Denoising Diffusion Probabilistic Models

Giovani Tavares de Andrade

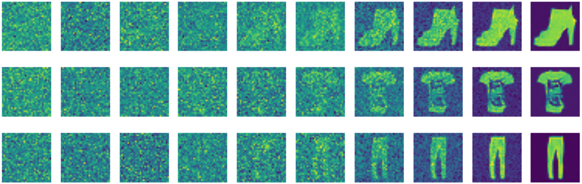
Instituto de Matemática e Estatística
(IME-USP)

11 / 2025

Outline

Overview

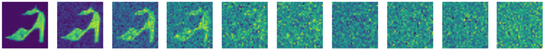
- Denoising Diffusion Probabilistic Models (DDPMs) generate images by iteratively removing noise from a signal until it looks like a real image.



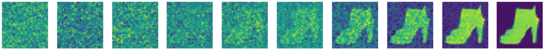
DDPMs Building Blocks

DDPMs are made of **two iterative processes** defined by different Markov Chains: 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 noise by sampling from a normal distribution according to another Markov Chain



DDPMs Building Blocks

1. **Forward Process' Distribution**

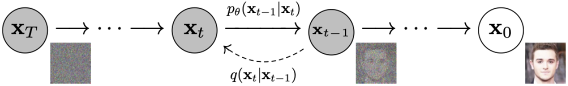
$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t} \times x_{t-1}; \beta_t I) \tag{1}$$

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' Distribution

In order to produce signals that are pure Gaussian Noise, we need to sample from the following Normal Distribution which we will simply call forward (**q**):

$$\mathbf{x_t} \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\,\mathbf{x_0}, (\mathbf{1} - \bar{\alpha}_t)\mathbf{I}) \tag{5}$$

$$\mathbf{x_t} \sim \mathbf{q}(\mathbf{x_t}|\mathbf{x_0}) \tag{6}$$

- The forward process produces a noisy image $\mathbf{x_t}$ by sampling from a normal distribution from which the mean is a function of the noisy-free image $\mathbf{x_0}$

$\sim \mathcal{N}(\sqrt{\bar{\alpha}_t}$


$, (\mathbf{1} - \bar{\alpha}_t)\mathbf{I})$

Forward **q**'s Parameters

```python
class DDPMScheduler(nn.Module):
    def __init__(self, num_time_steps: int=1000):

        super().__init__()
        self.beta = torch.linspace(1e-4, 0.02, num_time_steps, requires_grad=False)

        # Alpha is defined as 1 - beta, i.e., 1 - Variance at step t
        alpha = 1 - self.beta

        # The variances are held constant during training, requires_grad = False
        self.alpha = torch.cumprod(alpha, dim=0).requires_grad_(False)

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

Given an input tensor $\mathbf{x_0}$ one can sample from $\mathbf{q}(\mathbf{x_t}|\mathbf{x_0})$ by making the attribution:

```python
scheduler = DDPMScheduler()
a = scheduler.alpha[t].view(batch_size,1,1,1)
#Adds the gaussian noise to the input according to the variance schedule,
# which is the actual forward pass
x = (torch.sqrt(a)*x) + (torch.sqrt(1-a)*e)
```

Reverse Process' Transitions Distribution

- $$p(x_T) = \mathcal{N}(x_T; 0; 1) \tag{7}$$
  $$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{8}$$
  $$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t); \Sigma_\theta(x_t, t)) \tag{9}$$
- What we ultimately want is to have a $\mathbf{x}_0$ that is as likely as possible. We can marginalize $\mathbf{x}_0$ using the latent variables $\mathbf{x_{1:T}}$, i.e., by using the noisy images.

Reverse Process' Prior

- Ideally, the Reverse Process would enable us to sample from:

$$p(\mathbf{x}_0) = \int p(\mathbf{x_0}, \mathbf{x_{1:T}})\mathbf{d_{1:T}} \tag{10}$$

$$\tag{11}$$

- **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 ELBO (L)
-

$$log[p_\theta(\mathbf{x_0})] = log \int_{\mathbf{x_{1:T}}} p(\mathbf{x_0}, \mathbf{x_{1:T}}) d\mathbf{x_{1:T}} \tag{12}$$

$$log[p_\theta(\mathbf{x_0})] = log \int_{\mathbf{x_{1:T}}} p(\mathbf{x_0}, \mathbf{x_{1:T}}) \frac{q(\mathbf{x_{1:T}}|\mathbf{x_0})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})} d\mathbf{x_{1:T}} \tag{13}$$

by definition, (14)

$$\mathbb{E}_q[f(\mathbf{x}_{1:T})] = \int q(\mathbf{x}_{1:T}|\mathbf{x}_0)\, f(\mathbf{x}_{1:T})\, d\mathbf{x}_{1:T} \tag{15}$$

$$\implies log[p_\theta(\mathbf{x_0})] = log(\mathbb{E}_q\left[\frac{p(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})}\right]) \tag{16}$$

the Jensen's inequality tells us (17)

$$f(\mathbb{E}(\mathbf{X})) \geq \mathbb{E}(f(\mathbf{X})) \tag{18}$$

for any concave function f. log is concave, hence: (19)

$$log[p_\theta(\mathbf{x_0})] \geq \mathbb{E}_q\left[log \frac{p(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})}\right] \tag{20}$$

We define the Evidence Lower Bound L as: (21)

$$L := \mathbb{E}_q\left[- log \frac{p(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})}\right] \tag{22}$$

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:

$$L := \mathbb{E}_q\left[-\,log\frac{p(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})}\right] \tag{24}$$

$$L = \mathbb{E}_q\left[-\,log\frac{p(\mathbf{x_T})}{q(\mathbf{x_T}|\mathbf{x_0})}\right] - logp_\theta(\mathbf{x_0}|\mathbf{x_1}) - \mathbb{E}_q\left[\sum_{t=2}^{T}\mathbf{D_{KL}}\left[q(\mathbf{x_{t-1}}|\mathbf{x_t},\mathbf{x_0})||p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}))\right]\right] \tag{25}$$

- **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.**

Defining $q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0})$

$$q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0}) = \frac{q(\mathbf{x_t}|\mathbf{x_{t-1}}, \mathbf{x_0})q(\mathbf{x_{t-1}}|\mathbf{x_0})}{q(\mathbf{x_t}|\mathbf{x_0})} \tag{26}$$

$$\text{we know the q distribution from Definition, hence} \tag{27}$$

$$q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0}) \text{ is a product of known Gaussians over another known Gaussian} \tag{28}$$

$$\mu_q(\mathbf{x_t}, \mathbf{x_0}) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x_0}}{(1-\bar{\alpha}_t)} \tag{29}$$

$$\Sigma_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I} \tag{30}$$

$$\implies q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0}) = \mathcal{N}(\mathbf{x_{t-1}}; \mu_q(\mathbf{x_t}, \mathbf{x_0}); \Sigma_q(t)) \tag{31}$$

Defining $p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t})$

$$p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}) = \mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_\theta(t)) \tag{32}$$

$$\text{from DDPM's paper:} \tag{33}$$

$$\Sigma_\theta(t) = \Sigma_q(t) \tag{34}$$

$$\text{we are only left with the distribution's mean } \mu_\theta \tag{35}$$

$$\implies p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}) = \mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t)) \tag{36}$$

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

$$d_1(x) = \mathcal{N}(\mu_1, \sigma^2) \tag{37}$$

$$d_2(x) = \mathcal{N}(\mu_2, \sigma^2) \tag{38}$$

$$\text{The KL divergence } D_{KL}(d_1|d_2) \text{ is given by:} \tag{39}$$

$$D_{KL}(d_1|d_2) = \frac{(\mu_1 - \mu_2)^2}{2\sigma^2} \tag{40}$$

$$\text{Hence,} \tag{41}$$

$$\mathbf{D_{KL}}\left[q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0})||p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t})\right] = \mathbf{D_{KL}}\left(\mathcal{N}(\mathbf{x_{t-1}}; \mu_q(\mathbf{x_t}, \mathbf{x_0}); \Sigma_q(t)), \mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t))\right) \tag{42}$$

$$= \frac{1 - \bar{\alpha}_t}{2(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}||(\mu_q - \mu_\theta)_2^2|| \tag{43}$$

- **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
-

$$\mu_q(\mathbf{x_t}, \mathbf{x_0}) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x_0}}{(1-\bar{\alpha}_t)} \tag{44}$$

$$\text{we can define the prediction} \tag{45}$$

$$\mu_\theta(\mathbf{x_t}) := \hat{\mu}_q(\mathbf{x_t}, \mathbf{x_0}) \tag{46}$$

$$= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_\theta}{(1-\bar{\alpha}_t)} \tag{47}$$

$$\implies \mathbf{D_{KL}}\left(\mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t)), \mathcal{N}(\mathbf{x_{t-1}}; \mu_q(\mathbf{x_t}, \mathbf{x_0}); \Sigma_q(t))\right) \tag{48}$$

$$= \frac{(1-\bar{\alpha}_t)(\bar{\alpha}_{t-1})}{2(1-\alpha_t)(1-\bar{\alpha}_{t-1})}||(\mathbf{x}_\theta - \mathbf{x_0})_2^2|| \tag{49}$$

- **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

- We can rewrite $\mathbf{x_t}$ and $\mathbf{x_0}$ as functions of the added gaussian noises
- $$q(\mathbf{x_t}|\mathbf{x_0}) := \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} \times x_0; (1 - \bar{\alpha}_t)\mathbb{I}) \quad (50)$$
  which let's us write (51)
  $$\mathbf{x_t} = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \quad (52)$$
  $$\implies \mathbf{x_0} = \frac{\mathbf{x_t} - \sqrt{1 - \bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}} \quad (53)$$
  for a Standard Gaussian Noise ϵ. (54)
  We can now define our prediction $\hat{\epsilon} = \epsilon_\theta$ (55)
  $$\mathbf{x}_\theta = \frac{\mathbf{x_t} - \sqrt{1 - \bar{\alpha}_t}\epsilon_\theta}{\sqrt{\bar{\alpha}_t}} \quad (56)$$
  $$\implies \frac{(1 - \bar{\alpha}_t)(\bar{\alpha}_{t-1})}{2(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}||(\mathbf{x}_\theta - \mathbf{x_0})_2^2|| = \frac{(1 - \bar{\alpha}_t)(\bar{\alpha}_{t-1})}{2(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})} \frac{(1 - \alpha_t)^2}{(1 - \bar{\alpha}_t)\alpha_t}||(\epsilon_\theta - \epsilon)_2^2|| \quad (57)$$
- **The DDPM paper authors mention that optimizing $||(\epsilon_\theta - \epsilon)_2^2||$ without the scaling factor with the cumulative noise α_t is enough.**

Noise Predictor Training

1: **repeat**
2: $\mathbf{x_0} \sim \mathbf{q}(\mathbf{x_0})$ ▷ Sample image from training set
3: $\mathbf{t} \sim \mathbf{Uniform}(\{\mathbf{1}, \dots, \mathbf{T}\})$ ▷ 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

-

$$p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}) = \mathcal{N}(\mathbf{x_{t-1}}; \mu_\theta(\mathbf{x_t}); \Sigma_q(t)) \tag{58}$$

$$\mu_\theta(\mathbf{x_t}) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}\mathbf{x_t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_\theta}{(1-\bar{\alpha}_t)} \tag{59}$$

$$\mathbf{x}_\theta = \frac{\mathbf{x_t} - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta}{\sqrt{\bar{\alpha}_t}} \tag{60}$$

$$\implies \mu_\theta(\mathbf{x_t}) = \frac{\mathbf{x_t}}{\sqrt{\alpha_t}} - \frac{(1-\alpha_t)(\sqrt{1-\bar{\alpha}_t})}{(1-\bar{\alpha}_t)(\sqrt{\alpha_t})}\epsilon_\theta = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x_t} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta\right) \tag{61}$$

$$\Sigma_\theta(t) = \Sigma_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I} \tag{62}$$

We now have defined $\mathbf{x_{t-1}}$'s mean and variance given $\mathbf{x_t}$ which let's us write it as (63)

$$\mathbf{x_{t-1}} = \mu_\theta(\mathbf{x_t}) + \sqrt{\Sigma_\theta(t)}\mathbf{z} \tag{64}$$

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{65}$$

Sampling Algorithm

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

References I

[1] Ho, J., Jain, A., and Abbeel, P. (2020). *Denoising Diffusion Probabilistic Models*. https://arxiv.org/pdf/2006.11239

[2] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. https://arxiv.org/abs/1503.03585

[3] 3Blue1Brown. (n.d.). *Neural Networks [7.8]: Deep Learning - Variational Bound (YouTube)*. https://www.youtube.com/watch?v=pStDscJh2Wo

[4] Jake Tae. (2021). *A Step Up with Variational Autoencoders*. https://jaketae.github.io/study/vae/

[5] Jake Tae. (2021). *From ELBO to DDPM*. https://jaketae.github.io/study/elbo/

References II

[6] Yang, X. (2017). *Understanding the Variational Lower Bound*. https://xyang35.github.io/2017/04/14/variational-lower-bound/

[7] AI Coffee Break with Letitia. (n.d.). *Denoising Diffusion Probabilistic Models — DDPM Explained (YouTube)*. https://www.youtube.com/watch?v=H45lF4sUgiE&t=880s

Hora da Implementação :)