This article can be cited as O. A. Alade, R. Sallehuddin and N. H. M. Radzi, Enhancing Extreme Learning Machines Using Cross-Entropy Moth-Flame Optimization Algorithm, International Journal of Artificial Intelligence, vol. 20, no. 1, pp. 46-67, 2022.

Enhancing Extreme Learning Machines Using Cross-Entropy Moth-Flame Optimization Algorithm

Oyekale Abel Alade[1], Roselina Sallehuddin[2], Nor Haizan Mohamed Radzi[2]

[1]Computer Science Department, Federal Polytechnic,
912101, Bida Nigeria'
aaoyekale2@live.utm.my

[2]School of Computing, Universiti Teknologi Malaysia,
81310, Skudai, Johor, Malaysia;
roselina@utm.my, haizan@utm.my

ABSTRACT

Extreme Learning Machines (ELM) is a fast learning algorithm that eliminates the tuning of input parameters (weights and biases) of the hidden layer. However, ELM does not guarantee the optimal setting of the weights and biases due to random input parameters initialization. Therefore, ELM suffers from instability of output, large network size, and degrade generalization performance. To overcome these problems, an efficient co-evolutionary hybrid model namely as Cross-Entropy Moth-Flame Optimization (CEMFO-ELM) model is proposed to train a neural network for the selection of optimal input weights and biases. The hybrid model balanced the exploration and exploitation of the search space, and then selected optimal input weights and biases for ELM. The co-evolutionary algorithm reduced the chances of been trapped into the local extremum in the search space. Accuracy, stability, and percentage improvement ratio (PIR%) were the metrics used to evaluate the performance of the proposed model when simulated on some classification datasets for machine learning from the University of California, Irvine repository. The co-evolutionary scheme was compared with its constituent individual ELM-based enhanced meta-heuristic schemes (CE-ELM and MFO-ELM). The co-evolutionary meta-heuristic algorithm enhances the selection of optimal parameters for ELM. It improves the accuracy of ELM in all the simulations, and the stability of ELM was improved in all, up to 53% in Breast cancer simulation. Also, it has better convergences than the comparative ELM hybrid model in all the simulations.

Keywords: Cross-entropy, convergence, Extreme Learning Machines, meta-heuristic algorithms, Moth-flame Optimization, swarm intelligence optimization.

Mathematics Subject Classification: 68T20, 68T07

Computing Classification System: Computing methodologies, Machine learning, Machine learning approaches, Bio-inspired approaches, Artificial life

1. INTRODUCTION

Machine learning techniques have been used in recent times to solve classification tasks (Ahmed, Brickman, et al., 2019; Yi, Kong, et al., 2020). It is an approach where computers draw inferences from environmental data without explicit programming. As an artificial intelligence scheme, it automatically learns when exposing to new data (Breck, Polyzotis, et al., 2019) without being explicitly programmed (Liu, Wang, et al., 2017). Machine learning adjusts program behaviour relative to the patterns in the data it is exposed to. It is being used recently in some tasks like expert systems (Pozna and Precup, 2014), speech recognition, auto-drive cars, understanding the human genome,

effective web browsing, decision making, and many other (Albu, Precup, et al., 2019; Fong, Li, et al., 2020). Multilayer Feedforward Neural Network (MLFN) is popular in the implementation of Machine learning algorithms (Guliyev and Ismailov, 2018). However, Single Layer Feedforward Neural Network (SLFN) is proved in literature to have good learning ability with tolerable error (Qing, Zeng, et al., 2020).

There have been intensive studies on SLFN in recent times. Some of the algorithms are back-propagation algorithms (Aljarah, Faris, et al., 2018; Das, Kuhoo, et al., 2019; Duan, Li, et al., 2018): Radial Basis Function (RBF), Fourier series, wavelet networks (Lu, Qiu, et al., 2016), the Levenberg-Marquardt algorithm (Mikhaylov and Tarakanov, 2020), Artificial Neural Network (ANN) (Shahid, Rappon, et al., 2019). These SLFN classification algorithms have parameters whose values cannot be estimated directly from datasets (Saporetti, Duarte, et al., 2019). The weights and biases, for example, greatly influence the performance of classification algorithms. Most algorithms adopt the backpropagation (BP) method, in which the parameters have to be tuned to minimize error.

Huang *et al.*, (2004) proposed Extreme Learning Machines (ELM) to solve the parameter tuning challenges in SLFN. ELM randomly determines the initial parameters (weights and biases). The weights of the output layer are determined analytically using Moore Penrose's general inverse (pseudo-inverse) matrix. ELM learning principle is a linear model, where the input weights and biases are randomly initialized, without being further updated. Therefore, the weights and biases no longer require iterative tuning as they were in the conventional feedforward networks (Chen, Kloft, et al., 2018). ELM is embraced in research, and it has been applied in many areas (Alade, Selamat, et al., 2018; Toprak, 2018). However, the acceptance of ELM and its wide usage has opened up some gaps for several issues. The hidden layer of ELM requires a higher number of neurons (Eshtay, Faris, et al., 2020; Faris, Mirjalili, et al., 2020b).

More so, the random assignment of input parameters challenges the stability of ELM. Over-parameterized design of ELM usually results in an ill-conditioning problem when it is updated with recursive least squares during the training phase (Zhang, Yang, et al., 2018), and may eventually lower the degree of its accuracy. Therefore, the outputs from ELM are usually sensitive to data perturbation and thereby unstable (Eshtay, Faris, et al., 2018b). This may prove the output of ELM to be a poor estimation of the truth, which may lead to a wrong decision. This may affect its deployment for some time-sensitive applications. Filling these gaps have been responded to in many ways by researchers.

There are many variants of ELM in literature aiming at ensuring an optimal number of neurons in the hidden layer. They are Incremental-ELM (Huang, Li, et al., 2008), Online-sequential ELM (Nan-Ying Liang, Guang-Bin Huang, et al., 2006), pruning-ELM (Rong, Ong, et al., 2008), Voting-based ELM (Huang, 2008), two-stage ELM (Zhao, Wang, et al., 2012), Ensemble ELM (Albadr and Tiun, 2017) and many more. Multiple tests of errors comparisons are required to determine the optimal number of hidden neurons. These approaches are time-consuming and they require larger randomness (Z. Tian, Ren, et al., 2019). More so, they could not adequately solve the challenge of how the optimal number of the neuron is determined.

Wang, Cao, et al., (2011), proposed an Efficient Extreme Learning Machine (EELM) as a high-quality feature mapping algorithm. EELM properly selects input weights and biases and then calculates the output weights. Its focus was to ensure that the column of the hidden neuron output *H* is fully ranked. The work improved learning rates and ensured a robust network structure. Janakiraman *et al.* (2016) proposed a stochastic gradient. Their approach incorporated a notion of stable learning that preserves the simplicity and generalization of standard ELM. The results' evaluation employed the Lyapunov approach for error measure and the boundedness of the learning rates. The results over-fit, tend to fall into a local minimum, and the accuracy needs improvement.

Optimization techniques have greatly contributed to the enhancement of base learner algorithms such as ELM and other machine learning algorithms. Precup, Hedrea, et al., (2021) discussed artificial intelligence techniques in terms of nature-inspired optimization algorithms and neural networks systems engineering. Precup, David, et al., (2021) proposed an approach to the SMA-based tuning of cost-effective fuzzy controllers for servo systems. The validation of the work showed its superiority over some other meta-heuristic algorithms. To improve the computational time, self-assembly algorithms, Zapata, Perozo, et al., (2020) employed a co-evolutionary scheme of a classic self-assembly algorithm (CSA) and particle swarm optimization algorithm (PSO). This approach balanced the exploration and the exploitation of the search space. This literature is proofs that meta-heuristic optimization technics can be adapted to the selection of parameters of ELM to improve its classification performance.

Metaheuristic algorithms (MAs) combine randomness and optimization of the search, to guide the search process to ensure efficiency in the search space and provide optimal solutions for hard optimization problems. The operating principle of most MA algorithms is based on either bionic (bio-inspired) or physical phenomena. The bio-inspired optimization techniques are a better MA option to optimize the parameters of ELM. Li, Shuang, et al., (2019) used MA to guide the exploration and exploitation search process. The MA is problem-independent and can provide a good solution to both simple and complex tasks. Recently, Eshtay et al., (2018a) observed that there is improvement in the performance of ELM when it is hybrid with meta-heuristic optimization algorithms to select input weights and biases. Alshamiri, Singh, et al., (2018) proposed a two swarm intelligence based metaheuristic techniques with Artificial Bee Colony (ABC) and Invasive Weed Optimization (IWO). The hybrid algorithm tuned the input weights and hidden biases. They tested the proposed on different benchmark classification data sets. The simulations showed a good generalization performance in comparison to other techniques.

Most MA optimization techniques used to enhance ELM are population-based algorithms. Some of them are Particle Swarm Optimized ELM (PSO-ELM) (Vidhya and Kamaraj, 2017), Genetic Algorithm ELM (GA-ELM) (Zhou, Zhou, et al., 2020), Cuckoo Search Optimization ELM (CSO-ELM) (R. Wang, Li, et al., 2018), Fire-flight algorithm (FA) (Li, Liu, et al., 2019), Bat Swarm Optimization (BSO) (Alihodzic, Tuba, et al., 2017), Artificial Immune Systems ELM (AIS-ELM) (H. Tian, Li, et al., 2018), Artificial Bee Colony (ABC-ELM) (Yang Wang, Wang, et al., 2017), Differential Evolution (DE-ELM) (Saporetti, 2019), Improved Grey Wolf Optimization (IGWO) model (Cai, Gu, et al., 2019), etc. Despite

the relative success of these metaheuristics approaches in terms of flexibility and efficiency towards solution finding, they continue to suffer slow convergence and often get stuck at the local optima. These issues challenge the accuracy of ELM.

This research work proposes a co-evolutionary scheme of two stochastic algorithms to address the issue of parameters selection for ELM. The co-evolutionary algorithm employs a hybrid of Cross-Entropy (CE) and Moth-Flame Optimization (MFO) schemes. The two schemes are from different backgrounds. CE is a stochastic algorithm that has its root from a physical phenomenon, while MFO is based on swarm intelligent optimization technique.

The rest of this paper follows thus: Section 2 explains the ELM classification algorithm. We discussed the two (2) meta-heuristic schemes used in this study in fair detail in Section 3. The description of the hybrid scheme of the CEMFO-ELM algorithm is presented in Section 4. Sections 5 and 6 present implementation and the discussion of results respectively, and in Section 7, we conclude the paper.

2. EXTREME LEARNING MACHINE

ELM aimed at training SLFN for the smallest norm of output weights (β) (Huang, 2004). This proposition was for the good generalization performance of feedforward neural networks. ELM randomly generates the initial weights and biases by a continuous piecewise probability function. Moore Penrose inverse matrix (H) was used to determine the output weight of ELM analytically. This helps the algorithm to remove the problem of long training phase common to other gradient descent algorithms and minimizes norm among the least square solution.

For an ELM with an input dataset X of N instances, each instance has d -dimensional feature and belongs to 1 or m classes in the set, we represented the dataset as $\left(x_k, y_k\right) \in X, k = 1,..., N$ x_k is the input vector $x_k \in R^n$, and y_k is the expected result $y_k \in R^m$. An SLFN of size L , training vectors of $\left[X_k = x_{k,1}, x_{k,2},..., x_{k,n}\right]^T$, $k = 1,..., N$, an activation function $g(x)$ and an output vector $Y_k = \left[y_{k,1},..., y_{k,m}\right]^T$ are mathematically modelled in terms of:

$$\mathrm{y}_k = \sum_{k=1}^{L} \beta_j g\left(w_j . x_k + b_j\right), \quad k \in [1, N] \tag{1}$$

where $w_j = \left[w_{j,1}, w_{j,2},..., w_{j,n}\right]^T$ is a weight vector connecting the k^{th} input node to j^{th} neuron, and b_j bias of j^{th} hidden neuron. $\beta_j = \left[\beta_{j,1}, \beta_{j,2},..., \beta_{j,n}\right]^T$ is the output weight vector of connecting the hidden neurons and the m output neurons, $w_j . x_k$ is the inner product of w_j and x_k . Equation **Error! Reference source not found.** is compactly presented as follows for N system of equations:

$$Y = H\beta \tag{2}$$

where $H, \beta,$ and Y are:

$$H\left(w_1, \ldots, w_L, b_1, \ldots, b_L\right) = \begin{bmatrix} g\left(w_1 x_1 + b_1\right) & \cdots & g\left(w_L x_1 + b_L\right) \\ \vdots & \ddots & \vdots \\ g\left(w_1 x_N + b_1\right) & \cdots & g\left(w_L x_N + b_L\right) \end{bmatrix}_{N \times L} \tag{3}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_N^T \end{bmatrix} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1L} \\ \vdots & \ddots & \vdots \\ \beta_{N1} & \cdots & \beta_{NL} \end{bmatrix}_{N \times L} \tag{4}$$

$$\tilde{Y} = \begin{bmatrix} \tilde{y}_1^T \\ \vdots \\ \tilde{y}_N^T \end{bmatrix} = \begin{bmatrix} \tilde{y}_{11} & & \tilde{y}_{1m} \\ \vdots & \cdots & \vdots \\ \tilde{y}_{N1} & & \tilde{y}_{Nm} \end{bmatrix}_{N \times m} \tag{5}$$

To calculate β analytically, the minimum norm least-square solution is found:

$$\hat{\beta} = \min_{\beta} \boldsymbol{H}\beta - \boldsymbol{Y} = \boldsymbol{H}^{\dagger} Y \tag{6}$$

$$\boldsymbol{H}^{\dagger} = (\boldsymbol{H}^T \mathbf{H})^{-1} \mathbf{H}^T \tag{7}$$

$\boldsymbol{H}^{\dagger}$ is the pseudo inverse matrix of the hidden layer output $\boldsymbol{H}$. Figure 1 shows the structure of ELM.

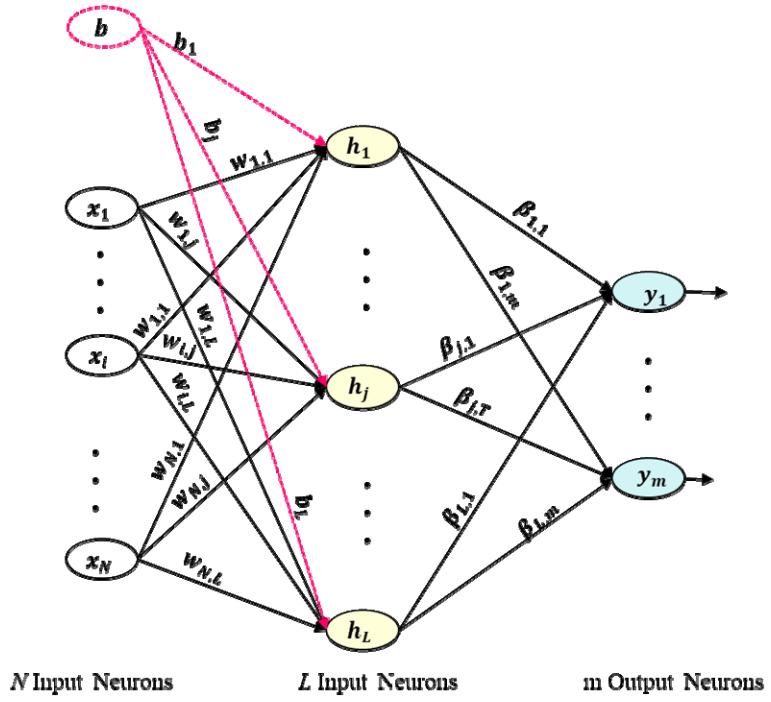


Figure 1 The structure of Extreme Learning Machines

3. META-HEURISTIC ALGORITHMS

A metaheuristic is a "master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality" (Glover, 1989). More so, all MAs involve a compromise of randomization and local search. That is, they generate particles randomly, e.g. using

$$X_{kj} = r \otimes \left(ub_j - lb_j\right) + lb_j \tag{8}$$

and apply a function as a search operator for exploitation depending on the optimization algorithm being used, where x_{kj} is moth k^{th} in feature location j^{th} of the search space, $k = 1,...,n$ $j = 1,...,d$; lb and ub are minimum and maximum limits of particles' position in the space, $\otimes$ is the Hadamard product. Meta-heuristics models attempt to ensure quality solutions to hard optimization tasks at a reasonable time. The two meta-heuristics considered in the paper are discussed below.

3.1. Cross-Entropy Optimization Scheme

CE scheme is a minimization scheme based on Kullback–Leibler model (Botev, Kroese, et al., 2013). It is an adaptive importance sampling technique, therefore, we used it as a stochastic algorithm for optimizations. CE can solve several complex estimation and optimization problems. It is good for the exploration of the search space. Therefore, in this study, we used it to enhance the exploration of a population-based optimization algorithm. The CE optimization model is expressed in

$$\min\left(S\left(x\right): X \in \mathrm{R} \wedge \boldsymbol{n} \to \boldsymbol{R}\right) \tag{9}$$

CE operator enhances the diversification of particles (guesses) in the search space. It generates particles using

$$x_i = \mu + \sigma \otimes rnd() \tag{10}$$

where $rnd()$ is a random number generator that produces a Gaussian distribution. Then it uses two parameters to update the positions of the particles, which are the mean μ and the standard deviation σ . The mean attempts to seek the points with the best solutions, while the standard deviation continues to reduce until the parameters' focus is the region of the best solution. The parameters are then updated with

$$M = \left(1 - lr\right) \otimes \mu + lr \otimes samples \tag{11}$$

$$S = (1 - lr) \otimes \sigma + lr \otimes samples \tag{12}$$

where $M = \left(\mu_1, \mu_2, ..., \mu_n\right)$ and $S = \left(\sigma_1, \sigma_2, ..., \sigma_n\right)$ $i = 1, 2, ..., n$. Learning rate lr performs the variation in each iteration in equations (11) and (12) for these operations.

3.2. Moth-Flame Optimization Scheme

Moth-Flame Optimisation (MFO) is a recent swarm-based intelligent MA (Mirjalili, 2015). The particles of MFO are randomly initialized as moths. Each moth represents a candidate solution, and the positions of moths are the features in the search domain. The moth positions are the parameters of ELM to be optimized. These positions of the moth are initialized randomly within [-1 and 1] (Mirjalili, 2015; Sayed, Darwish, et al., 2019). The structure of MFO is modelled in two parts – the moth and the flame. The moth positions are initialized randomly using the equation (8). The encoding scheme in Figure 2 defines the dimension of the moth.

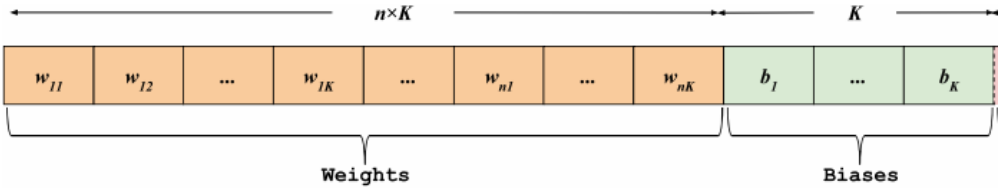


Figure 2 MFO individual encoding scheme

The structure of the individual moth encoding scheme is composed of *n* inputs (the dimension of the input data) and the K hidden nodes. L is the length of each candidate solution in the MFO search space. *d* is calculated with

$$d = n \times K + K \tag{13}$$

MFO initializes individual particles in the swarm with L dimension.

The fitness values represent the current best position of each moth, and it is calculated with

$$OM = \min(1 - Accuracy) \tag{14}$$

The moth positions are represented in the matrix *M* and the corresponding fitness function for each moth *OM* in

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,d} \\ m_{2,1} & \ddots & \ddots & m_{2,d} \\ \vdots & \ddots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,d} \end{bmatrix}, \quad OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \tag{15}$$

n is the moths' population, and d is the dimension.

The position vector of each moth in the search space is regulated by a flag operator to ensure optimal fitness values. The fitness values are calculated using a fitness function. *OM* in equation (15) is a vector that contains the corresponding fitness values of each moth.

A similar matrix to the moths' (F) is constructed for the flame as follows:

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \cdots & F_{1,d} \\ F_{2,1} & \ddots & \ddots & F_{2,d} \\ \vdots & \ddots & \ddots & \vdots \\ F_{n,1} & F_{n,2} & \cdots & F_{n,d} \end{bmatrix}, \quad OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix} \tag{16}$$

n is the flames' population (the same as moths' population). *OF* contains the corresponding fitness values for each flame – equation (16). The flames are the best moth positions attained so far. Both the moths and the flames are candidate solutions in the search space. The moths (the search agents) move around the flames. The moths search around the flames and update accordingly if there is a better solution. The process continues until it reaches the maximum set iteration.

The effectiveness of the algorithm strongly depends on the distance D between the moths and the flame. D is calculated in terms of

$$D_i = \left| F_j - M_i \right| \tag{17}$$

where M_i is the i^{th} moth, F_i the j^{th} flame, and D_i is the distance between the i^{th} moth and the j^{th} flame. Moths update mechanism uses a logarithmic spiral in

$$S\left(M_i, F_i\right) = D_i e^{bt} \operatorname{Cos}\left(2\pi t\right) + F_j \tag{18}$$

Equation (18) meets the standard conditions set in literature for moth update thus: (i) the spiral should begin from a moth, (ii) a flame's position should be the terminal point of the spiral, (iii) Fluctuation of the spiral should be within the search space (Mirjalili, 2015).

S function model controls the flying of a moth around a flame. D_j indicates the Euclidian distance between i^{th} moth and j^{th} flame, b is a parameter that defines the shape of the spiral model, it decreases linearly from -1 to -2 with iteration. It determines the convergence of the algorithm. t parameter is randomly generated between [-1,1] which specifies how close the next moth position should be to the flame. $2\pi t$ in the equation is the distance between successive turns of the spirals.

The moths' position update relative to *n* locations in the space can degrade the exploitation of the candidate solution. Therefore, the following adaptive model is employed to determine the number of flames:

$$flameNumber = round\left(N - l \times \frac{N-1}{T} \right) \tag{19}$$

N is the flames' population, l – the current generation, and T – maximum number of generations. In the last generation, the moths' position update is computed concerning the best flame. The best moth is the optimum approximation returned after the simulation. The decrement in the flames is to maintain a balance between exploration and exploitation space.

4. HYBRID CEMFO-ELM

We proposed a hybrid model of Cross-Entropy (CE) and Moth-Flame Optimization (MFO) algorithms to balance the exploration and exploitation of the search process and ensure the selection of optimal input weights and biases. The hybrid model is named Cross-Entropy Moth-Flame Optimization Based Extreme Learning Machine (CEMFO-ELM). The implementation proposed a fusion of CE and MFO operators. As a co-evolutionary technique, CEMFO-ELM preserves the global optimization capability of CE for fast convergence advantage, and the good exploitation of MFO algorithm in a local search space. The two algorithms complement their respective advantages. Hence, it selects and presents optimal parameters of weights and biases needed to enhance the ELM performance. The algorithm of the hybrid model is shown in Algorithm 1.

Algorithm 1: Cross Entropy Moth-Flame Optimization Algorithm

1: Initialize parameters dimension *d*, lower and upper bound lb & ub, learning rate *lr, k*_sample, generation iterMax
2: Generate random samples using equation (8)
3: Compute fitness with equation (14)
4: Calculate means and deviations (11) and (12)
5: Sort guesses X
6: Select the initial best as the $k_{\_sample}$
7: While loop
8: Generate new guesses with the mean and standard deviation and all the parameters equation **Error! Reference source not found.**
9: Calculate the fitness with equation (14)
10: Calculate the new mean and deviation with the CE parameters (11) & (12)
11: Sort the new guesses SX
12: Calculate the distance of Moths X X to Flame SX with equation (17)
13: Update X with spiral equation (18)
14: Determine the best
15: End while

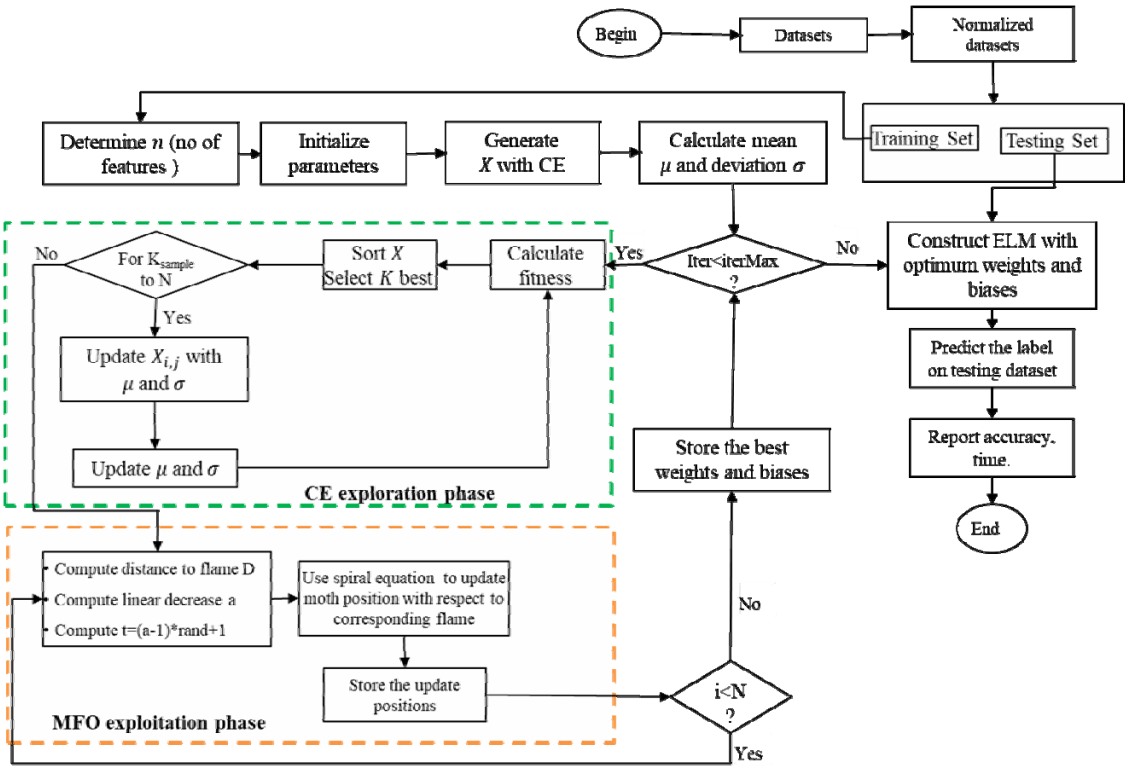


Figure 3. The flowchart of the CEMFO-ELM algorithm

Figure 3 is the flowchart of the proposed hybrid model. The green region of the diagram represents the CE exploration phase, while the part marked with orange colour shows the MFO exploitation phase

.

5. METHOD OF IMPLEMENTATION

There are two main functions in a meta-heuristic algorithm – the exploration and the exploitation functions. These functions should be balanced to ensure the efficient performance of a meta-heuristic algorithm. The essence of training a neural network is to choose the best parameters for the neural network. The popular parameters of ELM are weights and biases. As shown in the Algorithm I above, optimization was used to training and select the best of these particles that carry the optimal parameters (input weights *w* and biases *b*). The subsections below explain the hybridization process, tools, data preparation, and the conditions of implementation.

5.1. Exploration and Exploitation Processes of CEMFO-ELM

The optimization problem is to select optimal input weights *w* and biases *b* for ELM neural network. In the exploration phase, the proposed CEMFO-ELM algorithm used the CE operator to generate guesses using equation **Error! Reference source not found.** as shown in Algorithm 1. The fitness value of each particle is calculated with equation (14). The particles are arranged in descending order

of the fitness values. The best CE $k_{\_sample}$ $k_{\_samples}$ is selected. The mean and standard deviation are computed using equations (11) and (12) respectively. These are used in equation **Error! Reference source not found.** to generate new guesses to enhance the exploration. The learning rate *lr* in equations (11) and (12) varies the parameters (means and standard deviation) of each iteration.

In the exploitation phase, the MFO receives the first matrix as moths and the new guesses as moths from CE – equations (16) and (16) respectively. The MFO operator considered the distance of each moth from the flame in equation (18), it then used the distance in the spiral equation (18) to determine the next position in the search space. This mechanism promotes the exploitation, and attempt to avoid local minimal. The best solution so far is stored as flame. In the end, the best candidate solution is reshaped into input weights w and hidden neuron biases b.

In the testing phase, the weights w and biases b returned from the training are used to construct an ELM classifier. The classifier is applied to the normalized testing datasets and predicts the labels. The algorithm's performance was measured by accuracy, network size, and stability as discussed below.

Different network sizes ranging from 5 to 50 neurons (at an interval of 5) have been experimented with. However, we maintained the sizes of the optimal networks for the four (4) algorithms (CEMFO-ELM, CE-ELM, MFO-ELM, and ELM).

5.2. Implementation Tools

The implementation of the proposed CEMFO-ELM used classification datasets selected from the UCI repository. In carrying out these experiments, the tasks are coded using MATLAB release 18a software. The code was executed on hardware with an Intel® Core™ i5-5200U CPU@3.60GHz, the x64-based processor with installed Memory (RAM) of 16.0GB, 2000 GB(2TB) SATA-3G HDD, running on Window 10 (64-bit) operating system.

5.3. Dataset description

Table 1 describes the summary of the datasets.

Table 1*:* Summary of classification datasets

Dataset	#Samples	#Train Instances	#Test Instances	#Features
Blood	748	493	255	4
Breast	699	461	238	9
Diabetes	768	507	261	8
Liver	345	228	117	6
Phoneme	5404	3567	1837	5

Five (5) classification datasets from the medical domain were employed in this study. Four (4) datasets are selected from the UCI repository (Lichman, 2013) and one (1) from a datahub website.

The dataset can be accessed with the link given in section 5.4. We ensure that the datasets are devoid of missing datapoint. The datasets with missing data points were imputed using the Predictive Mean Matching (PMM) imputation technique (Alade, Sallehuddin, et al., 2019). This is because ELM is not robust to missing data points and outliers. The study used binary datasets of various sizes to ensure effective evaluation of the algorithm. The attributes range from 4 (for Blood dataset) to 9 (for each of Breast and Diabetes datasets), and the sample size ranges from few hundreds (for Liver dataset) to some thousands (for Phoneme dataset).

The links to the codes and datasets respectively are given below:
https://cutt.ly/mycodes
https://cutt.ly/mydatasets

5.4. Experimental Condition

The datasets are partitioned into ratios 70:30 for training and testing respectively. The attributes in the datasets are normalized in a range of [0, 1]. This is done to put the attributes on the same scale and thereby prevent the results from been skewed towards attributes with high values. The linear transformation model in equation (20) was used for the normalization of the datasets (Eshtay, 2018a)

$$y = \left(\frac{x - min}{max - min} \right) \square \ (newMax - newMin) + newMin \tag{20}$$

where x is a data point, *min* and *max* are the feature vectors' minimum and maximum values, and *newMin* and *newMax* are the new features' minimum and maximum values after transformation.

Table 2 shows the parameter settings for the implementation of the algorithm.

Table 2: Parameter settings for the experiments

Algorithm	Parameter	Value
MFO	Moth size (N) Maximum iteration (iterMax) Uniform random number Convergence constant (a) Shape constant (*b*) Next moth-to-flame position (t)	100 100 -1 to 1 -1 to -2 1 -1 to 1
CE	Guess Maximum iteration K_{sample} Learning rate Uniform random number	100 100 2 0.7 -1 to 1
CEMFO-ELM	Maximum iteration Number of particles Uniform random number Convergence constant (*a*) Shape constant ($_b$) K_{sample} Learning rate	100 100 -1 to 1 -1 to -2 1 2% 0.7

In the training phase of the implementation, 100 particles were randomly generated. Each particle contains the weights and biases to be optimized. The particles are within -1 to 1 using equation **Error! Reference source not found.** (Al-Betar, Awadallah, et al., 2019). Each simulation is terminated at the end of 100 iterations. The parameters settings of CE are based on (Cui, Zhai, et al., 2016), and for MFO are based on (Mirjalili, 2015).

5.5. Performance Evaluation

The effectiveness of the ELM and the proposed hybrid model are evaluated using classification network size, stability, accuracy, and execution time. The metrics are well established and widely used in literature (Eshtay, 2018a; Nahato, Nehemiah, et al., 2016; Su and Cai, 2016). In addition, another method to evaluate the stability of the proposed algorithms is the performance improvement rate (PIR%) (Gabi, Ismail, et al., 2018). The performance metrics used in this research are discussed below.

Accuracy (also known as classification rate) is a measure of the number of successful hits relative to the total number of classifications. It shows the predictive power of classification algorithms. It is by far the most commonly used metric for assessing the performance of classifiers over the years. The expression of accuracy is

$$Accuracy = \frac{number\ of\ correctly\ classified\ samples}{total\ number\ of\ training\ samples} \tag{21}$$

The stability of the machine learning algorithms is measured with a standard deviation of the accuracy of the testing datasets. We adopted the stability measure in (Faris, Mirjalili, et al., 2020a):

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}} \tag{22}$$

where σ is the standard deviation, x_i is the accuracy of i^{th} prediction is the mean prediction value, and N is the total number of training samples. When the value σ is small, the algorithm is stable and vice versa.

The Performance Improvement Rate (PIR%) measures the improvement of the optimized algorithms (Gabi, 2018) on ELM. The measure helps to discover the efficiency of the proposed algorithms in improving the accuracy and stability of the basic ELM. PIR% can be evaluated with

$$PIR\% = \frac{A_s - B_s}{B_s} \times 100 \tag{23}$$

where A_s and B_s are the scores of two comparative algorithms, A_s is the proposed algorithm while B_s is the benchmark algorithm.

6. DISCUSSION OF RESULTS

We compared CEMFO-ELM with the implementation of the two constituent CE-ELM, MFO-ELM, and ELM algorithms in this section. A detailed description of the results is discussed below.

6.1. Accuracy and Network Size

Accuracy and network size are used to evaluate the improvement of the hybrid model on the comparative algorithms. This section describes the results of the hybrid of the Cross-Entropy and Moth-Flame Optimization algorithm. A performance measure of the CEMFO-ELM, CE-ELM, MFO-ELM and traditional ELM algorithms based on the 5 datasets are reported in Figure 4a-e. The figure shows the mean classification accuracy for thirty (30) independent simulations, and the network sizes used to attain the classification accuracy. Generally, there is an improvement in the classification accuracy of ELM in all the simulations. The improvements in accuracy and the network size on each dataset are discussed below.

For the Blood dataset, Figure 4a shows that CEMFO-ELM improves ELM for all the SLFNs constructed. CEMFO-ELM used an SLFN training structure with 10 hidden neurons to reach the best accuracy of 81.31%; whereas, it takes CE-ELM and MFO-ELM 20 and 15 nodes respectively to reach their best performances of 81.15% and 80.69%. It outperforms CE-ELM in 90% of the entire simulations, that is, in all except 20 neurons. It is better than MFO-ELM in 60% of the simulations.

CEMFO-ELM in Figure 4b, improves ELM classification accuracy for the Breast cancer dataset. It uses 10 hidden neurons to reach the best accuracy of 98.99%; whereas it takes CE-ELM and MFO-ELM 20 and 10 neurons respectively to reach their best accuracies of 97.73% and 98.16%. It outperforms CE-ELM in 90% of the entire simulations. It is better than MFO-ELM in 70% of the experiments. Therefore, CEMFO-ELM requires less number of neurons to reach its best accuracy, which is the highest of all.

In Figure 4c, CEMFO-ELM requires 15 neurons to reach the best accuracy of 79.97%. CE-ELM, MFO-ELM, and ELM also use 15 neurons each to reach their best accuracies of 77.81%, 78.52%, and 77.87% respectively. All the meta-heuristic algorithms enhanced the accuracy of the basic ELM in the simulations on Diabetes dataset. In 70% of the experiments, CEMFO-ELM improves ELM classification accuracies. It performs better compared to CE-ELM and MFO-ELM in all the simulations. On the Bupa liver dataset, the simulation results show that CEMFO-ELM, CE-ELM, and MFO-ELM algorithms reached their best accuracies with 10 neurons, but 15 neurons for ELM (Figure 4d). CEMFO-ELM has the best accuracy of 78.22% to prove its superiority over CE-ELM, MFO-ELM with 73.62%, and 76.29% accuracies respectively, though they require the same network size.

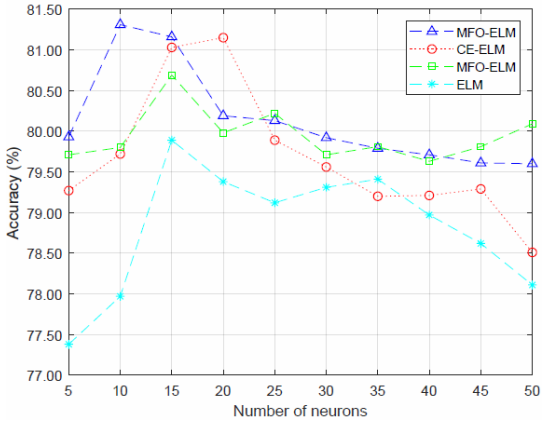
The network structure of all the algorithms is complex in the simulation of the Phoneme dataset (Figure 4e). CEMFO-ELM has the best accuracy of 83.52% with the 50 neurons. It performed better than CE-ELM in 60%, but its accuracy is less than the MFO-ELM in 90%. Although MFO-ELM has the best network size of 40 neurons, its accuracy is less than CEMFO-ELM and CE-ELM. All the algorithms reach their best classification accuracies with 50 neurons except MFO-ELM that attains its best with 40 neurons. The three metaheuristic algorithms improve the accuracy of ELM classifications in all the runs.

From the results above, CEMFO-ELM has the best accuracies in all the simulations on the five datasets. It also has the best network size in the Blood dataset, a tie with MFO-ELM and ELM in the Breast cancer dataset; a tie with CE-ELM, MFO-ELM, and ELM on Diabetes dataset, a tie with CE-ELM, MFO-ELM on Bupa Liver dataset, and competitive network size on Phoneme dataset's simulations. Therefore, CEMFO-ELM improves ELM accuracies better than the two other meta-heuristic algorithms with competitive network sizes.

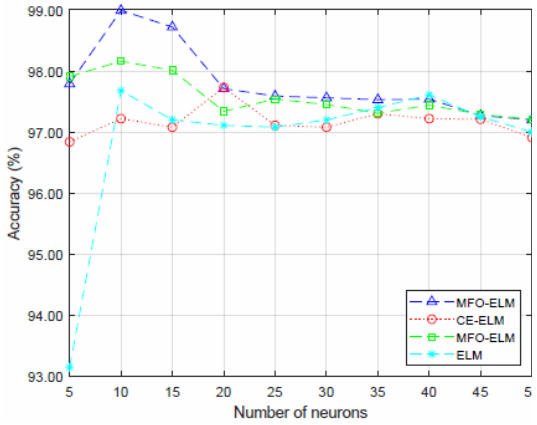
On the Bupa liver dataset, the simulation results show that CEMFO-ELM, CE-ELM, and MFO-ELM algorithms reached their best accuracies with 10 neurons, but 15 neurons for ELM (Figure 4d). CEMFO-ELM has the best accuracy of 78.22% to prove its superiority over CE-ELM, MFO-ELM with 73.62%, and 76.29% accuracies respectively, though they require the same network size.

The network structure of all the algorithms is complex in the simulation of the Phoneme dataset (Figure 4e). CEMFO-ELM has the best accuracy of 83.52% with the 50 neurons. Although MFO-ELM has the best network size of 40 neurons, its accuracy is less than CEMFO-ELM and CE-ELM. The three metaheuristic algorithms improve the accuracy of ELM classifications in all the runs.

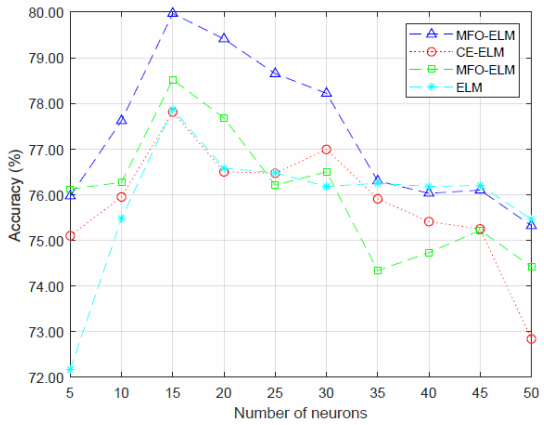
From the results, CEMFO-ELM has the best accuracies in all the simulations on the five datasets. It also has the best network size in the Blood dataset, a tie with MFO-ELM and ELM in the Breast cancer dataset; a tie with CE-ELM, MFO-ELM, and ELM on Diabetes dataset, a tie with CE-ELM, MFO-ELM on Bupa Liver dataset, and competitive network size on Phoneme dataset's simulations. Therefore, CEMFO-ELM improves ELM accuracies better than the two individual meta-heuristic algorithms with competitive network sizes.



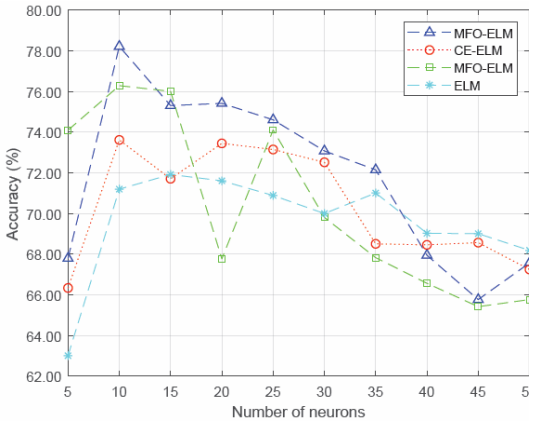
(a) Blood dataset



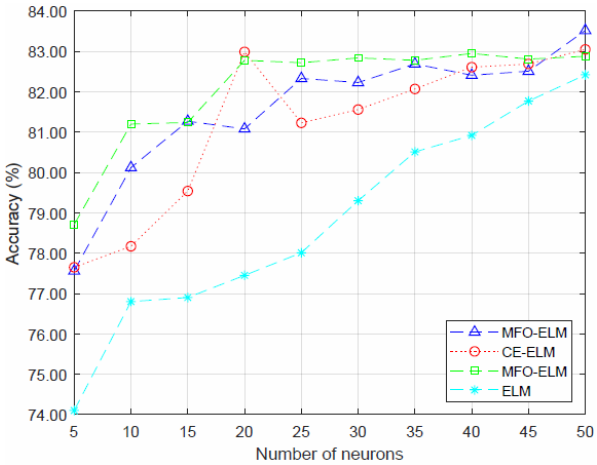
(b) Breast cancer dataset



(d) Diabetes dataset



(e) Bupa liver dataset



(e) Phoneme dataset

Figure 4. Comparison of accuracy of CEMFO-ELM, CE-ELM, MFO-ELM, and ELM algorithms

6.2. Stability

Stability defines the error minimization of the proposed hybrid model. This section discusses the enhancement of the stability of ELM. The standard deviations from the average accuracies are the metric that measures of the stability of the algorithms. The standard deviations of thirty (30) independent runs are observed, and the average was computed for each SLFN structure. The mean standard deviation for each algorithm on each dataset is used to investigate the stability of the algorithm. The algorithm with the least deviation has the best stability.

The average stability measure of the four (4) algorithms is in *Table 3*. The best comparative stability measure of the algorithms for the five datasets is bold in the table. CEMFO-ELM has the least mean deviation measure for all the simulations.

Table 3: Stability measure of CEMFO-ELM, CE-ELM, MFO-ELM, and ELM

Dataset	CEMFO-ELM	CE-ELM	MFO-ELM	ELM

Blood	**0.0043**	0.0248	0.0064	0.0085
Breast	**0.0037**	0.0063	0.0052	0.0080
Diabetes	**0.0078**	0.0137	0.0122	0.0122
Liver	**0.0118**	0.0233	0.0190	0.0246
Phoneme	**0.0049**	0.0066	0.0068	0.0097

From Table 3, we computed the percentage rate of improvement (PIR) of each meta-heuristic algorithm on ELM using the equation (23). The computation of the PIR% on the Blood dataset is illustrated with these examples:

CEMFO vs ELM: $$PIR\% = \left(\frac{0.0043 - 0.0085}{0.0085} * 100 \right) = -49.41\%$$

CE vs ELM: $$PIR\% = \left(\frac{0.0248 - 0.0085}{0.0085} * 100 \right) = -43.53\%$$

MFO vs ELM: $$PIR\% = \left(\frac{0.0064 - 0.0085}{0.0085} * 100 \right) = -24.71\%$$

Table 4: Summary of PIR% of each meta-heuristic algorithm on ELM

	ELM	**CE-ELM**		**MFO-ELM**		**CEMFO-ELM**	
Dataset	**Stability**	**Stability**	**PIR%**	**Stability**	**PIR%**	**Stability**	**PIR%**
Blood	0.0085	0.0248	-43.5294	0.0064	-24.7059	0.0043	**-49.4118**
Breast	0.0080	0.0063	-21.2500	0.0052	-35.0000	0.0037	**-53.7500**
Diabetes	0.0122	0.0137	12.2951	0.0122	0.0000	0.0078	**-36.0656**
Liver	0.0246	0.0233	-5.2846	0.0190	-22.7642	0.0118	**-52.0325**
Phoneme	0.0097	0.0066	-31.9588	0.0068	-29.8969	0.0049	**-49.4845**

$$PIR\% = \left(\frac{0.0064 - 0.0085}{0.0085} * 100 \right) = -24.71\%$$

The percentage PIR of CEMFO-ELM, MFO-ELM, and CE-ELM over ELM, are shown in Table 4. The results quantitatively show the improvements of the meta-heuristic algorithms on the stability of ELM, and the superior performance of CEMFO-ELM over the other ELM enhanced algorithms.

Furthermore, we also employed Boxplot as a quantitative approach to evaluate the stability of CEMFO-ELM further. The compact a Boxplot is, the more stable the algorithm (Wang, Li, Wang, & Gao, 2018). Figure 5a-e presents the boxplots of CEMFO-ELM, CE-ELM, MFO-ELM, and ELM for the datasets. The Boxplots represent the distribution of the average accuracy over 30 independent runs on each dataset. From the figure, the Boxplot for CEMFO-ELM is more compact than CE-ELM and MFO-ELM. More so, the parts outside the quartiles are less than the comparative algorithms, which shows that its accuracy is more stable.

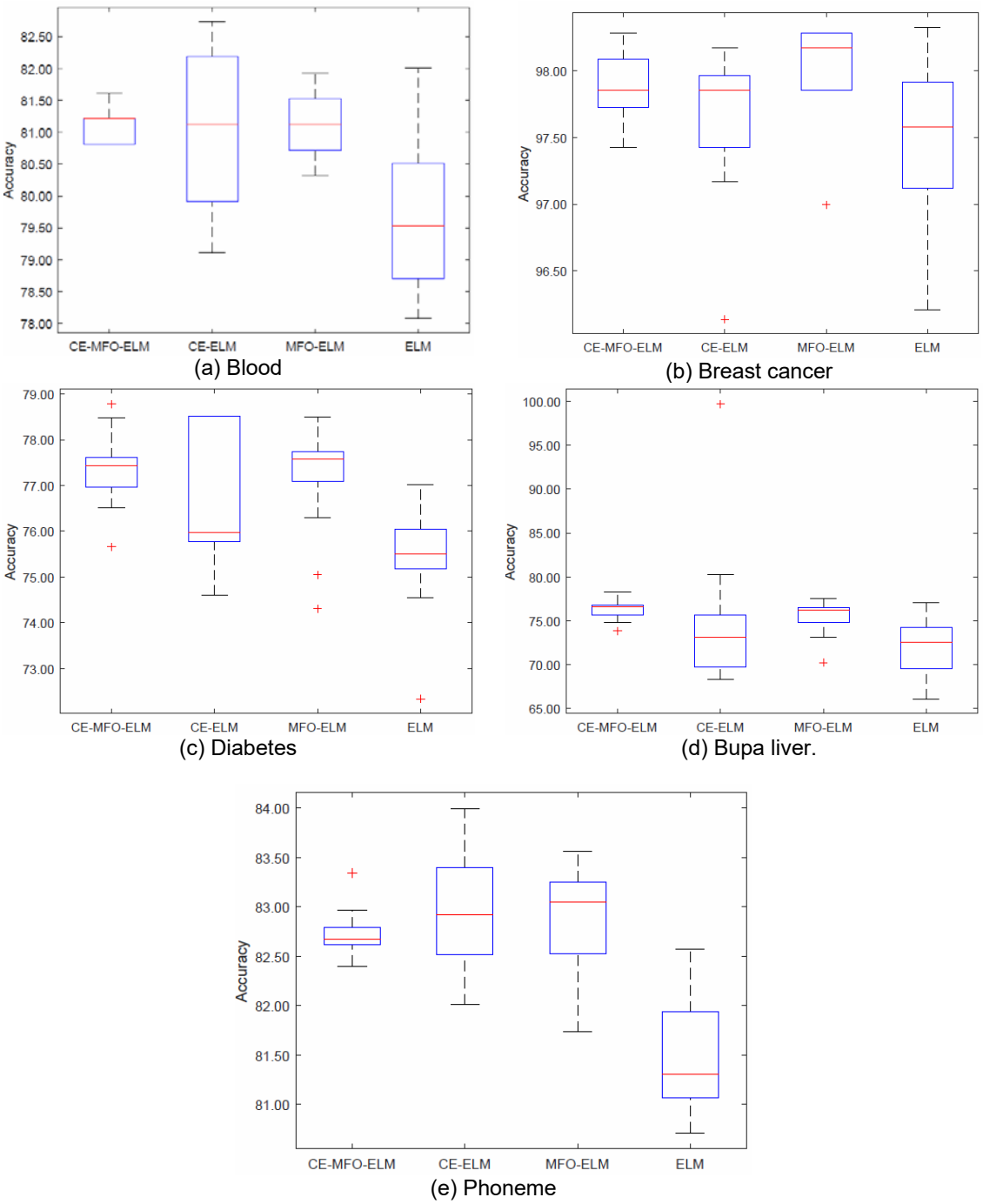
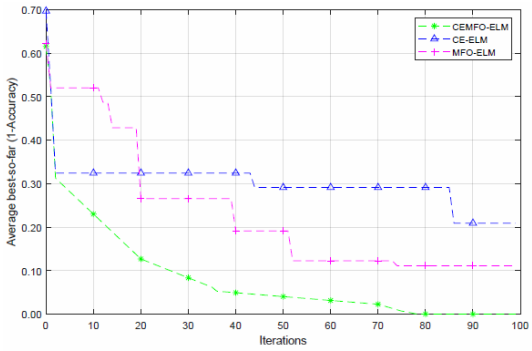


Figure 5 Boxplot showing the stability of CEMFO-ELM, CE-ELM, MFO-ELM, and ELM for the datasets

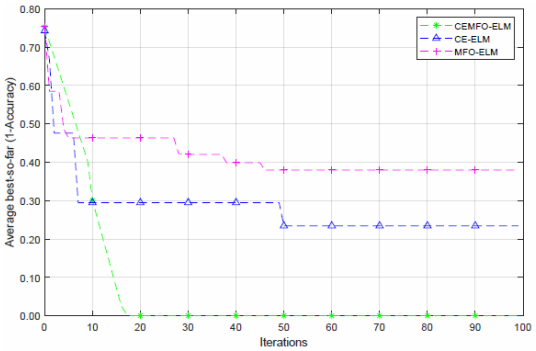
6.3. Convergence of the meta-heuristic algorithms

The convergence test is an important performance index of an optimization algorithm. Figure 6 shows the convergence comparison of the CEMFO-ELM, CE-ELM, and MFO-ELM simulations. The convergence is the iterative descent of the fitness values (error minimization) during the simulation of the comparative algorithms. The figure shows that the proposed CEMFO-ELM converges faster than CE and MFO in all five datasets. CEMFO-ELM completely converged before the 20^{th} iterations in Breast cancer and Diabetes datasets, before the 80^{th} iteration for Blood and Bupa liver datasets, and

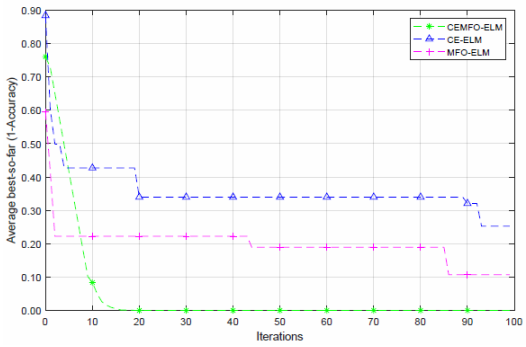
the 90th iteration for the Phoneme dataset. However, the CE-ELM and MFO-ELM had sub-optimal convergence as there are stagnations. CE and MFO did not completely converge in any of the simulations but entered local minimal after the first few iterations in all the simulations.



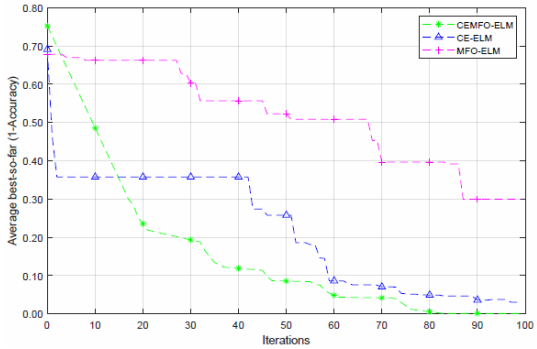
a. Blood



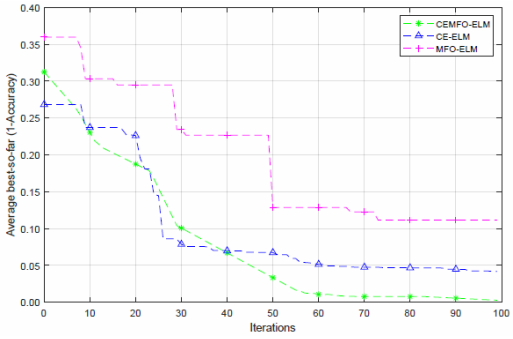
b. Breast cancer



c. Diabetes



d. Bupa liver



e. Phoneme

Figure 6 Convergence curves of CEMFO-ELM, CE-ELM, and MFO-ELM on the datasets.

Therefore, CEMFO-ELM has better searchability under the same experimental conditions in Table 2 than the two individual hybrids with ELM. In all the simulations in Figure 6a-e, the convergences of CEMFO-ELM show a satisfactory local minimal avoidance than the comparative algorithms. This proves that the CEMFO enhancement balanced the exploration and exploitation process in the search space and selected the best particle in the region of feasible solution.

The outstanding performance of CEMFO-ELM over other algorithms is attributed to the hybrid of CE algorithm with swarm intelligent MFO algorithm. The spiral search of moth for flame traversed the best

feasible solution. Also, the particles of the cross-entropy operator eventually push away the searching processes from local optimal. Therefore, the vector is in a balanced state that ensures better convergence. It also eliminates non-optimal solutions and ensures more active solutions.

7. CONCLUSION

This paper presents a hybrid of two optimization schemes to address the problem associated with the random initialization of weights and biases of ELM. Individually, CE-ELM and MFO-ELM metaheuristic algorithms were used to enhance ELM performance. However, they do not attain the optimum solution expected of the algorithm because they could not effectively balance the exploration and exploitation in the search space. Therefore, this work considered the use of a co-evolutionary algorithm. CE operator was employed as the exploration module, which used a probability distribution function to set the initial population for the search algorithm. The MFO operator served as the exploitation module using the spiral function to ensure optimal local search. The operators enhanced the exploration and exploitation processes. The co-evolutionary algorithm improved the accuracy, network size, and stability of ELM. The improvements were analytically evaluated both qualitatively and quantitatively through the use of figures and tables. The results of the experiments show that the proposed CEMFO-ELM algorithm improved the classification accuracies, network size, and stability of ELM with a good convergence rate. More so, its performance improvement on ELM is better when compared with CE-ELM and MFO-EM algorithms.

ACKNOWLEDGEMENTS

This research was supported by the Ministry of Higher Education (MOHE), Malaysia, through the Fundamental Research Grant Scheme FRGS/1/2019/ICT02/UTM/02/13, the Research Management Centre (RMC), UTM, and ALI@S, a research group.

REFERENCES

Ahmed, M. U., Brickman, S., Dengg, A., Fasth, N., Mihajlovic, M., and Norman, J. (2019). A machine learning approach to classify pedestrians' events based on IMU and GPS. *International Journal of Artificial Intelligence*, **17**(2), 154–167.

Al-Betar, M. A., Awadallah, M. A., Abu Doush, I., Hammouri, A. I., Mafarja, M., and Alyasseri, Z. A. A. (2019). Island flower pollination algorithm for global optimization. *The Journal of Supercomputing*, **75**(8), 5280–5323.

Alade, O. A., Sallehuddin, R., Radzi, N. H. M., and Selamat, A. (2019). Missing data characteristics and the choice of imputation technique: An empirical study. *International Conference of Reliable Information and Communication Technology*, 88–97. Springer, Cham.

Alade, O. A., Selamat, A., and Sallehuddin, R. (2018). A review of advances in extreme learning machine techniques and its applications. In F. Saeed, N. Gazem, S. Patnaik, A. S. Saed Balaid, & F. Mohammed (Eds.), *Recent Trends in Information and Communication Technology* (pp. 885–895). Cham: Springer, Cham.

Albadr, M. A. A., and Tiun, S. (2017). Extreme learning machine: A review. *International Journal of Applied Engineering Research*, **12**(14), 4610–4623.

Albu, A., Precup, R. E., and Teban, T. A. (2019). Results and challenges of artificial neural networks used for decision-making and control in medical applications. *Facta Universitatis, Series: Mechanical Engineering*, **17**(3), 285–308.

Alihodzic, A., Tuba, E., and Tuba, M. (2017). An upgraded bat algorithm for tuning extreme learning machines for data classification. *Proceedings of the Genetic and Evolutionary Computation Conference Companion.*, 125–126.

Aljarah, I., Faris, H., and Mirjalili, S. (2018). Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing*, **22**(1), 1–15.

Alshamiri, A. K., Singh, A., and Surampudi, B. R. (2018). Two swarm intelligence approaches for tuning extreme learning machine. *International Journal of Machine Learning and Cybernetics*, **9**(8), 1271–1283.

Botev, Z. I., Kroese, D. P., Rubinstein, R. Y., and L'Ecuyer, P. (2013). The cross-entropy method for optimization from estimation to optimization. *Handbook of Statistics*, 35–59.

Breck, E., Polyzotis, N., Roy, S., Whang, S. E., and Zinkevich, M. (2019). Data validation for machine learning. *SysML*, 1--14.

Cai, Z., Gu, J., Luo, J., Zhang, Q., Chen, H., Pan, Z., … Li, C. (2019). Evolving an optimal kernel extreme learning machine by using an enhanced grey wolf optimization strategy. *Expert Systems with Applications*, **138**, 112814.

Chen, Y., Kloft, M., Yang, Y., Li, C., and Li, L. (2018). Mixed kernel based extreme learning machine for electric load forecasting. *Neurocomputing*, **312**, 90–106.

Cui, Y., Zhai, J., and Wang, X. (2016). Extreme learning machine based on cross entropy. *Proceedings - International Conference on Machine Learning and Cybernetics*, **2**, 1066–1071.

Das, S. R., Kuhoo, Mishra, D., and Rout, M. (2019). An optimized feature reduction based currency forecasting model exploring the online sequential extreme learning machine and krill herd strategies. *Physica A: Statistical Mechanics and Its Applications*, **513**, 339–370.

Duan, M., Li, K., Yang, C., and Li, K. (2018). A hybrid deep learning CNN–ELM for age and gender classification. *Neurocomputing*, **275**, 448–461.

Eshtay, M., Faris, H., Heidari, A. A., Al-Zoubi, A. M., and Aljarah, I. (2020). AutoRWN: automatic construction and training of random weight networks using competitive swarm of agents. *Neural Computing and Applications*, **33**(11), 5507–5524.

Eshtay, M., Faris, H., and Obeid, N. (2018a). Improving Extreme Learning Machine by Competitive Swarm Optimization and its application for medical diagnosis problems. *Expert Systems with Applications*, **104**, 134–152.

Eshtay, M., Faris, H., and Obeid, N. (2018b). Metaheuristic-based extreme learning machines: a review of design formulations and applications. *International Journal of Machine Learning and Cybernetics*, **10**(6), 1543–1561.

Faris, H., Mirjalili, S., Aljarah, I., Mafarja, M., and Heidari, A. A. (2020a). *Nature-Inspired Optimizers* (S. Mirjalili, J. Song Dong, & A. Lewis, eds.). Cham: Springer International Publishing.

Faris, H., Mirjalili, S., Aljarah, I., Mafarja, M., and Heidari, A. A. (2020b). *Salp Swarm Algorithm: Theory, Literature Review, and Applications in Extreme Learning Machines* (Vol. 2020). Springer International Publishing.

Fong, S. J., Li, G., Dey, N., Crespo, R. G., and Herrera-Viedma, E. (2020). Composite Monte Carlo decision making under high uncertainty of novel coronavirus epidemic using hybridized deep learning and fuzzy rule induction. *Applied Soft Computing*, **93**, 106282.

Gabi, D., Ismail, A. S., Zainal, A., Zakaria, Z., and Abraham, A. (2018). Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. *Neural Computing and Applications*, **30**(6), 1845–1863.

Glover, F. (1989). Tabu Search—Part I. *ORSA Journal on Computing*, **1**(3), 190–206.

Guliyev, N. J., and Ismailov, V. E. (2018). Approximation capability of two hidden layer feedforward neural networks with fixed weights. *Neurocomputing*, **316**, 262–269.

Huang, G. Bin, Li, M. Bin, Chen, L., and Siew, C. K. (2008). Incremental extreme learning machine with fully complex hidden nodes. *Neurocomputing*, **71**(4–6), 576–583.

Huang, G. Bin, Zhu, Q., and Siew, C. (2004). Extreme Learning Machine : A New Learning Scheme of Feedforward Neural Networks. *IEEE International Joint Conference on Neural Networks*, **2**, 985–990.

Janakiraman, V. M., Nguyen, X. L., and Assanis, D. (2016). Stochastic gradient based extreme learning machines for stable online learning of advanced combustion engines. *Neurocomputing*, **177**, 304–316.

Li, G., Liu, P., Le, C., and Zhou, B. (2019). A novel hybrid meta-heuristic algorithm based on the cross-entropy method and firefly algorithm for global optimization. *Entropy*, **21**(5).

Li, G., Shuang, F., Zhao, P., and Le, C. (2019). An improved butterfly optimization algorithm for engineering design problems using the cross-entropy method. *Symmetry*, **11**(8).

Lichman, M. (2013). *UCI machine learning repository, University of California, School of Information & Computer Science*. Retrieved from http://archive.ics.uci.edu/ml.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, **234**, 11–26.

Lu, S., Qiu, X., Shi, J., Li, N., Lu, Z.-H., Chen, P., … Zhang, Y. (2016). A pathological brain detection system based on extreme learning machine optimized by bat algorithm. *CNS & Neurological Disorders - Drug Targets*, **16**(1), 23–29.

Mikhaylov, A., and Tarakanov, S. (2020). Development of Levenberg-Marquardt theoretical approach for electric networks. *Journal of Physics: Conference Series*, **1515**(5).

Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, **89**, 228–249.

Nahato, K. B., Nehemiah, K. H., and Kannan, A. (2016). Hybrid approach using fuzzy sets and extreme learning machine for classifying clinical datasets. *Informatics in Medicine Unlocked*, **2**, 1–11.

Nan-Ying Liang, Guang-Bin Huang, Saratchandran, P., Sundararajan, N., Liang, N.-Y., Huang, G.-B., … Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, **17**(6), 1411–1423.

Pozna, C., and Precup, R. E. (2014). Applications of signatures to expert systems modelling. *Acta Polytechnica Hungarica*, **11**(2), 21–39.

Precup, R.-E., David, R.-C., Roman, R.-E., Szedlak-Stinean, A.-I., and Petriu, E. M. (2021). Optimal tuning of interval type-2 fuzzy controllers for nonlinear servo systems using slime mould algorithm. *International Journal of Systems Science*, DOI: 10.1080/00207721.2021.1927236.

Precup, R.-E., Hedrea, E.-L., Roman, R.-C., Petriu, E. M., Szedlak-Stinean, A.-I., and Bojan-Dragos, C.-A. (2021). Experiment-based approach to teach optimization techniques. *IEEE Transactions on Education*, **64**(2), 88–94.

Qing, Y., Zeng, Y., Li, Y., and Huang, G. Bin. (2020). Deep and wide feature based extreme learning machine for image classification. *Neurocomputing*, **412**, 426–436.

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

Saporetti, C. M., Duarte, G. R., Fonseca, T. L., Da Fonseca, L. G., and Pereira, E. (2019). Extreme learning machine combined with a differential evolution algorithm for lithology identification. *Revista de Informática Teórica e Aplicada*, **25**(4), 43.

Sayed, G. I., Darwish, A., and Hassanien, A. E. (2019). Binary whale optimization algorithm and binary moth flame optimization with clustering algorithms for clinical breast cancer diagnoses. *Journal of Classification*.

Shahid, N., Rappon, T., and Berta, W. (2019). Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PLoS ONE*, **14**(2), 1–22.

Su, H., and Cai, Y. (2016). Firefly algorithm optimized extreme learning machine for hyperspectral

image classification. *International Conference on Geoinformatics*, **2016**, 41201341.

Tian, H., Li, S., Wu, T., and Yao, M. (2018). An extreme learning machine based on artificial immune system. *Computational Intelligence and Neuroscience*, **2018**, 1–10.

Tian, Z., Ren, Y., and Wang, G. (2019). Short-term wind speed prediction based on improved PSO algorithm optimized EM-ELM. *Energy Sources, Part A: Recovery, Utilization and Environmental Effects*, **41**(1), 26–46.

Toprak, A. (2018). Extreme learning machine (ELM)-based classification of benign and malignant cells in breast cancer. *Medical Science Monitor*, **24**, 6537–6543.

Vidhya, S., and Kamaraj, V. (2017). Particle swarm optimized extreme learning machine for feature classification in power quality data mining. *Automatika*, **58**(4), 487–494.

Wang, R., Li, J., Wang, J., and Gao, C. (2018). Research and application of a hybridwind energy forecasting system based on data processing and an optimized extreme learning machine. *Energies*, **11**(7).

Wang, Yang, Wang, A., Ai, Q., and Sun, H. (2017). A novel artificial bee colony optimization strategy-based extreme learning machine algorithm. *Progress in Artificial Intelligence*, **6**(1), 41–52.

Wang, Yuguang, Cao, F., and Yuan, Y. (2011). A study on effectiveness of extreme learning machine. *Neurocomputing*, **74**(16), 2483–2490.

Yi, D., Kong, L., and Zhao, Y. (2020). Contrast-Enhancing Snapshot Narrow-Band Imaging Method for Real-Time Computer-Aided Cervical Cancer Screening. *Journal of Digital Imaging*, **33**(1), 211–220.

Zapata, H., Perozo, N., Angulo, W., and Contreras, J. (2020). A hybrid swarm algorithm for collective construction of 3D structures. *International Journal of Artificial Intelligence*, **18**(1), 1–18.

Zhang, X., Yang, Z., Cao, F., Cao, J., Wang, M., and Cai, N. (2018). Conditioning optimization of extreme learning machine by multitask beetle antennae swarm algorithm. *ArXiv Preprint ArXiv:1811.09100*.

Zhao, J., Wang, Z., and Park, D. S. (2012). Online sequential extreme learning machine with forgetting mechanism. *Neurocomputing*, **87**, 79–89.

Zhou, Y., Zhou, N., Gong, L., and Jiang, M. (2020). Prediction of photovoltaic power output based on similar day analysis, genetic algorithm and extreme learning machine. *Energy*, **204**, 117894.