Article

Training neural networks with a procedure guided by BNF grammars

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 are parametric machine learning models that have been applied successfully to an extended series of classification and regression problems found in the recent literature. For the effective identification of the parameters of the artificial neural networks, a series of optimization techniques have been proposed in the relevant literature, which, although they present good results in many cases, either the optimization method used is not efficient and the training error of the network is trapped in sub-optimal values, or the neural network exhibits the phenomenon of over-training which means that it has poor results when applied to data that was not present during the training. This paper proposes an innovative technique for constructing the weights of artificial neural networks based on appropriate BNF grammars, used in the evolutionary process of Grammatical Evolution. The new procedure locates an interval of values for the parameters of the artificial neural network, and the optimization method effectively locates the network parameters within this interval. The new technique was applied to a wide range of data classification and adaptation problems covering a number of scientific areas and the experimental results were more than promising.

Keywords: Neural networks; Genetic algorithms; Grammatical Evolution; Evolutionary algorithms

Citation: Tsoulos, I.G.; Charilogis, V. Training neural networks with a procedure guided by BNF grammars. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

Many real world problems can be formulated as classification or regression problems and subsequently, the can be tackled by machine learning models. Such problems occur in physics [1,2], chemistry [3,4], economics [5,6], medicine [7,8] etc. A typical representative of machine learning models with widespread use due to their remarkable adaptation capabilities are artificial neural networks [9,10]. This model has been successfully applied to a wide series of practical problems, such as image processing [11], time series forecasting [12], forecast of solar irradiance [13], medical diagnosis [14], solutions of differential equations [15,16], agriculture problems [17,18] etc.

Typically, a neural network is expressed as a function $N(\overrightarrow{x},\overrightarrow{w})$, provided that the vector $\overrightarrow{x}$ represents the input vector or pattern for the neural network and the vector $\overrightarrow{w}$ stands for the set of parameters of the network, that should be calculated. The vector $\overrightarrow{w}$ is also called weight vector. The adaptation of the parameter vector $\overrightarrow{w}$ is performed by minimizing the so-called training error, which is defined as:

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

In equation 1 the set $\left(\overrightarrow{x_i},y_i\right),\ i=1,...,M$ represents the train set of the neural network, where the values y_i are considered as the expected outputs for the $\overrightarrow{x_i}$ patterns. Various methods have been proposed in the relevant literature to train neural networks, such as the Back Propagation method [19], the RPROP method [20], the Adam optimization method [21] etc. Also, a series of modern approaches have also been incorporated to train neural networks, such as the Quasi Newton methods [22], the Tabu Search method [23], Simulated Annealing [24], genetic Algorithms [25], Particle Swarm Optimization (PSO) [26], Differential Evolution [27], Ant Colony Optimization [28]. Recently, Askarzadeh et al suggested the Bird Mating Optimizer (BMO) [29]as the training algorithm for a series of benchmark datasets. Benardos et al suggested the application of a genetic algorithm to obtain the best architecture of a neural network, in order to efficiently train a neural network [30]. Karaboga introduced the usage of the Artificial Bee Colony (ABC) algorithm on training artificial neural networks [31]. Among others, Chen at al [32] has proposed a hybrid method that combines particle swarm optimization and Cuckoo Search [33] to optimize the weight vector. Also, due to the widespread application of parallel programming techniques, a series of papers that exploit such methods are published recently regarding the training of neural networks [34–36].

Another important aspect of the research conducted on artificial neural networks is the initialization techniques used, which play an important role in the final result that these networks have. Such initialization methods include the usage of decision trees [37], incorporation of the Cauchy's inequality [38], application of discriminant learning [39], usage of genetic algorithms [40] etc. Also recently, Sodhi et al proposed an interval based weight initialization method for neural networks [41]. An overview of the methods used for weight initialization can be found in the work of Sousa [42].

However, the most important problem that can be caused by the use of different initialization and/or training techniques for artificial neural networks is the phenomenon of overtraining of these networks, where the network has poor performance on patterns that were not used in the training of the network. This problem was tackled in the recent bibliography by a series of methods, such as the weight sharing technique [43], weight pruning techniques that can reduce the number of weights [44–46], the dropout method [47], weight decaying [48,49], the Sarprop technique [50], positive correlation methods [51] etc.

This paper proposes an innovative process for training artificial neural networks, which consists of three individual stages. During the execution of the first stage, an initial estimate of the value interval of the artificial neural network parameters is made using a smart global optimization technique, such as Genetic Algorithms [52]. The generated result of the first stage is used in the next phase of the algorithm, where a process using Grammatical Evolution [53] is used to more efficiently identify the value interval of the parameters of the artificial neural network. In this phase, the value intervals for the network parameters are constructed using a Backus–Naur (BNF) grammar [54]. In the last stage of the proposed method, the parameters accompanying the artificial neural networks are initialized within the optimal value interval of the previous process and an optimization algorithm is used to effectively train the network within this interval.

The remaining of this article is organized as follows: in section 2 the proposed method is discussed in detail, in section 3 the used datasets as well as the conducted experiments are discussed and finally, in section 4 some conclusions are presented.

2. Materials and Methods

This section presents the basic principles of Grammatical Evolution and then the proposed 3-stage weight adjustment process.

2.1. Grammatical Evolution

Grammatical Evolution is an evolutionary algorithm, where the chromosomes are vectors of positive integer values. Each chromosome is a series of production rules from the underlying BNF grammar and it can be used to construct valid programs in the target language. During the recent years, Grammatical Evolution was applied in a wide range of problems from different areas, such as music synthesis [55], video games [56,57], design of fractal curves [58], constant creation [59], robotics [60,61], modeling glycemia in humans

[62], Wikipedia taxonomies [63], economics [64] etc. The grammars used in the Grammatical Evolution procedure are expressed as sets $G = (N, T, S, P)$ where

- N represents the set of non-terminal symbols included in the grammar.
- T stands for the set of terminal symbols. Every non - terminal symbol is replaced by a series of terminal symbols withe the application of the associated production rules.
- S is a non - terminal symbol, that stands for the start symbol of the grammar.
- P is the set of production rules that are used to create terminal symbols from non - terminal symbols.

For every chromosome, Grammatical Evolution starts from the symbol S and through a series of steps it produces valid programs with terminal symbols by substituting non-terminal symbols with the right hand of the selected production rule. The selection of the production rule is accomplished in two steps:

- Obtain the next element from the under - processing chromosome. Denote this element with V
- The next rule is selected according to: Rule = V mod N_R. The number N_R represents the total number of production rules for the non – terminal symbol that is under consideration.

2.2. The proposed method

2.2.1. The first phase of the method

At the beginning of the process, an initial estimate of the range of values for the parameters of the artificial neural network must be made. This estimate should also reflect the specificities of each data set to be processed and therefore arbitrary values cannot be used. For this reason, an optimization procedure is used to train the neural network. The final result of this process can be used as a good estimate of the range of values for these parameters. In the current work, the global optimization method of Genetic Algorithm was used as the training procedure of the first phase. This method was chosen because of its great flexibility, its many applications in many scientific fields, and its ability to be parallelized. The form of neural network used here was also proposed in [65] and it is defined as follows:

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

In the previous equation the symbol H denotes the number of processing units and the value d represents the the dimension of the input pattern $\overrightarrow{x}$. As a consequence the size of vector $\overrightarrow{w}$ is $n = (d+2)H$. The function $\sigma(x)$ represents the sigmoid function defined as:

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

Each chromosome in this genetic algorithm represents a possible set of parameters for the artificial neural network and during optimization the training error is minimized, as captured in equation 1. The main steps of this procedure have as follows:

1. **Initialization step**.
   (a) **Define** as N_c the number of chromosomes and as N_g the maximum number of generations allowed before termination.
   (b) **Define** as p_s the selection rate with $p_s \leq 1$.
   (c) **Define** as p_m the mutation rate with $p_m \leq 1$.
   (d) **Initialize** randomly the chromosomes $g_i,\ i = 1, \ldots, N_c$.
   (e) **Set** $k = 0$, the generation number.
2. **Genetic operations step**.

   (a) Fitness calculation. For every chromosome $c_i,\ i = 1, \ldots, N_c$ create the corresponding neural network $N(c_i, x)$ and set the corresponding fitness value with the following formula:

$$f_i = \sum_{j=1}^{M} \left(N\left(c_i, x_j\right) - y_j\right)^2$$

   (b) Application of selection. The chromosomes and the corresponding fitness values are sorted according to the fitness and the best $(1 - p_s) \times N_c$ of them are copied intact to the next generation. The rest of the chromosomes are replaced by new chromosomes produced during the crossover and the mutation procedure.
   (c) Application of crossover. During this process, for every pair of constructed chromosomes $(\tilde{z}, \tilde{w})$ two chromosomes (z, w) are selected from the current population using tournament selection. The construction of the new chromosomes is performed using the following:

$$\begin{array}{rcl} \tilde{z_i} & = & a_i z_i + (1 - a_i) w_i \\ \tilde{w_i} & = & a_i w_i + (1 - a_i) z_i \end{array} \qquad (4)$$

   The values a_i are random numbers with $a_i \in [-0.5, 1.5]$ [66].
   (d) Application of mutation. For every element of each chromosomes a random number $r \in [0, 1]$ is drawn. This element is altered randomly when $r \leq p_m$.

3. **Termination check**.
   (a) Set $k = k + 1$
   (b) If $k \leq N_g$ goto step 2
4. **Finalization step**.
   (a) **Select** the chromosome g^* with the lowest fitness value from the population.
   (b) **Define** the vectors $\overrightarrow{L}, \overrightarrow{R}$ with the properties

$$\begin{array}{rcl} L_i & = & -F|g_i^*| \\ R_i & = & F|g_i^*| \end{array}, \ i = 1, \ldots, n$$

   The parameter F is a positive number with the property $F \geq 1$.
   (c) The final outcome of the first phase is the vectors $\overrightarrow{L}$ and $\overrightarrow{R}$

2.2.2. The second phase of the method

In the second phase of the proposed methodology, an attempt is made to estimate the optimal value intervals (bounds) of the artificial neural network parameters. This phase initiates from the final outcome of the previous phase, the vectors $\overrightarrow{L}$ and $\overrightarrow{R}$. The estimation of the optimal value intervals is performed through Grammatical Evolution, which utilizes the grammar shown in Figure 1.

```
S::=<expr>   (0)
<expr> ::=  (<xlist> , <command>,<command>)  (0)
            |<expr>,<expr>                   (1)
<xlist>::=x1    (0)
            | x2 (1)
            .........
            | xn (n-1)
<command>  ::= NOTHING (0)
            |        SHRINK  (1)
            |        EXPAND  (2)
            |        SHRINK_EXPAND (3)
            |        EXPAND_SHRINK (4)
            |        SHRINK_SHRINK (5)
            |        EXPAND_EXPAND (6)
```

Figure 1. BNF grammar used in the implemented work. The numbers in parentheses denote the sequence number of the current production rule and they are used during the Grammatical Evolution procedure in order to produce valid programs. Symbols enclosed in < > are considered non - terminal symbols of the grammar. The parameter n denotes the number of parameters for the used neural network.

This grammar proposes a series of commands that will be applied to the bounds of the parameters of the neural network. These commands are a series of expressions in the form (identifier,left_command,right_command). The identifier represents the parameter where the command will be applied, the left_command indicates the command that will be executed on the left bound of this parameter and the right_command stands for the command that will be applied on the right bound of the corresponding parameter. These commands are:

1. NOTHING. Using this command does not change the limits of the corresponding parameter.
2. SHRINK. With this command, the corresponding bound is reduced by 50%.
3. EXPAND. Activating this command results in the corresponding bound being expanded by 50%.
4. SHRINK_EXPAND. This command firstly reduces the corresponding bound by 50% and afterwards this bound is expanded by 50%.
5. EXPAND_SHRINK. This command firstly expands by 50% the corresponding bound and subsequently this bound is reduced by 50%.
6. SHRINK_SHRINK. The execution of this command shrinks the corresponding bound two times.
7. EXPAND_EXPAND. This command expands the corresponding bound two times.

As a full working example consider a neural network with $H = 2$ processing nodes and the consider also the dimension of the input patterns to be $d = 2$. The total number of variables for the neural network is calculated as $n = (d+2)H = 8$. Also, we consider that the initial values of bounds are $\overrightarrow{L} = (-2,-4,-1,0,-4,-2,-1,-2)$ and $\overrightarrow{R} = (2,4,1,0,4,2,1,2)$. Also, consider the chromosome

$$g = [10, 15, 2, 9, 16, 4, 15, 19, 21]$$

The production steps for this example are shown in Table 1.

Table 1. Production steps for the example considered here.

CHROMOSOME	ACTION	EXPRESSION
10,15,2,9,16,4,15,19,21	10 mod 2=0	<expr>,<expr>
15,2,9,16,4,15,19,21	15 mod 2 =1	(<xlist>,<command>,<command>),<expr>
2,9,16,4,15,19,21	2%8=2	(x3,<command>,<command>),<expr>
9,16,4,15,19,21	9%7=2	(x3,EXPAND,<command>),<expr>
16,4,15,19,21	16%7=2	(x3,EXPAND,EXPAND),<expr>
4,15,19,21	4%2=0	(x3,EXPAND,EXPAND),(<xlist>,<command>,<command>)
15,19,21	15%8=7	(x3,EXPAND,EXPAND),(x8,<command>,<command>)
19,21	19%7=5	(x3,EXPAND,EXPAND),(x8,SHRINK_EXPAND,<command>)
21	21%7=0	(x3,EXPAND,EXPAND),(x8,SHRINK_EXPAND,NOTHING)

The final outcome of these steps is the expression 184

$$p = (x_3, \text{EXPAND}, \text{EXPAND}), (x8, \text{SHRINK\_EXPAND}, \text{NOTHING})$$

After the application of the program p to the vectors $\overrightarrow{L}$ and $\overrightarrow{R}$ the following vectors will be created: $\overrightarrow{L} = (-2, -4, -2, 0, -4, -2, -1, -1)$, $\overrightarrow{R} = (2, 4, 2, 0, 4, 2, 1, 2)$. The algorithm used in this phase is a variation of the genetic algorithm, that incorporates the Grammatical Evolution procedure to create programs that will be applied on the bounds of the neural network parameters. The process used in this phase is a genetic algorithm, where the chromosomes consist of value intervals for the parameters of the artificial neural network. The fitness of each chromosome is considered as an interval, and it is calculated through the process outlined in Algorithm 1. 185–192

Algorithm 1 Calculation of fitness value for every produced program p.

```
1. **Set** as $N_s$ the number of random samples that will be used.
2. **Set** $f_l = \infty$
3. **Set** $f_u = -\infty$
4. **Apply** the program $p$ to the bounds of the neural network to produce the new bounds $\overrightarrow{L_p}, \overrightarrow{R_p}$.
5. **For** $i = 1, \ldots, N_s$ **do**
    (a) **Create** randomly the set $w_p \in \left[\overrightarrow{L_p}, \overrightarrow{R_p}\right]$ as a new set of parameters for the neural network.
    (b) **Calculate** the training error $E_w = \sum_{k=1}^{M} \left(N\left(\overrightarrow{x}_k, \overrightarrow{w_p}\right) - y_k\right)^2$
    (c) **If** $E_w \le f_l$ then $f_l = E_w$
    (d) **If** $E_w \ge f_u$ then $f_u = E_w$
6. **End For**
7. **Return** as fitness value the interval $[f_l, f_u]$.
```

The steps of the proposed algorithm for the second phase are the following: 193

1. **Initialization Step**. 194
   (a) **Define** as N_c the population size and denote as N_g the maximum number of generations allowed. 195–196
   (b) **Define** as p_s the selection rate with $p_s \le 1$. 197
   (c) **Define** as p_m the mutation rate with $p_m \le 1$. 198
   (d) **Define** as p_l the used local search rate with $p_l \le 1$ 199
   (e) **Set** as N_s the number of samples that will be used during the fitness calculation. 200
   (f) **Set** as N_L the number of chromosomes that will participate in the local search procedure. 201–202

   (g) **Initialize** randomly the chromosomes $g_i,\ i = 1, \ldots, N_c$. Each chromosome is a set of randomly selected positive integers.
   (h) **Set** $k = 0$, the generation number.

2. **Fitness calculation step**.
   (a) **For** every chromosome $g_i,\ i = 1, \ldots, N_c$ **do**
      i. **Create** the corresponding program p_i using the Grammatical Evolution procedure and the associated BNF grammar outlined in Figure 1.
      ii. **Calculate** the fitness value f_i for program p_i using the algorithm shown in Algorithm 1.
   (b) **End For**
3. **Genetic operations step**.
   (a) Application of selection. During the selection procedure the chromosome are firstly sorted according to their fitness values. The comparison of any pair of fitness values $f_a = [a_1, a_2]$ and $f_b = [b_1, b_2]$ is performed using the following operator:

$$L^*(f_a, f_b) = \begin{cases} \text{TRUE}, & a_1 < b_1, \text{OR } (a_1 = b_1 \text{ AND } a_2 < b_2) \\ \text{FALSE}, & \text{OTHERWISE} \end{cases} \quad (5)$$

   The $(1 - p_s) \times N_c$ chromosomes with the lowest fitness values according to the previous operator are copied intact to the next generation. The remaining of them are replaced by chromosomes produced through crossover and mutation.
   (b) Application of crossover. During this procedure, for every pair of produced chromosomes $(\tilde{z}, \tilde{w})$ two chromosomes (z, w) are selected from the current population with the assistance of tournament selection. The new chromosomes are constructed using one - point crossover, that is graphically outlined in Figure 2.
   (c) Application of mutation. Every element of each chromosome is altered with probability p_m.
4. **Local search step**.
   (a) **For** each chromosome $g_i,\ i = 1, \ldots, N_c$ **do**
      i. **Draw** a random number $r \in [0, 1]$
      ii. **If** $r \leq p_l$ then alter randomly chromosome g_i with the procedure described in Algorithm 2.
   (b) **End for**
5. **Termination check step**.
   (a) **Set** $k = k + 1$
   (b) **If** $k \geq N_g$ terminate else goto step 2.
   (c) **Report** g^* as the final outcome of this algorithm having the lowest fitness value among the chromosomes of the population. The application of this chromosome to the bounds of the neural network will produce the set of bounds $\overrightarrow{L^*}$ and $\overrightarrow{R^*}$.

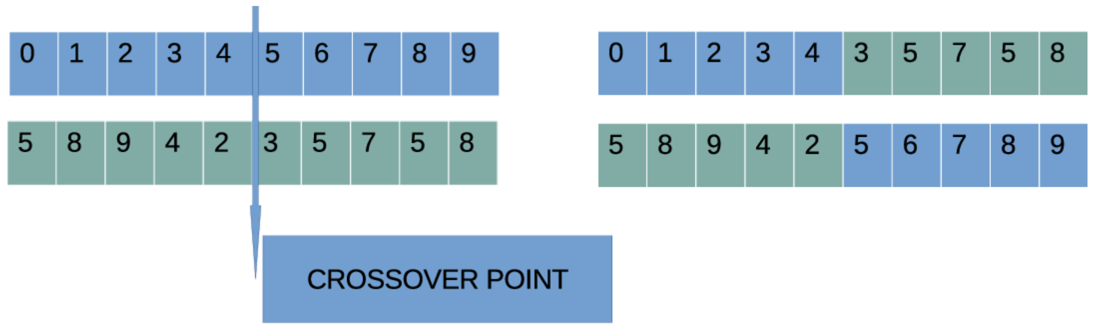

Figure 2. An example of one - point crossover mechanism, used in the Grammatical Evolution procedure.

Algorithm 2 Local search procedure applied on a chromosome g.

```
1. Set as $f_g$ the corresponding fitness value of chromosome $g$.
2. Create randomly the set $C = \{x_1, x_2, \ldots, x_{N_l}\}$ of $N_l$ chromosomes from the current population.
3. For $i = 1, \ldots, N_{cr}$ do
   (a) Execute the one - point crossover using chromosomes $g$ and $x_i$. The one - point crossover will create two chromosomes: $g_1$ and $g_2$ and the related fitness values are $f_{g1}$ and $f_{g2}$.
   (b) If $f_{g_1} \le f_g$ then $g = g_1$
   (c) else if $f_{g_2} \le f_g$ then $g = g_2$
   (d) End if
4. End For
```

2.2.3. The final phase of the method

During the last phase of the proposed method, the bounds $\overrightarrow{L^*}$ and $\overrightarrow{R^*}$ produced by the previous phase are evaluated using a modified genetic algorithm, used to train the artificial neural network.

1. **Initialization step**.
   (a) **Set** as N_c the number of chromosomes and as N_g the maximum number of generations.
   (b) **Set** as p_s the selection rate and as p_m the mutation rate.
   (c) **Initialize** randomly each chromosome $g_i,\ i = 1, \ldots, N_c$ inside the bounds $\overrightarrow{L^*}$ and $\overrightarrow{R^*}$ of the previous procedure.
   (d) **Set** $k = 0$ the generation number.
2. **Fitness Calculation step**. For every chromosome $g_i,\ i = 1, \ldots, N_c$ create the corresponding neural network $N(g_i, x)$ and set the corresponding fitness value with the following formula:

$$f_i = \sum_{j=1}^{M} \left(N\left(g_i, x_j\right) - y_j\right)^2$$

3. **Application of genetic operations**.
   (a) Application of selection. The chromosomes are sorted according to their fitness. The best $(1 - p_s) \times N_c$ of them will be copied to the next generation, while the remaining ones will be substituted by chromosomes produced during crossover and mutation.

   (b) Application of crossover. For each pair of produced chromosomes $(\tilde{z}, \tilde{w})$ two chromosomes z and w from the current population will be selected through tournament selection. The new chromosomes will be formed with a process similar with this of the crossover procedure of the first stage of the current work.
   (c) Application of mutation. Every element of each chromosome is altered with probability p_m.

4. **Termination check step**.
   (a) **Set** $k = k + 1$
   (b) **If** $k \le N_G$ then goto step 2.
5. **Local search step**.
   (a) **Obtain** the best chromosome g^* from the population and form the corresponding neural network $N(g^*, x)$
   (b) **Minimize** the associated training error

$$f^* = \sum_{j=1}^{M} \left(N\left(g_i^*, x_j\right) - y_j\right)^2$$

   using a local search procedure. In the proposed method the BFGS variant of Powell [67] was used.

3. Results

The new training procedure for the construction of weights of neural networks was tested on a wide series of classification and regression datasets, found in the recent literature and in the following websites:

1. The UCI website, https://archive.ics.uci.edu/(accessed on 24 November 2024)[68]
2. The Keel website, https://sci2s.ugr.es/keel/datasets.php(accessed on 24 November 2024)[69].
3. The Statlib URL ftp://lib.stat.cmu.edu/datasets/index.html(accessed on 24 November 2024).

3.1. Datasets

The following classification datasets were incorporated in the conducted experiments:

1. **Alcohol**. This dataset is related to experiments on alcohol consumption [70].
2. **Australian**, related to economic risk in bank transactions [71].
3. **Bands**, related to printing problems [72], with two distinct classes.
4. **Cleveland**, a medical dataset studied in many research papers [73,74].
5. **Circular** dataset, which is an artificial dataset.
6. **Dermatology**, a medical dataset related to dermatology problems [75] with 6 classes.
7. **Ecoli**, that is related to issues about proteins [76].
8. **Haberman**, a medical dataset used in the detection of breast cancer.
9. **Hayes-roth** dataset [77], a dataset with 3 classes.
10. **Heart**, a medical dataset about heart diseases [78] with two classes.
11. **HeartAttack**, a medical dataset related to the presence of heart diseases, with two classes.
12. **Hepatitis**, a dataset used for to detect the presence of hepatitis.
13. **Housevotes**, that uses data from the Congressional voting in USA [79].
14. **Ionosphere**, that used data from various measurements in the ionosphere [80,81].
15. **Liverdisorder**, which is a medical dataset [82,83] with two classes.
16. **Lymography** [84], which is a dataset with four distinct classes.
17. **Magic**, this dataset contains data from simulation regarding gamma particles [85].
18. **Mammographic**, a medical dataset used for the detection of breast cancer [86].

19. **Page Blocks** dataset [90], which is incorporated for the detection of page layout in documents.
20. **Parkinsons**, a dataset used to detect the presence of Parkinson's disease [87,88].
21. **Pima**, a medical dataset that used to detect the presence of diabetes[89].
22. **Phoneme**, where the purpose of this dataset is to distinguish between nasal and oral sounds.
23. **Popfailures**, a dataset related to climate measurements [91].
24. **Regions2**, used detect issues in the liver from various liver biopsy images [92].
25. **Ring**, a dataset related to some multivariate normal distributions.
26. **Saheart**, used to detect the presence of some heart diseases.[93].
27. **Segment** dataset [94], which is used for image processing .
28. **Statheart**, a medical dataset about heart diseases.
29. **Spambase**, a dataset used for the detection of spam emails.
30. **Spiral**, which is an artificial dataset.
31. **Student**, a dataset related to some experiments in schools [95].
32. **Tae**, this dataset is related to evaluations of teaching performance.
33. **Transfusion**, which is a medical dataset [96].
34. **Wdbc**, a medical dataset related to the detection of breast cancer [97,98].
35. **Wine**, a dataset related to the quality of wines [99,100].
36. **EEG** dataset, which is a medical dataset related to EEG measurements[101,102]. From the original dataset the following cases were obtained: Z_F_S, ZO_NF_S, Z_O_N_F_S and ZONF_S.
37. **Zoo**, that used to predict the class of some animals [103] .

Also the following regression datasets were used in the conducted experiments:

1. **Abalone**, a dataset regarding the prediction of the age of abalones [104].
2. **Airfoil**, a dataset derived from NASA [105] with 5 features.
3. **Auto**, a dataset used for the prediction of fuel consumption in cars.
4. **BK**, related to basketball games. The dataset has 4 features.
5. **BL**, a dataset used in some electricity experiments and it contains 7 features.
6. **Baseball**, a dataset with 16 features used for the estimation of the income of baseball players.
7. **Concrete**, a dataset with 8 features and it was used in civil engineering [106].
8. **DEE**, a dataset with 6 features, used in the prediction of electricity prices.
9. **EU**, a dataset originated in the STATLIB website.
10. **FA**, that contains related to body fat.
11. **Friedman**, a synthetic dataset used in various benchmarks [107].
12. **FY,** this dataset used to measure the longevity of fruit flies.
13. **HO**, a dataset founded in the STATLIB repository with 13 features.
14. **Housing**, used to predict the price of houses [108].
15. **Laser**, which is a dataset with 4 features and it has been used in various laser experiments.
16. **LW**, a dataset with 9 features used to measure the weight of babes.
17. **Mortgage**, an economic dataset with 15 features.
18. **Plastic**, a dataset related to the detection of pressure on plastics.
19. **PL**, a dataset with 2 features founded in the STATLIB repository.
20. **RealEstate,** a dataset found in the STATLIB website with 5 features.
21. **Quake**, a dataset used in earthquake measurements with 3 features.
22. **SN**, a dataset that provides experimental measurements related to trellising and pruning, with 11 features.
23. **Stock**, a dataset with 9 features for the prediction of the prices of various stocks.
24. **Treasury**, an economic dataset with 15 features.
25. **TZ**, derived from the STATLIB website and it has 60 features.
26. **VE** dataset, derived from the STALIB repository.

3.2. Experimental results

The algorithm used in the current work was coded in ANSI C++. All the runs were performed 30 times using different seeds for the random generator each time. The validation of the experiments was performed using the 10 - fold validation technique. The experiments were carried out on an AMD Ryzen 5950X with 128GB of RAM. The used operating system was Debian Linux. In Table 2 the values used in all experimental parameters are listed.

Table 2. The values used for all experimental parameters of the proposed method.

PARAMETER	MEANING	VALUE
N_g	Number of generations allowed	200
N_c	Number of chromosomes	500
N_s	Number of samples	50
N_l	Number of chromosome used in local search	20
p_s	Selection rate	0.1
p_m	Mutation rate	0.05
p_l	Local search rate	0.001
H	Number of processing nodes	10

The table 3 outlines the experimental results for the classification datasets and in Table 4 the experimental results for the regression datasets are listed. The following applies to these tables:

1. The column DATASET represents the used dataset.
2. The column ADAM denotes the application of the ADAM optimizer to a neural network with $H = 10$ processing nodes.
3. The column BFGS stands for the application of the BFGS variant of Powell to a neural network with $H = 10$ processing nodes.
4. The column GENETIC stands for the usage of a genetic algorithm on a neural network with $H = 10$ processing nodes. The values for the parameters of this genetic algorithm are shown in Table 2.
5. The column GNN stands for the incorporation of the proposed method to a neural network with $H = 10$ processing nodes with parameter settings outlined in Table 2.
6. The row denoted as AVERAGE represents the average classification or regression error for all datasets.

Table 3. Experimental results for the classification datasets. Numbers in cells denote the average classification error as calculated on the corresponding test set.

DATASET	ADAM	BFGS	GENETIC	GNN
ALCOHOL	57.78%	41.50%	39.57%	16.63%
AUSTRALIAN	35.65%	38.13%	32.21%	14.69%
BALANCE	12.27%	8.64%	8.97%	7.98%
BANDS	36.92%	36.67%	34.92%	34.38%
CLEVELAND	67.55%	77.55%	51.60%	44.89%
CIRCULAR	19.95%	6.08%	5.99%	4.68%
DERMATOLOGY	26.14%	52.92%	30.58%	11.79%
ECOLI	64.43%	69.52%	54.67%	51.42%
HABERMAN	29.00%	29.34%	28.66%	27.01%
HAYES-ROTH	59.70%	37.33%	56.18%	34.99%
HEART	38.53%	39.44%	28.34%	14.83%
HEARTATTACK	45.55%	46.67%	29.03%	19.23%
HEPATITIS	68.13%	72.47%	62.12%	56.67%
HOUSEVOTES	7.48%	7.13%	6.62%	6.23%
IONOSPHERE	16.64%	15.29%	15.14%	15.71%
LIVERDISORDER	41.53%	42.59%	31.11%	31.89%
LYMOGRAPHY	39.79%	35.43%	28.42%	19.26%
MAGIC	40.55%	17.30%	21.75%	15.67%
MAMMOGRAPHIC	46.25%	17.24%	19.88%	16.33%
PARKINSONS	24.06%	27.58%	18.05%	11.53%
PAGE BLOCKS	34.27%	8.49%	6.84%	7.66%
PHONEME	29.43%	15.58%	15.55%	16.59%
PIMA	34.85%	35.59%	32.19%	25.81%
POPFAILURES	5.18%	5.24%	5.94%	5.25%
REGIONS2	29.85%	36.28%	29.39%	29.24%
RING	28.80%	29.44%	28.80%	21.68%
SAHEART	34.04%	37.48%	34.86%	29.92%
SEGMENT	49.75%	68.97%	57.72%	31.21%
SONAR	30.33%	25.85%	22.40%	22.80%
SPAMBASE	48.05%	18.16%	6.37%	6.17%
SPIRAL	47.67%	47.99%	48.66%	41.97%
STATHEART	44.04%	39.65%	27.25%	18.20%
STUDENT	5.13%	7.14%	5.61%	4.98%
TAE	60.20%	51.58%	49.84%	48.89%
TRANSFUSION	25.68%	25.84%	24.87%	23.34%
WDBC	35.35%	29.91%	8.56%	3.70%
WINE	29.40%	59.71%	19.20%	9.14%
Z_F_S	47.81%	39.37%	10.73%	8.35%
Z_O_N_F_S	78.79%	65.67%	64.81%	60.83%
ZO_NF_S	47.43%	43.04%	21.54%	6.29%
ZONF_S	11.99%	15.62%	4.36%	2.77%
ZOO	14.13%	10.70%	9.50%	6.03%
AVERAGE	**36.91%**	**34.19%**	**27.11%**	**21.11%**

Table 4. Experimental results for the series of regression datasets. The numbers in cells correspond to the average regression error of each test set.

DATASET	ADAM	BFGS	GENETIC	GNN
ABALONE	4.30	5.69	7.17	4.32
AIRFOIL	0.005	0.003	0.003	0.001
AUTO	70.84	60.97	12.18	8.69
BK	0.0252	0.28	0.027	0.039
BL	0.622	2.55	5.74	0.013
BASEBALL	77.90	119.63	103.60	47.25
CONCRETE	0.078	0.066	0.0099	0.004
DEE	0.63	2.36	1.013	0.212
EU	604.253	608.99	252.97	148.76
FA	0.048	0.426	0.025	0.097
FRIEDMAN	22.896	1.263	1.249	1.573
FY	0.038	0.22	0.65	0.121
HO	0.035	0.62	2.78	0.05
HOUSING	80.998	97.38	43.26	14.79
LASER	0.03	0.015	0.59	0.0029
LW	0.028	2.98	1.90	0.313
MORTGAGE	9.24	8.23	2.41	0.0222
PLASTIC	11.713	20.323	2.791	2.186
PY	0.321	0.578	0.56	0.111
PL	0.117	0.29	0.28	0.0205
REALESTATE	186.778	170.28	81.19	70.928
QUAKE	0.07	0.42	0.04	0.04
SN	0.026	0.40	2.95	0.025
STOCK	180.893	302.43	3.881	2.986
TREASURY	11.16	9.91	2.929	0.0633
TZ	0.07	3.27	5.38	0.652
VE	0.359	1.92	2.43	0.0493
AVERAGE	**46.795**	**52.648**	**19.926**	**11.234**

The table 3 illustrates the performance of four neural network training methods—ADAM, BFGS, GENETIC, and GNN—across a variety of datasets. Lower error rates indicate better performance, and based on the average error rates, GNN emerges as the most effective method, with an average error rate of 21.1%. It is followed by GENETIC, with an average error of 27.1%, while BFGS and ADAM exhibit higher error rates, with averages of 34.1% and 36.9%, respectively. Delving deeper, GNN consistently outperforms the other methods across many datasets, achieving notably low error rates. For example, it records an error of just 2.77% on the ZONF_S dataset and 6.17% on SPAMBASE, demonstrating its robustness even in challenging cases. GENETIC also delivers competitive performance, with low error rates in datasets such as SPIRAL (48.66%) and Z_O_N_F_S (64.81%). However, in some datasets, like SPAMBASE (6.37%), its performance does not match that of GNN. BFGS, while less effective on average compared to GNN and GENETIC, excels in specific datasets. For instance, it achieves remarkably low error rates on CLEVELAND (77.55%) and SEGMENT (68.97%), outperforming other methods in these cases. However, its overall performance is less consistent across the full spectrum of datasets. ADAM, despite its stability and lack of significant deviations, has the highest average error rate (36.9%) among the methods. It does not significantly outperform in any particular dataset. For instance, in the HEPATITIS dataset, it records an error of 68.13%, which is notably higher than GNN's (56.67%) and GENETIC's (62.12%). In summary, GNN stands out as the most effective method, achieving the lowest average error rate and excelling across diverse datasets. GENETIC ranks as the second-best option, demonstrating stable performance with relatively low error rates. BFGS, while generally less competitive, performs exceptionally well in certain datasets, making it suitable for specific applications. Finally, ADAM, although consistent, is the least competitive overall, with higher average error rates compared to the other methods.

The table 4 presents the regression error achieved by four training methods—ADAM, BFGS, GENETIC, and GNN—on a variety of datasets. Lower error values indicate better performance. A comparison of the methods reveals that GNN consistently delivers the lowest regression error on average, making it the most effective overall. GENETIC follows with relatively competitive results, while BFGS and ADAM exhibit higher error rates across most datasets. A closer look at the results shows that GNN achieves remarkable performance on several datasets. For instance, in the AIRFOIL dataset, it records the lowest error of 0.001, outperforming all other methods. Similarly, it achieves an error of just 0.004 in CONCRETE, 0.013 in BL, and 0.0222 in MORTGAGE, demonstrating its strong capability across various scenarios. The GENETIC method also performs well in certain datasets, such as AUTO (12.18) and PLASTIC (2.791), though it falls short of GNN in most cases. BFGS exhibits decent performance in specific datasets but is generally less effective than GNN and GENETIC. For example, in FRIEDMAN, it achieves a competitive error of 1.263, but it struggles in datasets such as STOCK (302.43) and BASEBALL (119.63). ADAM, while consistent in its results, records the highest errors in many cases, such as REALESTATE (186.778) and EU (604.253), indicating weaker performance relative to the other methods. Overall, the results highlight the superiority of GNN, which not only achieves the lowest regression error on average but also demonstrates robustness and reliability across a wide range of datasets. GENETIC ranks as the second-best approach, delivering solid performance in key datasets. BFGS, while showing potential in certain tasks, is less competitive on average, and ADAM, despite its consistency, struggles to match the effectiveness of the other methods in reducing regression error.

The statistical comparison between the used methods for the classification datasets is depicted in Figure 3 while for the regression datasets is graphically outlined in Figure 4.

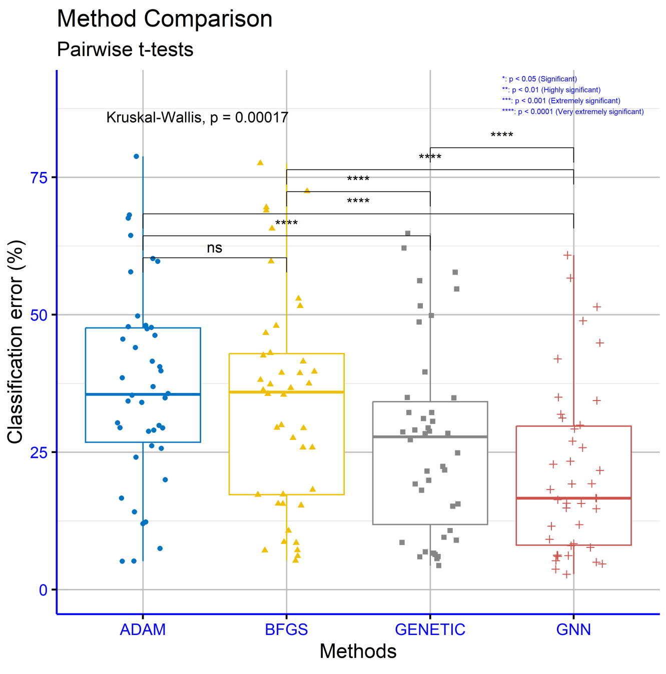

Figure 3. Statistical comparison between the used methods for the classification datasets.

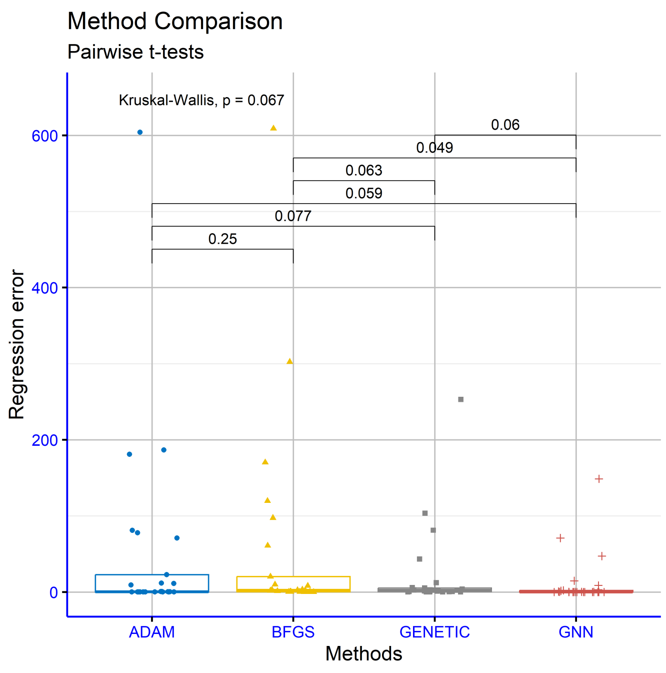

Figure 4. Statistical comparison between the used methods for the regression datasets.

Figure 3 highlights significant differences between the methods ADAM, BFGS, GENETIC, and GNN in terms of error. The Kruskal-Wallis test, which is used to assess whether there are overall differences among more than two groups, yielded a p-value of 0.00017. This extremely small value, which is below the standard significance level , indicates that at least one method differs significantly from the others. Based on this result, it was necessary to perform pairwise comparisons to determine which methods exhibit statistically significant differences from each other. The comparison between ADAM and BFGS resulted in a p-value of 0.17, which is above the significance threshold. This suggests that there is no statistically significant difference between these two methods, and their performance in terms of error is comparable. In contrast, the comparison between ADAM and GENETIC yielded a p-value of 9.9e-07, a very small value that indicates a statistically significant difference, with GENETIC outperforming ADAM. Similarly, the comparison of ADAM with GNN produced a p-value of 2.1e-10, demonstrating a significant difference and showing that GNN performs better than ADAM. The comparison between BFGS and GENETIC yielded a p-value of 8.6e-05, highlighting that GENETIC is significantly superior to BFGS. Furthermore, the comparison of BFGS with GNN resulted in a p-value of 1.7e-07, which shows that GNN is clearly better than BFGS. Finally, the comparison between GENETIC and GNN yielded a p-value of 1.9e-06, indicating a statistically significant difference, with GNN outperforming GENETIC. In summary, the results reveal a clear hierarchy in the performance of the methods. GNN emerges as the method with the lowest mean error, showing statistically significant superiority over all other methods. GENETIC ranks second, as it significantly outperforms ADAM and BFGS but falls short of GNN. On the other hand, ADAM and BFGS do not exhibit a statistically significant difference from each other and appear to be on the same level, with higher errors compared to the other two methods. These results underscore the clear advantage of GNN over the other methods for the given problem.

Figure 4 explores potential differences between the methods ADAM, BFGS, GENETIC, and GNN regarding regression error. The Kruskal-Wallis test, a non-parametric method used to assess whether there are overall differences among more than two groups, produced a p-value of 0.067. While this value is slightly above the standard significance threshold , it is very close, suggesting that there may be underlying differences between the methods worth further investigation. Pairwise comparisons revealed additional insights. The comparison between ADAM and BFGS yielded a p-value of 0.25, which is well above the significance threshold, indicating no significant difference between these methods. However, the comparison between ADAM and GENETIC resulted in a p-value of 0.077, which, while not strictly significant, is notably close to the 0.05 threshold, hinting at a potential difference. Similarly, the comparison of ADAM with GNN produced a p-value of 0.059, again close to the significance level, suggesting a possible distinction between these methods. The comparison between BFGS and GENETIC yielded a p-value of 0.063, further emphasizing a borderline result that approaches significance. For BFGS versus GNN, the p-value was 0.049, just below the 0.05 threshold, indicating a statistically significant difference, with GNN showing better performance. Finally, the comparison between GENETIC and GNN resulted in a p-value of 0.06, which, although slightly above the significance level, remains close enough to warrant attention. In summary, while the overall Kruskal-Wallis test did not reach the conventional significance level, the p-values from the pairwise comparisons suggest that certain methods, particularly GNN, may exhibit better performance compared to others. These borderline p-values, close to the 0.05 threshold, highlight the potential for meaningful differences, particularly between GNN and other methods, and suggest the need for further investigation or larger sample sizes to confirm these trends.

4. Conclusions

The proposed method utilizes Grammatical Evolution for training artificial neural networks, leveraging BNF grammars to construct and optimize the network's parameters. The methodology is structured into three phases. In the first phase, an initial estimation of the network parameters is performed through an optimization process, such as genetic algorithms. These provide the necessary flexibility and allow adaptation to the specific characteristics of each dataset. In the second phase, the BNF rules of Grammatical Evolution are employed to further refine the parameter bounds. Finally, in the third phase, the network is trained within these optimized bounds. The results indicate that the proposed method outperforms other approaches, such as ADAM or standard genetic algorithms, in both classification and regression problems. The errors recorded using the GE-based method are significantly lower than those of other methods. Notably, the reduction in errors demonstrates the method's enhanced parameter adaptation process and its ability to avoid overfitting.

Despite the impressive results, there is room for further improvements and extensions to the method. A significant proposal is the integration of parallel processing during the genetic search phase, which could dramatically reduce the time required for the training process. Additionally, allowing the method to extend and shrink parameter bounds with dynamic rates instead of a fixed 50% value could enable a more adaptive process tailored to the specifics of each dataset. Another suggestion involves improving the local search optimization during the second phase. Employing more efficient local search algorithms could further reduce training errors and enhance the method's performance. Simultaneously, broadening the application of the method to more datasets is crucial, both to evaluate its generalizability and robustness and to derive insights into its adaptability across various data types. Moreover, incorporating automated mechanisms for selecting the initial parameter intervals based on the properties of each dataset would be beneficial. Such a feature could further enhance the method's performance and flexibility. Overall, the proposed methodology is a promising approach for training neural networks. However, integrating the aforementioned improvements could make it even more effective and widely applicable.

Author Contributions: I.G.T. and V.C. conceived the idea and methodology. I.G.T. performed the experiments, employing several datasets, and provided the comparative experiments. V.C. performed the statistical analysis. All authors prepared 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 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. M. Mjahed, The use of clustering techniques for the classification of high energy physics data, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **559**, pp. 199-202, 2006.
2. M Andrews, M Paulini, S Gleyzer, B Poczos, End-to-End Event Classification of High-Energy Physics Data, Journal of Physics: Conference Series **1085**, 2018.
3. P. He, C.J. Xu, Y.Z. Liang, K.T. Fang, Improving the classification accuracy in chemistry via boosting technique, Chemometrics and Intelligent Laboratory Systems 70, pp. 39-46, 2004.
4. J.A. Aguiar, M.L. Gong, T.Tasdizen, Crystallographic prediction from diffraction and chemistry data for higher throughput classification using machine learning, Computational Materials Science **173**, 109409, 2020.
5. I. Kaastra, M. Boyd, Designing a neural network for forecasting financial and economic time series, Neurocomputing **10**, pp. 215-236, 1996.
6. R. Hafezi, J. Shahrabi, E. Hadavandi, A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price, Applied Soft Computing **29**, pp. 196-210, 2015.
7. S.S. Yadav, S.M. Jadhav, Deep convolutional neural network based medical image classification for disease diagnosis. J Big Data **6**, 113, 2019.
8. L. Qing, W. Linhong , D. Xuehai, A Novel Neural Network-Based Method for Medical Text Classification, Future Internet **11**, 255, 2019.
9. C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
10. G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.
11. M. Egmont-Petersen, D. de Ridder, H. Handels, Image processing with neural networks—a review, Pattern recognition **35**, pp. 2279-2301, 2002.
12. M. Khashei, M. Bijari, An artificial neural network (p,d,q) model for timeseries forecasting, Expert Systems with Applications **37**, pp. 479-489, 2010.
13. A. Mellit, A. Massi Pavan, A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy, Solar Energy **84**, pp. 807-821, 2010.
14. F. Amato, A. López, E. María Peña-Méndez, P. Vaňhara,A. Hampl, J. Havel, Artificial neural networks in medical diagnosis, Journal of Applied Biomedicine **11**, pp. 47-58, 2013.
15. Y. Shirvany, M. Hayati, R. Moradian, Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations, Applied Soft Computing **9**, pp. 20-29, 2009.
16. A. Malek, R. Shekari Beidokhti, Numerical solution for high order differential equations using a hybrid neural network—Optimization method, Applied Mathematics and Computation **183**, pp. 260-271, 2006.
17. A. Topuz, Predicting moisture content of agricultural products using artificial neural networks, Advances in Engineering Software **41**, pp. 464-470, 2010.
18. A. Escamilla-García, G.M. Soto-Zarazúa, M. Toledano-Ayala, E. Rivas-Araiza, A. Gastélum-Barrios, Abraham,Applications of Artificial Neural Networks in Greenhouse Technology and Overview for Smart Agriculture Development, Applied Sciences **10**, Article number 3835, 2020.
19. D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by back-propagating errors, Nature **323**, pp. 533 - 536 , 1986.
20. M. Riedmiller and H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm, Proc. of the IEEE Intl. Conf. on Neural Networks, San Francisco, CA, pp. 586–591, 1993.
21. 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.
22. B. Robitaille and B. Marcos and M. Veillette and G. Payre, Modified quasi-Newton methods for training neural networks, Computers & Chemical Engineering **20**, pp. 1133-1140, 1996.
23. 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.

24. A. Yamazaki, M. C. P. de Souto,T. B. Ludermir, 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 **1**, pp. 547-552 , 2002.
25. F. H. F. Leung, H. K. Lam, S. H. Ling and P. K. S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, IEEE Transactions on Neural Networks **14**, pp. 79-88, 2003
26. C. Zhang, H. Shao and Y. Li, Particle swarm optimization for evolving artificial neural network, IEEE International Conference on Systems, Man, and Cybernetics, , pp. 2487-2490, 2000.
27. J. lonen, J.K. Kamarainen, J. Lampinen, Differential Evolution Training Algorithm for Feed-Forward Neural Networks, Neural Processing Letters **17**, pp. 93–105, 2003.
28. K.M. Salama, A.M. Abdelbar, Learning neural network structures with ant colony algorithms, Swarm Intell **9**, pp. 229–265, 2015.
29. A. Askarzadeh, A. Rezazadeh, Artificial neural network training using a new efficient optimization algorithm, Applied Soft Computing **13**, pp. 1206-1213, 2013.
30. P.G. Benardos, G.-C. Vosniakos, Optimizing feedforward artificial neural network architecture, Engineering Applications of Artificial Intelligence **20**, pp. 365-382, 2007.
31. D. Karaboga and B. Akay, "Artificial Bee Colony (ABC) Algorithm on Training Artificial Neural Networks," 2007 IEEE 15th Signal Processing and Communications Applications, Eskisehir, Turkey, 2007, pp. 1-4, doi: 10.1109/SIU.2007.4298679.
32. J.F. Chen, Q.H. Do,H.N. Hsieh, Training Artificial Neural Networks by a Hybrid PSO-CS Algorithm, Algorithms **8**, pp. 292-308, 2015.
33. X.S. Yang, S. Deb, Engineering Optimisation by Cuckoo Search, Int. J. Math. Model. Numer. Optim. **1**, 330–343, 2010.
34. S. Scanzio, S. Cumani, R. Gemello, F. Mana, P. Laface, Parallel implementation of Artificial Neural Network training for speech recognition, Pattern Recognition Letters **31**, pp. 1302-1309, 2010.
35. X. Sierra-Canto, F. Madera-Ramirez and V. Uc-Cetina, "Parallel Training of a Back-Propagation Neural Network Using CUDA," 2010 Ninth International Conference on Machine Learning and Applications, Washington, DC, USA, 2010, pp. 307-312, doi: 10.1109/ICMLA.2010.52.
36. F. Åström, R. Koker, A parallel neural network approach to prediction of Parkinson's Disease, Expert Systems with Applications **38**, , pp. 12470-12474, 2011.
37. I. Ivanova, M. Kubat, Initialization of neural networks by means of decision trees, Knowledge-Based Systems **8**, pp. 333-344, 1995.
38. J.Y.F. Yam, T.W.S. Chow, A weight initialization method for improving training speed in feedforward neural network, Neurocomputing **30**, pp. 219-232, 2000.
39. K. Chumachenko, A. Iosifidis, M. Gabbouj, Feedforward neural networks initialization based on discriminant learning, Neural Networks **146**, pp. 220-229, 2022.
40. F. Itano, M. A. de Abreu de Sousa, E. Del-Moral-Hernandez, Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm, In: 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 2018, pp. 1-8, 2018.
41. S.S. Sodhi, P. Chandra, Interval based Weight Initialization Method for Sigmoidal Feedforward Artificial Neural Networks, AASRI Procedia **6**, pp. 19-25, 2014.
42. C. A. R. de Sousa, "An overview on weight initialization methods for feedforward neural networks," 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 2016, pp. 52-59, doi: 10.1109/IJCNN.2016.7727180.
43. S.J. Nowlan and G.E. Hinton, Simplifying neural networks by soft weight sharing, Neural Computation 4, pp. 473-493, 1992.
44. 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.
45. M.C. Mozer and P. Smolensky, Skeletonization: a technique for trimming the fat from a network via relevance assesment. In D.S. Touretzky (Ed.), Advances in Neural Processing Systems, Volume 1, pp. 107-115, San Mateo CA: Morgan Kaufmann, 1989.
46. M. Augasta and T. Kathirvalavakumar, Pruning algorithms of neural networks — a comparative study, Central European Journal of Computer Science, 2003.
47. Nitish Srivastava, G E Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan R Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, Journal of Machine Learning Research **15**, pp. 1929-1958, 2014.
48. A. Gupta, S.M. Lam, Weight decay backpropagation for noisy data, Neural Networks **11**, pp. 1127-1138, 1998.
49. 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.
50. N.K. Treadgold, T.D. Gedeon, Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm, IEEE Trans. on Neural Networks 9, pp. 662-668, 1998.
51. M.D. Shahjahan, M. Kazuyuki, Neural network training algorithm with possitive correlation, IEEE Trans. Inf & Syst. **88**, pp. 2399-2409, 2005.
52. Mirjalili, S., & Mirjalili, S. (2019). Genetic algorithm. Evolutionary algorithms and neural networks: theory and applications, 43-55.
53. M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5**,pp. 349–358, 2001.
54. J. W. Backus. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. Proceedings of the International Conference on Information Processing, UNESCO, 1959, pp.125-132.

55. A.O. Puente, R. S. Alfonso, M. A. Moreno, Automatic composition of music by means of grammatical evolution, In: APL '02: Proceedings of the 2002 conference on APL: array processing languages: lore, problems, and applications July 2002 Pages 148–155.
56. E. Galván-López, J.M. Swafford, M. O'Neill, A. Brabazon, Evolving a Ms. PacMan Controller Using Grammatical Evolution. In: , et al. Applications of Evolutionary Computation. EvoApplications 2010. Lecture Notes in Computer Science, vol 6024. Springer, Berlin, Heidelberg, 2010.
57. N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, M. O'Neill, Evolving levels for Super Mario Bros using grammatical evolution, 2012 IEEE Conference on Computational Intelligence and Games (CIG), 2012, pp. 304-31.
58. A. Ortega, A. A. Dalhoum, M. Alfonseca, Grammatical evolution to design fractal curves with a given dimension, IBM Journal of Research and Development **47**, pp. 483-493, 2003.
59. I. Dempsey, M.O' Neill, A. Brabazon, Constant creation in grammatical evolution, International Journal of Innovative Computing and Applications **1** , pp 23–38, 2007.
60. R. Burbidge, J. H. Walker and M. S. Wilson, "Grammatical evolution of a robot controller," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 2009, pp. 357-362, doi: 10.1109/IROS.2009.5354411.
61. Peabody, C., & Seitzer, J. (2015, March). GEF: a self-programming robot using grammatical evolution. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 29, No. 1).
62. J. I. Hidalgo, J. M. Colmenar, J.L. Risco-Martin, A. Cuesta-Infante, E. Maqueda, M. Botella, J.A. Rubio, Modeling glycemia in humans by means of Grammatical Evolution, Applied Soft Computing **20**, pp. 40-53, 2014.
63. L. Araujo, J. Martinez-Romo, A. Duque, Discovering taxonomies in Wikipedia by means of grammatical evolution. Soft Comput **22**, pp. 2907–2919, 2018.
64. C. Martín, D. Quintana, P. Isasi, Grammatical Evolution-based ensembles for algorithmic trading, Applied Soft Computing **84**, 105713, 2019.
65. I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing **72**, pp. 269-277, 2008.
66. P. Kaelo, M.M. Ali, Integrated crossover rules in real coded genetic algorithms, European Journal of Operational Research **176**, pp. 60-76, 2007.
67. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
68. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository, https://archive.ics.uci.edu.
69. 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.
70. 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.
71. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
72. B. Evans, D. Fisher, Overcoming process delays with decision tree induction. IEEE Expert **9**, pp. 60-66, 1994.
73. 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.
74. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
75. 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.
76. P. Horton, K.Nakai, A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins, In: Proceedings of International Conference on Intelligent Systems for Molecular Biology **4**, pp. 109-15, 1996.
77. 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.
78. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
79. 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.
80. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
81. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
82. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
83. J. Mcdermott, R.S. Forsyth, Diagnosing a disorder in a classification benchmark, Pattern Recognition Letters **73**, pp. 41-43, 2016.
84. 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.
85. Heck, D., Knapp, J., Capdevielle, J. N., Schatz, G., & Thouw, T. (1998). CORSIKA: A Monte Carlo code to simulate extensive air showers.

86. 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.
87. 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.
88. 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.
89. 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.
90. F. Esposito F., D. Malerba, G. Semeraro, Multistrategy Learning for Document Recognition, Applied Artificial Intelligence **8**, pp. 33-84, 1994.
91. 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.
92. 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.
93. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
94. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
95. 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.
96. 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.
97. 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.
98. 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.
99. 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.
100. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
101. 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.
102. 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
103. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
104. 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
105. 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).
106. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
107. Friedman, J. (1991): Multivariate Adaptative Regression Splines. Annals of Statistics, 19:1, 1--141.
108. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.