High quality vs Diversity using GANs on MNIST

Evan Azoulay, Alejandro Jorba, Aziz Agrebi

Problem Statement

Goal : Train a GAN to generate synthetic images of handwritten digits that resemble those in the MNIST dataset.

Figure – Images of Real and Generated Handwritten Digits

Approaches

- Vanilla GAN
- WGAN
- WGAN-CP
- Latent Space Adaptation

Vanilla GAN

Optimization Problem

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))]$$

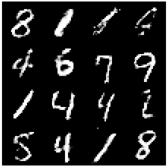


Figure – Samples Generated Using Vanilla GAN

Vanilla GAN

Issues with Vanilla GAN :

- Mode Collapse : the generator produces limited variations, resulting in similar outputs.
- Training Instability : BCELoss (or equivalently JS Divergence) may cause issues when the supports of the data and generated distributions are disjoint.
- Vanishing Gradients : the generator may not improve due to small gradient updates.

How WGAN Improves Vanilla GANs

Wasserstein GAN (WGAN)

- **Wasserstein Distance** : Replaces JS divergence with Wasserstein distance, providing a more stable measure of distribution difference.
- **Training Stability** : Smoother gradients from Wasserstein distance improve stability during training.

Wasserstein Distance Formula

$$W(P, Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$$

- Represents the minimum effort required to align the generated data distribution with the real data distribution.

Using the Kantorovich-Rubinstein duality :

$$W(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_\theta}[f(x)]$$

WGAN Loss Function and Architecture Changes

WGAN Loss Functions

- **Discriminator (Critic) Loss** : Approximates the Wasserstein distance by maximizing the difference in the discriminator's output between real and generated samples.

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}}[D(x)] + \mathbb{E}_{z\sim p_z}[D(G(z))]$$

- **Generator Loss** : Minimizes the discriminator's output for generated samples, encouraging the generator to produce data closer to the real distribution.

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z}[D(G(z))]$$

Architecture Changes

- Sigmoid activation in the discriminator is removed resulting in outputs values in $(-\infty, \infty)$ instead of binary classification.
- Weight clipping is applied to enforce the 1-Lipschitz constraint.

WGAN-GP : Adding Gradient Penalty

Gradient Penalty

- To address the limitations of weight clipping in WGAN, a gradient penalty term is introduced to enforce the 1-Lipschitz constraint more effectively.
- This penalty improves convergence stability without the downsides of clipping.

Final WGAN-GP Loss Function :

$$L = \underbrace{\mathbb{E}_{\tilde{x}\sim P_G}[D(\tilde{x})] - \mathbb{E}_{x\sim P_r}[D(x)]}_{\text{Original WGAN Discriminator loss}} + \lambda \underbrace{\mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\right]}_{\text{Gradient penalty}}$$

Training a Classifier

- CNN trained on MNIST to classify digits.
- Achieved 99% accuracy – reliable "oracle" for GAN output evaluation.
- Baseline for quality assessment by comparing generated images to real images.

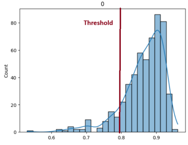
Evaluating Generated Images

- For each generated image $G(z)$, predict its class with CNN.
- Measure similarity between $G(z)$ and real MNIST samples using cosine similarity :

$$\text{cosine\_similarity}(G(z), x_{\text{real}}) = \max_{x \in x_{\text{real}}} \frac{G(z) \cdot x}{\|G(z)\| \cdot \|x\|}$$

Selecting High-Quality Samples

- Use cosine similarity scores to separate high- and low-quality samples.
- Class-specific thresholds ensure both quality and diversity.
- High-quality samples are selected for further latent space refinement.



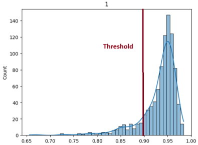


Figure – Cosine-similarity distribution with threshold for classes 0 and 1

Modeling Latent Distribution

- Analyze latent vectors $(z_j^{(y)})$ for each digit class y.
- Test each dimension z_i for Gaussian distribution using Shapiro-Wilk test.
- Estimate means μ_{yi} and standard deviations σ_{yi} for each class.

Generating Optimized Latent Vectors

- Sample new vectors from $N(\mu_y, \text{diag}(\sigma_y^2))$ for each class.
- Generate class-specific images with improved quality and diversity.
- Balanced generation : Produce 1,000 images per digit class, addressing class imbalance.

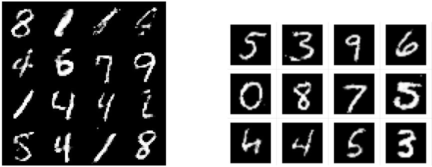


Figure – Generated images without optimizing latent vectors on the left and after on the right

Approach	**FID**	**Precision**	**Recall**
Vanilla GAN	27.96	0.54	0.21
WGAN-CP	31.99	0.52	0.21
Latent Space Adaptation	24.17	0.56	0.2

Figure – Scores of the 3 methods

References I

M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein GAN*, arXiv preprint arXiv :1701.07875, 2017.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, *Improved Training of Wasserstein GANs*, arXiv preprint arXiv :1704.00028, 2017.

T. Issenhuth, U. Tanielian, D. Picard, and J. Mary, *Latent reweighting, an almost free improvement for GANs*, arXiv preprint arXiv :2110.09803, 2021. https://arxiv.org/abs/2110.09803