mathematics

Article

Feed-Forward Neural Networks Training with Hybrid Taguchi Vortex Search Algorithm for Transmission Line Fault Classification

Melih Coban [1,2,*] and Suleyman Sungur Tezcan [2]

[1] Department of Electrical Electronic Engineering, Bolu Abant Izzet Baysal University, Golkoy, Bolu 14030, Turkey
[2] Department of Electrical-Electronic Engineering, Gazi University, Maltepe, Ankara 06570, Turkey
* Correspondence: melihcoban@ibu.edu.tr or melih.coban@gazi.edu.tr

Abstract: In this study, the hybrid Taguchi vortex search (HTVS) algorithm, which exhibits a rapid convergence rate and avoids local optima, is employed as a new training algorithm for feed-forward neural networks (FNNs) and its performance was analyzed by comparing it with the vortex search (VS) algorithm, the particle swarm optimization (PSO) algorithm, the gravitational search algorithm (GSA) and the hybrid PSOGSA algorithm. The HTVS-based FNN (FNNHTVS) algorithm was applied to three datasets (iris classification, wine recognition and seed classification) taken from the UCI database (the machine learning repository of the University of California at Irvine) and to the 3-bit parity problem. The obtained statistical results were recorded for comparison. Then, the proposed algorithm was used for fault classification on transmission lines. A dataset was created using 735 kV, 60 Hz, 100 km transmission lines for different fault types, fault locations, fault resistance values and fault inception angles. The FNNHTVS algorithm was applied to this dataset and its performance was tested in comparison with that of other classifiers. The results indicated that the performance of the FNNHTVS algorithm was at least as successful as that of the other comparison algorithms. It has been shown that the FNN model trained with HTVS can be used as a capable alternative algorithm for the solution of classification problems.

Keywords: fault classification; HTVS algorithm; optimization; training feed-forward neural networks

MSC: 65K10; 68T07

check for updates

Citation: Coban, M.; Tezcan, S.S. Feed-Forward Neural Networks Training with Hybrid Taguchi Vortex Search Algorithm for Transmission Line Fault Classification. *Mathematics* **2022**, *10*, 3263. https://doi.org/10.3390/math10183263

Academic Editors: Xinchao Zhao, Xingquan Zuo, Yinan Guo and Kunpeng Kang

Received: 17 August 2022
Accepted: 6 September 2022
Published: 8 September 2022

1. Introduction

An artificial neural network (ANN) is a computational model based on the human nervous system and it is a useful modeling tool. For this reason, ANNs have been researched with interest in many disciplines such as engineering, finance, technology, etc. ANN structures inspired by biological neural networks have been developed and used in classification [1–3], signal processing [4,5] and prediction tasks [6–8], as well as in various other studies [9–12]. In the successful use of an ANN, it is important to choose the training algorithm, the activation function in the neurons, the neural network structure and the parameters (weights and biases) correctly. The training algorithms used in the training of networks aim to create a suitable network structure for the problem by finding the optimal weights and bias parameters. For example, studies have been conducted in an attempt to find the optimal weights and biases by keeping the network topology and activation function constant [13–15].

There is a need for a training set that includes suitable features for network training. The parameters of the network are regulated by the training algorithms using the training data [16]. In this context, the main purpose of network training is to ensure harmony between network output and real output by means of training algorithms.

There are many algorithms and methods in the literature that can be used in ANN training. The most commonly used mathematical methods are the back-propagation (BP) [17], gradient descent (GD) [18], conjugate gradient (CG) [19] and Levenberg–Marquardt (LM) methods [20]. Many heuristic algorithms can be used to construct the appropriate network in FNN training.

In [14], the PSOGSA algorithm was proposed for FNN training. The obtained results were compared with the PSO-based FNN (FNNPSO). It has been observed that the PSOGSA-based FNN (FNNPSOGSA) algorithm produces better results compared to the PSO-based FNN (FNNPSO) and GSA-based FNN (FNNGSA) algorithms.

A study was conducted to investigate the effectiveness of the use of the VS algorithm in FNN training [13]. In [13], the performance of the VS-based FNN (FNNVS) was compared with the performance of an FNN trained with other optimization algorithms using different classification problems. The obtained results showed that the VS algorithm can be used in FNN training. Furthermore, the discrete-continuous version of the vortex search algorithm was used to determine the sizes and locations of PV sources [21]. In [22], the optimal selection of conductors in three-phase distribution networks was performed through the use of a discrete version of the vortex search algorithm.

It can be used in models obtained as a result of hybridizing classical training algorithms and heuristic optimization algorithms in ANN training. In [23], a new method was presented, based on hybridizing the artificial bee colony (ABC) algorithm and the LM algorithm (ABC-LM). The authors carried out this study to prevent the LM algorithm from getting stuck on local minimums and the ABC algorithm converged slowly to global minimums.

Heidari et al. [24] presented a stochastic training algorithm in their study. They suggested that the grasshopper optimization algorithm (GOA) performed well in the solution of optimization problems and could also be used in the training of multilayer perceptron (MLP) neural networks. The GOAMLP model was compared with other efficient algorithms using five different classification problems. The authors stated that the use of GOAMLP contributed to obtaining accurate classification performance.

In [25], the dragonfly algorithm (DA) was used in FNN training. Experiments were conducted on classification problems and a civil engineering problems. The obtained results showed that the DA was quite successful in FNN training. Additionally, they tried to emphasize the avoidance of the local optima.

In [26], weights and biases parameters of the FNN were optimized by means of the whale optimization algorithm (WOA). Within the scope of the study, comparisons were made with different algorithms through classification problems. The authors stated that it performed better in terms of its avoidance of the local optimum and its convergence rate. In addition to the studies mentioned above, other studies have been conducted using optimization algorithms for ANN training. These include studies of the krill-herd algorithm (KHA) [27], the cuckoo search (CS) algorithm [28] and the the symbiotic organism search (SOS) algorithm [29]. Table 1 presents some algorithms used in FNN training. The main purpose of these studies was to train the FNN structure in the best way. The main difference between these studies is that they used different algorithms from one another. The algorithm presented in this study is different from these, and it was also used for transmission line fault classification.

Table 1. Brief summary of algorithms used in the literature for FNN training.

Reference	Algorithm
Faris, H. et al., 2016 [1]	Multi-Verse Optimizer
Sag, T. et al., 2021 [13]	Vortex Search Algorithm
Mirjalili, S. et al., 2012 [14]	Hybrid PSO-GSA algorithm
Pashaei, E. et al., 2021 [15]	Enhanced Black Hole Algorithm
Hagan, M.T. et al., 1994 [20]	Marquardt Algorithm
Ozturk, C. et al., 2011 [23]	Hybrid Artificial Bee Colony Algorithm
Heidari, A.A. et al., 2019 [24]	Grasshopper Optimization Algorithm
Gulcu, S., 2022 [25]	Dragonfly Algorithm
Aljarah, I. et al., 2018 [26]	Whale Optimization Algorithm
Lari, N.S. et al., 2014 [27]	Krill-Herd Algorithm
Jiao-hong, Y. et al., 2014 [28]	Cuckoo Search Algorithm
Wu, H. et al., 2016 [29]	Symbiotic Organisms Search Algorithm
Zhang, J.R. et al., 2007 [30]	Hybrid PSO-BP Algorithm
Mirjalili, S. et al., 2014 [31]	Biogeography-based Optimizer

To the best of our knowledge, this is the first study conducted on HTVS-based FNN training. In this study, our main purpose was not to find the most suitable FNN structure for a test problem or to obtain the smallest error value that could be achieved. Rather,the primary purpose of this study was to present the use of the HTVS algorithm [32] in FNN training and to compare its performance with that of the VS [33], PSO [34], PSOGSA [14] and GSA [35]. Therefore, 3-bit parity, iris classification, wine recognition and seed classification benchmark datasets were used for performance comparisons. In order to show that the HTVS algorithm had a competitive character compared to other algorithms used in FNN training, tests were conducted using different hidden neuron numbers in the FNN structure.

The second main purpose of the study was to show that the proposed algorithm can be used in fault classification on transmission lines. For this purpose, a transmission line of 735 kV, 60 Hz and 100 km longwas modeled as frequency-dependent with the help of Matlab/Simulink. Fault data were produced and recorded on the modeled transmission line. Using these data, the FNNHTVS algorithm and the optimization algorithm-based FNNVS, FNNPSO, FNNPSOGSA and FNNGSA algorithms were compared. Additionally, the performance of the proposed algorithm was compared with that of classifiers such as a support vector machine (SVM), the K-nearest neighbor (KNN) method and an FNN with LM and Naive Bayes (NB). The results showed that the FNNHTVS algorithm was quite successful.

The main contributions of this study are briefly listed as follows.

- The HTVS algorithm is presented for the first time as an alternative algorithm to overcome slow convergence and local optimum problems in FNN training.
- The effectiveness of the HTVS algorithm in FNN training is demonstrated.
- It has been proven that the FNNHTVS structure can achieve results comparable to and better than those of other successful algorithms in classification studies.
- It has been shown that the FNNHTVS algorithm can be used as an alternative algorithm for transmission line short-circuit fault classification tasks.

The remainder of this paper is organized as follows. In Section 2 we explain the basics concept of the FNN, the HTVS algorithm and FNN training using HTVS. In Section 3 we present the experimental results and a discussion of the performance of the algorithms. In Section 4 we present an evaluation of the performance of FNNHTVS in fault classification. In Section 5, we present our conclusions.

2. Basic Principles

2.1. Feed-Forward Neural Network

FNNs are neural networks that have forward data flow. They are frequently used in classification and regression problems. Neurons are represented as processing units for FNNs. In the structure of each neuron, there are activation functions that may be radial, linear, sigmoid, etc. Neurons generate output data based on input data using activation functions. In FNNs, neurons in each layer are fed only by the neurons of the previous layer. Neurons are arranged in layers and the outputs of neurons in one layer are input to the next layer over weights. The input layer transmits the information it receives from the external environment to the neurons in the next layer without making any changes [36]. The first layer is the input, the last layer is the output and the layers between these two are called hidden layers. Figure 1 presents the basic structure of a three-layer FNN. The number of neurons in the input and output layer varies depending on the nature of the problem. The number of hidden layers and the number of neurons in the hidden layers are chosen according to the complexity of the problem being studied [13]. The overall goal of network training is to minimize the difference between the target output and the achieved output. The FNN training process is completed by updating the weights in its structure and the bias parameters used to balance these weights in each cycle.

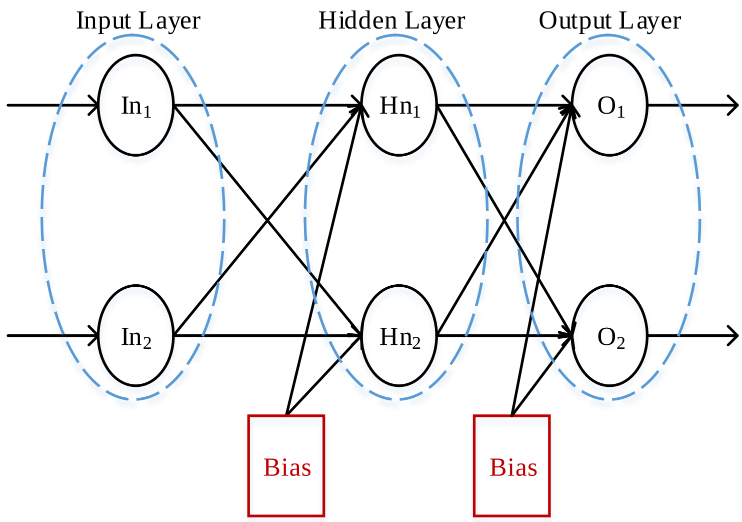

Figure 1. General FNN structure for (2-2-2).

2.2. HTVS Algorithm

HTVS is an optimization algorithm created by hybridizing the VS algorithm and the Taguchi orthogonal array approach (TOAA) [32]. This algorithm has shown successful results in optimization problems [32]. For this algorithm the use of orthogonal arrays (OAs) in the population generation phase is preferable. Since there are OAs in the structure of the HTVS algorithm, the computational cost is slightly higher than that of VS. However, the disadvantages of VS, such as its slow convergence and the fact that it can become trapped in local minima, are compensated for in this way. In the HTVS algorithm, randomly generated candidate solutions are evenly distributed in the search space via TOAA. The developed candidate solutions are used for the VS algorithm. HTVS is an optimization algorithm that can achieve highly effective results using fewer iterations.

The working principle of HTVS can be briefly explained as follows. Firstly, candidate solutions are distributed through TOAA. In the OA, columns represent the parameters that need to be optimized. Each row describes a possible combination of the level values for these parameters. The problem size and OA columns are compatible with each other. Secondly, OA-related level values are determined for each candidate solution in order to improve the candidate solutions produced. Optimum level values are determined for each candidate solution and they are selected for OA training. Finally, the optimized candidate solutions are sent to the VS search space (circle). The best solution produced by the candidate solutions is determined as the best of that iteration. If the solution obtained as a result of an iteration is better than the previous results, it is saved and kept as the best solution. These operations are performed until a specified number of iterations has been reached. The pseudo-code of the HTVS is presented in Algorithm 1. At the beginning of the algorithm, necessary definitions, such as problem boundaries, size, number of iterations, reduction ratio coefficient, etc., are set. The desired OA is created according to the problem dimensions. Then, candidate solutions are created and checked to see if they are within the boundaries. Each level value is determined for each candidate solution. These level values are associated with the OA. Optimum level values are determined for each parameter. The level difference is reduced by means of the reduction ratio coefficient and this process is continued until the target error value is reached. Thus, the candidate solutions are improved. The improved candidate solutions are sent to the vortex circle for examination. If the iteration's best value is better than the global best value, the iteration's best value is selected and recorded as the global best value. Then, the radius of the vortex circle is updated and reduced. The algorithm's steps continue until the maximum iteration number is reached. More detailed information about the HTVS and VS algorithms can be found in [32,33], respectively.

Algorithm 1: HTVS Algorithm.

```
Start Algorithm
Define algorithm parameters;
Define problem dimensions;
Generate OA;
for max. iter. do
    Shift candidate solutions into boundaries;
    for Each candidate solutions do
        Define Level Difference;
        while Error value > Target error value do
            Determine candidate solution levels;
            Select optimal level;
            Find improved candidate solutions;
        end
    end
    Choose best iteration solution;
    if iteration best value < global best value then
        Memorized iteration best value as global best value;
    end
    Update circle radius;
end
End Algorithm
```

2.3. FNN Training Using HTVS

In this section, the basics of using the HTVS algorithm in FNN training are explained. In this study, optimal weights and biases were selected to improve the FNN's performance. The activation function and FNN structure remained constant. By means of the optimum values obtained, we ensured that the FNN reached the minimum error. In order to create the FNNHTVS structure discussed in this article, the fitness function and encoding strategy should be determined. A fitness function must be defined based on the mean square error (MSE) value in the FNN output.

The fitness function is produced as in [30]. For an FNN with a structure as in Figure 2, the fitness function is calculated by following the steps outlined below, where n is the number of inputs, h_n is the number of hidden layer nodes and o_n is equal to the output number. To calculate each hidden node,

$$f(x_l) = 1/\left(1 + exp\left(-\left(\sum_{k=1}^{n} w_{kl} \cdot x_l - b_l\right)\right)\right), \quad l = 1, 2, \ldots, h_n \tag{1}$$

where w_{kl} is the connection weight from input nodes to hidden layer nodes. b_l stands for the hidden layer node bias and x_l is the lth input for the network. After calculating the output of the hidden nodes, the output is evaluated as follows:

$$out_m = \sum_{i=1}^{h_n} w_{mi} \cdot f(h_i) - b_m, \quad m = 1, 2, \ldots, o_n \tag{2}$$

where w_{mi} is the weight value from the hidden layer nodes to the output layer nodes and b_m is used to express the output layer nodes' biases. MSE is determined as follows:

$$MSE = \frac{1}{n_s} \sum_{q=1}^{n_s} \left(\sum_{i=1}^{o_n} out_i^z - t_i^z \right)^2 \tag{3}$$

where t_i is equal to the real value, n_s is the number of samples used for training and z stands for an output node.

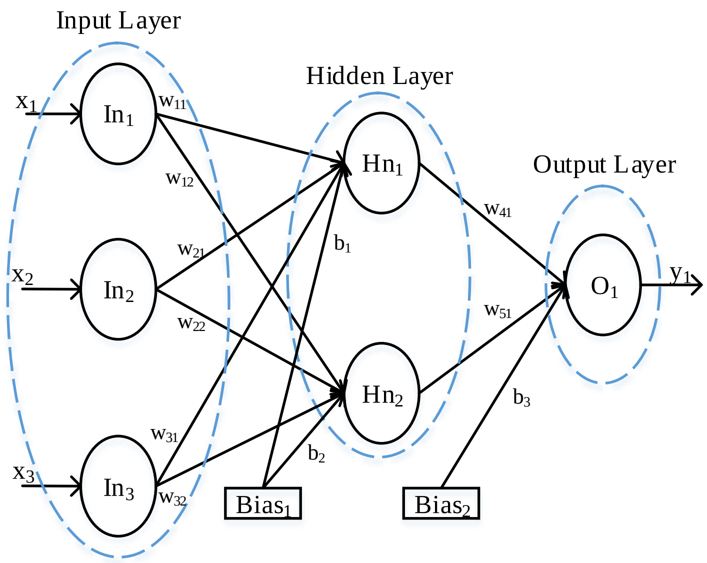

Figure 2. Candidate solutions of 3-2-1 FNN structure.

After the fitness function was created, the coding strategy was chosen. In this study, the matrix encoding strategy, used in studies related to FNN training, was chosen. The candidate solution matrix (CSM) consists of the combination of the four matrices described below. Figure 2 shows the weights and biases for an FNN with a 3-2-1 topology.

CSM is performed as follows:

$$CSM = \begin{bmatrix} weight_1 & weight_2' & bias_1 & bias_2 \end{bmatrix}$$

$$Weight_1 = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}, \quad Bias_1 = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$Weight_2' = \begin{bmatrix} w_{51} \\ w_{61} \end{bmatrix}, \quad Bias_2 = \begin{bmatrix} b_3 \end{bmatrix}$$

where $Weight_1$ is the weight matrix from the input layer to the hidden layer, $Bias_1$ is the hidden layer node bias matrix, $Weight_2'$ is the transpose weight matrix for from the hidden layer to the output layer and $Bias_2$ is the output layer node bias matrix.

After the fitness function and coding strategy are determined, a mesh structure suitable for the data set is determined. The steps of the FNNHTVS algorithm are followed in order to find the best values of the weights and biases. A flowchart diagram of the FNNHTVS algorithm is shown in Figure 3.

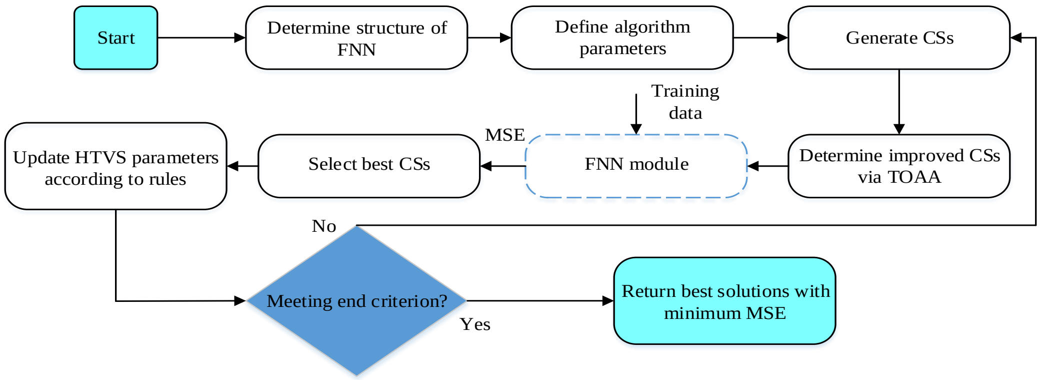

Figure 3. Flowchart of the FNNHTVS approach.

The FNN structure is determined for the classification dataset. HTVS parameters are defined, such as the maximum iteration number, the initial range of candidate solutions, the population size, etc. The dimensions of the problem are equal to the total number of weights and biases. To achieve the best values of weights and biases, candidate solutions are constructed depending on the OA and improved candidate solutions are determined. The feed-forward calculation is first applied for each sample in the dataset. Then, the errors, which are the difference between the calculated and desired values, are found. Finally, MSE is calculated. The best candidate solutions (CSs) are selected and parameters are updated according to rules. FNNHTVS continues until meeting the end criterion.

3. Validation of the FNNHTVS via Benchmark Datasets

In this section, the proposed FNNHTVS training algorithm is compared with the FNNVS, FNNPSO, FNNGSA and FNNPSOGSA algorithms. All algorithms are run on FNNs with the same structure. To analyze the performance of the FNNHTVS algorithm and to compare it with other algorithms, four frequently used classification problems were selected. These are iris classification, wine recognition, seed classification and the 3-bit parity problem. The first three of these were taken from the UCI machine learning repository of the University of California at Irvine [37]. The fourth problem is the 3-bit parity problem. The input and output values related to this problem are given in Table 2.

Table 2. 3-bit parity problem.

Input	000	001	010	011	100	101	110	111
Output	0	1	1	0	1	0	0	1

The problems chosen for comparison are classification problems that are frequently used in the literature [13–15]. The features and class numbers related to the problems are expressed in Table 3.

Table 3. Dataset information.

Problem	N. of Features	N. of Classes	N. of Samples
3-bit parity	3	2	8
Seeds	7	3	210
Iris	4	3	150
Wine	13	3	178

The parameters common to all algorithms were kept the same. For all algorithms, the population size and maximum iteration were set to 30 and 100, respectively. An initial range of candidate solutions of $[-50, 50]$ was preferred so that all the training algorithms could search within a wider space. Additionally, these algorithms contain user-controlled parameters. Table 4 presents these parameters.

Table 4. Special parameters for each algorithm.

Algorithm	Parameter	Value
HTVS	Level difference	0.8
PSO	C_1 and C_2 constants	2
	Inertia weights	[0.9, 0.5]
GSA	a	20
	Gravitational constant	1
	Initial acceleration and mass	0
PSOGSA	C_1' and C_2' constants	1
	Gravitational constant	1
	Inertia weights	[0.9, 0.5]

In this study, a network structure with 1 input, 1 hidden and 1 output (i-h-o) layer was selected. The Sigmoid function was determined for each node as the activation function. The algorithms were compared using benchmark datasets for 11 different numbers of hidden nodes. The algorithms were run until they reached the maximum number of iterations. Each algorithm was run 30 different times for each case. MSE was chosen as the comparison parameter and the mean, standard deviation (std. dev.) and best and worst values of the obtained data were recorded. These recorded statistical values provided information for the comparison. However, the Wilcoxon signed rank (WSR) pairwise comparison test was also applied to make a stronger comparison. The WSR test was used to determine which of the two comparing methods was superior. In this study, the statistical significance value was 0.05 for the WSR test. For each problem, the FNNHTVS algorithm was compared with other algorithms separately and measures of superiority, equality and loss were noted. Detailed information about the WSR test can be found in [38].

3.1. 3-Bit Parity Problem

The 3-bit parity problem is a frequently used nonlinear problem. It is an important problem used to measure the performance of training algorithms against nonlinear problems. In the three-input, single-output 3-bit parity problem, if the number of ones in the inputs is odd, the output is one; if even, the output is zero. The input and output sets of this problem are expressed in Table 2. H_n is the number of hidden nodes with H_n= 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 20, 30. For the 3-bit parity problem, a 3-H_n-1 FNN structure is used. This structure has a total of $(5H_n + 1)$ parameters, $4H_n$ weights and $H_n + 1$ biases, and the parameter range is taken as $[-50, 50]$. Algorithms were evaluated based on the mean, standard deviation and the best and worst value of MSE. The statistical results obtained after 30 independent runs are shown in Table A1 .

Looking at Table A1 from a general perspective, it can be observed that the FNNHTVS algorithm performed better than the other compared algorithms. The proposed algorithm for all hidden nodes obtained the best mean MSE values. This indicates that it effectively escaped the local minimum. We determined that the FNNGSA algorithm had the lowest standard deviations, except for hidden nodes 7, 15, 20 and 30. When the best and worst MSE values were examined, we found that the best values belonged to the FNNHTVS algorithm. The closest follower of the FNNHTVS algorithm was the FNNVS algorithm.

Additionally, the WSR test results are presented in Table 5. The Winner column in Table 5 shows in how many cases (11 different hidden nodes) the two compared algorithms outperformed each other. The column specified as Equal shows the number of cases where the algorithms could not outperform each other. As a result of the paired comparisons, the superiority of the FNNHTVS algorithm can be observed. Within the framework of the results, the effectiveness of the proposed training algorithm for this nonlinear problem has been shown.

Table 5. WSR test results for the 3-bit parity problem.

Method	Winner (FNNHTVS/Method 2)	Equal
FNNHTVS vs. FNNVS	11/0	0
FNNHTVS vs. FNNPSO	11/0	0
FNNHTVS vs. FNNPSOGSA	11/0	0
FNNHTVS vs. FNNGSA	11/0	0

3.2. Iris Classification Problem

The iris dataset is the best-known and most commonly used dataset in the pattern recognition literature [37]. The dataset consists of four inputs and three classes. The dataset contains a total of 150 samples, fifty for each class. The first class is classified as *Iris setosa*, the second class is *Iris Versicolor* and the third class is *Iris Virginica*.For the iris classification problem, a 4-H_n-3 FNN structure is used. This structure has a total of $8H_n + 3$ parameters, $7H_n$ weights and $H_n + 3$ biases, and the initial parameter range is taken as $[-50, 50]$. The statistical results obtained after 30 independent runs are presented in Table A2.

Based on the MSE results shown in Table A2, the FNNHTVS training algorithm displayed the best mean values for all cases except $H_n = 30$. For $H_n = 30$, the proposed training algorithm was ranked third.FNNHTVS had the smallest values for $H_n = 6, 9, 10, 12, 20, 30$. For other cases, it was most often ranked third. Its performance was competitive with that of the other compared algorithms in terms of its robust operation. In this problem, it was observed that the FNNVS algorithm exhibited the worst standard deviation value. In terms of the MSE values, FNNHTVS was ranked first in 7 of 11 FNN structures with $H_n = 7, 8, 9, 10, 12, 15, 30$.

The pairwise comparisons are presented in Table 6. As a result of comparing FNNHTVS and FNNVS, FNNHTVS won in nine cases and lost in one case. The lost H_n value was determined to be 30.The two compared algorithms were not able to outperform each other for $H_n = 4$. In addition, the FNNHTVS algorithm lost to the FNNPSO algorithm for $H_n = 30$.

Table 6. WSR test results for the iris classification problem.

Method	Winner (FNNHTVS/Method 2)	Equal
FNNHTVS vs. FNNVS	9/1	1
FNNHTVS vs. FNNPSO	10/1	0
FNNHTVS vs. FNNPSOGSA	11/0	0
FNNHTVS vs. FNNGSA	11/0	0

3.3. Wine Recognition Problem

These data are the result of a chemical analysis of wines grown in the same region in Italy and produced from three different types of grapes [37]. Within the scope of the analysis, the amounts of 13 components found in wine types were recorded. Therefore, the dataset consists of 13 features. Wine types are divided into three classes according to these inputs. The dataset contains 178 samples. In the wine recognition dataset, there are 59 data samples for the first class, 71 for the second class and 48 for the third class. For the wine recognition problem, a 13-H_n-3 FNN structure is used. This structure has a total of $17H_n + 3$ parameters, $16H_n$ weights and $H_n + 3$ biases, and the initial parameter range is taken as $[-50, 50]$. The statistical results obtained after 30 independent runs are presented in Table A3.

For all H_n values, the FNNHTVS training algorithm achieved the best statistical values and showed superior performance. The FNNVS training algorithm was also a follower of the proposed algorithm in terms of performance. The WSR test results presented in Table 7 support the claim that the proposed algorithm outperformed the other compared algorithms.

Table 7. WSR test results for the wine recognition problem.

Method	Winner (FNNHTVS/Method 2)	Equal
FNNHTVS vs. FNNVS	11/0	0
FNNHTVS vs. FNNPSO	11/0	0
FNNHTVS vs. FNNPSOGSA	11/0	0
FNNHTVS vs. FNNGSA	11/0	0

3.4. Seed Classification Problem

This dataset, which can be used in performance evaluations of classification and cluster analysis algorithms, includes the results of the classification of three different wheat seeds. The dataset consists of seven inputs and three classes [39]. The dataset contains 210 samples, 70 for each class. The first class is classified as Kama, the second class is Rosa and the third class is Canadian. For this problem, a 7-H_n-3 FNNstructure is used. This structure has a total of $11H_n + 3$ parameters, $10H_n$ weights and $H_n + 3$ biases, and the initial parameter range is taken as $[-50, 50]$. The statistical results obtained after 30 independent runs are demonstrated in Table A4.

In terms of all statistical parameters shown in Table A4, the FNNHTVS training algorithm outperformed the other algorithms. In the WSR test, it outperformed all the compared algorithms. The WSR test results are presented in Table 8.

Table 8. WSR test results for the seed classification problem.

Method	Winner (FNNHTVS/Method 2)	Equal
FNNHTVS vs. FNNVS	11/0	0
FNNHTVS vs. FNNPSO	11/0	0
FNNHTVS vs. FNNPSOGSA	11/0	0
FNNHTVS vs. FNNGSA	11/0	0

4. Performance Evaluation in Fault Classification

Short circuit fault classification is one of the important issues that are studied in order to more accurately intervene in response to faults occurring in transmission lines. Furthermore,some fault location algorithms need to know the fault class. This situation increases the importance of fault classification. For fault classification, various classification properties are obtained at first. Then, using these features and different artificial intelligence techniques, fault types are classified.

In this section of our study, short-circuit faults occurring on a 735 kV, 60 Hz, 100 km transmission line was modeled as frequency-dependent with the help of Matlab/Simulink. Classification data were produced by introducing short circuit faults into the model, which is shown in Figure 4. Classification was carried out with the FNNHTVS algorithm, the validity of which has been shown in the previous section. The performance of the FNNHTVS algorithm in fault classification was compared not only with the FNNVS, FNNPSO, FNNPSOGSA and FNNGSA algorithms, but also with other classifiers (SVM, KNN, FNN with LM and NB).

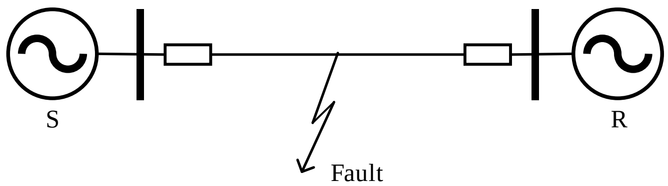

Figure 4. 735-kV, 60-Hz, 100-km transmission system model.

The selected classification features need to be specific and consistent for each fault type. In this study, post-fault one-cycle line currents and the zero sequence component of the line currents were taken as the input data. In each fault condition, the three-phase currents and the zero component were reduced by means of a certain method. In this reduction method, the highest peak value of the three phase currents was found in any fault, then each line current and zero component were divided by this peak value and the signals were scaled. The transmission line model studied here was a frequency-dependent model. Three-phase current signals and the zero sequence component for one cycle post-fault were sampled with a sampling frequency of 20 kHz and recorded. Measurements were made from the sending side of the transmission line. The root mean square (RMS) values of these recorded signals were calculated. The dataset was created using different fault resistance values, fault locations, fault types and fault inception angles. Single line to ground (SLG), line to line (LL), line to line to ground (LLG) and three-phase symmetric ground (LLLG) faults were generated in each phase.A random fault resistance value was chosen between 0.1 and 150 ohms. The fault inception angles (FIA) were determined as 0, 30, 45, 90 150 or 270. The fault location was chosen as 10, 20, 30, 50, 60, 80 or 90 km. A total of 250 data were created, 175 of which were training data and 75 were test data. The proposed algorithm and all other algorithms for comparison were run 30 different times. In each independent run, training and test samples were randomly selected from the created dataset.

Based on the formula $H_n = 2I_n + 1$ presented in [26,31], $H_n = 9$ was used. I_n is the input number. For the fault classification problem, an 4-9-4 FNN structure is used. This structure has a total of 85 parameters, 72 weights and 13 biases, and the parameter range is taken as $[-50, 50]$. The maximum iteration number was equal to 100 for all algorithms, which were evaluated based on the mean, standard deviation and best and worst MSE and accuracy values. The statistical results obtained after 30 independent runs are shown in Table 9. When Table 9 is examined, it can be observed that the FNNHTVS algorithm had a lower mean MSE and higher mean classification accuracy, compared to the other methods. A box plot graph is shown in Figure 5 and a convergence curve is depicted in Figure 6. It can be observed that the FNNHTVS algorithm had a low standard deviation and reached a lower mean MSE value in fewer iterations. The convergence curve was obtained by taking the average of 30 different runs. The FNNHTVS algorithm was compared with the methods of SVM, KNN, FNN with LM and NB. When the results shown in Table 10 are examined, it can be seen that the proposed algorithm exhibited a very competitive structure in relation to the other classifiers.

Table 9. Statistical fault classification results for $H_n = 9$.

Algorithms	MSE					Accuracy (%)				
	Mean	Median	Std. Dev.	Best	Worst	Mean	Median	Std. Dev.	Best	Worst
FNNHTVS	0.00944	0.01136	0.00590	6.89125×10^{-32}	0.01714	99.1111	98.6666	0.99380	100	96
FNNVS	0.01489	0.01413	0.01055	1.40023×10^{-17}	0.04374	98.3555	98.6666	1.31918	100	94.6666
FNNPSO	0.04263	0.03521	0.02908	0.00227	0.12496	97.9555	98.6666	2.49997	100	88
FNNPSOGSA	0.05369	0.04075	0.03241	0.01066	0.12397	97.0666	98.6666	3.21846	100	85.3333
FNNGSA	0.23634	0.24117	0.03002	0.16531	0.28993	75.0285	74.8571	10.6037	94.8571	49.7142

Table 10. Statistical results showing the accuracy of FNNHTVS and other classifiers.

Algorithms	Mean	Median	Std. Dev.	Best	Worst
FNN (LM)	97.9111	98.6666	5.24012	100	70.6666
KNN	98.4762	98.6666	1.25486	100	94.6666
SVM	98.8571	98.6666	1.02151	100	97.3333
Naive Bayes	98.8804	98.6666	0.88012	100	97.3333
FNNHTVS	99.1111	98.6666	0.99380	100	96

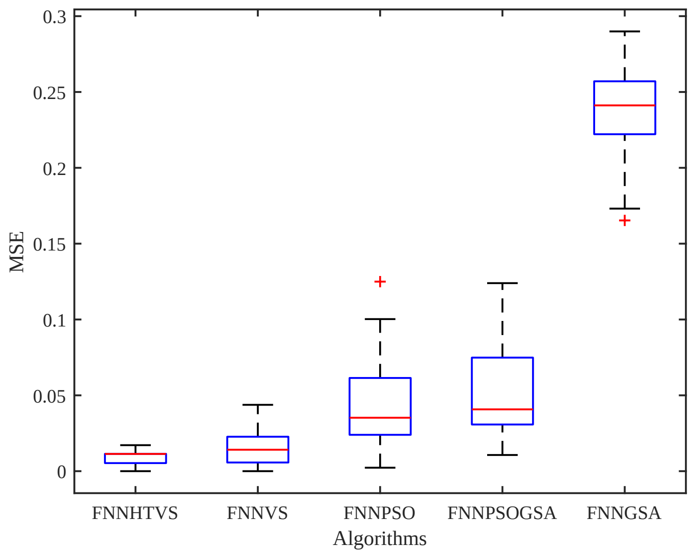

Figure 5. Box plot chart for fault classification.

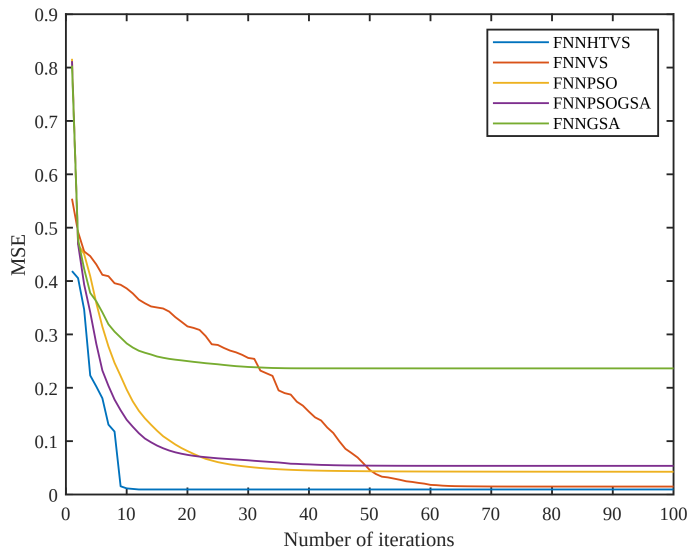

Figure 6. Convergence curve for fault classification.

The accuracy values obtained in the fault classification studies may vary depending on the sample number, data type and the transmission line model studied. Therefore, it would be more accurate to compare the FNNHTVS algorithm with the classifiers and algorithms used in this study. The comparison of the FNNHTVS training algorithm with the studies related to fault classification in the literature could create a misleading impression due to the differences in the datasets studied. Considering this situation, some studies in the literature are presented in Table 11, along with their important features. In [40], the discrete wavelet (DW)-based SVM method was used. The average accuracy rate was approximately the same as for FNNHTVS. In [41], fault classification and fault location tasks were undertaken using the multiclass SVM (MCSVM) method. In [42], it was observed that the classification accuracy decreased as the fault resistance increased. In a study using the Poincare-based correlation (PbC) method, the authors stated that higher classification rates were obtained for fault resistances up to 100 and 120 ohms. Based on the results shown in Table 11, we concluded that the FNNHTVS algorithm, with a mean accuracy rate of 99.1111%, obtained successful results that are compatible with those presented in the literature.

Table 11. Comparative assessment.

	Malathi, V. et al., 2010 [40]	Ekici, S., 2012 [41]	Mukherjee, A. et al., 2022 [42]	FNNHTVS
Line length (km)	225	360	150	100
Frequency (Hz)	50	-	-	60
Voltage (kV)	240	380	270	735
Method	DW-SVM	MCSVM	PbC	FNNHTVS
Fault resistance (ohm)	1–200	10–1000	0–150	0–150
FIA	36–126°	-	-	0–270°
Predicted class	4	4	4	4
Class. accuracy (%)	99.11	99	98.143	99.1111

5. Conclusions

Many heuristic optimization algorithms have been used in the training of ANNs to determine the optimal values of weights and biases due to factors such as the non-linearity of problem types and their very large dimensions. In this study, the usability of the HTVS algorithm, which has not been used for this purpose in the literature before, was examined in the training of FNNs. The HTVS algorithm was used to train FNNs and its performance was analyzed. It was compared with other methods on test problems and a short circuit fault classification problem in a transmission line. In order to compare the training performance of the algorithms, all samples of the datasets in Section 3 were used as training data and the MSE of training error was calculated. These problems were used to demonstrate the validity of the FNNHTVS algorithm. All algorithms were run 30 different times for each problem. The FNN training process was stopped when the maximum number of iterations was reached. For each optimization algorithm, the maximum number of iterations was 100, the population size was 30, and the initial candidate solution interval was $[-50, 50]$. As shown in Section 4, 70% of the data were used as a training set and the remaining 30% were used as a test set in the fault classification problem. The performance of the FNNHTVS algorithm was also compared with that of the SVM, KNN, FNN with LM and NB classifiers in the task of fault classification. Performance evaluations of the algorithms were undertaken by listing the results obtained from all stages separately. Based on the obtained results, we concluded that the HTVS algorithm is a viable approach in the training of FNNs for classification purposes.

The following ideas can be explored in future works:

- It may be interesting to detect fault locations using FNNHTVS;
- The HTVS algorithm could be used to train other types of ANNs; and
- The optimal structure of FNNs could be determined using HTVS, including the number of nodes and the number of hidden layers.

Author Contributions: Conceptualization, M.C.; methodology, M.C.; software, M.C.; validation, M.C.; formal analysis, M.C. and S.S.T.; investigation, M.C.; data curation, M.C.; writing—original draft preparation, M.C.; writing—review and editing, M.C. and S.S.T.; visualization, M.C.; supervision, S.S.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors wish to express their appreciation to the reviewers for their helpful suggestions, which greatly improved the presentation of this paper.

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

Abbreviations

The following abbreviations are used in this study:

FNN	Feed-Forward Neural Network
HTVS	Hybrid Taguchi Vortex Search

VS	Vortex Search
PSO	Particle Swarm Optimization
GSA	Gravitational Search Algorithm
PSOGSA	Particle Swarm Optimization Gravitational Search Algorithm
UCI	Machine Learning Repository of the University of California at Irvine
FNNHTVS	Hybrid Taguchi Vortex Search-based Feed-forward Neural Network
ANN	Artificial Neural Network
BP	Back-Propagation
GD	Gradient Descent
CG	Conjugate Gradient
LM	Levenberg–Marquardt
FNNPSO	Particle Swarm Optimization-Based Feed-Forward Neural Network
FNNPSOGSA	PSOGSA-based Feed-Forward Neural Network
FNNGSA	Gravitational Search Algorithm-Based Feed-Forward Neural Network
FNNVS	Vortex Search-Based Feed-Forward Neural Network
ABC	Artificial Bee Colony
GOA	Grasshopper Optimization Algorithm
MLP	Multilayer Perceptron
GOAMLP	Grasshopper Optimization Algorithm-Based Multilayer Perceptron
DA	Dragonfly Algorithm
WOA	Whale Optimization Algorithm
KHA	Krill-Herd Algorithm
CS	Cuckoo Search
SOS	Symbiotic Organism Search
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
NB	Naive Bayes
Max. Iter.	Maximum Iteration
TOAA	Taguchi Orthogonal Array Approach
OAs	Orthogonal Arrays
MSE	Mean Square Error
CSM	Candidate Solution Matrix
CSs	Candidate Solutions
Std. Dev.	Standart Deviation
WSR	Wilcoxon Signed Rank
RMS	Root Mean Square
SLG	Single Line to Ground
LL	Line to Line
LLG	Line to Line to Ground
LLLG	Three phase symmetric ground
DW	Discrete Wavelet
MCSVM	Multiclass Support Vector Machine
PbC	Poincare-Based Correlation

Appendix A. Statistical Results

Table A1. Statistical results (MSE) for the 3-bit parity problem.

Hidden Nodes	Parameters	FNNHTVS	FNNVS	FNNPSO	FNNPSOGSA	FNNGSA
4	mean	2.46929×10^{-2}	1.74843×10^{-1}	2.82043×10^{-1}	2.73659×10^{-1}	4.82205×10^{-1}
	std. dev.	4.74911×10^{-2}	1.51402×10^{-1}	9.26122×10^{-2}	8.93695×10^{-2}	1.73568×10^{-2}
	best	7.55448×10^{-21}	1.39264×10^{-14}	1.12847×10^{-1}	1.24014×10^{-1}	4.24011×10^{-1}
	worst	1.60879×10^{-1}	4.58333×10^{-1}	4.18578×10^{-1}	4.57686×10^{-1}	5.04945×10^{-1}
5	mean	9.63464×10^{-3}	1.26501×10^{-1}	2.75948×10^{-1}	2.38619×10^{-1}	4.82409×10^{-1}
	std. dev.	3.28141×10^{-2}	1.30249×10^{-1}	7.29964×10^{-2}	7.01424×10^{-2}	1.95679×10^{-2}
	best	6.62840×10^{-39}	5.06917×10^{-21}	9.35993×10^{-2}	1.27838×10^{-1}	3.94271×10^{-1}
	worst	1.74005×10^{-1}	3.75015×10^{-1}	4.18856×10^{-1}	4.29496×10^{-1}	5.01850×10^{-1}

Table A1. *Cont.*

Hidden Nodes	Parameters	FNNHTVS	FNNVS	FNNPSO	FNNPSOGSA	FNNGSA
6	mean	1.07093×10^{-2}	1.63833×10^{-1}	2.53330×10^{-1}	2.35092×10^{-1}	4.78239×10^{-1}
	std. dev.	2.93566×10^{-2}	1.46849×10^{-1}	9.22590×10^{-2}	8.21892×10^{-2}	2.08078×10^{-2}
	best	8.82973×10^{-27}	5.81165×10^{-16}	5.41118×10^{-2}	1.12504×10^{-1}	4.12376×10^{-1}
	worst	1.25001×10^{-1}	4.16674×10^{-1}	3.85710×10^{-1}	4.10193×10^{-1}	5.02324×10^{-1}
7	mean	3.89732×10^{-4}	1.52462×10^{-1}	2.59321×10^{-1}	2.13170×10^{-1}	4.73091×10^{-1}
	std. dev.	1.94526×10^{-3}	1.48291×10^{-1}	7.65263×10^{-2}	8.89791×10^{-2}	2.78423×10^{-2}
	best	3.08697×10^{-43}	7.01922×10^{-24}	1.30899×10^{-2}	6.36175×10^{-2}	4.03514×10^{-1}
	worst	1.06678×10^{-2}	5.00000×10^{-1}	4.33768×10^{-1}	4.52949×10^{-1}	5.01753×10^{-1}
8	mean	1.55137×10^{-2}	1.03572×10^{-1}	2.76456×10^{-1}	2.16024×10^{-1}	4.73142×10^{-1}
	std. dev.	4.06209×10^{-2}	1.16576×10^{-1}	1.03673×10^{-1}	8.54150×10^{-2}	1.92504×10^{-2}
	best	1.62182×10^{-39}	3.26241×10^{-31}	3.50026×10^{-2}	7.82187×10^{-2}	4.30022×10^{-1}
	worst	1.25328×10^{-1}	3.75000×10^{-1}	4.20465×10^{-1}	4.10480×10^{-1}	5.08010×10^{-1}
9	mean	8.33333×10^{-3}	1.41648×10^{-1}	2.87803×10^{-1}	2.14414×10^{-1}	4.62800×10^{-1}
	std. dev.	3.17135×10^{-2}	1.26004×10^{-1}	8.33543×10^{-2}	7.56429×10^{-2}	3.88811×10^{-2}
	best	3.14792×10^{-40}	4.13176×10^{-19}	1.44202×10^{-1}	7.93862×10^{-2}	3.85171×10^{-1}
	worst	1.25000×10^{-1}	3.75000×10^{-1}	4.33781×10^{-1}	3.66044×10^{-1}	5.15740×10^{-1}
10	mean	1.66667×10^{-2}	1.63635×10^{-1}	2.59155×10^{-1}	2.24921×10^{-1}	4.49624×10^{-1}
	std. dev.	4.32182×10^{-2}	1.04256×10^{-1}	9.58570×10^{-2}	6.07222×10^{-2}	3.93542×10^{-2}
	best	4.47502×10^{-48}	2.34936×10^{-16}	5.34658×10^{-2}	7.80592×10^{-2}	3.52046×10^{-1}
	worst	1.25000×10^{-1}	3.75000×10^{-1}	4.06478×10^{-1}	3.48740×10^{-1}	5.09670×10^{-1}
12	mean	1.71400×10^{-2}	1.00002×10^{-1}	2.37016×10^{-1}	2.02146×10^{-1}	4.58024×10^{-1}
	std. dev.	4.31070×10^{-2}	1.10837×10^{-1}	8.66082×10^{-2}	8.98497×10^{-2}	3.15063×10^{-2}
	best	1.19825×10^{-62}	6.10052×10^{-25}	2.31919×10^{-2}	7.19333×10^{-2}	3.84548×10^{-1}
	worst	1.25000×10^{-1}	3.75000×10^{-1}	4.03867×10^{-1}	3.95908×10^{-1}	5.13763×10^{-1}
15	mean	8.33333×10^{-3}	1.37322×10^{-1}	2.75541×10^{-1}	1.63485×10^{-1}	4.39196×10^{-1}
	std. dev.	3.17135×10^{-2}	1.14827×10^{-1}	8.07514×10^{-2}	7.68703×10^{-2}	4.77070×10^{-2}
	best	1.89087×10^{-75}	9.42598×10^{-24}	7.19125×10^{-2}	2.77272×10^{-2}	3.30466×10^{-1}
	worst	1.25000×10^{-1}	3.75000×10^{-1}	4.42468×10^{-1}	3.38089×10^{-1}	5.07388×10^{-1}
20	mean	4.16667×10^{-3}	1.70834×10^{-1}	2.34259×10^{-1}	2.49060×10^{-1}	4.93816×10^{-1}
	std. dev.	2.28218×10^{-2}	8.97992×10^{-2}	8.78904×10^{-2}	1.58705×10^{-1}	1.58475×10^{-1}
	best	9.27099×10^{-100}	3.63070×10^{-29}	5.42893×10^{-2}	6.75910×10^{-4}	1.09362×10^{-1}
	worst	1.25000×10^{-1}	2.50000×10^{-1}	4.37647×10^{-1}	5.03318×10^{-1}	8.68272×10^{-1}
30	mean	8.41417×10^{-73}	1.91668×10^{-1}	2.69017×10^{-1}	3.28764×10^{-1}	5.57519×10^{-1}
	std. dev.	4.60821×10^{-72}	1.34282×10^{-1}	8.80375×10^{-2}	1.54711×10^{-1}	8.00067×10^{-2}
	best	1.69134×10^{-166}	8.16898×10^{-38}	4.92869×10^{-2}	7.18583×10^{-5}	3.75444×10^{-1}
	worst	2.52403×10^{-71}	5.00000×10^{-1}	4.18743×10^{-1}	6.25000×10^{-1}	6.93499×10^{-1}

Table A2. Statistical results (MSE) in the iris classification problem.

Hidden Nodes	Parameters	FNNHTVS	FNNVS	FNNPSO	FNNPSOGSA	FNNGSA
4	mean	1.14862×10^{-1}	1.68751×10^{-1}	2.01325×10^{-1}	2.15957×10^{-1}	4.64779×10^{-1}
	std. dev.	1.06685×10^{-1}	1.50803×10^{-1}	5.22724×10^{-2}	3.53812×10^{-2}	5.46828×10^{-2}
	best	2.99539×10^{-2}	1.33366×10^{-2}	8.45694×10^{-2}	1.46577×10^{-1}	3.85026×10^{-1}
	worst	3.65792×10^{-1}	3.86443×10^{-1}	3.10293×10^{-1}	2.79219×10^{-1}	6.06090×10^{-1}
5	mean	6.34653×10^{-2}	1.62361×10^{-1}	1.84219×10^{-1}	1.92374×10^{-1}	4.34723×10^{-1}
	std. dev.	4.37378×10^{-2}	1.43383×10^{-1}	5.23417×10^{-2}	4.11192×10^{-2}	3.80232×10^{-2}
	best	2.64633×10^{-2}	1.86090×10^{-2}	8.53060×10^{-2}	1.04880×10^{-1}	3.79576×10^{-1}
	worst	2.45174×10^{-1}	3.66705×10^{-1}	2.72920×10^{-1}	2.80629×10^{-1}	5.14518×10^{-1}
6	mean	4.68511×10^{-2}	1.36568×10^{-1}	1.81852×10^{-1}	1.84580×10^{-1}	4.27138×10^{-1}
	std. dev.	1.91529×10^{-2}	1.41366×10^{-1}	4.42480×10^{-2}	4.06917×10^{-2}	5.56950×10^{-2}
	best	2.63022×10^{-2}	1.35111×10^{-2}	9.05435×10^{-2}	9.49799×10^{-2}	3.05662×10^{-1}
	worst	1.20172×10^{-1}	4.73654×10^{-1}	2.70677×10^{-1}	2.52811×10^{-1}	5.67574×10^{-1}

Table A2. *Cont.*

Hidden Nodes	Parameters	FNNHTVS	FNNVS	FNNPSO	FNNPSOGSA	FNNGSA
7	mean	5.64944×10^{-2}	1.16451×10^{-1}	1.92708×10^{-1}	1.81135×10^{-1}	4.10110×10^{-1}
	std. dev.	6.35361×10^{-2}	1.27068×10^{-1}	5.36873×10^{-2}	4.98007×10^{-2}	6.73658×10^{-2}
	best	1.33333×10^{-2}	1.62844×10^{-2}	1.15527×10^{-1}	9.28957×10^{-2}	2.99464×10^{-1}
	worst	3.46725×10^{-1}	3.56866×10^{-1}	3.25583×10^{-1}	2.90122×10^{-1}	5.91297×10^{-1}
8	mean	7.25993×10^{-2}	1.13436×10^{-1}	1.60370×10^{-1}	1.60365×10^{-1}	4.09623×10^{-1}
	std. dev.	6.10813×10^{-2}	1.09866×10^{-1}	5.21421×10^{-2}	3.76046×10^{-2}	5.45519×10^{-2}
	best	2.15981×10^{-2}	2.29368×10^{-2}	6.82389×10^{-2}	9.08569×10^{-2}	3.31659×10^{-1}
	worst	2.50296×10^{-1}	3.58895×10^{-1}	2.64006×10^{-1}	2.29685×10^{-1}	5.58969×10^{-1}
9	mean	3.94528×10^{-2}	1.53900×10^{-1}	1.48810×10^{-1}	1.75455×10^{-1}	3.71082×10^{-1}
	std. dev.	1.30927×10^{-2}	1.30269×10^{-1}	3.95891×10^{-2}	4.77214×10^{-2}	4.59631×10^{-2}
	best	1.98988×10^{-2}	2.62890×10^{-2}	7.89208×10^{-2}	1.02002×10^{-1}	2.87191×10^{-1}
	worst	7.31698×10^{-2}	3.73252×10^{-1}	2.53782×10^{-1}	2.80107×10^{-1}	4.96640×10^{-1}
10	mean	3.51741×10^{-2}	1.34166×10^{-1}	1.52510×10^{-1}	1.75090×10^{-1}	3.96100×10^{-1}
	std. dev.	1.04808×10^{-2}	1.20707×10^{-1}	4.31790×10^{-2}	3.58333×10^{-2}	5.75770×10^{-2}
	best	1.33428×10^{-2}	2.55888×10^{-2}	7.93357×10^{-2}	9.93091×10^{-2}	2.97023×10^{-1}
	worst	6.02462×10^{-2}	3.67249×10^{-1}	2.64466×10^{-1}	2.39029×10^{-1}	4.91150×10^{-1}
12	mean	3.74717×10^{-2}	1.35911×10^{-1}	1.47051×10^{-1}	1.93487×10^{-1}	3.85262×10^{-1}
	std. dev.	1.29253×10^{-2}	1.18291×10^{-1}	3.77622×10^{-2}	1.31807×10^{-1}	6.21025×10^{-2}
	best	1.33333×10^{-2}	2.05661×10^{-2}	6.78784×10^{-2}	9.12593×10^{-2}	3.05040×10^{-1}
	worst	8.38928×10^{-2}	3.53335×10^{-1}	2.06790×10^{-1}	8.24107×10^{-1}	5.65557×10^{-1}
15	mean	6.38422×10^{-2}	1.08019×10^{-1}	1.48758×10^{-1}	2.39809×10^{-1}	4.09646×10^{-1}
	std. dev.	7.72433×10^{-2}	1.14276×10^{-1}	3.74622×10^{-2}	1.95200×10^{-1}	1.21881×10^{-1}
	best	1.29753×10^{-2}	2.38645×10^{-2}	7.14960×10^{-2}	9.19578×10^{-2}	2.74557×10^{-1}
	worst	3.40175×10^{-1}	3.80079×10^{-1}	1.99074×10^{-1}	8.06625×10^{-1}	7.41139×10^{-1}
20	mean	7.53001×10^{-2}	1.25018×10^{-1}	1.25229×10^{-1}	3.11545×10^{-1}	6.59167×10^{-1}
	std. dev.	9.31963×10^{-2}	1.21136×10^{-1}	3.22619×10^{-2}	2.06710×10^{-1}	2.66077×10^{-1}
	best	2.46229×10^{-2}	1.34936×10^{-2}	5.50465×10^{-2}	8.49606×10^{-2}	2.01427×10^{-1}
	worst	3.47971×10^{-1}	3.60454×10^{-1}	1.80171×10^{-1}	8.14597×10^{-1}	6.21821×10^{-1}
30	mean	1.99517×10^{-1}	1.07584×10^{-1}	1.19432×10^{-1}	3.16707×10^{-1}	8.10926×10^{-1}
	std. dev.	1.42793×10^{-1}	9.60862×10^{-2}	3.23971×10^{-2}	1.66360×10^{-1}	2.33804×10^{-1}
	best	2.00044×10^{-2}	2.13355×10^{-2}	7.04566×10^{-2}	5.27357×10^{-2}	4.77002×10^{-1}
	worst	3.45483×10^{-1}	3.79904×10^{-1}	1.73480×10^{-1}	7.35598×10^{-1}	6.53800×10^{-1}

Table A3. Statistical results (MSE) in the wine recognition problem.

Hidden Nodes	Parameters	FNNHTVS	FNNVS	FNNPSO	FNNPSOGSA	FNNGSA
4	mean	1.94378×10^{-2}	1.33566×10^{-1}	1.69763×10^{-1}	1.65540×10^{-1}	4.61577×10^{-1}
	std. dev.	1.28846×10^{-2}	1.03747×10^{-1}	6.20304×10^{-2}	4.76383×10^{-2}	7.87211×10^{-2}
	best	2.73226×10^{-5}	1.67689×10^{-2}	8.40273×10^{-2}	8.71521×10^{-2}	2.62065×10^{-1}
	worst	5.78721×10^{-2}	4.12618×10^{-1}	3.45308×10^{-1}	2.74366×10^{-1}	5.76831×10^{-1}
5	mean	1.18868×10^{-2}	1.51479×10^{-1}	1.49253×10^{-1}	1.51786×10^{-1}	4.30638×10^{-1}
	std. dev.	6.50078×10^{-3}	1.13153×10^{-1}	3.75319×10^{-2}	4.80765×10^{-2}	4.73713×10^{-2}
	best	4.08510×10^{-9}	2.13987×10^{-2}	6.83132×10^{-2}	6.21954×10^{-2}	3.49284×10^{-1}
	worst	2.80562×10^{-2}	4.49282×10^{-1}	2.24855×10^{-1}	2.95900×10^{-1}	5.23850×10^{-1}
6	mean	1.46168×10^{-2}	1.58700×10^{-1}	1.30736×10^{-1}	1.36542×10^{-1}	4.10015×10^{-1}
	std. dev.	8.51222×10^{-3}	1.21009×10^{-1}	4.24100×10^{-2}	3.87619×10^{-2}	6.41373×10^{-2}
	best	1.58983×10^{-8}	8.33856×10^{-3}	4.28716×10^{-2}	6.82425×10^{-2}	2.59599×10^{-1}
	worst	2.80899×10^{-2}	4.71227×10^{-1}	2.29977×10^{-1}	2.34809×10^{-1}	5.50772×10^{-1}
7	mean	1.00025×10^{-2}	1.33992×10^{-1}	1.30839×10^{-1}	1.38147×10^{-1}	4.19058×10^{-1}
	std. dev.	7.39580×10^{-3}	1.12228×10^{-1}	4.53306×10^{-2}	4.37701×10^{-2}	5.85164×10^{-2}
	best	1.66979×10^{-11}	1.72366×10^{-2}	6.74613×10^{-2}	7.40539×10^{-2}	2.59164×10^{-1}
	worst	2.24719×10^{-2}	4.18049×10^{-1}	2.24881×10^{-1}	2.73788×10^{-1}	5.32682×10^{-1}

Table A3. *Cont.*

Hidden Nodes	Parameters	FNNHTVS	FNNVS	FNNPSO	FNNPSOGSA	FNNGSA
8	mean	4.20276×10^{-3}	1.96334×10^{-1}	1.22014×10^{-1}	1.40364×10^{-1}	4.11930×10^{-1}
	std. dev.	4.49867×10^{-3}	1.15859×10^{-1}	3.97087×10^{-2}	5.15103×10^{-2}	5.71548×10^{-2}
	best	4.04669×10^{-14}	6.40472×10^{-2}	7.04695×10^{-2}	7.81411×10^{-2}	3.08821×10^{-1}
	worst	1.21787×10^{-2}	4.66286×10^{-1}	2.49579×10^{-1}	2.96196×10^{-1}	5.38821×10^{-1}
9	mean	2.99296×10^{-3}	1.63649×10^{-1}	1.12534×10^{-1}	1.16186×10^{-1}	3.79386×10^{-1}
	std. dev.	4.25057×10^{-3}	1.20513×10^{-1}	3.08861×10^{-2}	3.59304×10^{-2}	6.44611×10^{-2}
	best	2.47483×10^{-13}	3.36361×10^{-2}	4.83561×10^{-2}	4.40372×10^{-2}	2.34178×10^{-1}
	worst	1.12372×10^{-2}	5.22962×10^{-1}	1.71889×10^{-1}	2.15566×10^{-1}	4.71524×10^{-1}
10	mean	2.25964×10^{-3}	1.83726×10^{-1}	1.03152×10^{-1}	1.26672×10^{-1}	3.92796×10^{-1}
	std. dev.	3.55391×10^{-3}	1.24834×10^{-1}	4.53593×10^{-2}	3.83574×10^{-2}	6.46792×10^{-2}
	best	4.29008×10^{-16}	4.16381×10^{-2}	5.09519×10^{-2}	6.44176×10^{-2}	2.62258×10^{-1}
	worst	1.12508×10^{-2}	4.76774×10^{-1}	2.06652×10^{-1}	2.32131×10^{-1}	5.06974×10^{-1}
12	mean	2.13262×10^{-3}	1.95910×10^{-1}	1.04961×10^{-1}	1.40683×10^{-1}	3.70757×10^{-1}
	std. dev.	3.30063×10^{-3}	1.16776×10^{-1}	2.62366×10^{-2}	5.58901×10^{-2}	5.98702×10^{-2}
	best	8.17277×10^{-16}	4.49510×10^{-2}	5.94720×10^{-2}	6.41966×10^{-2}	2.25114×10^{-1}
	worst	1.12401×10^{-2}	4.44748×10^{-1}	1.52097×10^{-1}	3.04571×10^{-1}	4.95184×10^{-1}
15	mean	1.87675×10^{-3}	1.63760×10^{-1}	8.06230×10^{-2}	1.52749×10^{-1}	3.61846×10^{-1}
	std. dev.	3.19606×10^{-3}	1.02461×10^{-1}	1.85006×10^{-2}	9.47948×10^{-2}	8.58804×10^{-2}
	best	1.48717×10^{-16}	4.51008×10^{-2}	4.98801×10^{-2}	7.40337×10^{-2}	2.03846×10^{-1}
	worst	1.12360×10^{-2}	4.49571×10^{-1}	1.18488×10^{-1}	4.52816×10^{-1}	5.57647×10^{-1}
20	mean	3.76476×10^{-4}	1.59726×10^{-1}	7.97941×10^{-2}	1.42076×10^{-1}	3.76138×10^{-1}
	std. dev.	1.42621×10^{-3}	9.16628×10^{-2}	4.62824×10^{-2}	6.59667×10^{-2}	1.11050×10^{-1}
	best	2.01689×10^{-23}	5.06107×10^{-2}	4.01168×10^{-2}	8.13262×10^{-2}	2.30746×10^{-1}
	worst	5.62821×10^{-3}	4.21890×10^{-1}	2.99984×10^{-1}	3.89202×10^{-1}	7.29064×10^{-1}
30	mean	7.44123×10^{-22}	1.91386×10^{-1}	6.83553×10^{-2}	2.25535×10^{-1}	3.70193×10^{-1}
	std. dev.	3.29131×10^{-21}	9.77874×10^{-2}	5.69610×10^{-2}	1.94427×10^{-1}	1.79224×10^{-1}
	best	1.48816×10^{-56}	4.49687×10^{-2}	2.95764×10^{-2}	6.16741×10^{-2}	1.71175×10^{-1}
	worst	1.75528×10^{-20}	4.15806×10^{-1}	3.58487×10^{-1}	7.66503×10^{-1}	9.70961×10^{-1}

Table A4. Statistical results (MSE) in the seed classification problem.

Hidden Nodes	Parameters	FNNHTVS	FNNVS	FNNPSO	FNNPSOGSA	FNNGSA
4	mean	1.10847×10^{-1}	1.93308×10^{-1}	2.01770×10^{-1}	1.86016×10^{-1}	4.07302×10^{-1}
	std. dev.	1.60338×10^{-2}	1.18671×10^{-1}	4.48398×10^{-2}	3.99702×10^{-2}	3.94880×10^{-2}
	best	8.31405×10^{-2}	7.88864×10^{-2}	1.03638×10^{-1}	1.22263×10^{-1}	3.45558×10^{-1}
	worst	1.42066×10^{-1}	3.94932×10^{-1}	2.67137×10^{-1}	2.90377×10^{-1}	5.33628×10^{-1}
5	mean	1.01320×10^{-1}	2.07712×10^{-1}	1.81230×10^{-1}	1.73523×10^{-1}	4.09495×10^{-1}
	std. dev.	1.59207×10^{-2}	1.22636×10^{-1}	4.81278×10^{-2}	3.93328×10^{-2}	5.45169×10^{-2}
	best	7.20300×10^{-2}	5.82352×10^{-2}	1.14616×10^{-1}	1.09551×10^{-1}	3.07478×10^{-1}
	worst	1.49775×10^{-1}	3.99913×10^{-1}	3.01366×10^{-1}	2.72784×10^{-1}	5.48163×10^{-1}
6	mean	9.36586×10^{-2}	1.66201×10^{-1}	1.65856×10^{-1}	1.71439×10^{-1}	3.94956×10^{-1}
	std. dev.	1.08448×10^{-2}	1.01809×10^{-1}	3.00229×10^{-2}	5.10343×10^{-2}	4.75287×10^{-2}
	best	7.62807×10^{-2}	5.50233×10^{-2}	8.90293×10^{-2}	1.23650×10^{-1}	3.08159×10^{-1}
	worst	1.16848×10^{-1}	4.01519×10^{-1}	2.18201×10^{-1}	4.04750×10^{-1}	5.14334×10^{-1}
7	mean	8.91858×10^{-2}	1.67699×10^{-1}	1.55629×10^{-1}	1.67584×10^{-1}	3.79103×10^{-1}
	std. dev.	1.24263×10^{-2}	1.05127×10^{-1}	2.79420×10^{-2}	4.43436×10^{-2}	5.27818×10^{-2}
	best	6.09093×10^{-2}	5.65455×10^{-2}	9.05990×10^{-2}	1.14863×10^{-1}	2.94214×10^{-1}
	worst	1.22892×10^{-1}	3.96726×10^{-1}	1.96405×10^{-1}	3.50567×10^{-1}	5.37897×10^{-1}
8	mean	9.18808×10^{-2}	1.78536×10^{-1}	1.53213×10^{-1}	1.51797×10^{-1}	3.70108×10^{-1}
	std. dev.	1.26865×10^{-2}	1.14783×10^{-1}	2.21297×10^{-2}	2.61319×10^{-2}	4.10571×10^{-2}
	best	6.11826×10^{-2}	5.72383×10^{-2}	1.03633×10^{-1}	1.13652×10^{-1}	2.90596×10^{-1}
	worst	1.09795×10^{-1}	3.95296×10^{-1}	2.01519×10^{-1}	2.40753×10^{-1}	4.33540×10^{-1}

Table A4. *Cont.*

Hidden Nodes	Parameters	FNNHTVS	FNNVS	FNNPSO	FNNPSOGSA	FNNGSA
9	mean	8.58805×10^{-2}	1.79317×10^{-1}	1.56830×10^{-1}	1.60936×10^{-1}	3.53928×10^{-1}
	std. dev.	6.93764×10^{-3}	1.05511×10^{-1}	3.33739×10^{-2}	2.82649×10^{-2}	6.73966×10^{-2}
	best	7.43803×10^{-2}	7.50251×10^{-2}	1.02041×10^{-1}	1.14364×10^{-1}	1.70935×10^{-1}
	worst	1.00271×10^{-1}	4.48567×10^{-1}	2.21748×10^{-1}	2.38765×10^{-1}	4.84104×10^{-1}
10	mean	8.58320×10^{-2}	1.39362×10^{-1}	1.49952×10^{-1}	1.67026×10^{-1}	3.71755×10^{-1}
	std. dev.	1.13822×10^{-2}	6.70709×10^{-2}	2.61584×10^{-2}	4.86881×10^{-2}	6.20682×10^{-2}
	best	5.55225×10^{-2}	6.66799×10^{-2}	1.02490×10^{-1}	1.20878×10^{-1}	2.68974×10^{-1}
	worst	1.07147×10^{-1}	3.86976×10^{-1}	2.28392×10^{-1}	3.69949×10^{-1}	5.88624×10^{-1}
12	mean	7.69695×10^{-2}	1.72056×10^{-1}	1.31553×10^{-1}	1.77467×10^{-1}	3.52560×10^{-1}
	std. dev.	8.74892×10^{-3}	8.54826×10^{-2}	2.64553×10^{-2}	9.03774×10^{-2}	7.04322×10^{-2}
	best	6.23604×10^{-2}	7.98103×10^{-2}	9.64765×10^{-2}	1.03684×10^{-1}	2.45908×10^{-1}
	worst	9.13157×10^{-2}	3.90539×10^{-1}	2.11989×10^{-1}	4.09575×10^{-1}	5.68246×10^{-1}
15	mean	7.42578×10^{-2}	1.70866×10^{-1}	1.25953×10^{-1}	1.48903×10^{-1}	3.71133×10^{-1}
	std. dev.	8.84990×10^{-3}	9.90537×10^{-2}	2.27677×10^{-2}	5.71889×10^{-2}	8.20515×10^{-2}
	best	6.19053×10^{-2}	8.61301×10^{-2}	8.98466×10^{-2}	9.61550×10^{-2}	2.50839×10^{-1}
	worst	1.00014×10^{-1}	4.08016×10^{-1}	1.67739×10^{-1}	4.04558×10^{-1}	5.57981×10^{-1}
20	mean	7.64729×10^{-2}	1.66421×10^{-1}	1.16223×10^{-1}	2.21587×10^{-1}	4.08647×10^{-1}
	std. dev.	8.87865×10^{-3}	9.34156×10^{-2}	1.98504×10^{-2}	1.70788×10^{-1}	1.91840×10^{-1}
	best	5.76150×10^{-2}	6.33511×10^{-2}	7.22399×10^{-2}	1.01568×10^{-1}	2.04698×10^{-1}
	worst	9.27819×10^{-2}	4.85761×10^{-1}	1.83814×10^{-1}	7.32398×10^{-1}	9.24442×10^{-1}
30	mean	7.45181×10^{-2}	1.99154×10^{-1}	1.20852×10^{-1}	3.32331×10^{-1}	8.27154×10^{-1}
	std. dev.	8.67849×10^{-3}	1.01211×10^{-1}	5.55258×10^{-2}	2.83933×10^{-1}	2.21347×10^{-1}
	best	5.78553×10^{-2}	9.16919×10^{-2}	7.69777×10^{-2}	8.98902×10^{-2}	4.31377×10^{-1}
	worst	9.52382×10^{-2}	3.80953×10^{-1}	3.95592×10^{-1}	8.36375×10^{-1}	8.74650×10^{-1}

References

1. Faris, H.; Aljarah, I.; Mirjalili, S. Training feedforward neural networks using multi-verse optimizer for binary classification problems. *Appl. Intell.* **2016**, *45*, 322–332. [CrossRef]
2. Coban, M.; Tezcan, S.S. Detection and classification of short-circuit faults on a transmission line using current signal. *Bull. Pol. Acad. Sci. Tech. Sci.* **2021**, *69*, 1–9.
3. Almeida, A.R.; Almeida, O.M.; Junior, B.F.; Barreto, L.H.; Barros, A.K. ICA feature extraction for the location and classification of faults in high-voltage transmission lines. *Electr. Power Syst. Res.* **2017**, *148*, 254–263. [CrossRef]
4. Fernandez-Blanco, E.; Rivero, D.; Pazos, A. EEG signal processing with separable convolutional neural network for automatic scoring of sleeping stage. *Neurocomputing* **2020**, *410*, 220–228. [CrossRef]
5. Nardo, F.D.; Morbidoni, C.; Cucchiarelli, A.; Fioretti, S. Influence of EMG-signal processing and experimental set-up on prediction of gait events by neural network. *Biomed. Signal Process Control.* **2021**, *63*, 102232. [CrossRef]
6. Ravesh, N.R.; Ramezani, N.; Ahmadi, I.; Nouri, H. A hybrid artificial neural network and wavelet packet transform approach for fault location in hybrid transmission lines. *Electr. Power Syst. Res.* **2022**, *204*, 107721. [CrossRef]
7. Gashteroodkhani, O.A.; Majidi, M.; Etezadi-Amoli, M.; Nematollahi, A.F.; Vahidi, B. A hybrid SVM-TT transform-based method for fault location in hybrid transmission lines with underground cables. *Electr. Power Syst. Res.* **2019**, *170*, 205–214. [CrossRef]
8. Duarte Soares, L.; de Souza Queiroz, A.; López, G.P.; Carreño-Franco, E.M.; López-Lezama, J.M.; Muñoz-Galeano, N. BiGRU-CNN Neural Network Applied to Electric Energy Theft Detection. *Electronics* **2022**, *11*, 693. [CrossRef]
9. Arin, E.; Ozbayoglu, A.M. Deep learning based hybrid computational intelligence models for options pricing. *Comput. Econ.* **2020**, *59*, 39–58. [CrossRef]
10. Li, S.; Fan, Z. Evaluation of urban green space landscape planning scheme based on PSO-BP neural network model. *Alex. Eng. J.* **2022**, *61*, 7141–7153. [CrossRef]
11. Mao, X.; Song, S.; Ding, F. Optimal BP neural network algorithm for state of charge estimation of lithium-ion battery using PSO with Levy flight. *J. Energy Storage* **2022**, *49*, 104139. [CrossRef]
12. Singh, M.P.; Singh, G. Two phase learning technique in modular neural network for pattern classification of handwritten Hindi alphabets. *Mach. Learn. Appl.* **2021**, *6*, 100174. [CrossRef]
13. Sağ, T.; Jalil, Z.A.J. Vortex search optimization algorithm for training of feed-forward neural network. *Int. J. Mach. Learn.* **2021**, *12*, 1517–1544. [CrossRef]
14. Mirjalili, S.; Hashim, S.Z.M.; Sardroudi, H.M. Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Appl. Math. Comput* **2012**, *218*, 11125–11137. [CrossRef]

15. Pashaei, E.; Pashaei, E. Training feedforward neural network using enhanced black hole algorithm: A case study on COVID-19 related ACE2 Gene expression classification. *Arab. J. Sci. Eng .* **2021**, *46*, 3807–3828. [CrossRef] [PubMed]
16. Hassanpour, M.; Vaferi, B.; Masoumi, M.E. Estimation of pool boiling heat transfer coefficient of alumina water-based nanofluids by various artificial intelligence (AI) approaches. *Appl. Therm. Eng.* **2018**, *128*, 1208–1222. [CrossRef]
17. Yves Chauvin, D.E.R. *Backpropagation Theory, Architectures, and Applications*; Psychology Press: London, UK, 1995.
18. Robbins, H.; Monro, S. A stochastic approximation method. *Ann. Math. Stat.* **1951**, *22*, 400 – 407. [CrossRef]
19. van der Smagt, P.P. Minimisation methods for training feedforward neural networks. *Neural. Netw.* **1994**, *7*, 1–11. [CrossRef]
20. Hagan, M.T.; Menhaj, M.B. Training Feedforward networks with the Marquardt Algorithm. *IEEE Trans. Neural Netw.* **1994**, *5*, 989–993. [CrossRef]
21. Cortés-Caicedo, B.; Molina-Martin, F.; Grisales-Noreña, L.F.; Montoya, O.D.; Hernández, J.C. Optimal design of PV Systems in electrical distribution networks by minimizing the annual equivalent operative costs through the discrete-continuous vortex search algorithm. *Sensors* **2022**, *22*, 851. [CrossRef]
22. Martínez-Gil, J.F.; Moyano-García, N.A.; Montoya, O.D.; Alarcon-Villamil, J.A. Optimal Selection of conductors in three-phase distribution networks using a discrete version of the vortex search algorithm. *Computation* **2021**, *9*, 80. [CrossRef]
23. Ozturk, C.; Karaboga, D. Hybrid Artificial Bee Colony algorithm for neural network training. In Proceedings of the CEC 2011, New Orleans, LA, USA, 5–8 June 2011; pp. 84–88.
24. Heidari, A.A.; Faris, H.; Aljarah, I.; Mirjalili, S. An efficient hybrid multilayer perceptron neural network with grasshopper optimization. *Soft Comput.* **2019**, *23*, 7941–7958. [CrossRef]
25. Gülcü, Ş. Training of the feed forward artificial neural networks using dragonfly algorithm. *Appl. Soft Comput.* **2022**, *124*, 109023. [CrossRef]
26. Aljarah, I.; Faris, H.; Mirjalili, S. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Comput.* **2018**, *22*, 1–15. [CrossRef]
27. Lari, N.S.; Abadeh, M.S. Training artificial neural network by krill-herd algorithm. In Proceedings of the ITAIC 2014, Chongqing, China, 20–21 December 2014; pp. 63–67.
28. Yi, J.-h.; Xu, W.-h.; Chen, Y.-t. Novel Back Propagation Optimization by Cuckoo Search Algorithm. *Sci. World J.* **2014**, *2014*, 878262. [CrossRef]
29. Wu, H.; Zhou, Y.; Luo, Q.; Basset, M.A. Training feedforward neural networks using symbiotic organisms search algorithm. *Comput. Intell. Neurosci.* **2016**, *2016*, 9063065. [CrossRef]
30. Zhang, J.R.; Zhang, J.; Lok, T.M.; Lyu, M.R. A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. *Appl. Math. Comput.* **2007**, *185*, 1026–1037. [CrossRef]
31. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Let a biogeography-based optimizer train your Multi-Layer Perceptron. *Inf. Sci.* **2014**, *269*, 188–209. [CrossRef]
32. Saka, M.; Çoban, M.; Eke, I.; Tezcan, S.S.; Taplamacioğlu, M.C. A novel hybrid global optimization algorithm having training strategy: Hybrid Taguchi-vortex search algorithm. *Turk. J. Elec. Eng. Comp. Sci.* **2021**, *29*, 1908–1928. [CrossRef]
33. Dogan, B.; Ölmez, T. A new metaheuristic for numerical function optimization: Vortex Search algorithm. *Inf. Sci.* **2015**, *293*, 125–145. [CrossRef]
34. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
35. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
36. Lawrence, J. *Introduction to Neural Networks: Design, Theory, and Applications*, 5th ed.; California Scientific Software: New York, NY, USA, 1994.
37. Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available online: http://archive.ics.uci.edu/ml (accessed on 11 July 2022) .
38. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
39. Charytanowicz, M.; Niewczas, J.; Kulczycki, P.; Kowalski, P.A.; Łukasik, S.; Zak, S. Complete gradient clustering algorithm for features analysis of X-Ray images. *Adv. Intell. Syst. Comput.* **2010**, *69*, 15–24.
40. Malathi, V.; Marimuthu, N.S.; Baskar, S. Intelligent approaches using support vector machine and extreme learning machine for transmission line protection. *Neurocomputing* **2010**, *73*, 2160–2167. [CrossRef]
41. Ekici, S. Support Vector Machines for classification and locating faults on transmission lines. *Appl. Soft Comput.* **2012**, *12*, 1650–1658. [CrossRef]
42. Mukherjee, A.; Chatterjee, K.; Kundu, P.K.; Das, A. Application of Poincaré analogous time-split signal-based statistical correlation for transmission line fault classification. *Electr. Eng.* **2022**, *4*, 1057–1075. [CrossRef]