Article

Train neural networks with a hybrid method that incorporates a novel simulated annealing procedure

Ioannis G. Tsoulos[1,*], Vasileios Charilogis[2] and 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: In this paper, an innovative hybrid technique is proposed for the efficient training of artificial neural networks, which are used both in class learning problems and in data fitting problems. This hybrid technique combines the well-tested technique of Genetic Algorithms with an innovative variant of Simulated Annealing, in order to achieve high learning rates for the neural networks. This variant was applied periodically to randomly selected chromosomes from the population of the Genetic Algorithm in order to reduce the training error achieved by these chromosomes. The proposed method was tested on a wide series of classification and data fitting problems from the relevant literature and the results were compared against other methods. The comparison with other neural network training techniques as well as the statistical comparison revealed that the proposed method is significantly superior, as it managed to significantly reduce the neural network training error in the majority of the used datasets.

Keywords: Artificial neural networks; Evolutionary techniques; Genetic algorithms; Simulated Annealing

Citation: Tsoulos, I.G.; Charilogis, V.; Tsalikakis, D. Train neural networks with a hybrid method that incorporates a novel simulated annealing procedure. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

Artificial Neural networks (ANNs) [1,2] are widely used parametric tools, where a series of methods have been developed to identify the optimal set of these parameters, commonly called weights or processing units. ANNs have been used in a variety of scientific problems, such as problems from physics [3–5], chemistry [6–8], economics [9–11], medicine [12,13] etc. Furthermore, in recent years, neural networks have been incorporated into a variety of practical problems, such as flood simulation [14], solar radiation prediction [15], agricultural problems [16], solution of problems in wireless communications [17], mechanical applications [18] etc.

Commonly, a neural network is expressed as function $N(\overrightarrow{x}, \overrightarrow{w})$, where the vector $\overrightarrow{x}$ expresses the input pattern and the vector $\overrightarrow{w}$ represents the weight vector of the neural network. The methods aimed at training the artificial neural network deal with the efficient adjustment of the weight vector $\overrightarrow{w}$ to minimize the training error defined as:

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

The set set $\left(\overrightarrow{x_i}, y_i\right),\ i = 1, ..., M$ defines the train set for the neural network, where the value y_i represent the the actual output for pattern $\overrightarrow{x_i}$. Neural networks can be expressed also in close analytic form, as show in [19]. As it was shown any neural network can be expressed as function

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

The parameter H defines the number of processing units and the constant d represents the dimension of pattern $\overrightarrow{x}$. The function $\sigma(x)$ is called sigmoid function, and it is defined as:

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

The number of elements in the weight vector are calculated as: $n = (d+2)H$. Of course, other activation functions may be used, such as the tanh function defined as

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

with similar approximation capabilities. Also, Guarnieri et al proposed the usage of an adaptive spline activation function for neural networks [20]. Furthermore, Ertuğrul proposed the trained activation function in neural networks [21]. A systematic review of activation functions for Artificial Neural Networks can be found in the paper of Rasamoelina et al [22].

In the recent bibliography, a series of methods have been proposed to minimize the equation 1, such as the Back Propagation method [23,24], the Levenberg-Marquardt method [25], the RPROP method [26–28], Quasi Newton methods [29,30], Particle Swarm Optimization [31,32], the Differential Evolution method [33], etc. The first four methods are local optimization methods, used in a series of research papers but they can be easily trapped in local minima of the error function of Equation 1. On the other hand, methods like Particle Swarm Optimization or Differential Evolution are considered global optimization methods aiming to discover the global minimum of such functions and as a consequence they can avoid local minima of the error function. A survey of stochastic methods for training neural networks can be found in the work of Zhang and Suganthan [34].

Due to the wide application of artificial neural networks in various fields, but also due to the difficulties faced by traditional optimization techniques in minimizing the training error, a series of hybrid techniques have been developed to more effectively reduce this error. Among these methods there is the method of Yaghini et al. [35] that combines Particle Swarm Optimization and the Back Propagation technique. Also, Chen at al [36] has proposed a hybrid method that incorporates particle swarm optimization and Cuckoo Search [37].

Another important issue of neural networks that has been thoroughly studied in the recent literature is the initialization of the parameters for the network. The methods developed for the initialization issue include utilization of decision trees [38], an initialization technique based on the Cauchy's inequality [39], discriminant learning [40] etc. A recent paper by Narkhede et al. [41] presents various techniques for the initialization of the parameters.

Due to the complexity of the training techniques but also due to the fact that the number of required parameters increases with the increase in the dimension of the problem, a number of training techniques have been developed that take advantage of modern parallel computing structures. For example, there are implementations of neural networks on GPU cards [42], incorporation of GPU programming techniques on neural network training for face recognition [43], molecular dynamics simulation using neural networks that are executed on GPU cards [44] etc. A comparative study of GPU programming models used for neural network training can be found in the work of Pallipuram et al [45].

In this work, the use of a hybrid optimization technique is proposed for the training of artificial neural networks. In this hybrid technique, Genetic Algorithms are used as a basic technique for training neural networks. Genetic algorithms, which were initially suggested by John Holland [46], are inspired by biology, and are form trial solutions of any optimization problem. These solutions are improved gradually by a process that mimics natural evolution, such as mutation, natural selection, and crossover [47–49]. Genetic algorithms have proven their efficiency, and they have been applied on a wide series of problems, such as networking [50], robotics [51,52], energy problems [53,54] etc.

Genetic algorithms have been extensively studied in the modern literature for the training of artificial neural networks or for the efficient creation of their structure. For example, the work of Arifovic et al [55] used to select the optimal architecture of an artificial neural network. Also, Leung et al proposed a novel genetic algorithm [56] to adjust the parameters and the structure of neural networks. Gao et al proposed an efficient genetic algorithm [57] with a new diffusing operator for neural network training. Recently, Ahmadizar et al combined genetic algorithm with grammatical evolution for optimal training of neural networks [58]. Additionally, Kobrunov and Priezzhev suggested a hybrid genetic algorithm [59] for efficient neural network training. Although Genetic Algorithms can satisfactorily train an artificial neural network, in many cases they get trapped in local minimum of the training error and this results in poor performance of the neural network when applied to the test set. To improve the performance of genetic algorithms, it is proposed to periodically apply a minimization technique to randomly selected chromosomes of the genetic population.

This minimization method that is applied here, is a modified version of the Simulated Annealing method [60]. Simulated annealing has been applied in many cases, such as police district design [61], portfolio problems [62], energy problems [63] etc. The new method was tested on a wide series of classification and regression problems, and it was compared against other optimization methods. From the experimental comparison of the results, it appears that the proposed technique significantly improves the performance of genetic algorithms in the training of artificial neural networks.

Genetic algorithms have been used in conjuction with Simulated Annealing in a series of research papers in the recent literature, such as the work of Yu et al that combines genetic algorithm with simulated annealing for large scale system energy integration [64]. Also, Ganesh and Punniyamoorthy used hybrid genetic algorithms for optimization of continuous - time prediction planning [65]. Additionally, Hwang and He suggested the usage of simulated annealing to improve a genetic algorithm that was applied on engineering problems [66]. Furthermore, Li and Wei applied a genetic algorithm that was enhanced with a Simulated Annealing method on multi-reservoir systems [67]. A method that combines a genetic algorithm with simulated annealing was also used in Smart City problems recently [68].

The rest of this article is divided as follows: in section 2 the proposed method is discussed in detail. In section 3 the used datasets are presented as well as the experimental results, and finally, in section 4 the results are discussed thoroughly and some guidelines for future research are provided.

2. The proposed method

The new Simulated Annealing variant is described in this section, as well as the overall algorithm, that will be used to train artificial neural networks for classification and regression problems.

2.1. The new Simulated Annealing variant

A new variant of the Simulated Annealing method is utilized as a local search procedure in the Genetic Algorithm. This method has been applied to many problems and is distinguished for its adaptability but also for the ability to aim for lower values of the objective function, especially if combined with intelligent techniques to reduce the temperature factor. In the proposed modification of the method, the optimization procedure initiates from the current state of a chromosome and by applying stochastic techniques a search is made for nearby representations with lower values of the error function. The steps of the proposed method are illustrated in Algorithm 1.

Algorithm 1 The used variant of the Simulated Annealing algorithm.

procedure siman(x_0)

1. **Set** $k = 0$, $T_0 > 0$, $\epsilon > 0$,$r_T > 0$, $r_T < 1$. The parameter T_0 defines the initial temperature of the algorithm.
2. **Set** $N_{eps} > 0$, a positive integer number. This number defines the number of samples that will be created in every iteration.
3. **Set** the parameter $F \in [0,1]$. This value specifies the range of changes that can be made to an element of a chromosome, as a percentage of its original value.
4. **Set** the positive integer parameter N_R. This parameter indicates the number of possible random changes in the chromosome.
5. **Set** $x_b = x_0, f_b = f(x_b)$.
6. **For** $i = 1 \ldots N_{\mathbf{eps}}$
   (a) **Set** $x^t = x_k$ as a candidate point.
   (b) **For** $j = 1 \ldots N_R$
      i. **Set** $p = \mathbf{rand}\left(1, \mathbf{size}\left(x^t\right)\right)$, a randomly selected position in the chromosome.
      ii. **Set** $x_p^t = x_p^t + \mathbf{rand}(-F, F)x_p^t$
   (c) **EndFor**
   (d) **If** $f\left(x^t\right) \le f(x_k)$ **then** $x_{k+1} = x^t$
   (e) **Else Set** $x_{k+1} = x^t$ with probability $\min\left\{1, \exp\left(-\frac{f\left(x^t\right) - f(x_k)}{T_k}\right)\right\}$
   (f) **If** $f\left(x^t\right) < f_b$ **then** $x_b = x^t, f_b = f\left(x^t\right)$.
7. **EndFor**
8. **Set** $T_{k+1} = T_k r_T$
9. **Set** $k = k + 1$.
10. **If** $T_k \le \epsilon$ **stop.**
11. **Goto step** 6.
12. **Return** x_b

end siman

The method initiates from chromosome x_0 and in every iteration and it produces random points near to the original chromosome. The integer parameter N_R defines the number of changes that will be made in the chromosome and the double precision parameter F controls the magnitude of changes. The algorithm starts from high values of the temperature T_0, which is linearly decreased in each iteration. At high temperatures, the algorithm more readily accepts values that it can with higher function values to more efficiently explore the search space, but at lower values the algorithm focuses on improving the best function value it has discovered.

2.2. The overall algorithm

A genetic algorithm is used as the base algorithm for neural network training. Genetic algorithms have been used also in the recent bibliography for neural network training in various cases, such as for drug design [69], gear fault detection [70], forecasting models [71] etc. The genetic algorithm is enhanced by the addition of a periodical application of the new Simulated Annealing variant, described in the previous subsection. The main steps of the overall algorithm are listed below.

1. **Initialization Step**
   (a) **Define** as N_c the number of chromosomes and as N_g the maximum number of generations.
   (b) **Define** the selection rate p_s and the mutation rate p_m with $p_s \in [0,1]$ and $p_m \in [0,1]$.

   (c) **Set** as N_I the number of generations passed before the modified Simulated Algorithm will be applied.
   (d) **Set** as N_K the number of chromosomes that will be altered by the modified Simulated Annealing algorithm.
   (e) **Perform** a random initialization of the N_c chromosomes. Each chromosome represents a different set of randomly initialized weights for the neural network.
   (f) **Set** $k = 0$.
2. **For** each chromosome $g_i,\ i = 1, \ldots, N_c$
   (a) **Formulate** a neural network $N(\overrightarrow{x}, \overrightarrow{g_i})$
   (b) **Calculate** the fitness $f_i = \sum_{j=1}^{M} \left(N(\overrightarrow{x}_j, \overrightarrow{g_i}) - y_j\right)^2$ of chromosome g_i and for the given dataset.
3. **Genetic operations step**
   (a) **Selection procedure.** The chromosomes are sorted with respect to the associated fitness values. The first $(1 - p_s) \times N_c$ chromosomes having the lowest fitness values are copied to the next generation. The rest of the chromosomes are replaced by offspings produced in the crossover procedure.
   (b) **Crossover procedure**: In the crossover procedure pairs of chromosomes are selected from the population using tournament selection. For each pair (z, w) of selected parents two new chromosomes $\tilde{z}$ and $\tilde{w}$ are formulated using the following scheme

$$\begin{array}{rcl} \tilde{z}_i & = & a_i z_i + (1 - a_i) w_i \\ \tilde{w}_i & = & a_i w_i + (1 - a_i) z_i \end{array} \qquad (5)$$

   where $i = 1, \ldots, n$. The randomly selected values a_i are chosen in the range $[-0.5, 1.5]$ [72].
   (c) **Mutation procedure**:
      i. **For** each chromosome $g_i,\ i = 1, \ldots, N_c$ do
         A. **For** every element $j = 1, \ldots, n$ of g_i a random number $r \in [0, 1]$ is produced. The corresponding element is altered randomly if $r \le p_m$.
      ii. **EndFor**
4. **Local method step**
   (a) **If** $k \bmod N_I = 0$ **then**
      i. **For** $i = 1, \ldots, N_K$ do
         A. **Select** a random chromosome g^r
         B. **Apply** the siman algorithm: $g^r = \mathrm{siman}(g^r)$ of subsection 2.1.
      ii. **EndFor**
   (b) **Endif**
5. **Termination Check Step**
   (a) **Set** $k = k + 1$
   (b) **If** $k \ge N_g$ then goto **Termination** Step, else goto 2b.
6. **Termination step**
   (a) **Denote** as g^* the chromosome with the lowest fitness value.
   (b) **Formulate** the neural network $N(\overrightarrow{x}, \overrightarrow{g^*})$
   (c) **Apply** a local search procedure to g^*. The local search method used in the current work is a BFGS variant of Powell [73].
   (d) **Apply** the neural network $N(\overrightarrow{x}, \overrightarrow{g^*})$ on the test of the objective problem and report the result.

The overall algorithm is also outlined graphically as a series of steps in Figure 1.

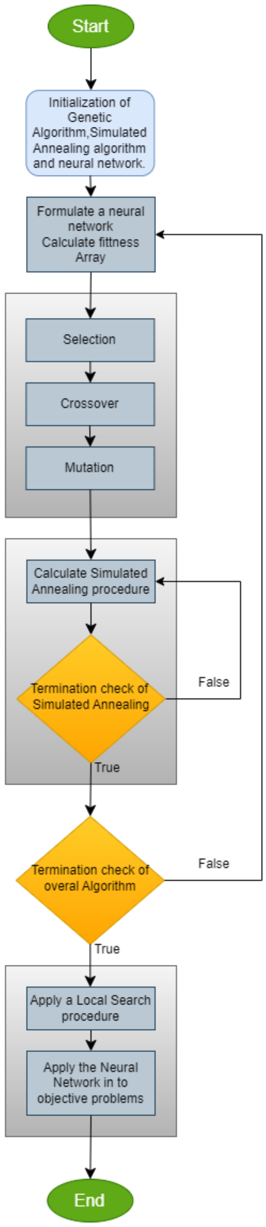

Figure 1. The overall proposed algorithm.

3. Results

The proposed work was tested on a series of well - known classification and regression datasets from the recent bibliography and it was compared other optimization methods, used to train neural networks. The used datasets can be obtained freely from the following websites:

1. The UCI dataset repository, https://archive.ics.uci.edu/ml/index.php(accessed on 18 June 2024)[74]
2. The Keel repository, https://sci2s.ugr.es/keel/datasets.php(accessed on 18 June 2024)[75].
3. The Statlib URL http://lib.stat.cmu.edu/datasets/(accessed on 18 June 2024).

3.1. Classification datasets

A series of classification datasets were used in the conducted experiments. Their description has as follows:

1. **Appendictis** a medical dataset, suggested in [76].
2. **Australian** dataset [77], used in credit card transactions.
3. **Bands** dataset, used to detect printing problems.
4. **Balance** dataset [78], which is related to some psychological experiments.
5. **Circular** dataset, which is an artificial dataset.
6. **Cleveland** dataset, a medical dataset [79,80].
7. **Dermatology** dataset [81], which is a dataset related to dermatological deceases.
8. **Ecoli** dataset, a dataset about protein localization sites of proteins[82].
9. **Fert** dataset. Fertility dataset related to relation of sperm concentration and demographic data.
10. **Heart** dataset [83], a medical dataset used to detect heart diseases.
11. **HeartAttack** dataset, used to predict heart attacks.
12. **HouseVotes** dataset [84], related to votes in the U.S. House of Representatives.
13. **Liverdisorder** dataset [85], used to detect liver disorders.
14. **Parkinsons** dataset, used to detect the Parkinson's disease (PD)[86].
15. **Pima** dataset [87], a medical dataset used to detect the presence of diabetes.
16. **Popfailures** dataset [88], a dataset related to climate measurements.
17. **Regions2** dataset, related to hepatitis C [89].
18. **Saheart** dataset [90], used to detect heart diseases.
19. **Segment** dataset [91], used in image processing tasks.
20. **Sonar** dataset [92], used to discriminate sonar signals.
21. **Spiral** dataset, an artificial dataset.
22. **Wdbc** dataset [93], a medical dataset used to detect cancer..
23. **Wine** dataset, used to detect the quality of wines. [94,95].
24. **Eeg** datasets, a dataset related to EEG measurements [96] and the following cases were used: Z_F_S, ZO_NF_S and ZONF_S.
25. **Zoo** dataset [97], used to classify animals in seven predefined categories.

3.2. Regression datasets

The description of the used regression datasets has as follows:

1. **Airfoil** dataset, a dataset provided by NASA [98].
2. **BK** dataset [99], used for points prediction in a basketball game.
3. **BL** dataset, it contains measurements from an experiment related to electricity.
4. **Baseball** dataset, used to calculate the income of baseball players.
5. **Dee** dataset, used to calculate the price of electricity.
6. **EU,** downloaded from the STALIB repository.
7. **FY,** This dataset measures the longevity of fruit flies.
8. **HO** dataset, downloaded from the STALIB repository.
9. **Housing** dataset, mentioned in [100].
10. **LW** dataset, related to risk factors associated with low weight babies.
11. **MORTGAGE** dataset, related to economic data from USA.
12. **MUNDIAL,** provided from the STALIB repository.
13. **PL** dataset, provided from the STALIB repository.
14. **QUAKE** dataset, that is used to measure the strength of a earthquake.
15. **REALESTATE,** provided from the STALIB repository.
16. **SN** dataset. It contains measurements from an experiment related to trellising and pruning.
17. **Treasury** dataset, related to economic data from USA.
18. **VE** dataset, provided from the STALIB repository.

3.3. Experimental results

A series of experiments were conducted to test the efficiency of the used method as well as its stability. The experiments were conducted using the freely available optimization environment of Optimus, that can be downloaded from https://github.com/itsoulos/

GlobalOptimus/(accessed on 18 June 2024). The experiments were conducted 30 times using different seeds for the random generator each time. The experiments were validated using the method of 10 - fold cross validation. The average classification error is reported for the classification datasets and the average regression error is shown for the regression error. The errors are reported on the test set. The experiments were executed on a system equipped with an AMD Ryzen 5950X processor, 128GB of RAM. The used operating system was the Debian Linux operating system. The values of the parameters for all used algorithms are shown in Table 1.

Table 1. Values for the experimental parameters.

PARAMETER	MEANING	VALUE
N_c	Number of chromosomes	500
N_g	Number of generations	200
L_I	Number of generations for local search	20
L_K	Number of chromosomes in local search	20
p_s	Selection rate	0.10
p_m	Mutation rate	0.05
H	Number of weights	10
F	Range of changes in Simulated Annealing	0.10
N_R	Number of changes in Simulated Annealing	20

The comparative results for the classification datasets are listed in Table 2 and the results for the regression datasets are shown in Table 3. The following applies to all tables with experimental results:

1. The column DATASET denotes the name of the used dataset.
2. The column BFGS denotes the application of the BFGS optimization method to train a neural network with H processing nodes. The method used here is the BFGS variant of Powell [73].
3. The column PSO denotes the application of a Particle Swarm Optimizer with N_c particles to train a neural network with H processing nodes. In the current work the improved PSO method as suggested by Charilogis and Tsoulos is used [101].
4. The column GENETIC stands for the application of a Genetic Algorithm with the parameters shown in Table 1 to train a neural network with H processing nodes. The genetic algorithm used here is a variant proposed by Tsoulos [102].
5. The column PROPOSED denotes the application of the proposed method with the parameters of Table 1 on a neural network with H hidden nodes.
6. The row AVERAGE denotes the average classification or regression error for all datasets.

Table 2. Experimental results using a series of optimization methods for the classification datasets. Numbers in cells denote average classification error as measured on the test set. The bold notation is used to identify the method with the lowest average classification error.

DATASET	BFGS	PSO	GENETIC	PROPOSED
APPENDICITIS	**18.00%**	25.00%	24.40%	22.60%
AUSTRALIAN	38.13%	38.30%	36.64%	**32.42%**
BALANCE	8.64%	**7.97%**	8.36%	8.10%
BANDS	36.67%	36.61%	34.92%	**34.53%**
CIRCULAR	6.08%	**4.24%**	5.13%	4.35%
CLEVELAND	77.55%	62.31%	57.21%	**42.62%**
DERMATOLOGY	52.92%	17.69%	16.60%	**12.12%**
ECOLI	69.52%	61.30%	54.67%	**47.18%**
FERT	**23.20%**	24.00%	28.50%	25.20%
HEART	39.44%	34.67%	26.41%	**16.59%**
HEARTATTACK	46.67%	37.83%	29.03%	**20.13%**
HOUSEVOTES	7.13%	7.87%	**7.00%**	7.13%
LIVERDISORDER	42.59%	39.82%	37.09%	**32.88%**
PARKINSONS	27.58%	23.58%	**16.58%**	16.63%
PIMA	35.59%	35.17%	34.21%	**30.08%**
POPFAILURES	5.24%	7.80%	**4.17%**	5.44%
REGIONS2	36.28%	31.43%	33.53%	**27.69%**
SAHEART	37.48%	34.80%	34.85%	**34.56%**
SEGMENT	68.97%	53.88%	46.30%	**28.41%**
SONAR	25.85%	24.70%	22.40%	**19.80%**
SPIRAL	47.99%	46.31%	47.67%	**44.54%**
WDBC	29.91%	9.98%	7.87%	**5.66%**
WINE	59.71%	32.71%	22.88%	**10.59%**
Z_F_S	39.37%	38.73%	24.60%	**11.10%**
ZO_NF_S	43.04%	30.38%	21.54%	**6.86%**
ZONF_S	15.62%	6.92%	4.36%	**2.48%**
ZOO	10.70%	9.20%	9.50%	**7.60%**
AVERAGE	**35.18%**	**29.01%**	**25.79%**	**20.64%**

Table 3. Experimental results for different optimization methods on a series of regression datasets. Numbers in cells denote average regression error as measure on the test set. The bold notation is used to express the method with the lowest average regression error.

DATASET	BFGS	PSO	GENETIC	PROPOSED
AIRFOIL	0.003	0.001	0.001	**0.001**
BK	0.36	0.33	0.26	**0.18**
BL	1.09	2.49	2.23	**0.42**
BASEBALL	119.63	82.81	64.60	**57.47**
DEE	2.36	0.43	0.47	**0.23**
EU	607.61	407.35	252.97	**216.65**
FY	0.19	**0.05**	0.65	0.23
HO	0.62	**0.03**	0.37	0.06
HOUSING	97.38	43.28	35.97	**23.77**
LW	0.26	**0.03**	0.54	0.27
MORTGAGE	8.23	1.47	0.40	**0.05**
MUNDIAL	**0.05**	0.08	1.22	0.28
PL	0.11	0.06	0.03	**0.02**
QUAKE	0.09	0.06	0.12	**0.06**
REALESTATE	128.94	81.41	81.19	**72.95**
SN	0.16	0.40	0.20	**0.05**
TREASURY	9.91	2.32	0.44	**0.26**
VE	1.92	**0.32**	2.43	1.63
AVERAGE	**54.38**	**34.61**	**24.67**	**20.81**

The statistical comparison between the used methods for the classification datasets is shown in Figure 2.

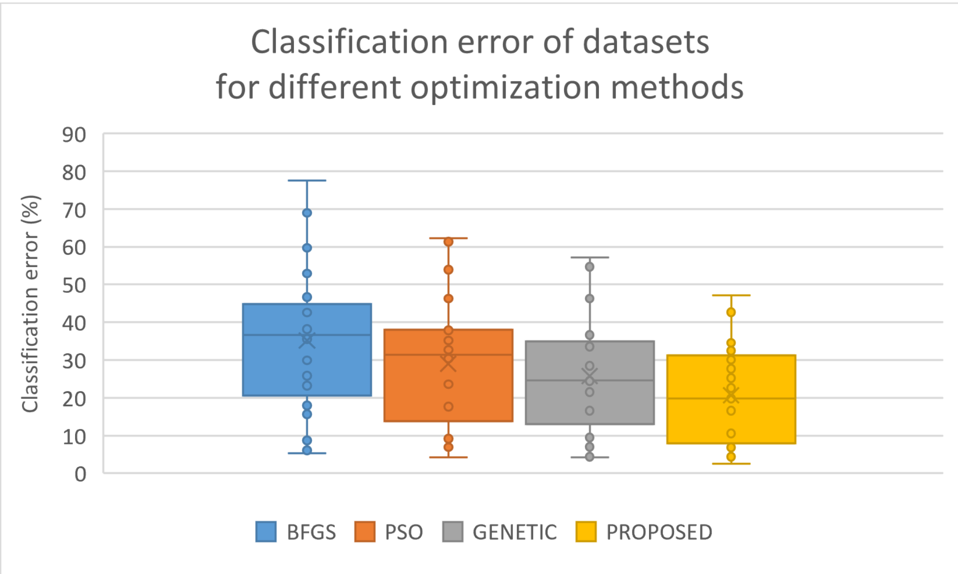

Figure 2. Statistical comparison of the used optimization methods for the classification datasets.

As the comparison of the experimental results and their statistical comparison shows, the genetic algorithm method significantly outperforms the others in terms of accuracy. However, the proposed technique, which is an extension of genetic algorithms, significantly improves their performance on almost all datasets. In several datasets, the reduction in error in the test set can reach up to 80% compared to genetic algorithms.

Nevertheless, the proposed method significantly may increase the total execution time and this can be observed from the graph of Figure 3.

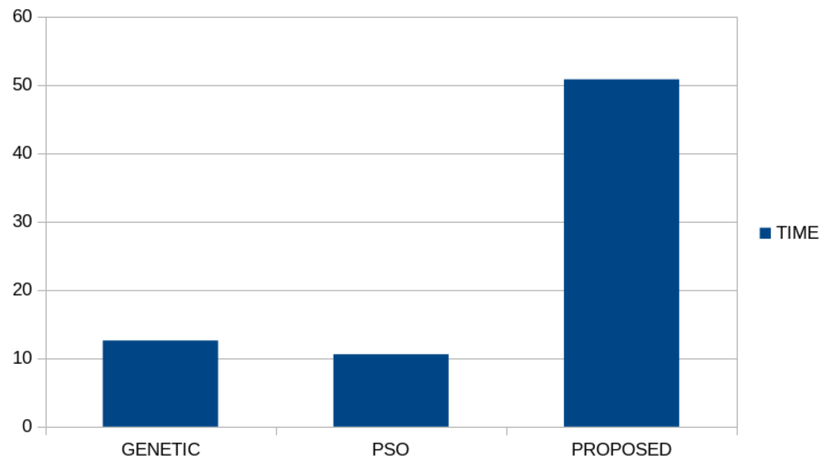

Figure 3. Time comparison of the average execution time for the classification datasets.

The proposed technique significantly improves the performance of the genetic algorithm in minimizing the training error of the artificial neural networks but requires significantly more computing time. However, the required computing time could be significantly reduced either by using parallel computing techniques.

Additionally, in order to explore the stability and the robustness of the proposed method, further experiments were conducted with different values for the critical parameters of the method. The results in Table 4 depict the application of the proposed method on classification datasets, with different values for the critical parameter F, which controls the magnitude of changes in the Simulated Annealing variant.

Table 4. Experimental results using different values for the critical parameter F. The experiments were executed on the classification datasets. The numbers in cells denote average classification error, as measured on the test set.

DATASET	$F = 0.05$	$F = 0.10$	$F = 0.15$
APPENDICITIS	22.30%	22.60%	24.20%
AUSTRALIAN	33.78%	32.42%	28.72%
BALANCE	8.16%	8.10%	8.26%
BANDS	34.81%	34.53%	33.97%
CIRCULAR	4.22%	4.35%	4.38%
CLEVELAND	46.24%	42.62%	44.58%
DERMATOLOGY	16.69%	12.12%	9.94%
ECOLI	50.64%	47.18%	45.24%
FERT	26.60%	25.20%	25.90%
HEART	23.96%	16.59%	15.15%
HEARTATTACK	25.70%	20.13%	19.97%
HOUSEVOTES	6.74%	7.13%	7.44%
LIVERDISORDER	34.50%	32.88%	32.50%
PARKINSONS	16.53%	16.63%	15.68%
PIMA	33.18%	30.08%	26.33%
POPFAILURES	4.52%	5.44%	5.89%
REGIONS2	30.86%	27.69%	26.40%
SAHEART	35.68%	34.56%	32.67%
SEGMENT	32.53%	28.41%	26.15%
SONAR	21.40%	19.80%	19.80%
SPIRAL	45.15%	44.54%	44.23%
WDBC	7.38%	5.66%	4.91%
WINE	16.06%	10.59%	8.82%
Z_F_S	18.20%	11.10%	8.60%
ZO_NF_S	16.80%	6.86%	6.22%
ZONF_S	2.92%	2.48%	2.42%
ZOO	7.60%	7.60%	6.80%
AVERAGE	**23.08%**	**20.64%**	**19.82%**

The proposed method is shown to improve when the critical parameter F increases from 0.05 to 0.10 but does not improve further for a larger increase in the value of the parameter. Therefore, for small changes in chromosome values there is no significant improvement from applying the minimization technique, but larger variations yield more significant reductions in classification error. Also, an experiment was conducted using different values for the parameter N_R, which determines the number of changes in the chromosomes. The results for this experiment and for the classification datasets are shown in Table 5 and the statistical comparison is shown in Figure 4.

Table 5. Experiments using the parameter N_R of the proposed algorithm. The experiments were performed by applying the proposed method on the used classification datasets. The numbers in cells stand for the average classification error, as measured on the corresponding test set.

DATASET	$N_R = 10$	$N_R = 20$	$N_R = 30$
APPENDICITIS	23.70%	22.60%	22.50%
AUSTRALIAN	32.60%	32.42%	31.51%
BALANCE	8.36%	8.10%	8.05%
BANDS	34.28%	34.53%	33.75%
CIRCULAR	4.48%	4.35%	4.51%
CLEVELAND	43.38%	42.62%	43.24%
DERMATOLOGY	13.97%	12.12%	11.26%
ECOLI	47.79%	47.18%	47.06%
FERT	26.50%	25.20%	26.70%
HEART	20.67%	16.59%	16.18%
HEARTATTACK	23.20%	20.13%	20.43%
HOUSEVOTES	7.30%	7.13%	7.44%
LIVERDISORDER	32.50%	32.88%	33.09%
PARKINSONS	16.63%	16.63%	15.26%
PIMA	31.89%	30.08%	28.04%
POPFAILURES	4.43%	5.44%	5.48%
REGIONS2	29.71%	27.69%	26.99%
SAHEART	34.28%	34.56%	33.26%
SEGMENT	29.19%	28.41%	27.46%
SONAR	20.95%	19.80%	20.05%
SPIRAL	44.17%	44.54%	44.20%
WDBC	6.48%	5.66%	5.45%
WINE	12.76%	10.59%	10.41%
Z_F_S	13.50%	11.10%	8.70%
ZO_NF_S	15.14%	6.86%	7.28%
ZONF_S	2.44%	2.48%	2.38%
ZOO	7.40%	7.60%	7.60%
AVERAGE	**21.77%**	**20.64%**	**20.31%**

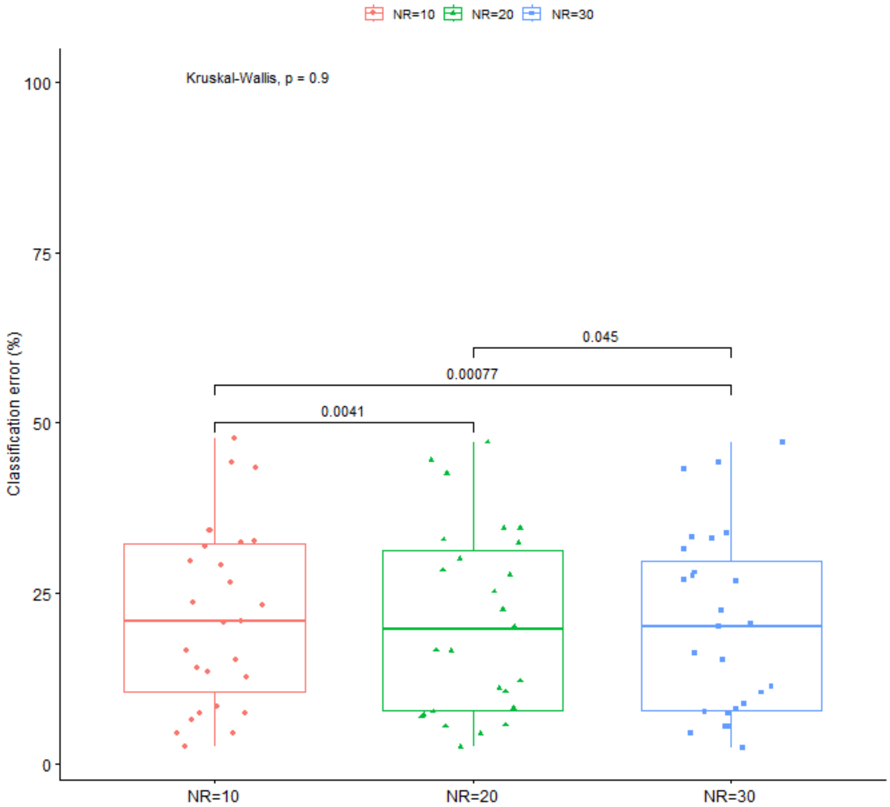

Figure 4. Statistical comparison for the results obtained by the proposed method as applied on the classification datasets, using different values of the parameter N_R.

311 In the case of this parameter, no noticeable differences are observed as the value of the parameter increases. This means that even a limited number of changes (e.g. 10-20) can yield significant reductions in classification errors. Finally, an experiment was conducted to measure the effect of the parameter N_I to the produced results. The experimental results for different values of the N_I parameter are shown in Table 6 and the statistical comparison is depicted in Figure 5.

Table 6. Experiments using the proposed method on the classification datasets for various values of the parameter N_I. Numbers in cells denote average classification error as measured on the test set.

DATASET	$L_I = 10$	$L_I = 20$	$L_I = 30$
APPENDICITIS	24.20%	22.60%	24.10%
AUSTRALIAN	30.49%	32.42%	33.22%
BALANCE	8.50%	8.10%	8.44%
BANDS	34.08%	34.53%	34.22%
CIRCULAR	4.29%	4.35%	4.36%
CLEVELAND	44.58%	42.62%	43.10%
DERMATOLOGY	10.63%	12.12%	12.54%
ECOLI	45.24%	47.18%	47.67%
FERT	25.90%	25.20%	27.30%
HEART	15.44%	16.59%	19.26%
HEARTATTACK	19.87%	20.13%	21.83%
HOUSEVOTES	7.44%	7.13%	6.65%
LIVERDISORDER	32.50%	32.88%	32.85%
PARKINSONS	15.89%	16.63%	15.79%
PIMA	28.96%	30.08%	31.28%
POPFAILURES	5.13%	5.44%	4.76%
REGIONS2	25.74%	27.69%	28.98%
SAHEART	32.67%	34.56%	34.33%
SEGMENT	26.55%	28.41%	28.62%
SONAR	19.80%	19.80%	21.69%
SPIRAL	43.82%	44.54%	43.85%
WDBC	5.48%	5.66%	5.95%
WINE	8.82%	10.59%	11.65%
Z_F_S	8.60%	11.10%	12.13%
ZO_NF_S	6.22%	6.86%	9.06%
ZONF_S	2.42%	2.48%	2.64%
ZOO	6.80%	7.60%	7.10%
AVERAGE	**20.00%**	**20.64%**	**21.24%**

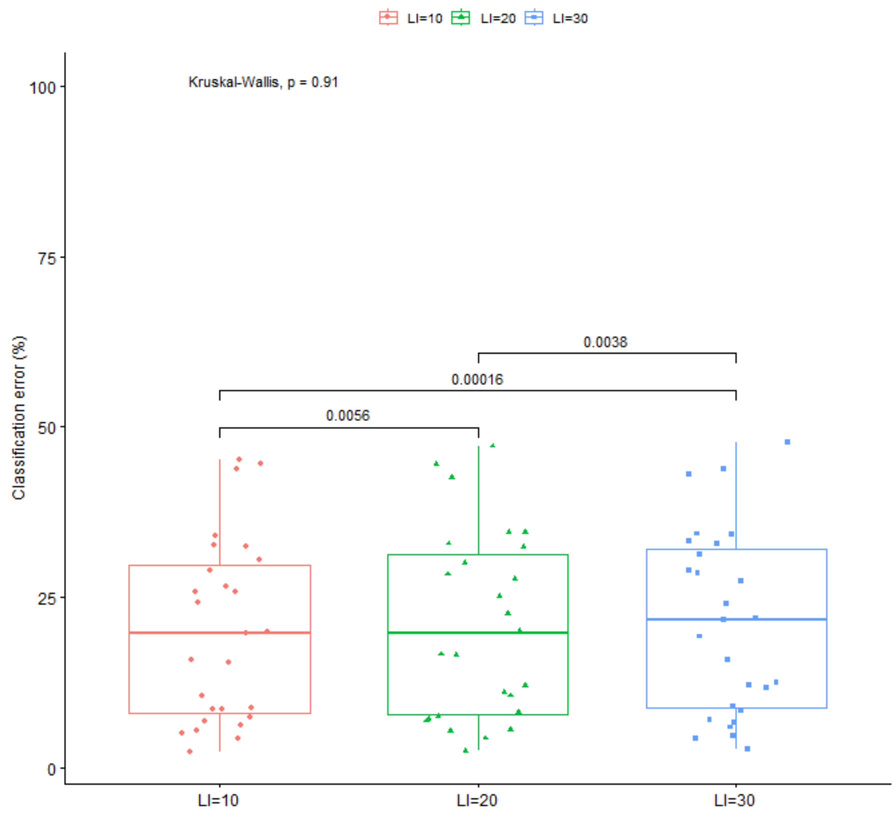

Figure 5. Statistical comparison for the results obtained by the proposed method as applied on the classification datasets, using different values of the parameter L_I.

Once again, the performance of the proposed technique appears not to be significantly affected by the change of parameter N_I. The method performs slightly better for lower values of the N_I parameter, since the smaller this parameter is, the more often the variant of Simulated Annealing will be applied to the genetic population. However, this reduction is limited and therefore there does not appear to be a drastic effect of this particular parameter on the behavior of the algorithm.

4. Conclusions

A new variant of the Simulated Annealing method is introduced in the current work, which aims to improve the effectiveness of Genetic Algorithms in the task of training neural networks. This new method improves the performance of genetic population chromosomes, which are randomly selected from the population. This method brings random changes to the selected chromosomes and the course of the optimization is determined by parameters, such as the temperature of the method. For high temperature values, the method accepts error values that may be higher than the initial one, in order to achieve the optimal exploration of the research space, but as the temperature decreases, the method focuses on the optimal values of the error function. The main contributions of the current work are:

1. Periodic application of an intelligent stochastic technique based on Simulated Annealing. This technique improves the training error of randomly selected chromosomes.
2. By using parameters, the changes that this stochastic method can cause in the chromosomes are controlled.
3. This stochastic technique can be used without modification in both classification and data fitting problems.

The new training method is quite general and has been successfully applied to a variety of data classification and data fitting problems. This new technique significantly improves the performance of Genetic Algorithms in almost all data sets that were used, and in fact, in several of them the reduction in the error can reach up to 80%. The proposed method achieved a significant reduction in error compared to all the techniques with which it was compared. This reduction starts on average from 20% for the case of genetic algorithms and ends in a reduction of 45% for the case of the BFGS optimization method. Furthermore, the technique's behavior and performance are not significantly affected by any variations in its critical parameters except for the F parameter, which controls the magnitude of changes that can be made to a chromosome. However, the effect of this parameter seems to decrease for large values.

However, this new technique may affect the execution time of the Genetic Algorithm as it adds a new computational part. This overhead in computational time may be reduced by using modern parallel programming techniques from recent literature [103]. Furthermore, the effect of the temperature reduction mechanism on the performance of the Simulated Annealing variant could be studied and more sophisticated minimization techniques could be tested. Also, an effort could be made to apply the new technical training to other machine learning models, as, for example, the Radial Basis Function (RBF) networks [104].

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.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

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

References

1. C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
2. G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.
3. P. Baldi, K. Cranmer, T. Faucett et al, Parameterized neural networks for high-energy physics, Eur. Phys. J. C **76**, 2016.
4. J. J. Valdas and G. Bonham-Carter, Time dependent neural network models for detecting changes of state in complex processes: Applications in earth sciences and astronomy, Neural Networks **19**, pp. 196-207, 2006
5. G. Carleo,M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, pp. 602-606, 2017.
6. Lin Shen, Jingheng Wu, and Weitao Yang, Multiscale Quantum Mechanics/Molecular Mechanics Simulations with Neural Networks, Journal of Chemical Theory and Computation **12**, pp. 4934-4946, 2016.
7. Sergei Manzhos, Richard Dawes, Tucker Carrington, Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces, Int. J. Quantum Chem. **115**, pp. 1012-1020, 2015.
8. Jennifer N. Wei, David Duvenaud, and Alán Aspuru-Guzik, Neural Networks for the Prediction of Organic Chemistry Reactions, ACS Central Science **2**, pp. 725-732, 2016.
9. Lukas Falat and Lucia Pancikova, Quantitative Modelling in Economics with Advanced Artificial Neural Networks, Procedia Economics and Finance **34**, pp. 194-201, 2015.
10. Mohammad Namazi, Ahmad Shokrolahi, Mohammad Sadeghzadeh Maharluie, Detecting and ranking cash flow risk factors via artificial neural networks technique, Journal of Business Research **69**, pp. 1801-1806, 2016.
11. G. Tkacz, Neural network forecasting of Canadian GDP growth, International Journal of Forecasting **17**, pp. 57-69, 2001.
12. Igor I. Baskin, David Winkler and Igor V. Tetko, A renaissance of neural networks in drug discovery, Expert Opinion on Drug Discovery **11**, pp. 785-795, 2016.

13. Ronadl Bartzatt, Prediction of Novel Anti-Ebola Virus Compounds Utilizing Artificial Neural Network (ANN), Chemistry Faculty Publications **49**, pp. 16-34, 2018.
14. M.B. Kia, S. Pirasteh, B. Pradhan B. et al, An artificial neural network model for flood simulation using GIS: Johor River Basin, Malaysia, Environ Earth Sci **67**, pp. 251–264, 2012.
15. A.K. Yadav, S.S. Chandel, Solar radiation prediction using Artificial Neural Network techniques: A review, Renewable and Sustainable Energy Reviews **33**, pp. 772-781, 2014.
16. M.A. Getahun, S.M. Shitote, C. Zachary, Artificial neural network based modelling approach for strength prediction of concrete incorporating agricultural and construction wastes, Construction and Building Materials **190**, pp. 517-525, 2018.
17. M. Chen, U. Challita, W. Saad, C. Yin and M. Debbah, Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial, IEEE Communications Surveys & Tutorials **21**, pp. 3039-3071, 2019.
18. K. Peta, J. Żurek, Prediction of air leakage in heat exchangers for automotive applications using artificial neural networks, In: 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, pp. 721-725, 2018.
19. I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing **72**, pp. 269-277, 2008.
20. S. Guarnieri, F. Piazza, A. Uncini, Multilayer feedforward networks with adaptive spline activation function, IEEE Transactions on Neural Networks **10**, pp. 672-683, 1999.
21. Ö.F. Ertuğrul, A novel type of activation function in artificial neural networks: Trained activation function, Neural Networks **99**, pp. 148-157, 2018.
22. A. D. Rasamoelina, F. Adjailia, P. Sinčák, A Review of Activation Function for Artificial Neural Network, In: 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, pp. 281-286, 2020.
23. D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by back-propagating errors, Nature **323**, pp. 533 - 536 , 1986.
24. T. Chen and S. Zhong, Privacy-Preserving Backpropagation Neural Network Learning, IEEE Transactions on Neural Networks **20**, , pp. 1554-1564, 2009.
25. B. M. Wilamowski, S. Iplikci, O. Kaynak and M. O. Efe, "An algorithm for fast convergence in training neural networks," IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222), Washington, DC, USA, 2001, pp. 1778-1782 vol.3, doi: 10.1109/IJCNN.2001.938431.
26. M. Riedmiller and H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm, Proc. of the IEEE Intl. Conf. on Neural Networks, San Francisco, CA, pp. 586–591, 1993.
27. T. Pajchrowski, K. Zawirski and K. Nowopolski, Neural Speed Controller Trained Online by Means of Modified RPROP Algorithm, IEEE Transactions on Industrial Informatics **11**, pp. 560-568, 2015.
28. Rinda Parama Satya Hermanto, Suharjito, Diana, Ariadi Nugroho, Waiting-Time Estimation in Bank Customer Queues using RPROP Neural Networks, Procedia Computer Science **135**, pp. 35-42, 2018.
29. B. Robitaille and B. Marcos and M. Veillette and G. Payre, Modified quasi-Newton methods for training neural networks, Computers & Chemical Engineering **20**, pp. 1133-1140, 1996.
30. Q. Liu, J. Liu, R. Sang, J. Li, T. Zhang and Q. Zhang, Fast Neural Network Training on FPGA Using Quasi-Newton Optimization Method,IEEE Transactions on Very Large Scale Integration (VLSI) Systems **26**, pp. 1575-1579, 2018.
31. C. Zhang, H. Shao and Y. Li, Particle swarm optimisation for evolving artificial neural network, IEEE International Conference on Systems, Man, and Cybernetics, , pp. 2487-2490, 2000.
32. Jianbo Yu, Shijin Wang, Lifeng Xi, Evolving artificial neural networks using an improved PSO and DPSO **71**, pp. 1054-1060, 2008.
33. J. Ilonen, J.K. Kamarainen, J. Lampinen, Differential Evolution Training Algorithm for Feed-Forward Neural Networks. Neural Processing Letters **17**, pp. 93–105, 2003.
34. Le Zhang and P.N. Suganthan, A survey of randomized algorithms for training neural networks, Information Sciences **364-365**, pp. 146-155, 2016.
35. M. Yaghini, M.M. Khoshraftar, M. Fallahi, A hybrid algorithm for artificial neural network training, Engineering Applications of Artificial Intelligence **26**, pp 293-301, 2013.
36. J.F. Chen, Q.H. Do,H.N. Hsieh, Training Artificial Neural Networks by a Hybrid PSO-CS Algorithm, Algorithms **8**, pp. 292-308, 2015.
37. X.S. Yang, S. Deb, Engineering Optimisation by Cuckoo Search, Int. J. Math. Model. Numer. Optim. **1**, 330–343, 2010.
38. I. Ivanova, M. Kubat, Initialization of neural networks by means of decision trees, Knowledge-Based Systems **8**, pp. 333-344, 1995.
39. J.Y.F. Yam, T.W.S. Chow, A weight initialization method for improving training speed in feedforward neural network, Neurocomputing **30**, pp. 219-232, 2000.
40. K. Chumachenko, A. Iosifidis, M. Gabbouj, Feedforward neural networks initialization based on discriminant learning, Neural Networks **146**, pp. 220-229, 2022.
41. M.V. Narkhede, P.P. Bartakke, M.S. Sutaone, A review on weight initialization strategies for neural networks, Artif Intell Rev **55**, pp. 291–322, 2022.
42. K-S Oh, K. Jung, GPU implementation of neural networks, Pattern Recognition **37**, pp. 1311-1314, 2004.

43. A.A. Huqqani, E.Schikuta, S. Ye, P. Chen, Multicore and GPU Parallelization of Neural Networks for Face Recognition, Procedia Computer Science **18**, pp. 349-358, 2013.
44. M. Zhang, K. Hibi, J. Inoue, GPU-accelerated artificial neural network potential for molecular dynamics simulation, Computer Physics Communications **285**, 108655, 2023.
45. Pallipuram, V.K., Bhuiyan, M. & Smith, M.C. A comparative study of GPU programming models and architectures using neural networks. J Supercomput **61**, 673–718, 2012.
46. Holland, J.H. Genetic algorithms. Sci. Am. **267**, 66–73, 1992.
47. Stender, J. Parallel Genetic Algorithms: Theory & Applications; IOS Press: Amsterdam, The Netherlands, 1993.
48. Goldberg, D. Genetic Algorithms in Search, Optimization and Machine Learning; Addison-Wesley Publishing Company: Reading, MA, USA, 1989.
49. Michaelewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs; Springer: Berlin/Heidelberg, Germany, 1996.
50. Y.H. Santana, R.M. Alonso, G.G. Nieto, L. Martens, W. Joseph, D. Plets, Indoor genetic algorithm-based 5G network planning using a machine learning model for path loss estimation, Appl. Sci. **12**, 3923. 2022.
51. X. Liu, D. Jiang, B. Tao, G. Jiang, Y. Sun, J. Kong, B. Chen, Genetic algorithm-based trajectory optimization for digital twin robots, Front. Bioeng. Biotechnol **9**, 793782, 2022.
52. K. Nonoyama, Z.Liu, T. Fujiwara, M.M. Alam, T. Nishi, Energy-efficient robot configuration and motion planning using genetic algorithm and particle swarm optimization, Energies **15**, 2074, 2022.
53. K. Liu, B. Deng, Q. Shen, J. Yang, Y. Li, Optimization based on genetic algorithms on energy conservation potential of a high speed SI engine fueled with butanol–gasoline blends, Energy Rep. **8**, pp. 69–80, 2022.
54. G. Zhou, S. Zhu, S. Luo, Location optimization of electric vehicle charging stations: Based on cost model and genetic algorithm, Energy **247**, 123437, 2022.
55. 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.
56. F. H. F. Leung, H. K. Lam, S. H. Ling, P. K. S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, IEEE Transactions on Neural Networks **14**, pp. 79-88, 2003.
57. Q. Gao, K. Qi, Y. Lei, Z. He, An Improved Genetic Algorithm and Its Application in Artificial Neural Network Training, In: 2005 5th International Conference on Information Communications & Signal Processing, Bangkok, pp. 357-360, 2005.
58. F. Ahmadizar, K. Soltanian, F. AkhlaghianTab, I. Tsoulos, Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm, Engineering Applications of Artificial Intelligence **39**, pp. 1-13, 2015.
59. A. Kobrunov, I. Priezzhev, Hybrid combination genetic algorithm and controlled gradient method to train a neural network, Geophysics **81**, pp. 35-43, 2016.
60. S. Kirkpatrick, C.D. Gelatt Jr, M.P. Vecchi, Optimization by simulated annealing, Science **220**, pp. 671-680, 1983.
61. S.J. D'Amico, S-J. Wang, R. Batta, C.M. Rump, A simulated annealing approach to police district design Computers & Operations Research **29**, pp. 667-684, 2002.
62. Y. Crama, M. Schyns, Simulated annealing for complex portfolio selection problems}, journal = {European Journal of Operational Research **150**, pp. 546-571, 2003.
63. K.M. El-Naggar, M.R. AlRashidi, M.F. AlHajri, A.K. Al-Othman, Simulated Annealing algorithm for photovoltaic parameters identification, Solar Energy **86**, pp. 266-274, 2012.
64. H. Yu, H. Fang, P. Yao, Y. Yuan, A combined genetic algorithm/simulated annealing algorithm for large scale system energy integration, Computers & Chemical Engineering **24**, pp. 2023-2035, 2000.
65. K. Ganesh, M. Punniyamoorthy, Optimization of continuous-time production planning using hybrid genetic algorithms-simulated annealing, Int J Adv Manuf Technol **26**, pp. 148–154, 2005.
66. S.-F. Hwang, R.-S. He, Improving real-parameter genetic algorithm with simulated annealing for engineering problems, Advances in Engineering Software **37**, pp. 406-418, 2006.
67. X.G. Li, X. Wei, An Improved Genetic Algorithm-Simulated Annealing Hybrid Algorithm for the Optimization of Multiple Reservoirs, Water Resour Manage **22**, pp. 1031–1049, 2008.
68. P. Suanpang, P. Jamjuntr, K. Jermsittiparsert, P. Kaewyong, Tourism Service Scheduling in Smart City Based on Hybrid Genetic Algorithm Simulated Annealing Algorithm, Sustainability **14**, 16293, 2022.
69. L. Terfloth, J. Gasteiger, Neural networks and genetic algorithms in drug design, Drug Discovery Today **6**, pp. 102-108, 2001.
70. B. Samanta, Artificial neural networks and genetic algorithms for gear fault detection, Mechanical Systems and Signal Processing **18**, pp. 1273-1282, 2004.
71. F. Yu, X. Xu, A short-term load forecasting model of natural gas based on optimized genetic algorithm and improved BP neural network, Applied Energy **134**, pp. 102-113, 2014.
72. P. Kaelo, M.M. Ali, Integrated crossover rules in real coded genetic algorithms, European Journal of Operational Research **176**, pp. 60-76, 2007.
73. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
74. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository. 2023. Available online: https://archive.ics.uci.edu (accessed on 18 February 2024).

75. 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.
76. 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.
77. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
78. T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, Machine Learning **16**, pp. 59-88, 1994.
79. 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.
80. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
81. 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.
82. 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.
83. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
84. 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.
85. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
86. 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.
87. 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.
88. 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.
89. 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.
90. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
91. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
92. R.P. Gorman, T.J. Sejnowski, Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets, Neural Networks **1**, pp. 75-89, 1988.
93. W.H. Wolberg, O.L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, Proc Natl Acad Sci U S A. **87**, pp. 9193–9196, 1990.
94. 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.
95. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
96. 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, Phys. Rev. E **64**, pp. 1-8, 2001.
97. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
98. T.F. Brooks, D.S. Pope, A.M. Marcolini, Airfoil self-noise and prediction. Technical report, NASA RP-1218, July 1989.
99. J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.
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. V. Charilogis, I.G. Tsoulos, Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions , Information **13**, 217, 2022.
102. I.G. Tsoulos, Modifications of real code genetic algorithm for global optimization, Applied Mathematics and Computation **203**, pp. 598-607, 2008.
103. A. Bevilacqua, A methodological approach to parallel simulated annealing on an SMP system. Journal of Parallel and Distributed Computing **62**, pp. 1548-1570, 2002.

104. J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation 3, pp. 246-257, 1991.