sets

Tables 5 and 6 show the test performance obtained using ELM, TROP-ELM, SB-ELM, and SPI-ELM based on 20 UCI data sets with sigmoid and RBF random hidden neurons, respectively. We note that the test accuracy of SPI-ELM was better than that of TROP-ELM and SB-ELM when equipped with the sigmoid function, and also better than SB-ELM and highly competitive with TROP-ELM when using the RBF function for most of the data sets according to their optimal configurations. Both SPI-ELM and TROP-ELM had much better average test accuracies than ELM for most of the data sets, whereas the performance of SB-ELM differed little from that of vanilla ELM.

Table 5. Test accuracy of ELM-based models for UCI data sets* (sigmoid neurons)

Model/Data set	ELM		TROP-ELM		SB-ELM		SPI-ELM	
	Average	Max	Average	Max	Average	Max	Average	Max
Wbcd	0.9635	0.9927	**0.9788**	0.9927	0.9635	0.9845	0.9759	0.9927
Australia	0.8565	0.9058	**0.8674**	0.9130	0.8665	0.8986	**0.8674**	0.9275
Credit	0.8246	0.8768	0.8341	0.8841	0. 845	0. 885	**0.8464**	0.8986
Heart	0.8056	0.8889	0.8426	0.9444	**0.8889**	0.8889	0.8574	0.9444
Tic-Tac	**0.9870**	1	0.9053	0.9346	0.8292	0.8541	0.9797	0.9948
Iono	0.8943	0.9429	0.8929	0.9571	**0.9571**	0.9857	0.8943	0.9714
Pima	0.7273	0.7987	0.7753	0.8182	**0.8095**	0.8312	0.7869	0.8217
Bupa	0.7043	0.7826	0.7101	0.7826	0.7004	0.7826	**0.7116**	0.7826
Loan	0.76	0.8214	**0.7957**	0.8571	0.7664	0.7857	**0.7957**	0.8571
Parkinson	**0.9205**	1	0.8795	0.9744	0.8744	0.8974	0.9103	1
Sonar	0.7381	0.8571	0.7690	0.9286	**0.8405**	0.9047	0.8214	0.9286
Iris	0.9833	1	0.9833	1	0. 97	1	**0.9844**	1
Wine	0.9111	1	0.9046	0.9722	0.8361	0.8611	**0.9556**	1
Balance	0.8752	0.9120	**0.9552**	0.9733	0.9360	0.9440	0.94	0.9620
Nursery	0.9888	0.9931	0.9888	0.9944	0.9377	0.9425	**0.9896**	0.9944
Zoo	0.9450	0.9875	0.9450	1	0.90	1	**0.9550**	1
Segment	0.9459	0.9502	**0.9548**	0.9761	0.9539	0.9732	0.9494	0.9610
Ecoli	0.8075	0.9403	**0.9207**	0.9638	0.8738	0.9125	0.8942	0.9581
Pendigit	0.9911	0.9932	0.9941	0.9964	**0.9952**	0.9973	0.9945	0.9964
Optdigit	0.9884	0.9911	0.9893	0.9954	**0.9895**	0.9920	**0.9895**	0.9934
Average	0.8809	0.9317	0.8943	0.9429	0.8844	0.9177	**0.9050**	0.9492

*Bold values indicate the best value under the same conditions.

Table 6. Test accuracy of ELM-based models for UCI data sets* (RBF neurons)

Model/Data set	ELM		TROP-ELM		SB-ELM		SPI-ELM	
	Average	Max	Average	Max	Average	Max	Average	Max
Wbcd	0.9496	0.9927	**0.9781**	1	0.9642	0.9708	0.9526	0.9781
Australia	0.8543	0.8913	0.8725	0.9130	**0.9235**	0.9493	0.8688	0.9275
Credit	0.8261	0.8551	0.8406	0.8768	0.8320	0.8551	**0.8435**	0.8913
Heart	0.8019	0.9074	0.7019	0.8333	**0.88**	0.9077	0.8481	0.9444
Tic-Tac	0.9812	0.9948	**0.9865**	1	0.8234	0.8750	0.9804	0.9948
Iono	0.8871	0.9429	0.9186	0.9714	**0.9571**	1	0.8814	0.9571
Pima	0.7403	0.7727	0.7610	0.8117	0.7422	0.8182	**0.7682**	0.8117
Bupa	0.6942	0.7681	0.6957	0.7971	0.6870	0.7391	**0.7043**	0.7826
Loan	0.7650	0.8286	0.7929	0.85	**0.8086**	0.8214	0.7986	0.8714
Parkinson	0.9077	1	0.8897	0.9487	**0.9513**	0.9744	0.9128	1
Sonar	0.7643	0.8571	0.7952	0.9048	**0.9029**	0.9524	0.7762	0.8810
Iris	0.9633	1	0.9833	1	0. 97	1	**0.9867**	1
Wine	0.9278	0.9722	**0.9713**	0.9722	0.8889	0.8889	0.9572	1
Balance	0.8952	0.9360	0.9480	0.9760	**0.9892**	1	0.9592	0.9760
Nursery	0.98558	0.9931	0.9891	0.9944	0.9244	0.9282	**0.9910**	0.9955
Zoo	0.8850	1	**0.9391**	0.9929	0.895	0.9	0.9150	1
Segment	0.9368	0.9577	**0.9649**	0.9855	0.9608	0.9719	0.9517	0.9701
Ecoli	0.8567	0.9104	**0.9120**	0.9683	0.70	0.7015	0.8746	0.9552
Pendigit	0.9927	0.9946	0.9961	0.9978	0.9972	0.9972	**0.9980**	0.9991
Optdigit	0.9905	0.9911	0.9927	0.9954	0.9822	0.9902	**0.9944**	0.9954
Average	0.8803	0.9283	0.8965	0.9395	0.8847	0.9121	**0.8981**	0.9466

*Bold values indicate the best value under the same conditions.

In Tables 7 and 8, for TROP-ELM, SB-ELM and SPI-ELM, "# neur" indicates the number of hidden neurons before sparsification and "#used" shows the number of effective hidden neurons after sparsification. For SPI-ELM, the number of effective hidden neurons after sparsification (# used) was determined by S_p. For example, in the case of the sigmoid hidden neurons in Table 7, for the Wisconsin breast cancer data set (Wbcd), ELM used 109 hidden neurons whereas TROP-ELM, SB-ELM, and SPI-ELM used 88, 8, and 44 hidden neurons, respectively. The optimal average performance of

TROP-ELM with 88 neurons was 0.9788, that for SB-ELM with 8 hidden neurons was 0.9635, that for SPI-ELM with 44 hidden neurons was 0.9759, and the accuracy of ELM with 109 hidden neurons was 0.9635, as shown in Table 5.

We note that TROP-ELM uses the least angle regression (LARS) [58] algorithm and a disadvantage of the iterative LARS algorithm is its sensitivity to noise, which may affect the performance of the network. According to experiments conducted by Weisberg [58], variable selection using LARS appears to be problematic with highly correlated variables, thereby limiting the application of LARS to high dimensional data. A detailed discussion of these problems was given by Weisberg [58].

SB-ELM uses a sparsity approach based on RVM [17] but RVM obtains poor predictions for testing samples located far from the relevance vectors. Hence, the performance of SB-ELM is also affected by this issue. This problem was discussed in detail in a previous study [21].

Table 7. Number of hidden neurons used with several UCI data sets* (sigmoid hidden neurons)

Data set	"Sigmoid" Hidden Neurons			
	ELM	TROP-ELM	SB-ELM	SPI-ELM
	# neur	(# neur, # used)	(# neur, # used)	(# neur, S_p, #used)
Wbcd	109	(218, 88)	(180, **8**)	(218, 0.2, 44)
Australia	110	(221, 81)	(120, **9**)	(221, 0.15, 33)
Credit	166	(110, 42)	(160, 35)	(221, 0.15, **33**)
Heart	65	(65, 33)	(140, **6**)	(86, 0.15, 13)
Tic-Tac	230	(306, 176)	(160, **33**)	(460, 0.25, 115)
Iono	56	(112, 51)	(120, **13**)	(141, 0.2, 28)
Pima	123	(246, 41)	(60, **9**)	(184, 0.1, 18)
Bupa	55	(55, 24)	(80, 9)	(55, 0.05, **3**)
Loan	112	(224, 55)	(40, **6**)	(168, 0.1, 17)
Parkinson	62	(47, **24**)	(160, **6**)	(125, 0.35, 44)
Sonar	66	(66, 27)	(180, 19)	(83, 0.2, **17**)
Iris	24	(48, 30)	(20, **2**)	(48, 0.15, 7)
Wine	28	(57, 29)	(140, **4**)	(57, 0.15, 9)
Balance	150	(250, 72)	(140, 7)	(250, 0.05, **13**)
Nursery	4147	(4147, 815)	(200, 26)	(4147, 0.1, 415)
Zoo	32	(24, 18)	(140, **3**)	(49, 0.25, 12)

Segment	307	(554, 426)	(180, 5)	(554, 0.1, **55**)
Ecoli	54	(108, 90)	(160, 19)	(108, 0.15, **16**)
Pendigit	2638	(2638, 762)	(100, **7**)	(2638, 0.1, 264)
Optdigit	2698	(1798, 539)	(200, **9**)	(2698, 0.05, 135)
Sum	11232	(-, 3423)	(-, **235**)	(-, -, 1291)

*Bold values indicate the best value under the same conditions.

Table 8. Number of hidden neurons used with UCI data sets* (RBF neurons)

Data set	"RBF" Hidden Neurons			
	ELM	TROP-ELM	SB-ELM	SPI-ELM
	# neur	(# neur, # used)	(# neur, # used)	(# neur, S_p, #used)
Wbcd	109	(109, 48)	(40, 18)	(109, 0.05, **5**)
Australia	110	(221, 69)	(60, **19**)	(221, 0.15, 33)
Credit	110	(166, 56)	(140, 28)	(166, 0.1, **17**)
Heart	43	(65, 9)	(140, 5)	(43, 0.05, **2**)
Tic-Tac	383	(230, 104)	(140, **37**)	(613, 0.35, 215)
Iono	84	(84, 37)	(140, 23)	(112, 0.15, **17**)
Pima	123	(123, 32)	(100, 13)	(123, 0.05, **6**)
Bupa	55	(55, 23)	(140, 9)	(55, 0.05, **3**)
Loan	112	(112, 23)	(20, 7)	(112, 0.05, **6**)
Parkinson	47	(31, 18)	(100, 23)	(78, 0.2, **16**)
Sonar	50	(50, **16**)	(140, 111)	(133, 0.35, 47)
Iris	24	(24, 16)	(20, **3**)	(72, 0.25, 18)
Wine	28	(28, 24)	(120, **13**)	(114, 0.35, 40)
Balance	100	(200, 59)	(180, **7**)	(200, 0.15, 30)
Nursery	4147	(4147, 1020)	(200, **56**)	(4147, 0.05, 208)
Zoo	32	(32, 26)	(140, 31)	(49, 0.25, **12**)
Segment	307	(307, 225)	(200, **12**)	(554, 0.2, 110)
Ecoli	54	(108, 79)	(200, **8**)	(81, 0.1, **8**)
Pendigit	2638	(2638, 711)	(180, **130**)	(2638, 0.1, 264)
Optdigit	2698	(2698,909)	(180, **124**)	(1798, 0.1, 180)
Sum	11254	(-, 3504)	(-, **677**)	(-, -, 1237)

*Bold values indicate the best value under the same conditions.

Table 9 compares the training times for ELM, TROP-ELM, SB-ELM, and SPI-ELM using sigmoid and RBF functions. Recursive matrix decomposition is used in

SPI-ELM; thus, it was several times faster than TROP-ELM and SB-ELM, but slower than ELM. For example, using the Nursery data set, SPI-ELM was almost 10 times faster than TROP-ELM and almost 20 times faster than SB-ELM. Indeed, TROP-ELM required much more time because it uses the cascade L_1-L_2 approach [4], where the algorithm first ranks the neurons with LARS (L_1-regularization) and then select the best neurons with Tikhonov-regularized PRESS (TR-PRESS) algorithm (L_2 regularization) among the ranked neurons. These steps are fairly time consuming. SB-ELM employs the idea of sparsity in RVM; thus, Cholesky decomposition is required to compute the posterior weight covariance matrix [17], which is very time consuming and it depends on the cubic number of hidden neurons. Hence, training is very time consuming using SB-ELM. Although SB-ELM uses few hidden neurons, with a maximum of 200, tuning the seed parameter and hidden neurons based on a fivefold cross validation increased the time required for training, as shown in Table 9.

Table 9. Comparison of the training times in seconds

”Sigmoid” Hidden Neurons					”RBF” Hidden Neurons			
Classifier / data	ELM	TROP-ELM	SB-ELM	SPI-ELM	ELM	TROP-ELM	SB-ELM	SPI-ELM
Wbcd	0.0045	1.2967	35.6528	0.7528	0.0041	1.5956	57.6959	0.8131
Australia	0.0035	6.8416	38.8827	2.4820	0.0041	9.3979	59.7922	1.0651
Credit	0.0072	2.3423	41.4557	1.1109	0.0041	2.4842	58.5914	0.2478
Heart	0.0011	0.1817	26.4821	0.1136	0.0010	0.1723	21.6298	0.0439
Tic-tac	0.0152	16.4593	64.5217	5.2833	0.0433	13.5854	61.7675	5.8481
Iono	0.0013	0.6517	35.893	0.1210	0.0019	0.4451	42.8463	0.1352
Pima	0.0042	10.5326	29.486	2.1441	0.0046	1.5291	42.3799	0.6043
Bupa	0.0011	0.3619	19.3483	0.0945	0.0013	0.3379	25.8038	0.1746
Loan	0.0034	6.9119	30.3701	1.2774	0.0038	0.9459	39.7952	0.7962
Parkinson	0.0012	0.0869	19.5859	0.0949	0.0073	0.2755	20.8047	0.1597
Sonar	0.0014	0.1788	28.3079	0.0747	0.0008	0.1062	22.0478	0.0882
Iris	0.0066	0.0738	39.6604	0.0147	0.0048	0.2422	38.4232	0.0943
Wine	0.0007	0.1242	49.0718	0.0694	0.0021	0.2793	37.4886	0.1353
Balance	0.0082	0.3224	77.9478	0.1076	0.0027	4.7051	80.0379	1.2672
Nursery	5.5947	296.205	455.1173	30.8228	5.829	289.3453	570.1628	39.7336
Zoo	0.0004	0.0316	146.2442	0.0315	0.0003	0.0486	135.5614	0.0526
Segment	0.0606	207.8026	595.1866	56.3619	0.0781	244.821	570.0803	19.8052
Ecoli	0.0013	0.5869	27.0971	0.1428	0.0036	0.3668	87.1808	0.1125

Pendigit	3.9995	103.282	3.146e+3	15.9981	3.8801	99.345	2.818e+3	17.3282
Optdigit	17.5628	191.425	2.295e+3	47.4657	14.4592	157.5235	1.631e+3	26.4565
Sum	27.2789	846.001	7201.311	164.5637	24.3362	827.5519	6421.09	114.962

In terms of the prediction time, the empirical results in Table 10 show that SPI-ELM was obviously the fastest with all the data sets due to the sparse nature of the model. It should be noted that with a smaller number of hidden neurons, the prediction time using SPI-ELM was much less than that with SB-ELM, and also less than that using both ELM and TROP-ELM. According to Tables 7 and 8, SB-ELM is much sparser than TROP-ELM and the proposed SPI-ELM, but its prediction time was much longer. This is because SB-ELM uses a pairwise coupling strategy for multiclass problems and p_k, $k = 1...m$ probabilities must be estimated, where m is the total number of classes. The problem of estimating p_k is equivalent to solving a linear equality-constrained convex quadratic programming problem. Hence, its prediction time is high.

Table 10. Comparison of the testing times in seconds

”Sigmoid” Hidden Neurons					”RBF” Hidden Neurons			
Classifier / data	ELM	TROP-ELM	SB-ELM	SPI-ELM	ELM	TROP-ELM	SB-ELM	SPI-ELM
Wbcd	1.666e-4	5.90e-4	0.0014	**6.49e-5**	1.295e-4	0.0068	6.855e-4	**9.298e-5**
Australia	1.304e-4	5.144e-4	9.156e-4	**5.856e-5**	1.301e-4	4.319e-4	7.085e-4	**6.460e-5**
Credit	6.128e-5	6.9e-2	0.0014	**5.826e-5**	3.595e-4	0.0013	6.774e-4	**3.049e-5**
Heart	2.388e-5	7.845e-4	0.0011	**1.807e-5**	**2.928e-4**	7.646e-4	5.944e-4	**2.976e-4**
Tic-tac	7.395e-4	3.372e-4	0.0014	**6.40e-5**	1.289e-4	0.0028	7.776e-4	**5.886e-5**
Iono	9.358e-5	4.202e-4	6.318e-4	**3.170e-6**	**1.781e-5**	8.081e-4	6.230e-4	2.877e-5
Pima	1.343e-4	3.719e-4	7.127e-4	**4.437e-5**	1.325e-4	0.0071	6.840e-4	**1.204e-4**
Bupa	9.358e-5	6.658e-2	5.92e-4	**7.237e-5**	3.402e-4	0.0062	6.246e-4	**2.835e-4**
Loan	1.340e-4	4.516e-4	6.478e-4	**4.528e-5**	1.259e-4	0.0066	6.499e-4	**9.147e-5**
Parkins	3.773e-5	7.686e-4	5.639e-4	**6.037e-6**	1.117e-5	0.0066	6.056e-4	**2.928e-6**
Sonar	4.014e-5	3.565e-4	5.922e-4	**3.019e-5**	1.389e-5	7.362e-4	6.774e-4	**5.132e-6**
Iris	9.870e-5	4.108e-4	0.0016	**2.656e-5**	2.825e-4	0.0072	0.0017	**3.954e-5**
Wine	3.278e-4	4.525e-4	0.0019	**2.777e-5**	3.151e-4	0.0069	0.0019	**4.83e-6**
Balance	1.654e-4	5.947e-4	0.0041	**4.166e-5**	1.247e-4	5.491e-4	0.0038	**5.433e-5**
Nursery	0.0186	0.0148	0.0898	**0.0126**	0.0331	0.0139	0.0964	**0.0117**
Zoo	3.350e-5	3.476e-4	0.0059	**1.894e-5**	**1.177e-5**	5.340e-4	0.006	3.894e-5
Segment	4.978e-4	0.0115	0.0249	**1.675e-4**	2.465e-4	0.0255	0.026	**1.528e-4**
Ecoli	3.082e-4	6.125e-4	3.572e-3	**3.079e-5**	3.314e-4	8.054e-4	0.0059	**1.389e-5**

Pendigit	0.0058	0.0089	0.1499	**0.0022**	0.0066	0.0096	0.3696	**0.0019**
Optdigit	0.03	0.1387	0.087	**0.002**	0.0290	0.1355	0.1837	**0.0023**
Sum	2.87e-3	1.58e-2	0.01754	**8.79e-4**	7.17e-2	0.0120	0.7023	**8.64e-4**

*Bold values indicate the best value under the same conditions.

Figure 6 shows the number of hidden neurons with ELM versus the effective number of hidden neurons using TROP-ELM and SPI-ELM with the sigmoid function. SB-ELM only uses a maximum of 200 hidden neurons; thus, we excluded this model from the comparison. For SPI-ELM, there was a constant trend in the growth of the number of effective hidden neurons in most cases. We note that high accuracy could be obtained with a small S_p and large L when using SPI-ELM. For TROP-ELM, the number of effective hidden neurons appeared to increase in a linear manner with respect to L because the network selects the combination of ranked hidden neurons that minimizes the training error [4]. Therefore, TROP-ELM could use a large number of hidden neurons and the pruned model was still large.

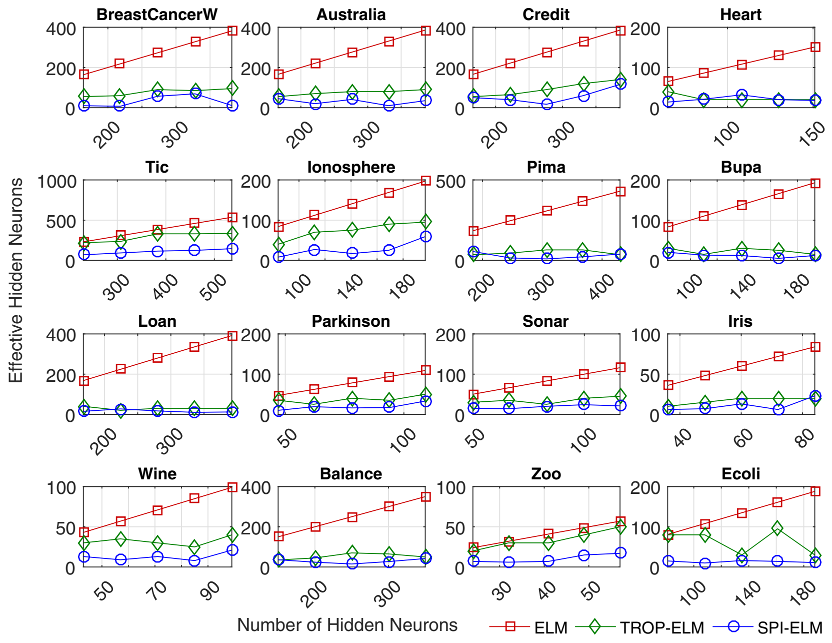

Figure 6: Number of hidden neurons versus the number of effective hidden neurons.

Thus, SPI-ELM was apparently sparser compared with TROP-ELM, as shown in Table 7, Table 8, and Fig. 6. The performance of SPI-ELM was also better than that of TROP-ELM, as demonstrated in Table 5, Table 6, and Fig. 7. Similar results were obtained using SPI-ELM with RBF hidden neurons. Figure 7 indicates that as the number of hidden neurons increased, ELM generalized poorly, whereas the sparse models such as SPI-ELM and TROP-ELM could generalize fairly well despite the large number of hidden neurons.

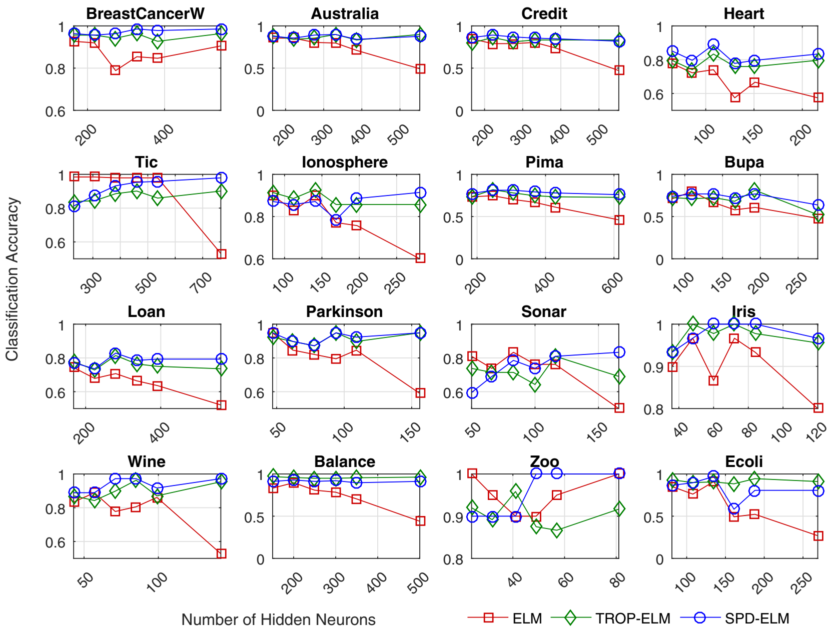

Figure 7: Number of hidden neurons versus the classification accuracy.

Figure 8 shows the reduction in the condition number for sparse models with L. Thus, with a moderate condition number, SPI-ELM achieved higher accuracy with a large L, as shown in Fig. 7. In contrast to ELM, SPI-ELM is not affected by the overfitting problem and the performance of SPI-ELM is also relatively insensitive to L, as shown in Fig. 6 and Fig. 7. According to these experiments, one of the main advantages of SPI-ELM is that the degree of network sparseness can be controlled easily by the sparsity factor S_p.

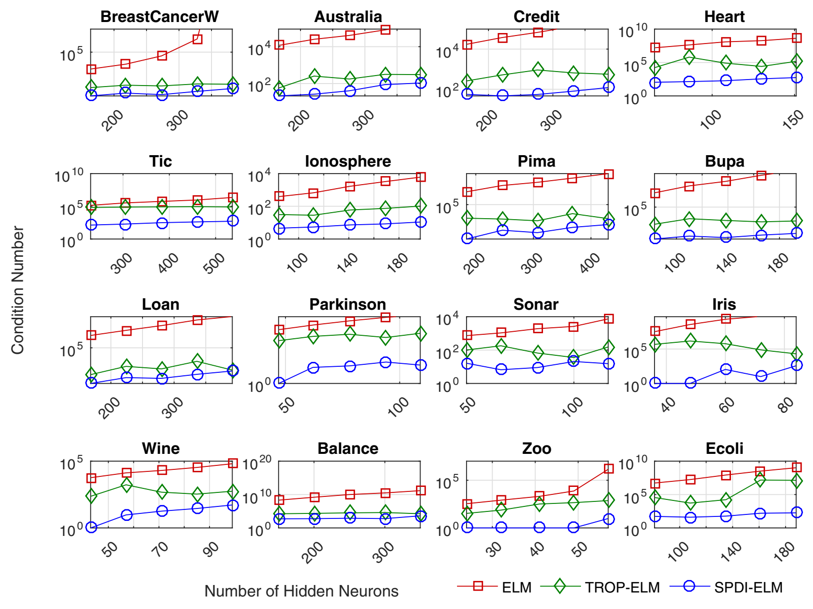

Figure 8: Number of hidden neurons versus the condition number (log-scale).

5.3. *Results based on CV Databases*

Tables 11 and 12 show the performance of the four ELM-based models using three real-world databases (GTSRB, BTSC, and MNIST) with sigmoid and RBF functions, respectively. The maximum accuracy as well as the corresponding training and testing times are presented with triple (Acc, Tr time, Te time).

Table 11. Performance of ELM-based models with real-world databases (sigmoid hidden neurons)

Model/Data set	GTSRB			BTSC			MNIST		
	(Acc, Tr time, Te time)			(Acc, Tr time, Te time)			(Acc, Tr time, Te time)		
ELM	0.9327	7min 2s	0.31s	0.9672	54s	0.09s	0.9785	21min 49s	0.9s
TROP-ELM	0.9341	6h 57min	0.21s	0.9795	48min	0.07s	0.9785	7h 26min	0.22s
SB-ELM	0.8523	12h 36min	4.52s	0.9708	1h 7min	0.81s	0.9478	10h 17min	0.37s
SPI-ELM	**0.9492**	2h 14min	0.29s	**0.9811**	26min	0.08s	**0.9812**	2h 11min	0.24s

Boldface indicates the highest accuracy among the classifiers under the same conditions.

Table 12. Performance of ELM-based models with CV databases (RBF neurons)

Model/Data set	GTSRB			BTSC			MNIST		
	(Acc, Tr time, Te time)			(Acc, Tr time, Te time)			(Acc, Tr time, Te time)		
ELM	0.9216	12min 28s	0.36s	0.9651	57s	0.10s	0.9620	20min 32s	0.88s
TROP-ELM	0.9345	6h 35min	0.23s	0.9788	46min	0.06s	0.9595	7h 19min	0.26s
SB-ELM	0.8527	10h 19min	4.56s	0.9711	1h 12min	0.79s	0.9429	9h 52min	0.34s
SPI-ELM	**0.9451**	2h 25min	0.31s	**0.9811**	28min	0.08s	**0.9682**	2h 29min	0.17s

Boldface indicates the highest accuracy among the classifiers under the same conditions.

Tables 11 and 12 show that SPI-ELM using the sigmoid and RBF functions outperformed the other three methods with all real-world databases. SPI-ELM with the sigmoid function had the highest test accuracies of 0.9492 and 0.9812 with GTSRB and MNIST, respectively. For BTSC database, SPI-ELM attains highest test accuracy 0.9811 for both sigmoid and RBF hidden neurons. For GTSRB with 12,630 test samples and the sigmoid function, there were significant improvements of 1.65%, 1.51%, and 9.69% for 208, 190, and 1223 test samples in terms of the correct predictions compared with ELM, TROP-ELM, and SB-ELM, respectively.

The prediction time was better using SPI-ELM than ELM and SB-ELM, and the prediction time with TROP-ELM was the best among all the learners. We also found that the testing time using SB-ELM with GTSRB was much slower than that with MNIST. This is because SB-ELM uses pairwise coupling for prediction where several probabilities that are equal to the number of classes must be computed. BTSC comprises 62 classes compared with 43 classes in the GTSRB and 10 classes in the MNIST database; thus, the computing prediction time was longer for BTSC and GTSRB. The training time with SPI-ELM was slower than ELM but much faster than TROP-ELM and SB-ELM. For TROP-ELM and SPI-ELM, we imposed a break point for termination if training required more than 10 h but we did not change the model settings for SB-ELM because it only uses 200 hidden neurons. Note that SB-ELM with the same settings was also applied to face recognition databases, as described in a previous study [59]. For example, Table 13 shows the training time using ELM, TROP-ELM, and SPI-ELM with the sigmoid function based on the GTSRB database with respect to different numbers of hidden neurons. The training time with TROP-ELM was almost 20 times slower than that using ELM and almost 2 times slower than using SPI-ELM.

Table 13. Training time for ELM-based models using the GTSRB with the sigmoid function

Model/Data set	GTSRB					
	15K	20K	25K	30K	35K	40K
ELM	3min 8s	7min 2s	11min 34s	20min 58s	34min 56s	56min 17s
TROP-ELM	1h 11min	3h 7min	6h 57min	$>$10h	$>$10h	$>$10h
SPI-ELM	57min 22s	1h 49min	3h 37min	6h 11min	9h 22min	$>$10h

$>$10h: indicates that training was terminated if it exceeded 10 hours

Finally, Tables 14 and 15 show the numbers of effective hidden neurons for the ELM models based on the sigmoid and RBF functions, respectively. SPI-ELM could prune thousands of hidden neurons while still achieving the best test accuracy, and training and testing times. Moreover, the number of hidden neurons exceeded the break point for termination in TROP-ELM; thus, this model could not handle more than 25K hidden neurons and the number of effective hidden neurons was less than that with SPI-ELM in many cases. However, according to Tables 11 and 12, the testing performance of TROP-ELM was worse than that of SPI-ELM and its training time was long, which made it impractical. SB-ELM uses only 200 hidden neurons and it is sparser than other networks, but it still had the worst training time, as shown in Table 11 and Table 12.

Table 14. Number of sigmoid hidden neurons used with real-world databases

Model/Data set	GTSRB	BTSC	MNIST
	(# neur, S_p, # used)	(# neur, S_p, # used)	(# neur, S_p, # used)
ELM	(20K, -, 20K)	(3500, -, 3500)	(30K, -, 30K)
TROP-ELM	(25K, -, 13911)	(4000, -, 1125)	(25K, -, 12703)
SB-ELM	(200, -, 21)	(200, -, 33)	(200, -, 101)
SPI-ELM	(30K,0.5,15K)	(4000, 0.3, 1200)	(30K,0.5,15K)

Table 15. Number of RBF hidden neurons used with real-world databases

Model/Data set	GTSRB	BTSC	MNIST
	(# neur, S_p, # used)	(# neur, S_p, # used)	(# neur, S_p, # used)
ELM	(25K, -, 25K)	(3500, -, 3500)	(30K, -, 30K)
TROP-ELM	(25K, -, 14917)	(4000, -, 1292)	(25K, -, 13409)
SB-ELM	(200, -, 24)	(200, -, 46)	(200, -, 107)
SPI-ELM	(30K, 0.6, 18K)	(4000, 0.3, 1200)	(30K,0.4,12K)

6. Conclusion

In this study, we proposed an incremental method for sparsifying vanilla ELM with a novel indicator driven by the condition number in the ELM design matrix, which we call SPI-ELM. SPI-ELM can effectively prune the collinear hidden neurons in the network, which are a major cause of the ill-conditioned problem. The degree of sparseness in the model can be controlled deterministically by an explicit sparsity factor. Therefore, SPI-ELM outperforms ELM with significantly smaller networks. To compensate for the negative effect of the sparsification process on the learning speed using SPI-ELM, we introduced an iterative matrix decomposition algorithm to mitigate this problem. SPI-ELM is also competitive with the state-of-the-art TROP-ELM and SB-ELM methods in terms of accuracy, as well as being superior to TROP-ELM and SB-ELM in terms of its training and prediction complexity. SPI-ELM is compact and learns rapidly; thus, it is suitable for practical applications where lightweight memory is required. In this study, SPI-ELM was established by incrementally adding hidden neurons one by one, but another approach could involve adding more than one hidden neuron during each iteration in a chunking approach. Moreover, considerations of the multicollinearity among the bases and observing the bases that are correlated with each other may also be addressed in future research.

Conflict of interest

None declared.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2013R1A2A2A01015624).

References

[1] I. Iosifidis, A. Tefas, A. Pitas, Sparse extreme learning machine classifier exploiting intrinsic graphs, Pattern Recogn Lett 65 (2015) 192–196.

[2] A. Iosifidis, Extreme learning machine based supervised subspace learning, Neurocomputing 167 (2015) 888–896.

[3] I. Iosifidis, T. A., A. Pitas, Graph embedded extreme learning machine, IEEE Trans. Cybern 46 (2016) 311–324.

[4] A. Miche, Y. Heeswijk, M. V. Bas, P. Simula, O. Lendasse, Trop-elm: A double-regularized elm using lars and tikhonov regularization, Neurocomputing 74 (2011) 2413–2421.

[5] X. Yang, Y. Wu, Q. M. J. Wang, Y. Zeeshan, K. M. Lin, Y. X., Data partition learning with multiple extreme learning machines, IEEE Trans. Cybern 45 (2015) 1463–1475.

[6] C. Huang, G. B. Zhu, Q. Y. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (2006) 489–501.

[7] P.-K. Luo, J. Vong, C.-M. Wong, Sparse bayesian extreme learning machine for multi-classification, IEEE T Neural Networ Learn Syst 25 (2014) 836–843.

[8] C. M. Bishop, Pattern recognition and machine learning, in: Springer 2nd ed. (2006) 738.

[9] C.-K. Huang, G. B. Li, M.-B. Chen, L. Siew, Incremental extreme learning machine with fully complex hidden nodes, Neurocomputing 71 (2008) 576–583.

[10] R. Huang, G. B. Zhou, H. Ding, Z. X., Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man, Cybern. Part B 42 (2012) 513–529.

[11] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE T Neural Networ 20 (2009) 1352–1357.

[12] Y. Ye, Y. Qin, Qr factorization based incremental extreme learning machine with growth of hidden nodes, Pattern Recogn Lett 65 (2015) 177–183.

[13] X. Li, J. Hua, C. Tang, G. Y., A fast training algorithm for extreme learning machine based on matrix decomposition, Neurocomputing 173 (2016) 1951–1958.

[14] Y. Zhou, H., Huang, G-B., Lin, Z., Wang, Han., & Soh, Stacked extreme learning machines,, IEEE Transactions on Cybernetics 45 (9) (2015) 2013–2025.

[15] Z. Bai, G.-B. Huang, D. Wang, H. Wang, M. B. Westover, Sparse extreme learning machine for classification, IEEE Transactions on Cybernetics 44 (10) (2014) 1858–1870.

[16] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: optimally pruned extreme learning machine, IEEE T Neural Networ. 21 (1) (2009) 158–162.

[17] M. Tipping, Sparse Bayesian learning and the relevance vector machine., Mach. Learn. Res., 1 (2001) 211–244.

[18] D. MacKay, Bayesian methods for backpropagation networks., in: Models of Neural Networks III, Springer, 1994, Ch. 6, pp. 211–254. `doi:10.1007/978-1-4612-0723-8_6`.

[19] R. Neal, Bayesian Learning for Neural Networks., 1st Edition, Springer-Verlag, New York, 1996. `doi:10.1007/978-1-4612-0745-0`.

[20] J. Candela, E. Snelson, O. Williams, Sensible priors for sparse Bayesian learning, Tech. rep. (2007).

[21] J. Q. Rasmussen, C.E. & Candela, Healing the relevance vector machine through augmentation, in: in Proc of 22nd International Conference on Machine Learning,, Bonn, Germany, 2005, pp. 689 – 696. doi:10.1145/1102351.1102438.

[22] A. Grigorievskiy, A., Miche, Y., Käpylä, M., & Lendasse, Singular value decomposition update and its application to (Inc)-OP-ELM, Neurocomputing 174 (2016) 99–108.

[23] Z. Zhao, G., Shen, Z., & Man, Robust Input Weight Selection for Well-Conditioned Extreme Learning Machine., International Journal of Inf. Tech. 17 (1) (2011) 1–18.

[24] J. P. Matias, T., Souza, F., Araújo, R., Gonçalves, N., & Barreto, On-line Sequential Extreme Learning Machine Based on Recursive Partial Least Squares, Journal of Process Contr. 27 (2015) 15–21.

[25] J. C. Principe, B. Chen, Universal Approximation with Convex Optimization: Gimmick or Reality?, In IEEE Computational Intelligence Magazine. (2015) 68–77.

[26] C.-K. Huang, G.-B., Chen, L, & Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Neural Networ., 17 (4) (2006) 879–892.

[27] L. Huang, G-B., & Chena, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.

[28] P. Kassani, E. Kim, Pseudoinverse matrix decomposition based incremental extreme learning machine with growth of hidden nodes, International Journal of Fuzzy Logic and Intelligent Syst 16 (2) (2016) 125–130.

[29] F. Ye, Y., Squartini, S., & Piazza, Incremental-based extreme learning machine algorithms for time-variant neural networks., in: Int. Conf. on Intelligent Computing,, Springer-Verlag, Berlin Heidelberg, 2010, pp. 9–16.

[30] Y. C. Zhang, R., Lan, Y., Huang, G-B & Soh, Extreme learning machine with adaptive growth of hidden nodes and incremental updating of output weights, in: AIS'11 Proceedings of the Second international conference on Autonomous and intelligent systems, Springer-Verlag Berlin, Heidelberg, Burnaby, BC, Canada, 2011, pp. 253–262.

[31] Q.-H. Denga, W-Y., Baic, Z., Huang, G-B., & Zheng, A fast SVD-hidden-nodes based extreme learning machine for large-scale data analytics., Neural Networks, 77 (2016) 14–28.

[32] X. Lu, S., Zhang, G., & Wang, A rank reduced matrix method in extreme learning machine, in: International Symposium on Neural Networks, Springer-Verlag: Berlin Heidelberg, 2012, pp. 72–79. doi:10.1007/978-3-642-31346-2_9.

[33] M.-Y. Wei, S., Lu, H., Lu, Y., & Wang, An improved weight optimization and Cholesky decomposition based regularized extreme learning machine for gene expression data classification., Springer International Publishing: Switzerland. (2014) 55–66.

[34] X. Wang, J., Wang, X., Gu, S., Liao, W., & Fang, An Improved extreme learning algorithm based on truncated singular value decomposition, in: In Chinese Control and Decision Conference (CCDC)., 2015. doi:10.1109/CCDC.2015.7162193.

[35] M. Bi, J., Bennett, K., Embrechts, M., Breneman, C., & Song, Dimensionality reduction via sparse support vector machines, Machine Learning Research 3 (2002) 1229–1243.

[36] J. Bi, Y. Chen, J. Z. Wang, A sparse support vector machine approach to region-based image categorization, in: In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005. doi:10.1109/CVPR.2005.48.

[37] P. H. Kassani, A. B. J. Teoh, A new sparse model for traffic sign classification

using soft histogram of oriented gradients, Appl. Soft Comput. 52 (2017) 231–246.

[38] W. Li, X., Mao, W., & Jiang, Fast sparse approximation of extreme learning machine, Neurocomputing 128 (2014) 96–103.

[39] S. Lu, K., Ding, Z., & Ge, Sparse-representation-based graph embedding for traffic sign recognition., IEEE Trans. Intell. Transp. System 13 (4) (2012) 1515–1524.

[40] G. Saarinen, S., Bramley, R., & Cybenko, Ill-conditioning in neural network training problems, SIAM Journal on Scientif. Comput. 14 (3) (1993) 693–714.

[41] C.-K. Huang, G.-B., Zhu, Q.-Y. & Siew, Extreme learning machine: A new learning scheme of feedforward neural networks, in: In Proc: IJCNN, 2004, pp. 985–990.

[42] H. Nielsen, Theory of the backpropagation neural network., in: In IJCNN, International Joint Conference on Neural Nets,, IEEE, 1989. `doi:10.1109/IJCNN.1989.118638.`

[43] S.-H. Sankar, A., Spielman, D, A., Teng, Smoothed Analysis of the Condition Numbers and Growth Factors of Matrices, SIAM. J. Matrix Anal. & Appl., 28 (2) (2006) 446–476.

[44] Y. Wang, F. Cao, Y. Yuan, A study on effectiveness of extreme learning machine, Neurocomputing 74 (16) (2011) 2483–2490.

[45] D. N. Gujarati, D. Porter, Basic Econometrics, McGraw-Hill, New York, NY, 2009, Ch. Chapter 10.

[46] J. Callaghan, K. & Chen, Revisiting the collinear data problem: an assessment of estimator 'ill-conditioning' in linear regression, Practical Assessment, Research & Evaluation 13 (5).

[47] D. Meyer, C, Matrix analysis and applied linear algebra, SIAM, 2000, Ch. 7, p. 718 pages.

[48] UCI, University of California Irvine (UCI) Machine Learning Repository, http://archive.ics.uci.edu/ml/.

[49] G. Saville, D.J. & Wood, A method for teaching statistics using N-dimensional geometry., The American Statistician, 40 (3) (1986) 205–214.

[50] T. N. E. Greville, Some applications of the pseudoinverse of a matrix., SIAM Review, 2 (1) (1960) 15–22. doi:10.1137/1002004.

[51] D. Albert, Regression and the moore-penrose pseudoinverse, mathematics in science and engineering (1972) 1–179.

[52] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, The German traffic sign recognition benchmark: A multiclass classification competition,, in: In Proc. Int. Joint Conf. Neural Nets., 2011.

[53] P. Lecun, Y., Bottou, L., Bengio, Y., & Haffner, Gradient-based learning applied to document recognition., in: In Proceedings of the IEEE., 1998, pp. 2278–2324. doi:10.1109/5.726791.

[54] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection., in: In Computer Vision and Pattern Recognition, (CVPR) Conference, IEEE, 2005. doi:10.1109/CVPR.2005.177.

[55] R. Timofte, K. Zimmermann, L. Gool, Multi-view traffic sign detection recognition and 3d localisation, Machine Vision and Applic 25 (2014) 633–647.

[56] R. C. Wu, T. F., Lin, C. J, & Weng, Probability estimates for multiclass classification by pairwise coupling, Mach. Learn. Res., 5 (2004) 975–1005.

[57] W. F. Vong, C. M., Wong, P. K, & Ip, A new framework of simultaneous-fault diagnosis using pairwise probabilistic multi-label classification for time-dependent patterns., IEEE Trans. Ind. Electron., 60 (8) (2013) 3372–3385.

[58] S. Weisberg, Discussion of Least Angle Regression by Efron., The Annals of Statistics 32 (2) (2004) 490–494.

[59] P.-K. Vong, C-M., Tai, K. I., Pun, C. M., & Wong, Fast and accurate face detection by sparse Bayesian extreme learning machine., Neural Comput & Applic. 26 (5) (2015) 1149–1156.

Peyman Hosseinzadeh Kassani received his master degree in computer science from University of Economic Sciences, Tehran, Iran in 2012 and became a Member of IEEE in 2013. He is currently doing his PhD at Yonsei University, Seoul, South Korea. He is with computational intelligence laboratory in electrical and electronics engineering department of Yonsei University. He has worked on object detection and pattern classification. His general interests lie in machine learning and pattern recognition and their application to computer

Andrew Beng Jin Teoh obtained his B.Eng. (Electronic) in 1999 and Ph.D. degree in 2003 from National University of Malaysia. He is currently an associate professor in Electrical and Electronic Engineering Department, Yonsei University, South Korea. His research, for which he has received funding, focuses on biometric applications and biometric security. His current research interests are Machine Learning and Information Security. He has published more than 250 international refereed journal papers, conference papers, edited several book chapters and edited book volumes. He served and is serving as a guest editor of IEEE Signal Processing Magazine, associate editor of IEEE Biometrics Compendium and editorin- chief of IEEE Biometrics Council Newsletter. He was a program co-chair of ICONIP 2014, area chair of ICPR 2016 and ICIP 2017, track chair and TPC for several conferences related to computer vision, pattern recognition and biometrics.

Euntai Kim was born in Seoul, Korea, in 1970. He received B.S., M.S., and Ph.D. degrees in Electronic Engineering, all from Yonsei University, Seoul, Korea, in 1992, 1994, and 1999, respectively. From 1999 to 2002, he was a Full-Time Lecturer in the Department of Control and Instrumentation Engineering, Hankyong National University, Kyonggi-do, Korea. Since 2002, he has been with the faculty of the School of Electrical and Electronic Engineering, Yonsei University, where he is currently a Professor. He was a Visiting Scholar at the University of Alberta, Edmonton, AB, Canada, in 2003, and also was a Visiting Researcher at the Berkeley Initiative in Soft Computing, University of California, Berkeley, CA, USA, in 2008. His current research interests include computational intelligence and statistical machine learning and their application to intelligent robotics, unmanned vehicles, and robot vision.