Diffusion Models in Depth: From a Theoretical and a Practical Perspective

Gregory Sedykh

June 28th 2024

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE
Department of Informatics

Bachelor thesis for the degree of Bachelor of Science in Computer Science
Supervised by Prof. Stéphane Marchand-Maillet

Abstract

Contents

1 Introduction

Diffusion models have gained widespread popularity since 2020, as models such as DALL-E, Stable Diffusion and Midjourney have proven to be capable of generating high-quality images given a text prompt. Furthermore, OpenAI's recent announcement of Sora has shown that diffusion models have now also become highly capable of generating minute long high-definition videos from a text prompt. [1]

These models date back to 2015, where the idea of a diffusion model appeared, based on diffusion processes used in thermodynamics. [6]
Denoising Diffusion Probabilistic Models (DDPMs) were a development of the original diffusion probabilistic model introduced in 2015. [3]
Subsequently, OpenAI improved upon the original DDPMs which did not have ideal log likelihoods [3] while also using less forward passes and therefore speeding up the sampling process. [5]
The most recent progress done by OpenAI has allowed their diffusion models to obtain better metrics and better sample quality than Generative Adversarial Networks (GANs) which were previously considered the state-of-the-art in image generation. [2]

The fairly recent apparition of diffusion models means not only that there is still a lot to be discovered about them, but also that progress is being made rapidly.
The theory behind diffusion models was mainly founded when Ho et al. [3] introduced their DDPMs in 2020, but many improvements have been made upon their work since then.
To understand what these models are and how they work, it is crucial to understand how DDPMs were developed, what choices were made when developing them and why these choices were made, as well as why changes were made to the original model and how they were made.

This report aims to provide an overview of the theory behind diffusion models, as well as a practical guide on how to implement a simple diffusion model using PyTorch, in order to compare what the theory shows us and what the practical implementation gives us.

2 Denoising Diffusion Probabilistic Models

A **diffusion model** is a generative model that consists of two Markov chains, one forward and one reverse.
Given an input (e.g. an image), the forward process will destroy the image by gradually adding Gaussian noise at each step of the process. [3]
The reverse process' objective is to "reverse" the forward process, starting from the noisy image and step-by-step, estimating the noise that was added to the image at each step and removing it until the first step is reached, where we should obtain the original image. [3]

2.1 Forward Process

The forward process is a Markov process that starts from the original image x_0 and adds Gaussian noise during T steps, where each step $i \in [1, T]$ has a size $\beta_i \in \{\beta_1, ..., \beta_T\}$, which results in a more noisy image x_i. [3]

Formally, we obtain:

$$q(x_1, ..., x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{2}$$

Equation 2 gives us a single step forward: given the previous image x_{t-1}, we add noise from a Gaussian distribution with mean $\sqrt{1-\beta_t}x_{t-1}$ and variance $\beta_t I$ to obtain the next image x_t.
Equation 1 gives us the full forward process from the original image x_0 to the final image x_T.

It is important to note that as $T \to \infty$ and with a correct choice of β_t, x_T will become a sample of an isotropic Gaussian distribution ($\mathcal{N}(0, I)$). [5] [6].
This is important for the reverse process, as it will allow us to take a sample $x_T \sim \mathcal{N}(0, I)$ and reverse the forward process to obtain the original image x_0 (however this cannot be done so simply, as seen in section 3) [5].

The reparametrisation trick says that for a univariate Gaussian distribution where $z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$, a valid reparametrisation would be $z = \mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$ is just noise. [4]
Ho et al. [3] use this trick to be able to sample x_t at any arbitrary step t of the forward process.

Let $\alpha_t = 1 - \beta_t$, then:

$$x_t \sim \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$
$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t I}\epsilon_{t-1} \qquad \epsilon_{t-1} \sim \mathcal{N}(0, I)$$
$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

From this, we can apply it again to x_{t-1} and obtain:

$$x_{t-1} = \sqrt{\alpha_{t-1}} x_{t-2} + \sqrt{1 - \alpha_{t-1}} \epsilon_{t-2}$$

Therefore, we get:

$$x_t = \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{\alpha_t (1 - \alpha_{t-1})} \epsilon_{t-2} + \sqrt{1 - \alpha_t} \epsilon_{t-1}$$

Using the reparametrisation trick again, we can write:

$$\sqrt{\alpha_t (1 - \alpha_{t-1})} \epsilon_{t-2}$$

as

$$w \sim \mathcal{N}(0, \alpha_t (1 - \alpha_{t-1}))$$

and

$$\sqrt{1 - \alpha_t} \epsilon_{t-1}$$

as

$$z \sim \mathcal{N}(0, 1 - \alpha_t)$$

Since the sum of two Gaussian distributions is also a Gaussian distribution, we can write:

$$w + z \sim \mathcal{N}(0, (\alpha_t (1 - \alpha_{t-1}) + 1 - \alpha_t) I)$$

Which finally gives us:

$$x_t = \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \bar{\epsilon}_{t-2}$$

We can recursively apply this until x_0.
Let $\bar{\alpha}_t = \prod_{i=0}^{t} \alpha_i$:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \bar{\epsilon} \tag{3}$$

This will give us a way to find a noisy image at step t of the forward process, given the original image x_0 (we ignore the noise $\bar{\epsilon} \sim \mathcal{N}(0, I)$):

$$q(x_t | x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I) \tag{4}$$

We now also know that $1 - \bar{\alpha}_t$ is the variance of the noise added at step t of the forward process. [5]

Choosing the variances β_t is an important step when developing the forward process.
Ho et al. [3] chose to use a linear schedule starting from $\beta_1 = 10^{-4}$ and increasing linearly until $\beta_T = 0.02$ in order for them to be small compared to the pixel values that were in $[-1, 1]$. [3]
However we will see in section 3 that this is not the best choice, as this destroyed the images too quickly closer to the end of the process while providing little to sample quality, which made the model be sub-optimal for 64×64 and 32×32 images. [5]

Finally, Ho et al. [3] chose $T = 1000$ in order to be able to compare their model with Sohl et al.'s [6] model, which also used $T = 1000$ for most image experiments.
We will again see in section 3 that so many steps can make the sampling slow and that there are better choices. [5]

2.2 Reverse Process

As mentioned previously, with a correct choice of β_t, x_T will become a sample of an isotropic Gaussian distribution as $T \to \infty$. [5, 6]
This should mean that we can take a sample $x_T \sim \mathcal{N}(0, I)$ and reverse the forward process to obtain the original image x_0. However, this means that we need to use the entire dataset in order to figure out $q(x_{t-1}|x_t)$, which in practice cannot be done. [5]
Therefore, an estimation is found using a neural network. [5]

The reverse process is defined as follows [3]:

$$p_\theta(x_0, ..., x_T) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{5}$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{6}$$

Equation 5 gives us the full reverse process, starting from the final image x_T to the original image x_0.
Equation 6 gives us the reverse process at step t, where we estimate the noise that was added to the image at step t of the forward process in order to find the image x_{t-1}.
Two parameters must be estimated to find the reverse process at step t: the mean $\mu_\theta(x_t, t)$ and the variance $\Sigma_\theta(x_t, t)$.
At first, Ho et al. [3] used neural networks to estimate both the mean and the variance, but explain that estimating the variance was not necessary as there was little difference between the estimated variance $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}$ and the actual values $\sigma_t^2 = \beta_t$ [3]. Thus they set $\Sigma_\theta(x_t, t) = \sigma_t^2 I$.
As for the mean, it is estimated using a neural network trained to optimise a variational lower bound (VLB) of the negative log likelihood $\mathbb{E}[-\log p_\theta(x_0)]$ [3]

2.2.1 Variational Lower Bound

In order to find the negative log likelihood $\mathbb{E}[-\log p_\theta(x_0)]$, we would need to know $p_\theta(x_0)$ which means going through T steps, which is computationally expensive.
Thus, Ho et al. [3] use a variational lower bound (VLB) to estimate the negative log likelihood [3, 6]. The variational lower bound given by Dederik et al. [4] leads us to:

$$VLB = \log p_\theta(x_0) - D_{KL}(q(x_{1:T}|x_0)\|p_\theta(x_{1:T}|x_0)) \tag{7}$$

Since we need to minimise the loss with the negative log likelihood, we can write:

$$-VLB = -\log p_\theta(x_0) + D_{KL}(q(x_{1:T}|x_0)\|p_\theta(x_{1:T}|x_0)) \tag{8}$$

We know that the Kullback-Leibler divergence D_{KL} is non-negative, which means that we can write:

$$\begin{aligned}
-\log p_\theta(x_0) &\leq -\log p_\theta(x_0) + D_{KL}(q(x_{1:T}|x_0)\|p_\theta(x_{1:T}|x_0)) && (9)\\
&= -\log p_\theta(x_0) + \mathbb{E}_q[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)}] && (10)\\
&= -\log p_\theta(x_0) + \mathbb{E}_q[\log \frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_0|x_{1:T})p_\theta(x_{1:T})}{p_\theta(x_0)}}] && (11)\\
&= -\log p_\theta(x_0) + \mathbb{E}_q[\log \frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_0,x_{1:T})}{p_\theta(x_0)}}] && (12)\\
&= -\log p_\theta(x_0) + \mathbb{E}_q[\log \frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_{0:T})}{p_\theta(x_0)}}] && (13)\\
&= -\log p_\theta(x_0) + \mathbb{E}_q[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} + \log p_\theta(x_0)] && (14)\\
&= \mathbb{E}_q[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}] && (15)
\end{aligned}$$

We can subsequently define L_{VLB} as the loss to be minimised:

$$L_{VLB} = \mathbb{E}_q[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}] \tag{16}$$

Ho et al. [3] reformulate the loss in order to reduce variance:

$$L_{VLB} = \mathbb{E}_q[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}] \tag{17}$$

$$= \mathbb{E}_q[\log \frac{\prod_{t\geq 1} q(x_t|x_{t-1})}{p(x_T)\prod_{t\geq 1} p_\theta(x_{t-1}|x_t)}] \tag{18}$$

$$= \mathbb{E}_q[-\log(p_\theta(x_T)) + \log \frac{\prod_{t\geq 1} q(x_t|x_{t-1})}{\prod_{t\geq 1} p_\theta(x_{t-1}|x_t)}] \tag{19}$$

$$= \mathbb{E}_q[-\log(p_\theta(x_T)) + \sum_{t\geq 1} \log \frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}] \tag{20}$$

$$= \mathbb{E}_q[-\log(p_\theta(x_T)) + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \sum_{t\geq 2} \log \frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}] \tag{21}$$

$$= \mathbb{E}_q[-\log(p_\theta(x_T)) + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \sum_{t\geq 2} \log \frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)p_\theta(x_{t-1}|x_t)}] \tag{22}$$

$$= \mathbb{E}_q[-\log(p_\theta(x_T)) + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \sum_{t\geq 2} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} + \sum_{k\geq 2} \log \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}] \tag{23}$$

$$= \mathbb{E}_q[-\log(p_\theta(x_T)) + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \log \frac{q(x_T|x_0)}{q(x_1|x_0)} + \sum_{t\geq 2} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}] \tag{24}$$

$$= \mathbb{E}_q[-\log(p_\theta(x_T)) + \log \frac{q(x_T|x_0)}{p_\theta(x_0|x_1)} + \sum_{t\geq 2} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}] \tag{25}$$

$$= \mathbb{E}_q[-\log(p_\theta(x_0|x_1)) + \log \frac{q(x_T|x_0)}{p_\theta(x_T)} + \sum_{t\geq 2} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}] \tag{26}$$

$$= \mathbb{E}_q[D_{KL}(q(x_T|x_0)\|p(x_T)) + \sum_{t\geq 2} D_{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t)) - \log(p_\theta(x_0|x_1))] \tag{27}$$

This formulation of the loss can be split up into 3 parts:

$$L_T := D_{KL}(q(x_T|x_0)\|p(x_T))$$

$$L_{1:T-1} = \sum_{t\geq 2} D_{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t))$$

$$L_0 = -\log(p_\theta(x_0|x_1))$$

It is worth noting that at equation 22, the terms of q are conditioned on x_0; this is in order to be able to easily compute the forward process posteriors, as they are much easier to find given the original image x_0.
Another important point is that L_T can be omitted since q has no learnable parameters and with the fact that it will almost become a Gaussian distribution, the $D_{KL}(q(x_T|x_0)\|p(x_T))$ (Kullback-Leibler Divergence between a nearly Gaussian distribution and a Gaussian distribution $p(x_T) = \mathcal{N}(x_t; 0, I)$) will be close to 0. [3]

We thus find ourselves with:

$$L_{VLB} = \mathbb{E}_q[\sum_{t\geq 2} D_{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t)) - \log(p_\theta(x_0|x_1))] \tag{28}$$

2.2.2 Simplified training objective

2.2.3 U-Net

2.3 Comparison with other diffusion models

3 Improvements upon DDPMs

4 Results

5 Conclusion

References

[1] Tim Brooks et al. “Video generation models as world simulators”. In: (2024). URL: https://openai.com/research/video-generation-models-as-world-simulators.

[2] Prafulla Dhariwal and Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis*. 2021. arXiv: 2105.05233 [cs.LG].

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[4] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML].

[5] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG].

[6] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585 [cs.LG].