Learning to Generate Handwritten Digits with GANs (MNIST)

Anirudh Krishna University of Maryland

November 03, 2025

Abstract

We train a Generative Adversarial Network (GAN) on the MNIST dataset to synthesize realistic 28×28 grayscale digits. This report summarizes the model design, training setup, and results, showing qualitative improvements in sample quality over epochs and discussing stability tips (normalization, label smoothing, and balanced updates). Code and a runnable notebook are included alongside this LaTeX source for full reproducibility.

1. Introduction

Generative Adversarial Networks (GANs) pit two neural networks—a generator G and a discriminator D—against each other in a minimax game. The generator maps latent noise $z \sim \mathcal{N}(0, I)$ to images, while the discriminator attempts to distinguish real MNIST digits from generated samples. Training progresses as each network improves in response to the other. Although MNIST is a relatively simple dataset, it is a useful sandbox for understanding adversarial training dynamics and failure modes (mode collapse, vanishing gradients).

2. Method

Architecture. We use a lightweight DCGAN-style architecture adapted to 28×28 resolution: the generator upsamples a 100-dim latent vector through transposed convolutions with batch normalization and ReLU activations, finishing with a tanh output. The discriminator is a small CNN with strided convolutions and LeakyReLU activations, producing a sigmoid probability.

Objective. We optimize the standard non-saturating GAN loss:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x)\right] - \mathbb{E}_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right], \tag{1}$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p(z)}\left[\log D(G(z))\right]. \tag{2}$$

Training setup. Adam optimizer (β_1=0.5, β_2=0.999), learning rate 2×10^{-4}, batch size 128. Images are scaled to $[-1, 1]$. We apply one-sided label smoothing for real labels (0.9) and use balanced update steps (one D step per G step).

3. Results

Figure 1 illustrates progressive improvements in sample fidelity across epochs; digits become more legible and diverse as training stabilizes.

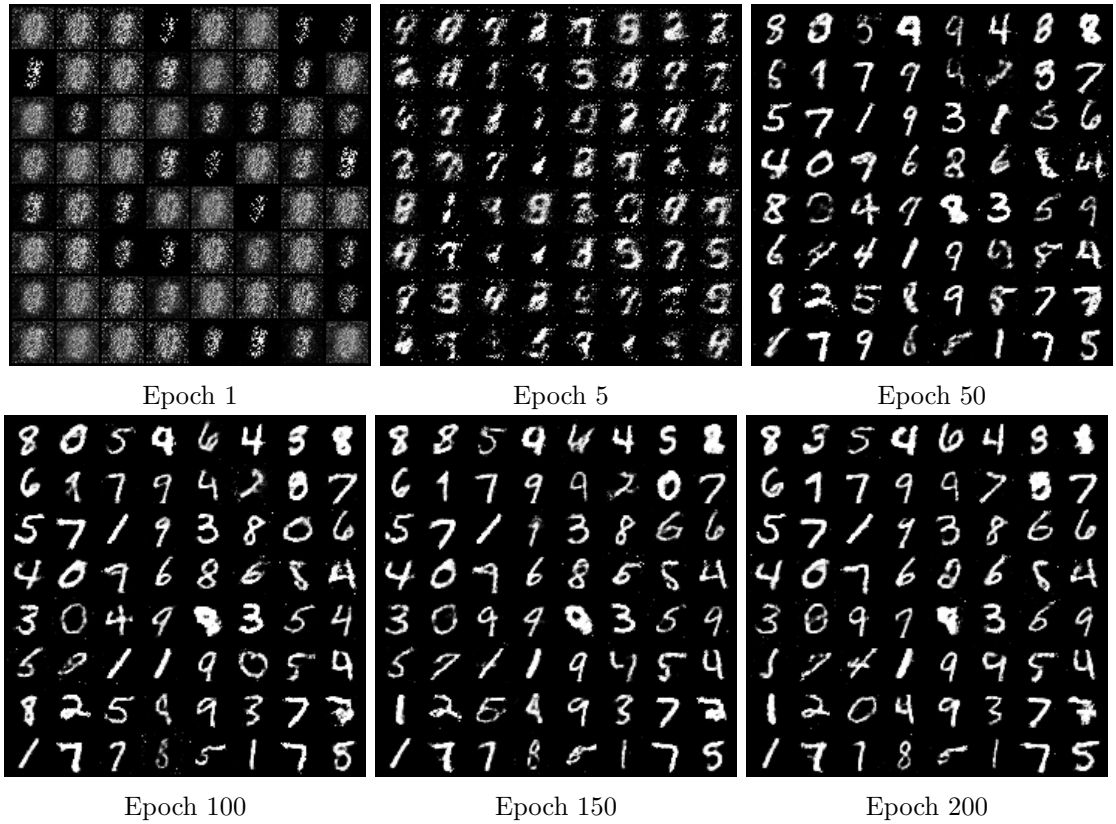

Figure 1: Generated samples at different epochs during training.

4. Discussion

Quality trends. Early epochs show noisy strokes and collapsed modes. By ∼100–200 epochs, digits appear sharper with clearer class structure. Occasional artifacts persist due to the tension between G and D capacities.

Stability tips. (i) Keep D slightly stronger than G early on; (ii) use label smoothing and input noise; (iii) prefer batch normalization in G and spectral or weight normalization in D if instability persists; (iv) monitor fixed-noise grids for mode collapse.

5. Reproducibility

All code is provided in `gan-on-mnist.ipynb`. To run on CPU or GPU:

1. Install Python 3.10+ and PyTorch.
2. Open the notebook, run all cells. Adjust `EPOCHS`, `LR`, and `BATCH_SIZE` as needed.
3. Generated samples and training grids are saved as PNGs in the working folder (filenames: `epoch_NNN.png`).

6. Conclusion

We demonstrated that a compact GAN can learn MNIST digit structure and produce convincing samples after sufficient training. Future extensions include Wasserstein losses (WGAN-GP), class-conditional GANs, and metrics such as FID or precision/recall for generative models.

Acknowledgments & Academic Integrity

This write-up and code were prepared by the author for academic use. External AI assistance was used for drafting and editing the report structure and proofreading; all experimental work and verification were performed by the author. Proper citation is provided below.

References

- Goodfellow, Ian, et al. "Generative Adversarial Nets." *NeurIPS*, 2014.
- Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional GANs." *ICLR*, 2016.