Article

Combining constructed artificial neural networks with parameter constraint techniques to achieve better generalization properties

Ioannis G. Tsoulos[1,*], Vasileios Charilogis[2], Dimitrios Tsalikakis[3]

[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
[3] Department of Engineering Informatics and Telecommunications, University of Western Macedonia, 50100 Kozani, Greece; tsalikakis@gmail.com
* Correspondence: itsoulos@uoi.gr

Abstract: This study presents a novel hybrid approach combining grammatical evolution with constrained genetic algorithms to overcome key limitations in automated neural network design. The proposed method addresses two critical challenges: the tendency of grammatical evolution to converge to suboptimal architectures due to local optima, and the common overfitting problems in evolved networks. Our solution employs grammatical evolution for initial architecture generation while implementing a specialized genetic algorithm that simultaneously optimizes network parameters within dynamically adjusted bounds. The genetic component incorporates innovative penalty mechanisms in its fitness function to control neuron activation patterns and prevent overfitting. Comprehensive testing across 53 diverse datasets shows our method achieves superior performance compared to traditional optimization techniques, with an average classification error of 21.18% versus 36.45% for ADAM, while maintaining better generalization capabilities. The constrained optimization approach proves particularly effective in preventing premature convergence, and the penalty system successfully mitigates overfitting even in complex, high-dimensional problems. Statistical validation confirms these improvements are significant (p < 1.1e-08) and consistent across multiple domains including medical diagnosis, financial prediction, and physical system modeling. This work provides a robust framework for automated neural network construction that balances architectural innovation with parameter optimization while addressing fundamental challenges in evolutionary machine learning.

Keywords: Grammatical Evolution; Genetic Programming; Neural networks; Local Optimization

Citation: Tsoulos, I.G.; Charilogis, V.; Tsalikakis D. Combining constructed artificial neural networks with parameter constraint techniques to achieve better generalization properties. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

A basic machine learning technique with a wide range of applications in data classification and regression problems is artificial neural networks [1,2]. Artificial neural networks are parametric machine learning model, in which learning is achieved by effectively adjusting their parameters through any optimization technique. The optimization procedure minimizes the so - called training error of an artificial neural network and it is defined as:

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

In this equation the function $N\left(\overrightarrow{x},\overrightarrow{w}\right)$ represents the artificial neural network which is applied on a vector $\overrightarrow{x}$ and the vector $\overrightarrow{w}$ denotes the parameter vector of the neural network. The set $\left(\overrightarrow{x_i},y_i\right),\ i=1,...,M$ represents the training set of the objective problem and the values y_i are the expected outputs for each pattern $\overrightarrow{x_i}$.

Artificial neural networks have been applied in a wide series of problems appeared in real - world problems, such as image processing [3], time series forecasting [4], credit card analysis [5], problems derived from physics [6,7] etc. Due to the widespread use of these machine learning models, a number of techniques have been proposed to minimize the equation 1, such as the Back Propagation algorithm [8,9], the RPROP algorithm [10, 11], the ADAM optimization method [12] etc. Also, recently a series of more advanced global optimization methods have been proposed to tackle the training of neural networks. Among them one can locate the incorporation of Genetic Algorithms [13], the usage of the Particle Swarm Optimization (PSO) method [14], the Simulated Annealing method [15], the Differential Evolution technique [16], the Artificial Bee Colony (ABC) method [17] etc. Furthermore, Sexton et al suggested the usage of the tabu search algorithm for optimal neural network training [18], Zhang et al proposed a hybrid algorithm that incorporated the PSO method and the Back Propagation algorithm to efficient train artificial neural networks [19]. Also, recently Zhao et al introduced a new Cascaded Forward Algorithm to train artificial neural networks [20]. Furthermore, due to the rapid spread of the use of parallel computing techniques, a series of computational techniques have emerged that exploit parallel computing structures for faster training of artificial neural networks [21,22].

However, the above techniques, although extremely effective, nevertheless have a number of problems such as, for example, trapping in local minima of the error function or the phenomenon of overifitting, where the artificial neural network exhibits reduced performance when applied to data that was not present during the training process. The overfitting problem has been studied by many researchers that have proposed a series of methods to handle this problem, such as weight sharing [23,24], pruning [25,26], early stopping [27,28], weight decaying [29,30] etc. Also, many researchers propose as a solution to the above problem the dynamic creation of the architecture of artificial neural networks using programming techniques. For example the Genetic Algorithms were proposed to create dynamically the optimal architecture of neural networks [31,32] or the PSO method [33]. Siebel et al, suggested the usage of evolutionary reinforcement learning for the optimal design of artificial neural networks [34]. Also, Jaafra et al provided a review on the usage of Reinforcement learning for neural architecture search [35]. In the same direction of research, Pham et al proposed a method for efficient identification of the architecture of neural networks through parameters sharing [36]. Also, the method of Stochastic Neural Architecture search was suggested by Xie et al in a recent publication [37]. Moreover, Zhou et al introduced a Bayesian approach for neural architecture search [38].

Recently, genetic algorithms have been incorporated to identify the optimal set of parameters of neural networks for drug discovery [39]. Kim et al proposed [40] genetic algorithms to train neural networks for predicting preliminary cost estimates. Moreover, Kalogirou proposed the usage of genetic algorithms for effective training of neural networks for the optimization of solar systems [41]. The ability of neural networks to perform feature selection with the assistance of genetic algorithms was also studied in the work of Tong et al [42]. Recently, Ruehle provided a study of the string landscape using genetic algorithms to train artificial neural networks [43].

A method that was proposed relatively recently and it is based on Grammatical Evolution [44], dynamically identifies both the optimal architecture of artificial neural networks and the optimal values of its parameters [45]. This method has been applied in a series of problems in the recent literature, such as problems presented in chemistry [46], identification of the solution of differential equations [47], medical problems [48], problems related to education [49], autism screening [50] etc. A key advantage of this technique is that it can isolate from the initial features of the problem those that are most important in training the model, thus significantly reducing the required number of parameters that need to be identified.

However, the method of constructing artificial neural networks can easily get trapped in local minima of the training error since it does not have any technique to avoid them. Furthermore, although the method can get quite close to a minimum of the training error, it often does not reach it since there is no technique in the method to train the generated parameters. In this technique, it is proposed to enhance the original method of constructing artificial neural networks by periodically applying a modified genetic algorithm to randomly selected chromosomes of Grammatical Evolution. This modified genetic algorithm preserves the architecture created by the Grammatical Evolution method and effectively locates the parameters of the artificial neural network by reducing the training error. In addition, the proposed genetic algorithm through appropriate penalty factors imposed on the fitness function prevents the artificial neural network from overfitting.

The motivation of the proposed method is the need to address two main challenges in training artificial neural networks: getting trapped in local minima and the phenomenon of overfitting. Getting trapped in local minima limits the model's ability to minimize training error, leading to poor performance on test data. Overfitting similarly reduces generalization, as the model adapts excessively to the training data. The proposed method combines Grammatical Evolution with a modified genetic algorithm to address these problems. Grammatical Evolution is used for the dynamic construction of the neural network's architecture, while the genetic algorithm optimizes the network's parameters while preserving its structure. Additionally, penalty factors are introduced in the cost function to prevent overfitting. A key innovation is the use of an algorithm that measures the network's tendency to lose generalization capability when neuron activations become saturated. This is achieved by monitoring the input values of the sigmoid function and imposing penalties when they exceed a specified range. Experimental tests showed that the method outperforms other techniques such as ADAM, BFGS, and RBF networks, in both classification and regression problems. Statistical analysis confirmed the significant improvement in performance, with very low p-values in the comparisons.

For the suggested work, the main contributions are as follows:

1. A novel hybrid framework that effectively combines grammatical evolution for neural architecture search with constrained genetic algorithms for parameter optimization, addressing both structural design and weight training simultaneously.
2. An innovative penalty mechanism within the genetic algorithm's fitness function that dynamically monitors and controls neuron activation patterns to prevent overfitting, demonstrated to reduce test error by an average of 15.27% compared to standard approaches.
3. Comprehensive experimental validation across 53 diverse datasets showing statistically significant improvements ($p < 1.1 \times 10^{-8}$) over traditional optimization methods, with particular effectiveness in medical and financial domains where overfitting risks are critical.
4. Detailed analysis of the method's computational characteristics and scalability, providing practical guidelines for implementation in real-world scenarios with resource constraints.

Although the method can construct the correct network structure, its parameters often remain suboptimal. This means the network fails to fully exploit its architecture's potential, resulting in lower performance compared to other approaches. Furthermore, the absence of efficient training mechanisms leads to increased training times, making the method less practical for applications requiring quick results. In specific application scenarios, these limitations become even more apparent. For instance, with high-dimensional data, the method struggles to identify relationships between features while computational times become prohibitive. With limited training data, the constructed networks tend to overfit, resulting in poor generalization to new data. For real-time applications, the high computational complexity makes the method impractical. Compared to other approaches like traditional neural networks with backpropagation, modern deep learning architectures, or meta-learning methods, grammatical evolution appears inferior in several aspects. It requires significantly more computational resources, consistently achieves lower performance, and presents scalability limitations. These factors restrict the method's application in production systems where stability and result predictability are crucial. The practical implications of these limitations are substantial. The method requires extensive hyperparameter tuning to produce acceptable results, while its performance can be unpredictable and vary significantly between different runs. For successful application to real-world problems, additional processing and result validation are often necessary. Despite its limitations, the method offers interesting capabilities for the automatic construction of neural network architectures. However, to become truly competitive against existing approaches, it requires the development of more sophisticated optimization algorithms to reduce local minima trapping, the integration of efficient parameter training mechanisms, and improvements in method scalability. Only by addressing these issues can grammatical evolution emerge as an attractive alternative in the field of automated neural network design.

The main contributions of the proposed work can be summarized as follows:

1. Periodic application of an optimization technique to randomly selected chromosomes with the aim of improving the performance of the selected neural network but also of faster finding the global minimum of the error function.
2. The training of the artificial neural network by the optimization method is done in such a way as not to destroy the architecture of the neural network that Grammatical Evolution has already constructed.
3. The training of the artificial neural network from the optimization function is carried out using a modified fitness function, where an attempt is made to adapt the network parameters without losing its generalization properties.

The remaining of this article is organized as follows: in section 2 the proposed method and the accompanied genetic algorithm are introduced, in section 3 the experimental datasets and the series of experiments conducted are listed and discussed thoroughly followed by the section 4, where a discussion on the experimental results is provided and final in section 5 some conclusions are discussed.

2. Method description

This section provides a details description of the original neural network construction method and continuous with the proposed genetic algorithm and concludes with the overall algorithm.

2.1. The neural construction method

The neural construction method utilizes the technique of Grammatical Evolution to produce artificial neural networks. Grammatical Evolution is an evolutionary process where the chromosomes are vectors of positive integers. These integers represent rules from a Backs - Naur form (BNF) grammar [51] of the target language. The method was incorporated in various cases, such as data fitting [52,53], composition of music [54], video games [55,56], energy problems [57], cryptography [58], economics [59] etc. Any BNF grammar is defined as a set $G = (N, T, S, P)$ where the letters have the following definitions:

- The set N represents the non - terminal symbols of the grammar.
- The set T contains the terminal symbols of the grammar.
- The start symbol of the grammar is denoted as S.
- The production rules of the grammar are enclosed in the set P

The Grammatical Evolution production procedure initiates from the starting symbol S and following a series of steps, the method creates valid programs by replacing non-terminal symbols with the right hand of the selected production rule. The selection scheme has as:

- **Read** the next element V from the chromosome that is being processed.
- **Select** the next production rule following the equation:Rule = V mod N_R. The symbol N_R represents the total number of production rules for the under processing non – terminal symbol.

The process of producing valid programs through the Grammatical Evolution method is depicted graphically in Figure 1.

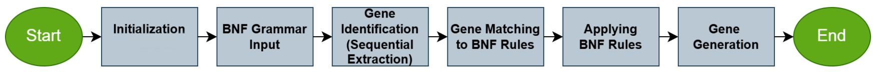


Figure 1. The Grammatical Evolution process used to produce valid programs.

The grammar used for the neural construction procedure is shown in Figure 2. The numbers shown in parentheses are the increasing numbers of the production rules for each non - terminal symbol. The constant d denotes the number of features in every pattern of the input dataset.

```
S:=<Expression>                           (0)
<Expression>::=<Neuron>                    (0)
            | <Neuron> + <Expression>      (1)
<Neuron>::=<Number>*sig(<Sum>+<Number>) (0)
<Sum>::= <Number>*<Xlist>              (0)
            |    <Sum>+<Sum>            (1)
<Xlist>::= x1        (0)
              |    x2  (1)
              ..............
              |    xd  (d-1)
<Number>::= (<Dlist>.<Dlist>)          (0)
              |    (-<Dlist>.<Dlist>) (1)
<Dlist>::= <Digit>              (0)
              | <Digit><Dlist> (1)
<Digit>::= 0       (0)
              |  1 (1)
              ...........
              |  9 (9)
```

Figure 2. The proposed grammar for the construction of artificial neural networks through Grammatical Evolution.

The used grammar produces artificial neural networks with the following form:

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

The term H stands for the number of processing units (weights) of the neural network. The function $\sigma(x)$ represents the sigmoid function. The total number of parameters for this network are computed through the following equation:

$$n = (d+2)H \tag{3}$$

For example the following form:

$$N(x) = 1.9\text{sig}(10.5x_1 + 3.2x_3 + 1.4) + 2.1\text{sig}(2.2x_2 - 3.3x_3 + 3.2) \tag{4}$$

denotes a produced neural network for a problem with 3 inputs (x_1, x_2, x_3) and the number of processing nodes is $H = 2$. The neural network produced can be shown graphically in Figure 3.

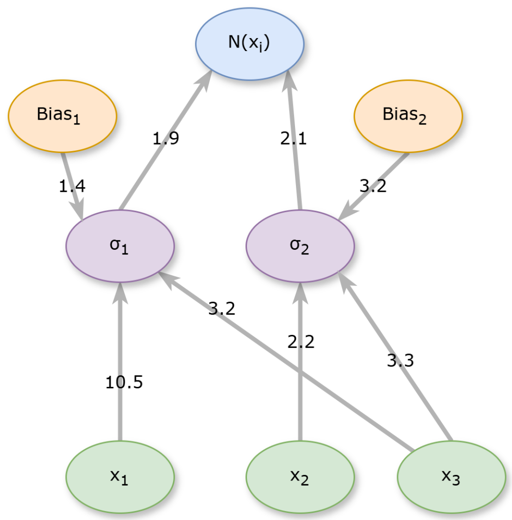


Figure 3. An example of a produced neural network.

2.2. The used genetic algorithm

In the original method of constructing artificial neural networks, it is proposed in this work to introduce the concept of local search, through the periodic application of a genetic algorithm which should maintain the structure of the neural network constructed by the original method. Additionally, a second goal of this genetic algorithm should be to avoid the problem of overfitting that could arise from simply applying a local optimization method to the previous artificial neural network. For the first goal of the modified genetic algorithm consider the example neural network shown before:

$$N(x) = 1.9\text{sig}(10.5x_1 + 3.2x_3 + 1.4) + 2.1\text{sig}(2.2x_2 - 3.3x_3 + 3.2) \quad (5)$$

The weight vector $\overrightarrow{w}$ for this neural network would be

$$\overrightarrow{w} = [1.9, 10.5, 0.0, 3.2, 1.4, 2.1, 0.0, 2.2, -3.3, 3.2] \quad (6)$$

In order to protect the structure of this artificial neural network, the modified genetic algorithm should allow changes in the parameters of this network within a value interval, which can be considered to be the pair of vectors $\left[\overrightarrow{L}, \overrightarrow{R}\right]$. The elements for the vector $\overrightarrow{L}$ are defined as

$$L_i = -F \times |w_i|,\ i = 1, \ldots, n \quad (7)$$

where F is positive number with $F > 1$. Likewise the right bound for the parameters $\overrightarrow{R}$ is defined from the following equation:

$$R_i = F \times |w_i|, i = 1, \ldots, n \quad (8)$$

For the example weight vector of equation 6 and for $F = 2$ the following vectors are used:

$$\begin{array}{lcl} L & = & [-3.8, -21.0, 0.0, -6.4, -2.8, -4.2, 0.0, -4.4, -6.6, -6.4] \\ R & = & [\ \ 3.8,\ \ 21.0, 0.0,\ \ 6.4,\ \ 2.8,\ \ 4.2, 0.0,\ \ 4.4,\ \ 6.6,\ \ 6.4] \end{array}$$

The modified genetic algorithm should also prevent the artificial neural networks it trains from the phenomenon of overfitting, which would lead to poor results on the test dataset. For this reason a quantity derived from the publication of Anastasopoulos et al. [60] is utilized here. The sigmoid function, that is used as the activation function of neural networks is defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (9)$$

A plot for this function is shown in Figure 4.

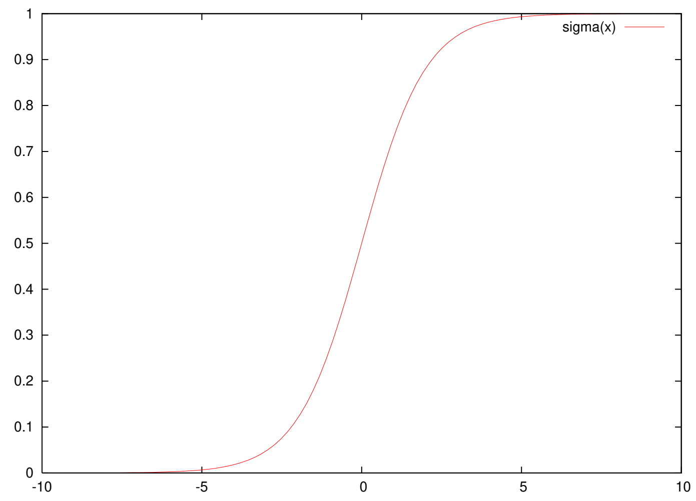


Figure 4. Plot of the sigmoid function $\sigma(x)$.

As is clear from the equation and the figure, as the value of the parameter x increases, the function tends very quickly to 1. On the other hand, the function will take values very close to 0 as the parameter x decreases. This means that the function very quickly loses the generalizing abilities it has and therefore large changes in the value of the parameter x will not cause proportional variations in the value of the sigmoid function. Therefore, the quantity $B\left(N\left(\overrightarrow{x},\overrightarrow{w}\right),a\right)$ was introduced in that paper to measure this effect. This quantity is calculated through the process of Algorithm 1. This function may be used to avoid overfitting, by limiting the parameters of the neural network to intervals that depend on the objective problem. The user defined parameter a is used here as a limit for the input value of the sigmoid unit. If this value exceeds a, then probably the neural network has a reduced generalization ability, since the sigmoid output will be the same regardless of any change in the input value.

Algorithm 1 The algorithm used to calculate the bounding quantity for neural network $N(x,w)$.

```
function evalB$\left(N\left(\overrightarrow{x},\overrightarrow{w}\right),a\right)$
1.  Inputs: The Neural network $N\left(\overrightarrow{x},\overrightarrow{w}\right)$ and the double precision value $a$, $a>1$.
2.  Set $s=0$
3.  For $i=1..H$ Do
    (a)  For $j=1..M$ Do
         i.   Calculate $v=\sum_{kT=1}^{d} w_{(d+2)i-(d+i)+k}x_{jk}+w_{(d+2)i}$
         ii.  If $|v|>a$ set $s=s+1$
    (b)  EndFor
4.  EndFor
5.  Return $\frac{s}{H\star M}$
End Function
```

The overall proposed modified genetic algorithm is shown in Algorithm 2.

Algorithm 2 The modified Genetic Algorithm.

```
Function mGA$\left(\overrightarrow{L}, \overrightarrow{R}, a, \lambda\right)$
1. Inputs: The bound vectors $\overrightarrow{L}, \overrightarrow{R}$ and the bounding factor $a$ and $\lambda$ a positive value with $\lambda > 1$.
2. Set as $N_K$the number of allowed generations and as $N_G$ the number of used chromosomes.
3. Set as $p_S$ the selection rate and as $p_M$ the mutation rate.
4. Initialize $N_G$ chromosomes inside the bounding boxes $\overrightarrow{L}, \overrightarrow{R}$.
5. Set $k = 0$, the generation number.
6. For $i = 1, \ldots, N_G$
    (a) Obtain the corresponding neural network $N_i\left(\overrightarrow{x}, \overrightarrow{g_i}\right)$ for the chromosome $g_i$.
    (b) Set $e_i = \sum_{j=1}^{M}\left(N_i\left(\overrightarrow{x}_j, \overrightarrow{w_i}\right) - y_j\right)^2$
    (c) Set $B_i = \text{eval}\left(N_i\left(\overrightarrow{x}, \overrightarrow{g_i}\right), a\right)$ using the algorithm 1.
    (d) Set $f_i = e_i \times \left(1 + \lambda B_i^2\right)$ as the fitness value of chromosome $g_i$
7. End For
8. Select the best $(1 - p_s) \times N_G$ chromosomes, that will be copied intact to the next generation. The remaining will be substituted by individuals produced by crossover and mutation.
9. Set $k = k + 1$
10. If $k \le N_K$ goto step 6.
End function
```

2.3. The overall algorithm

The overall algorithm uses the procedures presented previously to achieve greater accuracy in calculations as well as to avoid overfitting phenomena. The steps of the overall algorithm have as follows:

1. **Initialization**.
    (a) **Set** as N_C the number of chromosomes for the Grammatical Evolution procedure and as N_G the maximum number of allowed generations.
    (b) **Set** as p_S the selection rate and as p_M the mutation rate.
    (c) **Let** N_I be the number of chromosomes to which the modified genetic algorithm will be periodically applied.
    (d) **Let** N_T be the number of generations that will pass before applying the modified genetic algorithm to randomly selected chromosomes.
    (e) **Set** the weight factor F with $F > 1$.
    (f) **Set** the values $N_K,\ a,\ \lambda$ used in the modified genetic algorithm.
    (g) **Initialize** randomly the N_C chromosomes as sets of randomly selected integers.
    (h) **Set** the generation number $k = 0$
2. **Fitness Calculation**.
    (a) **For** $i = 1, \ldots, N_C$ **do**
        i. **Obtain** the chromosome g_i
        ii. **Create** the corresponding neural network $N_i\left(\overrightarrow{x}, \overrightarrow{w}\right)$ using Grammatical Evolution.
        iii. **Set** the fitness value $f_i = \sum_{j=1}^{M}\left(N_i\left(\overrightarrow{x}_j, \overrightarrow{w}\right) - y_j\right)^2$
    (b) **End For**
3. **Genetic Operations**.
    (a) **Select** the best $(1 - p_s) \times N_G$ chromosomes, that will be copied intact to the next generation.
    (b) **Create** $p_S N$ chromosomes using one - point crossover. For every couple (c_1, c_2) of produced offsprings two distinct chromosomes are selected from the cur-

rent population using tournament selection. An example of the one - point crossover procedure is shown graphically in Figure 5.
   (c) For every chromosome and for each element select a random number $r \leq 1$. Alter the current element when $r \leq p_M$

4. **Local search.**
   (a) **If** k mod $N_T = 0$ **then**
      i. **Set** $S = \left\{ g_{r_1}, g_{r_2}, \ldots, g_{r_{N_I}} \right\}$ a group of N_I randomly selected chromosomes from the genetic population.
      ii. **For** every member $g \in S$ **do**
         A. **Obtain** the corresponding neural network $N_g\left(\overrightarrow{x}, \overrightarrow{w}\right)$ for the chromosome g.
         B. **Create** the left bound vector $\overrightarrow{L_g}$ and the right bound vector $\overrightarrow{R_g}$ for g using the equations 7,8 respectively.
         C. **Set** $g = \text{mga}\left(\overrightarrow{L_g}, \overrightarrow{R_g}, a, \lambda\right)$ using the steps of algorithm 2.
      iii. **End For**
   (b) **Endif**
5. **Termination Check**.
   (a) **Set** $k = k + 1$
   (b) **If** $k \leq N_G$ goto **Fitness Calculation**.
6. **Application to the test set**.
   (a) **Obtain** the chromosome g^* with the lowest fitness value and create through Grammatical Evolution the corresponding neural network $N^*\left(\overrightarrow{x}, \overrightarrow{w}\right)$
   (b) **Apply** the neural network $N^*\left(\overrightarrow{x}, \overrightarrow{w}\right)$ and report the corresponding error value.

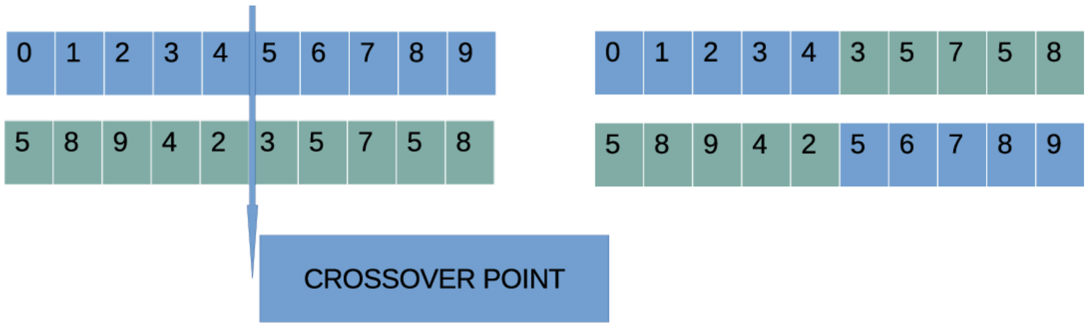


Figure 5. An example of the one - point crossover procedure.

Also the main steps of the overall algorithm are graphically illustrated in Figure 6.

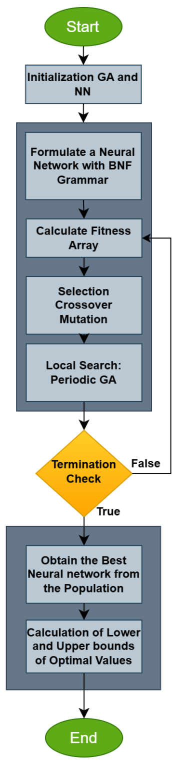

Start
Initialization GA and NN
Formulate a Neural Network with BNF Grammar
Calculate Fitness Array
Selection Crossover Mutation
Local Search: Periodic GA
Termination Check
False
True
Obtain the Best Neural network from the Population
Calculation of Lower and Upper bounds of Optimal Values
End

Figure 6. The flowchart of the overall algorithm.

3. Experimental results

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

1. The UCI database, https://archive.ics.uci.edu/(accessed on 22 January 2025)[61]
2. The Keel website, https://sci2s.ugr.es/keel/datasets.php(accessed on 22 January 2025)[62].
3. The Statlib URL https://lib.stat.cmu.edu/datasets/index(accessed on 22 January 2025).

3.1. Experimental datasets

The following datasets were utilized in the conducted experiments:

1. **Appendictis** which is a medical dataset [63].
2. **Alcohol**, which is dataset regarding alcohol consumption [64].
3. **Australian**, which is a dataset produced from various bank transactions [65].
4. **Balance** dataset [66], produced from various psychological experiments.
5. **Cleveland**, a medical dataset which was discussed in a series of papers [67,68].
6. **Circular** dataset, which is an artificial dataset.
7. **Dermatology**, a medical dataset for dermatology problems [69].
8. **Ecoli**, which is related to protein problems [70].
9. **Glass** dataset, that contains measurements from glass component analysis.
10. **Haberman**, a medical dataset related to breast cancer.
11. **Hayes-roth** dataset [71].
12. **Heart**, which is a dataset related to heart diseases [72].
13. **HeartAttack**, which is a medical dataset for the detection of heart diseases
14. **Housevotes**, a dataset which is related to the Congressional voting in USA [73].
15. **Ionosphere**, a dataset that contains measurements from the ionosphere [74,75].
16. **Liverdisorder**, a medical dataset that was studied thoroughly in a series of papers[76, 77].

17. **Lymography** [78].
18. **Mammographic**, which is a medical dataset used for the prediction of breast cancer [79].
19. **Parkinsons**, which is a medical dataset used for the detection of Parkinson's disease [80,81].
20. **Pima**, which is a medical dataset for the detection of diabetes[82].
21. **Phoneme**, a dataset that contains sound measurements.
22. **Popfailures**, a dataset related to experiments regarding climate [83].
23. **Regions2**, a medical dataset applied to liver problems [84].
24. **Saheart**, which is a medical dataset concerning heart diseases[85].
25. **Segment** dataset [86].
26. **Statheart**, a medical dataset related to heart diseases.
27. **Spiral**, an artificial dataset with two classes.
28. **Student**, which is a dataset regarding experiments in schools [87].
29. **Transfusion**, which is a medical dataset [88].
30. **Wdbc**, which is a medical dataset regarding breast cancer [89,90].
31. **Wine**, a dataset regarding measurements about the quality of wines [91,92].
32. **EEG**, which is dataset regardingEEG recordings [93,94]. From this dataset the following cases were used: Z_F_S, ZO_NF_S, ZONF_S and Z_O_N_F_S.
33. **Zoo**, which is a dataset regarding animal classification [95] .

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

1. **Abalone**, which is a dataset about the age of abalones [96].
2. **Airfoil**, a dataset founded in NASA [97].
3. **Auto**, a dataset related to the consumption of fuels from cars.
4. **BK**, which is used to predict the points scored in basketball games.
5. **BL**, a dataset that contains measurements from electricity experiments.
6. **Baseball**, which is a dataset used to predict the income of baseball players.
7. **Concrete**, which is a civil engineering dataset [98].
8. **DEE**, a dataset that is used to predict the price of electricity.
9. **Friedman**, which is an artificial dataset[99].
10. **FY,** which is a dataset regarding the longevity of fruit flies.
11. **HO**, a dataset located in the STATLIB repository.
12. **Housing**, regarding the price of houses [100].
13. **Laser**, which contains measurements from various physics experiments.
14. **LW**, a dataset regarding the weight of babes.
15. **Mortgage**, a dataset that contains measurements from the economy of USA.
16. **PL** dataset, located in the STALIB repository.
17. **Plastic**, a dataset regarding problems occurred with the pressure on plastics.
18. **Quake**, a dataset regarding the measurements of earthquakes.
19. **SN**, a dataset related to trellising and pruning.
20. **Stock**, which is a dataset regarding stocks.
21. **Treasury**, a dataset that contains measurements from the economy of USA.

3.2. Experiments

The software used in the experiment was coded in C++ with the assistance of the freely available Optimus environment [101]. Every experiments was conducted 30 times and each time different seed for the random generator was used. The experiments were validated using the ten - fold cross validation technique. The average classification error, as measured in the corresponding test set was reported for the classification datasets. This error is calculated through the following formula:

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

Here the test set T is a set $T = (x_i, y_i),\ i = 1, \ldots, N$. Likewise, the average regression error is reported for the regression datasets. This error can be obtained using the following equation:

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

The experiments were executed on an AMD Ryzen 5950X with 128GB of RAM and the used operating system was Debian Linux. The values for the parameters of the proposed method are shown in Table 1.

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

PARAMETER	MEANING	VALUE
N_C	Chromosomes	500
N_G	Maximum number of generations	500
N_K	Number of generations for the modified Genetic Algorithm	50
p_S	Selection rate	0.1
p_M	Mutation rate	0.05
N_T	Generations before local search	20
N_I	Chromosomes participating in local search	20
a	Bounding factor	10.0
F	Scale factor for the margins	2.0
λ	Value used for penalties	100.0

The parameter values have been chosen in such a way that there is a balance between the speed of the proposed method and its efficiency. In the following tables that describe the experimental results the following notation is used:

1. The column DATASET represents the used dataset.
2. The column ADAM represents the incorporation of the ADAM optimization method [12] to train a neural network with $H = 10$ processing nodes.
3. The column BFGS stands for the usage of a BFGS variant of Powell [102] to train an artificial neural network with $H = 10$ processing nodes.
4. The column GENETIC represents the incorporation of a Genetic Algorithm with the same parameter set as provided in Table 1 to train a neural network with $H = 10$ processing nodes.
5. The column RBF describes the experimental results obtained by the application of a Radial Basis Function (RBF) network [103,104] with $H = 10$ hidden nodes.
6. The column NNC stands for the usage of the original neural construction method.
7. The column NEAT represents the usage of the NEAT method (NeuroEvolution of Augmenting Topologies) [105].
8. The column PRUNE stands for the the usage of OBS pruning method [106], as coded in Fast Compressed Neural Networks library [107].
9. The column DNN represents the application of the deep neural network provided in the Tiny Dnn library, which is available from https://github.com/tiny-dnn/tiny-dnn(accessed on 7 September 2025). The network was trained using the AdaGrad optimizer [108].
10. The column PROPOSED denotes the usage of the proposed method.
11. The row AVERAGE represents the average classification or regression error for all datasets in the corresponding table.

Based on the table 2 with 36 classification datasets and nine methods (lower percentage implies lower error), the PROPOSED method attains the lowest mean error (21.18%) and the best average rank across datasets (1.83 on a 1–9 scale). The current work achieves the best result in 18 out of 36 datasets. The remaining per-dataset best results are distributed as follows: RBF 4, PRUNE 4, DNN 3, NEAT 3, NNC 3, and GENETIC 1, while ADAM and BFGS record no first-place finishes. In head-to-head, dataset-wise comparisons, the proposed method outperforms each alternative on the vast majority of datasets: versus ADAM on 35/36 datasets, versus BFGS on 33/36, versus GENETIC on 32/36, versus RBF on 30/36, versus NEAT on 33/36, versus PRUNE on 31/36, versus DNN on 32/36, and versus NNC on 32/36. The average absolute error reduction of PROPOSED relative to each competitor, computed as "competitor − PROPOSED" in percentage points and then averaged over the 36 datasets, is 14.39 (ADAM), 13.51 (BFGS), 6.12 (GENETIC), 8.60 (RBF), 11.90 (NEAT), 7.53 (PRUNE), 6.64 (DNN), and 2.60 (NNC). The corresponding mean relative reductions, computed as (competitor − PROPOSED)/competitor and then averaged per dataset, are 39.24%, 34.29%, 22.83%, 26.50%, 37.09%, 20.91%, 22.95%, and 12.65%. The few datasets where the proposed method is worse are concentrated in specific cases: ADAM only on POPFAILURES; BFGS on CIRCULAR, PHONEME, and POPFAILURES; GENETIC on CIRCULAR, GLASS, LIVERDISORDER, and PHONEME; RBF on APPENDICITIS, CIRCULAR, GLASS, HABERMAN, LIVERDISORDER, and SPIRAL; NEAT on ECOLI, HABERMAN, and LIVERDISORDER; PRUNE on ALCOHOL, DERMATOLOGY, IONOSPHERE, LYMOGRAPHY, and POPFAILURES; DNN on HABERMAN, HOUSEVOTES, SEGMENT, and ZONF_S; and NNC on AUSTRALIAN, HEARTATTACK, HOUSEVOTES, and IONOSPHERE. The table's AVERAGE row is consistent with this picture: PROPOSED has the lowest mean error (21.18%) among all methods. Relative to the second-best mean in that row (NNC: 24.79%), PROPOSED reduces error by 3.61 percentage points, corresponding to about a 14.56% relative reduction. Median errors tell the same story: PROPOSED has a median of 19.24%, lower than NNC (21.21%) and DNN (25.83%). Overall, PROPOSED shows consistent superiority in terms of error, as reflected by the mean, the across-dataset ranking, and the dataset-wise head-to-head counts, with only a small number of dataset-specific exceptions.

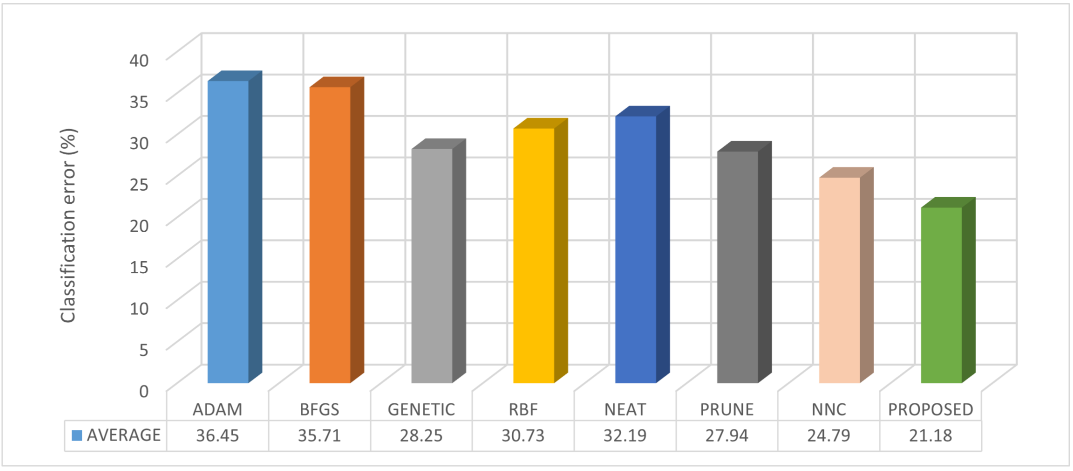
Table 2. Experimental results using a variety of machine learning methods for the classification datasets.

DATASET	ADAM	BFGS	GENETIC	RBF	NEAT	PRUNE	DNN	NNC	PROPOSED
APPENDICITIS	16.50%	18.00%	24.40%	12.23%	17.20%	15.97%	17.30%	14.40%	14.30%
ALCOHOL	57.78%	41.50%	39.57%	49.32%	66.80%	15.75%	39.04%	37.72%	35.60%
AUSTRALIAN	35.65%	38.13%	32.21%	34.89%	31.98%	43.66%	35.03%	14.46%	14.55%
BALANCE	12.27%	8.64%	8.97%	33.53%	23.14%	9.00%	24.56%	23.65%	7.84%
CLEVELAND	67.55%	77.55%	51.60%	67.10%	53.44%	51.48%	63.28%	50.93%	46.41%
CIRCULAR	19.95%	6.08%	5.99%	5.98%	35.18%	12.76%	21.87%	12.66%	6.92%
DERMATOLOGY	26.14%	52.92%	30.58%	62.34%	32.43%	9.02%	24.26%	21.54%	20.54%
ECOLI	64.43%	69.52%	54.67%	59.48%	43.44%	60.32%	60.79%	49.88%	48.82%
GLASS	61.38%	54.67%	52.86%	50.46%	55.71%	66.19%	56.05%	56.09%	53.52%
HABERMAN	29.00%	29.34%	28.66%	25.10%	24.04%	29.38%	25.73%	27.53%	26.80%
HAYES-ROTH	59.70%	37.33%	56.18%	64.36%	50.15%	45.44%	44.65%	33.69%	31.00%
HEART	38.53%	39.44%	28.34%	31.20%	39.27%	27.21%	30.67%	15.67%	15.45%
HEARTATTACK	45.55%	46.67%	29.03%	29.00%	32.34%	29.26%	32.97%	20.87%	21.77%
HOUSEVOTES	7.48%	7.13%	6.62%	6.13%	10.89%	5.81%	3.13%	3.17%	3.78%
IONOSPHERE	16.64%	15.29%	15.14%	16.22%	19.67%	11.32%	12.57%	11.29%	11.94%
LIVERDISORDER	41.53%	42.59%	31.11%	30.84%	30.67%	49.72%	32.21%	32.35%	31.32%
LYMOGRAPHY	39.79%	35.43%	28.42%	25.50%	33.70%	22.02%	24.07%	25.29%	23.72%
MAMMOGRAPHIC	46.25%	17.24%	19.88%	21.38%	22.85%	38.10%	19.83%	17.62%	16.74%
PARKINSONS	24.06%	27.58%	18.05%	17.41%	18.56%	22.12%	21.32%	12.74%	12.63%
PHONEME	29.43%	15.58%	15.55%	23.32%	22.34%	29.35%	22.68%	22.50%	21.52%
PIMA	34.85%	35.59%	32.19%	25.78%	34.51%	35.08%	32.63%	28.07%	23.34%
POPFAILURES	5.18%	5.24%	5.94%	7.04%	7.05%	4.79%	6.83%	6.98%	5.72%
REGIONS2	29.85%	36.28%	29.39%	38.29%	33.23%	34.26%	33.42%	26.18%	23.81%
SAHEART	34.04%	37.48%	34.86%	32.19%	34.51%	37.70%	35.11%	29.80%	28.04%
SEGMENT	49.75%	68.97%	57.72%	59.68%	66.72%	60.40%	32.04%	53.50%	48.20%
SPIRAL	47.67%	47.99%	48.66%	44.87%	48.66%	50.38%	45.64%	48.01%	44.95%
STATHEART	44.04%	39.65%	27.25%	31.36%	44.36%	28.37%	30.22%	18.08%	17.93%
STUDENT	5.13%	7.14%	5.61%	5.49%	10.20%	10.84%	6.93%	6.70%	4.05%
TRANSFUSION	25.68%	25.84%	24.87%	26.41%	24.87%	29.35%	25.92%	25.77%	23.16%
WDBC	35.35%	29.91%	8.56%	7.27%	12.88%	15.48%	9.43%	7.36%	4.95%
WINE	29.40%	59.71%	19.20%	31.41%	25.43%	16.62%	27.18%	13.59%	9.94%
Z_F_S	47.81%	39.37%	10.73%	13.16%	38.41%	17.91%	9.27%	14.53%	7.97%
Z_O_N_F_S	78.79%	65.67%	64.81%	48.70%	77.08%	71.29%	67.80%	48.62%	39.28%
ZO_NF_S	47.43%	43.04%	21.54%	9.02%	43.75%	15.57%	8.50%	13.54%	6.94%
ZONF_S	11.99%	15.62%	4.36%	4.03%	5.44%	3.27%	2.52%	2.64%	2.60%
ZOO	14.13%	10.70%	9.50%	21.93%	20.27%	8.53%	16.20%	8.70%	6.60%
AVERAGE	**36.45%**	**35.71%**	**28.25%**	**30.73%**	**32.19%**	**27.94%**	**27.82%**	**24.79%**	**21.18%**

Additionally, the average classification error for all methods is illustrated in Figure 7. Additionally, the dimension for each classification dataset and the number of distinct classes are provided in Table 3.

Table 3. Dimension and number of classes for each classification dataset.

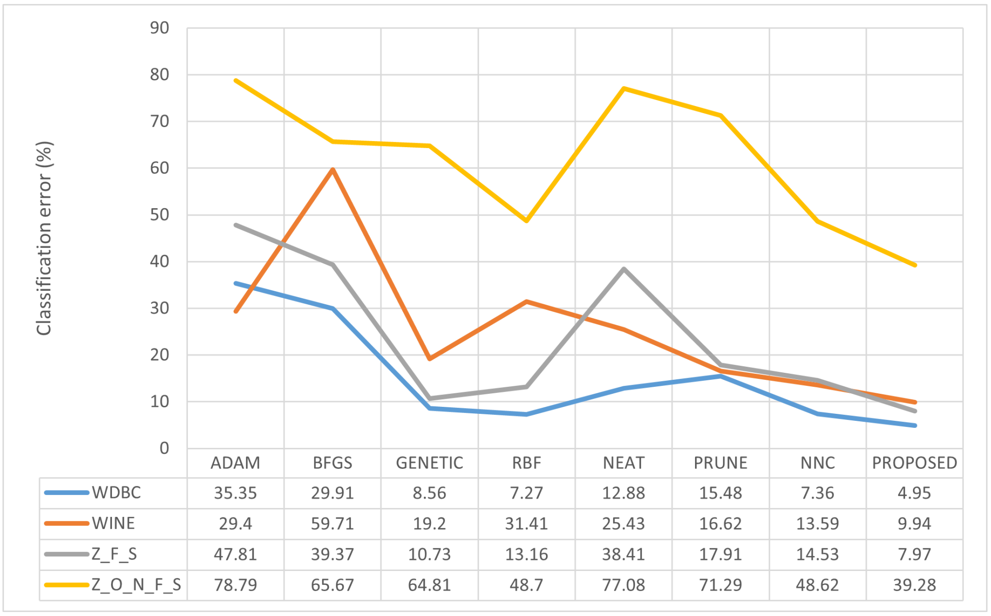
DATASET	FEATURES	CLASSES
APPENDICITIS	7	2
ALCOHOL	154	4
AUSTRALIAN	14	2
BALANCE	4	3
CLEVELAND	13	5
CIRCULAR	5	2
DERMATOLOGY	34	6
ECOLI	7	8
GLASS	9	6
HABERMAN	3	2
HAYES-ROTH	5	3
HEART	13	2
HEARTATTACK	13	2
HOUSEVOTES	16	2
IONOSPHERE	34	2
LIVERDISORDER	6	2
LYMOGRAPHY	18	4
MAMMOGRAPHIC	5	2
PARKINSONS	22	2
PHONEME	5	2
PIMA	8	2
POPFAILURES	18	2
REGIONS2	18	5
SAHEART	9	2
SEGMENT	19	7
SPIRAL	2	2
STATHEART	13	2
STUDENT	5	4
TRANSFUSION	4	2
WDBC	30	2
WINE	13	3
Z_F_S	21	3
Z_O_N_F_S	21	5
ZO_NF_S	21	3
ZONF_S	21	2
ZOO	16	7



	ADAM	BFGS	GENETIC	RBF	NEAT	PRUNE	NNC	PROPOSED
AVERAGE	36.45	35.71	28.25	30.73	32.19	27.94	24.79	21.18

Figure 7. The average classification error for all used datasets.

Also, a line plot is provided in Figure 8for a series of selected datasets to depict the effectiveness of the proposed method.



	ADAM	BFGS	GENETIC	RBF	NEAT	PRUNE	NNC	PROPOSED
WDBC	35.35	29.91	8.56	7.27	12.88	15.48	7.36	4.95
WINE	29.4	59.71	19.2	31.41	25.43	16.62	13.59	9.94
Z_F_S	47.81	39.37	10.73	13.16	38.41	17.91	14.53	7.97
Z_O_N_F_S	78.79	65.67	64.81	48.7	77.08	71.29	48.62	39.28

Figure 8. Line plot for a series of classification datasets.

In table 4 with 21 regression datasets and nine methods (lower values indicate lower error), the proposed method achieves the lowest mean error (4.28 versus 6.29 for NNC in the AVERAGE row) and the lowest median (0.036). Its average rank across datasets is 1.67 on a 1–9 scale, lower than any competitor (next best: NNC 3.83, DNN 4.69, RBF 4.79). At the dataset level, the current work is the strict winner on 12 of 21 datasets and ties for best on 2 more (AIRFOIL with PRUNE and LW with NNC), thus topping 14/21 datasets overall. The remaining first places are taken by ADAM (ABALONE and FY), GENETIC (FRIEDMAN and STOCK), RBF (BK and DEE), and NNC (HO). The gap from the dataset-wise best alternative is typically small: PROPOSED is within 5% of the best on 15/21 datasets and within 10% on 16/21, with the largest shortfalls appearing mainly on FRIEDMAN (5.34 versus 1.249) and, more mildly, on DEE (0.22 versus 0.17) and STOCK (4.69 versus 3.88). In head-to-head, dataset-wise comparisons, PROPOSED has lower error than each alternative in the vast majority of cases: versus ADAM it is better on 19/21 datasets (worse on ABALONE and FY), versus BFGS on 20/21 (worse only on FRIEDMAN), versus GENETIC on 19/21 (worse on FRIEDMAN and STOCK), versus RBF on 18/21 (worse on BK, DEE, and FY), versus NEAT on 21/21, versus PRUNE on 19/21 with 1 tie (worse on FY), versus DNN on 18/21 (worse on BK, FRIEDMAN, and FY), and versus NNC on 19/21 with 1 tie (worse on HO, tie on LW). The average absolute error reduction of PROPOSED relative to each competitor, computed as "competitor − PROPOSED" and then averaged over the 21 datasets, is approximately 18.18 (ADAM), 26.01 (BFGS), 5.03 (GENETIC), 5.74 (RBF), 10.37 (NEAT), 11.12 (PRUNE), 7.00 (DNN), and 2.01 (NNC); the corresponding mean relative reductions are about 62%, 64%, 44%, 53%, 80%, 50%, 44%, and 46%. The AVERAGE row is consistent with this picture: PROPOSED has the lowest mean error (4.28), outperforming the second-best NNC by 2.01 units (about a 32% relative reduction when computed from the reported means) and leaving larger margins against DNN, RBF, GENETIC, PRUNE, NEAT, ADAM, and BFGS. Overall, PROPOSED exhibits consistent superiority in terms of error as reflected by mean and median values, average rank, and the per-dataset win counts, with the most notable exceptions confined to a few datasets that appear to have distinct error scales.

Table 4. Experimental results using a variety of machine learning methods on the regression datasets.

DATASET	ADAM	BFGS	GENETIC	RBF	NEAT	PRUNE	DNN	NNC	PROPOSED
ABALONE	4.30	5.69	7.17	7.37	9.88	7.88	6.91	5.08	4.47
AIRFOIL	0.005	0.003	0.003	0.27	0.067	0.002	0.004	0.004	0.002
AUTO	70.84	60.97	12.18	17.87	56.06	75.59	13.26	17.13	9.09
BK	0.0252	0.28	0.027	0.02	0.15	0.027	0.02	0.10	0.023
BL	0.622	2.55	5.74	0.013	0.05	0.027	0.006	1.19	0.001
BASEBALL	77.90	119.63	103.60	93.02	100.39	94.50	110.22	61.57	48.13
CONCRETE	0.078	0.066	0.0099	0.011	0.081	0.0077	0.021	0.008	0.005
DEE	0.63	2.36	1.013	0.17	1.512	1.08	0.31	0.26	0.22
FRIEDMAN	22.90	1.263	1.249	7.23	19.35	8.69	2.75	6.29	5.34
FY	0.038	0.19	0.65	0.041	0.08	0.042	0.039	0.11	0.043
HO	0.035	0.62	2.78	0.03	0.169	0.03	0.026	0.015	0.016
HOUSING	80.99	97.38	43.26	57.68	56.49	52.25	65.18	25.47	15.47
LASER	0.03	0.015	0.59	0.03	0.084	0.007	0.045	0.025	0.0049
LW	0.028	2.98	1.90	0.03	0.03	0.02	0.023	0.011	0.011
MORTGAGE	9.24	8.23	2.41	1.45	14.11	12.96	9.74	0.30	0.023
PL	0.117	0.29	0.29	2.118	0.09	0.032	0.056	0.047	0.029
PLASTIC	11.71	20.32	2.791	8.62	20.77	17.33	3.82	4.20	2.17
QUAKE	0.07	0.42	0.04	0.07	0.298	0.04	0.098	0.96	0.036
SN	0.026	0.40	2.95	0.027	0.174	0.032	0.027	0.026	0.024
STOCK	180.89	302.43	3.88	12.23	12.23	39.08	12.95	8.92	4.69
TREASURY	11.16	9.91	2.93	2.02	15.52	13.76	11.41	0.43	0.068
AVERAGE	**22.46**	**30.29**	**9.31**	**10.02**	**14.65**	**15.40**	**11.28**	**6.29**	**4.28**

The statistical comparison depicted in Figure 9, indicates that all pairwise comparisons between the current method and the alternative models are highly statistically significant. Under the conventional star notation, **** denotes p<0.0001, while the ***** flag for PROPOSED versus RBF signals even stronger statistical evidence in favor of PROPOSED. Overall, the findings confirm that PROPOSED consistently achieves lower error than every comparator across the classification datasets, with a negligible likelihood that these differences are due to chance.

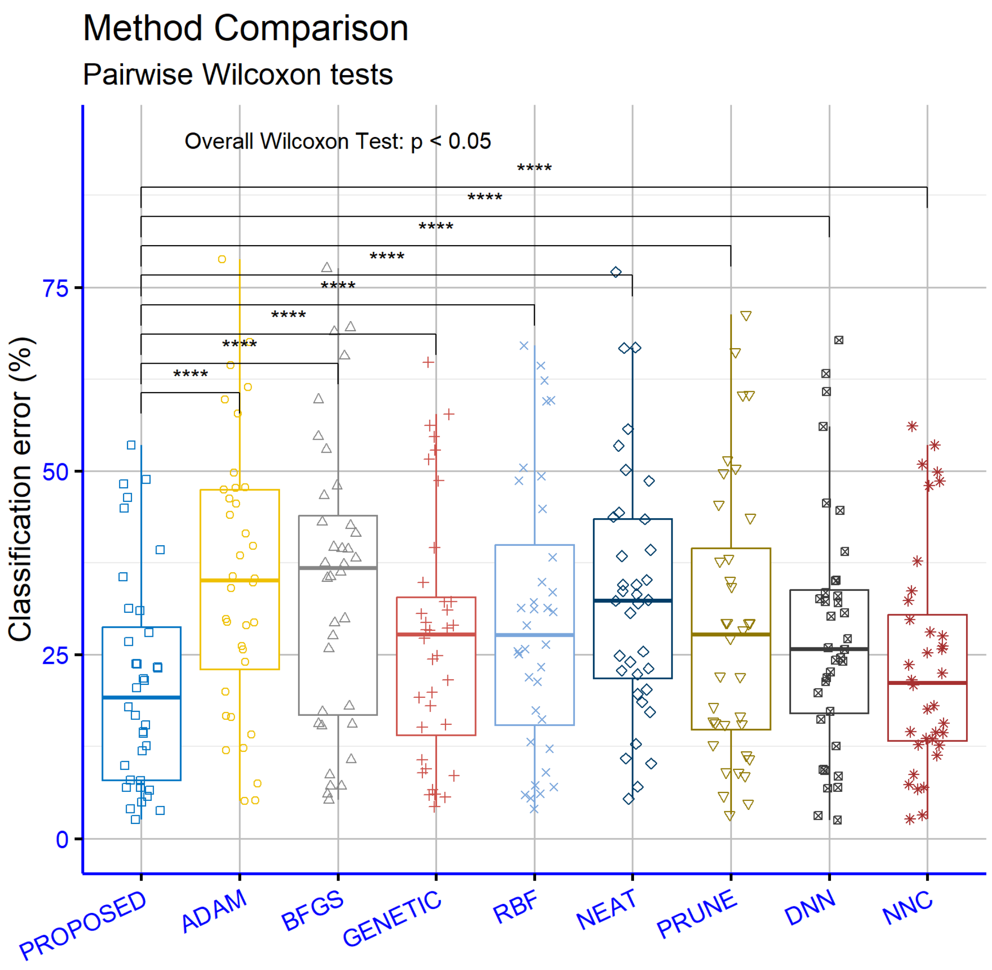


Figure 9. Statistical comparison of the machine learning models for the classification datasets.

In Figure 10, the statistical comparison on the regression datasets indicates that the proposed model differs significantly from all alternative methods. Evidence is particularly strong against ADAM, BFGS, RBF, PRUNE, and NNC (p=*, i.e., p≤0.001), with the strongest signal observed against NEAT (p=, i.e., p≤0.0001). Comparisons against GENETIC and DNN are also statistically significant but comparatively weaker (p=, i.e., p≤0.01). Taken together, the results consistently support that the proposed model attains lower error than every competing approach across the examined datasets, with a very low probability that the observed differences are due to chance. Note that significance levels reflect the strength of statistical evidence rather than effect magnitude; for a fuller interpretation it is advisable to also report absolute or relative error differences and suitable effect size metrics.

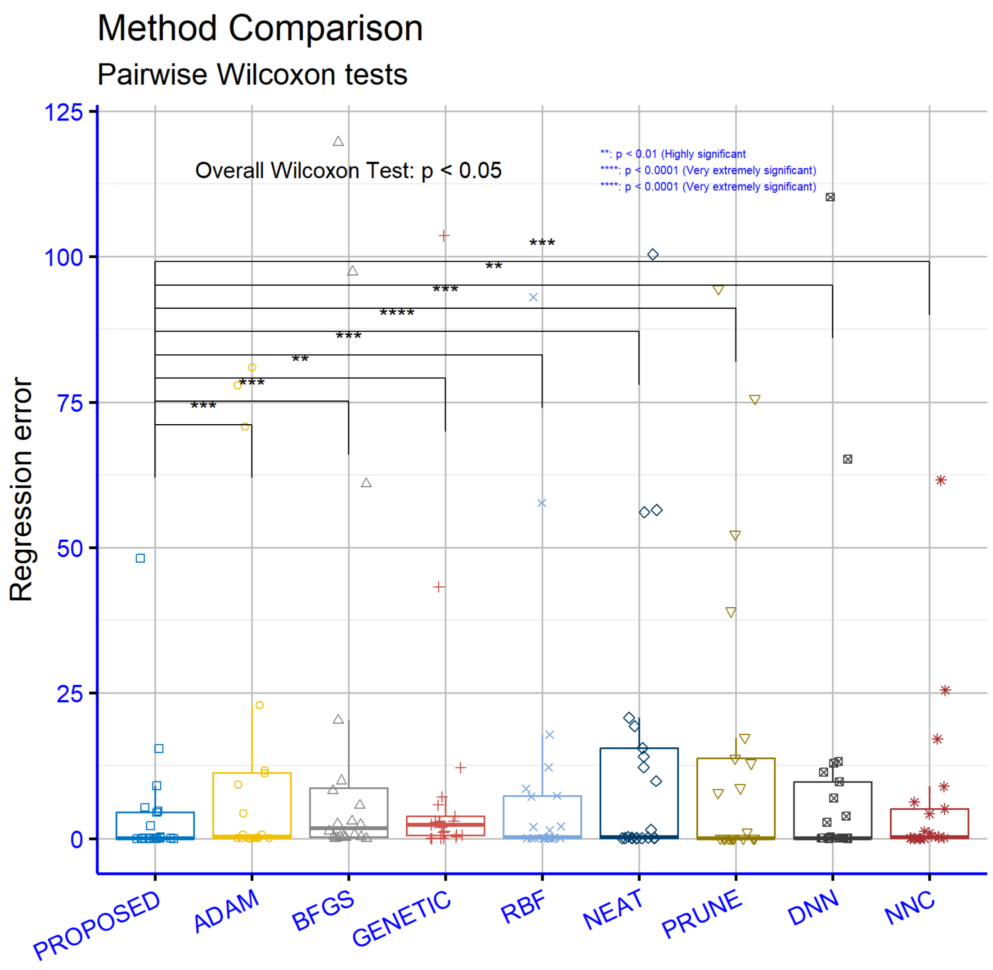


Figure 10. Statistical comparison between the used methods for the regression datasets.

3.3. Experiments with different crossover mechanism 479

Also, in order to illustrated the robustness of the proposed method, another experiment 480
was conducted were the uniform crossover procedure was used for the Neural Network 481
Construction method and the proposed one instead of the one - point crossover. The 482
experimental results for the classification datasets are shown in Table 5and for regression 483
datasets in Table 6. Although, the one - point crossover mechanism is proposed as the 484
crossover procedure in the original article of Grammatical Evolution. 485

Table 5. Experimental results for the classification datasets were a comparison is made against the original one - point crossover method and the uniform crossover procedure.

DATASET	NNC ONE-POINT	NNC -UNIFORM	PROPOSED ONE-POINT	PROPOSED UNIFORM
APPENDICITIS	14.40%	14.20%	14.30%	14.40%
ALCOHOL	37.72%	42.34%	35.60%	39.70%
AUSTRALIAN	14.46%	14.13%	14.55%	14.35%
BALANCE	23.65%	20.73%	7.84%	7.61%
CLEVELAND	50.93%	51.45%	46.41%	46.28%
CIRCULAR	12.66%	17.59%	6.92%	11.86%
DERMATOLOGY	21.54%	30.09%	20.54%	26.86%
ECOLI	49.88%	48.12%	48.82%	48.88%
GLASS	56.09%	57.43%	53.52%	52.43%
HABERMAN	27.53%	27.17%	26.80%	26.70%
HAYES-ROTH	33.69%	36.61%	31.00%	33.62%
HEART	15.67%	16.41%	15.45%	14.96%
HEARTATTACK	20.87%	21.50%	21.77%	21.27%
HOUSEVOTES	3.17%	3.44%	3.78%	3.43%
IONOSPHERE	11.29%	11.80%	11.94%	11.77%
LIVERDISORDER	32.35%	32.65%	31.32%	32.32%
LYMOGRAPHY	25.29%	28.21%	23.72%	25.14%
MAMMOGRAPHIC	17.62%	18.04%	16.74%	16.24%
PARKINSONS	12.74%	11.63%	12.63%	12.74%
PHONEME	22.50%	23.46%	21.52%	21.32%
PIMA	28.07%	27.95%	23.34%	24.43%
POPFAILURES	6.98%	6.80%	5.72%	6.01%
REGIONS2	26.18%	25.71%	23.81%	25.21%
SAHEART	29.80%	30.52%	28.04%	29.13%
SEGMENT	53.50%	54.78%	48.20%	52.26%
SPIRAL	48.01%	48.35%	44.95%	45.03%
STATHEART	18.08%	18.85%	17.93%	18.59%
STUDENT	6.70%	6.15%	4.05%	4.10%
TRANSFUSION	25.77%	25.58%	23.16%	23.96%
WDBC	7.36%	8.07%	4.95%	6.31%
WINE	13.59%	14.41%	9.94%	11.76%
Z_F_S	14.53%	18.33%	7.97%	10.13%
Z_O_N_F_S	48.62%	51.10%	39.28%	44.90%
ZO_NF_S	13.54%	14.52%	6.94%	8.24%
ZONF_S	2.64%	2.82%	2.60%	2.78%
ZOO	8.70%	10.40%	6.60%	8.70%
AVERAGE	**24.79%**	**24.76%**	**21.18%**	**22.32%**

Table 6. Experimental results for the regression datasets using two crossover methods: the one point crossover and the uniform crossover.

DATASET	NNC ONE-POINT	NNC UNIFORM	PROPOSED ONE-POINT	PROPOSED UNIFORM
ABALONE	5.08	5.40	4.47	4.55
AIRFOIL	0.004	0.004	0.002	0.003
AUTO	17.13	20.06	9.09	11.10
BK	0.10	0.018	0.023	0.018
BL	1.19	0.018	0.001	0.001
BASEBALL	61.57	63.44	48.13	49.99
CONCRETE	0.008	0.009	0.005	0.006
DEE	0.26	0.28	0.22	0.24
FRIEDMAN	6.29	6.98	5.34	5.85
FY	0.11	0.04	0.043	0.04
HO	0.015	0.016	0.016	0.011
HOUSING	25.47	26.68	15.47	16.89
LASER	0.025	0.041	0.0049	0.008
LW	0.011	0.012	0.011	0.011
MORTGAGE	0.30	0.29	0.023	0.037
PL	0.047	0.046	0.029	0.024
PLASTIC	4.20	5.20	2.17	2.30
QUAKE	0.96	0.036	0.036	0.036
SN	0.026	0.026	0.024	0.024
STOCK	8.92	10.89	4.69	8.31
TREASURY	0.43	0.38	0.068	0.072
AVERAGE	**6.29**	**6.66**	**4.28**	**4.74**

Analysis of the results in Table 5 demonstrates that the proposed method systematically outperforms both variants of NNC, regardless of the crossover type employed. Specifically, the proposed method with one-point crossover achieves a significantly lower average error rate (21.18%) compared to NNC (24.79%). A similar performance gap is observed with uniform crossover, where the proposed method maintains superiority (22.32% vs NNC's 24.76%). The advantage is particularly pronounced in several datasets: for BALANCE (7.61-7.84% vs 20.73-23.65%), CIRCULAR (6.92-11.86% vs 12.66-17.59%), Z_F_S (7.97-10.13% vs 14.53-18.33%), and ZO_NF_S (6.94-8.24% vs 13.54-14.52%). Notably, in datasets like WDBC and WINE, the proposed method reduces the error by nearly half compared to NNC. Interestingly, the choice of crossover type shows minimal impact on performance for both methods. The marginal differences between one-point and uniform crossover (with average errors remaining stable for each method) suggest that the proposed method's superiority stems from its fundamental architecture rather than the recombination technique. These experimental results confirm the robustness of the proposed method, which maintains consistent superiority across various classification datasets while significantly reducing average error rates compared to standard NNC approaches.

Table 6 further validates the proposed method's superiority for regression datasets. The proposed method achieves lower average errors with both one-point (4.28) and uniform crossover (4.74) compared to NNC (6.29 and 6.66 respectively). The performance gap is especially notable in key datasets: AUTO (9.09-11.1 vs 17.13-20.06), HOUSING (15.47-16.89 vs 25.47-26.68), and PLASTIC (2.17-2.3 vs 4.2-5.2). In several cases (AIRFOIL, LASER, MORTGAGE), the proposed method reduces errors to a fraction of NNC's values. Particularly impressive results appear in BL (0.001) and QUAKE (0.036), where the proposed method achieves remarkably low, crossover-invariant errors. The crossover type shows slightly less impact on the proposed method (average difference of 0.46) than on NNC (0.37 difference), though one-point crossover yields marginally better results for both. These comprehensive results confirm that the proposed method maintains its superiority in regression problems, delivering consistently and significantly improved performance over NNC. Its ability to achieve lower errors across diverse problems, independent of crossover selection, solidifies its position as a more reliable and effective approach.

3.4. Experiments with the critical parameter N_K

Another experiment was conducted where the parameter N_K was altered in the range $[5, \ldots, 50]$ and the results for the regression datasets are depicted in Figure 11.

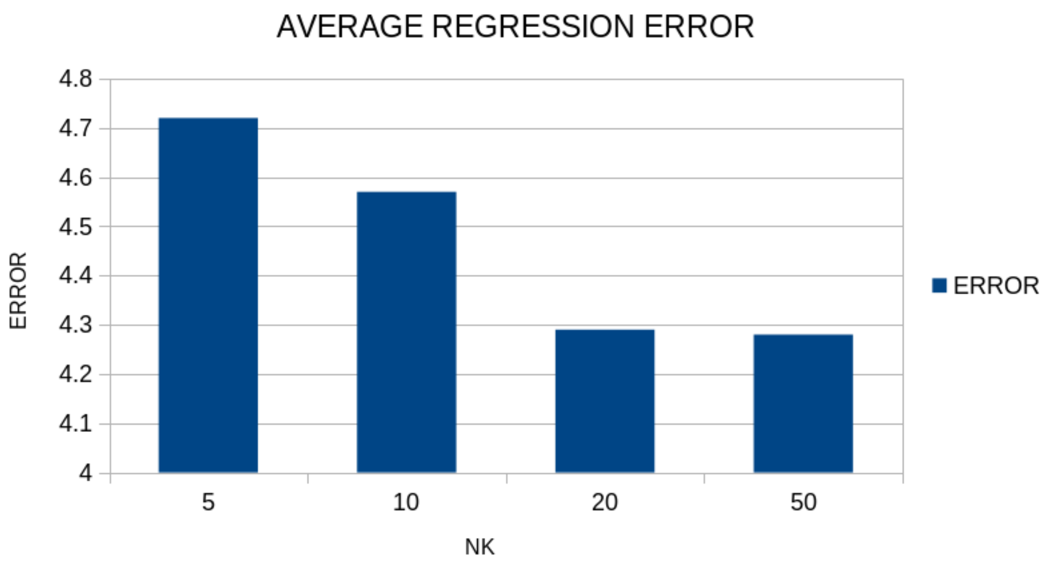


Figure 11. Average regression error for the regression datasets and the proposed method using a variety of values for the parameter N_K.

Additionally a series of experiments was conducted where the parameter N_I was changed from 10 to 40 and the results are graphically presented in Figure 12.

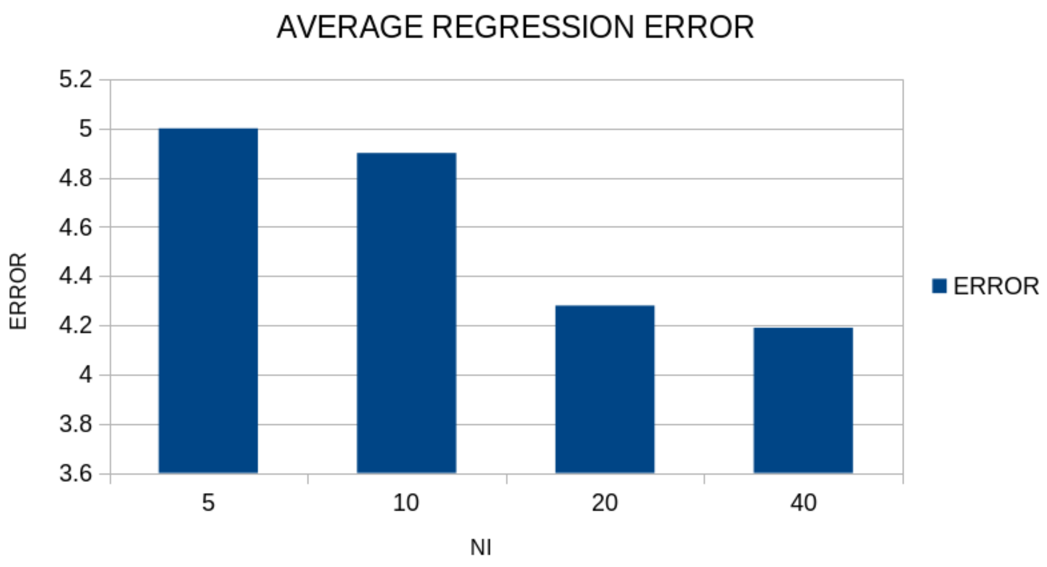


Figure 12. Experimental results for the regression datasets and the proposed method using a variety of values for the parameter N_I.

This figure presents the relationship between the number of chromosomes (N_I) participating in the secondary genetic algorithm and the resulting regression error. We observe that as the number of chromosomes increases, the error decreases, indicating improvement in the model's performance. Specifically, for $N_I = 5$, the error is 4.99, while for $N_I = 10$, the error drops to 4.89. This trend continues with further increase in chromosomes: for $N_I = 20$, the error reaches 4.27, and for $N_I = 40$, the error reaches 4.18. This error reduction shows that using more chromosomes in the secondary genetic algorithm leads to better optimization of the neural network's parameters, resulting in error minimization. However, the improvement is not linear. We observe that the difference in error between $N_I = 5$ and $N_I = 10$ is 0.10, while between $N_I = 20$ and $N_I = 40$ it is only 0.09. This may indicate that beyond a certain point, increasing chromosomes has progressively smaller impact on error reduction. This phenomenon may be due to factors such as algorithm convergence or the existence of an optimization threshold beyond which improvement becomes more difficult. Furthermore, the selection of N_I may be influenced by computational constraints. Using more chromosomes increases the computational load, so the performance improvement must be balanced against resource costs. For example, transitioning from $N_I = 20$ to $N_I = 40$ leads to error reduction of only 0.09, which may not justify the doubling of computational cost in certain scenarios. In summary, the table confirms that increasing the number of chromosomes improves the model's performance, but its effect becomes smaller as N_I grows larger. This means that the optimal selection of N_I depends on a combination of factors, such as the desired accuracy, available computational resources, and the nature of the problem.

Also, in Figure 13 the average execution time for the regression dataset is plotted. In this graph the original neural network construction method is depicted as well as the proposed one using a series of values for the parameter N_I. As is was expected, the execution time increases when the parameter N_I is increased.

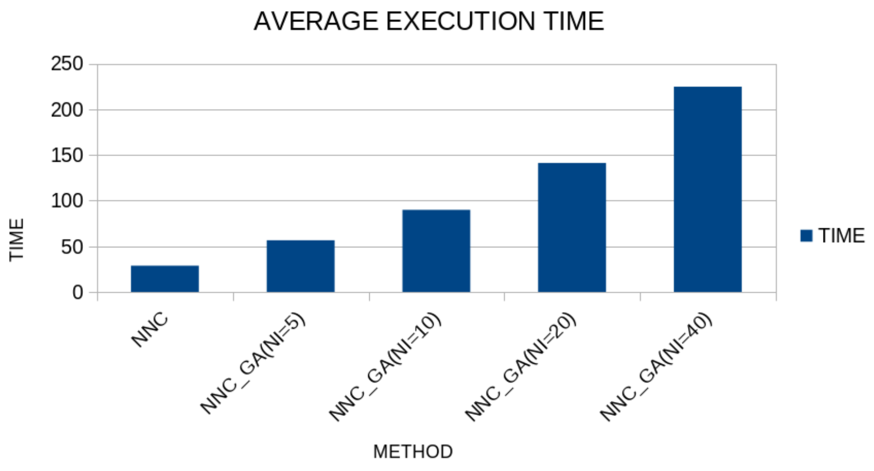


Figure 13. Average execution time for the regression datasets using the original neural network construction method and the proposed one and different values of parameter N_I.

3.5. *A series of practical examples*

As practical applications of the proposed method to real - world problems we consider two cases from the recent bibliography. In the first case, consider the prediction of the duration of forest fires as presented for the Greek territory in a recent publication [109]. Using data from the Greek Fire Service, an attempt is made to predict the duration of forest fires for the years 2014-2023. Figure 14 depicts a comparison for the classification error for this problem for the years 2014-2023 between the original Neural Network Construction method, denoted as NNC and the proposed method which is denoted as NNC_GA in the plot.

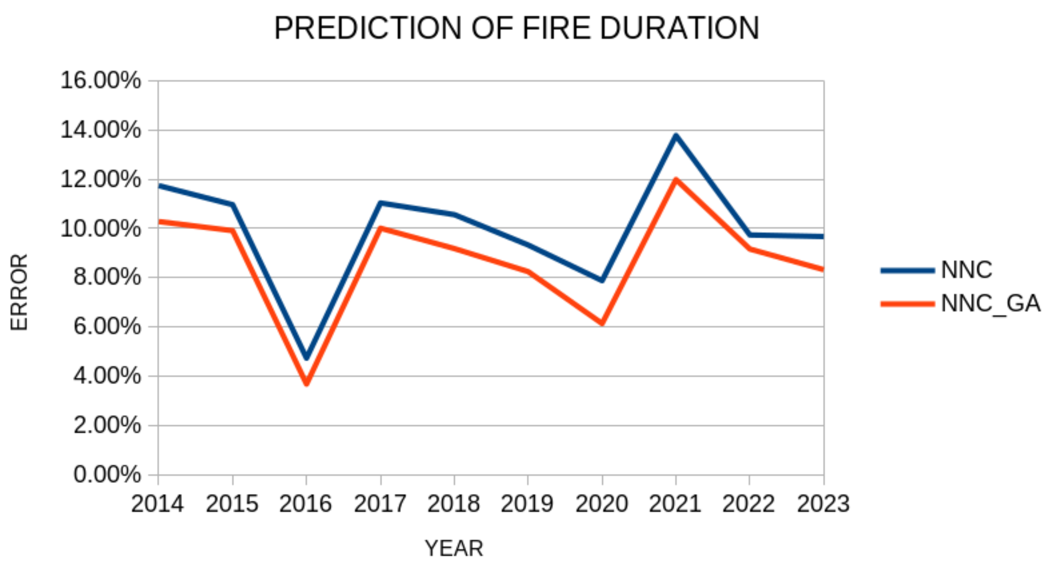


Figure 14. Comparison of the original NNC method and the proposed modification (NNC_GA) for the prediction of forest fires for the Greek teritory. The horizontal axis denotes the year and the vertical the obtained classification error.

As is evident, the proposed method has a lower error in estimating the duration of forest fires in all years from 2014 to 2023 compared to the original artificial neural network construction technique. Although, the proposed method requires significantly more time than the original method as depicted in Figure 15.

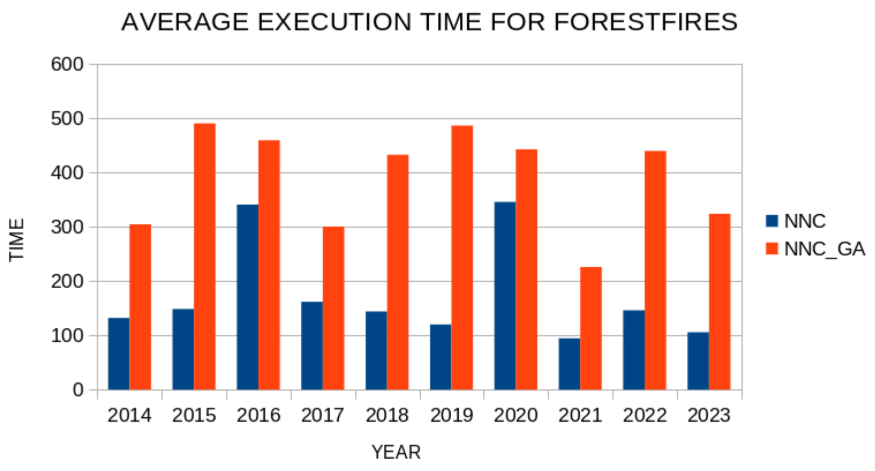


Figure 15. Average execution time for the fores fires problem, using the original neural network construction method and the proposed one.

The second practical example is the PIRvision dataset [110], that contains occupancy detection data that was collected from a Synchronized Low-Energy Electronically-chopped Passive Infra-Red sensing node in residential and office environments. The dataset contains 15302 patterns and the dimension of each pattern is 59. The experimental results validated with ten - fold cross validation, using a series of methods and the proposed one are depicted in Figure 16.

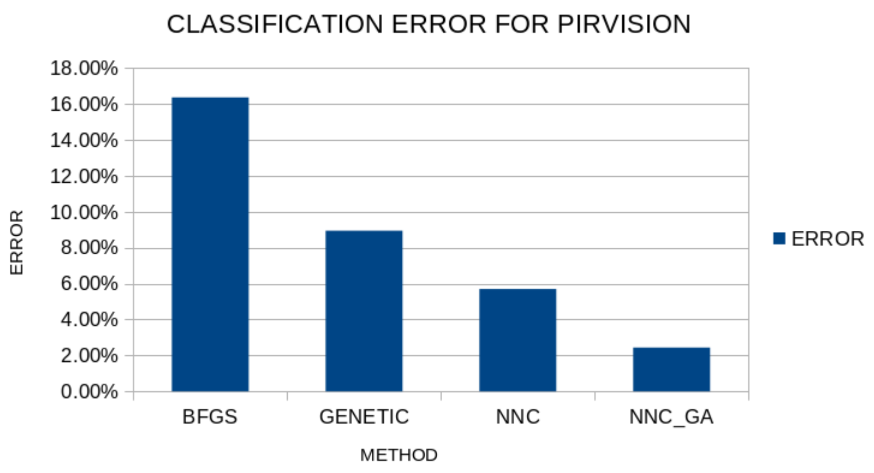


Figure 16. Average classification error for the PIRvision dataset using a series of machine learning methods.

It is evident that the proposed modification of the neural network construction method outperforms significantly the other techniques in terms of average classification error for this particular dataset.

4. Discussion

This study presents an interesting approach combining grammatical evolution with modified genetic algorithms for constructing artificial neural networks. However, a comprehensive analysis of the results and practical implications reveals several aspects that require further investigation and critical examination. While the experimental findings demonstrate certain improvements over traditional techniques, the interpretation and significance of these improvements have not been analyzed with the depth and critical thinking required for a complete method evaluation. Regarding classification performance, the method shows an average error rate of 21.18% compared to ADAM's 36.45% and BFGS's 35.71%. However, these comparative metrics conceal significant performance variations across different datasets. For instance, on Cleveland and Ecoli datasets, classification error reaches 46.41% and 48.82% respectively, while on Housevotes and Zoo it drops below 7%. This substantial performance variation suggests the method may be highly sensitive to dataset-specific characteristics, which isn't sufficiently analyzed in the results presentation. Furthermore, the lack of analysis regarding variation across the 30 repetitions of each experiment raises questions about the method's stability and reliability in real-world applications.The statistical significance of results, while supported by extremely low p-values (1.9e-07 to 1.1e-08), doesn't account for the dynamics of different problem types. In noisy datasets or those with significant class imbalance like Haberman and Liverdisorder, the method shows notable performance fluctuations that remain unexplained. Additionally, the absence of analysis regarding dataset characteristics affecting performance (such as dimensionality, sample size, or degree of linear separability) makes it difficult to determine the optimal conditions for the method's application.Computational resources and execution times present another critical but underexplored issue.

While the study mentions using an AMD Ryzen 5950X system with 128GB RAM, comprehensive reporting of computational requirements is missing. Specifically, it would be essential to present average training times per dataset category (classification vs regression), the method's scalability regarding number of features and samples, memory consumption during the grammatical evolution process, and the impact of various parameters (like population size and generation count) on execution times. This lack of information makes practical implementation assessment challenging, especially for real-world problems where computational resources and time constraints are crucial factors. Regarding limitations, while the study acknowledges issues like local optima and overfitting, their analysis remains superficial. For example, in datasets like Z_O_N_F_S with 39.28% error, it's not investigated whether this results from insufficient solution space exploration due to grammatical evolution parameters, limitations in the grammar used for architecture generation, excessive network complexity leading to overfitting, or inadequacies in parameter training mechanisms.Practical application and robustness require more thorough examination. Beyond controlled experimental scenarios, there's missing information about the ease of applying the method to real-world, unprocessed datasets, the required expertise for optimal parameter tuning, the method's resilience to noisy, incomplete or imbalanced data, and the interpretability of results and generated architectures.Moreover, comparisons with contemporary approaches like transformers, convolutional neural networks, or reinforcement learning methods in domains where they dominate (e.g., natural language processing, computer vision, robotics) are completely absent from the study. This evaluation gap significantly limits our understanding of the method's relative value compared to state-of-the-art alternatives. The method's generalizability to new application domains hasn't been adequately explored. While results are presented across various fields (medicine, physics, economics), critical information is missing about the flexibility and adaptability of the used grammar across different domains, required modifications for new data types (time series, graphs, spatial data), knowledge transfer capability between different applications, and domain knowledge requirements for appropriate grammar design. In summary, while the proposed method introduces interesting mechanisms for improving automated neural network design, this analysis reveals numerous aspects needing further investigation.

For a complete and objective evaluation, it would be necessary to conduct a much more detailed analysis of result stability and variation, comparisons with alternative contemporary approaches beyond the basic techniques examined, thorough evaluation of scalability and computational requirements for large datasets, in-depth investigation of real-world implementation challenges and limitations, and analysis of generalization and adaptation capability to new domains and data types. Only through such a holistic and critical approach could we obtain a complete picture of this methodology's value, capabilities and limitations. The current results, while encouraging, leave significant gaps in our understanding of how and under what conditions the method can truly provide value compared to existing approaches in automated neural network design.

5. Conclusions

The article presents a method for constructing artificial neural networks by integrating Grammatical Evolution (GE) with a modified Genetic Algorithm (GA) to improve generalization properties and reduce overfitting. The method proves to be effective in designing neural network architectures and optimizing their parameters. The modified genetic algorithm avoids local minima during training and addresses overfitting by applying penalty factors to the fitness function, ensuring better generalization to unseen data. The method was evaluated on a variety of classification and regression datasets from diverse fields, including physics, chemistry, medicine, and economics. Comparative results indicate that the proposed method achieves lower error rates on average compared to traditional optimization and machine learning techniques, highlighting its stability and adaptability. The results, analyzed through statistical metrics such as p-values, provide strong evidence of the method's superiority over competing models in both classification and regression tasks. A key innovation of the method is the combination of dynamic architecture generation and parameter optimization within a unified framework. This approach not only enhances performance but also reduces the computational complexity associated with manually designing neural networks. Additionally, the use of constraint techniques in the genetic algorithm ensures the preservation of the neural network structure while enabling controlled optimization of parameters. Future explorations could focus on testing the method on larger and more complex datasets, such as those encountered in image recognition, natural language processing, and genomics, to evaluate its scalability and effectiveness in real-world applications. Furthermore, the integration of other global optimization methods, such as Particle Swarm Optimization, Simulated Annealing, or Differential Evolution, could be considered to further enhance the algorithm's robustness and convergence speed. Concurrently, the inclusion of regularization techniques, such as dropout or batch normalization, could improve the method's generalization capabilities even further. Reducing computational cost is another important area of investigation, and the method could be adapted to leverage parallel computing architectures, such as GPUs or distributed systems, making it feasible for training on large datasets or for real-time applications. Finally, customizing the grammar used in Grammatical Evolution based on the specific characteristics of individual fields could improve the method's performance in specialized tasks, such as time-series forecasting or anomaly detection in cybersecurity.

The proposed technique attempts to maintain the parameters of artificial neural networks within ranges of values in which the neural network is likely to have good generalization properties. A possible future improvement could be to find this interval of values either with some technique that utilizes derivatives or with interval arithmetic techniques [111,112].

Author Contributions: V.C. and I.G.T. conducted the experiments, employing several datasets and provided the comparative experiments. D.T. and 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.

Institutional Review Board Statement: Not applicable.

Institutional Review Board Statement: Not applicable.

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

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

References

1. Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11).
2. Suryadevara, S., & Yanamala, A. K. Y. (2021). A Comprehensive Overview of Artificial Neural Networks: Evolution, Architectures, and Applications. Revista de Inteligencia Artificial en Medicina, 12(1), 51-76.
3. M. Egmont-Petersen, D. de Ridder, H. Handels, Image processing with neural networks—a review, Pattern Recognition **35**, pp. 2279-2301, 2002.
4. G.Peter Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing **50**, pp. 159-175, 2003.
5. Z. Huang, H. Chen, C.-Jung Hsu, W.-Hwa Chen, S. Wu, Credit rating analysis with support vector machines and neural networks: a market comparative study, Decision Support Systems **37**, pp. 543-558, 2004.
6. P. Baldi, K. Cranmer, T. Faucett et al, Parameterized neural networks for high-energy physics, Eur. Phys. J. C **76**, 2016.
7. Baldi, P., Cranmer, K., Faucett, T., Sadowski, P., & Whiteson, D. (2016). Parameterized neural networks for high-energy physics. The European Physical Journal C, 76(5), 1-7.
8. Vora, K., & Yagnik, S. (2014). A survey on backpropagation algorithms for feedforward neural networks. International Journal of Engineering Development and Research, 1(3), 193-197.
9. K. Vora, S. Yagnik, A survey on backpropagation algorithms for feedforward neural networks, International Journal of Engineering Development and Research **1**, pp. 193-197, 2014.
10. Pajchrowski, T., Zawirski, K., & Nowopolski, K. (2014). Neural speed controller trained online by means of modified RPROP algorithm. IEEE transactions on industrial informatics, 11(2), 560-568.
11. Hermanto, R. P. S., & Nugroho, A. (2018). Waiting-time estimation in bank customer queues using RPROP neural networks. Procedia Computer Science, 135, 35-42.
12. 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.

13. Reynolds, J., Rezgui, Y., Kwan, A., & Piriou, S. (2018). A zone-level, building energy optimisation combining an artificial neural network, a genetic algorithm, and model predictive control. Energy, 151, 729-739.
14. Das, G., Pattnaik, P. K., & Padhy, S. K. (2014). Artificial neural network trained by particle swarm optimization for non-linear channel equalization. Expert Systems with Applications, 41(7), 3491-3496.
15. Sexton, R. S., Dorsey, R. E., & Johnson, J. D. (1999). Beyond backpropagation: using simulated annealing for training neural networks. Journal of Organizational and End User Computing (JOEUC), 11(3), 3-10.
16. Wang, L., Zeng, Y., & Chen, T. (2015). Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. Expert Systems with Applications, 42(2), 855-863.
17. Karaboga, D., & Akay, B. (2007, June). Artificial bee colony (ABC) algorithm on training artificial neural networks. In 2007 IEEE 15th Signal Processing and Communications Applications (pp. 1-4). IEEE.
18. 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.
19. J.-R. Zhang, J. Zhang, T.-M. Lok, M.R. Lyu, A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training, Applied Mathematics and Computation **185**, pp. 1026-1037, 2007.
20. G. Zhao, T. Wang, Y. Jin, C. Lang, Y. Li, H. Ling, The Cascaded Forward algorithm for neural network training, Pattern Recognition **161**, 111292, 2025.
21. K-Su Oh, K. Jung, GPU implementation of neural networks, Pattern Recognition **37**, pp. 1311-1314, 2004.
22. M. Zhang, K. Hibi, J. Inoue, GPU-accelerated artificial neural network potential for molecular dynamics simulation, Computer Physics Communications **285**, 108655, 2023.
23. S.J. Nowlan and G.E. Hinton, Simplifying neural networks by soft weight sharing, Neural Computation 4, pp. 473-493, 1992.
24. Nowlan, S. J., & Hinton, G. E. (2018). Simplifying neural networks by soft weight sharing. In The mathematics of generalization (pp. 373-394). CRC Press.
25. 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.
26. M. Augasta and T. Kathirvalavakumar, Pruning algorithms of neural networks — a comparative study, Central European Journal of Computer Science, 2003.
27. Lutz Prechelt, Automatic early stopping using cross validation: quantifying the criteria, Neural Networks **11**, pp. 761-767, 1998.
28. X. Wu and J. Liu, A New Early Stopping Algorithm for Improving Neural Network Generalization, 2009 Second International Conference on Intelligent Computation Technology and Automation, Changsha, Hunan, 2009, pp. 15-18.
29. N. K. Treadgold and T. D. Gedeon, Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm,IEEE Transactions on Neural Networks **9**, pp. 662-668, 1998.
30. 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.
31. J. Arifovic, R. Gençay, Using genetic algorithms to select architecture of a feedforward artificial neural network, Physica A: Statistical Mechanics and its Applications **289**, pp. 574-594, 2001.
32. P.G. Benardos, G.C. Vosniakos, Optimizing feedforward artificial neural network architecture, Engineering Applications of Artificial Intelligence **20**, pp. 365-382, 2007.
33. B.A. Garro, R.A. Vázquez, Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms, Computational Intelligence and Neuroscience, 369298, 2015.
34. Siebel, N. T., & Sommer, G. (2007). Evolutionary reinforcement learning of artificial neural networks. International Journal of Hybrid Intelligent Systems, 4(3), 171-183.
35. Jaafra, Y., Laurent, J. L., Deruyver, A., & Naceur, M. S. (2019). Reinforcement learning for neural architecture search: A review. Image and Vision Computing, 89, 57-66.
36. Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In International conference on machine learning (pp. 4095-4104). PMLR.
37. Xie, S., Zheng, H., Liu, C., & Lin, L. (2018). SNAS: stochastic neural architecture search. arXiv preprint arXiv:1812.09926.
38. Zhou, H., Yang, M., Wang, J., & Pan, W. (2019, May). Bayesnas: A bayesian approach for neural architecture search. In International conference on machine learning (pp. 7603-7613). PMLR.
39. L. Terfloth, J. Gasteige, Neural networks and genetic algorithms in drug design, Drug Discovery Today **6**, pp. 102-108, 2001.
40. Kim, G. H., Seo, D. S., & Kang, K. I. (2005). Hybrid models of neural networks and genetic algorithms for predicting preliminary cost estimates. Journal of computing in civil engineering, 19(2), 208-211.
41. Kalogirou, S. A. (2004). Optimization of solar systems using artificial neural-networks and genetic algorithms. Applied Energy, 77(4), 383-405.
42. Tong, D.L., Mintram, R. Genetic Algorithm-Neural Network (GANN): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection. Int. J. Mach. Learn. & Cyber. 1, 75–87 (2010).
43. Ruehle, F. Evolving neural networks with genetic algorithms to study the string landscape. J. High Energ. Phys. 2017, 38 (2017).
44. M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5**,pp. 349–358, 2001.
45. I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing **72**, pp. 269-277, 2008.

46. G.V. Papamokos, I.G. Tsoulos, I.N. Demetropoulos, E. Glavas, Location of amide I mode of vibration in computed data utilizing constructed neural networks, Expert Systems with Applications **36**, pp. 12210-12213, 2009.
47. I.G. Tsoulos, D. Gavrilis, E. Glavas, Solving differential equations with constructed neural networks, Neurocomputing **72**, pp. 2385-2391, 2009.
48. I.G. Tsoulos, G. Mitsi, A. Stavrakoudis, S. Papapetropoulos, Application of Machine Learning in a Parkinson's Disease Digital Biomarker Dataset Using Neural Network Construction (NNC) Methodology Discriminates Patient Motor Status, Frontiers in ICT 6, 10, 2019.
49. V. Christou, I.G. Tsoulos, V. Loupas, A.T. Tzallas, C. Gogos, P.S. Karvelis, N. Antoniadis, E. Glavas, N. Giannakeas, Performance and early drop prediction for higher education students using machine learning, Expert Systems with Applications **225**, 120079, 2023.
50. E.I. Toki, J. Pange, G. Tatsis, K. Plachouras, I.G. Tsoulos, Utilizing Constructed Neural Networks for Autism Screening, Applied Sciences **14**, 3053, 2024.
51. 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.
52. C. Ryan, J. Collins, M. O'Neill, Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1998. Lecture Notes in Computer Science, vol 1391. Springer, Berlin, Heidelberg, 1998.
53. M. O'Neill, M., C. Ryan, Evolving Multi-line Compilable C Programs. In: Poli, R., Nordin, P., Langdon, W.B., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1999. Lecture Notes in Computer Science, vol 1598. Springer, Berlin, Heidelberg, 1999.
54. 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.
55. 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.
56. 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.
57. D. Martínez-Rodríguez, J. M. Colmenar, J. I. Hidalgo, R.J. Villanueva Micó, S. Salcedo-Sanz, Particle swarm grammatical evolution for energy demand estimation, Energy Science and Engineering **8**, pp. 1068-1079, 2020.
58. C. Ryan, M. Kshirsagar, G. Vaidya, G. et al. Design of a cryptographically secure pseudo random number generator with grammatical evolution. Sci Rep **12**, 8602, 2022.
59. C. Martín, D. Quintana, P. Isasi, Grammatical Evolution-based ensembles for algorithmic trading, Applied Soft Computing **84**, 105713, 2019.
60. Anastasopoulos, N., Tsoulos, I.G., Karvounis, E. et al. Locate the Bounding Box of Neural Networks with Intervals. Neural Process Lett 52, 2241–2251 (2020).
61. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository, https://archive.ics.uci.edu.
62. 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.
63. Weiss, Sholom M. and Kulikowski, Casimir A., Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems, Morgan Kaufmann Publishers Inc, 1991.
64. 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.
65. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
66. T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, Machine Learning **16**, pp. 59-88, 1994.
67. 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.
68. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
69. 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.
70. 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.
71. 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.
72. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
73. 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.

74. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
75. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
76. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
77. J. Mcdermott, R.S. Forsyth, Diagnosing a disorder in a classification benchmark, Pattern Recognition Letters **73**, pp. 41-43, 2016.
78. 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.
79. 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.
80. 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.
81. 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.
82. 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.
83. 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.
84. 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.
85. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
86. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
87. 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.
88. 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.
89. 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.
90. 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.
91. 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.
92. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
93. 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.
94. 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
95. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
96. 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
97. 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).
98. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
99. Friedman, J. (1991): Multivariate Adaptative Regression Splines. Annals of Statistics, 19:1, 1--141.
100. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
101. I.G. Tsoulos, V. Charilogis, G. Kyrou, V.N. Stavrou, A. Tzallas, Journal of Open Source Software **10**, 7584, 2025.

102. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
103. J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation **3**, pp. 246-257, 1991.
104. G.A. Montazer, D. Giveki, M. Karami, H. Rastegar, Radial basis function neural networks: A review. Comput. Rev. J **1**, pp. 52-74, 2018.
105. K. O. Stanley, R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, Evolutionary Computation **10**, pp. 99-127, 2002.
106. Zhu, V., Lu, Y., & Li, Q. (2006). MW-OBS: An improved pruning method for topology design of neural networks. Tsinghua Science and Technology, 11(4), 307-312.
107. Grzegorz Klima, Fast Compressed Neural Networks, available from http://fcnn.sourceforge.net/.
108. Ward, R., Wu, X., & Bottou, L. (2020). Adagrad stepsizes: Sharp convergence over nonconvex landscapes. Journal of Machine Learning Research, 21(219), 1-30.
109. C. Kopitsa, I.G. Tsoulos, V. Charilogis, A. Stavrakoudis, Predicting the Duration of Forest Fires Using Machine Learning Methods, Future Internet **16**, 396, 2024.
110. Emad-Ud-Din, M., & Wang, Y. (2023). Promoting occupancy detection accuracy using on-device lifelong learning. IEEE Sensors Journal, 23(9), 9595-9606.
111. M.A. Wolfe, Interval methods for global optimization, Applied Mathematics and Computation **75**, pp. 179-206, 1996.
112. T. Csendes and D. Ratz, Subdivision Direction Selection in Interval Methods for Global Optimization, SIAM J. Numer. Anal. **34**, pp. 922–938, 1997.