CS5242 : Neural Networks and Deep Learning

Lecture 8 : Diffusion Models

Semester 2 2024/25

Xavier Bresson

https://x.com/xbresson

Department of Computer Science
National University of Singapore (NUS)

Outline

- Introduction
- Vanilla diffusion models (DDPM)
- Lower bound on data distribution
- Noising process
- Denoising process
- Learning to denoise
- Conclusion

Outline

Introduction

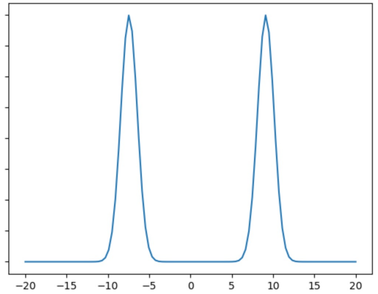
- Generative models can be broadly categorized based on the type of data they generate -- discrete or continuous.
- For discrete data (e.g. a sequence of words, where each word belongs to a finite dictionary of tokens), the most common models are auto-regressive models. These include Recurrent Neural Networks (RNNs) and Transformer-based Language Models (LMs).
- For continuous data (e.g. an image represented as a grid of pixels with continuous values in a RGB color space $[0,255]^3$), the primary generative models include:
- Variational Autoencoders (VAEs): Stable and robust during training, but often produce less accurate results.
- Generative Adversarial Networks (GANs): Capable of generating highly realistic outputs, but difficult to train.
- Diffusion Models (DMs): Combine robustness in training with high-quality outputs. DMs have become the state-of-the-art for image generation in computer vision.

Lecture approach

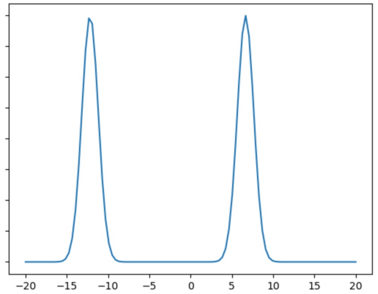
- In this lecture, we focus on Diffusion Models.
- Our main goal is to derive the governing equations of diffusion models from fundamental statistical principles -- using only basic probabilistic tools such as Bayes' theorem, the expectation of function of random variables, etc.
- The lecture is self-contained.
- While the full derivation is lengthy, each step is sound and requires no additional computations beyond what is presented.

Datasets

- We will use the following datasets :
  - An artificial dataset of mixtures of two identical Gaussian distributions with random shifts. The data dimensionality is $d_x = N$ (all sequences have N continuous variables).
  - The MNIST image dataset with dimensionality $d_x = N_x \times N_y \times 3$ (N_x pixel width, N_y pixel height, and each pixel has 3 color values).

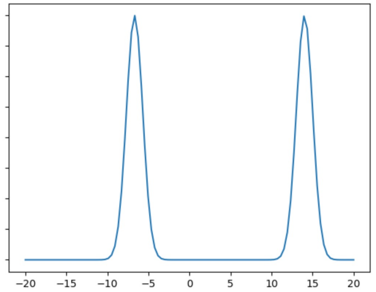


x_1

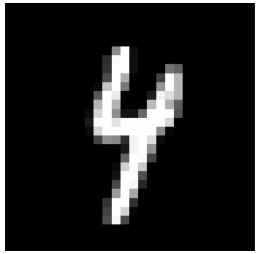


x_2

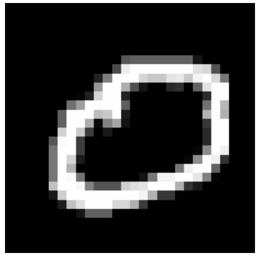
...



x_n

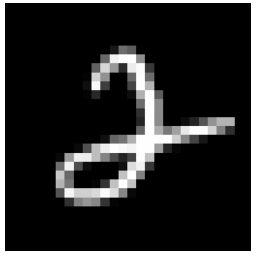


x_1



x_2

...



x_n

Outline

- Introduction
- Vanilla diffusion models (DDPM)
- Lower bound on data distribution
- Noising process
- Denoising process
- Learning to denoise
- Conclusion

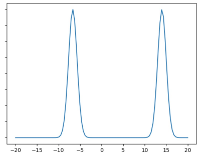
Generative process

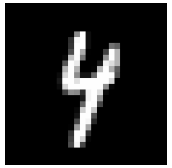
- DDPM (Ho-Jain-Abbeel 2020) : Produce new data using a sequence of denoising steps.
  Denoising Diffusion Probabilistic Models

$$x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_{t-1} \leftarrow x_t \leftarrow x_{t+1} \leftarrow \cdots \leftarrow x_{T-1} \leftarrow x_T$$

$$x_0 \sim p(x)$$

Probability of training/real data (what we want to estimate)





$$x_{t-1} \sim p(x_{t-1}|x_t) = \mathcal{N}(\mu_{\text{denoise}}(x_t, n_t), \sigma^2_{\text{denoise}}(t)\mathrm{I})$$

$$x_{t-1} = a^x_t x_t - b^n_t n_t + \sigma^2_{\text{denoise}}(t) z, \;\; z \sim \mathcal{N}(0, \mathrm{I})$$

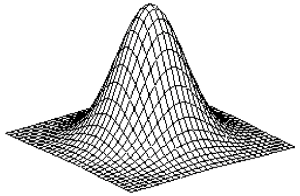
x_{t-1} is produced with a linear combination of the current denoised data x_t, an estimation of the noise n_t and a pure random noise z.

Starting from pure noise $x = x_T$, gradually remove noise to generate intermediate states x_{T-1}, x_{T-2},… until reaching a clean data $x = x_0$, which belongs to the training distribution p(x).

Backward process

$$x_T \sim \mathcal{N}(0, \mathrm{I})$$

x_T is sampled from a prior (unconditional) probability.



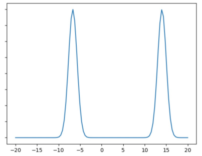
Forward process

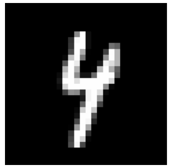
- The denoising steps are learned from the forward pass, which consists in adding noise to the original data :

$$x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_{t-1} \longrightarrow x_t \longrightarrow x_{t+1} \longrightarrow \cdots \longrightarrow x_{T-1} \longrightarrow x_T$$

$$x_0 \sim p(x)$$

Probability of training/real data





$$x_t \sim p(x_t|x_{t-1}) = \mathcal{N}\left(\mu_{\text{noise}}(x_{t-1}), \sigma^2_{\text{noise}}(t)\mathrm{I}\right)$$

$$x_t = c^x_t x_{t-1} + \sigma^2_{\text{noise}}(t)z, \;\; z \sim \mathcal{N}(0, \mathrm{I})$$

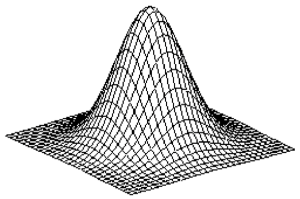
x_t is computed with a linear combination of the current noisy data x_{t-1} and a pure random noise z.

Starting with clean data $x = x_0$ from the training set, add noise to generate intermediate states x_1, x_2,... until reaching a high noise level $x = x_T$, where the original structure is no longer recognizable.

Forward process

$$x_T \sim \mathcal{N}(0, \mathrm{I})$$

x_T samples from a prior probability



Outline

Data distribution

- **The goal is to estimate the prior data distribution p(x) from the sequence of noisy steps (forward process) and denoised steps (backward process).**
- **We start by expressing the data distribution w.r.t. the intermediate states $\mathbf{x}_1$, $\mathbf{x}_2$,... :**

$$\begin{aligned}\log p(x) &= \log p(x_0), \ \text{ with } x = x_{t=0}\\ &= \log \int p(x_0, x_1, ..., x_T) dx_1 dx_2 ... dx_T, \ \text{ marginal integration}\\ &= \log \int p(x_{0:T}) dx_{1:T}, \ \text{ with the sequence notation } x_{0:T} = x_0, x_1, ..., x_T\\ &= \log \int dx_{1:T} p(x_{0:T}) \frac{p(x_{1:T}|x_0)}{p(x_{1:T}|x_0)}, \ \text{ where } p(x_{1:T}|x_0) \text{ is the conditional}\end{aligned}$$

probability of having the sequence $x_{1:T}$ starting from the original data x_0.

Data distribution

- **Then, we decompose the data distribution in terms of forward and backward probabilities :**

$$\log p(x) = \log \mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\frac{p(x_{0:T})}{p(x_{1:T}|x_0)}\right], \text{ def of expectation: } \int_x p(x)f(x)dx = \mathbb{E}_{x\sim p(x)}[f(x)]$$

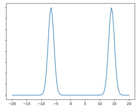
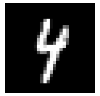
$$\geq \mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \frac{p(x_{0:T})}{p(x_{1:T}|x_0)}\right], \text{ Jensen's inequality: } \varphi(\mathbb{E}[f(x)]) \geq \mathbb{E}[\varphi(f(x))], \ \varphi \text{ concave}$$

with chain rule, we have

$p(x_{0:T}) = p(x_T)\ p(x_0|x_1)\ \Pi_{t=2}^T\ p(x_{t-1}|x_t)$, given the backward chain process

$p(x_{1:T}|x_0) = p(x_1|x_0)\ \Pi_{t=2}^T\ p(x_t|x_{t-1}, x_0)$, given the forward chain process

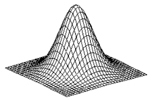
$$\geq \underbrace{\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p(x_0|x_1)}{p(x_1|x_0)}\right]}_{\textbf{(First term)}} + \underbrace{\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \Pi_{t=2}^T \frac{p(x_{t-1}|x_t)}{p(x_t|x_{t-1}, x_0)}\right]}_{\textbf{(Second term)}}$$

$p(x_{t+1}|x_t)$ Forward process (noising step)

$$x_0 \rightleftarrows x_1 \rightleftarrows \cdots \rightleftarrows x_{t-1} \rightleftarrows x_t \rightleftarrows x_{t+1} \rightleftarrows \cdots \rightleftarrows x_{T-1} \rightleftarrows x_T$$

$p(x_{t-1}|x_t)$ Backward process (denoising step)



Lower bound

- **Let us develop the second term of the lower bound :**

$$\mathbb{E}\Big[\log \Pi_{t=2}^T \frac{p(x_{t-1}|x_t)}{p(x_t|x_{t-1},x_0)}\Big] = \mathbb{E}\Big[\log \Pi_{t=2}^T \frac{p(x_{t-1}|x_t)}{\frac{p(x_{t-1}|x_t,x_0)\ p(x_t|x_0)}{p(x_{t-1}|x_0)}}\Big],\ \text{Bayes theorem: } p(a|b)p(b)=p(b|a)p(a)$$

Here: $p(a|b,x_0)p(b|x_0)=p(b|a,x_0)p(a|x_0)$

$$= \mathbb{E}\Big[\log\Big(\Pi_{t=2}^T \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}\ \Pi_{t=2}^T \frac{p(x_{t-1}|x_0)}{p(x_t|x_0)}\Big)\Big]$$

$$= \mathbb{E}\Big[\log\Big(\Pi_{t=2}^T \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}\ \frac{p(x_1|x_0)}{p(x_T|x_0)}\Big)\Big],\ \text{terms cancel by recursion}$$

$$= \mathbb{E}\Big[\log \Pi_{t=2}^T \underbrace{\frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}}\Big] + \mathbb{E}\Big[\log \frac{p(x_1|x_0)}{p(x_T|x_0)}\Big],\ \text{as } \log(ab)=\log a+\log b$$

A key objective is to reformulate the denoising process $p(x_{t-1}|x_t,x_0)$ to be independent of the clean data x_0 with $p(x_{t-1}|x_t)$ in order to generate new data where x_0 is unknown.

Lower bound

- **Putting together the first term and second term of the lower bound :**

$$
\begin{aligned}
\log p(x) &\geq \underbrace{\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p(x_0|x_1)}{p(x_1|x_0)}\right]}_{\textbf{(First term)}} + \underbrace{\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \Pi_{t=2}^{T} \frac{p(x_{t-1}|x_t)}{p(x_t|x_{t-1},x_0)}\right]}_{\textbf{(Second term)}} \\
&\geq \mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p(x_0|x_1)}{p(x_1|x_0)}\right] + \mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \Pi_{t=2}^{T} \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}\right] \\
&\qquad + \mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \frac{p(x_1|x_0)}{p(x_T|x_0)}\right] \textbf{(previous slide)} \\
&\geq \mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p(x_0|x_1)}{p(x_1|x_0)}\frac{p(x_1|x_0)}{p(x_T|x_0)}\right] + \mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \Pi_{t=2}^{T} \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}\right] \\
&\geq \mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p(x_0|x_1)}{p(x_T|x_0)}\right] + \mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \Pi_{t=2}^{T} \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}\right] \\
&\geq \underbrace{\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log p(x_0|x_1)\right]}_{\textbf{(Term 1)}} + \underbrace{\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \frac{p(x_T)}{p(x_T|x_0)}\right]}_{\textbf{(Term 2)}} \\
&\qquad + \underbrace{\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\left[\log \Pi_{t=2}^{T} \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}\right]}_{\textbf{(Term 3)}}
\end{aligned}
$$

Lower bound

- **Focusing on Term 1 and Term 2 :**

$$\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\Big[\log p(x_0|x_1)\Big] \quad \textbf{(Term 1)}$$

$$= \mathbb{E}_{x_1\sim p(x_1|x_0)}\Big[\log p(x_0|x_1)\Big], \ \text{function is independent of } x_{2:T}$$

$$\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\Big[\log \frac{p(x_T)}{p(x_T|x_0)}\Big] \quad \textbf{(Term 2)}$$

$$= \mathbb{E}_{x_T\sim p(x_T|x_0)}\Big[\log \frac{p(x_T)}{p(x_T|x_0)}\Big], \ \text{function is independent of } x_{1:T-1}$$

$$= -D_{\text{KL}}\big(p(x_T|x_0), p(x_T)\big), \ \text{definition of Kullback-Leibler divergence}$$

$$D_{\text{KL}}(p_1, p_2) = -\int_x p_1(x) \log \frac{p_2(x)}{p_1(x)} dx = -\mathbb{E}_{x\sim p_1(x)}\Big[\log \frac{p_2(x)}{p_1(x)}\Big]$$

Lower bound

- **Expanding Term 3 :**

$$\mathbb{E}_{x_{1:T}\sim p(x_{1:T}|x_0)}\Big[\log \Pi_{t=2}^{T} \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}\Big]$$

$$= \mathbb{E}_{x_{t-1},x_t\sim p(x_{t-1},x_t|x_0)}\Big[\log \Pi_{t=2}^{T} \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}\Big], \text{ function only dependends of } x_{t-1},x_t$$

$$= \sum_{t=2}^{T} \mathbb{E}_{x_{t-1},x_t\sim p(x_{t-1},x_t|x_0)}\Big[\log \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}\Big], \text{ as } \log \Pi_{t=2}^{T} = \sum_{t=2}^{T} \log$$

$$= \sum_{t=2}^{T} \int dx_{t-1}dx_t \; p(x_{t-1},x_t|x_0) \log \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}, \text{ def of expectation}$$

$$= \sum_{t=2}^{T} \int dx_{t-1}dx_t \; p(x_{t-1}|x_t,x_0)\; p(x_t|x_0) \log \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}, \text{ Bayes: } p(a,b|x_0)=p(a|b,x_0)p(b|x_0)$$

$$=, -\sum_{t=2}^{T} \mathbb{E}_{x_t\sim p(x_t|x_0)} \; D_{\mathrm{KL}}\big(p(x_{t-1}|x_t,x_0), p(x_{t-1}|x_t)\big) \text{ definition of Kullback-Leibler and expectation}$$

The KL distance is computed in the next slide.

with $D_{\mathrm{KL}}\big(p(x_{t-1}|x_t,x_0), p(x_{t-1}|x_t)\big) = \int dx_{t-1} p(x_{t-1}|x_t,x_0) \log \frac{p(x_{t-1}|x_t)}{p(x_{t-1}|x_t,x_0)}$

and $\mathbb{E}_{x_t\sim p(x_t|x_0)} D_{\mathrm{KL}} = \int dx_t p(x_t|x_0) D_{\mathrm{KL}}\big(p(x_{t-1}|x_t,x_0), p(x_{t-1}|x_t)\big)$

Denoising probabilities

- **The backward distributions are defined as Gaussian distributions :**

Given that

$$p(x_{t-1}|x_t,x_0) = \mathcal{N}(\mu_{\text{denoise}_1}(x_t,x_0), \sigma^2(t)\mathrm{I}),\ \text{backward pass (denoising process) conditioned on } x_t, x_0$$

$$p(x_{t-1}|x_t) = \mathcal{N}(\mu_{\text{denoise}_2}(x_t), \sigma^2(t)\mathrm{I}),\ \text{backward pass (denoising process) conditioned on } x_t$$

then the KL distance between these two distributions is

$$D_{\mathrm{KL}}\big(p(x_{t-1}|x_t,x_0), p(x_{t-1}|x_t)\big) = \frac{1}{2\sigma^2(t)}\Big\|\mu_{\text{denoise}_1}(x_t,x_0) - \mu_{\text{denoise}_2}(x_t)\Big\|_2^2$$

since $D_{\mathrm{KL}}(p_1,p_2) = \dfrac{1}{2}\Big[\log\Big(\dfrac{\sigma_2^2}{\sigma_1^2}\Big) + \dfrac{\sigma_1^2}{\sigma_2^2} - 1 + \dfrac{\|\mu_2-\mu_1\|_2^2}{\sigma_2^2}\Big],$

for $p_1 = \mathcal{N}(\mu_1, \sigma_1^2\mathrm{I}),\ p_2 = \mathcal{N}(\mu_2, \sigma_2^2\mathrm{I}),$

which reduces to

$D_{\mathrm{KL}}(p_1,p_2) = \dfrac{\|\mu_2-\mu_1\|_2^2}{2\sigma^2}$ for two Gaussians with same covariance.

Lower bound

- **At this stage, the lower bound estimate of the prior data distribution p(x) is :**

$$\log p(x) \geq \underbrace{\mathbb{E}_{x_1 \sim p(x_1|x_0)}\Big[\log p(x_0|x_1)\Big]}_{\textbf{(Term 1)}} - \underbrace{D_{\mathrm{KL}}\big(p(x_T|x_0), p(x_T)\big)}_{\textbf{(Term 2)}}$$
$$- \sum_{t=2}^{T} \mathbb{E}_{x_t \sim p(x_t|x_0)} \underbrace{D_{\mathrm{KL}}\big(p(x_{t-1}|x_t, x_0), p(x_{t-1}|x_t)\big)}_{\frac{1}{2\sigma^2(t)}\big\|\mu_{\mathrm{denoise}_1}(x_t,x_0) - \mu_{\mathrm{denoise}_2}(x_t)\big\|_2^2}$$

(Term 3)

with

$$p(x_{t-1}|x_t, x_0) = \mathcal{N}(\mu_{\mathrm{denoise}_1}(x_t, x_0), \sigma^2(t)\mathrm{I}), \ \text{backward pass (denoising process)}$$
$$p(x_{t-1}|x_t) = \mathcal{N}(\mu_{\mathrm{denoise}_2}(x_t), \sigma^2(t)\mathrm{I}), \ \text{backward pass (denoising process)}$$

Outline

Noising process

- **Let us prove that**

$$p(x_t|x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathrm{I}\big)$$

when we choose : $p(x_t|x_{t-1}) = \mathcal{N}\big(\sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathrm{I}\big)$,

where parameters α_t are the schedulers that control the noise level (see next slide).

By recursion, we have

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}, \epsilon_{t-1} \sim \mathcal{N}\big(0,\mathrm{I}\big), \text{ sample definition of } p(x_t|x_{t-1})$$

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

$$x_t = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \underbrace{\sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1}}_{z}, \text{ and for the covariance part}$$

$$\mathbb{E}\big[zz^T\big] = \big[(\sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}})^2 + (\sqrt{1-\alpha_t})^2\big]\mathrm{I} = \big[1-\alpha_t\alpha_{t-1}\big]\mathrm{I}$$

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

After multiple recursions, we finally have

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0, \epsilon_0 \sim \mathcal{N}\big(0,\mathrm{I}\big), \text{ with } \bar{\alpha}_t = \Pi_{i=1}^t\alpha_i$$

Noising process

- **A few observations on the noising process**

  The noising process $p(x_t|x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathrm{I}\big)$ with sample
  $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0, \epsilon_0 \sim \mathcal{N}\big(0, \mathrm{I}\big)$
  has an analytical form that only depends on the original clean data x_0.

  This means that no neural network is required to compute a noisy version of x_0 (fast process).

  For $t = 0$, $x_{t=0} = x_0$ (original clean image) as $\bar{\alpha}_{t=0} = 1$ and

  for $t = T$, $x_T = \epsilon_T, \epsilon_T \sim \mathcal{N}\big(0, \mathrm{I}\big)$ (Gaussian distribution) as $\bar{\alpha}_{t=T} = 0$.

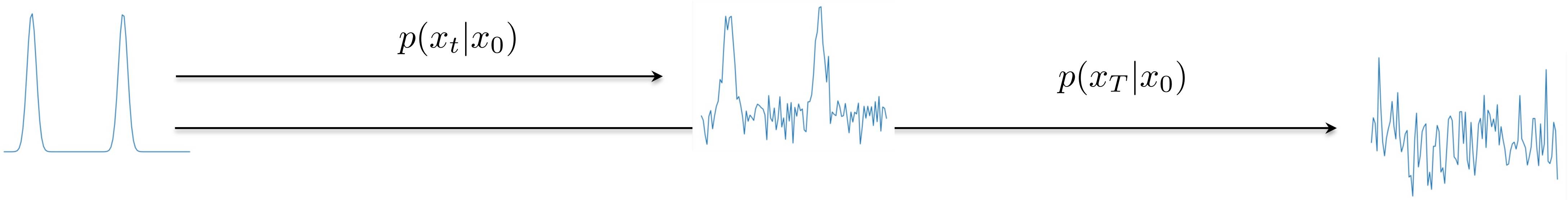
  Parameters α_t are called schedulers to transition smoothly from image to pure noise s.a.

  $\alpha_t = \{1, 1-\delta, 1-2\delta, ..., , 1-t\delta, ..., 0\},$ where $\delta = \dfrac{T}{N_t}$ is the time step and N_t the number of steps.

Noising process

- **Illustration :**

$$x_0 \rightleftarrows x_1 \rightleftarrows \cdots \rightleftarrows x_{t-1} \underset{}{\overset{p(x_t|x_{t-1})}{\rightleftarrows}} x_t \rightleftarrows x_{t+1} \rightleftarrows \cdots \rightleftarrows x_{T-1} \rightleftarrows x_T$$



$$p(x_t|x_{t-1}) = \mathcal{N}\big(\sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathrm{I}\big)$$

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

$$p(x_t|x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathrm{I}\big)$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0, \epsilon_0 \sim \mathcal{N}\big(0,\mathrm{I}\big), \text{ with } \bar{\alpha}_t = \Pi_{i=1}^t \alpha_i$$

closed-form solution (no learning is required)

Outline

- Introduction
- Vanilla diffusion models (DDPM)
- Lower bound on data distribution
- Noising process
- Denoising process
- Learning to denoise
- Conclusion

Denoising process

- **Returning to the backward distribution :**

$$
\begin{aligned}
p(x_{t-1}|x_t,x_0) &= \mathcal{N}\big(\mu_{\text{denoise}_1}(x_t,x_0),\sigma^2(t)\mathrm{I}\big)\\
&= \frac{p(x_t|x_{t-1},x_0)p(x_{t-1}|x_0)}{p(x_t|x_0)},\ \text{Bayes theorem}
\end{aligned}
$$

with

$$p(x_t|x_{t-1},x_0) = p(x_t|x_{t-1}) = \mathcal{N}\big(\sqrt{\alpha_t}x_{t-1},(1-\alpha_t)\mathrm{I}\big)$$

$$p(x_{t-1}|x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_{t-1}}x_0,(1-\bar{\alpha}_{t-1})\mathrm{I}\big)$$

$$p(x_t|x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}x_0,(1-\bar{\alpha}_t)\mathrm{I}\big)$$

The backward probability $p(x_{t-1}|x_t,x_0)$ is defined as the product of three Gaussians.

Gaussian product

- **The product of the three Gaussians can be written as :**

$$
\begin{aligned}
p(x_{t-1}|x_t, x_0) &\propto \exp\Big(-\frac{(x_t - \sqrt{\alpha_t} x_{t-1})^2}{2(1-\alpha_t)} - \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}} x_0)^2}{2(1-\bar{\alpha}_{t-1})} + \frac{(x_t - \sqrt{\bar{\alpha}_t} x_0)^2}{2(1-\bar{\alpha}_t)} \Big) \\
&\propto \exp\Big(-\frac{(x_t - \mu_{\text{denoise}_1}(x_t, x_0))^2}{2\sigma^2(t)} \Big)
\end{aligned}
$$

Solving the above quadratic equation provides the mean and the covariance of $p(x_{t-1}|x_t, x_0)$:

$$
\begin{aligned}
\mu_{\text{denoise}_1}(x_t, x_0) &= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t} x_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t} x_0 \\
\sigma^2(t) &= \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}
\end{aligned}
$$

Denoising process

- **Finally, we arbitrarily design the one-step backward probability $p(x_{t-1}|x_t)$ by analogy to the mean and the covariance of $p(x_{t-1}|x_t,x_0)$:**

$$p(x_{t-1}|x_t) = \mathcal{N}(\mu_{\text{denoise}_2}(x_t), \sigma^2(t)\mathrm{I})$$

with

$$\mu_{\text{denoise}_2}(x_t) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}x_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\tilde{x}_0(x_t)$$

where $\tilde{x}_0(x_t)$ is the approximate denoised data x_0 from x_t.

Recall that $p(x_{t-1}|x_t,x_0) = \mathcal{N}(\mu_{\text{denoise}_1}(x_t,x_0), \sigma^2(t)\mathrm{I})$

with

$$\mu_{\text{denoise}_1}(x_t,x_0) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}x_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}x_0$$

$$\sigma^2(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$

Denoising loss

- **Coming back to the KL term :**

$$D_{\mathrm{KL}}\big(p(x_{t-1}|x_t,x_0),p(x_{t-1}|x_t)\big)=\frac{1}{\sigma^2(t)}\big\|\mu_{\mathrm{denoise}_1}(x_t,x_0)-\mu_{\mathrm{denoise}_2}(x_t)\big\|_2^2$$

$$\text{with } \mu_{\mathrm{denoise}_1}(x_t,x_0)=\frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}x_t+\frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}x_0$$

$$\text{and } \mu_{\mathrm{denoise}_2}(x_t)=\frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}x_t+\frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\tilde{x}_0(x_t)$$

We have

$$D_{\mathrm{KL}}\big(p(x_{t-1}|x_t,x_0),p(x_{t-1}|x_t)\big)=\frac{1}{2\sigma^2(t)}\frac{(1-\alpha_t)^2\bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2}\big\|\tilde{x}_0(x_t)-x_0\big\|_2^2$$

The last equation makes it clear that the backward pass is a denoising process! When the square error term is minimized then function $\tilde{x}_0(x_t)$ has denoised x_t to be as close as possible to the original clean data x_0.

Back to maximizing the lower bound

- **Going back to the lower bound estimate of the prior data distribution p(x) :**

$$\log p(x) \geq \underbrace{\mathbb{E}_{x_1 \sim p(x_1|x_0)} \Big[\log p(x_0|x_1) \Big]}_{\textbf{(Term 1)}} - \underbrace{D_{\mathrm{KL}}\big(p(x_T|x_0), p(x_T)\big)}_{\textbf{(Term 2)}}$$

$$- \sum_{t=2}^{T} \underbrace{\mathbb{E}_{x_t \sim p(x_t|x_0)}}_{\textbf{(Term 3)}} \frac{1}{2\sigma^2(t)} \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2} \big\| \tilde{x}_0(x_t) - x_0 \big\|_2^2$$

with

$$p(x_t|x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, \sigma^2(t)\mathrm{I})$$

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \epsilon_0, \epsilon_0 \sim \mathcal{N}\big(0, \mathrm{I}\big)$$

and specifically for $t = 1$:

$$x_1 \sim p(x_1|x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, \sigma^2(1)\mathrm{I})$$

$$x_1 = \sqrt{\bar{\alpha}_1} x_0 + \sqrt{1-\bar{\alpha}_1} \epsilon_0, \epsilon_0 \sim \mathcal{N}\big(0, \mathrm{I}\big)$$

Maximizing the lower bound

- **Estimating Term 1 :**

$$\mathbb{E}_{x_1\sim p(x_1|x_0)}\Big[\log p(x_0|x_1)\Big]$$

with $\log p(x_0|x_1) = \log \mathcal{N}(\mu_{\text{denoise}_2}(x_1), \sigma^2(1)\mathrm{I})$

$$\text{and } \mu_{\text{denoise}_2}(x_1) = \frac{(1-\bar{\alpha}_0)\sqrt{\alpha_1}}{1-\bar{\alpha}_1}x_1 + \frac{(1-\alpha_1)\sqrt{\bar{\alpha}_0}}{1-\bar{\alpha}_1}\tilde{x}_0(x_1)$$
$$= \tilde{x}_0(x_1) \text{ as } \alpha_0 = \bar{\alpha}_0 = 1 \text{ and } \alpha_1 = \bar{\alpha}_1 = \delta$$

which provides

$$\log p(x_0|x_1) \propto \text{ct} - \frac{1}{2\sigma^2(1)}\frac{(1-\alpha_1)^2\bar{\alpha}_0}{(1-\bar{\alpha}_1)^2}\Big\|\tilde{x}_0(x_1) - x_0\Big\|_2^2$$

$$\text{and } \; -\mathbb{E}_{x_1\sim p(x_1|x_0)}\frac{1}{2\sigma^2(1)}\frac{(1-\alpha_1)^2\bar{\alpha}_0}{(1-\bar{\alpha}_1)^2}\Big\|\tilde{x}_0(x_1) - x_0\Big\|_2^2 + \text{ct}$$

which is equivalent to Term 3 when $t=1$ (ct can be ignored when optimizing) :

$$-\sum_{t=2}^{T}\mathbb{E}_{x_t\sim p(x_t|x_0)}\frac{1}{2\sigma^2(t)}\frac{(1-\alpha_t)^2\bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2}\Big\|\tilde{x}_0(x_t) - x_0\Big\|_2^2$$

Maximizing the lower bound

- **Combining Term 1 and Term 3, we have :**

$$\log p(x) \geq -D_{\mathrm{KL}}\big(p(x_T|x_0), p(x_T)\big)$$
$$- \sum_{t=1}^{T} \mathbb{E}_{x_t \sim p(x_t|x_0)} \frac{1}{2\sigma^2(t)} \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2} \|\tilde{x}_0(x_t) - x_0\|_2^2$$

with

$$p(x_t|x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, \sigma^2(t)\mathrm{I})$$
$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0, \epsilon_0 \sim \mathcal{N}\big(0, \mathrm{I}\big)$$

Additionally, for Term 2 :

$$p(x_T|x_0) = p(x_T) = \mathcal{N}(0, \mathrm{I}) \ \forall x_0, \text{ which implies}$$
$$D_{\mathrm{KL}}\big(p(x_T|x_0), p(x_T)\big) = 0$$

Finally, we have the lower bound :

$$\log p(x) \geq - \sum_{t=1}^{T} \mathbb{E}_{x_t \sim p(x_t|x_0)} \frac{1}{2\sigma^2(t)} \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2} \|\tilde{x}_0(x_t) - x_0\|_2^2$$

Outline

MSE loss

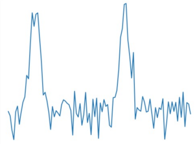
- **Learning to denoise images :**

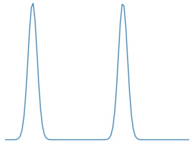
Recall that the noisy process $p(x_t|x_0)$ has a closed-form solution
and hence does not require any learning.
However, the denoising function $\tilde{x}_0(\cdot)$ is unknown and needs to be learned
with a network $x_\theta(\cdot) = \tilde{x}_0(\cdot),$ where θ are the learnable parameters.
The lower bound function :

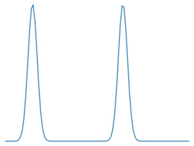
$$-\sum_{t=1}^{T} \mathbb{E}_{x_t \sim p(x_t|x_0)} \frac{1}{2\sigma^2(t)} \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2} \big\| x_\theta(x_t) - x_0 \big\|_2^2$$

is simplified by dropping the weight coefficients, and the loss to optimize is :

$$L(\theta) = \sum_{t=1}^{T} \mathbb{E}_{x_t \sim p(x_t|x_0)} \big\| x_\theta(x_t) - \underbrace{x_0} \big\|_2^2, \text{ with } x_\theta\Big(\quad\Big) =$$







and $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \epsilon_0, \epsilon_0 \sim \mathcal{N}(0, \mathrm{I})$.

Noise prediction

- **Instead of predicting the denoised image, an equivalent approach is to predict the noise added to the clean image :**

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0, \epsilon_0 \sim \mathcal{N}(0, \mathrm{I})$$

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0\right), \text{ clean image}$$

Substituting x_0 into

$$\mu_{\text{denoise}_1}(x_t, x_0) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t} x_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t} x_0$$

We have

$$\mu_{\text{denoise}_1}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \epsilon_0$$

As previously, we opt for the simplest design (by analogy) for the mean μ_{denoise_2} :

$$\mu_{\text{denoise}_2}(x_t) = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \epsilon_\theta(x_t)$$

where $\epsilon_\theta(x_t)$ is the approximated noise ϵ_0 added to x_0 to produce the noisy image x_t.

Denoising loss

- **As previously, considering the KL term :**

$$D_{\mathrm{KL}}\big(p(x_{t-1}|x_t, x_0), p(x_{t-1}|x_t)\big) = \frac{1}{2\sigma^2(t)}\big\|\mu_{\mathrm{denoise}_1}(x_t, x_0) - \mu_{\mathrm{denoise}_2}(x_t)\big\|_2^2$$

$$\text{with } \mu_{\mathrm{denoise}_1}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon_0$$

$$\text{and } \mu_{\mathrm{denoise}_2}(x_t) = \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon_\theta(x_t)$$

We now have

$$D_{\mathrm{KL}}\big(p(x_{t-1}|x_t, x_0), p(x_{t-1}|x_t)\big) = \frac{1}{2\sigma^2(t)}\frac{(1-\alpha_t)^2\bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2}\big\|\epsilon_\theta(x_t) - \epsilon_0\big\|_2^2$$

MSE loss

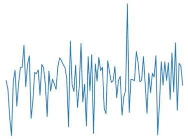
- **Learning to predict noise :**

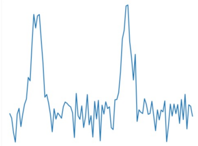
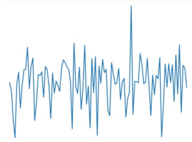
  The lower bound function becomes

$$-\sum_{t=1}^{T}\mathbb{E}_{x_t\sim p(x_t|x_0)}\left[\frac{1}{2\sigma^2(t)}\frac{(1-\alpha_t)^2\bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2}\left\|\epsilon_\theta(x_t)-\epsilon_0\right\|_2^2\right]$$

  And the final loss is (dropping the weight coefficients) :

$$L(\theta)=\sum_{t=1}^{T}\mathbb{E}_{x_t\sim p(x_t|x_0)}\left[\ \left\|\epsilon_\theta(x_t)-\underbrace{\epsilon_0}\right\|_2^2\ \right],$$



with $\epsilon_\theta\Big($  $\Big) =$ 

  When the square error term is minimized then function $\epsilon_\theta(x_t)$ predicts the noise ϵ_0 that was added to the original clean data x_0.

Training steps

- **Training a diffusion model to denoise images :**

Sample a batch of training data.

Draw random time steps for the batch :

$$t \sim \text{Uniform}([1, ..., T])$$

Produce noisy samples :

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_0, \epsilon_0 \sim \mathcal{N}(0, \mathrm{I})$$

Compute the MSE loss :

$$L(\theta) = \sum_{\text{sampled } t} \left\| \varepsilon_\theta(x_t) - \varepsilon_0 \right\|_2^2, \text{ with the network}$$

$$\varepsilon_\theta(x) = \text{Transformer/UNet}_\theta(x) \in \mathbb{R}^{b \times d_x}, \ x \in \mathbb{R}^{b \times d_x}$$

Backpropagation :

Compute gradient of the loss and update net parameters θ.

Generative steps

- **Inference : Generate new data**

Start with $x_T \sim \mathcal{N}\big(0, \mathrm{I}\big)$

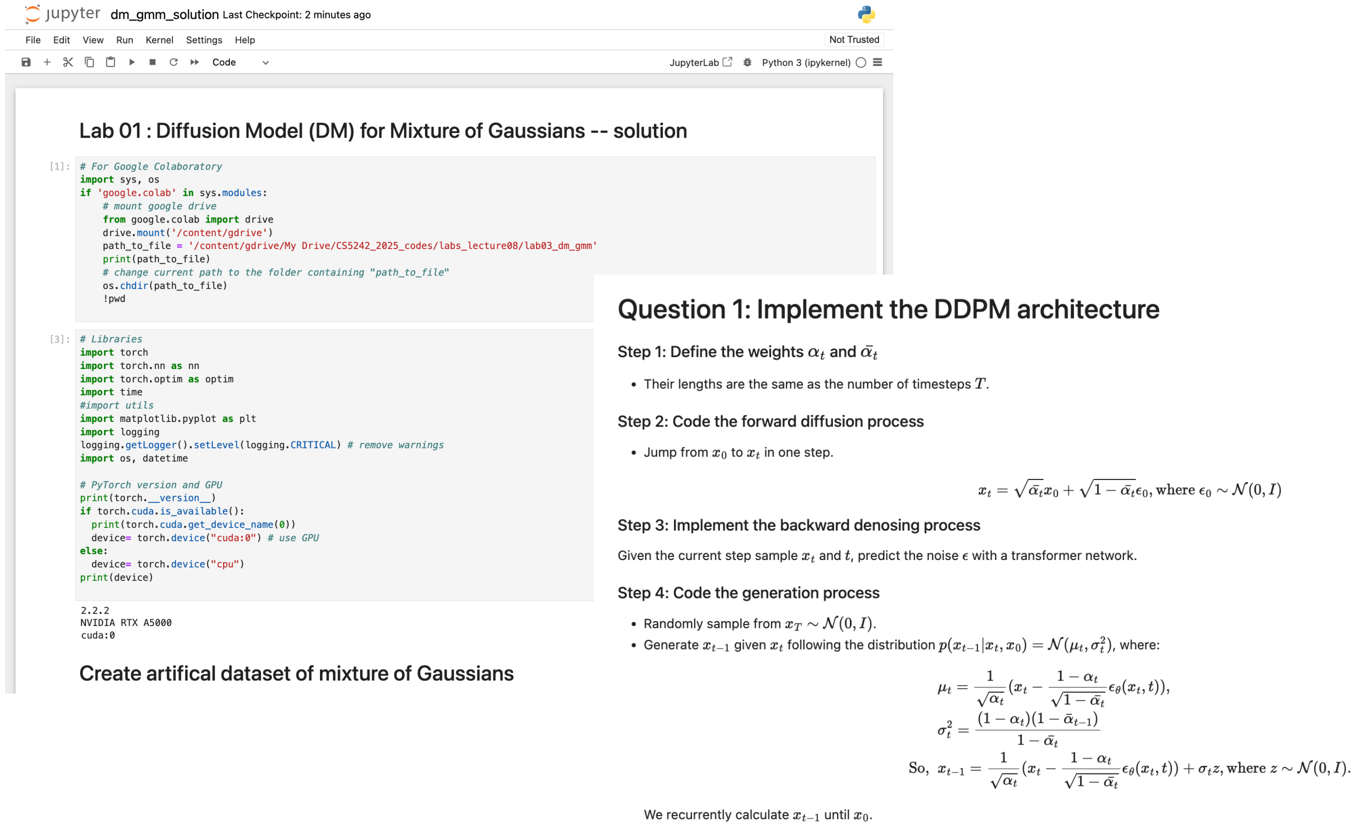
Compute the sequence $x_{T-1}, x_{T-2}, ..., x_0$ by sampling at $t = T, T-1, ..., 1$

$x_{t-1} \sim p_\theta\big(x_{t-1}|x_t\big) = \mathcal{N}\big(\mu_{\text{denoise}_2}\big(x_t\big), \sigma^2(t)\mathrm{I}\big)$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \epsilon_\theta\big(x_t\big) + \sigma^2(t) z, \;\; z \sim \mathcal{N}\big(0, \mathrm{I}\big)$$

Lab 1 : DDPM for mixture of Gaussians

- DDPM with Transformers for artificial 1D dataset.



jupyter dm_gmm_solution Last Checkpoint: 2 minutes ago

File Edit View Run Kernel Settings Help Not Trusted

Code JupyterLab Python 3 (ipykernel)

Lab 01 : Diffusion Model (DM) for Mixture of Gaussians -- solution

```
[1]:
# For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/CS5242_2025_codes/labs_lecture08/lab03_dm_gmm'
    print(path_to_file)
    # change current path to the folder containing "path_to_file"
    os.chdir(path_to_file)
    !pwd
```

```
[3]:
# Libraries
import torch
import torch.nn as nn
import torch.optim as optim
import time
#import utils
import matplotlib.pyplot as plt
import logging
logging.getLogger().setLevel(logging.CRITICAL) # remove warnings
import os, datetime

# PyTorch version and GPU
print(torch.__version__)
if torch.cuda.is_available():
  print(torch.cuda.get_device_name(0))
  device= torch.device("cuda:0") # use GPU
else:
  device= torch.device("cpu")
print(device)
```

```
2.2.2
NVIDIA RTX A5000
cuda:0
```

Create artifical dataset of mixture of Gaussians

Question 1: Implement the DDPM architecture

Step 1: Define the weights α_t and $\bar{\alpha}_t$

- Their lengths are the same as the number of timesteps T.

Step 2: Code the forward diffusion process

- Jump from x_0 to x_t in one step.

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0, \text{ where } \epsilon_0 \sim \mathcal{N}(0, I)$$

Step 3: Implement the backward denosing process

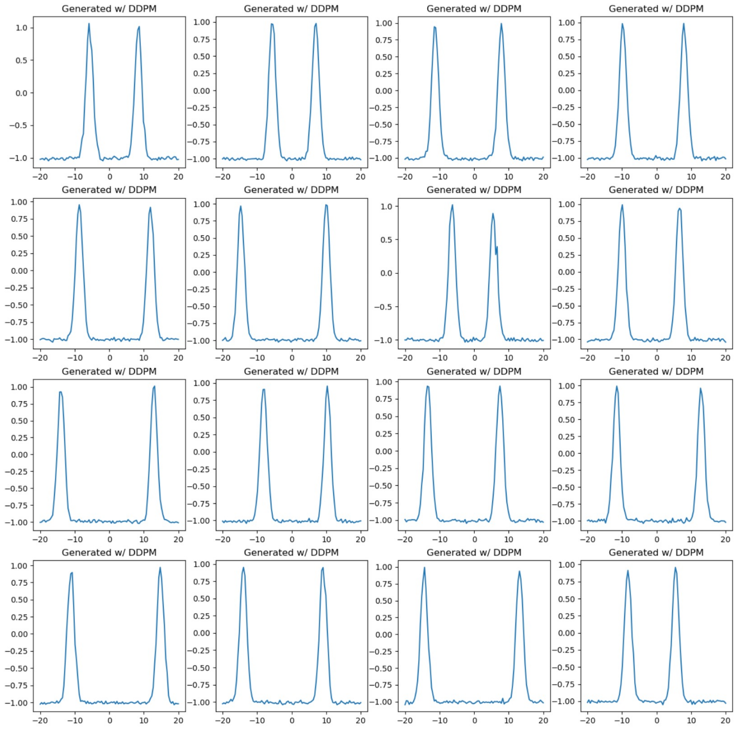
Given the current step sample x_t and t, predict the noise ϵ with a transformer network.

Step 4: Code the generation process

- Randomly sample from $x_T \sim \mathcal{N}(0, I)$.
- Generate x_{t-1} given x_t following the distribution $p(x_{t-1}|x_t, x_0) = \mathcal{N}(\mu_t, \sigma_t^2)$, where:

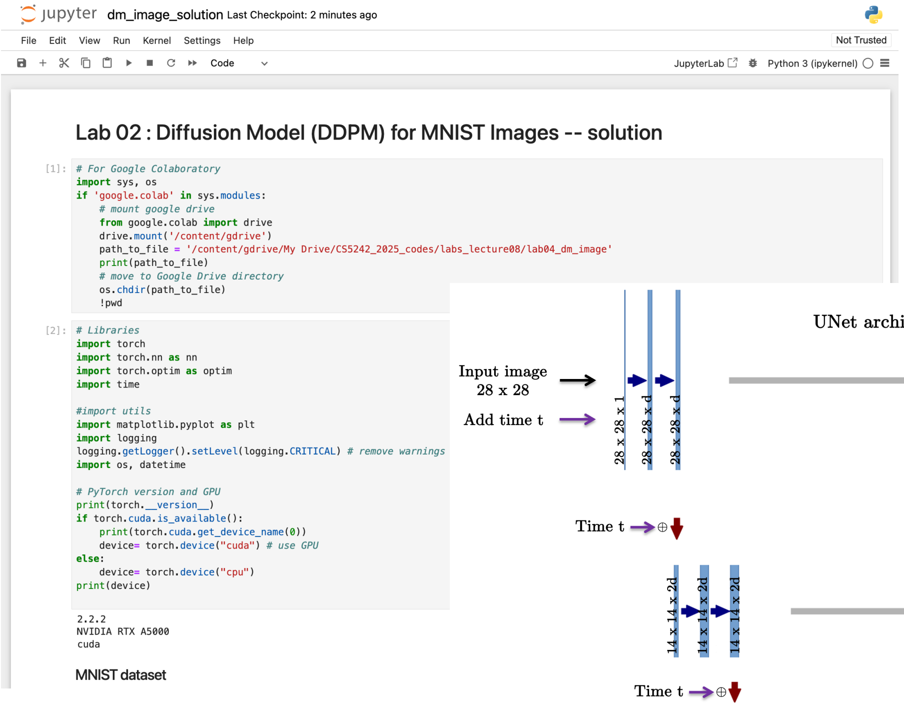
$$\mu_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right),$$
$$\sigma_t^2 = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$
$$\text{So, } x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z, \text{ where } z \sim \mathcal{N}(0, I).$$

We recurrently calculate x_{t-1} until x_0.



Lab 2 : DDPM for images

- DDPM with CNNs (UNet) for 2D MNIST images.


jupyter dm_image_solution Last Checkpoint: 2 minutes ago

File Edit View Run Kernel Settings Help Not Trusted

Code JupyterLab Python 3 (ipykernel)

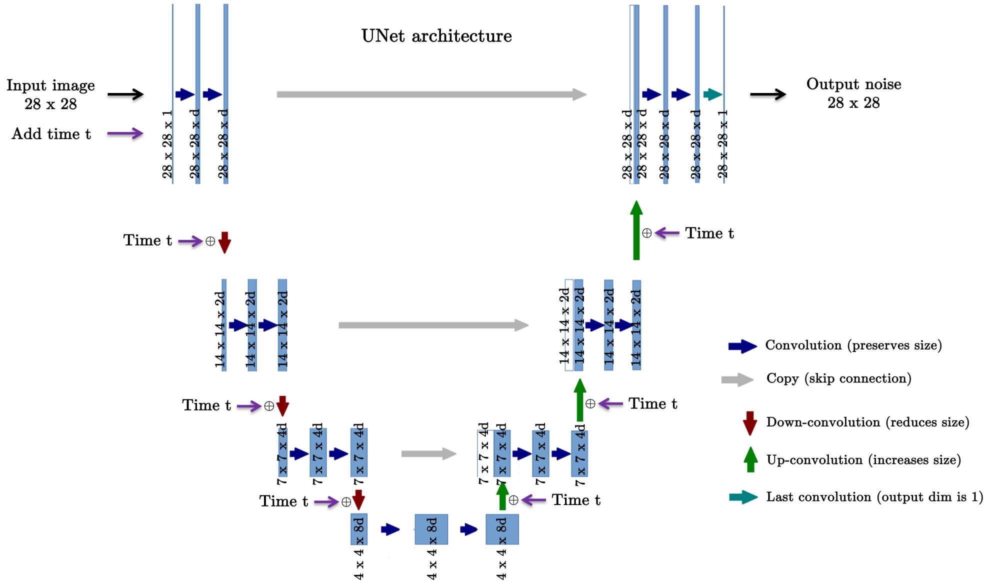
Lab 02 : Diffusion Model (DDPM) for MNIST Images -- solution

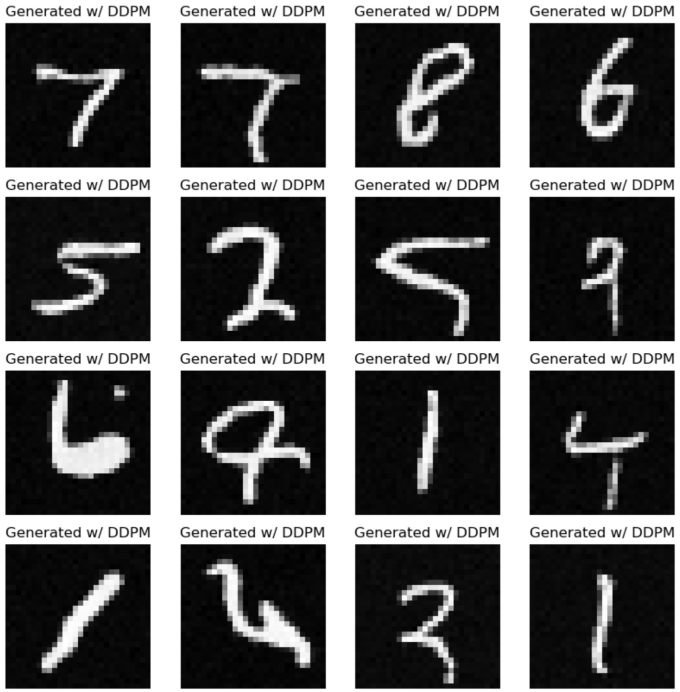
```
[1]: # For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/CS5242_2025_codes/labs_lecture08/lab04_dm_image'
    print(path_to_file)
    # move to Google Drive directory
    os.chdir(path_to_file)
    !pwd
```

```
[2]: # Libraries
import torch
import torch.nn as nn
import torch.optim as optim
import time

#import utils
import matplotlib.pyplot as plt
import logging
logging.getLogger().setLevel(logging.CRITICAL) # remove warnings
import os, datetime

# PyTorch version and GPU
print(torch.__version__)
if torch.cuda.is_available():
    print(torch.cuda.get_device_name(0))
    device= torch.device("cuda") # use GPU
else:
    device= torch.device("cpu")
print(device)
```

```
2.2.2
NVIDIA RTX A5000
cuda
```

MNIST dataset





Outline

- Introduction
- Vanilla diffusion models (DDPM)
- Lower bound on data distribution
- Noising process
- Denoising process
- Learning to denoise
- **Conclusion**

Conclusion

- DDPM (Vanilla Diffusion Model)
  - Surprisingly simple idea!
  - Forward process: Gradually add noise to clean data using a closed-form expression -- this step is fast and analytically tractable.
  - Reverse process: Train a network to denoise the data step-by-step, learning the reverse of the noising process.
  - Generation: Start from pure Gaussian noise and iteratively apply the denoising network to generate new samples.
  - The process is stochastic, so different outputs can be produced from the same initial noise due to the inherent randomness in the model.
  - Most of the math (statistics and algebra) is focused on deriving the correct scaling relationships between the clean image, the denoised prediction, and the added noise.

Conclusion

- Generative models trained on massive datasets are also known as foundation models.
- They have revolutionized text and image processing, powering breakthrough tools like ChatGPT and Stable Diffusion, and driving the rapid growth of the Generative AI (GenAI) industry.
- As these models continue to evolve, especially in their reasoning capabilities, they are becoming increasingly powerful tools -- that will assist us across a wide spectrum of tasks and domains.

Questions?