biomimetics

MDPI

Article

Aquila Optimization-Assisted Artificial Neural Network for Classification Problems

Gokhan Kayhan * and Seyma Hasbolat Unal

Department of Computer Engineering, Ondokuz Mayis University, Kurupelit, Samsun 55139, Türkiye; seyma.hasbolat@bil.omu.edu.tr

* Correspondence: gkayhan@omu.edu.tr

Abstract

Artificial Neural Networks (ANNs) are models that learn patterns in input-output data. Since traditional optimization methods often get trapped in local optima when determining weight and bias values, identifying optimal parameters and enhancing network performance remain significant research areas. Heuristic algorithms are also generally used in solving optimization problems and are used to train ANNs. In the study, the parameter optimization of the ANN model was carried out using the Aquila Optimizer (AO), a recent metaheuristic algorithm, and a hybrid Aquila Optimizer optimized ANN model (AOANN) was proposed. Hybridization of algorithms contributes to the improvement of optimization performance. In this study, the proposed model was assessed on empirical datasets, including Cancer, Iris, Glass, and Wine, and its performance was compared with that of well-established ANN models. The results of the evaluation revealed that the proposed AOANN, a soft computation model, demonstrated stability in solving classification problems.

Keywords: artificial neural networks; metaheuristic optimization; aquila optimizer; classification problems; machine learning

1. Introduction

In artificial neural networks (ANNs), a system is trained by using training data that has input values and their desired output values so that the system can learn [1]. The training consists in adjusting the connection weights so as to minimize this error of the output unit [2,3]. Different mathematical and stochastic approaches have been used to learning ANNs. It is noteworthy that the backpropagation (BP), Levenberg-Marquardt (LM) and Scaled Conjugate Gradient (SCG) are gradient based learning algorithms and most commonly used [4]. These algorithms update the connection weights by calculating and feeding back the error amount between ANN output and channeled current, from the output layer to the input layer, which aims to minimize the error based on ANN's output [3].

The hidden layer's activation function must be differentiable, as traditional BP algorithms involve derivative operations. These operations can be time-consuming and may get stuck in local optima in multimodal problems [5,6]. Due to these limitations, researchers have turned to metaheuristic optimization algorithms, which are more effective in global searches and stochastic approaches, to optimize ANN connection weights due to their random properties. For certain types of problems, metaheuristic algorithms have demonstrated greater effectiveness than conventional mathematical approaches in training ANNs [7].

Check for updates

Academic Editor: Heming Jia

Received: 11 March 2026
Revised: 26 March 2026
Accepted: 30 March 2026
Published: 2 April 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Optimization processes aim to minimize resource usage and determine the most appropriate solution methods for reaching the final problem states [8]. Metaheuristic optimization is designed to find global solutions for problems that are difficult or impossible to solve with mathematical optimization methods, conducting fewer trials and producing results close to the exact solution within an acceptable timeframe [9,10]. These algorithms draw inspiration from the instinctive behaviors of living organisms and natural phenomena for survival [11]. They generally involve diversification (exploration) for global searches and intensification (exploitation) for local searches to find the optimum solution in the search space [12]. During the exploration phase, global results are found by scattering randomly at various points in the search space, while more precise solutions are sought in a local region during the exploitation phase. The effectiveness of the algorithms is largely influenced by how well the exploration and exploitation parameters are balanced [10]. Various hybrid methodologies have been proposed in the literature to achieve an optimal balance between these phases. For instance, the search capability of the AO has been refined by employing the GWO's 'alpha' position as a guiding mechanism and incorporating quasi-opposition-based learning (QOBL) throughout the process. These integrations bolster the algorithm's stability within uncertain search landscapes, thereby facilitating high performance even in the presence of noise [13]. Alternatively, the Multi-Strategy Aquila Optimizer (MSAO) stands out in the literature as a robust approach to addressing the limitations of AO in high-dimensional spaces. By incorporating the dream-sharing strategy from the Dream Optimization Algorithm (DOA) and dynamic opposition-based learning, this method provides a reformulated equilibrium between global search and local refinement [14].

In recent years, hybrid systems combining ANNs and metaheuristic algorithms have become widespread. Literature reviews show that metaheuristic algorithms outperform traditional mathematical models in ANN weight training. The use of metaheuristic algorithms to optimize ANN parameters has accelerated over the last decade. In 2003, the Particle Swarm Optimization (PSO) was used in ANN training, achieving convergence with fewer errors and at a faster rate than the traditional BP algorithm, requiring six times fewer calculations and iterations [15]. ANN training with the Ant Colony Optimizer (ACO) for pattern classification in the medical field showed faster and lower error rates compared to traditional algorithms [16]. In 2007, the Artificial Bee Colony (ABC) algorithm optimized connection weights in signal-processing studies, outperforming PSO hybridized with an ANN and the Differential Evolution (DE) algorithm [17]. The TST metaheuristic algorithm, developed based on the Tabu Search (TS) algorithm with superior global search capability, achieved results five times faster than TS in MLP training [18]. Kattan et al. used an improved Harmony Search (HS) algorithm for MLP training, showing great success compared to traditional BP methods [19]. A neurogenetic approach combining the Genetic Algorithm (GA) and Neural Network (NN) effectively solved the Resource-Constrained Project Scheduling Problem (RCPSP), demonstrating high convergence success [20].

PSO, Imperialistic Competitive Algorithm (ICA), and traditional LM algorithms were tested in ANN training for parameter estimation affecting Stirling heat engines' performance, with metaheuristic algorithms showing high success [21]. Aljarah et al. used the Whale Optimization Algorithm (WOA) for MLP training, finding it had a high speed of convergence and accuracy compared to six metaheuristic algorithms and BP algorithms [22]. In another study, GA was used in MLP training to determine the shortest distance in vehicle routing problems, achieving 1% convergence as the best result in the literature by testing 7 benchmark sample problems [23]. The Butterfly Optimization Algorithm (BOA) was initially utilized for training a Multi-Layer Perceptron (MLP), where it exhibited effective

learning capability across various benchmark datasets(Iris, Balloon, Breast Cancer, Heart and XOR) achieving strong and consistent performance outcomes [24].

A hybrid method combining an ANN and the Cuckoo Search (CS) algorithm detected structural damages more accurately and faster than GA, PSO, and ANN algorithms alone [2]. The Jaya algorithm hybridized with an ANN correctly predicted crack lengths in engineering materials [25]. To classify medical data for disease detection, thirteen metaheuristic algorithms and gradient-based traditional LM algorithms were tested against the Equilibrium Optimization (EO) algorithm [4,11]. Hybrid metaheuristic algorithms, created by combining superior features of several metaheuristic algorithms, have become widespread for optimizing ANN weights to solve complex problems more successfully.

Tran-Ngoc et al. trained ANNs with a hybrid HGACS by combining CS and GA algorithms for damage detection in layered composite structures. This method achieved high accuracy with low computational cost due to its local search success, working parallel to the Gradient Descent (GD) method [26]. A face recognition system was developed using a Deep Neural Network (DNN) trained with a hybrid Colliding Bodies Optimization (CBO) and Multi-Verse Optimizer (MVO) algorithm, achieving an accuracy of 92.3% [27]. Alwaisi and Baykan [28] used the Shuffled Frog Leaping Algorithm (SFLA) to train ANNs for classification tasks, outperforming the conventional BP-centered algorithm in accuracy. AlRassas et al. developed the AO-ANFIS model by combining the AO algorithm and the Adaptive Neuro-Fuzzy Inference System (ANFIS) for time series forecasting in oil production, showing superior performance compared to other ANFIS hybrid methods [29]. This study used the AO metaheuristic algorithm to optimize ANN connection weights, addressing the limitations of traditional optimization methods prone to local optima. The AO metaheuristic algorithm [9] was evaluated on a set of benchmark functions and engineering problems, including 23 classical functions, 29 CEC2017 functions, 10 CEC2019 functions, and 7 engineering problems, demonstrating performance that is either superior to or comparable with other widely recognized metaheuristic methods. Additionally, empirical investigations of engineering problems showed the AO's capability in solving real-world applications. Aarthi et al. proposed a Deep Recurrent Neural Network model supported by the Aquila Optimization Algorithm for high-accuracy, category-based classification of online shaming content [30]. Ozden and Iseri combined the COOT algorithm and ANN model, developing the COOT-ANN hybrid method with successful results [31]. Hurtado-Mora et al. proposed a hybrid neuroevolutionary framework integrating GA and NEAT to improve neural network training and mitigate the local optimum problem during the learning process, demonstrating improved classification performance [32]. Kumar et al. proposed a hybrid metaheuristic algorithm, AO-HHO, that combines the global search capability of the AO with the local exploitation of Harris Hawks Optimization(HHO) to optimize Convolutional Neural Network(CNN) hyperparameters for brain tumor classification, achieving 87.34% accuracy on a dataset of 7,023 MRI images while reducing training time to 77.85 s, outperforming conventional methods such as PSO, GA, and WOA [33]. Unlike conventional approaches that employ metaheuristic algorithms solely for weight optimization, the integration of AO into the ANN in this study constitutes an original approach by embedding the optimization mechanism directly into the learning process. The main contributions of this study are summarized as follows:

- A novel hybrid ANN model, termed AOANN, is proposed.
- The AOANN model achieves effective and robust performance across different classification datasets.
- Compared to ANN models optimized using GD based methods, AOANN reduces the likelihood of becoming trapped in local minima.
- AOANN demonstrates high efficacy in solving classification problems.

- By leveraging heuristics and stochastic processes, AOANN offers a versatile alternative to traditional gradient-based optimization methods.

2. Materials and Methods

This section details the fundamental working principles of traditional gradient-based learning algorithms used in the weight training of ANNs and the metaheuristic AO algorithm. It summarizes the characteristics and performance metrics of the four datasets employed in our experimental study to evaluate the AO algorithm against alternative approaches. Furthermore, the inspiration behind the proposed AOANN model, its functioning, and the optimization process of the connection parameters of ANNs are explained.

2.1. Datasets

In this study, the performance of the AOANN method was assessed through experiments conducted on widely used datasets, including Wine, Breast Cancer, Iris, and Glass (Table 1). These datasets represent real-world applications, although at a moderate scale. All datasets were sourced from the UCI Machine Learning Repository [34].

Table 1. Specification of the datasets used for classification.

DataSet Name	Samples	Features	Classes
Wine	178	13	3
Breast Cancer	699	9	2
Iris	150	4	3
Glass	214	9	2

2.2. Learning Algorithms

In an ANN, many learning algorithms have working principles that are different from each other to produce the most proper output according to the input values provided to the network and optimize the connection weights to minimize error. These algorithms are used to make neural networks work in the fastest, most efficient, and most cost-effective way.

2.2.1. GD Optimization Algorithm

The GD is an algorithm often used in machine-learning problems to find the local/global minimum or maximum point by starting a function's starting point and moving gradually according to the determined step size (η). In the search for the minimum, the step direction is determined by the negative gradient of the objective function at the current position. In the GD method, the current position is updated, as in Equation (1), with each iteration by completing the derivative operation until the minimum point is reached.

$$X_{n+1} = X_n - \eta . \frac{\partial}{\partial X_n} f(X_n) \quad (1)$$

When we use the GD method in an ANN, the error propagation process consists of two stages: forward calculation and backward error propagation. Forward Calculation: The forward calculation of the input vector given as an input to the system in the ANN is performed with Equations (2) and (3) (X_i input values, Y_j^h hidden layer output values, Z_j output layer output values). The output values (Y_j) of the hidden layer neurons are represented by the forward calculation process in Equation (2) (number of hidden layers (h), number of hidden layer neurons (n), bias (b)):

$$Y_j^h = f(\sum_{i=1}^{n} X_i . W_{ij} + b_j^h) \quad (2)$$

The output values (Z_j) of the output-layer neurons are represented by the forward calculation process in Equation (3) (number of output neurons (k), bias (c)):

$$Z_j = f(\sum_{i=1}^{k} Y_i.V_{ij} + c_j) \tag{3}$$

The error of the network is determined as the difference between the desired output (P_i) and the current output (Z_i). The error function value (E_P) is obtained using the least-squares method from the multidimensional outputs resulting from an iteration, as Equation (4) shows.

$$E_P = \frac{1}{2}\sum_{i=1}^{k}(P_i - Z_i)^2 = \frac{1}{2}\sum_{i=1}^{k}(e_i)^2 \tag{4}$$

Backward Error Propagation: The E_P value obtained as a result of the forward calculation of the network is propagated back into the network, and the weights and bias are updated by taking the margin of error [35]. According to the GD method, weight update operations are performed as in Equations (5)–(8) (weight values (W_n), gradient of the network (g_n)):

$$W_{n+1} = W_n + \Delta W_n \tag{5}$$

$$\Delta W_n = -\eta.g_n \tag{6}$$

$$g_n = \frac{\partial E_P}{\partial W_n} \tag{7}$$

$$W_{n+1} = W_n - \eta.\frac{\partial}{\partial W_n}E_P(W) \tag{8}$$

Equation (9) shows the BP of the E_P from the output layer to the hidden layer.

$$\frac{\partial E_P}{\partial V_{ij}} = \frac{\partial E_P}{\partial Z_j}.\frac{\partial Z_j}{\partial Net_{Z_j}}.\frac{\partial Net_{Z_j}}{\partial V_{ij}} \tag{9}$$

Equation (10) shows the weight-update process between the hidden and the output layers.

$$V_{ij(new)} = V_{ij(former)} - \eta.\frac{\partial E_P}{\partial V_{ij}} \tag{10}$$

Equation (11) shows the BP of the E_P from the hidden layer.

$$\frac{\partial E_P}{\partial W_{ij}} = \frac{\partial E_P}{\partial V_{ij}}.(\frac{\partial V_{ij}}{\partial Y_j^h}.\frac{\partial Y_j^h}{\partial Net_{Y_j}}.\frac{\partial Net_{Y_j}}{\partial W_{ij}}) \tag{11}$$

Equation (12) shows the weight-update process between the input and hidden layers.

$$W_{ij_{(new)}} = W_{ij_{(former)}} - \eta.\frac{\partial E_P}{\partial W_{ij}} \tag{12}$$

The learning parameter (η) value used in the weight updates determines the optimal step length at the optimum point of finding. Choosing steps that are too small may cause the processes to be too slow, and choosing steps that are too large may cause a deviation from the result, bypassing the optimum point. The learning coefficient value is determined according to the problem type.

2.2.2. LM Optimization Algorithm

The LM algorithm is an optimization method that is often used to find an objective function's minimum point and reduces the processing load of the Newton optimization algorithm. Equation (13) shows the convergence process of Newton's algorithm to the optimum [36].

$$X_{n+1} = X_n - H_n^{-1} \cdot \frac{\partial}{\partial X_n} f(X_n) \tag{13}$$

In the LM algorithm, the simplified version of the Jacobian matrix (J) (see Equation (19)) is used instead of the Hessian matrix (H), which has a smaller processing load (see Equation (14)).

$$H = J^T * J \tag{14}$$

The LM algorithm uses Equations (15)–(18) to update connection weights in the training of an ANN [36] (weight values (W_n), gradient of the network (g_n), error value (e_n)):

$$g_n = J^T . e_n \tag{15}$$

$$\Delta W_n = -(J^T * J)^{-1} . g_n \tag{16}$$

$$W_{n+1} = W_n + \Delta W_n \tag{17}$$

$$W_{n+1} = W_n - (J^T * J)^{-1} . (J^T . e_n) \tag{18}$$

$$J = \begin{bmatrix} \frac{\partial e_{11}}{\partial W_1} & \frac{\partial e_{11}}{\partial W_2} & \cdots & \frac{\partial e_{11}}{\partial W_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{1M}}{\partial W_1} & \frac{\partial e_{1M}}{\partial W_2} & \cdots & \frac{\partial e_{1M}}{\partial W_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{P1}}{\partial W_1} & \frac{\partial e_{P1}}{\partial W_2} & \cdots & \frac{\partial e_{P1}}{\partial W_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{PM}}{\partial W_1} & \frac{\partial e_{PM}}{\partial W_2} & \cdots & \frac{\partial e_{PM}}{\partial W_N} \end{bmatrix} \tag{19}$$

To improve the LM algorithm's performance, Equation (20) is used instead of the Hessian matrix.

$$H = (J^T * J) + \mu . \mathrm{I} \tag{20}$$

I is the unit matrix, and μ is the Marquardt parameter. The μ parameter is a significant scalar value for the LM algorithm. This value takes variable values between 0 and 1, allowing LM to use the power of two algorithms. When the value is close to zero, the Newton algorithm acts as the gradient-descent algorithm when a large number is selected [37].

2.2.3. SCG Optimization Algorithm

The SCG algorithm, which is created by combining the classical conjugate gradient (CG) algorithm and some features of the LM algorithm, is an optimization algorithm that performs a search according to the conjugate directions to find the optimum point of the objective function in nonlinear problem solving [38,39]. In the SCG algorithm used to optimize an ANN, iterative weighting is used for the search direction by scaling the conjugate of the direction determined in the previous step and the second-degree derivative of the error function (Hessian matrix) as the step size [40]. The search direction and step size change with each iteration and are therefore not fixed [38]. The SCG algorithm updates connection weights in the training of an ANN using Equations (21)–(31), as described in [38,41]. The following symbols are used in the equations: W_n, the weight values of the network; ΔW_n, the weight change; g_n, the gradient of the network (direction of steepest

descent); α_n, the step size; D_n, the search direction (conjugate vector); σ_n, the weight change for the second derivative; λ_n, the parameter for editing Hessian uncertainty; and E_P, the error function. The weight change is calculated [42] by the scalar product of the step size and the search direction (see Equations (21) and (22)).

$$W_{n+1} = W_n + \Delta W_n \tag{21}$$

$$\Delta W_n = \alpha_n . D_n \tag{22}$$

The derivative of the E_P with respect to the W_n weights is calculated to determine g_n (as described in Equation (23)).

$$g_n = -\nabla E_P(W_n) = -E_P^{'}(W_n) \tag{23}$$

Initially, the direction vector, D_0, is determined as the negative gradient of the E_P (see Equation (24)). In other words, the algorithm starts searching in the steepest descent direction.

$$D_0 = -\nabla E_P(W_0) = -g_0 \tag{24}$$

In the first iteration, a search is performed in the D_0 direction. The next search direction, D_n, is calculated to be the conjugate of the earlier search direction (see Equation (25)).

$$D_n = -g_n + \beta_n * D_{n-1} \tag{25}$$

In the algorithm, the scaling parameters λ_n and σ_n, which the user initially determines for scaling the Hessian matrix, are selected as in Equations (26) and (27).

$$0 < \sigma_n \leq 10^{-4} \tag{26}$$

$$0 < \lambda_n \leq 10^{-6} \tag{27}$$

The Hessian matrix approximation, s_n, is calculated as in Equation (28).

$$s_n = \frac{E_P^{'}(W_n + \sigma_n . D_n) - E_P^{'}(W_n)}{\sigma_n} + \lambda_n . D_n \tag{28}$$

The step size, α_n, used in the weight update, is calculated as in Equation (29).

$$\alpha_n = \frac{{D_n}^T . g_n}{{D_n}^T . H . D_n + \lambda_n . ||D_n||^2} \tag{29}$$

The β_n factor used in the algorithm and the new search direction, D_{n+1}, are calculated as in Equations (30) and (31).

$$\beta_n = \frac{|g_{n+1}|^2 - {g_{n+1}}^T . g_n}{{g_n}^T . g_n} \tag{30}$$

$$D_{n+1} = -g_{n+1} + \beta_n * D_n \tag{31}$$

2.2.4. Aquila Optimization Algorithm

AO is a metaheuristic optimization algorithm inspired by the Aquila's hunting process of searching, surrounding, and attacking their prey [9]. Aquila optimization tries to find the optimum point of the objective function by following the Aquila's four hunting steps [9,43]. Figure 1 presents the flowchart of the AO model. The population-based AO algorithm starts problem-solving with candidate solutions represented by random numbers within a specific range of limit values (UB: Upper Bound and LB: Lower Bound). The candidate

solutions are initially determined using Equations (32) and (33). The candidate solutions that that make up the population with N elements are expressed as (X_i), the solution with the best fitness value that the candidate solutions have ever received are expressed as (X_{best}), and the population size is expressed as (D).

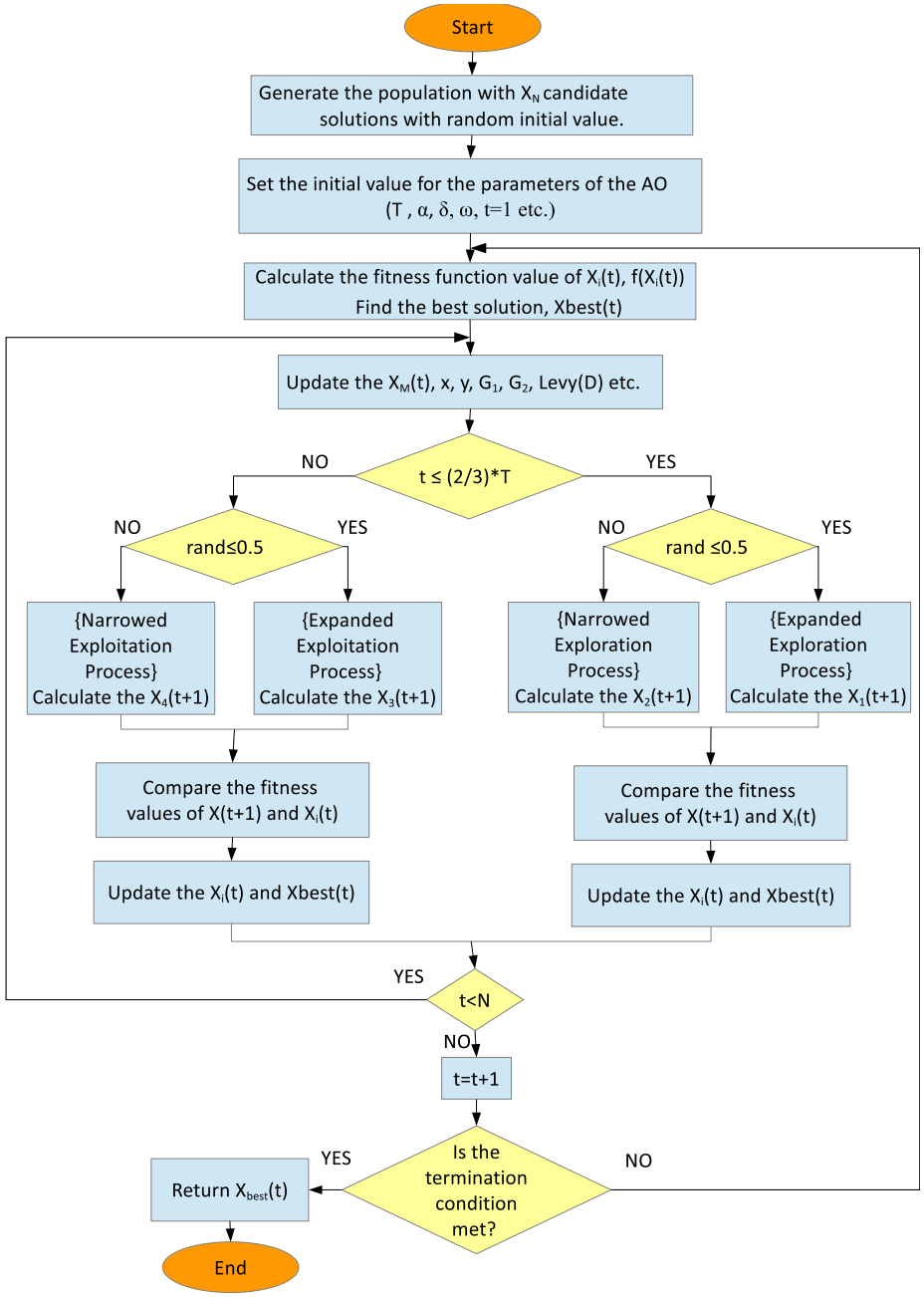


Figure 1. The Flowchart of the AO method.

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,D} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ X_{N,1} & X_{N,2} & \cdots & X_{N,D} \end{bmatrix} \quad (32)$$

$$X_{ij} = rand * (UB_j - LB_j) + LB_j \quad (33)$$

$$i = 1, 2, \ldots, N \; j = 1, 2, \ldots, D \quad (34)$$

In the AO algorithm, the positions of candidate solutions are updated at each iteration. The parameter update process consists of four consecutive phases, transitioning gradually from exploration to exploitation. Initially, the algorithm conducts a broad search to explore the solution space and then narrows the search within a focused region. In the later stages, attention shifts toward exploitation, where the search is further refined around promising areas. Let T represent the maximum number of iterations, and let t denote the current iteration index.; the exploration phase continues as long as the inequality in Equation (35) is satisfied. Once this condition no longer holds, the algorithm moves to the exploitation phase, corresponding to the local search process.

$$t \leq \frac{2}{3}.T \tag{35}$$

2.2.5. Expanded Exploration Process

The entire search region is extensively explored, and the suitability values are compared by scanning multiple regions for the optimum point. In the expanded exploration phase, the position values are updated as in Equation (36).

$$X_1(t+1) = X_{\text{best}}(t)\left(1 - \frac{t}{T}\right) + \left(X_M(t) - X_{\text{best}}(t)\right) \cdot \text{rand} \tag{36}$$

The rand value represents the random numbers between 0–1, and the X_M value represents the average value of the positions of all candidate solutions calculated in t iterations. It is calculated as in Equation (37).

$$X_M(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \tag{37}$$

$$\forall j = 1, 2, 3, \ldots, D \tag{38}$$

2.2.6. Narrowed Exploration Process

Exploration continues in a region that was narrowed in the earlier method in the search space. Spiral movements are made toward the optimum point [9]. In the narrowed exploration phase, the position values are updated as in Equation (39).

$$X_2(t+1) = X_{\text{best}}(t) \cdot \text{Levy}(D) + X_R(t) + (y - x) \cdot \text{rand} \tag{39}$$

The x and y values representing the spiral movements and the Levy flight equation are used at this stage. X_R(t) represents a randomly selected candidate solution in the t^{th} iteration. Levy flight is calculated as in Equation (40).

$$Levy(D) = s * \frac{u * \sigma}{|v|^{\frac{1}{\beta}}} \tag{40}$$

The u and v values are randomly selected between 0 and 1, and the s value is set as 0.01. The σ value is calculated as in Equation (41).

$$\sigma = \frac{\Gamma(1+\beta) * sin\frac{\pi\beta}{2}}{\Gamma(\frac{1+\beta}{2}) * \beta * 2^{(\frac{\beta-1}{2})}} \tag{41}$$

β is a fixed number with a value of 1.5. The calculation of the x and y values, which provide convergence to the optimum point by making spiral movements, is made using Equations (42)–(46).

$$x = \mathrm{r} * \cos\theta \tag{42}$$

$$y = \mathrm{r} * \sin\theta \tag{43}$$

$$r = \mathrm{r}_1 + \mathrm{U} * \mathrm{D}_1 \tag{44}$$

$$\theta = -\omega * \mathrm{D}_1 + \theta_1 \tag{45}$$

$$\theta_1 = \frac{3 * \pi}{2} \tag{46}$$

r_1 is a number from 1 to 20 that represents the number of cycles, U is a fixed number with a value of 0.00565, D_1 is an integer from 1 to problem size D, and ω is a fixed number with the value of 0.005.

2.2.7. Expanded Exploitation Process

As a result of the compression of the target solution into a narrow area, a detailed local search is started in that area. In the expanded exploitation phase, the position values are updated as in Equation (47).

$$X_3(t+1) = \left(X_{\text{best}}(t) - X_M(t)\right) \cdot \alpha - \text{rand} + \left((UB - LB) \cdot \text{rand} + LB\right) \cdot \delta \tag{47}$$

The parameters α and δ have a fixed value as small as 0.1.

2.2.8. Narrowed Exploitation Process

Local search operations are performed for the possible optimum solution, which is compressed into an exceedingly small region in the search space. In the narrowed exploitation phase, the position values are updated as in Equation (48).

$$X_4(t+1) = QF \cdot X_{\text{best}}(t) - G_1 \cdot X(t) \cdot \text{rand} - G_2 \cdot \text{Levy}(D) + \text{rand} \cdot G_1 \tag{48}$$

At this stage, the QF value, which is the quality function used to ensure balance in the exploration and exploitation processes; the G_1 value, which is used to determine random exploitation movements in the candidate solution's immediate vicinity; and the G_2 value, which decreases from 2 to 0, indicating the flight slope, are used in the method to track the best solution from the first position to the last position. QF, G_1, and G_2 values are calculated as in Equations (49)–(51).

$$QF(t) = t^{\frac{2*rand-1}{(1-T)^2}} \tag{49}$$

$$G_1 = 2 * rand - 1 \tag{50}$$

$$G_2 = 2 * (1 - \frac{t}{T}) \tag{51}$$

In the AO algorithm, the best solution value (X_{best}), as a result of all iterations, is the solution to the problem. The Aquila optimization algorithm's processing steps [29,43] are given in Algorithm 1.

2.2.9. Proposed AOANN Model

Gradient-based optimization algorithms are computationally efficient but are prone to getting stuck in local minima. On the other hand, metaheuristic algorithms often have higher computational complexity, but they have the advantage of avoiding local minima by searching a wider solution space. Gradient-based algorithms optimize model parameters

by iteratively updating them according to the gradient of the loss function with respect to those parameters. However, when the loss function has multiple local minima, these algorithms can get stuck in a suboptimal solution. Metaheuristic algorithms, on the other hand, employ a more diverse search strategy, often guided by heuristics, to find the global optimum solution. They do not rely solely on the gradient of the loss function, and therefore can explore regions of the solution space that gradient-based algorithms may miss. The AO, a powerful metaheuristic approach, has been employed to optimize the parameters of ANNs, leveraging the exploration and exploitation capabilities inherent in metaheuristic algorithms. The developed AOANN hybrid model ensures that the ANN used in problem solving approaches the solution in the fastest way possible with fewer errors and produces accurate and consistent results. Figure 2 shows the flowchart of the AOANN model. Algorithm 2 presents the processing steps of the developed AOANN.

Algorithm 1 AO Algorithm

Set the parameters (T, α, δ, ω, etc.) and generate random population, X_N
while (termination condition not met) **do**
  Calculate the fitness function value of X_i(t) and find the X_{best}(t)
  for (i=1,2, ..., N) **do**
    Calculate the X_M(t) and update parameters (x,y,G1,Levy(D) etc.)
    if $\left(t \leq \frac{2}{3} * T\right)$ **then**
      if $rand \leq 0.5$ **then**
        Expanded Exploration: $X_i(t) \xrightarrow{\text{Equation (36)}} X_1(t+1)$
        if f(X_1(t+1)) < f(X_i(t)) **then**
          X_i(t) ← X_1(t+1)
          if f(X_1(t+1)) < f(X_{best}) **then** X_{best} ← X_1(t+1)
          end if
        end if
      else
        Narrowed Exploration: $X_i(t) \xrightarrow{\text{Equation (39)}} X_2(t+1)$
        if f(X_2(t+1)) < f(X_i(t)) **then**
          X_i(t) ← X_2(t+1)
          if f(X_2(t+1)) < f(X_{best}) **then** X_{best} ← X_2(t+1)
          end if
        end if
      end if
    else
      if $rand \leq 0.5$ **then**
        Expanded Exploitation: $X_i(t) \xrightarrow{\text{Equation (47)}} X_3(t+1)$
        if f(X_3(t+1)) < f(X_i(t)) **then**
          X_i(t) ← X_3(t+1)
          if f(X_3(t+1)) < f(X_{best}) **then** X_{best} ← X_3(t+1)
          end if
        end if
      else
        Narrowed Exploitation: $X_i(t) \xrightarrow{\text{Equation (48)}} X_4(t+1)$
        if f(X_4(t+1)) < f(X_i(t)) **then**
          X_i(t) ← X_4(t+1)
          if f(X_4(t+1)) < f(X_{best}) **then** X_{best} ← X_4(t+1)
          end if
        end if
      end if
    end if
  end for
end while
Return X_{best}

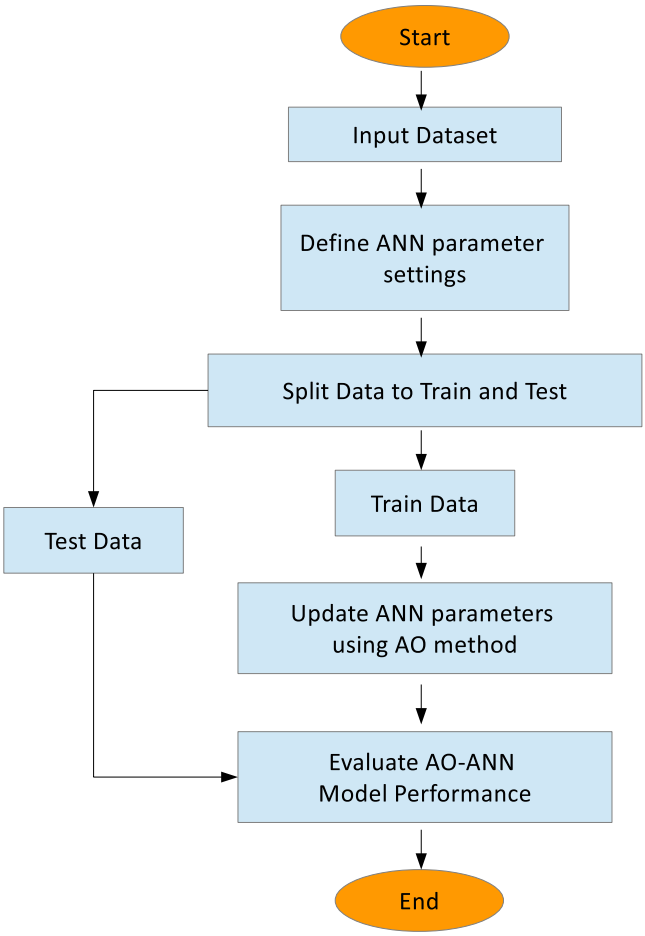


Figure 2. The Flowchart of the Proposed AOANN Model.

Algorithm 2 Proposed AOANN Algorithm

Inputs: Dataset, Number of candidate solutions (N) and max. iteration(T)
Build the ANN network and define ANN parameter settings
Generate a random population of N candidate solutions (X_{ND}) (D is dimension size of the solutions)
Set the parameters of the AO
Split Data to Train and Test
t ← 1
while $t \leq T$ **do**
    Determine the fitness function value of each X_i(t)
    Find the best solution, X_{best} (t)
    Update ANN parameters using AO method
    t ← t+1
end while
Return X_{best}
Evaluate AOANN Model Performance

2.2.10. Complexity Analysis of the AOANN Algorithm

This section provides a detailed complexity analysis of the AO algorithm utilized for parameter optimization in ANNs. Comprehending this complexity is pivotal for accurately assessing the computational requirements and performance efficacy of the optimization process.

Initialization Phase: During the initialization phase, the following operations are performed:

- Data and Parameter Initialization: The process of initially setting up the dataset and the ANN parameters typically exhibits a complexity of $O(B)$; this implies that the computational cost of the operation is proportional to the network's construction complexity, denoted as B.
- Random Population Generation: The process of creating a random initial population consisting of N possible solutions (candidate solutions), each with dimension D has a complexity of $O(N \times D)$.
- AO Parameters and Data Splitting: The processes of determining the fixed parameters of the AO algorithm and splitting the dataset into training and testing groups generally possess a constant time complexity of $O(1)$.

The total complexity for the initialization phase is:

$$O(B) + O(N \times D) + O(M)$$

where M denotes the size of the dataset.

Main Optimization Loop: In the main optimization loop, the following operations are carried out:

- Fitness Function Evaluation: Calculating the fitness value of every candidate solution in the population has a complexity of $O(N \times D)$.
- Best Solution Determination: Identifying the solution with the highest performance among all candidate solutions is typically executed with a complexity of $O(N)$.
- Updating ANN Parameters Using AO: Renewing the ANN parameters using the AO algorithm incurs a complexity cost of $O(N \times D)$ for each iteration.
- Incrementing Iteration Counter: The operation of advancing the iteration counter by one unit has a complexity of $O(1)$, as it is a constant-time operation.

The total complexity for the main optimization loop across T iterations is:

$$T \times (O(N \times D) + O(N) + O(N \times D)) = O(T \times N \times D)$$

where T denotes the maximum number of iterations.

Final Steps: The final steps include:

- Returning the Best Solution: This operation has a complexity of $O(1)$.
- Evaluating Model Performance: The performance evaluation of the AOANN model (ANN optimized by the AO) has a complexity of $O(E)$, where (E) is specific to the evaluation method and test conditions.

Overall Complexity Analysis: Combining all phases, the overall time complexity is:

$$O(B) + O(N \times D) + O(M) + O(T \times N \times D) + O(E)$$

where

- $O(B)$: Complexity of setting up the ANN;
- $O(N \times D)$: Complexity of generating the initial population and fitness evaluations;
- $O(M)$: Complexity of data splitting;
- $O(T \times N \times D)$: Complexity of the main optimization loop;
- $O(E)$: Complexity of performance evaluation.

For large values of T, N, and D, the term $O(T \times N \times D)$ dominates the overall complexity. This indicates that the AO algorithm significantly impacts the complexity and is the primary factor in the computational requirements of the optimization process.

The overall complexity grows proportionally with the iteration count, the number of candidate solutions, and the solution dimension. The AO algorithm, being the most

complex part of the process, is the key determinant of the optimization complexity, overshadowing other stages.

2.2.11. Performance Metrics

In classification problems, the confusion matrix (Table 2) is often used in testing the system with the connection weights obtained from the training. Multiple evaluation metrics can be calculated with this matrix, which shows the possible situations of the actual value and the estimated value in a dataset classified as positive and negative.

Table 2. Confusion Matrix.

	Predicted Positive	**Predicted Negative**
Actually Positive	True Positive (TP)	False Negative (FN)
Actually Negative	False Positive (FP)	True Negative (TN)

True Positive (TP) refers to instances correctly identified as positive, while True Negative (TN) represents cases accurately recognized as negative. In contrast, a False Positive (FP) arises when a negative sample is mistakenly classified as positive, whereas a False Negative (FN) occurs when a positive sample is incorrectly classified as negative [44]. Metric of Accuracy (ACC) is calculated [45–47] as the ratio of correctly predicted values to the number of all predictions (see Equation (52)).

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{52}$$

The F1-score is a performance evaluation metric obtained by computing the harmonic mean of the Recall and Precision values derived from the confusion matrix. It ranges from 0 to 1, where values approaching 1 signify a higher likelihood of model success [45,46,48]. The calculation of the F1-score is expressed in Equation (53) [47].

$$\text{F1-score} = 2 \cdot \frac{\mathrm{Recall} \cdot \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}} \tag{53}$$

Recall indicates how much of the data determined as positive is predicted as such [45,46]. It is calculated [47] as in Equation (54).

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{54}$$

Precision indicates how often the positively predicted values are actually positive [45,46]. It refers to the quality of the classification process [44]. It is calculated [47] as in Equation (55).

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{55}$$

The CE is an evaluation method that measures the difference between probability distributions belonging to different sets of events and allows for the measurement of model performance by being preferred as an error function in classification problems [49]. In

an ANN with K classes and N data, the CE value between the actual output (T) and the predicted output (Y) is calculated [50] as in Equation (56).

$$\mathrm{CE} = -\sum_{n=1}^{N}\sum_{i=1}^{K} T_{ni} \cdot \log(Y_{ni}) \tag{56}$$

Receiver Operating Characteristic (ROC) represents the curves drawn between the points (0, 0) and (1, 1) as a result of showing the FP-Rate (see Equation (57)) and TP-Rate (see Equation (58)) on the x- and y-axis, respectively. It is often preferred in machine learning (ML) to evaluate models' performance. The fact that the TP rate is high and the FP rate is low indicates that the models' performance is high [51].

$$\text{FP-Rate}\,(\mathrm{FPR}) = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}} \tag{57}$$

$$\text{TP-Rate}\,(\mathrm{TPR}) = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{58}$$

The Receiver Operating Characteristic Area Under the Curve (AUC) is used to evaluate the performance of classifier models and takes values between 0 and 1. A value close to 1 indicates that the model's predictions are highly accurate, a value of 0.5 indicates that the predictions are randomly distributed, and values close to 0 indicate that the predictions are made in the wrong direction [51].

Cohen's kappa (CK) is an evaluation method that shows the harmony between the actual output and the predicted output. Kappa takes values between −1 and +1. The fact that the kappa value is close to +1 in the model indicates that the randomness is low and that the harmony and success are high [52]. CK score is calculated as in Equations (59)–(63).

$$\mathrm{CK} = \frac{P_O - P_E}{1 - P_E} \tag{59}$$

$$P_O = \frac{\mathrm{TP}+\mathrm{TN}}{(\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN})} \tag{60}$$

$$d = \frac{(\mathrm{TNFP}).(\mathrm{TN}+\mathrm{FN})}{(\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN})^2} \tag{61}$$

$$y = \frac{(\mathrm{FN}+\mathrm{TP}).(\mathrm{FP}+\mathrm{TP})}{(\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN})^2} \tag{62}$$

$$P_E = d + y \tag{63}$$

3. Experimental Results

In this paper, a new hybrid ANN model with the AO was proposed and evaluated over classification datasets. The effectiveness of the proposed AOANN algorithm was tested by comparing it to the Levenberg-Marquardt optimized ANN (LMANN), Scaled Conjugate Gradient optimized ANN (SCGANN), and Gradient Descent optimized ANN (GDANN) algorithms. Experiments were performed using an Intel i5 processor using a 3.2 GHz clock and 8 GB RAM, while the models were implemented using MATLAB 2021 software. The model had a single hidden layer, and the neuron number at this layer was selected based on a trial-and-error methodology. Hence, 10 neurons were found as enough for the Cancer, Glass, and Iris datasets in the models; for the Wine dataset, three neurons were enough. During the training of the AOANN model, 100,000 iterations, LB = −5, UB = 5, and a population of 20 were fixed as parameter values. These are parameter values used at the initialization of the SCGANN, LMANN, and GDANN models and

running them for 1000 epochs. All these parameters were determined by trial and error. Metaheuristics are iterative search processes conducted on data towards the minimization of any objective function. Each iteration might be defined as a step in the search and hence involves a movement from one point to another; thus, it is different from epoch by the step carried out. The performance of the novel model was evaluated using four datasets: Wine, Cancer, Iris, and Glass. The rigorous evaluation included the K-Fold validation scheme with 5 folds throughout the entire training and subsequent testing procedure. The performances of the algorithms used for neural network training were measured by two primary metrics, ACC and Cross-Entropy (CE). Additionally, the models' test-phase performances were compared based on several complementary measures of evaluation, including the CK, F1-score, Receiver Operating Characteristic–Area Under the Curve (ROC-AUC), and Confusion Matrix. For each model, the average performance on its corresponding dataset was computed by taking the arithmetic mean of the CE and ACC values obtained across all folds. The aggregated results of these evaluations are shown in Table 3.

Table 3. Average Evaluation Results Across Datasets (5-Fold).

		Train Data		Test Data	
Dataset	**Method**	**ACC**	**CE**	**ACC**	**CE**
Wine	AO	1.0000	0.0071	0.9832	0.0307
	SCG	1.0000	0.0000	0.9662	0.0429
	LM	1.0000	0.0000	0.9775	0.0118
	GD	0.9298	0.1277	0.8871	0.1426
Cancer	AO	0.9971	0.0070	0.9542	0.1415
	SCG	1.0000	0.0000	0.9370	0.3981
	LM	0.9993	0.0007	0.9384	0.0579
	GD	0.9503	0.0714	0.9428	0.0776
Iris	AO	1.0000	0.0000	0.9533	0.4577
	SCG	1.0000	0.0000	0.9533	0.2718
	LM	1.0000	0.0000	0.9333	0.0368
	GD	0.9217	0.1371	0.9333	0.1408
Glass	AO	0.9953	0.0109	0.9441	0.1179
	SCG	1.0000	0.0000	0.9349	0.3413
	LM	1.0000	0.0000	0.9394	0.0570
	GD	0.9066	0.1221	0.9158	0.1231

The aim of the model training was CE, as generally it is preferred as a standard objective function in classification problems. By minimizing the CE, one gets better classification; hence, the lower the value, the better the performance, and a value of zero corresponds to no error. ACC is a metric that expresses the ratio of correctly predicted samples with respect to the total number of instances in the dataset, including true positives and true negatives. Its value varies between 0 and 1, where values closer to 1 correspond to superior classification performances.

As seen from Table 3, the AO method performed better compared to the other methods evaluated in this study. More specifically, the AOANN model performed with accuracy rates of 98.32% on the Wine dataset, 95.42% on the Cancer dataset, and 94.41% on the Glass dataset. On the Iris dataset, the highest test accuracy of 95.33% was achieved by both the AOANN and SCGANN models. The minimum CE values obtained during the training and test phases verify that the classification was done in a consistent and realistic manner. This observation can be seen more clearly in Table 4, which presents the results of a representative sample fold. In addition to the ACC and CE metrics, the F1-score further

justifies the proposed method, since the F1-score provides a more appropriate evaluation of the classification performance, especially in datasets with an imbalanced class distribution. Table 4 depicts the performance measures obtained on a randomly selected fold during the training phase, and thus allows for a comparative evaluation of the test data based on fold-wise ACC, F1-score, CK, and CE metrics. An examination of the F1-score and CK metrics along with ACC and CE results provides a better understanding of the performance of the proposed method, since these metrics are more informative, especially in multi-class and imbalanced datasets. Furthermore, the AUC, which refers to the area under the ROC curve, is widely recognized as a basic indicator of the effectiveness of any classifier. An AUC value of 1.0 corresponds to the best possible classification. Based on the test data given in Table 4, it can be noted that

- For the Wine dataset, the AOANN model demonstrated optimal performance, achieving the highest metrics across test ACC, CE, F1-score, and CK.
- On the Cancer dataset, while AOANN and GDANN exhibited identical test ACC values, the AOANN model yielded superior results for both the F1-score and CK metrics.
- The analysis of the Iris dataset indicated that both the AOANN and GDANN models achieved the maximum recorded values for test ACC, F1-score and CK.
- Furthermore, for the Glass dataset, the AOANN model once again produced the most favorable performance across all evaluated metrics: test ACC, CE, F1-score and CK.

Table 4. Sample Fold Evaluation Results for the Datasets.

		Train Data		**Test Data**			
Dataset	**Method**	**ACC**	**CE**	**ACC**	**CE**	**F1-Score**	**CK**
Cancer	AO	0.9893	0.0181	0.9714	0.0919	0.9687	0.9372
	SCG	1.0000	0.0000	0.9429	0.2905	0.9363	0.8706
	LM	0.9982	0.0018	0.9429	0.0518	0.9362	0.8719
	GD	0.9481	0.0667	0.9714	0.0576	0.9680	0.9366
Wine	AO	1.0000	0.0087	1.0000	0.0147	1.0000	1.0000
	SCG	1.0000	0.0000	0.9722	0.0547	0.9753	0.9590
	LM	1.0000	0.0000	0.9722	0.0218	0.9753	0.9590
	GD	0.9366	0.1508	0.9722	0.1476	0.9721	0.9577
Iris	AO	1.0000	0.0000	1.0000	0.0000	1.0000	1.0000
	SCG	1.0000	0.0000	0.9667	0.0134	0.9681	0.9500
	LM	1.0000	0.0000	0.9667	0.0220	0.9681	0.9500
	GD	0.9500	0.1337	1.0000	0.1375	1.0000	1.0000
Glass	AO	0.9883	0.0168	0.9767	0.0494	0.9695	0.9370
	SCG	1.0000	0.0000	0.9302	0.2035	0.9063	0.8111
	LM	1.0000	0.0000	0.9767	0.0235	0.9695	0.9370
	GD	0.9123	0.0955	0.9070	0.0919	0.8812	0.7564

The classification performance of the proposed AOANN model on the Wine dataset, which contains three classes, is confirmed through the confusion matrix presented in Figure 3. The matrices also highlight the classification inaccuracies of the alternative models. Specifically, the SCGANN and LMANN networks incorrectly labeled some class 2 instances as class 1, whereas the GDANN model confused certain class 3 samples with class 2. As illustrated in Figure 4, which shows the ROC curves and AUC values for all methods, the AOANN and SCGANN models demonstrate superior performance, while GDANN performs the weakest. Because the Wine dataset is balanced, the F1-score metric effectively reflects classification success. Likewise, the AOANN model's classification accuracy on the Cancer dataset (a binary classification task) is validated by the confusion

matrix presented in Figure 5. This matrix also discloses the error distribution among the other models: AOANN and GDANN each produced 4 misclassifications, whereas SCGANN and LMANN misclassified 8 samples. According to the ROC curves and AUC values displayed in Figure 6, AOANN shows a slightly lower AUC than the other methods. Nevertheless, due to the class imbalance in this dataset, the F1-score provides a more meaningful indicator of model performance, as the ROC curve represents an average across all thresholds. Finally, for the Iris dataset, another three-class dataset, the AOANN model's effectiveness is validated by the confusion matrix in Figure 7. The same figure also reveals that SCGANN and LMANN misclassified several class 3 instances as class 2, indicating a consistent pattern in their classification tendencies.

Figure 8 presents the ROC curves and corresponding AUC values for all methods evaluated on the Iris dataset. As illustrated in the figure, the proposed AOANN model exhibits classification performance comparable to that of the SCGANN, GDANN, and LMANN models. Figure 9 shows the confusion matrices of the AOANN model and the remaining models evaluated on the Glass dataset. This matrix confirms the accuracy achieved by the AOANN model on the Glass dataset, which represents a binary classification task. Moreover, it reveals the distribution of misclassifications among the competing models. Specifically, the AOANN and LMANN models each produced one incorrect classification, while SCGANN and GDANN resulted in three and four errors, respectively. Figure 10 presents the ROC curves and corresponding AUC values for all methods applied to the Glass dataset. The results clearly show that the proposed AOANN model attains the best performance, while the GDANN model exhibits the weakest classification ability.

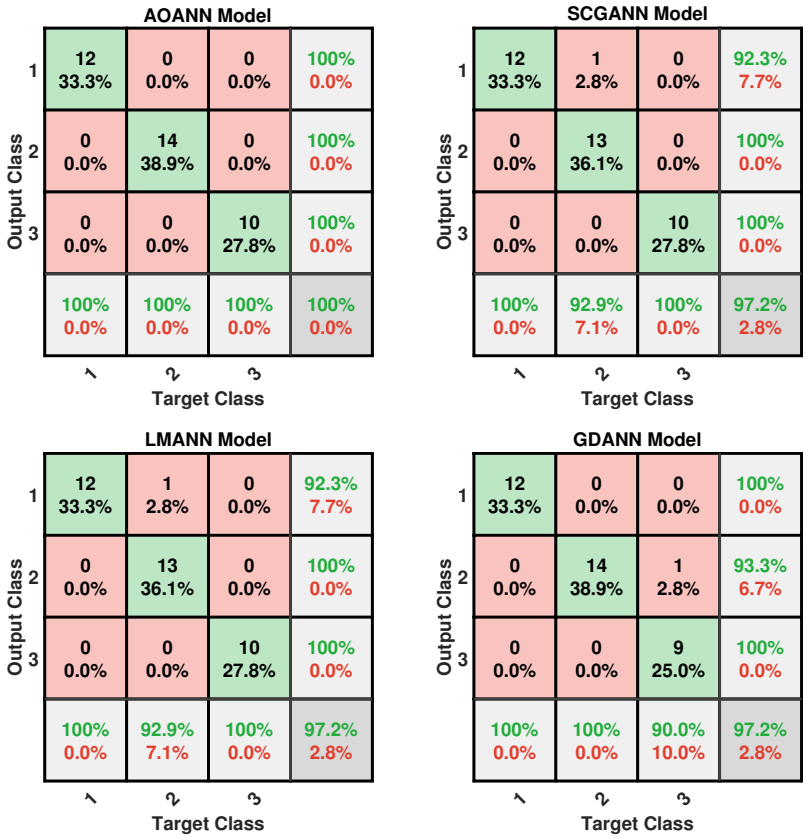


Figure 3. Confusion matrix for the Wine test dataset across the evaluated models.

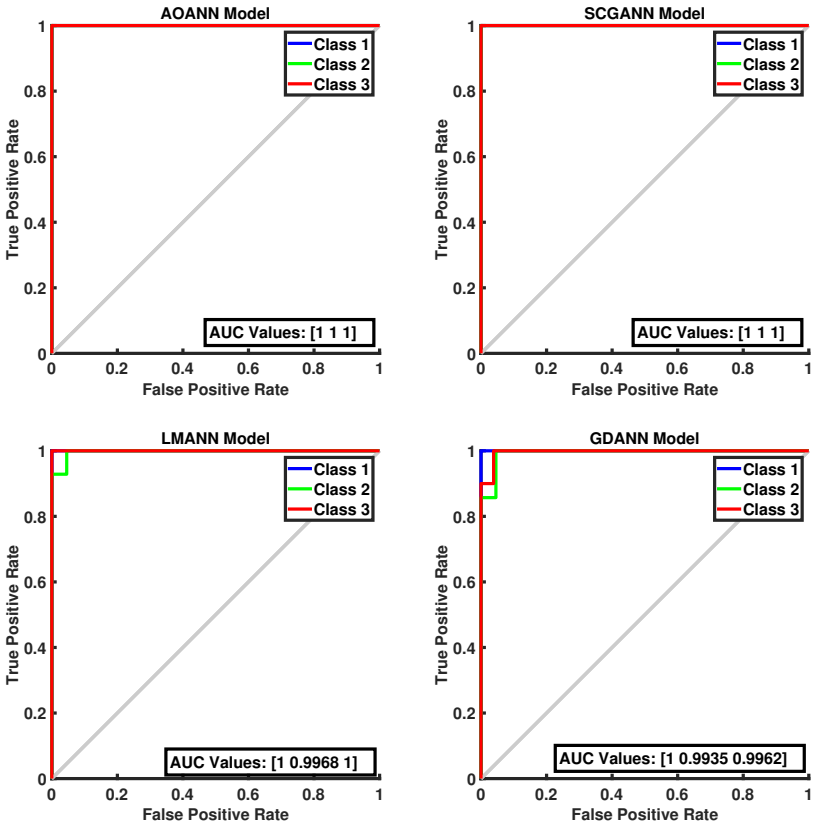


Figure 4. ROC Curve for the Wine test dataset across the evaluated models.



Figure 5. Confusion matrix for the Breast Cancer test dataset across the evaluated models.

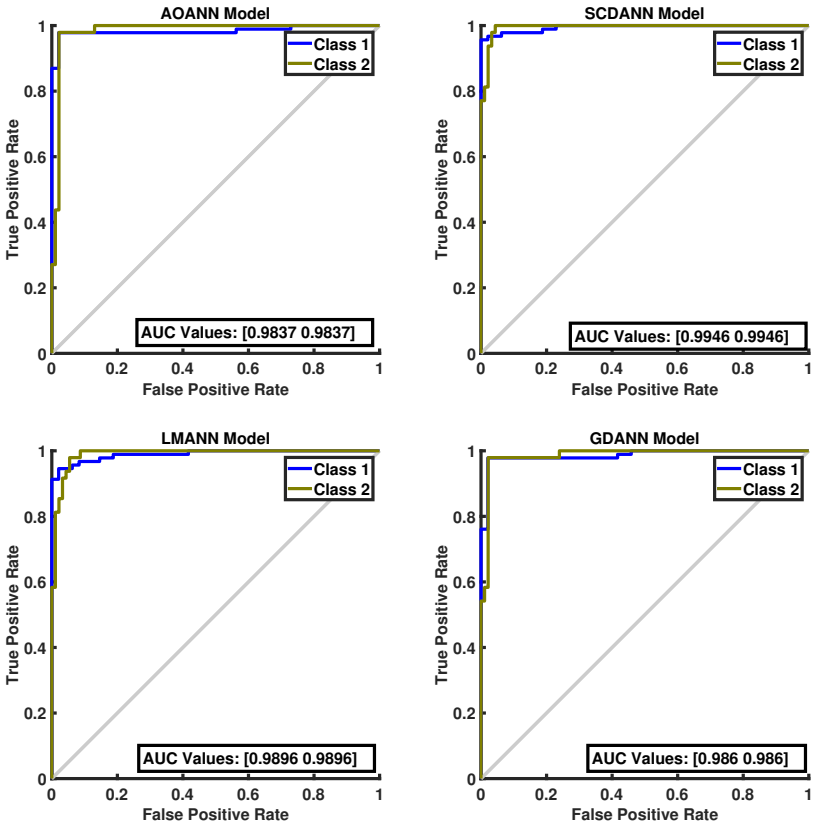


Figure 6. ROC Curve for the Breast Cancer test dataset across the evaluated models.



Figure 7. Confusion matrix for the Iris test dataset across the evaluated models.

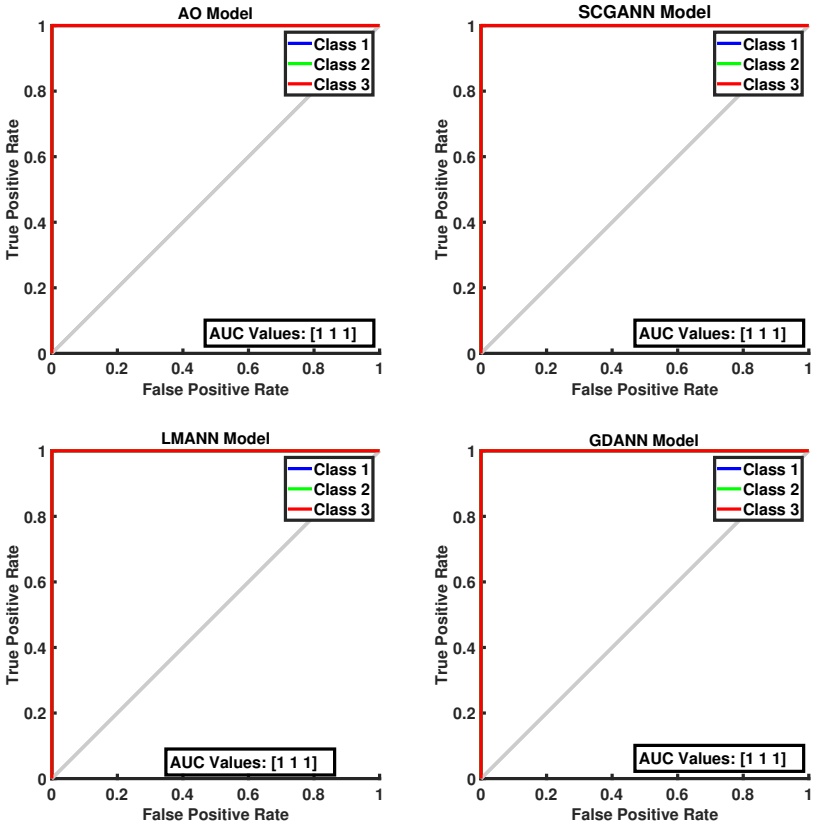


Figure 8. ROC Curve for the Iris test dataset across the evaluated models.



Figure 9. Confusion matrix for the Glass test dataset across the evaluated models.

The proposed AOANN model consistently achieves superior accuracy and lower error rates across all benchmarks. Although it requires more computational effort than gradient-based baselines, this is a deliberate trade-off justified by its ability to avoid local minima and establish stable classification boundaries. As summarized in Table 5, the synergy between the AO and ANN architecture ensures a robust training process, prioritizing solution quality over execution speed.

Table 5. Summary of Performance Metrics and Evolutionary Advantages of AOANN.

Metric	AOANN Advantage	Academic Significance
ACC	Highest classification rates	Superior discriminative capacity
CE	Lowest variance values	Model stability
AUC/ROC	Values nearest to 1.0	Maximum class separation power
F1-Score	Balanced precision and recall	Reliability in sensitive diagnostics

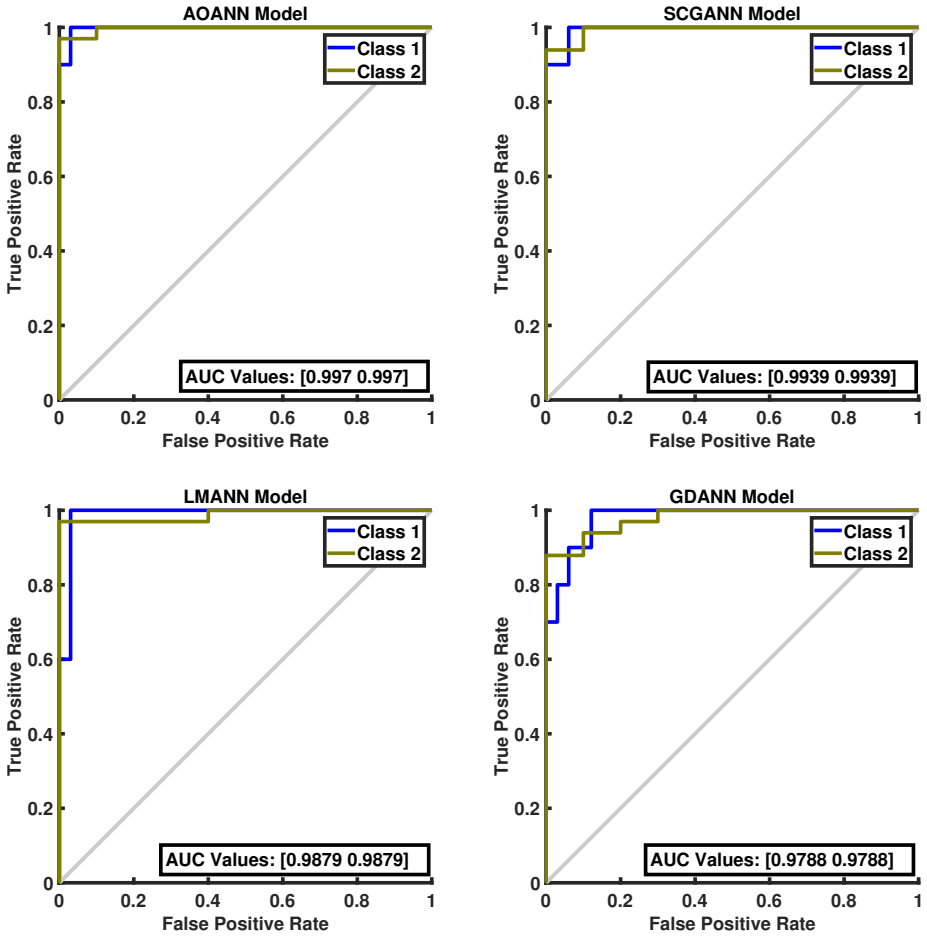


Figure 10. ROC Curve for the Glass test dataset across the evaluated models.

4. Conclusions and Discussion

While gradient-based algorithms are computationally efficient, metaheuristic algorithms possess the advantage of escaping local minima, making them particularly suitable for optimization problems where gradient-based methods may fail. Therefore, the trade-off between computational complexity and the ability to avoid local minima must be carefully considered when selecting an appropriate optimization algorithm for a given problem. In this study, an enhanced hybrid ANN model, referred to as AOANN, was introduced to effectively handle both binary and multi-class classification tasks. The model integrates the recently developed metaheuristic optimization algorithm, AO, for optimizing ANN

parameters. The effectiveness of the proposed AOANN model was assessed using four benchmark datasets (Wine, Iris, Breast Cancer and Glass) obtained from the UCI repository and compared with well-known models such as LMANN, SCGANN, and GDANN. Experimental results revealed that AOANN achieved perfect ACC (100%) on the Iris and Wine test datasets, while attaining 97.67% and 97.14% accuracy on the Glass and Breast Cancer test datasets, respectively. Moreover, the AOANN model achieved notably high F1-score and CK values across all test datasets compared to other methods. Specifically, the F1-score and CK values were both 1.0 for the Iris and Wine datasets. For the Glass dataset, these values were 0.9695 and 0.9370, respectively, while for the Breast Cancer dataset they were 0.9687 and 0.9372. These results are consistent with the theoretical expectation that metaheuristic-based optimization improves generalization performance by enabling a better balance between exploration and exploitation, thereby leading to more robust and accurate classification outcomes. Overall, the findings demonstrate that the AOANN model outperforms other algorithms across all evaluated datasets.

In future work, the proposed model will be further developed to address regression problems and to optimize hyperparameters in deep learning architectures. Furthermore, a more comprehensive parameter sensitivity analysis could yield deeper insights into the model's behavior. Therefore, we intend to incorporate such an investigation in future work to more thoroughly examine how different parameters influence the performance of the AOANN model. Additionally, future studies could evaluate the AOANN model on higher-dimensional datasets to further explore its scalability and performance on large-scale non-linear problems. Finally, a memetic approach integrating the AO for global exploration with gradient-based methods for local fine-tuning will be explored to further enhance the training efficiency of neural architectures.

Author Contributions: Conceptualization, G.K.; methodology, G.K.; software, G.K. and S.H.U.; writing—original draft preparation, S.H.U.; writing—review and editing, G.K.; supervision, G.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/index.php]; accessed on 10 March 2026.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AO	Aquila Optimizer
ANN	Artificial Neural Network
AOANN	Aquila Optimizer optimized ANN
SCGANN	Scaled Conjugate Gradient optimized ANN
LMANN	Levenberg–Marquardt optimized ANN
GDANN	Gradient Descent optimized ANN
UCI	University of California, Irvine (Machine Learning Repository)
ML	Machine Learning
BP	Backpropagation
CNN	Convolutional Neural Network
GA	Genetic Algorithm

PSO	Particle Swarm Optimization
SCG	Scaled Conjugate Gradient
LM	Levenberg–Marquardt
GD	Gradient Descent
ACC	Metric of Accuracy
CE	Cross-Entropy
CK	Cohen's Kappa

References

1. Mazlan, A.U.; Sahabudin, N.A.B.; Remli, M.A.; Ismail, N.S.N.; Mohamad, M.S.; Warif, N.B.A. Supervised and Unsupervised Machine Learning for Cancer Classification: Recent Development. In Proceedings of the 2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), Shah Alam, Malaysia, 26 June 2021; pp. 392–395. [CrossRef]
2. Tran-Ngoc, H.; Khatir, S.; De Roeck, G.; Bui-Tien, T.; Abdel Wahab, M. An efficient artificial neural network for damage detection in bridges and beam-like structures by improving training parameters using cuckoo search algorithm. *Eng. Struct.* **2019**, *199*, 109637. [CrossRef]
3. Whittington, J.; Bogacz, R. Theories of Error Back-Propagation in the Brain. *Trends Cogn. Sci.* **2019**, *23*, 235–250. [CrossRef] [PubMed]
4. Tapas, S.; Jayri, B.; Péricles, B.C.M. Artificial Neural Network training using metaheuristics for medical data classification: An experimental study. *Expert Syst. Appl.* **2022**, *193*, 116423. [CrossRef]
5. Parthdev, B.; Vaishali. Back Propogation Algorithm: An Artificial Neural Network Approach. *Int. J. Eng. Res. Technol.* **2018**, *5*, 1–6.
6. Fei, H.; Jing, J.; Qinghua, L.; Benyue, S. A survey on metaheuristic optimization for random single-hidden layer feedforward neural network. *Neurocomputing* **2019**, *335*, 261–273. [CrossRef]
7. Hemeida, A.M.; Hassan, S.A.; Mohamed, A.A.A.; Alkhalaf, S.; Mahmoud, M.M.M.; Senjyu, T.S.; Eldin, A.M.B. Nature-inspired algorithms for feed-forward neural network classifiers: A survey of one decade of research. *Ain Shams Eng. J.* **2020**, *11*, 659–675. [CrossRef]
8. Hajipour, V.; Kheirkhah, A.; Tavana, M.; Absi, N. Novel Pareto-Based Meta-Heuristics for Solving Multi-Objective Multi-Item Capacitated Lot-Sizing Problems. *Int. J. Adv. Manuf. Technol.* **2015**, *80*, 31–45. [CrossRef]
9. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-qaness, M.A.; Gandomi, A.H. Aquila Optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
10. Salcedo-Sanz, S. Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures. *Phys. Rep.* **2016**, *655*, 1–70. [CrossRef]
11. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl.-Based Syst.* **2020**, *191*, 105190. [CrossRef]
12. Abualigah, L.; Diabat, A.; Geem, Z.W. A Comprehensive Survey of the Harmony Search Algorithm in Clustering Applications. *Appl. Sci.* **2020**, *10*, 3827. [CrossRef]
13. Varshney, M.; Kumar, P.; Ali, M.; Gulzar, Y. Using the Grey Wolf Aquila Synergistic Algorithm for Design Problems in Structural Engineering. *Biomimetics* **2024**, *9*, 54. [CrossRef] [PubMed]
14. Kan, H.; Xiao, Y.; Gao, Z.; Zhang, X. Improved Multi-Strategy Aquila Optimizer for Engineering Optimization Problems. *Biomimetics* **2025**, *10*, 620. [CrossRef]
15. Gudise, V.; Venayagamoorthy, G. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In Proceedings of the 2003 IEEE Swarm Intelligence Symposium, SIS'03 (Cat. No.03EX706), Indianapolis, IN, USA, 24–26 April 2003; pp. 110–117. [CrossRef]
16. Blum, C.; Socha, K. Training feed-forward neural networks with ant colony optimization: An application to pattern classification. In Proceedings of the Fifth International Conference on Hybrid Intelligent Systems, HIS 2005, Rio de Janeiro, Brazil, 6–9 November 2005. [CrossRef]
17. Karaboga, D.; Akay, B. Artificial Bee Colony (ABC) Algorithm on Training Artificial Neural Networks. In Proceedings of the 2007 IEEE 15th Signal Processing and Communications Applications, Eskisehir, Turkey, 11–13 June 2007; pp. 1–4. [CrossRef]
18. Dengiz, B.; Alabas-Uslu, C.; Dengiz, O. A tabu search algorithm for the training of neural networks. *J. Oper. Res. Soc.* **2009**, *60*, 282–291. [CrossRef]
19. Kattan, A.; Abdullah, R.; Abdul Salam, R. Harmony Search Based Supervised Training of Artificial Neural Networks. In Proceedings of the 2010 International Conference on Intelligent Systems, Modelling and Simulation, Liverpool, UK, 27–29 January 2010; pp. 105–110. [CrossRef]
20. Agarwal, A.; Colak, S.; Erenguc, S. A Neurogenetic approach for the resource-constrained project scheduling problem. *Comput. Oper. Res.* **2011**, *38*, 44–50. [CrossRef]

21. Toghyani, S.; Ahmadi, M.H.; Kasaeian, A.; Mohammadi, A.H. Artificial neural network, ANN-PSO and ANN-ICA for modelling the Stirling engine. *Int. J. Ambient. Energy* **2016**, *37*, 456–468. [CrossRef]
22. Aljarah, I.; Faris, H.; Mirjalili, S. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Comput.* **2018**, *22*, 1–15. [CrossRef]
23. Apak, G. Çoklu Depolu araç Rotalama Probleminin Hibrid Algoritmalar Yöntemiyle çözülmesi/Solving Multi-Depot Vehicle Routing Problems via Hybrid Algorithms. Master's Thesis, Çukurova University, Institute of Social Sciences, Department of Management, Adana, Türkiye, 2018.
24. Irmak, B.; Gulcu, S. Training of the feed-forward artificial neural networks using butterfly optimization algorithm. *MANAS J. Eng.* **2021**, *9*, 160–168. [CrossRef]
25. Khatir, S.; Boutchicha, D.; Le Thanh, C.; Tran-Ngoc, H.; Nguyen, T.; Abdel-Wahab, M. Improved ANN technique combined with Jaya algorithm for crack identification in plates using XIGA and experimental analysis. *Theor. Appl. Fract. Mech.* **2020**, *107*, 102554. [CrossRef]
26. Tran-Ngoc, H.; Khatir, S.; Ho-Khac, H.; De Roeck, G.; Bui-Tien, T.; Wahab, M.A. Efficient Artificial neural networks based on a hybrid metaheuristic optimization algorithm for damage detection in laminated composite structures. *Compos. Struct.* **2021**, *262*, 113339. [CrossRef]
27. Soni, N.; Sharma, E.K.; Kapoor, A. Hybrid meta-heuristic algorithm based deep neural network for face recognition. *J. Comput. Sci.* **2021**, *51*, 101352. [CrossRef]
28. Alwaisi, S.S.A.; Baykan, O.K. Training of Artificial Neural Network Using Metaheuristic Algorithm. *Int. J. Intell. Syst. Appl. Eng.* **2017**, *5*, 12–16. [CrossRef]
29. AlRassas, A.M.; Al-qaness, M.A.; Ewees, A.A.; Ren, S.; Abd Elaziz, M.; Damaševičius, R.; Krilavičius, T. Optimized ANFIS model using Aquila Optimizer for oil production forecasting. *Processes* **2021**, *9*, 1194. [CrossRef]
30. Aarthi, B.; Chelliah, B.J. Deep recurrent neural network-based Aquila optimization-based online shaming emotion analysis. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6882. [CrossRef]
31. Ozden, A.; Iseri, I. COOT optimization algorithm on training artificial neural networks. *Knowl. Inf. Syst.* **2023**, *65*, 3353–3383. [CrossRef]
32. Hurtado-Mora, H.A.; Herrera-Barajas, L.A.; González-del Ángel, L.J.; Pichardo-Ramírez, R.; García-Ruiz, A.H.; Lira-García, K.E. Enhancing Neural Network Training Through Neuroevolutionary Models: A Hybrid Approach to Classification Optimization. *Mathematics* **2025**, *13*, 1114. [CrossRef]
33. Kumar, M.; Mohd, N.; Shivam, G.; Goyal, A.; Parashar, D.; Khan, R. Hybrid Aquila optimizer–Harris Hawks optimization for CNN hyperparameter tuning in brain tumor classification. *Sci. Rep.* **2026**. [CrossRef]
34. Blake, C.; Merz, C. *UCI Repository of ML Databases*; University of California: Irvine, CA, USA, 1998.
35. Yang, G.R.; Wang, X.J. Artificial neural networks for neuroscientists: A primer. *Neuron* **2020**, *107*, 1048–1070. [CrossRef]
36. Mishra, S.; Prusty, R.; Hota, P.K. Analysis of Levenberg-Marquardt and Scaled Conjugate gradient training algorithms for artificial neural network based LS and MMSE estimated channel equalizers. In Proceedings of the 2015 International Conference on Man and Machine Interfacing (MAMI), Bhubaneswar, India, 17–19 December 2015; pp. 1–7. [CrossRef]
37. Wilamowski, B.M. Neural network architectures and learning algorithms. *IEEE Ind. Electron. Mag.* **2009**, *3*, 56–63. [CrossRef]
38. Møller, M.F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.* **1993**, *6*, 525–533. [CrossRef]
39. Sivasankari, K.; Thanushkodi, K.; Kalaivanan, K. Automated seizure detection using multilayer feed forward network trained using scaled conjugate gradient method. In Proceedings of the 2013 International Conference on Current Trends in Engineering and Technology (ICCTET), Coimbatore, India, 3 July 2013; pp. 195–198.
40. Nayak, S.; Kumar, N.; Choudhury, B. Scaled conjugate gradient backpropagation algorithm for selection of industrial robots. *Int. J. Comput. Appl.* **2017**, *7*, 92–101. [CrossRef]
41. Waheeb, W.; Ghazali, R.; Deris, M.M. Content-based SMS spam filtering based on the scaled conjugate gradient backpropagation algorithm. In Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, China, 15–17 August 2015; pp. 675–680.
42. Babani, L.; Jadhav, S.; Chaudhari, B. Scaled conjugate gradient based adaptive ANN control for SVM-DTC induction motor drive. In Proceedings of the 2th IFIP WG 12.5 International Conference and Workshops, AIAI 2016, Thessaloniki, Greece, 16–18 September 2016; pp. 384–395.
43. Wang, S.; Jia, H.; Abualigah, L.; Liu, Q.; Zheng, R. An improved hybrid aquila optimizer and harris hawks algorithm for solving industrial engineering optimization problems. *Processes* **2021**, *9*, 1551. [CrossRef]
44. Al-Imran, M.; Ripon, S.H. Network Intrusion Detection: An Analytical Assessment Using Deep Learning and State-of-the-Art Machine Learning Models. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 200. [CrossRef]
45. Huang, J.; He, R.; Chen, J.; Li, S.; Deng, Y.; Wu, X. Boosting Advanced Nasopharyngeal Carcinoma Stage Prediction Using a Two-Stage Classification Framework Based on Deep Learning. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 184. [CrossRef]

46. Ni, Q.; Fan, Z.; Zhang, L.; Zhang, B.; Zheng, X.; Zhang, Y. Daily Activity Recognition and Tremor Quantification from Accelerometer Data for Patients with Essential Tremor Using Stacked Denoising Autoencoders. *Int. J. Comput. Intell. Syst.* **2022**, *15*, 1. [CrossRef]
47. Rezvi, M.A.; Moontaha, S.; Trisha, K.A.; Cynthia, S.T.; Ripon, S. Data mining approach to analyzing intrusion detection of wireless sensor network. *Indones. J. Electric. Eng. Comput. Sci.* **2021**, *21*, 516–523.
48. Zhao, W.; Tian, H.; Wu, Y.; Cui, Z.; Feng, T. A New Item-Based Collaborative Filtering Algorithm to Improve the Accuracy of Prediction in Sparse Data. *Int. J. Comput. Intell. Syst.* **2022**, *15*, 15. [CrossRef]
49. Wang, H.; Wu, F.; Lei, D. Prediction of Ship Heave Motion Using Regularized BP Neural Network with Cross Entropy Error Function. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 192. [CrossRef]
50. Liu, S.; Tan, N.; Yang, H.; Lukač, N. An Intelligent Question Answering System of the Liao Dynasty Based on Knowledge Graph. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 170. [CrossRef]
51. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [CrossRef]
52. Sadighi, M.; Motamedvaziri, B.; Ahmadi, H.; Moeini, A. Assessing landslide susceptibility using machine learning models: A comparison between ANN, ANFIS, and ANFIS-ICA. *Environ. Earth Sci.* **2020**, *79*, 536. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.