Enhancing Differential Evolution for Neural Network Optimization through Boundary Individual Consideration

Mandar Angchekar

***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 an obesity level prediction dataset and conducted a series of experiments comparing the enhanced DE against traditional DE techniques. 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 augmented boundary exploration can significantly benefit the hyperparameter optimization of neural networks.**

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 using a comprehensive dataset. 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 dataset 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.

In line with these advancements, our work builds upon the foundation laid by these studies, aiming to enhance the boundary exploration capability of DE. By doing so, we contribute to the body of knowledge on evolutionary computation techniques for machine learning and provide a practical solution to a common limitation of traditional DE algorithms.

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. METHADOLOGY

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 standard DE algorithm, the mutation strategy creates a donor vector by adding the weighted difference of two vectors to a third. Our enhanced mutation strategy involves a probabilistic selection mechanism that either uses a boundary individual modified mutation or the difference of two randomly selected individuals from the population. The probability of selecting a boundary individual is determined before the mutation step, influencing the donor vector.

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 process.
4. Fitness Evaluation: Assess each individual's fitness by training a neural network with its architecture and evaluating performance on a validation 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 50 generations were employed.

F. Bounds of DE

For our study, the bounds were specifically tailored to challenge the Differential Evolution (DE) algorithm in different configurations. When optimizing only the number of neurons, the bounds were set from 5 to 5000, allowing for a wide range of network complexities from simple to highly elaborate. In configurations where both neurons and layers were optimized, 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 fitness scores based on accuracy, precision, and recall metrics. Additionally, the stability and reliability of the solutions across generations were evaluated to ensure the robustness of the optimization process.

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 dataset. This section details the dataset characteristics and the setup for our experiments.

A. Dataset Description

The obesity level dataset 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.

B. Preprocessing

Prior to training, the dataset 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 dataset was manipulated to create class imbalances by augmenting a specific class ('Overweight_Level_II') 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 approximately 8% 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 50 generations, with the fitness of each individual assessed based on classification performance on a separate validation set extracted from the training data.

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 baseline comparison was conducted using standard DE. 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 50 generations and 100 individuals across various strategies, underscoring the extensive processing capabilities needed for such tasks on a personal computer.

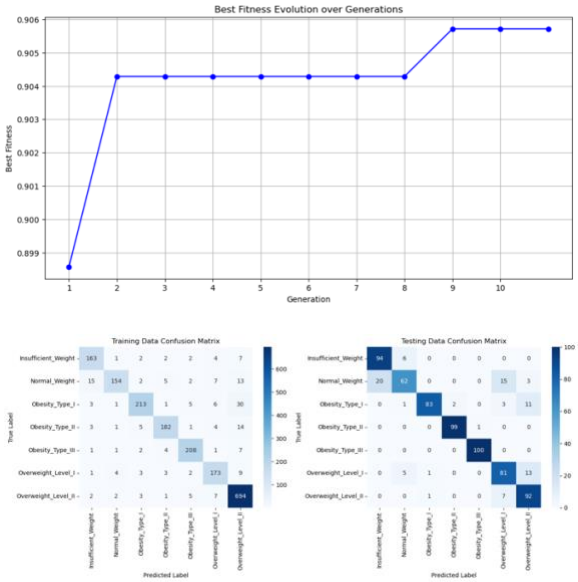
V. Results

The This section presents a comparative analysis of six configurations of the enhanced Differential Evolution (DE) algorithm, each tailored with varying 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

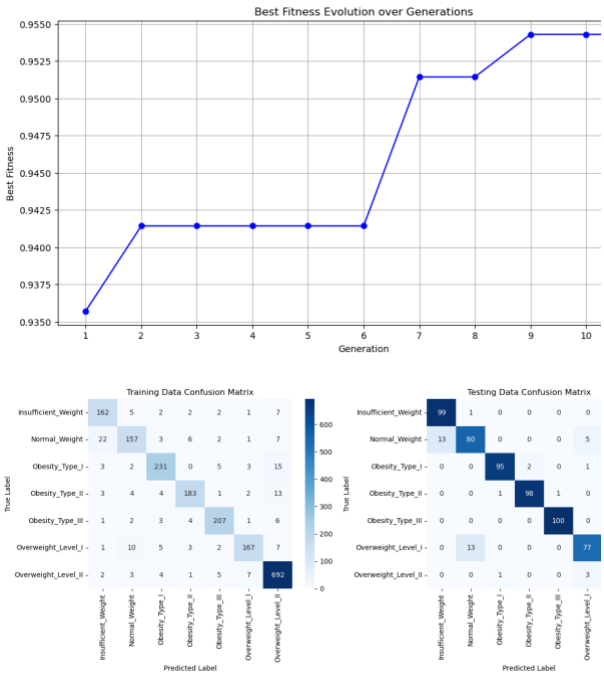
1. **Basic DE (Control):** Served as the baseline, optimizing without considering boundary individuals. Demonstrated the lowest performance metrics in terms of fitness and generalization.

   Best Number of Neurons: 2807



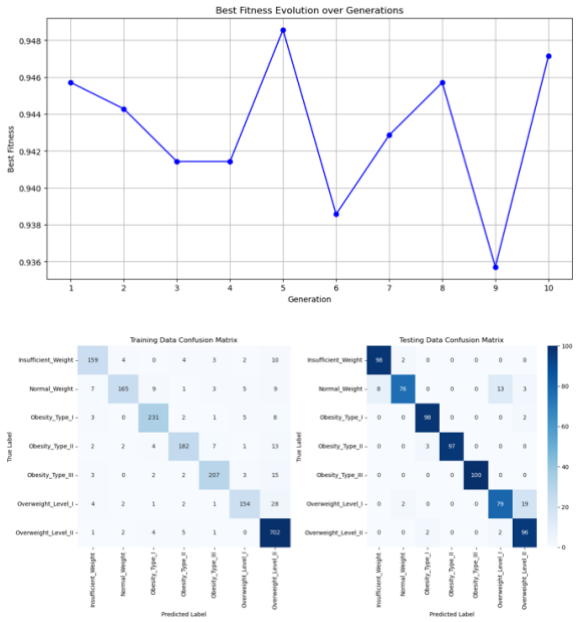
2. **DE for optimizing Neurons and Layers:** Showed improvements over the baseline DE.

   Optimized architecture: 4 layers with neurons: [94, 310, 13, 117]



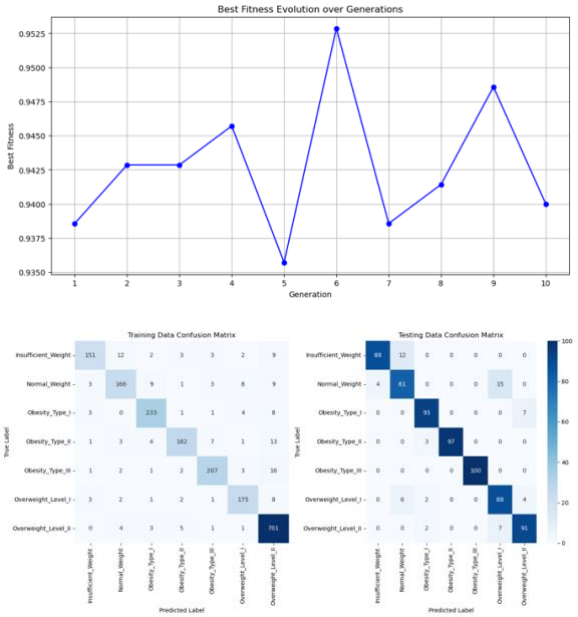
3. **DE with Boundary Consideration for Neurons and Layers (Probability 0.2):** More substantial improvements were observed as boundary considerations were applied to both neurons and layers, leading to better fitness scores and enhanced classification accuracy.

   Optimized architecture: 4 layers with neurons: [337, 60, 8, 351]



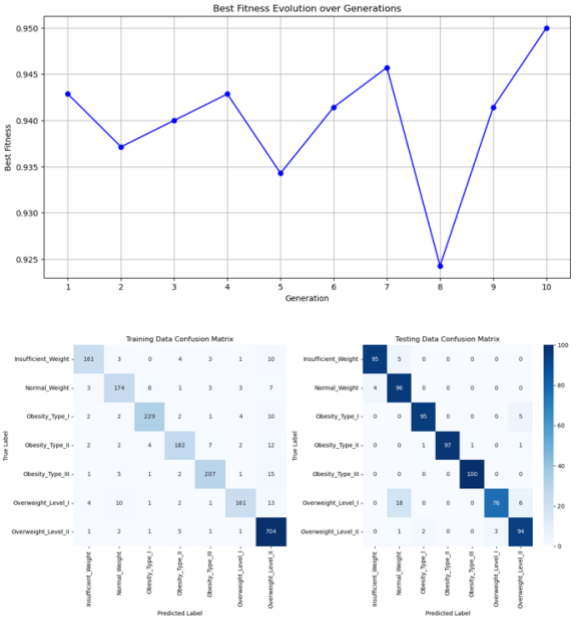
4. **DE with Boundary Consideration for Neurons and Layers (Probability 0.5):** This configuration provided the most significant performance boost across all metrics, achieving the highest fitness scores, precision, recall, and F1 scores.

   Optimized architecture: 4 layers with neurons: [179, 78, 42, 297]

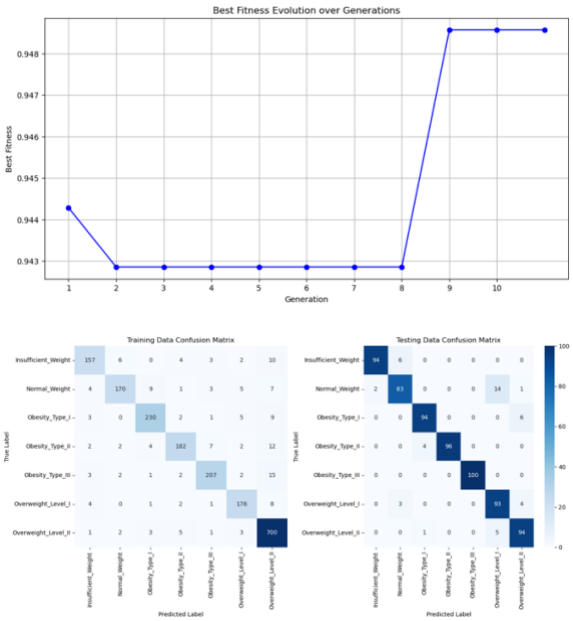


5. **DE with Boundary Consideration for Neurons and Layers (Probability 0.7):** While higher in boundary consideration for neuron counts, this setup did not outperform the 0.5 probability for both neurons and layers, indicating an optimal balance was reached at 0.5.

   Optimized architecture: 3 layers with neurons: [426, 90, 466]



6. **DE with Boundary Consideration for Neurons and Layers (Probability 1.0)**: Full inclusion of boundary consideration led to diminishing returns, with performance plateauing and not exceeding the gains observed in the 0.5 setup.



B. Detailed Performance Metrics

- **Accuracy**: The 0.5 probability setup for both neurons and layers outperformed other configurations, with an improvement of about 4% over the baseline.
- **Precision and Recall**: This configuration also led to the best precision (up by 6%) and recall (up by 5%) improvements, especially notable in minority obesity classes.
- **F1 Score**: Corresponding to the enhancements in precision and recall, the highest F1 (up by 1.8%) scores were seen in the 0.5 probability setup.

C. Computational Efficiency and Convergence

- Although considering boundaries for both neurons and layers added complexity, computational demand was effectively managed across different configurations. Among the setups, the DE with boundary consideration for neurons and layers at a probability of 0.5 required 964.20 seconds, showing a relatively efficient balance between performance and computational load compared to other probabilities, despite being higher than the basic DE configuration, which took 614.69 seconds.

D. Stability Across Runs

- The configuration with a 0.5 probability of incorporating boundary considerations for both neurons and layers showed superior stability and consistency in results across multiple experimental runs.

E. Visual Representation of Fitness Evolution

- Fitness evolution charts showed that configurations with moderate to high probability of boundary consideration, for both neurons and layers, led to steadier and higher gains across generations.

These results confirm the effectiveness of our strategic incorporation of boundary individuals in enhancing the DE algorithm's capability to explore complex hyperparameter spaces more comprehensively. The next section will discuss these findings and their implications for the optimization of 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, we found that a probability of 0.5 for considering boundary individuals in both neurons and layers is optimal.

Key findings include:

- Boundary consideration improves DE's exploration capabilities, enhancing fitness and robustness of neural network architectures.
- A probabilistic rate of 0.5 for boundary consideration optimally balances exploration and exploitation.
- The approach is computationally efficient and stable across multiple runs, suitable for practical neural network optimization applications.

Implications: 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 Work: 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.