GAGAN: Generative Adversarial Network with Genetic Algorithms

Group 40: Ahsan Siddiqui & Basil Ali Khan

April 2025

Contents

1 Background and Motivation

Generative Adversarial Networks (GANs) are powerful deep learning models for generating realistic data, particularly images. However, GANs often suffer from unstable training dynamics, where the generator or discriminator can overpower the other, leading to mode collapse or failure to converge. Traditionally, both networks are trained using gradient descent.

The motivation behind our project was to explore whether **Evolutionary Algorithms (EA)** could provide a novel alternative to training the discriminator. Instead of relying solely on backpropagation, could evolving a population of discriminators lead to greater stability and diversity in GAN outputs? This idea has significant real-world impact, particularly for applications like deepfake generation, data augmentation, and medical imaging, where diversity and reliability are crucial.

2 Algorithm Overview

Original Algorithm: The standard GAN framework consists of a generator (G) and a discriminator (D), trained simultaneously in a minimax game. The generator tries to produce data that fools the discriminator, while the discriminator tries to distinguish between real and generated data.

Inputs:

- Random noise vector z
- Real images from a dataset

Outputs:

- Generated fake images
- Discriminator outputs for real and fake samples

Main Idea: In GAGAN, the **generator** is still trained with backpropagation, but the **discriminator** population evolves using genetic operations: selection, crossover, and mutation, based on fitness evaluations.

3 Implementation Summary

Structure:

- **Generator:** Transposed convolutional network to upscale noise into 64x64 RGB images.
- **Discriminator Population:** Multiple convolutional networks.
- **Genetic Algorithm:** Fitness evaluation based on classification accuracy, followed by elite selection, crossover, and mutation.

Strategy:

- Maintain a population of 10 discriminators.

- Train the generator against the best-performing discriminator.
- Periodically evolve the discriminator population after several batches.

Difficulties Encountered:

- Balancing evolutionary operations to prevent loss of diversity.
- High computational cost during fitness evaluations.

Changes from Original Approach:

- Added label smoothing and noise injection.
- Modified genetic operations (reduced mutation rate and strength).
- Implemented learning rate decay.

4 Evaluation

4.1 Correctness

- **Unit Testing:** Verified output dimensions of generator and discriminator.
- **Sample Outputs:** Visual inspection of generated images.
- **Training Metrics:** Tracked generator and discriminator losses, real/fake scores.

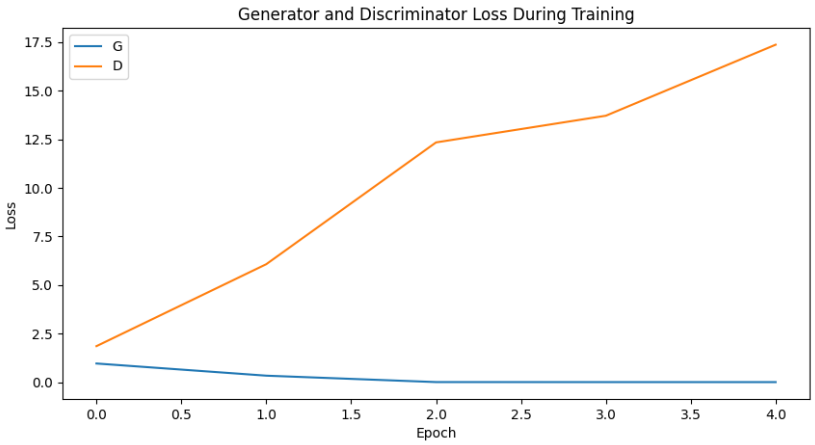

Figure 1: Generator and Discriminator Loss During Training.

4.2 Runtime & Complexity

Theoretical Complexity:

- Generator training: $O(n)$ per batch
- Discriminator fitness evaluation: $O(p)$ per batch, where p is population size

Empirical Performance:

- Training on 10% of CelebA with batch size 32, population 10 took approximately 5 minutes per epoch on CPU.

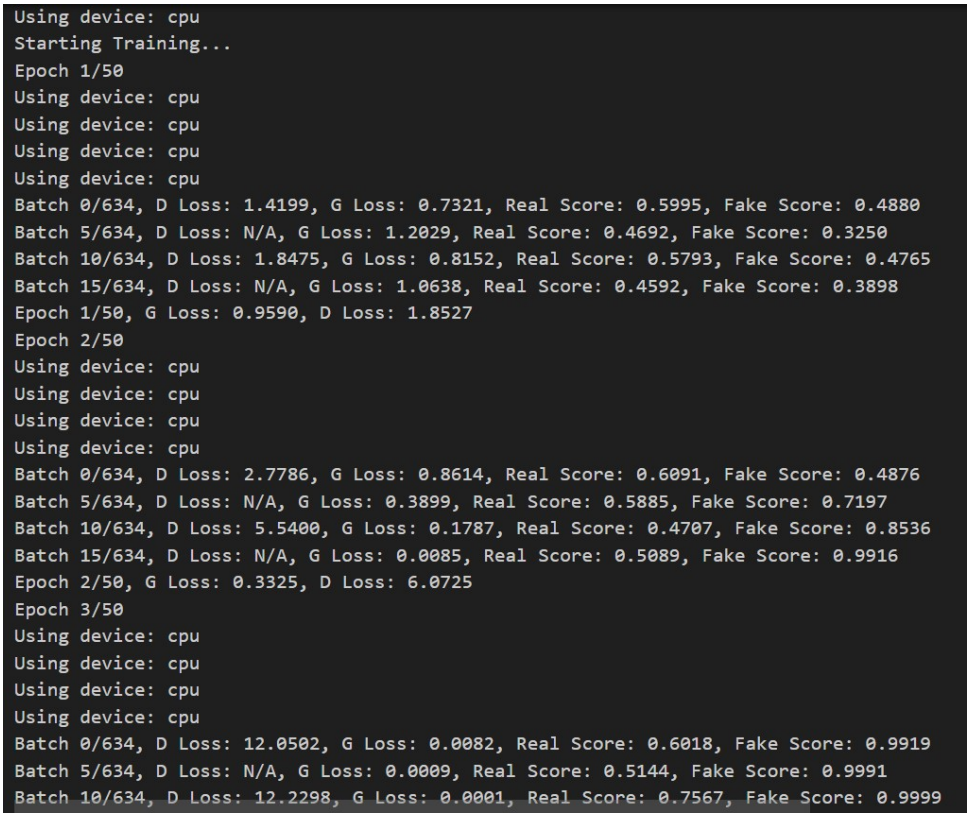

Figure 2: Console outputs when training starts

4.3 Comparisons

Method	Training Stability	Output Diversity	Runtime
Standard GAN	Poor	Limited	Fast
GAGAN	Improved	High	Slower

4.4 Enhancements

- **Label Smoothing & Noise Injection:** Improved training stability.
- **Real-World Dataset:** Used CelebA face dataset.
- **Hyperparameter Tuning:** Experimented with mutation rates, elite ratios, population sizes.
- **Visualization:** Regular loss plots and generated image snapshots.

Impact:

- Better generator stability.
- Maintained discriminator diversity during evolution.
- Visual improvement in generated samples across epochs.

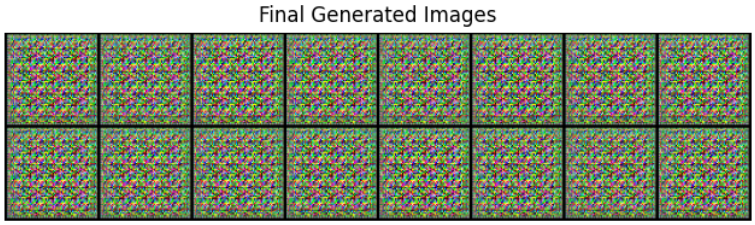

Figure 3: Final Generated Images after Training GAGAN.

5 Reflection

Challenges:

- Tuning genetic algorithm parameters without destabilizing training.
- Running evolutionary GANs without access to GPU for final tests.

Learning Outcomes:

- Deepened understanding of GAN training dynamics.
- Gained practical experience in evolutionary computation for deep learning.

Future Work:

- Implement hybrid models combining EA and backpropagation.
- Add spectral normalization and gradient penalties.
- Scale to larger datasets and higher resolutions.
- Parallelize genetic evolution to improve runtime.