Dauphine | PSL

Data Science Project A2 - IASD 2022-23
Diffusion Model for handwritten digits generation

Group Theo_Max_Chris: Christian Kayo, Théophane Vallaeys, Maximilien Wemaere

1 Introduction

The aim of this project was to study and implement a generative model to learn how to generate handwritten digits from the MNIST dataset. Many methods have been developed to tackle data generation, including (variational) autoencoders, normaliing flows, or GANs. We choose to work with diffusion models, which are one of the basis for some state of the art models [Wang et al., 2022].

2 Diffusion models

Diffusion models are a class of generative models with the main advantages of generating more diverse images than a GAN due to the generation process. There already exist several diffusion-based architectures. This project will focus on the most widely used one, which is Denoising Diffusion Probabilistic Models (DDPM) proposed by Ho, Jain and Abbeel in 2020 [Jonathan Ho, 2020]. The main idea behind a DDPM is a two step process: gradually adding noise to images (forward process) and then training a neural network to denoise the samples and invert the process (backward process).

2.1 The forward diffusion process

The forward diffusion process is a Markov Chain following a manually chosen distribution, and therefore doesn't require any training. Starting with the initial image, we add Gaussian noise for T successive steps, parameterized by a noise schedule $\beta_1, \ldots, \beta_T$, and obtain a list of gradually nosier samples. The prediction of probability density at time t only depends on the immediate predecessor at time $t-1$ and therefore, the conditional probability density can be computed as follows:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta}x_{t-1}, \beta_t I) \tag{1}$$

$$x_t = \sqrt{1-\beta}x_{t-1} + \beta_t \varepsilon_t \text{ where } \varepsilon_t \sim \mathcal{N}(\mathbf{0}; \mathbf{1}) \tag{2}$$

However, it can be costly to go through each of the successive noising steps. As each noising step sample from independent Gaussian distributions (Markov chain), we can express a chain of noising steps as sampling from a single Gaussian, expressed below, where $\alpha_t = 1|\beta_t$ and $\overline{\alpha_t} = \prod_{i=1}^{t} \alpha_i$.

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha_t}}x_0, (1-\overline{\alpha_t})I) \tag{3}$$

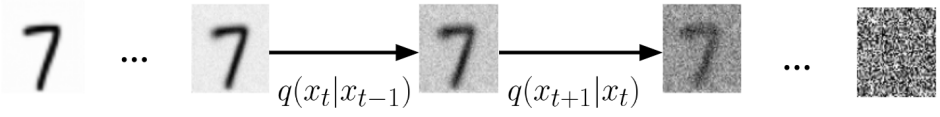


Figure 1: The forward diffusion process

2.2 The reverse process

Given that the forward process is a Markov chain described by a known distribution $q(x_t|x_{t-1})$, we want to train a model to learn the reverse probability distribution $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$. μ_θ and Σ_θ are usually given by a neural network. This idea is that, given the Markov chain, $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and therefore we can sample images from the input distribution by sampling x_T at random and applying the reverse process.

To achieve better performance [Jonathan Ho, 2020], we can express μ_θ as a function of the noise ε_θ, which will be predicted by the model: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t, t))$. Our analysis is that it reduces the dependence of the model on the exact value of t and forces it to better follow the noise schedule.

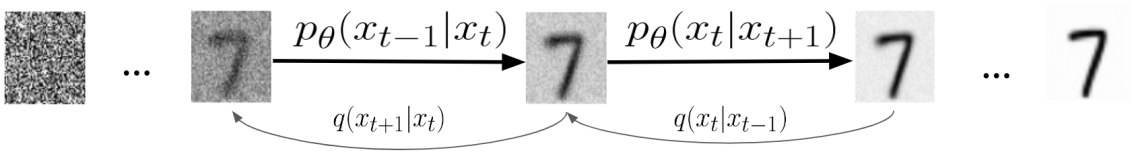


Figure 2: The reverse diffusion process

3 Conducted experiments

In our experiments, unless specified, we set $\Sigma_\theta(x_t, t) = 0$ and therefore use a deterministic backward process with $x_{t-1} = \mu_\theta(x_t, t)$. Some minor experiments not shown here include the choice of hyper-parameters and size for the neural networks (we found it best to choose a network a bit smaller than the diffusion models trained on CIFAR10). We also implemented both the prediction of μ_θ or ε_θ for every architecture, and chose to keep the U-Net using the ε_θ prediction for the later experiments. T was set to 400, and a few experiments confirmed it to be a good choice for this setting.

3.1 Choosing a network architecture

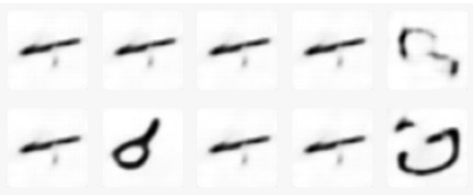
All our neural networks architectures had three inputs: the noisy image, a sinusoidal embedding of the timestep t, and a one-hot encoding of the label of the image (used for conditional generation).

3.1.1 Autoencoder and VAE

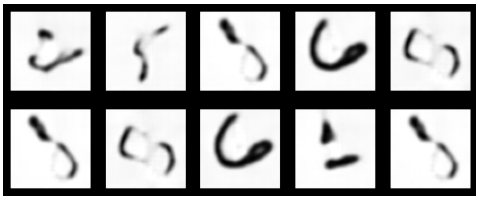
The autoencoder [Geoffrey E. Hinton, 1944] architecture is a classical architecture for image generation and seemed to fit as the subroutine of the generation process. As the idea is to encode the image in a small embedding space, we though that a convolutional autoencoder architecture could help to extract the meaningful information from the noisy image.

The problem that ocured is that the DDPM process predicts μ_θ, which is a random noise without structure and therefore is not compressible into a small dimension. That's why we added a skip connection and a few convolutional layers at the end. The idea was that if the network inner parts can predict an estimate of the original image, the noise can be obtained by subtraction with the noised image.

We both implemented a simple autoencoder as well as a variational autoencoder [Kingma and Welling, 2013], using the same architecture expect for the embedding layer. The later yield better results.



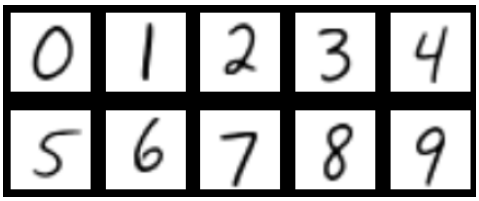
(a) Results with an Autoencoder



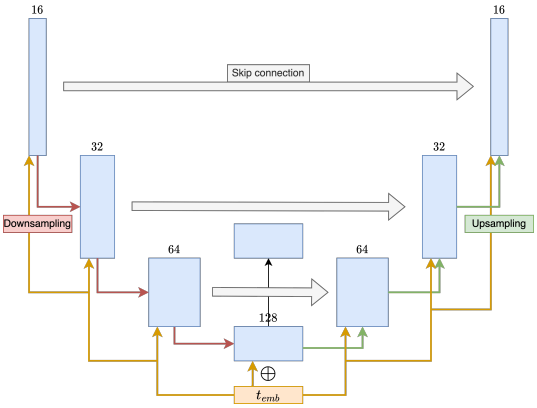
(b) Results with a VAE

3.1.2 U-Net

Our best results came from using a U-net [Ronneberger et al., 2015]. Inspired by the autoencoder, the idea is to use a fully-convolutional architecture to progressively downsample the layers while using an increasing number of layers, and then doing the reverse operations by upscaling the layers. A key difference with autoencoders is that there is a skip connection added at each level, allowing to get back the details from the input image, as well as to stabilize the gradient on large networks.



(a) Result with a U-net



(b) U-Net architecture

3.2 Noise scheduling

As seen in part 2, the noise is generated thanks to the noise scheduler which choose the values of β_t.

The noise β_t is usually taken to be a gradually decreasing value, such that $\overline{\alpha}_t$, which is correlated with the noisy simple quality, starts at 1 ($t = 0$) and ends up close to 0 ($t = T$). It should give better results when $\overline{\alpha}_t$ decreases slowly at the beginning and the end, but linearly at the center [Nichol and Dhariwal, 2021]. But our experiments didn't match those results.

We compared a linear scheduler (4), a quadratic scheduler (5), a cosine scheduler (6) and a sigmoid scheduler (7).

$$\beta_{t,linear} = \beta_0 \frac{1-t}{T} + \beta_T \frac{t}{T} \quad (4)$$

$$\beta_{t,quadratic} = \left(\sqrt{\beta_0}\frac{1-t}{T} + \sqrt{T}\frac{t}{T}\right)^2 \quad (5)$$

$$\beta_{t,cosine} = 1 - \frac{cos^2(\frac{t/T+s}{1+s}\frac{\pi}{2})}{cos^2(\frac{(t-1)/T+s}{1+s}\frac{\pi}{2})}) \quad (6)$$

$$\beta_{t,sigmoid} = \sigma(12\frac{t}{T} - 6)(\beta_T - \beta_0) + \beta_0 \quad (7)$$

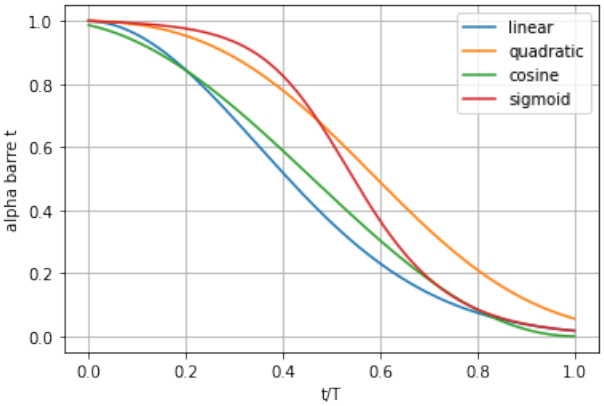


Figure 5: Evolution of $\overline{\alpha_t}$

While the linear scheduler gave a FID of 41.43, we had values of 85.88 for the quadratic one, 95.74 for the sigmoid scheluder and 215.80 with cosine. This results matched the sample quality we observed, and were consistent with multiple values of T between 200 and 2000.

3.3 Conditional image generation and discriminator

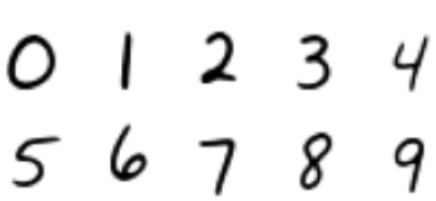
We introduced conditional image generation by giving the targeted label as an input to the neural network, with one-hot encoding, in the middle layer of the U-Net. To force the network to use this

label to generate the output, we added a discriminator. This is a neural network trained to recognize the MNIST digits. Every training step, we use the following loss, with $\hat{y}$ the real label given as an input and $\mathcal{D}$ the discriminator.

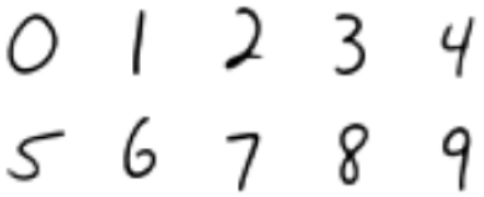
$$L = L_{simple} + \gamma L_{discriminator} \tag{8}$$

$$= ||\mu_\theta(x_t, \hat{y}, t) - \hat{x}_{t-1}||_2 + \gamma||\mathcal{D}(\mu_\theta(x_t, \hat{y}, t)) - \hat{y}||_2 \tag{9}$$

We obtained good results with this method, and never saw samples from the wrong class, but we didn't measure the classification error. Surprisingly, the results didn't change without the discriminator. We are not sure if the FID improved, as is seemed to unstable.



(a) Result with the discriminator



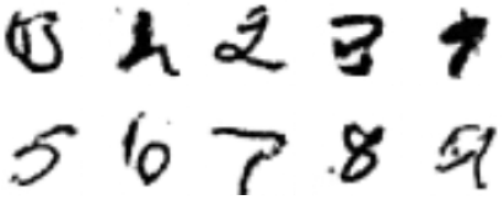
(b) Results without the discriminator

3.4 Faster image sampling

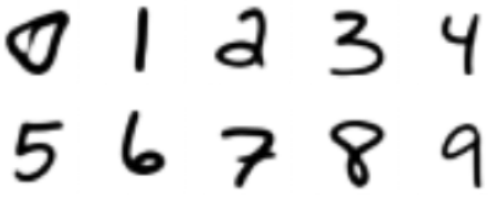
The last important part of our experiments was to compare methods to improve sampling speed, as it was the most time-consuming operation. We implemented a Denoising Diffusion Implicit Model [Song et al., 2020]: the idea is that, if we can predict noise, we can use it to predict x_0 instead of x_{T-1}. This first step will produce a first rough estimate of x_0, that probably won't be as good as with all the T steps. We will then add noise again to generate x_{t_2}, predict a new x_0, and again for K steps with $T = t_1 > t_2 > \cdots > t_K = 0$.

The second method we tried is to sub-sample t values. After training with a large T, we used a smaller $T' = K$ for sampling, by setting $\alpha'_k = \prod_{t=t_k}^{t_{k-1}} \alpha_t$, and $\beta'_k = 1 - \alpha'_k$.

Both methods didn't need changes to the neural network, and were therefore compared using the same U-Net weights. The DDIM showed pretty noisy results, be it with a small T or while keeping the original T value. In comparison, the sub-sampling of T didn't change the result nor the FID, even with as low as 5 steps for the generation. The FID stayed around 49 for the sub-sampling method. But with the DDIM, it reached 42: we think that this is due to the poor quality of the pre-trained FID on this dataset and the increase in the diversity of images due to the noise, more than to a real increase in quality.



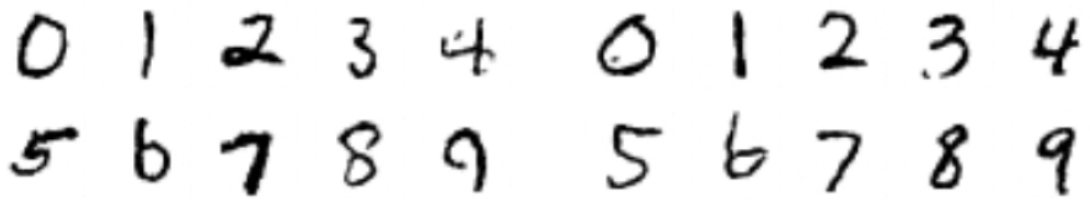
(a) Result using the DDIM



(b) Results using sub-sampling of t

3.5 Sampling with a variance

The problem of learning Σ_θ is hard due to the instability of the loss and the training when used [Nichol and Dhariwal, 2021]. Even more, in our setting, using a variance added too much noise to the results, be it with $\sigma_t = \beta_t$ or $\sigma_t = \tilde{\beta}_t$. As the recommended way to learn variance is an interpolation between those two values, we didn't experiment further. As with the DDIM, the results shown didn't required to train a new network and are using a previously trained model, and the FID decreased, probably due to the random noise. When submited to the test server (on lamsade.dauphine.fr), the FID went fromo 25 to 60 and the precision and recall decreased, confirming that this method was, in reality, worse than with $\sigma = 0$.



(a) With $\sigma_t = \beta_t$ (b) With $\sigma_t = \tilde{\beta}_t$

4 Conclusion

- Using the FID with a pre-trained InceptionV3 network was not reflective on sample quality on MNIST: the network should be fine-tuned. We didn't took the time to do it as the FID was given by an external library, and the setup had already been time-consuming.

(a) FID of 111.02 (b) FID of 339.05

- The results were unstable, with large differences between experiments. We don't know if this is only due to the FID measure, or if there is another problem or unstability. This is not due to the fact that the distribution is estimated with a random sampling, as the FID was consistently stable for a given model with the same weights.
- Denoising result was basically settled from the first steps. We should use a better noise scheduling, or another noising method (like Gaussian blur, or noise inside a VAE latent space), or try to investigate why the sinusoidal noise scheduler isn't as good as the linear one.

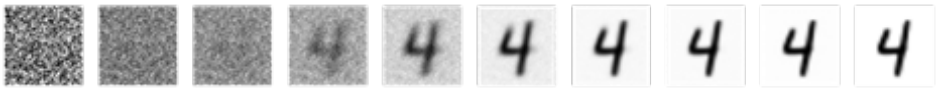


Figure 10: Too fast denoising

References

[Geoffrey E. Hinton, 1944] Geoffrey E. Hinton, R. S. Z. (1944). Autoencoders, minimum description length and helmholtz free energy.

[Jonathan Ho, 2020] Jonathan Ho, Ajay Jain, P. A. (2020). Denoising Diffusion Probabilistic Models.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes.

[Nichol and Dhariwal, 2021] Nichol, A. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

[Song et al., 2020] Song, J., Meng, C., and Ermon, S. (2020). Denoising diffusion implicit models.

[Wang et al., 2022] Wang, Z., Zheng, H., He, P., Chen, W., and Zhou, M. (2022). Diffusion-gan: Training gans with diffusion.