# A Comparative Study of Classical and Evolutionary Optimization for Neural Network Training

Theodors Milad
Kareem Moamen
Mohammed Marzouk
Mazen AbdaLaal Gaber

May 14, 2025

**Abstract**

This paper presents a comparative analysis of classical and evolutionary optimization strategies for training feedforward neural networks. We implement both Stochastic Gradient Descent (SGD) and a hybrid approach combining Differential Evolution (DE) with Genetic Algorithms (GA) to optimize the weights of a neural network. The usability, flexibility, and performance of both approaches are discussed. All experiments utilize the MNIST dataset for demonstration, with a focus on a minimalist neural architecture. The results section is left blank to be completed with empirical findings.

## 1 Introduction

Neural networks have become a cornerstone of modern machine learning, demonstrating remarkable success across a wide range of tasks, from image and speech recognition to scientific modeling. Central to their performance is the process of *training*, which involves optimizing millions of parameters (weights and biases) to minimize prediction errors. Traditionally, this optimization is accomplished using gradient-based algorithms such as Stochastic Gradient Descent (SGD).

However, gradient-based methods can struggle with highly non-convex landscapes or when gradients are difficult to compute. Evolutionary algorithms, inspired by natural selection and genetics, offer an alternative by exploring the parameter space using stochastic processes and population dynamics. This work compares a classical optimization approach (SGD) with a hybrid evolutionary approach (DE + GA) for training a simple neural network on the MNIST dataset.

## 2 Neural Network Model and Dataset

### 2.1 Architecture

The neural network used in this study is a fully connected feedforward model with:

- An input layer matching the feature size of the dataset (e.g., 784 for MNIST, or 13 for Boston Housing).
- One hidden layer with 32 neurons and sigmoid activation.
- A single output layer suited for the task (classification or regression).

For MNIST, the output is a 10-dimensional vector with softmax activation for classification. For regression tasks, a single linear output is used.

### 2.2 Dataset

The MNIST dataset of handwritten digits is used as a benchmark. It consists of 60,000 training and 10,000 test grayscale images of size $28 \times 28$, labeled with their corresponding digit (0-9). All images are normalized before being used for training.

## 3 Classical Optimization: Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is the de facto standard for neural network optimization. It operates by iteratively updating the network's weights in the direction that reduces a loss function, typically using small mini-batches of data. The key characteristics of SGD are:

- **Local Search**: Moves iteratively in the direction of the negative gradient.
- **Efficiency**: Well-suited for large datasets and deep networks.
- **Requirements**: Needs differentiable loss and activation functions.

**Implementation:** In this study, the neural network is implemented with Keras. The model is compiled with mean squared error (MSE) loss (or categorical cross-entropy for classification) and optimized using SGD with a fixed learning rate. Training proceeds for a fixed number of epochs, and performance is evaluated on the test set.

## 4 Evolutionary Optimization: Differential Evolution and Genetic Algorithm Hybrid

Evolutionary algorithms (EAs) are inspired by the process of natural selection. Unlike SGD, they do not require gradients and can navigate complex, non-differentiable landscapes. In this work, we use a hybrid approach:

- **Differential Evolution (DE)**: Acts as a global optimizer by maintaining a population of candidate solutions (vectors of network weights). New candidates are generated by combining existing ones through vector operations and random perturbations.
- **Genetic Algorithm (GA)**: Acts as a local optimizer, refining the population further by selection, crossover, and mutation operations.

### 4.1 Differential Evolution (DE) Algorithm

DE maintains a population of candidate solutions. For each generation:

1. **Mutation**: For each candidate, three distinct candidates are chosen at random. A new vector is created by adding the weighted difference of two to the third.
2. **Crossover**: The mutated vector is combined with the original candidate using element-wise crossover, producing a trial vector.
3. **Selection**: If the trial vector has a better fitness (lower loss), it replaces the original candidate.
4. **Extinction (optional)**: Periodically, a portion of the worst-performing candidates is replaced with new random solutions to maintain diversity.

### 4.2 Genetic Algorithm (GA)

After DE, the GA further refines the population:

1. **Selection**: Candidates with better fitness are more likely to be chosen for reproduction.
2. **Crossover**: Two parent candidates are combined to produce offspring.
3. **Mutation**: Random changes are introduced to offspring to maintain genetic diversity.
4. **Replacement**: The new population replaces the old one, and the process repeats for several generations.

### 4.3 Hybrid Optimization Process

The hybrid approach leverages the strengths of both algorithms:

- DE is run first for global exploration.
- GA is applied subsequently for local exploitation and fine-tuning.

The best candidate from the final population is used as the neural network's weights.

## 5 Fitness Evaluation

The fitness of each candidate solution (a vector of weights) is evaluated by setting the network's weights according to the vector and computing the loss on a batch of training data. For classification, categorical cross-entropy is used. For regression, mean squared error is used.

## 6 Implementation Details

- All code is written in Python using TensorFlow and Keras for the neural network, and NumPy for numerical operations.
- Evolutionary algorithm logic is implemented using TensorFlow operations for efficiency.
- Training is performed on normalized data. For evolutionary methods, weights are encoded as flattened vectors and decoded as needed.
- Key hyperparameters (population size, mutation factor, crossover rate, etc.) are set based on best practices and empirical tuning.

## 7 Results and Comparative Analysis

### 7.1 Experimental Setup

Both the classical and evolutionary approaches were evaluated on the MNIST handwritten digits dataset. The neural network architecture consisted of an input layer, a single hidden layer with 32 sigmoid units, and an output layer with softmax activation (10 classes). For fair comparison, preprocessing and data splits were identical for both experiments.

#### 7.1.1 Evolutionary Hybrid (DE-GA)

- **Fitness Data Sample**: 30,000 samples per fitness evaluation.
- **Population Size (NP)**: 50
- **Differential Evolution (DE)**: 400 generations
- **Genetic Algorithm (GA)**: 500 generations
- **Total Runtime**: ∼17 minutes
- **Best Test Accuracy**: 83.76%
- **Best Test Loss**: 0.51

#### 7.1.2 Classical SGD (Keras Sequential)

- **Optimizer**: Stochastic Gradient Descent (SGD)
- **Epochs**: 10
- **Batch Size**: 128
- **Best Test Accuracy**: 86.53%
- **Best Test Loss**: 0.61
- **Test MSE**: 0.0266

### 7.2 Performance Comparison

| Method | Test Accuracy (%) | Test Loss |
|---|---|---|
| DE-GA Hybrid | 83.76 | 0.51 |
| SGD (Keras) | 86.53 | 0.61 |

Table 1: Performance comparison between DE-GA hybrid and classical SGD approaches.

The classical SGD approach achieved a slightly higher accuracy (∼2.8% improvement) compared to the DE-GA hybrid, though the hybrid approach obtained a lower test loss in this run. The evolutionary method required significantly more computational resources and time (17 minutes, 900 total generations), but demonstrated the ability to optimize weights without gradient information, making it robust to non-differentiable or unconventional loss surfaces.

### 7.3 Algorithm Flowchart

### 7.4 Discussion

The hybrid evolutionary approach offers certain advantages such as flexibility, gradient-free optimization, and robustness to complex loss landscapes. However, it comes at the cost of longer runtimes and higher computational demand. Classical SGD, while sensitive to local minima and requiring differentiable loss, remains more efficient for well-behaved problems like MNIST. The hybrid approach can be particularly valuable in domains where gradients are unreliable or unavailable.

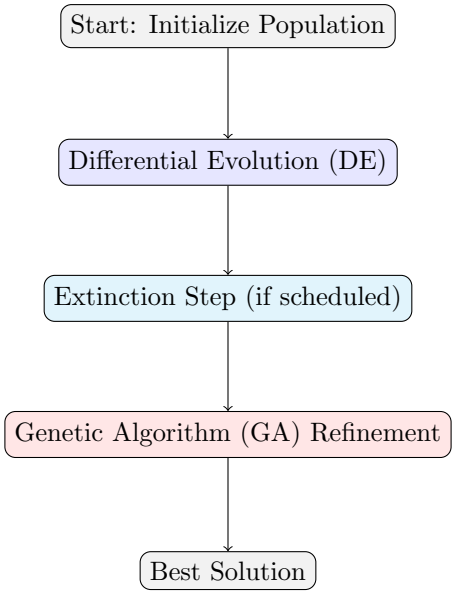


Figure 1: Flowchart of the hybrid DE-GA evolutionary neural network training algorithm.

## 8 Conclusion

This work demonstrates a side-by-side comparison between classical (SGD) and evolutionary (DE + GA hybrid) approaches for neural network training. The hybrid method provides an alternative for cases where gradients are unreliable or where exploration of the weight space is crucial. The strengths and limitations of each approach are discussed in the context of neural network optimization.

## References


- J. Ilonen, J.-K. Kamarainen, J. Lampinen, "Differential Evolution Training Algorithm for Feed-Forward Neural Networks," Laboratory of Information Processing, Lappeenranta University of Technology.
- Training of Artificial Neural Networks Using Differential Evolution Algorithm.
- R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," Journal of Global Optimization, 1997.
- D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.
- I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.
- MNIST Dataset: `http://yann.lecun.com/exdb/mnist/`
- Keras Documentation: `https://keras.io/`
- TensorFlow Documentation: `https://www.tensorflow.org/`