AI

MDPI

Article

Training Artificial Neural Networks Using a Global Optimization Method That Utilizes Neural Networks

Ioannis G. Tsoulos * and Alexandros Tzallas

Department of Informatics and Telecommunications, University of Ioannina, 451 10 Ioannina, Greece; tzallas@uoi.gr
* Correspondence: itsoulos@uoi.gr

Abstract: Perhaps one of the best-known machine learning models is the artificial neural network, where a number of parameters must be adjusted to learn a wide range of practical problems from areas such as physics, chemistry, medicine, etc. Such problems can be reduced to pattern recognition problems and then modeled from artificial neural networks, whether these problems are classification problems or regression problems. To achieve the goal of neural networks, they must be trained by appropriately adjusting their parameters using some global optimization methods. In this work, the application of a recent global minimization technique is suggested for the adjustment of neural network parameters. In this technique, an approximation of the objective function to be minimized is created using artificial neural networks and then sampling is performed from the approximation function and not the original one. Therefore, in the present work, learning of the parameters of artificial neural networks is performed using other neural networks. The new training method was tested on a series of well-known problems, a comparative study was conducted against other neural network parameter tuning techniques, and the results were more than promising. From what was seen after performing the experiments and comparing the proposed technique with others that have been used for classification datasets as well as regression datasets, there was a significant difference in the performance of the proposed technique, starting with 30% for classification datasets and reaching 50% for regression problems. However, the proposed technique, because it presupposes the use of global optimization techniques involving artificial neural networks, may require significantly higher execution time than other techniques.

Keywords: global optimization; neural networks; stochastic methods

Citation: Tsoulos, I.G.; Tzallas, A. Training Artificial Neural Networks Using a Global Optimization Method That Utilizes Neural Networks. *AI* **2023**, *4*, 491–508. https://doi.org/10.3390/ai4030027

Academic Editors: Kenji Suzuki and José Machado

Received: 16 May 2023
Revised: 12 July 2023
Accepted: 14 July 2023
Published: 20 July 2023

1. Introduction

Artificial neural networks (ANNs) are parametric models [1–3] in machine learning, and they are widely used in pattern recognition problems. A series of practical problems from the fields of physics [4–6], chemistry [7–9], economics [10–12], medicine [13,14], etc., can be transformed to pattern recognition problems and then solved using artificial neural networks. Furthermore, neural networks have been used with success to solve differential equations [15–17], solar radiation prediction [18,19], spam detection [20–22], etc. Moreover, variations of artificial neural networks have been employed to solve agricultural problems [23,24], facial expression recognition [25], prediction of the speed of wind [26], the gas consumption problem [27], intrusion detection [28], hydrological systems [29], etc. Furthermore, Swales and Yoon discussed the application of artificial neural networks to investment analysis in their work [30].

A neural network can be denoted as a function $N(\vec{x}, \vec{w})$ where the vector $\vec{x}$ stands for the input vector and the vector $\vec{w}$ is the set of the parameters of the neural network that should be estimated. The input vector is usually called pattern in the relevant literature

and the vector $\overrightarrow{w}$ is usually called the weight vector. Artificial neural network training methods adjust the vector of weights in order to minimize the following quantity:

$$E\left(N\left(\overrightarrow{x}, \overrightarrow{w}\right)\right) = \sum_{i=1}^{M}\left(N\left(\overrightarrow{x}_i, \overrightarrow{w}\right) - y_i\right)^2 \quad (1)$$

In the previous equation, what will be called training error, the set $\left(\overrightarrow{x_i}, y_i\right)$, $i = 1, ..., M$ is the input training dataset for the neural network with M patterns. The value y_i is the expected output for the pattern $\overrightarrow{x_i}$. Equation (1) can be minimized with respect to the weight vector using any local or global optimization method such as the back-propagation method [31,32], the hill climbing method [33], the RPROP method [34–36], quasi Newton methods [37,38], simulated annealing [39,40], genetic algorithms [41,42], particle swarm optimization [43,44], differential optimization methods [45,46], evolutionary computation [47], the whale optimization algorithm [48], etc. Furthermore, Cui et al. suggested the usage of a new stochastic optimization algorithm that simulates the plant growing process for neural network training. Furthermore, recently, the bird mating optimizer [49] was suggested as a training method for artificial neural networks [50], and hybrid methods have been developed by various researchers to optimize the weight vector, such as the work of Yaghini et al. [51] that combined particle swarm optimization with a back-propagation algorithm to minimize the error function. Moreover, Chen et al. [52] used a hybrid technique that combines particle swarm optimization and cuckoo search [53] to optimize the weight vector of neural networks.

In addition, many researchers have addressed the issue of the initial values for the weights of neural networks, such as the incorporation of decision trees [54], an initialization method using Cauchy's inequality [55], incorporation of discriminant learning [56], methods based on genetic algorithms [57], etc. A paper discussing all the aspects of weight initialization strategies was written by Narkhede et al. [58].

Moreover, various groups of researchers are dealing with the issue of constructing the structure of artificial neural networks, such as the incorporation of genetic algorithms [41], the usage of the grammatical evolution method [59] for the construction of neural networks [60], a construction and pruning approach to optimize the structure of ANNs [61], usage of cellular automata [62], etc. Furthermore, because the training of artificial neural networks with optimization methods requires a significantly longer computing time, parallel techniques have been developed that take advantage of modern parallel computing units [63–65].

Another area of research in the field of artificial neural networks that attracts a multitude of researchers is the problem of overfitting that occurs in many cases. In this problem, although the artificial neural network has achieved a satisfactory level of training, this is not reflected in unknown patterns that were not present during training. This set of patterns will be called the test set in the following. Commonly used methods that tackle the overfitting problem are weight sharing [66,67], methods that reduce the number of parameters (weight pruning) [68–70], the method of dropout [71,72], weight decaying methods [73,74], the Sarprop method [75], positive correlation methods [76], etc.

In this paper, the use of a recent global minimization technique [77] called NeuralMinimizer, is proposed to find the optimal set for the weights of artificial neural networks. This innovative global minimization technique constructs an approximation of the objective function to be minimized using a limited number of its samples. These limited samples form the training set of an artificial neural network that can be trained with any optimization method. Subsequently, the sampling for the continuation of the global optimization method is not performed by the objective function but by the previously trained artificial neural network. The samples obtained by artificial neural networks before being fed into the global minimization method are classified and those with the smallest functional value will finally be input into the global minimization method. From the experimental results, it was shown that this global minimization method requires a limited number of samples

from the objective function to find the global minimum and is also more efficient than other techniques for discovery of the global minimum. Therefore, this paper proposes using artificial neural networks to train other artificial neural networks. This new procedure will be tested on a series of known problems in order to evaluate its effectiveness.

The rest of this article is organized as follows: Section 2 describes the proposed method, Section 3 lists the experimental datasets and the results obtained by the incorporation of various methods, and finally, Section 4 discusses some conclusions.

2. The Proposed Method

In this section, some basic principles for artificial neural networks are presented and then a new training method that incorporates a modified version of the NeuralMinimizer global optimization technique is outlined.

2.1. Preliminaries

Let us consider an artificial neural network with only one hidden layer in which the sigmoid function is used as an activation function. The output value for every node in this layer is calculated as:

$$o_i(x) = \sigma(p_i x + \theta_i), \tag{2}$$

where the value p_i is the weight vector, and θ_i denotes the bias for the node i. The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{3}$$

and it is graphically illustrated in Figure 1.

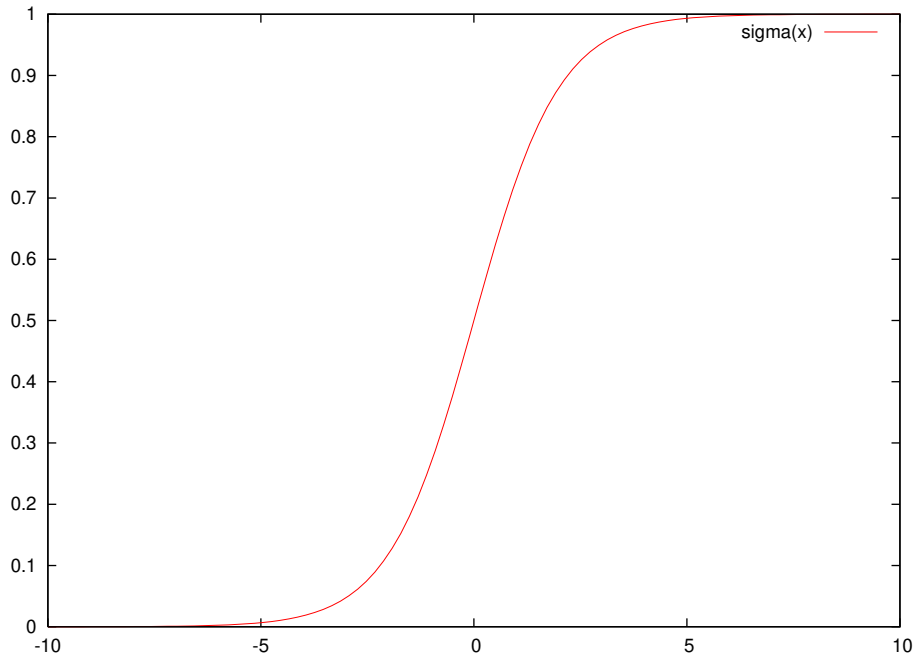

Figure 1. The sigmoid function $\sigma(x)$.

When the neural network has H processing nodes, the output can be formulated as:

$$N(x) = \sum_{i=1}^{H} v_i o_i(x), \tag{4}$$

where v_i stands for the output weight for node i. Hence, by using one vector for all the parameters (weights and biases) the neural network can be written in the following form:

$$N(\overrightarrow{x}, \overrightarrow{w}) = \sum_{i=1}^{H} w_{(d+2)i-(d+1)} \sigma \left(\sum_{j=1}^{d} x_j w_{(d+2)i-(d+1)+j} + w_{(d+2)i} \right) \quad (5)$$

where d is the dimension of vector $\overrightarrow{x}$. From Equation (5) we can conclude that number of elements in the weight vector is:

$$d_w = (d+2)H \quad (6)$$

2.2. The Modified NeuralMinimizer Method

In its original version, the NeuralMinimizer method employed RBF neural networks [78] to build a model of the objective function. Even though radial basis function (RBF) networks have been used with success in a variety of problems [79–82], it is not possible to apply them to the training of the parameters of an artificial neural network due to the large dimension of the problem, as shown in Equation (6). Hence, in the current work, the RBF network has been replaced by an artificial neural network that implements Equation (5). The training of the artificial neural network was performed using a local minimization technique that is not particularly demanding in calculations and storage space, such as limited memory BFGS (L-BFGS) [83]. Obviously, any other technique that is not extremely memory intensive could be used in its place. Such a technique could be the Adam method [84], the SGD method [85,86], or even a simple global minimization method such as a genetic algorithm with a limited number of chromosomes. The L-BFGS method is a variation of the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method [87] using a limited amount of computer memory. This local minimization method has found wide application in difficult and memory-intensive optimization problems such as image reconstruction [88], inverse eigenvalue problems [89], seismic waveform tomography [90], etc. Because of the application of this technique to large-dimensional problems, a number of modifications have been proposed that make use of modern parallel computing systems [91–93]. A numerical study on the limited memory BFGS methods is provided in the work of Morales [94]. In the original publication on the NeuralMinimizer optimization method, an RBF neural network was used to generate the approximation function of the objective function. However, this would not always be possible in cases where the objective function to be minimized is the error of an artificial neural network, because an artificial neural network usually has a large number of parameters, and this would require an extremely large storage space for training the global minimization method's RBF neural network. Of course, in some cases with small artificial neural networks, an RBF neural network could be used as the training model in the global optimization method NeuralMinimizer. However, when an artificial neural network is used to approach large and complex problems (something extremely common) then a relatively under powered RBF neural network should be used. In those cases, the RBF network will not be able to be an efficient approximation of the artificial neural network, which is, of course, the original aim of the NeuralMinimizer method.

In the following, the main steps of the modified NeuralMinimizer method for the training of neural networks are listed. In these steps, the neural network used by the NeuralMinimizer method will be called $N_N(x, w)$.

1. **Initialization** step.
   (a) **Set** H the number of weights of the neural network. In the current method the same number of weights was used for both $N(x, w)$ and $N_N(x, w)$ artificial neural networks.
   (b) **Set** N_S as the samples that will be initially drawn from $N(x, w)$. At this stage, the training error for the artificial neural network will be used as an objective function to minimize
   (c) **Set** N_T as the number of points that will be utilized as local minimization method starters in every iteration.

(d) **Set** N_R as the number of samples that will be drawn from the $N_N(x, w)$ network in each iteration.
(e) **Set** N_G as the maximum number of iterations allowed.
(f) **Set** Iter = 0, the current iteration number.
(g) **Set** (w^*, y^*) as the global minimum discovered by the method. Initially $y^* = \infty$, $w^* = (0, 0, \ldots, 0)$

2. **Creation** Step.
   (a) **Set** $T = \emptyset$, the used training set for the $N_N(x, w)$ neural network.
   (b) **For** $i = 1, .., N_S$ do
      i. **Draw** a new sample w_i from $N(x, w)$.
      ii. **Calculate** $y_i = f(w_i)$ using Equation 1
      iii. $T = T \cup (w_i, y_i)$
   (c) **EndFor**
   (d) **Train** the $N_N(x, w)$ neural network on set T using the L-BFGS method.
3. **Sampling** Step
   (a) **Set** $T_R = \emptyset$
   (b) **For** $i = 1, .., N_R$ **do**
      i. **Produce randomly** a sample (w_i, y_i) from $N_N(x, w)$ neural network
      ii. **Set** $T_R = T_R \cup (x_i, y_i)$
   (c) **EndFor**
   (d) **Sort** T_R in ascending with respect to the values y_i
4. **Optimization** Step.
   (a) **For** $i = 1, .., N_T$ do
      i. **Get** the next item (w_i, y_i) from T_R.
      ii. **Train** the neural network $N(x, w_i)$ on the training set of the objective problem using the L-BFGS method and obtain the corresponding training error y_i.
      iii. **Update** $T = T \cup (w_i, y_i)$
      iv. **Train** the network $N_N(x, w)$ again on the modified set T. In this step, the original training set used by $N_N(x, w)$ is updated to include the new discovered local minimum. This operation is used in order to construct a more accurate approximation of the real objective function.
      v. **If** $y_i \leq y^*$ then $w^* = w_i, y^* = y_i$
      vi. **If** the termination rule proposed in [95], then apply the produced network $N(x, w^*)$ on the test set of the objective problem, report the test error and **terminate**.
   (b) **EndFor**
5. **Set** iter=iter+1
6. **Goto** to Sampling step.

A flowchart of the proposed method is graphically outlined in Figure 2.

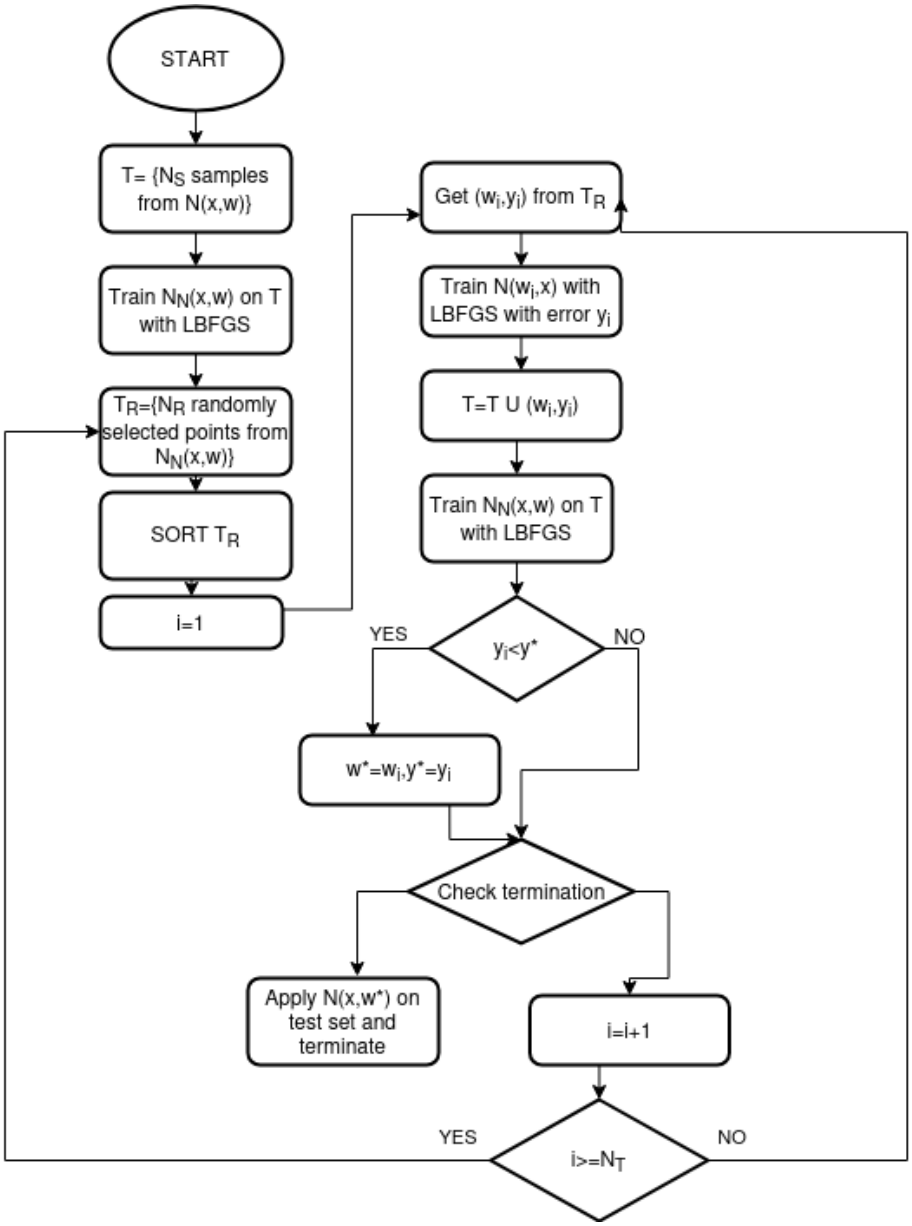

Figure 2. The flowchart of the proposed method.

3. Experiments

The effectiveness of the proposed artificial neural network training technique was evaluated using a series of data sets from the relevant literature. These datasets have been studied by various researchers in the relevant literature and cover a wide range of research areas from physics to economics. These datasets are freely available from the following websites:

1. The UCI repository, https://archive.ics.uci.edu/(accessed on 12 July 2023) [96]
2. The Keel repository, https://sci2s.ugr.es/keel/datasets.php(accessed on 17 June 2023) [97].
3. The Statlib URL ftp://lib.stat.cmu.edu/datasets/index.html (accessed on 17 June 2023). This repository is used mainly for the regression datasets.

3.1. Experimental Datasets

The following classification datasets from the relevant literature were used in the experiments:

1. **Appendicitis,** a dataset used for medical purposes; it was found in [98,99].

2. **Australian** dataset [100], an economic dataset, related to bank transactions.
3. **Balance** dataset [101], which is related to psychological experiments.
4. **Cleveland** dataset, a medical dataset found in the following research papers [102,103].
5. **Bands** dataset, related to printing problems [104].
6. **Dermatology** dataset [105], a dataset related to dermatology problems.
7. **Hayes-Roth** dataset [106].
8. **Heart** dataset [107], a medical dataset used to detect heart diseases.
9. **HouseVotes** dataset [108], related to the Congressional voting records of USA.
10. **Ionosphere** dataset, related to measurements from the ionosphere an thoroughly studied in a series of research papers [109,110].
11. **Liverdisorder** dataset [111,112], a medical dataset.
12. **Lymography** dataset [113].
13. **Mammographic** dataset [114], a medical dataset related to breast cancer diagnosis.
14. **Page Blocks** dataset [115].
15. **Parkinsons** dataset [116,117], a medical dataset applied to the Parkinson's decease.
16. **Pima** dataset [118], a medical dataset.
17. **Popfailures** dataset [119], a dataset related to meteorological data.
18. **Regions2** dataset, a medical dataset for liver biopsy images [120].
19. **Saheart** dataset [121], a medical dataset related to heart diseases.
20. **Segment** dataset [122], a dataset related to image segmentation.
21. **Wdbc** dataset [123], a dataset related to breast tumors.
22. **Wine** dataset, a dataset related to chemical analysis of wines [124,125].
23. **Eeg** datasets [126,127], an EEG dataset, and the following cases were used in the experiments:
    (a) Z_F_S,
    (b) ZO_NF_S
    (c) ZONF_S.
24. **Zoo** dataset [128], suggested for the detection of the proper classes of animals.

A table showing the number of classes for every classification dataset is shown in Table 1.

Table 1. Description for every classification dataset.

DATASET	CLASSES
Appendicitis	2
Australian	2
Balance	3
Cleveland	5
Bands	2
Dermatology	6
Hayes-Roth	3
Heart	2
Housevotes	2
Ionosphere	2
Liverdisorder	2
Lymography	4
Mammographic	2
Page Blocks	5

Table 1. *Cont.*

DATASET	CLASSES
Parkinsons	2
Pima	2
Popfailures	2
Regions2	5
Saheart	2
Segment	7
Wdbc	2
Wine	3
Z_F_S	3
ZO_NF_S	3
ZONF_S	2
Zoo	7

The following regression datasets were used:

1. **Abalone** dataset [129], proposed to predict the age of abalones.
2. **Airfoil** dataset, a dataset provided by NASA [130], created from a series of aerodynamic and acoustic tests.
3. **Baseball** dataset, a dataset using baseball games.
4. **BK** dataset [131], used to predict the points scored in a basketball game.
5. **BL** dataset, used in machine problems.
6. **Concrete** dataset [132], a dataset proposed to calculate the compressive strength of concrete.
7. **Dee** dataset, used to detect the electricity energy prices.
8. **Diabetes** dataset, a medical dataset.
9. **Housing** dataset [133].
10. **FA** dataset, used to fit body fat to other measurements.
11. **MB** dataset [131].
12. **Mortgage** dataset. The goal is to predict the 30-year conventional mortgage rate.
13. **PY** dataset, (pyrimidines problem) [134].
14. **Quake** dataset, used to approximate the strength of a earthquake given its the depth of its focal point, its latitude and its longitude.
15. **Treasure** dataset, which contains economic data information from the USA, where the goal is to predict the 1-month CD Rate.
16. **Wankara** dataset, a weather dataset.

3.2. Experimental Setup

The proposed method was tested on the regression and classification problems mentioned previously, and it was compared against the results of several other well-known optimization methods in the relevant literature. For greater reliability of the experimental results, the 10-fold validation technique was employed for every classification or regression dataset. Every experiment was executed 30 times, with different initialization for the random generator each time. Furthermore, the srand48() random generator of the C-programming language was utilized. The used code was implemented in ANSI C++ using the freely available OPTIMUS optimization library available from https://github.com/itsoulos/OPTIMUS/(accessed on 18 July 2023). For the case of the classification datasets, the average classification error was measured for every method. For regression datasets, the average mean squared error was measured in the test set. The number of hidden nodes for the neural networks was set to $H = 10$ for every method. All

the experiments were performed using an AMD Ryzen 5950X with 128 GB of RAM. The running operating system was Debian Linux. The methods used in the experimental results are the following:

1. A genetic algorithm with 200 chromosomes was used to train a neural network with H hidden nodes. This method was denoted as GENETIC in the tables holding the experimental results.
2. A radial basis function (RBF) network [78] with H hidden nodes.
3. The Adam optimization method [84]. Here, the method was used to minimize the train error of a neural network with H hidden nodes.
4. The resilient back-propagation (RPROP) optimization method [34–36] was also employed to train a neural network with H hidden nodes.
5. The NEAT method (NeuroEvolution of Augmenting Topologies) [135].

The values used for every parameter are listed in Table 2 and they are similar to the values used in the original publication of the NeuralMinimizer method.

Table 2. Experimental settings.

PARAMETER	MEANING	VALUE
H	Number of weights	10
N_S	Start samples	50
N_T	Starting points	100
N_R	Samples drawn from the first network	$10 \times N_T$
N_G	Maximum number of iterations	200

3.3. Experimental Results

The experimental results for the classification datasets are shown in Table 3 and those of the regression datasets are shown in Table 4. The column PROPOSED represents the usage of the proposed method to train a neural network with H hidden nodes. Furthermore, the Figure 3 shows a scatter plot and the Wilcoxon signed-rank test for the classification datasets. In the same direction, Figure 4 shows the scatter plot for the regression datasets.

Table 3. Experimental results for the classification datasets. The numbers in cells denote average classification error of 30 independent runs.

DATASET	GENETIC	RBF	ADAM	RPROP	NEAT	PROPOSED
Appendicitis	18.10%	12.23%	16.50%	16.30%	17.20%	22.30%
Australian	32.21%	34.89%	35.65%	36.12%	31.98%	21.59%
Balance	8.97%	33.42%	7.87%	8.81%	23.14%	5.46%
Bands	35.75%	37.22%	36.25%	36.32%	34.30%	33.06%
Cleveland	51.60%	67.10%	67.55%	61.41%	53.44%	45.41%
Dermatology	30.58%	62.34%	26.14%	15.12%	32.43%	4.14%
Hayes Roth	56.18%	64.36%	59.70%	37.46%	50.15%	35.28%
Heart	28.34%	31.20%	38.53%	30.51%	39.27%	17.93%
HouseVotes	6.62%	6.13%	7.48%	6.04%	10.89%	5.78%
Ionosphere	15.14%	16.22%	16.64%	13.65%	19.67%	16.31%
Liverdisorder	31.11%	30.84%	41.53%	40.26%	30.67%	33.02%
Lymography	23.26%	25.31%	29.26%	24.67%	33.70%	25.64%
Mammographic	19.88%	21.38%	46.25%	18.46%	22.85%	16.37%
PageBlocks	8.06%	10.09%	7.93%	7.82%	10.22%	5.44%

Table 3. *Cont.*

DATASET	GENETIC	RBF	ADAM	RPROP	NEAT	PROPOSED
Parkinsons	18.05%	17.42%	24.06%	22.28%	18.56%	14.47%
Pima	32.19%	25.78%	34.85%	34.27%	34.51%	25.61%
Popfailures	5.94%	7.04%	5.18%	4.81%	7.05%	5.57%
Regions2	29.39%	38.29%	29.85%	27.53%	33.23%	22.73%
Saheart	34.86%	32.19%	34.04%	34.90%	34.51%	34.03%
Segment	57.72%	59.68%	49.75%	52.14%	66.72%	37.28%
Wdbc	8.56%	7.27%	35.35%	21.57%	12.88%	5.01%
Wine	19.20%	31.41%	29.40%	30.73%	25.43%	7.14%
Z_F_S	10.73%	13.16%	47.81%	29.28%	38.41%	7.09%
ZO_NF_S	8.41%	9.02%	47.43%	6.43%	43.75%	5.15%
ZONF_S	2.60%	4.03%	11.99%	27.27%	5.44%	2.35%
ZOO	16.67%	21.93%	14.13%	15.47%	20.27%	4.20%

Table 4. Average regression error for the regression datasets.

DATASET	GENETIC	RBF	ADAM	RPROP	NEAT	PROPOSED
ABALONE	7.17	7.37	4.30	4.55	9.88	4.50
AIRFOIL	0.003	0.27	0.005	0.002	0.067	0.003
BASEBALL	103.60	93.02	77.90	92.05	100.39	56.16
BK	0.027	0.02	0.03	1.599	0.15	0.02
BL	5.74	0.01	0.28	4.38	0.05	0.0004
CONCRETE	0.0099	0.011	0.078	0.0086	0.081	0.003
DEE	1.013	0.17	0.63	0.608	1.512	0.30
DIABETES	19.86	0.49	3.03	1.11	4.25	1.24
HOUSING	43.26	57.68	80.20	74.38	56.49	18.30
FA	1.95	0.02	0.11	0.14	0.19	0.01
MB	3.39	2.16	0.06	0.055	0.061	0.05
MORTGAGE	2.41	1.45	9.24	9.19	14.11	3.50
PY	105.41	0.02	0.09	0.039	0.075	0.03
QUAKE	0.04	0.071	0.06	0.041	0.298	0.039
TREASURY	2.929	2.02	11.16	10.88	15.52	3.72
WANKARA	0.012	0.001	0.02	0.0003	0.005	0.002

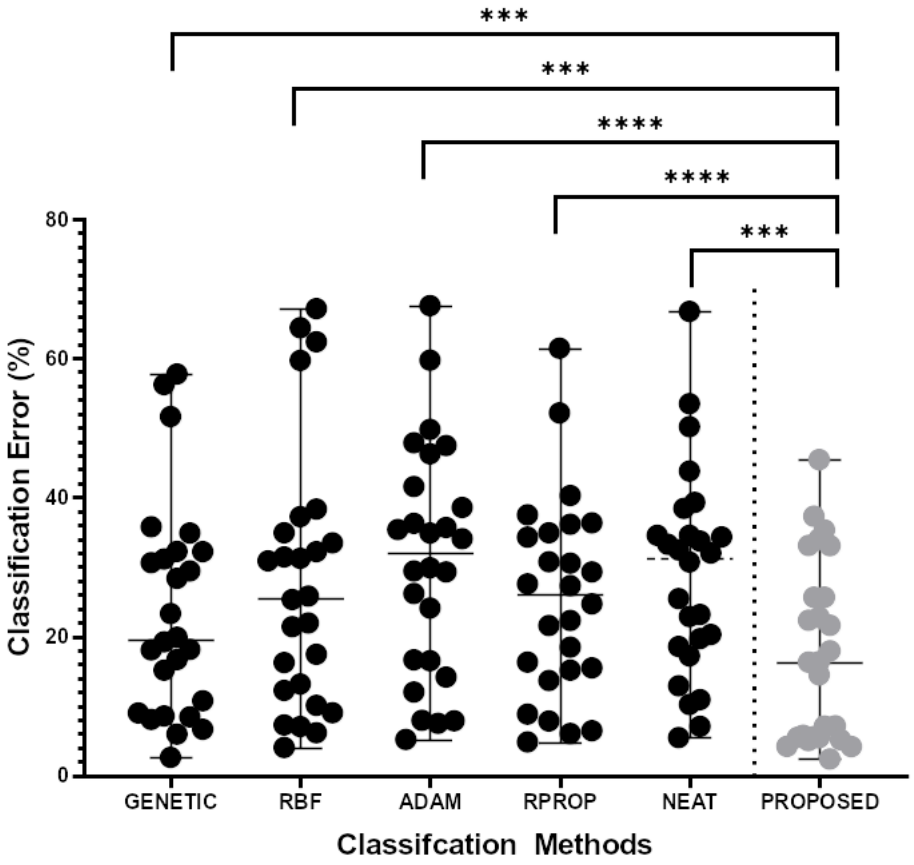

Figure 3. Scatter plot representation and the two-sample paired (Wilcoxon) signed-rank test results of the comparison for each of the five (5) clas-sification methods (GENETIC, RBF, ADAM, RPROP, NEAT) with the PROPOSED method regarding the classification error in twenty-six (26) different public available classification datasets. The stars only intend to flag significance levels for three of the most used groups. A p-value of less than 0.001 is flagged with three stars (***). A p-value of less than 0.0001 is flagged with four stars (****).

The experimental results and their graphical representation demonstrate the superiority of the proposed technique over the others in terms of the average error, as measured in the test set. For example, in the case of datasets used for classification, the proposed method outperforms the remaining techniques in 19 out of 26 datasets (73% percent). Furthermore, in several cases, the percentage reduction in error exceeds 50%. For the classification problems, the immediate most effective training method after the proposed one is the genetic algorithm and, on average, the proposed technique achieves lower classification error than the genetic algorithm error by 24%. Moreover, in regression problems, the next most effective method after the proposed one is the RBF neural network with small differences from the ADAM optimizer. However, in the case of regression problems, the improvement in average error using the proposed technique exceeds 49%. Of course, the proposed technique is quite time-consuming, because it requires the continuous training of an artificial neural network.

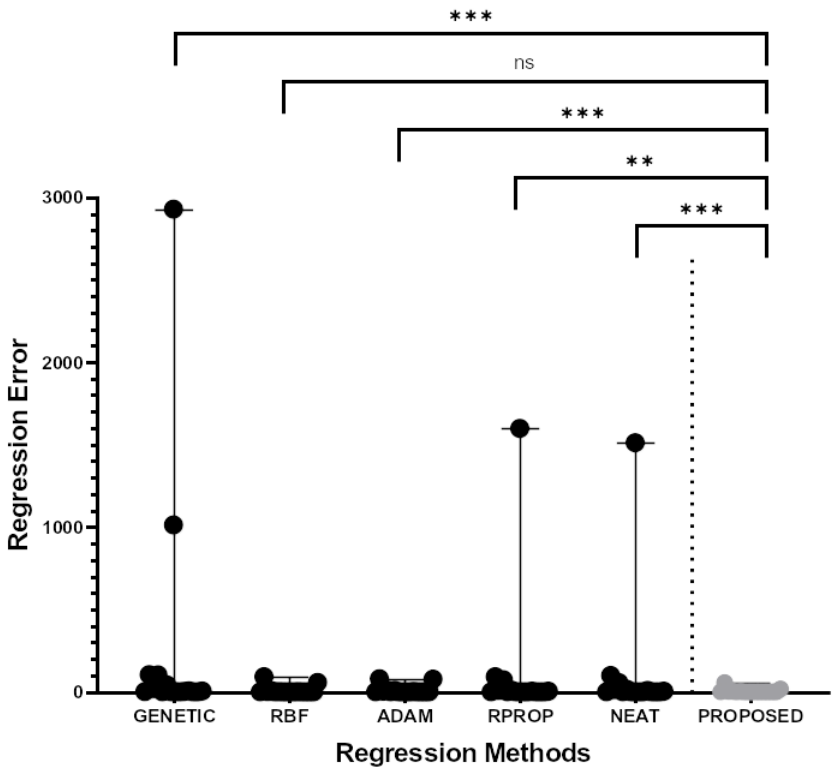

Figure 4. Scatter plot representation and the two-sample paired (Wilcoxon) signed-rank test of the comparison for each of the five (5) regression methods (GENETIC, RBF, ADAM, RPROP, NEAT) with the PROPOSED method regarding the regression error in sixteen (16) different publicly available classification datasets. The stars only intend to flag significance levels for three of the most used groups. A p-value of less than 0.01 is flagged with two stars (**). A p-value of less than 0.001 is flagged with three stars (***). The notation "ns" denotes "not significant".

4. Conclusions

In this work, the application of a recent global minimization method for the training of artificial neural networks was proposed. The application of this method was used in artificial neural networks both for classification problems and for regression problems. This new global minimization method constructs an approximation of the objective function using neural networks. This construction is performed with a limited number of samples from the objective function. However, each time a local minimization takes place, this approximation is readjusted. Subsequently, the sampling for the minimization is performed from the approximate function and not from the objective one, even taking samples from the approximation with the smallest function value in order to speed up the discovery of the global minimum. In this particular case, the artificial neural network of the global minimization method is used to train the artificial neural network. However, due to the large time and storage requirements of artificial neural networks, the RBF network of the original NeuralMinimizer method was replaced with an artificial neural network that was trained using the local minimization method L-BFGS. The new artificial neural network training technique is tested on a wide collection of classification and regression problems from the relevant literature and is shown to significantly improve the learning error over other established artificial neural network training techniques. This improvement is 25% on average for the case of classification problems and rises significantly to 50% for regression problems. The proposed method outperforms the other methods and models in the majority of cases. For example, in the classification datasets, the proposed method outperforms the genetic algorithm in 22 datasets, the RBF model in 21 datasets, the ADAM optimizer in 23 cases, the RPROP optimizer in 22 cases and finally, the NEAT method in 25 datasets.

Nevertheless, the proposed procedure can be extremely slow, especially as the size of the artificial neural network increases. The size of the artificial neural network directly depends on the dimension of the input dataset. Future improvements to the methodology may include the use of parallel programming techniques, such as parallel implementations of the L-BFGS optimization method, in order to accelerate the training of artificial neural networks by taking advantage of modern computing structures. Furthermore, in the present phase, as a minimization method in step 4 of the proposed training method, a local minimization method is used. Future extensions could explore the possibility of also using global minimization techniques in this step, although care should be taken to make use of parallel computing techniques to avoid long execution times.

Author Contributions: I.G.T. and A.T. conceived of the idea and the methodology and I.G.T. implemented the corresponding software. I.G.T. conducted the experiments, employing objective functions as test cases, and provided the comparative experiments. A.T. performed the necessary statistical tests. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call Research–Create–Innovate, project name "Create a system of recommendations and augmented reality applications in a hotel" (project code: T1EDK-03745).

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

References

1. Bishop, C. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, 1995.
2. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
3. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef] [PubMed]
4. Baldi, P.; Cranmer, K.; Faucett, T.; Sadowski, P.; Whiteson, D. Parameterized neural networks for high-energy physics. *Eur. Phys. J. C* **2016**, *76*, 235. [CrossRef]
5. Valdas, J.J.; Bonham-Carter, G. Time dependent neural network models for detecting changes of state in complex processes: Applications in earth sciences and astronomy. *Neural Netw.* **2006**, *19*, 196–207. [CrossRef] [PubMed]
6. Carleo, G.; Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **2017**, *355*, 602–606. [CrossRef]
7. Shen, L.; Wu, J.; Yang, W. Multiscale Quantum Mechanics/Molecular Mechanics Simulations with Neural Networks. *J. Chem. Theory Comput.* **2016**, *12*, 4934–4946. [CrossRef]
8. Manzhos, S.; Dawes, R.; Carrington, T. Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces. *Int. J. Quantum Chem.* **2015**, *115*, 1012–1020. [CrossRef]
9. Wei, J.N.; Duvenaud, D.; Aspuru-Guzik, A. Neural Networks for the Prediction of Organic Chemistry Reactions. *ACS Cent. Sci.* **2016**, *2*, 725–732. [CrossRef]
10. Falat, L.; Pancikova, L. Quantitative Modelling in Economics with Advanced Artificial Neural Networks. *Procedia Econ. Financ.* **2015**, *34*, 194–201. [CrossRef]
11. Namazi, M.; Shokrolahi, A.; Maharluie, M.S. Detecting and ranking cash flow risk factors via artificial neural networks technique. *J. Bus. Res.* **2016**, *69*, 1801–1806. [CrossRef]
12. Tkacz, G. Neural network forecasting of Canadian GDP growth. *Int. J. Forecast.* **2001**, *17*, 57–69. [CrossRef]
13. Baskin, I.I.; Winkler, D.; Tetko, I.V. A renaissance of neural networks in drug discovery. *Expert Opin. Drug Discov.* **2016**, *11*, 785–795. [CrossRef] [PubMed]
14. Bartzatt, R. Prediction of Novel Anti-Ebola Virus Compounds Utilizing Artificial Neural Network (ANN). *World J. Pharm. Res.* **2018**, *7*, 16.
15. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **1998**, *9*, 987–1000. [CrossRef] [PubMed]
16. Effati, S.; Pakdaman, M. Artificial neural network approach for solving fuzzy differential equations. *Inf. Sci.* **2010**, *180*, 1434–1457. [CrossRef]

17. Rostami, F.; Jafarian, A. A new artificial neural network structure for solving high-order linear fractional differential equations. *Int. J. Comput. Math.* **2018**, *95*, 528–539. [CrossRef]
18. Yadav, A.K.; Chandel, S.S. Solar radiation prediction using Artificial Neural Network techniques: A review. *Renew. Sustain. Energy Rev.* **2014**, *33*, 772–781. [CrossRef]
19. Qazi, A.; Fayaz, H.; Wadi, A.; Raj, R.G.; Rahim, N.A.; Khan, W.A. The artificial neural network for solar radiation prediction and designing solar systems: A systematic literature review. *J. Clean. Prod.* **2015**, *104*, 1–12. [CrossRef]
20. Wu, C.H. Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks. *Expert Syst. Appl.* **2009**, *36*, 4321–4330. [CrossRef]
21. Ren, Y.; Ji, D. Neural networks for deceptive opinion spam detection: An empirical study. *Inf. Sci.* **2017**, *385–386*, 213–224. [CrossRef]
22. Madisetty, S.; Desarkar, M.S. A Neural Network-Based Ensemble Approach for Spam Detection in Twitter. *IEEE Trans. Comput. Soc. Syst.* **2018**, *5*, 973–984. [CrossRef]
23. Topuz, A. Predicting moisture content of agricultural products using artificial neural networks. *Adv. Eng.* **2010**, *41*, 464–470. [CrossRef]
24. Escamilla-García, A.; Soto-Zarazúa, G.M.; Toledano-Ayala, M.; Rivas-Araiza, E.; Gastélum-Barrios, A. Applications of Artificial Neural Networks in Greenhouse Technology and Overview for Smart Agriculture Development. *Appl. Sci.* **2020**, *10*, 3835. [CrossRef]
25. Boughrara, H.; Chtourou, M.; Ben Amar, C.; Chen, L. Facial expression recognition based on a mlp neural network using constructive training algorithm. *Multimed. Tools Appl.* **2016**, *75*, 709–731. [CrossRef]
26. Liu, H.; Tian, H.Q.; Li, Y.F.; Zhang, L. Comparison of four Adaboost algorithm based artificial neural networks in wind speed predictions. *Energy Convers. Manag.* **2015**, *92*, 67–81. [CrossRef]
27. Szoplik, J. Forecasting of natural gas consumption with artificial neural networks. *Energy* **2015**, *85*, 208–220. [CrossRef]
28. Bahram, H.; Navimipour, N.J. Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. *ICT Express* **2019**, *5*, 56–59.
29. Chen, Y.S.; Chang, F.J. Evolutionary artificial neural networks for hydrological systems forecasting. *J. Hydrol.* **2009**, *367*, 125–137. [CrossRef]
30. Swales, G.S.; Yoon, Y. Applying Artificial Neural Networks to Investment Analysis. *Financ. Anal. J.* **1992**, *48*, 78–80. [CrossRef]
31. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
32. Chen, T.; Zhong, S. Privacy-Preserving Backpropagation Neural Network Learning. *IEEE Trans. Neural Netw.* **2009**, *20*, 1554–1564. [CrossRef] [PubMed]
33. Chalup, S.; Maire, F. A study on hill climbing algorithms for neural network training. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; IEEE: Toulouse, France, 1999; Volume 3, pp. 2014–2021.
34. Riedmiller, M.; Braun, H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm. In Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, USA, 28 March–1 April 1993; IEEE: Toulouse, France, 1993; pp. 586–591.
35. Pajchrowski, T.; Zawirski, K.; Nowopolski, K. Neural Speed Controller Trained Online by Means of Modified RPROP Algorithm. *IEEE Trans. Ind. Informatics* **2015**, *11*, 560–568. [CrossRef]
36. Hermanto, R.P.S.; Nugroho, A. Waiting-Time Estimation in Bank Customer Queues using RPROP Neural Networks. *Procedia Comput. Sci.* **2018**, *135*, 35–42. [CrossRef]
37. Robitaille, B.; Marcos, B.; Veillette, M.; Payre, G. Modified quasi-Newton methods for training neural networks. *Comput. Chem. Eng.* **1996**, *20*, 1133–1140. [CrossRef]
38. Liu, Q.; Liu, J.; Sang, R.; Li, J.; Zhang, T.; Zhang, Q. Fast Neural Network Training on FPGA Using Quasi-Newton Optimization Method. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *26*, 1575–1579. [CrossRef]
39. Yamazaki, A.; de Souto, M.C.P.; Ludermir, T.B. Optimization of neural network weights and architectures for odor recognition using simulated annealing. In Proceedings of the 2002 International Joint Conference on Neural Networks, IJCNN'02, Honolulu, HI, USA, 12–17 May 2002; IEEE: Toulouse, France, 2002; Volume 1, pp. 547–552.
40. Da, Y.; Xiurun, G. An improved PSO-based ANN with simulated annealing technique. *Neurocomputing* **2005**, *63*, 527–533. [CrossRef]
41. Leung, F.H.F.; Lam, H.K.; Ling, S.H.; Tam, P.K.S. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Trans. Neural Netw.* **2003**, *14*, 79–88. [CrossRef]
42. Yao, X. Evolving artificial neural networks. *Proc. IEEE* **1999**, *87*, 1423–1447.
43. Zhang, C.; Shao, H.; Li, Y. Particle swarm optimisation for evolving artificial neural network. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Nashville, TN, USA, 8–11 October 2000; IEEE: Toulouse, France, 2000; pp. 2487–2490.
44. Yu, J.; Wang, S.; Xi, L. Evolving artificial neural networks using an improved PSO and DPSO. *Neurocomputing* **2008**, *71*, 1054–1060. [CrossRef]

45. Ilonen, J.; Kamarainen, J.K.; Lampinen, J. Differential Evolution Training Algorithm for Feed-Forward Neural Networks. *Neural Process. Lett.* **2003**, *17*, 93–105. [CrossRef]
46. Slowik, A.; Bialko, M. Training of artificial neural networks using differential evolution algorithm. In Proceedings of the 2008 Conference on Human System Interactions, Krakow, Poland, 25–27 May 2008; IEEE: Toulouse, France, 2008; pp. 60–65.
47. Rocha, M.; Cortez, P.; Neves, J. Evolution of neural networks for classification and regression. *Neurocomputing* **2007**, *70*, 2809–2816. [CrossRef]
48. Aljarah, I.; Faris, H.; Mirjalili, S. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Comput.* **2018**, *22*, 1–15. [CrossRef]
49. Askarzadeh, A.; Rezazadeh, A. Artificial neural network training using a new efficient optimization algorithm. *Appl. Soft Comput.* **2013**, *13*, 1206–1213. [CrossRef]
50. Cui, Z.; Yang, C.; Sanyal, S. Training artificial neural networks using APPM. *Int. J. Wirel. And Mobile Comput.* **2012**, *5*, 168–174. [CrossRef]
51. Yaghini, M.; Khoshraftar, M.M.; Fallahi, M. A hybrid algorithm for artificial neural network training. *Eng. Appl. Artif. Intell.* **2013**, *26*, 293–301. [CrossRef]
52. Chen, J.F.; Do, Q.H.; Hsieh, H.N. Training Artificial Neural Networks by a Hybrid PSO-CS Algorithm. *Algorithms* **2015**, *8*, 292–308. [CrossRef]
53. Yang, X.S.; Deb, S. Engineering Optimisation by Cuckoo Search. *Int. J. Math. Model. Numer. Optim.* **2010**, *1*, 330–343. [CrossRef]
54. Ivanova, I.; Kubat, M. Initialization of neural networks by means of decision trees. *Knowl.-Based Syst.* **1995**, *8*, 333–344. [CrossRef]
55. Yam, J.Y.F.; Chow, T.W.S. A weight initialization method for improving training speed in feedforward neural network. *Neurocomputing* **2000**, *30*, 219–232. [CrossRef]
56. Chumachenko, K.; Iosifidis, A.; Gabbouj, M. Feedforward neural networks initialization based on discriminant learning. *Neural Netw.* **2022**, *146*, 220–229. [CrossRef]
57. Itano, F.; de Sousa, M.A.d.A.; Del-Moral-Hernandez, E. Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; IEEE: Toulouse, France, 2018; pp. 1–8.
58. Narkhede, M.V.; Bartakke, P.P.; Sutaone, M.S. A review on weight initialization strategies for neural networks. *Artif. Intell. Rev.* **2022**, *55*, 291–322. [CrossRef]
59. O'Neill, M.; Ryan, C. Grammatical evolution. *IEEE Trans. Evol. Comput.* **2001**, *5*, 349–358. [CrossRef]
60. Tsoulos, I.G.; Gavrilis, D.; Glavas, E. Neural network construction and training using grammatical evolution. *Neurocomputing* **2008**, *72*, 269–277. [CrossRef]
61. Han, H.G.; Qiao, J.F. A structure optimisation algorithm for feedforward neural network construction. *Neurocomputing* **2013**, *99*, 347–357. [CrossRef]
62. Kim, K.J.; Cho, S.B. Evolved neural networks based on cellular automata for sensory-motor controller. *Neurocomputing* **2006**, *69*, 2193–2207. [CrossRef]
63. Martínez-Zarzuela, M.; Díaz Pernas, F.J.; Díez Higuera, J.F.; Rodríguez, M.A. Fuzzy ART Neural Network Parallel Computing on the GPU. In *Computational and Ambient Intelligence. IWANN 2007*; Lecture Notes in Computer Science; Sandoval, F., Prieto, A., Cabestany, J., Graña, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4507.
64. Sierra-Canto, X.; Madera-Ramirez, F.; Uc-Cetina, V. Parallel Training of a Back-Propagation Neural Network Using CUDA. In Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications, Washington, DC, USA, 12–14 December 2010; IEEE: Toulouse, France, 2010; pp. 307–312.
65. Huqqani, A.A.; Schikuta, E.; Chen, S.Y.P. Multicore and GPU Parallelization of Neural Networks for Face Recognition. *Procedia Comput. Sci.* **2013**, *18*, 349–358. [CrossRef]
66. Nowlan, S.J.; Hinton, G.E. Simplifying neural networks by soft weight sharing. *Neural Comput.* **1992**, *4*, 473–493. [CrossRef]
67. Kim, J.K.; Lee, M.Y.; Kim, J.Y.; Kim, B.J.; Lee, J.H. An efficient pruning and weight sharing method for neural network. In Proceedings of the 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Seoul, Republic of Korea, 26–28 October 2016; IEEE: Toulouse, France, 2016; pp. 1–2.
68. Hanson, S.J.; Pratt, L.Y. Comparing biases for minimal network construction with back propagation. In *Advances in Neural Information Processing Systems*; Touretzky, D.S., Ed.; Morgan Kaufmann: San Mateo, CA, USA, 1989; Volume 1, pp. 177–185.
69. Mozer, M.C.; Smolensky, P. Skeletonization: A technique for trimming the fat from a network via relevance assesment. In *Advances in Neural Processing Systems*; Touretzky, D.S., Ed.; Morgan Kaufmann: San Mateo, CA, USA, 1989; Volume 1, pp. 107–115.
70. Augasta, M.; Kathirvalavakumar, T. Pruning algorithms of neural networks—a comparative study. *Cent. Eur. Comput. Sci.* **2003**, *3*, 105–115. [CrossRef]
71. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
72. Iosifidis, A.; Tefas, A.; Pitas, I. DropELM: Fast neural network regularization with Dropout and DropConnect. *Neurocomputing* **2015**, *162*, 57–66. [CrossRef]
73. Gupta, A.; Lam, S.M. Weight decay backpropagation for noisy data. *Neural Netw.* **1998**, *11*, 1127–1138. [CrossRef] [PubMed]

74. Carvalho, M.; Ludermir, T.B. Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay. In Proceedings of the 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06), Rio de Janeiro, Brazil, 13–15 December 2006; IEEE: Toulouse, France, 2006; p. 5.
75. Treadgold, N.K.; Gedeon, T.D. Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm. *IEEE Trans. Neural Netw.* **1998**, *9*, 662–668. [CrossRef] [PubMed]
76. Shahjahan, M.D.; Kazuyuki, M. Neural network training algorithm with possitive correlation. *IEEE Trans. Inf. Syst.* **2005**, *88*, 2399–2409. [CrossRef]
77. Tsoulos, I.G.; Tzallas, A.; Karvounis, E.; Tsalikakis, D. NeuralMinimizer: A Novel Method for Global Optimization. *Information* **2023**, *14*, 66. [CrossRef]
78. Park, J.; Sandberg, I.W. Universal Approximation Using Radial-Basis-Function Networks. *Neural Comput.* **1991**, *3*, 246–257. [CrossRef] [PubMed]
79. Mai-Duy, N.; Tran-Cong, T. Numerical solution of differential equations using multiquadric radial basis function networks. *Neural Netw.* **2001**, *14*, 185–199. [CrossRef]
80. Mai-Duy, N. Solving high order ordinary differential equations with radial basis function networks. *Int. J. Numer. Meth. Engng.* **2005**, *62*, 824–852. [CrossRef]
81. Laoudias, C.; Kemppi, P.; Panayiotou, C.G. Localization Using Radial Basis Function Networks and Signal Strength Fingerprints in WLAN. In Proceedings of the GLOBECOM 2009—2009 IEEE Global Telecommunications Conference, Honolulu, HI, USA, 30 November–4 December 2009; IEEE: Toulouse, France, 2009; pp. 1–6.
82. Azarbad, M.; Hakimi, S.; Ebrahimzadeh, A. Automatic recognition of digital communication signal. *Int. J. Energy Inf. Commun.* **2012**, *3*, 21–33.
83. Liu, D.C.; Nocedal, J. On the Limited Memory Method for Large Scale Optimization. *Math. Program.* **1989**, *45*, 503–528. [CrossRef]
84. Kingma, D.P.; Ba, J.L. ADAM: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
85. Wang, L.; Yang, Y.; Min, R.; Chakradhar, S. Accelerating deep neural network training with inconsistent stochastic gradient descent. *Neural Netw.* **2017**, *93*, 219–229. [CrossRef]
86. Sharma, A. Guided Stochastic Gradient Descent Algorithm for inconsistent datasets. *Appl. Soft Comput.* **2018**, *73*, 1068–1080. [CrossRef]
87. Fletcher, R. A new approach to variable metric algorithms. *Comput. J.* **1970**, *13*, 317–322. [CrossRef]
88. Wang, H.; Gemmeke, H.; Hopp, T.; Hesser, J. Accelerating image reconstruction in ultrasound transmission tomography using L-BFGS algorithm. In *Medical Imaging 2019: Ultrasonic Imaging and Tomography; 109550B (2019)*; SPIE Medical Imaging: San Diego, CA, USA, 2019. [CrossRef]
89. Dalvand, Z.; Hajarian, M. Solving generalized inverse eigenvalue problems via L-BFGS-B method. *Inverse Probl. Sci. Eng.* **2020**, *28*, 1719–1746. [CrossRef]
90. Rao, Y.; Wang, Y. Seismic waveform tomography with shot-encoding using a restarted L-BFGS algorithm. *Sci. Rep.* **2017**, *7*, 8494. [CrossRef] [PubMed]
91. Fei, Y.; Rong, G.; Wang, B.; Wang, W. Parallel L-BFGS-B algorithm on GPU. *Comput. Graph.* **2014**, *40*, 1–9. [CrossRef]
92. D'Amore, L.; Laccetti, G.; Romano, D.; Scotti, G.; Murli, A. Towards a parallel component in a GPU—CUDA environment: A case study with the L-BFGS Harwell routine. *Int. J. Comput. Math.* **2015**, *92*, 59–76. [CrossRef]
93. Najafabadi, M.M.; Khoshgoftaar, T.M.; Villanustre, F.; Holt, J. Large-scale distributed L-BFGS. *J. Big Data* **2017**, *4*, 22. [CrossRef]
94. Morales, J.L. A numerical study of limited memory BFGS methods. *Appl. Math. Lett.* **2002**, *15*, 481–487. [CrossRef]
95. Tsoulos, I.G. Modifications of real code genetic algorithm for global optimization. *Appl. Math. Comput.* **2008**, *203*, 598–607. [CrossRef]
96. Kelly, M.; Longjohn, R.; Nottingham, K. The UCI Machine Learning Repository. 2023. Available online: https://archive.ics.uci.edu (accessed on 18 July 2023).
97. Alcalá-Fdez, J.; Fernandez, A.; Luengo, J.; Derrac, J.; García, S.; Sánchez, L.; Herrera, F. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *J.-Mult.-Valued Log. Soft Comput.* **2011**, *17*, 255–287.
98. Weiss, S.M.; Kulikowski, C.A. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 1991.
99. Wang, M.; Zhang, Y.Y.; Min, F. Active learning through multi-standard optimization. *IEEE Access* **2019**, *7*, 56772–56784. [CrossRef]
100. Quinlan, J.R. Simplifying Decision Trees. *Int.-Man-Mach. Stud.* **1987**, *27*, 221–234. [CrossRef]
101. Shultz, T.; Mareschal, D.; Schmidt, W. Modeling Cognitive Development on Balance Scale Phenomena. *Mach. Learn.* **1994**, *16*, 59–88. [CrossRef]
102. Zhou, Z.H.; Jiang, Y. NeC4.5: Neural ensemble based C4.5. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 770–773. [CrossRef]
103. Setiono, R.; Leow, W.K. FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks. *Appl. Intell.* **2000**, *12*, 15–25. [CrossRef]
104. Evans, B.; Fisher, D. Overcoming process delays with decision tree induction. *IEEE Expert* **1994**, *9*, 60–66. [CrossRef]
105. Demiroz, G.; Govenir, H.A.; Ilter, N. Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals. *Artif. Intell. Med.* **1998**, *13*, 147–165.

106. Hayes-Roth, B.; Hayes-Roth, B.F. Concept learning and the recognition and classification of exemplars. *J. Verbal Learn. Verbal Behav.* **1977**, *16*, 321–338. [CrossRef]
107. Kononenko, I.; Šimec, E.; Robnik-Šikonja, M. Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF. *Appl. Intell.* **1997**, *7*, 39–55. [CrossRef]
108. French, R.M.; Chater, N. Using noise to compute error surfaces in connectionist networks: A novel means of reducing catastrophic forgetting. *Neural Comput.* **2002**, *14*, 1755–1769. [CrossRef]
109. Dy, J.G.; Brodley, C.E. Feature Selection for Unsupervised Learning. *J. Mach. Learn. Res.* **2004**, *5*, 845–889.
110. Perantonis, S.J.; Virvilis, V. Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis. *Neural Process. Lett.* **1999**, *10*, 243–252. [CrossRef]
111. Garcke, J.; Griebel, M. Classification with sparse grids using simplicial basis functions. *Intell. Data Anal.* **2002**, *6*, 483–502. [CrossRef]
112. Mcdermott, J.; Forsyth, R.S. Diagnosing a disorder in a classification benchmark. *Pattern Recognit. Lett.* **2016**, *73*, 41–43. [CrossRef]
113. Cestnik, G.; Konenenko, I.; Bratko, I. Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. In *Progress in Machine Learning*; Bratko, I., Lavrac, N., Eds.; Sigma Press: Wilmslow, UK, 1987; pp. 31–45.
114. Elter, M.; Schulz-Wendtland, R.; Wittenberg, T. The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process. *Med. Phys.* **2007**, *34*, 4164–4172. [CrossRef] [PubMed]
115. Malerba, F.E.F.D.; Semeraro, G. Multistrategy Learning for Document Recognition. *Appl. Artif. Intell.* **1994**, *8*, 33–84.
116. Little, M.; Mcsharry, P.; Roberts, S.; Costello, D.; Moroz, I. Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection. *BioMed. Eng. OnLine* **2007**, *6*, 23. [CrossRef]
117. Little, M.A.; McSharry, P.E.; Hunter, E.J.; Spielman, J.; Ramig, L.O. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Trans. Biomed. Eng.* **2009**, *56*, 1015–1022. [CrossRef]
118. Smith, J.W.; Everhart, J.E.; Dickson, W.C.; Knowler, W.C.; Johannes, R.S. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care*; IEEE Computer Society Press: Piscataway, NJ, USA; American Medical Informatics Association: Bethesda, MD, USA, 1988; pp. 261–265.
119. Lucas, D.D.; Klein, R.; Tannahill, J.; Ivanova, D.; Brandon, S.; Domyancic, D.; Zhang, Y. Failure analysis of parameter-induced simulation crashes in climate models. *Geosci. Model Dev.* **2013**, *6*, 1157–1171. [CrossRef]
120. Giannakeas, N.; Tsipouras, M.G.; Tzallas, A.T.; Kyriakidi, K.; Tsianou, Z.E.; Manousou, P.; Hall, A.; Karvounis, E.C.; Tsianos, V.; Tsianos, E. A clustering based method for collagen proportional area extraction in liver biopsy images. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, Milan, Italy, 25–29 August 2015; IEEE: Toulouse, France, 2015; pp. 3097–3100.
121. Hastie, T.; Tibshirani, R. Non-parametric logistic and proportional odds regression. *JRSS-C (Appl. Stat.)* **1987**, *36*, 260–276. [CrossRef]
122. Dash, M.; Liu, H.; Scheuermann, P.; Tan, K.L. Fast hierarchical clustering and its validation. *Data Knowl. Eng.* **2003**, *44*, 109–138. [CrossRef]
123. Wolberg, W.H.; Mangasarian, O.L. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. Natl. Acad. Sci. USA* **1990**, *87*, 9193–9196. [CrossRef] [PubMed]
124. Raymer, M.; Doom, T.E.; Kuhn, L.A.; Punch, W.F. Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. *IEEE Trans. Syst. Man. Cybern.* **2003**, *33*, 802–813. [CrossRef]
125. Zhong, P.; Fukushima, M. Regularized nonsmooth Newton method for multi-class support vector machines. *Optim. Methods Softw.* **2007**, *22*, 225–236. [CrossRef]
126. Andrzejak, R.G.; Lehnertz, K.; Mormann, F.; Rieke, C.; David, P.; Elger, C.E. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Phys. Rev.* **2001**, *64*, 061907. [CrossRef]
127. Tzallas, A.T.; Tsipouras, M.G.; Fotiadis, D.I. Automatic Seizure Detection Based on Time-Frequency Analysis and Artificial Neural Networks. *Comput. Intell. Neurosci.* **2007**, *2007*, 80510. [CrossRef]
128. Koivisto, M.; Sood, K. Exact Bayesian Structure Discovery in Bayesian Networks. *J. Mach. Learn. Res.* **2004**, *5*, 549–573.
129. Nash, W.J.; Sellers, T.L.; Talbot, S.R.; Cawthor, A.J.; Ford, W.B. The Population Biology of Abalone (*haliotis* species). In *Blacklip Abalone (H. rubra) from the North Coast and Islands of Bass Strait*; Tasmania, I., Ed.; Technical Report; Sea Fisheries Division: Tasmania, Australia, 1994; ISSN 1034-3288.
130. Brooks, T.F.; Pope, D.S.; Marcolini, A.M. *Airfoil Self-Noise and Prediction*; Technical report; NASA: Washington, DC, USA, 1989.
131. Simonoff, J.S. *Smooting Methods in Statistics*; Springer: Berlin/Heidelberg, Germany, 1996.
132. Yeh, I.C. Modeling of strength of high performance concrete using artificial neural networks. *Cem. Concr. Res.* **1998**, *28*, 1797–1808. [CrossRef]
133. Harrison, D.; Rubinfeld, D.L. Hedonic prices and the demand for clean ai. *J. Environ. Econ. Manag.* **1978**, *5*, 81–102. [CrossRef]

134. King, R.D.; Muggleton, S.; Lewis, R.; Sternberg, M.J.E. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proc. Natl. Acad. Sci. USA* **1992**, *89*, 11322–11326. [CrossRef]
135. Stanley, K.O.; Miikkulainen, R. Evolving Neural Networks through Augmenting Topologies. *Evol. Comput.* **2002**, *10*, 99–127. [CrossRef] [PubMed]