Denoising Diffusion Probabilistic Models

Giovani Tavares

Instituto de Matemática e Estatística
(IME-USP)

11 / 2025

Estrutura da apresentação

Overview

- Denoising Diffusion Probabilistic Models (DDPMs) are models capable of predicting *noise* from a noisy input.

Overview

- Denoising Diffusion Probabilistic Models (DDPMs) are models capable of predicting *noise* from a noisy input.
- By using such prediction, a sampling algorithm can be used to remove the noise from the input which results in a denoised output.

DDPMs Building Blocks

DDPMs are made of two processes: a **forward** and a **reversion** process.

1. **Forward Process**: gradually adds noise to a image by sampling from a normal distribution according to a Markov Chain
2. **Reverse Process**: removes added noise by sampling from another normal distribution according to another Markov Chain

The core idea of the training of DDPMs involve learning the reversion process' distribution's parameters.

DDPMs Building Blocks

DDPMs are made of two processes: a **forward** and a **reversion** process.

1. **Forward Process' Distribution**

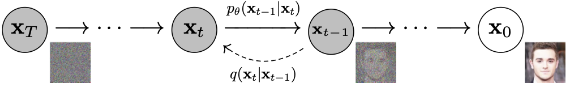
$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t} \times x_{t-1}; \beta_t I) \tag{1}$$

2. **Reverse Process' Distribution**

$$p(x_T) = \mathcal{N}(x_T; 0; 1) \tag{2}$$

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{3}$$

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t); \Sigma_\theta(x_t, t)) \tag{4}$$



Forward Process’ Transitions Distribution

- $$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t} \times x_{t-1}; \beta_t I) \tag{5}$$
- **According to the original DDPM paper, the Variance Schedule $\beta_1, ..., \beta_T$ sequence that defines the noisy images distribution are held constant.**

Forward Process' Transitions Distribution

- $$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t} \times x_{t-1}; \beta_t I) \tag{6}$$
- According to the original DDPM paper, the **Variance Schedule** $\beta_1, ..., \beta_T$ sequence that defines the noisy images distribution are held constant.
- **Ideally, one would need a single transition from x_0 to get to x_t, with $t > 0$.**

Forward Process' Transitions Distribution

- $$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t} \times x_{t-1}; \beta_t I) \tag{7}$$
- According to the original DDPM paper, the **Variance Schedule** $\beta_1, ..., \beta_T$ sequence that defines the noisy images distribution are held constant.
- Ideally, one would need a single transition from x_0 to get to x_t, with $t > 0$.
- **The authors of the paper achieve such ideal scenario by defining a cumulative noise α_t presented in the following slide.**

Cumulative Noise

The Cumulative Noise (α_t) added to an input x_0 up to the t-th step is defined as:

$$\alpha_t := 1 - \beta_t \tag{8}$$

$$\bar{\alpha}_t := \prod_{s=1}^{t} a_s \tag{9}$$

which leads to (10)

$$q(x_t|x_0) := \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} \times x_0; (1 - \bar{\alpha}_t)I) \tag{11}$$

Variance Scheduler Pytorch Implementation

```python
class VarianceScheduler(nn.Module):

    def __init__(self, T: int=1000):
        super().__init__()
        self.beta = torch.linspace(1e-4, 0.02, T, requires_grad=False)
        alpha = 1 - self.beta
        self.alpha = torch.cumprod(alpha, dim=0).requires_grad_(False)

    def forward(self, t):
        return self.beta[t], self.alpha[t]

```

Reverse Process' Transitions Distribution

- $$p(x_T) = \mathcal{N}(x_T; 0; 1) \tag{12}$$
$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{13}$$
$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t); \Sigma_\theta(x_t, t)) \tag{14}$$

Reverse Process' Transitions Distribution

-

$$p(x_T) = \mathcal{N}(x_T; 0; 1) \tag{15}$$

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{16}$$

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t); \Sigma_\theta(x_t, t)) \tag{17}$$

- **The core idea of the chain is that the transitions remove each a little bit of the noise from the initial state up until the noise-free state $\mathbf{x_0}$**

Reverse Process' Transitions Distribution

- $$p(x_T) = \mathcal{N}(x_T; 0; 1) \tag{18}$$
  $$p_\theta(x_{0:T}) := p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{19}$$
  $$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t); \Sigma_\theta(x_t, t)) \tag{20}$$
- The core idea of the chain is that the transitions remove each a little bit of the noise from the initial state up until the noise-free state $\mathbf{x_0}$
- **What we ultimately want is to have a $\mathbf{x}_0$ that is as likely as possible. We can use the standard rule of probability to obtain a marginalization of $p(\mathbf{x}_0)$ using the latent variables $\mathbf{x_{1:T}}$**

Reverse Process' Prior

- **Ideally, the Reverse Process would let us sample from:**

$$p(\mathbf{x}_0) = \int p(\mathbf{x_0}, \mathbf{x_{1:T}})\mathbf{d_{1:T}} \tag{21}$$

$$\tag{22}$$

Reverse Process' Prior

- Ideally, the Reverse Process would let us sample from:

$$p(\mathbf{x}_0) = \int p(\mathbf{x_0}, \mathbf{x_{1:T}})\mathbf{d_{1:T}} \tag{23}$$

(24)

- **From the defition above, we see that $p(\mathbf{x}_0)$ is very complex due to its multidimensionality, which makes it intractable. That is why in DDPMs, $p(\mathbf{x}_0)$ is never computed directly, but instead its lower bound.**

Evidence Lower Bound (ELBO)

- **DDPMs are not trained to sample from $p(\mathbf{x}_0)$, but instead to maximize its lower bound**

Evidence Lower Bound (ELBO)

- DDPMs are not trained to sample from $p(\mathbf{x}_0)$, but instead to maximize its lower bound ELBO (L)
-

$$log[p_\theta(\mathbf{x_0})] = log \int_{\mathbf{x_{1:T}}} p(\mathbf{x_0}, \mathbf{x_{1:T}}) d\mathbf{x_{1:T}} \tag{25}$$

$$log[p_\theta(\mathbf{x_0})] = log \int_{\mathbf{x_{1:T}}} p(\mathbf{x_0}, \mathbf{x_{1:T}}) \frac{q(\mathbf{x_{1:T}}|\mathbf{x_0})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})} d\mathbf{x_{1:T}} \tag{26}$$

$$log[p_\theta(\mathbf{x_0})] = log(\mathbb{E}_q\left[\frac{p(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})}\right]) \tag{27}$$

$$\text{the Jensen's inequality tells us} \tag{28}$$

$$f(\mathbb{E}(\mathbf{X})) \geq \mathbb{E}(f(\mathbf{X})) \tag{29}$$

$$\text{for any concave function f.} \tag{30}$$

$$\text{log is concave, hence:} \tag{31}$$

$$log[p_\theta(\mathbf{x_0})] \geq \mathbb{E}_q\left[log \frac{p(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})}\right] \tag{32}$$

$$\text{If we define the Evidence Lower Bound L as:} \tag{33}$$

$$L := \mathbb{E}_q\left[-log \frac{p(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})}\right] \tag{34}$$

$$\implies -log[p_\theta(\mathbf{x_0})] \leq L \tag{35}$$

Evidence Lower Bound (ELBO)

- DDPMs are not trained to sample from $p(\mathbf{x}_0)$, but instead to maximize its lower bound ELBO (L)
- **With more algebraic manipulation and using the fact that both the forward and reverse processes are Markov Chains, one can derive the following equation:**

$$L := \mathbb{E}_q \left[- \log \frac{p(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}} | \mathbf{x_0})} \right] \tag{36}$$

$$L = \mathbb{E}_q \left[- \log \frac{p(\mathbf{x_T})}{q(\mathbf{x_T} | \mathbf{x_0})} \right] - \log p_\theta(\mathbf{x_0} | \mathbf{x_1}) - \mathbb{E}_q \left[\sum_{t=2}^{T} \mathbf{D_{KL}} \left[q(\mathbf{x_{t-1}} | \mathbf{x_t}, \mathbf{x_0}) || p_\theta(\mathbf{x_{t-1}} | \mathbf{x_t})) \right] \right] \tag{37}$$

Evidence Lower Bound (ELBO)

- DDPMs are not trained to sample from $p(\mathbf{x}_0)$, but instead to maximize its lower bound ELBO (L)
- With more algebraic manipulation and using the fact that both the forward and reverse processes are Markov Chains, one can derive the following equation:

$$L := \mathbb{E}_q\left[-\log\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \tag{38}$$

$$L = \mathbb{E}_q\left[-\log\frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)}\right] - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) - \mathbb{E}_q\left[\sum_{t=2}^{T}\mathbf{D}_{\mathbf{KL}}\left[q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\right]\right] \tag{39}$$

- **We can conclude that maximizing ELBO (L) is equivalent to minimizing the KL-Divergence between $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, i.e., minimizing the divergence between the forward and reverse processes' distributions.**

Computing the KL Divergence Between q and p_θ

$$q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0}) = \frac{q(\mathbf{x_t}|\mathbf{x_{t-1}}, \mathbf{x_0})q(\mathbf{x_{t-1}}|\mathbf{x_0})}{q(\mathbf{x_t}|\mathbf{x_0})} \tag{40}$$

$$\text{we know the q distribution from Definition, hence} \tag{41}$$

$$q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0}) \text{ is a product of known Gaussians over another known Gaussian} \tag{42}$$

$$\mu_q(\mathbf{x_t}, \mathbf{x_0}) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x_0}}{(1-\bar{\alpha}_t)} \tag{43}$$

$$\Sigma_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I} \tag{44}$$

$$\implies q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0}) = \mathcal{N}(\mathbf{x_{t-1}}; \mu_q(\mathbf{x_t}, \mathbf{x_0}); \Sigma_q(t)) \tag{45}$$

Computing the KL Divergence Between q and p_θ

$$p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}) = \mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_\theta(t)) \tag{46}$$

$$\text{from the definition of the reverse process:} \tag{47}$$

$$\Sigma_\theta(t) = \Sigma_q(t) \tag{48}$$

$$\text{we are only left with the distribution's mean } \mu_\theta \tag{49}$$

$$\implies p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}) = \mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t)) \tag{50}$$

Computing the KL Divergence Between q and p_θ

- **Now we know we are trying to compute the KL Divergence between two Gaussians with the exact same variance.**

Computing the KL Divergence Between q and p_θ

- Now we know we are trying to compute the KL Divergence between two Gaussians with the exact same variance.
- **For that, there is the following result that arives from the definition of such divergence**

$$d_1(x) = \mathcal{N}(\mu_1, \sigma^2) \tag{51}$$

$$d_2(x) = \mathcal{N}(\mu_2, \sigma^2) \tag{52}$$

The KL divergence $D_{KL}(d_1|d_2)$ is given by: (53)

$$D_{KL}(d_1|d_2) = \frac{(\mu_1 - \mu_2)^2}{2\sigma^2} \tag{54}$$

Hence, (55)

$$\mathbf{D_{KL}}\left[q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0})||p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t})\right] = \mathbf{D_{KL}}\left(\mathcal{N}(\mathbf{x_{t-1}}; \mu_q(\mathbf{x_t}, \mathbf{x_0}); \Sigma_q(t)), \mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t))\right) \tag{56}$$

$$= \frac{1 - \bar{\alpha}_t}{2(1-\alpha_t)(1-\bar{\alpha}_{t-1})}||(\mu_q - \mu_\theta)_2^2|| \tag{57}$$

Computing the KL Divergence Between q and p_θ

- Now we know we are trying to compute the KL Divergence between two Gaussians with the exact same variance.
- For that, there is the following result that arives from the definition of such divergence

$$d_1(x) = \mathcal{N}(\mu_1, \sigma^2) \tag{58}$$

$$d_2(x) = \mathcal{N}(\mu_2, \sigma^2) \tag{59}$$

$$\text{The KL divergence } D_{KL}(d_1 | d_2) \text{ is given by:} \tag{60}$$

$$D_{KL}(d_1 | d_2) = \frac{(\mu_1 - \mu_2)^2}{2\sigma^2} \tag{61}$$

$$\text{Hence,} \tag{62}$$

$$\mathbf{D_{KL}}\left[q(\mathbf{x_{t-1}} | \mathbf{x_t}, \mathbf{x_0}) || p_\theta(\mathbf{x_{t-1}} | \mathbf{x_t})\right] = \mathbf{D_{KL}}\left(\mathcal{N}(\mathbf{x_{t-1}}; \mu_q(\mathbf{x_t}, \mathbf{x_0}); \Sigma_q(t)), \mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t))\right) \tag{63}$$

$$= \frac{1 - \bar{\alpha}_t}{2(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})} ||(\mu_q - \mu_\theta)_2^2|| \tag{64}$$

- **We just need to minimize the difference between the means of the reverse and forward processes' distributions, i.,e., minimize $||(\mu_q - \mu_\theta)_2^2||$.**

Defining The Model's Prediction

- **We can use the prediction of our model as the forward process' mean**

Defining The Model's Prediction

- We can use the prediction of our model as the forward process' mean
- $$\mu_q(\mathbf{x_t}, \mathbf{x_0}) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x_0}}{(1-\bar{\alpha}_t)} \tag{65}$$
$$\text{we can define the prediction} \tag{66}$$
$$\mu_\theta(\mathbf{x_t}) := \hat{\mu}_q(\mathbf{x_t}, \mathbf{x_0}) \tag{67}$$
$$= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_\theta}{(1-\bar{\alpha}_t)} \tag{68}$$

Defining The Model's Prediction

- We can use the prediction of our model as the forward process' mean
-

$$\mu_q(\mathbf{x_t}, \mathbf{x_0}) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x_0}}{(1-\bar{\alpha}_t)} \tag{69}$$

$$\text{we can define the prediction} \tag{70}$$

$$\mu_\theta(\mathbf{x_t}) := \hat{\mu}_q(\mathbf{x_t}, \mathbf{x_0}) \tag{71}$$

$$= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_\theta}{(1-\bar{\alpha}_t)} \tag{72}$$

$$\implies \mathbf{D_{KL}}\left(\mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t)), \mathcal{N}(\mathbf{x_{t-1}}; \mu_q(\mathbf{x_t}, \mathbf{x_0}); \Sigma_q(t))\right) \tag{73}$$

$$= \frac{(1-\bar{\alpha}_t)(\bar{\alpha}_{t-1})}{2(1-\alpha_t)(1-\bar{\alpha}_{t-1})}||(\mathbf{x}_\theta - \mathbf{x_0})_2^2|| \tag{74}$$

Defining The Model's Prediction

- We can use the prediction of our model as the forward process' mean
-

$$\mu_q(\mathbf{x_t}, \mathbf{x_0}) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x_0}}{(1-\bar{\alpha}_t)} \tag{75}$$

$$\text{we can define the prediction} \tag{76}$$

$$\mu_\theta(\mathbf{x_t}) := \hat{\mu}_q(\mathbf{x_t}, \mathbf{x_0}) \tag{77}$$

$$= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_\theta}{(1-\bar{\alpha}_t)} \tag{78}$$

$$\implies \mathbf{D_{KL}}\left(\mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t)), \mathcal{N}(\mathbf{x_{t-1}}; \mu_q(\mathbf{x_t}, \mathbf{x_0}); \Sigma_q(t))\right) \tag{79}$$

$$= \frac{(1-\bar{\alpha}_t)(\bar{\alpha}_{t-1})}{2(1-\alpha_t)(1-\bar{\alpha}_{t-1})}||(\mathbf{x}_\theta - \mathbf{x_0})_2^2|| \tag{80}$$

- **Theoretically, this equation could be used as the loss function directly, but we can rewrite the images $\mathbf{x}_\theta$ and $\mathbf{x_0}$ in function of the Gaussian noises.**

Defining The Model's Prediction

-
$$q(x_t|x_0) := \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} \times x_0; (1-\bar{\alpha}_t)\mathbb{I}) \tag{81}$$
$$\text{which let's us write} \tag{82}$$
$$\mathbf{x_t} = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{83}$$
$$\implies \mathbf{x_0} = \frac{\mathbf{x_t} - \sqrt{1-\bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}} \tag{84}$$
$$\text{for a Standard Gaussian Noise } \epsilon. \tag{85}$$
$$\text{We can now define our prediction } \hat{\epsilon} = \epsilon_\theta \tag{86}$$
$$\mathbf{x}_\theta = \frac{\mathbf{x_t} - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta}{\sqrt{\bar{\alpha}_t}} \tag{87}$$
$$\implies \frac{(1-\bar{\alpha}_t)(\bar{\alpha}_{t-1})}{2(1-\alpha_t)(1-\bar{\alpha}_{t-1})}||(\mathbf{x}_\theta - \mathbf{x_0})_2^2|| = \frac{(1-\bar{\alpha}_t)(\bar{\alpha}_{t-1})}{2(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t}||(\epsilon_\theta - \epsilon)_2^2|| \tag{88}$$
- **The DDPM paper authors mention that optimizing $||(\epsilon_\theta - \epsilon)_2^2||$ without the scaling factor with the cumulative noise α_t is enough.**

Python

```python
    def print()
```

C

```c
#include <stdio.h>

int main() {
    int numero = 5;
    int dobro = 2 * numero;

    printf("O dobro de %d eh %d\n", numero, dobro);
    return 0;
}
```

C++

```cpp
#include <iostream>
using namespace std;

int main() {
    int numero = 5;
    int dobro = 2 * numero;

    cout << "O dobro de " << numero;
    cout << " eh " << dobro << endl;
    return 0;
}
```

R

```r
# Função para calcular o dobro
calcular_dobro <- function(x) {
  return(2 * x)
}

# Testando a função
numero <- 5
resultado <- calcular_dobro(numero)
print(paste("O dobro de", numero, "é", resultado))
```

Java

```java
public class Exemplo {
    public static void main(String[] args) {
        int numero = 5;
        int dobro = 2 * numero;

        System.out.println("O dobro de " + numero +
                          " eh " + dobro);
    }
}
```

Noise Predictor Training

1: **repeat**
2: $\mathbf{x_0} \sim \mathbf{q}(\mathbf{x_0})$ ▷ Sample image from training set
3: $\mathbf{x_0} \sim \mathbf{Uniform}(\{\mathbf{1}, \dots, \mathbf{T}\})$ ▷ Sample the step of the Forward Process Markov Chain
4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Sample standard gaussian noise to be added to the input
5: $\mathbf{x_t} = \sqrt{\bar{\alpha}_\mathbf{t}}\mathbf{x_0} + \sqrt{\mathbf{1} - \bar{\alpha}_\mathbf{t}}\epsilon$ ▷ Forward Process/ Generating Noisy Image
6: Take Gradient Descent Step on $\nabla_\theta(||\epsilon - \epsilon_\theta(\mathbf{x_t}, \mathbf{t})||)$
7: **until** converged

Sampling Algorithm Derivation

-

$$p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}) = \mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t)) \tag{89}$$

$$\mu_\theta(\mathbf{x_t}) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_\theta}{(1-\bar{\alpha}_t)} \tag{90}$$

$$\mathbf{x}_\theta = \frac{\mathbf{x_t} - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta}{\sqrt{\bar{\alpha}_t}} \tag{91}$$

$$\implies \mu_\theta(\mathbf{x_t}) = \frac{\mathbf{x_t}}{\sqrt{\alpha_t}} - \frac{(1-\alpha_t)(\sqrt{1-\bar{\alpha}_t})}{(1-\bar{\alpha}_t)(\sqrt{\alpha_t})}\epsilon_\theta = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x_t} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta\right) \tag{92}$$

$$\Sigma_\theta(t) = \Sigma_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I} \tag{93}$$

We now have defined $\mathbf{x_{t-1}}$'s mean and variance given $\mathbf{x_t}$ which let's us write it as (94)

$$\mathbf{x_{t-1}} = \mu_\theta(\mathbf{x_t}) + \sqrt{\Sigma_\theta(t)}\mathbf{z} \tag{95}$$

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{96}$$

Sampling Algorithm

repeat
  $\mathbf{x_T} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Sample random noisy image
  $\mathbf{T} \sim \mathbf{Uniform}(\{\mathbf{1}, \ldots, \mathbf{1000}\})$ ▷ Sample random length of the Denoising Chain
  for $\mathbf{t} = \mathbf{T}, \ldots, \mathbf{1}$ **do**
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$ else $\mathbf{z} = \mathbf{0}$
    $\mathbf{x_{t-1}} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x_t} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x_t}, \mathbf{t})\right) + \sqrt{\Sigma_\theta(t)}\mathbf{z}$ ▷ Sampling $\mathbf{x_{t-1}}$ from $p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t})$
  end for
return $\mathbf{x_0}$

Fim da apresentação!