# DENOISING DIFFUSION PROBABILISTIC MODELS

NGUYEN VAN MINH, University of Information Technology, Vietnam
NGUYEN VAN HONG THAI, University of Information Technology, Vietnam
NGUYEN HUY PHUOC, University of Information Technology, Vietnam

## 1 Giới thiệu

Trong vật lý diffusion là hiện tượng khuếch tán của các phân tử (hoặc ion, năng lượng...) từ vùng có mật độ cao hơn sang vùng có mật độ thấp hơn.
Mô hình khuếch tán là lớp mô hình sinh trong học máy, được thiết kế để tạo ra dữ liệu mới bằng cách mô phỏng quá trình nhiễu hóa và khử nhiễu dữ liệu, được sáng tạo dựa trên nguyên lí nhiệt động lực học.

### 1.1 Ý tưởng tổng quan

Ý tưởng của phương pháp này là biến đổi phân bố dữ liệu thành một phân bố có thể lấy mẫu được. Việc sinh dữ liệu sẽ bắt đầu từ phân bố này, sau đó biến đổi ngược về phân bố ban đầu là 2 quá trình chính để khởi tạo và huấn luyện mô hình. Mô hình cần học được phép biến đổi ngược. Quá trình biến đổi này được mô tả bằng một chuỗi các phân bố, chúng ta sẽ sử dụng quá trình ngẫu nhiên để mô tả chuỗi này.
Hai quá trình cơ bản của Diffusion model được cụ thể hóa như dưới đây:

- Quá trình khuếch tán thuận. Quá trình này chuyển phân phối phân phối dữ liệu phức tạp sang một phân phối đơn giản và có thể dễ dàng làm việc.
- Quá trình đảo ngược làm ngược lại quá trình trên, chuyển hóa mẫu phân phối đơn giản về lại với ảnh thực được sử dụng để sinh dữ liệu. Chúng ta sử dụng mạng neural để học cách thực hiện quá trình đảo ngược.

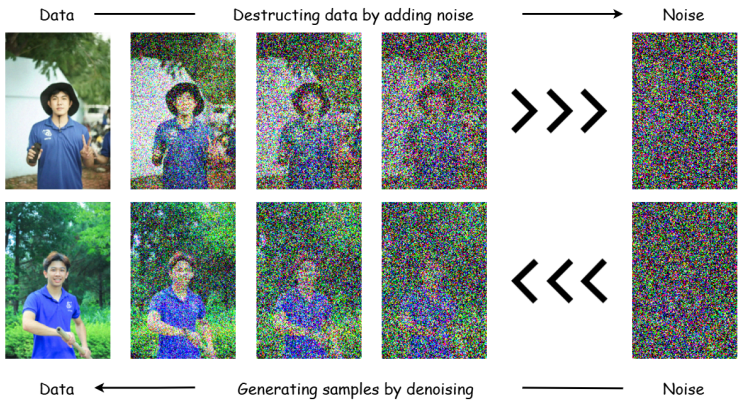


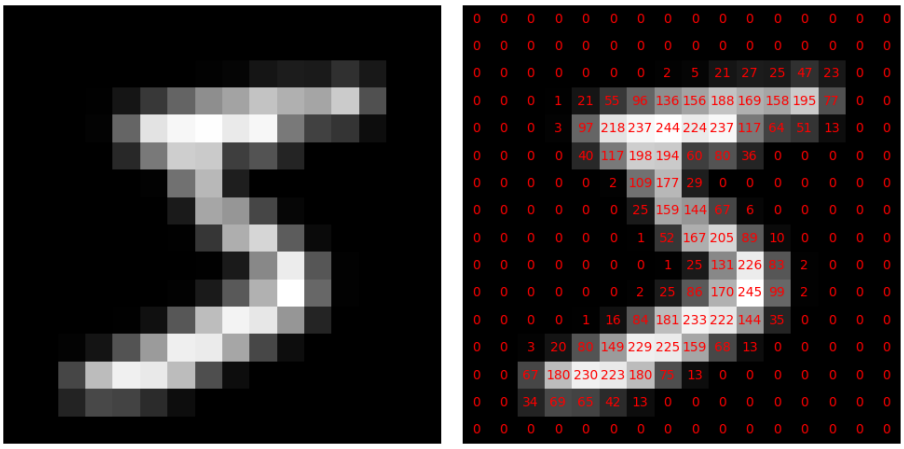

Fig. 1. Ví dụ

Authors' Contact Information: Nguyen Van Minh, 23520945@gm.uit.edu.vn, University of Information Technology, Thu Duc, Ho Chi Minh City, Vietnam; Nguyen Van Hong Thai, 23521418@gm.uit.edu.vn, University of Information Technology, Thu Duc, Ho Chi Minh City, Vietnam; Nguyen Huy Phuoc, 23521234@gm.uit.edu.vn, University of Information Technology, Thu Duc, Ho Chi Minh City, Vietnam.

# 2 Denoising Diffusion Probabilistic Models (DDPM)

## 2.1 Pixel space

- Là không gian lưu trữ dữ liệu, ta chuyển đổi hình ảnh thành ma trận với mỗi phần tử biểu thị giá trị đen trắng hoặc giá trị màu tại một pixel cụ thể. đây cũng sẽ là không gian dữ liệu mà mô hình sẽ làm việc.




- Tuy nhiên, các giá trị pixel cần được chuẩn hóa [0,1] trước khi đưa vào mô hình, nhằm tránh các vấn đề Gradient Exploding/Vanishing khi làm việc với khoảng [0,255] và giúp giảm phương sai dữ liệu, cải thiện tốc độ hội tụ của thuật toán tối ưu

## 2.2 Các định nghĩa toán liên quan

2.2.1 Chuỗi Markov. Đây là một chuỗi biến đổi toán học mô tả một quá trình ngẫu nhiên mà trạng thái sau chỉ phụ thuộc trạng thái hiện tại và không phụ thuộc vào các trạng thái trước đó:

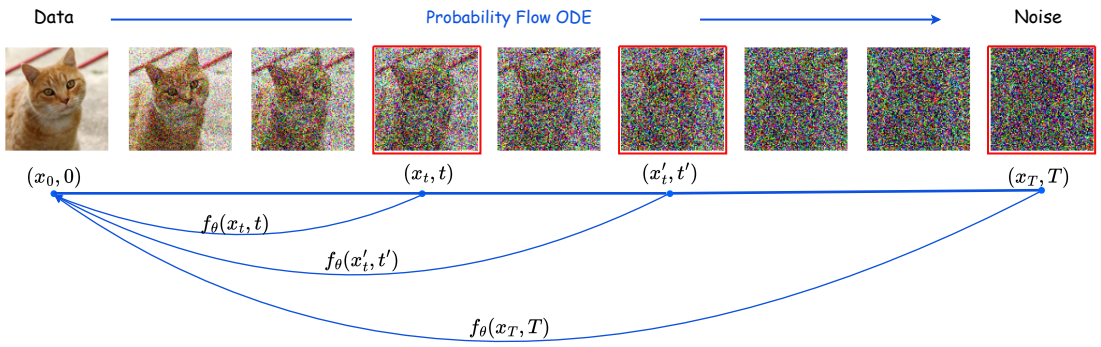
$$P(X_{n+m} = i \mid X_1, \ldots, X_n) = P(X_{n+m} = i \mid X_n).$$

2.2.2 Nhân biến đổi. Đây là một hàm mô tả xác suất chuyển từ trạng thái này sang trạng thái khác trong một quá trình ngẫu nhiên. Trong chuỗi Markov, nhân biến đổi chính là ma trận chuyển tiếp hoặc phân phối có điều kiện.

$$K(x_{t-1}, x_t) = P(X_t = x_t \mid X_{t-1} = x_{t-1})$$

## 2.3 Quá trình khuếch tán thuận

2.3.1 Công thức. Đây là một quá trình "encoder"của mô hình. Quá trình thuận xuất phát từ phân phối của dữ liệu $q(x_0)$ và chuyển đổi dần dần thành phân phối có thể dễ dàng lấy mẫu $q(x_T) \approx \mathcal{N}(x_T; 0, \mathbf{I})$. Ở mỗi quá trình biến đổi điểm dữ liệu, ta sẽ thêm dần dần một lượng nhỏ nhiễu $\epsilon$, nhiễu này sẽ được lấy mẫu dựa vào phân phối gauss (tức là $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$).
Ta có thể hiểu: $x_t = x_{t-1} + \epsilon_{t-1}$,



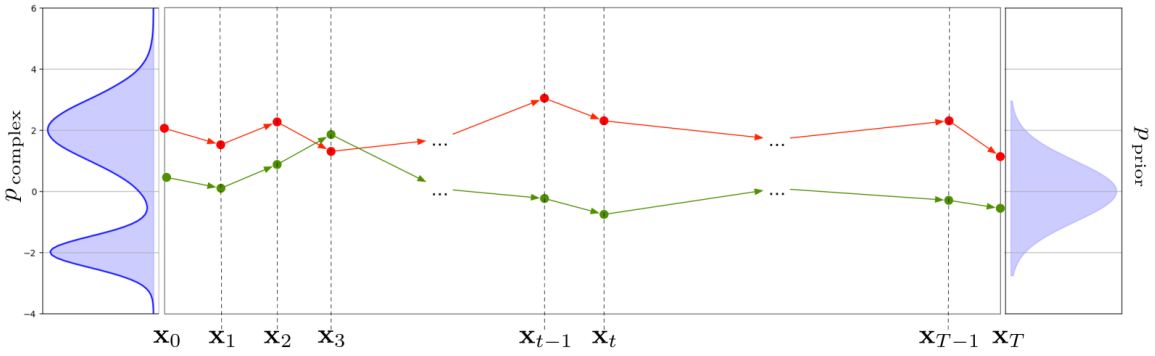

- $\epsilon$ là lượng nhiễu được thêm vào.
- $x_0$ là dữ liệu; $x_1$, $x_2$, ..., $x_T$ là các biến ẩn có cùng số chiều với $x_0$. $T$ là số bước biến đổi.

Tuy nhiên, nếu sử dụng công thức trên thì khi giá trị nhiễu quá lớn quá trình làm nhiễu quá nhanh sẽ khiến cho mô hình không học được gì từ đó. Bởi lẽ vậy ta cần kiểm soát độ lớn nhiễu thông qua giá trị $Beta_t$ (tốc độ khếch tán) và đồng thời giảm bớt sự ảnh hưởng của dữ liệu gốc. Ta có công thức:

$$x_t = \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t}\epsilon_t \text{ với } \epsilon_t \sim \mathcal{N}(0, \mathbf{I})$$

Quá trình khuếch tán trên là mô tả bằng một chuỗi Markov, nghĩa là trạng thái $x_t$ chỉ phụ thuộc vào $x_{t-1}$. Và từ đó, ta ký hiệu $q$ là hàm mật độ của quá trình khuếch tán, ta có:

$$q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)\ \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t\,\mathbf{I}\right)$$



2.3.2 Khó khăn. Ta có một điều cần đảm bảo , giá trị $T$ cần phải rất lớn và kéo theo tốc độ khếch tán phải đủ nhỏ. $T$ phải đủ lớn là vì:

2.3.3 Nhiễu hóa dần dần . Quá trình thuận thêm nhiễu Gaussian tại mỗi bước:

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

với $\beta_t$ là lượng nhiễu rất nhỏ. Khi $T$ lớn:

- Nhiễu được thêm dần dần, đảm bảo thông tin dữ liệu $x_0$ bị phá hủy từ từ.
- Nếu $T$ nhỏ, nhiễu $\beta_t$ phải lớn hơn, dẫn đến mất thông tin nhanh chóng và gây khó khăn trong quá trình khử nhiễu.

2.3.4 Đảm bảo phân phối $x_T$ gần $\mathcal{N}(0,\mathbf{I})$. Mục tiêu của quá trình thuận là biến $x_0$ thành nhiễu trắng $x_T \sim \mathcal{N}(0,\mathbf{I})$. Với $T$ lớn:

- Nhiễu được tích lũy dần dần, đảm bảo $x_T$ tiệm cận phân phối Gaussian chuẩn.
- Nếu $T$ nhỏ, $x_T$ có thể chưa đủ nhiễu hóa, làm giảm hiệu quả quá trình ngược.

2.3.5 Chất lượng tái tạo dữ liệu. Số bước $T$ lớn cho phép khử nhiễu dần dần, cải thiện chất lượng dữ liệu tái tạo (như hình ảnh hoặc âm thanh). Với $T$ nhỏ, dữ liệu tái tạo dễ bị mờ hoặc thiếu chi tiết.

2.3.6 Tổng quát hóa công thức. Từ đây ta có một vấn đề, nhìn vào công thức trình bày ở trên ta cần phải tính một cách tuần tự $x_1, x_2, ..., x_t$ mà vì $t$ lớn nên việc tính toán rất mất thời gian, vì vậy ta có một đề xuất tổng quát hóa công thức trên tức là ta sẽ tính $x_t$ từ $x_0$.

2.3.7 Biểu diễn tại từng bước. Ở mỗi bước, $x_t$ được sinh ra từ $x_{t-1}$:

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

2.3.8 Biểu diễn cho $t$.

$$\begin{aligned} & x_t = \sqrt{\beta_t x_{t-1}} + \sqrt{1-\beta_t}\epsilon_{t-1} \\ \implies & x_t = \sqrt{\beta_t}\left(\sqrt{\beta_{t-1}x_{t-2}} + \sqrt{1-\beta_{t-1}}\epsilon_{t-2}\right) + \sqrt{1-\beta_t}\epsilon_{t-1} \quad \text{(mở rộng } x_{t-1}\text{)} \end{aligned}$$

Khai triển: $x_t = \sqrt{\beta_{t-1}\beta_t}x_{t-2} + \underbrace{\sqrt{\beta_t(1-\beta_{t-1})}\epsilon_{t-2}}_{\text{RV1}} + \underbrace{\sqrt{1-\beta_t}\epsilon_{t-1}}_{\text{RV2}}$

2.3.9 Tổng quát hóa cho $t$. Hai số hạng cụ thể là RV1 và RV2 ở trên là hai biến ngẫu nhiên phân phối chuẩn với trung bình bằng không và phương sai $\beta_t(1-\beta_t)$, và $(1-\beta)$ tương ứng.

Mà ta có 2 tính chất, với hai biến ngẫu nhiên độc lập $X$ và $Y$:

- $E[X+Y] = E[X] + E[Y]$
- $VAR[X+Y] = VAR[X] + VAR[Y]$

Áp dụng tính chất lên $RV1$ và $RV2$, ta có:

$$\Rightarrow x_t = \sqrt{\beta_t \beta_{t-1}} x_{t-2} + \underbrace{\sqrt{\beta_t(1-\beta_{t-1})}\epsilon_{t-2}}_{\text{RV1}} + \underbrace{\sqrt{1-\beta_t}\epsilon_{t-1}}_{\text{RV2}}$$

$$= \sqrt{\beta_t \beta_{t-1}} x_{t-2} + \sqrt{1-\beta_t\beta_{t-1}}\bar{z}_{t-2} \quad (\bar{z}_{t-2}\text{là biến sau khi hợp nhất})$$

$$= \cdots$$

$$\Rightarrow x_t = \sqrt{\bar{\beta}_t} x_0 + \sqrt{1-\bar{\beta}_t}\epsilon \quad (\text{vì } \bar{\beta}_t = \prod_{i=1}^{T} \beta_i)$$

Từ đây, ta có rút ra công thức tính nhanh $x_t$ từ $x_0$ như sau:
$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\beta}_t}x_0, (1-\bar{\beta}_t)\mathbf{I})$

## 2.4 Quá trình đảo ngược

2.4.1 Tổng quan. Quá trình đảo ngược cũng là một chuỗi Markov có những trạng thái như quá trình thuận nhưng theo chiều ngược lại. Phân phối của quá trình sinh gồm T bước áp dụng nhân biến đổi (tương tự quá trình thuận):

$$p(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p(x_{t-1} \mid x_t)$$

Với $p(x_T) = \mathcal{N}(x_T; 0, I)$, $x_t$ là điểm cuối cùng của quá trình thuận hoặc điểm được sinh ngẫu nhiên ban đầu.

Để đảo ngược, ta cần xác định nhân biến đổi ngược $p(x_{t-1} \mid x_t)$ và biến đổi T bước để thu được $x_0$. Điểm dữ liệu tại bước $t$ tuân theo phân phối Gaussian tuy nhiên, dữ liệu gốc hoặc dữ liệu sinh thì không tuân theo phân phối gaussian.

Vì thế ý tưởng là ta sẽ cố gắng sấp xỉ $p(x_{t-1} \mid x_t)$, tức là chúng ta sẽ định nghĩa một phân phối mới $q(x_{t-1} \mid x_t)$. Với $\beta_t$ nhỏ thì quá trình thuận và quá trình đảo ngược sẽ có cùng functional form.

Còn mean và covariance của phân phối này thì chúng ta có thể sử dụng mạng neural để ước lượng. Ta có thể viết nhân biến đổi ngược dưới dạng tổng quát nhất như sau:

$$p(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Chi phí tính toán của Diffusion Model chủ yếu đến từ chi phí tính toán của hai mô hình $\mu_\theta(x_t, t)$ và $\Sigma_\theta(x_t, t)$.

2.4.2 Chi tiết.

2.4.3 Khởi tạo tại bước $T$. Ở bước đầu tiên, trạng thái khởi tạo là từ một phân phối Gaussian chuẩn:

$$x^{(T)} \sim \mathcal{N}(0, I)$$

Đây là trạng thái hoàn toàn ngẫu nhiên, tương ứng với dữ liệu đã được thêm nhiễu hoàn toàn qua quá trình thuận (forward process).

2.4.4 Biến đổi tại mỗi bước đảo ngược. Quá trình từ $x^{(T)}$ về $x^{(0)}$ được thực hiện qua $T$ bước ngược lại, mỗi bước tuân theo quy tắc:

$$p(x^{(t-1)} \mid x^{(t)}) = \mathcal{N}(x^{(t-1)}; \mu_\theta(x^{(t)}, t), \Sigma_\theta(x^{(t)}, t))$$

Ước lượng mean ($\mu_\theta$). Hàm mean $\mu_\theta(x^{(t)}, t)$ được định nghĩa như sau:

$$\mu_\theta(x^{(t)}, t) = \frac{1}{\sqrt{\alpha_t}} \left( x^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x^{(t)}, t) \right)$$

Trong đó:

- $\epsilon_\theta(x^{(t)}, t)$: Hàm mạng neural ước lượng nhiễu được thêm vào tại bước $t$.
- $\beta_t$: Lượng nhiễu tại bước $t$ (noise schedule).
- $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

Xử lý nhiễu ngẫu nhiên. Tại mỗi bước, mẫu ngẫu nhiên được rút ra từ phân phối Gaussian:

$$x^{(t-1)} \sim \mathcal{N}(\mu_\theta(x^{(t)}, t), \Sigma_\theta(x^{(t)}, t))$$

Trong đó $\Sigma_\theta(x^{(t)}, t)$ thường được giả định là một hàm đơn giản của $\beta_t$, ví dụ: $\Sigma_\theta(x^{(t)}, t) = \beta_t I$.

2.4.5 Đặc điểm của từng bước.

- Tại bước đầu ($t = T$): Nhiễu thống trị, trạng thái hoàn toàn ngẫu nhiên ($x^{(T)} \sim \mathcal{N}(0, I)$).
- Bước trung gian ($1 < t < T$): Quá trình kết hợp thông tin nhiễu ($\epsilon_\theta$) và trạng thái hiện tại để tái tạo trạng thái trước đó.
- Bước cuối cùng ($t = 1$): Loại bỏ hoàn toàn nhiễu, khôi phục trạng thái $x^{(0)}$.

## 2.5 Hàm mục tiêu

Hàm mất mát được tính theo khoảng cách trung bình trong không gian Euclide đối với phân phối được dự đoán ra và phân phối gốc trong quá trình nhiễu hóa và khử nhiễu.

Toàn bộ quá trình đảo ngược tối ưu hóa theo hàm log-likelihood của dữ liệu gốc $x^{(0)}$:

$$L = \mathbb{E}_{x^{(0)}, \epsilon \sim \mathcal{N}(0,I), t} \left[ \|\epsilon - \epsilon_\theta(x^{(t)}, t)\|^2 \right]$$

Hàm mất mát này tối ưu mạng $\epsilon_\theta$, giúp mô hình học cách dự đoán nhiễu được thêm vào dữ liệu.

Sự biến đổi trong quá trình nhiễu hóa và khử nhiễu của model DDPM cũng tuần tự theo các bước như model VAEs nên chúng ta có ý tưởng hàm loss của DDPM dựa trên VAEs.

Hàm mất mát này tối ưu mạng $\epsilon_\theta$, giúp mô hình học cách dự đoán nhiễu được thêm vào dữ liệu.

Bình thường theo hàm loss của VAEs ta có là:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \; - \; D_{KL}(q_\phi(z|x) \parallel p_\theta(z))$$

Trong đó,

- Ta có $p_\theta(x_0)$ phân phối ngược mà chúng ta đang mô hình hóa. $\theta$ biểu diễn các tham số mà mô hình phải tối ưu hóa để thực hiện phân phối ngược
- $q_\phi(z|x)$ là phân phối xấp xỉ, được mô hình hóa bởi mạng suy diễn của VAE. $\phi$ là các tham số của mạng suy diễn, thường được tối ưu hóa trong quá trình huấn luyện.
- $D_{\mathrm{KL}}$ là phân kỳ Kullback-Leibler để đo lường sự khác biệt giữa hai phân phối.

## 2.6 Tối ưu hàm Negative Log Likelihood trong DDPM

2.6.1 Hàm loss. Ta không thể tính toán hàm loss trên trực tiếp được vì không tính được hai phân phối thành phần trên. Do đó, ta cần biến đổi $L_{\mathrm{VLB}}$ như sau để đưa về tính cách khoảng KL Divergence giữa các phân phối Gaussian (đó là lý do mà chúng ta phải thiết kế quá trình thuận thật cẩn thận để mọi thứ đều là Gaussian):

2.6.2 Tối ưu hàm loss. Toàn bộ quá trình đảo ngược tối ưu hóa theo hàm log-likelihood của dữ liệu gốc $x^{(0)}$:

$$L = \mathbb{E}_{x^{(0)},\epsilon\sim\mathcal{N}(0,I),t}\left[\|\epsilon - \epsilon_\theta(x^{(t)}, t)\|^2\right]$$

Hàm mất mát này tối ưu mạng $\epsilon_\theta$, giúp mô hình học cách dự đoán nhiễu được thêm vào dữ liệu.

Bình thường theo hàm loss của VAEs ta có là

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \; - \; D_{KL}(q_\phi(z|x) \parallel p_\theta(z))$$

$$\begin{aligned}
\log p_\theta(x_0) &\geq \mathbb{E}_{q(x_{1:T}|x_0)}\big[\log p_\theta(x_0|x_{1:T})\big] - D_{\mathrm{KL}}\big(q(x_{1:T}|x_0)\|p_\theta(x_{1:T})\big) \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\big[\log p_\theta(x_0|x_{1:T})\big] - \mathbb{E}_{q(x_{1:T}|x_0)}\big[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T})}\big] \\
&= \mathbb{E}_q\big[\log p_\theta(x_0|x_{1:T})\big] - \mathbb{E}_q\big[\log q(x_{1:T}|x_0) - \log p_\theta(x_{1:T})\big] \\
&= \mathbb{E}_q\big[\log p_\theta(x_0|x_{1:T}) - \log q(x_{1:T}|x_0) + \log p_\theta(x_{1:T})\big] \\
&= \mathbb{E}_q\big[\log p_\theta(x_0|x_{1:T}) + \log p_\theta(x_{1:T}) - \log q(x_{1:T}|x_0)\big] \\
&= \mathbb{E}_q\big[\log \frac{p_\theta(x_0|x_{1:T})p_\theta(x_{1:T})}{q(x_{1:T}|x_0)}\big] \\
&= \mathbb{E}_q\big[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\big].
\end{aligned}$$

Việc huấn luyện được thực hiện bằng cách tối ưu chặn trên của Negative Log Likelihood:

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

Trong đó,

- Ta có $p_\theta(x_0)$ phân phối ngược mà chúng ta đang mô hình hóa. $\theta$ biểu diễn các tham số mà mô hình phải tối ưu hóa để thực hiện phân phối ngược
- $q$ là phân phối tiến (forward process), mô tả cách biến $x_0$ được "nhiễu hóa"qua các bước.

Ở đây chúng ta áp dụng định nghĩa chuẩn về kỳ vọng, KL-Divergence và logarit vào phương trình ban đầu ta được như sau:

$$L_{\text{VLB}} = \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right]$$
$$= \mathbb{E}_q \left[ \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)} \right]$$
$$= \mathbb{E}_q \left[ -\log p_\theta(x_T) + \sum_{t=1}^{T} \log \frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)} \right]$$

Để biến đổi tiếp thì chúng ta nhắc lại quy tắc Bayes:

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

$$
\begin{aligned}
L_{\text{VLB}} &= \mathbb{E}_{q(x_{1:T}|x_0)}\left[-\log p_\theta(x_T) + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t)}{p_\theta(x_{t-1}|x_t)} + \log\frac{q(x_0|x_1)}{p_\theta(x_0|x_1)}\right] \\
&= \mathbb{E}_q\left[-\log p(x_T) + \log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \sum_{t=2}^{T}\log\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}\right] \\
&= \mathbb{E}_q\left[-\log p(x_T) + \log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)} + \log\frac{q(x_T|x_0)}{q(x_1|x_0)} + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}\right] \\
&= \mathbb{E}_q\left[-\log p(x_T) - \log p_\theta(x_0|x_1) + \log q(x_1|x_0) - \log q(x_1|x_0) + \log q(x_T|x_0) + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[-\log\frac{p(x_T)}{q(x_T|x_0)} - \log p_\theta(x_0|x_1) + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[-\log p_\theta(x_0|x_1)\right] - \mathbb{E}_{q(x_T|x_0)}\left[\log\frac{p(x_T)}{q(x_T|x_0)}\right] - \sum_{t=2}^{T}\mathbb{E}_{q(x_t,x_{t-1}|x_0)}\left[\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[-\log p_\theta(x_0|x_1)\right] + D_{\text{KL}}\left(q(x_T|x_0)\,\|\,p(x_T)\right) + \sum_{t=2}^{T}\mathbb{E}_{q(x_t|x_0)}\left[D_{\text{KL}}\left(q(x_{t-1}|x_t,x_0)\,\|\,p_\theta(x_{t-1}|x_t)\right)\right] \\
&= -L_0 + L_T + \sum_{t=2}^{T} L_{t-1}
\end{aligned}
$$

2.6.3 Thành phần $L_T$.

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

Xét 2 thành phần của KL divergence bên trên. Thành phần đầu tiên, $q(x_T|x_0)$ chỉ phụ thuộc vào quá trình thuận và không chứa tham số tối ưu được. Thành phần thứ hai, $p_\theta(x_T)$ được chọn trước là $\mathcal{N}(0, I)$. Do đó, $L_T$ là hằng số trong quá trình huấn luyện và có thể bỏ qua thành phần này.

2.6.4 Thành phần $L_0$. Nếu như các transition kernel $p(x_{t-1}|x_t)$ với $t > 1$ chiếu một không gian liên tục này sang không gian liên tục khác thì với $t = 1$ nó cần chiếu một không gian liên tục về một không gian rời rạc (không gian của input).

Do sự khác biệt đó, thành phần cuối cùng của quá trình đào ngược được tính theo cách riêng:

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

$$\delta_+^i(x) = \begin{cases}\infty & \text{if } x = 1\\ x + \frac{1}{255} & \text{if } x < 1\end{cases},\quad \delta_-^i(x) = \begin{cases}-\infty & \text{if } x = -1\\ x - \frac{1}{255} & \text{if } x > -1\end{cases}$$

Trong đó, $D$ là chiều của dữ liệu, $i$ là chỉ số để lấy ra số chiều. Giả sử mỗi thành phần màu của ảnh đầu vào được chia vào 256 bin. Công thức trên tính xác suất $p_\theta(x_0|x_1)$ rời vào bin đó. Xác suất này tính được sử dụng CDF của phân phối Gaussian.

Công thức trên chỉ áp dụng với dữ liệu ảnh đầu vào gồm những số integer $\{0, 1, \ldots, 255\}$ được scale tuyến tính về $[-1, 1]$.

### 2.7 Một số vấn đề mà DDPM gặp phải

- Quá trình suy diễn (khử nhiễu) diễn ra chậm, do rằng không như quá trình thuận có công thức tổng quát. Quá trình nghịch bắt buộc ta đi từng giai đoạn 1, mỗi giai đoạn rất tốn thời gian vì mỗi bước yêu cầu chạy qua toàn bộ mạng. Và như ta đã phân tích số bước $T$ cần đủ lớn để mô hình hoạt động tốt.
- Chất lượng mẫu, ảnh cấu thành có thể không đảm bảo các chi tiết cao, nhỏ có thể bị mờ. Điều này, là do quá trình khử nhiễu ngược trong DDPM dựa trên việc mô hình ước lượng nhiễu ở mỗi bước. Tuy nhiên, nhiễu trong mỗi bước thường được tạo ra từ một phân phối Gaussian, mà việc khử nhiễu từ phân phối này không thể tái tạo hoàn hảo các chi tiết đặc thù.
- Khả năng mở rộng của DDPM, rất kém hiệu quả nếu ta áp dụng trên các loại dữ liệu khác 2D (như ảnh) dữ liệu 3D có cấu trúc không gian phức tạp hơn so với dữ liệu 2D. Điều này có thể yêu cầu các mô hình học sâu phải hiểu rõ hơn về mối quan hệ không gian ba chiều giữa các điểm. Dữ liệu video có yếu tố thời gian và mối quan hệ giữa các khung hình. Điều này đòi hỏi mô hình phải hiểu và duy trì các phụ thuộc động trong quá trình tạo mẫu, điều này phức tạp hơn rất nhiều so với việc tạo mẫu từ ảnh tĩnh.
- Hạn chế trong điều kiện đầu vào và đầu ra, chỉ giải quyết được khi điều kiện là image-to-image còn các điều kiện khác (text-to-image,...) chưa thể thực hiện được.

## 3 Tiêu chí đánh giá

### 3.1 FID score

3.1.1 Mô tả. FID đo khoảng cách giữa phân phối của các đặc trưng từ hình ảnh sinh ra ($p_g$) và hình ảnh thật ($p_r$). FID sử dụng đặc trưng trích xuất từ một mạng tiền huấn luyện để biểu diễn hình ảnh trong không gian tiềm ẩn.
FID dựa trên sự so sánh hai phân phối Gaussian (phân phối của đặc trưng từ hình ảnh thật và hình ảnh sinh ra).

3.1.2 Công thức.

$$FID = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}})$$

Trong đó:

- $\mu_r, \Sigma_r$: Trung bình và ma trận hiệp phương sai của đặc trưng hình ảnh thật.
- $\mu_g, \Sigma_g$: Trung bình và ma trận hiệp phương sai của đặc trưng hình ảnh sinh ra.
- $\|\mu_r - \mu_g\|_2^2$: Khoảng cách Euclidean giữa hai vector trung bình.
- $\mathrm{Tr}(\cdot)$: Dấu vết (trace) của ma trận.

3.1.3 Các bước sử dụng.

(1) Sinh hình ảnh bằng mô hình cần đánh giá.
(2) Dùng mô hình Inception v3 để trích xuất đặc trưng của hình ảnh.

(3) Tính trung bình và ma trận hiệp phương sai của các đặc trưng cho cả hình ảnh thật và hình ảnh sinh ra.
(4) Tính FID dựa trên công thức trên.

### 3.2 Inception Score

3.2.1 Mô tả. Inception Score đánh giá chất lượng và sự đa dạng của hình ảnh sinh ra dựa trên dự đoán phân loại từ mạng Inception v3.

- Chất lượng: Hình ảnh sắc nét sẽ có phân phối dự đoán $p(y|x)$ tập trung vào một lớp cụ thể.
- Đa dạng: Hình ảnh sinh ra khác nhau sẽ tạo ra phân phối $p(y)$ (trung bình của các $p(y|x)$) đồng đều trên các lớp.

3.2.2 Công thức.

$$IS = \exp\left(\mathbb{E}_x\left[D_{KL}(p(y|x)\|p(y))\right]\right)$$

Trong đó:

- $p(y|x)$: Phân phối dự đoán xác suất nhãn $y$ của một hình ảnh $x$.
- $p(y) = \frac{1}{N}\sum_{i=1}^{N} p(y|x_i)$: Phân phối tổng hợp từ tất cả hình ảnh sinh ra.
- $D_{KL}(p(y|x)\|p(y))$: Khoảng cách Kullback-Leibler (KL) giữa $p(y|x)$ và $p(y)$.

3.2.3 Cách sử dụng.

(1) Dùng mô hình Inception v3 để trích xuất phân phối dự đoán $p(y|x)$ cho hình ảnh sinh ra.
(2) Tính phân phối trung bình $p(y)$.
(3) Tính Inception Score dựa trên công thức trên.

## 4 Thực nghiệm, kết quả, đánh giá

### 4.1 VAEs

VAEs được thiết kế để học cách biểu diễn ẩn (latent representation) của dữ liệu, kết hợp bộ tự mã hóa và phương pháp biến nhân. Hàm mục tiêu

Hàm mục tiêu của VAEs là Evidence Lower Bound (ELBO). Đây là một giới hạn dưới của log-likelihood của dữ liệu, chúng ta sẽ tối lưu hàm ELBO này theo hướng cực tiểu hóa hàm cận dưới.

Công thức được biểu diễn như sau

$$\text{ELBO} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - KL(z, N(0, I_d))$$

Hàm này có 2 thành phần chính

- Phần tái tạo dùng để reconstruct lại input ban đầu. Các hàm loss thông dụng là Mean Square Error hay Mean Absolute Error. Trong trường hợp ảnh nhị phân, ta có thể sử dụng binary cross entropy.

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$$

- Phần KL divergence sử dụng KL divergence (khoảng cách giữa 2 phân phối xác suất) giữa phân phối chuẩn với trung bình $\mathbb{E}(z)$ và phương sai $\mathbb{V}(z)$ với phân phối chuẩn chuẩn tắc $d$ chiều $N(0, I_d)$.

$$KL(z, N(0, I_d)) = \frac{1}{2} \sum_{i=1}^{d} \left( \mathbb{V}(z_i) - \log \mathbb{V}(z_i) - 1 + \mathbb{E}(z_i)^2 \right)$$

#### 4.1.1 Vấn đề mà VAEs gặp phải trong quá trình huấn luyện để tối ưu hóa hàm mục tiêu.

- Hiện tượng "Posterior Collapse": KL divergence $KL(z, N(0, I_d))$ có thể trở nên quá lớn, khiến mô hình ưu tiên tối ưu hóa phần này thay vì phần tái tạo.
- Nếu KL divergence quá lớn, mô hình sẽ ưu tiên làm cho phân phối ẩn gần với phân phối tiên nghiệm, bỏ qua việc tái tạo dữ liệu. Ngược lại, nếu reconstruction term quá lớn, mô hình có thể bỏ qua việc học một phân phối ẩn có cấu trúc tốt.
- Khoảng cách giữa ELBO và log-likelihood có thể lớn, mô hình sẽ không thể tối ưu hóa đúng với phân phối đúng.
- Khi $\mathbb{V}(z_i)$ tiến gần đến 0, $\log \mathbb{V}(z_i)$ tiến đến $-\infty$, gây ra các vấn đề số học.

#### 4.1.2 Điều mà DDDP khắc phục được của VAEs.

- Không cần giả định phân phối dữ liệu:
  - Diffusion Models không sử dụng latent space như VAEs, mà dựa trên quy trình nhiễu dần dần:
    - Quá trình Forward: Nhiễu dần dữ liệu $x_0$ qua nhiều bước để thu được phân phối Gaussian $x_T$.
    - Quá trình Reverse: Học cách tái tạo dữ liệu từ nhiễu, từng bước một.
- Không cần regularization bằng KL Divergence:
  - Không gian tiềm ẩn trong VAEs bị regularize bởi KL Divergence, dẫn đến mất thông tin. DDPM không yêu cầu bước này.

### 4.2 Kết quả với bộ dữ liệu MNIST

#### 4.2.1 DDPM.

Chạy thực nghiệm DDPM trên tập dữ liệu MNIST và tiến hành khử nhiễu từ nhiễu random và nhận được mẫu ảnh như dưới đây.

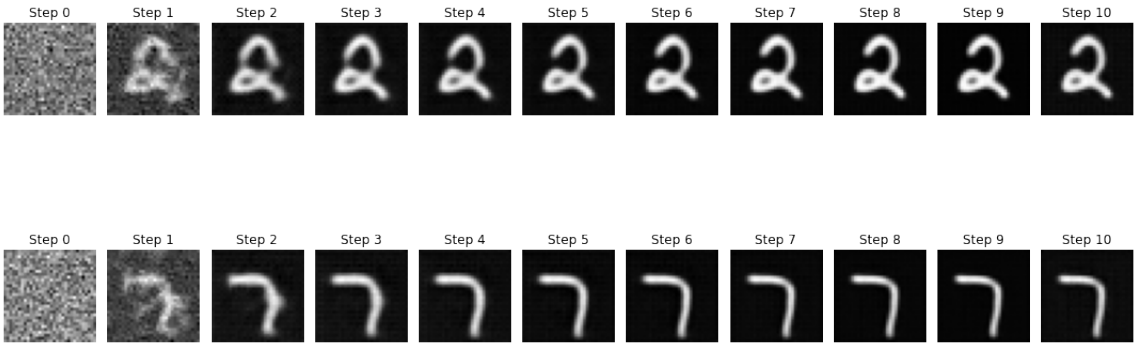



Fig. 2. 10 epochs, 1000 steps per epochs, 300ms/epochs

4.2.2 GAN. Chạy thực nghiệm GAN trên tập dữ liệu MNIST và tiến hành khử nhiễu từ nhiều random và nhận được mẫu ảnh như dưới đây.

Fig. 3. 30 epochs, 550 steps per epoch,60ms per steps

4.2.3 VAEs. Chạy trên tập dữ liệu MNIST sau đó tiến hành khử nhiều và thu được kết quả như dưới đây.

Fig. 4. 10 Epochs, 547 steps per epoch, 30ms per steps

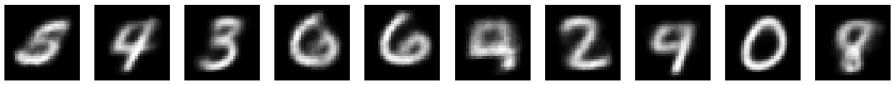


Fig. 5. 30 Epochs, 547 steps per epoch, 30ms per steps

Fig. 6. 70 Epochs, 547 steps per epoch, 50ms per steps

## 5 Trích dẫn

Nội dung của dự án được tham khảo từ các nguồn:

- Diffusion Models cơ bản[1];
- Sinh dữ liệu với mô hình diffusion và mô hình dạng SDE tổng quát[2]
- Alex Nichol, Prafulla Dhariwal, Improved Denoising Diffusion Probabilistic Models[3]
- A deep dive into DDPMs[4]
- Inga Strümke, Helge Langseth, Lecture Notes in Probabilistic Diffusion Models[5]
- Kevin E. Wu, Kevin K. Yang, Rianne van den Berg, Sarah Alamdari, James Y. Zou, Alex X. Lu Ava P. Amini, Protein structure generation via folding diffusion[6]

[1] https://viblo.asia/p/diffusion-models-co-ban-phan-1-E1XVOx884Mz#_ii-diffusion-trong-deep-learning-6
[2] https://viblo.asia/p/sinh-du-lieu-voi-mo-hinh-diffusion-va-mo-hinh-dang-sde-tong-quat-Ljy5V3jMKra#_cong-thuc-cac-phan-bo-trong-qua-trinh-thuan-cua-mo-hinh-diffusion-16
[3] https://arxiv.org/pdf/2102.09672
[4] https://magic-with-latents.github.io/latent/posts/ddpms/part3/
[5] https://arxiv.org/html/2312.10393v1/#bib.bibx3
[6] https://www.nature.com/articles/s41467-024-45051-2#Sec10