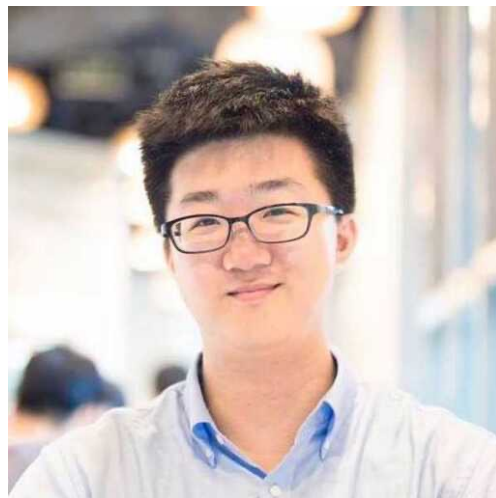
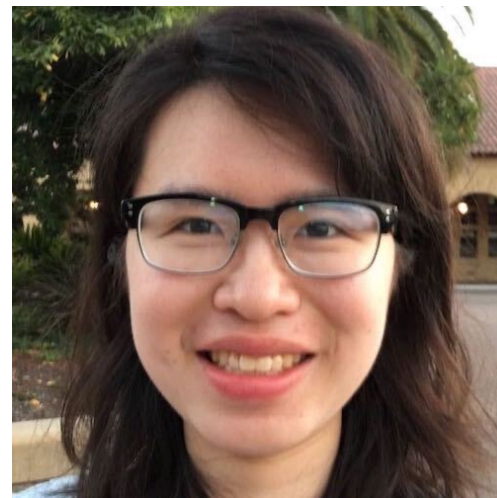


# Denoising Diffusion Models: A Generative Learning Big Bang

Jiaming Song



Chenlin Meng



Arash Vahdat



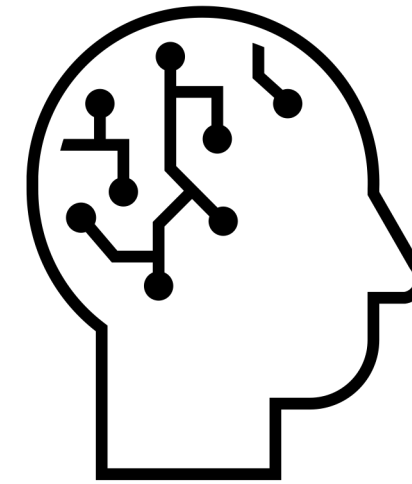
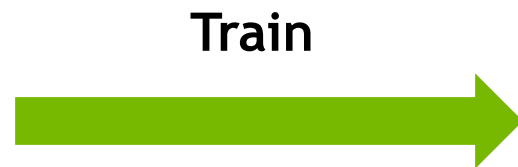
# Deep Generative Learning

Learning to generate data

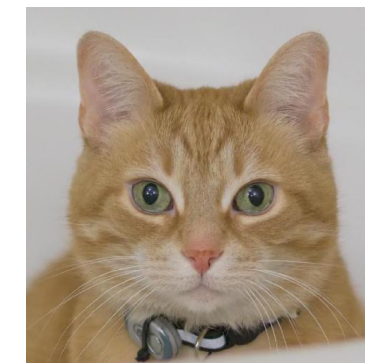
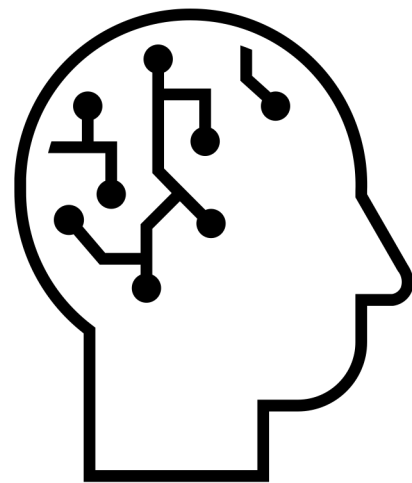
*Hope to learn true distribution*



Samples from a Data Distribution



Neural Network



# Generative Learning Applications

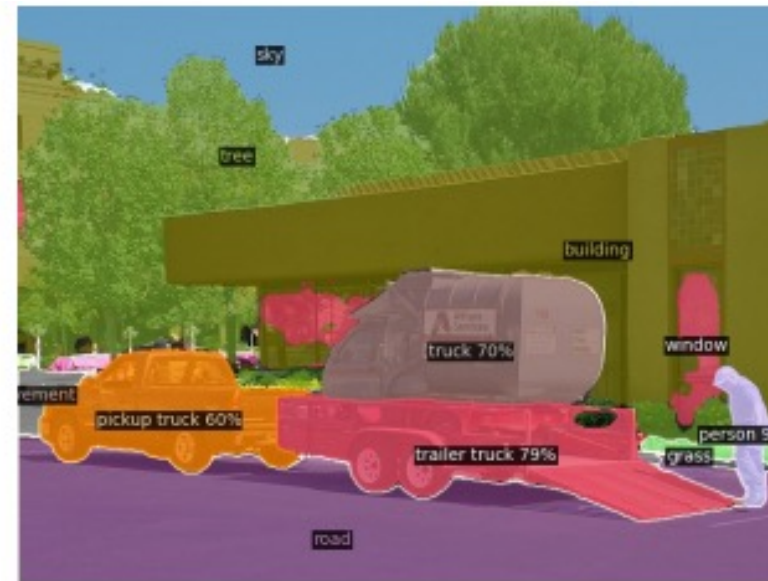
Art & Design



Content Generation



Representation Learning



Entertainment



# The Landscape of Deep Generative Learning

Bayesian Networks

Restricted Boltzmann Machines

Variational Autoencoders

Normalizing Flows

Energy-based Models

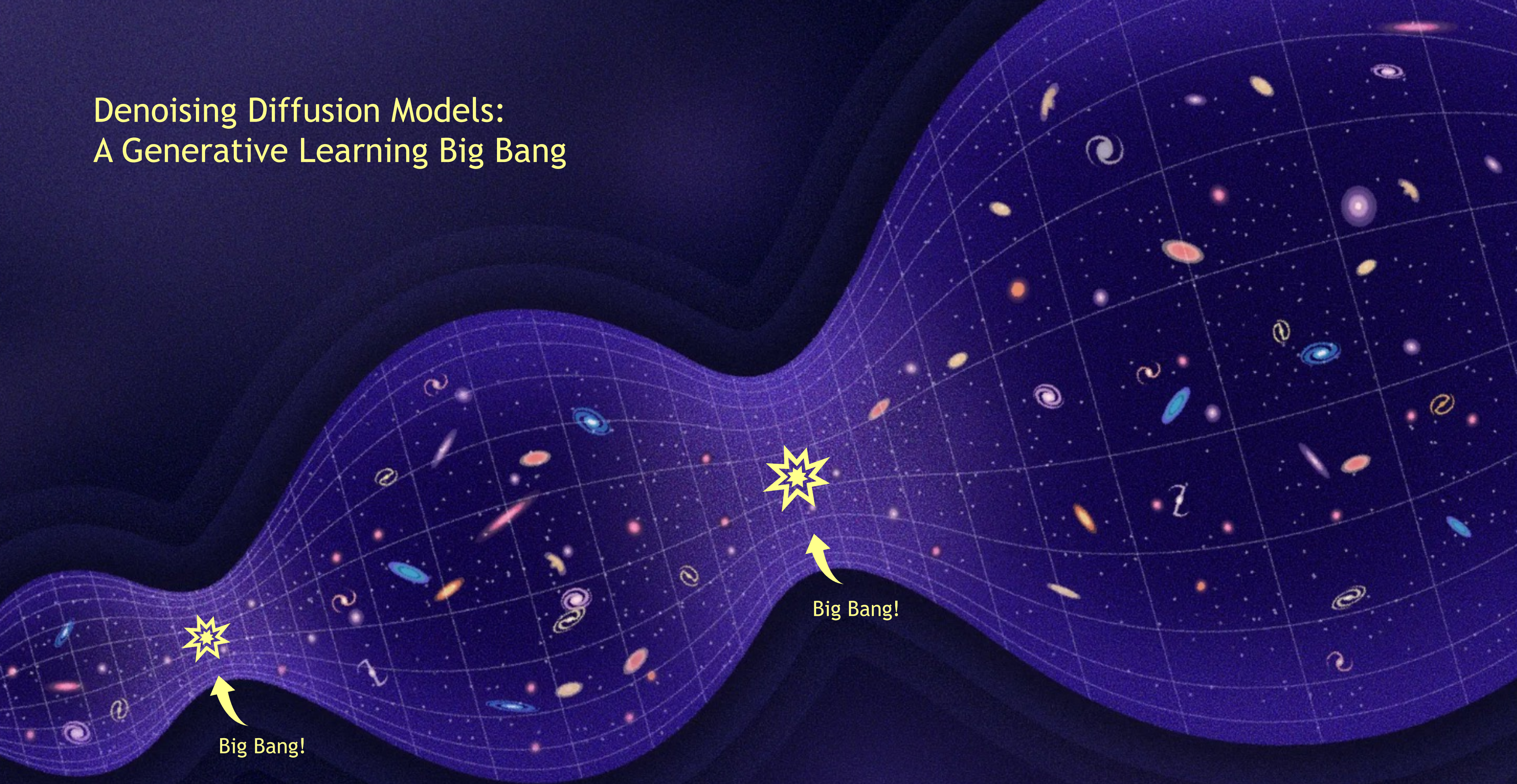
Generative Adversarial Networks

Autoregressive Models

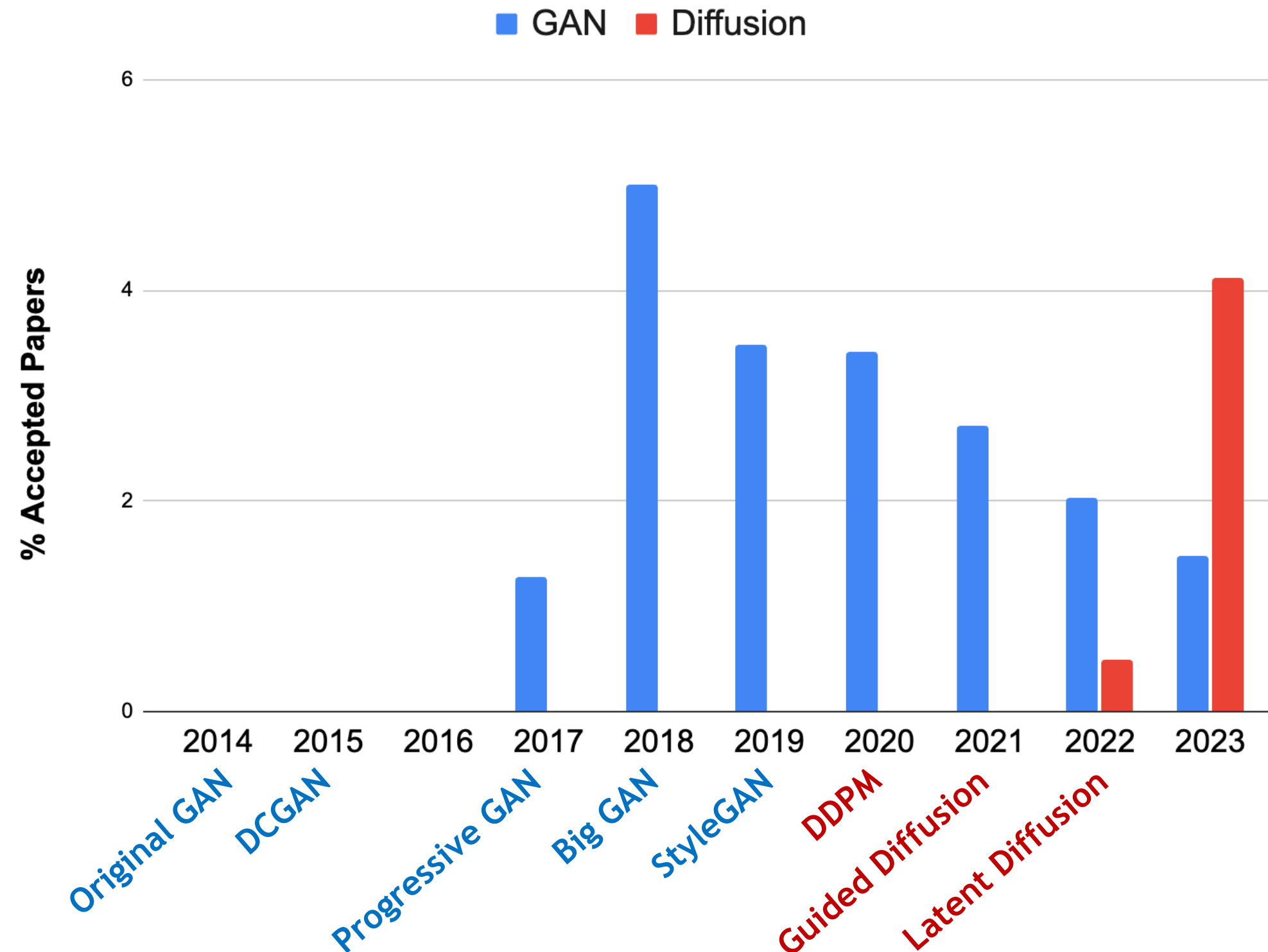
Denoising Diffusion Models



# Denoising Diffusion Models: A Generative Learning Big Bang



# We May Not Know Cosmology, But We Know CVPR



# Today's Program

| Title   | Speaker | Time        |
|---|---------|-------------|
| Fundamentals: Training, Sampling, Acceleration, Guidance                                    | Arash   | 09:00-10:00 |
| Break   |         |             |
| Image Applications: Architecture, Editing, Personalization, Fine-tuning, "Low-level" Vision | Chenlin | 10:15-11:15 |
| Break   |         |             |
| Other domains: Inverse problems, Video, 3D, Large Content Generation                        | Jiaming | 11:30-12:30 |

<https://cvpr2023-tutorial-diffusion-models.github.io/>

# Disclaimer



# Acknowledgment

Some Slides Adapted From

## CVPR 2022 Tutorial:

Denoising Diffusion-based Generative Modeling:  
Foundations and Applications

**Dates:** Sunday June 19, 8:30 am - 12:10 pm (CDT)



Karsten Kreis

Ruiqi Gao

Arash Vahdat

<https://cvpr2022-tutorial-diffusion-models.github.io/>

# Today's Program

| Title   | Speaker | Time        |
|---|---------|-------------|
| Fundamentals: Training, Sampling, Acceleration, Guidance                                    | Arash   | 09:00-10:00 |
| Break   |         |             |
| Image Applications: Architecture, Editing, Personalization, Fine-tuning, "Low-level" Vision | Chenlin | 10:15-11:15 |
| Break   |         |             |
| Other domains: Inverse problems, Video, 3D, Large Content Generation                        | Jiaming | 11:30-12:30 |

# Outline

- Part (1): Denoising Diffusion Probabilistic Models *DDPM*
- Part (2): Score-based Generative Modeling with Differential Equations *SDE*
- Part (3): Accelerated Sampling
- Part (4): Conditional Generation and Guidance

# Outline

**Part (1): Denoising Diffusion Probabilistic Models**

Part (2): Score-based Generative Modeling with Differential Equations

Part (3): Accelerated Sampling

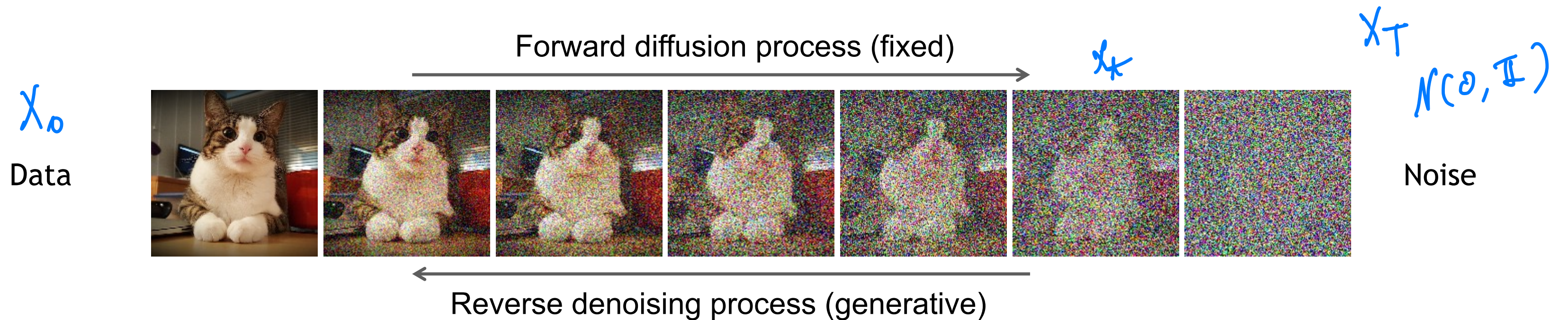
Part (4): Conditional Generation and Guidance

# Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

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



[Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015](#)

[Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020](#)

[Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021](#)

$$x_t \sim \mathcal{N}(x_{t+1}, \mathbf{I})$$

$$q(x_t) = \int q(x_t | x_{t+1}) q(x_{t+1}) dx_{t+1}$$

$$= \int \mathcal{N}(x_t; x_{t+1}, \mathbf{I}) q(x_{t+1}) dx_{t+1}$$

$$= \int \mathcal{N}(x_t - x_{t+1}; 0, \mathbf{I}) q(x_{t+1}) dx_{t+1} = \mathcal{N}(0, \mathbf{I}) * q(x_{t+1}) \Rightarrow x_t = \mathcal{N}(0, \mathbf{I}) + \underline{x_{t+1}}$$

$$p(x_1, x_2) = p(x_2) p(x_1 | x_2)$$

# Forward Diffusion Process

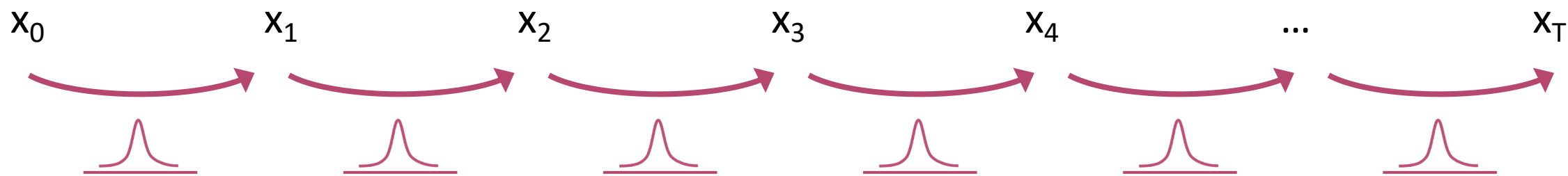
The formal definition of the forward process in T steps:

Forward diffusion process (fixed) →

Data



Noise



$$q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t \mathbf{I}) \quad \rightarrow \quad q(x_{1:T} | x_0) = \prod_{t=1}^T q(x_t | x_{t-1}) \quad \text{(joint)}$$

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

# Diffusion Kernel

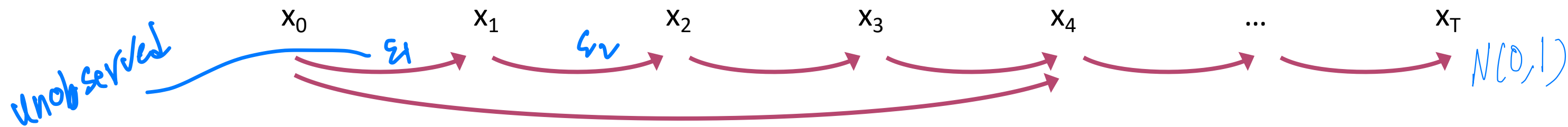
$$\begin{aligned}
 q(x_2|x_0) &= \int q(x_2|x_1, \beta) q(x_1|x_0) dx_1 \\
 &= \int \underbrace{q(x_2|x_1)}_{\text{Norm}_{x_1, C}} \underbrace{q(x_1|x_0)}_{\text{Norm}_{x_0, C}} dx_1 \\
 &= \mathbb{R} \int \underbrace{\text{Norm}_{x_1, C}}_{\text{Norm}_{x_2}(\bar{\mu}, \bar{\sigma}^2)} dx_1 \\
 &= \text{Norm}_{x_2}(\bar{\mu}, \bar{\sigma}^2)
 \end{aligned}$$

Forward diffusion process (fixed)

Data



Noise



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

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

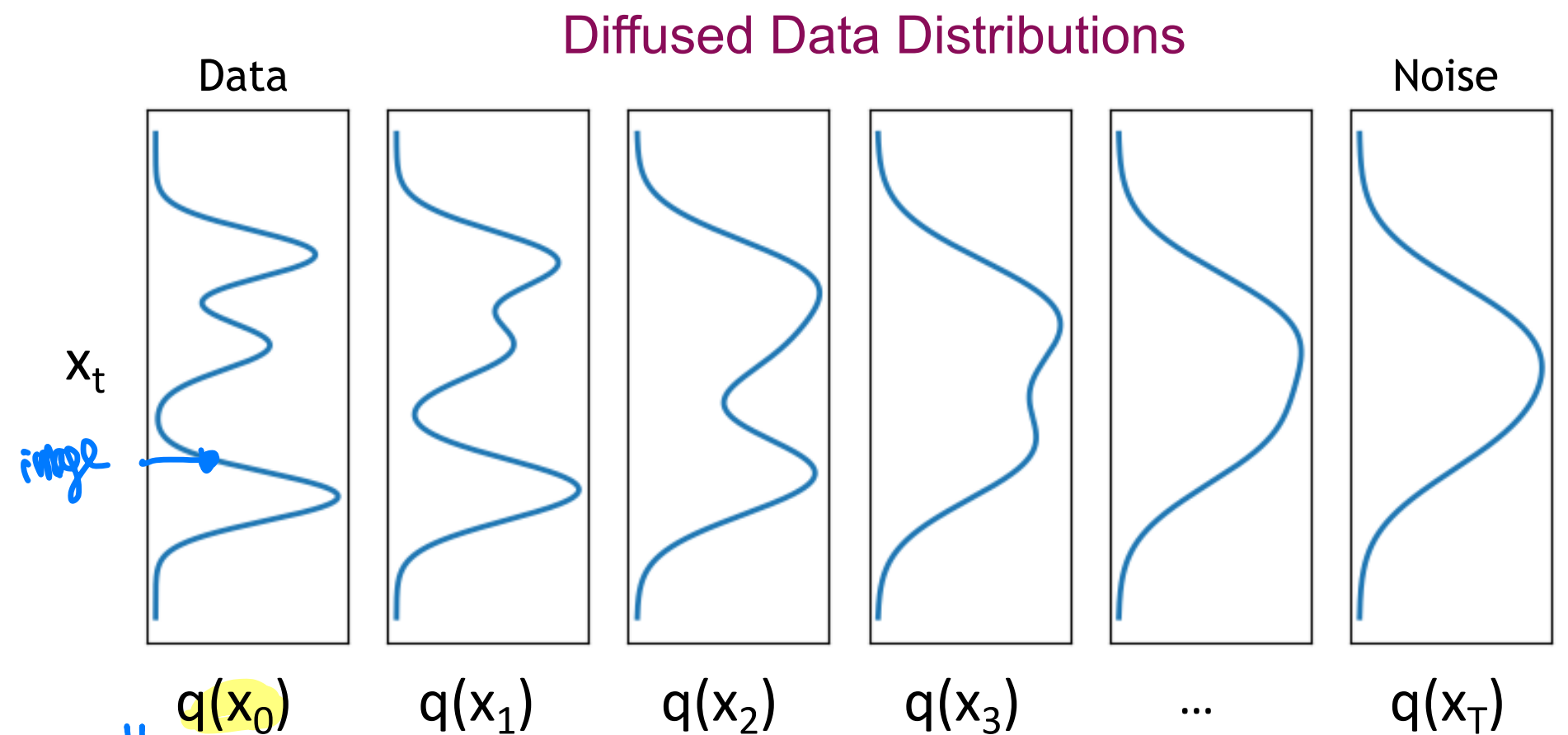
$\beta_t$  values schedule (i.e., the noise schedule) is designed such that  $\bar{\alpha}_T \rightarrow 0$  and  $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

$T \rightarrow \infty$

# What happens to a distribution in the forward diffusion?

So far, we discussed the diffusion kernel  $q(\mathbf{x}_t|\mathbf{x}_0)$  but what about  $q(\mathbf{x}_t)$ ?

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



The diffusion kernel is Gaussian convolution.

*becomes smoother and smoother*

We can sample  $\mathbf{x}_t \sim q(\mathbf{x}_t)$  by first sampling  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$  and then sampling  $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$  (i.e., ancestral sampling).



# Generative Learning by Denoising

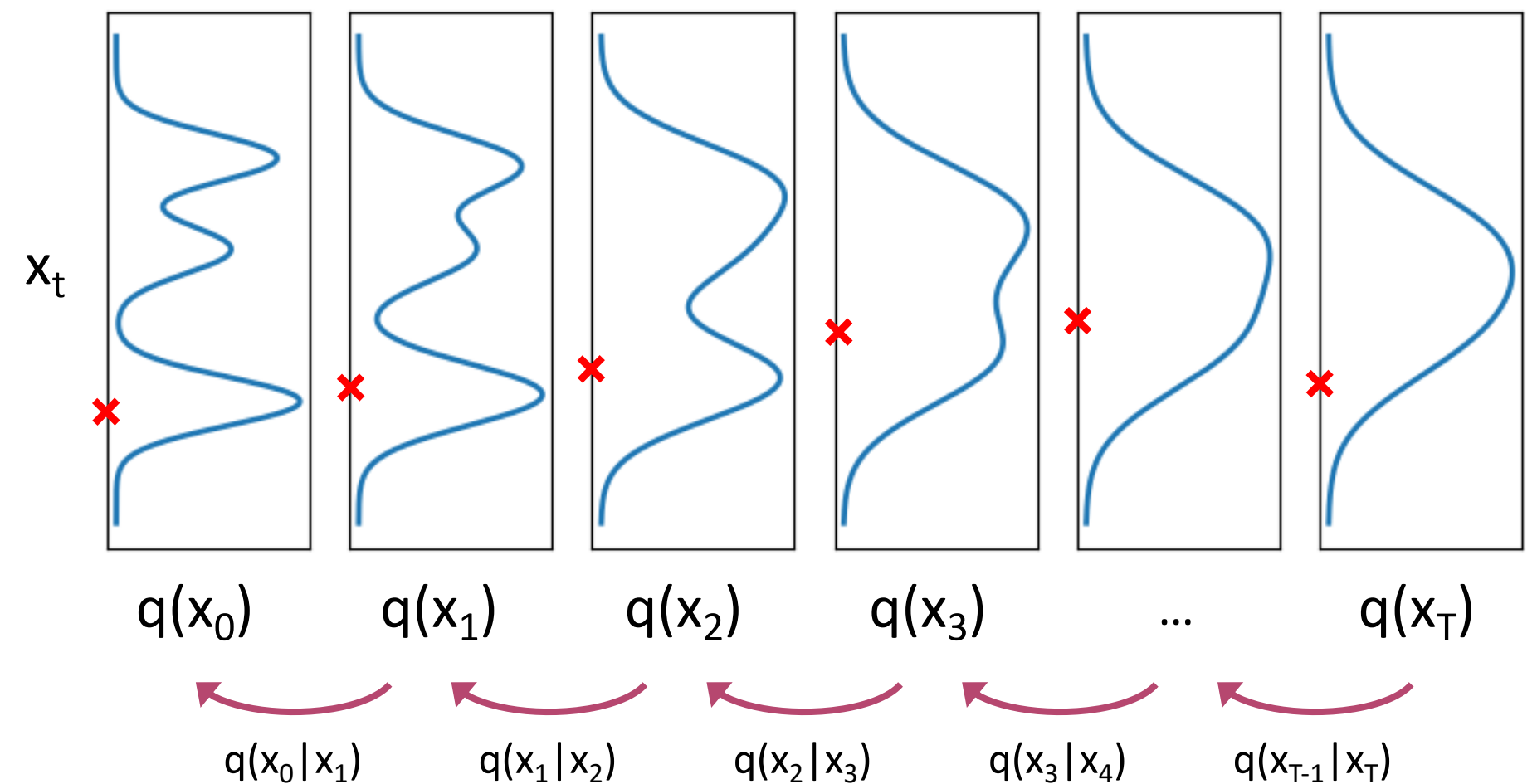
Recall, that the diffusion parameters are designed such that  $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

**Generation:**

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

Iteratively sample  $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{True Denoising Dist.}}$

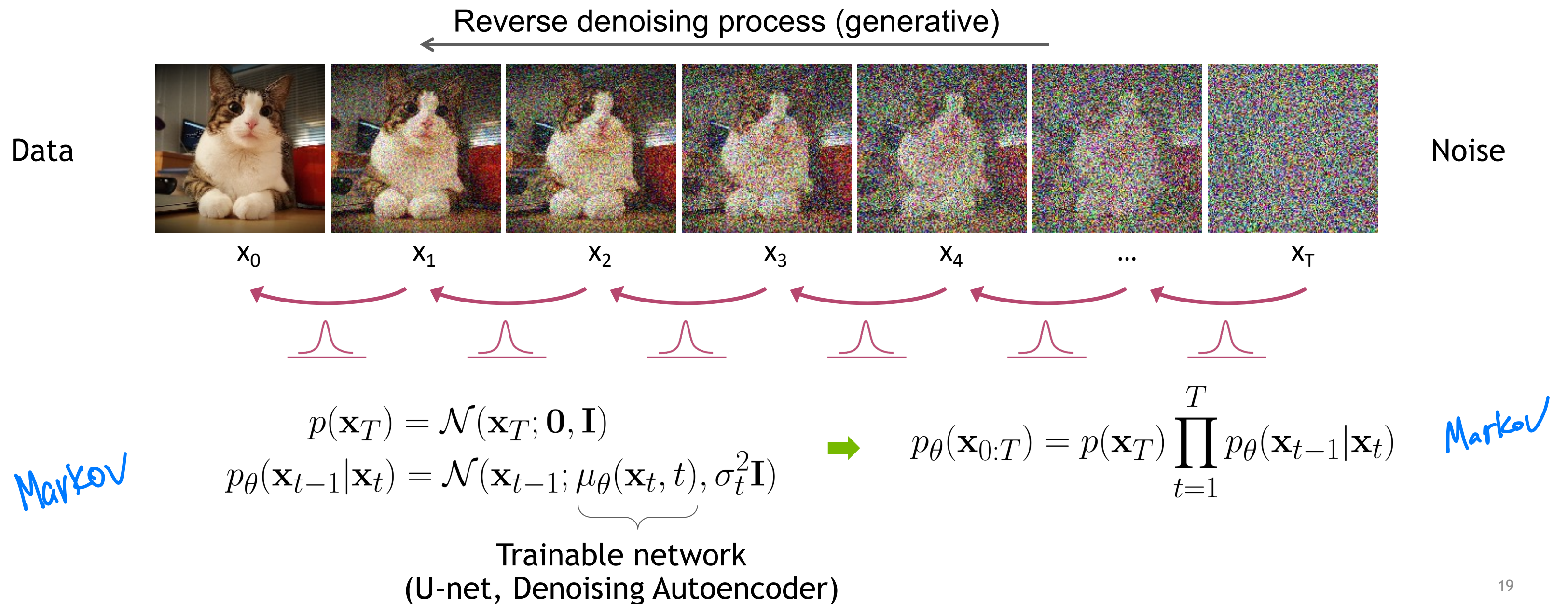
Diffused Data Distributions



Can we approximate  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ ? Yes, we can use a **Normal distribution** if  $\beta_t$  is small in each forward diffusion step.

# Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



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

tractable when conditioned on  $\mathbf{x}_0$

$$q(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t+1}; \hat{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \hat{\Sigma}_t)$$

# Learning Denoising Model

Variational upper bound

derivation: DDPM

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

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

ML  
min

$$\mathbb{E}_{q(\mathbf{x}_0)} [-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ -\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] =: L$$

data distribution  $q(\mathbf{x}_{0:T})$

Recall that  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}$ . [Ho et al. NeurIPS 2020](#) parameterized the mean of denoising model via:

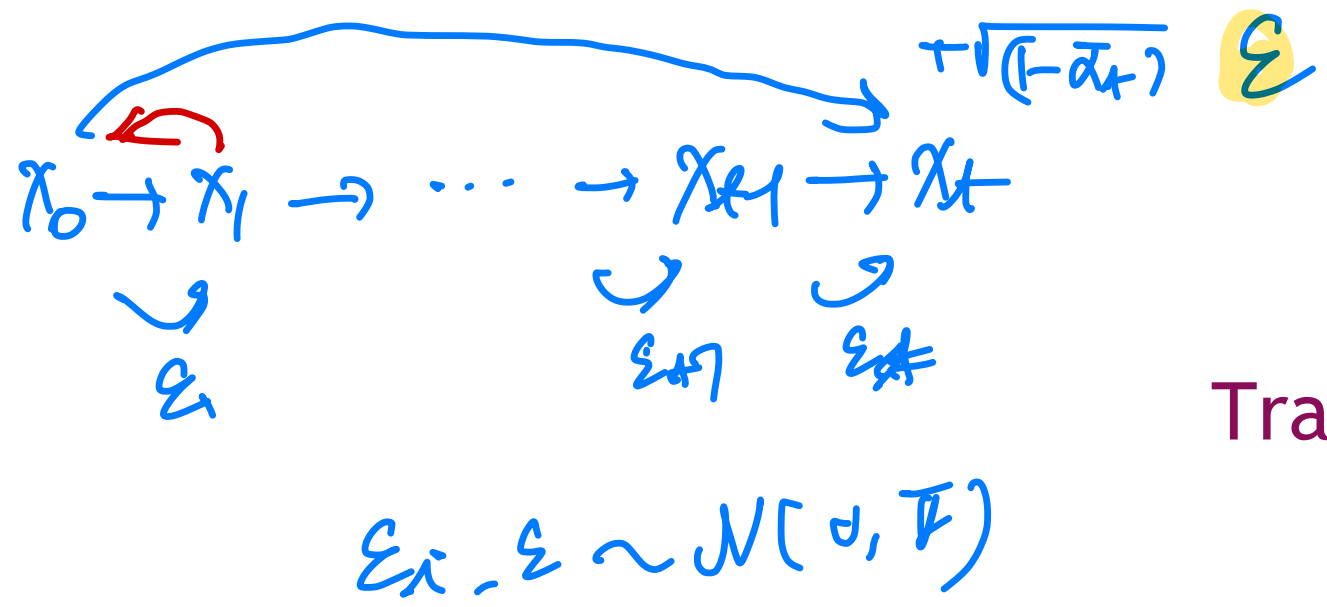
$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}} \left( \mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right)$$

Der.  $q(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t+1} | \mathbf{x}_t)$   
 $p_\theta(\mathbf{x}_{t+1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+1} | \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$

Using a few simple arithmetic operations, we can write down the variational objective as:

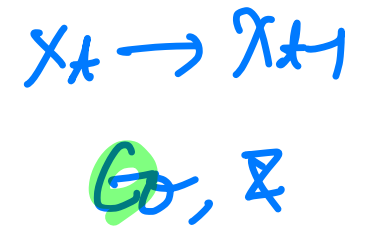
$$L = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), t \sim \mathcal{U}\{1, T\}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \lambda_t \left\| \boldsymbol{\epsilon} - \underbrace{\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon},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

[Ho et al. NeurIPS 2020](#) observe that simply setting  $\lambda_t$  to 1 for all  $t$  works best in practice.



# Summary

## Training and Sample Generation



### Algorithm 1 Training

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

$\epsilon_{\theta}(x_t, t)$

unobserved

$x_t$ : observed

### Algorithm 2 Sampling

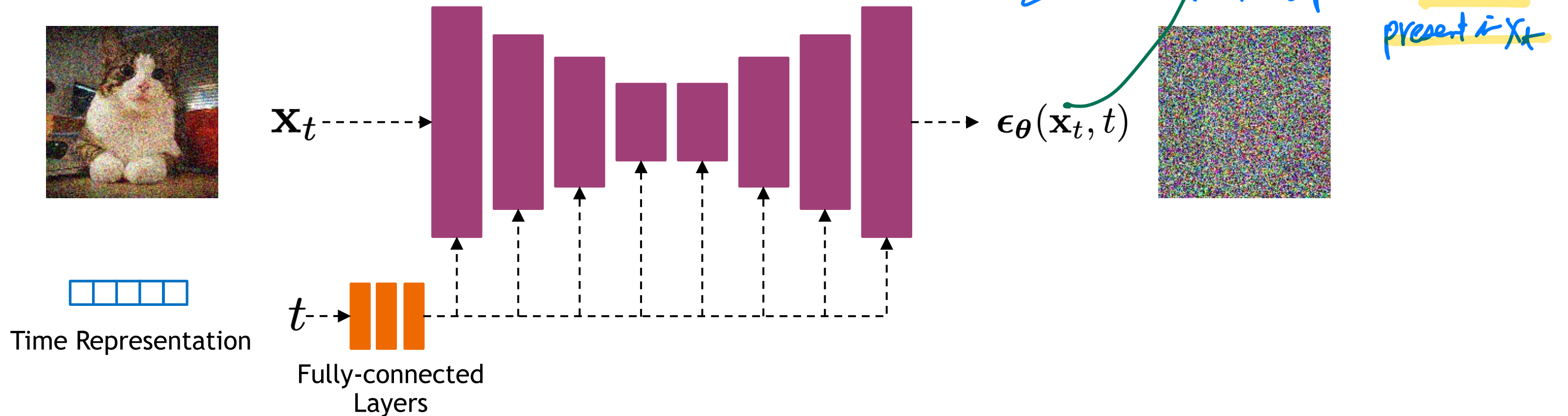
- 1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 2: **for**  $t = T, \dots, 1$  **do**
- 3:  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$   $t > 0$ , or  $\mathbf{z} = 0$   $P_0(x_{t+1}(x_t))$
- 4:  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
- 5: **end for**
- 6: **return**  $\mathbf{x}_0$

$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$

# Implementation Considerations

## Network Architectures

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent  $\epsilon_{\theta}(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see [Dhariwal and Nichol NeurIPS 2021](#))

# Outline

Part (1): Denoising Diffusion Probabilistic Models

**Part (2): Score-based Generative Modeling with Differential Equations**

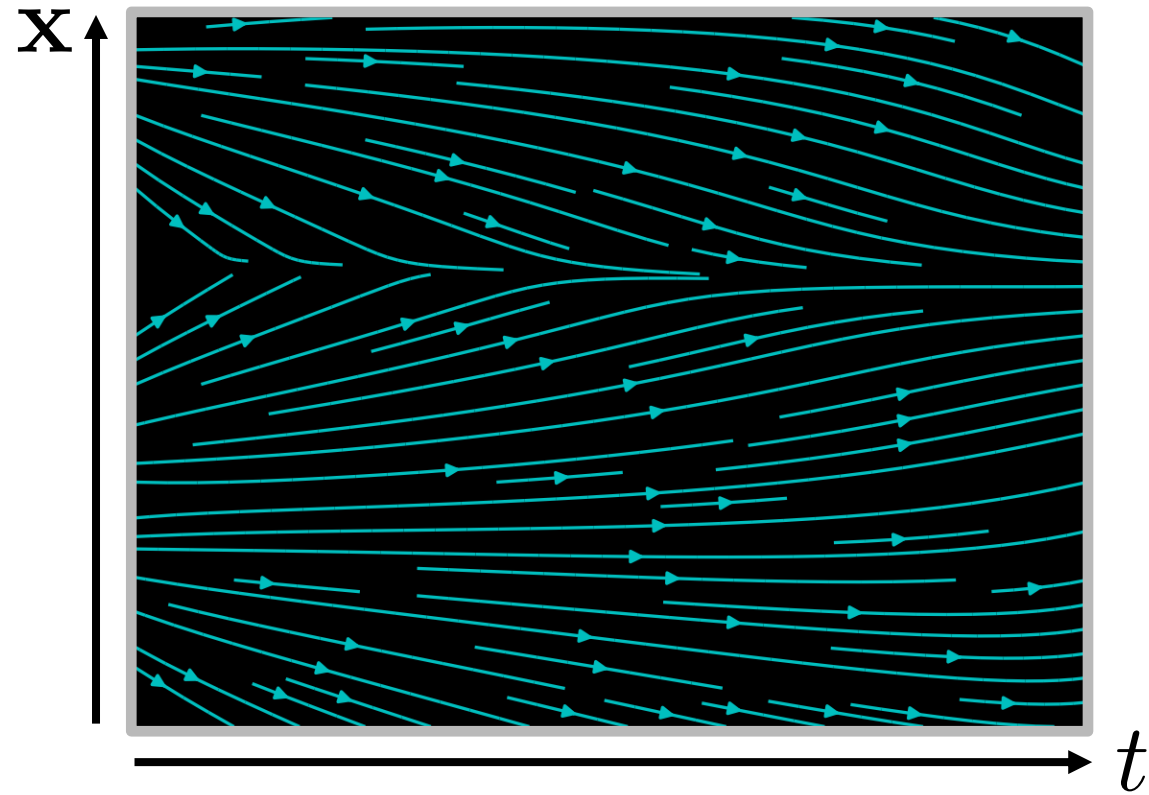
Part (3): Accelerated Sampling

Part (4): Conditional Generation and Guidance

# Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

$$\frac{dx}{dt} = f(x, t) \quad \text{or} \quad dx = f(x, t)dt$$

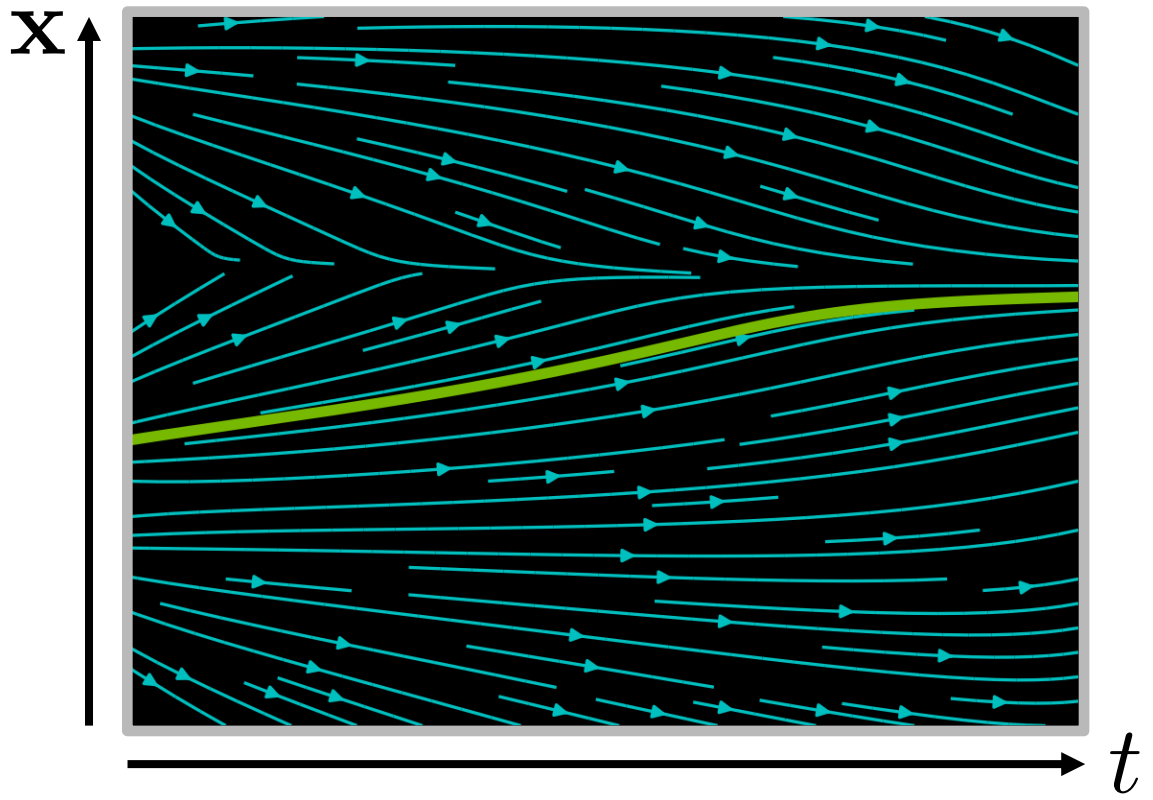


Brownian motion = Wiener process

# Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

$$\frac{dx}{dt} = f(x, t) \text{ or } dx = f(x, t)dt$$



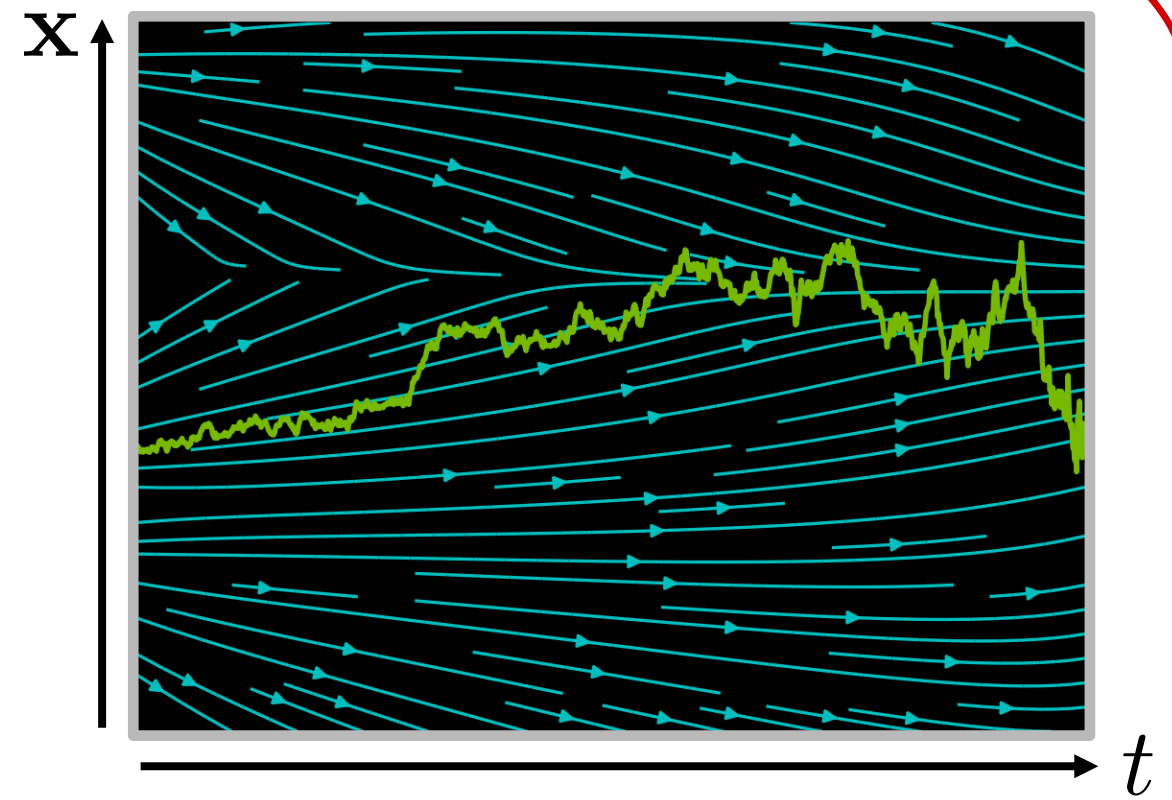
Analytical Solution:  $x(t) = x(0) + \int_0^t f(x, \tau)d\tau$

Iterative Numerical Solution:  $x(t + \Delta t) \approx x(t) + f(x(t), t)\Delta t$

Stochastic Differential Equation (SDE):

$$\frac{dx}{dt} = \underbrace{f(x, t)}_{\text{drift coefficient}} + \underbrace{\sigma(x, t)}_{\text{diffusion coefficient}} \omega_t'$$

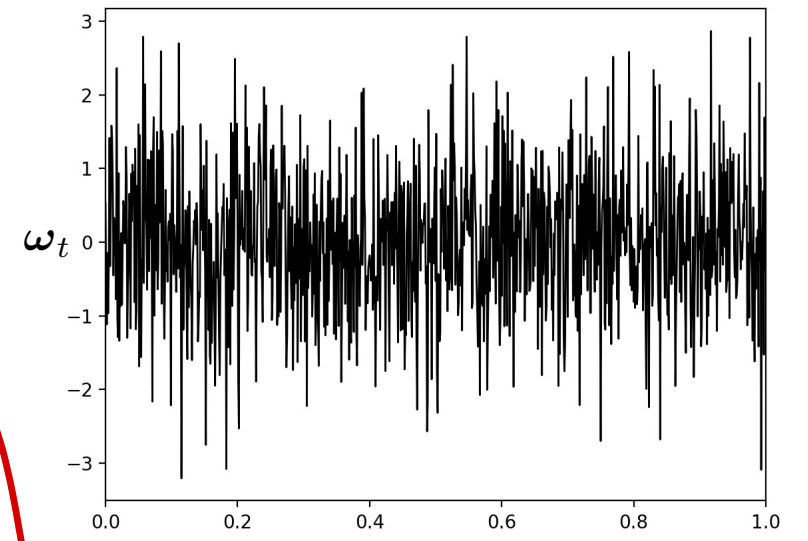
$$(dx = f(x, t)dt + \sigma(x, t)d\omega_t)$$



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

derivative of Brownian motion

Wiener Process (Gaussian White Noise)



$$\frac{dW_t}{dt} = W_t'$$

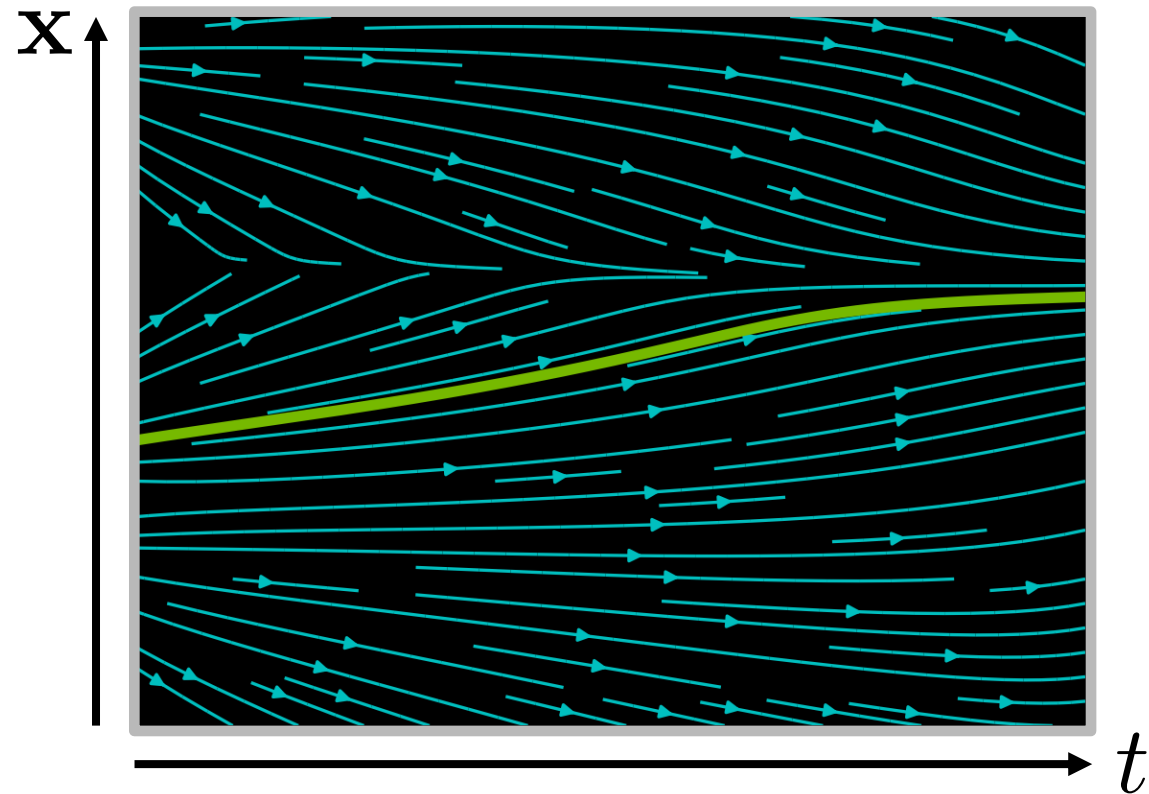
$$W_t' dt = dW_t$$



# Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

$$\frac{dx}{dt} = f(x, t) \quad \text{or} \quad dx = f(x, t)dt$$



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

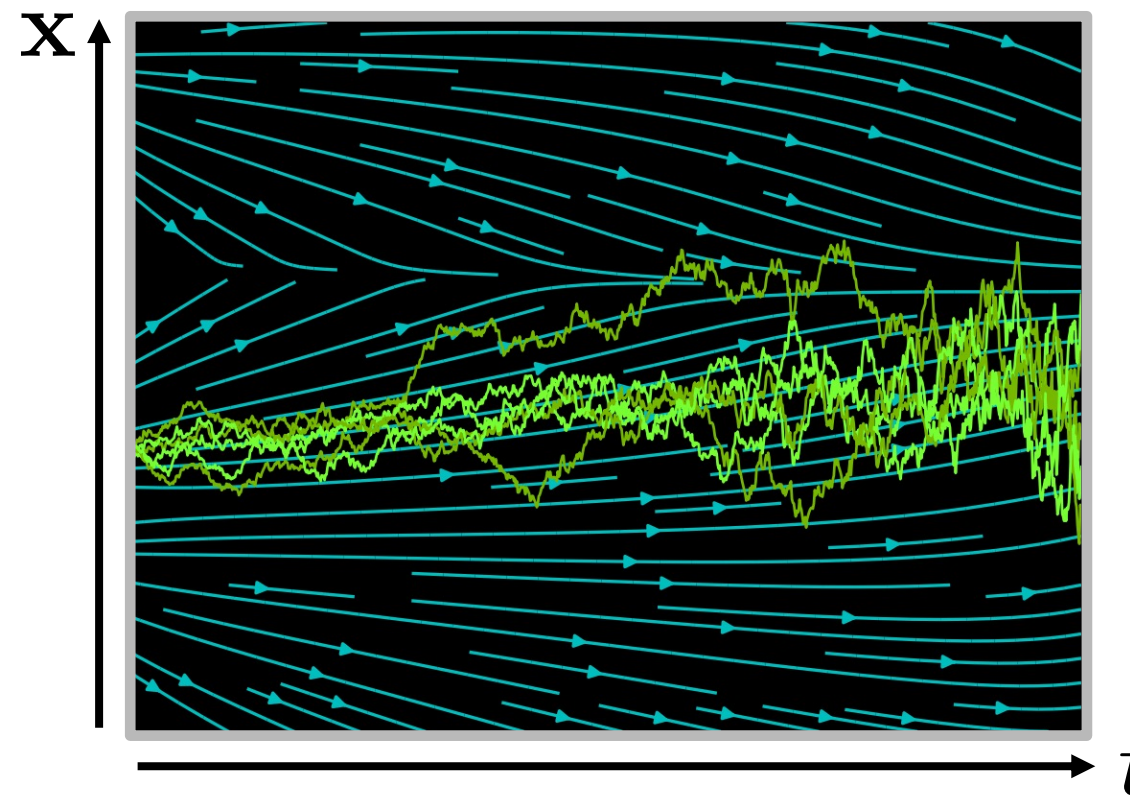
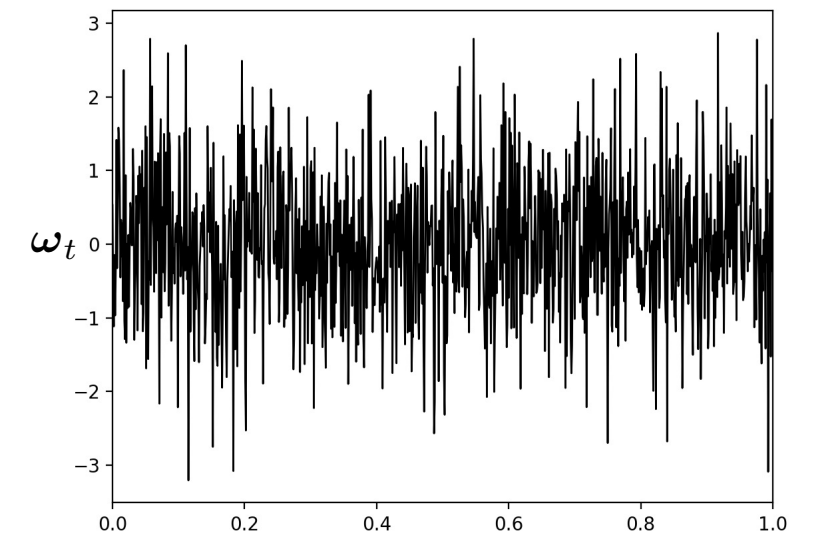
Iterative Numerical Solution: 
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t) \Delta t$$

Stochastic Differential Equation (SDE):

$$\frac{dx}{dt} = \underbrace{f(x, t)}_{\text{drift coefficient}} + \underbrace{\sigma(x, t)\omega_t}_{\text{diffusion coefficient}}$$

$$\left( dx = f(x, t)dt + \sigma(x, t)d\omega_t \right)$$

Wiener Process  
(Gaussian White Noise)



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

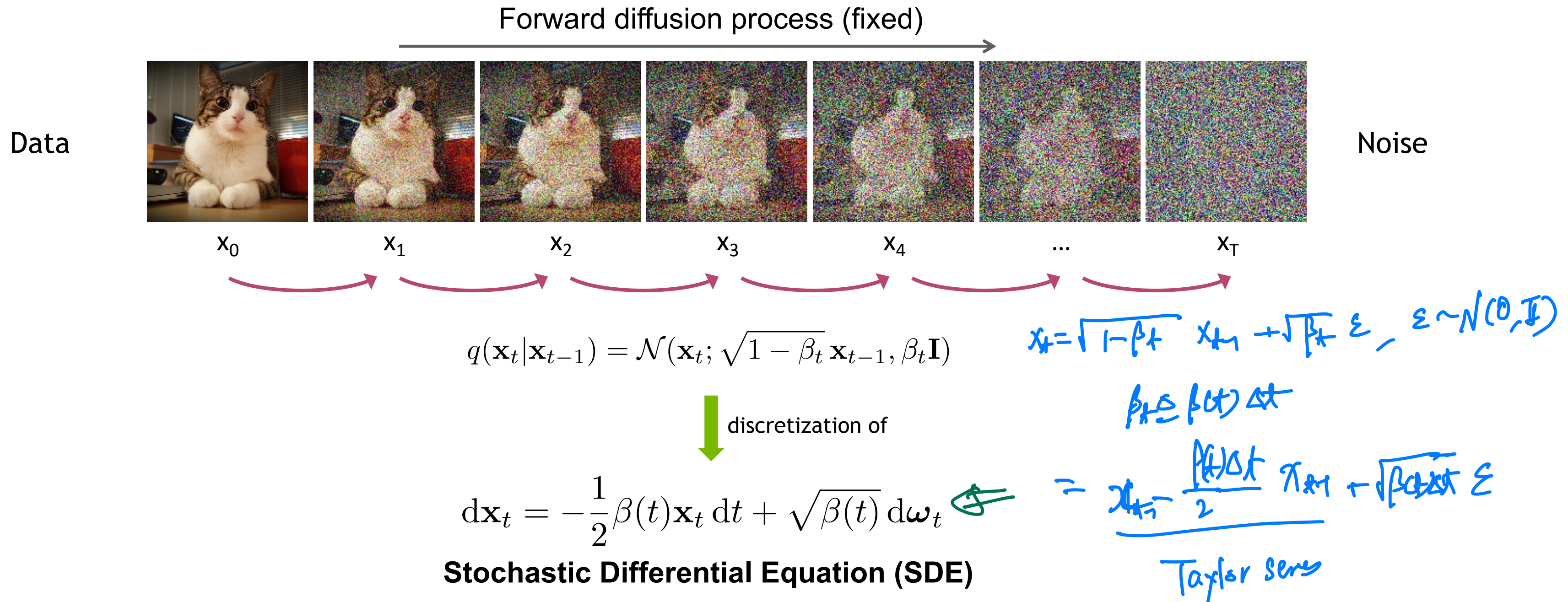
$dx = f(x, t)dt + \sigma(x, t)dW$   
 $W$ : Brownian

$x(t+\Delta) \approx x(t) + f \cdot \Delta t + \sigma \frac{(N(t+\Delta) - N(t))}{\Delta t}$

By def.  $\sim \mathcal{N}(\Delta W; 0, \Delta t \mathbb{I})$   
 $\sim \sqrt{\Delta t} \mathcal{N}_{\Delta t}(\mathbf{0}, \mathbb{I})$

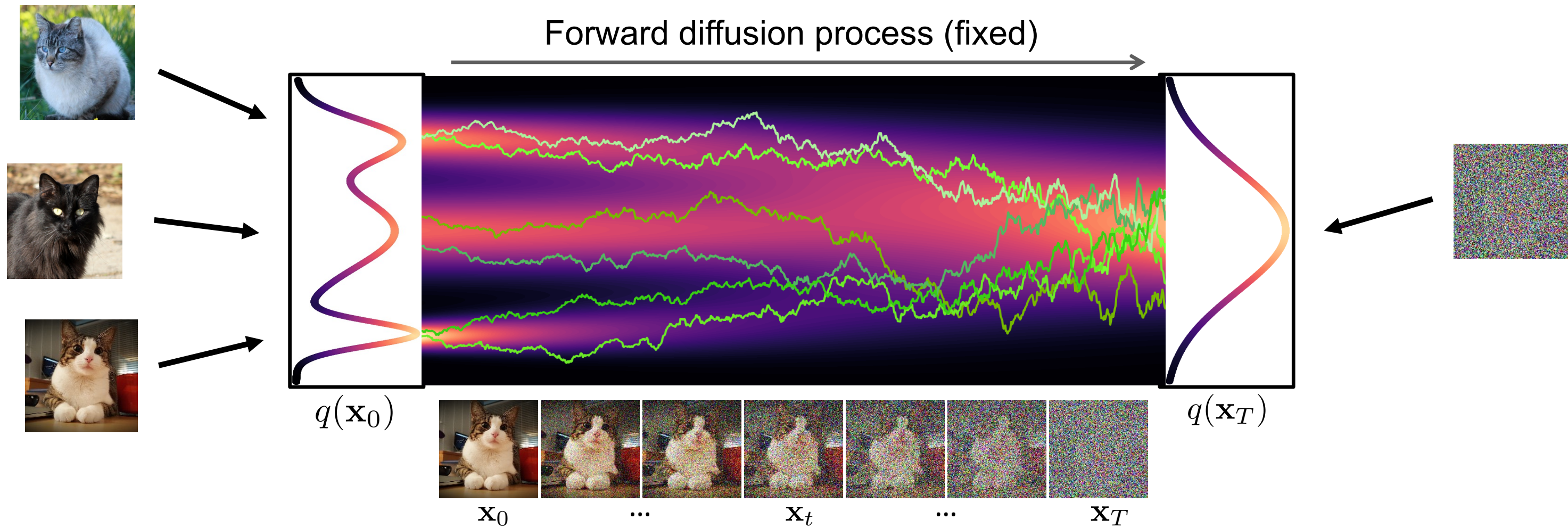
# Forward Diffusion Process as Stochastic Differential Equation

Consider the limit of many small steps:



SDE: Itô integral  
 ODE: Lebesgue Integral

# Forward Diffusion Process as Stochastic Differential Equation



**Forward Diffusion SDE:**

$$dx_t = \underbrace{-\frac{1}{2}\beta(t)x_t dt}_{\text{drift term}} + \underbrace{\sqrt{\beta(t)} d\omega_t}_{\text{diffusion term}}$$

*affine*

drift term  
(pulls towards mode)

*origin*

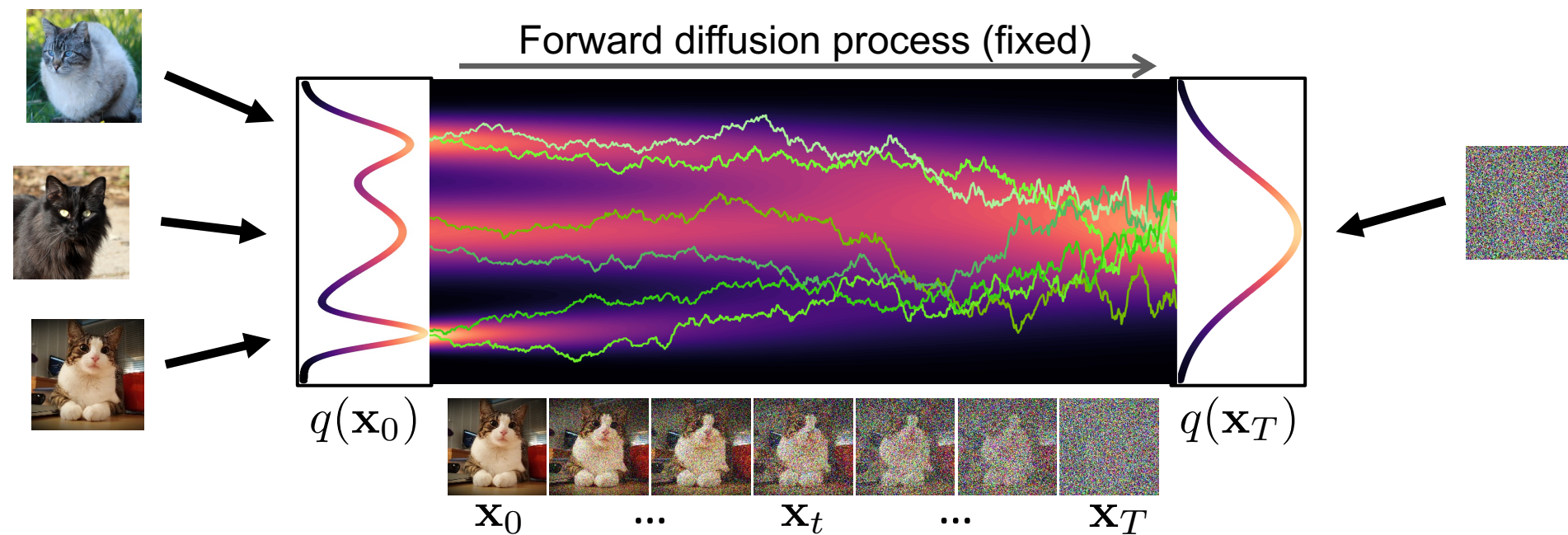
diffusion term  
(injects noise)

*→ sel → Gaussian*

*→ q(x\_t) : FPK → mean over ODE*

*≠ N(0, I)*

# The Generative Reverse Stochastic Differential Equation

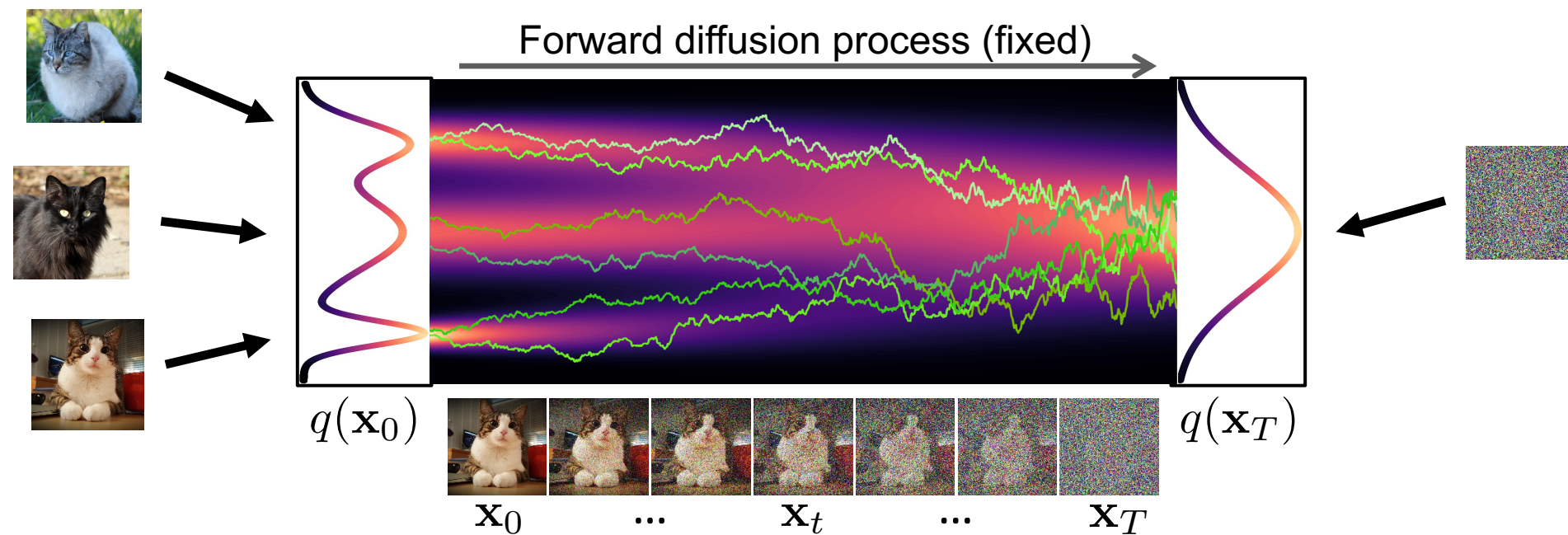


**Forward Diffusion SDE:**

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)} d\omega_t$$

But what about the reverse direction, necessary for generation?

# The Generative Reverse Stochastic Differential Equation



**Forward Diffusion SDE:**

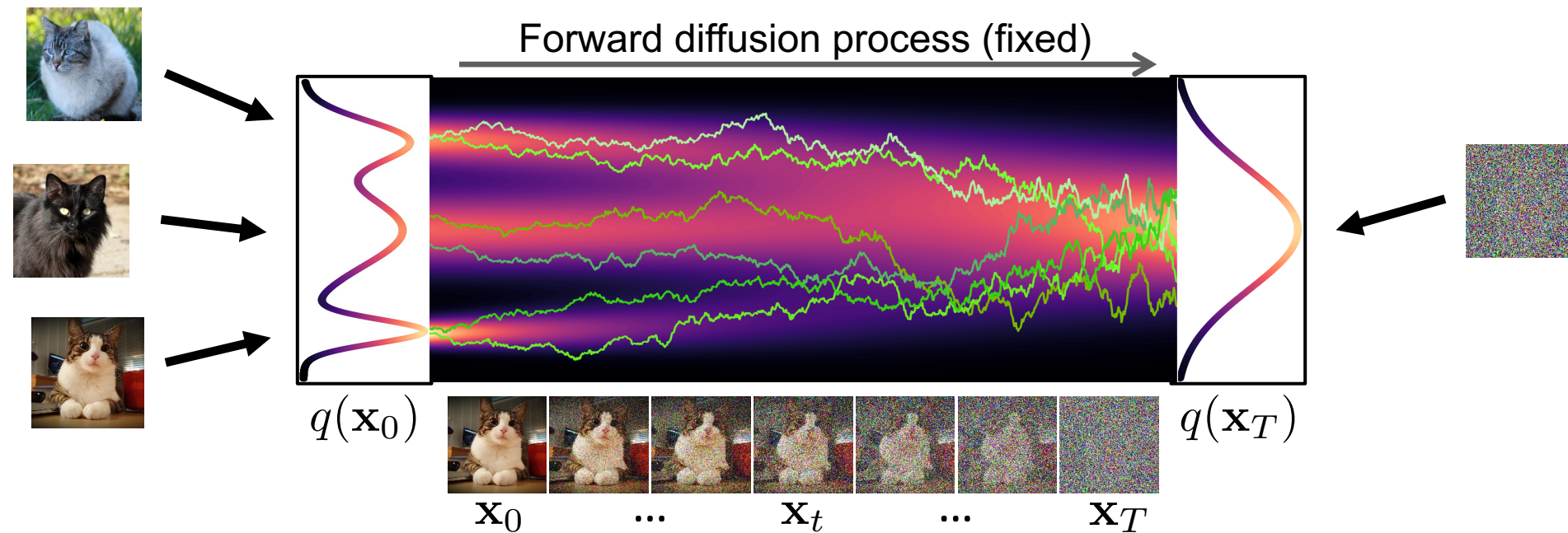
$$d\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t}_{\text{drift term}} dt + \underbrace{\sqrt{\beta(t)} d\omega_t}_{\text{diffusion term}}$$

**Reverse Generative Diffusion SDE:**

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

➔ **Simulate reverse diffusion process: Data generation from random noise!**

# The Generative Reverse Stochastic Differential Equation



**Forward Diffusion SDE:**

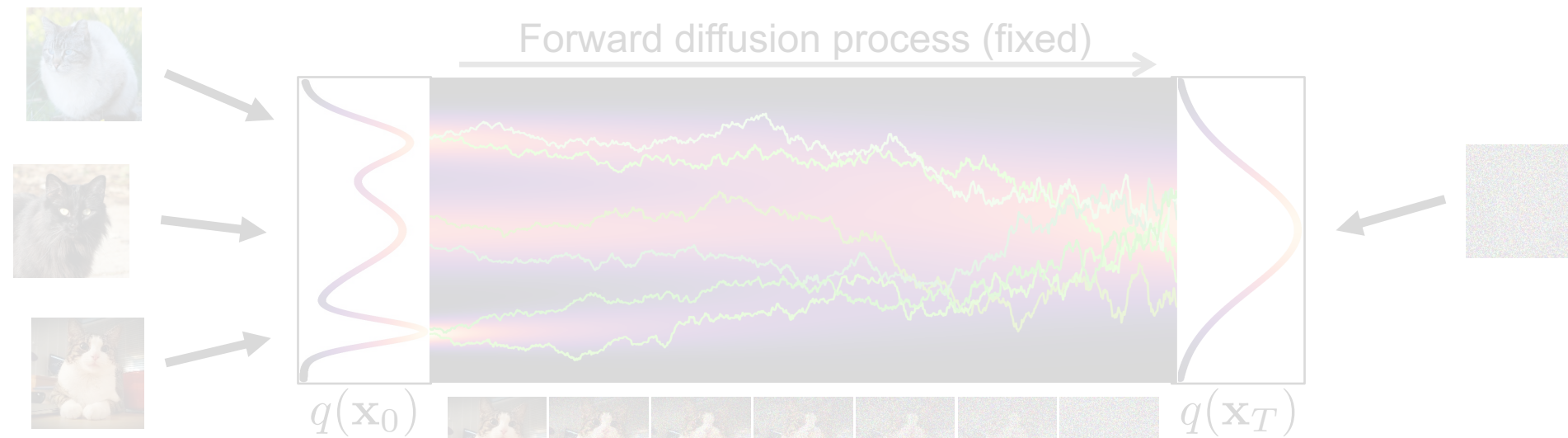
$$d\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t}_{\text{drift term}} dt + \underbrace{\sqrt{\beta(t)} d\omega_t}_{\text{diffusion term}}$$

**Reverse Generative Diffusion SDE:**

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

➔ **Simulate reverse diffusion process: Data generation from random noise!**

# The Generative Reverse Stochastic Differential Equation



**But how to get the score function  $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ ?**

Forward Diffusion SDE:

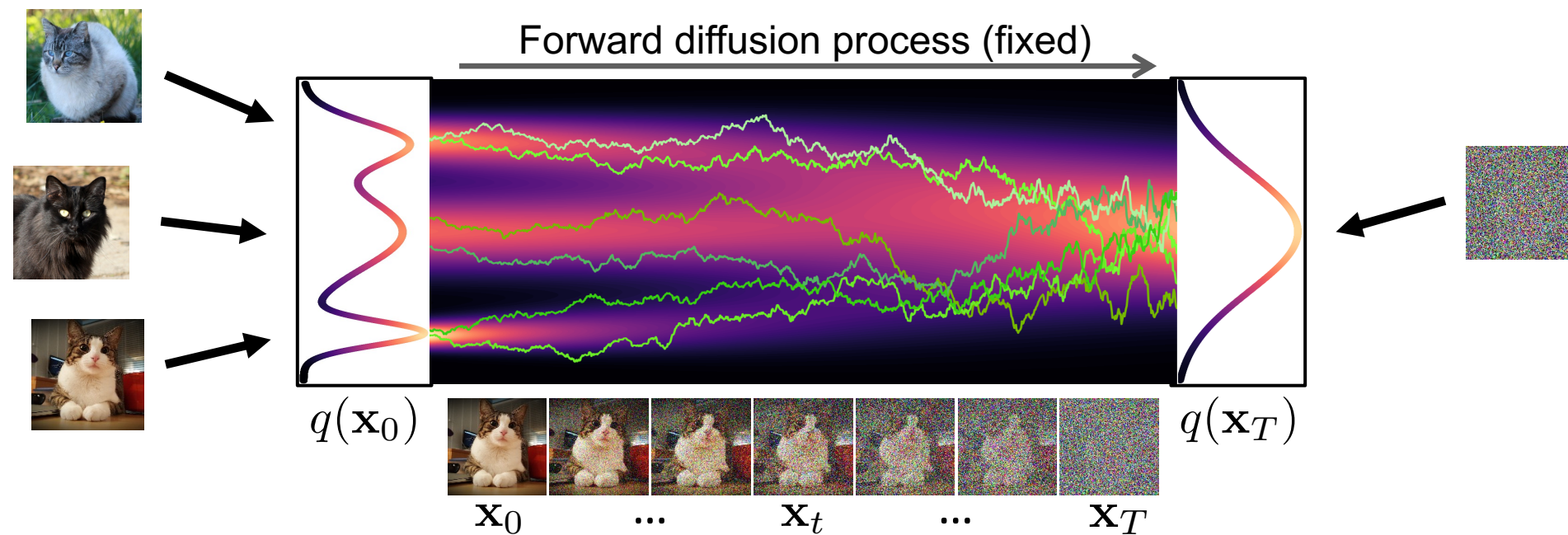
$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)} d\omega_t$$

Reverse Generative Diffusion SDE:

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

➔ Simulate reverse diffusion process: Data generation from random noise!

# Score Matching



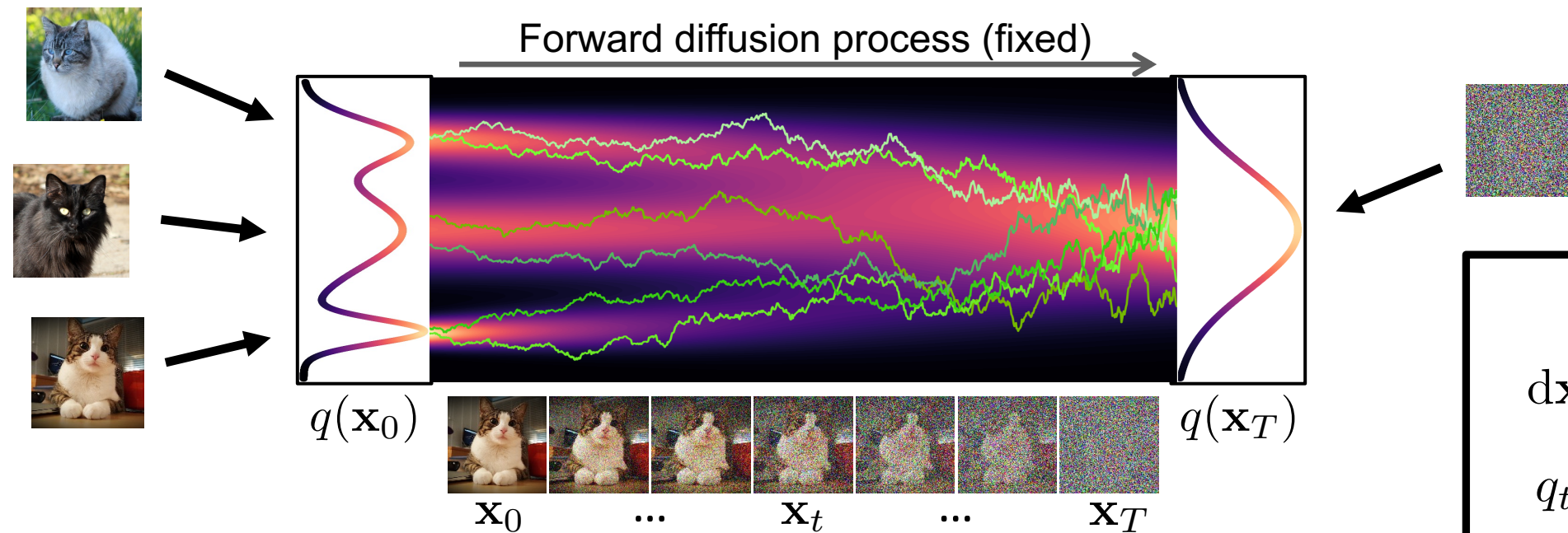
- Naïve idea, learn model for the score function by direct regression?

$$\min_{\theta} \underbrace{\mathbb{E}_{t \sim \mathcal{U}(0, T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_t \sim q_t(\mathbf{x}_t)}}_{\text{diffused data } \mathbf{x}_t} \underbrace{\|\mathbf{s}_{\theta}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)\|_2^2}_{\text{score of diffused data (marginal)}}$$

➔ **But  $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$  (score of the *marginal diffused density*  $q_t(\mathbf{x}_t)$ ) is not tractable!**



# Denoising Score Matching



“Variance Preserving” SDE:

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)} d\omega_t$$

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

*Eq. (29) in Song*

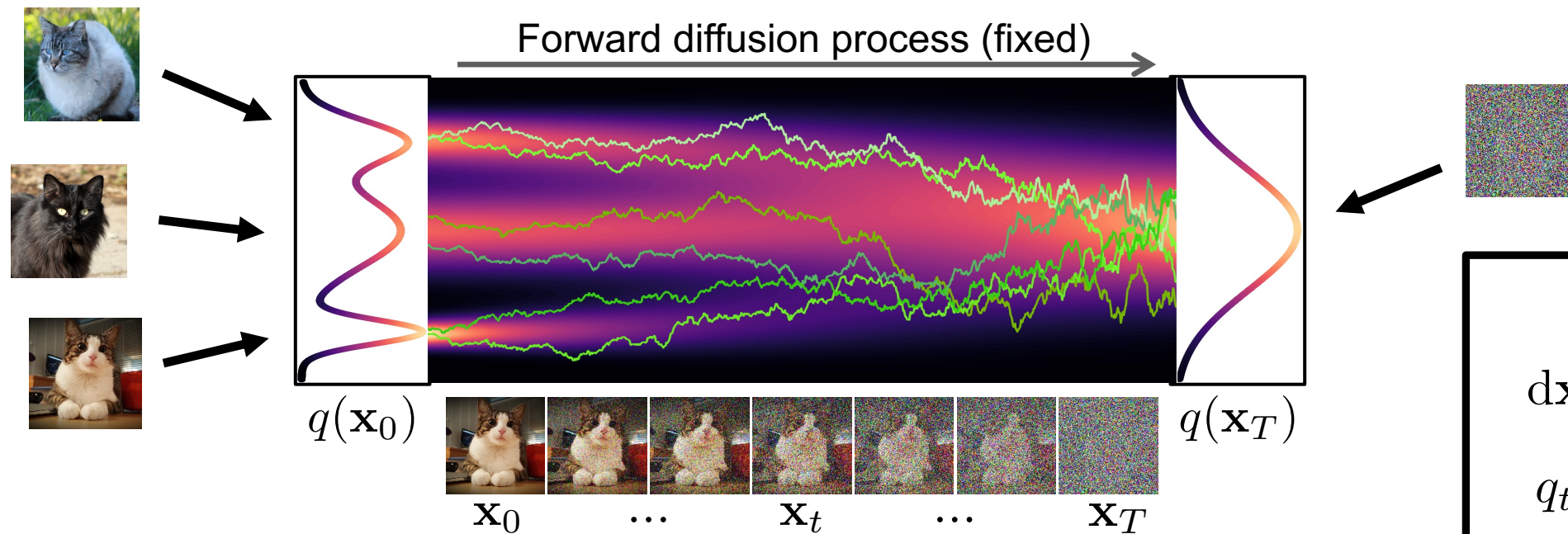
- Instead, diffuse individual data points  $\mathbf{x}_0$ . Diffused  $q_t(\mathbf{x}_t|\mathbf{x}_0)$  *is* tractable!
- Denoising Score Matching:

$$\min_{\theta} \underbrace{\mathbb{E}_{t \sim \mathcal{U}(0, T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_0 \sim q_0(\mathbf{x}_0)}}_{\text{data sample } \mathbf{x}_0} \underbrace{\mathbb{E}_{\mathbf{x}_t \sim q_t(\mathbf{x}_t|\mathbf{x}_0)}}_{\text{diffused data sample } \mathbf{x}_t} \underbrace{\|s_{\theta}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0)\|_2^2}_{\text{score of diffused data sample}}$$

➔ **After expectations,  $s_{\theta}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ !**

# Denoising Score Matching

## Implementation 1: Noise Prediction



“Variance Preserving” SDE:

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)} d\omega_t$$

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

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

• **Denoising Score Matching:**

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

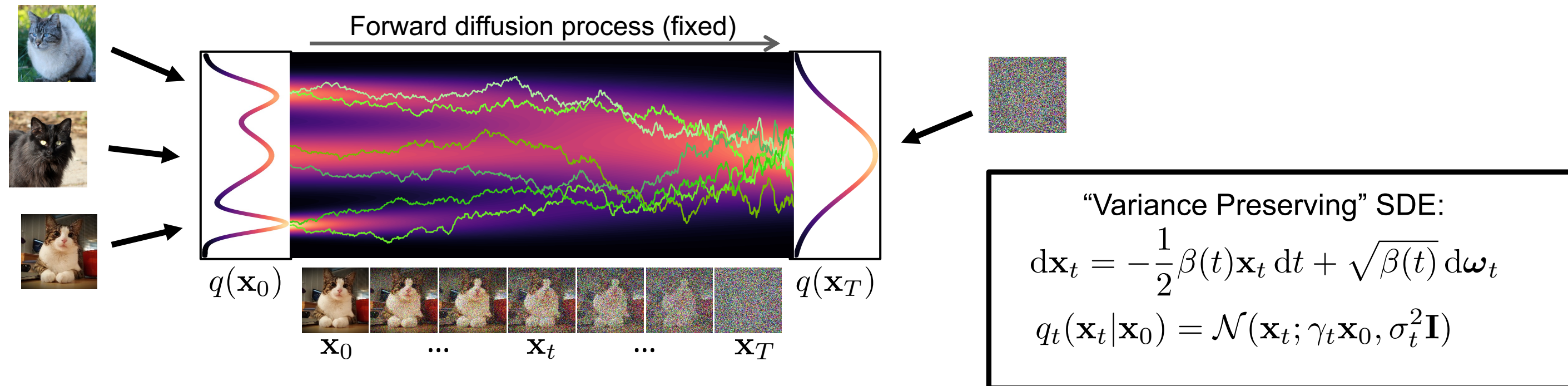
• Re-parametrized sampling:  $\mathbf{x}_t = \gamma_t\mathbf{x}_0 + \sigma_t\epsilon$      $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$     (4) in L03-2 (DDPM)

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

• Neural network model:  $\mathbf{s}_{\theta}(\mathbf{x}_t, t) := -\frac{\epsilon_{\theta}(\mathbf{x}_t, t)}{\sigma_t}$

→  $\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}(0, T)} \mathbb{E}_{\mathbf{x}_0 \sim q_0(\mathbf{x}_0)} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \frac{1}{\sigma_t^2} \|\epsilon - \epsilon_{\theta}(\mathbf{x}_t, t)\|_2^2$     (12) in L03-2

# Denoising Score Matching



- Instead, diffuse individual data points  $\mathbf{x}_0$ . Diffused  $q_t(\mathbf{x}_t|\mathbf{x}_0)$  *is* tractable!
- **Denoising Score Matching:**

$$\min_{\theta} \underbrace{\mathbb{E}_{t \sim \mathcal{U}(0, T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_0 \sim q_0(\mathbf{x}_0)}}_{\text{data sample } \mathbf{x}_0} \underbrace{\mathbb{E}_{\mathbf{x}_t \sim q_t(\mathbf{x}_t|\mathbf{x}_0)}}_{\text{diffused data sample } \mathbf{x}_t} \underbrace{\|\mathbf{s}_{\theta}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0)\|_2^2}_{\text{neural network score of diffused data sample}}$$

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

**Same objectives in Part (1)!**

# Different Parameterizations

Noise prediction network:

$$\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}(0, T), x_0 \sim q_0(x_0), \epsilon \sim \mathcal{N}(0, I)} \lambda(t) \|\epsilon - \epsilon_{\theta}(x_t, t)\|_2^2$$

More sophisticated model

parametrizations and loss

Denoising network:

$$\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}(0, T), x_0 \sim q_0(x_0), \epsilon \sim \mathcal{N}(0, I)} \lambda'(t) \|x_0 - \hat{x}_{\theta}(x_t, t)\|_2^2$$

weightings possible!

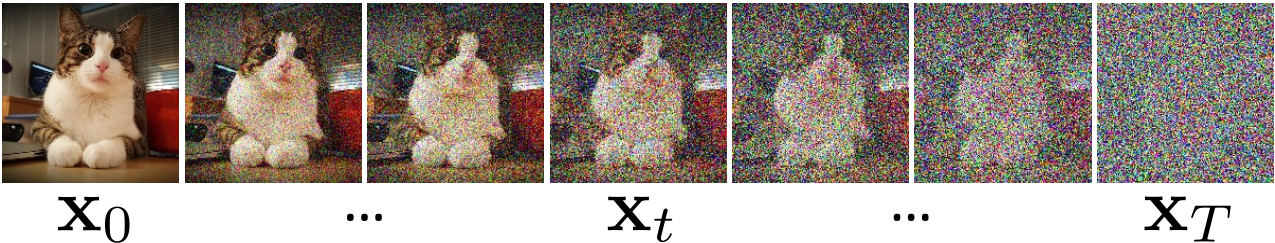
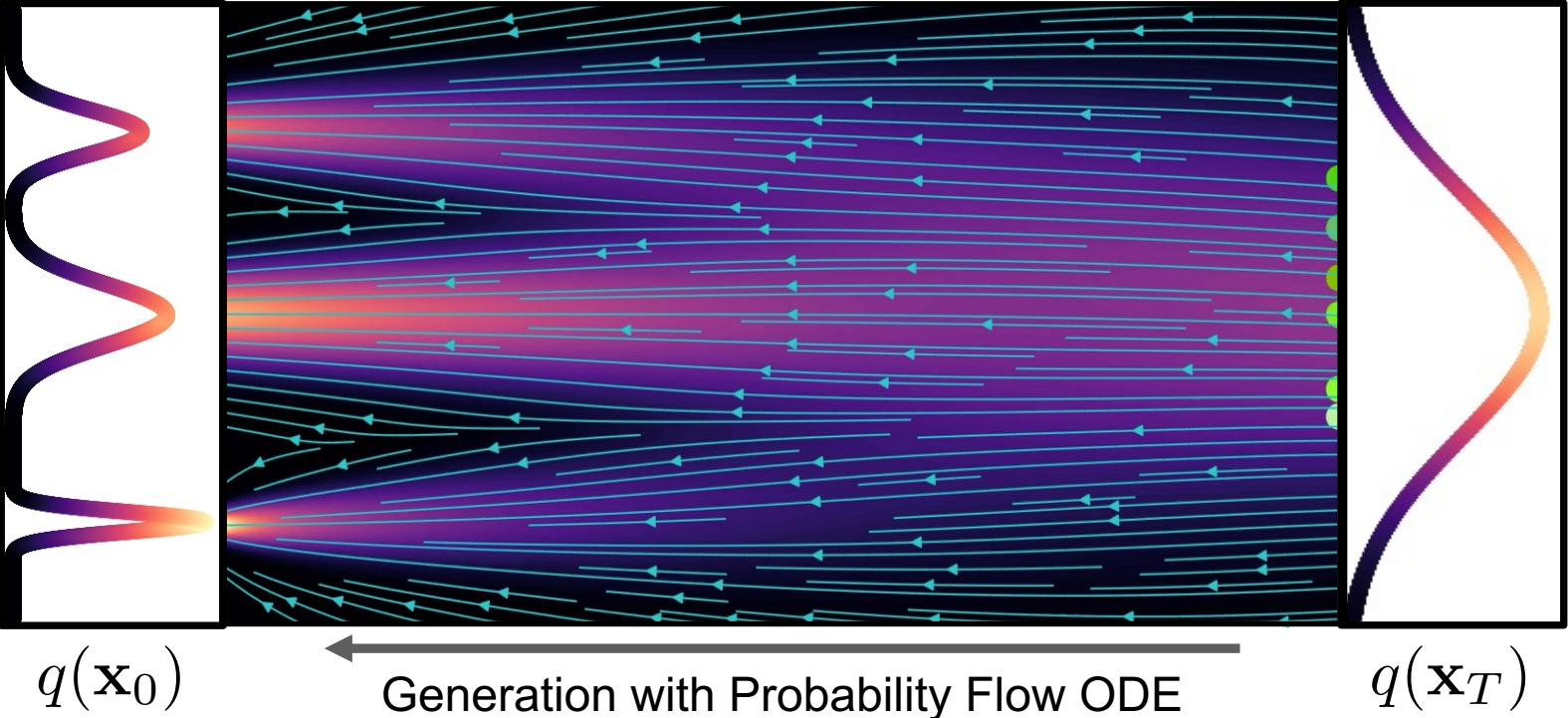
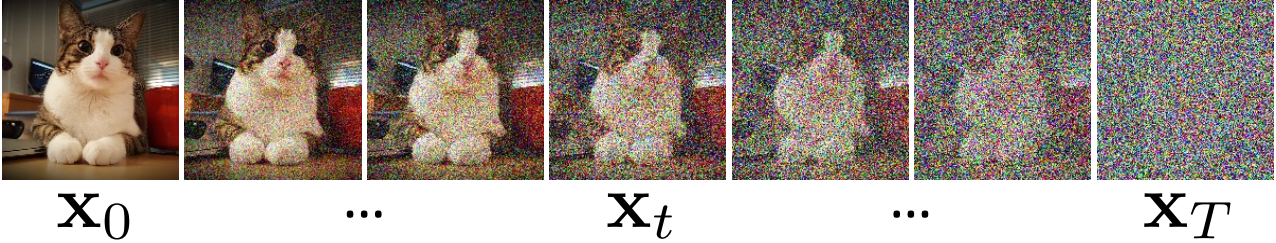
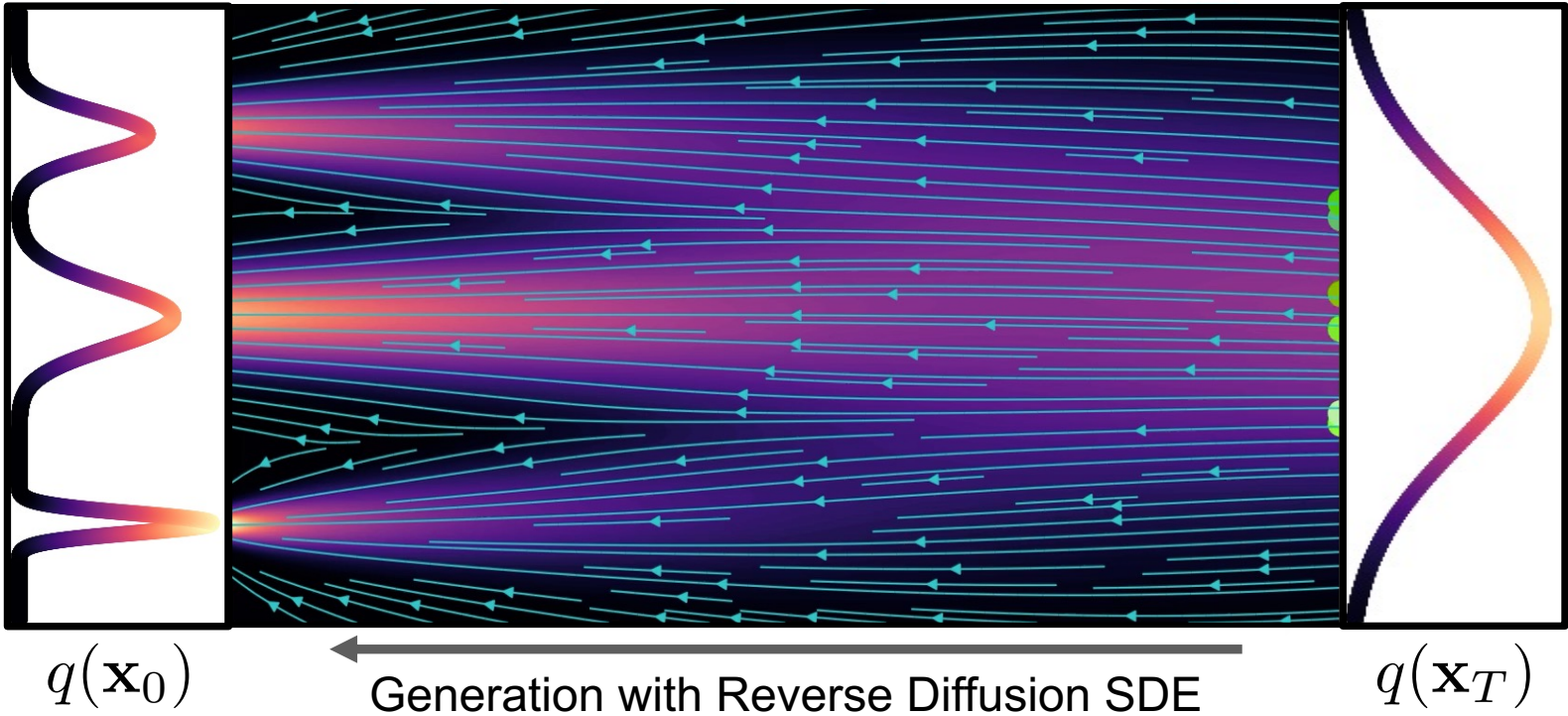
Karras et al., ["Elucidating the Design Space of Diffusion-Based Generative Models"](#), NeurIPS 2022

v-prediction:

$$v := \gamma_t \epsilon - \sigma_t x_0$$

$$\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}(0, T), x_0 \sim q_0(x_0), \epsilon \sim \mathcal{N}(0, I)} \lambda''(t) \|v - v_{\theta}(x_t, t)\|_2^2$$

# Synthesis with SDE vs. ODE



- **Generative Reverse Diffusion SDE (stochastic):**

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t) [\mathbf{x}_t + 2\mathbf{s}_\theta(\mathbf{x}_t, t)] dt + \sqrt{\beta(t)} d\bar{\omega}_t$$

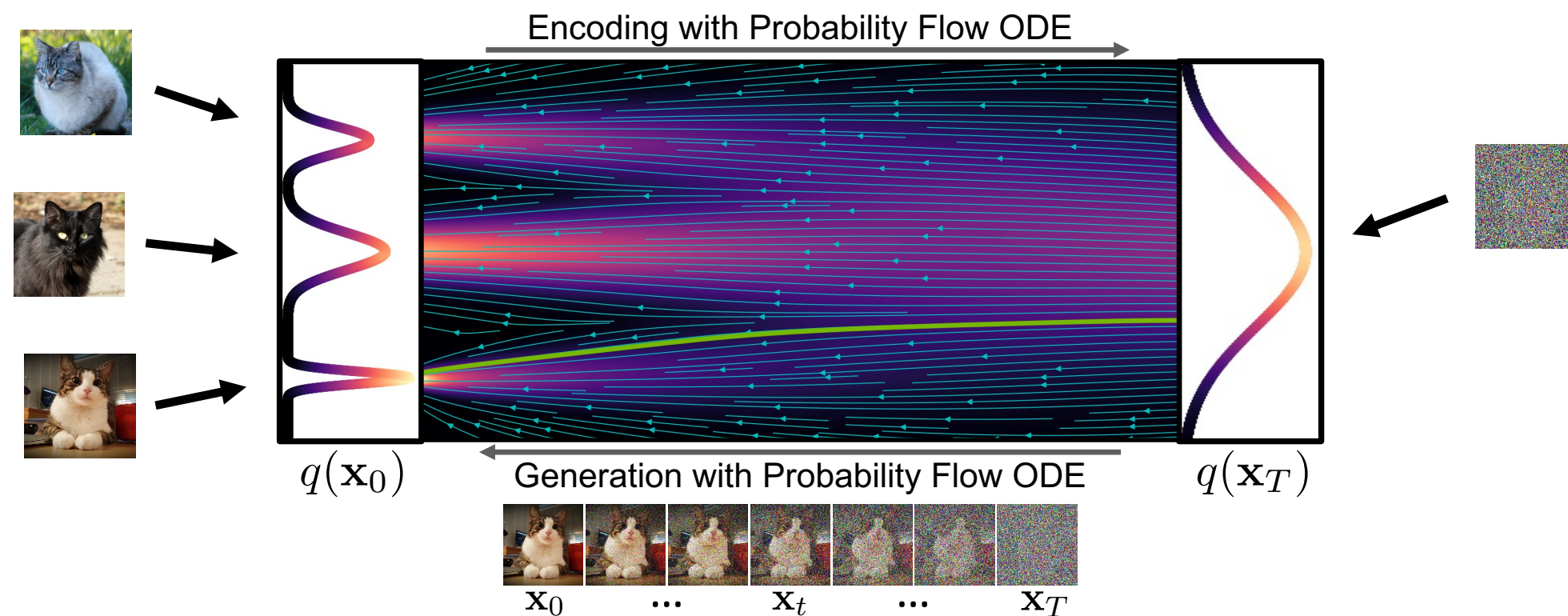
- **Generative Probability Flow ODE (deterministic):**

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t) [\mathbf{x}_t + \mathbf{s}_\theta(\mathbf{x}_t, t)] dt$$

*Eq. (37) in Song*

# Probability Flow ODE

## Diffusion Models as Neural ODEs



- ➔ Enables use of advanced ODE solvers
- ➔ Deterministic encoding and generation (semantic image interpolation, etc.)
- ➔ Log-likelihood computation (instantaneous change of variables):

(39) in Sou

# Outline

Part (1): Denoising Diffusion Probabilistic Models

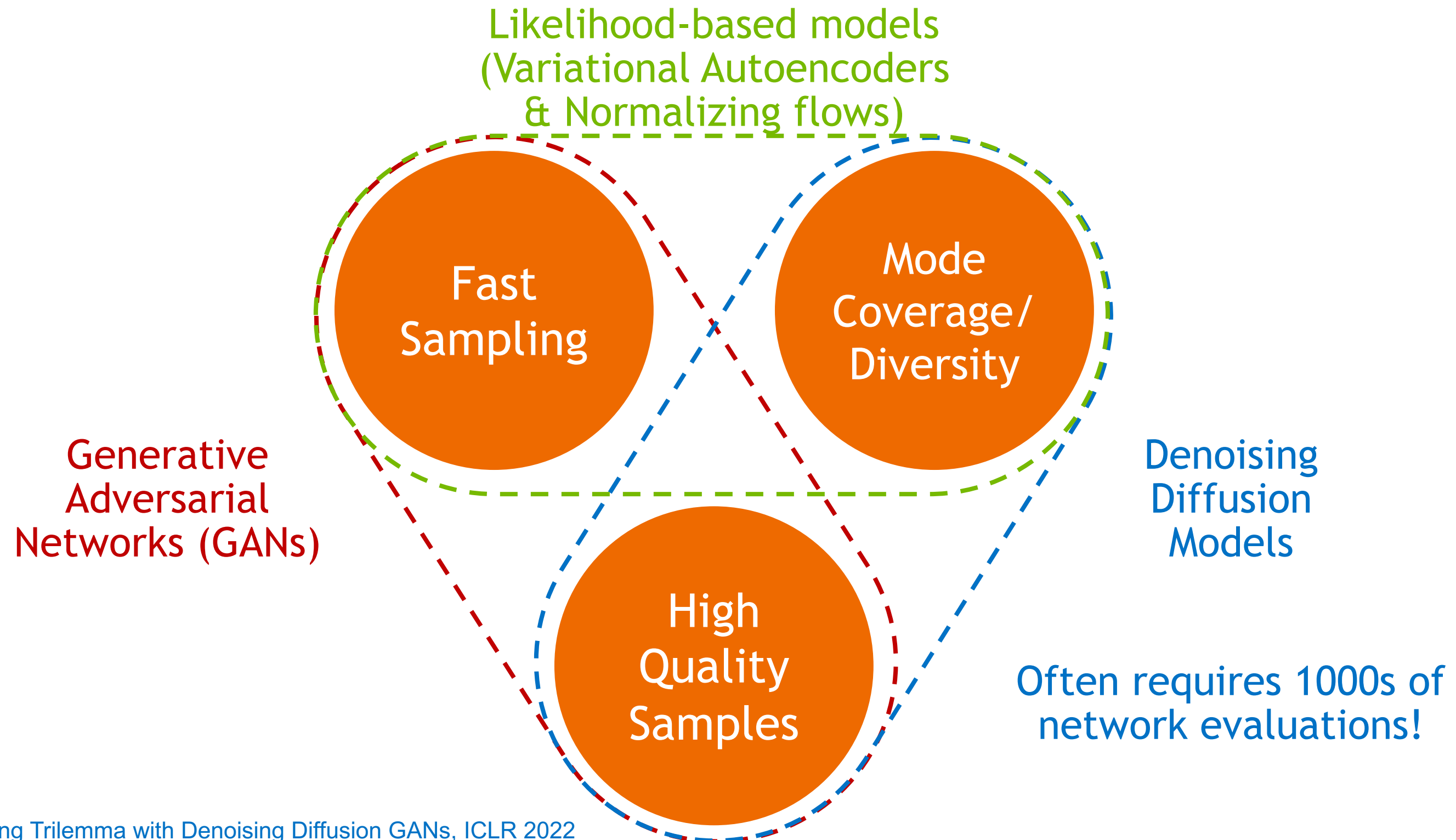
Part (2): Score-based Generative Modeling with Differential Equations

**Part (3): Accelerated Sampling**

Part (4): Conditional Generation and Guidance

# What makes a good generative model?

## The generative learning trilemma

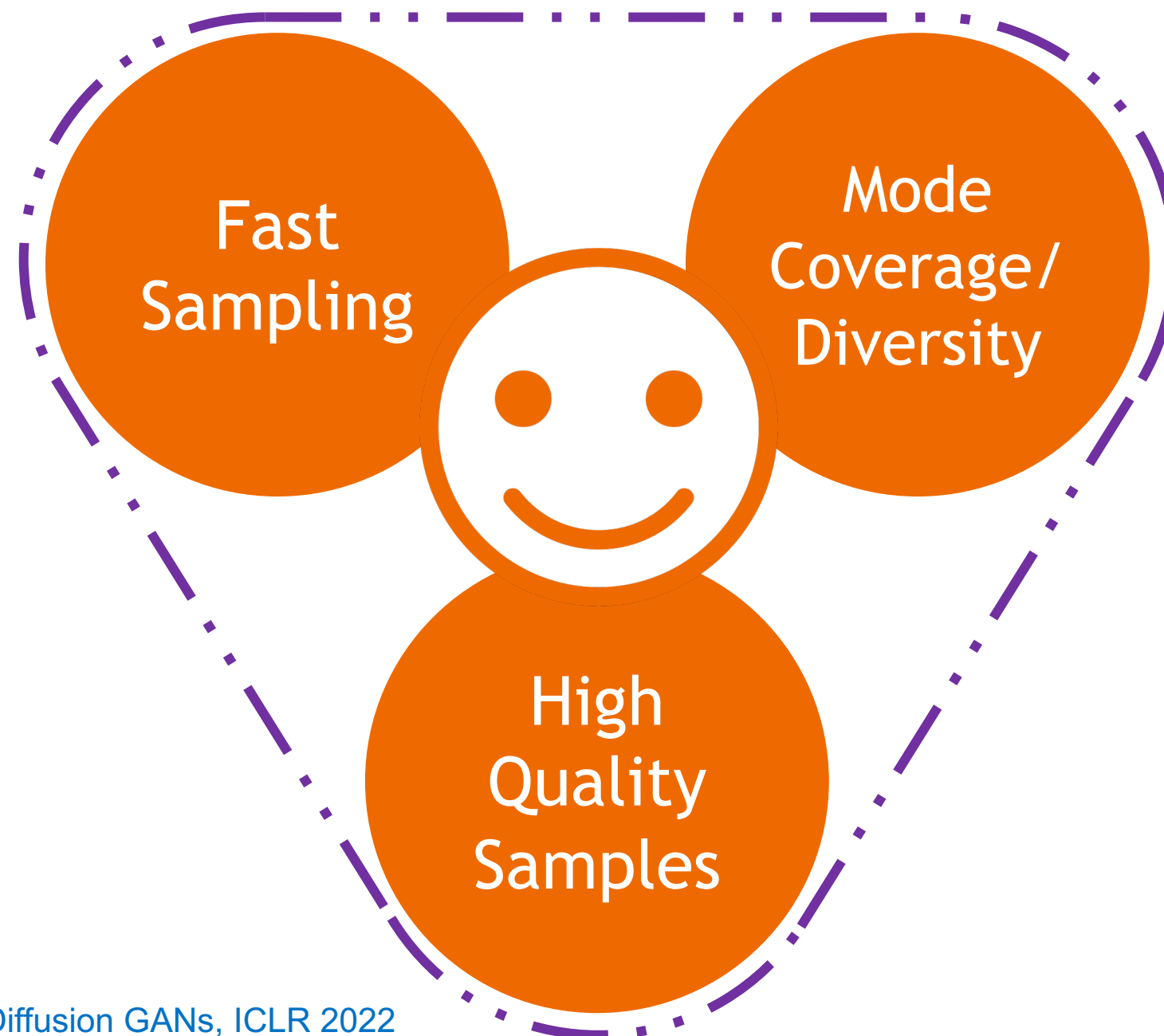




# What makes a good generative model?

The generative learning trilemma

Tackle the trilemma by accelerating diffusion models



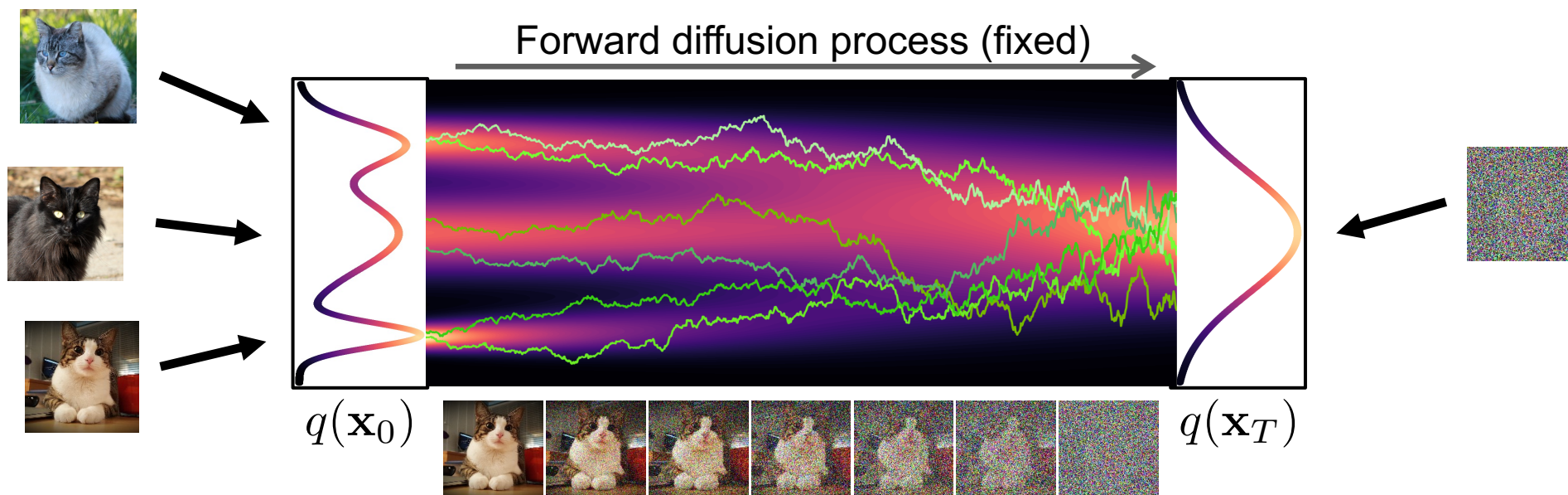
# Acceleration Techniques

Advanced ODE/SDE Solvers

Distillation Techniques

Low-dim. Diffusion Processes

Advanced Diffusion Processes



# Acceleration Techniques

Advanced  
ODE/SDE  
Solvers

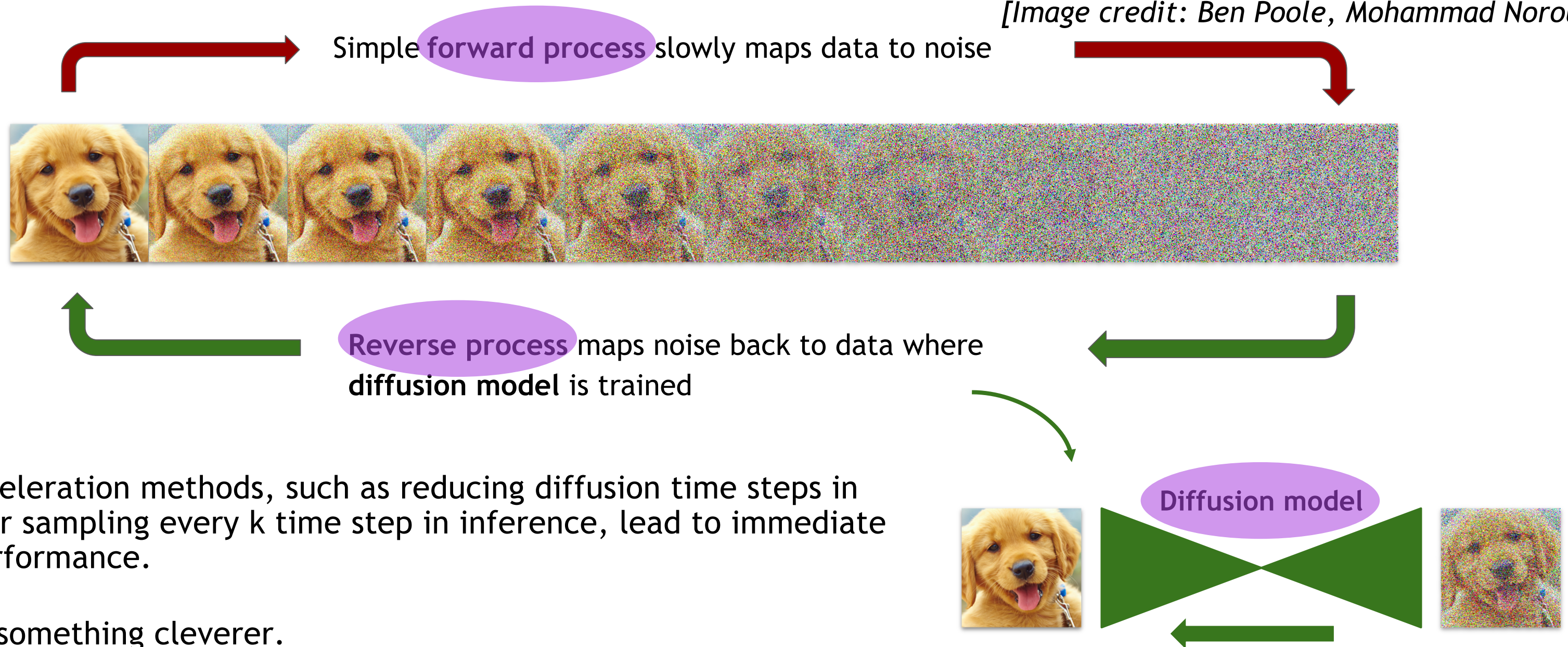
Distillation  
Techniques

Low-dim.  
Diffusion  
Processes

Advanced  
Diffusion  
Processes

# How to accelerate diffusion models?

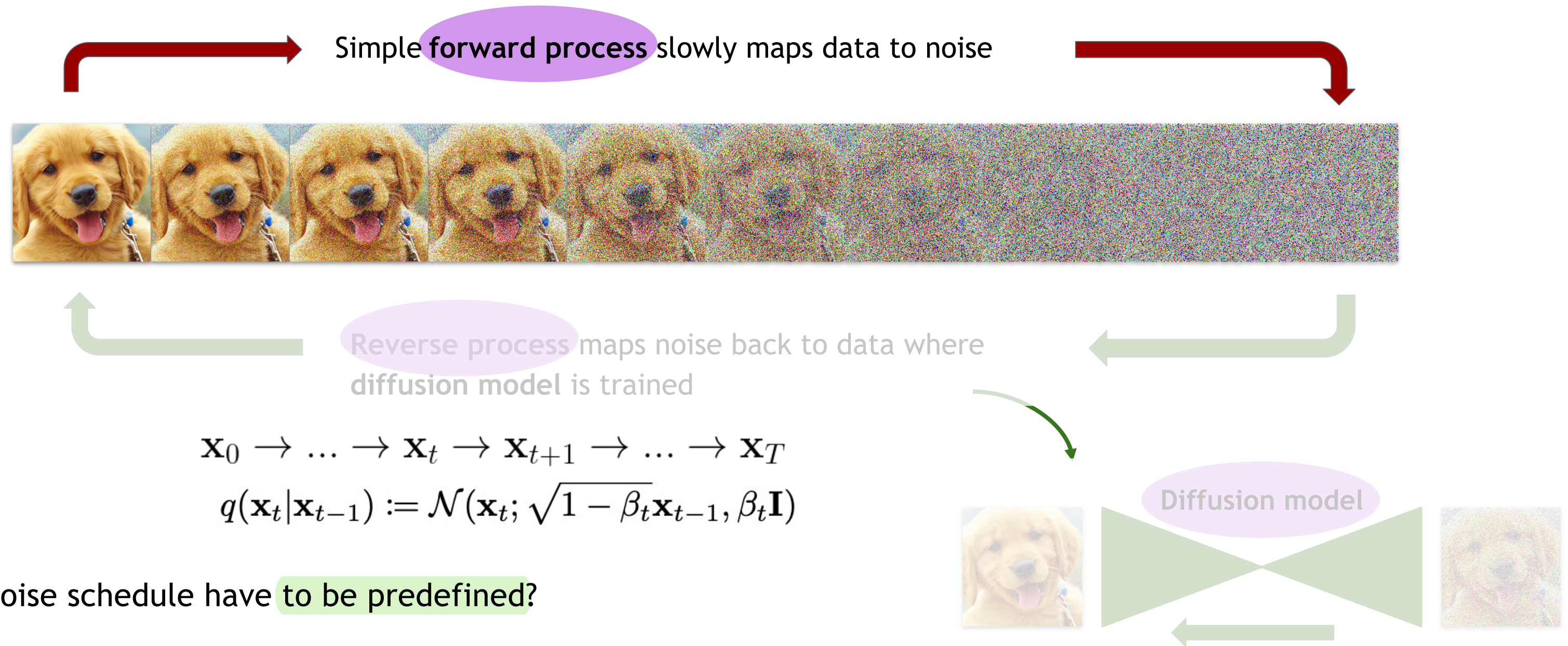
[Image credit: Ben Poole, Mohammad Norouzi]



- Naïve acceleration methods, such as reducing diffusion time steps in training or sampling every k time step in inference, lead to immediate worse performance.
- We need something cleverer.
- Given a limited number of functional calls, usually much less than 1000s, how to improve performance?

# (1/3) Advanced forward process

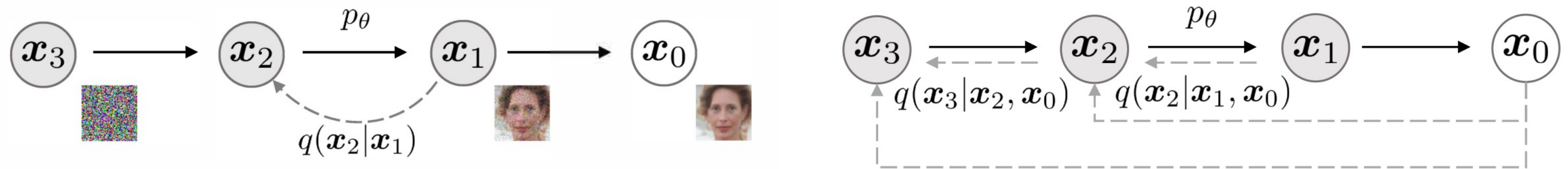
The reverse process will be changed accordingly



- Does the noise schedule have to be predefined?
- Does it have to be a Markovian process?
- Is there any faster mixing diffusion process?

# Denoising diffusion implicit models (DDIM)

Non-Markovian diffusion process



## Main Idea

Design a family of non-Markovian diffusion processes and corresponding reverse processes.

The process is designed such that the model can be optimized by the same surrogate objective as the original diffusion model.

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

Therefore, can take a pretrained diffusion model but with more choices of sampling procedure.

# Denoising diffusion implicit models (DDIM)

## How to define the non-Markovian forward process?

Two mean functions are parameterized by a simple linear combination of  $\mathbf{x}_t$  and epsilon via (4), (6), (7) of DDPM

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

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

Recall that the KL divergence in the variational upper bound can be written as:

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

since both are normal and have the same variance

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

If we assume loss weighting  $\lambda_t$  can be arbitrary values, the above formulation holds as long as We need to assume three things: A1 A2 A3

A1  $q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$  (make sure  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)} \epsilon$ )

If we define as follows:

A2 **Forward process:**  $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\sigma}_t^2 \mathbf{I})$ ,  $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = a\mathbf{x}_t + b\epsilon = a\mathbf{x}_t + b \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}}$  linear comb of  $\mathbf{x}_t$  and epsilon Replace the epsilon by the linear combination of  $\mathbf{x}_t$  and  $\mathbf{x}_0$

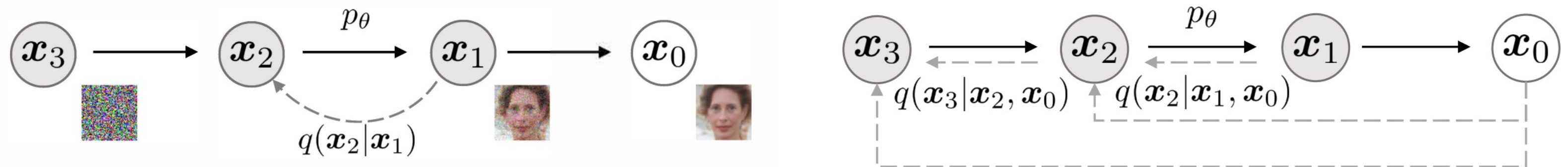
A3 **Reverse process:**  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \tilde{\sigma}_t^2 \mathbf{I})$ ,  $\mu_\theta(\mathbf{x}_t, t) = a\mathbf{x}_t + b\epsilon_\theta(\mathbf{x}_t, t) = a\mathbf{x}_t + b \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \hat{\mathbf{x}}_0}{\sqrt{1 - \bar{\alpha}_t}}$  predicted noise

A1 A2 A3 -->  $L_{t-1}$  still holds w/o assuming Markov.

**No need to specify  $q(\mathbf{x}_t | \mathbf{x}_{t-1})$  to be a Markovian process!** (assume  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \hat{\mathbf{x}}_0 + \sqrt{(1 - \bar{\alpha}_t)} \epsilon_\theta(\mathbf{x}_t, t)$ ) predicted clean sample given the predicted noise

# Denoising diffusion implicit models (DDIM)

Non-Markovian diffusion process



forward process posterior

For the forward process  $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\sigma}_t^2 \mathbf{I})$ ,  $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = a\mathbf{x}_t + b\epsilon = a\mathbf{x}_t + b \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}}$ , need to choose  $a, b$  such that  $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$

Define a family of forward processes that meets the above requirement:

for different values of  $\tilde{\sigma}_t$

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

Same surrogate objective if

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

But, this holds if  $\rightarrow$

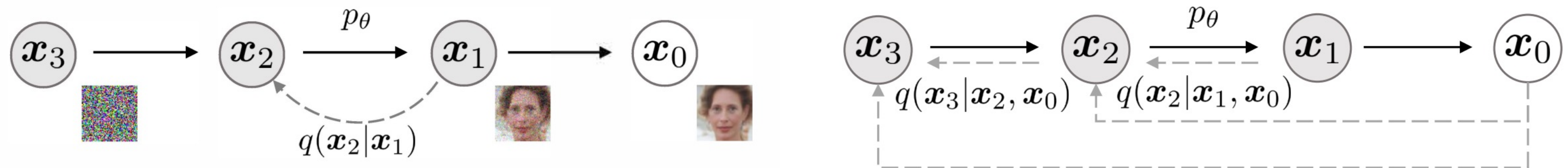
The corresponding reverse process is

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



# DDIM sampler

## Deterministic generative process



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

- DDIM sampler -  $\tilde{\sigma}_t^2 = 0, \forall t$ 
  - a deterministic generative process, with randomness from only  $t=T$ .  
starting sample

Train using DDPM.  
Sample using DDIM.

# ODE interpretation

## Deterministic generative process

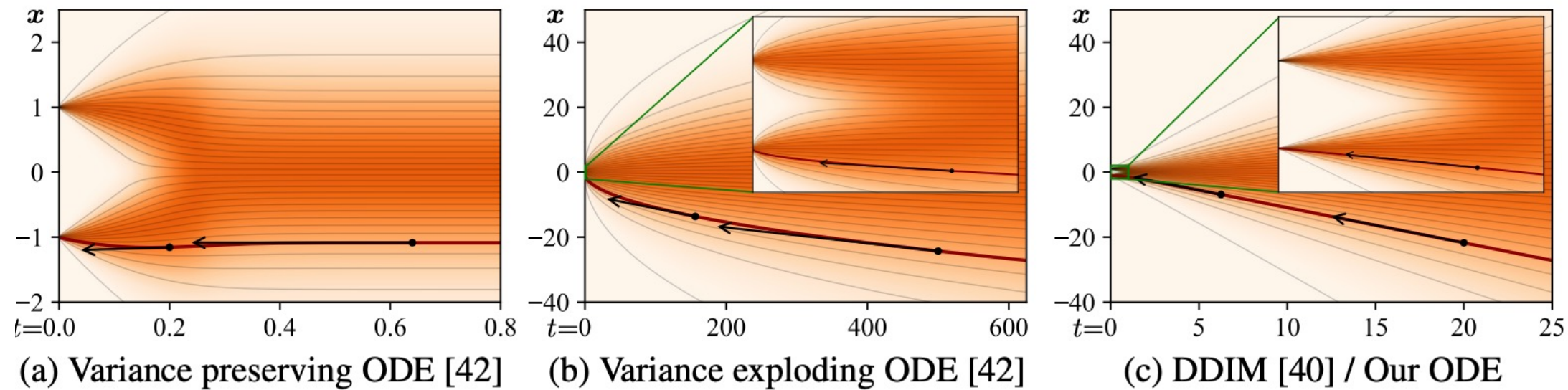


Fig. 3 in Karras

- DDIM sampler can be considered as an integration rule of the following ODE:

$$d\bar{\mathbf{x}}(t) = \epsilon_{\theta}^{(t)} \left( \frac{\bar{\mathbf{x}}(t)}{\sqrt{\eta^2 + 1}} \right) d\eta(t); \quad \bar{\mathbf{x}} = \mathbf{x} / \sqrt{\bar{\alpha}}, \eta = \sqrt{1 - \bar{\alpha}} / \sqrt{\bar{\alpha}}$$

SNR

- With the optimal model, the ODE is equivalent to a probability flow ODE of a “variance-exploding” SDE:

$$d\bar{\mathbf{x}} = -\frac{1}{2}g(t)^2 \nabla_{\bar{\mathbf{x}}} \log p_t(\bar{\mathbf{x}}) dt, \quad g(t) = \sqrt{\frac{d\eta^2(t)}{dt}}$$

- Sampling procedure can be different from standard Euler’s method: wrt.  $d\eta(t)$  vs wrt.  $dt$

better

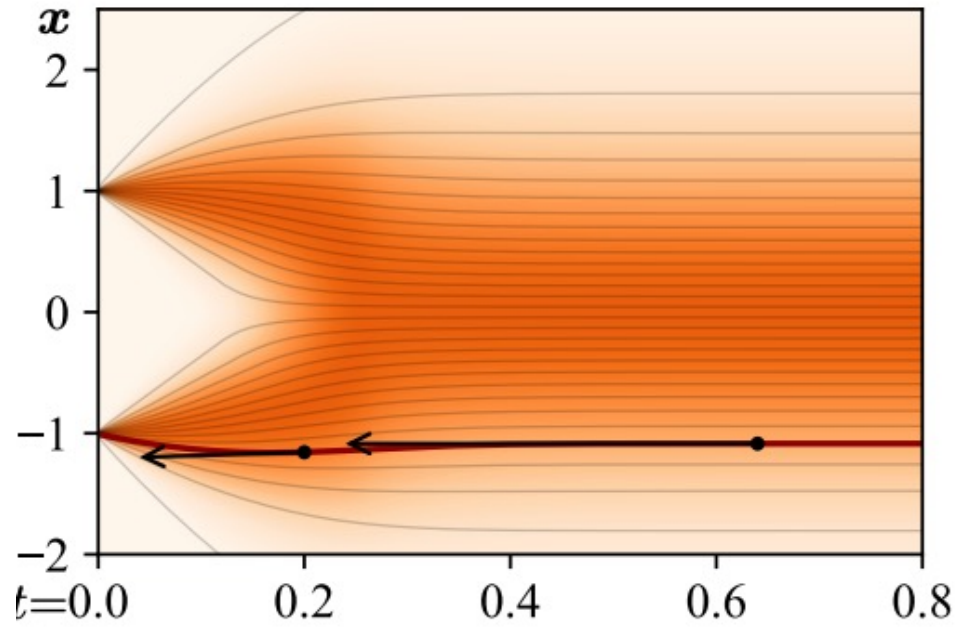
[Song et al., “Denoising Diffusion Implicit Models”, ICLR 2021.](#)

[Karras et al., “Elucidating the Design Space of Diffusion-Based Generative Models”, arXiv 2022.](#)

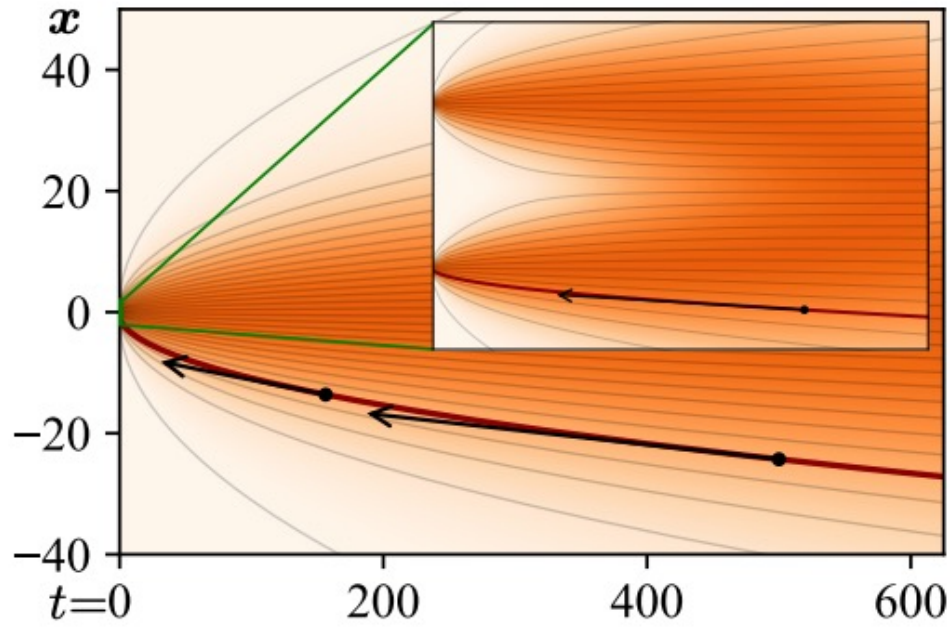
[Salimans & Ho, “Progressive distillation for fast sampling of diffusion models”, ICLR 2022.](#)

# DDIM sampler

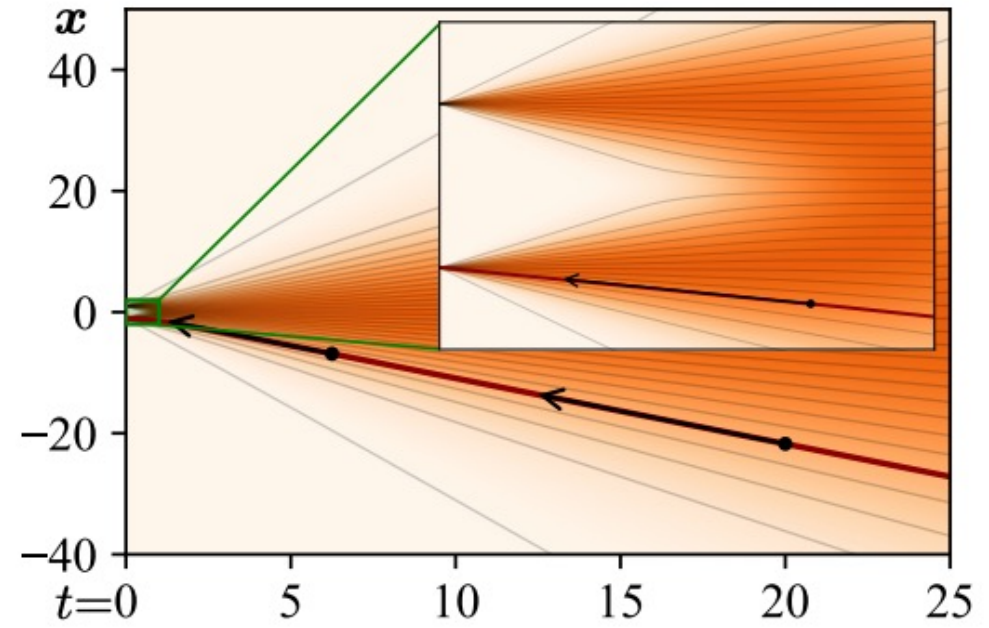
Faster & low curvature



(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.

[Song et al., “Denoising Diffusion Implicit Models”, ICLR 2021.](#)

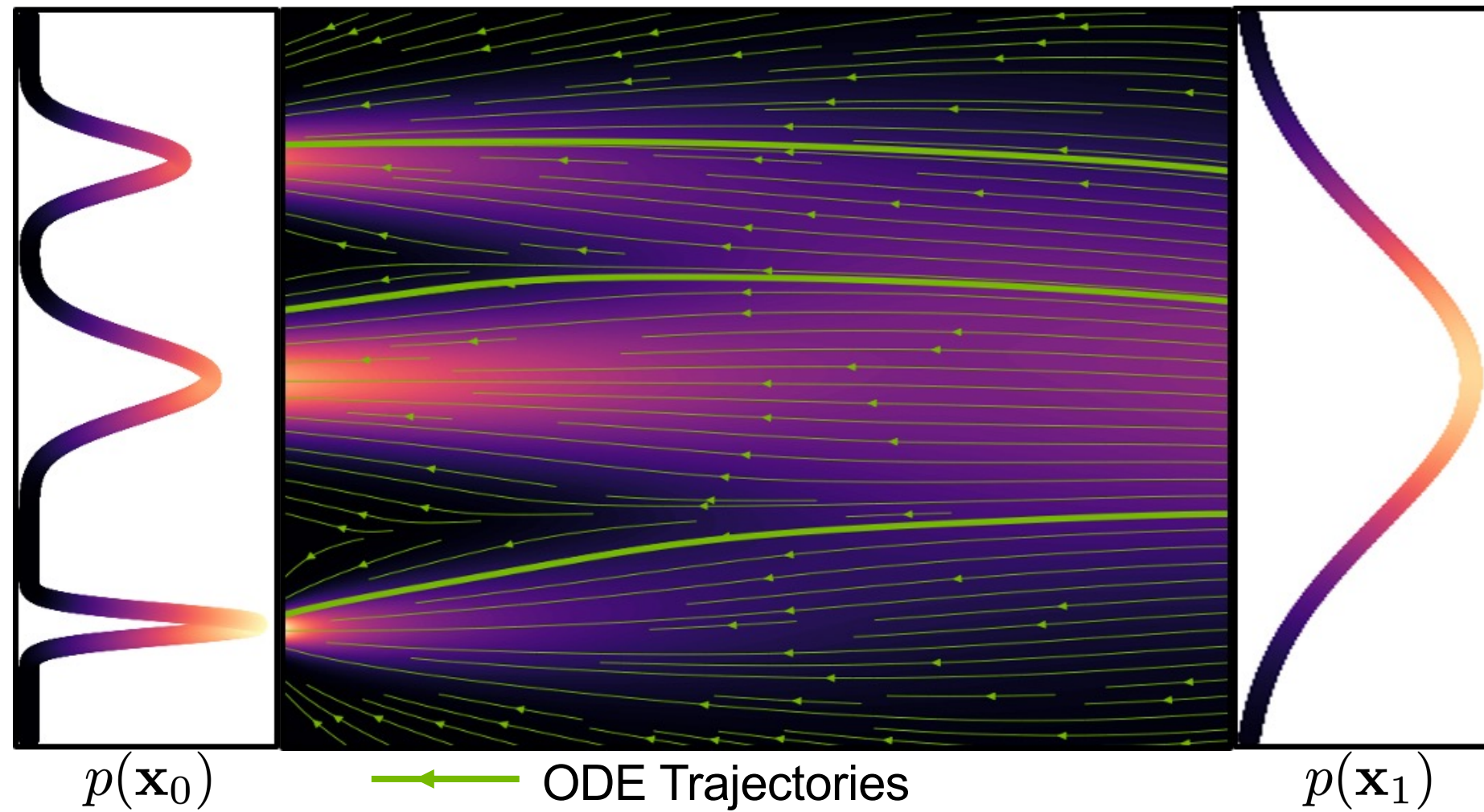
[Karras et al., “Elucidating the Design Space of Diffusion-Based Generative Models”, arXiv 2022.](#)

[Salimans & Ho, “Progressive distillation for fast sampling of diffusion models”, ICLR 2022.](#)

# Generative ODEs

Solve ODEs with as little function evaluations as possible

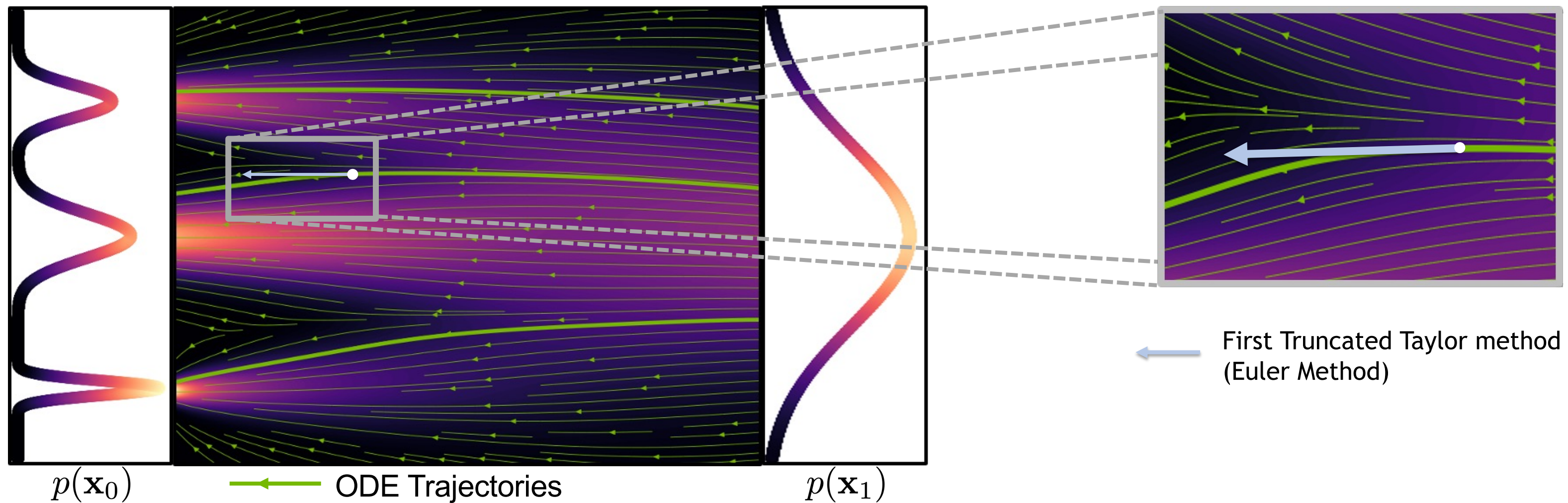
$$dx = \epsilon_{\theta}(x, t)dt$$



# Generative ODEs

Solve ODEs with as little function evaluations as possible

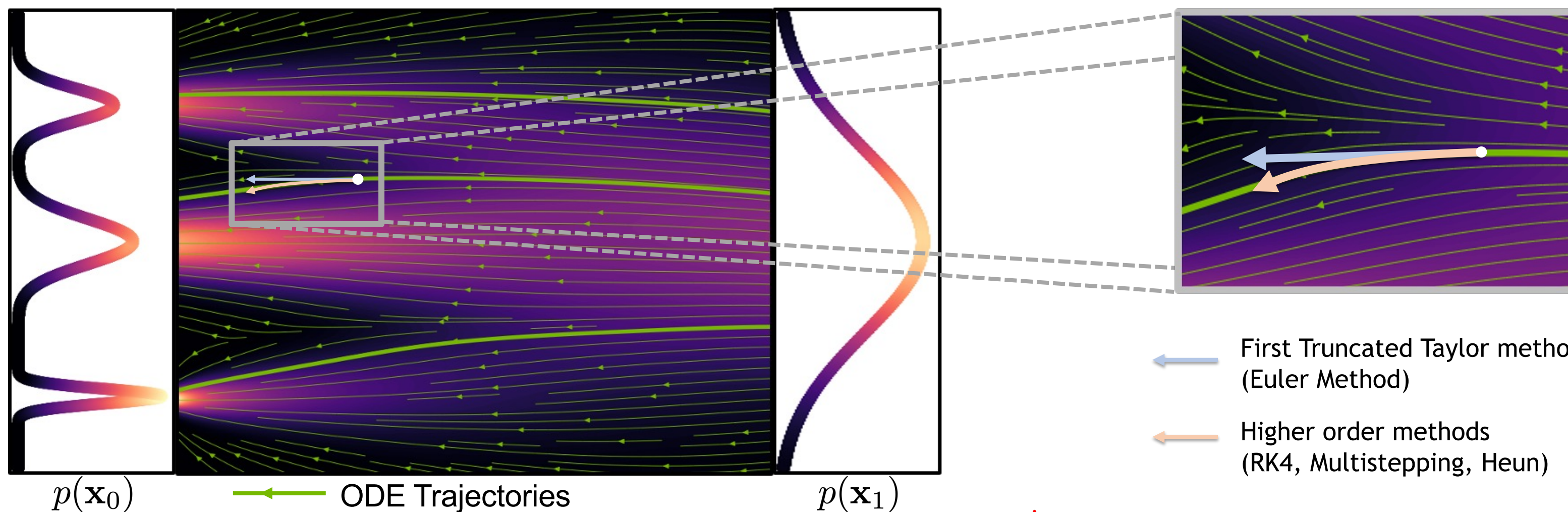
$$dx = \epsilon_{\theta}(x, t)dt$$



# Generative ODEs

Solve ODEs with as little function evaluations as possible

$$dx = \epsilon_{\theta}(x, t)dt$$



 Approximate the higher order terms numerically

# A Rich Body of Work on ODE/SDE Solvers for Diffusion Models

- Runge-Kutta adaptive step-size ODE solver:
  - [Song et al., “Score-Based Generative Modeling through Stochastic Differential Equations”, ICLR, 2021](#)
- Higher-Order adaptive step-size SDE solver:
  - [Jolicœur-Martineau et al., “Gotta Go Fast When Generating Data with Score-Based Models”, arXiv, 2021](#)
- Reparametrized, smoother ODE:
  - [Song et al., “Denoising Diffusion Implicit Models”, ICLR, 2021](#)
  - [Zhang et al., “gDDIM: Generalized denoising diffusion implicit models”, arXiv 2022](#)
- Higher-Order ODE solver with linear multisteping:
  - [Liu et al., “Pseudo Numerical Methods for Diffusion Models on Manifolds”, ICLR, 2022](#)
- Exponential ODE Integrators:
  - [Zhang and Chen, “Fast Sampling of Diffusion Models with Exponential Integrator”, arXiv, 2022](#)
  - [Lu et al., “DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps”, NeurIPS, 2022](#)
  - [Lu et al., “DPM-Solver++: Fast Solver for Guided Sampling of Diffusion Probabilistic Models”, NeurIPS 2022](#)
- Higher-Order ODE solver with Heun’s Method:
  - [Karras et al., “Elucidating the Design Space of Diffusion-Based Generative Models”, NeurIPS, 2022](#)
- Many more:
  - [Zhao et al., “UniPC: A Unified Predictor-Corrector Framework for Fast Sampling of Diffusion Models”, arXiv 2023](#)
  - [Shih et al., “Parallel Sampling of Diffusion Models”, arxiv 2023](#)
  - [Chen et al., “A Geometric Perspective on Diffusion Models”, arXiv 2023](#)

# Acceleration Techniques

Advanced  
ODE/SDE  
Solvers

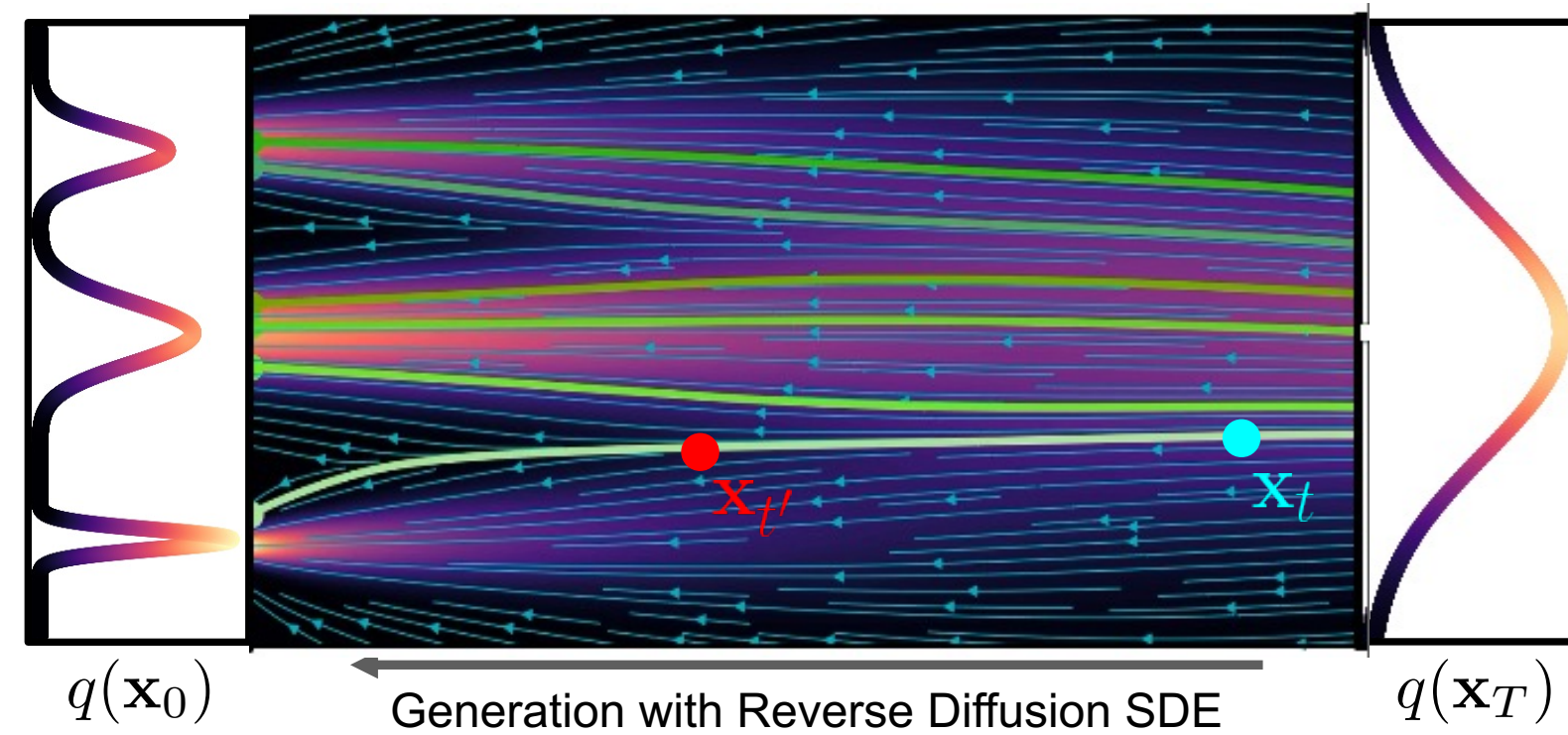
Distillation  
Techniques

Low-dim.  
Diffusion  
Processes

Advanced  
Diffusion  
Processes



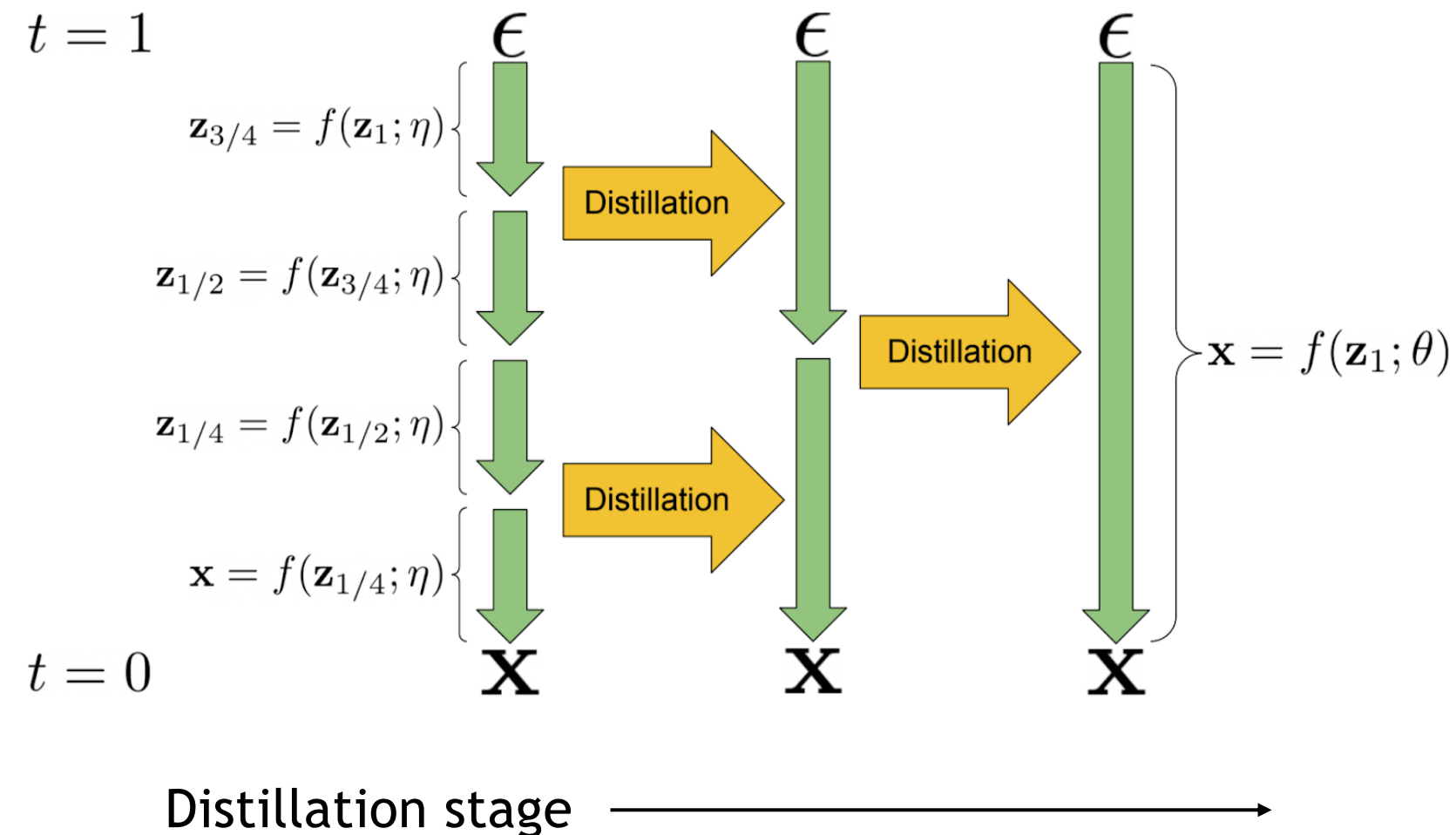
# ODE Distillation



Can we train a neural network to directly predict  $\mathbf{x}_{t'}$  given  $\mathbf{x}_t$  ?

# Progressive Distillation

- Distill a deterministic ODE sampler to the same model architecture.
- At each stage, a “student” model is learned to distill two adjacent sampling steps of the “teacher” model to one sampling step.
- At next stage, the “student” model from previous stage will serve as the new “teacher” model.



---

**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  $w(\cdot)$

**while** not converged **do**

$\mathbf{x} \sim \mathcal{D}$   $\triangleright$  Sample data

$t \sim U[0, 1]$   $\triangleright$  Sample time

$\epsilon \sim N(0, I)$   $\triangleright$  Sample noise

$\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$   $\triangleright$  Add noise to data

$\tilde{\mathbf{x}} = \mathbf{x}$   $\triangleright$  Clean data is target for  $\hat{\mathbf{x}}$

$\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$   $\triangleright$  log-SNR

$L_{\theta} = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_{\theta}(\mathbf{z}_t)\|_2^2$   $\triangleright$  Loss

$\theta \leftarrow \theta - \gamma \nabla_{\theta} L_{\theta}$   $\triangleright$  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  $w(\cdot)$

**Require:** Student sampling steps  $N$

**for**  $K$  iterations **do**

$\theta \leftarrow \eta$   $\triangleright$  Init student from teacher

**while** not converged **do**

$\mathbf{x} \sim \mathcal{D}$

$t = i/N, i \sim \text{Cat}[1, 2, \dots, N]$

$\epsilon \sim N(0, I)$

$\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$

# 2 steps of DDIM with teacher

$t' = t - 0.5/N, t'' = t - 1/N$

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

$\mathbf{z}_{t''} = \alpha_{t''} \hat{\mathbf{x}}_{\eta}(\mathbf{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}} (\mathbf{z}_{t'} - \alpha_{t'} \hat{\mathbf{x}}_{\eta}(\mathbf{z}_{t'}))$

$\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - (\sigma_{t''}/\sigma_t) \mathbf{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t) \alpha_t}$   $\triangleright$  Teacher  $\hat{\mathbf{x}}$  target

$\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$

$L_{\theta} = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_{\theta}(\mathbf{z}_t)\|_2^2$

$\theta \leftarrow \theta - \gamma \nabla_{\theta} L_{\theta}$

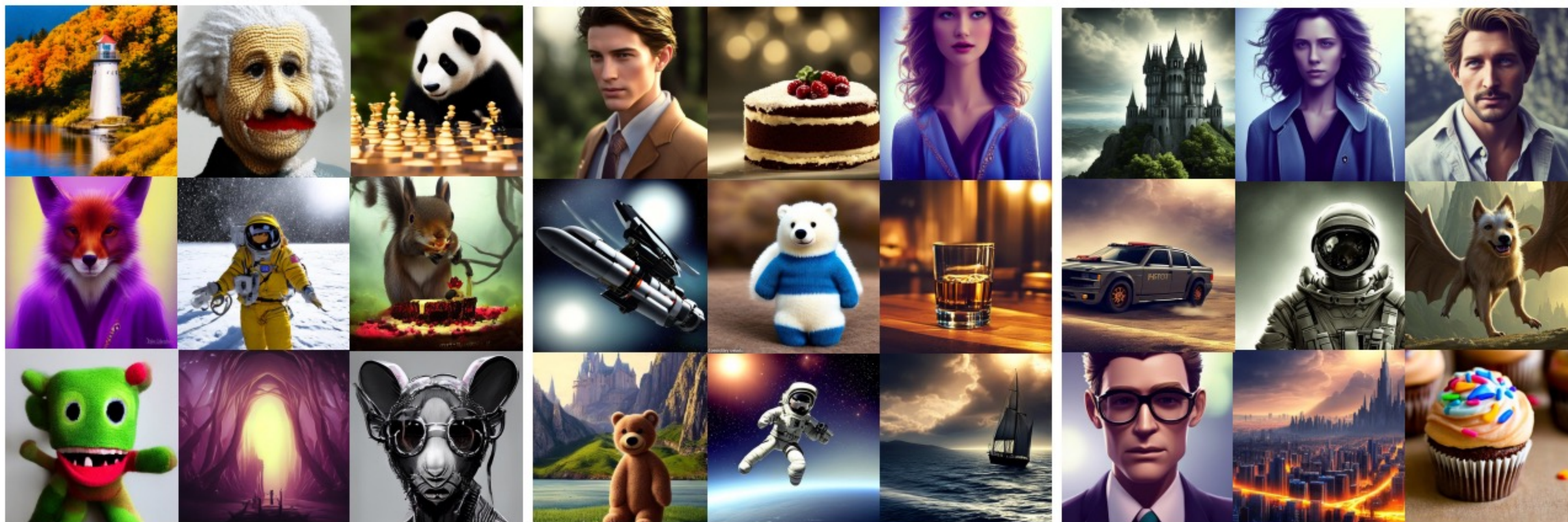
**end while**

$\eta \leftarrow \theta$   $\triangleright$  Student becomes next teacher

$N \leftarrow N/2$   $\triangleright$  Halve number of sampling steps

**end for**

# Progressive Distillation in Latent Space

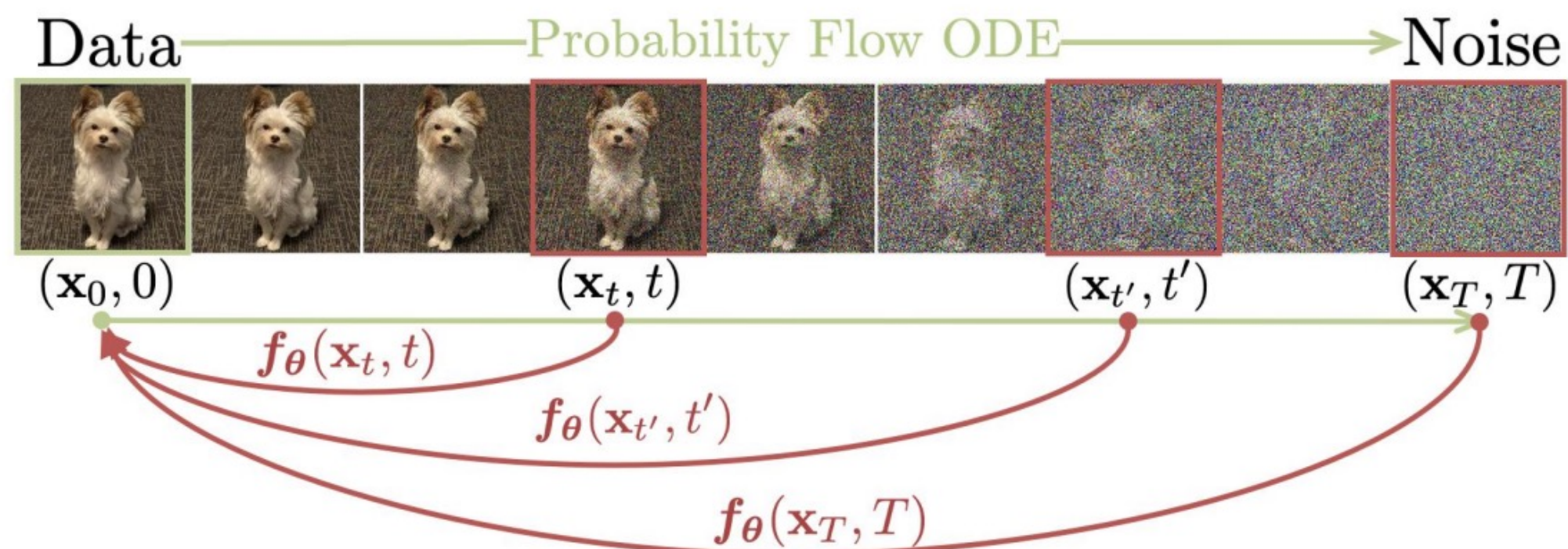


(a) 2 denoising steps

(b) 4 denoising steps

(c) 8 denoising steps

# Consistency Distillation



Points on the same trajectory should generate the same  $\mathbf{x}_0$

Assume  $f_\theta(\mathbf{x}_t, t)$  is the current estimation of  $\mathbf{x}_0$

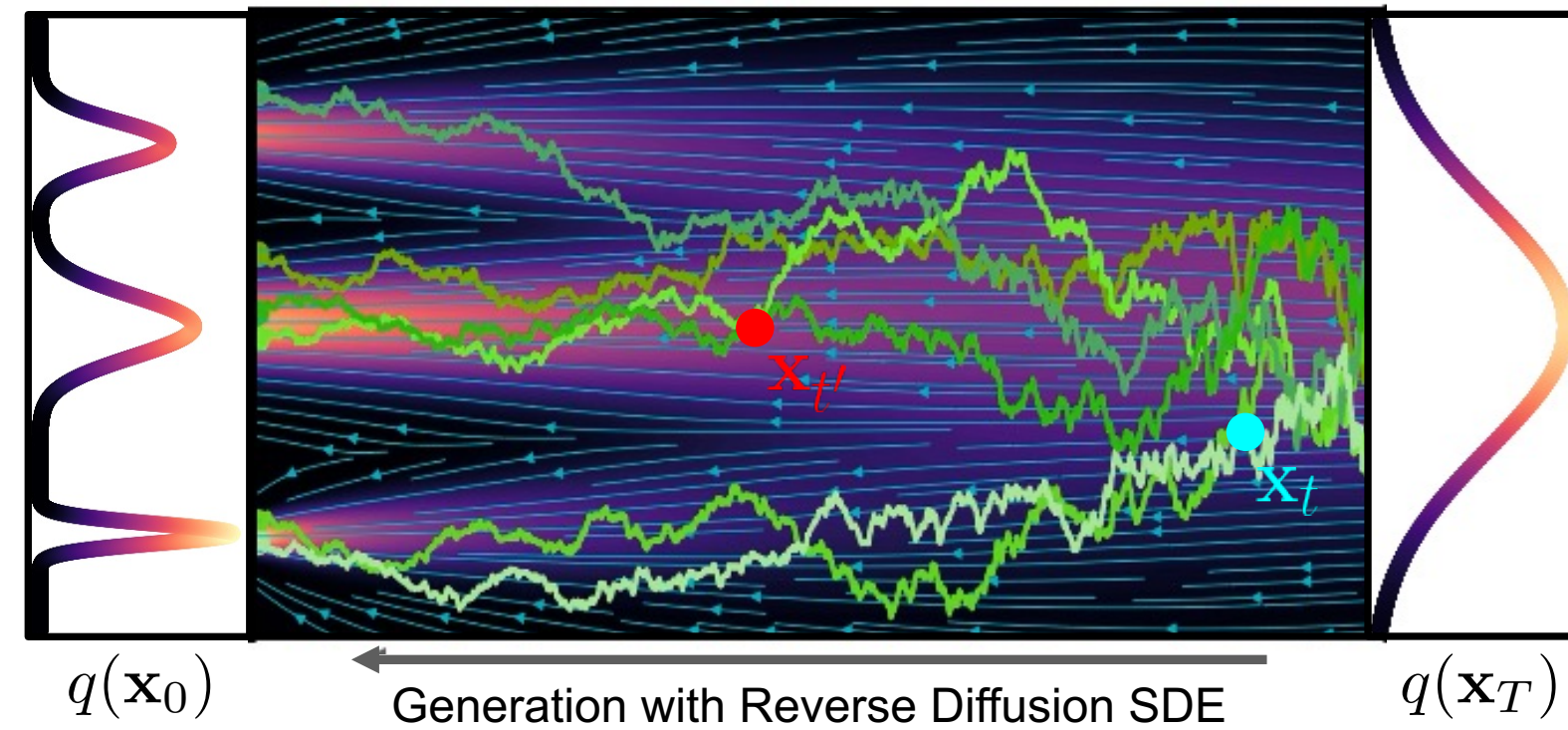
Basic idea:

- Find  $\mathbf{x}_t$  and  $\mathbf{x}_{t'}$  on a trajectory by solving generative ODE in  $[t, t']$

- Minimize:

$$\min_{\theta} \|f_{\text{EMA}}(\mathbf{x}_t, t) - f_\theta(\mathbf{x}_{t'}, t')\|_2^2$$

# SDE Distillation

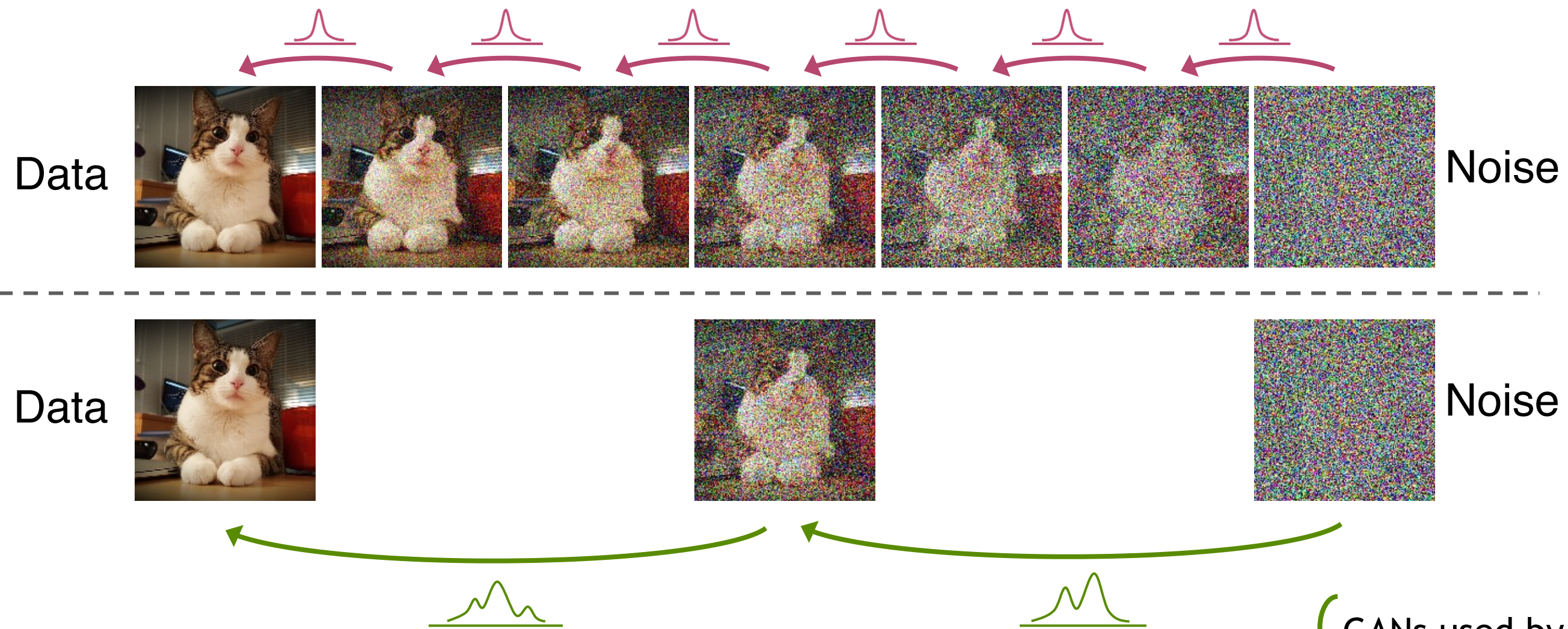


Can we train a neural network to directly predict **distribution of  $\mathbf{x}_{t'}$**  given  $\mathbf{x}_t$ ?

# Advanced Approximation of Reverse Process

Normal assumption in denoising distribution holds only for small step

## Denoising Process with Uni-modal Normal Distribution



Requires more complicated functional approximators!

GANs used by Xiao et al.

Energy-based models by Gao et al.

[Xiao et al., "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs", ICLR 2022.](#)

[Gao et al., "Learning energy-based models by diffusion recovery likelihood", ICLR 2021.](#)

# Training-based Sampling Techniques

- Knowledge distillation:
  - Luhman and Luhman, [Knowledge Distillation in Iterative Generative Models for Improved Sampling Speed](#), arXiv 2021
- Learned Samplers:
  - Watson et al., ["Learning Fast Samplers for Diffusion Models by Differentiating Through Sample Quality"](#), ICLR 2022
- Neural Operators:
  - Zheng et al., [Fast Sampling of Diffusion Models via Operator Learning](#), ICML 2023
- Wavelet Diffusion Models:
  - Phung et al., ["Wavelet Diffusion Models Are Fast and Scalable Image Generators"](#), CVPR 2023
- Distilled ODE Solvers:
  - Dockhorn et al., ["GENIE: Higher-Order Denoising Diffusion Solvers"](#), NeurIPS 2022



# Acceleration Techniques

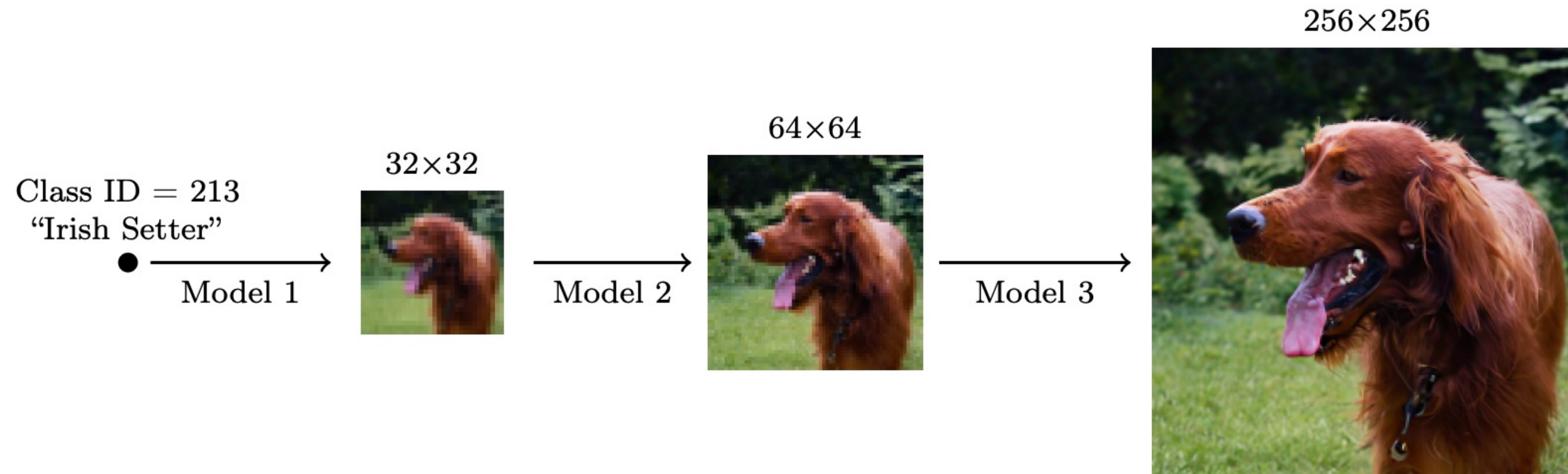
Advanced  
ODE/SDE  
Solvers

Distillation  
Techniques

Low-dim.  
Diffusion  
Processes

Advanced  
Diffusion  
Processes

# Cascaded Generation Pipeline



Cascaded Diffusion Models outperform Big-GAN in FID and IS and VQ-VAE2 in Classification Accuracy Score.

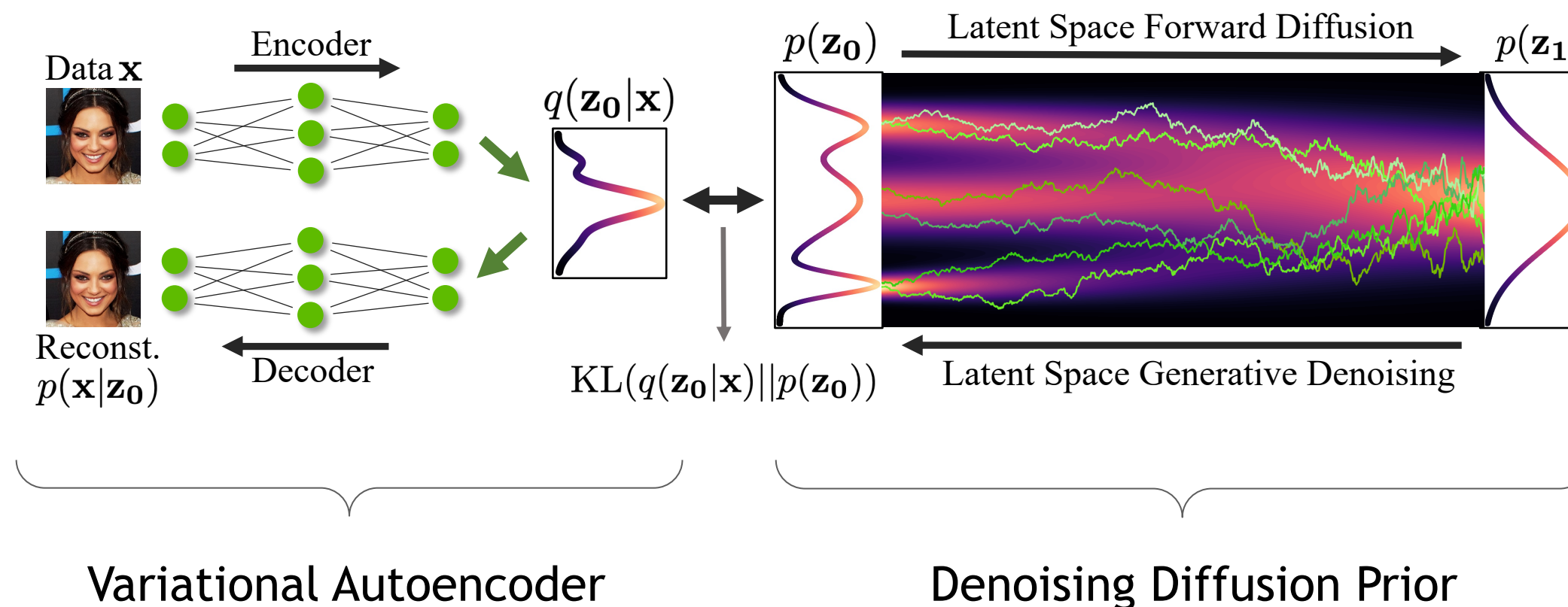
[Ho et al., "Cascaded Diffusion Models for High Fidelity Image Generation", 2021.](#)

[Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.](#)

[Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.](#)

# Latent Diffusion Models

Variational autoencoder + score-based prior



## Main Idea

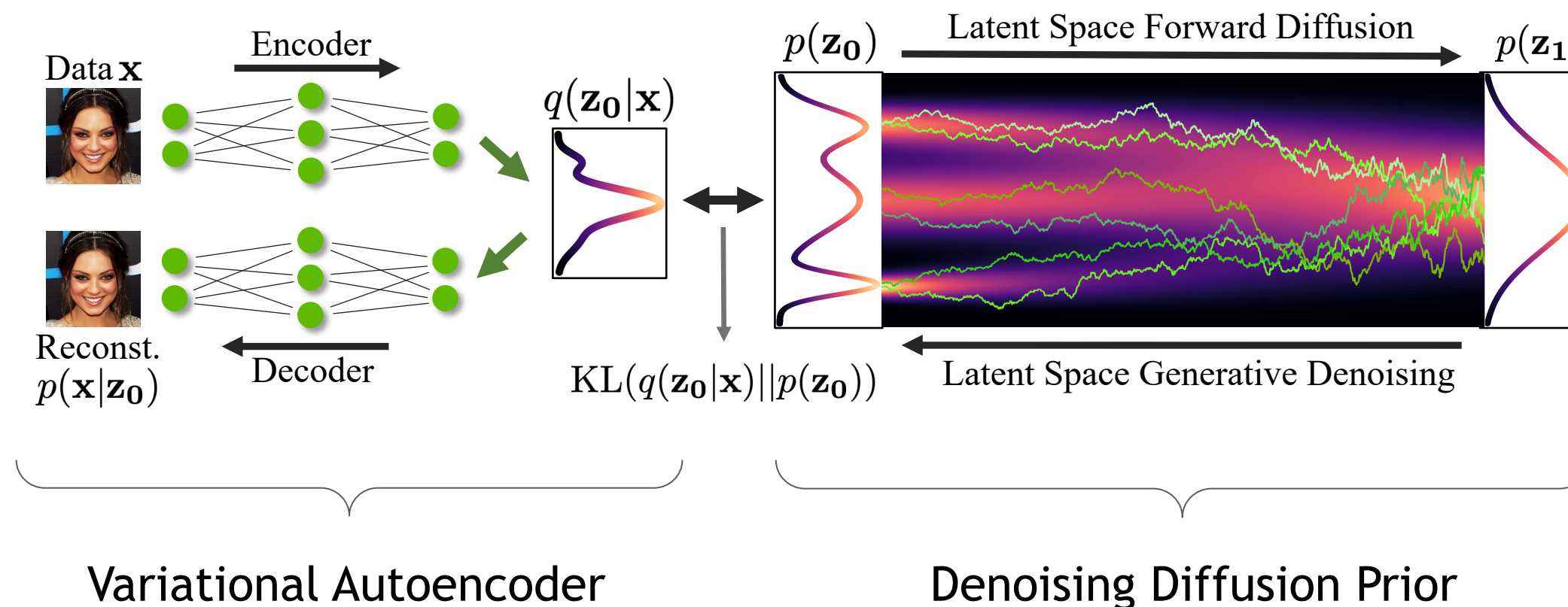
Encoder maps the input data to an embedding space

Denoising diffusion models are applied in the latent space



# Latent Diffusion Models

Variational autoencoder + score-based prior

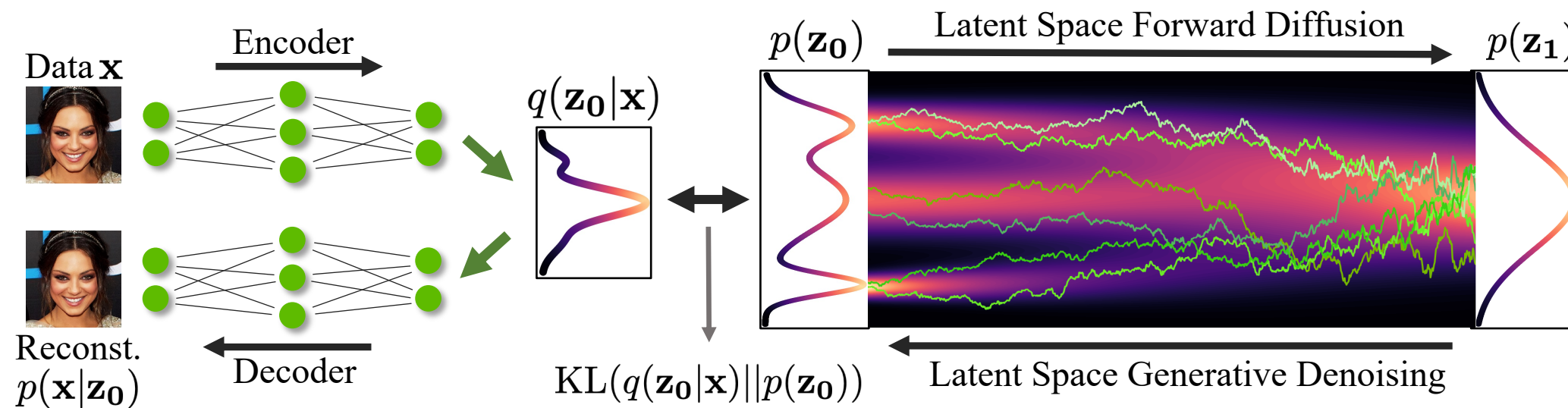


Advantages:

- (1) The distribution of latent embeddings close to Normal distribution → *Simpler denoising, Faster synthesis!*
- (2) Latent space → *More expressivity and flexibility in design!*
- (3) Tailored Autoencoders → *More expressivity, Application to any data type (graphs, text, 3D data, etc.) !*

# Latent Diffusion Models

## End-to-End Training objective

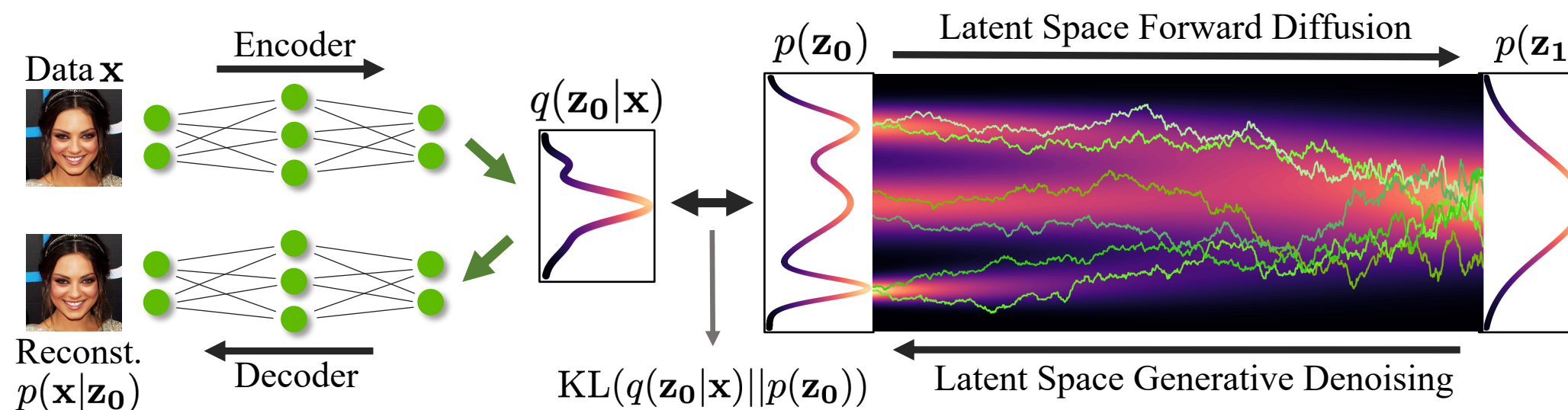


$$\begin{aligned}
 \mathcal{L}(\mathbf{x}, \phi, \theta, \psi) &= \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})} [-\log p_\psi(\mathbf{x}|\mathbf{z}_0)] + \text{KL}(q_\phi(\mathbf{z}_0|\mathbf{x}) || p_\theta(\mathbf{z}_0)) \\
 &= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})} [-\log p_\psi(\mathbf{x}|\mathbf{z}_0)]}_{\text{reconstruction term}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})} [\log q_\phi(\mathbf{z}_0|\mathbf{x})]}_{\text{negative encoder entropy}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})} [-\log p_\theta(\mathbf{z}_0)]}_{\text{cross entropy}}
 \end{aligned}$$

$$\begin{aligned}
 CE(q(\mathbf{z}_0|\mathbf{x}) || p(\mathbf{z}_0)) &= \underbrace{\mathbb{E}_{t \sim \mathcal{U}[0,1]}}_{\text{time sampling}} \left[ \underbrace{\frac{g(t)^2}{2}}_{\text{Forward diffusion}} \underbrace{\mathbb{E}_{q(\mathbf{z}_t, \mathbf{z}_0|\mathbf{x})}}_{\text{Diffusion kernel}} \left[ \underbrace{\|\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{z}_0) - \nabla_{\mathbf{z}_t} \log p(\mathbf{z}_t)\|_2^2}_{\text{Trainable score function}} \right] \right] + \underbrace{\frac{D}{2} \log(2\pi e \sigma_0^2)}_{\text{Constant}}
 \end{aligned}$$

# Latent Diffusion Models

## Two-stage Training



The seminal work from Rombach et al. CVPR 2022:

- Two stage training: train autoencoder first, then train the diffusion prior
- Focus on compression without of any loss in reconstruction quality
- Demonstrated the expressivity of latent diffusion models on many conditional problems

The efficiency and expressivity of latent diffusion models + open-source access fueled a large body of work in the community

# Additional Reading

More on low-dimensional diffusion models:

- Sinha et al., ["D2C: Diffusion-Denoising Models for Few-shot Conditional Generation"](#), NeurIPS 2021
- Daras et al., ["Score-Guided Intermediate Layer Optimization: Fast Langevin Mixing for Inverse Problems"](#), ICML 2022
- Zhang et al., ["Dimensionality-Varying Diffusion Process"](#), arXiv 2022.

# Acceleration Techniques

Advanced  
ODE/SDE  
Solvers

Distillation  
Techniques

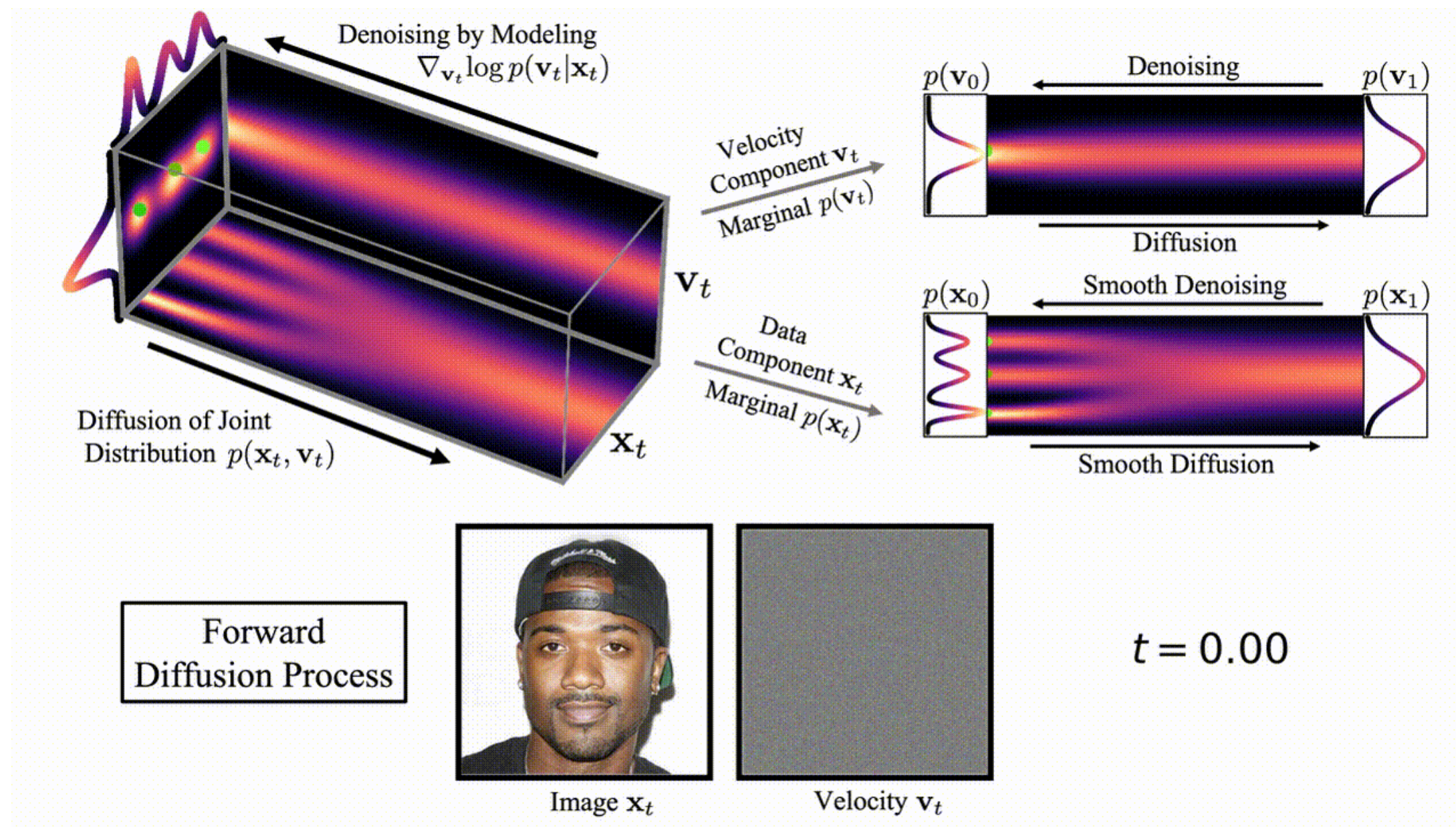
Low-dim.  
Diffusion  
Processes

Advanced  
Diffusion  
Processes



# “Momentum-based” diffusion

Introduce a velocity variable and run diffusion in extended space



Main idea: Inject noise only into  $\mathbf{v}_t$ , faster mixing through Hamiltonian component!

# Additional Reading

- Schrödinger Bridge:
  - Bortoli et al., ["Diffusion Schrödinger Bridge"](#), NeurIPS 2021
  - Chen et al., ["Likelihood Training of Schrödinger Bridge using Forward-Backward SDEs Theory"](#), ICLR 2022
- Diffusion Processes on Manifolds:
  - Bortoli et al., ["Riemannian Score-Based Generative Modelling"](#), NeurIPS 2022
- Cold Diffusion:
  - Bansal et al., ["Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise"](#), arXiv 2022
- Diffusion for Corrupted Data:
  - Daras et al., ["Soft Diffusion: Score Matching for General Corruptions"](#), TMLR 2023
  - Delbracio and Milanfar, ["Inversion by Direct Iteration: An Alternative to Denoising Diffusion for Image Restoration"](#), arXiv 2023
  - Luo et al., ["Image Restoration with Mean-Reverting Stochastic Differential Equations"](#), ICML 2023
  - Liu et al., ["I2SB: Image-to-Image Schrödinger Bridge"](#), ICML 2023
- Blurring Diffusion Process:
  - Hoogeboom and Salimans, ["Blurring Diffusion Models"](#), ICLR 2023
  - Rissanen et al., ["Generative Modelling With Inverse Heat Dissipation"](#), ICLR 2023

# Outline

Part (1): Denoising Diffusion Probabilistic Models

Part (2): Score-based Generative Modeling with Differential Equations

Part (3): Accelerated Sampling

**Part (4): Conditional Generation and Guidance**

# Impressive Conditional Diffusion Models

## Text-to-image generation

### DALL·E 2

*“a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese”*



### IMAGEN

*“A photo of a raccoon wearing an astronaut helmet, looking out of the window at night.”*



# Conditioning and Guidance Techniques



Explicit  
Conditions

Classifier  
Guidance

Classifier-  
free  
Guidance

# Conditioning and Guidance Techniques



Explicit  
Conditions

Classifier  
Guidance

Classifier-  
free  
Guidance

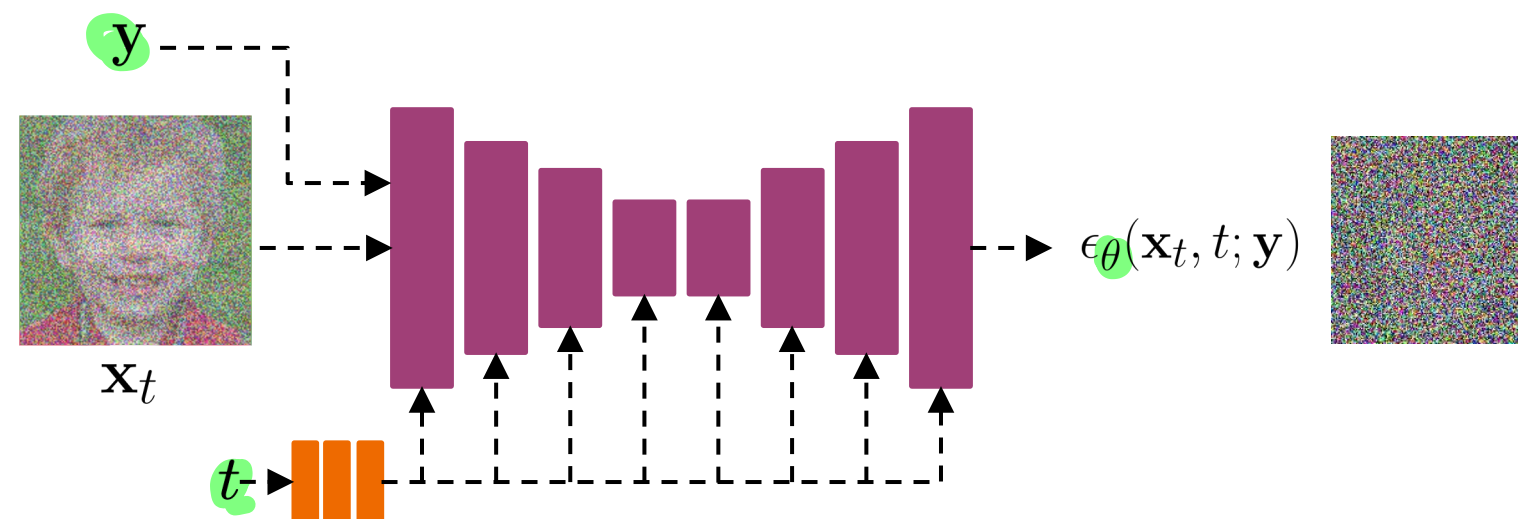
# Explicit Conditional Training

Conditional sampling can be considered as training  $p(\mathbf{x}|\mathbf{y})$  where  $\mathbf{y}$  is the input conditioning (e.g., text) and  $\mathbf{x}$  is generated output (e.g., image)

Train the score model for  $\mathbf{x}$  conditioned on  $\mathbf{y}$  using:

$$\mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim p_{\text{data}}(\mathbf{x}, \mathbf{y})} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \mathbb{E}_{t \sim \mathcal{U}[0, T]} \|\epsilon_{\theta}(\mathbf{x}_t, t; \mathbf{y}) - \epsilon\|_2^2$$

The conditional score is simply a U-Net with  $\mathbf{x}_t$  and  $\mathbf{y}$  together in the input.



# Conditioning and Guidance Techniques



Explicit  
Conditions

Classifier  
Guidance

Classifier-  
free  
Guidance



# Classifier Guidance: Bayes' Rule in Action

Recall that when sampling from a conditional model  $p(\mathbf{x}|\mathbf{y})$  we basically need an estimation of  $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y})$

P34

$$-\frac{E_{\theta}(\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}))}{\partial t}$$

Using Bayes' rule, we have:

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log \frac{p(\mathbf{y}|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{y})} = \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$$

Classifier gradient                      Score model

↓    ↓

Introduce a guidance scale ( $w$ ):

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) \approx w \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$$

$$\left( p(\mathbf{x}_t|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x}_t)^w p(\mathbf{x}_t) \right)$$

[Song et al., "Score-Based Generative Modeling through Stochastic Differential Equations", ICLR, 2021](#)

[Nie et al., "Controllable and Compositional Generation with Latent-Space Energy-Based Models", NeurIPS 2021](#)

[Dhariwal and Nichol, "Diffusion models beat GANs on image synthesis", NeurIPS 2021.](#)

# Conditioning and Guidance Techniques



Explicit  
Conditions

Classifier  
Guidance

Classifier-  
free  
Guidance

# Classifier-free guidance

Get guidance by Bayes' rule on conditional diffusion models

- Recall that classifier guidance requires training a classifier.

- Using Bayes' rule again:

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \quad \mathbf{y}: \text{label}$$

↑
↑  
 Conditional diffusion model      Unconditional diffusion model

- Instead of training an additional classifier, get an “implicit classifier” by jointly training a conditional and unconditional diffusion model. In practice, the conditional and unconditional models are trained together by randomly dropping the condition of the diffusion model at certain chance.

- The modified score with this implicit classifier included is:

$$\begin{aligned}
 (1+w) \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) &= (1+w) (\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)) + \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \\
 &= (1+w) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) - w \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)
 \end{aligned}$$

*from classifier guidance*

$$\hat{\epsilon}_\theta(z_\lambda, c) = (1+w)\epsilon_\theta(z_\lambda, c) - w\epsilon_\theta(z_\lambda)$$

# Classifier-free guidance

Trade-off for sample quality and sample diversity



Non-guidance

$\omega = 0$  high temperature

$\omega = 1$

$\omega = 3$

low temperature

FID ↓ → sample fidelity

ISF → more fidelity

Large guidance weight ( $\omega$ ) usually leads to better individual sample quality but less sample diversity.

# Summary

We reviewed diffusion fundamentals in 4 parts:

- Discrete-time diffusion models
- Continuous-time diffusion models
- Accelerated sampling from diffusion models
- Guidance and conditioning.

Next, we will review different applications and use cases of diffusion models after a break.


# Small Announcement

 Pinned Tweet



**Arash Vahdat**  **#CVPR2023** 

@ArashVahdat

 We are looking for candidates with a strong background in generative learning on text, image, video, graph & 3D data to join our team [@nvidia](#). We strive for high-impact forward-looking research.

Apply via:

[bit.ly/3P9yxMC](https://bit.ly/3P9yxMC)

[bit.ly/43ZgpZW](https://bit.ly/43ZgpZW)

# Thanks!

*"That's all Folks!"*

<https://cvpr2023-tutorial-diffusion-models.github.io/>



@ArashVahdat

