MDPI

Article

Gen2Gen: Efficiently training artificial neural networks using a series of genetic algorithms

Ioannis G. Tsoulos[1,*], Vasileios Charilogis[2]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece; itsoulos@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece; v.charilog@uoi.gr
* Correspondence: itsoulos@uoi.gr

Abstract: Artificial neural networks have been used in a multitude of applications in various research areas in recent decades, providing excellent results in both data classification and data fitting. Their success is based on the effective identification (training) of their parameters using optimization techniques, and hence a series of programming methods have been developed for training these models. However, many times these techniques either can identity only some local minima of the error function with poor overall results or present overfitting problems in which the performance of the artificial neural network is significantly reduced when it is applied on different data from the training set. This manuscript introduces a method for the efficient training of artificial neural networks, where a series of genetic algorithms is applied to the network parameters in several stages. In the first stage, an initial identification of the network value interval is made, in the second stage, the initial estimate of the value interval is improved, and in the third stage, the final adjustment of the network parameters within the previously identified value interval takes place. The new method was tested on some classification and regression problems found in the relevant literature, and the experimental results was compared against the results obtained by the application of other well - known methods used for neural network training.

Keywords: Genetic algorithms; Evolutionary computation; Artificial neural networks

Received:
Revised:
Accepted:
Published:

Citation: Tsoulos, I.G.; Charilogis, V. Gen2Gen: Efficiently training artificial neural networks using a series of genetic algorithms. *Journal Not Specified* **2025**, *1*, 0. https://doi.org/

1. Introduction

A widely used machine learning model is artificial neural network [1,2]. In most cases artificial neural networks are defined as parametric machine learning models, where learning is achieved by calculating the values of their parameters through some optimization technique. The underlying optimization procedure should minimize the associated training error, calculated as:

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

The function $N\left(\overrightarrow{x},\overrightarrow{w}\right)$ stands for the artificial neural network and the input vector $\overrightarrow{x}$ represents the input pattern. Also, the input vector $\overrightarrow{w}$ stands for the parameter vector of the neural network. The set $\left(\overrightarrow{x_i},y_i\right),\ i=1,...,M$ defines the so - called training set of the objective problem, where the values y_i are the expected outputs for every pattern $\overrightarrow{x_i}$. A closed form that can be used to represent neural networks was used in [3], where it is defined as:

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

Here, the constant H defines the number of processing units and the symbol d stands for the dimension of the input pattern $\overrightarrow{x}$. The function $\sigma(x)$ represents the sigmoid function:

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

Following equation 2 it is deduced that the total number of elements in the weight vector are: $n = (d+2)H$. Also, alternative activation functions can be used. As an example consider the the tanh defined as:

$$\tanh(x) = \frac{e^{2x}+1}{e^{2x}-1} \tag{4}$$

Moreover, Guarnieri et al introduced the usage of an adaptive spline function as the activation function of neural networks [4]. Similarly, Ertuğrul introduced the trained activation function [5]. Recently, Rasamoelina et al [6] published a review on activation functions for artificial neural networks.

Artificial neural networks have been incorporated in a wide series of problems appeared in real - world problems, such as image processing [7], time series forecasting [8], credit card analysis [9], problems derived from physics [10,11], medicine [12,13], mechanical applications [14] etc. During the past years a series of optimization methods have been incorporated to tackle the equation 1. Among them one can detect the Back Propagation algorithm [15,16], the RPROP algorithm [17,18], the ADAM optimization method [19] etc. Moreover, many advanced global optimization methods have been also used, such as Genetic Algorithms [20], the Particle Swarm Optimization (PSO) method [21], the Simulated Annealing method [22], the Differential Evolution technique [23], the Artificial Bee Colony (ABC) method [24] etc. In the same direction of research, Sexton et al proposed the application of the tabu search algorithm for optimal neural network training [25]. Additionally, Zhang et al introduced a hybrid algorithm that combines PSO and the Back Propagation algorithm to efficient train artificial neural networks [26]. Also, in a recently published paper, Zhao et al proposed the usage of a new Cascaded Forward Algorithm to train artificial neural networks [27]. Additionally, the widespread use of parallel processing techniques in recent years has resulted in a series of related works on the training of artificial neural networks that utilize such techniques [28,29].

Nevertheless, the previous optimization methods faced a number of problems such as, for example, they can identity only local minima of the error function or they suffer from the phenomenon of overifitting. In overfitting, the artificial neural network exhibits reduced performance when it used on data that was not used in the training process. This problem has been tackled by various researchers in the past and some methods have been introduced, such as weight sharing [30,31], pruning [32,33], early stopping [34,35], weight decaying [36,37] etc. Furthermore, many researched proposed as a solution the evolution of the architecture of neural networks using some programming techniques. For example the Genetic Algorithms were proposed in a series of papers [38,39] to create the architecture of neural networks or the PSO method [40]. Additionally, Siebel at al. introduced the usage of evolutionary reinforcement learning for the optimal design of artificial neural networks [41]. Also, a review on the usage of Reinforcement learning for neural architecture search is provided by Jaafra et al. [42]. Recently, Pham et al introduced a method for the optimal identification of the architecture of neural networks with parameters sharing [43]. Similarly, Xie et al proposed the incorporation of Stochastic Neural Architecture search [44] for the construction of the architecture of neural networks. Also, Zhou et al used a Bayesian approach for neural architecture search [45].

In this paper, a three - stage technique is proposed, which aims, on the one hand, at the effective training of artificial neural networks and, on the other, at limiting the phenomenon of overifitting. In the first phase of the new method, a genetic algorithm is used to make an initial estimate of the value interval for the parameters of the artificial neural network. In this genetic algorithm, a modified version of the training error of the artificial neural network is used in order to avoid the phenomenon of the network parameters taking large values and consequently the phenomenon of overifitting. In the second phase of the process, an interval technique using a genetic algorithm is utilized to locate the optimal interval for the parameters of the artificial neural network using the best chromosome obtained by the algorithm of the first phase. Finally, in the third phase a simple genetic algorithm is incorporated to train the artificial neural network using the bounds located in the second phase of the method. The proposed technique was tested on a series of classification and regression problems from various research fields and it was compared against traditional methods used for the training of neural networks.

The proposed method consists of three distinct stages which have as their ultimate goal to improve the generalization ability of artificial neural networks. In the first phase, an estimate of the initial values for the value intervals of the parameters of the artificial neural network is made. In this phase, the use of sigmoid functions is taken into account and the observation that they can lose their generalization abilities when the inputs to them exceed certain values in absolute value. For this reason, the parameters of the artificial neural network are limited in such a way that the inputs of the problem are also taken into account. In the second phase, an interval technique undertakes to identify a reliable value interval for the parameters of the artificial neural network, using the information from the first phase, and finally in the third phase, an optimization method, such as a genetic algorithm, can be used for the final phase of training the parameters of the artificial neural network.

The remaining of this article have the following sections: The section 2 describes the proposed method, the section 3 outlines the experimental datasets and the series of experiments conducted and final the section 4 presents some conclusions.

2. Method description

This section describes in details the three distinct algorithms used in every stage of the proposed method.

2.1. The algorithm of the first stage

The activation function used commonly in neural networks is the sigmoid function defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{5}$$

An example plot for this function is outlined in Figure 1.

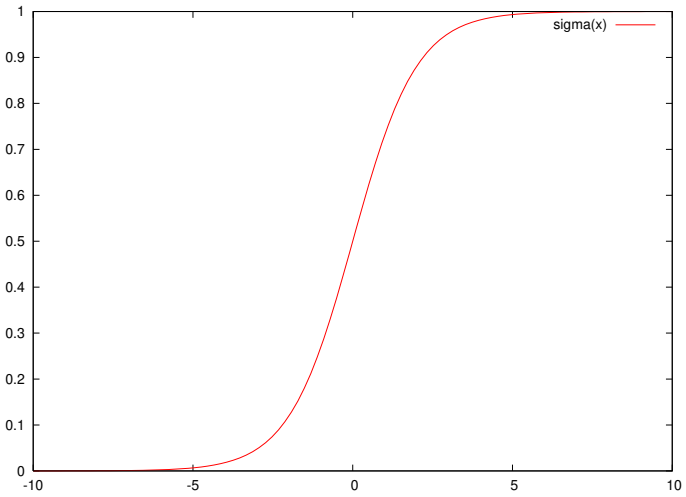

Figure 1. An example plot of the sigmoid function in the range $[-10, 10]$. The function tends quickly to 0 as $x \to -\infty$ and moves quickly to 1 as $x \to \infty$.

As it is clearly shown in this figure, the sigmoid function very quickly takes on the value 1 as x goes towards infinity and very quickly takes on the value 0 as x goes towards minus infinity. This behavior has the direct result that the function loses its generalization capabilities very quickly and these are limited to a small range of values.

The sigmoid function converges very fast to 1 as $x \to +\infty$ and to 0 as $x \to -\infty$. The consequence of this effect is that the corresponding computing unit has the same output for different input values and different values for the computational nodes and hence the corresponding unit. Based on the previous consideration one can define the function $B\left(N\left(\overrightarrow{x}, \overrightarrow{w}\right), a\right)$, which stands for the percentage of how many times the absolute value of the input argument of the sigmoid units exceeds a limit a . This function can be used to avoid the phenomenon of overfitting by limiting the values of the parameters of the artificial neural network to specified intervals which will also directly depend on the inputs presented to the sigmoid functions. This function is described in Algorithm 1. The parameter a is used as a heuristic measure for the value that is input to the sigmoid function. If the input to the function is greater, in absolute values, than this parameter, then we can consider that the sigmoid function loses any generalization abilities, as now the result of the sigmoid will be the same (0 or 1) regardless of changes in its input.

Algorithm 1 Calculating the quantity $B\left(N\left(\overrightarrow{x}, \overrightarrow{w}\right), a\right)$ with $a > 0$ for a a provided neural network $N(x, w)$.

1. **Function** $B\left(N\left(\overrightarrow{x}, \overrightarrow{w}\right), a\right)$
2. **Inputs**: The neural network $N\left(\overrightarrow{x}, \overrightarrow{w}\right)$ and the bound factor $a > 0$.
3. **Set** $k = 0$
4. **For** $i = 1..H$ **Do**
   (a) **For** $j = 1..M$ **Do**
      i. Set $v = \sum_{k=1}^{d}\left(w_{(d+2)i-(d+i)+k}x_{jk}\right) + w_{(d+2)i}$
      ii. **If** $|v| > a$ **then** $k = k + 1$
   (b) **EndFor**
5. **EndFor**
6. **Return** $\frac{k}{H \star M}$
7. **End Function**

The steps of the used algorithm are outlined below:

1. **Initialization step**.
   (a) **Set** N_g as the maximum number of allowed generations.
   (b) **Set** N_c the number of used chromosomes. Each chromosome is considered as a vector of $n = (d+2)H$ double precision values. The value d is used to represent the dimension of the input pattern and the constant H defines the number of the nodes of the neural network. Every value in the chromosomes is initialized randomly in the range $[-I_0, I_0]$, $I_0 > 0$.
   (c) **Set** p_s the selection rate, where $p_s \leq 1$.
   (d) **Set** p_m the mutation rate, where $p_m \leq 1$.
   (e) **Set** $k = 0$ the generation counter.
2. **Fitness calculation step**.
   (a) **For** $i = 1, \ldots, N_c$ **do**
      i. **Create** the neural network $N_i(\overrightarrow{x}, \overrightarrow{g_i})$ for the chromosome $\overrightarrow{g_i}$.
      ii. **Set** $E_i = \sum_{j=1}^{M} \left(N(\overrightarrow{x_j}, \overrightarrow{g_i}) - y_j\right)^2$
      iii. **Set** $b_i = B(N(\overrightarrow{x}, \overrightarrow{g_i}), a)$ using the function of Algorithm 1.
      iv. **Set** $f_i = E(N(\overrightarrow{x}, \overrightarrow{g_i})) \times \left(1 + \lambda b_i^2\right)$ as value for the fitness of chromosome g_i, with $\lambda > 1$.
   (b) **End For**
3. **Application of genetic operations**.
   (a) **Copy** the best $(1 - p_s) \times N_c$ chromosomes of the current population intact to the next generation. The remaining of chromosomes will be replaced by new chromosomes produced during crossover and mutation.
   (b) **Perform** the crossover procedure. For each new element, two parents $z = (z_1, z_2, ..., z_n)$, $w = (w_1, w_2, ..., w_n)$ are selected from the current population using using tournament selection. After the selection of the parents, the new offsprings $\tilde{z}$ and $\tilde{w}$ are formed using the following:

$$\begin{aligned} \tilde{z_i} &= r_i z_i + (1 - r_i) w_i \\ \tilde{w_i} &= r_i w_i + (1 - r_i) z_i \end{aligned} \tag{6}$$

   where r_i are random numbers in $[-0.5, 1.5]$ [46].
   (c) **Perform** the mutation procedure, as proposed in [47]: For every chromosome and each element select a random number $r \in [0, 1]$. If $r \leq p_m$ alter the corresponding element g_{ij} as

$$g_{ij} = \begin{cases} g_{ij} + \Delta(\mathrm{t}, b_{g,i} - g_{ij}) & t = 0 \\ g_{ij} - \Delta(\mathrm{t}, g_{ij} - a_{g,i}) & t = 1 \end{cases} \tag{7}$$

   The number t is a random number that can be 0 or 1 and the function $\Delta(\mathrm{t}, y)$ is calculated as:

$$\Delta(t, y) = y\left(1 - \omega^{\left(1 - \frac{t}{N_t}\right)z}\right) \tag{8}$$

   where $\omega \in [0, 1]$ is a random number and z is parameter defined from the user.
4. **Termination check step**.
   (a) **Set** $k = k + 1$
   (b) **If** $k < N_g$ go to Fitness Calculation step.
5. **Final Step**.
   (a) **Get** the chromosome g^* having the lowest fitness value in the population.

(b) **Produce** the vectors $L^*$and R^* using the following:

$$\begin{aligned} L_i^* &= -f\left|g_i^*\right|,\ i=1,\ldots,n \\ R_i^* &= f\left|g_i^*\right|,\ i=1,\ldots,n \end{aligned}$$

where $f > 1$. These vectors will be used in the following phase of the proposed algorithm.

2.2. The algorithm of the second stage

In the second phase of the current work, a bound method is applied using the vectors L^* and R^* of the previous stage, to discover the optimal bound for the parameters of the network. During this phase, a modified genetic algorithm is incorporated, where the chromosomes are defined as interval sets $\left[\overrightarrow{L_k}, \overrightarrow{R_k}\right]$. Also, the fitness value for each chromosome is considered as an interval $f = [f_1, f_2]$. The function $D(a, b)$ is introduced here for the comparison of two intervals $a = [a_1, a_2]$ and $b = [b_1, b_2]$. This function is defined as:

$$D(a,b) = \begin{cases} \text{TRUE}, & a_1 < b_1, \text{OR } (a_1 = b_1 \text{ AND } a_2 < b_2) \\ \text{FALSE}, & \text{OTHERWISE} \end{cases} \tag{9}$$

A modified genetic algorithm is incorporated here to locate the most promising interval for the weights of the neural network. This procedure uses the vectors L^* and R^* of the previous algorithm. Each chromosome is considered as a set of intervals, which is initialized randomly inside the vectors L^* and R^*. The steps of the algorithm for the second phase are presented below:

1. **Initialization step**.
   (a) **Set** as N_g the maximum number of allowed generations and as N_c the total number of chromosomes.
   (b) **Set** as p_s the selection rate and as p_m the mutation rate.
   (c) **Initialize** every chromosomes$g_i = \left[\overrightarrow{L_i}, \overrightarrow{R_i}\right]$, $i = 1, \ldots, N_c$ randomly inside the produced vectors $L^*$and R^* from the previous phase.
   (d) **Set** as N_s the number of samples used in the fitness calculation step.
   (e) **Set** $k = 0$, the generation counter.
2. **Fitness calculation step**.
   (a) **For** $i = 1, \ldots, N_c$ **do**
      i. **Calculate** the fitness f_i of each chromosome g_i using the procedure provided in Algorithm 2.
   (b) **End For**
3. **Application of genetic operators**.
   (a) Selection procedure. Copy the best $(1 - p_s) \times N_c$ chromosomes to the next generation without changes. The remaining ones will be replaced by offsprings created using the crossover and the mutation procedure. The sorting is performed using the operator $D(a, b)$ for the fitness values.
   (b) Crossover procedure. Perform the crossover procedure, where for every couple $(\widetilde{z}, \widetilde{w})$ of produced chromosomes, two parents (z, w)will be chosen using tournament selection. The new chromosomes will be produced using the one - point crossover method, graphically presented in Figure 2.
   (c) Mutation procedure. For each element of each chromosome a random number $r \in [0, 1]$ is drawn. The corresponding element is altered randomly when $r \leq p_m$.

4. **Termination check step**.
   (a) **Set** $k = k + 1$
   (b) **If** $k < N_g$ go to Fitness calculation step.
5. **Final step**.
   (a) **Obtain** the best chromosome from the population g^*
   (b) **Produce** the corresponding set of intervals $\left[\overrightarrow{L^*}, \overrightarrow{R^*}\right]$

Algorithm 2 Fitness calculation function.

1. **function** fitness(g, N_s)
2. **Input**: The chromosome $g = \left[\overrightarrow{L_g}, \overrightarrow{R_g}\right]$ and the number of random samples N_s.
3. **Draw** N_s random samples in g and create the set $S_a = \left\{\overrightarrow{s_1}, \overrightarrow{s_2}, \ldots, \overrightarrow{s_{N_s}}\right\}$.
4. **Set** $f_{\min} = \infty$
5. **Set** $f_{\max} = -\infty$
6. **For** $i = 1, \ldots, N_s$ **do**
   (a) **Set** $E_i = \sum_{j=1}^{M} \left(N\left(\overrightarrow{x_j}, \overrightarrow{s_i}\right) - y_j\right)^2$
   (b) **If** $E_i < f_{\min}$ set $f_{\min} = E_i$
   (c) **If** $E_i > f_{\max}$ set $f_{\max} = E_i$
7. **End For**
8. **Return** as fitness value the quantity $f_g = [f_{\min}, f_{\max}]$
9. **End Function**

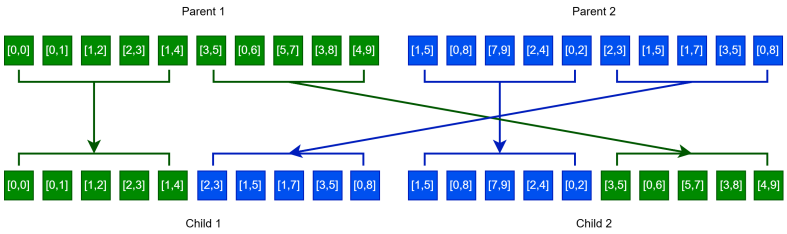

Figure 2. An example of the one - point crossover procedure.

2.3. The final training algorithm

During the final step of the proposed method, a genetic algorithm is incorporated to train the artificial neural network inside the bounds $\left[\overrightarrow{L^*}, \overrightarrow{R^*}\right]$ produced in the final step of the previous phase of the method. The main steps of this algorithm are listed below:

1. **Initialization step**.
   (a) **Set** as N_g the maximum number of allowed generations and as N_c the total number of chromosomes.
   (b) **Set** as p_s the selection rate and as p_m the mutation rate.
   (c) **Initialize** randomly the chromosomes g_i, $i = 1, \ldots, N_c$ as random vectors with $n = (d + 2)H$ elements inside the bounds $\left[\overrightarrow{L^*}, \overrightarrow{R^*}\right]$.
   (d) **Set** $k = 0$, the generation counter.
2. **Fitness calculation step**.
   (a) **For** $i = 1, \ldots, N_c$ **do**
      i. **Create** the neural network $N_i\left(\overrightarrow{x}, \overrightarrow{g_i}\right)$ for the chromosome $\overrightarrow{g_i}$.
      ii. **Calculate** the associated fitness value f_i as $f_i = \sum_{j=1}^{M} \left(N\left(\overrightarrow{x_j}, \overrightarrow{g_i}\right) - y_j\right)^2$
   (b) **End For**

3. **Incorporation of genetic operators.** Apply the same genetic operators as in the first phase of the proposed algorithm, described in subsection 2.1.
4. **Termination check step**.
   (a) **Set** $k = k + 1$
   (b) **If** $k < N_g$ go to Fitness Calculation step of the current algorithm.
5. **Testing step**.
   (a) **Obtain** the chromosome with the lowest fitness value in the population and denote it as g^*.
   (b) **Produce** the associated neural network $N\left(\vec{x}, \vec{g^*}\right)$
   (c) **Apply** a local search procedure to the error function for this network. The local search procedure used was a BFGS method of Powell [48].
   (d) **Apply** the neural network on the associated test set of the problem to obtain the test error.

3. Results

The validation of the proposed method was performed using a wide series of classification and regression datasets, available from various sources from the Internet. These datasets were downloaded from:

1. The UCI database, https://archive.ics.uci.edu/(accessed on 30 April 2025)[49]
2. The Keel website, https://sci2s.ugr.es/keel/datasets.php(accessed on 30 April 2025)[50].
3. The Statlib URL https://lib.stat.cmu.edu/datasets/index(accessed on 30 April 2025).

3.1. Experimental datasets

The following datasets were utilized in the conducted experiments:

1. **Appendictis** which is a medical dataset [51].
2. **Alcohol**, which is dataset regarding alcohol consumption [52].
3. **Australian**, which is a dataset produced from various bank transactions [53].
4. **Balance** dataset [54], produced from various psychological experiments.
5. **Cleveland**, a medical dataset which was discussed in a series of papers [55,56].
6. **Circular** dataset, which is an artificial dataset.
7. **Dermatology**, a medical dataset for dermatology problems [57].
8. The **Hayes-roth** dataset, which was initially suggested in [58].
9. **Heart**, which is a dataset related to heart diseases [59].
10. **HeartAttack**, which is related to heart diseases
11. **Housevotes**, a dataset which contains data from the Congressional voting in USA [60].
12. **Ionosphere**, which is related to measurements derived from the ionosphere [61,62].
13. **Liverdisorder**, a medical dataset [63,64].
14. The **Lymography** dataset [65].
15. **Mammographic**, which is related to the presence of breast cancer [66].
16. **Parkinsons**, which is a medical dataset used for the detection of Parkinson's disease [67,68].
17. **Pima**, which is related to the presence of diabetes[69].
18. **Popfailures**, a dataset related to experiments regarding climate [70].
19. **Regions2**, a medical dataset applied to liver problems [71].
20. **Saheart**, which is a medical dataset concerning heart diseases[72].
21. **Segment** dataset [73].
22. The Sonar dataset, related to sonar signals [74].

23. **Statheart**, a medical dataset related to heart diseases.
24. **Spiral**, which was created artificially and contains two distinct classes.
25. **Student**, which is a dataset regarding experiments in schools [75].
26. **Transfusion**, which is also a dataset used for medical purposes [76].
27. **Wdbc**, which is used for the detection of breast cancer [77,78].
28. **Wine**, a dataset used to detect the quality of wines [79,80].
29. **EEG**, which is a dataset regarding EEG recordings [81,82] and from this dataset the following cases were used: Z_F_S, ZO_NF_S, ZONF_S and Z_O_N_F_S.
30. **Zoo**, which is a dataset regarding animal classification [83] .

Moreover a series of regression datasets was adopted in the performed experiments. The list with the regression datasets has as follows:

1. **Abalone**, which is a dataset for the detection of the age of abalones [84].
2. **Airfoil**, founded in NASA [85].
3. **Auto**, a dataset used to predict the fuel consumption in cars.
4. **BK**, which is used to predict the points scored in basketball games.
5. **BL**, a dataset that contains measurements from electricity experiments.
6. **Baseball**, which is a dataset used to predict the income of baseball players.
7. **Concrete**, which is a civil engineering dataset [86].
8. **DEE**, a dataset that is used to predict the price of electricity.
9. **Friedman**, which is an artificial dataset[87].
10. **FY,** which is a dataset regarding the longevity of fruit flies.
11. **HO**, a dataset located in the STATLIB repository.
12. **Housing**, regarding the price of houses [88].
13. **Laser**, which is used in physics experiments.
14. The **MB** dataset, originated in the Smoothing Methods in Statistics.
15. The **NT** dataset[89].
16. **Mortgage**, a dataset that contains data from the economy of USA.
17. **PL** dataset, located in the STALIB repository.
18. **Plastic**, a dataset regarding problems occurred with the pressure on plastics.
19. The **PY** dataset [90].
20. **Quake**, a dataset regarding the measurements of earthquakes.
21. **SN**, a dataset related to trellising and pruning.
22. **Stock**, which related to the prices of stocks.
23. **Treasury**, a dataset that contains measurements from the economy of USA.

3.2. Experimental results

The software used in the experiment was coded in C++ with the assistance of the freely available Optimus environment [91]. Each experiment was conducted 30 times and in every execution a different seed for the random generator was used. For the validation of the experimental results, the ten - fold cross validation technique was used. The average classification error, as measured in the corresponding test set was reported for the classification datasets. The classification error is computed using the following formula:

$$E_C\left(N\left(\overrightarrow{x},\overrightarrow{w}\right)\right) = 100 \times \frac{\sum_{i=1}^{N}\left(\text{class}\left(N\left(\overrightarrow{x_i},\overrightarrow{w}\right)\right) - y_i\right)}{N} \tag{10}$$

For the calculation of this error, the test set T is a set $T = (x_i, y_i)$, $i = 1, \ldots, N$ is used. Similarly, the average regression error is reported for the regression datasets and it is calculated as follows:

$$E_R\left(N\left(\vec{x}, \vec{w}\right)\right) = \frac{\sum_{i=1}^{N}\left(N\left(\vec{x_i}, \vec{w}\right) - y_i\right)^2}{N} \tag{11}$$

Table 1 contains the values used for the experimental parameters of the proposed method. The results obtained for the classification datasets are depicted in Table 2 and for the regression datasets in Table 3. The following notations were used for the experimental tables:

1. The column DATASET is used to denote the name of the dataset.
2. The column BFGS represents the results obtained by the training of a neural network with $H = 10$ processing nodes using the BFGS optimization method [48] . This method terminates either when the derivative is zero or when a maximum number of iterations is reached. In the experiments performed, this number was set to 2000.
3. The column ADAM is used to denote the training of a neural network with $H = 10$ processing nodes using the ADAM optimization method [19]. The parameters used for the conducted experiments were the following:$b_1 = 0.9$, $b_2 = 0.999$ and the maximum number of iterations was set to 10000.
4. The column NEAT represents the incorporation of the NEAT method (NeuroEvolution of Augmenting Topologies) [92]. The population size was set to 500, as in the case of the proposed method.
5. The column RBF is used to denote the usage of a Radial Basis Function (RBF) network [93,94] with 10 processing nodes. The network was trained with the original training method incorporated in RBF networks with two distinct phases: during the first phase the centers and the variances of the model were calculated using the k-means algorithm [95] and during the second phase the weights of the network was obtained by solving a linear system of equations.
6. The column GENETIC denotes the usage of a genetic algorithm to train a neural network with $H = 10$ processing nodes. The parameters used in this algorithm are listed in Table 1.
7. The column PROPOSED denotes experimental results of the proposed method.
8. The row AVERAGE represents the average classification or regression error for all datasets.

Table 1. The values for the parameters of the proposed method.

PARAMETER	MEANING	VALUE
N_c	Chromosomes	500
N_g	Maximum number of generations	200
p_S	Selection rate	0.1
p_M	Mutation rate	0.05
H	Number of nodes	10
I_0	Initialization factor	10.0
a	Bounding factor	10.0
f	Scale factor for the margins	2.0
λ	Value used for penalties	100.0

The Table 2 presented shows the error rates resulting from the incorporation of the mentioned machine learning models on the used classification datasets. The columns refer to the models (BFGS, ADAM, NEAT, RBF, GENETIC, PROPOSED), while the rows correspond to the datasets. From the analysis of the data, it is observed that the PROPOSED

model exhibits the lowest error rates in many datasets, such as "HouseVotes" (3.05%), "Dermatology" (5.97%), and "ZONF_S" (2.35%). Furthermore, it has the lowest average error rate (19.49%) when a comparison is made against to the other models, indicating overall superior performance. The NEAT model shows the highest error rates in several cases, such as "Cleveland" (77.55%) and "Segment" (68.97%), while ADAM and BFGS exhibit high errors in datasets like "Z_F_S" (47.81% and 39.37%, respectively). However, ADAM has a slightly better average error rate (33.73%) compared to other traditional models like BFGS (33.50%) and NEAT (32.77%). The GENETIC model, although competitive in some datasets like "Hayes Roth" (56.18%) and "Z_O_N_F_S" (64.81%), has a higher average error rate (25.68%) compared to RBF (28.54%). RBF, though not the best in terms of accuracy, demonstrates balanced performance in many cases, with low error rates in datasets such as "Popfailures" (7.04%) and "HouseVotes" (6.13%). Overall, the PROPOSED model stands out as the most effective, with the lowest average error rates and exceptional performance across multiple dataset categories, making it the preferred choice for classification tasks.

Table 2. Experimental results using the incorporated machine learning methods on the classification datasets. The numbers in cells represent average classification error for the test set.

DATASET	BFGS	ADAM	NEAT	RBF	GENETIC	PROPOSED
Alcohol	41.50%	57.78%	66.80%	49.38%	39.57%	26.24%
Appendicitis	18.00%	16.50%	17.20%	12.23%	18.10%	14.90%
Australian	38.13%	35.65%	31.98%	34.89%	32.21%	31.64%
Balance	8.64%	7.87%	23.14%	33.42%	8.97%	7.80%
Cleveland	77.55%	67.55%	53.44%	67.10%	51.60%	47.51%
Circular	6.08%	19.95%	35.18%	5.98%	5.99%	5.42%
Dermatology	52.92%	26.14%	32.43%	62.34%	30.58%	5.97%
Hayes Roth	37.33%	59.70%	50.15%	64.36%	56.18%	39.28%
Heart	39.44%	38.53%	39.27%	31.20%	28.34%	16.85%
HeartAttack	46.67%	45.55%	32.34%	29.00%	29.03%	23.77%
HouseVotes	7.13%	7.48%	10.89%	6.13%	6.62%	3.05%
Ionosphere	15.29%	16.64%	19.67%	16.22%	15.14%	8.75%
Liverdisorder	42.59%	41.53%	30.67%	30.84%	31.11%	29.53%
Lymography	35.43%	29.26%	33.70%	25.50%	28.42%	17.17%
Mammographic	17.24%	46.25%	22.85%	21.38%	19.88%	16.45%
Parkinsons	27.58%	24.06%	18.56%	17.41%	18.05%	17.46%
Pima	35.59%	34.85%	34.51%	25.78%	32.19%	27.25%
Popfailures	5.24%	5.18%	7.05%	7.04%	5.94%	4.66%
Regions2	36.28%	29.85%	33.23%	38.29%	29.39%	25.88%
Saheart	37.48%	34.04%	34.51%	32.19%	34.86%	31.59%
Segment	68.97%	49.75%	66.72%	59.68%	57.72%	42.43%
Sonar	25.85%	30.33%	34.10%	27.90%	22.40%	19.30%
Spiral	47.99%	48.90%	50.22%	44.87%	48.66%	44.67%
Statheart	39.65%	44.04%	44.36%	31.36%	27.25%	18.90%
Student	7.14%	5.13%	10.20%	5.49%	5.61%	4.33%
Transfusion	25.84%	25.68%	24.87%	26.41%	24.87%	23.60%
Wdbc	29.91%	35.35%	12.88%	7.27%	8.56%	8.69%
Wine	59.71%	29.40%	25.43%	31.41%	19.20%	7.27%
Z_F_S	39.37%	47.81%	38.41%	13.16%	10.73%	5.33%
Z_O_N_F_S	65.67%	78.79%	77.08%	48.70%	64.81%	53.15%
ZO_NF_S	43.04%	47.43%	43.75%	9.02%	21.54%	5.82%
ZONF_S	15.62%	11.99%	5.44%	4.03%	4.36%	2.35%
ZOO	10.70%	14.13%	20.27%	21.93%	9.50%	6.07%
AVERAGE	**33.50%**	**33.73%**	**32.77%**	**28.54%**	**25.68%**	**19.49%**

Executions were carried out using scripts in the R language, based on the experimental measurement tables, to determine the significance levels of the experiments using the critical parameter p. In Figure 3, the significance levels are presented, referring to classification datasets and comparing the performance of the PROPOSED model with various machine learning models. The comparisons include the following cases: PROPOSED vs BFGS with p=**** (Very extremely significant), PROPOSED vs ADAM with p=****, PROPOSED vs NEAT with p=****, PROPOSED vs RBF with p=****, and PROPOSED vs GENETIC with p=****. The results provide a clear assessment of the statistical significance of the differences in performance between the PROPOSED model and the other models. The lower the p-value, the stronger the indication that the observed difference in performance is not due to random factors but reflects the genuine superiority of the PROPOSED model.

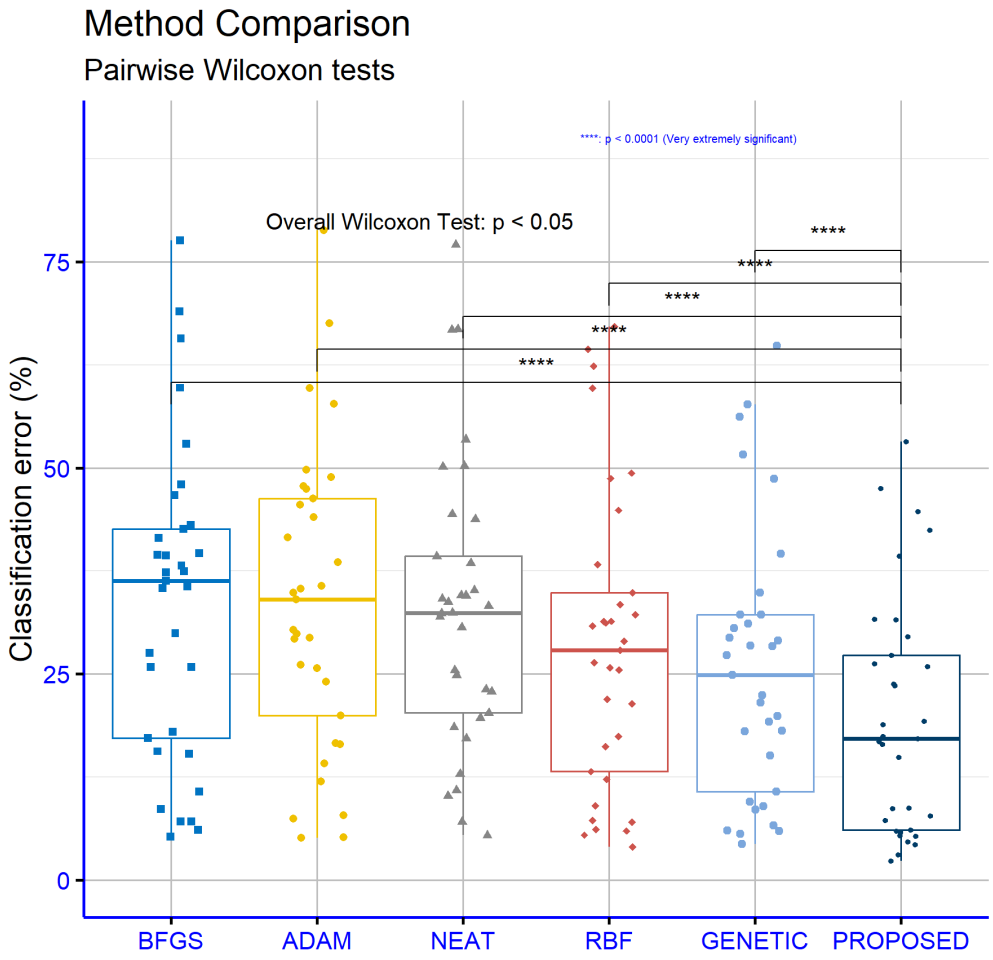

Figure 3. Comprehensive statistical evaluation of the performance of different machine learning algorithms on benchmark classification datasets.

As an example of a comparison in terms of the drop in error in the control set between the proposed method and the simple genetic algorithm, consider Figure 4.

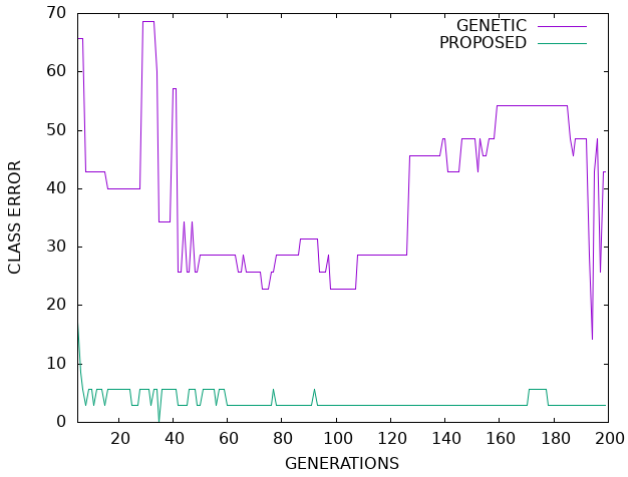

Figure 4. Example of the execution progress of the genetic algorithm and the proposed method for the classification problem of Dermatology. The figure outlines the classification error as calculated on the test set.

As can be seen from the figure above, the proposed method has a lower error in the control set compared to the simple genetic algorithm and furthermore presents very small fluctuations in the value of this specific error.

The Table 3 presents the obtained regression errors from the usage of different machine learning models on regression datasets. The columns refer to the models (BFGS, ADAM, NEAT, RBF, GENETIC, PROPOSED), while the rows correspond to the datasets. From the analysis of the data, it is evident that the PROPOSED model managed to achieve the lowest average error (5.33). This model exhibits exceptionally low errors in datasets such as "BL" (0.001), "HO" (0.012), and "Concrete" (0.004). The RBF model ranks second in performance, with an average error of 9.19, and stands out for its low values in datasets like "Laser" (0.03) and "PY" (0.012). The GENETIC model, with an average error of 8.1, demonstrates competitive performance in certain datasets like "Plastic" (2.79) and "Treasury" (2.93), but generally falls short compared to PROPOSED and RBF. The NEAT model has a higher average error (12.84), though it performs well in datasets such as "Stock" (12.23). The ADAM and BFGS models exhibit the highest average errors, 19.62 and 26.43 respectively, indicating less reliable overall performance. However, ADAM performs well in datasets like "BK" (0.03) and "FY" (0.038), while BFGS achieves good values in specific datasets like "Airfoil" (0.003). Overall, the PROPOSED model significantly outperforms the others across most datasets, showcasing the best overall accuracy. RBF and GENETIC are also reliable in specific cases, while ADAM and BFGS, although less competitive, deliver good results in certain datasets.

Table 3. Experimental results using the incorporated machine learning methods on the regression datasets. The numbers in cells represent average regression on the test set.

DATASET	BFGS	ADAM	NEAT	RBF	GENETIC	PROPOSED
Abalone	5.69	4.30	9.88	7.37	7.17	4.42
Airfoil	0.003	0.005	0.067	0.27	0.003	0.003
Auto	60.97	70.84	56.06	17.87	12.18	12.10
Baseball	119.63	77.90	100.39	93.02	103.60	79.30
BK	0.28	0.03	0.15	0.02	0.03	0.017
BL	2.55	0.28	0.05	0.013	5.74	0.001
Concrete	0.066	0.078	0.081	0.011	0.0099	0.004
Dee	2.36	0.630	1.512	0.17	1.013	0.21
Housing	97.38	80.20	56.49	57.68	43.26	20.74
Friedman	1.26	22.90	19.35	7.23	1.249	3.569
FA	0.426	0.11	0.19	0.015	0.025	0.011
FY	0.22	0.038	0.08	0.041	0.65	0.038
HO	0.62	0.035	0.169	0.03	2.78	0.012
Laser	0.015	0.03	0.084	0.03	0.59	0.004
MB	0.129	0.06	0.061	2.16	0.051	0.048
Mortgage	8.23	9.24	14.11	1.45	2.41	0.85
NT	0.129	0.12	0.33	8.14	0.006	0.006
PL	0.29	0.117	0.098	2.12	0.28	0.022
Plastic	20.32	11.71	20.77	8.62	2.79	2.20
PY	0.578	0.09	0.075	0.012	0.564	0.016
Quake	0.42	0.06	0.298	0.07	0.12	0.037
SN	0.40	0.026	0.174	0.027	2.95	0.024
Stock	302.43	180.89	12.23	12.23	3.88	3.25
Treasury	9.91	11.16	15.52	2.02	2.93	1.11
AVERAGE	**26.43**	**19.62**	**12.84**	**9.19**	**8.10**	**5.33**

The figure 5 displays the results of statistical significance tests conducted on regression datasets, aiming to evaluate the statistical significance of performance differences between the proposed method (PROPOSED) and other machine learning methods. The p-values obtained from the statistical tests are extremely low, indicating strongly statistically significant differences: for the comparison PROPOSED vs BFGS, the p-value is ***, for the comparison PROPOSED vs ADAM, the p-value is ***, for the comparison PROPOSED vs NEAT, the p-value is ****, for the comparison PROPOSED vs RBF, the p-value is *** and for the comparison PROPOSED vs GENETIC, the p-value is ***. These findings demonstrate that the proposed method does not differ randomly from the other methods but exhibits statistically significant superiority in performance. The presence of three or four asterisks indicates that the differences are at least highly significant, meaning that the probability of the observed differences being due to chance is less than 0.1%.

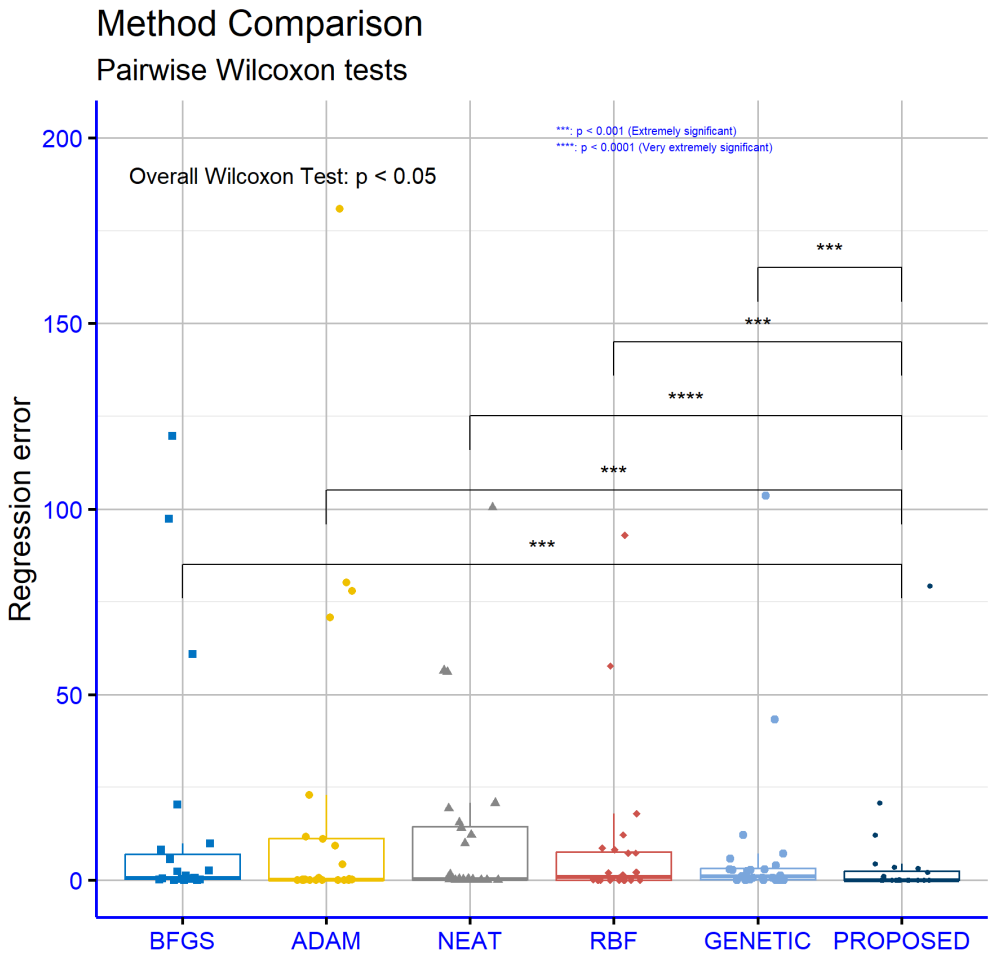

Figure 5. Detailed statistical assessment of the experimental performance of machine learning algorithms on a range of regression datasets.

Furthermore, to clarify the effect of the number of generations parameter on the speed of the method, another experiment was performed, in which the number of generations was gradually increased from 50 to 400 and the execution times were compared between the simple genetic algorithm method and the proposed procedure. The results are presented graphically in Figure 6.

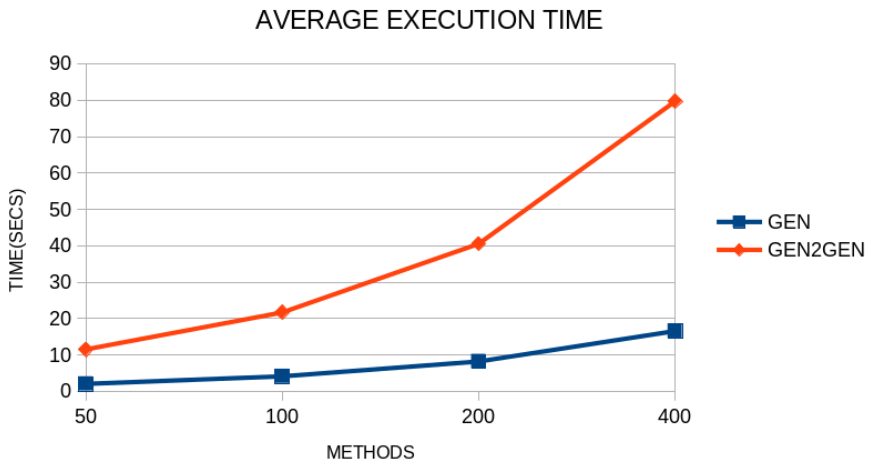

Figure 6. Average execution time and comparison between the original genetic algorithm and the proposed method.

As expected, the time required by the proposed method increases significantly as the number of generations increases. This is of course because the proposed method consists of a series of genetic algorithms executed in serial. However, the execution time could be significantly reduced by using parallel programming techniques, since genetic algorithms can be parallelized relatively easily.

An addition experiment was performed using a variety of values for the initialization factor I_0 and the regression datasets. The average regression error from this experiment and for each value of I_0 is depicted graphically in Figure 7.

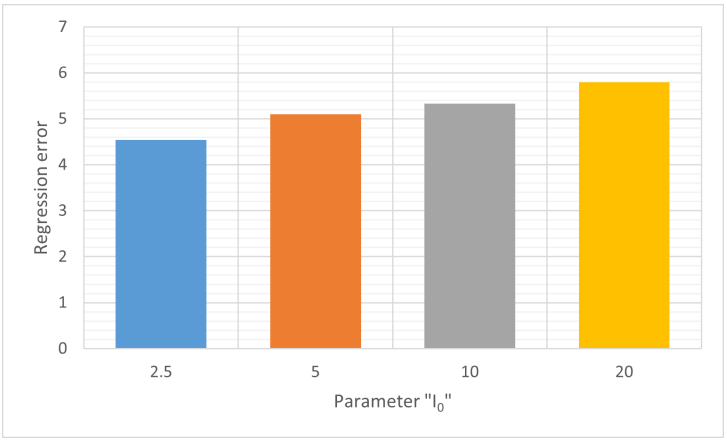

Figure 7. Experiments using different values of the initialization factor I_0 and the used regression datasets.

The obtained regression error remains low for lower value of the initialization factor and increases as this factor gets higher values. This means that initializing the parameters of the neural network in a value interval with smaller extreme values and a smaller range gives the artificial neural network better generalization capabilities.

Also, a similar experiment was conducted using different values of the scale factor f and the utilized regression datasets. The average regression error for this experiment is outlined graphically in Figure 8.

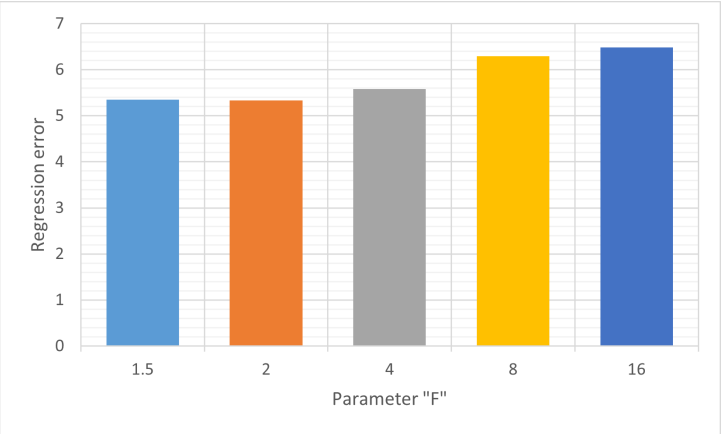

Figure 8. The average regression error obtained by the usage of the proposed method on the regression datasets for a variety of values of the scale factor f.

As it can observed, as the scale factor increases the average regression error increases also. The conclusion from this experiment is that the artificial neural network is able to generalize more efficiently when its parameter values are limited to a smaller range of values than those identified in the first phase.

Furthermore, in order to assess the contribution of the local optimization method BFGS to the performance of the proposed method, an additional experiment was performed where the proposed method was executed without the use of the method BFGS in the final stage. The experimental results from the above experiment are presented in detail in Table 4 for the classification datasets.

Table 4. Experimental results using the genetic algorithm, the proposed method without the incorporation of the BFGS local search method and the proposed method with the addition of the BFGS method. The experiments were conducted on the mentioned classification datasets and the numbers in cells represent average classification error for the test set.

DATASET	GENETIC	PROPOSED(NO BFGS)	PROPOSED
Alcohol	39.57%	26.32%	26.24%
Appendicitis	18.10%	16.00%	14.90%
Australian	32.21%	28.09%	31.64%
Balance	8.97%	7.81%	7.80%
Cleveland	51.60%	46.24%	47.51%
Circular	5.99%	5.51%	5.42%
Dermatology	30.58%	8.83%	5.97%
Hayes Roth	56.18%	42.38%	39.28%
Heart	28.34%	18.37%	16.85%
HeartAttack	29.03%	19.50%	23.77%
HouseVotes	6.62%	3.48%	3.05%
Ionosphere	15.14%	10.03%	8.75%
Liverdisorder	31.11%	30.94%	29.53%
Lymography	28.42%	20.79%	17.17%
Mammographic	19.88%	16.59%	16.45%
Parkinsons	18.05%	16.21%	17.46%
Pima	32.19%	31.11%	27.25%
Popfailures	5.94%	4.61%	4.66%
Regions2	29.39%	25.10%	25.88%
Saheart	34.86%	31.20%	31.59%
Segment	57.72%	40.87%	42.43%
Sonar	22.40%	25.55%	19.30%
Spiral	48.66%	46.13%	44.67%
Statheart	27.25%	17.59%	18.90%
Student	5.61%	3.88%	4.33%
Transfusion	24.87%	22.99%	23.60%
Wdbc	8.56%	8.43%	8.69%
Wine	19.20%	6.53%	7.27%
Z_F_S	10.73%	6.73%	5.33%
Z_O_N_F_S	64.81%	49.68%	53.15%
ZO_NF_S	21.54%	7.52%	5.82%
ZONF_S	4.36%	2.28%	2.35%
ZOO	9.50%	13.90%	6.07%
AVERAGE	**25.68%**	**20.04%**	**19.49%**

As can be seen from the study of the above results, the local optimization method BFGS improves the results of the proposed method in some problems, but on average the classification error of the proposed method remains low compared to a simple genetic algorithm.

3.3. A practical example

As a practical example of application with many patterns, consider the PIRvision dataset, which was presented in 2023 [96]. This dataset contains data from occupancy detection. The associated data were collected from a Synchronized Low-Energy Electronically-chopped Passive Infra-Red sensing node in residential and office environments. The dataset has 15302 patterns and each pattern has 59 features. The following methods were applied on this dataset in the conducted experiments:

1. RBF, which represents the application of the RBF network with 10 processing nodes.

2. BFGS, that stands for the BFGS method which used to train a neural network with $H = 10$ processing nodes.
3. GENETIC, which represents a genetic algorithm incorporated to to train a neural network with $H = 10$ processing nodes.
4. GEN2GEN, that represents the proposed method.

The results were validated using the ten - fold cross validation method and they are presented graphically in Figure 9.

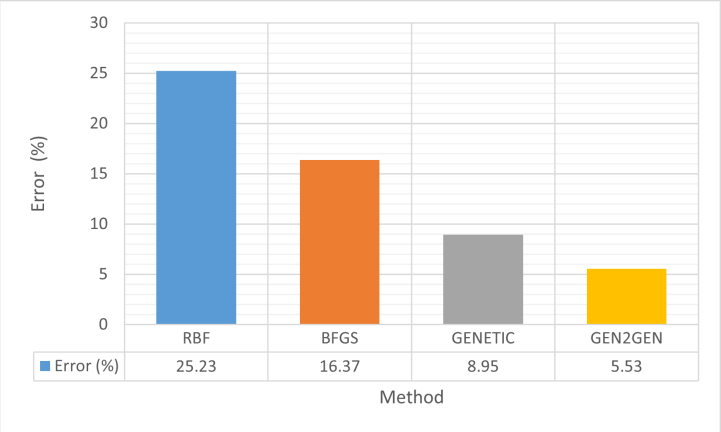

Figure 9. Results obtained for the PIRvision dataset, using a variety of methods and the proposed method.

As is evident from the experimental results, the proposed method significantly reduces the classification error, especially compared to the simple genetic algorithm and is around 5%.

4. Conclusions

The proposed method for training artificial neural networks is based on the application of genetic algorithms in three distinct phases,with the primary objective of efficient training and minimizing overfitting, a common challenge in modern optimization techniques. The first phase focuses on identifying an initial interval for the values of the network. This phase is crucial as it sets the initial positioning of the parameters within a range that avoids excessively large values, which could limit the model's generalization capability. The proposed value range is determined using a genetic algorithm that incorporates a modified error calculation, penalizing large parameter values. This step reduces the risk of overfitting to the training data, enhancing the model's capacity to respond effectively to unseen data. In the second phase, the method employs an optimized genetic algorithm to identify the ideal parameter value bounds within the initially defined range. This process makes the method particularly effective as it focuses on intervals already evaluated as suitable while incorporating representative samples from the initial value range to assess accuracy. The use of genetic algorithms allows for gradual and adaptive improvement, eliminating local minima a frequent issue in traditional optimization methods. The third phase focuses on training the neural network within the optimized parameter bounds. In this phase, a genetic algorithm used to minimize the training error, followed by a local optimization step using the BFGS method. This local optimization ensures further accuracy improvement, fully utilizing the model's potential. The experiments conducted demonstrate the clear superiority of the proposed method compared to other established techniques. For classification datasets, the method achieved significantly lower error rates compared to techniques like ADAM, BFGS, and NEAT. For example, in datasets such as Dermatology and HouseVotes, the error rate was nearly halved compared to alternative

methods. Similar results were observed for regression datasets, where the method achieved the highest accuracy across nearly all datasets. The lowest average error achieved highlights its consistent and versatile performance. A notable innovation of the method is its approach to tackling overfitting. The genetic algorithms enable exploration across a wide range of values without being constrained to local minima. Simultaneously, the incorporation of penalties for large parameter values prevents excessive adaptation to the training data. This is particularly important as overfitting often limits the performance of artificial neural networks when applied to unseen data. Experiments with varying initial parameters, such as the initialization factor and the scale factor, provide valuable insights into model configuration. For instance, smaller initial value ranges contributed to better generalization, while larger scale factor values led to higher error, emphasizing the importance of tighter parameter bounds. This indicates that careful parameter selection is critical for overall performance.

The proposed method paves new paths for the application of genetic algorithms in training process of neural networks. Its adaptive nature makes it suitable for a wide range of applications, from medical diagnosis and forecasting of physical phenomena to the optimization of industrial processes. Future steps could include its application in deep learning networks, the integration of hybrid methods combining genetic algorithms with other optimization techniques, and the use of distributed computing environments to accelerate training. This approach has the potential to become a benchmark for effective and reliable training of artificial neural networks.

Although the proposed method demonstrates clear superiority across a wide range of classification and regression problems, it is important to note that the experiments and analysis focus primarily on relatively simple neural network architectures. While the authors mention the potential application of the method to deep neural networks and hybrid approaches, the article does not provide a thorough discussion of the challenges that may arise when extending the method to such contexts. Deep neural networks, characterized by their multilayered structure and large number of parameters, introduce significant issues related to the stability of the evolutionary process, the efficiency of training, and computational cost. Furthermore, the dynamic interaction between genetic algorithms and other optimization techniques may require specialized adaptations to ensure both effectiveness and scalability when applied to more complex or hybrid environments. Including a brief discussion of these aspects would enhance the completeness of the article, offering a more nuanced perspective on the strengths and limitations of the proposed approach and helping to guide future research efforts towards optimizing the method for use in truly deep and hybrid network architectures.

Author Contributions: V.C. and I.G.T. conducted the experiments, employing several datasets and provided the comparative experiments. V.C. performed the statistical analysis and prepared the manuscript. 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.

Acknowledgments: This research has been financed by the European Union : Next Generation EU through the Program Greece 2.0 National Recovery and Resilience Plan , under the call RESEARCH – CREATE – INNOVATE, project name “iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques" (project code:TAEDK-06195).

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

References

1. Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11).
2. Suryadevara, S., & Yanamala, A. K. Y. (2021). A Comprehensive Overview of Artificial Neural Networks: Evolution, Architectures, and Applications. Revista de Inteligencia Artificial en Medicina, 12(1), 51-76.
3. I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing **72**, pp. 269-277, 2008.
4. S. Guarnieri, F. Piazza, A. Uncini, Multilayer feedforward networks with adaptive spline activation function, IEEE Transactions on Neural Networks **10**, pp. 672-683, 1999.
5. Ö.F. Ertuğrul, A novel type of activation function in artificial neural networks: Trained activation function, Neural Networks **99**, pp. 148-157, 2018.
6. A. D. Rasamoelina, F. Adjailia, P. Sinčák, A Review of Activation Function for Artificial Neural Network, In: 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, pp. 281-286, 2020.
7. M. Egmont-Petersen, D. de Ridder, H. Handels, Image processing with neural networks—a review, Pattern Recognition **35**, pp. 2279-2301, 2002.
8. G.Peter Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing **50**, pp. 159-175, 2003.
9. Z. Huang, H. Chen, C.-Jung Hsu, W.-Hwa Chen, S. Wu, Credit rating analysis with support vector machines and neural networks: a market comparative study, Decision Support Systems **37**, pp. 543-558, 2004.
10. P. Baldi, K. Cranmer, T. Faucett et al, Parameterized neural networks for high-energy physics, Eur. Phys. J. C **76**, 2016.
11. Baldi, P., Cranmer, K., Faucett, T., Sadowski, P., & Whiteson, D. (2016). Parameterized neural networks for high-energy physics. The European Physical Journal C, 76(5), 1-7.
12. Igor I. Baskin, David Winkler and Igor V. Tetko, A renaissance of neural networks in drug discovery, Expert Opinion on Drug Discovery **11**, pp. 785-795, 2016.
13. Ronadl Bartzatt, Prediction of Novel Anti-Ebola Virus Compounds Utilizing Artificial Neural Network (ANN), Chemistry Faculty Publications **49**, pp. 16-34, 2018.
14. K. Peta, J. Żurek, Prediction of air leakage in heat exchangers for automotive applications using artificial neural networks, In: 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, pp. 721-725, 2018.
15. Vora, K., & Yagnik, S. (2014). A survey on backpropagation algorithms for feedforward neural networks. International Journal of Engineering Development and Research, 1(3), 193-197.
16. K. Vora, S. Yagnik, A survey on backpropagation algorithms for feedforward neural networks, International Journal of Engineering Development and Research **1**, pp. 193-197, 2014.
17. Pajchrowski, T., Zawirski, K., & Nowopolski, K. (2014). Neural speed controller trained online by means of modified RPROP algorithm. IEEE transactions on industrial informatics, 11(2), 560-568.
18. Hermanto, R. P. S., & Nugroho, A. (2018). Waiting-time estimation in bank customer queues using RPROP neural networks. Procedia Computer Science, 135, 35-42.
19. D. P. Kingma, J. L. Ba, ADAM: a method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), pp. 1–15, 2015.
20. Reynolds, J., Rezgui, Y., Kwan, A., & Piriou, S. (2018). A zone-level, building energy optimisation combining an artificial neural network, a genetic algorithm, and model predictive control. Energy, 151, 729-739.
21. Das, G., Pattnaik, P. K., & Padhy, S. K. (2014). Artificial neural network trained by particle swarm optimization for non-linear channel equalization. Expert Systems with Applications, 41(7), 3491-3496.
22. Sexton, R. S., Dorsey, R. E., & Johnson, J. D. (1999). Beyond backpropagation: using simulated annealing for training neural networks. Journal of Organizational and End User Computing (JOEUC), 11(3), 3-10.
23. Wang, L., Zeng, Y., & Chen, T. (2015). Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. Expert Systems with Applications, 42(2), 855-863.
24. Karaboga, D., & Akay, B. (2007, June). Artificial bee colony (ABC) algorithm on training artificial neural networks. In 2007 IEEE 15th Signal Processing and Communications Applications (pp. 1-4). IEEE.
25. R.S. Sexton, B. Alidaee, R.E. Dorsey, J.D. Johnson, Global optimization for artificial neural networks: A tabu search application. European Journal of Operational Research **106**, pp. 570-584, 1998.
26. J.-R. Zhang, J. Zhang, T.-M. Lok, M.R. Lyu, A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training, Applied Mathematics and Computation **185**, pp. 1026-1037, 2007.
27. G. Zhao, T. Wang, Y. Jin, C. Lang, Y. Li, H. Ling, The Cascaded Forward algorithm for neural network training, Pattern Recognition **161**, 111292, 2025.
28. K-Su Oh, K. Jung, GPU implementation of neural networks, Pattern Recognition **37**, pp. 1311-1314, 2004.

29. M. Zhang, K. Hibi, J. Inoue, GPU-accelerated artificial neural network potential for molecular dynamics simulation, Computer Physics Communications **285**, 108655, 2023.
30. S.J. Nowlan and G.E. Hinton, Simplifying neural networks by soft weight sharing, Neural Computation 4, pp. 473-493, 1992.
31. Nowlan, S. J., & Hinton, G. E. (2018). Simplifying neural networks by soft weight sharing. In The mathematics of generalization (pp. 373-394). CRC Press.
32. S.J. Hanson and L.Y. Pratt, Comparing biases for minimal network construction with back propagation, In D.S. Touretzky (Ed.), Advances in Neural Information Processing Systems, Volume 1, pp. 177-185, San Mateo, CA: Morgan Kaufmann, 1989.
33. M. Augasta and T. Kathirvalavakumar, Pruning algorithms of neural networks — a comparative study, Central European Journal of Computer Science, 2003.
34. Lutz Prechelt, Automatic early stopping using cross validation: quantifying the criteria, Neural Networks **11**, pp. 761-767, 1998.
35. X. Wu and J. Liu, A New Early Stopping Algorithm for Improving Neural Network Generalization, 2009 Second International Conference on Intelligent Computation Technology and Automation, Changsha, Hunan, 2009, pp. 15-18.
36. N. K. Treadgold and T. D. Gedeon, Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm,IEEE Transactions on Neural Networks **9**, pp. 662-668, 1998.
37. M. Carvalho and T. B. Ludermir, Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay, 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06), Rio de Janeiro, Brazil, 2006, pp. 5-5.
38. J. Arifovic, R. Gençay, Using genetic algorithms to select architecture of a feedforward artificial neural network, Physica A: Statistical Mechanics and its Applications **289**, pp. 574-594, 2001.
39. P.G. Benardos, G.C. Vosniakos, Optimizing feedforward artificial neural network architecture, Engineering Applications of Artificial Intelligence **20**, pp. 365-382, 2007.
40. B.A. Garro, R.A. Vázquez, Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms, Computational Intelligence and Neuroscience, 369298, 2015.
41. Siebel, N. T., & Sommer, G. (2007). Evolutionary reinforcement learning of artificial neural networks. International Journal of Hybrid Intelligent Systems, 4(3), 171-183.
42. Jaafra, Y., Laurent, J. L., Deruyver, A., & Naceur, M. S. (2019). Reinforcement learning for neural architecture search: A review. Image and Vision Computing, 89, 57-66.
43. Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In International conference on machine learning (pp. 4095-4104). PMLR.
44. Xie, S., Zheng, H., Liu, C., & Lin, L. (2018). SNAS: stochastic neural architecture search. arXiv preprint arXiv:1812.09926.
45. Zhou, H., Yang, M., Wang, J., & Pan, W. (2019, May). Bayesnas: A bayesian approach for neural architecture search. In International conference on machine learning (pp. 7603-7613). PMLR.
46. A.A. Huqqani, E. Schikuta, S. Ye Peng Chen, Multicore and GPU Parallelization of Neural Networks for Face Recognition, Procedia Computer Science **18**, pp. 349-358, 2013.
47. P. Kaelo, M.M. Ali, Integrated crossover rules in real coded genetic algorithms, European Journal of Operational Research **176**, pp. 60-76, 2007.
48. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
49. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository, https://archive.ics.uci.edu.
50. J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17, pp. 255-287, 2011.
51. Weiss, Sholom M. and Kulikowski, Casimir A., Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems, Morgan Kaufmann Publishers Inc, 1991.
52. Tzimourta, K.D.; Tsoulos, I.; Bilero, I.T.; Tzallas, A.T.; Tsipouras, M.G.; Giannakeas, N. Direct Assessment of Alcohol Consumption in Mental State Using Brain Computer Interfaces and Grammatical Evolution. Inventions 2018, 3, 51.
53. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
54. T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, Machine Learning **16**, pp. 59-88, 1994.
55. Z.H. Zhou,Y. Jiang, NeC4.5: neural ensemble based C4.5," in IEEE Transactions on Knowledge and Data Engineering **16**, pp. 770-773, 2004.
56. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
57. G. Demiroz, H.A. Govenir, N. Ilter, Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals, Artificial Intelligence in Medicine. **13**, pp. 147–165, 1998.

58. B. Hayes-Roth, B., F. Hayes-Roth. Concept learning and the recognition and classification of exemplars. Journal of Verbal Learning and Verbal Behavior **16**, pp. 321-338, 1977.
59. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
60. R.M. French, N. Chater, Using noise to compute error surfaces in connectionist networks: a novel means of reducing catastrophic forgetting, Neural Comput. **14**, pp. 1755-1769, 2002.
61. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
62. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
63. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
64. J. Mcdermott, R.S. Forsyth, Diagnosing a disorder in a classification benchmark, Pattern Recognition Letters **73**, pp. 41-43, 2016.
65. G. Cestnik, I. Konenenko, I. Bratko, Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. In: Bratko, I. and Lavrac, N., Eds., Progress in Machine Learning, Sigma Press, Wilmslow, pp. 31-45, 1987.
66. M. Elter, R. Schulz-Wendtland, T. Wittenberg, The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process, Med Phys. **34**, pp. 4164-72, 2007.
67. M.A. Little, P.E. McSharry, S.J Roberts et al, Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection. BioMed Eng OnLine **6**, 23, 2007.
68. M.A. Little, P.E. McSharry, E.J. Hunter, J. Spielman, L.O. Ramig, Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Trans Biomed Eng. **56**, pp. 1015-1022, 2009.
69. J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, R.S. Johannes, 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, pp.261-265, 1988.
70. D.D. Lucas, R. Klein, J. Tannahill, D. Ivanova, S. Brandon, D. Domyancic, Y. Zhang, Failure analysis of parameter-induced simulation crashes in climate models, Geoscientific Model Development **6**, pp. 1157-1171, 2013.
71. N. Giannakeas, M.G. Tsipouras, A.T. Tzallas, K. Kyriakidi, Z.E. Tsianou, P. Manousou, A. Hall, E.C. Karvounis, V. Tsianos, E. Tsianos, A clustering based method for collagen proportional area extraction in liver biopsy images (2015) Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, 2015-November, art. no. 7319047, pp. 3097-3100.
72. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
73. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
74. Gorman, R.P.; Sejnowski, T.J. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. Neural Netw. 1988, 1, 75–89.
75. P. Cortez, A. M. Gonçalves Silva, Using data mining to predict secondary school student performance, In Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) (pp. 5–12). EUROSIS-ETI, 2008.
76. I-Cheng Yeh, King-Jang Yang, Tao-Ming Ting, Knowledge discovery on RFM model using Bernoulli sequence, Expert Systems with Applications **36**, pp. 5866-5871, 2009.
77. Jeyasingh, S., & Veluchamy, M. (2017). Modified bat algorithm for feature selection with the Wisconsin diagnosis breast cancer (WDBC) dataset. Asian Pacific journal of cancer prevention: APJCP, 18(5), 1257.
78. Alshayeji, M. H., Ellethy, H., & Gupta, R. (2022). Computer-aided detection of breast cancer on the Wisconsin dataset: An artificial neural networks approach. Biomedical signal processing and control, 71, 103141.
79. M. Raymer, T.E. Doom, L.A. Kuhn, W.F. Punch, Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, **33** , pp. 802-813, 2003.
80. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
81. R. G. Andrzejak, K. Lehnertz, F.Mormann, C. Rieke, P. David, and C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state," Physical Review E, vol. 64, no. 6, Article ID 061907, 8 pages, 2001.
82. A. T. Tzallas, M. G. Tsipouras, and D. I. Fotiadis, "Automatic Seizure Detection Based on Time-Frequency Analysis and Artificial Neural Networks," Computational Intelligence and Neuroscience, vol. 2007, Article ID 80510, 13 pages, 2007. doi:10.1155/2007/80510

83. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
84. Nash, W.J.; Sellers, T.L.; Talbot, S.R.; Cawthor, A.J.; Ford, W.B. The Population Biology of Abalone (_Haliotis_ species) in Tasmania. I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait, Sea Fisheries Division; Technical Report No. 48; Department of Primary Industry and Fisheries, Tasmania: Hobart, Australia, 1994; ISSN 1034-3288
85. Brooks, T.F.; Pope, D.S.; Marcolini, A.M. Airfoil Self-Noise and Prediction. Technical Report, NASA RP-1218. July 1989. Available online: https://ntrs.nasa.gov/citations/19890016302 (accessed on 14 November 2024).
86. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
87. Friedman, J. (1991): Multivariate Adaptative Regression Splines. Annals of Statistics, 19:1, 1--141.
88. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
89. Mackowiak, P.A., Wasserman, S.S., Levine, M.M., 1992. A critical appraisal of 98.6 degrees f, the upper limit of the normal body temperature, and other legacies of Carl Reinhold August Wunderlich. J. Amer. Med. Assoc. 268, 1578–1580
90. R.D. King, S. Muggleton, R. Lewis, M.J.E. Sternberg, Proc. Nat. Acad. Sci. USA **89**, pp. 11322–11326, 1992.
91. I.G. Tsoulos, V. Charilogis, G. Kyrou, V.N. Stavrou, A. Tzallas, Journal of Open Source Software **10**, 7584, 2025.
92. K. O. Stanley, R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, Evolutionary Computation **10**, pp. 99-127, 2002.
93. J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation **3**, pp. 246-257, 1991.
94. G.A. Montazer, D. Giveki, M. Karami, H. Rastegar, Radial basis function neural networks: A review. Comput. Rev. J **1**, pp. 52-74, 2018.
95. MacQueen, J.: Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, No. 14, pp. 281-297, 1967.
96. Emad-Ud-Din, M., & Wang, Y. (2023). Promoting occupancy detection accuracy using on-device lifelong learning. IEEE Sensors Journal, 23(9), 9595-9606.