Deep Generative Models

Lecture 9

Roman Isachenko

AI Masters

2026, Spring

Recap of Lecture 7

Let us perturb the original data with Gaussian noise $q(\mathbf{x}_\sigma|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2 \cdot \mathbf{I})$.

$$q(\mathbf{x}_\sigma) = \int q(\mathbf{x}_\sigma|\mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}.$$

Then the solution of

$$\frac{1}{2}\mathbb{E}_{q(\mathbf{x}_\sigma)}\left\|\mathbf{s}_{\boldsymbol{\theta},\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)\right\|_2^2 \to \min_{\boldsymbol{\theta}}$$

satisfies $\mathbf{s}_{\boldsymbol{\theta},\sigma}(\mathbf{x}_\sigma) \approx \mathbf{s}_{\boldsymbol{\theta},0}(\mathbf{x}_0) = \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$ if σ is sufficiently small.

Theorem (Denoising Score Matching)

$$\mathbb{E}_{q(\mathbf{x}_\sigma)}\left\|\mathbf{s}_{\boldsymbol{\theta},\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma)\right\|_2^2 =$$
$$= \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{q(\mathbf{x}_\sigma|\mathbf{x})}\left\|\mathbf{s}_{\boldsymbol{\theta},\sigma}(\mathbf{x}_\sigma) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x})\right\|_2^2 + \text{const}(\boldsymbol{\theta})$$

Here, $\nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma|\mathbf{x}) = -\frac{\mathbf{x}_\sigma - \mathbf{x}}{\sigma^2} = -\frac{\boldsymbol{\epsilon}}{\sigma}$. $\mathbf{s}_{\boldsymbol{\theta},\sigma}(\mathbf{x}_\sigma)$ attempts to **denoise** a corrupted sample.

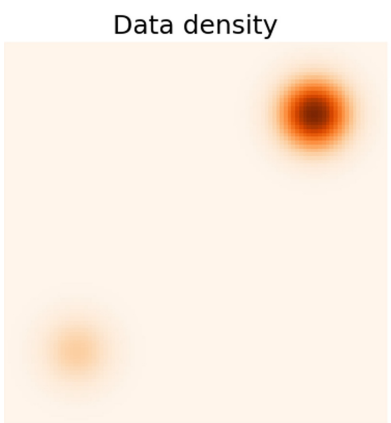
Vincent P. A Connection Between Score Matching and Denoising Autoencoders, 2010

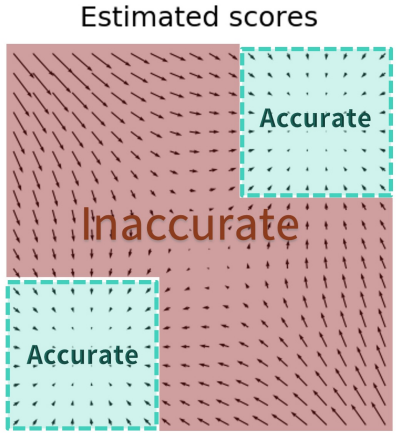
Recap of Lecture 7

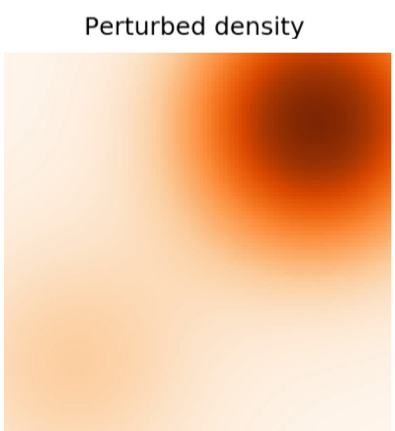
$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{\mathcal{N}(0,\mathbf{I})}\left\|\mathbf{s}_{\boldsymbol{\theta},\sigma}(\mathbf{x}+\sigma\boldsymbol{\epsilon})+\frac{\boldsymbol{\epsilon}}{\sigma}\right\|_2^2 \to \min_{\boldsymbol{\theta}}$$

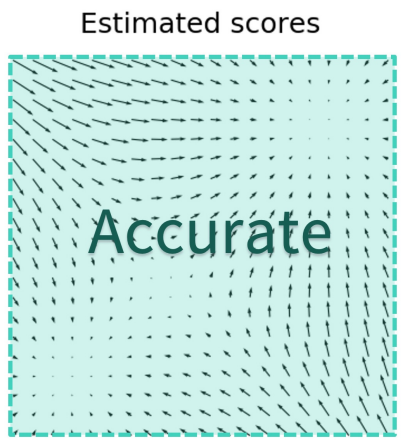
$$\mathbf{x}_{l+1} = \mathbf{x}_l + \frac{\eta}{2}\cdot\mathbf{s}_{\boldsymbol{\theta},\sigma}(\mathbf{x}_l) + \sqrt{\eta}\cdot\boldsymbol{\epsilon}_l$$

- For **small** σ, $\mathbf{s}_{\boldsymbol{\theta},\sigma}(\mathbf{x})$ becomes inaccurate and Langevin dynamics fails to traverse modes
- For **large** σ, robustness in low-density regions is achieved, but the model learns a distribution that is overly corrupted









Song Y. Generative Modeling by Estimating Gradients of the Data Distribution, blog post, 2021

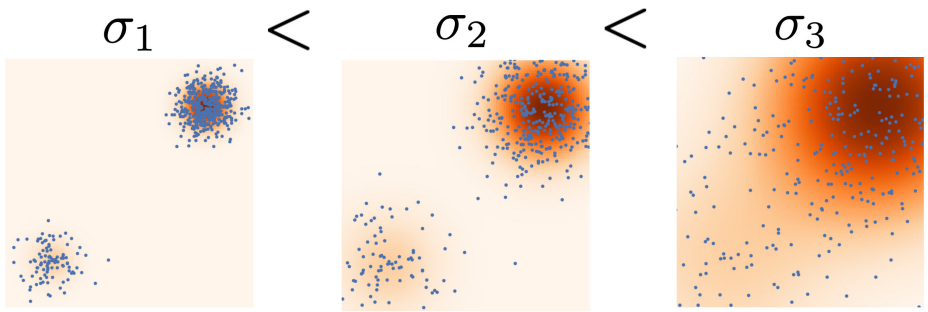
Recap of Lecture 7

Noise-Conditioned Score Network

- Define a sequence of noise levels: $\sigma_1 < \sigma_2 < \cdots < \sigma_T$.
- Train a denoised score function $\mathbf{s}_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t)$ for each noise level:

$$\sum_{t=1}^{T} \sigma_t^2 \cdot \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})} \left\| \mathbf{s}_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}) \right\|_2^2 \to \min_{\boldsymbol{\theta}}$$

- Sample using **annealed** Langevin dynamics (for $t = 1, \ldots, T$).



Song Y. et al. Generative Modeling by Estimating Gradients of the Data Distribution, 2019

Recap of Lecture 7

NCSN Training

1. Obtain a sample $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$.
2. Sample noise level $t \sim U\{1, T\}$ and noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.
3. Construct noisy image $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \cdot \boldsymbol{\epsilon}$.
4. Compute the loss $\mathcal{L} = \sigma_t^2 \cdot \|\mathbf{s}_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t) + \frac{\boldsymbol{\epsilon}}{\sigma_t}\|^2$.

NCSN Sampling (Annealed Langevin Dynamics)

- Sample $\mathbf{x}_0 \sim \mathcal{N}(0, \sigma_T^2 \cdot \mathbf{I}) \approx q(\mathbf{x}_T)$.
- Apply L steps of Langevin dynamics:

$$\mathbf{x}_l = \mathbf{x}_{l-1} + \frac{\eta_t}{2} \cdot \mathbf{s}_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_{l-1}) + \sqrt{\eta_t} \cdot \boldsymbol{\epsilon}_l.$$

- Update $\mathbf{x}_0 := \mathbf{x}_L$ and proceed to the next σ_t.

Song Y. et al. Improved Techniques for Training Score-Based Generative Models, 2020

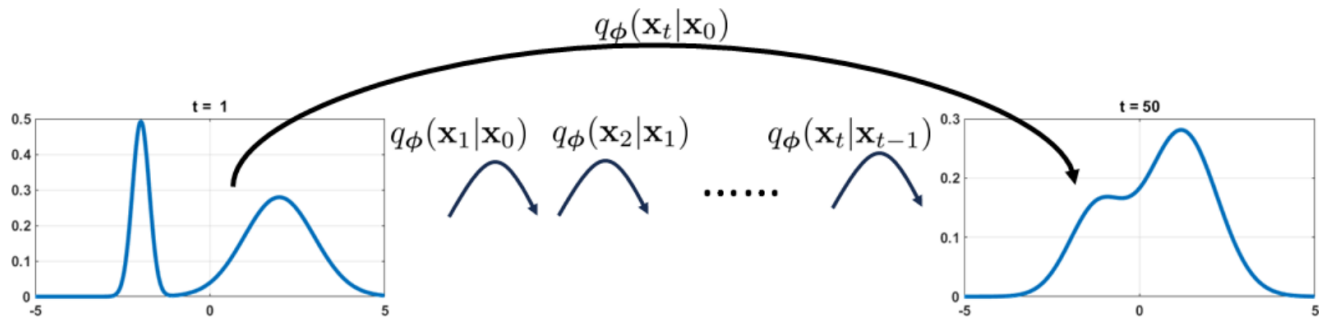
Recap of Lecture 7

Forward Gaussian Diffusion Process

Let $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$, $\beta_t \ll 1$, $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

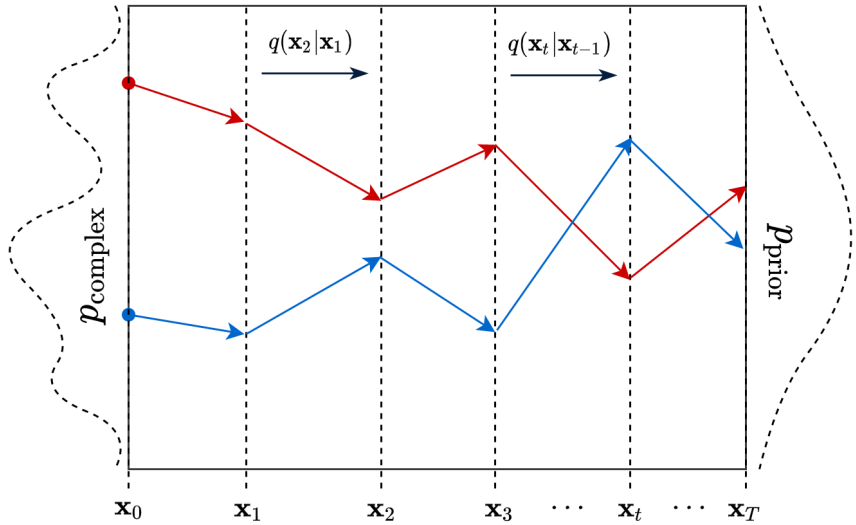
$$\mathbf{x}_t = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I});$$
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}).$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1}, \beta_t \cdot \mathbf{I});$$
$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0, (1-\bar{\alpha}_t) \cdot \mathbf{I}).$$



Recap of Lecture 7

Diffusion describes the process where particles migrate from regions of high density to regions of low density.

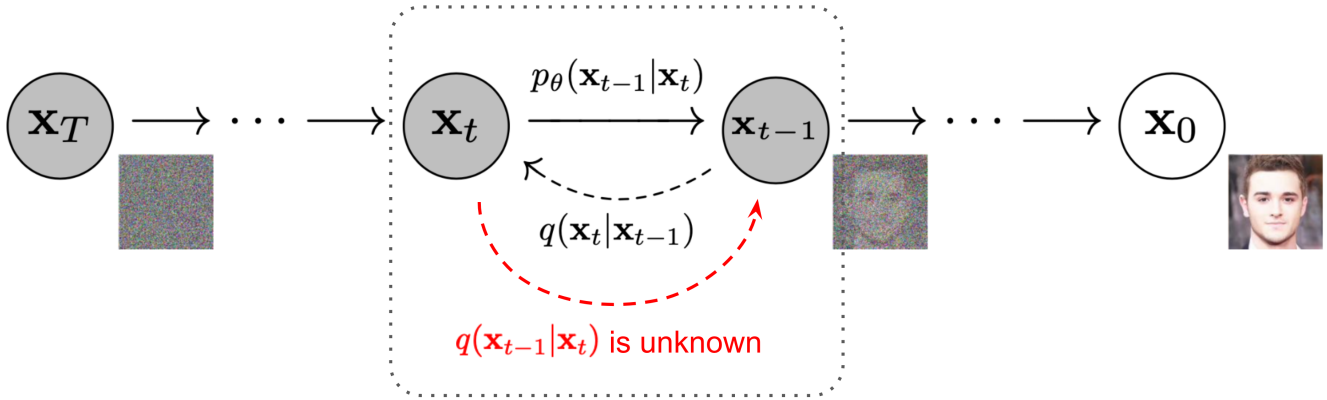


1. $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$;
2. $\mathbf{x}_t = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, $t \geq 1$;
3. $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$, for $T \gg 1$.

If we can invert this process, we would have a way to sample $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ using noise samples, i.e. $p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$.
Hence, our objective becomes to reverse this process.

Das A. An Introduction to Diffusion Probabilistic Models, blog post, 2021

Recap of Lecture 7



Reverse Process (Ancestral Sampling)

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)} \approx p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t), \boldsymbol{\sigma}^2_{\boldsymbol{\theta},t}(\mathbf{x}_t)\right)$$

The Feller theorem guarantees this approximation is valid.

Forward Process

1. $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$;
2. $\mathbf{x}_t = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}$;
3. $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$.

Reverse Process

1. $\mathbf{x}_T \sim p_\infty(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$;
2. $\mathbf{x}_{t-1} = \boldsymbol{\sigma}_{\boldsymbol{\theta},t}(\mathbf{x}_t) \cdot \boldsymbol{\epsilon} + \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$;
3. $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}(\mathbf{x})$;

Weng L. What are Diffusion Models?, blog post, 2021

Recap of Previous Lecture

Forward Process: Converts an arbitrary distribution $p_{\text{data}}(\mathbf{x})$ into the standard normal $\mathcal{N}(0, \mathbf{I})$ by incrementally adding noise.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t\,\mathbf{I});$$
$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0, (1-\bar{\alpha}_t)\,\mathbf{I}).$$

Reverse Process: This intractable distribution can be well-approximated by a Gaussian (with unknown parameters) when β_t is sufficiently small.

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)} \approx \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t), \boldsymbol{\sigma}^2_{\boldsymbol{\theta},t}(\mathbf{x}_t)\right)$$

Conditioned Reverse Process: This Gaussian, with known parameters, describes how to denoise a noisy image $\mathbf{x}_t$ when the final clean image $\mathbf{x}_0$ is known.

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\,\mathbf{I})$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

Recap of Previous Lecture

- $\mathbf{z} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ represents the latent variables.
- Variational posterior distribution:

$$q(\mathbf{z}|\mathbf{x}) = q(\mathbf{x}_1, \ldots, \mathbf{x}_T|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}).$$

- Generative model and prior:

$$p_{\theta}(\mathbf{x}|\mathbf{z}) = p_{\theta}(\mathbf{x}_0|\mathbf{x}_1); \quad p_{\theta}(\mathbf{z}) = \prod_{t=2}^{\top} p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) \cdot p(\mathbf{x}_T)$$

ELBO

$$\log p_{\theta}(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} = \mathcal{L}_{\phi,\theta}(\mathbf{x}) \to \max_{q,\theta}$$

$$\mathcal{L}_{\phi,\theta}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) - \mathrm{KL}\big(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T)\big) - \\ - \sum_{t=2}^{\top} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \mathrm{KL}\big(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\|p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)\big)}_{\mathcal{L}_t}$$

Das A. An introduction to Diffusion Probabilistic Models, blog post, 2021

Recap of Previous Lecture

ELBO of the Gaussian Diffusion Model

$$\mathcal{L}_{\phi,\theta}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) - \mathrm{KL}\big(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T)\big) - \\ - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \mathrm{KL}\big(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\big)}_{\mathcal{L}_t}$$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\big(\boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t), \boldsymbol{\sigma}^2_{\theta,t}(\mathbf{x}_t)\big)$$

It is assumed that $\boldsymbol{\sigma}^2_{\theta,t}(\mathbf{x}_t) = \tilde{\beta}_t \mathbf{I}$.

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \left\| \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t) \right\|^2 \right]$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

Recap of Previous Lecture

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\frac{1}{2\tilde{\beta}_t}\left\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t)\right\|^2\right]$$

Reparameterization

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t(1-\bar{\alpha}_t)}} \cdot \boldsymbol{\epsilon}$$

$$\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t(1-\bar{\alpha}_t)}} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$$

$$\mathcal{L}_t = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[\frac{(1-\alpha_t)^2}{2\tilde{\beta}_t\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta},t}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}\right)\right\|^2\right]$$

At each step in the reverse diffusion process, our goal is to predict the noise $\boldsymbol{\epsilon}$ added by the forward process!

Simplified Objective

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

Ho J. Denoising Diffusion Probabilistic Models, 2020

Recap of Previous Lecture

Training DDPM

1. Draw a sample $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$.
2. Sample a timestep $t \sim U\{1, T\}$ and noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.
3. Obtain the noisy image: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \boldsymbol{\epsilon}$.
4. Compute the loss: $\mathcal{L}_{\text{simple}} = \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)\|^2$.

Sampling with DDPM

1. Sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
2. Compute the mean of $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t), \sigma_t^2 \cdot \mathbf{I})$:

$$\boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$$

3. Generate a denoised image: $\mathbf{x}_{t-1} = \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

Ho J. Denoising Diffusion Probabilistic Models, 2020

Outline

Outline

Denoising Diffusion as a Score-Based Generative Model

DDPM Objective

$$\mathcal{L}_t = \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})} \left[C_{1,t} \cdot \| \boldsymbol{\epsilon}_{\boldsymbol{\theta}, t}(\mathbf{x}_t) - \boldsymbol{\epsilon} \|_2^2 \right]$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

Denoising Diffusion as a Score-Based Generative Model

DDPM Objective

$$\begin{aligned}\mathcal{L}_t &= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{1,t}\cdot\|\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\boldsymbol{\epsilon}\|_2^2\right]\\ &= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{2,t}\cdot\left\|\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)}{\sqrt{1-\bar{\alpha}_t}}-\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}\right\|_2^2\right]\end{aligned}$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

Denoising Diffusion as a Score-Based Generative Model

DDPM Objective

$$\mathcal{L}_t = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{1,t}\cdot\|\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\boldsymbol{\epsilon}\|_2^2\right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{2,t}\cdot\left\|\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)}{\sqrt{1-\bar{\alpha}_t}}-\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}\right\|_2^2\right]$$

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

Ho J. Denoising Diffusion Probabilistic Models, 2020

Denoising Diffusion as a Score-Based Generative Model

DDPM Objective

$$\mathcal{L}_t = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{1,t}\cdot\|\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\boldsymbol{\epsilon}\|_2^2\right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{2,t}\cdot\left\|\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)}{\sqrt{1-\bar{\alpha}_t}}-\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}\right\|_2^2\right]$$

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

$$\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0) = -\frac{\mathbf{x}_t-\sqrt{\bar{\alpha}_t}\cdot\mathbf{x}_0}{1-\bar{\alpha}_t} = -\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}.$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

Denoising Diffusion as a Score-Based Generative Model

DDPM Objective

$$\mathcal{L}_t = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{1,t}\cdot\|\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\boldsymbol{\epsilon}\|_2^2\right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{2,t}\cdot\left\|\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)}{\sqrt{1-\bar{\alpha}_t}}-\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}\right\|_2^2\right]$$

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

$$\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0) = -\frac{\mathbf{x}_t-\sqrt{\bar{\alpha}_t}\cdot\mathbf{x}_0}{1-\bar{\alpha}_t} = -\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}.$$

We can reparameterize the model as:

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

Ho J. Denoising Diffusion Probabilistic Models, 2020

Denoising Diffusion as a Score-Based Generative Model

DDPM Objective

$$\mathcal{L}_t = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{1,t}\cdot\|\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\boldsymbol{\epsilon}\|_2^2\right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[C_{2,t}\cdot\left\|\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)}{\sqrt{1-\bar{\alpha}_t}}-\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}\right\|_2^2\right]$$

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

$$\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0) = -\frac{\mathbf{x}_t-\sqrt{\bar{\alpha}_t}\cdot\mathbf{x}_0}{1-\bar{\alpha}_t} = -\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}.$$

We can reparameterize the model as:

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

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[C_{2,t}\cdot\left\|\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0)\right\|_2^2\right]$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

DDPM vs NCSN: Objectives

DDPM Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)}\mathbb{E}_{t\sim U\{1,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[C_{2,t}\left\|\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0)\right\|_2^2\right]$$

$$\mathbf{x}_t=\sqrt{\bar{\alpha}_t}\cdot\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\cdot\boldsymbol{\epsilon}$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

DDPM vs NCSN: Objectives

DDPM Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)}\mathbb{E}_{t\sim U\{1,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[C_{2,t}\left\|\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0)\right\|_2^2\right]$$

$$\mathbf{x}_t=\sqrt{\bar{\alpha}_t}\cdot\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\cdot\boldsymbol{\epsilon}$$

In practice, this coefficient is often omitted.

Ho J. Denoising Diffusion Probabilistic Models, 2020

DDPM vs NCSN: Objectives

DDPM Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)}\mathbb{E}_{t\sim U\{1,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[C_{2,t}\left\|\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0)\right\|_2^2\right]$$

$$\mathbf{x}_t=\sqrt{\bar{\alpha}_t}\cdot\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\cdot\boldsymbol{\epsilon}$$

In practice, this coefficient is often omitted.

NCSN Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)}\mathbb{E}_{t\sim U\{1,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left\|\mathbf{s}_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t)-\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0)\right\|_2^2$$

$$\mathbf{x}_t=\mathbf{x}_0+\sigma_t\cdot\boldsymbol{\epsilon}$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

DDPM vs NCSN: Objectives

DDPM Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)}\mathbb{E}_{t\sim U\{1,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[C_{2,t}\left\|\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t)-\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0)\right\|_2^2\right]$$

$$\mathbf{x}_t=\sqrt{\bar{\alpha}_t}\cdot\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\cdot\boldsymbol{\epsilon}$$

In practice, this coefficient is often omitted.

NCSN Objective

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)}\mathbb{E}_{t\sim U\{1,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left\|\mathbf{s}_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t)-\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0)\right\|_2^2$$

$$\mathbf{x}_t=\mathbf{x}_0+\sigma_t\cdot\boldsymbol{\epsilon}$$

Maximizing the ELBO leads to the same objective as denoising score matching!

Ho J. Denoising Diffusion Probabilistic Models, 2020

DDPM vs NCSN: Sampling

DDPM Sampling (Ancestral Sampling)

$$\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$$

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

DDPM vs NCSN: Sampling

DDPM Sampling (Ancestral Sampling)

$$
\begin{aligned}
\mathbf{x}_T &\sim \mathcal{N}(0, \mathbf{I}) \\
\mathbf{x}_{t-1} &= \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\
&= \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t(1-\bar{\alpha}_t)}} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}
\end{aligned}
$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

DDPM vs NCSN: Sampling

DDPM Sampling (Ancestral Sampling)

$$
\begin{aligned}
\mathbf{x}_T &\sim \mathcal{N}(0, \mathbf{I}) \\
\mathbf{x}_{t-1} &= \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\
&= \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t(1-\bar{\alpha}_t)}} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\
&= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}
\end{aligned}
$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

DDPM vs NCSN: Sampling

DDPM Sampling (Ancestral Sampling)

$$
\begin{aligned}
\mathbf{x}_T &\sim \mathcal{N}(0, \mathbf{I}) \\
\mathbf{x}_{t-1} &= \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\
&= \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t(1-\bar{\alpha}_t)}} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\
&= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}
\end{aligned}
$$

NCSN Sampling (Annealed Langevin Dynamics)

- Sample $\mathbf{x}_T^0 \sim \mathcal{N}(0, \sigma_T^2 \mathbf{I}) \approx q(\mathbf{x}_T)$.
- Perform L steps of Langevin dynamics:

$$\mathbf{x}_t^l = \mathbf{x}_t^{l-1} + \frac{\eta_t}{2} \cdot \mathbf{s}_{\boldsymbol{\theta},\sigma_t}(\mathbf{x}_t^{l-1}) + \sqrt{\eta_t} \cdot \boldsymbol{\epsilon}_t^l.$$

- Set $\mathbf{x}_{t-1}^0 = \mathbf{x}_t^L$ and move to the next σ_t.

Ho J. Denoising Diffusion Probabilistic Models, 2020

DDPM vs NCSN: Summary

Summary

- Different Markov chains:
  - DDPM: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \boldsymbol{\epsilon}$;
  - NCSN: $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \cdot \boldsymbol{\epsilon}$.
  - One can generalize to $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\alpha_t \cdot \mathbf{x}_0, \sigma_t^2 \mathbf{I})$.

Kingma D. et al. Variational Diffusion Models, 2021
Song Y. et al. Score-Based Generative Modeling through Stochastic Differential Equations, 2020

DDPM vs NCSN: Summary

Summary

- Different Markov chains:
  - DDPM: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \boldsymbol{\epsilon}$;
  - NCSN: $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \cdot \boldsymbol{\epsilon}$.
  - One can generalize to $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\alpha_t \cdot \mathbf{x}_0, \sigma_t^2 \mathbf{I})$.
- The objectives coincide: ELBO $\equiv$ score-matching.

Kingma D. et al. Variational Diffusion Models, 2021
Song Y. et al. Score-Based Generative Modeling through Stochastic Differential Equations, 2020

DDPM vs NCSN: Summary

Summary

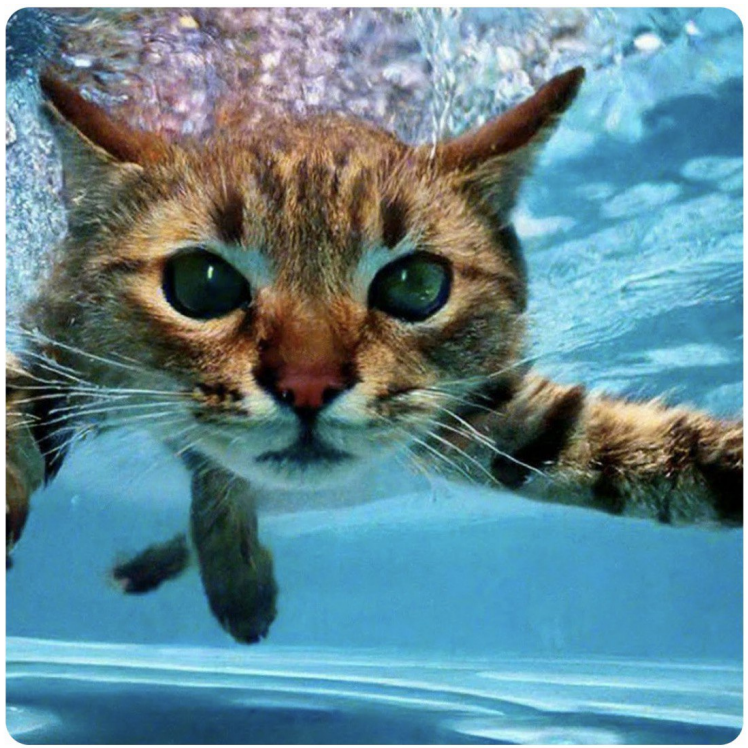
- Different Markov chains:
  - DDPM: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \boldsymbol{\epsilon}$;
  - NCSN: $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \cdot \boldsymbol{\epsilon}$.
  - One can generalize to $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\alpha_t \cdot \mathbf{x}_0, \sigma_t^2 \mathbf{I})$.
- The objectives coincide: ELBO $\equiv$ score-matching.
- The sampling procedures differ:
  - Ancestral sampling in DDPM;
  - Annealed Langevin dynamics for NCSN;
  - Hybrid approaches that combine both updates are possible.

Kingma D. et al. Variational Diffusion Models, 2021
Song Y. et al. Score-Based Generative Modeling through Stochastic Differential Equations, 2020

Outline

Guidance

- Up to now, we have focused on **unconditional** generative models $p_\theta(\mathbf{x})$.
- In practice, most generative models are **conditional** (in diffusion era it is called guided): $p_\theta(\mathbf{x}|\mathbf{y})$.
- Here, $\mathbf{y}$ might denote a class label or **text** (as in text-to-image tasks).



Кот ныряет в бассейн, как ребенок на обложке альбома Nevermind, реалистично

рука человека с пятью пальцами, ни четыремя, ни шестью, а с 5 (пять) пальцами

Conditional Models

In practice, we're typically interested in learning conditional models (sampling from conditional distribution $p_{\text{data}}(\mathbf{x}|\mathbf{y})$).

- $\mathbf{y} = \emptyset$, $\mathbf{x}$ = image $\Rightarrow$ unconditional image model
- $\mathbf{y}$ = class label, $\mathbf{x}$ = image $\Rightarrow$ class-conditional image model
- $\mathbf{y}$ = text prompt, $\mathbf{x}$ = image $\Rightarrow$ text-to-image model
- $\mathbf{y}$ = image, $\mathbf{x}$ = image $\Rightarrow$ image-to-image model
- $\mathbf{y}$ = image, $\mathbf{x}$ = text $\Rightarrow$ image-to-text (image captioning) model
- $\mathbf{y}$ = English text, $\mathbf{x}$ = Russian text $\Rightarrow$ sequence-to-sequence model (machine translation) model
- $\mathbf{y}$ = sound, $\mathbf{x}$ = text $\Rightarrow$ speech-to-text (automatic speech recognition) model
- $\mathbf{y}$ = text, $\mathbf{x}$ = sound $\Rightarrow$ text-to-speech model

Label Guidance

Label: Ostrich (10th ImageNet class)

VQ-VAE (Proposed) **BigGAN deep**

Razavi A., Oord A., et al. Generating Diverse High-Fidelity Images with VQ-VAE-2, 2019

Text Guidance

Prompt: a stained glass window of a panda eating bamboo
Left: $\gamma = 1$, Right: $\gamma = 3$.

Nichol A., et al. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models, 2022

Guidance in Generative Models

How to make guided model?

Instead of sampling from $p_{\theta}(\mathbf{x})$, we sample from $p_{\theta}(\mathbf{x}|\mathbf{y})$.

Guidance in Generative Models

How to make guided model?

Instead of sampling from $p_{\theta}(\mathbf{x})$, we sample from $p_{\theta}(\mathbf{x}|\mathbf{y})$.

Given **supervised** data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, we can treat $\mathbf{y}$ as an additional model input:

Guidance in Generative Models

How to make guided model?

Instead of sampling from $p_{\theta}(\mathbf{x})$, we sample from $p_{\theta}(\mathbf{x}|\mathbf{y})$.

Given **supervised** data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, we can treat $\mathbf{y}$ as an additional model input:

- $p_{\theta}(x_j|\mathbf{x}_{1:j-1}, \mathbf{y})$ for AR models;

Guidance in Generative Models

How to make guided model?

Instead of sampling from $p_{\theta}(\mathbf{x})$, we sample from $p_{\theta}(\mathbf{x}|\mathbf{y})$.

Given **supervised** data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, we can treat $\mathbf{y}$ as an additional model input:

- $p_{\theta}(x_j|\mathbf{x}_{1:j-1}, \mathbf{y})$ for AR models;
- Encoder $q_{\phi}(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and decoder $p_{\theta}(\mathbf{x}|\mathbf{z}, \mathbf{y})$ for VAEs;

Guidance in Generative Models

How to make guided model?

Instead of sampling from $p_{\theta}(\mathbf{x})$, we sample from $p_{\theta}(\mathbf{x}|\mathbf{y})$.

Given **supervised** data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, we can treat $\mathbf{y}$ as an additional model input:

- $p_{\theta}(x_j|\mathbf{x}_{1:j-1}, \mathbf{y})$ for AR models;
- Encoder $q_{\phi}(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and decoder $p_{\theta}(\mathbf{x}|\mathbf{z}, \mathbf{y})$ for VAEs;
- $G_{\theta}(\mathbf{z}, \mathbf{y})$ for NFs and GANs;

Guidance in Generative Models

How to make guided model?

Instead of sampling from $p_{\theta}(\mathbf{x})$, we sample from $p_{\theta}(\mathbf{x}|\mathbf{y})$.

Given **supervised** data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, we can treat $\mathbf{y}$ as an additional model input:

- $p_{\theta}(x_j|\mathbf{x}_{1:j-1}, \mathbf{y})$ for AR models;
- Encoder $q_{\phi}(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and decoder $p_{\theta}(\mathbf{x}|\mathbf{z}, \mathbf{y})$ for VAEs;
- $G_{\theta}(\mathbf{z}, \mathbf{y})$ for NFs and GANs;
- $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$ for DDPMs.

Guidance in Generative Models

How to make guided model?

Instead of sampling from $p_{\theta}(\mathbf{x})$, we sample from $p_{\theta}(\mathbf{x}|\mathbf{y})$.

Given **supervised** data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, we can treat $\mathbf{y}$ as an additional model input:

- $p_{\theta}(x_j|\mathbf{x}_{1:j-1}, \mathbf{y})$ for AR models;
- Encoder $q_{\phi}(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and decoder $p_{\theta}(\mathbf{x}|\mathbf{z}, \mathbf{y})$ for VAEs;
- $G_{\theta}(\mathbf{z}, \mathbf{y})$ for NFs and GANs;
- $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$ for DDPMs.

Challenge

- Empirically, images sampled with this procedure do not fit well enough to the desired label $\mathbf{y}$.
- Being able to control the strength of guidance is especially valuable.

Outline

Classifier Guidance

DDPM Sampling

1. Sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
2. Generate the denoised image (unconditional generation):

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance

DDPM Sampling

1. Sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
2. Generate the denoised image (unconditional generation):

$$\begin{aligned}\mathbf{x}_{t-1} &= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\ &= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}\end{aligned}$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance

DDPM Sampling

1. Sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
2. Generate the denoised image (unconditional generation):

$$\begin{aligned}\mathbf{x}_{t-1} &= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\ &= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}\end{aligned}$$

Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance

DDPM Sampling

1. Sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
2. Generate the denoised image (unconditional generation):

$$\begin{aligned}\mathbf{x}_{t-1} &= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon} \\ &= \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + \sigma_t \cdot \boldsymbol{\epsilon}\end{aligned}$$

Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

What is the link between $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t)$ and $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y})$?

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guided Score Function

Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guided Score Function

Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

Guided Generation

$$\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log \left(\frac{p_\theta(\mathbf{x}_t) p(\mathbf{y}|\mathbf{x}_t)}{p(\mathbf{y})} \right)$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guided Score Function

Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) + \sigma_t \cdot \epsilon$$

Guided Generation

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log \left(\frac{p_\theta(\mathbf{x}_t) p(\mathbf{y}|\mathbf{x}_t)}{p(\mathbf{y})} \right) \\ &= \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)\end{aligned}$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guided Score Function

Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) + \sigma_t \cdot \epsilon$$

Guided Generation

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log \left(\frac{p_\theta(\mathbf{x}_t) p(\mathbf{y}|\mathbf{x}_t)}{p(\mathbf{y})} \right) \\
&= \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \\
&= \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)
\end{aligned}$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guided Score Function

Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) + \sigma_t \cdot \epsilon$$

Guided Generation

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log \left(\frac{p_\theta(\mathbf{x}_t) p(\mathbf{y}|\mathbf{x}_t)}{p(\mathbf{y})} \right) \\
&= \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \\
&= \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)
\end{aligned}$$

Guided Score Function

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}).$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guided Score Function

Guided Generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) + \sigma_t \cdot \epsilon$$

Guided Generation

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log \left(\frac{p_\theta(\mathbf{x}_t) p(\mathbf{y}|\mathbf{x}_t)}{p(\mathbf{y})} \right) \\
&= \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \\
&= \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)
\end{aligned}$$

Guided Score Function

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}).$$

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guidance Scale

Guided Score Function

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guidance Scale

Guided Score Function

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

- Let us assume $\mathbf{y}$ is a class label.
- $p(\mathbf{y}|\mathbf{x}_t)$ is a classifier for noisy inputs.
- $p(\mathbf{y}|\mathbf{x}_t)$ is responsible for model guidance.

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guidance Scale

Guided Score Function

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

- Let us assume $\mathbf{y}$ is a class label.
- $p(\mathbf{y}|\mathbf{x}_t)$ is a classifier for noisy inputs.
- $p(\mathbf{y}|\mathbf{x}_t)$ is responsible for model guidance.

Guidance Scale

It is a natural idea to scale up the contribution of the guidance

$$\mathbf{s}^{\gamma}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Guidance Scale

Guided Score Function

$$\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

- Let us assume $\mathbf{y}$ is a class label.
- $p(\mathbf{y}|\mathbf{x}_t)$ is a classifier for noisy inputs.
- $p(\mathbf{y}|\mathbf{x}_t)$ is responsible for model guidance.

Guidance Scale

It is a natural idea to scale up the contribution of the guidance

$$\mathbf{s}^{\gamma}_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\theta,t}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

- The guidance scale γ adjusts the strength of classifier guidance.
- $\mathbf{s}^{\gamma}_{\theta,t}(\mathbf{x}_t, \mathbf{y})$ is not the true guided score function $\mathbf{s}_{\theta,t}(\mathbf{x}_t, \mathbf{y})$.

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Distribution Sharpening

Scaled Guided Score Function

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Distribution Sharpening

Scaled Guided Score Function

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Scaled Conditional Distribution

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Distribution Sharpening

Scaled Guided Score Function

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Scaled Conditional Distribution

$$\begin{aligned}\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \\ &= \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)^{\gamma}\end{aligned}$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Distribution Sharpening

Scaled Guided Score Function

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Scaled Conditional Distribution

$$\begin{aligned}
\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \\
&= \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)^{\gamma} \\
&= \nabla_{\mathbf{x}_t} \log \left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_t) p(\mathbf{y}|\mathbf{x}_t)^{\gamma}}{Z} \right)
\end{aligned}$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Distribution Sharpening

Scaled Guided Score Function

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Scaled Conditional Distribution

$$\begin{aligned}
\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \\
&= \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)^{\gamma} \\
&= \nabla_{\mathbf{x}_t} \log \left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_t) p(\mathbf{y}|\mathbf{x}_t)^{\gamma}}{Z} \right)
\end{aligned}$$

Note: Increasing γ sharpens $p(\mathbf{y}|\mathbf{x}_t)$, making it more contrast

$$\hat{p}(\mathbf{y}|\mathbf{x}_t) \propto p(\mathbf{y}|\mathbf{x}_t)^{\gamma}.$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Overview

Training

- Train the DDPM as before.
- Train an additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ on noisy data (note that it is dependent on time t).

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Classifier Guidance: Overview

Training

- Train the DDPM as before.
- Train an additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ on noisy data (note that it is dependent on time t).

Guided DDPM Sampling

1. Sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
2. Generate the denoised image (using scaled guided score function):

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Outline

Classifier-Free Guidance

- The previous approach relies on training an additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ for noisy images.
- We now introduce a method to sidestep this requirement.

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance

- The previous approach relies on training an additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ for noisy images.
- We now introduce a method to sidestep this requirement.

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance

- The previous approach relies on training an additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ for noisy images.
- We now introduce a method to sidestep this requirement.

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Bayes theorem

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log \left(\frac{p_{\theta}(\mathbf{x}_t|\mathbf{y}) p(\mathbf{y})}{p_{\theta}(\mathbf{x}_t)} \right)$$

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance

- The previous approach relies on training an additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ for noisy images.
- We now introduce a method to sidestep this requirement.

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Bayes theorem

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log \left(\frac{p_{\theta}(\mathbf{x}_t|\mathbf{y}) p(\mathbf{y})}{p_{\theta}(\mathbf{x}_t)} \right) \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t)\end{aligned}$$

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance

- The previous approach relies on training an additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ for noisy images.
- We now introduce a method to sidestep this requirement.

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Bayes theorem

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log \left(\frac{p_{\theta}(\mathbf{x}_t|\mathbf{y}) p(\mathbf{y})}{p_{\theta}(\mathbf{x}_t)} \right) \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t)\end{aligned}$$

Scaled Guided Score Function

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance

- The previous approach relies on training an additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ for noisy images.
- We now introduce a method to sidestep this requirement.

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Bayes theorem

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log \left(\frac{p_{\theta}(\mathbf{x}_t|\mathbf{y})p(\mathbf{y})}{p_{\theta}(\mathbf{x}_t)} \right) \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t)\end{aligned}$$

Scaled Guided Score Function

$$\begin{aligned}&\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) = \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \left(\nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) \right)\end{aligned}$$

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance

- The previous approach relies on training an additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ for noisy images.
- We now introduce a method to sidestep this requirement.

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Bayes theorem

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log \left(\frac{p_{\theta}(\mathbf{x}_t|\mathbf{y})p(\mathbf{y})}{p_{\theta}(\mathbf{x}_t)} \right) \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t)\end{aligned}$$

Scaled Guided Score Function

$$\begin{aligned}&\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) = \\ &= \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \left(\nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) \right) = \\ &= (1-\gamma) \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y})\end{aligned}$$

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance: Formulation

Scaled Guided Score Function

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y}) = (1-\gamma) \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p_{\theta}(\mathbf{x}_t|\mathbf{y})$$

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance: Formulation

Scaled Guided Score Function

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}) = (1-\gamma) \cdot \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y})$$

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = (1-\gamma) \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$$

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance: Formulation

Scaled Guided Score Function

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}) = (1-\gamma) \cdot \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y})$$

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = (1-\gamma) \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$$

Naive training approach

- Train an unguided score function model $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$.
- Train a guided score function model $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$.

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance: Formulation

Scaled Guided Score Function

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}) = (1-\gamma) \cdot \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y})$$

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = (1-\gamma) \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$$

Naive training approach

- Train an unguided score function model $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$.
- Train a guided score function model $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$.
- Use their convex combination at inference.

Guided Sampling

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance: Formulation

Scaled Guided Score Function

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}) = (1-\gamma) \cdot \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y})$$

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = (1-\gamma) \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$$

Naive training approach

- Train an unguided score function model $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t)$.
- Train a guided score function model $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$.
- Use their convex combination at inference.

Guided Sampling

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

How to avoid training two separate score function models?

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Classifier-Free Guidance

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = (1-\gamma) \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$$

CFG algorithm

- Introduce "the absence of conditioning" label $\mathbf{y} = \varnothing$.

Holderrieth P., Erives E. An Introduction to Flow Matching and Diffusion Models, 2025

Classifier-Free Guidance

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = (1-\gamma) \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$$

CFG algorithm

- Introduce "the absence of conditioning" label $\mathbf{y} = \varnothing$.
- Use it to get unguided score function $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) = \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \varnothing)$.

Holderrieth P., Erives E. An Introduction to Flow Matching and Diffusion Models, 2025

Classifier-Free Guidance

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = (1-\gamma) \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$$

CFG algorithm

- Introduce "the absence of conditioning" label $\mathbf{y} = \varnothing$.
- Use it to get unguided score function $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) = \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \varnothing)$.
- Train a single model $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$ using **supervised** data, but artificially drop the labels $\mathbf{y}$ with some fixed probability (simulating the case of $\mathbf{y} = \varnothing$).

Holderrieth P., Erives E. An Introduction to Flow Matching and Diffusion Models, 2025

Classifier-Free Guidance

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = (1-\gamma) \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$$

CFG algorithm

- Introduce "the absence of conditioning" label $\mathbf{y} = \varnothing$.
- Use it to get unguided score function $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) = \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \varnothing)$.
- Train a single model $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$ using **supervised** data, but artificially drop the labels $\mathbf{y}$ with some fixed probability (simulating the case of $\mathbf{y} = \varnothing$).
- Apply the model twice during inference to get $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \varnothing)$ and $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$.

Holderrieth P., Erives E. An Introduction to Flow Matching and Diffusion Models, 2025

Classifier-Free Guidance

$$\mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = (1-\gamma) \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) + \gamma \cdot \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$$

CFG algorithm

- Introduce "the absence of conditioning" label $\mathbf{y} = \varnothing$.
- Use it to get unguided score function $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) = \mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \varnothing)$.
- Train a single model $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$ using **supervised** data, but artificially drop the labels $\mathbf{y}$ with some fixed probability (simulating the case of $\mathbf{y} = \varnothing$).
- Apply the model twice during inference to get $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \varnothing)$ and $\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$.

Guided Sampling

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \mathbf{s}^{\gamma}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}$$

Holderrieth P., Erives E. An Introduction to Flow Matching and Diffusion Models, 2025

Outline

Discrete-Time Normalizing Flows

Change of Variable Theorem (CoV)

Let $\mathbf{x}$ be a random variable with density $p(\mathbf{x})$, and let $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$ be a differentiable and **invertible** transformation. If $\mathbf{z} = \mathbf{f}(\mathbf{x})$, $\mathbf{x} = \mathbf{f}^{-1}(\mathbf{z}) = \mathbf{g}(\mathbf{z})$, then

$$p(\mathbf{x}) = p(\mathbf{z}) |\det(\mathbf{J_f})| = p(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \left| \det \left(\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

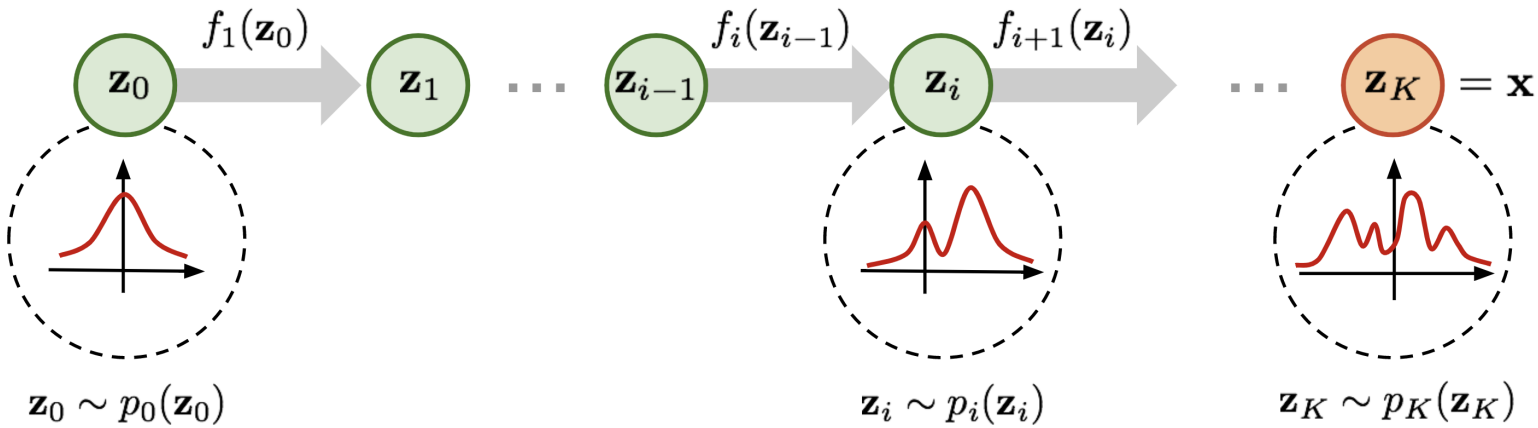
https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Discrete-Time Normalizing Flows

Change of Variable Theorem (CoV)

Let $\mathbf{x}$ be a random variable with density $p(\mathbf{x})$, and let $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$ be a differentiable and **invertible** transformation. If $\mathbf{z} = \mathbf{f}(\mathbf{x})$, $\mathbf{x} = \mathbf{f}^{-1}(\mathbf{z}) = \mathbf{g}(\mathbf{z})$, then

$$p(\mathbf{x}) = p(\mathbf{z}) |\det(\mathbf{J_f})| = p(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \left| \det \left(\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$



$$\log p_\theta(\mathbf{x}) = \log p(\mathbf{f}_K \circ \cdots \circ \mathbf{f}_1(\mathbf{x})) + \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}} \right) \right|.$$

https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Towards Continuous-Time Normalizing Flows

- Up to this point, we have considered discrete-time normalizing flows:

$$\mathbf{x}_{t+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t); \quad \log p(\mathbf{x}_{t+1}) = \log p(\mathbf{x}_t) - \log \left| \det \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_t)}{\partial \mathbf{x}_t} \right|.$$

- Let us now move to the general case of continuous time, using a mapping $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^m$ to describe continuous dynamics.

Towards Continuous-Time Normalizing Flows

- Up to this point, we have considered discrete-time normalizing flows:

$$\mathbf{x}_{t+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t); \quad \log p(\mathbf{x}_{t+1}) = \log p(\mathbf{x}_t) - \log \left| \det \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_t)}{\partial \mathbf{x}_t} \right|.$$

- Let us now move to the general case of continuous time, using a mapping $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^m$ to describe continuous dynamics.

Continuous-Time Dynamics

Consider an Ordinary Differential Equation (ODE):

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0.$$

Towards Continuous-Time Normalizing Flows

- Up to this point, we have considered discrete-time normalizing flows:

$$\mathbf{x}_{t+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t); \quad \log p(\mathbf{x}_{t+1}) = \log p(\mathbf{x}_t) - \log \left| \det \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_t)}{\partial \mathbf{x}_t} \right|.$$

- Let us now move to the general case of continuous time, using a mapping $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^m$ to describe continuous dynamics.

Continuous-Time Dynamics

Consider an Ordinary Differential Equation (ODE):

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0.$$

$$\mathbf{x}(t_1) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) dt + \mathbf{x}_0$$

Here, $\mathbf{f}_{\boldsymbol{\theta}} : \mathbb{R}^m \times [t_0, t_1] \to \mathbb{R}^m$ is a **vector field**.

Ordinary Differential Equations (ODEs)

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0.$$

$$\mathbf{x}(t_1) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt + \mathbf{x}_0$$

Flow

Let call **the flow** $\boldsymbol{\psi} : \mathbb{R}^m \times [t_0, t_1] \to \mathbb{R}^m$ the solution of ODE:

$$\frac{d\boldsymbol{\psi}_t(\mathbf{x}_0)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\mathbf{x}_0), t); \quad \text{with initial condition } \boldsymbol{\psi}_0(\mathbf{x}_0) = \mathbf{x}_0.$$

Numerical Solution of ODEs

$$\boldsymbol{\psi}_t(\mathbf{x}_0) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt + \mathbf{x}_0$$

Ordinary Differential Equations (ODEs)

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0.$$

$$\mathbf{x}(t_1) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt + \mathbf{x}_0$$

Flow

Let call **the flow** $\boldsymbol{\psi} : \mathbb{R}^m \times [t_0, t_1] \to \mathbb{R}^m$ the solution of ODE:

$$\frac{d\boldsymbol{\psi}_t(\mathbf{x}_0)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\mathbf{x}_0), t); \quad \text{with initial condition } \boldsymbol{\psi}_0(\mathbf{x}_0) = \mathbf{x}_0.$$

Numerical Solution of ODEs

$$\boldsymbol{\psi}_t(\mathbf{x}_0) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt + \mathbf{x}_0 \approx \texttt{ODESolve}_f(\mathbf{x}_0, \boldsymbol{\theta}, t_0, t_1).$$

Ordinary Differential Equations (ODEs)

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0.$$

$$\mathbf{x}(t_1) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt + \mathbf{x}_0$$

Flow

Let call **the flow** $\boldsymbol{\psi} : \mathbb{R}^m \times [t_0, t_1] \to \mathbb{R}^m$ the solution of ODE:

$$\frac{d\boldsymbol{\psi}_t(\mathbf{x}_0)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\mathbf{x}_0), t); \quad \text{with initial condition } \boldsymbol{\psi}_0(\mathbf{x}_0) = \mathbf{x}_0.$$

Numerical Solution of ODEs

$$\boldsymbol{\psi}_t(\mathbf{x}_0) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt + \mathbf{x}_0 \approx \texttt{ODESolve}_f(\mathbf{x}_0, \boldsymbol{\theta}, t_0, t_1).$$

Here, we require the numerical routine $\texttt{ODESolve}_f(\mathbf{x}_0, \boldsymbol{\theta}, t_0, t_1)$.

Numerical Solution of ODEs

$$\psi_t(\mathbf{x}_0) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt + \mathbf{x}_0 \approx \texttt{ODESolve}_f(\mathbf{x}_0, \boldsymbol{\theta}, t_0, t_1).$$

$\texttt{ODESolve}_f(\mathbf{x}_0, \boldsymbol{\theta}, t_0, t_1)$ consists of sequence of iterative update steps.

Image credit: https://en.wikipedia.org/wiki/Heun's_method

Numerical Solution of ODEs

$$\psi_t(\mathbf{x}_0) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt + \mathbf{x}_0 \approx \texttt{ODESolve}_f(\mathbf{x}_0, \boldsymbol{\theta}, t_0, t_1).$$

$\texttt{ODESolve}_f(\mathbf{x}_0, \boldsymbol{\theta}, t_0, t_1)$ consists of sequence of iterative update steps.

Euler Update Step

$$\frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)$$

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h \cdot \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)$$

Image credit: https://en.wikipedia.org/wiki/Heun's_method

Numerical Solution of ODEs

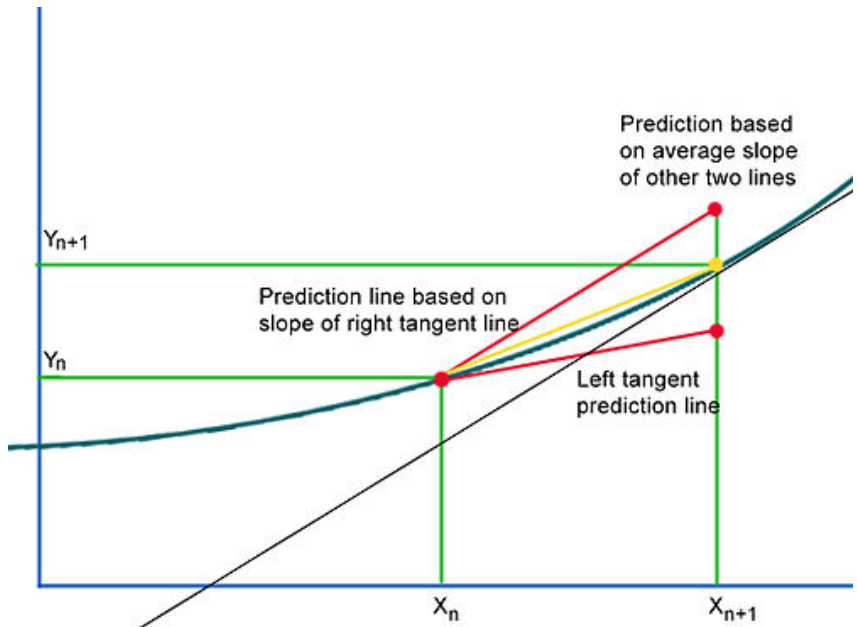
$$\psi_t(\mathbf{x}_0) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt + \mathbf{x}_0 \approx \texttt{ODESolve}_f(\mathbf{x}_0, \boldsymbol{\theta}, t_0, t_1).$$

$\texttt{ODESolve}_f(\mathbf{x}_0, \boldsymbol{\theta}, t_0, t_1)$ consists of sequence of iterative update steps.

Euler Update Step

$$\frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)$$

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h \cdot \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)$$



Heun's Update Step

$$\mathbf{x}'(t+h) = \mathbf{x}(t) + h \cdot \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)$$

$$\mathbf{x}(t+h) = \mathbf{x}(t) + \frac{h}{2} \cdot \left(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) + \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}'(t+h), t+h)\right)$$

Image credit: https://en.wikipedia.org/wiki/Heun's_method

Continuous-Time Normalizing Flows: Neural ODE

Neural ODE

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0$$

Euler `ODESolve`

$$\mathbf{x}(t + h) = \mathbf{x}(t) + h \cdot \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)$$

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Continuous-Time Normalizing Flows: Neural ODE

Neural ODE

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0$$

Euler `ODESolve`

$$\mathbf{x}(t + h) = \mathbf{x}(t) + h \cdot \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)$$

- Consider $[t_0, t_1] = [0, 1]$ for simplicity.
- If $\mathbf{x}(0)$ is a random variable with density $p_0(\mathbf{x})$,
- Then, for any t, $\mathbf{x}(t)$ is a random variable with density $p_t(\mathbf{x})$.

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Continuous-Time Normalizing Flows: Intuition

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0$$

Grathwohl W. et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models, 2018

Continuous-Time Normalizing Flows: Intuition

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0$$

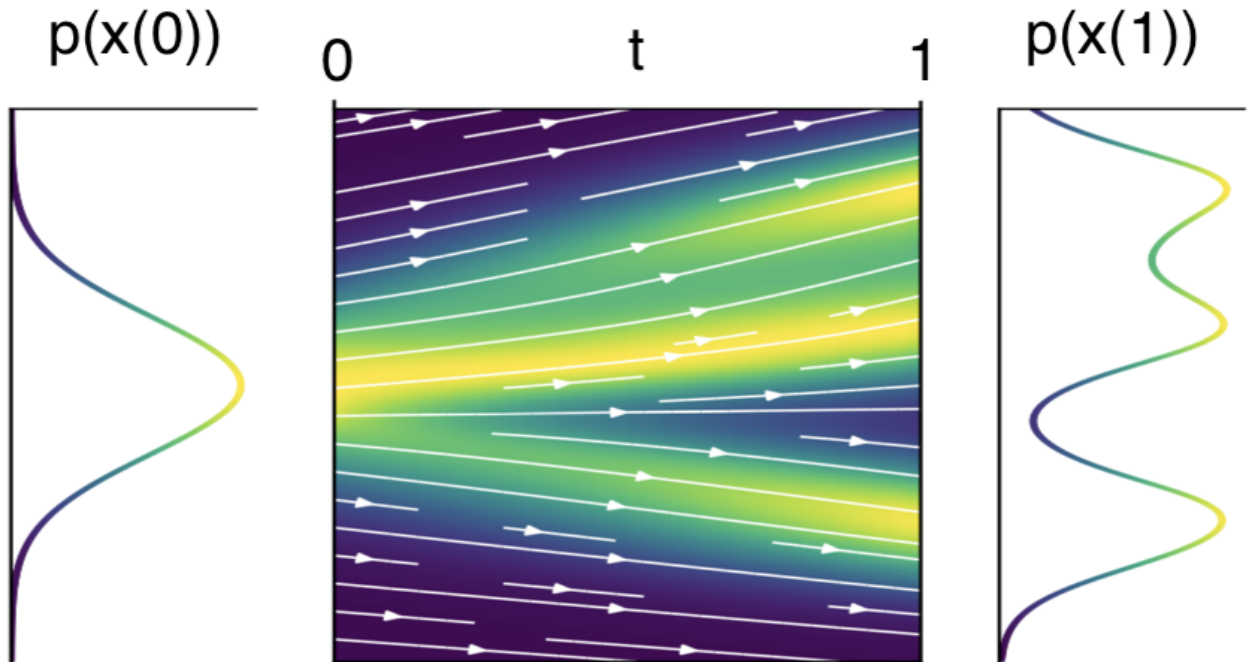
- $p_t(\mathbf{x}) = p(\mathbf{x}, t)$ describes the **probability path** interpolating between $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$.
- What is the difference between $p_t(\mathbf{x}(t))$ and $p_t(\mathbf{x})$?

Grathwohl W. et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models, 2018

Continuous-Time Normalizing Flows: Intuition

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0$$

- $p_t(\mathbf{x}) = p(\mathbf{x}, t)$ describes the **probability path** interpolating between $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$.
- What is the difference between $p_t(\mathbf{x}(t))$ and $p_t(\mathbf{x})$?



Grathwohl W. et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models, 2018

Continuous-Time Normalizing Flows: Reversibility

Theorem (Picard)

If $\mathbf{f}$ is continuously differentiable with a bounded derivative in $\mathbf{x}$ and continuous in t, then the ODE has a **unique solution** given by a flow $\boldsymbol{\psi}_t$.

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Continuous-Time Normalizing Flows: Reversibility

Theorem (Picard)

If $\mathbf{f}$ is continuously differentiable with a bounded derivative in $\mathbf{x}$ and continuous in t, then the ODE has a **unique solution** given by a flow $\boldsymbol{\psi}_t$.

This guarantees the ODE is **uniquely reversible**.

$$\boldsymbol{\psi}_1(\mathbf{x}_0) = \mathbf{x}_0 + \int_0^1 \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\mathbf{x}_0), t)dt$$

$$\mathbf{x}(1) = \mathbf{x}(0) + \int_0^1 \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt$$

$$\mathbf{x}(0) = \mathbf{x}(1) + \int_1^0 \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt$$

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Continuous-Time Normalizing Flows: Reversibility

Theorem (Picard)

If $\mathbf{f}$ is continuously differentiable with a bounded derivative in $\mathbf{x}$ and continuous in t, then the ODE has a **unique solution** given by a flow $\boldsymbol{\psi}_t$.

This guarantees the ODE is **uniquely reversible**.

$$\boldsymbol{\psi}_1(\mathbf{x}_0) = \mathbf{x}_0 + \int_0^1 \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\mathbf{x}_0), t)dt$$

$$\mathbf{x}(1) = \mathbf{x}(0) + \int_0^1 \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt$$

$$\mathbf{x}(0) = \mathbf{x}(1) + \int_1^0 \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt$$

Note: Unlike discrete-time flows, $\mathbf{f}$ need not be invertible (uniqueness ensures bijection).

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Continuous-Time Normalizing Flows: Reversibility

Theorem (Picard)

If $\mathbf{f}$ is continuously differentiable with a bounded derivative in $\mathbf{x}$ and continuous in t, then the ODE has a **unique solution** given by a flow $\boldsymbol{\psi}_t$.

This guarantees the ODE is **uniquely reversible**.

$$\boldsymbol{\psi}_1(\mathbf{x}_0) = \mathbf{x}_0 + \int_0^1 \mathbf{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\mathbf{x}_0), t)dt$$

$$\mathbf{x}(1) = \mathbf{x}(0) + \int_0^1 \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt$$

$$\mathbf{x}(0) = \mathbf{x}(1) + \int_1^0 \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)dt$$

Note: Unlike discrete-time flows, $\mathbf{f}$ need not be invertible (uniqueness ensures bijection). How can we compute $p_t(\mathbf{x})$ at arbitrary t?

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Summary

- DDPM and NCSN are intimately connected at the objective level.
- Classifier guidance provides a technique to turn an unconditional model into a conditional one by training an auxiliary classifier on noisy data.
- Classifier-free guidance removes the need for such a classifier, yielding a practical recipe now widely used.
- Continuous-time normalizing flows leverage neural ODEs to define continuous-time trajectories $\mathbf{x}(t)$, relaxing many constraints of discrete-time flows.
- If $\mathbf{x}_0$ is a random variable, this yields a **probability path** $p_t(\mathbf{x})$ as time evolves.