Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/200412

Hybrid Electric Eel Foraging Optimization and Quadratic Interpolation Optimization for Multilayer Perceptron Training

*Imad El Karkri** *and Abdelghni Lakehal*

Polydisciplinary Faculty Larache, Abdelmalek Essaadi University, Larache, Morocco

Received: 22 Mar. 2026, Revised: 12 Jun. 2026, Accepted: 21 Jun. 2026
Published online: 1 Jul. 2026

Abstract: Multilayer perceptron (MLP) training is a difficult optimization task due to the complexity of searching for suitable weights and biases. Classical backpropagation may suffer from slow convergence, sensitivity to initialization, and stagnation in local optima. To address these issues, this paper proposes a new hybrid metaheuristic method, called EEFO-QIO, for MLP training. The proposed approach combines Electric Eel Foraging Optimization (EEFO) and Quadratic Interpolation Optimization (QIO), benefiting from the exploration capability of EEFO and the exploitation strength of QIO, while a restart mechanism is used to reinitialize candidate solutions when stagnation occurs, except for the global best solution. The method is applied to optimize the weights and biases of the MLP on five benchmark classification datasets: XOR, Balloon, Iris, Breast Cancer, and Heart. Its performance is evaluated using mean squared error, classification accuracy, and the Friedman test. The results show that the proposed EEFO-QIO method provides robust and competitive performance for MLP training.

Keywords: Multilayer perceptron, Electric Eel Foraging Optimization, Quadratic Interpolation Optimization, Hybrid metaheuristic, Neural network training, Classification accuracy

1 Introduction

Artificial neural network (ANN) models have become common tools in artificial intelligence (AI) and machine-learning applications because they can capture complex relationships within data. Their design is inspired by the way the human brain processes information, which has made them useful for several applications, such as classification, prediction, regression, and pattern recognition. Among ANN architectures, multilayer perceptrons (MLPs) are frequently adopted because they have a simple structure, can represent nonlinear mappings, and perform effectively in supervised learning tasks.

The performance of an MLP mainly depends on how well its weights and biases are adjusted during training. Finding appropriate values for these parameters is not a simple task, since the problem becomes more difficult when the network is large and the corresponding search space is nonlinear and high-dimensional. In most cases, MLPs are trained using gradient-based methods, with backpropagation being the most common learning algorithm [1]. Although backpropagation is simple and effective in many applications, it has some important weaknesses. The obtained solution can be strongly affected by the initial parameter values, and the convergence process may become slow when the error surface is complicated. The choice of learning rate strongly affects the training behavior; a very small step size can slow the update process, while an excessively large step size may produce unstable learning or premature convergence. Moreover, because gradient-based methods use local derivative information, they may become trapped in local optima, and their performance can deteriorate when the size and complexity of the search space increase.

To address these drawbacks, metaheuristic algorithms have been considered as effective alternatives for ANN training. These methods search the parameter space without using gradient information, which makes them suitable for complex, nonlinear, and multimodal optimization problems. Consequently, many researchers have adopted metaheuristic techniques to tune the

* Corresponding author e-mail: elkarkri.imad@gmail.com

trainable parameters of feedforward neural-network models. For example, the work of Aljarah and coauthors [2] investigated the use of the Whale Optimization Algorithm for training feedforward neural networks and evaluated it on different classification datasets. The reported results indicated that this approach improved convergence and helped avoid local optima compared with backpropagation and several evolutionary algorithms. In a related study, Benmessahel et al. [3] applied Multi-Verse Optimization to train a multilayer ANN for intrusion detection. Their model achieved competitive results on security benchmark datasets and performed better than an ANN trained with particle swarm optimization (PSO).

Several additional swarm-intelligence and evolutionary techniques have been examined for training neural networks. Chatterjee et al. [4] developed a PSO-trained ANN to predict the failure probability of multistory concrete buildings, and the obtained model produced better prediction results than a conventional feedforward ANN. In another work, Tezel and coauthors [5] relied on the Artificial Algae Algorithm to tune ANN weights and reported reduced prediction errors on various UCI classification datasets. Yamany and colleagues [6] adopted Moth-Flame Optimization for selecting appropriate weight and bias values in feedforward multilayer neural networks. Their experiments reported improved classification performance and a strong ability to avoid poor local solutions in comparison with genetic algorithm (GA), PSO, ant colony optimization (ACO), and Evolution Strategy. In addition, Jaddi et al. [7] introduced a neural network optimized by a modified Bat Algorithm and demonstrated its effectiveness in classification, time-series prediction, and rainfall forecasting.

These works suggest that metaheuristic algorithms offer a valuable alternative to traditional gradient-based training, particularly when the ANN error landscape is complex, high-dimensional, or irregular. However, their effectiveness is still linked to the way the search process manages exploration and exploitation. Limited exploration can lead the algorithm to converge too early toward a local optimum. Conversely, insufficient exploitation may slow the improvement of promising solutions and lower the final accuracy. Therefore, hybrid training strategies that combine wide search capability with efficient local refinement remain an important research direction.

In this work, EEFO-QIO is presented as a hybrid metaheuristic method for training multilayer perceptron networks. The learning task is handled as an optimization task, where each candidate solution contains the complete set of MLP parameters, and the classification error measures its quality. The proposed method combines the optimizer proposed in [8], namely Electric Eel Foraging Optimization (EEFO), with the quadratic interpolation-based optimizer introduced in [9], namely Quadratic Interpolation Optimization (QIO), along with a restart strategy to improve the training search. EEFO-QIO is tested using five benchmark datasets for classification: XOR, Balloon, Iris, Breast Cancer, and Heart.

The contributions of this study can be summarized as follows:

- EEFO-QIO is developed for MLP training; within this framework, the MLP parameters are tuned by using the global exploration behavior of Electric Eel Foraging Optimization and the local refinement mechanism of Quadratic Interpolation Optimization.
- The method includes a restart strategy that maintains population diversity and helps prevent stagnation and premature convergence during learning.
- The proposed approach is validated on five benchmark classification problems, namely the XOR, Balloon, Iris, Breast Cancer, and Heart datasets; its performance is examined in terms of classification accuracy, convergence behavior, and robustness.

The paper then proceeds as follows. Section 2 gives the preliminaries of the multilayer perceptron and the two optimizers used in this study, EEFO and QIO. Section 3 details the proposed EEFO-QIO learning scheme. Section 4 presents the experimental findings and performance evaluation. Section 5 discusses the obtained results and the limitations of the proposed method. Section 6 concludes the paper.

2 Preliminaries

2.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) is a common feedforward ANN architecture [10]. It is used in classification, regression, and function approximation tasks because it can model nonlinear relationships and provide good generalization. In this network, signals move forward only, starting from the input layer and passing through the hidden layer or layers until they reach the output layer. An MLP architecture is generally formed by input, hidden, and output layers. Processing units in neighboring layers are linked through weighted connections, and a bias value is assigned to each unit. For a given unit j, the net input is computed as:

$$v_j = \sum_{i=1}^{n} w_{ij} x_i + b_j \tag{2.1}$$

where x_i denotes the input component with index i, w_{ij} represents the weight linking this input to unit j, b_j denotes the corresponding bias, and n gives the total number of inputs. The resulting net value is then transformed by an activation function. In the present study, the sigmoid function is adopted:

$$f(v_j) = \frac{1}{1+e^{-v_j}} \tag{2.2}$$

Thus, the output of unit j is given by:

$$O_j = f(v_j) \tag{2.3}$$

Each classification or regression task requires an appropriate network architecture. After the structure is fixed, a learning algorithm adjusts the trainable weights and biases with the aim of lowering the prediction error. This learning process is carried out by minimizing the total error of the network, defined as:

$$E(W(t)) = \frac{1}{2}\sum_{l=1}^{s}\sum_{k=1}^{m}\left(d_k^l - O_k^l\right)^2 \tag{2.4}$$

where $E(W(t))$ denotes the total error at iteration t, $W(t)$ represents the set of network parameters at iteration t, d_k^l and O_k^l are the desired and obtained outputs of unit k for training instance l, respectively, m is the number of output units, and s is the number of training instances.

Since the optimization of weights and biases is a nonlinear and multimodal problem, the training stage is considered the most critical part of the MLP. For this reason, efficient optimization methods are needed to obtain better network performance.

2.2 Electric Eel Foraging Optimization (EEFO)

Electric Eel Foraging Optimization (EEFO) belongs to the class of population-based metaheuristic algorithms and is motivated by the collective hunting behavior observed in electric eels. In this algorithm, an eel denotes one candidate solution, whereas the best individual obtained during the search is treated as the prey and represented by x_{prey}. The population is updated by means of four behavioral operators, namely interaction, resting, hunting, and migration. Through these operators, EEFO aims to coordinate diversification in the search space with intensification around promising regions.

In the interaction behavior, the position of an eel is modified according to another eel selected at random, together with either the population mean or a randomly generated position. This phase mainly supports the exploratory search.

$$v_i(t+1) = \begin{cases} x_j(t) + C\left(\bar{x}(t) - x_i(t)\right), & p_1 > 0.5, \\ x_j(t) + C\left(x_r(t) - x_i(t)\right), & p_1 \le 0.5. \end{cases} \tag{2.5}$$

Otherwise, the update is performed by:

$$v_i(t+1) = \begin{cases} x_i(t) + C\left(\bar{x}(t) - x_j(t)\right), & p_2 > 0.5, \\ x_i(t) + C\left(x_r(t) - x_j(t)\right), & p_2 \le 0.5. \end{cases} \tag{2.6}$$

where C, p_1, and p_2 are random numbers, $x_r(t)$ is a randomly generated position, and $\bar{x}(t)$ denotes the average position of the current population.

In the resting behavior, a resting region is first constructed around a projected point $Z(t)$, given by:

$$Z(t) = Low + z(t)(Up - Low) \tag{2.7}$$

where $z(t)$ is a normalized random value. The initial scale of the resting area is defined as:

$$\alpha_0 = 2e^{\frac{T_{\max}-t}{T_{\max}}} \tag{2.8}$$

Then, a resting position is generated by:

$$R_i(t+1) = Z(t) + \alpha\left|Z(t) - x_{\text{prey}}(t)\right| \tag{2.9}$$

where $\alpha = \alpha_0 \sin(2\pi r_2)$ and $r_2 \in (0,1)$. After that, the eel position is updated inside the resting region according to:

$$v_i(t+1) = R_i(t+1) + n_2\left(R_i(t+1) - \text{round}(\text{rand}) \times x_i(t)\right) \tag{2.10}$$

where $n_2 \sim N(0,1)$. This behavior enhances local exploitation around promising regions.

In the hunting behavior, the prey is assumed to move within a hunting area centered around the best solution. Its new position is computed as:

$$H_{\text{prey}}(t+1) = x_{\text{prey}}(t) + \beta\left|\bar{x}(t) - x_{\text{prey}}(t)\right| \tag{2.11}$$

where β controls the hunting range. Then, the eel curls around the prey and updates its position by:

$$v_i(t+1) = H_{\text{prey}}(t+1) + \eta\left(H_{\text{prey}}(t+1) - \text{round}(\text{rand}) \times x_i(t)\right) \tag{2.12}$$

where the curling factor η is defined as:

$$\eta = e^{r_4 \frac{t}{T_{\max}}}\cos(2\pi r_4) \tag{2.13}$$

with $r_4 \in (0,1)$. This mechanism strengthens the local search around the prey.

In the migrating behavior, eels move from the resting region toward the hunting area. The position update is expressed as:

$$v_i(t+1) = -r_5 R_i(t+1) + r_6 H_r(t+1) - L\left(H_r(t+1) - x_i(t)\right) \tag{2.14}$$

where $H_r(t+1)$ is a position in the hunting area, r_5 and r_6 are random numbers, and L is a Lévy flight term used to improve exploitation and help the search escape local optima.

The transition between exploration and exploitation is controlled by the energy factor:

$$E(t) = 4r_7\left(1 - \frac{t}{T_{\max}}\right) \tag{2.15}$$

where $r_7 \in (0,1)$. When $E(t) > 1$, the algorithm performs the interacting behavior to favor exploration. Otherwise, it switches to local exploitation by applying one of the resting, migrating, or hunting behaviors. After generating a new position, the solution is evaluated and the global best x_{prey} is updated if a better candidate is found.

Algorithm 1 Pseudocode of the EEFO algorithm

Require: Population size N, maximum number of iterations $T_{\max}$
Ensure: Best solution x_{prey}
Initialize the population of eels randomly.
Evaluate all candidate solutions and determine the best one x_{prey}.
for $t = 1$ to $T_{\max}$ **do**
  for each eel x_i **do**
    Compute the energy factor $E(t)$ using
    Eq. (2.15).
    if $E(t) > 1$ **then**
      Update x_i using the interacting behavior in
      Eqs. (2.5) and (2.6).
    else
      Select one of the resting, migrating, or
      hunting behaviors.
      Update x_i using Eqs. (2.7)–(2.14).
    end if
    Apply boundary control and evaluate the new
    solution.
    Keep the new position if it improves the current
    one.
  end for
  Update the global best solution x_{prey}.
end for
return x_{prey}

2.3 Quadratic Interpolation Optimization (QIO)

Quadratic Interpolation Optimization (QIO) is a population-based optimizer that relies on quadratic interpolation to identify promising solutions. Its basic principle consists of building a quadratic approximation using three objective-function points and then employing the minimum point of this approximation to direct the search process. Through this mechanism, QIO uses the curve-fitting property of quadratic interpolation to guide the population toward more favorable areas in the solution space.

The quadratic interpolation model is defined as:

$$L(x) = \alpha x^2 + \beta x + \delta \tag{2.16}$$

where α, β, and δ are coefficients determined from three interpolation points. Once the quadratic model is constructed, its minimizer is obtained by setting the derivative to zero, which gives

$$x^* = -\frac{\beta}{2\alpha} \tag{2.17}$$

This minimizer is treated as an approximation of an improved candidate solution and is then employed to update the population.

The search process in QIO is divided into two stages: exploration and exploitation. During the exploration stage, three individuals are chosen, and generalized quadratic interpolation is used to obtain a predicted point. This point is then modified by a perturbation term to promote diversity and help the algorithm investigate unexplored areas. The corresponding update rule is given by:

$$\begin{aligned} v_i(t+1) = {} & x^*_{i,r_1,r_2}(t) + w_1\left(x_{r_3}(t) - x^*_{i,r_1,r_2}(t)\right) \\ & + \text{round}\left(0.5(0.05 + r_1)\right)\frac{\log(r_2)}{r_3} \end{aligned} \tag{2.18}$$

where $x^*_{i,r_1,r_2}(t)$ is the minimizer obtained by generalized quadratic interpolation, and w_1, r_1, r_2, and r_3 are random or control parameters.

In the exploitation phase, QIO uses the current best solution together with two randomly selected individuals to produce a more refined search step around promising areas. The update rule is given by:

$$\begin{aligned} v_i(t+1) = {} & x^*_{\text{best},r_1,r_2}(t) \\ & + w_2\left(x_{\text{best}} - \text{round}(1+\text{rand})\right)\left(\frac{Ub - Lb}{Ub_{rD} - Lb_{rD}}\right) \end{aligned} \tag{2.19}$$

where $x^*_{\text{best},r_1,r_2}(t)$ is the minimizer generated from the best solution and two random individuals, Ub and Lb are the upper and lower bounds of the search space, Ub_{rD} and Lb_{rD} denote the upper and lower bounds associated with a randomly selected dimension rD, and w_2 is an adaptive parameter defined by:

$$w_2 = 3\left(1 - \frac{t}{T_{\max}}\right) \tag{2.20}$$

which gradually decreases with iterations to strengthen local exploitation.

After generating a trial position, QIO adopts a greedy selection mechanism. The updated solution is accepted only if it improves the objective value:

$$x_i(t+1) = \begin{cases} x_i(t), & f(x_i(t)) \le f(v_i(t+1)), \\ v_i(t+1), & f(x_i(t)) > f(v_i(t+1)). \end{cases} \tag{2.21}$$

Through this mechanism, QIO combines the estimation capability of quadratic interpolation with population-based search, allowing the algorithm to refine candidate solutions effectively. Owing to its strong local search behavior, QIO is particularly suitable for exploitation and is therefore used in this work as the exploitation component of the proposed hybrid method.

3 EEFO-QIO for Training MLP

EEFO-QIO is introduced as a training method for the MLP. Its objective is to minimize the network error on the training set by adjusting the MLP trainable parameters. The proposed method combines the strong exploratory behavior of EEFO with the local refinement ability of QIO.

Algorithm 2 Pseudocode of the QIO algorithm

Require: Population size N, maximum number of iterations $T_{\max}$
Ensure: Best solution x_{best}
Initialize the population randomly.
Evaluate the fitness of all individuals.
Determine the best solution x_{best}.
for $t = 1$ to $T_{\max}$ **do**
  for each individual x_i **do**
    Generate the quadratic-interpolation
    minimizer using Eqs. (2.16) and (2.17).
    Generate a random number to choose the
    search phase.
    if the selected phase is exploration **then**
      Update the trial position $v_i(t+1)$
      using Eq. (2.18).
    else
      Update the trial position $v_i(t+1)$ using the
      exploitation rule in Eq. (2.19).
      Compute the adaptive coefficient w_2 using
      Eq. (2.20).
    end if
    Evaluate the trial solution $v_i(t+1)$.
    Update the current individual $x_i(t+1)$ using
    the greedy selection rule in Eq. (2.21).
  end for
  Update the global best solution x_{best}.
end for
return x_{best}

Assume that the MLP structure includes n_i units in the input layer, n_h units in the hidden layer, and n_o units in the output layer. Each candidate solution is represented by a real-valued vector that stores all trainable parameters of the model. Accordingly, the dimension of each search agent is defined as:

$$D = (n_i \times n_h) + (n_h \times n_o) + n_h + n_o \tag{3.1}$$

where the first two terms correspond to the weights between consecutive layers, while the last two terms represent the biases of the hidden and output layers, respectively.

After initialization, each individual is decoded into the corresponding MLP parameters, and its fitness is evaluated using the total error function defined in Eq. (2.4). In this way, the training of the MLP is transformed into a continuous optimization problem, where the optimizer seeks the parameter vector that minimizes the network error.

The proposed hybrid strategy relies on a simple switching mechanism between EEFO and QIO. Since EEFO is more suitable for global exploration and QIO is more effective for local exploitation, the search process is defined as follows:

$$x_i(t+1) = \begin{cases} \text{EEFO\_exploration}\left(x_i(t)\right), & E(t) > 0.5, \\ \text{QIO\_exploitation}\left(x_i(t)\right), & E(t) \le 0.5. \end{cases} \tag{3.2}$$

where $E(t)$ denotes the energy factor at iteration t. When $E(t) > 0.5$, the exploration phase of EEFO is activated to investigate new regions of the search space. Otherwise, the exploitation phase of QIO is applied to refine the search around promising solutions. In this manner, the proposed method attempts to maintain a better balance between diversification and intensification during the training process.

To further reduce the risk of premature convergence, a stagnation-handling strategy is incorporated. When the algorithm detects stagnation, it reinitializes the positions of all candidate solutions except the global best solution x_{GB}. The stagnation condition is triggered when either the number of consecutive iterations with improvement below a tolerance threshold exceeds a predefined limit, or the population diversity becomes smaller than a given threshold. This condition is expressed as:

$$\left(n^{TOL} > n^{res}\right) \vee \left(DIV < \varepsilon_{DIV}\right) \tag{3.3}$$

where n^{TOL} denotes the number of consecutive iterations for which the improvement remains below the tolerance threshold, n^{res} is the maximum allowed number of such iterations before restarting, and DIV is the diversity measure of the population, defined as:

$$DIV = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{D}\sum_{j=1}^{D}\left|\text{median}(x_j) - x_{ij}\right|\right) \tag{3.4}$$

and ε_{DIV} denotes the diversity threshold. When the condition in Eq. (3.3) holds, the current global best solution is kept unchanged, whereas the other individuals are randomly relocated within the search space. This restart strategy increases the diversity of the population and helps the search move away from stagnant regions.

The proposed EEFO-QIO method proceeds as follows. First, the MLP structure and the control parameters of the hybrid optimizer are defined. Next, an initial population is generated, and the fitness of each individual is computed using the MLP error function. During each iteration, the energy factor is calculated to determine whether the EEFO exploration phase or the QIO exploitation phase should be applied. The generated candidate solutions are then assessed, and only better solutions are accepted using greedy selection. The global best solution is updated throughout the search, and the restart strategy is triggered whenever stagnation is detected. The iterative procedure stops when the termination condition is reached. At the end, the best obtained solution represents the optimized weights and biases of the trained MLP.

The main steps of the proposed method are summarized in Algorithm 3.

Algorithm 3 Pseudocode of the proposed EEFO-QIO method

Require: Population size N, maximum number of iterations $T_{\max}$, MLP architecture, training dataset, stagnation limit n^{res}, diversity threshold ε_{DIV}
Ensure: Best weights and biases of the MLP
Define the MLP architecture and encode all weights and biases into a solution vector using Eq. (3.1).
Initialize the population randomly within the search bounds.
Evaluate the fitness of each individual using the MLP error function in Eq. (2.4).
Determine the global best solution x_{GB}.
for $t = 1$ to $T_{\max}$ **do**
    Compute the energy factor $E(t)$.
    for each individual x_i **do**
        if $E(t) > 0.5$ **then**
            Generate a new candidate solution using the EEFO exploration phase.
        else
            Generate a new candidate solution using the QIO exploitation phase.
        end if
        Evaluate the new candidate solution using Eq. (2.4).
        Keep the new position if it improves the current one.
    end for
    Update the global best solution x_{GB}.
    Compute the stagnation indicators n^{TOL} and DIV using Eq. (3.4).
    if the stagnation condition in Eq. (3.3) is satisfied **then**
        Preserve x_{GB} and reinitialize all other individuals randomly within the search space.
    end if
end for
return the best solution x_{GB}

4 Experimental Results

4.1 Simulation Platform

All simulations are implemented in Python 3 and carried out on a personal computer equipped with an Intel Core i5-1335U CPU at 3.4 GHz, Intel Iris Graphics, and 16 GB RAM.

4.2 Datasets

The classification ability of EEFO-QIO is investigated using five benchmark datasets with different difficulty levels. Such datasets are commonly adopted in previous studies to compare metaheuristic-based MLP training methods. They cover simple logical cases as well as real classification problems, which allows the robustness and generalization of the proposed method to be examined. The datasets considered in this work were mainly obtained from the UCI Machine Learning Repository [11]. For every dataset, the available samples are randomly split into two subsets: 70% for learning and 30% for evaluation. Table 1 reports the main properties of these datasets.

In all experiments, the hidden-layer size was set using the rule $2 \times n + 1$, where n represents the input-layer size. This choice gives a uniform way to define the MLP architecture for all datasets. Before starting the optimization, the MLP parameters were initialized randomly in the interval $[-10, 10]$.

Table 1: Main characteristics of the classification datasets.

Classification dataset	Attributes	Classes
3-bits XOR	3	2
Balloon	4	2
Iris	4	3
Breast cancer	9	2
Heart	22	2

4.3 Parameter Settings

For a consistent comparison, EEFO-QIO and the other algorithms were tested using similar experimental settings whenever possible. The population size and the maximum iteration number were kept identical for all methods, whereas the remaining parameters were set based on the original descriptions of the algorithms or values commonly adopted in previous studies. The compared methods include Particle Swarm Optimization (PSO) [12], African Vultures Optimization Algorithm (AVOA) [13], Quadratic Interpolation Optimization (QIO), Electric Eel Foraging Optimization (EEFO), and the proposed EEFO-QIO. Table 2 presents the parameter values adopted in the experiments.

Table 2: Parameter settings of the proposed and compared algorithms.

Algorithm	Parameter	Value
EEFO-QIO	n^{res}; ε_{DIV}	200; 0.001
AVOA	L_1; L_2; w	0.8; 0.2; 2.5
	P_1; P_2; P_3	0.6; 0.4; 0.6
PSO	c_1; c_2; ω	1.8; 2; 1
All algorithms	Agents	100
	Max. iterations	1000
	Runs	31

4.4 Performance Evaluation and Statistical Analysis

This subsection assesses EEFO-QIO using mean squared error (MSE), classification accuracy, convergence behavior, and the Friedman rank test. The numerical outcomes obtained by the compared algorithms, including the mean value, standard deviation, and classification accuracy, are given in Table 3. The global statistical ranking based on classification accuracy is provided in Table 4. Moreover, Fig. 1 shows the convergence curves of the compared methods on the selected datasets, namely 3-bits XOR, Iris, Balloon, Breast Cancer, and Heart.

Table 3: Mean, standard deviation, and accuracy results.

Dataset	Metric	PSO	AVOA	QIO	EEFO	EEFO-QIO
3-bits XOR	Mean	7.32E−2	5.82E−2	6.17E−3	5.89E−3	3.79E−3
	Std	8.45E−2	9.64E−2	7.52E−3	4.31E−3	2.63E−3
	Accuracy	85	86	98	98	100
Balloon	Mean	5.82E−2	1.31E−2	5.46E−8	7.46E−8	5.91E−9
	Std	7.25E−2	2.36E−3	6.69E−8	8.21E−8	1.40E−9
	Accuracy	82	84	99	98	100
Iris	Mean	8.33E−1	3.11E−3	6.41E−2	1.08E−2	6.17E−3
	Std	6.54E−1	1.27E−3	3.22E−2	1.90E−2	2.70E−3
	Accuracy	79	97	82	84	96
Breast Cancer	Mean	9.72E−2	8.51E−2	4.66E−3	2.73E−3	3.79E−3
	Std	5.28E−2	6.80E−5	3.11E−4	1.91E−5	3.79E−5
	Accuracy	94	95	97	99	98
Heart	Mean	9.12E−1	9.89E−1	5.78E−1	1.95E−1	1.09E−1
	Std	2.28E−3	3.10E−2	2.41E−2	5.14E−3	2.13E−3
	Accuracy	71	70	72	74	76

Table 3 shows that EEFO-QIO achieves the best or very competitive results on most datasets in terms of mean squared error, standard deviation, and classification accuracy. In particular, it attains the best performance on 3-bits XOR, Balloon, and Heart, with the lowest errors and the highest accuracies. Although AVOA performs slightly better on Iris and EEFO ranks first on Breast Cancer, the proposed method remains highly competitive overall, confirming the effectiveness of the hybrid strategy. Table 4 shows that EEFO-QIO obtains the highest average Friedman rank of 4.60 among all compared algorithms. It is followed by EEFO with an average rank of 3.70 and QIO with an average rank of 3.10. AVOA achieves an average rank of 2.40, while PSO records the lowest average rank of 1.20. These results indicate that the proposed hybrid method provides the most consistent classification performance across the considered datasets.

Table 4: Friedman rank according to classification accuracy.

Dataset	PSO	AVOA	QIO	EEFO	EEFO-QIO
3-bits XOR	1.00	2.00	3.50	3.50	5.00
Balloon	1.00	2.00	4.00	3.00	5.00
Iris	1.00	5.00	2.00	3.00	4.00
Breast Cancer	1.00	2.00	3.00	5.00	4.00
Heart	2.00	1.00	3.00	4.00	5.00
Average rank	1.20	2.40	3.10	3.70	4.60

The convergence curves in Fig. 1 show that EEFO-QIO provides the fastest and most stable reduction of MSE on the selected datasets. The proposed method drops sharply in the early iterations and reaches a near-zero error faster than all competing algorithms, while EEFO converges much more slowly and PSO shows a smoother but delayed descent. On the Heart dataset, EEFO-QIO again exhibits the best convergence behavior, achieving the lowest MSE in fewer iterations, which confirms its stronger search capability on the most difficult problem. For the Iris dataset, EEFO-QIO also shows a very rapid decrease in MSE and attains the best final convergence profile. Although QIO and PSO eventually reach very low error values, their convergence is slower than that of the proposed method. In contrast, AVOA and especially EEFO require more iterations and remain less competitive during most of the search process. Overall, these curves indicate that the hybridization of EEFO and QIO improves both convergence speed and solution quality by combining efficient exploration with stronger exploitation.

5 Discussion

The obtained results indicate that EEFO-QIO performs well as an MLP training method. This behavior can be attributed to the complementary functions of EEFO and QIO. EEFO supports exploration by guiding candidate solutions toward diverse areas of the search domain, which lowers the possibility of convergence to poor local solutions. In contrast, QIO enhances exploitation by applying quadratic interpolation to improve promising candidates. The energy-based switching mechanism enables the optimizer to select between these two search behaviors according to the current stage of training, instead of depending on one fixed update rule throughout the process.

The restart mechanism also contributes to the robustness of the method. When the improvement becomes very small for several consecutive iterations or when the diversity measure decreases below the threshold, the population may no longer explore useful new regions. In this case, reinitializing the candidate solutions while preserving the global best solution restores diversity without losing the best information obtained so far. This explains the stable convergence behavior observed in Fig. 1.

From Table 3, EEFO-QIO gives the highest accuracy on 3-bits XOR, Balloon, and Heart, while maintaining competitive results on Iris and Breast Cancer. This suggests that the method performs well on simple logical

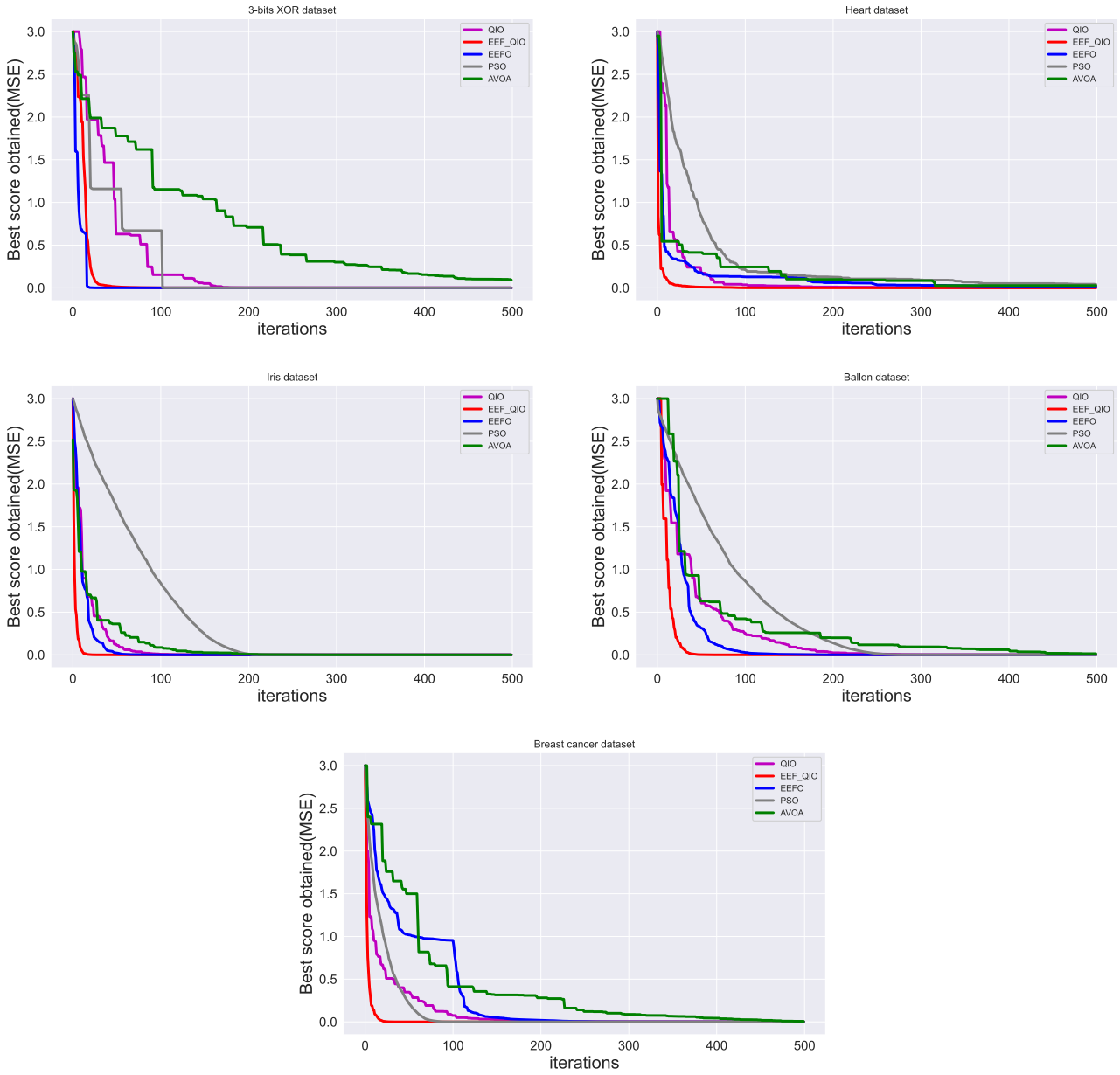

Fig. 1: Convergence curves of the algorithms on the 3-bits XOR, Heart, Iris, Balloon, and Breast Cancer datasets.

datasets and can also handle real-world classification tasks. The Friedman ranking reported in Table 4 reinforces this result, as EEFO-QIO obtains the highest average rank relative to the other algorithms. However, the results on Iris and Breast Cancer also reveal that the proposed approach does not dominate all datasets. This behavior agrees with the No Free Lunch theorem [14], which states that no optimizer can be superior for every problem. Future work may focus on adaptive control of the switching threshold and evaluation on larger datasets.

6 Conclusion

The present study introduced EEFO-QIO as a hybrid metaheuristic approach for Multilayer Perceptron (MLP) training. EEFO is used to support global search, whereas QIO performs local refinement, allowing the network weights and biases to be optimized more effectively. The experiments conducted on five benchmark classification datasets showed that the proposed approach provides competitive and robust results based on mean squared error and classification accuracy. The Friedman test further confirmed that EEFO-QIO obtains the highest

average rank compared with the other algorithms. Moreover, the convergence curves showed faster and more stable convergence on several datasets. These findings indicate that the proposed hybrid strategy can be considered an effective alternative for MLP training. Future work may focus on applying the method to larger datasets and more complex neural network architectures.

References

[1] Wang, L., Zeng, Y., Chen, T. "Back propagation neural network with adaptive differential evolution algorithm for time series forecasting." *Expert Systems with Applications* 42.2 (2015): 855–863.

[2] I. Aljarah, H. Faris, and S. Mirjalili, "Optimizing connection weights in neural networks using the whale optimization algorithm," *Soft Computing*, vol. 22, no. 1, pp. 1–15, 2018.

[3] I. Benmessahel, K. Xie, and M. Chellal, "A new evolutionary neural networks based on intrusion detection systems using multiverse optimization," *Applied Intelligence*, vol. 48, pp. 2315–2327, 2018.

[4] S. Chatterjee, S. Sarkar, S. Hore, N. Dey, A. S. Ashour, and V. E. Balas, "Particle swarm optimization trained neural network for structural failure prediction of multistoried RC buildings," *Neural Computing and Applications*, vol. 28, no. 8, pp. 2005–2016, 2017.

[5] G. Tezel, S. A. Uymaz, and E. Yel, "Combining artificial algae algorithm to artificial neural network for optimization of weights," *Data Science and Applications*, vol. 1, pp. 37–44, 2018.

[6] W. Yamany, M. Fawzy, A. Tharwat, and A. E. Hassanien, "Moth-flame optimization for training multi-layer perceptrons," in *Proceedings of the 2015 11th International Computer Engineering Conference (ICENCO)*, IEEE, pp. 267–272, 2015.

[7] N. S. Jaddi, S. Abdullah, and A. R. Hamdan, "Optimization of neural network model using modified bat-inspired algorithm," *Applied Soft Computing*, vol. 37, pp. 71–86, 2015.

[8] Zhao, W., Wang, L., Zhang, Z., Fan, H., Zhang, J., Mirjalili, S., et al. "Electric eel foraging optimization: A new bio-inspired optimizer for engineering applications." *Expert Systems with Applications* 238 (2024): 122200.

[9] Zhao, W., Wang, L., Zhang, Z., Mirjalili, S., Khodadadi, N., Ge, Q. "Quadratic interpolation optimization (QIO): A new optimization algorithm based on generalized quadratic interpolation and its applications to real-world engineering problems." *Computer Methods in Applied Mechanics and Engineering* 417 (2023): 116446.

[10] Yang, J., Ma, J. "Feed-forward neural network training using sparse representation." *Expert Systems with Applications* 116 (2019): 255–264.

[11] Frank, A. *UCI Machine Learning Repository*. 2010. Available at: `http://archive.ics.uci.edu/ml`.

[12] Kennedy, J., Eberhart, R. "Particle swarm optimization." *Proceedings of ICNN'95 - International Conference on Neural Networks* 4 (1995): 1942–1948.

[13] Abdollahzadeh, B., Gharehchopogh, F. S., Mirjalili, S. "African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems." *Computers & Industrial Engineering* 158 (2021): 107408.

[14] Wolpert, D. H., Macready, W. G. "No free lunch theorems for optimization." *IEEE Transactions on Evolutionary Computation* 1.1 (2002): 67–82.

Imad El Karkri received a Master's degree in Applied Mathematics from the Polydisciplinary Faculty of Larache, Abdelmalek Essaadi University, Morocco. He is currently a Ph.D. student. His research interests include neural networks and metaheuristic optimization.

Abdelghni Lakehal is a professor of mathematics at Abdelmalek Essaadi University, Morocco. His research interests include applied mathematics, optimization, artificial intelligence, and metaheuristic algorithms. He has supervised and contributed to research works in these areas, with applications to computational intelligence and machine learning.