International Journal of Industrial Engineering Computations 13 (2022) 617–640

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

# Optimizing the learning process of multi-layer perceptrons using a hybrid algorithm based on MVO and SA

**Ömer Yılmaz[a*], Adem Alpaslan Altun[b] and Murat Köklü[b]**

[a]*Department of Information Technologies, Tokat Vocational and Technical Anatolian High School, 60100, Tokat, Turkey*
[b]*Department of Computer Engineering, Faculty of Technology, Konya Selcuk University, 42130, Konya, Turkey*

**C H R O N I C L E**

*Article history*:
Received November 10 2021
Received in Revised Format May 2 2022
Accepted May 19 2022
Available online May, 19 2022

Keywords:
*Optimization*
*Training neural network*
*Multi-layer perceptron*
*Meta-heuristic algorithms*
*Hybrid optimization algorithm*
*Simulated annealing*
*Multi-verse optimizer*

**A B S T R A C T**

Artificial neural networks (ANNs) are one of the artificial intelligence techniques used in real-world problems and applications encountered in almost all industries such as education, health, chemistry, food, informatics, logistics, transportation. ANN is widely used in many techniques such as optimization, modelling, classification and forecasting, and many empirical studies have been carried out in areas such as planning, inventory management, maintenance, quality control, econometrics, supply chain management and logistics related to ANN. The most important and just as hard stage of ANNs is the learning process. This process is about finding optimal values in the search space for different datasets. In this process, the values generated by training algorithms are used as network parameters and are directly effective in the success of the neural network (NN). In classical training techniques, problems such as local optimum and slow convergence are encountered. Meta-heuristic algorithms for the training of ANNs in the face of this negative situation have been used in many studies as an alternative. In this study, a new hybrid algorithm namely MVOSANN is suggested for the training of ANNs, using Simulated annealing (SA) and Multi-verse optimizer (MVO) algorithms. The suggested MVOSANN algorithm has been experimented on 12 prevalently classification datasets. The productivity of MVOSANN has been compared with 12 well-recognized and current meta-heuristic algorithms. Experimental results show that MVOSANN produces very successful and competitive results.



## 1. Introduction

In recent years, meta-heuristic algorithms have been frequently used in computational methods to increase efficiency and quality, reduce costs, and solve various complex problems encountered during the development and management of business processes by using production resources at an optimum level. Although the best solution cannot always be found with meta-heuristic algorithms, they are preferred by researchers because they have features such as ease of application, their effectiveness in complex problems, adaptability to different problems, and so on (Kaya & Fığlalı 2018; Talbi, 2009). The common goal in such algorithms is to get the best solution in the existing circumstances for difficult and complex problems. To obtain the best result, the exploration and exploitation features of the algorithms should be strong, and the balance should be struck between these two processes.

One way to equilibrate between exploration and exploitation may be to combine algorithms (Mirjalili & Hashim, 2010; Mafarja & Mirjalili, 2017). A hybrid algorithm is to combine one algorithm with another algorithm or algorithms using the preferred features of the algorithms. In many studies, it is seen that methods such as improvement, modification and hybridization have been tried to increase the success of algorithms. Hybrid algorithms can be essentially classified into two

* Corresponding author
E-mail: omerryilmaz@gmail.com (Ö. Yılmaz)

2022 Growing Science Ltd.
doi: 10.5267/j.ijiec.2022.5.003

groups. The first group is integrative hybrids, which are constituted by integrating a subsidiary algorithm into a primary algorithm. The second group is collaborative hybrids, a combination of two or more algorithms that run sequentially or in parallel (Ting et al., 2015). Many studies have been carried out on functions such as function optimization and real-world problems with hybrid algorithms.

The process of training for artificial neural networks (ANNs) is an important implementation of meta-heuristic algorithms and hybrid versions of these algorithms. ANNs, generally defined as a mathematical model of nerve cells in the human brain and the connection between these cells, is one of the most researched artificial intelligence techniques today (McCulloch & Pitts, 1943; Anderson, 1995). ANNs are widely used in many fields such as classification, prediction, and optimization. One of the different neural networks (NNs) structures suggested in the literature is the feed-forward multi-layer perceptron (MLP). MLP has multi-layer neuron architecture. At this structure, the first layer is called the input layer, the last layer is the output layer, and the other layers between the first layer and the last layer are named as hidden layers. Neurons in all layers from the first to the last layer are connected to neurons in the following layer. The information flow is forward and there is no feedback. One of the reasons why this technique is preferred by researchers is its suitability for solving nonlinear and complex problems (Faris et al., 2016).

One of the situations that significantly affect performance in MLP and other ANNs structures is the training process. In this process, the output value generated at the end of each iteration is compared with the target value. Depending on the error, the biases and weights are updated. These operations to minimize the error are known as training the network. Supervised, unsupervised and reinforced learning methods are used during the education process. In MLP training, gradient-based and stochastic methods are used for the supervised learning method. Gradient-based conventional methods for instance the back propagation algorithm (Rumelhart et al., 1986) have disadvantages such as local minima, slow convergence, and dependence on initial values (Gori & Tesi, 1992; Gupta & Sexton, 1999). Stochastic methods, such as meta-heuristic algorithms, are other methods recommended as an alternative in the training process to overcome the disadvantages of conventional methods. GA (Seiffert, 2001), ABC (Karaboga et al., 2007), MOA (Mirjalili & Sadiq, 2011), AFS (Hasan et al., 2011), PSOGSA (Mirjalili et al., 2012), FA (Brajevic & Tuba, 2013; Alweshah, 2014), GWO (Mirjalili et al., 2014a), SSO (Mirjalili et al., 2015), BBO (Mirjalili et al., 2014b), MFO (Yamany et al., 2015), CSSO (Abedinia & Amjady, 2015), MBA (Tuba et al., 2015), MVO (Faris et al., 2016), MPSO (Kolay et al., 2016), SOS (Wu et al., 2016), GSO (Alboaneen et al., 2017), WOA (Aljarah et al., 2018), IMBO (Faris et al., 2018), SSA (Abusnaina et al., 2018; Bairathi & Gopalani, 2019) algorithms can be given as examples of meta-heuristic and hybrid algorithms used in the training of ANNs.

In this study, MVO which is a population-based algorithm and SA which is a single-solution based algorithm used for the training of the MLP, in an integrative hybrid structure. Our aim is to increase the success of MLP training with this hybrid model. Lately, a new hybrid model has been suggested for function optimization, in which MVO and SA algorithms are used together (Yılmaz et al., 2022). However, regarding to the no free lunch (NFL) theorem, a single optimization technique is not sufficient to solve all optimization problems (Wolpert & Macready, 1997). MLP training also means a different optimization problem for each different data set. For this reason, the proposed algorithm for function optimization has been modified and adapted for MLP training. In the literature, a hybrid model that uses MVO and SA algorithms for the training of ANNs not found and this model will be implemented for the first time.

The remaining of this article is established as follows: in the second section, a general description of MLP - NN is given. In the third section elaborate information regarding the suggested hybrid algorithm (MVOSANN) and its components is explained. In the fourth section, it is shown how to use the MVOSANN algorithm to train MLP. Comparing experiment results with other algorithms are given in the fifth section. In this section, also, the performance of the suggested algorithm is interpreted. Finally, in the sixth section, the results of the implementation were evaluated, and recommendations were made for future studies.

## 2. Feed-forward multilayer perceptron

ANNs are mathematical models whose names and structures are taken inspiration by the human brain. They mimic the nerve cells in the human brain and the connections between these cells. ANNs can be classified into different types to be used for various targets. Feed-forward MLP is one of the commonly used types of ANNs. The input layer, the hidden layers, and the output layer are the parts that constitute the MLP. Neurons are interconnected in one direction and forward. A simple example of a feed-forward MLP architecture with a single hidden layer is shown in Fig.1. Each neuron in one layer connects to all neurons in other layers. Each neuron has a different weight and bias values. If the output of any neuron is above the specified threshold, that node is activated, and data is sent to the next layer. Otherwise, no data is sent to the next layer. The calculation of the weighted sum of the input values is shown below:

$$v_j = \sum_{i=1}^{n} w_{ij} I_i + \beta_j \tag{1}$$

where, $v_j$ is the net input value of neuron $j$ in the hidden layer, $w_{ij}$ is the connection weight connecting $I_i$ to neuron $j$, $I_i$ indicates the neuron $i$ in the input layer, $\beta_j$ is a bias weight of neuron $j$, and $n$ is the total number of neuron inputs.

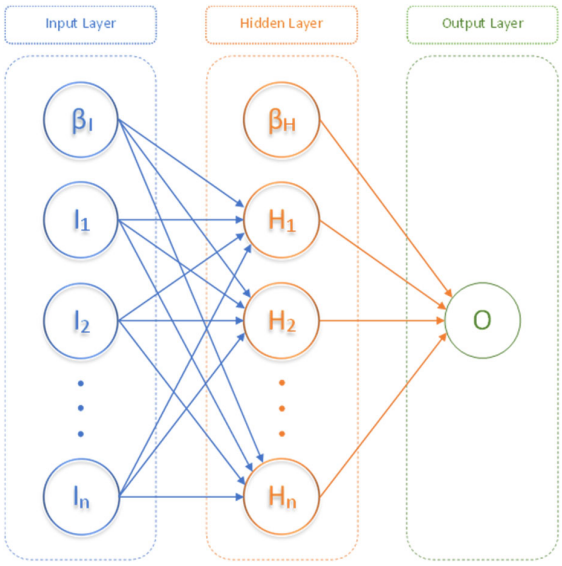


**Fig. 1.** Multilayer perceptron network with single hidden layer

The outputs of neurons are activated with an activation function which is dependent on the value of the summation function. In general, this function is a nonlinear function like hyperbolic tangent sigmoid. The hyperbolic tangent sigmoid function is shown as follows:

$$f(n)=\frac{2}{1+e^{-2n}}-1 \tag{2}$$

In the hidden layer, the output of each neuron is calculated as follows:

$$y_j=f_j\left(\sum_{i=1}^{n} w_{ij}I_i+\beta_j\right) \tag{3}$$

After the NN is created, the input data presented to it is expected to give the desired output. For the NN to achieve this, it needs to update the network parameters, that is, the weights. This stage is called the training process, which is a tough stage that shows the power of the network. The goal here is to minimize the error among the actual values and the calculated output values. This process is performed using a training algorithm. The system generates an output value by using the values generated at the end of the training and the inputs presented to it.

## 3. MVOSANN algorithm

### *3.1. Multi-verse optimizer (MVO)*

In 2016, population-based multiverse optimization, taking inspiration from the concepts of wormholes, black holes, and white holes in the big bang theory and in the multiverse theory, was suggested by Seyedali Mirjalili et al. (Mirjalili et al., 2016). In this algorithm, where wormholes, black holes, and white holes are mathematically modelled for exploration, exploitation, and local search, the fitness function for each universe is referred by an inflation rate. Each universe represents a candidate solution, and each object represents a variable in the candidate solution.

In the algorithm process, sometimes, there is an exchange of objects between universes. Universes with high inflation rate prone to send their objects to universes with low inflation rate, while universes with low inflation rate prone to receive objects from universes with high inflation rate. Optimization is started with the above steps. In each new iteration, the universes are ordered accordingly their inflation rates at the end of the previous iteration. The conceptional model of the suggested algorithm is shown in Fig. 2. The following rules apply in optimization:

• The odds that universes have white holes increase if inflation rates are higher.
• The odds that universes have black holes increase if inflation rates are lower.
• Universes with high inflation rate prone to dispatch objects via white holes.
• Universes with low inflation rate prone to get more objects from black holes.
• Wormholes can cause random object movement from the best universe to other universes.

The mathematical model of MVO is as follows:

$$UNV = \begin{bmatrix} o_1^1 & o_1^2 & \cdots & o_1^k \\ o_2^1 & o_2^2 & \cdots & o_2^k \\ \vdots & \vdots & \ddots & \vdots \\ o_n^1 & o_n^2 & \cdots & o_n^k \end{bmatrix} \tag{4}$$

where, $n$ is the number of universes (candidate solutions), $k$ is the number of parameters (variables).

$$o_i^j = \begin{cases} o_h^j & rn1 < NRM(UNV_i) \\ o_i^j & rn1 \geq NRM(UNV_i) \end{cases} \tag{5}$$

where, $o_i^j$ defines the $j^{th}$ parameter of $i^{th}$ universe, $o_h^j$ defines the $j^{th}$ parameter of $h^{th}$ universe selected by a roulette wheel selection, $rn1$ is a number generated randomly in the range [0, 1], $NRM(UNV_i)$ is normalized inflation rate of the $i^{th}$ universe, $UNV_i$ defines the $i^{th}$ universe.

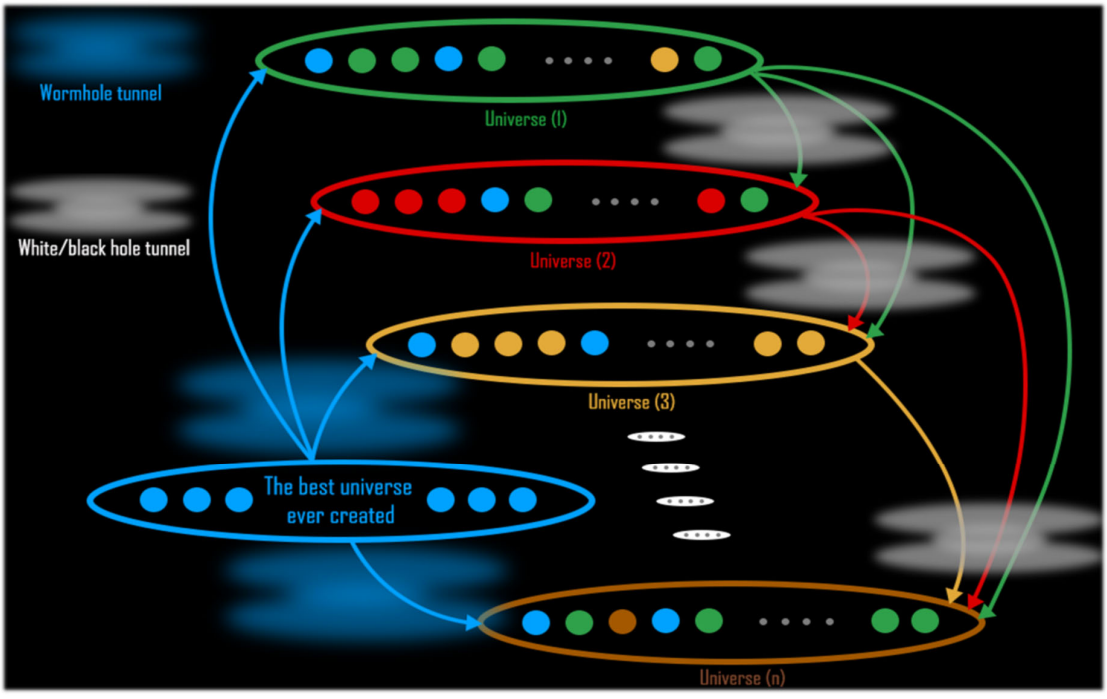


**Fig. 2.** Conceptional Model of MVO Algorithm.

Wormholes, sometimes randomly replace objects of universes with objects of the best universe to preserve and exploit the diversity of universes. The mathematical model of this mechanism is as follows.

$$o_i^j = \begin{cases} \begin{cases} O_j + Travelling\_DR \times \left( \left( uppb_j - lowb_j \right) \times rn4 + lowb_j \right) & rn3 < 0.5 \\ O_j - Travelling\_DR \times \left( \left( uppb_j - lowb_j \right) \times rn4 + lowb_j \right) & rn3 \geq 0.5 \end{cases} & rn2 < Wormhole\_EP \\ o_i^j & rn2 \geq Wormhole\_EP \end{cases} \quad (6)$$

where, $O_j$ defines the $j^{th}$ parameter of best universe generated up to now, $o_i^j$ defines the $j^{th}$ parameter of $i^{th}$ universe, $uppb_j$ is the upper bound of $j^{th}$ variable, $lowb_j$ is the lower bound of $j^{th}$ variable, *Travelling_DR* and *Wormhole_EP* are the coefficients, and *rn*2, *rn*3, *rn*4 are numbers acquired randomly in the range [0, 1].

MVO has two coefficients: Travelling distance rate (*Travelling_DR*) and Wormhole existence probability (*Wormhole_EP*) (Eq. 6). The mathematical model for both coefficients is as follows:

$$Wormhole\_EP = \min + l \times \left( \frac{\max - \min}{L} \right) \quad (7)$$

where, *l* is the current iteration, *L* is the maximum iteration, *min* is the minimum value equal to 0.2, and *max* is the maximum value equal to 1.

$$Travelling\_DR = 1 - \frac{l^{1/p}}{L^{1/p}} \quad (8)$$

where, *p* defines the accuracy of exploitation on iterations. Earlier and more accurate local search - exploitation is possible with a higher *p* value. Algorithm 1 shows the pseudocode of MVO.

**Algorithm 1.** MVO

```
Create random universes (UNV)
Initialize BestUNV,Wormhole_EP,Travelling_DR
SUNV = Sorted universes
NRM = Normalize inflation rates (fitness) of the universes
while Time < Max_time
    Update Travelling_DR, Wormhole_EP
    Evaluate the fitness of all universes
    for each universe indexed by i
        Black_Hole_Index = i;
        for each object indexed by j
            rn1 = random([0, 1]);
            if rn1 < NRM(UNV_i)
                White_Hole_Index = RouletteWheelSelection(-NRM);
                UNV(Black_Hole_Index, j) = SUNV(White_Hole_Index, j);
            end if
            rn2 = random([0, 1]);
            if rn2 < Wormhole_EP
                rn3 = random([0, 1]);
                rn4 = random([0, 1]);
                if rn3 < 0.5
                    UNV(i,j)=BestUNV(j)+Travelling_DR ×((uppb(j)-lowb(j))×rn4+lowb(j));
                else
                    UNV(i,j)= BestUNV(j)-Travelling_DR ×((uppb(j)-lowb(j))×rn4+lowb(j));
                end if
            end if
        end for
    end for
```

**28: end while**

*3.2. Simulated annealing (SA)*

Simulated annealing (SA) (Kirkpatrick et al., 1983), proposed by Kirkpatrick et al. in 1983, is one of the local search algorithms based on hill climbing method used in solving optimization problems. In the algorithm, where the creation of new solutions is based on predetermined rules or performed randomly, the newly created solution is compared with the current solution in each iteration. To find the global best and not get stuck with the local best, the algorithm, can accept new solutions that only enhance the current solution, as well as worse results that provide predetermined criteria. These criteria are decided by the Boltzmann probability. The following equation (Eq. 9) shows this mechanism:

$$Prb = e^{\left(\frac{-(F(X)-F(X_0))}{T_k}\right)} \tag{9}$$

where, *Prb* is defined as the acceptance probability. $F(X_0)$ defines the objective function for current solution, $F(X)$ defines the objective function for candidate solution. SA uses the acceptance probability mechanism to decide whether the candidate solution should be considered as the current solution when $F(X)$ is worse than $F(X_0)$. $T_k$ is the temperature value at time *k, and* in each iteration, value of $T_k$ is calculated as follows:

$$T_{k+1} = T_k \times c \tag{10}$$

where, $c$ refers to the temperature coefficient, $T_k$ is the initial temperature value and $T_{k+1}$ is the temperature at time *k*. Algorithm 2 presents the pseudocode of SA algorithm.

**Algorithm 2.** SA

```
T = Temperature value
T0 = Final temperature value
c = Temperature coefficient
A = First solution
f(A) = Fitness value of the first solution
while T > T0
    A′ = a new solution in the A neighbourhood
    f(A') = Calculate the fitness value of A′
    if f(A') < f(A)
        NSol = A';
        f(NSol) = f(A');
    else
        Δf = f(A') - f(A)
        r = random[0, 1];
        if r > exp(− Δf / T )
            NSol = A′;
            f(NSol) = f(A');
        else
            NSol = A;
            f(NSol) = f(A);
        end if
    end if
    A = NSol;
    T = T × c
end while
```

*3.3. MVOSANN algorithm*

MVO is a population-based meta-heuristic algorithm that produces very accomplished results in the face of many problems. Despite

the many advantages of the MVO algorithm, it also has disadvantages such as low accuracy, local minimum, and slow convergence (Jia et al., 2019; Song et al., 2020). In this part of the study, to eliminate these disadvantages improvements in the structure of the original MVO algorithm are suggested.

According to the uncertainty principle in physics, the momentum and position of a particle cannot be determined precisely at the same time, there is always an error, uncertainty and blurring possible (Heisenberg, 1985). According to the many-worlds interpretation (MWI), which is based on the uncertainty principle, there is a single and universal wave function for the entire universe as reality itself. This universal wave function, as the wave function of everything, includes all possibilities in the known world and an infinite number of parallel worlds in which every possibility exists (Everett et al., 1973).

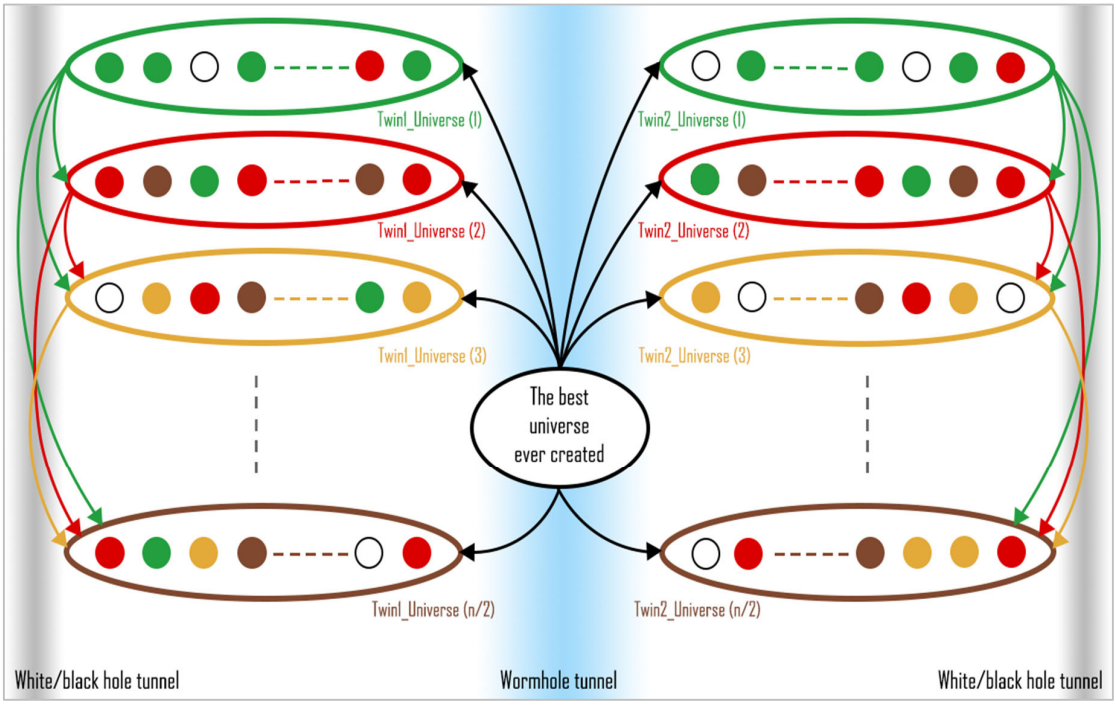


**Fig. 3.** Conceptional Model of MVOSANN Algorithm

Inspired by the multiverse theory and the big bang theory, the MVO algorithm has been modified based on the uncertainty principle in physics and the MWI. To get the global best without being stuck with local minimums search agents and their twins perform the search by acting according to the general rules of the MVO and the acceptance mechanism of the SA. In this method, we used the SA algorithm in the integrative hybrid model structure, where exploration and exploitation can be increased. Fig. 3 shows the conceptional model of the suggested MVOSANN algorithm.

In the suggested model, unlike the MVO algorithm, two initial populations with the same universes are created (Eq. 11). At each iteration, search agents in this twin population act according to the general rules of the MVO algorithm, to which the SA algorithm's acceptance probability feature is added. Each object represents a variable in the candidate solution and each universe in the populations represents a candidate solution.

$$TP = \left[ \begin{bmatrix} o_1^1 & o_1^2 & \cdots & o_1^k \\ o_2^1 & o_2^2 & \cdots & o_2^k \\ \vdots & \vdots & \ddots & \vdots \\ o_{n/2}^1 & o_{n/2}^2 & \cdots & o_{n/2}^k \end{bmatrix} \begin{bmatrix} o_1^1 & o_1^2 & \cdots & o_1^k \\ o_2^1 & o_2^2 & \cdots & o_2^k \\ \vdots & \vdots & \ddots & \vdots \\ o_{n/2}^1 & o_{n/2}^2 & \cdots & o_{n/2}^k \end{bmatrix} \right] \tag{11}$$

where, $k$ is the number of parameters (variables), $n$ is the number of universes (candidate solutions).

In the MVO algorithm, updating the variables in a candidate solution is accomplished with a easy location update (Jia et al., 2019). The values of the fitness function of the solutions generated using the acceptance probability of the SA algorithm will not be prone to decrease constantly, and in some circumstances, also, the solutions with high fitness function values will be accepted and the searching for the global best will be effected. Equation (Eq. 5), which was used in the exploration stage of the original MVO algorithm, is suggested as follows to increase the exploration and exploitation of the universe around the best solution.

**Algorithm 3.** MVOSANN

```
Create twin populations randomly (TP)
Initialize BestUNV,Wormhole_EP,Travelling_DR, T, c
SUNV= Sorted universes
NRM = Normalize inflation rates (fitness) of the universes
while Time < Max_time
    Update Travelling_DR, Wormhole_EP
    Evaluate the fitness of universes of all populations
    Sort all universes of all populations
    TP[0] = SUNV[0:(N/2)] (Assign the first half of the sorted universes to the first of the twin populations)
    TP[1] = TP[0] (Equalize twin population)
    for each population indexed by k
        for each universe indexed by i
            Black_Hole_Index = i;
            for each object indexed by j
                rn1 = random ([0, 1]);
                if rn1 < NRM(SUNV [i,:])
                    White_Hole_Index = RouletteWheelSelection(-NRM);
                    TP[k][Black_Hole_Index][ j] = SUNV[White_Hole_Index,j];
                else
                    Calculate the fitness's of SUNV[Black_Hole_Index,:] and
                                                TP[k][Black_Hole_Index][:] (f( ))
                    Δf = f(SUNV[Black_Hole_Index,:]) - f(TP[k][Black_Hole_Index][:])
                    rn2 = random ([0, 1]);
                    if rn2 < exp (-Δf / T)
                        White_Hole_Index = RouletteWheelSelection(-NRM);
                        TP[k][Black_Hole_Index] [ j] = SUNV[White_Hole_Index,j];
                    end if
                end if
                rn3 = random ([0, 1]);
                if rn3 < Wormhole_EP
                    rn4 = random ([0, 1]);
                    rn5 = random ([0, 1]);
                    if rn4 < 0.5
                        TP[k][i][j]=BestUNV(j)+Travelling_DR ×((uppb(j)-lowb(j)×rn5+lowb(j));
                    else
                        TP[k][i][j]=BestUNV(j)-Travelling_DR×((uppb(j)-lowb(j))×rn5+lowb(j));
                    end if
                end if
            end for
        end for
    end for
```

**41:** $T = T \times c$

**42: end while**

$$o_i^j = \begin{cases} o_h^j & rn1 < NRM\left(SUNV\left[i,:\right]\right) \\ o_h^j & rn2 < Prb \quad rn1 \geq NRM\left(SUNV\left[i,:\right]\right) \\ o_i^j & rn2 \geq Prb \end{cases} \tag{12}$$

where, $o_i^j$ defines the $j^{th}$ parameter of $i^{th}$ universe, $o_h^j$ defines the $j^{th}$ parameter of $h^{th}$ universe chosen by a roulette wheel selection, $rn1$ and $rn2$ are numbers procured randomly in the range [0, 1], $NRM\left(SUNV\left[i,:\right]\right)$ is normalized inflation rate of the $i^{th}$ sorted universe, $SUNV\left[i,:\right]$ describes the $i^{th}$ sorted universe. $Prb$ is defined as the acceptance probability (Eq. 9).

At the end of the iteration, the universes of the twin population clusters, which are no longer alike, are sorted collectively according to their inflation rates at each new iteration. The first universe in this ranking is the best universe ever found. After this stage, the first half of the collectively sorted universes with the best values is selected. This selected universe set creates the twin population set of the new iteration. The pseudo code of the MVOSANN algorithm is given in Algorithm 3.

This set of twin populations, with universes containing identical objects at the beginning, diverges from similarity at the end of each iteration. As a result of different exploration and exploitation possibilities in the universes of this twin population cluster at the same time, it is more likely to reach the best inflation value. Because, as mentioned in the uncertainty principle and the MWI, the state of an object cannot be determined simultaneously and there are always parallel worlds where other different possibilities take place.

## 4. MVOSANN for training MLP

In this section, it is explained how the MVOSANN algorithm is used in training a single hidden layer MLP network. As mentioned in the introduction, the aim of training the neural network is to find the best weight and bias values. This is the main task of the MVOSANN algorithm in this study. Each universe (search agent) in the algorithm consists of the connection weights among the input layer to the hidden layer ( $Iw_{nm}$ ), the weights among the hidden layer to the output layer ( $Hw_m$ ), and the bias weights ( $\beta_m$ ).

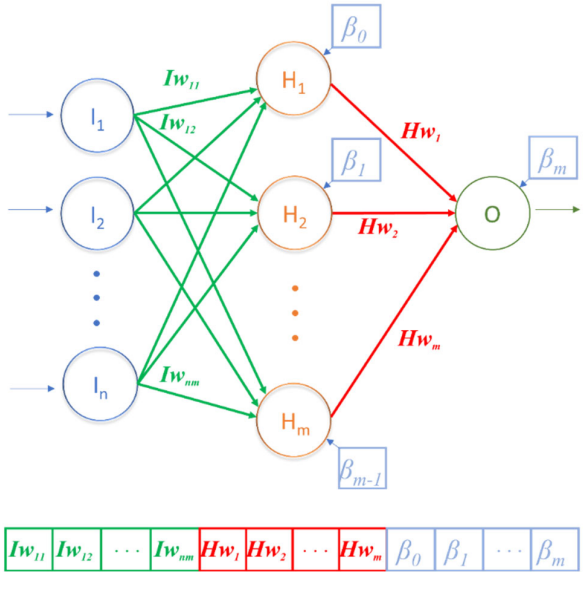


**Fig. 4.** Representation of the MVOSANN universe for the MLP network

The representation of the universes consisting of real numbers in the range of [−1, 1] is given in Fig. 4 in vector structure.

Equation (Eq. 13) shows how the number of objects in each universe is calculated.

$$IndividualLength = (m \times n) + (2 \times m) + 1 \tag{13}$$

where, $m$ is the number of neurons in the hidden layer, $n$ is the number of input features.

In all data sets, the mean square error (MSE) fitness function, which is based on calculating the difference between the real values and the values predicted by MLP, was used to measure the fitness value of the universes produced by MVOSANN. The aim here is to minimize the MSE value until the last iteration. The mathematical representation of MSE is as follows.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 \tag{14}$$

where, $y$ is the actual values, and $\hat{y}$ is the predicted values, $n$ is the number of samples in the training dataset.

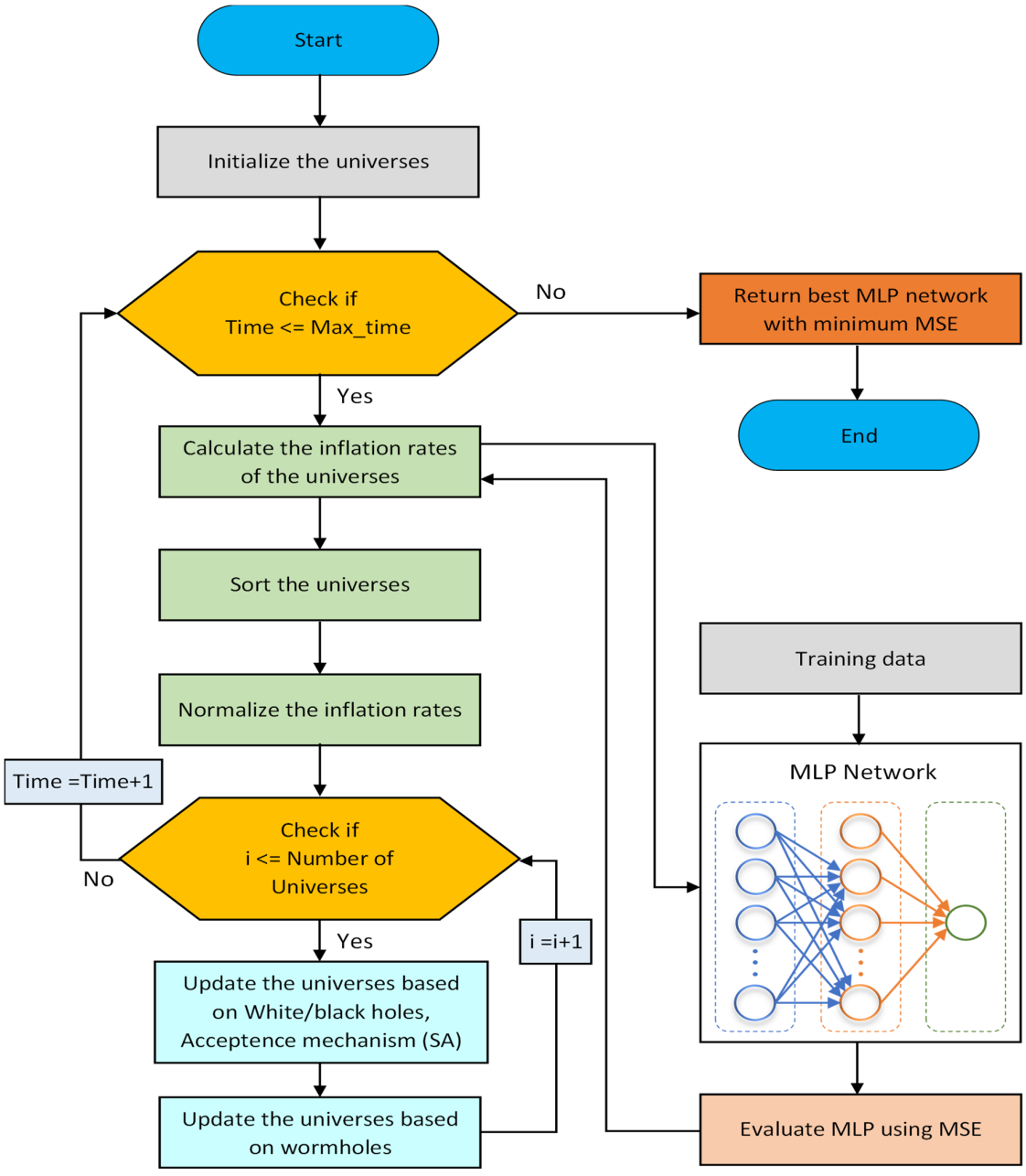


**Fig. 5.** General steps of the MVOSANN for the MLP network training

The MVOSANN algorithm is designed to train MLP networks as depicted in Fig. 5. We can summarize this approach with the following steps:

1. A predefined number of candidate solution sets are randomly generated, where each universe (candidate solution) represents an MLP network.
2. Universes consisting of all weights, including bias weights, are assigned to MLP networks. All generated MLP networks are evaluated using a fitness function (MSE). The aim is to reach the lowest MSE value according to the examples in the dataset used.
3. In this step, the location update of the universes is performed.
4. Steps 2 and 3 repeat up to the maximum number of iterations.

## 5. Experimental results

### *5.1. Datasets and comparison algorithms*

In this study, a comprehensive analysis was carried out to examine the effectiveness of the MVOSANN in training MLP neural networks. MVO (Mirjalili et al., 2016) and 12 algorithms prevalently aforesaided in the literature: Grey Wolf Optimizer (GWO) (Mirjalili et al., 2017), Cuckoo Search (CS) (Yang & Deb, 2009), Harris Hawks Optimization (HHO) (Heidari et al., 2019), Genetic Algorithms (GA) (Holland, 1992), Differential Evolution (DE) (Storn & Price, 1997), Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), Firefly Algorithm (FA) (Yang, 2008, 2009), Salp Swarm Algorithm (SSA) (Mirjalili et al., 2017), Sine Cosine Algorithm (SCA) (Mirjalili, 2016), Whale Optimization Algorithm (WOA) ( Mirjalili & Lewis, 2016), and JAYA Algorithm (JAYA) (Rao, 2016) were used to evaluate the achievement of the MVOSANN algorithm.

MVOSANN and other algorithms have been tested on 12 data sets used in many studies in the literature. These datasets, selected from the University of California, Irvine Machine Learning Repository (UCI), are: Abalone (Dua & Graff, 2019), Balance Scale (Dua & Graff, 2019), Blood (Yeh et al., 2009), Breast Cancer (Wolberg & Mangasarian, 1990; Bennett & Mangasarian, 1992), Diabetes (Dua & Graff, 2019), Acute Inflammations (Diagnosis I, Diagnosis II) (Czerniak & Zarzycki, 2003), Glass Identification (Dua & Graff, 2019), Iris (Dua & Graff, 2019), Liver disorders (Dua & Graff, 2019), Raisin (Çınar et al., 2020), Vertebral (Dua & Graff, 2019). Information about these datasets, such as the number of features, the number of training samples, and the number of test samples, are given in Table 1.

**Table 1**
Classification datasets

| Dataset | Features | Training Samples | Testing Samples |
|---|---|---|---|
| Abalone | 8 | 2756 | 1421 |
| Balance Scale | 4 | 412 | 213 |
| Blood | 4 | 493 | 255 |
| Breast Cancer | 8 | 461 | 238 |
| Diabetes | 8 | 506 | 262 |
| Diagnosis I | 6 | 79 | 41 |
| Diagnosis II | 6 | 79 | 41 |
| Glass | 10 | 141 | 73 |
| Iris | 4 | 99 | 51 |
| Liver | 6 | 79 | 41 |
| Raisin | 8 | 594 | 306 |
| Vertebral | 6 | 204 | 106 |

### *5.2. Experimental setup*

Experiments were conducted using the EvoloPy-NN repository created by Faris in 2016 (Faris, 2016; Faris et al., 2016c). Evolopy-NN is an open-source framework consisting of classic and recent meta-heuristic algorithms coded with Python, with a user-friendly interface. Various studies have been carried out on artificial neural network training using the Evolopy-NN repository (Faris et al., 2016a; Faris et al., 2016b; Aljarah et al., 2018a; Aljarah et al., 2018b). All algorithms have been experimented on a personal computer with 16 GB (RAM), Intel Core i7-10875H 2.30 GHz (CPU), and 64-Bit Windows 11 (operating system).

Table 2 shows the parameter settings of the algorithms used in this study. All datasets are divided into 34 % for testing, and 66 % for training. The number of search agents for the algorithms is set to 50 and the number of iterations to 100. Each algorithm was run individually 30 times to obtain balanced performance results.

**Table 2**
Parameters and values of MVOSANN and other algorithms

| Algorithm | Parameter | Value |
|---|---|---|
| MVOSANN | *Maximum wormhole existence probability* | 1 |
| | *Minimum wormhole existence probability* | 0.2 |
| | *p* | 6 |
| | *T* | 1 |
| | *c* | 0.88 |
| MVO | *Maximum wormhole existence probability* | 1 |
| | *Minimum wormhole existence probability* | 0.2 |

| | | |
|---|---|---|
| | *p* | 6 |
| GA | *Crossover probability* | 0.9 |
| | *Mutation probability* | 0.1 |
| | *Selection mechanism* | Roulette wheel |
| FA | *Alpha* | 0.5 |
| | *Beta* | 0.2 |
| | *Gamma* | 1 |
| SSA | *No custom parameters* | |
| GWO | *No custom parameters* | |
| WOA | *b* | 1 |
| PSO | *Vmax* | 6 |
| | *c1* | 2 |
| | *c2* | 2 |
| | *wMax* | 0.9 |
| | *wMin* | 0.2 |
| CS | $p_a$ | 0.25 |
| HHO | *No custom parameters* | |
| SCA | *a* | 2 |
| JAYA | *No custom parameters* | |
| DE | *Crossover probability* | 0.9 |
| | *Differential weight* | 0.5 |

In this study, the method in which the number of hidden neurons is 2 × N + 1 was chosen; N is the number of features in each dataset. For each dataset, all input features values are normalized in the range [0,1] with Min-max normalization technique (Eq. 15).

$$v' = \frac{v_i - \min_A}{\max_A - \min_A} \tag{15}$$

where, $v'$ is the normalized value of $v$ in the range $\left[\min_A, \max_A\right]$

*5.3. Evaluation metrics*

A confusion matrix is a table (Fig. 6), that helps us evaluate the accuracy of a classification model on a dataset with actual values. On this table is a summary of the results predicted by the model in a classification problem (Stehman, 1997). In the model, evaluation was made by calculating the number of correctly predicted samples belong to the class (true positives), the number of incorrectly predicted samples belong to the class (false positives), the number of correctly predicted samples that do not belong to the class (true negatives), and the number of incorrectly predicted samples that do not belong to the class (false negatives). How the metrics used for binary classification and multiclass classification are calculated using the confusion matrix values, are shown in Table 3. (Sokolova & Lapalme, 2009).

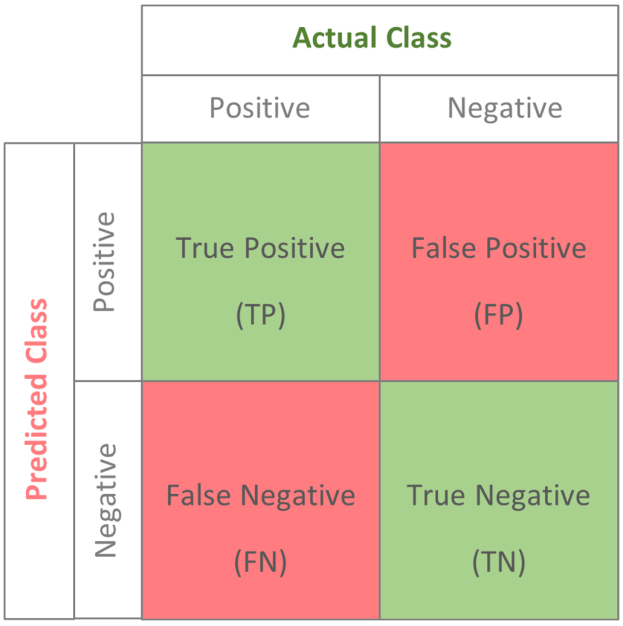


**Fig. 6.** Confusion Matrix

In this study, during the performance evaluation of algorithms, accuracy, recall, precision, error rate and F1 score calculations were made depending on whether each data set was in a binary classification group or a multiclass classification group. In the multi-class classification group, the calculations were made using the micro-average method, considering the class imbalance in the data sets. In addition, the MSE values (Eq. 14) based on the calculation of the difference among the actual values and the predicted values of MLP, and the standard deviations of the accuracy and MSE values were calculated. All algorithms were evaluated for each data set with the results obtained from the calculations. Also, a nonparametric Wilcoxon rank sum test (Wilcoxon, 1992) was applied to the results to analyse the relationship among the algorithms, and the p value was accepted to be less than 0.05 (5E-02) in this test to evaluate the statistically significant difference.

**Table 3**
Measures for binary classification and multi-class classification

| Binary classification | | Multiclass classification | |
|---|---|---|---|
| **Measure** | **Formula** | **Measure** | **Formula** |
| Accuracy | $\frac{TP+TN}{TP+FN+FP+TN}$ | Average_Accuracy | $\frac{\sum_{i=1}^{k}\frac{TP_i+TN_i}{TP_i+FN_i+FP_i+TN_i}}{k}$ |
| Precision | $\frac{TP}{TP+FP}$ | $\text{Precision}_{\mu}$ | $\frac{\sum_{i=1}^{k}TP_i}{\sum_{i=1}^{k}(TP_i+FP_i)}$ |
| Recall | $\frac{TP}{TP+FN}$ | $\text{Recall}_{\mu}$ | $\frac{\sum_{i=1}^{k}TP_i}{\sum_{i=1}^{k}(TP_i+FN_i)}$ |
| F1_score | $2\times\frac{Precision\times Recall}{Precision+Recall}$ | $\text{F1\_score}_{\mu}$ | $2\times\frac{Precision_{\mu}\times Recall_{\mu}}{Precision_{\mu}+Recall_{\mu}}$ |
| Error rate | $\frac{FP+FN}{TP+FN+FP+TN}$ | $\text{Error\_rate}_{\mu}$ | $\frac{\sum_{i=1}^{k}\frac{FP_i+FN_i}{TP_i+FN_i+FP_i+TN_i}}{k}$ |

### *5.3. Results and discussions*

To evaluate the suggested MVOSANN algorithm, accuracy, MSE, recall, precision, error rate and F1 score results of all algorithms obtained from 12 data sets were compared. For each dataset and 30 trials, Table 4 shows the accuracy averages,

standard deviations of the mean accuracies of all algorithms, and best accuracies. According to the results, MVOSANN performs much better than all other algorithms by having the best average accuracy values in all data sets. In addition, MVOSANN has low standard deviation values for all data sets, which shows that the proposed algorithm is robust and stable. Considering the best accuracy values, MVOSANN, 5 achieved the highest accuracy values for the data set, and it can be said that it is superior when compared to other algorithms.

**Table 4**
Results of classification accuracy for all datasets and algorithms

| Datasets | | Algorithms | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MVOSA | MVOGA | FA | SSA | PSO | GWO | WOA | CS | HHO | SCA | JAYA | DE | |
| Abalone | AVE | **0.8230** | 0.821 | 0.8047 | 0.8139 | 0.8100 | 0.8122 | 0.8155 | 0.7877 | 0.8066 | 0.8009 | 0.8013 | 0.8040 | 0.8012 |
| | STD | **0.0031** | 0.004 | 0.0101 | 0.0080 | 0.0077 | 0.0098 | 0.0058 | 0.0150 | 0.0131 | 0.0100 | 0.0128 | 0.0157 | 0.0199 |
| | BEST | 0.8291 | 0.831 | 0.8249 | 0.8270 | 0.8256 | 0.8277 | 0.8242 | 0.8186 | 0.8326 | 0.8136 | **0.8418** | 0.8319 | 0.8291 |
| Balance | AVE | **0.8376** | 0.830 | 0.7746 | 0.8144 | 0.8239 | 0.8282 | 0.8194 | 0.7573 | 0.7850 | 0.7624 | 0.7156 | 0.7654 | 0.7019 |
| | STD | **0.0490** | 0.049 | 0.0787 | 0.0682 | 0.0587 | 0.0664 | 0.0519 | 0.0776 | 0.0919 | 0.0660 | 0.1043 | 0.1273 | 0.0874 |
| | BEST | 0.9296 | 0.910 | 0.9296 | 0.9155 | 0.9343 | 0.9343 | 0.9108 | 0.9249 | 0.9014 | 0.8873 | 0.8638 | **0.9437** | 0.8498 |
| Blood | AVE | **0.7906** | 0.788 | 0.7837 | 0.7829 | 0.7808 | 0.7843 | 0.7822 | 0.7838 | 0.7829 | 0.7838 | 0.7826 | 0.7835 | 0.7822 |
| | STD | **0.0038** | 0.005 | 0.0051 | 0.0054 | **0.0038** | 0.0074 | 0.0048 | 0.0067 | 0.0065 | 0.0070 | 0.0074 | 0.0072 | 0.0083 |
| | BEST | 0.7961 | 0.796 | 0.7922 | 0.7922 | 0.7882 | **0.8000** | 0.7922 | 0.7922 | 0.7961 | 0.7922 | 0.7922 | 0.7961 | 0.7961 |
| Breast Cancer | AVE | **0.9777** | 0.975 | 0.9678 | 0.9711 | 0.9721 | 0.9655 | 0.9746 | 0.9682 | 0.9664 | 0.9696 | 0.9602 | 0.9578 | 0.9571 |
| | STD | **0.0044** | 0.005 | 0.0085 | 0.0063 | 0.0063 | 0.0072 | 0.0055 | 0.0089 | 0.0085 | 0.0049 | 0.0107 | 0.0103 | 0.0104 |
| | BEST | **0.9874** | 0.983 | 0.9832 | 0.9832 | 0.9832 | 0.9748 | 0.9832 | 0.9790 | 0.9790 | 0.9790 | 0.9832 | 0.9748 | 0.9748 |
| Diabetes | AVE | **0.7579** | 0.753 | 0.7246 | 0.7490 | 0.7480 | 0.7439 | 0.7497 | 0.6957 | 0.7247 | 0.7118 | 0.7078 | 0.7230 | 0.7289 |
| | STD | 0.0057 | **0.005** | 0.0236 | 0.0121 | 0.0075 | 0.0136 | 0.0063 | 0.0329 | 0.0229 | 0.0254 | 0.0267 | 0.0231 | 0.0235 |
| | BEST | 0.7710 | 0.763 | 0.7748 | 0.7748 | 0.7672 | 0.7748 | 0.7672 | **0.7824** | 0.7672 | 0.7481 | 0.7595 | 0.7672 | 0.7710 |
| Diagnosis I | AVE | **0.9317** | 0.915 | 0.8431 | 0.8455 | 0.8683 | 0.8537 | 0.9049 | 0.8106 | 0.9065 | 0.8350 | 0.8293 | 0.8756 | 0.8236 |
| | STD | 0.1057 | 0.109 | 0.1170 | 0.1127 | 0.1177 | 0.1133 | 0.1091 | 0.1296 | 0.1150 | 0.1143 | 0.1137 | 0.1321 | **0.1050** |
| | BEST | **1.0000** | **1.000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** |
| Diagnosis II | AVE | **0.8333** | 0.825 | 0.7894 | 0.8138 | 0.8163 | 0.8024 | 0.8146 | 0.8000 | 0.8057 | 0.8073 | 0.7829 | 0.7911 | 0.7911 |
| | STD | **0.0129** | 0.019 | 0.0606 | 0.0244 | 0.0210 | 0.0407 | 0.0165 | 0.0422 | 0.0359 | 0.0074 | 0.0688 | 0.0596 | 0.0662 |
| | BEST | **0.8537** | **0.853** | 0.8293 | **0.8537** | **0.8537** | **0.8537** | **0.8537** | 0.8293 | 0.8293 | 0.8293 | 0.8293 | 0.8293 | **0.8537** |
| Glass | AVE | **0.8810** | 0.868 | 0.8625 | 0.8606 | 0.8722 | 0.8750 | 0.8653 | 0.8769 | **0.8810** | 0.8755 | 0.8736 | 0.8514 | 0.8597 |
| | STD | **0.0135** | 0.025 | 0.0297 | 0.0246 | 0.0191 | 0.0287 | 0.0147 | 0.0455 | 0.0364 | 0.0220 | 0.0395 | 0.0331 | 0.0615 |
| | BEST | 0.9028 | 0.930 | 0.9306 | 0.9167 | 0.9167 | 0.9167 | 0.9028 | 0.9306 | **0.9583** | 0.9167 | 0.9444 | 0.9306 | **0.9583** |
| Iris | AVE | **0.7607** | 0.741 | 0.7033 | 0.7467 | 0.7273 | 0.7380 | 0.7267 | 0.7020 | 0.7147 | 0.7060 | 0.7067 | 0.7153 | 0.7160 |
| | STD | 0.0299 | 0.024 | 0.0283 | 0.0325 | **0.0170** | 0.0491 | 0.0184 | 0.0280 | 0.0283 | 0.0253 | 0.0275 | 0.0460 | 0.0373 |
| | BEST | 0.8200 | 0.780 | 0.7400 | 0.8400 | 0.7600 | **0.9200** | 0.7600 | 0.7400 | 0.7800 | 0.7400 | 0.7400 | 0.8400 | 0.8200 |
| Liver | AVE | **0.7686** | 0.762 | 0.6952 | 0.7395 | 0.7359 | 0.7455 | 0.7517 | 0.6322 | 0.6870 | 0.6630 | 0.6658 | 0.6661 | 0.6661 |
| | STD | 0.0132 | **0.012** | 0.0361 | 0.0259 | 0.0231 | 0.0234 | 0.0180 | 0.0460 | 0.0307 | 0.0463 | 0.0458 | 0.0375 | 0.0342 |
| | BEST | **0.7966** | **0.796** | 0.7712 | 0.7797 | 0.7797 | 0.7797 | 0.7797 | 0.7119 | 0.7627 | 0.7627 | 0.7542 | 0.7458 | 0.7203 |
| Raisin | AVE | **0.8596** | 0.859 | 0.8477 | 0.8527 | 0.8529 | 0.8507 | 0.8558 | 0.8354 | 0.8450 | 0.8402 | 0.8404 | 0.8385 | 0.8369 |
| | STD | **0.0041** | **0.004** | 0.0074 | 0.0061 | 0.0051 | 0.0131 | 0.0053 | 0.0145 | 0.0123 | 0.0147 | 0.0152 | 0.0148 | 0.0186 |
| | BEST | **0.8693** | **0.869** | 0.8595 | 0.8627 | 0.8627 | 0.8660 | 0.8627 | 0.8529 | 0.8627 | 0.8660 | 0.8562 | 0.8627 | 0.8627 |
| Vertebral | AVE | **0.8777** | 0.873 | 0.8189 | 0.8509 | 0.8604 | 0.8440 | 0.8701 | 0.7833 | 0.8220 | 0.7947 | 0.7906 | 0.8057 | 0.8088 |
| | STD | 0.0107 | **0.007** | 0.0380 | 0.0368 | 0.0242 | 0.0398 | 0.0178 | 0.0480 | 0.0331 | 0.0362 | 0.0390 | 0.0370 | 0.0327 |
| | BEST | 0.8962 | 0.886 | 0.8962 | **0.9151** | 0.8962 | 0.8962 | 0.8962 | 0.8585 | 0.8774 | 0.8679 | 0.8491 | 0.8774 | 0.8774 |

In 30 trials, for all datasets and all algorithms, the mean of MSE values, the standard deviation of the mean MSE values, and the lowest MSE values are shown in Table 5. As can be seen from the results, MVOSANN has the best average MSE for all datasets. When the standard deviation values are examined, it is also supported that MVOSANN is a robust and stable algorithm. In addition, MVOSANN achieved the lowest MSE value for the 9 dataset and is in the front row compared to other algorithms.

**Table 5**
Results of MSE for all datasets and algorithms

| Datasets | | Algorithms | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MVOSA | MVO | GA | FA | SSA | PSO | GWO | WOA | CS | HHO | SCA | JAYA | DE |
| Abalone | AVE | **0.126** | 0.1275 | 0.1332 | 0.1299 | 0.1311 | 0.1311 | 0.1290 | 0.1379 | 0.1329 | 0.1340 | 0.1351 | 0.1345 | 0.1352 |
| | STD | **0.000** | **0.0008** | 0.0017 | 0.0022 | 0.0015 | 0.0025 | 0.0012 | 0.0038 | 0.0010 | 0.0022 | 0.0017 | 0.0022 | 0.0017 |
| | BES | **0.125** | 0.1259 | 0.1287 | 0.1259 | 0.1279 | 0.1268 | 0.1272 | 0.1299 | 0.1312 | 0.1304 | 0.1316 | 0.1296 | 0.1319 |
| Balance | AVE | **0.043** | 0.0461 | 0.0599 | 0.0532 | 0.0531 | 0.0491 | 0.0521 | 0.0616 | 0.0569 | 0.0610 | 0.0632 | 0.0591 | 0.0610 |
| | STD | **0.002** | **0.0021** | 0.0032 | 0.0045 | 0.0034 | 0.0034 | 0.0029 | 0.0044 | 0.0023 | 0.0033 | 0.0035 | 0.0041 | 0.0032 |
| | BES | **0.040** | 0.0425 | 0.0519 | 0.0447 | 0.0465 | 0.0424 | 0.0483 | 0.0546 | 0.0516 | 0.0532 | 0.0574 | 0.0494 | 0.0560 |
| Blood | AVE | **0.154** | 0.1548 | 0.1581 | 0.1571 | 0.1572 | 0.1558 | 0.1569 | 0.1602 | 0.1579 | 0.1595 | 0.1598 | 0.1588 | 0.1599 |
| | STD | 0.000 | 0.0009 | 0.0010 | 0.0016 | 0.0011 | 0.0012 | 0.0010 | 0.0038 | **0.0006** | 0.0022 | 0.0012 | 0.0015 | 0.0016 |
| | BES | **0.152** | 0.1534 | 0.1560 | 0.1544 | 0.1547 | 0.1535 | 0.1547 | 0.1563 | 0.1559 | 0.1563 | 0.1576 | 0.1548 | 0.1554 |
| Breast Cancer | AVE | **0.032** | 0.0339 | 0.0425 | 0.0389 | 0.0386 | 0.0420 | 0.0359 | 0.0478 | 0.0445 | 0.0413 | 0.0510 | 0.0520 | 0.0543 |
| | STD | **0.001** | 0.0015 | 0.0025 | 0.0028 | 0.0017 | 0.0033 | 0.0016 | 0.0045 | 0.0017 | 0.0013 | 0.0049 | 0.0049 | 0.0047 |
| | BES | **0.031** | 0.0320 | 0.0362 | 0.0349 | 0.0350 | 0.0342 | 0.0316 | 0.0392 | 0.0405 | 0.0369 | 0.0417 | 0.0430 | 0.0414 |
| Diabetes | AVE | **0.150** | 0.1517 | 0.1675 | 0.1561 | 0.1557 | 0.1570 | 0.1539 | 0.1842 | 0.1685 | 0.1716 | 0.1755 | 0.1705 | 0.1721 |
| | STD | 0.001 | **0.0009** | 0.0061 | 0.0027 | 0.0018 | 0.0027 | 0.0010 | 0.0143 | 0.0035 | 0.0069 | 0.0051 | 0.0043 | 0.0045 |
| | BES | **0.148** | 0.1496 | 0.1571 | 0.1519 | 0.1525 | 0.1517 | 0.1517 | 0.1582 | 0.1627 | 0.1583 | 0.1685 | 0.1623 | 0.1629 |
| Diagnosis I | AVE | **0.090** | 0.0913 | 0.0963 | 0.0931 | 0.0936 | 0.0921 | 0.0928 | 0.1049 | 0.0959 | 0.1001 | 0.1021 | 0.0976 | 0.1003 |
| | STD | **0.000** | 0.0008 | 0.0019 | 0.0015 | 0.0012 | 0.0029 | 0.0009 | 0.0147 | 0.0015 | 0.0077 | 0.0038 | 0.0028 | 0.0023 |
| | BES | 0.090 | **0.0898** | 0.0936 | 0.0902 | 0.0912 | 0.0899 | 0.0914 | 0.0941 | 0.0927 | 0.0933 | 0.0951 | 0.0934 | 0.0967 |
| Diagnosis II | AVE | **0.112** | 0.1177 | 0.1459 | 0.1292 | 0.1319 | 0.1169 | 0.1351 | 0.1492 | 0.1409 | 0.1482 | 0.1515 | 0.1444 | 0.1509 |
| | STD | **0.002** | 0.0046 | 0.0049 | 0.0083 | 0.0071 | 0.0070 | 0.0054 | 0.0089 | 0.0041 | 0.0063 | 0.0066 | 0.0083 | 0.0052 |
| | BES | **0.107** | 0.1096 | 0.1354 | 0.1123 | 0.1198 | **0.1078** | 0.1247 | 0.1311 | 0.1314 | 0.1347 | 0.1340 | 0.1265 | 0.1394 |
| Glass | AVE | **0.028** | 0.0296 | 0.0397 | 0.0361 | 0.0355 | 0.0380 | 0.0325 | 0.0532 | 0.0419 | 0.0408 | 0.0490 | 0.0481 | 0.0534 |
| | STD | 0.001 | 0.0015 | 0.0037 | 0.0028 | 0.0022 | 0.0037 | **0.0012** | 0.0126 | 0.0017 | 0.0046 | 0.0060 | 0.0043 | 0.0051 |
| | BES | **0.025** | 0.0266 | 0.0343 | 0.0299 | 0.0288 | 0.0321 | 0.0299 | 0.0368 | 0.0375 | 0.0353 | 0.0395 | 0.0418 | 0.0422 |
| Iris | AVE | **0.011** | 0.0126 | 0.0189 | 0.0142 | 0.0167 | 0.0147 | 0.0158 | 0.0231 | 0.0186 | 0.0214 | 0.0215 | 0.0192 | 0.0207 |
| | STD | **0.001** | 0.0013 | 0.0015 | 0.0030 | 0.0018 | 0.0023 | 0.0017 | 0.0034 | **0.0010** | 0.0020 | 0.0019 | 0.0019 | 0.0020 |
| | BES | 0.009 | 0.0111 | 0.0161 | **0.0090** | 0.0127 | 0.0110 | 0.0121 | 0.0166 | 0.0161 | 0.0181 | 0.0179 | 0.0158 | 0.0155 |
| Liver | AVE | **0.192** | 0.1960 | 0.2157 | 0.2026 | 0.2064 | 0.1972 | 0.2024 | 0.2257 | 0.2147 | 0.2212 | 0.2206 | 0.2158 | 0.2191 |
| | STD | **0.002** | 0.0029 | 0.0025 | 0.0055 | 0.0038 | 0.0044 | 0.0026 | 0.0064 | 0.0026 | 0.0061 | 0.0039 | 0.0046 | 0.0036 |
| | BES | 0.189 | 0.1905 | 0.2119 | 0.1927 | 0.1989 | **0.1890** | 0.1969 | 0.2137 | 0.2096 | 0.2089 | 0.2124 | 0.2098 | 0.2126 |
| Raisin | AVE | **0.102** | 0.1036 | 0.1152 | 0.1086 | 0.1108 | 0.1050 | 0.1087 | 0.1249 | 0.1139 | 0.1209 | 0.1186 | 0.1160 | 0.1176 |
| | STD | **0.001** | 0.0018 | 0.0031 | 0.0038 | 0.0026 | 0.0036 | 0.0017 | 0.0092 | 0.0019 | 0.0058 | 0.0027 | 0.0034 | 0.0029 |
| | BES | **0.098** | 0.0992 | 0.1100 | 0.1012 | 0.1074 | 0.0990 | 0.1051 | 0.1106 | 0.1092 | 0.1124 | 0.1135 | 0.1100 | 0.1110 |
| Vertebral | AVE | **0.128** | 0.1314 | 0.1475 | 0.1378 | 0.1398 | 0.1348 | 0.1364 | 0.1631 | 0.1478 | 0.1562 | 0.1559 | 0.1518 | 0.1551 |
| | STD | 0.001 | **0.0018** | 0.0046 | 0.0057 | 0.0042 | 0.0038 | 0.0040 | 0.0108 | 0.0037 | 0.0069 | 0.0058 | 0.0060 | 0.0061 |
| | BES | **0.124** | 0.1278 | 0.1385 | 0.1285 | 0.1343 | 0.1265 | 0.1306 | 0.1490 | 0.1345 | 0.1424 | 0.1448 | 0.1412 | 0.1429 |

Table 6 contains the results obtained using the Recall, Error rate, F1 score, and Precision evaluation measures of all algorithms for all datasets. According to the results, in the training process, MVOSANN has the best values in 7 data sets (Balance, Blood, Diabetes, Diagnosis I, Glass, Iris, Liver) for precision measurement, 9 data sets (Abalone, Balance, Breast Cancer, Diabetes, Diagnosis I, Glass, Iris, Liver, Raisin) for recall measurement, 11 data sets (Abalone, Balance, Blood, Diabetes, Diagnosis I, Diagnosis II, Glass, Iris, Liver, Raisin, Vertebral) for F1 score measurement, and 11 data sets (Abalone, Balance, Blood, Diabetes, Diagnosis I, Diagnosis II, Glass, Iris, Liver, Raisin, Vertebral) for error rate measurement. In the experimenting stage, MVOSANN has the best values in 7 data sets (Blood, Diabetes, Diagnosis I, Glass, Iris, Liver, Vertebral) for precision measurement, 9 data sets (Abalone, Breast Cancer, Diabetes, Diagnosis I, Diagnosis II, Glass, Iris, Liver, Raisin) for recall measurement, 12 data sets (Abalone, Balance, Blood, Breast Cancer, Diabetes, Diagnosis I, Diagnosis II, Glass, Iris, Liver, Raisin, Vertebral) for F1 score measurement, and 12 data sets (Abalone, Balance, Blood, Breast Cancer, Diabetes, Diagnosis I, Diagnosis II, Glass, Iris, Liver, Raisin, Vertebral) for error rate measurement. The outcomes again show the supremacy of MVOSANN over other algorithms according to these four evaluation measures.

**Table 6**
Results of precision, recall, F1 score, error rate for all datasets (continued overleaves)

| Datasets | Algorithms | Train | | | | Test | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 Score | Error | Precision | Recall | F1 Score | Error |
| Abalone | MVOSANN | 0.9027 | **0.7815** | **0.8377** | **0.2055** | 0.8857 | **0.8488** | **0.8668** | **0.1770** |
| | MVO | 0.9018 | 0.7794 | 0.8361 | 0.2074 | 0.8844 | 0.8469 | 0.8652 | 0.1790 |
| | GA | **0.9055** | 0.7474 | 0.8184 | 0.2247 | 0.8878 | 0.8159 | 0.8499 | 0.1953 |
| | FA | 0.9003 | 0.7719 | 0.8310 | 0.2130 | 0.8804 | 0.8400 | 0.8597 | 0.1861 |
| | SSA | 0.9020 | 0.7650 | 0.8279 | 0.2159 | 0.8804 | 0.8332 | 0.8561 | 0.1900 |
| | PSO | 0.8933 | 0.7745 | 0.8288 | 0.2167 | 0.8771 | 0.8425 | 0.8587 | 0.1878 |
| | GWO | 0.9006 | 0.7749 | 0.8330 | 0.2108 | 0.8795 | 0.8437 | 0.8612 | 0.1845 |
| | WOA | 0.8961 | 0.7220 | 0.7989 | 0.2460 | 0.8794 | 0.7974 | 0.8358 | 0.2123 |
| | CS | 0.9037 | 0.7488 | 0.8183 | 0.2251 | **0.8883** | 0.8187 | 0.8514 | 0.1934 |
| | HHO | 0.9019 | 0.7459 | 0.8163 | 0.2276 | 0.8821 | 0.8160 | 0.8475 | 0.1991 |
| | SCA | 0.9017 | 0.7418 | 0.8132 | 0.2306 | 0.8846 | 0.8143 | 0.8473 | 0.1987 |
| | JAYA | 0.8932 | 0.7593 | 0.8193 | 0.2261 | 0.8776 | 0.8286 | 0.8510 | 0.1960 |
| | DE | 0.8942 | 0.7545 | 0.8161 | 0.2288 | 0.8787 | 0.8230 | 0.8480 | 0.1988 |
| Balance | MVOSANN | **1.0000** | **0.5420** | **0.7023** | **0.2468** | 0.9715 | 0.7200 | **0.8246** | **0.1624** |
| | MVO | **1.0000** | 0.5132 | 0.6771 | 0.2623 | 0.9701 | 0.7096 | 0.8149 | 0.1695 |
| | GA | **1.0000** | 0.3462 | 0.5098 | 0.3523 | 0.9661 | 0.6107 | 0.7320 | 0.2254 |
| | FA | **1.0000** | 0.4089 | 0.5770 | 0.3185 | 0.9786 | 0.6728 | 0.7880 | 0.1856 |
| | SSA | **1.0000** | 0.3980 | 0.5676 | 0.3244 | 0.9807 | 0.6896 | 0.8031 | 0.1761 |
| | PSO | 0.9992 | 0.4596 | 0.6263 | 0.2914 | 0.9490 | **0.7249** | 0.8128 | 0.1718 |
| | GWO | **1.0000** | 0.4191 | 0.5891 | 0.3130 | 0.9777 | 0.6814 | 0.7987 | 0.1806 |
| | WOA | 0.9985 | 0.3110 | 0.4716 | 0.3715 | 0.9825 | 0.5641 | 0.7013 | 0.2427 |
| | CS | 0.9989 | 0.3583 | 0.5211 | 0.3460 | 0.9645 | 0.6304 | 0.7435 | 0.2150 |
| | HHO | **1.0000** | 0.3096 | 0.4710 | 0.3720 | **0.9880** | 0.5693 | 0.7122 | 0.2376 |
| | SCA | 0.9956 | 0.3170 | 0.4723 | 0.3689 | 0.9746 | 0.4951 | 0.6226 | 0.2844 |
| | JAYA | 0.9975 | 0.3785 | 0.5408 | 0.3354 | 0.9505 | 0.6070 | 0.7222 | 0.2346 |
| | DE | 0.9968 | 0.3107 | 0.4651 | 0.3721 | 0.9532 | 0.4835 | 0.6126 | 0.2981 |
| Blood | MVOSANN | **0.7821** | 0.9799 | **0.8699** | **0.2236** | **0.7977** | 0.9711 | **0.8759** | **0.2094** |
| | MVO | 0.7795 | 0.9819 | 0.8691 | 0.2256 | 0.7945 | 0.9732 | 0.8748 | 0.2119 |
| | GA | 0.7718 | 0.9881 | 0.8667 | 0.2319 | 0.7881 | 0.9789 | 0.8732 | 0.2163 |
| | FA | 0.7731 | 0.9855 | 0.8664 | 0.2317 | 0.7886 | 0.9765 | 0.8725 | 0.2171 |
| | SSA | 0.7715 | 0.9864 | 0.8658 | 0.2331 | 0.7877 | 0.9746 | 0.8712 | 0.2192 |
| | PSO | 0.7793 | 0.9816 | 0.8687 | 0.2263 | 0.7915 | 0.9730 | 0.8729 | 0.2157 |
| | GWO | 0.7737 | 0.9841 | 0.8663 | 0.2316 | 0.7891 | 0.9741 | 0.8719 | 0.2178 |
| | WOA | 0.7703 | 0.9899 | 0.8664 | 0.2329 | 0.7852 | **0.9856** | 0.8740 | 0.2162 |
| | CS | 0.7729 | 0.9853 | 0.8662 | 0.2321 | 0.7886 | 0.9765 | 0.8725 | 0.2171 |
| | HHO | 0.7697 | **0.9910** | 0.8665 | 0.2330 | 0.7853 | 0.9854 | 0.8740 | 0.2162 |
| | SCA | 0.7722 | 0.9879 | 0.8668 | 0.2316 | 0.7865 | 0.9808 | 0.8728 | 0.2174 |
| | JAYA | 0.7781 | 0.9823 | 0.8682 | 0.2275 | 0.7904 | 0.9741 | 0.8726 | 0.2165 |
| | DE | 0.7739 | 0.9861 | 0.8671 | 0.2305 | 0.7865 | 0.9801 | 0.8726 | 0.2178 |
| Breast Cancer | MVOSANN | 0.9781 | **0.9598** | 0.9689 | 0.0403 | 0.9832 | **0.9830** | **0.9831** | **0.0223** |
| | MVO | 0.9767 | 0.9591 | 0.9678 | 0.0416 | 0.9820 | 0.9807 | 0.9813 | 0.0246 |
| | GA | 0.9824 | 0.9507 | 0.9663 | 0.0433 | 0.9862 | 0.9648 | 0.9753 | 0.0322 |
| | FA | 0.9808 | 0.9560 | 0.9683 | 0.0409 | 0.9843 | 0.9718 | 0.9780 | 0.0289 |
| | SSA | 0.9825 | 0.9564 | 0.9693 | 0.0396 | 0.9858 | 0.9718 | 0.9787 | 0.0279 |
| | PSO | 0.9813 | 0.9480 | 0.9643 | 0.0458 | **0.9864** | 0.9611 | 0.9735 | 0.0345 |
| | GWO | 0.9763 | 0.9584 | 0.9673 | 0.0424 | 0.9799 | 0.9817 | 0.9808 | 0.0254 |
| | WOA | 0.9815 | 0.9461 | 0.9634 | 0.0469 | 0.9857 | 0.9658 | 0.9757 | 0.0318 |
| | CS | 0.9791 | 0.9480 | 0.9632 | 0.0473 | 0.9853 | 0.9637 | 0.9742 | 0.0336 |
| | HHO | **0.9837** | 0.9555 | **0.9694** | **0.0394** | 0.9845 | 0.9692 | 0.9768 | 0.0304 |
| | SCA | 0.9768 | 0.9406 | 0.9582 | 0.0535 | 0.9818 | 0.9577 | 0.9695 | 0.0398 |
| | JAYA | 0.9781 | 0.9395 | 0.9583 | 0.0534 | 0.9808 | 0.9550 | 0.9676 | 0.0422 |
| | DE | 0.9726 | 0.9405 | 0.9561 | 0.0565 | 0.9786 | 0.9563 | 0.9672 | 0.0429 |

**Table 6.** Continued

| Datasets | Algorithms | Train | | | | Test | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 Score | Error | Precision | Recall | F1 Score | Error |
| Diabetes | MVOSANN | **0.7666** | **0.5419** | **0.6348** | **0.2119** | **0.7428** | **0.5191** | **0.6109** | **0.2421** |
| | MVO | 0.7604 | 0.5388 | 0.6306 | 0.2146 | 0.7355 | 0.5122 | 0.6037 | 0.2463 |
| | GA | 0.7338 | 0.4674 | 0.5665 | 0.2398 | 0.7074 | 0.4309 | 0.5313 | 0.2754 |
| | FA | 0.7464 | 0.5194 | 0.6124 | 0.2235 | 0.7350 | 0.4931 | 0.5899 | 0.2510 |
| | SSA | 0.7549 | 0.5176 | 0.6140 | 0.2211 | 0.7321 | 0.4931 | 0.5890 | 0.2520 |
| | PSO | 0.7589 | 0.5151 | 0.6133 | 0.2206 | 0.7332 | 0.4750 | 0.5758 | 0.2561 |
| | GWO | 0.7591 | 0.5339 | 0.6268 | 0.2161 | 0.7329 | 0.4990 | 0.5936 | 0.2503 |
| | WOA | 0.6768 | 0.3820 | 0.4853 | 0.2704 | 0.6495 | 0.3670 | 0.4669 | 0.3043 |
| | CS | 0.7275 | 0.4847 | 0.5805 | 0.2374 | 0.6938 | 0.4479 | 0.5427 | 0.2753 |
| | HHO | 0.7144 | 0.4562 | 0.5550 | 0.2469 | 0.6684 | 0.4243 | 0.5177 | 0.2882 |
| | SCA | 0.7142 | 0.4289 | 0.5308 | 0.2540 | 0.6819 | 0.3962 | 0.4960 | 0.2922 |
| | JAYA | 0.7377 | 0.4756 | 0.5741 | 0.2373 | 0.6917 | 0.4424 | 0.5361 | 0.2770 |
| | DE | 0.7282 | 0.4558 | 0.5565 | 0.2441 | 0.7228 | 0.4312 | 0.5353 | 0.2711 |
| Diagnosis I | MVOSANN | **1.0000** | **0.7500** | **0.8571** | **0.1266** | **1.0000** | **0.8667** | **0.9142** | **0.0683** |
| | MVO | **1.0000** | **0.7500** | **0.8571** | **0.1266** | **1.0000** | 0.8349 | 0.8943 | 0.0846 |
| | GA | **1.0000** | **0.7500** | **0.8571** | **0.1266** | **1.0000** | 0.6937 | 0.7995 | 0.1569 |
| | FA | **1.0000** | **0.7500** | **0.8571** | **0.1266** | **1.0000** | 0.6984 | 0.8042 | 0.1545 |
| | SSA | **1.0000** | **0.7500** | **0.8571** | **0.1266** | **1.0000** | 0.7429 | 0.8332 | 0.1317 |
| | PSO | **1.0000** | **0.7500** | **0.8571** | **0.1266** | **1.0000** | 0.7143 | 0.8151 | 0.1463 |
| | GWO | **1.0000** | **0.7500** | **0.8571** | **0.1266** | **1.0000** | 0.8143 | 0.8819 | 0.0951 |
| | WOA | 0.9926 | 0.7417 | 0.8486 | 0.1333 | 0.9856 | 0.6444 | 0.7715 | 0.1894 |
| | CS | **1.0000** | **0.7500** | **0.8571** | **0.1266** | 1.0000 | 0.8175 | 0.8821 | 0.0935 |
| | HHO | **1.0000** | **0.7500** | **0.8571** | **0.1266** | 1.0000 | 0.6778 | 0.7889 | 0.1650 |
| | SCA | **1.0000** | **0.7500** | **0.8571** | **0.1266** | 1.0000 | 0.6667 | 0.7813 | 0.1707 |
| | JAYA | **1.0000** | **0.7500** | **0.8571** | **0.1266** | 1.0000 | 0.7571 | 0.8348 | 0.1244 |
| | DE | **1.0000** | **0.7500** | **0.8571** | **0.1266** | 1.0000 | 0.6556 | 0.7758 | 0.1764 |
| Diagnosis II | MVOSANN | 0.8853 | 0.8812 | **0.8830** | **0.1359** | 0.7993 | **0.9653** | **0.8705** | **0.1667** |
| | MVO | 0.8796 | **0.8848** | 0.8819 | 0.1380 | 0.8242 | 0.9236 | 0.8580 | 0.1748 |
| | GA | 0.8917 | 0.7616 | 0.8039 | 0.2059 | 0.9303 | 0.7222 | 0.7999 | 0.2106 |
| | FA | 0.8839 | 0.8616 | 0.8709 | 0.1473 | 0.9101 | 0.7931 | 0.8289 | 0.1862 |
| | SSA | 0.8876 | 0.8449 | 0.8646 | 0.1532 | 0.9132 | 0.7903 | 0.8291 | 0.1837 |
| | PSO | 0.8725 | 0.8783 | 0.8750 | 0.1460 | 0.8518 | 0.8542 | 0.8313 | 0.1976 |
| | GWO | 0.8919 | 0.8275 | 0.8562 | 0.1599 | 0.9398 | 0.7514 | 0.8221 | 0.1854 |
| | WOA | 0.9277 | 0.6543 | 0.7558 | 0.2363 | 0.9268 | 0.7431 | 0.8103 | 0.2000 |
| | CS | 0.8607 | 0.8362 | 0.8407 | 0.1823 | 0.8910 | 0.7944 | 0.8223 | 0.1943 |
| | HHO | **0.9324** | 0.6783 | 0.7693 | 0.2253 | **0.9744** | 0.7000 | 0.8077 | 0.1927 |
| | SCA | 0.8188 | 0.8072 | 0.7850 | 0.2435 | 0.8715 | 0.7875 | 0.8053 | 0.2171 |
| | JAYA | 0.8764 | 0.7790 | 0.8071 | 0.2084 | 0.9253 | 0.7347 | 0.8026 | 0.2089 |
| | DE | 0.8825 | 0.7420 | 0.7767 | 0.2325 | 0.8928 | 0.7861 | 0.8092 | 0.2089 |
| Glass | MVOSANN | **0.9909** | **0.8892** | **0.9373** | **0.1031** | **1.0000** | **0.8618** | **0.9257** | **0.1190** |
| | MVO | 0.9909 | 0.8851 | 0.9350 | 0.1066 | **1.0000** | 0.8468 | 0.9168 | 0.1319 |
| | GA | 0.9872 | 0.8756 | 0.9280 | 0.1176 | **1.0000** | 0.8403 | 0.9129 | 0.1375 |
| | FA | 0.9890 | 0.8780 | 0.9302 | 0.1141 | **1.0000** | 0.8382 | 0.9117 | 0.1394 |
| | SSA | 0.9897 | 0.8840 | 0.9338 | 0.1085 | **1.0000** | 0.8516 | 0.9197 | 0.1278 |
| | PSO | 0.9890 | 0.8780 | 0.9301 | 0.1141 | **1.0000** | 0.8548 | 0.9214 | 0.1250 |
| | GWO | 0.9906 | 0.8821 | 0.9332 | 0.1094 | **1.0000** | 0.8435 | 0.9150 | 0.1347 |
| | WOA | 0.9773 | 0.8783 | 0.9247 | 0.1232 | **1.0000** | 0.8570 | 0.9221 | 0.1231 |
| | CS | 0.9841 | 0.8821 | 0.9301 | 0.1146 | **1.0000** | **0.8618** | 0.9253 | **0.1190** |
| | HHO | 0.9874 | **0.8892** | 0.9356 | 0.1059 | **1.0000** | 0.8554 | 0.9219 | 0.1245 |
| | SCA | 0.9809 | 0.8694 | 0.9214 | 0.1279 | 0.9988 | 0.8543 | 0.9202 | 0.1264 |
| | JAYA | 0.9852 | 0.8715 | 0.9246 | 0.1228 | **1.0000** | 0.8274 | 0.9051 | 0.1486 |
| | DE | 0.9844 | 0.8585 | 0.9163 | 0.1345 | 0.9993 | 0.8376 | 0.9097 | 0.1403 |

**Table 6.** Continued

| Datasets | Algorithms | Train | | | | Test | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 Score | Error | Precision | Recall | F1 Score | Error |
| Iris | MVOSANN | **1.0000** | **0.6242** | **0.7682** | **0.2480** | **1.0000** | **0.6480** | **0.7856** | **0.2393** |
| | MVO | **1.0000** | 0.6081 | 0.7559 | 0.2587 | **1.0000** | 0.6196 | 0.7645 | 0.2587 |
| | GA | **1.0000** | 0.5384 | 0.6997 | 0.3047 | **1.0000** | 0.5637 | 0.7201 | 0.2967 |
| | FA | **1.0000** | 0.6010 | 0.7501 | 0.2633 | **1.0000** | 0.6275 | 0.7701 | 0.2533 |
| | SSA | **1.0000** | 0.5672 | 0.7236 | 0.2857 | **1.0000** | 0.5990 | 0.7489 | 0.2727 |
| | PSO | **1.0000** | 0.5934 | 0.7438 | 0.2683 | **1.0000** | 0.6147 | 0.7592 | 0.2620 |
| | GWO | **1.0000** | 0.5601 | 0.7179 | 0.2903 | **1.0000** | 0.5980 | 0.7481 | 0.2733 |
| | WOA | 0.9990 | 0.5202 | 0.6839 | 0.3170 | **1.0000** | 0.5618 | 0.7185 | 0.2980 |
| | CS | **1.0000** | 0.5505 | 0.7096 | 0.2967 | 1.0000 | 0.5804 | 0.7336 | 0.2853 |
| | HHO | **1.0000** | 0.5283 | 0.6911 | 0.3113 | 1.0000 | 0.5676 | 0.7235 | 0.2940 |
| | SCA | **1.0000** | 0.5439 | 0.7041 | 0.3010 | 1.0000 | 0.5686 | 0.7242 | 0.2933 |
| | JAYA | **1.0000** | 0.5601 | 0.7168 | 0.2903 | 1.0000 | 0.5814 | 0.7331 | 0.2847 |
| | DE | **1.0000** | 0.5576 | 0.7152 | 0.2920 | 1.0000 | 0.5824 | 0.7346 | 0.2840 |
| Liver | MVOSANN | **0.7262** | **0.5474** | **0.6241** | **0.2759** | **0.7444** | **0.6920** | **0.7170** | **0.2314** |
| | MVO | 0.7113 | 0.5365 | 0.6115 | 0.2852 | 0.7365 | 0.6860 | 0.7101 | 0.2373 |
| | GA | 0.6613 | 0.4421 | 0.5243 | 0.3302 | 0.6932 | 0.5273 | 0.5915 | 0.3048 |
| | FA | 0.6953 | 0.5161 | 0.5922 | 0.2972 | 0.7144 | 0.6427 | 0.6761 | 0.2605 |
| | SSA | 0.6837 | 0.4937 | 0.5730 | 0.3075 | 0.7119 | 0.6327 | 0.6691 | 0.2641 |
| | PSO | 0.7144 | 0.5260 | 0.6045 | 0.2872 | 0.7250 | 0.6460 | 0.6823 | 0.2545 |
| | GWO | 0.6820 | 0.5140 | 0.5860 | 0.3038 | 0.7251 | 0.6687 | 0.6953 | 0.2483 |
| | WOA | 0.6353 | 0.3414 | 0.4367 | 0.3568 | 0.6169 | 0.3607 | 0.4446 | 0.3678 |
| | CS | 0.6743 | 0.4365 | 0.5240 | 0.3266 | 0.6812 | 0.5160 | 0.5789 | 0.3130 |
| | HHO | 0.6747 | 0.3842 | 0.4823 | 0.3361 | 0.6487 | 0.4300 | 0.5114 | 0.3370 |
| | SCA | 0.6509 | 0.3961 | 0.4716 | 0.3501 | 0.6844 | 0.4533 | 0.5166 | 0.3342 |
| | JAYA | 0.6589 | 0.4354 | 0.5098 | 0.3373 | 0.6627 | 0.4933 | 0.5436 | 0.3339 |
| | DE | 0.6849 | 0.4042 | 0.4882 | 0.3363 | 0.6908 | 0.4640 | 0.5261 | 0.3339 |
| Raisin | MVOSANN | 0.8713 | **0.8897** | **0.8804** | **0.1209** | 0.8291 | **0.9059** | **0.8658** | **0.1404** |
| | MVO | 0.8703 | 0.8872 | 0.8787 | 0.1225 | 0.8304 | 0.9035 | 0.8654 | 0.1405 |
| | GA | **0.8764** | 0.8617 | 0.8688 | 0.1300 | 0.8334 | 0.8697 | 0.8508 | 0.1523 |
| | FA | 0.8717 | 0.8763 | 0.8740 | 0.1264 | 0.8304 | 0.8867 | 0.8575 | 0.1473 |
| | SSA | 0.8740 | 0.8728 | 0.8734 | 0.1265 | 0.8347 | 0.8802 | 0.8568 | 0.1471 |
| | PSO | 0.8725 | 0.8808 | 0.8762 | 0.1241 | 0.8276 | 0.8867 | 0.8554 | 0.1493 |
| | GWO | 0.8738 | 0.8725 | 0.8731 | 0.1268 | **0.8360** | 0.8852 | 0.8599 | 0.1442 |
| | WOA | 0.8681 | 0.8510 | 0.8591 | 0.1395 | 0.8271 | 0.8490 | 0.8373 | 0.1646 |
| | CS | 0.8720 | 0.8618 | 0.8662 | 0.1328 | 0.8288 | 0.8715 | 0.8486 | 0.1550 |
| | HHO | 0.8730 | 0.8504 | 0.8612 | 0.1369 | 0.8341 | 0.8497 | 0.8414 | 0.1598 |
| | SCA | 0.8687 | 0.8637 | 0.8652 | 0.1342 | 0.8257 | 0.8656 | 0.8436 | 0.1596 |
| | JAYA | 0.8721 | 0.8568 | 0.8639 | 0.1347 | 0.8286 | 0.8551 | 0.8406 | 0.1615 |
| | DE | 0.8750 | 0.8455 | 0.8589 | 0.1383 | 0.8348 | 0.8420 | 0.8369 | 0.1631 |
| Vertebral | MVOSANN | 0.8738 | 0.8802 | **0.8770** | **0.1634** | **0.9127** | 0.9147 | **0.9136** | **0.1223** |
| | MVO | 0.8732 | 0.8802 | 0.8767 | 0.1639 | 0.9096 | 0.9116 | 0.9105 | 0.1267 |
| | GA | 0.8400 | 0.8825 | 0.8600 | 0.1902 | 0.8598 | 0.8902 | 0.8742 | 0.1811 |
| | FA | 0.8635 | 0.8810 | 0.8720 | 0.1712 | 0.8901 | 0.9009 | 0.8953 | 0.1491 |
| | SSA | 0.8580 | **0.8933** | 0.8752 | 0.1686 | 0.8886 | **0.9182** | 0.9030 | 0.1396 |
| | PSO | **0.8805** | 0.8627 | 0.8700 | 0.1699 | 0.9027 | 0.8756 | 0.8868 | 0.1560 |
| | GWO | 0.8694 | 0.8832 | 0.8761 | 0.1654 | 0.9040 | 0.9138 | 0.9087 | 0.1299 |
| | WOA | 0.7950 | 0.8719 | 0.8303 | 0.2358 | 0.8262 | 0.8800 | 0.8513 | 0.2167 |
| | CS | 0.8402 | 0.8802 | 0.8578 | 0.1931 | 0.8662 | 0.8880 | 0.8753 | 0.1780 |
| | HHO | 0.8169 | 0.8756 | 0.8446 | 0.2134 | 0.8376 | 0.8813 | 0.8585 | 0.2053 |
| | SCA | 0.8167 | 0.8716 | 0.8403 | 0.2193 | 0.8349 | 0.8818 | 0.8557 | 0.2094 |
| | JAYA | 0.8320 | 0.8810 | 0.8518 | 0.2031 | 0.8581 | 0.8751 | 0.8633 | 0.1943 |
| | DE | 0.8394 | 0.8689 | 0.8508 | 0.2008 | 0.8636 | 0.8711 | 0.8643 | 0.1912 |

In Fig. 7 and Fig. 8, convergence curves are given to observe the convergence behaviour of the MVOSANN algorithm and to compare its performance with other algorithms. The figure shows that MVOSANN has the fastest convergence rate and the lowest MSE values in all datasets. MVOSANN algorithm has acquired powerful results by surpassing other algorithms in terms of robustness and stability.

The MVOSANN algorithm outclasses all other algorithms used in the comparing. To understand the relationship among the proposed algorithm and other algorithms, non-parametric Wilcoxon rank-sum statistical test was applied. The results of this test are shown in Table 7. The algorithm with the best average value gets the N/A value and is compared with other algorithms. According to table, the MVOSANN algorithm has a p value less than 0.05 for all data sets and for all algorithms, and it is seen to be statistically significant.

MLP training is a problem that has many local solutions and is also difficult to solve. As can be seen from the results obtained, the MVOSANN algorithm shows the best productivity against this difficult problem compared to the other 12 algorithms used for comparison. It has also proven this success in all the different search spaces of 12 different data sets. The reason for this success of MVOSANN is the high exploration - exploitation ability it has and the ability to avoid the local minimum.

**Table 7**
Wilcoxon rank sum test *p*-values (N/A = not applicable)

| Datasets | Algorithms | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MVOSANN | MVO | GA | FA | SSA | PSO | GWO | WOA | CS | HHO | SCA | JAYA | DE |
| Abalone | N/A | 4.98E-04 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Balance | N/A | 5.10E-05 | 2.87E-11 | 2.74E-10 | 7.03E-11 | 5.31E-08 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Blood | N/A | 4.97E-03 | 2.87E-11 | 7.73E-10 | 1.04E-10 | 9.18E-07 | 9.44E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 5.23E-11 | 3.88E-11 |
| Breast Cancer | N/A | 1.54E-04 | 2.87E-11 | 3.88E-11 | 3.51E-11 | 4.29E-11 | 1.06E-08 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Diabetes | N/A | 2.92E-04 | 2.87E-11 | 4.29E-11 | 2.87E-11 | 5.77E-11 | 6.37E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Diagnosis I | N/A | 9.77E-04 | 2.87E-11 | 3.06E-09 | 5.23E-11 | 2.96E-03 | 1.15E-10 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Diagnosis II | N/A | 6.97E-06 | 2.87E-11 | 1.86E-10 | 2.87E-11 | 0.054609 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Glass | N/A | 1.41E-03 | 2.87E-11 | 4.29E-11 | 9.44E-11 | 2.87E-11 | 5.23E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Iris | N/A | 3.26E-03 | 2.87E-11 | 8.93E-05 | 7.03E-11 | 8.70E-08 | 1.04E-10 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Liver | N/A | 9.85E-06 | 2.87E-11 | 4.01E-10 | 2.87E-11 | 1.29E-05 | 3.88E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Raisin | N/A | 4.97E-03 | 2.87E-11 | 4.00E-09 | 2.87E-11 | 4.22E-05 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |
| Vertebral | N/A | 5.39E-07 | 2.87E-11 | 2.74E-10 | 2.87E-11 | 1.63E-08 | 7.76E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 | 2.87E-11 |

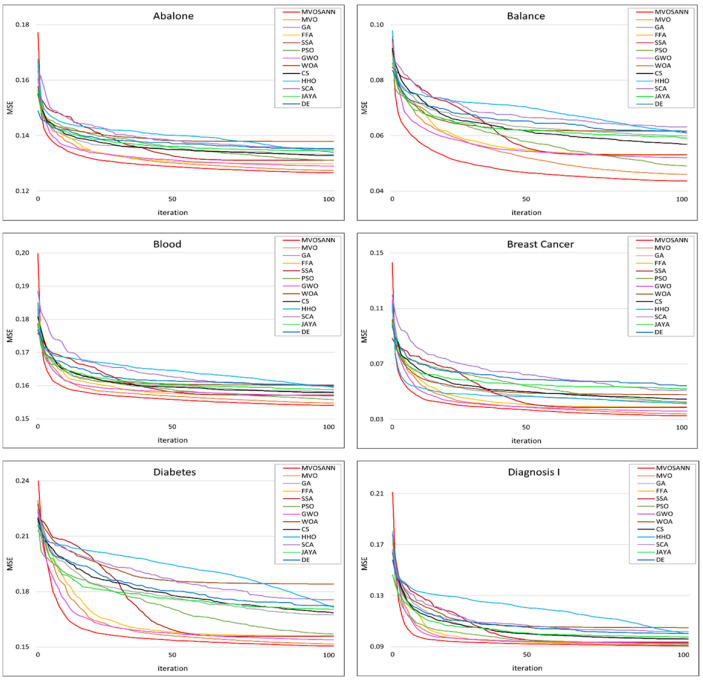


**Fig. 7.** Convergence curves of Abalone, Balance, Blood, Breast Cancer, Diabetes, Diagnosis I datasets.

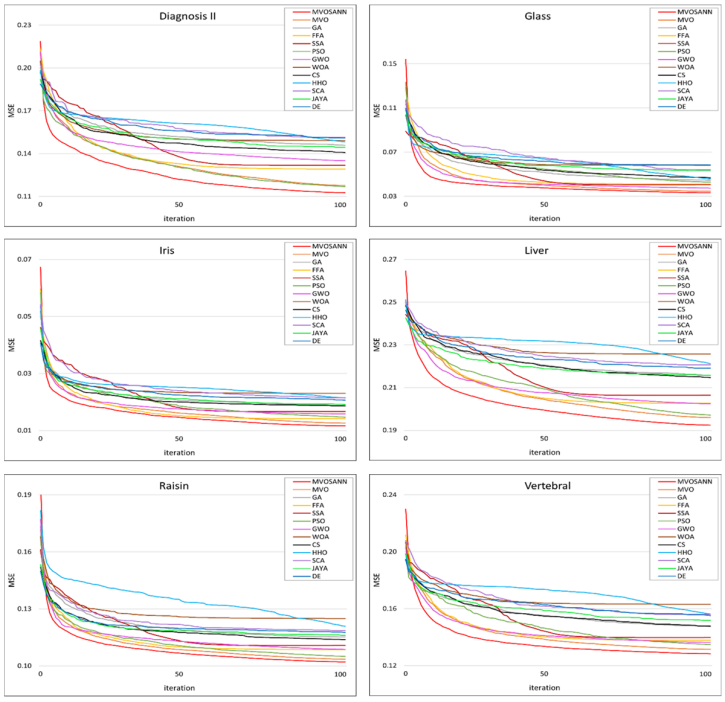


**Fig. 8.** Convergence curves of Diagnosis II, Glass, Iris, Liver, Raisin, Vertebral datasets.

## 6. Conclusion remarks

One of the problems met in implementations of ANNs is related to the training of the ANNs model. Training in ANNs is one of the determinants that directly affects the achievement of the model. In the studies, it is seen that many different techniques are used in ANNs training. Today, meta-heuristic algorithms are preferred over traditional algorithms during the training stage in ANNS models. Meta-heuristic algorithms appear to produce more effective results than traditional algorithms. An algorithm cannot be expected to perform highly against every problem. Therefore, it may be necessary to work on different algorithms for different problems or to develop existing algorithms.

In this article, the MVOSANN algorithm, which was developed based on the MVO and SA algorithms, is tested for the training of the feed forward MLP. The suggested algorithm (MVOSANN) has been used to optimize biases and weights in MLP for better results. The algorithm has been experimented on 12 datasets with different characteristics. The outcomes were compared with the outcomes of 12 various meta-heuristic algorithms. Experimental results suggest that the MVOSANN algorithm significantly enhances the performance of the MVO algorithm and carries out better than other algorithms. As a result, MVOSANN is an effective alternative for training ANNs.

In the future, larger and different datasets can be studied with the proposed MVOSANN algorithm. In addition, the MVOSANN algorithm can be experimented for the training of different model neural networks or real-world problems.

## References

Abedinia, O., & Amjady, N. (2015). Short-term wind power prediction based on Hybrid Neural Network and chaotic shark smell optimization. *International journal of precision engineering and manufacturing-green technology*, *2*(3), 245-254.

Abusnaina, A. A., Ahmad, S., Jarrar, R., & Mafarja, M. (2018, June). Training neural networks using salp swarm algorithm for pattern classification. In *Proceedings of the 2nd international conference on future networks and distributed systems* (pp. 1-6).

Alboaneen, D. A., Tianfield, H., & Zhang, Y. (2017, December). Glowworm swarm optimization for training multi-layer perceptrons. In *Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies* (pp. 131-138).

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

Aljarah, I., Faris, H., Mirjalili, S., & Al-Madi, N. (2018b). Training radial basis function networks using biogeography-based optimizer. *Neural Computing and Applications*, *29*(7), 529-553.

Alweshah, M. (2014). Firefly algorithm with artificial neural network for time series problems. *Research Journal of Applied Sciences, Engineering and Technology*, *7*(19), 3978-3982.

Anderson, J. A. (1995). *An introduction to neural networks*. MIT press.

Bairathi, D., & Gopalani, D. (2019). Salp swarm algorithm (SSA) for training feed-forward neural networks. In *Soft computing for problem solving* (pp. 521-534). Springer, Singapore.

Bennett, K. P., & Mangasarian, O. L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, *1*(1), 23-34.

Brajevic, I., & Tuba, M. (2013). Training feed-forward neural networks using firefly algorithm. In *Proceedings of the 12th International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED'13)* (pp. 156-161).

Czerniak, J., & Zarzycki, H. (2003). Application of rough sets in the presumptive diagnosis of urinary system diseases. In *Artificial intelligence and security in computing systems* (pp. 41-51). Springer, Boston, MA.

Çınar, İ., Köklü, M., & Taşdemir, Ş. (2020). Classification of raisin grains using machine vision and artificial intelligence methods. *Gazi Mühendislik Bilimleri Dergisi (GMBD)*, *6*(3), 200-209.

Dua, D., & Graff, C. (2019). UCI Machine Learning Repository [http://archive. ics. uci. edu/ml]. Irvine, CA: University of California. *School of Information and Computer Science*.

Everett, H., Wheeler, J. A., DeWitt, B. S., Cooper, L. N., Van Vechten, D. & Graham, N. (1973). DeWitt, Bryce; Graham, R. Neill (eds.). *The Many-Worlds Interpretation of Quantum Mechanics*. Princeton Series in Physics. Princeton, NJ: Princeton University Press.

Faris, H. (2016). *EVOLOPY_NN*. [Repository]. https://github.com/7ossam81/EvoloPy-NN

Faris, H., Aljarah, I., & Mirjalili, S. (2016a). Training feedforward neural networks using multi-verse optimizer for binary classification problems. *Applied Intelligence*, *45*(2), 322-332.

Faris, H., Aljarah, I., Al-Madi, N., & Mirjalili, S. (2016b). Optimizing the learning process of feedforward neural networks using lightning search algorithm. *International Journal on Artificial Intelligence Tools*, *25*(06), 1650033.

Faris, H., Aljarah, I., Mirjalili, S., Castillo, P. A., & Guervós, J. J. M. (2016c). EvoloPy: An Open-source Nature-inspired Optimization Framework in Python. In *IJCCI (ECTA)* (pp. 171-177).

Faris, H., Aljarah, I., & Mirjalili, S. (2018). Improved monarch butterfly optimization for unconstrained global search and neural network training. *Applied Intelligence*, *48*(2), 445-464.

Gori, M., & Tesi, A. (1992). On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(1), 76-86.

Gupta, J. N., & Sexton, R. S. (1999). Comparing backpropagation with a genetic algorithm for neural network training. *Omega*, *27*(6), 679-684.

Hasan, S., Tan, S. Q., Shamsuddin, S. M., & Sallehuddin, R. (2011). Artificial neural network learning enhancement using artificial fish swarm algorithm. In *Proceedings of the 3rd International Conference on Computing and Informatics (ICOCI),* (pp 117–122).

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris Hawks Optimization: Algorithm and Applications. *Future Generation Computer Systems*, 97, 849–872.

Heisenberg, W. (1985). Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik. In *Original Scientific Papers Wissenschaftliche Originalarbeiten* (pp. 478-504). Springer, Berlin, Heidelberg.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, *267*(1), 66-73.

Jia, H., Peng, X., Song, W., Lang, C., Xing, Z., & Sun, K. (2019). Hybrid multiverse optimization algorithm with gravitational search algorithm for multithreshold color image segmentation. *IEEE Access*, *7*, 44903-44927.

Karaboga, D., Akay, B., & Ozturk, C. (2007). Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In *International conference on modeling decisions for artificial intelligence* (pp. 318-329). Springer, Berlin, Heidelberg.

Kaya, S., & Fığlalı, N. (2018). Çok amaçlı esnek atölye tipi çizelgeleme problemlerinin çözümünde meta sezgisel yöntemlerin kullanımı. *Harran Üniversitesi Mühendislik Dergisi*, *3* (3), 222-233.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, (pp. 1942– 1948), IEEE.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

Kolay, E., Tunç, T., & Eğrioğlu, E. (2016). Classification with some artificial neural network classifiers trained a modified particle swarm optimization. *American Journal of Intelligent Systems*, *6*(3), 59-65.

Mafarja, M. M., & Mirjalili, S. (2017). Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing*, *260*, 302-312.

McCulloch, W.S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5**,** 115–133 (1943)

Mirjalili, S., & Hashim, S. Z. M. (2010). A new hybrid PSOGSA algorithm for function optimization. In *2010 international conference on computer and information application* (pp. 374-377). IEEE.

Mirjalili, S., & Sadiq, A. S. (2011). Magnetic optimization algorithm for training multi layer perceptron. In *2011 IEEE 3rd International Conference on Communication Software and Networks* (pp. 42-46). IEEE.

Mirjalili, S., Hashim, S. Z. M., & Sardroudi, H. M. (2012). Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*, *218*(22), 11125-11137.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014a). Let a biogeography-based optimizer train your multi-layer perceptron. *Information Sciences*, *269*, 188-209.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014b). Grey wolf optimizer. *Advances in engineering software*, *69*, 46-61.

Mirjalili, S. (2015). How effective is the Grey Wolf optimizer in training multi-layer perceptrons. *Applied Intelligence*, *43*(1), 150-161

Mirjalili, S. Z., Saremi, S., & Mirjalili, S. M. (2015). Designing evolutionary feedforward neural networks using social spider optimization algorithm. *Neural Computing and Applications*, *26*(8), 1919-1928.

Mirjalili, S. (2016). SCA: a sine cosine algorithm for solving optimization problems. *Knowledge-based systems*, *96*, 120-133.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software*, *95*, 51-67.

Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications*, *27*(2), 495-513.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163-191.

Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*, *7*(1), 19-34.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning Internal Representations by Error Propagation. In D. E. Rumelhart & J. L. Mcclelland (ed.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations* (pp. 318--362) . MIT Press.

Seiffert, U. (2001). Multiple layer perceptron training using genetic algorithms. In *ESANN* (pp. 159-164).

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, *45*(4), 427-437.

Song, R., Zeng, X., & Han, R. (2020). An Improved Multi-Verse Optimizer Algorithm For Multi-Source Allocation Problem. *International Journal of Innovative Computing, Information and Control, 16*(6), 1845–1862.

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, *62*(1), 77-89.

Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, 11(4), 341–359.

Talbi, E. G. (2009). *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons.

Ting, T. O., Yang, X. S., Cheng, S., & Huang, K. (2015). Hybrid metaheuristic algorithms: past, present, and future. *Recent advances in swarm intelligence and evolutionary computation*, 71-83.

Tuba, M., Alihodzic, A., & Bacanin, N. (2015). Cuckoo search and bat algorithm applied to training feed-forward neural networks. In *Recent advances in swarm intelligence and evolutionary computation* (pp. 139-162). Springer, Cham.

Wilcoxon, F. (1992). Individual Comparisons by Ranking Methods. *Breakthroughs in Statistics*, 196–202, Springer, New York, NY.

Wolberg, W. H., & Mangasarian, O. L. (1990). Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the national academy of sciences*, *87*(23), 9193-9196.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67-82.

Wu, H., Zhou, Y., Luo, Q., & Basset, M. A. (2016). Training feedforward neural networks using symbiotic organisms search algorithm. *Computational intelligence and neuroscience*, *2016*.

Yamany, W., Fawzy, M., Tharwat, A., & Hassanien, A. E. (2015). Moth-flame optimization for training multi-layer perceptrons. In *2015 11th International computer engineering Conference (ICENCO)* (pp. 267-272). IEEE.

Yang, X. S. (2008). *Nature-inspired metaheuristic algorithms*, Luniver press. *Beckington, UK*, 242-246.

Yang, X. S., & Deb, S. (2009). Cuckoo Search via L´evy Flights. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC),* (pp. 210–214), IEEE.

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms* (pp. 169-178). Springer, Berlin, Heidelberg.

Yeh, I. C., Yang, K. J., & Ting, T. M. (2009). Knowledge discovery on RFM model using Bernoulli sequence. *Expert Systems with Applications*, *36*(3), 5866-5871.

Yılmaz, Ö., Altun, A., & Köklü, M. (2022). A new hybrid algorithm based on MVO and SA for function optimization. *International Journal of Industrial Engineering Computations*, *13*(2), 237-254.