The GANtastics team members:
Ángel Luque
Arij Boubaker
Lyna Bouikni

1 Introduction, Context, and Aims

Generative Adversarial Networks (GANs) have garnered significant attention, especially for their ability to generate high-quality synthetic data. They consist of two neural networks — the Generator and the Discriminator — engaged in a continuous adversarial game. The Generator aims to create data that is indistinguishable from real data, while the Discriminator endeavors to distinguish between real and generated data. This competitive interaction encourages the improvement of both networks, leading to the generation of increasingly realistic data.

The primary aim of this work is to first train a Vanilla Generative Adversarial Network (GAN) on the MNIST dataset while adhering to the constraints of the assignment, **i.e,** the architecture of the Generator is kept fixed. Then, we seek to explore and implement two enhancements to the Vanilla GAN with the intent to refine the quality of generated data. By systematically documenting the effects of these improvements, our study aspires to offer insights that may inform future research and applications of GANs in various domains of machine learning and artificial intelligence.

2 Materials and methods:

The following section provide a comprehensive exposition of the employed methods in this study. All the codes presented are available at: **https://github.com/Master-IASD/assignment2-2023-the-gantastics**

Our approach involves two primary enhancements to the Vanilla GAN architecture:

1. **Hyperparameter Optimization:** We begin by tuning the hyperparameters of the Vanilla GAN. This process involves systematically adjusting various parameters like learning rates, batch sizes, and discriminator network layers, aiming to enhance the model's performance on the MNIST dataset.
2. **Implementation of F-GAN and Discriminator Rejection Sampling:** Subsequent to hyperparameter optimization, we implement the f-GAN framework, by leveraging different f-divergences. Additionally, we implement DRS, a technique designed to refine the selection process of generated samples, thereby improving the quality of data generation.

A more comprehensive presentation of the latter methods is presented in the following sections. In the evaluation of our model, we measured its performance using several key metrics: Fréchet Inception Distance, Precision and Recall. These metrics provide insights into both the quality of the generated images and the efficiency of the learning process.

2.1 Vanilla GAN:

2.1.1 Architecture description:

2.1.1.1 Generator Network:

- **Input:** The generator network begins with a latent variable z, which is a 100-dimensional vector sampled from a standard normal distribution $z \in \mathbb{R}^{100}$.
- **Output:** The generator's task is to map the latent space vector z to the data space, producing an output $G(z)$ which is a 784-dimensional vector corresponding to a flattened image $G(z) \in \mathbb{R}^{784}$.
- **Transformation:** The generator uses a series of linear transformations followed by nonlinear activation functions to transform the input z. The transformation can be represented as: $G(z) = \tanh(W_4 \cdot \text{LeakyReLU}(W_3 \cdot \text{LeakyReLU}(W_2 \cdot \text{LeakyReLU}(W_1 \cdot z + b_1) + b_2) + b_3) + b_4)$

2.1.1.2 Discriminator Network:

- **Input:** The discriminator network takes in either real data or fake data generated by G. The input x is a 784-dimensional vector $x \in \mathbb{R}^{784}$.
- **Output:** The discriminator outputs a single scalar $D(x)$ representing the probability that x is from the real data distribution. The output is bounded between 0 and 1, $D(x) \in [0, 1]$.
- **Transformation:** Similar to the generator, the discriminator uses a series of linear and nonlinear transformations: $D(x) = \sigma(W_4' \cdot \text{LeakyReLU}(W_3' \cdot \text{LeakyReLU}(W_2' \cdot \text{LeakyReLU}(W_1' \cdot x + b_1') + b_2') + b_3') + b_4')$

2.1.2 Hyperparameters Tuning

For the training of the Vanilla GAN, the following hyperparameters were chosen after several trials in order to optimize the balance between training stability and convergence speed: **Epochs** $= 50$ $\|$ **Batch Size** $= 64$ $\|$ **Learning Rate** $= 0.0001$

2.2 F-GAN:

F-GANs, or f-divergence Generative Adversarial Networks, represent a class of generative models that enhance traditional GANs by using f-divergences to create more adaptable objective functions for the generator and discriminator. This allows for customization to the data and desired output characteristics, offering a versatile framework for generative models. There are many divergence functions that we can use, but we have decided to focus our efforts in Kullback-Leibler divergence (KLD), Jensen-Shannon divergence (JS), and the original loss function of the project Binary Cross Entropy loss (BCE, for comparison with the others). These divergences can be expressed as:

Binary Cross-Entropy (BCE):

$$\text{BCE}(y, \hat{y}) = -\left(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right)$$

Kullback-Leibler Divergence (KLD):

$$D_{\text{KL}}(P \,\|\, Q) = \sum_i P(i) \cdot \log\left(\frac{P(i)}{Q(i)}\right)$$

Jensen-Shannon Divergence (JS):

$$D_{\text{JSD}}(P \,\|\, Q) = \frac{1}{2} D_{\text{KL}}(P \,\|\, M) + \frac{1}{2} D_{\text{KL}}(Q \,\|\, M)$$

Having this, we designed a code that allowed us to test all these training configurations, and save their training for different number of epochs. These changes in the code, and also the generation, have potentially resulted on our code failing to properly run in the testing platform. This, combined with the lack of time (platform only tests once a day), has resulted in us not having metrics values for these results, so we will base our assessment in the visual quality of the samples. For each of the configurations, we will show 20 random samples (not cherry picked), in the hope that they represent the general results of the training.

2.3 Discriminator Rejection Sampling (DRS) Implementation:

In our study, DRS was employed as a post-processing step to enhance the quality of samples generated by the F-GAN trained on the MNIST dataset. The main implementations steps are:

2.3.1 Discriminator Scoring

The discriminator, assigns a score to each sample produced by the generator. Higher scores indicate samples that are more likely to be real. These scores are then used to compute an acceptance probability for each sample.

2.3.2 Acceptance Probability Calculation

For a batch of generated samples, the acceptance probabilities are calculated based on the discriminator's output scores, a dynamically estimated maximum score M, and a hyperparameter γ. The function $F(x)$ is computed using the discriminator's logits, adjusted by M and γ, followed by the application of the sigmoid function to determine the acceptance probabilities.

2.3.3 Dynamic Estimation of M

The value of M is dynamically estimated as the maximum discriminator score observed over the course of generation. It represents the upper bound on the quality of generated samples and is used to calibrate the acceptance threshold.

2.3.4 Sample Generation and Selection

Samples are produced in batches. Each batch is evaluated by the discriminator, and an acceptance threshold is applied based on the calculated acceptance probabilities. Samples with acceptance probabilities above a certain threshold are saved, while others are discarded. This process is repeated until the desired number of samples is generated (10.000 samples).

3 Results and Discussion

3.1 Vanilla GAN Results

Here are the results for the Vanilla GAN:

Time (s)	FID	Precision	Recall
111.4	52.44	0.52	0.18

Figure 1: Quantitative performance metrics for the Vanilla GAN.

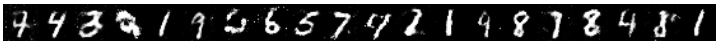
Figure 2: Sample output from the Vanilla GAN, illustrating the clarity of generated handwritten digits.

The Vanilla GAN achieves a moderate FID of 52.44, suggesting decent image quality but not high-fidelity. A precision of 0.52 indicates that over half of the generated images convincingly mimic real data. However, a low recall of 0.18 points to a lack of diversity in the images produced. Visually, some generated digits are clear, while others are blurred, reflecting the quantitative findings. Improvements might require model refinements and hyperparameter tuning to enhance both precision and recall.

3.2 F-GAN results:

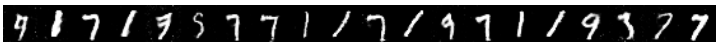
Below are the results for training the F-GAN with the three loss functions at 50, 100, 150 and 200 epochs:

Results for BCE Loss Function:



(a) Epochs = 50

(b) Epochs = 100

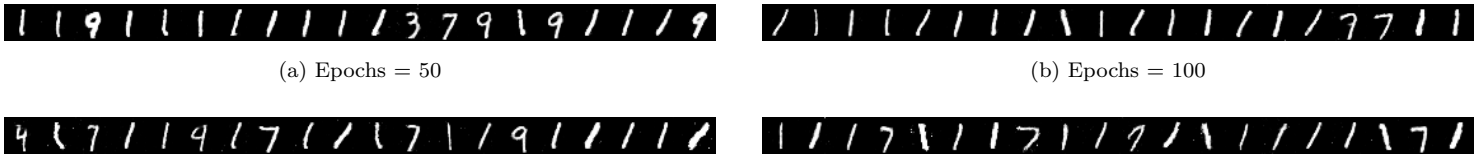


(c) Epochs = 150

(d) Epochs = 200

Figure 3: BCE Loss Function Results

Results for KLD Loss Function:



(a) Epochs = 50

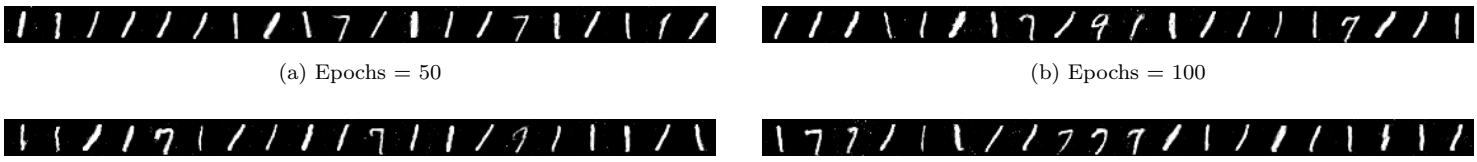
(b) Epochs = 100

(c) Epochs = 150

(d) Epochs = 200

Figure 4: KLD Loss Function Results

Results for JS Loss Function:



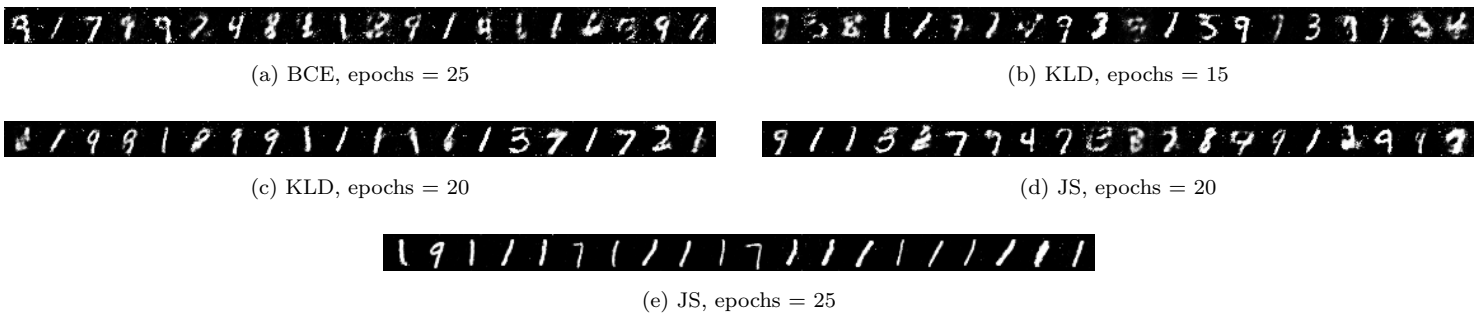
(a) Epochs = 50

(b) Epochs = 100

(c) Epochs = 150

(d) Epochs = 200

Figure 5: JS Loss Function Results



(a) BCE, epochs = 25

(b) KLD, epochs = 15

(c) KLD, epochs = 20

(d) JS, epochs = 20

(e) JS, epochs = 25

Figure 6: Various Loss Function Results at Different Epochs

We can see that 25 epochs for BCE outputs very bad precision, resulting in very bad quality numbers. However, in the cases for KLD and JS, we can say that even if 15 or 20 epochs output worse precision (even though JS is a bit better than KLD in that field), we can achieve a higher recall score. It's also noticeable than in only 5 epochs, in both of these cases, we experience a huge change in precision and recall. This suggest that it would be interesting in further research to find a "sweet spot" between these configurations, probably choosing a smaller learning rate, in order to find a better tradeoff between precision and recall.

The performance of loss functions and epoch counts varies significantly. Optimal epoch determination is challenging for classifiers like ours, where loss is not inherently informative, FID focuses on precision over recall, and visual assessments are subjective. The MNIST dataset, however, simplifies identifying recall visually by checking for variety in the digits presented. Although tracking precision/recall is useful, it necessitates a balance, influenced by the non-intuitive parameter of epochs. Generally, fewer epochs may yield higher recall but lower precision, and vice versa for more epochs. This is consistent across all settings, though the progression rate varies: for BCE, the trade-off becomes notable close to 200 epochs, while for JS and KLD, it's observable as early as 50 epochs, with little change beyond that. Exploring lower epochs could yield further insights (see Figure 6).

3.3 DRS results:

The efficacy of Discriminator Rejection Sampling (DRS) combined with various loss functions was examined across different training epochs:

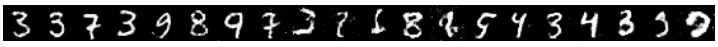
Results for F-GAN trained with BCE Loss Function + DRS:

(a) Epochs = 25

(b) Epochs = 50

(c) Epochs = 100

(d) Epochs = 150



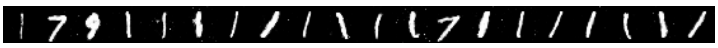
(e) Epochs = 200

Results for F-GAN trained with JS Loss Function + DRS:

(f) Epochs = 20

(g) Epochs = 25

(h) Epochs = 50



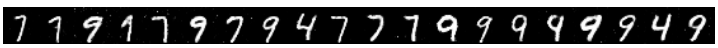
(i) Epochs = 100

Results for F-GAN trained with KLD Loss Function + DRS:

(j) Epochs = 15

(k) Epochs = 20

(l) Epochs = 50



(m) Epochs = 100

Based on the results shown above, DRS led to an improvement in both the quality and diversity of the generated samples compared to the previous results.The BCE loss exhibits an improvement in both clarity and diversity from 25 to 200 epochs. With JS and KLD losses, initial epochs produce diverse representations, suggesting efficient early learning. However, as training progresses to later epochs, we note improved precision but reduced diversity. Stability in the visual quality of digits using KLD loss from epochs 15 to 50 suggests a swift achievement of an optimal precision-recall balance, which then levels off. JS loss follows a similar trend, albeit reaching the quality peak in fewer epochs. DRS accentuated the intrinsic properties of each loss function: BCE's consistent learning over time and JS/KLD's swift quality saturation. This highlights the criticality of epoch optimization when training F-GANs with DRS to finely tune the balance between precision and recall.

4 Conclusion

This project was centered on the generation of 10,000 samples employing Generative Adversarial Networks. Our initial approach employed a vanilla GAN configuration, which underwent extensive hyperparameter optimization to serve as a reliable baseline. Despite this, the resulting sample quality was not as high as anticipated. The subsequent application of the f-GAN framework, which incorporates f-divergences for optimization, provided only marginal improvements. While it demonstrated a theoretical advancement over the vanilla GAN, it did not significantly enhance sample diversity or fidelity.The introduction of Discriminator Rejection Sampling (DRS) was the pivotal enhancement in our study. DRS led to an improvement in both the quality and diversity of the generated samples, validating its effectiveness in refining the generative process.The progression through these various GAN configurations emphasizes the criticality of strategic model refinement. The implementation of discriminator-based techniques, such as DRS, has proven to be particularly promising for improving the quality of sample generation. The findings from this research contribute a valuable framework for future applications and development in GAN-based generative tasks, particularly in optimizing sample quality and diversity.