Deep Convolutional Generative Adversarial Networks for MNIST Image Generation: A Comparative Study of Architecture Configurations

Michael Johnson
ECE5570–Machine Learning at Scale
Assignment 4
December 9, 2025

***Abstract*—This paper presents a comparative study of Deep Convolutional Generative Adversarial Networks (DCGANs) applied to MNIST digit generation. I investigate the effects of different architectural configurations on training stability and output quality by comparing a baseline model with a modified architecture featuring altered latent dimensions, network depth, and regularization. The experiments demonstrate how hyperparameter choices significantly impact model performance, with particular focus on identifying mode collapse and training instability. The baseline configuration produces diverse, high-quality digit samples, while the modified architecture exhibits severe mode collapse, illustrating the sensitivity of GANs to architectural decisions. These findings align with established theoretical understanding of GAN training challenges and provide practical insights for architecture selection.**

***Index Terms*—Generative Adversarial Networks, DCGAN, MNIST, Mode Collapse, Deep Learning, Image Generation**

I. Introduction and Objectives

A. Motivation for Image Generation

Image generation is a core challenge in machine learning, enabling applications in data augmentation, creative AI, and simulation. Generative models can synthesize realistic data, support downstream tasks, and advance our understanding of high-dimensional distributions.

B. Brief Overview of GANs and Diffusion Models

Generative Adversarial Networks (GANs) [1] are a class of generative models where a generator and discriminator compete in a minimax game. DCGANs [2] use convolutional architectures for improved image synthesis. Diffusion models, though not the focus of this report, are a newer class of generative models that iteratively denoise random noise to generate data, and are noted for their stability and sample quality.

II. Methods

A. Dataset and Preprocessing

The MNIST dataset of 60,000 handwritten digit images (28x28 grayscale) was used. Images were normalized to $[-1, 1]$ to match the generator's Tanh output.

B. DCGAN Architecture and Training Regime

The generator maps a latent vector $z \sim \mathcal{N}(0, I)$ to an image using transposed convolutions, batch normalization, and ReLU activations (Tanh at output). The discriminator uses strided convolutions, batch normalization, and LeakyReLU, with adaptive pooling to ensure a 1×1 output. Dropout is optionally applied for regularization. Training alternates between updating the discriminator (real vs. fake) and the generator (fooling the discriminator) using binary cross-entropy loss. Fixed random seeds and deterministic cuDNN settings were used for reproducibility.

C. Diffusion Model Configuration

While this report focuses on DCGANs, diffusion models were reviewed in the course and are referenced for context. No diffusion model experiments were conducted in this work.

D. Cluster, Container, and Slurm Setup

Experiments were run on a university HPC cluster using SLURM for job scheduling. Apptainer containers ensured consistent environments (Python 3.11, PyTorch, CUDA). SLURM scripts specified resource allocation (GPU, CPUs, memory) and experiment parameters.

III. Results

A. Visual Examples from Both Models

Visual inspection of generated samples shows:

- **Baseline:** By epoch 30, the generator produces diverse, realistic digits. All classes (0-9) are represented, with varied handwriting styles and no mode collapse.
- **Modified:** The generator collapses to producing repeated, grid-like patterns with little diversity. Digits are not recognizable, and mode collapse is severe.

B. Quantitative Metrics or Proxy Measures

Generator and discriminator losses were recorded for each experiment. The baseline model's losses stabilized, while the modified model's losses indicated instability and collapse. (Loss curves can be plotted from saved CSVs.)

C. *Training and Inference Performance*

Training was performed for 50 epochs per experiment. Resource usage and convergence time were tracked via SLURM logs. The baseline model converged quickly and stably; the modified model did not improve with additional epochs.

IV. Discussion

A. *Analysis of Mode Collapse, Stability, Sample Diversity*

The baseline DCGAN avoided mode collapse and produced high-quality, diverse samples. The modified model, with reduced latent dimension, increased depth, and dropout, suffered from severe mode collapse and instability, as predicted by GAN theory.

B. *Interpretation of Diffusion Model Behaviour*

No diffusion model was implemented, but course notes suggest diffusion models are less prone to mode collapse and training instability compared to GANs.

C. *Limitations of Your Approach*

Only two configurations were tested, and evaluation was primarily qualitative. Results are specific to MNIST; more complex datasets and quantitative metrics (e.g., FID) would strengthen conclusions.

V. Conclusion

This study demonstrates the sensitivity of DCGANs to architectural choices. The baseline model achieved stable, diverse digit generation, while the modified model failed due to mode collapse. Careful tuning of latent dimension, depth, and regularization is essential for successful GAN training. Future work should include systematic hyperparameter studies, quantitative evaluation, and experiments on more complex datasets and with diffusion models.

Appendix

- Python 3.11, PyTorch (version as in container)
- Apptainer container: dcgan_diffusion.sif
- SLURM cluster: GPU partition, 8 CPUs, 32GB RAM
- Deterministic settings: seed=77, cuDNN deterministic

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[2] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *International Conference on Learning Representations (ICLR)*, 2016.

[3] T. Salimans, I. Goodfellow, W. Zaremba, et al., “Improved techniques for training GANs,” in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[4] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” in *International Conference on Machine Learning (ICML)*, 2017.