# Improving GAN on MNIST

**Group name: mayuna**

Marc KASPAR
Yuyan ZHAO
Nan AN

# Contents

# 1 Introduction

This report explores methods to enhance GAN performance on the MNIST dataset, addressing key challenges like mode collapse and unstable training. We begin with f-GANs, testing various f-divergences, including Jensen-Shannon (JS) and Kullback-Leibler (KL). Next, we implement Wasserstein GAN (W-GAN) using Earth-Mover (EM) distance. However, both methods have limitations in achieving diverse and stable sample quality. To address these, we further explore Mode-Seeking regularization, Rejection Sampling, and Latent Space Interpolation, which give the best result with FID 27.72, Precision 0.49, Recall 0.16.

# 2 f-GANs

The f-GAN framework provides a generalized approach to extending GANs to any f-divergence, enhancing the performance and adaptability of generative models through more flexible divergence selection [4].

In our experiment, we adapted the Vanilla GAN code to create an f-GAN and explored the effects of using Kullback-Leibler (KL) divergence, Reverse KL divergence, and Jensen-Shannon (JS) divergence on the model's performance and convergence behavior: First, the final layer activation function of the discriminator is no longer sigmoid; instead, it is set to the recommended activation function by the selected f-divergence, as shown in Table 1. Additionally, the loss functions for both the generator and discriminator were replaced with forms suitable for f-divergence.

| **Name** | **Output activation** $g_f$ | **Conjugate** $f^*(t)$ | **Threshold** $f'(1)$ |
|---|---|---|---|
| Kullback-Leibler (KL) | $v$ | $\exp(t-1)$ | 1 |
| Reverse KL | $-\exp(-v)$ | $-1-\log(-t)$ | $-1$ |
| Jensen-Shannon | $\log(2)-\log(1+\exp(-v))$ | $-\log(2-\exp(t))$ | 0 |

Table 1: Activation functions, Conjugates, Thresholds for various f-divergences.

Throughout our results, we found that the best performance was achieved using the JS divergence with FID 45.53. While f-GANs offer flexibility in choosing divergences, the performance can vary significantly depending on the divergence used on a specific dataset. Users may rely on empirical testing to determine the most effective divergence for their application, which can be time-consuming and resource-intensive.

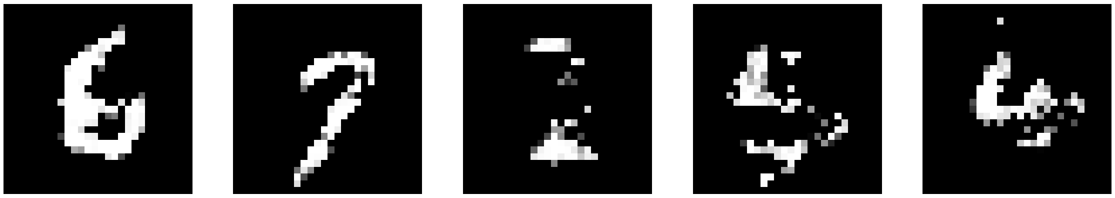
# 3 Wasserstein GAN

We next implemented Wasserstein GAN (W-GAN). Our algorithm was based on the search paper [1]. W-GAN replaces the traditional Jensen-Shannon loss function in GANs with the Earth-Mover distance (EM) or Wasserstein-1 distance defined as follows:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x-y\|\right]$$

where $\Pi(P_r, P_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $P_r$ and $P_g$. It measures the "cost" of transforming $P_r$ into $P_g$. The EM distance benefits from some useful properties. If $Z$ is a random variable and $P_\theta$ is the distribution of $g_\theta(Z)$, then if $g$ is continuous in $\theta$, so is $W(P_r, P_\theta)$. On top of that, if $g$ is Locally Lipschitz, then it is differentiable almost everywhere. WGAN generally converges more stably and addresses issues like mode collapse more effectively than standard GANs.

Here are some sampled images:



The overall results are disappointing, the FID is 87.5474. so we decided to try other strategies.

# 4 Building Model-Mode-Seeking GANs

Through adversarial training, the gradients from $D$ will guide $G$ toward generating samples with the distribution similar to the real data one.

## 4.1 Mode Collapse

The mode collapse problem with GANs is well known in the literature. During our training process, several typical situations may occur:

- Generator Loss increases to 100 as epochs progress, while Discriminator Loss approaches 0.
- Generator Loss and Discriminator Loss oscillate without effective convergence, resulting in poor-quality generated samples.

These are indicators of mode collapse. Since our generator architecture is relatively simple, we can use Mode Seeking GANs to mitigate this issue.

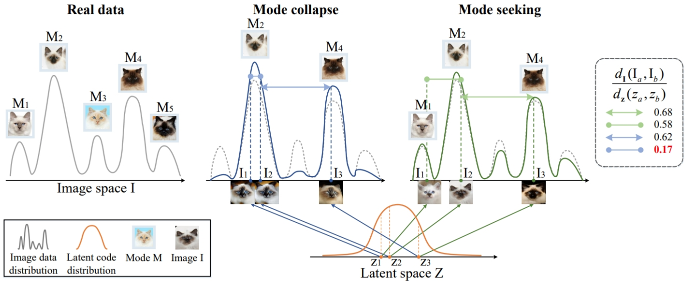


Figure 1: Illustration of motivation

From the data distribution when mode collapse occurs, we observe that for latent vectors $z_1$ and $z_2$, the distance between their mapped images $I_1$ and $I_2$ will decrease disproportionately as the distance between two latent vectors decreases.

## 4.2 Mode Seeking Regularization

As discussed in [3]to address this issue, we can use a mode-seeking regularization term to directly maximize the ratio of the distance between $G(c, z_1)$ and $G(c, z_2)$ with respect to the distance between $z_1$ and $z_2$:

$$L_{ms} = \max_G \left( \frac{d_I(G(c, z_1), G(c, z_2))}{d_z(z_1, z_2)} \right),$$

where $d^*(\cdot)$ denotes the distance metric. The regularization term offers a virtuous circle for training cGANs. It encourages the generator to explore the image space and enhances the chances for generating samples of minor modes.

As shown in Figure 2, the proposed regularization term can be easily integrated with existing cGANs by appending it to the original objective function:

$$L_{new} = L_{ori} + \lambda_{ms} L_{ms},$$

where $L_{ori}$ denotes the original objective function, and $\lambda_{ms}$ is the weight to control the importance of the regularization. For example, in categorical generation tasks,

$$L_{ori} = \mathbb{E}_{c,y}[\log D(c, y)] + \mathbb{E}_{c,z}[\log(1 - D(c, G(c, z)))],$$

where $c$, $y$, and $z$ represent the class labels, real images, and noise vectors, respectively.

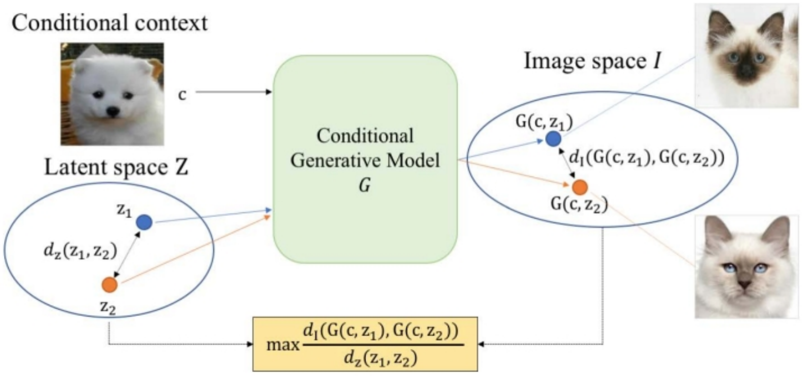

Conditional context
c
Latent space Z
$z_1$
$d_z(z_1, z_2)$
$z_2$
Conditional Generative Model $G$
Image space $I$
$G(c, z_1)$
$d_I(G(c, z_1), G(c, z_2))$
$G(c, z_2)$
$\max \frac{d_I(G(c,z_1), G(c,z_2))}{d_z(z_1,z_2)}$

Figure 2: Proposed regularization

The blue and orange points on the left represent two vectors $z_1$ and $z_2$ in the latent space $Z$, with their distance in the latent space denoted as $d_z(z_1, z_2)$. The two images on the right (different cat faces) are in the image space $I$, obtained by mapping $z_1$ and $z_2$ to the image space through the generator $G$, represented as $G(c, z_1)$ and $G(c, z_2)$, respectively. When mode collapse occurs, even if the two latent vectors $z_1$ and $z_2$ are close to each other, their mapped results are likely to fall into the same mode (e.g., similar cat faces).

The term $L_{ms}$ aims to maximize the ratio between the distance $d_I(G(c, z_1), G(c, z_2))$ in the image space between $G(c, z_1)$ and $G(c, z_2)$, and the distance $d_z(z_1, z_2)$ between $z_1$ and $z_2$.

$\lambda_{ms}$: a hyperparameter used to control the weight of the mode-seeking regularization term in the total loss.

## 4.3 Hyperparameter Selection

We can observe the quality of model training results from the loss values. Generator loss is one way to evaluate model performance

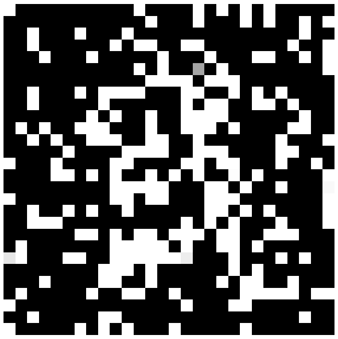


Figure 3: Generator loss 100

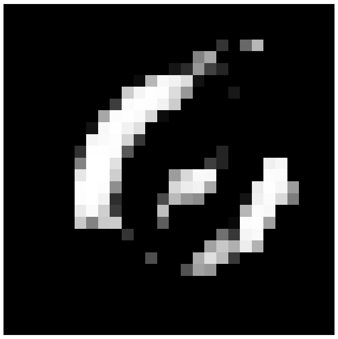


Figure 4: G loss stabilized 24

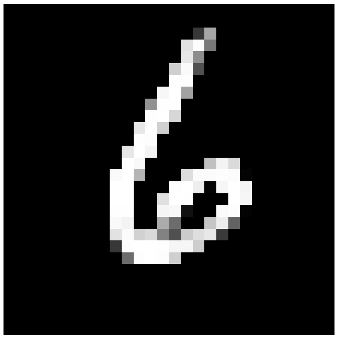


Figure 5: G loss stabilized 13

### 4.3.1 Dropout

Since our generator model is fixed, and I observed that the generator loss often remains high after multiple runs, I decided to add dropout layers in the discriminator.[1] Experimental results indicate that adding two dropout layers with a rate of 0.3 achieved the best performance, reducing the FID score by 6 compared to dropout rates of 0.1 and 0.15.

### 4.3.2 Choosing Mode-Seeking Loss Parameter $\lambda$

When keeping other parameters constant, setting $\lambda$ to 0.1 in the original model resulted in an FID score of 41.01192947773154, while setting $\lambda$ to 0.05 achieved a significantly improved FID score of 35.84250018197383.

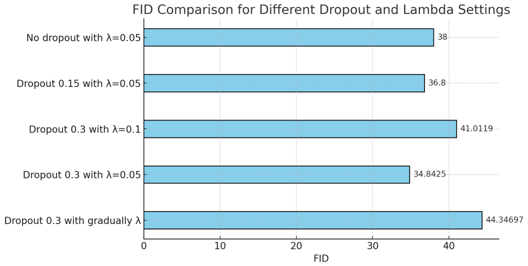


Figure 6: FID for parameter selection

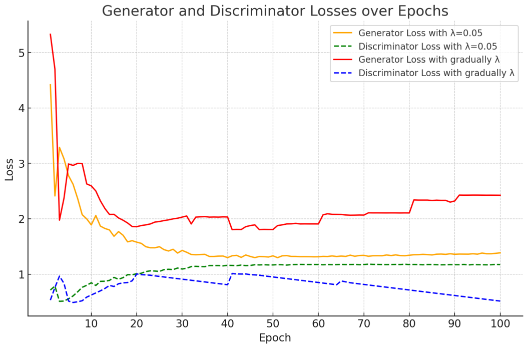


Figure 7: Generator and Discriminator Loss

## 4.4 Increase Mode-Seeking Lambda Gradually

To achieve better precision and recall, I aimed to increase the Lambda parameter gradually[2]. However, to prevent the generator from becoming too weak initially and causing a sudden loss explosion (which occurs if Lambda is set to a high value from the start, leading to an increase in FID), I initially set Lambda to 0.05 and increased it by 0.01 every 10 epochs: args.mode_seeking_lambda += 0.01

As a result, the Generator Loss initially showed a downward trend with good performance. However, after 20 epochs, the loss ceased to decrease and instead slightly increased as Lambda continued to rise.

We observe from Figure 7 that Generator Loss stops decreasing after 20 epochs and instead slightly increases as Lambda grows. The primary reason for this phenomenon may be the initial weakness of the generator: when the generator starts off relatively weak, it may struggle to adapt to the gradually increasing regularization strength. This disrupts the balance in adversarial training, as the generator is forced to produce more diverse samples as Lambda increases, but the discriminator may become too strong, making it challenging for the generator to improve effectively.

However, I believe that if the generator structure were customizable, gradually increasing the Lambda value would be an effective approach to improve the model.

# 5 Optimizing the Generative Model

## 5.1 Rejection Sampling

In the generation phase, the discriminator is used as a quality control tool by providing each generated sample with a confidence score (confidence) to evaluate its authenticity. After generating a batch of images ($x$), the discriminator assesses the "realness" score of each generated image, and only images with discriminator scores above this threshold are saved in the samples folder.

Rejection Sampling led to a reduction in FID by 5, achieving an FID score of 29.

[1] I also experimented with adding Gaussian noise and data augmentation; however, these methods resulted in performance degradation, likely due to the generator's limited capacity.

### 5.2 Latent Space Interpolation

In Generative Adversarial Networks (GANs), the input to the generator is typically a random noise vector $z$. This vector comes from a latent space, and the generator maps the latent space vectors into high-dimensional samples (such as images).

Assuming we have two latent vectors $z_1$ and $z_2$, the interpolation formula is as follows[5]:

$$z_{\text{interp}} = \alpha \cdot z_1 + (1 - \alpha) \cdot z_2$$

where $\alpha$ is a random number between [0, 1] (each sample has an independent random value). By changing the value of $\alpha$, we can generate a series of new vectors $z_{\text{interp}}$ that range from $z_1$ to $z_2$. These new vectors are fed into the generator, producing a series of samples.

This technique allows the generator to produce a continuous range of variations between two randomly selected latent vectors, thus enhancing the diversity and smoothness of generated samples.

Rejection Sampling led to a reduction in FID by 2, achieving an FID score of 27.

## 6 Conclusion

We initially experimented with f-GAN and WGAN. Finally, we applied techniques such as Mode-Seeking Regularization, Rejection Sampling, and Latent Space Interpolation to address common issues like mode collapse and to enhance the quality and diversity of generated images.

Due to limitations of the initial generator architecture, I believe that exploring Semi-Supervised Learning and optimizing the exploration of the latent space are also interesting topics for improving this experiment.

## References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[2] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[3] Qifeng Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1429–1437, 2019.

[4] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization, 2016.

[5] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.