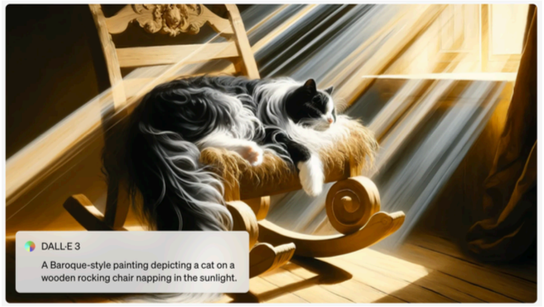
Diffusion Models

(Score-based generative models, Flow matching)

Yinbo Chen

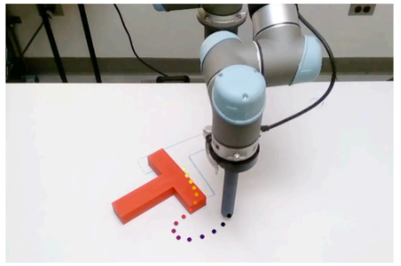
Generative modeling



Image

Video

Audio



Policy

Task definition

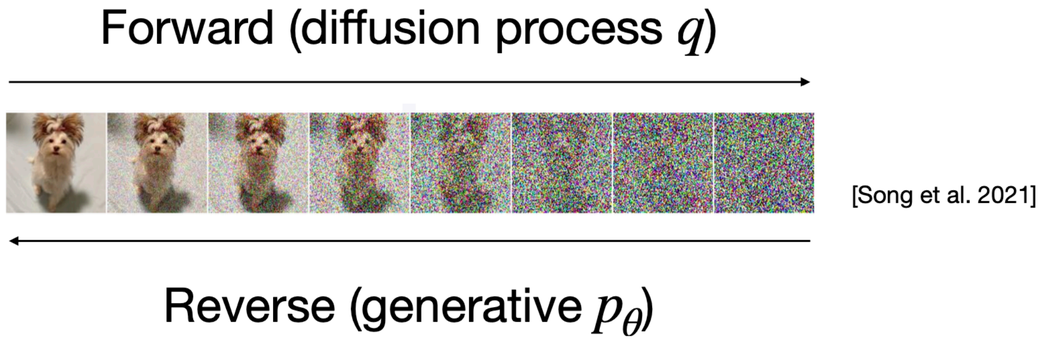
- Given a dataset of samples $x \sim q(x)$
- x is a vector in general (e.g., image: dimension = $h * w * 3$)
- Learn a model that generates samples from $p_\theta(x) \approx q(x)$

Overview

- **Basics:** discrete time, continuous time (SDE / ODE)
- **Applications:** acceleration, conditional generation
- Score distillation
- (Topics not covered: DreamBooth, engineering …)

Denoising Diffusion Probabilistic Models

“Generate by denoising”



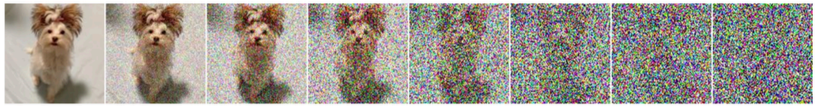
[Song et al. 2021]

- Forward process defines joint distribution $q(x_0, \ldots, x_T)$
- Learn model p_θ to reverse: sample noise x_T, denoise step by step
- (θ is the network parameters, p_θ denotes the generated distribution)

Forward process q

Gradually add noise

Forward (diffusion process q)

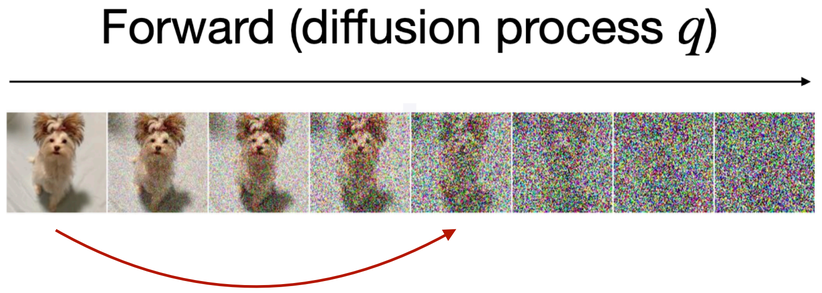


[Song et al. 2021]

- Defined by a set of β_t
- $q(x_t \mid x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$
  - i.e., $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\varepsilon \quad \varepsilon \sim \mathcal{N}(0, I)$
- $\sqrt{1-\beta_t}$ is just to preserve the variance
  - Assume $Var(x_0) = 1$, after each step Var is still 1

Forward process q

Add noise

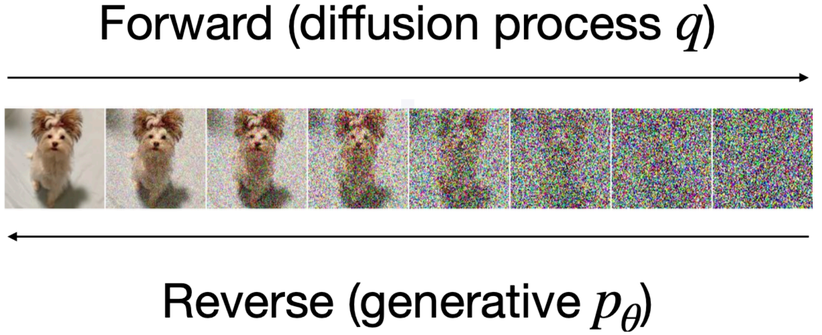


[Song et al. 2021]

- Defined by a set of β_t
- $q(x_t \mid x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$
  - i.e., $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\varepsilon$
- Let $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$
- $q(x_t \mid x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$

Reverse process p_θ

Denoise



[Song et al. 2021]

- How to model $q(x_{t-1} \mid x_t)$?
  - It is Gaussian if β_t are small enough

- $p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2 I)$
  - Use mean network μ_θ to define p_θ (empirically $\sigma_t^2 = \beta_t$ is good [Ho et al. 2020])
- Model defined, how to train it?

Training loss $\mathscr{L}$: maximize data likelihood

Evidence lower bound (ELBO)

- **ELBO**: $\log p_\theta(x) \geq E_{z\sim q(z|x)}[\log \frac{p_\theta(x,z)}{q(z\,|\,x)}] =: \mathscr{L}$
- In diffusion models: "x" is x_0, "z" is $x_{1\ldots T}$.
- DDPM [Ho et al. 2020] shows with $\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t,t))$,

  $\mathscr{L} = E_{x_0,t,\varepsilon}[w_t\,||\,\varepsilon - \varepsilon_\theta(x_t,t)\,||^2]$, where $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$

DDPM [Ho et al. 2020]

Discard w_t in $\mathcal{L}$ empirically works well

- DDPM [Ho et al. 2020] shows with $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t, t))$,

  $\mathcal{L} = E_{x_0,t,\varepsilon}[w_t || \varepsilon - \varepsilon_\theta(x_t, t) ||^2]$, where $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$

Algorithm 1 Training

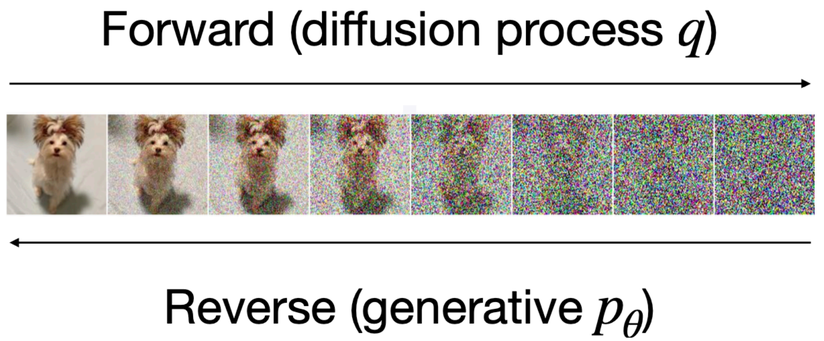
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

Algorithm 2 Sampling

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

Score-based generative models

SDE in continuous time



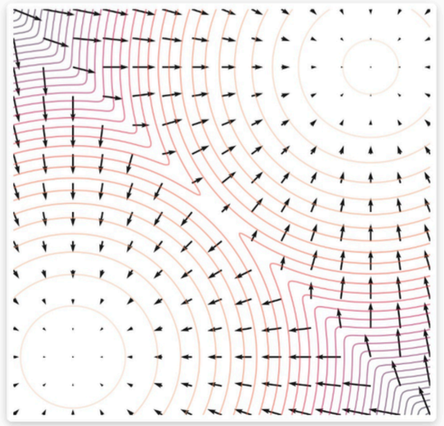
[Song et al. 2021]

- $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\varepsilon$
  - Infinitely small step size → continuous time

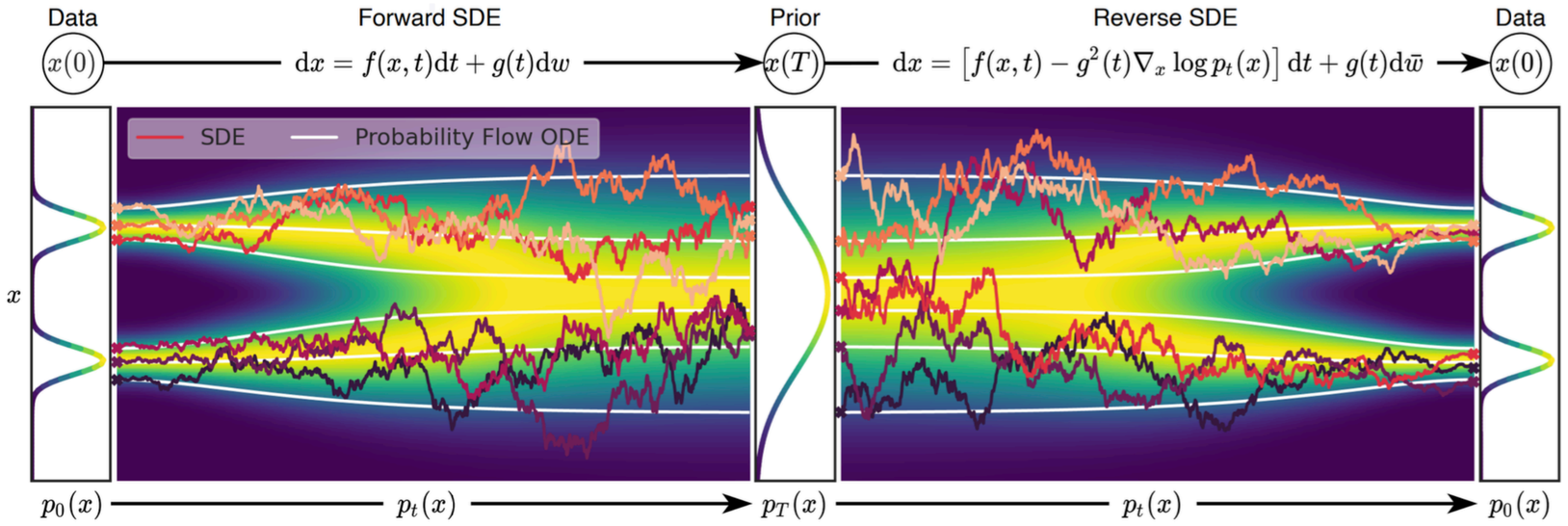
- SDE: $dx = -\frac{1}{2}\beta(t)x\ dt + \sqrt{\beta(t)}\ dw$
  - dw: Wiener process (infinitely small noise)

Reverse the SDE

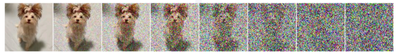
Based on score function $\nabla_x \log p(x)$



Score function (the vector field) and density function (contours) of a mixture of two Gaussians.



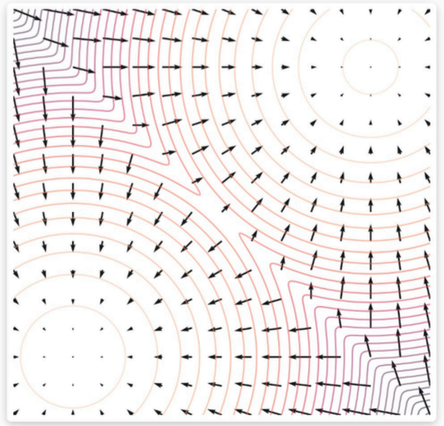
[Song et al. 2021]



How to get $\nabla_{x_t} \log q(x_t)$?

Denoising score matching

- $\min_{\theta} E_{x_0,t,\varepsilon} \| s_\theta(x_t, t) - \nabla_{x_t} \log q(x_t \mid x_0) \|^2$
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$
  - At minimum point (not necessarily 0), $s_\theta(x_t, t) \approx \nabla_{x_t} \log q(x_t)$



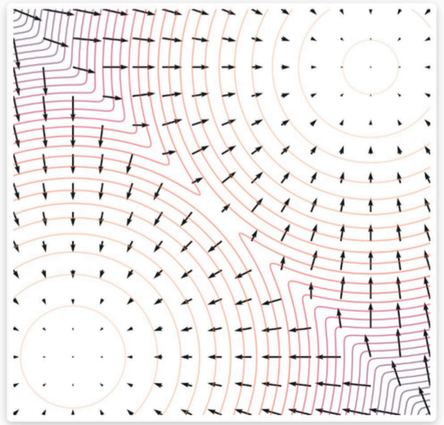
Score function (the vector field) and density function (contours) of a mixture of two Gaussians.

- In diffusion, $q(x_t \mid x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I)$
- $$\nabla_{x_t} \log q(x_t \mid x_0) = \Sigma^{-1}(\mu - x_t) = \frac{-\varepsilon}{\sqrt{1 - \bar{\alpha}_t}}$$

How to get $\nabla_{x_t} \log q(x_t)$?

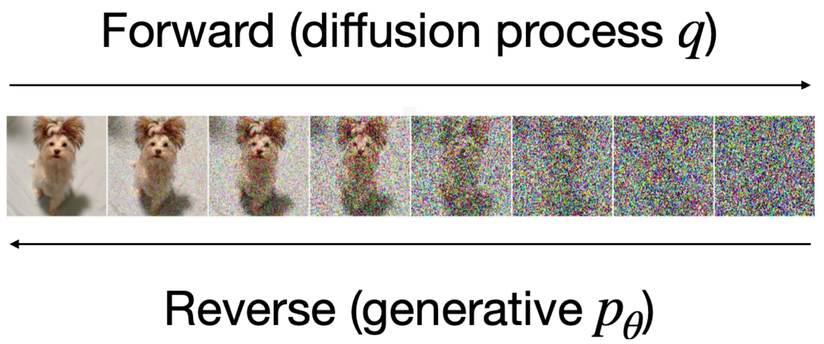
Denoising score matching

- $\min_{\theta} E_{x_0,t,\varepsilon} || s_\theta(x_t, t) - \frac{-\varepsilon}{\sqrt{1-\bar{\alpha}_t}} ||^2$
- Let $s_\theta(x_t, t) = \frac{-\varepsilon_\theta(x_t, t)}{\sqrt{1-\bar{\alpha}_t}}$
- It becomes: $\min_{\theta} E_{x_0,t,\varepsilon} \frac{1}{1-\bar{\alpha}_t} || \varepsilon_\theta(x_t, t) - \varepsilon ||^2$
- Same denoising objective! (DDPM derived from ELBO)



Score function (the vector field) and density function (contours) of a mixture of two Gaussians.

Summary



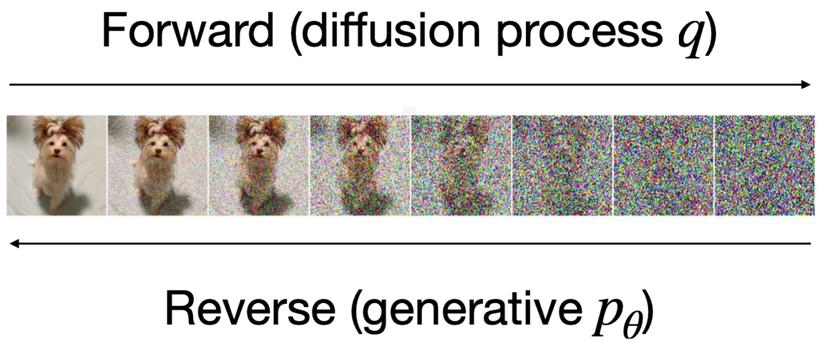
[Song et al. 2021]

- Learn a noise predictor: $E_{x_0,t,\varepsilon}\left[w_t \left|\left| \varepsilon_\theta(x_t, t) - \varepsilon \right|\right|^2\right]$
- Get score functions at all noisy dist.: $\nabla_{x_t} \log q(x_t) \approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \varepsilon_\theta(x_t, t)$

- Under L2 loss, the optimal $\varepsilon_\theta(x_t, t)$ is the expected ε given x_t (mean)
- In theory, w_t does not change the optimal point (but changes ELBO or not)

Generalized view

General formulation



[Song et al. 2021]

- A diffusion process is defined by $x_t = \alpha_t x_0 + \sigma_t \varepsilon$ (a different α than before)
  - **Signal ratio:** α_t from 1 to 0
  - **Noise ratio:** σ_t from 0 to 1

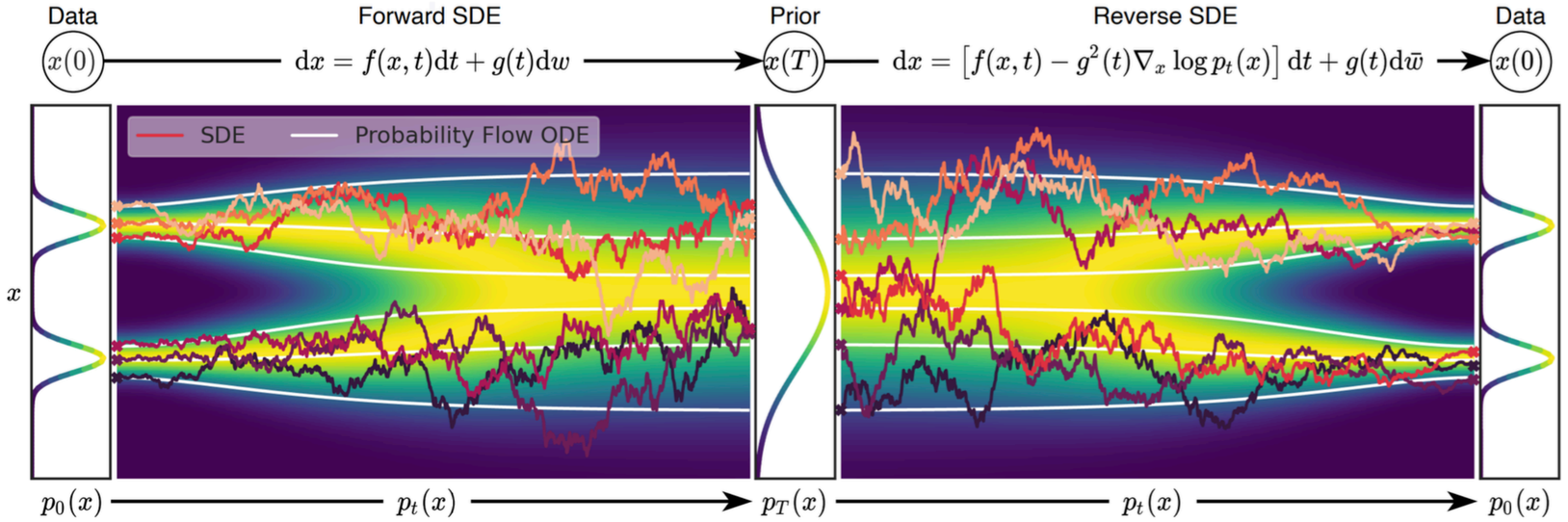
- The change rate of α_t defines the previous "β_t" and we don't force variance 1

Sample prediction

- $x_t = \alpha_t x_0 + \sigma_t \varepsilon$, so $x_0 = (x_t - \sigma_t \varepsilon)/\alpha_t$
- Define $x_\theta(x_t, t)$ as $x_\theta = (x_t - \sigma_t \varepsilon_\theta)/\alpha_t$
- $||x_\theta - x_0||^2 = ||(x_t - \sigma_t \varepsilon_\theta)/\alpha_t - (x_t - \sigma_t \varepsilon)/\alpha_t||^2 = (\frac{\sigma_t}{\alpha_t})^2 ||\varepsilon_\theta - \varepsilon||^2$

- Predict x_0, i.e., $E_{x_0,t,\varepsilon}||x_\theta(x_t, t) - x_0||^2$, is just a time-reweighed ε-prediction
- Furthermore, predict any $A\varepsilon + Bx_0$ from x_t learns the same score function

Probability Flow ODE (PF-ODE)



[Song et al. 2021]

- Reverse ODE: $dx = \left[f(x,t) - \frac{1}{2} g(t)^2 \nabla_x \log p_t(x)\right] dt$

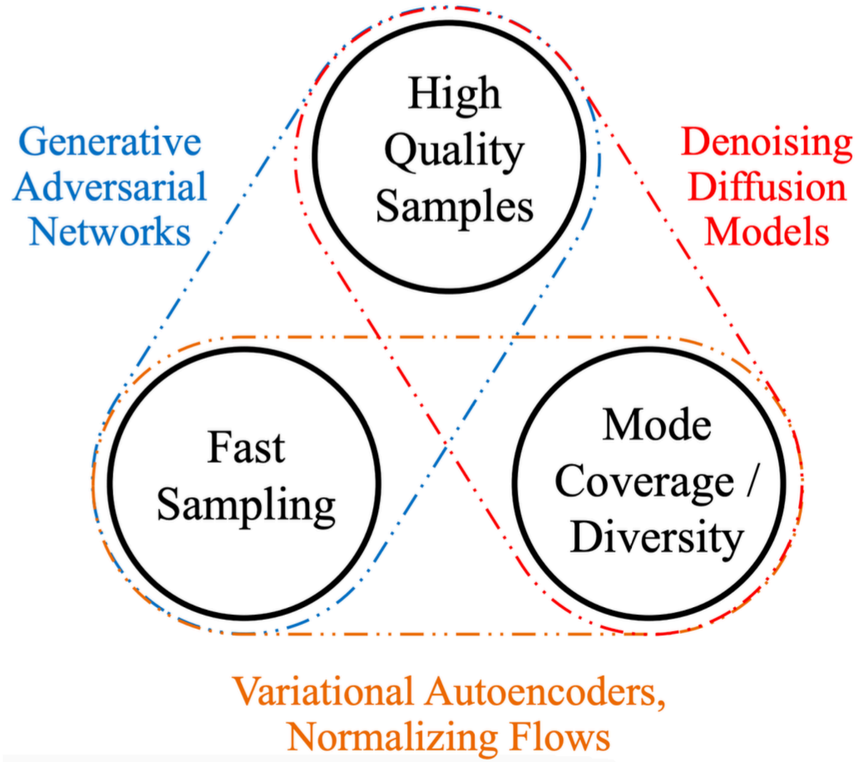
- Gives the same marginal distributions $p_t(x)$ (but a different joint distribution)

Flow matching

- $x_t = \alpha_t x_0 + \sigma_t \varepsilon$
  - **Signal ratio:** $\alpha_t = 1 - t$
  - **Noise ratio:** $\sigma_t = t$
- Prediction: $\varepsilon - x_0$, i.e., $\mathscr{L} = E_{x_0,t,\varepsilon} \left|\left| v_\theta(x_t, t) - (\varepsilon - x_0) \right|\right|^2$

- Is simple, is ELBO [Kingma et al., 2023] (Used in many recent SOTA models)

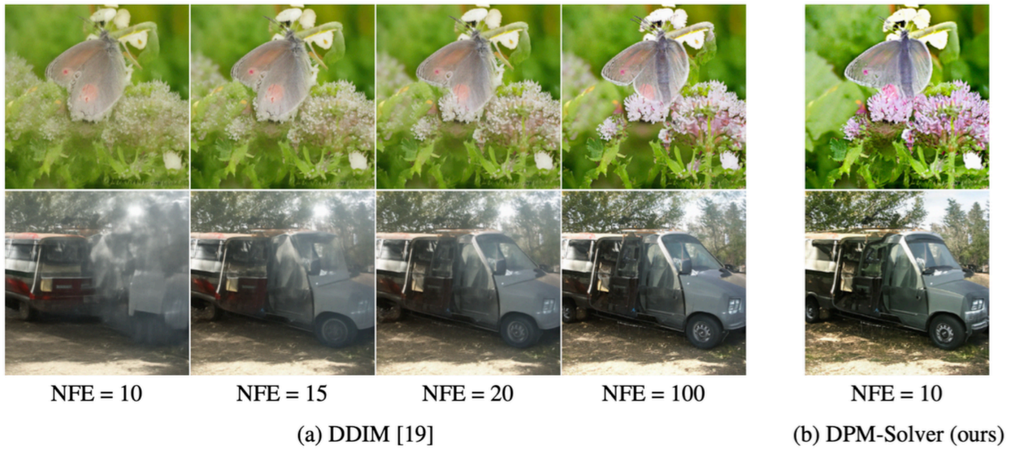
Generative learning trilemma



[Xiao et al. 2022]

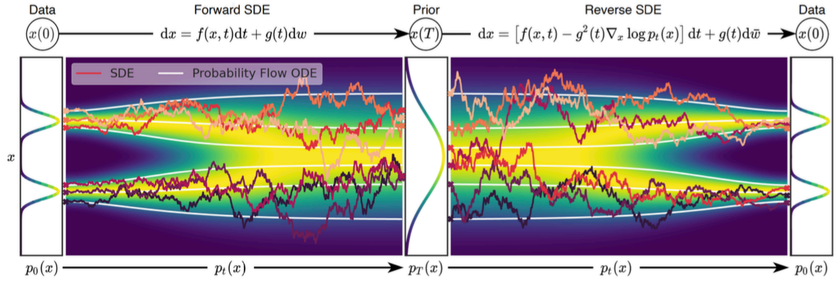
Sampling acceleration

1. Better ODE/SDE solvers



(a) DDIM [19] (b) DPM-Solver (ours)

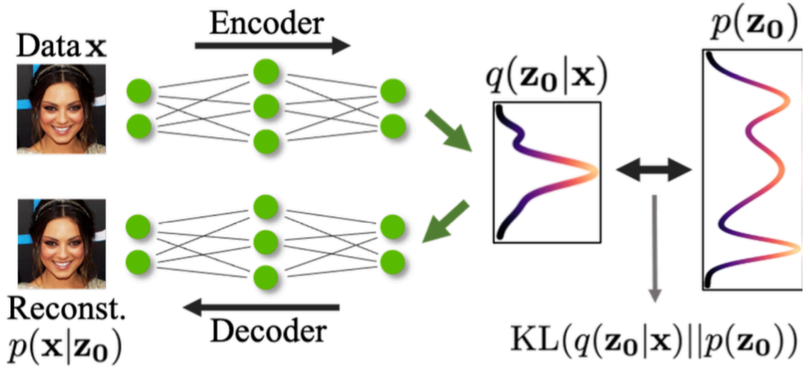
[Lu et al. 2022]



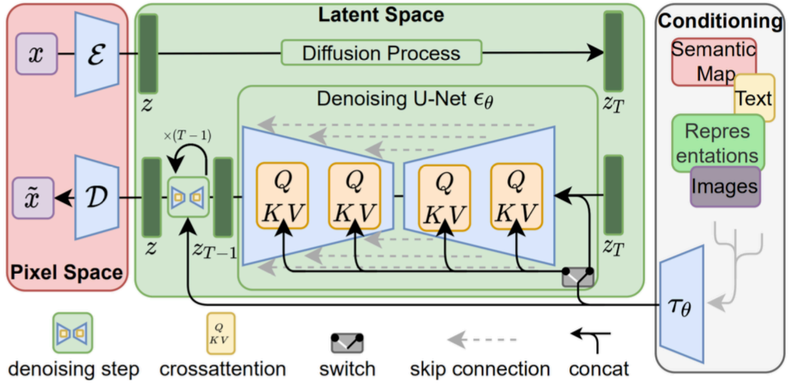
- Higher order methods:
  RK4, Heun …
- Exponential ODE Integrators:
  DPM Solver …
- …

Sampling acceleration

2. Latent space



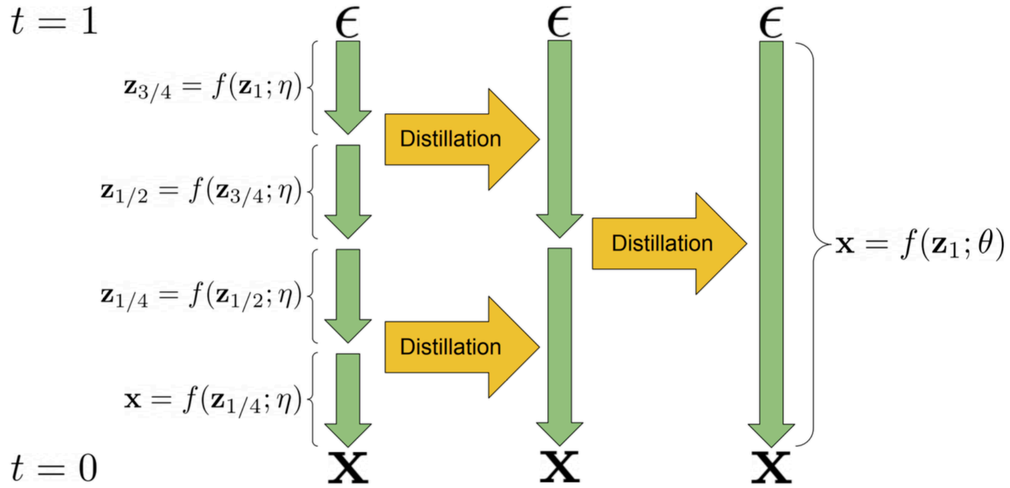
[Vahdat et al. 2021]



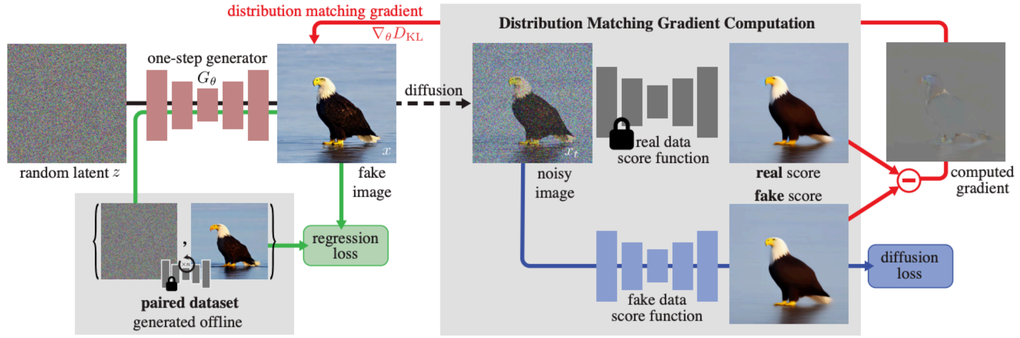
[Rombach et al. 2021]

Sampling acceleration

3. Distillation



[Salimans et al. 2022]



[Yin et al. 2024]

Conditional generation

1. Make the network conditional

- Simply, we learn $\nabla_{x_t} \log q(x_t \mid c)$

- Training: $E_{x_0,t,\varepsilon,c}\left[w_t \left|\left| \varepsilon_\theta(x_t, t, c) - \varepsilon \right|\right|^2\right]$

Conditional generation

2. Classifier-based guidance

- $q(x_t \mid c)q(c) = q(x_t)q(c \mid x_t)$. Take log and gradient to x_t:

$$\nabla_{x_t} \log q(x_t \mid c) = \nabla_{x_t} \log q(x_t) + \nabla_{x_t} \log q(c \mid x_t)$$

- $\nabla_{x_t} \log q(x_t)$: unconditional model
- $\nabla_{x_t} \log q(c \mid x_t)$: a separate classifier (BP gradient to the input)

Conditional generation

3. Classifier-free guidance

- $\nabla_{x_t} \log q(x_t \mid c) = \nabla_{x_t} \log q(x_t) + \nabla_{x_t} \log q(c \mid x_t)$
  - $\nabla_{x_t} \log q(c \mid x_t) = \nabla_{x_t} \log q(x_t \mid c) - \nabla_{x_t} \log q(x_t)$

- $s_\theta^{CFG}(x_t, t, c) = s_\theta(x_t, t, \varnothing) + CFG \cdot (s_\theta(x_t, t, c) - s_\theta(x_t, t, \varnothing))$
- During training, condition c is randomly dropped (e.g., 10%)

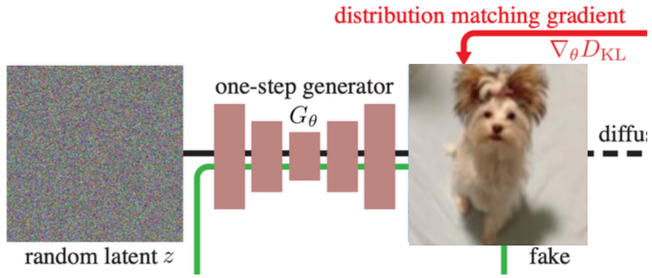
Score distillation: one-step generation

[Poole et al. 2022]

[Wang et al. 2023]

[Luo et al. 2023]

[Yin et al. 2023]



[Yin et al. 2023]

$$D_{KL}\left(p_{\text{fake}} \parallel p_{\text{real}}\right) = \mathop{\mathbb{E}}_{x\sim p_{\text{fake}}}\left(\log\left(\frac{p_{\text{fake}}(x)}{p_{\text{real}}(x)}\right)\right)$$
$$= \mathop{\mathbb{E}}_{\substack{z\sim\mathcal{N}(0;\mathbf{I})\\ x=G_\theta(z)}} -\big(\log\, p_{\text{real}}(x) - \log\, p_{\text{fake}}(x)\big).$$

(1)

$$\nabla_\theta D_{KL} = \mathop{\mathbb{E}}_{\substack{z\sim\mathcal{N}(0;\mathbf{I})\\ x=G_\theta(z)}}\Big[-\big(s_{\text{real}}(x) - s_{\text{fake}}(x)\big)\,\nabla_\theta G_\theta(z)\Big],$$

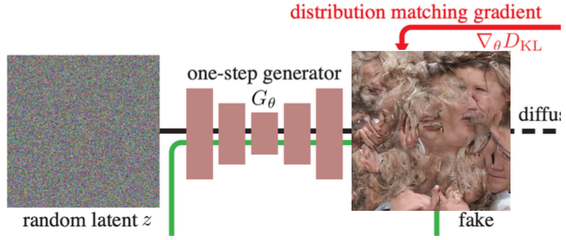
(2)

where $s_{\text{real}}(x) = \nabla_x \log\, p_{\text{real}}(x)$, $s_{\text{fake}}(x) = \nabla_x \log\, p_{\text{fake}}(x)$

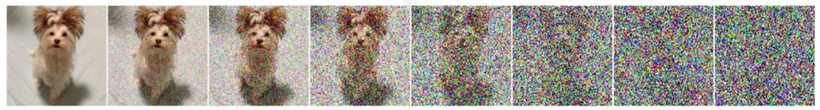
Score distillation

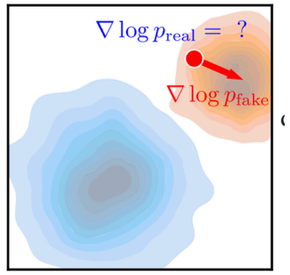
$$\nabla_\theta D_{KL} = \mathop{\mathbb{E}}_{\substack{z\sim\mathcal{N}(0;\mathbf{I}) \\ x=G_\theta(z)}} \Big[-\big(s_{\text{real}}(x) - s_{\text{fake}}(x)\big)\, \nabla_\theta G_\theta(z) \Big]$$

Is this all we need?



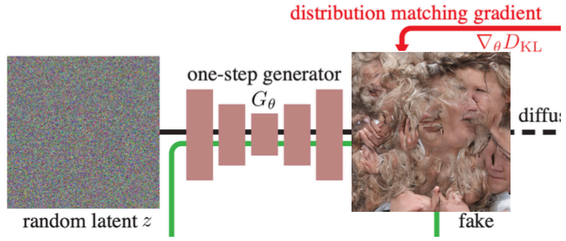
Issue: for a bad generated x, the score function may not be available, as it is OOD of $\nabla_{x_t} \log p(x_t)$ at any t





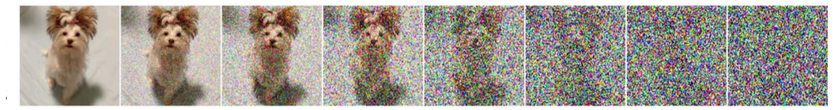
Score distillation

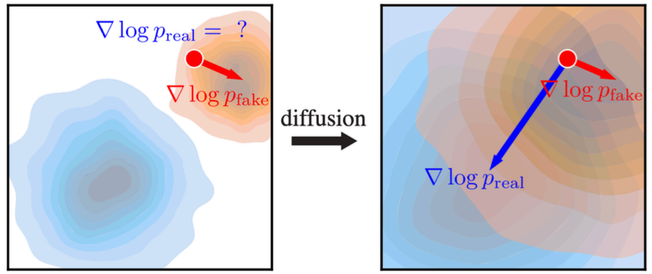
$$\nabla_\theta D_{KL} = \mathop{\mathbb{E}}_{\substack{z\sim\mathcal{N}(0;\mathbf{I}) \\ x=G_\theta(z)}} \Big[-\big(s_{\text{real}}(x) - s_{\text{fake}}(x)\big)\, \nabla_\theta G_\theta(z) \Big]$$



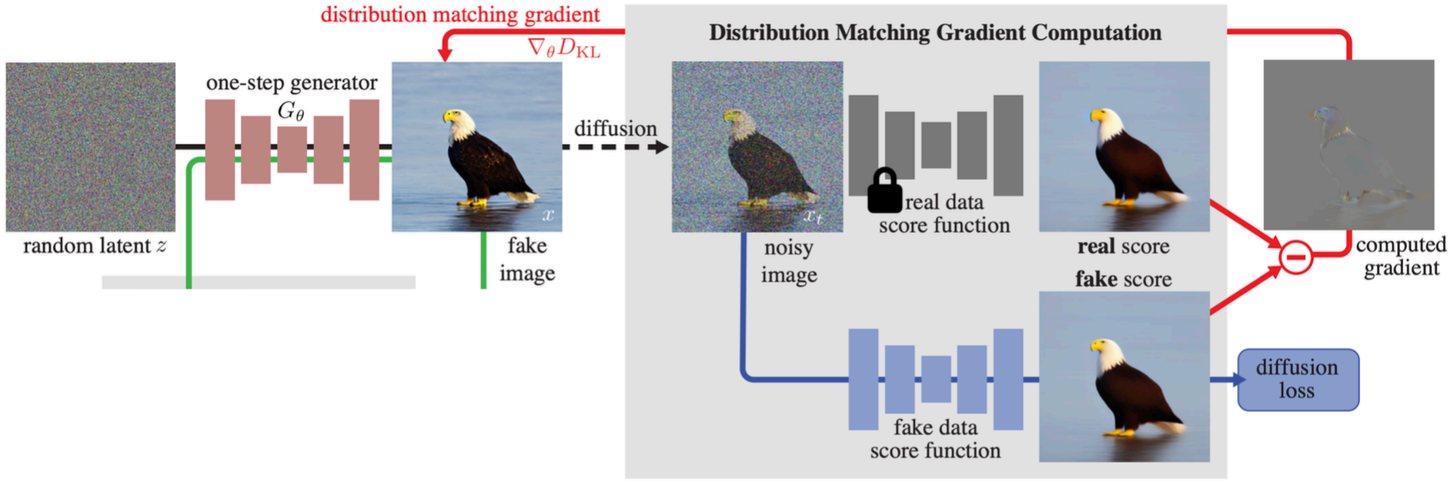
Compute on diffused distributions:

$$\nabla_\theta D_{KL} \simeq \mathop{\mathbb{E}}_{z,t,x,x_t} \left[w_t \alpha_t \big(s_{\text{fake}}(x_t,t) - s_{\text{real}}(x_t,t)\big)\, \nabla_\theta G_\theta(z) \right], \tag{7}$$





Score distillation

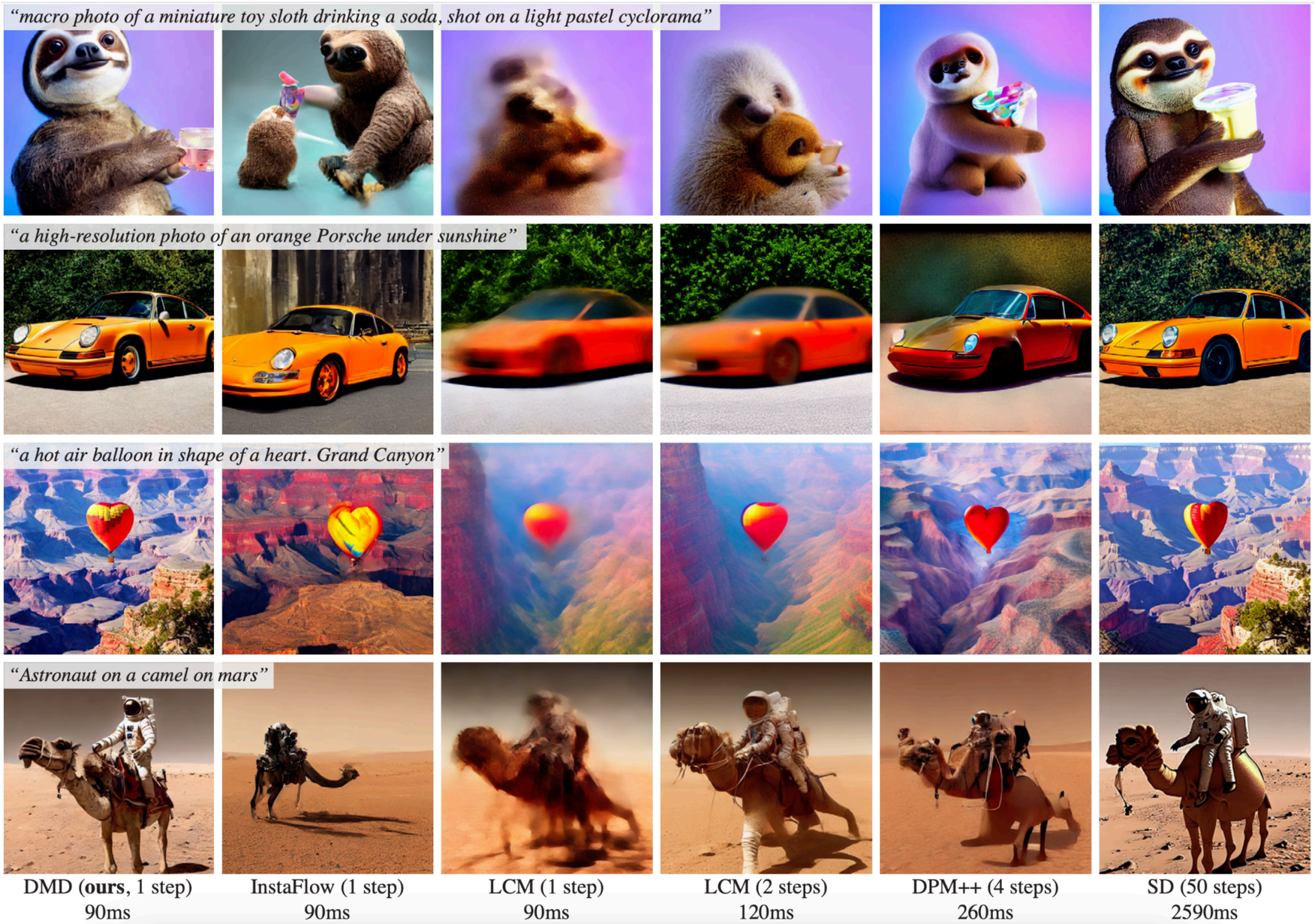


Generator: G_θ

$$\nabla_\theta D_{KL} \simeq \mathop{\mathbb{E}}_{z,t,x,x_t} \left[w_t \alpha_t \left(s_{\text{fake}}(x_t, t) - s_{\text{real}}(x_t, t) \right) \nabla_\theta G_\theta(z) \right], \tag{7}$$

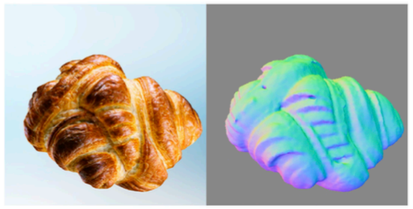
Fake network: s_ϕ^{fake}

A loss to make some representation follow some distribution (with DM scores)

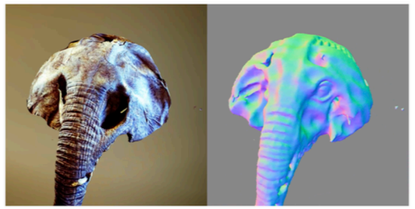

"macro photo of a miniature toy sloth drinking a soda, shot on a light pastel cyclorama"
"a high-resolution photo of an orange Porsche under sunshine"
"a hot air balloon in shape of a heart. Grand Canyon"
"Astronaut on a camel on mars"
DMD (**ours**, 1 step)
90ms
InstaFlow (1 step)
90ms
LCM (1 step)
90ms
LCM (2 steps)
120ms
DPM++ (4 steps)
260ms
SD (50 steps)
2590ms

Score distillation: text-to-3D

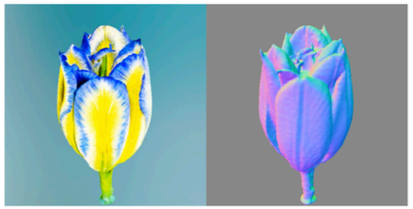
Michelangelo style statue of dog reading news on a cellphone.



A delicious croissant.



An elephant skull.

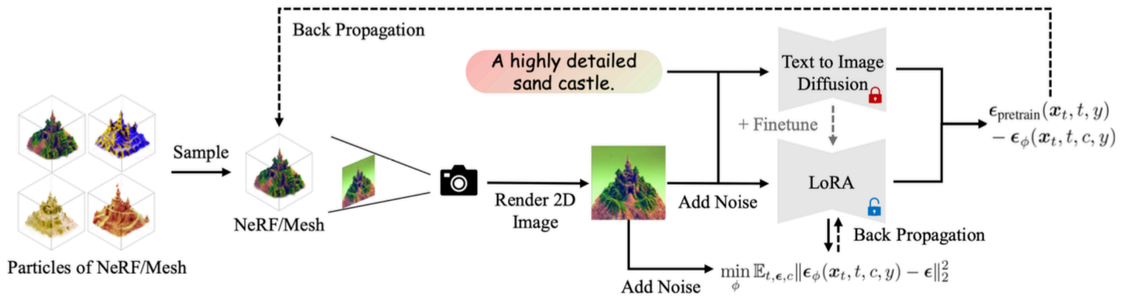


A blue tulip.

Optimize a learnable 3D representation,

such that the rendered views follow the text-to-image distribution (pretrained DM)

[Wang et al. 2023]



Summary

- **Basics:** discrete time, continuous time (SDE / ODE)
- **Applications:** acceleration, conditional generation
- Score distillation
- (Topics not covered: DreamBooth, engineering …)