International Journal of Machine Learning and Cybernetics (2019) 10:1543–1561
https://doi.org/10.1007/s13042-018-0833-6

ORIGINAL ARTICLE

Metaheuristic-based extreme learning machines: a review of design formulations and applications

Mohammed Eshtay[1] · **Hossam Faris**[1] · **Nadim Obeid**[1,2]

Received: 17 November 2017 / Accepted: 29 May 2018 / Published online: 8 June 2018

Abstract
Extreme learning machine (ELM) is a novel and recent machine learning algorithm which was first proposed by Huang et al. (Proceedings of 2004 IEEE international joint conference on, pp 985–990, 2004). Over the last decade, ELM has gained a remarkable research interest with tremendous audiences from different domains in a short period of time due to its impressive characteristics over other single hidden-layer feedforward neural networks. Although ELM enjoys powerful advantages, it still has some potential weaknesses like performance sensitivity to the initial condition of the input weights, number of hidden neurons, and the selection of activation functions. In order to overcome the limitations of classical ELM, many metaheuristic algorithms including the evolutionary algorithms, swarm intelligence, memetic and trajectory algorithms have been proposed for optimizing the different components of ELM by researchers aiming to improve the generalization performance of ELM networks for different types of complex problems and applications. Therefore our review paper intent to conduct a deep study of the important aspects of applying metaheuristic algorithms for optimizing ELM networks. Three main streams of research lines are identified: the optimization of input weights and hidden biases, selection of hidden neurons, and optimization of activation functions. Furthermore, this paper will discuss a wide spectrum of applications of metaheuristic-based ELM models. We will highlight the strengths of these models and the improvements that are suggested in the literature to overcome their weaknesses. We touch upon several interesting and challenging open issues in optimizing ELM using metaheuristics.

Keywords Extreme learning machine · ELM · Metaheuristic · Artificial Neural networks · ANN · Evolutionary Algorithm

1 Introduction

Artificial neural networks (ANNs) are information processing and mathematical models which are inspired by the biological neural systems. They have many advantages that make them widely applied by different classification and regression problems, such as high prediction power, ability to model dynamic and complex systems, ease of implementation, and their parallel nature [7].

ANNs consist of a number of processing elements called "neurons" that are distributed over a number of layers (i.e. input layer, output layer and zero or more hidden layers). In feedforward neural networks (FFNN), neurons are fully connected with the neurons in the proceeding layer. Single-hidden-layer feedforward neural networks (SLFN) are considered to be one of the most popular neural networks topologies. It was shown in the literature that SLFN are universal approximators which are able to approximate any continuous function [16, 40]. SLFNs are typically trained by gradient descent methods such as Backpropagation (BP) [82, 83]. In spite of their popularity, gradient descent based training algorithms such as BP, suffer major drawbacks such as high dependency of the initial weights of the network, high probability of being trapped in local minima and slow convergence [18, 25, 26, 30, 31, 56].

✉ Hossam Faris
hossam.faris@ju.edu.jo

Mohammed Eshtay
m.eshtay@fgs.ju.edu.jo

Nadim Obeid
nadim@ju.edu.jo

[1] King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan

[2] Department of Computer Science, King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan

To overcome the problems of classical feedforward neural networks with gradient descent training methods, ELM were first proposed by Huang et al. [48]. In ELM, the weights that connect the input layer with the hidden layer along with the hidden biases are randomly initialized, then the connection weights between the hidden layer and the output layer are analytically determined by finding a least-square solution using a simple method like the Moore–Penrose (MP) generalized inverse.

Unlike other training algorithms, ELM enjoys many advantages which makes it distinctive among its counterpart, for instance, ELM has a simple architecture which is easy to implement and apply. In addition to that, the learning speed of ELM is very fast compared to other learning algorithms such as BP; as training can be accomplished in seconds or minutes, which is significantly less than many other conventional learning methods. In general, ELM enjoys a high generalization performance, and it is possible to use a different activation functions [23]. Moreover, ELM can avoid many problems of traditional gradient bases algorithms such as learning rate and local minima. [20, 46, 49, 93, 98].

Despite its pros, ELM suffers from certain drawbacks: ELM performance is sensitive towards the initialization of the structure of the network. The initial settings of the weights and biases, and the number of neurons affect the performance of ELM. In addition, ELM needs more hidden neurons than the traditional tuning-based methods in many cases [15, 32].

Different approaches were proposed in the literature to solve these problems and to improve the performance of ELM networks. One type of these approaches that has gained a wide interest is the metaheuristic-based approach. Metaheuristic algorithms are efficient methods that are designed to provide acceptable or near optimal solutions for hard optimization problems. They actually guide the search process to efficiently explore the search space trying to fulfill this goal. Many of these algorithms are derived from biological or physical systems [11]. Some of the advantages of these algorithms are: problem independent, can stochastically guide the search process in order to find near optimal solutions, and can be used to solve problems ranging from simple search to complex problems [11, 12]. One year after the release of ELM, metaheuristic algorithms started to be intensively investigated and applied in designing and optimizing ELM network. Metaheuristic algorithms have shown significant improvement in the performance of ELM networks at both theoretical and empirical levels.

In literature, there are few published papers that surveyed ELM. Some of these papers concentrated on ELM variants, while other papers discussed ELM in general and targeted their applications. For example, one of the earliest surveys was conducted by Huang et al. [46], where they surveyed ELM and its theories, and discussed the variants of ELM. Another work was conducted by Ding et al. [19] in which they reviewed ELM variants and discussed various applications that were handled using ELM in the literature. In the aforementioned paper, the authors listed Evolutionary ELM among the ELM variants but without a detailed analysis. In another paper, Ding et al. [20] surveyed the latest research of ELM theory, algorithms and applications. They introduced three examples of Evolutionary ELMs. Huang et al. [41] gave some examples about the research that used Evolutionary ELM in other variants of ELMs. The paper reviewed ELM and its theories for classification and regression. Cao and Lin [13] surveyed ELM for high dimensional and large data applications. However, Evolutionary ELMs were not taken into consideration. Another review was presented in [1] in which the authors reviewed the advances of ELM and its applications without taking into consideration the Evolutionary ELM too. Interestingly, the analysis of the exiting surveys in the literature showed that either they briefly referred to the Evolutionary ELM or they do not refer to them at all. This finding was one of the main motivations for this study. Unlike the previous surveys and reviews of ELMs, the objective of this paper is to conduct a comprehensive review of the important aspects and design issues of metaheuristic-based ELM networks. Moreover, we identify the main research lines for this specific type of ELMs and their applications.

In this paper, the review refers to all of the previous research that have discussed the metaheuristic-based ELM models by referring to the prestigious publishers such as Elsevier, Springer, IEEE, and others. To show the expansion of the metaheuristic-based ELM models in the literature, Fig. 1 depicts the correspondence between the year and the number of publications that combine ELM with metaheuristics. We can observe that the number of publications per year was constant with only one publication for the first 6 years and then the number remarkably increased to reach its peak in 2013. Furthermore, the research has been very active in this field in the last 3 years. From another point of view, Fig. 2 lists the number of publications for each metaheuristic algorithm that has been used to optimize ELM. As it can be noticed, the most used algorithms are differential evolution (DE), particle swarm algorithm (PSO) and genetic algorithm (GA).

This paper is structured as follows: Sect. 2 gives a brief description of FFNN. Section 3 describes the classical ELM model. Section 4 reviews the metaheuristic algorithms and their classification. In Sect. 5, we identify and review the main approaches in the literature that utilize metaheuristic algorithms to optimize ELM networks. Section 6 lists most of the applications of metaheuristic-based ELMs. Finally, Sect. 7 concludes the paper and suggests some possible future research directions.

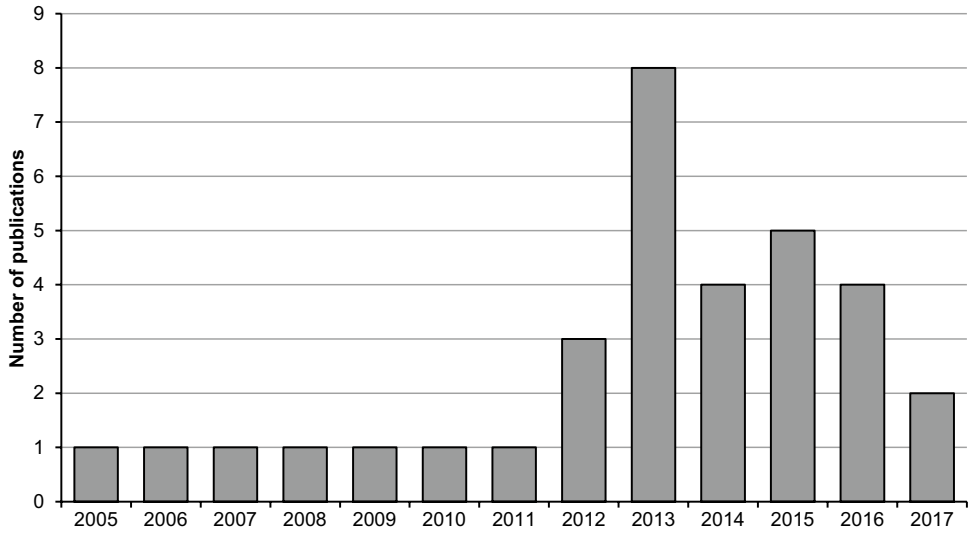

Fig. 1 Number of publications per year

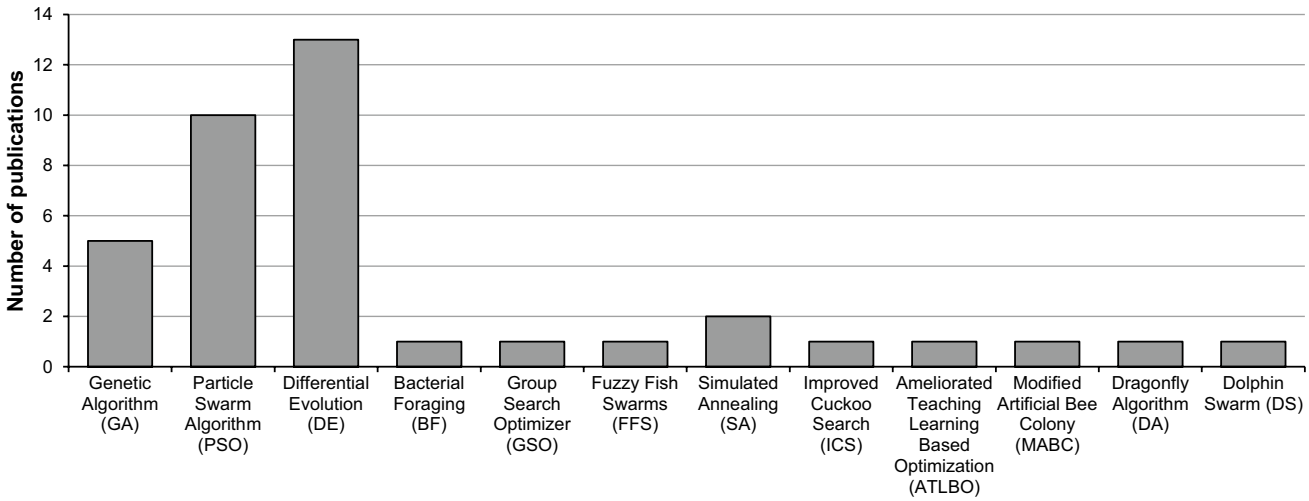

Fig. 2 Number of publications per each metaheuristic algorithm used in optimizing ELM networks

2 Feedforward neural networks

Artificial neural network (ANN) is an information processing system inspired by the biological neural network system in a human brain. The architecture of ANN consists of a set of neurons, weights, and layers. The simplest model of neural networks is the single layer perceptron (SLP) [81] which consists of input and output layers. SLP is a neural network that does not have any hidden layer, which makes it unable to approximate nonlinear continuous functions. In order to solve this problem, multilayer perceptron (MLP) [99] was proposed. In MLP, one layer of hidden neurons or more are added between input and output layers. MLP with single hidden layer and finite number of hidden neurons with any sigmoidal nonlinear activation function is capable of approximating any measurable function to any desirable degree of accuracy [40].

Feedforwad neural networks (FFNN) consist of a set of layers, each of which contains a number of neurons (nodes). The first layer receives the input and the last one produces the output. Each neuron in the layer is connected to all neurons of the next layer by a weight (forward link) as shown in Fig. 3. The neurons in the hidden and output layers are used to process the incoming information from the weighted links. The output of the neurons y_i can be calculated as given in the following equation:

$$y_i = \varphi_i\left(\sum_{k=1}^{n} w_k^i \times z_k^i + b_i\right), \tag{1}$$

where n is the number of features or number of input links, z^i are the inputs of the ith sample, w^i is the weight, b^i is the bias and φ_i is the activation function of the neuron i.

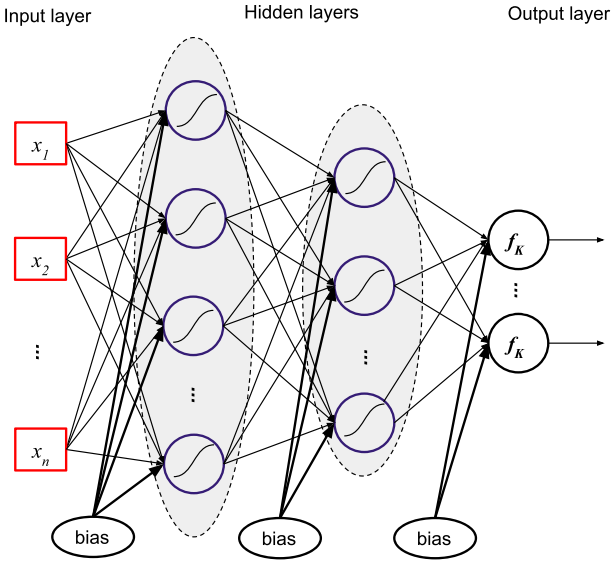

Fig. 3 Multilayer perceptron

The goal of using FFNN is to solve non-linear and complex problems. In order to perform this task at its best, we need to optimize the connection weights in the network using a training algorithm. FFNN is trained using a training dataset consists of input-output pairs (x,y) where $x = [x_1, x_2, \ldots, x_N]$ and $y = [y_1, y_2, \ldots, y_N]$. The input vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ has the target output $y_i = (y_{i1}, y_{i2}, \ldots, y_{im})$, after training, the output of FFNN is $\hat{y}_i = (\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{im})$.

The performance of the trained FFNN is measured by finding the distance between the predicted output $\hat{y}_i$ and the target output y_i. The aim is to minimize this value which can be expressed by the following mean squared error equation:

$$E = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{m} \left(y_{ij} - \hat{y}_{ij}\right)^2, \tag{2}$$

where N is the number of samples.

FNN can be used to solve regression and classification problems using supervised learning methods. In literature, there are many other forms of neural networks that can be applied for different purposes [5, 38, 58]. However in this paper, we only consider FFNN which is used by ELM (Fig. 4).

3 Extreme learning machine

3.1 Classical ELM

Extreme learning machine is a successful successor of training neural network using backpropagation (BP) algorithm. The main characteristic of ELM is its ability to overcome the drawbacks of training SLFNs using BP such as the local

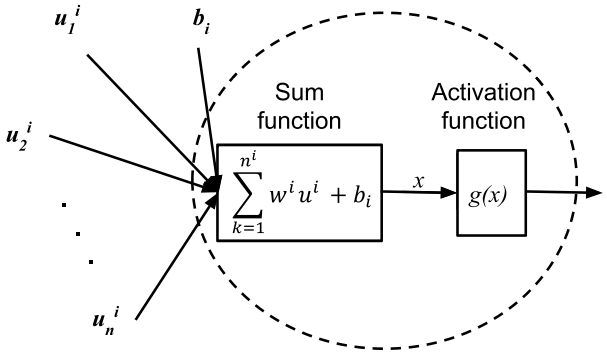

Fig. 4 MLP neuron

minima and time constraints [14]. ELM learning phase is composed of two stages: (1) assigning random weights for the connections between input layer and hidden layer and the biases, and then produce the hidden layer output matrix H. (2) finding the output weight using the least square algorithm [49]. In fact, ELM turned the learning process into a problem of solving linear system which gives this method low computational complexity. To simply train SLFN, we need to find the least squares solution $\hat{\beta}$ of the linear system $H\beta = T$.

Suppose we want to train SLFNs using N distinct samples (x_i, t_i), K neurons hidden layer and activation function $g(x)$. $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T$ is the n dimensional input vector of the ith sample, $t_i = [t_{i1}, t_{i2}, \ldots, t_{il}]^T$ is the output vector. In such a network, we will have $W_{K\times n}$ input weights (weights that connect the input neurons to the hidden neurons), $b_{K\times 1}$ the bias of hidden layer, and the output weights $\beta_{l\times K}$.

The output function of ELM can be formed as given in Eq. (3).

$$f_K(x) = \sum_{j=1}^{K} \beta_j h_j(x) = h(x)\beta, \tag{3}$$

where β_j is the weight vector that connects the hidden neuron j to the output neurons(≥ 1). $\beta = [\beta_1, \beta_2, \ldots, \beta_K]$ is the weight vector that connects the hidden layer to the output layer with number of neurons ≥ 1, and $h(x) = [h_1(x), h_2(x), \ldots, h_K(x)]$ is the output of the hidden layer. $(hx)$for a particular application, can be expressed as:

$$h_j(x) = G(w_j, b_j, x), w_i, x \in R^d, b_i \in R, \tag{4}$$

where G is a non-linear piecewise continuous function. Many activation functions can be used in the hidden neurons of the hidden layer. Among the most commonly used functions are sigmoid, hardlimit and sine. (w_j, b_j) are the parameters of the j_{th} hidden neuron and x represents an instance of the training samples as shown in Fig. 5.

Formula 3 can be rewritten as $H\beta = T$, where $H_{N\times K}$ is the output matrix of the hidden layer

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \dots & h_K(x_1)) \\ \vdots & & \vdots \\ h_1(x_N) & \dots & h_K(x_N)) \end{bmatrix} = \begin{bmatrix} G(w_1, b_1, x_1) & \dots & G(w_K, b_K, x_1) \\ \vdots & & \vdots \\ G(w_1, b_1, x_N) & \dots & G(w_K, b_K, x_N) \end{bmatrix}_{N\times K} \tag{5}$$

where $W_i = [W_{i1}, W_{i2}, \dots, W_{in}]^T$ is the weight vector that connects the input neurons to the ith hidden neuron, $x_i = [x_{i1}, \dots, x_{in}]$ is the ith sample of training set, b_i is the bias value of the ith hidden neuron, β is the output weight matrix, and T is the target output.

ELM not only tries to reach the smallest value of the training error, but also it tries to get the smallest norm of the output weights which is likely to increase the generalization performance of feedforward neural network according to Bartletts theory [8]

$$minimize : \|H\beta - T\|^2, \|\beta\|. \tag{6}$$

Figure 5 shows SLFN prepared by ELM where the weights and biases that connect the input neurons and hidden neurons are randomly initialized, and the weights between hidden neurons and the output layer are analytically determined using the least square solution of the system: $\hat{\beta} = H^{\dagger}T$, where $H^{\dagger}$ is the MP generalized inverse of the matrix H.

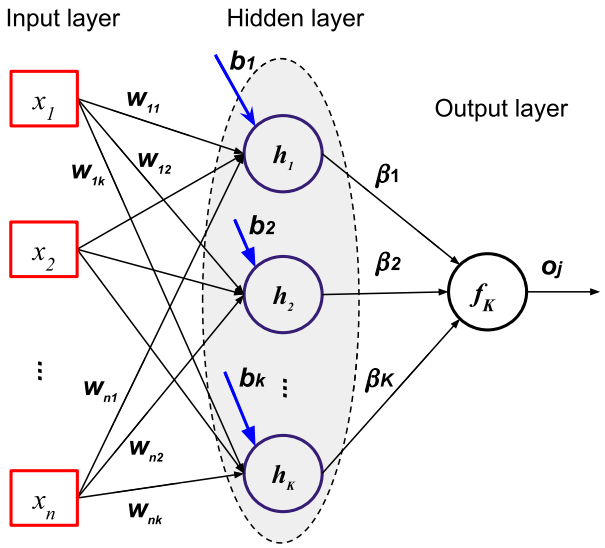

Fig. 5 Architecture of the Single hidden layer—feedforward neural network

Algorithm 1: ELM algorithm for training SLFN.

Input : $D = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, i = 1, 2, \cdots N\}$
//Training dataset D, which contains a set of training instances and their associated class labels.
g(x): activation function.
K: Number of hidden neurons
Output: Output weight β
Steps:
step 1: Initialize weights w_i and biases b_i randomly, $i = 1, 2, \cdots, K$. $w_i$is the vector of weights that connect the hidden neuron i to all input neurons.
step 2: Calculate the hidden output layer matrix **H**
step 3: Find output weights β
$\beta = H^{\dagger}T$

There are various methods to determine the MP generalized inverse such as singular value decomposition method (SVD) or orthogonal projection method. In the case of orthogonal projection method which is common and efficient solution of ELM , the value of $H^{\dagger}$ is $(H^TH)^{-1}H^T$ if (H^TH) is non-singular or $H^T(HH^T)^{-1}$ if (HH^T) is non-singular. Another way to stabilize the performance of ELM and to increase its generalization performance is to apply the ridge regression theory [36] by adding positive integer to the diagonal of the matrix (H^TH) or (HH^T) when calculating β. Accordingly the ELM output weights can be determined using the following equation depending on the size of the training datasets:

$$\beta = \begin{cases} H^T\left(\frac{I}{C} + HH^T\right)^{-1}T, & when\, N \le K \\ \left(\frac{I}{C} + H^TH\right)^{-1}H^TT & when\, N > K \end{cases} \tag{7}$$

The simple learning algorithm of ELM can be described as given in Algorithm 1.

3.2 Universal approximation and classification capabilities of ELM

The universal approximation capability was proven for generalized SLFN (eg. [39, 61]) taking into consideration that the tuning of hidden neurons parameters are performed through the training phase, and the activation functions used in the hidden neurons are continuous and differentiable. Huang et al. [43] proved that SLFN with maximum N hidden neurons and activation function such as sigmoid, ramp or radial basis is able to learn N distinct data samples with zero error. On the other hand, ELM uses random hidden neurons which mean that all the parameters of the hidden neurons (input weights and hidden biases in case of additive hidden nodes or centers, and impact factors in case of RBF networks) are set randomly without the need to tune them

using training sets. Remarkably, Huang et al. [45] proved that even with this random generation of parameters, ELM is still a universal approximator. It has been shown that ELM can use any nonlinear piecewise continuous random hidden neurons, and at the same time preserve the universal approximation property. Moreover Huang et al. [47] showed that ELM with different types of random hidden neuron networks is able to identify different disjoint regions (classification capability). ELM with quite enough number of hidden neurons and non-constant piecewise continuous activation function is able to approximate any complex decision boundaries in classification.

3.3 Multilayer ELM

As mentioned previously, classical ELM and its variants have very attractive properties such as good generalization performance and fast learning speed. These properties have played a great role in increasing the significance of ELM as a machine learning research topic. Following that, ELM and its variants have been applied for different classification tasks, and performed well in various applications. Nevertheless, ELM still faces some issues when used for practical applications such as voice recognition. In many applications, we need some feature learning before classification which introduces the need for multilayer option. Derived from this need, Kasun et al. [55] proposed a multilayer learning algorithm based on ELM called multilayer extreme learning machine (ML-ELM). The basic block of this learning algorithm is the autoencoder. Autoencoder is used as a feature extractor and can be employed in training multilayer networks. Tang et al. [95] extended ELM to propose hierarchal ELM (H-ELM) to deal with multilayer as depicted in Fig. 6. H-ELM tries to utilize the advantages of ELM theories presented in [45] that were omitted in [55]. H-ELM does not only preserve the efficiency of training, but also enhances the performance of ELM. The learning procedure of H-ELM consists of two stages: (1) feature encoding and (2) feature classification. In the first stage, H-ELM uses N-Layer of unsupervised learning for feature extraction. The output of this stage is a set of sparse features. Each layer of this stage can be treated as an independent module, and once the features of each hidden layer are extracted, they are fixed and never been tuned. The second stage uses the output of the last layer in unsupervised learning stage which (i.e. high level extracted features of the input data) as an input for the classification stage. The output is randomly perturbed (random projections of the extracted features) and taken as inputs to ELM to produce the output of the whole network.

3.4 Local receptive fields based ELM

The original ELM introduces fully connected neural network in which all the input neurons are connected to the hidden neurons. This ELM architecture has been considerably researched and been applied to many fields and applications. However, there are some fields such as image processing and speech recognition where we can benefit from strong local connections. Huang et al. [44] studied the ability of ELM to support local receptive fields and proposed local receptive fields based extreme learning machine (ELM-LRF). They showed that ELM theories are naturally valid for local receptive fields and it can work with any type of random hidden neurons if they are randomly generated using continuous probability distribution. An example of local receptive fields that can be employed by ELM are the random convolutional hidden neurons. ELM-LRF preserved the properties of ELM by generating the hidden neurons randomly and determining the output weights analytically.

However unlike ELM which focuses only on the weights between the input and hidden layers, ELM-LRF introduces randomness in two parts: the first is the random connections (the density of connections between input neurons and hidden neurons that is determined by different types of application probability distributions). The second is the weights between input neurons and hidden neurons that are generated randomly [42]. In local connections we can observe dense connections around some input neurons and sparse elsewhere. As mentioned above the fully connected ELM is extensively studied while ELM that employees local

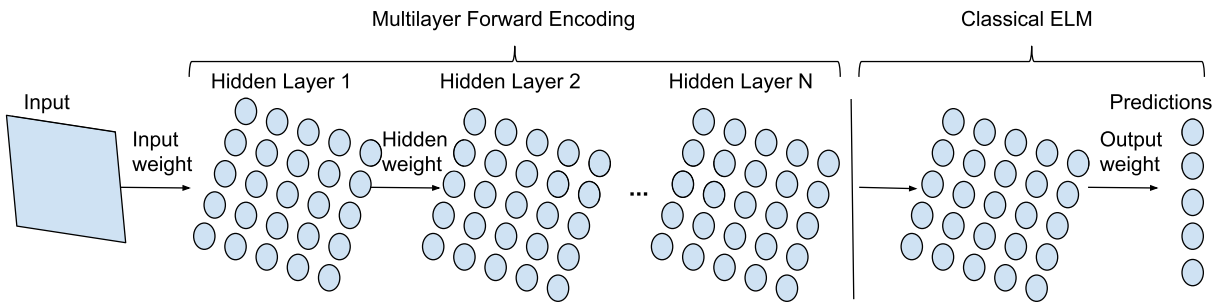

Fig. 6 Framework of H-ELM

connections is very promising research area and need more investigations.

4 Metaheuristics

The metaheuristics term is widely used to refer to a class of stochastic search algorithms that incorporate randomization and local search [103]. The main goal of metaheuristic algorithms is to find the optimal solution of hard optimization problems in a reasonable time. Theoretically, if we have unlimited time and all regions of the search space are accessible, then we will be able to find the optimality [103]. In hard optimization problems we are actually looking for a quality solution in a reasonable time taking into consideration that the optimal solution may not be obtained [10].

As previously mentioned, metaheuristic algorithms combine randomness and local search together. Dealing with large search space when solving an optimization problems makes it impossible to find every posiible solution. Therefore adding some degree of randomness is necessary and very helpful for exploring different regions and for increasing the diversity of the solutions, thus finding an optimal or near-optimal solution in an acceptable running time [103].

Metaheuristic algorithms must take into consideration two processes to be able to give an acceptable results, the first is the exploration (diversification) process, and the other one is the exploitation (intensification) process. The role of exploration is to explore the regions in the search space, looking for optimal solutions. It can be helpful in generating diverse solutions and is also useful in avoiding falling into local optima. On the other hand, exploitation is essential for concentrating the search on the promising area that possesses the good solutions.

Any metaheuristic algorithm must show a good balance between these two components in order to achieve a good performance. The main differences between metaheuristic algorithms are recognized by the the way that they combine exploration and exploitation [12]. Metaheuristic algorithms are good for complex, nonlinear and non-differential problems but usually they are suitable for a specific class of problems. This is actually confirmed by No the Free Lunch (NFL) [35, 100] theorem which states that if algorithm A performs better than algorithm B in a specific class of problems it will face a degradation in the performance in other classes of problems where B may perform better.

In the literature, metaheuristic algorithms are classified in many ways. One of the classifications that is widely used in the literature is to classify them into trajectory-based metaheuristics and population-based metaheuristics [12]. Trajectory-based algorithms use single solution and evolve to reach some satisfactory solution. They tend to focus on the exploitation process. In contrast, population-based algorithms start with a population of solutions and they tend to perform more exploration. In addition to these, another promising class of metaheuristic algorithms is memetic algorithms, where they combine trajectory and population based techniques together. Figure 7 shows the classification of metaheuristic algorithms with some examples.

4.1 Trajectory-based metaheuristics

In this section, a trajectory-based metaheuristics, which also called a single-solution metaheuristics are going to be described. This type of algorithms starts with a single solution and applies some heuristics that are inspired by nature or adapted from some phenomena. The process keeps improving the solution until a satisfactory solution is obtained [75]. Examples of trajectory algorithms are simulated annealing (SA) [57], tabu search (TS) [29], greedy randomized adaptive search procedure (GRASP) [27], variable neighborhood search (VNS) [70], guided local search (GLS) [96] and iterated local search (ILS) [91]. SA and TS are among the most commonly used and applied trajectory algorithms in the literature.

SA is inspired by the physical process of annealing, in which, the material temperature is raised and then cooling is done slowly until the material reaches low energy state. The algorithm starts with initial solution and initial value of parameter T. At each iteration, a new solution s' is selected randomly from the neighborhood of s. The acceptance of the new solution is determined according to the objective function for s and s', and the current value of T. SA uses Monte Carlo method to determine the acceptance probability of the new solution.

Unlike SA which does not learn from the past and does not use memory, TS explicitly uses memory by a mechanism inspired by human memory to manage local search. The idea of TS is to memorize recently visited areas of search space and prevent returning to them, which is referred to as cycling. TS maintains a tabu list that retains the recent solutions or some attributes of them. If the list is short, the search will exploit the area. On the contrary, if the list is long, it forces the search to explore more areas in the search space. These algorithms are called local search algorithms because they start with a current solution and then they try to modify it by changing some of its components.

4.2 Population-based metaheuristics

Population-based metaheuristics start with a collection of solutions rather than a single solution as in the case of trajectory metaheuristics. These algorithms are inspired by evolution theory, animal behavior in nature, biology, or other

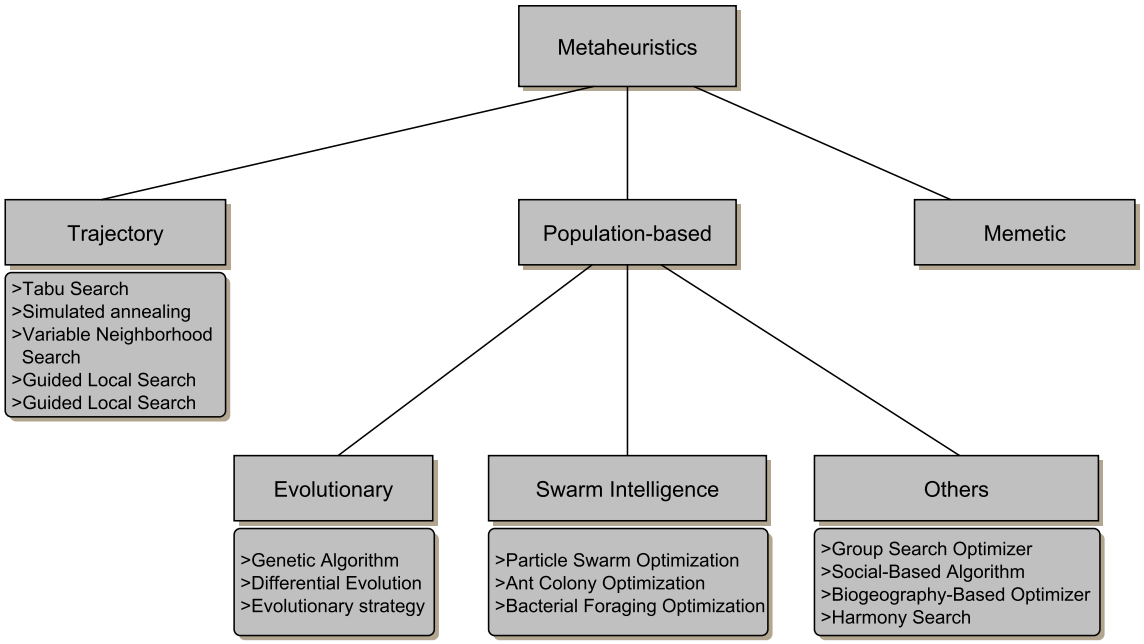

Fig. 7 Classification of metaheuristic algorithms with examples

natural phenomena. Population-based algorithms show more exploration abilities when compared to trajectory-based. They can be classified into two popular categories: evolutionary algorithms and Swarm intelligence algorithms.

4.2.1 Evolutionary algorithms

Evolutionary algorithms search the complex search space for the best solution using a methodology derived from the nature such as mutation, selection and reproduction. Examples of evolutionary algorithms are genetic algorithm (GA) [37], evolution strategy (ES) [88], Differential evolution (DE) [90] and genetic programming (GP) [59]. The basic idea behind all existing variants of evolutionary algorithms is almost similar. They all generate a set of solutions to form a population, then they calculate the fitness of individuals and apply a number of reproduction operators while keeping the best individual.

In evolutionary algorithms, a set of candidate solutions at a given iteration is called a generation. The solutions are also called individuals and their quality as a solution is calculated using a predefined fitness function. In each generation, solutions with the best fitness values have higher probability to be selected for reproduction to form the next generation. Evolutionary algorithms usually apply a set of operators on the individuals to reproduce new generations.

A typical example of evolutionary algorithms is GA which was first proposed by John Holland in 1975. Since then, it has been used in many applications to find the optimal or near-optimal solutions [67]. Natural selection from biological evolution is the base of genetic algorithm.

4.2.2 Swarm intelligence

Swarm intelligence (SI) algorithms mimic the social behavior of swarms or flocks (animals, insects, fish, ...). The most popular example of this category is particle swarm optimization (PSO) [22]. Some of other widely used SI algorithms are Ant colony optimization (ACO) [21], bacterial foraging optimization (BFO) [79], fish swarm algorithm (FSA) [64] and artificial bee colony (ABC) [53]. The main idea of SI algorithms is to start with a swarm (a set of solutions that are randomly generated) and then modify these solutions depending on a number of heuristics inspired by the behavior of swarms in nature.

PSO [22] is inspired by the search techniques that are used by flocks of birds to discover unknown places. It uses a population (swarm) of solutions (particles) that are modified during the iterations. PSO has gained a huge popularity because of its simplicity and efficiency [2]. Each particle in PSO represents a solution. Basically, the search process in PSO depends on two main factors: each particle retains its best experience which is called (pbest) and the best among the whole swarm which is called (gbest) [87]. The particles have position and velocity in the search space where they will be changed using a predefined equations in order to find the global optimum.

ACO is another well-regarded metaheuristic algorithm [21]. It simulates the way that ants follow to find the path

between the colony and the food. Ants in nature use pheromone as a chemical communication medium between each other to find the shortest path to the source of food.

4.3 Memetic algorithms

Memetic algorithm (MA) is a class of metaheuristic algorithms that combines population-based with trajectory-based metaheuristic algorithms in order to incorporate global optimization with local search techniques [72]. Evolutionary algorithms such as GA and DE perform well on the side of exploration but they are poor when exploiting the neighborhood. On the other hand, trajectory-based algorithms such as SA and TS search exploit the neighborhood of the solution very well but they may get trapped in a local optima. MA is a promising and growing category of metaheuristics that try to solve these problems by combining global optimization techniques with local search methods [34, 60].

5 Metaheuristic formulation of the ELM components

The performance of SLFNs is highly affected by two main factors: the structure of the network and the learning algorithm. One of the most interesting properties about FFNN setup is that it can be seen as an optimization problem [105]. The goal is to find the best model that establishes the best predictive relationship between the data and the output. In this section, we discuss the models presented in the literature to optimize the FFNN regarding specific learning algorithm which is ELM.

ELM gains its popularity as a learning algorithm for SLFN due to its good generalization and fast learning characteristics, especially with large and complex datasets. The number of neurons in the hidden layer, the initial values of weights between layers, biases connected to the hidden neurons and activation function, all form a high influence on the performance of the ELM [69]. Metaheuristics were applied in the literature to optimize the different components of the network such as the structure, input weights, and the activation functions.

In general, before applying any metaheuristic algorithm for optimizing a given problem, there are two crucial design issues that are very important to address:

- The design of the individual (also known as chromosome or particle in GA and PSO, respectively) which represents the solution of the given problem.
- The selection of the fitness function that is going to be used to evaluate the quality of the generated solutions by the algorithm.

Similarly, utilizing metaheuristics for optimizing ELM networks have to address the two aforementioned points. The design of the solution in the metaheuristic algorithm depends on the components of the ELM network that are intended to be optimized. For instance, we can optimize only the weights and biases of the network and keep its structure fixed, or we can simultaneously optimize different components such as the weights and biases, and the number of neurons.

After conducting a thorough review of what has been done in the literature, we have identified three main approaches for optimizing ELM networks based on metaheuristic algorithms. The main stream and the most studied approach is to utilize a metaheuristic algorithm for optimizing only the weights and biases which represent the links that connect the input and the hidden layers neurons. The second approach which gained less interest in the literature is the simultaneous optimization of the weights and biases in addition to the structure of the network (i.e. number of neurons). The third approach is concerned with optimizing the values that are produced by the hidden neurons. We have noticed that the latter approach is so far the least investigated among the other approaches.

After designing the individual encoding of the ELM components that were pre-specified for optimization, any metaheuristic can be utilized to optimize these components. In the case of trajectory-based metaheuristics the algorithm starts with one individual, while in the case of population-based metaheuristics it starts with a set of individuals that represents the first generation. In both cases, the quality of the individual must be evaluated using a predefined fitness function. We noticed that the most popular fitness function used in the reviewed papers is the root mean squared error (RMSE) which can be measured as given in Eq. (8):

$$RMSE = \sqrt{\frac{\sum_{j=1}^{N} \left\| \sum_{i=1}^{K} \beta_i g(w_i.x_j + b_i) - t_i \right\|}{m \times N}}. \tag{8}$$

In the following three subsections, we thoroughly review the main three approaches proposed in the literature for optimizing ELM networks using metaheuristic algorithms.

5.1 Optimizing input weights and biases

Computing output weights in classical ELM based on random input weights and hidden biases could lead to a non-optimal performance and could result in ill-condition [33, 69, 74]. Moreover, random inputs weights and hidden biases could lead to overfitting because the ELM will learn the training data too well [89].

Realizing these drawbacks, much previous research have proposed the combination of metaheuristic algorithms with ELM to improve the selection of the input weights that are used to produce the output matrix. Instead of starting with totally random input weights and hidden biases, a metaheuristic algorithm is used to help in conditioning the weights and biases of ELM and to search for a better combination of them, and consequently to give a better solution.

Different types of metaheuristic algorithms were proposed in the literature for optimizing the input weights and hidden biases in ELM networks. The most studied approach is the random initialization of a population of solutions where each of which represents a possible values of input weights and hidden biases. We have identified that the general encoding scheme of the individual used in most of works that follow this approach can be depicted as shown in Fig. 8. In this scheme, the individual contains n elements depending on the number of input neurons and number of hidden neurons as given in Eq. (9):

$$n = Z \times K + K, \tag{9}$$

where Z is the number of features and K is the number of hidden neurons.

In Fig. 9 we draw a flowchart for the general model that deploys any metaheuristic algorithm for optimizing the input weights and biases of ELM. The first step is to generate the initial population of random individuals. The next step is to calculate the fitness value for each individual. Usually, the fitness value is the RMSE of the resultant ELM model that starts with the input weights and biases represented by the individual. Then a new generation is constructed by applying the metaheuristic operators on the current generation. The process continues until the termination condition is met.

Following this approach, many of the proposed algorithms aimed at optimizing the input weights and hidden biases in order to improve the performance of ELM and minimize the number of neurons in the hidden layer.

In the early beginning of ELM, the problem of randomized weights was identified, and many researchers worked on this problem. One of the earliest work in this approach was proposed by Zhu et al. [80] where the DE algorithm is used to optimize the input weights. Their evolutionary

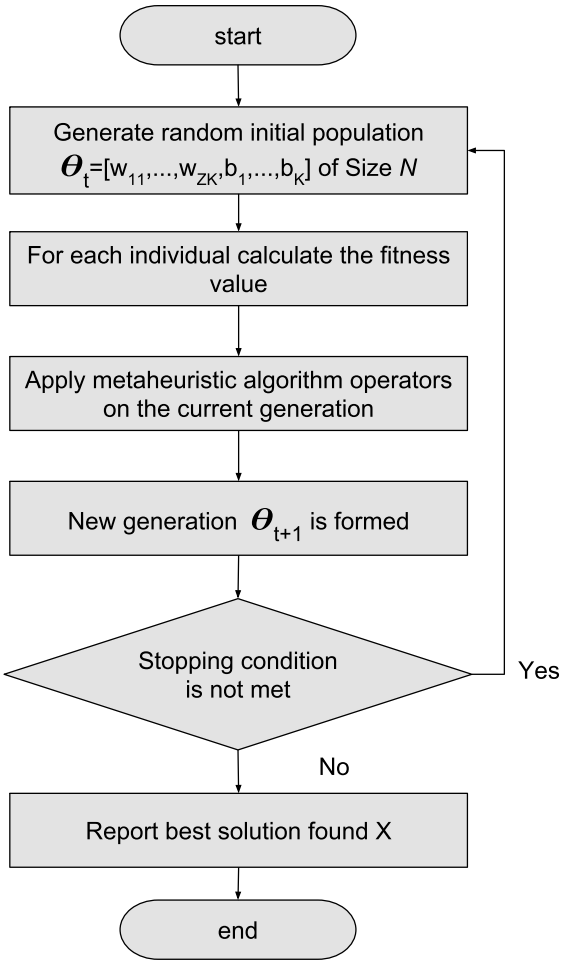

Fig. 9 Combining ELM with metaheuristic algorithm

ELM (E-ELM) showed a good generalization performance with more compact networks when experimented on different classification tasks.

In fact, the work of Zhu et al. [80] paved an important research line which got the attention of many researchers. For example, Huynh and Won [52] proposed a modified version of the E-ELM algorithm of their predecessors in [80]. Their evolutionary least square ELM (ELS-ELM) specifies the input weights and the hidden biases using DE, except

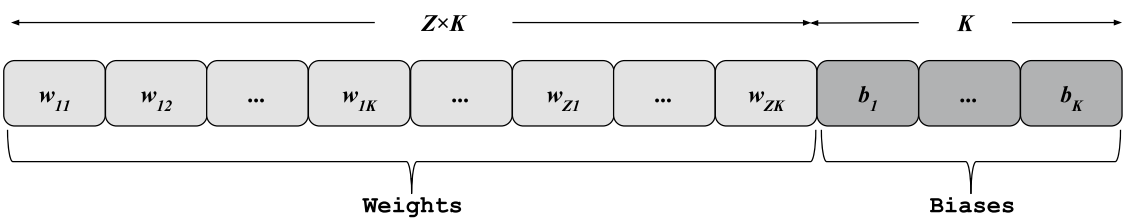

Fig. 8 Individual

that instead of starting with random individuals in the first generation, they used least square scheme to initialize the first generation.

Another work based on the work of Zhu et al. [80] was also proposed by Sánchez-Monedero et al. [85]. They slightly modified the E-ELM model by replacing the typical RMSE based fitness function with a new fitness function that combines the accuracy rate with the sensitivity. For verifying the proposed model, the authors conducted their experiments based on 8 binary and multi-class classification datasets and compared their approach with E-ELM and other multi-objective MLP network. The results showed that the proposed model improved the sensitivity rates while maintaining the accuracy for imbalanced datasets.

Many work in this area which investigate the application of the well-regarded PSO algorithm or one of its variations for the task of optimizing hidden neurons and biases in ELM have been proposed. For example, Xu and Shu [104] proposed PSO for this task with an attempt to minimize the number of parameters that are manually set in the model. However, the performance of their proposed PSO-ELM model was tested based on one dataset only and compared to the classical BP network and other PSO-based classifier. Although PSO-ELM model showed competitive results, it was slower than the BP-based network.

A variant of PSO was also proposed by Han et al. [33] to condition the input and biases weights in ELM. In their approach, they introduced an improved particle swarm optimization (IPSO-ELM) to select the weights of input and biases attached to Moore–Penrose to determine the output weights. Interestingly, the selection of input and biases weights were optimized based on the RMSE calculated over a validation dataset unlike most of the models that calculate their fitness over the training set. Another interesting feature in their model is the incorporation of the norm of the output weights in the optimization process. Compared to the classical ELM and other evolutionary ELM networks, their model obtained lower error values with more compact networks over three different datasets.

Another variant of PSO was proposed by Pacifico and Ludermir [77] to optimize the input weights and biases in ELM. Their approach which was called CPSOS-ELM and based on a modified version of PSO uses the local best neighborhood of PSO population through a mechanism called population stereotyping. Compared to ELM, PSO-ELM, LM and PSO-LM over four classification datasets, their proposed model achieved best results in three datasets.

Another swarm-based algorithm called bacterial foraging (BF) algorithm [79] was applied by Cho et al. [15] for optimizing ELM. BF is a swarm-based algorithm that mimics the process which bacteria follow while searching for food. The authors used BF to search for the optimal solution that represents the best input weights and hidden biases for ELM network. Their proposed algorithm was applied on two cases: the first is a regression problem which is the California Housing dataset, and the second is a medical classification task. Their experiments showed that the performance of the BF-ELM networks outperformed the classical ELM, SVM and BP network in both cases.

A hybrid approach that combines ELM with Group Search Optimizer (GSO) was proposed in [89]. In GSO-ELM, the goal was to find the best weights and biases with a reduced number of hidden neurons to achieve a good performance. The authors applied four different variations of GSO. The fitness function that was used in this algorithm was RMSE. For evaluation, the method was tested using six different classification datasets obtained from the UCI repository. The model was implemented in four variations and all were compared to the original ELM, PSO-ELM and LM. All the models based on GSO performed better than ELM, PSO-ELM and LM.

Following the same classical approach, improved cuckoo search (ICS) was used to optimize the input weights and the hidden biases of ELM in [71]. Similar to the original ELM, the output weights are determined using MP generalized inverse. Their proposed model which was called ICSELM was tested on four medical classification datasets obtained from UCI. Unlike most of the previous works, they used a wider range of different evaluation measures to assess the performance of the different models. Such measurements include the accuracy, sensitivity, specificity, G-mean, F-score, norm of the output weights and the Area Under ROC Curve (AUC). Their experiments showed that the proposed ICSELM outperformed other ELM models.

Memetic algorithms were also utilized for optimizing the same parameters in ELM. Zhang et al. [111] proposed a model based on a memetic algorithm to improve the classification accuracy. Their M-ELM model combined population-based optimization methods with SA as a local heuristic search method. In contrast to most of the previous works, they used classification accuracy as a fitness function. Another noticeable point in their work is that they experimented the M-ELM model on a large number of benchmark datasets (i.e. 22 classification datasets). Although M-ELM showed improvement in classification accuracy especially in datasets with large number of features, it consumes more execution time compared to other evolutionary-based ELM models. Recently, a model based on dolphin swarm algorithm is proposed to optimize the input weights and hidden biases of ELM [97]. Each individual is called dolphin and is evaluated using RMSE. They compared their model to the classical ELM and three state-of-the-art models that optimize this component of ELM. They conducted their experiments based on five regression datasets and seven classification datasets. The proposed model showed better generalization performance

than the start-of-the-art methods but failed to reduce the number of hidden neurons compared to the other methods, and it was slower than the standard ELM.

Other examples of this classical approach with RMSE as fitness function are DE-Coral Reef-ELM in [107], dragonfly algorithm (DA) with ELM in [84] and flower pollination algorithm ELM in [76].

In Table 1, we summarize the harvested work that has been done to optimize the weights and biases of ELM using metaheuristic algorithms. We have noticed that the two mostly used metaheuristic algorithms are DE and PSO, and their variants. Also, we can see that metaheuristic-based ELMs are used for both classification and regression supervised learning problems with more focus on classification.

For validation, some authors tested their proposed model on a single dataset that represents a specific domain [104, 105]. Other authors benchmarked their model based on a small number of datasets of a specific type of machine learning task such as classification, regression or both [14, 15]. While other studies utilized a higher number of datasets like in [69, 111] with 16 and 22 datasets, respectively.

In this section, we have reviewed the techniques that used different metaheuristic algorithms to optimize weights and biases in ELM. In the next section, we will review the approaches that added one more component to the optimization process, which is the number of hidden neurons.

5.2 Optimizing number of neurons

Like any other neural network, the number of hidden neurons in ELM has a significant effect on its performance. Moreover, random input weights and hidden biases in classical ELM make the network tend to require a higher number of hidden neurons in order to perform well [33, 69, 80, 85]. From the literature, we have identified four approaches that addressed this problem, which were based on utilizing metaheuristic algorithms to optimize the number of neurons in the hidden layer.

The first approach was proposed by Suresh et al. [94] where they developed a modified GA for the automatic selection of hidden neurons and their corresponding input weights and biases. Their real-coded GA (RCGA-ELM) modified updating operators to deal with variable-length individuals. In fact, two genetic operators were used: the first selected the hidden neurons, while the second optimized the values of the input weights and biases. In their implementation, the individual was encoded as a two-dimensional array. If we have n dimensional input and L hidden neurons, then we will have the m

Table 1 Summary of main works in the literature that applied metaheuristics for optimizing the input weights and biases in ELM networks

Model	Classification	Regression	Datasets	Metaheuristic algorithm	Fitness function
E-ELM [80]	✓		4 datasets	Differential evolution	RMSE
PSO-ELM [104]		✓	1 dataset	Particle swarm optimizer	RMSE
BF [15]	✓	✓	2 datasets	Bacterial foraging	$\frac{1}{1+MSE_{trn}}$
ELS-ELM [52]		✓	4 datasets	Differential evolution	RMSE
E-ELM-CS [85]	✓		8 datasets	Differential evolution	$\frac{1}{(1-\lambda)C+\lambda S}$
GSO-ELM [89]	✓		6 datasets	Group search optimizer	RMSE
AFSA-ELM[17]	✓		5 datasets	Artificial fish swarm algorithm	Accuracy
MAFSA-ELM[17]	✓		5 datasets	Adaptive modified artificial fish swarm algorithm	Accuracy
SaE-ELM [14]	✓	✓	11 datasets	Self-adaptive differential evolution algorithm	RMSE
GA-ELM [105]			1 dataset	Modified genetic algorithm	$\frac{1}{1+RMSE}$
IPSO-ELM [33]	✓		3 datasets	Particle swarm optimizer	RMSE
CPSOS-ELM [78]	✓		4 datasets	Particle swarm optimizer	RMSE
O-ELM [69]	✓	✓	6 datasets	Genetic algorithm Differential evolution Simulated annealing	RMSE
QPSO-ELM [106]	✓		5 datasets	Quantum-behaved particle swarm optimization	$\frac{\gamma}{2}RMSE + \frac{1}{2}\|\beta\|$
ICSELM [71]	✓		4 datasets	Improved cuckoo search	Accuracy + norm
M-ELM [111]	✓		22 datasets	Differential evolution + simulated annealing	Classification accuracy
SA-ELM [74]		✓	8 datasets	Ameliorated teaching learning based optimization (ATLBO)	RMSE
DS-ELM [97]	✓	✓	13 datasets	Dolphin swarm algorithm	RMSE
CSO-ELM [24]	✓		15 datasets	Competitive swarm optimizer	Accuracy

individuals where each of which can be represented as given in Eq. (10):

$$b_i = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} & b_1 \\ w_{21} & w_{22} & \dots & w_{2n} & b_2 \\ & \vdots & & & \\ w_{m1} & w_{m2} & \dots & w_{mn} & b_L \end{bmatrix} \tag{10}$$

$w_{j1} \dots w_{jn}, b_j$ are the weights and biases for the hidden neuron j. Each row represents one neuron.

The second approach was proposed by Xue et al. [102]. They found that in the former approach there were many genetic parameters that needed to be tuned artificially. Therefore, they proposed a model called variable-length PSO (VPSO-ELM) to optimize two components of ELM: the input weights and the number of neurons. In order to make PSO applicable for this problem, a modified version was proposed to deal with individuals that have variable length; due to the difference in the number of neurons. The way that individuals were encoded in this model was adopted from [94].

Following a different approach, Huang and Lai [51] used PSO with structural risk minimization to optimize only the number of neurons. The target was to find the effective number of hidden neurons with a good generalization performance. In their PSO variant, the position P represented the number of hidden neurons in ELM.

The fourth approach was proposed by Freire and Barreto in [28]. They proposed a model called Adaptive Number of hidden neurons approach (ANHNA) with a goal to automatically select the required number of neurons in addition to biases and slopes of the activation function. The optimization process was performed using DE and PSO. In ANHNA, the individual (chromosome) was encoded as shown in Eq. (11) where $\overrightarrow{C_i}$ represents a vector of real numbers with a dimension that is equal to $3 \times Q$ where Q is the number of neurons:

$$\overrightarrow{C_i} = \left[T_{i1}, \dots, T_{iQ}, a_{i1}, \dots, a_{iQ}, b_{i1}, \dots, b_{iQ}\right] \tag{11}$$

In Eq. (11), T_{ij} represents the *jth* neurons in the *ith* individual. If the value of $T_{ij} \geq 0.5$ then the neuron is activated, else the neuron is deactivated.

Table 2 lists the models that were designed to optimize the structure of the SLFN (number of hidden neurons) used by ELM. In comparison with Table 1, we can see that the amount of work that has been done so far to optimize the number of neurons is much less than those for optimizing the input weights and hidden biases. Moreover, only DE and PSO have been used for optimizing this component. The maximum number of datasets tested is eight, and we can always see that only one type of problems, either classification or regression, is addressed in the articles.

5.3 Optimizing activation functions

One of the fundamental decisions when constructing neural networks is the selection of the activation function. The selection of activation function is problem-dependent and the performance of FFNN can highly be affected by this selection [62, 75].

Activation functions are used to transfer input signals into output signals. The main role of these functions is to make neural networks able to solve nonlinear problems [54]. Similarly, the performance of ELM networks can be improved by a proper selection of the activation function used by neurons (also known as node optimization). This optimization can be performed by keeping the input weights and hidden biases fixed while optimizing the activation function or by optimizing both components simultaneously. ELM supports a wide range of activation functions [98]. In the original ELM, the activation functions listed in Table 3 were implemented.

One of the main works that addressed the automatic selection of the activation functions was presented by Matias et al. [68]. The authors developed a model called genetically optimized ELM (GO-ELM). In this model, the output weights were calculated using least squares algorithm with Tikhonovs regularization. GA was used to optimize different components of ELM simultaneously. These components are the input weights and biases, the hidden layer components represented by the number of neurons, and the activation function used in each neuron. The activation function of each neuron in this model can be zero, sigmoid function, or linear function. Their work was tested using five benchmark datasets. The results showed equal or better results when compared to IGA-SLFN, SaE-ELM, LM-SLFN, and ELM in four datasets. Later on, the same authors proposed a learning framework

Table 2 Summary of main works in the literature that applied metaheuristics for optimizing number of neurons in ELM networks

Model	Classification	Regression	Datasets	Metaheuristic algorithm
RCGA-ELM [94]		✓	3 datasets	Genetic algorithm
SRM-ELM [51]	✓		6 datasets	Particle swarm optimizer
ANHNA[28]		✓	4 datasets	Particle swarm optimizer Differential evolution
VPSO-ELM [102]	✓		8 datasets	Variable-length particle swarm optimizer

Table 3 Activation functions

Name	Function
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Sine	$f(x) = \sin x$
Hardlim	$f(x) = 1\ if\ n \geq 0,\ \ 0\ \ otherwise$
Triangular basis	$f(x) = 1 - abs(x)\ if\ \ -1 \leq x \leq 1,\ \ 0\ \ otherwise$
Radial basis	$f(x) = e^{-x^2}$

for SLFN called optimized ELM (O-ELM) [69]. In this framework, the structure and parameters are optimized using three different optimization methods: GA, DE (population-based) and SA (trajectory-based). Experiments were conducted based on 16 benchmark datasets. The GA version of O-ELM performed better than IGA-SLFN, SaE-ELM, LM-SLFN, and ELM in 10 out of the 16 datasets.

A different approach was implemented by Li et al. [62]. The authors optimized a special type of ELM called tunable activation function ELM (TAF-ELM). In this model, a weighted combination of activation functions was embedded in each hidden neuron. Then, DE was used to optimize these weights in addition to the input weights, hidden biases and activation functions. TAF-ELM algorithm showed better performance than TAF-BP and TAF-MFNN algorithms when applied to four different datasets representing different problems such as classification and approximation.

6 Applications of metaheuristic based ELM

Metaheuristic-based ELM models have been used to solve a wide range of challenging and complex real world problems in different areas. In the following subsections, we highlight the main applications of these models, while in Table 4, we list most of the applications found in the literature.

6.1 Medical applications

Many models of metaheuristic-based ELMs have been proposed and applied for medical applications. Cho et al. [15] proposed Evolutionary ELM for optimizing ELM using BF. The model was applied on a real medical classification dataset called "Pima Indians Diabetes Database". Mohapatra et al. [71] proposed an improved cuckoo search for optimizing ELM and applied their model on four medical classification data: breast cancer, diabetes, bupa, hepatitis. The authors used different important evaluation measures that are commonly used in the medical field. The results showed that the proposed model could be promising for medical applications.

Another metaheuristic-based ELM model for medical applications was recently proposed by Eshaty et al. [24]. The model used a hybrid methodology that utilized Competitive Swarm Optimizer to optimize ELM (CSO-ELM). The performance of this model was tested using 15 medical classification problems and also tested for function approximation of SinC function. The model used ELM and Regularized ELM. CSO-ELM not only increased the generalization performance of ELM compared to PSO and DE, but also

Table 4 Summary of main metaheuristic based ELMs designed for specific applications

Model	Year	Application	Metaheuristic algorithm
PSO-ELM [104]	2006	Production prediction	PSO
BF-ELM [15]	2007	Medical data classification	BF
RCGA-ELM [94]	2009	Visual quality of JPEG images	GA
E-ELMcv,IE-ELMcv [66]	2013	Image analysis	DE
ELM-IPSO [6]	2013	Medical-brain tumor tissue characterization	PSO
GO-ELM [68]	2013	Temperature of cement kiln plant	PSO
Modified GA-ELM [105]	2013	Power system economic dispatch	GA
QPSO-ELM [92]	2014	Handwriting numeral recognition	PSO
ICS-ELM [71]	2015	Medical classification	Improved CS
ELM-MABC [63]	2015	Short-term load forecasting	Modified ABC
BDE-ELM,SaDE-ELM,TDE-ELM [65]	2016	Prediction of effluent from WWTP	DE
GA-ELM [50]	2016	SO_2 emissions prediction	GA
SaE-ELM [73]	2016	Soil temperature prediction	DE
Self-adjusting ELM [74]	2016	Pulverized coal furnace thermal efficiency	ATLBO
SaELM [86]	2017	Water network management	DE
CSO-ELM [24]	2018	Medical classification	CSO

significantly decreased the training time of Evolutionary ELM.

A specific medical application for brain tumor tissue characterization was investigated by Arunadevi and Deepa [6]. The authors proposed an improved PSO to optimize ELM (ELM-IPSO) for classifying and segmenting of brain tissue and tumor from 3D MRI tumor images. Interestingly, ELM-IPSO scored 98.25% accuracy when applied to SPL Harvard benchmark dataset which contains contrast and non-contrast 3D MR brain images. Moreover, the model was tested using real-time datasets.

6.2 Image analysis and applications

In image applications, Suresh and Babu [94] introduced RCGA-ELM to estimate the visual quality of JPEG images. The results of their experiments showed that the model improved the assessment process of image quality of ELM classifier. For image analysis, Liu and Wang [66] proposed an approach based on E-ELM for face images classification. The authors improved the classification accuracy of E-ELM by introducing two variants of E-ELM: cross-validation-based evolutionary ELM (E-ELMcv) and cross-validation-based improved evolutionary ELM (IE-ELMcv). In both cases, the authors introduced cross validation to avoid using extra validation set for training. They selected classification accuracy as their fitness function. Experiments were applied on four datasets of face images. Experimental results showed that the proposed algorithms outperformed other tested algorithms in terms of accuracy, and they were effective in image analysis.

6.3 Environmental applications

Different variants of metaheuristic-based ELM models were deployed and investigated in remarkable environmental applications. For example, it is well known that SO_2 is a harmful pollutant for the environment and it is very hard and challenging to control the percent of SO_2 in the air due to the complex and uncontrollable emission processes in many countries. Huang et al. [50] proposed a GA-based ELM model for predicting SO_2 concentration in the exhausted emissions into air as a result of production process of chemical fertilizers. Their GA-ELM model showed relatively accurate and efficient prediction results.

Another environmental application was proposed by Nahvi et al. [73] to predict the soil temperature in different depths. Soil temperature is important in many fields such as hydrology, agriculture and atmospheric researches. A self-adaptive evolutionary ELM (SaE-ELM) was introduced to manage this application. SaE-ELM used DE to optimize ELM input weights and hidden biases. This approach was compared to ELM and validated against GP and ANN. Their SaE-ELM showed higher generalization performance.

6.4 Power, control, and other engineering applications

In power engineering, the application of metaheuristic-based ELM models was investigated in short-term load forecasting (STLF). Li et al. [63] developed a modified artificial bee colony (MABC) to tune the parameters of ELM to forecast the load power. Accurate forecasting results are important in this application to improve the electrical power system and to minimize operating costs. ELM-MABC was tested using two datasets: ISO New England data and North American electric utility data. Another power application that can greatly benefit from optimization is the power system economic dispatch problem. The idea is to optimize set of parameters to minimize the generation costs. Yang et al. [105] used a modified GA to optimize ELM, and applied it for power system economic dispatch.

In control engineering, Niu et al. [74] proposed a self-adjusting ELM for pulverized coal furnace thermal efficiency. The problem was both complex and non-linear which makes it an appropriate candidate to be solved using metaheuristic-based ELM. Another application in control engineering was investigated by Lin et al. [65] for predicting the effluent from wastewater treatment plant (WWTP). The authors proposed three variations of evolutionary ELM algorithms for the problem: basic differential evolution ELM (BDE-ELM), self-adaptive DE (SaDE-ELM), and another improved variation of DE with local search (TDE-ELM). TDE-ELM performed better than all other algorithms, and the experiments showed that the three applied evolutionary ELMs outperformed the traditional ELM.

In cement kiln plants, Matias et al. [68] proposed a metaheuristic based ELM to predict the temperature in the burning zone. The model utilized GA to tune the structure of ELM in order to use it for estimating the temperature in this application. The results showed that the developed model was able to replace the physical pyrometer sensor in estimating the temperature of the burning zone.

In maintenance and rehabilitation, Sattar et al. [86] Proposed a prediction model for ongoing maintenance/rehabilitation of the water network. The model estimates the next potential pipe failure within the network. In their work, the authors compared ELM with three of its variants. One of these variants was self-adaptive ELM (SaELM). SaELM optimizes ELM using DE. The results showed that ELM outperformed its three other variants. The work also showed that ELM based models performed better than other models like ANN, support vector machine (SVM), and non-linear regression (NNR).

7 Conclusions and possible future directions

In this paper, we provided the first literature review on the studies that addressed optimizing ELM networks using metaheuristic algorithms. Two main types of works in this domain were reviewed: the first focused on the works that targeted the improvement of ELM by optimizing its different components using metaheuristic algorithms, while the second focused on the applications of metaheuristic-based ELM networks in real-life problems.

Based on the review presented in this work, we can see that metaheuristic algorithms have been intensively investigated over the last decade for optimizing the components of ELM networks. Despite the popularity of metaheuristic-based ELMs and their recent advances, there are still several areas that need further investigation:

- The state-of-the-art applications of metaheuristic-based ELMs in real life specific problems are still at early stage. It would be very interesting for the researchers and practitioners to see how metaheuristic-based ELM networks would perform in a wide range of common and popular applications which are not investigated enough by this approach, such as patter recognition, health and medical applications, environmental modeling, and engineering problems.
- To the best of our knowledge, there is still no work that have investigated the performance of this type of models at large-scale datasets with very high dimensionality. In the era of big data, such studies could be extremely useful for some domains like bioinformatics, image processing and other real-time engineering problems.
- With the rapid increase in the amount of training data in massive applications, recently, there have been interesting studies conducted on developing various ELM version to overcome the memory problems and costly large matrix operations [9, 108–110]. On the other side, the efficiency of metaheuristic algorithms have been also studied at the large scales for same reasons [3, 4, 101]. However, the integration between the two research lines still at a very early stage and have the potential for promising performance outcome.
- As it was shown in Sect. 5, the works that addressed optimizing the structure and activation functions are much less than those addressing optimizing the input weights and biases. Further investigation can be made at the level of those two components. It would also be very interesting to see the applications of these approaches in real-life problems.
- Based on our survey it is noticed that the most commonly implemented fitness functions in ELM are the MSE and accuracy rates. However, to the best of our knowledge, there is no study that compares the performance of ELM based on these different fitness functions. We believe that such a study would be very beneficial and interesting for the audience in this field.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

References

1. Alade OA , Selamat A, Sallehuddin R (2017) A review of advances in extreme learning machine techniques and its applications. In: International conference of reliable information and communication technology. Springer, pp 885–895
2. Alexandridis A, Famelis IT, Tsitouras C, Simos T, Tsitouras C (2016) Particle swarm optimization for complex nonlinear optimization problems. In: AIP conference proceedings, vol 1738. AIP Publishing, pp 480120
3. Aljarah I, Ludwig SA (2012) Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In: Nature and biologically inspired computing (NaBIC), 2012 fourth world congress on. IEEE, pp 104–111
4. Aljarah I, Ludwig SA (2013) A new clustering approach based on glowworm swarm optimization. In: Evolutionary computation (CEC), 2013 IEEE congress on. IEEE, pp 2642–2649
5. Almeida LB (1989) Backpropagation in perceptrons with feedback. In: Neural computers. Springer, pp199–208
6. Arunadevi B, Deepa SN (2013) Brain tumor tissue categorization in 3D magnetic resonance images using improved PSO for extreme learning machine. Progress Electromagn Res B 49:31–54
7. Azzini A, Tettamanzi AGB (2011) Evolutionary anns: a state of the art survey. Intell Artif 5(1):19–35
8. Bartlett PL (1998) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inf Theory 44(2):525–536
9. Bi X, Zhao X, Wang G, Zhang P, Wang C (2015) Distributed extreme learning machine with kernels based on mapreduce. Neurocomputing 149:456–463
10. Leonora B, Marco D, Gambardella LM, Gutjahr WJ (2009) A survey on metaheuristics for stochastic combinatorial optimization. Nat Comput 8(2):239–287
11. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Computi Surv 35(3):268–308
12. BoussaïD I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237:82–117
13. Cao J, Lin Z (2015) Extreme learning machines on high dimensional and large data applications: a survey. Math Probl Eng
14. Cao J, Lin Z, Huang G-B (2012) Self-adaptive evolutionary extreme learning machine. Neural Process Lett 36(3):285–305
15. Cho J-H, Lee D-J, Chun M-G (2007) Parameter optimization of extreme learning machine using bacterial foraging algorithm. J Korean Ins Intell Syst 17(6):807–812

16. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 2(4):303–314
17. de Oliveira JFL, Ludermir TB (2012) An evolutionary extreme learning machine based on fuzzy fish swarms. In: Proceedings on the international conference on artificial intelligence (ICAI), p 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)
18. Ding S, Chunyang S, Junzhao Y (2011) An optimizing bp neural network algorithm based on genetic algorithm. Artif Intell Rev 36(2):153–162
19. Ding S, Xinzheng X, Nie R (2014) Extreme learning machine and its applications. Neural Comput Appl 25(3–4):549–556
20. Ding S, Zhao H, Zhang Y, Xinzheng X, Nie R (2015) Extreme learning machine: algorithm, theory and applications. Artif Intell Rev 44(1):103–115
21. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B Cybern 26(1):29–41
22. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Micro machine and human science, 1995. MHS'95., Proceedings of the sixth international symposium on. IEEE, pp 39–43
23. Ertuğrul ÖF, Kaya Y (2014) A detailed analysis on extreme learning machine and novel approaches based on elm. Am J Comput Sci Eng 1(5):43–50
24. Eshtay M, Faris H, Obeid N (2018) Improving extreme learning machine by competitive swarm optimization and its application for medical diagnosis problems. Expert Syst Appl 104:134
25. Faris H, Aljarah I, Mirjalili S (2016) Training feedforward neural networks using multi-verse optimizer for binary classification problems. App Intell 45(2):322–332
26. Faris H, Aljarah I, Al-Madi N, Mirjalili S (2016) Optimizing the Learning Process of Feedforward Neural Networks Using Lightning Search Algorithm. Int J Artif Intell Tools 25(06):1650033
27. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Glob Optim 6(2):109–133
28. Freire A, Barreto G (2014) A new model selection approach for the elm network using metaheuristic optimization. In: European symposium on artificial neural networks, computational intelligence and machine learning (ESANN)
29. Glover F (1989) Tabu search part I. ORSA J Comput 1(3):190–206
30. Gori M, Tesi A (1992) On the problem of local minima in backpropagation. IEEE Trans Pattern Anal Mach Intell 1:76–86
31. Gupta JND, Sexton RS (1999) Comparing backpropagation with a genetic algorithm for neural network training. Omega 27(6):679–684
32. Hagan MT, Menhaj MB (1994) Training feedforward networks with the Marquardt algorithm. IEEE Trans Neural Netw 5(6):989–993
33. Han F, Yao H-F, Ling Q-H (2013) An improved evolutionary extreme learning machine based on particle swarm optimization. Neurocomputing 116:87–93
34. Hart WE, Krasnogor N, Smith JE (2004) Recent advances in memetic algorithms, volume 166. Springer, Berlin
35. Ho Y-C, Pepyne DL (2002) Simple explanation of the no-free-lunch theorem and its implications. J Optim Theory Appl 115(3):549–570
36. Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. Technometrics 12(1):55–67
37. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge
38. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Nat Acad Sci 79(8):2554–2558
39. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4(2):251–257
40. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366
41. Huang G, Huang G-B, Song S, You K (2015) Trends in extreme learning machines: a review. Neural Netw 61:32–48
42. Huang G-B (2015) What are extreme learning machines? Filling the gap between frank Rosenblatts dream and John von Neumanns puzzle. Cogn Comput 7(3):263–278
43. Huang G-B, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9(1):224–229
44. Huang G-B, Bai Z, Kasun LLC, Vong CM (2015) Local receptive fields based extreme learning machine. IEEE Comput Intell Mag 10(2):18–29
45. Huang G-B, Chen L, Siew CK et al (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
46. Guang-Bin H, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2(2):107–122
47. Huang G-B, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B Cybern 42(2):513–529
48. Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Neural networks, 2004. Proceedings of 2004 IEEE international joint conference on, vol 2. IEEE, pp 985–990
49. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501
50. Huang Q, Jiang C, Huang Y (2016) The prediction method of SO_2 concentration in sulfuric acid production process based on GA-ELM. In: Intelligent human-machine systems and cybernetics (IHMSC), 2016 8th international conference on, vol 2. IEEE, pp 140–143
51. Huang Y, Lai D (2012) Hidden node optimization for extreme learning machine. Aasri Procedia 3:375–380
52. Huynh HT, Won Y (2008) Evolutionary algorithm for training compact single hidden layer feedforward neural networks. In: International joint conference on neural networks (IJCNN). IEEE, pp 3028–3033
53. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department
54. Karlik B, Olgac AV (2011) Performance analysis of various activation functions in generalized MLP architectures of neural networks. Int J Artif Intell Expert Syst 1(4):111–122
55. Kasun LLC, Zhou H, Huang G-B, Vong CM (2013) Representational learning with elms for big data
56. Kaya Y, Kayci L, Tekin R, Ertuğrul ÖF (2014) Evaluation of texture features for automatic detecting butterfly species using extreme learning machine. J Exp Theor Artif Intell 26(2):267–281
57. Kirkpatrick S, Gelatt CD, Vecchi MP et al (1983) Optimization by simulated annealing. Science 220(4598):671–680
58. Kohonen T (1990) The self-organizing map. Proc IEEE 78(9):1464–1480

59. Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection, vol 1. MIT Press, Cambridge
60. Krasnogor N, Smith J (2005) A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans Evol Comput 9(5):474–488
61. Leshno M, Lin VY, Pinkus A, Schocken S (1993) Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw 6(6):861–867
62. Li B, Li Y, Rong X (2013) The extreme learning machine learning algorithm with tunable activation function. Neural Comput Appl 22(3–4):531–539
63. Li S, Wang P, Goel L (2015) Short-term load forecasting by wavelet transform and evolutionary extreme learning machine. Electr Power Syst Res 122:96–103
64. Li X, Shao Z, Qian J et al (2002) An optimizing method based on autonomous animats: fish-swarm algorithm. Syst Eng Theory Pract 22(11):32–38
65. Lin M, Zhang C, Su C (2016) Prediction of effluent from WWTPS using differential evolutionary extreme learning machines. In: Control conference (CCC), 2016 35th Chinese. IEEE, pp 2034–2038
66. Liu N, Wang H (2013) Evolutionary extreme learning machine and its application to image analysis. J Signal Proces Syst 73(1):73–81
67. Malhotra R, Singh N, Singh Y (2011) Genetic algorithms: concepts, design for optimization of process controllers. Comput Inf Sci 4(2):39
68. Matias T, Araújo R, Antunes CH, Gabriel D (2013) Genetically optimized extreme learning machine. In: Emerging technologies and factory automation (ETFA), 2013 IEEE 18th conference on. IEEE, pp 1–8
69. Matias T, Souza F, Arajo R, Antunes CH (2014) Learning of a single-hidden layer feedforward neural network using an optimized extreme learning machine. Neurocomputing 129:428–436
70. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100
71. Mohapatra P, Chakravarty S, Dash PK (2015) An improved cuckoo search based extreme learning machine for medical data classification. Swarm Evol Comput 24:25–49
72. Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical Report C3P Report 826, California Institute of Technology
73. Nahvi B, Habibi J, Mohammadi K, Shamshirband S, Razgan OSA (2016) Using self-adaptive evolutionary algorithm to improve the performance of an extreme learning machine for estimating soil temperature. Comput Electron Agric 124:150–160
74. Niu P, Ma Y, Li M, Yan S, Li G (2016) A kind of parameters self-adjusting extreme learning machine. Neural Process Lett 44(3):813–830
75. Ojha VK, Abraham A, Snášel V (2017) Metaheuristic design of feedforward neural networks: a review of two decades of research. Eng Appl Artif Intell 60:97–116
76. Salam MA, Hegazy O, Soliman OS (2015) Fpa-elm model for stock market prediction. Int J Adv Res Comput Sci Softw Eng 5(2):1050–1063
77. Pacifico LDS, Ludermir TB (2006) Evolutionary extreme learning machine based on particle swarm optimization and clustering strategies. In: International joint conference on neural networks (IJCNN). IEEE, pp 1–6
78. Pacifico LDS, Ludermir TB (2013) Evolutionary extreme learning machine based on particle swarm optimization and clustering strategies. In: Neural networks (IJCNN), the 2013 international joint conference on. IEEE, pp 1–6
79. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst 22(3):52–67
80. Qin-Yu Z, Qin AK, Suganthan PN, Huang G-B (2005) Evolutionary extreme learning machine. Pattern recognition 38(10):1759–1763
81. Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65(6):386
82. Rumelhart DE, Hinton GE, Williams RJ (1988) Neurocomputing: foundations of research chapter Learning Representations by back-propagating errors. MIT Press, Cambridge, pp 696–699
83. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning international representations by error propagation. In: Rumelhart DE, McCleland JL (eds) Parallel distributed processing: exploration in the microstructure of cognition, vol 1, Chap 8. MIT Press, Cambridge
84. Salam MA, Zawbaa HM, Emary E, Ghany KKA, Parv B (2016) A hybrid dragonfly algorithm with extreme learning machine for prediction. In: INnovations in Intelligent SysTems and Applications (INISTA), 2016 international symposium on. IEEE, pp 1–6
85. Sánchez-Monedero J, Hervas-Martinez C, Gutiérrez PA, Ruz MC, Moreno MCR, Cruz-Ramirez M (2010) Evaluating the performance of evolutionary extreme learning machines by a combination of sensitivity and accuracy measures. Neural Netw World 20(7):899
86. Sattar AMA, Erturul ÖF, Gharabaghi B, McBean EA, Cao J (2017) Extreme learning machine model for water network management. J Neural Comput Appl 2017:1–13
87. Schuh MA, Angryk RA, Sheppard JW (2012) Evolving kernel functions with particle swarms and genetic programming. In: FLAIRS conference
88. Schwefel H-P (1987) Collective phenomena in evolutionary systems
89. Silva DNG, Pacifico LDS, Ludermir TB (2011) An evolutionary extreme learning machine based on group search optimization. In: Congress on evolutionary computation (CEC). IEEE, pp 574–580
90. Storn R, Price K (1997) Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
91. Stützle TG (1999) Local search algorithms for combinatorial problems: analysis, improvements, and new applications, vol 220. Infix Sankt Augustin
92. Sun X, Qin L (2014) An extreme learning machine based on quantum particle swarm optimization and its application in handwritten numeral recognition. In: Software engineering and service science (ICSESS), 2014 5th IEEE international conference on. IEEE, pp 323–326
93. Sun Z-L, Choi T-M, Kin-Fan A, Yong Y (2008) Sales forecasting using extreme learning machine with applications in fashion retailing. Decis Support Syst 46(1):411–419
94. Suresh S, Babu RV, Kim HJ (2009) No-reference image quality assessment using modified extreme learning machine classifier. Appl Soft Comput 9(2):541–552
95. Tang J, Deng C, Huang G-B (2016) Extreme learning machine for multilayer perceptron. IEEE Trans Neural Netw Learn Syst 27(4):809–821
96. Voudouris C (1997) Guided local search for combinatorial optimisation problems. PhD Thesis, University of Essex
97. Wang J, Ye K, Cao J, Wang T, Xue A, Cheng Y, Yin C (2017) Doa estimation of excavation devices with elm and music-based hybrid algorithm. Cogn Comput 9:1–17
98. Wang Y, Cao F, Yuan Y (2011) A study on effectiveness of extreme learning machine. Neurocomputing 74(16):2483–2490
99. Werbos PJ (1974) Beyond regression: new tools for prediction and analysis in the behavioral science. Ph. D. Thesis, Harvard University

100. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
101. Xu X, Ji Z, Yuan F, Liu X (2014) A novel parallel approach of cuckoo search using mapreduce. In: 2014 international conference on computer, communications and information technology (CCIT 2014). Atlantis Press
102. Xue B, Ma X, Gu J, Li Y (2013) An improved extreme learning machine based on variable-length particle swarm optimization. In: International conference on robotics and biomimetics (ROBIO). IEEE, pp 1030–1035
103. Yang X-S (2010) Nature-inspired metaheuristic algorithms. Luniver Press, Bristol
104. You X, Shu Y (2006) Evolutionary extreme learning machine-based on particle swarm optimization. Adv Neural Netw ISNN 2006:644–652
105. Yang H, Yi J, Zhao J, Dong ZY (2013) Extreme learning machine based genetic algorithm and its application in power system economic dispatch. Neurocomputing 102:154–162
106. Yang Z, Wen X, Wang Z (2015) Qpso-elm: an evolutionary extreme learning machine based on quantum-behaved particle swarm optimization. In: International conference on advanced computational intelligence (ICACI). IEEE, pp 69–72
107. Yang Z, Zhang T, Zhang D (2016) A novel algorithm with differential evolution and coral reef optimization for extreme learning machine training. Cogn Neurodyn 10(1):73–83
108. Zhai J, Hong-yu X, Wang X (2012) Dynamic ensemble extreme learning machine based on sample entropy. Soft Comput 16(9):1493–1502
109. Zhai J, Zang L, Zhou Z (2018) Ensemble dropout extreme learning machine via fuzzy integral for data classification. Neurocomputing 275:1043–1052
110. Zhai J, Zhang S, Wang C (2017) The classification of imbalanced large data sets based on mapreduce and ensemble of elm classifiers. Int J Mach Learn Cybern 8(3):1009–1017
111. Zhang Y, Cai Z, Wu J, Wang X, Liu X (2015) A memetic algorithm based extreme learning machine for classification. In: Neural networks (IJCNN), 2015 international joint conference on neural networks. IEEE, pp 1–8