Homework 3 report

Machine Learning for Signal Processing

Institute of Computer Science and Engineering

06/03/2024

Supervised by :
Do Huu Phu
dohuuphu25.ee11@nycu.edu.tw

Produced by :
PAULY Alexandre
alexandre.pauly@cy-tech.fr

Contents

1 Introduction

Context:

As part of the Machine Learning for Signal Processing module, an assignment focusing on generative models has been initiated. This assignment is designed to reinforce the theoretical concepts covered within the module by implementing and exploring Generative Adversarial Network (GAN) and Denoising Diffusion Probabilistic Model (DDPM) algorithms in Python. The primary goal is to implementing these techniques without relying on pre-existing libraries, thereby deepening understanding and mastery of the underlying principles of generative models.

Objective:

In this report, we focus on understanding and implementing GAN and DDPM from scratch. The main objective is to explain in detail each step of the implementation of these two techniques, including the underlying mathematical principles, computational procedures, and visualization techniques. Moreover, we will demonstrate the application of GAN and DDPM on the MNIST dataset, a collection of handwritten digits, by plotting the original data distribution and the resulting generations. Through this comparative analysis, we seek to elucidate the distinctive characteristics, advantages, and limitations of these two methods.

2 Dataset

The data used in this study is derived from the MNIST dataset, which is extensively employed for handwritten character recognition and image classification tasks. This dataset comprises images of handwritten digits from zero to nine, each represented as a matrix of grayscale pixels. These images serve as digital representations of handwritten digits captured under various conditions, including diverse writing styles, different sizes, and varying levels of noise. In this study, we will use 60 000 images from this MNIST.

Figure 1: Examples of data from the MNIST dataset

To implement the generation methods, the data are resized, converted to PyTorch tensors and normalized to have a mean of 0.5 and a standard deviation of 0.5.

3 Implementation

3.1 Generative Adversarial Network (GAN)

GAN are a data generation technique based on machine learning, used to create realistic synthetic data. It consist of two distinct neural networks, a generator and a discriminator. The generator attempts to produce data that mimics real data, while the discriminator evaluates whether the data comes from the generator or the real dataset. Through this competitive dynamic, the generator learns to create increasingly realistic data, while the discriminator improves its ability to distinguish real from synthetic.

To implement this method, we start by implemente a function for initializing convolutional and batch normalization layer weights. Next, we need to construct a generator and a discriminator. The generator network takes a noise vector as input and generates an image. And the discrimator network takes an image and predict if this image is real or a generation. The interest of this network is to train the generator to predict realistic images. The discriminator acts as an evaluator to help guide the generator's learning process. By distinguishing real data from generated data, the discriminator provides a feedback signal to the generator.

The fundamental principle of this method is based on a competitive dynamic between the generator and the discriminator. The generator attempts to deceive the discriminator by producing increasingly realistic data, while the discriminator continually improves its ability to distinguish the true from the false. This competition is essential for both networks to progress simultaneously, pushing the generator to produce high quality samples.

With this model, the loss function used is the Binary Cross Entropy because we need to predict two classes : True image or generated image. And Adam is chosen as the optimizer because of its ability to adapt learning rates, manage gradients efficiently, stabilize training and be economically efficient in terms of memory. These features make it an invaluable tool for navigating the unique challenges posed by training GANs.

During training, each batch of images passes through several key stages.

First, the discriminator is updated with real data. Real images are fed into the discriminator, then loss is calculated and gradients are accumulated. Next, the discriminator is updated using the calculated gradients and the generated data. The images generated by the generator are sent to the discriminator, then the loss is calculated and the gradients are accumulated. The discriminator is then updated using the calculated gradients.

Finally, the generator is updated. Its aim is to trick the discriminator into generating realistic images. To do this, the loss is calculated using the discriminator outputs for the generated images, then the gradients are accumulated. The generator is updated using the calculated gradients.

This training loop is repeated for a specified number of epochs, allowing the generator and discriminator to improve progressively.

Finally, we visualize the results by plotting the original data and images generated and results will be analysed using a FID score.

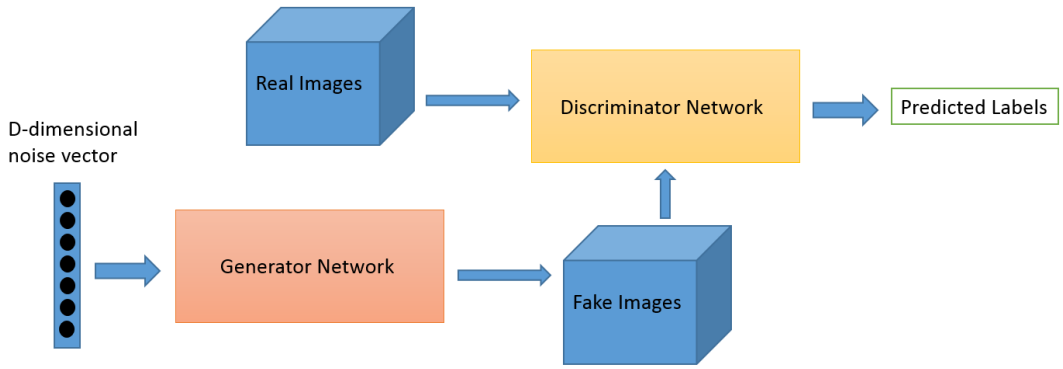


Figure 2: GAN overview

Advantages:

- **Synthetic data generation:** GANs can generate new synthetic data that resembles some known data distribution, which can be useful for data augmentation, anomaly detection, or creative applications.
- **High-quality results:** GANs can produce high-quality results in image synthesis.
- **Unsupervised learning:** GANs can be trained without labeled data, making them suitable for unsupervised learning tasks, where labeled data is scarce or difficult to obtain.

Disadvantages:

- **Training instability:** GANs can be difficult to train, with the risk of instability, mode collapse, or failure to converge.
- **Computational cost:** GANs can require a lot of computational resources and can be slow to train, especially for high-resolution images or large datasets.
- **Overfitting:** GANs can overfit the training data, producing synthetic data that is too similar to the training data and lacking diversity.

3.2 Denoising Diffusion Probabilistic Models (DDPM)

Probabilistic Diffusion Models (DDPM) are a class of generative models used for image generation. They are based on the principle of diffusion, which involves modeling the process of generating an image from initial noise through progressive diffusion.

The diffusion process begins with a noisy image and applies a series of diffusion steps to progressively refine the image and make it closer to the distribution of real data. Each diffusion step is characterized by a diffusion equation that determines how the noise is propagated through the image. Ultimately, the diffusion model is trained to predict the next diffusion step from the current image and associated noise.

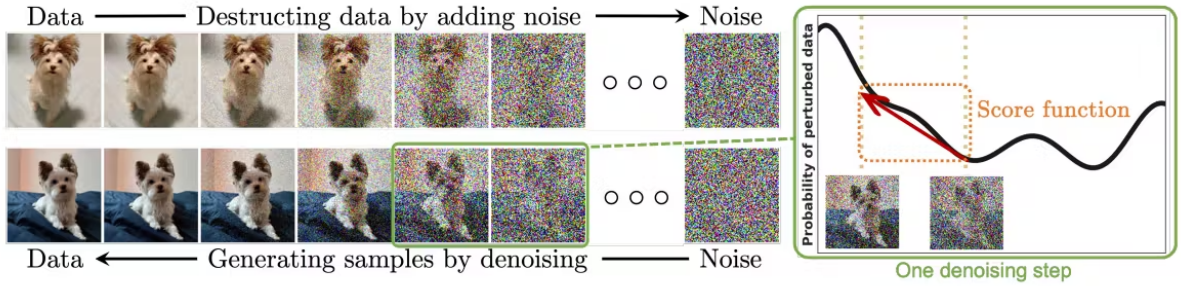


Figure 3: Principle of diffusion

To implement DDPM using a UNet model to generate images samples from a dataset, we need to start by configurate the UNet model.

The interest of this neural network lies in its ability to capture information at different spatial scales while preserving fine details of the image. It features both encoding and decoding layers, thus allowing to avoid vanishing gradient problems and helping to retain important information throughout the diffusion process. Moreover, across different layers, it retains useful information, hence operating at multiple scales. But, in comparison to a classic UNet, the model used in this study is significantly deeper and more complex due to several enhancements explained in the following.

For more details, during the encoding phase, the input image is progressively downsampled through a series of convolutional and pooling layers. At each encoding step, a ResNet block is utilized to extract features of the image at that spatial scale. The ResNet block typically consists of multiple convolutional layers followed by an activation function, such as the ReLU function, and possibly a pooling layer to reduce dimensionality. Then, residual connections within the ResNet blocks bypass vanishing gradient issues and aid in preserving important information throughout the encoding process.

During the decoding phase, features extracted from encoding layers are merged to reconstruct the image at a higher spatial resolution. At each decoding step, features extracted from the corresponding encoding step's ResNet blocks are concatenated with

decoded features from the previous step. This enables the model to capture information at different spatial scales and preserve fine details of the image. The ResNet blocks in the decoding phase also assist in merging features extracted at different spatial scales to efficiently reconstruct the image at a higher resolution.

Next, we need to define a diffusion model. This is based on the idea of modeling the progressive transformation from an initial probability distribution to the distribution of real data through a diffusion process. Instead of directly modeling the probability distribution of the data, the diffusion model represents how an initial noise is progressively transformed to generate realistic data. And at each diffusion step, a transformation is applied to the initial noise to generate an estimation of the image at that step.

During the training time, the loss function used is the Mean Square Error (MSE). MSE is commonly used in generative modeling tasks as it measures the average squared difference between the generated and ground truth images. Adam is chosen as the optimizer because of its ability to adapt learning rates, manage gradients efficiently, stabilize training and be economically efficient in terms of memory. These features make it an invaluable tool for navigating the unique challenges posed by training DDPMs.

Finally, we visualize the results by plotting the original data and images generated and results will be analysed using a FID score.

Advantages:

- **High-quality generation:** DDPMs can generate high-quality images with fine detail by modeling the process of gradual diffusion from the initial noise to the final image.
- **Training stability:** Using probabilistic diffusion techniques, DDPMs are generally more stable to train than other image generation architectures, such as GANs, because they don't require competition between a generator and a discriminator.

Disadvantages:

- **Computational complexity:** Image generation with DDPMs can be slower than other approaches due to the sequential nature of the diffusion process, requiring several steps to generate a single image.
- **Dependence on image size:** DDPM performance can vary according to the size and complexity of the image to be generated. Models may struggle to effectively capture fine detail in high-resolution or highly complex images.

4 Results

4.1 Generative Adversarial Network (GAN)

Generation quality :

Overall, it doesn't take a large number of epochs to start seeing results and discerning the shapes of each generated image. As shown by the batch of images in figure 19, increasing the number of epochs progressively improves the clarity of the results.

For a single epoch, the training time is too short to obtain satisfactory visual results. The generated digits lack quality and are partially erased, although it is still possible to discern the generated digit.

With five epochs of training, the quality of the generated images improves, making it easier to identify the less clear digits.

From twenty epochs onwards, most of the generated images closely resemble the images from the original dataset, especially for simple digits like 1 and 7. In this prediction sample, most digits are very well generated, although the MNIST dataset presents a wide variety of handwriting styles, complicating the generation task. The best-generated digits are often those that are most uniformly written or drawn in the original dataset. For example, the 5's are almost all similar, while there are many variations for the 3's. These differences make generation more difficult.

Although the image quality improves with more epochs, the difference between twenty and fifty epochs is minimal for certain digits. Among these results, the digits 8, 6, 5, and 1 are very well generated.

Figure 4: True images from MNIST

(a) Generated images with 1 epoch

(b) Generated images with 5 epochs

(c) Generated images with 20 epochs

(d) Generated images with 50 epochs

Figure 5: Generation comparison of GAN with different number of epochs

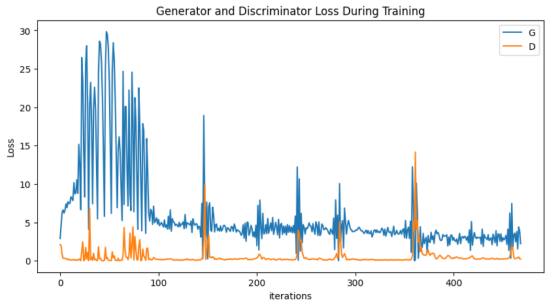
The following graphs show that the GAN model, with different configurations, follows an expected learning trajectory, with fluctuations in losses and confidence scores reflecting the competitive and evolving nature of this training. The losses of the two networks are closely linked, as can be seen from the oscillations that occur at the same times.

However, we can observe that the discriminator's loss oscillates over the iterations. When the loss is low, it means the discriminator correctly distinguishes between real and fake images. The oscillations indicate that the generator periodically manages to fool the discriminator by producing realistic images, temporarily disrupting the discriminator's ability to distinguish them, which results in an increase in the discriminator's loss. The

discriminator then adjusts and reduces its loss again, creating a cycle of oscillations.

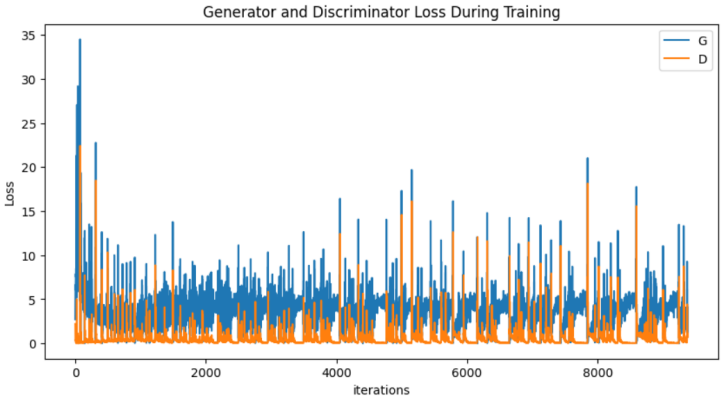
For 5 epochs (Figure 6b), the discriminator's loss curve started at 2.0947 and ended at 0.2552. Although it spiked to 3.5999 at one point, it quickly recovered and stabilized with cycles showing progressively lower loss with each oscillation. As shown in the graph, outside of these oscillations, the discriminator's loss remained below 1.

In contrast, the generator was much less stable. The oscillations of both networks are closely linked, as shown in the graph. The generator started with a higher error of 8.3555, which is significantly higher than that of the discriminator, but it quickly stabilized.



(a) Loss on generated images with 1 epoch

(b) Loss on generated images with 5 epochs



(c) Loss on generated images with 20 epochs

Figure 6: Loss comparison of GAN with different number of epochs

For 20 epochs, the pattern was quite similar, except in this case, the training phase experienced a period without major oscillations between 1500 and 3500 iterations. During this time, the two networks likely reached a temporary equilibrium where the generator produced images realistic enough that the discriminator had more difficulty distinguishing them from real images, but not to the point of completely fooling it. This is reflected in the loss values calculated during training. If we look at the third column, the results are very close to 1, indicating that the discriminator effectively identifies real images. Additionally, this period corresponds to a phase where the error was very low.

```
[3/20][0/469]   Loss_D: 0.3396  Loss_G: 3.8389  D(x): 0.9299    D(G(z)): 0.2009 / 0.0339
[3/20][50/469]  Loss_D: 0.1338  Loss_G: 3.9046  D(x): 0.9186    D(G(z)): 0.0416 / 0.0287
[3/20][100/469] Loss_D: 0.8533  Loss_G: 3.1235  D(x): 0.8562    D(G(z)): 0.4531 / 0.0590
[3/20][150/469] Loss_D: 0.2829  Loss_G: 4.0163  D(x): 0.8601    D(G(z)): 0.1067 / 0.0287
[3/20][200/469] Loss_D: 3.0300  Loss_G: 0.4555  D(x): 0.0826    D(G(z)): 0.0021 / 0.6916
[3/20][250/469] Loss_D: 0.2087  Loss_G: 3.3046  D(x): 0.8809    D(G(z)): 0.0694 / 0.0517
[3/20][300/469] Loss_D: 0.3153  Loss_G: 3.0670  D(x): 0.8246    D(G(z)): 0.0956 / 0.0675
[3/20][350/469] Loss_D: 2.9131  Loss_G: 0.8088  D(x): 0.1210    D(G(z)): 0.0045 / 0.5281
[3/20][400/469] Loss_D: 0.3536  Loss_G: 3.9974  D(x): 0.8914    D(G(z)): 0.1887 / 0.0273
[3/20][450/469] Loss_D: 0.1893  Loss_G: 4.2089  D(x): 0.9006    D(G(z)): 0.0733 / 0.0213
[4/20][0/469]   Loss_D: 0.0953  Loss_G: 4.0647  D(x): 0.9446    D(G(z)): 0.0351 / 0.0269
[4/20][50/469]  Loss_D: 0.0901  Loss_G: 4.0782  D(x): 0.9527    D(G(z)): 0.0358 / 0.0254
[4/20][100/469] Loss_D: 0.3969  Loss_G: 4.5888  D(x): 0.9550    D(G(z)): 0.2608 / 0.0160
[4/20][150/469] Loss_D: 0.2121  Loss_G: 4.1740  D(x): 0.8324    D(G(z)): 0.0140 / 0.0264
[4/20][200/469] Loss_D: 0.3183  Loss_G: 3.3533  D(x): 0.8952    D(G(z)): 0.1699 / 0.0507
[4/20][250/469] Loss_D: 0.0366  Loss_G: 4.6167  D(x): 0.9752    D(G(z)): 0.0110 / 0.0139
[4/20][300/469] Loss_D: 0.0784  Loss_G: 4.6236  D(x): 0.9717    D(G(z)): 0.0457 / 0.0155
[4/20][350/469] Loss_D: 0.4193  Loss_G: 2.2937  D(x): 0.8256    D(G(z)): 0.1651 / 0.1342
[4/20][400/469] Loss_D: 0.3111  Loss_G: 3.4135  D(x): 0.9427    D(G(z)): 0.2036 / 0.0450
[4/20][450/469] Loss_D: 1.1597  Loss_G: 8.7726  D(x): 0.9963    D(G(z)): 0.6088 / 0.0003
[5/20][0/469]   Loss_D: 1.6991  Loss_G: 1.5254  D(x): 0.2581    D(G(z)): 0.0014 / 0.2889
[5/20][50/469]  Loss_D: 0.2383  Loss_G: 3.4550  D(x): 0.9349    D(G(z)): 0.1459 / 0.0434
[5/20][100/469] Loss_D: 0.0359  Loss_G: 4.4917  D(x): 0.9898    D(G(z)): 0.0246 / 0.0169
[5/20][150/469] Loss_D: 0.0611  Loss_G: 4.1490  D(x): 0.9578    D(G(z)): 0.0163 / 0.0252
[5/20][200/469] Loss_D: 0.2221  Loss_G: 2.8834  D(x): 0.8938    D(G(z)): 0.0897 / 0.0843
[5/20][250/469] Loss_D: 1.0398  Loss_G: 1.7425  D(x): 0.4987    D(G(z)): 0.1566 / 0.2246
[5/20][300/469] Loss_D: 0.1735  Loss_G: 3.7903  D(x): 0.9409    D(G(z)): 0.1011 / 0.0305
[5/20][350/469] Loss_D: 0.8897  Loss_G: 1.7831  D(x): 0.5106    D(G(z)): 0.0675 / 0.2294
[5/20][400/469] Loss_D: 0.1375  Loss_G: 3.2426  D(x): 0.9193    D(G(z)): 0.0408 / 0.0615
[5/20][450/469] Loss_D: 0.4889  Loss_G: 1.5863  D(x): 0.6696    D(G(z)): 0.0425 / 0.2717
[6/20][0/469]   Loss_D: 1.0257  Loss_G: 1.4240  D(x): 0.4430    D(G(z)): 0.0061 / 0.3026
```

Figure 7: Extract from losses during training time

These oscillations, although sometimes abrupt, indicate an improvement in the generator's ability to produce quality images. The competitive process between these two networks is crucial for the success of GANs, and these fluctuations are an indicator of their simultaneous progression.

FID score :

As the results shown in the table are not at all good, it is impossible to carry out an analysis. This is probably due to the predictions shown in figure 8. Ground truth and genereated images don't show the same class. So, for this reason, the result of the FID score will increase.

Table 1: FID score of each epochs

Number of epochs	FID score
1	1.186e+80
5	1.297e+109
20	3.136e+93
50	-3.752e+92

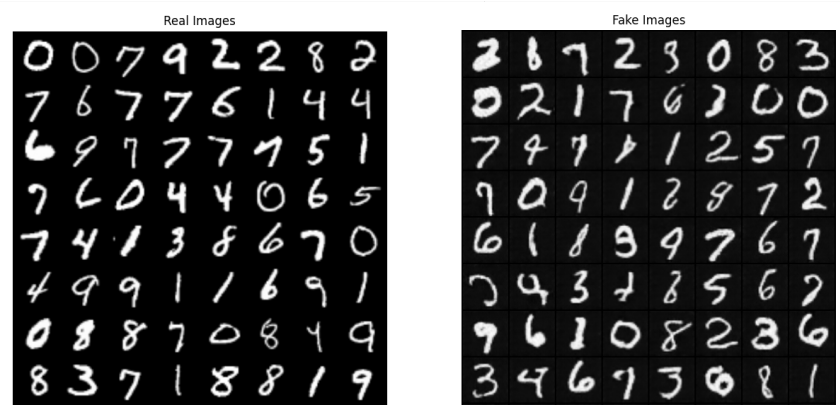


Figure 8: Images after generation

But to avoid this issue, we can compare true number with the same number generated for better results.

Now, the results are more interesting. Initially, if we compare the visual results, Figure 12 is clearly less well generated compared to Figures 10 and 11. However, when we consider the FID score, the difference is not significant. The best FID score is achieved by Figure 10, which is visually the best generated, but the difference is small from the FID point of view. This can be explained by the variety within the MNIST dataset. The FID score calculated between these images might be higher because this particular image from the dataset may not be well-represented in the generated data.

Table 2: FID score of each generation

Image	**FID score**
Fig 10	215.21
Fig 11	220.52
Fig 12	240.21

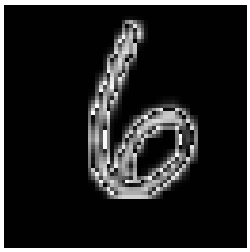


Figure 9: Real image

Figure 10: Image generated

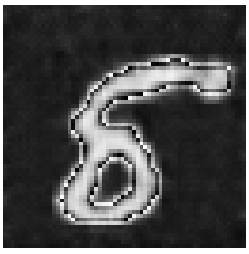


Figure 11: Image generated

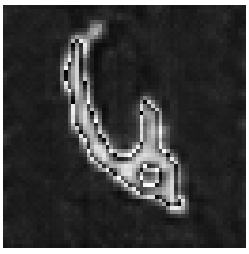


Figure 12: Image generated

Figure 13: Comparison of generation on number 6

Training time :

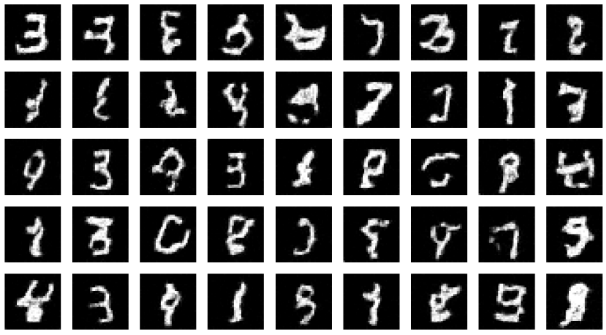
Regarding training time, GAN requires a significant investment. The required time obviously depends on the number of real data samples and the number of generations, but results can be obtained in approximately 5 minutes for 5 epochs of 469 steps each. It takes about 20 minutes for 20 epochs and 50 minutes for 50 epochs, averaging about 1 minute per epoch. This is relatively fast compared to other methods. This model generates 128 new images.

4.2 Denoising Diffusion Probabilistic Models (DDPM)

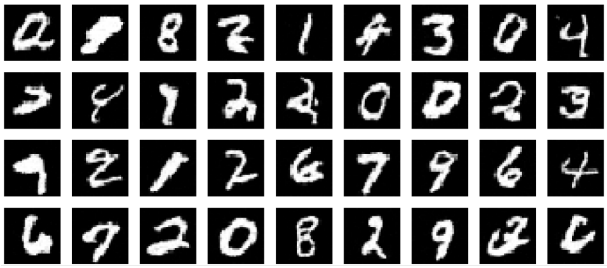
Generation quality :

Similar to the previous method, it doesn't take a large number of epochs to start seeing results and discerning the shapes of each generated image. As shown in Figure 14, increasing the number of epochs progressively improves the clarity of the results.

Between 1000 and 5000 epochs, there's a significant improvement where the generated digits are no longer blurred and begin to resemble correct generation. However, only a small portion is generated very well, notably the 9 and a few other isolated cases. It would be interesting to train for more epochs, but due to the required training time and computational cost, it was not necessarily feasible to train such a model.



(a) Generated images with 1000 epochs



(b) Generated images with 5000 epochs

Figure 14: Generation comparison of DDPM with different number of epochs

FID score :

Analyzing the FID score, it's unnecessary to calculate it over the entire generated dataset as explained with the GAN method. However, for reference, Table 3 shows the results obtained.

Table 3: FID score of each epochs

Number of epochs	**FID score**
1000	-3.942e+104
5000	1.145e+96

We will directly compare the true numbers with their generated counterparts. Fundamentally, Figure 19 shows a real image and three generations, all with fairly similar calligraphy but not always a good representation. Once again, the FID score demonstrates that the generated images are not entirely resembling, which is confirmed by examining the images in Figure 19.

Table 4: FID score of each generations

Image	**FID score**
Fig 16	250.80
Fig 17	243.87
Fig 18	269.41

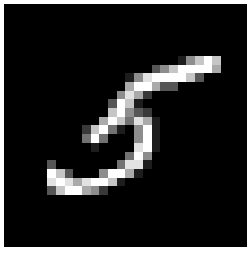


Figure 15: Real image

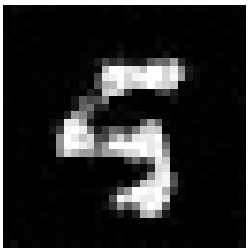


Figure 16: Image generated

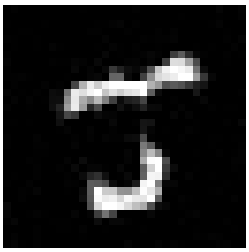


Figure 17: Image generated

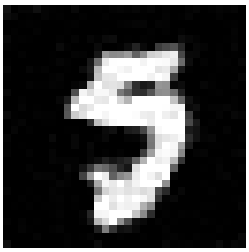


Figure 18: Image generated

Figure 19: Comparison of generation on number 5

Training time:

In terms of efficiency, this method proves to be less effective than the previous one. As said in the Implementation part (3.2), this method need more time for generation. For 5000 epochs and using Google Colab, it needs 43min to finish 81 generations. And 12min for 1000 epochs. In fact, the number of generations will considerably influence the efficiency of the algorithm, as will the number of epochs, but the result is much better.

5 Conclusion

In conclusion, the application of GANs and DDPMs for image generation presents different results. Through the examination of generated images, it's evident that both methods show improvement in generation quality with increased epochs of training. However, despite progress, there remains a notable gap between generated images and real data, particularly in terms of fine details and accuracy.

The competitive nature of GANs, where a generator network competes against a discriminator network, fosters the production of increasingly realistic images over epochs. This process is reflected in the oscillations of discriminator loss and the gradual improvement of generated images. However, the quality of generated images can vary significantly depending on factors such with the dataset diversity.

Similarly, DDPMs demonstrate potential in generating images with realistic features, although with different training dynamics compared to GANs. While DDPMs do not rely on a discriminative network, their training process involves diffusion steps to model the data distribution.

Moreover, the evaluation of FID scores provides quantitative insights into the similarity between generated and real images. Despite some differences, FID scores serve as a useful metric for assessing generation quality and tracking progress over training epochs, but need to be used only on separately images.

In summary, the advancements in GANs and DDPMs offer interesting results on image generation.

References

[1] « Generative Adversarial Network (GAN) ». GeeksforGeeks, 15 janvier 2019, link.

[2] Gitau, Antony M. "A Friendly Introduction to Denoising Diffusion Probabilistic Models." Medium, 9 July 2023, link.

[3] "Complete Guide to Generative Adversarial Networks (GANs)." Paperspace Blog, 19 Apr. 2021, link.

[4] Tremblay, Charles. "F1-score F-beta score, compromis entre Precision et Recall en classification." Kobia, 17 Nov. 2021, link.