Pitas, I., Iosifidis, A., & Tefas, A. (2015). DropELM: Fast Neural Network Regularization with Dropout and DropConnect. Neurocomputing, 162, 57-66. DOI: 10.1016/j.neucom.2015.04.006

DropELM: Fast Neural Network Regularization with Dropout and DropConnect

Alexandros Iosifidis, Anastasios Tefas and Ioannis Pitas

Department of Informatics, Aristotle University of Thessaloniki
Thessaloniki 54124, Greece Tel,Fax: +30-2310996304

{*aiosif,tefas,pitas*}*@aiia.csd.auth.gr*

Abstract

In this paper, we propose an extension of the Extreme Learning Machine algorithm for Single-hidden Layer Feedforward Neural network training that incorporates Dropout and DropConnect regularization in its optimization process. We show that both types of regularization lead to the same solution for the network output weights calculation, which is adopted by the proposed DropELM network. The proposed algorithm is able to exploit Dropout and DropConnect regularization, without computationally intensive iterative weight tuning. We show that the adoption of such a regularization approach can lead to better solutions for the network output weights. We incorporate the proposed regularization approach in several recently proposed ELM algorithms and show that their performance can be enhanced without requiring much additional computational cost.

Keywords: Single Hidden Layer Feedforward Networks; Extreme Learning Machine; Regularization; Dropout; DropConnect

1. 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 Backpropagation algorithm [2], adjust both the input and output weights and the hidden layer bias values 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 chosen. By adopting the squared loss of the prediction error, the network output weights are, subsequently, analytically calculated. ELMs tend to reach not only 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 [3]. 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 global approximator properties [4, 5, 6, 7]. Due to its effectiveness and its fast learning process, the ELM network has been adopted in many classification problems [8, 9, 10, 11, 12, 13, 14, 15]. In addition, many ELM variants have been proposed in the last few years, extending the ELM network properties along different directions [5, 16, 17, 18, 19, 20, 21, 22, 23].

In order to enhance the generalization ability of networks trained by applying the Backpropagation algorithm and increase training speed, several heuristics are usually employed, including a decaying learning rate parameter (or dynamic learning rate parameter determination), the use of momentum parameter, minibatch-based weight adaptation using stochastic gradient descend, multiple network weights initializations, use of generative pre-training using a small learning rate, penalization of the weights l_2 norm, etc. Recently, the so-called *Dropout* approach has been proposed, in order to avoid network overfitting on the training data [24]. According to this approach, each training example is forward propagated, while randomly keeping the outputs of each layer with probability p, otherwise setting each layer outputs to zero with probability $(1 - p)$. The error is then backpropagated only through the weights corresponding to the activated layer outputs. The intuition in this process is that such a process prevents the network weights from collaborating with one another to memorize the training examples. *DropConnect* [25] has been proposed as a generalization of Dropout. In DropConnect, each training sample is forward propagated, while randomly keeping some of the network weight elements. Both these approaches can be regarded as a form of regularization on the training process of fully connected neural networks. Extensive experimental evaluation combining Dropout (or DropConnect) with the previously-mentioned heuristics [24, 25] indicates that the adoption of such a regularization approach enhances network generalization performance, when compared to networks trained by using Backpropagation without exploiting such operations.

In this paper we propose an extension of the ELM algorithm that is able to exploit Dropout and DropConnect regularization on its optimization process. We start by formulating an ELM optimization problem that is able to exploit Dropout regularization. This is performed by learning network output weights that provide a compromise between the minimization of both the network training error and the network response difference for the original hidden layer outputs and hidden layer outputs obtained by applying Dropout operations with a probability value p. Subsequently, we exploit DropConnect regularization on the network output weights and formulate an ELM optimization problem that is able to learn network output weights providing a compromise between the minimization of both the network training error and the network response difference between the network outputs determined by using the original output weights and output weights obtained by using DropConnect operations with a probability value p. We show that, in the limit case of infinite network training epoches, both proposed optimization problems lead to the same regularized ELM network, noted as DropELM hereafter, which is able to incorporate the Dropout and DropConnect regularization, without requiring time-consuming iterative network training. We evaluate the proposed approach in standard, as well as in facial image classification problems, where it is shown that the adoption of the proposed regularization term can enhance the performance of several recently proposed ELM variants.

The contributions of this paper are:

- a novel formulation of the ELM algorithm that is able to exploit Dropout regularization on the network hidden layer outputs,
- a novel formulation of the ELM algorithm that is able to exploit DropConnect regularization on the network output weights,
- a proof that, in the limiting case of infinite network training epoches, both proposed ELM formulations lead to the same regularized ELM network, i.e., the proposed DropELM network.
- Finally, we show that the adoption of the proposed regularization approach can lead to better solutions for the network output weights.

The rest of the paper is organized as follows. We provide an overview of the ELM, RELM, Dropout and DropConnect algorithms in Section 2. The proposed DropELM network is described in Subsection 3. Experimental results evaluating its performance are provided in Section 4. Finally, conclusions are drawn in Section 5.

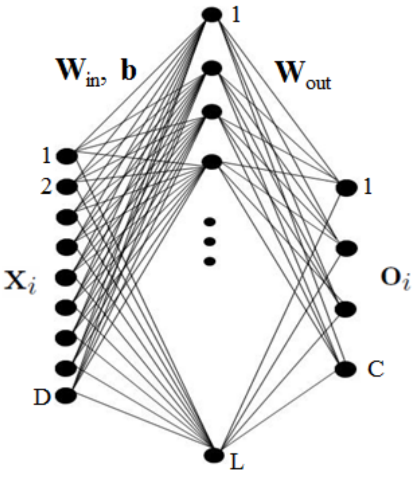

Figure 1: *SLFN network topology.*

2. ELM networks and Dropout/DropConnect approaches

In this section, we briefly describe the ELM, RELM, Dropout and DropConnect algorithms proposed in [1], [21, 26, 27, 28, 29], [24] and [25], respectively.

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 $c_i \in \{1, \ldots, C\}$ that can be used to train a SLFN network consisting 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, as illustrated in Figure 1. The number L of hidden layer neurons is usually selected to be much larger than the number of classes [21, 22], 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 chosen, while the network output weights $\mathbf{W}_{out} \in \mathbb{R}^{L \times C}$ are analytically calculated, as subsequently described.

Let us denote by $\mathbf{v}_j$, $\mathbf{w}_k$, 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:

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

It has been shown [21, 22] 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 and Radial Basis Function (RBF). By storing the network hidden layer outputs $\boldsymbol{\phi}_i \in \mathbb{R}^L$ corresponding to all the training vectors $\mathbf{x}_i,\ i = 1, \dots, N$ in a matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_N]$, or:

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

the network response for all the training data $\mathbf{O} \in \mathbb{R}^{C \times N}$ can be expressed in a matrix form as:

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

2.1. Extreme Learning Machine

ELM algorithm [1] assumes zero training error. That is, it is assumed that $\mathbf{o}_i = \mathbf{t}_i,\ i = 1, \dots, N$, or by using a matrix notation $\mathbf{O} = \mathbf{T}$, where $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_N]$ is a matrix containing the network target vectors. By using (3), the network output weights $\mathbf{W}_{out}$ can be analytically calculated by:

$$\mathbf{W}_{out} = \boldsymbol{\Phi}^{\dagger}\ \mathbf{T}^T, \tag{4}$$

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:

$$\mathbf{o}_l = \mathbf{W}_{out}^T \boldsymbol{\phi}_l, \tag{5}$$

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

2.2. Regularized Extreme Learning Machine

The calculation of the network output weights $\mathbf{W}_{out}$ through (4) 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]. In this case, the network output weights are calculated by solving the following optimization problem:

$$\textbf{Minimize:} \quad \mathcal{J}_{RELM} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \tag{6}$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T\boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i, \quad i = 1, ..., N, \tag{7}$$

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 $c > 0$. By substituting the constraints (10) in (9) and determining the saddle point of $\mathcal{J}_{RELM}$ with respect to $\mathbf{W}_{out}$, the network output weights are obtained by:

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

where $\mathbf{I} \in \mathbb{R}^{L\times L}$ is the identity matrix.

The RELM algorithm described above has also been extended in order to exploit statistical properties of the training data in the ELM space, which are described through appropriate regularization terms [26, 27, 28, 29]. These methods optimize the following optimization problem for the calculation of the network output weights:

$$\textbf{Minimize:} \quad \mathcal{J}_{DELM} = \frac{1}{2}tr\left(\mathbf{W}_{out}^T\mathbf{S}\mathbf{W}_{out}\right) + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \tag{9}$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T\boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i, \quad i = 1, ..., N, \tag{10}$$

where $tr(\cdot)$ denotes the trace operator and $\mathbf{S} \in \mathbb{R}^{L\times L}$ is a matrix describing relationships of the training data that are subject to minimization[1]. Specifically, in [26] the matrix S describes the within-class scatter of the training data representations in the ELM space, while in [27, 28], S describes the local within-class scatter of the training data, by exploiting intrinsic graph structures. In [29] both intrinsic and penalty graphs are exploited. The network output weights are finally obtained by:

$$\mathbf{W}_{out} = \left(\mathbf{\Phi}\mathbf{\Phi}^T + \frac{1}{c}\mathbf{S}\right)^{-1}\mathbf{\Phi}\mathbf{T}^T. \tag{11}$$

[1] It should be noted here that a regularized version, i.e. $\tilde{\mathbf{S}} = r_1\mathbf{S} + r_2\mathbf{I}$, is usually used in these methods in order to avoid singularity issues.

Clearly, these methods are extensions of the ELM and RELM algorithms, since (4) and (8) is a special cases of (11) for $\mathbf{S} = \mathbf{0}$ and $\mathbf{S} = \mathbf{I}$, respectively.

Similar to the ELM case, 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:

$$\mathbf{o}_l = \mathbf{W}_{out}^T \boldsymbol{\phi}_l. \tag{12}$$

2.3. Iterative ELM network training based on Dropout

Dropout has been proposed in [24] as a form of regularization for fully connected neural network layers. Let us consider the case of a SLFN network where Dropout is applied to the network hidden layer outputs in an iterative process that is used in order to learn the network output weights. The main idea of Dropout is the generation of *synthetic hidden layer outputs* $\boldsymbol{\phi}_{i,t}$:

$$\boldsymbol{\phi}_{i,t} = \mathbf{m}_{i,t} \circ \boldsymbol{\phi}_i, \tag{13}$$

where $\circ$ denotes the Hadamard (element-wise) product of two vectors [30, 31] and $\mathbf{m}_{i,t} \in \mathbb{R}^L$ is a binary mask vector with each element being equal to 1 with probability p and equal to 0 with probability $(1-p)$. The elements of $\mathbf{m}_{i,t}$ are drawn independently from a Bernoulli distribution using a probability value p. In the above, we have introduced another index denoting the epoch of the iterative learning process, i.e., $t = 1, \dots, N_T$, where N_T is the maximal number of training epoches. After calculating the network output $\mathbf{o}_{i,t}$ corresponding to $\boldsymbol{\phi}_{i,t}$, the network output weights $\mathbf{W}_{out}$ corresponding to the elements of $\boldsymbol{\phi}_i$ that have *survived* are updated in order to follow the network error gradient. It should be noted here that a different mask vector $\mathbf{m}_{i,t}$ is independently selected for each iteration of the iterative optimization process.

2.4. Iterative ELM network training based on DropConnect

DropConnect has been proposed in [25] as a generalization of Dropout, in which each connection, rather than each output unit, can be dropped with probability $1-p$. Let us consider the case of a SLFN network where DropConnect is applied in order to learn the network output weights $\mathbf{W}_{out}$. DropConnect is similar to Dropout as it introduces dynamic sparsity in the network training process, but differs in that the sparsity is on the weights $\mathbf{W}_{out}$, rather than the hidden layer output vectors $\boldsymbol{\phi}_i$. The main idea of DropConnect is the generation of *synthetic network weights* $\mathbf{W}_{out}^{i,t}$ given by:

$$\mathbf{W}_{out}^{i,t} = \mathbf{M}_{i,t} \circ \mathbf{W}_{out}, \quad i = 1, \dots, N, \quad t = 1, \dots, N_T. \tag{14}$$

In the above, $\mathbf{M}_{i,t} \in \mathbb{R}^{L \times C}$ is a binary mask matrix with each element being equal to $[\mathbf{M}_{i,t}]_{j,k} = 1$ with probability p and $[\mathbf{M}_{i,t}]_{j,k} = 0$ with probability $(1 - p)$. The elements of $\mathbf{M}_{i,t}$ are drawn independently from a Bernoulli distribution, using a probability value p. After calculating the network output $\mathbf{o}_{i,t}$ by using $\boldsymbol{\phi}_i$ and $\mathbf{W}_{out}^{i,t}$, the elements of the network output weights $\mathbf{W}_{out}$ that have *survived* are updated, in order to follow the network error gradient. Similar to the Dropout case, the binary mask matrix is selected independently for each iteration of the iterative optimization process.

3. The proposed DropELM algorithm

As has been previously mentioned, both Dropout and DropConnect-based learning schemes are able to increase the network generalization ability. However, both of them are time consuming and require high human supervision, as they are based on the Backpropagation algorithm. In the following, we propose an optimization process that is able to overcome these issues. We start by formulating an ELM optimization problem that is able to exploit Dropout regularization.

3.1. Exploiting Dropout in ELM

In order to exploit Dropout on the ELM optimization process, we propose to solve the following optimization problem for the calculation of the network output weights $\mathbf{W}_{out}$:

$$\textbf{Minimize:} \quad \mathcal{J}_1 = \frac{1}{2} tr\left(\mathbf{W}_{out}^T \mathbf{S} \mathbf{W}_{out}\right) + \frac{c}{2} \sum_{i=1}^{N} \|\boldsymbol{\xi}_i\|_2^2 \tag{15}$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T \boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i, \quad i = 1, ..., N, \tag{16}$$

$$\mathbf{W}_{out}^T \boldsymbol{\phi}_i = \mathbf{W}_{out}^T \boldsymbol{\phi}_{i,t}, \quad i = 1, ..., N, \quad t = 1, \ldots, N_T, \tag{17}$$

where $\boldsymbol{\phi}_{i,t}$ expresses the hidden layer output for the training vector $\mathbf{x}_i$, after applying the Dropout for the training epoch t:

$$\boldsymbol{\phi}_{i,t} = \mathbf{m}_{i,t} \circ \boldsymbol{\phi}_i. \tag{18}$$

That is, we introduce an additional constraint (17) requiring the network outputs for $\boldsymbol{\phi}_{i,t}$ to be equal to the ones obtained by employing the original hidden layer outputs $\boldsymbol{\phi}_i$. The importance of the constraint (17) in ELM optimization process is discussed in the Appendix. By setting $\tilde{\boldsymbol{\phi}}_{i,t} = \boldsymbol{\phi}_i - \boldsymbol{\phi}_{i,t}$, this constraint can be expressed by:

$$\mathbf{W}_{out}^T \left(\boldsymbol{\phi}_i - \boldsymbol{\phi}_{i,t}\right) = \mathbf{W}_{out}^T \tilde{\boldsymbol{\phi}}_{i,t} = \mathbf{0}. \tag{19}$$

We can directly calculate $\tilde{\boldsymbol{\phi}}_{i,t}$ by:

$$\tilde{\boldsymbol{\phi}}_{i,t} = \tilde{\mathbf{m}}_{i,t} \circ \boldsymbol{\phi}_i, \tag{20}$$

where $\tilde{\mathbf{m}}_{i,t} \in \mathbb{R}^L$ is a binary mask vector with each element being equal to 1 with probability $(1-p)$ and equal to 0 with probability p.

By substituting (16) in $\mathcal{J}_1$ and taking the equivalent dual problem with respect to (19), we obtain:

$$\mathcal{J}_{1,D} = \frac{1}{2} tr\left(\mathbf{W}_{out}^T \mathbf{S} \mathbf{W}_{out}\right) + \frac{c}{2} \|\mathbf{W}_{out}^T \boldsymbol{\Phi} - \mathbf{T}\|_F^2 + \frac{\lambda}{2N_T} \sum_{t=1}^{N_T} \|\mathbf{W}_{out}^T \tilde{\boldsymbol{\Phi}}_t\|_F^2, \tag{21}$$

where $\tilde{\boldsymbol{\Phi}}_t = [\tilde{\boldsymbol{\phi}}_{1,t}, \ldots, \tilde{\boldsymbol{\phi}}_{N,t}]$ and λ is a parameter denoting the importance of the Dropout regularizer in the optimization problem, satisfying $\lambda > 0$. By determining the min point of $\mathcal{J}_{1,D}$ with respect to $\mathbf{W}_{out}$, the network output weights are given by:

$$\mathbf{W}_{out} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \frac{1}{c}\mathbf{S} + \frac{\lambda}{c}\mathbf{R}_1\right)^{-1} \boldsymbol{\Phi}\mathbf{T}^T, \tag{22}$$

where:

$$\mathbf{R}_1 = \frac{1}{N_T} \sum_{t=1}^{N_T} \tilde{\boldsymbol{\Phi}}_t \tilde{\boldsymbol{\Phi}}_t^T. \tag{23}$$

By comparing (22) and (8), it can be seen that the contribution of Dropout on the ELM optimization process can be expressed by the regularization term $\mathbf{R}_1$. Clearly, the RELM algorithm is a special case of the proposed approach for $p = 1$.

We can define the following two cases for the matrix $\mathbf{R}_1$:

1. N_T is finite. $\mathbf{R}_1$ can be calculated by generating synthetic hidden layer output vectors $\tilde{\boldsymbol{\phi}}_{i,t},\ i = 1, \ldots, N,\ t = 1, \ldots, N_T$ through (20) and by using (23). While this choice can be used in order to incorporate Dropout on the ELM optimization process, it requires the calculation of $\tilde{\boldsymbol{\Phi}}_t$ and $\mathbf{R}_1$, which will increase the computational cost of the network training process. In addition, we have experimentally observed that, for values of N_T greater than 100 (which can be considered as a small value for neural network training epoches), it degenerates to the following case.
2. $N_T \to \infty$: Based on the weak law of large numbers [32], $\mathbf{R}_1$ converges to its expected value, as N_T becomes very large:

$$\mathbf{R}_1 = E\left[\frac{1}{N_T} \sum_{t=1}^{N_T} \tilde{\boldsymbol{\Phi}}_t \tilde{\boldsymbol{\Phi}}_t^T\right]. \tag{24}$$

Let us denote by $\mathbf{p} = [(1-p), \ldots, (1-p)]^T \in \mathbb{R}^L$ a vector having elements expressing the probability that the j-th element of $\boldsymbol{\phi}_i$ will not survive. Then, $\mathbf{R}_1$ can be obtained by:

$$\mathbf{R}_1 = \left(\mathbf{\Phi}\mathbf{\Phi}^T\right) \circ \mathbf{P}, \tag{25}$$

where $\mathbf{P} = \left[(\mathbf{p}\mathbf{p}^T) \circ (\mathbf{1}\mathbf{1}^T - \mathbf{I})\right] + \left[(\mathbf{p}\mathbf{1}^T) \circ \mathbf{I}\right]$ and $\mathbf{1} \in \mathbb{R}^L$ is a vector of ones. It is worth noting here that, since $\mathbf{R}_1 \in \mathbb{R}^{L \times L}$, the overall computational complexity for $\mathbf{W}_{out}$ calculation is equal to that of RELM (8), i.e., $O(L^3)$, which is the computational cost of the matrix inversion operation[2]. By combining (25) and (22), the network output weights are given by:

$$\mathbf{W}_{out} = \left(\left[\mathbf{\Phi}\mathbf{\Phi}^T\right] \circ \left[\mathbf{1}\mathbf{1}^T + \frac{\lambda}{c}\mathbf{P}\right] + \frac{1}{c}\mathbf{S} \right)^{-1} \mathbf{\Phi}\mathbf{T}^T. \tag{26}$$

3.2. Exploiting DropConnect in ELM

In order to exploit DropConnect on the ELM optimization process, we propose to solve the following optimization problem for the calculation of the network output weights $\mathbf{W}_{out}$:

$$\textbf{Minimize:} \quad \mathcal{J}_2 = \frac{1}{2} tr\left(\mathbf{W}_{out}^T \mathbf{S} \mathbf{W}_{out}\right) + \frac{c}{2} \sum_{i=1}^{N} \|\boldsymbol{\xi}_i\|_2^2 \tag{27}$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T \boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i, \quad i = 1, ..., N, \tag{28}$$

$$\mathbf{W}_{out}^T \boldsymbol{\phi}_i = \left(\mathbf{M}_{i,t} \circ \mathbf{W}_{out}\right)^T \boldsymbol{\phi}_i, \quad i = 1, ..., N, \quad t = 1, \ldots, N_T, \tag{29}$$

where $\mathbf{M}_{i,t} \in \mathbb{R}^{L \times C}$ is a binary mask matrix with each element being equal to $[\mathbf{M}_{i,t}]_{j,k} = 1$ with probability p and $[\mathbf{M}_{i,t}]_{j,k} = 0$ with probability $(1-p)$. That is, we introduce an additional constraint (29) requiring the network outputs obtained by using the weights $\mathbf{W}_{out}^{i,t} = \mathbf{M}_{i,t} \circ \mathbf{W}_{out}$ to be equal to the ones obtained by employing the original network output weights $\mathbf{W}_{out}$.

By substituting (28) in $\mathcal{J}_2$ and taking the equivalent dual problem with respect to (29), we obtain:

$$\mathcal{J}_{2,D} = \frac{1}{2} tr\left(\mathbf{W}_{out}^T \mathbf{S} \mathbf{W}_{out}\right) + \frac{c}{2} \|\mathbf{W}_{out}^T \mathbf{\Phi} - \mathbf{T}\|_F^2 + \frac{\lambda}{2N_T} \sum_{t=1}^{N_T} \sum_{i=1}^{N} \| \left(\mathbf{M}_{i,t} \circ \mathbf{W}_{out}\right)^T \boldsymbol{\phi}_i \|_2^2. \tag{30}$$

[2] This is the computational complexity of both RELM and DropELM when employing the Gauss-Jordan elimination algorithm. Faster matrix inversion algorithms have also been proposed having computational complexity equal to $O(L^{2.37})$, i.e., the Coppersmith-Winograd and Williams algorithms.

By exploiting that, when $N_T \to \infty$[3]:

$$\frac{1}{N_T}\sum_{t=1}^{N_T}\left(\mathbf{M}_{i,t}\circ\mathbf{W}_{out}\right)\left(\mathbf{M}_{i,t}\circ\mathbf{W}_{out}\right)^T = \left(\mathbf{W}_{out}\mathbf{W}_{out}^T\right)\circ\mathbf{P}, \tag{31}$$

we obtain:

$$\begin{aligned}\mathcal{J}_{2,D} &= \frac{1}{2}tr\left(\mathbf{W}_{out}^T\mathbf{S}\mathbf{W}_{out}\right) + \frac{c}{2}\|\mathbf{W}_{out}^T\boldsymbol{\Phi} - \mathbf{T}\|_F^2 + \frac{\lambda}{2N_T}\sum_{t=1}^{N_T}\sum_{i=1}^{N}\boldsymbol{\phi}_i^T\left[\left(\mathbf{W}_{out}\mathbf{W}_{out}^T\right)\circ\mathbf{P}\right]\boldsymbol{\phi}_i \\ &= \frac{1}{2}tr\left(\mathbf{W}_{out}^T\mathbf{S}\mathbf{W}_{out}\right) + \frac{c}{2}\|\mathbf{W}_{out}^T\boldsymbol{\Phi} - \mathbf{T}\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^{N}tr\left(\mathbf{D}_{\boldsymbol{\phi}_i}\mathbf{P}\mathbf{D}_{\boldsymbol{\phi}_i}\mathbf{W}_{out}\mathbf{W}_{out}^T\right), \quad (32)\end{aligned}$$

where $\mathbf{D}_{\boldsymbol{\phi}_i} = diag(\boldsymbol{\phi}_i)$.

By determining the saddle point of $\mathcal{J}_{2,D}$ with respect to $\mathbf{W}_{out}$, the network output weights are given by:

$$\mathbf{W}_{out} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \frac{1}{c}\mathbf{S} + \frac{\lambda}{c}\mathbf{R}_2\right)^{-1}\boldsymbol{\Phi}\mathbf{T}^T, \tag{33}$$

where:

$$\mathbf{R}_2 = \sum_{i=1}^{N} tr\left(\mathbf{D}_{\boldsymbol{\phi}_i}\mathbf{P}\mathbf{D}_{\boldsymbol{\phi}_i}\mathbf{W}_{out}\right) = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T\right)\circ\mathbf{P}. \tag{34}$$

By combining (34) and (33), the network output weights are given by:

$$\mathbf{W}_{out} = \left(\left[\boldsymbol{\Phi}\boldsymbol{\Phi}^T\right]\circ\left[\mathbf{1}\mathbf{1}^T + \frac{\lambda}{c}\mathbf{P}\right] + \frac{1}{c}\mathbf{S}\right)^{-1}\boldsymbol{\Phi}\mathbf{T}^T. \tag{35}$$

By comparing (35) and (8), it can be seen that the contribution of DropConnect on the ELM optimization process can be expressed by the regularization term $\mathbf{R}_2$. Clearly, RELM can be considered as a special case of (33), in the case where $\lambda = 0$. In addition, by comparing (35) and (26), it can be seen that, in the limit case $N_T \to \infty$, the two proposed ELM networks are equivalent. In the following subsection, we summarize the proposed DropELM algorithm for SLFN network training.

[3] We have experimentally observed that this is satisfied for N_T greater than 100.

3.3. Drop-Extreme Learning Machine

The proposed DropELM algorithm assigns random network input weights $\mathbf{W}_{in} \in \mathbb{R}^{D \times L}$ and the hidden layer bias values $\mathbf{b} \in \mathbb{R}^{L}$, similar to the ELM and RELM algorithms described in Section 2. The network output weights $\mathbf{W}_{out} \in \mathbb{R}^{L \times C}$ are subsequently analytically calculated by:

$$\mathbf{W}_{out} = \left(\left[\mathbf{\Phi}\mathbf{\Phi}^T \right] \circ \left[\mathbf{1}\mathbf{1}^T + \frac{\lambda}{c}\mathbf{P} \right] + \frac{1}{c}\mathbf{S} \right)^{-1} \mathbf{\Phi}\mathbf{T}^T. \tag{36}$$

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:

$$\mathbf{o}_l = \mathbf{W}_{out}^T \boldsymbol{\phi}_l, \tag{37}$$

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

4. Experiments

In this section, we present experiments conducted in order to illustrate the effect of exploiting the proposed regularization term in ELM-based algorithms. We have employed sixteen publicly available datasets to this end: ten datasets corresponding to standard classification problems coming from the machine learning repository of University of California Irvine (UCI) [33] and six facial image datasets, namely the ORL, AR and Extended YALE-B (face recognition) and the COHN-KANADE, BU and JAFFE (facial expression recognition) datasets. Table 1 provides information concerning the UCI data sets used. A brief description of the facial image datasets is provided in the following subsections. Experimental results are provided in subsection 4.3. In all our experiments, we compare the performance of the ELM variants proposed in [1, 21, 26, 27, 28, 29] with that of their DropELM counterparts. In facial image classification, we have downsized the facial images provided by the database to 40×30 pixels by subsampling to increase computation speed and vectorized the resulting images to produce 1200-dimensional facial vectors $\mathbf{x}_i$.

We follow the works in [34, 35] and select the hidden layer weights to be formed by the training data in order to enhance the performance of all the ELM variants. In the case where the cardinality of the training data is larger than $N > 1000$, we randomly select $L = 1000$ training samples to form the network hidden

Table 1: UCI data sets.

Data set	Samples	Dimensions	Classes
Abalone	4177	8	3
Australian	690	14	2
German	1000	24	2
Heart	270	13	2
Ionosphere	351	34	2
Iris	150	4	3
Madelon	2600	50	2
Optdigits	5620	64	10
Segmentation	2310	19	7
TAE	151	5	3

layer weights [34]. For the activation of the hidden layer neurons, we used the Radial Basis Function (RBF):

$$\Phi_{RBF}(\mathbf{x}_i, \mathbf{v}_j, b) = exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{v}_j\|_2^2}{2b^2} \Big), \tag{38}$$

where the value b is set equal to the mean Euclidean distance between the training data $\mathbf{x}_i$ and the network input weights $\mathbf{v}_j$, which is the natural scaling factor for the Euclidean distances between $\mathbf{x}_i$ and $\mathbf{v}_j$ for each dataset. For fair comparison, in all experiments, we made sure that the the same ELM space was used by all the competing algorithms. That is, we first map the training data to the feature space determined by the network hidden layer outputs, by using (38) and, subsequently, calculate the network output weights according to each optimization process. Regarding the optimal values of the regularization parameters c and λ used in different ELM-based classification schemes, they have been determined by following a grid search strategy. That is, for each classifier, multiple experiments have been performed by employing parameter values $c = 10^r$, $r = -3, \dots, 3$ and $\lambda = 10^q$, $q = -3, \dots, 3$ and the best performance is reported.

4.1. Face recognition datasets

The AR dataset [36] 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). Each person was recorded in two sessions, separated by an interval of two weeks. Example images of the dataset are illustrated in Figure 2.

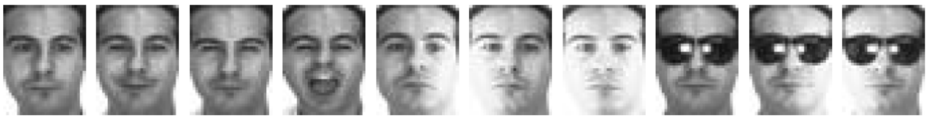

Figure 2: *Facial images depicting a person of the AR dataset.*

The ORL dataset [37] consists of 400 facial images depicting 40 persons (10 images each). 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 of up to 20 degrees. Example images of the dataset are illustrated in Figure 3.

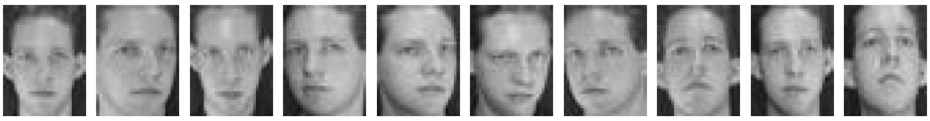

Figure 3: *Facial images depicting a person of the ORL dataset.*

The Extended YALE-B dataset [38] 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 4.

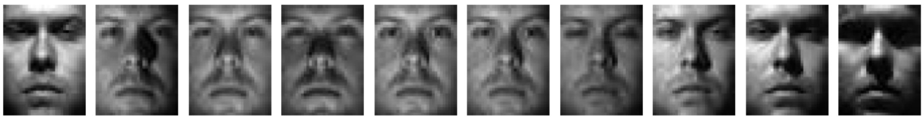

Figure 4: *Facial images depicting a person of the Extended YALE-B dataset.*

4.2. Facial expression recognition datasets

The BU dataset [39] consists of facial images depicting over 100 persons (60% feamale and 40% male) with a variety of ethnic/racial background, including White, Black, East-Asian, Middle-east Asian, Hispanic Latino and other types of persons. All expressions, except the neutral one, are expressed at four intensity levels. 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.

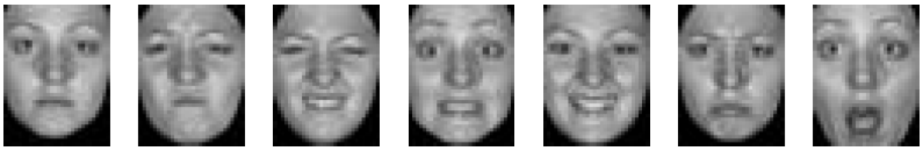

Figure 5: *Facial images depicting a person of the BU dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

The COHN-KANADE dataset [40] consists of facial images depicting 210 persons of age between 18 and 50 (69% female, 31% male, 81% Euro-American, 13% Afro-American and 6% other groups). We have randomly selected 35 images for each facial expression, i.e., anger, disgust, fear, happyness, sadness, surprise and neutral ones. Example images of the dataset are illustrated in Figure 6.

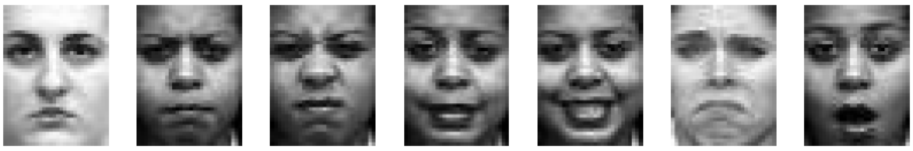

Figure 6: *Facial images from the COHN-KANADE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

The JAFFE dataset [41] consists of 210 facial images depicting 10 Japanese female persons. Each expression is depicted in 3 images for each person. Example images of the dataset are illustrated in Figure 7.

4.3. Experimental Results

In our first set of experiments, we have applied the 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

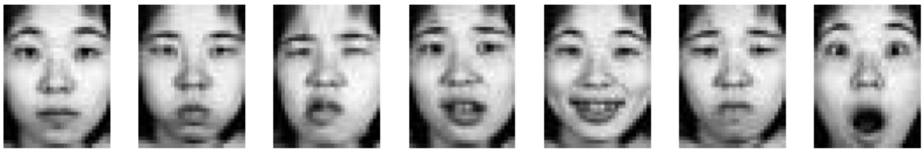

Figure 7: *Facial images depicting a person of the JAFFE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

Table 2: Performance on UCI datasets for ELM and RELM algorithms.

Dataset	ELM	DropELM	RELM	DropRELM
Abalone	$54.66 \pm 0.27\%$	$\mathbf{55.38 \pm 0.22\%}$ (0.2)	$54.27 \pm 0.23\%$	$\mathbf{55.04 \pm 0.2\%}$ (0.1)
Australian	$72.51 \pm 5.47\%$	$\mathbf{74.11 \pm 4.04\%}$ (0.1)	$67.73 \pm 0.22\%$	$\mathbf{67.97 \pm 0.43\%}$ (0.5)
German	$53.2 \pm 2.91\%$	$\mathbf{78.07 \pm 0.95\%}$ (0.2)	$77.07 \pm 0.35\%$	$\mathbf{77.77 \pm 1.02\%}$ (0.1)
Heart	$57.41 \pm 6.1\%$	$\mathbf{80 \pm 1.96\%}$ (0.3)	$78.02 \pm 0.86\%$	$\mathbf{78.4 \pm 0.93\%}$ (0.1)
Ionosphere	$72.22 \pm 0.97\%$	$\mathbf{91.55 \pm 0.44\%}$ (0.4)	$90.79 \pm 0.29\%$	$\mathbf{91.27 \pm 0.44\%}$ (0.1)
Iris	$79.33 \pm 4.62\%$	$\mathbf{98.44 \pm 0.38\%}$ (0.4)	$98.22 \pm 0.77\%$	$98.22 \pm 0.77\%$ (0.4)
Madelon	$63.08 \pm 1.14\%$	$\mathbf{63.29 \pm 0.75\%}$ (0.5)	$63.01 \pm 0.43\%$	$\mathbf{63.32 \pm 0.49\%}$ (0.5)
Optdigits	$99.04 \pm 0.01\%$	$\mathbf{99.24 \pm 0.01\%}$ (0.1)	$99 \pm 0.01\%$	$\mathbf{99.24 \pm 0.01\%}$ (0.1)
Segmentation	$47.6 \pm 8.94\%$	$\mathbf{90.97 \pm 0.15\%}$ (0.2)	$88.95 \pm 0.24\%$	$\mathbf{90.59 \pm 0.15\%}$ (0.1)
TAE	$49.37 \pm 3.71\%$	$\mathbf{53.46 \pm 2.18\%}$ (0.6)	$46.74 \pm 4.11\%$	$\mathbf{47.4 \pm 3.23\%}$ (0.6)

account the class labels of the samples on each database. That is, we randomly split the samples belonging to each class in five sets and we use four sets of all classes for training and the remaining ones for testing. This process is performed five times, one for each evaluation set. The performance of each algorithm is measured by calculating the mean classification rate over all folds. We perform five experiments on each dataset and report the mean classification rate and the observed standard deviation value for each algorithm. The results are shown in Tables 2-4. In these Tables we also report the probability value p by the ELM variants exploiting the proposed Dropout/Dropconnect regularization term. As can be seen, the incorporation of the proposed regularization term in each ELM variant enhances its performance. It is interesting to see that this improvement is very high for the standard ELM algorithm in most datasets. For the remaining ELM variants, already exploiting a regularization term, the improvement observed is not so impressive. However, it can be seen that in most of the cases, the incorporation of the proposed regularizer improves performance.

Table 3: Performance on UCI datasets for MVELM and MCVELM algorithms.

Dataset	MVELM	DropMVELM	MCVELM	DropMCVELM
Abalone	$55.94 \pm 0.32\%$	$\mathbf{56} \pm \mathbf{0.38}\%$ (0.1)	$55.94 \pm 0.34\%$	$\mathbf{56} \pm \mathbf{0.38}\%$ (0.1)
Australian	$73.04 \pm 0.38\%$	$\mathbf{74.54} \pm \mathbf{0.44}\%$ (0.1)	$73.04 \pm 0.38\%$	$\mathbf{74.54} \pm \mathbf{0.44}\%$ (0.1)
German	$78.2 \pm 1.11\%$	$\mathbf{78.77} \pm \mathbf{1.07}\%$ (0.1)	$78.27 \pm 0.65\%$	$\mathbf{78.63} \pm \mathbf{0.98}\%$ (0.1)
Heart	$78.89 \pm 1.9\%$	$\mathbf{80} \pm \mathbf{1.96}\%$ (0.3)	$78.64 \pm 1.54\%$	$\mathbf{80} \pm \mathbf{1.96}\%$ (0.3)
Ionosphere	$91.08 \pm 0.6\%$	$\mathbf{91.55} \pm \mathbf{0.44}\%$ (0.4)	$90.32 \pm 0.29\%$	$\mathbf{91.55} \pm \mathbf{0.44}\%$ (0.4)
Iris	$98.22 \pm 0.77\%$	$\mathbf{98.44} \pm \mathbf{0.38}\%$ (0.4)	$98.22 \pm 0.38\%$	$\mathbf{98.44} \pm \mathbf{0.38}\%$ (0.2)
Madelon	$63.08 \pm 1.14\%$	$\mathbf{63.29} \pm \mathbf{0.75}\%$ (0.5)	$63.09 \pm 1.12\%$	$\mathbf{63.29} \pm \mathbf{0.75}\%$ (0.5)
Optdigits	$99.09 \pm 0.01\%$	$\mathbf{99.24} \pm \mathbf{0.01}\%$ (0.2)	$99.09 \pm 0.01\%$	$\mathbf{99.24} \pm \mathbf{0.01}\%$ (0.2)
Segmentation	$92.84 \pm 0.2\%$	$\mathbf{93.74} \pm \mathbf{0.2}\%$ (0.1)	$92.86 \pm 0.17\%$	$\mathbf{93.74} \pm \mathbf{0.2}\%$ (0.1)
TAE	$51.41 \pm 2.2\%$	$\mathbf{53.24} \pm \mathbf{2.55}\%$ (0.3)	$52.53 \pm 3.84\%$	$\mathbf{53.46} \pm \mathbf{2.18}\%$ (0.6)

Table 4: Performance on UCI datasets for DRELM and GEELM algorithms.

Dataset	DRELM	DropDRELM	GEELM	DropGEELM
Abalone	$51.38 \pm 0.26\%$	$\mathbf{55.99} \pm \mathbf{0.39}\%$ (0.1)	$55.63 \pm 0.2\%$	$\mathbf{56.04} \pm \mathbf{0.38}\%$ (0.1)
Australian	$67.97 \pm 0.38\%$	$\mathbf{74.54} \pm \mathbf{0.44}\%$ (0.1)	$73.09 \pm 0.44\%$	$\mathbf{74.54} \pm \mathbf{0.36}\%$ (0.1)
German	$77.7 \pm 1.04\%$	$\mathbf{78.67} \pm \mathbf{1.01}\%$ (0.1)	$77.77 \pm 0.47\%$	$\mathbf{78.5} \pm \mathbf{0.69}\%$ (0.1)
Heart	$78.64 \pm 0.74\%$	$\mathbf{80} \pm \mathbf{1.96}\%$ (0.3)	$78.77 \pm 1.54\%$	$\mathbf{79.75} \pm \mathbf{1.83}\%$ (0.3)
Ionosphere	$91.17 \pm 0.29\%$	$\mathbf{91.55} \pm \mathbf{0.44}\%$ (0.4)	$88.89 \pm 0.76\%$	$\mathbf{91.65} \pm \mathbf{0.33}\%$ (0.5)
Iris	$98.22 \pm 0.77\%$	$\mathbf{98.67} \pm \mathbf{0.01}\%$ (0.3)	$97.56 \pm 0.38\%$	$\mathbf{98.44} \pm \mathbf{0.38}\%$ (0.2)
Madelon	$63.08 \pm 0.52\%$	$\mathbf{63.53} \pm \mathbf{0.93}\%$ (0.1)	$63.86 \pm 0.8\%$	$\mathbf{64.04} \pm \mathbf{0.84}\%$ (0.2)
Optdigits	$99.09 \pm 0.01\%$	$\mathbf{99.24} \pm \mathbf{0.01}\%$ (0.2)	$99.24 \pm 0.01\%$	$99.24 \pm 0.01\%$ (0.1)
Segmentation	$90.39 \pm 0.17\%$	$\mathbf{93.74} \pm \mathbf{0.2}\%$ (0.1)	$92.68 \pm 0.13\%$	$\mathbf{93.74} \pm \mathbf{0.16}\%$ (0.1)
TAE	$49.86 \pm 3.11\%$	$\mathbf{53.46} \pm \mathbf{2.18}\%$ (0.6)	$50.8 \pm 0.99\%$	$\mathbf{53.24} \pm \mathbf{2.55}\%$ (0.3)

Table 5: Training times (in seconds) in Abalone and OptDigits datasets.

Dataset	Abalone	OptDigits
ELM	0.5011	0.3937
DropELM	0.507	0.4055
RELM	0.5	0.4017
DropRELM	0.5003	0.4097
MVELM	1.1779	1.0544
DropMVELM	1.1897	1.0688
MCVELM	2.1003	2.674
DropMCVELM	2.1197	2.6917
DRELM	3.5806	3.5257
DropDRELM	3.5989	3.5464
GEELM	3.6832	3.5808
DropGEELM	3.7085	3.5993

In Table 5, we also report the time required for training each algorithm on the two largest UCI datasets used in our experiments. All the experiments were conducted on a 4-core, i7 4790, 3.6GHz PC with 32GB RAM using a MATLAB implementation. As can be seen in this Table, the exploitation of the proposed regularization term inserts a computational overhead, but the differences in training time are very low. This was expected, since the proposed regularization term exploits the already computed hidden layer outputs used in ELM-based neural network training.

In our second set of experiments, we have applied the ELM algorithms on the facial image datasets. Since there is no widely adopted experimental protocol for these datasets, we also perform the five-fold cross-validation procedure [42], by taking into account the class labels of the samples on each database, as in the UCI datasets. The observed performance is shown in Tables 6-8. Similar to the results obtained for the UCI datasets, the incorporation of the proposed regularization term enhances the performance of all the ELM variants in most datasets.

In addition, we compare the performance obtained by applying the proposed approach on the facial image datasets with some recently published state-of-the-art methods in Tables 9 and 10. As can be seen in these Tables, the proposed approach provides satisfactory performance in all cases.

Table 6: Performance on facial image datasets for ELM and RELM algorithms.

Dataset	ELM	DropELM	RELM	DropRELM
BU	57.57 ± 1.08%	**62.38 ± 1.43%** (0.3)	61.43 ± 1.38%	**62.48 ± 1.44%** (0.2)
Jaffe	55.24 ± 0.82%	**56.03 ± 1.2%** (0.2)	55.71 ± 0.95%	**56.19 ± 0.95%** (0.1)
Kanade	65.31 ± 2.04%	**66.67 ± 0.62%** (0.6)	65.71 ± 1.08%	**65.99 ± 0.62%** (0.1)
AR	97.65 ± 0.23%	**99.19 ± 0.17%** (0.1)	97.65 ± 0.17%	**98.91 ± 0.23%** (0.1)
ORL	98.33 ± 0.14%	**98.42 ± 0.14%** (0.2)	98.33 ± 0.14%	**98.42 ± 0.14%** (0.1)
Yale	98.14 ± 0.34%	**98.15 ± 0.33%** (0.1)	96.82 ± 0.17%	**97.31 ± 0.01%** (0.1)

Table 7: Performance on facial image datasets for MVELM and MCVELM algorithms.

Dataset	MVELM	DropMVELM	MCVELM	DropMCVELM
BU	61.62 ± 1.62%	**62.38 ± 1.43%** (0.3)	61 ± 0.57%	**62.48 ± 1.44%** (0.3)
Jaffe	55.71 ± 0.95%	**56.19 ± 0.95%** (0.1)	56.83 ± 0.55%	**58.89 ± 0.73%** (0.1)
Kanade	65.71 ± 1.08%	**66.67 ± 0.62%** (0.6)	66.39 ± 1.31%	**68.03 ± 2.01%** (0.1)
AR	99.21 ± 0.27%	99.21 ± 0.27% (0.1)	99.24 ± 0.29%	**99.33 ± 0.16%** (0.1)
ORL	98.42 ± 0.14%	98.42 ± 0.14% (0.1)	98.75 ± 0.14%	**98.83 ± 0.14%** (0.1)
Yale	98.03 ± 0.01%	**98.07 ± 0.01%** (0.1)	98.03 ± 0.01%	**98.08 ± 0.01%** (0.1)

Table 8: Performance on facial image datasets for DRELM and GEELM algorithms.

Dataset	DRELM	DropDRELM	GEELM	DropGEELM
BU	61.81 ± 1.84%	**62.48 ± 1.44%** (0.3)	61.81 ± 0.57%	**62.57 ± 1.11%** (0.3)
Jaffe	58.89 ± 1.1%	**59.21 ± 0.99%** (0.4)	58.89 ± 0.82%	**59.21 ± 0.48%** (0.3)
Kanade	66.67 ± 0.62%	**68.44 ± 1.31%** (0.1)	66.67 ± 0.62%	**68.98 ± 2.04%** (0.2)
AR	99.21 ± 0.2%	**99.33 ± 0.2%** (0.2)	99.23 ± 0.28%	**99.3 ± 0.29%** (0.2)
ORL	98.5 ± 0.14%	**98.83 ± 0.14%** (0.1)	98.75 ± 0.25%	**98.83 ± 0.14%** (0.2)
Yale	97.74 ± 0.18%	**98.07 ± 0.12%** (0.1)	98.16 ± 0.17%	**98.23 ± 0.15%** (0.1)

Table 9: Comparison of our results with some state-of-the-art methods on the BU, Jaffe and Kanade datasets.

	BU	Jaffe	Kanade
Method [43]	-	56.72%	**69.05%**
Method [44]	**66.4%**	-	**72.9%**
Method [45] - MMP	-	-	**70.3%**
Method [45] - RP	-	-	**75.2%**
Method [45] - SVM	-	-	**73.4%**
Method [45] - MMRP	-	-	**80.1%**
Proposed method	**62.57%**	**59.21%**	**68.98%**

5. Conclusions

In this paper, we proposed an extension of the Extreme Learning Machine algorithm for Single-hidden Layer Feedforward Neural network training that incorporates Dropout and DropConnect regularization in its optimization process. We have shown that Dropout and DropConnect applied to the network hidden layer outputs and the network output weights, respectively, lead to the same type of regularization, which is incorporated in the proposed DropELM algorithm. Experimental results showed that the incorporation of the proposed regularization term in several recently proposed ELM variants leads to enhanced classification performance, without requiring iterative network weights tuning.

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 A.

Here we discuss the importance of the constraint (17) in ELM optimization process. In the proposed method the network output weights $\mathbf{W}_{out}$ are calculated by:

$$\mathbf{W}_{out} = \left(\mathbf{\Phi}\mathbf{\Phi}^T + \frac{1}{c}\mathbf{S} + \frac{\lambda}{c}\mathbf{R}_1 \right)^{-1} \mathbf{\Phi}\mathbf{T}^T. \tag{A.1}$$

Table 10: Comparison of our results with some state-of-the-art methods on the AR, ORL and Yale datasets.

	AR	ORL	Yale
Method [46]	95.7%	-	98.1%
Method [47]	-	94.35%	94.76%
Method [48]	97%	-	-
Method [49]	74.67%	83.89%	-
Method [50] - LRC	-	98.25%	-
Method [50] - SRC	-	98.37%	-
Method [50] - WGSRC	-	**98.93**%	-
Method [45] - MMP	-	-	91%
Method [45] - RP	-	-	89.4%
Method [45] - SVM	-	-	85.6%
Method [45] - MMRP	-	-	97.2%
Method [51]	**99.5**%	-	-
Proposed method	**99.33**%	**98.83**%	**98.23**%

Following an approach similar to that of [27, 28], we have:

$$\mathbf{S} = r_1 \mathbf{\Phi} \mathbf{L} \mathbf{\Phi}^T + r_2 \mathbf{I}, \tag{A.2}$$

where it is assumed that the data representations in the ELM space are employed in order to form the vertices of a graph and $\mathbf{L}$ is the corresponding graph Laplacian matrix. In addition, the regularization term $\mathbf{R}_1$ can be expressed as follows:

$$\mathbf{R}_1 = (1-p)^2 \mathbf{\Phi}\mathbf{\Phi}^T + (1-p)^2 \mathbf{D}, \tag{A.3}$$

where $\mathbf{D} = (\mathbf{\Phi}\mathbf{\Phi}^T) \circ \mathbf{I}$. By substituting (A.2) and (A.3) in (A.1), we have:

$$\begin{aligned}
\mathbf{W}_{out} &= \left(\mathbf{\Phi}\mathbf{\Phi}^T + \frac{r_1}{c}\mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^T + \frac{r_2}{c}\mathbf{I} + \frac{\lambda(1-p)^2}{c}\mathbf{\Phi}\mathbf{\Phi}^T + \frac{\lambda(1-p)^2}{c}\mathbf{D} \right)^{-1} \mathbf{\Phi}\mathbf{T}^T \\
&= \left[\mathbf{\Phi} \left(\frac{c + \lambda(1-p)^2}{c}\mathbf{I} + \frac{r_1}{c}\mathbf{L} \right) \mathbf{\Phi}^T + \frac{1}{c}\left(r_2 \mathbf{I} + \lambda(1-p)^2 \mathbf{D} \right) \right]^{-1} \mathbf{\Phi}\mathbf{T}^T \\
&= \mathbf{A}^{-1}\mathbf{\Phi}\mathbf{T}^T. \qquad \text{(A.4)}
\end{aligned}$$

By observing the first term of $\mathbf{A} = \mathbf{Q} + \frac{1}{c}\mathbf{E}$, we can see that the incorporation of the constraint (17) in the ELM optimization process equally contributes to the diagonal of the matrix $\mathbf{Q} = \left(\frac{c+\lambda(1-p)^2}{c}\mathbf{I} + \frac{r_1}{c}\mathbf{L}\right)$. Thus, we would expect it to have minimal importance in the optimization process (since it is just a constant multiplication factor). On the other hand, the elements of diagonal matrix $\mathbf{E}$ have values equal to:

$$[\mathbf{E}]_{ii} = r_2 + \lambda(1-p)^2 \sum_{j=1}^{N} \boldsymbol{\phi}_{j,i}^2, \quad i = 1, \ldots, L. \tag{A.5}$$

That is, the incorporation of the constraint (17) in the ELM optimization process has the effect of changing the regularization term of (A.4), according to the importance of each ELM space dimension. This fact leads to better regularization of the matrix $\mathbf{A}$ and is expected to lead to better solutions.

As an example, let us consider the case where some (of the randomly chosen) network hidden layer weights are not appropriately chosen. For instance, in the case where the RBF activation function is used this situation may occur when some of the weights are far from all the training samples, leading to similar hidden layer neuron outputs (close to zero) for all the training samples. In the case where a sigmoid function is used, a similar result may occur for hidden layer weights close to zero. Clearly, the dimensions of the ELM space corresponding to such hidden layer weights do not contain much discriminative power, since the responses for all the training data are very similar. On the other hand, appropriately chosen weights, will lead to all sorts of response values for the training data. For example, in the case where the RBF activation function is used, the response values for training data similar to the weights will be large, the response values for training data at a distance from the weights will be moderate, while the response values for the remaining data will be close to zero. Thus, the ELM space dimensions corresponding to such weights will contain high discriminative power. The adoption of the regularization term in (A.5) will result to a lower regularization factor for the ELM space dimensions of the first category, when compared to ELM space dimensions corresponding to the latter one.

References

[1] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, IEEE International Joint Conference on Neural Networks 2 (2004) 985–990.

[2] D. Rumelhart, G. Hinton, R. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.

[3] 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 44 (2) (1998) 525–536.

[4] R. Zhang, Y. Lan, G. Huang, Z. Zu, Universal approximation of extreme learning machine with adaptive growth of hidden nodes, IEEE Transactions on Neural Networks and Learning Systems 23 (2) (2012) 365–371.

[5] G. Huang, L. Chen, C. Siew, Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Nodes, IEEE Transactions on Neural Networks 17 (4) (2006) 879–892.

[6] G. Huang, Y. Chen, H. Babri, Classification ability of single hidden layer feedforward neural networks, IEEE Transactions on Neural Networks 11 (3) (2000) 799–801.

[7] K. Hornik, Some new results on neural network approximation, Neural Networks 6 (8) (1993) 1069–1072.

[8] H. Rong, G. Huang, Y. Ong, Extreme learning machine for multi-categories classification applications, IEEE International Joint Conference on Neural Networks (2008) 1709–1713.

[9] B. Chacko, V. Krishnan, G. Raju, B. Anto, Handwritten character recognition using wavelet energy and extreme learning machine, International Journal of Machine Learning and Cybernetics 3 (2) (2012) 149–161.

[10] R. Minhas, A. Baradarani, S. Seifzadeh, Q. Jonathan Wu, Human action recognition using extreme learning machine based on visual vocabularies, Neurocomputing 73 (10-12) (2010) 1906–1917.

[11] Y. Lan, Y. Soh, G. Huang, Extreme Learning Machine based bacterial protein subcellular localization prediction, IEEE International Joint Conference on Neural Networks (2008) 1859–1863.

[12] T. Helmy, Z. Rasheed, Multi-category bioinformatics dataset classification using extreme learning machine, IEEE Evolutionary Computation (2009) 3234–3240.

[13] A. Iosifidis, A. Tefas, I. Pitas, Multi-view Human Action Recognition under Occlusion based on Fuzzy Distances and Neural Networks, European Signal Processing Conference (2012) 1129–1133.

[14] A. Iosifidis, A. Tefas, I. Pitas, Dynamic action recognition based on Dynemes and Extreme Learning Machine, Pattern Recognition Letters 34 (2013) 1890–1898.

[15] W. Zong, G. Huang, Face recognition based on Extreme Learning Machine, Neurocomputing 74 (16) (2011) 2541–2551.

[16] M. Li, G. Huang, P. Saratchandran, N. Sundararajan, Fully complex extreme learning machine, Neurocomputing 68 (13) (2005) 306–314.

[17] N. Liang, G. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate on-line sequantial learning algorithm for feedforward networks, IEEE Transactions on Neural Networks 17 (6) (2006) 1411–1423.

[18] G. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16) (2008) 3056–3062.

[19] G. Feng, G. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Transactions on Neural Networks 20 (8) (2009) 1352–1357.

[20] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: Optimally pruned extreme learning machine, IEEE Transactions on Neural Networks 21 (1) (2010) 158–162.

[21] G. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42 (2) (2012) 513–529.

[22] A. Iosifidis, A. Tefas, I. Pitas, Minimum Class Variance Extreme Learning Machine for Human Action Recognition, IEEE Transactions on Circuits and Systems for Video Technology 23 (11) (2013) 1968–1979.

[23] G. Huang, S. Song, J. Gupta, C. Wu, Semi-supervised and Unsupervised Extreme Learning Machine, IEEE Transactions on Cybernetics 44 (12) (2014) 2405–2417.

[24] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutsekever, R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv:1207.0580 [cs.NE] .

[25] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, R. Fergus, Regularization of Neural Network using DropConnect, International Conference on Machine Learning (2013) 1–9.

[26] A. Iosifidis, A. Tefas, I. Pitas, Minimum Class Variance Extreme Learning Machine for human action recognition, IEEE Transactions on Circuits and Systems for Video Technology 23 (11) (2013) 1968–1979.

[27] A. Iosifidis, A. Tefas, I. Pitas, Regularized extreme learning machine for multi-view semisupervised action recognition, Neurocomputing 145 (2014) 250–262.

[28] Y. Peng, S. Wang, B. Long, X.and Lu, Discriminative graph regularized Extreme Learning Machine for face recognition, Neurocomputing 149 (2015) 340–353.

[29] A. Iosifidis, A. Tefas, I. Pitas, Graph Embedded Extreme Learning Machine, IEEE Transactions on Cybernetics, accepted January 2015 .

[30] E. Million, The Hadamard Product, Technical Report (2007) 1–7.

[31] R. Horn, Topics in Matrix Analysis, Cambridge University Press, 1994 .

[32] P. Sen, J. Singer, Large sample methods in statistics, Chapman & Hall, 1993 .

[33] A. Frank, A. Asuncion, UCI Machine Learning Repository, 2010.

[34] W. Deng, Q. Zheng, K. Zhang, Reduced Extreme Learning Machine, International Conference on Computer Recognition Systems (2013) 63–69.

[35] A. Iosifidis, A. Tefas, I. Pitas, On the Kernel Extreme Learning Machine Classifier, Pattern Recognition Letters 54 (2015) 11–17.

[36] A. Martinez, A. Kak, Pca versus lda, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (2) (2001) 228–233.

[37] F. Samaria, A. Harter, Parameterisation of a stochastic model for human face identification, IEEE Workshop on Applications of Computer Vision (1994) 138–142.

[38] K. Lee, J. Ho, D. Kriegman, Acquiriing linear subspaces for face recognition under variable lighting, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (5) (2005) 684–698.

[39] L. Yin, X. Wei, Y. Sun, J. Wang, M. Rosato, A 3D facial expression database for facial behavior research, IEEE International Conference on Automatic Face and Gesture Recognition (2006) 211–216.

[40] T. Kanade, Y. Tian, J. Cohn, Comprehensive database for facial expression analysis, IEEE International Conference on Automatic Face and Gesture Recognition (2000) 46–53.

[41] M. Lyons, S. Akamatsu, M. Kamachi, J. Gyoba, Coding facial expressions with Gabor wavelets, IEEE International Conference on Automatic Face and Gesture Recognition (1998) 200–205.

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

[43] S. Nikitidis, A. Tefas, I. Pitas, Subclass Discriminant Nonnegative Matrix Factorization for facial image analysis, Pattern Recognition 45 (2012) 4080–4091.

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

[45] S. Nikitidis, A. Tefas, I. Pitas, Maximum Margin Projection Subspace Learning for Visual Data Analysis, IEEE Transactions on Image Processing 23 (10) (2014) 4413–4425.

[46] J. Wright, A. Yang, A. Ganesh, S. S.S., Y. Ma, Robust Face Recognition via Sparse Representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 1–18.

[47] X. Liu, S. Yan, J. H., Projective Nonnegative Graph Embedding, IEEE Transactions on Image Processing 19 (5) (2010) 1126–1137.

[48] A. James, One-sample face recognition with local similarity decisions, International Journal of Applied Pattern Recognition 1 (1) (2013) 61–80.

[49] Z. Sun, K. Lam, Z. Dong, H. Wang, Q. Gao, C. Zheng, Face Recognition with Multi-Resolution Spectral Feature Images, PLOS ONE, DOI: 10.1371/journal.pone.0055700 .

[50] X. Tang, G. Feng, J. Cai, Weighted group sparse representation for undersampled face recognition, Neurocomputing 145 (2014) 402–415.

[51] T. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, Y. Ma, PCANet: A Simple Deep Learning Baseline for Image Classification?, arXiv:1404.3606v2 .