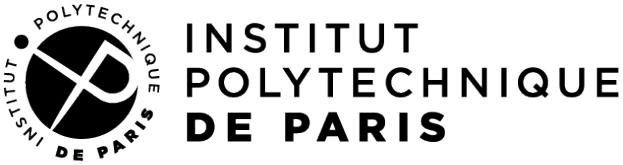
Introduction to Generative Models Report

Paper : Tackling the Generative Learning Trilemma with Denoising Diffusion GANs [10]

Ahmed KHAIRALDIN
ahmed.khairaldin@ensae.fr

Amine RAZIG
amine.razig@ensae.fr

Zakarya ELMIMOUNI
zakarya.elmimouni@ensae.fr



Institut Polytechnique Paris
Palaiseau, France
January 2025

Contents

Introduction

Recently, generative models have gained importance in several domains such as image generation. The goal of these models is to generate high-quality images starting from real image distributions. Diffusion models, for example, have achieved state-of-the-art performance in terms of the quality of the generated samples. Moreover, they exhibit good mode coverage, which refers to the ability of a generative model to represent the entire data distribution effectively, indicated by the likelihood. However, diffusion models suffer from slow sampling which makes them computationally expensive in practice.

High quality samples, mode coverage and fast sampling are thus three key requirements, current generative models try to achieve. However, as it is the case with diffusion models, even mainstream generative models cannot satisfy all three of them simultaneously. In [10], this is referred to the *generative learning trilemma*. As Figure 1 shows, Generative Adversarial Networks (GANs) [3], [1] are known for fast sampling of high quality outputs, but they often fail to represent the full diversity of the data distribution [9], [11]. Moreover Variational Autoencoders (VAEs) [6], [8], and Normalizing Flows [2], [7] obtain good mode coverage, but tend to produce samples of lower perceptual quality.

Our paper of interest [10] introduces a new model that tackles the generative trilemma using denoising diffusion models with fast sampling. The authors observe that the slow sampling is due to the Gaussian assumption for the denoising in diffusion models, which requires a large number of denoising steps. They remedy that by using a multimodal distribution modeled with conditional GANs to have few denoising steps. The authors claim outperforming GANs in terms of mode coverage and diffusion models in mode coverage and speed in sampling.

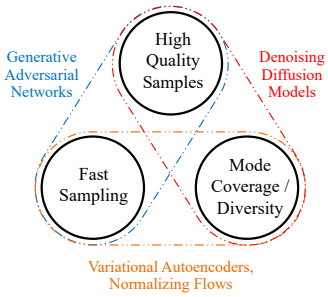


Figure 1: The Generative Learning Trilemma for mainstream generative models

In this report, we first discuss the theoretical foundations of this model based on the diffusion models theory, and more particularly Denoising Diffusion Probabilistic Models (DDPM). Then, we discuss sample quality experiments conducted by the authors of [10], and by us. Our code is available in the following drive link.

1 Model

As shown above in Figure 1, diffusion models are doing great job in coverage and they generate high quality samples, but they are quite slow, because of the large number of steps, denoted T, used in the denoising process. So for diffusion models to be competitive in terms of speed, the paper introduces a way to accelerate the sampling from diffusion models by reducing the number of steps T. However, when the number of denoising steps T is not large, or if the data distribution is non-Gaussian, there is no guarantee that the Gaussian assumption, made in diffusion models, holds on the denoising distribution. Thus, if we minimize the number of step size T in diffusion

we cannot assume that such an assumption still holds. On the top of figure 2, we observe that in the forward process the distribution is smoother as we move towards x_T this is due to the gaussian distribution of $q(x_t|x_{t-1})$. But at the bottom of Figure 2, we observe that in the reverse process, the conditional distribution of x_4 given x_5 resembles a Gaussian. However, when "jumping" from x_5 to x_3, this Gaussian assumption begins to break down, and it worsens further when "jumping" from x_5 to x_1, where the Gaussian assumption becomes entirely invalid.

1.1 The Forward Process

As in diffusion models, the forward process of the denoising diffusion GANs model consists in adding noise gradually to the clean image in order to get noisy images. More formally:

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \boldsymbol{\epsilon},$$

where β_t is a hyperparameter controlling the amount of noise. and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In other words:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\big(x_t \mid \sqrt{1-\beta_t}\, x_{t-1},\, \beta_t\, \mathbf{I}\big),$$

As t increases, the original information in x_0 becomes increasingly submerged in noise, making x_T nearly indistinguishable from pure noise. The key advantage is that this process is reversible: we can train a model (the *reverse process*) to gradually "denoise" and reconstruct coherent data starting from a noisy sample.

1.2 The Denoising Steps

We need to perform the reverse process of the forward pass, i.e., transitioning from x_T, which represents noise, back to a clean image x_0. A common way to achieve this in the literature is to approximate the reverse distribution $q(x_{t-1}|x_t)$ to a Gaussian distribution. In our paper of interest, they claim that such an assumption holds only in two cases: first in the limit of infinitésimal step size β_t, second if data marginal $q(x_t)$ is Gaussian. Outside these two assumptions, we cannot ensure that $q(x_{t-1} \mid x_t)$ is a Gaussian distribution.

Since The goal of the paper is to reduce the number of denoising steps in the reverse process in diffusion models to acheive fast sampling, the Gaussian assumption in the denoising process doesn't hold anymore.

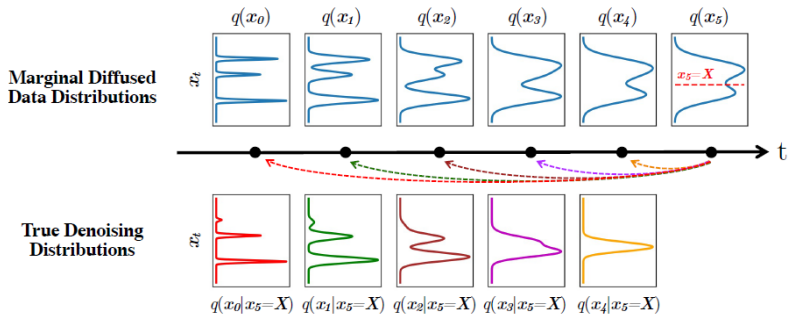


Figure 2: **Top**: The evolution of 1D data distribution $q(x_0)$ through the diffusion process. **Bottom**: The visualization of the true denoising distribution for varying step sizes conditioned on a fixed x_5

.

The idea is to model the denoising process with a multimodal distribution, using conditionnal GANs to approximate the true denoising distribution $q(x_{t-1}|x_t)$. More details on conditional GANs are available in our additional document.

1.3 Parametrizing the Implicit Denoising Model

A central idea in diffusion models is that instead of directly predicting x_{t-1} from x_t at each reverse step, we can factorize the reverse distribution as follows:

$$p_\theta(x_{t-1} \mid x_t) \;=\; q\big(x_{t-1} \mid x_t,\; x_0 = f_\theta(x_t, t)\big),$$

where: x_0 is the *clean* (or denoised) version of the data predicted by a neural network f_θ given the noisy sample x_t at time t. and $q(x_{t-1} \mid x_t, x_0)$ is the *posterior distribution* over x_{t-1}

Because the forward process is chosen to be Gaussian, the distribution $q(x_{t-1} \mid x_t, x_0)$ also has a Gaussian form. Hence, once they predict x_0 with f_θ, sampling x_{t-1} becomes a matter of drawing from a known Gaussian whose mean and variance can be expressed analytically.

For example in Denoising Diffusion Probabilistic Models (DDPM) [5]. they sample x_{t-1} using the formula:

$$x_{t-1} \;=\; \frac{1}{\sqrt{\alpha_t}}\Big(x_t \;-\; \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\Big) \;+\; \sigma_t\, z,$$

where:

- $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ come from the forward-noise schedule.
- σ_t is related to β_t (the variance term) and determines how much noise is re-injected at each reverse step.
- $z \sim \mathcal{N}(0, \mathbf{I})$ is fresh Gaussian noise sampled at each step.

By "subtracting" the predicted noise $\epsilon_\theta(x_t, t)$ from the noisy sample x_t, the procedure partially denoises x_t to yield x_{t-1}. This is repeated iteratively for $t = T, T-1, \ldots, 1$, until one ends up with a fully denoised sample x_0.

In our case in the denoising diffusion GANs, they similarly defined $p_\theta(x_{t-1}|x_t)$ by:

$$p_\theta\big(x_{t-1} \mid x_t\big) := \int p_\theta\big(x_0 \mid x_t\big)\, q\big(x_{t-1} \mid x_t, x_0\big)\, \mathrm{d}x_0 \;=\; \int p(z)\, q\Big(x_{t-1}\; x_t,\; x_0 = G_\theta\big(x_t, z, t\big)\Big)\, \mathrm{d}z.$$

where $p_\theta(x_0 \mid x_t)$ is the implicit distribution imposed by the GAN generator $G_\theta(x_t, z, t) : R^N \times R^L \times R \;\to\; R^N$ that outputs x_0 given x_t and an L-dimensional latent variable: $z \sim p(z) := \mathcal{N}(z; 0, I)$.

One can view the resulting $p_\theta(x_0 \mid x_t)$ as being implicitly defined by a generator G_θ that outputs x_0 given both the current noisy sample x_t and a latent variable $z \sim \mathcal{N}(0, \mathbf{I})$. The overall principle remains the same: the network infers a candidate for x_0, and then uses the Gaussian posterior $q(x_{t-1} \mid x_t, x_0)$ to sample x_{t-1}. the main difference between the model of the paper and the DDPM model is the fact that x_0 is predicted as a deterministic mapping of x_t, while in the paper x_0 is produced by the generator with random latent variable z.

Once the sample x_{t-1} is drawn (following the distribution in the previous equation), we feed it into a *time-dependent discriminator* D_ϕ together with x_t and the time step t. The discriminator's role is to output a scalar in $[0, 1]$ that indicates how "realistic" x_{t-1} is as the denoised version of x_t. Specifically, if x_{t-1} truly comes from the diffusion (or "reverse" diffusion) process, the discriminator should return a value close to 1, whereas poorly denoised samples should return values closer to 0.

To train this discriminator, they contrast real samples x_{t-1} with fake samples generated using the distribution $p_\theta(x_{t-1} \mid x_t)$.

The discriminator is trained by:

$$\min_{\phi} \sum_{t\geq 1} E_{q(x_t)} \Big[E_{q(x_{t-1}|x_t)}[-\log(D_\phi(x_{t-1}, x_t, t))] + E_{p_\theta(x_{t-1}|x_t)}[-\log(1 - D_\phi(x_{t-1}, x_t, t))] \Big], \tag{5}$$

where fake samples from $p_\theta(x_{t-1} \mid x_t)$ are contrasted against real samples from $q(x_{t-1} \mid x_t)$. The first expectation requires sampling from $q(x_{t-1} \mid x_t)$ which is unknown. However, we use the identity $q(x_t, x_{t-1}) = \int dx_0\, q(x_0)\, q(x_{t-1} \mid x_0)\, q(x_t \mid x_{t-1}, x_0)$ to rewrite the first expectation in Eq. 5 as:

$$E_{q(x_t)\, q(x_{t-1}|x_t)}[-\log(D_\phi(x_{t-1}, x_t, t))] = E_{q(x_0)\, q(x_{t-1}|x_0)\, q(x_t|x_{t-1})}[-\log(D_\phi(x_{t-1}, x_t, t))].$$

After training the discriminator in this way, we *update the generator* to produce samples x_{t-1} that maximize $\log D_\phi(x_{t-1}, x_t, t)$, ensuring that denoised samples increasingly appear "real" to D_ϕ.

$$\max_{\theta} \sum_{t\geq 1} E_{q(x_t)} E_{p_\theta(x_{t-1}|x_t)} \big[\log(D_\phi(x_{t-1}, x_t, t))\big],$$

which updates the generator with the non-saturating GAN objective [4].

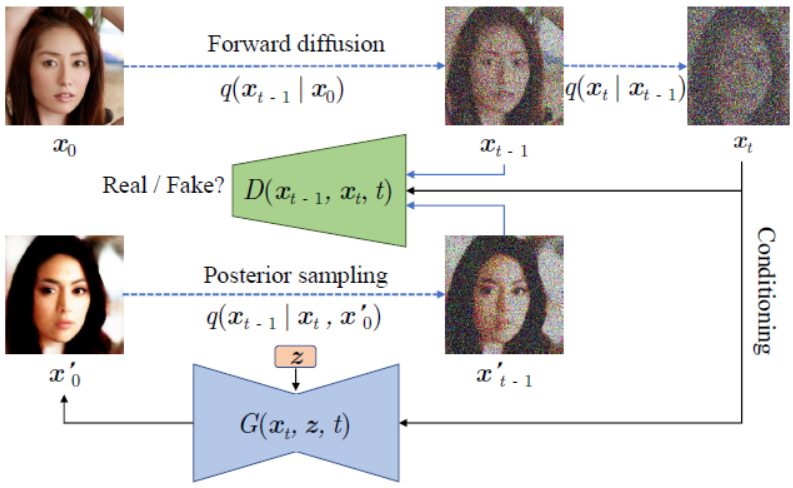


Figure 3: The training Process of denoising diffusion GANs

2 Experiments

For our experiments, we use the MNIST dataset which is a benchmark dataset widely used in machine learning. It consists of 70,000 grayscale images of handwritten digits (0-9), with 60,000 images in the training set and 10,000 in the test set. Each image is 28x28 pixels, making it computationally lightweight while still providing sufficient complexity for our tasks.

Since the model was introduced by an NVIDIA team using more than 8 powerful GPUs to train their models, and due to our limited resources, we were constrained in our training process. Even though we adapted the MNIST dataset to the denoising diffusion GANs model (by duplicating 1 channel into 3 channels), we could only train our model for 60 epochs for several days of trials with many memory usage problems. In our experiments, we compare

it with a DDPM model trained for 60 epochs on the MNIST dataset, following the implementation inspired by this GitHub repository.

In the following, we first discuss a quality comparison with other models in the experiments conducted by the authors. Then, we discuss the results obtained in our modest experience.

2.1 Paper's experience

Figure 3 shows the Fréchet Inception Distance (FID) score as a function of sampling time for different models applied to the CIFAR-10 dataset. The FID score is a metric used to evaluate the similarity between real images and generated images. We observe that the denoising diffusion GANs (DDG) model outperforms almost all baseline models, as a lower FID score indicates higher quality of the generated images. However, some StyleGAN models are competitive with the DDG model and may even outperform it. However, for a different dataset we could have different results. Moreover, we discuss other experiments showing that our model of interest is able to tackle the generative learning trilemma in our additional document.

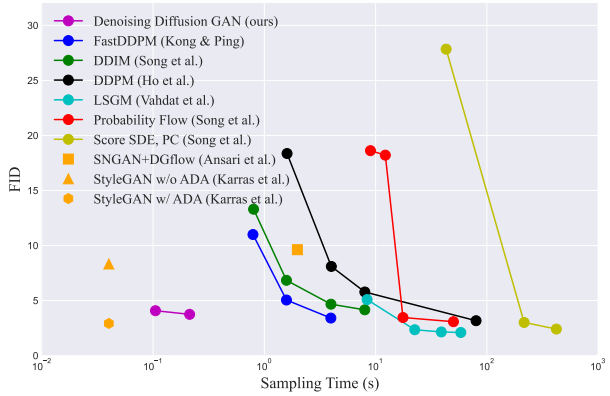


Figure 4: Sample quality vs sampling time trade-off.

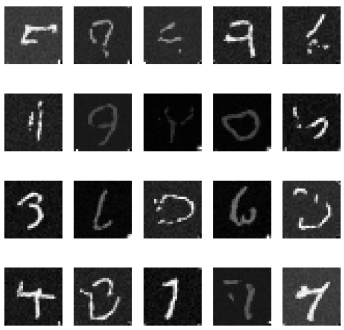
2.2 Our experiment

Since the model was introduced by an NVIDIA team using more than 8 powerful GPUs to train their models, and due to our limited resources, we were constrained in our training process. Even though we adapted the MNIST dataset to the denoising diffusion GANs model (by duplicating 1 channel into 3 channels), we could only train our model for 60 epochs for several days of trials with many memory usage problems. In our experiment, we compare it with a DDPM model trained for 60 epochs on the MNIST dataset, following the implementation inspired by this GitHub repository.

We compare generated samples for MNIST dataset of our model of interest and a DDPM model implemented in this GitHub repository. as shows Figure 4.

Since we could only train our model for 60 epochs due to limited resources, we cannot formally compare the quality of the generated samples. For example, the experiments in the reference paper required 1200 epochs and 8 GPUs running in parallel to train on the CIFAR-10 dataset. However, as shown in Figure 4, our model performs better within 60 epochs and with far fewer denoising steps (200 for DDPM versus only 4 for DDG).

Moreover, regarding the generated samples produced by the DDG, we observed that due to the adaptation of



(a) Generated samples with DDPM

(b) Generated samples with DDG

Figure 5: Comparison between generated samples DDPM vs. Denoising Diffusion GANs

the dataset to 3 channels (by duplicating and permuting grayscale data), colors appear in the generated images. This occurs because the model does not perfectly learn the 3-channel distribution during the limited 60 epochs of training. Since the dataset was artificially expanded to 3 channels, the channels are not representative of natural RGB colors. The model, therefore, associates different intensities in each channel in a way that creates visible "colors." With more training, the quality of the generated samples could improve significantly. Moreover, our model is characterized with its fast sampling due to small number of denoising steps (4 instead of 200) compared to the DDPM.

Conclusion

The Denoising Diffusion GANs model is an acceleration method that addresses the generative models' trilemma. It achieves faster sampling than diffusion models with far fewer denoising steps. Additionally, it is competitive with the best models in terms of mode coverage and sample quality.

This model enables us to work with high resolution images since it performed well with the LSUN-Church and Celeb-HQ-256 datasets as shown in the paper's experiments section. Moreover, it outperforms most of the baseline models in terms of mode coverage as seen with the experiment of the popular 25-Gaussian toy dataset (see Figure 6 in [10]).

A potential improvement to this model could be to incorporate the generated image before posterior sampling into the discrimination process. In Figure 3, we can add a discriminator which takes as input the generated image $x_0^{'}$ and the ground-truth image x_0

References

[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations (ICLR)*, 2017.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, volume 27, 2014.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680, 2014.

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014.

[7] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[8] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

[9] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, volume 29, 2016.

[10] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

[11] Shengjia Zhao, Jiaming Song, and Stefano Ermon. The information bottleneck theory of deep learning: An empirical study using gans. In *International Conference on Machine Learning (ICML)*, 2018.

Appendix

To reproduce our experiments: LINK

- Train the model on the MNIST dataset with the following command in the terminal:

```
python3 train_ddgan.py --dataset stackmnist --exp ddgan_MNIST0_exp1 --num_channels 3 \
--num_channels_dae 128 --num_timesteps 4 --num_res_blocks 2 --batch_size 64 --num_epoch 100 \
--ngf 64 --nz 100 --z_emb_dim 256 --n_mlp 4 --embedding_type positional --r1_gamma 0.02 \
--lr_d 1.25e-4 --lr_g 1.6e-4 --lazy_reg 15 --num_process_per_node 1 --ch_mult 1 2 2 2 \
--save_content
```

- Generate samples with the following command:

```
python3 test_ddgan.py --dataset stackmnist --exp ddgan_MNIST0_exp1  --num_channels 3 \
--num_channels_dae 128 --num_timesteps 4 --num_res_blocks 2 --nz 100 --z_emb_dim 256 \
--n_mlp 4 --ch_mult 1 2 2 2 --epoch_id 60
```

- For the DDPM results, you can check the model.ipynb notebook file handed with the rest of the code, also available this GitHub repository