L9-2 Diffusion Model

Generative Models

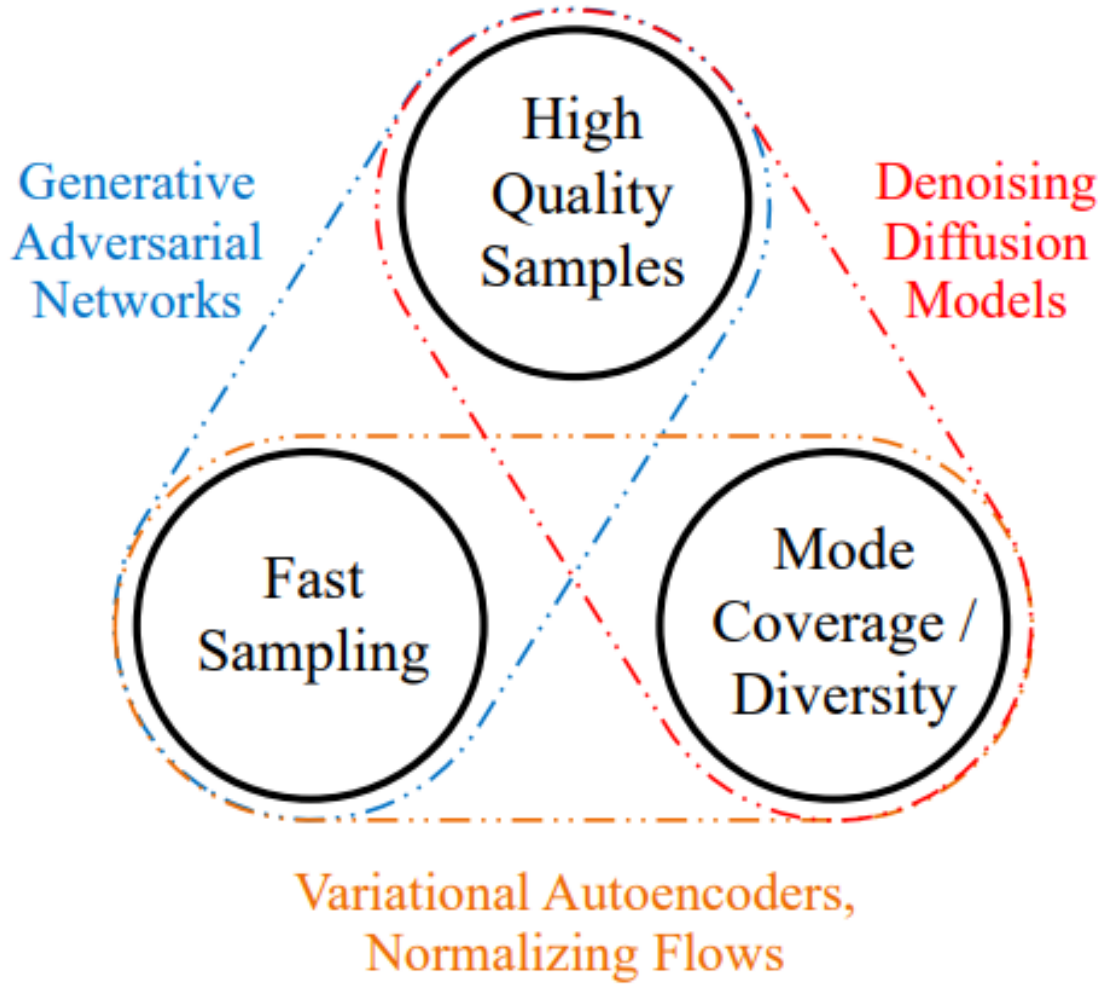
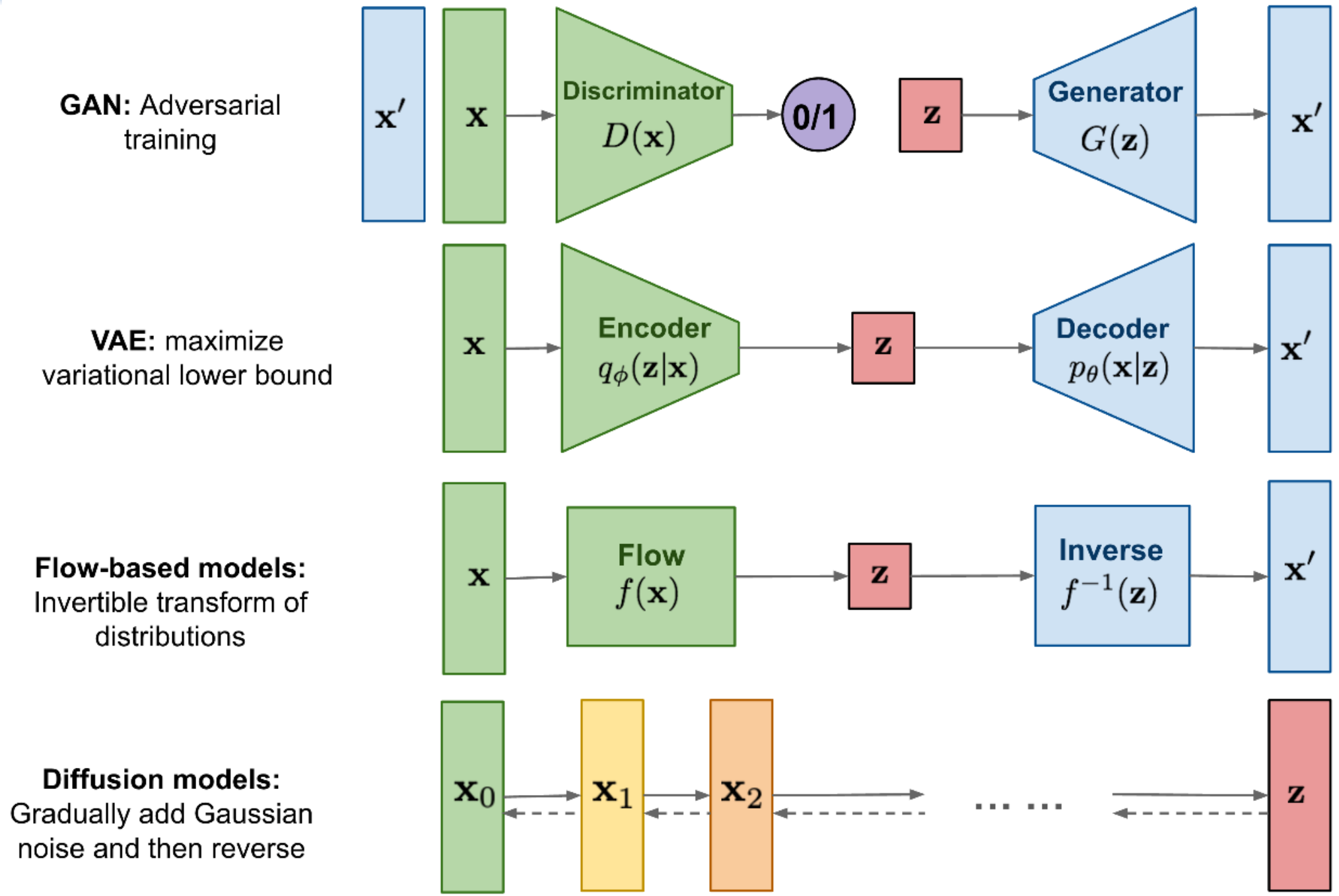


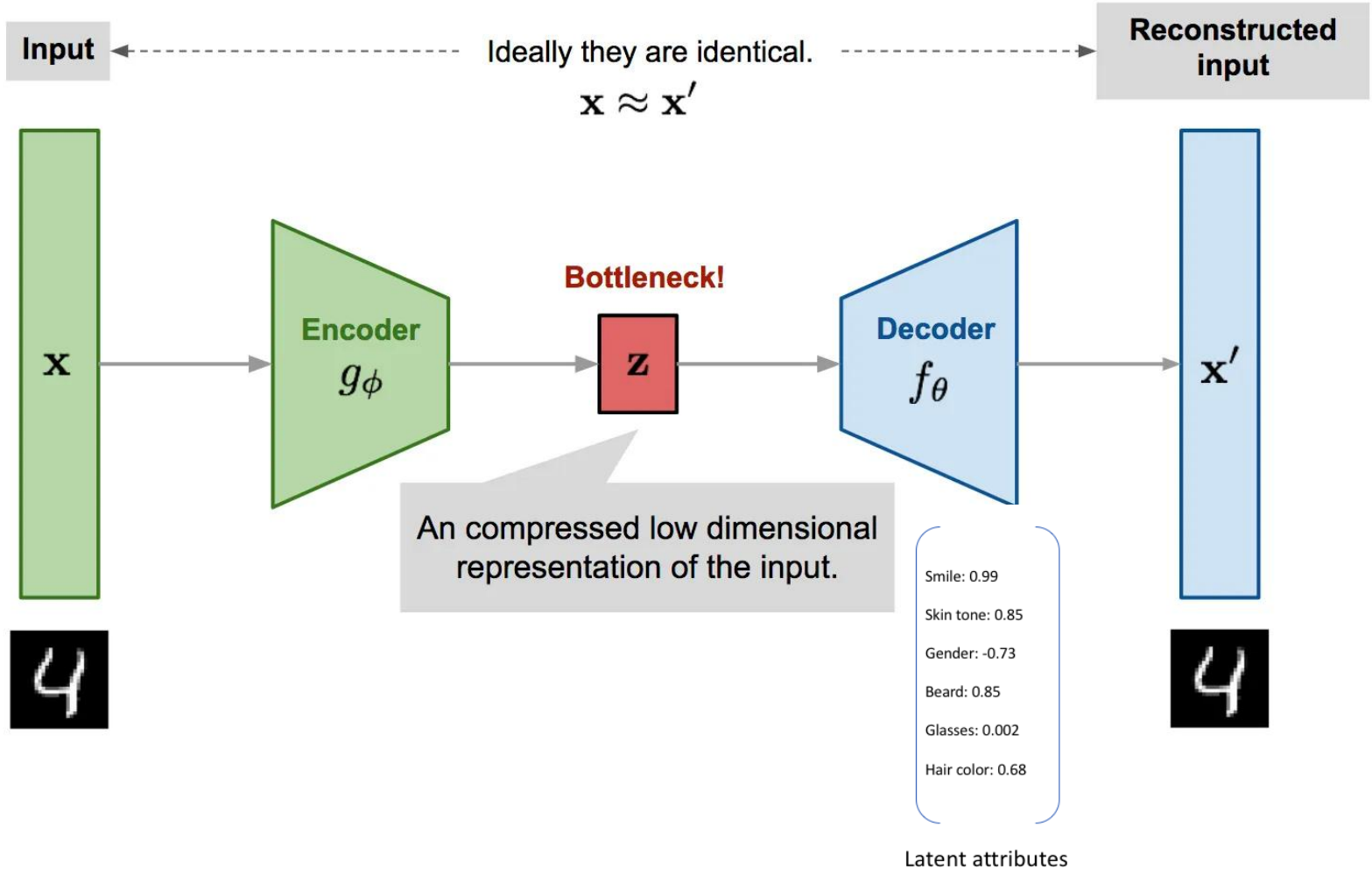
Figure 1: Generative learning trilemma.

https://arxiv.org/abs/2112.07804


GAN: Adversarial training
x′
x
Discriminator
D(x)
0/1
z
Generator
G(z)
x′
難訓練
VAE: maximize variational lower bound
x
Encoder
qϕ(z|x)
z
Decoder
pθ(x|z)
x′
效果模糊
Flow-based models: Invertible transform of distributions
x
Flow
f(x)
z
Inverse
f−1(z)
x′
Diffusion models: Gradually add Gaussian noise and then reverse
x0
x1
x2
z

VARIATIONAL AUTOENCODERS (VAE)

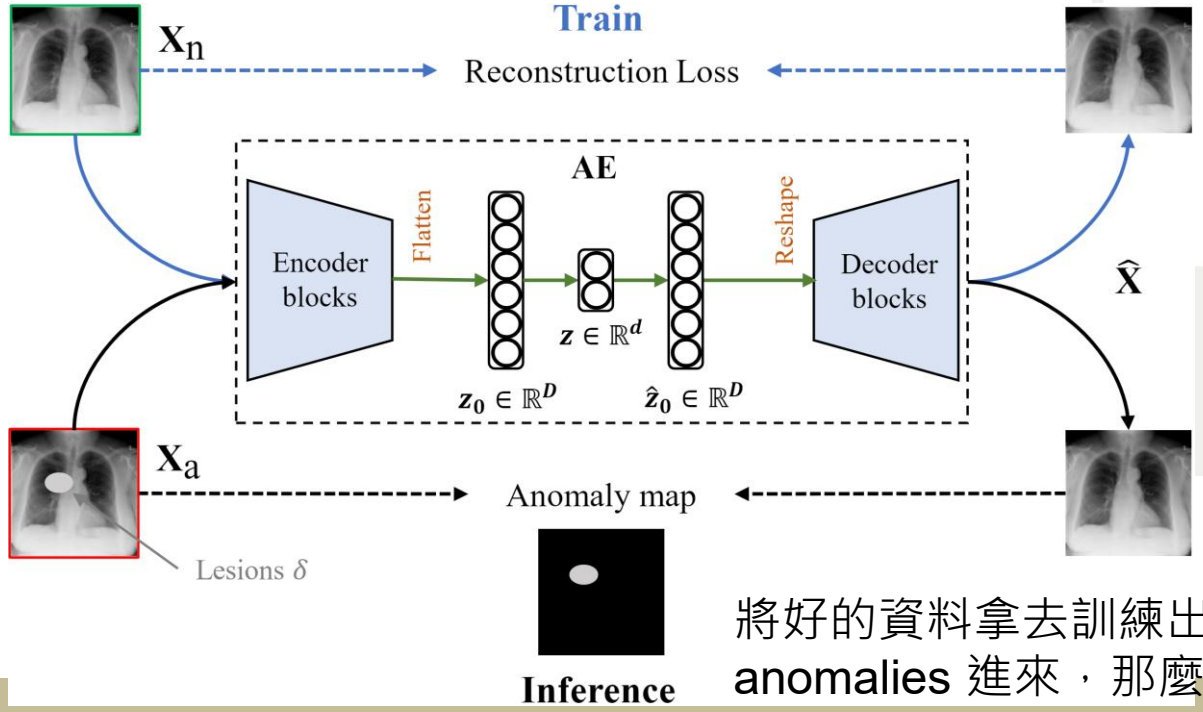
Variational AutoEncoder (VAE)



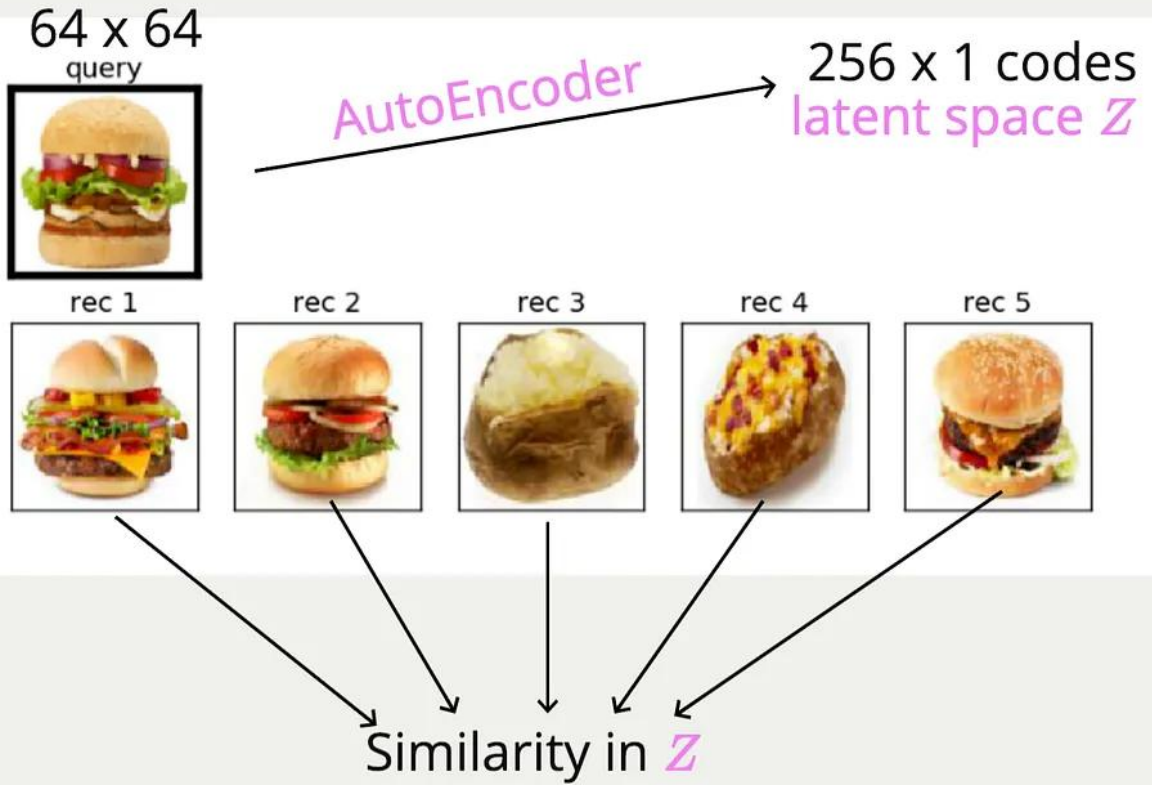
AutoEncoder是一種人工神經網路，主要用於無監督學習和特徵學習。它可以將輸入數據壓縮成較低維度的表示，然後再將其解壓縮成一個與原始數據相似的輸出。

Application of AutoEncoder

- Model pretrained weight
- Segmentation (Unet)
- Image to Caption
- Image retrieval



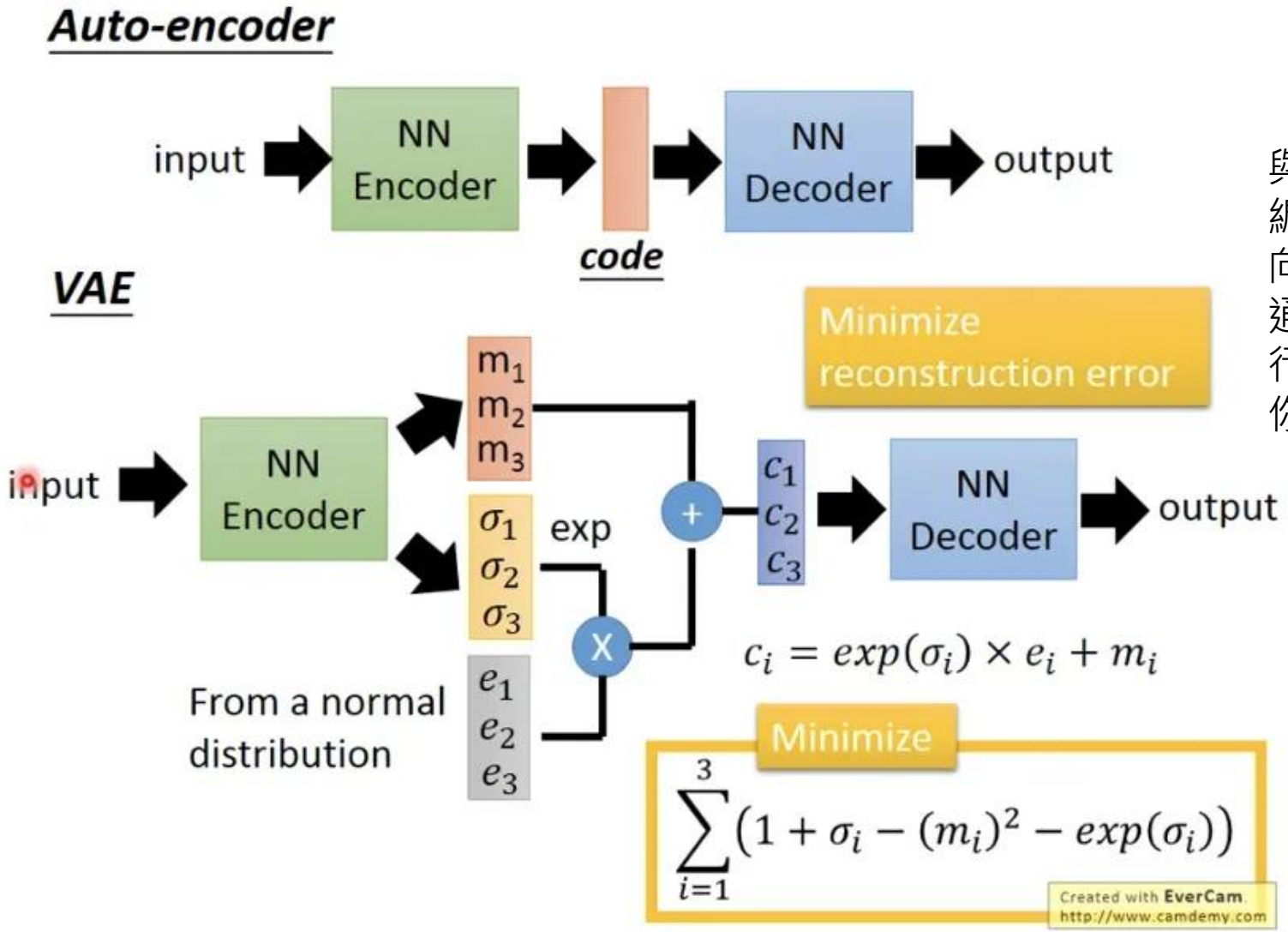
- Image Retrieval (unsupervised)



在latent space 上，計算距離

將好的資料拿去訓練出一個好的 Autoencoder，這時若有 anomalies 進來，那麼自然地 reconstruct 後的圖形就會壞掉。

Variational Autoencoder（VAE）

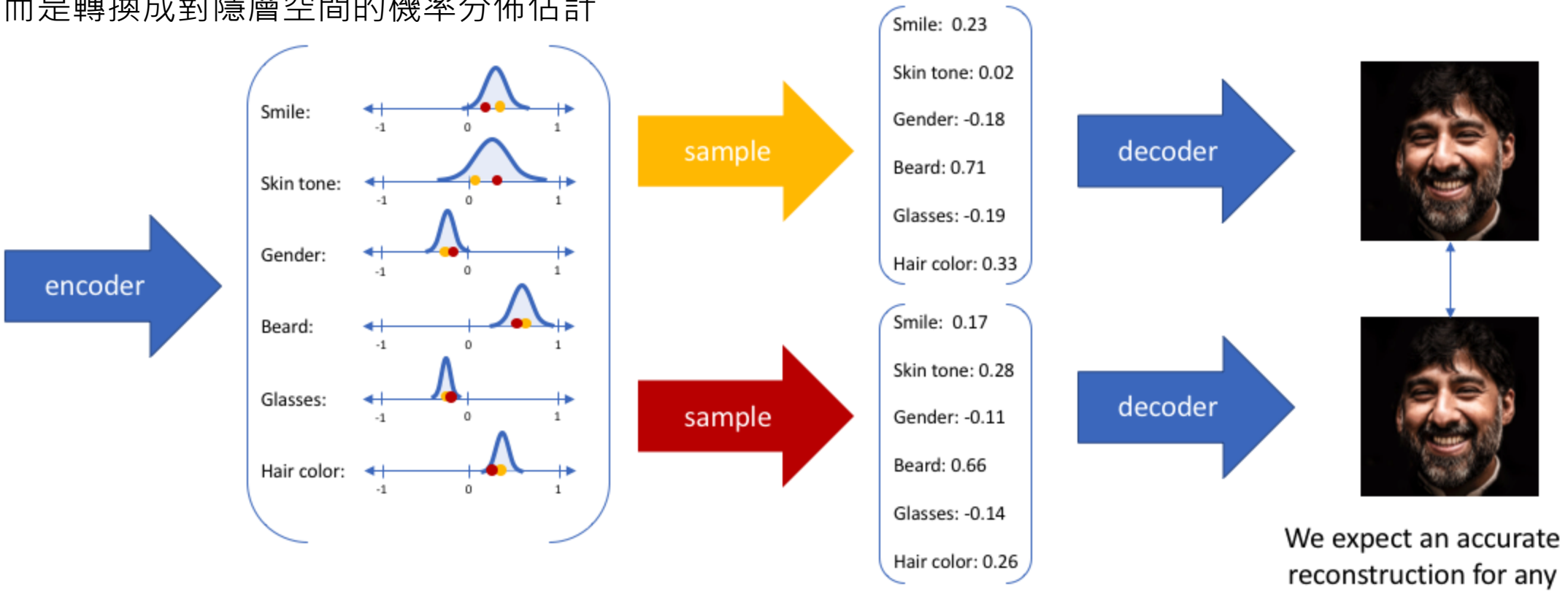


與 AutoEncoder 不同之處在於 VAE 在編碼過程增加了一些限制，迫使生成的向量遵從高斯分佈。由於高斯分佈可以通過其mean 和 standard deviation 進行參數化，因此 VAE 理論上是可以讓你控制要生成的圖片。

http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/GAN%20(v3).pdf

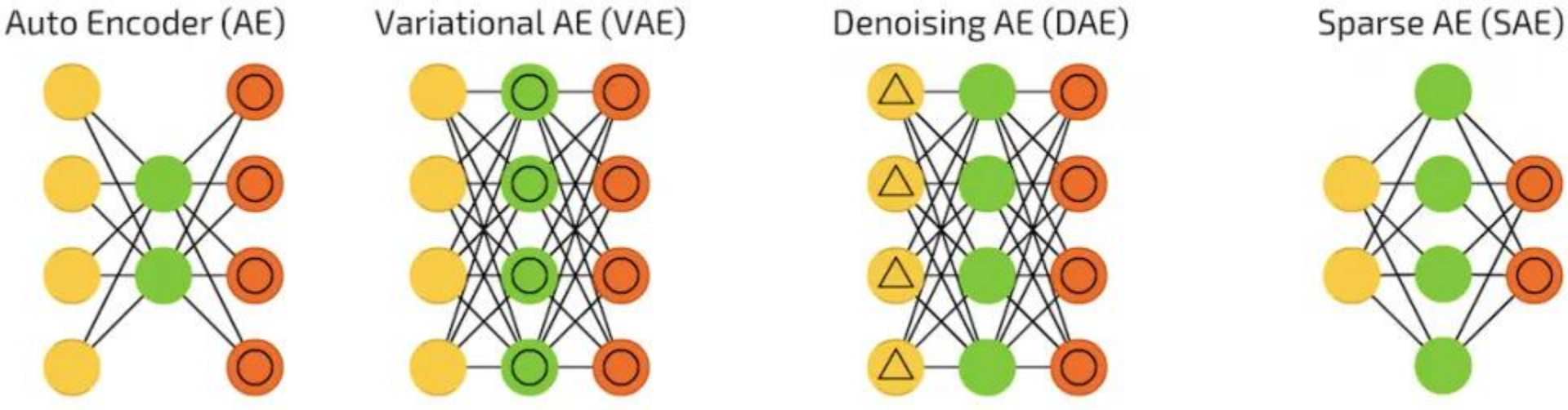
變分自編碼器（VAE）

變分自編碼器不再將輸入映射成隱層空間中的一個固定編碼，
而是轉換成對隱層空間的機率分佈估計



$$D_{KL}(q_\phi(z|x)\|p_\theta(z|x)) = log\, p_\theta(x) + D_{KL}(q_\phi(z|x)\|p(z)) - \mathbb{E}_{z\sim q_\phi(z|x)}\, log\, p_\theta(x|z)$$

AutoEncoder四大類型



DIFFUSION MODELS

https://youtu.be/EhndHhIvWWw?si=NyRLUix_swVSJcwm

Field anno 2020

- Autoregressive models
  - MADE, PixelRNN/CNN, Gated PixelCNN, PixelSNAIL
- Flow models
  - Autoregressive Flows, NICE, RealNVP, Glow, Flow++
- Latent Variable Models
  - VAE, IWAE, VQ-VAE
- GANs
  - GANs gave most realistic images, but have lots of bells and whistles + struggle with truly covering what's in data distribution
  - → Diffusion Models

2020 Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at `https://github.com/hojonathanho/diffusion`.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

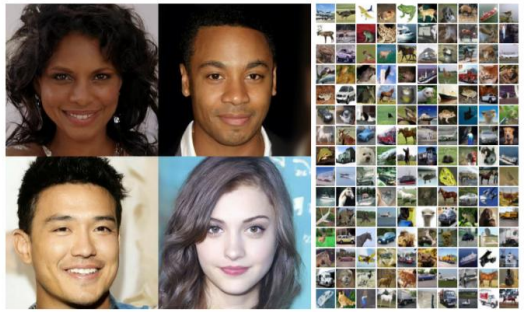


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

2020 Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

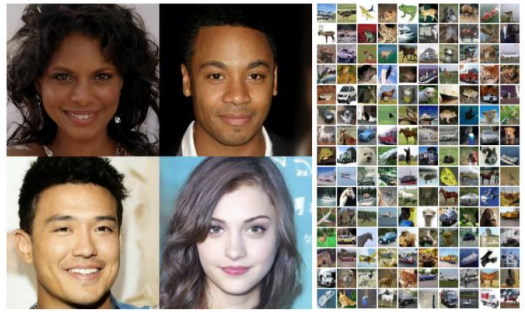


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Main contributions

- State-of-the-art in image generation
- Simpler training objective
- Connection to denoising score matching

Denoising Diffusion Probabilistic Models

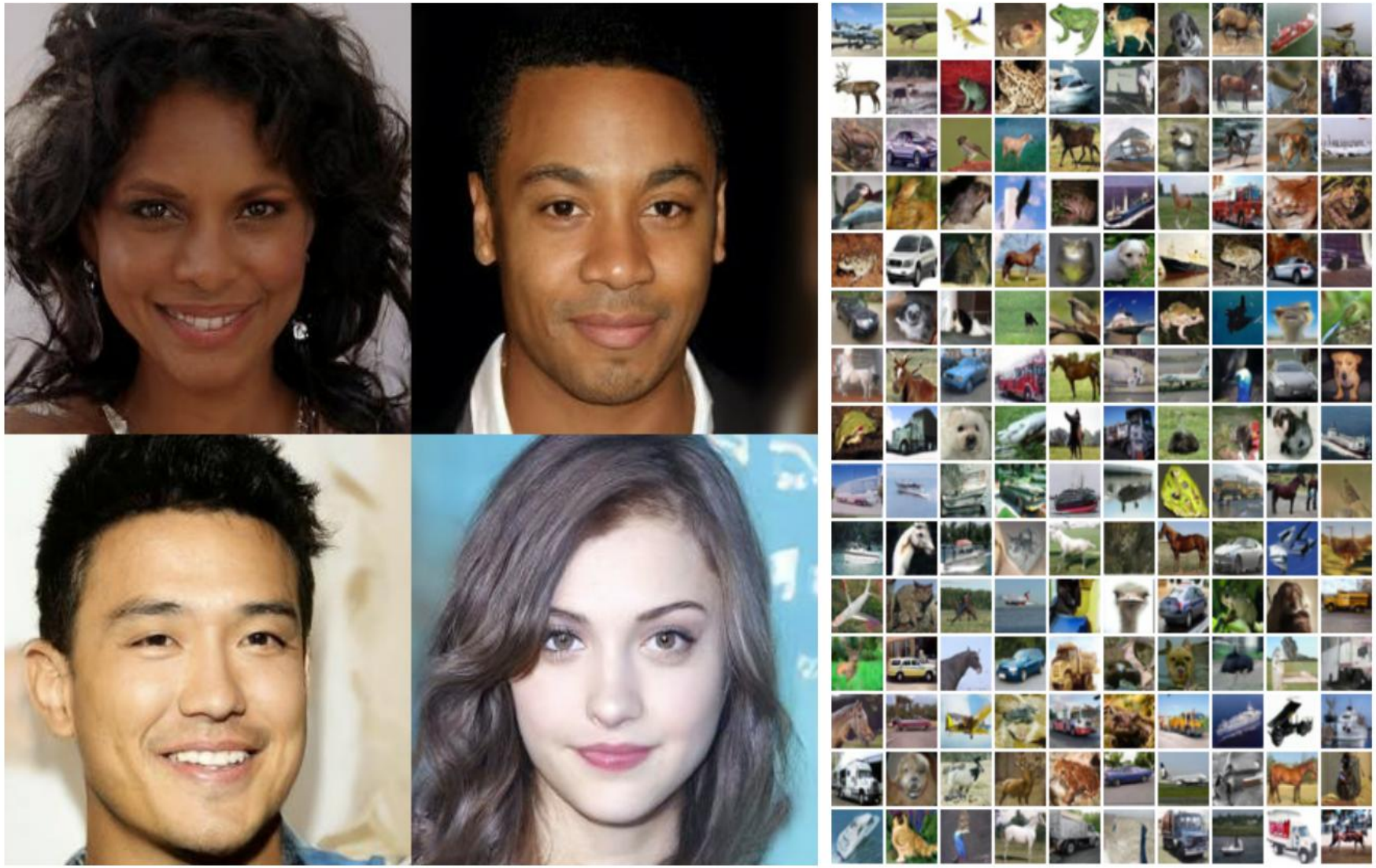
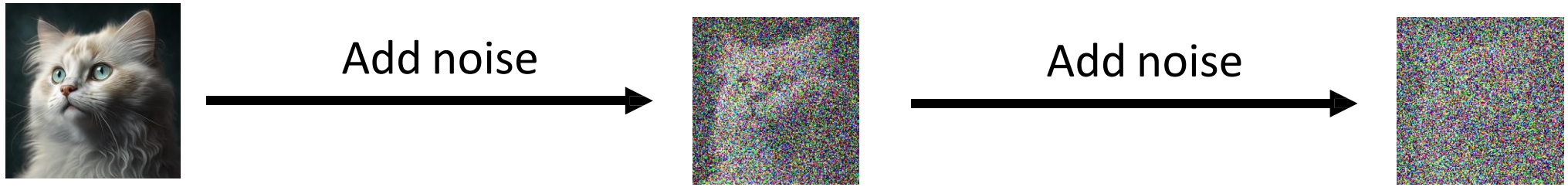


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

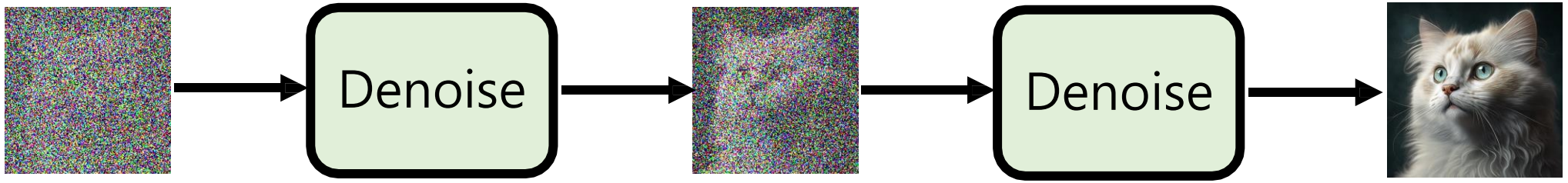
[Ho, Jain, Abbeel, 2020]

DDPM 基本概念

Forward Process

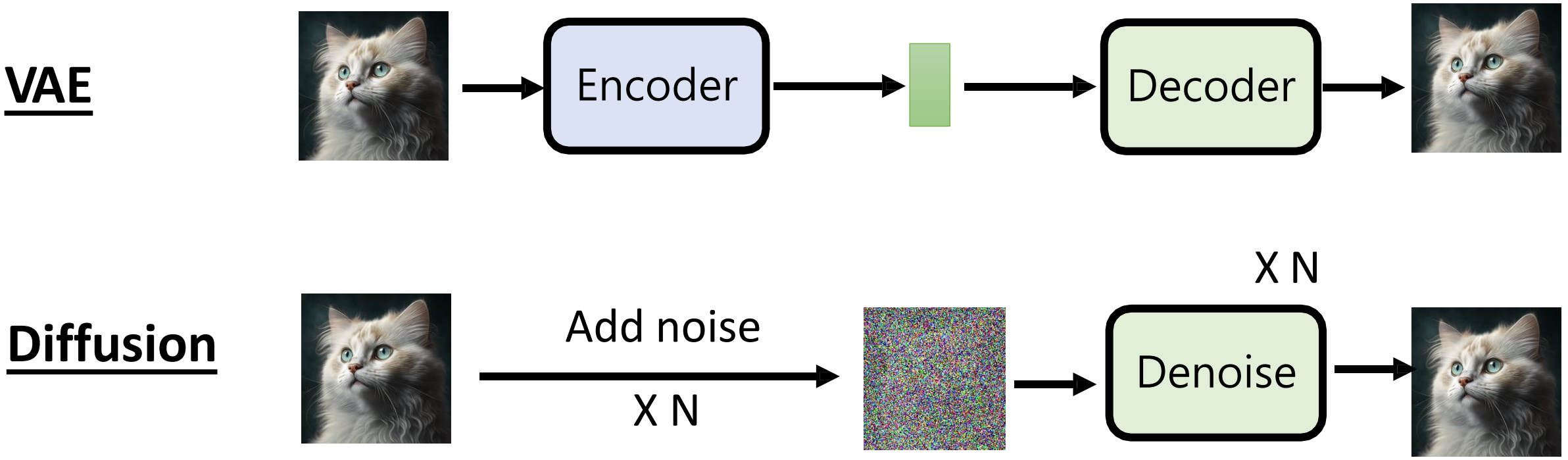


Reverse Process



Source: Hung-yi Lee

VAE vs. Diffusion Model



VAE 是利用 encoder 將圖片編碼至 latent space，再用 decoder 還原；
而 Diffusion model 是將原圖加入 noise 後再用 denoise 模型還原至原圖。

Source: Hung-yi Lee

Denoising Diffusion Probabilistic Models

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

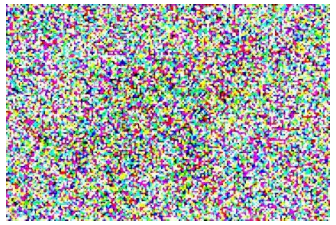
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Source: Hung-yi Lee

Training

x_0: clean image



ε: noise

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ ⟵ sample clean image
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ⟵ sample a noise
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right) \right\|^2$$

6: **until** converged

Noisy image

Target Noise

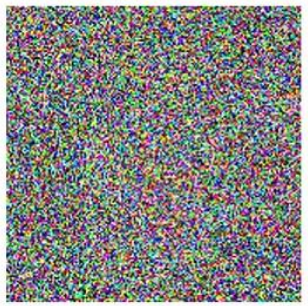
Noise predictor

$\bar{\alpha}_1, \bar{\alpha}_2, \dots \bar{\alpha}_T$ → smaller

Source: Hung-yi Lee

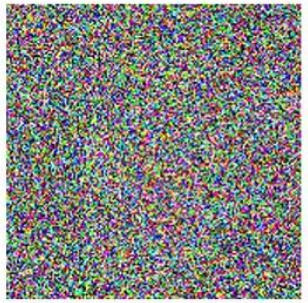
Training

x_0



ε

Sample t

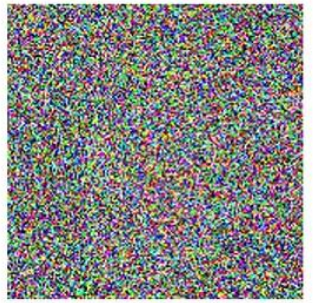
$$\sqrt{\bar{\alpha}_t}$$  $$+\sqrt{1-\bar{\alpha}}$$  $$=$$ 

x_0 ε

t

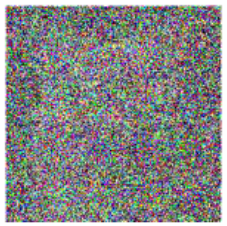
Noise Predicter

?????



ε

Inference



x_T

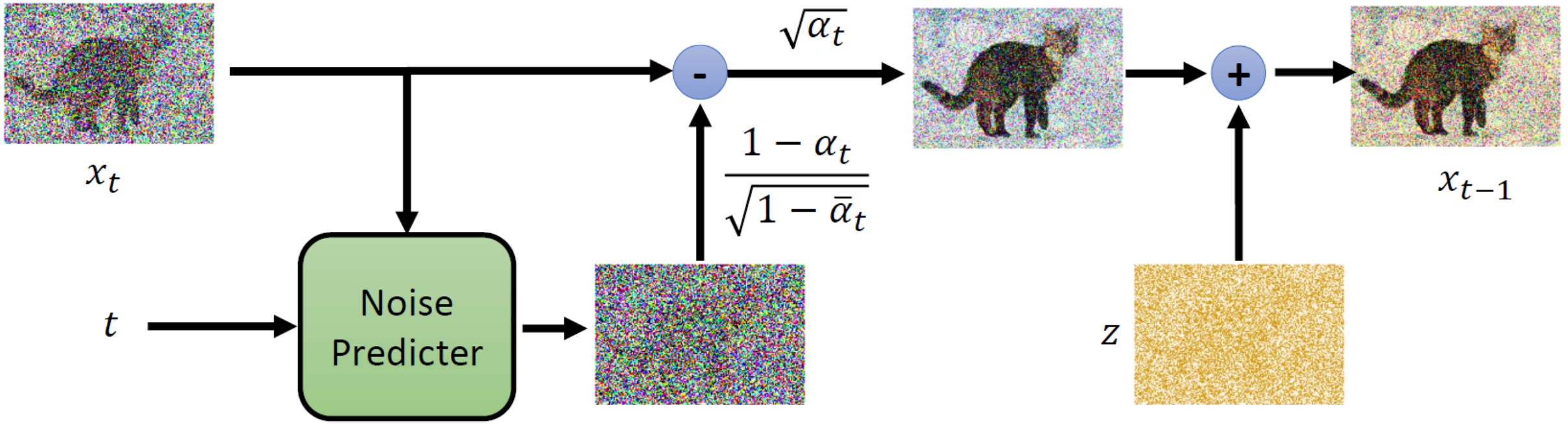
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

sample a noise?!

$\bar{\alpha}_1, \bar{\alpha}_2, \ldots \bar{\alpha}_T$

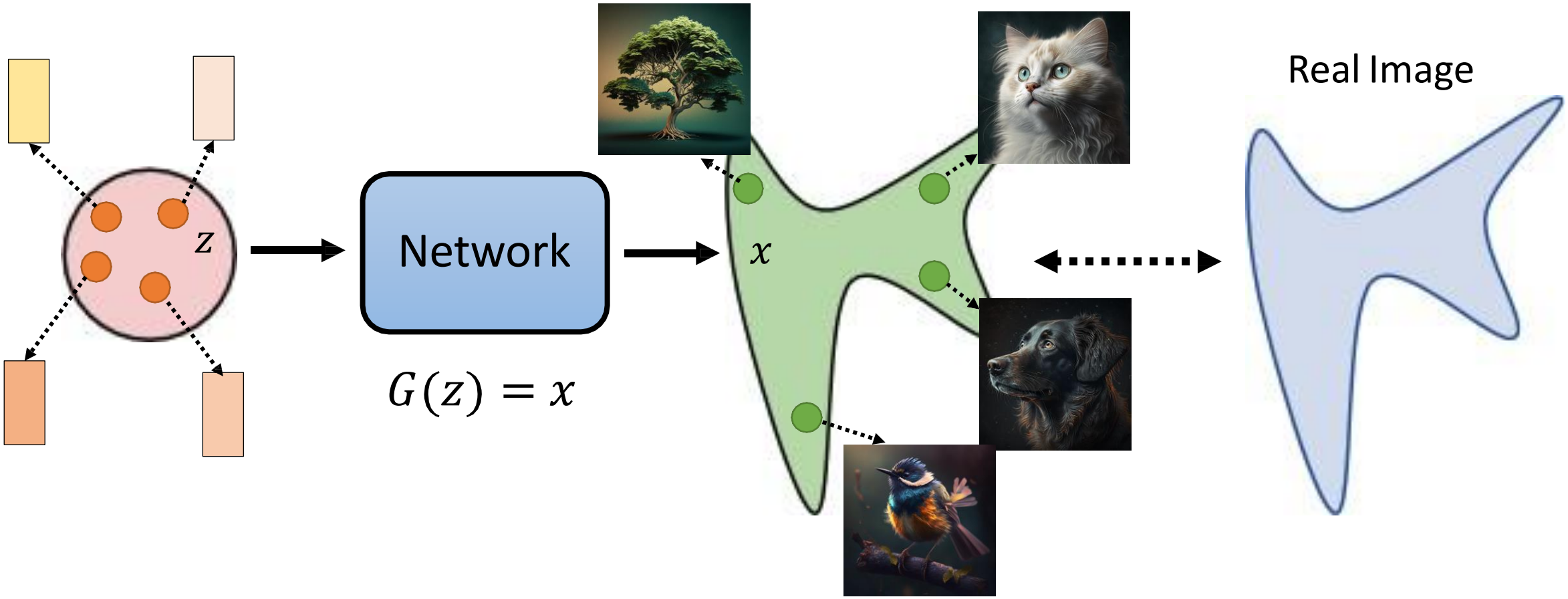
$\alpha_1, \alpha_2, \ldots \alpha_T$



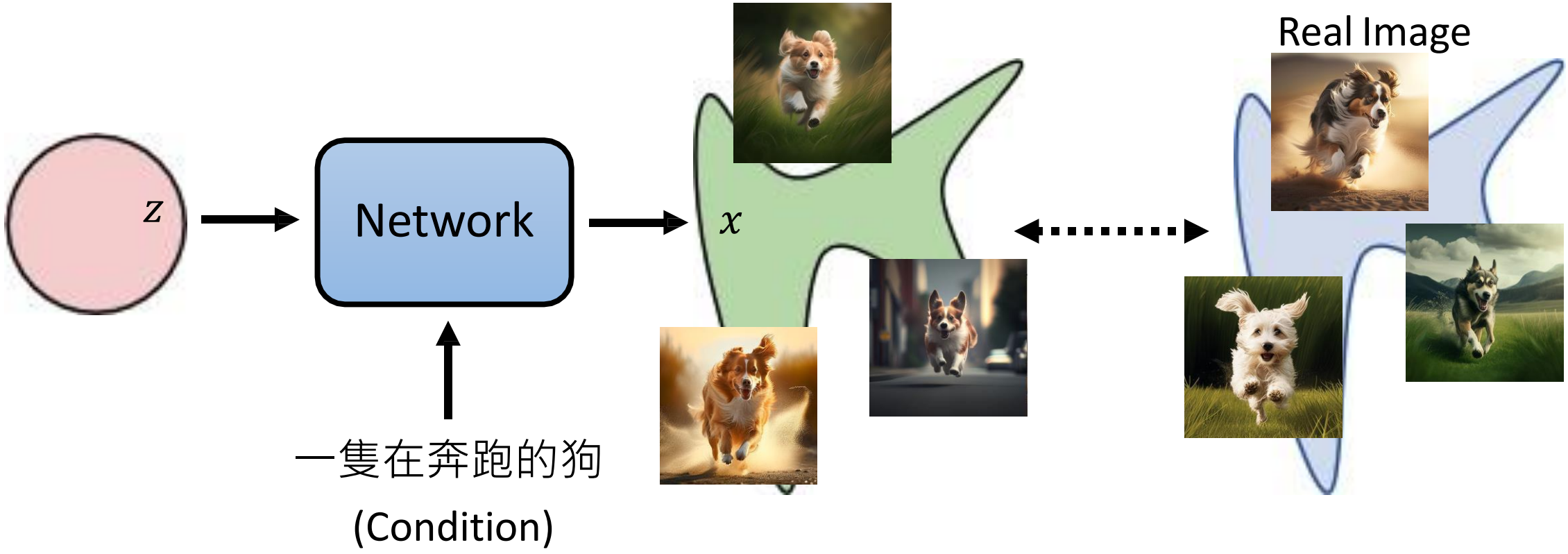
Source: Hung-yi Lee

MATH BEHIND DIFFUSION MODEL (DDPM)

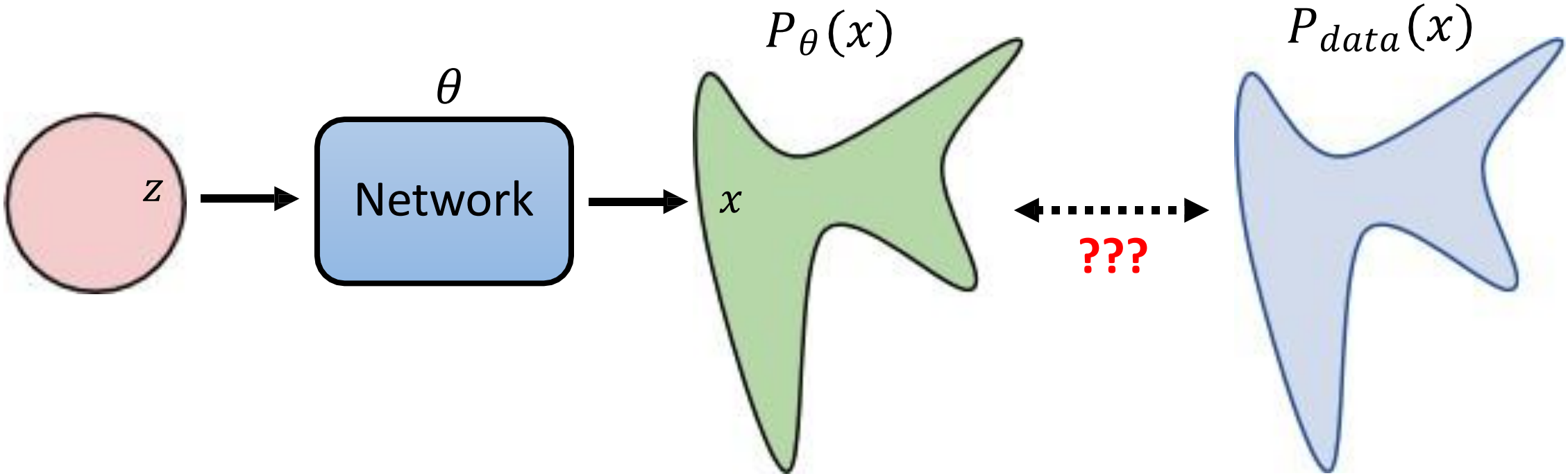
影像生成模型本質上的共同目標



影像生成模型本質上的共同目標



Maximum Likelihood Estimation



Sample $\{x^1, x^2, \ldots, x^m\}$ from $P_{data}(x)$

We can compute $P\ \ (x^i)$

???

$$\theta^* = arg\max_\theta \prod_{i=1}^{m} P_\theta(x^i)$$

從 Maximum Likelihood Estimation 的觀點分析 VAE 與 Diffusion Model

$$\theta^* = arg\max_\theta \prod_{i=1}^{m} P_\theta(x^i) = arg\max_\theta log \prod_{i=1}^{m} P_\theta(x^i)$$

Sample $x^1, x^2, \ldots, x^m$ from $P_{data}(x)$

$$= arg\max_\theta \sum_{i=1}^{m} logP_\theta(x^i) \approx arg\max_\theta E_{x\sim P_{data}}[logP_\theta(x)]$$

(not related to θ)

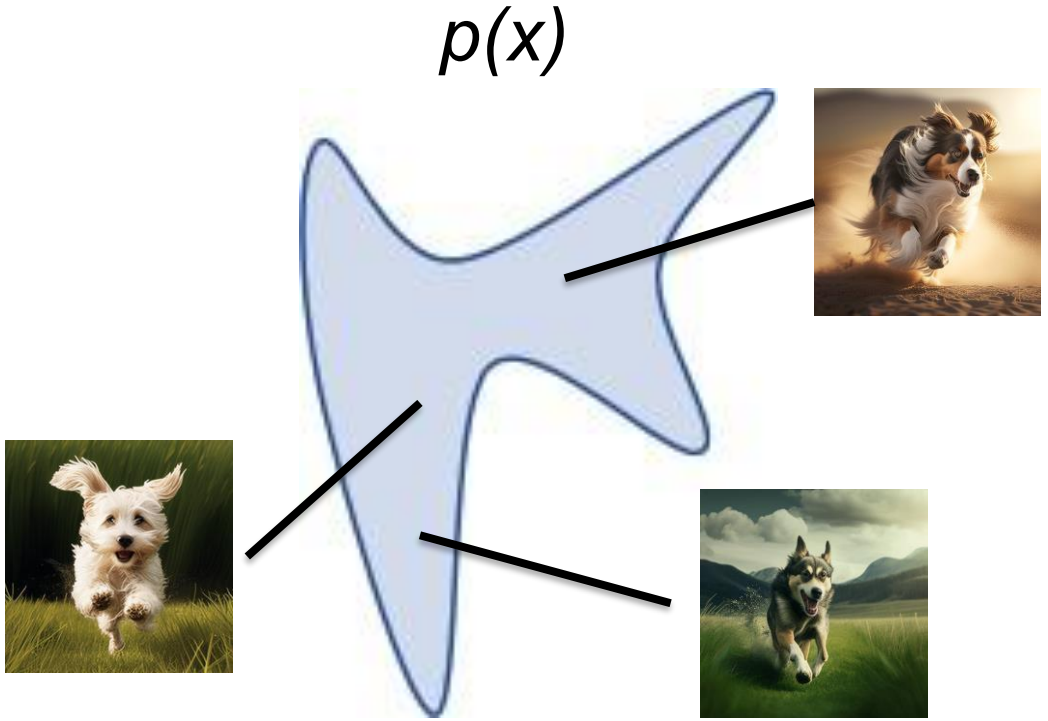
$$= arg\max_\theta \int_x P_{data}(x)logP_\theta(x)dx - \int_x P_{data}(x)logP_{data}(x)dx$$

Difference between P_{data} and P_θ

$$= arg\max_\theta \int_x P_{data}(x)log\frac{P_\theta(x)}{P_{data}(x)}dx = arg\min_\theta KL(P_{data}||P_\theta)$$

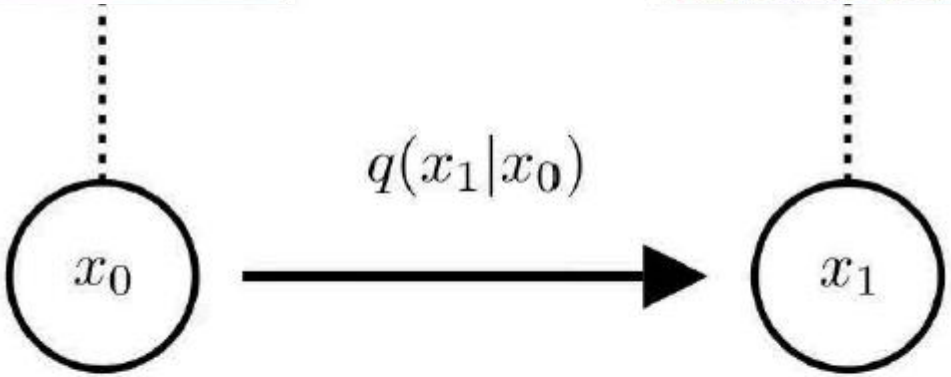
Maximum Likelihood = Minimize KL Divergence

資料生成的基本概念



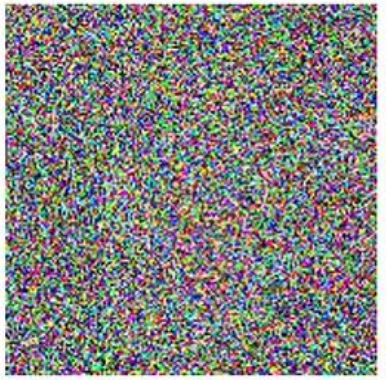
代表訓練資料的機率分布 p(x)

如何求得p(x)=?



$$x_1 = x_0 + \beta \cdot \epsilon \quad \Rightarrow \quad q(x_1|x_0) = \mathcal{N}(x_0, \beta)$$

$$\epsilon \sim \mathcal{N}(0, 1)$$



$$x_2 = x_1 + \beta \cdot \epsilon$$

x_0 x_1 x_2

$$q(x_1|x_0) = \mathcal{N}(x_0, \beta) \qquad q(x_2|x_1) = \mathcal{N}(x_1, \beta)$$

$$x_2 = x_0 + 2\beta \cdot \epsilon$$

x_0 x_2

$$q(x_2|x_0) = \mathcal{N}(x_0, 2\beta)$$

The Variance Explodes!

$$x_t = x_0 + t\beta \cdot \epsilon$$

x_0 → x_t

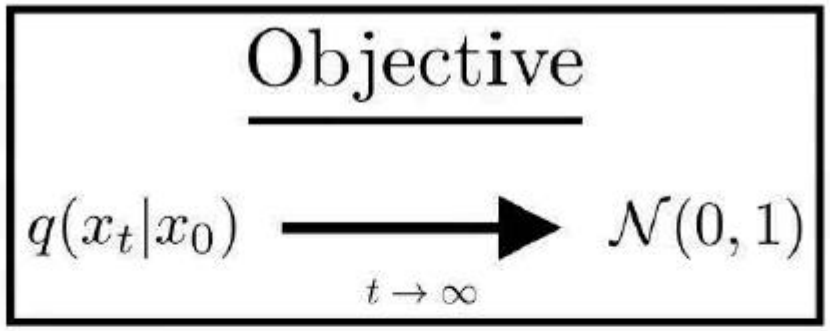
$$q(x_t \mid x_0) = \mathcal{N}(x_0, t\beta)$$

- 在時間步 t 時，q(Xt|X0) 的分佈仍然是以 X0 為中心的高斯分佈，方差為 tβ。
- 在這種設置中，平均值始終固定在 X0，並且從未被過程修改，而方差則持續無限增加。
- 結果是，這個過程**無法收斂到標準正態分佈**，因此被稱為「方差爆炸擴散」（variance exploding diffusion）

$$\not\rightarrow \mathcal{N}(0,1)$$

Objective

$$q(x_t|x_0) \xrightarrow[t \to \infty]{} \mathcal{N}(0,1)$$



$$q(x_t|x_{t-1}) = \sqrt{1-\beta}\ x_{t-1} + \beta \cdot \epsilon$$

方差保持擴散 (Variance Preserving Diffusion)

• 為了確保擴散過程能夠逐步將數據轉換為標準正態分佈，必須滿足兩個條件：**平均值必須收斂到零，方差必須收斂到一**。

DDPM 的作者修改了從一個步驟到下一個步驟的方式：

• **係數選擇：** 必須在分佈的平均值前引入一個係數，使其趨向於零。DDPM 作者選擇了 1−*β* 這個「神奇數字」。

• **收斂性：** 這種選擇產生了一個直接且方便的表達式，用來描述從無噪聲輸入 *X*0 到任何給定時間步 *t* 的條件分佈。

Objective

$$q(x_t|x_0) \xrightarrow[t \to \infty]{} \mathcal{N}(0,1)$$

$$q(x_t|x_{t-1}) = \sqrt{1-\beta}\ x_{t-1} + \beta \cdot \epsilon$$

$$\bar{\alpha}_t = (1-\beta)^t$$

一次算好，加滿noise $q(x_t|x_0) = \sqrt{\bar{\alpha}_t}\ x_0 + (1-\bar{\alpha}_t) \cdot \epsilon$

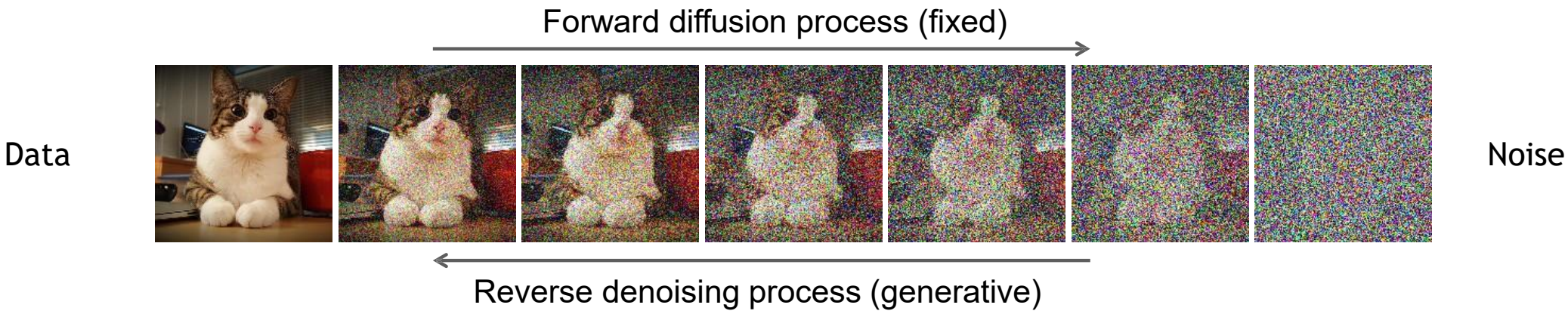
如果 β 被選擇在 0 和 1 之間，那麼隨著 t 增長到無窮大，αt bar 將收斂到零。此時，條件分佈的平均值趨向於零，而方差趨向於一。這就是一個合適的「方差保持擴散過程」

Denoising Diffusion Models

Learning to generate by denoising

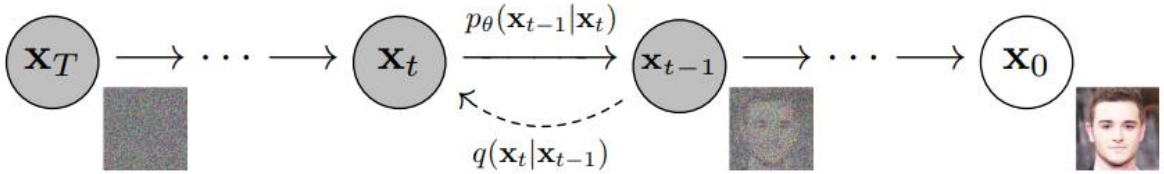
Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising

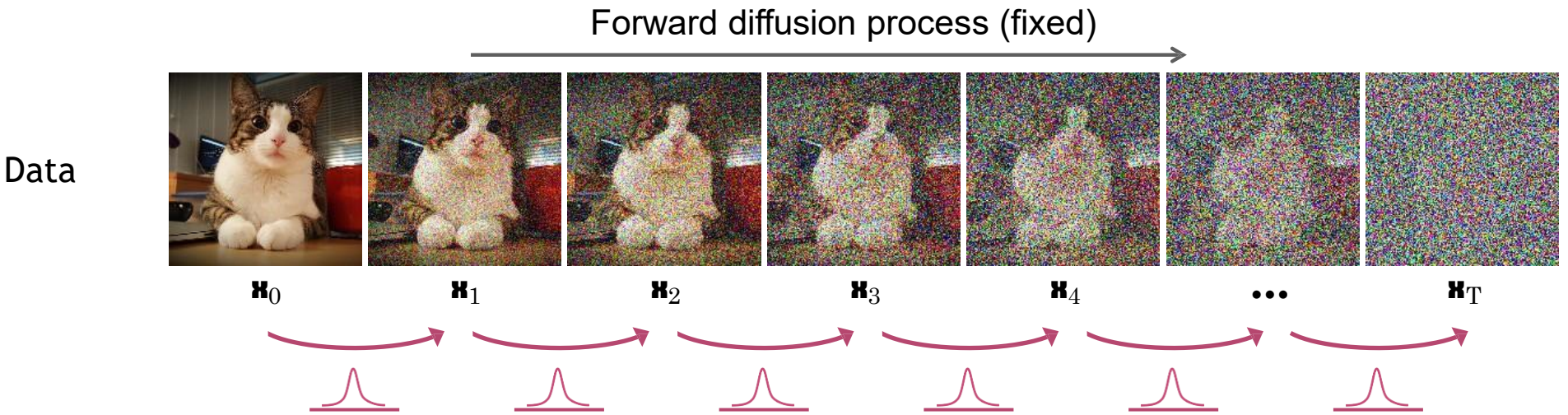


Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Forward Diffusion Process



The formal definition of the forward process in T steps:

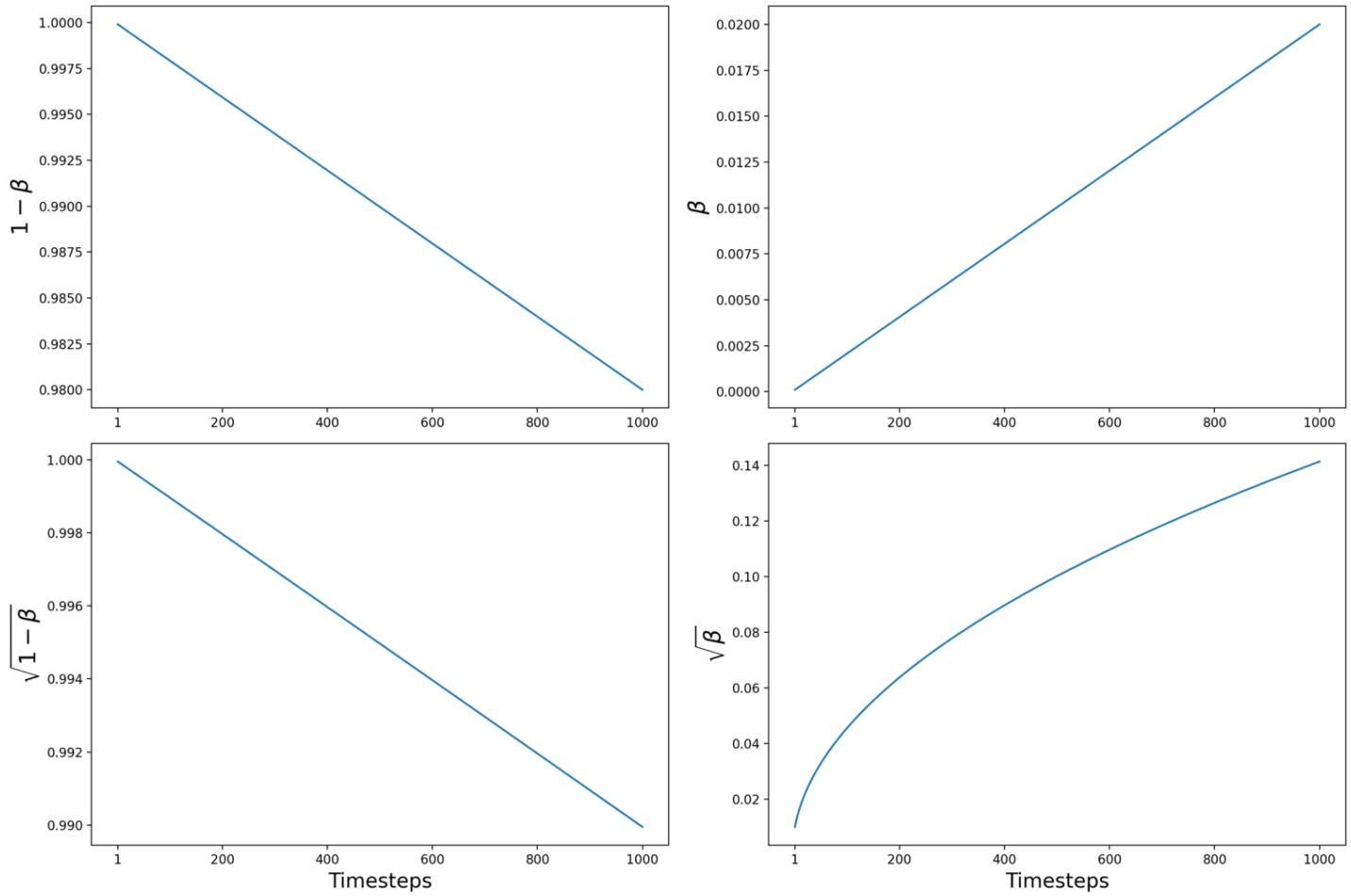


$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

mean variance (joint)

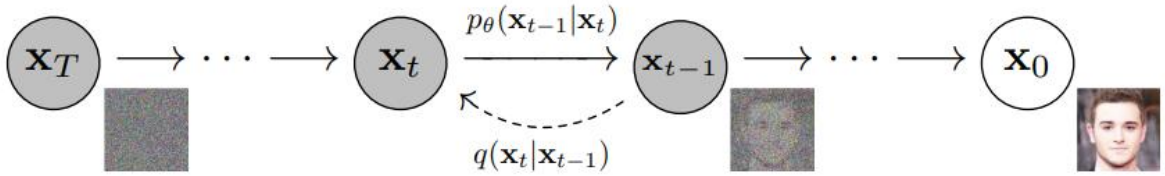
$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\,\epsilon \quad \ldots (3)$$
$$; \text{where} \quad \epsilon \sim \mathcal{N}(0, I)$$

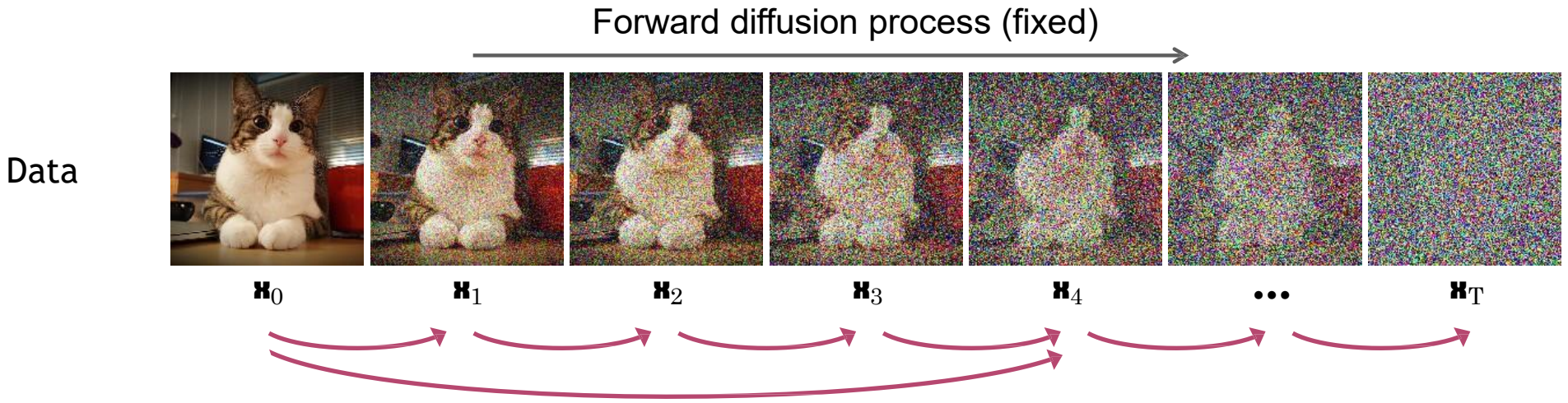
In practice, the authors of DDPMs use a "*linear variance scheduler*" and define **β** in the range **[0.0001, 0.02]** and set the total timesteps **T = 1000**



"Diffusion models scale down the data with each forward process step (by a $\sqrt{1-\beta_t}$ factor) so that variance does not grow when adding noise."

Diffusion Kernel



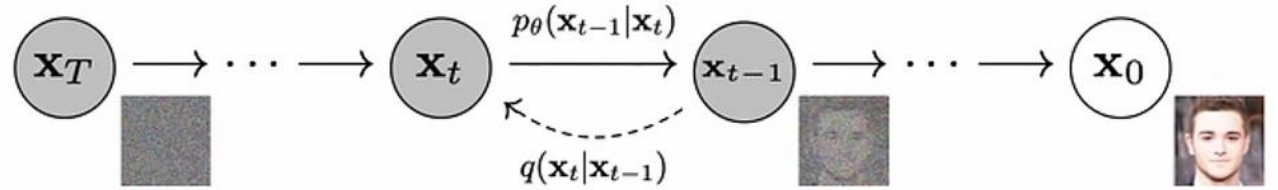


Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

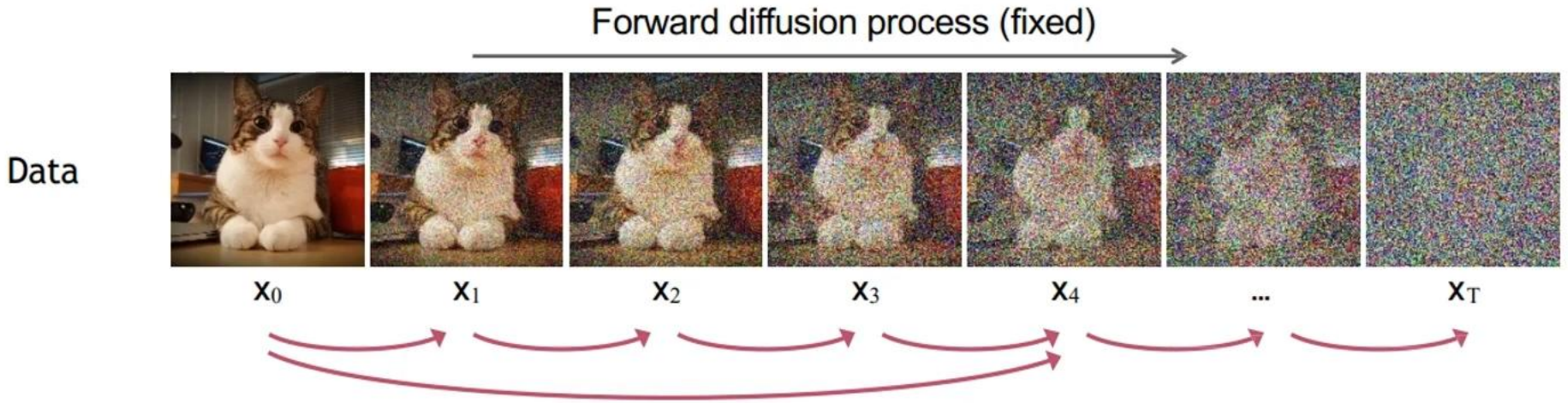
一次加滿noise For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Forward Process



The formal definition of the forward process in T steps:



$$
\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} && \text{;where } \epsilon_{t-1}, \epsilon_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-2}) + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\,\bar{\epsilon}_{t-2} && \text{;where } \bar{\epsilon}_{t-2} \text{ merges two Gaussians (*).} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\,\bar{\epsilon}_{t-2} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon
\end{aligned}
$$

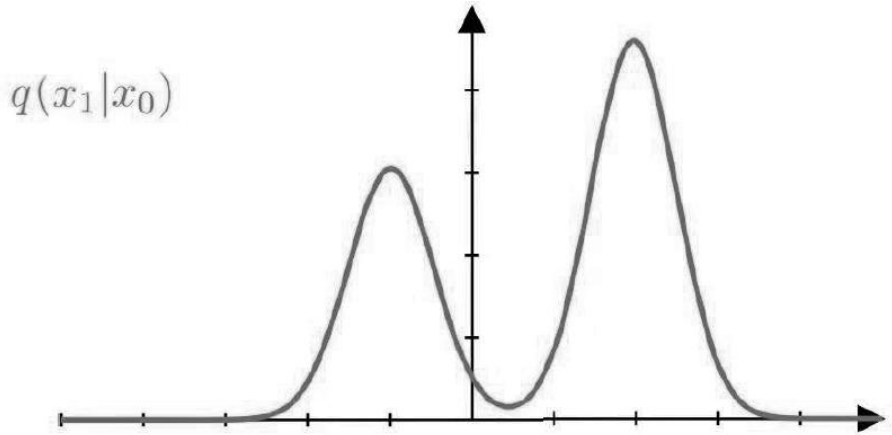
For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$

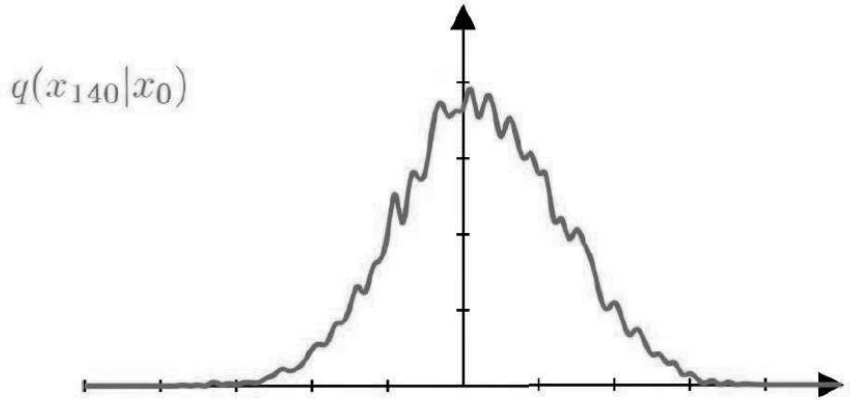
Objective

$$q(x_t|x_0) \xrightarrow[t \to \infty]{} \mathcal{N}(0,1)$$

$$q(x_t|x_0) = \sqrt{\bar{\alpha}_t}\, x_0 + (1 - \bar{\alpha}_t) \cdot \epsilon$$

$$\bar{\alpha}_t = (1 - \beta)^t$$





2 Background

Denoising Diffusion Probabilistic Models

Diffusion models [53] are latent variable models of the form $p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T}$, where $\mathbf{x}_1, \ldots, \mathbf{x}_T$ are latents of the same dimensionality as the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is called the *reverse process*, and it is defined as a Markov chain with learned Gaussian transitions starting at $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{1}$$

What distinguishes diffusion models from other types of latent variable models is that the approximate posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$, called the *forward process* or *diffusion process*, is fixed to a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, \ldots, \beta_T$:

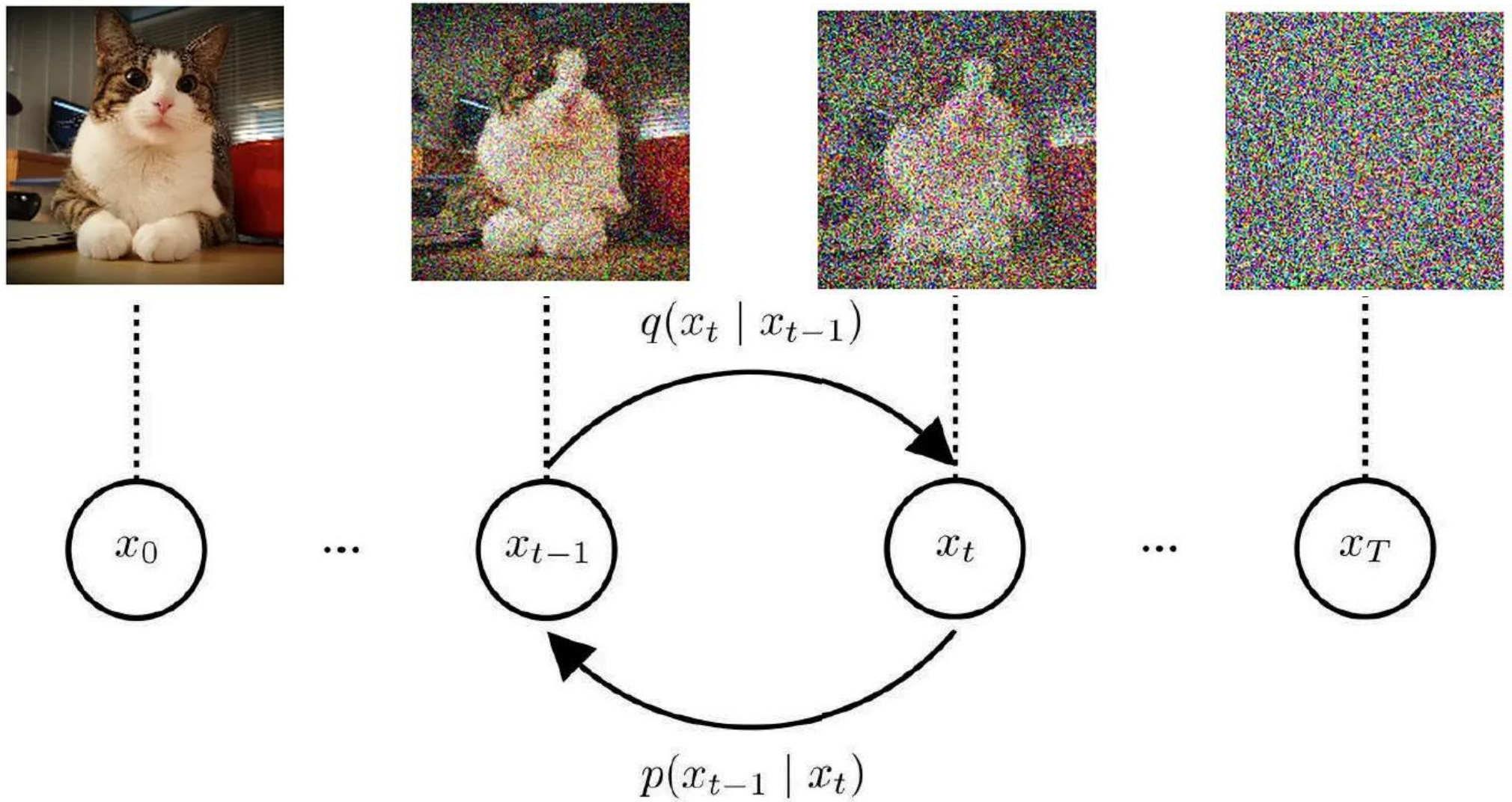
$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{2}$$

Training is performed by optimizing the usual variational bound on negative log likelihood:

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] =: L \tag{3}$$

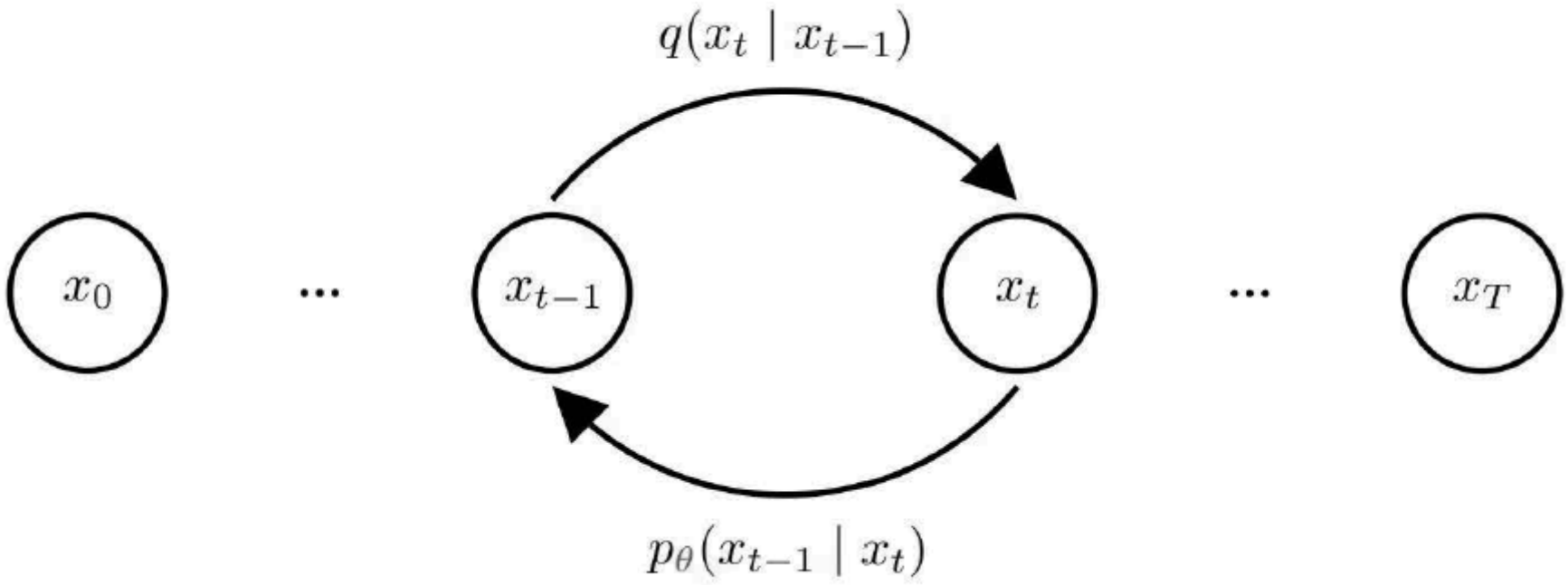
The forward process variances β_t can be learned by reparameterization [33] or held constant as hyperparameters, and expressiveness of the reverse process is ensured in part by the choice of Gaussian conditionals in $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, because both processes have the same functional form when β_t are small [53]. A notable property of the forward process is that it admits sampling $\mathbf{x}_t$ at an arbitrary timestep t in closed form: using the notation $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, we have

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{4}$$



$p_\theta(x_{t-1}|x_t)$ Learn with NN

How do we train the neural network?



We minimize the negative log-likelihood

$$\boxed{-\log p_\theta(x_0)}$$

We minimize the negative log-likelihood

訓練目標：最小化負對數似然 (Negative Log-Likelihood)

$$\boxed{-\log p_\theta(x_0)}$$

$$q(x_1, \ldots, x_T \mid x_0) = q(x_1 \mid x_0)\; q(x_2 \mid x_1, x_0) \;\ldots\; q(x_T \mid x_{T-1}, \ldots, x_0)$$

$$q(x_1, \ldots, x_T \mid x_0) = q(x_1 \mid x_0)\; q(x_2 \mid x_1) \;\ldots\; q(x_T \mid x_{T-1})$$

$$q(x_{1:T} \mid x_0) = q(x_1 \mid x_0)\; q(x_2 \mid x_1) \;\ldots\; q(x_T \mid x_{T-1}) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

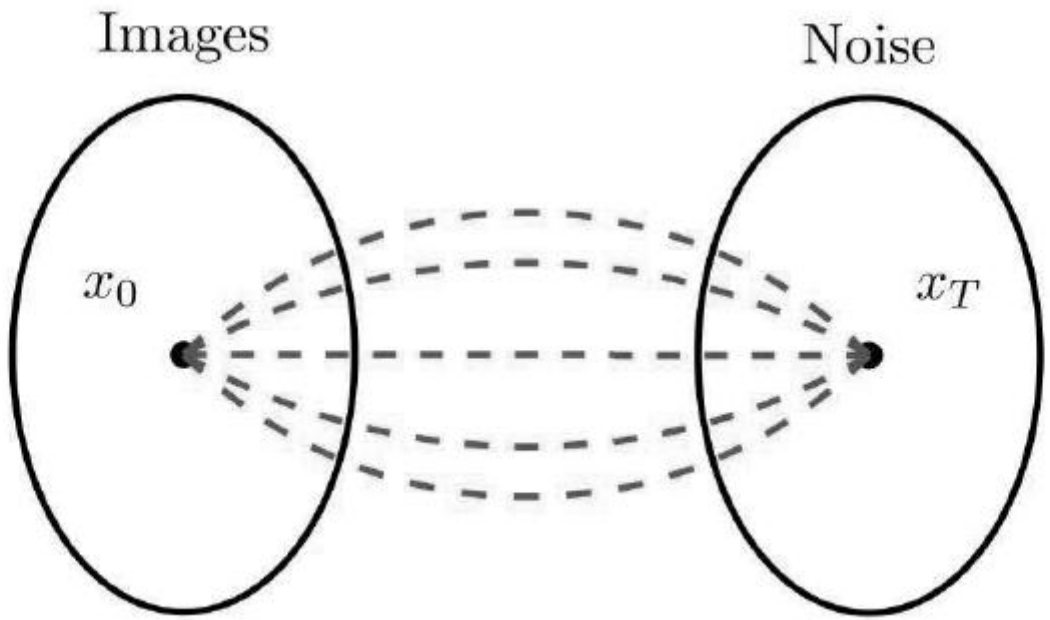
Complete forward process

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

Complete reverse process

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}) \qquad p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

$$-\log p_\theta(x_0) = -\log \int p_\theta(x_{0:T}) dx_{1:T}$$



直接計算此似然是難以處理的 (intractable)，因為它需要對所有可能的生成路徑求和

Forward

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

Backward

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

訓練目標：最小化負對數似然 (Negative Log-Likelihood)

$$-\log p_\theta(x_0) = -\log \int p_\theta(x_{0:T}) dx_{1:T}$$

用 Importance Sampling 來近似這個積分，用一個簡單的q(z)來抽樣出z

$$-\log p_\theta(x_0) = -\log \int q(x_{1:T} \mid x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} dx_{1:T}$$

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

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

Evidence lower bound (ELBO)

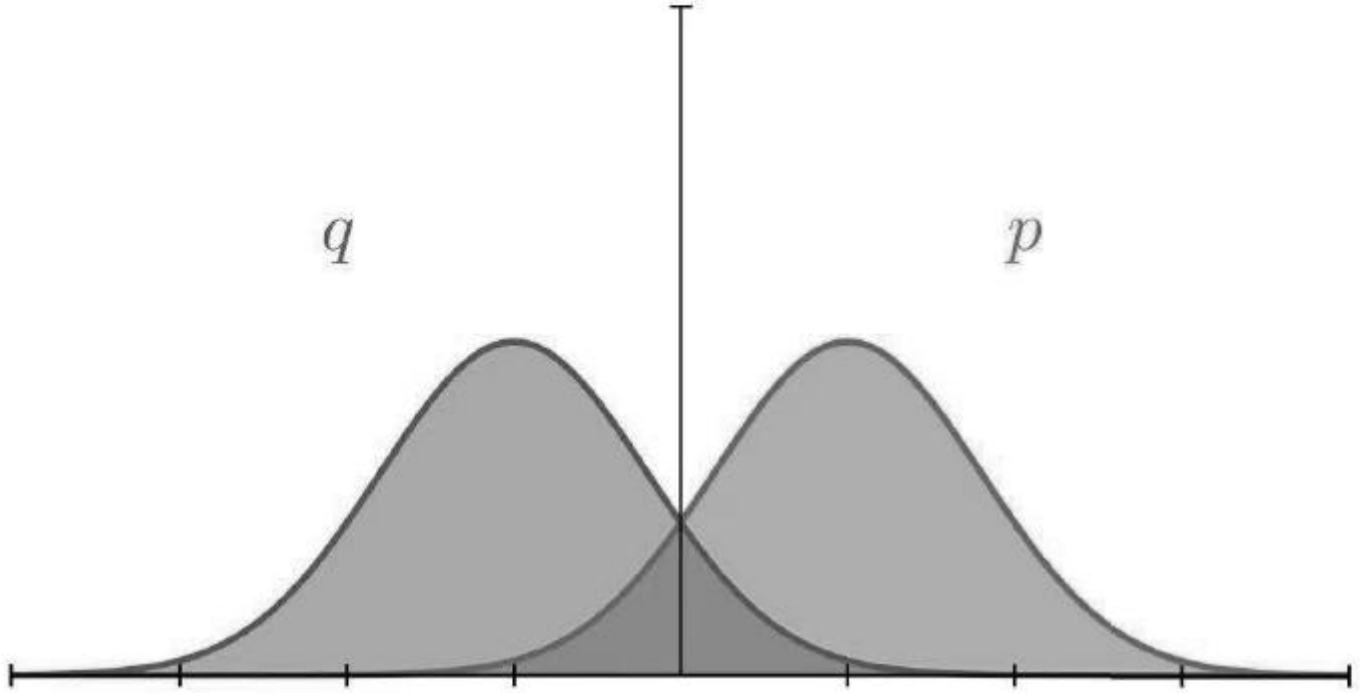
證據下界

簡化 ELBO

$$-\mathbb{E}_q \left[D_{\mathrm{KL}}(q(x_T \mid x_0) \,||\, p(x_T)) + \sum_{t>1} D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)) - \log p_\theta(x_0 \mid x_1) \right]$$

Very small

Does not depend on θ



Kullback-Leibler Divergence

最初，數學上「正確」的訓練目標是最小化生成真實圖像的「負對數似然」。不幸的是，這個目標是“難以處理的”，也就是說它無法直接計算。解決問題的第一步是將這個目標簡化為一個可實現的上限，稱為「證據下界」（ELBO）。透過一些代數技巧，ELBO 被表示為「Kullback-Leibler (KL) 散度」總和——KL 散度衡量的是去噪過程中每一步機率分佈之間的距離。

$$\mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p(\mathbf{x}_T))}_{L_T}+\sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Big] \tag{5}$$

A Extended derivations

Below is a derivation of Eq. (5), the reduced variance variational bound for diffusion models. This material is from Sohl-Dickstein et al. [53]; we include it here only for completeness.

$$L=\mathbb{E}_q\Big[-\log\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big] \tag{17}$$

$$=\mathbb{E}_q\Big[-\log p(\mathbf{x}_T)-\sum_{t\geq 1}\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\Big] \tag{18}$$

$$=\mathbb{E}_q\Big[-\log p(\mathbf{x}_T)-\sum_{t>1}\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}-\log\frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\Big] \tag{19}$$

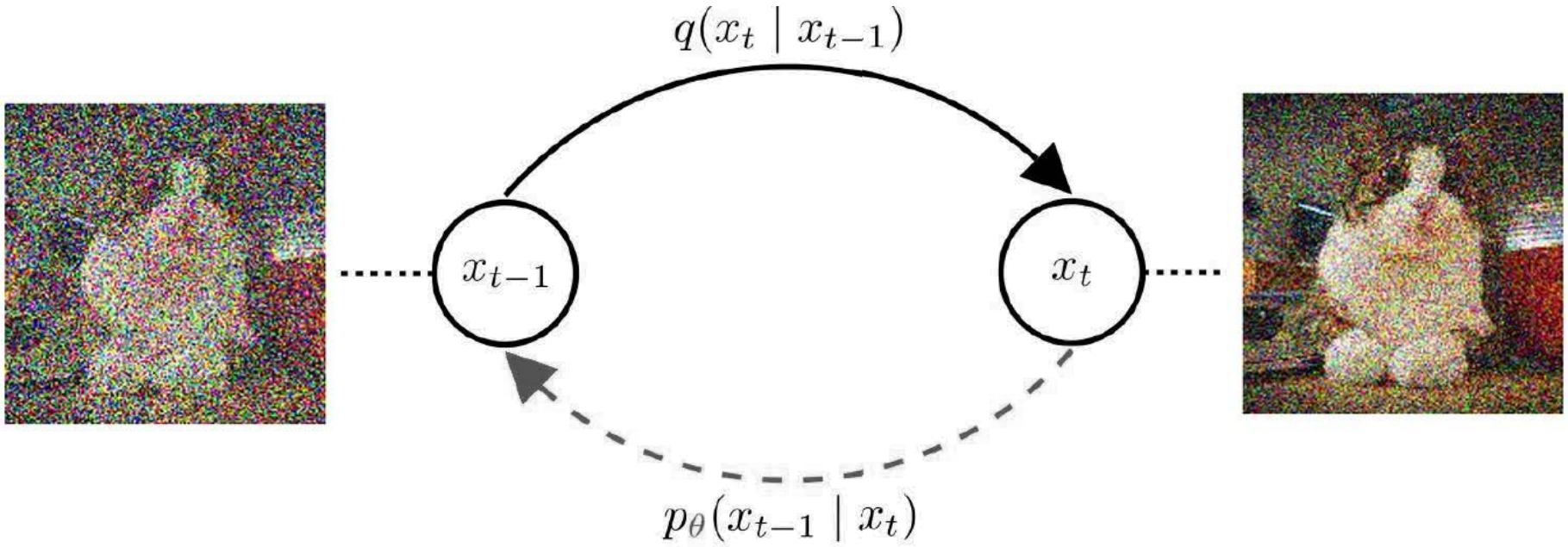
$$=\mathbb{E}_q\Big[-\log p(\mathbf{x}_T)-\sum_{t>1}\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\cdot\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}-\log\frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\Big] \tag{20}$$

$$=\mathbb{E}_q\Big[-\log\frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)}-\sum_{t>1}\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big] \tag{21}$$

簡化後的訓練目標

$$-\mathbb{E}_q \left[\sum_{t>1} D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)) \right]$$

Reverse step



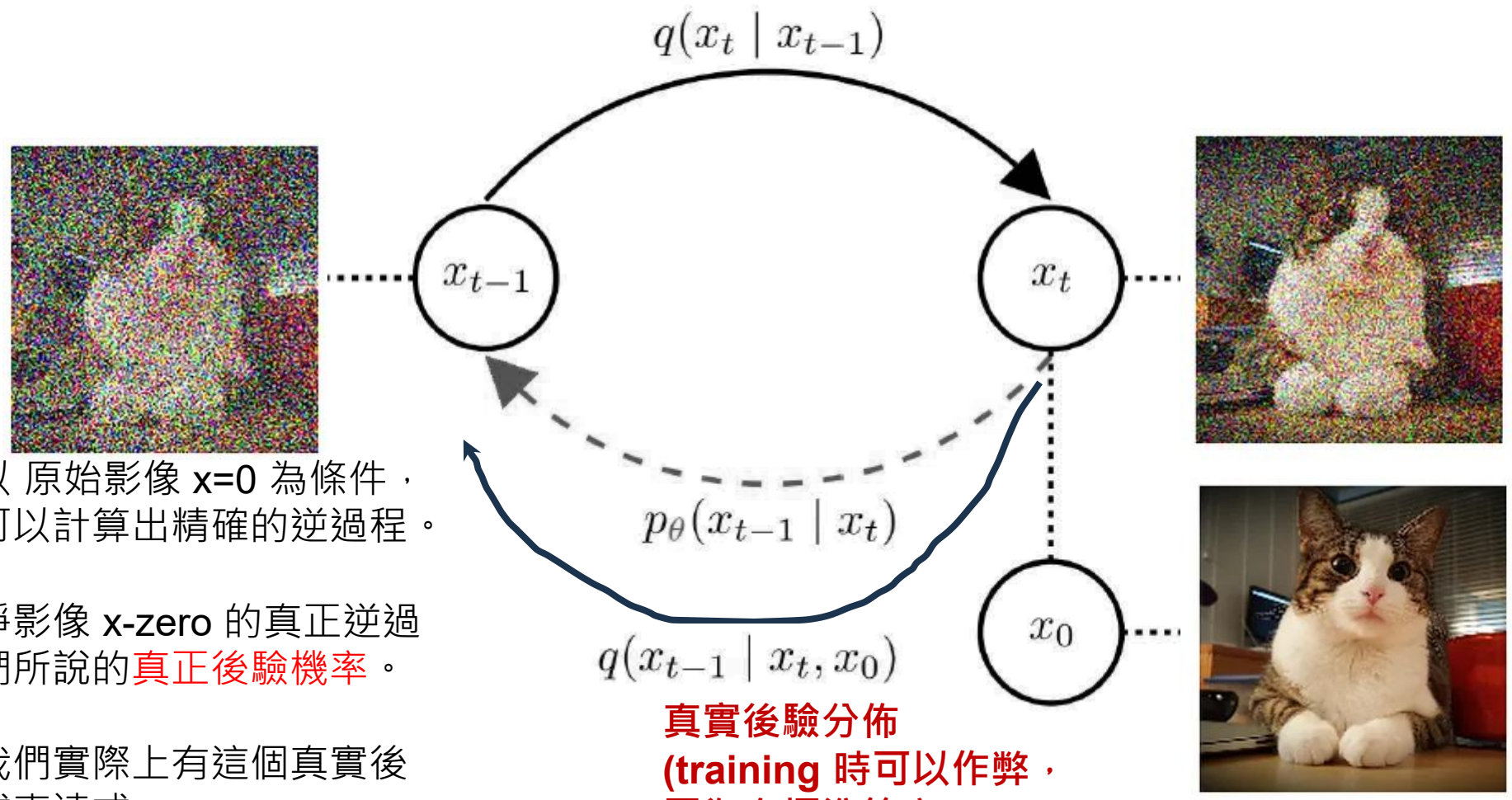
知道$q(x_t|x_{t-1})$，有一對應的reverse step $p_\theta(x_t|x_{t-1})$ ，

透過調整分佈 p 的參數(NN)來近似x_{t-1}

簡化後的訓練目標

$$-\mathbb{E}_q \left[\sum_{t>1} D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)) \right]$$

True posterior
要預測的目標



如果我們再以 原始影像 x=0 為條件，我們實際上可以計算出精確的逆過程。

這種使用乾淨影像 x-zero 的真正逆過程，就是我們所說的真正後驗機率。

好消息是，我們實際上有這個真實後驗分佈的閉式表達式。

真實後驗分佈
(training 時可以作弊，因為有標準答案)

$$q(x_{t-1} \mid x_t, x_0)$$

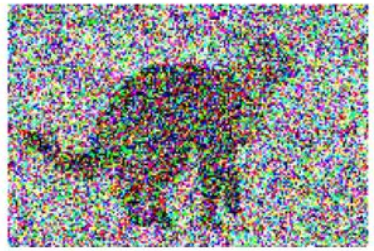
x_0

$q(x_t|x_0)$

$q(x_{t-1}|x_0)$

$q(x_t|x_{t-1})$

x_{t-1}



x_t

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

已知 Gaussian

已知 Gaussian

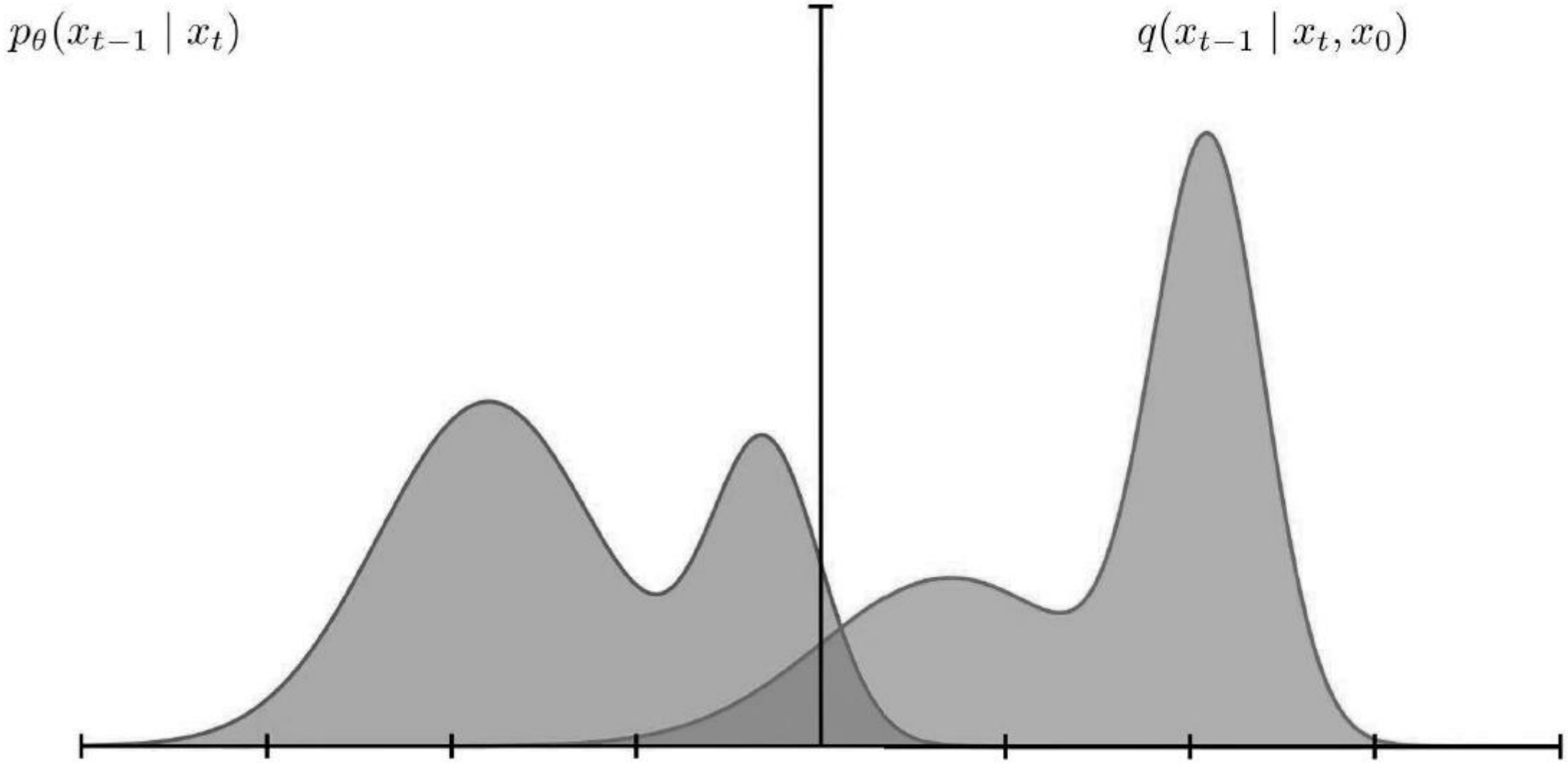
$$= \frac{q(x_{t-1}, x_t, x_0)}{q(x_t, x_0)} = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)q(x_0)}{q(x_t|x_0)q(x_0)} = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

已知 Gaussian

https://arxiv.org/pdf/2208.11970.pdf

- 存在閉式解：
  - 若以原始影像 x_0 為條件，逆過程（即真實後驗分佈）具有精確的閉式表達式。
- 推論限制：
  - 推論（Inference）時因無法預知 x_0，故無法直接使用此真實後驗。
- 訓練目標：
  - 但在訓練期間 x_0 已知，因此目標是訓練神經網路去匹配這個真實的後驗分佈。

- 一句話總結：
  - 雖然以 x_0 為條件的真實後驗分佈有閉式解，但推論時 x_0 未知；因此我們利用訓練時已知的 x_0，指導神經網路去逼近這個真實後驗。


$p_\theta(x_{t-1} \mid x_t)$
$q(x_{t-1} \mid x_t, x_0)$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta, \sigma_\theta)$$

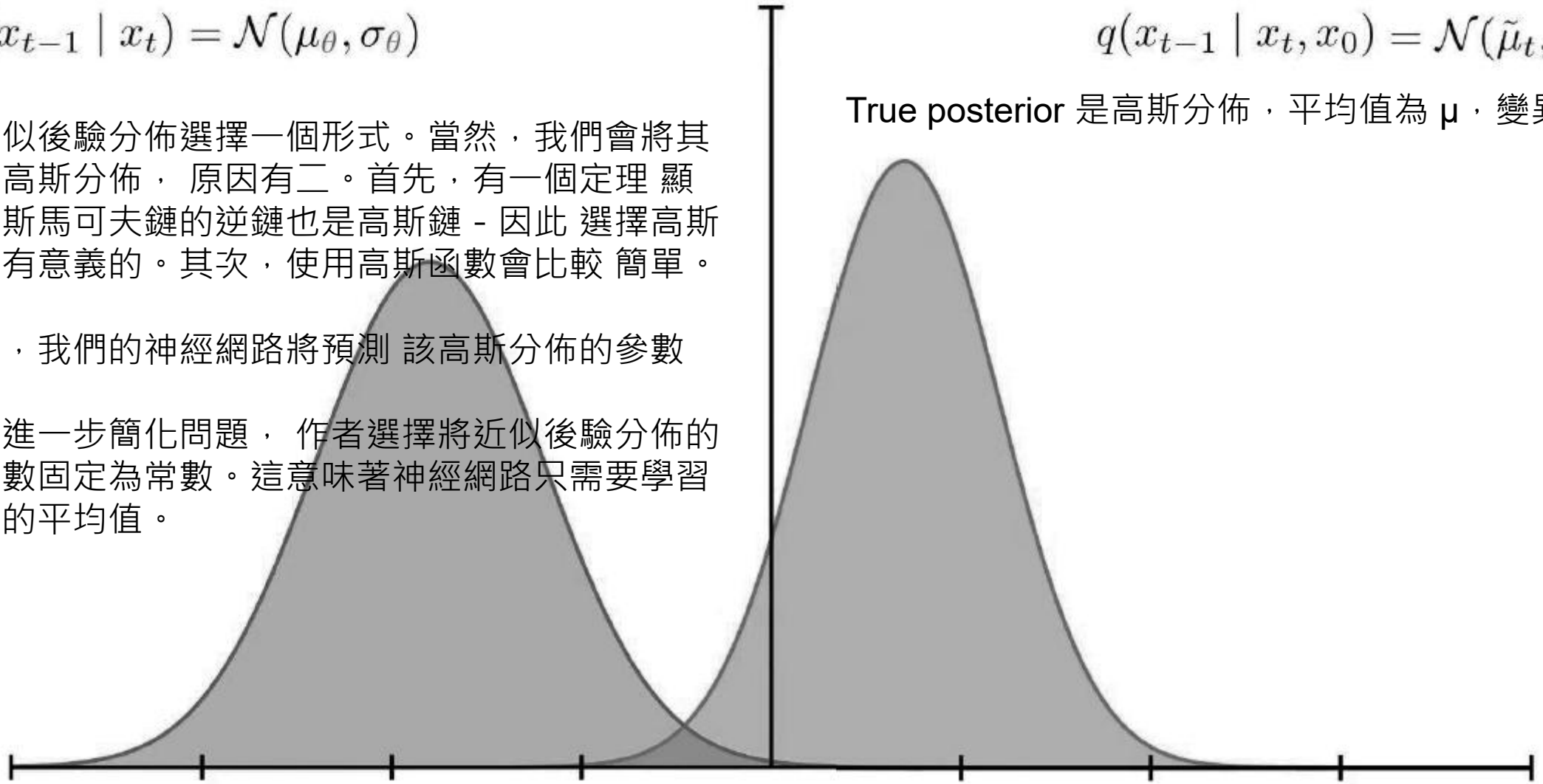
$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\tilde{\mu}_t, \tilde{\beta}_t)$$

True posterior 是高斯分佈，平均值為 μ，變異數為 β。

為近似後驗分佈選擇一個形式。當然，我們會將其設為高斯分佈， 原因有二。首先，有一個定理 顯示高斯馬可夫鏈的逆鏈也是高斯鏈 - 因此 選擇高斯鍊是有意義的。其次，使用高斯函數會比較 簡單。

因此，我們的神經網路將預測 該高斯分佈的參數

為了進一步簡化問題， 作者選擇將近似後驗分佈的變異數固定為常數。這意味著神經網路只需要學習分佈的平均值。

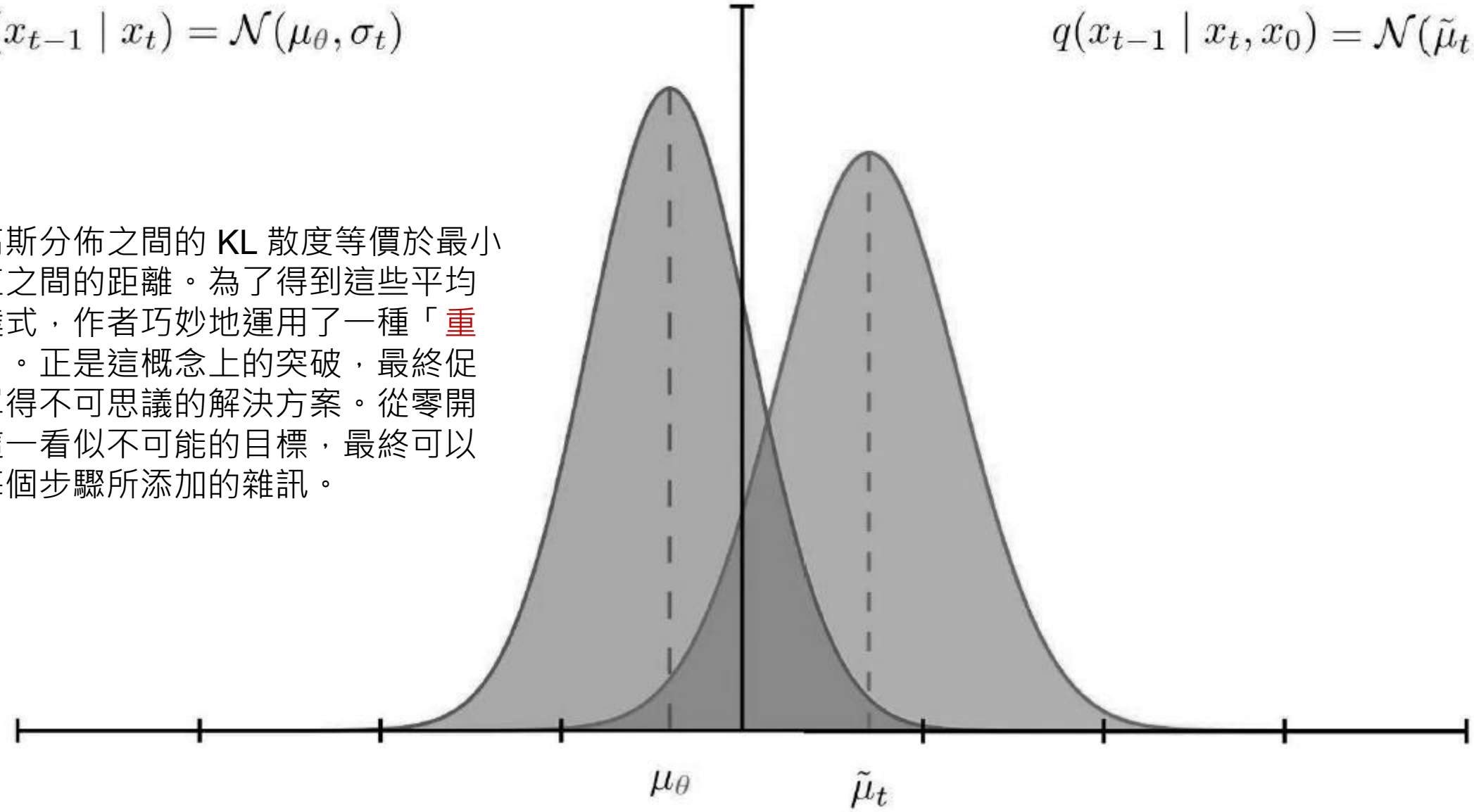


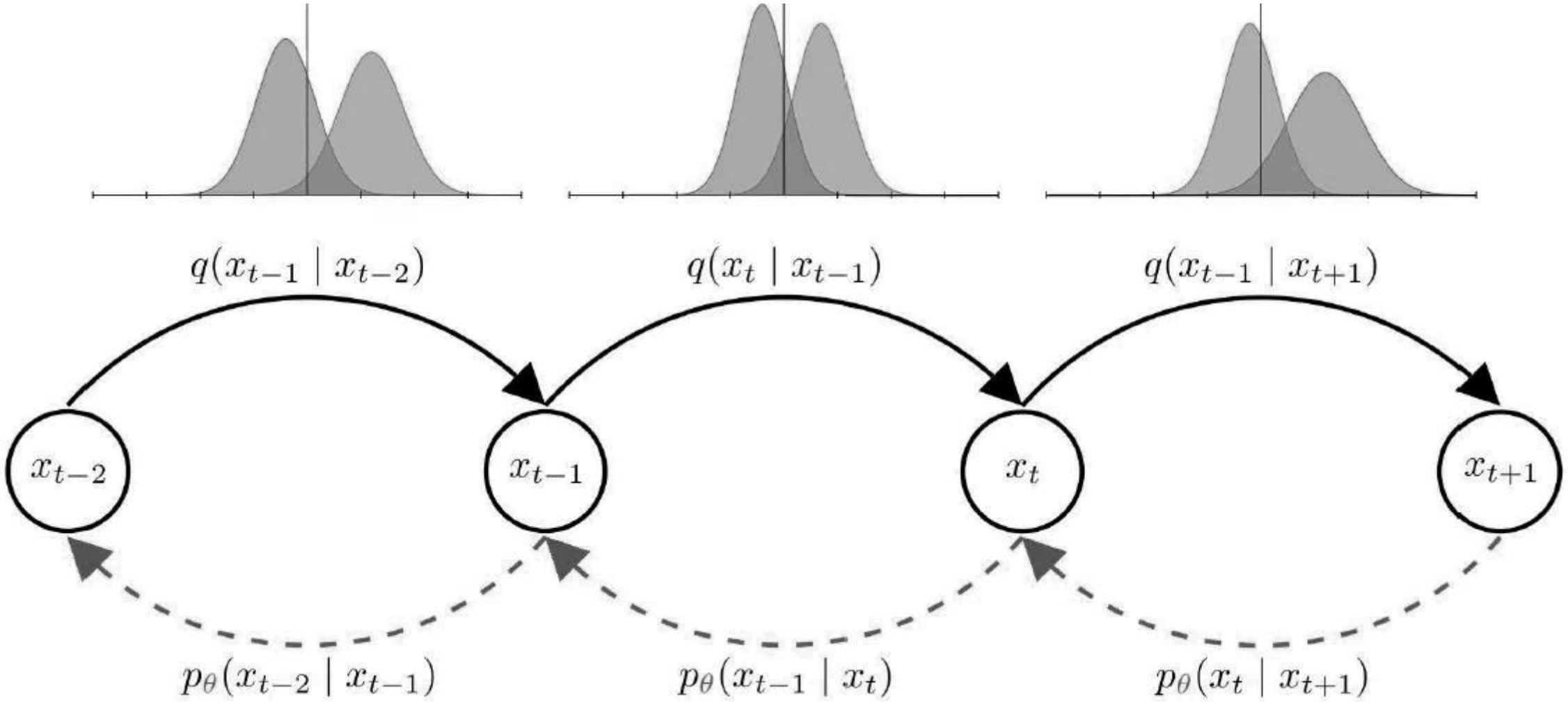
$$D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)) = \frac{1}{2\sigma_t^2}\|\tilde{\mu}_t - \mu_\theta\|^2$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta, \sigma_t)$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\tilde{\mu}_t, \tilde{\beta}_t)$$

最小化兩個高斯分佈之間的 KL 散度等價於最小化它們平均值之間的距離。為了得到這些平均值的簡單表達式，作者巧妙地運用了一種「重參數化技巧」。正是這概念上的突破，最終促成了幾乎簡單得不可思議的解決方案。從零開始產生影像這一看似不可能的目標，最終可以歸結為預測每個步驟所添加的雜訊。




$q(x_{t-1} \mid x_{t-2})$
$q(x_t \mid x_{t-1})$
$q(x_{t-1} \mid x_{t+1})$
x_{t-2}
x_{t-1}
x_t
x_{t+1}
$p_\theta(x_{t-2} \mid x_{t-1})$
$p_\theta(x_{t-1} \mid x_t)$
$p_\theta(x_t \mid x_{t+1})$

$$-\mathbb{E}_q\left[\sum_{t>1} D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t))\right]$$

最小化兩個高斯分佈之間的 KL 散度等價於最小化它們平均值之間的距離。

$$\mathbb{E}_q\left[\sum_{t>1} \frac{1}{2\sigma_t^2}\|\tilde{\mu}_t - \mu_\theta(x_t, t)\|^2\right]$$

True posterior 的close form

$$\tilde{\mu}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$$

reparameterization trick

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\epsilon \longrightarrow x_0 = \frac{1}{\sqrt{\alpha_t}}(x_0 - \sqrt{1-\alpha_t}\varepsilon)$$

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) \qquad \mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$$

帶回loss

Final loss function $\mathcal{L} = \mathbb{E}_q\left[\sum_{t>1} \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)}\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$

Target Noise

Noise predictor

模型不再學習「人臉長什麼樣子」這種極其複雜的概念，而是學習「這張圖片中剛剛添加了什麼噪音」這種更為簡單易行的任務。

最後得到的損失函數要簡單得多，就是 ε 和 εθ 之間的平方距離。因此，**εθ 就成了網路對添加到 x0 以產生 xt 的噪音的估計**。而最小化損失函數就意味著讓這個預測盡可能準確。

Recap of all the steps

1 Negative log-likelihood

$$-\log p_\theta(x_0) = -\log \int p_\theta(x_0, x_t) dx_t$$

→

2 Evidence lower bound

$$-\log p_\theta(x_0) \leq \mathbb{E}_q \left[-\log \frac{p_\theta(x_0, x_t)}{q(x_t \mid x_0)} \right]$$

→

3 Rewrite the ELBO

$$\mathbb{E}_q \big[D_{\mathrm{KL}}(q(x_T \mid x_0) \,||\, p(x_T)) + \sum_{t>1} D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)) - \log p_\theta(x_0 \mid x_1) \big]$$

↓

4 Simplify the ELBO

$$\mathbb{E}_q \left[\sum_{t>1} D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)) \right]$$

←

5 Use Gaussians

$$\mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \sum_{t>1} \|\tilde{\mu}_t - \mu_\theta(x_t, t)\|^2 \right]$$

←

6 Reparameterization trick

$$\mathbb{E}_q \left[\sum_{t>1} \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right]$$

$$\mathbb{E}_q \left[\sum_{t>1} \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right]$$

$$\mathbb{E}_{q,t} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right]$$

the training pipeline

```
#For MNIST dataset

for i in range(num_epochs):

  # 1) sample data from our loader
  for x,_ in train_loader:

    # 2) get random time step to make good generalization
    t = torch.randint(0,num_time_steps,(batch_size,))

    # 3) add noise to our sampled data (forward diffusion process)
    e = torch.randn_like(x, requires_grad=False)
    a = scheduler.alpha[t].view(batch_size,1,1,1).cuda()
    x = (torch.sqrt(a)*x) + (torch.sqrt(1-a)*e)

    # 4) pass defused data to model
    output = model(x, t)

    optimizer.zero_grad()
    # 5) compare output with noise - the objective is to predict the "noise"
    loss = criterion(output, e)

    loss.backward()
    optimizer.step()
```

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

https://medium.com/@mathparracho/denoising-diffusion-probabilistic-models-ddpm-generative-intuition-practice-ef6b10d232a2

the sampling pipeline

```
def sampling(model, scheduler, num_time_steps=1000):

    # 1) gaussian noise initialization for our final image
    x_t = torch.randn(1, 1, 32, 32).cuda()

    # 2) start the reverse diffusion process
    for t in reversed(range(1, num_time_steps + 1)):
    t_tensor = torch.tensor([t], dtype=torch.float32).cuda()

    # 3) create the noise to be added to our process
    z = torch.randn_like(x_t) if t > 1 else 0

    # 4) implementation of the "langevin dynamics" step
    alpha_t = scheduler.alpha[t]
    alpha_t_bar = scheduler.alpha_cumprod[t]
    sigma_t = scheduler.beta[t].sqrt()

    x_t = (1 / alpha_t.sqrt()) * (x_t - ((1 - alpha_t) / (1 - alpha_t_bar).sqrt()) * model(x_t, t_tensor)) + sigma_t * z

    return x_t.cpu().detach()
```

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

sample a noise?!

$\bar{\alpha}_1, \bar{\alpha}_2, \ldots \bar{\alpha}_T$

$\alpha_1, \alpha_2, \ldots \alpha_T$

```
27      with torch.no_grad():
28          x = torch.randn(n_samples, 1, image_size, image_size).to(device)
29
30          for t in reversed(range(timesteps)):
31              t_tensor = torch.ones(n_samples, device=device).long() * t
32
33              predicted_noise = model(x, t_tensor, type_t="timestep")
34
35              alpha = alphas[t]
36              alpha_cumprod = alphas_cumprod[t]
37              beta = betas[t]
38
39              if t > 0:
40                  noise = torch.randn_like(x)
41              else:
42                  noise = 0
43
44              x = (1 / torch.sqrt(alpha)) * (
45                  x - (beta / torch.sqrt(1 - alpha_cumprod)) * predicted_noise
46              ) + torch.sqrt(beta) * noise
```

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

sample a noise?!

$\bar{\alpha}_1, \bar{\alpha}_2, \ldots \bar{\alpha}_T$

$\alpha_1, \alpha_2, \ldots \alpha_T$

TRENDS OF DIFFUSION MODELS

生成模型的演進

- 三足鼎立的局面
  - GANs (生成對抗網絡): 取樣極快 (單次前向)，但訓練不穩定，容易模式崩潰 (Mode Collapse)。
  - VAEs (變分自編碼器): 訓練穩定，取樣快，但生成的圖像通常較模糊。
  - Diffusion Models (擴散模型): 生成品質最高 (SOTA)，多樣性極佳，訓練穩定，但推論速度極慢。
- 生成三難困境 (Generative Trilemma)
  - 很難同時滿足以下三點：
  - 高品質 (High Quality)
  - 快速取樣 (Fast Sampling)
  - 模式多樣性 (Mode Diversity)

生成式 AI 的範式轉移

過去：GANs 的瓶頸

生成對抗網絡 (GANs) 雖然生成速度快，但面臨著根本性的挑戰：

- 訓練不穩定： 生成器與判別器的平衡難以維持。
- 模式崩潰 (Mode Collapse)： 模型傾向於生成重複的樣本，缺乏多樣性。

現在與未來：擴散主導

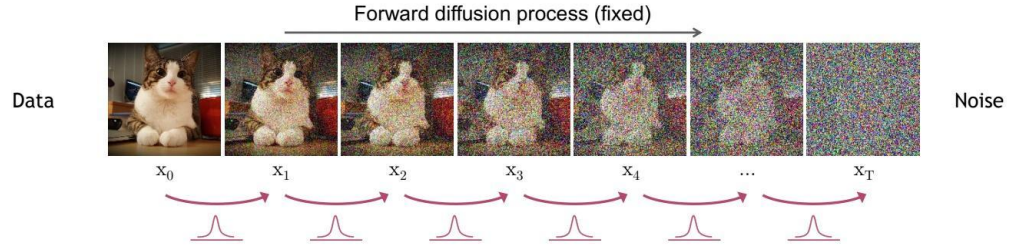
擴散模型 (Diffusion Models) 已成為當前的核心範式：

- 穩定性： 基於似然最大化與固定的去噪過程，訓練收斂更可靠。
- 趨勢 (2023-2025)： 從「可行性」走向「工業級效率」。

關鍵字：DiT, Flow Matching, Multimodal。

Ho et al. (2020) DDPM [arXiv:2006.11239]
Peebles & Xie (2022) DiT [arXiv:2212.09748]

Bottleneck of DDPM



- Why DDPM is slow for inference?
  - Operated on pixel space
    - High dimensionality, e.g. 512x512x3 = 786,000
    - => latent space: **latent diffusion** model
  - 1000s steps
    - Need small step for accurate backward process
    - => single step or few step?? Better solver**, flow matching, consistent model**

Latent Diffusion

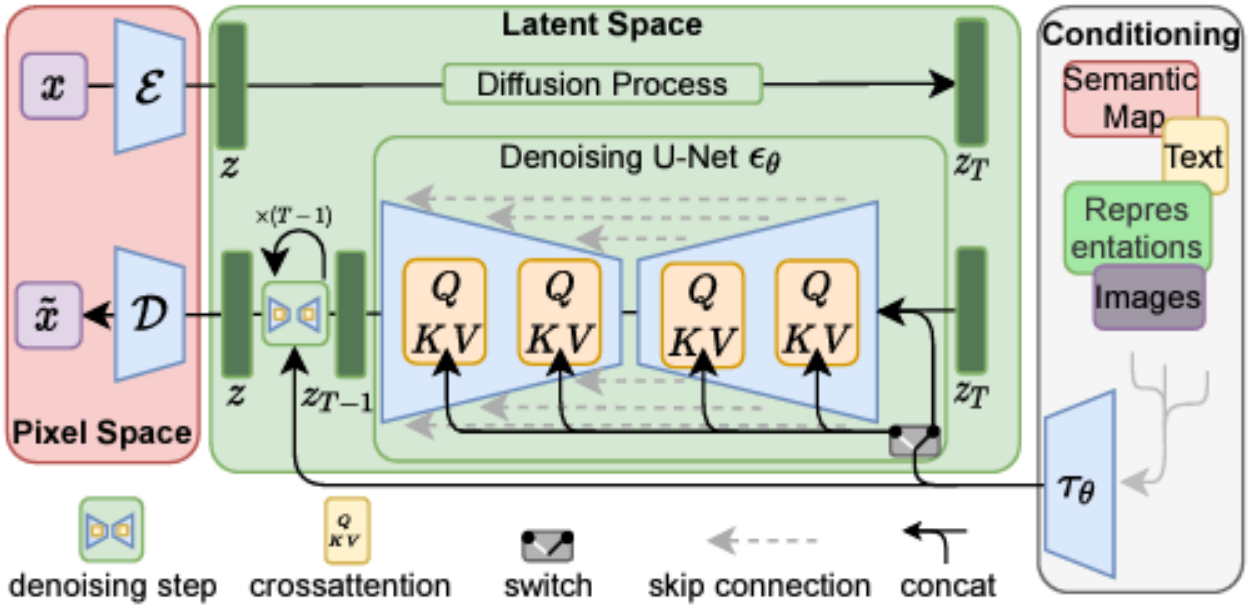
- 核心概念：感知壓縮 (Perceptual Compression)
- 圖像生成可以分解為兩個獨立階段：
  - 感知階段 (Perceptual):
    - 處理高頻細節 (紋理、邊緣)。這部分計算量大但語義訊息少。
  - 語義階段 (Semantic):
    - 處理圖像的結構與佈局。這才是擴散模型最擅長的部分。
- Solution
  - 利用一個預訓練的 Autoencoder (VQ-VAE 或 KL-VAE) 將圖像壓縮到低維潛在空間 (Latent Space)，在此空間進行擴散訓練。
  - 效益：
    - 對於 512x512 圖像，若下採樣係數 f=8，潛在空間僅為 64x64。計算量減少約 64 倍。

效率突破：潛在擴散模型 (LDM: latent diffusion model)

- 解決痛點：
  - 直接在像素空間 (Pixel Space) 處理高解析度圖像，算力成本過高且速度緩慢。
  - 學習很大一部分重點放在 pixel-space，相較難集中於 sematic 資訊。
- 感知壓縮 (Perceptual Compression)
  - **VAE 編碼器**： 將圖像壓縮至低維「潛在空間 (Latent Space)」，保留語義信息。
  - **潛在擴散Latent diffusion**： 在低維空間進行高效的去噪訓練。
  - **解碼器還原**： 生成後將潛在變量還原為高解析度圖像。
- Conditional input
  - 利用 **domain specific models 抽取 semantic 資訊後，再使用 attention 機制 [14] 與原本抽取圖像的 latent 結合。**

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]$$

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2\right]$$



Used in Stable Diffusion

https://arxiv.org/abs/2112.10752 High-Resolution Image Synthesis with Latent Diffusion Models

架構演進：從 U-Net 到 Transformer (DiT)

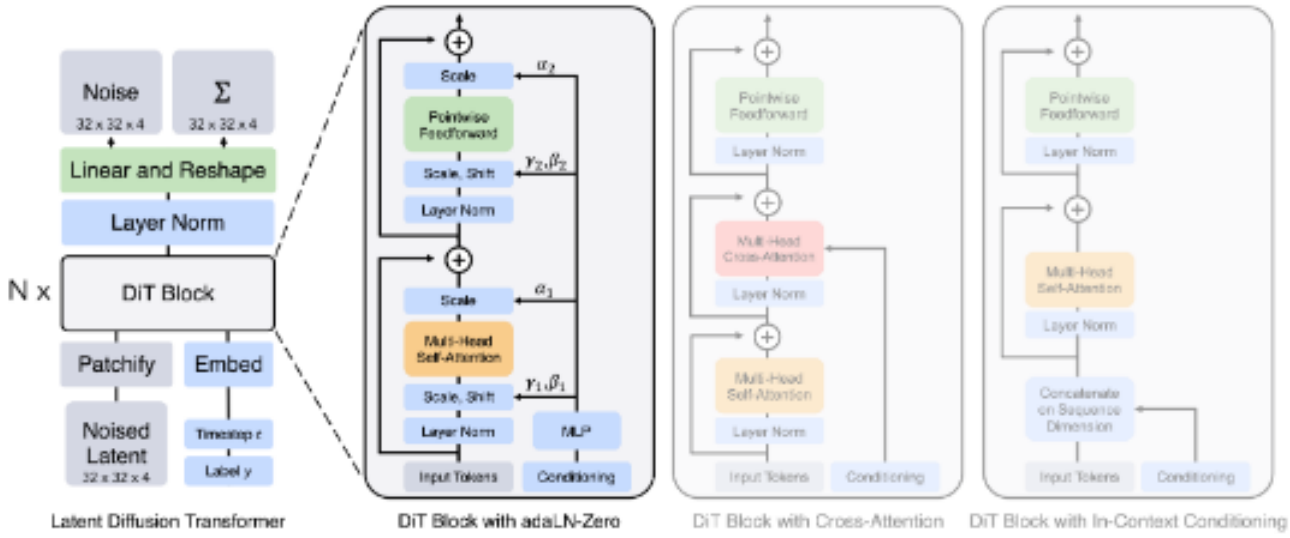
U-Net 的局限

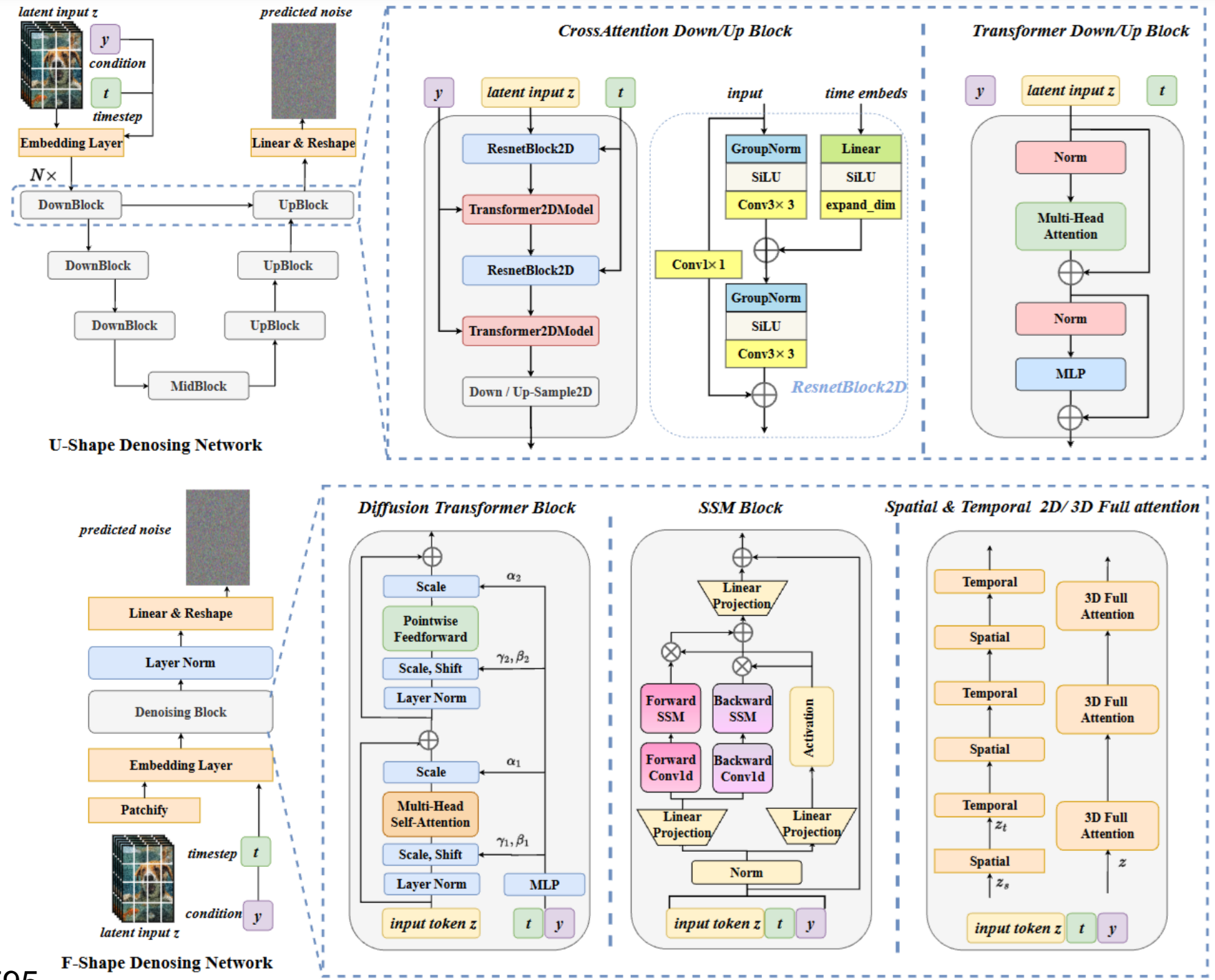
傳統卷積架構難以捕捉全局上下文 (Global Context)，且參數量擴展性 (Scalability) 有限。

Diffusion Transformer (DiT)

- Patchify： 將潛在特徵圖切分為 Patches，類似 ViT 的處理方式。
- Scaling Laws： 實驗證明，參數量越大，FID 分數越低（生成質量越好）。
- MM-DiT (SD3)： 雙流架構，獨立處理圖像與文本 Token，保留強大的語義理解能力。

DiT: Peebles & Xie (ICCV 2023) [arXiv:2212.09748]
SD3: Esser et al. (2024) [arXiv:2403.03206]





https://arxiv.org/abs/2410.11795

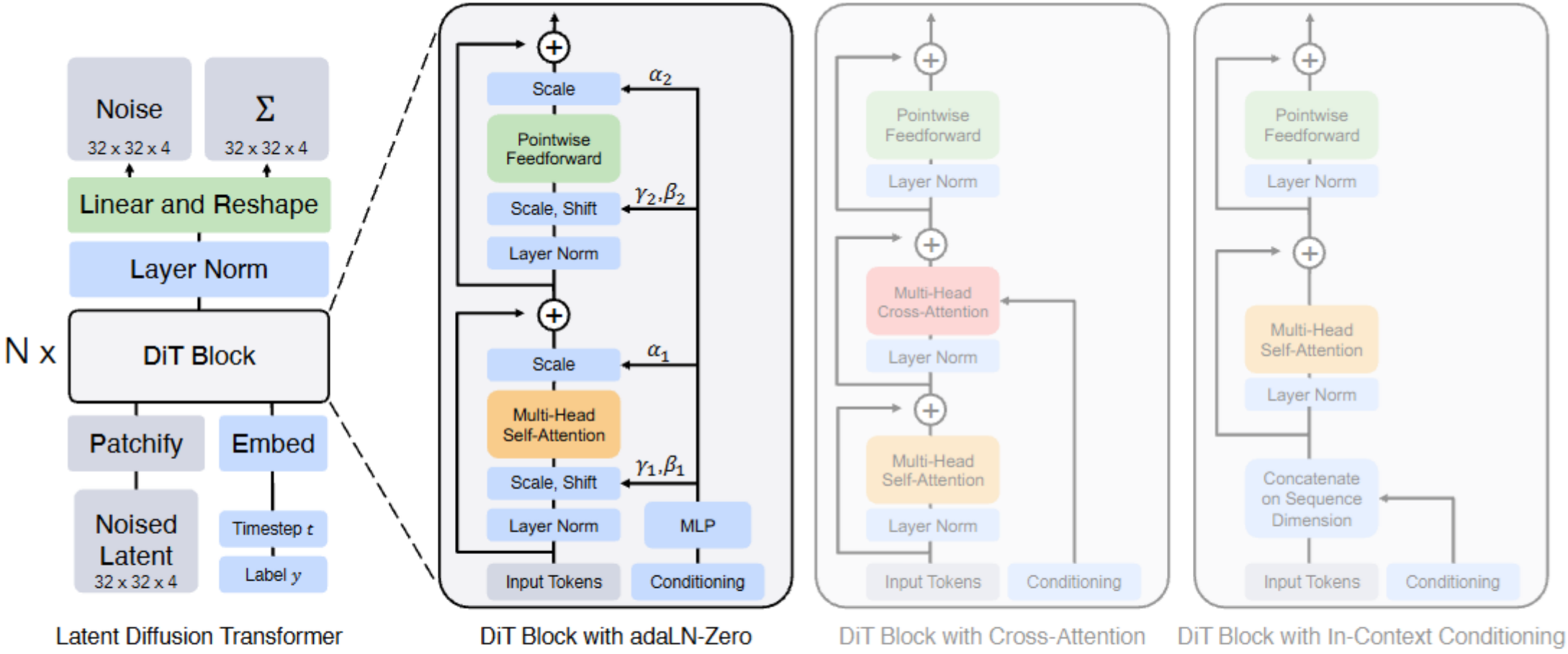
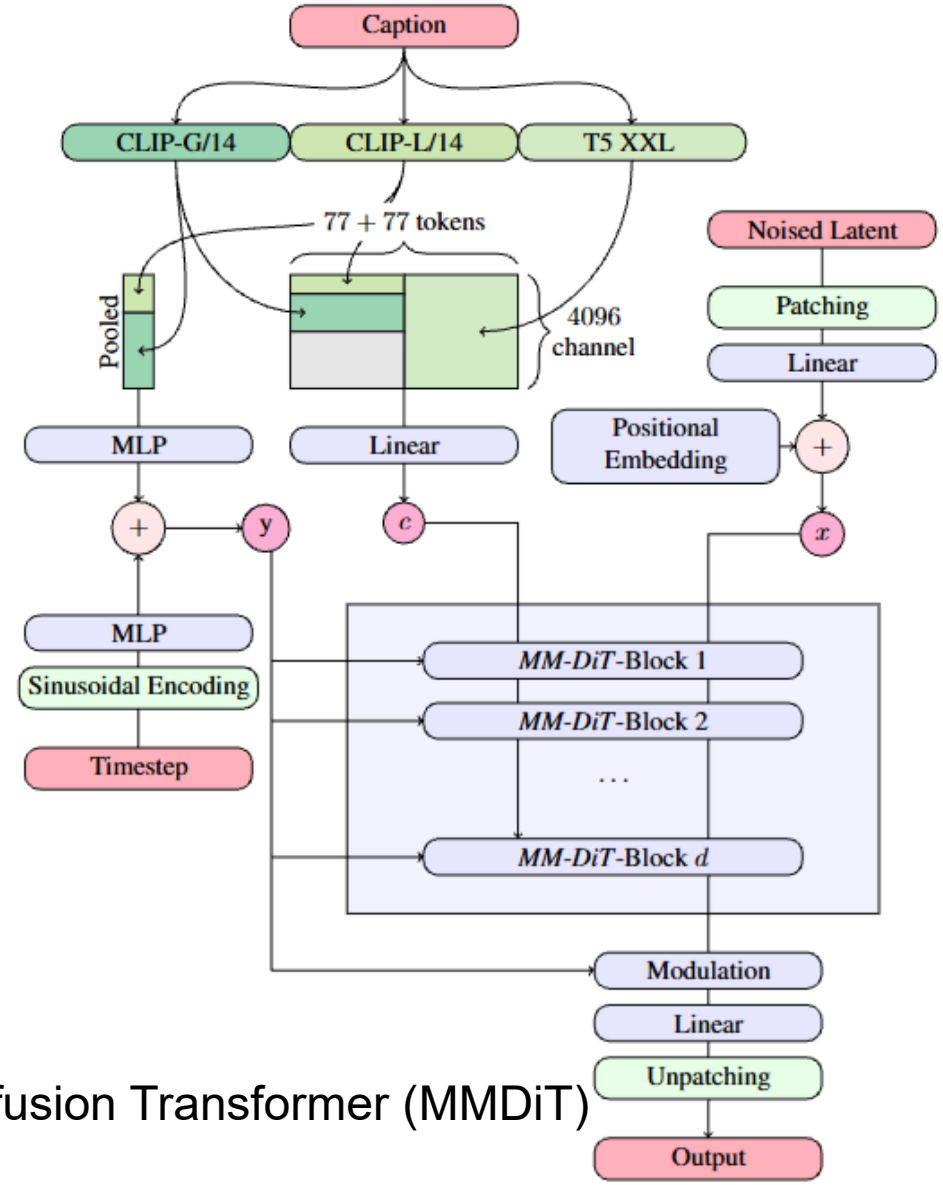
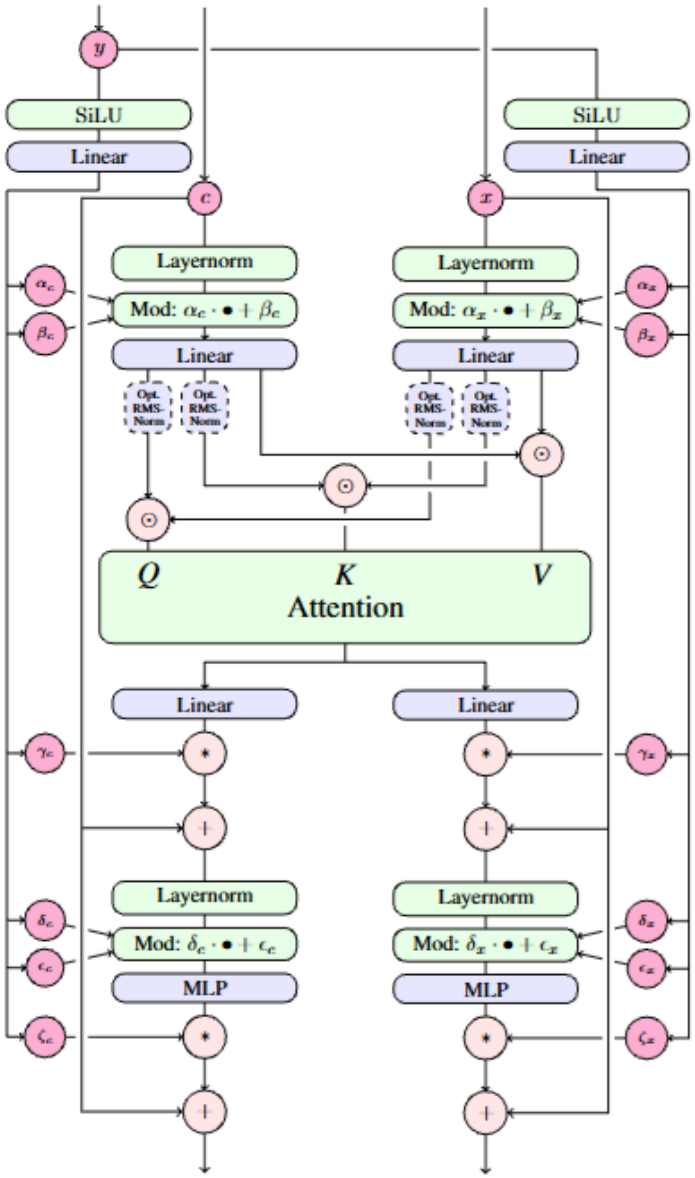


Figure 3. **The Diffusion Transformer (DiT) architecture.** *Left:* We train conditional latent DiT models. The input latent is decomposed into patches and processed by several DiT blocks. *Right:* Details of our DiT blocks. We experiment with variants of standard transformer blocks that incorporate conditioning via adaptive layer norm, cross-attention and extra input tokens. Adaptive layer norm works best.



(a) Overview of all components.

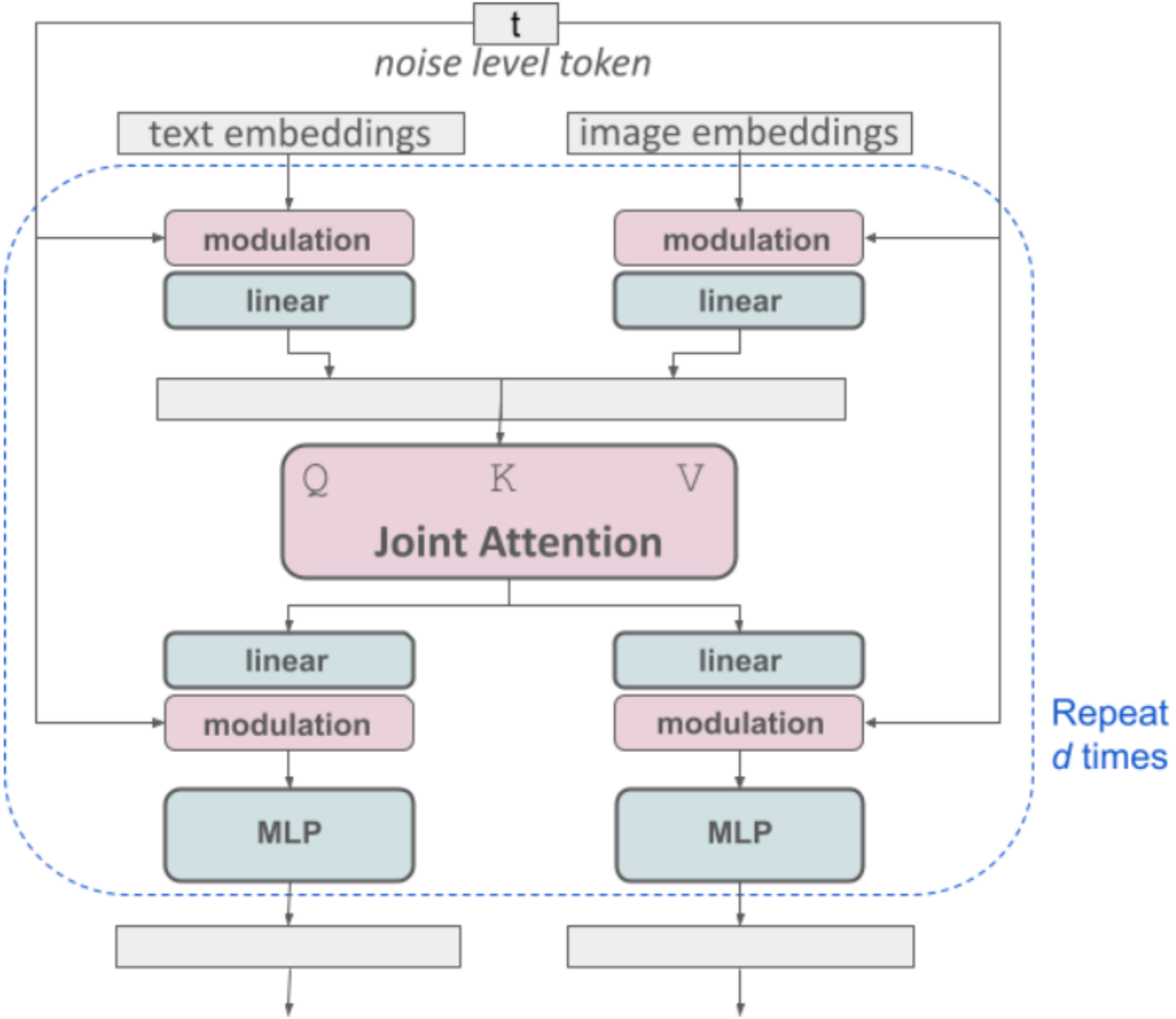


(b) One *MM-DiT* block

Figure 2. **Our model architecture.** Concatenation is indicated by $\odot$ and element-wise multiplication by $*$. The RMS-Norm for Q and K can be added to stabilize training runs. Best viewed zoomed in.

Multimodal Diffusion Transformer (MMDiT)

https://stability.ai/news/stable-diffusion-3-research-paper

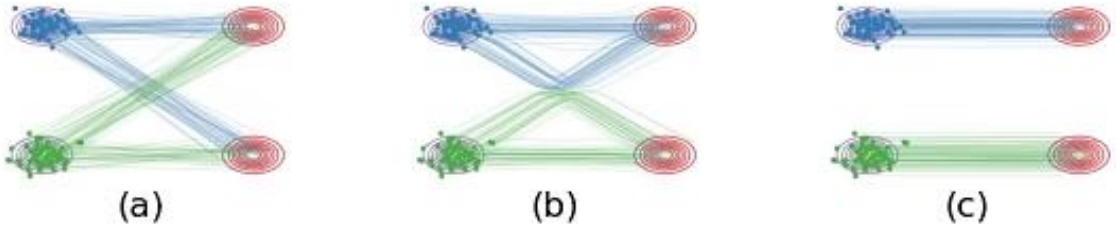

t
noise level token
text embeddings
image embeddings
modulation
linear
modulation
linear
Q
K
V
Joint Attention
linear
modulation
MLP
linear
modulation
MLP
Repeat
d times

加速與優化：Flow Matching

從隨機漫步到直線傳輸

傳統擴散路徑是彎曲且隨機的，導致採樣步數多。

- Rectified Flows (Flux.1, SD3)： 概念上將彎曲的擴散路徑「拉直」。建立噪聲到數據的最短直線路徑。
- 極速採樣： 這種線性化顯著減少了推理所需的步數，從 50 步降至 4-8 步即可獲得高質量結果。
- LCM： 另一種路徑，透過一致性蒸餾實現即插即用的 2-4 步生成。



DDPM

$$
\begin{aligned}
L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
&= \mathbb{E}_q\left[\log \frac{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^{T}\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T}\log \left(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\cdot\frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\right) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&= \mathbb{E}_q[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T}\underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]
\end{aligned}
$$

Rectified Flow

$$L = \int_0^1 \mathbb{E}_{\substack{X_0\sim\text{noise},\\ X_1\sim\text{data}}}[\|(X_1 - X_0) - v(X_t,t)\|^2]\,dt,$$

$$\text{with } X_t = t\,X_1 + (1-t)X_0$$

Rectified Flow: Liu et al. (ICLR 2023) [arXiv:2209.03003]

LCM: Luo et al. (2023) [arXiv:2310.04378]

https://arxiv.org/abs/2310.04378 https://arxiv.org/abs/2209.03003

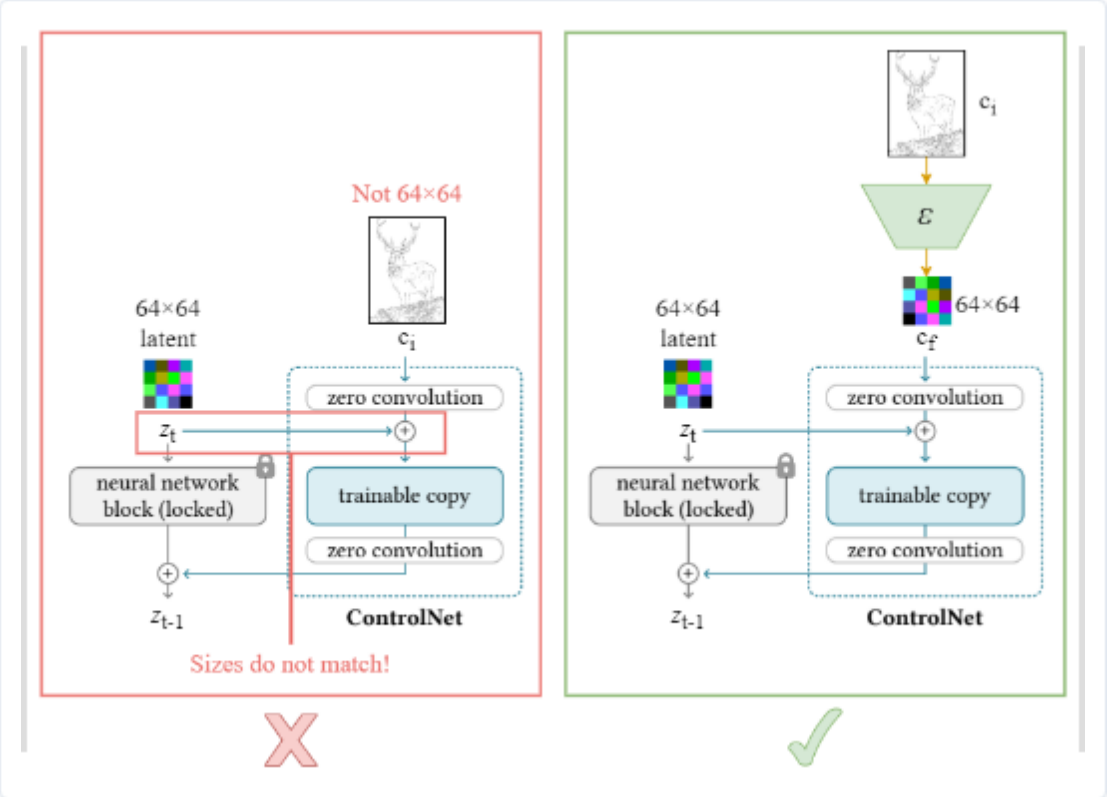
可控性精細化：ControlNet

精確控制空間結構

在不破壞大型預訓練模型能力的前提下，引入額外的控制條件。

- Zero Convolution： 使用零初始化的卷積層連接控制網絡，確保訓練初期不影響原模型。
- 副本鎖定機制： 鎖定原始模型的權重，僅訓練可訓練副本來學習條件控制（如邊緣檢測、骨架圖、深度圖）。
- IP-Adapter： 進一步解耦交叉注意力，完美融合「參考圖風格」與「文本描述」。

ControlNet: Zhang et al. (ICCV 2023) [arXiv:2302.05543]
IP-Adapter: Ye et al. (2023) [arXiv:2308.06721]



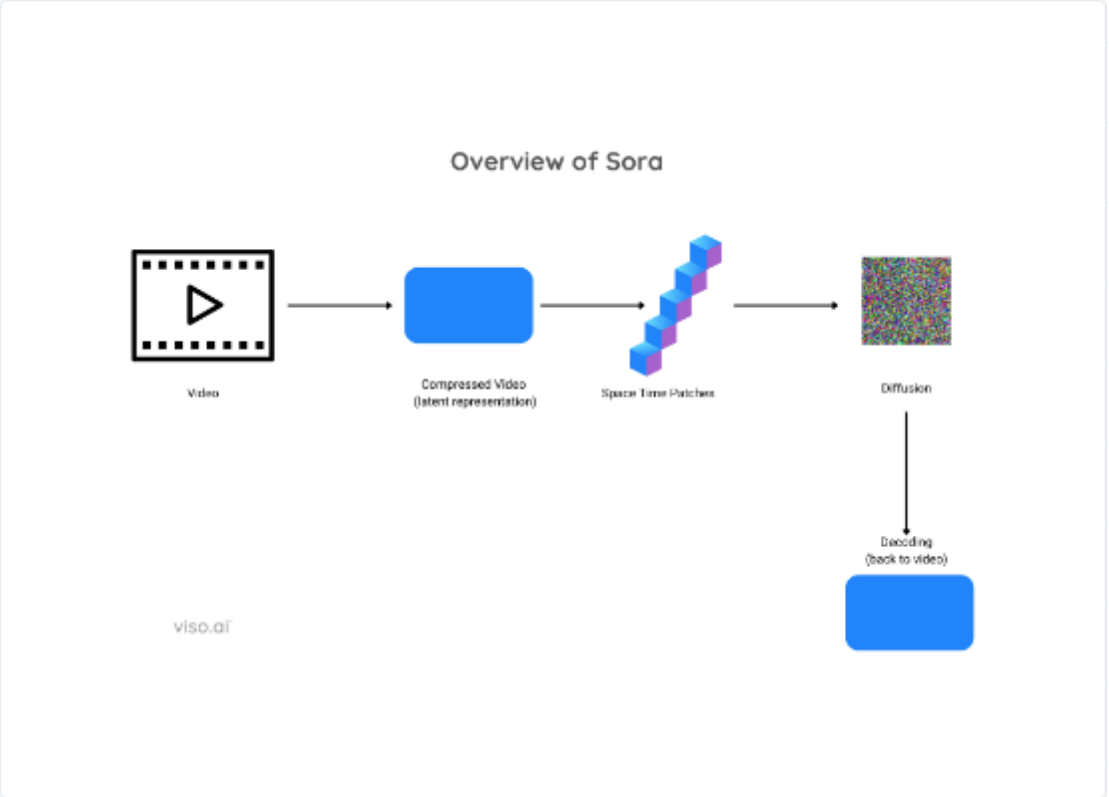
視頻生成：Sora 與時空一致性

Sora (Video DiT)

- 時空補丁 (Spacetime Patches)： 將視頻視為 3D 體積 (Volume)，切分為時空立方體，統一處理空間與時間維度
- 長距離依賴： 這種架構能有效捕捉視頻中跨越長時間跨度的物體恆常性。

Open-Sora / Latte

為了降低運算量，常採用分解注意力 (Factorized Attention)，將 3D Attention 拆解為「空間」與「時間」交替計算。



Sora: OpenAI Technical Report (2024)
Latte: Ma et al. (2024) [arXiv:2401.03048]

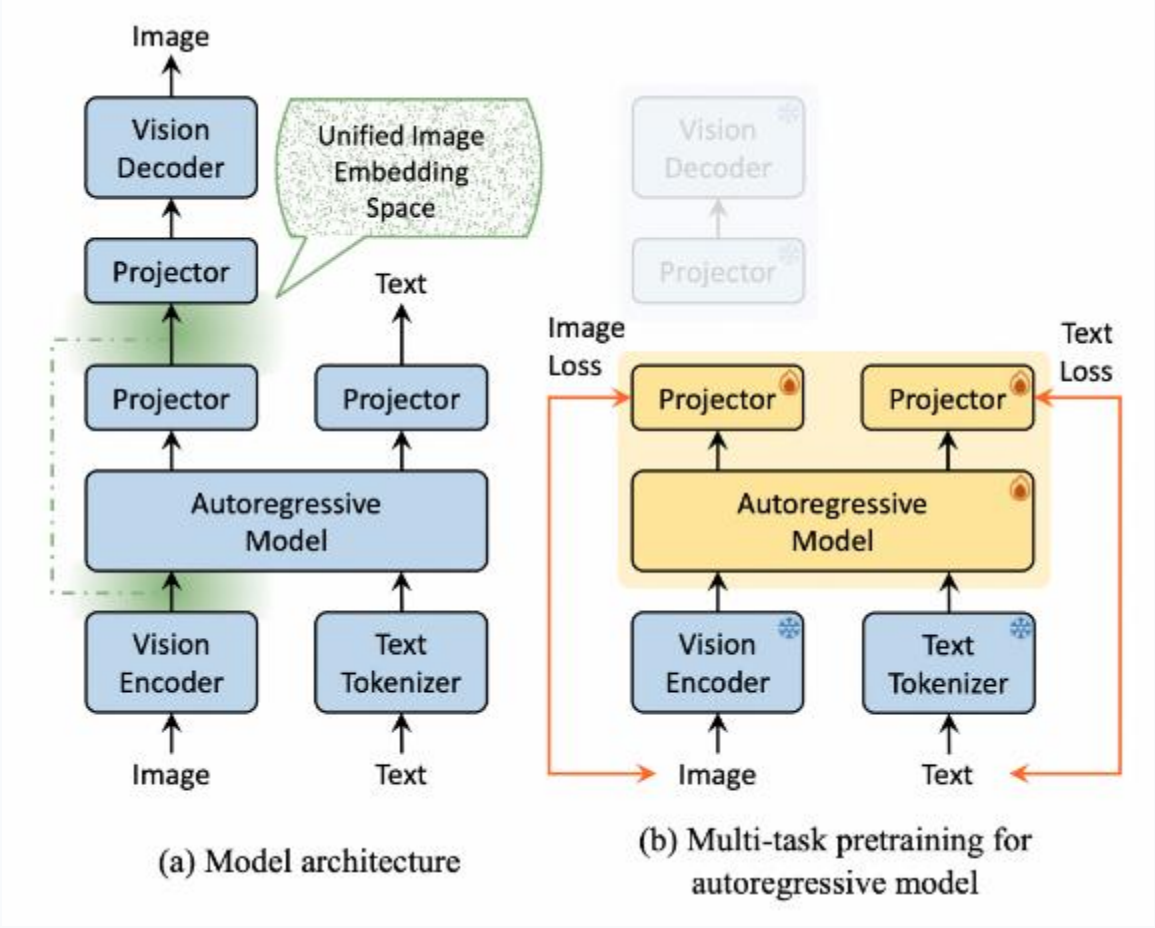
前沿趨勢：統一多模態模型

打破理解與生成的界限

未來的模型不再區分「理解模型 (如 GPT)」與「生成模型 (如 Stable Diffusion)」。

- 統一架構 (Janus, Show-o)： 在單一 Transformer 內同時處理自回歸 (文本/理解) 與 擴散 (圖像/生成)。
- 全能型輸入輸出： 任意模態輸入 (圖/文/音訊) -> 任意模態輸出。
- 3D 生成： 結合 Gaussian Splatting 與擴散先驗，解決生成速度與網格質量的問題。

Show-o: Xie et al. (2024) [arXiv:2408.12528]
Janus: Wu et al. (2024) [arXiv:2410.13848]

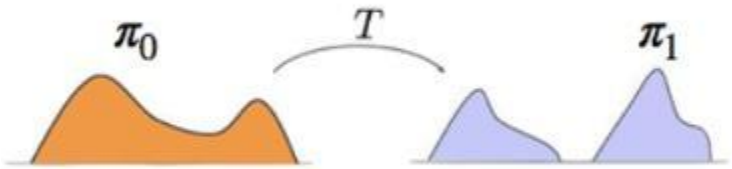


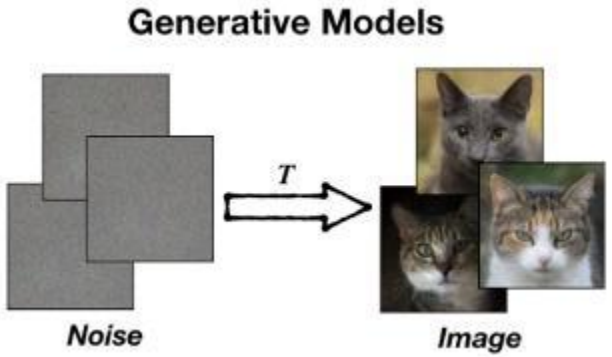
技術對照總結

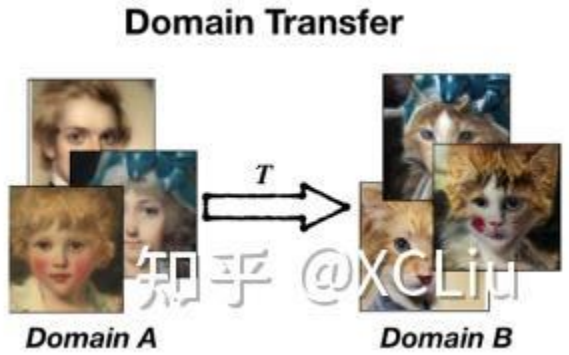
技術範式	核心機制	優勢與特性	代表模型
GAN	對抗訓練 (Generator vs Discriminator)	生成速度極快，但訓練極不穩定，易模式崩潰	StyleGAN, BigGAN
DDPM	馬爾可夫鏈去噪 (Markov Chain)	理論完備，生成質量高且多樣，但採樣極慢	DALL-E 2, Imagen
LDM	感知壓縮 + 潛在空間擴散	效率平衡點：大幅降低算力需求，保留高畫質	Stable Diffusion 1.5/XL
DiT	Vision Transformer 架構	Scaling Laws 發揮作用，更強的語義理解與擴展性	SD3, Sora
Flow Matching	最優傳輸路徑 (Optimal Transport)	極速生成：4-8 步採樣，畫質與速度兼得	Flux.1, SD3 Turbo

RECTIFIED FLOW (OPTIONAL)

問題-傳輸映射（將一個分佈搬運到另一個分佈）







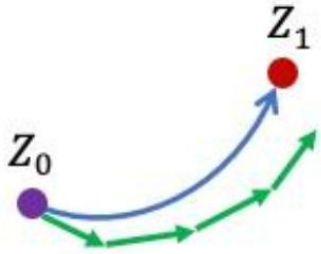
映射 T 是透過以下連續運動系統，也就是一個常微分方程(ordinary differential equation (ODE))，或叫流模型(flow)，來隱式定義

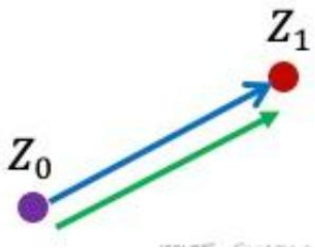
$$\frac{d}{dt}Z_t = v(Z_t, t)\ ,\ \ Z_0 \sim \pi_0\ ,\ \forall t \in [0,1].$$

NN

$$Z_{t+\epsilon} = Z_t + \epsilon v(Z_t, t)$$

Step size





藍色：真實 ODE 軌跡；綠色：Euler 法得到的離散軌跡。左：彎曲的運動軌跡需要較小的步長來離散化才能得到較小誤差，所以需要更多的步數；右：筆直的運動軌跡甚至可以在計算機裡用一步進行完美的模擬。

https://arxiv.org/abs/2209.03003

Rectified Flow-基於直線 ODE 學習生成模型

假設我們有從兩個分佈中的取樣 $X_0 \sim \pi_0$ ， $X_1 \sim \pi_1$ (例如 X_0 是從 π_0 出來的隨機噪聲， X_1 是一個隨機的資料(服從 π_1))。我們把 X_0 和 X_1 用一個線性內插法連結起來，得到

$$X_t = tX_1 + (1-t)X_0, \quad t \in [0,1].$$

現在，如果我們拿 X_t 對時間 t 求導，我們其實已經可以得到一個能夠將資料從 $X_0 \sim \pi_0$ 傳送到 $X_1 \sim \pi_1$ 的"ODE"了，

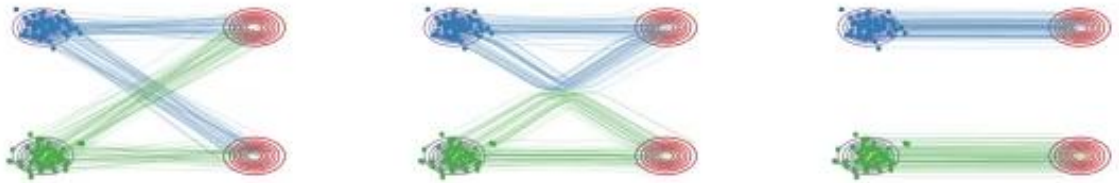
$$\frac{d}{dt}X_t = X_1 - X_0, \qquad \forall t \in [0,1].$$

但是，這個"ODE"並不實用而且很奇怪，所以要打個引號：它不是一個「因果」(causal)，或者「可前向模擬」(forward simulatable)的系統，因為要計算 X_t 在 t 時刻的速度 $(X_1 - X_0)$ 需要提前(在 $t < 1$ 軌道時知道 8OD}。如果我們都已經知道 X_1 了，那其實就沒有必要模擬 ODE 了。

那我們能不能學習 v，使得我們想要的「可前向模擬」的 ODE $\frac{d}{dt}Z_t = v(Z_t, t)$ 能盡可能逼近剛才這個「不可前向模擬」的過程呢？最簡單的方法就是最佳化 v 來最小化這兩個系統的速度函數（分別是 v 和 $X_1 - X_0$）之間的平方誤差：

$$\min_v \int_0^1 \mathbb{E}_{X_0 \sim \pi_0, X_1 \sim \pi_1} \left[||(X_1 - X_0) - v(X_t, t)||^2 \right] dt, \quad \text{where} \quad X_t = tX_1 + (1-t)X_0.$$

這是一個標準的最佳化任務。我們可以將 v 設定成一個神經網絡，並用隨機梯度下降或 Adam 來優化，進而得到我們的可模擬 ODE 模型。



圖(a)：當我們用直線連接 X_0 和 X_1 時，有些線會在中間的地方相交，這是導致 $\frac{d}{dt}X_t = X_1 - X_0$ 非因果的原因(在交叉點， X_t 既可以沿藍線走，也可以沿綠線走，因此粒子不知該向岔路的哪一邊走

圖(b)：我們學習出的 ODE 因為必須是因果的，所以不能出現道路相交的情況，它會在原來相交的地方把道路交換成不交叉的形式。這樣，我們學習出來的 ODE 仍然保留了原來的基本路徑，但做了一個重組來避免相交的情況。這樣的結果是，圖(a)和圖(b)裡的系統在每個時刻 t 的邊際分佈是一樣的，即使總體的路徑不一樣。

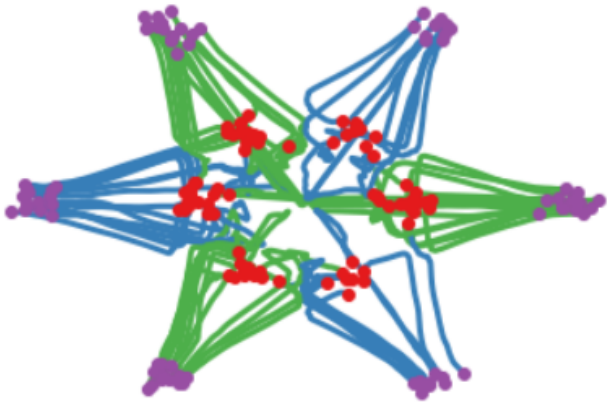
我們的方法起名為 Rectified Flow。這裡 rectified 是“拉直”，“規整”的意思。我們這個框架其實也可以用來推導解釋其他的擴散模型(如 DDPM)。我們論文裡有詳細說明，這裡就不贅述了。我們現在的演算法版本應該是在已知的演算法空間裡最簡單的選項了。我們提供了 Colab Notebook 來幫助大家透過實踐來理解這個過程。

Reflow-拉直軌跡，一步生成

因為 Rectified Flow 要在直線軌跡的交叉點做路徑重組，所以上面的 ODE 模型(或者說 flow)的軌跡仍然可能是彎曲的(如上面的圖(b))，不能達到一步生成。我們提出一個「Reflow」方法，將 ODE 的軌跡進一步變直。

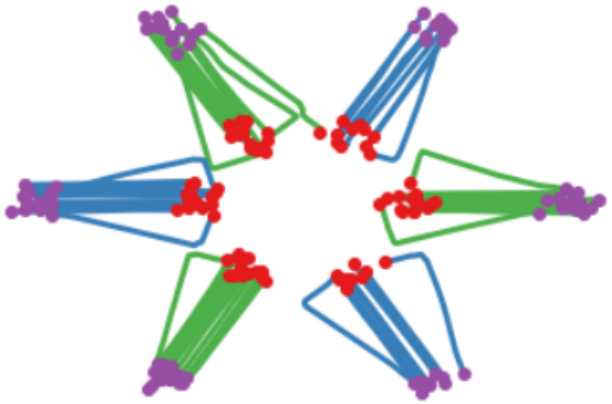
具體的做法非常簡單：假設我們從 π_0 取樣出一批 X_0 。然後，從 X_0 出發，我們模擬上面學出的 flow(叫它 1-Rectified Flow)，得到 $X_1 = \text{Flow}_1(X_0)$ 。我們用這樣得到的 (X_0, X_1) 對來學習一個新的"2-Rectified Flow"：

這裡，2-Rectified Flow 和 1-Rectified Flow 在訓練過程中唯一的區別是資料配對不同：在 1-Rectified Flow 中， X_0 與 X_1 是隨機或任意配對的；在 2-Rectified Flow 中，X_0 與 X_1 是透過 Flow1-Rectified Flow 的配對。上面的動圖中，圖(c)展示了 Reflow 的效果。因為從 1-Rectified Flow 出來的 (X_0, X_1) 已經有很好的配對，他們的直線插值交叉數減少，所以 2-Rectified Flow 的軌跡也就（比起 1-Rectified Flow）變得很直了(雖然仔細看還不完美)。理論上，我們可以重複 Reflow 多次，從而得到 3-Rectified Flow, 4-Rectified Flow... 我們可以證明這個過程其實是在單調地減小最優傳輸理論中的傳輸代價(transport cost)，而且最終收斂到完全直的狀態。當然，實際上，因為每次 v 優化得不完美，多次 Reflow 會累積誤差，所以我們不建議做太多次的 Reflow。幸運的是，在我們的實驗中，我們發現對生成圖片和許多我們感興趣的問題而言，像上面的圖(c)一樣，1 次 Reflow 已經可以得到非常直的軌跡了，配合蒸餾足夠達到一步生成的效果了。



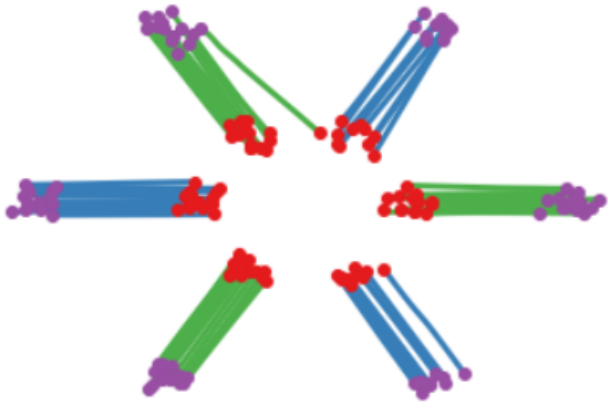
(a)The 1st rectified flow Z^1

$Z^1 = \texttt{RectFlow}((X_0, X_1))$



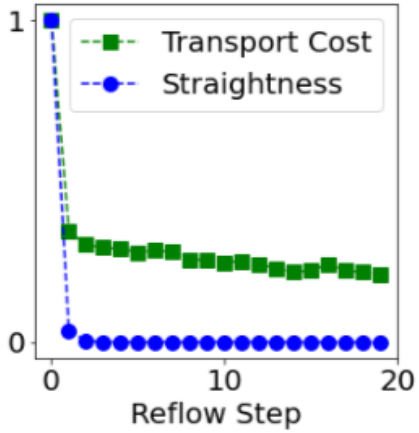
(b) Reflow Z^2

$Z^2 = \texttt{RectFlow}((Z_0^1, Z_1^1))$



(c) Reflow Z^3

$Z^3 = \texttt{RectFlow}((Z_0^2, Z_1^2))$



(d) Transport cost, Straightness

Method	NFE(↓)	IS (↑)	FID (↓)	Recall (↑)
ODE	*One-Step Generation (Euler solver, N=1)*			
1-Rectified Flow (+*Distill*)	1	1.13 (***9.08***)	378 (*6.18*)	0.0 (*0.45*)
2-Rectified Flow (+*Distill*)	1	8.08 (*9.01*)	12.21 (***4.85***)	0.34 (*0.50*)
3-Rectified Flow (+*Distill*)	1	8.47 (*8.79*)	8.15 (*5.21*)	0.41 (***0.51***)
VP ODE [73] (+*Distill*)	1	1.20 (*8.73*)	451 (*16.23*)	0.0 (*0.29*)
sub-VP ODE [73] (+*Distill*)	1	1.21 (*8.80*)	451 (*14.32*)	0.0 (*0.35*)
ODE	*Full Simulation (Runge–Kutta (RK45), Adaptive N)*			
1-Rectified Flow	127	**9.60**	**2.58**	**0.57**
2-Rectified Flow	110	9.24	3.36	0.54
3-Rectified Flow	104	9.01	3.96	0.53
VP ODE [73]	140	9.37	3.93	0.51
sub-VP ODE [73]	146	9.46	3.16	0.55
SDE	*Full Simulation (Euler solver, N=2000)*			
VP SDE [73]	2000	9.58	2.55	0.58
sub-VP SDE [73]	2000	9.56	2.61	0.58

(a) Results using the DDPM++ architecture.

Method	NFE(↓)	IS (↑)	FID (↓)	Recall (↑)
GAN	*One-Step Generation*			
SNGAN [52]	1	8.22	21.7	0.44
StyleGAN2 [28]	1	9.18	8.32	0.41
StyleGAN-XL [66]	1	-	1.85	0.47
StyleGAN2 + ADA [28]	1	9.40	2.92	0.49
StyleGAN2 + DiffAug [98]	1	9.40	5.79	0.42
TransGAN + DiffAug [26]	1	9.02	9.26	0.41
GAN with U-Net	*One-step Generation*			
TDPM (T=1) [99]	1	8.65	8.91	0.46
Denoising Diffusion GAN (T=1) [91]	1	8.93	14.6	0.19
ODE	*One Step Generation (Euler solver, N=1)*			
DDIM Distillation [47]	1	8.36	9.36	0.51
NCSN++ (VE ODE) [73] (+*Distill*)	1	1.18 (*2.57*)	461 (*254*)	0.0 (*0.0*)
ODE	*Full Simulation (Runge–Kutta (RK45), Adaptive N)*			
NCSN++ (VE ODE) [73]	176	9.35	5.38	0.56
SDE	*Full Simulation (Euler solver)*			
DDPM [23]	1000	9.46	3.21	0.57
NCSN++ (VE SDE) [73]	2000	9.83	2.38	0.59

(b) Recent results with different architectures reported in literature.

Table 1: Results on CIFAR10 unconditioned image generation. Fréchet Inception Distance (FID) and Inception Score (IS) measure the quality of the generated images, and recall score [38] measures diversity. The number of function evaluation (NFE) denotes the number of times we need to call the main neural network during inference. It coincides with the number of discretization steps N for ODE and SDE models.

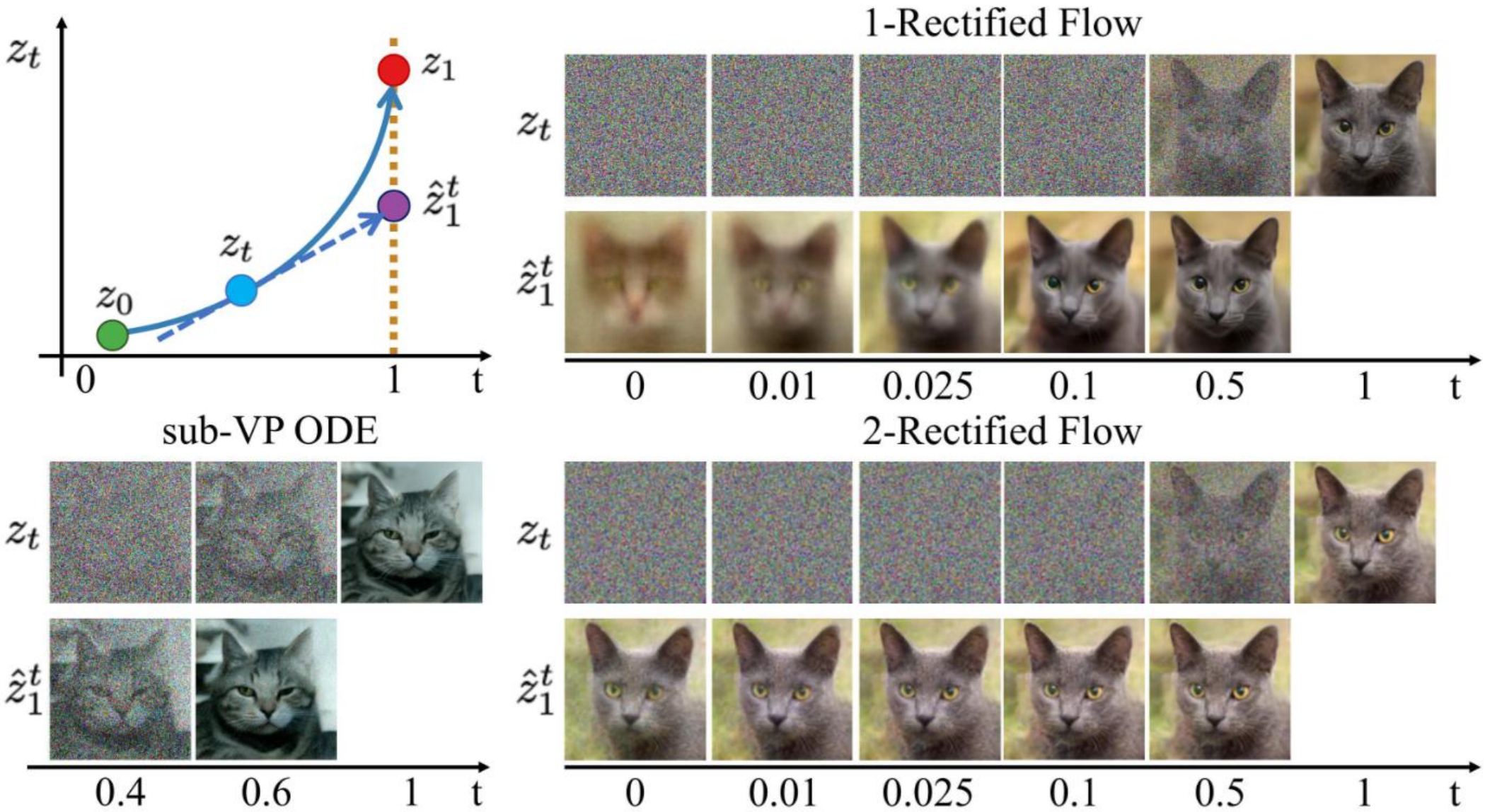


Figure 10: Sample trajectories z_t of different flows on the AFHQ Cat dataset, and the extrapolation $\hat{z}_1^t = z_t + (1 - t)v(z_t, t)$ from different z_t. The same random seed is adopted for all three methods. The $\hat{z}_1^t$ of 2-rectified flow is almost independent with t, indicating that its trajectory is almost straight.

作者也提出了 2-Rectified，即訓練時不隨機選取點對 (x_0, x_T)，而是利用 1-Rectified 的模型來產生訓練的點對，對軌跡進行修正。

從證明過程也可以看出來，Rectified Flow 只對 x_t, x_T 為單調關係時才適用，但目前的擴散模型幾乎都為直線單調式的，所以也不太影響模型的適用範圍。

由於是隨機選取點，進行分佈到分佈的映射，因此實驗上對輪數的要求會比較高，對應硬體資源需求也會比較高。

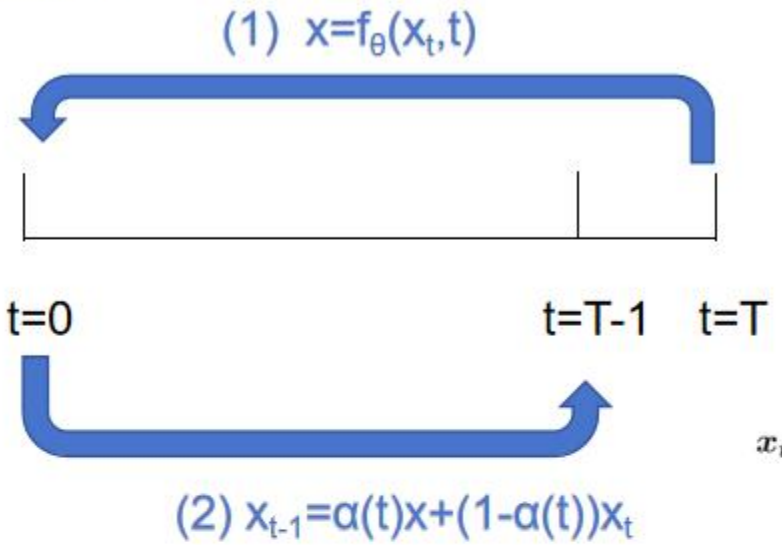
相較於之前的 Flow Matching，Rectified Flow 的形式更加簡單，不需要考慮前向過程，直接學習分佈之間的映射，一步到位。

關於 Rectified Flow 更細節的地方可以參考最新的《 Diffusion 學習筆記（二十） - 深入理解 Rectified Flow，完善統一擴散框架》。

方法介绍

- 推理过程

DDPM/DDIM



Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

$$x_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(x_t)}{\sqrt{\alpha_t}} \right)}_{\text{“predicted } x_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(x_t)}_{\text{“direction pointing to } x_t\text{”}}$$

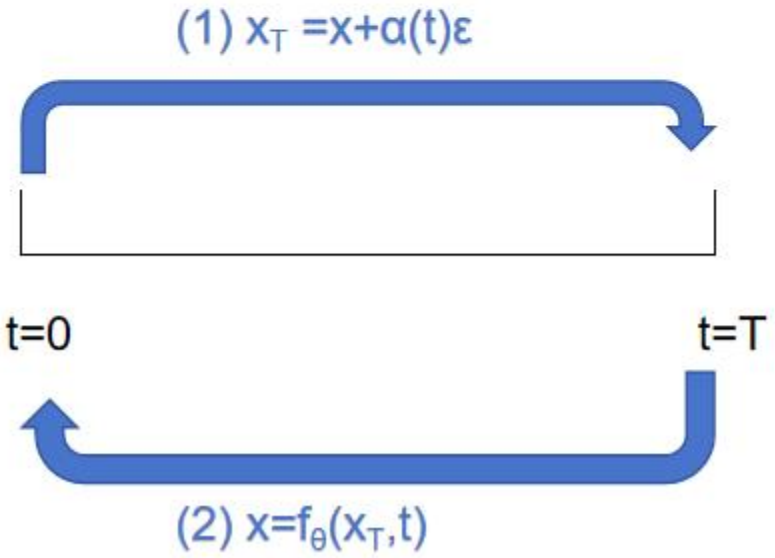
Consistency Models

$$f : (\mathbf{x}_t, t) \mapsto \mathbf{x}_\epsilon$$

$$f(\mathbf{x}_\epsilon, \epsilon) = \mathbf{x}_\epsilon$$

$$f(\mathbf{x}_t, t) = f(\mathbf{x}_{t'}, t')$$

$$f_\theta(\mathbf{x}, t) = \begin{cases} \mathbf{x} & t = \epsilon \\ F_\theta(\mathbf{x}, t) & t \in (\epsilon, T] \end{cases}$$



Algorithm 1 Multistep Consistency Sampling

Input: Consistency model $f_\theta(\cdot,\cdot)$, sequence of time points $\tau_1 > \tau_2 > \cdots > \tau_{N-1}$, initial noise $\hat{\mathbf{x}}_T$
$\mathbf{x} \leftarrow f_\theta(\hat{\mathbf{x}}_T, T)$
for $n = 1$ **to** $N-1$ **do**
  Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\hat{\mathbf{x}}_{\tau_n} \leftarrow \mathbf{x} + \sqrt{\tau_n^2 - \epsilon^2}\mathbf{z}$ random x_T
  $\mathbf{x} \leftarrow f_\theta(\hat{\mathbf{x}}_{\tau_n}, \tau_n)$ predicted x_0
end for
Output: $\mathbf{x}$

方法介绍

- 训练方法

DDPM

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Consistency Models

Algorithm 2 Consistency Distillation (CD)

Input: dataset $\mathcal{D}$, initial model parameter θ, learning rate η, ODE solver $\Phi(\cdot, \cdot; \phi)$, $d(\cdot, \cdot)$, $\lambda(\cdot)$, and μ
$\theta^- \leftarrow \theta$ ODE Solver是已经训好的Diffusion模型
repeat
随机采样，步数采样 Sample $\mathbf{x} \sim \mathcal{D}$ and $n \sim \mathcal{U}[1, N-1]$
噪声由采样N+1 Sample $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \boldsymbol{I})$
Diffusion估计的N $\hat{\mathbf{x}}^{\phi}_{t_n} \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \phi)$
$\mathcal{L}(\theta, \theta^-; \phi) \leftarrow$
N的去噪等于N+1的去噪 $\lambda(t_n) d(\boldsymbol{f}_\theta(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\theta^-}(\hat{\mathbf{x}}^{\phi}_{t_n}, t_n))$
$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta, \theta^-; \phi)$
EMA移动平均 $\theta^- \leftarrow \text{stopgrad}(\mu\theta^- + (1-\mu)\theta)$
until convergence

Algorithm 3 Consistency Training (CT)

Input: dataset $\mathcal{D}$, initial model parameter θ, learning rate η, step schedule $N(\cdot)$, EMA decay rate schedule $\mu(\cdot)$, $d(\cdot, \cdot)$, and $\lambda(\cdot)$
$\theta^- \leftarrow \theta$ and $k \leftarrow 0$
repeat
随机采样，步数采样 Sample $\mathbf{x} \sim \mathcal{D}$, and $n \sim \mathcal{U}[1, N(k)-1]$
噪声采样 Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$\mathcal{L}(\theta, \theta^-) \leftarrow$
N的去噪等于N+1的去噪 $\lambda(t_n) d(\boldsymbol{f}_\theta(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), \boldsymbol{f}_{\theta^-}(\mathbf{x} + t_n \mathbf{z}, t_n))$
$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta, \theta^-)$
$\theta^- \leftarrow \text{stopgrad}(\mu(k)\theta^- + (1-\mu(k))\theta)$
EMA移动平均 $k \leftarrow k + 1$
until convergence