Supervised by Benjamin Negrevergne and Alexandre Vérine

Assignment 2
Learning the latent representations (GANs)

GANgineers

2024-2025

NEAU Matthieu

CHATZILOZOS Efstathios

KASMI Abderrahmane

I. WGANs Overview

Motivation

When training Vanilla GANs, two common problems can arise: Mode Collapse and Instability at train time which may lead to non convergence of the model

1. Mode Collapse

Mode Collapse occurs when the generator manages to fool the discriminator in a narrow way.

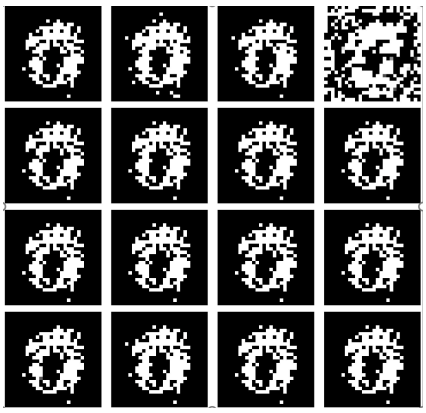

Figure 1: Mode Collapse example in GAN training. Obtained for 170 epochs on Vanilla GAN,

2. Instability

Instability refers to oscillations in GAN training, where the generator and discriminator repeatedly fail to converge.

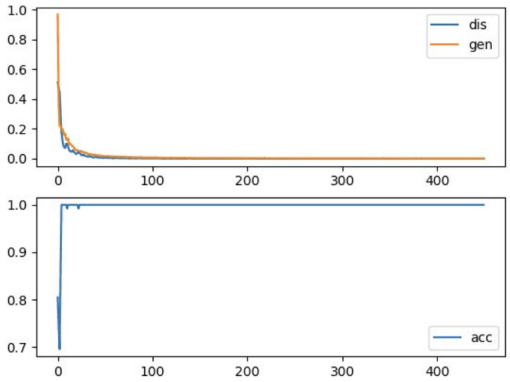

Figure 2: Instability during GAN training [1]

Wasserstein GANs make use of the Earth Mover distance, of which we use the dual formulation:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} E_{x \sim p_r}[f(x)] - E_{x \sim p_g}[f(x)]$$

This distance induces a weaker topology (i.e. distributions converge more easily with respect to that distance). The Wasserstein duality criterion is enforced by WGANs by clipping the discriminator's weights. Weight clipping is very delicate as mentioned in [2]:

> "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 (such as in RNNs)."

We therefore suggest to study the enforcement of weight clipping on the discriminator to try to set an optimal value for our task.

II. Discriminator Weight Clipping

The methodology is to plot the discriminator's weights before clipping on at various points in the training process. We consider that the clipping value is appropriate if a reasonable fraction of the weights saturate (i.e. need to be clipped). This shows that the clipping value is neither too small nor too big. All the weights exceeding

As we are using Pytorch's default Kaiming Uniform initialization, where the weights are initialized by:

$$W \sim \mathcal{U}\left(-\sqrt{\frac{6}{\text{fan\_in}}}, \sqrt{\frac{6}{\text{fan\_in}}}\right)$$

and our layers have dimensions around 1000, this gives us an order of magnitude to start adjusting our clipping value at 10^{-2}.

Observation
We note that the deeper layers tend to saturate quicker. We hypothesize this is due to the LeakyReLU activation function which reduces the updates during backpropagation as the slope is set to 0.2 on the negatives

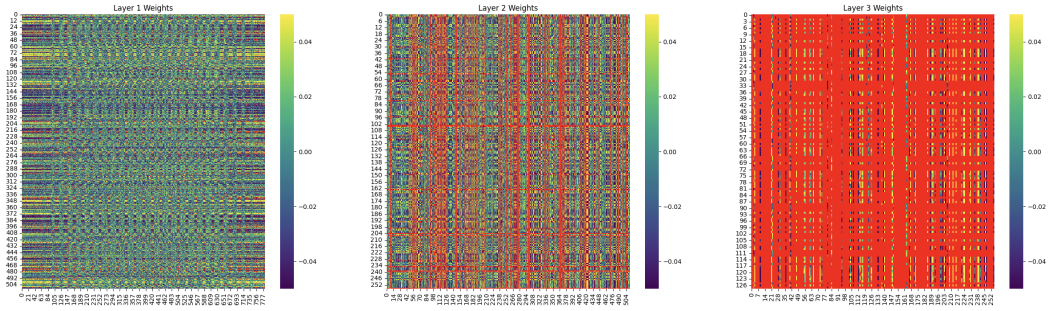

Figure 3: Illustration of saturation in deeper layers with a clipping value set to 0.05 at 5 epochs

This phenomenon is observed at various epochs and common to all the different clipping values (see here).

Choosing Appropriate Clipping value
We plot the weight activations at 15 epochs, an arbitrary but reasonable number where any warm up effects should have worn out.

At 15 epochs, we plot activations for clipping values in the range $[0.01, 0.05, 0.1]$ and we note that the activations have a reasonable amount of saturation for 0.01, which we keep for further experiments. All the plots can be found (here).

III. Gradient Penalty

The original WGAN introduced the Earth Mover (Wasserstein) distance to address instabilities in traditional GANs, enforcing a Lipschitz continuity constraint on the discriminator (critic). How-

ever, this constraint was implemented through weight clipping, which introduces several challenges like **limiting the critic's capacity, vanishing/exploding gradients and artifacts in generated samples.**

Gradient Penalty Mechanism
To overcome these issues, WGAN-GP replaces weight clipping with a gradient penalty [4] in the critic's loss. This approach penalizes the model if the gradient norm deviates from 1, ensuring smooth, stable training and enhancing sample quality. The gradient penalty term is defined as:

$$\text{Gradient Penalty} = \lambda_{\text{GP}} E_{\hat{x}} \left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2 \right]$$

where $\hat{x}$ represents interpolated samples and λ_{GP} controls the penalty's strength. The **Gradient Penalty** is then added to the critic's loss

Advantages of Gradient Penalty

- **Full Capacity Utilization:** The discriminator can now fully exploit its capacity to distinguish between real and fake samples.
- **More Stable Training:** Encouraging the gradient norm to be close to 1 results in smoother updates and reduces issues related to vanishing or exploding gradients.
- **Improved Sample Quality:** A better-trained discriminator provides more informative gradients to the generator, which can produce higher-quality samples that are more indistinguishable from real data.

IV.Discriminator Rejection Sampling

In GANs, once training is complete, the discriminator is typically discarded, leaving only the generator for sample generation. However, the discriminator has insights about the quality of the generated samples. Discriminator Rejection Sampling (DRS) [3] utilizes the discriminator post-training to improve the quality of generated samples without retraining the model.

Given that the discriminator's output reflects how "realistic" a sample is, we can apply rejection sampling based on these scores and DRS will selectively filter out lower-quality samples according to the discriminator's judgement.

Algorithm Steps

1. **Generate Samples:** Use the trained generator to produce a batch of fake samples.
2. **Compute Discriminator Scores:** Pass the generated samples through the discriminator to obtain logits (confidence scores).
3. **Calculate Acceptance Probabilities:** Determine the probability of accepting each sample based on its discriminator score. A typical approach is to apply the sigmoid function with a threshold τ:

   $$p_{\text{accept}} = \sigma(D(x_{\text{fake}}) - \tau)$$

   where σ is the sigmoid function.

4. **Accept or Reject Samples:** For each sample, draw a random number from a uniform distribution between 0 and 1. Accept the sample if this number is less than the computed acceptance probability.
5. **Iterate:** Repeat until the desired number of samples is obtained.

V. Optimal Transport

General principle

When we have a 1-Lipschitz continuous discriminator (or an approximation thereof), we can enhance the quality of generation using optimal transport [5]. Based on the dual formulation of the Wasserstein distance, the optimal transformation that maps $y \sim p_g$ to $x \sim p_r$ is given by:

$$T(y) = \arg\min_x \{\|x - y\|_2 - D(x)\}$$

This approach seeks the x that has the highest discriminator score within the neighborhood of the generated sample y. This method can be applied in two spaces:

- **Latent space**: on the noise vector z
- **Target space**: directly on the generated sample y

The algorithm below demonstrates how to use gradient descent to apply optimal transport in the target space:

Algorithm 1 Target space optimal transport by gradient descent

Require: trained D (approximately 1-Lipschitz continuous), sample y, learning rate ϵ and small vector δ
  Initialize $x \leftarrow y$
  for n_{trial} in range(N_{updates}) **do**
    $x \leftarrow x - \epsilon \nabla_x \{\|x - y + \delta\|_2 - D(x)\}$ *(δ is for preventing overflow.)*
  return x

Results

The following results demonstrate the application of optimal transport in the target space of a WGAN with gradient penalty (Optimal transport was applied for 20 epochs with a learning rate of 0.01):

Figure 4: Comparison of generated samples

	WGAN-GP without OT	WGAN-GP with OT
FID	24.12	22.64

Figure 5: FID scores for WGAN-GP with and without optimal transport

VI. Implementation Details

This section describes the configuration for our best-performing model, which incorporates both gradient penalty (GP) and discriminator rejection sampling (DRS) (generate.py). **This model currently holds the number 1 spot on the evaluation platform.**

Training Configuration (train.py and utils.py)
For the generator, we set a learning rate of 1×10^{-3} with $\beta_1 = 0.5$ and $\beta_2 = 0.9$; the discriminator used a learning rate of 3×10^{-4} with the same β values. Training was conducted with a batch size of 64 for 200 epochs, saving checkpoints every 10 epochs.

To enforce the Lipschitz constraint, we incorporated a gradient penalty coefficient $\lambda_{\text{GP}} = 10$ in the discriminator's loss. Both networks were updated equally (one generator update per discriminator update). Xavier initialization was used for all weights to maintain gradient flow and the discriminator's final layer had no activation to allow raw score outputs for the Wasserstein distance (model.py).

Fine-Tuning
After 200 epochs, we identified a balanced checkpoint at around 120 epochs where the generator was not easily fooling the discriminator and the discriminator was not overconfident. From this checkpoint, we conducted 20 additional fine-tuning epochs with reduced learning rates (5×10^{-4} for the generator, 1.5×10^{-4} for the discriminator), leading to improved sample quality.

Evaluation Metrics
We evaluated generated samples FID; lower FID scores indicate closer alignment with the real data distribution. Additionally, we measured precision and recall to assess both quality and diversity of generated samples.

VII. Evaluation Platform Results

Our WGAN-GP model with Discriminator Rejection Sampling (DRS) achieved top performance on the platform, holding the number 1 spot with an FID score of 17.02, precision of 0.8, and recall of 0.55. The following figure shows a sample of the generated images and the performance metrics from the platform.

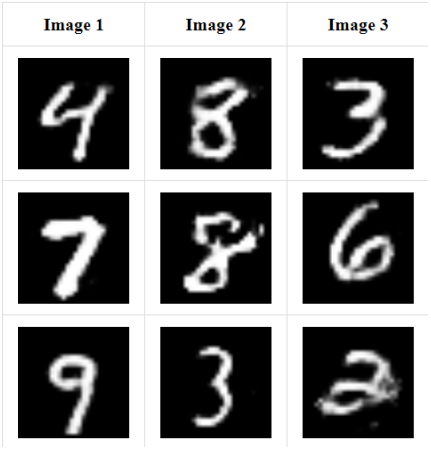

Figure 6: Generated samples on the platform

Project Name	Time	FID	Precision	Recall
gangineers	75.0	17.02	0.8	0.55

Figure 7: Platform performance metrics (Gangineers)

References

[1] Sudarshan Adiga, Mohamed Adel Attia, Wei-Ting Chang, and Ravi Tandon. On the tradeoff between mode collapse and sample quality in generative adversarial networks. *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1184–1188, 2018.

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

[3] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling, 2019.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

[5] Akinori Tanaka. Discriminator optimal transport. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.