Graph Embedded Extreme Learning Machine

Alexandros Iosifidis, Member, IEEE, Anastasios Tefas, Member, IEEE, and Ioannis Pitas, Fellow, IEEE

***Abstract*— In this paper, we propose a novel extension of the Extreme Learning Machine algorithm for Single-hidden Layer Feedforward Neural network training that is able to incorporate Subspace Learning (SL) criteria on the optimization process followed for the calculation of the network's output weights. The proposed Graph Embedded Extreme Learning Machine (GEELM) algorithm is able to naturally exploit both intrinsic and penalty SL criteria that have been (or will be) designed under the Graph Embedding framework. In addition, we extend the proposed GEELM algorithm in order to be able to exploit SL criteria in arbitrary (even infinite) dimensional ELM spaces. We evaluate the proposed approach on eight standard classification problems and nine publicly available datasets designed for three problems related to human behaviour analysis, i.e., the recognition of human face, facial expression and activity. Experimental results denote the effectiveness of the proposed approach, since it outperforms other ELM-based classification schemes in all the cases.**

***Index Terms*— Extreme Learning Machine, Graph Embedding, Human action recognition, Facial Image Classification.**

I. Introduction

Extreme Learning Machine (ELM) is a relatively new algorithm for Single-hidden Layer Feedforward Neural (SLFN) networks training that leads to fast network training requiring low human supervision [1]. Conventional SLFN network training approaches, like the Back-propagation [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, gradient descend-based learning techniques are generally slow and may decrease the network's generalization ability, since the solution may be trapped in local minima. In ELM the input weights and the hidden layer bias values of the SLFN network are randomly assigned. By adopting the squared loss of the prediction error, 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. 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 SLFN networks trained by using the ELM algorithm 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].

Based on the observation that ELM optimization process can be seen from a Subspace Learning perspective [11], in this paper we propose a novel extension of the ELM algorithm that is able to exploit criteria employed by SL techniques for linear and nonlinear data projection. We formulate the ELM optimization problem as the one of learning the network output weights that provide a compromise between the minimization of both the network training error and the adopted SL criterion. This is achieved by incorporating an appropriate regularization term on the ELM optimization problem. In order to derive a compact solution that can be exploited for the calculation of the network output weights under several existing SL criteria (and in order to be able to design new ones) we design the proposed regularization term within the Graph Embedding framework [12], [13]. That is, it is assumed that the training data form the vertex set of an (intrinsic) graph $\mathcal{G}$ whose corresponding weight matrix expresses the relationships of the training data that are subject minimization. Furthermore, a penalty graph $\mathcal{G}^p$ can also be defined, whose corresponding weight matrix penalizes specific characteristics of the relationships between the training data. By using such an approach, an elegant interpretation of several linear and nonlinear SL techniques can be obtained. In addition, by exploiting new graph structures describing desired intrinsic and penalty data relationships, new SL techniques can be designed. Here it should be noted that previous work [11], [14], [15], [16] exploits only SL criteria that are subject to minimization in the ELM optimization process. In terms of the Graph Embedding framework, this means that only intrinsic graph structures have been investigated so far. In this paper we exploit network output weights solutions that exploit both intrinsic and penalty graph structures. Such an approach has the potential of a better generalization performance, when compared to the works in [11], [14], [15], [16].

We evaluate the proposed approach in three classification problems relating to human behaviour analysis, namely the recognition of human face, facial expression and activity. Experimental results on nine publicly available databases denote the effectiveness of the proposed algorithms. In these experiments, we exploit criteria used in two unsupervised and three supervised SL techinques, namely Laplacian Eigenmaps (LE) [17], Locally Linear Embedding (LLE) [18], Linear Discriminant Analysis (LDA) [19], Marginal Fisher Analysis (MDA) [12] and Local Fisher Discriminant Analysis (LFDA) [20]. In all the cases, we compare the performance of the proposed approach with that of the standard ELM [1], the Regularized ELM (RELM) [21], the Minimum Class Variance ELM (MCVELM) [11] and the Discriminant Graph Regularized ELM (DGRELM) [15] algorithms.

A. Iosifidis, A. Tefas and I. Pitas are with the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece. e-mail: {aiosif,tefas,pitas}@aiia.csd.auth.gr.

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).

The contributions of the paper are:

- A novel extension of the ELM algorithm that is able to exploit both intrinsic and penalty SL criteria in its optimization problem is proposed.
- The proposed Graph Embedded Extreme Learning Machine (GEELM) algorithm is also extended in order to exploit both intrinsic and penalty SL criteria in arbitrary-dimensional ELM spaces.
- We evaluate the proposed approach on eight standard classification problems and nine publicly available datasets relating to human behavior analysis, where its effectiveness is proven by comparing its performance with that of other relating classification methods.

The rest of the paper is organized as follows. We provide an overview of the related previous work in Section II. The proposed GEELM algorithm is described in Section III. An extension of the proposed GEELM algorithm that can be exploited in order to incorporate SL criteria in arbitrary-dimensional ELM spaces is described in Subsection III-A and a discussion on several aspects of the proposed GEELM algorithm including its time complexity and its connection to the Spectral Regression framework [22], [23]. Experimental results evaluating the performance of the proposed approach in face, facial expression and activity recognition problems are described in Section IV. Finally, conclusions are drawn in Section V.

II. Previous Work

In this section, we briefly describe the ELM, RELM and MCVELM algorithms proposed in [1], [21] and [11], respectively. Subsequently, we briefly describe the Graph Embedding framework [12].

Let us denote by $\{\mathbf{x}_i, c_i\}_{i=1,...,N}$ a set of N vectors $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding class labels $c_i \in \{1, \ldots, C\}$. We would like to use $\{\mathbf{x}_i, c_i\}_{i=1,...,N}$ in order to train a SLFN network. Such a network 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 [21], [11], 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 $t_{ik} = -1$ otherwise. 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 [21], [11] that, several activation functions $\Phi(\cdot)$ can be employed for the calculation of the network hidden layer outputs, like the sigmoid, sine, Gaussian, hard-limiting, Radial Basis Function (RBF) and the RBF-χ^2 ones. 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 $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N]$, equation (1) can be expressed in a matrix form as:

$$\mathbf{O} = \mathbf{W}_{out}^T \boldsymbol{\Phi}, \tag{2}$$

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

Standard ELM algorithm [1] assumes zero training error. That is, it is assumed that $\mathbf{o}_i = \mathbf{t}_i$, $i = 1, \ldots, N$, or by using a matrix notation $\mathbf{O} = \mathbf{T}$, where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$ is a matrix containing the network target vectors. By using (2), the network output weights $\mathbf{W}_{out}$ can be analytically calculated by:

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

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 given vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T \boldsymbol{\phi}_l, \tag{4}$$

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

The calculation of the network output weights $\mathbf{W}_{out}$ through (3) 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 [21], where the network output weights are calculated by solving the following optimization problem:

$$\textbf{Minimize:} \quad \mathcal{J}_{RELM} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \tag{5}$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T \boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i,\ i = 1, ..., N, \tag{6}$$

where $\boldsymbol{\xi}_i \in \mathbb{R}^C$ is the error vector corresponding to $\mathbf{x}_i$ and c is a parameter denoting the importance of the training error in the optimization problem, satisfying $c > 0$. Based on the Karush-Kuhn-Tucker (KKT) theorem [24], the network output weights $\mathbf{W}_{out}$ can be determined by solving the dual optimization problem:

$$\begin{aligned} \mathcal{J}_{D,RELM} &= \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \\ &- \sum_{i=1}^{N}\mathbf{a}_i\left(\mathbf{W}_{out}^T\boldsymbol{\phi}_i - \mathbf{t}_i + \boldsymbol{\xi}_i\right), \end{aligned} \tag{7}$$

which is equivalent to (5). 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:

$$\mathbf{W}_{out} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \frac{1}{c}\mathbf{I}\right)^{-1} \boldsymbol{\Phi}\mathbf{T}^T, \tag{8}$$

or

$$\mathbf{W}_{out} = \mathbf{\Phi}\left(\mathbf{K} + \frac{1}{c}\mathbf{I}\right)^{-1}\mathbf{T}^T, \quad (9)$$

where $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the *ELM kernel matrix*, having elements equal to $[\mathbf{K}]_{i,j} = \boldsymbol{\phi}_i^T\boldsymbol{\phi}_j$ [25]. In [21] it has been also shown that, by exploiting the kernel trick [26], [27], [28], $\mathbf{K}$ can be any kernel matrix defined over the input data $\mathbf{x}_i$. By using (9), the network response for a given vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T\boldsymbol{\phi}_l = \mathbf{T}\left(\mathbf{K} + \frac{1}{c}\mathbf{I}\right)^{-1}\mathbf{k}_l, \quad (10)$$

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

B. Extreme Learning Machine exploiting variance criteria

By allowing small training errors and trying to minimize both the norm of the network output weights and the (within-class/total) variance of the training vectors in the feature space determined by the network target vectors $\mathbb{R}^C$, $\mathbf{W}_{out}$ can be calculated by solving the following optimization problem:

$$\textbf{Minimize:} \quad \mathcal{J}_{MCVELM} = \frac{1}{2}\|\mathbf{S}_w^{\frac{1}{2}}\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2, \quad (11)$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T\boldsymbol{\phi}_i - \mathbf{t}_i = \boldsymbol{\xi}_i, \;\; i = 1, ..., N. \quad (12)$$

In the above, $\mathbf{S}_w$ is a matrix expressing either the variance of classes forming the classification problem [11], or the variance of the entire training set [15]. In the first case $\mathbf{S}_w$ is defined by:

$$\mathbf{S}_w = \sum_{j=1}^{C}\sum_{i=1}^{N}\frac{\beta_{ij}}{N_j}(\boldsymbol{\phi}_i - \boldsymbol{\mu}_j)(\boldsymbol{\phi}_i - \boldsymbol{\mu}_j)^T, \quad (13)$$

while in the second case, it is defined by:

$$\mathbf{S}_w = \sum_{i=1}^{N}(\boldsymbol{\phi}_i - \boldsymbol{\mu})(\boldsymbol{\phi}_i - \boldsymbol{\mu})^T. \quad (14)$$

In (13), β_{ij} is an index denoting if vector $\boldsymbol{\phi}_i$ belongs to class j, i.e., $\beta_{ij} = 1$, if $c_i = j$ and $\beta_{ij} = 0$ otherwise. $N_j = \sum_{i=1}^{N}\beta_{ij}$ is the number of training vectors belonging to class j and $\boldsymbol{\mu}_j = \frac{1}{N_j}\sum_{i=1}^{N}\beta_{ij}\boldsymbol{\phi}_i$ is the mean vector of class j. In (14), $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\phi}_i$ is the mean vector of the entire training set.

By substituting (12) in (11) and solving for $\frac{\vartheta\mathcal{J}_{MCVELM}}{\vartheta\mathbf{W}_{out}} = 0$, $\mathbf{W}_{out}$ is given by:

$$\mathbf{W}_{out} = \left(\mathbf{\Phi}\mathbf{\Phi}^T + \frac{1}{c}\mathbf{S}_w\right)^{-1}\mathbf{\Phi}\mathbf{T}^T. \quad (15)$$

By using (15), the network response for a given vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T\boldsymbol{\phi}_l = \mathbf{T}\mathbf{\Phi}^T\left(\mathbf{\Phi}\mathbf{\Phi}^T + \frac{1}{c}\mathbf{S}_w\right)^{-1}\boldsymbol{\phi}_l. \quad (16)$$

The calculation of the network output weights $\mathbf{W}_{out}$ through (15) is sometimes inaccurate, since the matrix $\left(\mathbf{\Phi}\mathbf{\Phi}^T + \frac{1}{c}\mathbf{S}_w\right)$ may be singular. In order to address this problem, an additional dimensionality reduction step performed by applying Principal Component Analysis on the network hidden layer outputs has been proposed in [11].

C. Graph Embedding

The Graph Embedding framework [12] assumes that the training data $\mathbf{x}_i$, $i = 1, \ldots, N$ are employed in order to form the vertex set of an undirected weighted graph $\mathcal{G} = \{\mathbf{X}, \mathbf{V}\}$, where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ and $\mathbf{V} \in \mathbb{R}^{N\times N}$ is a similarity matrix whose elements denote the relationships between the graph vertices $\mathbf{x}_i$. Furthermore, a penalty graph $\mathcal{G}^p = \{\mathbf{X}, \mathbf{V}^p\}$ can be defined, whose weight matrix $\mathbf{V}^p \in \mathbb{R}^{N\times N}$ penalizes specific relationships between the graph vertices $\mathbf{x}_i$.

In the case of linear data projections[1], the original data $\mathbf{x}_i \in \mathbb{R}^D$ are projected to a low-dimensional feature space $\mathbb{R}^d$, $d < D$, by applying a linear transformation, i.e., $\mathbf{s}_i = \mathbf{V}^T\mathbf{x}_i$. This can be achieved by optimizing for:

$$\begin{aligned}\mathbf{V}^* &= \underset{tr(\mathbf{V}^T\mathbf{X}\mathbf{C}\mathbf{X}^T\mathbf{V})=q}{argmin}\sum_{i,j=1}^{N}\|\mathbf{V}^T\mathbf{x}_i - \mathbf{V}^T\mathbf{x}_j\|_2^2 W_{ij} \\ &= \underset{tr(\mathbf{V}^T\mathbf{X}\mathbf{C}\mathbf{X}^T\mathbf{V})=q}{argmin} tr\left(\mathbf{V}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{V}\right), \quad (17)\end{aligned}$$

where $tr(\cdot)$ is the trace operator, q is a constant value (usually $q = 1$) and $\mathbf{L} \in \mathbb{R}^{N\times N}$ is the so-called graph Laplacian matrix defined as $\mathbf{L} = \mathbf{D} - \mathbf{V}$. $\mathbf{D}$ is the diagonal degree matrix having elements $D_{ii} = \sum_{j=1}^{N}V_{ij}$. $\mathbf{C} \in \mathbb{R}^{N\times N}$ is a constraint matrix that is used in order to avoid trivial solutions and is typically a diagonal matrix for scale normalization, or the graph Laplacian matrix of $\mathcal{G}^p$, that is $\mathbf{C} = \mathbf{L}^p = \mathbf{D}^p - \mathbf{V}^p$.

The solution of (17) is obtained by solving the generalized eigenvalue decomposition problem:

$$\mathbf{S}_i\mathbf{v} = \lambda\mathbf{S}_p\mathbf{v}, \quad (18)$$

where $\mathbf{S}_i = \mathbf{X}\mathbf{L}\mathbf{X}^T$ and $\mathbf{S}_p = \mathbf{X}\mathbf{C}\mathbf{X}^T$. That is, the columns of the transformation matrix $\mathbf{W}$ are formed by the eigenvectors of the matrix $\mathbf{S} = \mathbf{S}_p^{-1}\mathbf{S}_i$ corresponding to the d minimal eigenvalues λ_i. In the case where the matrix $\mathbf{S}_p$ is singular, its generalized pseudoinverse can be used instead, i.e., we can perform eigenanalysis to the matrix $\mathbf{S} = \mathbf{S}_p^{\dagger}\mathbf{S}_i$.

From the above, it can be seen that the matrix $\mathbf{S} = \mathbf{S}_p^{\dagger}\mathbf{S}_i$ can be employed in order to describe both intrinsic and penalty relationships between the training data. We employ $\mathbf{S}$ in order to devise an extension of the ELM algorithm that is able to naturally incorporate such SL criteria in its optimization process, as will be described in the next Section.

III. Graph Embedded Extreme Learning Machine

In this Section, we describe in detail the proposed Graph Embedded ELM (GEELM) algorithm. After the calculation of the network hidden layer outputs $\boldsymbol{\phi}_i, i = 1, \ldots, N$, we propose to calculate the network output weights $\mathbf{W}_{out}$ by optimizing for:

$$\begin{aligned}\textbf{Minimize:} \quad \mathcal{J}_{GEELM} &= \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \\ &+ \frac{\lambda}{2}tr\left(\mathbf{W}_{out}^T\mathbf{S}\mathbf{W}_{out}\right) \quad (19)\end{aligned}$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T\boldsymbol{\phi}_i - \mathbf{t}_i = \boldsymbol{\xi}_i, \;\; i = 1, ..., N, \quad (20)$$

[1]While nonlinear projections can also be exploited, we are interested in linear ones, since as has been explained in previous Sections the second step of ELM corresponds to a linear data mapping process. This is due to the ELM assumption that nonlinear data relationships can be well described by using appropriate hidden layer activation functions $\Phi(\cdot)$.

where $\mathbf{S}$ is a matrix expressing SL criteria, as described in subsection II-C, and λ is a trade off parameter betweeen the two regularization terms of $\mathbf{W}_{out}$, satisfying $\lambda > 0$. By substituting the constraints (20) in $\mathcal{J}_{GEELM}$ and solving for $\frac{\vartheta \mathcal{J}_{GEELM}}{\vartheta \mathbf{W}_{out}} = 0$, $\mathbf{W}_{out}$ is given by:

$$\mathbf{W}_{out} = \left(\mathbf{\Phi}\mathbf{\Phi}^T + \frac{1}{c}\Big(\mathbf{I} + \lambda \mathbf{S}\Big) \right)^{-1} \mathbf{\Phi}\mathbf{T}^T. \tag{21}$$

The calculation of the network output weights $\mathbf{W}_{out}$ through (21) can always be achieved, since the matrix $\Big(\mathbf{\Phi}\mathbf{\Phi}^T + \frac{1}{c}\left(\mathbf{I} + \lambda \mathbf{S}\right)\Big)$ is nonsingular.

After the calculation of the network output weights $\mathbf{W}_{out}$, the network response for a given vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T \boldsymbol{\phi}_l, \tag{22}$$

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

We can define the following two cases for the matrix $\mathbf{S}$:

- $\mathbf{S}$ is used in order to express intrinsic training data relationships (that is, $\mathbf{S}_p = \mathbf{I}$). In this case, we assume that the network hidden layer output vectors $\boldsymbol{\phi}_i$ corresponding to the training data $\mathbf{x}_i$, $i = 1, \ldots, N$ are used in order to form the vertex set of an (intrinsic) 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 $\boldsymbol{\phi}_i$. $\mathbf{S}$ 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 $D_{ii} = \sum_{j=1}^{N} V_{ij}$. It should be noted here that the calculation of $\mathbf{S}$ in the ELM space $\mathbb{R}^L$, rather than in the input space $\mathbb{R}^D$ has the advantage that nonlinear relationships between the input data $\mathbf{x}_i$ can be better expressed.
- $\mathbf{S}$ is used in order to express both intrinsic and penalty training data relationships. In this case, we assume that the network hidden layer output vectors $\boldsymbol{\phi}_i$ corresponding to the training data $\mathbf{x}_i$, $i = 1, \ldots, N$ are used in order to form the vertex set of an (intrinsic) 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 $\boldsymbol{\phi}_i$ that are subject minimization. Furthermore, a penalty graph $\mathcal{G} = \{\mathbf{\Phi}, \mathbf{V}^p\}$ is also defined, whose weight matrix $\mathbf{V}^p \in \mathbb{R}^{N \times N}$ penalizes relationships between the graph vertices $\boldsymbol{\phi}_i$ that are subject maximization. The matrices $\mathbf{S}_i = \mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^T$ and $\mathbf{S}_p = \mathbf{\Phi}\mathbf{L}^p\mathbf{\Phi}^T$ are defined, where $\mathbf{L}$, $\mathbf{L}^p$ are the graph Laplacian matrices of $\mathcal{G}$ and $\mathcal{G}^p$, respectively. $\mathbf{S}$ can be defined by using $\mathbf{S}_i$, $\mathbf{S}_p$ as $\mathbf{S} = \mathbf{S}_p^{-1}\mathbf{S}_i$. In the case where $\mathbf{S}_p$ is singular, the matrix $\mathbf{S}$ can be defined by $\mathbf{S} = \mathbf{S}_p^{\dagger}\mathbf{S}_i$. Similar to the first case, the calculation of $\mathbf{S}_i$, $\mathbf{S}_p$ in the ELM space $\mathbb{R}^L$, rather than in the input space $\mathbb{R}^D$ has the advantage that nonlinear intrinsic and penalty relationships between the input data $\mathbf{x}_i$ can be better expressed.

By following such an analysis, the proposed GEELM algorithm can naturally exploit SL criteria expressing both intrinsic and penalty relationships between the training data for $\mathbf{W}_{out}$ calculation. In our experiments we have exploited the ones adopted by LE [17], LLE [18], LDA [19], MDA [12] and LFDA [20] methods. In the following we describe the graph structures used in these cases.

Both LE and LLE construct a neighboring graph $\mathcal{G} = \{\mathbf{\Phi}, \mathbf{V}\}$ such that:

$$V_{ij} = \begin{cases} v_{ij} & \boldsymbol{\phi}_j \in \mathcal{N}_i, \\ 0, & otherwise, \end{cases} \tag{23}$$

where $\mathcal{N}_i$ denotes the neighborhood of $\boldsymbol{\phi}_i$. In our experiments we have used 5-NN graphs, that have been shown to provide satisfactory performance in many classification problems. In LE, v_{ij} is a measure of the similarity between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, like the heat kernel function defined by $v_{ij} = exp\left(-\frac{\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\|_2^2}{2\sigma^2}\right)$, where σ is a parameter scaling the Euclidean distance between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$. In LLE, v_{ij} is determined by applying a local fitting process solving for:

$$\min_{\sum_{j \in \mathcal{N}_i} v_{ij} = 1} \|\boldsymbol{\phi}_i - \sum_{\boldsymbol{\phi}_j \in \mathcal{N}_i} v_{ij} \left(\boldsymbol{\phi}_j - \boldsymbol{\phi}_i\right)\|_2^2. \tag{24}$$

In LDA, MDA and LFDA the structure of the corresponding graphs is determined by the labeling information that is available for the training vectors $\boldsymbol{\phi}_i$. LDA uses the following graph weights:

$$V_{ij} = \begin{cases} \frac{1}{N_{c_i}}, & c_j = c_i, \\ 0, & otherwise, \end{cases} \tag{25}$$

$$V_{ij}^p = \begin{cases} \frac{1}{N} - \frac{1}{N_{c_i}}, & c_j = c_i, \\ \frac{1}{N}, & otherwise, \end{cases} \tag{26}$$

for the intrinsic and penalty graphs, respectively. MDA uses graph weights defined by:

$$V_{ij} = \begin{cases} 1, & c_i = c_j \ and \ \boldsymbol{\phi}_j \in \mathcal{N}_i, \\ 1, & c_i = c_j \ and \ \boldsymbol{\phi}_i \in \mathcal{N}_j, \\ 0, & otherwise, \end{cases} \tag{27}$$

$$V_{ij}^p = \begin{cases} 1, & c_i \neq c_j \ and \ \boldsymbol{\phi}_j \in \mathcal{N}_i, \\ 1, & c_i \neq c_j \ and \ \boldsymbol{\phi}_i \in \mathcal{N}_j, \\ 0, & otherwise. \end{cases} \tag{28}$$

Finally, LFDA uses graph weights defined by:

$$V_{ij} = \begin{cases} \frac{v_{ij}}{N_{c_i}}, & c_j = c_i, \\ 0, & otherwise. \end{cases} \tag{29}$$

$$V_{ij}^p = \begin{cases} v_{ij}\left(\frac{1}{N} - \frac{1}{N_{c_i}}\right), & c_j = c_i, \\ \frac{1}{N}, & otherwise. \end{cases} \tag{30}$$

where v_{ij} is a measure of the similarity between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, like the heat kernel function. The processing steps followed for the calculation of the GEELM network parameters are illustrated in Algorithm 1.

In the above discussion, we assume that the network hidden layer outputs for the training vectors $\mathbf{x}_i$ can be calculated using $\Phi(\cdot)$. However, as has been discussed in subsection II-A, ELM can be formulated as a kernel machine, where the dimensionality of the ELM space is arbitrary (possibly infinite). We extend the proposed GEELM algorithm to this direction and describe a kernel algorithm in the following subsection. We refer to this version of the GEELM algorithm as Graph Embedded Kernel ELM (GEKELM), hereafter.

Algorithm 1 Graph Embedded Extreme Learning Machine

Input $\{\mathbf{x}_i, c_i\}$, $i = 1, \ldots, N$, L, $\Phi(\cdot)$.
Randomly assign input parameters $\mathbf{W}_{in}, \mathbf{b}$.
Calculate the network hidden layer outputs $\mathbf{\Phi}$.
Calculate graph weights $\mathbf{V}, \mathbf{V}^p$ from (23) - (30).
Calculate graph Laplacian matrices $\mathbf{L} = \mathbf{D} - \mathbf{V}$ and $\mathbf{L}^p = \mathbf{D}^p - \mathbf{V}^p$.
Calculate $\mathbf{S}_i = \mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^T$, $\mathbf{S}_p = \mathbf{\Phi}\mathbf{L}^p\mathbf{\Phi}^T$ and $\mathbf{S} = \mathbf{S}_p^{\dagger}\mathbf{S}_i$.
Calculate $\mathbf{W}_{out}$ from (21).
return $\mathbf{W}_{in}, \mathbf{b}, \mathbf{W}_{out}$.

A. Graph Embedded Kernel Extreme Learning Machine

In order to derive a kernel formulation of the proposed GEELM algorithm, we exploit the assumption that the network output weights $\mathbf{W}_{out}$ can be expressed as a linear combination of the training vectors (represented in the ELM space) [26], [27], [28], i.e.,:

$$\mathbf{W}_{out} = \sum_{i=1}^{N} \boldsymbol{\phi}_i \boldsymbol{\alpha}_i^T = \mathbf{\Phi}\mathbf{A}^T, \tag{31}$$

where $\mathbf{A} \in \mathbb{R}^{C \times N}$ is a matrix containing the reconstruction weights of $\mathbf{W}_{out}$, with respect to the training vectors in the ELM space.

By substituting (31) and (20) in (19) we obtain:

$$\begin{aligned} \mathcal{J}_{GEKELM} &= \frac{1}{2} tr\left(\mathbf{A}\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{A}^T\right) + \frac{c}{2} tr\left(\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{A}^T\mathbf{A}\mathbf{\Phi}^T\mathbf{\Phi}\right) \\ &- c\, tr\left(\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{A}^T\mathbf{T}\right) + tr\left(\mathbf{T}^T\mathbf{T}\right) \\ &+ \frac{\lambda}{2} tr\left(\mathbf{A}\mathbf{\Phi}^T\mathbf{S}\mathbf{\Phi}\mathbf{A}^T\right). \end{aligned} \tag{32}$$

Similar to the GEELM case, we can define the following two cases for the matrix $\mathbf{S}$:

- $\mathbf{S}$ is used to express intrinsic training data relationships. In this case, $\mathbf{S} = \mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^T$, 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 $D_{ii} = \sum_{j=1}^{N} V_{ij}$. In this case, (32) can be expressed as follows:

$$\begin{aligned} \mathcal{J}_{GEKELM} &= \frac{1}{2} tr\left(\mathbf{A}\mathbf{K}\mathbf{A}^T\right) + \frac{c}{2} tr\left(\mathbf{K}\mathbf{A}^T\mathbf{A}\mathbf{K}\right) \\ &- c\, tr\left(\mathbf{K}\mathbf{A}^T\mathbf{T}\right) + tr\left(\mathbf{T}^T\mathbf{T}\right) \\ &+ \frac{\lambda}{2} tr\left(\mathbf{A}\mathbf{K}\mathbf{L}\mathbf{K}\mathbf{A}^T\right). \end{aligned} \tag{33}$$

  By solving for $\frac{\vartheta \mathcal{J}_{GEKELM}}{\vartheta \mathbf{A}} = 0$, $\mathbf{A}$ is given by:

$$\mathbf{A} = \mathbf{T}\left(\frac{1}{c}\mathbf{I} + \mathbf{K}\left(\frac{\lambda}{c}\mathbf{L} + \mathbf{I}\right)\right)^{-1}. \tag{34}$$

- $\mathbf{S}$ is used to express both intrinsic and penalty training data relationships. In this case, $\mathbf{S} = \mathbf{S}_p^{\dagger}\mathbf{S}_i = \left(\mathbf{\Phi}\mathbf{L}^p\mathbf{\Phi}^T\right)^{\dagger}\mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^T$, where $\mathbf{L} \in \mathbb{R}^{N \times N}$, $\mathbf{L}^p \in \mathbb{R}^{N \times N}$ are the Laplacian matrices of the intrinsic and penalty graphs, respectively, defined by $\mathbf{L} = \mathbf{D} - \mathbf{V}$ and $\mathbf{L}^p = \mathbf{D}^p - \mathbf{V}^p$. $\mathbf{D}$, $\mathbf{D}^p$ are the diagonal degree matrices of $\mathcal{G}$ and $\mathcal{G}^p$, respectively. Thus, (32) can be expressed as follows:

$$\begin{aligned} \mathcal{J}_{GEKELM} &= \frac{1}{2} tr\left(\mathbf{A}\mathbf{K}\mathbf{A}^T\right) + \frac{c}{2} tr\left(\mathbf{K}\mathbf{A}^T\mathbf{A}\mathbf{K}\right) \\ &- c\, tr\left(\mathbf{K}\mathbf{A}^T\mathbf{T}\right) + tr\left(\mathbf{T}^T\mathbf{T}\right) \\ &+ \frac{\lambda}{2} tr\left(\mathbf{A}(\mathbf{L}^p)^{\dagger}\mathbf{L}\mathbf{K}\mathbf{A}^T\right). \end{aligned} \tag{35}$$

  By solving for $\frac{\vartheta \mathcal{J}_{GEKELM}}{\vartheta \mathbf{A}} = 0$, $\mathbf{A}$ is given by:

$$\mathbf{A} = \mathbf{T}\left(\frac{1}{c}\mathbf{I} + \frac{\lambda}{c}(\mathbf{L}^p)^{\dagger}\mathbf{L} + \mathbf{K}\right)^{-1}. \tag{36}$$

After the calculation of the reconstruction weight matrix $\mathbf{A}$, the network response for a given vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T\boldsymbol{\phi}_l = \mathbf{A}\mathbf{\Phi}^T\boldsymbol{\phi}_l = \mathbf{A}\mathbf{k}_l. \tag{37}$$

As can be seen by the above discussion, the proposed GEELM algorithm can also be considered as a kernel machine, where the dimensionality of the ELM space is arbitrary (even infinite). The processing steps followed for the calculation of the GEKELM network parameters are illustrated in Algorithm 2.

Algorithm 2 Graph Embedded Kernel Extreme Learning Machine

Input $\{\mathbf{x}_i, c_i\}$, $i = 1, \ldots, N$, $\kappa(\cdot, \cdot)$, $\mathbf{x}_t$.
Calculate the training kernel matrix $\mathbf{K}$ and test kernel vector $\mathbf{k}_t$ from $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and $k_{t,i} = \kappa(\mathbf{x}_i, \mathbf{x}_t)$.
Calculate graph weights $\mathbf{V}, \mathbf{V}^p$ from (23) - (30).
Calculate graph Laplacian matrices $\mathbf{L} = \mathbf{D} - \mathbf{V}$ and $\mathbf{L}^p = \mathbf{D}^p - \mathbf{V}^p$.
Calculate $\mathbf{A}$ from (36).
return $\mathbf{K}, \mathbf{k}_t, \mathbf{A}$.

B. Discussion

In this Section, we discuss several aspects of the proposed Graph Embedded ELM algorithms. We start by showing that the calculation of the graph Laplacian matrices in the ELM space $\mathbb{R}^L$ (or in the kernel space for the kernel formulation of the proposed GEKELM algorithm) rather in the input space $\mathbb{R}^D$ has the advantage that nonlinear relationships between the input data $\mathbf{x}_i$ can be exploited, enhancing performance. We have created a synthetic dataset consisting of three classes, formed by 200 1000D data each and following Normal distributions. The classes are centered at $-\mathbf{1}$, $\mathbf{0}$ and $\mathbf{1}$, respectively, and have standard deviations equal to $0.8 \cdot \mathbf{1}$, where $\mathbf{1}, \mathbf{0} \in \mathbb{R}^{1000}$ are vectors having elements equal to 1 and 0, respectively.

We have applied the proposed GEELM algorithms on the above-described synthetic dataset. We have randomly chosen 100 samples per class in order to train the competing algorithms and used the remaining samples for testing their performance. We have used $L = 100$ hidden neurons and optimized the regularization parameters c, λ by applying two-fold cross-validation on the training set. The obtained experimental results are illustrated in Table I. As can be seen in this

TABLE I

PERFORMANCE OF GEELM METHODS ON THE SYNTHETIC DATASET.

	Input space $\mathbf{R}^D$	ELM space $\mathbf{R}^L$
GEELM (LE)	73%	**74.33**%
GEELM (LLE)	83.33%	**85.67**%
GEELM (LDA)	73%	73%
GEELM (MDA)	85%	**88.33**%
GEELM (LFDA)	74.33%	**89**%

TABLE II

PERFORMANCE OF ELM METHODS ON THE SYNTHETIC DATASET USING DATA REPRESENTATIONS IN DIFFERENT SPACES.

Space	RELM	GEELM (LDA)	GEELM (MDA)	GEELM (LFDA)
Input	71.66%	73%	88.33%	89%
LDA	73.66%	74.33%	89%	89%
MDA	73.33%	73.66%	88.66%	89.33%
LFDA	74.33%	74.33%	89%	89%

Table, the calculation of the matrices used in the ELM methods exploiting graph structures used in LE, LLE, MDA and LFDA in the ELM space $\mathbf{R}^L$ leads to an enhanced performance, when compared to the case where the corresponding matrices calculation is performed in the input space $\mathbf{R}^D$. It can also be seen that in the LDA case, the performance obtained by following the two approaches is the same. This can be explained by the fact that in this case the corresponding graph Laplacian matrices are calculated based on the training data labels only and, thus, the corresponding ELM solutions are identical.

Subsequently, in order to observe the effect of applying the ELM variants in different feature spaces, we repeated the above-described experiment using data representations in the input space, as well as in feature spaces determined by applying LDA, MDA and LFDA on the training data. The classification rates obtained by applying RELM and the proposed ELM methods exploiting graph structures used in LDA, MDA and LFDA are illustrated in Table II. As can be seen in this Table, the adoption of data representations in different feature spaces may impact the performance of the RELM algorithm, since the adoption of discriminant data representations obtained by applying LDA, MDA and LFDA on the training data enhances its performance. A similar result is observed also for the proposed ELM methods exploiting graph structures for the calculation of the network parameters.

Regarding the time complexity of the proposed algorithms, for general graph structures we have the following two cases:

- For the GEELM solution given in (21), one needs to calculate the matrix $\mathbf{S}$ first. In the case where $\mathbf{S}$ takes into account only intrinsic graph structures, the time complexity of this process is equal to $O(L^2N)$. In the case where $\mathbf{S}$ takes into account both intrinsic and penalty graph structures, the time complexity of this process is equal to $O(L^3+L^2N)$. Subsequently, a matrix inversion and two matrix multiplication steps having time complexity equal to $O(L^3+L^2N)$ are required. Thus, the time complexity of the proposed GEELM algorithms in the case of general graph structures is equal to $O(L^3+L^2N)$.
- For the GEKELM solution given in (34), one needs to calculate the matrix $\mathbf{L}$ first, having time complexity equal to $O(N^2D)$. For the the GEKELM solution given in (36), one needs to calculate the matrix $(\mathbf{L}^p)^\dagger\mathbf{L}$ first, having time complexity equal to $O(N^3+N^2D)$. Subsequently, a matrix inversion and a matrix multiplication steps having time complexity equal to $O(N^3+N^2)$ are required. Thus, the time complexity of (34) is equal to $O(N^3+N^2D)$.

It should be noted here that, in the case of graph structures defined in several SL methods (like LDA and PCA) the calculation of the corresponding graph Laplacian matrices is performed by taking into account only the labeling information of the training data and, thus, the time complexity of the corresponding GEELM algorithms can be considered to be equal to that of the regularized ELM solution [21].

Finally, we show how the proposed methods can be interpreted in terms of a related framework, i.e., Spectral Regression [22], [23]. Let us consider the case of the two-class classification problem expressed by the outputs of the k-th output neuron of the ELM network. The network output weight associated with the k-th output neuron can be obtained by applying a two-step process:

- Solution of the generalized eigenanalysis problem $\mathbf{S}_i\mathbf{q} = \mu\mathbf{S}_p\mathbf{q}$. This process will lead to the determination of a data projection matrix $\mathbf{Q} \in \mathbb{R}^{L\times d}$, where d is the minimum rank of the matrices $\mathbf{S}_i, \mathbf{S}_p$. The matrix $\mathbf{Q}$ can be used in order to map the training data to the space $\mathbb{R}^d$ by applying $\mathbf{Y} = \mathbf{Q}^T\mathbf{\Phi}$.
- Calculation of the the vector $\mathbf{w}_k$ which satisfies $\mathbf{w}_k^T\mathbf{Y} = \mathbf{t}_k$, where $\mathbf{t}_k$ is a vector containing the target values for the k-th binary classification problem. In reality, such $\mathbf{w}_k$ may not exist. A possible way is to find the $\mathbf{w}_k$ which can best fit the equation in the least squares sense:

$$\mathbf{w}_k = \underset{\mathbf{a}_k}{arming} \sum_{i=1}^{N}(\mathbf{w}_k^T\mathbf{y}_i - t_{ki})^2, \tag{38}$$

where t_{ki} is the target value for the i-th training sample. In the case where $N < d$, this problem is ill posed. The most popular way to solve this problem is to impose a penalty on the norm of $\mathbf{w}_k$, i.e.:

$$\mathbf{w}_k = \underset{\mathbf{w}_k}{arming} \left(\sum_{i=1}^{N}(\mathbf{w}_k^T\mathbf{y}_i - t_{ki})^2 + a\|\mathbf{w}_k\|_2^2 \right). \tag{39}$$

The optimal solution of (39) is given by:

$$\mathbf{w}_k = \left(\mathbf{Y}\mathbf{Y}^T + a\mathbf{I}\right)^{-1}\mathbf{Y}\mathbf{t}_k. \tag{40}$$

The network output weights can be subsequently obtained by using $\mathbf{W}_{out} = [\mathbf{Q}\mathbf{w}_1, \ldots, \mathbf{Q}\mathbf{w}_C]$.

IV. EXPERIMENTS

In this section, we present experiments conducted in order to evaluate the performance of the proposed GEELM algorithm. We have employed eight standard classification problem datasets and nine datasets relating to human behavior analysis to this end, namely the ORL, AR and Extended YALE-B (face recognition), the COHN-KANADE, BU and JAFFE (facial

expression recognition) and the Hollywood2, Olympic Sports and Hollywood 3D (human action recognition) datasets. A brief description of the datasets is provided in the following subsections. Experimental results are provided in subsection IV-E. In all our experiments we compare the performance of the GEELM-based classification schemes with that of relating classification schemes, i.e., ELM [1], RELM [21], Kernel ELM (KELM) [21], MCVELM [11] and Discriminative Graph Regularized ELM (DGRLEM) [14] based classification.

In all the experiments on standard classification problems, we have employed the RBF kernel function for the ELM methods exploiting a kernel formulation:

$$\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\Big), \tag{41}$$

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

$$\Phi(\mathbf{x}_i, \mathbf{q}_j, b) = exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{q}_j\|_2^2}{2b^2}\Big), \tag{42}$$

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

In all the experiments involving facial image classification, we have resized the facial images provided by the databases in 40×30 pixel images and vectorized these images in order to create vectors $\mathbf{x}_i \in \mathbb{R}^{1200}$ and used the RBF kernel function and RBF activation function given in (41) and (42), respectively.

In human action recognition, we use the state-of-the-art method proposed in [29] as a baseline approach. We employ the Bag of Words (BoW)-based video representation by using HOG, HOF, MBHx, MBHy and (normalized) Trajectory descriptors evaluated on the trajectories of densely sampled interest points. We follow [29] and use 4000 codewords for each BoW representation. Classification is performed by employing the RBF-χ^2 kernel [30], where different descriptors are combined by concatenating the five video representations:

$$\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = exp\Big(- \frac{1}{2A} \sum_{d=1}^{D} \frac{(x_{id} - x_{jd})^2}{x_{id} + x_{jd}}\Big). \tag{43}$$

A is set to the mean value of the χ^2 distances between the training vectors. In the case of ELM formulations exploiting random input weights $\mathbf{q}_j$, we have employed the multi-channel RBF-χ^2 activation function:

$$\Phi(\mathbf{x}_i, \mathbf{q}_j, b) = exp\Big(- \frac{1}{2b} \sum_{d=1}^{D} \frac{(x_{id} - q_{jd})^2}{x_{id} + q_{jd}}\Big). \tag{44}$$

where the value b has been set equal to the mean χ^2 distance between the training vectors $\mathbf{x}_i$ and the network input weights $\mathbf{q}_j$.

The number of hidden layer neurons has been set equal to $L = 1000$ for all the ELM methods exploiting random hidden layer parameters, a value that has been shown to provide satisfactory performance in many classification problems [21], [11]. For fair comparison, in all the experiments, we make sure that the the same ELM space is used in all the ELM variants. That is, we first map the training data in the ELM space and, subsequently, calculate the network output weights according to each ELM algorithm. Regarding the optimal values of the regularization parameters (c, λ) used in different ELM-based classification schemes, they have been determined by following a grid search strategy, which is applied on the training data, e.g., on the four splits used for training in the face datasets where the five-fold cross-validation process is applied. The values used in our experiments are: $c = 10^r$, $r = -6, \ldots, 6$ and $\lambda = 10^p$, $p = -6, \ldots, 6$. All the experiments have been conducted on a 4-core, i7 - 4790, 3.6GHz PC with 32GB RAM using a MATLAB implementation.

A. Standard classification problem datasets

We have employed eight publicly available datasets from the machine learning repository of University of California Irvine (UCI) [31]. Table III provides information concerning the data sets used in our experiments.

TABLE III
UCI DATASETS DETAILS.

Dataset	Samples	Dimensions (D)	Classes (C)
Column	310	6	3
Glass	241	9	6
Indians	768	8	2
Libras	360	90	2
Relax	182	12	2
Spectf	267	44	2
Synth.Cont.	600	60	6
TicTacToe	958	9	2

B. Face recognition datasets

1) The ORL dataset [32]: consists of 400 facial images depicting 40 persons. The images were captured at different times and with different conditions, in terms of lighting, facial expressions (smiling/not smiling) and facial details (open/closed eyes, with/without glasses). Facial images were taken in frontal position with a tolerance for face rotation and tilting up to 20 degrees. Example images of the dataset are illustrated in Figure 1.

Fig. 1. *Facial images depicting a person of the ORL dataset.*

2) The AR dataset [33]: consists of over 4000 facial images depicting 70 male and 56 female faces. In our experiments we have used the preprocessed (cropped) facial images provided by the database, depicting 100 persons (50 males and 50 females) having a frontal facial pose, performing several expressions (anger, smiling and screaming), in different illumination conditions (left and/or right light) and with some occlusions (sun glasses and scarf). Example images of the dataset are illustrated in Figure 2.

Fig. 2. *Facial images depicting a person of the AR dataset.*

3) The Extended YALE-B dataset [34]: consists of facial images depicting 38 persons in 9 poses, under 64 illumination conditions. In our experiments we have used the frontal cropped images provided by the database. Example images of the dataset are illustrated in Figure 3.

Fig. 3. *Facial images depicting a person of the Extended YALE-B dataset.*

C. Facial expression recognition datasets

1) The COHN-KANADE dataset [35]: consists of facial images depicting 210 persons of age between 18 and 50. We have randomly selected 35 images for each facial expression, i.e., anger, disgust, fear, happyness, sadness, surprise and neutral. Example images of the dataset are illustrated in Figure 4.

Fig. 4. *Facial images from the COHN-KANADE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

2) The BU dataset [36]: consists of facial images depicting over 100 persons with a variety of ethnic/racial background, including White, Black, East-Asian, Middle-east Asian, Hispanic Latino and others. 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 5.

Fig. 5. *Facial images depicting a person of the BU dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

3) The JAFFE dataset [37]: consists of 210 facial images depicting 10 Japanese female persons. Each of the persons is depicted in 3 images for each expression. Example images of the dataset are illustrated in Figure 6.

Fig. 6. *Facial images depicting a person of the JAFFE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

D. Action recognition datasets

1) The Hollywood2 dataset [38]: consists of 1707 videos depicting 12 actions. The videos have been collected from 69 different Hollywood movies. Example video frames of the dataset are illustrated in Figure 7. We used the standard training-test split provided by the database (823 videos are used for training and performance is measured in the remaining 884 videos). Training and test videos come from different movies. The performance is evaluated by computing the average precision (AP) for each action class and reporting the mean AP over all classes (mAP) [39].

Fig. 7. *Video frames of the Hollywood2 dataset depicting instances of all the twelve actions.*

2) The Olympic Sports dataset [40]: consists of 783 videos depicting athletes practicing 16 sports. Example video frames of the dataset are illustrated in Figure 8. We used the standard training-test split provided by the database (649 videos are used for training and performance is measured in the remaining 134 videos). The performance is evaluated by computing the mean Average Precision (mAP) over all classes [39].

Fig. 8. *Video frames of the Olympic Sports dataset depicting instances of all the sixteen actions.*

3) The Hollywood 3D dataset [41]: consists of 951 video pairs (left and right channel) depicting 13 actions and a 'no action' class collected from Hollywood movies. Example video frames of this dataset are illustrated in Figure 9. We used the standard (balanced) training-test split provided by the database (643 videos are used for training and performance is measured in the remaining 308 videos). Training and test videos come from different movies. The performance is evaluated by computing both the mean AP over all classes (mAP) [39].

TABLE IV

PERFORMANCE FOR ELM METHODS EXPLOITING RANDOM PARAMETERS ON UCI DATASETS.

Dataset	ELM	RELM	MCVELM	DGRELM	**GEELM (LE)**	**GEELM (LLE)**	**GEELM (LDA)**	**GEELM (MDA)**	**GEELM (LFDA)**
Column	39.68%	80.11%	82.11%	82.11%	82.11%	82.11%	**82.67%**	**82.56%**	**82.56%**
Glass	40.65%	64.15%	65.67%	65.67%	**66.25%**	**66.03%**	**66.25%**	**66.76%**	**66.25%**
Indians	56.12%	70.99%	71.47%	72.5%	**72.68%**	**72.76%**	**72.78%**	**72.76%**	**72.76%**
Libras	61.39%	80.56%	80.83%	80.56%	80.56%	**80.83%**	**82.22%**	**82.22%**	**82.22%**
Relax	45.05%	50%	50.58%	50.58%	50.38%	**50.77%**	**52.88%**	**53.27%**	**53.27%**
Spectf	62.92%	62.35%	63.63%	63.4%	63.4%	**75.21%**	**66.9%**	**65.01%**	**64.27%**
Synth.Con.	87.67%	96.67%	96.83%	96.83%	96.83%	**97%**	**97%**	**97.33%**	**97.33%**
TicTacToe	89.25%	97.59%	97.66%	97.59%	**97.74%**	**97.74%**	**98.04%**	**99.02%**	**99.02%**

TABLE V

PERFORMANCE FOR ELM METHODS EXPLOITING KERNEL FORMULATIONS ON UCI DATASETS.

Dataset	KELM	**GEKELM (LE)**	**GEKELM (LLE)**	**GEKELM (LDA)**	**GEKELM (MDA)**	**GEKELM (LFDA)**
Column	82.22%	82.22%	82.22%	**83.22%**	**84.11%**	**84.11%**
Glass	63.79%	**64.15%**	**64.55%**	**64.55%**	**69.99%**	**68.66%**
Indians	71.99%	**72.17%**	**72.19%**	**74.07%**	**74.07%**	**74.96%**
Libras	83.06%	**85%**	**85%**	**85.83%**	**86.39%**	**86.39%**
Relax	50%	**57.88%**	**56.15%**	**52.5%**	**59.04%**	**54.42%**
Spectf	66.9%	66.9%	66.9%	**68.75%**	**75.21%**	**75.21%**
Synth.Con.	96.83%	**97.83%**	**97.83%**	**97.67%**	**97.67%**	**97.67%**
TicTacToe	98.09%	**99.02%**	**99.1%**	**99.25%**	**99.25%**	**99.77%**

TABLE VI

TRAINING TIMES (SECONDS) OF ELM METHODS EXPLOITING RANDOM PARAMETERS ON UCI DATASETS.

Dataset	ELM	RELM	MCVELM	DGRELM	**GEELM (LE)**	**GEELM (LLE)**	**GEELM (LDA)**	**GEELM (MDA)**	**GEELM (LFDA)**
Column	0.038	0.092	0.134	0.133	0.179	0.186	0.292	0.314	0.346
Glass	0.026	0.087	0.122	0.121	0.168	0.177	0.293	0.273	0.288
Indians	0.192	0.171	0.267	0.267	0.261	0.275	0.414	0.512	0.493
Libras	0.055	0.098	0.155	0.154	0.195	0.193	0.32	0.337	0.346
Relax	0.017	0.08	0.113	0.112	0.172	0.168	0.281	0.307	0.314
Spectf	0.036	0.096	0.135	0.137	0.175	0.178	0.37	0.363	0.355
Synth.Con.	0.124	0.122	0.192	0.191	0.219	0.237	0.371	0.427	0.486
TicTacToe	0.259	0.149	0.298	0.298	0.299	0.329	0.419	0.61	1.72

TABLE VII

TRAINING TIMES (SECONDS) OF ELM METHODS EXPLOITING KERNEL FORMULATIONS ON UCI DATASETS.

Dataset	KELM	**GEKELM (LE)**	**GEKELM (LLE)**	**GEKELM (LDA)**	**GEKELM (MDA)**	**GEKELM (LFDA)**
Column	0.002	0.007	0.011	0.014	0.028	0.025
Glass	0.001	0.004	0.006	0.008	0.016	0.014
Indians	0.019	0.064	0.081	0.385	0.194	0.21
Libras	0.004	0.011	0.014	0.023	0.041	0.039
Relax	0.002	0.004	0.006	0.008	0.011	0.011
Spectf	0.002	0.005	0.008	0.011	0.019	0.019
Synth.Con.	0.017	0.036	0.046	0.12	0.114	0.108
TicTacToe	0.032	0.114	0.14	0.363	0.382	0.322

Fig. 9. *Video frames of the Hollywood 3D dataset depicting instances of twelve actions.*

E. Experimental Results

In our first set of experiments, we have applied the competing algorithms on the UCI datasets. Since there is not a widely adopted experimental protocol for these datasets, we perform the five-fold cross-validation procedure [42] by taking into account the class labels of the data. That is, we randomly split the data belonging to each class in five sets and we use four splits of all the classes for training and the remaining splits for evaluation. This process is performed five times, one for each evaluation split. On each fold, we have normalized the training data to have zero mean and unit standard deviation. Test data were normalized accordingly. Experimental results obtained by applying the competing algorithms are illustrated in Tables IV and V for the ELM methods exploiting random parameters and kernel formulations, respectively. As can be seen in these Tables, the incorporation of the local class information on the ELM optimization process enhances classification performance, since the GEELM methods exploiting

graph structures used in LLE outperform the ELM, RELM, MCVELM and DGRELM methods in most cases. In addition, the incorporation of SL criteria expressing both intrinsic and penalty data relationships provides classification performance. Comparing the obtained classification rates, it can be seen that the kernel ELM formulations outperform the original ones in most cases. In Tables VI and VII, we also provide the mean training times of each ELM variant on the UCI datasets. As expected, the computational cost of the proposed methods is higher, when compared to the original ones. This is due to the fact that the proposed methods require the computation of the matrices expressing the corresponding SL criteria.

In our second set of experiments, we have applied the competing algorithms on the face recognition datasets. Details of these datasets are illustrated in Table VIII. Since there is not a widely adopted experimental protocol for these datasets, we perform the five-fold cross-validation procedure [42] by taking into account the ID labels of the persons on each database. That is, we randomly split the facial images depicting the same person in five sets and we use four splits of all the persons for training and the remaining splits for evaluation. This process is performed five times, one for each evaluation split. Experimental results obtained by applying the competing algorithms are illustrated in Tables IX and X for the ELM methods exploiting random parameters and kernel formulations, respectively. As can be seen in these Tables, the incorporation of the local class information on the ELM optimization process enhances facial image classification (in terms of face recognition), since the GEELM methods exploiting graph structures used in LE and LLE outperform the ELM, RELM, MCVELM and DGRELM methods in all the cases. In addition, the incorporation of SL criteria expressing both intrinsic and penalty data relationships provides enhanced facial image classification performance. Comparing the obtained classification rates, it can be seen that the kernel ELM formulations outperform the original ones in most cases. In Tables XI and XII, we also provide the mean training times of each ELM variant on the face recognition datasets.

TABLE VIII
FACIAL IMAGE DATASETS DETAILS.

Dataset	Samples	Dimensions (D)	Classes (C)
AR	2600	1200	100
ORL	400	1200	40
Yalle	2432	1200	38
BU	700	1200	7
JAFFE	210	1200	7
KANADE	245	1200	7

In Table XIII, we also compare the performance obtained by following the proposed approach with that of other, recently proposed state-of-the-art, methods evaluating their performance on the AR, ORL and Yalle databases using the same experimental protocol. As can be seen, the proposed approach achieves satisfactory performance, outperforming the remaining methods, in all cases.

In our third set of experiments, we have applied the competing algorithms on the facial expression recognition datasets. Details of these datasets are illustrated in Table VIII. Since there is not a widely adopted experimental protocol for these datasets too, we apply the five-fold cross-validation procedure [42] by employing the facial expression labels. That is, we randomly split the facial images depicting the same expression in five sets and we use four splits of all the expressions for training and the remaining splits for evaluation. This process is performed five times, one for each evaluation split. Experimental results obtained by applying the competing algorithms are illustrated in Tables IX and X for the ELM methods exploiting random parameters and kernel formulations, respectively. As can be seen in these Tables, the incorporation of local class information on the ELM optimization process enhances facial image classification (in terms of facial expression recognition), since the GEELM methods exploiting graph structures used in LE and LLE outperform the ELM, RELM, MCVELM and DGRELM methods in all the cases. The incorporation of SL criteria expressing both intrinsic and penalty data relationships seems not to enhance performance, when compared to the GEELM methods exploiting graph structures used in LE and LLE. However, it can be seen that, the incorporation of SL criteria expressing both intrinsic and penalty data relationships enhances performance when compared to the ELM, RELM, KELM, MCVELM and DGRELM methods in all the cases. In addition, it can be see in Tables IX and X, the kernel formulations of the ELM networks generally achieve the highest performance. The confusion matrices obtained by applying the KELM and the proposed GEKELM algorithms are provided in Figure 10. In Tables XI and XII, we also provide the mean training times of each ELM variant on the facial expression recognition datasets.

TABLE XIII
COMPARISON OF OUR RESULTS WITH SOME STATE-OF-THE-ART METHODS ON THE AR, ORL AND YALLE DATASETS.

	AR	ORL	Yalle
Method [43]	95.7%	-	98.1%
Method [44]	-	94.35%	94.76%
Method [45]	97%	-	-
Method [46]	74.67%	83.89%	-
Method [47]	-	98.93%	-
Method [48]	-	-	97.2%
Proposed method	**99.81**%	**99**%	**98.52**%

In Table XIV, we also compare the performance obtained by applying the proposed approach with that of other, recently proposed state-of-the-art, methods evaluating their performance on the BU, Jaffe and Kanade databases. As can be seen, the proposed approach achieves satisfactory performance in all the cases. Specifically, it can be seen that the proposed approach outperforms [49] and [48] in all the cases, while the method in [50] provides the best performance. This may be explained by the fact that the resolution of the facial images used in [50] 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 the proposed approach outperforms MMP and SVM-based facial image classification.

Finally, we have applied the competing algorithms on the human action recognition datasets. We have employed the

TABLE IX

PERFORMANCE FOR ELM METHODS EXPLOITING RANDOM PARAMETERS.

Dataset	ELM	RELM	MCVELM	DGRELM	**GEELM (LE)**	**GEELM (LLE)**	**GEELM (LDA)**	**GEELM (MDA)**	**GEELM (LFDA)**
AR	97.26%	98.58%	98.81%	99%	99%	**99.12%**	**99.23%**	**99.12%**	**99.27%**
ORL	92.25%	96.75%	98%	98%	98%	**98.25%**	**98.25%**	**98.5%**	**98.5%**
Yalle	97.25%	97.25%	97.74%	98.15%	98.15%	98.15%	**98.23%**	**98.23%**	**98.31%**
BU	35.29%	65.29%	66.29%	66.29%	**67.14%**	**66.57%**	**66.71%**	**66.71%**	**67.14%**
JAFFE	43.33%	50.95%	50.95%	51.43%	50.95%	50.95%	51.43%	51.43%	**55.24%**
KANADE	64.49%	72.65%	73.06%	73.06%	73.06%	73.06%	**73.47%**	**73.47%**	73.06%
Ol.Sports	63.33%	81.79%	81.9%	81.85%	**82.42%**	**81.98%**	**82.22%**	**82.08%**	**82.12%**
Holl.2	21.54%	54.49%	55.03%	55.08%	**55.14%**	**55.24%**	55.05%	55.08%	**55.2%**
Holl.3D	23.12%	27.68%	27.82%	27.8%	**27.92%**	**27.94%**	**28.98%**	**28.66%**	**29.17%**

TABLE X

PERFORMANCE FOR ELM METHODS EXPLOITING KERNEL FORMULATIONS.

Dataset	KELM	**GEKELM (LE)**	**GEKELM (LLE)**	**GEKELM (LDA)**	**GEKELM (MDA)**	**GEKELM (LFDA)**
AR	99.42%	**99.62%**	**99.69%**	**99.69%**	**99.69%**	**99.81%**
ORL	98.5%	**98.75%**	98.5%	**98.75%**	**99%**	**99%**
Yalle	98.15%	**98.52%**	**98.44%**	**98.52%**	**98.44%**	**98.52%**
BU	67.43%	67.43%	**67.57%**	**67.57%**	**67.86%**	**67.86%**
JAFFE	55.71%	**57.14%**	55.71%	55.71%	**57.14%**	**57.14%**
KANADE	67.35%	**68.57%**	67.35%	**67.76%**	**70.61%**	**69.8%**
Olympic Sports	88.85%	**88.92%**	**88.94%**	**89.74%**	**88.97%**	**89.12%**
Hollywood2	61.34%	**62.07%**	**62.07%**	**62.5%**	**62.5%**	**62.5%**
Hollywood 3D	31.14%	31.14%	**31.34%**	**31.8%**	**31.23%**	**31.79%**

TABLE XI

TRAINING TIMES (SECONDS) OF ELM METHODS EXPLOITING RANDOM PARAMETERS.

Dataset	ELM	RELM	MCVELM	DGRELM	**GEELM (LE)**	**GEELM (LLE)**	**GEELM (LDA)**	**GEELM (MDA)**	**GEELM (LFDA)**
AR	1.583	2.05	2.887	2.858	3.214	3.446	4.756	4.67	4.29
ORL	0.151	0.21	0.307	0.307	0.486	0.498	0.667	1.714	1.676
Yalle	1.558	1.53	2.812	2.131	2.867	2.118	2.065	2.446	2.125
BU	0.334	0.38	1.666	0.499	1.502	1.514	0.807	1.729	1.899
JAFFE	0.077	0.189	0.227	0.236	0.341	0.342	0.579	1.709	1.696
KANADE	0.087	0.179	0.233	0.238	0.449	0.45	0.608	1.709	1.6295
Ol.Sports	0.271	0.166	0.3285	0.3287	0.587	0.514	0.692	0.862	0.809
Holl.2	0.205	0.143	0.342	0.342	0.421	0.459	0.593	0.872	0.817
Holl.3D	0.173	0.137	0.295	0.296	0.564	0.505	0.595	0.762	0.884

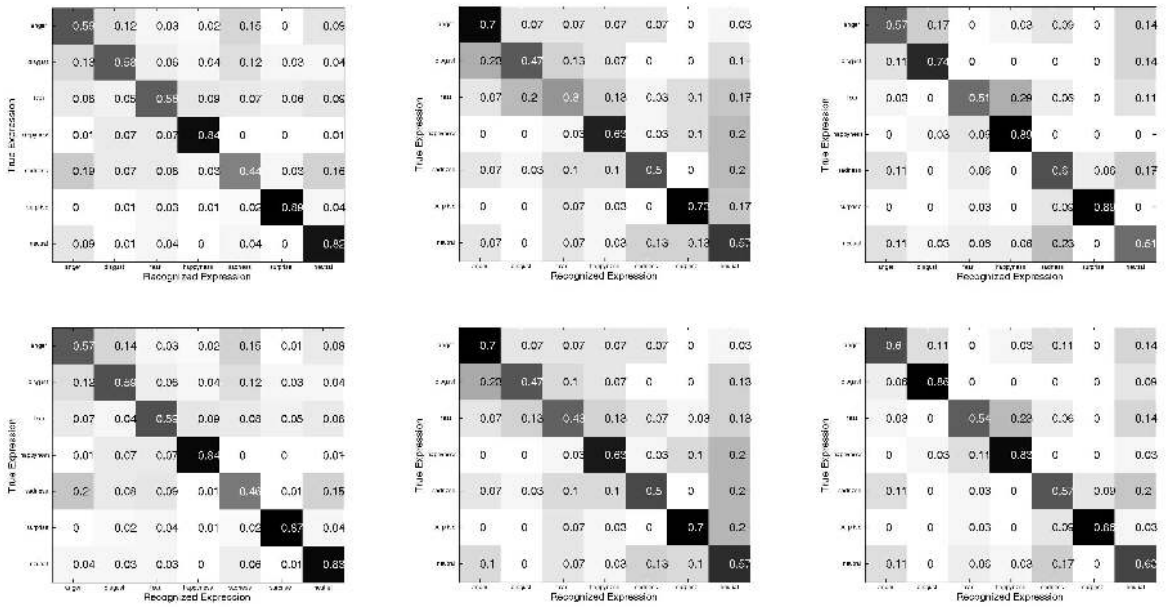
TABLE XII

TRAINING TIMES (SECONDS) OF ELM METHODS EXPLOITING KERNEL FORMULATIONS.

Dataset	KELM	**GEKELM (LE)**	**GEKELM (LLE)**	**GEKELM (LDA)**	**GEKELM (MDA)**	**GEKELM (LFDA)**
AR	2.645	3.618	3.693	4.722	4.579	4.433
ORL	0.028	0.087	0.095	0.136	0.167	0.165
Yalle	0.988	1.345	1.461	1.937	2.907	2.835
BU	0.101	0.225	0.253	0.655	0.499	0.483
JAFFE	0.014	0.043	0.045	0.059	0.075	0.071
KANADE	0.016	0.044	0.053	0.072	0.098	0.09
Olympic Sports	0.12	1.46	1.598	1.572	2.497	2.458
Hollywood2	0.2	0.225	1.271	0.914	1.275	1.324
Hollywood 3D	0.1	0.143	1.24	1.129	1.08	1.01

standard training/test splits provided by the databases and illustrated in Table XV. We illustrate the performance of the ELM methods exploiting random parameters and kernel formulations in Tables IX and X, respectively. Similarly to the facial image classification cases, it can be seen that the proposed GEELM methods outperform the ELM, RELM, KELM, MCVELM and DGRELM methods in most of the cases. Kernel ELM formulations achieve higher performance, when compared to the ones exploiting random parameters. This is in line with the state-of-the-art action recognition methods denoting that BoF-based action video representation should be combined with kernel classification schemes using the χ^2 kernel function. The AP values obtained for different actions when using the RELM, the KELM and the proposed approach are illustrated Tables XVI, XVII and XVIII for the Olympic Sports, Hollywood2 and Hollywood 3D datasets, respectively. As can be observed, the proposed formulations clearly outperform the RELM and KELM ones in most cases. In Tables XI and XII, we also provide the mean training times of each ELM variant on the action recognition datasets.

We also compare the performance obtained by adopting the Dense Trajectory-based video description combined with the



(a) (b)

Fig. 10. *Confusion matrices obtained by applying the KELM (top) and the proposed GEKELM (bottom) algorithms on the a) BU, b) Jaffe and c) Kanade facial expression recognition datasets.*

TABLE XIV

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

	BU	Jaffe	Kanade
Method [49]	-	56.72%	69.05%
Method [48]	66.4%	-	72.9%
Method [50] - MMP	-	-	70.3%
Method [50] - RP	-	-	**75.2**%
Method [50] - SVM	-	-	73.4%
Method [50] - MMRP	-	-	**80.1**%
Proposed method	**67.86**%	**57.14**%	**73.47**%

BoW model and the GEKELM classifier with that of some other state-of-the-art methods evaluating their performance on these datasets in Table XIX. As can be seen in this Table, the proposed approach provides satisfactory performance in all the cases.

TABLE XV

ACTION RECOGNITION DATASETS DETAILS.

Dataset	Training Samples	Test Samples	Classes (C)
Olympic Sports	649	134	16
Hollywood2	823	884	12
Hollywood 3D	643	308	13

Overall, the adoption of SL criteria under the Graph Embedding framework for ELM-based classification clearly enhances classification performance, when compared to the ELM, RELM and KELM methods. The exploitation of local class information seems to be a better choice for the proposed GEELM method, since GEELM formulations exploiting graph the structures used in LE, LLE, MDA and LFDA outperform the one exploiting the global class information used in LDA, as well as the MCVELM and DGRELM methods.

TABLE XVI

AVERAGE PRECISION VALUES ON THE OLYMPIC SPORTS DATASET.

	RELM	**GEELM**	KELM	**GEKELM**
Basket lay-up	95.87%	**96.33**%	96.69%	**97.76**%
Bowling	73.33%	**73.9**%	90.03%	**91.19**%
Clean & Jerk	80.6%	**81.11**%	88.19%	**89.51**%
Discus	90.58%	**91.28**%	92.33%	**93.31**%
Diving 3m	100%	100%	100%	100%
Diving 10m	98.18%	**98.85**%	100%	100%
Hammer	91.07%	**91.57**%	96.36%	**97.31**%
H. Jump	71.43%	**71.94**%	71.62%	**72.97**%
Javelin	100%	100%	100%	100%
L. Jump	88.31%	**88.74**%	88.31%	**89.45**%
P. Vault	82.8%	**83.34**%	84.7%	**85.77**%
Shot Put	54.09%	**54.49**%	85.45%	**86.69**%
Snatch	74.78%	**75.42**%	77.08%	**78.54**%
T. Jump	43.99%	**44.62**%	67.27%	**68.5**%
T. Serve	84.99%	**85.65**%	100%	100%
Vault	77.78%	**78.42**%	83.6%	**87.72**%
Mean	81.79%	**82.22**%	88.85%	**89.74**%

TABLE XVII

AVERAGE PRECISION VALUES ON THE HOLLYWOOD2 DATASET.

	RELM	**GEELM**	KELM	**GEKELM**
An. Phone	22.56%	**23.11**%	38.2%	**39.12**%
Dr. Car	86.7%	**86.81**%	90.54%	**91.85**%
Eat	62.79%	**63.85**%	66.43%	**67.94**%
Fight	77.41%	**77.58**%	79.83%	**80.39**%
G.O. Car	50.86%	**51.19**%	59.44%	**60.31**%
H. Shake	33.25%	**33.63**%	39.78%	**40.73**%
Hug	42.11%	**42.84**%	45.33%	**48.61**%
Kiss	56.14%	**56.18**%	63.2%	**63.42**%
Run	77.8%	**78.02**%	83.29%	**83.87**%
Sit D.	61.78%	**62.49**%	70.1%	**70.37**%
Sit up	18.85%	**19.27**%	24.95%	**27.72**%
Stand up	66.99%	**67.27**%	75.1%	**75.55**%
Mean	54.77%	**55.2**%	61.34%	**62.5**%

TABLE XVIII
AVERAGE PRECISION VALUES ON THE HOLLYWOOD 3D DATASET.

	RELM	**GEELM**	KELM	**GEKELM**
Dance	34.42%	**35.68**%	38.91%	**39.59**%
Drive	**51.93**%	47.89%	65.01%	**65.52**%
Eat	10.72%	**12.73**%	8.04%	**8.87**%
Hug	8.95%	**15.97**%	10.56%	**11.2**%
Kick	19.21%	**21.13**%	22.76%	**23.35**%
Kiss	36.88%	**38.61**%	43.23%	**43.95**%
Punch	11.53%	**13.39**%	11.73%	**12.29**%
Run	**9.25**%	9.02%	26.78%	**27.55**%
Shoot	49.84%	**50.55**%	48.78%	**49.34**%
Sit down	44.18%	**46.93**%	47.35%	**48.19**%
Stand up	8.16%	**9.84**%	9.85%	**10.46**%
Swim	45.09%	**49.29**%	57.58%	**58.62**%
Use phone	**42.93**%	42.12%	29.11%	**29.68**%
No action	14.63%	**15.26**%	16.22%	**16.48**%
Mean	27.68%	**31.14**%	31.14%	**31.79**%

TABLE XIX
COMPARISON OF OUR RESULTS WITH SOME STATE-OF-THE-ART METHODS ON THE OLYMPIC SPORTS, HOLLYWOOD2 AND HOLLYWOOD 3D DATASETS.

	Olympic Sports	Hollywood2	Hollywood 3D
Method [41]	-	-	15%
Method [51]	-	-	26.11%
Method [52]	77.33%	-	-
Method [53]	-	61.9%	-
Method [54]	82.7%	-	-
Method [55]	-	61%	-
Method [56]	80.6%	59.5%	-
Method [57]	83.2%	**62.5**%	-
Proposed method	**89.74**%	**62.5**%	**31.79**%

V. CONCLUSIONS

In this paper, we proposed a novel extension of the Extreme Learning Machine algorithm for SLFN network training that is able to incorporate SL criteria on the optimization process followed for the calculation of the network's output weights. The proposed Graph Embedded ELM (GEELM) algorithm is able to naturally exploit both intrinsic and penalty SL criteria that have been (or will be) designed under the Graph Embedding framework. The proposed GEELM algorithm has also been extended in order to exploit both intrinsic and penalty SL criteria in arbitrary (even infinite) dimensional ELM spaces. Extensive evaluation on nine publicly available datasets shows the effectiveness of the proposed approach.

REFERENCES

[1] G. Huang, Q. Zhu, and C. Siew, "Extreme Learning Machine: a new learning scheme of feedforward neural networks," *IEEE International Joint Conference on Neural Networks*, 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 Transactions on Neural Networks*, 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 Transactions on Information 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 Transactions on Neural Networks and Learning Systems*, 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 Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[7] G. Huang, D. Wang, and Y. Lan, "Extreme Learning Machine: a survey," *International Journal of Machine Learning and CYbernetics*, vol. 2, no. 2, pp. 107–122, 2011.

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

[9] ——, "An insight to Extreme Learning Machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, pp. 379–390, 2014.

[10] J. Cao, T. Chen, and J. Fan, "Fast Online Learning Algorithm for landmark recognition based on BoW framework," *IEEE Conference on Industrial Electronics and Applications*, 2014.

[11] A. Iosifidis, A. Tefas, and I. Pitas, "Minimum Class Variance Extreme Learning Machine for human action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 11, pp. 1968–1979, 2013.

[12] S. Yan, D. Xu, B. Zhang, and H. Zhang, "Graph Embedding and extensions: A general framework for dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, 2007.

[13] G. Arvanitidis and A. Tefas, "Exploiting graph embedding in support vector machines," *IEEE International Workshop on Machine Learning for Signal Processing*, 2012.

[14] Y. Peng, S. Wang, X. Long, and B. L. Lu, "Discriminative graph regularized Extreme Learning Machine for face recognition," *Neurocomputing*, vol. 149, pp. 340–353, 2015.

[15] A. Iosifidis, A. Tefas, and I. Pitas, "Minimum Variance Extreme Learning Machine for human action recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014.

[16] ——, "Regularized extreme learning machine for multi-view semi-supervised action recognition," *Neurocomputing*, vol. 145, pp. 250–262, 2014.

[17] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[18] L. Saul and S. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003.

[19] R. Duda, P. Hart, and D. Stork, "Pattern classification, 2nd ed." *Wiley-Interscience*, 2000.

[20] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local Fisher Discriminant Analysis," *Journal of Machine Learning Research*, vol. 8, pp. 1027–1061, 2007.

[21] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[22] D. Cai, X. He, and J. Han, "Spectral Regression: A unified approach for sparse subspace learning," *International Conference on Data Mining*, 2007.

[23] ——, "Spectral Regression for efficient regularized subspace learning," *International Conference on Computer Vision*, 2007.

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

[25] B. Frenay and M. Verleysen, "Using SVMs with randomised feature spaces: An extreme learning approach," *European Symposium of Artificial Neural Networks*, 2010.

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

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

[28] A. Argyriou, C. Micchelli, and M. Pontil, "When is there a representer theorem? vector versus matrix regularizers," *Journal of Machine Learning Research*, vol. 10, pp. 2507–2529, 2009.

[29] H. Wang and Schmid, "Action recognition with improved trajectories," *International Conference on Computer Vision*, 2013.

[30] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[31] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

[32] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," *IEEE Workshop on Applications of Computer Vision*, 1994.

[33] A. Martinez and A. Kak, "PCA versus LDA," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[34] K. Lee, J. Ho, and D. Kriegman, "Acquiriing linear subspaces for face recognition under variable lighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 684–698, 2005.

[35] T. Kanade, Y. Tian, and J. Cohn, "Comprehensive database for facial expression analysis," *IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

[36] L. Yin, X. Wei, Y. Sun, J. Wang, and M. Rosato, "A 3D facial expression database for facial behavior research," *IEEE International Conference on Automatic Face and Gesture Recognition*, 2006.

[37] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with Gabor wavelets," *IEEE International Conference on Automatic Face and Gesture Recognition*, 1998.

[38] M. Marszalek, I. Laptev, and C. Schmid, "Actions in context," *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[39] M. Zhu, "Recall, Precision and Average Precision," *Canada*, 2004.

[40] J. Niebles, C. Chend, and L. Fei-Fei, "Modeling temporal structure of decomposable mition segemnts for activity classification," *European Conference on Computer Vision*, 2010.

[41] S. Hadfield and R. Bowden, "Hollywood 3D: Recognizing actions in 3D natural scenes," *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[42] P. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. Prentice-Hall, 1982.

[43] J. Wright, A. Yang, A. Ganesh, S. S.S., and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 1–18, 2009.

[44] X. Liu, S. Yan, and J. H., "Projective Nonnegative Graph Embedding," *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1126–1137, 2010.

[45] A. James, "One-sample face recognition with local similarity decisions," *International Journal of Applied Pattern Recognition*, vol. 1, no. 1, pp. 61–80, 2013.

[46] Z. Sun, K. Lam, Z. Dong, H. Wang, Q. Gao, and C. Zheng, "Face recognition with multi-resolution spectral feature images," *PLOS ONE, DOI: 10.1371/journal.pone.0055700*, 2013.

[47] X. Tang, G. Feng, and J. Cai, "Weighted group sparse representation for undersampled face recognition," *Neurocomputing*, vol. 145, pp. 402–415, 2014.

[48] S. Nikitidis, A. Tefas, and I. Pitas, "Projected Gradients for Subclass Discriminant Nonnegative Subspace Learning," *IEEE Transactions on Cybernetics, in press*, 2014.

[49] ——, "Subclass Discriminant Nonnegative Matrix Factorization for facial image analysis," *Pattern Recognition*, vol. 45, pp. 4080–4091, 2012.

[50] ——, "Maximum Margin Projection Subspace Learning for visual data analysis," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4413–4425, 2014.

[51] K. Konda and R. Memisevic, "Unsupervised learning of depth and motion," *arXiv:1312.3429v2*, 2013.

[52] W. Brendel and S. Todorovic, "Learning spatiotemporal graphs of human activities," *International Conference on Computer Vision*, 2011.

[53] E. Vig, M. Dorr, and D. Cox, "Space-variant descriptor sampling for action recognition based on saliency and eye movements," *European Conference on Computer Vision*, 2012.

[54] A. Gaidon, Z. Harchaoui, and C. Schmid, "Recognizing activities with cluster-tries of tracklets," *British Machine Vision Conference*, 2012.

[55] S. Mathe and C. Sminchisescu, "Dynamic eye movement datasets and learnt saliency models for visual action recognition," *European Conference on Computer Vision*, 2012.

[56] Y. Jiang, Q. Dai, X. Xue, W. Liu, and C. Ngo, "Trajectory-based modeling of human actions with motion reference points," *European Conference on Computer Vision*, 2012.

[57] M. Jain, H. Jegou, and P. Bouthemy, "Better exploiting motion for better action recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

Alexandros Iosifidis (M'14) received a Diploma in Electrical & Computer Engineering in 2008 and a Master of Engineering in the area of Mechatronics in 2010 from the Democritus University of Thrace, Greece. He received a Ph.D. in Informatics in 2014 from the Aristotle University of Thessaloniki, Greece. From January 2015, he has joined the Multimedia Group of the Department of Signal Processing in Tampere University of Technology as a postdoctoral researcher. In 2014 he was a postdoctoral researcher at the Artificial Intelligence and Information Analysis laboratory of the Department of Informatics in Aristotle University of Thessaloniki. Dr. Iosifidis has participated in 5 research projects financed by national and European funds. He has co-authored 18 journal papers and 31 papers in international conferences. His research interests include image/video processing, computer vision and pattern recognition.

Anastasios Tefas (S'97-M'04) received the B.Sc. in informatics in 1997 and the Ph.D. degree in informatics in 2002, both from the Aristotle University of Thessaloniki, Greece. Since 2013 he has been an Assistant Professor at the Department of Informatics, Aristotle University of Thessaloniki. From 2008 to 2012, he was a Lecturer at the same University. From 2006 to 2008, he was an Assistant Professor at the Department of Information Management, Technological Institute of Kavala. From 2003 to 2004, he was a temporary lecturer in the Department of Informatics, University of Thessaloniki. From 1997 to 2002, he was a researcher and teaching assistant in the Department of Informatics, University of Thessaloniki. Dr. Tefas participated in 12 research projects financed by national and European funds. He has co-authored 52 journal papers, 135 papers in international conferences and contributed 7 chapters to edited books in his area of expertise. Over 2450 citations have been recorded to his publications and his H-index is 25 according to Google scholar. His current research interests include computational intelligence, pattern recognition, statistical machine learning, digital signal and image processing and computer vision.

Ioannis Pitas (SM'94-F'07) (IEEE fellow, IEEE Distinguished Lecturer, EURASIP fellow) received the Diploma and PhD degree in Electrical Engineering, both from the Aristotle University of Thessaloniki, Greece. Since 1994, he has been a Professor at the Department of Informatics of the same University. He served as a Visiting Professor at several Universities.

His current interests are in the areas of image/video processing, intelligent digital media, machine learning, human centered interfaces, affective computing, computer vision, 3D imaging and biomedical imaging. He has published over 790 papers, contributed in 39 books in his areas of interest and edited or (co-)authored another 10 books. He has also been member of the program committee of many scientific conferences and workshops. In the past he served as Associate Editor or co-Editor of eight international journals and General or Technical Chair of four international conferences. He participated in 68 R&D projects, primarily funded by the European Union and is/was principal investigator/researcher in 40 such projects. He has 19800+ citations (Source Publish and Perish), 7317+ (Scopus) to his work and h-index 68+ (Source Publish and Perish), 44+ (Scopus).