Data Science Lab

Learning latent space representations

Generative Adversarial Networks

Groupe GANERGY

Elhadi CHITER elhadi.chiter@dauphine.eu
Othman HICHEUR othman.hicheur@dauphine.eu
Fatma-Zohra REZKELLAH fatma-zohra.rezkellah@dauphine.eu

Submission Date : November 12, 2024

2024/2025 - Semester I

Contents

1 Introduction

A generative adversarial netwrok (GAN) is an unsupervised technique that aims to generate samples of images it was trained on. It's composed of a first network that is known as the *the generator*, that maps random noise to the output data space. The second network is known as *the discriminator*, that plays the role of an investigator (given a real and a fake image, it should be able to classify them correctly). Hence, this creates an adversarial learning where the generator tries to fool the discriminator by generating plausible images. The idea seems to be simple, however training GANs is difficult: the learning algorithm can be unstable, and although GANs may learn to generate realistic samples, this does not imply that they learn to generate all possible samples. This is known as *mode dropping* and an extreme version of it is *mode collapse* (the generator entirely or mostly ignores the latent representation and collapses all samples to one or a few points). [4]

This assignment tackles these limits by introducing new architectures and new losses to train GANs. We mainly focused on three approaches: playing with the architecture of the networks, modifying the loss function and initializing the random noise differently. In the coming sections, we will begin by motivating our work, then we will delve into the new models we've studied: WGAN, SAGAN & GMGAN and finally conclude with results.

2 Motivation

GAN training is characterized by a minimax game between the generator and the discriminator, where the generator tries to find new ways to fool the discriminator (minimizes its loss), which in return searches for new ways to distnguish fake images from real ones (maximizes its loss).

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} [\log (1 - D(G(\mathbf{z})))]$$

An optimal discriminator D^*, if the generator is fixed, is defined as the probability of a sample x to be from the real distribution rather than the generated one (We get the optimal discriminator by differentiating the discriminator objective function and then solving for it when it's zero).

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_z(x)}$$

By substituting this optimal discriminator value in the discriminator loss, we will see that its becomes ***the Jensen Shanon divergence between the real distribution and the generated distribution***.

However, using this divergence as a criterion to optimize GANs might be a potential problem, due to the fact that If the probability distributions are completely disjoint, then this distance is infinite and the discriminator can perfectly separate generated and real samples from each other. In this scenario, the gradients used to update the generator become very small (or vanish entirely).

Unfortunately, in most cases, the distributions of generated samples and real examples may really be disjoint. There may be little or no overlap between their subspaces, and the result is very small or no gradients. [4]

3 GANs

3.1 Vanilla GAN

In this section, we'll show you the results we obtained from the Vanilla GAN. We also tried adding Batch Normalization layers to the generator since many of the research papers we've read used it, so we decided to try it out as well.
The training hyperparameters are as follows: **Learning rate:** 0.0001, **Batch size:** 64, **Optimizer:** Adam.

Quantitative comparison The quantitative comparison of the two models, on the three metrics, is presented in Table 1.

	FID	**Precision**	**Recall**
Vanilla GAN	29.28	0.54	0.23
Vanilla GAN Batch Norm	26.68	0.54	0.19

Table 1: Comparison of GAN models based on FID, Precision, and Recall.

As seen in the table, the addition of batch normalization resulted in a lower FID score, indicating improved image quality. However, precision remained constant, and recall slightly decreased, suggesting that while the batch-normalized GAN produces higher-quality images, the diversity of the samples did not improve.

3.2 Wasserstein GANs

In order to mitigate the limits of the Jensen Shanon distance, the authors in the Wasserstein GAN paper [1] proposed to use the wasserstein distance to compute the loss of the GAN. The Wasserstein or (for discrete distributions) earth mover's distance is the quantity of work required to transport the probability mass from one distribution to create the other. The Wasserstein distance is well-defined even when the distributions are disjoint and decreases smoothly as they become closer to one another [4].

As a result, the discriminator's (also called the critic's) loss is defined this way, where it's goal is to maximize the difference between the expected values of its output for real and generated samples.

$$L_D = -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))]$$

To ensure the critic (Discriminator) loss function remains 1-Lipschitz continuous (required for Wasserstein distance), the authors in [1] use weight clipping of its parameters.
The generator's objective is to maximize the critic's output for generated samples, effectively minimizing the Wasserstein distance.

$$L_G = -\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))]$$

Another way to ensure that the discriminator loss function remains 1-Lipschitz continuous is using gradient penalty that was introduced in the paper [3].

	FID	Precision	Recall
WGAN	58.41	0.5	0.27
WGAN GP	40.45	0.51	0.32

Table 2: Comparison of WGAN models based on FID, Precision, and Recall.

3.2.1 Wasserstein GAN Gradient Penalty

Quoting from the original paper of WGAN [1] : "Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing gradients when the number of layers is big, or batch normalization is not used."

As a result, the loss function for WGAN gradient penalty includes a term that penalizes the discriminator when its gradients norm deviates from 1.

$$L_D = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))] + \lambda \cdot \mathbb{E}_{\hat{\mathbf{x}}\sim \mathbb{P}_{\hat{\mathbf{x}}}}\left(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1\right)^2]$$

where $\hat{x}$ is an interpolated sample between real and generated data points.
In table 2, we can see that indeed WGAN improves the diversity of the generated images (the recall is larger than the one of the vanilla GAN). In figure 1, we can see how WGAN gradient penalty stabilizes the training. The discriminator loss (which represents the wasserstein distance between the real and generated distribution) approches zero. As for the generator loss, we can also see it's minimized as we move in epochs.

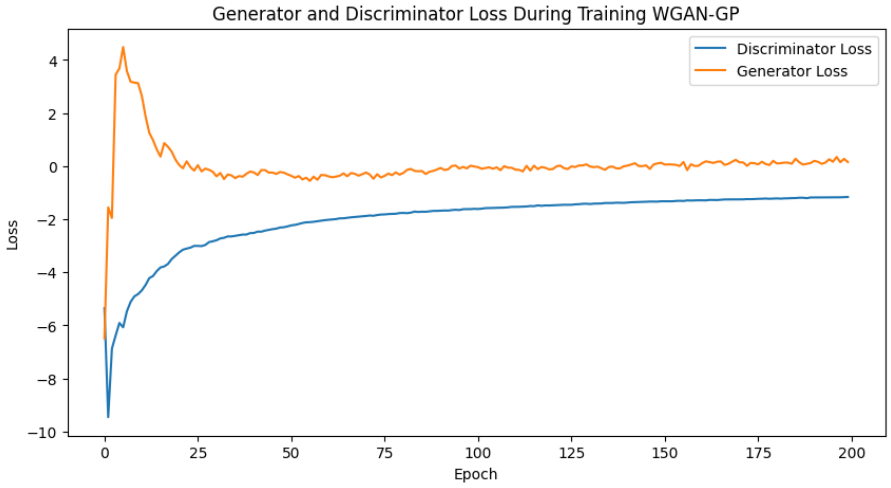

Figure 1: Plots of the generator and discriminator loss in the WGAN gradient penalty.

3.3 Self Attention GANs

3.3.1 Structure and specificity

An other way to stabilize the training of GAN is the use of spectral normalization on the discriminator and of two different learning rates for generator and discriminator [5] in

order to compensate slow learning of the regularized discriminator (two-timescale update rule). Moreover, we can use an attention layer at the end of the discriminator [5].
Given an input x for the attention layer, we compute the query $Q(x) = W_q x$, the key $K(x) = W_k x$, the and value $V(x) = W_v x$, where W_k, W_q and W_v are learned parameters. Then, we use a dot product between $Q(x)$ and $K(x)$ and we apply a softmax, the result is α. Finally, we compute $O = \alpha \cdot V(x)$ and return $y = \gamma \cdot O + x$, where γ is trainable parameter [5].

3.3.2 Loss Functions

For self attention GAN, we use the same loss for the generator as before :

$$\mathcal{L}_G = -\mathbb{E}_{z \sim p_z}[D(G(z))] \tag{1}$$

However, according to [5], we use the hinge version of the adversarial loss for the discriminator :

$$\mathcal{L}_D = \mathbb{E}_{x \sim p_{\text{data}}}[\max(0, 1 - D(x))] + \mathbb{E}_{z \sim p_z}[\max(0, 1 + D(G(z)))] \tag{2}$$

3.3.3 Results

In this part, we compare self attention GAN [5] with batch normalization on the generator and attention layer on the discriminator, and vanilla GAN with a hinge loss on the discriminator, batch normalization on the generator and without attention layer. For all of them, we use the same hyperparameters : **Learning rate generator:** 0.0001, **Learning rate discriminator:** 0.0004, **Batch size:** 64 and **Optimizer:** Adam.

Quantitative comparison Here is the comparison of the two architectures, with different epochs for the Self Attention GAN. It is important to note that the results for the vanilla GAN are from our proper experiments and are not available on the platform, we evaluated these models ourselves.

	FID	**Precision**	**Recall**
Vanilla GAN Batch Norm Hinge loss	20.98	0.51	0.47
$\text{SAGAN}_{200epochs}$	18.19	0.77	0.58

Table 3: Comparison of GAN models based on FID, Precision, and Recall for SAGAN and Vanilla GAN.

As seen in the previous table, the use of the hinge loss on the discriminator increased the recall and decreased the FID. Moreover, the addition of attention layer at the end of the discriminator improved the results again, especially over the precision which reached 0.77 for 200 epochs. All of these observations have shown that with attention layer and hinge loss on the discriminator, the quality of the generated images increased, as well as their diversity, while converging closer to the initial distribution.

Loss dynamics As it presented in the figure 2, the generator loss goes to 0 and the discriminator loss decreased with the number of epochs.

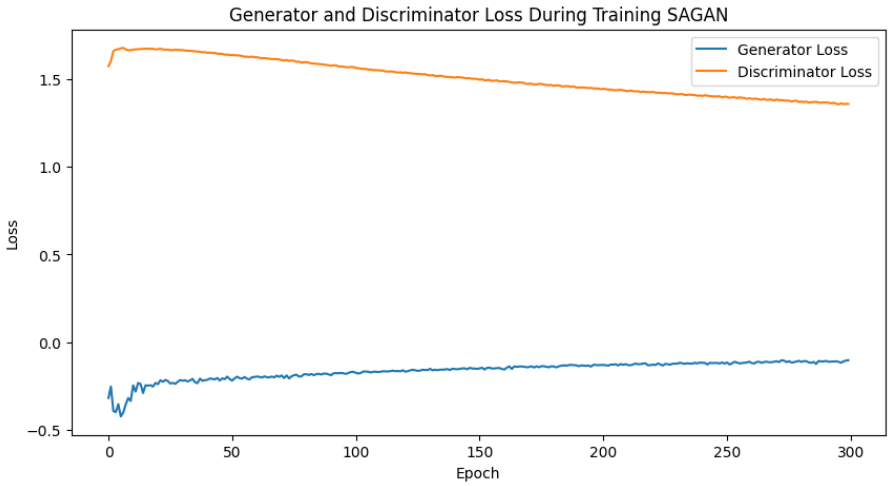

Figure 2: Plots of the generator and discriminator loss in the Self Attention GAN.

3.4 Gaussiane Mixture GANs

In order to test how the initialization of the random noise input of the generator affects the generations, we decided to implement a gaussian mixture initialization inspired from [2]. We implemented the unsupervised static Gaussian Mixture GAN. By static we mean that the parameters of the mixture of Gaussians distribution are fixed before training the model, and cannot change during the training process.

The unsupervised GM GAN has the same loss as the Vanilla GAN.

	FID	**Precision**	**Recall**
GM GAN	41.01	0.85	0.48

Table 4: GM GAN Results

The results in table 4 were obtained from a 5 gaussians mixture, with an ADAM optimizer and learning rate equal to $5 * 10^{-4}$. We can see that the precison and recall are quiet high compared to the other models. This suggest that the initialization of the random noise plays a role in the image generation.

4 Visual Results

Figure 3 shows the different samples we've got from our different tested models. As you can see in image (a), generated images of the Vanilla GAN are of a plausible quality, however they lack diversity. As for image (b), which corresponds to images generated from the WGAN-GP, we can see that the images are diverse, however the quality is not that good. The images generated from the SAGAN, which are highlighted in image (c), are the best generated samples, the model was able to generate good quality samples while keeping them diverse. In final, images generated from the GM-GAN in image (d) have a plausible quality but lack a bit of diversity (diversity with reference to the generated numbers).

((a)) Generated Image of VGAN

((b)) Generated Image of WGAN-GP

((c)) Generated Image of SAGAN

((d)) Generated Image of GM-GAN

Figure 3: Generated MNIST samples from different GANs

5 Conclusion

In this report, we have studied and evaluated different implementations of GANs for image synthesis and tackled their limits such as training instability, mode dropping and mode collapse. We also investigated the quality-diversity trade-of of the generated samples. We first explored Wasserstein GANs that improved training stability and mitigated mode dropping by generating good quality diversified images. Therefore, self attention was adopted in the discriminator architecture under self attention GANs as we believed that it will help capture details across different parts of the image to better distinguish real images from fake ones. Indeed, it was able to generate very high quality and pretty diversified samples. Next, we focused on random noise input initialization of the generator by studying Unsupervised Static Gaussian Mixture GANs that confirmed our hypothesis that the random noise initialization plays a role in improving the quality and the diversity of the synthetic images. To sum up, this project made us realize how challenging the training of GANs can be and how sensitive it is to hyperparameters' choice, a misstep can lead to total chaos.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Matan Ben-Yosef and Daphna Weinshall. Gaussian mixture generative adversarial networks for diverse datasets, and the unsupervised clustering of images. *arXiv preprint arXiv:1808.10356*, 2018.

[3] Ishaan Gulrajani, Gintare Karolina Dziugaite, Nikos Komodakis, and Eric Tzeng. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[4] Priya L. Mehta and Trevor Darrell. *Understanding Deep Learning: From Theory to Practice*. MIT Press, 2023.

[5] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2019.