Quick Hidden Layer Size Tuning in ELM for Classification Problems

Audi Albtoush[1,✉], **Manuel Fernandez-Delgado**[2], **Haitham Maarouf**[2], **Asmaa Jameel Al Nawaiseh**[3]

[1]Faculty of Computer Science and Information Technology, Jerash University, Jordan
[2]Santiago De Compostela University, Spain
[3]Software Engineering Department, Mut'ah university, Jordan
A.albtoush@jpu.edu.jo✉

Abstract

The extreme learning machine is a fast neural network with outstanding performance. However, the selection of an appropriate number of hidden nodes is time-consuming, because training must be run for several values, and this is undesirable for a real-time response. We propose to use moving average, exponential moving average, and divide-and-conquer strategies to reduce the number of training's required to select this size. Compared with the original, constrained, mixed, sum, and random sum extreme learning machines, the proposed methods achieve a percentage of time reduction up to 98% with equal or better generalization ability.

Keywords: *Extreme Learning Machine, Number of Hidden Nodes, Moving Average, Exponential Moving Average, Divide-and-conquer.*

Received: 03 December 2023
Accepted: 20 December 2023
Online: 04 January 2024
Published: 15 July 2024

1 Introduction

The extreme learning machine (ELM) was proposed by Huang et al. [14] to reduce the computational cost of back-propagation training in feed-forward neural networks. The distinguishing feature of ELM is that input weights and biases of hidden neurons are randomly generated, while output weights are calculated from activation's in the hidden layer and true outputs using matrix pseudo-inversion. This avoids iterative training and saves much time compared to existing networks [21]. The number of the hidden nodes defines the network size and influences the ELM performance and speed [26, 4]. However, a great generalization requires to choose the optimal hidden node and this requires time [16, 17]. As well, when the network size does not fit the data, disappointing results may be achieved [31, 5, 30].

There are four main groups of methods to select the hidden layer size. 1) **Random selection** [15, 2] that may lead to over-fitting or under-fitting problems [24]. An example is I-ELM [12], that randomly generates hidden nodes each time, adding to the ELM only the most appropriated. 2) **Constructive methods**, that add dynamically hidden nodes, are consuming-time [18] and may lead to sub-optimal sizes [9]. Some of these methods [11] add nodes one by one freezing the weights of existing nodes, stopping when a pre-set maximum size or training performance is achieved. The improved incremental ELM [26] adds at each learning step an additional offset to the output matrix of the hidden layer before calculating the output weights of the new hidden node. 3) **Pruning methods**, that start with a large hidden layer and reduce its size using statistical criteria, sparse regression [25] or multiple response [21]. The pruned ELM [22] starts with a large hidden layer and then gets rid of the nodes with the lowest relevance to the class labels using a chi-squared test and an information gain criterion. 4) **Other approaches**. The paper [29] reduces the network size to save training time and updates the weights in the direction of maximum reduction in squared error. Real-coded genetic algorithms [27] were used to select the optimal number of the number of hidden nodes, alongside with their input weights and bias. Particle swarm optimization [16] and singular value decomposition [7] were also used to optimize the hidden layer size of the ELM the paper [4] avoids the tuning by using a bounded estimation of the hidden layer size from the data population [6][28][23][3].

The existing approaches attempt to improve the ELM size but are iterative and time-consuming, and may lead to over-fitting or underfitting. We propose three methods, MA-ELM, EMA-ELM and DC-ELM, which use moving average, exponential moving average and divide-and-conquer, respectively, to optimize the search for a good number of hidden nodes. The section 1 in the supplementary material briefly describes the ELM network, whose architecture is shown in Figure 1, while algorithm 1 lists its pseudo-code. Section 5-7 describe the details of the proposed methods, whose results are reported in section 13, while section 14 summarizes the conclusions of this work.

2 Extreme Learning Machine

The ELM is a fast learning method based on random weight initialization and Moore-Penrose generalized inversion of the hidden layer output matrix. Fig. 1 shows the ELM architecture. The main and most attractive

ELM characteristics are:

- The bias and input weights of the hidden nodes are random values.
- Since the hidden layer weights are independent of the input sample, they do not require iterative updates.
- The output weights of the hidden nodes are computed using an analytical non-iterative method.

Paper [13] showed the ability of the SLFN to set randomly the input weights and bias. Compared with traditional SLFNs, the original ELM randomly fixes the input weights and bias and analytically determines output weights. Let N be the number of training patterns, I the number of ELM inputs, C the number of ELM outputs (classes in a classification problem) and H the number of hidden nodes. Given a training pattern $\mathbf{x}_n = (x_{n1}, \ldots, x_{nI}, 1) \in^{I+1}$, the output function $f_c(\mathbf{x}_n)$ of the c-th output neuron, with $c = 1, \ldots, C$, is given by:

$$f_c(\mathbf{x}_n) = \sum_{h=1}^{H} b_{hc} g(\mathbf{a}_h^T \mathbf{x}_n), \quad c = 1, \ldots, C \tag{1}$$

where $g(\cdot)$ is the activation function, while $\mathbf{a}_h = (a_{h1}, \ldots, a_{h,I+1}) \in^{I+1}$ is the weight vector connecting the input layer and the h-th hidden neuron, $a_{h,I+1}$ is the bias of hidden neuron h, and b_{hc} is the output weight between the h-th hidden and the c-th output neurons. For N training examples $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^{N} \in^{I+1} \times^{C}$ with true outputs $\mathbf{t}_n = (t_{n1}, \ldots, t_{nC})$, the ELM can approximate the output with zero error, so that:

$$\sum_{n=1}^{N} \sum_{c=1}^{C} |f_c(\mathbf{x}_n) - t_{nc}| = 0 \tag{2}$$

so there exist $\mathbf{a}_h$ and $\mathbf{b}_h = (b_{h1}, \ldots, b_{hC}) \in^{C}$ such that:

$$\sum_{h=1}^{H} b_{hc} g(\mathbf{a}_h^T \mathbf{x}_n) = t_{nc},\ n = 1, \ldots, N, c = 1, \ldots, C \tag{3}$$

The above N equations can be written compactly as $\mathbf{HB} = \mathbf{T}$, where $\mathbf{H}$ is a $N \times H$-order matrix with items $H_{nh} = g(\mathbf{a}_h^T \mathbf{x}_n)$, while $\mathbf{B} = \{b_{hc}\}_{hc=1}^{H,C}$ and $\mathbf{T} = \{t_{nc}\}_{nc=1}^{N,C}$. The input weights are randomly generated, and the $H \times C$-order matrix $\mathbf{B}$ with the output weights can be calculated as $\mathbf{B} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose pseudo-inverse of $\mathbf{H}$. In general, the input data $\mathbf{X}$ and weights $\mathbf{A}$ must be scaled in the range [-1,1]. The ELM training method is compiled by algorithm 5.

3 Algorithm of MA-ELM and EMA-ELM

Algorithm 2 lists the operation of MA-ELM and EMA-ELM. In this code, the function ELM is defined in algorithm 1. The function ACC calculates the accuracy of ELM trained using the corresponding H argument. The notation $t_n, z_n \leftarrow \arg\max_{c=1,\ldots,C} \{t_{cn}, v_{cn}\}$ means that t_n (resp. z_n) is the argument that maximizes t_{cn} (resp. v_{cn}) over $c = 1, \ldots, C$, where v_{cn} are the outputs of the output neurons (see Figure 1). The function $\delta(t_n, z_n)$ is 1 when $t_n = z_n$ and 0 otherwise. The notation $M_k|\mathcal{E}_k$ in lines 10 and 16 means that function M_k given by Eq. (1) (resp. function $\mathcal{E}_k$ in Eq. (2)) in the paper is used for MA-ELM (resp. for EMA-ELM).

4 Algorithm of DC-ELM

Algorithm 3 describes the procedure for DC-ELM, where $|\mathcal{H}|$ denotes the cardinal of set $\mathcal{H}$ and function ACC is defined in algorithm 2.

5 Moving Average Extreme Learning Machine

The relationship between the number H of hidden neurons and the network performance is not subject to a specific statistical distribution or type of non-linearity. So, we propose to select a good H value using the moving average (MA), a simple statistical tool and indicator for technical analysis, that makes extensive use of the points to be estimated, widely applied in science, engineering and financial applications [19, 20]. The MA creates an averaging of a set of user-defined and constantly updated values A_k, in our case the training accuracy of ELM using different values H_k of H. The objective is to smooth trends in values by filtering out noise from fluctuations of extreme values. The n-width moving average M_{k+1} of accuracy A_k, for $k = 1, 2, \ldots$, is defined by Eq. (4):

$$M_{k+1} = \frac{A_k + A_{k-1} + \ldots + A_{k-n+1}}{n} = \frac{1}{n} \sum_{i=1}^{n} A_{k-i+1}, \quad k = n+1, \ldots, K \tag{4}$$

Thus, M_{k+1} is the average of the previous n values $A_k, \ldots, A_{k-n+1}$, that compose the "data window" of width n. In the applications of MA, often this width n is randomly selected from the set {2,3,4}. Let $\{H_k\}_{k=1}^{K}$ be a set of K equally spaced values of H defined as $H_k = H_1 + (k-1)\Delta H$, for $k = 1, \ldots, K$. Our proposal, named MA-ELM, uses MA to estimate the accuracy A_{n+1} that ELM is expected to achieve with H_{n+1} hidden neurons. This estimation of Eq. (4), denoted as A'_{n+1}, uses the n previous accuracies $\{A_{k-i+1}\}_{i=1}^{n}$ achieved by ELM with $\{H_{k-i+1}\}_{k=1}^{n}$ hidden neurons (these accuracies are calculated by training ELM, so they require time), so that $A'_{n+1} = M_{n+1}$. The A'_{n+1} is compared to A_{n+1}, that is the true accuracy calculated by training the ELM using H_{n+1} hidden neurons. A difference $|A'_{n+1} - A_{n+1}|$ above a threshold τ is considered high, so that MA-ELM uses the true value A_{n+1}. In this case, for $n+2$ both the true A_{n+2} and estimated A'_{n+2} accuracies are calculated, what requires to train

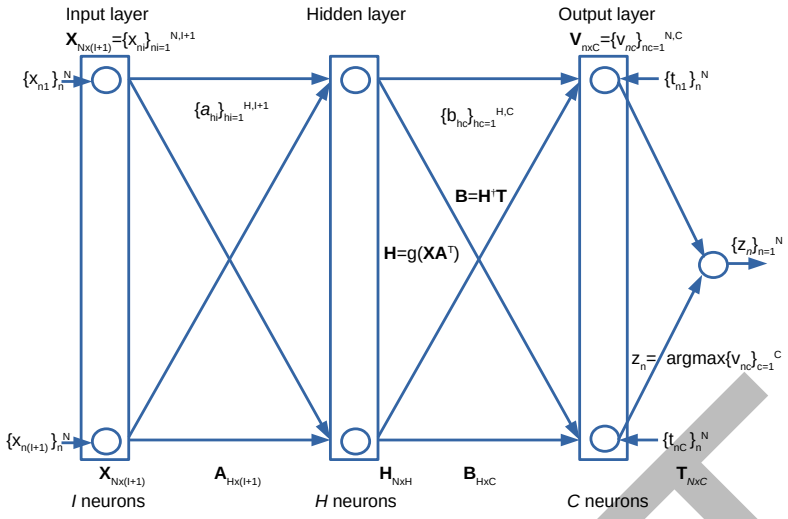

Figure 1: Architecture of the extreme learning machine.

Algorithm 1 Extreme learning machine (ELM). $[\mathbf{A}, \mathbf{B}]$=ELM($\mathbf{X}$,$\mathbf{T}$,H,g)

Data: $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N}$: training set, $\mathbf{x}_n \in^{I+1}$; $\mathbf{T} = \{t_{nc}\}_{nc=1}^{N,C}$: true output; H: number of hidden neurons; g: activation function
Randomly assign $H \times (I+1)$-order input weight matrix $\mathbf{A}$ from some probability density function.
Calculate the hidden layer activity matrix $\mathbf{H} = g(\mathbf{X}\mathbf{A}^T)$.
Calculate the output weight matrix $\mathbf{B} = \mathbf{H}^\dagger\mathbf{T}$.
Result: $\mathbf{A} = \{a_{hi}\}_{hi=1}^{H,I+1}$: input weights; $\mathbf{B} = \{b_{hc}\}_{hc=1}^{H,C}$: output weights.

Algorithm 2 Algorithm for MA-ELM and EMA-ELM. $[\mathbf{A}, \mathbf{B}]$=MA|EMA-ELM($\mathbf{X}$,$\mathbf{T}$,$\{H_k\}_{k=1}^{K}$,g)

Data: $\mathbf{X} = \{x_{ni}\}_{ni=1}^{N,I+1}$: training set; $\mathbf{T} = \{t_{nc}\}_{nc=1}^{N,C}$: true output; $\{H_k\}_{k=1}^{K}$: set of numbers of hidden neurons; g: activation function
Select randomly $n \in \{2,3,4\}$; $\tau \leftarrow 0.5$; `train`$\leftarrow$`True`.
Function ACC($\mathbf{X}, \mathbf{T}, H, g$):
  $[\mathbf{A}, \mathbf{B}] \leftarrow$ELM($\mathbf{X}, \mathbf{T}, H, g$). $\mathbf{V} = \{v_{nc}\}_{nc=1}^{N,C} \leftarrow g(\mathbf{X}\mathbf{A}^T)\mathbf{B}$.
  $\left\{t_n, z_n \leftarrow \arg\max_{c=1,\ldots,C} \{t_{nc}, v_{nc}\}\right\}_{n=1}^{N}$; $\delta(x,x) = 1$, $\delta(x,y) = 0$; $Acc \leftarrow \frac{100}{N}\sum_{i=1}^{N}\delta(t_n, z_n)$.
  return Acc.
end
for $k \leftarrow 1, K$ **do**
  if `train` **then**
    $A_k \leftarrow$ACC($\mathbf{X}, \mathbf{T}, H_k, g$).
  if $k > n$ **then**
    $A'_k \leftarrow M_k|\mathcal{E}_k$ // M_k calculated using Eq. (1), $\mathcal{E}_k$ using Eq. (2)
    if `train` **and** $|A_k - A'_k| < \tau$ **then**
      $A_k \leftarrow A'_k$; `train`$\leftarrow$`False`.
$k^* \leftarrow \arg\max_{k=1,\ldots,K} \{A_k\}$; $H^* \leftarrow H_{k^*}$; $[\mathbf{A}, \mathbf{B}] \leftarrow$ELM($\mathbf{X}, \mathbf{T}, H^*, g$).
Result: $\mathbf{A} = \{a_{hi}\}_{hi=1}^{H,I+1}$: input weights; $\mathbf{B} = \{b_{hc}\}_{hc=1}^{H,C}$: output weights.

again the ELM for H_{n+2}, and the difference test is repeated. On the contrary, when $|A'_{n+1} - A_{n+1}| \leq \tau$, the difference is considered to be low and MA-ELM accepts the value estimated by moving average, so that $A_k = A'_k = M_k$ for $k = n+2, \ldots, K$. We set τ=0.5, a restrictive value for accuracies, that are in %. When $\{A_k\}_{k=1}^{K}$ are available, either estimated by MA in Eq. (4) or evaluated by ELM training, the selected value H^* for the test is the one with the highest accuracy $H^* = H_{k^*}$, with $k^* = \arg\max_{k=1,\ldots,K}\{A_k\}$. The operation of MA-ELM is compiled by algorithm 2 in section 2 of the supplementary material. It is expected that for

Algorithm 3 DC-ELM Algorithm. $[\mathbf{A}, \mathbf{B}]$=DC-ELM($\mathbf{X}$,$\mathbf{T}$,$\{H_k\}_{k=1}^{K}$,g)

Data: $\mathbf{X} = \{x_{ni}\}_{ni=1}^{N,I+1}$: training set; $\mathbf{T} = \{t_{nc}\}_{nc=1}^{N,C}$: true output; $\{H_k\}_{k=1}^{K}$: set of numbers of hidden neurons; g: activation function
Function DC($\mathcal{H}$)
$l \leftarrow |\mathcal{H}|$; $\mathcal{H}^* \leftarrow \mathcal{H}$.
if $l > 1$ **then**
  $p \leftarrow \lceil l/2 \rceil$; Split $\mathcal{H}$ into subsets $\mathcal{H}_1 = \{H_k\}_{k=1}^{p}$ and $\mathcal{H}_2 = \{H_k\}_{k=p+1}^{l}$.
  if ACC($\mathbf{X}, \mathbf{T}, \max\{\mathcal{H}_1\}, g$)>ACC($\mathbf{X}, \mathbf{T}, \max\{\mathcal{H}_2\}, g$) **then**
    $\mathcal{H}^* \leftarrow \mathcal{H}_1$.
  else
    $\mathcal{H}^* \leftarrow \mathcal{H}_2$.
return $\mathcal{H}^*$.
end
$H^* \leftarrow$DC($\{H_k\}_{k=1}^{K}$); $[\mathbf{A}, \mathbf{B}] \leftarrow$ELM($\mathbf{X}, \mathbf{T}, H^*, g$).
Result: $\mathbf{A} = \{a_{hi}\}_{hi=1}^{H,I+1}$: input weights; $\mathbf{B} = \{b_{hc}\}_{hc=1}^{H,C}$: output weights.

many H_k values the A_k will be estimated using MA, thus saving the training times. In this case, MA-ELM would be faster than ELM. In the opposite case, if the difference test was never true all the A_k values would require to train the ELM and no time would be saved.

6 Exponential Moving Average Extreme Learning Machine

Exponential moving average (EMA) is a type of MA widely in artificial intelligence and deep learning [8], that predicts a value (ELM accuracy A_k) weighting differently the last value and previous values. The n-width exponential moving average of A_k, denoted as $\mathcal{E}_{k+1}$, is defined by Eq. (5):

$$\mathcal{E}_k = (1-\gamma)A_k + \frac{\gamma}{n-1}\sum_{i=2}^{n} A_{k-i+1}, \quad \gamma = \frac{2}{1+n} \tag{5}$$

where $n \in \{2, 3, 4\}$ as in MA. Depending on n, the weight $1-\gamma$ of A_k is lower (for n=2) or higher (for $n > 2$) than the weights of the previous samples $\frac{\gamma}{n-1}$, as shown by Table 1. Algorithm 2 in section 2 of the supplementary material lists the pseudo-code for EMA-ELM.

7 Divide-and-conquer Extreme Learning Machine

In artificial intelligence, divide-and-conquer (DC) is a strategy for the design of algorithms characterized by dividing a problem into sub-problems that are simple enough to be directly solved. This strategy was used e.g. to solve kernel support vector machine by clustering data [10], and to apply ELM to big data [7], creating parts of the hidden layer by applying singular value decomposition to subsets of the whole dataset.

We propose to use DC to select the best number H^* of hidden neurons in ELM. The proposed method (DC-ELM) first divides the set $\mathcal{H} = \{H_k\}_{k=1}^{K}$ in two disjoint subsets $\mathcal{H}_1 = \{H_k\}_{k=1}^{p}$ and $\mathcal{H}_2 = \{H_k\}_{k=p+1}^{K}$ with $p = \lceil l/2 \rceil$ is the ceiling integer function applied on $l/2$ and $l = |\mathcal{H}|$ is the number of values in $\mathcal{H}$. Then, DC-ELM proceeds by comparing accuracies achieved by the ELM using the largest H values of $\mathcal{H}_1$ and $\mathcal{H}_2$. The subset that provides the best performance is selected to be splitted again by DC, and the other subset is discarded. The algorithm continues recursively until it reaches a subset with only one value, that corresponds to the best accuracy. However, a reduced number of H values, and of ELM trainings, was executed, so that a (possibly large) amount of time is saved. Algorithm 3 in section 3 of the supplementary material describes DC-ELM.

Fig. 2 shows an example of operation of DC-ELM over a set of $K = 10$ numbers of hidden layer sizes $\mathcal{H} = \{H_k\}_{k=1}^{K} = \{10k\}_{k=1}^{K}$. We set $p = \lceil K/2 \rceil = 5$ and $\mathcal{H}$ is splitted in $\mathcal{H}_1$={10, 20, 30, 40, 50} and $\mathcal{H}_2$={60, 70, 80, 90, 100}. Their highest values are H=50 and H=100, with accuracies (calculated training ELM) 70% and 83%, respectively. Since $83 > 70$, $\mathcal{H}_2$ is selected for the next iteration. Now, K=5 and p=3, so $\mathcal{H}$ is splitted into $\mathcal{H}_1 = \{60, 70, 80\}$ and $\mathcal{H}_2 = \{90, 100\}$, with largest values H=80 (acc=85%) and H=100 (acc=83%), so $\mathcal{H}_1$ is selected. In this case, the accuracy required for H=100 is already calculated in previous iterations. The method proceeds until a subset of size one is achieved, in this example for $H^* = 70$. Fig. 2 plots the training accuracy of ELM and DC-ELM. Each marker is a training execution. The number of trainings is expected to be much less than K, so DC-ELM saves many trainings, and a considerable time, achieving the same performance as ELM.

8 Experimental Methodology

The models MA-ELM, EMA-ELM and DC-ELM were compared with the original ELM and the constrained extreme learning machine, CELM [31], whose code was

Table 1: Values of γ and the weights of the current and previous samples for each value of n.

n	γ	$1-\gamma$	$\gamma/(n-1)$
2	0.67	0.33	0.67
3	0.5	0.5	0.25
4	0.4	0.6	0.13

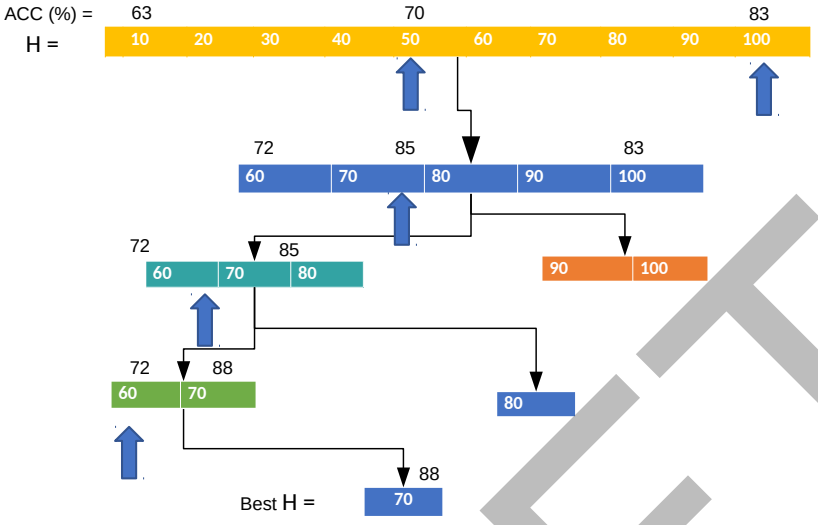

Figure 2: Example of the DC-ELM operation.

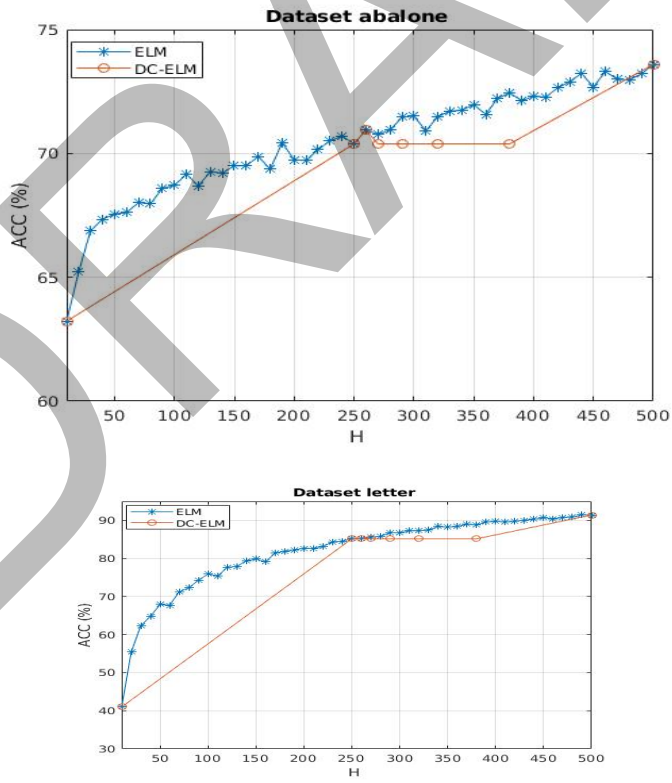

Figure 2: Accuracy vs. H during ELM and DC-ELM training for datasets `(a) abalone` and `(b) letter`.

downloaded from GitHub site [1]. Table 1 in the supplementary material reports the list of 27 datasets used for the experimental work. The experiments were executed in Matlab R2018 on a computer equipped with 8

[1] https://github.com/wentaozhu/constrained-extreme-learning-machine

Intel Core i7-4790k processors at 4GHz, with 16 GB RAM under Ubuntu 18.04 operative system. Each dataset was splited into training, validation, and testing sets, using standard 4-fold cross-validation. The performance measurement was the accuracy (ACC) in %. Each algorithm used K=50 hidden neurons from H=10 to H=500 with step 10, so that $\mathcal{H} = \{10k\}_{k=1}^{K}$ and the number of trainings is K for ELM, that must be trained for all the H-values, and $K' \leq K$ for MA-ELM, EMA-ELM or DC-ELM, with K' depending on the required iterations. The percentage of reduction in time, denoted as β (in %), of our proposals with respect to ELM (or CELM) is $\beta = 100(T - T')/T$, where T and $T' \leq T$ are the times spent by our proposals and by ELM, respectively.

9 Dataset List

We used a collection of 27 classification benchmark datasets selected from the UCI Machine Learning Repository [2], whose specifications (numbers N and I of patterns and inputs, respectively) are listed in Table 2.

10 Comparison of MA-ELM with ELM, CELM and MELM

Table 3 reports the accuracies achieved by MA-ELM and ELM. This table also reports the percentage of reduction in the number K of hidden nodes tried, defined as $\alpha(\%) = \frac{100(K-K')}{K}$. The proposed method EM-ELM achieves a time reduction α and β up to 94.9% and 94%, respectively, with respect to the original ELM. The ratio between the times of ELM and MA-ELM is 9.11, which means that MA-ELM is on average nine times faster than the original ELM achieving the same average accuracy (79.9%). For instance, in the `seeds` dataset, α=94.4% and MA-ELM achieves more accuracy (95.3%) than ELM (93.9%), while the times are 0.12 and 2.15 s., respectively, with a time reduction percentage β=96.2%. From eq. 5 in the paper, we have $T/T' = 100/(100 - \beta)$, so that a value β=98%, such as in dataset `wine`, means that ELM is about 50 times slower than MA-ELM, because $T/T' = 100/(100 - 98) = 100/2 = 50$.

We also combined the moving average to the constrained, constrained sum and constrained difference ELM (CELM, CSELM and DELM, respectively), that provide alternatives to randomly select numbers of hidden neurons [31]. Table 4 reports the results of CELM, CSELM and DELM compared to their MA- counterparts MA-CELM, MA-DSELM and MA-DELM, respectively. The average accuracy of MA-CELM (80.8%) outperforms CELM (80.3%) with average β up to 92.1%. Analogously, MA-CSELM and MA-DELM achieve 81.1% and 80.5% outperforming CSELM and DELM (81% and 80.4%, respectively), with average β up to 83.8% and 91.7%. Moreover, in `mammograph` dataset the CELM achieved ACC=66.3% with β=93.7%, while EMA-CELM achieved ACC=78.6%. With respect to MA-CSELM, in the `australian` dataset achieved ACC=84.9%, were CSELM achieved ACC=76.5% with β=90.4%.

The MA was also combined with mixed, sum and random sum ELM (MELM, RSELM and SELM, respectively [31]), and the comparison to the original methods (i.e., without MA) is reported in Table 5. The MA-SELM achieved an average ACC=80.5% outperforming SELM (ACC=79.9%) with a high average β=94.3%. Besides, MA-RSELM achieved avg. ACC=80.4% while RSELM achieved 80.2% with β up to 94.8%. Moreover, in MA-MELM the avg. ACC=81.0% while MELM achieved 80.6% with average β=87.5%.

11 Comparison of EMA-ELM with ELM, CELM and MELM

In this subsection the proposed EMA-ELM is compared with original ELM. As in the case of moving average, the conducted experiments were performed by randomly choosing the window width n randomly in the set $\{2, 3, 4\}$. The results showed a significant improvement in the percentage of reduction in the number of hidden sizes tried (α), in the time (T') and in the percentage of reduction in time (β), while achieving similar or higher accuracy, as reported in Table 6. The EMA-ELM achieved slightly better ACC=80.0% than the original ELM (79.9%) and with β ranging between 60% and 96%. The time ratio (bottom of the table) reports that EMA-ELM is in average 7.4 faster than than ELM. In addition, EMA-ELM proved that hidden size do not have to be the largest in order to achieve better performance in some case. For example, in the `australian` dataset achieved in original ELM best H=500 with ACC=76.2% and T=10.78 sec., where the EMA-ELM selected H=70 with ACC=85.4% and T'=1.19 s., α=89% and β=86%. In the the `synthetic` dataset, the original ELM achieved ACC=94.8% and T=11.76 s. with H=170 hidden neurons, while EMA-ELM achieved better ACC=95.3% with H=120, T'=2.62 s., α=77.7% and β=76%.

Analogously to MA-ELM, Table 7 reports the comparison of EMA-CELM, EMA-CSELM and EMA-DELM with CELM, CSELM and DELM, respectively. These results prove that EMA-CELM achieves average β=95.1% and average ACC=80.4%, while CELM achives avg. ACC=80.3%. The comparison of EMA-CSELM and CSELM leads to average β=92.6% and avg. ACC=81.2% for EMA-CSELM outperforming CSELM (81.0%). Moreover, in the `monks2` dataset the DELM achieves ACC=53.9%, while EMA-DELM achieves ACC=55.3% with β=79.5%. In average over all the datasets, EMA-DELM achieves β=96%. Note that the percentage β of reduction in time (1.8%) is very low in dataset `chess` because the condition

[2] https://archive.ics.uci.edu (Visited May, 2021)

Table 2: Collection of UCI classification datasets.

Original Name	Dataset	N	I
Abalone	abalone	4,177	8
MicroMass	msa	931	1300
Australian Sign Language signs	australian	6,650	15
MONK's Problems	monks2	432	7
Chess(King-Rook vs. King)	chess	18,056	6
Nursery	nursery	12,960	8
Congressional Voting Records	voting	435	16
Pima Indians Diabetes data	pima	768	8
Connect-4	connect-4	67,557	42
Planning Relax	planning	182	13
Connectionist Bench	sonar	208	60
Seeds	seeds	210	7
Energy efficiency	energy-heat	768	8
Shuttle Landing Control	shuttle	57,977	6
Hepatitis C Virus (HCV)	hepatitis	1,385	29
South German Credit	german	1,000	21
Image Segmentation	imseg	2,310	19
SPECT Heart	heart	267	22
Ionosphere	ionosphere	351	34
Statlog (Vehicle Silhouettes)	vehicle	946	18
Letter Recognition	letter	20,000	16
Synthetic Control Chart	synthetic	600	60
MAGICGamma Telescope	magic	19,020	11
Tic-Tac-Toe Endgame	tictac	958	9
MammographicMass	mammograph	961	6
Wine	wine	178	13
MiniBooNE	miniboone	130065	50

Table 3: Comparison of MA-ELM and ELM.

	ELM			MA-ELM					
Dataset	ACC	H	T(s)	ACC	H	T'(s)	M	α(%)	β(%)
abalone	**66.2**	500	29.27	64.6	60	2.41	6	91.8	88
australian	76.2	500	10.78	**84.9**	40	0.8	4	92.6	92
chess	**42.4**	490	304.6	30.7	100	243.75	10	20.0	80
connect-4	**75.5**	500	1296.25	68.1	30	119.37	3	90.8	94
energy-heat	**92.8**	390	7.83	88.7	90	0.82	9	89.5	82
german	65.3	240	6.69	**71.9**	110	1.04	11	84.5	78
heart	76.7	160	3.84	**80.1**	40	0.4	4	89.6	92
hepatitis	78.7	90	2.27	**81.3**	70	0.4	7	82.4	86
imseg	**94.7**	460	19.36	89.9	110	2.85	11	85.3	78
ionosphere	83.7	200	5.66	**88.0**	120	0.89	12	84.3	76
letter	**77.0**	490	157.68	71.9	200	51.54	20	67.3	60
magic	**84.7**	500	130.87	79.3	60	13.8	6	89.5	88
mammograph	68.4	370	6.99	**77.2**	40	0.51	4	92.7	92
miniboone	**90.7**	490	1799.25	86.3	60	228.57	6	87.3	88
monks2	52.9	280	5.07	**63.4**	160	1.34	16	73.6	68
msa	**98.8**	30	7.41	**98.8**	30	0.55	3	92.6	94
nursery	**92.6**	470	108.55	88.3	50	10.8	5	90.1	90
pima	66.8	500	7.53	**76.0**	50	0.55	5	92.7	90
planning	**52.4**	130	2.13	**52.4**	130	0.49	13	77.0	74
seeds	93.9	80	2.15	**95.3**	20	0.12	2	94.4	96
shuttle	**96.8**	410	406.30	90.5	80	41.31	8	89.8	84
sonar	**72.6**	130	4.35	72.2	90	0.66	9	84.8	82
synthetic	**94.8**	170	11.76	93.2	80	1.73	8	85.3	84
tictac	**97.8**	320	10.58	95.4	80	1.47	8	86.1	84
vehicle	**80.1**	480	9.55	79.3	130	1.55	13	83.8	74
voting	90.9	150	5.35	**93.3**	50	0.51	5	90.5	90
wine	**95.0**	10	2.34	**95.0**	10	0.12	2	94.9	98
Avg.	**79.9**			**79.9**		9.11		84.6	96.2

$|A'_k - A_k| < \tau$ on algorithm 2 is rarely fulfilled, so the ELM must be trained for almost all the H values.

The results in Table 8 compare SELM, CELM and CSELM to their versions combined with EMA. The EMA-SELM achieved average β=93.1% with ACC=80.2% outperforming SELM (79.9%). As well, in `monks2` dataset SELM achieved ACC=49% while EMA-SELM achieves 54.8% with β=71.7%.

Table 4: Accuracy and β of CELM, CSELM and DELM with and without MA.

	CELM	MA-CELM		CSELM	MA-CSELM		DELM	MA-DELM	
Dataset	ACC	ACC	β	ACC	ACC	β	ACC	ACC	β
abalone	**66.6**	64.7	92.8	**67.1**	65.4	87.8	**66.5**	64.3	94.4
australian	78.1	**85.4**	94.7	76.5	**84.9**	90.4	77.1	**85.5**	93.5
chess	**43.4**	30.7	87.2	**49.2**	47.8	11.1	**43.2**	31.2	86.1
connect-4	**76.7**	73.8	85.9	**77.4**	73.3	82.1	**76.6**	73.9	86.1
energy-heat	**94.4**	88.1	94.6	**93.7**	92.7	83.1	**94.3**	88.1	93.4
german	60.3	**71.6**	91.3	64.8	**69.2**	84.4	65.9	**70.0**	92.3
heart	79.0	**82.0**	94.7	73.4	**78.6**	89.7	76.3	**81.5**	92.6
hepatitis	79.3	**86.5**	94.2	**81.3**	**81.3**	85.0	76.1	**82.6**	91.6
imseg	**95.4**	87.8	94.3	**95.0**	90.5	86.9	**95.4**	88.0	93.2
ionosphere	**85.7**	83.7	94.6	82.3	**86.0**	83.6	84.3	**84.5**	93.1
letter	**75.7**	65.2	85.8	**74.2**	69.7	71.3	**75.7**	65.8	84.3
magic	**84.5**	80.0	94.8	**84.8**	80.8	90.1	**84.5**	80.3	93.5
mammograph	66.3	**78.6**	93.7	73.3	**77.4**	84.7	74.1	**76.9**	94.9
miniboone	**90.3**	86.8	91.8	**90.5**	83.1	90.6	**90.3**	84.9	93.3
monks2	52.2	**60.2**	92.8	49.8	**59.7**	89.1	53.9	**64.4**	94.1
msa	99.4	**100**	94.9	**99.5**	**99.5**	93.7	99.7	**100**	95.4
nursery	**94.3**	88.9	94.0	**95.5**	90.5	86.3	**94.3**	89.3	93.0
pima	69.3	**76.2**	95.2	66.8	**76.6**	94.4	66.4	**75.3**	94.3
planning	50.5	**64.7**	88.4	**54.5**	**54.5**	78.5	49.3	**62.5**	84.8
seeds	92.9	**93.9**	93.9	93.4	**93.8**	93.1	93.9	**94.8**	94.6
shuttle	**98.6**	94.3	94.0	**97.9**	91.5	88.7	**98.3**	93.5	95.2
sonar	75.2	**75.9**	86.8	**81.4**	**81.4**	80.3	**76.1**	75.4	89.5
synthetic	**97.2**	95.3	88.8	**96.7**	95.3	88.0	**97.5**	94.0	90.5
tictac	**98.5**	97.5	89.9	**98.5**	97.6	85.0	**98.8**	97.9	89.2
vehicle	**80.0**	76.2	87.6	**80.8**	77.3	85.7	**80.5**	75.3	87.8
voting	85.8	**94.2**	94.5	92.7	**94.5**	90.4	84.5	**94.5**	94.4
wine	97.9	**98.6**	95.1	**97.2**	**97.2**	93.6	**98.6**	**98.6**	94.0
Avg.	80.3	**80.8**	92.1	81.0	**81.1**	83.8	80.4	**80.5**	91.7

Table 5: Accuracy and β of MELM, RSELM and SELM with and without MA.

	MELM	MA-MELM		RSELM	MA-RSELM		SELM	MA-SELM	
Dataset	ACC	ACC	β	ACC	ACC	β	ACC	ACC	β
abalone	**66.9**	65.2	90.6	**66.4**	64.2	95.4	**66.3**	66.0	90.8
australian	74.3	**85.5**	91.2	73.9	**85.7**	96.2	77.2	**85.7**	89.4
chess	**46.1**	36.6	68.4	**43.3**	30.7	87.7	**49.3**	48.6	5.2
connect-4	**77.2**	73.3	80.8	**76.9**	74.2	86.3	**77.5**	72.7	82.8
energy-heat	**94.1**	88.7	86.1	**94.9**	84.4	94.4	**95.2**	89.1	82.9
german	62.0	**69.4**	89.1	58.7	**70.5**	96.2	62.0	**67.2**	84.5
heart	74.1	**84.1**	94.2	77.8	**82.6**	98.1	71.9	**79.7**	82.7
hepatitis	85.2	**85.2**	87.7	80.6	**86.5**	96.5	**83.3**	**83.3**	81.2
imseg	**95.4**	90.9	84.0	**95.5**	89.0	94.0	**94.9**	90.2	82.3
ionosphere	82.5	**86.4**	89.1	84.6	**85.3**	95.0	85.1	**85.3**	76.4
letter	**75.4**	71.0	69.3	**71.9**	63.7	86.5	**74.0**	70.7	62.7
magic	**84.6**	81.1	90.7	**84.0**	77.9	93.7	**84.8**	81.0	88.4
mammograph	71.0	**78.6**	90.4	68.2	**77.1**	97.0	69.3	**77.1**	93.1
miniboone	**90.5**	87.5	90.7	**89.9**	83.0	93.5	**90.5**	88.1	87.3
monks2	46.5	**62.0**	85.6	53.0	**65.1**	98.1	49.0	**61.1**	86.6
msa	**98.7**	**98.7**	94.3	99.7	**99.8**	97.4	**100**	**100**	93.1
nursery	**95.4**	89.7	91.4	**94.2**	89.2	94.5	**95.2**	89.3	91.5
pima	68.5	**75.5**	93.9	68.4	**75.7**	97.2	67.3	**77.3**	80.1
planning	**48.8**	**48.8**	77.5	59.3	**68.1**	98.0	**56.7**	**56.7**	75.0
seeds	**93.9**	93.8	91.7	93.8	**95.8**	98.4	93.8	**94.3**	91.7
shuttle	**98.7**	94.2	93.1	**99.0**	91.1	95.1	**97.9**	91.6	88.8
sonar	78.0	**78.4**	78.5	**74.6**	74.4	97.1	**80.4**	**80.4**	78.8
synthetic	**97.2**	96.0	88.0	**94.2**	90.0	91.6	**96.2**	94.5	88.0
tictac	**98.6**	97.3	86.8	**98.5**	98.3	96.3	98.3	**98.4**	83.0
vehicle	**79.5**	**78.2**	82.3	**79.6**	75.3	94.3	**80.6**	76.4	86.2
voting	**94.2**	93.9	92.2	86.0	**94.2**	97.7	—	—	—
wine	**97.8**	**97.8**	93.5	**98.6**	**98.6**	98.4	**98.6**	**98.6**	93.2
Avg	80.6	**81.0**	87.5	80.2	**80.4**	94.8	79.9	**80.5**	94.3

Table 6: Comparison of ELM and EMA-ELM.

	ELM			EMA-ELM					
Dataset	ACC	H	T(s)	ACC	H	T'(s)	M	α(%)	β(%)
abalone	**66.2**	500	29.27	64.6	70	3.52	7	88.0	86
australian	76.2	500	10.78	**85.4**	70	1.19	7	89.0	86
chess	**42.4**	490	304.6	34.9	200	118.69	20	61.0	60
connect-4	**75.5**	500	1296.25	68.1	30	119.69	3	90.8	94
energy-heat	**92.8**	390	7.83	88.7	90	0.89	9	88.6	82
german	65.3	240	6.69	**68.3**	140	1.7	14	74.6	72
heart	**76.7**	160	3.84	**76.7**	160	1.06	16	72.4	68
hepatitis	**78.7**	90	2.27	**78.7**	90	0.45	9	80.2	82
imseg	**94.7**	460	19.36	91.8	140	4.29	14	77.8	72
ionosphere	83.7	200	5.66	**84.5**	110	0.98	11	82.7	78
letter	**77.0**	490	157.68	68.5	150	40.16	15	74.5	70
magic	**84.6**	500	130.87	81.2	90	19.01	9	85.5	82
mammograph	68.4	370	6.99	**78.5**	170	1.73	17	75.3	66
miniboone	**90.7**	490	1799.25	88.5	100	359.54	10	80.0	80
monks2	52.9	280	5.07	**64.1**	20	0.21	2	95.9	96
msa	**98.8**	30	7.41	**98.8**	30	0.7	3	90.6	94
nursery	**92.6**	470	108.55	89.2	100	18.55	10	82.9	80
pima	66.8	500	7.53	**76.8**	90	1.07	9	85.8	82
planning	**52.4**	130	2.13	**52.4**	130	0.79	13	62.9	74
seeds	93.9	80	2.15	**95.3**	20	0.17	2	92.1	96
shuttle	96.8	410	406.30	**90.8**	90	57.57	9	85.8	82
sonar	**72.6**	130	4.35	**72.6**	130	1.26	13	71.0	74
synthetic	94.8	170	11.76	**95.3**	120	2.62	12	77.7	76
tictac	**97.8**	320	10.58	97.5	140	2.11	14	80.1	72
vehicle	**80.1**	480	9.55	79.6	150	2.24	15	76.5	70
voting	90.9	150	5.35	**93.3**	50	0.47	5	91.2	90
wine	**95.0**	10	2.34	**95.0**	10	0.12	1	94.9	98
Avg.	79.9			**80.0**		7.4		81.8	80.1

With respect to EMA-RSELM and RSELM, the average β reaches 96.7% with ACC=80.5% compared to 80.2% with RSELM. In the `voting` dataset the RSELM achieves ACC=86% while EMA-RSELM reaches 92.7% with β=85.6%. Moreover, comparing EMA-MELM and MELM the average β is 95.9%, while EMA-MELM achieves avg. ACC=81% and MELM ACC=80.6%. In the `pima` dataset the MELM recorded ACC=68.5% while EMA-MELM achieved 74.7% with β=95.9%. Again, β is very slow, in fact it is negative (-4.1%), in dataset `chess` because the accuracy is always calculated by training ELM and never estimated using EMA, so EMA-ELM is slightly slower than ELM.

12 Comparison of DC-ELM with ELM, CELM and MELM

Now, we will compare DC-ELM with the original ELM and CELMs. The divide-and-conquer method is used to reach the best-hidden size instead of training all hidden neurons in traditional ways. The ELM is only trained using the candidate numbers of hidden neurons, and ignoring the unsuitable numbers of hidden neurons. This reduces the number of training executions in DC-ELM compared to ELM. Table 9 reports the accuracy, best H and elapsed time (T) of the original ELM, alongside with the accuracy, best H, time (T'), number M of training executed, percentage α of reduction in M and percentage β of reduction in the training time of DC-ELM for each UCI benchmark classification dataset. The results show that the proposed method DC-ELM achieves α values between 75.4% and 84.2% and β between 82% and 84%. The DC-ELM slightly overcomes ELM in terms of ACC, with average values 80% and 79.9%, respectively. For instance, in the `miniboone` dataset, DC-ELM achieved T'=355.93 sec, while the original ELM achieved T= 1799.25 s. with an ACC (90.6%) close to ELM (90.7%), with α=80.2% and β=84%. Regarding the `shuttle` dataset, DC-ELM achiveds T'=85.04 s. and ACC=97.5% outperforming ELM (T=406.30 s., ACC=96.8%) with α=79.1% and β= 84%. In addition, in the `nursery` dataset, DC-ELM achieved T'=21.78 s. while the ELM spends T=108.55 s., whereas the ACC=92.7% in DC-ELM and 92.6% in ELM, with α=79.9% and β=84%.

The results also showed that there is no significant difference between ELM and DC-ELM in choosing the appropriate H. For example, in dataset `tictac`, the original ELM selected H=320 with T=10.58 s., which is the same H value selected by DC-ELM with lower T'=2.21 s., with α=79.1% and β=84%. In `german` dataset, the original ELM selected H=240 and spent T=6.69 s., which is the same H selected by DC-ELM with T'=1.51 s., α=77.4% and β=80%, and both methods achieved the same ACC=65.3%. This proves that DC-ELM has higher ACC and more speed in choosing the appropriate hidden neuron compared to the original ELM. The time ratio (last line in Table 9) between

Table 7: Accuracy and β of CELM, CSELM and DELM with and without EMA.

	CELM	EMA-CELM		CSELM	EMA-CSELM		DELM	EMA-DELM	
Dataset	ACC	ACC	β	ACC	ACC	β	ACC	ACC	β
abalone	**66.6**	65.6	84.5	**67.1**	65.6	82.0	**66.5**	65.1	88.1
australian	78.1	**85.2**	78.8	76.5	**84.1**	71.2	77.1	**85.2**	79.3
chess	**43.4**	41.9	9.5	**49.2**	**49.2**	1.1	**43.2**	**43.2**	1.8
connect-4	**76.7**	74.4	76.9	**77.4**	75.2	63.3	**76.6**	74.2	79.4
energy-heat	**94.4**	89.7	77.8	**93.7**	92.7	74.3	**94.3**	88.5	80.2
german	60.3	**64.8**	66.4	**64.8**	**64.8**	52.3	65.9	**67.5**	60.8
heart	**79.0**	**79.0**	72.1	**73.4**	**73.4**	69.5	**76.3**	**76.3**	69.5
hepatitis	**79.3**	**79.3**	80.8	**81.3**	**81.3**	78.5	**76.1**	**76.1**	82.8
imseg	**95.4**	92.2	76.9	95.0	**92.5**	79.4	**95.4**	92.0	77.0
ionosphere	**85.7**	85.4	81.1	**82.3**	**82.3**	73.8	84.3	**86.0**	76.4
letter	**75.7**	73.6	34.5	**74.2**	72.9	29.0	**75.7**	73.1	45.3
magic	**84.5**	81.4	85.5	**84.8**	81.8	86.2	**84.5**	81.1	88.6
mammograph	66.3	**78.3**	84.2	73.3	**75.7**	59.8	74.1	**76.9**	95.6
miniboone	**90.3**	87.6	85.2	**90.5**	88.5	79.6	**90.3**	87.1	86.0
monks2	**52.2**	49.5	72.2	49.8	**56.9**	81.0	53.9	**55.3**	79.5
msa	**99.4**	**99.4**	93.0	**99.5**	**99.5**	89.4	**99.7**	**99.7**	93.8
nursery	**94.3**	88.9	87.0	**95.5**	93.5	66.8	**94.3**	89.4	87.8
pima	69.3	**75.8**	95.1	66.8	**72.9**	55.5	66.4	**76.8**	89.3
planning	**50.5**	**50.5**	70.9	**54.5**	**54.5**	69.5	**49.3**	**49.3**	74.3
seeds	92.9	**93.3**	90.2	**93.4**	**93.4**	86.9	**93.9**	92.9	90.7
shuttle	**98.6**	94.8	91.2	**97.9**	92.1	83.3	**98.3**	95.6	91.7
sonar	**75.2**	**75.2**	73.8	**81.4**	**81.4**	73.4	**76.1**	**76.1**	73.6
synthetic	**97.2**	95.3	85.1	**96.7**	96.0	82.3	**97.5**	95.8	84.0
tictac	**98.5**	98.0	84.7	**98.5**	98.3	75.0	**98.8**	98.0	85.6
vehicle	80.0	**80.5**	63.4	80.8	**82.0**	68.8	80.5	**80.5**	64.1
voting	85.8	**94.2**	85.2	92.7	**94.5**	85.8	84.5	**93.9**	78.8
wine	**97.9**	**97.9**	92.8	**97.2**	**97.2**	92.6	**98.6**	**98.6**	93.2
Avg.	80.3	**80.4**	95.1	81.0	**81.2**	92.6	80.4	**80.5**	96.0

Table 8: Accuracy and β of MELM, RSELM and SELM with and without EMA.

	MELM	EMA-MELM		RSELM	EMA-RSELM		SELM	EMA-SELM	
Dataset	ACC	ACC	β	ACC	ACC	β	ACC	ACC	β
abalone	**66.9**	65.6	82.3	**66.4**	65.0	87.4	**66.3**	65.0	87.2
australian	74.3	**84.9**	94.5	73.9	**83.5**	78.4	77.2	**85.7**	87.0
chess	**46.1**	**46.1**	-5.6	**43.3**	**43.3**	-0.4	**49.3**	**49.3**	-4.1
connect-4	**77.2**	74.1	77.3	**76.9**	74.8	74.5	**77.5**	74.1	75.6
energy-heat	**94.1**	86.5	95.4	**94.9**	90.9	86.7	**95.2**	91.1	76.8
german	62.0	**70.8**	86.3	**58.7**	**58.7**	64.2	62.0	**64.8**	55.3
heart	74.1	**78.2**	81.1	**77.8**	**77.8**	82.3	71.9	**74.5**	66.3
hepatitis	**85.2**	**85.2**	80.2	**80.6**	**80.6**	92.7	83.3	**83.9**	92.4
imseg	**95.4**	91.5	77.0	**95.5**	92.5	79.4	**94.9**	91.6	77.2
ionosphere	82.5	**83.0**	76.1	84.6	**85.4**	85.9	85.1	**85.3**	74.2
letter	**75.4**	72.7	42.6	**71.9**	71.8	00.3	**74.0**	71.1	45.2
magic	**84.6**	81.8	85.2	**84.0**	80.6	82.1	**84.8**	81.0	87.9
mammograph	71.0	**78.6**	90.4	68.2	**77.4**	73.7	69.3	**77.1**	93.1
miniboone	**90.5**	87.7	84.5	**89.9**	86.7	76.3	**90.5**	87.6	84.0
monks2	46.5	**54.6**	79.3	**53.0**	**53.0**	53.8	49.0	**54.8**	71.7
msa	**98.7**	**98.7**	93.1	**99.7**	**99.7**	93.0	**100**	**100**	92.2
nursery	**95.4**	90.7	82.6	**94.2**	89.5	81.1	95.2	**93.0**	71.1
pima	68.5	**74.7**	95.9	68.4	**75.7**	59.5	67.3	**75.5**	86.8
planning	**48.8**	**48.8**	72.8	**59.3**	**59.3**	87.9	**56.7**	**56.7**	66.2
seeds	**93.9**	93.8	89.8	**93.8**	**93.8**	95.1	93.8	**94.3**	88.6
shuttle	**98.7**	93.6	90.8	**99.0**	93.0	87.8	**97.9**	91.3	87.2
sonar	**78.0**	**78.0**	74.9	**74.6**	**74.6**	87.6	**80.4**	**80.4**	73.8
synthetic	**97.2**	94.7	86.7	**94.2**	92.7	85.5	**96.2**	95.8	83.9
tictac	**98.6**	98.5	79.2	**98.5**	98.3	89.0	**98.3**	97.8	80.7
vehicle	79.5	**80.9**	68.4	79.6	**83.0**	54.2	**80.6**	76.4	85.1
voting	94.2	**94.5**	84.0	86.0	**92.7**	85.6	—	—	—
wine	**97.8**	**97.8**	91.6	**98.6**	**98.6**	97.0	**98.6**	**98.6**	90.5
Avg.	80.6	**81.0**	95.9	80.2	**80.5**	96.7	79.9	**80.2**	93.1

Table 9: Comparison between DC-ELM and ELM.

	ELM			DC-ELM					
Dataset	ACC	H	T(s)	ACC	H	T'(s)	M	α(%)	β(%)
abalone	**66.2**	500	29.27	**66.2**	500	6.22	8	78.8	84
australian	**76.2**	500	10.78	**76.2**	500	2.21	8	79.5	84
chess	**42.4**	490	304.6	42.1	500	59.68	8	80.4	84
connect-4	**75.5**	500	1296.25	**75.5**	500	268.51	8	79.3	84
energy-heat	92.8	390	7.83	**93.0**	500	1.72	8	78.0	84
german	**65.3**	240	6.69	**65.3**	240	1.51	10	77.4	80
heart	**76.7**	160	3.84	**76.7**	160	0.69	8	82.0	84
hepatitis	**78.7**	90	2.27	**78.7**	90	0.45	8	80.2	84
imseg	94.7	460	19.36	**94.9**	500	4.06	8	79.0	84
ionosphere	83.7	200	5.66	**84.9**	210	1.15	9	79.7	82
letter	**77.0**	490	157.68	**77.0**	500	32.23	8	79.6	84
magic	**84.7**	500	130.87	**84.7**	500	29.36	9	77.6	82
mammograph	**68.4**	370	6.99	63.2	500	1.72	8	75.7	84
miniboone	**90.7**	490	1799.25	90.6	500	355.93	8	80.2	84
monks2	**52.9**	280	5.07	56.3	320	1.12	8	77.9	84
msa	98.8	30	7.41	**99.9**	40	1.17	8	84.2	84
nursery	92.6	470	108.55	**92.7**	500	21.78	8	79.9	84
pima	**66.8**	500	7.53	**66.8**	500	1.69	8	77.6	84
planning	**52.4**	130	2.13	**52.7**	130	0.45	9	78.9	82
seeds	**93.9**	80	2.15	**93.9**	90	0.45	8	79.1	84
shuttle	96.8	410	406.30	**97.5**	500	85.04	8	79.1	84
sonar	**72.6**	130	4.35	**72.6**	130	0.87	9	80.0	82
synthetic	**94.8**	170	11.76	**94.8**	210	2.33	9	80.2	82
tictac	**97.8**	320	10.58	**97.8**	320	2.21	8	79.1	84
vehicle	**80.1**	480	9.55	79.8	500	2.18	8	77.2	84
voting	90.9	150	5.35	**91.0**	180	1.06	9	80.2	82
wine	95.0	10	2.34	**96.5**	40	0.44	8	81.2	84
Avg.	79.9			**80.0**		4.9		79.3	83.4

Table 10: Accuracy and β of CELM, CSELM and DELM with and without DC.

	CELM	DC-CELM		CSELM	DC-CSELM		DELM	DC-DELM	
Dataset	ACC	ACC	β	ACC	ACC	β	ACC	ACC	β
abalone	**66.6**	**66.6**	75.0	**67.1**	66.4	78.3	66.5	**67.1**	80.4
australian	**78.1**	76.4	79.2	76.5	**79.4**	79.5	77.1	**78.4**	78.2
chess	**43.4**	**43.4**	80.0	**49.2**	49.1	80.7	43.2	**49.2**	81.0
connect-4	**76.7**	**76.7**	80.6	77.4	**77.5**	81.6	76.6	**77.4**	81.5
energy-heat	**94.4**	94.2	77.1	93.7	**94.4**	78.6	**94.3**	93.9	78.2
german	**60.3**	57.6	77.7	**64.8**	61.5	79.5	**65.9**	61.8	80.3
heart	**79.0**	**79.0**	80.3	**73.4**	68.9	77.9	**76.3**	73.4	80.6
hepatitis	**79.3**	**79.3**	79.1	**81.3**	80.6	81.8	76.1	**81.3**	82.8
imseg	95.4	**95.8**	78.0	95.0	**95.4**	80.9	**95.4**	95.0	78.1
ionosphere	**85.7**	**85.7**	81.5	**82.3**	81.7	81.8	**84.3**	82.3	80.7
letter	**75.7**	**75.7**	79.6	**74.2**	74.0	81.0	**75.7**	74.2	79.8
magic	**84.5**	**84.5**	79.7	84.8	**84.8**	78.0	84.5	**84.8**	78.2
mammograph	66.3	**67.9**	74.9	**73.3**	69.4	77.2	74.1	**77.1**	80.2
miniboone	**90.3**	**90.3**	80.0	90.5	**90.5**	80.6	90.3	**90.5**	81.1
monks2	**52.2**	**52.2**	76.3	49.8	**50.7**	76.9	**53.9**	47.5	78.0
msa	99.4	**100**	83.9	99.5	**100**	85.4	99.7	**100**	86.5
nursery	**94.3**	**94.3**	79.3	95.5	**95.7**	80.1	94.3	**95.6**	80.8
pima	**69.3**	**69.3**	76.3	66.8	**67.8**	77.8	66.4	**67.7**	76.6
planning	50.5	**51.6**	77.5	**54.5**	50.5	79.3	49.3	**54.5**	79.3
seeds	92.9	**93.3**	80.7	93.4	**95.8**	83.0	93.9	**94.8**	80.5
shuttle	98.6	**98.8**	78.6	97.9	**98.1**	78.9	**98.3**	98.1	80.3
sonar	**75.2**	**75.2**	79.2	**81.4**	79.8	81.2	76.1	**81.4**	81.2
synthetic	**97.2**	**97.2**	79.5	**96.7**	96.0	79.9	**97.5**	96.7	78.4
tictac	**98.5**	**98.5**	78.5	**98.5**	98.4	80.5	**98.8**	98.5	80.4
vehicle	**80.0**	78.9	75.8	**80.8**	77.7	76.9	**80.5**	79.7	75.1
voting	85.8	**86.9**	75.2	92.7	**93.9**	82.2	84.5	**92.7**	82.3
wine	97.9	**98.6**	80.8	97.2	**98.6**	83.8	98.6	**99.3**	83.1
Avg.	**80.3**	**80.3**	78.7	**81.0**	80.6	80.1	80.4	**81.2**	80.2

Table 11: Accuracy and β of MELM, RSELM and SELM with and without DC.

	MELM	DC-MELM		RSELM	DC-RSELM		SELM	DC-SELM	
Dataset	ACC	ACC	β	ACC	ACC	β	ACC	ACC	β
abalone	**66.9**	**66.9**	76.6	**66.4**	**66.4**	78.6	**66.3**	66.1	76.8
australian	**74.3**	**74.3**	78.5	**73.9**	**73.9**	77.8	77.2	**77.7**	77.0
chess	**46.1**	**46.1**	79.8	**43.3**	**43.3**	80.1	**49.3**	**49.3**	80.2
connect-4	**77.2**	**77.2**	80.6	**76.9**	**76.9**	81.1	**77.5**	**77.5**	80.0
energy-heat	**94.1**	93.9	76.9	**94.9**	**94.9**	78.6	**95.2**	93.6	76.3
german	62.0	**63.3**	78.2	**58.7**	58.1	79.6	**62.0**	59.0	77.1
heart	**74.1**	70.4	78.1	**77.8**	**77.8**	85.1	**71.9**	68.2	76.2
hepatitis	**85.2**	**85.2**	81.0	**80.6**	**80.6**	88.6	**83.3**	81.3	78.0
imseg	**95.4**	**95.4**	78.5	95.5	**95.6**	75.7	94.9	**95.4**	78.2
ionosphere	**82.5**	81.7	78.6	**84.6**	**84.6**	84.4	**85.1**	**85.1**	78.8
letter	**75.4**	**75.4**	79.6	**71.9**	**71.9**	79.7	**74.0**	**74.0**	79.7
magic	**84.6**	**84.6**	79.2	**84.0**	**84.0**	79.3	**84.8**	**84.8**	79.4
mammograph	**71.0**	**71.0**	77.0	**68.2**	67.9	76.8	69.3	**72.9**	74.6
miniboone	**90.5**	**90.5**	81.2	**89.9**	**89.9**	80.3	**90.5**	**90.5**	79.7
monks2	46.5	**49.3**	76.8	**53.0**	**53.0**	76.4	49.0	**52.6**	75.5
msa	**98.7**	**98.7**	85.2	**100**	99.2	79.2	**100**	99.9	83.9
nursery	**95.4**	**95.4**	77.8	**94.2**	**94.2**	79.3	**95.2**	**95.2**	79.1
pima	**68.5**	**68.5**	73.4	**68.4**	**68.4**	76.9	67.3	**67.5**	77.0
planning	**48.8**	**48.8**	77.5	**59.3**	**59.3**	86.8	**56.7**	**56.7**	76.4
seeds	93.9	**94.8**	80.3	93.8	**95.2**	88.5	93.8	**93.9**	79.0
shuttle	98.7	**98.8**	80.4	99.0	**99.1**	78.9	97.9	**98.1**	76.2
sonar	**78.0**	**78.0**	80.1	**74.6**	**74.6**	86.7	**80.4**	76.4	78.1
synthetic	**97.2**	**97.2**	78.0	**94.2**	**94.2**	82.8	**96.2**	**96.2**	79.1
tictac	98.6	**98.8**	77.7	98.5	**98.6**	79.0	**98.4**	**98.4**	78.0
vehicle	79.5	**80.7**	76.7	**79.6**	**79.6**	77.3	**80.6**	78.7	76.6
voting	**94.2**	**94.2**	79.8	**86.0**	85.1	81.6	—	—	—
wine	97.8	**97.9**	81.3	**98.6**	**98.6**	89.3	**98.6**	**98.6**	83.7
Avg.	**80.6**	**80.6**	85.2	**80.2**	**80.2**	81.1	**79.9**	79.6	76.2

the original ELM and DC-ELM is $T/T' = 4.87$, which means that ELM is about five times slower than DC-ELM.

Similarly to MA-ELM and EMA-ELM, the proposed method DC-ELM was also combined with SELM, CELM and CSELM. The results are reported in Table 10, proving that DC-ELM achieves accuracy similar or higher than CSELMs with high β values. The DC-CELM achieves an average β=78.7%, with the same ACC=80.3% as CELM. Comparing CSELM and DC-CSELM, the percentage of training time reduction β ranges between 76.9% to 85.3%. Considering average values, the average ACC is 80.6% and 81% for DC-CSELM and CSELM, respectively. For instance, in `shuttle` dataset the CSELM achieved ACC=97.9% while DC-CSELM achieved ACC=98.1 with β=78.9%. The combination of DC-DELM and DELM, leaded to a β range between from 75.1% and 86.5%, with an average ACC=81.2% slightly higher than DC-DELM (80.4%). In `hepatitis` dataset DELM and DC-DELM achieve ACC=76.1% and 81.3%, respectively, while in `connect` $-$ `4` dataset, DC-ELM outperformed DELM with ACC=77.4% and 76.6%, respectively, with β=81.5%.

Table 11 reports the comparison of MELM, RSELM and SELM with and without DC. The DC-MELM (ACC=80.6%) equals MELM in terms of performance leading to a high average β up to 85.2%. The comparison of DC-RSELM and RSELM reports a β value ranging from 76.4% to 89.3%, also with the same average ACC=80.2%. Finally, the average ACC was 79.9% and 79.6% using SELM and DC-SELM, also with a high average β=76.2%.

13 Results and Discussion

Table 12 compares the average results of the proposed methods, MA-ELM, EMA-ELM and DC-ELM in terms of Acc and β to: 1) classical ELM, in column 3; 2) constrained CELM, constrained sum CSELM, and difference DELM, in columns 4-6; and 3) mixed MELM, random sum RSELM and sum SELM, in columns 7-9. These networks are alternatives existing in the literature to select the number of hidden neurons [31]. The values of Acc and β in Table 12 are averaged over the 27 datasets (section 8). The detailed comparison for each dataset is reported in Tables 2-10 (sections 5-7) of the supplementary material.

The third column of Table 12 compares MA-ELM, EMA-ELM and DC-ELM to the classical ELM. The MA, EMA and DC achieve equal or slightly higher accuracy of the ELM, but they largely accelerate the selection of H with time reductions (β) above 80% reaching 96.2% with MA. Thus, MA-ELM performed the H tuning in only 3.8% of the time spent by ELM, with similar or better accuracy.

Columns 4-6 report the comparison of CELM, CSELM and DELM to their corresponding versions using MA, EMA and DC. These versions perform better, specially MA-CELM that achieves 80.8% while

Table 12: Comparison of Acc and β, both in %, of MA, EMA and DC with ELM, CELM, CSELM and DELM, and with MELM, RSELM and SELM (in bold the best results for each column).

		ELM	CELM	CSELM	DELM	MELM	RSELM	SELM	Mean
	Acc	79.9	80.3	81	80.4	80.6	80.2	79.9	80.33
Acc	MA	79.9	**80.8**	81.1	80.5	**81**	80.4	79.9	80.51
	EMA	**80**	80.4	**81.2**	80.5	**81**	**80.5**	**80.2**	**80.54**
	DC	**80**	80.3	80.6	**81.2**	80.6	80.2	79.9	80.40
β	MA	**96.2**	82.1	83.8	91.7	87.5	94.8	**94.3**	90.06
	EMA	80.1	**95.1**	**92.6**	**96**	**95.9**	**96.7**	93.1	**92.78**
	DC	83.4	80.1	80.1	80.2	85.2	81.1	76.2	80.90

CELM achieves 80.3%. The difference is lower with CSELM (81.2% and 81.1% with EMA-CSELM and CSELM). However, DC-DELM outperforms DELM, 81.2% vs. 80.4%. In terms of β (lower part of columns 4-6), the time savings are again above 80%, being specially higher for EMA versions, that are above 92%.

Finally, columns 7-9 of Table 12 compare MELM, RSELM and SELM with their variants using MA, EMA and DC. The MA-MELM and EMA-MELM achieve 81% of accuracy, outperforming MELM (80.6%). Besides, EMA-RSELM outperforms RSELM (80.5% vs. 80.2%), and EMA-SELM outperforms SELM (80.2% vs. 79.9%). The EMA versions achieve the highest β with MELM and RSELM (above 95%), while and MA-SELM achieves the highest β (94.3%).

Comparing MA, EMA and DC, the last column reports the average values over columns 3-9, reporting that EMA versions achieve both the highest accuracy and time savings, about 92.78%, while MA and DC achieve only 90.06% and 80.9%, respectively. Therefore, EMA seems to be the fastest approach, although MA-ELM achieves the highest time saving (β=96.2%) with respect to ELM, being EMA-ELM and DC-ELM much slower (80-83%). Overall, MA-ELM, EMA-ELM and DC-ELM perform fairly well, achieving accuracy slightly above the classical ELM and the CELM, CSELM, SELM, MELM, RSELM and SELM networks, and accelerating very much the hyper-parameter tuning.

14 Conclusions

The number H of hidden nodes influences the ELM performance and speed, so great generalization requires an optimal H. We propose MA-ELM, EMA-ELM and DC-ELM, that use respectively moving-average, exponential-moving-average and divide-and-conquer to reduce the number of trainings required to select H. They achieve equal or slighly higher accuracy than ELM, constrained (CELM), constrained sum (CSELM) and constrained difference (DELM), mixed (MELM), sum (SELM) and random sum (RSELM). However, they reduce the time spent in training to trial different H values between 80% and 97%, depending on the dataset and the ELM version, specially with MA and EMA, that achieves the largest accuracy and time savings. The MA, EMA and DC-ELM may bring significant performance advantages, compared to existing alternatives, in applications where factors limit the number of hidden neurons, such a multiuser eye-tracking system [1] and other real-life applications.

References

[1] Al-Btoush, A., Abbadi, M., Hassanat, A., Tarawneh, A., Hasanat, A., and Prasath, S. New features for eye-tracking systems: Preliminary results. In *Intl Conf Inf Comm Syst* (2019), IEEE, pp. 179–184.

[2] Al-Btoush, A., Fernández-Delgado, M., Cernadas, E., and Barro, S. Extreme learning machine with confidence interval based bias initialization. In *2021 Second International Conference on Intelligent Data Science Technologies and Applications (IDSTA)* (2021), IEEE, pp. 23–30.

[3] Al Nawaiseh, A. J., Albtoush, A., Al Nawaiseh, S. J., et al. Evaluate database management system quality by analytic hierarchy process (ahp) and simple additive weighting (saw) methodolog. *MENDEL 28*, 2 (2022), 67–75.

[4] Albtoush, A., Fernández-Delgado, M., Cernadas, E., and Barro, S. Quick extreme learning machine for large-scale classification. *Neural Computing and Applications 34*, 8 (2022), 5923–5938.

[5] Cao, L., Yue, Y., Zhang, Y., and Cai, Y. Improved crow search algorithm optimized extreme learning machine based on classification algorithm and application. *IEEE Access 9* (2021), 20051–20066.

[6] Dani, Y., Gunawan, A. Y., Khodra, M. L., and Indratno, S. W. Detecting outliers using modified recursive pca algorithm for dynamic streaming data. *MENDEL 29*, 2 (2023), 237–244.

[7] Deng, W., Bai, Z., Huang, G., and Zheng, Q. A fast SVD-hidden-nodes based extreme learning machine for large-scale data analytics. *Neural Netw 77* (2016), 14–28.

[8] Elsheikh, A., Saba, A., Elaziz, M., Lu, S., et al. Deep learning-based forecasting model for

COVID-19 outbreak in Saudi Arabia. *Process Saf Environ 149* (2021), 223–233.

[9] Feng, G., Huang, G.-B., Lin, Q., and Gay, R. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE T Neur Netw 20*, 8 (2009), 1352–1357.

[10] Hsieh, C.-J., Si, S., and Dhillon, I. A divide-and-conquer solver for kernel support vector machines. In *Intl Conf Mach Learn* (2014), pp. 566–574.

[11] Huang, G.-B., and Chen, L. Convex incremental extreme learning machine. *Neurocomputing 70*, 16-18 (2007), 3056–3062.

[12] Huang, G.-B., and Chen, L. Enhanced random search based incremental extreme learning machine. *Neurocomputing 71*, 16-18 (2008), 3460–3468.

[13] Huang, G.-B., Zhu, Q.-Y., Mao, K., Siew, C.-K., Saratchandran, P., and Sundararajan, N. Can threshold networks be trained directly? *IEEE T Circuits Syst II: Express Briefs 53*, 3 (2006), 187–191.

[14] Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. Extreme learning machine: a new learning scheme of feedforward neural networks. In *IEEE Intl J Conf Neur Netw* (2004), vol. 2, pp. 985–990.

[15] Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. Extreme learning machine: theory and applications. *Neurocomputing 70*, 1-3 (2006), 489–501.

[16] Huang, Y., and Lai, D. Hidden node optimization for extreme learning machine. *AASRI Procedia 3* (2012), 375–380. Conf on Modelling, Identification and Control.

[17] Lai, J., Wang, X., Li, R., Song, Y., and Lei, L. BD-ELM: a regularized extreme learning machine using biased dropconnect and biased dropout. *Math Probl Eng 2020* (2020).

[18] Lan, Y., Soh, Y. C., and Huang, G.-B. A constructive enhancement for online sequential extreme learning machine. In *Intl J Conf Neural Netw* (2009), Ieee, pp. 1708–1713.

[19] Mahmoudi, M. R., and Baroumand, S. Modeling the stochastic mechanism of sensor using a hybrid method based on seasonal autoregressive integrated moving average time series and generalized estimating equations. *ISA Transactions* (2021), 300–305.

[20] Marques, N. C., and Gomes, C. Implementing an intelligent moving average with a neural network. In *Proc European Conf Artif Intel*. IOS Press, 2010, pp. 1129–1130.

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

[22] Rong, H.-J., Ong, Y.-S., Tan, A.-H., and Zhu, Z. A fast pruned-extreme learning machine for classification problem. *Neurocomputing 72*, 1-3 (2008), 359–366.

[23] Schuler, J. P. S., Romani, S., Abdel-Nasser, M., Rashwan, H., and Puig, D. Grouped pointwise convolutions reduce parameters in convolutional neural networks. *MENDEL 28*, 1 (2022), 23–31.

[24] Sheela, K. G., and Deepa, S. N. Review on methods to fix number of hidden neurons in neural networks. *Math Probl Eng 2013* (2013), 1–11.

[25] Similä, T., and Tikka, J. Multiresponse sparse regression with application to multidimensional scaling. In *Intl Conf Artif Neural Netw* (2005), Springer, pp. 97–102.

[26] Song, S., Wang, M., and Lin, Y. An improved algorithm for incremental extreme learning machine. *Syst Sci & Control Eng 8*, 1 (2020), 308–317.

[27] Suresh, S., Saraswathi, S., and Sundararajan, N. Performance enhancement of extreme learning machine for multi-category sparse data classification problems. *Engin Appl Artif Intel 23*, 7 (2010), 1149–1157.

[28] Uribe, I. M. Predictive model of the enso phenomenon based on regression trees. *MENDEL 29*, 1 (2023), 7–14.

[29] Yu, D., and Deng, L. Efficient and effective algorithms for training single-hidden-layer neural networks. *Pattern Recogn Lett 33*, 5 (2012), 554–558.

[30] Zhu, Q.-Y., Qin, A. K., Suganthan, P. N., and Huang, G.-B. Evolutionary extreme learning machine. *Patt Recogn 38*, 10 (2005), 1759–1763.

[31] Zhu, W., Miao, J., and Qing, L. Constrained extreme learning machines: A study on classification cases. arXiv:1501.06115, 2015.