

Indian Institute of Information Technology, Allahabad



Diffusion Model

By

Dr. Satish Kumar Singh & Dr. Shiv Ram Dubey

Computer Vision and Biometrics Lab

Department of Information Technology

Indian Institute of Information Technology, Allahabad



TEAM

Computer Vision and Biometrics Lab (CVBL)

Department of Information Technology

Indian Institute of Information Technology Allahabad

Course Instructors

Dr. Satish Kumar Singh, Associate Professor, IIIT Allahabad (Email: sk.singh@iiita.ac.in)

Dr. Shiv Ram Dubey, Assistant Professor, IIIT Allahabad (Email: srdubey@iiita.ac.in)

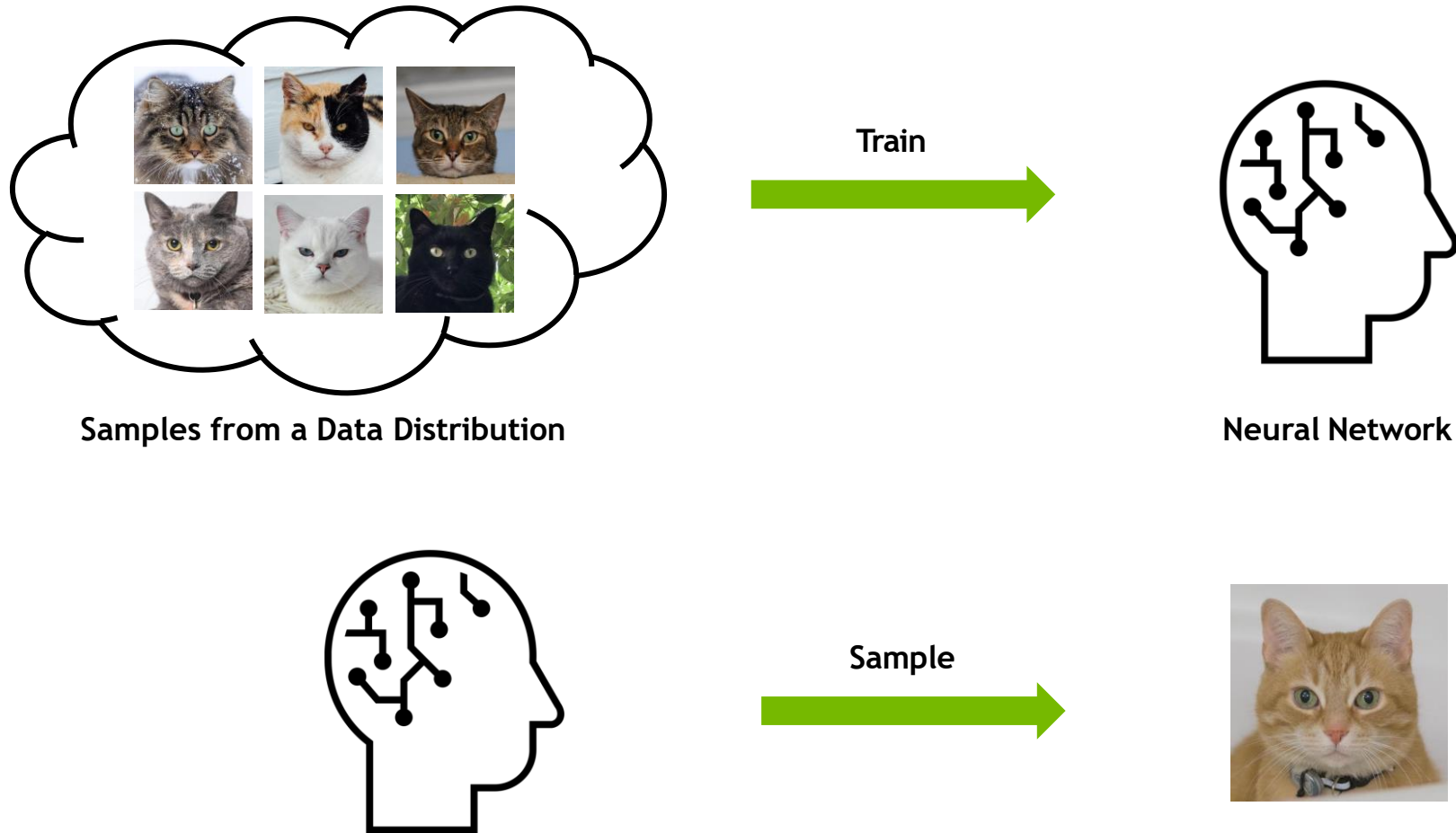


DISCLAIMER

The content (text, image, and graphics) used in this slide are adopted from many sources for Academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

Deep Generative Learning

Learning to generate data



Application (1): Content Generation

StyleGAN3 example images

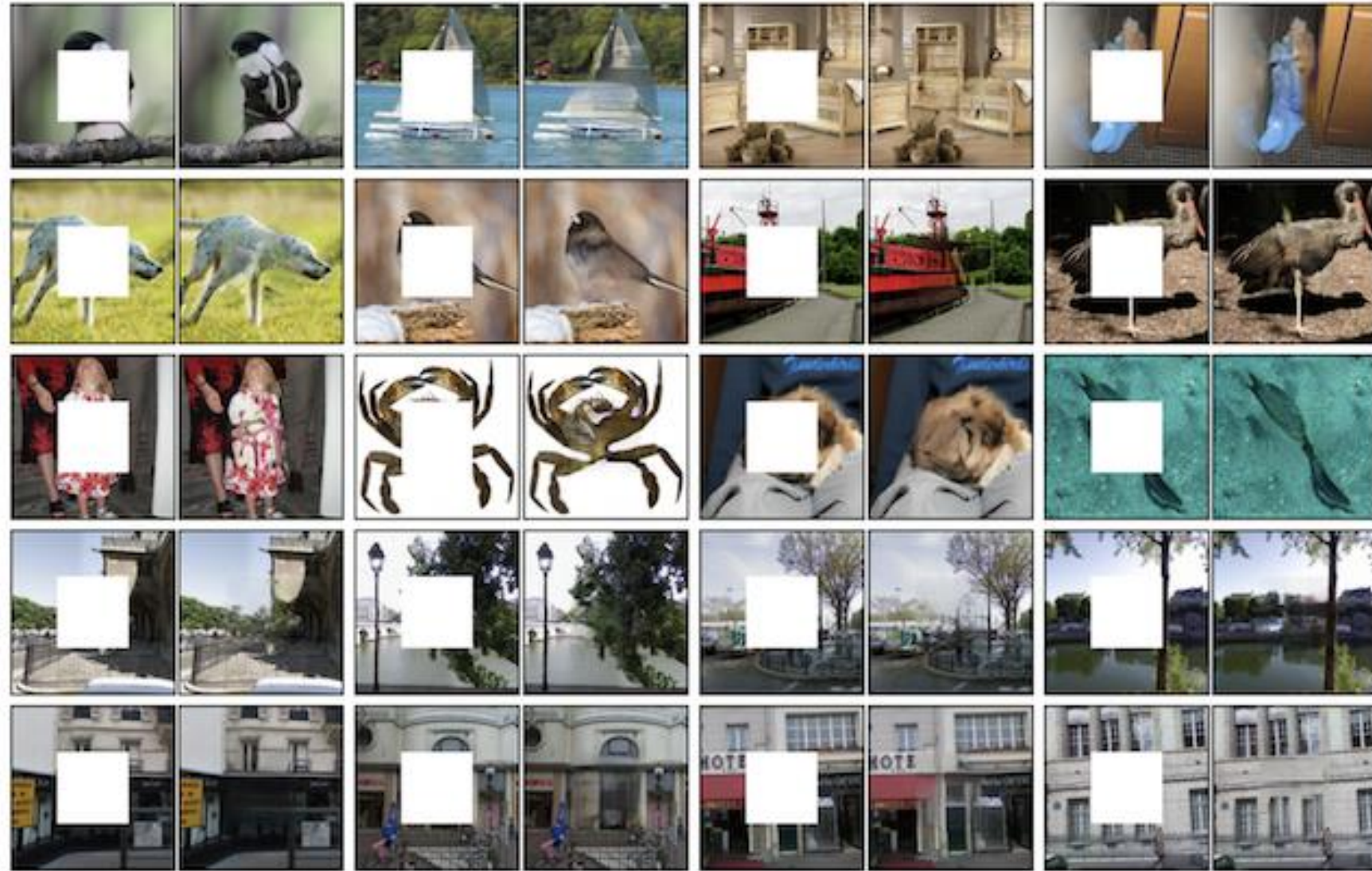


Application (2): Image to Image Translation

Photo Inpainting

Example of GAN-Generated
Photograph Inpainting Using
Context Encoders.

Taken from Context Encoders:
Feature Learning by Inpainting
describe the use of GANs,
specifically Context Encoders, 2016.



Application (3): Representation Learning

Learning from limited labels



The Landscape of Deep Generative Learning

A landscape of mountains at sunset. The sky is a gradient of blue, orange, and yellow. The mountains are silhouetted against the bright sky, with some peaks appearing as dark shapes. The overall mood is serene and expansive.

Variational
Autoencoders

Autoregressive
Models

Normalizing
Flows

Generative
Adversarial
Networks

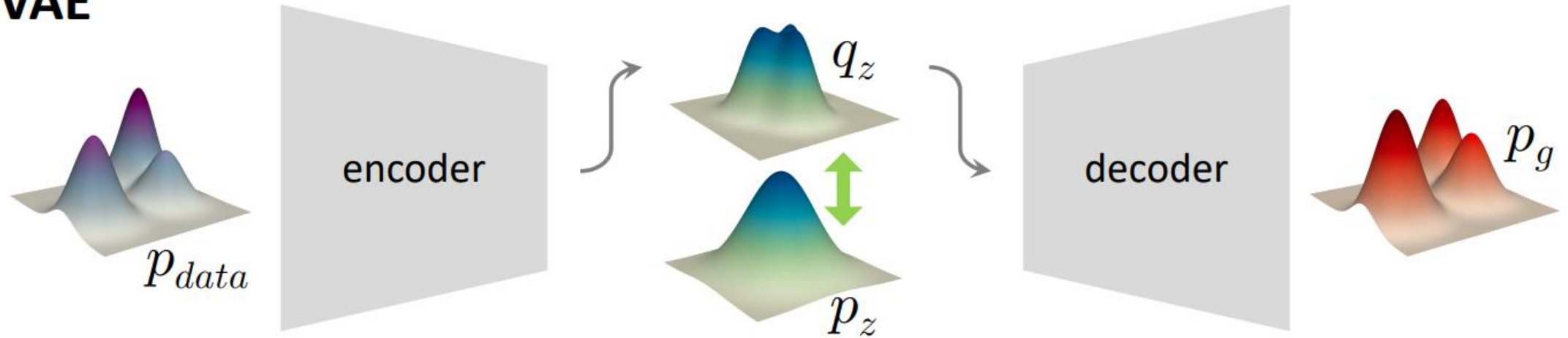
Energy-based
Models

Denoising
Diffusion Models

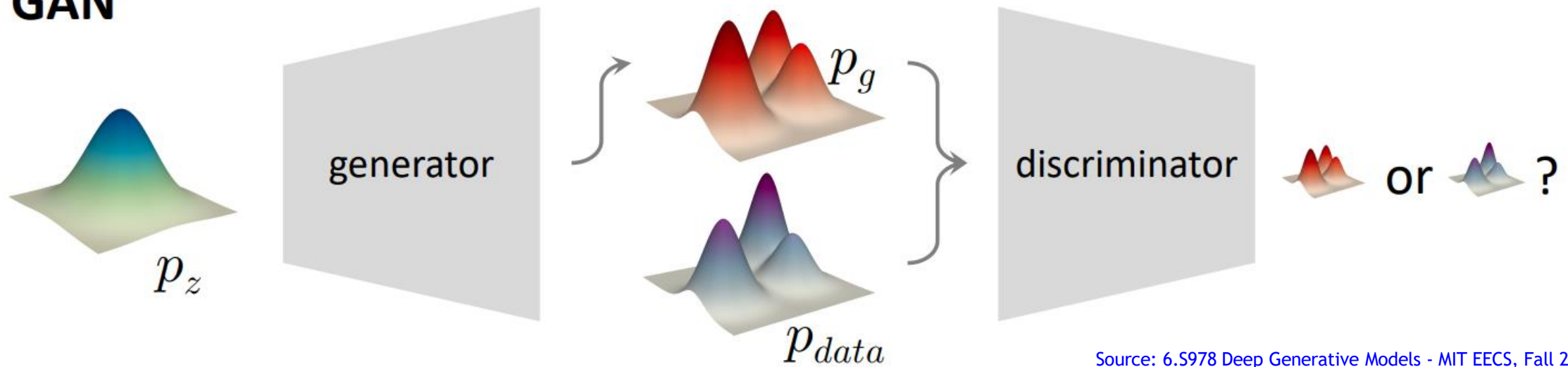


Generative Models

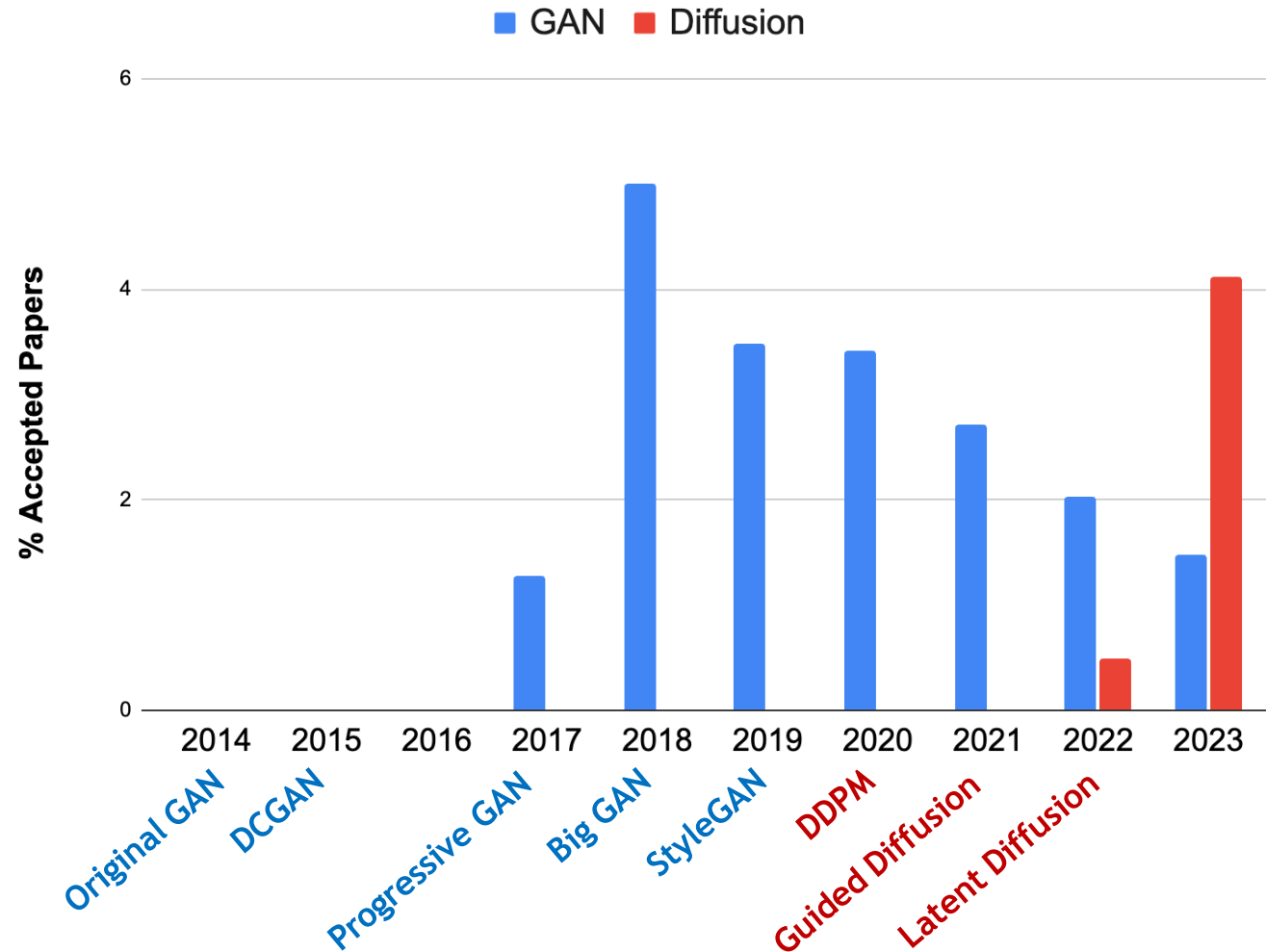
VAE



GAN



As per CVPR



*Disclaimer: search is performed based on paper titles for counting the number of papers in each topic, hence the statistics are likely to be biased.

Source: [CVPR2023-tutorial-diffusion-models](#)

Denoising Diffusion Models

Emerging as powerful generative models, outperforming GANs



[“Diffusion Models Beat GANs on Image Synthesis”](#)
Dhariwal & Nichol, OpenAI, 2021



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

[Source: CVPR2022-tutorial-diffusion-models](#)

Text-to-Image Generation: Successful Applications

DALL·E 2

“a teddy bear on a skateboard in times square”



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

Imagen

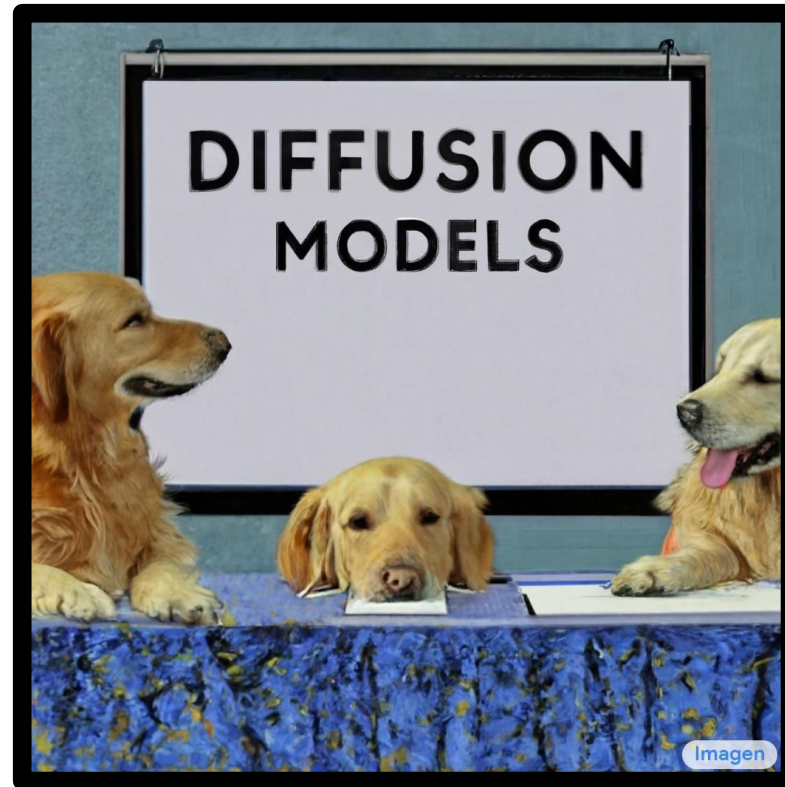
A group of teddy bears in suit in a corporate office celebrating the birthday of their friend. There is a pizza cake on the desk.



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

[Source: CVPR2022-tutorial-diffusion-models](#)

Denoising Diffusion Probabilistic Models (DDPM)

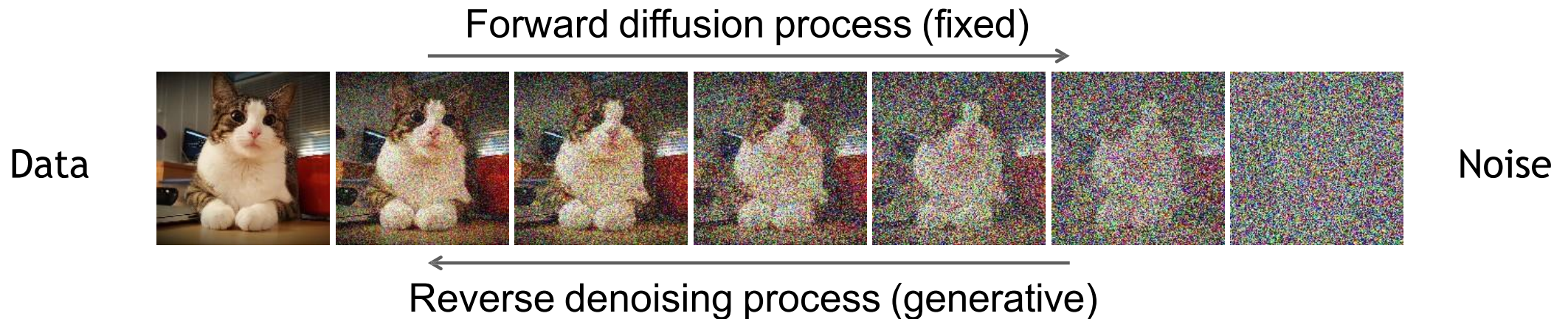


Denoising Diffusion Models

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](#)

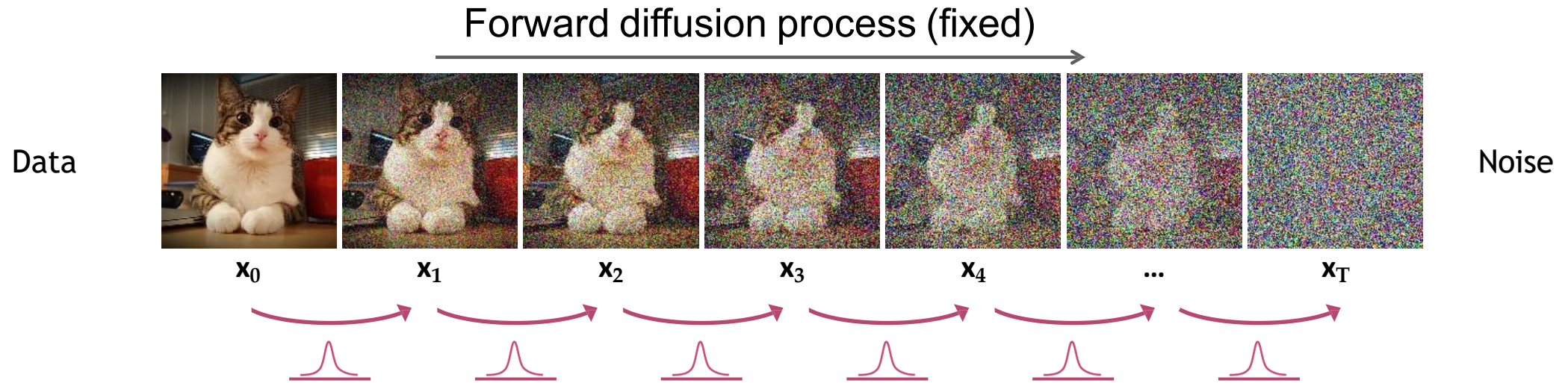
[Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020](#)

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

[Source: CVPR2022-tutorial-diffusion-models](#)

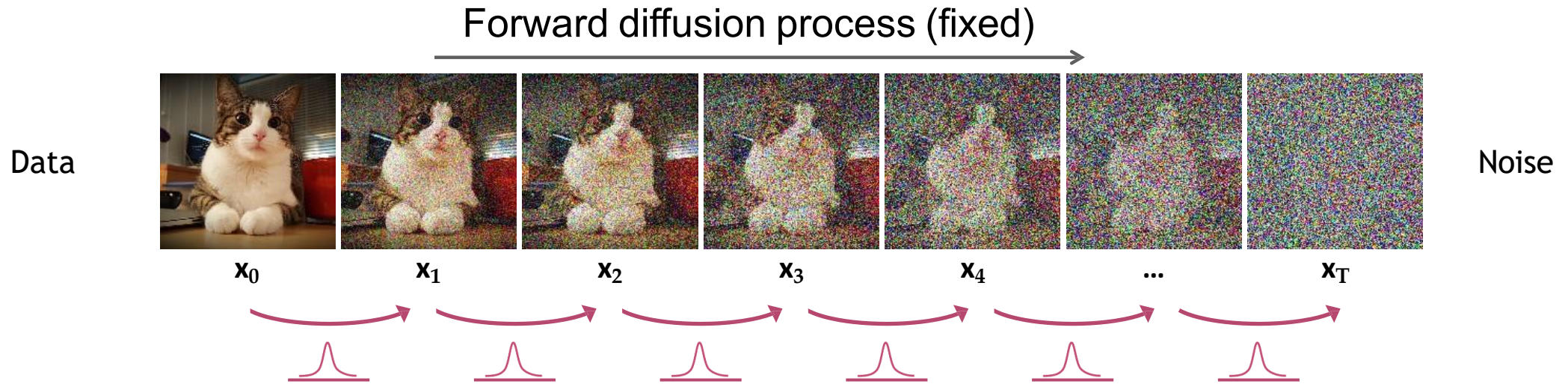
Forward Diffusion Process

The formal definition of the forward process in T steps:



Forward Diffusion Process

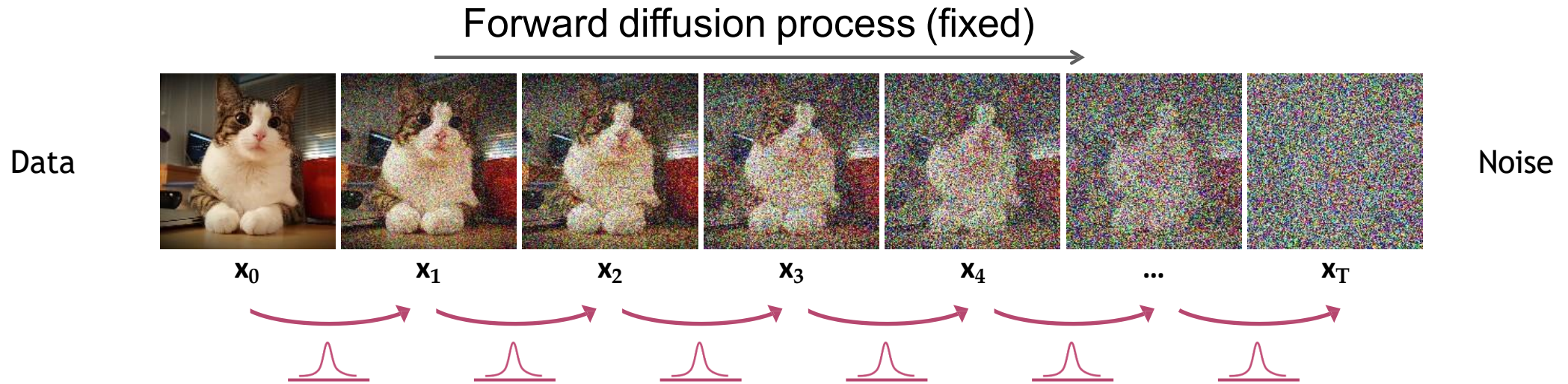
The formal definition of the forward process in T steps:



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

Forward Diffusion Process

The formal definition of the forward process in T steps:

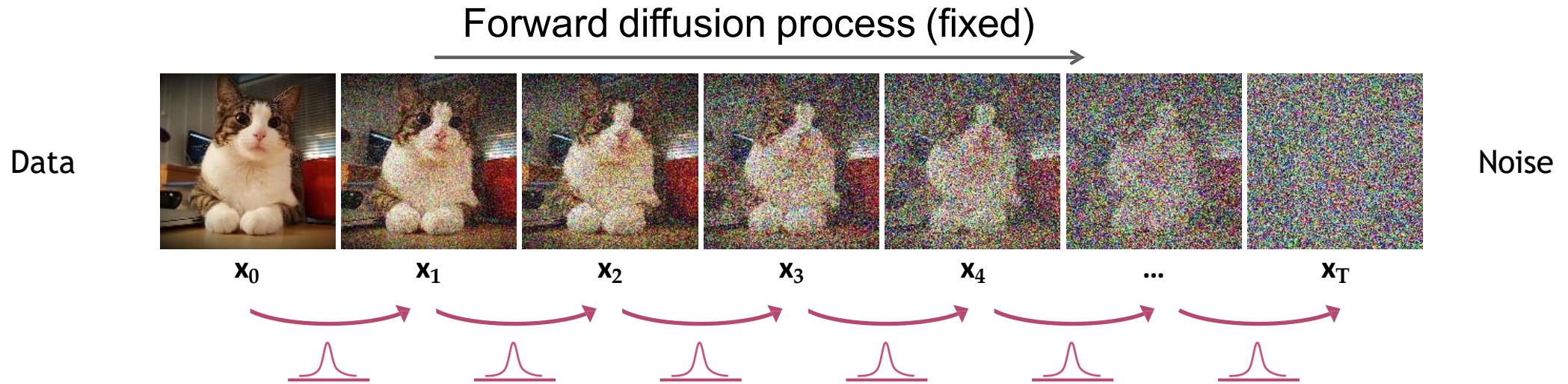


$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

coefficients:
variance preserving

Forward Diffusion Process

The formal definition of the forward process in T steps:

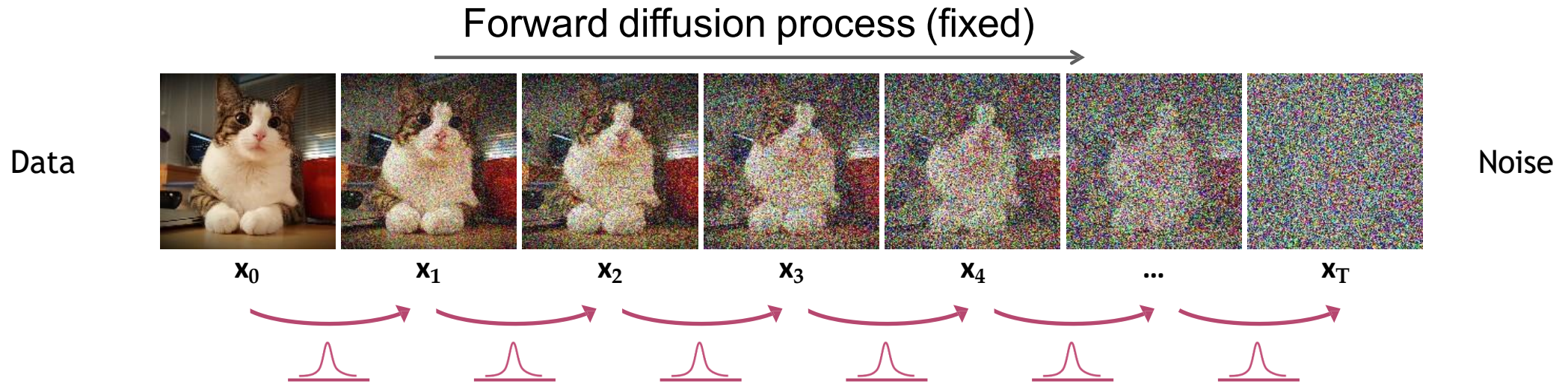


$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

t : “schedule”,
key to Diffusion Models’ success

Forward Diffusion Process

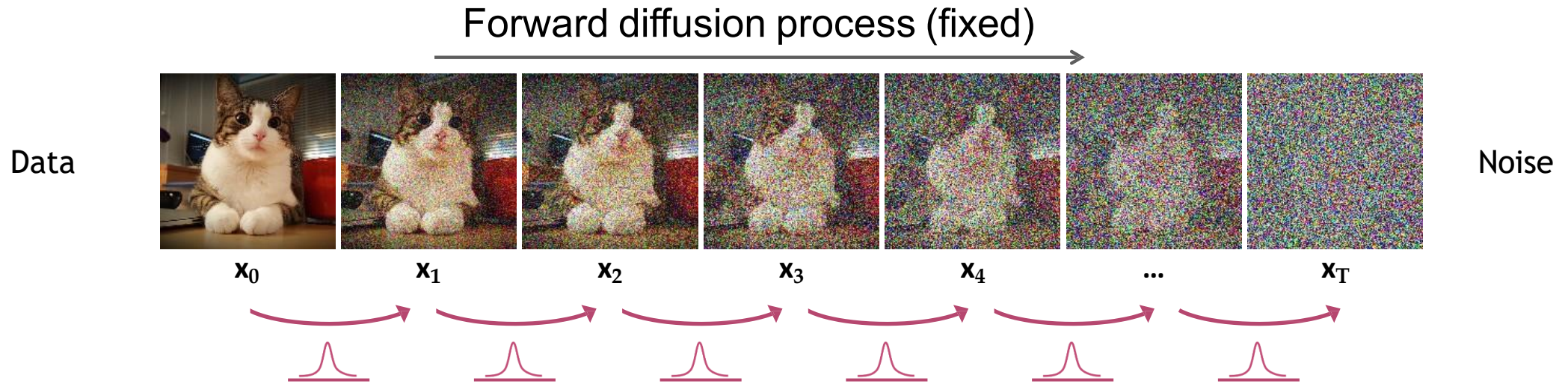
The formal definition of the forward process in T steps:



$$x_t = \underbrace{\sqrt{1 - \beta_t} x_{t-1}}_{\text{mean of } x_t} + \underbrace{\sqrt{\beta_t}}_{\text{std of } x_t} \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

Forward Diffusion Process

The formal definition of the forward process in T steps:

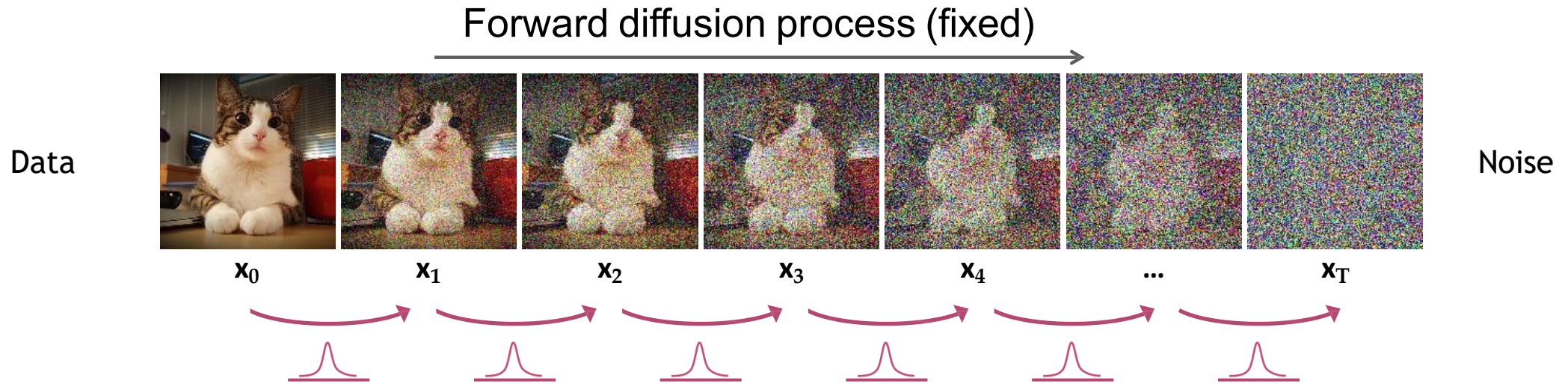


Kernel

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

Forward Diffusion Process

The formal definition of the forward process in T steps:

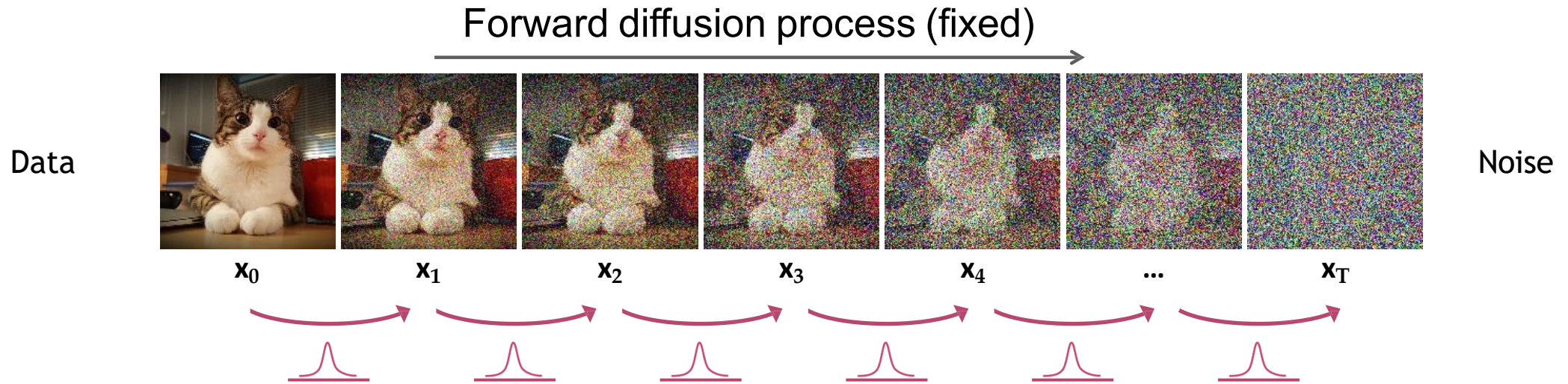


Kernel

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \underbrace{\sqrt{1 - \beta_t} \mathbf{x}_{t-1}}_{\text{mean of } x_t}, \underbrace{\beta_t \mathbf{I}}_{\text{var of } x_t})$$

Forward Diffusion Process

The formal definition of the forward process in T steps:



Kernel

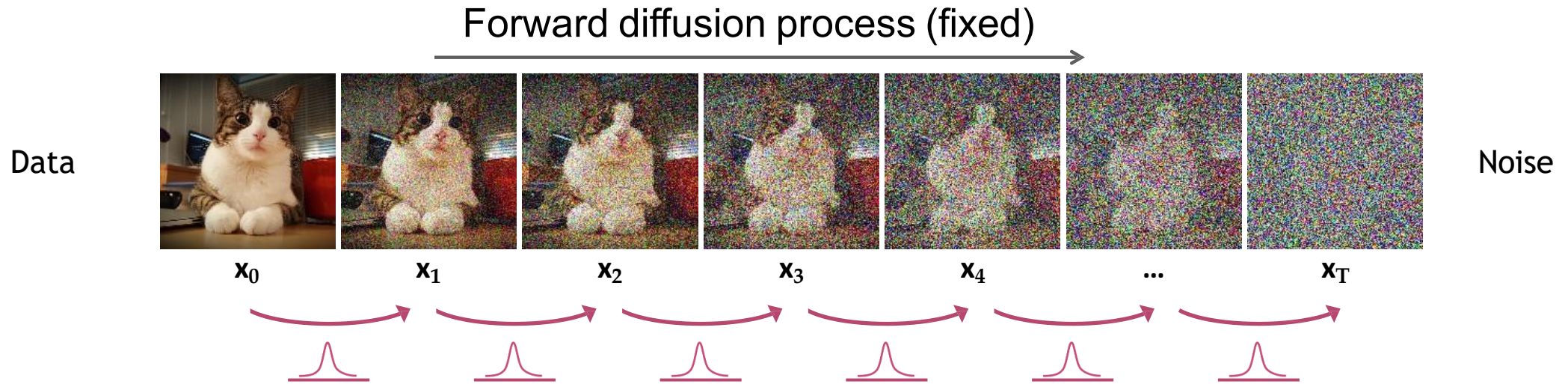
$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

identity matrix

- sampling is i.i.d.
- dim = dim of data

Forward Diffusion Process

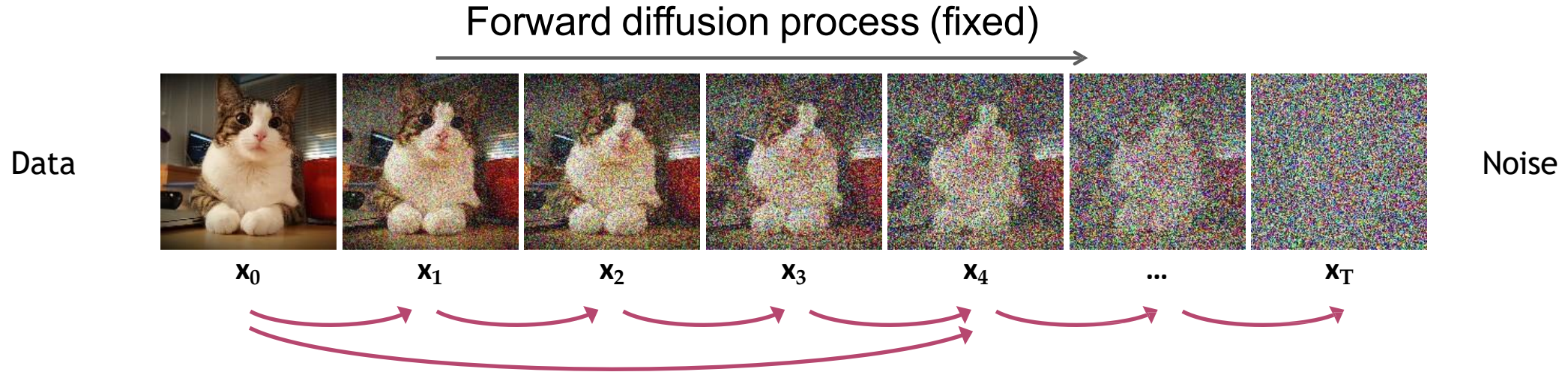
The formal definition of the forward process in T steps:



Kernel

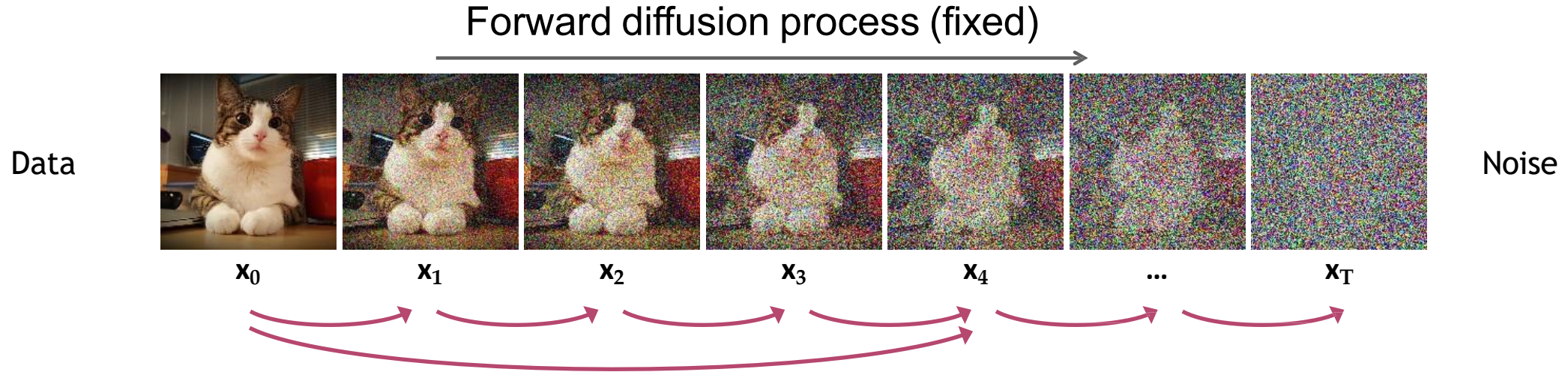
$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad \rightarrow \quad q(\mathbf{x}_{1:T} | \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}) \quad (\text{joint})$$

Diffusion Kernel



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)

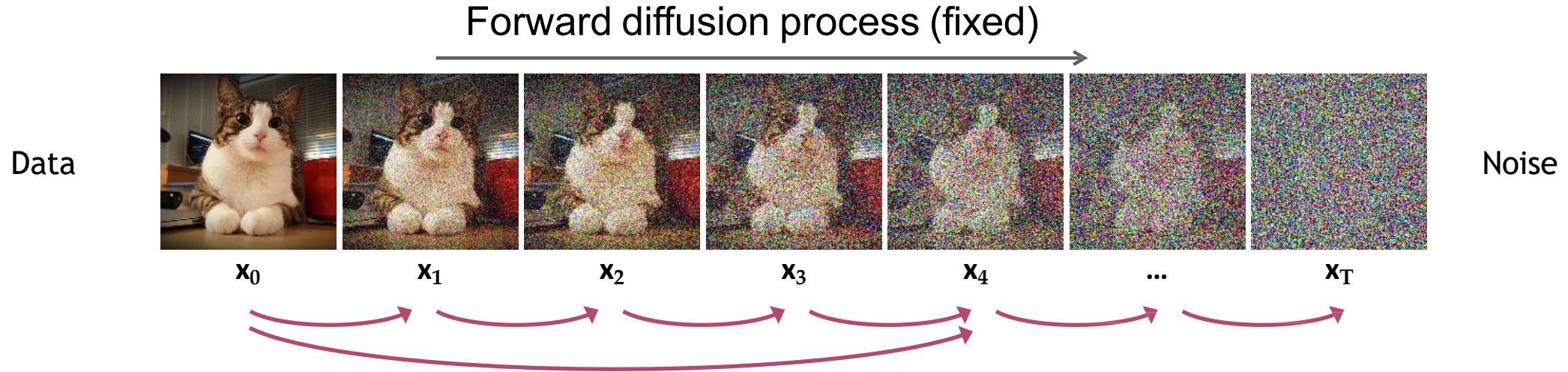
Diffusion Kernel



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

Diffusion Kernel



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

β_t values schedule (i.e., the noise schedule) is designed such that

$$\bar{\alpha}_T \rightarrow 0 \quad \text{and} \quad q(\mathbf{x}_T | \mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

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 diffused data distribution $q(\mathbf{x}_t)$?

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 diffused data distribution $q(\mathbf{x}_t)$?

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

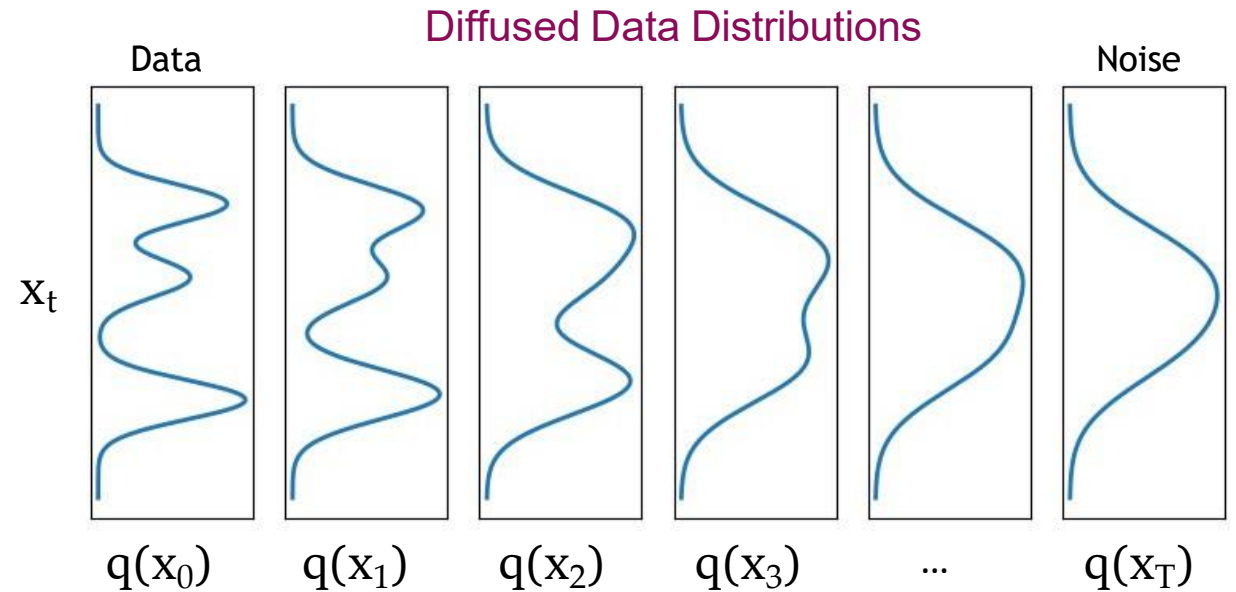
The diffusion kernel is Gaussian convolution.

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 diffused data distribution $q(\mathbf{x}_t)$?

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

The diffusion kernel is Gaussian convolution.

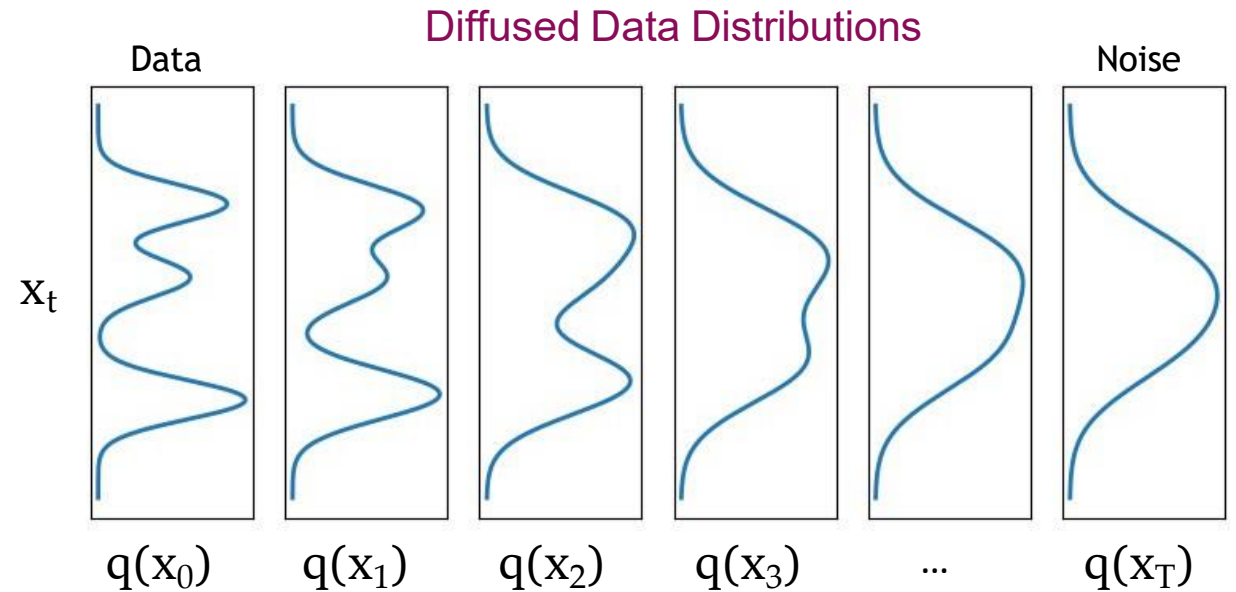


What happens to a distribution in the forward diffusion?

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

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

The diffusion kernel is Gaussian convolution.



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).

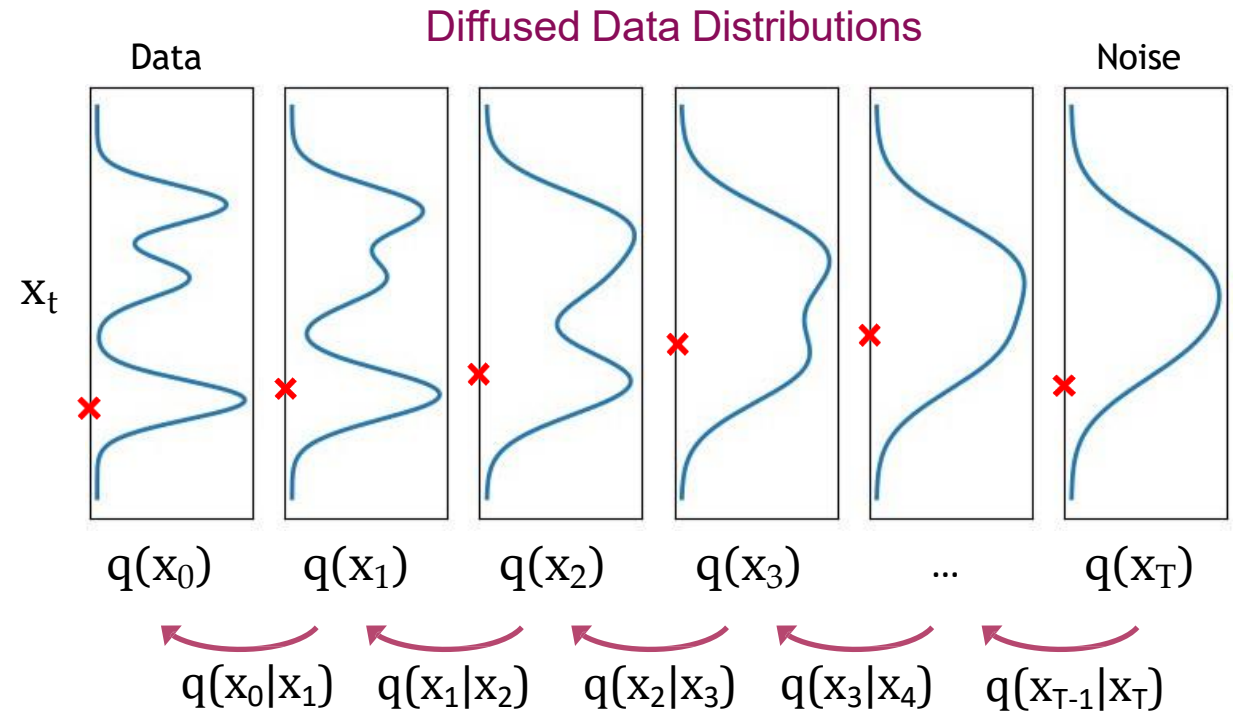
Generative Learning by Denoising

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



Generative Learning by Denoising

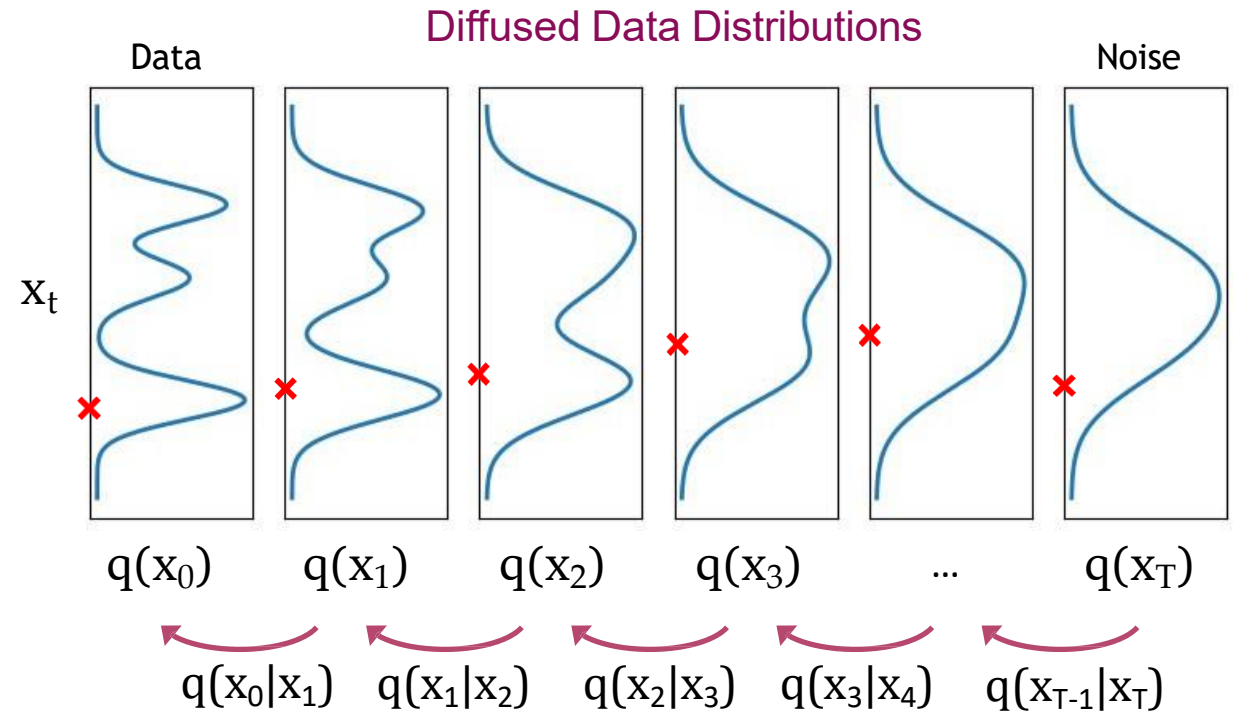
Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Generation:

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

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

In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$
is intractable.



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:

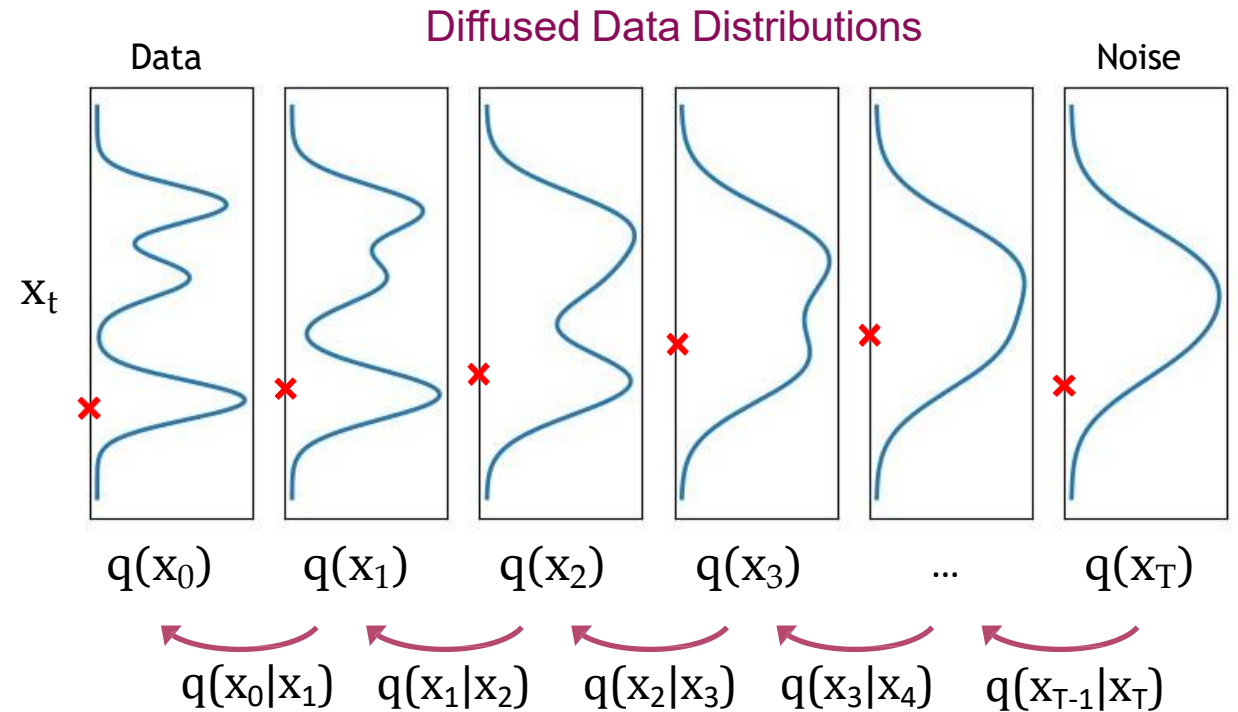
Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

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

In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$
is intractable.

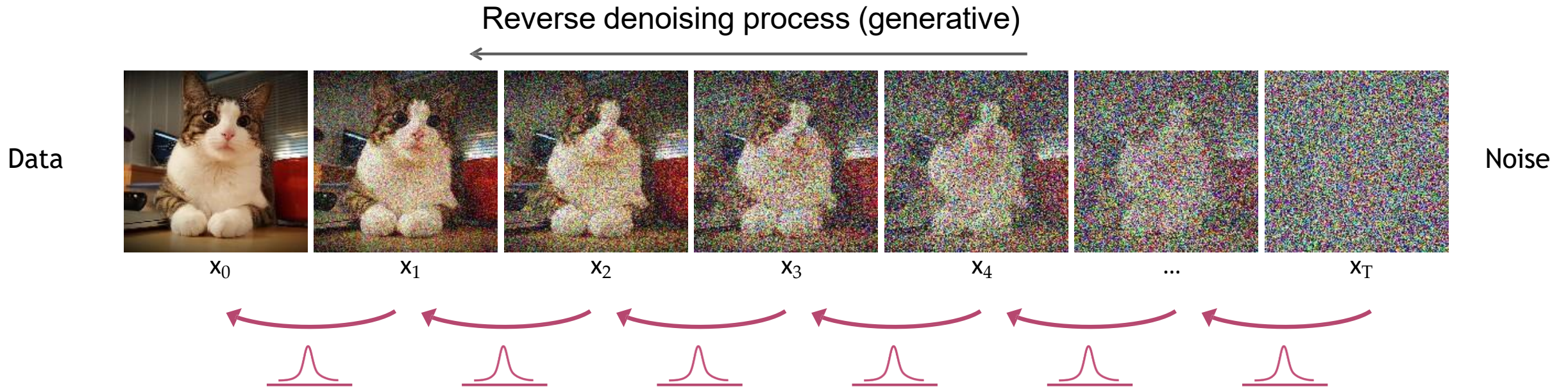
Can we approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$?

Yes, we can use a **Normal distribution** if β_t is small in each forward diffusion step.



Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



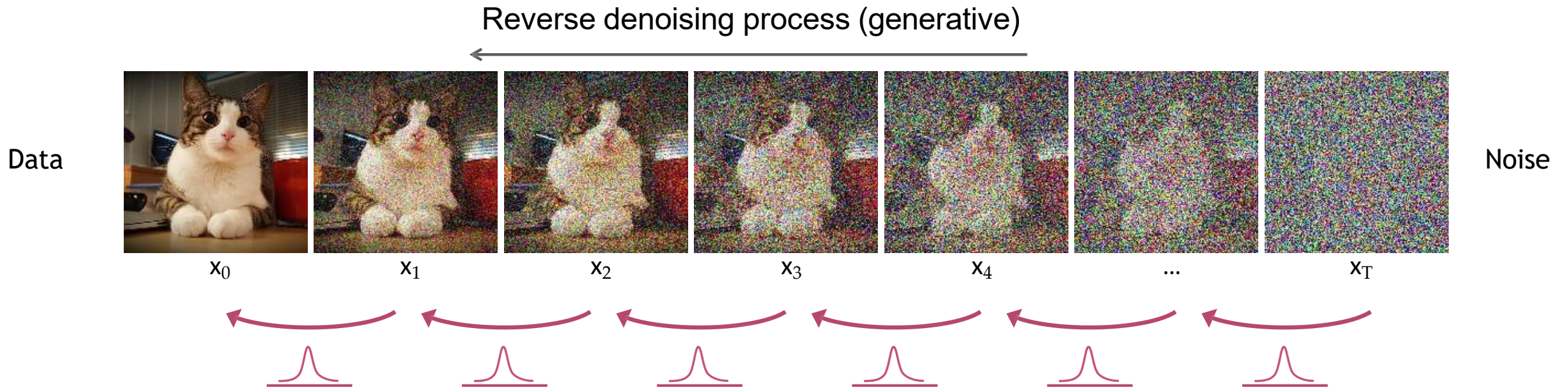
$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_{\theta}(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2 \mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_{\theta}(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2 \mathbf{I}) \quad \rightarrow \quad p_{\theta}(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Trainable network
(U-net, Denoising Autoencoder)

Learning Denoising Model

Recall that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$

[Ho et al. NeurIPS 2020](#) proposed the denoising model using a *noise-prediction* network and represented the mean as:

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

They use this *noise-prediction* network to compute the loss between the added noise and predicted noise:

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

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

Training and Sample Generation

Algorithm 1 Training

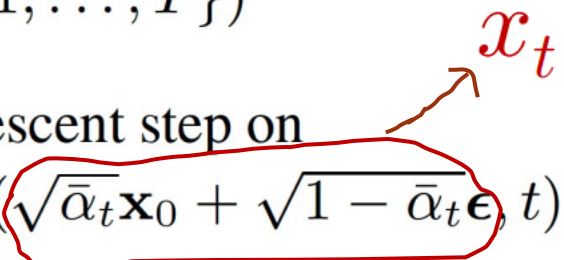
- 1: **repeat**
 - 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
 - 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
 - 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 5: Take gradient descent step on
$$\nabla_{\theta} \left\| \epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 -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

x_t

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
```



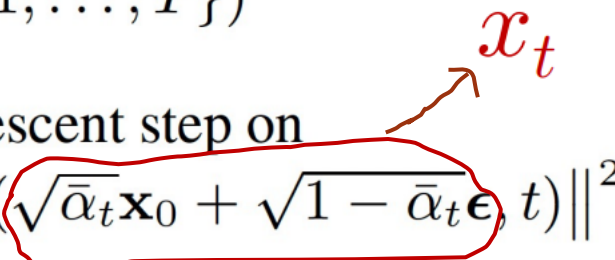
Algorithm 2 Sampling

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

Training and Sample Generation

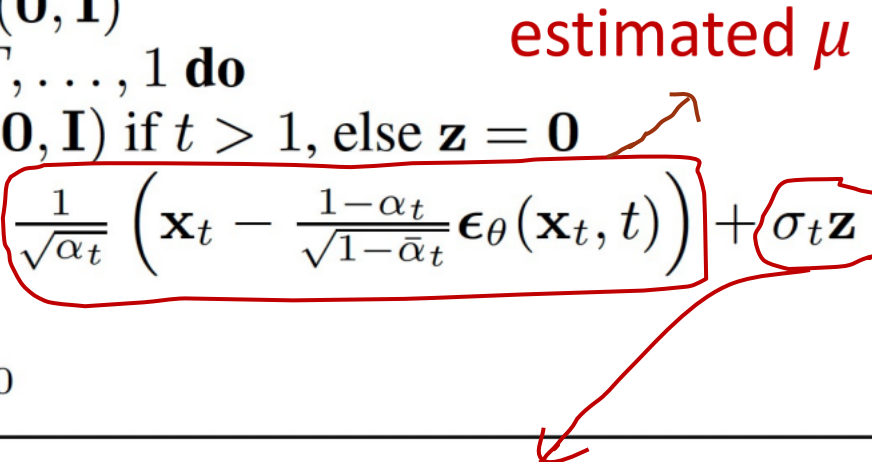
Algorithm 1 Training

```
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
```



Algorithm 2 Sampling

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

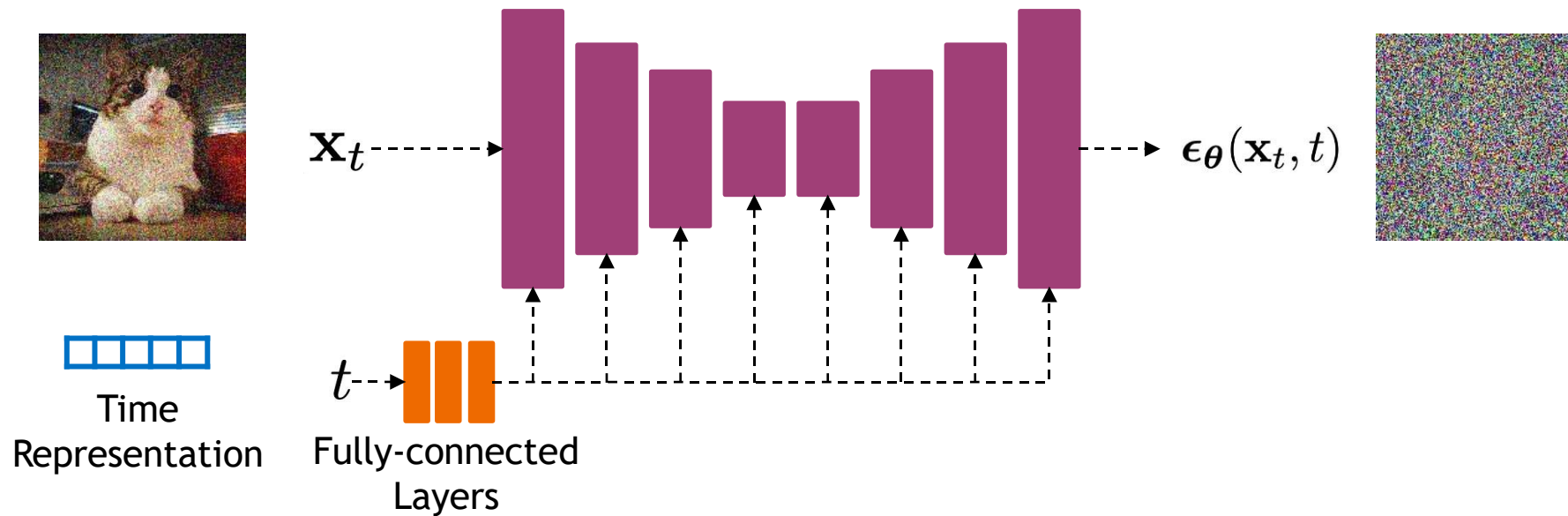


sampling from
estimated distribution

Implementation Considerations: Network Architectures

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent

$$\epsilon_{\theta}(\mathbf{x}_t, t)$$

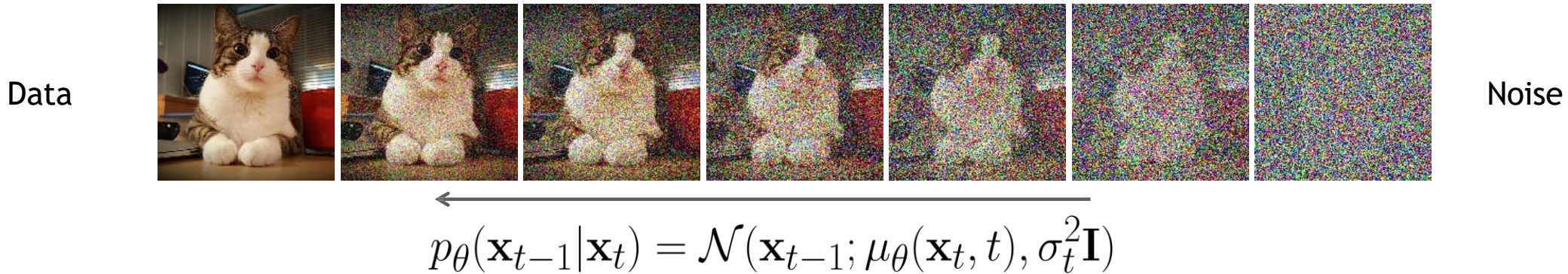


Time representation: sinusoidal positional embeddings or random Fourier features.

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](#))

Diffusion Parameters: Noise Schedule

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



Above, β_t and σ_t^2 control the variance of the forward diffusion and reverse denoising processes respectively.

Often a linear schedule is used for β_t , and σ_t^2 is set equal to β_t .

[Kingma et al. NeurIPS 2022](#) introduce a new parameterization of diffusion models using signal-to-noise ratio (SNR), and show how to learn the noise schedule by minimizing the variance of the training objective.

We can also train σ_t^2 while training the diffusion model by minimizing the variational bound ([Improved DPM by Nichol and Dhariwal ICML 2021](#)) or after training the diffusion model ([Analytic-DPM by Bao et al. ICLR 2022](#)).

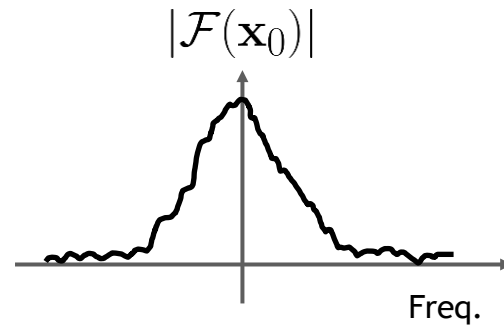
What happens to an image in the forward diffusion process?

Recall that sampling from $q(\mathbf{x}_t|\mathbf{x}_0)$ is done using $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)} \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

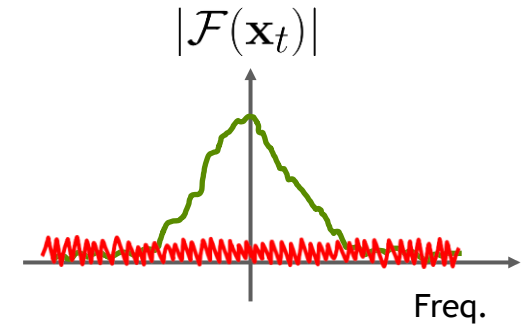
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)} \epsilon$$

Fourier Transform

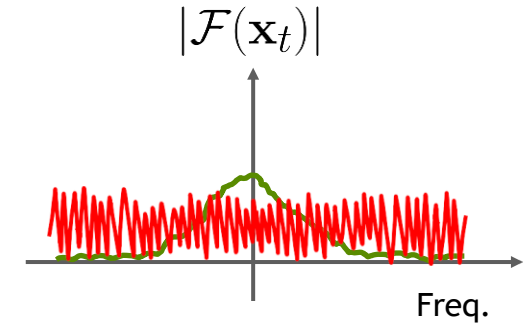
$$\mathcal{F}(\mathbf{x}_t) = \sqrt{\bar{\alpha}_t} \mathcal{F}(\mathbf{x}_0) + \sqrt{(1 - \bar{\alpha}_t)} \mathcal{F}(\epsilon)$$



Small t
 $\bar{\alpha}_t \sim 1$

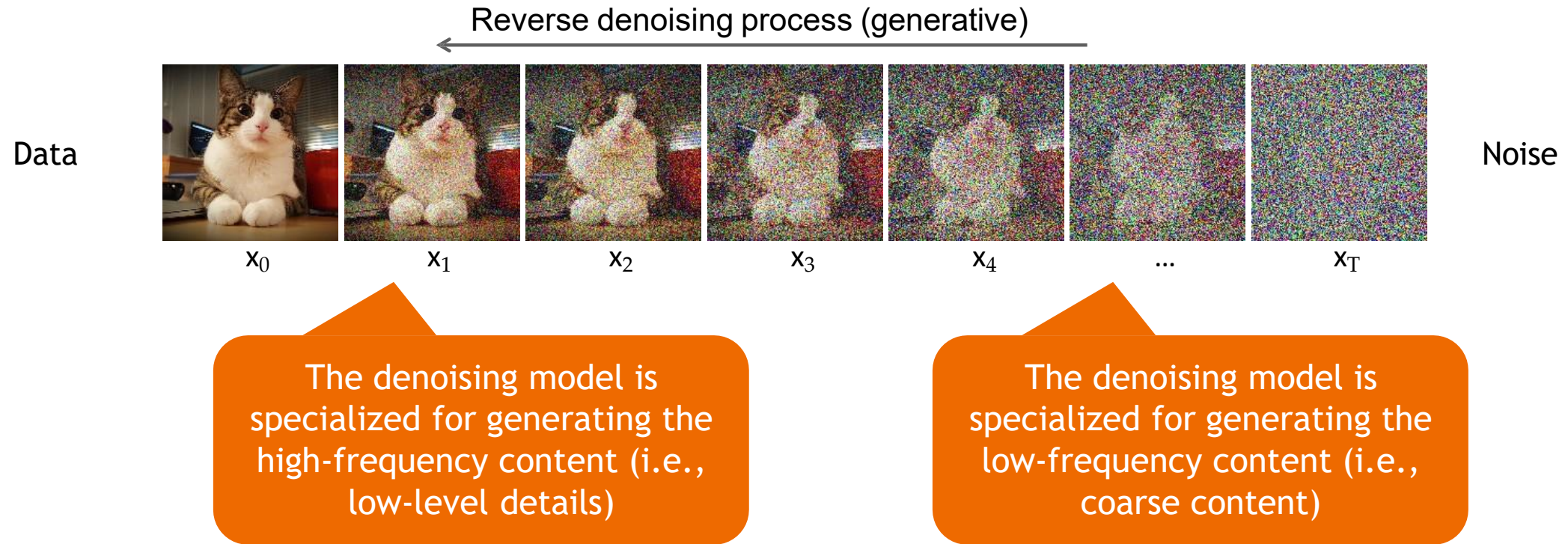


Large t
 $\bar{\alpha}_t \sim 0$



In the forward diffusion, the high frequency content is perturbed faster.

Content-Detail Tradeoff



The weighting of the training objective for different timesteps is important!

Denoising Diffusion Probabilistic Models: Summary

The model is trained by sampling from the forward diffusion process and training a denoising model to predict the noise. We discussed how the forward process perturbs the data distribution or data samples.

The devil is in the details:

- Network architectures
- Objective weighting
- Diffusion parameters (i.e., noise schedule)

See [“Elucidating the Design Space of Diffusion-Based Generative Models” by Karras et al.](#) for important design decisions.

Score-based Generative Modeling with Differential Equations

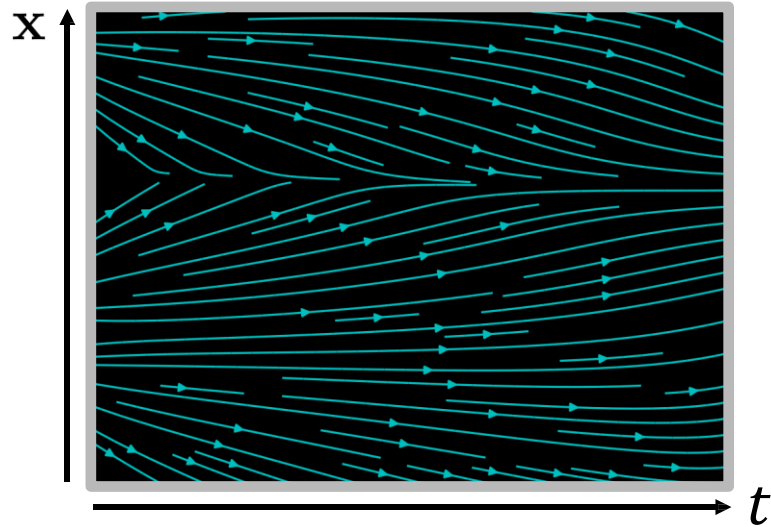
Crash Course in Differential Equations

Score-based Generative Modeling with Differential Equations

Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

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

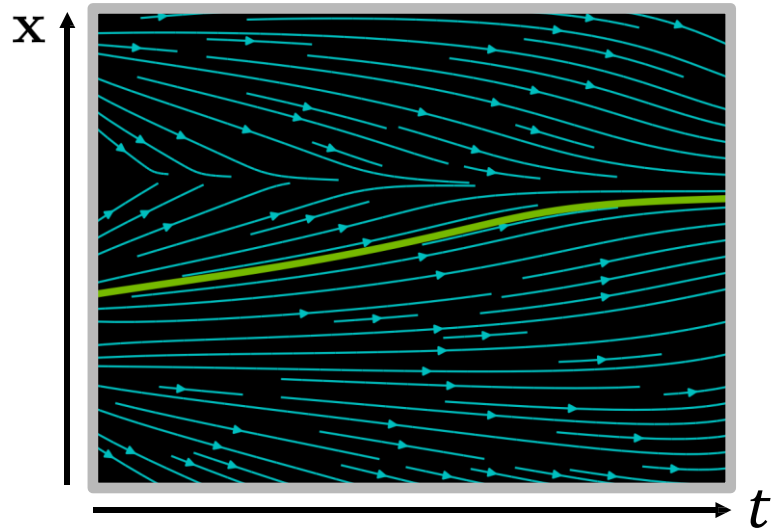


Score-based Generative Modeling with Differential Equations

Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

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



Analytical Solution:

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

Iterative Numerical Solution:

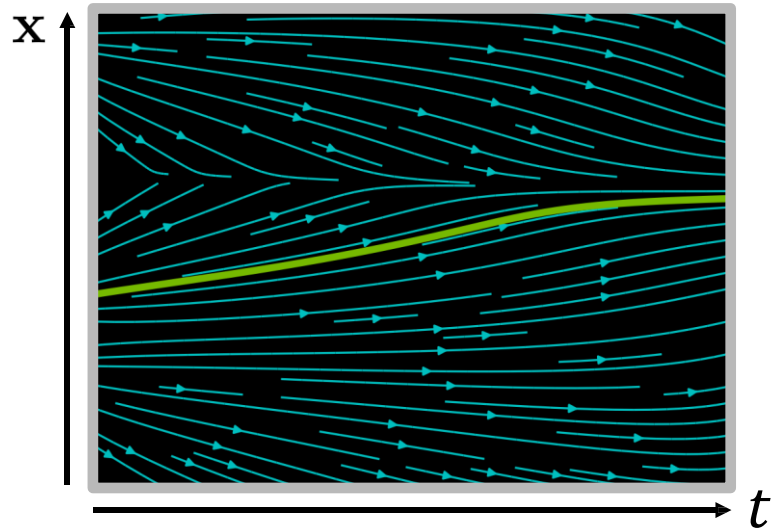
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

Score-based Generative Modeling with Differential Equations

Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

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



Analytical Solution:

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

Iterative Numerical Solution:

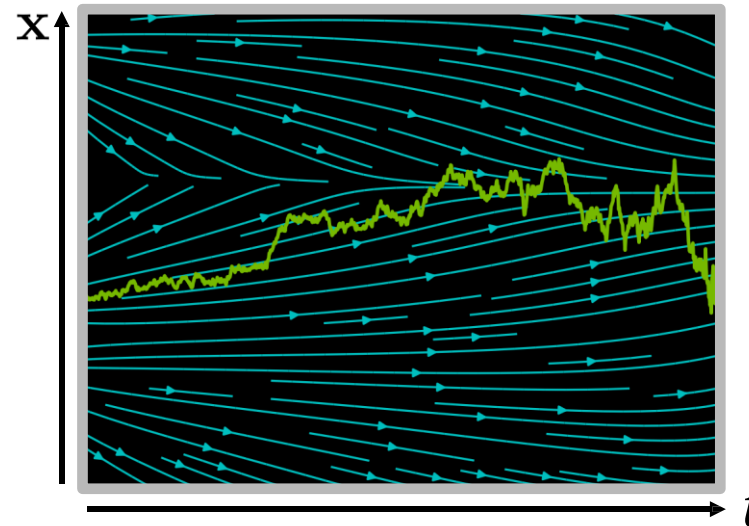
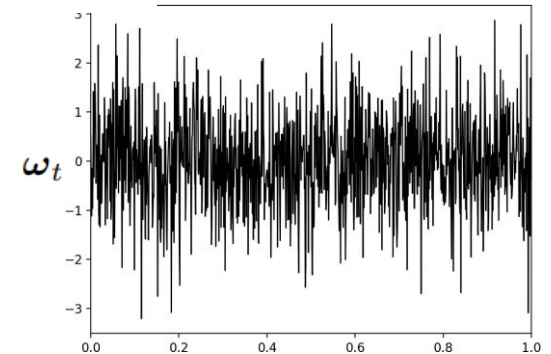
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

Stochastic Differential Equation (SDE):

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

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

Wiener Process
(Gaussian
White Noise)



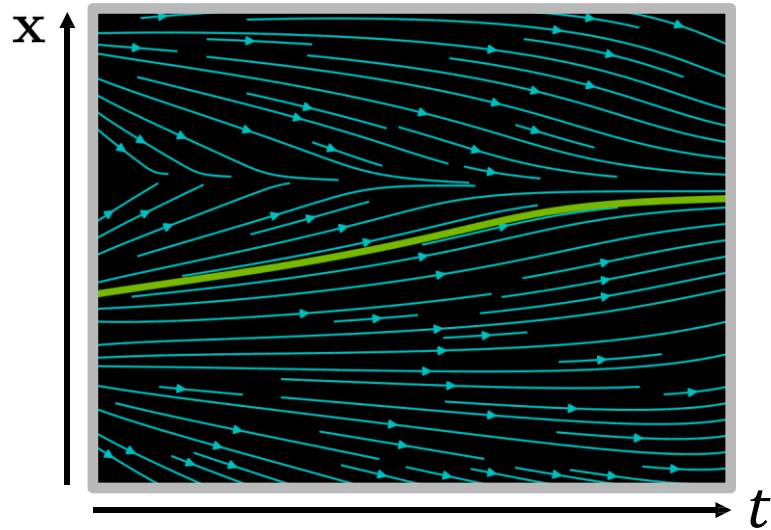
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Score-based Generative Modeling with Differential Equations

Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

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



Analytical Solution:

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

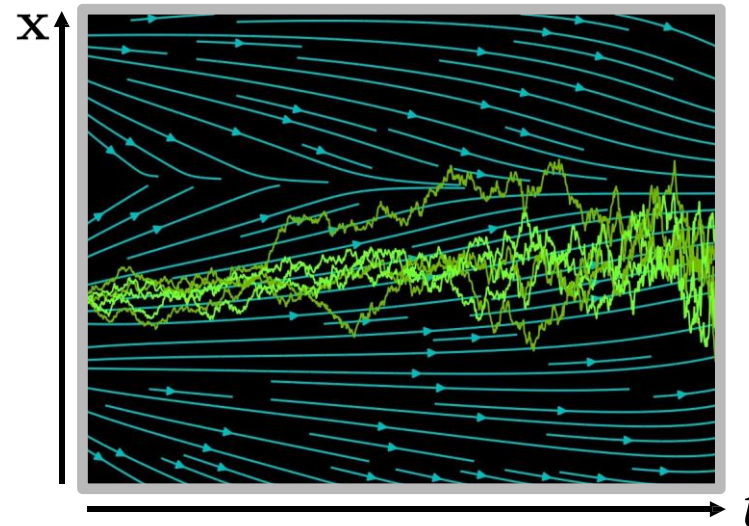
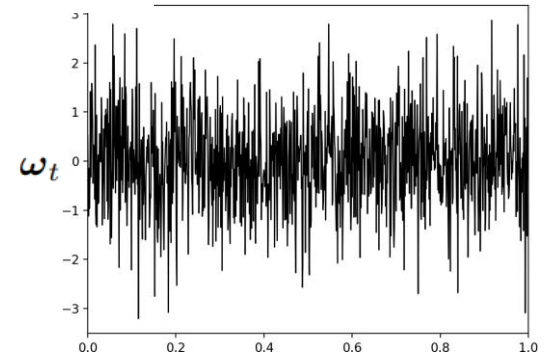
Iterative Numerical Solution:

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

Stochastic Differential Equation (SDE):

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

Wiener Process
(Gaussian
White Noise)

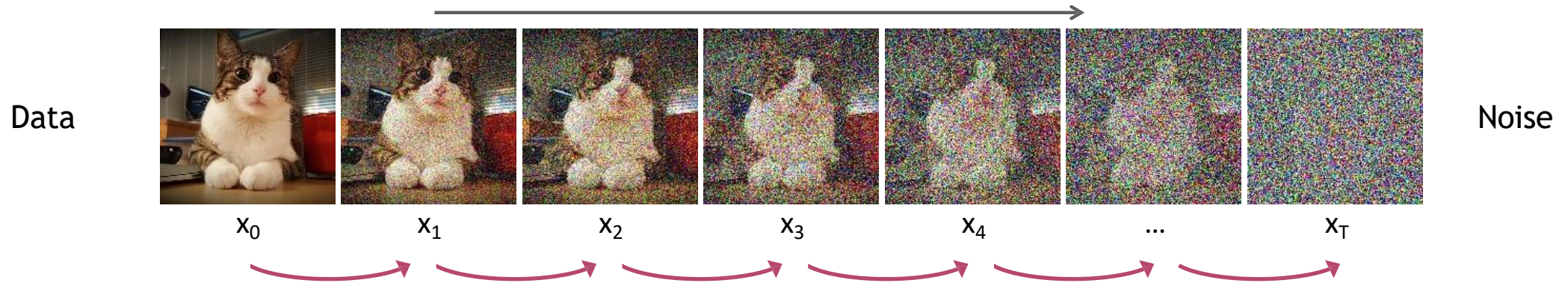


$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Forward Diffusion Process as Stochastic Differential Equation

Consider the limit of many small steps:

Forward diffusion process (fixed)



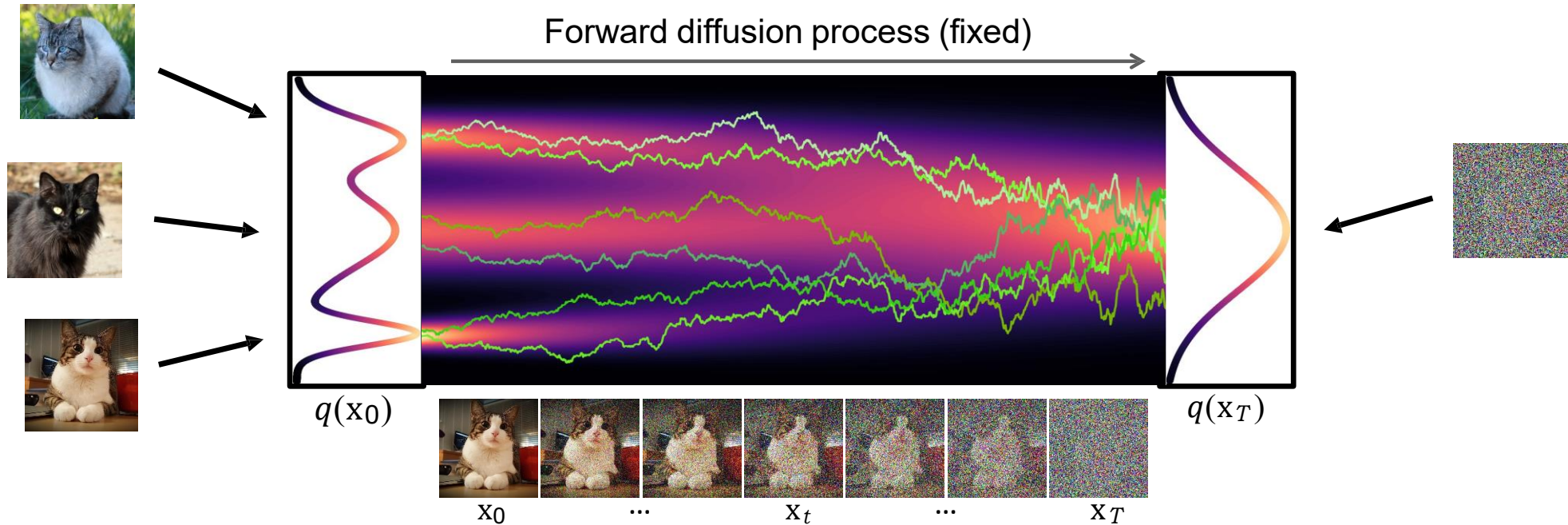
$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

↓ discretization of

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

Stochastic Differential Equation (SDE)

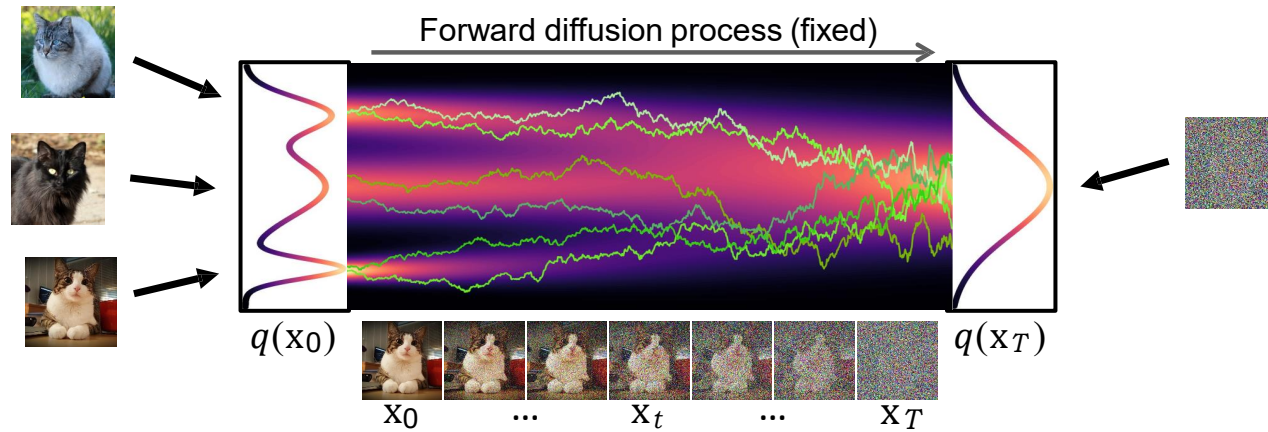
Forward Diffusion Process as Stochastic Differential Equation



Forward Diffusion SDE:

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

The Generative Reverse Stochastic Differential Equation

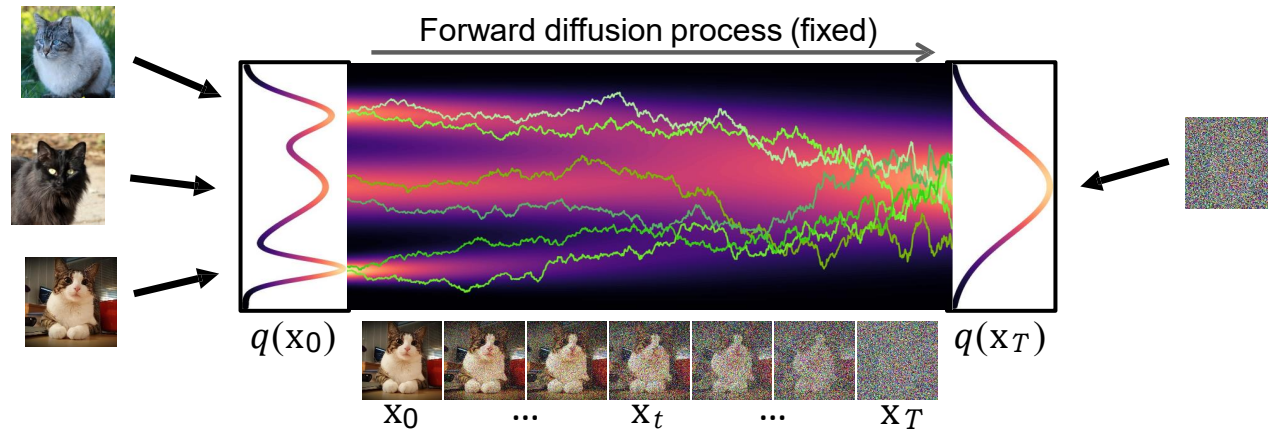


Forward Diffusion SDE:

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

But what about the reverse direction, necessary for generation?

The Generative Reverse Stochastic Differential Equation



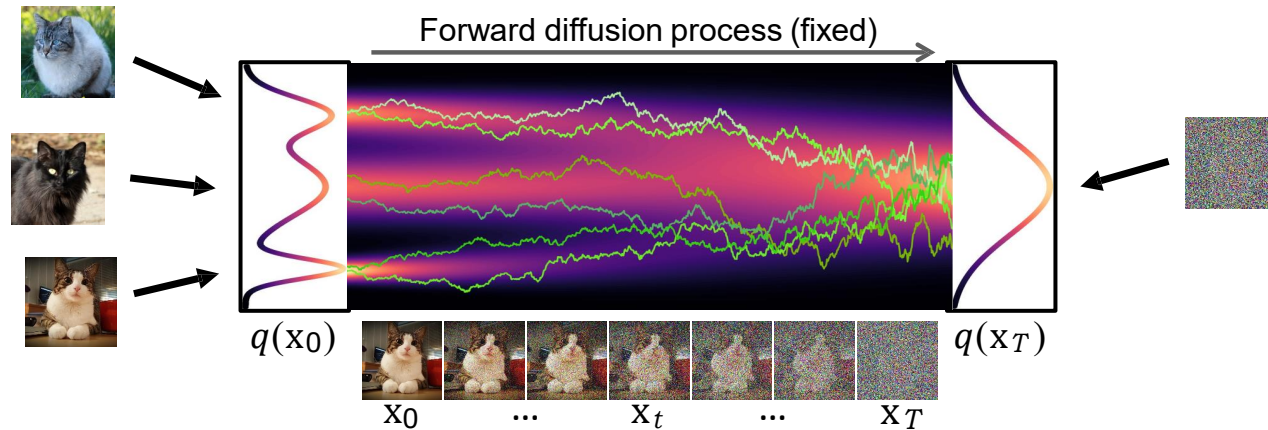
Forward Diffusion SDE:

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

**Reverse Generative
Diffusion SDE:**

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

The Generative Reverse Stochastic Differential Equation



Forward Diffusion SDE:

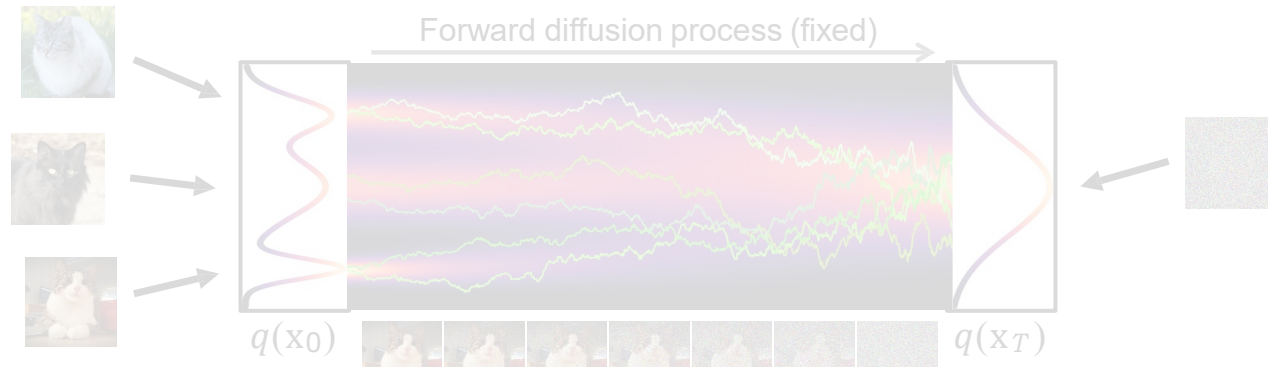
$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)} d\omega_t$$

**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!

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:

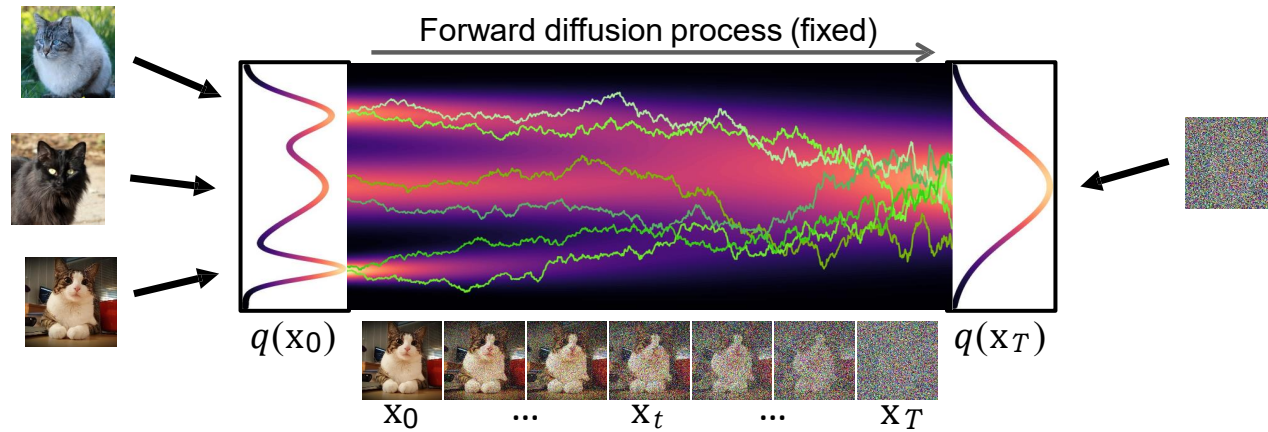
$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)} d\omega_t$$

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!

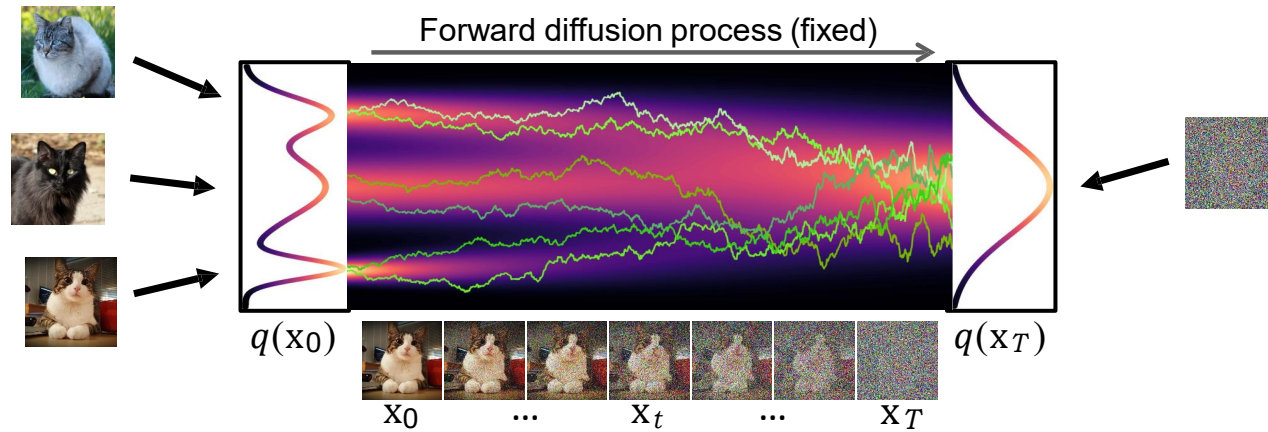
Score Matching



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

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

Score Matching

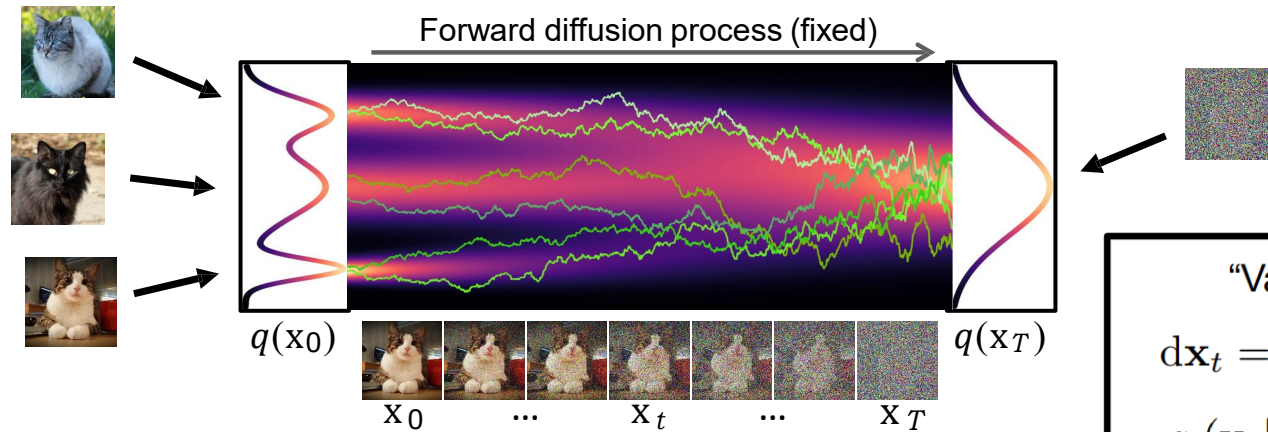


- Naïve idea, learn model for the score function by direct regression?

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

➡ 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!

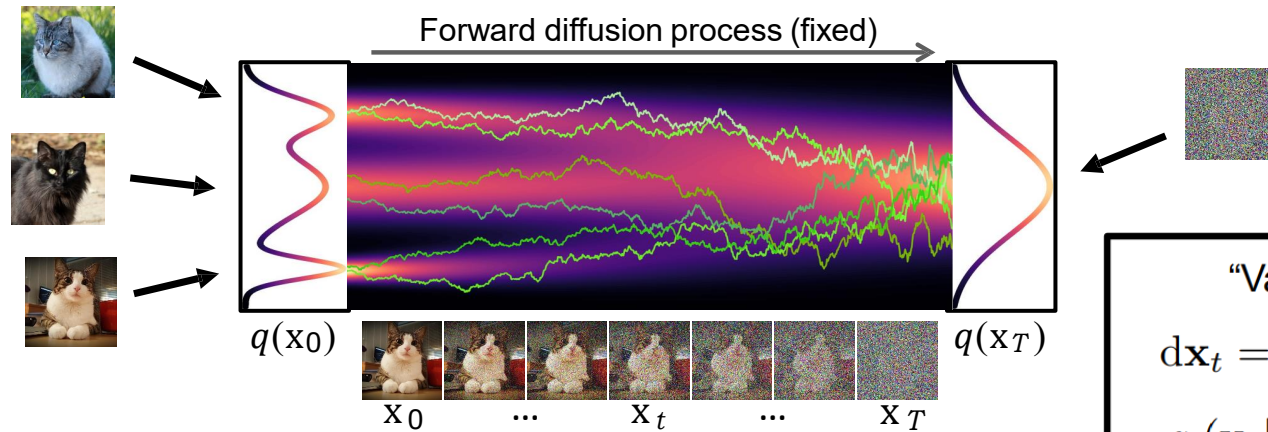
Denoising Score Matching



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

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

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:

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

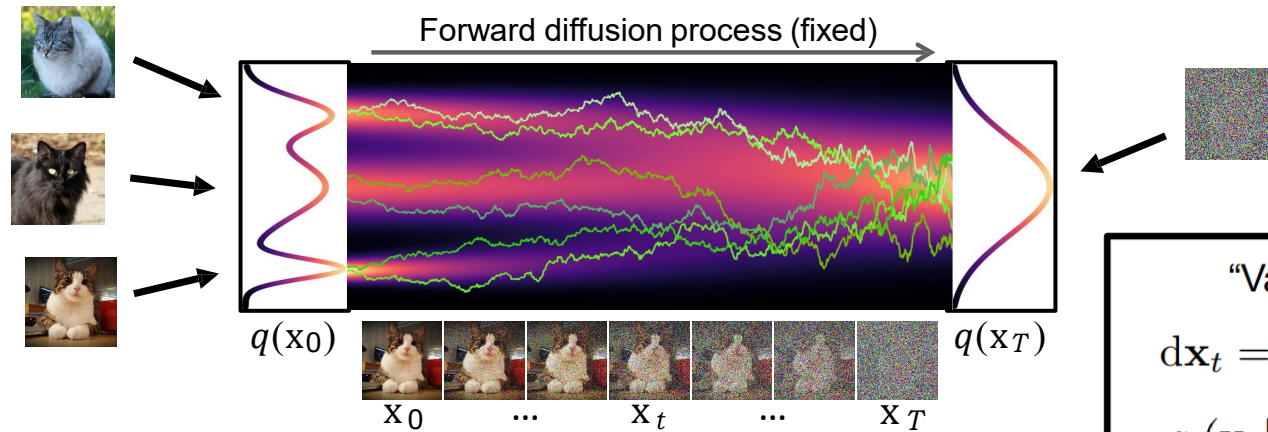
Vincent, in *Neural Computation*, 2011
 Song and Ermon, *NeurIPS*, 2019
 Song et al. *ICLR*, 2021



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

Source: CVPR2023-tutorial-diffusion-models

Denoising Score Matching



“Variance Preserving” SDE:

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

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

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

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

neural network

Vincent, in *Neural Computation*, 2011
 Song and Ermon, *NeurIPS*, 2019
 Song et al. *ICLR*, 2021

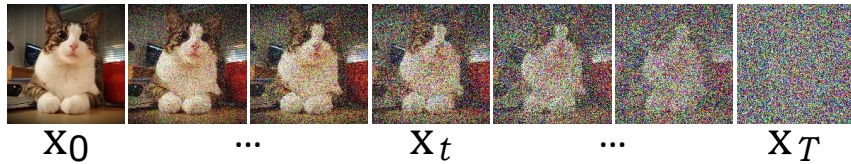
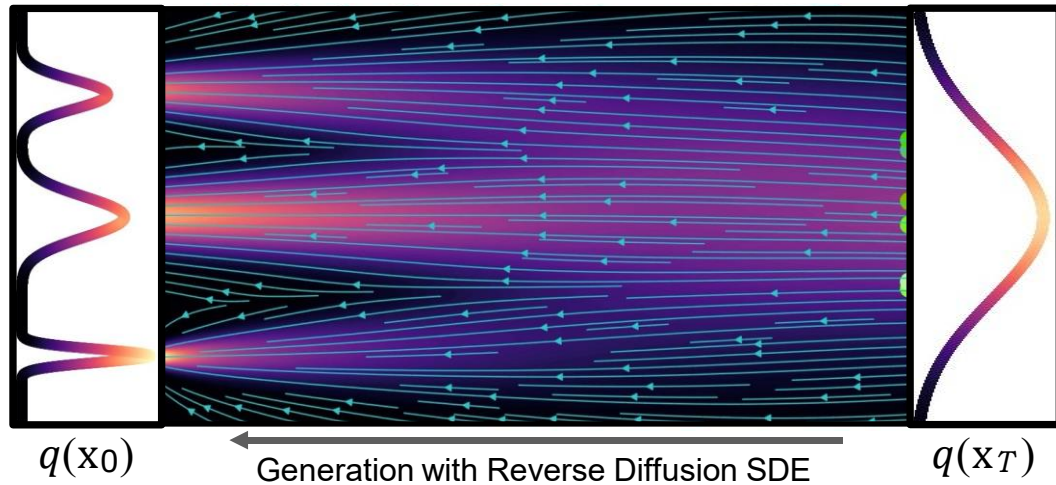


$$\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}(0, T)} \mathbb{E}_{\mathbf{x}_0 \sim q_0(\mathbf{x}_0)} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \frac{1}{\sigma_t^2} \|\epsilon - \epsilon_{\theta}(\mathbf{x}_t, t)\|_2^2$$

Same objectives in DDPM!

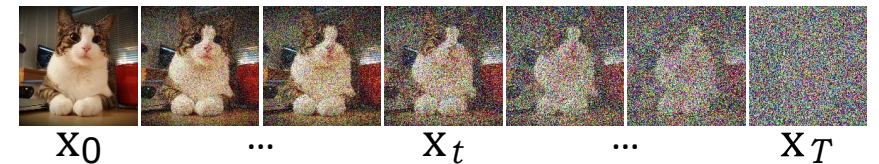
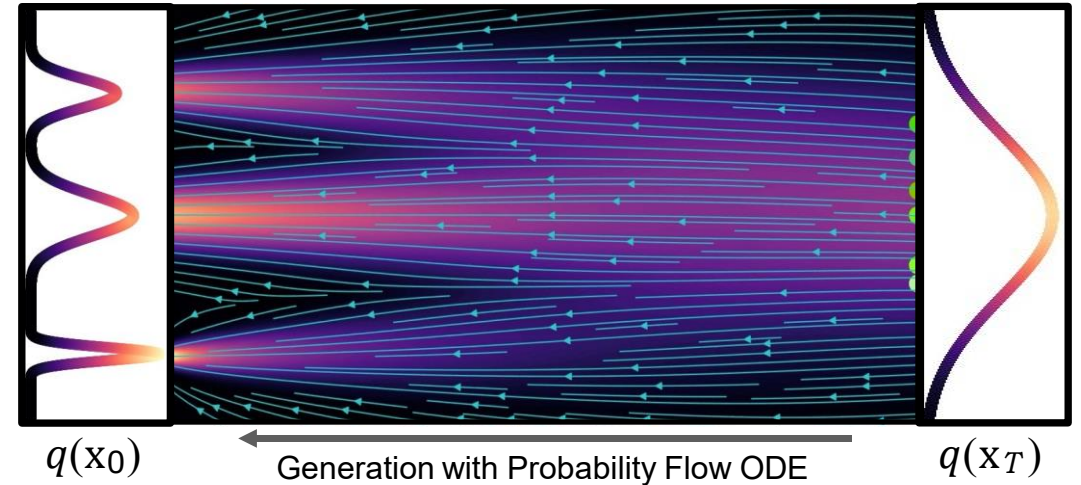
Source: CVPR2023-tutorial-diffusion-models

Synthesis with SDE vs. ODE



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

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

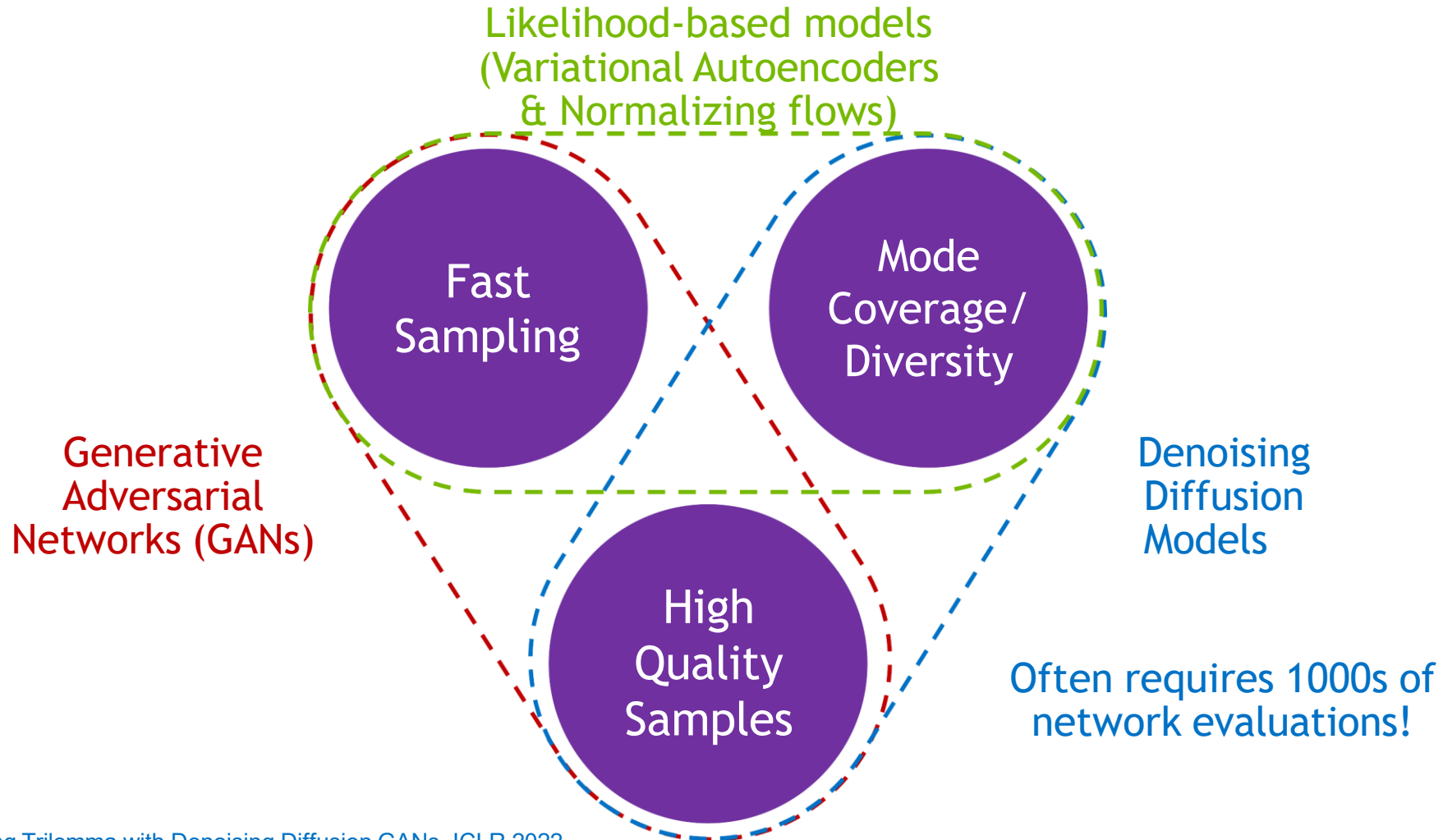


- **Generative Probability Flow ODE (deterministic):**

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

What makes a good generative model?

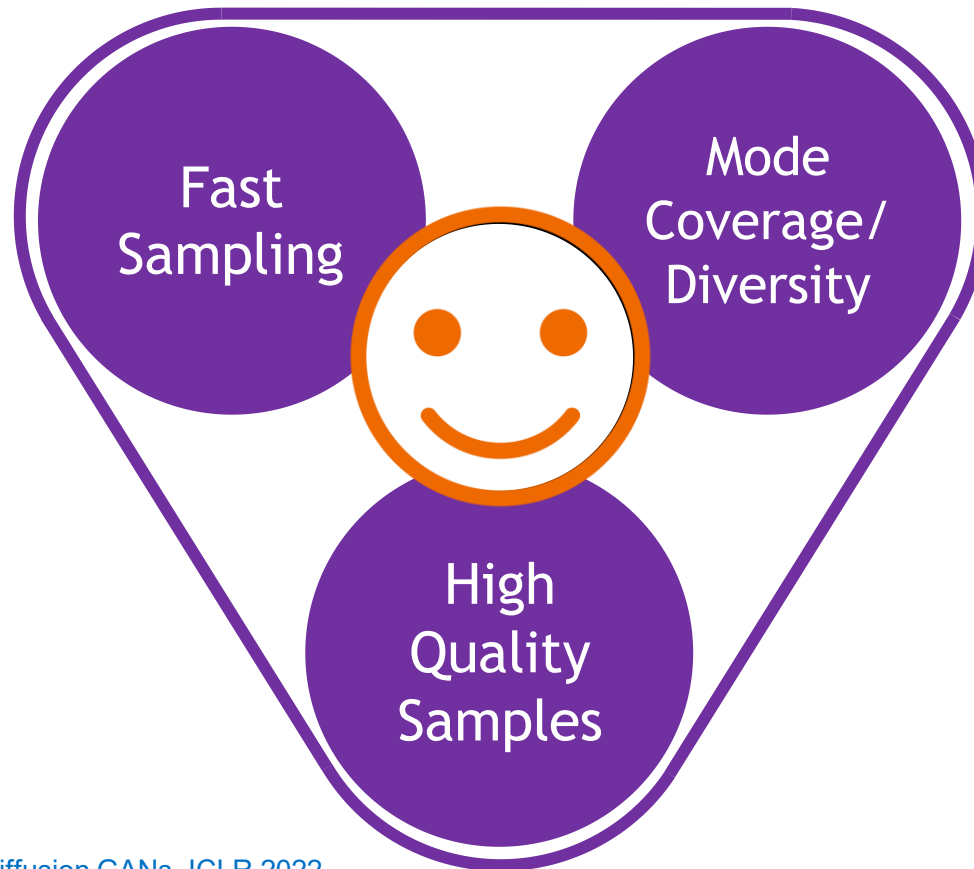
The generative learning trilemma



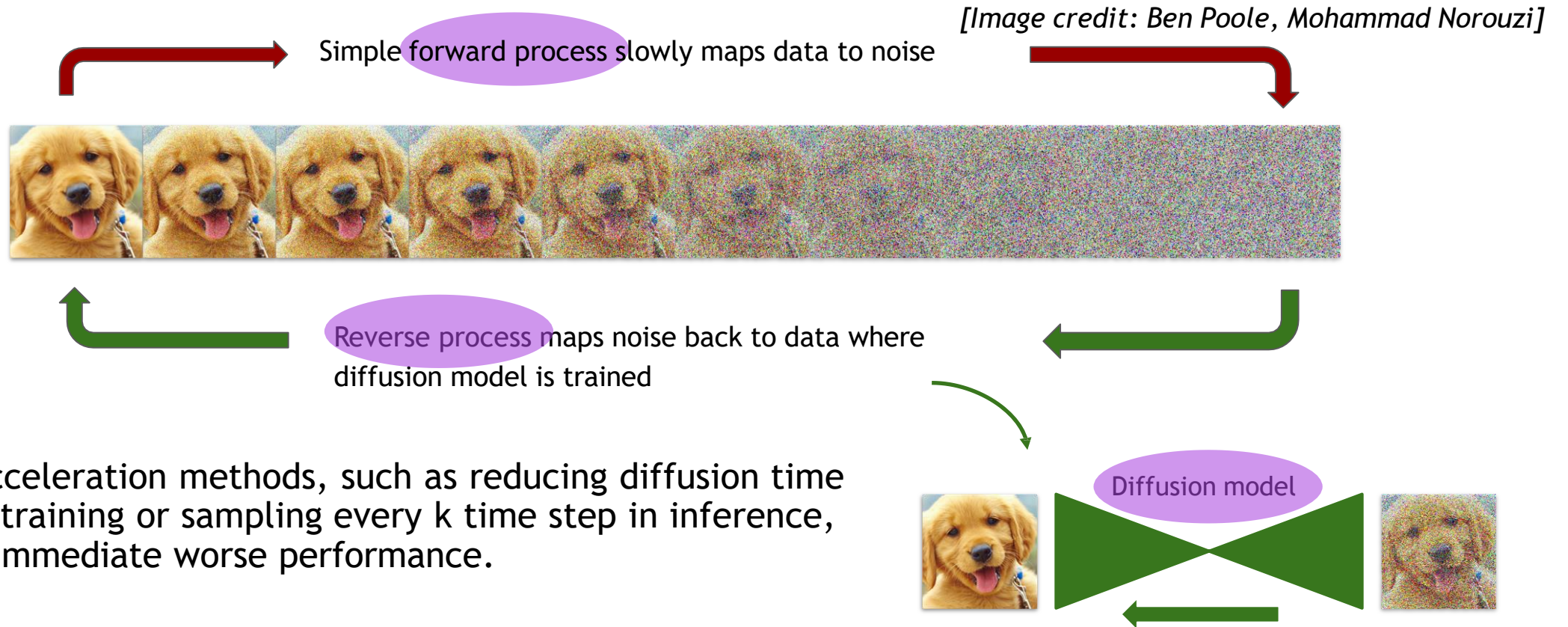
What makes a good generative model?

The generative learning trilemma

Tackle the trilemma by accelerating diffusion models



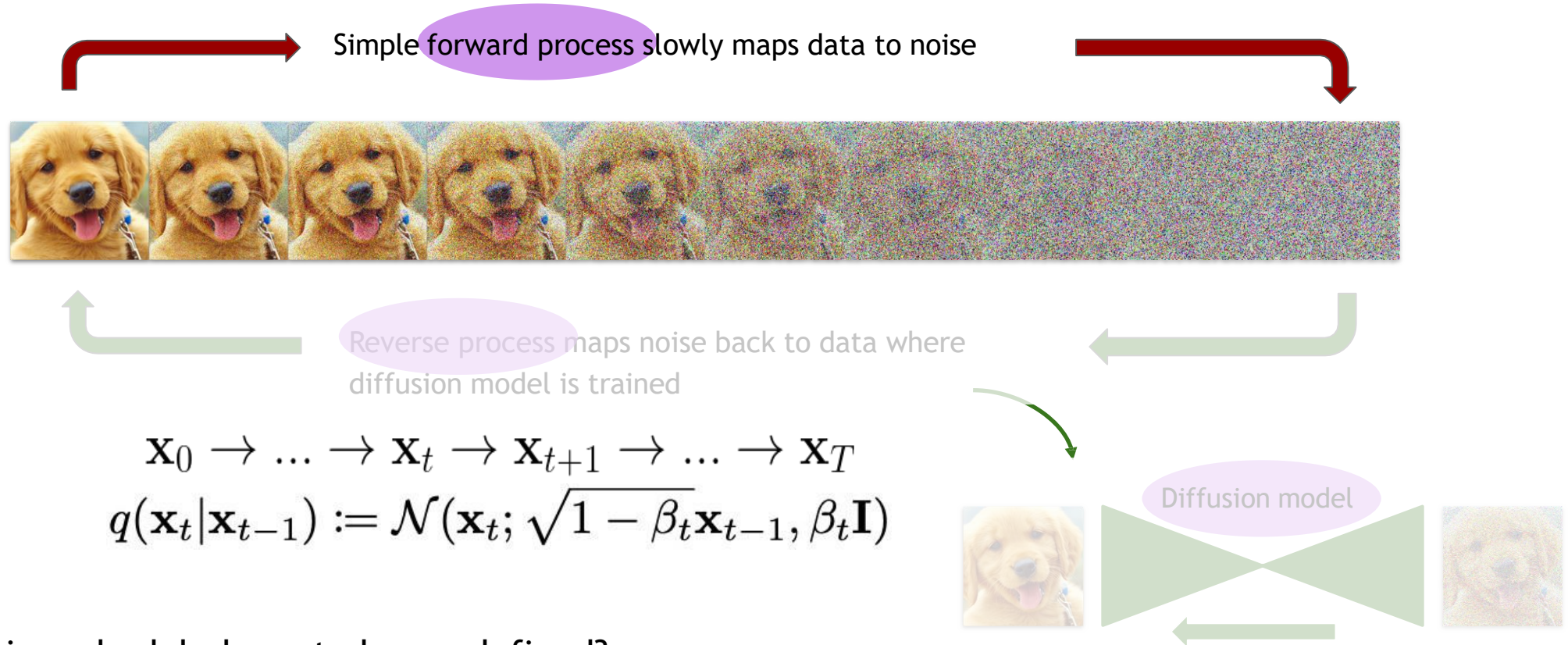
How to accelerate diffusion models?



- 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.

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?

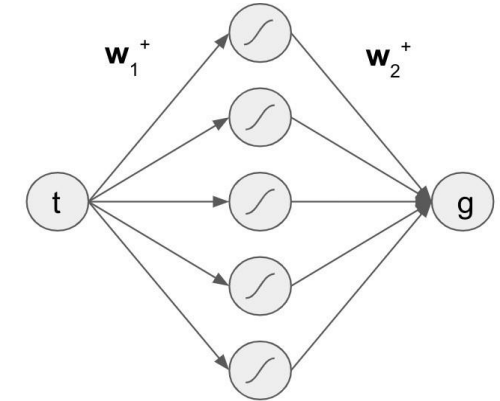
Variational Diffusion Models

Learnable diffusion process

- Given the forward process $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I})$
- Directly parametrize the variance through a learned function γ_η :

$$1 - \bar{\alpha}_t = \text{sigmoid}(\gamma_\eta(t))$$

- $\gamma_\eta(t)$: a monotonic MLP.
 - Strictly positive weights & monotonic activations (e.g. sigmoid)



Variational Diffusion Models

New parametrization of training objectives

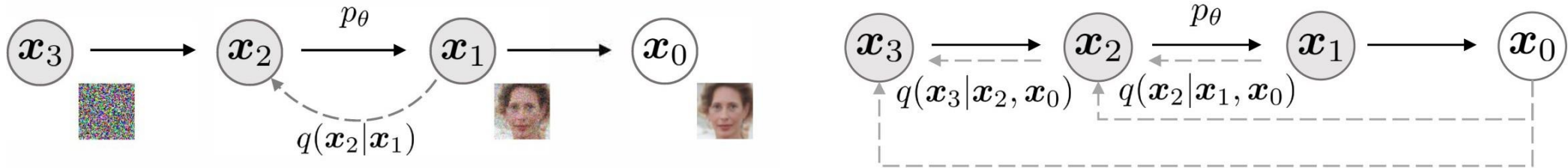
- Optimizing variational upper bound of diffusion models can be simplified to the following training objective:

$$\mathcal{L}_T = \frac{T}{2} \mathbb{E}_{\mathbf{x}_0, \epsilon, t} [(\exp(\gamma_\eta(t) - \gamma_\eta(t-1)) - 1) \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|_2^2]$$

- Learning noise schedule improves likelihood estimation of diffusion models, given fewer diffusion steps.

Denoising Diffusion Implicit Models (DDIM)

Non-Markovian diffusion process



Main Idea

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

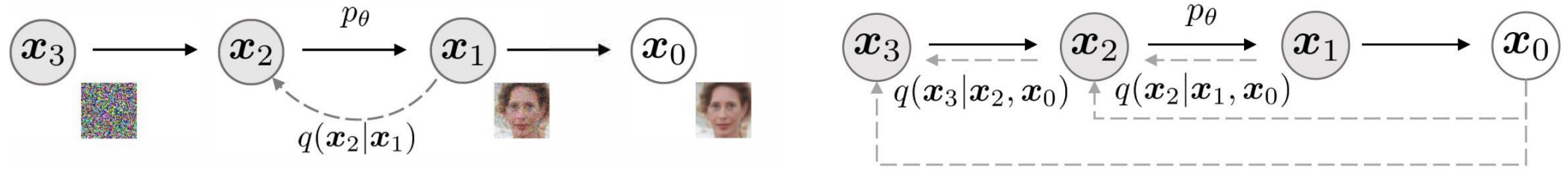
The process is designed such that the model can be optimized by the same surrogate objective as the original diffusion model.

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

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

DDIM Sampler

Deterministic generative process

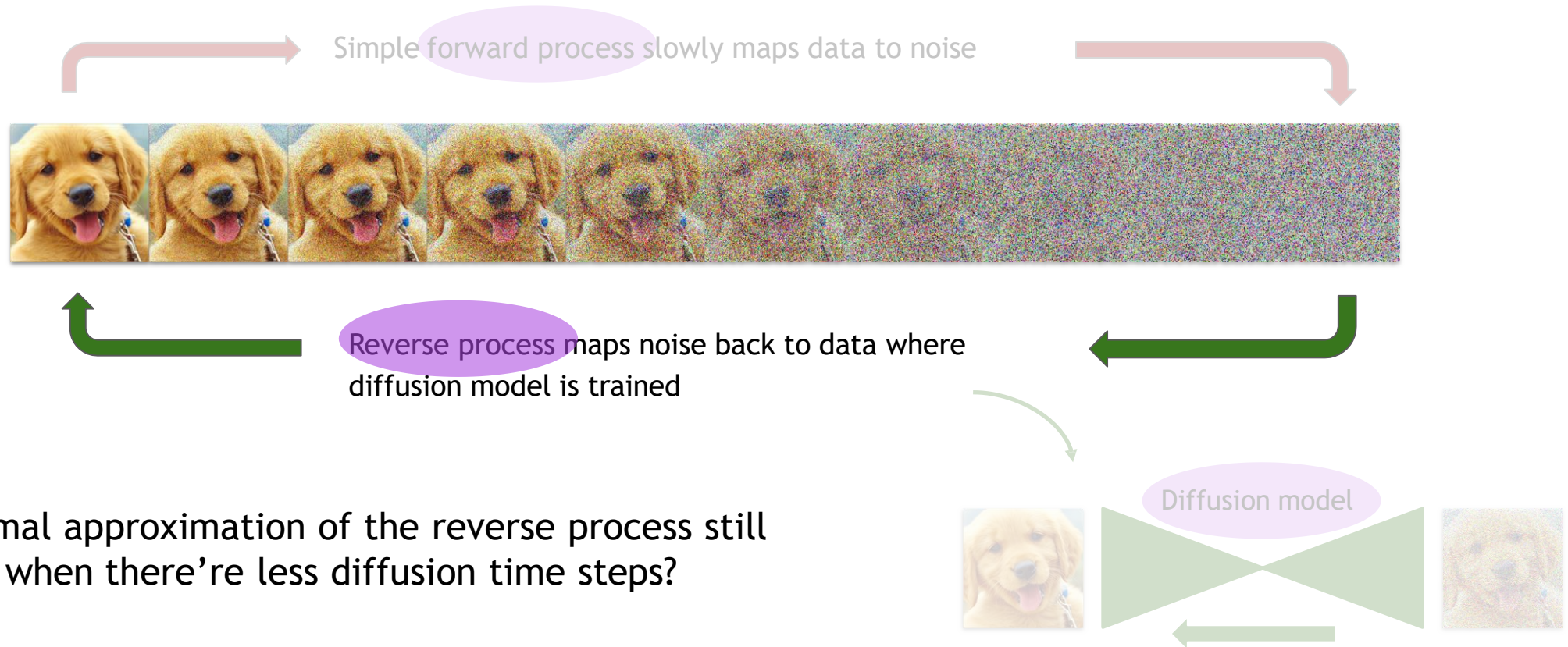


$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{x}}_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \tilde{\sigma}_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\hat{\mathbf{x}}_0}{\sqrt{1 - \bar{\alpha}_t}}, \tilde{\sigma}_t^2 \mathbf{I}\right)$$

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

Advanced Reverse Process

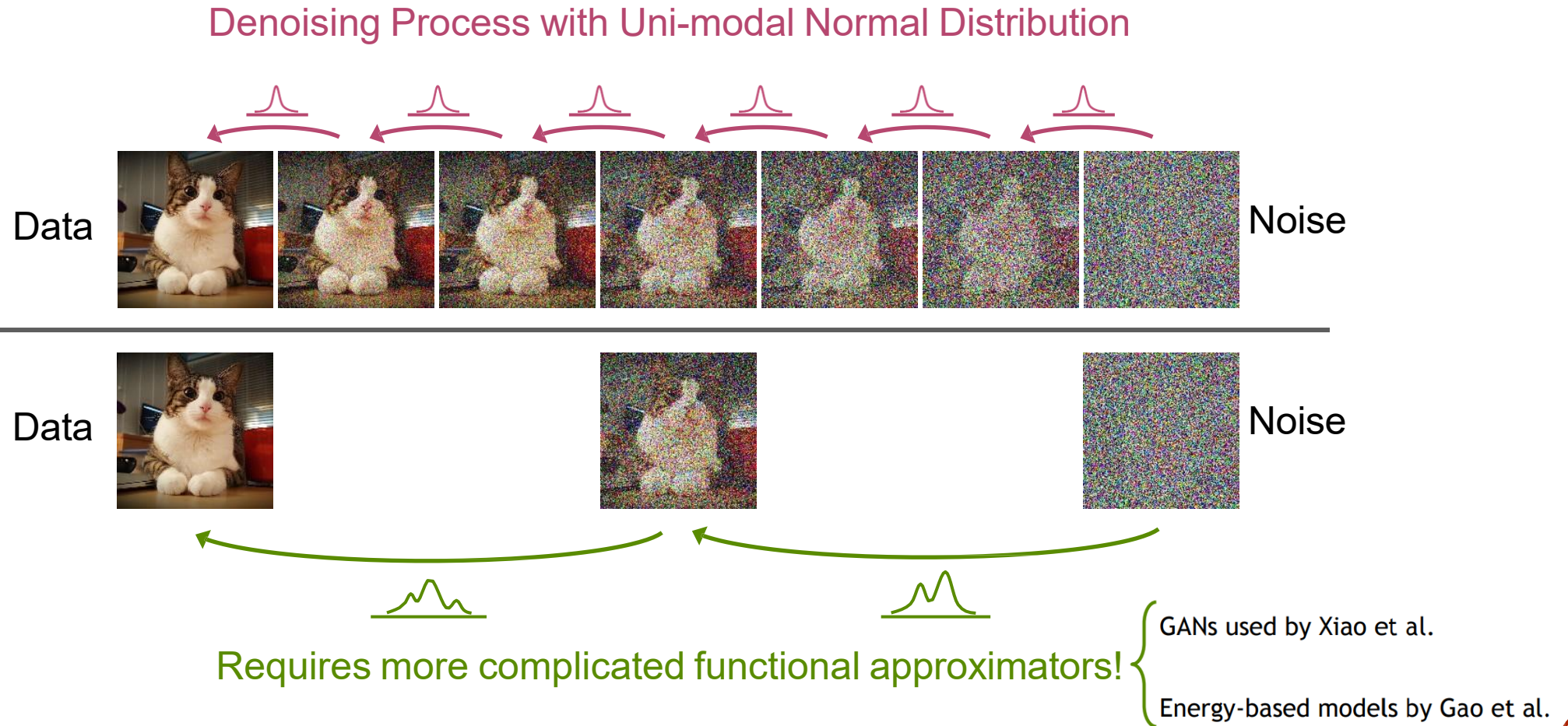
Approximate reverse process with more complicated distributions



- Q: is normal approximation of the reverse process still accurate when there're less diffusion time steps?

Advanced Approximation of Reverse Process

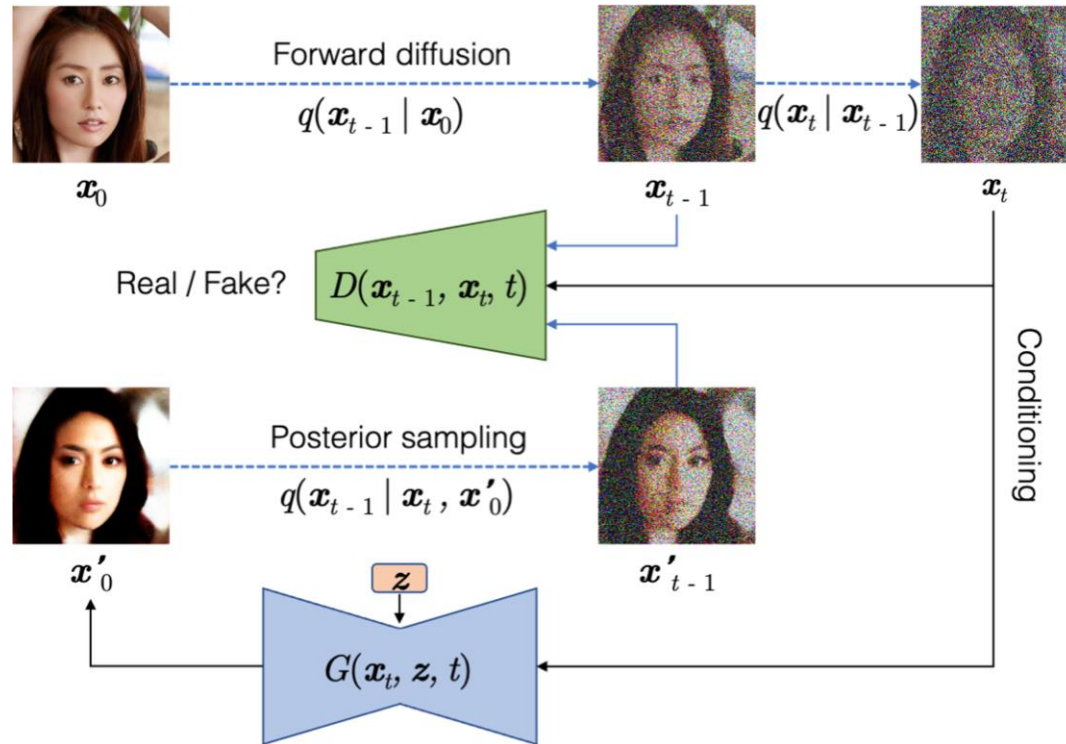
Normal assumption in denoising distribution holds only for small step



Denoising Diffusion GANs

Approximating reverse process by conditional GANs

$$\min_{\theta} \sum_{t \geq 1} \mathbb{E}_{q(\mathbf{x}_t)} [D_{\text{adv}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t) || p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t))]$$

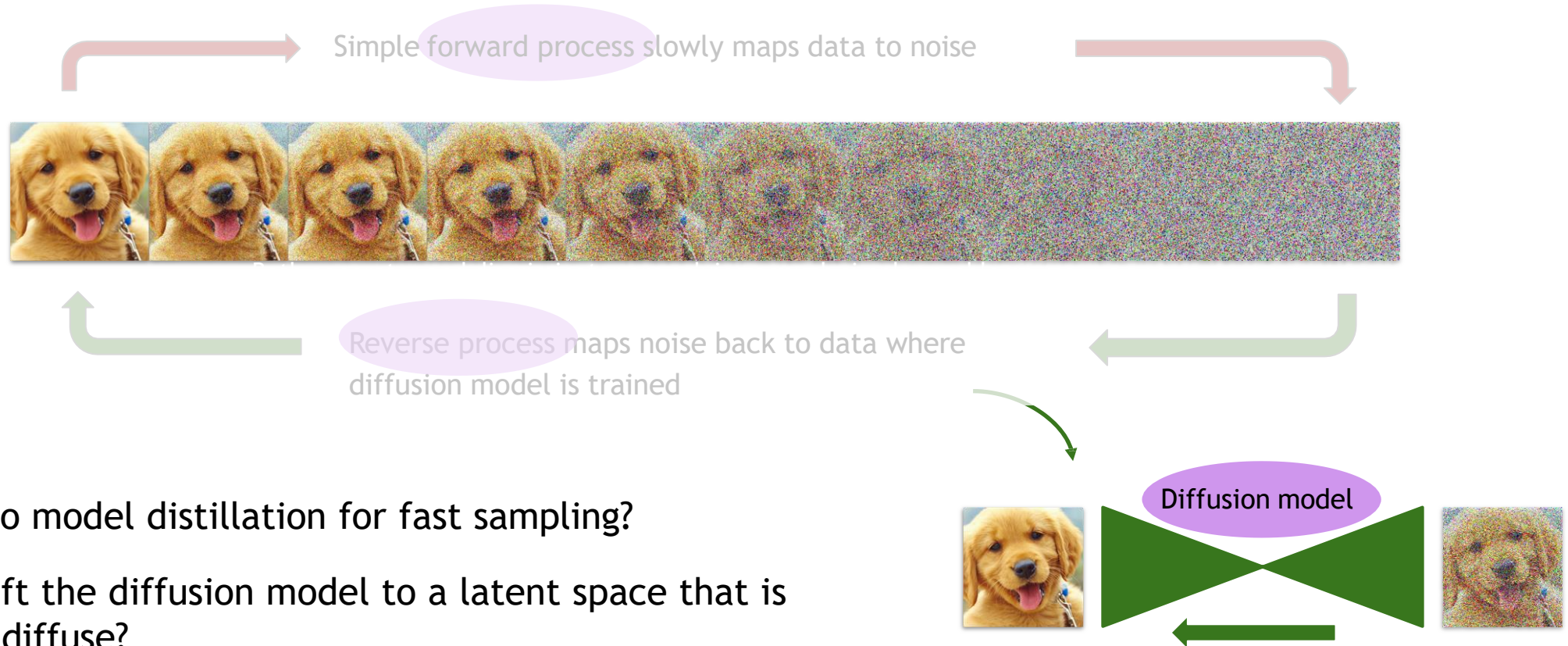


Compared to a one-shot GAN generator:

- Both generator and discriminator are solving a much simpler problem.
- Stronger mode coverage
- Better training stability

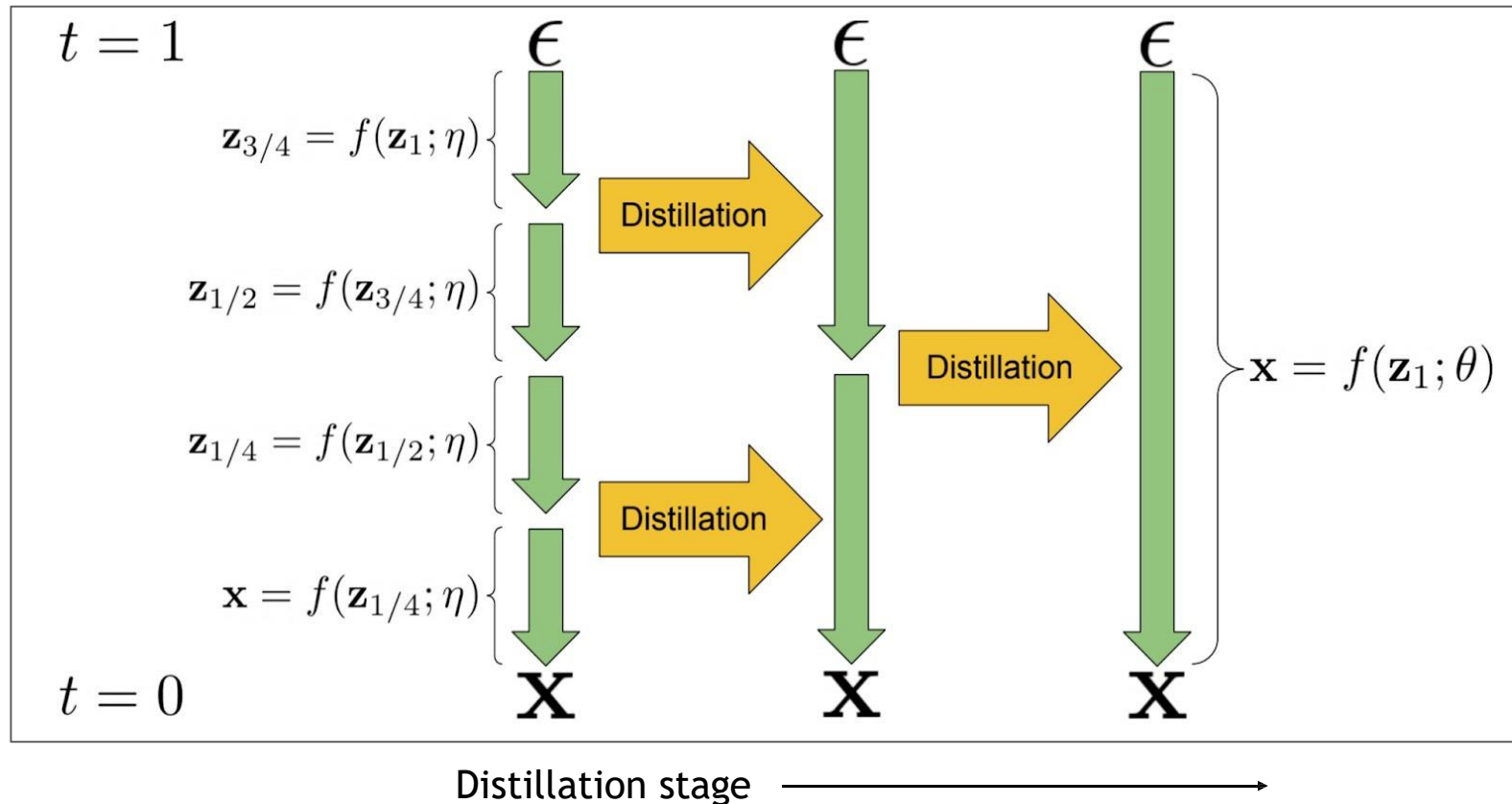
Advanced Modeling

Model Distillation & Latent Space Modeling



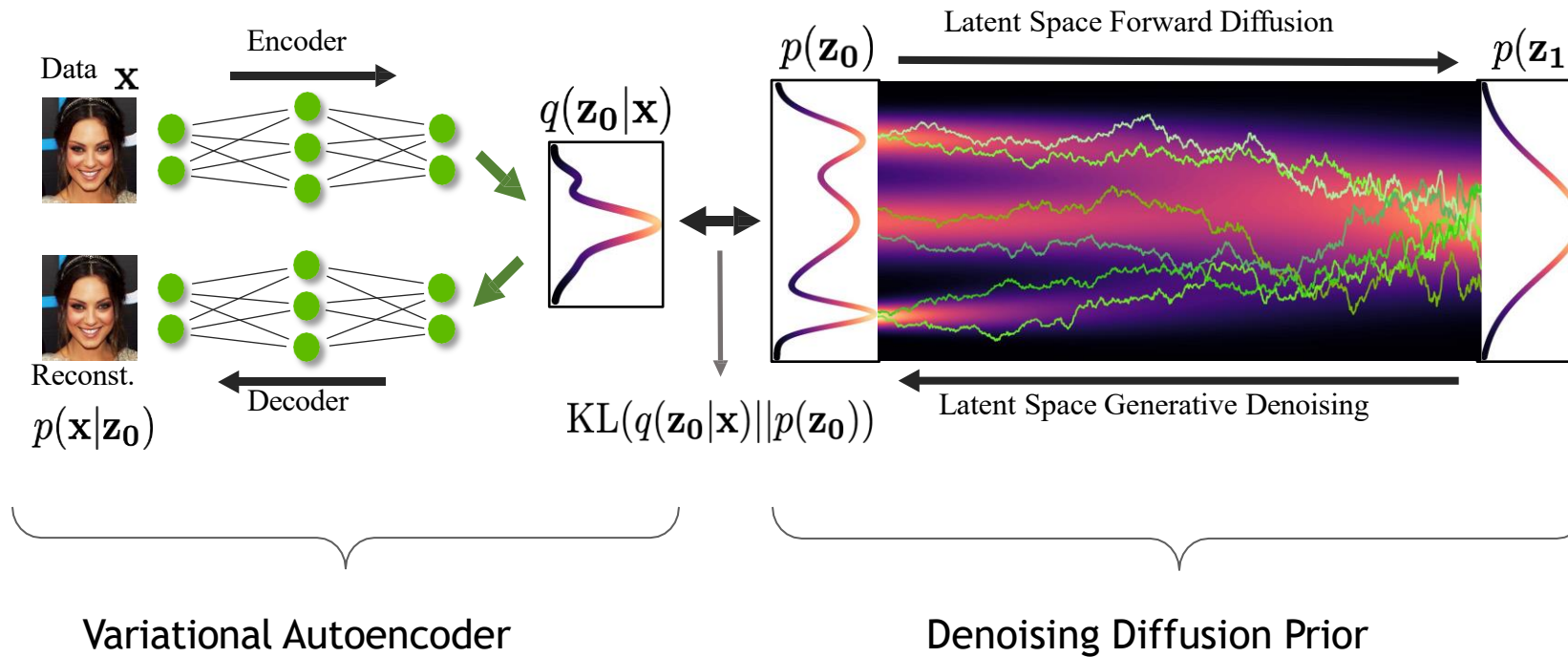
Progressive Distillation

- Distill a deterministic DDIM 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.



Latent-Space Diffusion Models

Variational autoencoder + score-based prior



Main Idea

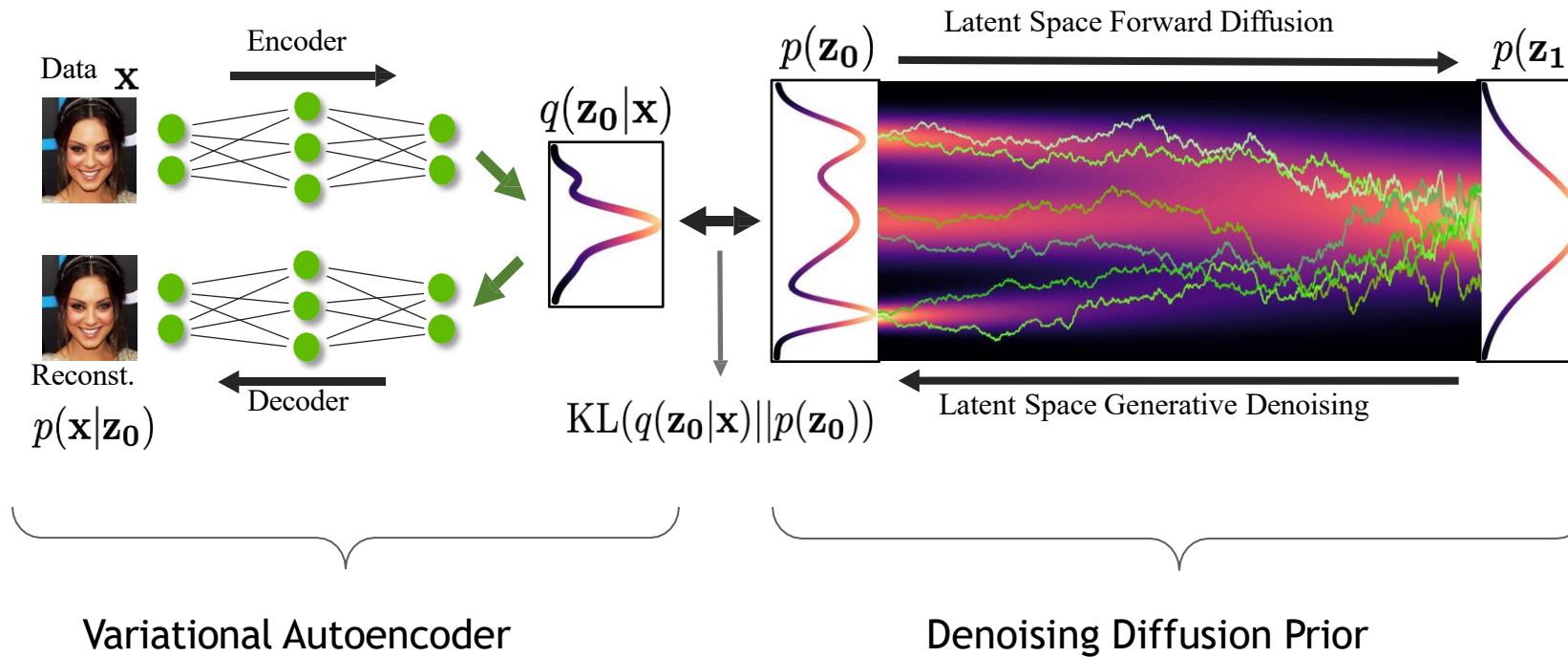
Encoder maps the input data to an embedding space

Denoising diffusion models are applied in the latent space



Latent-Space Diffusion Models

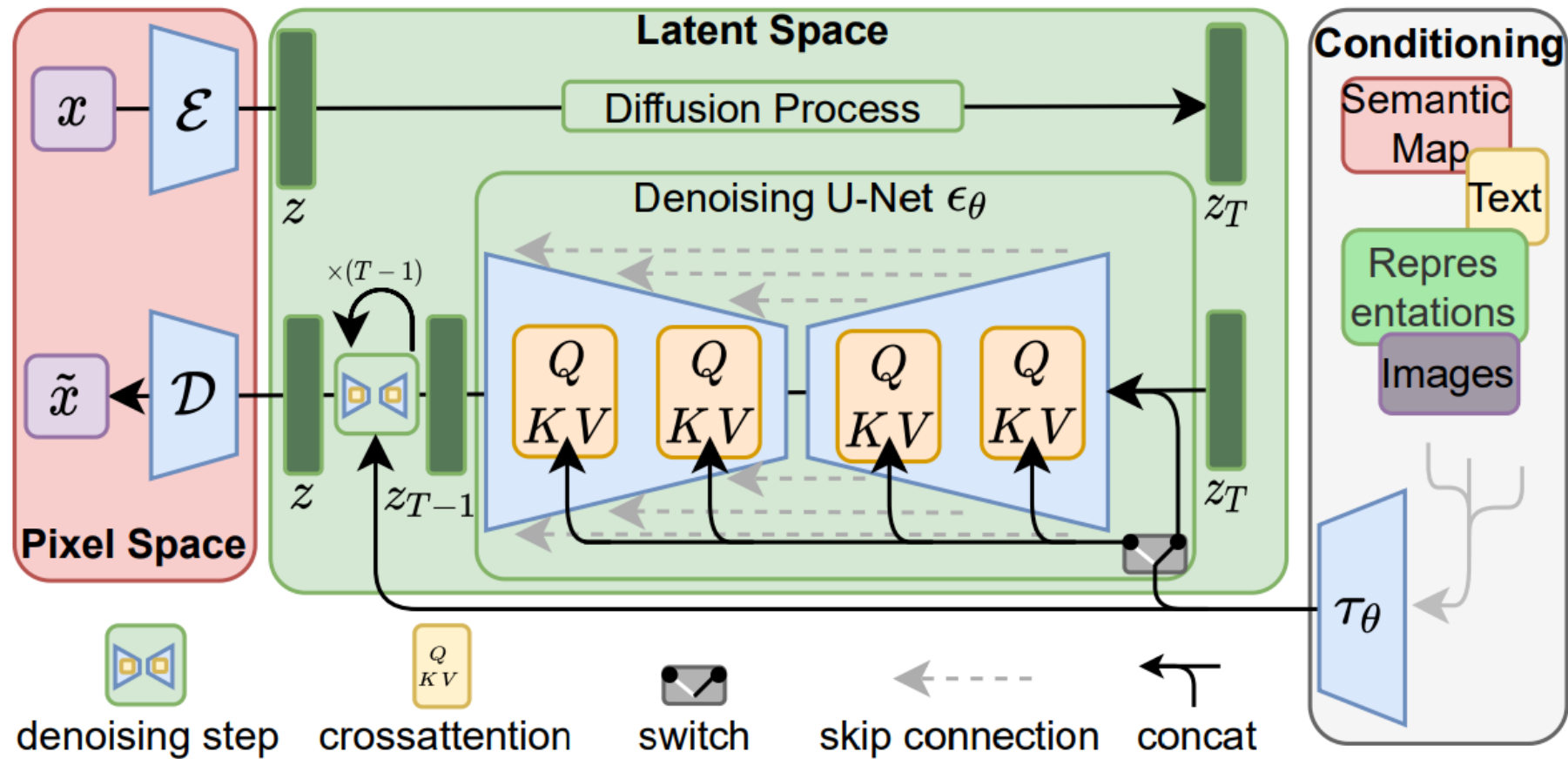
Variational autoencoder + score-based prior



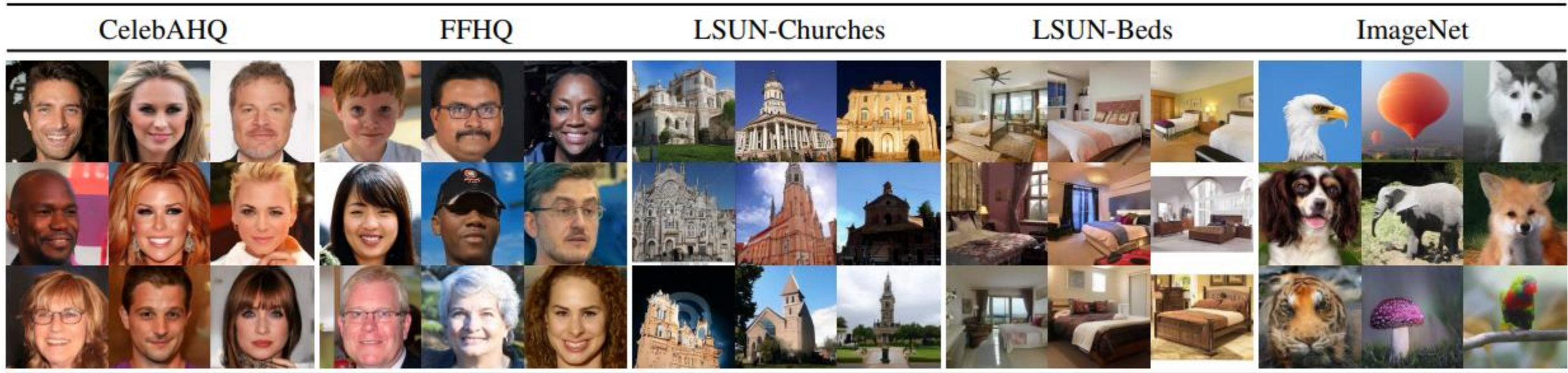
Advantages:

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

Latent-Space Diffusion Models



Latent-Space Diffusion Models: Generated Samples



Thank You

QUESTIONS?

