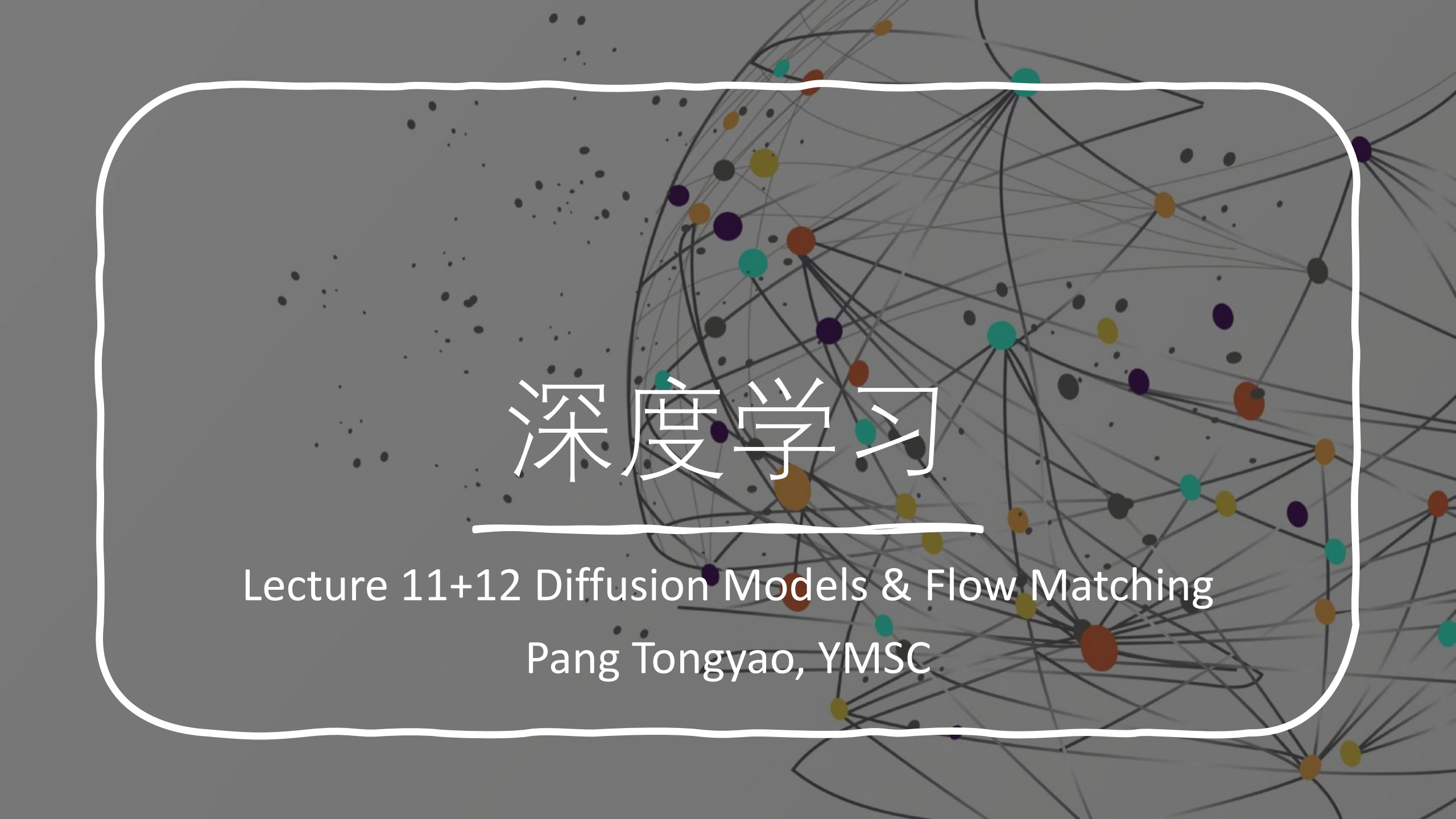


深度学习

Lecture 11+12 Diffusion Models & Flow Matching

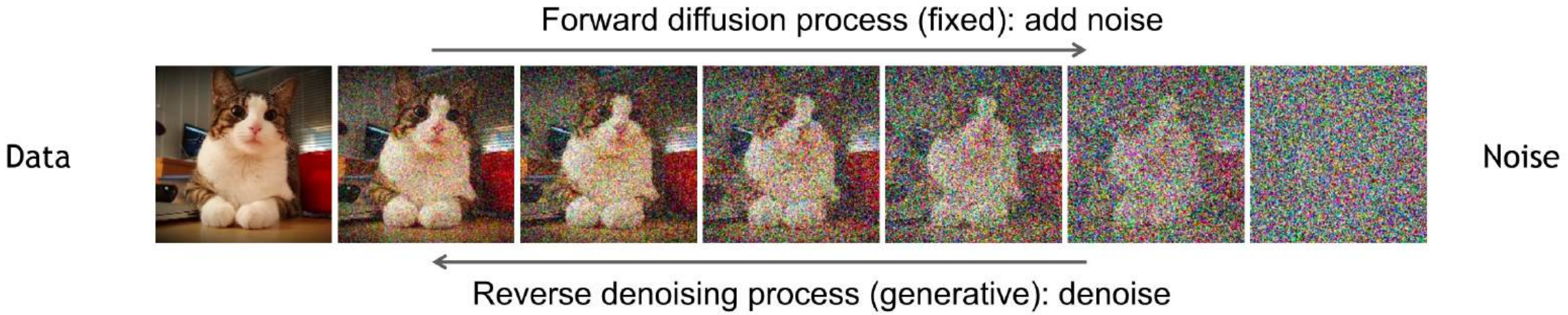
Pang Tongyao, YMSC

Diffusion Models

Reference: Kreis, Gao, Vahdat, “Denoising Diffusion-based Generative Modeling: Foundations and Applications”, CVPR2022 Tutorial.

DDPM

- Denoising Diffusion Probabilistic Models (DDPM): the first working diffusion models

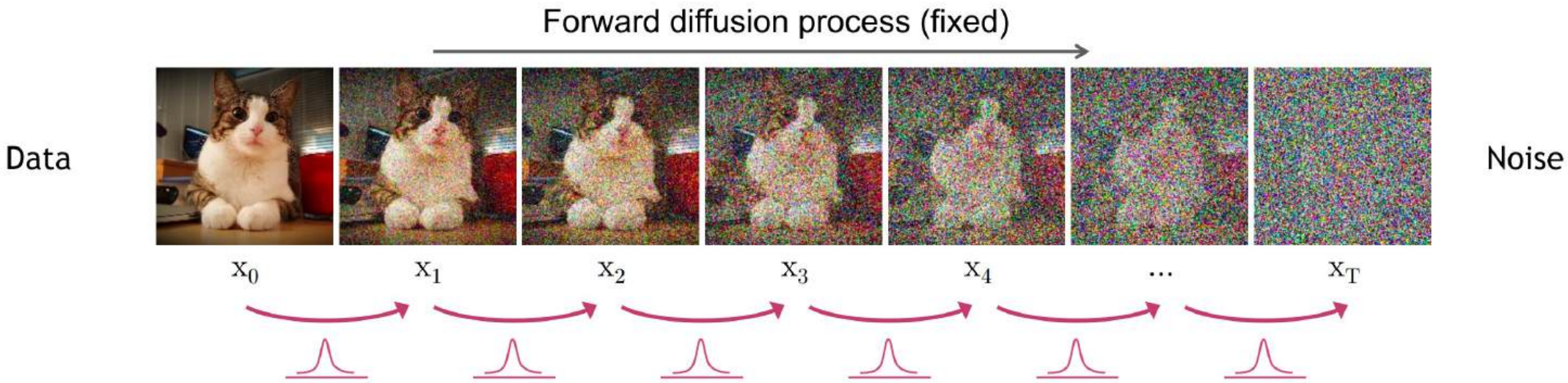


Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020

DDPM

- Forward process(fixed): add noise

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



DDPM

- Forward process(fixed): add noise

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

The goal of the forward process is to transfer the data distribution to a normal distribution:

$$x_T \approx \mathcal{N}(0, \boldsymbol{I})$$

$$x_T = \sqrt{\bar{\alpha}_T}x_0 + (1-\bar{\alpha}_T)\epsilon$$

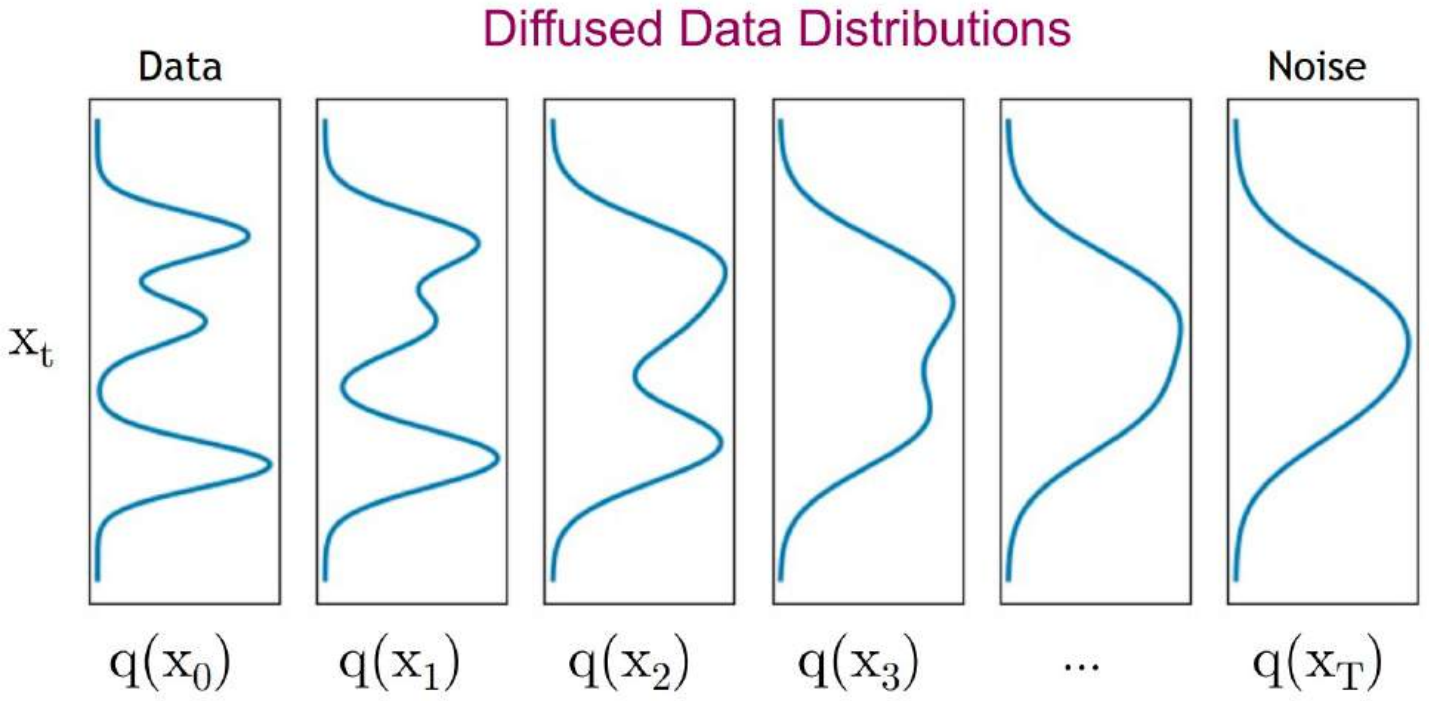
$$\sqrt{\bar{\alpha}_T} \to 0, 1-\bar{\alpha}_T \to 1$$

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon, \epsilon \sim \mathcal{N}(0, \boldsymbol{I})$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon, \epsilon \sim \mathcal{N}(0, \boldsymbol{I})$$

$$\alpha_t = 1-\beta_t, \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

DDPM



$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} d\mathbf{x}_0$$

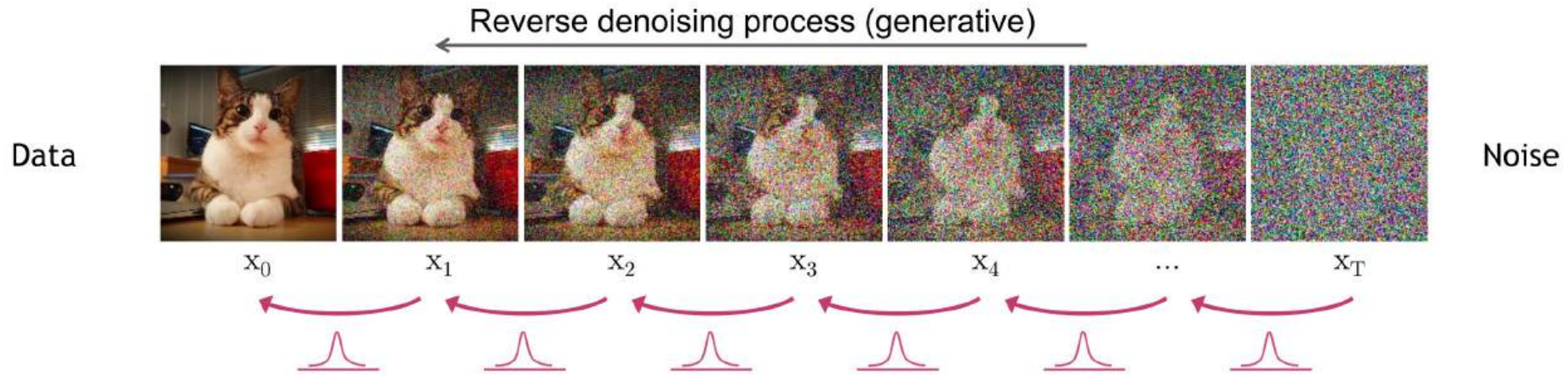
← Gaussian convolution

DDPM

- Reverse process(learned): denoise

Denoising NN

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

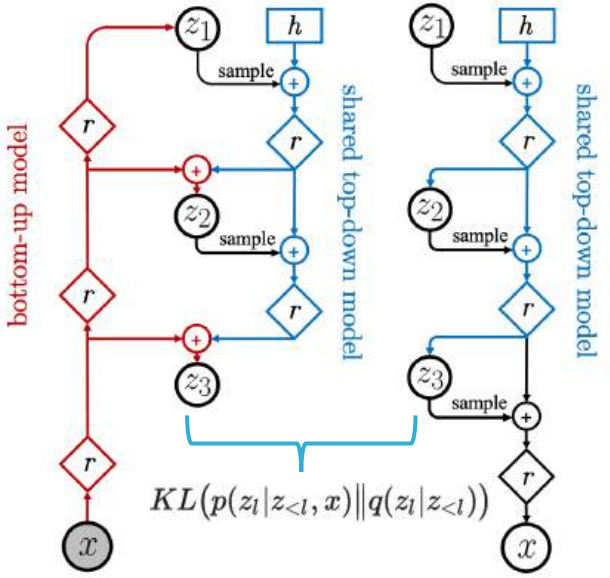


DDPM

- Variational Inference (ELBO)

$$-\log p(x) = -\int q_x(z) log p(x|z) dz + KL(q_x(z)||p(z)) - KL(q_x(z)||p(z|x))$$

Similar as Hierarchical VAE



(a) Bidirectional Encoder (b) Generative Model

$$Latent\ code\ z = \{x_1, x_2, \dots, x_T\}$$

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

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

the forward process is decomposed into a similar form as the reverse one for easier KL divergence calculation

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020

DDPM

- Since $q(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}|x_t)$ are both Gaussian, their KL divergence is

$$L_{t-1} = D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] + C$$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

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

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

DDPM

- Training loss

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \Big[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon, t)||^2 \Big]$$

- DDPM sets $\lambda_t = 1$:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}} \Big[\big\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \big\|^2 \Big]$$

x_t

The training process is parallel for different timestep

Algorithm 1 Training

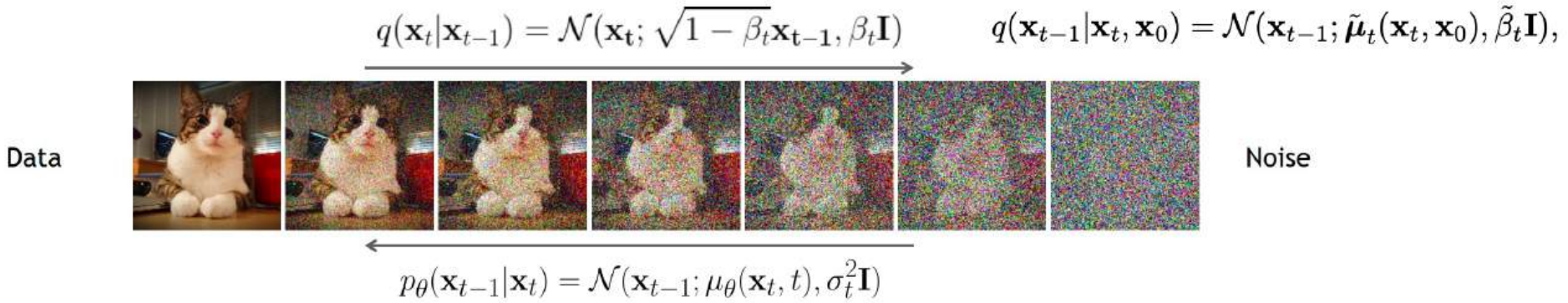
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \big\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \big\|^2$
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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \Big) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

DDPM

- Noise schedule



- Common seen schedule of β_t: linear, cosine …
- σ_t^2 can be fixed to β_t or $\tilde{\beta}_t$. [Nichol et al.] Learns it in the form of

$$\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1-v) \log \tilde{\beta}_t)$$

Nichol et al. “Improved Denoising Diffusion Probabilistic Models”, ICML, 2021

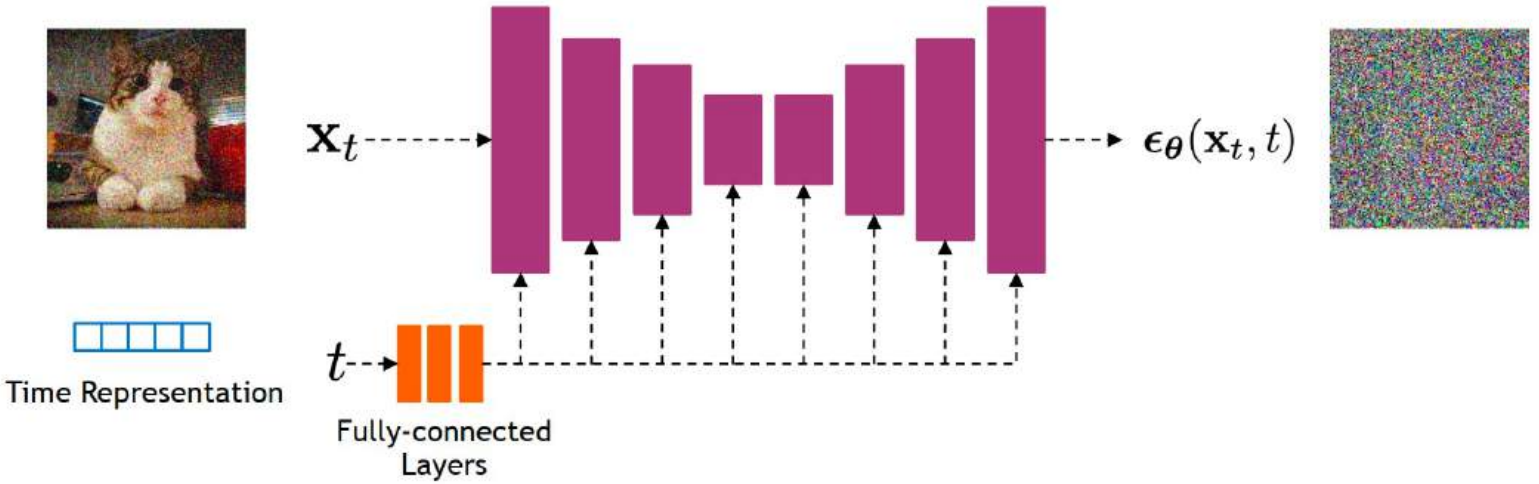
DDPM vs. VAE

For DDPM:

- The encoder is fixed, which leads to more stable training (no "posterior collapse" issue common in VAEs).
- The latent variables have the same dimension as the data, unlike VAEs, which typically compress data into lower-dimensional latent codes.
- The denoising model is shared across different timestep.
- DDPM allows parallel sampling of timesteps during training, while inference (sampling) remains sequential and time-consuming (1000+ steps).
- The model is trained with some reweighting of the variational bound.

Architecture

- Unet architecture of $\epsilon_\theta(x_t, t)$



- t encoding: sinusoidal position embeddings or random Fourier features

Dhariwal et al. , “Diffusion Models Beat GANs on Image Synthesis”

Conditional Diffusion

- Conditional denoising NN $\epsilon(z_t, t, y)$ via attention mechanisms [1]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V, \text{ with}$$

$$Q = W_Q^{(i)} \cdot \varphi_i(z_t),\ K = W_K^{(i)} \cdot \tau_\theta(y),\ V = W_V^{(i)} \cdot \tau_\theta(y).$$

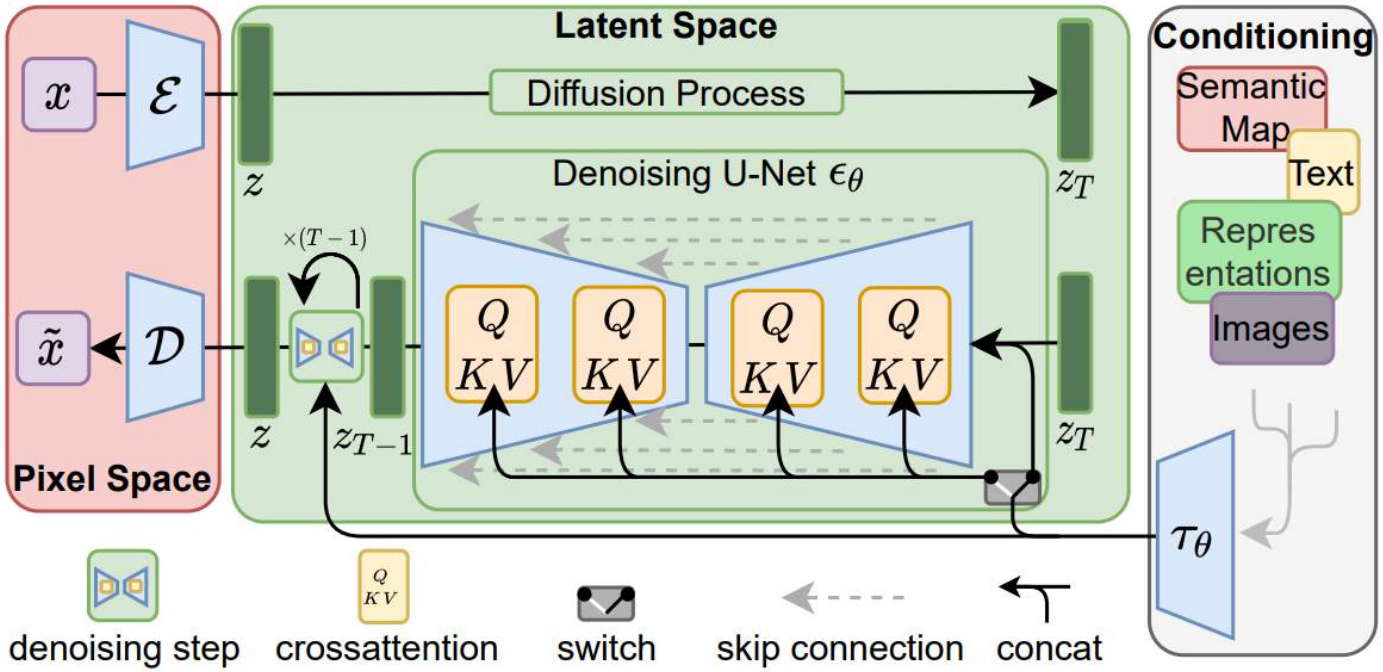
- Conditioning via Bayes‘ Rule (e.g. y is a low resolution image) [2]:

$$\hat{\epsilon}(x_t) := \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$$

1. Rombach et al. “High-Resolution Image Synthesis with Latent Diffusion Models”.
2. Dhariwal et al. “Guided Diffusion from Self-Supervised Diffusion Features”.

Latent Diffusion Models

- Train the diffusion model in the latent space of a pre-trained encoder.



Rombach et al. “High-Resolution Image Synthesis with Latent Diffusion Models”.

Classifier-free Guidance (CFG)

- Classifier Guidance: reliant on gradients from an image classifier $p(y|x_t)$

$$\hat{\epsilon} = \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t} \cdot \nabla_{x_t} \log p(y|x_t)$$

- Classifier-free Guidance : During training, you randomly delete the text prompt (e.g. 10% of the time)

$$\tilde{\epsilon} = \epsilon_\theta(x_t, \varnothing) + w \cdot (\epsilon_\theta(x_t, c) - \epsilon_\theta(x_t, \varnothing))$$

- CFG guides itself by comparing its “Conditioned” and “Unconditioned” predictions.

DDIM Sampling

- Denoising Diffusion Implicit Models (DDIMs) generalize DDPMs via a class of non-Markovian diffusion processes that lead to the same training objective.
- It can take a pretrained diffusion model but produce high quality samples much faster.
- It can perform semantically meaningful image interpolation directly in the latent space.

Song et al., “Denoising Diffusion Implicit Models”, ICLR 2021

DDIM Sampling

- Recall DDPM training objective

$$L_{t-1} = D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] + C$$

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

- The training loss holds, as long as

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\epsilon,$$

Forward process: $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\sigma}_t^2\mathbf{I}), \quad \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = a\mathbf{x}_t + b\epsilon = a\mathbf{x}_t + b\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}$

Reverse process: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \tilde{\sigma}_t^2\mathbf{I}), \quad \mu_\theta(\mathbf{x}_t, t) = a\mathbf{x}_t + b\epsilon_\theta(\mathbf{x}_t, t) = a\mathbf{x}_t + b\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\hat{\mathbf{x}}_0}{\sqrt{1-\bar{\alpha}_t}}$

(a, b) are free

DDIM Sampling

To ensure $\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\epsilon,$

consider a set of forward processes

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

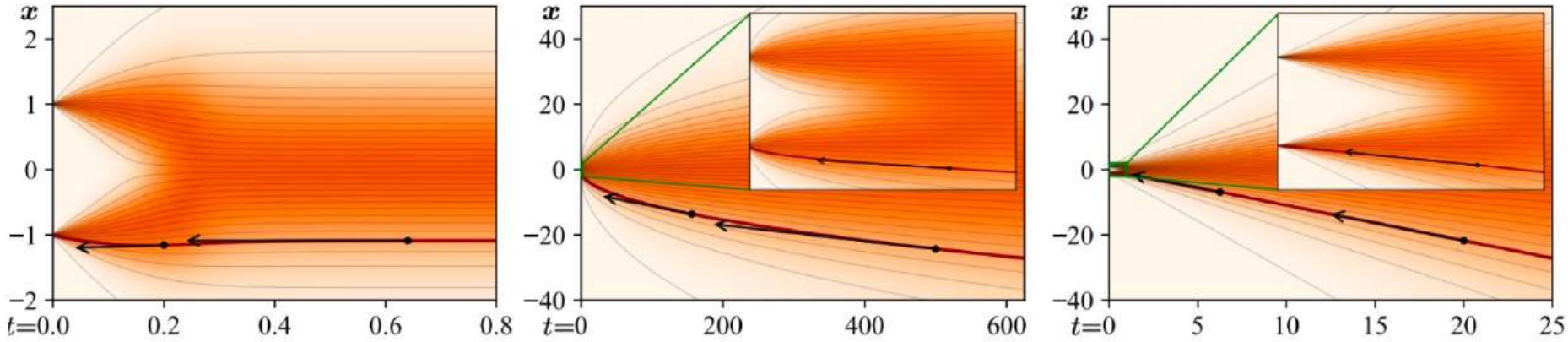
The corresponding reverse processes is

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

DDIM sampler sets $\tilde{\sigma}_t = 0$, so it is deterministic.

$$\hat{\boldsymbol{x}}_0 := (\boldsymbol{x}_t - \sqrt{1-\alpha_t}\cdot\epsilon_\theta^{(t)}(\boldsymbol{x}_t))/\sqrt{\alpha_t}.$$

DDIM Sampling



(a) Variance preserving ODE [42] (b) Variance exploding ODE [42] (c) DDIM [40] / Our ODE

(Karras et al.) argues that the ODE of DDIM is favored, as the tangent of the solution trajectory always points towards the denoiser output.

Leads to largely linear solution trajectories with low curvature.

Low curvature means less truncation errors accumulated over the trajectories.

Karras et al. “Elucidating the Design Space of Diffusion-Based Generative Models”.

DDIM Sampling

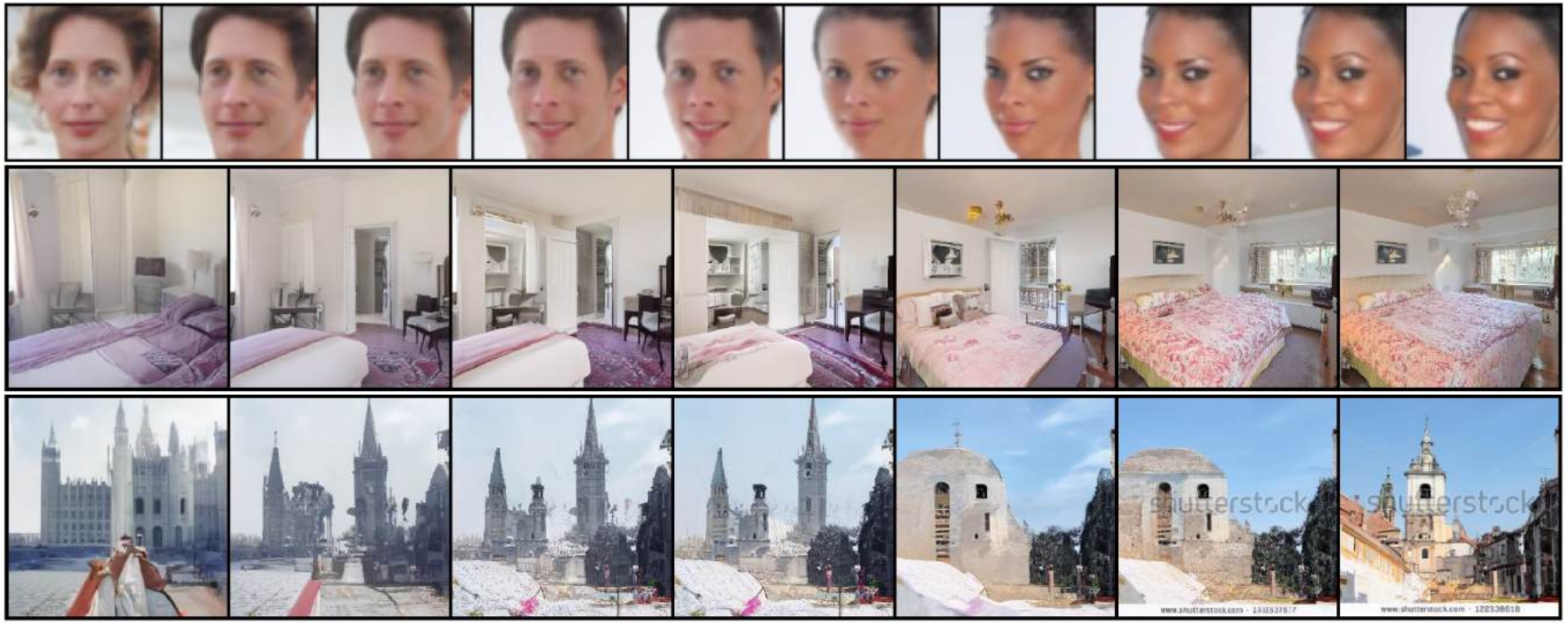
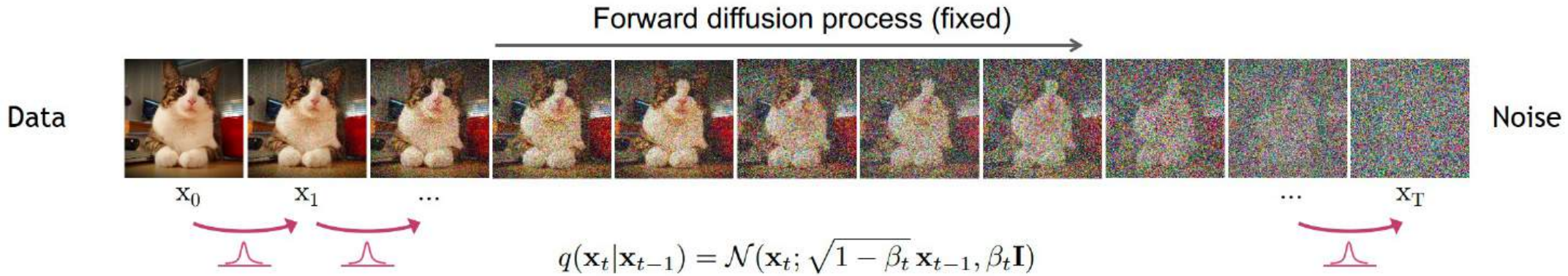


Figure 6: Interpolation of samples from DDIM with $\dim(\tau) = 50$.

Score-based Diffusion Models

Diffusion SDEs



Consider the limit of infinitely small time step

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} + \sqrt{\beta_t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) \\
&= \sqrt{1-\beta(t)\Delta t}\,\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) && (\beta_t := \beta(t)\Delta t) \\
&\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) && \text{(Taylor expansion)}
\end{aligned}$$

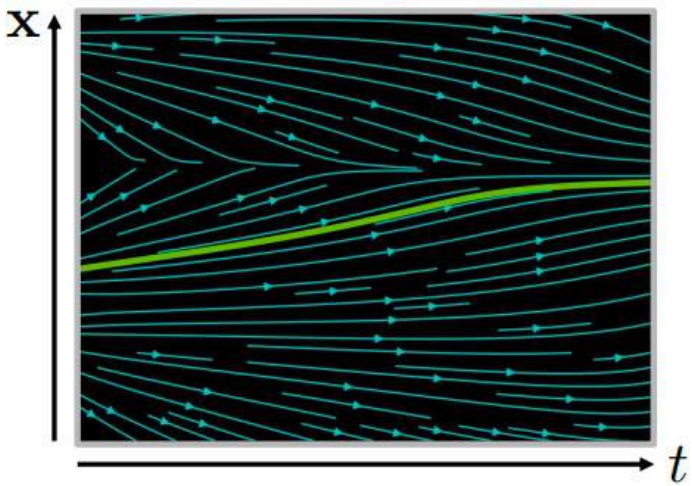
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

Song et al. “Score-Based Generative Modeling through Stochastic Differential Equations”. ICLR, 2021

Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Analytical Solution:

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$$

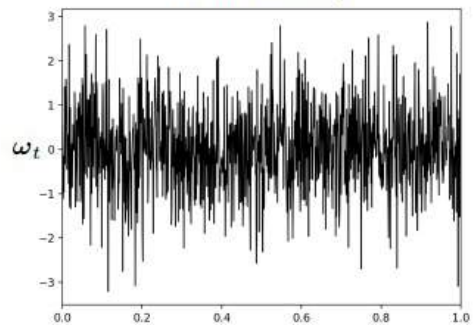
Iterative Numerical Solution:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

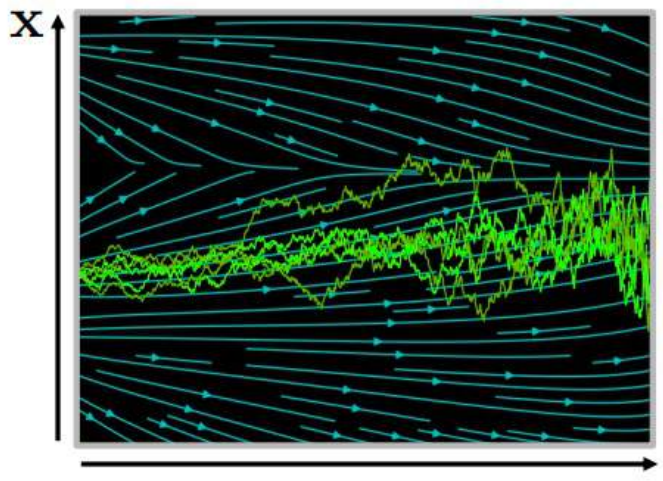
Stochastic Differential Equation (SDE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift coefficient}} + \underbrace{\sigma(\mathbf{x}, t)\boldsymbol{\omega}_t}_{\text{diffusion coefficient}}$$

Wiener Process (Gaussian White Noise)

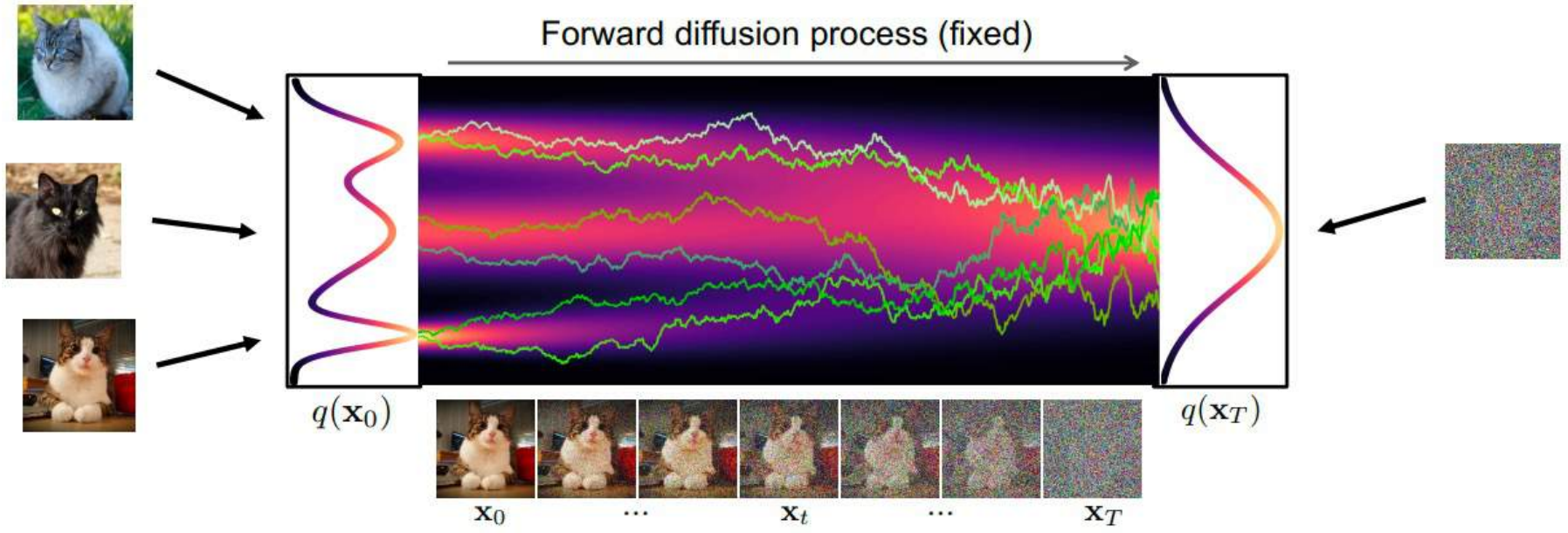


$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \sigma(\mathbf{x}, t)\mathrm{d}\boldsymbol{\omega}_t \right)$$



$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

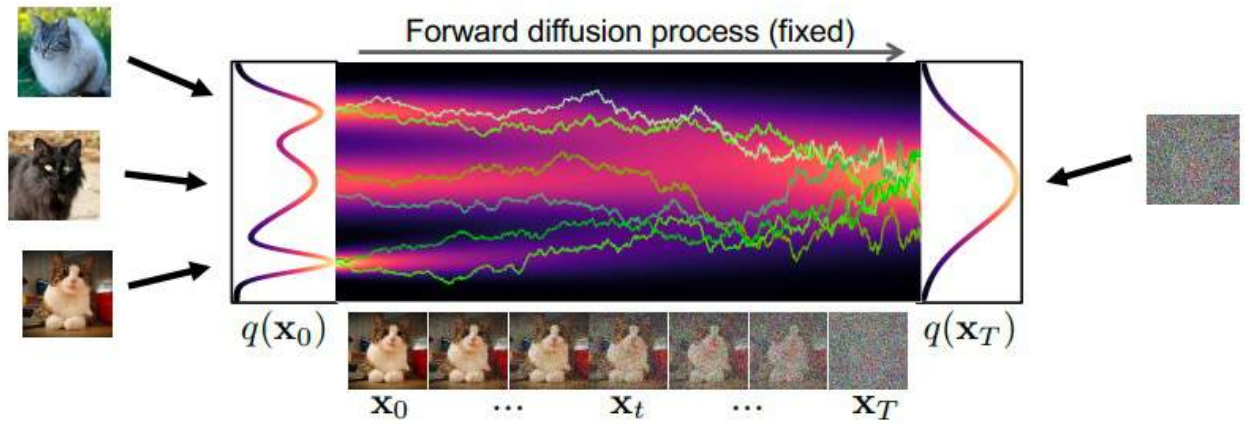
Diffusion SDEs



Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\text{drift term (pulls towards mode)}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\text{diffusion term (injects noise)}}$$

Diffusion SDEs



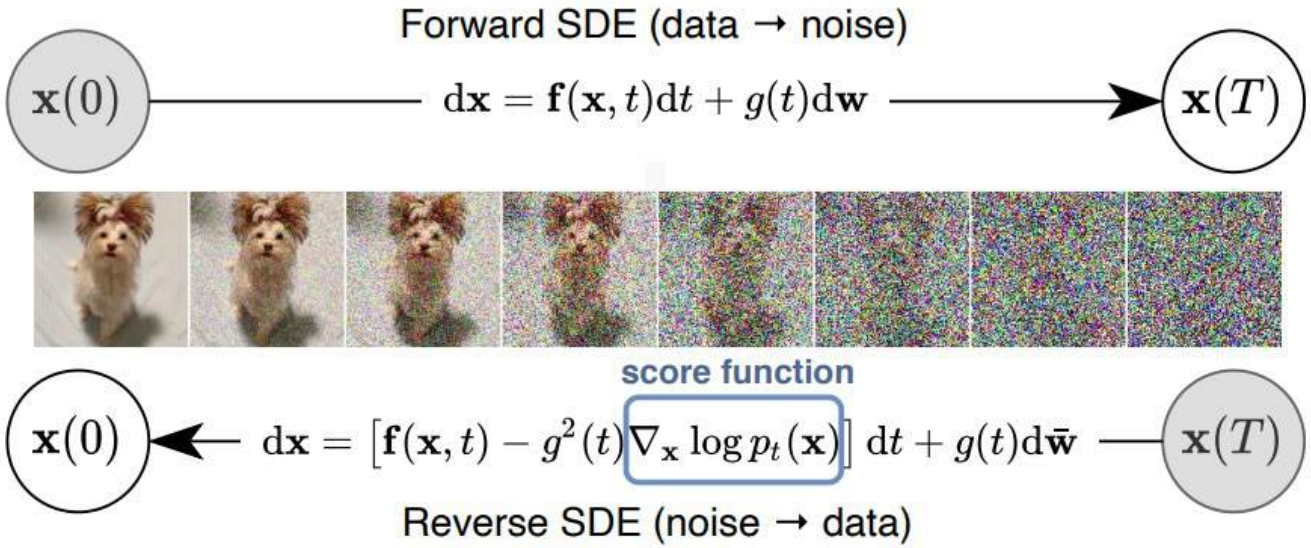
Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

Reverse Generative Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \overbrace{\left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}_{\text{“Score Function”}}\right]}^{\text{drift term}}\mathrm{d}t + \overbrace{\sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t}^{\text{diffusion term}}$$

Diffusion SDEs



Training goal: approximate score function $\nabla_x \log p_t(x)$ with NN $s_\theta(x_t, t)$.

Intractable

Score Matching

- Explicit score matching

$$\mathcal{L}_{\text{ESM}} = \mathbb{E}_{X_t}\left[\frac{1}{2}\|\mathbf{s}_\theta(X_t,t) - \nabla \log q(X_t,t)\|^2\right]$$

- Expanded quadratic form

$$\mathcal{L}_{\text{ESM}} = \mathbb{E}_{X_t}\left[\frac{1}{2}\|\mathbf{s}_\theta(X_t,t)\|^2 - \mathbf{s}_\theta(X_t,t)^T \nabla \log q(X_t,t) + \frac{1}{2}\|\nabla \log q(X_t,t)\|^2\right]$$

- Rearranged form

$$\mathcal{L}_{\text{ESM}} - \frac{1}{2}\mathcal{I}(q(X_t,t)) = \mathbb{E}_{X_t}\left[\frac{1}{2}\|\mathbf{s}_\theta(X_t,t)\|^2 - \mathbf{s}_\theta(X_t,t)^T \nabla \log q(X_t,t)\right]$$

Fisher information is a constant irrelevant to θ: $\mathcal{I}(q(X_t,t)) = \|\nabla \log q(X_t,t)\|^2$

Huang et al. “A Variational Perspective on Diffusion-Based Generative Models and Score Matching”, Neurips, 2021

Score Matching

- Implicit score matching

$$
\begin{aligned}
E_{X_t}\left[\mathbf{s}_\theta(X_t,t)^T\nabla\log q(X_t,t)\right] &= \int \mathbf{s}_\theta(x_t,t)^T\nabla\log q(x_t,t)q(x_t,t)dx_t && \text{(Integral form)}\\
&= \int \nabla\cdot(q\mathbf{s}_\theta)dx_t - \int q\nabla\cdot\mathbf{s}_\theta dx_t && \text{(Integration by parts)}\\
&= \underbrace{\int \nabla\cdot(q\mathbf{s}_\theta)dx_t}_{=0} - E_{X_t}\left[\nabla\cdot\mathbf{s}_\theta(X_t,t)\right] && \text{(Vanishing boundary term)}\\
&\equiv -E_{X_t}\left[\nabla\cdot\mathbf{s}_\theta(X_t,t)\right] && \text{(Final equivalence)}
\end{aligned}
$$

- Sliced Score matching (approximate the trace of the Jacobian)

$$\mathbb{E}_x[\nabla\cdot f_\theta(x)] = \mathbb{E}_v\mathbb{E}_x[v^T\nabla_x f_\theta(x)v], v\sim N(0,I)$$

Score Matching

- Denoising score matching (used in diffusion models):

$$\begin{aligned}
\int q(x,t) S_\theta(x_t,t)^T \nabla \log q(x_t,t)\, dx_t &= \int S_\theta(x_t,t)^T \nabla q(x_t,t)\, dx_t \\
&= \int S_\theta(x_t,t)^T \nabla \int q(x_t|x_0) q(x_0,0)\, dx_0\, dx_t \\
&= \int \int q(x_0,0) S_\theta^T \nabla q(x_t|x_0)\, dx_t\, dx_0 \\
&= \int \int q(x_0,0) q(x_t|x_0) S_\theta^T \nabla \log q(x_t|x_0)\, dx_t\, dx_0 \\
&= \mathbb{E}_{X_0,X_t}\left[S_\theta^T \nabla \log q(X_t|X_0)\right],
\end{aligned}$$

where $q(x_t|x_0)$ denotes the conditional density of X_t given X_0. Combining this with $\mathbb{E}_{X_t}[\|S_\theta\|^2]$, we have:

$$\begin{aligned}
\mathbb{E}_{X_0,X_t}\left[\frac{1}{2}\|S_\theta\|^2 - S_\theta^T \nabla \log q(X_t|X_0)\right] &= \mathbb{E}_{X_0,X_t}\left[\frac{1}{2}\|S_\theta - \nabla \log q(X_t|X_0)\|^2\right] \\
&\quad - \frac{1}{2}\mathbb{E}_{X_0}[\mathcal{I}(q(x_t|X_0))] \quad \text{(constant)}
\end{aligned}$$

Denoising Score Matching

- Denoising score matching gives

$$\mathbb{E}_{x_t} ||s_\theta(x_t, t) - \nabla \log q(x_t)||^2$$
$$= \mathbb{E}_{x_0, x_t} ||s_\theta(x_t, t) - \nabla \log q(x_t|x_0)||^2 + c$$

- $\nabla \log p(x_t|x_0)$ is analytically tractable.

Learning to estimate the score function $\nabla \log q(x_t)$ of the marginal distribution is equivalent to estimating the score function $\nabla \log q(x_t|x_0)$ of the noise-corrupted conditional distribution.

"Variance Preserving" SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

$$q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \gamma_t\mathbf{x}_0, \sigma_t^2\mathbf{I})$$

$$\gamma_t = e^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - e^{-\int_0^t \beta(s)ds}$$

Denoising score matching

- Reparametrize:

$$\mathbf{x}_t = \gamma_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0) = -\nabla_{\mathbf{x}_t} \frac{(\mathbf{x}_t - \gamma_t \mathbf{x}_0)^2}{2\sigma_t^2} = -\frac{\mathbf{x}_t - \gamma_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\gamma_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} - \gamma_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}}{\sigma_t}$$

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) := -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}$$

- Training objective:

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

- More forward SDEs

$$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}} \mathrm{d}\mathbf{w}.$$

Variance Exploding (VE) SDE

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)\mathrm{d}s})}\mathrm{d}\mathbf{w}.$$

sub-VP SDE

Recaps: Score-based Generative Models

- The score based generative models learn the score function $\nabla_{\mathrm{x}} \log p(x)$ instead of the density function $p(x)$.
- Goal: Find $s_\theta(x) \approx p(x)$, Pros: the score-based model is independent of the normalizing constant.
- Training objective: score matching (minimizing the Fisher divergence)

$$\min_\theta ||s_\theta(x) - \nabla_x \log p(x)||_2^2$$

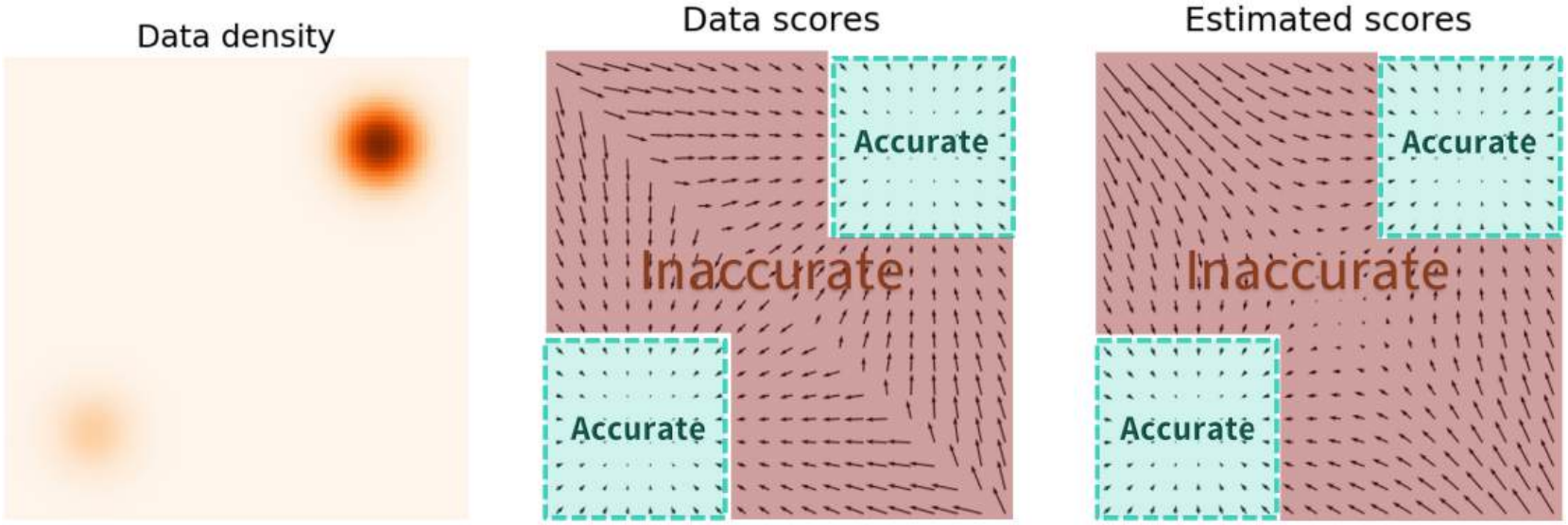
- This enables sampling through discrete-time Langevin dynamics.

$$x_{t+1} = x_t + \delta s_\theta(x_t) + \sqrt{2\delta} z_t, z_t \sim N(0, I)$$

Pitfalls

- Since the differences are weighted by $p(x)$, they are largely ignored in low density regions.

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2] = \int p(\mathbf{x}) \|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2 \mathrm{d}\mathbf{x}.$$



Estimated scores are only accurate in high density regions.

From Song Yang's Blog: https://yang-song.net/blog/2021/score/

Noise Perturbations

- To avoid low density issues, one solution is to perturb data points with noise. The question is how to schedule the noise scale.
- Tradeoff: large noise scale eliminate low-density regions , enables jumping between model, but overly distorts original data.
- Diffusion model: perturb images with multiple scale of Gaussian noise

Annealed Langevin Dynamics

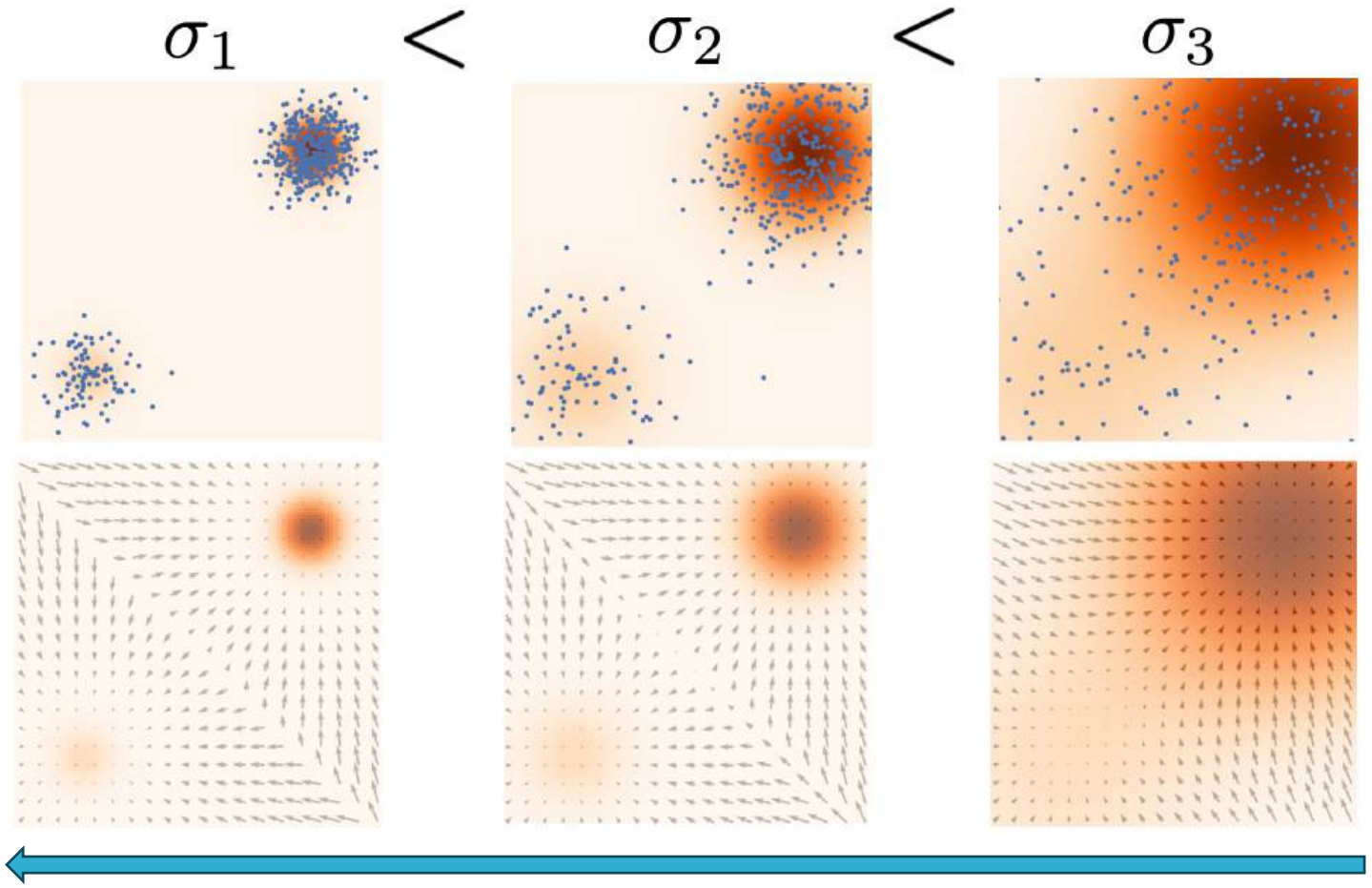
- Pertubed distribution

$$p_{\sigma_i}(\mathbf{x}) = \int p(\mathbf{y}) \mathcal{N}(\mathbf{x}; \mathbf{y}, \sigma_i^2 I) \mathrm{d}\mathbf{y}.$$

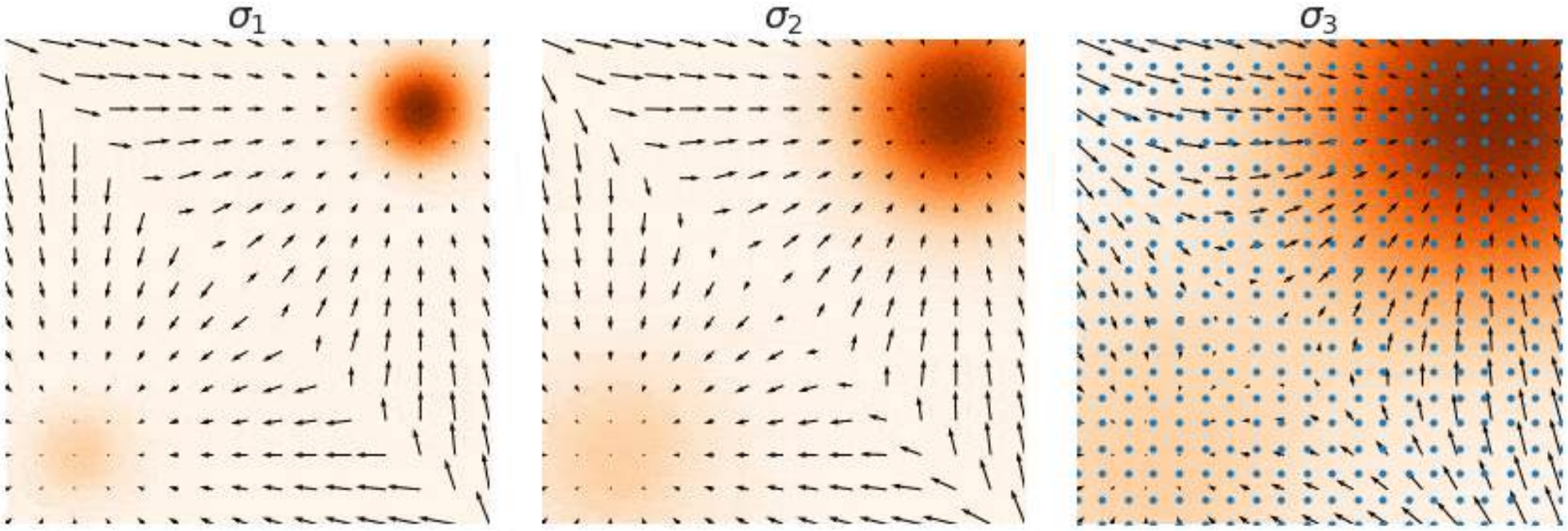
- Score-matching

$$\mathbf{s}_\theta(\mathbf{x}, i) \approx \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$$

- Sampling by progressively reducing noise scales



Yang Song, and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. NeurIPS 2019 (Oral)



- Annealed Langevin dynamics combine a sequence of Langevin chains with gradually decreasing noise scales.
- Start with large noise for more mode coverage.
- Faster mixing rate than fixed-noise Langevin.

Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ ▷ α_i is the step size.
  for $t \leftarrow 1$ to T **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\,\mathbf{z}_t$
  end for
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

Diffusion ODEs

- The following SDE and ODE have the same marginal distributions at all time step.

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}},$$

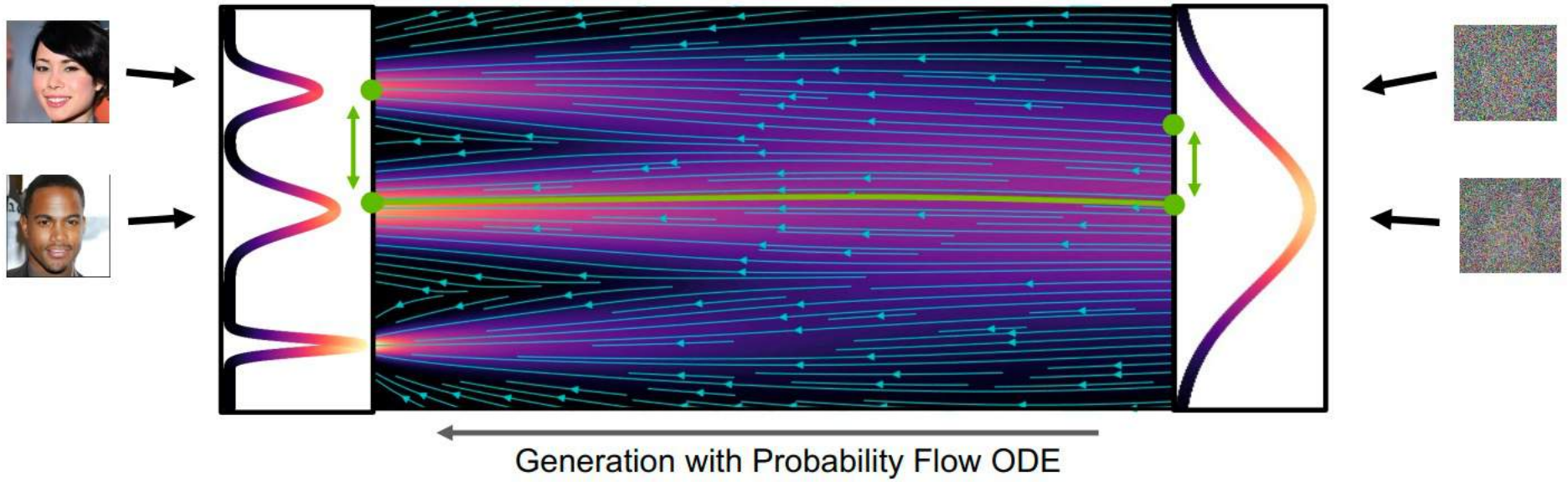
$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right]\mathrm{d}t, \quad \text{(Continuous Normalizing Flow)}$$

- The equivalent ODE can help compute the exact likelihood

$$\log p_0(\mathbf{x}(0)) = \log p_T(\mathbf{x}(T)) + \int_0^T \nabla \cdot \tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)\mathrm{d}t, \quad \text{(Continuous Change of Variable)}$$

$$\nabla \cdot \tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t) = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\boldsymbol{\epsilon}^{\mathsf{T}} \nabla \tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t) \boldsymbol{\epsilon}], \quad \text{(Skilling-Hutchinson trace estimator)}$$

Diffusion ODEs



Continuous changes in latent space result in continuous, semantically meaningful changes in data space.

Recall DDIM

- Forward process and reverse process

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

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

- They correspond to the equivalent SDEs that have the sample marginal distribution as the original one

$$dx = [f(x,t) - (1-\frac{\lambda}{2})g(t)^2\nabla_{\mathrm{x}}\log p_t(x)]dt + \sqrt{1-\lambda}\mathrm{g}(t)dw$$

- DDIM corresponds to the deterministic ODE ($\lambda = 1$) with a special sampling scheme.

Faster Solvers for Diffusion ODEs

- DPM-solver: Rearrange the ODE from $\frac{d\mathbf{x}_t}{dt} + \frac{1}{2}\beta_t \mathbf{x}_t = \frac{\beta_t}{2\sqrt{1-\alpha_t}}\epsilon_\theta(\mathbf{x}_t, t)$

  1. Find the integrator factor $I(t) = \exp(\int \frac{1}{2}\beta_t dt)$.
  2. Multiply the whole ODE by the integrator factor

$$\alpha_t = \exp\left(-\int_0^t \beta_\tau d\tau\right) \quad \therefore I(t) = \frac{1}{\sqrt{\alpha_t}}$$

$$\frac{d}{dt}\left(\frac{\mathbf{x}_t}{\sqrt{\alpha_t}}\right) = \frac{1}{\sqrt{\alpha_t}}\left(\frac{\beta_t}{2\sqrt{1-\alpha_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) \xrightarrow{\text{integral solution}} \mathbf{x}_s = \frac{\sqrt{\alpha_s}}{\sqrt{\alpha_t}}\mathbf{x}_t + \sqrt{\alpha_s}\int_t^s \frac{\beta_\tau}{2\sqrt{\alpha_\tau}\sqrt{1-\alpha_\tau}}\epsilon_\theta(\mathbf{x}_\tau, \tau)d\tau$$

  3. Change of Variable: Log-SNR $\lambda = \frac{1}{2}\log\left(\frac{\alpha}{1-\alpha}\right)$. $\quad dt = -\frac{2(1-\alpha_t)}{\beta_t}d\lambda$

$$\mathbf{x}_s = \frac{\sqrt{\alpha_s}}{\sqrt{\alpha_t}}\mathbf{x}_t - \sqrt{\alpha_s}\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\epsilon_\theta(\hat{\mathbf{x}}_\lambda, \lambda)d\lambda$$

Zhang and Chen, "Fast Sampling of Diffusion Models with Exponential Integrator"
Lu et al. "DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Step"

DPM-Solver

- Numeircal scheme: approximate the integral

$$\mathbf{x}_s = \frac{\sqrt{\alpha_s}}{\sqrt{\alpha_t}}\mathbf{x}_t - \sqrt{\alpha_s}\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\epsilon_\theta(\hat{\mathbf{x}}_\lambda, \lambda)d\lambda$$

- DPM-Solver-1 (First Order / DDIM):

$$\mathbf{x}_s \approx \frac{\sqrt{\alpha_s}}{\sqrt{\alpha_t}}\mathbf{x}_t - \sqrt{\alpha_s}\epsilon_\theta(\mathbf{x}_t, t)\int_{\lambda_t}^{\lambda_s} e^{-\lambda}d\lambda \quad \Rightarrow \quad \mathbf{x}_s = \frac{\sqrt{\alpha_s}}{\sqrt{\alpha_t}}\mathbf{x}_t - \sqrt{\alpha_s}\left(\frac{\sigma_t}{\sqrt{\alpha_t}} - \frac{\sigma_s}{\sqrt{\alpha_s}}\right)\epsilon_\theta(\mathbf{x}_t, t)$$

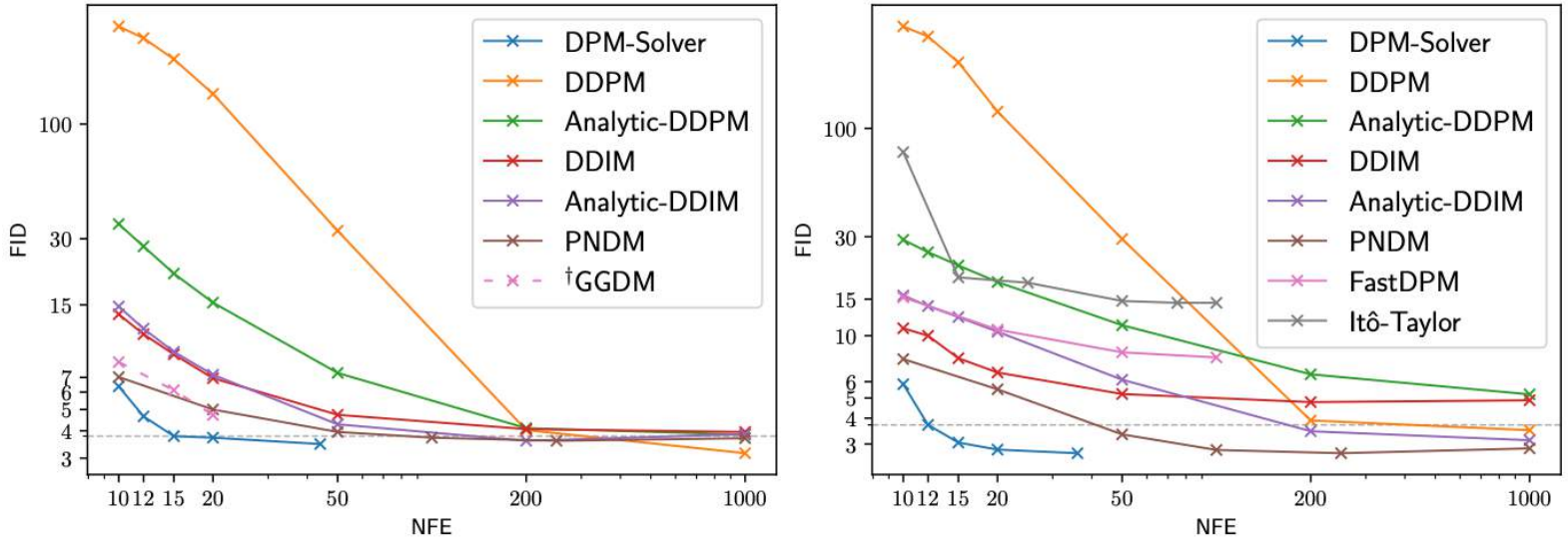
- DPM-Solver-2 (Second Order)

$$\epsilon_\theta(\hat{\mathbf{x}}_\lambda, \lambda) \approx \epsilon_\theta(\mathbf{x}_{t_i}, t_i) + \underbrace{\frac{\epsilon_\theta(\mathbf{x}_{t_i}, t_i) - \epsilon_\theta(\mathbf{x}_{t_{i+1}}, t_{i+1})}{\lambda_{t_i} - \lambda_{t_{i+1}}}}_{\text{Slope } D_i} \cdot (\lambda - \lambda_{t_i})$$

Despite the more expensive steps, higher-order are usually more efficient since they require much fewer steps to converge.

Diffusion ODEs

- Faster sampling using high-order ODE solvers, e.g. Runge-Kutta, exponential integrators, et al.



CIFAR-10 (Left)
CelebA 64x64 (right)

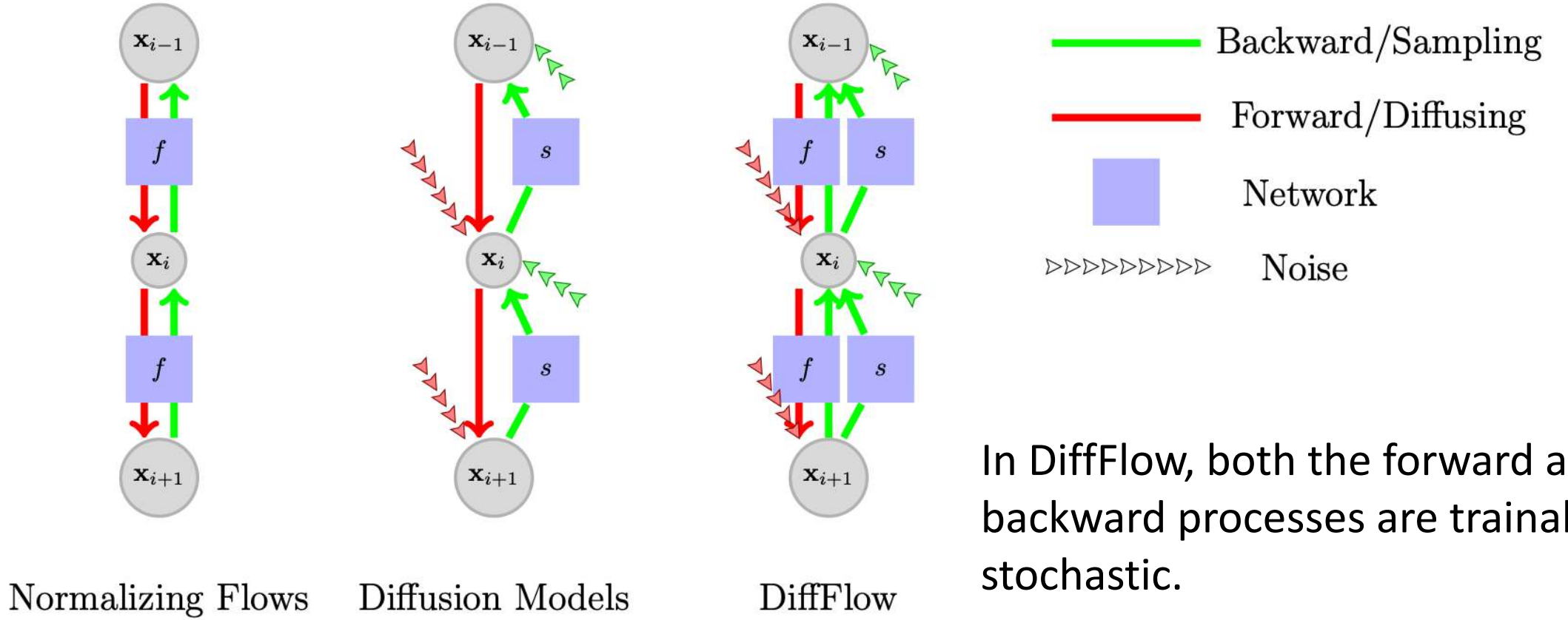
Zhang and Chen, "Fast Sampling of Diffusion Models with Exponential Integrator"
Lu et al. "DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Step"

Summary

- DDPM is a specific type of Hierarchical VAE.
- Sore-based generative models builds up a SDE framework that
  - can leverage broad existing literature on advanced and fast SDE and ODE solvers.
  - have equivalent deterministic probability flow ODE and thus allows exact likelihood computation and semantic interpolation.
  - can connect to other generative models like normalizing flows.

More Diffusion Training and Acceleration

DiffFlow



In DiffFlow, both the forward and the backward processes are trainable and stochastic.

Zhang et al. “Diffusion Normalizing Flow ”, 2021

DiffFlow

- Forward and backward process of DiffFlow

$$\begin{aligned}\mathbf{x}_{i+1} &= \mathbf{x}_i + \mathbf{f}_i(\mathbf{x}_i)\Delta t_i + g_i \delta_i^F \sqrt{\Delta t_i} \\ \mathbf{x}_i &= \mathbf{x}_{i+1} - [\mathbf{f}_{i+1}(\mathbf{x}_{i+1}) - g_{i+1}^2 \mathbf{s}_{i+1}(\mathbf{x}_{i+1})]\Delta t_i + g_{i+1}\delta_i^B \sqrt{\Delta t_i},\end{aligned}$$

- KL divergence between trajectory distributions

$$KL(p_F(\tau)|p_B(\tau)) = \underbrace{\mathbb{E}_{\tau\sim p_F}[\log p_F(\mathbf{x}_0)]}_{L_0} + \underbrace{\mathbb{E}_{\tau\sim p_F}[-\log p_B(\mathbf{x}_N)]}_{L_N} + \sum_{i=1}^{N-1} \underbrace{\mathbb{E}_{\tau\sim p_F}[\log \frac{p_F(\mathbf{x}_i|\mathbf{x}_{i-1})}{p_B(\mathbf{x}_{i-1}|\mathbf{x}_i)}]}_{L_i}.$$

$$L := \mathbb{E}_{\tau\sim p_F}[-\log p_B(\mathbf{x}_N) + \sum_i \frac{1}{2}(\delta_i^B(\tau))^2] = \mathbb{E}_{\delta^F;\mathbf{x}_0\sim p_0}[-\log p_B(\mathbf{x}_N) + \sum \frac{1}{2}(\delta_i^B(\tau))^2],$$

$$\delta_i^B(\tau) = \frac{1}{g_{i+1}\sqrt{\Delta t}}\left[\mathbf{x}_i - \mathbf{x}_{i+1} + [\mathbf{f}_{i+1}(\mathbf{x}_{i+1}) - g_{i+1}^2 \mathbf{s}_{i+1}(\mathbf{x}_{i+1})]\Delta t\right].$$

Zhang et al. “Diffusion Normalizing Flow ”, 202

DiffFlow

- Differentiating through the operations of the forward pass requires unrolling networks N times .
- When a naive backpropagation strategy is used, the memory consumption explodes quickly.

Algorithm 2 Stochastic Adjoint Algorithm for DiffFlow

Input: Forward trajectory $\{\mathbf{x}_i\}_{i=0}^{N}$

$\frac{\partial L}{\partial \mathbf{x}_N} = \frac{1}{2}\frac{\partial (\delta_N^B(\tau))^2}{\partial \mathbf{x}_N} - \frac{\partial \log p_B(\mathbf{x}_N)}{\mathbf{x}_N}$

$\frac{\partial L}{\partial \theta} = 0$

for $i = N, N-1, \cdots, 1$ **do**

$\frac{\partial L}{\partial \mathbf{x}_{i-1}} = (\frac{\partial L}{\partial \mathbf{x}_i} + \frac{1}{2}\frac{\partial (\delta_i^B(\tau))^2}{\partial \mathbf{x}_i})\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} + \frac{1}{2}\frac{\partial (\delta_i^B(\tau))^2}{\partial \mathbf{x}_{i-1}}$

$\frac{\partial L}{\partial \theta} += \frac{1}{2}\frac{\partial (\delta_i^B(\tau))^2}{\partial \theta} + (\frac{\partial L}{\partial \mathbf{x}_i} + \frac{1}{2}\frac{\partial (\delta_i^B(\tau))^2}{\partial \mathbf{x}_i})\frac{\partial \mathbf{x}_i}{\partial \theta}$

end for

Zhang et al. “Diffusion Normalizing Flow ”, 2021

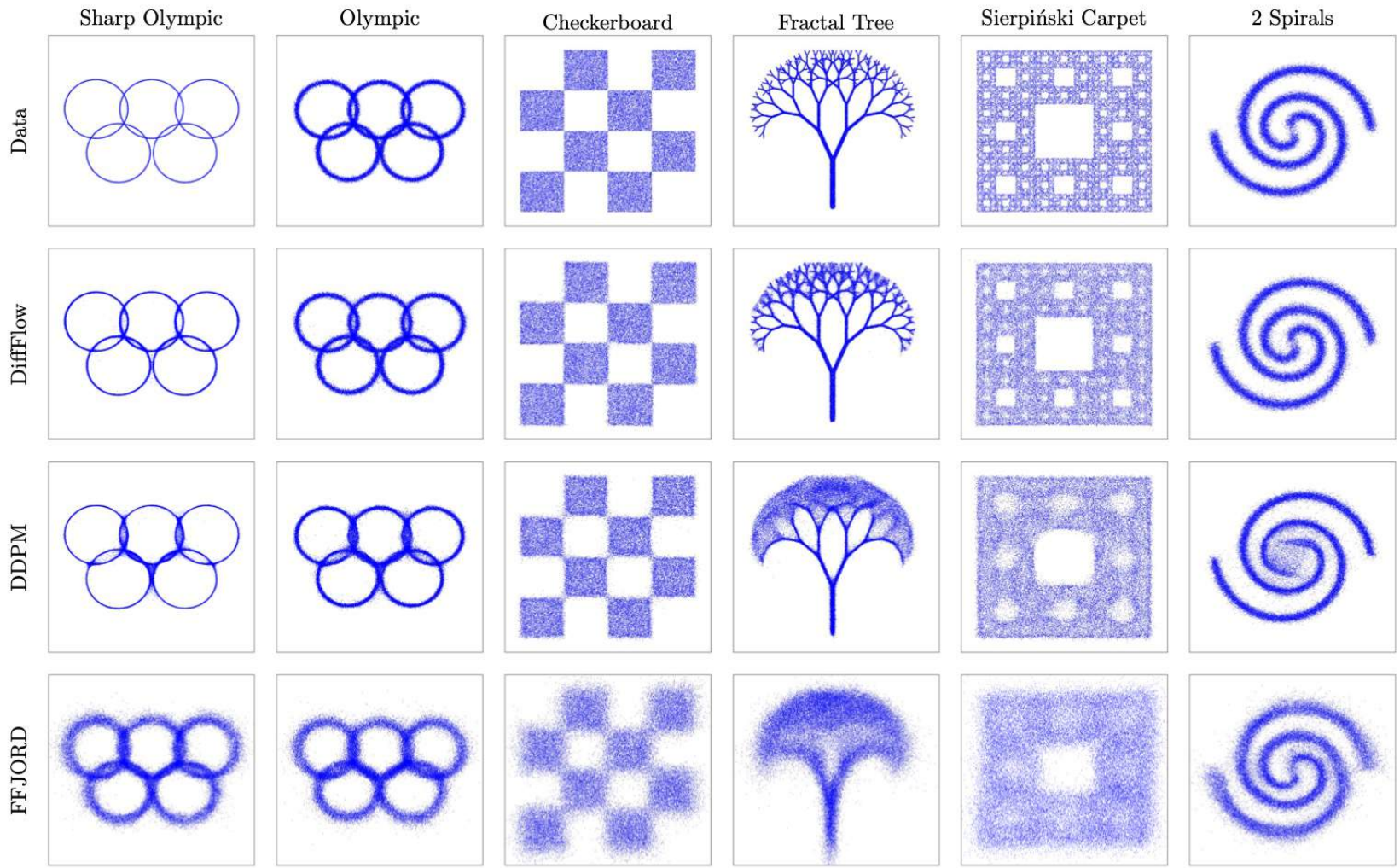


Figure 5: Samples from DiffFlow, DDPM and FFJORD on 2-D datasets. All three models have reasonable performance on datasets that have smooth underlying distributions. But only DiffFlow is capable to capture complex patterns and provides sharp samples when dealing with more challenging datasets.

EM Algorithm

- Training the forward and backward model alternatively

$$\pi^{2n+1}(\boldsymbol{x},\boldsymbol{z}) = \underset{\pi(\boldsymbol{x},\boldsymbol{z})}{\arg\min} \left\{ D_{\mathrm{KL}}(\pi(\boldsymbol{x},\boldsymbol{z}) || \pi^{2n}(\boldsymbol{x},\boldsymbol{z})) : \ \pi_{\boldsymbol{x}}(\boldsymbol{x}) = \mu(\boldsymbol{x}) \right\},$$

$$\pi^{2n+2}(\boldsymbol{x},\boldsymbol{z}) = \underset{\pi(\boldsymbol{x},\boldsymbol{z})}{\arg\min} \left\{ D_{\mathrm{KL}}(\pi(\boldsymbol{x},\boldsymbol{z}) || \pi^{2n+1}(\boldsymbol{x},\boldsymbol{z})) : \ \pi_{\boldsymbol{z}}(\boldsymbol{z}) = \nu(\boldsymbol{z}) \right\}, \qquad n \geq 0,$$

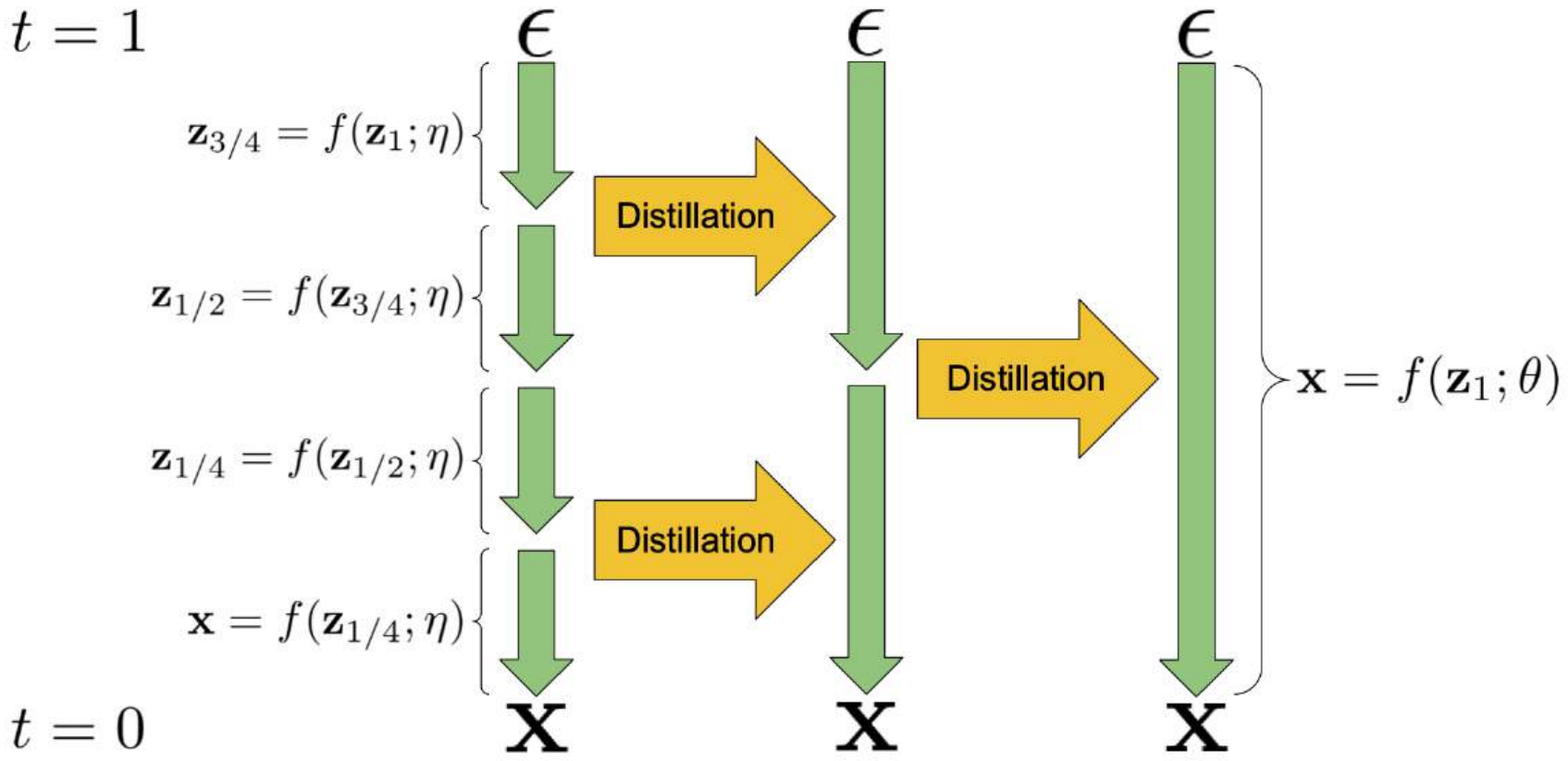
$$\theta_{n+1} = \underset{\theta}{\arg\min}\, \mathcal{L}_{D_{\mathrm{KL}}}(\phi_n, \theta), \qquad \phi_{n+1} = \underset{\phi}{\arg\min}\, \mathcal{L}_{D_{\mathrm{KL}}}(\phi, \theta_{n+1}),$$

$$\mathcal{L}_D(\phi,\theta) := D\left(q^{\phi}(\boldsymbol{z}|\boldsymbol{x})\mu(\boldsymbol{x}) \middle|\middle| p^{\theta}(\boldsymbol{x}|\boldsymbol{z})\nu(\boldsymbol{z})\right),$$

EM Algorithm

Vagars et al. “TRANSPORT MEETS VARIATIONAL INFERENCE: CONTROLLED MONTE CARLO DIFFUSIONS”, 2024

Progressive Distillation



Salimans and ho, "Progressive Distillation for Fast Sampling of Diffusion Models", 2022

Algorithm 1 Standard diffusion training

Require: Model $\hat{\mathbf{x}}_\theta(\mathbf{z}_t)$ to be trained
Require: Data set $\mathcal{D}$
Require: Loss weight function $w()$

while not converged **do**
  $\mathbf{x} \sim \mathcal{D}$ ▷ Sample data
  $t \sim U[0, 1]$ ▷ Sample time
  $\epsilon \sim N(0, I)$ ▷ Sample noise
  $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$ ▷ Add noise to data

  $\tilde{\mathbf{x}} = \mathbf{x}$ ▷ Clean data is target for $\hat{\mathbf{x}}$
  $\lambda_t = \log[\alpha_t^2/\sigma_t^2]$ ▷ log-SNR
  $L_\theta = w(\lambda_t)\|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$ ▷ Loss
  $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$ ▷ Optimization
end while

Algorithm 2 Progressive distillation

Require: Trained teacher model $\hat{\mathbf{x}}_\eta(\mathbf{z}_t)$
Require: Data set $\mathcal{D}$
Require: Loss weight function $w()$
Require: Student sampling steps N
for K iterations **do**
  $\theta \leftarrow \eta$ ▷ Init student from teacher
  while not converged **do**
    $\mathbf{x} \sim \mathcal{D}$
    $t = i/N, \;\; i \sim Cat[1, 2, \ldots, N]$
    $\epsilon \sim N(0, I)$
    $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$
    `# 2 steps of DDIM with teacher`
    $t' = t - 0.5/N, \quad t'' = t - 1/N$
    $\mathbf{z}_{t'} = \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_t) + \frac{\sigma_{t'}}{\sigma_t}(\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\eta(\mathbf{z}_t))$
    $\mathbf{z}_{t''} = \alpha_{t''} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}}(\mathbf{z}_{t'} - \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}))$
    $\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - (\sigma_{t''}/\sigma_t)\mathbf{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t)\alpha_t}$ ▷ Teacher $\hat{\mathbf{x}}$ target
    $\lambda_t = \log[\alpha_t^2/\sigma_t^2]$
    $L_\theta = w(\lambda_t)\|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$
    $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$
  end while
  $\eta \leftarrow \theta$ ▷ Student becomes next teacher
  $N \leftarrow N/2$ ▷ Halve number of sampling steps
end for

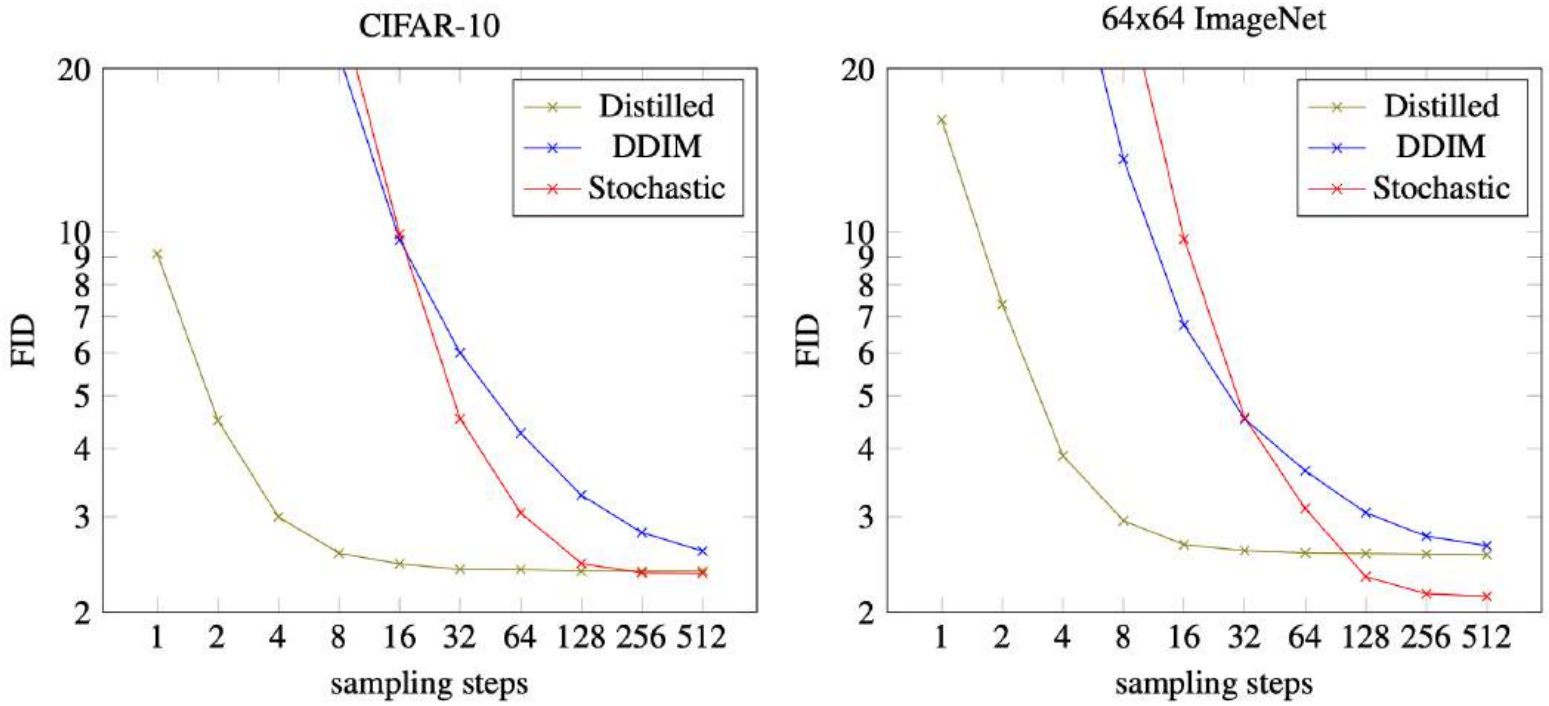
$$\tilde{\mathbf{z}}_{t''} = \alpha_{t''}\tilde{\mathbf{x}} + \frac{\sigma_{t''}}{\sigma_t}(\mathbf{z}_t - \alpha_t \tilde{\mathbf{x}}) = \mathbf{z}_{t''}$$

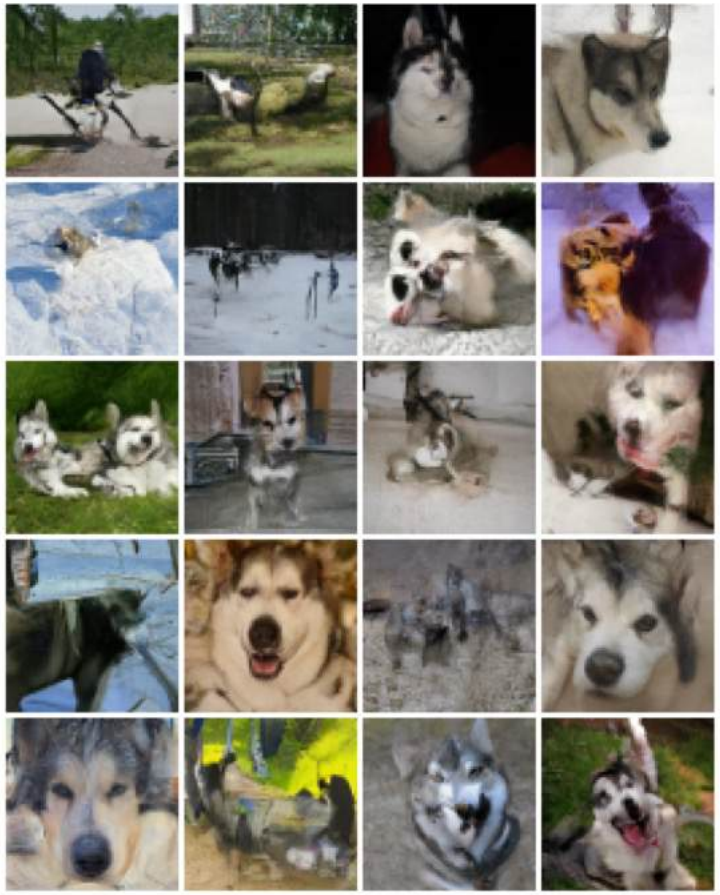
$$= \left(\alpha_{t''} - \frac{\sigma_{t''}}{\sigma_t}\alpha_t\right)\tilde{\mathbf{x}} + \frac{\sigma_{t''}}{\sigma_t}\mathbf{z}_t = \mathbf{z}_{t''}$$

$$\left(\alpha_{t''} - \frac{\sigma_{t''}}{\sigma_t}\alpha_t\right)\tilde{\mathbf{x}} = \mathbf{z}_{t''} - \frac{\sigma_{t''}}{\sigma_t}\mathbf{z}_t$$

$$\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - \frac{\sigma_{t''}}{\sigma_t}\mathbf{z}_t}{\alpha_{t''} - \frac{\sigma_{t''}}{\sigma_t}\alpha_t}$$

Progressive Distillation



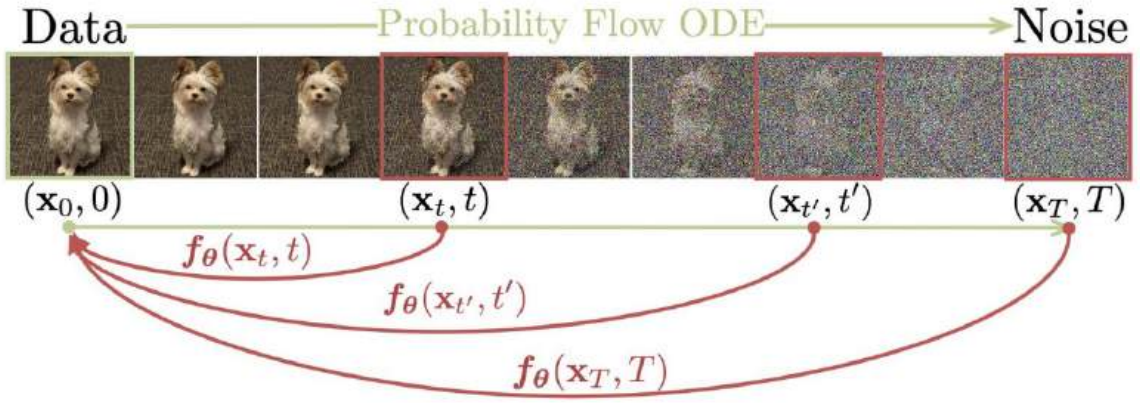


(c) 1 sampling step

Salimans and ho, “Progressive Distillation for Fast Sampling of Diffusion Models", 2022

Consistency Models

- Map any point on the probability floe ODE trajectory to its origin



Algorithm 1 Multistep Consistency Sampling

Input: Consistency model $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot,\cdot)$, sequence of time points $\tau_1 > \tau_2 > \cdots > \tau_{N-1}$, initial noise $\hat{\mathbf{x}}_T$

$\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_T, T)$

for $n = 1$ **to** $N-1$ **do**

  Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

  $\hat{\mathbf{x}}_{\tau_n} \leftarrow \mathbf{x} + \sqrt{\tau_n^2 - \epsilon^2}\mathbf{z}$

  $\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{\tau_n}, \tau_n)$

end for

Output: $\mathbf{x}$

- One step generation: $f_\theta(x_T, T)$;
- Multistep generation: alternating denoising and injecting noise steps

Consistency Models. Yang SongPrafulla, Dhariwal,Mark Chen, Ilya Sutskever, 2023

Training w/ Pre-trained Models

- Consistency Distillation (CD):

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

One ODE solver step using pretrained diffusion model

$$\mathbb{E}[\lambda(t_n)\| \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}, t_n)\|]$$

Weighting function

Student model

Teacher model

$\boldsymbol{\theta}^- = \mathrm{EMA}(\boldsymbol{\theta})$

$$\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n} = \mathbf{x}_{t_{n+1}} - (t_n - t_{n+1}) t_{n+1} \boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1})$$

→ one step DDIM from the pre-trained diffusion model

(the forward model is $dx = \sqrt{2t}\mathrm{dw}$)

Training w/o Pre-trained Models

- Consistency Training (CT) loss

$$\mathbb{E}[\lambda(t_n)\|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}+t_{n+1}\mathbf{z},t_{n+1})-\boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}+t_n\mathbf{z},t_n)\|]$$

- When $|t_{n+1}-t_n|\to 0$, the continuous CT loss with ℓ_2 metrics is

$$\mathbb{E}\left[\lambda(t)\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)^{\intercal}\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)}{\partial t}+\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)}{\partial \mathbf{x}_t}\cdot\frac{\mathbf{x}_t-\mathbf{x}}{t}\right)\right]$$

METHOD	NFE (↓)	FID (↓)	Prec. (↑)	Rec. (↑)
ImageNet 64 × 64				
PD† (Salimans & Ho, 2022)	1	15.39	0.59	0.62
DFNO† (Zheng et al., 2022)	1	8.35		
CD†	1	6.20	0.68	0.63
PD† (Salimans & Ho, 2022)	2	8.95	0.63	**0.65**
CD†	2	**4.70**	**0.69**	0.64
ADM (Dhariwal & Nichol, 2021)	250	**2.07**	0.74	0.63
EDM (Karras et al., 2022)	79	2.44	0.71	**0.67**
BigGAN-deep (Brock et al., 2019)	1	4.06	**0.79**	0.48
CT	1	13.0	0.71	0.47
CT	2	11.1	0.69	0.56

Flow Matching

Reference: An introduction to Flow Matching · Cambridge MLG Blog

Recaps: Continuous Normalizing Flow

- The flow is

$$\frac{dx_t}{dt} = u_\theta(t, x_t)$$

- Continuous Change of Variables

$$\frac{d \log p_t(x_t)}{dt} = -\text{div}\left(u_\theta(t, x_t)\right)$$

- Challenges of training Continuous Normalizing Flow (CNF)
  - Expensive numerical ODE simulations at training time
  - Estimators for the divergence are hard to scale with high dimension

Flow Matching

- Flow matching is a simulation-free way to train CNF

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1]}\mathbb{E}_{x\sim p_t}\left[\|u_\theta(t,x) - u(t,x)\|^2\right].$$

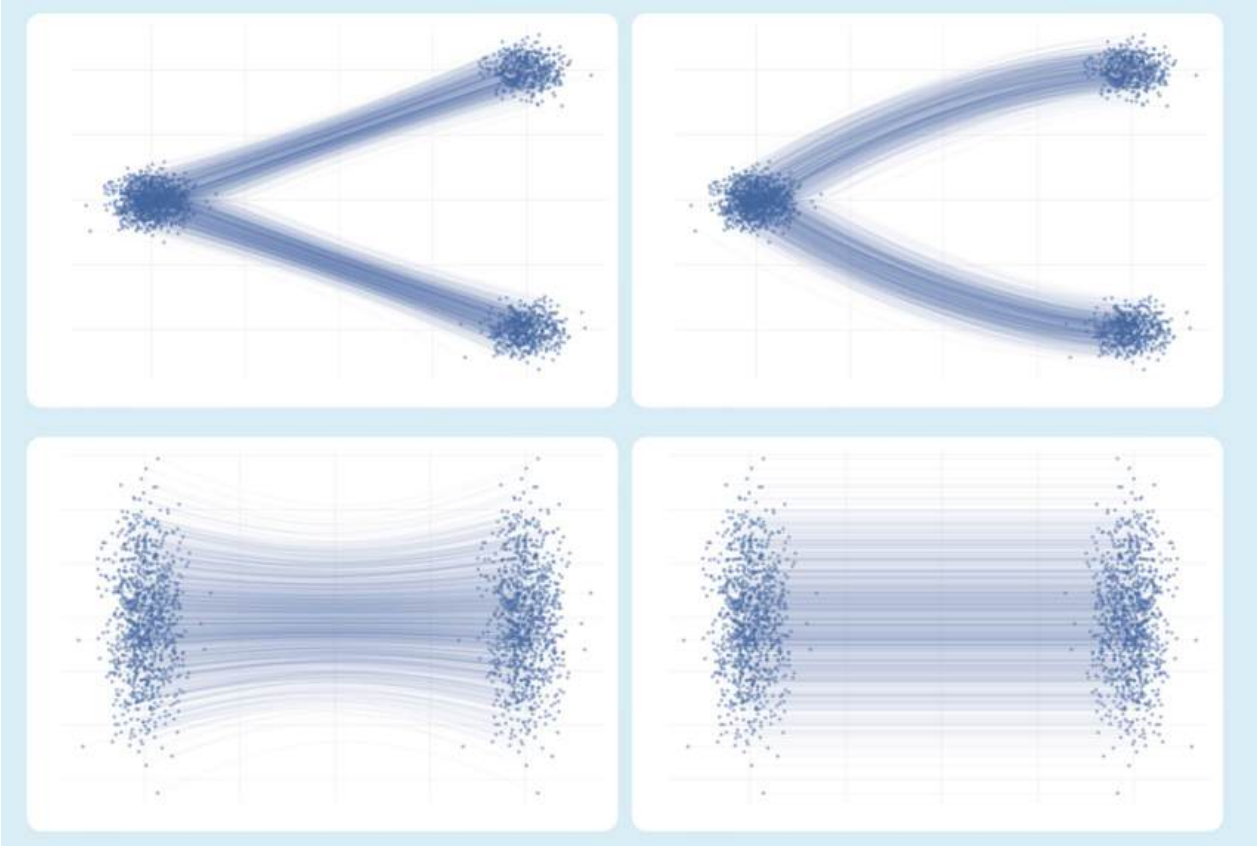
$u(t,x)$ is the "true" vector field that maps p_0 to p_1

- Flow matching is just performing regression on $u_t(x)$ $\forall t \in [0,1]$.
- The problem is how to do the regression without having to compute explicitly $u(t,x)$.

Non-uniqueness of Vector Field

- There are indeed many valid choices for $u(t, x)$ that maps p_0 to p_1

Different paths with the same endpoint marginals.



Conditional Flow

- The idea is quite similar as score-matching for diffusion model: turn the marginal score-matching to conditional score matching.
- The marginal distribution can be expressed as

$$p_t(x_t) = \int q_1(x_1)\, p_{t|1}(x_t \mid x_1)\, \mathrm{d}x_1.$$

- The transport equation for the conditional probability is

$$\frac{\partial p_t(x \mid x_1)}{\partial t} = -\nabla \cdot \big(u_t(x \mid x_1) p_t(x \mid x_1)\big).$$

- The marginal vector $u_t(x)$ can be expressed as

$$u_t(x) = \mathbb{E}_{x_1 \sim p_{1|t}}\left[u_t(x \mid x_1)\right]$$
$$= \int u_t(x \mid x_1) \frac{p_t(x \mid x_1) q_1(x_1)}{p_t(x)} \mathrm{d}x_1.$$

Posterior $p(x_1|x_t)$

$$\Longrightarrow \quad u_t(x) = \mathbb{E}_{x_1|x_t} u(x_t|x_1)$$

The probability path p_t satisfies the transport equation:

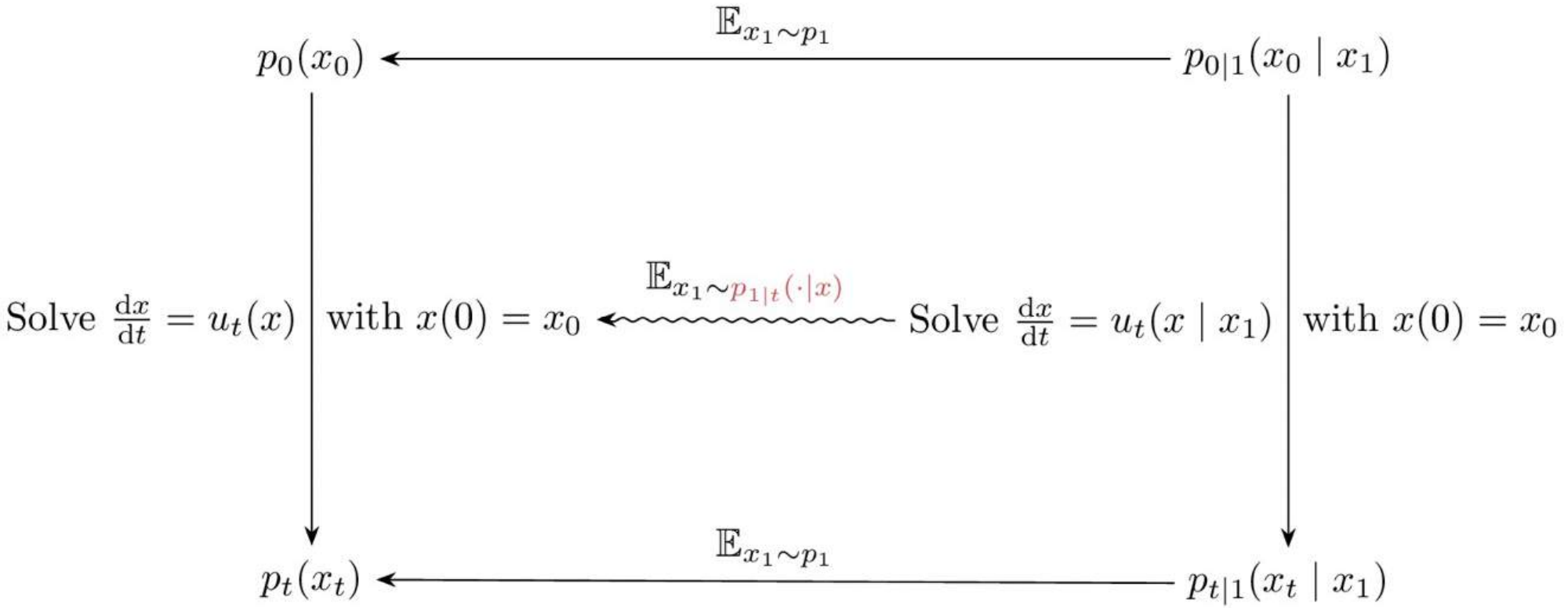
$$\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot \big(u_t(x) p_t(x)\big).$$ (rearrangement of the continuous change of variable)

Writing out the left-hand side:

$$\begin{aligned}
\frac{\partial p_t(x)}{\partial t} &= \frac{\partial}{\partial t} \int p_t(x \mid x_1) q(x_1)\, \mathrm{d}x_1 \\
&= \int \frac{\partial}{\partial t} \big(p_t(x \mid x_1)\big) q(x_1)\, \mathrm{d}x_1 \\
&= -\int \nabla \cdot \big(u_t(x \mid x_1) p_t(x \mid x_1)\big) q(x_1)\, \mathrm{d}x_1 \\
&= -\int \nabla \cdot \big(u_t(x \mid x_1) p_t(x \mid x_1) q(x_1)\big)\, \mathrm{d}x_1 \\
&= -\nabla \cdot \int u_t(x \mid x_1) p_t(x \mid x_1) q(x_1)\, \mathrm{d}x_1 \\
&= -\nabla \cdot \left(\int u_t(x \mid x_1) \frac{p_t(x \mid x_1) q(x_1)}{p_t(x)} p_t(x)\, \mathrm{d}x_1 \right) \\
&= -\nabla \cdot \left(\int u_t(x \mid x_1) \frac{p_t(x \mid x_1) q(x_1)}{p_t(x)}\, \mathrm{d}x_1\, p_t(x) \right) \\
&= -\nabla \cdot \big(u_t(x) p_t(x)\big)
\end{aligned}$$

$$\begin{aligned}
u_t(x) &= \mathbb{E}_{x_1 \sim p_{1|t}} \left[u_t(x \mid x_1) \right] \\
&= \int u_t(x \mid x_1) \frac{p_t(x \mid x_1) q_1(x_1)}{p_t(x)} \mathrm{d}x_1.
\end{aligned}$$

Conditional Flow

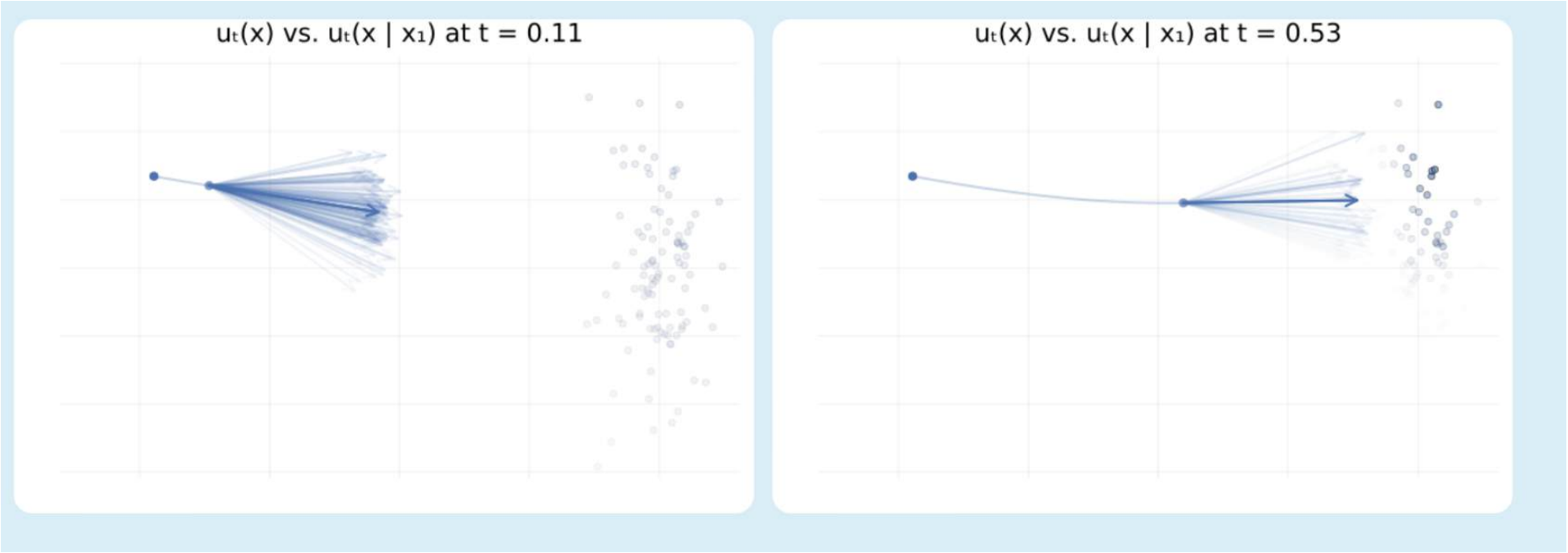


The relation between the vector fields $u_t(x)$, $u_t(x|x_1)$, and their induced marginal and conditional densities. (t=0: normal distribution, t=1: data distribution)

Example

- Practical example of conditional vector field ($\mu_t(x_1) \to x_1, \sigma_t(x_1) \to 0$)

$$x_t = \mu_t(x_1) + \sigma_t(x_1)x_0 \qquad p_t(x \mid x_1) = \mathcal{N}(x; \mu_t(x_1), \sigma_t(x_1)^2 \mathrm{I}) \qquad u_t(x \mid x_1) = \frac{\dot{\sigma}_t(x_1)}{\sigma_t(x_1)}(x - \mu_t(x_1)) + \dot{\mu}_t(x_1)$$



Conditional flow vs. marginal flow

Conditional Flow Matching

- Flow Matching → Conditional Flow Matching

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],x\sim p_t}\left[\|u_\theta(t,x) - u(t,x)\|^2\right],$$

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],x_1\sim q,x_t\sim p_t(x|x_1)}\left[\|u_\theta(t,x) - u_t(x \mid x_1)\|^2\right].$$

$$\mathcal{L}_{FM}(\theta) = \mathcal{L}_{CFM}(\theta) + const.$$

(Recaps: score-matching → conditional score matching in diffusion model:

$\mathbb{E}_{x_t}||s_\theta(x_t,t) - \nabla \log p(x_t)\,||^2 = \mathbb{E}_{x_1,x_t}||s_\theta(x_t,t) - \nabla \log p(x_t|x_1)\,||^2 + c$)

- Opposed to CNF that trains the flow using likelihood loss and does not put any preference over the vector field, CFM loss does specify one via the choice of a conditional vector field.

$$\|u_\theta(t,x) - u_t(x \mid x_1)\|^2 = \|u_\theta(t,x)\|^2 + \|u_t(x \mid x_1)\|^2 - 2\langle u_\theta(t,x), u_t(x \mid x_1)\rangle,$$

$$\|u_\theta(t,x) - u_t(x)\|^2 = \|u_\theta(t,x)\|^2 + \|u_t(x)\|^2 - 2\langle u_\theta(t,x), u_t(x)\rangle.$$

Taking the expectation over the last inner product term:

$$\begin{aligned}
\mathbb{E}_{x\sim p_t}\,\langle u_\theta(t,x), u_t(x)\rangle &= \int \langle u_\theta(t,x), \int u_t(x|x_1)\frac{p_t(x \mid x_1)q(x_1)}{p_t(x)}dx_1\rangle p_t(x)\,\mathrm{d}x \\
&= \int \langle u_\theta(t,x), \int u_t(x \mid x_1)p_t(x \mid x_1)q(x_1)dx_1\rangle\,\mathrm{d}x \\
&= \int\int \langle u_\theta(t,x), u_t(x \mid x_1)\rangle p_t(x \mid x_1)q(x_1)dx_1\,\mathrm{d}x \\
&= \mathbb{E}_{q_1(x_1)p(x|x_1)}\,\langle u_\theta(t,x), u_t(x \mid x_1)\rangle.
\end{aligned}$$

$$\mathbb{E}_{p_t}\|u_\theta(t,x)\|^2 = \int \|u_\theta(t,x)\|^2 p_t(x \mid x_1)q(x_1)\,\mathrm{d}x\,\mathrm{d}x_1 = \mathbb{E}_{q_1(x_1)p(x|x_1)}\|u_\theta(t,x)\|^2$$

Thus: $\mathbb{E}_{x_t}||u_\theta(t,x_t) - u_t(x_t|x_1)||^2 = \mathbb{E}_{x_0,x_t}||u_\theta(t,x_t) - u_t(x_t|x_1)||^2 + c$

Conditional Paths

- The marginal paths does not cross(uniqueness of ODE solution), while the conditional paths may cross.

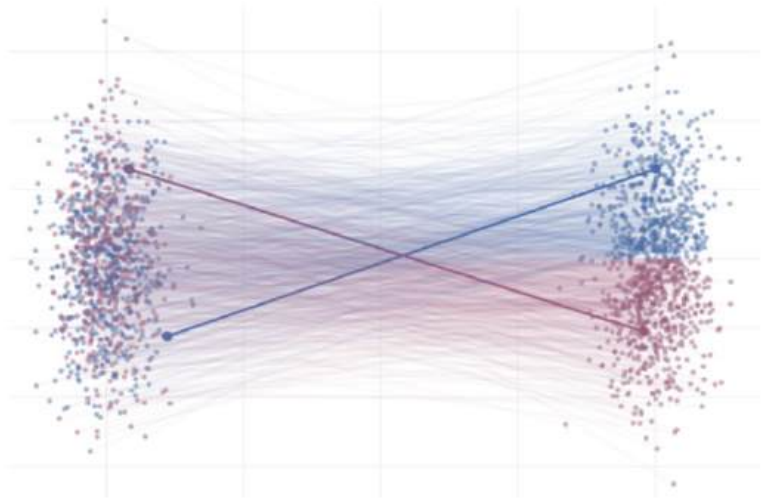


Figure 16: Realizations of conditional paths from $p_0 = p_1 = \mathcal{N}(0, 1)$ for two different $x_1^{(i)}, x_1^{(2)} \sim q$ with conditional vector field given by $u_t(x \mid x_1) = (1 - t)x + tx_1$.

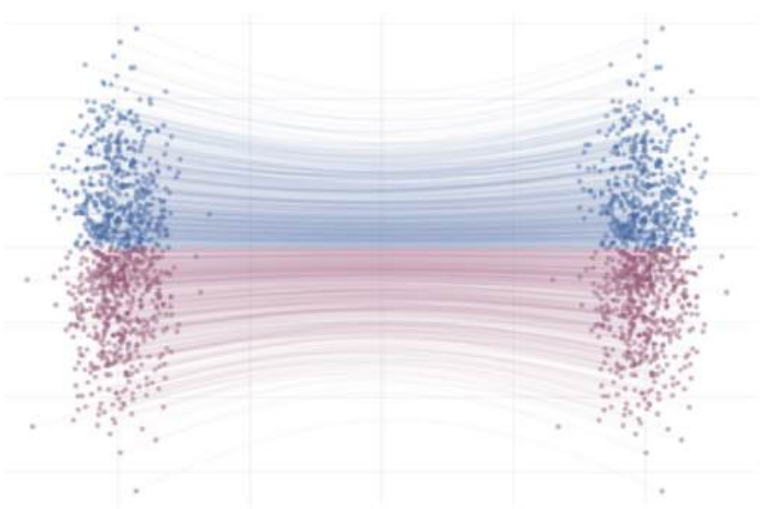


Figure 17: Paths from p_0 to p_1 following the true marginal vector field $u_t(x)$. Paths are highlighted by the sign of the 2nd vector component.

Conditional Paths

- When learning a parameterized vector field via stochastic gradient descent (SGD), we approximate the CFM loss as:

$$\mathcal{L}_{\text{CFM}}(\theta) \approx \frac{1}{2}\left\|u_\theta(t, x_t^{(1)}) - u(t, x_t^{(1)} \mid x_1^{(1)})\right\| + \frac{1}{2}\left\|u_\theta(t, x_t^{(2)}) - u(t, x_t^{(2)} \mid x_1^{(2)})\right\|$$

where $t \sim \mathcal{U}[0,1]$, $x_1^{(1)}, x_1^{(2)} \sim q_1$, and $x_t^{(1)} \sim p_t(\cdot \mid x_1^{(1)}), x_t^{(2)} \sim p_t(\cdot \mid x_1^{(2)})$.

- Training issue: The high variance of the conditional vector field may lead to low convergence rate when the sampled conditional vector field is quite different from the marginal one.
- Sampling issue: Non-straight marginal paths→ODE hard to integrate→more discretization steps→slow sampling at inference.

Coupling

- The coupling of x_0 and x_1 effects the conditional and marginal path.
- Good coupling, e.g. the optimal transport coupling,

$$q(x_1, x_0) = \pi(x_1, x_0) \in \arg\inf_{\pi \in \Pi} \int \|x_1 - x_0\|_2^2 \mathrm{d}\pi(x_1, x_0)$$

yields small training variance and faster sampling.

Coupling in DDPM is one-side.

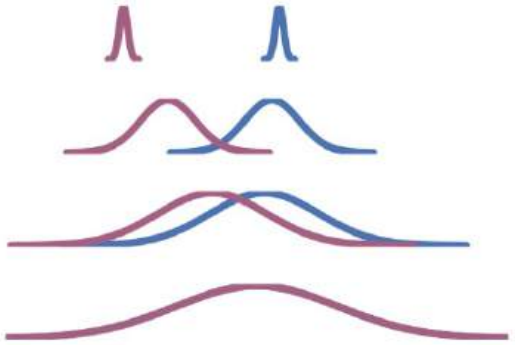


Figure 24: One-sided conditioning (Lipman et al., 2022)

Figure 25: Two-sided conditioning (Tong et al., 2023)

Figure 26: OT coupling (Tong et al., 2023)

Experiments

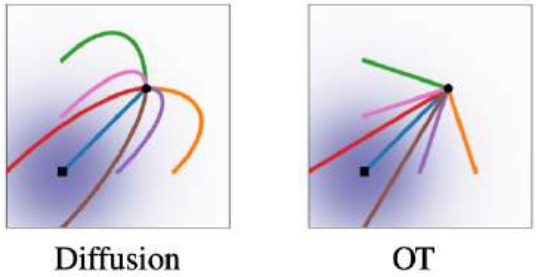


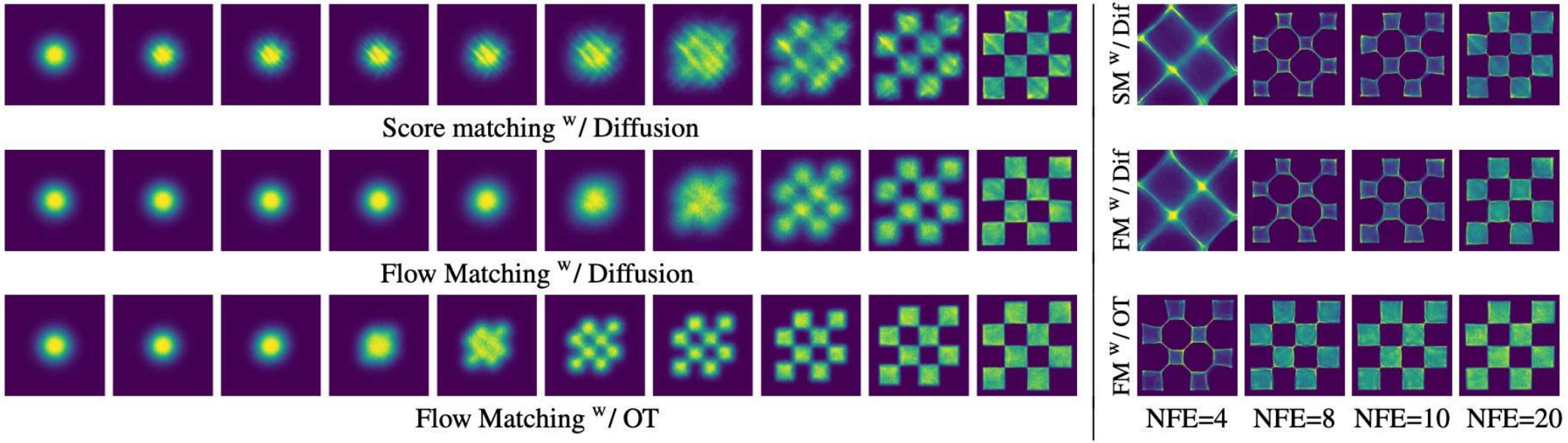
Figure 3: Diffusion and OT trajectories.

- Conditional diffusion vector field

$$u_t(x|x_1) = \frac{\alpha'_{1-t}}{1-\alpha^2_{1-t}}\left(\alpha_{1-t}x - x_1\right)$$

- OT vector field

$$u_t(x|x_1) = \frac{x_1 - (1-\sigma_{\min})x}{1-(1-\sigma_{\min})t}, \qquad \mu_t(x_1) = tx_1, \quad \sigma(x_1) = (1-t) + t\sigma_{min}$$

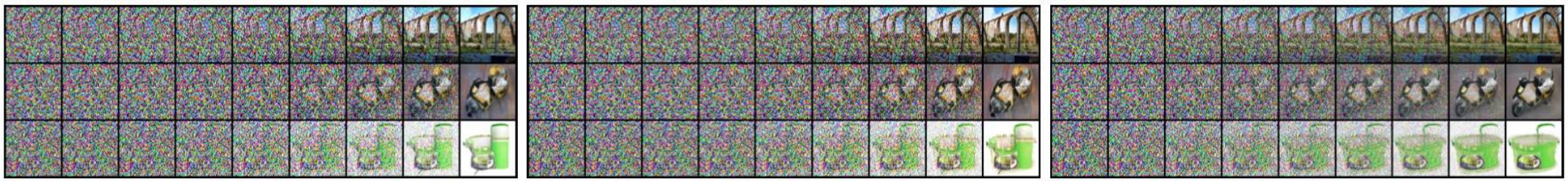


Lipman, Yaron and Chen, Ricky T. Q. and Ben-Hamu, Heli and Nickel, Maximilian and Le, Matt (2022). Flow Matching for Generative Modeling.

Model	CIFAR-10 NLL↓	CIFAR-10 FID↓	CIFAR-10 NFE↓	ImageNet 32×32 NLL↓	ImageNet 32×32 FID↓	ImageNet 32×32 NFE↓	ImageNet 64×64 NLL↓	ImageNet 64×64 FID↓	ImageNet 64×64 NFE↓
Ablations									
DDPM	3.12	7.48	274	3.54	6.99	262	3.32	17.36	264
Score Matching	3.16	19.94	242	3.56	5.68	178	3.40	19.74	441
ScoreFlow	3.09	20.78	428	3.55	14.14	195	3.36	24.95	601
Ours									
FM ^{w}/ Diffusion	3.10	8.06	183	3.54	6.37	193	3.33	16.88	187
FM ^{w}/ OT	**2.99**	**6.35**	**142**	**3.53**	**5.02**	**122**	**3.31**	**14.45**	**138**

Model	ImageNet 128×128 NLL↓	ImageNet 128×128 FID↓
MGAN (Hoang et al., 2018)	–	58.9
PacGAN2 (Lin et al., 2018)	–	57.5
Logo-GAN-AE (Sage et al., 2018)	–	50.9
Self-cond. GAN (Lučić et al., 2019)	–	41.7
Uncond. BigGAN (Lučić et al., 2019)	–	25.3
PGMGAN (Armandpour et al., 2021)	–	21.7
FM ^{w}/ OT	**2.90**	**20.9**

Table 1: Likelihood (BPD), quality of generated samples (FID), and evaluation time (NFE) for the same model trained with different methods.



Score Matching w/ Diffusion | Flow Matching w/ Diffusion | Flow Matching w/ OT

Figure 6: Sample paths from the same initial noise with models trained on ImageNet 64×64. The OT path reduces noise roughly linearly, while diffusion paths visibly remove noise only towards the end of the path. Note also the differences between the generated images.

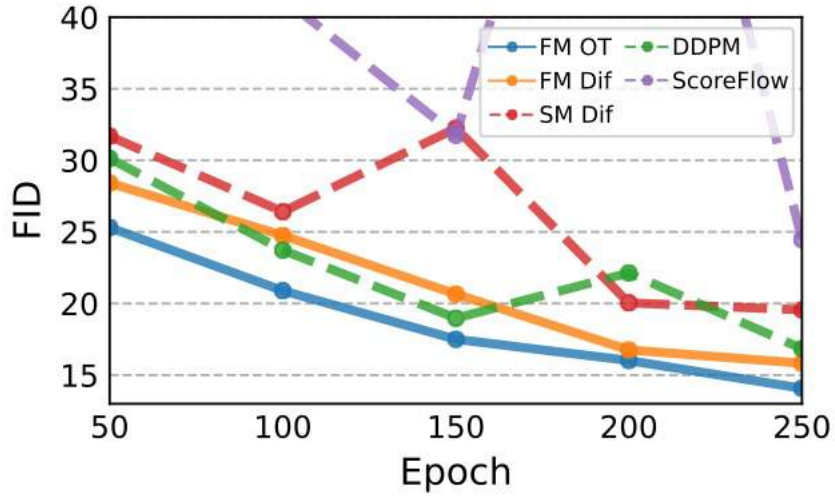
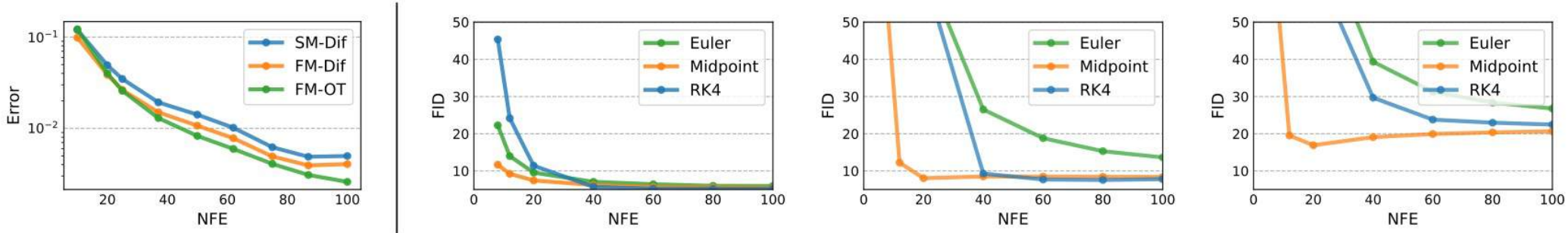


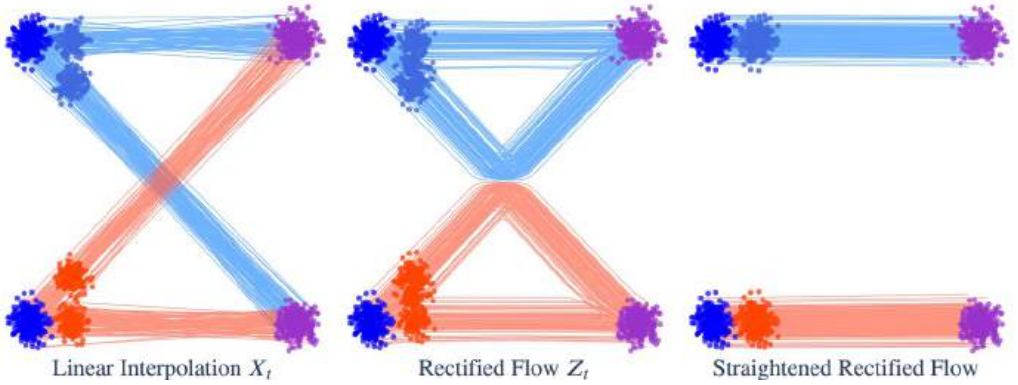
Figure 5: Image quality during training, ImageNet 64×64.



Error of ODE solution | Flow matching ^{w}/ OT | Flow matching ^{w}/ Diffusion | Score matching ^{w}/ Diffusion

Figure 7: Flow Matching, especially when using OT paths, allows us to use fewer evaluations for sampling while retaining similar numerical error (left) and sample quality (right). Results are shown for models trained on ImageNet 32×32, and numerical errors are for the midpoint scheme.

Rectified Flow



- Given a process $\{X_t\}$, the rectified Flow: denote $\{Z_t\} = Rectify\{X_t\}$

$$\dot{Z}_t = v_t^*(Z_t) \quad \text{with} \quad v_t^*(x) = \mathbb{E}\left[\dot{X}_t \mid X_t = x\right],$$

- $\{Z_t\}$ preserves marginal distribution with less cost

$$\mathbf{Law}(Z_t) = \mathbf{Law}(X_t), \quad \forall t \in [0,1], \qquad \mathbb{E}\left[c(Z_1 - Z_0)\right] \leq \mathbb{E}\left[c(X_1 - X_0)\right], \quad \forall \text{convex } c : \mathbb{R}^d \to \mathbb{R}.$$

- k-Rectified Flow: apply rectified flow for k times

$$\texttt{Reflow:} \qquad \{Z_t^{k+1}\} = \texttt{Rectify}(\texttt{Interp}(Z_0^k, Z_1^k)),$$

- Use linear interpolation to straighten the path: $x_t = tx_0 + (1-t)x_1$

Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow, 2022

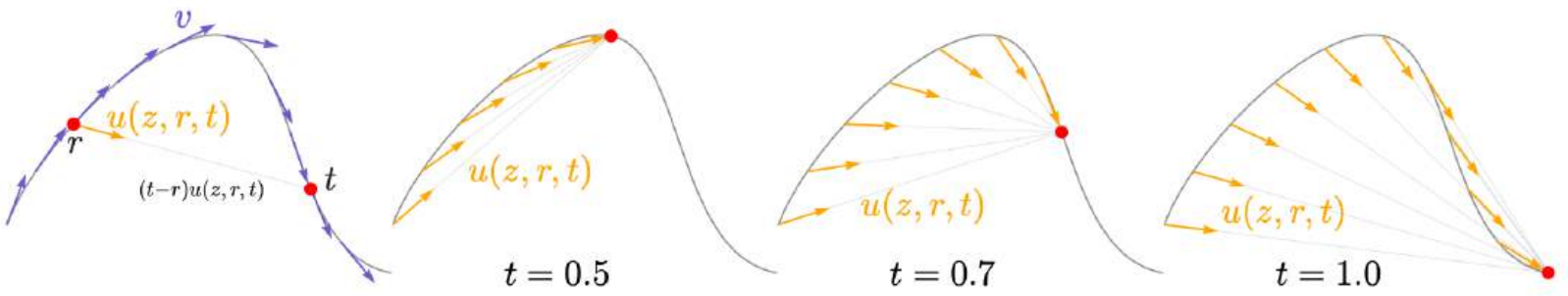
Mean Flow

- Flow model

$$\frac{d}{dt}z_t = v(z_t, t) \qquad z_r = z_t - \int_r^t v(z_\tau, \tau)d\tau$$

- Introduce **average velocity** for sampling acceleration

$$u(z_t, r, t) \triangleq \frac{1}{t-r}\int_r^t v(z_\tau, \tau)d\tau.$$



Mean Flows for One-step Generative Modeling ,2025

Mean Flow

- MeanFlow Identity

$$\frac{d}{dt}(t-r)u(z_t,r,t) = \frac{d}{dt}\int_r^t v(z_\tau,\tau)d\tau \implies u(z_t,r,t) + (t-r)\frac{d}{dt}u(z_t,r,t) = v(z_t,t)$$

$$\underbrace{u(z_t,r,t)}_{\text{average vel.}} = \underbrace{v(z_t,t)}_{\text{instant. vel.}} - (t-r)\underbrace{\frac{d}{dt}u(z_t,r,t)}_{\text{time derivative}}$$

$$\frac{d}{dt}u(z_t,r,t) = v(z_t,t)\partial_z u + \partial_t u,$$

- Training

$$\mathcal{L}(\theta) = \mathbb{E}\left\|u_\theta(z_t,r,t) - \mathrm{sg}(u_{\mathrm{tgt}})\right\|_2^2,$$
$$\text{where} \quad u_{\mathrm{tgt}} = v(z_t,t) - (t-r)\left(v(z_t,t)\partial_z u_\theta + \partial_t u_\theta\right),$$

- Sampling:

$$z_r = z_t - (t-r)u(z_t,r,t)$$

Recall Continuous CT loss

$$\mathbb{E}\left[\lambda(t)\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)^{\mathsf{T}}\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)}{\partial t} + \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}_t,t)}{\partial \mathbf{x}_t}\cdot\frac{\mathbf{x}_t-\mathbf{x}}{t}\right)\right]$$

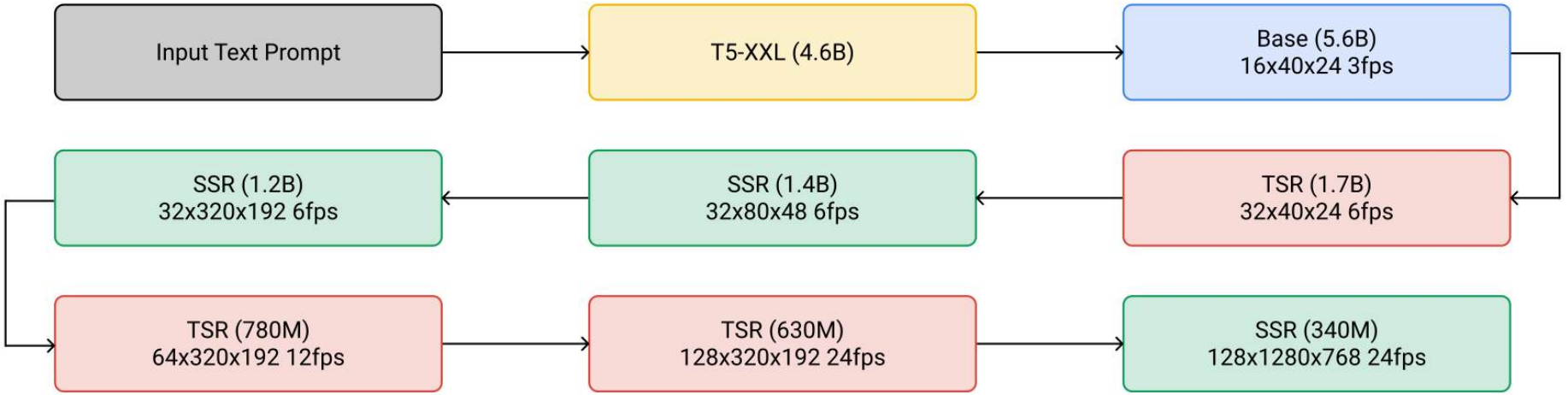
Applications

Video Generation

- Video data: $T \times C \times H \times W$
- Challenges in video generation
  - sampling speed: diffusion models for image generation are already very slow to sample; even slower for sampling 4D data.
  - temporal coherence and consistency: the flow of videos should be smooth.

Video Generation

- Cascaded model for faster sampling



- There are 1 frozen text encoder, 1 base video diffusion model, 3 SSR (spatial super-resolution), and 3 TSR (temporal super- resolution) models – for a total of 7 video diffusion models. They can be trained in parallel.
- Generate entire blocks of video frames at a time to capture the temporal coherence.

Ho et al. “IMAGEN VIDEO: HIGH DEFINITION VIDEO GENERATION WITH DIFFUSION MODELS ”, 2022

Video Generation

- Space-time separable U-Net denoising block for faster sampling

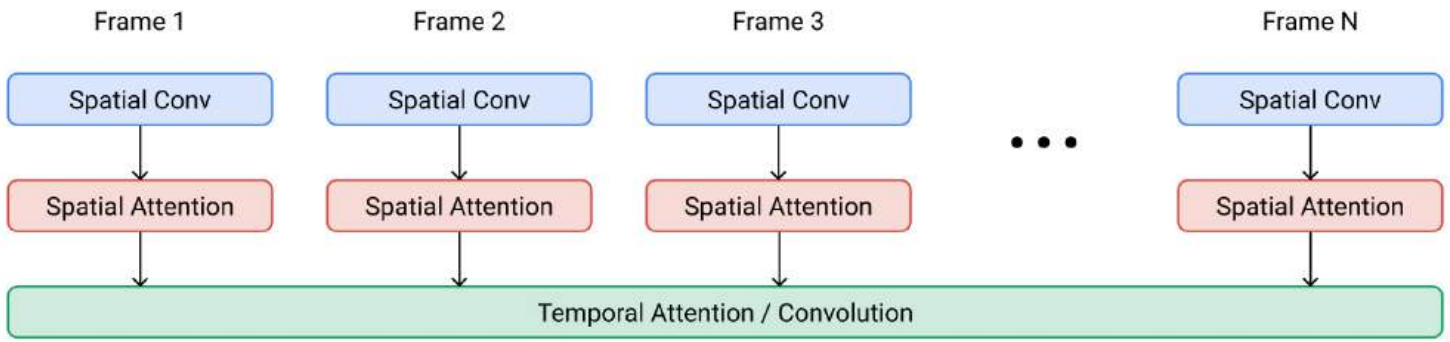
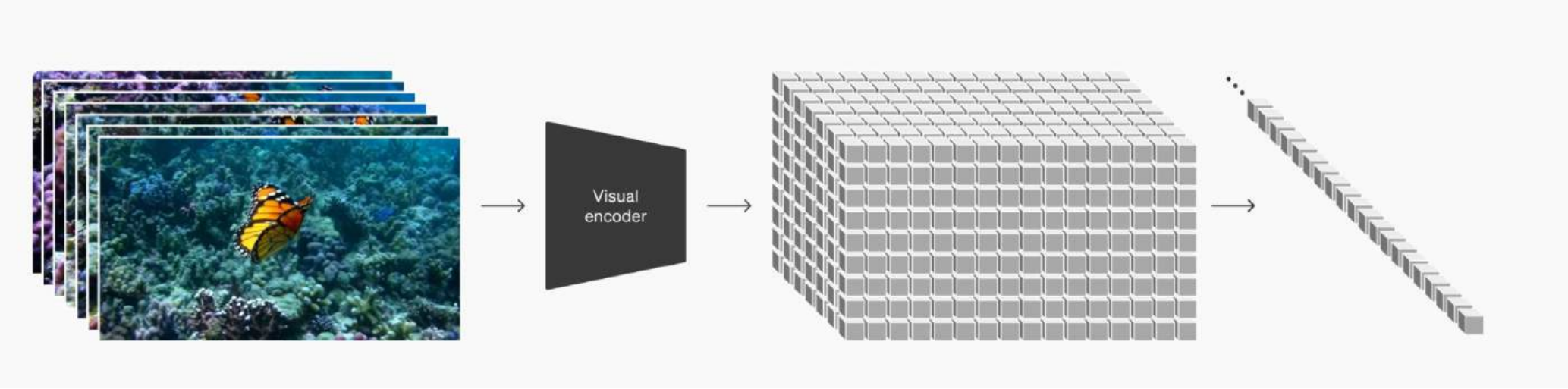


Figure 7: Video U-Net space-time separable block. Spatial operations are performed independently over frames with shared parameters, whereas the temporal operation mixes activations over frames.

Ho et al. “IMAGEN VIDEO: HIGH DEFINITION VIDEO GENERATION WITH DIFFUSION MODELS ”, 2022

SORA

- SORA is a diffusion transformer.



- They first compress videos into a lower-dimensional latent space, and then decompose the representation into spacetime patches.
- These spacetime patches act as tokens in the diffusion transformer generative model.

https://openai.com/index/video-generation-models-as-world-simulators/

Diffusion Transformer

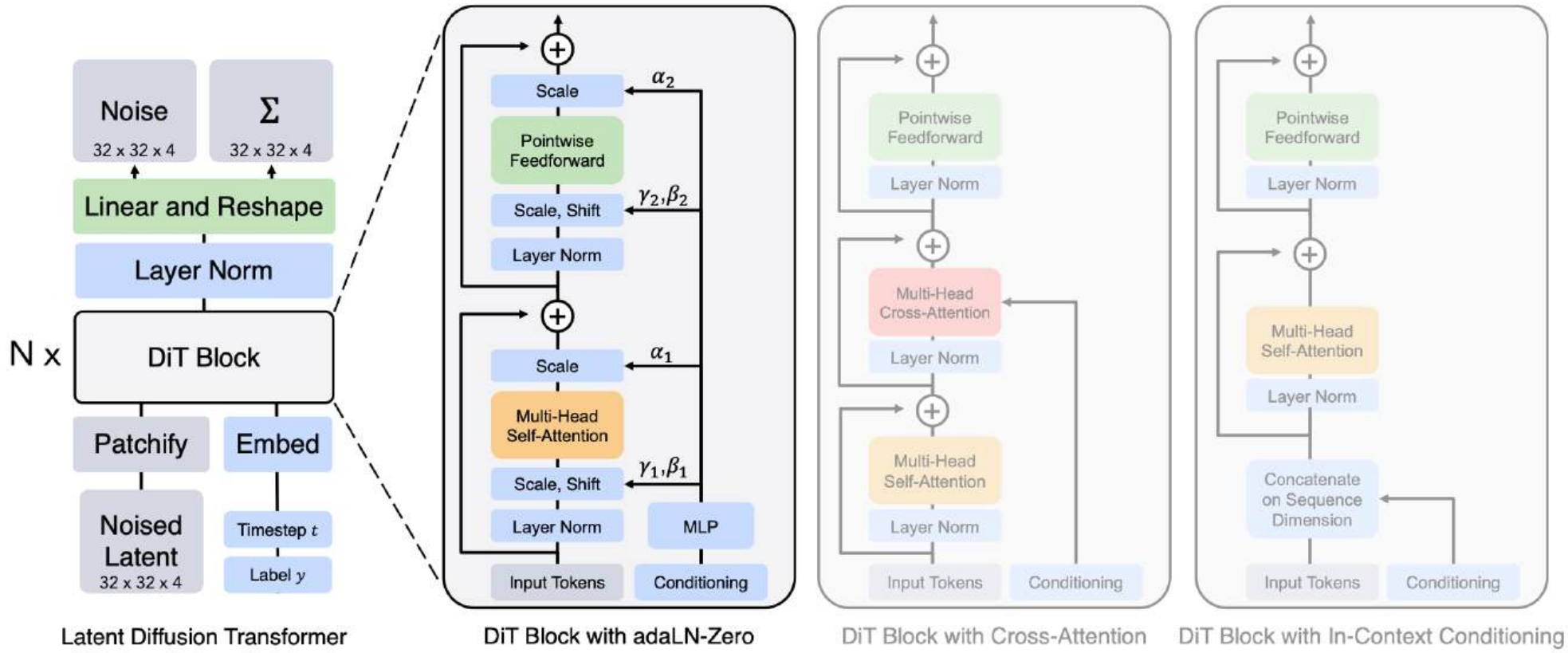


Figure 3. **The Diffusion Transformer (DiT) architecture.** *Left:* We train conditional latent DiT models. The input latent is decomposed into patches and processed by several DiT blocks. *Right:* Details of our DiT blocks. We experiment with variants of standard transformer blocks that incorporate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

Peebles et al. “Scalable Diffusion Models with Transformers ”, 2023

Diffusion Transformer

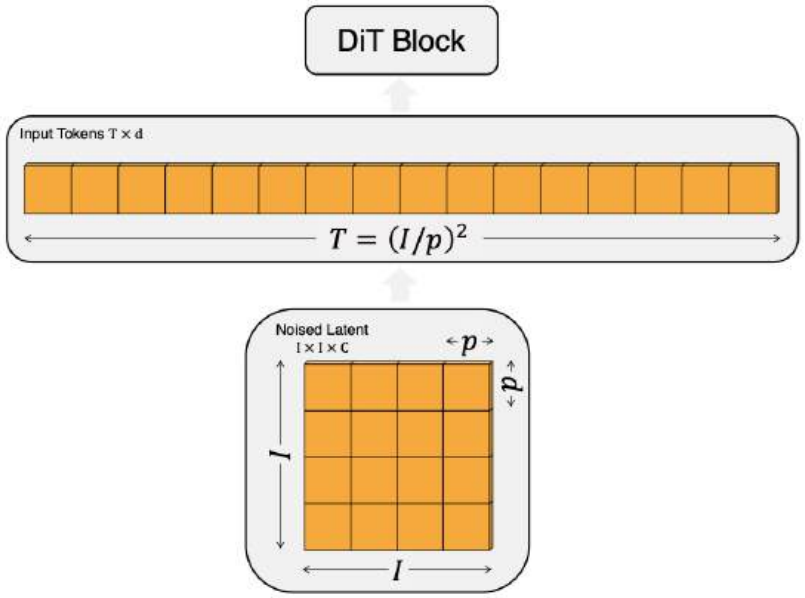
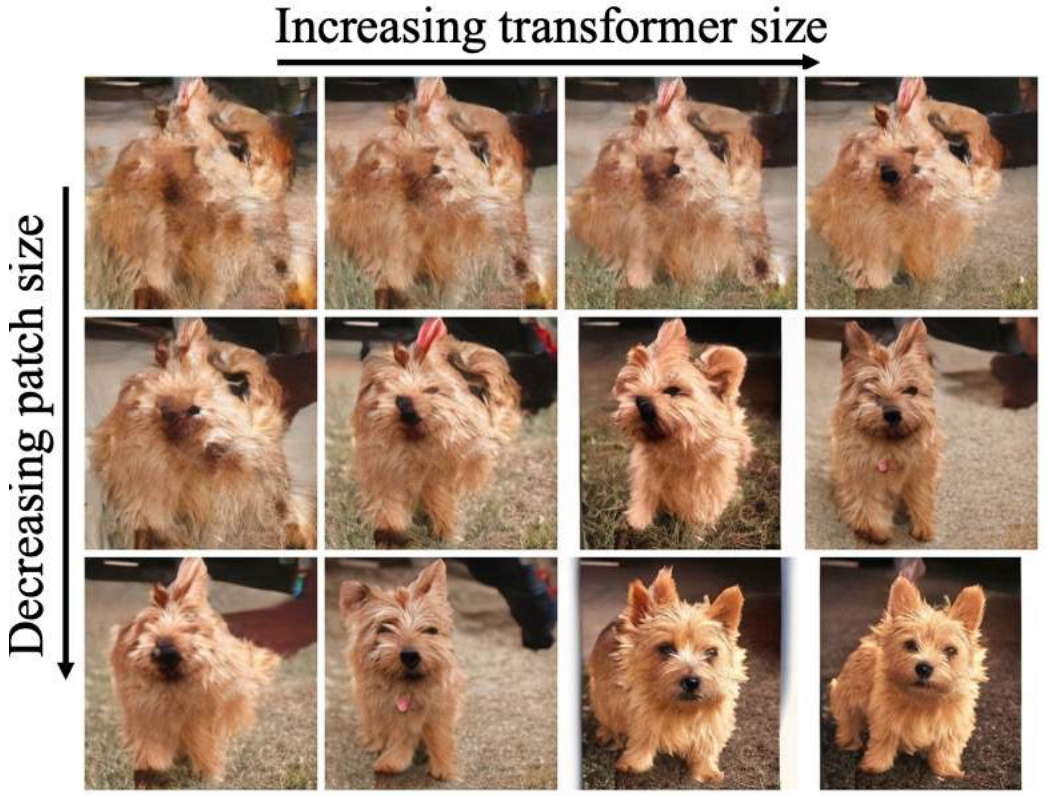


Figure 4. **Input specifications for DiT.** Given patch size $p \times p$, a spatial representation (the noised latent from the VAE) of shape $I \times I \times C$ is "patchified" into a sequence of length $T = (I/p)^2$ with hidden dimension d. A smaller patch size p results in a longer sequence length and thus more Gflops.

p=1, it is more like PixcelCNN.



Scaling law of DiT: More compute-intensive DiT models have significantly-higher sample quality.

SORA

Base Compute

4 × compute

32 × compute

Scaling law in SORA: Sample quality improves markedly as training compute increases.

Limitations of SORA

- It does not accurately model the physics of many basic interactions.

Limitations of SORA

- Incoherencies in long video samples.

DALL-E 2: Text-to-image Generation

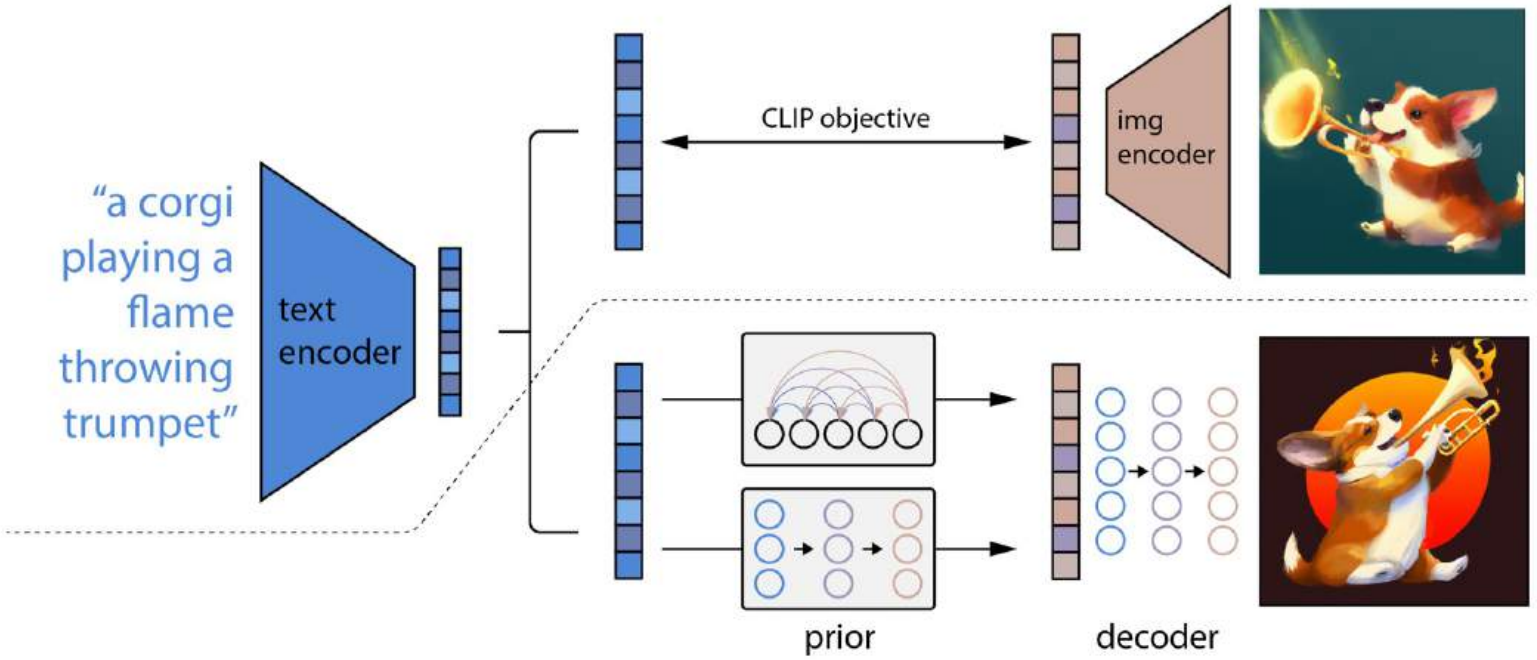
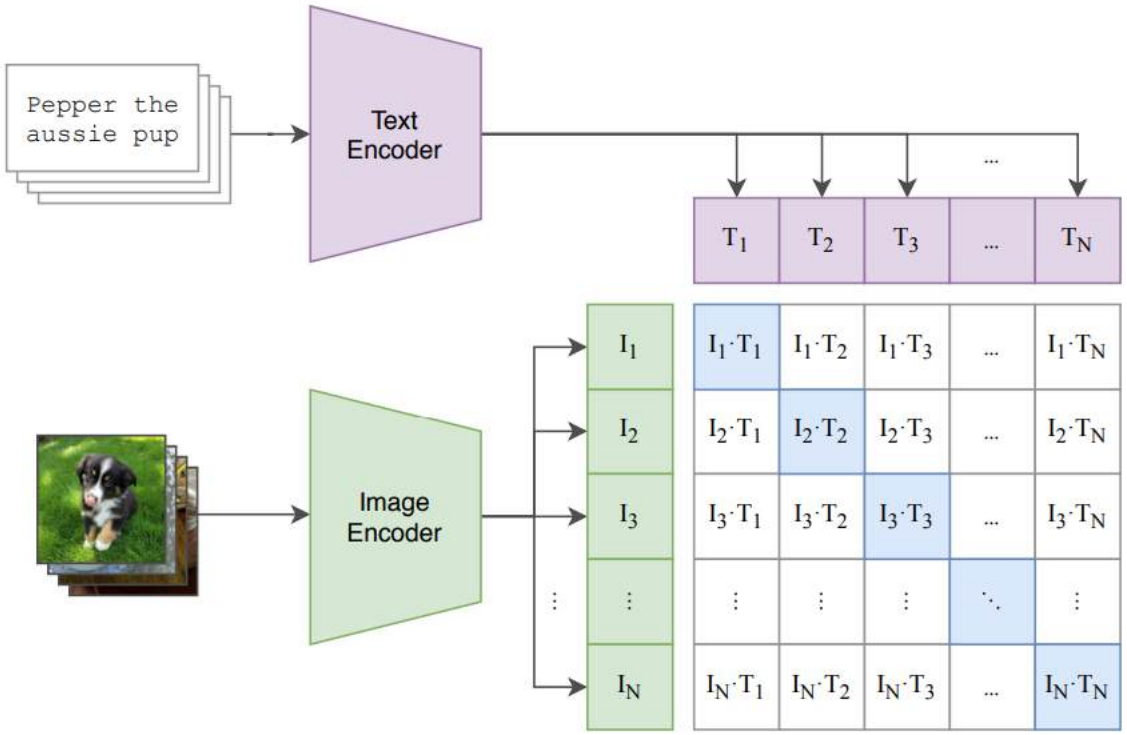


Figure 2: A high-level overview of unCLIP. Above the dotted line, we depict the CLIP training process, through which we learn a joint representation space for text and images. Below the dotted line, we depict our text-to-image generation process: a CLIP text embedding is first fed to an autoregressive or diffusion prior to produce an image embedding, and then this embedding is used to condition a diffusion decoder which produces a final image. Note that the CLIP model is frozen during training of the prior and decoder.

Ramesh et al. "Hierarchical Text-Conditional Image Generation with CLIP Latents", 2022

DALL-E 2: Text-to-image Generation

- CLIP (Contrastive Language–Image Pre-training)



Contrastive learning:

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-))} \right]$$

score for the positive pair

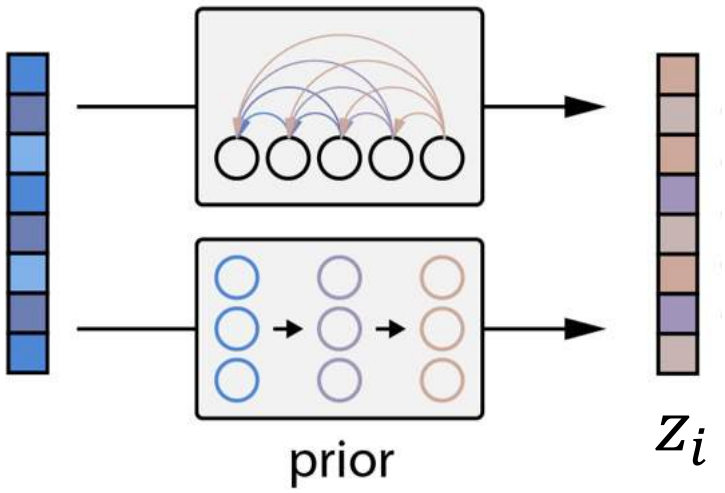
score for the N-1 negative pairs

positive pairs: diagonal elements
negative pairs: others

Radford, et al. "Learning transferable visual models from natural language supervision." 2021
Ramesh et al. "Hierarchical Text-Conditional Image Generation with CLIP Latents", 2022

DALL-E 2: Text-to-image Generation

- Prior: learn $p(z_i|y)$ y: $caption$, z_i: $image\ embedding$



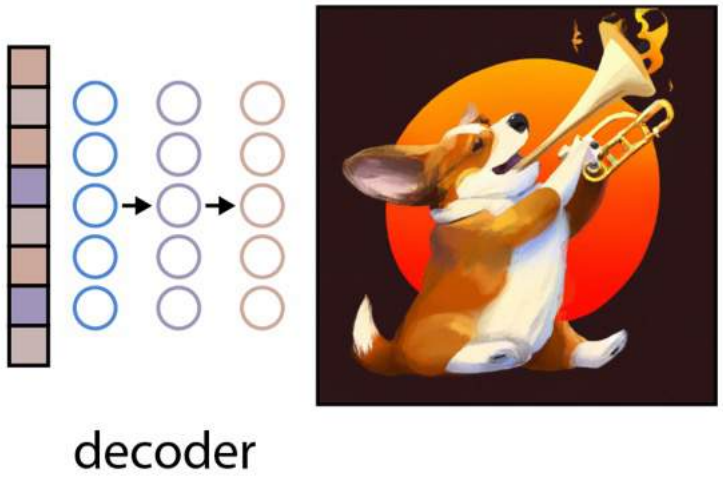
- Autoregressive (AR) prior: quantize z_i and predict it autoregressively.
- Diffusion prior: continuous diffusion generative model conditioned on y.

Ramesh et al. “Hierarchical Text-Conditional Image Generation with CLIP Latents”, 2022

DALL-E 2: Text-to-image Generation

- Image generation:

$$P(x|y) = P(x, z_i|y) = P(x|z_i, y)P(z_i|y).$$



Learn $p(x|z_i, y)$

Cascaded diffusion models: 1 base model (64x64), 2 super-resolution models (64x64 → 256x256, 256x256 → 1024x1024).

Ramesh et al. “Hierarchical Text-Conditional Image Generation with CLIP Latents”, 2022

DALL-E 2: Text-to-image Generation

Figure 4: Variations between two images by interpolating their CLIP image embedding and then decoding with a diffusion model. We fix the decoder seed across each row. The intermediate variations naturally blend the content and style from both input images.

interpolate the conditioner; different latent codes produces different trajectories.

Ramesh et al. “Hierarchical Text-Conditional Image Generation with CLIP Latents”, 2022

DALL-E 2: Text-to-image Generation

a photo of an adult lion → a photo of lion cub

a photo of a landscape in winter → a photo of a landscape in fall

Figure 5: Text diffs applied to images by interpolating between their CLIP image embeddings and a normalised difference of the CLIP text embeddings produced from the two descriptions. We also perform DDIM inversion to perfectly reconstruct the input image in the first column, and fix the decoder DDIM noise across each row.

Ramesh et al. “Hierarchical Text-Conditional Image Generation with CLIP Latents”, 2022

DALL-E 3: Trained on Synthetic Captions

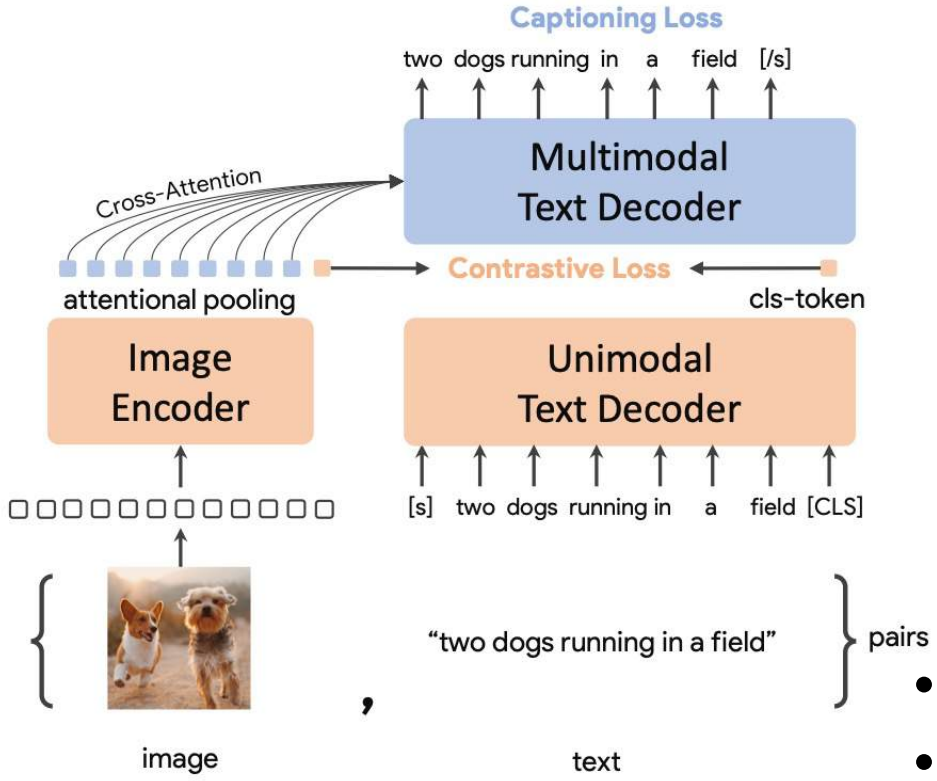
text-to-image generation

An astronaut riding a horse in photorealistic style

Image captioning

DALL-E 3: Trained on Synthetic Captions

CoCa Captioner



$$\mathcal{L}_{\text{Con}} = -\frac{1}{N}\Big(\underbrace{\sum_{i}^{N}\log\frac{\exp(x_i^\top y_i/\sigma)}{\sum_{j=1}^{N}\exp(x_i^\top y_j/\sigma)}}_{\text{image-to-text}} + \underbrace{\sum_{i}^{N}\log\frac{\exp(y_i^\top x_i/\sigma)}{\sum_{j=1}^{N}\exp(y_i^\top x_j/\sigma)}}_{\text{text-to-image}}\Big),$$

$$\mathcal{L}_{\text{Cap}} = -\sum_{t=1}^{T}\log P_\theta(y_t|y_{<t}, x).$$

- Fine-tune CoCa on descriptive synthetic captions
- Apply the new captioner to text-to-image datasets

Xu et al. "CoCa: Contrastive Captioners are Image-Text Foundation Models ", 2022

DALL-E 3: Trained on Synthetic Captions

Image

short synthetic captions :

a white modern bathtub sits on a wooden floor.

descriptive synthetic captions :

this luxurious bathroom features a mod- ern freestanding bathtub in a crisp white finish. the tub sits against a wooden ac- cent wall with glass-like panels, creating a serene and relaxing ambiance. three pendant light fixtures hang above the tub, adding a touch of sophistication. a large window with a wooden panel provides natural light, while a potted plant adds a touch of greenery. the freestanding bath- tub stands out as a statement piece in this contemporary bathroom.

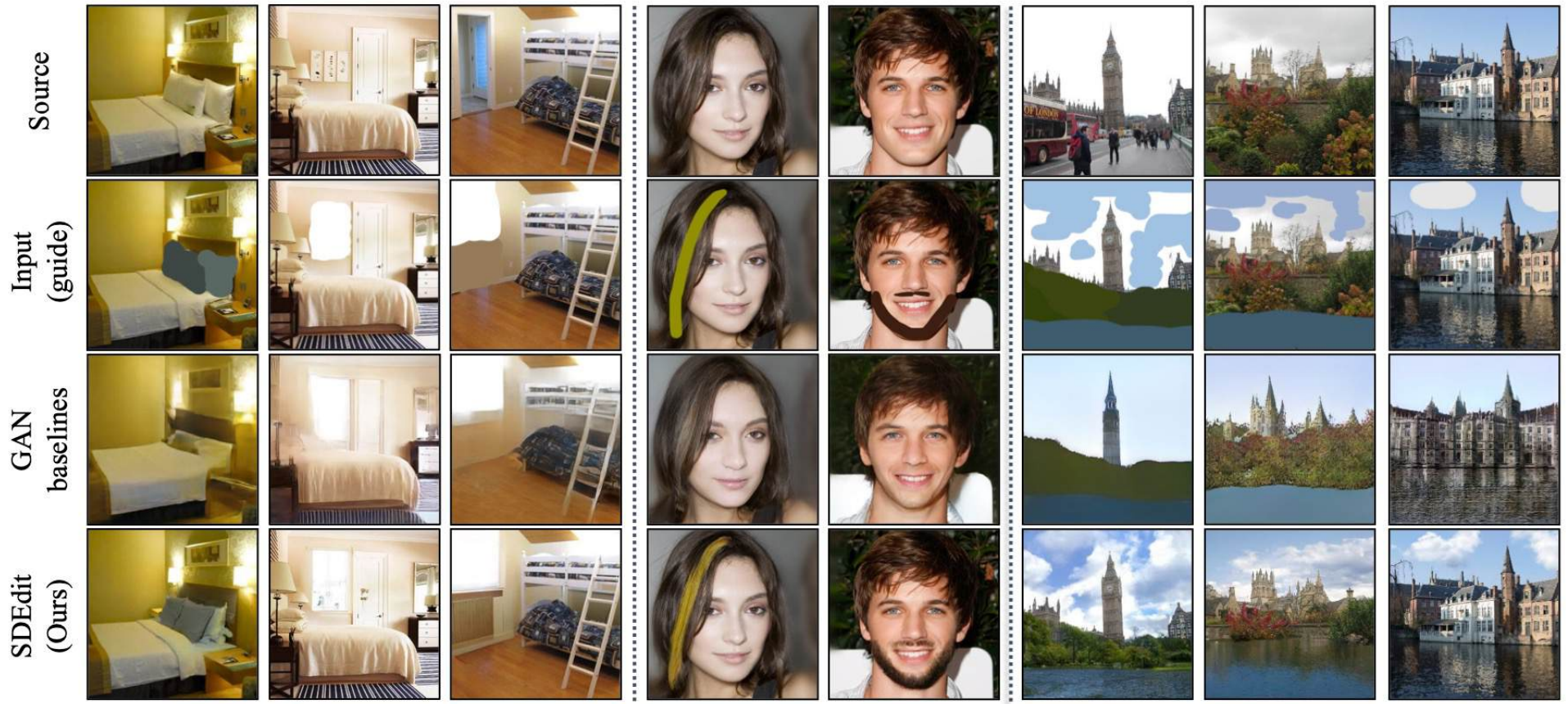
"Improving Image Generation with Better Captions", OpenAI

DALL-E 3: Trained on Synthetic Captions

DALL·E 2 · An expressive oil painting of a chocolate chip cookie being dipped in a glass of milk, depicted as an explosion of flavors.

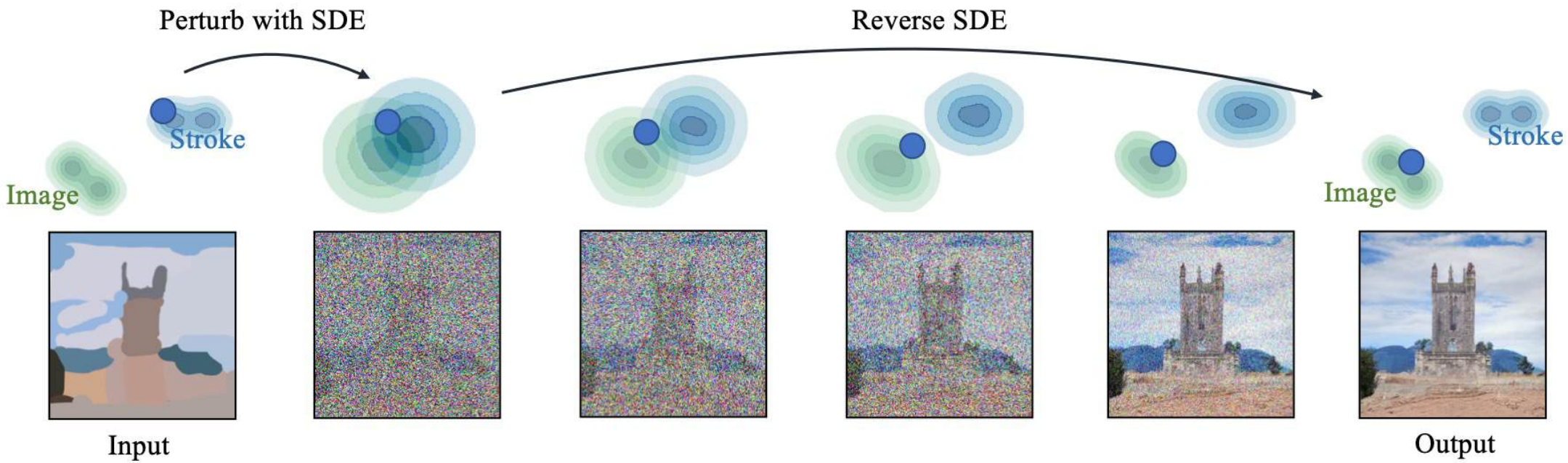
DALL·E 3 · An expressive oil painting of a chocolate chip cookie being dipped in a glass of milk, depicted as an explosion of flavors.

SDEdit: Image Editing



Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”, 2021

SDEdit: Image Editing



guide $\xrightarrow{\text{forward SDE}}$ latent code at t_0 $\xrightarrow{\text{reverse SDE}}$ edited image

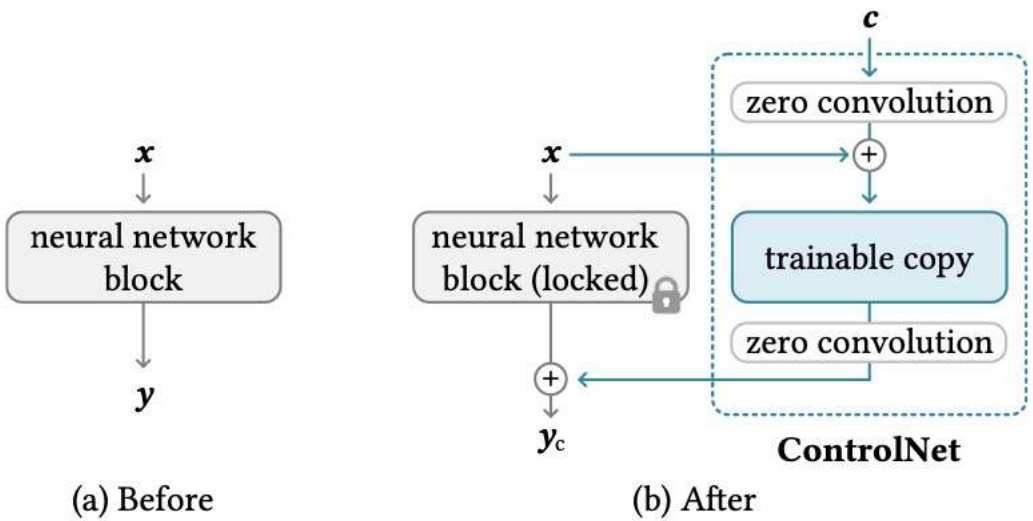
Meng et al. "SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations", 2021

SDEdit: Image Editing

Change the setting to the 1920s with an old school car (Video source: SORA)

Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”, 2021

ControlNet



(a) Before (b) After

Figure 2: A neural block takes a feature map x as input and outputs another feature map y, as shown in (a). To add a ControlNet to such a block we lock the original block and create a trainable copy and connect them together using zero convolution layers, *i.e.*, 1×1 convolution with both weight and bias initialized to zero. Here c is a conditioning vector that we wish to add to the network, as shown in (b).

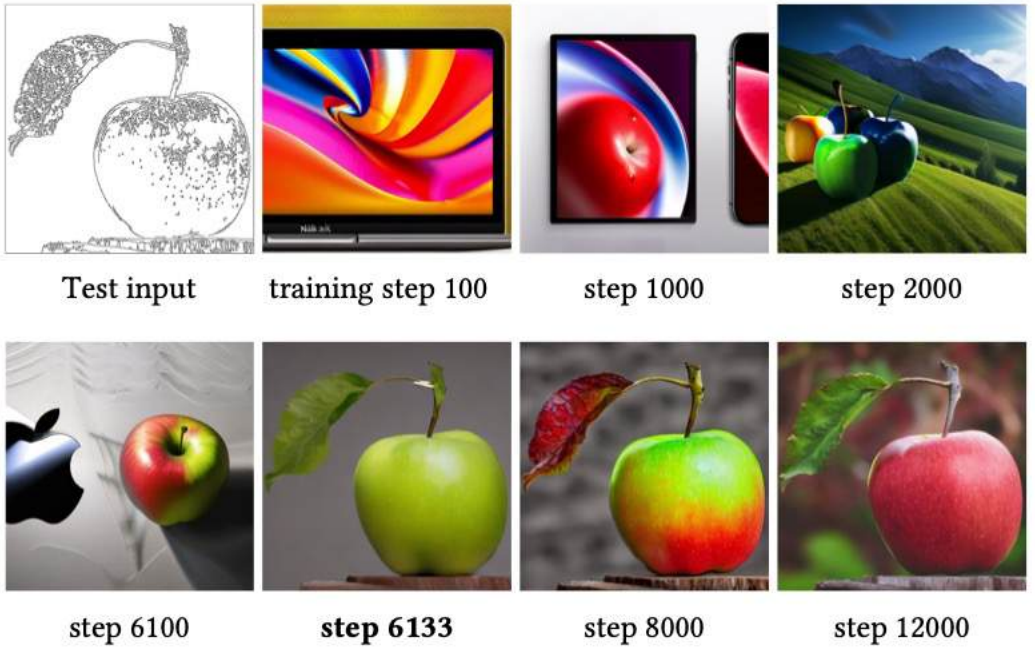


Figure 4: The sudden convergence phenomenon. Due to the zero convolutions, ControlNet always predicts high-quality images during the entire training. At a certain step in the training process (*e.g.*, the 6133 steps marked in bold), the model suddenly learns to follow the input condition.

Zhang, "Adding Conditional Control to Text-to-Image Diffusion Models", 2023

ControlNet

Zhang, "Adding Conditional Control to Text-to-Image Diffusion Models", 2023

Stable Diffusion3

- Flow Matching Framework:

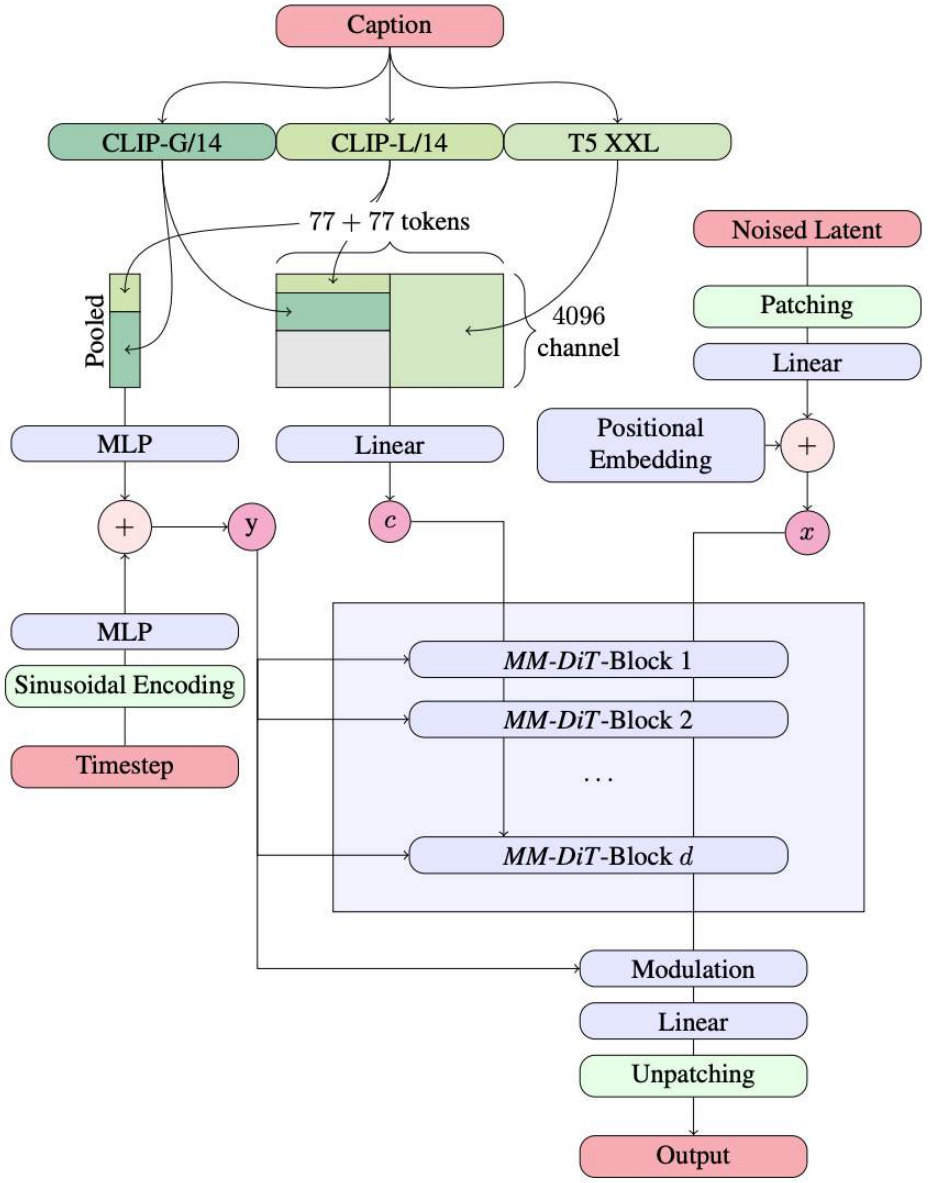
$$z_t = a_t x_0 + b_t \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, I) \ .$$

$$\mathcal{L}_{CFM} = \mathbb{E}_{t, p_t(z|\epsilon), p(\epsilon)} ||v_\Theta(z, t) - u_t(z|\epsilon)||_2^2$$ (In practice, reweighting is used)

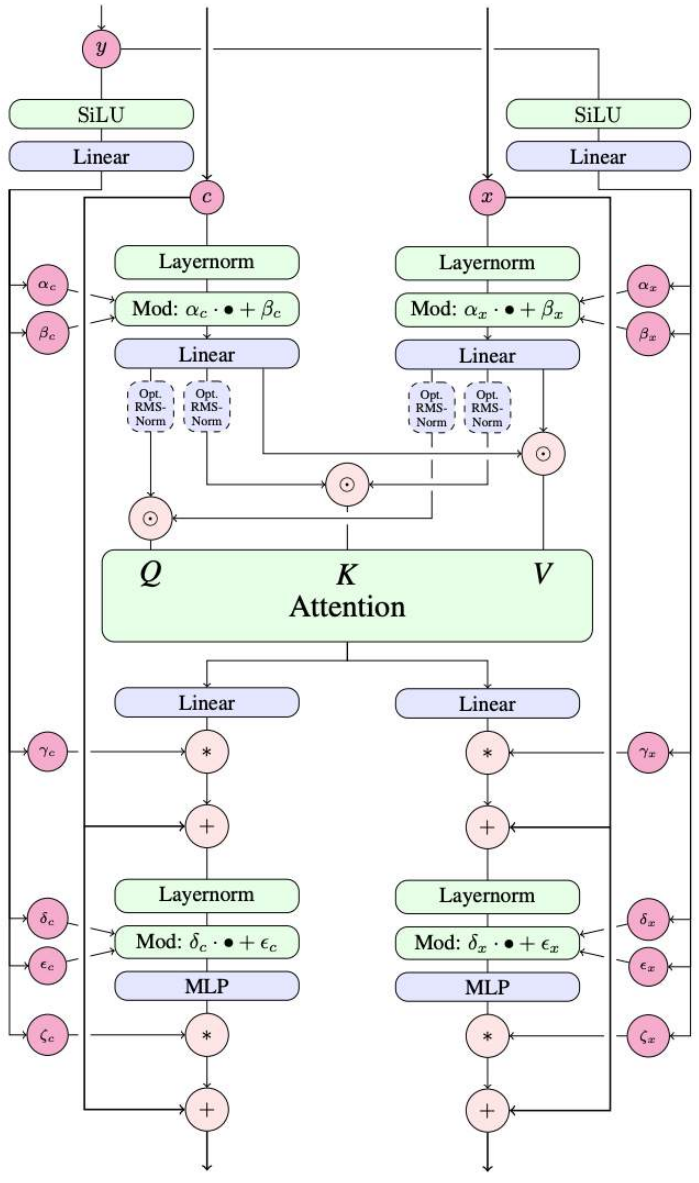
$$\psi_t(\cdot|\epsilon) : x_0 \mapsto a_t x_0 + b_t \epsilon$$
$$u_t(z|\epsilon) \coloneqq \psi_t'(\psi_t^{-1}(z|\epsilon)|\epsilon) \quad \Rightarrow \quad z_t' = u_t(z_t|\epsilon) = \frac{a_t'}{a_t} z_t - \epsilon b_t (\frac{a_t'}{a_t} - \frac{b_t'}{b_t}) \ .$$

- Flow Trajectories
  - Rectified flow $z_t = (1 - t)x_0 + t\epsilon \ ,$
  - EDM $z_t = x_0 + b_t \epsilon \qquad b_t = \exp F_{\mathcal{N}}^{-1}(t|P_m, P_s^2)$
  - Cosine $z_t = \cos(\frac{\pi}{2} t) x_0 + \sin(\frac{\pi}{2} t) \epsilon \ .$

Scaling Rectified Flow Transformers for High-Resolution Image Synthesis, 2024



(a) Overview of all components.



(b) One *MM-DiT* block

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

variant	rank averaged over all	5 steps	50 steps
rf/lognorm(0.00, 1.00)	1.54	1.25	1.50
rf/lognorm(1.00, 0.60)	2.08	3.50	2.00
rf/lognorm(0.50, 0.60)	2.71	8.50	1.00
rf/mode(1.29)	2.75	3.25	3.00
rf/lognorm(0.50, 1.00)	2.83	1.50	2.50
eps/linear	2.88	4.25	2.75
rf/mode(1.75)	3.33	2.75	2.75
rf/cosmap	4.13	3.75	4.00
edm(0.00, 0.60)	5.63	13.25	3.25
rf	5.67	6.50	5.75
v/linear	6.83	5.75	7.75
edm(0.60, 1.20)	9.00	13.00	9.00
v/cos	9.17	12.25	8.75
edm/cos	11.04	14.25	11.25
edm/rf	13.04	15.25	13.25
edm(-1.20, 1.20)	15.58	20.25	15.00

Table 1. **Global ranking of variants.** For this ranking, we apply non-dominated sorting averaged over EMA and non-EMA weights, two datasets and different sampling settings.

Logit-Normal Sampling : $\text{logit}(t) = \log \frac{t}{1-t}$

$$\pi_{\text{ln}}(t; m, s) = \frac{1}{s\sqrt{2\pi}} \frac{1}{t(1-t)} \exp\left(-\frac{(\text{logit}(t) - m)^2}{2s^2}\right)$$

variant	ImageNet CLIP	ImageNet FID	CC12M CLIP	CC12M FID
rf	0.247	49.70	0.217	94.90
edm(-1.20, 1.20)	0.236	63.12	0.200	116.60
eps/linear	0.245	48.42	0.222	*90.34*
v/cos	0.244	50.74	0.209	97.87
v/linear	0.246	51.68	0.217	100.76
rf/lognorm(0.50, 0.60)	**0.256**	80.41	0.233	120.84
rf/mode(1.75)	*0.253*	**44.39**	0.218	94.06
rf/lognorm(1.00, 0.60)	0.254	114.26	**0.234**	147.69
rf/lognorm(-0.50, 1.00)	0.248	45.64	0.219	**89.70**
rf/lognorm(0.00, 1.00)	0.250	*45.78*	*0.224*	89.91

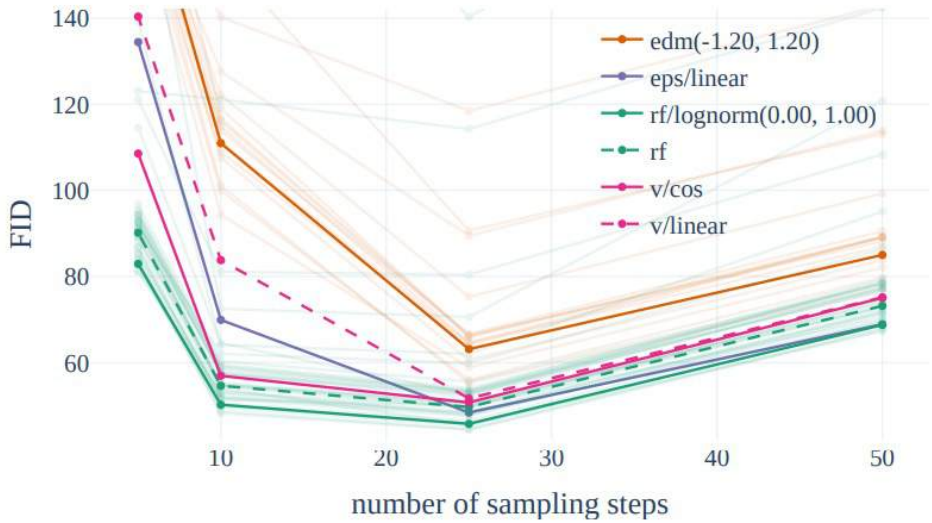
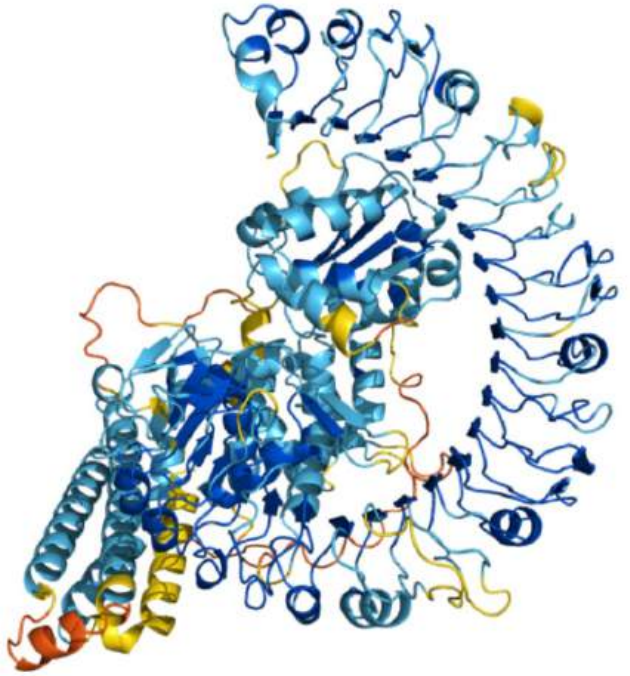


Figure 3. **Rectified flows are sample efficient.** Rectified Flows perform better then other formulations when sampling fewer steps. For 25 and more steps, only `rf/lognorm(0.00, 1.00)` remains competitive to `eps/linear`.

AlphaFold

- Protein folding
  - Proteins are made up of amino acids that fold into specific 3D shapes.
  - The shape of a protein determines its function.
  - Predicting protein structures is computationally complex and has been a longstanding challenge in biology.

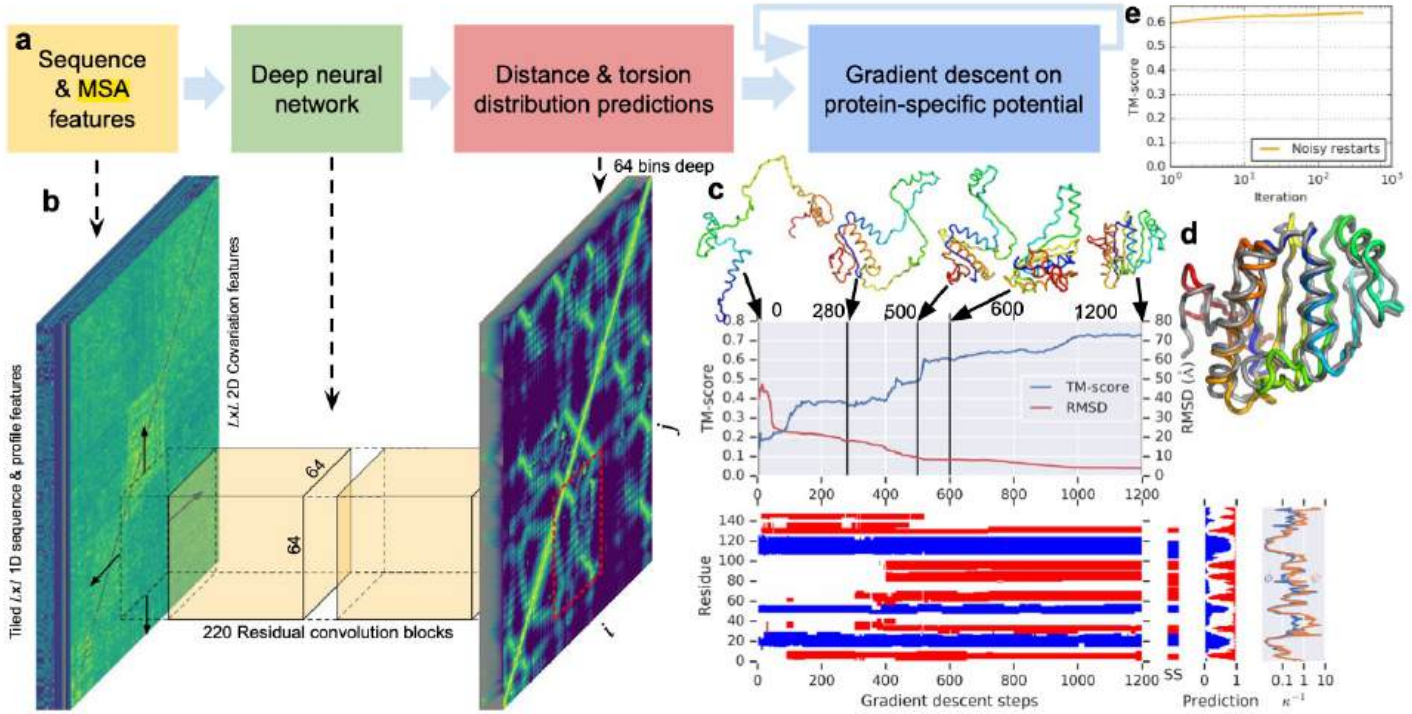


Q8W3K0: A potential plant disease resistance protein. Mean pLDDT 82.24.

Senior, A.W. et al. (2020). “Improved protein structure prediction using potentials from deep learning.” Nature.

AlphaFold

- Given: amino acid sequence; output: probability of distance and torsion (ϕ, ψ) between adjacent amino acids.



Multiple Sequence Alignment (MSA): aligning the sequence of interest with a set of related sequences

Probability of (ϕ, ψ): probabilities on discrete bins.

Senior, A.W. et al. (2020). "Improved protein structure prediction using potentials from deep learning." Nature.

Alphafold

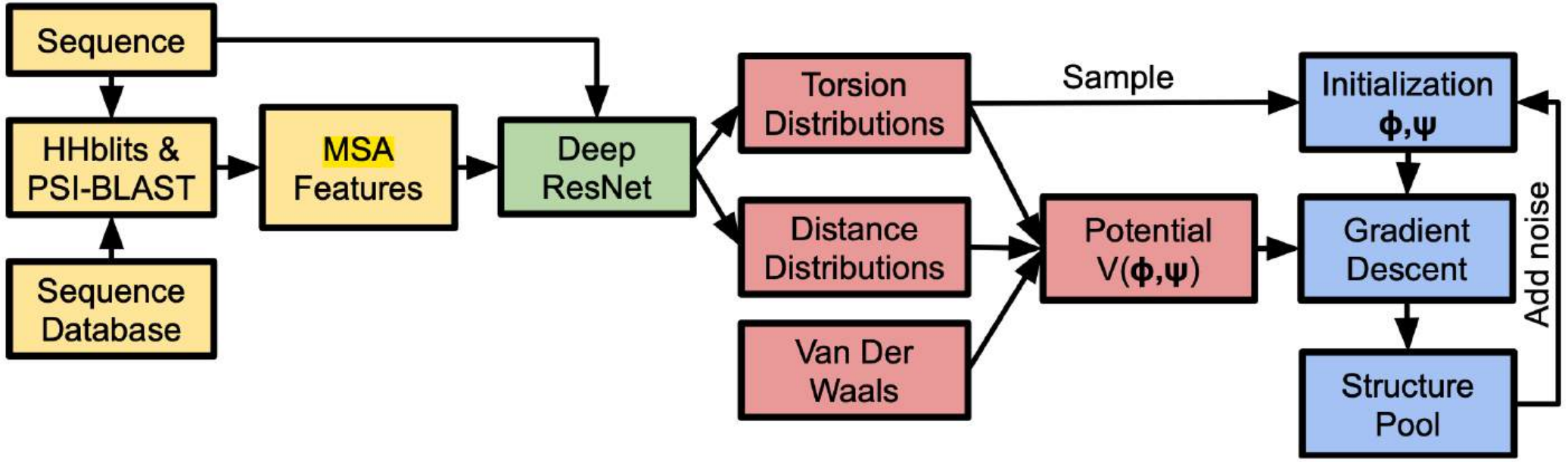
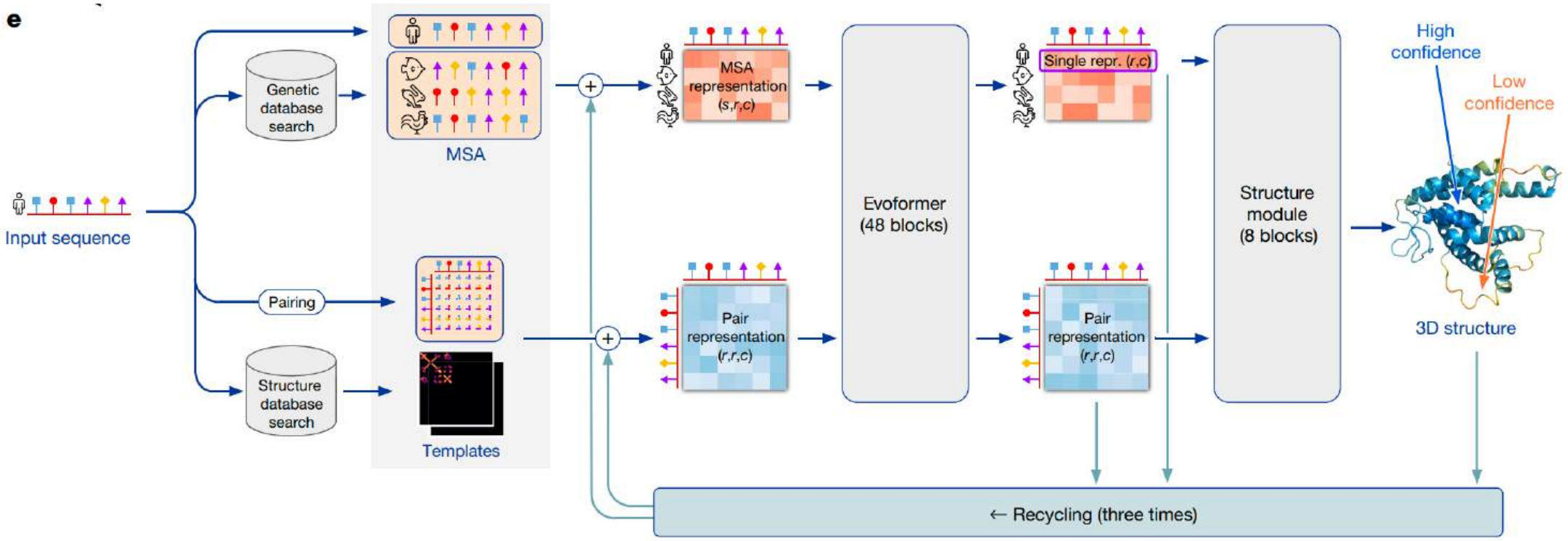


Fig. 5 | A schematic of the folding system. Feature extraction stages are shown in yellow, structure-prediction neural network in green, potential construction in red and structure realisation in blue.

Senior, A.W. et al. (2020). "Improved protein structure prediction using potentials from deep learning." Nature.

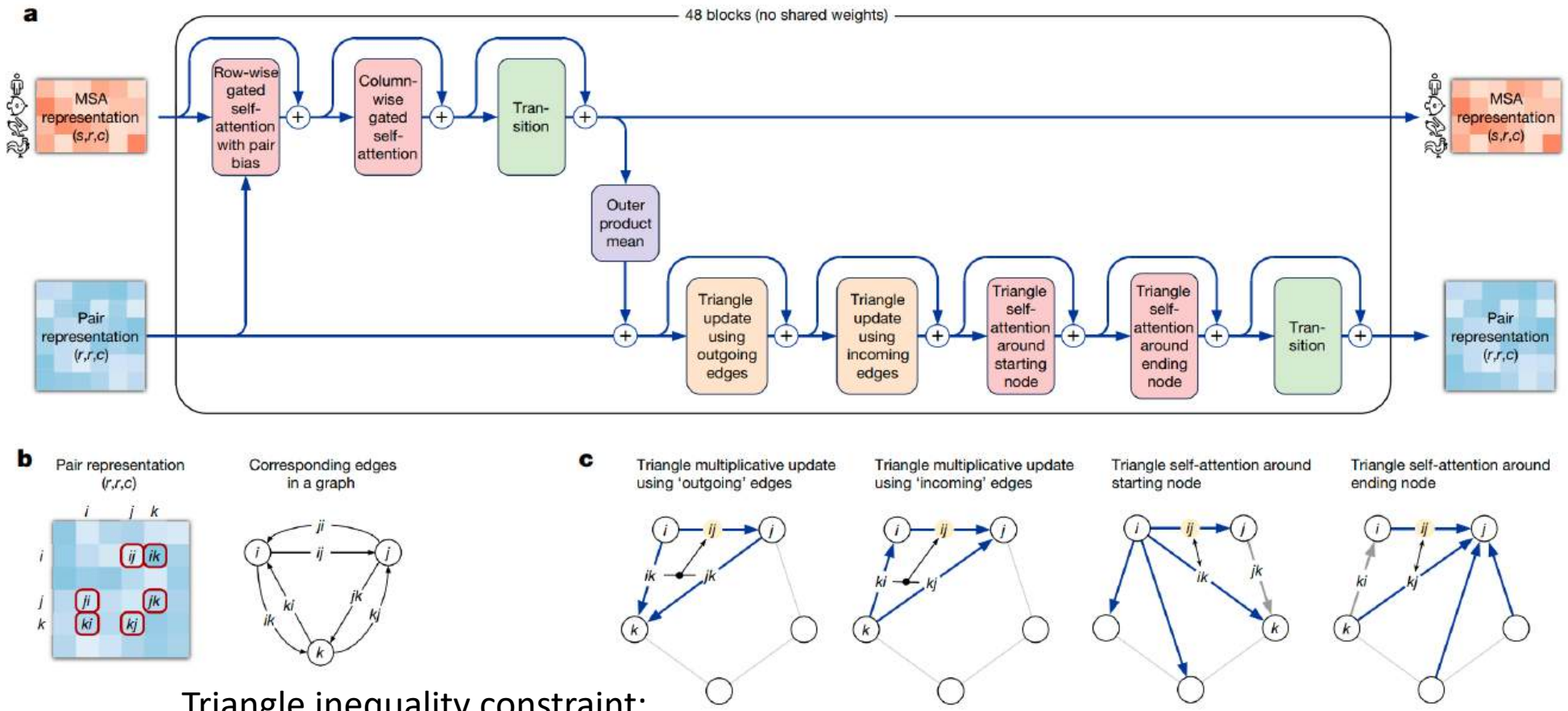
Alphafold2



An end-to-end protein structure prediction model

Jumper, J. et al. (2021). “Highly accurate protein structure prediction with AlphaFold.” Nature.

Alphafold2



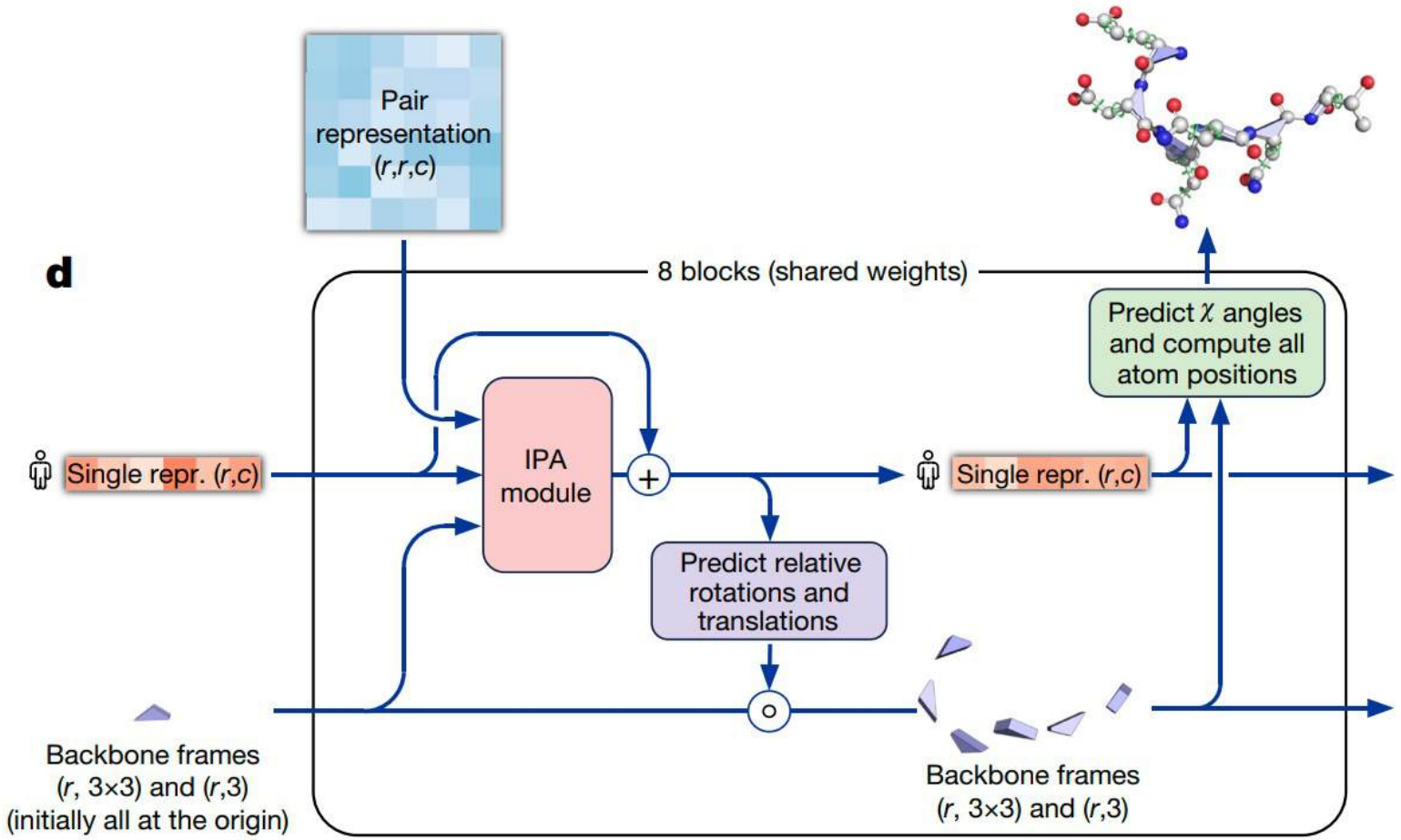
Triangle inequality constraint:

$$d(a,b) + d(b,c) \geq d(a,c)$$

Triangular multiplicative update

Jumper, J. et al. (2021). "Highly accurate protein structure prediction with AlphaFold." Nature.

Alphafold2



Jumper, J. et al. (2021). “Highly accurate protein structure prediction with AlphaFold.” Nature.

Alphafold3

- No computation based on torsion, distance …
- It directly generates the 3D structure of proteins like generating images!