big data and cognitive computing

MDPI

Article

Training Neural Networks with a Procedure Guided by BNF Grammars

Ioannis G. Tsoulos * and Vasileios Charilogis

Department of Informatics and Telecommunications, University of Ioannina, 45110 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 overfitting 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

check for updates

Academic Editor: Salvador García López

Received: 6 December 2024
Revised: 30 December 2024
Accepted: 31 December 2024
Published: 2 January 2025

Citation: Tsoulos, I.G.; Charilogis, V. Training Neural Networks with a Procedure Guided by BNF Grammars. *Big Data Cogn. Comput.* **2025**, *9*, 5. https://doi.org/10.3390/bdcc9010005

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

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

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

In Equation (1) the set $(\overrightarrow{x_i}, y_i)$, $i = 1, \ldots, M$ 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 et 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 overfitting 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 method proposed here is quite general and can be applied to other neural network architectures, such as recurrent neural networks [55] or convolutional neural networks [56]. The only requirement for the method to be applicable to such architectures is that the parameters of the corresponding machine learning models are available and that the initial value range for these parameters is known.

Furthermore, from a technical point of view, the method can be applied to models that have a huge number of parameters without any technical limitation, however, it will be necessary to use large-sized chromosomes in the Grammatical Evolution process, in order

to make the process of finding the most reliable value interval for the parameters of the machine learning model in question more efficient.

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 [57], video games [58,59], design of fractal curves [60], constant creation [61], robotics [62,63], modeling glycemia in humans [64], Wikipedia taxonomies [65], economics [66] 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 [67] and it is defined as follows:

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

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:

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

These neural networks have one input level, where the pattern is presented to the neural network and one hidden level with H processing nodes. A representation of this neural networks is depicted in Figure 1.

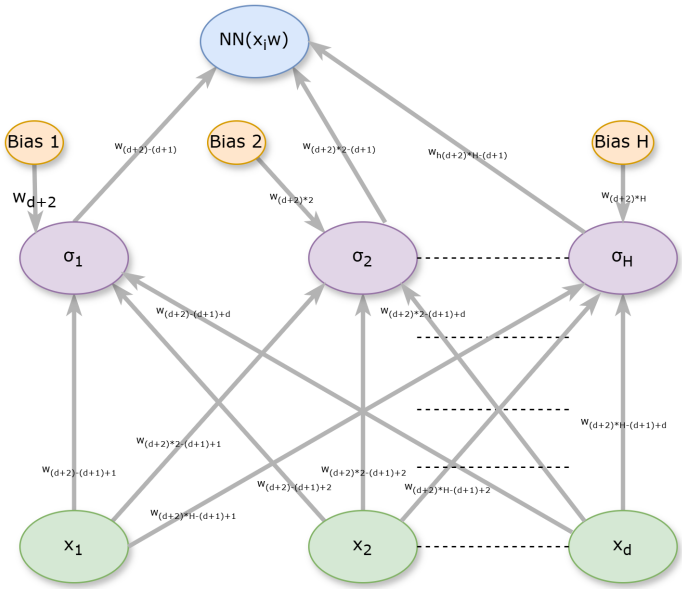


Figure 1. The structure of the neural network incorporated in the current work.

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:

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

   (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:

$$\begin{aligned} \tilde{z}_i &= a_i z_i + (1 - a_i) w_i \\ \tilde{w}_i &= a_i w_i + (1 - a_i) z_i \end{aligned} \tag{4}$$

The values a_i are random numbers with $a_i \in [-0.5, 1.5]$ [68].

(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

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

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 2.

```
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 2. 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	(x_3,<command>,<command>),<expr>
9,16,4,15,19,21	9%7 = 2	(x_3,EXPAND,<command>),<expr>
16,4,15,19,21	16%7 = 2	(x_3,EXPAND,EXPAND),<expr>
4,15,19,21	4%2 = 0	(x_3,EXPAND,EXPAND),(<xlist>,<command>,<command>)
15,19,21	15%8 = 7	(x_3,EXPAND,EXPAND),(x_8,<command>,<command>)
19,21	19%7 = 5	(x_3,EXPAND,EXPAND),(x_8,SHRINK_EXPAND,<command>)
21	21%7 = 0	(x_3,EXPAND,EXPAND),(x_8,SHRINK_EXPAND,NOTHING)

The final outcome of these steps is the expression

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

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 analytical form for this example is the neural network defined as:

$$N(\overrightarrow{x}, \overrightarrow{w}) = w_1\sigma(x_1w_2 + x_2w_3 + w_4) + w_5\sigma(x_1w_6 + x_2w_7 + w + w_8)$$

For this specific example, the following applies to each variable of the artificial neural network:

1. $x_1 \in [-2, 2]$
2. $x_2 \in [-4, 4]$
3. $x_3 \in [-2, 2]$
4. $x_4 \in [0, 0]$. This variable can be considered as fixed to 0.0
5. $x_5 \in [-4, 4]$
6. $x_6 \in [-2, 2]$
7. $x_7 \in [-1, 1]$
8. $x_8 \in [-1, 2]$

Furthermore, the method can significantly limit the value ranges of the artificial neural network parameters or even set some of them to zero. In this way, the method can indirectly construct the structure of the neural network. A figure depicting the previously mentioned example neural network is shown in Figure 3.

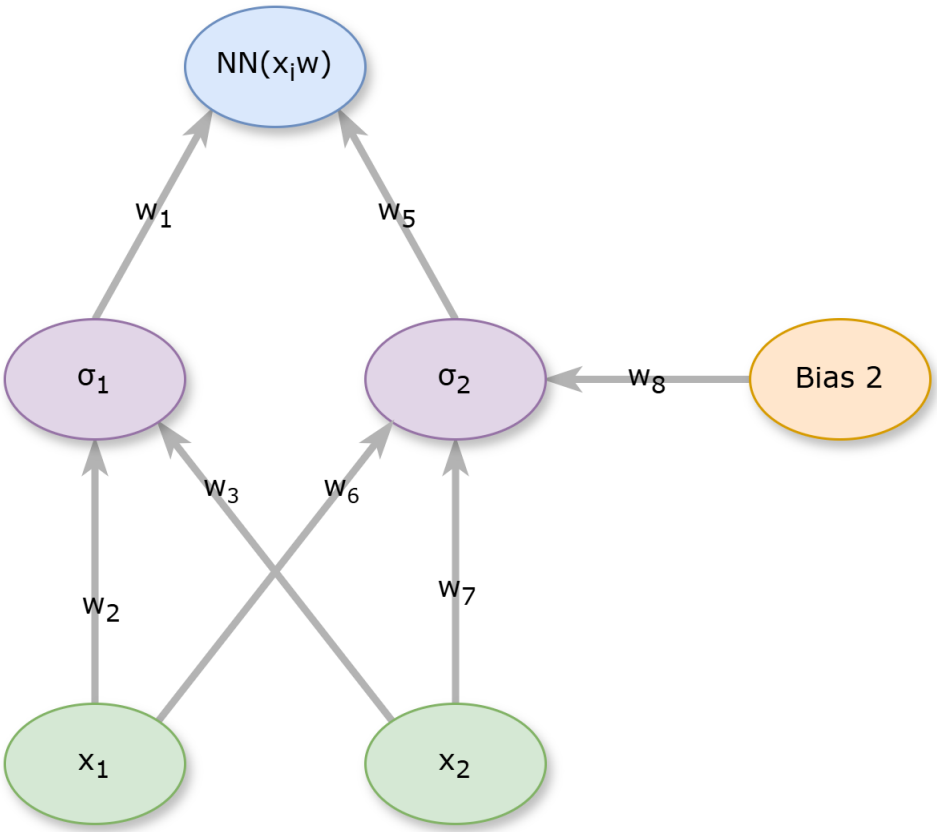


Figure 3. A representation of the example neural network presented here.

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.

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** $E_w \leq f_l$ then $f_l = E_w$
   (d) **If** $E_w \geq f_u$ 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:

1. **Initialization Step**.
   (a) **Define** as N_c the population size and denote as N_g the maximum number of generations allowed.
   (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) **Define** as p_l the used local search rate with $p_l \leq 1$
   (e) **Set** as N_s the number of samples that will be used during the fitness calculation.
   (f) **Set** as N_L the number of chromosomes that will participate in the local search procedure.
   (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 2.
      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 4.

(c) Application of mutation. Every element of each chromosome is altered with probability p_m. During this procedure a random integer value is selected and it replaces the chosen element of the under processing chromosome.

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 $\overrightarrow{R^*}$.

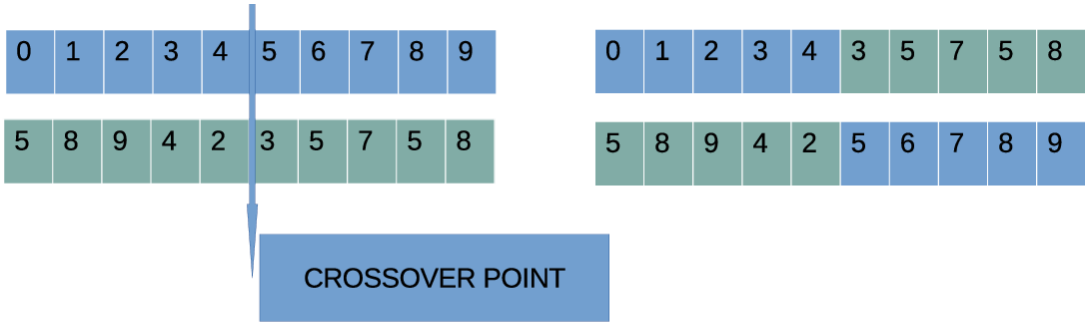


Figure 4. An example of one-point crossover mechanism, used in the Grammatical Evolution procedure.

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_{g_1} and f_{g_2}.
   (b) **If** $f_{g_1} \leq f_g$ **then** $g = g_1$
   (c) **else if** $f_{g_2} \leq f_g$ **then** $g = g_2$
   (d) **End if**
4. **End For**

The purpose of this phase of the algorithm is to identify the most promising range of values for the parameters of the artificial neural network through the application of partitioning rules. Furthermore, the above procedure will significantly reduce the local minima contained in the error function, since the initial search space will be significantly limited to only those regions that may contain the global minimum or approximations thereof.

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:
$$f_i = \sum_{j=1}^{M} \left(N(g_i, x_j) - y_j \right)^2$$
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 $(\widetilde{z}, \widetilde{w})$ 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** $k \leq N_G$ 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
$$f^* = \sum_{j=1}^{M} \left(N(g_i^*, x_j) - y_j \right)^2$$
   using a local search procedure. In the proposed method the BFGS variant of Powell [69] was used.

The steps of the overall algorithm are outlined graphically in Figure 5.

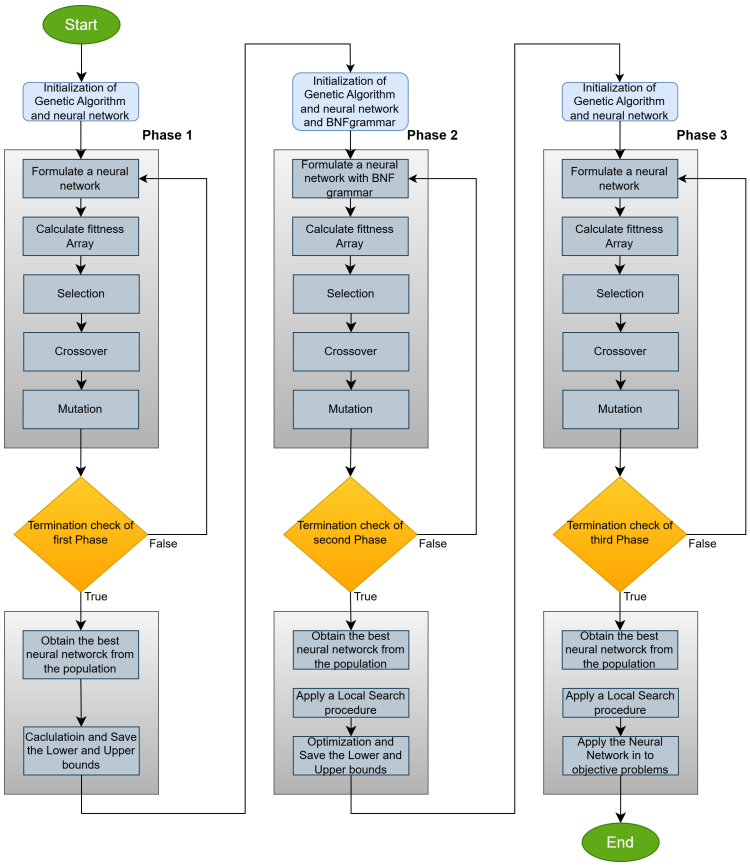


Figure 5. The steps of the proposed algorithm as a flowchart.

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) [70]
2. The Keel website, https://sci2s.ugr.es/keel/datasets.php (accessed on 24 November 2024) [71].
3. The Statlib URL ftp://lib.stat.cmu.edu/datasets/index (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 [72].
2. **Australian**, related to economic risk in bank transactions [73].
3. **Bands**, related to printing problems [74], with two distinct classes.
4. **Cleveland**, a medical dataset studied in many research papers [75,76].
5. **Circular** dataset, which is an artificial dataset.
6. **Dermatology**, a medical dataset related to dermatology problems [77] with 6 classes.
7. **Ecoli**, that is related to issues about proteins [78].
8. **Haberman**, a medical dataset used in the detection of breast cancer.
9. **Hayes-roth** dataset [79], a dataset with 3 classes.
10. **Heart**, a medical dataset about heart diseases [80] 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 [81].
14. **Ionosphere**, that used data from various measurements in the ionosphere [82,83].
15. **Liverdisorder**, which is a medical dataset [84,85] with two classes.
16. **Lymography** [86], which is a dataset with four distinct classes.
17. **Magic**, this dataset contains data from simulation regarding gamma particles [87].
18. **Mammographic**, a medical dataset used for the detection of breast cancer [88].
19. **Page Blocks** dataset [89], which is incorporated for the detection of page layout in documents.
20. **Parkinsons**, a dataset used to detect the presence of Parkinson's disease [90,91].
21. **Pima**, a medical dataset that used to detect the presence of diabetes [92].
22. **Phoneme**, where the purpose of this dataset is to distinguish between nasal and oral sounds.
23. **Popfailures**, a dataset related to climate measurements [93].
24. **Regions2**, used detect issues in the liver from various liver biopsy images [94].
25. **Ring**, a dataset related to some multivariate normal distributions.
26. **Saheart**, used to detect the presence of some heart diseases [95].
27. **Segment** dataset [96], 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 [97].
32. **Tae**, this dataset is related to evaluations of teaching performance.
33. **Transfusion**, which is a medical dataset [98].
34. **Wdbc**, a medical dataset related to the detection of breast cancer [99,100].
35. **Wine**, a dataset related to the quality of wines [101,102].
36. **EEG** dataset, which is a medical dataset related to EEG measurements [103,104]. 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 [105].

The number of classes for each used dataset are shown in Table 2.

Table 2. The column DATASET denotes the number of the dataset and the Number of Classes represent the distinct number of classes.

DATASET	Number of Classes
ALCOHOL	4
APPENDICITIS	2
AUSTRALIAN	2
BANDS	2
CLEVELAND	5
CIRCULAR	2
DERMATOLOGY	6
ECOLI	8
HABERMAN	2
HEARTATTACK	2
HAYES ROTH	3
HEART	2

Table 2. *Cont.*

DATASET	Number of Classes
HEPATITIS	20
HOUSEVOTES	2
IONOSPHERE	2
LIVERDISORDER	2
LYMOGRAPHY	4
MAGIC	2
MAMMOGRAPHIC	2
PAGE BLOCKS	5
PARKINSONS	2
PIMA	2
PHONEME	2
POPFAILURES	2
RING	2
SPIRAL	2
REGIONS2	5
SAHEART	2
SEGMENT	7
STATHEART	2
SPAMBASE	2
SPIRAL	2
STUDENT	2
TAE	3
TRANSFUSION	2
WDBC	2
WINE	3
Z_F_S	3
ZO_NF_S	3
ZONF_S	2
ZOO	7

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

1. **Abalone**, a dataset regarding the prediction of the age of abalones [106].
2. **Airfoil**, a dataset derived from NASA [107] 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 [108].
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 [109].
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 [110].
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.

The number of features for each regression dataset incorporated in the current work are shown in Table 3.

Table 3. The regression datasets used in the experiments. The column DATASET denotes the used dataset and the column FEATURES the number of features in this dataset.

DATASET	FEATURES
ABALONE	8
AIRFOIL	5
AUTO	5
BASEBALL	16
BK	4
BL	7
CONCRETE	8
DEE	6
EU	2
FA	18
FRIEDMAN	5
HO	13
HOUSING	13
LASER	4
LW	9
MORTGAGE	15
PLASTIC	2
PL	2
REALESTATE	5
QUAKE	3

Table 3. *Cont.*

DATASET	FEATURES
SN	11
STOCK	9
TREASURY	15
TZ	60
VE	7

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 128 GB of RAM. The used operating system was Debian Linux. In Table 4 the values used in all experimental parameters are listed.

Table 4. 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 5 outlines the experimental results for the classification datasets and in Table 6 the experimental results for the regression datasets are listed. The classification error is defined as:

$$E_C(M(x)) = 100 \times \frac{\sum_{i=1}^{N} (\text{class}(M(x_i)) - y_i)}{N} \tag{6}$$

where $M(x)$ denotes the underlying machine learning model and set T stands for the train dataset. Also, the regression error is defined as:

$$E_R(M(x)) = \frac{\sum_{i=1}^{N} (M(x_i) - y_i)^2}{N} \tag{7}$$

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 4.

5. The column NEAT defines the application of the NEAT method (NeuroEvolution of Augmenting Topologies) [111] as implemented in EvolutionNet package freely available from https://github.com/BiagioFesta/EvolutionNet (accessed on 31 December 2024). The number of used generations is the same as in the GENETIC algorithm.
6. The column RBF stands for the usage of a Radial Basis Function (RBF) network [112,113] with $H = 10$ processing nodes.
7. 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 4.
8. The row denoted as AVERAGE represents the average classification or regression error for all datasets.

Table 5. 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	NEAT	RBF	GNN
ALCOHOL	57.78%	41.50%	39.57%	66.80%	49.32%	16.63%
AUSTRALIAN	35.65%	38.13%	32.21%	31.98%	34.89%	14.69%
BALANCE	12.27%	8.64%	8.97%	23.14%	33.53%	7.98%
BANDS	36.92%	36.67%	34.92%	34.30%	37.17%	34.38%
CLEVELAND	67.55%	77.55%	51.60%	53.44%	67.10%	44.89%
CIRCULAR	19.95%	6.08%	5.99%	35.18%	5.98%	4.68%
DERMATOLOGY	26.14%	52.92%	30.58%	32.43%	62.34%	11.79%
ECOLI	64.43%	69.52%	54.67%	43.44%	59.48%	51.42%
HABERMAN	29.00%	29.34%	28.66%	24.04%	25.10%	27.01%
HAYES-ROTH	59.70%	37.33%	56.18%	50.15%	64.36%	34.99%
HEART	38.53%	39.44%	28.34%	39.27%	31.20%	14.83%
HEARTATTACK	45.55%	46.67%	29.03%	32.34%	29.00%	19.23%
HEPATITIS	68.13%	72.47%	62.12%	67.04%	64.63%	56.67%
HOUSEVOTES	7.48%	7.13%	6.62%	10.89%	6.13%	6.23%
IONOSPHERE	16.64%	15.29%	15.14%	19.67%	16.22%	15.71%
LIVERDISORDER	41.53%	42.59%	31.11%	30.67%	30.84%	31.89%
LYMOGRAPHY	39.79%	35.43%	28.42%	33.70%	25.50%	19.26%
MAGIC	40.55%	17.30%	21.75%	24.85%	21.28%	15.67%
MAMMOGRAPHIC	46.25%	17.24%	19.88%	22.85%	21.38%	16.33%
PARKINSONS	24.06%	27.58%	18.05%	18.56%	17.41%	11.53%
PAGE BLOCKS	34.27%	8.49%	6.84%	10.22%	10.09%	7.66%
PHONEME	29.43%	15.58%	15.55%	22.34%	23.32%	16.59%
PIMA	34.85%	35.59%	32.19%	34.51%	25.78%	25.81%
POPFAILURES	5.18%	5.24%	5.94%	7.05%	7.04%	5.25%
REGIONS2	29.85%	36.28%	29.39%	33.23%	38.29%	29.24%
RING	28.80%	29.44%	28.80%	30.85%	21.67%	21.68%
SAHEART	34.04%	37.48%	34.86%	34.51%	32.19%	29.92%
SEGMENT	49.75%	68.97%	57.72%	66.72%	59.68%	31.21%
SONAR	30.33%	25.85%	22.40%	34.10%	27.90%	22.80%

Table 5. *Cont.*

DATASET	ADAM	BFGS	GENETIC	NEAT	RBF	GNN
SPAMBASE	48.05%	18.16%	6.37%	35.77%	29.35%	6.17%
SPIRAL	47.67%	47.99%	48.66%	48.66%	44.87%	41.97%
STATHEART	44.04%	39.65%	27.25%	44.36%	31.36%	18.20%
STUDENT	5.13%	7.14%	5.61%	10.20%	5.49%	4.98%
TAE	60.20%	51.58%	49.84%	60.67%	60.02%	48.89%
TRANSFUSION	25.68%	25.84%	24.87%	24.87%	26.41%	23.34%
WDBC	35.35%	29.91%	8.56%	12.88%	7.27%	3.70%
WINE	29.40%	59.71%	19.20%	25.43%	31.41%	9.14%
Z_F_S	47.81%	39.37%	10.73%	38.41%	13.16%	8.35%
Z_O_N_F_S	78.79%	65.67%	64.81%	77.08%	48.70%	60.83%
ZO_NF_S	47.43%	43.04%	21.54%	43.75%	9.02%	6.29%
ZONF_S	11.99%	15.62%	4.36%	5.44%	4.03%	2.77%
ZOO	14.13%	10.70%	9.50%	20.27%	21.93%	6.03%
AVERAGE	**36.91%**	**34.19%**	**27.11%**	**33.72%**	**30.52%**	**21.11%**

Table 6. 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	NEAT	RBF	GNN
ABALONE	4.30	5.69	7.17	9.88	7.37	4.32
AIRFOIL	0.005	0.003	0.003	0.067	0.27	0.001
AUTO	70.84	60.97	12.18	56.06	17.87	8.69
BK	0.0252	0.28	0.027	0.15	0.02	0.039
BL	0.622	2.55	5.74	0.05	0.013	0.013
BASEBALL	77.90	119.63	103.60	100.39	93.02	47.25
CONCRETE	0.078	0.066	0.0099	0.081	0.011	0.004
DEE	0.63	2.36	1.013	1.51	0.17	0.212
EU	604.253	608.99	252.97	397.27	235.14	148.76
FA	0.048	0.426	0.025	0.19	0.015	0.097
FRIEDMAN	22.896	1.263	1.249	19.35	7.23	1.573
FY	0.038	0.22	0.65	0.08	0.04	0.121
HO	0.035	0.62	2.78	0.17	0.03	0.05
HOUSING	80.998	97.38	43.26	56.49	57.68	14.79
LASER	0.03	0.015	0.59	0.084	0.03	0.0029
LW	0.028	2.98	1.90	0.17	0.03	0.313
MORTGAGE	9.24	8.23	2.41	14.11	1.45	0.0222
PLASTIC	11.713	20.323	2.791	20.77	8.62	2.186
PY	0.321	0.578	0.56	0.075	0.012	0.111
PL	0.117	0.29	0.28	0.098	2.12	0.0205

Table 6. *Cont.*

DATASET	ADAM	BFGS	GENETIC	NEAT	RBF	GNN
REALESTATE	186.778	170.28	81.19	155.46	76.50	70.928
QUAKE	0.07	0.42	0.04	0.298	0.07	0.04
SN	0.026	0.40	2.95	0.174	0.027	0.025
STOCK	180.893	302.43	3.881	215.82	12.23	2.986
TREASURY	11.16	9.91	2.929	15.52	2.02	0.0633
TZ	0.07	3.27	5.38	0.098	0.036	0.652
VE	0.359	1.92	2.43	0.045	0.024	0.0493
AVERAGE	**46.795**	**52.648**	**19.926**	**39.42**	**19.34**	**11.234**

The statistical analysis of Table 5 highlights interesting patterns regarding model performance and the nature of the datasets. The GNN model achieves the lowest average error rate (21.11%), making it the overall best-performing model. This suggests that it is particularly effective across the dataset collection. In contrast, NEAT, despite an average error rate of 33.72%, excels in specific datasets such as Z_O_N_F_S, where it performs almost equivalently to GNN. The analysis reveals significant variations in error rates across datasets, indicating differing levels of complexity and challenges faced by the models. For instance, datasets like Z_O_N_F_S, where errors are low for all models, are likely characterized by well-defined patterns and minimal class overlap. On the other hand, datasets such as SPAMBASE and MAGIC appear more complex, with high error rates across many models. In SPAMBASE, GNN's performance (6.17%) is remarkable, indicating its superior capability in handling such data compared to other models. ADAM, with an average error rate of 36.91%, demonstrates consistent but not top-tier performance. This model seems reliable for medium-complexity datasets, such as BANDS and AUSTRALIAN, where it achieves comparable or better results than other models. However, it underperforms in high-dimensional datasets, such as SEGMENT, where BFGS significantly outperforms it. BFGS exhibits strong performance in problems like DERMATOLOGY, likely due to its adaptability to data with well-defined boundaries. The GENETIC model shows intriguing behavior, excelling in datasets such as ZONF_S and RING, but lagging in higher complexity problems like HEPATITIS. This indicates potential limitations in processing high-dimensional or noisy data. Conversely, RBF appears particularly sensitive to dataset-specific characteristics, performing well in certain datasets like BALANCE but falling short in problems such as SPIRAL and HOUSEVOTES. The comparison between models underscores the necessity of tailored model selection for each dataset. For instance, GNN would be the preferred choice for problems demanding the lowest possible error, while ADAM would be ideal for cases where stability and generalization are more critical. NEAT and GENETIC are suitable for problems with unique data structures, whereas RBF and BFGS prove useful in more specialized applications. The significant variability among models and datasets also highlights the complexity of selecting appropriate machine learning algorithms. While GNN stands out as the overall leader, each model may prove superior under specific conditions, depending on the nature and dimensionality of the data, the noise level, and the class overlap.

The statistical analysis of Table 6 indicates that the GNN model achieves the lowest average error (11.234), establishing it as the best performer across the datasets for regression problems. RBF follows with an average error of 19.34, while GENETIC ranks third with an average error of 19.926. The models ADAM, BFGS, and NEAT exhibit significantly higher average errors (46.795, 52.648, and 39.42, respectively), with BFGS showing the poorest

overall performance. A detailed dataset-wise analysis reveals significant differences in the models' performances. In the AIRFOIL dataset, GNN demonstrates exceptionally low error (0.001), proving to be highly suitable for this problem. Similarly, the RBF model performs well in problems such as BASEBALL and BL, recording errors of 93.02 and 0.013, respectively. In the CONCRETE dataset, GNN once again excels with an error of 0.004, and in the HOUSING dataset, it achieves a substantially lower error (14.79) compared to the other models, highlighting its adaptability to high-complexity data. ADAM exhibits significant performance variability, with extremely high errors in datasets such as STOCK (180.893) and REALESTATE (186.778), but relatively better performance in problems like MORTGAGE (9.24). NEAT shows similar instability, with high errors in datasets such as LASER (0.084) and TZ (0.098), while performing relatively well in PY (0.075). GENETIC, although ranked third overall, performs moderately in problems like HOUSING (43.26) and STOCK (3.881) but shows good performance in datasets like PLASTIC (2.791). RBF displays comparable instability, achieving low errors in problems such as BL (0.013) but significantly higher errors in datasets like PL (2.12). The comparison of models highlights GNN's superiority in most cases. GNN's lower errors in datasets such as LASER, STOCK, and TREASURY suggest that this model is particularly well-suited for problems requiring precision in continuous variables. RBF proves reliable in selected problems but with greater variability. ADAM and NEAT appear to face more constraints, particularly in datasets with high dimensionality or noise. This analysis underscores the necessity for careful selection of the appropriate regression model, considering the specific characteristics of each dataset. While GNN emerges as the best overall performer, the other models demonstrate strengths in specific applications, making their use suitable under particular conditions.

The statistical comparison between the used methods for the classification datasets is depicted in Figure 6 while for the regression datasets is graphically outlined in Figure 7.

The statistical analysis conducted using the Wilcoxon Test (Figure 6) to compare the models from Table 5 reveals significant differences in performance, as evidenced by the extremely low p-values. The p-value is an indicator of the probability that the observed differences in performance between models are due to random factors, with values below the conventional threshold (e.g., 0.05) being considered statistically significant. In this analysis, all p-values are exceptionally low, indicating that the differences between GNN and the other models are statistically robust. The comparison between GNN and ADAM yielded a p-value of 9.1×10^{-13}, indicating that the probability of the observed differences being random is nearly zero. This result highlights the clear superiority of GNN over ADAM. Similarly, the p-value of 2×10^{-11} for the comparison between GNN and BFGS demonstrates that GNN achieves significantly lower errors than BFGS, confirming the latter's inferior performance. The analysis of the pairs GNN and GENETIC ($p = 5.2 \times 10^{-9}$) and GNN and NEAT ($p = 5.7 \times 10^{-10}$) indicates that although the GENETIC and NEAT models have specific strengths in individual datasets, their overall performance does not approach that of GNN. The differences are statistically significant, clearly establishing GNN as the superior choice under the conditions examined. Lastly, the comparison between GNN and RBF ($p = 1.8 \times 10^{-8}$) demonstrates that GNN also outperforms RBF, despite RBF showing strong results in selected datasets. This analysis provides a comprehensive view of GNN's superiority, not only in terms of average error but also from a statistical perspective, confirming that the performance differences are not products of random variations in the data. This substantial differentiation underscores the value of GNN as a particularly effective model for classification problems and highlights the need for further exploration of the structure and characteristics that contribute to its performance. Moreover, it reinforces the importance of applying statistical techniques such as the Wilcoxon Test to make informed decisions when comparing different machine learning algorithms.

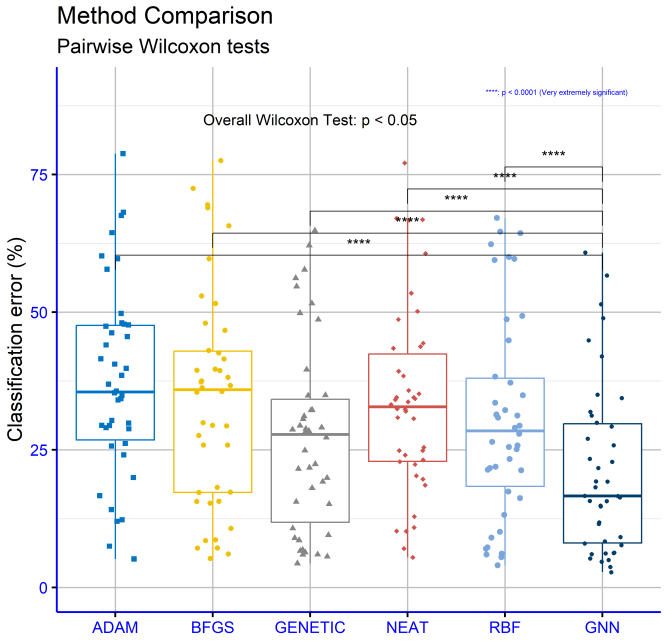


Figure 6. Statistical comparison between the used methods for the classification datasets. Every method is denoted with different color.

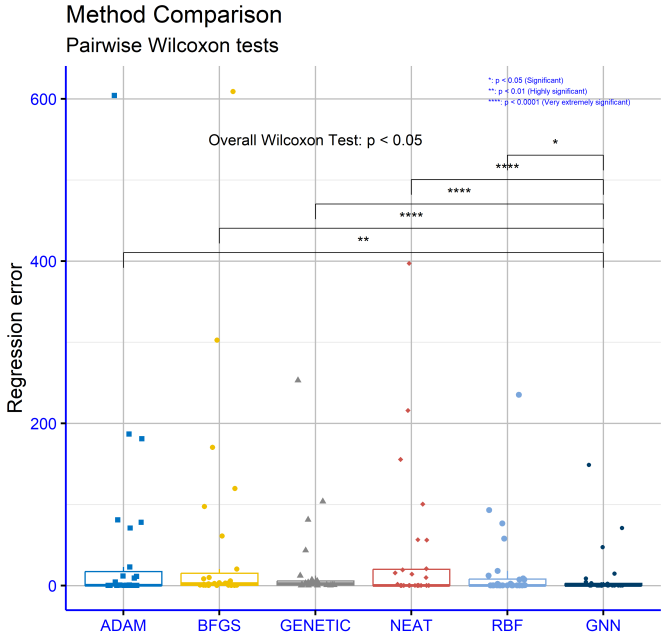


Figure 7. Statistical comparison between the used methods for the regression datasets. Every method is denoted with different color.

In Figure 7, the comparison between GNN and ADAM yields a p-value of 0.0011, indicating that GNN demonstrates significantly better performance than ADAM. Although this value is higher than the others, it remains statistically significant, underscoring GNN's superiority in terms of error rates. The difference between GNN and BFGS is even more pronounced, with a p-value of 2.8×10^{-7}, signifying that GNN outperforms BFGS with very high statistical power. The p-values for the comparisons of GNN with the models GENETIC ($p = 3.9 \times 10^{-5}$), NEAT ($p = 5.5 \times 10^{-5}$), and RBF ($p = 0.013$) further strengthen the evidence of GNN's superiority. While RBF appears to exhibit relatively better performance than the other models compared with GNN, the p-value of 0.013 indicates that the difference is still

statistically significant and supports the preference for GNN. These results confirm GNN's superiority in regression problems concerning its overall performance, suggesting that the lower error rates it achieves are not due to random factors but reflect its effectiveness. This analysis highlights the importance of using statistical tools like the Wilcoxon Test to evaluate the performance of machine learning algorithms and emphasizes the need for a tailored model selection process, taking into account the specific characteristics of each problem.

4. Conclusions

The article highlights a novel methodology for training artificial neural networks using Backus-Naur Form grammars combined with Grammatical Evolution. This approach provides a structured framework for defining and optimizing neural network architectures, enhancing their adaptability and performance across various problem domains. The proposed three-phase process includes parameter specification, evolutionary optimization of parameter bounds, and final training with specialized optimization algorithms. The methodology demonstrates strong versatility and achieves high performance across diverse architectures such as recurrent neural networks and convolutional neural networks, handling both classification and regression tasks effectively. The use of Backus-Naur Form grammars constrains the parameter search space, reducing computational complexity and mitigating risks associated with local minima or suboptimal solutions. The experimental findings reveal that GNN outperforms traditional optimization methods, including ADAM, BFGS, and GENETIC, consistently achieving lower error rates. This result highlights the effectiveness of the proposed method in initializing and optimizing network parameters. GNN's generalization capabilities and adaptability to different datasets further underscore its potential in real-world applications requiring precision and reliability. The results confirm that the lower error rates achieved by GNN are not due to random factors but reflect the robustness of the method. Future explorations could extend the methodology to more complex deep learning architectures, including transformers and graph neural networks, where the dimensionality and complexity of parameters are significantly higher. Dynamic and non-stationary environments such as real-time data streams and adaptive control systems present another promising area of application.

Incorporating the methodology into neural architecture search frameworks could automate the discovery of optimal architectures, leveraging Backus-Naur Form grammars to balance exploration and exploitation efficiently. Multimodal data scenarios that combine textual, visual, and auditory inputs could also benefit from this approach by enabling hybrid architectures capable of processing diverse information types effectively. Addressing scalability concerns through parallel and distributed computing frameworks could enable the method to handle large-scale datasets and complex architectures. Domain-specific grammars tailored to particular problem types, such as time-series forecasting or anomaly detection, could further refine the applicability and efficiency of the method. Combining the evolutionary optimization process with gradient-based techniques could provide a hybrid approach, enhancing convergence speed while maintaining flexibility in exploring the parameter space.

Empirical studies across broader datasets and theoretical investigations into the convergence properties and computational complexity of the method would offer deeper insights into its strengths and limitations. The methodology represents a significant advancement in neural network training, offering robust adaptability and precision across diverse applications. Its potential for scalability, domain-specific customizations, and hybrid optimization approaches underscores its relevance for addressing increasingly complex and dynamic challenges in machine learning.

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.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

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. Mjahed, M. The use of clustering techniques for the classification of high energy physics data. *Nucl. Instrum. Methods Phys. Res. Sect. A Accel. Spectrometers Detect. Assoc. Equip.* **2006**, *559*, 199–202. [CrossRef]
2. Andrews, M.; Paulini, M.; Gleyzer, S.; Poczos, B. End-to-End Event Classification of High-Energy Physics Data. *J. Phys. Conf. Ser.* **2018**, *1085*, 042022. [CrossRef]
3. He, P.; Xu, C.J.; Liang, Y.Z.; Fang, K.T. Improving the classification accuracy in chemistry via boosting technique. *Chemom. Intell. Lab. Syst.* **2004**, *70*, 39–46. [CrossRef]
4. Aguiar, J.A.; Gong, M.L.; Tasdizen, T. Crystallographic prediction from diffraction and chemistry data for higher throughput classification using machine learning. *Comput. Mater. Sci.* **2020**, *173*, 109409. [CrossRef]
5. Kaastra, I.; Boyd, M. Designing a neural network for forecasting financial and economic time series. *Neurocomputing* **1996**, *10*, 215–236. [CrossRef]
6. Hafezi, R.; Shahrabi, J.; Hadavandi, E. A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price. *Appl. Soft Comput.* **2015**, *29*, 196–210. [CrossRef]
7. Yadav, S.S.; Jadhav, S.M. Deep convolutional neural network based medical image classification for disease diagnosis. *J. Big Data* **2019**, *6*, 113. [CrossRef]
8. Qing, L.; Linhong, W.; Xuehai, D. A Novel Neural Network-Based Method for Medical Text Classification. *Future Internet* **2019**, *11*, 255. [CrossRef]
9. Bishop, C. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, 1995.
10. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
11. Egmont-Petersen, M.; de Ridder, D.; Handels, H. Image processing with neural networks—A review. *Pattern Recognit.* **2002**, *35*, 2279–2301. [CrossRef]
12. Khashei, M.; Bijari, M. An artificial neural network (p,d,q) model for time series forecasting. *Expert Syst. Appl.* **2010**, *37*, 479–489. [CrossRef]
13. Mellit, A.; Massi Pavan, A. A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy. *Sol. Energy* **2010**, *84*, 807–821. [CrossRef]
14. Amato, F.; López, A.; Peña-Méndez, E.M.; Vaňhara, P.; Hampl, A.; Havel, J. Artificial neural networks in medical diagnosis. *J. Appl. Biomed.* **2013**, *11*, 47–58. [CrossRef]
15. Shirvany, Y.; Hayati, M.; Moradian, R. Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations. *Appl. Soft Comput.* **2009**, *9*, 20–29. [CrossRef]
16. Malek, A.; Shekari Beidokhti, R. Numerical solution for high order differential equations using a hybrid neural network—Optimization method. *Appl. Math. Comput.* **2006**, *183*, 260–271. [CrossRef]
17. Topuz, A. Predicting moisture content of agricultural products using artificial neural networks. *Adv. Eng. Softw.* **2010**, *41*, 464–470. [CrossRef]
18. Escamilla-García, A.; Soto-Zarazúa, G.M.; Toledano-Ayala, M.; Rivas-Araiza, E.; Gastélum-Barrios, A. Applications of Artificial Neural Networks in Greenhouse Technology and Overview for Smart Agriculture Development. *Appl. Sci.* **2020**, *10*, 3835. [CrossRef]
19. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]

20. Riedmiller, M.; Braun, H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm. In Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, USA, 28 March–1 April 1993; pp. 586–591.
21. Kingma, D.P.; Ba, J.L. ADAM:A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
22. Robitaille, B.; Marcos, B.; Veillette, M.; Payre, G. Modified quasi-Newton methods for training neural networks. *Comput. Chem. Eng.* **1996**, *20*, 1133–1140. [CrossRef]
23. Sexton, R.S.; Alidaee, B.; Dorsey, R.E.; Johnson, J.D. Global optimization for artificial neural networks: A tabu search application. *Eur. J. Oper. Res.* **1998**, *106*, 570–584. [CrossRef]
24. Yamazaki, A.; de Souto, M.C.P.; Ludermir, T.B. Optimization of neural network weights and architectures for odor recognition using simulated annealing. In Proceedings of the 2002 International Joint Conference on Neural Networks, IJCNN'02, Honolulu, HI, USA, 12–17 May 2002; Volume 1, pp. 547–552.
25. Leung, F.H.F.; Lam, H.K.; Ling, S.H.; Tam, P.K. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Trans. Neural Netw.* **2003**, *14*, 79–88. [CrossRef] [PubMed]
26. Zhang, C.; Shao, H.; Li, Y. Particle swarm optimization for evolving artificial neural network. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Nashville, TN, USA, 8–11 October 2000; pp. 2487–2490.
27. lonen, J.; Kamarainen, J.K.; Lampinen, J. Differential Evolution Training Algorithm for Feed-Forward Neural Networks. *Neural Processing Lett.* **2003**, *17*, 93–105.
28. Salama, K.M.; Abdelbar, A.M. Learning neural network structures with ant colony algorithms. *Swarm Intell* **2015**, *9*, 229–265. [CrossRef]
29. Askarzadeh, A.; Rezazadeh, A. Artificial neural network training using a new efficient optimization algorithm. *Appl. Soft Comput.* **2013**, *13*, 1206–1213. [CrossRef]
30. Benardos, P.G.; Vosniakos, G.-C. Optimizing feedforward artificial neural network architecture. *Eng. Appl. Artif. Intell.* **2007**, 20, 365–382. [CrossRef]
31. Karaboga, D.; Akay, B. Artificial Bee Colony (ABC) Algorithm on Training Artificial Neural Networks. In Proceedings of the 2007 IEEE 15th Signal Processing and Communications Applications, Eskisehir, Turkey, 11–13 June 2007; pp. 1–4. [CrossRef]
32. Chen, J.F.; Do, Q.H.; Hsieh, H.N. Training Artificial Neural Networks by a Hybrid PSO-CS Algorithm. *Algorithms* **2015**, 8, 292–308. [CrossRef]
33. Yang, X.S.; Deb, S. Engineering Optimisation by Cuckoo Search. *Int. J. Math. Model. Numer. Optim.* **2010**, *1*, 330–343. [CrossRef]
34. Scanzio, S.; Cumani, S.; Gemello, R.; Mana, F.; Laface, P. Parallel implementation of Artificial Neural Network training for speech recognition. *Pattern Recognit. Lett.* **2010**, *31*, 1302–1309. [CrossRef]
35. Sierra-Canto, X.; Madera-Ramirez, F.; Uc-Cetina, V. Parallel Training of a Back-Propagation Neural Network Using CUDA. In Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications, Washington, DC, USA, 12–14 December 2010; pp. 307–312. [CrossRef]
36. Åström, F.; Koker, R. A parallel neural network approach to prediction of Parkinson's Disease. *Expert Syst. Appl.* **2011**, *38*, 12470–12474. [CrossRef]
37. Ivanova, I.; Kubat, M. Initialization of neural networks by means of decision trees. *Knowl.-Based Syst.* **1995**, *8*, 333–344. [CrossRef]
38. Yam, J.Y.F.; Chow, T.W.S. A weight initialization method for improving training speed in feedforward neural network. *Neurocomputing* **2000**, *30*, 219–232. [CrossRef]
39. Chumachenko, K.; Iosifidis, A.; Gabbouj, M. Feedforward neural networks initialization based on discriminant learning. *Neural Netw.* **2022**, *146*, 220–229. [CrossRef]
40. Itano, F.; Sousa, M.A.d.d.; Del-Moral-Hernandez, E. Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
41. Sodhi, S.S.; Chandra, P. Interval based Weight Initialization Method for Sigmoidal Feedforward Artificial Neural Networks. *AASRI Procedia* **2014**, *6*, 19–25. [CrossRef]
42. de Sousa, C.A.R. An overview on weight initialization methods for feedforward neural networks. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 52–59. [CrossRef]
43. Nowlan, S.J.; Hinton, G.E. Simplifying neural networks by soft weight sharing. *Neural Comput.* **1992**, *4*, 473–493. [CrossRef]
44. Hanson, S.J.; Pratt, L.Y. Comparing biases for minimal network construction with back propagation. In *Advances in Neural Information Processing Systems*; Touretzky, D.S., Ed.; Morgan Kaufmann: San Mateo, CA, USA, 1989; Volume 1, pp. 177–185.
45. Mozer, M.C.; Smolensky, P. Skeletonization: A technique for trimming the fat from a network via relevance assesment. In *Advances in Neural Processing Systems*; Touretzky, D.S., Ed.; Morgan Kaufmann: San Mateo, CA, USA, 1989; Volume 1, pp. 107–115.
46. Augasta, M.; Kathirvalavakumar, T. Pruning algorithms of neural networks—A comparative study. *Cent. Eur. Comput. Sci.* **2003**, *3*, 105–115. [CrossRef]

47. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
48. Gupta, A.; Lam, S.M. Weight decay backpropagation for noisy data. *Neural Netw.* **1998**, *11*, 1127–1138. [CrossRef]
49. Carvalho, M.; Ludermir, T.B. Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay. In Proceedings of the 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06), Rio de Janeiro, Brazil, 13–15 December 2006; p. 5.
50. Treadgold, N.K.; Gedeon, T.D. Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm. *IEEE Trans. Neural Netw.* **1998**, *9*, 662–668. [CrossRef]
51. Shahjahan, M.D.; Kazuyuki, M. Neural network training algorithm with possitive correlation. *IEEE Trans. Inf. Syst.* **2005**, *88*, 2399–2409. [CrossRef]
52. Mirjalili, S.; Mirjalili, S. Genetic algorithm. In *Evolutionary Algorithms and Neural Networks: Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 43–55.
53. O'Neill, M.; Ryan, C. Grammatical evolution. *IEEE Trans. Evol. Comput.* **2001**, *5*, 349–358. [CrossRef]
54. Backus, J.W. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. In Proceedings of the International Conference on Information Processing, UNESCO, Paris, France, 15–20 June 1959; pp. 125–132.
55. Lukoševičius, M.; Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* **2009**, *3*, 127–149. [CrossRef]
56. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Chen, T. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
57. Puente, A.O.; Alfonso, R.S.; Moreno, M.A. Automatic composition of music by means of grammatical evolution. In Proceedings of the APL '02: Proceedings of the 2002 Conference on APL: Array Processing Languages: Lore, Problems, and Applications, Madrid, Spain, 22–25 July 2002; pp. 148–155.
58. Galván-López, E.; Swafford, J.M.; O'Neill, M.; Brabazon, A. Evolving a Ms. PacMan Controller Using Grammatical Evolution. In *Applications of Evolutionary Computation*; EvoApplications 2010. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6024.
59. Shaker, N.; Nicolau, M.; Yannakakis, G.N.; Togelius, J.; O'Neill, M. Evolving levels for Super Mario Bros using grammatical evolution. In Proceedings of the 2012 IEEE Conference on Computational Intelligence and Games (CIG), Granada, Spain, 11–14 September 2012; pp. 304–331.
60. Ortega, A.; Dalhoum, A.A.; Alfonseca, M. Grammatical evolution to design fractal curves with a given dimension. *IBM J. Res. Dev.* **2003**, *47*, 483–493. [CrossRef]
61. Dempsey, I.; Neill, M.O.; Brabazon, A. Constant creation in grammatical evolution. *Int. J. Innov. Comput. Appl.* **2007**, *1*, 23–38. [CrossRef]
62. Burbidge, R.; Walker, J.H.; Wilson, M.S. Grammatical evolution of a robot controller. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; pp. 357–362. [CrossRef]
63. Peabody, C.; Seitzer, J. GEF: A self-programming robot using grammatical evolution. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; Volume 29.
64. Hidalgo, J.I.; Colmenar, J.M.; Risco-Martin, J.L.; Cuesta-Infante, A.; Maqueda, E.; Botella, M.; Rubio, J.A. Modeling glycemia in humans by means of Grammatical Evolution. *Appl. Soft Comput.* **2014**, *20*, 40–53. [CrossRef]
65. Araujo, L.; Martinez-Romo, J.; Duque, A. Discovering taxonomies in Wikipedia by means of grammatical evolution. *Soft Comput.* **2018**, *22*, 2907–2919. [CrossRef]
66. Martín, C.; Quintana, D.; Isasi, P. Grammatical Evolution-based ensembles for algorithmic trading. *Appl. Soft Comput.* **2019**, *84*, 105713. [CrossRef]
67. Tsoulos, I.G.; Gavrilis, D.; Glavas, E. Neural network construction and training using grammatical evolution. *Neurocomputing* **2008**, *72*, 269–277. [CrossRef]
68. Kaelo, P.; Ali, M.M. Integrated crossover rules in real coded genetic algorithms. *Eur. J. Oper. Res.* **2007**, *176*, 60–76. [CrossRef]
69. Powell, M.J.D. A Tolerant Algorithm for Linearly Constrained Optimization Calculations. *Math. Program.* **1989**, *45*, 547–566. [CrossRef]
70. Kelly, M.; Longjohn, R.; Nottingham, K. The UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu (accessed on 31 December 2024).
71. Alcalá-Fdez, J.; Fernandez, A.; Luengo, J.; Derrac, J.; García, S.; Sánchez, L.; Herrera, F. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *J. -Mult.-Valued Log. Soft Comput.* **2011**, *17*, 255–287.
72. 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. [CrossRef]
73. Quinlan, J.R. Simplifying Decision Trees. *Int. J. Man-Mach. Stud.* **1987**, *27*, 221–234. [CrossRef]
74. Evans, B.; Fisher, D. Overcoming process delays with decision tree induction. *IEEE Expert* **1994**, *9*, 60–66. [CrossRef]

75. Zhou, Z.H.; Jiang, Y. NeC4.5: neural ensemble based C4.5. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 770–773. [CrossRef]
76. Setiono, R.; Leow, W.K. FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks. *Appl. Intell.* **2000**, *12*, 15–25. [CrossRef]
77. Demiroz, G.; Govenir, H.A.; Ilter, N. Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals. *Artif. Intell. Med.* **1998**, *13*, 147–165.
78. Horton, P.; Nakai, K. A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins. *Proc. Int. Conf. Intell. Syst. Mol. Biol.* **1996**, *4*, 109–115. [PubMed]
79. Hayes-Roth, B.; Hayes-Roth, F. Concept learning and the recognition and classification of exemplars. *J. Verbal Learn. Verbal Behav.* **1977**, *16*, 321–338. [CrossRef]
80. Kononenko, I.; Šimec, E.; Robnik-Šikonja, M. Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF. *Appl. Intell.* **1997**, *7*, 39–55. [CrossRef]
81. French, R.M.; Chater, N. Using noise to compute error surfaces in connectionist networks: a novel means of reducing catastrophic forgetting. *Neural Comput.* **2002**, *14*, 1755–1769. [CrossRef] [PubMed]
82. Dy, J.G.; Brodley, C.E. Feature Selection for Unsupervised Learning. *J. Mach. Learn. Res.* **2004**, *5*, 845–889.
83. Perantonis, S.J.; Virvilis, V. Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis. *Neural Process. Lett.* **1999**, *10*, 243–252. [CrossRef]
84. Garcke, J.; Griebel, M. Classification with sparse grids using simplicial basis functions. *Intell. Data Anal.* **2002**, *6*, 483–502. [CrossRef]
85. Mcdermott, J.; Forsyth, R.S. Diagnosing a disorder in a classification benchmark. *Pattern Recognit. Lett.* **2016**, *73*, 41–43. [CrossRef]
86. Cestnik, G.; Konenenko, I.; Bratko, I. Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. In *Progress in Machine Learning*; Bratko, I., Lavrac, N., Eds.; Sigma Press: Wilmslow, UK, 1987; pp. 31–45.
87. Heck, D.; Knapp, J.; Capdevielle, J.N.; Schatz, G.; Thouw, T. CORSIKA: A Monte Carlo Code to Simulate Extensive Air Showers. 1998. Available online: https://digbib.bibliothek.kit.edu/volltexte/fzk/6019/6019.pdf (accessed on 8 December 2024).
88. Elter, M.; Schulz-Wendtland, R.; Wittenberg, T. The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process. *Med. Phys.* **2007**, *34*, 4164–4172. [CrossRef]
89. Malerba, F.E.F.D.; Semeraro, G. Multistrategy Learning for Document Recognition. *Appl. Artif. Intell.* **1994**, *8*, 33–84.
90. Little, M.; Mcsharry, P.; Roberts, S.; Costello, D.; Moroz, I. Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection. *BioMed Eng. OnLine* **2007**, *6*, 23. [CrossRef] [PubMed]
91. Little, M.A.; McSharry, P.E.; Hunter, E.J.; Spielman, J.; Ramig, L.O. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Trans. Biomed. Eng.* **2009**, *56*, 1015–1022. [CrossRef]
92. Smith, J.W.; Everhart, J.E.; Dickson, W.C.; Knowler, W.C.; Johannes, R.S. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care*; IEEE Computer Society Press: Piscataway, NJ, USA, 1988; pp. 261–265.
93. Lucas, D.D.; Klein, R.; Tannahill, J.; Ivanova, D.; Brandon, S.; Domyancic, D.; Zhang, Y. Failure analysis of parameter-induced simulation crashes in climate models. *Geosci. Dev.* **2013**, *6*, 1157–1171. [CrossRef]
94. Giannakeas, N.; Tsipouras, M.G.; Tzallas, A.T.; Kyriakidi, K.; Tsianou, Z.E.; Manousou, P.; Hall, A.; Karvounis, E.C.; Tsianos, V.; Tsianos, E. A clustering based method for collagen proportional area extraction in liver biopsy images. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, Milan, Italy, 25–29 August 2015; pp. 3097–3100.
95. Hastie, T.; Tibshirani, R. Non-parametric logistic and proportional odds regression. *JRSS-C (Appl. Stat.)* **1987**, *36*, 260–276. [CrossRef]
96. Dash, M.; Liu, H.; Scheuermann, P.; Tan, K.L. Fast hierarchical clustering and its validation. *Data Knowl. Eng.* **2003**, *44*, 109–138. [CrossRef]
97. Cortez, P.; Silva, A.M.G. Using data mining to predict secondary school student performance. In Proceedings of the 5th Future Business Technology Conference (FUBUTEC 2008), EUROSIS-ETI, Porto, Portugal, 9–11 April 2008; pp. 5–12.
98. Yeh, I.C.; Yang, K.J.; Ting, T.M. Knowledge discovery on RFM model using Bernoulli sequence. *Expert. Appl.* **2009**, *36*, 5866–5871. [CrossRef]
99. Jeyasingh, S.; Veluchamy, M. Modified bat algorithm for feature selection with the wisconsin diagnosis breast cancer (WDBC) dataset. *Asian Pac. J. Cancer Prev. APJCP* **2017**, *18*, 1257. [PubMed]
100. Alshayeji, M.H.; Ellethy, H.; Gupta, R. Computer-aided detection of breast cancer on the Wisconsin dataset: An artificial neural networks approach. *Biomed. Signal Processing Control.* **2022**, *71*, 103141. [CrossRef]
101. Raymer, M.; Doom, T.E.; Kuhn, L.A.; Punch, W.F. Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. *IEEE Trans. Syst. Cybern. Part B Cybern. Publ. IEEE Syst. Man, Cybern. Soc.* **2003**, *33*, 802–813. [CrossRef]

102. Zhong, P.; Fukushima, M. Regularized nonsmooth Newton method for multi-class support vector machines. *Optim. Methods Softw.* **2007**, *22*, 225–236. [CrossRef]
103. Andrzejak, R.G.; Lehnertz, K.; Mormann, F.; Rieke, C.; David, P.; Elger, C.E. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Phys. Rev. E* **2001**, *64*, 061907. [CrossRef] [PubMed]
104. Tzallas, A.T.; Tsipouras, M.G.; Fotiadis, D.I. Automatic Seizure Detection Based on Time-Frequency Analysis and Artificial Neural Networks. *Comput. Intell. Neurosci.* **2007**, *2007*, 80510. [CrossRef]
105. Koivisto, M.; Sood, K. Exact Bayesian Structure Discovery in Bayesian Networks. *J. Mach. Learn. Res.* **2004**, *5*, 549–573.
106. Nash, W.J.; Sellers, T.L.; Talbot, S.R.; Cawthor, A.J.; Ford, W.B. The Population Biology of Abalone (_Haliotis_ species) in Tasmania. In *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
107. 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).
108. Yeh, I.C. Modeling of strength of high performance concrete using artificial neural networks. *Cem. Concr. Res.* **1998**, *28*, 1797–1808. [CrossRef]
109. Friedman, J. Multivariate Adaptative Regression Splines. *Ann. Stat.* **1991**, *19*, 141.
110. Harrison, D.; Rubinfeld, D.L. Hedonic prices and the demand for clean ai. *J. Environ. Econ. Manag.* **1978**, *5*, 81–102. [CrossRef]
111. Stanley, K.O.; Miikkulainen, R. Evolving Neural Networks through Augmenting Topologies. *Evol. Comput.* **2002**, *10*, 99–127. [CrossRef] [PubMed]
112. Park, J.; Sandberg, I.W. Universal Approximation Using Radial-Basis-Function Networks. *Neural Comput.* **1991**, *3*, 246–257. [CrossRef]
113. Montazer, G.A.; Giveki, D.; Karami, M.; Rastegar, H. Radial basis function neural networks: A review. *Comput. Rev. J.* **2018**, *1*, 52–74.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.