# Deep Learning

# Diffusion

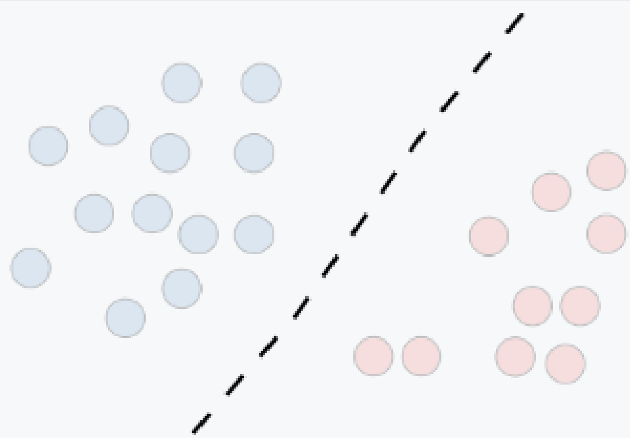
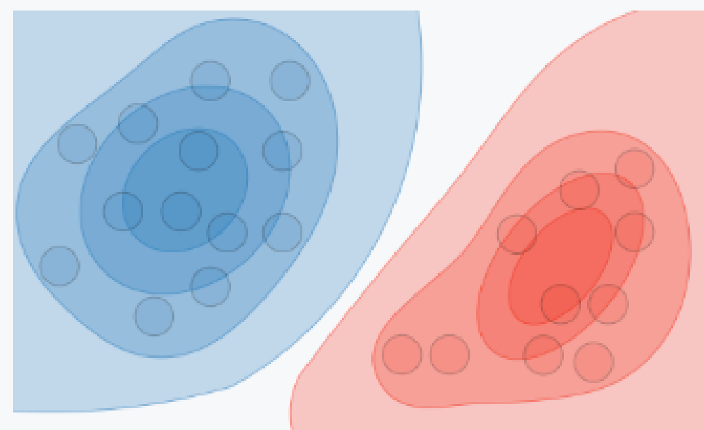
Hao Chen

## Fall 2024

## Attendance: @

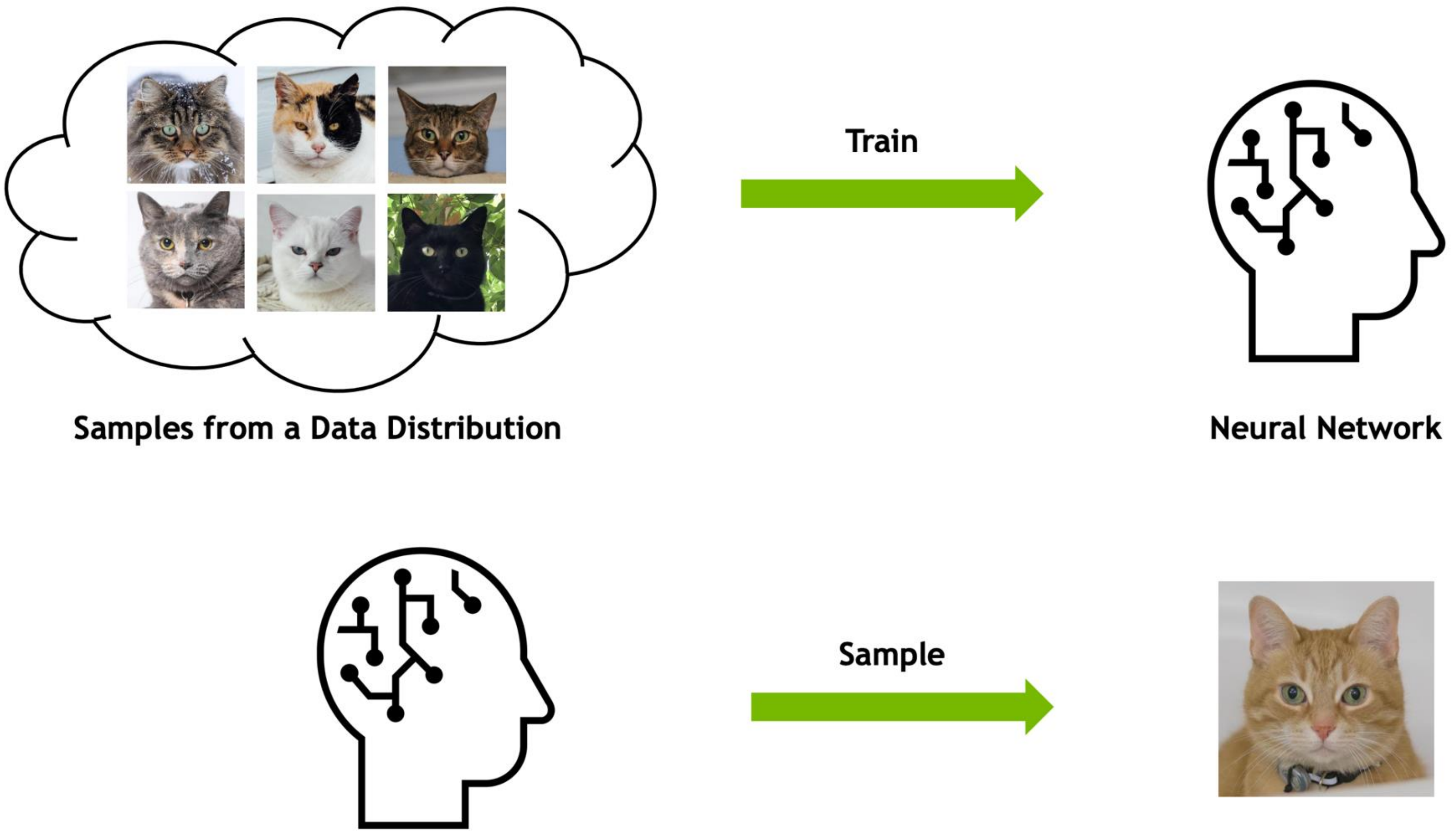
# Generative vs. Discriminative

- Generative models learn the data distribution

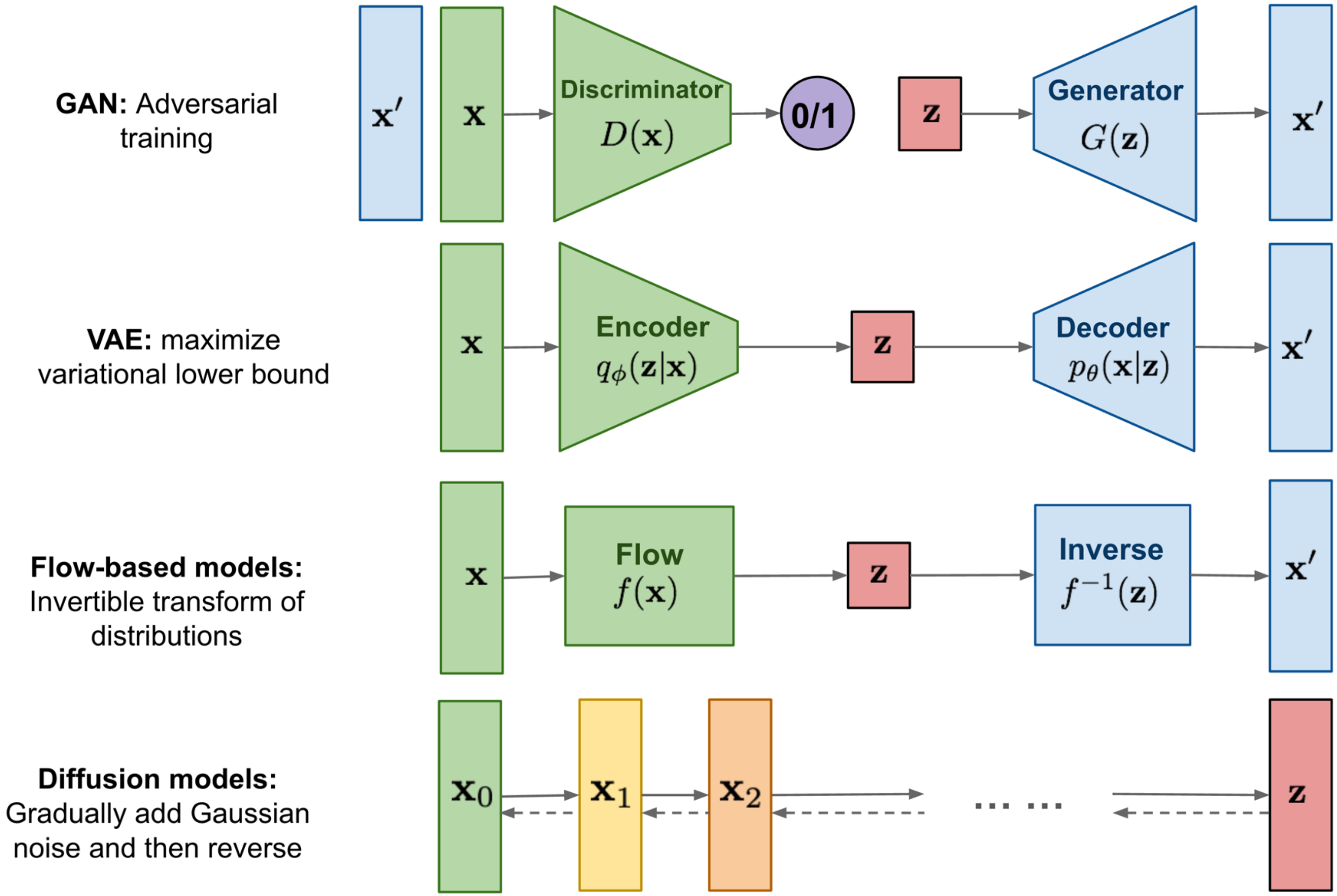
| | Discriminative model | Generative model |
|---|---|---|
| **Goal** | Directly estimate $P(y\|x)$ | Estimate $P(x\|y)$ to then deduce $P(y\|x)$ |
| **What's learned** | Decision boundary | Probability distributions of the data |
| **Illustration** |  |  |

# Generative Models

- Learning to generate data

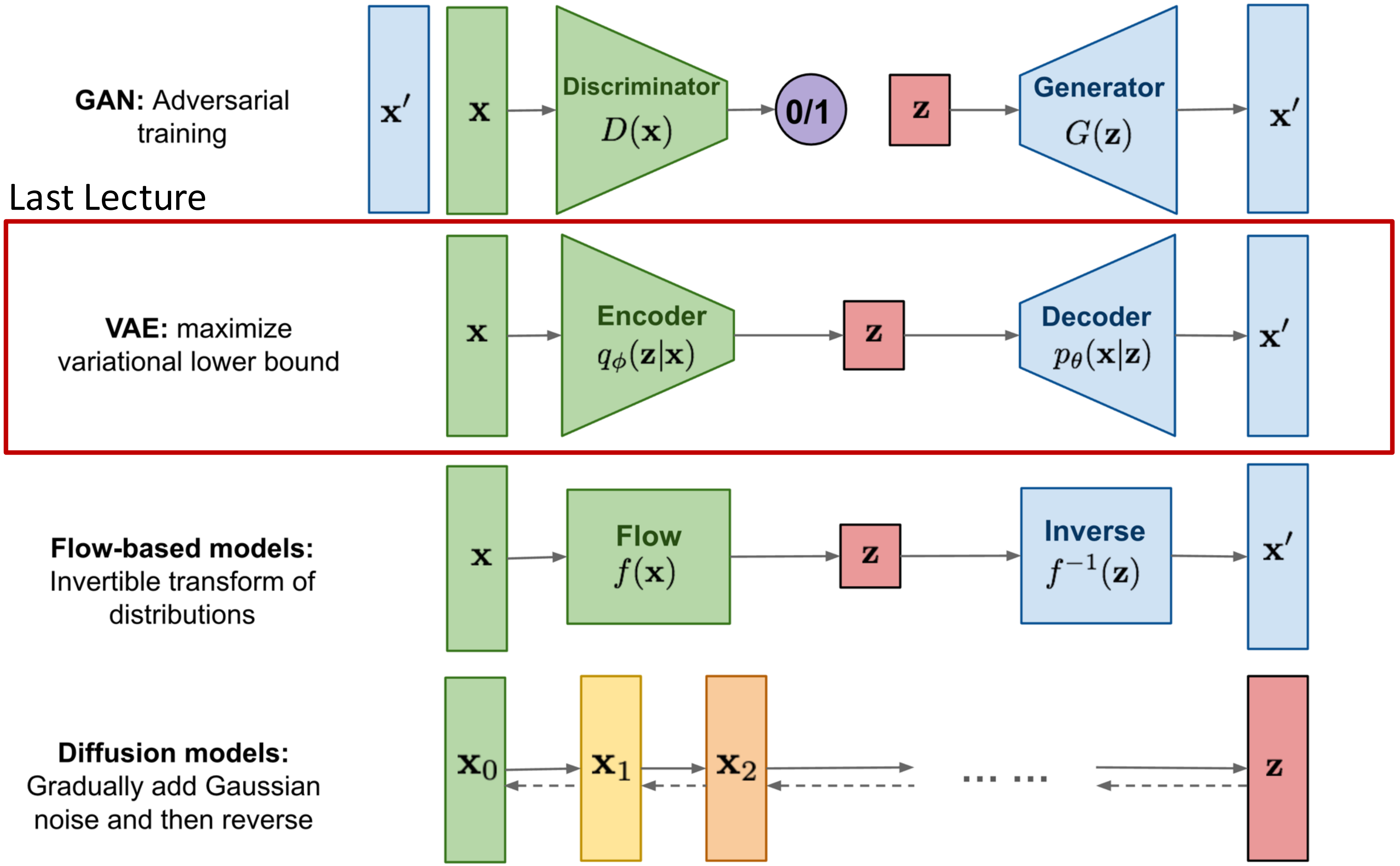


# Generative Models



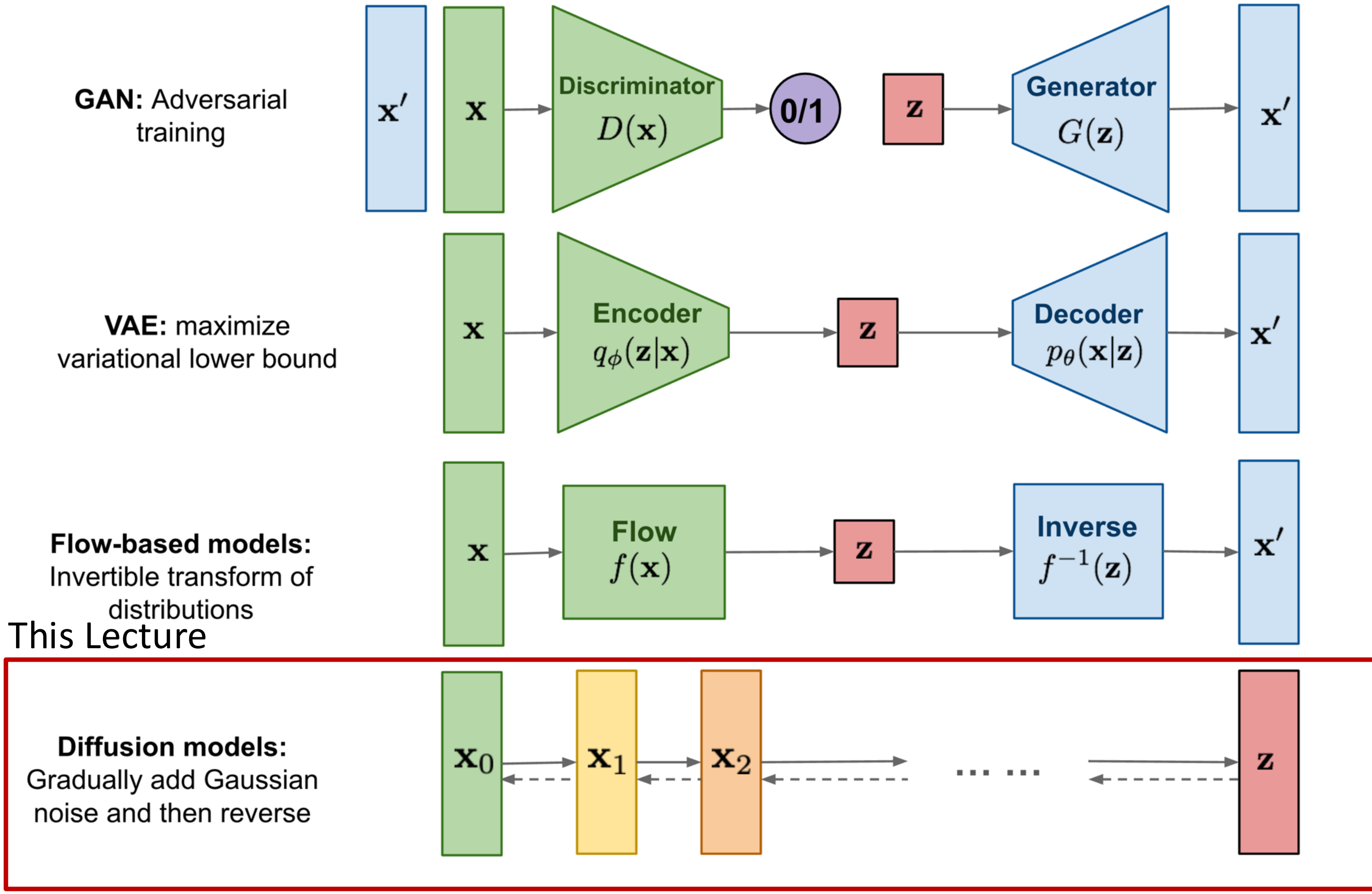

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

# Generative Models




https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

# Generative Models




https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

# A Fast Evolving Field

VAEs, 2013

GANs, 2014

PixelCNN, 2016

BigGAN, 2019

Imagen, 2022


SORA 2024

# Content

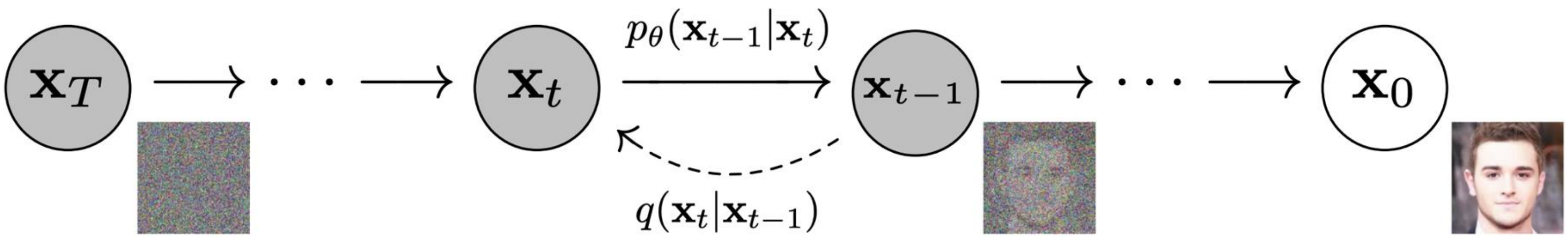
- Denoising Diffusion Model Basics
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Denoising Diffusion Implicit Model (DDIM)
- Conditional Diffusion Models
- Applications of Diffusion Models

# Content

- Diffusion Model Basics
  - Diffusion Models as Stacking VAEs
  - Diffusion Models: Forward, Reverse, Training, Sampling
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Denoising Diffusion Implicit Model (DDIM)
- Conditional Diffusion Models
- Applications of Diffusion Models

# Denoising Diffusion Models

- what we often see about diffusion models




$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{\bar{\alpha}_t} \mathbf{x}_0,\left(1-\bar{\alpha}_t\right) \mathbf{I}\right)\right)$$

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

Forward diffusion process

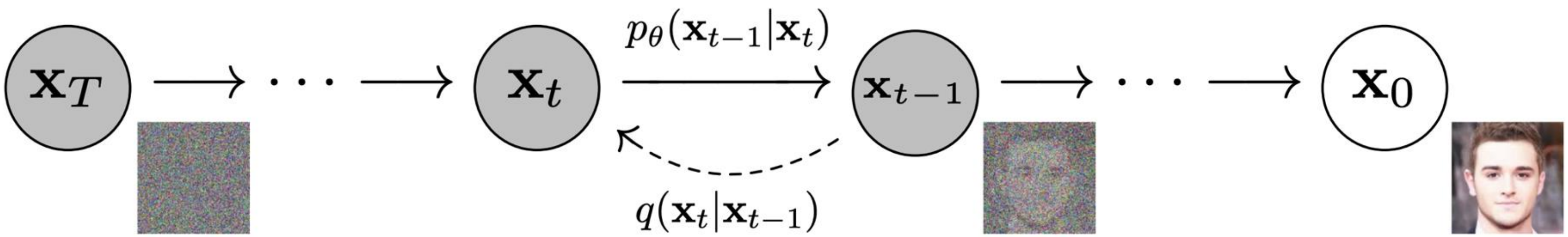
$$p\left(\mathbf{x}_T\right)=\mathcal{N}\left(\mathbf{x}_T ; \mathbf{0}, \mathbf{I}\right)$$

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

Reverse denoising process

# Denoising Diffusion Models

- what we often see about diffusion models




$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{\bar{\alpha}_t} \mathbf{x}_0,\left(1-\bar{\alpha}_t\right) \mathbf{I}\right)\Big)$$

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

Forward diffusion process

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

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

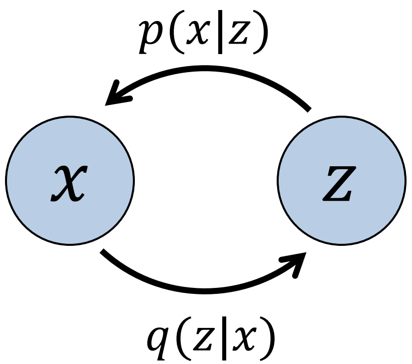
Reverse denoising process

- this lecture: denoising diffusion is a stack of VAEs

# Recap: Variational Autoencoders

- VAEs: a likelihood-based generative model

- **Encoder**: an inference model that approximates the posterior $q(z|x)$

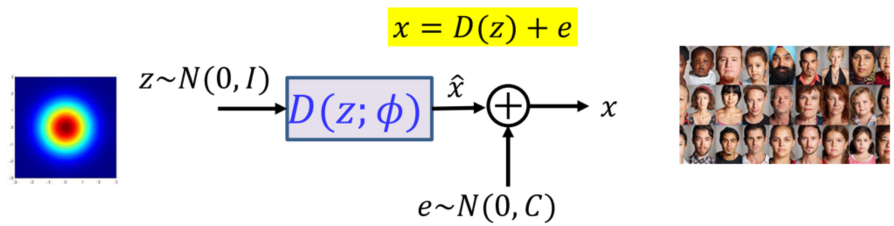


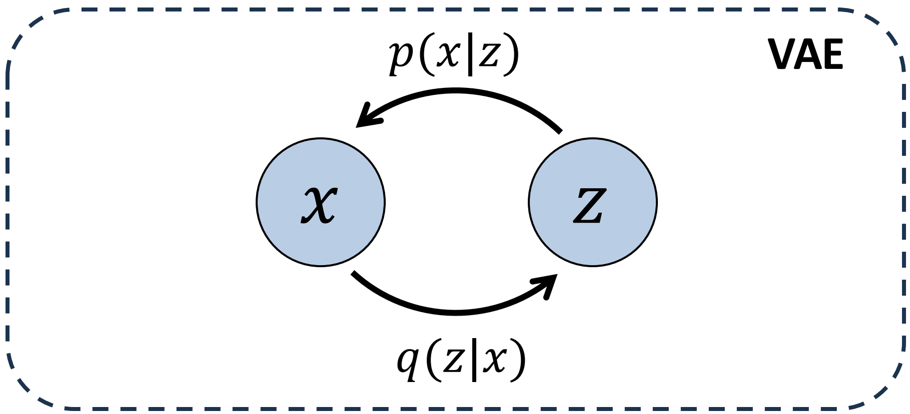

- **Decoder**: a generative model that transforms a Gaussian variable z to real data

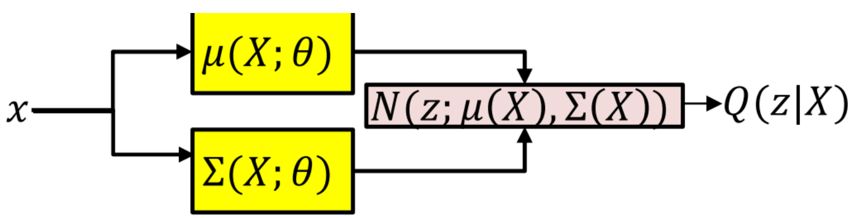
- **Training**: maximize the ELBO

# Recap: Variational Autoencoders

**Decoder**: transforms a Gaussian variable to real data



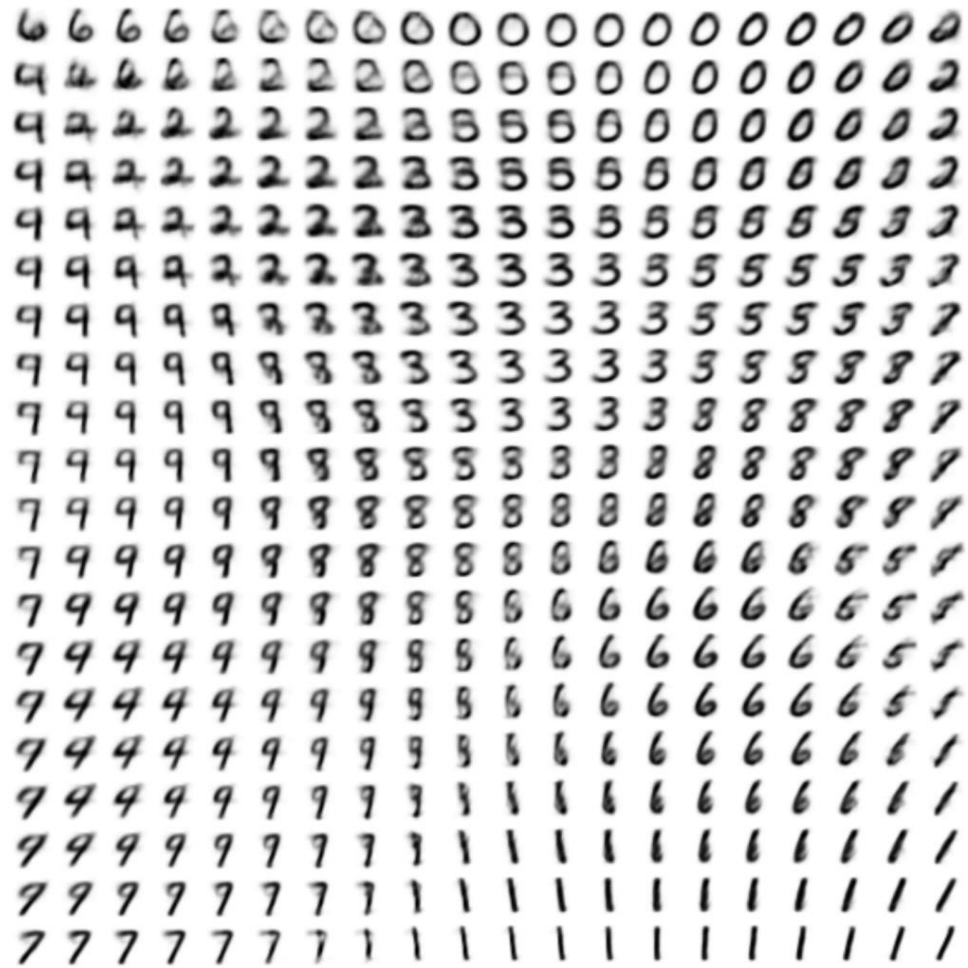




**Encoder**: an inference model approximates the posterior, i.e. Gaussian

# VAEs are good, but…

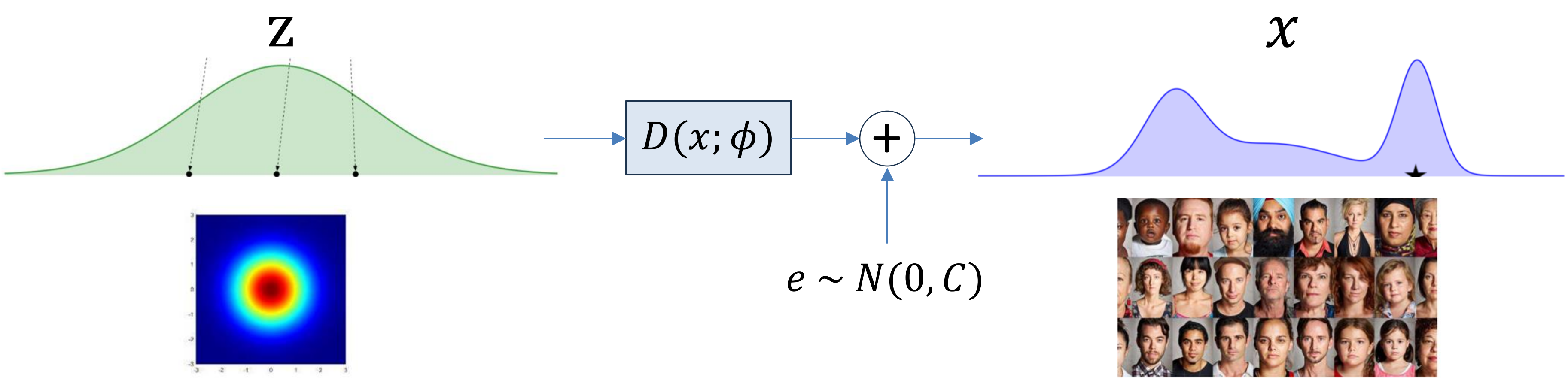
- Blurry results



Kingma et al. Auto-Encoding Variational Bayes

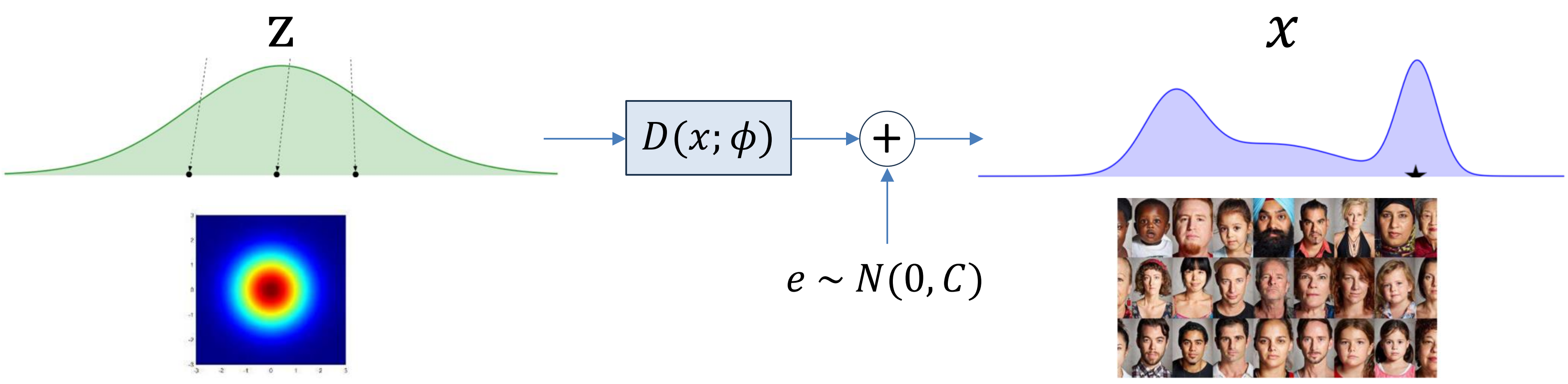
# Limitations of VAEs

- Decoder must transform a standard Gaussian all the way to the target distribution in **one-step**
  - Often too large a gap
  - Blurry results are generated



# Limitations of VAEs

- Decoder must transform a standard Gaussian all the way to the target distribution in **one-step**
  - Often too large a gap
  - Blurry results are generated

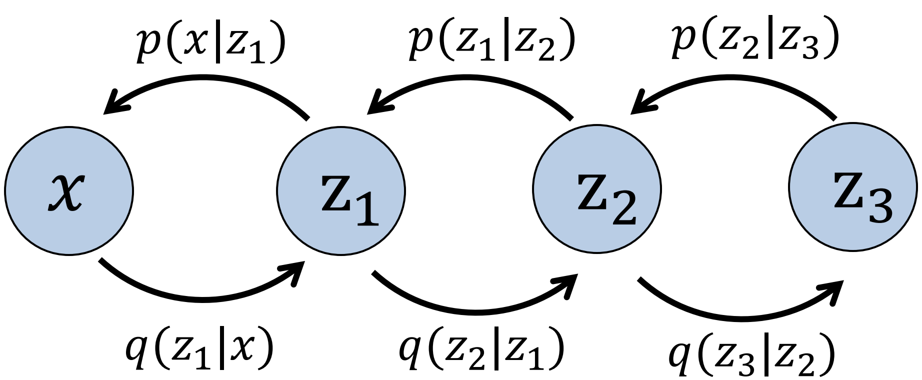



- Solution: have some intermediate latent variables to reduce the gap of each step

# Hierarchical VAEs

- Hierarchical VAEs – Stacking VAEs on top of each other
  - Multiple (T) intermediate latent
  - Joint distribution $p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_T) p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t)$
  - Posterior $q_\phi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x}) = q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}) \prod_{t=2}^{T} q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})$

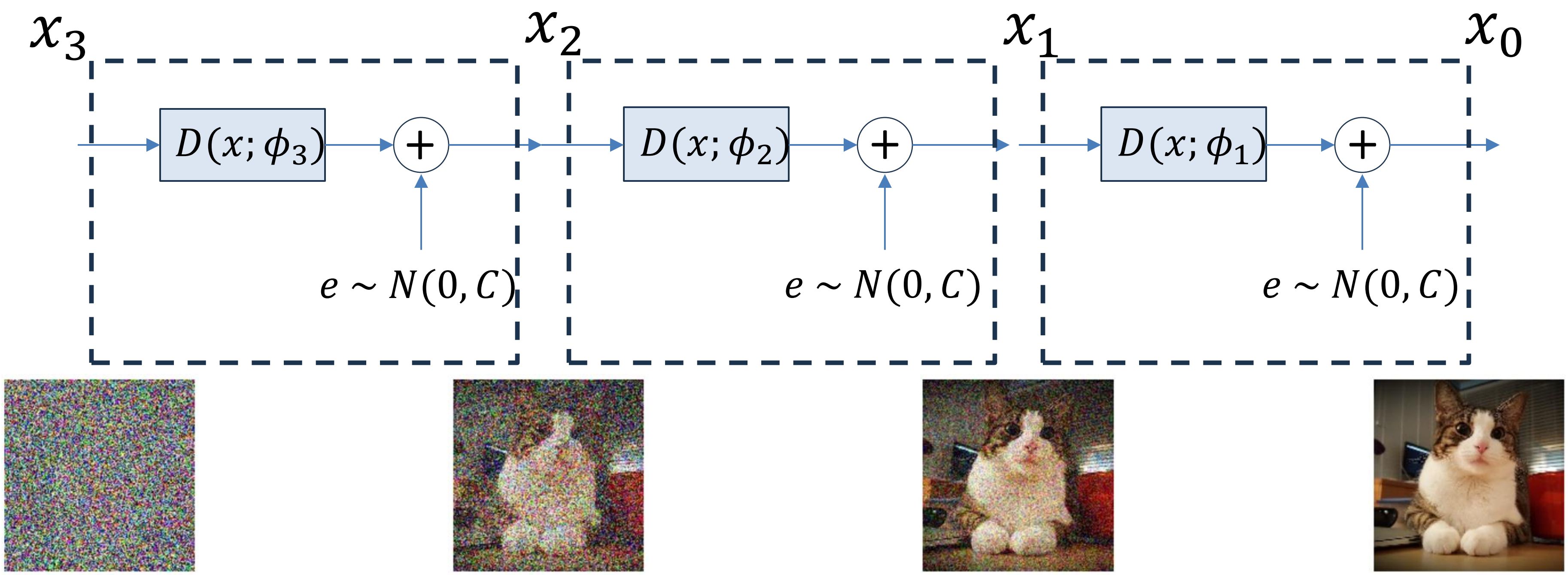
- Better likelihood achieved!




Sønderby et al. Ladder Variational Networks. 2016

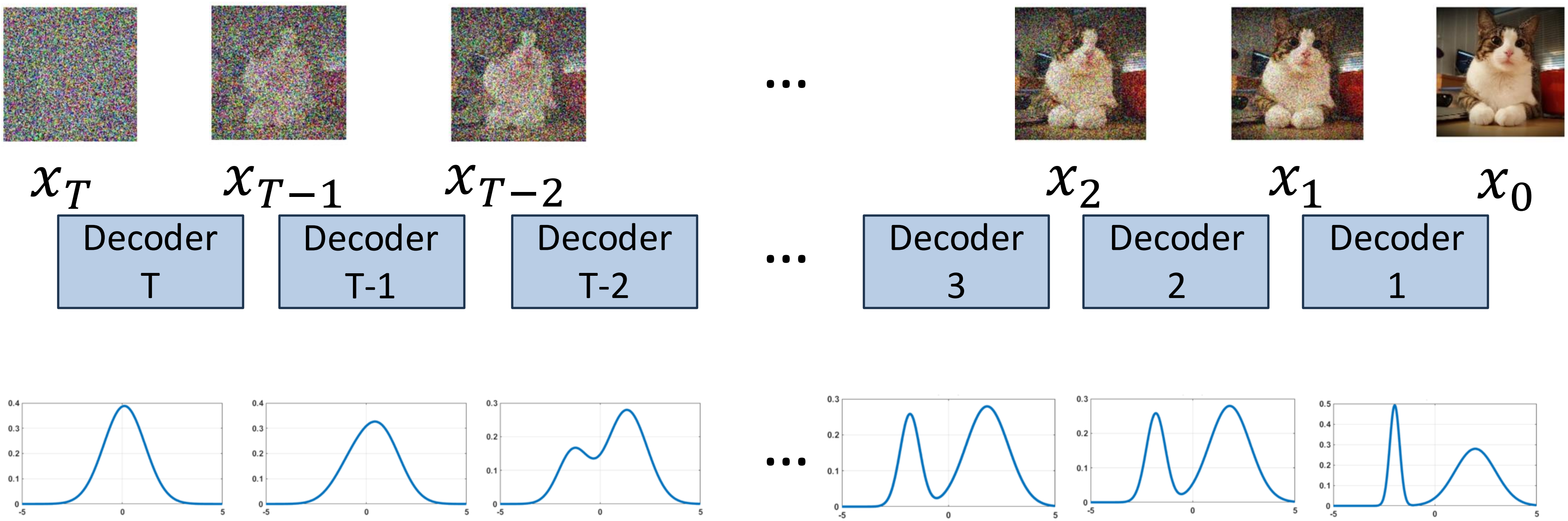
# Stacking VAEs

- Each step, the decoder removes part of the noise
- Provides a seed model closer to final distribution



# Stacking VAEs

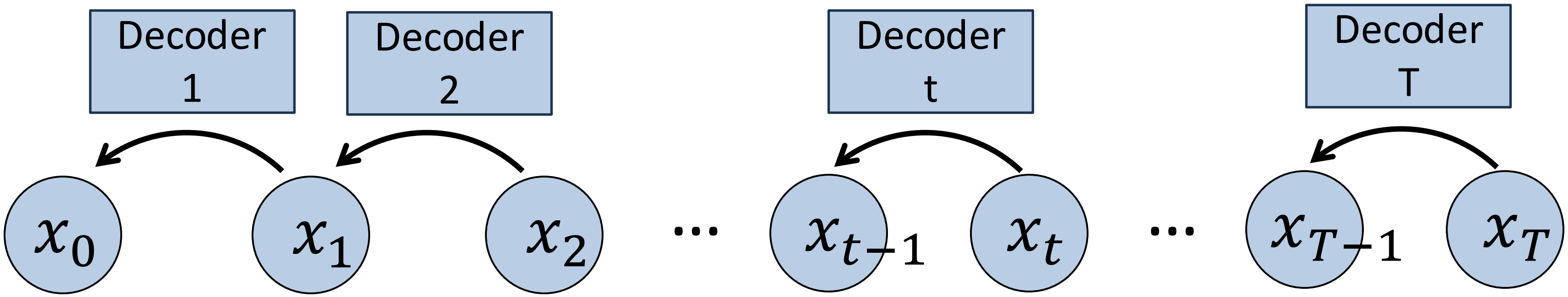
- We can have many many steps (in total T)…
- Each step incrementally recovers the final distribution




- Looks familiar?

# Diffusion Models are Stacking VAEs

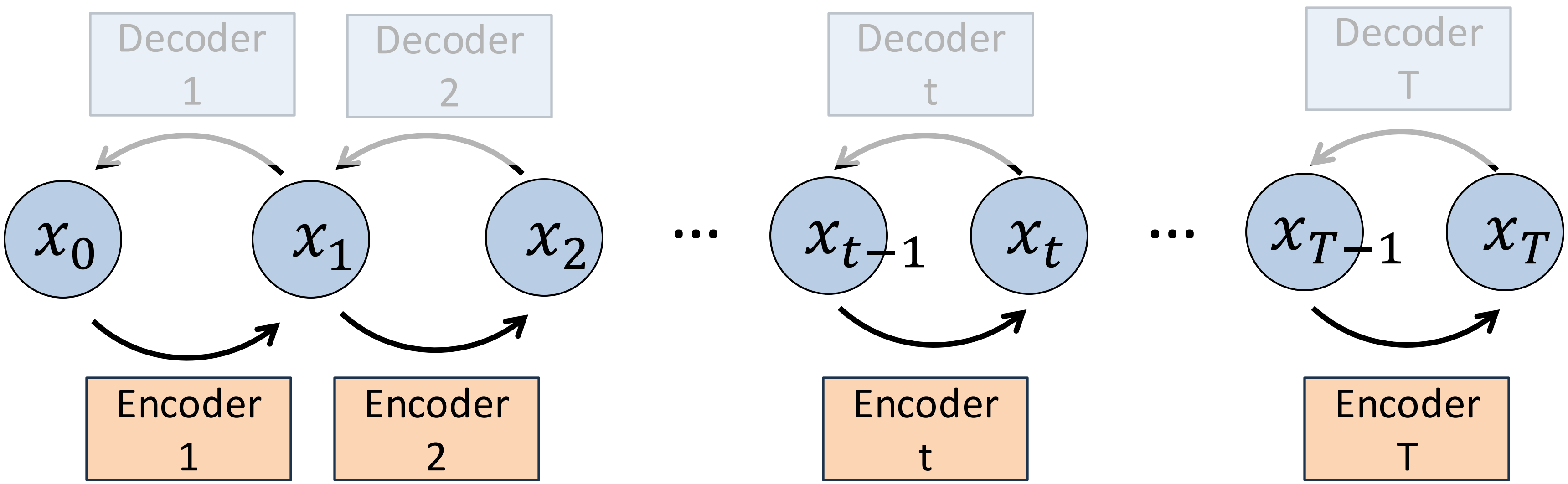
- Diffusion models are special cases of Stacking VAEs




- The reverse denoising process is the stack of decoders
- What about encoders?

# Diffusion Models are Stacking VAEs

- Diffusion models are special case of Stacking VAEs



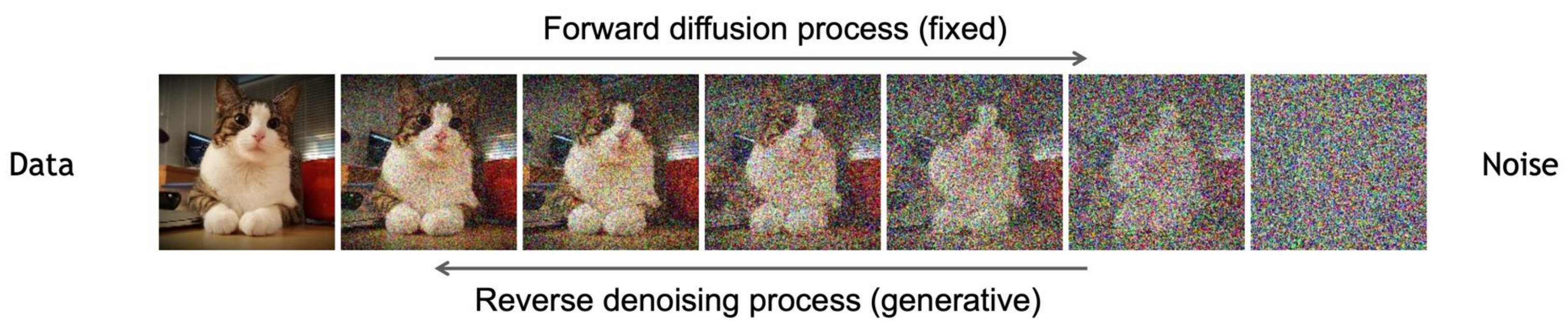

- In VAEs, encoders are learned with KL-divergence between the posterior and the prior
- Suffers from the 'posterior-collapse' issue
- Diffusion models use **fixed inference encoders**

Chen et al. Variational Lossy Autoencoder. 2016

# Poll

# Denoising Diffusion Models

- Diffusion models have two processes
- **Forward diffusion process** gradually adds noise to input
- **Reverse denoising process** learns to generate data by denoising

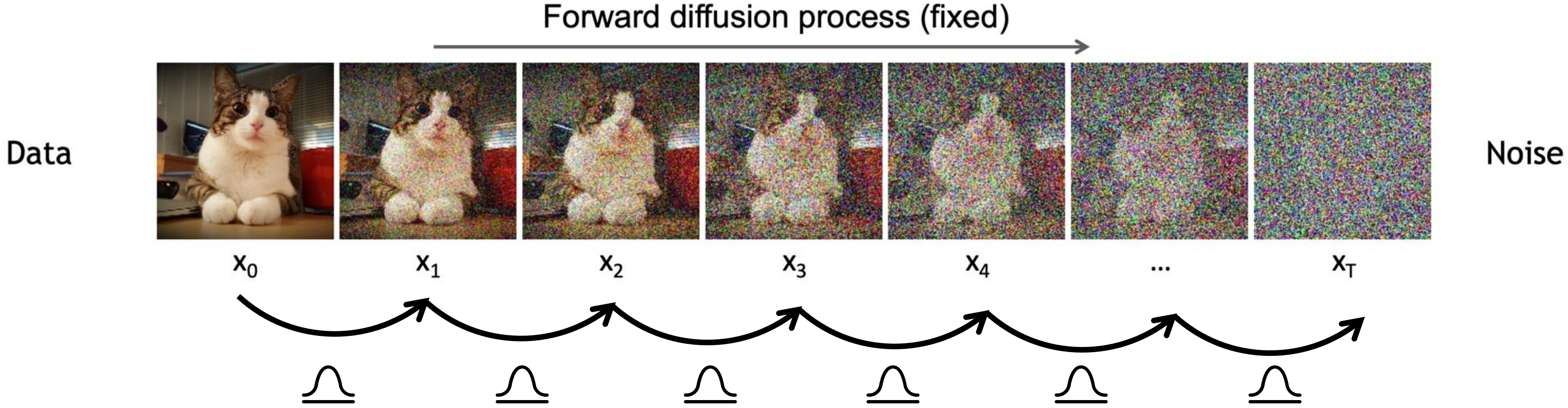


# Forward Diffusion Process

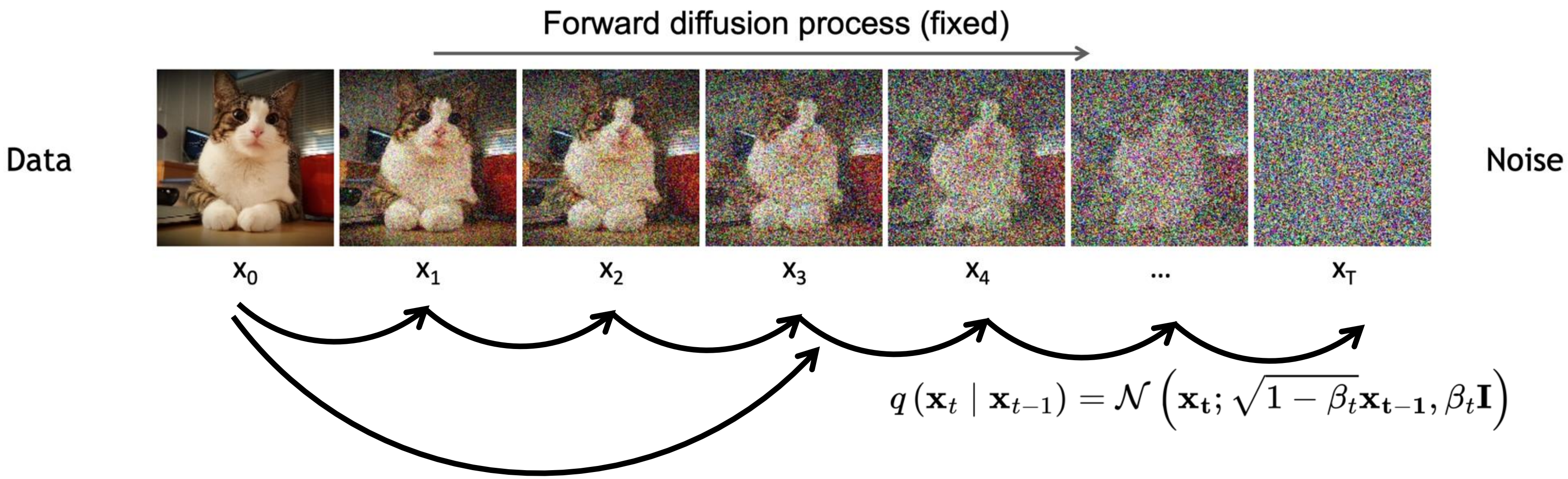
- Forward diffusion process is stacking **fixed** VAE encoders
  - gradually adding Gaussian noise according to schedule $\beta_t$

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

$$q\left(\mathbf{x}_{1: T} \mid \mathbf{x}_0\right)=\prod_{t=1}^T q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)$$



# Forward Diffusion Process




- The forward process allows sampling of $x_t$ at arbitrary timestep $t$ in closed form:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right) \qquad \bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

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

- The noise schedule ($\beta_t$ values) is designed such that

$$q(\mathbf{x}_T \mid \mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

# Reverse Denoising Process

- Generation process
  - Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$
  - Iteratively sample $\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$

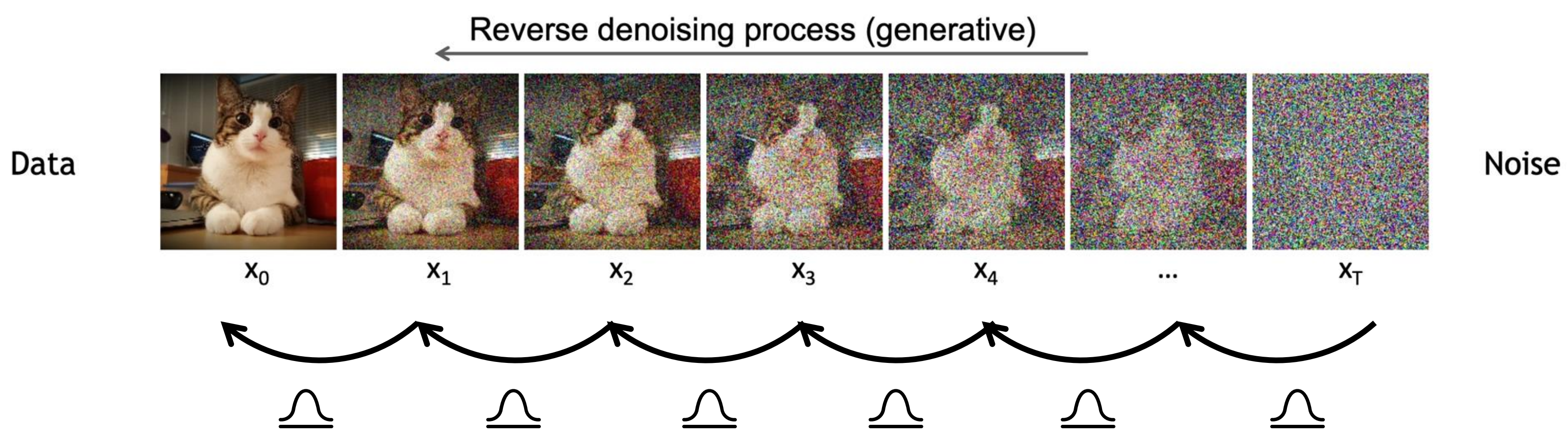
- $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ not directly tractable
- But can be estimated with a Gaussian distribution if $\beta_t$ is small at each step
  - The purpose of our stack of VAE decoders!

# Reverse Denoising Process

- Reverse diffusion process is stacking **learnable** VAE decoders
  - Predicting the mean and std of added Gaussian Noise

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

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

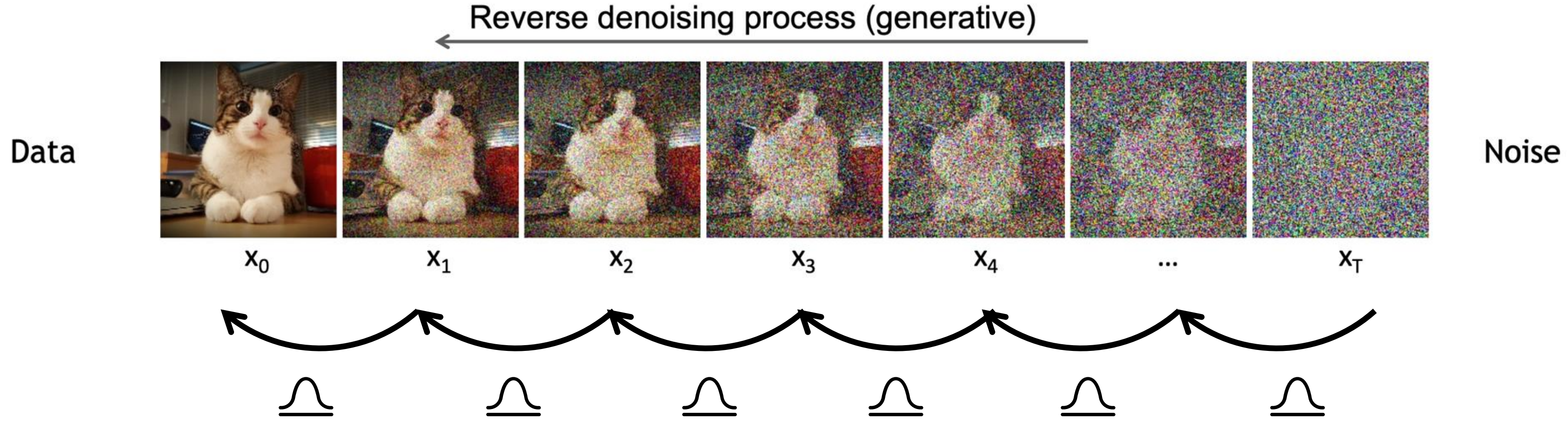


# Reverse Denoising Process

- Reverse diffusion process is stacking **learnable** VAE decoders
  - Predicting the mean and std of added Gaussian Noise

$$p\left(\mathbf{x}_T\right)=\mathcal{N}\left(\mathbf{x}_T ; \mathbf{0}, \mathbf{I}\right) \qquad p_\theta\left(\mathbf{x}_{0: T}\right)=p\left(\mathbf{x}_T\right) \prod_{t=1}^T p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$$

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



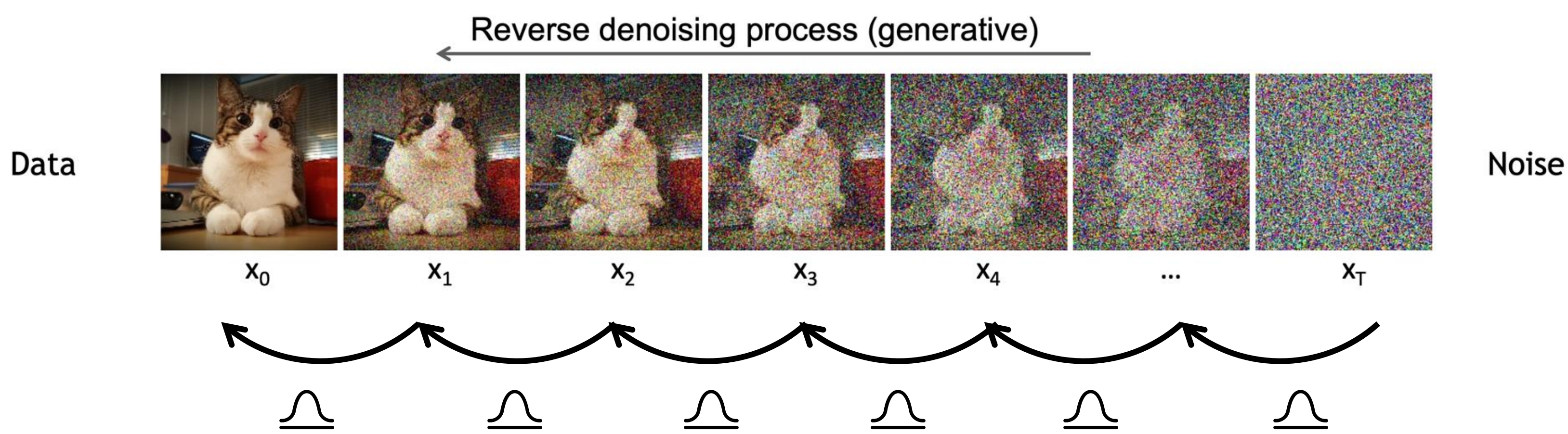
# Reverse Denoising Process

- Reverse diffusion process is stacking **learnable** VAE decoders
  - Predicting the mean and std of added Gaussian Noise

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

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}}_{\text{Trainable Network, Shared Across All Timesteps}})$$

Trainable Network, Shared Across All Timesteps



# Learning the Denoising Model

- Denoising models are trained with variational upper bound (negative ELBO), as VAEs

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

- which derives to:

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

- tractable posterior distribution (closed-form)

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

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

Ho et al. Denoising Diffusion Probabilistic Models. 2020.
Slide credit to: Ruiqi Gao CS231N

# Learning the Denoising Model

- Denoising models are trained with variational upper bound (negative ELBO), as VAEs

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

- which derives to:

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

constant　　　　Scaling

- tractable posterior distribution (closed-form)

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

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

Ho et al. Denoising Diffusion Probabilistic Models. 2020.
Slide credit to: Ruiqi Gao CS231N

# Learning the Denoising Model

- Denoising models are trained with variational upper bound (negative ELBO), as VAEs

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

- which derives to:

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

- tractable posterior distribution (closed-form)

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

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



Ho et al. Denoising Diffusion Probabilistic Models. 2020.
Slide credit to: Ruiqi Gao CS231N

# Parameterizing the Denoising Model

- KL divergence has a simple form between Gaussians

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

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

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

- Trainable network predicts the noise mean

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

- Final Objective

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

# Simplified Training Objective

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

- $\lambda_t$ ensures the weighting for correct maximum likelihood estimation

- In DDPM, this is further simplified to:

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

# Summary: Training and Sampling

**Algorithm 1** Training

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

**Algorithm 2** Sampling

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

# Summary: Noise Schedule

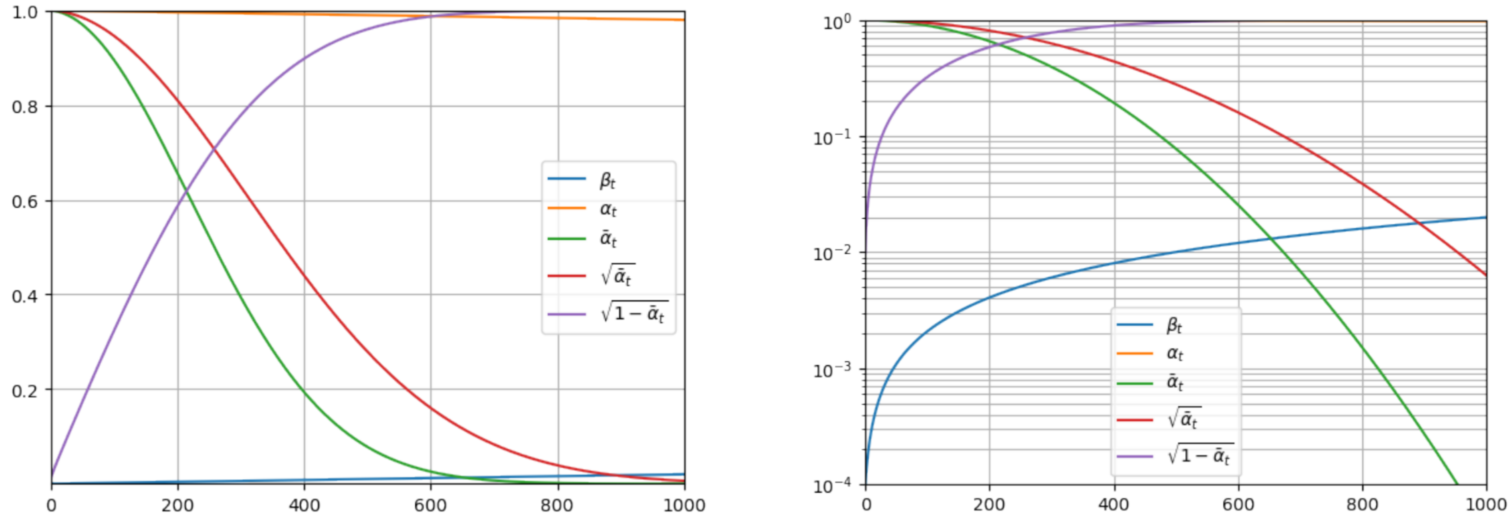
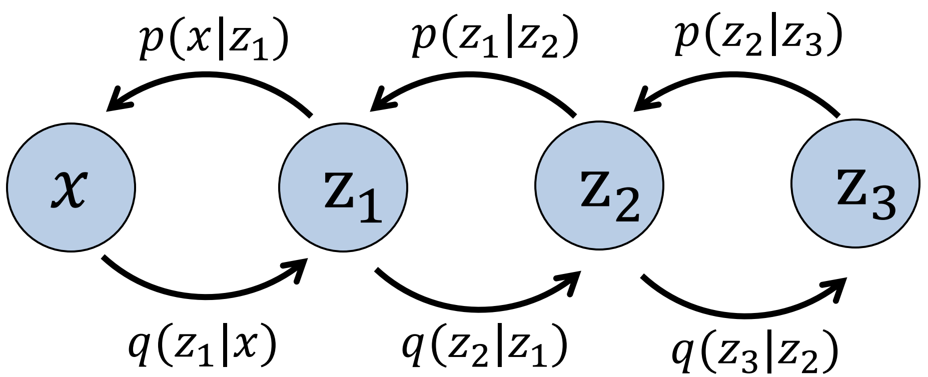



Figure 2: Parameter values for $\beta = [10^{-4}, 0.02]$ over 1000 time steps $t$ using a linear schedule. The information in the two figures are the same, but the right-hand side uses log-scale on the $y$-axis to show the speed of which $\bar{\alpha}_t$ goes towards zero.

# Connection with Hierarchical VAEs

- Diffusion models are special case of Hierarchical VAEs
  - Fixed inference models in forward process
  - Latent variables have same dimension as data
  - ELBO is decomposed to each timestep: faster to train
  - Model is trained with some weighting of ELBO




Ho et al. Denoising Diffusion Probabilistic Models. 2020.
Slide credit to: Ruiqi Gao CS231N

# Poll

# Content

- Diffusion Model Basics
  - Diffusion Models as Stacking VAEs
  - Diffusion Models: Forward, Reverse, Training, Sampling
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Classifier-Free Guidance for Conditional Models
- Applications of Diffusion Models

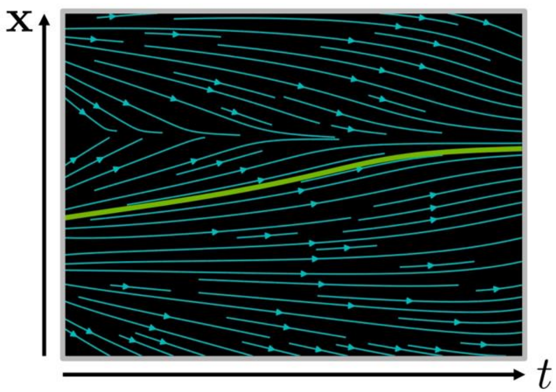
# Why SDEs?

- A unified framework for interpreting diffusion models and score-based generation models
  - Variants of diffusion-based and flow-based models

# Stochastic 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

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

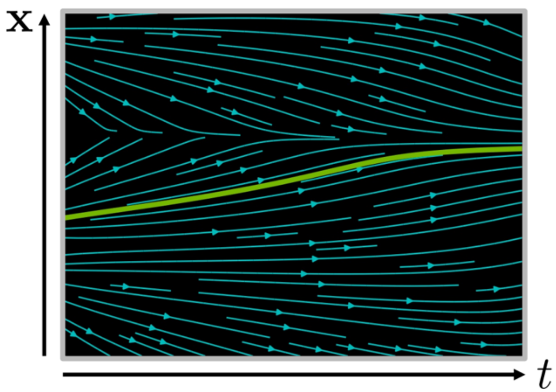


Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

# Stochastic Differential Equations

**Ordinary Differential Equation (ODE):**

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



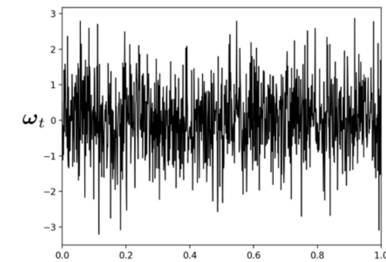

Analytical Solution: $\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$

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

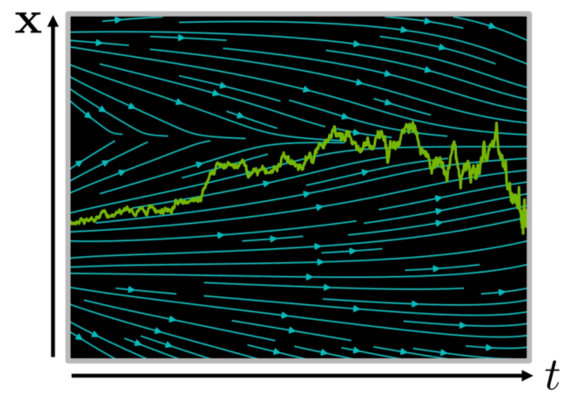
**Stochastic Differential Equation (SDE):**

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

Wiener Process (Gaussian White Noise)




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




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

# Score Matching

- General form of probability density function

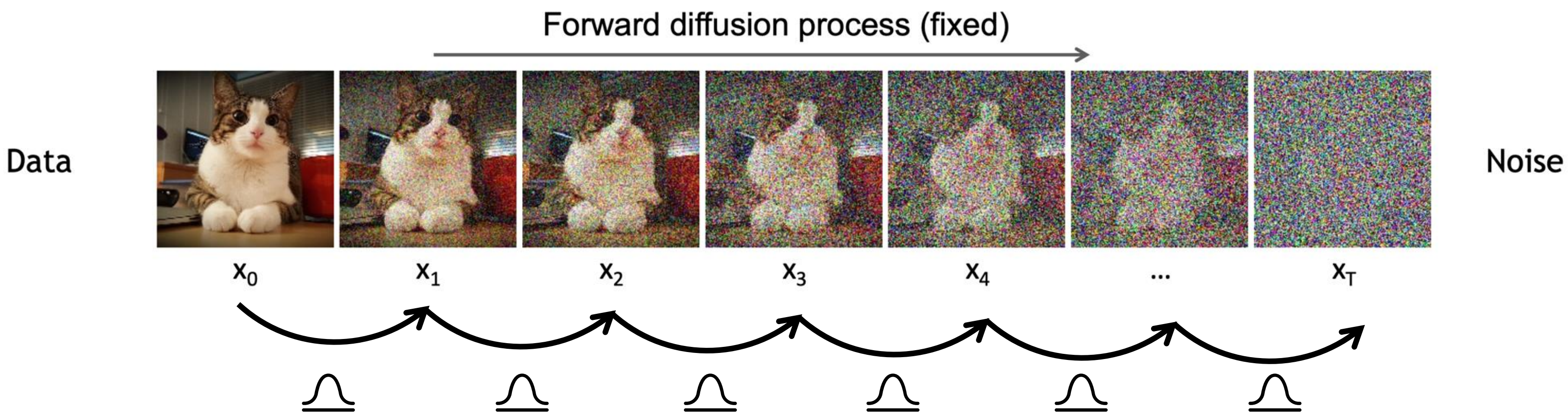
$$p_\theta(\mathbf{x}) = \frac{e^{-f_\theta(\mathbf{x})}}{Z_\theta}$$

- Maximizing the log-likelihood requires us to know $Z_\theta$
  - Often intractable

- Instead, we can model the score function

$$\nabla_\mathbf{x} \log p(\mathbf{x})$$

# Forward Diffusion Process as SDEs



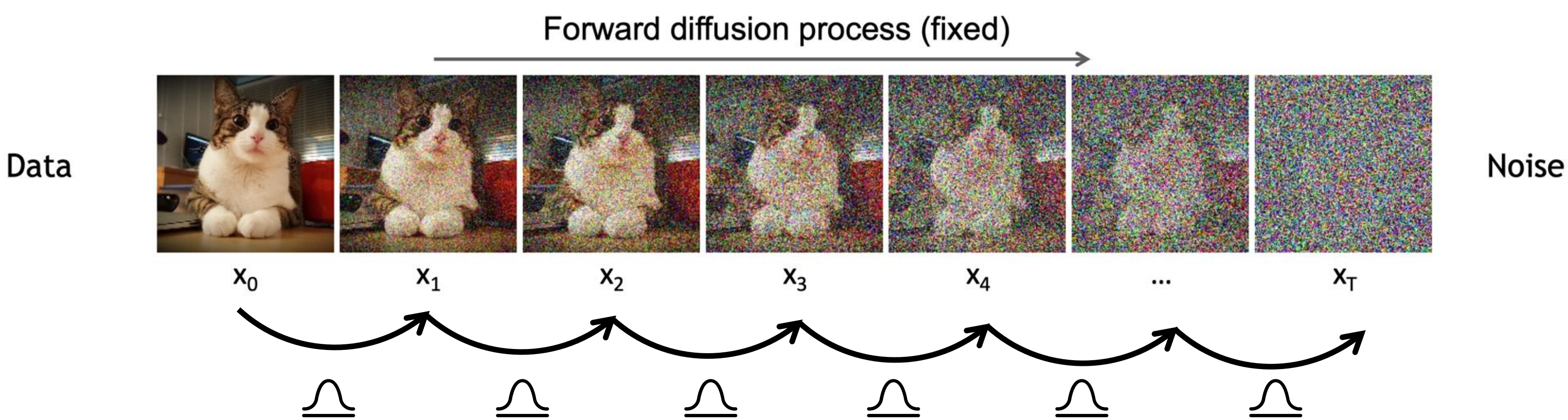

- Consider a forward process with many many small steps (continuous time)

$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) = \mathcal{N}\left(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t \mathbf{I}\right)$$

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

# Forward Diffusion Process as SDEs




- Consider a forward process with many many small steps

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t \mathbf{I}\right)$$

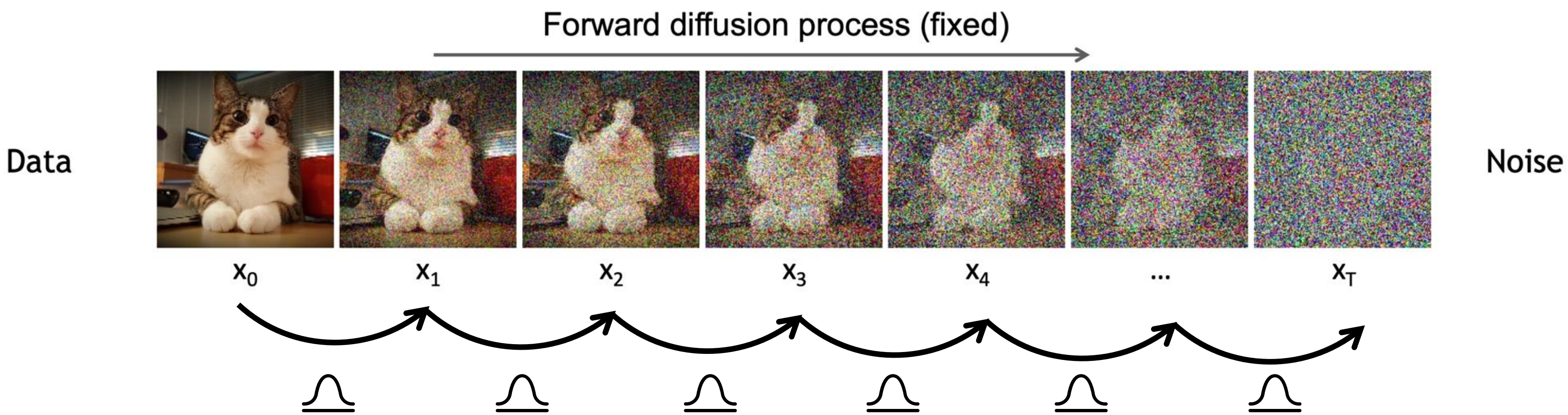
$$\begin{aligned}\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= \sqrt{1-\beta(t)\Delta t}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I})\end{aligned} \qquad (\beta_t := \beta(t)\Delta t)$$

Allows different size along t — Step size



Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

# Forward Diffusion Process as SDEs



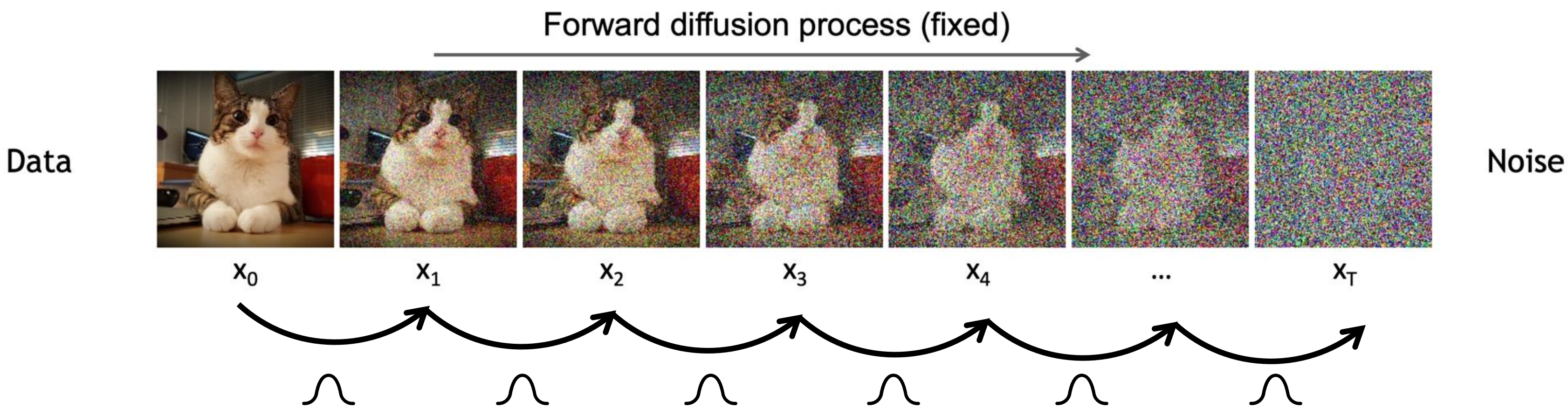

- Consider a forward process with many many small steps

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t \mathbf{I}\right)$$

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

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

# Forward Diffusion Process as SDEs




- An iterative update that can be viewed as SDEs

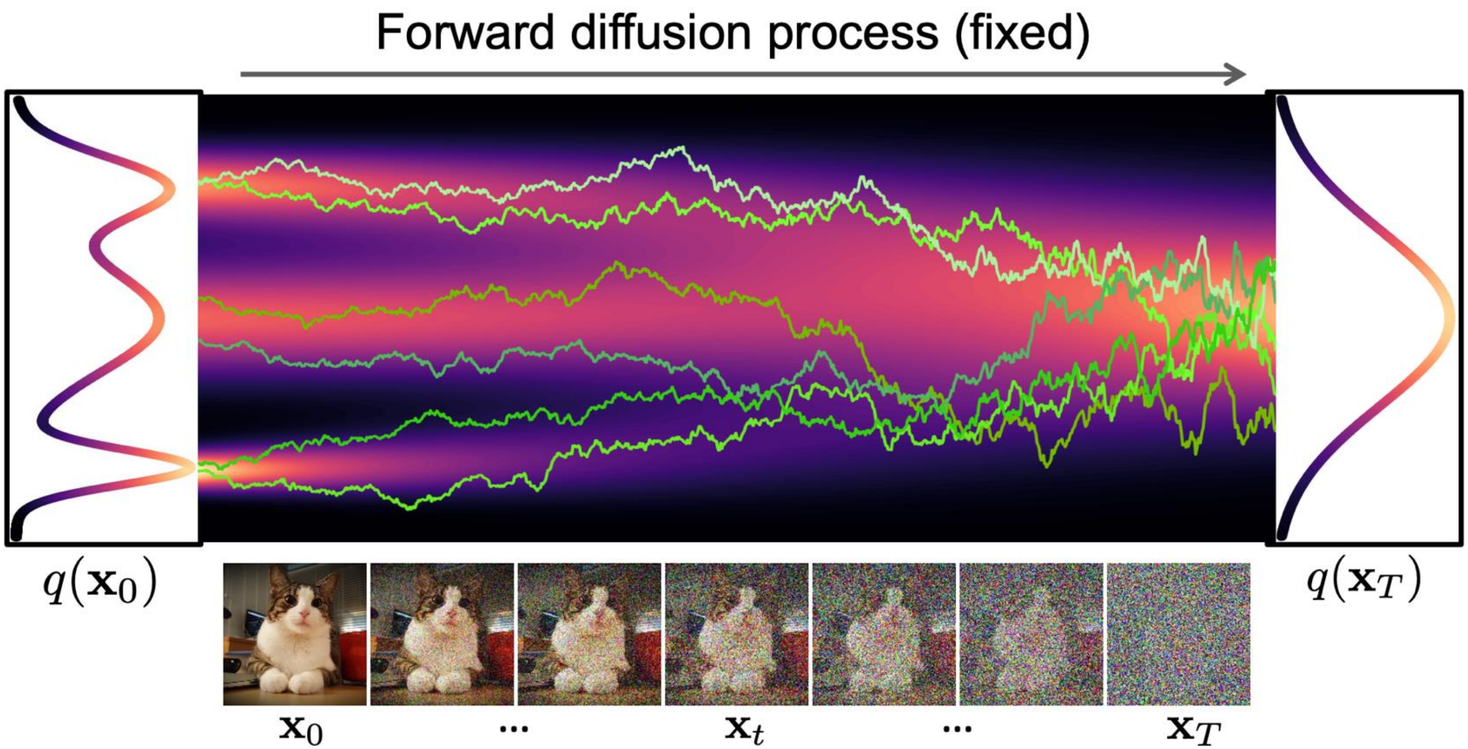
$$\mathbf{x}_t \approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

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

Stochastic Differential Equation (SDE)

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

# Forward Diffusion Process as SDEs




$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\text{Drift Term}} + \underbrace{\sqrt{\beta(t)}\mathrm{d}\boldsymbol{\omega}_t}_{\text{Diffusion Term}}$$

Drift Term
(Pulls toward the mode)

Diffusion Term
(Injects Noise)

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

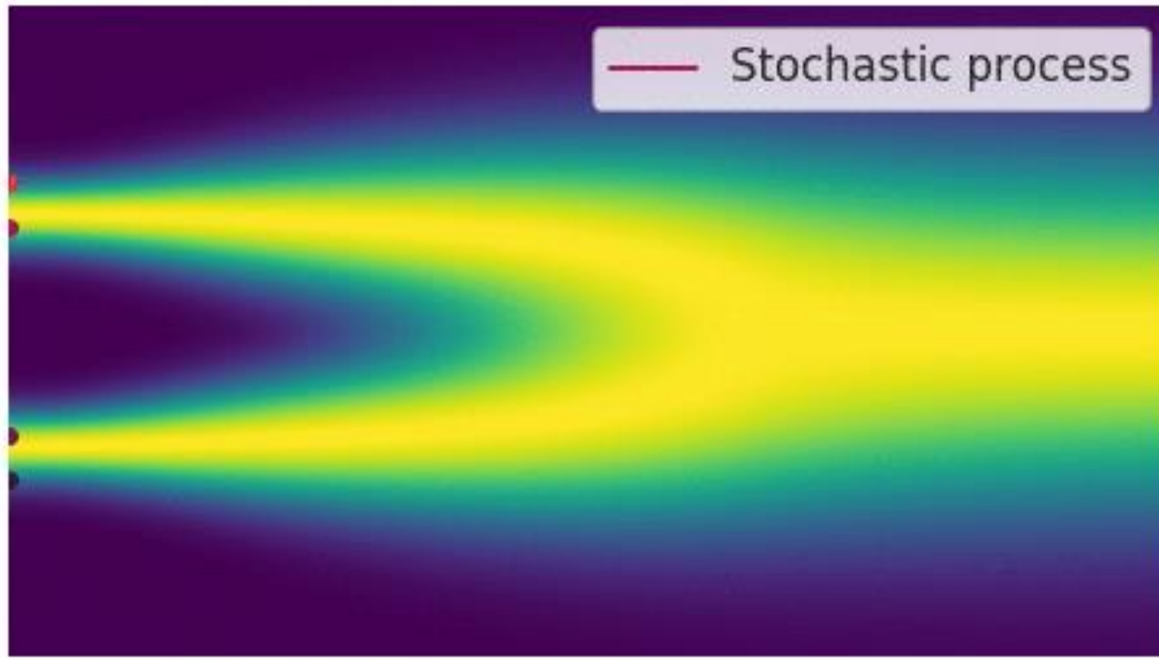
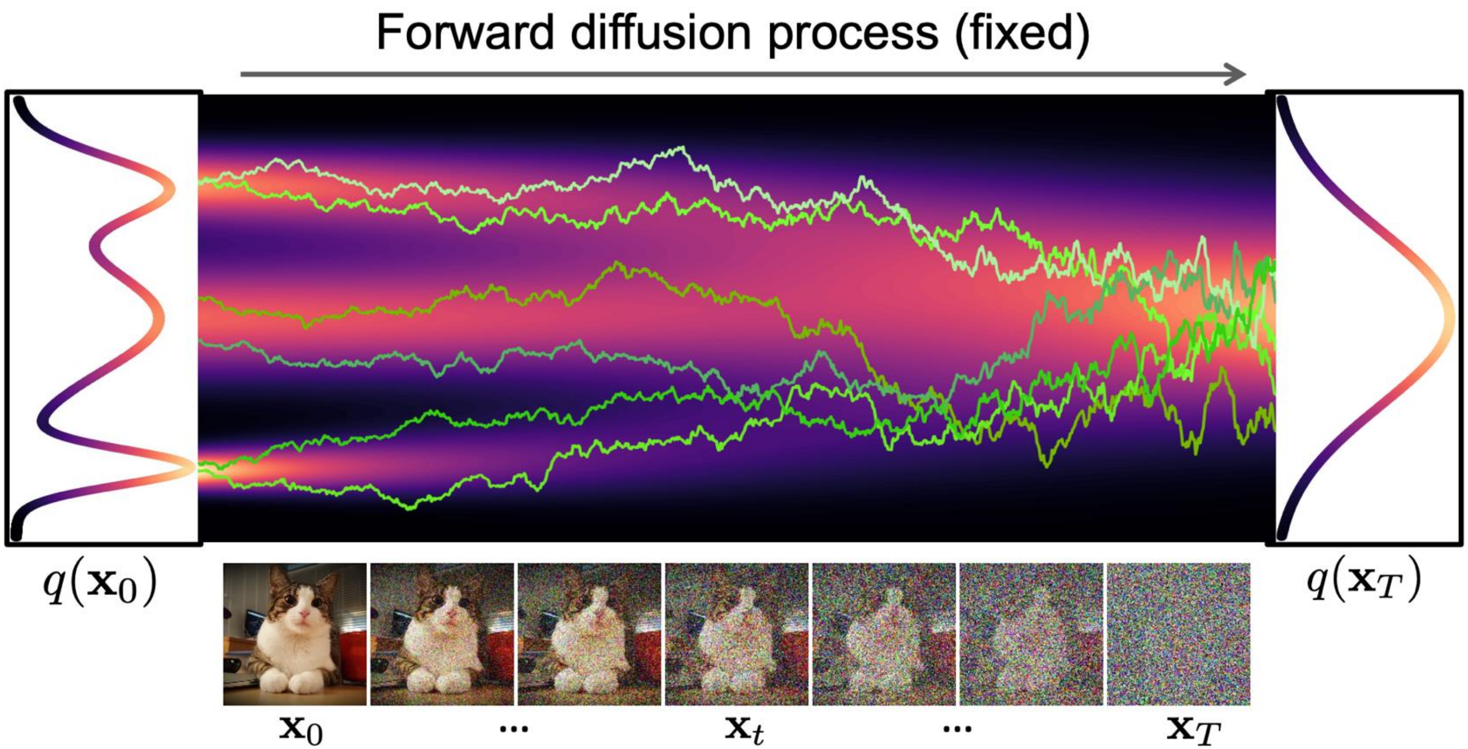





Figure credit to: https://yang-song.net/blog/2021/score/

# Generative Reverse SDEs

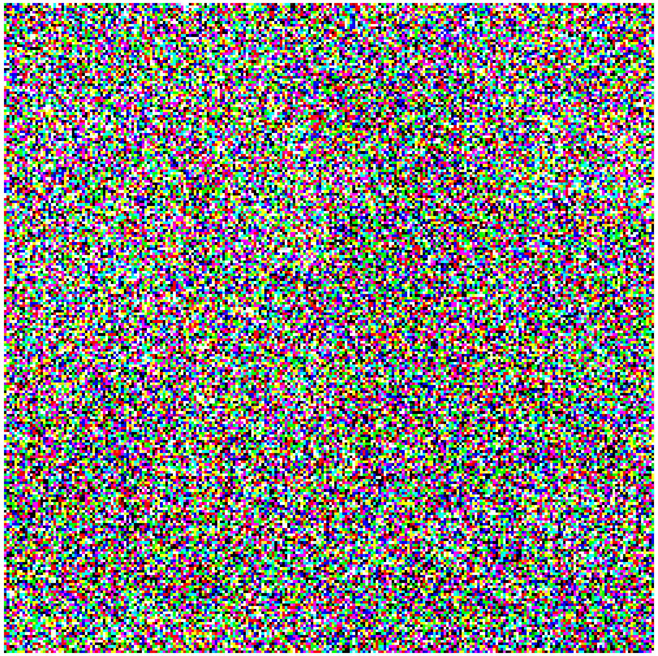



- The forward SDE has a reverse form:

$$d\mathbf{x}_t = \left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)\right]dt + \sqrt{\beta(t)}d\bar{\boldsymbol{\omega}}_t$$



Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/



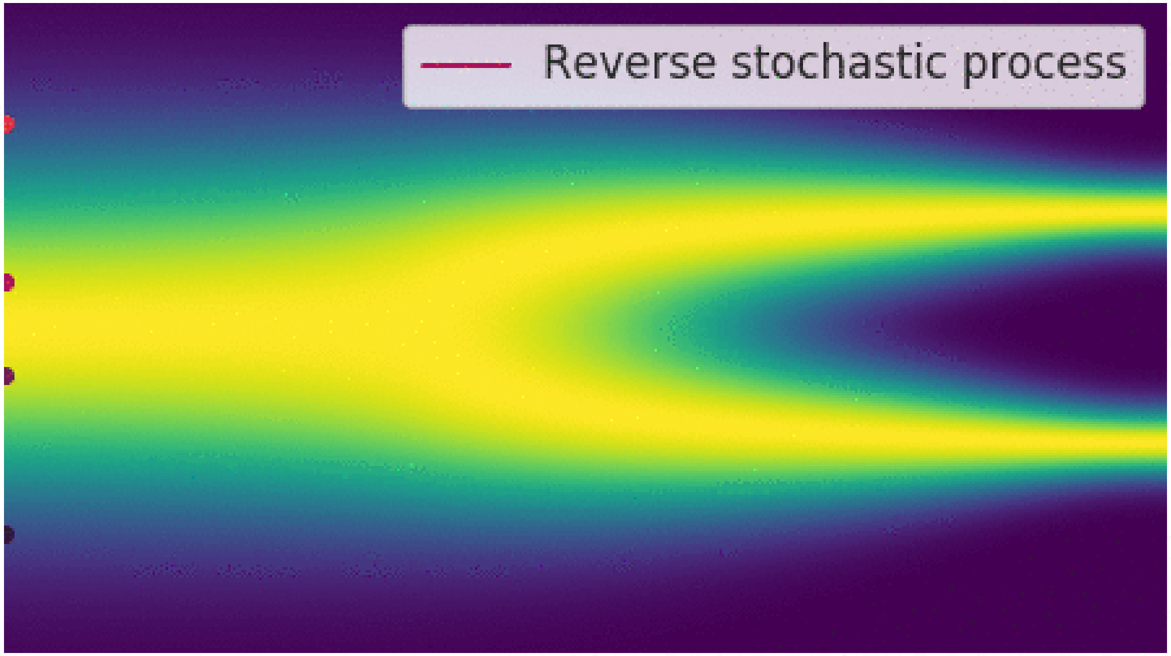
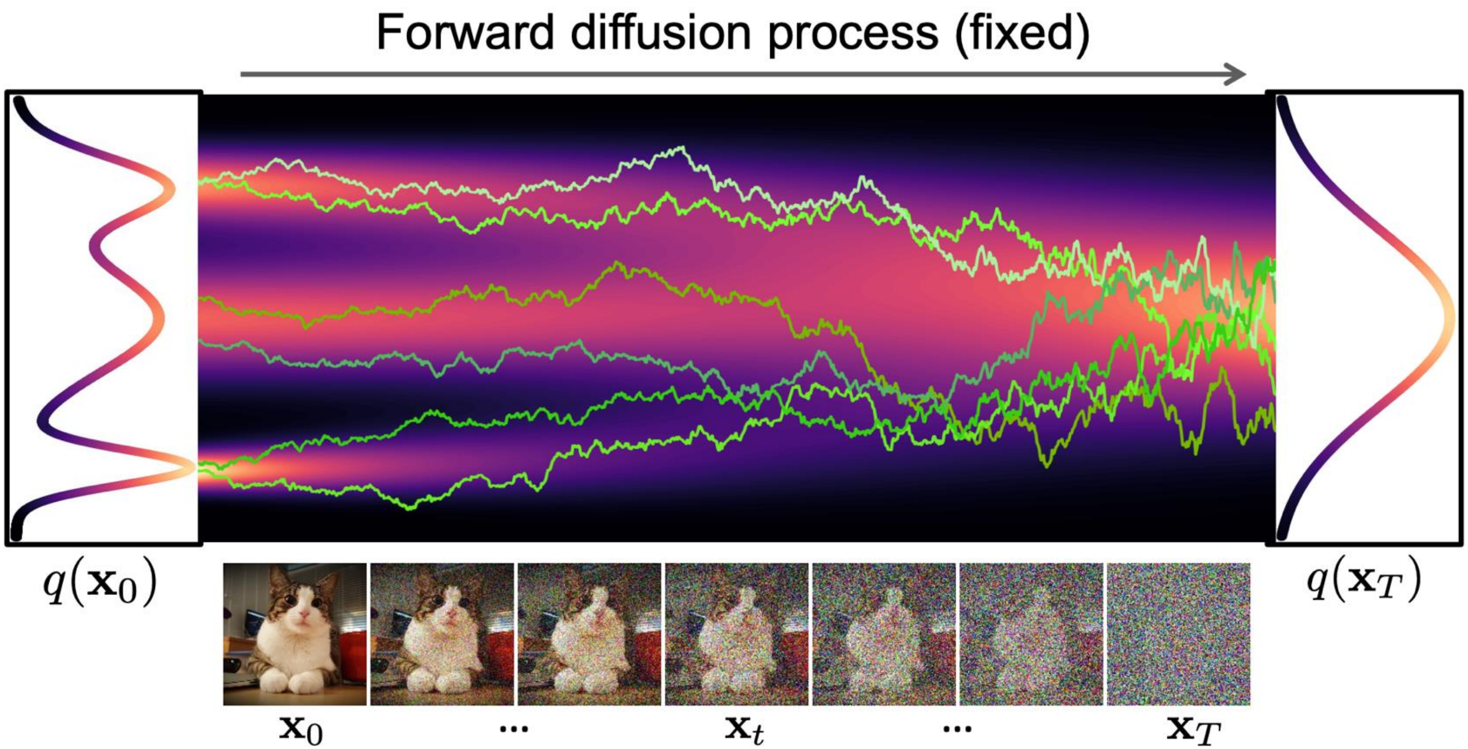



Figure credit to: https://yang-song.net/blog/2021/score/

# Generative Reverse SDEs




- The forward SDE has a reverse form:

$$\mathrm{d}\mathbf{x}_t = \left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\nabla_{\mathbf{x}_t} \log q_t\left(\mathbf{x}_t\right)\right] \mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

Score function

How to get it?



Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

# Denoising Score Matching

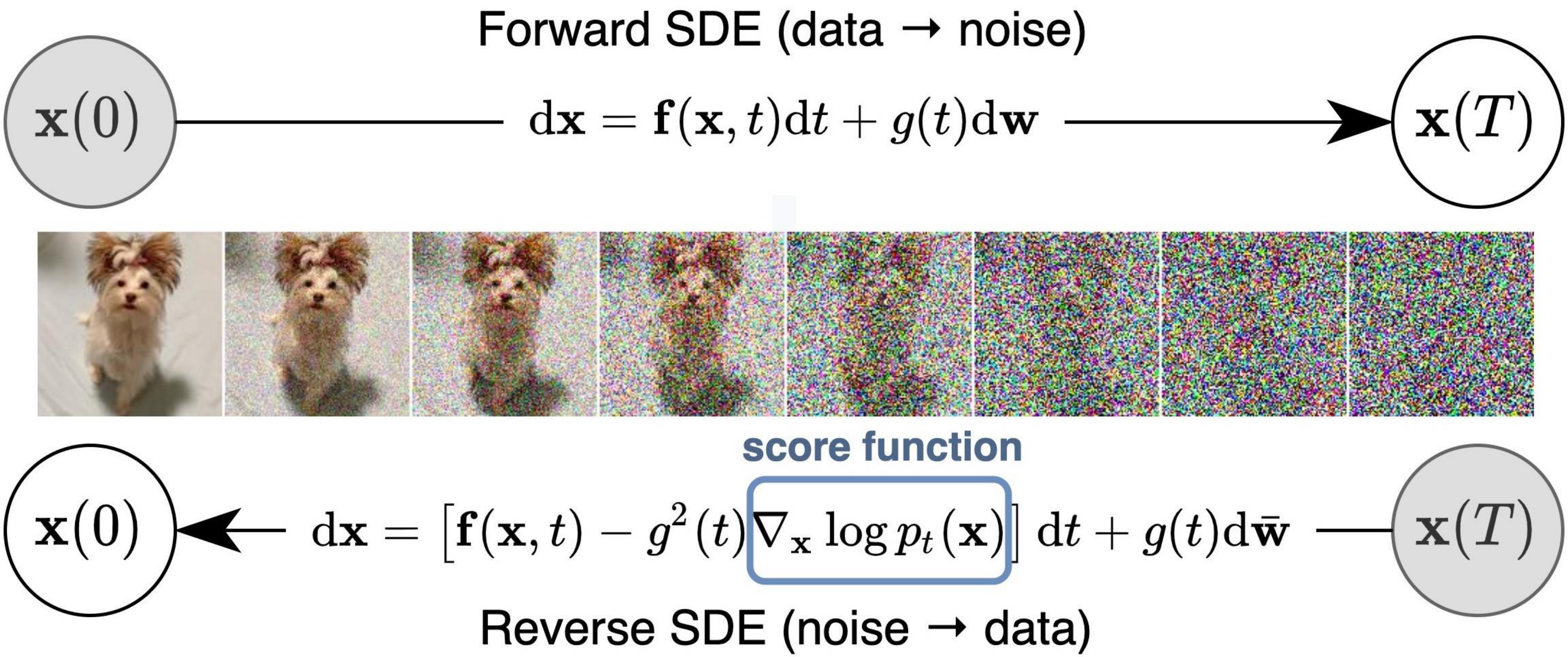
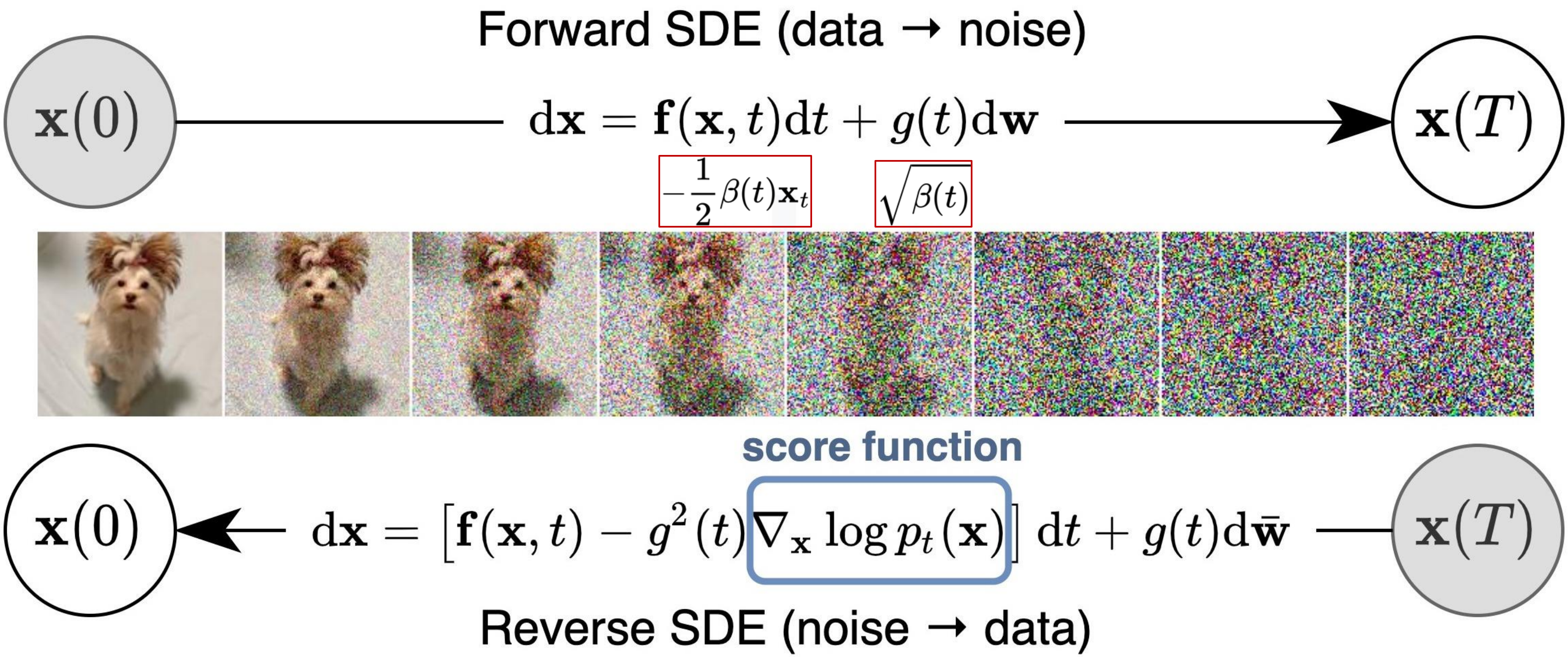





Figure credit to: https://yang-song.net/blog/2021/score/

# Denoising Score Matching

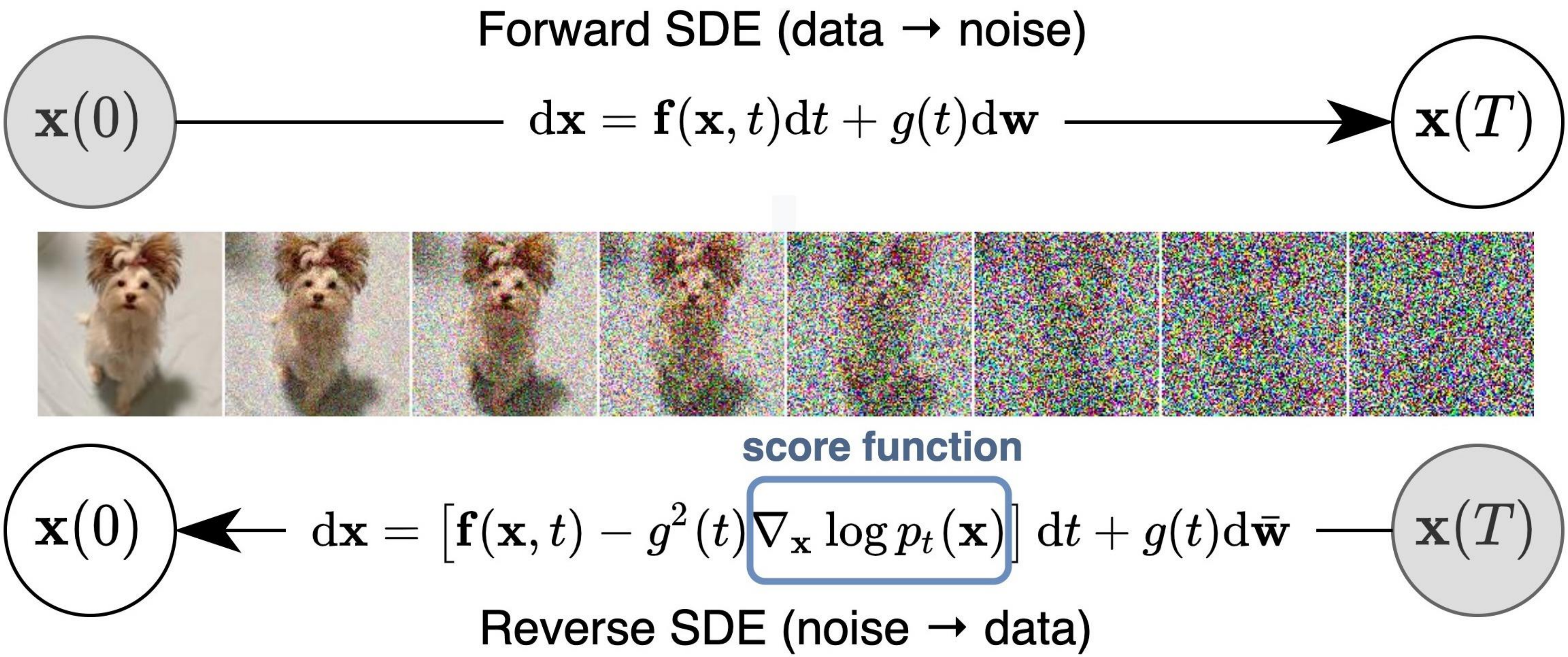



$$\mathrm{d}\mathbf{x}_t = \left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\nabla_{\mathbf{x}_t} \log q_t\left(\mathbf{x}_t\right)\right]\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\overline{\boldsymbol{\omega}}_t$$



Figure credit to: https://yang-song.net/blog/2021/score/

# Denoising Score Matching




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

Looks similar?



Figure credit to: https://yang-song.net/blog/2021/score/

# Denoising Score Matching

- Denoising score matching objective

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

- Re-parametrized sampling:

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

- Score function:

$$\nabla_{\mathbf{x}_t} \log q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right) = -\nabla_{\mathbf{x}_t} \frac{\left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right)^2}{2\sigma_t^2} = -\frac{\mathbf{x}_t - \alpha_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} - \alpha_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}}{\sigma_t}$$

- Denoising network:

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

- Final objective:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)} \mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \hat{w}(t) \cdot \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right) \right\|_2^2 \quad \hat{w}(t) = \frac{\tilde{w}(t)}{\sigma_t}$$

# Weighted Diffusion Objective

- Denoising score matching objective with loss weighting

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

- Loss weights trade-off between
  - good perceptual quality: $\lambda(t) = \sigma_t^2$
  - maximum likelihood: $\lambda(t) = \beta(t)$

- More complicated model parametrization and loss weighting leads to different diffusion model variants in the literature!

# Poll

# Content

- Diffusion Model Basics
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Denoising Diffusion Implicit Model (DDIM)
- Conditional Diffusion Models
- Applications of Diffusion Models

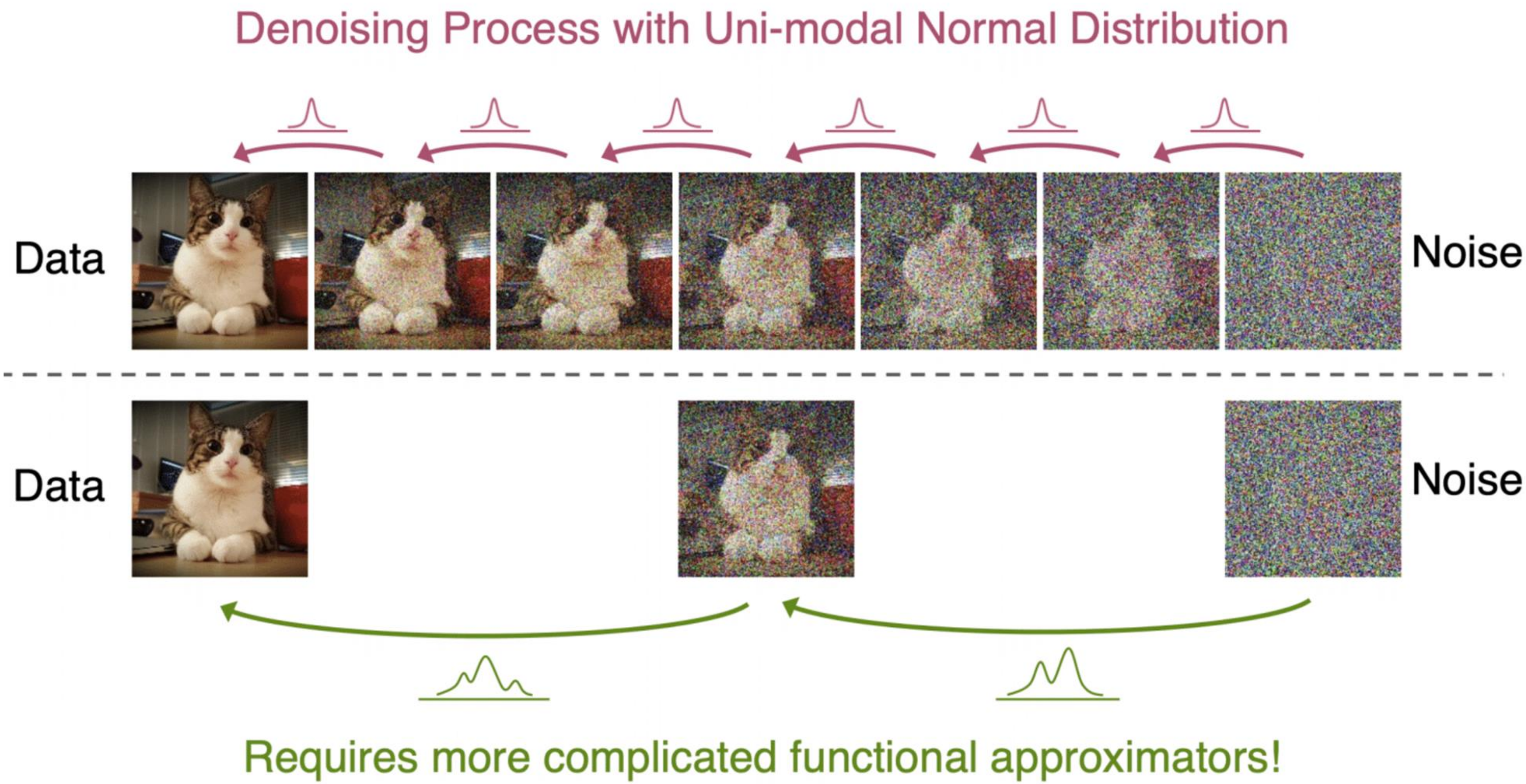
# Many Steps in Diffusion

- Slow in generation

- In Training, we randomly sample one time step

- But in inference, we must transit from T to 0
  - 1000 steps
  - extremely slow for raw images/signals

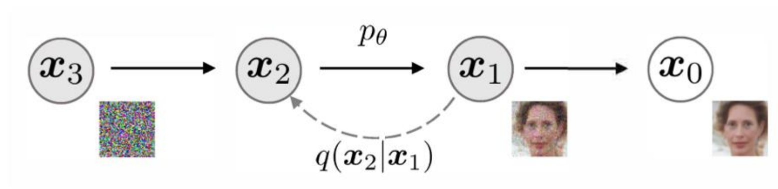
# Can we do generation with less steps?






Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

# DDPM

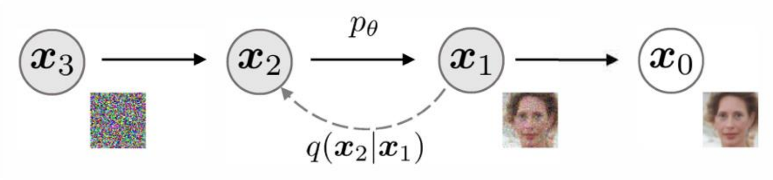



$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)=\mathcal{N}\left(\mathbf{x}_{\mathbf{t}} ; \sqrt{1-\beta_t} \mathbf{x}_{\mathbf{t}-\mathbf{1}}, \beta_t \mathbf{I}\right)$$

$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{\bar{\alpha}_t} \mathbf{x}_0,(1-\bar{\alpha}_t) \mathbf{I})\right)$$

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

# DDPM




Only depends on previous step

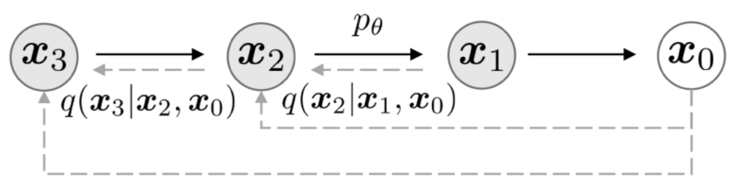
$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)=\mathcal{N}\left(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}\right)$$

$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right)\Big)$$

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

Only used during training

# DDIM




$$q_\sigma\left(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0\right) := q_\sigma\left(\boldsymbol{x}_T \mid \boldsymbol{x}_0\right) \prod_{t=2}^{T} q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right)$$

$$q_\sigma\left(\boldsymbol{x}_T \mid \boldsymbol{x}_0\right) = \mathcal{N}\left(\sqrt{\alpha_T}\boldsymbol{x}_0, (1-\alpha_T)\boldsymbol{I}\right)$$

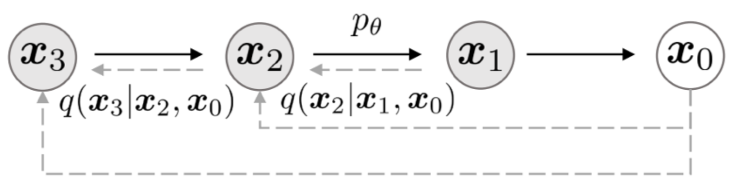
$$q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\boldsymbol{I}\right)$$

- A Non-Markovian Forward Process

$$q_\sigma\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0\right) = \frac{q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) q_\sigma\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0\right)}{q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0\right)}$$

Song et al. Denoising Diffusion Implicit Models. 2021.

# DDIM




- Backward process

$$p_\theta^{(t)}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \begin{cases} \mathcal{N}\left(f_\theta^{(1)}(\boldsymbol{x}_1), \sigma_1^2 \boldsymbol{I}\right) & \text{if } t = 1 \\ q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, f_\theta^{(t)}(\boldsymbol{x}_t)\right) & \text{otherwise,} \end{cases}$$

$$f_\theta^{(t)}(\boldsymbol{x}_t) := \left(\boldsymbol{x}_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta^{(t)}(\boldsymbol{x}_t)\right) / \sqrt{\alpha_t}$$

Song et al. Denoising Diffusion Implicit Models. 2021.

# DDPM vs DDIM

**Algorithm** DDPM Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
**for all** $t$ from $T$ to 1 **do**
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mu \leftarrow \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t))$
  $\mathbf{x}_{t-1} \leftarrow \mu + \sigma_t\epsilon$ Stochastic
**end for**
**return** $\mathbf{x}_0$

**Algorithm** DDIM Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
**for all** $t$ from $T$ to 1 **do**
  $\bar{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t)$
  $\bar{\mathbf{x}}_0 \leftarrow \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}}{\sqrt{\bar{\alpha}_t}}$ Estimate $\mathbf{x}_0$
  $\mathbf{x}_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\bar{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\bar{\epsilon}$
**end for**
**return** $\mathbf{x}_0$

# DDIM with Fewer Steps Sampling

**DDIM**

Increasing
Sub-sequence

$$[1,\ldots,T] \Longrightarrow [\tau_0 = 0,\ldots,\tau_S = T]$$

E.g., $\tau = [0,10,20,30,\ldots,1000]$

**Algorithm** Original DDIM Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
**for all** $t$ from $T$ to 1 **do**
  $\bar{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t)$
  $\bar{\mathbf{x}}_0 \leftarrow \dfrac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}}{\sqrt{\bar{\alpha}_t}}$
  $\mathbf{x}_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\bar{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\bar{\epsilon}$
**end for**
**return** $\mathbf{x}_0$

**Algorithm** Fewer-Steps DDIM Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
**for all** $s$ from $S$ to 1 **do**
  $t \leftarrow \tau_s$
  $t' \leftarrow \tau_{s-1}$
  $\bar{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t)$
  $\bar{\mathbf{x}}_0 \leftarrow \dfrac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}}{\sqrt{\bar{\alpha}_t}}$
  $\mathbf{x}_{t'} \leftarrow \sqrt{\bar{\alpha}_{t'}}\bar{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t'}}\bar{\epsilon}$
**end for**
**return** $\mathbf{x}_0$

# DDIM Results

Table 1: CIFAR10 and CelebA image generation measured in FID. $\eta = 1.0$ and $\hat{\sigma}$ are cases of DDPM (although Ho et al. (2020) only considered $T = 1000$ steps, and $S < T$ can be seen as simulating DDPMs trained with $S$ steps), and $\eta = 0.0$ indicates DDIM.

| | | CIFAR10 (32 × 32) | | | | | CelebA (64 × 64) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $S$ | 10 | 20 | 50 | 100 | 1000 | 10 | 20 | 50 | 100 | 1000 |
| $\eta$ | 0.0 | **13.36** | **6.84** | **4.67** | **4.16** | 4.04 | **17.33** | **13.73** | **9.17** | **6.53** | 3.51 |
| | 0.2 | 14.04 | 7.11 | 4.77 | 4.25 | 4.09 | 17.66 | 14.11 | 9.51 | 6.79 | 3.64 |
| | 0.5 | 16.66 | 8.35 | 5.25 | 4.46 | 4.29 | 19.86 | 16.06 | 11.01 | 8.09 | 4.28 |
| | 1.0 | 41.07 | 18.36 | 8.01 | 5.78 | 4.73 | 33.12 | 26.03 | 18.48 | 13.93 | 5.98 |
| | $\hat{\sigma}$ | 367.43 | 133.37 | 32.72 | 9.99 | **3.17** | 299.71 | 183.83 | 71.71 | 45.20 | **3.26** |

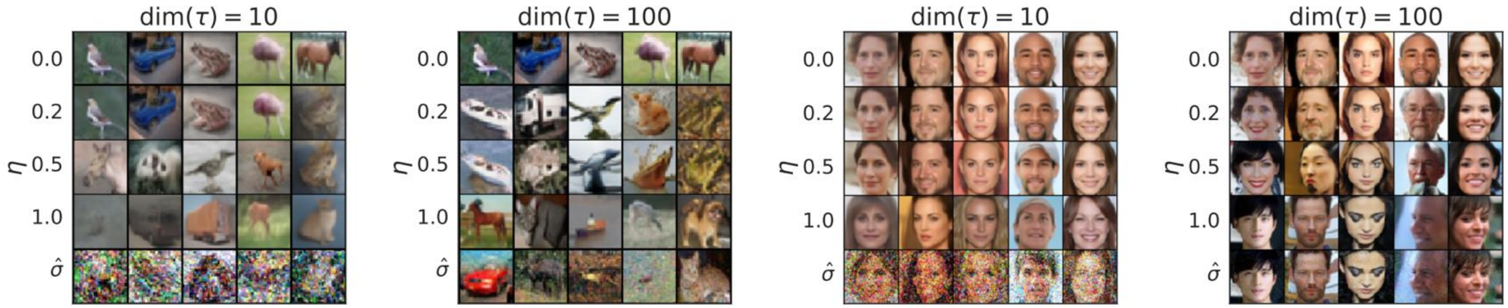



Figure 3: CIFAR10 and CelebA samples with $\dim(\tau) = 10$ and $\dim(\tau) = 100$.

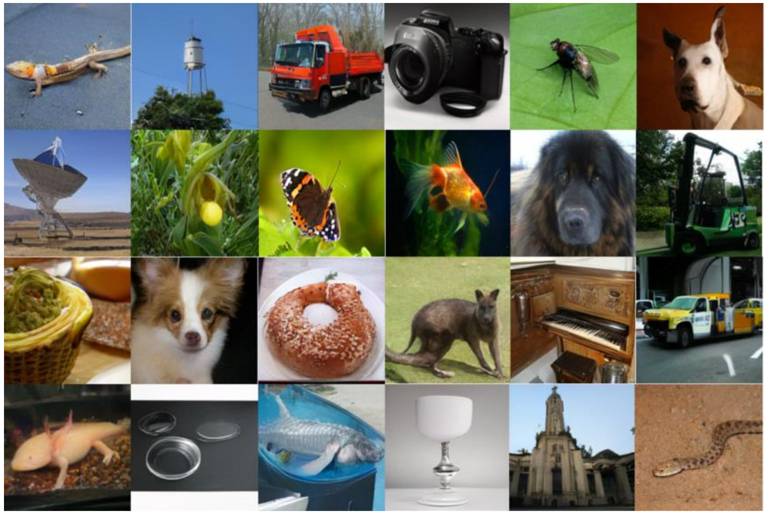
# Poll

# Content

- Diffusion Model Basics
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Denoising Diffusion Implicit Model (DDIM)
- Conditional Diffusion Models
- Applications of Diffusion Models

# Conditional Diffusion Models

- Un-conditional
- Conditional



$$p\left(\boldsymbol{x}_{0:T}\right)=p\left(\boldsymbol{x}_{T}\right)\prod_{t=1}^{T}p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_{t}\right)$$

$$p\left(\boldsymbol{x}_{0:T}\mid y\right)=p\left(\boldsymbol{x}_{T}\right)\prod_{t=1}^{T}p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_{t},y\right)$$

More controllable!

# Conditional Score Matching

- Score matching with conditional information

$$\begin{aligned}\nabla \log p\left(\boldsymbol{x}_t \mid y\right) &= \nabla \log \left(\frac{p\left(\boldsymbol{x}_t\right) p\left(y \mid \boldsymbol{x}_t\right)}{p(y)}\right) \\ &= \nabla \log p\left(\boldsymbol{x}_t\right)+\nabla \log p\left(y \mid \boldsymbol{x}_t\right)-\nabla \log p(y) \\ &= \underbrace{\nabla \log p\left(\boldsymbol{x}_t\right)}_{\text{unconditional score}}+\underbrace{\nabla \log p\left(y \mid \boldsymbol{x}_t\right)}_{\text{adversarial gradient}}\end{aligned}$$

# Classifier Guidance

- Use a discriminative classifier for $\nabla \log p(y \mid \boldsymbol{x}_t)$

$$\nabla \log p(\boldsymbol{x}_t \mid y) = \nabla \log p(\boldsymbol{x}_t) + \gamma \nabla \log p(y \mid \boldsymbol{x}_t)$$

- $\gamma$ controls the strength of the condition

- Limitations:
  - Need a separate classifier
  - Conditioning depends on the performance of classifier

# Classifier-Free Guidance

- Score matching with conditional information

$$\nabla \log p(\boldsymbol{x}_t \mid y) = \nabla \log p(\boldsymbol{x}_t) + \gamma \nabla \log p(y \mid \boldsymbol{x}_t)$$

$$\nabla \log p(y \mid \boldsymbol{x}_t) = \nabla \log p(\boldsymbol{x}_t \mid y) - \nabla \log p(\boldsymbol{x}_t)$$

- Classifier-free guidance

$$\begin{aligned}\nabla \log p(\boldsymbol{x}_t \mid y) &= \nabla \log p(\boldsymbol{x}_t) + \gamma \left(\nabla \log p(\boldsymbol{x}_t \mid y) - \nabla \log p(\boldsymbol{x}_t)\right) \\ &= \nabla \log p(\boldsymbol{x}_t) + \gamma \nabla \log p(\boldsymbol{x}_t \mid y) - \gamma \nabla \log p(\boldsymbol{x}_t) \\ &= \underbrace{\gamma \nabla \log p(\boldsymbol{x}_t \mid y)}_{\text{conditional score}} + \underbrace{(1-\gamma) \nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}}\end{aligned}$$

Ho et al. Classifier-Free Diffusion Guidance. 2022.

# Training of Classifier-Free Guidance

- For conditional embeddings
  - Randomly drop **p** original conditionals with an additional unconditional class

$$\mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon-\epsilon_\theta\left(z_t,t,\tau_\theta(y)\right)\|_2^2\right]$$

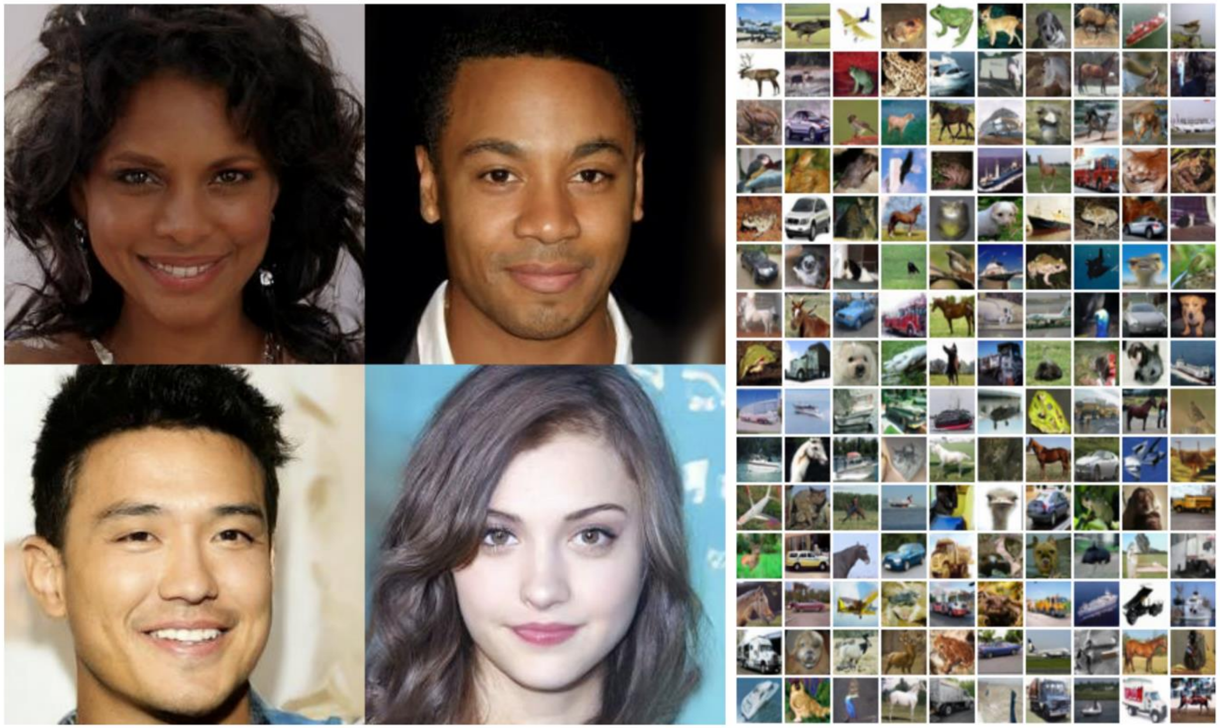
Ho et al. Classifier-Free Diffusion Guidance. 2022.

# Content

- Diffusion Model Basics
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Denoising Diffusion Implicit Model (DDIM)
- Conditional Diffusion Models
- Applications of Diffusion Models

# DDPM

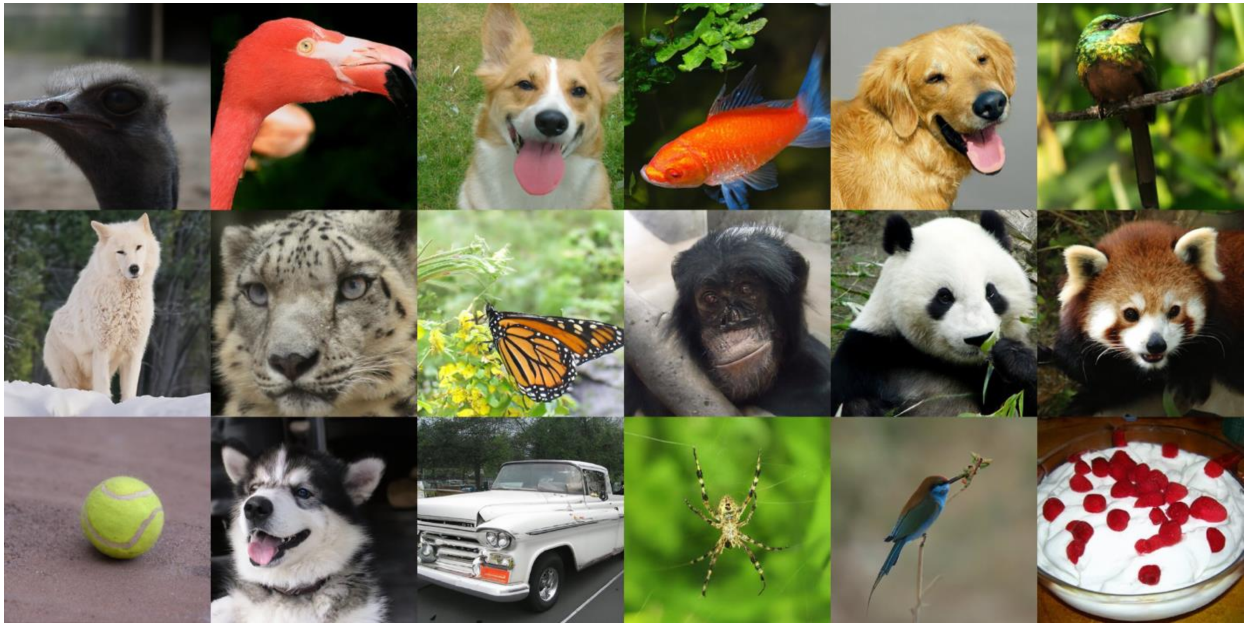
- Training diffusion models on raw images with a U-Net model



Ho et al. Denoising Diffusion Probabilistic Models. 2020.

# Diffusion Models Beat GANs

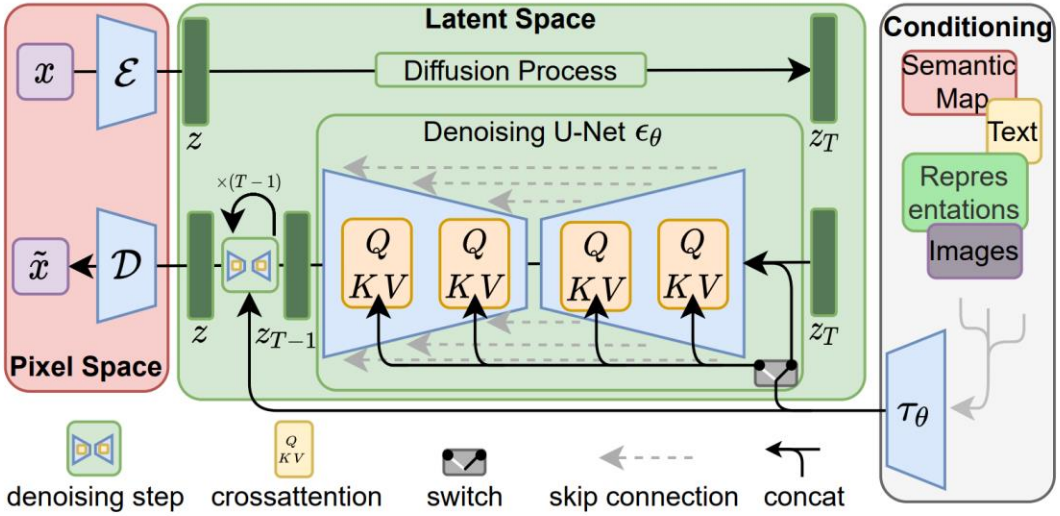
- Larger denoising model with sophisticated design
  - Adaptive group normalization
  - Attention layers in U-Net



Dhariwal et al. Diffusion Models Beat GANs on Image Synthesis. 2021.

# Latent Diffusion Models (LDMs)

- Learn diffusion on VAE's latent
  - Yet another VAE! Except pre-trained.



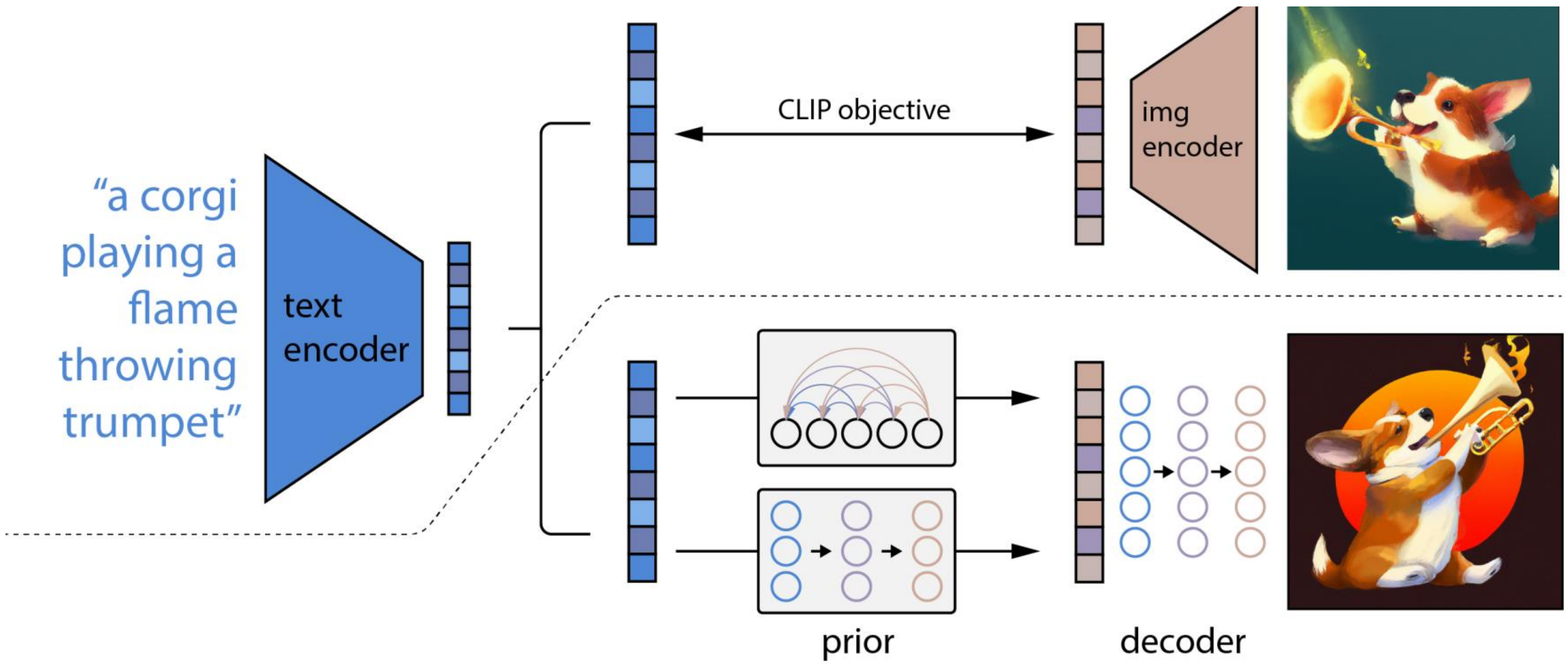

Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models. 2022.

# Stable Diffusion

- Large-scale text-conditional LDMs
  - With VAEs trained also on larger datasets

Stability AI. https://github.com/Stability-AI/stablediffusion

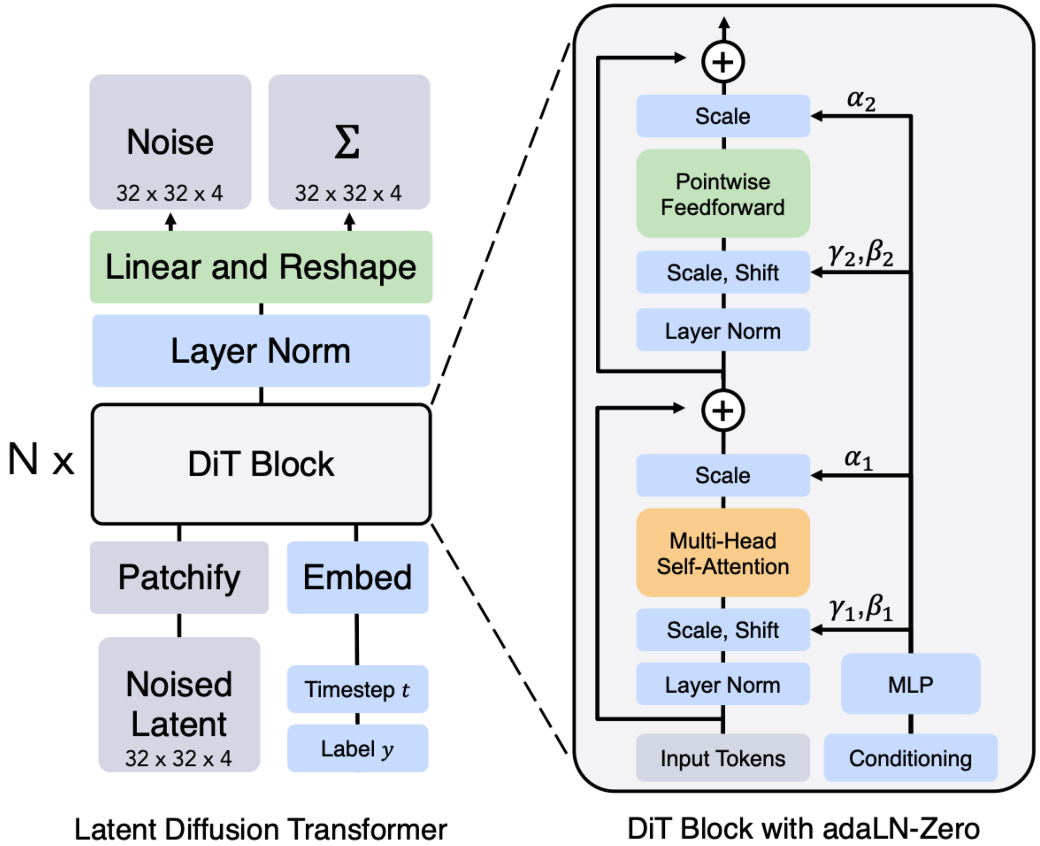
# DALLE




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

# DiT

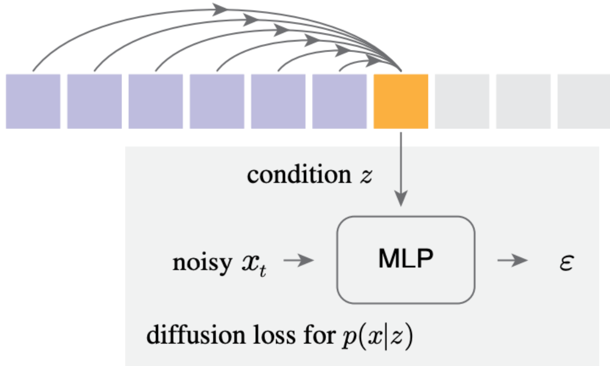
- A transformer architecture for diffusion models

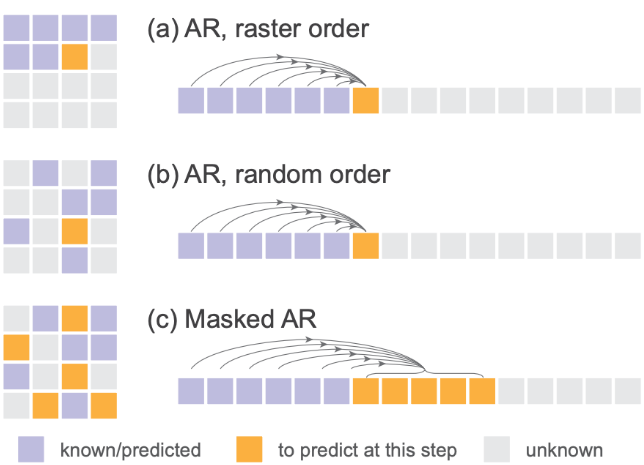



Peebles et al. Scalable Diffusion Models with Transformers. 2020.

# MAR

- An autoregressive model with diffusion loss




Li et al. Autoregressive Image Generation without Vector Quantization. 2024.