

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

Gregory Igor 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

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Denoising Diffusion Probabilistic Models</b>	<b>3</b>
2.1	Forward Process . . . . .	3
2.2	Reverse Process . . . . .	5
2.2.1	Variational Lower Bound . . . . .	5
2.2.2	Simplified training objective . . . . .	12
2.2.3	Model architecture . . . . .	13
2.3	Score-based formulation . . . . .	15
2.3.1	Score matching . . . . .	15
2.3.2	Sampling . . . . .	16
2.3.3	Noise Conditional Score Networks . . . . .	18
2.3.4	Equivalence between DDPM and NCSN . . . . .	19
<b>3</b>	<b>Improvements upon DDPMs</b>	<b>20</b>
3.1	Improved likelihood . . . . .	20
3.2	Denoising Diffusion Implicit Models (DDIMs) . . . . .	22
3.3	Guidance . . . . .	24
3.3.1	Classifier guidance . . . . .	24
3.3.2	Classifier-free guidance . . . . .	25
<b>4</b>	<b>Comparison with other diffusion models</b>	<b>26</b>
<b>5</b>	<b>Implementation</b>	<b>27</b>
<b>6</b>	<b>Conclusion</b>	<b>28</b>
<b>A</b>	<b>Convolution of two Gaussian distributions</b>	<b>29</b>

---

# 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. [15]

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. [11]

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 information in the image by gradually adding Gaussian noise at each step of the process. [3]

The reverse process' objective is to "reverse" the forward process and learn how to do so, 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 which results in a more noisy image  $x_t$ . [3].

Each step  $t \in [1, T]$  has a size  $\beta_t \in \{\beta_1, \dots, \beta_T\}$ .

This process is defined by  $q$  which is a probability distribution that takes as input an image  $x_t$  and outputs its likelihood.

$$q : \mathbb{R}^D \rightarrow [0, 1] \quad (1)$$

where  $D$  is the data dimensionality (e.g. for a  $64 \times 64$  RGB image,  $D = 64 \times 64 \times 3$ ).

Pixels in the image  $x_t$  are assumed to be mutually independent since the covariance matrix ( $\beta_t I$ ) is diagonal.

Formally, we obtain the following Markov process:

$$q(x_1, \dots, x_T | x_0) = \prod_{t=1}^T q(x_t | x_{t-1}) \quad (2)$$

where:

$$q(x_t | x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t I\right) \quad (3)$$

Equation ?? gives us a single step forward: given the previous image  $x_{t-1}$ ,  $x_t$  is obtained by sampling a  $D$ -dimensional Gaussian distribution with mean  $\sqrt{1 - \beta_t} x_{t-1}$  and variance  $\beta_t I$ .

Equation ?? gives us the full forward process from the original image  $x_0$  to the final image  $x_T$ .

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. [6]

In this case, we can see that the image becomes more random because of the added Gaussian noise from  $\epsilon_t$  at each step, since we are sampling:

$$x_t = q(x_t | x_{t-1}) \quad (4)$$

$$= \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} I \epsilon_t \quad \epsilon_t \sim \mathcal{N}_D(0, I) \quad (5)$$

It is important to note that as  $T \rightarrow \infty$  and with a correct choice of  $\beta_t$ ,  $x_T$  will become a sample of an isotropic Gaussian distribution ( $\mathcal{N}(0, I)$ ). [11] [15].

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) [11].

Ho et al. [3] use the reparametrisation trick to be able to sample  $x_t$  at any arbitrary step  $t$  of the forward process.

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

$$\begin{aligned} x_t &\sim \mathcal{N}\left(\sqrt{1 - \beta_t}x_{t-1}, \beta_t I\right) \\ x_t &= \sqrt{1 - \beta_t}x_{t-1} + \sqrt{\beta_t}I\epsilon_{t-1} \quad \epsilon_{t-1} \sim \mathcal{N}_D(0, I) \\ x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1 - \alpha_t}\epsilon_{t-1} \end{aligned}$$

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:

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

We can write:

$$x_t = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \mathcal{N}(0, \alpha_t(1 - \alpha_{t-1})I) + \sqrt{1 - \alpha_t}\epsilon_{t-1} \quad (6)$$

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \mathcal{N}(0, \alpha_t(1 - \alpha_{t-1})I) + \mathcal{N}(0, (1 - \alpha_t)I) \quad (7)$$

The convolution of the two Gaussian distributions  $\mathcal{N}(\mu_1, \sigma_1^2)$  and  $\mathcal{N}(\mu_2, \sigma_2^2)$  gives us a new Gaussian distribution with mean  $\mu_1 + \mu_2$  and variance  $\sigma_1^2 + \sigma_2^2$  (see Appendix A):

$$x_t = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \mathcal{N}(0, (\alpha_t(1 - \alpha_{t-1}) + 1 - \alpha_t)I) \quad (8)$$

Which finally gives us:

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

Where  $\epsilon_{t-1}, \epsilon_{t-2}, \dots \sim \mathcal{N}(0, I)$  and  $\bar{\epsilon}_{t-2}$  is the convolution of the noise distributions  $\epsilon_{t-1}$  and  $\epsilon_{t-2}$ .

We can recursively apply this backwards until  $x_0$ .

Let  $\bar{\alpha}_t = \prod_{i=0}^t \alpha_i$ :

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \quad (9)$$

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  $\epsilon \sim \mathcal{N}_D(0, I)$ ):

$$q(x_t|x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I) \quad (10)$$

---

We now also know that  $1 - \bar{\alpha}_t$  is the equivalent variance of the noise added to  $x_0$  to obtain  $x_t$  during the forward process. [11]

Choosing the variances  $\beta_t$  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  $64 \times 64$  and  $32 \times 32$  images. [11]

Finally, Ho et al. [3] chose  $T = 1000$  in order to be able to compare their model with Sohl et al.'s [15] 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. [11]

## 2.2 Reverse Process

As mentioned previously, with a correct choice of  $\beta_t$ ,  $x_T$  will become a sample of an isotropic Gaussian distribution over  $\mathbb{R}^D$  as  $T \rightarrow \infty$ . [11, 15]

This should mean that we can take a sample  $x_T \sim \mathcal{N}_D(0, I)$  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. [11]

Therefore, an estimation  $p_\theta(x_{t-1}|x_t)$  of  $q(x_{t-1}|x_t)$  is found using a neural network, with  $\theta$  the parameters of the neural network. [11]

The reverse process is defined as follows [3]:

$$p_\theta(x_0, \dots, x_T) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t) \quad (11)$$

where

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (12)$$

Equation ?? gives us the full reverse process, starting from the final image  $x_T$  to the original image  $x_0$ .

Equation ?? 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  $t$  (from  $x_t$  to  $x_{t-1}$ ): 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  $\sigma_t^2 = \beta_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}$  and the actual values  $\sigma_t^2 = \beta_t$  [3]. Thus they set  $\Sigma_\theta(x_t, t) = \sigma_t^2 I = \beta_t I$ .

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* or more commonly known as *ELBO*) to estimate the negative log likelihood [3, 15].

The variational lower bound given by Dederik et al. is defined as: [6]

$$VLB = \mathbb{E}_{q_\phi(x|z)} [-\log q_\phi(z|x) + \log p_\theta(x, z)] \quad (13)$$

$$= -D_{KL}(q_\phi(z|x) \| p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] \quad (14)$$

where  $X, Z$  are jointly distributed random variables with distribution  $p_\theta(x, z)$ ,  $x \sim p_\theta$  and  $q_\phi$  is any distribution.

This gives us:

$$VLB = \log p_\theta(x_0) - D_{KL}(q(x_{1:T}|x_0) \| p_\theta(x_{1:T}|x_0)) \quad (15)$$

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

$$-VLB = -\log p_\theta(x_0) + D_{KL}(q(x_{1:T}|x_0) \| p_\theta(x_{1:T}|x_0)) \quad (16)$$

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

$$-\log p_\theta(x_0) \leq -\log p_\theta(x_0) + D_{KL}(q(x_{1:T}|x_0) \| p_\theta(x_{1:T}|x_0)) \quad (17)$$

$$= -\log p_\theta(x_0) + \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)} \right] \quad (18)$$

$$= -\log p_\theta(x_0) + \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_0, x_{1:T})}{p_\theta(x_0)}} \right] \quad (19)$$

$$= -\log p_\theta(x_0) + \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_{0:T})}{p_\theta(x_0)}} \right] \quad (20)$$

$$= -\log p_\theta(x_0) + \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} + \log p_\theta(x_0) \right] \quad (21)$$

$$= \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right] \quad (22)$$

so that we obtain that:

$$-\log p_\theta(x_0) \leq \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right] \quad (23)$$

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

$$L_{VLB} = \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right] \quad (24)$$


---



---

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

$$L_{VLB} = \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right] \quad (25)$$

$$= \mathbb{E}_q \left[ \log \frac{\prod_{t \geq 1} q(x_t|x_{t-1})}{p_\theta(x_T) \prod_{t \geq 1} p_\theta(x_{t-1}|x_t)} \right] \quad (26)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \log \frac{\prod_{t \geq 1} q(x_t|x_{t-1})}{\prod_{t \geq 1} p_\theta(x_{t-1}|x_t)} \right] \quad (27)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \log \prod_{t \geq 1} q(x_t|x_{t-1}) - \log \prod_{t \geq 1} p_\theta(x_{t-1}|x_t) \right] \quad (28)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \sum_{t \geq 1} \log q(x_t|x_{t-1}) - \sum_{t \geq 1} \log p_\theta(x_{t-1}|x_t) \right] \quad (29)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \sum_{t \geq 1} \log q(x_t|x_{t-1}) - \log p_\theta(x_{t-1}|x_t) \right] \quad (30)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \sum_{t \geq 1} \log \frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)} \right] \quad (31)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \sum_{t \geq 2} \log \frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)} \right] \quad (32)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \sum_{t \geq 2} \log \frac{q(x_{t-1}|x_t, x_0) q(x_t|x_0)}{q(x_{t-1}|x_0) p_\theta(x_{t-1}|x_t)} \right] \quad (33)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \sum_{t \geq 2} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} + \sum_{k \geq 2} \log \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} \right] \quad (34)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \log \frac{q(x_T|x_0)}{q(x_1|x_0)} + \sum_{t \geq 2} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} \right] \quad (35)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_T)) + \log \frac{q(x_T|x_0)}{p_\theta(x_0|x_1)} + \sum_{t \geq 2} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} \right] \quad (36)$$

$$= \mathbb{E}_q \left[ -\log(p_\theta(x_0|x_1)) + \log \frac{q(x_T|x_0)}{p_\theta(x_T)} + \sum_{t \geq 2} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} \right] \quad (37)$$

$$= \mathbb{E}_q \left[ D_{KL}(q(x_T|x_0) \| p(x_T)) \right. \\ \left. + \sum_{t \geq 2} D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) - \log(p_\theta(x_0|x_1)) \right] \quad (38)$$


---

---

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

$$L_T = D_{KL} (q(x_T|x_0) \parallel p(x_T)) \quad (39)$$

$$L_{1:T-1} = \sum_{t \geq 2} D_{KL} (q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \quad (40)$$

$$L_0 = -\log(p_\theta(x_0|x_1)) \quad (41)$$

It is worth noting that at equation ??, 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  $x_0$ .

Ho et al. [3] define it as:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I) \quad (42)$$

Using Bayes' theorem, we find:

$$q(x_{t-1}|x_t, x_0) = q(x_t|x_{t-1}, x_0) \frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} \quad (43)$$

Assuming that  $q(x_t|x_{t-1}, x_0) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I)$ , then:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I) \frac{\mathcal{N}(\sqrt{\bar{\alpha}_{t-1}}x_0, (1 - \bar{\alpha}_{t-1})I)}{\mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I)} \quad (44)$$

$$= \frac{1}{\sqrt{2\pi(1 - \bar{\alpha}_t)(1 - \bar{\alpha}_{t-1})(\sqrt{1 - \beta_t})}} e^{-\frac{1}{2} \left( \frac{(x_t - \sqrt{1 - \beta_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0)^2}{1 - \bar{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{1 - \bar{\alpha}_t} \right)} \quad (45)$$

*Note:* we rewrote  $q(x_t|x_{t-1}, x_0)$  as  $q(x_t|x_{t-1})$  since  $x_0$  adds no new information not already present in  $x_{t-1}$ .

We can ignore the constant factor in front and thus obtain:

$$= \exp \left( -\frac{1}{2} \left( \frac{x_t^2 - 2\sqrt{\alpha_t}x_{t-1}x_t + \alpha_t x_{t-1}^2}{\beta_t} + \frac{x_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}x_0x_{t-1} + \bar{\alpha}_{t-1}x_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{1 - \bar{\alpha}_t} \right) \right) \quad (46)$$

$$= \exp \left( -\frac{1}{2} \left( \frac{x_t^2}{\beta_t} - \frac{2\sqrt{\alpha_t}x_{t-1}x_t}{\beta_t} + \frac{\alpha_t x_{t-1}^2}{\beta_t} + \frac{x_{t-1}^2}{1 - \bar{\alpha}_{t-1}} - \frac{2\sqrt{\bar{\alpha}_{t-1}}x_0x_{t-1}}{1 - \bar{\alpha}_{t-1}} + \frac{\bar{\alpha}_{t-1}x_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{1 - \bar{\alpha}_t} \right) \right) \quad (47)$$

$$= \exp \left( -\frac{1}{2} \left( x_{t-1}^2 \left( \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) + x_{t-1} \left( \frac{-2\sqrt{\alpha_t}x_t}{\beta_t} - \frac{2\sqrt{\bar{\alpha}_{t-1}}x_0}{1 - \bar{\alpha}_{t-1}} \right) + C(x_t, x_0) \right) \right) \quad (48)$$


---

where:

$$C(x_t, x_0) = \frac{x_t^2}{\beta_t} + \frac{\bar{\alpha}_{t-1}x_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\alpha_t}x_0)^2}{1 - \bar{\alpha}_t} \quad (49)$$

Considering  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha}_t = \prod_{i=0}^t \alpha_i$ , we can simplify the expression to:

$$= \exp \left( -\frac{1}{2} \left( x_{t-1}^2 \left( \frac{\alpha_t}{1 - \alpha_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) - 2 \left( \frac{\sqrt{\alpha_t}x_t}{1 - \alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}x_0}{1 - \bar{\alpha}_{t-1}} \right) x_{t-1} + C(x_t, x_0) \right) \right) \quad (50)$$

$$= \exp \left( -\frac{1}{2} \left( x_{t-1}^2 \left( \frac{\alpha_t(1 - \bar{\alpha}_{t-1}) + (1 - \alpha_t)}{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})} \right) - 2 \left( \frac{\sqrt{\alpha_t}x_t}{1 - \alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}x_0}{1 - \bar{\alpha}_{t-1}} \right) x_{t-1} + C(x_t, x_0) \right) \right) \quad (51)$$

$$= \exp \left( -\frac{1}{2} \left( x_{t-1}^2 \left( \frac{\alpha_t - \alpha_t\bar{\alpha}_{t-1} + 1 - \alpha_t}{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})} \right) - 2 \left( \frac{\sqrt{\alpha_t}x_t}{1 - \alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}x_0}{1 - \bar{\alpha}_{t-1}} \right) x_{t-1} + C(x_t, x_0) \right) \right) \quad (52)$$

$$= \exp \left( -\frac{1}{2} \left( x_{t-1}^2 \left( \frac{1 - \bar{\alpha}_{t-1}}{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})} \right) - 2 \left( \frac{\sqrt{\alpha_t}x_t}{1 - \alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}x_0}{1 - \bar{\alpha}_{t-1}} \right) x_{t-1} + C(x_t, x_0) \right) \right) \quad (53)$$

$$= \exp \left( -\frac{1}{2} \left( \left( \frac{1 - \bar{\alpha}_{t-1}}{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})} \right) \left( x_{t-1}^2 - 2 \frac{\frac{\sqrt{\alpha_t}x_t}{1 - \alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}x_0}{1 - \bar{\alpha}_{t-1}}}{\frac{1 - \bar{\alpha}_{t-1}}{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}} x_{t-1} + \frac{C(x_t, x_0)}{\frac{1 - \bar{\alpha}_{t-1}}{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}} \right) \right) \right) \quad (54)$$

$$= \exp \left( -\frac{1}{2} \left( \left( \frac{1 - \bar{\alpha}_{t-1}}{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})} \right) \left( x_{t-1}^2 - 2 \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)x_0}{1 - \bar{\alpha}_t} + \frac{C(x_t, x_0)}{\frac{1 - \bar{\alpha}_{t-1}}{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}} \right) \right) \right) \quad (55)$$

As for the final term, since:

$$C(x_t, x_0) = \frac{x_t^2}{\beta_t} + \frac{\bar{\alpha}_{t-1}x_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\alpha_t}x_0)^2}{1 - \bar{\alpha}_t} \quad (56)$$

$$= \frac{x_t^2}{1 - \alpha_t} + \frac{\bar{\alpha}_{t-1}x_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{x_t^2 - 2\sqrt{\alpha_t}x_tx_0 + \alpha_tx_0^2}{1 - \bar{\alpha}_t} \quad (57)$$

We obtain:

$$C(x_t, x_0) / \frac{1 - \bar{\alpha}_{t-1}}{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})} \quad (58)$$

---


$$= \frac{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_{t-1}} \left( \frac{x_t^2}{1 - \alpha_t} + \frac{\bar{\alpha}_{t-1}x_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{x_t^2 - 2\sqrt{\alpha_t}x_tx_0 + \alpha_tx_0^2}{1 - \bar{\alpha}_t} \right) \quad (59)$$

$$= x_t^2 \left( \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} - \frac{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}{(1 - \bar{\alpha}_t)^2} \right) \quad (60)$$

$$\begin{aligned} &+ x_0^2 \left( \frac{(1 - \alpha_t)(\bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} - \frac{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})\bar{\alpha}_t}{(1 - \bar{\alpha}_t)^2} \right) \\ &+ \frac{2x_tx_0\sqrt{\bar{\alpha}_t}(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}{(1 - \bar{\alpha}_t)^2} \\ &= x_t^2 \left( \frac{(1 - \bar{\alpha}_t)(1 - \bar{\alpha}_{t-1}) - (1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}{(1 - \bar{\alpha}_t)^2} \right) \quad (61) \end{aligned}$$

$$\begin{aligned} &+ x_0^2 \left( \frac{(1 - \bar{\alpha}_t)(1 - \alpha_t)\bar{\alpha}_{t-1} - (1 - \alpha_t)(1 - \bar{\alpha}_{t-1})\bar{\alpha}_t}{(1 - \bar{\alpha}_t)^2} \right) \\ &+ \frac{2x_tx_0\sqrt{\bar{\alpha}_t}(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}{(1 - \bar{\alpha}_t)^2} \end{aligned}$$

$$= x_t^2 \frac{(1 - \bar{\alpha}_{t-1})^2 \alpha_t}{(1 - \bar{\alpha}_t)^2} + x_0^2 \frac{(1 - \bar{\alpha}_t)^2 \bar{\alpha}_{t-1}}{(1 - \bar{\alpha}_t)^2} + \frac{2x_tx_0\sqrt{\bar{\alpha}_t}(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}{(1 - \bar{\alpha}_t)^2} \quad (62)$$

$$= \frac{(x_t\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1}) + x_0\sqrt{\bar{\alpha}_{t-1}}(1 - \bar{\alpha}_t))^2}{(1 - \bar{\alpha}_t)^2} \quad (63)$$

$$= \left( \frac{(x_t\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1}) + x_0\sqrt{\bar{\alpha}_{t-1}}(1 - \bar{\alpha}_t))}{(1 - \bar{\alpha}_t)} \right)^2 \quad (64)$$


---

---

We can now replace this value into the original expression:

$$q(x_{t-1}|x_t, x_0) = \exp \left( -\frac{1}{2} \left( \left( \frac{1}{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}}} \right) \right. \right. \quad (65)$$

$$\left( x_{t-1}^2 - 2 \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t} \right. \\ \left. \left. + \left( \frac{(x_t\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1}) + x_0\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t))}{(1-\bar{\alpha}_t)} \right)^2 \right) \right) \quad (66)$$

$$= \exp \left( -\frac{1}{2} \left( \frac{1}{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}}} \right) \right. \quad (67)$$

$$\left. \left( x_{t-1} - \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t} \right)^2 \right)$$

$$= \mathcal{N} \left( x_{t-1}; \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t}, \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} I \right) \quad (68)$$

$$= \mathcal{N} \left( x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I \right) \quad (69)$$

We have therefore obtained  $\tilde{\mu}(x_t, x_0)$  and  $\tilde{\beta}_t$  mentioned in Ho et al.'s [3] paper. The authors also rearrange  $\tilde{\mu}(x_t, x_0)$  in order to remove the dependency on  $x_0$ :

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \quad (70)$$

$$\Leftrightarrow x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}} \quad (71)$$

$$\Rightarrow \tilde{\mu}(x_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t) \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}}}{1-\bar{\alpha}_t} \quad (72)$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + (1-\alpha_t) \left( x_t - \sqrt{1-\bar{\alpha}_t}\epsilon \right) \frac{\sqrt{\bar{\alpha}_{t-1}}}{\sqrt{\bar{\alpha}_t}}}{1-\bar{\alpha}_t} \quad (73)$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t}{1-\bar{\alpha}_t} + \frac{(1-\alpha_t) \left( x_t - \sqrt{1-\bar{\alpha}_t}\epsilon \right) \frac{1}{\sqrt{\bar{\alpha}_t}}}{1-\bar{\alpha}_t} \quad (74)$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t + \frac{(1-\alpha_t)}{(1-\bar{\alpha}_t)\sqrt{\bar{\alpha}_t}} x_t - \frac{(1-\alpha_t)(\sqrt{1-\bar{\alpha}_t})}{(1-\bar{\alpha}_t)\sqrt{\bar{\alpha}_t}} \epsilon \quad (75)$$

$$= \left( \frac{\sqrt{\alpha_t}\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_{t-1})}{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_t)} + \frac{(1-\alpha_t)}{(1-\bar{\alpha}_t)\sqrt{\bar{\alpha}_t}} \right) x_t - \frac{(1-\alpha_t)(\sqrt{1-\bar{\alpha}_t})}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{(1-\bar{\alpha}_t)}\sqrt{\bar{\alpha}_t}} \epsilon \quad (76)$$

$$= \left( \frac{\alpha_t(1-\bar{\alpha}_{t-1}) + (1-\alpha_t)}{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_t)} \right) x_t - \frac{(1-\alpha_t)}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{\bar{\alpha}_t}} \epsilon \quad (77)$$

$$= \left( \frac{\alpha_t - \bar{\alpha}_t + 1 - \alpha_t}{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_t)} \right) x_t - \frac{1-\alpha_t}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{\bar{\alpha}_t}} \epsilon \quad (78)$$


---

---


$$= \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1 - \alpha_t}{\sqrt{(1 - \bar{\alpha}_t)} \sqrt{\alpha_t}} \epsilon \quad (79)$$

$$= \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right) \quad (80)$$

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 Kullback-Leibler Divergence

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

$$L_{VLB} = \mathbb{E}_q \left[ \sum_{t \geq 2} D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) - \log(p_\theta(x_0|x_1)) \right] \quad (81)$$

## 2.2.2 Simplified training objective

Ho et al. [3] simplify the training objective by making several changes in order to obtain an easy quantity for minimising the loss.

First of all,  $L_{1:T-1}$  can be written as such:

$$L_{1:T-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) \quad (82)$$

$$= D_{KL}(\mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, t), \sigma_t^2 I) \| \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)) \quad (83)$$

The Kullback-Leibler divergence between the two multivariate Gaussian distributions is given by:

$$D_{KL}(\mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, t), \sigma_t^2 I) \| \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)) \quad (84)$$

$$= \frac{1}{2} \left( \text{tr}((\sigma_t^2 I)^{-1}(\sigma_t^2 I)) - d + (\mu_\theta - \tilde{\mu}_t)^T (\sigma_t^2 I)^{-1} (\mu_\theta - \tilde{\mu}_t) + \log \frac{\det(\sigma_t^2 I)}{\det(\sigma_t^2 I)} \right) \quad (85)$$

$$= \frac{1}{2} \left( d - d + (\mu_\theta - \tilde{\mu}_t)^T (\sigma_t^2 I)^{-1} (\mu_\theta - \tilde{\mu}_t) + \log 1 \right) \quad (86)$$

$$= \frac{1}{2} \left( (\mu_\theta - \tilde{\mu}_t)^T (\sigma_t^2 I)^{-1} (\mu_\theta - \tilde{\mu}_t) \right) \quad (87)$$

$$= \frac{1}{2\sigma_t^2} \left( (\mu_\theta - \tilde{\mu}_t)^T (\mu_\theta - \tilde{\mu}_t) \right) \quad (88)$$

$$= \frac{1}{2\sigma_t^2} \|\mu_\theta - \tilde{\mu}_t\|_2^2 \quad (89)$$

However, Ho et al. [3] rewrite this formula so that it depends only on the noise  $\epsilon$ :

$$\frac{1}{2\sigma_t^2} \|\mu_\theta - \tilde{\mu}_t\|_2^2 = \frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right) - \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon_\theta \right) \right\|_2^2 \quad (90)$$

$$= \frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \frac{1 - \alpha_t}{\sqrt{(1 - \bar{\alpha}_t)}} (\epsilon - \epsilon_\theta) \right\|_2^2 \quad (91)$$

$$= \frac{1}{2\sigma_t^2} \frac{1}{\alpha_t} \frac{(1 - \alpha_t)^2}{1 - \bar{\alpha}_t} \|\epsilon - \epsilon_\theta\|_2^2 \quad (92)$$

$$= \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \|\epsilon - \epsilon_\theta\|_2^2 \quad (93)$$


---

---

Furthermore, they found that simply ignoring the factor in front gave better results, giving us:

$$L_{1:T-1} = \|\epsilon - \epsilon_\theta\|_2^2 \quad (94)$$

As for the second term  $L_0$ , the authors define  $p_\theta(x_0|x_1)$  as:

$$p_\theta(x_0|x_1) = \prod_{i=1}^D \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2) dx \quad (95)$$

where  $\delta_+(x)$  and  $\delta_-(x)$  are defined as:

$$\delta_+(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases} \quad (96)$$

$$\delta_-(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases} \quad (97)$$

with  $D$  the data dimensionality and  $i$  one coordinate from the image.

This is a decoder for the last step in order to get pixel values between  $\{0, 1, \dots, 255\}$  from the values in  $[-1, 1]$  used for the neural network [3, 11]

However the authors also ignored the entire term, giving us:

$$L_{\text{simple}}(\theta) = \mathbb{E}_q [\|\epsilon - \epsilon_\theta\|_2^2] \quad (98)$$

### 2.2.3 Model architecture

Ho et al. [3] as well as the following papers [16, 11, 2] use a U-Net style architecture for the neural network to estimate the noise.

The U-Net architecture is a convolutional neural network (CNN) that was originally designed for image segmentation in biomedical imaging [13]. It consists of an encoder (or contracting path) and a decoder (or expansive path) that are connected by a bottleneck.

The encoder takes in the noisy image as input and repeatedly applies unpadded convolutions followed by max pooling layers in order to reduce the spatial dimensions of the image. [13]

The bottleneck contains 3 unpadded convolutions each followed by a rectified linear unit (ReLU) activation function. [13]

The decoder then applies up-convolutions followed by unpadded convolutions to increase the spatial dimensions of the image, with at each step a concatenation of the feature maps from the encoder. [13]

However, Ho et al. [3] made some changes to the original U-Net architecture.

First, they state that they used 4 feature map resolutions for their  $32 \times 32$  resolution model, and 6 for the  $256 \times 256$  resolution model. For each of these feature maps, they used two convolutional residual blocks similar of those of a Wide ResNet [25] consisting of a convolutional layer, a ReLU activation function, a convolutional layer, a group normalisation layer, and another ReLU activation function.

The "residual" part of the block comes from the fact that the result at the end of the block on the encoder side is concatenated with the result at the end of the block on the decoder side in order to consider features that could have been lost during the downsampling. [8]

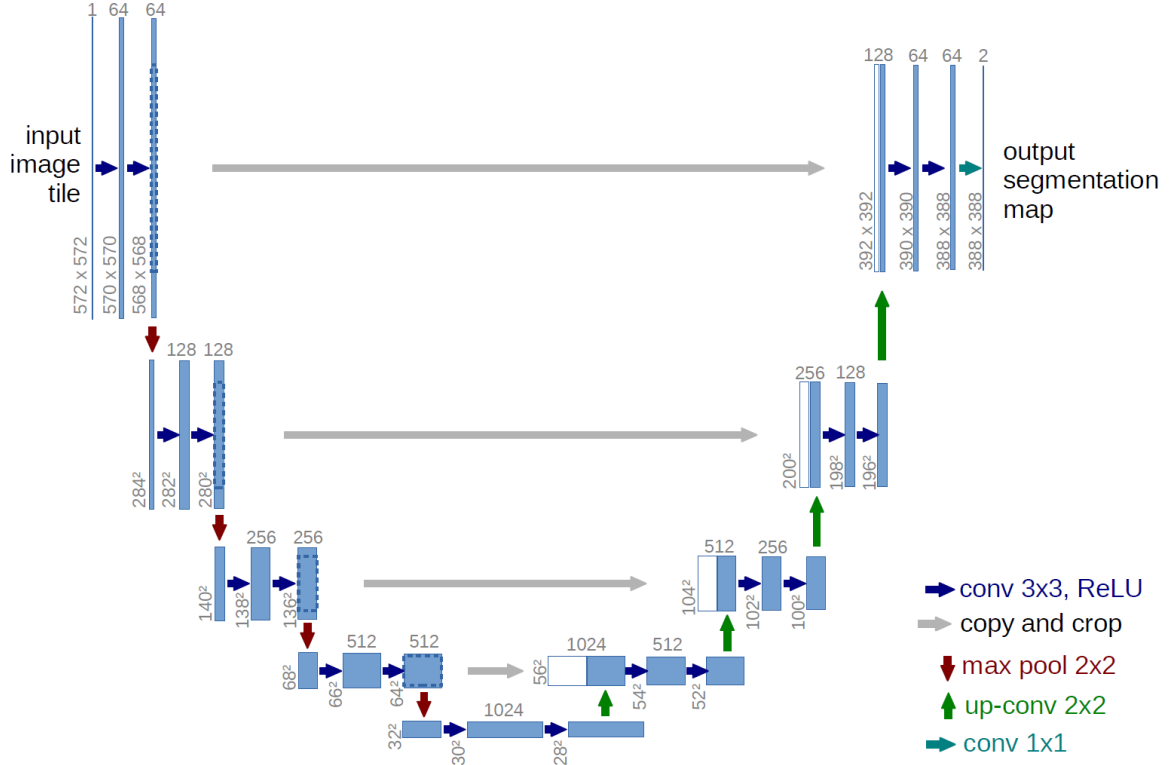


Figure 1: Original U-Net architecture [13]

They used group normalisation [24] rather than weight normalisation [14], which is a normalisation technique that divides the channels into groups and computes the mean and standard deviation for each group [3].

Finally, they state that they used self-attention layers from the Transformer architecture [20] at the  $16 \times 16$  resolution between the two convolutional residual blocks. [3]

Since the model needs to be able to predict the noise at any timestep  $t$ , the authors added a time embedding to the input of the network using the sinusoidal positional embedding from the Transformer architecture [20], which allows the parameters to be shared across all timesteps.

The sinusoidal positional embedding is given by [20]:

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (99)$$

$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (100)$$

where  $pos$  is the position and  $i$  is the dimension

Ho et al. [3] present two algorithms: one for training the model and one for sampling from the model:



---

**Algorithm 1** Training

---

```

1: repeat
2:    $x_0 \sim q(x_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\epsilon \sim \mathcal{N}(0, I)$ 
5:   Take gradient descent step on:
6:      $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}x_0 + \sqrt{1 - \alpha_t}\epsilon, t)\|^2$ 
7: until converged

```

---

**Algorithm 2** Sampling

---

```

1:  $x_T \sim \mathcal{N}(0, I)$ 
2: for  $t = T, \dots, 1$  do
3:    $z \sim \mathcal{N}(0, I)$  if  $t > 0$ , else  $z = 0$ 
4:    $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(x_t, t) \right) + \sigma_t z$ 
5: end for
6: return  $x_0$ 

```

---

## 2.3 Score-based formulation

The DDPM paper [3] allows us to predict either the next denoised image, the mean of the next denoised image or the noise itself.

However they mention another formulation of the DDPM model that is equivalent: a score-based generative modeling version

### 2.3.1 Score matching

This formulation uses score matching [5], either denoising score matching [21] or sliced score matching [19].

Essentially, score matching avoids optimizing the likelihood which can sometimes be difficult to find because of normalisation constants, and instead optimises a score function which is the gradient of the log-likelihood.

Suppose we have a probability density function  $p_{\theta}(x)$  that models an i.i.d. dataset  $\{x_i\}_{i=1}^N$  of the p.d.f.  $p(x)$ .

Let  $f_{\theta}(x)$  be a real valued function which is an unnormalized density function and  $Z_{\theta} = \int e^{-f_{\theta}(x)} dx$  be the normalisation constant such that  $\int p_{\theta}(x) dx = 1$ .

Then  $p_{\theta}(x)$  is defined as:

$$p_{\theta}(x) = \frac{e^{-f_{\theta}(x)}}{Z_{\theta}} \quad (101)$$

However for a lot of models,  $Z_{\theta}$  cannot be computed. [10]

The score function  $s_{\theta} : \mathbb{R}^D \rightarrow \mathbb{R}^D$  is defined as [18]:

$$s_{\theta}(x) = \nabla_x \log p(x) \quad (102)$$

If we try to train a model to optimise this score function, we get:

$$s_{\theta}(x) = \nabla_x \log p_{\theta}(x) \quad (103)$$

$$= \nabla_x \log \frac{e^{-f_{\theta}(x)}}{Z_{\theta}} \quad (104)$$

$$= \nabla_x (-f_{\theta}(x) - \log Z_{\theta}) \quad (105)$$

$$= -\nabla_x f_{\theta}(x) - \nabla_x \log Z_{\theta} \quad (106)$$

$$= -\nabla_x f_{\theta}(x) \quad (107)$$

Since  $Z_{\theta}$  is a constant with respect to  $x$ ,  $\nabla_x \log Z_{\theta} = 0$  which means we don't need to compute the intractable normalisation constant  $Z_{\theta}$ .

---

This means we could train the score network  $s_\theta$  to match the true score function  $s(x) = \nabla_x \log p(x)$ . The model should then optimise the following Fisher divergence [5, 10]:

$$\frac{1}{2} \mathbb{E}_{x \sim p(x)} [\|s_\theta(x) - s(x)\|_2^2] \quad (108)$$

However, since we don't know the true score function  $s(x)$  which is the score function of the true data distribution  $p(x)$ , another formulation was shown by Hyvärinen [5] to be equivalent:

$$\mathbb{E}_{x \sim p(x)} \left[ \text{tr}(\nabla_x s_\theta(x)) + \frac{1}{2} \|s_\theta(x)\|_2^2 \right] \quad (109)$$

This is the score matching objective we want to minimise. [5].

It can also be computed using the datapoints with the following formula [5]:

$$\approx \frac{1}{N} \sum_{i=1}^N \text{tr}(\nabla_x s_\theta(x_i)) + \frac{1}{2} \|s_\theta(x_i)\|_2^2 \quad (110)$$

A problem that arises with this formulation is that the Jacobian matrix  $\nabla_x s_\theta(x)$  is not suitable for deep neural networks as it requires backpropagation to compute all its diagonal elements to find the trace [18, 19]. To solve this problem, Song et al. [18] propose either denoising score matching or sliced score matching.

**Denoising score matching** [21] is a method that slightly noises a datapoint and then score matches the denoised datapoint. This is done by convolving the true data distribution  $p$  with a noising kernel  $q_\sigma$  but such that  $p(x) \approx q_\sigma(x)$ . Given a datapoint  $x$ , we get a noised datapoint  $\tilde{x}$  by sampling from  $q_\sigma(\tilde{x}|x)$ .

The score matching now estimates the score of the noised data distribution  $q_\sigma(\tilde{x})$ , which leads us to the following objective [18, 21]:

$$\frac{1}{2} \mathbb{E}_{q_\sigma(\tilde{x}|x)p(x)} [\|s_\theta(\tilde{x}) - \nabla_x \log q_\sigma(\tilde{x}|x)\|_2^2] \quad (111)$$

**Sliced score matching** [19] projects the vectors of the vector field  $s_\theta(x)$  onto random directions to obtain a scalar field. This will approximate the trace of the Jacobian matrix and it becomes much easier to compute. [18]

We get the following objective [19]:

$$\mathbb{E}_{p_v} \mathbb{E}_{x \sim p(x)} \left[ v^T \nabla_x s_\theta(x) v + \frac{1}{2} \|s_\theta(x)\|_2^2 \right] \quad (112)$$

The approximation of the Jacobian matrix  $v^T \nabla_x s_\theta(x) v$  can be rewritten as [19]:

$$v^T \nabla_x s_\theta(x) v = v^T \nabla_x (v^T s_\theta(x)) \quad (113)$$

The gradient of the inner product  $\nabla_x (v^T s_\theta(x))$  requires only one backpropagation pass and requires another inner product with  $v^T$  to approximate the trace. [19]

### 2.3.2 Sampling

Using the score function mentioned above, we can sample from the model by using Langevin dynamics [18, 22], which will give us the definition of score-based generative modeling. [18]

---

Let  $x_0 \sim \pi(x)$  where  $\pi(x)$  is an arbitrary distribution (such as a Gaussian distribution),  $\epsilon > 0$  a fixed step size and  $z_t \sim \mathcal{N}(0, I)$ .

Using Stochastic Gradient Langevin Dynamics (SGLD) [18, 22], we can recursively sample using the following formula:

$$x_t = x_{t-1} + \frac{\epsilon}{2} \nabla_x \log p(x_{t-1}) + \sqrt{\epsilon} z_t \quad (114)$$

When  $T \rightarrow \infty$  and  $\epsilon \rightarrow 0$ , the distribution of  $x_T$  will converge to the true data distribution  $p(x)$  therefore the sample will become one from the true data distribution. [18, 22]

$z_t$  is used to add some Gaussian noise at each step because if there was no stochasticity at each step, the sampling would become deterministic and all the samples would essentially be the same.

Since we don't know the true data distribution  $p(x)$ , we can use the score function  $s_\theta(x)$  trained to match the true score function  $s(x)$  to sample from the model.

$$x_t = x_{t-1} + \frac{\epsilon}{2} s_\theta(x_{t-1}) + \sqrt{\epsilon} z_t \quad (115)$$

There is however a problem with this method: the score function is well approximated at high density regions where there are a lot of data points but not at low density regions. [18]

Song and Ermon [18] show this in figure ?? by comparing the real score function  $\nabla_x \log p(x)$  and the estimated score function  $s_\theta(x)$  for a mixture of two Gaussian distributions.

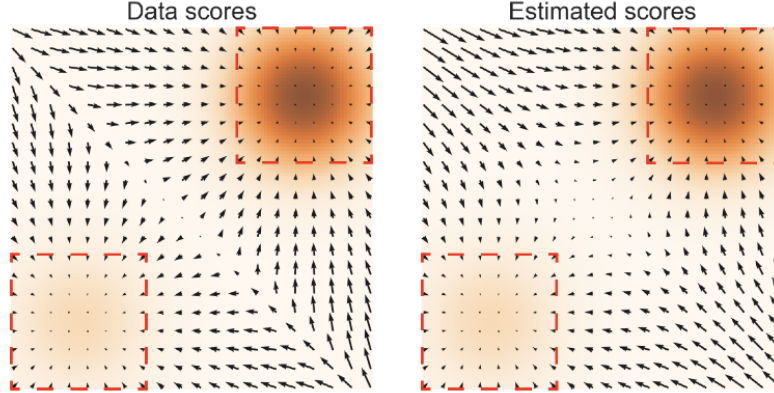


Figure 2: Data scores represent the real score function  $\nabla_x \log p(x)$  and estimated scores represent the estimated score function  $s_\theta(x)$ . We can see that the regions in the red rectangles are the darker high density regions where the score function is well approximated, while the lighter lower density regions outside the red rectangles were not well estimated. [18]

This means that when sampling using Langevin dynamics, the samples will ultimately be incorrect. [18] To correct this, Song and Ermon [18] use annealed Langevin dynamics.

The idea is to add noise to the data points in order to improve the score function estimation in low density regions. [17] However adding too much noise will make the score function estimation worse in high density regions. [17].

Therefore, they control the amount of noise added using a scale of different standard deviations  $\sigma_i$  with  $i = 1, 2, \dots, L$  such that  $\sigma_1 < \sigma_2 < \dots < \sigma_L$ . [17, 18]

To sample, the same Langevin dynamics method is used going from  $i = L, L-1, \dots, 1$  with  $\sigma_i$  decreasing at each step. This is the annealed Langevin dynamics method they used, and the results can be seen in figure ??, where we clearly see that the annealed Langevin dynamics samples are much more similar to the original data samples than the simple Langevin dynamics samples. [17, 18]

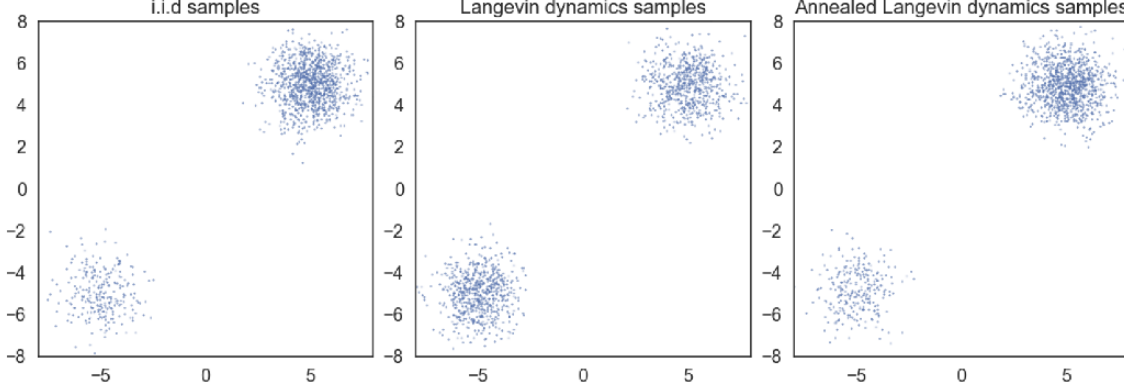


Figure 3: Left: Samples from data distribution. Middle: Samples from model using Langevin dynamics. Right: Samples from model using annealed Langevin dynamics. [18]

### 2.3.3 Noise Conditional Score Networks

Noise Conditional Score Networks (NCSN) are a score-based generative model that use a neural network to estimate the score function of the previously mentioned noised distribution and then use annealed Langevin dynamics to sample from the model. [18]

Given the increasing sequence of standard deviations  $\sigma_i$  with  $i = 1, 2, \dots, L$ , the perturbed data distribution is defined as [18]:

$$q_{\sigma_i}(x) = \int p(t) \mathcal{N}(x; t, \sigma_i^2) dt \quad (116)$$

Song and Ermon [18] then define the Noise Conditional Score Network  $s_\theta$  that estimates the scores of all the perturbed data distributions  $q_{\sigma_i}(x)$  given a data point  $x$ :

$$s_\theta : \mathbb{R}^D \times \mathbb{R} \rightarrow \mathbb{R}^D \quad (117)$$

$$s_\theta(x, \sigma) \approx \nabla_x \log q_\sigma(x) \quad \forall \sigma \in \{\sigma_1, \sigma_2, \dots, \sigma_L\} \quad (118)$$

To train the NCSN, the authors chose denoising score matching [21] as its goal is to estimate the score function of a noised data distribution while being quicker than sliced score matching [19]. [18]

Let  $q_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x}|x, \sigma^2, I)$  be the noise distribution that noised the data point  $x$  to get the noised data point  $\tilde{x}$ .

Its score function is given by [18]:

$$\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) = -\frac{\tilde{x} - x}{\sigma^2} \quad (119)$$

Let  $\lambda(\sigma_i)$  be a positive coefficient function, often chosen to be  $\lambda(\sigma_i) = \sigma_i^2$  [17, 18]  
The denoising score matching objective is given by [18]:

$$\mathcal{L}(\theta, \{\sigma_i\}_{i=1}^L) = \frac{1}{2L} \sum_{i=1}^L \lambda(\sigma_i) \mathbb{E}_p(x) \mathbb{E}_{\tilde{x} \sim \mathcal{N}(x, \sigma_i^2 I)} \left[ \left\| s_\theta(\tilde{x}, \sigma_i) + \frac{\tilde{x} - x}{\sigma_i^2} \right\|_2^2 \right] \quad (120)$$

To sample from the model, Song and Ermon [18] define an algorithm that uses the previously mentioned annealed Langevin dynamics:

---

**Algorithm 3** Sampling from NCSN

---

**Require:**  $\{\sigma_i\}_{i=1}^L, \epsilon, T$

```

1:  $\tilde{x}_0 \sim \pi(x)$ 
2: for  $i = 1, 2, \dots, L$  do
3:    $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ 
4:   for  $t = 1, 2, \dots, T$  do
5:      $z_t \sim \mathcal{N}(0, I)$ 
6:      $\tilde{x}_t \leftarrow \tilde{x}_{t-1} + \frac{\alpha_i}{2} s_\theta(\tilde{x}_{t-1}, \sigma_i) + \sqrt{\alpha_i} z_t$ 
7:   end for
8:    $\tilde{x}_0 \leftarrow \tilde{x}_T$ 
9: end for
10: return  $\tilde{x}_T$ 
```

---

### 2.3.4 Equivalence between DDPM and NCSN

Ho et al.'s DDPM paper [3] showed that the DDPM model is equivalent to the NCSN model with denoising score matching and annealed Langevin dynamics.

This can be seen by the fact that sampling algorithm ?? of the DDPM paper is in fact very similar to the sampling algorithm ?? of the NCSN paper, but rather than estimating a score, the DDPM model estimates the noise. [3]

Similarly, the score function  $s_\theta(x, \sigma)$  can be rewritten in terms of the DDPM's forward process as [23]:

$$s_\theta(x, \sigma) \approx \nabla_x \log q_\sigma(x) \Leftrightarrow s_\theta(x_t, t) \approx \nabla_{x_t} \log q(x_t) \quad (121)$$

$$= \mathbb{E}_{q(x_0)} [\nabla_{x_t} \log q(x_t | x_0)] \quad (122)$$

$$= \mathbb{E}_{q(x_0)} [\nabla_{x_t} \log \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I)] \quad (123)$$

$$= \mathbb{E}_{q(x_0)} \left[ \nabla_{x_t} \log e^{-\frac{1}{2} \left( \frac{x_t - \sqrt{\bar{\alpha}_t} x_0}{\sqrt{1 - \bar{\alpha}_t}} \right)^T \left( \frac{x_t - \sqrt{\bar{\alpha}_t} x_0}{\sqrt{1 - \bar{\alpha}_t}} \right)} \right] \quad (124)$$

$$= \mathbb{E}_{q(x_0)} [\Sigma^{-1} (x_t - \sqrt{\bar{\alpha}_t} x_0)] \quad (125)$$

$$= \mathbb{E}_{q(x_0)} \left[ \Sigma^{-1} \left( x_t - \sqrt{\bar{\alpha}_t} \left( \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta}{\sqrt{\bar{\alpha}_t}} \right) \right) \right] \quad (126)$$

$$= \mathbb{E}_{q(x_0)} [\Sigma^{-1} (\sqrt{1 - \bar{\alpha}_t} \epsilon_\theta)] \quad (127)$$

$$= -\frac{1}{1 - \bar{\alpha}_t} \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta \quad (128)$$

$$= -\frac{\epsilon_\theta}{\sqrt{1 - \bar{\alpha}_t}} \quad (129)$$

We can see that this quantity is used in sampling algorithm ?? of the DDPM paper [3].

---

### 3 Improvements upon DDPMs

As mentioned previously, the DDPM paper [3] had certain choices that limited its performance. Several papers [11, 16] following it have improved upon the original DDPM in various ways.

#### 3.1 Improved likelihood

Ho et al. [3] admit that their model, despite its high sample quality, did not have a competitive log-likelihood in comparison to other likelihood-based models such as VAEs and autoregressive models [11]

The *Improved Denoising Diffusion Probabilistic Models* paper published by Nichol et al. [11] in 2021 implemented certain changes in order to improve the log-likelihood of the model not only on simpler datasets such as CIFAR-10 [7] that the original DDPM was trained on, but also on more complex datasets such as ImageNet. [6, 12, 11].

In order to improve the log-likelihood, Nichol et al. [11] started by investigating the fixed variance  $\Sigma_\theta(x_t, t) = \sigma_t^2 I = \beta_t I$  of the original DDPM and whether or not it could be worth learning it. Recall that Ho et al. [3] saw no difference in sample quality when using a learned variance  $\tilde{\beta}_t$  and therefore opted for using the fixed variance  $\beta_t$ .

Nichol et al. [11] came to the same conclusion, finding that as the number of steps  $T$  increased (the authors also mentioned that they used  $T = 4000$  for their models rather than  $T = 1000$  from the original paper), the difference between  $\beta_t$  and  $\tilde{\beta}_t$  remained very small. However, they were still interested in the possibility of learning the variance in order to improve the log-likelihood. [11].

For this, they used an interpolation approach, where they parametrised the variance as an interpolation between the fixed variance  $\beta_t$  and a learned variance  $\tilde{\beta}_t$  in the log domain, with  $v$  an output vector from the model containing one component per dimension:

$$\Sigma_\theta(x_t, t) = \exp \left( v \log \beta_t + (1 - v) \log \tilde{\beta}_t \right) \quad (130)$$

As the simplified training objective  $L_{\text{simple}}$  used in the original DDPM paper [3] did not include the variance, Nichol et al. [11] introduced a new training objective  $L_{\text{hybrid}}$ , defined as:

$$L_{\text{hybrid}} = L_{\text{simple}} + \lambda L_{\text{VLB}} \quad (131)$$

with  $\lambda = 0.001$  in order to keep  $L_{\text{simple}}$  as the main training objective. [11]

At first, they found that the hybrid objective  $L_{\text{hybrid}}$  achieved a better log-likelihoods and was easier to optimise than just  $L_{\text{VLB}}$ , contrary to what they had expected. [11]

However, they obtained the best log-likelihoods by optimising  $L_{\text{VLB}}$  directly but with importance sampling, which is a technique where samples of a difficult distribution are taken from a distribution that is easier to compute and then the original distribution is approximated by weighted samples. [11]

They defined it as:

$$L_{\text{VLB}} = \mathbb{E}_{t \sim p_t} \left[ \frac{L_t}{p_t} \right] \quad \text{where } p_t \propto \sqrt{\mathbb{E}[L_t^2]} \text{ and } \sum p_t = 1 \quad (132)$$

Another change that Nichol et al. [11] made was to use a cosine noising schedule rather than the linear noising schedule used in the original DDPM paper [3].

They explain this change by the fact that towards the end of the forward process, the image is already very noisy and yet more noise is added while not really improving the model, particularly for  $32 \times 32$  and  $64 \times 64$  images. [11]

In effect, the input images get destroyed by noise very quickly with the linear noising schedule converging to zero rapidly. [11]

They define the cosine schedule as:

$$f(t) = \cos\left(\frac{\frac{t}{T} + s}{1 + s} \cdot \frac{\pi}{2}\right)^2 \quad (133)$$

$$\bar{\alpha}_t = \frac{f(t)}{f(0)} \quad (134)$$

$$\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}} \quad (135)$$

$T$  is the total number of timesteps,  $t$  is the timestep and  $s$  is a small offset used to prevent  $\beta_t$  from being too small when  $t = 0$  in order to allow the noise  $\epsilon$  to be predicted more accurately at the beginning of the process. [11].

With these changes, Nichol et al. [11] were able to achieve a negative log-likelihood (NLL) of 2.94 bits/dim on CIFAR-10 [7] and 3.53 bits/dim on ImageNet [12], compared to the original DDPM paper [3] which had a NLL of 3.70 on CIFAR-10 and 3.77 on ImageNet. [11].

Model	ImageNet	CIFAR
Glow (Kingma & Dhariwal, 2018)	3.81	3.35
Flow++ (Ho et al., 2019)	3.69	3.08
PixelCNN (van den Oord et al., 2016c)	3.57	3.14
SPN (Menick & Kalchbrenner, 2018)	3.52	-
NVAE (Vahdat & Kautz, 2020)	-	2.91
Very Deep VAE (Child, 2020)	3.52	2.87
PixelSNAIL (Chen et al., 2018)	3.52	2.85
Image Transformer (Parmar et al., 2018)	3.48	2.90
Sparse Transformer (Child et al., 2019)	3.44	<b>2.80</b>
Routing Transformer (Roy et al., 2020)	<b>3.43</b>	-
DDPM (Ho et al., 2020)	3.77	3.70
DDPM (cont flow) (Song et al., 2020b)	-	2.99
Improved DDPM (ours)	<b>3.53</b>	<b>2.94</b>

Figure 4: Comparison of NLLs of different models [11]

Moreover, Nichol et al. [11] were able to improve sampling speed due to the aforementioned changes, even though they trained their model with  $T = 4000$  timesteps.

They were able to train their model with 4000 timesteps, but use only 100 timesteps for sampling which brought them near-optimal FID (Fréchet Inception Distance) scores when using the  $L_{\text{hybrid}}$  training objective. [11]

They had tried to reduce the number of sampling steps with the original fixed variance versions from the DDPM paper [3] but found that the sample quality degraded a lot more. [11]

---

### 3.2 Denoising Diffusion Implicit Models (DDIMs)

As mentioned previously, the DDPM paper [3] has a forward process and a reverse process which are both Markov processes.

While the forward process can be easily calculated for a specific timestep  $t$  with equation ??, the reverse process still requires 1000 timesteps in the case of Ho et al.'s [3] model, and Nichol et al.'s model requires 100 and 4000 timesteps for sampling and training respectively. [11]

This is time-consuming and computationally expensive compared to GANs which only require one forward pass through the network rather than thousands to produce a sample. [16]

To address this issue, Song et al. [16] introduced the Denoising Diffusion Implicit Models (DDIMs) in 2020.

The main objective of DDIMs was to accelerate sampling by making the forward process non-Markovian, which would allow the reverse process to require less iterations. [16]

Song et al. [16] define a family  $\mathcal{Q}$  of forward processes, each indexed by  $\sigma \in \mathbb{R}_+^T$ :

$$q_\sigma(x_{1:T}|x_0) = q_\sigma(x_T|x_0) \prod_{t=2}^T q_\sigma(x_{t-1}|x_t, x_0) \quad (136)$$

where

$$q_\sigma(x_T|x_0) = \mathcal{N}(\sqrt{\alpha_T}x_0, (1 - \alpha_T)I) \quad (137)$$

and for all  $t > 1$ :

$$q_\sigma(x_{t-1}|x_t, x_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}x_0 + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \frac{x_t - \sqrt{\alpha_t}x_0}{\sqrt{1 - \alpha_t}}, \sigma_t^2 I\right) \quad (138)$$

It is interesting to note that when  $\sigma_t^2 = (1 - \alpha_t)(1 - \bar{\alpha}_{t-1})/(1 - \bar{\alpha}_t)$ , we obtain equation ??, which was the definition of the forward process in the original DDPM paper [3, 16].

Song et al. [16] also note when  $\sigma_t^2 = 0$ , then  $x_{t-1}$  is fixed and known in advance.

Therefore, we can see that as Song et al. [16] state, the value of  $\sigma$  controls the stochasticity (or more simply, the randomness) of the forward process.

The authors use Bayes' rule similarly to the original DDPM paper [3] in equation ??, but instead to find the forward process  $q_\sigma(x_t|x_{t-1}, x_0)$ :

$$q_\sigma(x_t|x_{t-1}, x_0) = \frac{q_\sigma(x_{t-1}|x_t, x_0) q_\sigma(x_t|x_0)}{q_\sigma(x_{t-1}|x_0)} \quad (139)$$

Since  $q_\sigma(x_t|x_{t-1}, x_0)$  means that  $x_t$  not only depends on  $x_{t-1}$  but also on  $x_0$ , we see that the forward process is no longer Markovian, which was what the authors were aiming to achieve. [16]

However, since the reverse process in the original DDPM paper [3] was an estimation of the forward process which was Markovian, a new reverse process was defined.

Their idea of the reverse process is to sample an image  $x_T$  like in the original DDPM paper [3], predict  $x_0$  from  $x_T$  and then use this prediction of  $x_0$  to sample  $x_{T-1}$ . [16]

This is repeated until we reach  $x_1$  and then  $x_0$  is predicted from  $x_1$ . [16]

Similar to equation ?? from the DDPM paper, Song et al. [16] define the prediction of  $x_0$  from  $x_t$  as:

$$f_\theta^{(t)}(x_t) = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \epsilon(x_t, t)}{\sqrt{\bar{\alpha}_t}} \quad (140)$$



---

Given this, the reverse process is defined as [16]:

$$p_\theta(x_T) = \mathcal{N}_D(0, I) \quad (141)$$

$$p_\theta^{(t)}(x_{t-1}|x_t) = \begin{cases} \mathcal{N}(f_\theta^{(1)}(x_1), \sigma_1^2 I) & \text{if } t = 1 \\ q_\sigma(x_{t-1}|x_t, f_\theta^{(t)}(x_t)) & \text{if } t > 1 \end{cases} \quad (142)$$

Therefore,  $x_{t-1}$  is sampled from  $x_t$  [16]:

$$x_{t-1} = \sqrt{\alpha_{t-1}} \left( \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \epsilon(x_t, t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \epsilon_\theta(x_t, t) + \sigma_t \epsilon_t \quad (143)$$

In order to train the model, Song et al. [16] define the following *variational inference objective*:

$$J_\sigma(\epsilon_\theta) = \mathbb{E}_{q_\sigma(x_{0:T})} [\log q_\sigma(x_{1:T}|x_0) - \log p_\theta(x_{0:T})] \quad (144)$$

$$= \mathbb{E}_{q_\sigma(x_{0:T})} \left[ \log q_\sigma(x_T|x_0) + \sum_{t=2}^T \log q_\sigma(x_{t-1}|x_t, x_0) - \sum_{t=1}^T \log p_\theta^{(t)}(x_{t-1}|x_t) - \log p_\theta(x_T) \right] \quad (145)$$

This would mean that for different choices of  $\sigma$ , we would have a different objective to optimise. [16] However, the authors [16] prove that  $J_\sigma$  is in fact equivalent to  $L_\gamma$  for certain choices of  $\gamma$ , where  $L_\gamma$  is defined as:

$$L_\gamma(\epsilon_\theta) = \sum_{t=1}^T \gamma_t \mathbb{E}_{x_0 \sim q(x_0), \epsilon_t \sim \mathcal{N}(0, I)} \left[ \|\epsilon_\theta^{(t)}(\sqrt{\alpha_t}x_0 + \sqrt{1 - \alpha_t}\epsilon_t) - \epsilon_t\|_2^2 \right] \quad (146)$$

where  $\gamma$  is a vector of length  $T$  with positive values that depend on  $\alpha$ .

We easily see that when  $\gamma = 1$ , we get  $L_1$  which is the training objective from the original DDPM paper. [3, 16]

Song et al. prove that

$$\forall \sigma > 0, \exists \gamma \in \mathbb{R}_+^T \text{ and } \exists C \in \mathbb{R} \text{ s.t. } J_\sigma = L_\gamma + C \quad (147)$$

This means that the optimal solution for  $J_\sigma$  is the same as the optimal solution for  $L_1$ , which means that the training objective for DDIMs can be kept the same as the one for DDPMs if the parameters being trained  $\theta$  are not the same for all timesteps. [16]

In order to speed up the sampling process, Song et al. [16] skip some timesteps in the reverse process. To do this, the forward process is redefined on a subset of timesteps of size  $S$ ,  $\{x_{\tau_1}, \dots, x_{\tau_S}\}$  and such that  $q(x_{\tau_i}|x_0) = \mathcal{N}(\sqrt{\alpha_{\tau_i}}x_0, (1 - \alpha_{\tau_i})I)$

Since this is a subset of the timesteps, the training can be done all timesteps and the sampling can be done on the subset. [16]

With the same amount of training timesteps  $T = 1000$  and the same training objective as the DDPM paper [3], Song et al. [16] were able to achieve solid FID scores on CIFAR-10 [7] and CelebA [9] datasets, with far fewer sampling timesteps, ranging from 10 to 100 timesteps. [16]

A new hyperparameter  $\eta \in \mathbb{R}_+$  is introduced to control the interpolation between the DDPM  $\sigma$  and the deterministic  $\sigma$  of the DDIM. [16]:

$$\sigma_{\tau_i}(\eta) = \eta \sqrt{\frac{1 - \alpha_{\tau_{i-1}}}{1 - \alpha_{\tau_i}}} \sqrt{1 - \frac{\alpha_{\tau_i}}{\alpha_{\tau_{i-1}}}} \quad (148)$$

The results obtained by Song et al. [16] show that the DDIM outperformed the DDPM for 10, 20, 50, 100 and 1000 timesteps and was only slightly worse at 1000 timesteps and  $\sigma_{\tau_i} = \hat{\sigma}_{\tau_i} = \sqrt{1 - \alpha_{\tau_i}/\alpha_{\tau_{i-1}}}$

$S$	CIFAR10 ( $32 \times 32$ )					CelebA ( $64 \times 64$ )					
	10	20	50	100	1000	10	20	50	100	1000	
$\eta$	0.0	<b>13.36</b>	<b>6.84</b>	<b>4.67</b>	<b>4.16</b>	4.04	<b>17.33</b>	<b>13.73</b>	<b>9.17</b>	<b>6.53</b>	3.51
	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
$\hat{\sigma}$	367.43	133.37	32.72	9.99	<b>3.17</b>	299.71	183.83	71.71	45.20	<b>3.26</b>	

Figure 5: Comparison of FID scores (lower is better) of DDPMs and DDIMs trained on  $S$  timesteps,  $S < T$  and  $\eta = 0$  is DDIM,  $\eta = 1$  and  $\hat{\sigma}$  are DDPMs [16]

### 3.3 Guidance

Recall that when sampling from a DDPM or a DDIM, we start with an image of Gaussian noise  $x_T$  and we denoise the image until we get some image  $x_0$  that resembles an image from the original dataset. We may however be interested in sampling something specific; for instance, from some text. In this case it is helpful to guide the denoising process towards whatever we specified.

#### 3.3.1 Classifier guidance

Classifier guidance was introduced by Dhariwal and Nichol in their *Diffusion Models Beat GANs on Image Synthesis* paper [2] in order to improve the samples generated.

To do this, they trained a classifier  $p_\Phi(y|x_t, t)$  on noisy images [2] since pre-trained classifiers were are generally trained on dataset images and not on noised ones.

The reverse process is then guided towards a class  $y$  using the gradient  $\nabla_{x_t} \log p_\Phi(y|x_t, t)$ . [2] The authors prove [2] that if we have a reverse process such as that from the DDPM  $p_\theta(x_t, x_{t+1})$ , to condition it on class  $y$ , we can sample from the following process instead:

$$p_{\theta, \Phi}(x_t | x_{t+1}, y) = Z p_\theta(x_t | x_{t+1}) p_\Phi(y | x_t) \quad (149)$$

where  $Z$  is a normalization constant.

Using the score-based formulation, we can easily find the new objective to train the model with classifier guidance. [10]

Remember that the score-based generative model learned to estimate  $\nabla_{x_t} \log q(x_t)$  ??.

If we want to condition the model on class  $y$ , we are interested in estimating  $\nabla_{x_t} \log q(x_t | y)$ . [10]

With Bayes' rule, we can rewrite this as [10]:

$$q(x_t | y) = \frac{q(y | x_t) q(x_t)}{q(y)} \quad (150)$$

$$\Leftrightarrow \log q(x_t | y) = \log \left[ \frac{q(y | x_t) q(x_t)}{q(y)} \right] \quad (151)$$

$$\Leftrightarrow \nabla_{x_t} \log q(x_t | y) = \nabla_{x_t} \log q(x_t) + \nabla_{x_t} \log q(y | x_t) - \nabla_{x_t} \log q(y) \quad (152)$$

$$= \nabla_{x_t} \log q(x_t) + \nabla_{x_t} \log q(y | x_t) \quad (153)$$

We can see that  $\nabla_{x_t} \log q(x_t)$  is the original unconditional score from the score-based generative model and  $\nabla_{x_t} \log q(y | x_t)$  is the gradient of the log-likelihood of the class  $y$  given the image  $x_t$ . [2, 10].

To control the strength of the guidance, the authors introduce a hyperparameter  $\gamma$  that multiplies the

classifier gradient [2, 10]:

$$\nabla_{x_t} \log q(x_t | y) = \nabla_{x_t} \log q(x_t) + \gamma \nabla_{x_t} \log q(y | x_t) \quad (154)$$

Trivially, when  $\gamma = 0$ , the model doesn't use the classifier and we get the original score-based generative model.

Dhariwal and Nichol [2] gave two algorithms to train the model with classifier guidance: one for DDPMs and one that also works for DDIMs that uses the score-based generative model formulation:

---

**Algorithm 4** Training a DDPM/DDIM with classifier guidance

---

**Require:** *Model*  $\epsilon_\theta(x_t, t)$ , *Classifier*  $p_\Phi(y | x_t)$  with  $y$  the class label

```

1: Sample  $x_T \sim \mathcal{N}(0, I)$ 
2: for  $t = T, T-1, \dots, 1$  do
3:    $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t, t) - \sqrt{1 - \bar{\alpha}_t} \nabla_{x_t} p_\Phi(y | x_t)$ 
4:    $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left( \frac{x_t - \hat{\epsilon} \sqrt{1 - \bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}} \right) + \hat{\epsilon} \sqrt{1 - \bar{\alpha}_{t-1}}$ 
5: end for
6: return  $x_0$ 
```

---

### 3.3.2 Classifier-free guidance

Despite the impressive performance of the *ablated diffusion model* that used classifier guidance, Ho and Salimans made an improvement by using classifier-free guidance. [4]

Classifier guidance requires a classifier to be trained on noisy images. This means that not only does it require training a classifier in addition to the generative model, but it also requires the classifier to be trained on noisy images for which pre-trained classifiers do not generally exist [4].

Therefore, Ho and Salimans [4] introduced a classifier-free guidance method which does not require a trained classifier at all.

The authors use an unconditional diffusion model  $q(x_t)$  and a conditional diffusion model  $q(x_t | y)$  where  $y$  is the class label. [4, 10].

Starting from equation ??:

$$\nabla_{x_t} \log q(x_t | y) = \nabla_{x_t} \log q(x_t) + \gamma \nabla_{x_t} \log q(y | x_t) \quad (155)$$

$$= \nabla_{x_t} \log q(x_t) + \gamma (\nabla_{x_t} \log q(x_t | y) - \nabla_{x_t} \log q(x_t)) \quad (156)$$

$$= \nabla_{x_t} \log q(x_t) + \gamma \nabla_{x_t} \log q(x_t | y) - \gamma \nabla_{x_t} \log q(x_t) \quad (157)$$

$$= \gamma \nabla_{x_t} \log q(x_t | y) + \nabla_{x_t} \log q(x_t) - \gamma \nabla_{x_t} \log q(x_t) \quad (158)$$

$$= \gamma \nabla_{x_t} \log q(x_t | y) + (1 - \gamma) \nabla_{x_t} \log q(x_t) \quad (159)$$

$\gamma \nabla_{x_t} \log q(x_t | y)$  is the score of the conditional model and  $(1 - \gamma) \nabla_{x_t} \log q(x_t)$  is the score of the unconditional model [10].

However it isn't necessary to train a conditional model and an unconditional model as we can just train the conditional model  $q(x_t | y)$  and simply ignore the conditioning information  $y$  (by setting it to null, for example) to get the unconditional model  $q(x_t)$ . [4, 10]

---

## 4 Comparison with other diffusion models

---

## 5 Implementation

---

## 6 Conclusion

---

## A Convolution of two Gaussian distributions

Equation ?? had given us:

$$x_t = \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \mathcal{N}(0, \alpha_t (1 - \alpha_{t-1}) I) + \mathcal{N}(0, (1 - \alpha_t) I)$$

We can find the distribution of the sum of two independant 1D random variables  $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$  and  $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$  as the convolution of two Gaussian distributions.

Let  $Z = X + Y$  with  $X$  and  $Y$  independent.

$$\begin{aligned} f_Z(z) &= \int_{-\infty}^{\infty} f_X(x) f_Y(z-x) dx \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma_X^2}} \exp\left[-\frac{(x-\mu_X)^2}{2\sigma_X^2}\right] \frac{1}{\sqrt{2\pi\sigma_Y^2}} \exp\left[-\frac{(z-x-\mu_Y)^2}{2\sigma_Y^2}\right] dx \\ &= \frac{1}{2\pi\sigma_X\sigma_Y} \int_{-\infty}^{\infty} \exp\left[-\frac{(x-\mu_X)^2}{2\sigma_X^2}\right] \exp\left[-\frac{(z-x-\mu_Y)^2}{2\sigma_Y^2}\right] dx \\ &= \frac{1}{2\pi\sigma_X\sigma_Y} \int_{-\infty}^{\infty} \exp\left[-\frac{(x-\mu_X)^2}{2\sigma_X^2} - \frac{(z-x-\mu_Y)^2}{2\sigma_Y^2}\right] dx \\ &= \frac{1}{2\pi\sigma_X\sigma_Y} \int_{-\infty}^{\infty} \exp\left[-\frac{x^2 - 2\mu_X x + \mu_X^2}{2\sigma_X^2} - \frac{z^2 + x^2 - 2xz - 2z\mu_Y + 2\mu_Y x + \mu_Y^2}{2\sigma_Y^2}\right] dx \\ &= \frac{1}{2\pi\sigma_X\sigma_Y} \int_{-\infty}^{\infty} \exp\left[-\frac{\sigma_Y^2(x^2 - 2\mu_X x + \mu_X^2)}{2\sigma_X^2\sigma_Y^2} - \frac{\sigma_X^2(z^2 + x^2 - 2xz - 2z\mu_Y + 2\mu_Y x + \mu_Y^2)}{2\sigma_X^2\sigma_Y^2}\right] dx \\ &= \frac{1}{2\pi\sigma_X\sigma_Y} \int_{-\infty}^{\infty} \exp\left[-\frac{\sigma_Y^2 x^2 - 2\mu_X \sigma_Y^2 x + \mu_X^2 \sigma_Y^2 + \sigma_X^2 z^2 + \sigma_X^2 x^2 - 2xz\sigma_X^2 + 2z\mu_Y \sigma_X^2 - 2\mu_Y x \sigma_X^2 + \sigma_X^2 \mu_Y^2}{2\sigma_X^2\sigma_Y^2}\right] dx \\ &= \frac{1}{2\pi\sigma_X\sigma_Y} \int_{-\infty}^{\infty} \exp\left[-\frac{(\sigma_X^2 + \sigma_Y^2) x^2 - 2x(\mu_X \sigma_Y^2 + z\sigma_X^2 - \mu_Y \sigma_X^2) + \sigma_X^2(z - \mu_Y)^2 + \mu_X^2 \sigma_Y^2}{2\sigma_X^2\sigma_Y^2}\right] dx \end{aligned}$$

Let  $\sigma_Z = \sqrt{\sigma_X^2 + \sigma_Y^2}$ .

$$\begin{aligned} &= \frac{1}{\sqrt{2\pi} \frac{\sigma_X \sigma_Y}{\sigma_Z}} \frac{1}{\sqrt{2\pi} \sigma_Z} \int_{-\infty}^{\infty} \exp\left[-\frac{x^2 - 2x \frac{\sigma_X^2(z-\mu_Y) + \sigma_Y^2 \mu_X}{\sigma_Z^2} + \frac{\sigma_X^2(z-\mu_Y)^2 + \mu_X^2 \sigma_Y^2}{\sigma_Z^2}}{2 \left(\frac{\sigma_X \sigma_Y}{\sigma_Z}\right)^2}\right] dx \\ &= \frac{1}{\sqrt{2\pi} \frac{\sigma_X \sigma_Y}{\sigma_Z}} \frac{1}{\sqrt{2\pi} \sigma_Z} \int_{-\infty}^{\infty} \exp\left[-\frac{\left(x - \frac{\sigma_X^2(z-\mu_Y) + \sigma_Y^2 \mu_X}{\sigma_Z^2}\right)^2 - \left(\frac{\sigma_X^2(z-\mu_Y) + \sigma_Y^2 \mu_X}{\sigma_Z^2}\right)^2 + \frac{\sigma_X^2(z-\mu_Y)^2 + \sigma_Y^2 \mu_X^2}{\sigma_Z^2}}{2 \left(\frac{\sigma_X \sigma_Y}{\sigma_Z}\right)^2}\right] dx \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi} \frac{\sigma_X \sigma_Y}{\sigma_Z}} \exp\left[-\frac{\left(x - \frac{\sigma_X^2(z-\mu_Y) + \sigma_Y^2 \mu_X}{\sigma_Z^2}\right)^2}{2 \left(\frac{\sigma_X \sigma_Y}{\sigma_Z}\right)^2}\right] \end{aligned}$$


---

---


$$\begin{aligned}
& \frac{1}{\sqrt{2\pi}\sigma_Z} \exp \left[ -\frac{\sigma_Z^2 (\sigma_X^2 (z - \mu_Y)^2 + \sigma_Y^2 \mu_X^2) - (\sigma_X^2 (z - \mu_Y) + \sigma_Y^2 \mu_X)^2}{2\sigma_Z^2 (\sigma_X \sigma_Y)^2} \right] dx \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi} \frac{\sigma_X \sigma_Y}{\sigma_Z}} \exp \left[ -\frac{\left( x - \frac{\sigma_X^2 (z - \mu_Y) + \sigma_Y^2 \mu_X}{\sigma_Z^2} \right)^2}{2 \left( \frac{\sigma_X \sigma_Y}{\sigma_Z} \right)^2} \right] \frac{1}{\sqrt{2\pi}\sigma_Z} \exp \left[ -\frac{(z - (\mu_X + \mu_Y))^2}{2\sigma_Z^2} \right] dx \\
&= \frac{1}{\sqrt{2\pi}\sigma_Z} \exp \left[ -\frac{(z - (\mu_X + \mu_Y))^2}{2\sigma_Z^2} \right] \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi} \frac{\sigma_X \sigma_Y}{\sigma_Z}} \exp \left[ -\frac{\left( x - \frac{\sigma_X^2 (z - \mu_Y) + \sigma_Y^2 \mu_X}{\sigma_Z^2} \right)^2}{2 \left( \frac{\sigma_X \sigma_Y}{\sigma_Z} \right)^2} \right] dx \\
&= \frac{1}{\sqrt{2\pi}\sigma_Z} \exp \left[ -\frac{(z - (\mu_X + \mu_Y))^2}{2\sigma_Z^2} \right] \\
&= \mathcal{N}(z; \mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)
\end{aligned}$$

Given this, we can refer back to our case and therefore we find:

$$\begin{aligned}
x_t &= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \mathcal{N}(0, \alpha_t (1 - \alpha_{t-1}) I) + \mathcal{N}(0, (1 - \alpha_t) I) \\
&= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \mathcal{N}(0, \alpha_t ((1 - \alpha_{t-1}) + 1 - \alpha_t) I)
\end{aligned}$$



---

## 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] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: [2207.12598 \[cs.LG\]](#).
- [5] Aapo Hyvarinen. *Estimation of Non-Normalized Statistical Models by Score Matching*. 2005. URL: <https://jmlr.org/papers/volume6/hyvarinen05a/hyvarinen05a.pdf>.
- [6] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: [1312.6114 \[stat.ML\]](#).
- [7] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. “CIFAR-10 (Canadian Institute for Advanced Research)”. In: (2024). URL: <http://www.cs.toronto.edu/~kriz/cifar.html>.
- [8] Zhitong Lai et al. *Rethinking Skip Connections in Encoder-decoder Networks for Monocular Depth Estimation*. 2022. arXiv: [2208.13441 \[cs.CV\]](#).
- [9] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.
- [10] Calvin Luo. *Understanding Diffusion Models: A Unified Perspective*. 2022. arXiv: [2208.11970 \[cs.LG\]](#).
- [11] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: [2102.09672 \[cs.LG\]](#).
- [12] Aaron van den Oord et al. *Conditional Image Generation with PixelCNN Decoders*. 2016. arXiv: [1606.05328 \[cs.CV\]](#).
- [13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: [1505.04597 \[cs.CV\]](#).
- [14] Tim Salimans and Diederik P. Kingma. *Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks*. 2016. arXiv: [1602.07868 \[cs.LG\]](#).
- [15] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: [1503.03585 \[cs.LG\]](#).
- [16] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022. arXiv: [2010.02502 \[cs.LG\]](#).
- [17] Yang Song. *Blog post: Generative Modeling by Estimating Gradients of the Data Distribution*. 2021. URL: <https://yang-song.net/blog/2021/score/>.
- [18] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. 2020. arXiv: [1907.05600 \[cs.LG\]](#).
- [19] Yang Song et al. *Sliced Score Matching: A Scalable Approach to Density and Score Estimation*. 2019. arXiv: [1905.07088 \[cs.LG\]](#).
- [20] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: [1706.03762 \[cs.CL\]](#).
- [21] Pascal Vincent. *A Connection Between Score Matching and Denoising Autoencoders*. 2010. URL: [https://www.iro.umontreal.ca/~vincentp/Publications/smdae\\_techreport.pdf](https://www.iro.umontreal.ca/~vincentp/Publications/smdae_techreport.pdf).
- [22] Max Welling and Yee Whye Teh. *Bayesian Learning via Stochastic Gradient Langevin Dynamics*. 2011. URL: <https://www.stats.ox.ac.uk/~teh/research/compstats/WelTeh2011a.pdf>.

- 
- [23] Lilian Weng. “What are diffusion models?” In: *lilianweng.github.io* (July 2021). URL: <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>.
  - [24] Yuxin Wu and Kaiming He. *Group Normalization*. 2018. arXiv: [1803.08494 \[cs.CV\]](#).
  - [25] Sergey Zagoruyko and Nikos Komodakis. *Wide Residual Networks*. 2017. arXiv: [1605.07146 \[cs.CV\]](#).