Deep Learning Practical Work 2-de

William KHIEU
Akli Mohamed Ait-Oumeziane

December 24, 2024

Contents

Section 1 - Generative Adversarial Networks

1.1 General Principle

1. Interpret the equations (6) and (7). What would happen if we only used one of the two ?

- Equation (6) describes the objective function of the **generator** G. The generator aims to produce fake images $\tilde{x} = G(z)$ such that the discriminator D classifies them as real. The generator thus maximizes:

$$\mathbb{E}_{z\sim P(z)}[\log D(G(z))],$$

  which encourages $D(G(z)) \to 1$. If successful, this means G has generated images that are indistinguishable from real data.

- Equation (7) describes the objective function of the **discriminator** D. The discriminator tries to maximize its ability to distinguish real images x^* (from the dataset) and fake images $\tilde{x} = G(z)$ (produced by the generator). It does so by maximizing:

$$\mathbb{E}_{x^*\in \text{Data}}[\log D(x^*)],$$

  which encourages $D(x^*) \to 1$, and:

$$\mathbb{E}_{z\sim P(z)}[\log(1 - D(G(z)))],$$

  which encourages $D(G(z)) \to 0$.

If we only used one of the two equations:

- If only G is trained (Equation 6), The generator G would have no feedback about whether its generated images are realistic since D is not being updated to challenge it. This leads to G producing poor-quality images without improving.
- IIf only D is trained (Equation 7), The discriminator would become overly powerful and perfectly classify real and fake images. However, G would not improve, so the system would stagnate, and $G(z)$ would remain unrealistic.

2. Ideally, what should the generator G transform the distribution $P(z)$ to ?

The generator G should transform the distribution $P(z)$, which is typically $\mathcal{N}(0, I)$, into the data distribution $\mathcal{D}ata$. Ideally, the mapping $z \to G(z)$ should ensure that the generated samples $\tilde{x}$ are indistinguishable from real samples x^* in the dataset. In other words:

$$G(P(z)) \approx \mathcal{D}ata.$$

When this is achieved, D will not be able to distinguish between x^* and $\tilde{x} = G(z)$, meaning $D(x) \approx 0.5$ for all x.

If G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$.

3. Remark that the equation (6) is not directly derived from the equation 5. This is justified by the authors to obtain more stable training and avoid the saturation of gradients. What should the "true" equation be here ?

The "true" equation for the generator, derived directly from Equation (5), would be:

$$\min_G \mathbb{E}_{z\sim P(z)}[\log(1 - D(G(z)))].$$

This equation makes sense because the generator's goal is to minimize $D(G(z))$, meaning it wants the discriminator to classify $G(z)$ as real (ideally, $D(G(z)) = 1$).

However, using this equation can cause training instability because when $D(G(z))$ is very small (the discriminator strongly classifies $G(z)$ as fake), the gradients of $\log(1 - D(G(z)))$ saturate. This leads to very slow updates for G.

To address this, the authors modify the generator's objective (Equation 6) to maximize:

$$\mathbb{E}_{z\sim P(z)}[\log D(G(z))].$$

This reformulation avoids the saturation issue by providing more informative gradients for G, ensuring more stable and efficient training.

1.2 Architecture of the networks

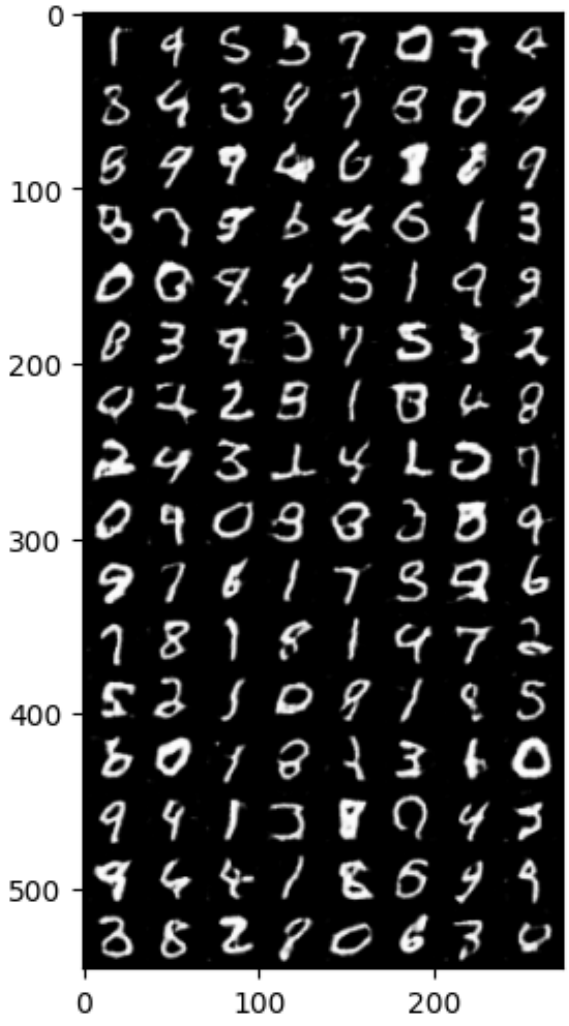
4. Comment on the training of of the GAN with the default settings (progress of the generations, the loss, stability, image diversity, etc).

We can observe that, as training progresses, the generated digits seem more and more realistic. However, the loss values exhibit significant instability, oscillating across epochs, which is a common trait in GAN training, indicating their tendency not to converge in practice.

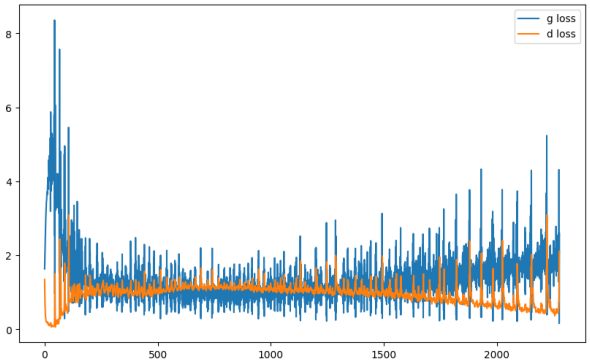
While the generator does improve image quality over time, the generated results still appear noticeably synthetic and limited in diversity, primarily resembling digits such as 0, 4, 6, and 9. This observation raises concerns about a potential *mode collapse* issue, where the generator learns to produce only a subset of possible outputs that are sufficient to deceive the discriminator but lacks overall diversity.

5. Comment on the diverse experiences that you have performed with the suggestions above. In particular, comment on the stability on training, the losses, the diversity of generated images, etc

- **Increasing the number of epochs:** Increasing the number of epochs lead to more instability in the training, with spikes in the loss values and oscillations in the generated images. Toward the end, the background of the images becomes more noisy.
- **Reducing n_z:** Reducing the dimensionality of the noise vector z led to a reduction in diversity among the generated outputs.
- **Increasing n_z:** Increasing the dimensionality of the noise vector z actually reduced the quality of the generated images. The images were more noisy and less realistic. This could be due to low number of epochs since it was kept at 5.

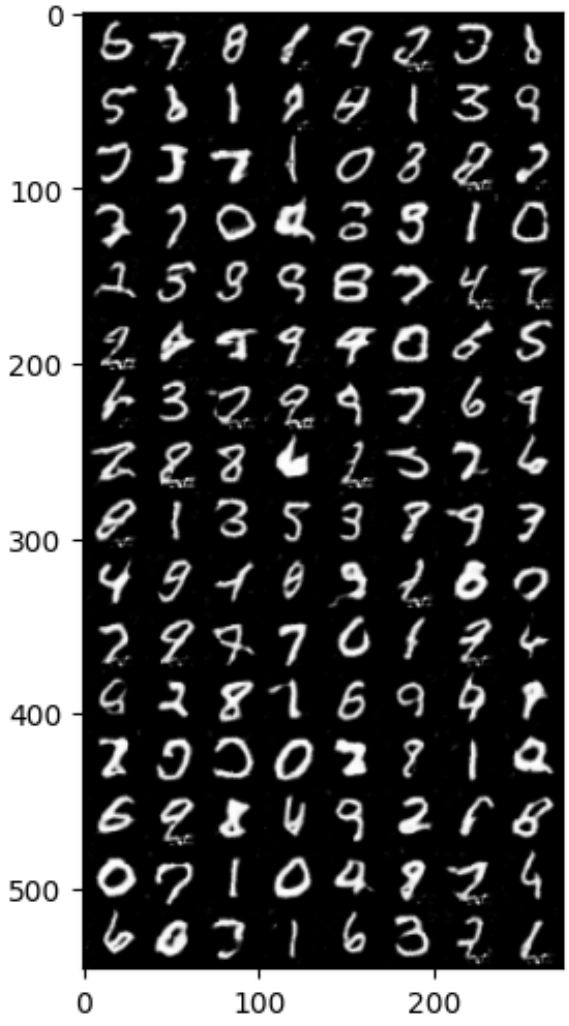


(a) GAN Base Image

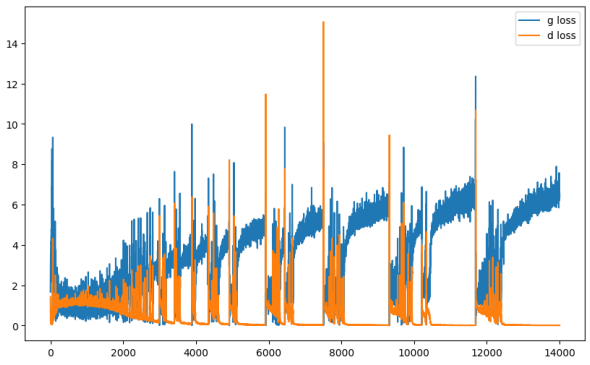


(b) GAN Base Curve

Figure 1: GAN unconditional and its curve.

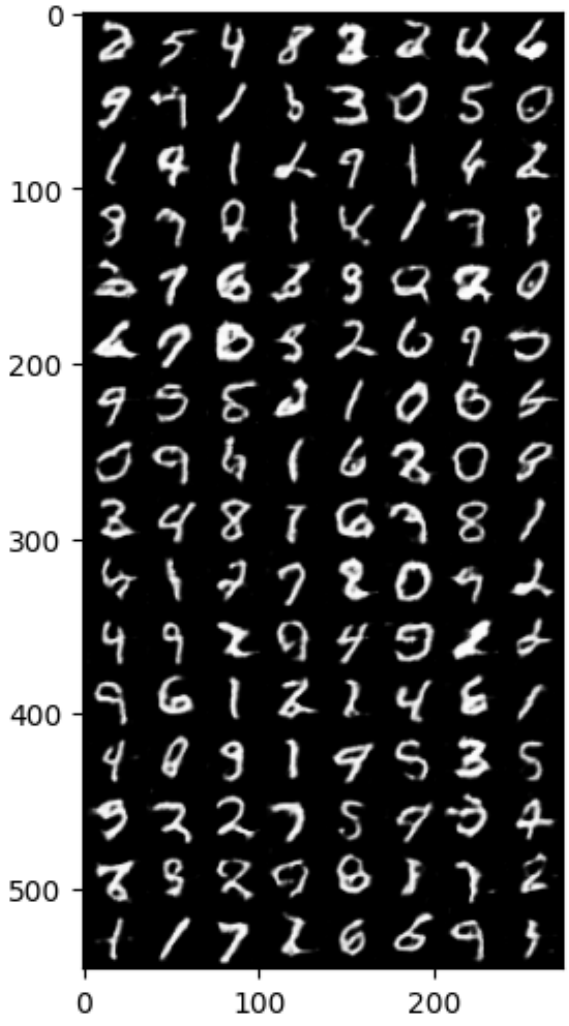


(a) GAN epoch 30 Image

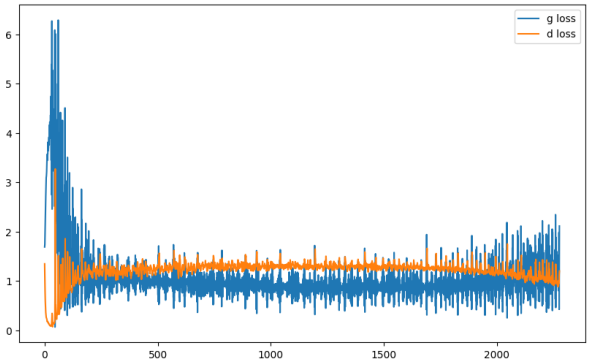


(b) GAN epoch 30 Curve

Figure 2: GAN unconditional with 30 eopch and its curve.

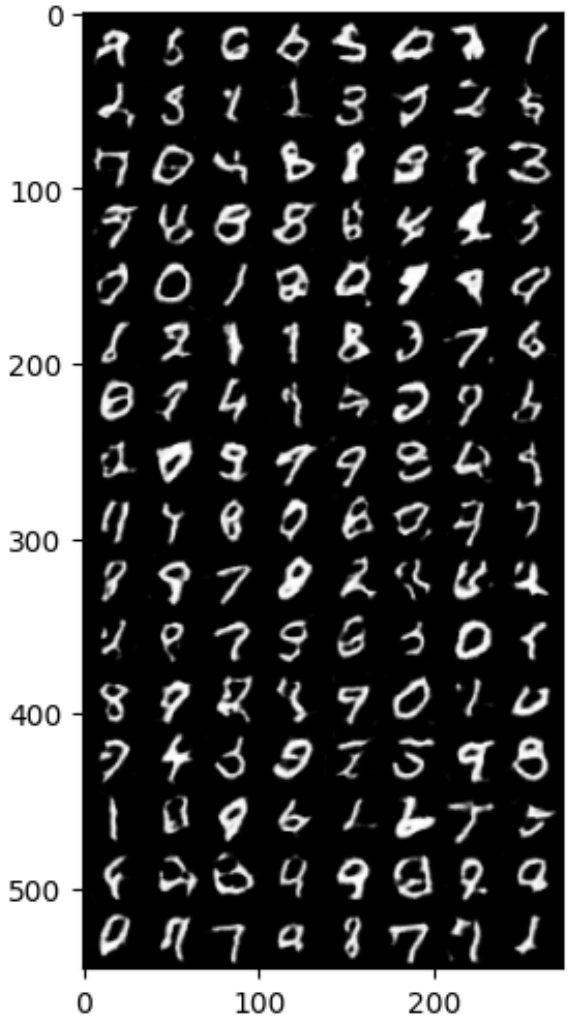


(a) GAN nz=10 Image

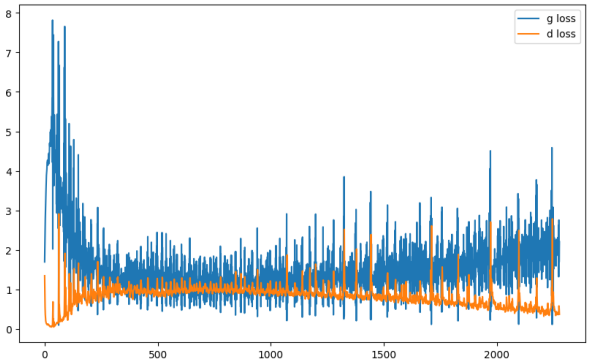


(b) GAN nz=10 Curve

Figure 3: GAN unconditional with nz=10 and its curve.



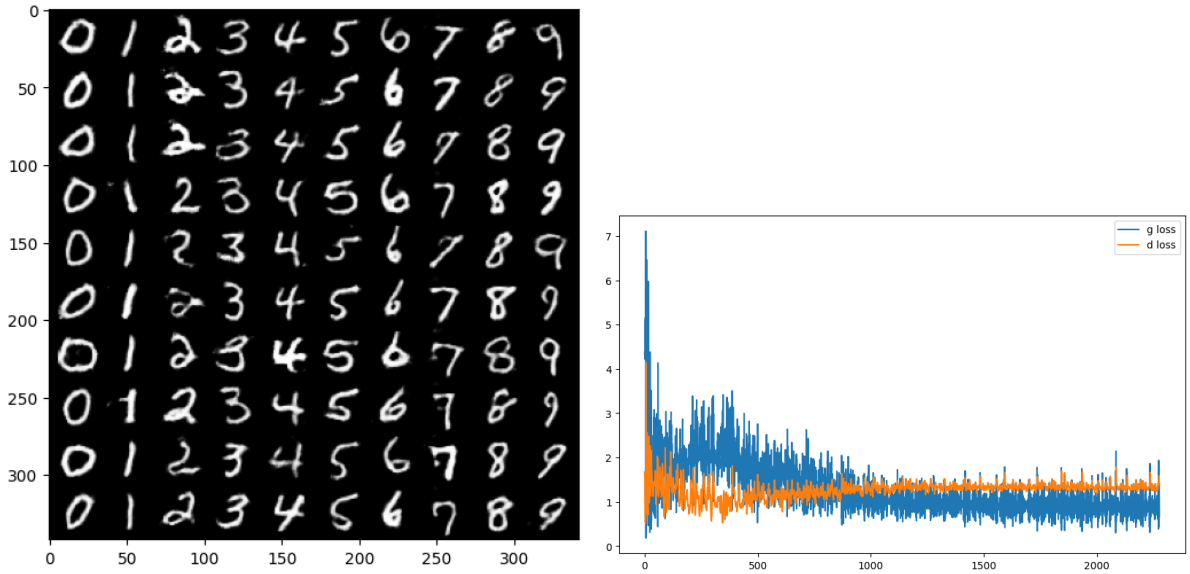
(a) GAN nz=1000 Image



(b) GAN nz=1000 Curve

Figure 4: GAN unconditional with nz=1000 and its curve.

Section 2 - Conditional Generative Adversarial Networks



(a) Conditional GAN Image

(b) Conditional GAN Curve

Figure 5: GAN conditional and its curve.

6. Comment on your experiences with the conditional DCGAN.

Like in the unconditional case, the conditional DCGAN exhibits a similar pattern of training instability, with oscillating loss values and slow convergence. However, the conditional model appears to achieve slightly better results in terms of image quality and diversity. The generated images are more realistic and varied, showing a broader range of digits and styles compared to the unconditional model (all digits are generated). This improvement is expected since the conditional model can leverage additional information (the class labels) to guide the generation process, leading to more accurate and diverse outputs.

7. Could we remove the vector y from the input of the discriminator (so having $cD(x)$ instead of $cD(x, y)$) ?

No, removing y from the input of the discriminator fundamentally changes the nature of the conditional GAN. The purpose of y in $cD(x, y)$ is to provide the discriminator with information about the condition the generator was supposed to adhere to. If $cD(x)$ is used instead of $cD(x, y)$, the discriminator will no longer check whether the generated image $\tilde{x}$ adheres to the condition y. Instead, it will evaluate images purely on their realism without considering whether they match the intended attributes. This undermines the conditional GAN's ability to learn the desired conditional mapping and would likely lead to poor performance in generating images conditioned on y.

8. Was your training more or less successful than the unconditional case ? Why ?

The training of a conditional GAN is more successful than the unconditional case in terms of achieving targeted generation. This is because the conditional input y provides additional in-

formation that guides the generator towards creating specific outputs that match the desired conditions.

In the unconditional case, the generator is tasked with modeling the entire data distribution without any guiding structure, which can make training more challenging. By contrast, the conditional GAN reduces the complexity of the task by breaking it into simpler conditional distributions, one for each y. This focused guidance improves the quality of generated images and their adherence to specific attributes.

9. Test the code at the end. Each column corresponds to a unique noise vector z. What could z be interpreted as here ?

In this context, z can be interpreted as a **latent representation** or **style vector** that captures high-level, abstract variations in the generated images that are **independent of the condition** y. For example, in the case of MNIST digits:

- Each **row** of the generated images corresponds to a single z, where the same noise vector generates digits of all classes y.
- Within each row, the condition y specifies the digit class (e.g., 0 through 9).
- z can therefore represent variations such as the digit's stroke thickness, orientation, or writing style, which are independent of the digit class y.

This separation allows the GAN to disentangle the class-specific information y from the style-specific information z, leading to more controlled and interpretable image generation.