Neural Comput & Applic
DOI 10.1007/s00521-014-1803-x

ORIGINAL ARTICLE

Fast and accurate face detection by sparse Bayesian extreme learning machine

Chi Man Vong · Keng Iam Tai · Chi Man Pun · Pak Kin Wong

Received: 9 August 2014 / Accepted: 9 December 2014

Abstract Real-time face detection is an important research topic in computer vision and pattern recognition. One of the effective methods in face detection is model-based approach which employs neural network technique for the construction of classification model. Relevant techniques such as support vector machines are fast in training an accurate model which is, however, relatively slow in execution time. The reason is due to the large size of the constructed model. In this paper, the main contribution is to apply a new method called sparse Bayesian extreme learning machine (SBELM) for real-time face detection because SBELM can minimize the model size with nearly no compromise on the accuracy and have fast execution time. Several benchmark face datasets were employed for the evaluation of SBELM against other state-of-the-art techniques. Experimental results show that SBELM achieves fastest execution time with high accuracy over the benchmark face datasets. A MATLAB toolbox of SBELM is also available on our Web site.

Keywords Face recognition · Face detection · Sparse Bayesian · Extreme learning machine

C. M. Vong (✉) · K. I. Tai · C. M. Pun
Department of Computer and Information Science, University of Macau, Macau, SAR, China
e-mail: cmvong@umac.mo

P. K. Wong
Department of Electromechanical Engineering, University of Macau, Macau, SAR, China

1 Introduction

Real-time face detection is an important research topic which aims to locate human faces from a scene in a very short time with high accuracy. Its applications [1, 2] include real-time human tracking, video surveillance, and security monitoring for important buildings. In the literature, various algorithms have been proposed to tackle this challenging problem including skin color-based method [3], motion-based method [4], mixture of skin color and motion [5], edge-orientation matching [6], and Hausdorff distance [7]. However, all of the above methods suffer from different drawbacks that cannot be easily generalized to complicated unconstrained scenes. From 1990s, neural network approaches have been extensively employed for face detection because it can accurately detect human faces on many unconstrained scenes. Common techniques of neural network approaches include multi-layer perceptron (MLP) [8] and support vector machine (SVM) [9, 10] which usually take a relatively long training time. In recent years, extreme learning machine (ELM) [11, 12] and kernelized ELM (K-ELM) [13, 14] have been proposed to resolve the issue of training time in different aspects [15–17] while the detection accuracy is even improved in many cases. Nevertheless, a major requirement for real-time face detection in addition to accuracy and training time is the model execution time, which is a significant unresolved issue about neural network approaches. From this viewpoint, the motivation of the paper is to construct a model that possesses:

(a) fast execution time for real-time face detection;
(b) high detection accuracy.

The execution time is affected by the model size because the larger the model, the longer the execution time.

For MLP, the model size is dependent on the number of hidden nodes. For applications with thousands of input features, the high number of hidden nodes will generate a huge number of weights w_k, $k = 0$ to L (L is the number of hidden neurons). These weights w_k not only occupy a large amount of memory but also take longer model execution time. For SVM, there is no hidden node but the model size depends on a set of support vectors extracted from its training data. In fact, the case for SVM may be even worse in terms of execution time. It is not surprising that highly nonlinear applications such as face detection can easily require hundreds to thousands of support vectors for the achievement of high detection accuracy. Similar situation happens in K-ELM. In order to tackle the issue, a novel learning algorithm called sparse Bayesian extreme learning machine (SBELM) [18] is employed in this paper.

SBELM inherits the fast training time from ELM and the sparsity of weights from sparse Bayesian learning (SBL) approach [19, 20]. SBELM learns the output weights w_k of ELM classifier by imposing a hierarchical independent hyperprior α_k on each w_k so that a lot of w_k are automatically tuned to zeros during learning phase, while the parameters of hidden layer are randomly generated as in conventional ELM. Derivatives of SBL mainly vary in the hyperprior distribution $p(\alpha_k)$, among which Gamma distribution is commonly assumed as the learning strategy. A classic instance of such algorithm is relevance vector machine (RVM) [21] which is the Bayesian approach for SVM [9]. With w_k tuned to zeros, the corresponding hidden neurons are also pruned, leading to a sparse representation of ELM model with fast execution time. Hence, SBELM has the advantages of both ELM (universal approximation and efficient learning speed) and SBL (high generalization and sparsity).

As a summary, the contribution of our paper is the integration of SBELM with face detection techniques for real-time face detection without compromise of detection accuracy. The paper is organized as follows. Section 2 presents the brief of ELM and the details of SBELM. Section 3 describes the experimental setup of the application such as benchmark dataset, feature extraction, parameters setup in different methods, and experimental environment. The experimental results of accuracy, training time, execution time, and model size followed by analysis and discussion are given in Sect. 4. Finally, a conclusion is drawn in Sect. 5.

2 Sparse Bayesian extreme learning machine (SBELM)

2.1 Extreme learning machine for classification

Given a set of N training dataset $\mathcal{D} = (x_i, t_i)$, $i = 1$ to N with $\mathbf{x}_i$ a d dimensional vector and the observed output $t_i\{0, 1\}$. The output function of ELM with L hidden neurons is represented by

$$f(\mathbf{x}) = sign\left(\sum_{k=1}^{L} w_k h_k(\boldsymbol{\theta}_k; \mathbf{x})\right) = sign(\mathbf{h}(\boldsymbol{\Theta}; \mathbf{x})\mathbf{w}) \tag{1}$$

where $\mathbf{h}(\boldsymbol{\Theta}; \mathbf{x}) = [h_1(\boldsymbol{\theta}_1; \mathbf{x}), \ldots, h_L(\boldsymbol{\theta}_L; \mathbf{x})]$ is the hidden feature mapping with respect to input $\mathbf{x}$. The parameters of hidden layer $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L]$ are randomly generated, and $\mathbf{w}$ is the weight vector of all hidden neurons to an output neuron. $h_k(\cdot)$ is the activation function of hidden layer. The inner part of (1) can be written as

$$\mathbf{HW} = \mathbf{T} \tag{2}$$

where $\mathbf{H}$ is the $N \times L$ hidden layer feature-mapping mat, whose elements are as follows:

$$\mathbf{H} = \begin{bmatrix} h_1(\boldsymbol{\theta}_1; \mathbf{x}_1) & \ldots & h_L(\boldsymbol{\theta}_L; \mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\boldsymbol{\theta}_1; \mathbf{x}_N) & \cdots & h_L(\boldsymbol{\theta}_L; \mathbf{x}_N) \end{bmatrix}. \tag{3}$$

the ith row of $\mathbf{H}$ is the hidden layer's output vector for an instance $\mathbf{x}$, and (2) is a linear system, which is solved by

$$\mathbf{W} = \mathbf{H}^{\dagger}\mathbf{T}, \mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T \tag{4}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse [22] of matrix $\mathbf{H}$. A modified version of (4) is employed by adding a regulator value $1/\lambda$ to the diagonal of $\mathbf{H}^T\mathbf{H}$ in L2-regularized ELM.

2.2 SLM for classification

SBELM learns the output weights $\mathbf{W}$ by Bayesian mechanism, rather than the analytic solution in (4). Although the details and derivation of SBELM classification can be found in our previous work [18], it is still described here to make the paper self-contained. In SBELM, the output weight $\mathbf{W}$ is learned by Bayesian method rather than analytically calculating $\mathbf{H}^{\dagger}$. The input for SBELM is the hidden layer outputs $\mathbf{H} = [\mathbf{h}_1(\boldsymbol{\Theta}; \mathbf{x}_1), \ldots, \mathbf{h}_N(\boldsymbol{\Theta}; \mathbf{x}_N)]^T \in \mathcal{R}^{N\times L}$ with $\mathbf{h}_i(\boldsymbol{\Theta}; \mathbf{x}_i) = [h_1(\boldsymbol{\theta}_1; \mathbf{x}_i), \ldots, h_L(\boldsymbol{\theta}_L; \mathbf{x}_i)], i = 1\ldots N$. Since fact detection is a binary classification problem, every training data can be considered as an independent Bernoulli event, i.e., $p(t|\mathbf{x})$ is a Bernoulli distribution whose likelihood is expressed as

$$p(\mathbf{t}|\mathbf{w}, \mathbf{h}) = \sigma\{\mathcal{Y}(\mathbf{h}; \mathbf{w})\}^t[1 - \sigma\{\mathcal{Y}(\mathbf{h}; \mathbf{w})\}]^{1-t} \tag{5}$$

where $\sigma[\mathcal{Y}(\mathbf{h}; \mathbf{w})] = \frac{1}{1+e^{-\mathcal{Y}(\mathbf{h};\mathbf{w})}}$, $\mathcal{Y}(\mathbf{h}; \mathbf{w}) = \mathbf{hw}$, $\mathbf{t} = (t_1\ldots t_N)^T$, $t_i \in \{0, 1\}$, and $\mathbf{w} = (w_0\ldots w_L)^T$. A zero-mean Gaussian prior distribution over each parameter w_k conditions on an ARD hyperparameter α_k [19, 20] is given by

$$p(w_k|\alpha_k) = \mathcal{N}(w_k|0, \alpha_k^{-1}) \tag{6}$$

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{k=0}^{L} \frac{\alpha_k}{\sqrt{2\pi}} \exp\left(-\frac{\alpha_k w_k^2}{2}\right) \tag{7}$$

where $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_L]^T$. Importantly, there always exists an independent α_k associated with each w_k. The ARD prior selects the significant hidden neurons by controlling some values of w_k's to zero. Next step is to marginalize the likelihood over $\mathbf{t}$ conditioned on $\boldsymbol{\alpha}$ and$\mathbf{H}$. The values of $\boldsymbol{\alpha}$ are determined by maximizing the marginal likelihood by integrating the weight parameters $\mathbf{w}$

$$p(\mathbf{t}|\boldsymbol{\alpha}, \mathbf{H}) = \int p(\mathbf{t}|\mathbf{w}, \mathbf{H}) p(\mathbf{w}|\boldsymbol{\alpha}) \mathrm{d}\mathbf{w} \tag{8}$$

The core learning procedure involves establishing the distribution of $p(\mathbf{t}|\boldsymbol{\alpha}, \mathbf{H})$ to determine $\boldsymbol{\alpha}$ by maximizing the marginal likelihood. However, the integral of (8) is intractable; ARD approximates a Gauss for it with Laplace approximation approach, which is achieved by evaluating a quadratic Taylor expansion of log-posterior function. The mean and covariance of the approximated Gauss are the Laplace's mode and negated inversed second-derivative (Hessian matrix) respectively. Thus,

$$\begin{aligned} \ln\{p(\mathbf{t}|\mathbf{w}, \mathbf{H}) p(\mathbf{w}|\boldsymbol{\alpha})\} &= \ln\left\{\prod_{i=1}^{N} y_i^{t_i} (1 - y_i)^{1 - t_i}\right\} \\ &\quad + \ln\left\{\prod_{k=0}^{L} \frac{\alpha_k}{\sqrt{2\pi}} \exp\left(-\frac{\alpha_k w_k^2}{2}\right)\right\} \\ &= \sum_{i=1}^{N} \{t_i \ln y_i + (1 - t_i) \ln(1 - y_i)\} \\ &\quad - \frac{1}{2} \mathbf{w}^{\mathbf{T}} \mathbf{A} \mathbf{w} + const \end{aligned} \tag{9}$$

where $y_i = \sigma\{\mathcal{Y}(\mathbf{h}_i; \mathbf{w})\}$ and $\mathbf{A} = \mathrm{diag}(\boldsymbol{\alpha})$. Here, we let $const = \sum_{k=0}^{L} \left(\ln \alpha_k - \frac{1}{2} \ln 2\pi\right)$ which is not associated with $\mathbf{w}$. Generally, it is efficient to find the Laplace's mode $\widehat{\mathbf{w}}$ using the Newton–Raphson method *iterative reweighted least squares* (*IRLS*). Given (9), its gradient $\nabla\mathbf{E}$ and Hessian matrix Φ need to be figured out.

$$\begin{aligned} \nabla\mathbf{E} &= \nabla_{\mathbf{w}} \ln\{p(\mathbf{t}|\mathbf{w}, \mathbf{H}) p(\mathbf{w}|\boldsymbol{\alpha})\} = \sum_{i=1}^{N} (t_i - y_i) \mathbf{h}_i - \mathbf{A}\mathbf{w} \\ &= \mathbf{H}^{\mathrm{T}} (\mathbf{t} - \mathbf{y}) - \mathbf{A}\mathbf{w} \end{aligned} \tag{10}$$

$$\Phi = \nabla_{\mathbf{w}} \nabla_{\mathbf{w}} \ln\{p(\mathbf{t}|\mathbf{w}, \mathbf{H}) p(\mathbf{w}|\boldsymbol{\alpha})\} = -(\mathbf{H}^{\mathrm{T}} \mathbf{B} \mathbf{H} + \mathbf{A}) \tag{11}$$

where $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$, $\mathbf{B}$ is an $N \times N$ diagonal matrix with element $\beta_i = y_i(1 - y_i)$ and $\frac{\mathrm{d}\sigma}{\mathrm{d}\mathcal{Y}} = \sigma(1 - \sigma)$. Therefore, by utilizing *IRLS,* $\widehat{\mathbf{w}}$ is obtained by

$$\mathbf{w}_{\mathrm{new}} = \mathbf{w}_{\mathrm{old}} - \Phi^{-1} \nabla\mathbf{E} = (\mathbf{H}^{\mathrm{T}} \mathbf{B} \mathbf{H} + \mathbf{A})^{-1} \mathbf{H}^{\mathrm{T}} \mathbf{B} \widehat{\mathbf{t}} \tag{12}$$

where $\widehat{\mathbf{t}} = \mathbf{H}\mathbf{w} + \mathbf{B}^{-1}(\mathbf{t} - \mathbf{y})$. The center $\widehat{\mathbf{w}}$ and covariance matrix $\boldsymbol{\Sigma}$ of Gauss distribution over $\mathbf{w}$ by Laplace approximation are

$$\widehat{\mathbf{w}} = \Sigma \mathbf{H}^{\mathrm{T}} \mathbf{B} \widehat{\mathbf{t}} \quad \text{and} \quad \boldsymbol{\Sigma} = (\mathbf{H}^{\mathrm{T}} \mathbf{B} \mathbf{H} + \mathbf{A})^{-1} \tag{13}$$

Therefore, we get $p(\mathbf{t}|\mathbf{w}, \mathbf{H}) p(\mathbf{w}|\boldsymbol{\alpha}) \propto \mathcal{N}(\widehat{\mathbf{w}}, \boldsymbol{\Sigma})$. After gaining Gaussian approximation for $\mathbf{w}$, the integral of product of the two prior probability functions of (8) becomes tractable. The log marginal likelihood is as follows:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}) = \ln p(\mathbf{t}|\boldsymbol{\alpha}, \mathbf{H}) &= -\frac{1}{2} \Big[N \ln(2\pi) + \ln\left|\mathbf{B} + \mathbf{H}\mathbf{A}\mathbf{H}^{\mathrm{T}}\right| \\ &\quad + (\widehat{\mathbf{t}})^T (\mathbf{B} + \mathbf{H}\mathbf{A}\mathbf{H}^{\mathrm{T}})^{-1} \widehat{\mathbf{t}}\Big] \\ &= -\frac{1}{2} \left[N \ln(2\pi) + \ln|\mathbf{C}| + \left(\widehat{\mathbf{t}}\right)^T \mathbf{C}^{-1} \widehat{\mathbf{t}}\right] \end{aligned} \tag{14}$$

where $\mathbf{C} = \mathbf{B} + \mathbf{H}\mathbf{A}\mathbf{H}^{\mathrm{T}}$. Set the differential of $\mathcal{L}(\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$ to zero,

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha})}{\partial \alpha_k} = \frac{1}{2\alpha_k} - \frac{1}{2} \Sigma_{kk} - \frac{1}{2} \widehat{\mathrm{w}}_k^2 = 0 \Rightarrow \alpha_k^{\mathrm{new}} = \frac{1 - \alpha_k \Sigma_{kk}}{\widehat{\mathrm{w}}_k^2} \tag{15}$$

By setting w_k and α_k with initial values, $\widehat{\mathbf{w}}$ and $\boldsymbol{\Sigma}$ are updated from (13). By using these two values, $\boldsymbol{\alpha}$ is updated through (15); again, the operation continues to maximize the marginal likelihood function until reaching the convergence criteria (e.g., when the difference between the maximum α_k (empirically, $\log(\alpha_k)$ is preferred) in two successive iterations is lower than a predefined accuracy, or the maximum number of iterations). After $\widehat{\mathbf{w}}$ converges, then $\mathcal{Y}(\mathbf{h}; \widehat{\mathbf{w}}) = \mathbf{h}\widehat{\mathbf{w}}$ and the final decision is given by $\sigma[\mathcal{Y}(\mathbf{h}; \widehat{\mathbf{w}})] = \left(1 + e^{-\mathcal{Y}(\mathbf{h}; \widehat{\mathbf{w}})}\right)^{-1}$.

2.3 Algorithm and practical guideline of SBELM

In this subsection, some practical guidelines for running SBELM classification is discussed, whose MATLAB toolbox is available at our Web page [23]. The algorithm of SBELM for classification is summarized at Algorithm 1. From practical experimental trials, the input weights $\boldsymbol{\Theta}_{N \times L} = [\boldsymbol{\theta}_1^T; \ldots; \boldsymbol{\theta}_L^T]$ are randomly generated but are strongly advised to be within interval $[-1, 1]$ under uniform distribution, N is the number of training data and L is the number of hidden neurons. The activation function for the hidden layer is usually sigmoid function $h_k(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}_k^T \mathbf{x}}}$ or RBF function $h_k(\mathbf{x}) = \exp\left(-\boldsymbol{\theta}_k^T \mathbf{x}\right)$. Newton method is described in Step 1 to minimize the negative convex optimization by keeping the hyperprior $\boldsymbol{\alpha}$ fixed in the whole step. From our previous results [18], SBELM can usually achieve best performance with $L < 200$ for most applications.

Algorithm 1. SBELM for Classification

```
**Initialization:**
Given initial number of hidden nodes $L$ (Usually $L < 200$)
Randomly generates input weights $\boldsymbol{\Theta}_{N\times L} = [\boldsymbol{\theta}_1^T; \cdots; \boldsymbol{\theta}_L^T]$
Calculate the outputs of hidden layer $\mathbf{H}(\boldsymbol{\Theta};\mathbf{x}) = [1, \mathbf{h}_1(\boldsymbol{\theta}_1;\mathbf{x}), \cdots, \mathbf{h}_L(\boldsymbol{\theta}_L;\mathbf{x})]$
$\mathbf{w} = [\mathbf{0}]_{L\times 1}$, $\boldsymbol{\alpha} = 10^{-5} * [\mathbf{1}]_{L\times 1}$
**Step 1:** Estimation of output weights $\mathbf{w}$
1. $\boldsymbol{\Sigma} = [\mathbf{0}]_{L\times L}$;   //Hessian matrix
2. $\mathbf{g} = [\mathbf{0}]_{L\times 1}$;   // Gradient
3. *While* $i = 1$ *to* $N$
// 3.1. Sequentially calculate the mapping of every input $\mathbf{x}_i$ to $\mathbf{h}(\mathbf{x}_i)$ with random
      ELM hidden weights.
    3.1. $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma} + y_i(1 - y_i) * \mathbf{h}(\mathbf{x}_i)^T \mathbf{h}(\mathbf{x}_i)$;
    3.2. $\mathbf{g} \leftarrow \mathbf{g} + (-1) * (t_i - y_i) * \mathbf{h}(\mathbf{x}_i)^T$;
*End while*
4. $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma} + \mathrm{diag}(\boldsymbol{\alpha})$;
5. $\mathbf{g} \leftarrow \mathbf{g} + \boldsymbol{\alpha} \bullet \mathbf{w}$;   // • denotes dot product.
6. Conduct the inversion of $\boldsymbol{\Sigma}$;
7. Find stepsize $\lambda$ with linesearch method.
8. $\mathbf{w} \leftarrow \mathbf{w} - \lambda \boldsymbol{\Sigma}^{-1}\mathbf{g}$;
9. If norm($\mathbf{g}$) is under a pre-defined gradient tolerence, then go to step 2.
Otherwise, go to step 1.
**Step 2:** Estimation of hyper parameter $\boldsymbol{\alpha}$.
*For every* $\alpha_k$,
10. $\alpha_k \leftarrow (1 - \alpha_k \Sigma^{-1}{}_{kk.})/w_k^2$;  //$\Sigma^{-1}{}_{kk}$ is the $k$th diagonal element of $\boldsymbol{\Sigma}^{-1}$
*End for*;
**Step 3:** Pruning nodes
*For every* $\alpha_k$,
11. If $\alpha_k >$ predefined maximum
      prune $\alpha_k$, $h_k(\mathbf{x})$, $w_k$;  $L \leftarrow L - 1$;
12. If the absolute difference between two successive logarithm values of $\alpha_k$ is
lower given Tolerance, then stop. Otherwise, repeat Step 1 to Step 3.
```

3 Experiments

3.1 Benchmark face databases

The experiments involve four benchmark face databases: CMU [24], FERET [25], MIT [26], and ORL [27]. The properties of these databases are shown in Table 1. The databases are randomly divided into two parts for training and testing, in which training data accounted for 70 %, while testing data accounted for 30 % in every face database. For example, in CMU, there are 925 images of which 404 are faces and 521 are non-face as shown in Fig. 1. Then, the number of training data for CMU $= 404 \times 70\ \% + 521 \times 70\ \% = 283 + 365 = 658$. The remaining 267 images constitute the test data.

Table 1 Description of four benchmarking face databases

Face database/number	Total images	Face image	Non-face image	Image size
CMU	925	404	521	64 × 64
FERET	933	301	632	80 × 80
MIT	7,087	2,706	4,381	20 × 20
ORL	360	160	200	92 × 112

3.2 Feature extraction

Although the raw images can be trained directly, it is preferable to first extract important features from the raw images because the dimensionality and the training time can be greatly reduced. For example, in ORL, an image of 100 × 100 can be reduced to a feature vector of about 200 only. Therefore, prior to training the classifiers, the training data are preprocessed using feature extraction method. Currently there are four popular methods in the literature, namely *principal components analysis* (PCA) [28], *scale-invariant feature transform* (SIFT) [29], PCA-SIFT [30], and *Speed Up Robust Features* (SURF) [31]. However, SIFT and SURF suffer from both long computation time and irregular number of extracted features for different images. Although PCA-SIFT can resolve the issue of irregular number of extracted features, the long computation time is even more severe. Under the objective of constructing a real-time classifier, only PCA is chosen for the experiments. After extracting features using these methods, four neural network approaches (SVM, ELM, K-ELM, and SBELM) are applied to construct the classifiers, respectively.

3.3 Parameters setup

There are four different training methods in this experiment. All of the training methods employ sigmoid as kernel function, or activation function. SVM takes a regularization parameter C and sigmoid kernel parameter γ, whose values are by default set to 1.0. Although optimal parameters are preferred, it is insignificant in this experiment because our objective is a comparison of algorithms over model execution time. In fact, there is no much improvement (up to 1–3 %) on the accuracy with exhaustive search on the optimal parameters. Similarly, K-ELM takes the same set of parameters C and γ, and they are set to 1.0 by default as well. For ELM and SBELM, no kernel is employed in the model but a simple number of hidden neurons are necessary. However, ELM has a relatively unstable accuracy over the number of hidden neurons. Usually, the user needs to train an ELM model through a sequence of guesses of number of hidden neurons, e.g., from 10 to 300 with interval 10. However, SBELM does not have this restriction because SBELM's accuracy is relatively stable over the number of hidden neurons, i.e., the accuracy is not much different between 200 hidden neurons and 300 hidden neurons. Therefore, an upper bound of 200 hidden neurons can be simply assumed.

3.4 Experimental environment and evaluation

The experiments run on a workstation with a 6-core Intel Xeon 3.0 GHz, 8 GB RAM. PCA and SBELM

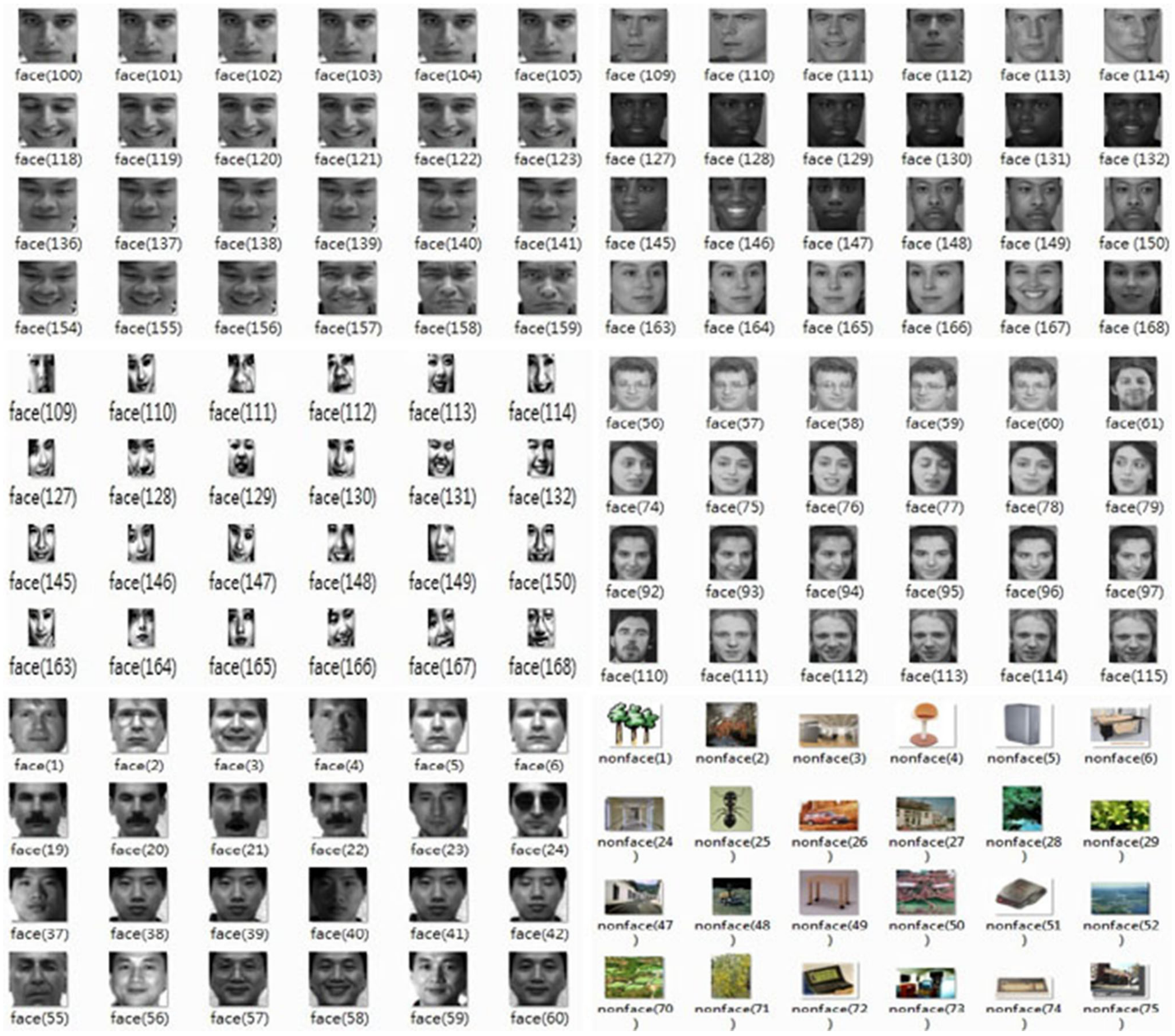

Fig. 1 A sample of face and non-face images

were implemented in MATLAB platform. The MATLAB toolboxes [32–34] for SVM, ELM, and K-ELM are available online in their respective websites. The evaluation includes several aspects of accuracy, training time, and execution time. In fact, execution time can already reflect the model size, and hence, it is not included in the evaluation. For the sake of fair comparison, every training method was run for ten times. Every time, a new pair of training set (70 %) and test set (30 %) was extracted from the face databases according to the way described in Sect. 3.1. From the ten runs, the mean and standard deviation of accuracy, training, and execution for different training methods are produced for comparison.

4 Results and discussion

The face detection results are evaluated over four benchmark face databases CMU, FERET, MIT, and ORL under different aspects. Every result is obtained by running ten times with different randomly drawn training and test sets, as discussed in Sect. 3.1.

4.1 Accuracy

The accuracies of the four learning methods over the benchmark face databases are shown in Table 2. The bold figures in Tables 2, 3, 4, 5 represent the best performance (highest accuracy, or shortest time) among the four compared methods

Table 2 Accuracy (%) over four face databases

	CMU	FERET	MIT	ORL
SVM	87.91 ± 0.94	82.47 ± 0.06	72.82 ± 0.60	88.26 ± 0.54
ELM[a]	97.25 ± 1.27	86.79 ± 2.16	**79.87 ± 1.21**	94.50 ± 2.27
K-ELM	**99.99 ± 0.01**	**92.79 ± 0.58**	79.47 ± 0.98	**98.12 ± 0.84**
SBELM[a]	99.19 ± 1.65	89.84 ± 2.28	78.82 ± 1.67	96.90 ± 1.59

[a] ELM and SBELM are trained using 200 hidden neurons

Table 3 Training time (ms) over four face databases (training set only)

	CMU	FERET	MIT	ORL
SVM	135.29 ± 5.36	86.02 ± 4.61	783.06 ± 46.42	143.69 ± 5.72
ELM[a]	269.89 ± 26.58	71.13 ± 15.86	**129.16 ± 9.16**	78.87 ± 8.92
K-ELM	**122.79 ± 149.25**	**13.20 ± 5.25**	147.40 ± 19.23	**22.96 ± 5.33**
SBELM[a]	3,809.80 ± 471.08	555.4 ± 53.14	1,492.8 ± 230.37	599.5 ± 24.61

[a] ELM and SBELM are trained using 200 hidden neurons

Table 4 Average execution time (ms) over four face databases (test set only)

	CMU	FERET	MIT	ORL
SVM	25.26 ± 2.30	22.24 ± 2.39	55.50 ± 2.43	27.58 ± 2.57
ELM[a]	1.52 ± 0.10	2.38 ± 0.97	1.13 ± 0.19	1.51 ± 0.44
K-ELM	3.50 ± 0.66	2.38 ± 0.97	6.42 ± 0.54	2.89 ± 0.61
SBELM[a]	**0.54 ± 0.03**	**0.50 ± 0.05**	**0.60 ± 0.07**	**0.70 ± 0.25**

[a] ELM and SBELM are trained using 200 hidden neurons

(SVM, ELM, K-ELM, and SBELM). It can be seen that kernel ELM mostly outperforms other methods with very low deviation. This may be explained that ELM searches in a large space for optimal parameters, but SVM searches from a linear plane [11]. Hence, ELM is able to find out a better solution than SVM. K-ELM inherits this property and also possesses the ability to map nonlinear input space into a high-dimensional feature space through kernel mapping, leading to an even better result as shown in Table 2. For the proposed SBELM, it is the second best among the four learning methods that is only worse than K-ELM. However, the objective of SBELM is the minimal target model size (i.e., shortest execution time) as explained later in Sect. 4.3.

4.2 Training time

From Table 3, K-ELM takes shortest training time among all methods; meanwhile, SBELM takes longest training time. It is because SBELM approximates the set of output weights iteratively that is very time-consuming. On the contrary, SVM, ELM, and K-ELM take analytical methods to directly calculate the set of output weights. Although SBELM is time-consuming, it takes only 0.5–3.8 s for the four databases, which is definitely acceptable. As a trade-off, the execution time of SBELM model is shortest among all methods, which is discussed in Sect. 4.3.

4.3 Execution time

The execution times of all models are shown in Table 4. From the results, SBELM outperforms all other methods and is up to 92 times faster. Since the execution time of SBELM is so short to measure, all execution times are measured over 100 runs, respectively, e.g., SBELM takes an average execution time of 0.54 ms to classify 100 PCA-preprocessed images (whose dimension is 200) from CMU. In other words, SBELM takes 0.0054 ms for a classification which is close to real time. This extremely fast execution time is addressed to the minimal number of hidden neurons returned by SBELM.

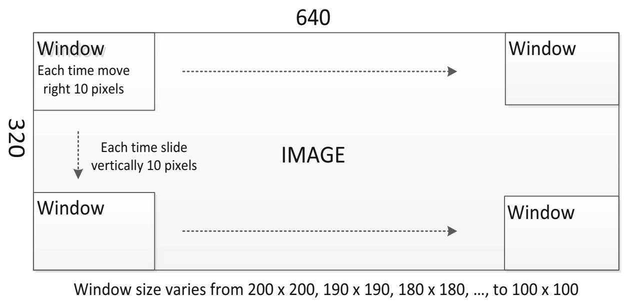
640
320
Window
Each time move right 10 pixels
Window
Each time slide vertically 10 pixels
IMAGE
Window
Window
Window size varies from 200 x 200, 190 x 190, 180 x 180, ..., to 100 x 100

Fig. 2 The intervals and varying sizes of the scanning window

Another comparison is the overall execution time on a real image. In this experiment, ten images of 640 × 360 are employed. The search window varies from 200 × 200, 190 × 190, 180 × 180, …, down to 100 × 100. The sliding interval is ten pixels horizontally and vertically, respectively, as shown in Fig. 2.

Therefore, to detect faces from a real image, it takes about 9,500 times of PCA feature extraction and

Table 5 Performance in time over ten real images of 640 × 360

	SVM	ELM	K-ELM	SB-ELM
Average classification time (ms)	2,306.42 ± 1.66	371.53 ± 0.36	983.71 ± 0.66	**207.45 ± 0.14**
Difference from shortest time (%)	1,014 %	80 %	375 %	0 %

classification. Since the comparison concentrates on the classification, the time for PCA extraction is not included. The corresponding results are shown in Table 5. The time shows the average classification time for one complete image of 640 × 360 ranges from about 20 to 230 ms. Again, SBELM outperforms other methods up to ten times. The time difference in percentage of the different methods is also presented for reference. The calculation is (current time − shortest time)/shortest time.

4.4 Model size

The model sizes of SVM and K-ELM depend on the number of training data to construct the classification models. On the contrary, the model sizes of ELM and SBELM depend on the number of hidden neurons. For large datasets, the model sizes of SVM and K-ELM become large, and hence, the classification time is long (as shown in Tables 4, 5). That explains why ELM and SBELM can run much faster. However, SBELM can produce a sparse representation of the hidden neurons so that the classification model is compact (Fig. 3). For example, for CMU, when 200 input hidden neurons are guessed, 28 nonzero hidden neurons are finally obtained which produces a 1:7 compression ratio in model size. At the same time, we can see that when the number of input hidden neurons $L > 100$, the number of resultant nonzero hidden neurons in the classification model remains in the same magnitude, while the corresponding accuracies do not vary too much (Fig. 4). Therefore, it provides a general heuristic of 200 input hidden neurons in SBELM training phase that can further reduce the tuning work of number of hidden neurons.

5 Conclusion

Face detection is a difficult and time-consuming problem because a face can be of different size in an image. Hence, different window sizes are employed to scan through an image in order to detect any face. From our experiments, about 9,500 times of scanning and classification are necessary for face detection on an image of moderate size (e.g., 640 × 360). Clearly, classification plays an important role for fast and accurate face detection. In this paper, SBELM is employed to minimize the number of hidden neurons in a model in order to provide a fast classifier for

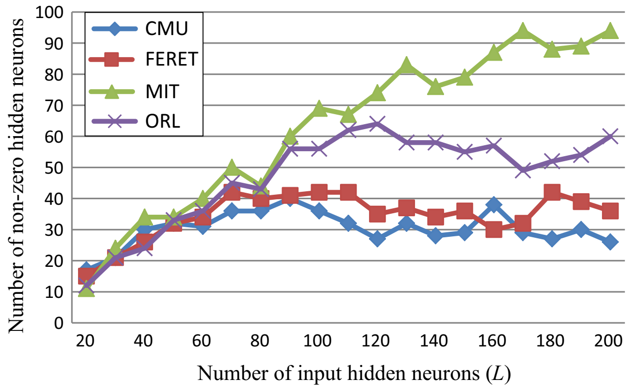

Fig. 3 The numbers of non-zero hidden neurons by SBELM over benchmark face databases

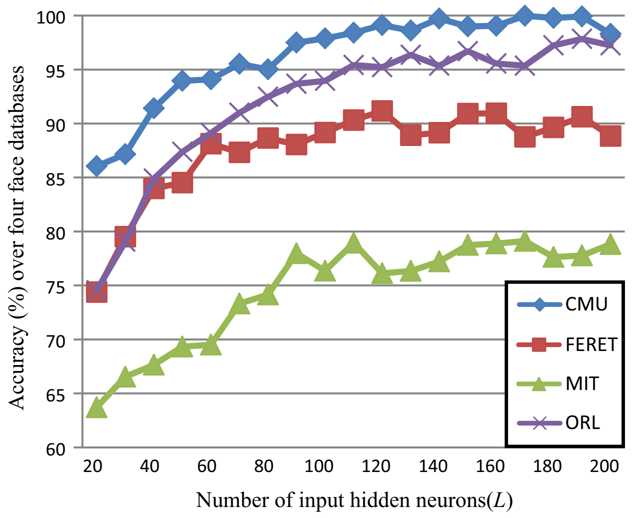

Fig. 4 The accuracy of input hidden neurons by SBELM over benchmark face database

real-time face detection with nearly no compromise in accuracy. SBELM is compared to three other state-of-the-art methods, namely SVM, ELM, and K-ELM in different performance aspects. Experimental results show that SBELM is the second best among these methods in accuracy, while SBELM takes the shortest classification time (up to ten times faster than SVM). Although SBELM takes longer training time than other methods, it is only one-off and insignificant to real-time face detection. In the future, we will further improve the accuracy and training speed of SBELM.

Acknowledgments The work is supported by Fundo para o Desenvolvimento das Ciencias e da Tecnologia (Grant No. FDCT/075/

2013/A) and University of Macau (Grant No. MYRG075(Y2-L2)-FST13-VCM).

References

1. Yilmaz A, Javed O, Shah M (2006) Object tracking: a survey. ACM Comput Surv 38:13
2. Hu W, Tan T, Wang L, Maybank S (2004) A survey on visual surveillance of object motion and behaviors. IEEE Trans Syst Man Cybern Part C Appl Rev 34:334
3. Ban Y, Kim SK, Kim S, Toh KA, Lee S (2014) Face detection based on skin color likelihood. Pattern Recogn 47:1573–1585
4. Wang L, Ding X, Fang C (2009) Face live detection method based on physiological motion analysis. Tsinghua Sci Technol 14:685–690
5. Sun J (2012) Face tracking based on skin color and motion prediction. In: Proceedings of the 2012 international conference on computer application and system modeling, pp 0657–0660
6. Fröba B, Küblbeck C (2001) Face detection and tracking using edge orientation information. In: Proceedings of SPIE—the international society for optical engineering, pp 583–594
7. Jesorsky O, Kirchberg KJ, Frischholz RW (2001) Robust face detection using the Hausdorff distance. In: LNCS vol 2091, third international conference on audio- and video- based biometric person authentication. Springer, Berlin, pp 90–95
8. Shilbayeh NF, Al-Qudah GA (2010) Face detection system based on MLP neural network. In: Proceedings of the 11th WSEAS international conference on neural networks, pp 238–243
9. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
10. Liu R, Zhang J, Wang L, Zhang M (2014) Research on face recognition method based on combination of SVM and LDA-PCA. In: Second international conference on communications, pp 101–109
11. Zong W, Huang GB (2011) Face recognition based on extreme learning machine. Neurocomputing 74:2541–2551
12. Huang GB, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybernet 2(2):107–122
13. Huang G-B, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B Cybern 42:513–529
14. Zong W, Zhou H, Huang GB, Lin Z (2011) Face recognition based on kernelized extreme learning machine. In: LNCS vol 6752, second international conference, AIS 2011. Springer, Berlin, pp 263–272
15. Wang XZ, Chen A, Feng HM (2011) Upper integral network with extreme learning mechanism. Neurocomputing 74(16): 2520–2525
16. Liu P, Huang YH, Lei M, Gong SY, Zhang GP (2014) Two-stage extreme learning machine for high-dimensional data. Int J Mach Learn Cybernet. doi:10.1007/s13042-014-0292-7
17. Fu AM, Wang XZ, He YL, Wang LS (2014) A study on residence error of training an extreme learning machine and its application to evolutionary algorithms. Neurocomputing 146:75–82
18. Luo J, Vong CM, Wong PK (2013) Sparse Bayesian extreme learning machine for multi-classification. IEEE Trans Neural Netw Learn Syst 25:836–843
19. Bishop CM (ed) (2006) Pattern recognition and machine learning. Springer, New York
20. MacKay DJC (1996) Bayesian methods for backpropagation networks. Model Neural Netw III 6:211–254
21. Tipping ME (2001) Sparse Bayesian learning and the relevance vector machine. Mach Learn Res 1:211–244
22. Banerjee KS (1973) Generalized inverse of matrices and its applications. Technometrics 15:197
23. SBELM Toolbox. http://www.fst.umac.mo/en/staff/fstcmv.html#software
24. CMU Faces Dataset. http://vasc.ri.cmu.edu//idb/html/face/frontal_images/index.html
25. FERET faces dataset. http://www.nist.gov/itl/iad/ig/colorferet.cfm
26. MIT faces dataset. http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html
27. ORL (AT&T) faces dataset. http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
28. Meher SS, Maben P (2014) Face recognition and facial expression identification using PCA. In: Souvenir of the 2014 IEEE international advance computing conference, IACC 2014, pp 1093–1098
29. Geng C, Jiang X (2009) Face recognition using sift features. In: Proceedings on international conference on image processing, ICIP, 2009, pp 3313–3316
30. Kamencay P, Breznan M, Jelsovka D, Zachariasova M (2012) Improved face recognition method based on segmentation algorithm using SIFT-PCA. In: 2012 35th international conference on telecommunications and signal processing, TSP 2012—proceedings, pp 758–762
31. Li J, Wang T, Zhang Y (2011) Face detection using SURF cascade. In Proceedings of the IEEE international conference on computer vision, pp 2183–2190
32. Libsvm Toolbox. http://www.csie.ntu.edu.tw/~cjlin/libsvm/
33. ELM Toolbox. http://www.ntu.edu.sg/home/egbhuang/elm_random_hidden_nodes.html
34. Kernelized ELM Toolbox. http://www.ntu.edu.sg/home/egbhuang/elm_kernel.html