NEURAL EVOLUTION FOR HANDWRITTEN DIGIT RECOGNITION

Supervised by: Dr. Amr Ghonim

MAY 15, 2025

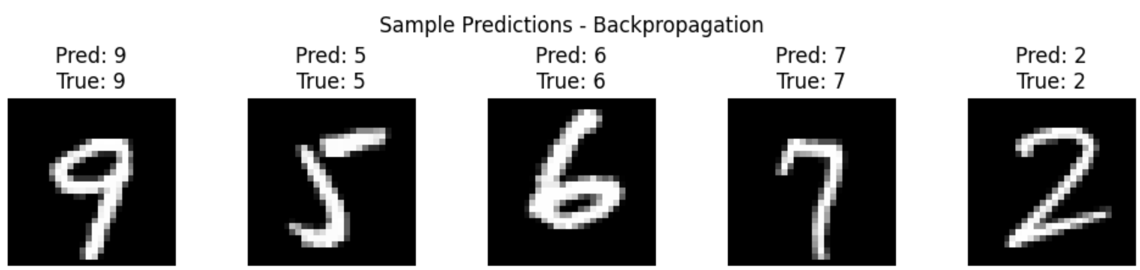

Table of Content

1. Project Idea

This project explores optimizing a neural network for classifying handwritten digits from the MNIST dataset using Differential Evolution (DE) and a hybrid DE + Genetic Algorithm (GA). Instead of relying on traditional gradient-based training methods like backpropagation, DE treats neural network parameters as a high-dimensional optimization problem. The hybrid DE+GA approach incorporates genetic operators—such as crossover and mutation—to improve population diversity and convergence. The goal is to increase classification accuracy using these evolutionary strategies, and compare their performance against standard training methods on a fixed network architecture (784 input, 128 hidden, 10 output neurons; 101,770 parameters).

2. Main Functionalities

- **Pure DE Implementation:** Uses DE to optimize the neural network with configurable mutation strategies (DE/rand/1, DE/best/1) and crossover methods (binomial, exponential). Also includes fitness sharing for population diversity.
- **Self-Adaptive DE Variant:** Enhances convergence by adapting the F and CR parameters for each individual and generation.
- **Hybrid DE+GA Algorithm:** Combines DE operations with GA's tournament selection, uniform crossover, and Gaussian mutation, improving diversity and potentially escaping local optima.
- **Backpropagation Benchmark:** Trains the same network using the Adam optimizer and categorical cross-entropy loss for comparison, typically achieving higher accuracy.
- **Automated Experiments:** Executes five different configurations for both DE and DE+GA (e.g., varied population sizes, crossover strategies),

logs results, and visualizes accuracy trends.
- **Visual Output:** Saves plots of learning curves and predictions for both DE and DE+GA runs, showing comparisons with backpropagation.
- **CSV Export:** Results are saved to Google Drive as CSV files for further analysis and reporting.

3. Similar Applications in the Market

- **NEAT (NeuroEvolution of Augmenting Topologies):** Evolves both neural network topology and weights. Unlike this project's fixed architecture, NEAT dynamically grows the network, offering greater flexibility but at increased complexity.
- **DeepNeuroevolution (Google):** Applies evolutionary methods to deep learning tasks, achieving competitive results on tasks like Atari game playing. However, it's more compute-intensive than this project's design.
- **EvoDeep:** A Python-based library that uses DE for training deep neural networks. It supports diverse network architectures but lacks an integrated visualization or educational interface.
- **OpenAI Evolution Strategies:** Focuses on scalable reinforcement learning. It uses evolutionary approaches at a much larger scale than this educational MNIST-focused implementation.

4. Literature Review

1. Storn & Price (1997) – Differential Evolution: A simple and efficient heuristic for global optimization: Introduced the foundational DE algorithm used in this project.

2. Das & Suganthan (2011) – Differential Evolution: A survey of the state-of-the-art: Explores self-adaptive DE strategies which were incorporated for improved convergence.

3. Stanley & Miikkulainen (2002) – Evolving neural networks through augmenting topologies (NEAT): Offers theoretical context for evolving neural architectures; inspiration for DE+GA hybridization.

4. Such et al. (2017) – Deep Neuroevolution: Describes evolutionary strategies achieving performance on par with gradient-based methods in deep reinforcement learning; the diversity mechanisms are echoed in this project.

5. Salimans et al. (2017) – Evolution strategies as a scalable alternative to reinforcement learning: Validates evolutionary approaches for large-scale learning, providing high-level motivation.

6. EvoDeep Project (GitHub) – Practical implementation of evolutionary deep learning: Guides practical aspects of DE in neural optimization, similar in goal but less focused on educational interpretability.

5. Dataset

- **Name:** MNIST Handwritten Digits
- **Source:** tensorflow.keras.datasets.mnist
- **Size:** 60,000 training images, 10,000 test images (28x28 grayscale)
- **Preprocessing:** Images normalized to [0,1], flattened to 784-

dimensional vectors, labels one-hot encoded for neural network training

- Validation Subset: 2,000-sample subset from the training set is used for fitness evaluation during DE runs to reduce computational cost

6. Algorithms/Approaches and Results

6.1 Approaches

- Pure Differential Evolution:

- **Mutation:** DE/rand/1, DE/best/1
- **Crossover:** Binomial and exponential strategies
- **Fitness sharing:** Encourages population diversity
- Self-adaptive F and CR parameters

- Hybrid DE+GA:

- Tournament selection (k=3)
- Uniform crossover (50% gene exchange)
- Gaussian mutation (rate = 0.05, strength = 0.2)
- Best among parent, DE trial, and GA child selected

- Backpropagation:

- 20 epochs using Keras with Adam optimizer
- Benchmark performance of ~90–95% accuracy

6.2 Experimental Setup

- Main DE Run: 200 generations, population size 200, DE/rand/1 mutation, binomial crossover

- 5 DE Experiments: Pop sizes: 50 and 100; Mutation: rand/1, best/1; Crossover: binomial and exponential; CR: 0.9; F: 0.4 and 0.8

- 5 Hybrid DE+GA Experiments: Same setup as DE, with additional GA crossover and mutation steps
- Backpropagation: Trained on full training set for 20 epochs; evaluated on the test set

6.3 Results

Experiment	Test Accuracy	Runtime (s)
Main DE Run	0.4589	4606.55
Rand/1, Pop50, Binomial (DE)	0.4518	382.24
Best/1, Pop50, Binomial (DE)	0.4518	406.02
Rand/1, Pop100, Binomial (DE)	0.4013	925.18
Rand/1, Pop50, Exponential (DE)	0.4518	392.07
Low F, Pop50, Binomial (DE)	0.5560	390.77
Rand/1, Pop50, Binomial (DE+GA)	0.6637	0.6637
Best/1, Pop50, Binomial (DE+GA)	0.5672	788.4097
Rand/1, Pop100, Binomial (DE+GA)	0.5995	1705.569
Rand/1, Pop50, Exponential (DE+GA)	0.6138	762.8834
Low F, Pop50, Binomial (DE+GA)	0.6674	786.73
Backpropagation	~0.90–0.95	~60–180

7. Development Platform

- Environment: Google Colab

- Hardware: Free tier (CPU-only, 12GB RAM, 100GB storage)

- Dependencies:

  - tensorflow: neural networks + dataset

  - numpy, pandas, matplotlib, seaborn: data handling + visualization

  - deap, streamlit, pyngrok: evolutionary tools and UI

- Execution Flow: Install dependencies via !pip install, mount Google Drive for saving outputs, run DE/DE+GA experiments sequentially, save CSV and PNG results for reporting

- Outputs:

  - project12_de_results.csv
  - project12_dega_results.csv