Multi-objective Training of Multilayer Perceptron Using the Generalized Differential Evolution 3 Algorithm

Wesley Marques Lima
Graduate Program on Computational Modeling and Systems
UNIMONTES-MG
Montes Claros, Brazil
wesley.marques@ufvjm.edu.br

Igor Caetano Diniz
Graduate Program on Informatics
PUC-Rio
Rio de Janeiro, Brazil
idiniz@inf.puc-rio.br

Honovan Paz Rocha
Institute of Engineering, Science and Technology
UFVJM-MG
Janaúba, Brazil
honovan.rocha@ufvjm.edu.br

***Abstract*—This paper investigates the training of Multilayer Perceptron (MLP) neural networks using evolutionary algorithms. Initially, a classical Differential Evolution (DE) algorithm is used to train the MLP with a single-objective approach that minimizes the Mean Squared Error (MSE). The training is then reformulated as a multi-objective optimization problem, using the Generalized Differential Evolution 3 (GDE3) algorithm to simultaneously minimize the MSE and the norm of the weight vector. Experiments on five benchmark binary classification datasets demonstrate that the multi-objective approach generates smoother decision boundaries and more stable models compared to single-objective training. While both approaches achieve similar average accuracy, the GDE3-based method consistently exhibits lower variance and better generalization. We argue that Pareto-based multi-objective optimization offers practical advantages for neural network training, particularly in applications where model robustness and stability are critical, even at the cost of increased running time.**

***Index Terms*—Multi-objective optimization, Differential Evolution, GDE3, neural networks, MLP, machine learning**

I. Introduction

Artificial Neural Networks (ANNs), particularly Multilayer Perceptrons (MLPs), have been widely adopted for solving classification and regression problems due to their ability to approximate complex nonlinear functions [1]. Traditionally, training MLPs involves minimizing a loss function—typically the Mean Squared Error (MSE)—through gradient-based methods such as backpropagation [2] and its variations. However, such approaches often face issues like overfitting and convergence to local minima, especially in high-dimensional and complex problem spaces [3], [4].

An alternative is to employ evolutionary algorithms, which offer global search capabilities and do not rely on gradient information. Among them, Differential Evolution (DE) has proven to be a robust and efficient population-based optimization method for continuous, nonlinear, and multimodal problems [5], [6]. In the context of neural network training, DE enables exploration of the weight space in a way that can overcome some of the limitations of traditional methods [7].

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

While single-objective training optimizes only the MSE, this may not be sufficient to guarantee good generalization performance. Excessively minimizing the training error can lead to overly complex decision boundaries and overfitting [8]. To address this, multi-objective approaches have been proposed, in which the network is trained to simultaneously optimize both the prediction error and a measure of complexity—such as the norm of the weight vector [4], [9]. This dual-objective formulation encourages simpler models with better generalization [10].

This work explores the application of a multi-objective version of Differential Evolution, known as Generalized Differential Evolution 3 (GDE3), to train MLPs [11]. Unlike common regularization techniques, which incorporate complexity control through penalty terms in the loss function, our approach formulates the problem as an explicit multi-objective optimization task. It performs an unconstrained Pareto-based search, treating both objectives—prediction error and model complexity—independently. This strategy is grounded in well-established concepts from multi-objective optimization theory [12], [13]. The experimental results, conducted on five benchmark binary classification datasets [14], demonstrate the effectiveness of this approach in producing more stable models and smoother decision boundaries.

The rest of the paper is structured as follows: Section II presents the Multi-objective optimization formulation to train Neural Networks. In Section III are described the evolutionary optimization approaches used to train neural networks. Section IV presents the methodology, including the MLP configuration, DE and GDE3 implementation details, and datasets used. Section V describes the experimental setup. Section VI discusses the results obtained and, finally, Section VII presents the conclusions and proposes directions for future work.

II. Multi-Objective Optimization Formulation of Neural Networks

We can formulate the training of an MLP as a multi-objective learning problem, where the goal is to minimize two objective functions, which are intrinsically linked to the control of the bias and variance of the network, and consequently, to the predictive risk. The work of [4] demonstrates that the norm of the weights can be used to control the variance. Thus, the search for simultaneous minimization of the mean squared error, represented by Equation 1, and the norm of the weight vector, represented by Equation 2, allows the reduction of bias and variance, resulting in a greater generalization power.

$$J_1 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1}$$

$$J_2 = \|\mathbf{w}\| = \sqrt{\sum_{i=1}^{n_w} w_i^2} \tag{2}$$

In Equation 1, y_i and $\hat{y}_i$ represent, respectively, the expected output and the actual output of the network for the $i-$th sample, while N is the total number of samples in the training set. In Equation 2, w_i denotes the $i-$th weight of the network, and n_w is the total number of weights.

This formulation has been used to solve the multi-objective learning problem by several researchers [7], [9], [15]–[19] and one of the first solutions to the optimization problem characterized by the joint minimization J_1 and J_2 is the use of the $\epsilon-$constrained approach, as presented by the MOBJ method described in 15. MOBJ stands out as one of the seminal methods for multi-objective training of neural networks. It is a pioneer in the explicit and separate optimization of objective functions. Using the $\epsilon-$constrained [20] method, MOBJ addresses the multi-objective problem through several single-objective problems, where the function J_1 is optimized with $J_2 \leq \epsilon$ as the constraint. This is evidenced in equations 3 and 4:

$$\min_{\mathbf{w}} \quad J_1 \tag{3}$$

$$s.a: \quad J_2 \leq \epsilon \tag{4}$$

To solve each constrained single-objective optimization problem, MOBJ uses the Ellipsoidal Algorithm [21]. This algorithm is a half-space exclusion technique with the ability to handle constrained nonlinear optimization problems. The adopted strategy consists of excluding areas where the solution is not found by progressively surrounding the solution with ellipsoids of decreasing size. This process is repeated until, at some point, only the desired solution remains.

III. Differential Evolution Algorithms

A. Single-objective Differential Evolution (DE)

The Differential Evolution algorithm was proposed by 22 as a simple and efficient optimization method for nonlinear problems with continuous variables. The DE is easy to use, with fast convergence and few parameters. Furthermore, it does not use derivatives, making it an interesting method for non-differentiable problems. DE algorithm is used in several applications, including the training of MLPs [19], [23], combinatorial optimization [24], among others.

The method works based on differential mutation, an operation that calculates the weighted difference between two vectors and adds it to a third vector, thus generating a new candidate solution. It is compared with a predetermined candidate solution from the population, and the candidate solution that best fits the objective function remains in the population for the next iteration. There are several variations of the differential mutation, but we will not explore them in this work.

The first step of the differential evolution algorithm is the creation of a population of candidate solutions. Each candidate solution is generated using random numbers uniformly distributed between a predefined maximum and minimum. The initial population generation can be defined as:

$$x_i^j = L_{inf}^j + Rand \cdot (L_{sup}^j - L_{inf}^j) \tag{5}$$

where $Rand$ is a random value between 0 and 1, x_i^j is the *j-th* variable of the *i-th* candidate solution, L_{inf}^j is the lower bound for the *j-th* variable, and L_{sup}^j is the upper bound for the *j-th* variable. After defining the initial population, the DE starts the mutation process. For each candidate solution in the population, a new candidate solution, v_i, is generated, which is calculated using an individual x_{r0} as the basis vector, added to the weighted difference between two individuals x_{r1} and x_{r2}, also from the population of candidate solutions. The individuals x_{r0}, x_{r1} and x_{r2} cannot be equal. Figure 1 illustrates how the mutant vector, v_i, is constructed in a two-dimensional space. Equation 6 describes how to compute the new candidate solution for iteration g:

$$v_{i,g} = x_{r0,g} + F \cdot (x_{r1,g} - x_{r2,g}) \tag{6}$$

where $F \in [0; 1]$, often called the mutation factor, determines the size of the perturbation that will be made to the basis vector x_{r0}.

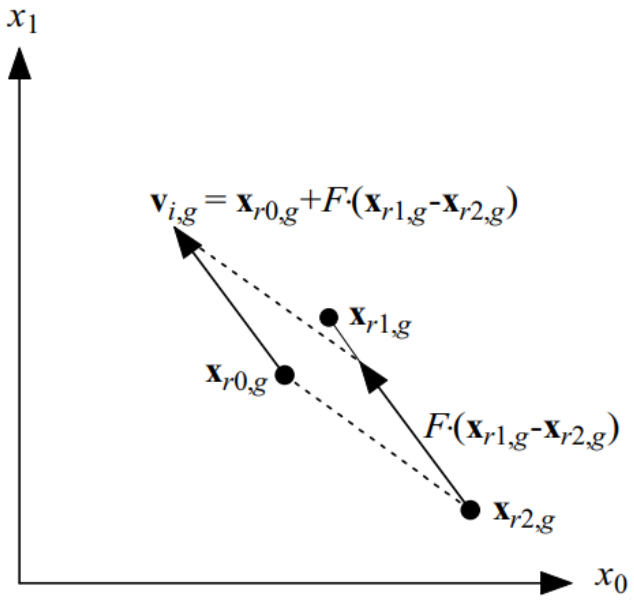
The following operation is the crossover step, at which each individual present in the set of candidate solutions undergoes a discrete recombination process with the differential vector v_i generated during the differential mutation. The formulation of this operation is given by the equation 7.

$$u_{i,g}^j = \begin{cases} v_{i,g}^j, & if \quad Rand \leq CR \quad or \quad j = \delta \\ x_{i,g}^j, & otherwise \quad otherwise \end{cases} \tag{7}$$

where $CR \in [0; 1]$ is the crossover probability and $\delta \in [1; n]$ is a random index to ensure that at least one variable of the individual is inherited from the mutant solution.

Finally, the selection process is carried out, at which the individual $x_{i,g}$ is compared with the newly generated individual, $u_{i,g}$. The permanence in the population for the next iteration is determined by the individual that best fits the objective function. Algorithm 1 shows the steps of DE.

Fig. 1: Representation of differential mutation in two dimensions 6



Algorithm 1: Differential Evolution Pseudocode

Input: CR=0.9, F=0.7, Population size(NP)=100,
Maximum number of iterations(GEN)=350
Output: Optimal Solution
$P_0 \longleftarrow$ Initial Population
while $G <$ *GEN* **do**
  $f \longleftarrow$ value of f for each x_i in the population
  i=0
  while $i <$ *NP* **do**
    Create u_i from equations 6 and 7
    if $f(u_i)$ *outperforms* $f(x_i)$ **then**
      $P_{G+1} \longleftarrow u_i$
    else
      $P_{G+1} \longleftarrow x_i$
    end
  end
end
Solution $\longleftarrow$**Solution** $x \in [P_G]$ **with best** $f(x)$

B. Generalized Differential Evolution 3

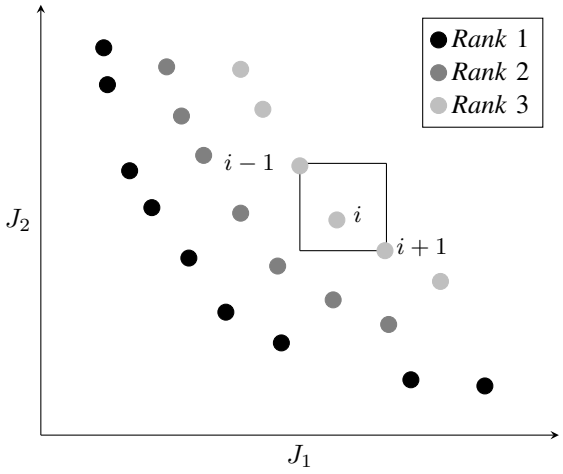
The Generalized Differential Evolution 3 (*The Third evolution step of Generalized Differential Evolution* - GDE3) algorithm was proposed by 11 as an extension of DE for optimizing multi-objective problems, allowing it to deal with an arbitrary number of objectives and constraints. GDE3 preserves the main DE operations, such as differential mutation and crossover, with minimal changes in relation to the original version.

One of the changes in GDE3 in relation to DE is in the selection operation, since there is now more than one objective to be considered. Thus, after the creation of the new individual $u_{i,g}$, it will be compared with $x_{i,g}$, but now the evaluation is made based on more than one objective, so a solution is considered superior to another only if it surpasses it in all objectives, that is, if it dominates the other. If both are non-dominated, that is, neither of them is surpassed in all objectives, both are stored in a set R_g of candidate solutions. The best candidate solutions POP (where POP represents the population size) of R_g will be preserved for the next iteration.

To identify the best POP solutions of R_g, two techniques are applied. First, a Pareto Sorting is performed, where the set R_g gives rise to several subsets or Pareto fronts. A representation of the Pareto fronts can be seen in Figure 2, in which the best Pareto front is called *Rank* 1, while the worst is identified as *Rank* 3. In the example presented, the subset of solutions *Rank* 1 is formed by selecting all non-dominated solutions of R_g. In turn, the subsets *Rank* 2 and *Rank* 3 are, respectively, the non-dominated solutions of R_g − *Rank* 1 and the non-dominated solutions of R_g − (*Rank* 1 ∪ *Rank* 2).

Then, the process of excluding the least desirable solutions present in R_g begins. This process starts from the worst Pareto front, and the solution to be eliminated is the one with the lowest *Crowding Distance* (CD) [25] value. Each time a solution is removed, the CD values are recalculated. The procedure is repeated until a set with POP solutions remains; these will compose the population for the next iteration. The steps of GDE3 are presented in Algorithm 2.

Fig. 2: Representation of Pareto Ordering and CD Calculation



Finally, upon reaching the stopping condition, which in this case is the number of iterations, the set of non-dominated solutions present in the population is returned as the final solution. This set represents the solutions that make up the Pareto frontier.

IV. Methodology

This section describes the implemented models, the datasets used for evaluation, and the experimental procedures adopted to assess model performance.

Algorithm 2: Pseudocode GDE3

Input: CR,F, NP, GEN
Output: Pareto Optimal Set
$P_0 \longleftarrow$ Initial Population
while $G <$ *GEN* **do**
  $f_1, f_2 \longleftarrow$ value of f_1, f_2 for each x_i in the population
  $i = 0$
  while $i < NP$ **do**
    Create u_i from equations 6 and 7
    if u_i *dominates* x_i **then**
      $R_G \longleftarrow u_i$
    else
      if x_i *dominates* u_i **then**
        $R_G \longleftarrow x_i$
      else
        $R_G \longleftarrow x_i, u_i$
      end
    end
  end
  if $R_G > NP$ **then**
    Classify R_G into **d** Pareto fronts
    $P_{G+1} \longleftarrow \emptyset$
    $\gamma \longleftarrow 1$
    while *Number of solutions in* $P_{G+1} < NP$ **do**
      $P_{G+1} \longleftarrow P_{G+1} \cup$ Front candidate solutions γ
      $\gamma \longleftarrow \gamma + 1$
    end
    while *Number of solutions in* $P_{G+1} > NP$ **do**
      Calculate CD for all solutions in the front $\gamma - 1$
      Exclude from P_{G+1} the candidate solution in the front $\gamma - 1$ with the smallest CD value
    end
  else
    $P_{G+1} \longleftarrow R_G$
  end
end
Pareto Estimate $\longleftarrow$ **Non-dominated solutions of** R_G

A. Implemented Models

An MLP neural network was implemented using a single hidden layer composed of 50 neurons and sigmoid activation functions in all layers. The sigmoid function at the output layer allows interpreting the output as the probability that a given input belongs to a class [1]. The network was trained using the Differential Evolution (DE) algorithm [5], [6].

Two variants of DE were used: (i) a classical single-objective DE minimizing the Mean Squared Error (MSE), and (ii) a multi-objective version known as GDE3 [11], which simultaneously minimizes the MSE and the norm of the weight vector [4], [9]. This approach aims to balance accuracy and model complexity to avoid overfitting [8], [10].

The modified mutation strategy for the DE algorithm is shown in Equation (8):

$$v_{i,g} = x_i + \lambda \cdot (x_{best} - x_i) + F \cdot (x_{r2} - x_{r3}) \tag{8}$$

where x_i is the i-th individual in the population, x_{best} is the best individual found so far, and F and λ are control parameters. Each individual encodes a set of neural network weights.

B. Datasets

Five binary classification datasets were used in the experiments:

TABLE I: Summary of Datasets Used

Dataset	Samples	Features
Two Circles	500	2
Two Moons	500	2
Banana	5300	2
Breast Cancer (WBC)	569	31
Diabetes	768	8

Two synthetic datasets (Two Circles and Two Moons) were chosen for their low dimensionality, allowing for visual inspection of decision boundaries. The Banana dataset is a real-world problem with overlapping classes. The WBC dataset provides a high-dimensional but relatively easy classification problem, while the Diabetes dataset is more challenging and widely used for benchmarking [14].

All datasets were normalized using Z-score normalization, as shown in Equation (9):

$$Z = \frac{x - \mu}{\sigma} \tag{9}$$

The selection of datasets was motivated by the desire to evaluate the models under varied complexity conditions. Synthetic datasets like Two Circles and Two Moons allow for direct visualization of decision boundaries, serving as intuitive benchmarks for generalization analysis. Banana represents a real-world scenario with overlapping classes in low dimensions, stressing robustness in boundary shaping. Breast Cancer offers a high-dimensional yet well-structured dataset for assessing performance in clean clinical data, while Diabetes provides a challenging case of noisy, moderate-dimensional data with class imbalance. This diversity ensures that both linear separability and real-world generalization issues are explored, making the evaluation of the GDE3 approach more comprehensive and realistic.

C. Training and Validation Procedure

Both training approaches (single and multi-objective) were evaluated using K-Fold cross-validation [8]. The datasets were split into 5 folds for testing (20% test split), and the training folds were further divided into 10 folds for internal validation.

In the multi-objective case, one Pareto-optimal solution was selected based on the best average accuracy on the validation folds. The selected solution was then tested on the test fold, and performance was measured using MSE, accuracy, and execution time.

D. Experimental Setup

All algorithms were implemented in Python. The Banana and Diabetes datasets were processed on a local machine (Intel Core i5-7200U, 8GB RAM, Ubuntu 22.04). The remaining datasets were processed on Google Colab (Intel Xeon CPU 2.20GHz, 12GB RAM).

The DE and GDE3 algorithms used the same parameters: $N_p = 100$, $N_{gen} = 350$, $F = 0.7$, $\lambda = 0.99$, and $CR = 0.9$. Initial weights were randomly generated in the range $[-0.5, 0.5]$ as per Equation (10):

$$w_i \sim \mathcal{U}(-0.5, 0.5) \tag{10}$$

This configuration was chosen based on preliminary tests and literature recommendations [9], [12].

V. Results and Discussion

This section presents the results obtained from training a Multilayer Perceptron (MLP) using two different strategies: single-objective optimization (minimizing only the Mean Squared Error – MSE) and multi-objective optimization (minimizing both MSE and the norm of the weight vector). Results include accuracy and execution time for five binary classification datasets, as well as visualizations of decision boundaries and Pareto fronts.

A. Experimental Setup

Five binary classification datasets were used [14]. Tables II and III summarize the results for the multi-objective and single-objective training approaches, respectively. For each method and dataset, we report the mean, minimum, and maximum accuracy, standard deviation, and average execution time (in seconds) over five independent runs.

TABLE II: Accuracy and execution time for multi-objective training

Dataset	Mean	Min	Max	Std Dev	Time (s)
Two Circles	0.9960	0.9900	1.0000	0.0048	19279.31
Two Moons	0.9920	0.9700	1.0000	0.0116	19153.61
Banana	0.8950	0.8858	0.9094	0.0081	41694.62
WBC	0.9701	0.9649	0.9736	0.0042	64314.14
Diabetes	0.7695	0.7337	0.8235	0.0306	9355.70

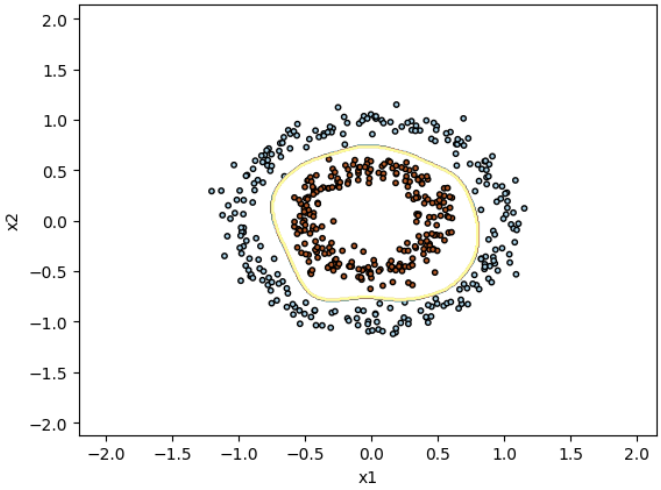
TABLE III: Accuracy and execution time for single-objective training

Dataset	Mean	Min	Max	Std Dev	Time (s)
Two Circles	0.9840	0.9400	1.0000	0.0224	10252.75
Two Moons	0.9940	0.9900	1.0000	0.0048	6979.39
Banana	0.8967	0.8801	0.9179	0.0140	23216.76
WBC	0.9735	0.9469	1.0000	0.0201	43591.54
Diabetes	0.7551	0.6470	0.8104	0.0564	3725.42

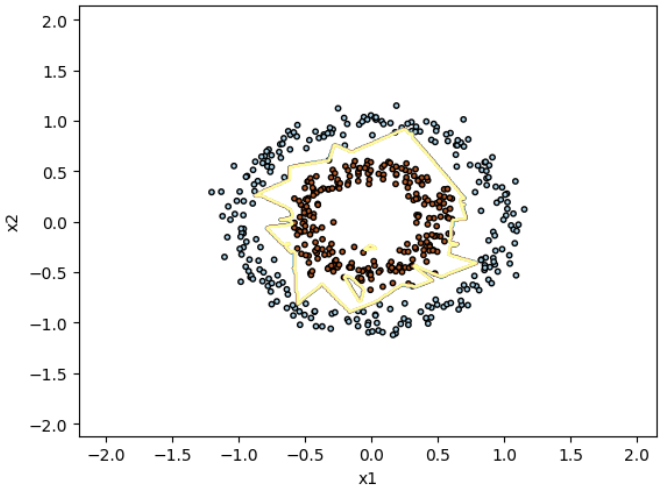
B. Decision Boundary Visualization

Figures 3, 4, and 5 illustrate the decision boundaries produced by the two training approaches on 2D datasets. The multi-objective approach generally yields smoother and more regular boundaries, which indicates better generalization [3], [4]. In binary classification, decision boundaries represent the hypersurfaces that separate classes in feature space. Their shape and complexity directly affect the model's generalization ability.

Fig. 3: Decision boundary – *Two Circles* dataset



(a) GDE3 (multi-objective)

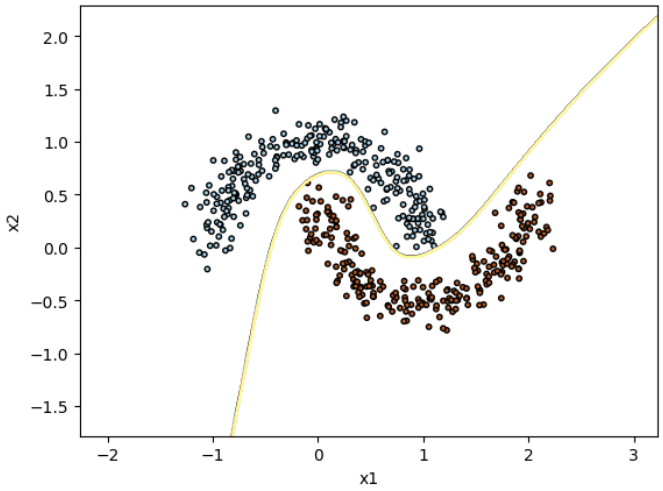


(b) DE (single-objective)

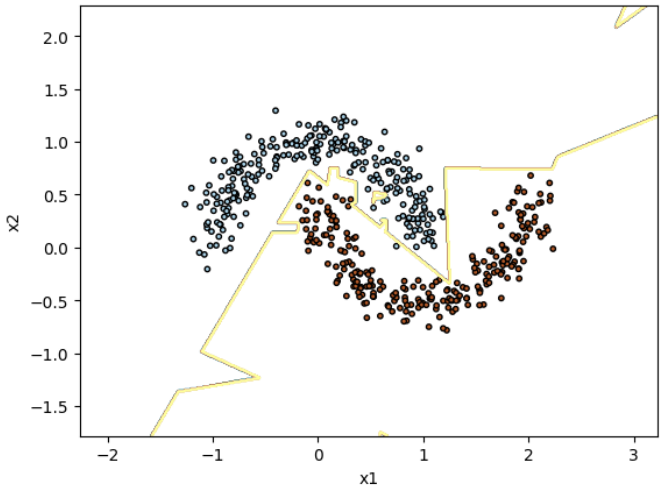
C. Pareto Front Analysis

Figures 6 to 10 show the estimated Pareto fronts generated by the GDE3 algorithm for each dataset. The fronts are well-distributed and show a large number of solutions, as no restrictions were imposed on objectives or population size [11], [12], [26].

Fig. 4: Decision boundary – *Two Moons* dataset

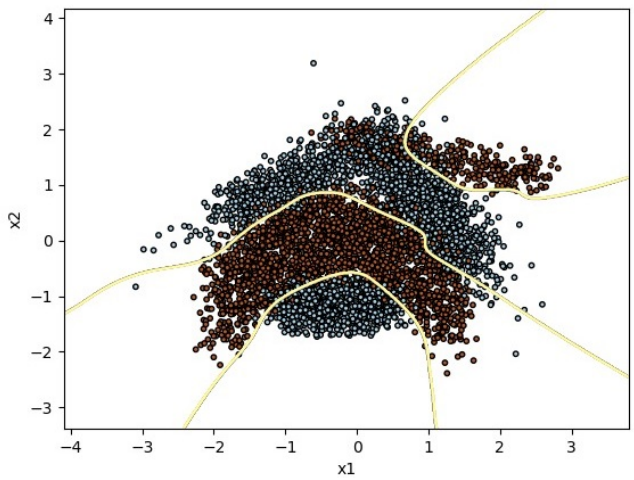


(a) GDE3 (multi-objective)

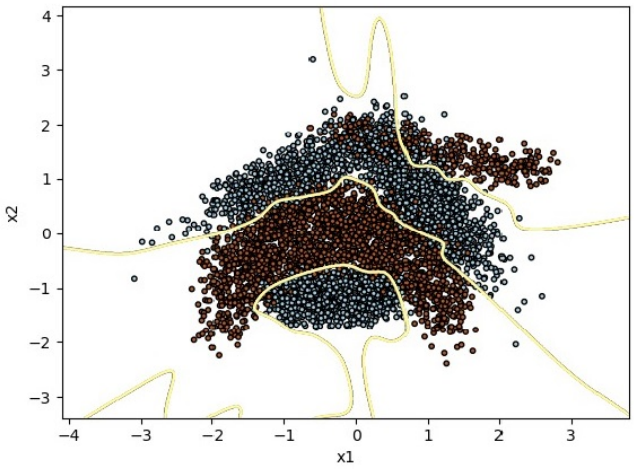


(b) DE (single-objective)

Fig. 5: Decision boundary – Banana dataset



(a) GDE3 (multi-objective)



(b) DE (single-objective)

D. Discussion

For synthetic datasets (*Two Circles* and *Two Moons*), both training strategies showed similar accuracy. However, decision boundaries generated by GDE3 are smoother and exhibit less overfitting, which is visually evident in the figures. This behavior aligns with findings in the literature that relate weight norm minimization to improved generalization [3], [4].

For the Banana dataset, accuracy was nearly the same for both methods. Still, the GDE3 approach provided lower standard deviation, indicating more stable performance.

In the *Wisconsin Breast Cancer* dataset, GDE3 again yielded better stability, with a standard deviation 0.0159 lower than that of DE. For the Diabetes dataset, GDE3 surpassed DE in both average accuracy and stability, consistent with the hypothesis that multi-objective optimization fosters more robust models [9], [12].

Execution time was higher for GDE3, which is expected due to the algorithm's multi-objective nature. Even so, the benefits in generalization may outweigh the cost in time [6], [11].

Although this study focuses on Differential Evolution and its multi-objective variant, several other optimization techniques have been applied to MLP training, such as Particle Swarm Optimization (PSO), Genetic Algorithms (GA), and more recently, Estimation of Distribution Algorithms (EDA). Compared to these methods, DE stands out for its simplicity, fewer hyperparameters, and consistent convergence in continuous spaces [6], [11]. However, future work could include empirical comparisons between GDE3 and other multi-objective algorithms, such as NSGA-II or MOEA/D, to provide a broader understanding of trade-offs in convergence speed, solution diversity, and computational cost.

The main drawback of the GDE3-based approach lies in its higher computational cost. The multi-objective formulation demands additional comparisons between individuals and the maintenance of Pareto fronts, which considerably increases the

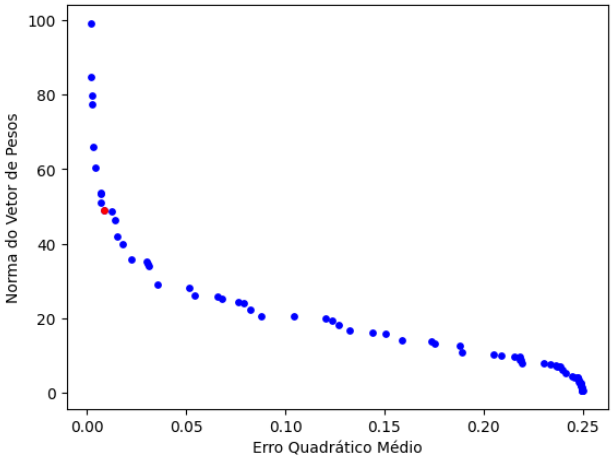


Fig. 6: Pareto front – *Two Circles*

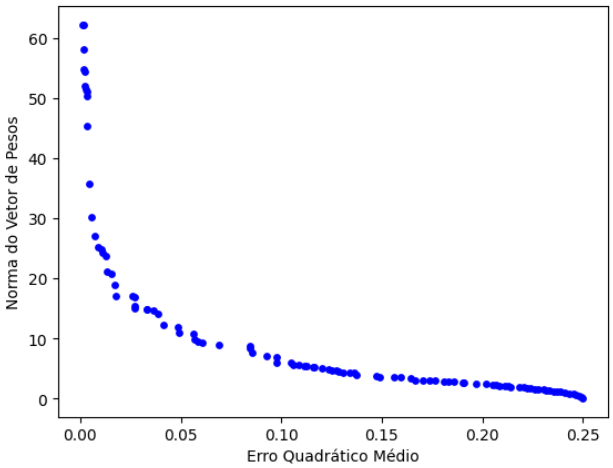


Fig. 7: Pareto front – *Two Moons*

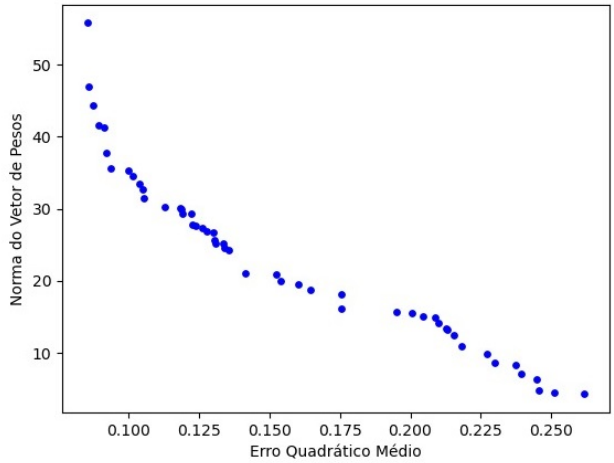


Fig. 8: Pareto front – Banana

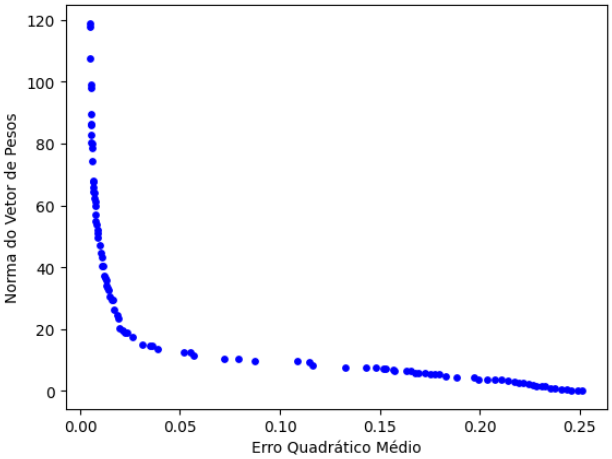


Fig. 9: Pareto front – *Wisconsin Breast Cancer*

execution time compared to single-objective DE. This limitation may hinder its direct application in real-time systems or scenarios where training time is critical. However, the stability gains, particularly in standard deviation and smoother decision surfaces, are relevant in domains where robustness outweighs computational cost — such as in biomedical diagnostics or fault detection. In such cases, the improved generalization can reduce deployment risks and improve model longevity.

Overall, the results show that multi-objective training using GDE3 improves stability and generalization while producing a diverse set of solutions.

VI. Conclusion and Future Work

This work investigated the training of Multilayer Perceptron (MLP) neural networks using two evolutionary optimization approaches. Initially, Differential Evolution (DE) was applied to minimize the Mean Squared Error (MSE) on training data [5]. Then, a multi-objective formulation was employed, using the Generalized Differential Evolution 3 (GDE3) algorithm to simultaneously minimize both the MSE and the norm of the weight vector [11].

Unlike common literature approaches that convert multi-objective problems into constrained single-objective ones [27], the proposed method performs an unconstrained search over the Pareto front [12], [26]. In doing so, it preserves the diversity and trade-offs between objectives throughout the training process.

Experimental results across five benchmark datasets demonstrated that the multi-objective approach not only provides comparable average accuracy to the single-objective method but also leads to more stable models, as evidenced by lower standard deviations. Visualizations of decision boundaries showed that GDE3 produces smoother separations, suggesting better generalization and reduced overfitting [4].

Running time was indeed longer for GDE3, due to its more complex objective evaluation and the need to maintain a Pareto front. However, the performance gains in terms of robustness and boundary smoothness justify this additional cost in many scenarios.

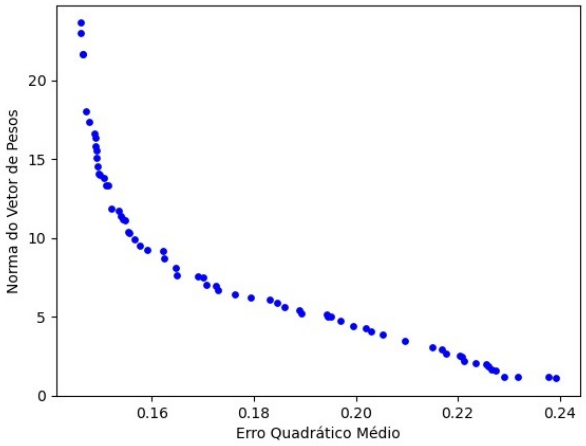


Fig. 10: Pareto front – Diabetes

A. Limitations

This study does not compare GDE3 with other multi-objective evolutionary algorithms (e.g., NSGA-II, SPEA2), nor does it explore the effect of changing population size or mutation strategies. Additionally, only one hidden layer was used in the MLP architecture, which limits conclusions about scalability to deeper networks. Despite these limitations, the findings provide a solid foundation for further investigations.

B. Future Work

Several directions may enhance this study:

- **Hyperparameter adaptation:** Future research could explore dynamic adjustment of the hyperparameters, such as differential mutation factor (F), where it could starts with a higher value and gradually decreasing it during optimization. This strategy may improve convergence and lead to more generalized networks.
- **Improved decision-making:** The final model was selected using simple cross-validation. More sophisticated decision-makers could be investigated to better exploit the Pareto set and select solutions with superior generalization performance.
- **Parallelization:** Implementing parallel versions of DE and GDE3 could allow the use of larger populations without a proportional increase in running time. This may result in better exploration of the solution space, especially for high-dimensional problems.
- **Pareto front quality analysis:** Assessing the quality of the generated Pareto fronts using appropriate metrics may provide deeper insight into the method's effectiveness and guide future improvements.

In summary, the GDE3-based multi-objective training strategy showed promising results for MLP optimization and opens up several opportunities for further enhancement and application to other neural architectures or problem domains.

REFERENCES

[1] S. Haykin, *Redes neurais: princípios e prática*. Bookman Editora, 2001.

[2] G. E. Hinton, "Connectionist learning procedures," in *Machine learning*. Elsevier, 1990, pp. 555–610.

[3] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[4] P. Bartlett, "For valid generalization the size of the weights is more important than the size of the network," *Advances in neural information processing systems*, vol. 9, 1996.

[5] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341–359, 1997.

[6] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[7] M. A. Costa, A. P. Braga, B. R. Menezes, R. A. Teixeira, and G. G. Parma, "Training neural networks with a multi-objective sliding mode control algorithm," *Neurocomputing*, vol. 51, pp. 467–473, 2003.

[8] S.-i. Amari, N. Murata, K.-R. Müller, M. Finke, and H. Yang, "Statistical theory of overtraining-is cross-validation asymptotically effective?" *Advances in neural information processing systems*, vol. 8, 1995.

[9] H. P. Rocha, M. A. Costa, and A. P. Braga, "Neural networks multiobjective learning with spherical representation of weights," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4761–4775, 2020.

[10] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 1999.

[11] S. Kukkonen and J. Lampinen, "Gde3: the third evolution step of generalized differential evolution," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1, 2005, pp. 443–450 Vol.1.

[12] C. A. C. Coello, *Evolutionary algorithms for solving multi-objective problems*, 2nd ed. Urbana, Illinois, USA: Springer, 2007, the Multi-objective Optimization Problem: p. 5–20.

[13] V. Chankong and Y. Y. Haimes, *Multiobjective decision making: theory and methodology*. Courier Dover Publications, 2008.

[14] C. L. Blake and C. J. Merz, "Uci repository of machine learning databases," *http://www. ics. uci. edu/˜ mlearn/MLRepository. html*, 1998.

[15] R. A. Teixeira, A. P. Braga, R. H. C. Takahashi, and R. R. Saldanha, "Improving generalization of MLPs with multi-objective optimization," *Neurocomputing*, vol. 35, no. 1–4, pp. 189–194, 2000.

[16] ——, "Recent advances in the MOBJ algorithm for training artificial neural networks." *International Journal of Neural Systems*, vol. 11, no. 03, pp. 265–270, 2001.

[17] M. A. Costa, A. P. Braga, and B. R. Menezes, "Improving generalization of MLPs with sliding mode control and the Levenberg-Marquardt algorithm," *Neurocomputing*, vol. 70, no. 7–9, pp. 1342–1347, 2007.

[18] ——, "Convergence analysis of sliding mode trajectories in multi-objective neural networks learning," *Neural Networks*, vol. 33, pp. 21–31, 2012.

[19] H. P. Rocha, M. A. Costa, and A. P. Braga, "Training Multi-Layer Perceptron with Multi-Objective Optimization and Spherical Weights Representation," in *Proceedings of the 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN*. Bruges, Belgium: Ciaco, 2015, pp. 131–136.

[20] V. Chankong and Y. Y. Haimes, "Multiobjective decision making: Theory and," *Methodology. New*, 1983.

[21] N. Z. Shor, "Cut-off method with space extension in convex programming problems," *Cybernetics*, vol. 13, no. 1, pp. 94–96, 1977.

[22] R. Storn and K. Price, "Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. international computer science institute, berkeley," *Berkeley, CA*, vol. 1995, 1995.

[23] R. C. T. De Souza, "Previsão de séries temporais utilizando rede neural treinada por filtro de kalman e evolução diferencial," 2008.

[24] A. L. M. Silva, "Algoritmo baseado em evolução diferencial para solução de problemas de otimização combinatória," 2014.

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[26] V. Pareto, *Cours d'économie politique*. Librairie Droz, 1964, vol. 1.

[27] C.-S. Chang, D. Xu, and H. Quek, "Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system," *IEE Proceedings-Electric Power Applications*, vol. 146, no. 5, pp. 577–583, 1999.