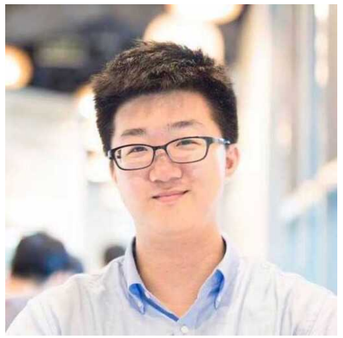
Denoising Diffusion Models: A Generative Learning Big Bang

Jiaming Song



Chenlin Meng

Arash Vahdat

NVIDIA

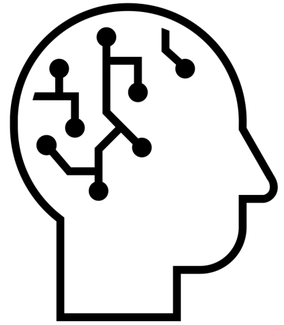
Stanford University

Deep Generative Learning

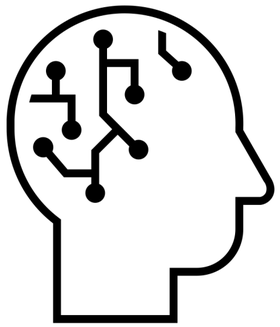
Learning to generate data

Samples from a Data Distribution

Train



Neural Network



Sample

Generative Learning Applications

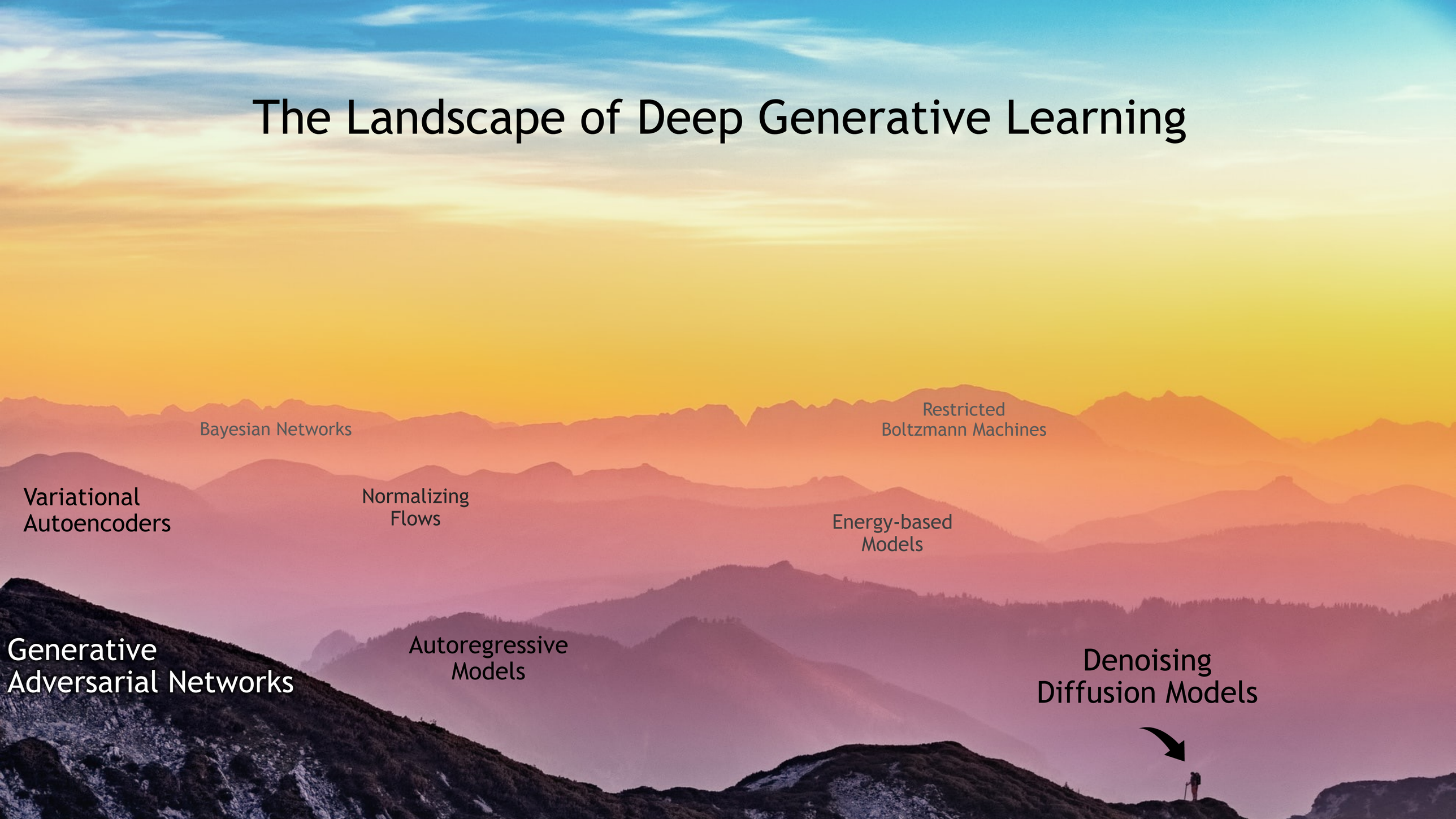
Art & Design

Content Generation

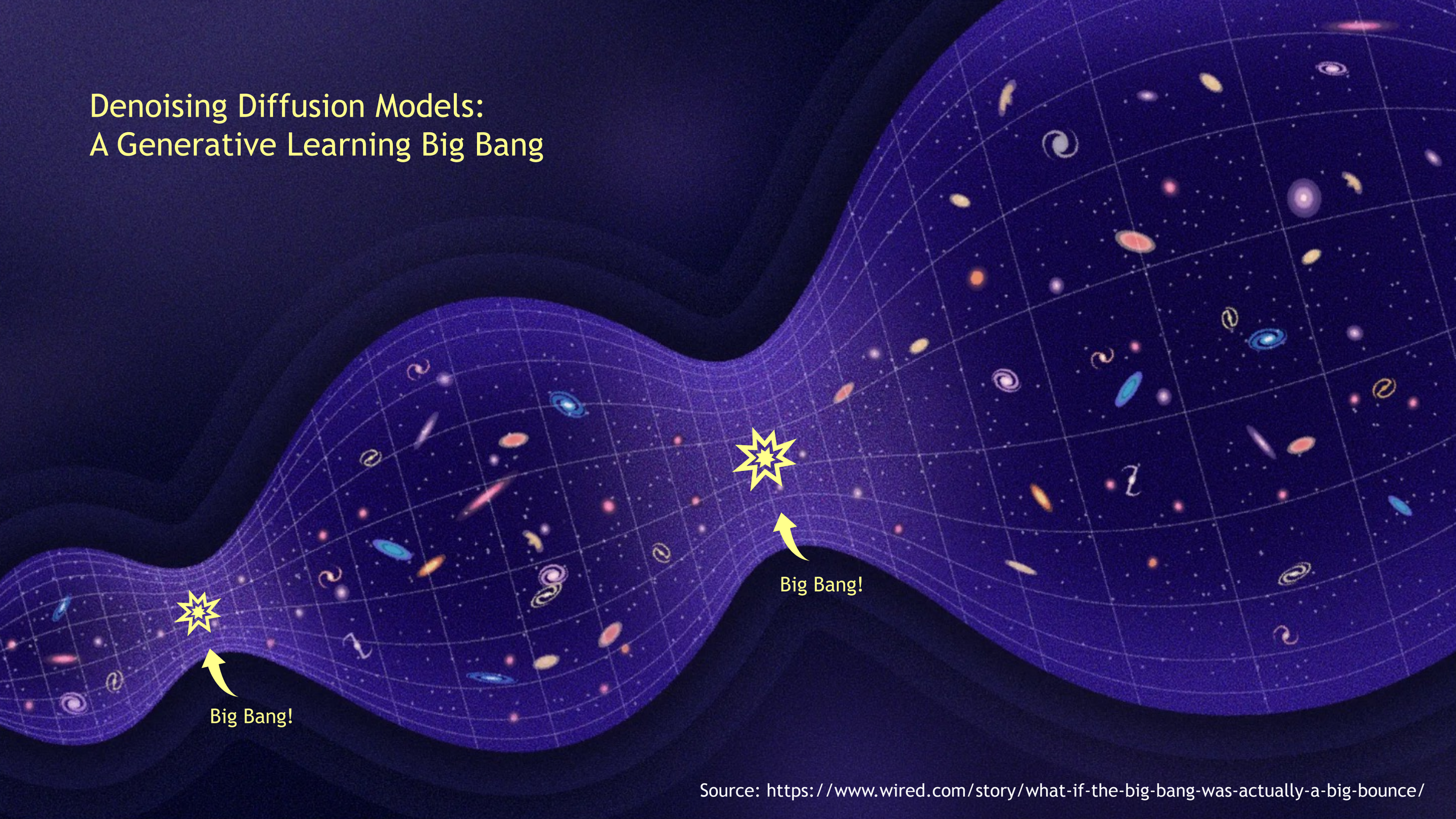
Representation Learning

Entertainment

The Landscape of Deep Generative Learning

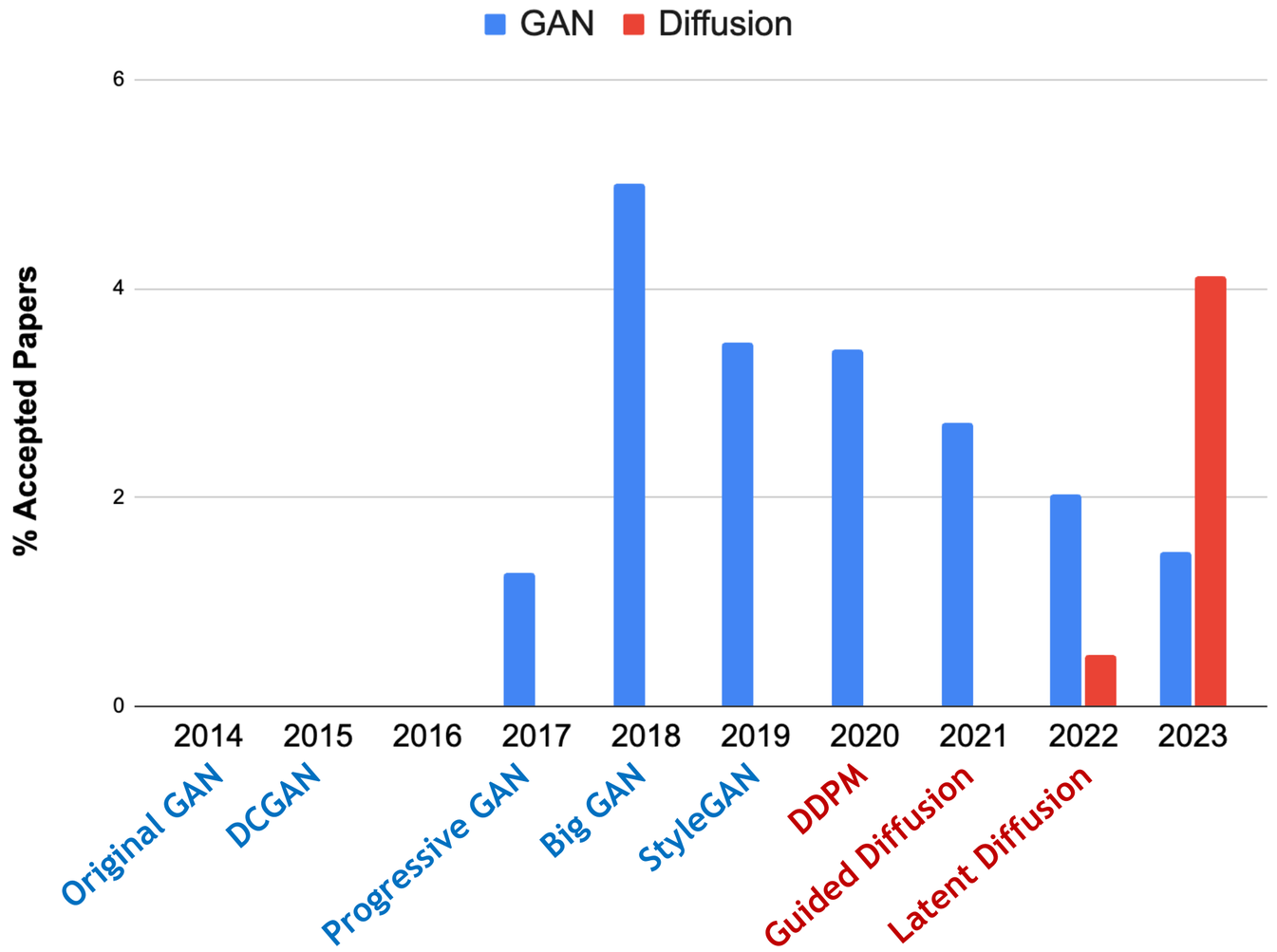


Denoising Diffusion Models: A Generative Learning Big Bang



Source: https://www.wired.com/story/what-if-the-big-bang-was-actually-a-big-bounce/

We May Not Know Cosmology, But We Know CVPR



*Disclaimer: We rely on paper titles for counting the number of papers in each topic. Our statistics are likely to be biased.

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

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

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

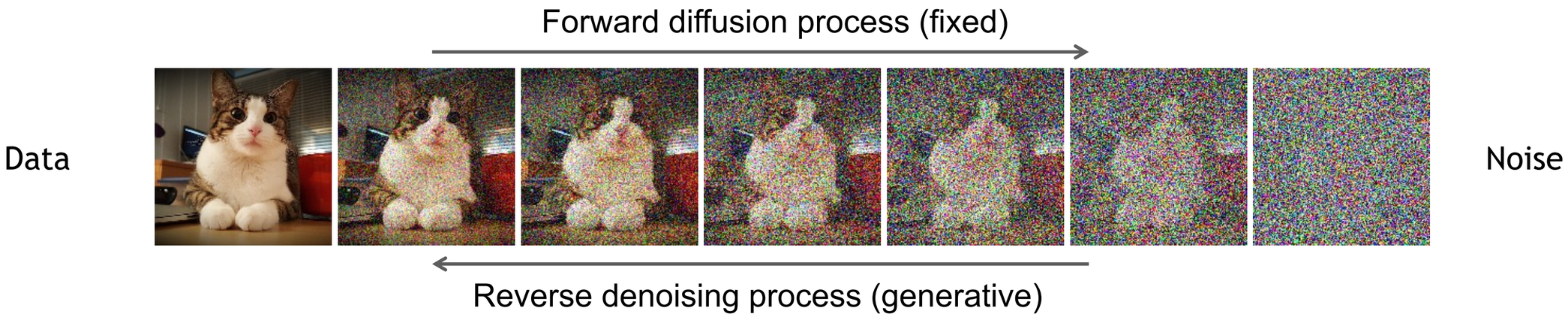
Part (4): Conditional Generation and Guidance

Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

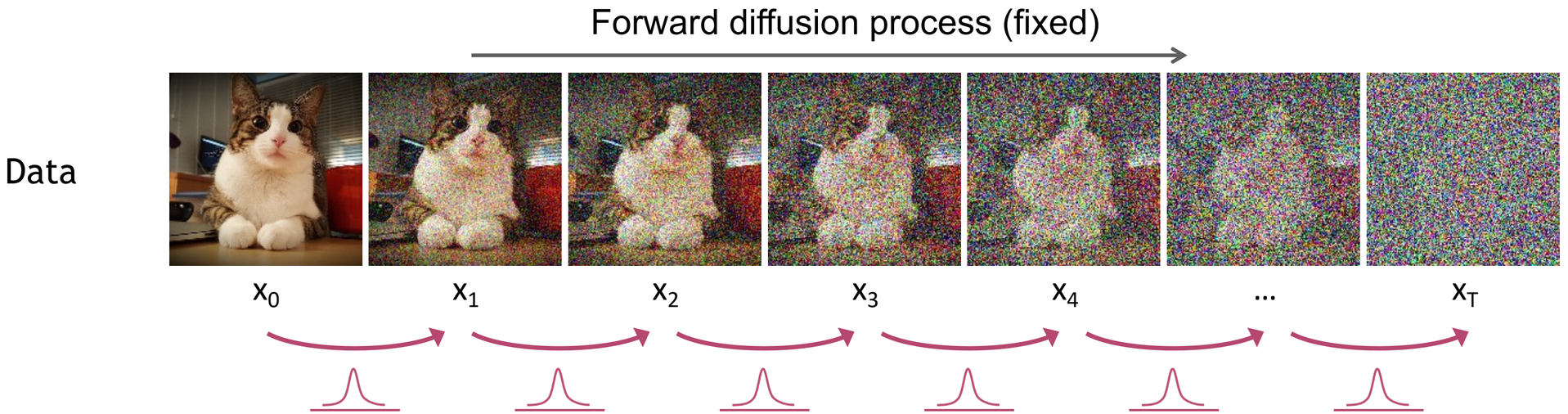
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



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

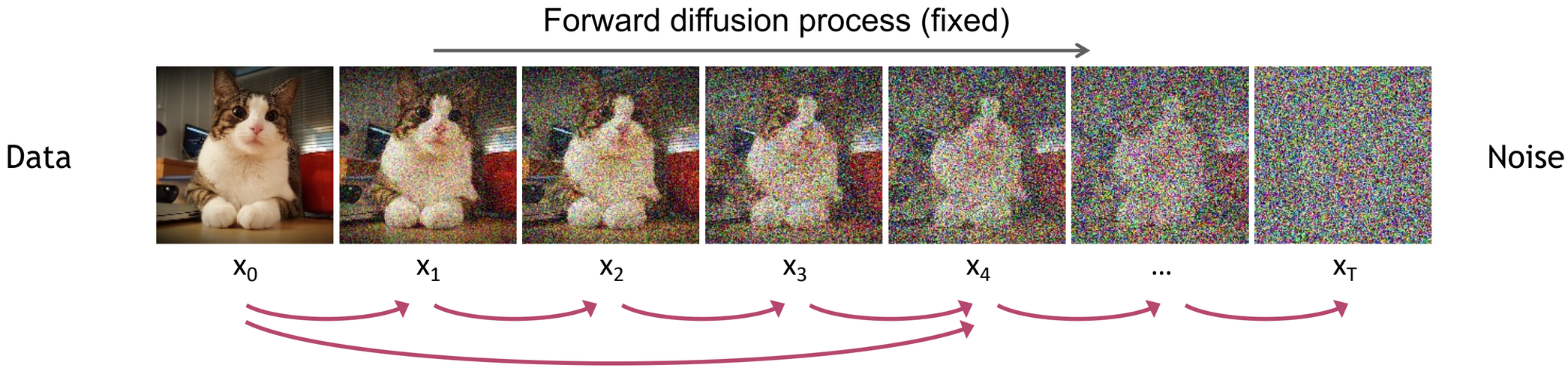
Forward Diffusion Process

The formal definition of the forward process in T steps:



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

Diffusion Kernel



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

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

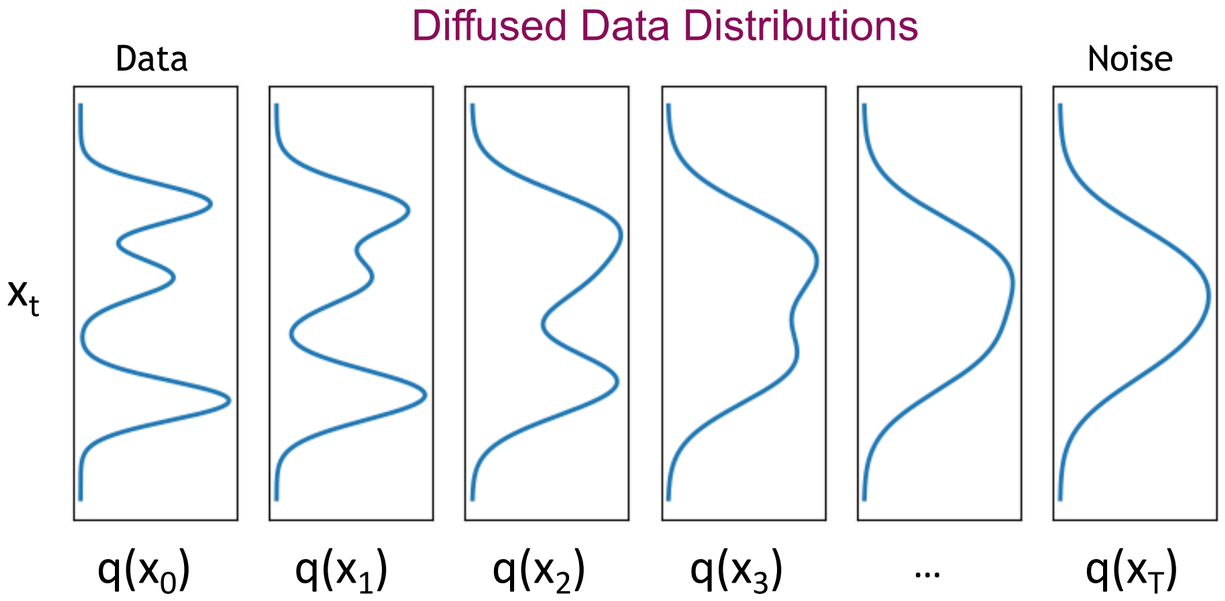
β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

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 $q(\mathbf{x}_t)$?

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

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

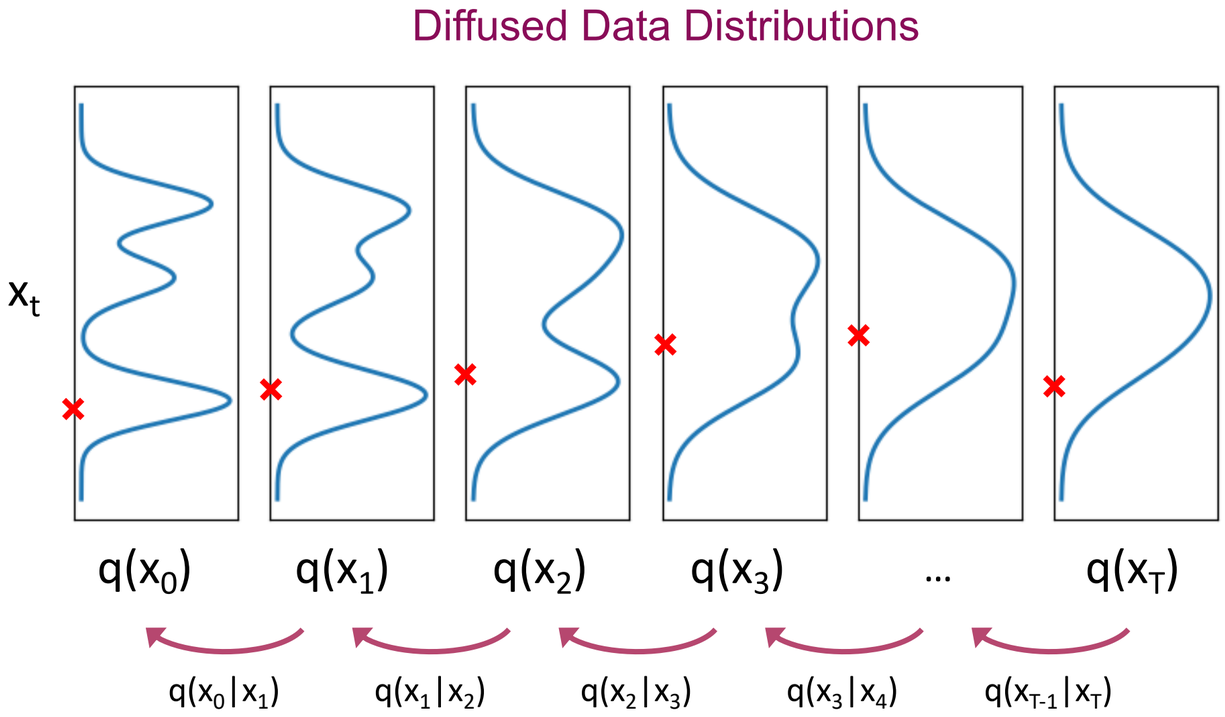
Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Generation:

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

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}$

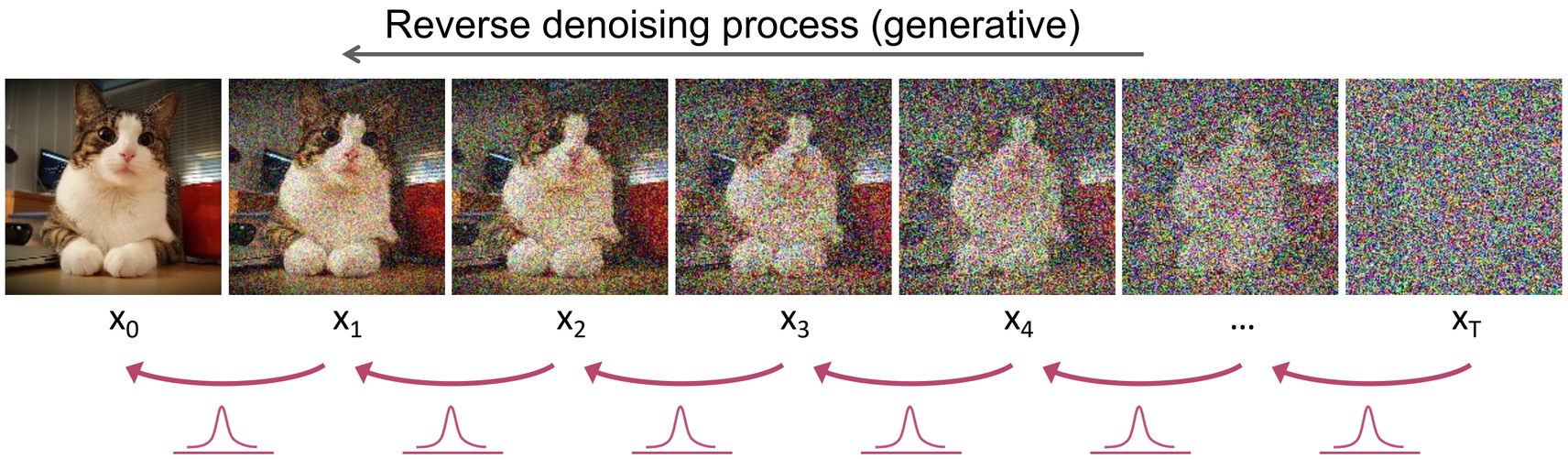
True Denoising Dist.



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:



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

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

Trainable network
(U-net, Denoising Autoencoder)

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Learning Denoising Model

Variational upper bound

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

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

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

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

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

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

Ho et al. NeurIPS 2020 observe that simply setting λ_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: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

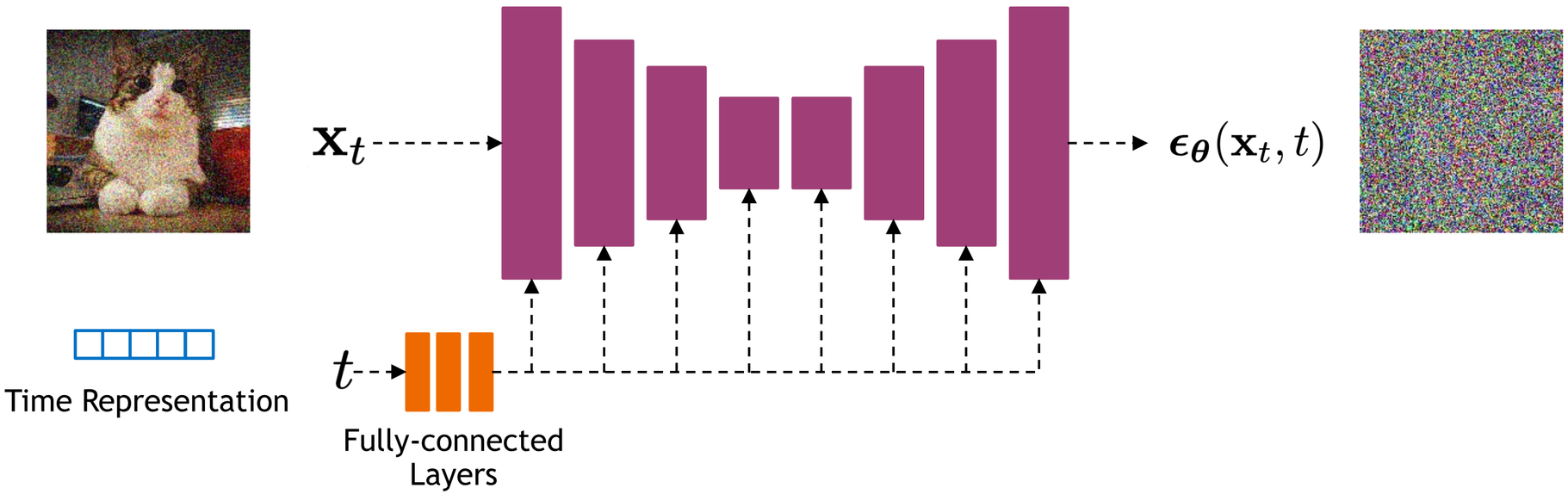
Algorithm 2 Sampling

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

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

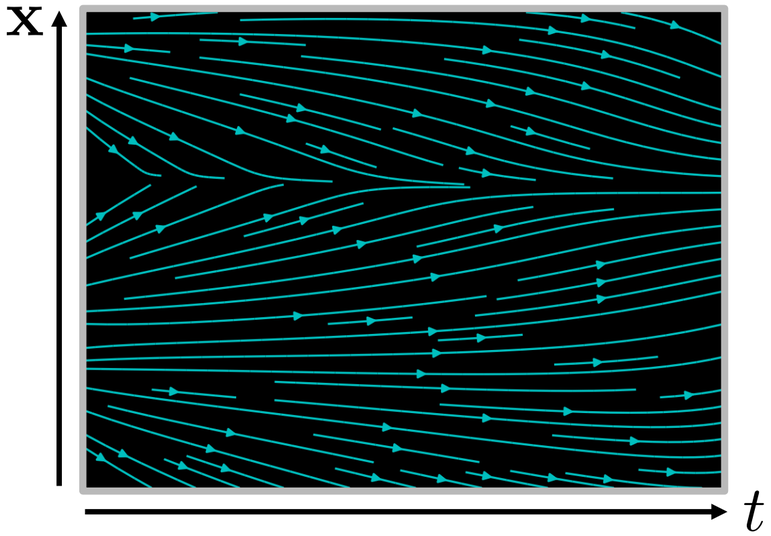
Part (3): Accelerated Sampling

Part (4): Conditional Generation and Guidance

Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

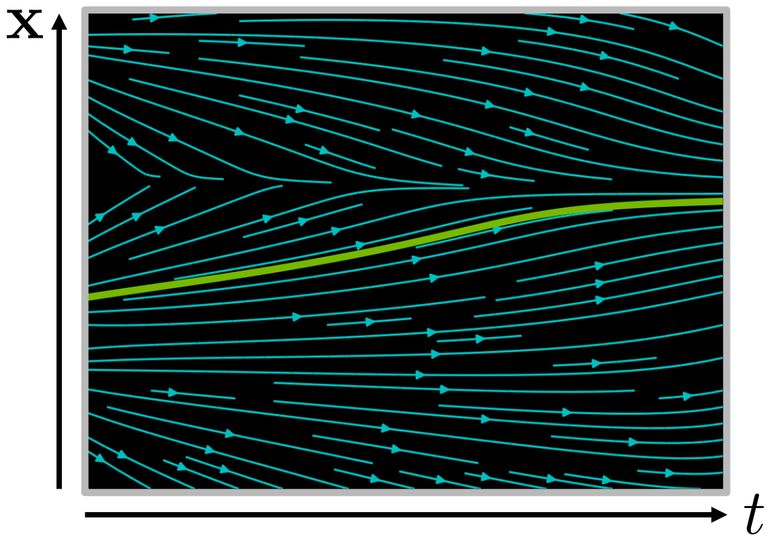
$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

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



Analytical Solution:

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

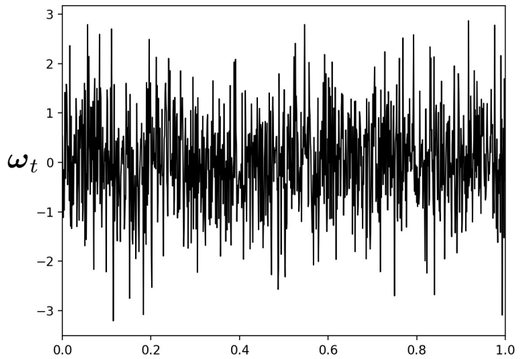
Iterative Numerical Solution:

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

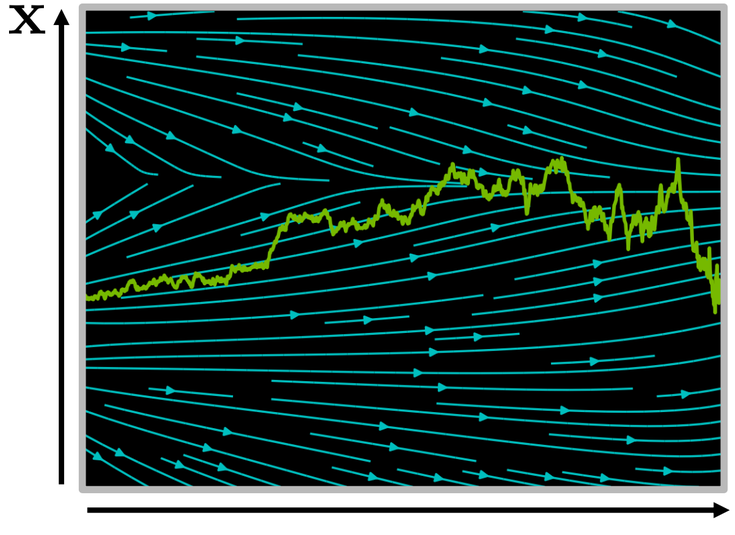
Stochastic Differential Equation (SDE):

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

Wiener Process (Gaussian White Noise)



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

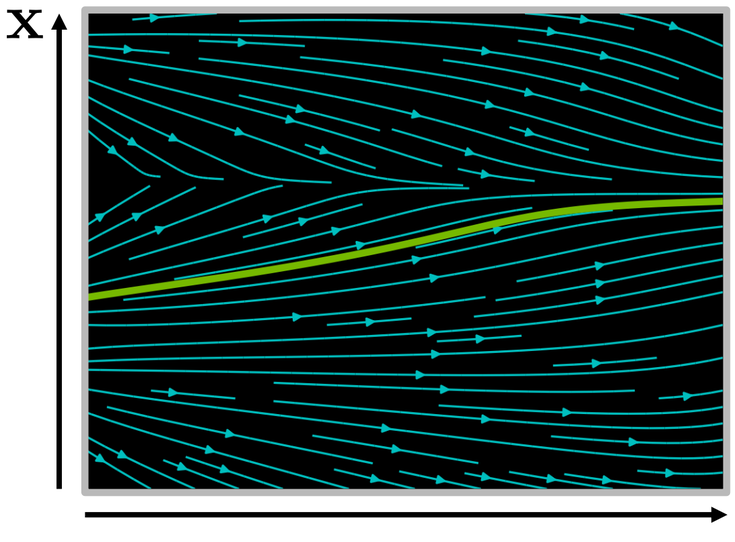


$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

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



Analytical Solution:

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

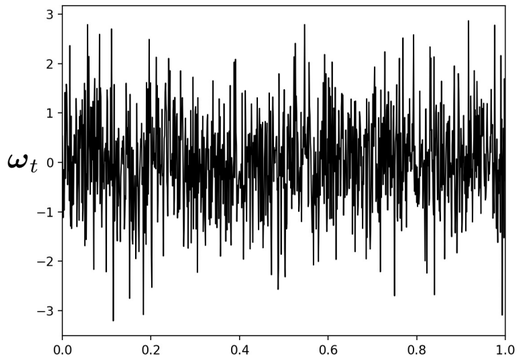
Iterative Numerical Solution:

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

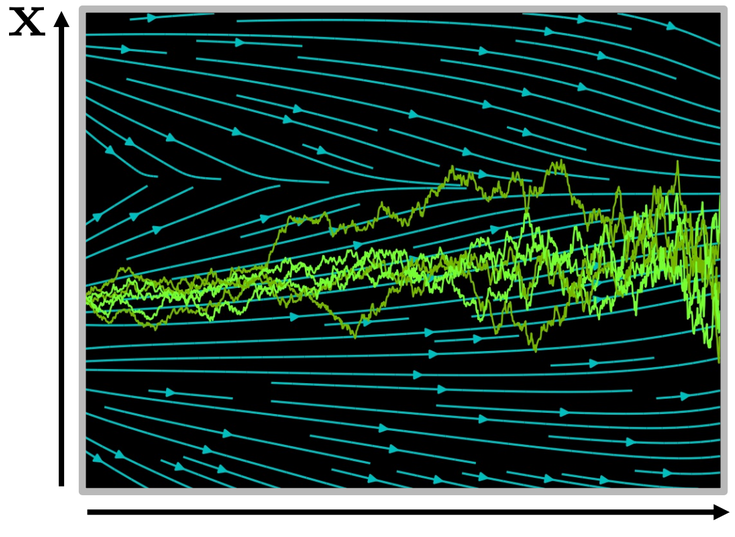
Stochastic Differential Equation (SDE):

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

Wiener Process (Gaussian White Noise)



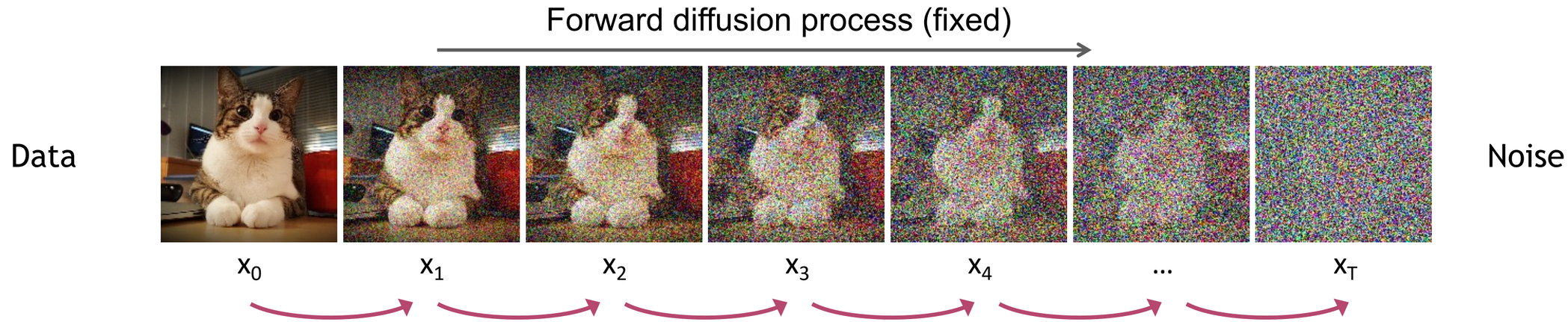
$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \sigma(\mathbf{x}, t)\mathrm{d}\boldsymbol{\omega}_t \right)$$



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

Forward Diffusion Process as Stochastic Differential Equation

Consider the limit of **many small steps:**



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

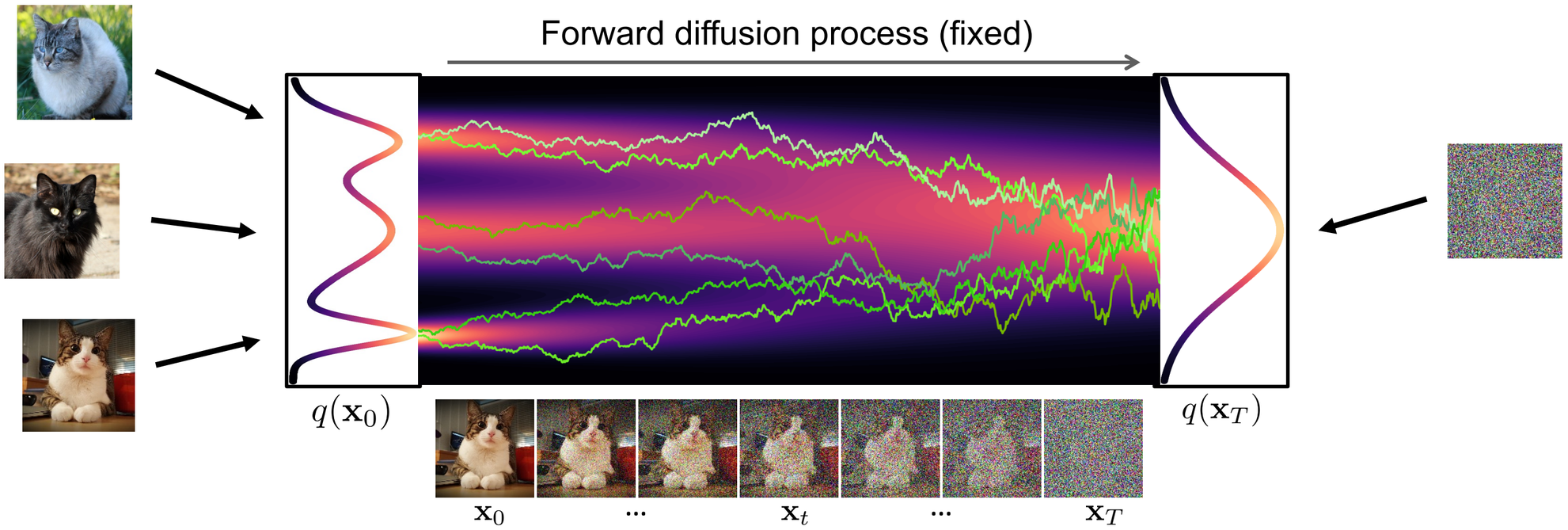
discretization of

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

Stochastic Differential Equation (SDE)

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

Forward Diffusion Process as Stochastic Differential Equation

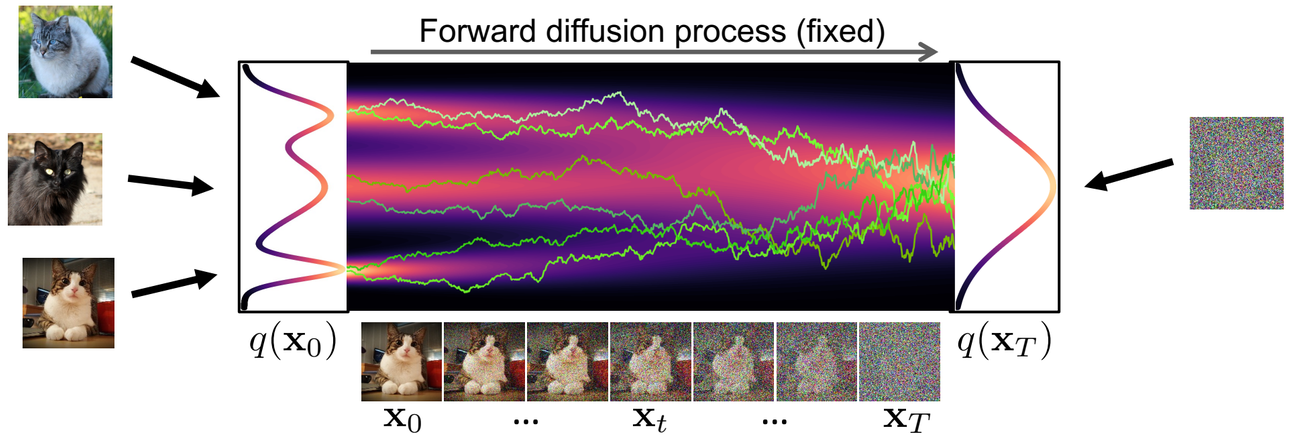


Forward Diffusion SDE:

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

Song et al., *ICLR*, 2021

The Generative Reverse Stochastic Differential Equation



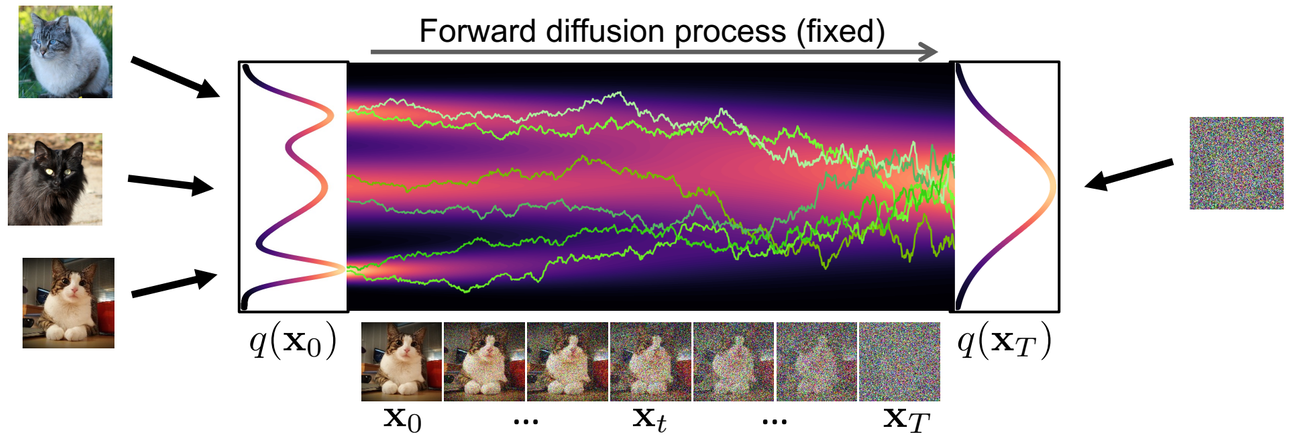
Forward Diffusion SDE:

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

But what about the reverse direction, necessary for generation?

Song et al., *ICLR*, 2021

The Generative Reverse Stochastic Differential Equation



Forward Diffusion SDE:

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

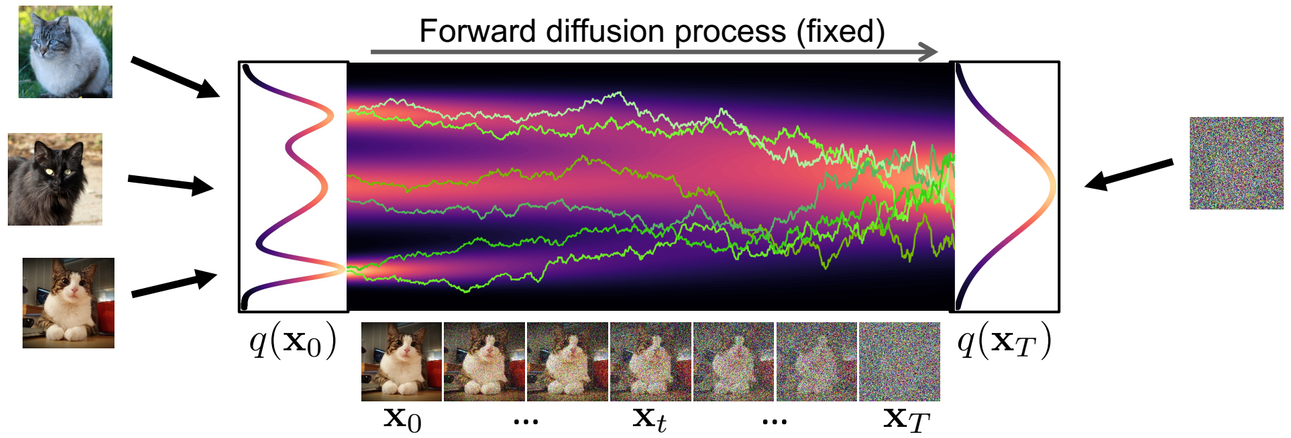
Reverse Generative Diffusion SDE:

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

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

Song et al., *ICLR*, 2021
Anderson, in *Stochastic Processes and their Applications*, 1982

The Generative Reverse Stochastic Differential Equation



Forward Diffusion SDE:

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

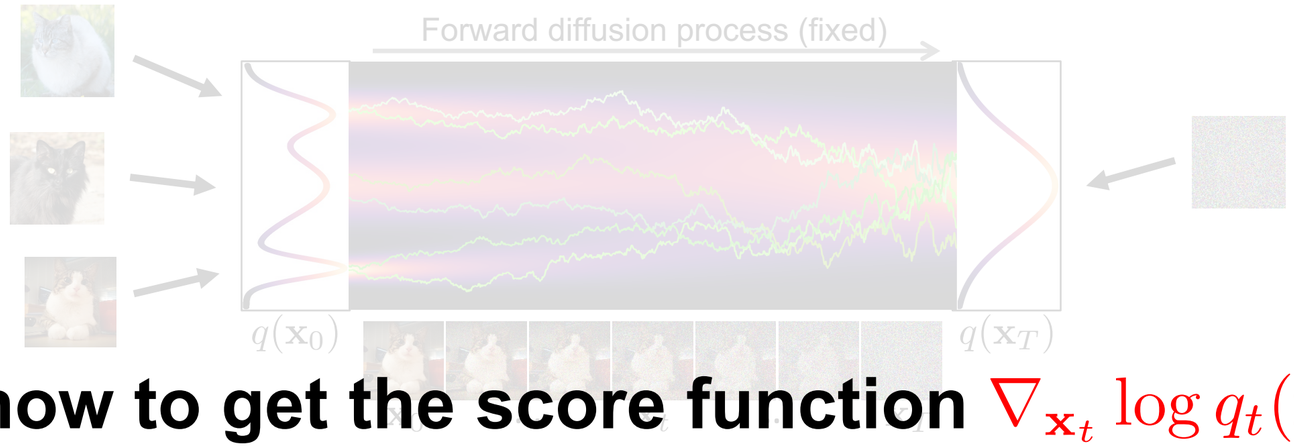
Reverse Generative Diffusion SDE:

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

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

Song et al., *ICLR*, 2021
Anderson, in *Stochastic Processes and their Applications*, 1982

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:

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

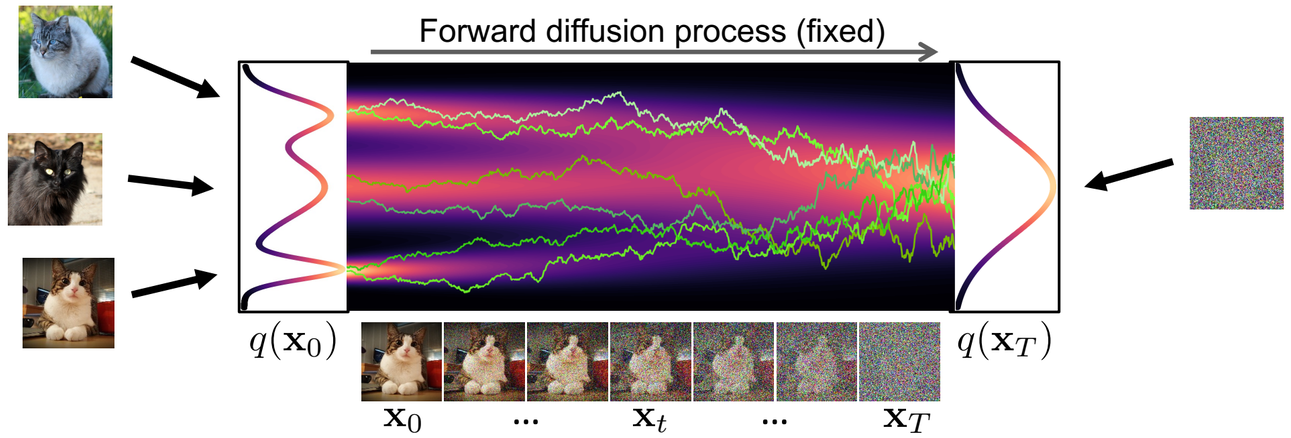
Reverse Generative Diffusion SDE:

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

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

Song et al., *ICLR*, 2021
Anderson, in *Stochastic Processes and their Applications*, 1982

Score Matching



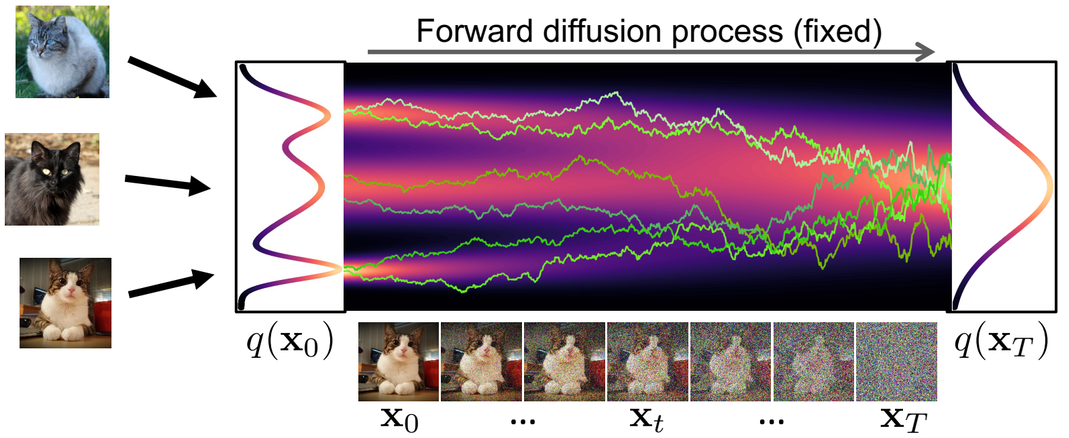
- Naïve idea, learn model for the score function by direct regression?

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

➡ **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!**

Vincent, "A Connection Between Score Matching and Denoising Autoencoders", *Neural Computation*, 2011
Song and Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution", *NeurIPS*, 2019

Denoising Score Matching



"Variance Preserving" SDE:

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

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

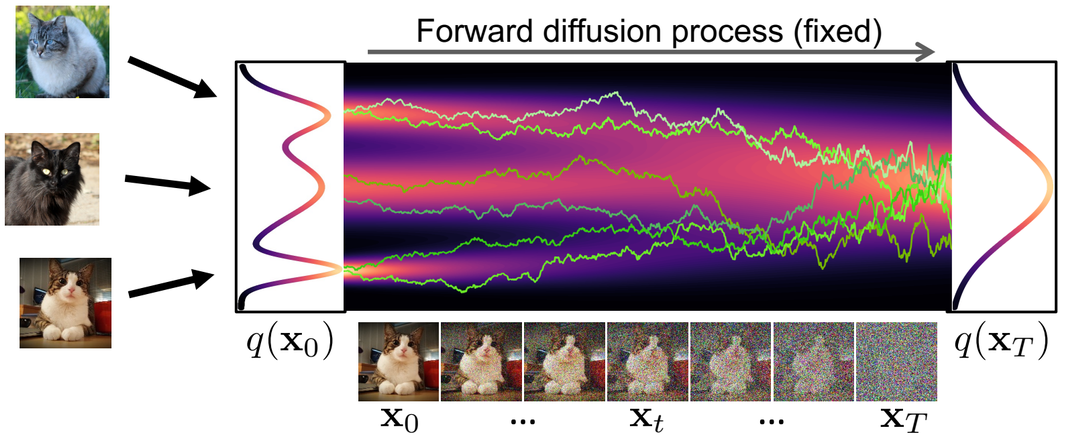
- Instead, diffuse individual data points $\mathbf{x}_0$. Diffused $q_t(\mathbf{x}_t|\mathbf{x}_0)$ ***is*** tractable!
- **Denoising Score Matching:**

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

➡ **After expectations,** $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)$**!**

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

Denoising Score Matching



"Variance Preserving" SDE:

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

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

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

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

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

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

Same objectives in Part (1)!

Different Parameterizations

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

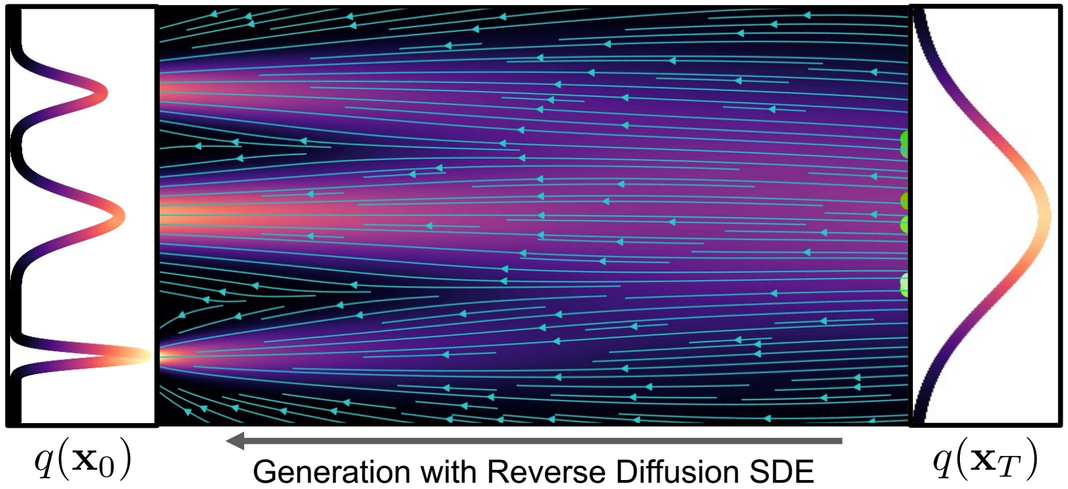
Denoising network: $\min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,T),\mathbf{x}_0\sim q_0(\mathbf{x}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\lambda'(t)||\mathbf{x}_0-\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2$

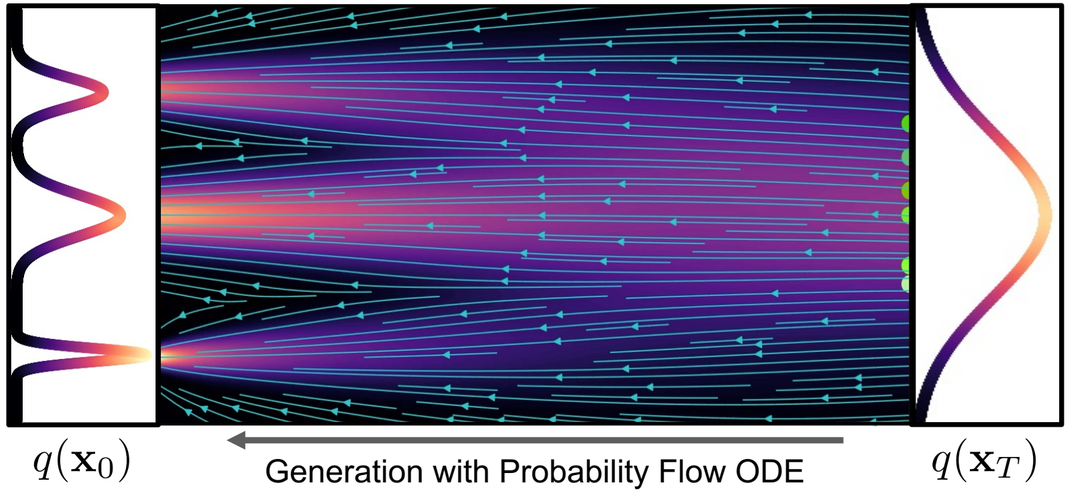
v-prediction: $\mathbf{v} := \gamma_t\boldsymbol{\epsilon} - \sigma_t\mathbf{x_0}$ $\min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,T),\mathbf{x}_0\sim q_0(\mathbf{x}_0),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\lambda'(t)||\mathbf{v}-\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2$

More sophisticated model parametrizations and loss weightings possible!

Karras et al., "Elucidating the Design Space of Diffusion-Based Generative Models", NeurIPS 2022

Synthesis with SDE vs. ODE





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

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

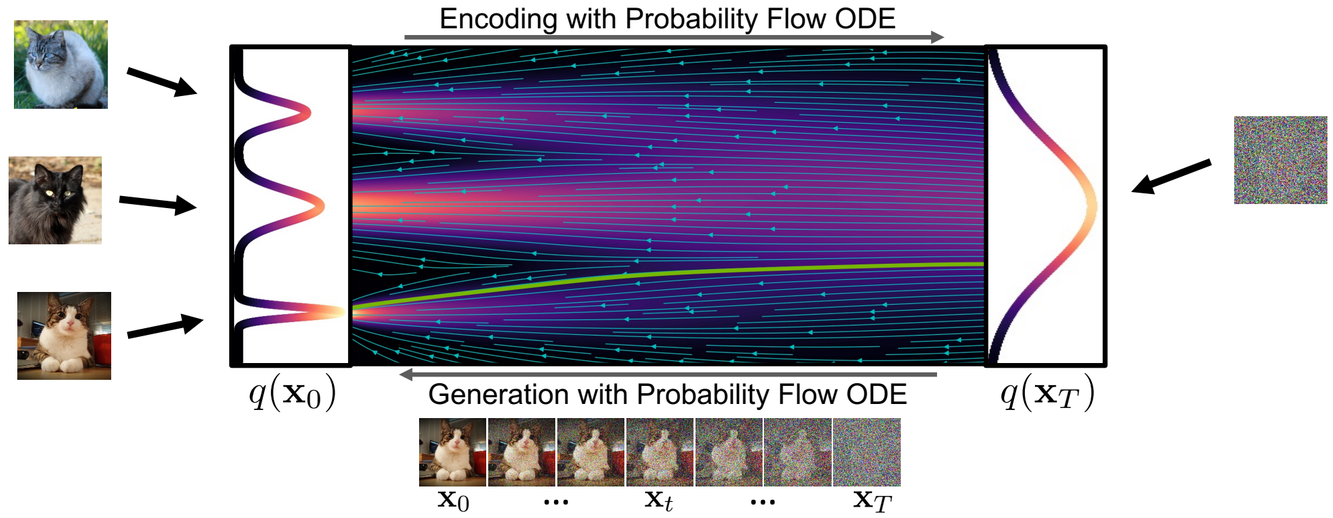
- **Generative Probability Flow ODE (deterministic):**

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

Song et al., *ICLR*, 2021

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):

Chen et al., *NeurIPS*, 2018
Grathwohl, *ICLR*, 2019
Song et al., *ICLR*, 2021

Outline

Part (1): Denoising Diffusion Probabilistic Models

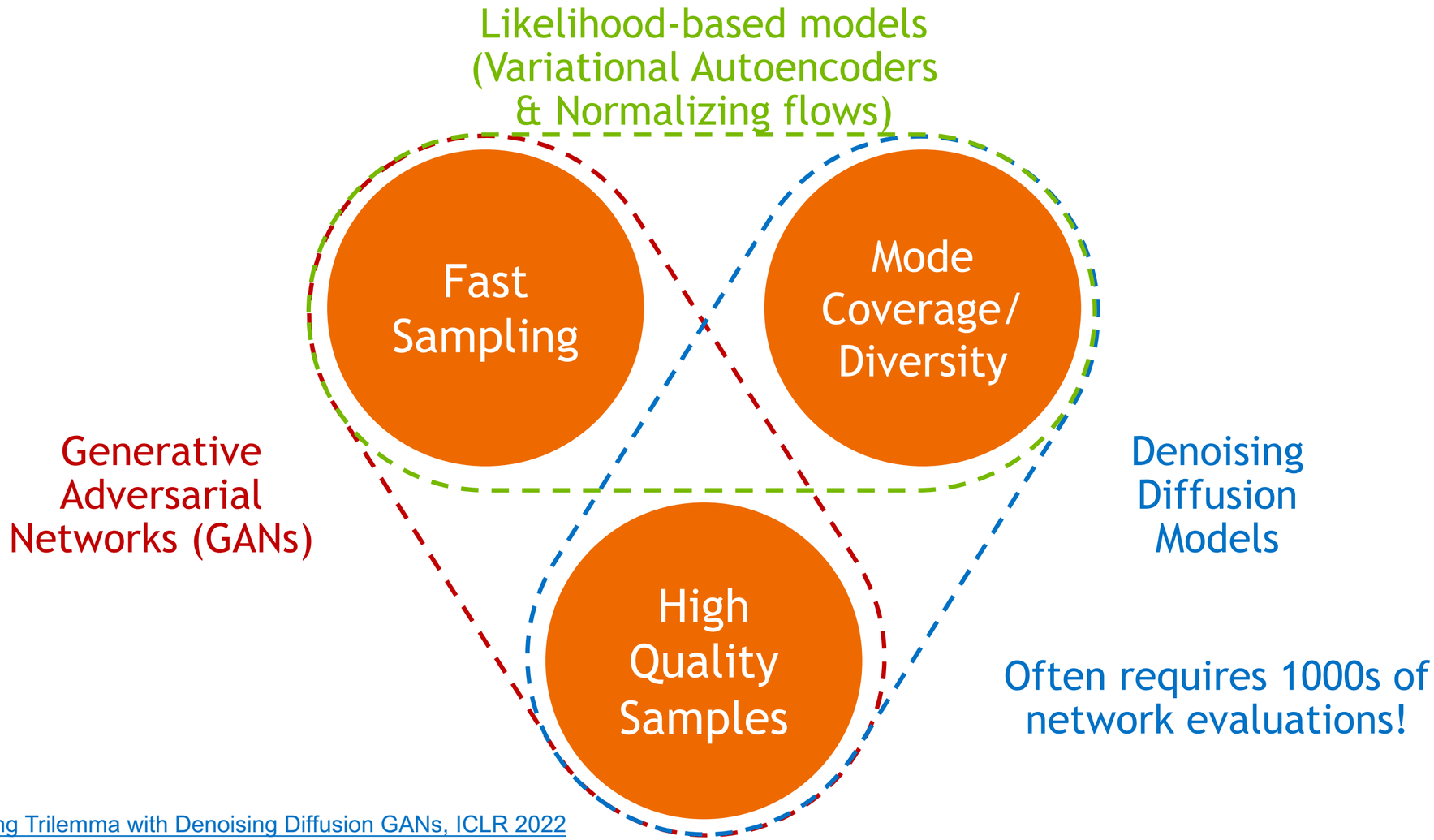
Part (2): Score-based Generative Modeling with Differential Equations

Part (3): Accelerated Sampling

Part (4): Conditional Generation and Guidance

What makes a good generative model?

The generative learning trilemma

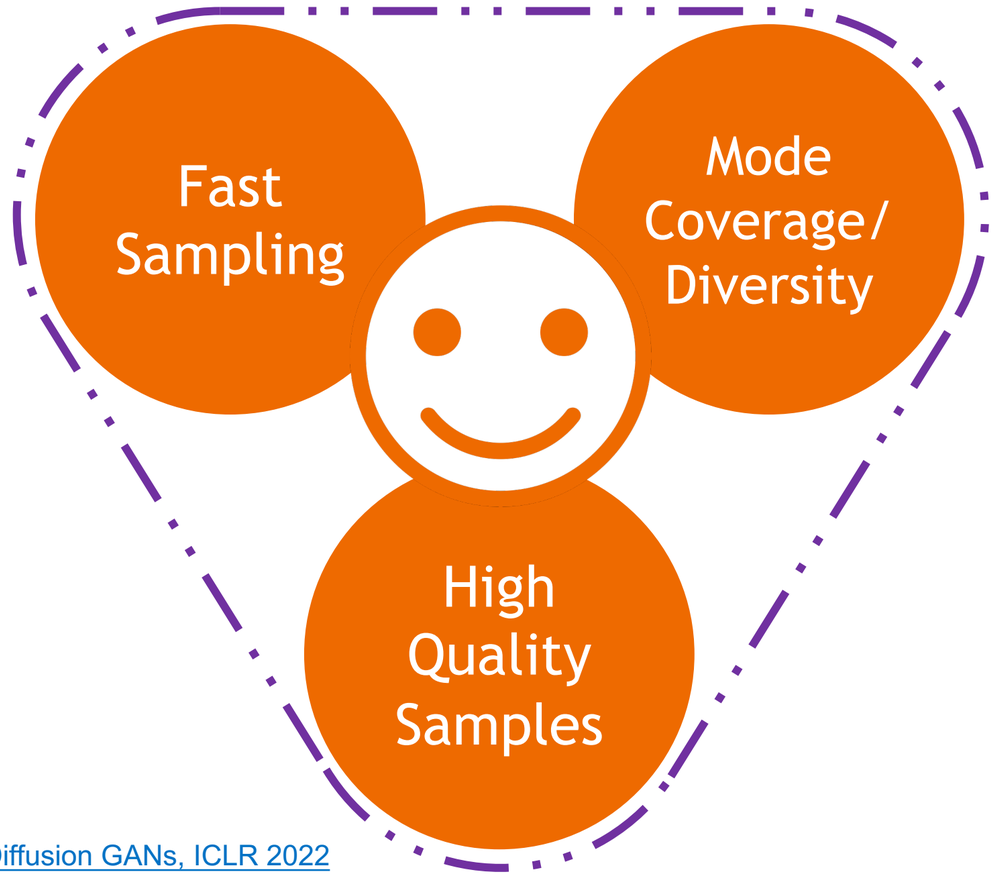


Tackling the Generative Learning Trilemma with Denoising Diffusion GANs, ICLR 2022

What makes a good generative model?

The generative learning trilemma

Tackle the trilemma by accelerating diffusion models



Tackling the Generative Learning Trilemma with Denoising Diffusion GANs, ICLR 2022

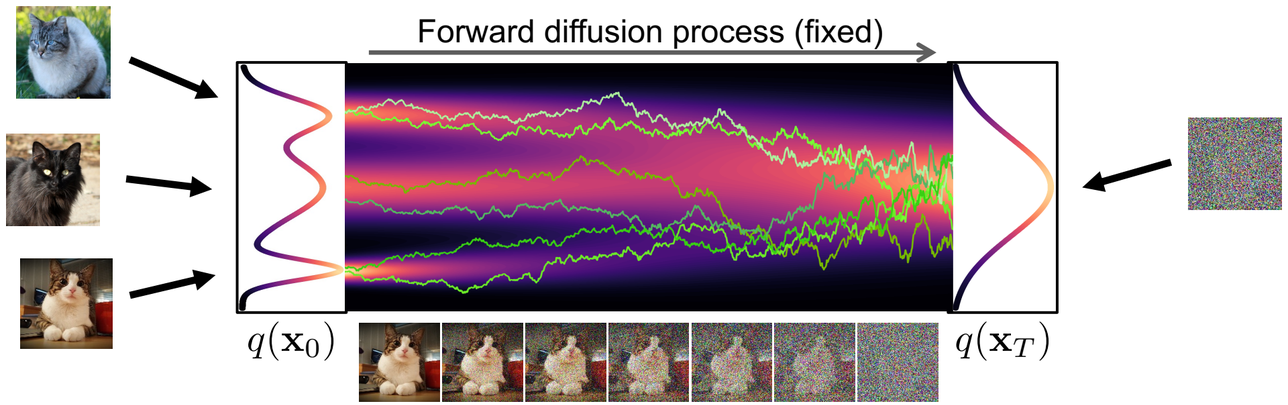
Acceleration Techniques

Advanced ODE/SDE Solvers

Distillation Techniques

Low-dim. Diffusion Processes

Advanced Diffusion Processes

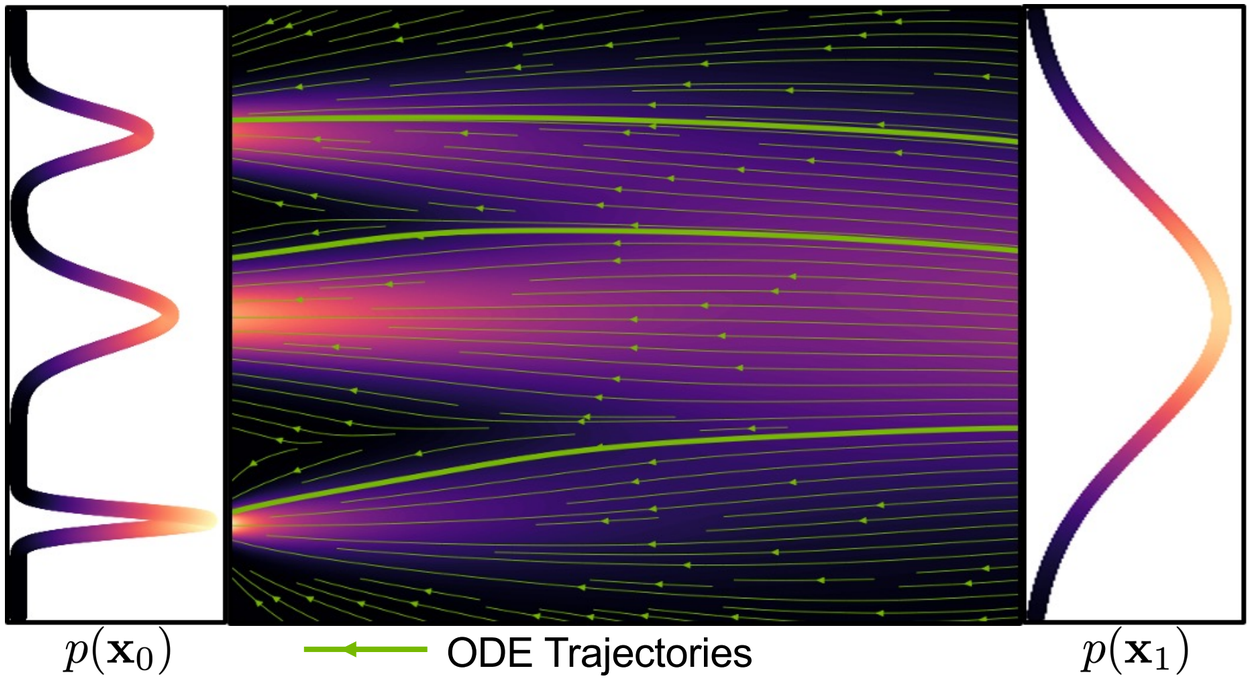


Acceleration Techniques

Generative ODEs

Solve ODEs with as little function evaluations as possible

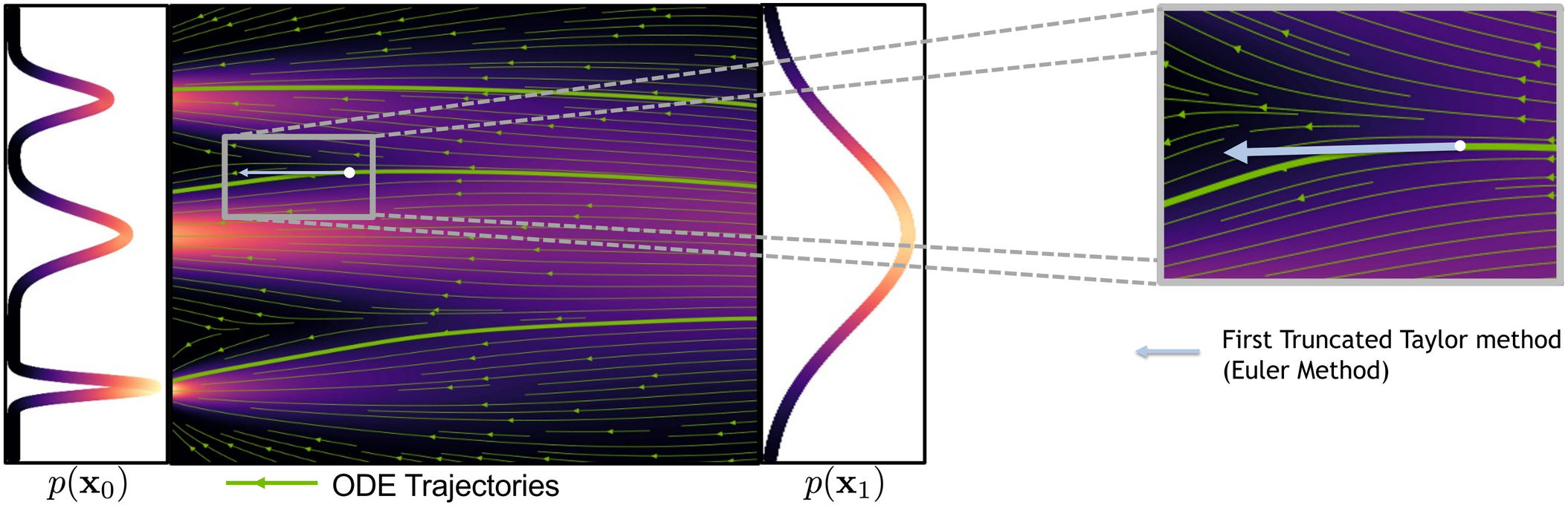
$$dx = \epsilon_\theta(x, t)dt$$



Generative ODEs

Solve ODEs with as little function evaluations as possible

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

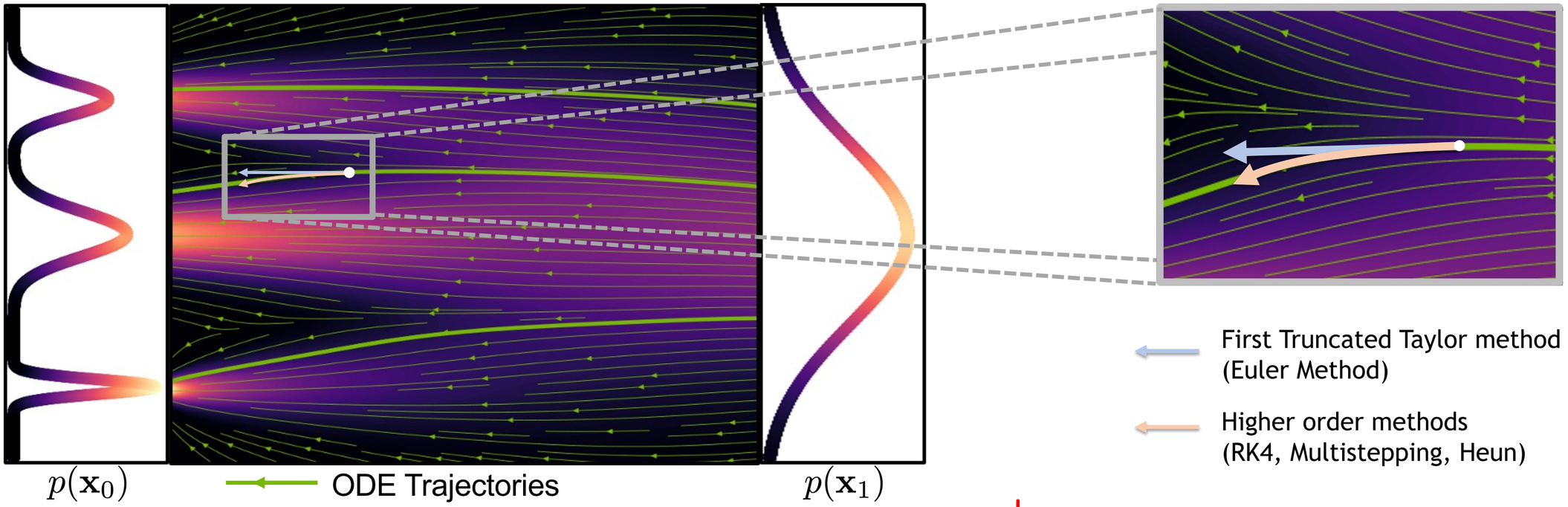


Song et al., "Denoising Diffusion Implicit Models (DDIM)", ICLR 2021

Generative ODEs

Solve ODEs with as little function evaluations as possible

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



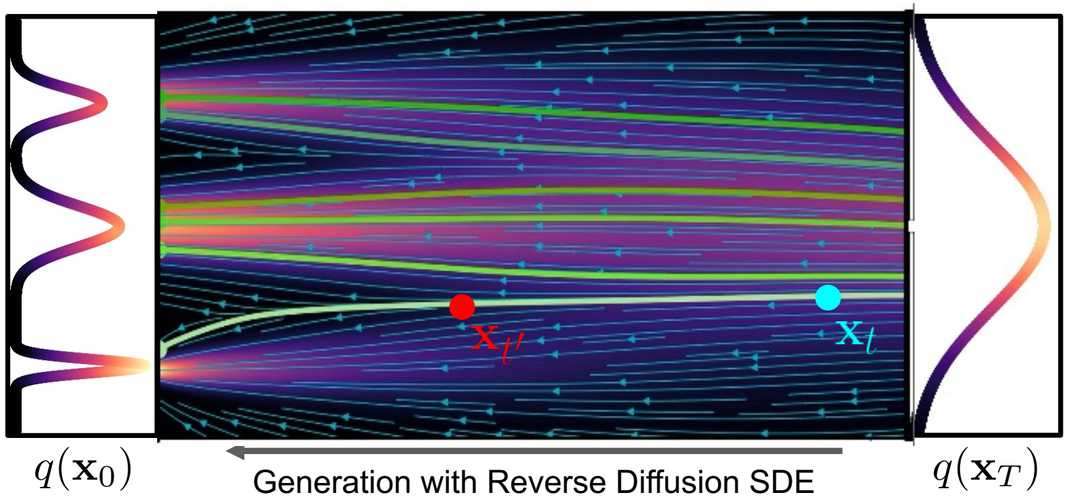
Approximate the higher order terms numerically

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:
  - Jolicoeur-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 multistepping:
  - 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

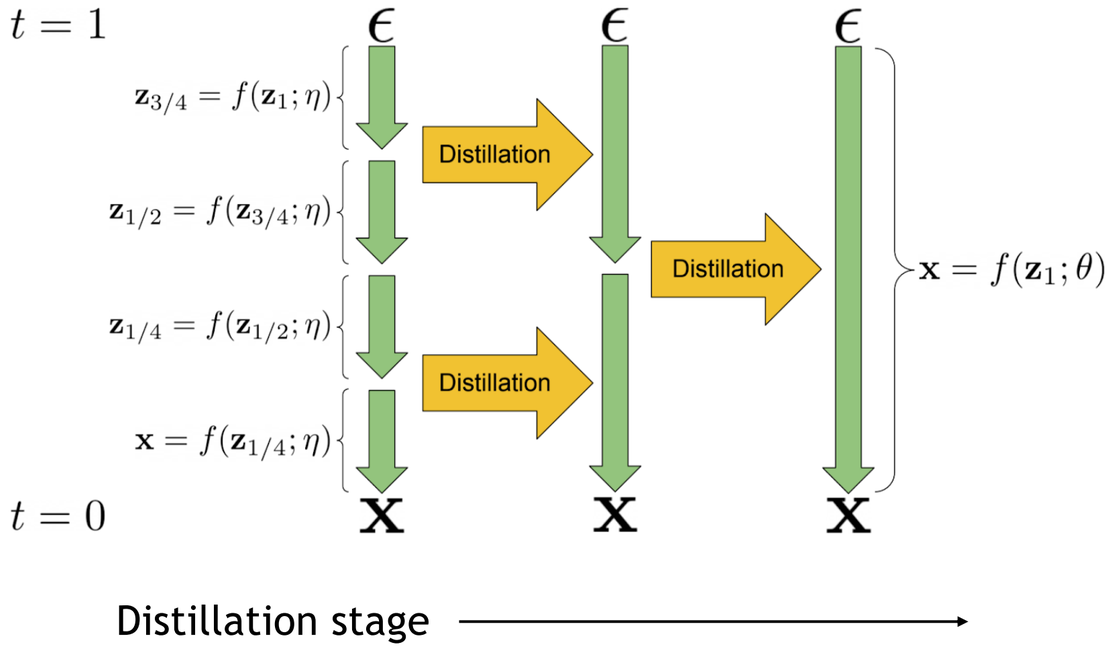
ODE Distillation



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.



Salimans & Ho, “Progressive distillation for fast sampling of diffusion models”, ICLR 2022.

Progressive Distillation in Latent Space

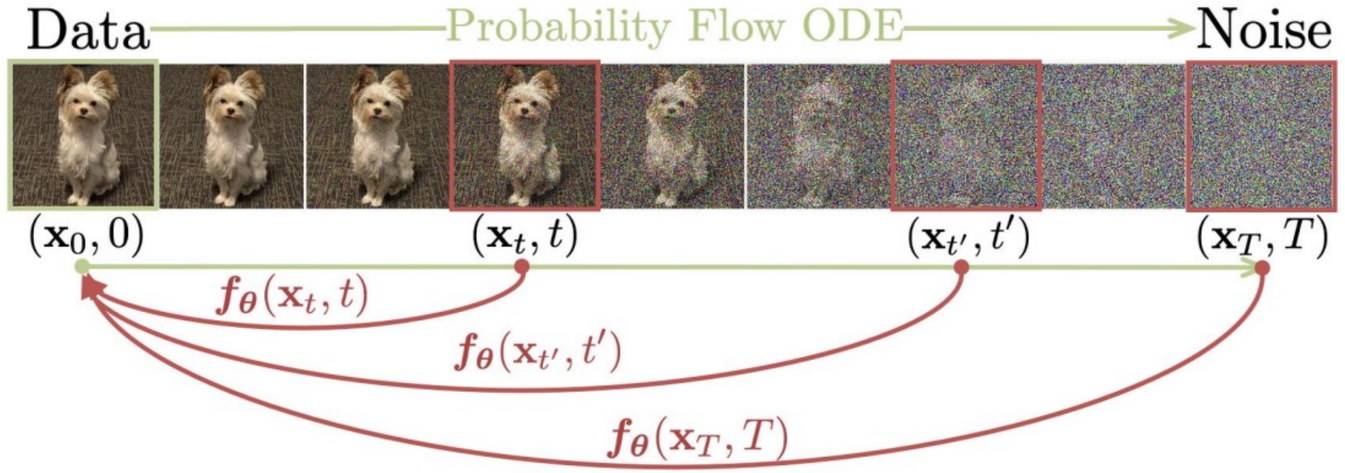
(a) 2 denoising steps

(b) 4 denoising steps

(c) 8 denoising steps

Meng et al., "On Distillation of Guided Diffusion Models", CVPR 2023 (Award Candidate)

Consistency Distillation



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

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

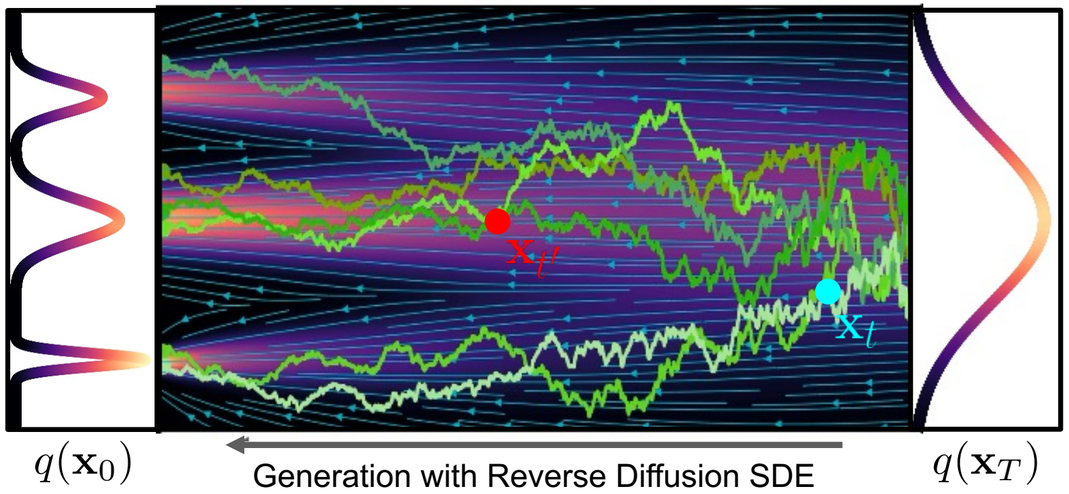
Basic idea:

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

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

Song et al., Consistency Models, ICML 2023

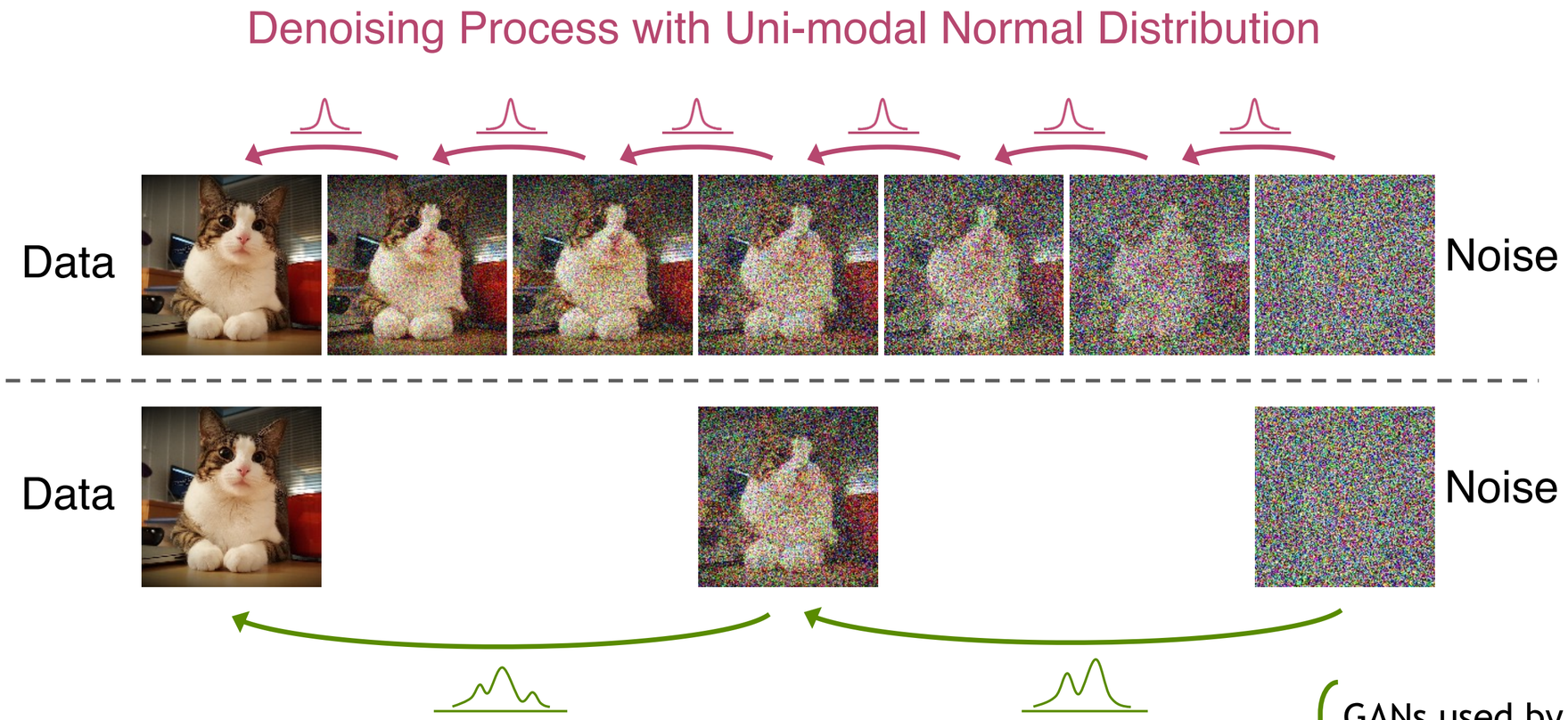
SDE Distillation



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



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.

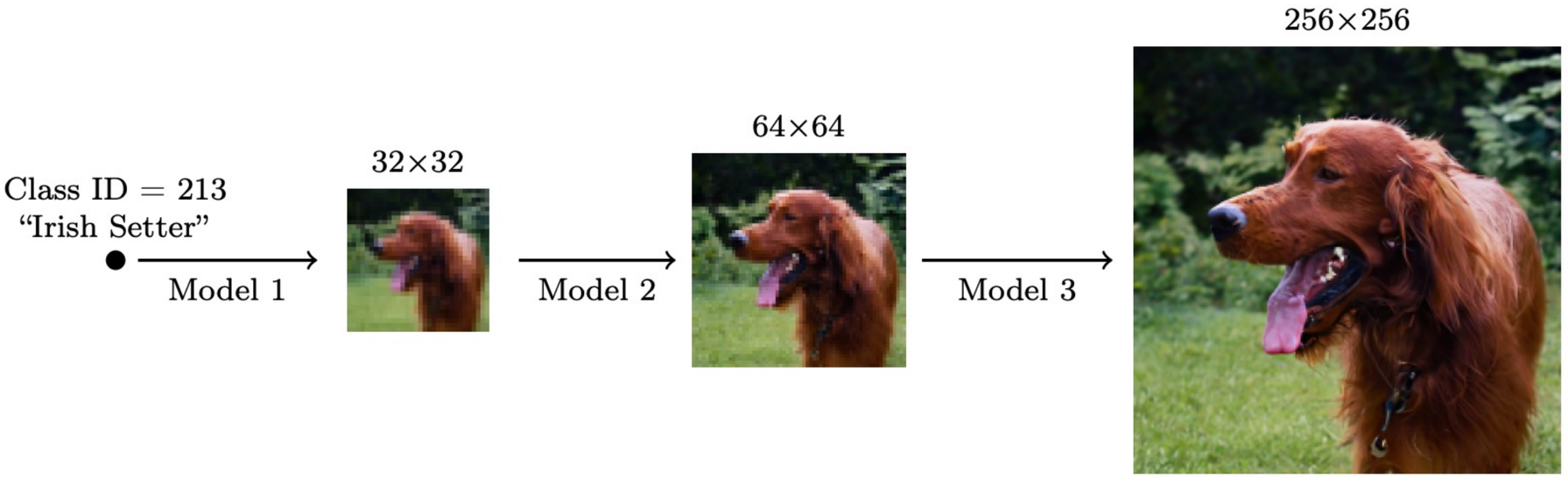
Training-based Sampling Techniques

- 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

Cascaded Generation

Pipeline

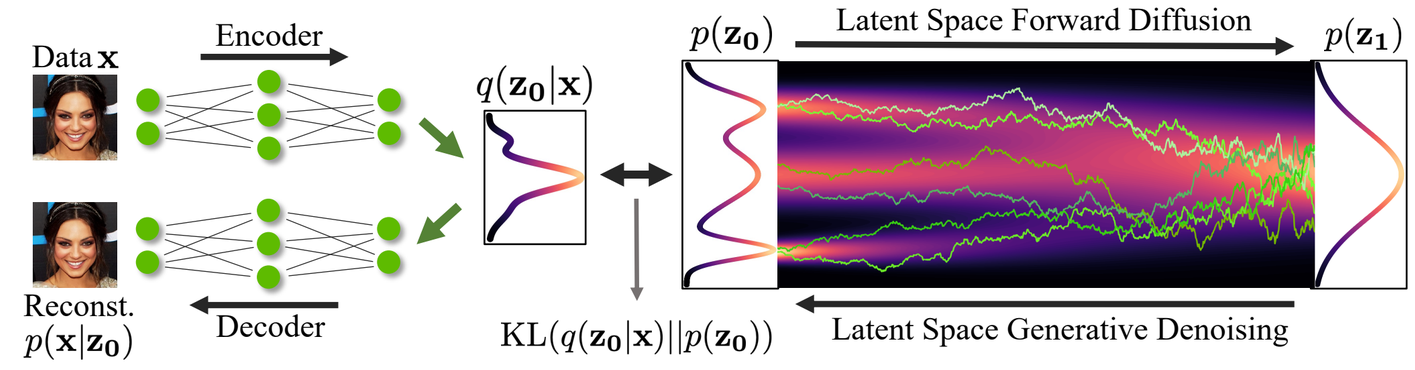


Cascaded Diffusion Models outperform Big-GAN in FID and IS and VQ-VAE2 in Classification Accuracy Score.

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

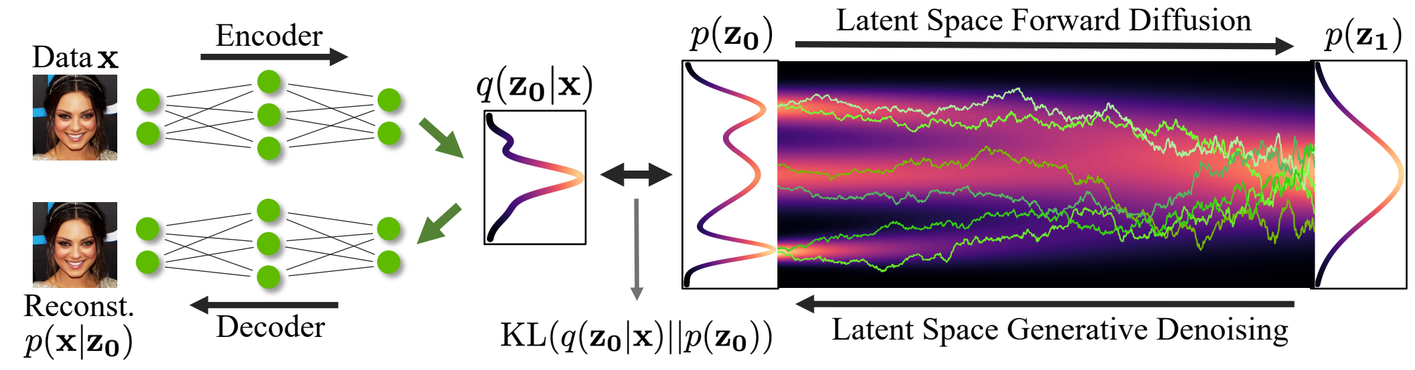
Encoder maps the input data to an embedding space

Denoising diffusion models are applied in the latent space

Vahdat et al., “Score-based generative modeling in latent space”, NeurIPS 2021.

Latent Diffusion Models

Variational autoencoder + score-based prior



Advantages:

(1) The distribution of latent embeddings close to Normal distribution → ***Simpler denoising, Faster synthesis!***

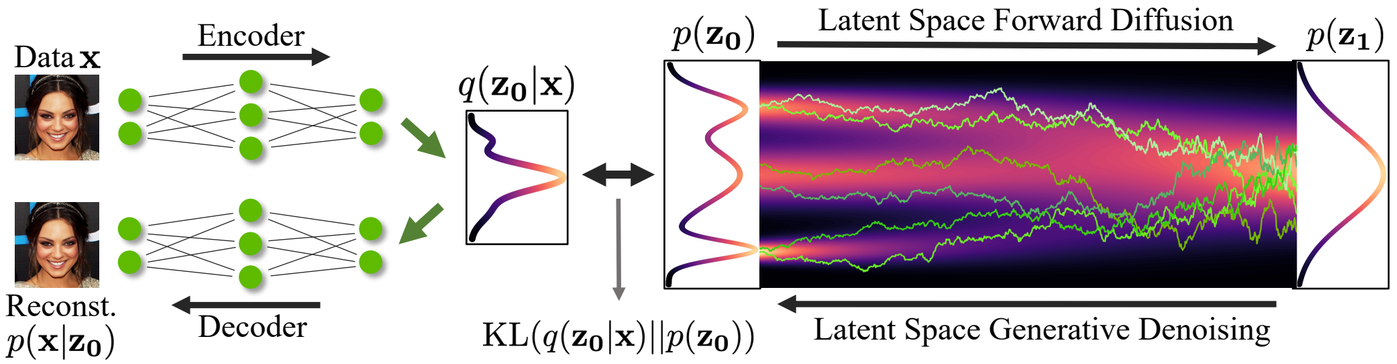
(2) Latent space → ***More expressivity and flexibility in design!***

(3) Tailored Autoencoders → ***More expressivity, Application to any data type (graphs, text, 3D data, etc.) !***

Vahdat et al., "Score-based generative modeling in latent space", NeurIPS 2021.

Latent Diffusion Models

End-to-End Training objective



$$\mathcal{L}(\mathbf{x}, \boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\psi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[-\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)\right] + \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}_0))$$

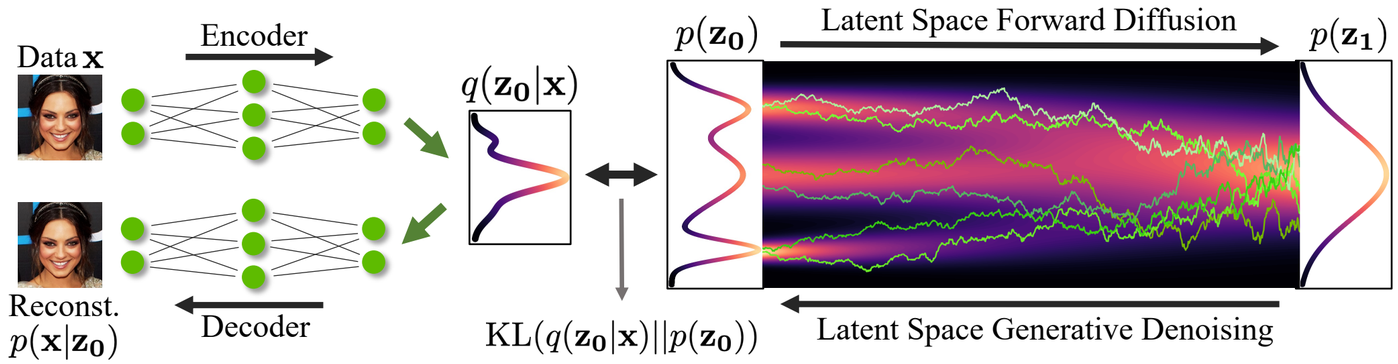
$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[-\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{reconstruction term}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})\right]}_{\text{negative encoder entropy}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{z}_0)\right]}_{\text{cross entropy}}$$

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

Vahdat et al., “Score-based generative modeling in latent space”, NeurIPS 2021.

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

Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models”, CVPR 2022.

Additional Reading

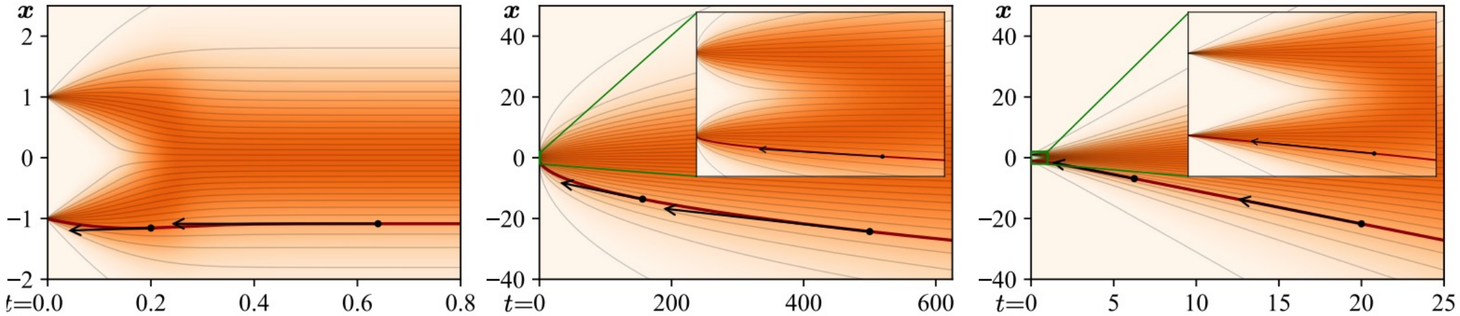
More on low-dimensional diffusion models:

- 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

ODE interpretation

Deterministic generative process



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

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

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

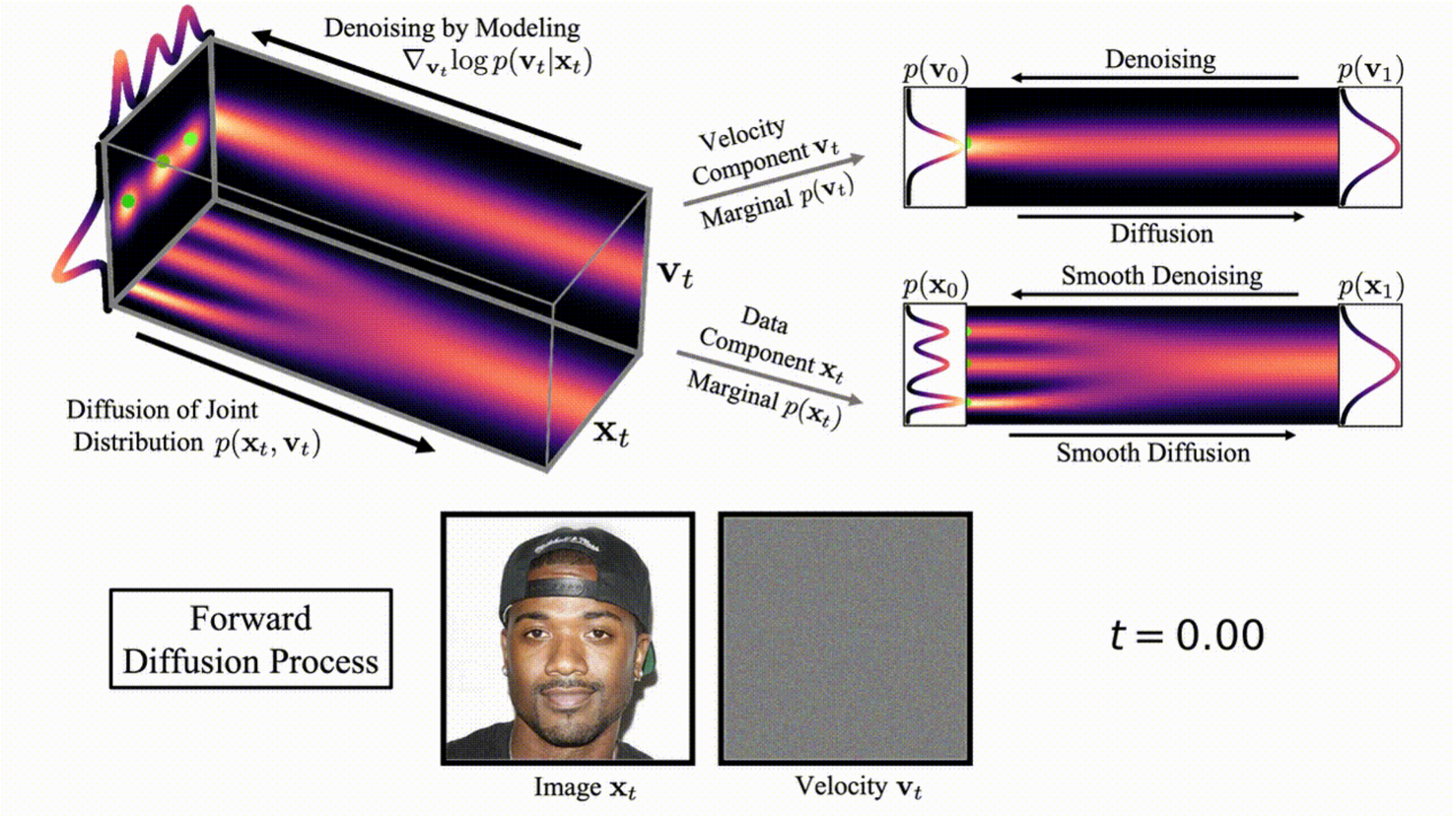
- Karras et al. argue that the ODE of DDIM is favored, as the tangent of the solution trajectory always points towards the denoiser output.

- This 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.

“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!

Dockhorn et al., “Score-Based Generative Modeling with Critically-Damped Langevin Diffusion”, ICLR 2022.

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.”

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.

Conditioning and Guidance Techniques

Explicit Conditions

Classifier Guidance

Classifier-free Guidance

Conditioning and Guidance Techniques

Explicit Conditions

Classifier Guidance

Classifier-free Guidance

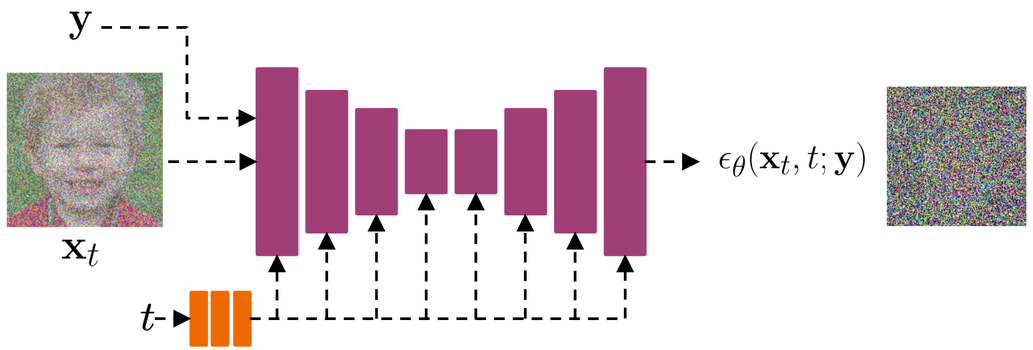
Explicit Conditional Training

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

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

Using Bayes' rule, we have:

Classifier gradient

Score model

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

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:

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

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:

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

Ho & Salimans, "Classifier-Free Diffusion Guidance", 2021.

Classifier-free guidance

Trade-off for sample quality and sample diversity

Non-guidance

$\omega = 1$

$\omega = 3$

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

Ho & Salimans, “Classifier-Free Diffusion Guidance”, 2021.

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
bit.ly/43ZgpZW

Thanks!

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

@ArashVahdat