Diffusion Models

Jinseong Park

Korea Institute for Advanced Study (KIAS), AI Fellow

The 17th KIAS CAC Summer School

2026.06.24

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

About Me

Jinseong Park

- Korea Institute for Advanced Study (KIAS), Center for AI and Natural Sciences (CAINS)
  - **AI Fellow (AI Assistant Professor)** (26.03 - **Present**)
  - AI Research Fellow (25.03 - 26.02)
- University of British Columbia (UBC), Department of Computer Science
  - Postdoctoral Fellow (25.06-25.07, Visiting Canada)

Education

- B.S. in Industrial and Management Engineering, POSTECH
- M.S. / Ph.D. in Industrial Engineering, Seoul National University

Research topic

- **Understand AI** and develop **Secure** (**Safe** and **Private**) **AI**
- **Trustworthy** Issues in **Generative AI**

Homepage

Contents

Introduction

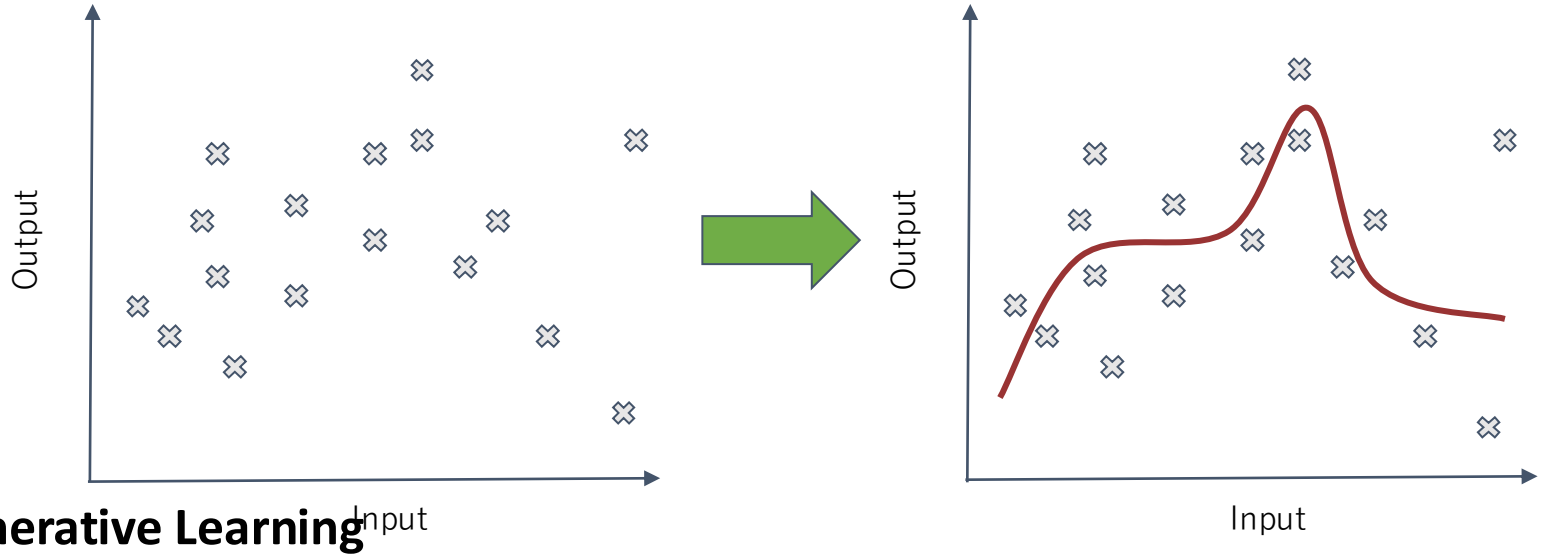
Reference

Tutorials and Lectures

- [CVPR26-tutorial] the-principles-of-diffusion-models
- CVPR 2022 Tutorial: Denoising Diffusion-based Generative Modeling
- NeurIPS 2024 Tutorial: Flow Matching for Generative Modeling
- What are Diffusion Models?
  - https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- MIT EECS6.S183: A Practical Introduction to Diffusion Models, 2026
- **Useful slides from Jaewoong Choi (SKKU Statistics, KIAS CAINS Alumni)**
  - https://sites.google.com/view/jaewoongchoi/home

Generative Models

Classical Supervised Learning



Generative Learning

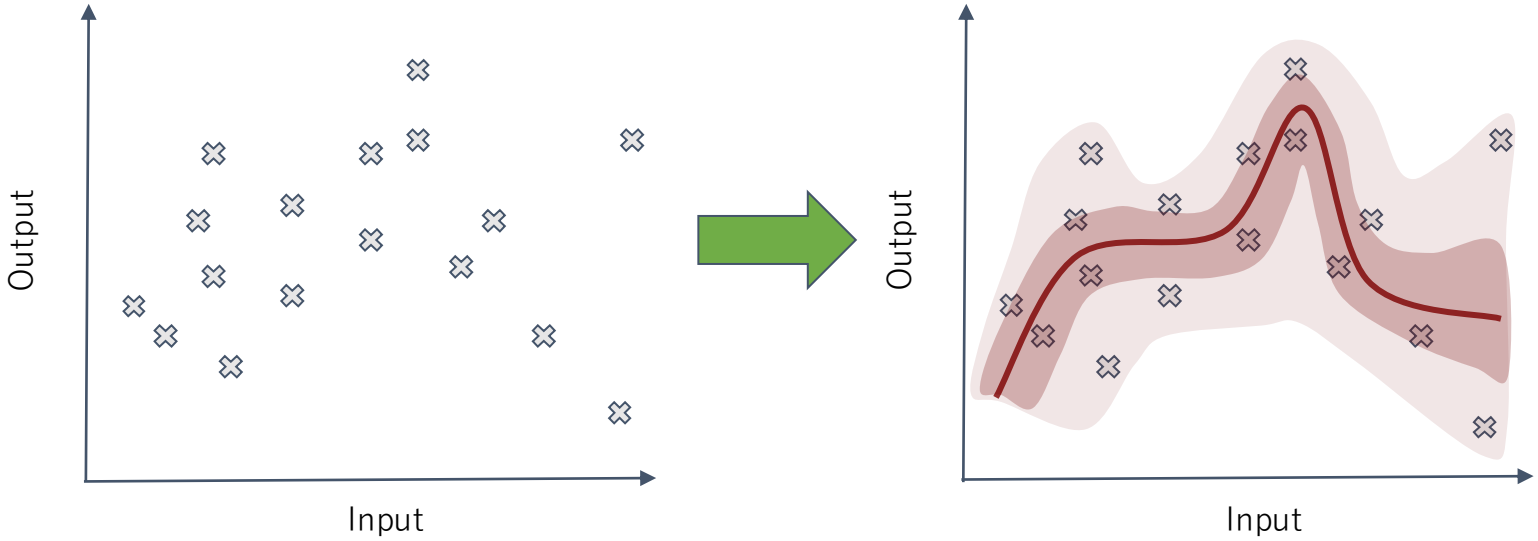
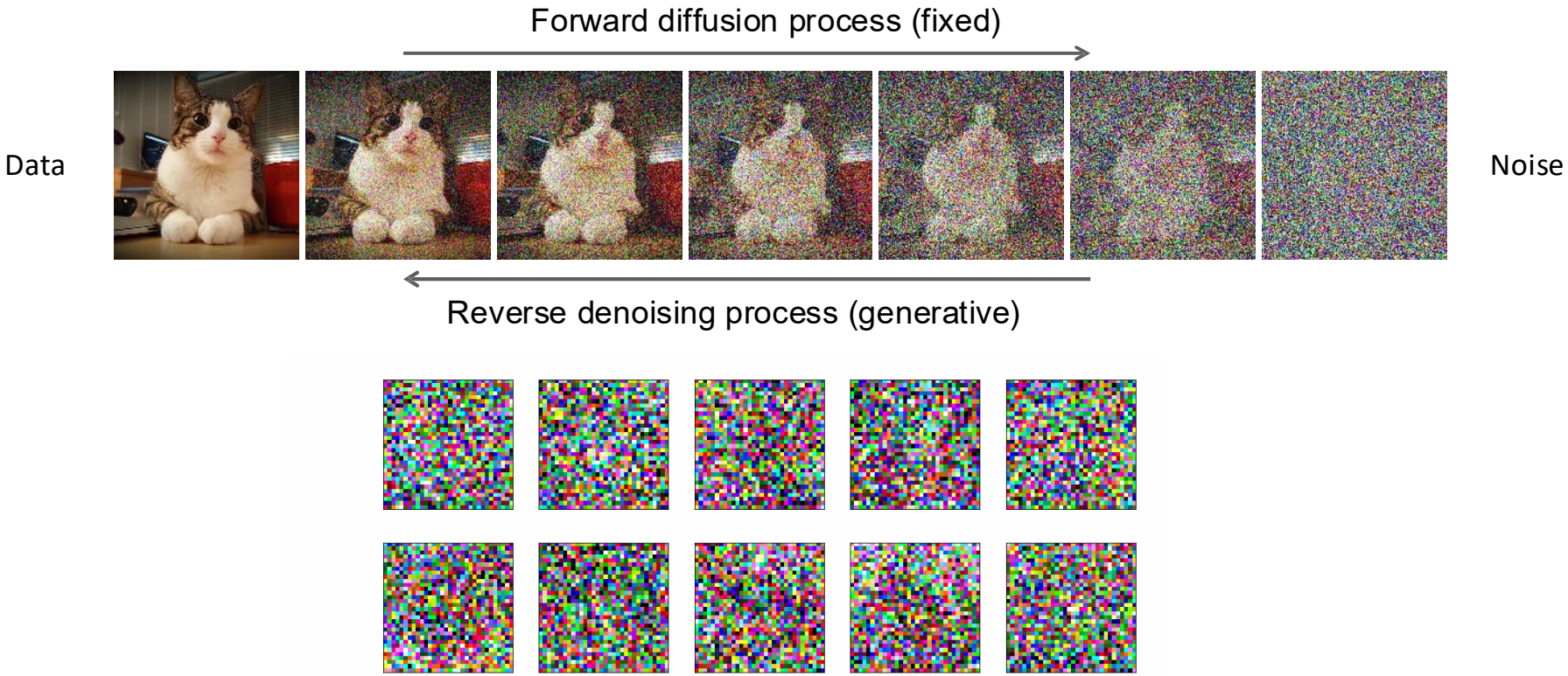


Image credit: SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis https://arxiv.org/pdf/2307.01952

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



Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Many-to-one: Text-to-image generation

Image credit: https://github.com/meanna/CVAE_CONV

Regression-based Model

Image credit: SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis https://arxiv.org/pdf/2307.01952

Latent Diffusion Model

Sketch-to-Image: coarse-grained control

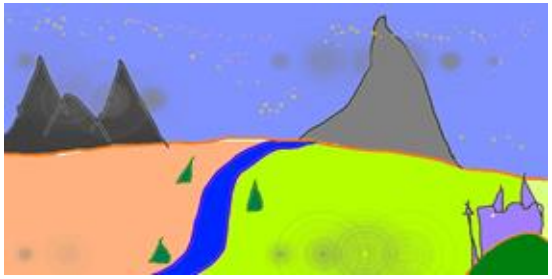


Image credit: https://github.com/CompVis/stable-diffusion

Image editing and composition

Figure 1. With just a few images (typically 3-5) of a subject (left), *DreamBooth*—our AI-powered photo booth—can generate a myriad of images of the subject in different contexts (right), using the guidance of a text prompt. The results exhibit natural interactions with the environment, as well as novel articulations and variation in lighting conditions, all while maintaining high fidelity to the key visual features of the subject.

Image credit: DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation https://arxiv.org/pdf/2208.12242

Diffusion “prior” for image restoration

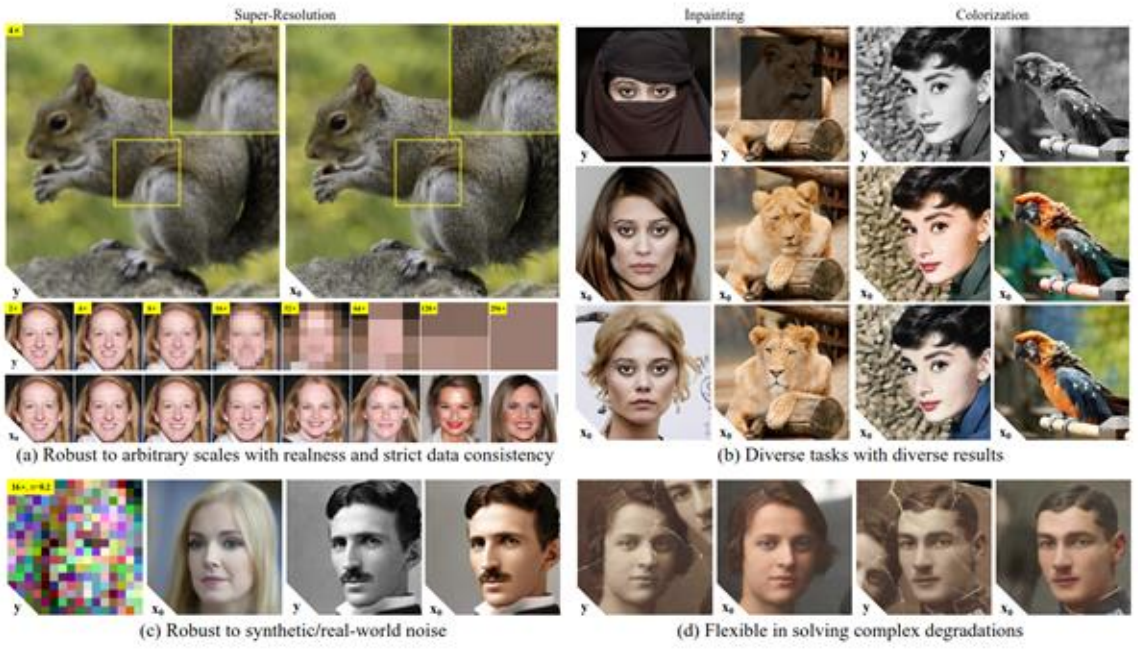


Image credit: Zero-Shot Image Restoration Using Denoising Diffusion Null-Space Model https://arxiv.org/pdf/2212.00490

Molecule Generation

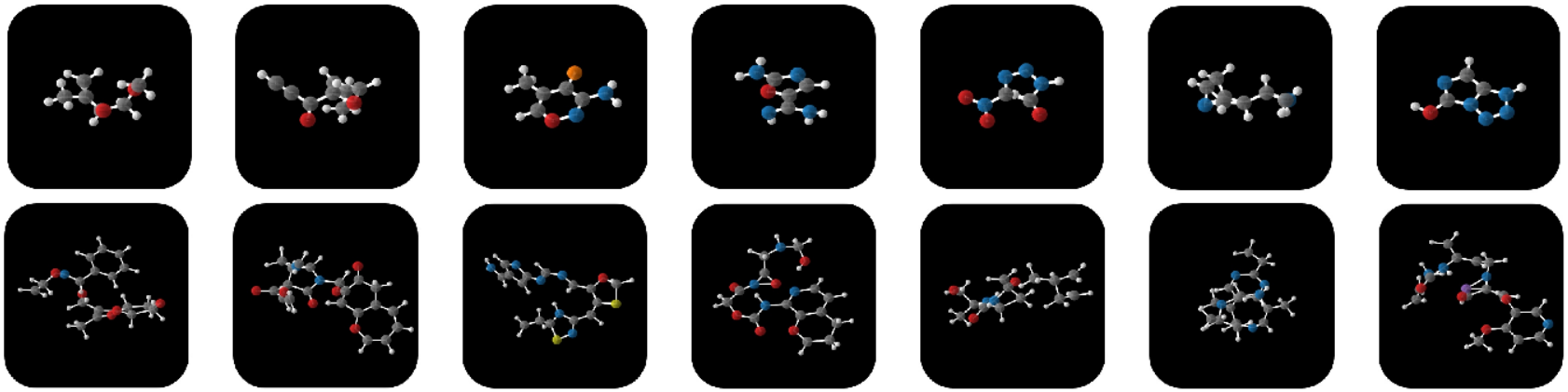
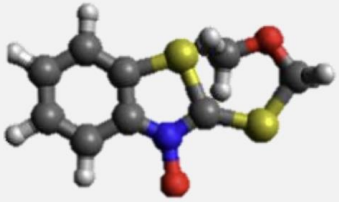
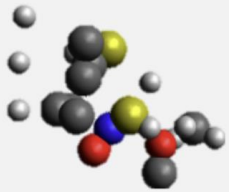


Figure 3. Selection of samples generated by the denoising process of our EDM trained on QM9 (up) and GEOM-DRUGS (down).

$E($  $) = 0.1$ $E($  $) = 100$

low energy → **stable** structure → **likely** to appear → **high** probability

high energy → **unstable** structure → **unlikely** to appear → **low** probability

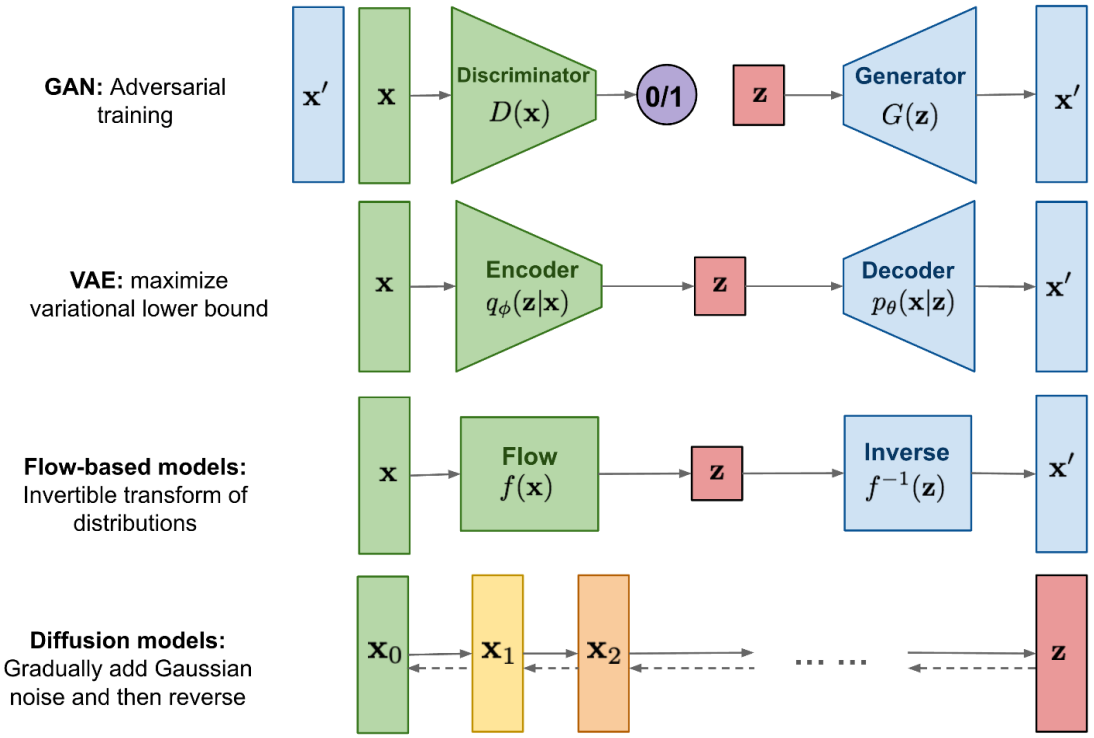
Contents

Generative Models

Generative Models

Generative models

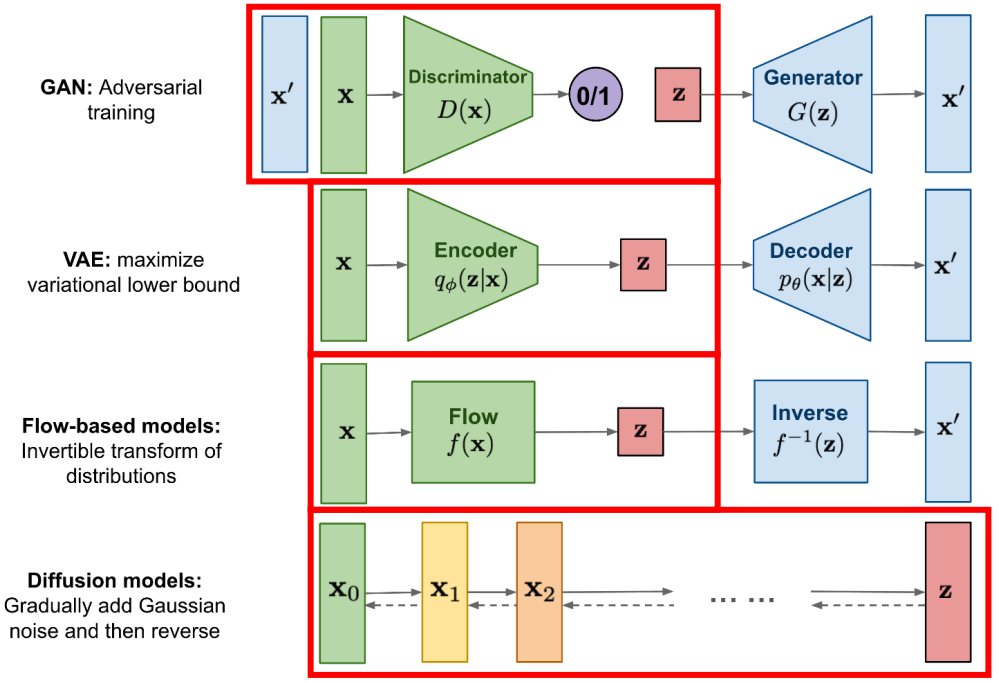
- Generate images, audios, time series, etc.
- GAN, VAE, Flow-based models, Diffusion models



Generative Models

Generative models

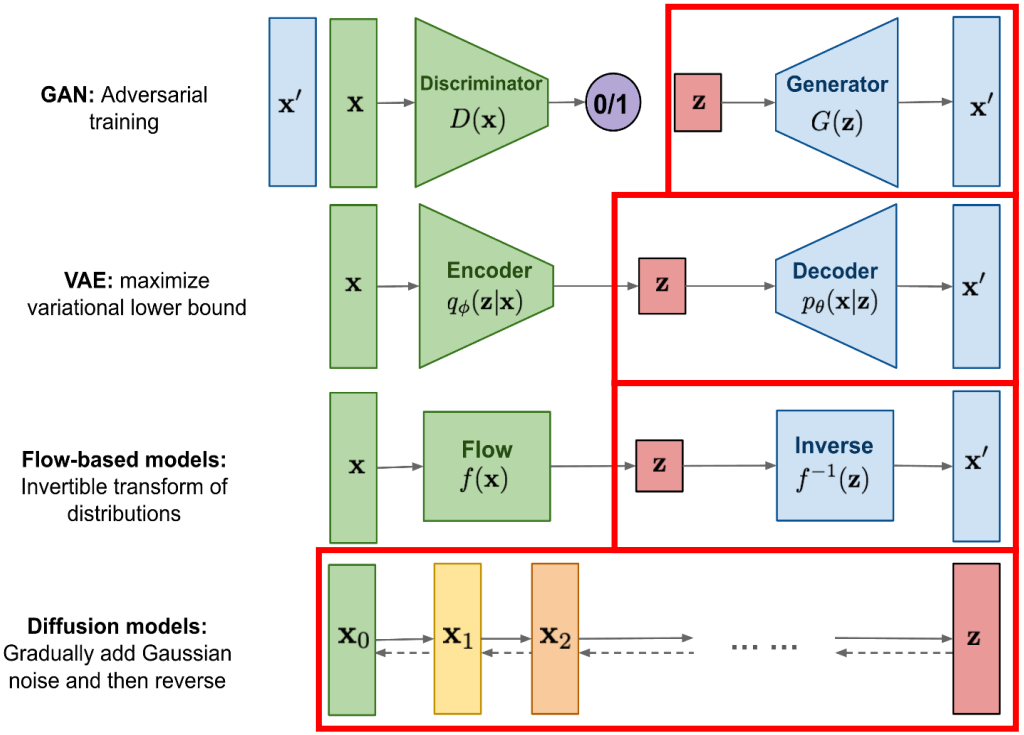
- Commonly aims to transform an input example x to a latent vector z that can be easily sampled from a known distribution.
- For example, $z \sim N(\mu, \sigma)$.
  - μ,σ can represent some features.
  - μ : Cat, Dog, …
  - σ : Black cat, White cat, …
- Generally, we call this **encoder**.



Generative Models

Generative models

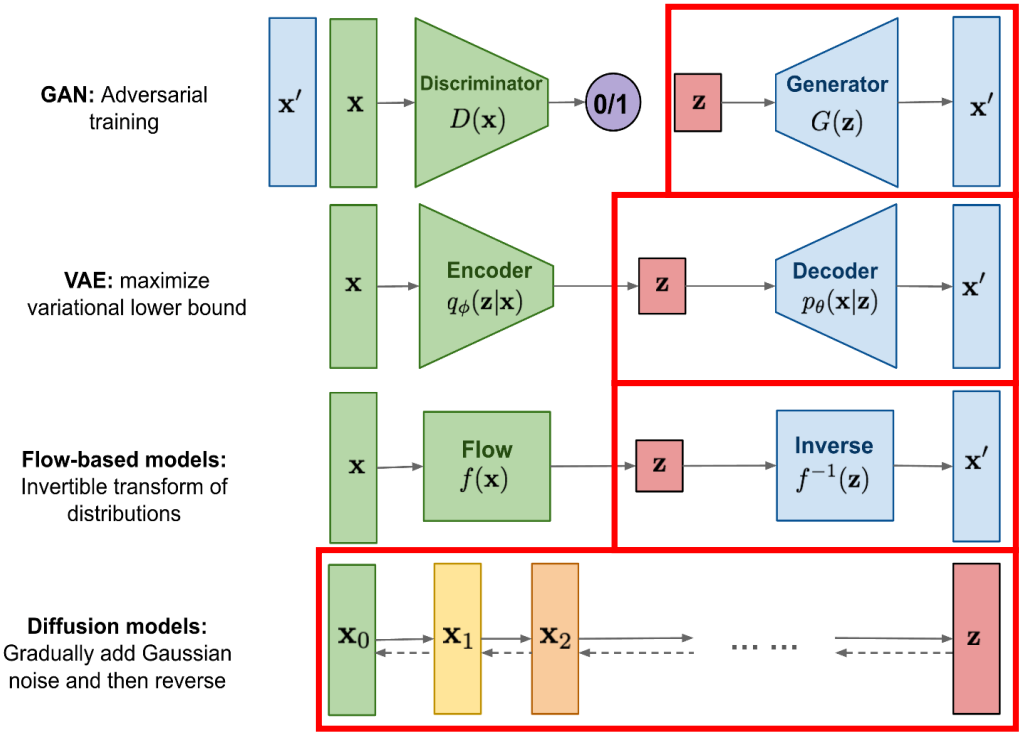
- For a sampled z, we transform z to an example x'.
  - Since we map input examples to with a known distribution, we can generate *infinite* examples.
- Generally, we call this ***decoder***.



Generative Models

Generative models

- GAN and VAE: encoder ≠ inverse decoder.
- Flow-based models (ODE): encoder = inverse decoder.
- ***Diffusion models.*** ***Is it natural?***



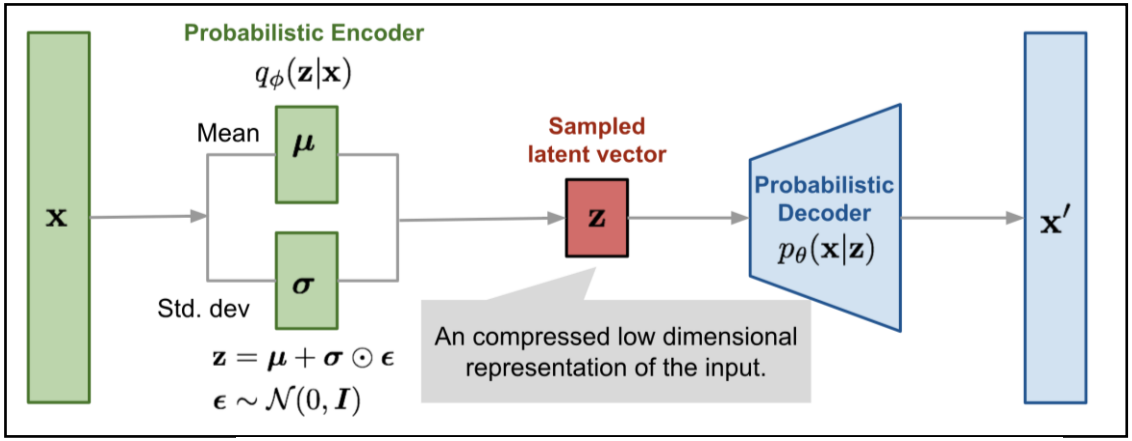
Contents

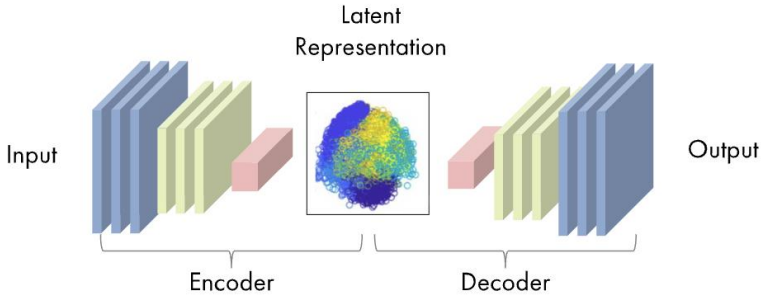
Variational AutoEncoder

Variational Autoencoder (VAE)

- **Variational Autoencoder (VAE)** models the data distribution using a conditional Gaussian decoder:
  - Latent prior: $z \sim \mathcal{N}(0, I)$
  - Conditional decoder: $p(x|z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))$, where $\mu_\theta, \Sigma_\theta$ are neural networks.
- Even though $p(x|z)$ is simple, the **marginal $p(x)$** is complex and flexible.

$$p_{model}(x) = \int p_\theta(x, z)\, dz = \int p_\theta(x|z)\, p(z)\, dz\,.$$





https://www.mathworks.com/discovery/autoencoder.html

Variational Autoencoder (VAE)

- **Q. How do we estimate $p_{model}(x)$?**
  - **A.** Maximize **ELBO**! (Or minimize **negative ELBO**)

$$p_{model}(x) = \int p_\theta(x,z)\, dz = \int p_\theta(x|z)\, p(z)\, dz\,.$$

- **ELBO (Evidence Lower BOund)** provides a lower bound on the log-likelihood $\log p(x;\theta)$.

$$\log p(x;\theta) \geq \int \log\big(p_\theta(x,z)\big)\, q_\phi(z|x) dz + H(q_\phi(z|x)) = \mathcal{L}(x;\theta,\phi)$$

Log-likelihood

$$\log p(x;\theta) = \mathcal{L}(x;\theta,\phi) + D_{KL}\big(q_\phi(z|x) \| p\,(z|x;\theta)\big)$$

ELBO — Gap to true posterior

  - The better $q_\phi(z|x)$ can approximate the posterior $p(z|x;\theta)$, the smaller the KL divergence.
  - Smaller KL → EBLO gets closer to true log-likelihood

- **Training goal**: Jointly optimize both θ and ϕ to maximize the ELBO over the dataset.

Standard VAE Objective

- VAE objective consists of two terms: a **reconstruction loss** and a **prior regularizer.**

$$\mathcal{L}(x;\theta,\phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)\|p(z))$$

Reconstruction Loss Regularizer

- Standard VAE Example
  - **Prior** $p_z(z) = \mathcal{N}(0, I)$.
  - **Gaussian Encoder** $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ with diagonal Σ_ϕ**.**
  - **Gaussian Decoder** $p_\theta(x|x) = \mathcal{N}(f_\theta(z), \sigma^2 I)$ where $\sigma > 0$ is a fixed **hyperparameter**.

Limitation of VAEs

- Standard VAE Example
  - **Prior** $p_z(z) = \mathcal{N}(0, I)$.
  - **Gaussian Encoder** $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ with diagonal Σ_ϕ**.**
  - **Gaussian Decoder** $p_\theta(x|x) = \mathcal{N}(f_\theta(z), \sigma^2 I)$ where $\sigma > 0$ is a fixed **hyperparameter**.

- **Q.** Is a **Gaussian** distribution sufficient as the variational approximation for the posterior distribution?

$$\log p(x;\theta) = \underbrace{\mathcal{L}(x;\theta,\phi)}_{\text{ELBO}} + \underbrace{D_{KL}\big(q_\phi(z|x)\|p\,(z|x;\theta)\big)}_{\text{Gap to true posterior}}$$

  - The ELBO becomes **tight** when the variational distribution $q_\phi(z|x)$ is **identical** to the true posterior distribution $p(z|x;\theta)$.
  - Will the true posterior distribution $p(z|x;\theta)$ be close to a Gaussian distribution?

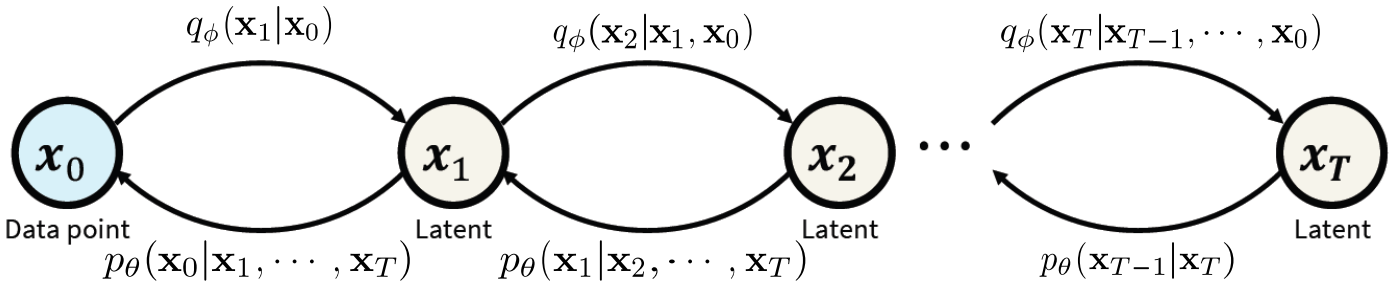
Hierarchical VAEs

- **Motivation: Go Beyond a Single Latent Variable**
  - Standard VAEs use a single latent variable $\boldsymbol{z}$ to capture all variations in $\mathbf{x}$.
  - Can we improve expressiveness by using a hierarchy of latent variables?

- **Make a hierarchical VAE.**
  - Data point: $\mathbf{x} \rightarrow \mathbf{x}_0$
  - Latent variables: $\mathbf{z} \rightarrow \mathbf{x}_{1:T}$

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|x_{t:T})$$

[For simplicity, the graph is for Markovian case.]



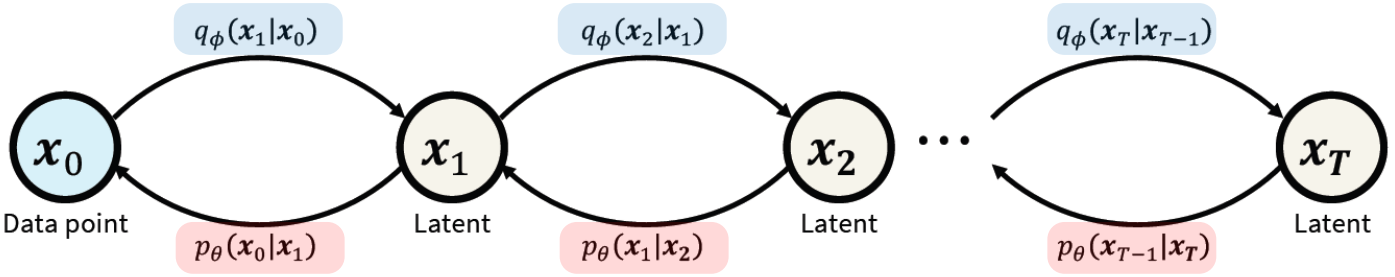
Hierarchical VAEs

- **Markov Assumption on Latent Variables**
  - Assume the latent variables follow a **Markov chain**.
  - The generative model becomes:

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|x_t)$$

  - The variational posterior becomes:

$$q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q_\phi(\mathbf{x}_t|x_{t-1})$$



Hierarchical VAEs

- **Markov Assumption on Latent Variables**
  - Assume the latent variables follow a **Markov chain**.
  - The generative model becomes:

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|x_t)$$

  - The variational posterior becomes:

$$q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q_\phi(\mathbf{x}_t|x_{t-1})$$

- **ELBO**

$$\begin{aligned}
\log p(\mathbf{x}_0) &= \log \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T} \\
&= \log \int p_\theta(\mathbf{x}_{0:T}) \frac{q_\phi(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{q_\phi(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}\, d\mathbf{x}_{1:T} \\
&= \log \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p_\theta(\mathbf{x}_{0:T})}{q_\phi(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right] \\
&\geq \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q_\phi(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right] \quad (\because \textbf{Jensen's Inequality})
\end{aligned}$$

Contents

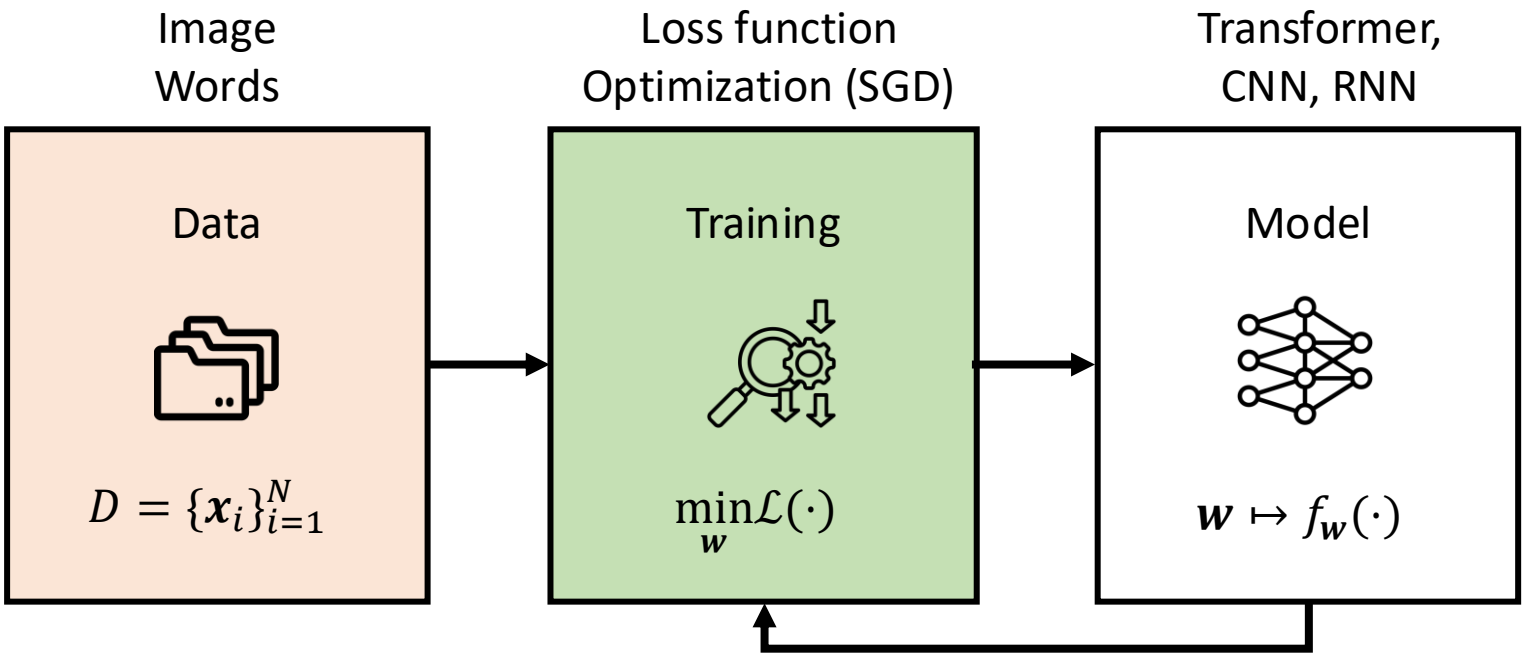
Learning Score Function

Machine Learning

Minimize the loss function with data, model, and optimization

Empirical Risk Minimization

$$\min_{\boldsymbol{w}\in\boldsymbol{\Theta}} \mathbb{E}_{\boldsymbol{x}\sim D}\mathcal{L}(\boldsymbol{x};\boldsymbol{w})$$

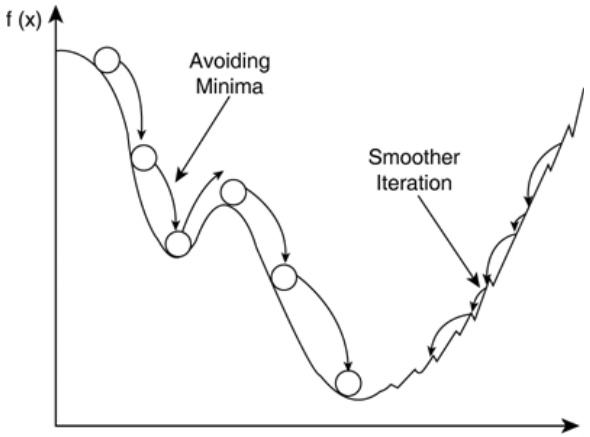


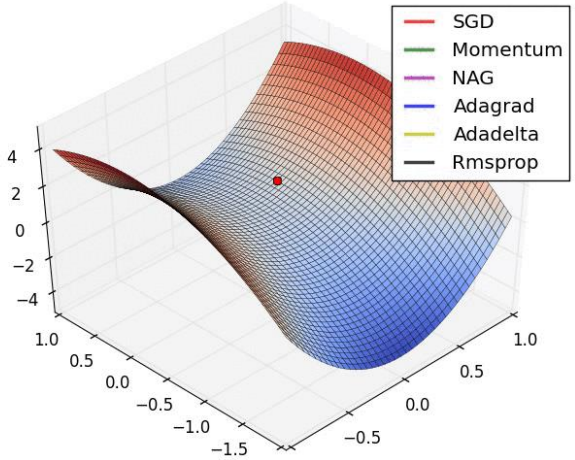
Optimization: (Stochastic) Gradient Descent methods

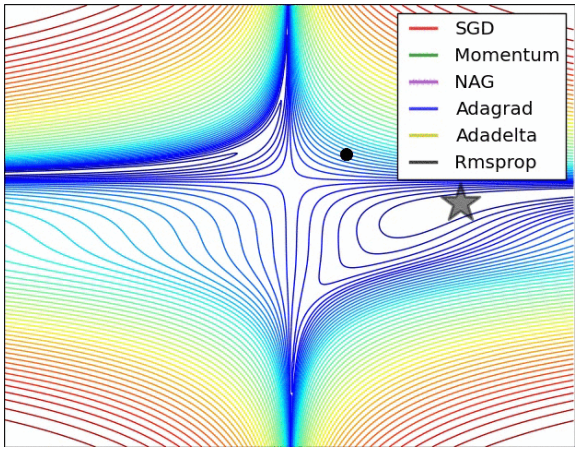
- For loss function $\mathcal{L}$, we update the weights $\theta_{t+1} = \theta_t - \nabla\mathcal{L}(x_t; \theta_t)$

Empirical Risk Minimization

$$\min_{\boldsymbol{w}\in\boldsymbol{\Theta}} \mathbb{E}_{\boldsymbol{x}\sim D}\mathcal{L}(x; \boldsymbol{w}) \qquad \text{(ERM)}$$





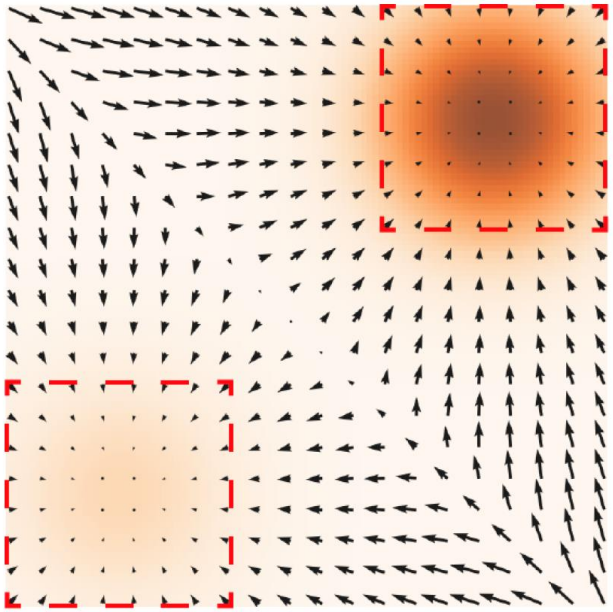


http://www.yaldex.com
https://shuuki4.github.io/deep%20learning/2016/05/20/Gradient-Descent-Algorithm-Overview.html

Gradient of Probability Density

- If the probability density function (pdf) is differentiable, we can compute the gradient of a probability density (***Score function***).

Score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$



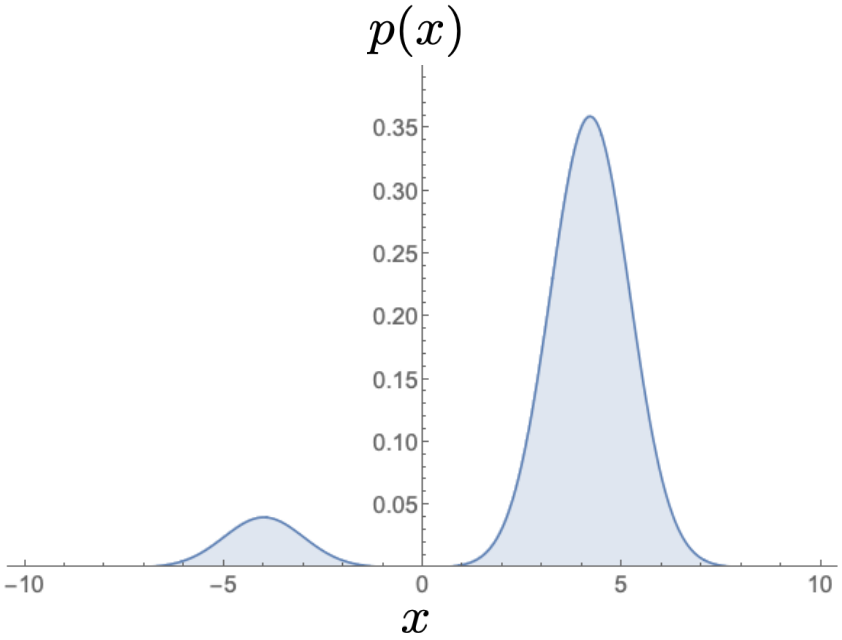
pdf $p_\theta(x)$ vs. Score $s_\theta(x)$

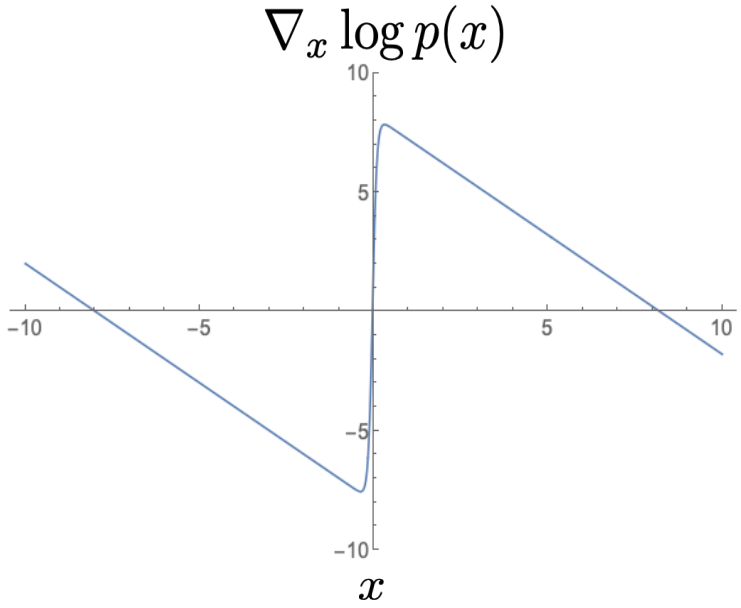
Song, Yang, and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution." *NeurIPS* 2019.

Gradient of Probability Density

- If the probability density function (pdf) is differentiable, we can compute the gradient of a probability density (***Score function***).

Score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$





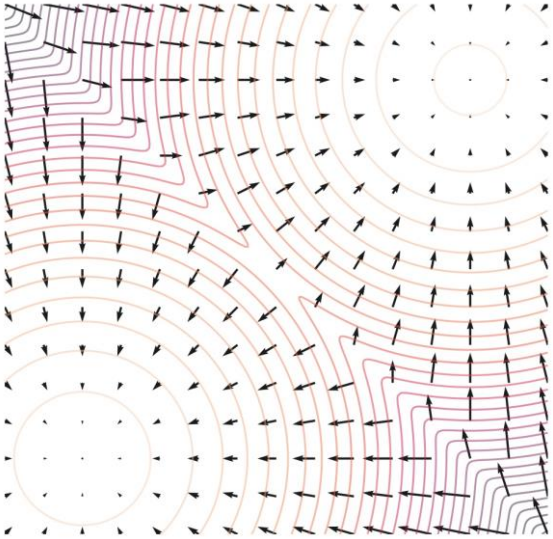
Score-Based Models

- **Q.** What is the most general model family of models that can be efficiently trained via score matching?
- **A. Score-based Models**
  - Directly learn the score function $s_\theta(x)$ (vector field of gradients)

$$s_\theta(\mathbf{x}) := \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$$

Neural Network

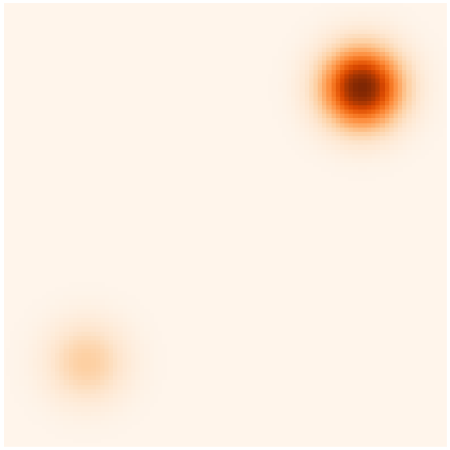
- **Observation:**
  - Score-based models generalize traditional likelihood-based generative models.
    - Autoregressive models
    - Normalizing flows
    - Energy-Based Models (EBM)



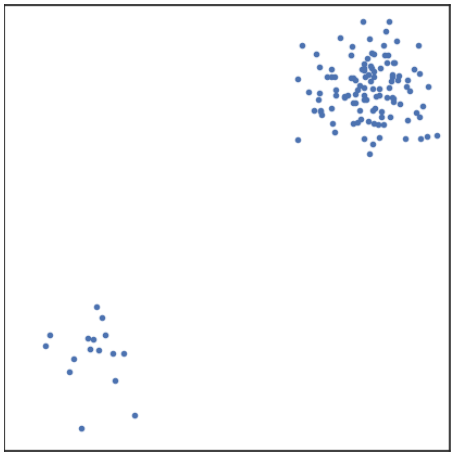
Score-Based Models

- **Formal Framework:**
  - Estimate the **score function** from i.i.d. samples **without access to the explicit density**.

Probability density
$p_{\text{data}}(\mathbf{x})$

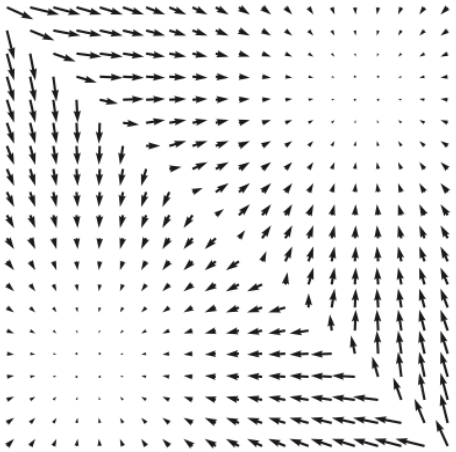


i.i.d. samples from $p_{data}(\mathbf{x})$
$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$



?

Score function approximation
$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$



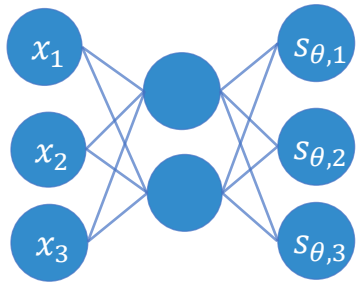
Score-Based Models

- **Objective:** Minimize the ***expected squared distance*** between the model and the true score

$$\frac{1}{2}\,\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\left\|\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x}) - s_\theta(\mathbf{x})\right\|_2^2\right]$$

- Use neural networks to model the score function $s_\theta(\mathbf{x})$

$$s_\theta(\mathbf{x}) = [s_{\theta,1}(\mathbf{x}), s_{\theta,2}(\mathbf{x}), \cdots, s_{\theta,d}(\mathbf{x})]$$



- **Requirements:**
  - The score model must be computationally efficient.
  - Must the model represent a *proper score function?*

Contents

Denoising Diffusion Models

DDPM (NeurIPS 2020)

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

What are diffusion models?

Inspired by **diffusion** process (Brownian motion)

- **Brownian motion (Physics)**
- **Ito Process (Economics)**

Diffusion models learn the reverse process to recover data from noise

Forward process adds noise to data

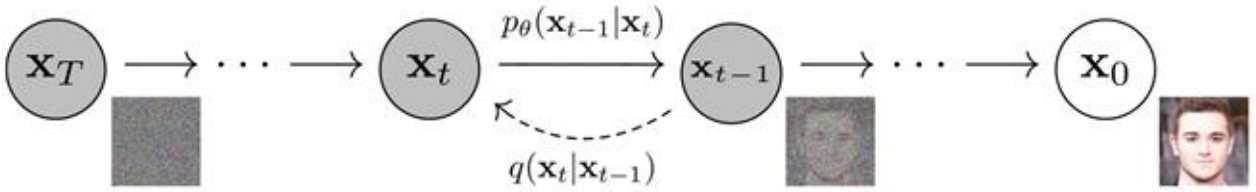


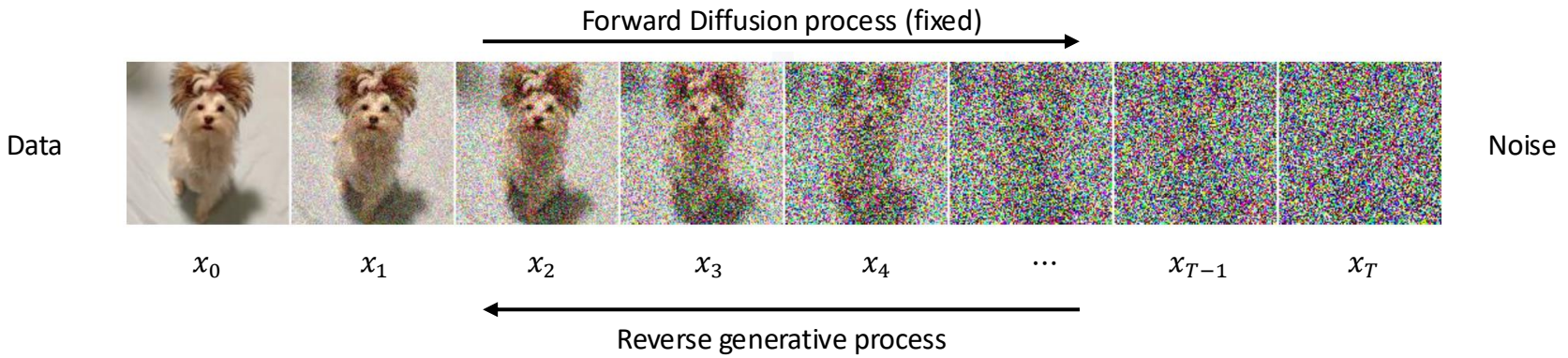
Image credit: Denoising Diffusion Probabilistic Models https://arxiv.org/pdf/2006.11239

Diffusion models learn the reverse process to recover data from noise

Image credit: https://quantgirluk.github.io/Understanding-Quantitative-Finance/brownian_motion.html

Denoising Diffusion Probabilistic Models

- DDPM consist of two processes:
  - **Forward process** adds noise to data.
  - **Reverse process** generates data from noise by denoising.



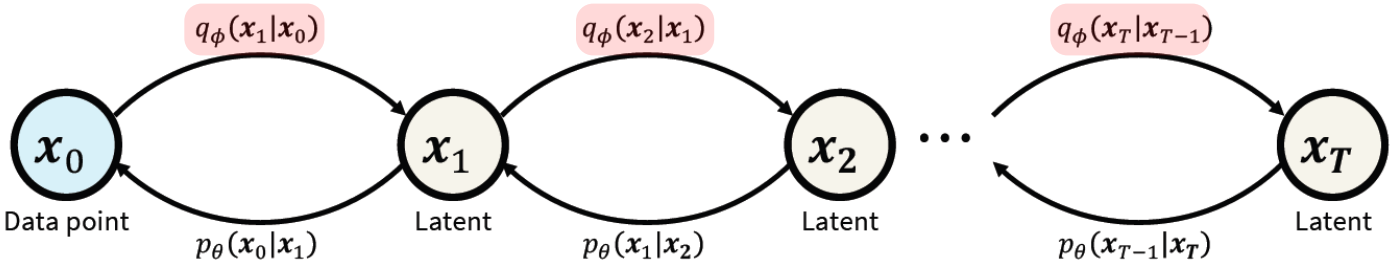
Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *NeurIPS,* 2020.
Song, Yang, et al. "Score-based generative modeling through stochastic differential equations." *ICLR*, 2021

Denoising Diffusion Probabilistic Models

- Consider a special case of the **Markovian hierarchical VAEs**, where:
  - The **latent variables** $\mathbf{x}_{1:T}$ **have the same dimension** as the data $\mathbf{x}_0$
  - The **variational posteriors** $q_\phi(\mathbf{x}_{t+1}|\mathbf{x}_t)$ are **not learned**, but instead **predefined**.

$$q_\phi(\mathbf{x}_{t+1}|\mathbf{x}_t) \rightarrow q(\mathbf{x}_{t+1}|\mathbf{x}_t)$$

  - This setup forms the foundation of diffusion-based generative models.



https://mhsung.github.io/kaist-cs492d-fall-2024/

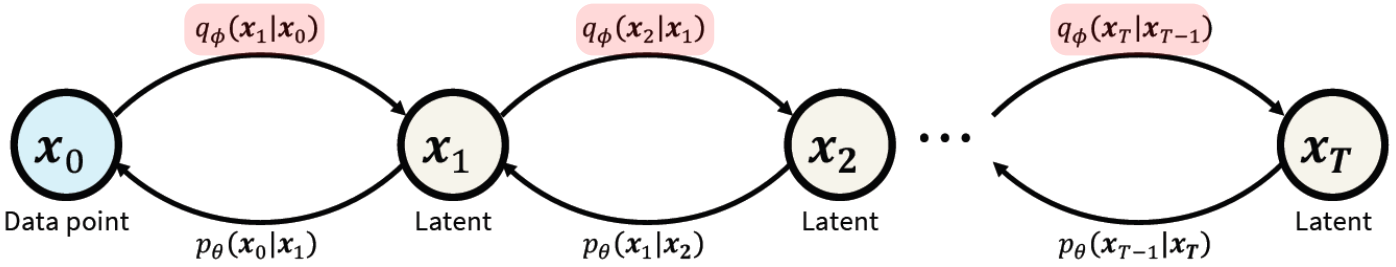
Terminology

- **Forward** Process (**predefined noise process**):

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{i=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

- **Reverse** Process (**learned denoising process**):

$$p_\theta(\mathbf{x}_0, \mathbf{x}_{1:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$



https://mhsung.github.io/kaist-cs492d-fall-2024/

Forward Process (Data → Latent)

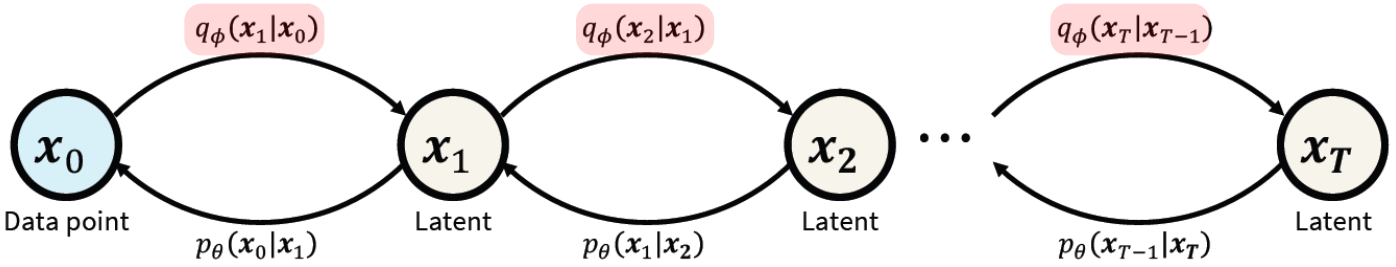
- In the **Forward process**, the transition distribution is specifically **predefined** as:

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

- Where $\{\beta_t \in (0,1)\}_{t=1}^{T}$ is a sequence of **time-dependent noise scales**, with:

$$\beta_1 \leq \beta_2 \leq \cdots \leq \beta_T$$

- Adds gradually increasing Gaussian noise to the data at each timestep.



https://mhsung.github.io/kaist-cs492d-fall-2024/

Forward Process (Data → Latent)

- Note that the reverse step $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ becomes a Gaussian form only when β_t is small ($\beta_t \ll 1$).

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

- **Common Strategies for β_t:**
  - Learned.
  - Constant
  - Linearly or quadratically increased.
  - Follows a cosine function
    - Nichol and Dhariwal, Improved Denoising Diffusion Probabilistic Models, ICML 2021.
  - EDM
    - Karras, Tero, et al., Elucidating the design space of diffusion-based generative models. NeurIPS 2022.

https://mhsung.github.io/kaist-cs492d-fall-2024/

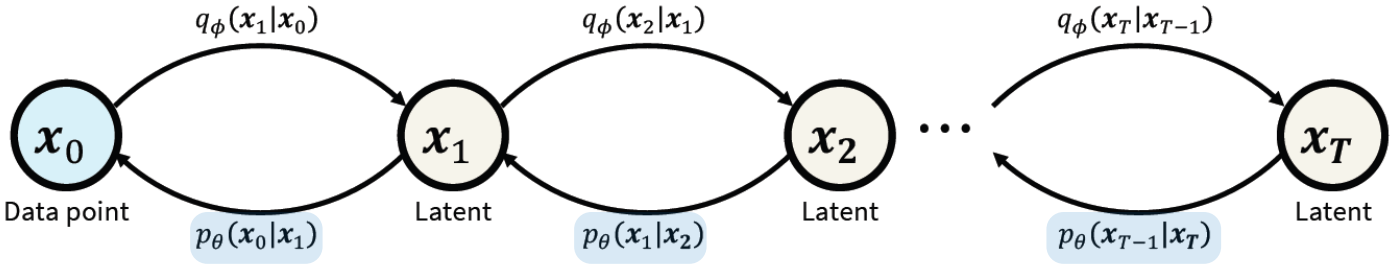
Reverse Process (Latent → Data)

- In the **Reverse (Generative) process**, the transition distribution is **modeled as**:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

Neural Network (U-Net)

- **Why Gaussian?**
  - If $\boldsymbol{\beta_t} \ll \mathbf{1}$, the reverse step $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is well-approximated with the variational distribution of the **same form** as the forward process $q(\mathbf{x}_t|\mathbf{x}_{t-1})$



https://mhsung.github.io/kaist-cs492d-fall-2024/

Diffusion ELBO

- How to minimize the *negative ELBO* in this case?

$$-\log p(\mathbf{x}_0) = -\log \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T} \leq \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q_\phi(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right]$$

$$\begin{aligned}
L &= \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \\
&= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \quad \left(\because p_\theta(\mathbf{x}_0, \mathbf{x}_{1:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t),\ q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{i=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \right) \\
&= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t > 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} \right]
\end{aligned}$$

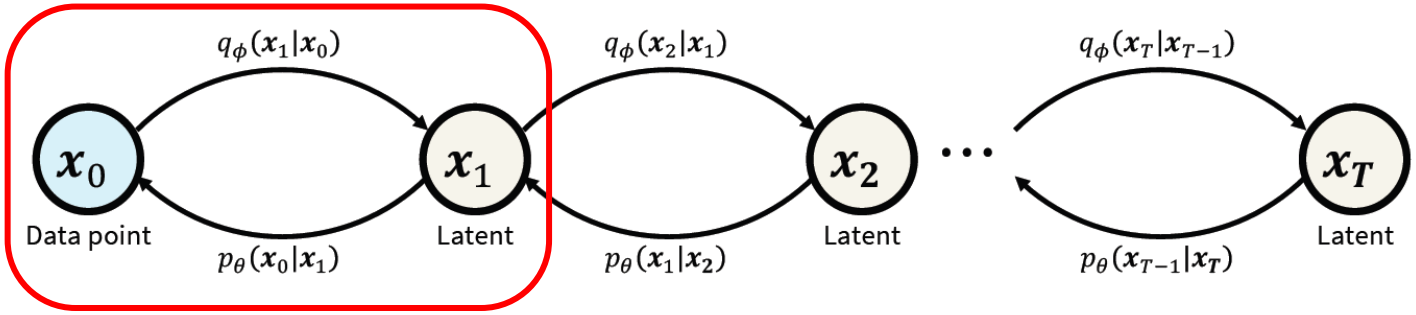
Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *NeurIPS* 2020.
Luo, Calvin. "Understanding diffusion models: A unified perspective."*arXiv preprint arXiv:2208.11970* (2022).

Diffusion ELBO

Reconstruction Term $\mathcal{L}_0$

- This term is **identical to the VAE reconstruction loss**, but applied **only to the final reverse step $\mathbf{x}_1 \rightarrow \mathbf{x}_0$.**
  - Usually, MSE loss in DDPM.

$$-\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]$$



Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *NeurIPS* 2020.
Luo, Calvin. "Understanding diffusion models: A unified perspective."*arXiv preprint arXiv:2208.11970* (2022).

Contents

Diffusion Training

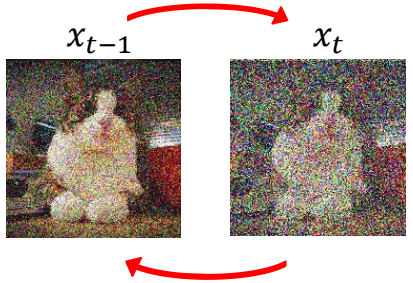
Diffusion Models

- **Training and Sampling**
  - Training

Same as sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$

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



$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon, t)||^2 \right]$$

Forward Convergence

Forward to arbitrary t step

- Let $\alpha_t = 1 - \beta_t$.

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

$$\{\alpha_t \in (0,1)\}_{t=1}^{T} \qquad \alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_T.$$

- **reparameterization trick**

Substituting recursively,

$$\mathbf{x}_1 = \sqrt{\alpha_1}\mathbf{x}_0 + \sqrt{1-\alpha_1}\epsilon_0$$
$$\mathbf{x}_2 = \sqrt{\alpha_2}\mathbf{x}_1 + \sqrt{1-\alpha_2}\epsilon_1$$

where $\epsilon_0, \epsilon_1 \sim \mathcal{N}(0, I)$

$$\begin{aligned} x_2 &= \sqrt{\alpha_2}\mathbf{x}_1 + \sqrt{1-\alpha_2}\epsilon_1 \\ &= \sqrt{\alpha_2}\left(\sqrt{\alpha_1}\mathbf{x}_0 + \sqrt{1-\alpha_1}\epsilon_0\right) + \sqrt{1-\alpha_2}\epsilon_1 \\ &= \sqrt{\alpha_2\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_2(1-\alpha_1)}\epsilon_0 + \sqrt{1-\alpha_2}\epsilon_1 \\ &= \sqrt{\alpha_2\alpha_1}\mathbf{x}_0 + \sqrt{1-\alpha_2\alpha_1}\bar{\epsilon}_0 \end{aligned}$$

where $\bar{\epsilon}_0 \sim \mathcal{N}(0, I)$

Then,

$$\therefore q(\mathbf{x}_2|\mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_2\alpha_1}\mathbf{x}_0, (1-\alpha_2\alpha_1)I)$$

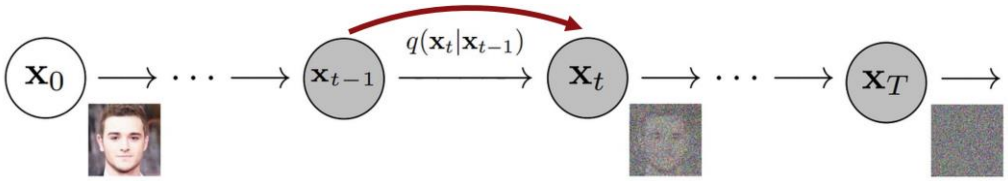
Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *NeurIPS* 2020.
Luo, Calvin. "Understanding diffusion models: A unified perspective."*arXiv preprint arXiv:2208.11970* (2022).

Forward Convergence

Forward to arbitrary t step

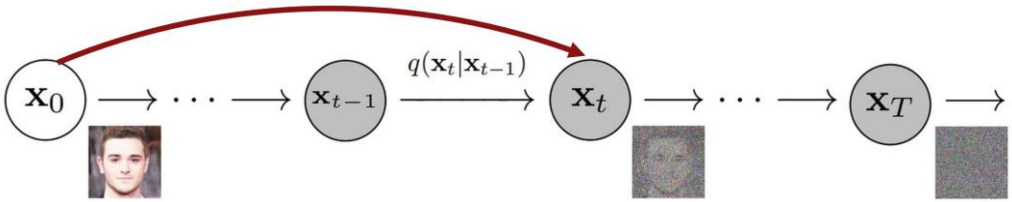
- **One-step Forward Process**

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$



- **Multi-step Forward Process (Closed Form)**

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \quad \text{where} \quad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$



Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *NeurIPS* 2020.
Luo, Calvin. "Understanding diffusion models: A unified perspective."*arXiv preprint arXiv:2208.11970* (2022).

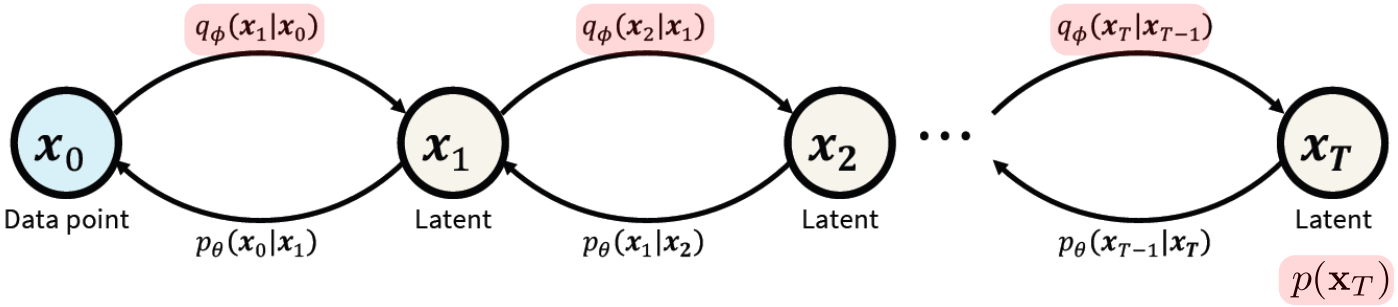
Forward Convergence

Prior Matching Term $\mathcal{L}_T$

- This term is **equivalent to the KL divergence term in VAE.**

$$\mathcal{L}_T = D_{KL}\left(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T)\right)$$

- But note that there is nothing to be optimized;
- $q(\mathbf{x}_T|\mathbf{x}_0)$ and $p(\mathbf{x}_T)$ are predefined.



Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *NeurIPS* 2020.
Luo, Calvin. "Understanding diffusion models: A unified perspective."*arXiv preprint arXiv:2208.11970* (2022).

Parametrization

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, t)||^2 \right]$$

- **Three options to define variational distribution $\boldsymbol{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}$:**

  1. **Mean predictor $\boldsymbol{\mu_\theta(\mathbf{x}_t, t)}$:**

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\mu_\theta(\mathbf{x}_t, t),\, \tilde{\sigma}_t^2 \mathbf{I}\right)$$

  2. **$\mathbf{x_0}$ predictor $\boldsymbol{\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t)}$:**

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t),\, \tilde{\sigma}_t^2 \mathbf{I}\right)$$

  3. **$\boldsymbol{\epsilon_t}$ predictor $\boldsymbol{\hat{\epsilon}_\theta(\mathbf{x}_t, t)}$**

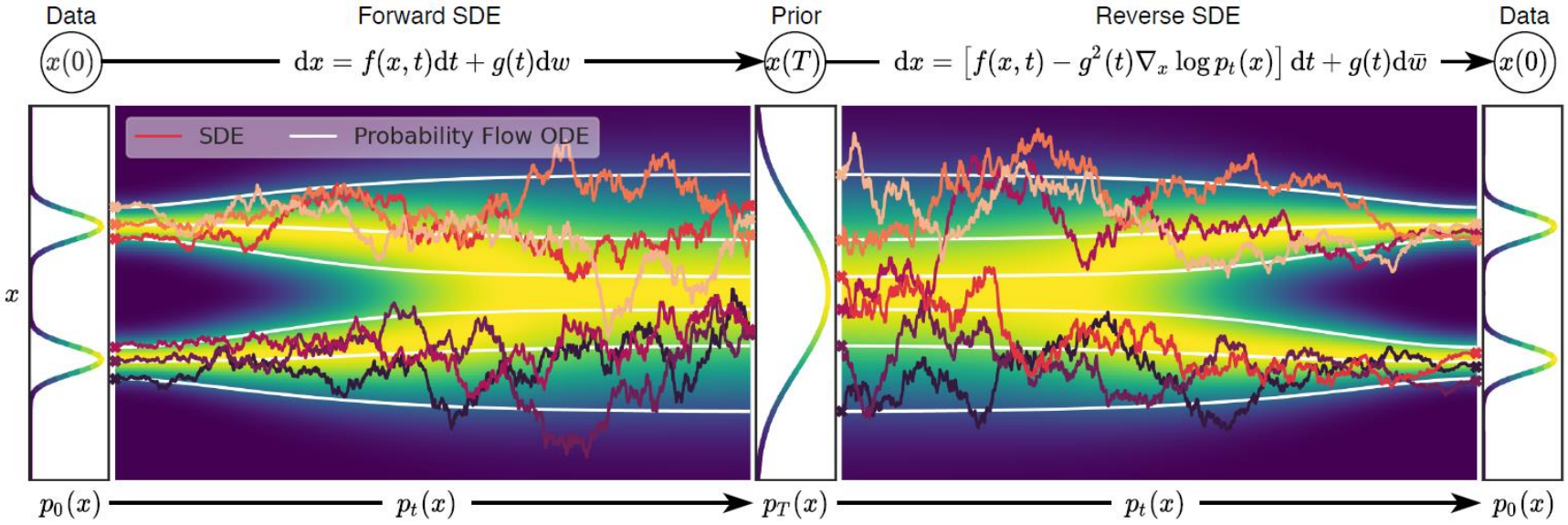
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t)\right),\, \tilde{\sigma}_t^2 \mathbf{I}\right)$$

Diffusion Model in Continuous Time

- In a **continuous** time setting, the data perturbation **(forward) process** is described by the following stochastic differential equation (SDE):

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

- $\mathrm{f}(\mathbf{x}, t)$: Drift coefficient
- $g(t)$: Diffusion coefficient
- dw: Infinitesimal white noise (called Brownian motion)



Song, Yang, et al. "Score-based generative modeling through stochastic differential equations." *ICLR* 2021.

Score matching and diffusion

- **Score based Generative model**
  - Denoising Score Matching

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t|\mathbf{x}_0)||_2^2$$

- Re-parametrized sampling: $\mathbf{x}_t=\gamma_t\mathbf{x}_0+\sigma_t\boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$
- Score function: $\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t|\mathbf{x}_0)=-\nabla_{\mathbf{x}_t}\frac{(\mathbf{x}_t-\gamma_t\mathbf{x}_0)^2}{2\sigma_t^2}=-\frac{\mathbf{x}_t-\gamma_t\mathbf{x}_0}{\sigma_t^2}=-\frac{\gamma_t\mathbf{x}_0+\sigma_t\boldsymbol{\epsilon}-\gamma_t\mathbf{x}_0}{\sigma_t^2}=-\frac{\boldsymbol{\epsilon}}{\sigma_t}$
- Neural network model: $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t):=-\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\sigma_t}$

➡ $\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\frac{1}{\sigma_t^2}||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2$

➡ **Same objectives as derived with variational approach**

Commonly used model architectures

Convolutional U-Nets

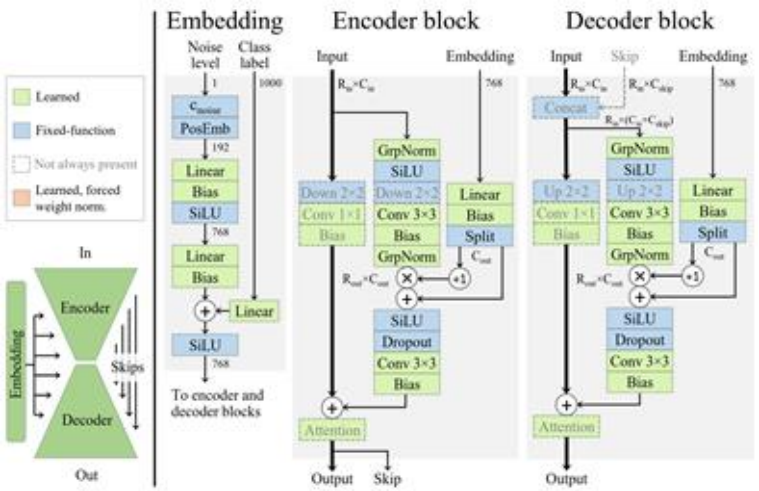


Image credit: Analyzing and Improving the Training Dynamics of Diffusion Models
https://arxiv.org/pdf/2312.02696

Patch-wise Transformers

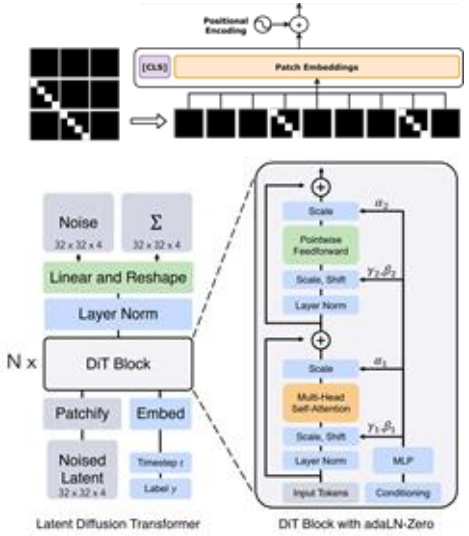


Image credit: Scalable Diffusion Models with Transformers
https://arxiv.org/pdf/2212.09748

Contents

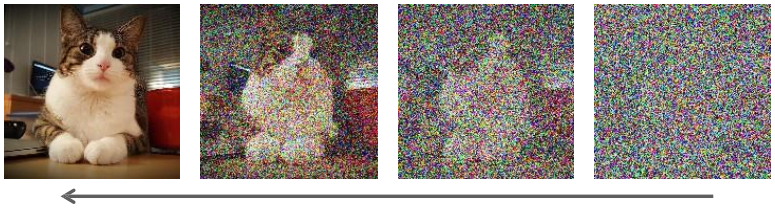
Diffusion Sampling

Diffusion Sampling

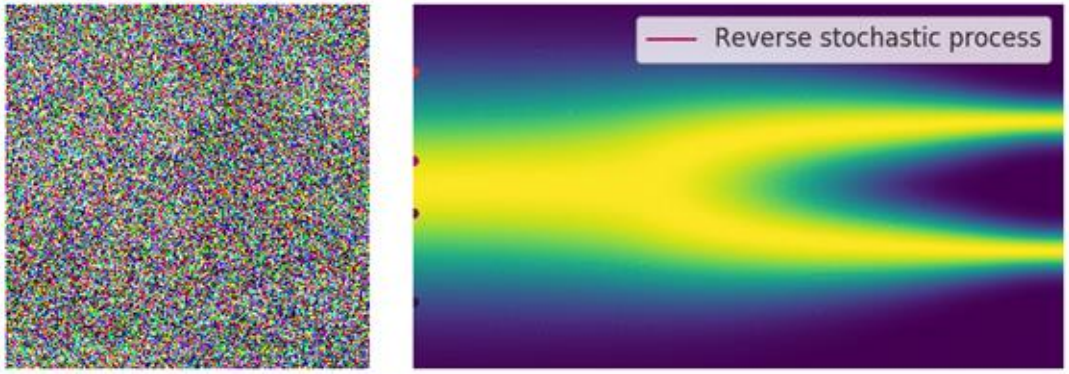
- **Training and Sampling**
  - Sampling

Algorithm 2 Sampling

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



$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}), \quad \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon \right)$$



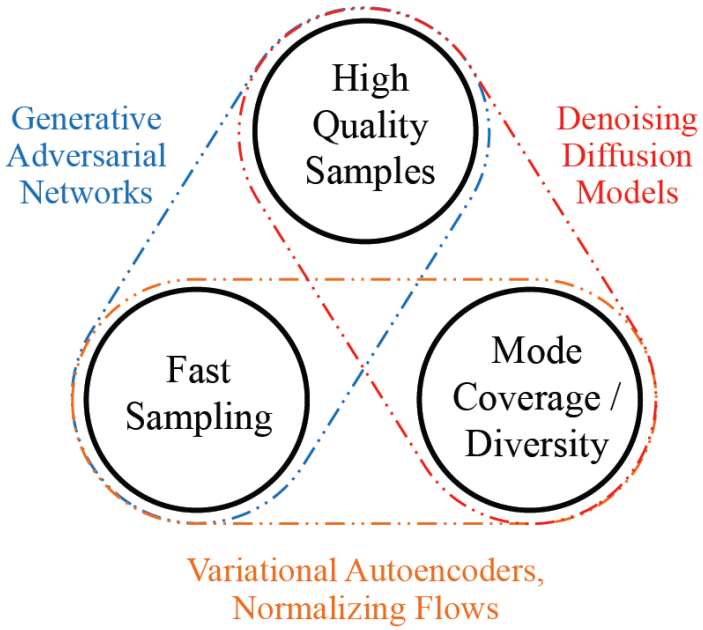
Reverse process. Image courtesy of Yang Song. https://yang-song.net/

Diffusion Sampling

Diffusion Models – Pros and Cons

- **(+)** High quality samples
- **(+)** Strong Diversity
- **(-) Very slow speed**

Q. How can we **speed up** the reverse process while **maintaining generation quality**?



Xiao, Zhisheng, Karsten Kreis, and Arash Vahdat. "Tackling the generative learning trilemma with denoising diffusion gans." *ICLR, 2022*.

Diffusion Sampling

Diffusion Models – Pros and Cons

- Sample generation from the DDPM model is approximately **1000x slower** than GANs due to the iterative sampling process.

- **Sampling Algorithm:**
  1. Sample $\mathrm{x}_T \sim \mathcal{N}(0, I)$.
  2. For $t = T, \cdots, 1$, repeat: $\boldsymbol{T = 1000}$
     1. Compute $\tilde{\boldsymbol{\mu}} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t) \right)$ · **Neural Network Evaluation**
     2. Sample $z_t \sim \mathcal{N}(0, I)$.
     3. Compute $\mathbf{x}_{t-1} = \tilde{\boldsymbol{\mu}} + \tilde{\sigma}_t \mathbf{z}_t$.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *NeurIPS* 2020.
Luo, Calvin. "Understanding diffusion models: A unified perspective." *arXiv preprint arXiv:2208.11970* (2022).

DDIM (ICLR 2021)

Published as a conference paper at ICLR 2021

DENOISING DIFFUSION IMPLICIT MODELS

Jiaming Song, Chenlin Meng & Stefano Ermon
Stanford University
{tsong,chenlin,ermon}@cs.stanford.edu

From DDPM to DDIM

- **DDPM's Markovian forward process:**

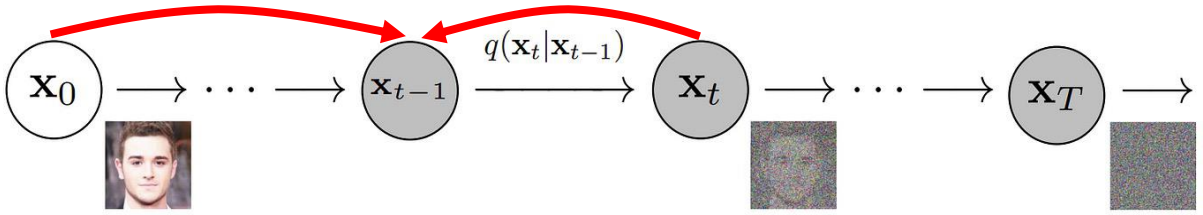
$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

  - $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is defined.
  - $q(\mathbf{x}_t|x_0)$ and $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ are derived from $q(\mathbf{x}_t|\mathbf{x}_{t-1})$.

- **DDIM's non-Markovian forward process:**

$$q_\sigma(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = q_\sigma(\mathbf{x}_T \mid \mathbf{x}_0) \prod_{t=2}^{T} q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$

  - $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is defined.
  - $q(\mathbf{x}_t|\mathbf{x}_0)$ and $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ are derived from $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$.



From DDPM to DDIM

One-step Estimator for $\mathbf{x_0}$ in DDPM

- **Q.** Given $\mathbf{x}_t$ and the noise prediction $\hat{\epsilon}_\theta(\mathbf{x}_t, t)$, what is the **best one-step prediction of $\mathbf{x_0}$** (denoted as $\hat{\mathbf{x}}_0(\mathbf{x}_t)$)?

$$\hat{\mathbf{x}}_0(\mathbf{x}_t) = \mathbb{E}_{q(\mathbf{x}_0|\mathbf{x}_t)}[\mathbf{x}_0] = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t)\right)$$

- **Why? (Connection to Tweedie's Formula)**
  - Given a prior $p(\mathbf{x_0})$ and a Gaussian perturbation $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$
  - then by the Tweedie's Formula:

$$\mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0|\mathbf{x}_t)}[\sqrt{\bar{\alpha}_t}\mathbf{x}_0] = \mathbf{x}_t + \sigma^2 \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) \approx \mathbf{x}_t + \sigma^2 s_\theta(\mathbf{x}_t)$$

$$= \sqrt{\bar{\alpha}_t}\hat{\mathbf{x}}_0(\mathbf{x}_t) \qquad = (1-\bar{\alpha}_t)\cdot\frac{-1}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}_\theta(\mathbf{x}_t, t)$$

DDIM

Review: DDPM Sampling Algorithm

- Goal: Generate samples by reversing the learned diffusion process using reverse transition:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t)\right), \tilde{\sigma}_t^2 I\right)$$

- **Sampling Algorithm:**

1. Sample $\mathrm{x}_T \sim \mathcal{N}(0, I)$.
2. For $t = T, \cdots, 1$, repeat:
   1. Compute $\tilde{\boldsymbol{\mu}} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t)\right)$.
   2. Sample $z_t \sim \mathcal{N}(0, I)$.
   3. Compute $\mathbf{x}_{t-1} = \tilde{\boldsymbol{\mu}} + \tilde{\sigma}_t \mathbf{z}_t$.

Same as sampling $\mathbf{x}_t \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *NeurIPS* 2020.
Luo, Calvin. "Understanding diffusion models: A unified perspective." *arXiv preprint arXiv:2208.11970* (2022).

DDIM

DDIM Reverse Process

$$q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}},\ \sigma_t^2\mathbf{I}\right)$$

- **Sampling Algorithm:**
  1. Sample $\mathrm{x}_T \sim \mathcal{N}(0, I)$.
  2. For $t = T, \cdots, 1$, repeat:
     1. Compute $\hat{\mathbf{x}}_0(\mathbf{x}_t) = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t)\right)$
     2. Compute $\tilde{\boldsymbol{\mu}} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\,\hat{\mathbf{x}}_0}{\sqrt{1-\bar{\alpha}_t}}$
     3. Sample $z_t \sim \mathcal{N}(0, I)$.
     4. Compute $\mathbf{x}_{t-1} = \tilde{\boldsymbol{\mu}} + \sigma_t \mathbf{z}_t$

DDIM

Loss Function

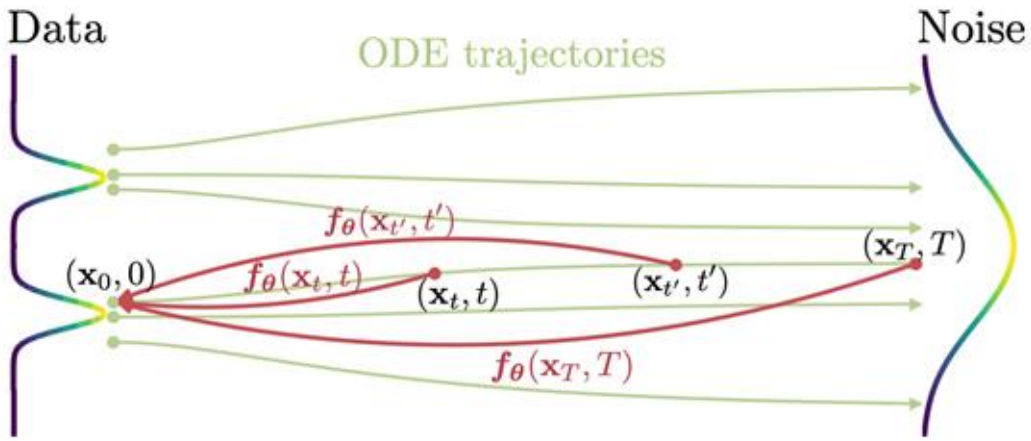
- The **ELBO remains unchanged** from DDPM, up to a constant.
- The variational likelihood distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ in the reverse process is learned by minimizing the **same loss function** as in DDPM:

$$\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[D_{\mathrm{KL}}\left(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right)\right]$$

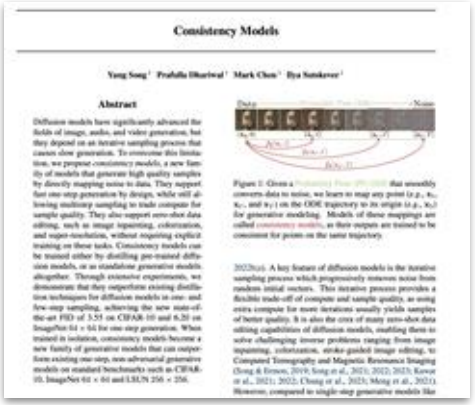
  - **No need to retrain the noise predictor!**
  - The same noise predictor $\hat{\epsilon}_\theta(\mathbf{x}_t, t)$, trained for DDPM, can be directly reused in the DDIM reverse process.
  - Using the same noise predictor $\hat{\epsilon}_\theta(\mathbf{x}_t, t)$, you can choose to perform either the **DDPM** or the **DDIM** sampling strategies.

Consistency models

Make distillated model to enable few step generator of from any t



$$\mathcal{L}_{\mathrm{CM}} = \|f_\theta(x_t, t) - \mathrm{stopgrad}(f_\theta(x_s, s))\|_2^2$$


Consistency Models

Abstract

Song, Yang, et al. "Consistency Models." *International Conference on Machine Learning*. PMLR, 2023.

Code

https://github.com/JinseongP/cac2026_diffusion

Contents

Conditional Generation

Conditional Generation

Classifier Guidance

- **Goal:** How can we make a diffusion model generate samples conditioned on a label?
  - How do we model the conditional distribution (ex) $p(\mathbf{x} \mid \text{"}\mathbf{Dog}\text{"})$ using diffusion?

Dog 

Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." *NeurIPS 2021 Workshop*.

Conditional Generation

Classifier Guidance

- **Given a condition variable as** y, from the Bayes' Rule:

$$p_\theta(\mathbf{x}_t \mid y) \propto p_\theta(\mathbf{x}_t) p(y \mid \mathbf{x}_t) \approx p_\theta(\mathbf{x}_t) p_\phi(y \mid \mathbf{x}_t)$$

Classifier from the noisy observation $\mathbf{x}_t$

- From the equivalence of **Denoising score matching** and **DDPM training:**

$$\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \hat{\epsilon}_\theta(\mathbf{x}_t, t)$$

- Therefore, the **conditional score** becomes:

$$\begin{aligned} \nabla_{\mathbf{x}_t} \log \left(p_\theta(\mathbf{x}_t) p_\phi(y \mid \mathbf{x}_t) \right) &= \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \hat{\epsilon}_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t) \end{aligned}$$

Unconditional score — **Classifier Guidance**

Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." *NeurIPS* 2021.

Conditional Generation

Classifier Guidance

- Equivalently, for the ϵ-predictor parameterization, we have

$$\tilde{\epsilon}(\mathbf{x}_t, y, t) := \hat{\epsilon}_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t}\,\nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t)$$

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\,\nabla_{x_t} \log p_\phi(y|x_t)$ — Conditional ϵ-predictor
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$ — Deterministic DDIM Sampling
end for
return x_0

Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." *NeurIPS* 2021.

Conditional Generation

Classifier-Free Guidance

- **Q.** How can we enhance the conditioning (i.e., **apply stronger guidance**)?
  - By extrapolating the conditional noise estimate with **guidance scale** $\omega \geq 0$:

$$\tilde{\epsilon}_w(\mathbf{x}_t, t) := \hat{\epsilon}_\theta(\mathbf{x}_t, t) - (1+\omega)\sqrt{1-\bar{\alpha}_t}\,\nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t)$$

$$\Rightarrow \quad -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\tilde{\epsilon}_w(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log \left(p_\theta(\mathbf{x}_t) \cdot p_\phi(y \mid \mathbf{x}_t)^{1+\omega}\right)$$

- We can remove the **dependency on the classifier $p_\phi(y|\mathbf{x}_t)$** as follows:

$$p(\mathbf{x}_t) \cdot p(y \mid \mathbf{x}_t)^{(\omega+1)} \propto p(\mathbf{x}_t) \cdot \left(\frac{p(\mathbf{x}_t \mid y)}{p(\mathbf{x}_t)}\right)^{(\omega+1)}$$

$$\propto p(\mathbf{x}_t \mid y)^{(\omega+1)} \cdot p(\mathbf{x}_t)^{(-\omega)}$$

Conditional dist. **Unconditional dist.**

Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." *NeurIPS 2021 Workshop*.

Conditional Generation

Classifier-Free Guidance

- **Goal:** Apply conditional generation without relying on a separate classifier:
- Modify the noise prediction network to **take the condition** (e.g. label) y as an additional input:

$$\hat{\epsilon}(\mathbf{x}_t, t) \quad \rightarrow \quad \hat{\epsilon}(\mathbf{x}_t, y, t)$$

- Train both **conditional** and **unconditional** models by using a **null condition** ϕ:

$\hat{\epsilon}(\mathbf{x}_t, \phi, t)$: Noise prediction ***without*** condition.

- **Q.** How can we enhance the conditioning (i.e., **apply stronger guidance**)?
  - We can extrapolate the conditional noise $\hat{\epsilon}(x_t, y, t)$ from the null-conditioned noise $\hat{\epsilon}(x_t, \phi, t)$.
  - For $\omega \geq 0$,

$$\tilde{\epsilon}_\theta(\mathbf{x}_t, y, t) = (1+\omega)\hat{\epsilon}_\theta(\mathbf{x}_t, y, t) - \omega\hat{\epsilon}_\theta(\mathbf{x}_t, \phi, t)$$

Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." *NeurIPS 2021 Workshop*.

Conditional Generation

Classifier-Free Guidance

Algorithm 1 Joint training a diffusion model with classifier-free guidance

Require: p_{uncond}: probability of unconditional training

repeat
  $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
  $\mathbf{c} \leftarrow \varnothing$ with probability p_{uncond} ▷ Randomly discard conditioning to train unconditionally
  $\lambda \sim p(\lambda)$ ▷ Sample log SNR value
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$ ▷ Corrupt data to the sampled log SNR value
  Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$ ▷ Optimization of denoising model
until converged

Algorithm 2 Conditional sampling with classifier-free guidance

Require: w: guidance strength
Require: $\mathbf{c}$: conditioning information for conditional sampling
Require: $\lambda_1, \ldots, \lambda_T$: increasing log SNR sequence with $\lambda_1 = \lambda_{\min}, \lambda_T = \lambda_{\max}$

$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = 1, \ldots, T$ **do**
  ▷ Form the classifier-free guided score at log SNR λ_t
  $\tilde{\boldsymbol{\epsilon}}_t = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{z}_t)$
  ▷ Sampling step (could be replaced by another sampler, e.g. DDIM)
  $\tilde{\mathbf{x}}_t = (\mathbf{z}_t - \sigma_{\lambda_t} \tilde{\boldsymbol{\epsilon}}_t)/\alpha_{\lambda_t}$
  $\mathbf{z}_{t+1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\lambda_{t+1}|\lambda_t}(\mathbf{z}_t, \tilde{\mathbf{x}}_t), (\tilde{\sigma}^2_{\lambda_{t+1}|\lambda_t})^{1-v} (\sigma^2_{\lambda_t|\lambda_{t+1}})^v)$ if $t < T$ else $\mathbf{z}_{t+1} = \tilde{\mathbf{x}}_t$
end for
return $\mathbf{z}_{T+1}$

Conditional Generation

Classifier-Free Guidance

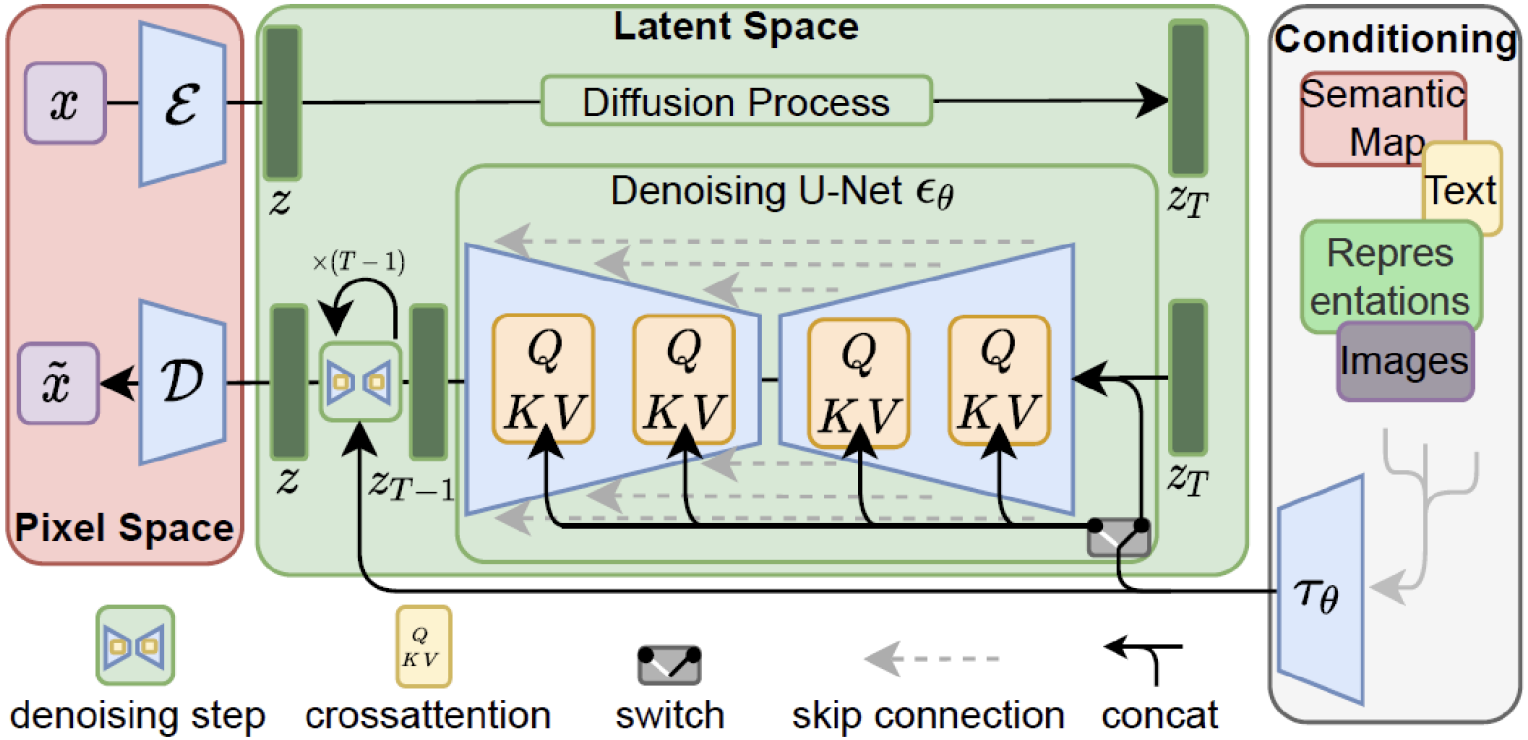
- (+) **Simple and practical**: Easy to implement.
- (+) **Flexible**: Can be used not only for labels but also for any additional information (e.g., text descriptions).
- (-) **Training overhead**: The noise predictor must be trained with both conditional and unconditional inputs.
- (-) **Evaluation cost**: The noise predictor needs to be evaluated twice per timestep in the generation process (once with y, once with null condition ϕ).

(a) $w = 0$ (b) $w = 1$ (c) $w = 2$ (d) $w = 4$

Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." *NeurIPS 2021 Workshop*.

Latent Diffusion Model (LDM)

- 1) Encoder – Decoder structure (Latent Diffusion, not pixel space)
- 2) Classifier-free Guidance (CFG) with Text embedding
- 3) U-Net + self attention + cross attention (condition)



Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.

Contents

Flow Matching

Flow Matching

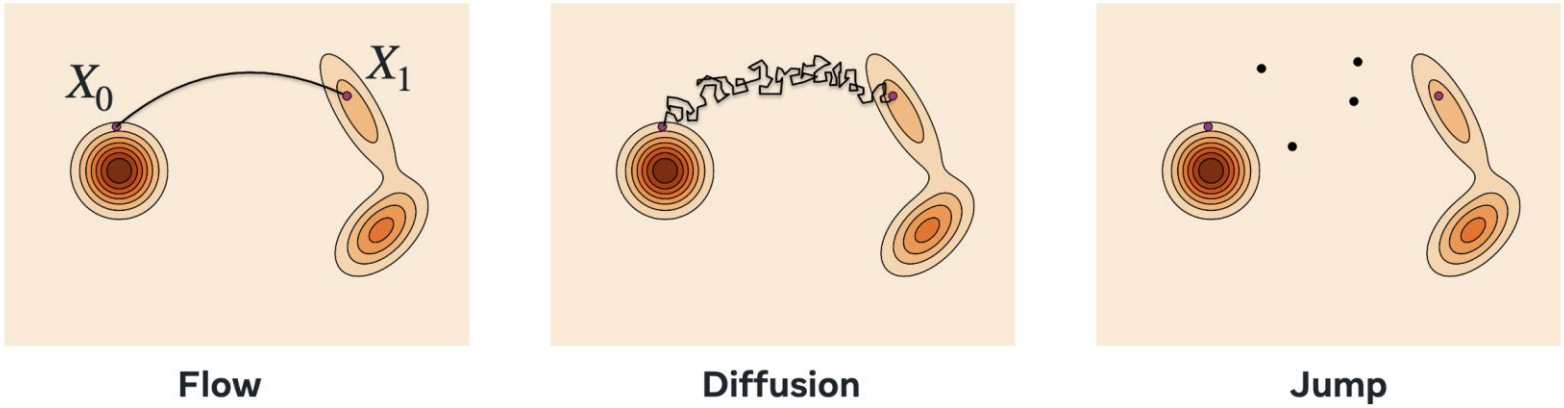
Diffusion Models: Forward corruption process, reverse-time SDE

- Usually, use **Gaussian** and **independent coupling** with (x_0, x_1)

Flow Matching: How to learn the velocity flow of (x_0, x_1)?

- Generalized version of diffusion models in **(continuous) Gaussian case**
- Continuous-time Markov process $(X_t)_{0 \le t \le 1}$

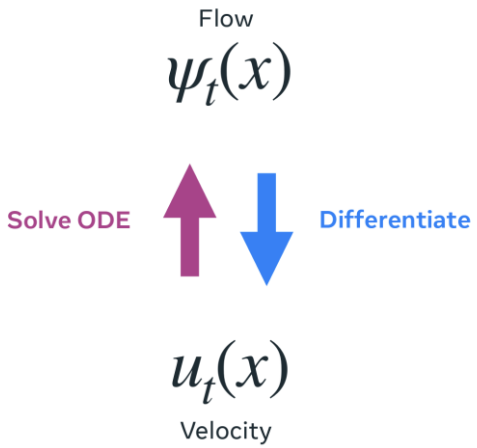
$$X_{t+h} \leftarrow \Phi_{t+h|t}(X_t)$$

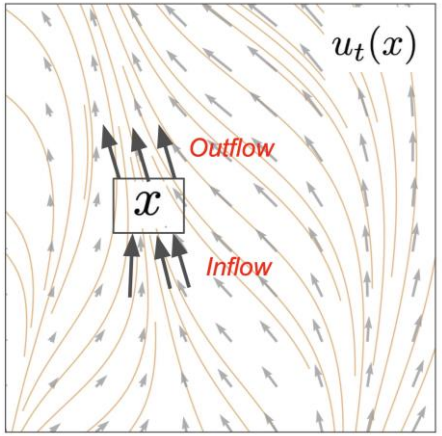


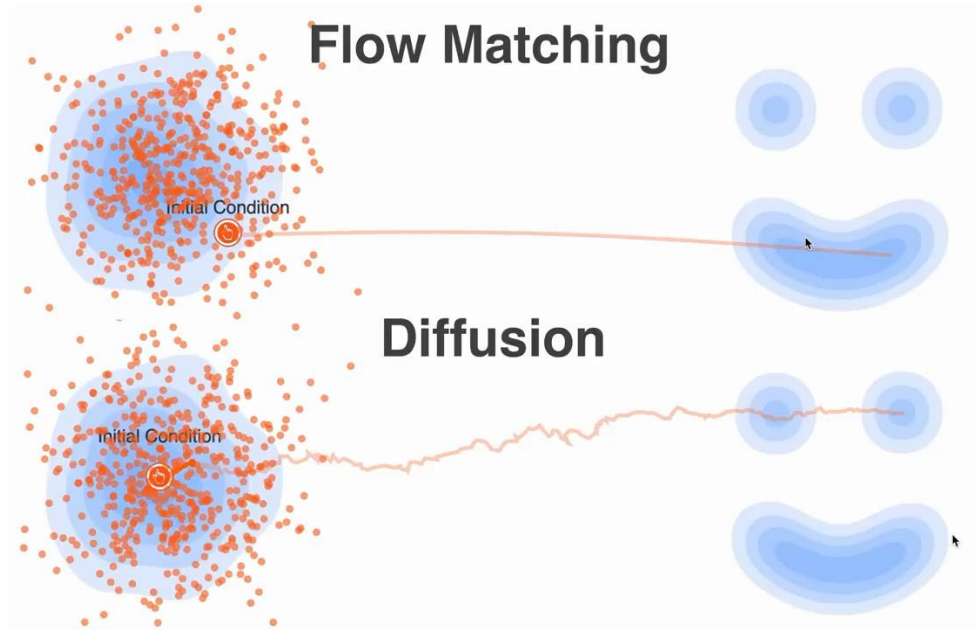
Flow Matching

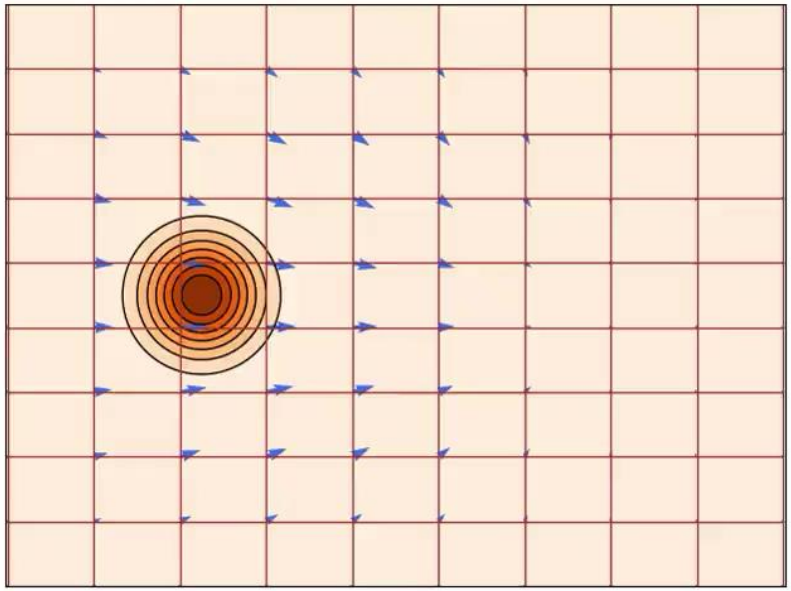
Flow and velocity

- $\frac{d}{dt}\Psi_t(x_t) = u_t\big(\Psi_t(x_t)\big)$
- continuity equation









Flow Matching

Flow Matching

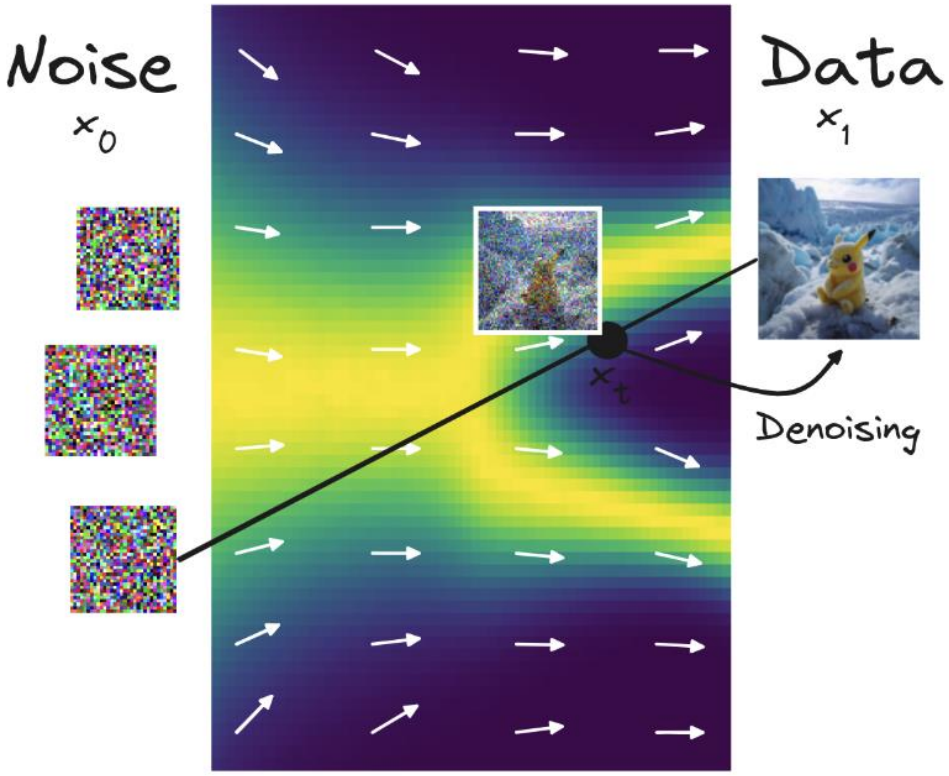
- Sample noise x_0, data x_1
- Interpolate with $t \in [0, 1]$

$$x_t = tx_1 + (1 - t)x_0$$

- Model the denoising direction

$$\mathrm{E}_{x_t,t}\left[x_1 \mid x_t, t\right]$$

- Flow v_θ points in that direction



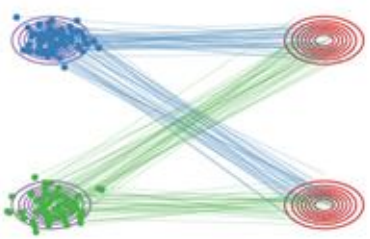
Lipman et al. 2023

Rectified flow

Flow matching have flexibility to design x_t

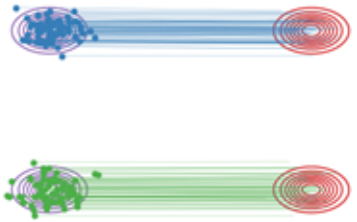
Rectified Flow: How about use linear path?

- Rectified flow (ICLR 23) was developed slightly before flow matching (ICLR 23)



Linear interpolation X_t

Rectified Flow Z_t



Straightened Rectified Flow

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

$$v(z,t) = \mathbb{E}[X_1 - X_0 \mid X_t = z],$$

Flow Straight and Fast:
Learning to Generate and Transfer Data with Rectified Flow

Xingchao Liu* University of Texas at Austin xcliu@utexas.edu

Chengyue Gong* University of Texas at Austin cygong@cs.utexas.edu

Qiang Liu University of Texas at Austin lqiang@cs.utexas.edu

Abstract

We present rectified flow, a surprisingly simple approach to learning (neural) ordinary differential equation (ODE) models to transport between two empirically observed distributions π_0 and π_1, hence providing a unified solution to generative modeling and domain transfer, among various other tasks involving distribution transport. The idea of rectified flow is to learn the ODE to follow the straight paths connecting the points drawn from π_0 and π_1 as much as possible. This is achieved by solving a straightforward nonlinear least squares optimization problem, which can be easily scaled to large models without introducing extra parameters beyond standard supervised learning. The straight paths are special and preferred because they are the shortest paths between two points, and can be simulated exactly without time discretization and hence yield computationally efficient models. We show that the procedure of learning a rectified flow from data, called rectification, turns an arbitrary coupling of π_0 and π_1 to a new deterministic coupling with provably non-increasing convex transport costs. In addition, recursively applying rectification allows us to obtain a sequence of flows with increasingly straight paths, which can be simulated accurately with coarse time discretization in the inference phase. In empirical studies, we show that rectified flow performs superbly on image generation, image-to-image translation, and domain adaptation. In particular, on image generation and translation, our method yields nearly straight flows that give high quality results even with *a single Euler discretization step*.

Image Credit: Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow Xingchao Liu, Chengyue Gong, Qiang Liu

Flow Matching

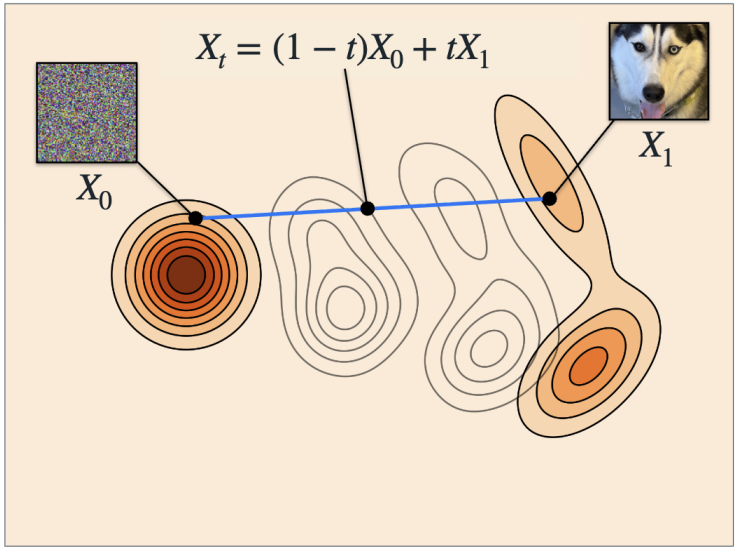
Nowadays, most flow matching uses linear path

- Marginal flow on $\mathbb{E}_{t,X_t}$ is hard. Use conditional flow of paired $\mathbb{E}_{t,(X_0,X_1)}$
- However, the prior distribution is not limited to Gaussian
- Related topic: Schrödinger bridge (in diffusion models), Optimal transport

- Arbitrary $X_0 \sim p, X_1 \sim q$
- Arbitrary coupling $(X_0, X_1) \sim \pi_{0,1}$

Why does it work?

- Build flow from conditional flows
- Regress conditional flows

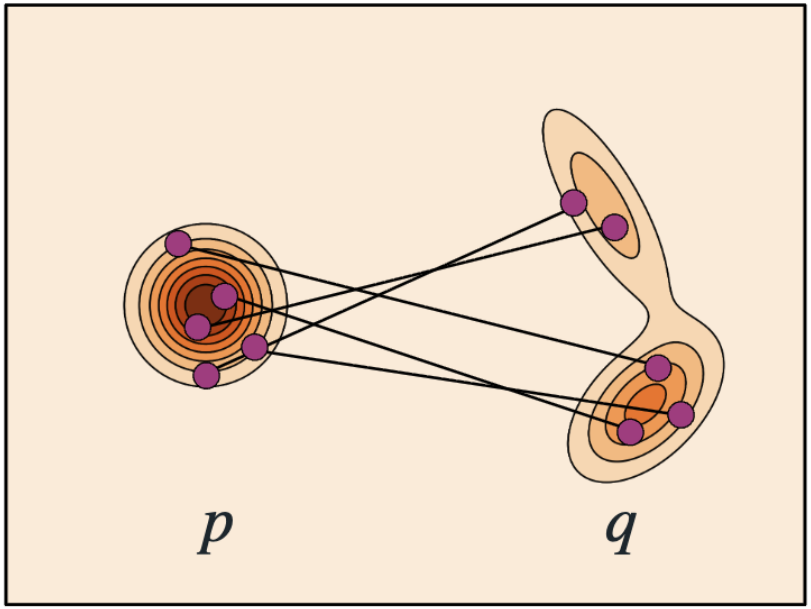


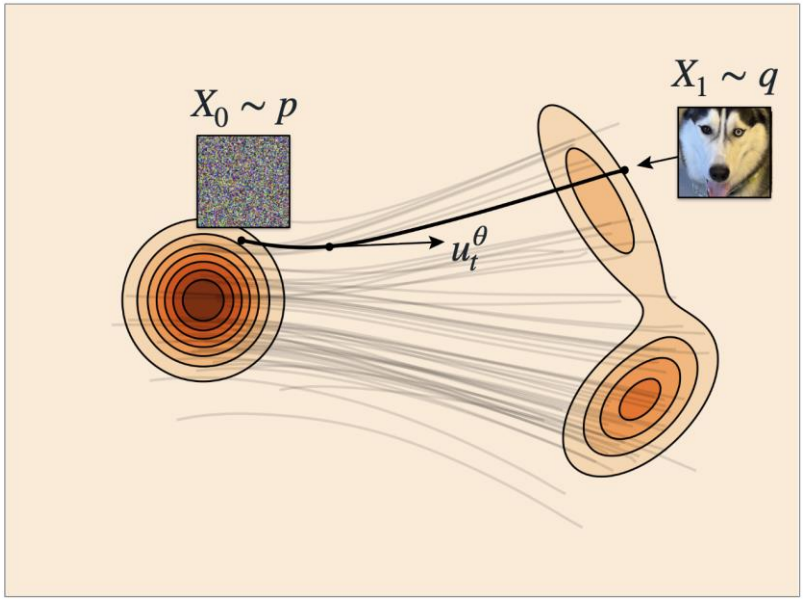
$$\mathbb{E}_{t,X_0,X_1}\left\| u_t^\theta(X_t) - (X_1 - X_0) \right\|^2$$

Flow Matching

For sampling, $\frac{d}{dt} X_t = u_t^\theta(X_t)$

- Use any ODE numerical solver (e.g., Midpoint)
- Straight line compared to diffusion (pros: fast, cons: less diversity)





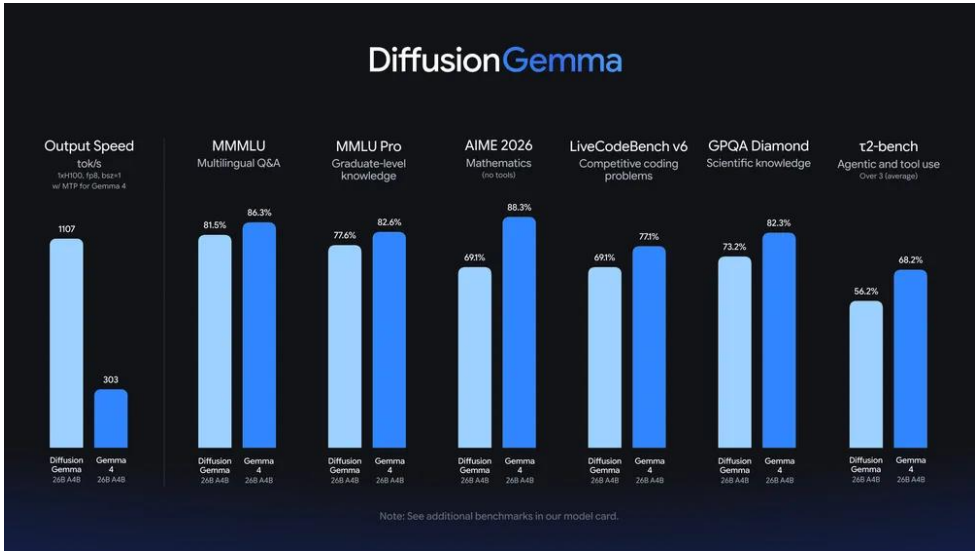
Contents

Diffusion + X

Discrete Diffusion Language Models

Diffusion Gemma (June 2026, Google)

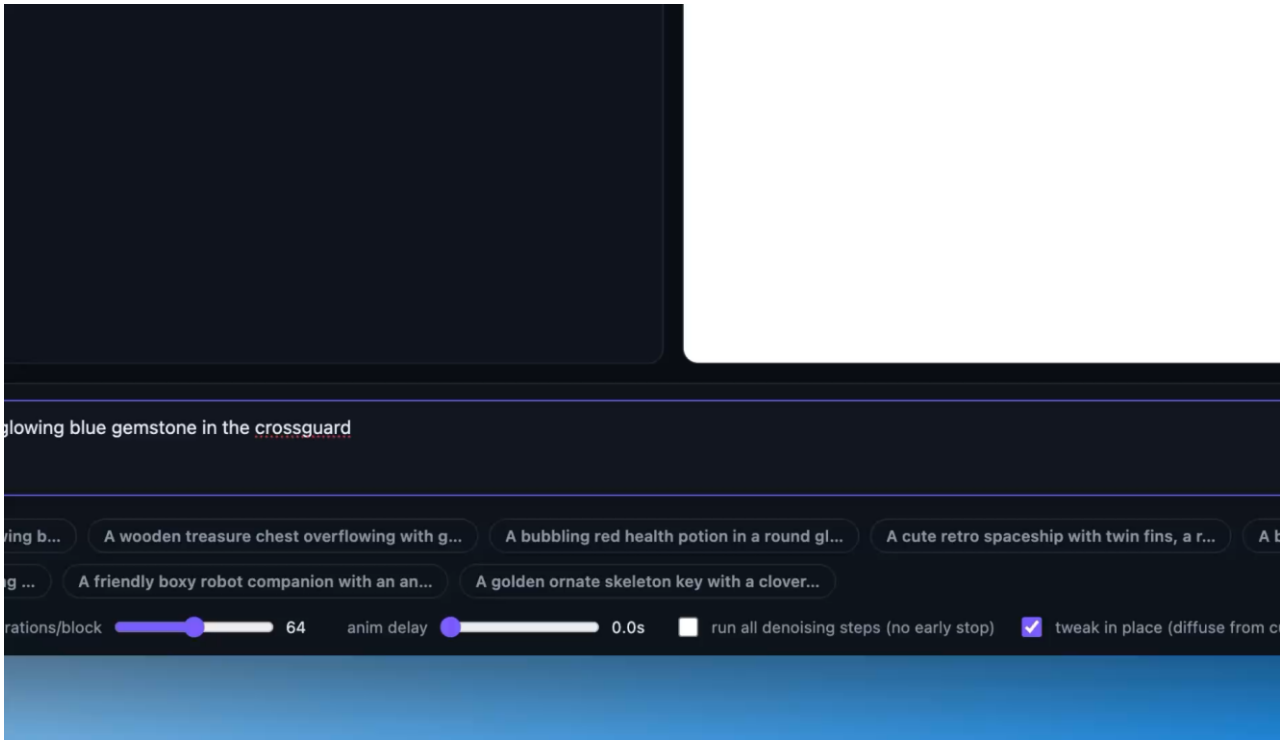
- https://storage.googleapis.com/gweb-uniblog-publish-prod/original_videos/dgemma_faster.mp4#t=0.001
- Faster generation speed compared to auto-regressive models



Discrete Diffusion Language Models

Diffusion Gemma (June 2026, Google)

- https://storage.googleapis.com/gweb-uniblog-publish-prod/original_videos/dgemma_faster.mp4#t=0.001
- Faster generation speed compared to auto-regressive models

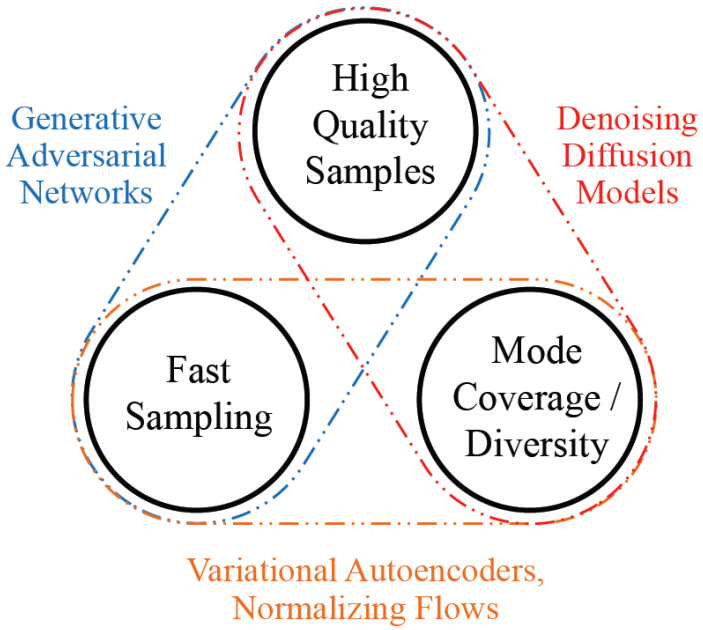


https://blog.google/innovation-and-ai/technology/developers-tools/diffusion-gemma-faster-text-generation/

Diffusion Generalization

Diffusion Models – Pros and Cons

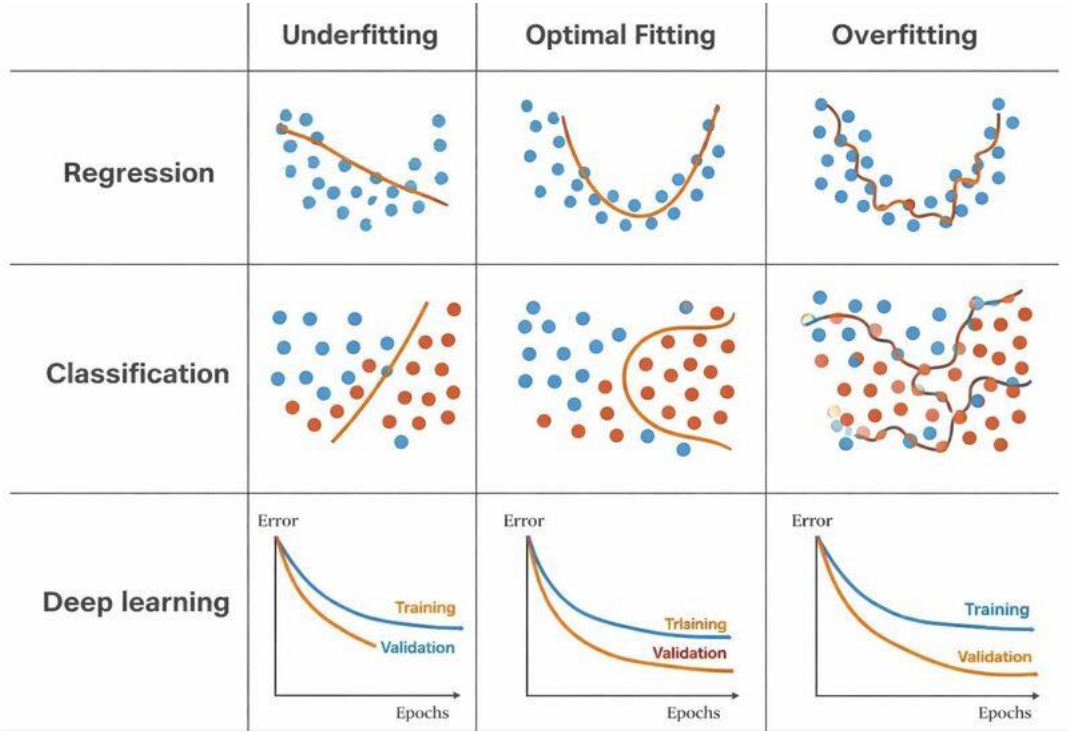
- (+) High quality samples
- (+) **Strong Diversity**
- (-) Very slow speed



Xiao, Zhisheng, Karsten Kreis, and Arash Vahdat. "Tackling the generative learning trilemma with denoising diffusion gans." *ICLR, 2022*.

Generalization

In supervised models, generalization means



https://www.linkedin.com/posts/israel-murimiro-41646a286_model-generalization-in-machine-learning-activity-7413534905101381632-u6xm

Generalization

In generative perspectives,

a diffusion model generalizes if it generates <u>novel</u> and <u>realistic</u> samples.

A diffusion model generalizes if

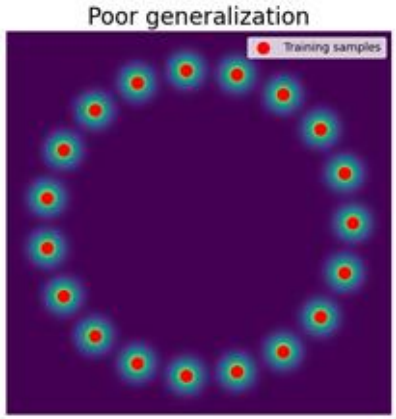
$$\mathop{\mathbb{E}}_{z\sim\mathcal{K}}\left[\log p_\theta(z)\right]$$

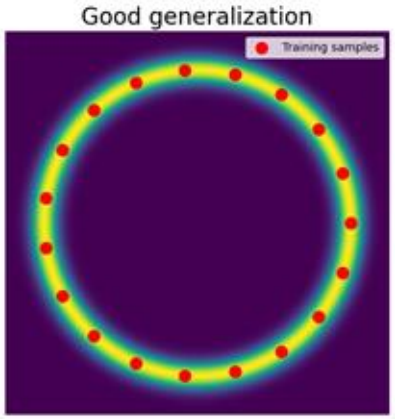
Average log-likelihood on the true distribution

is large relative to the empirical log-likelihood:

$$\frac{1}{n}\sum_{i=1}^{n}\log p_\theta(x_i)$$

Average log-likelihood on the training samples

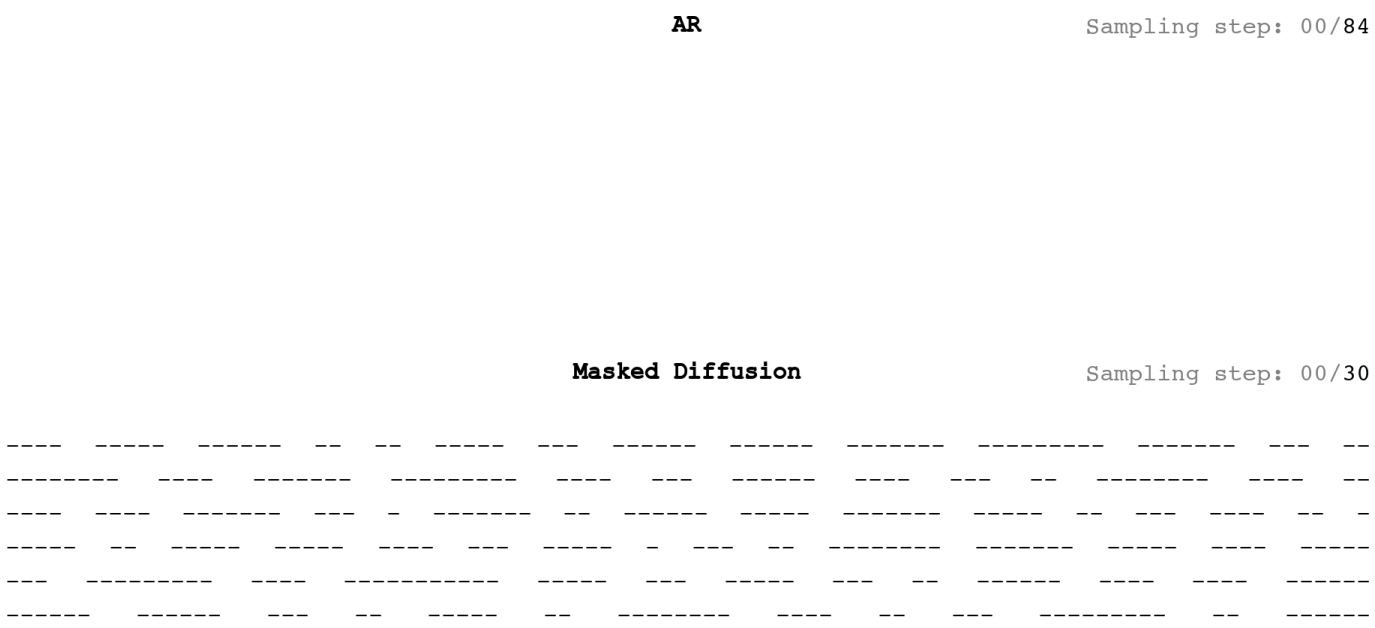




Hardness of training

Compared to autoregressive, training diffusion models have massive problems

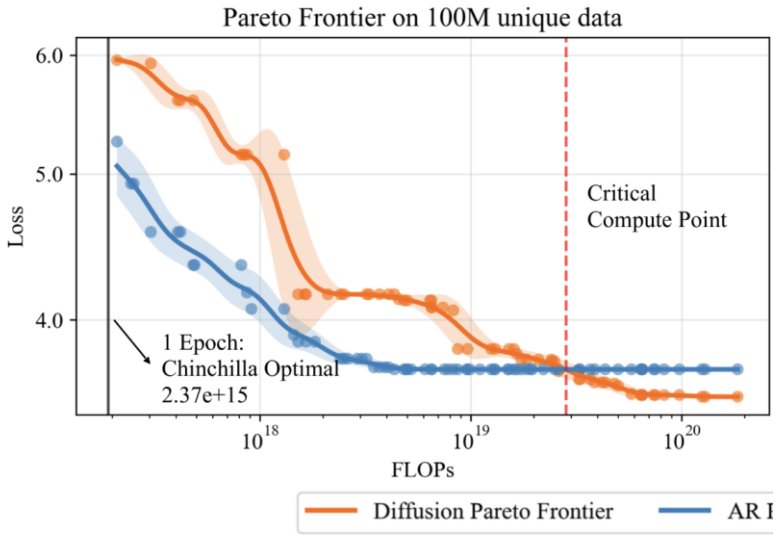
- Kim, Jaeyeon, et al. "Train for the Worst, Plan for the Best: Understanding Token Ordering in Masked Diffusions." *ICML 2026*.
- Prabhudesai, Mihir, et al. "Diffusion beats autoregressive in data-constrained settings." *Advances in Neural Information Processing Systems* 38 (2025).

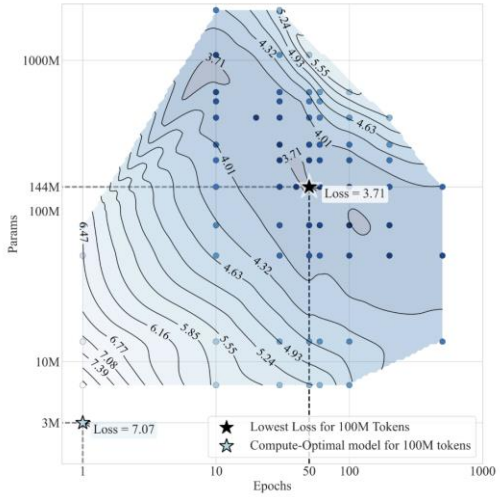


Hardness of training

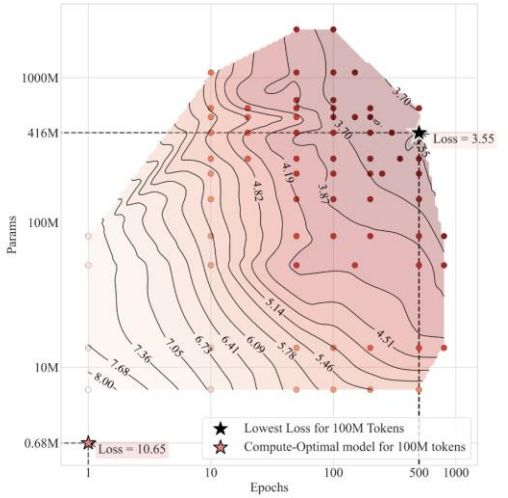
Compared to autoregressive, training diffusion models have massive problems

- Kim, Jaeyeon, et al. "Train for the Worst, Plan for the Best: Understanding Token Ordering in Masked Diffusions." *ICML 2026*.
- Prabhudesai, Mihir, et al. "Diffusion beats autoregressive in data-constrained settings." *Advances in Neural Information Processing Systems* 38 (2026).





Autoregressive: Best loss at 50 epochs (3.71)

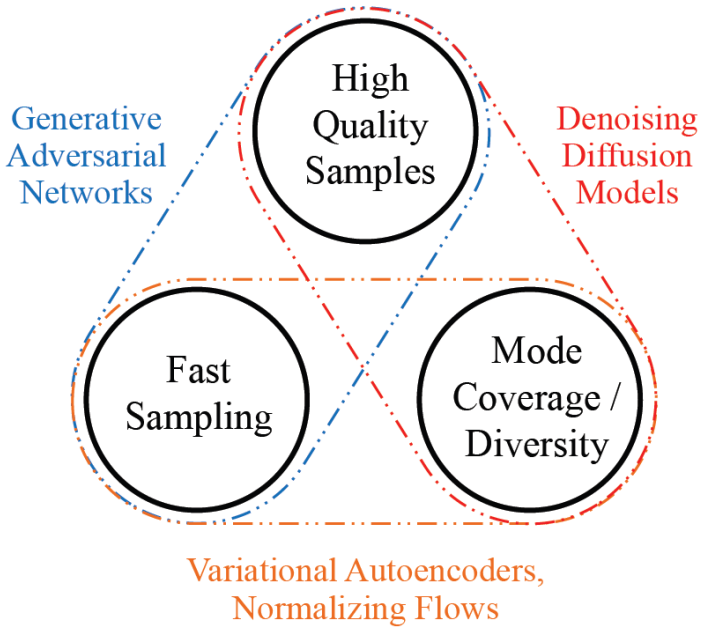


Diffusion: Best loss at 500 epochs (3.55)

Diffusion Generalization

Diffusion Models – Pros and Cons

- **(+) High quality samples**
- (+) Strong Diversity
- (-) Very slow speed



Xiao, Zhisheng, Karsten Kreis, and Arash Vahdat. "Tackling the generative learning trilemma with denoising diffusion gans." *ICLR, 2022*.

Code

https://github.com/JinseongP/cac2026_diffusion