Enhancing Differential Evolution for Neural Network Optimization through Boundary Individual Consideration

Mandar Angchekar

Dept of EECS, Syracuse University, Syracuse, NY 13244, mangchek@syr.edu

***Abstract*—This paper presents a novel approach to Differential Evolution (DE) for optimizing neural networks by incorporating boundary individual consideration during the mutation process. Standard DE often encounters exploration challenges at the search space boundaries, potentially hindering the discovery of optimal solutions. To address this, our enhanced DE algorithm strategically includes boundary individuals to ensure a thorough exploration of the boundary regions. We applied the improved algorithm to obesity level, stroke, and fetal health prediction datasets and conducted a series of experiments comparing the enhanced DE against traditional DE technique. The results demonstrated that the enhanced DE achieved superior performance, with more consistent and higher fitness scores across generations, as well as improved precision and recall in several classification categories. These findings support the hypothesis that boundary exploration can significantly benefit the hyperparameter optimization of neural networks.**

Keywords—Differential Evolution, Neural Network Optimization, Boundary Individual Consideration, Hyperparameter Tuning, Mutation Strategies, Evolutionary Algorithms, Machine Learning, Population-Based Algorithms, Classification Accuracy, Fitness Evolution

I. Introduction

In recent years, neural networks have emerged as powerful tools for a wide range of machine learning tasks. However, their success is heavily contingent upon the careful selection of architecture and hyperparameters. Differential Evolution (DE), a population-based optimization algorithm, has shown promise in effectively navigating the complex and multi-modal landscapes typical of neural network hyperparameter spaces. Yet, a notable limitation of traditional DE is its potential inefficiency in exploring the boundary regions of the search space, which may contain optimal or near-optimal solutions.

This research aims to improve the exploration capabilities of DE by introducing a mechanism that explicitly accounts for boundary individuals during the mutation step, hypothesizing that this will yield more robust exploration and, consequently, superior optimization results. The proposed algorithm's performance was evaluated by optimizing the hyperparameters of a neural network designed to classify individuals based on obesity levels, stroke, and fetal health using comprehensive datasets. Through a comparative analysis against conventional DE methods, the results illustrate the benefits of the proposed approach in finding more optimal neural network architectures, as evidenced by enhanced fitness evolution profiles and classification metrics.

This paper is organized as follows: Section II reviews the related work in the field, Section III details the methodology, followed by Section IV, which describes the datasets and experimental setup. Section V presents the results, Section VI discusses the conclusion, implications and future work.

II. Background/ Related Work

The performance of neural networks is highly sensitive to their hyperparameter settings, which include the architecture configuration—such as the number of layers and neurons per layer—as well as learning parameters like the learning rate and regularization factors. Manual tuning of these hyperparameters is a laborious and inefficient process, necessitating the use of optimization algorithms that can automate and systematize the search for optimal configurations.

Differential Evolution (DE) is a type of evolutionary algorithm particularly suited for continuous optimization problems. It has been widely applied in various domains, including the optimization of neural network hyperparameters. The traditional DE algorithm, introduced by Storn and Price (1997), involves mutation, crossover, and selection operators to evolve the candidate solutions towards an optimum. However, DE can suffer from limited exploration near the boundaries of the search space, potentially missing out on high-performing solutions.

Recent advancements in DE have introduced several strategies for improving exploration and exploitation, such as adaptive mutation strategies, hybridization with other optimization techniques, and population diversity maintenance mechanisms. However, explicit consideration of boundary individuals has not been extensively studied within the DE framework, particularly in the context of neural network hyperparameter optimization. Studies like those by Liang et al. (2021) discuss boundary protection in evolutionary algorithms, highlighting the potential benefits of this approach in diverse optimization scenarios.

In addition to the adaptations in DE, research on other evolutionary algorithms also reveals the importance of boundary handling. For example, Burczynski and Beluch (2001) [5] explored the use of boundary elements in evolutionary algorithms for structural analysis, providing valuable insights into how boundaries can influence the performance and results of evolutionary processes. Similarly, the work by Siew and Tanyimboh (2012) [8] on water distribution systems optimization introduced a penalty-free feasibility boundary convergent algorithm that further underscores the significance of effectively managing boundary conditions in optimization tasks.

Building on these insights, our work introduces an enhanced DE algorithm that explicitly incorporates boundary individuals during the mutation process, hypothesizing that such inclusion will promote a more thorough exploration of boundary regions and thus lead to more optimal neural network configurations. This approach is expected to mitigate the often-overlooked edge solutions, potentially enhancing both the precision and recall in complex classification tasks such as stroke, obesity, and fetal health prediction.

The following section will introduce our methodology, which describes the proposed enhancements to the DE algorithm and the rationale behind incorporating boundary individuals in the mutation process.

III. Methodology

The methodology section outlines the enhanced Differential Evolution (DE) algorithm used to address the optimization of neural network hyperparameters. Our approach modifies the mutation strategy of DE by integrating boundary individuals to foster exploration at the edges of the parameter space.

A. Differential Evolution Algorithm

Differential Evolution is an evolutionary algorithm that optimizes a problem by iteratively trying to improve a candidate solution with respect to a given measure of quality or fitness. DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. This process, which includes mutation, crossover, and selection, continues until a stopping criterion is met, typically a set number of generations or a satisfactory fitness level.

B. Boundary Individual Consideration

The core modification to the standard DE algorithm is the incorporation of boundary individuals in the mutation process. A boundary individual is one that lies at the edge of the current population in the hyperparameter space. By including such individuals, the algorithm is encouraged to explore parameter combinations that might otherwise be neglected. This approach is especially beneficial in high-dimensional search spaces, where the boundaries might hold the key to improved performance.

C. Mutation Strategy Enhancement

In the enhanced Differential Evolution (DE) algorithm, boundary individuals are identified using a novel approach. Firstly, each individual, including a reference individual 'a', is normalized relative to the established bounds, converting the search space into a unit hypercube. The identification process involves calculating two types of distances: the distance to the nearest boundary for each parameter, and the Euclidean distance from each individual to the reference individual 'a'. These distances are then aggregated to form a composite measure, with the individual possessing the smallest combined distance being selected as the boundary individual. This ensures that the chosen boundary individual is proximal to the edges of the search space while maintaining relevance to the current population distribution.

The mutation strategy of the algorithm incorporates a probabilistic mechanism to decide whether to include a boundary individual in the mutation process. Depending on the set probability, which varies (e.g., 1.0, 0.8 for high probability to 0.5, 0.2 for lower probabilities), the algorithm either emphasizes exploration near the search space edges or balances between exploration and exploitation. During each generation, for each individual in the population, a random decision is made based on this probability to determine if the mutation step will involve a boundary individual. If affirmative, the find_boundary_individual function is invoked, and the mutation process uses this boundary individual to generate a new candidate solution. Otherwise, the mutation continues with the traditional approach of using differences between randomly selected individuals. This adaptive approach ensures a dynamic exploration of the search space, enhancing the algorithm's capability to discover superior solutions by considering not only central regions but also the often-overlooked boundaries.

E.g.: decide whether to use boundary individual in mutation

```
if np.random.rand() < 0.2:
    boundary_individual=find_boundary_individual(pop, a, bounds)
    mutant=np.clip(a + F * (boundary_individual + b - c), 0, 1)
else:
    mutant=np.clip(a + F * (b - c), 0, 1)
```

D. Implementation Details

The enhanced DE algorithm was implemented as follows:

1. Initialization: A population of potential neural network architectures is randomly initialized within the predefined bounds of each hyperparameter.
2. Mutation: For each individual, decide probabilistically whether to use a boundary individual modified mutation in the mutation process.
3. Crossover and Selection: Perform crossover and selection as per the standard DE algorithm. These operations are essential components of DE, facilitating the combination and refinement of candidate solutions towards optimal configurations, as originally detailed by Storn and Price (1997) [3].
4. Fitness Evaluation: Assess each individual's fitness by training a neural network with its architecture and evaluating performance on a testing set.
5. Iteration: Repeat the mutation, crossover, selection, and fitness evaluation steps until the maximum number

of generations is reached or an acceptable fitness level is achieved.

E. Hyperparameters of DE

The DE hyperparameters, including the population size, mutation factor (F), and crossover rate (CR), were determined through preliminary experimentation. For this study, a population size of 100, a mutation factor of 0.1, and a crossover rate of 0.7 for 10 generations were employed.

F. Bounds of DE

For our study, optimizing both neurons and layers, the number of layers were set between 1 and 5 and for the number of neurons in each layer, the bounds were set between 5 and 500. This setup enabled the exploration of network architectures from shallow to deep, providing a comprehensive view of potential solutions within the defined parameter space.

G. Evaluation Criteria

The performance of the neural network architectures derived from the enhanced DE algorithm was measured using the generations vs fitness plot, where the fitness was considered to be the recall score and using confusion matrices for both training and testing set.

IV. Experimental setup/ data description

To evaluate the performance of the enhanced Differential Evolution (DE) algorithm, we conducted experiments on a well-known obesity level [20], stroke [21], and fetal health [22] prediction datasets. This section details the dataset characteristics and the setup for our experiments.

A. Dataset Description

Obesity level prediction (Dataset 1): It comprises several anthropometric and lifestyle attributes collected from individuals of varying obesity levels. The dataset contains both continuous and categorical variables, including age, height, weight, and eating habits. It encompasses a range of 2,111 instances, each labeled with one of seven obesity levels from underweight to type III obesity.

Stroke prediction (Dataset 2): It captures detailed medical, demographic, and lifestyle data from 5,110 individuals. It contains 12 attributes including age, gender, hypertension, heart disease, marital status, work type, residence type, average glucose level, BMI, smoking status, and stroke occurrence. This dataset is essential for analyzing stroke risk factors and developing predictive health models.

Fetal health prediction (Dataset 3): It comprises 2,126 entries from fetal cardiotocograms (CTGs), featuring 22 continuous variables that measure physiological indicators like fetal heart rate, decelerations, and uterine contractions. It also includes histogram data on heart rate variability. The dataset's output variable, fetal_health, classifies fetal conditions into three categories: Normal, Suspect, and Pathological, aiding in the development of predictive models for fetal well-being.

B. Preprocessing

Prior to training, the datasets underwent standard preprocessing steps:

1. Continuous features were normalized to have zero mean and unit variance.
2. Categorical features were encoded using one-hot encoding to convert them into a machine-readable form.
3. The dataset was split into a training set and a testing set, with stratification to maintain the proportion of classes consistent across splits.

C. Data Complexity Enhancement

To thoroughly assess the robustness of the enhanced DE algorithm, the dataset was modified to increase its complexity before optimization. This was achieved through the following steps:

1. **Imbalance Introduction**:

The datasets was manipulated to create class imbalances by augmenting a specific classes ('Overweight_Level_II', 'Stroke', 'Normal') to overrepresent it, reflecting real-world scenarios where some conditions or categories might be more prevalent than others.

2. **Noise Injection**:

Gaussian noise was added to the continuous features of the augmented class data to simulate measurement errors and inconsistencies in data collection. This step tests the algorithm's ability to discern underlying patterns amidst data variability.

3. **Data Shuffling and Random Relabeling**:

The training data was shuffled, and 5% of the data from each class was randomly relabeled with different classes. This introduces label noise, creating a more challenging scenario for the algorithm to correctly classify the data, thereby testing its resilience and robustness.

These modifications were designed to create a more challenging optimization task, thereby providing a stringent test of the enhanced DE algorithm's ability to navigate complex fitness landscapes.

D. Neural Network Architecture

The neural network used for classification was a fully connected feedforward network. The architecture—specifically the number of hidden layers and the number of neurons in each layer—was determined by the DE optimization process.

E. DE Experimental Setup

The enhanced DE was applied to optimize the neural network architecture. Each individual in the DE population represented a potential architecture with different configurations of layers and neurons. The DE ran for a maximum of 10 generations, with the fitness of each individual assessed based on classification performance on a separate testing set.

F. Evaluation Metrics

The effectiveness of the optimized neural network was evaluated using the following metrics:

1. Accuracy: The proportion of true results among the total number of cases examined.
2. Precision: The ratio of true positive observations to the total predicted positives for each class.
3. Recall: The ratio of true positive observations to all actual positives for each class.
4. F1 Score: The harmonic mean of precision and recall, providing a balance between the two in cases of class imbalance.

G. Baseline Comparison

To gauge the improvement offered by the enhanced DE, a comparison was conducted using a Neural Network and a standard DE algorithm. This approach provided a benchmark for evaluating the benefits of including boundary individuals in the mutation strategy.

H. Computational Resources

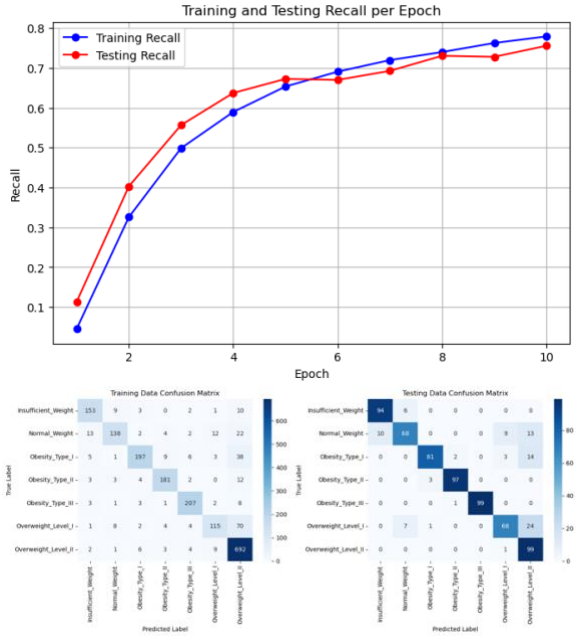
Experiments were conducted on a local machine to concurrently train multiple neural network architectures, which required significant computational resources. Specifically, the training involved 10 generations and 100 individuals across various strategies, underscoring the extensive processing capabilities needed for such tasks on a personal computer.

V. Results

This section presents a comparative analysis of a Shallow Neural Network, Differential Evolution (DE) algorithm, and configuration of the enhanced Differential Evolution (DE) algorithm, tailored with strategies for incorporating boundary individuals into the mutation process and focusing on different aspects of the neural network architecture—specifically the number of neurons and layers.

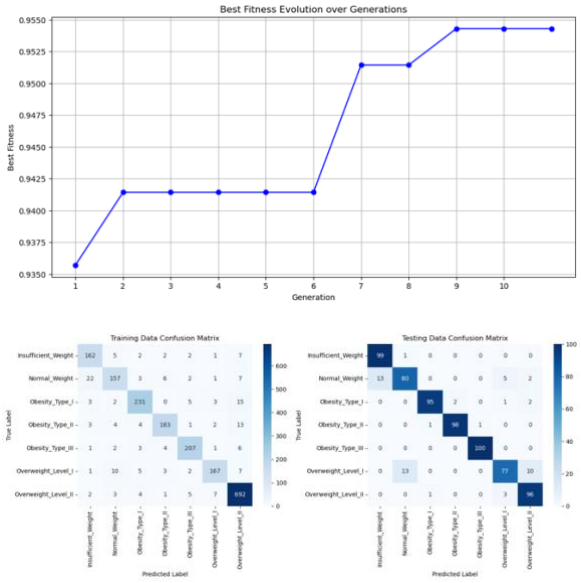
A. Comparative Performance Overview

1. **Shallow Neural Network (Dataset 1):** Served as the baseline model, utilizing a straightforward neural network without advanced optimization techniques.



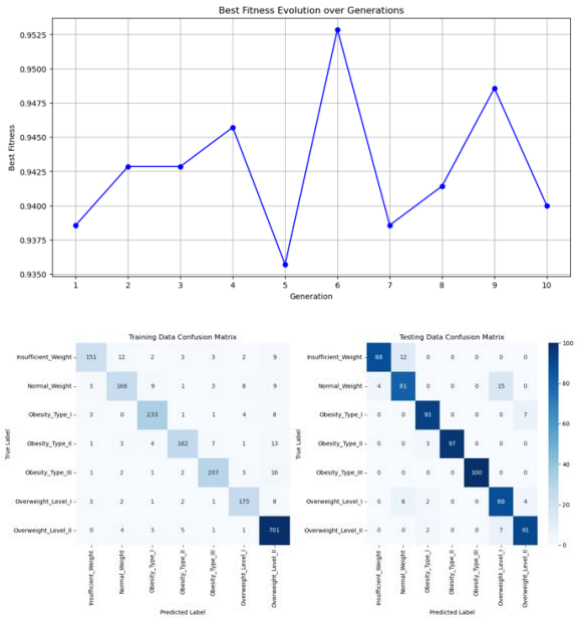
2. **DE for optimizing Neurons and Layers (Dataset 1):** Showed improvements over the baseline Neural Network.

Total computational time: 819.02 seconds. Optimized architecture: 4 layers with neurons: [94, 310, 13, 117]

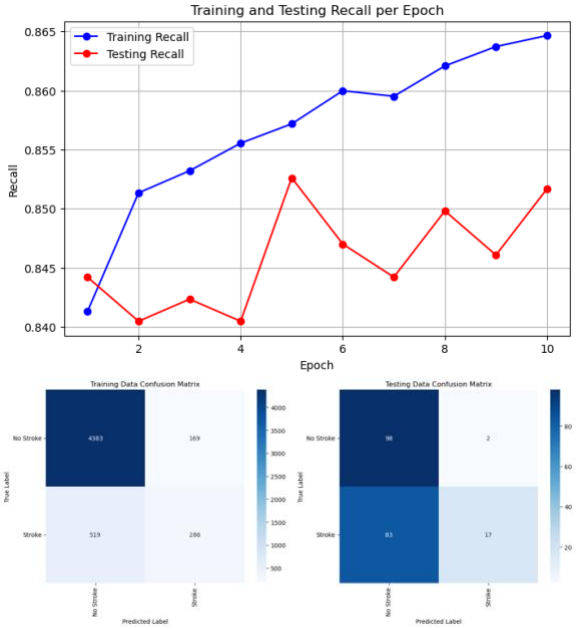


3. **DE with Boundary Consideration for Neurons and Layers (Probability 0.5):** This configuration provided the most significant performance boost across all metrics, achieving the highest fitness scores, precision, recall, and F1 scores.

Total computational time: 862.74 seconds. Optimized architecture: 4 layers with neurons: [179, 78, 42, 297]

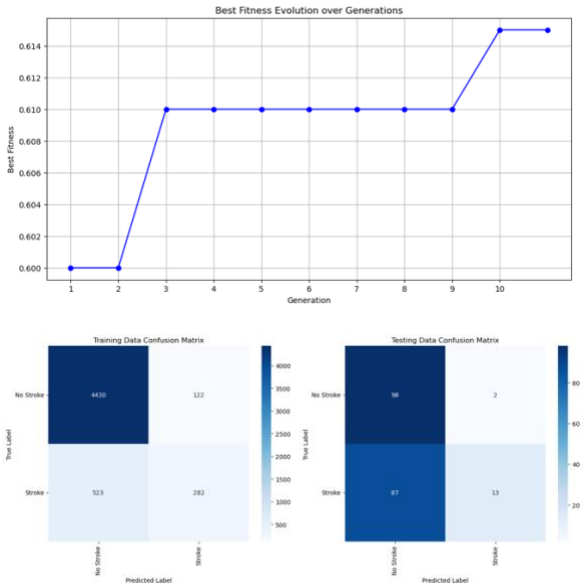


4. **Shallow Neural Network (Dataset 2):** Served as the baseline model, utilizing a straightforward neural network without advanced optimization techniques.



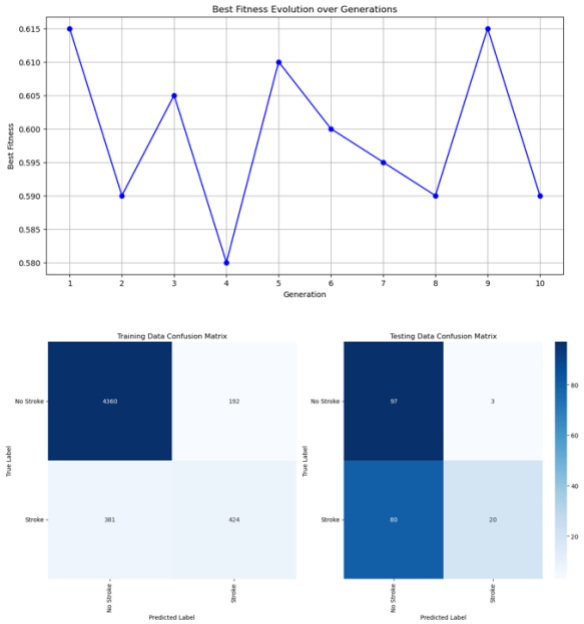
5. **DE for optimizing Neurons and Layers (Dataset 2):** Enhanced classification accuracy and fitness scores, indicating effective layer and neuron configuration.

   Total computational time: 1586.40 seconds. Optimized architecture: 4 layers with neurons: [471, 5, 362, 118]

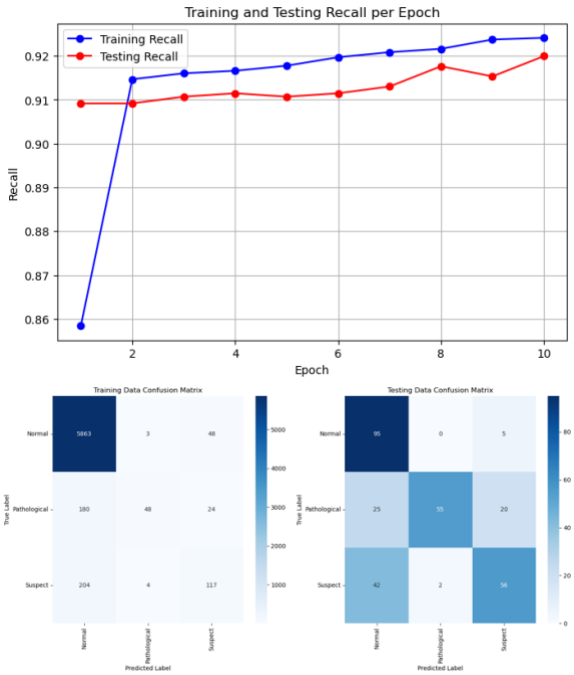


6. **DE with Boundary Consideration for Neurons and Layers with probability 0.8 (Dataset 2)**: Achieved the highest fitness scores, along with excellent precision and recall across all classes, allowing for aggressive experimentation within the solution space.

   Total computational time: 1980.60 seconds. Optimized architecture: 4 layers with neurons: [440, 78, 8, 141]

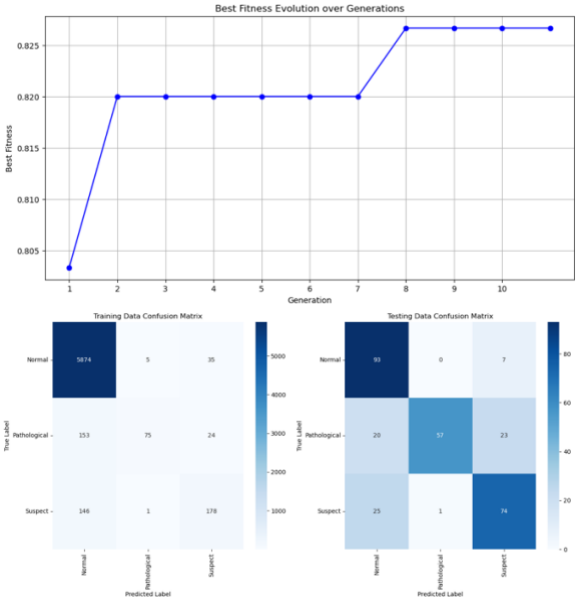


7. **Shallow Neural Network (Dataset 3):** Served as the baseline model, utilizing a straightforward neural network without advanced optimization techniques.



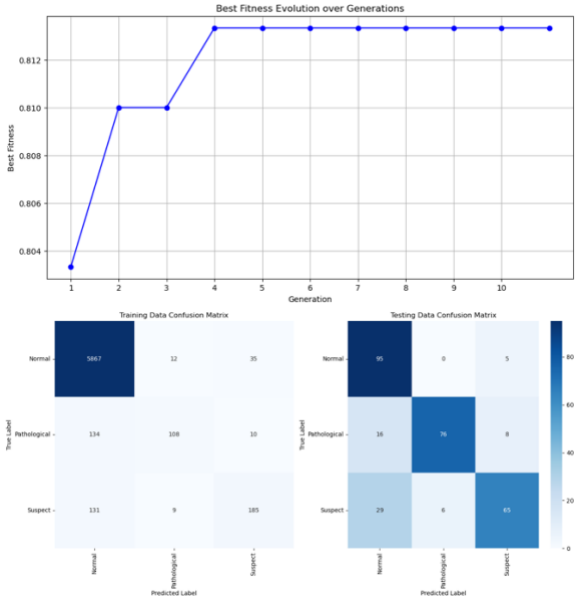
8. **DE for optimizing Neurons and Layers (Dataset 3):** Displayed improvements in fitness scores and training data accuracy, typically finding a more complex setup than the basic neural network.

Total computational time: 2393.91 seconds. Optimized architecture: 3 layers with neurons: [232, 104, 216]



9. **DE with Boundary Consideration for Neurons and Layers with probability 0.5 (Dataset 3):** This configuration yielded the highest improvements across all metrics, including precision, recall, and F1 scores, suggesting a more effective balance between model complexity and training strategy.

Total computational time: 1748.72 seconds. Optimized architecture: 3 layers with neurons: [387, 465, 139]



B. Detailed Performance Metrics

In dataset 1, the shallow network achieved metrics of accuracy and precision at 87.57% and 88.27%, respectively. DE improved accuracy and precision by approximately 3% to 90.56% and 90.86%. DE (0.5) further advanced these figures to 93.34% and 93.41%, showing a nearly 6% improvement over the shallow model. Notably, DE (0.5) also demonstrated greater stability in fitness across generations, with a computational time of 862.74 seconds, compared to 819.02 seconds for DE, indicating with increase in a few seconds we can not only get better performance but can also increase efficiency. This evidence supports the enhanced capability of DE (0.5) in balancing exploration and exploitation effectively

In dataset 2, compared to the baseline shallow neural network, which achieved a testing recall of 84.5%, DE (0.8) demonstrated improved precision and accuracy, with a slightly lower recall of 80%, yielding an overall testing accuracy for 'No Stroke' and 'Stroke' predictions at 97 out of 100 and 80 out of 100 cases, respectively. The best fitness evolution over generations for DE (0.8) showcased dynamic and substantial fluctuations, culminating in a peak fitness of 0.615, indicative of the strategy's effective exploration and exploitation of the solution space. Despite the longer computational time of 1980.60 seconds, the improvements in accuracy and precision underscore the potential of high consideration of boundary individuals in DE to develop robust and reliable models

In dataset 3, the shallow neural network achieved a testing recall of 91.8%, while the initial Differential Evolution (DE) optimization improved this to 92.6%, demonstrating a modest increase. The DE (0.5) model, which optimized both the number of neurons and layers with a probabilistic mutation of 0.5, significantly enhanced the recall to 95.3%, marking a 3.5% improvement over the shallow neural network. The DE (0.5) model exhibited stable fitness evolution after an initial improvement, indicating effective convergence. In terms of computational efficiency, the DE (0.5) model required 1748.72 seconds, which, although longer than the shallow network, resulted in substantial gains in recall and overall predictive

accuracy. This underscores the efficacy of DE (0.5) in balancing exploration and exploitation within the solution space, leading to a more robust and reliable fetal health prediction model.

C. Visual Representation of Fitness Evolution

Fitness evolution charts demonstrated that configurations with moderate to high probabilities of boundary consideration, for both neurons and layers, led to steadier and higher gains across generations. These results confirm the effectiveness of strategically incorporating boundary individuals in enhancing the Differential Evolution (DE) algorithm's ability to explore complex hyperparameter spaces more comprehensively. The next section will discuss these findings and their implications for optimizing neural network architectures.

VI. Conclusion

This research demonstrated that integrating boundary individual consideration into the Differential Evolution (DE) algorithm optimizes neural network architectures more effectively. By varying the probabilities of boundary consideration, it was found that a probability of 0.5 and 0.8 for considering boundary individuals in both neurons and layers is optimal. Key findings include that boundary consideration significantly improves DE's exploration capabilities, enhancing fitness and robustness of neural network architectures. Additionally, a probabilistic rate of 0.5 and 0.8 for boundary consideration optimally balances exploration and exploitation. This method has proven to be computationally efficient and stable across multiple runs, making it highly suitable for practical applications in neural network optimization.

These findings benefit machine learning, especially in automating neural network design for complex tasks, helping to efficiently discover optimal architectures and reduce trial-and-error methods.

Future research could explore adaptive boundary consideration mechanisms and extend this approach to other evolutionary algorithms to enhance applicability and impact.

References

[1] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, pp. 61-106, 2010. DOI 10.1007/s10462-009-9137-2.

[2] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4-31, February 2011.

[3] 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.

[4] J. Liu and J. Lampinen, "A Fuzzy Adaptive Differential Evolution Algorithm," *Soft Computing*, vol. 9, pp. 448-462, 2005, DOI 10.1007/s00500-004-0363-x.

[5] T. Burczynski and W. Beluch, "The identification of cracks using boundary elements and evolutionary algorithms," Engineering Analysis with Boundary Elements, vol. 25, pp. 313-322, 2001.

[6] Z. Liang, T. Luo, K. Hu, X. Ma, and Z. Zhu, "An Indicator-Based Many-Objective Evolutionary Algorithm With Boundary Protection," IEEE Transactions on Cybernetics, vol. 51, no. 9, pp. 4553-4567, September 2021.

[7] L. Dekhici, K. Guerraiche, and K. Belkadi, "Resolving the Evolving Environmental Economic Power Dispatching Problem Using an Enhanced Bat Algorithm," International Journal of Applied Metaheuristic Computing, vol. 11, no. 2, pp. 171-191, April-June 2020. DOI: 10.4018/IJAMC.2020040109.

[8] C. Siew and T. T. Tanyimboh, "Penalty-Free Feasibility Boundary Convergent Multi-Objective Evolutionary Algorithm for the Optimization of Water Distribution Systems," Water Resources Management, vol. 26, no. 15, pp. 4485-4507, December 2012. DOI: 10.1007/s11269-012-0158-2.

[9] X. Liu, G. Li, and P. Shao, "A Multi-Mechanism Seagull Optimization Algorithm Incorporating Generalized Opposition-Based Nonlinear Boundary Processing," Mathematics, vol. 10, no. 18, Article 3295, September 2022. DOI: 10.3390/math10183295.

[10] S. J. Metkar and A. J. Kulkarni, "Boundary Searching Genetic Algorithm: A Multi-objective Approach for Constrained Problems," in Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013, vol. 247, pp. 269-276, 2014. Edited by S. C. Satapathy, S. K. Udgata, and B. N. Biswal. DOI: 10.1007/978-3-319-02931-3_30.

[11] M. H. Tahan and S. Asadi, "EMDID: Evolutionary Multi-Objective Discretization for Imbalanced Datasets," Information Sciences, vol. 432, pp. 442-461, March 2018. DOI: 10.1016/j.ins.2017.12.023.

[12] P. B. C. Miranda, R. F. A. B. Morais, and R. M. A. Silva, "Using a Many-Objective Optimization Algorithm to Select Sampling Approaches for Imbalanced Datasets," in 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 2018, pp. 2324-2330. DOI: 10.1109/CEC.2018.8477988.

[13] S. E. Roshan and S. Asadi, "Improvement of Bagging Performance for Classification of Imbalanced Datasets Using Evolutionary Multi-Objective Optimization," Engineering Applications of Artificial Intelligence, vol. 87, Article 103319, January 2020. DOI: 10.1016/j.engappai.2019.103319.

[14] Y. Zhu, Y. Yan, Y. Zhang, and Y. Zhang, "EHSO: Evolutionary Hybrid Sampling in Overlapping Scenarios for Imbalanced Learning," Neurocomputing, vol. 417, pp. 333-346, December 2020. DOI: 10.1016/j.neucom.2020.08.060.

[15] P. Ducange, B. Lazzerini, and F. Marcelloni, "Multi-objective Genetic Fuzzy Classifiers for Imbalanced and Cost-Sensitive Datasets," Soft Computing, vol. 14, no. 7, pp. 713-728, May 2010. DOI: 10.1007/s00500-009-0460-y.

[16] G. Y. Wong, F. H. F. Leung, and S. H. Ling, "A Hybrid Evolutionary Preprocessing Method for Imbalanced Datasets," Information Sciences, vol. 454, pp. 161-177, July 2018. DOI: 10.1016/j.ins.2018.04.068.

[17] S. García, J. Derrac, I. Triguero, C. J. Carmona, and F. Herrera, "Evolutionary-Based Selection of Generalized Instances for Imbalanced Classification," Knowledge-Based Systems, vol. 25, no. 1, pp. 3-12, Special Issue, February 2012. DOI: 10.1016/j.knosys.2011.01.012.

[18] Z. Ning, Z. Jiang, and D. Zhang, "To Combat Multiclass Imbalanced Problems by Aggregating Evolutionary Hierarchical Classifiers," IEEE Transactions on Neural Networks and Learning Systems, Early Access, April 2024. DOI: 10.1109/TNNLS.2024.3383672.

[19] B. Krawczyk, M. Galar, Ł. Jelen, and F. Herrera, "Evolutionary Undersampling Boosting for Imbalanced Classification of Breast Cancer Malignancy," Applied Soft Computing, vol. 38, pp. 714-726, January 2016. DOI: 10.1016/j.asoc.2015.08.060.

[20] "Obesity level prediction dataset", Available online: https://www.kaggle.com/datasets/fatemehmehrparvar/obesity-levels (accessed on [4/10/2024])

[21] "Stroke prediction dataset", Available online: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset (accessed on [5/15/2024])

[22] "Fetal health prediction dataset", Available online: https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification (accessed on [5/20/2024])