Handwritten Digit Generation using GANs

Author: Nishu Kumari Singh
Dataset: MNIST (60,000 handwritten digits, 28×28 pixels)
Models: Vanilla GAN and DCGAN

Project Overview

This project explores the implementation and training of **Generative Adversarial Networks (GANs)** to synthesize handwritten digits. It focuses on the adversarial relationship between two neural networks:

- **Generator**: Creates images from random noise
- **Discriminator**: Distinguishes between real and generated images

The project demonstrates how these networks compete in a "minimax" game, ultimately producing realistic handwritten digit images from scratch.

Model Architectures

The project implements and compares two distinct architectures:

1. Vanilla GAN

Utilizes fully connected (Linear) layers. This serves as the foundational model to demonstrate basic adversarial training principles.

2. DCGAN (Deep Convolutional GAN)

Employs Deep Convolutional layers with upsampling (Transposed Convolutions) and stride convolutions. This architecture is optimized for spatial feature learning, resulting in higher-quality image generation.

Key Results

- **Successful Convergence**: Achieved stable adversarial training where both the Generator and Discriminator reached a balanced state
- **Performance Comparison**: The DCGAN outperformed the Vanilla GAN in capturing the fine details and strokes of handwritten digits
- **Visual Progression**: Training over 50 epochs showed a clear evolution from random Gaussian noise to sharp, recognizable digits

- **Technical Metrics**: Implementation of Binary Cross Entropy (BCE) loss and Adam optimizers (α=0.0002, β_1=0.5) provided consistent training dynamics

---

Technical Concepts Demonstrated

- **Adversarial Training Dynamics**: Managing the "minimax" game between networks
- **Stability Techniques**: Use of Batch Normalization, Dropout, and LeakyReLU activations to prevent mode collapse
- **GPU Acceleration**: Developed and tested using NVIDIA T4 GPU (approx. 30-40 mins runtime)
- **Deep Learning Frameworks**: PyTorch implementation with torchvision for dataset handling

---

How to Use

Prerequisites

```bash
pip install torch torchvision matplotlib numpy
```

Running the Project

1. **Environment**: Open the `.ipynb` file in Google Colab or a local Jupyter environment with GPU support
2. **Execution**: Run the cells sequentially to observe the training process and the final generated digit grids
3. **Visualization**: Generated images are displayed at regular intervals throughout training

---

Repository Structure

```
├── GENERATIVE_ADVERSARIAL_NETWORKS_(GANs).ipynb
└── README.md
```

Learning Outcomes

This project successfully demonstrates:

- Deep understanding of adversarial training dynamics

- Implementation of state-of-the-art generative models
- Practical experience with convolutional neural networks
- Ability to stabilize and optimize GAN training

Project Status: Completed successfully as an exploration of Deep Generative Models

---

Contact

For questions or collaboration opportunities, feel free to reach out!

Nishu Kumari Singh

License