Dauphine | PSL
UNIVERSITÉ PARIS

MASTER IASD

DATA SCIENCE PROJECT

Diffusion Model for Image Generation

Realised by:

ALJOHANI Renad

DJECTA Hibat Errahmen

PRÉAUT Clément

CHR Diffusion Team

Supervised by:

Mr. NEGREVERGNE Benjamin

Mr. VÉRINE Alexandre

2022/2023

Table of Contents

List of Figures

1 Introduction

Generating noise from data is easy, but generating data from noise is a generative model. Generative models are decades old. In machine learning, this type of model helps us to represent a set of data from a markov chain or just using a generational iterative process to do the same thing. In the case of this work, we are going to use them to be able to have high-level representations from high-dimensional data more precisely images. Many types of generative models exist in the literature, in our case we are going to focus on diffusion models.

Diffusion models recently have caught the attention of the computer vision community by producing photorealistic images. In this work, we have used a diffusion model to generate coherent images from noisy ones. We have started by a basic probabilistic model that we use to improve later by the employing of the concept of stochastic differential equations. All of the tests have been performed on MNIST dataset.

This report resumes our work and it's divided into four parts. The first one is a kind of a basic concepts representation. The second part concerns the basic probabilistic diffusion model, and the third can be seen as an improvement of this later. Later on the last part we will show some results and a comparison between them.

2 Theory of the diffusion model

Diffusion models are a new generative deep learning model that has been demonstrated to produce high-quality and diverse samples by destroying the input until only noise remains and then recovering the input from the noise using a neural network. They are used to generate data that is similar to the data on which they were trained. These models are being used in many different domains, such as audio generation and image generation. Furthermore, it consists of two steps: the forward diffusion process and the reverse diffusion process. In the foregoing process, the gaussian noise is gradually added to the data until it consists entirely of noise. The reverse process is used to learn conditional probability density distributions using a neural network model to eliminate noise.

2.1 Forward

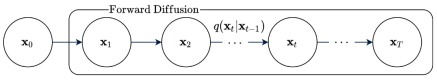

Figure 1: Forward processing

the forward diffusion process can be properly described as a Markov chain. Therefore, the forward diffusion is to adds Gaussian noise to an image until only the preceding image is pure noise. It depends on previous image and a specific variance. Small Gaussian noise is added to the sample of steps, to creating a series of samples with a high level of noise. The sequence of is knowing as variance schedule that describes how much noise we want to add in each time of steps. Is the previous image with less noise, which means the mean of our distribution is exactly the previous image multiplied with the variance of is normal distribution multiplied by identity.

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

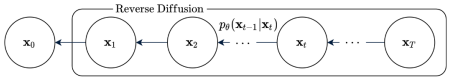

Figure 2: Backward processing

2.2 Backward

In the backward process, we must figure out the probability density at an earlier time step based on how the system right know. Firstly, we need to estimate when the t=1 thus producing a data sample from isotropic Gaussian noise notice that it cannot easily estimated because it needs to use the entire dataset. Consequently, we must train a neural network model to predict the based on previously learnt weights and the present condition at time t.

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2}$$

3 Development of a basic version

3.1 Comprehension of pytorch framework

As the project was required to use the framework Pytorch, a consequent time of the project was given to the well understanding of this framework. Usually, we use Tensorflow and Pytorch is a more detailed framework with sometimes more steps to code a neural network. It gave us the occasion to parametrize more in detail our neural network.

3.2 Definition of the Unet

The model must progressively denoise an image. So at each step, we must show as output a picture of the same dimension of the input. As it is also a computer vision problem, the use of a U-net seems interesting. This kind of neural networks showed in experimentation its efficiency to delimit the main component of an image(Ho et al. 2020) . The general architecture of the neural network is shown on figure 3.

We implemented a downscale using convolution with $stride = 2$ to divide the picture dimension by 2 until reaching a vector of 2 dimensions. Downsample layer uses `AvgPool2d` to help extraction of features.

Upsampling is done by the mode `'nearest'`. We also use between same level convolutional layers skip connections in order to maintain good training abilities for such a deep neural network.

Several residual layers are used in every downsampling and upsampling layer. They improve the accuracy of the model. Inside residual layers, we employe batch normalization to make the learning stable, dropout to avoid overfitting and relu as activation function.

In the same way, an attention layer is used at every level what enable the model to remember information of the high dimensional input while processing the reduced dimension vector.

In order to indicate to the model which step of the denoising it is training on, we use sinusoidal position embedding which is widely used for this purpose.

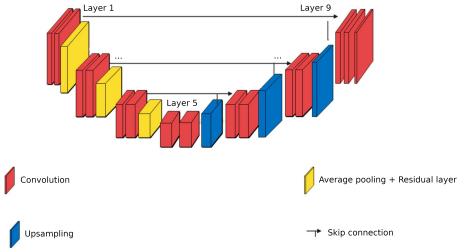

Figure 3: Schema of the unet

3.3 Choice of beta scheduling

For the first version of the unet, we chose a linear betas scheduling from 10^{-4} to 0.02 as it was used by Ho et al. in 2020. This enables us to have a functional simple model quickly, without fearing bugs from this simple parameterization (Song et al. 2020) .

3.4 Results

We tried to know how many epochs will be necessary to get a satisfying result. The graph 4 shows the evolution of the loss according to the number of epochs. We can see that the number begin to stabilize at 300 epochs.

Therefore, we trained this first version on 300 epochs using 300 timestep. On figure 5, we can see an example of denoising of a picture. For the purpose of verification of the training, the noise at the beginning comes from a noised 9 number picture. We display in the figure 5 the denoising process every 30 timestep. We can see that the training works because the generated picture at the end is a 9.

We can observe that the number begin to be visible only at the end of the denoising, which is unusual.

The figure 6 show denoising of gaussian noise for 100,200 and 300 training epochs. We can see that the visual quality of the result is equivalent for 200 and 300 epochs, and much less satisfying for 100 epochs. For 200 and 300 epochs, we can see well generated complex strucutred numbers as in row 3 column 1 and 4 for 200 epochs and rows 2 and 3 column 3 for 300 epochs. But there are still bad quality generated numbers as in row 4 column 3 for 200 epochs or row 1 column 1 for 300 epochs.

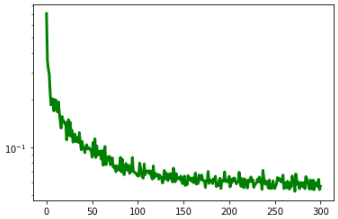

Figure 4: Evolution of loss according to the nulber of epochs.

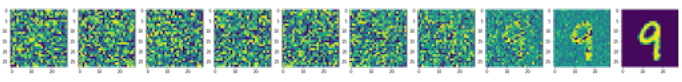

Figure 5: Example of the denoising of the train number 9.

Figure 6: Denoising of gaussian noise with 100,200 and 300 epochs trained model.

3.5 Limit of the first version

The first model suffers from limitations in performance and quality.For only 300 epochs, the training took 13122 seconds, that is to say more than 3.5 hours. This is very long considering the fact that it was trained on the GPU of the lamsade laboratory.

The fid was also not satisfying and was more than 100. The generation of the 1024 pictures used to compute the fid took a huge time.

4 Improvements thanks to SDM

4.1 Principle of SDM

Perturbing data with multiple noise scales is key to the success of the previous diffusion model method. In the literature, there are other approaches that generalize this idea further to an infinite number of noise scales. such that perturbed data distributions evolve according to a stochastic diffusion equation as the noise intensifies. Figure 7 is an overview of this (Song et al. 2020) .

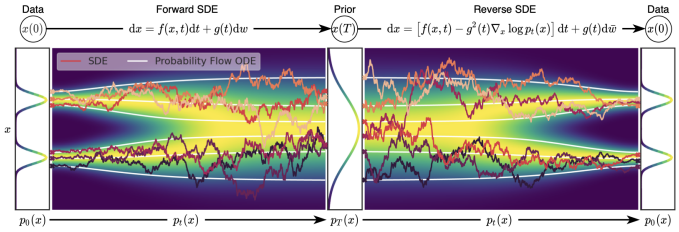

Figure 7: Overview of score-based generative modeling through SDEs (Song et al. 2020)

Here, p_0 is the data distribution and p_T is the prior distribution. The two processes of the diffusion model in this case can be modeled as the solution to an SDE.

p_T contains no information of p_0 such as a Gaussian distribution with fixed mean and variance. Here we design the SDE such that it diffuses the data distribution into a fixed prior distribution. This is derived from the continuous generalization of the previous method. This is the forward process and it's represented by this equation:

$$dx = f(x,t)dt + g(t)dw \tag{3}$$

where w is the standard Wiener process, f is called the drift coefficient of x(t), and g is known as the diffusion coefficient of x(t).

By starting from samples of p_T and reversing the process,we can obtain samples of p_0 using a reverse-time SDE like in:

$$dx = [f(x,t) - g^2(x) + \nabla_x log p_t(x)]dt + g(t)d\bar{w} \tag{4}$$

where $\bar{w}$ is a standard Wiener process when time flows backwards from T to 0, and dt is an infinitesimal negative timestep. Once the score of each marginal distribution is known for all t, we can derive the reverse diffusion process from and simulate it to sample from p_0 (Song et al. 2020) .

Crucially, the reverse-time SDE depends only on the time-dependent gradient field of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. Here we are still using a unet model. After training it, then we will simulate its results with numerical approaches to generate samples from p_0.

4.2 Results

We present here some results obtained after implementing this approach that prove its efficiency.

We start by recording the loss as in figure 8.

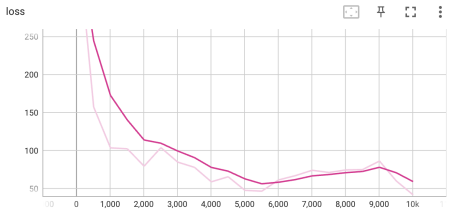

Figure 8: The evolution of loss over epochs

We notice that over 10000 epochs the loss decreases perfectly and this is a sign that our model learns something.

In the other hand, We calculate the bpd which gives us also a sign of the learning quality of our model. less the bpd better is the model. and it's our case.

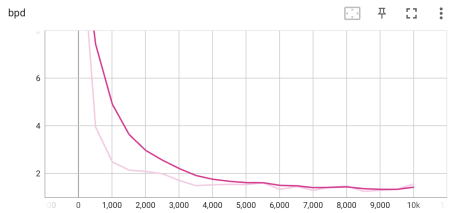

Figure 9: The evolution of Bits per dimensions over epochs

The figure 10 shows the generation of 16 pictures from gaussian noise after 7000 epochs. This model offers good visual with no irrelevant pictures. Nevertheless, we can see some ambiguous results as in row 1 column 4 where the result is a mi between a 4 and a 9.

The SDM improvement bring more learing speed and quality compared to the basic version. For example, over 5500 epochs, the SDM model took only 11 minutes to train which more acceptable than the 3,5 hours of the basic model.

Thanks to SSDM, we reach a satisfying fid of 25 at 10000 epochs.

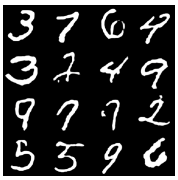

Figure 10: The evolution of Bits per dimensions over epochs

5 Conclusion

Diffusion models are an elegant and easy approach to deal with data generating and more precisely with image generation.In this report, we have discussed the basic concepts of diffusion models and their implementations. Also, we have presented a more advanced way to deal with this type of generative models using stochastic differential equations. Finally, we have made a comparison between our implementations. The improvement permit us to quickly have images in an enhanced quality. The wide-ranging capabilities of diffusion models are inspiring, and we do not yet know the real extent of their limitations.

Bibliography

Ho, Jonathan, Ajay Jain and Pieter Abbeel (2020). 'Denoising Diffusion Probabilistic Models'. In: DOI: 10.48550/ARXIV.2006.11239. URL: https://arxiv.org/abs/2006.11239.

Song, Yang et al. (2020). 'Score-Based Generative Modeling through Stochastic Differential Equations'. In: DOI: 10.48550/ARXIV.2011.13456. URL: https://arxiv.org/abs/2011.13456.