Approximate Kernel Extreme Learning Machine for Large Scale Data Classification

Alexandros Iosifidis, Anastasios Tefas, and Ioannis Pitas

***Abstract*— In this paper, we propose an approximation scheme of the Kernel Extreme Learning Machine algorithm for Single-hidden Layer Feedforward Neural network training that can be used for large scale classification problems. The Approximate Kernel Extreme Learning Machine is able to scale well in both computational cost and memory, while achieving good generalization performance. Regularized versions and extensions in order to exploit the total and within-class variance of the training data in the feature space are also proposed. Extensive experimental evaluation in medium-scale and large-scale classification problems denotes that the proposed approach is able to operate extremely fast in both the training and test phases and to provide satisfactory performance, outperforming relating classification schemes.**

***Index Terms*— Extreme Learning Machine, Large Scale Learning, Facial Image Classification.**

I. Introduction

Extreme Learning Machine (ELM) is a fast algorithm for Single-hidden Layer Feedforward Neural (SLFN) networks training that requires low human supervision [1]. Conventional SLFN network training approaches, like the Backpropagation [2] and the Levenberg-Marquardt [3] algorithms, adjust the input weights and the hidden layer bias values by following an optimization process, e.g., by applying gradient descend-based optimization. However, such learning techniques are generally slow and may decrease the generalization ability of the network, since the solution may be trapped in local minima. In standard ELM approaches the input weights and the hidden layer bias values of the SLFN network are randomly assigned and the network output weights are, subsequently, analytically calculated. ELMs not only tend to reach the smallest training error, but also the smallest output weight norm as well. For feedforward networks reaching a small training error, smaller output weight norm results in better generalization performance [4]. Despite the fact that the determination of the network hidden layer outputs is based on randomly assigned input weights, it has been proven that ELM networks have the properties of global approximators [5], [6]. Due to its effectiveness and its fast learning process, the ELM network has been adopted in many classification problems and many ELM variants have been proposed in the last few years, extending the ELM network properties along different directions [7], [8], [9], [10], [11], [12], [13].

A. Iosifidis is with the Department of Signal Processing, Tampere University of Technology, Finland and the Department of Informatics, Aristotle University of Thessaloniki, Greece. e-mail: alexandros.iosifidis@tut.fi
A. Tefas is with the Department of Informatics, Aristotle University of Thessaloniki, Greece. e-mail: tefas@aiia.csd.auth.gr
I. Pitas is with the Department of Electrical and Electronic Engineering, University of Bristol, UK and the Department of Informatics, Aristotle University of Thessaloniki, Greece.

Recently, kernel versions of the ELM algorithm have been proposed [14], [15], which have been shown to outperform the standard ELM approach that uses random input parameters [16]. A possible explanation of this fact is that kernel ELM networks have connections to infinite single-hidden layer feedforward networks [16]. However, the superiority in performance of kernel ELM formulations is accompanied by a higher computation cost and memory requirements, rendering the exploitation of kernel ELM networks prohibitive in large scale classification problems. Specifically, for a dataset consisting of N training data, kernel ELM approaches require the calculation of a matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$ having a quadratic $O(N^2)$ computational complexity with respect to N. The calculation of the network parameters requires the inversion of $\mathbf{K}$, having a cubic $O(N^3)$ computational complexity with respect to N [14], [16]. In order to make the application of ELM-based classification in large scale classification problems possible, an SMO-based optimization algorithm has been proposed in [15] for the case where the hinge loss of the prediction error is exploited. This method has the drawbacks that it still requires the calculation of the entire kernel matrix $\mathbf{K}$ and that the optimization process is still very slow for large scale datasets. For squared loss criteria, the regularized ELM approach exploiting random hidden layer parameters has been proposed in [14]. It has the disadvantage that, by employing randomly sampled hidden layer parameters, the obtained performance is inferior, when compared to the kernel approach [16].

Approximation approaches have been found to be both efficient and effective. A line of work in approximate methods determines a low-rank approximation of a Gram matrix of the form $\mathbf{K} \simeq \mathbf{Q} = \mathbf{C}\tilde{\mathbf{Q}}\mathbf{C}^T$, where $\mathbf{C} \in \mathbb{R}^{N \times n}$ and $\tilde{\mathbf{Q}} \in \mathbb{R}^{n \times n}$. $\mathbf{C}$ is formed by n (uniformly or data-dependent nonuniformly sampled) columns of $\mathbf{K}$ and $\tilde{\mathbf{Q}}$ is a matrix formed by the intersection between those n columns of $\mathbf{K}$ and the corresponding n rows of $\mathbf{K}$ [17]. By using such matrix approximation approaches, approximate Linear Algebra methods, like matrix multiplication and Singular Value Decomposition, and their application in kernel machines have been proposed that have provable guarantees on the quality of obtained approximate solution [18], [19], [20], [21], [17]. In addition, it has been recently shown that the adoption of such an approximate approach can be exploited in kernel-based clustering [22], [23], [24] with state-of-the-art performance. While the above-described approximate approach has the advantage that the entire kernel matrix needs not be computed, for KELM-based classification the the inversion of the corresponding approximate kernel matrix $\mathbf{Q} \in \mathbb{R}^{N \times N}$ still has a computational complexity equal to $O(N^3)$. Another approximate approach exploits a so-called "randomized kernel", which is constructed

by randomly sampling a small set n of the N training data [20], [25], [26]. This approach has the disadvantage that the corresponding methods exploit information appearing only in the subset of the training data employed for the calculation of the randomized kernel.

In this paper, we propose a novel approximate kernel ELM (noted as AKELM hereafter) formulation. We show that the proposed approach is able to scale well in memory and operates extremely fast, when compared to the kernel ELM approach, while achieving comparable, or even better, performance with that of kernel ELM. Extensions of the proposed AKELM approach exploiting regularization and in order to exploit the total and within-class variance of the training data in the feature space, are also proposed. We evaluate the proposed approach in medium and large scale classification problems, where we compare its performance with that of a) ELM and regularized ELM exploiting random hidden layer parameters [1], [14], b) kernel ELM applied on a subset of the training set and c) kernel ELM applied on the entire training set [14].

The novel contributions of the paper are the following ones:

- A novel approximate kernel ELM algorithm that is able to scale well in memory and operate extremely fast during both training and testing is proposed.
- The proposed Approximate Kernel Extreme Learning Machine (AKELM) algorithm is extended, in order to exploit regularization and the total and within-class variance of the training data in the feature space.

The rest of the paper is organized as follows. In Section II, we provide an overview of the ELM, regularized ELM and kernel ELM algorithms. The proposed AKELM algorithm is described in Section III. Regularized AKELM algorithm and AKELM exploting the total and within-class variance of the training data in the feature space are described in Subsections IV and III-B. Experimental results evaluating the performance of the proposed approach in medium and large scale datasets are described in Section IV. Finally, conclusions are drawn in Section V.

II. Overview of Extreme Learning Machines

In this section, we briefly describe the ELM, regularized ELM and kernel ELM proposed in [1] and [14]. Subsequently, we discuss the kernel ELM method proposed in [14] and compare it with the original kernel definition [27], [28], [29].

Let us denote by $\mathbf{X}$ a set of N vectors $\mathbf{x}_i \in \mathbb{R}^D$ and by $\mathbf{c}$ the corresponding class labels $c_i \in \{1, \ldots, C\}$, which will be used in order to train a SLFN network using the ELM algorithm. An ELM network is essentially a combination of C one-versus-rest classifiers. It consists of D input (equal to the dimensionality of $\mathbf{x}_i$), L hidden and C output (equal to the number of classes involved in the classification problem) neurons. The number of hidden layer neurons is usually selected to be much greater than the number of classes [14], [30], i.e., $L \gg C$. The elements of the network target vectors $\mathbf{t}_i = [t_{i1}, ..., t_{iC}]^T$, each corresponding to a training vector $\mathbf{x}_i$, are set to $t_{ik} = 1$ for vectors belonging to class k, i.e., when $c_i = k$, and to $t_{ik} = -1$ when $c_i \neq k$. In ELM-based approaches, the network input weights $\mathbf{W}_{in} \in \mathbb{R}^{D \times L}$ and the hidden layer bias values $\mathbf{b} \in \mathbb{R}^L$ are randomly assigned, while the network output weights $\mathbf{W}_{out} \in \mathbb{R}^{L \times C}$ are analytically calculated.

Let us denote by $\mathbf{q}_j$, $\mathbf{w}_k$, w_{kj} the j-th column of $\mathbf{W}_{in}$, the k-th row of $\mathbf{W}_{out}$ and the j-th element of $\mathbf{w}_k$, respectively. Given an activation function $\Phi(\cdot)$ for the network hidden layer and using a linear activation function for the network output layer, the response $\mathbf{o}_i = [o_{i1}, \ldots, o_{iC}]^T$ of the network corresponding to $\mathbf{x}_i$ is calculated by:

$$o_{ik} = \sum_{j=1}^{L} w_{kj}\, \Phi(\mathbf{q}_j, b_j, \mathbf{x}_i),\ k = 1, ..., C. \tag{1}$$

It has been shown that almost any nonlinear piecewise continuous activation functions $\Phi(\cdot)$ can be used for the calculation of the network hidden layer outputs, e.g. the sigmoid, sine, Gaussian, hard-limiting, Radial Basis Function (RBF), RBF-χ^2, Fourier series, etc [6], [31], [14], [32]. By storing the network hidden layer outputs $\boldsymbol{\phi}_i \in \mathbb{R}^L$ corresponding to all the training vectors $\mathbf{x}_i$, $i = 1, \ldots, N$ in a matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N]$, or:

$$\boldsymbol{\Phi} = \begin{bmatrix} \Phi(\mathbf{q}_1, b_1, \mathbf{x}_1) & \cdots & \Phi(\mathbf{q}_1, b_1, \mathbf{x}_N) \\ \cdots & \ddots & \cdots \\ \Phi(\mathbf{q}_L, b_L, \mathbf{x}_1) & \cdots & \Phi(\mathbf{q}_L, b_L, \mathbf{x}_N) \end{bmatrix}, \tag{2}$$

equation (1) can be expressed in a matrix form as:

$$\mathbf{O} = \mathbf{W}_{out}^T \boldsymbol{\Phi}, \tag{3}$$

where $\mathbf{O} \in \mathbb{R}^{C \times N}$ is a matrix containing the network responses for all training data $\mathbf{x}_i$.

A. Extreme Learning Machine

ELM assumes zero training error [1], by assuming that $\mathbf{o}_i = \mathbf{t}_i$, $i = 1, \ldots, N$, or in a matrix notation:

$$\mathbf{O} = \mathbf{T}, \tag{4}$$

where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$ is a matrix containing the network target vectors. By using (3), the network output weights $\mathbf{W}_{out}$ can be analytically calculated by:

$$\mathbf{W}_{out} = \boldsymbol{\Phi}^{\dagger}\, \mathbf{T}^T, \tag{5}$$

where $\boldsymbol{\Phi}^{\dagger} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T\right)^{-1} \boldsymbol{\Phi}$ is the generalized pseudo-inverse of $\boldsymbol{\Phi}^T$. After the calculation of the network output weights $\mathbf{W}_{out}$, the network response for a vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T \boldsymbol{\phi}_l, \tag{6}$$

where $\boldsymbol{\phi}_l$ is the network hidden layer output for $\mathbf{x}_i$.

B. Regularized and kernel Extreme Learning Machine

The calculation of the network output weights $\mathbf{W}_{out}$ through (5) is sometimes inaccurate, since the matrix $\boldsymbol{\Phi}\boldsymbol{\Phi}^T$ may be singular. A regularized version of the ELM algorithm that allows small training errors and tries to minimize the norm of the network output weights $\mathbf{W}_{out}$ has been proposed

in [14], where the network output weights are calculated by solving the following optimization problem:

$$\textbf{Minimize:} \quad \mathcal{J}_{RELM} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \quad (7)$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T\boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i, \ i = 1, ..., N, \quad (8)$$

where $\boldsymbol{\xi}_i \in \mathbb{R}^C$ is the error vector corresponding to $\mathbf{x}_i$ and λ is a parameter denoting the importance of the training error in the optimization problem, satisfying $\lambda > 0$. Based on the Karush-Kuhn-Tucker (KKT) theorem [33], the network output weights $\mathbf{W}_{out}$ can be determined by solving the dual optimization problem:

$$\begin{aligned}\mathcal{J}_{D,RELM} &= \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \\ &- \sum_{i=1}^{N}\mathbf{a}_i\left(\mathbf{W}_{out}^T\boldsymbol{\phi}_i - \mathbf{t}_i + \boldsymbol{\xi}_i\right), \end{aligned} \quad (9)$$

which is equivalent to (7). By calculating the derivatives of $\mathcal{J}_{D,RELM}$ with respect to $\mathbf{W}_{out}$, $\boldsymbol{\xi}_i$ and $\mathbf{a}_i$ and setting them equal to zero, the network output weights $\mathbf{W}_{out}$ are obtained by:

$$\mathbf{W}_{out} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\boldsymbol{\Phi}\mathbf{T}^T, \quad (10)$$

or

$$\begin{aligned}\mathbf{W}_{out} &= \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\mathbf{T}^T \\ &= \boldsymbol{\Phi}\left(\mathbf{K} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\mathbf{T}^T, \end{aligned} \quad (11)$$

where $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the *ELM kernel matrix*, having elements equal to $[\mathbf{K}]_{i,j} = \boldsymbol{\phi}_i^T\boldsymbol{\phi}_j$ [34]. By using (11), the network response for a given vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\begin{aligned}\mathbf{o}_l &= \mathbf{W}_{out}^T\boldsymbol{\phi}_l = \mathbf{T}\left(\mathbf{K} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{\phi}_l \\ &= \mathbf{T}\left(\mathbf{K} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\mathbf{k}_l, \end{aligned} \quad (12)$$

where $\mathbf{k}_l \in \mathbb{R}^N$ is a vector having its elements equal to $\mathbf{k}_{l,i} = \boldsymbol{\phi}_i^T\boldsymbol{\phi}_l$, $i = 1, \ldots, N$.

C. Discussion on kernel Extreme Learning Machines

The kernel ELM method described above follows the analysis used in regularized ELM and exploits the kernel matrix definition $\mathbf{K} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$, in order to derive an equation involving $\mathbf{o}_l$ and $\mathbf{K}$ (i.e. Eq. (12)). Since in this analysis $\boldsymbol{\phi}_i \in \mathbb{R}^L$, $\mathbf{K}$ is also defined in $\mathbb{R}^L$. Thus, the network output weights $\mathbf{w}_k$, $k = 1, \ldots, C$ are defined in $\mathbb{R}^L$ too. Kernel formulations define data representations in a feature space $\mathcal{F}$ of arbitrary (even infinite) dimensions, having the properties of Hilbert spaces [27], [28], [29]. Subsequently, they restrict the respective parameters to lie on the span of the training data representations in $\mathcal{F}$. That is, according to the Representer theory following the Mercer conditions, we should restrict the network output weights to be a linear combination of the training data (when represented in $\mathcal{F}$) [35], [36], i.e.:

$$\mathbf{W}_{out} = \boldsymbol{\Phi}\mathbf{A}^T, \quad (13)$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{|\mathcal{F}|\times N}$ denotes the training data representations in $\mathcal{F}$ and $\mathbf{A} \in \mathbb{R}^{C\times N}$ is a matrix containing the reconstruction weights of $\mathbf{W}_{out}$ with respect to $\boldsymbol{\Phi}$. Clearly, the case where $\mathcal{F}$ is defined to be the feature space determined by the outputs of the network hidden layer outputs $\mathbb{R}^L$, is a special case of the proposed approach. In this case, the dimensionality of the network hidden layer space L may be either finite or infinite, i.e., $L \to \infty$ [16].

By adopting the above ELM kernel definition, we can revisit the ELM and regularized ELM formulations. ELM solves (3), subject to the constraints in (4). This can be expressed as follows:

$$\mathbf{T} = \mathbf{W}_{out}^T\boldsymbol{\Phi} = \mathbf{A}\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \mathbf{A}\mathbf{K}. \quad (14)$$

In the case where $\mathbf{K}$ is non-singular[1], the optimal reconstruction matrix $\mathbf{A}$ is given by:

$$\mathbf{A} = \mathbf{T}\mathbf{K}^{-1}. \quad (15)$$

The network output for a vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T\boldsymbol{\phi}_l = \mathbf{A}\boldsymbol{\Phi}^T\boldsymbol{\phi}_l = \mathbf{A}\mathbf{k}_l. \quad (16)$$

That is, by using the Representer theory [27], [28], [29], the ELM network can be approached as a kernel machine, while by following the analysis in Section II-A this was not possible. It should be noted here that the above KELM algorithm achieves zero training error. This may degrade its generalization performance if the training set contains outliers.

Regularized ELM, minimizes (7), subject to the constraints in (8). By substituting (8) in (7) and using $\mathbf{W}_{out} = \boldsymbol{\Phi}\mathbf{A}^T$, we obtain:

$$\begin{aligned}\mathcal{J}_{RELM} &= \frac{1}{2}\|\boldsymbol{\Phi}\mathbf{A}^T\|_F^2 + \frac{\lambda}{2}\|\mathbf{A}\boldsymbol{\Phi}^T\boldsymbol{\Phi} - \mathbf{T}\|_F^2 \\ &= \frac{1}{2}Tr\left(\mathbf{A}\mathbf{K}\mathbf{A}^T\right) + \frac{\lambda}{2}\|\mathbf{A}\mathbf{K} - \mathbf{T}\|_F^2. \end{aligned} \quad (17)$$

In the case where $\mathbf{K}$ is non-singular, by solving $\nabla_{\mathbf{A}}\mathcal{J}_{RELM} = 0$, the reconstruction weights matrix $\mathbf{A}$ are given by:

$$\mathbf{A} = \mathbf{T}\left(\mathbf{K} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}. \quad (18)$$

The network output for a vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T\boldsymbol{\phi}_l = \mathbf{A}\boldsymbol{\Phi}^T\boldsymbol{\phi}_l = \mathbf{T}\left(\mathbf{K} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\mathbf{k}_l. \quad (19)$$

By comparing (19) and (12), it can be seen that, for the regularized ELM case, the approach followed in [14] provides the same network output with the one obtained by employing the Representer theory-based formulation in (19). However, as will be shown in the next Section, the adoption of the Representer theory-based analysis can be exploited in order to appropriately formulate approximate kernel ELM networks in both the ELM and the regularized ELM cases.

[1]In the case of independent and identically distributed training samples $\mathbf{x}_i$, $i = 1, \ldots, N$, this assumption is always valid.

III. Approximate Kernel Extreme Learning Machine

In this Section, we describe in detail the proposed Approximate Kernel ELM (AKELM). In order to obtain an approximate solution, we assume that the network output weights $\mathbf{W}_{out}$ lie on the span of a subset of the training data (represented in $\mathcal{F}$), i.e. $\mathbf{W}_{out} = \tilde{\mathbf{\Phi}}\mathbf{A}^T$, where $\tilde{\mathbf{\Phi}} \in \mathbb{R}^{|\mathcal{F}|\times n}$, where $\mathbf{A} \in \mathbb{R}^{C\times n}$ is a matrix containing the reconstruction weights of $\mathbf{W}_{out}$ with respect to $\tilde{\mathbf{\Phi}}$. The columns of $\tilde{\mathbf{\Phi}}$ are randomly selected from the columns of $\mathbf{\Phi}$, i.e.:

$$\tilde{\mathbf{\Phi}} = \mathbf{\Phi}\mathbf{E}\mathbf{M}, \tag{20}$$

where $\mathbf{E}$ is a random (column) permutation matrix and $\mathbf{M} \in \mathbb{R}^{N\times n}$ is a matrix with elements $M_{ii} = 1$ and $M_{ij} = 0, i \neq j$. AKELM solves (3), by setting $\mathbf{O} = \mathbf{T}$. This can be expressed as follows:

$$\mathbf{T} = \mathbf{W}_{out}^T\mathbf{\Phi} = \mathbf{A}\tilde{\mathbf{\Phi}}^T\mathbf{\Phi} = \mathbf{A}\tilde{\mathbf{K}}, \tag{21}$$

where $\tilde{\mathbf{K}} \in \mathbb{R}^{n\times N}$ is a submatrix of the original kernel matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$. The optimal reconstruction weight matrix $\mathbf{A}$ is given by:

$$\mathbf{A} = \mathbf{T}\tilde{\mathbf{K}}^T\left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T\right)^{-1} \tag{22}$$

and the network output for a vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T\boldsymbol{\phi}_l = \mathbf{A}\mathbf{\Phi}^T\boldsymbol{\phi}_l = \mathbf{T}\tilde{\mathbf{K}}\left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T\right)^{-1}\mathbf{k}_l. \tag{23}$$

It should be noted here that the calculation of $\mathbf{A}$ through (22) is always possible, since $n \leq N$. Analysis concerning the training error bound of the proposed AKELM algorithm is provided in the Appendix.

Comparing (15) and (23), it can be seen that the proposed AKELM algorithm has a much lower time complexity, when compared to the KELM in (15). Specifically, the KELM algorithm in (15) requires the following processing steps:

- Kernel matrix $\mathbf{K}$ calculation, having time complexity equal to $O(DN^2)$.
- Kernel matrix $\mathbf{K}$ inversion, having time complexity equal to $O(N^3)$.
- Reconstruction weights matrix $\mathbf{A}$ calculation, having time complexity equal to $O(CN^2)$.

The proposed AKELM algorithm requires the following processing steps:

- Calculation of the matrix $\tilde{\mathbf{K}}$ having time complexity equal to $O(nDN)$.
- Calculation of the matrix $\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T$, having time complexity equal to $O(n^2N)$.
- Reconstruction weights matrix $\mathbf{A}$ calculation, having time complexity equal to $O((n^2 + C)N)$.

From the above, the computational complexity of the KELM algorithm is equal to $O(N^3 + (D + C)N^2)$, while the time complexity of the proposed AKELM algorithm is equal to $O((2n^2 + nD + C)N)$. By setting $n = pN$ and $D = mN$, the time complexity of the KELM algorithm is equal to $O((m+1)N^3+CN^2)$ and the time complexity of the proposed AKELM algorithm is equal to $O((2p^2+p)N^3+CN)$. Taking into account that, for large scale classification problems, $m \ll 1$ and that satisfactory performance can be achieved by using a value $p \ll 1$, as shown in the experimental evaluation provided in Section IV, the computational cost of the proposed AKELM algorithm in the training phase is significantly lower than the one of the KELM algorithm. For example, in the Youtube Faces database [37], the proposed AKELM algorithms achieves a good performance by using a value of $p = 1.6 \cdot 10^{-3}$. In that database, where $m = 6 \cdot 10^{-3}$, the acceleration achieved by applying the proposed AKELM algorithm versus KELM is in the order of 10^3. In the test phase, the time complexity of KELM algorithm is equal to $O(CN^2 + D^2N) = O(CN^2 + m^2N^3)$, while the time complexity of the proposed AKELM algorithm is equal to $O(Cn^2 + D^2n) = O(Cp^2N^2 + m^2pN^2)$. Thus, in the test phase the computational cost of the proposed AKELM algorithm is significantly lower from that of KELM too. Regarding memory requirements, KELM employs a matrix of $N \times N$ dimensions, while the proposed AKELM a matrix of $pN \times N$ dimensions.

A. Regularized Approximate Kernel Extreme Learning Machine

Following a similar analysis as above, a regularized version of the proposed AKELM algorithm can be obtained by solving for:

$$\begin{aligned}\mathcal{J}_{RAKELM} &= \frac{1}{2}\|\tilde{\mathbf{\Phi}}\mathbf{A}^T\|_F^2 + \frac{\lambda}{2}\|\mathbf{A}\tilde{\mathbf{\Phi}}^T\mathbf{\Phi} - \mathbf{T}\|_F^2 \\ &= \frac{1}{2}tr\left(\mathbf{A}\tilde{\mathbf{\Phi}}^T\tilde{\mathbf{\Phi}}\mathbf{A}^T\right) + \frac{\lambda}{2}\|\mathbf{A}\tilde{\mathbf{K}} - \mathbf{T}\|_F^2 \\ &= \frac{1}{2}tr\left(\mathbf{A}\hat{\mathbf{K}}\mathbf{A}^T\right) + \frac{\lambda}{2}\|\mathbf{A}\tilde{\mathbf{K}} - \mathbf{T}\|_F^2, \end{aligned} \tag{24}$$

where $tr(\cdot)$ denotes the trace operator and $\hat{\mathbf{K}} \in \mathbb{R}^{n\times n}$. By solving $\nabla_{\mathbf{A}}\mathcal{J}_{RAKELM} = 0$, the reconstruction weights matrix $\mathbf{A}$ is given by:

$$\mathbf{A} = \mathbf{T}\tilde{\mathbf{K}}^T\left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + \frac{1}{\lambda}\hat{\mathbf{K}}\right)^{-1}. \tag{25}$$

The network output for a vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T\boldsymbol{\phi}_l = \mathbf{A}\mathbf{\Phi}^T\boldsymbol{\phi}_l = \mathbf{T}\tilde{\mathbf{K}}^T\left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + \frac{1}{\lambda}\hat{\mathbf{K}}\right)^{-1}\mathbf{k}_l. \tag{26}$$

The calculation of $\mathbf{A}$ through (25) is always possible, since $n \leq N$. It should be noted here that, compared to the AKELM algorithm described in the previous section, the calculation of the reconstruction weights matrix $\mathbf{A}$ through (26) requires an additional $n \times n$ matrix addition operations. Thus, its time complexity is equal to $O((2p^2+p)N^3+p^2N^2+CN)$, which is significantly lower than that of KELM algorithm. This is due to the fact that the matrix $\hat{\mathbf{K}}$ is a sub-matrix of $\tilde{\mathbf{K}}$ and, thus, the elements of $\hat{\mathbf{K}}$ need not to be computed. The computational cost during testing and the memory requirements of the regularized AKELM algorithm (noted as RAKELM hereafter) are the same with that of AKELM algorithm.

B. Approximate Kernel Extreme Learning Machine exploiting Variance Regularizer

By using a sufficiently large number of hidden layer neurons, ELM classification, when approached from a Subspace Learning (SL) point of view, can be considered to be a learning process formed by two processing steps. The first step corresponds to a (nonlinear) mapping process of the input space $\mathbb{R}^D$ to a high-dimensional feature space $\mathbb{R}^L$ (the so-called ELM space), preserving some properties of interest for the training data. In the second step, an optimization process is employed for the determination of a linear projection of the high-dimensional data to a low-dimensional feature space, where classification is performed by a linear classifier. Based on this observation, the ELM algorithm has been recently extended, in order to incorporate the within-class variance of the training data [32] and the variance of the entire training set [30] (expressed in the ELM space) in its optimization process, leading to enhanced classification performance.

In order to exploit variance criteria of the training data in $\mathcal{F}$, the following optimization problem is solved:

$$\mathcal{J}_{MVAKELM} = \frac{1}{2}\|\mathbf{S}^{\frac{1}{2}}\mathbf{W}_{out}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}_{out}^T\mathbf{\Phi} - \mathbf{T}\|_F^2, \quad (27)$$

where $\mathbf{S} \in \mathbb{R}^{|\mathcal{F}|\times|\mathcal{F}|}$ is a matrix (of arbitrary dimensions) expressing variance criteria in $\mathcal{F}$. By exploiting a Graph Embedded-based approach [38], we can assume that the training data representations in $\mathcal{F}$ are used in order to form the vertex set of a graph $\mathcal{G} = \{\mathbf{\Phi}, \mathbf{V}\}$, where $\mathbf{V} \in \mathbb{R}^{N\times N}$ is a similarity matrix whose elements denote the relationships between the graph vertices $\mathbf{\Phi}$. $\mathbf{S}$ can be defined by $\mathbf{S} = \mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^T$, where $\mathbf{L} \in \mathbb{R}^{N\times N}$ is the graph Laplacian matrix defined by $\mathbf{L} = \mathbf{D} - \mathbf{V}$, $\mathbf{D}$ being the diagonal degree matrix of $\mathcal{G}$ having elements $D_{ii} = \sum_{j=1}^{N} V_{ij}$. (27) can now be expressed as follows:

$$\begin{aligned}\mathcal{J}_{MVAKELM} &= \frac{1}{2}tr\left(\mathbf{A}\tilde{\mathbf{K}}\mathbf{L}\tilde{\mathbf{K}}^T\mathbf{A}^T\right) + \frac{\lambda}{2}tr\left(\tilde{\mathbf{K}}\mathbf{A}^T\mathbf{A}\tilde{\mathbf{K}}\right) \\ &- \lambda\, tr\left(\tilde{\mathbf{K}}\mathbf{A}^T\mathbf{T}\right) + tr\left(\mathbf{T}^T\mathbf{T}\right). \quad (28)\end{aligned}$$

By solving $\nabla_{\mathbf{A}}\mathcal{J}_{MVAKELM} = 0$, $\mathbf{A}$ is given by:

$$\mathbf{A} = \mathbf{T}\tilde{\mathbf{K}}^T\left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + \frac{1}{\lambda}\tilde{\mathbf{K}}\mathbf{L}\tilde{\mathbf{K}}^T\right)^{-1}. \quad (29)$$

As can be seen from (29), subspace learning criteria designed in the context of Graph Embedding framework can be incorporated in the calculation of the reconstruction weights matrix $\mathbf{A}$. However, the adoption of general graph structures would require a prohibitive additional computational cost and memory usage, since $\mathbf{L}^{N\times N}$. Fortunately, the Laplacian matrices used to define the total and within-class variance of the training data in Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), respectively, have the form:

$$\mathbf{L}_T = \frac{1}{N}\left(\mathbf{I} - \frac{1}{N}\mathbf{e}\mathbf{e}^T\right) \quad (30)$$

and

$$\mathbf{L}_w = \left(\mathbf{I} - \sum_{k=1}^{C}\frac{1}{N_k}\mathbf{e}_k\mathbf{e}_k^T\right), \quad (31)$$

where $\mathbf{e} \in \mathbb{R}^N$ is a vector of ones, $\mathbf{e}_k \in \mathbb{R}^N$ is a vector with elements equal to $e_{ki} = 1$ when $c_i = k$ and $e_{ik} = 0$ when $c_i \neq k$. N_k denotes the cardinality of class k. By substituting (30) and (31) in (29), we obtain:

$$\mathbf{A}_T = \mathbf{T}\tilde{\mathbf{K}}^T\left[\left(\frac{\lambda N+1}{\lambda N}\right)\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T - \frac{1}{\lambda N^2}(\tilde{\mathbf{K}}\mathbf{e})(\mathbf{e}\tilde{\mathbf{K}})^T\right]^{-1} \quad (32)$$

and

$$\mathbf{A}_w = \mathbf{T}\tilde{\mathbf{K}}^T\left[\left(\frac{\lambda N+1}{\lambda N}\right)\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T - \sum_{k=1}^{C}\frac{1}{\lambda N_k}(\tilde{\mathbf{K}}\mathbf{e}_k)(\mathbf{e}_k\tilde{\mathbf{K}})^T\right]^{-1}. \quad (33)$$

The calculation of $\mathbf{A}$ through (32) and (33) is always possible, since $n \leq N$. Compared to the RAKELM algorithm described in the previous section, the calculation of the reconstruction weights matrix $\mathbf{A}$ through (32) and (33) requires an additional multiplication of a $n \times N$ matrix with a vector of N elements and the calculation of a symmetric matrix through the multiplication of two vectors of N elements. Thus, its time complexity is equal to $O((2p^2+p)N^3 + (p^2+p+0.5)N^2 + CN)$, which is significantly lower than that of KELM algorithm. The computational cost during testing and the memory requirements of the AKELM algorithms exploiting variance criteria (noted as MVAKELM and MCVAKELM for the formulations in (32) and (33), respectively) are the same with that of AKELM and RAKELM algorithms.

C. Discussion

In the following, we highlight the difference between the proposed approach and those followed in [14], [39], where the optimization problem (7) is solved by using its dual formulation (9), in order to derive the two solutions for $\mathbf{W}_{out}$ given in (10) and (11). From these two solutions, one way to derive an ELM solution of reduced dimensions involving $\tilde{\mathbf{K}}$ is to use a subset of the training data $\tilde{\mathbf{X}} \subseteq \mathbf{X}$ as input weights, employ the kernel function $k(\cdot,\cdot)$ instead of $\Phi(\cdot)$ and calculate the network output weights through:

$$\mathbf{W}_{out} = \left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\tilde{\mathbf{K}}\mathbf{T}^T. \quad (34)$$

Subsequently, the network output for a vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T\mathbf{k}_l = \mathbf{T}\tilde{\mathbf{K}}^T\left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\mathbf{k}_l. \quad (35)$$

Comparing (25) and (34), it can be seen that the adopted regularization terms are different. This is due to the fact that (25), by following the Representer theory, regularizes the network output weights $\mathbf{W}_{out}$ in the feature space $\mathcal{F}$. However, in (34) the network output weights are regularized in the space determined by the network hidden layer outputs $\mathbb{R}^L$. A second observation is that the network output calculation for $\mathbf{x}_l$ through (35) involves the matrix $\tilde{\mathbf{K}}$ and the vector $\mathbf{k}_l$ defined in the space determined by the network hidden layer outputs $\mathbb{R}^L$ and not in the feature space $\mathcal{F}$ of arbitrary dimensionality. Both these differences are due to the fact that the solution in (34) corresponds to a regularized ELM

network, where some of the training data are employed, in order to form the input weight matrix, i.e. $\mathbf{W}_{in} = \tilde{\mathbf{X}}$, where $\tilde{\mathbf{X}} = \mathbf{XEM}$. A similar approach has also been proposed in [39], where the input data are mapped to a subspace of the kernel space following a Nyström-based kernel approximation. This representation is obtained by $\hat{\mathbf{\Phi}} = \tilde{\mathbf{K}}\hat{\mathbf{K}}^{-\frac{1}{2}}$. Using $\hat{\mathbf{\Phi}}$, optimization problems proposed for different ELM variants are subsequently solved [32], [40].

IV. Experiments

In this Section, we present experiments conducted in order to evaluate the performance of the proposed AKELM algorithm. We have employed four publicly available facial image datasets to this end. A brief description of the datasets is provided in the subsection IV-A. Experimental results are provided in subsection IV-B. In all our experiments we compare the performance and efficiency of the proposed AKELM algorithms with that of ELM [1], RELM [14] and Kernel ELM (KELM) [14] algorithms. All the experiments have been conducted on a 4-core, $i7-4790$, 3.6GHz PC with 32GB RAM using single floating point precision and a MATLAB implementation.

A. Data sets

1) JAFFE dataset [41]: consists of 210 facial images depicting 10 Japanese female persons performing emotions: anger, disgust, fear, happyiess, sadness, surprise and neutral. Each of the persons is depicted in 3 images for each expression. Example images of the dataset are illustrated in Figure 1.

Fig. 1. *Facial images depicting a person of the JAFFE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

2) COHN-KANADE dataset [42]: consists of facial images depicting 210 persons of age between 18 and 50 performing the seven emotions. Each facial expression is depicted in 35 facial images. Example images of the dataset are illustrated in Figure 2.

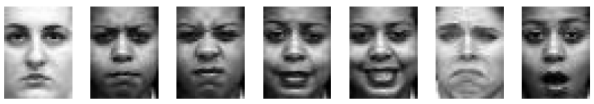

Fig. 2. *Facial images from the COHN-KANADE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

3) BU dataset [43]: consists of facial images depicting over 100 persons performing the seven emotions. The persons of the comes from a variety of ethnic/racial background, including White, Black, East-Asian, Middle-East Asian and Hispanic Latino. In our experiments, we have employed the images depicting the most expressive intensity of each facial expression. Example images of the dataset are illustrated in Figure 3.

Fig. 3. *Facial images depicting a person of the BU dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

TABLE I

Training set cardinality N on each evaluation round for the JAFFE, COHN-KANADE and BU datasets.

JAFFE	COHN-KANADE	BU
4200	4900	14000

4) Youtube Faces dataset [37]: consists of 621126 facial images depicting 1595 persons. All images have been downloaded from YouTube. In our experiments we have employed the facial images depicting persons in at least 500 images, resulting to a dataset of 370319 images and 340 classes. Example images of the dataset are illustrated in Figure 4.

Fig. 4. *Facial images depicting persons of the YouTube dataset.*

.

B. Experimental Results

In our first set of experiments, we have applied the various ELM algorithms on the JAFFE, COHN-KANADE and BU datasets. Since there is not a widely adopted experimental protocol for these datasets, we perform the five-fold cross-validation procedure [44], by taking into account the facial expression labels. That is, we randomly split the facial images depicting the same emotion in five sets and we use four splits of all the emotions for training and the remaining splits for evaluation. This process is performed five times, one for each evaluation split. On each evaluation round, we enrich the training set by following the approach proposed in [45]. Specifically, each training facial image was translated, rotated and scaled according to 25 different geometric transformations. Each eye has five possible positions: original position, one-pixel left, right, up and down (resulting into 25 pairs). Therefore, the cardinality of the training set on each evaluation round is multiplied by a factor of 25, since the enriched version of each training set consists of the original training (centered) facial images and the 24 shifted images for each centered

TABLE II

PERFORMANCE ON THE JAFFE DATASET.

L,n,$\tilde{N}$	ELM	KELM	**AKELM**	RELM	**RAKELM**	**MVAKELM**	**MCVAKELM**
25	33.81%	35.24%	**51.43**%	35.71%	**54.29**%	**54.29**%	**54.29**%
50	43.81%	45.71%	**51.43**%	39.52%	**54.76**%	**54.76**%	**55.24**%
100	45.24%	49.05%	**54.29**%	44.76%	**55.71**%	**56.19**%	**57.14**%
250	50%	50.95%	**56.19**%	50.48%	**57.62**%	**58.1**%	**58.57**%
500	47.67%	58.1%	**60**%	55.71%	**60.48**%	**60.48**%	**60.95**%
4200	-	60%	-	-	-	-	-

TABLE III

PERFORMANCE ON THE COHN-KANADE DATASET.

L,n,$\tilde{N}$	ELM	KELM	**AKELM**	RELM	**RAKELM**	**MVAKELM**	**MCVAKELM**
25	48.98%	29.39%	**49.8**%	48.98%	49.8%	49.8%	**50.61**%
50	**58.78**%	39.59%	56.73%	60%	**60** %	**60**%	**61.63**%
100	**64.9**%	47.76%	63.67%	64.08%	**65.31**%	**65.63**%	**66.94**%
250	67.76%	55.92%	**69.8**%	69.8%	**71.02**%	**71.02**%	**71.84**%
500	68.16%	66.53%	**70.2**%	71.43%	**76.73**%	**76.94**%	**76.94**%
4900	-	73.47%	-	-	-	-	-

TABLE IV

PERFORMANCE ON THE BU DATASET.

L,n,$\tilde{N}$	ELM	KELM	**AKELM**	RELM	**RAKELM**	**MVAKELM**	**MCVAKELM**
25	47.14%	28.43%	**47.71**%	47.14%	**47.71**%	**47.71**%	**48.07**%
50	51.14%	38%	**56.43**%	52.29%	**56.43**%	**56.71**%	**56.71**%
100	60.57%	46.43%	**62.43**%	60.86%	**62.43**%	**62.43**%	**62.71**%
250	67.43%	51.57%	**69.29**%	67.86%	**69.57**%	**69.57**%	**70.29**%
500	67.86%	57.57%	**69.14**%	69.19%	**70.43**%	**71**%	**71.22**%
14000	-	68.29%	-	-	-	-	-

image. The cardinality N of the training set used on each evaluation round for the JAFFE, COHN-KANADE and BU datasets is illustrated in Table I.

In all the experiments, we have resized the facial images in 40×30 pixel images and vectorized these images in order to create vectors $\mathbf{x}_i \in \mathbb{R}^{1200}$. For the ELM methods exploiting a kernel formulation we have employed the RBF kernel function:

$$\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\Big). \tag{36}$$

The value of σ is set equal to the mean Euclidean distance between the training vectors $\mathbf{x}_i$ that corresponds to the natural scaling value for each dataset. In the case of ELM formulations exploiting random input weights $\mathbf{q}_j$, we have employed the RBF activation function:

$$\Phi(\mathbf{x}_i, \mathbf{q}_j, b) = exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{q}_j\|_2^2}{2b^2}\Big), \tag{37}$$

where the value b is set equal to the mean Euclidean distance between the training data $\mathbf{x}_i$ and the network input weights $\mathbf{q}_j$.

We test the performance of the ELM algorithms exploiting random input weights for a number of hidden layer neurons $L = \{25, 50, 100, 250, 500\}$. We determine the cardinality of the reference data in the proposed AKELM algorithms accordingly, i.e. $n = \{25, 50, 100, 250, 500\}$. For the KELM approach, we test the original approach exploiting the entire training set in order to calculate the network weights through (11) and an approximate KELM approach, exploiting a subset of the training data for network weights calculation through (11). We set the cardinality of the reduced training set similar to the ELM and AKELM approaches, i.e. $\tilde{N} = \{25, 50, 100, 250, 500\}$. For fair comparison, in all the experiments, we use the same subset for the proposed AKELM methods and the KELM approach using a reduced training set. Regarding the optimal value of the regularization parameter λ used in all the regularized ELM formulations, it has been determined by following a line search strategy, which is applied on the training data, using the values $\lambda = 10^r$, $r = -6, \ldots, 6$.

Experimental results obtained by applying the competing algorithms on the JAFFE, COHN-KANADE and BU datasets are illustrated in Tables II, III and IV, respectively. As can be seen, the proposed AKELM approach outperforms both the ELM approach exploiting random input parameters and the KELM approach employing a subset of the training data in all cases. In addition, the proposed AKELM algorithm provides performance comparable with that of KELM applied in the entire training set. The incorporation of variance criteria in MVAKELM and MCVAKELM algorithms further enhances performance, since these two algorithms achieve the best performance in all the cases.

In Tables V, VI and VII we provide the computational cost and memory requirements of the ELM, KELM and AKELM algorithms, when applied to the JAFFE, COHN-KANADE and BU datasets. As can be seen, during testing the computational cost and the memory requirements of the proposed AKELM algorithm are the same with that of ELM. The computational cost of the proposed AKELM algorithm during training is

TABLE V

TRAINING AND TEST TIMES AND MEMORY DURING TRAINING IN JAFFE DATASET.

	Training time (sec)			Test time (sec)			Memory (MB)		
L,n,$\tilde{N}$	ELM	KELM	AKELM	ELM	KELM	AKELM	ELM	KELM	AKELM
25	0.047	0.001	0.047	0.001	0.001	0.001	0.897	0.006	0.897
50	0.054	0.002	0.054	0.001	0.001	0.001	1.794	0.024	1.794
100	0.07	0.004	0.071	0.001	0.001	0.001	3.364	0.084	3.364
250	0.12	0.018	0.121	0.002	0.002	0.002	8.075	0.484	8.075
500	0.21	0.08	0.211	0.005	0.005	0.005	16.15	1.938	16.15
4200	-	**9.24**	-	-	**0.258**	-	-	**134.58**	-

TABLE VI

TRAINING AND TEST TIMES AND MEMORY DURING TRAINING IN COHN-KANADE DATASET.

	Training time (sec)			Test time (sec)			Memory (MB)		
L,n,$\tilde{N}$	ELM	KELM	AKELM	ELM	KELM	AKELM	ELM	KELM	AKELM
25	0.058	0.001	0.058	0.001	0.001	0.001	1.05	0.006	1.05
50	0.065	0.002	0.066	0.001	0.001	0.001	2.09	0.024	2.09
100	0.079	0.004	0.08	0.001	0.001	0.001	3.93	0.084	3.93
250	0.145	0.018	0.147	0.003	0.003	0.003	9.42	0.484	9.42
500	0.243	0.08	0.245	0.005	0.005	0.005	18.84	1.938	18.84
4900	-	**12.11**	-	-	**0.314**	-	-	**183.18**	-

TABLE VII

TRAINING AND TEST TIMES AND MEMORY DURING TRAINING IN BU DATASET.

	Training time (sec)			Test time (sec)			Memory (MB)		
L,n,$\tilde{N}$	ELM	KELM	AKELM	ELM	KELM	AKELM	ELM	KELM	AKELM
25	0.179	0.001	0.18	0.002	0.001	0.002	2.67	0.006	2.67
50	0.197	0.002	0.198	0.002	0.001	0.002	5.34	0.024	5.34
100	0.237	0.004	0.239	0.002	0.001	0.002	10.68	0.084	10.68
250	0.414	0.018	0.416	0.004	0.003	0.004	26.7	0.484	26.7
500	0.704	0.08	0.706	0.009	0.005	0.009	53.41	1.938	53.41
14000	-	**152.17**	-	-	**1.265**	-	-	**1495.4**	-

slightly higher than that of ELM, but the differences are insignificant. On the other hand, the computational cost during both training and testing and the memory requirements of the KELM algorithm are significantly higher than the ones of ELM and AKELM.

In Table VIII, we also compare the performance obtained by applying AKELM-based classification with that of other, recently proposed, methods evaluating their performance on the BU, Jaffe and Kanade databases. As can be seen, the proposed AKELM methods achieve satisfactory performance in all the cases. Specifically, AKELM-based classification outperforms [46] and [47] in all cases, while MMRP proposed in [48] provides the best performance on the Cohn-Kanade dataset. This may be explained by the fact that the resolution of the facial images used in [48] was equal to 150×200 pixels, i.e., five times the resolution of the facial images used in our experiments. Even in this case, it can be seen that AKELM outperforms MMP and SVM-based facial image classification.

TABLE VIII

COMPARISON OF OUR RESULTS WITH SOME STATE-OF-THE-ART METHODS ON THE BU, JAFFE AND KANADE DATASETS.

	BU	Jaffe	Kanade
Method [46]	-	56.72%	69.05%
Method [47]	66.4%	-	72.9%
Method [48] - MMP	-	-	70.3%
Method [48] - RP	-	-	75.2%
Method [48] - SVM	-	-	73.4%
Method [48] - MMRP	-	-	**80.1**%
AKELM	**69.14**%	**60**%	70.2%
RAKELM	**70.43**%	**60.48**%	**76.73**%
MVAKELM	**71**%	**60.48**%	**76.94**%
MCVAKELM	**71.22**%	**60.95**%	**76.94**%

In our second set of experiments, we have applied the competing algorithms in Youtube Faces dataset. In these experiments, we have employed the facial image representation suggested in [37]. Specifically, the bounding box of the face detector output has been expanded by 2.2 of its original size and cropped from the video frame. The expanded bounding box is resized to an image of 200×200 pixels. We then use the centered 100×100 pixels to obtain the facial image, which is converted to grayscale. Facial images are subsequently aligned by fixing the coordinates of automatically detected facial feature points [49] and the Local Binary Patter (LBP) description [50] was employed, leading to a facial image representation $\mathbf{x}_i$ of $D = 1770$ dimensions. The RBF kernel function in (36) and the RBF activation function in (37) have been employed for the kernel ELM algorithms and the ELM algorithms exploiting random input parameters, respectively. Since there is no widely adopted experimental protocol on the YouTube Faces dataset for multi-class classification, we retain 80% of the facial images for training and the remaining 20% for testing, by taking into account the ID labels of the persons in the dataset. That is, we keep 80% of the facial

TABLE IX

PERFORMANCE (CR) ON THE YOUTUBE FACES DATASET.

L,n	ELM	**AKELM**	RELM	**RAKELM**	**MVAKELM**	**MCVAKELM**
25	7.57%	**23.93%**	17.9%	**23.93%**	**24.77%**	**26.19%**
50	31.3%	**36.28%**	31.31%	**36.28%**	**37.09%**	**40.4%**
100	45.18%	**51.16%**	45.18%	**51.17%**	**52.32%**	**59.16%**
250	67.32%	**73.84%**	67.37%	**73.84%**	**77.45%**	**77.85%**
500	54.72%	**84.45%**	54.74%	**84.47%**	**85.12%**	**86.64%**
1000	73.81%	**85.08%**	73.84%	**85.12%**	**87.32%**	**87.45%**
2000	77.03%	**89.93%**	77.03%	**89.99%**	**90.18%**	**91.36%**
5000	86.61%	**94.35%**	86.64%	**94.38%**	**95.69%**	**97.79%**

TABLE XI

TRAINING AND TEST TIMES AND MEMORY DURING TRAINING IN YOUTUBE FACES DATASET.

	Training time (sec)			Test time (sec)			Memory (MB)		
L,n/$\tilde{N}$	ELM	KELM	AKELM	ELM	KELM	AKELM	ELM	KELM	AKELM
25/-	3.26	-	3.27	0.78	-	0.78	28.25	-	28.25
50/500	3.57	0.06	3.52	0.84	2.02	0.84	56.51	3.81	56.51
100/1000	4.59	0.22	4.64	0.96	3.21	0.96	113.01	15.26	113.01
250/2000	6.11	1.40	5.91	1.35	5.55	1.35	282.53	95.37	282.53
500/5000	10.01	23.68	9.64	1.95	**13.06**	1.95	565.94	381.47	565.94
1000/10000	21.82	**182.81**	21.51	3.22	**25.85**	3.22	1131.9	1525.4	1131.9
2000/20000	42.96	**1411.8**	41.41	5.63	**50.47**	5.63	2263.9	1525.4	2263.9
5000/296257	322.52	$\mathbf{4 \cdot 10^5}$	322.06	13.07	**820.9**	13.07	4537.6	$\mathbf{3 \cdot 10^5}$	4537.6

TABLE X

PERFORMANCE (CR) OF KELM ON THE YOUTUBE FACES DATASET.

$\tilde{N}$	500	1000	2000	5000	10000	20000
CR	67.15%	76.61%	78.37%	83.14%	85.12%	90.18%

images depicting each person for training and the remaining facial images for evaluation.

We test the performance of the ELM algorithms exploiting random input weights for a number of hidden layer neurons $L = \{25, 50, 100, 250, 500, 1000, 2000, 5000\}$. We determine the cardinality of the reference data in the proposed AKELM algorithms accordingly, i.e. $n = \{25, 50, 100, 250, 500, 1000, 2000, 5000\}$. For KELM, we test the approximate KELM approach, exploiting a subset of the training data for network weights calculation through (11). Since in this classification problem the number of classes is equal to 340, we set the cardinality of the reduced training set to $\tilde{N} = \{500, 1000, 2000, 5000, 10000, 20000\}$. Regarding the optimal value of the regularization parameter λ used in all the regularized ELM formulations, it has been determined by following a line search strategy, which is applied on the training data, using the values $\lambda = 10^r,\ r = -6, \ldots, 6$.

Experimental results obtained by applying the competing algorithms on the YouTube Faces dataset are illustrated in Table IX. As can be seen, the proposed AKELM algorithm outperforms the ELM algorithm exploiting random input parameters in all cases. Since the number of classes in this classification problem is equal to 340, we test the performance of the KELM algorithm that exploits a subset of the training data using the values $\tilde{N} = \{1000, 2000, 5000, 10000, 20000\}$. The performance of KELM for different training set cardinalities $\tilde{N}$ is illustrated in Table X. KELM requires a relatively large training set in order to achieve performance comparable with the one obtained by applying the proposed AKELM algorithm. For example, KELM achieves a classification rate equal to 90.18% for $\tilde{N} = 20000$, while AKELM achieves a similar performance, 89.93%, for $n = 2000$. This fact also increases its computational cost in the test phase, since test kernel vectors have a higher dimension, i.e. $\mathbf{k}_l \in \mathbb{R}^{\tilde{N}}$, $\tilde{N} \gg n$. Overall, by using a value of $n = 5000$, the proposed AKELM algorithm achieves a classification rate equal to 94.35%, thus outperforming ELM providing a classification rate equal to 86.61% and KELM applied on a subset of $\tilde{N} = 20000$ training data providing a classification rate equal to 90.18%. The proposed RAKELM algorithm outperforms both RELM in all the cases, while the adoption of variance criteria in MVAKELM and MCVAKELM algorithm further enhances classification performance.

In Table XI we provide the computational cost and memory requirements of the ELM, KELM and AKELM algorithms, when applied to the YouTube Faces dataset. As can be seen, the computational cost during testing and the memory requirements of the proposed AKELM algorithm is the same with that of ELM. The computational cost of the proposed AKELM algorithm during training is similar to that of ELM. On the other hand, the computational cost during both training and testing of KELM are significantly higher than the ones of ELM and AKELM for $\tilde{N} > 10000$. It is expected that the application of the KELM algorithm on the YouTube Faces dataset would require the storage of a matrix of $3 \cdot 10^5$ MB and its training process would require $4 \cdot 10^5$ seconds (approximately 4.5 days). In addition, testing by using the KELM classifier trained on the entire training set would require 820 seconds. Even for the case where KELM is trained on a subset of $\tilde{N} = 20000$ training vectors, the training and test times of the KELM algorithm are significantly higher from those of the proposed AKELM algorithm.

Finally, in Table XII, we compare the performance of the ELM, KELM and AKELM algorithms in the case where each

TABLE XII
PERFORMANCE ON THE YOUTUBE FACES DATASET WITH RESPECT TO THE TRAINING TIME.

Training time (sec)	ELM	KELM	**AKELM**
10^0	67.32%	**78.37**%	73.84%
10^1	77.03%	83.14%	**89.93**%
10^2	86.61%	85.12%	**94.35**%

algorithm is allowed to exploit a constant training time for training. By exploiting a training time of the order of 1 second, KELM employing a reduced cardinality training set is able to achieve a performance equal to 78.37%, outperforming ELM and AKELM algorithms achieving a performance equal to 67.32% and 73.84%, respectively. For training times of the order of 10 and 10^2 seconds, the AKELM algorithm achieves classification rates equal to 89.93% and 94.35%, respectively, outperforming both ELM and KELM algorithms.

Overall, by observing Tables II-XII, it can be seen that the proposed AKELM consistently outperforms ELM, in both the standard and regularized cases, while it is able to outperform KELM, when similar training times are allowed for each algorithm (in the orders of 10^m, $m = 0, 1, 2$. In addition, the proposed AKELM algorithms scale well in both training/test time and memory. The incorporation of the total and within-class variance information of the training data in the feature space generally results to increased performance.

V. CONCLUSIONS

In this paper, we proposed an approximate solution of the kernel Extreme Learning Machine classifier that is able to scale well in both training/test computational cost and memory. We have extended the proposed Approximate Kernel Extreme Learning Machine classifier, in order to exploit regularization and the total and within-class variance of the training data in the feature space. Extensive experimental evaluation denotes that the proposed approach is able to operate extremely fast in both the training and test phases and to provide satisfactory performance, outperforming relating classification schemes in most cases.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 316564 (IMPART).

APPENDIX I
TRAINING ERROR BOUNDS

Let us denote by $\mathbf{b} \in \mathbb{R}^L$ an indicator vector, having elements equal to $b_i = 1$, if $\boldsymbol{\phi}_i \in \tilde{\boldsymbol{\Phi}}$ and $b_i = 0$, otherwise, where $\|\mathbf{b}\|_1 = n$. Let us denote by $\mathbf{w}_k$ and $\tilde{\mathbf{w}}_k$ the output weights corresponding to the k-th one-versus-rest classification problem of the KELM and AKELM algorithms, respectively.

The mean training error of the AKELM algorithm can be expressed as follows:

$$\begin{aligned} L(\mathbf{W}_{out}, \mathbf{b}) &= \min_{\mathbf{w}_k} \frac{1}{N} \sum_{k=1}^{C} \|\boldsymbol{\Phi}\mathbf{w}_k - \mathbf{t}_k\|_2^2 \\ &= \min_{\mathbf{a}_k} \frac{1}{N} \sum_{k=1}^{C} \|\mathbf{K}(\mathbf{a}_k \circ \mathbf{b}) - \mathbf{t}_k\|_2^2 \\ &= \min_{\mathbf{a}_k} \frac{1}{N} \sum_{k=1}^{C} \Big[(\mathbf{a}_k \circ \mathbf{b})^T \mathbf{K}\mathbf{K}(\mathbf{a}_k \circ \mathbf{b}) \\ &- 2\mathbf{t}_k \mathbf{K}\mathbf{K}(\mathbf{a}_k \circ \mathbf{b}) + \mathbf{t}_k^T \mathbf{t}_k \Big] \end{aligned} \tag{38}$$

The expected value of the training error can be expressed as follows:

$$\begin{aligned} E[L(\mathbf{W}_{out}, \mathbf{b})] &= E\Bigg[\min_{\mathbf{a}_k} \frac{1}{N} \sum_{k=1}^{C} \Big[(\mathbf{a}_k \circ \mathbf{b})^T \mathbf{K}\mathbf{K}(\mathbf{a}_k \circ \mathbf{b}) \\ &- 2\mathbf{t}_k \mathbf{K}\mathbf{K}(\mathbf{a}_k \circ \mathbf{b}) + \mathbf{t}_k^T \mathbf{t}_k \Big] \Bigg] \\ &\leq \min_{\mathbf{a}_k} \frac{1}{N} E\Bigg[\sum_{k=1}^{C} \Big[(\mathbf{a}_k \circ \mathbf{b})^T \mathbf{K}\mathbf{K}(\mathbf{a}_k \circ \mathbf{b}) \\ &- 2\mathbf{t}_k \mathbf{K}\mathbf{K}(\mathbf{a}_k \circ \mathbf{b}) + \mathbf{t}_k^T \mathbf{t}_k \Big] \Bigg] \\ &= \min_{\mathbf{a}_k} \frac{1}{N} \Bigg[\frac{n}{N} \mathbf{a}_k^T \left(\frac{n}{N} \mathbf{K}\mathbf{K} + d\mathbf{D} \right) \mathbf{a}_k \\ &- \frac{2n}{N} \mathbf{t}_k^T \mathbf{K}\mathbf{a}_k + \mathbf{t}_k^T \mathbf{t}_k \Bigg] = \mathcal{J}_k^{\mathbf{b}}, \end{aligned} \tag{39}$$

where $d = 1 - \frac{n}{N}$ and $\mathbf{D} = diag(\mathbf{K}\mathbf{K})$. Solving for $\nabla_{\mathbf{a}_k} \mathcal{J}_k^{\mathbf{b}} = 0$, we obtain:

$$\mathbf{a}_k = \left(\frac{n}{N} \mathbf{K}\mathbf{K} + d\mathbf{D} \right)^{-1} \mathbf{K}\mathbf{t}_k. \tag{40}$$

Substituting (40) in (39), we obtain:

$$\mathcal{J}_k^{\mathbf{b}} = \frac{1}{N} \left[\mathbf{t}_k^T \mathbf{t}_k - \frac{n}{N} \mathbf{t}_k^T \mathbf{K} \left(\frac{n}{N} \mathbf{K}\mathbf{K} - \frac{N-n}{N} \mathbf{D} \right)^{-1} \mathbf{K}\mathbf{t}_k \right]. \tag{41}$$

Thus, by setting $\mathbf{b} = \mathbf{1}$ where $\mathbf{1} \in \mathbb{R}^N$ is a vector of ones, the training error of the KELM algorithm is equal to:

$$\mathcal{J}_k^{\mathbf{1}} = \frac{1}{N} \mathbf{t}_k^T \mathbf{t}_k - \frac{1}{N} \mathbf{t}_k^T \mathbf{K} (\mathbf{K}\mathbf{K})^{-1} \mathbf{K}\mathbf{t}_k. \tag{42}$$

From (42) it can be seen that, in the case where $\mathbf{K}$ is invertible, KELM achieves zero training error. For the training error

bound of AKELM algorithm we have:

$$
\begin{aligned}
&E[L(\mathbf{W}_{out},\mathbf{b})] - L(\mathbf{W}_{out},\mathbf{1}) = \frac{1}{N}\mathbf{t}_k^T\mathbf{t}_k \\
&-\frac{n}{N^2}\mathbf{t}_k^T\mathbf{K}\left(\frac{n}{N}\mathbf{K}\mathbf{K}+\frac{N-n}{N}\mathbf{D}\right)^{-1}\mathbf{K}\mathbf{t}_k \\
&= \frac{1}{N}\mathbf{t}_k^T\mathbf{t}_k - \frac{1}{N}\mathbf{t}_k^T\mathbf{K}\left(\mathbf{K}\mathbf{K}+\frac{N-n}{n}\mathbf{D}\right)^{-1}\mathbf{K}\mathbf{t}_k \\
&= 1 - \frac{1}{N}\mathbf{t}_k^T\mathbf{K}\mathbf{K}\left(\mathbf{K}\mathbf{K}+\frac{1-p}{p}\mathbf{D}\right)^{-1}\mathbf{t}_k \\
&\leq 1 - \frac{1}{N}\mathbf{t}_k^T\mathbf{K}\mathbf{K}\left(\mathbf{K}\mathbf{K}+\frac{1-p}{p}\mathbf{I}\right)^{-1}\mathbf{t}_k \\
&= 1 - \frac{1}{N}\mathbf{t}_k^T\mathbf{U}\mathbf{\Lambda}\left(\mathbf{\Lambda}+\frac{1-p}{p}\mathbf{I}\right)^{-1}\mathbf{U}^T\mathbf{t}_k, \qquad (43)
\end{aligned}
$$

where we have set $n = pN$, $\mathbf{\Lambda} = diag(\lambda_1,\ldots,\lambda_N)$ is a diagonal matrix containing the eigenvalues of $\mathbf{KK}$ (and thus $\mathbf{\Lambda} = diag(l_1^2,\ldots,l_N^2)$, where l_i is the i-th eigenvalue of $\mathbf{K}$) and $\mathbf{U} \in \mathbb{R}^{N\times N}$ is an orthogonal matrix containing the corresponding eigenvectors. As can be seen in (43), the training error bound of the AKELM algorithm has a regularization form. In the case where $p = 1$, the training error bound of AKELM is equal to zero. For any other value of $0 < p < 1$, the training error bound is a function of p and of the eigenvalues λ_i, $i = 1,\ldots,N$.

To illustrate the result of (43), consider the special case where $\mathbf{KK}$ has its eigenvalues equal to $\alpha n = \alpha p N$. In this case, the error bound of the AKELM algorithm is equal to:

$$
E[L(\mathbf{W}_{out},\mathbf{b})] - L(\mathbf{W}_{out},\mathbf{1}) \leq 1 - \frac{\alpha p^2 N}{\alpha p^2 N - p + 1}. \qquad (44)
$$

Thus, for any value $\epsilon > 0$, the minimum value of p satisfying $E[L(\mathbf{W}_{out},\mathbf{b})] - L(\mathbf{W}_{out},\mathbf{1}) \leq \epsilon$ is given by:

$$
p = \frac{(\epsilon-1) + \sqrt{(1-\epsilon)^2 + 4\alpha N(1-\epsilon)}}{2\alpha N}. \qquad (45)
$$

For example, for a training error bound $\epsilon = 0.001$, when $\alpha = 10$ and $N = 100000$, a value of $p = 0.001$ should be used.

REFERENCES

[1] G. Huang, Q. Zhu, and C. Siew, "Extreme Learning Machine: a new learning scheme of feedforward neural networks," *Proc IEEE Int Jt Conf Neural Netw*, vol. 2, pp. 985–990, 2004.

[2] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[3] M. Hagan and M. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Trans. Neural Netw*, vol. 5, no. 6, pp. 989–993, 1994.

[4] P. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[5] R. Zhang, Y. Lan, G. Huang, and Z. Zu, "Universal approximation of Extreme Learning Machine with adaptive growth of hidden nodes," *IEEE Trans Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 365–371, 2012.

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

[7] G. Huang, D. Wang, and Y. Lan, "Extreme Learning Machine: a survey," *Int. J. of Mach. Learn. and Cybern.*, vol. 2, no. 2, pp. 107–122, 2011.

[8] G. Huang, "Extreme Learning Machine," *Springer*, 2013.

[9] ——, "An insight into Extreme Learning Machines: Random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.

[10] J. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on bow framework," *IEEE Conf. Industr. Electr. Appl.*, 2014.

[11] A. Iosifidis, A. Tefas, and I. Pitas, "DropELM: Fast neural network regularization with Dropout and DropConnect," *Neurocomputing*, vol. 162, pp. 57–66, 2015.

[12] A. Iosifidis, "Extreme learning machine based supervised subspace learning," *Neurocomputing*, vol. 167, pp. 158–164, 2015.

[13] L. Zhang and D. Zhang, "Domain adaptation extreme learning machines for drift compensation in E-Nose systems," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 7, pp. 1790–1801, 2015.

[14] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern*, vol. 42, no. 2, pp. 513–529, 2012.

[15] Z. Bai, G. Huang, W. Wang, H. Wang, and M. Westover, "Sparse Extreme Learning Machine for classification," *IEEE Trans Cybern.*, vol. 44, no. 10, pp. 1858–1870, 2014.

[16] A. Iosifidis, A. Tefas, and I. Pitas, "On the kernel Extreme Learning Machine classifier," *Pattern Recogn. Lett.*, vol. 54, pp. 11–17, 2015.

[17] P. Drineas and M. Mahoney, "On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-based Learning," *J. Mach. Learn. Res.*, vol. 6, pp. 2153–2275, 2005.

[18] J. Smola and B. Scholkopf, "Sparse greedy matrix approximation for machine learning," *Int. Conf. Mach. Learn.*, 2000.

[19] C. Williams and M. Seeger, "The effect of the input density distribution on kernel-based classifiers," *Int. Conf. Mach. Learn.*, 2000.

[20] ——, "Using the Nystrom method to speed up kernel machines," *Adv. Neural Inf. Process. Syst.*, 2001.

[21] P. Drineas, R. Kannan, and M. Mahoney, "Fast Monte Carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix," *SIAM J. Comput.*, vol. 36, no. 1, pp. 158–183, 2006.

[22] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nystrom method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, 2004.

[23] R. Chitta, R. Jin, T. Havens, and A. Jain, "Approximate kernel k-means: solution to large scale kernel clustering," *Int. Conf. Knowl. Disc. and Data Mining*, 2011.

[24] P. Drineas, R. Kannan, and M. Mahoney, "Scalable kernel clustering: Approximate kernel k-means," *arXiv:1402.3849v1*, pp. 1–15, 2014.

[25] D. Achilioptas, G. McSherry, and B. Scholkopf, "Sampling techniques for kernel methods," *Adv. Neural Inf. Process. Syst.*, 2002.

[26] M. Belabbas and P. Wolfe, "Spectral methods in machine learning and new strategies for very large datasets," *Proc. Nat. Acad. of Sciences*, vol. 106, no. 2, pp. 369–374, 2009.

[27] A. Smola, Z. Ovari, and R. Williamson, "Regularization with dot-product kernels," *Adv. Neural Inf. Process. Syst.*, 2000.

[28] B. Scholkpf and A. Smola, "Learning with kernels," 2001, MIT Press.

[29] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans Neural Netw.*, vol. 12, no. 2, pp. 181–201, 2001.

[30] A. Iosifidis, A. Tefas, and I. Pitas, "Minimum variance Extreme Learning Machine for human action recognition," *IEEE Int. Conf. Acoust., Speech and Sign. Proc.*, 2014.

[31] G. Huang and L. Chen, "Convex incremental Extreme Learning Machine," *Neurocomputing*, vol. 70, no. 16, pp. 3056–3062, 2008.

[32] A. Iosifidis, A. Tefas, and I. Pitas, "Minimum class variance Extreme Learning Machine for human action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 11, pp. 1968–1979, 2013.

[33] R. Fletcher, *Practical Methods of Optimization: Volume 2 Constrained Optimization*. Wiley, 1981.

[34] B. Frenay and M. Verleysen, "Using svms with randomised feature spaces: An extreme learning approach," *Europ. Symp. Art. Neural Netw.*, 2010.

[35] B. Schölkopf, R. Herbrich, and A. Smola, "A generalized representer theorem," *Conference on Computational Learning Theory*, 2001.

[36] A. Argyriou, C. Micchelli, and M. Pontil, "When is there a representer theorem? vector versus matrix regularizers," *J. Mach. Learn. Res.*, vol. 10, pp. 2507–2529, 2009.

[37] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," *Comp. Vision and Patt. Recogn.*, 2011.

[38] S. Yan, D. Xu, B. Zhang, and H. Zhang, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, 2007.

[39] A. Iosifidis and M. Gabbouj, "On the kernel Extreme Learning Machine speedup," *Pattern Recogn. Lett.*, vol. 68, pp. 205–210, 2015.
[40] A. Iosifidis, A. Tefas, and I. Pitas, "Graph embedded extreme learning machine," *IEEE Trans. Cyb.*, vol. 46, no. 1, pp. 311–324, 2016.
[41] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with Gabor wavelets," *IEEE Int. Conf. on Autom. Face and Gest. Recogn.*, 1998.
[42] T. Kanade, Y. Tian, and J. Cohn, "Comprehensive database for facial expression analysis," *IEEE Int. Conf. on Autom. Face and Gest. Recogn.*, 2000.
[43] L. Yin, X. Wei, Y. Sun, J. Wang, and M. Rosato, "A 3d facial expression database for facial behavior research," *IEEE Int. Conf. on Autom. Face and Gest. Recogn.*, 2006.
[44] P. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. Prentice-Hall, 1982.
[45] A. Maronidis, D. Bolis, A. Tefas, and I. Pitas, "Improving subspace learning for facial expression recognition using person dependent and geometrically enriched training sets," *Neural Networks*, vol. 24, no. 8, pp. 814–823, 2011.
[46] S. Nikitidis, A. Tefas, and I. Pitas, "Subclass discriminant nonnegative matrix factorization for facial image analysis," *Pattern Recognition*, vol. 45, pp. 4080–4091, 2012.
[47] ——, "Projected gradients for subclass discriminant nonnegative subspace learning," *IEEE Trans Cybern.*, vol. 44, no. 12, pp. 2806–2819, 2014.
[48] ——, "Maximum margin projection subspace learning for visual data analysis," *IEEE Trans. Image Process.*, vol. 23, no. 10, pp. 4413–4425, 2014.
[49] M. Everingham, J. Sivic, and A. Zisserman, ""Hello! My name is...Buffy" - Automatic naming of characters in TV video," *Brit. Mach. Vision Conf.*, 2006.
[50] T. Ahonen, A. Hadid, and P. M., "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, 2006.