Data Science Lab Project 2 (IASD 2024-2025) : Generative Adversarial Networks

Simon Liétar, Lucas Henneçon, Ilian Benaïssa-Lejay

November 12, 2024

1 Introduction

Generative Adversarial Networks (GANs) are a class of machine learning models that have gained popularity for their ability to generate realistic synthetic data. They consist of two components: a generator, which creates samples, and a discriminator, which evaluates them. In this project, we explored the development and improvement of GANs for generating MNIST-like images. We began with a vanilla GAN, then moved to the Wasserstein GAN (WGAN) model, which replaces the Jensen-Shannon divergence with the Wasserstein distance to achieve more stable training and improved results.

After observing some limitations with the standard WGAN, such as instability and slow convergence, we implemented the WGAN with Gradient Penalty (WGAN-GP). This variant imposes a gradient penalty on the discriminator to enforce the Lipschitz constraint, leading to enhanced stability and higher-quality image generation. Further, we incorporated rejection sampling to filter out low-quality generated images based on their Inception Score, thereby improving the quality of the final output. Metrics such as the Fréchet Inception Distance (FID) were used to quantitatively evaluate the quality of generated images.

2 Wasserstein GAN (WGAN)

2.1 Overview of WGAN

After training the Vanilla GAN, we tried another type of GAN called Wasserstein GAN which uses the Wasserstein distance instead of the Jensen-Shannon divergence [1]. Below is the formula of the Wasserstein distance:

$$W(P_r, P_G) = \inf_{\gamma \in \Pi(P_r, P_G)} E_{(x,y)\sim\gamma} \left[\|x - y\|\right]$$

It is shown in [1] that using the Kantorovich-Rubinstein duality, we have:

$$W(P_r, P_G) = \sup_{\|D\|_L \leq 1} E_{x\sim P_r}[D(x)] - E_{x\sim P_G}[D(x)]$$

where D must be 1-Lipschitz continuous. It is this expression that we will use in our model, which leads to the min-max expression of the WGAN:

$$\min_G \max_{\|D\|_L \leq 1} E_{x\sim P_r}[D(x)] - E_{x\sim P_G}[D(x)]$$

where D is the discriminator, G is the generator, P_G is the distribution of the images generated by the generator, P_r is the distribution of the real images. The other difference with the Vanilla GAN is that in a WGAN, we remove the sigmoid activation at the output of the discriminator. This implies that the output is not anymore bounded between 0 and 1. In the original paper, the discriminator is called critic, but we will keep the name discriminator in this paper.

2.2 Ensuring 1-Lipschitz continuity with weights clipping

Several methods exist to ensure the 1-Lipschitz continuity of D. The first one that we implemented is the one used in the original paper [1], which consists in clipping the weights of D between $[-c, c]$, where c is a hyperparameter. Fig.1 gives the pseudo-code of the algorithm. Note that [1] also introduces n_{critic}, which represents the number of gradient updates of the discriminator for one gradient update of the generator.

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.
while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Figure 1: Algorithm and hyperparameters for training D and G of WGAN as described in [1].

2.3 Results and Analysis

In this model, the loss of the discriminator is:

$$E_{x\sim P_r}[D(x)] - E_{x\sim P_G}[D(x)]$$

When training is complete, the discriminator should not be able to distinguish between the data sampled from the real distribution and the generated data, so this expression should tend towards 0. We can therefore plot this loss as a function of epochs to visualize the training of our model.

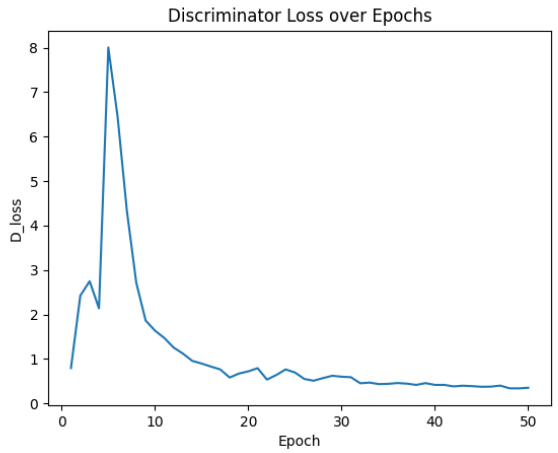

Figure 2: Loss of the discriminator of WGAN over epochs.

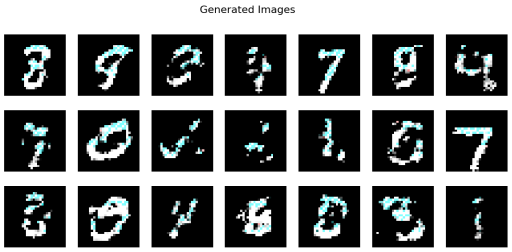

Figure 3: Digits generated by the WGAN.

We notice the learning curve shown in Figure 2 is unstable (with the pike around 8 epochs). Moreover, after 50 epochs, we obtained an FID = 69.8 and Figure 3 shows some generated digits whose appearance are not really satisfying. We supposed that the issues came from the fact that we ensured the 1-Lpischitz continuity of the discriminator by clipping its weights. Indeed, as mentionned in [1] weight clipping can lead to vanishing gradients if the clipping constant is too small or the model can fail to converge if it is too large.

3 WGAN with Gradient Penalty (WGAN-GP)

3.1 Overview of WGAN-GP

To tackle the issue of unstability during training and obtain better results, we then implemented another variant of the WGAN called WGAN with Gradient Penalty (WGAN-GP) [2]. This paper uses a new approach to ensure the 1-Lipschitz continuity of the discriminator by using gradient penalty instead of weight clipping. Below is the new loss of the disciminator:

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

This loss is based on the fact that a function is 1-Lipschitz continuous if and only if it has a gradient of norm less or equal 1 everywhere, so we add a term to ensure this constraint. We note that this term pushes the gradient towards 1 rather than constraining it below 1. However we remark empirically that it doesn't constrain the discriminator too much as mentioned in [2]. Figure 4 gives the pseudo-code of the algorithm that we implemented for the training of the generator and the discriminator of the WGAN-GP. In this algorithm, the gradient penalty is applied to a mixture of a real and a generated image:

$$\hat{x} = \epsilon x + (1-\epsilon)\tilde{x}$$

with x sampled from the real images, $\tilde{x}$ generated and $\epsilon \sim U[0,1]$. Moreover, we use Adam as optimizer instead of RMSProp for the initial WGAN.

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.
while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0,1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^m \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

Figure 4: Algorithm and hyperparameters for training D and G of WGAN-GP as described in [2].

3.2 Results and Analysis

Figure 5 shows the loss of the discriminator of WGAN-GP over epochs. We notice that the curve is much smoother than the one obtained with WGAN. Moreover, we managed to reach FID = 35.5, Precision = 0.45, Recall = 0.27 on the testing plateform after training our model for 100 epochs. FID is half that of the original WGAN, and the appearance of the digits generated is much more satisfying as shown in Figure 6. The downside of this method is that we must compute another gradient at each gradient update of the discriminator, which is computionally expensive and slows down the training of our model.

4 Rejection Sampling

4.1 Overview of Rejection Sampling

Rejection sampling is a method used to filter generated samples based on a specific threshold criterion, typically an evaluation metric, to improve the quality of accepted samples. The basic idea of rejection sampling is to generate candidate samples from a model and then accept only those that meet a predefined quality threshold. Mathematically, for a sample x generated from a distribution $p(x)$, it is accepted if it satisfies:

$$p(y|x) > \text{threshold}$$

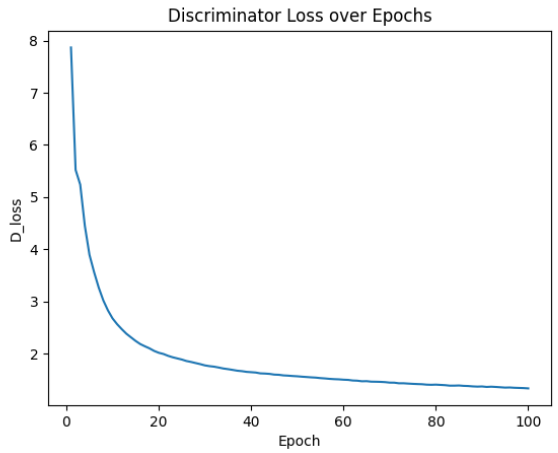

Figure 5: Loss of the discriminator of WGAN-GP over epochs.

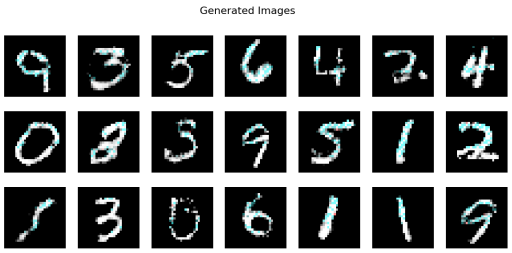

Figure 6: Digits generated by the WGAN-GP

where $p(y|x)$ represents the confidence score, or in this context, the quality score of the generated sample.

4.2 Our Rejection Sampling Method

In our method, we use a Convolutional Neural Network classifier trained on real MNIST images to evaluate the generated samples. To assess the quality of each generated sample batch, we compute the Inception Score (IS) based on the CNN's output. The Inception Score is a metric that measures both the diversity and quality of generated samples by analyzing the output probabilities across different categories. The score is computed using the following approach:

1. The generated samples are fed through a pre-trained CNN model, which outputs the conditional probability $p(y|x)$ for each image. 2. For a set of N generated images split into smaller groups (to reduce variance), the Inception Score is calculated by evaluating:

$$\text{IS} = \exp\left(E_x\left[\text{KL}(p(y|x) \parallel p(y))\right]\right)$$

where KL is the Kullback-Leibler divergence, $p(y|x)$ represents the conditional probability for a generated image x, and $p(y)$ is the marginal probability over all images.

4.3 Implementation of Rejection Sampling for Image Generation

To implement rejection sampling, we utilize a function that generates images in batches and evaluates their quality based on the Inception Score. Images with an Inception Score above a specified threshold are saved and retained for further evaluation, while batches that do not meet the score requirement are discarded. Below is an overview of the rejection sampling process used:

1. Batch Generation: A batch of images is generated using the GAN model.

2. Score Calculation: The Inception Score for the batch is computed using the CNN classifier.

3. Threshold Evaluation: If the score is above the defined threshold, the images in the batch are accepted and stored.

4. Batch Rejection: If the score does not meet the threshold, the batch is discarded.

Through this method, we selectively keep only the high-quality images generated by the GAN, thereby improving the overall quality of our sample set.

4.4 Results and Analysis

For our experiments, we set a threshold of 5.4 for the Inception Score, determined empirically.

Without any filtering, the generated images achieved an average Inception Score of 5.37. With a batch of 100 images, the FID score was calculated to be 64.24. This relatively high FID score can be attributed to the small batch size used (100 images instead of the 10,000 images used on the platform).

4.5 Impact of Varying Thresholds

Increasing the Inception Score threshold to 5.4 resulted in an improved mean Inception Score of 5.50 for the accepted images, and the FID score decreased to 62.22. Further experimentation showed that a threshold of 5.45 provided the best results with an FID score of 59.87.

Raising the threshold beyond 5.5 is impractical, as our GAN does not generate enough high-quality images at this level, making the generation time excessively long.

4.6 Evaluation with Larger Batches

Testing with a larger batch size (1,000 images) and setting the threshold to 5.45, we achieved an FID score of 24.43, which demonstrates the effectiveness of our approach in improving sample quality by filtering based on the Inception Score threshold.

4.7 Alternative using two adversarially-trained networks

We tried an alternative method for obtaining confidence scores on samples based on training a feed-forward model w^ϕ trained adversarially with the existing discriminator [4]. The idea behind this procedure is to redistribute the distribution of generated images closer to the real distribution, and thus to use the discriminator to evaluate the Wasserstein distance and train w^ϕ accordingly. The weights of the generation are kept frozen. Once the confidence score has been obtained, it can be used for rejection sampling as above, or through another method

proposed by the authors which consists in updating the samples z through gradient ascent prior to calling the generator as $x = G(z)$.

Unfortunately, we were not able to observe a significant difference between the scores of images that looked real and others. It is however likely that better hyperparameters could give satisfying results given that the authors were able to successfully apply their method to the MNIST dataset, albeit with a different discriminator architecture.

5 Computing metrics

We computed FID using the method described in the original paper [3], that is, by extracting embeddings on an intermediary layer of Inception-v3 and applying the formula described by the authors.

We implemented precision and recall using VGG-16 to produce embeddings and the k-nearest neighbors method with k = 3 [5]. However, this method proved very sensitive to k and to the number of samples, as already observed by the authors. Furthermore, this method is computationally expensive without an efficient algorithm. It is also made harder by the high dimensionality of the problem, which caused our initial kd-tree approach to be inefficient in this setting. Overall, we were unable to obtain a consistent algorithm for computing precision and recall.

Lastly, the use of Inception-v3 and VGG-16 networks is questionable given that those models are unnecessarily large and were trained on real pictures unlike the very simple MNIST digits. As an alternative, we trained a simple convolutional neural network classifying MNIST digits and extracted embeddings before the classification layer, with the aim of obtaining meaningful high-level features. This approach proved unsuccessful and did not provide in consistent results.

6 Conclusion

In this project, we evaluated multiple approaches to improve the quality and reliability of GAN-generated images. Starting with a vanilla GAN, we observed significant improvements in both visual quality and quantitative metrics through the implementation of WGAN and WGAN-GP. These improvements were enhanced further with rejection sampling, using an Inception Score threshold to filter generated samples, which successfully reduced the FID score.

Through experimentation, we found that setting an appropriate Inception Score threshold yielded a substantial improvement in sample quality, though higher thresholds increased generation time. Moreover, using larger batches for FID calculations provided more stable results, demonstrating the model's potential for high-quality image generation. Although computational constraints limited some aspects of the analysis, the results demonstrate the effectiveness of the WGAN-GP and rejection sampling methods in generating realistic samples.

Future work could explore optimizing the balance between quality and generation speed, as well as adapting these techniques to other datasets and applications.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[2] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017.

[4] Thibaut Issenhuth, Ugo Tanielian, David Picard, and Jeremie Mary. Latent reweighting, an almost free improvement for gans. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, page 3574–3583. IEEE, January 2022.

[5] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models, 2019.