



# More Generative Models

Jan 29, 2024

**Tali Dekel**

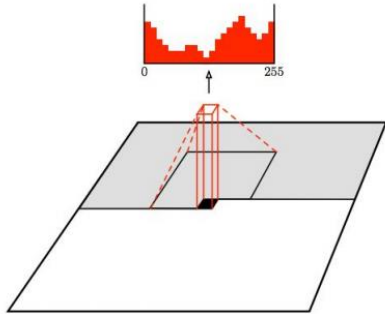


# Learn the probability distribution of natural images

## Generative Models Taxonomy

### Explicitly density estimation

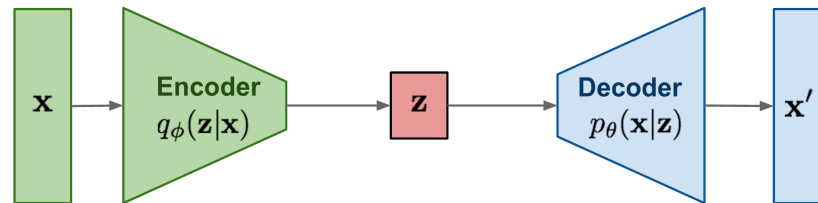
#### Autoregressive image generation



PixelRNN / PixelCNN

Directly maximize the log likelihood of the training data

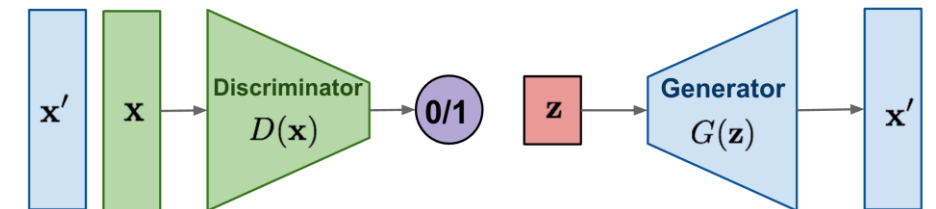
#### Variational Autoencoders (VAEs)



Maximize the variational lower bound of the data likelihood

### Implicit density estimation

#### Generative Adversarial Networks (GANs)



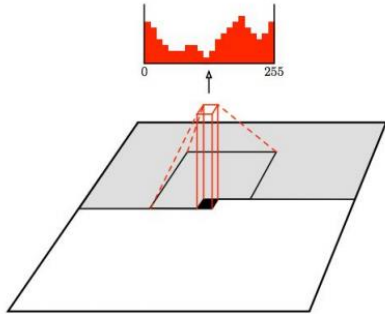
Does not model the density function, only care about sampling

# Learn the probability distribution of natural images

## Generative Models Taxonomy

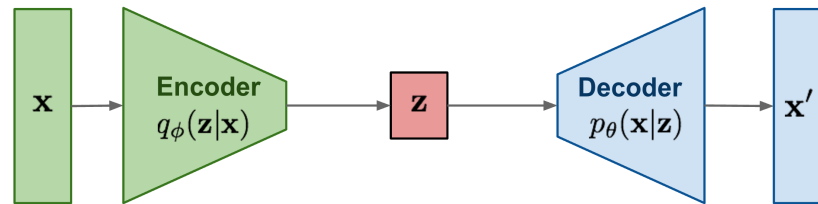
### Explicitly density estimation

Autoregressive image generation



PixelRNN / PixelCNN

Variational Autoencoders (VAEs)

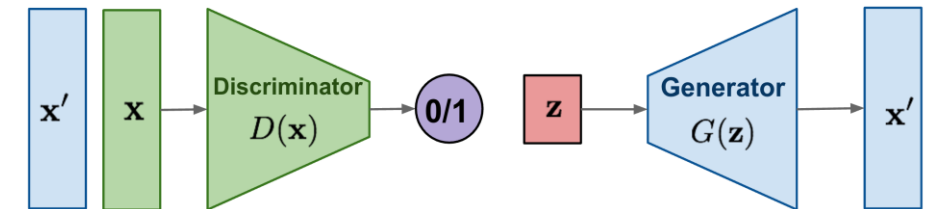


Vector Quantize - Variational Autoencoders (VQ-VAEs)

Diffusion Models

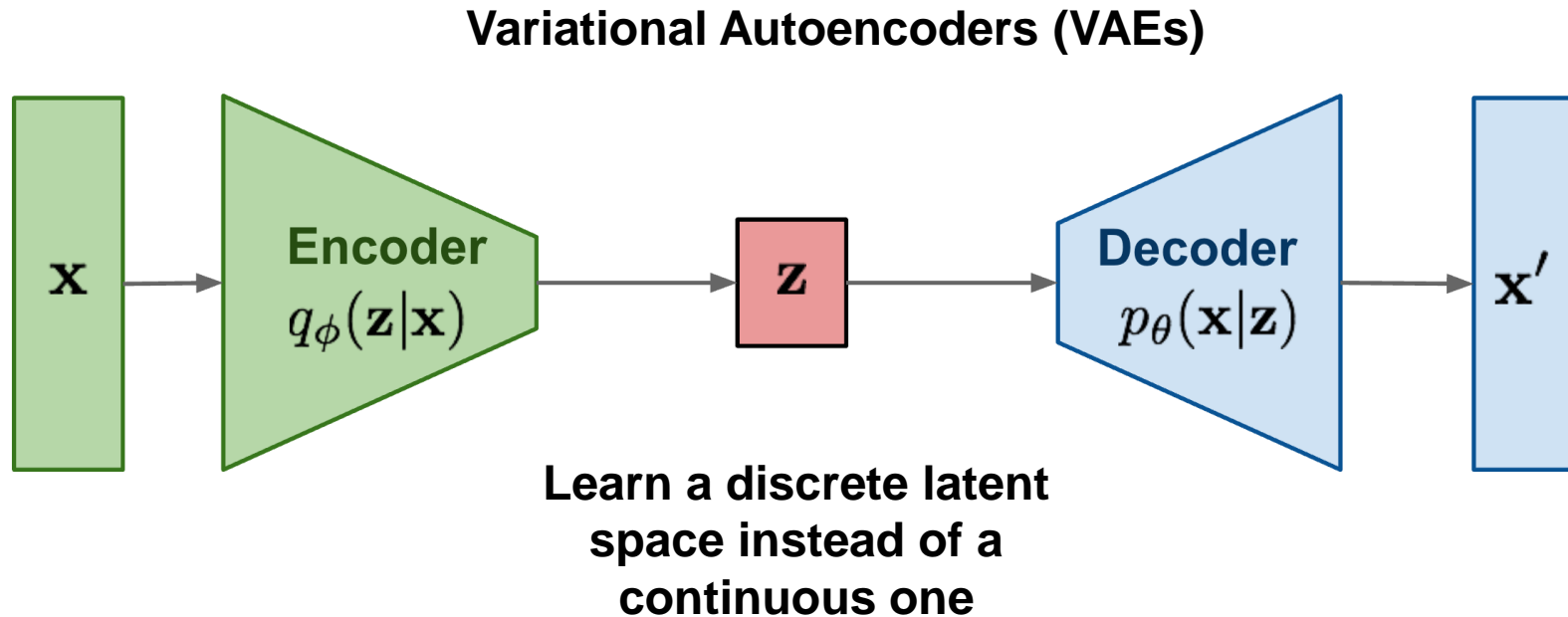
### Implicit density estimation

Generative Adversarial Networks (GANs)



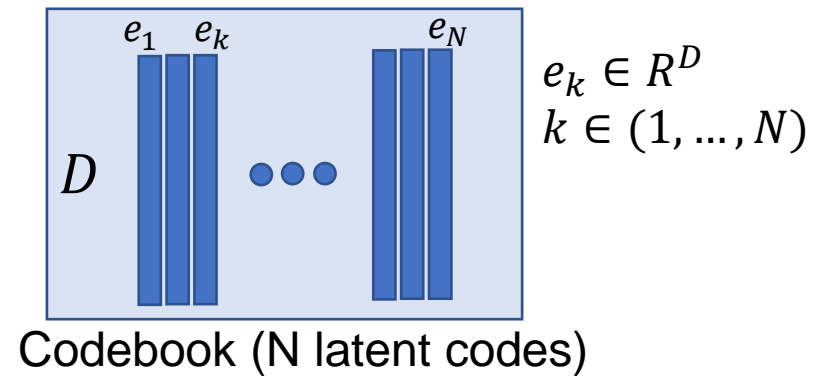
Vector Quantize - GAN (VQ-GAN)

# Vector Quantize VAE (VQ-VAE)

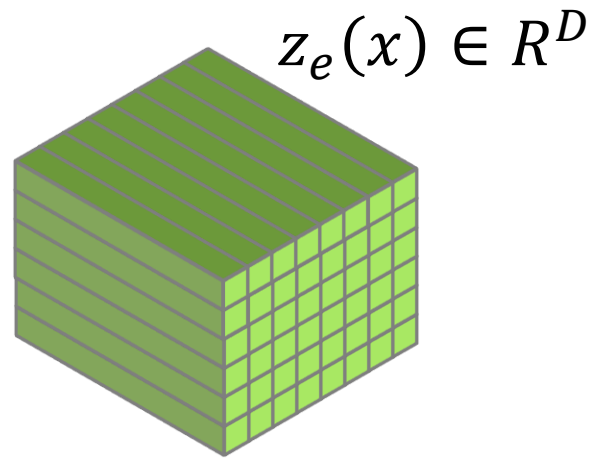


- Discrete objects
- Discrete attributes, e.g., colors, shape, size etc'
- Algorithms and architectures that operate on discrete data

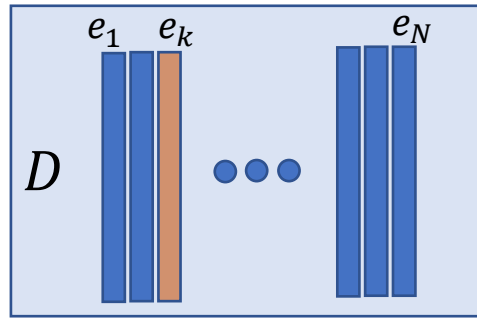
# Vector Quantize VAE (VQ-VAE)



Encoder  
(CNN)



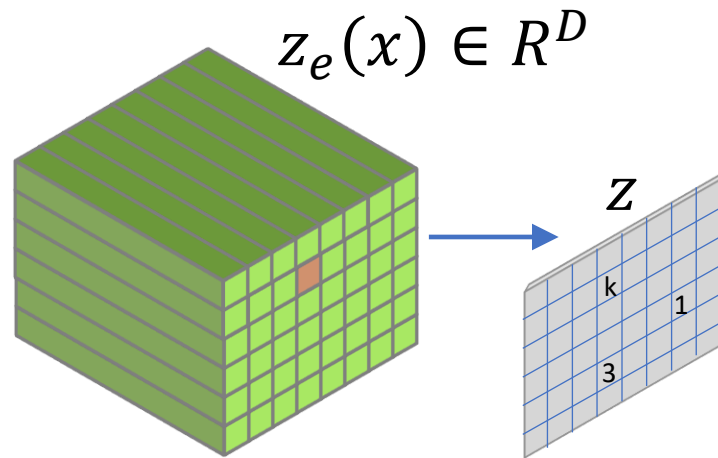
# Vector Quantize VAE (VQ-VAE)



Codebook (N latent codes)



Encoder  
(CNN)

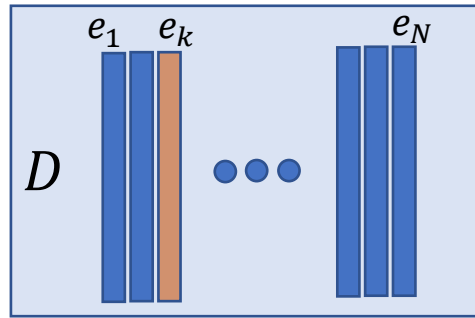


Nearest  
neighbours  
index grid

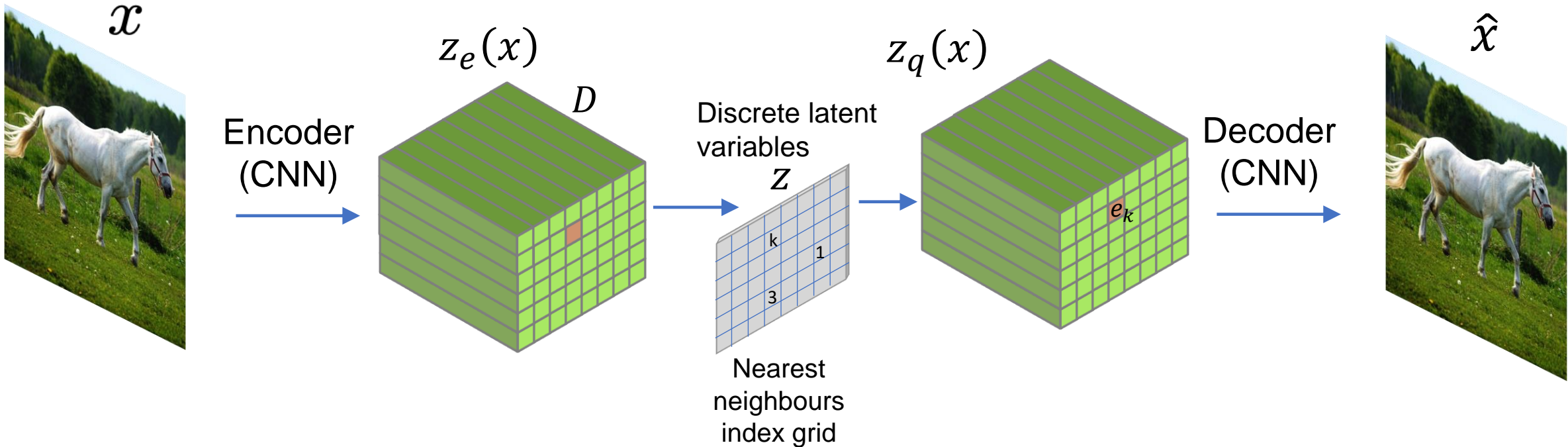
# Vector Quantize VAE (VQ-VAE)

Posterior categorical distribution:

$$q(z = k|x) = \begin{cases} 1 & \text{for } k = \operatorname{argmin}_j \|z_e(x) - e_j\|_2, \\ 0 & \text{otherwise} \end{cases}$$



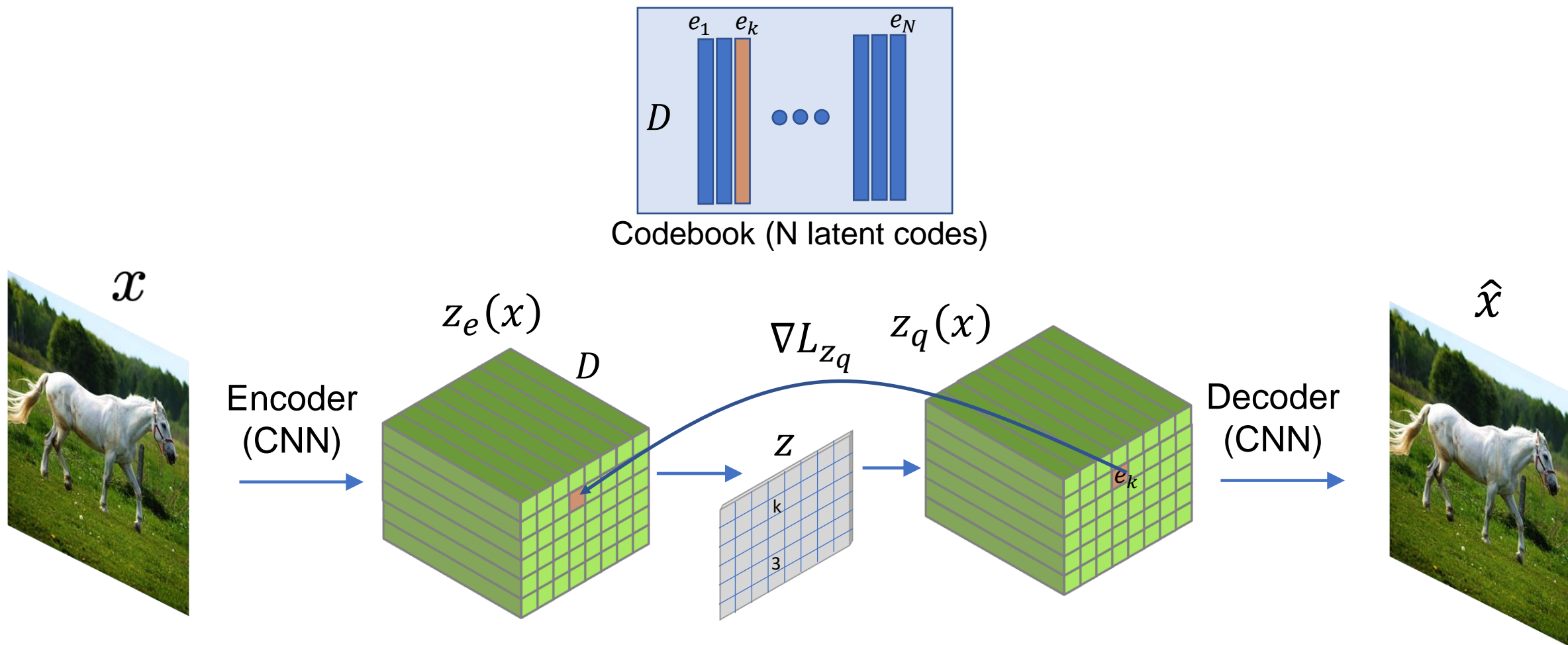
Codebook (N latent codes)



$$z_q(x) = e_k \quad \text{where} \quad k = \operatorname{argmin}_j \|z_e(x) - e_j\|_2^2$$



# Vector Quantize VAE (VQ-VAE)

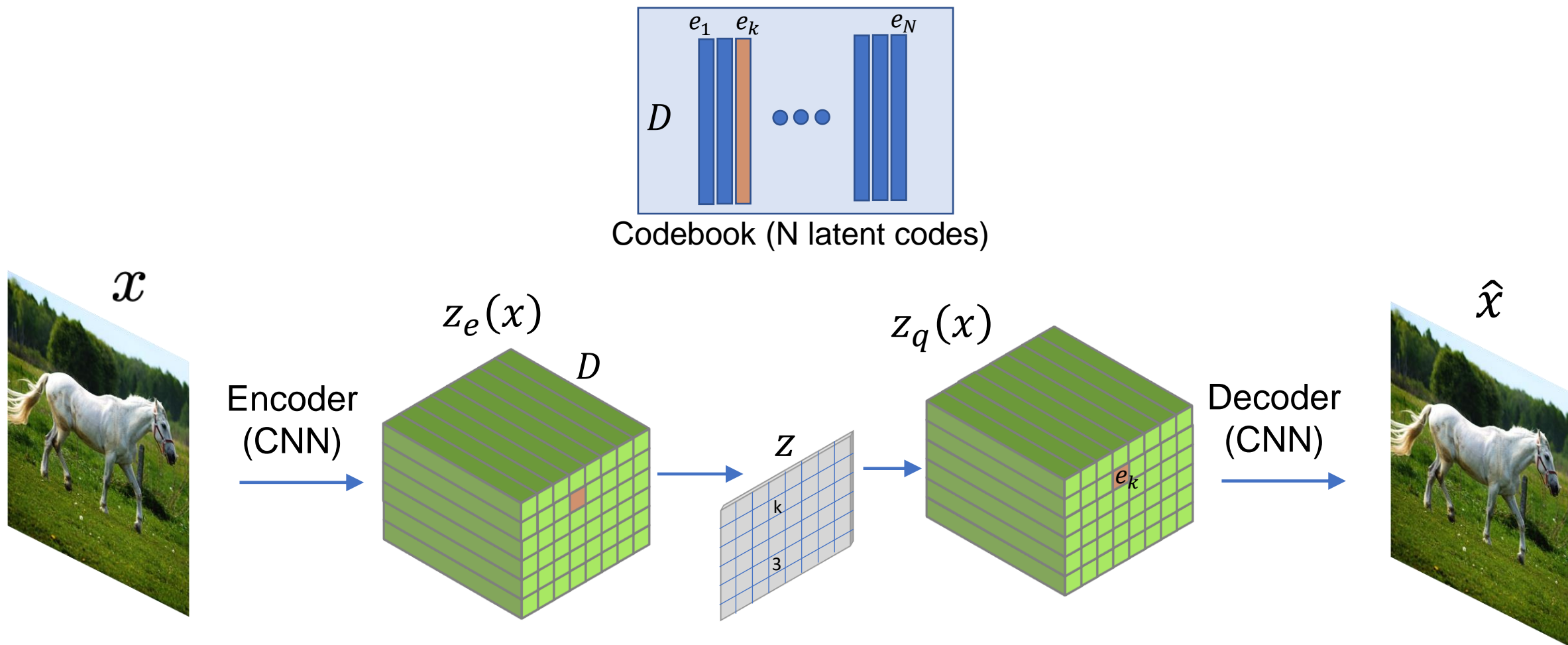


$$L = -\log p(x|z_q(x))$$

**Reconstruction Loss**

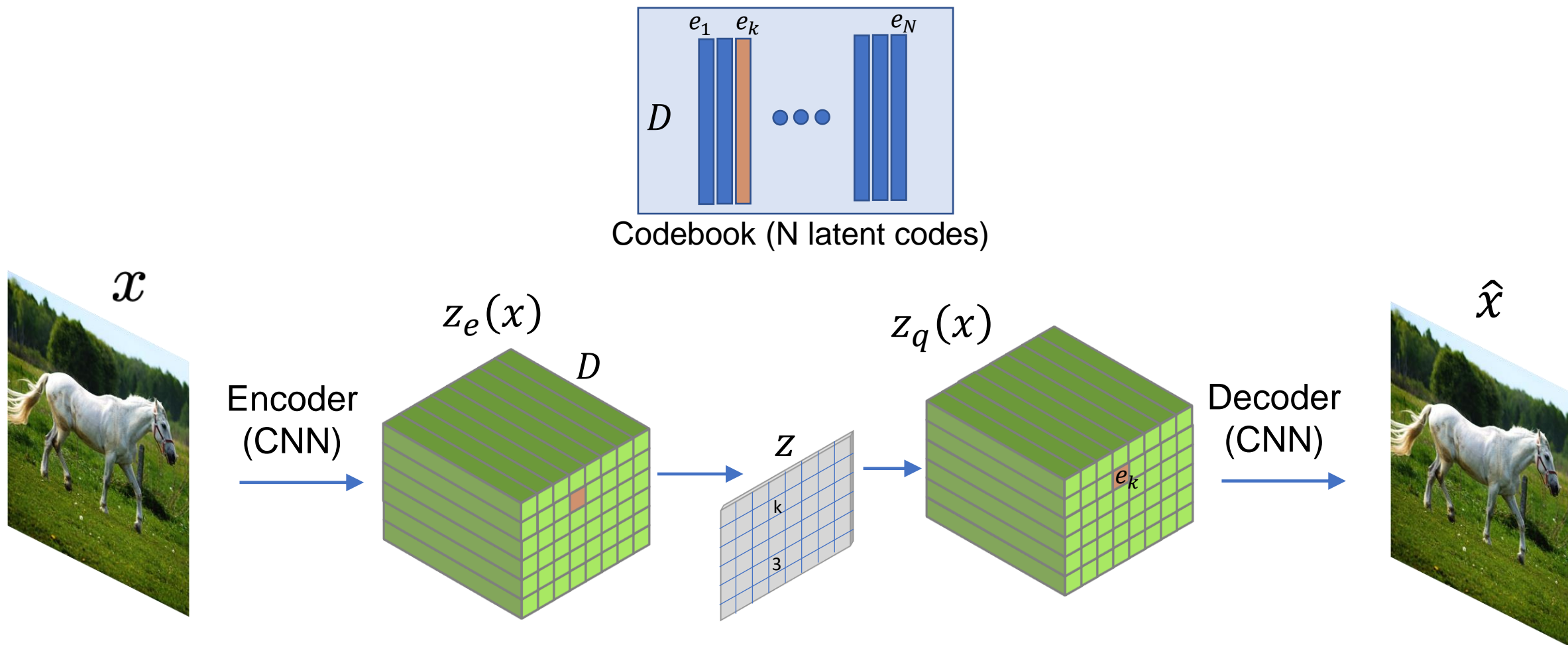


# Vector Quantize VAE (VQ-VAE)



$$L = \log p(x|z_q(x)) + \underbrace{\| \text{sg}[z_e(x)] - e \|_2^2}_{\text{codebook alignment}} + \beta \|z_e(x) - \text{sg}[e]\|_2^2,$$

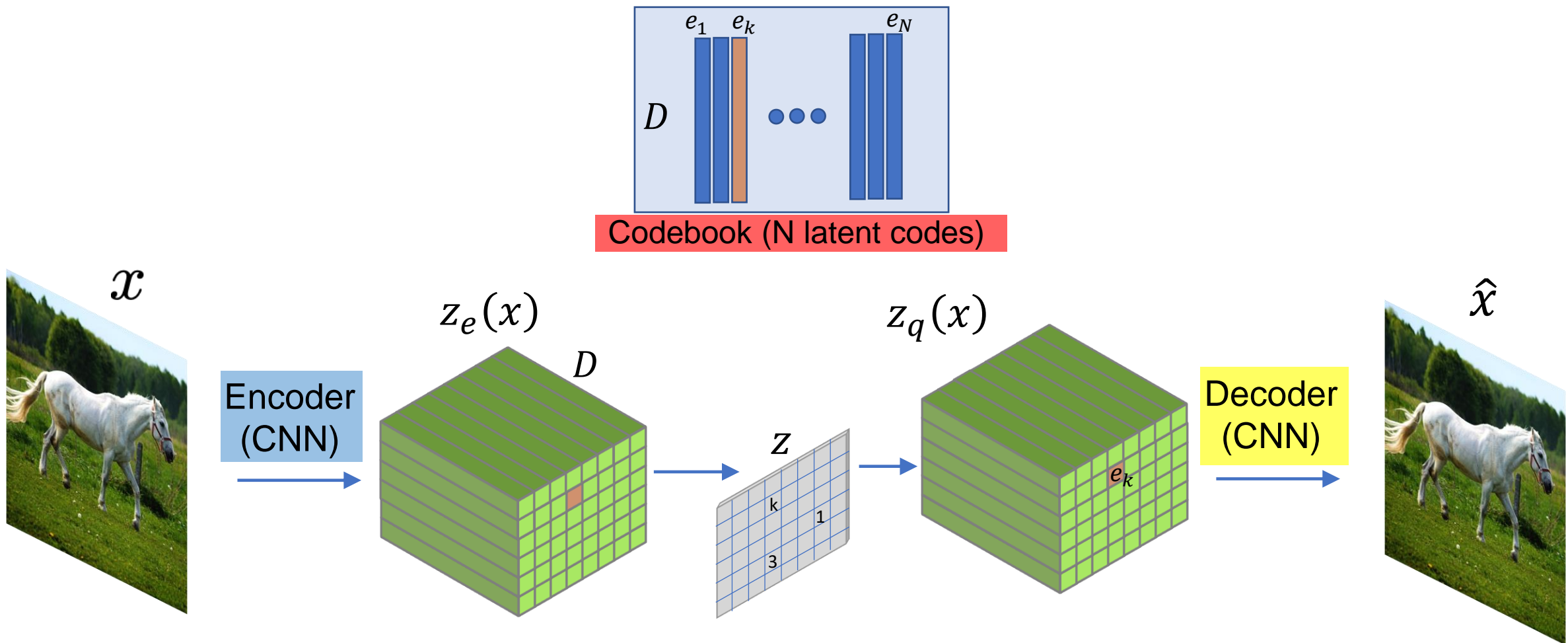
# Vector Quantize VAE (VQ-VAE)



$$L = \log p(x|z_q(x)) + \| \text{sg}[z_e(x)] - e \|_2^2 + \beta \| z_e(x) - \text{sg}[e] \|_2^2,$$

**commitment loss**

# Vector Quantize VAE (VQ-VAE)



$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \text{sg}[e]\|_2^2,$$

# VQ-VAE Reconstruction



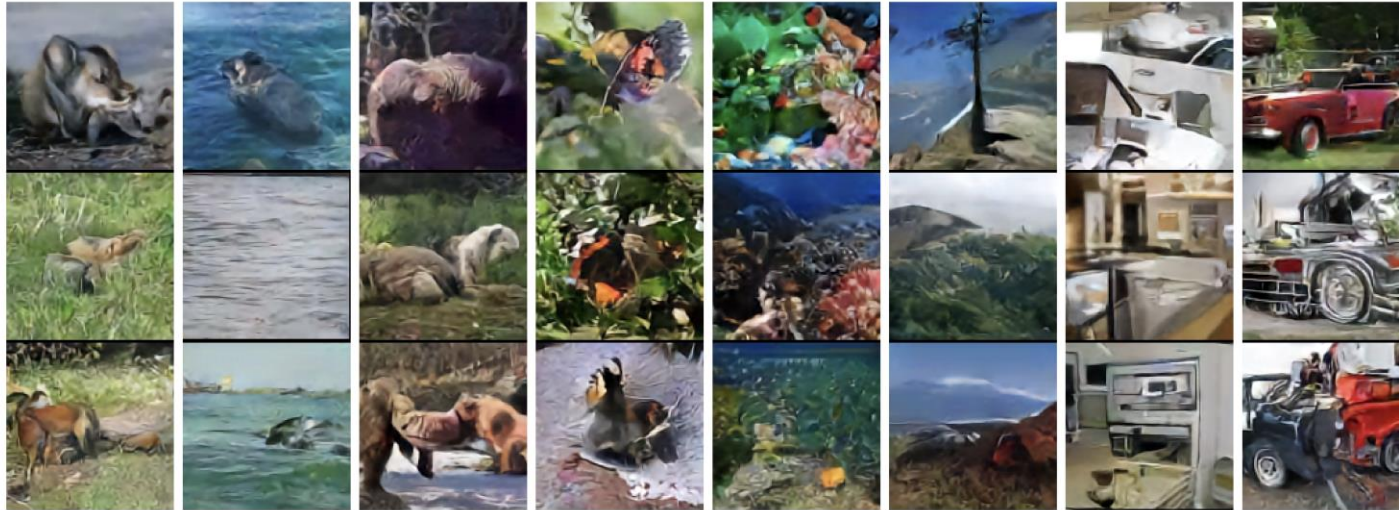
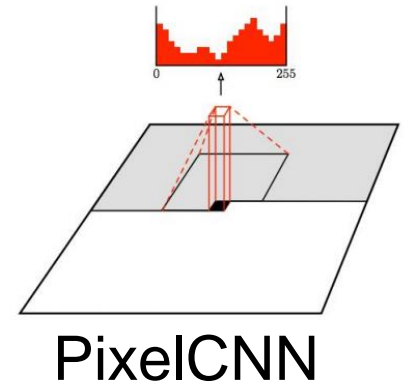
Figure 2: Left: ImageNet 128x128x3 images, right: reconstructions from a VQ-VAE with a 32x32x1 latent space, with  $K=512$ .



# VQ-VAE + PixelCNN

After training, fit an autoregressive distribution over  $z$

$$p_{\theta}(x) = \prod_{i=1}^n p_{\theta}(x_i | x_1, \dots, x_{i-1}) \quad \rightarrow \quad p_{\theta}(z) = \prod_{i=1}^n p_{\theta}(z_i | z_1, \dots, z_{i-1})$$

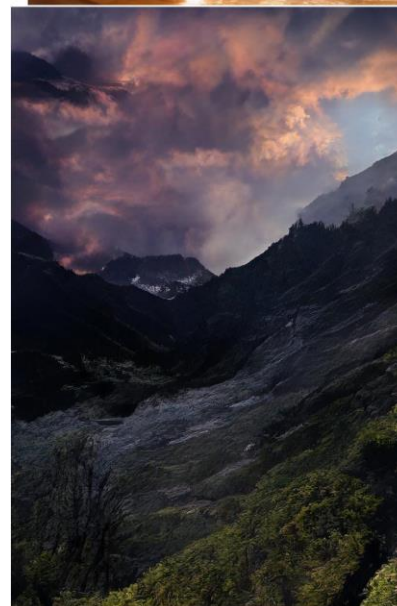


# VQ-VAE + PixelCNN



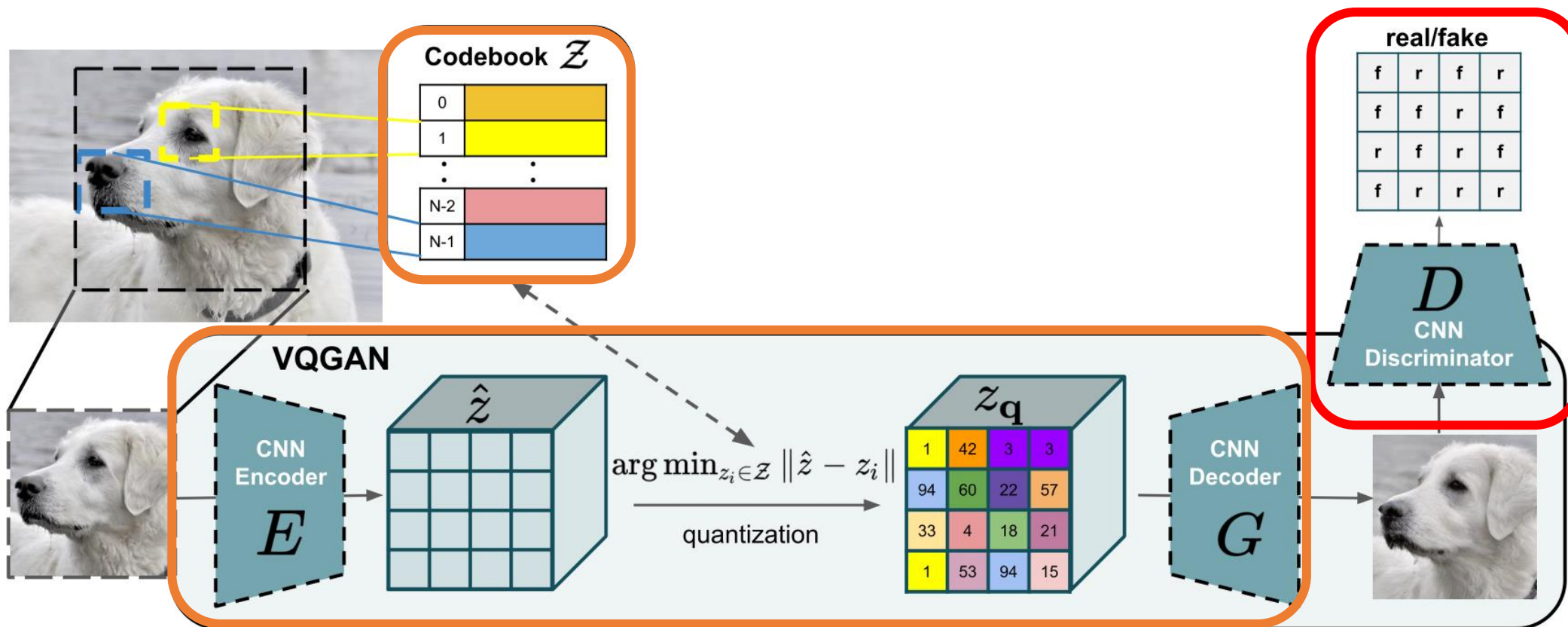


# VQ-VAE + GAN (VQ-GAN)



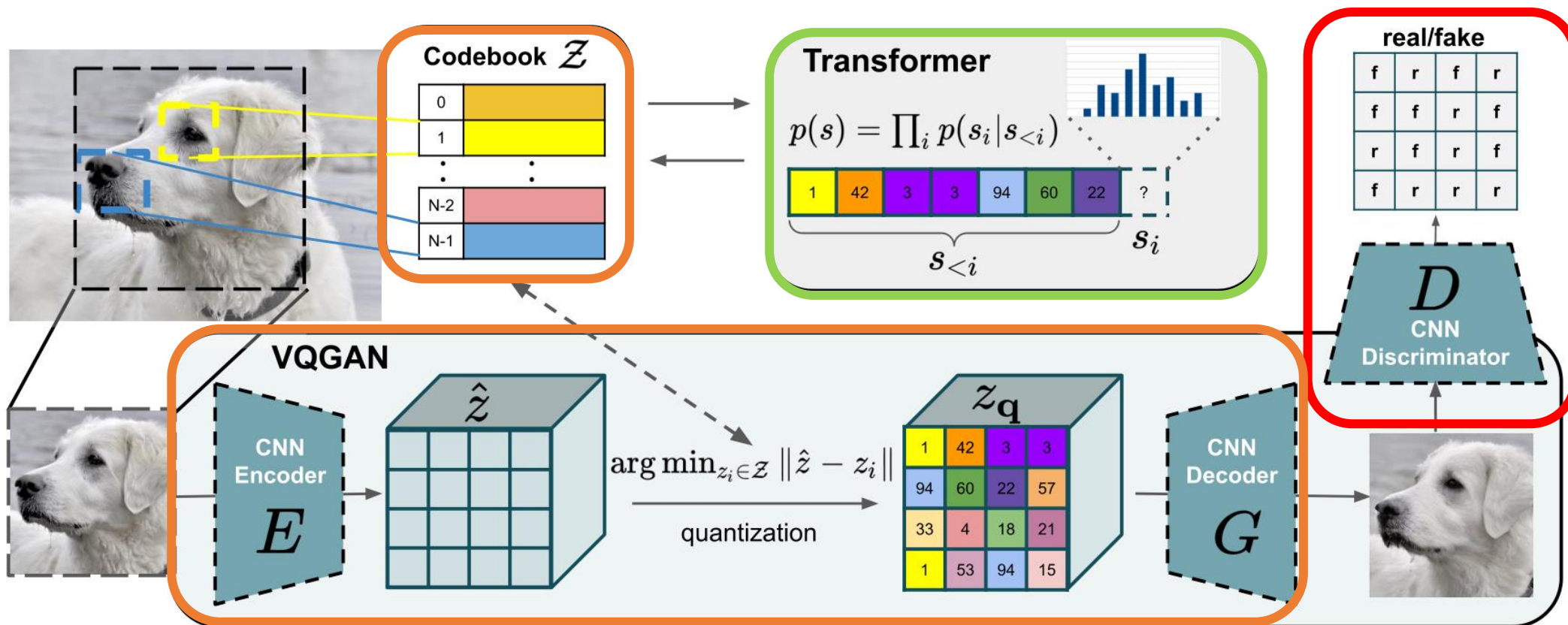


# VQ-GAN



## VQ-VAE + GAN

# VQ-GAN



VQ-VAE + GAN + Transformer

# VQ-GAN: CNN+Transformer

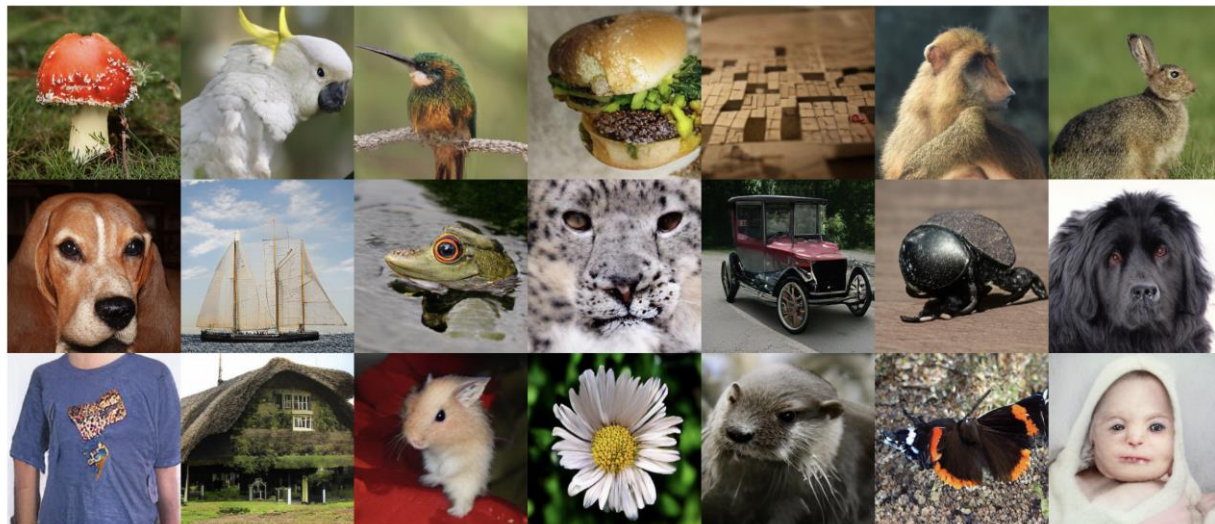


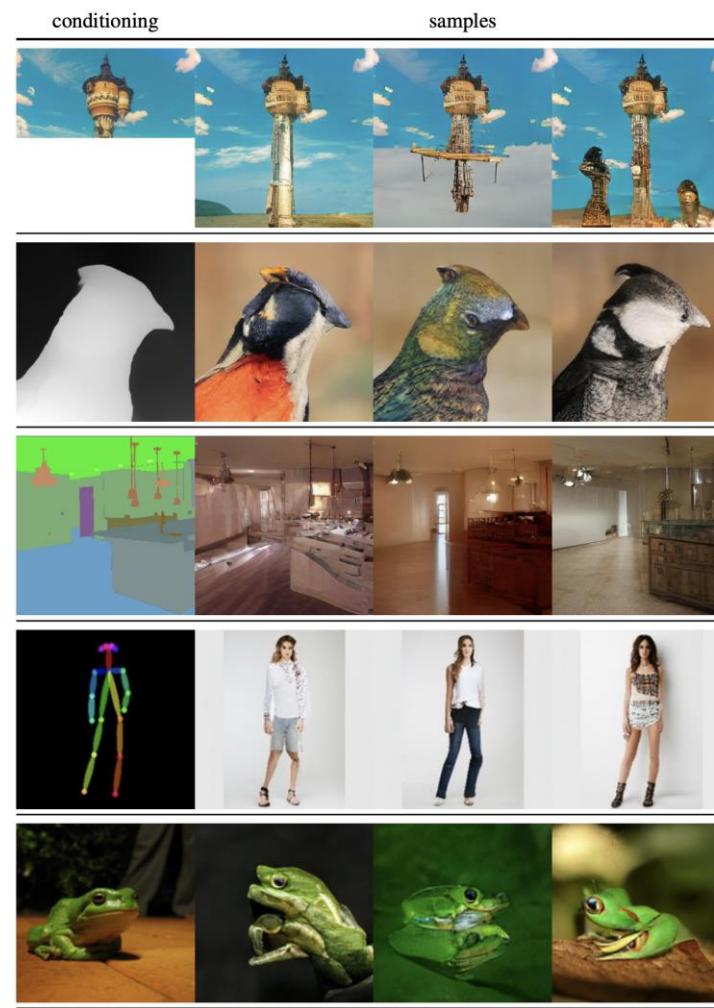
Figure 8. Samples from our class-conditional ImageNet model trained on  $256 \times 256$  images.

TEXT PROMPT an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES



DALL-E

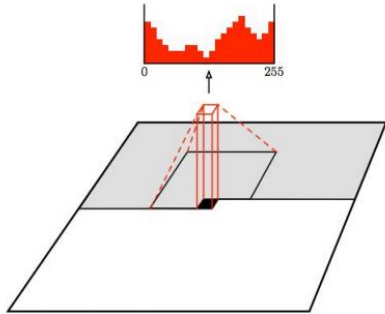


# Learn the probability distribution of natural images

## Generative Models Taxonomy

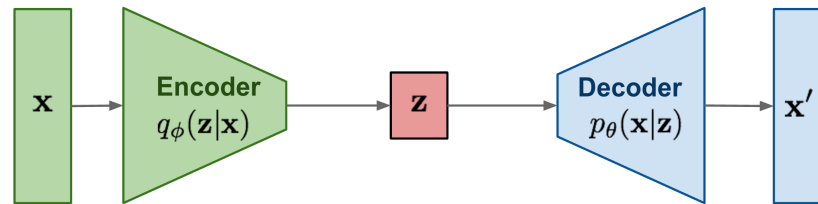
### Explicitly density estimation

#### Autoregressive image generation

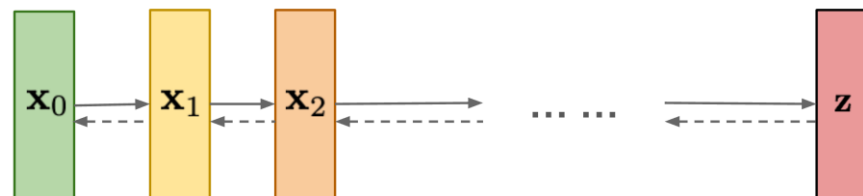


PixelRNN / PixelCNN

#### Variational Autoencoders (VAEs)

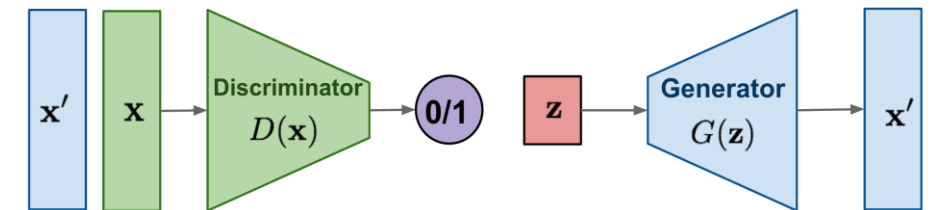


#### Diffusion Models



### Implicit density estimation

#### Generative Adversarial Networks (GANs)

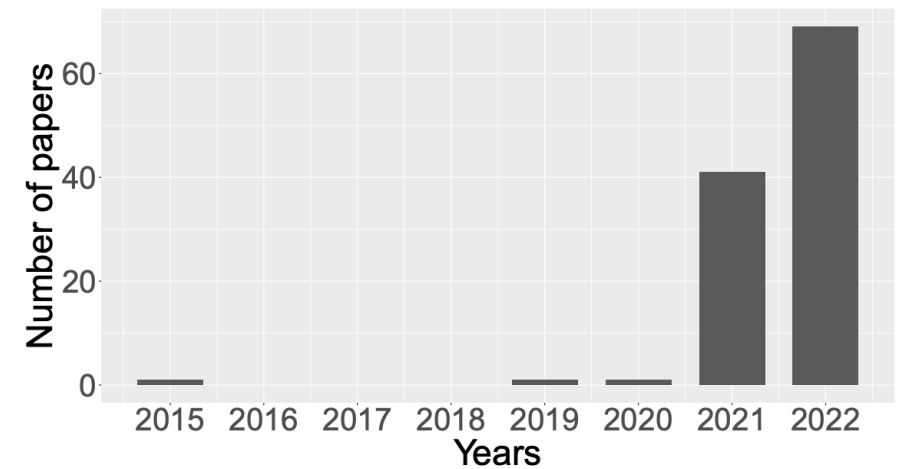




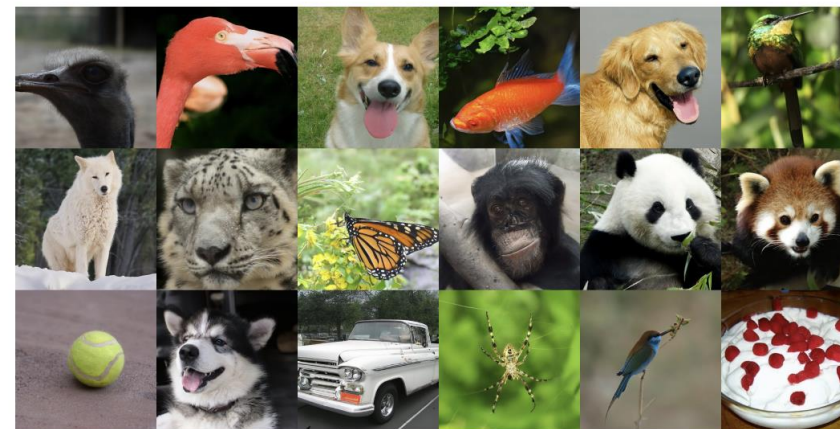
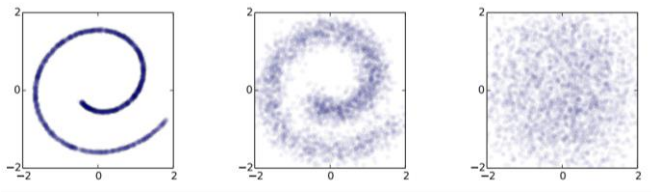
# Motivation



Imagen (Google) / DALLE-2 / StableDiffusion



# Papers



“Unsupervised Learning using Nonequilibrium Thermodynamics”

2015

“Denoising Diffusion Probabilistic Models (DDPM)”

2020

“Improved Denoising Diffusion Probabilistic Models”

2021

“Diffusion Models Beat GANs on Image Synthesis”

2021

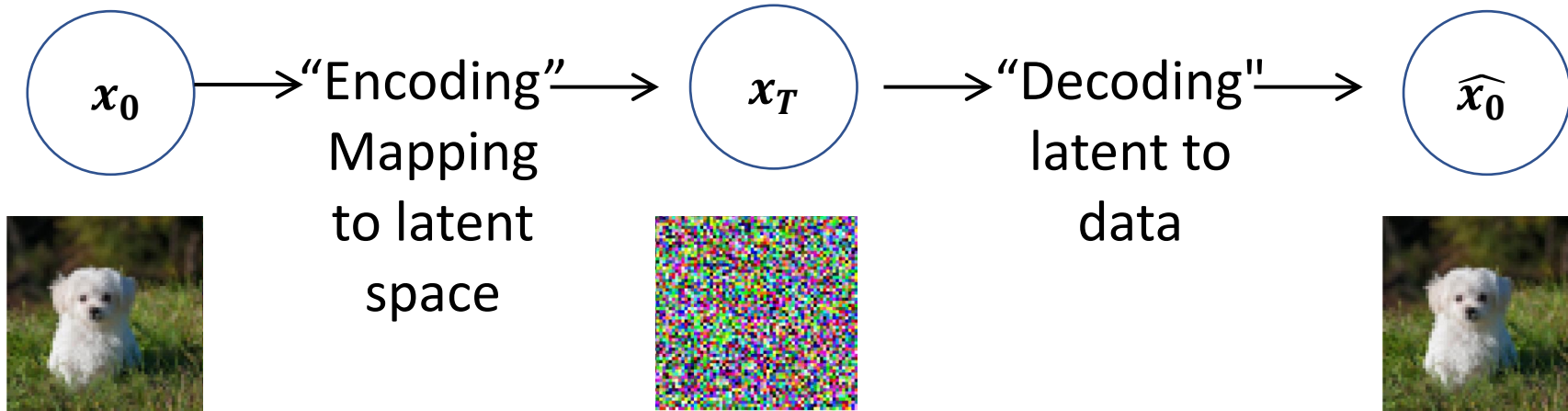


**Berkeley**  
UNIVERSITY OF CALIFORNIA



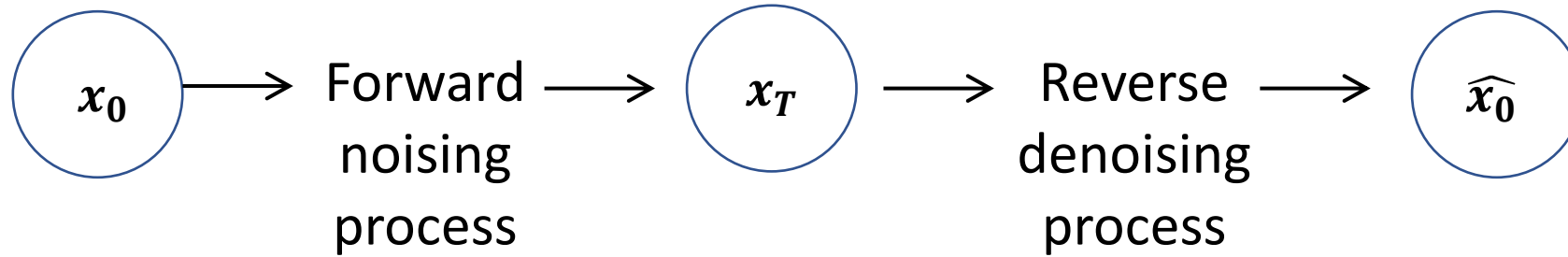
**OpenAI**

# Diffusion Models - Basics

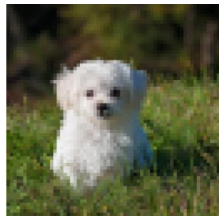




# Diffusion Models - Basics



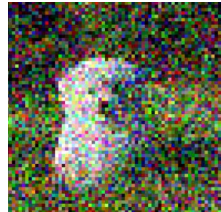
Forward  
noising  
process



$x_0$



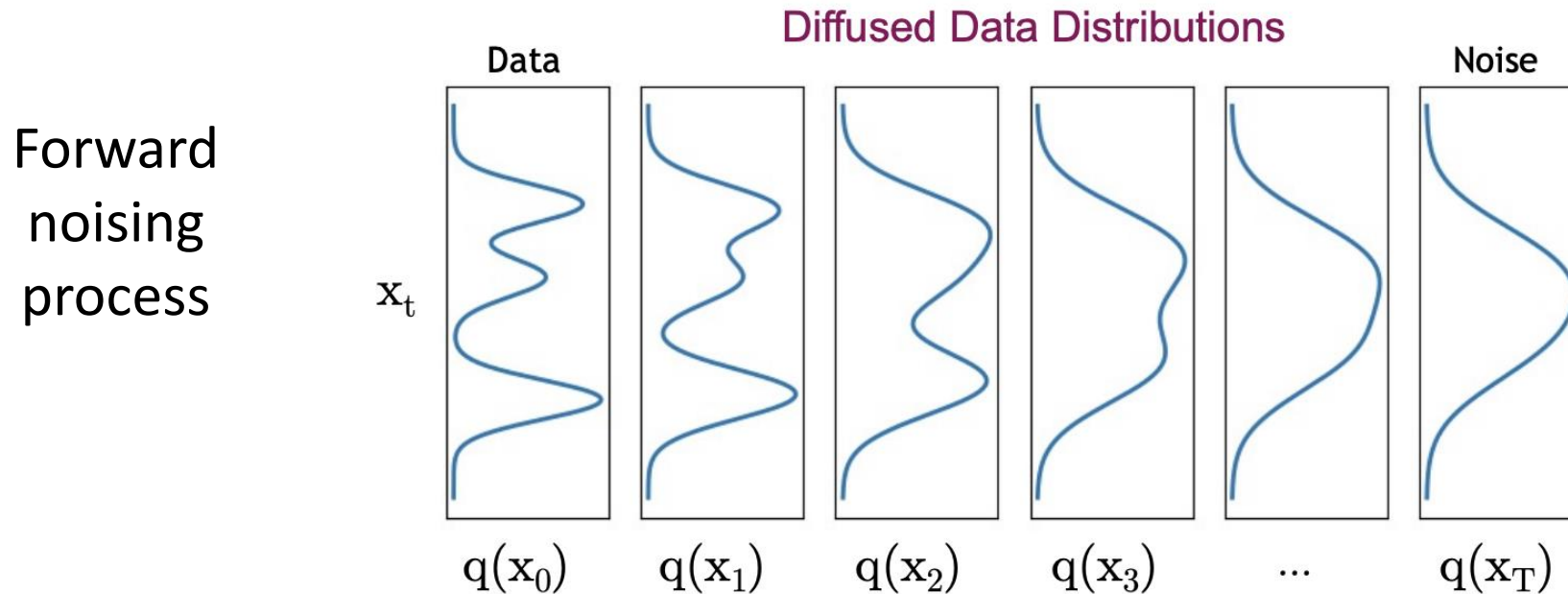
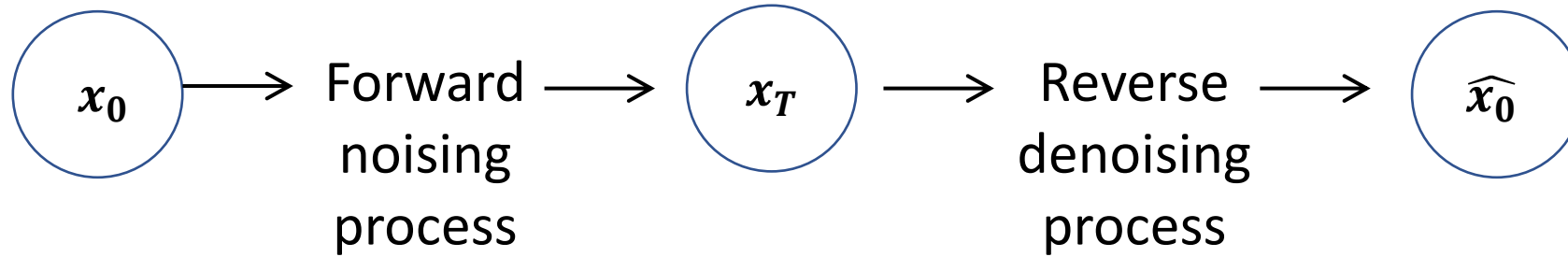
$x_1$



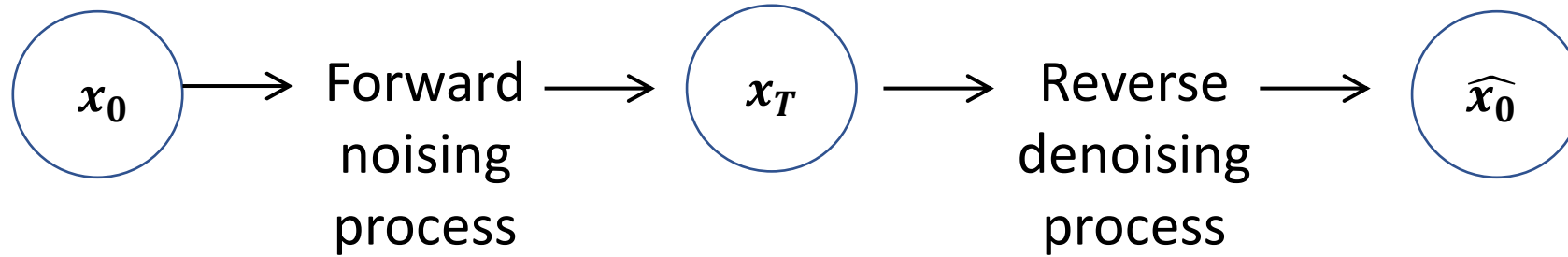
$x_T$

Destructing data by gradually adding noise

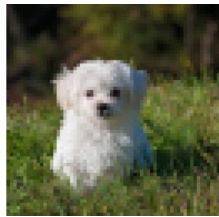
# Diffusion Models - Basics



# Diffusion Models - Basics



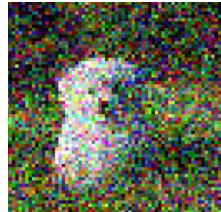
Reverse  
denoising  
process



$x_0$



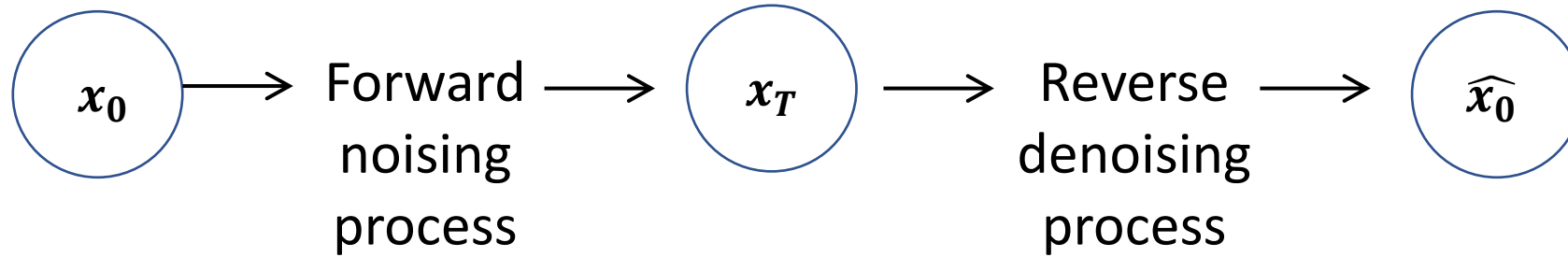
$x_1$



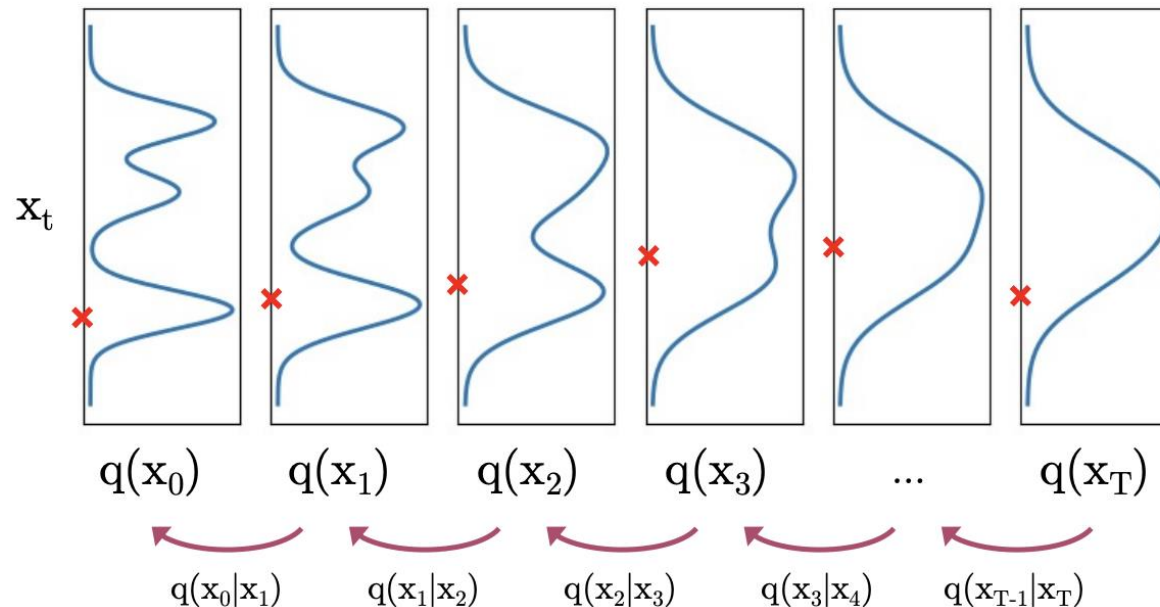
$x_T$

Generating samples by gradually reducing noise

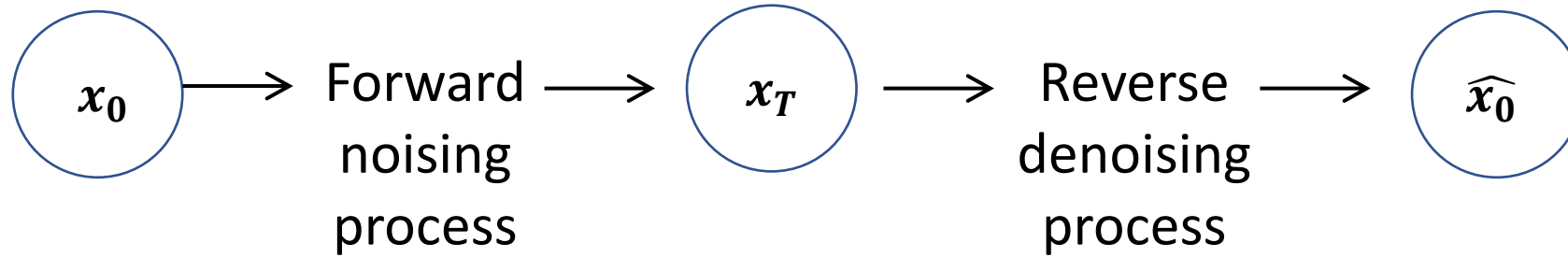
# Diffusion Models - Basics



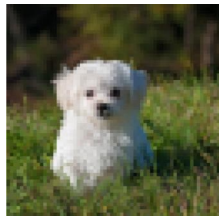
Diffused Data Distributions



# Diffusion Models - Basics



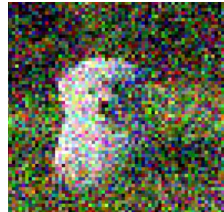
Forward  
noising  
process



$x_0$



$x_1$



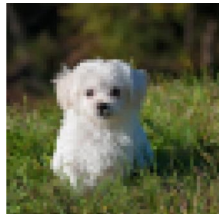
**EASY! (and fixed)**



$x_T$

Destructing data  
by gradually  
adding noise

Reverse  
denoising  
process



$x_0$



$x_1$



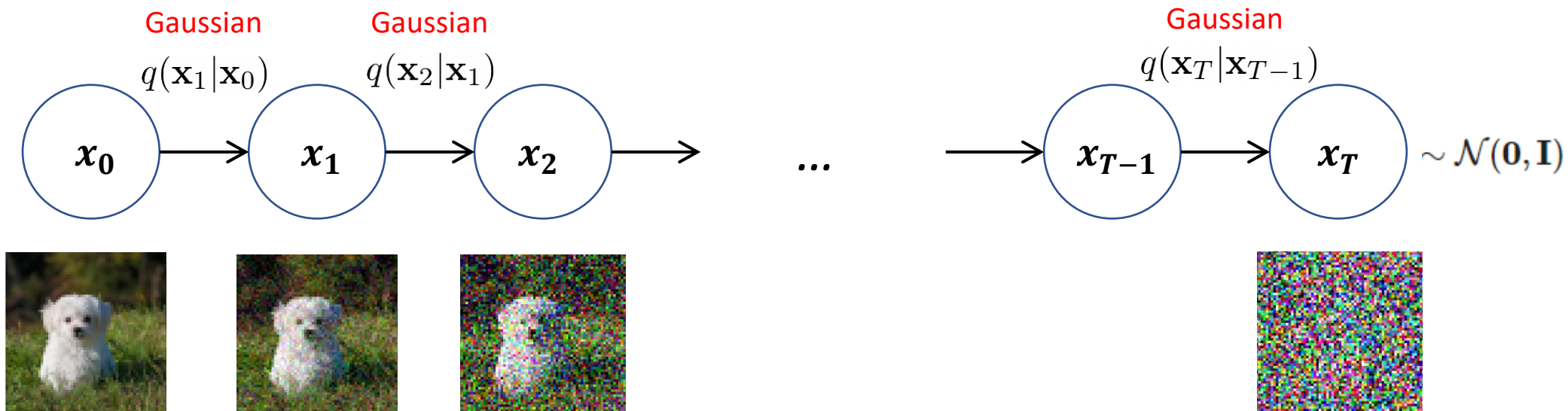
**?? NEED A PRIOR  
ABOUT THE  
WORLD**



$x_T$

Generating  
samples by  
gradually  
reducing noise

# Forward diffusion process



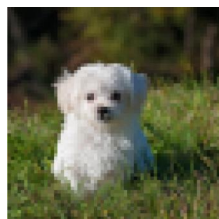
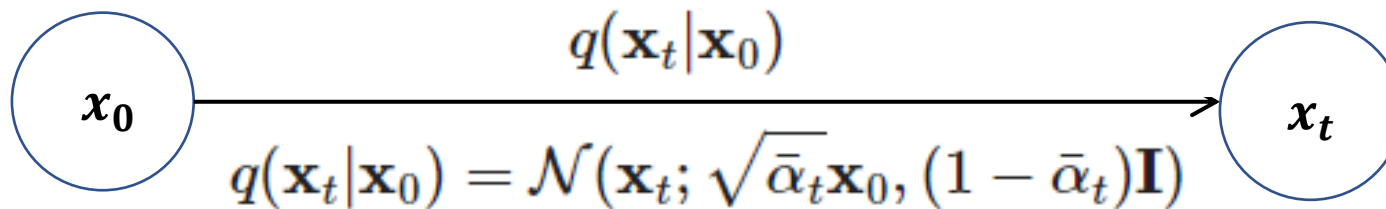
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad \beta_t \in (0,1)$$

noise schedule

(how much noise to add on step  $t$ )

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1})$$

# Forward diffusion process: $x_t$ in one step



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

Noise scheduling:

Set  $\{\beta_t\}$  s. t.,  $\bar{\alpha}_T \rightarrow 0$  and  $q(x_T|x_0) \approx \mathcal{N}(x_T; 0, \mathbf{I})$

$$\beta_1 < \beta_2 \dots < \beta_T$$

Define  $\alpha_t = 1 - \beta_t$ :

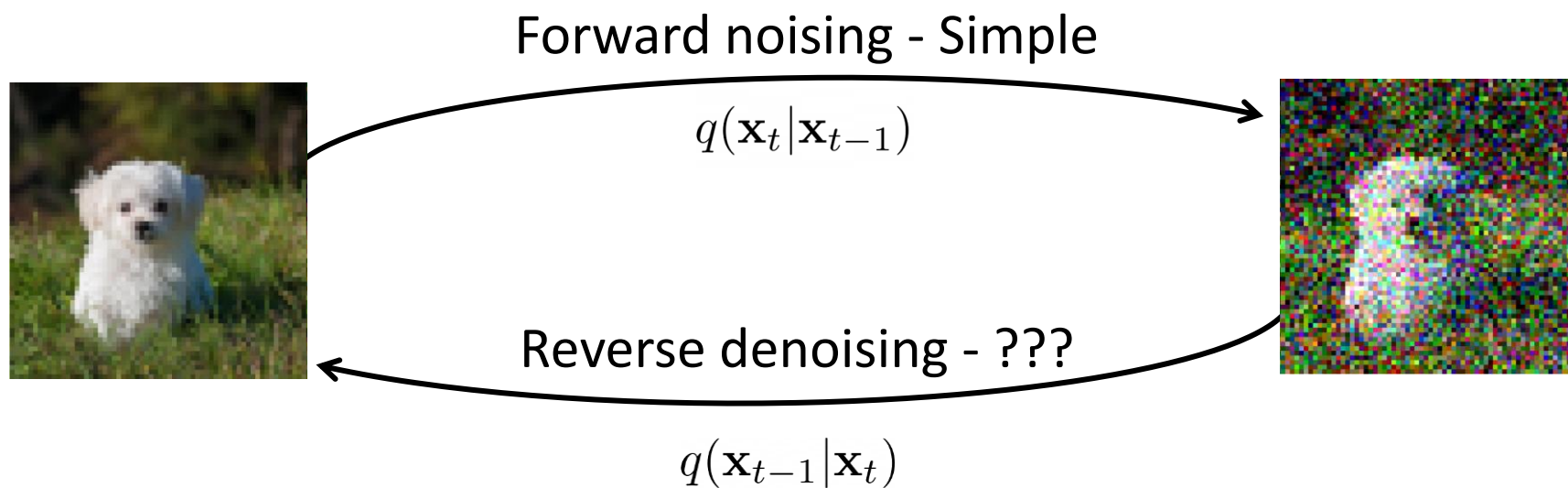
Reparameterization trick:

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1 - \alpha_t}\mathbf{z}_{t-1} & \mathbf{z}_{t-1} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \underbrace{\sqrt{\alpha_t(1 - \alpha_{t-1})}\mathbf{z}_{t-2} + \sqrt{1 - \alpha_t}\mathbf{z}_{t-1}}_{\sqrt{1 - \alpha_t\alpha_{t-1}}\bar{\mathbf{z}}_{t-2}} \\ &= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\mathbf{z} \end{aligned}$$

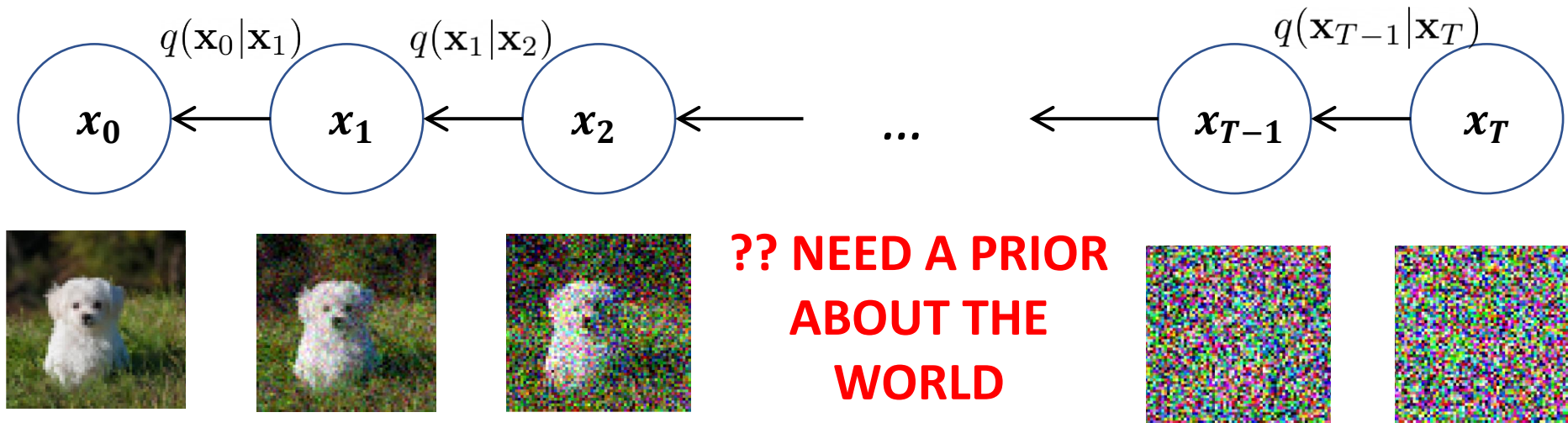
$$\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$$



# Reverse diffusion process



# Reverse diffusion process



Generation: sample  $x_T$  and iteratively sample:

$$x_{t-1} \sim q(x_{t-1}|x_t)$$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

**Intractable**

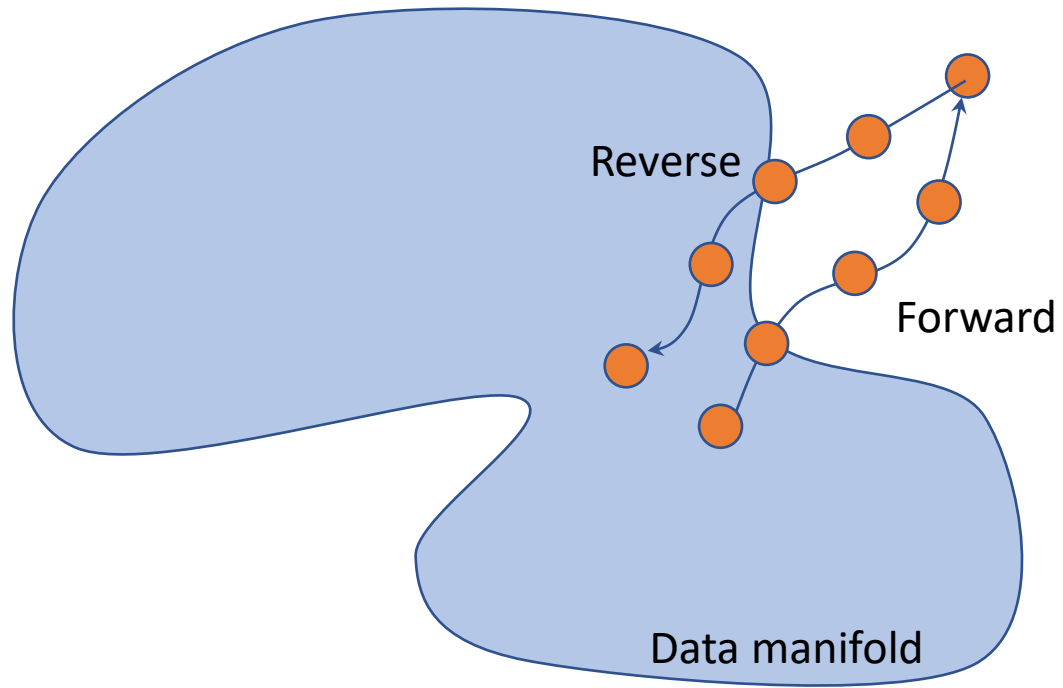
Can we approximate  $q(x_{t-1}|x_t)$ ?

**Yes!** if  $\beta_t \ll 1$  – small variance in each forward step

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 \mathbf{I})$$

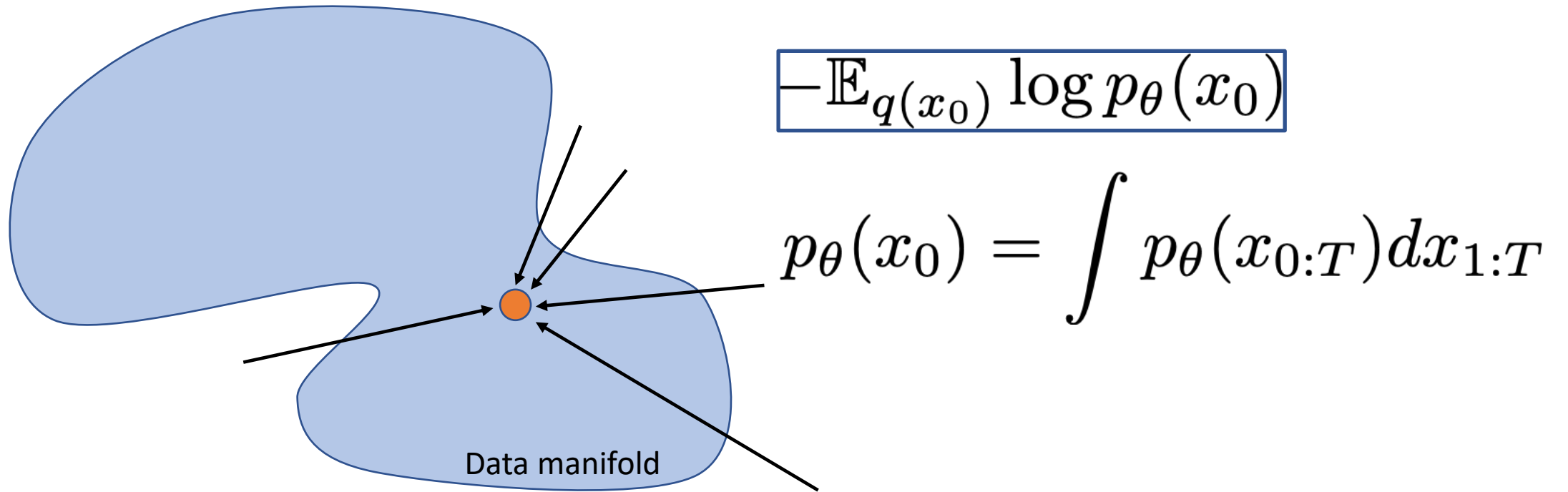
**Train a network to predict the reverse distribution**

# Forward and Reverse Diffusion



# Objective?

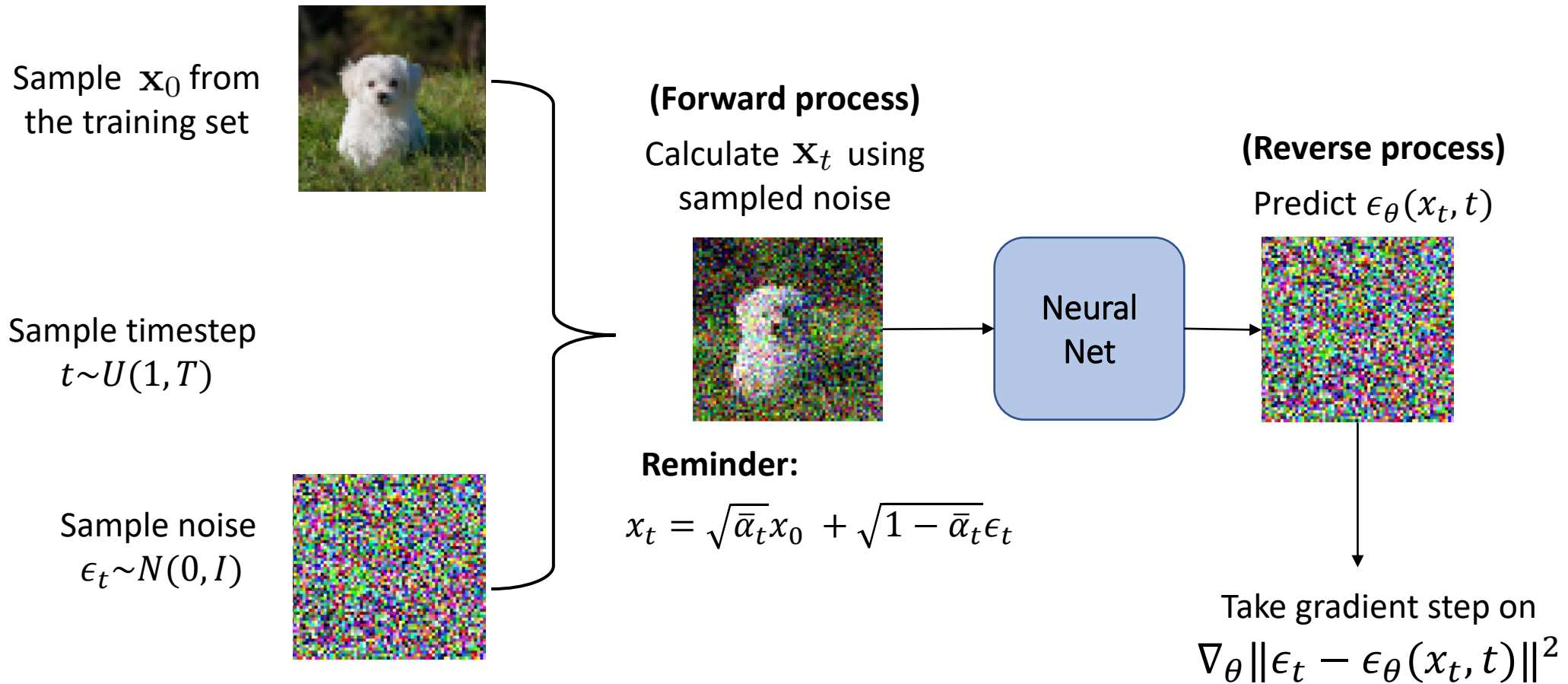
Ideally: maximize the likelihood of the data after reverse process, i.e., minimize:



$\log p_\theta(x_0) \geq \text{variational lower bound}$


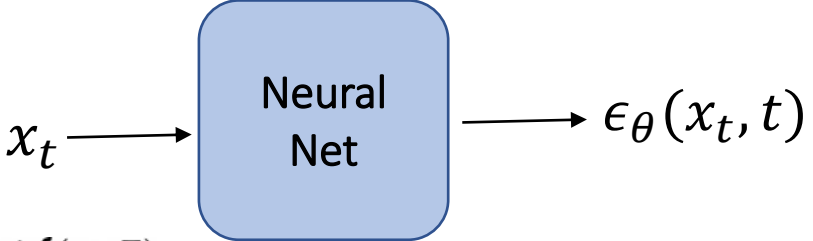
$-\log p_\theta(x_0) \leq \text{variational upper bound}$

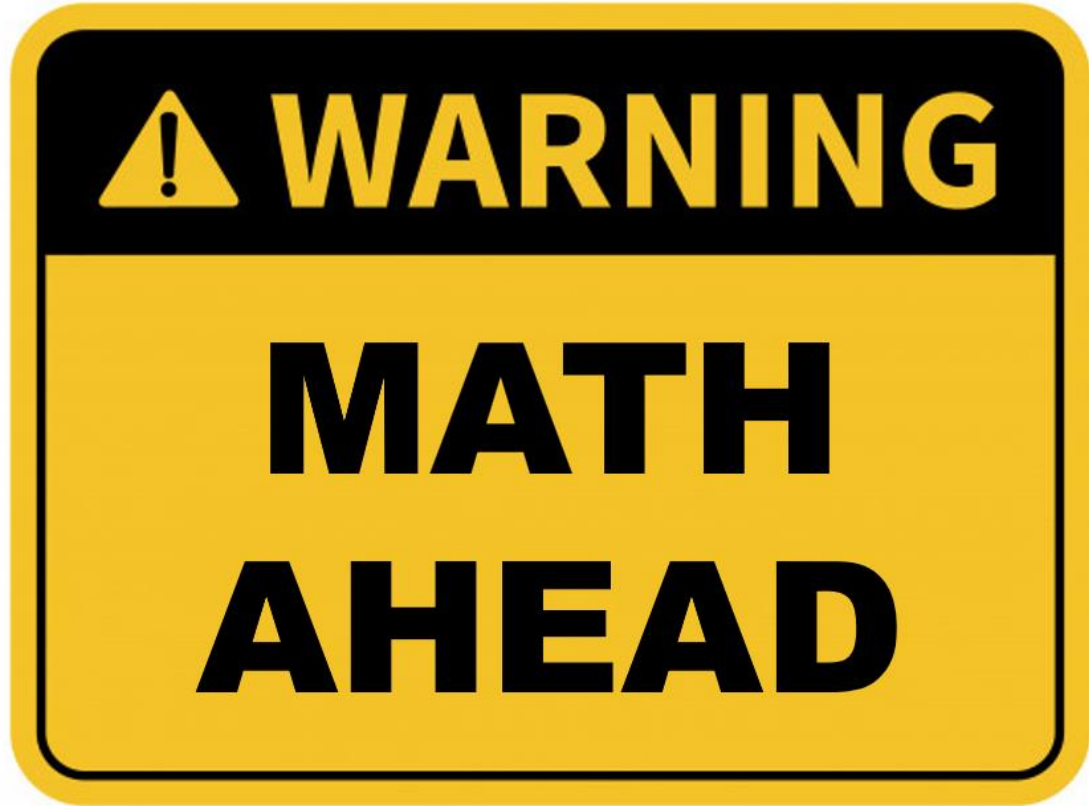
# Training



# Sampling Algorithm

Reminder:  $p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \sigma_t^2 \mathbf{I})$

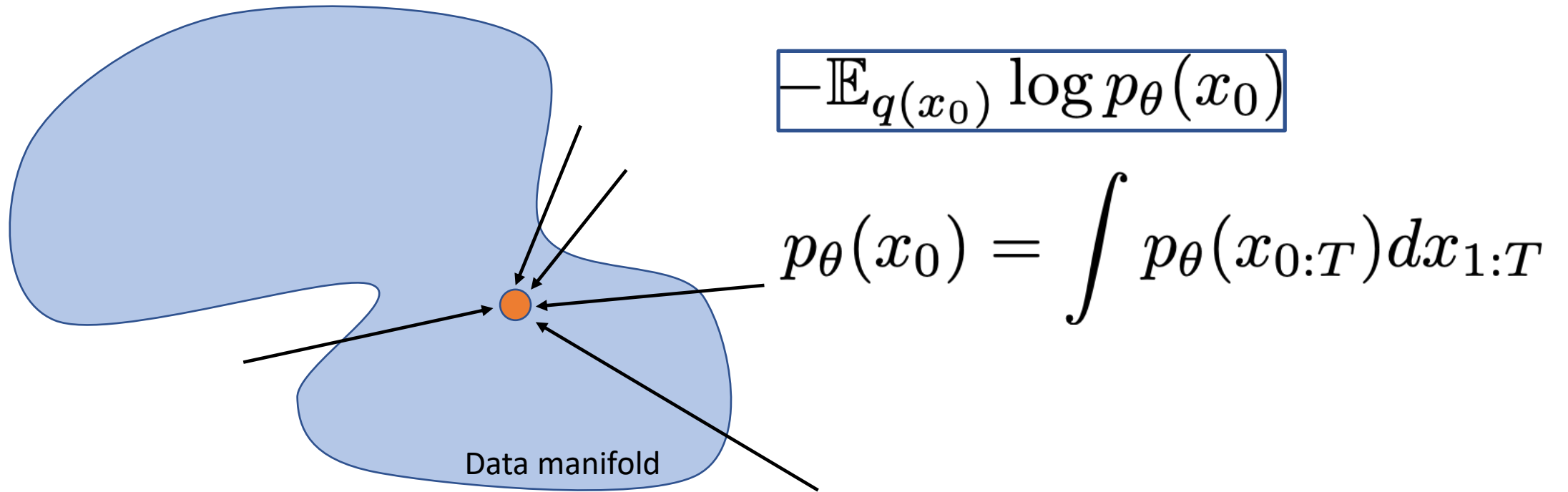
1. Sample  $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  
2. For  $t = T, \dots, 1$ :
  - “Predict noise” in  $x_t$  
  - Sample variance noise  $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - Sample from reverse distribution:
$$x_{t-1} = \mu_{\theta}(\epsilon_{\theta}, x_t, t) + \sigma_t^2 z$$
3. Return





# Objective?

Ideally: maximize the likelihood of the data after reverse process, i.e., minimize:



$\log p_\theta(x_0) \geq \text{variational lower bound}$

$-\log p_\theta(x_0) \leq \text{variational upper bound}$

# ELBO Objective

$x$  – observation

$z$  – latent

$$\log p_{\theta}(x) \geq \underbrace{\mathbb{E}_{q(z|x)} \log p_{\theta}(x|z)}_{\text{Reconstruction term}} - \underbrace{D_{KL}(q(z|x) || p_{\theta}(z))}_{\text{Latent prior term}}$$

$x_0$  – observation

$x_{1:T}$  – latent

# Objective

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

$$-\mathbb{E}_{q(x_0)} \log p_\theta(\mathbf{x}_0) \leq \mathbb{E}_{q(x_0)q(x_{1:T}|x_0)} \left[ \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right]$$

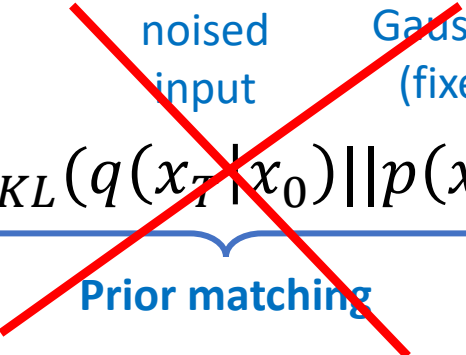
$$\begin{aligned} &= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} \right] \\ &= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\ &= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \left( \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)} \right) + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\ &= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\ &= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T | \mathbf{x}_0)}{q(\mathbf{x}_1 | \mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\ &= \mathbb{E}_q \left[ \log \frac{q(\mathbf{x}_T | \mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \right] \end{aligned}$$

# Objective

$$-\mathbb{E}_{q(x_0)} \log p_\theta(x_0) \leq \mathbb{E}_{q(x_0)q(x_{1:T}|x_0)} \left[ \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right] = \mathbb{E}_q \left[ \log p_\theta(x_T) + \sum_{t \leq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right] := L_{VLB}$$

$$L_{VLB} = \mathbb{E}_q \left[ \underbrace{-D_{KL}(q(x_T|x_0) || p(x_T))}_{\text{Prior matching}} - \sum_{t>1} D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1) \right]$$

Final noised input      Gaussian (fixed)





# Objective

$$-\mathbb{E}_{q(x_0)} \log p_\theta(x_0) \leq \mathbb{E}_{q(x_0)q(x_{1:T}|x_0)} \left[ \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right] = \mathbb{E}_q \left[ \log p_\theta(x_T) + \sum \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right] := L_{VLB}$$

$$L_{VLB} = \mathbb{E}_q \left[ \underbrace{-D_{KL}(q(x_T|x_0) || p(x_T))}_{\text{Prior matching}} - \sum_{t>1} \underbrace{D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t))}_{\text{Denoising matching}} + \underbrace{\log p_\theta(x_0|x_1)}_{\text{Reconstruction term}} \right]$$

Final noised input (crossed out), Gaussian (fixed) (crossed out), Gaussian (highlighted), Closed form!



$x_t$

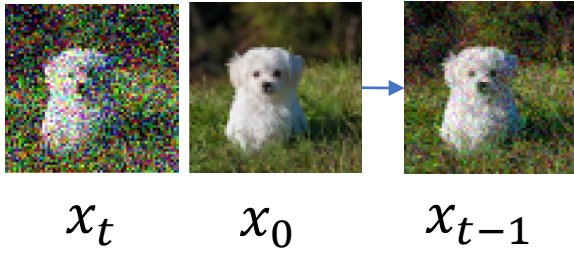
$x_0$

$x_{t-1}$

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

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

$$L_{VLB} = -\mathbb{E}_{q,t}[\mathbf{D}_{KL}(q(x_{t-1}|x_t, x_0)||p_{\theta}(x_{t-1}|x_t))]$$



$$q(x_{t-1}|x_t, x_0) := \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \mathbf{I})$$

$$p_{\theta}(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \sigma_t^2 \mathbf{I})$$

Learnable Fixed

Ground truth vs.  
predicted  
reverse process

Regression:

$$\begin{aligned} L_{t-1} &= D_{KL}(q(x_{t-1}|x_t, x_0)||p_{\theta}(x_{t-1}|x_t)) \\ &= \mathbb{E}_q \left[ \frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(x_t, x_0) - \mu_{\theta}(x_t, t)\|^2 \right] + C \end{aligned}$$

DDPM, Ho et al. NeurIPS 2020:

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right). \quad \begin{aligned} \epsilon &\sim \mathcal{N}(0,1) \\ x_t &= \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon \end{aligned}$$

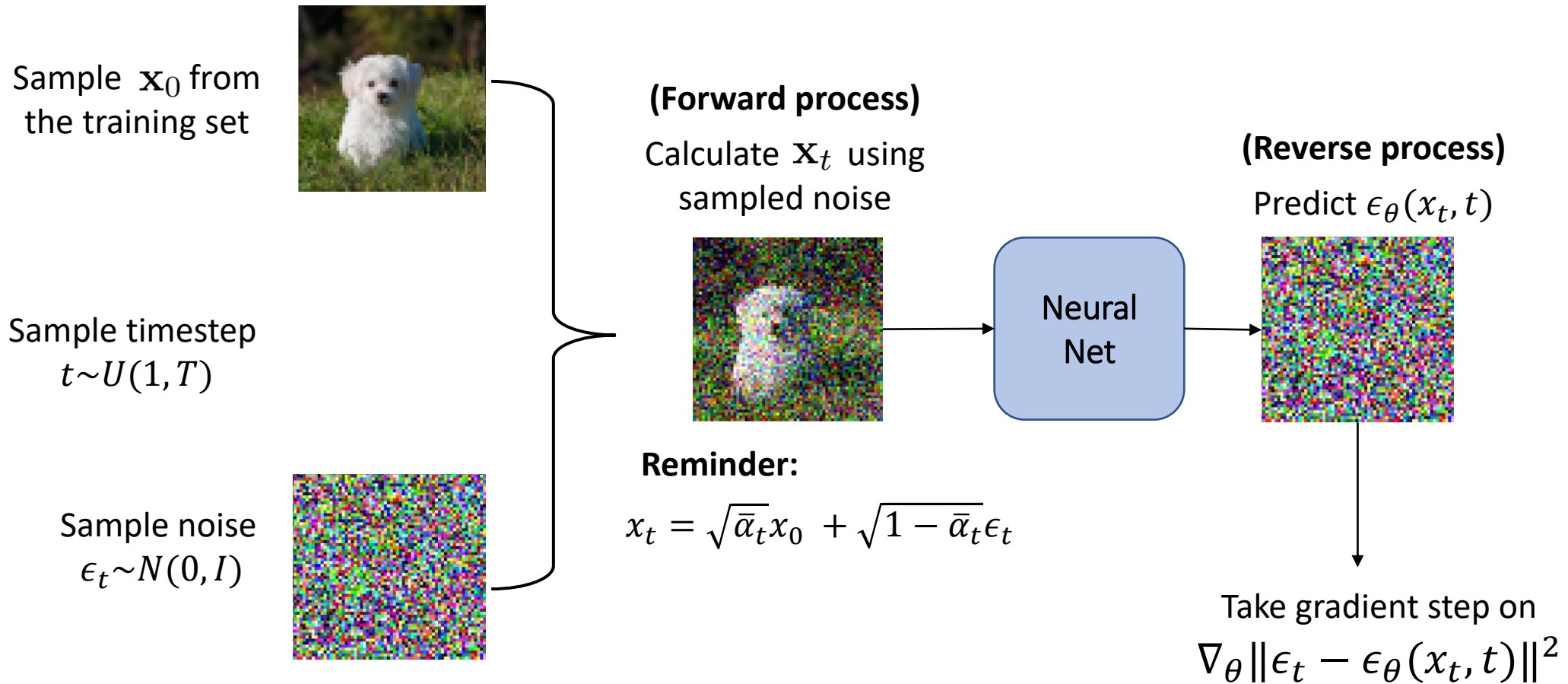
$$\mu_{\theta}(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta} \right)$$

Use a noise prediction network:

$$L_{t-1} = \mathbb{E}_{x_0, \epsilon} [\lambda_t \|\epsilon - \epsilon_{\theta}(x_t, t)\|^2] + C$$

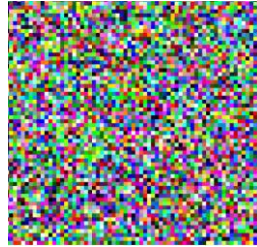
$$L_{simple} = \mathbb{E}_{x_0 \sim q(x_0), \epsilon \sim \mathcal{N}(0, I), t \sim U(1, T)} [\|\epsilon - \epsilon_{\theta}(x_t, t)\|^2]$$

# Training



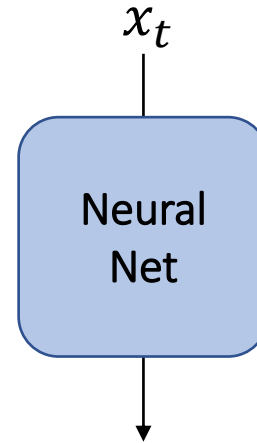
# Sampling Algorithm

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



2. For  $t = T, \dots, 1$ :

- “Predict noise” in current image  $\epsilon_{\theta}(x_t, t)$
- Sample variance noise  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Sample from reverse distribution:



3. Return  $\mathbf{x}_0$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \underbrace{x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(x_t, t)}_{\mu_{\theta}(x_t, t)} \right) + \tilde{\beta}_t \mathbf{z}$$

# Summary

---

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



# Predicted $\hat{x}$

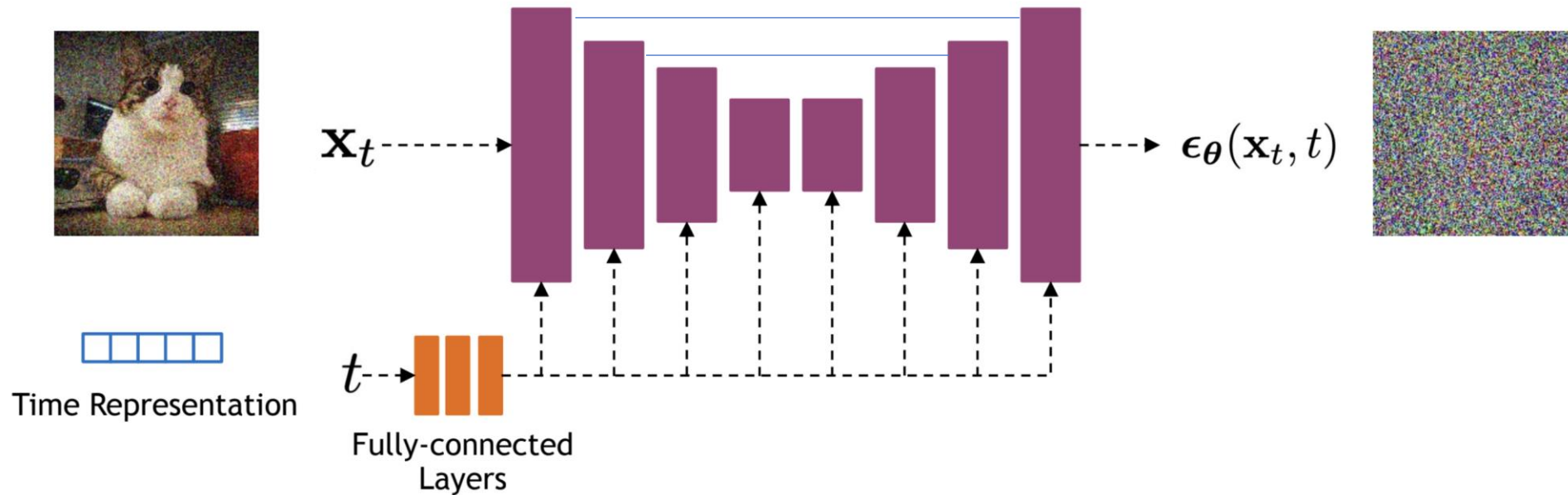
Recall (reparameterization trick):  $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$

$$\hat{x}_0 = \frac{x_t}{\sqrt{\bar{\alpha}_t}} - \frac{\sqrt{1 - \bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}$$



# Implementation

Often use U-Net architectures with ResNet blocks and self-attention layers

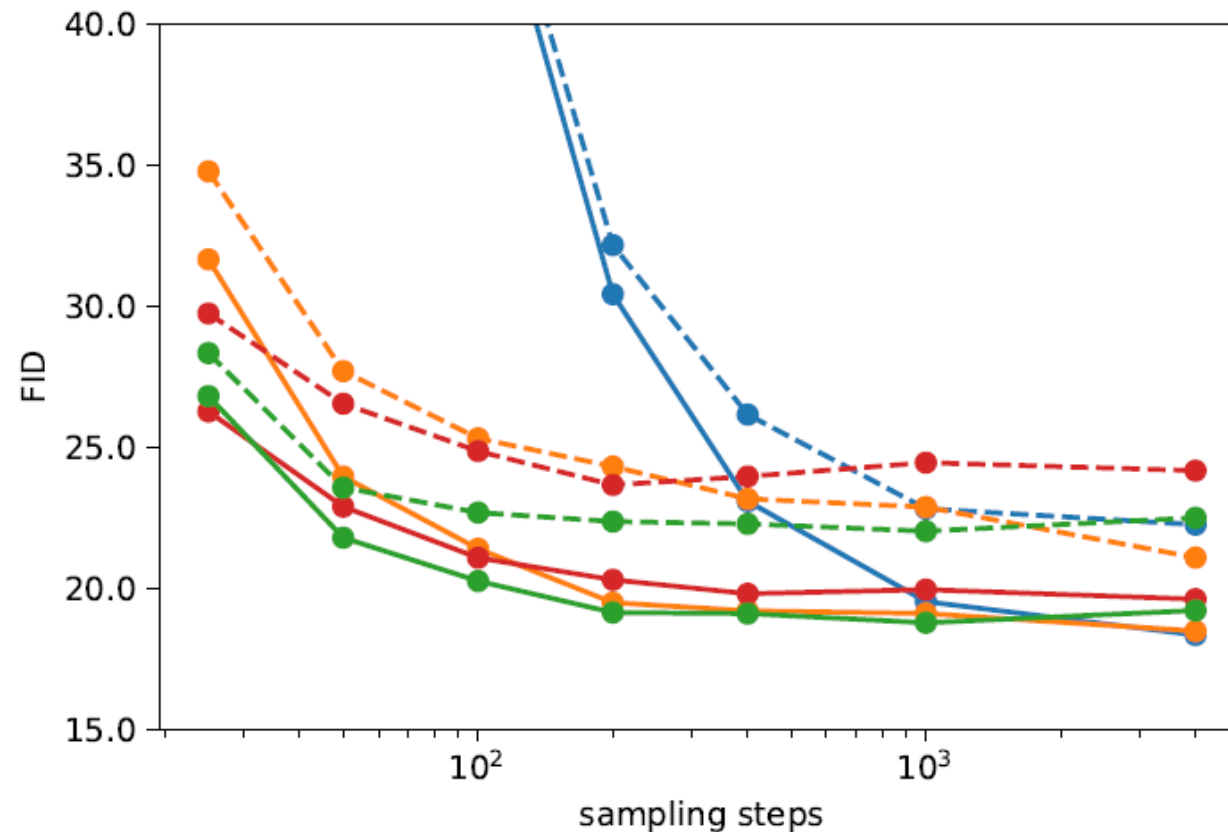


Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dhariwal and Nichol NeurIPS 2021)

# Improving sampling speed

Problem: sampling takes too long (several minutes for one 64\*64 image)

Solution: Just skip some sampling steps!





# Diffusion models beat GANs ?

BigGAN



DDPM improved





# Diffusion models beat GANs ?

BigGAN



DDPM improved





# Conditional Generation

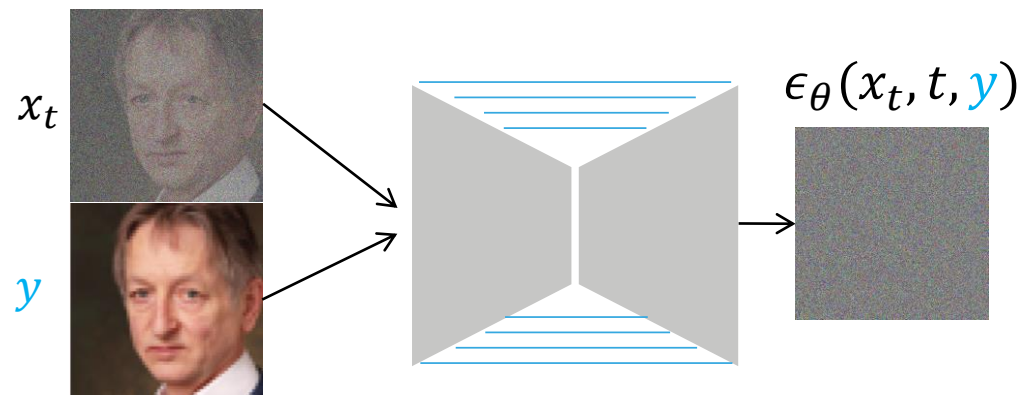
- Class label
- Text
- Another image
- Depth map

$$p_{\theta}(x_0 | \mathbf{y})$$

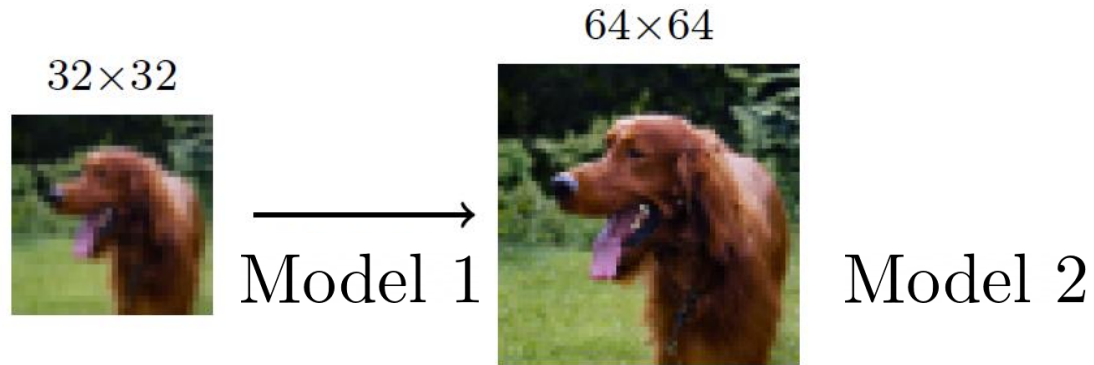
Let's just input the conditional signal:

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

**Image super-resolution:**  $\mathbf{y}$  is a low-resolution image,  $x$  is the corresponding high-resolution image  
Training data: a collection of high/low resolution images  $\{x_i, y_i\}$



# Cascaded Diffusion Models



Stacking diffusion models advantages:

- Smaller models
- Training in parallel
- Individual number of steps tuning

(64 × 64)



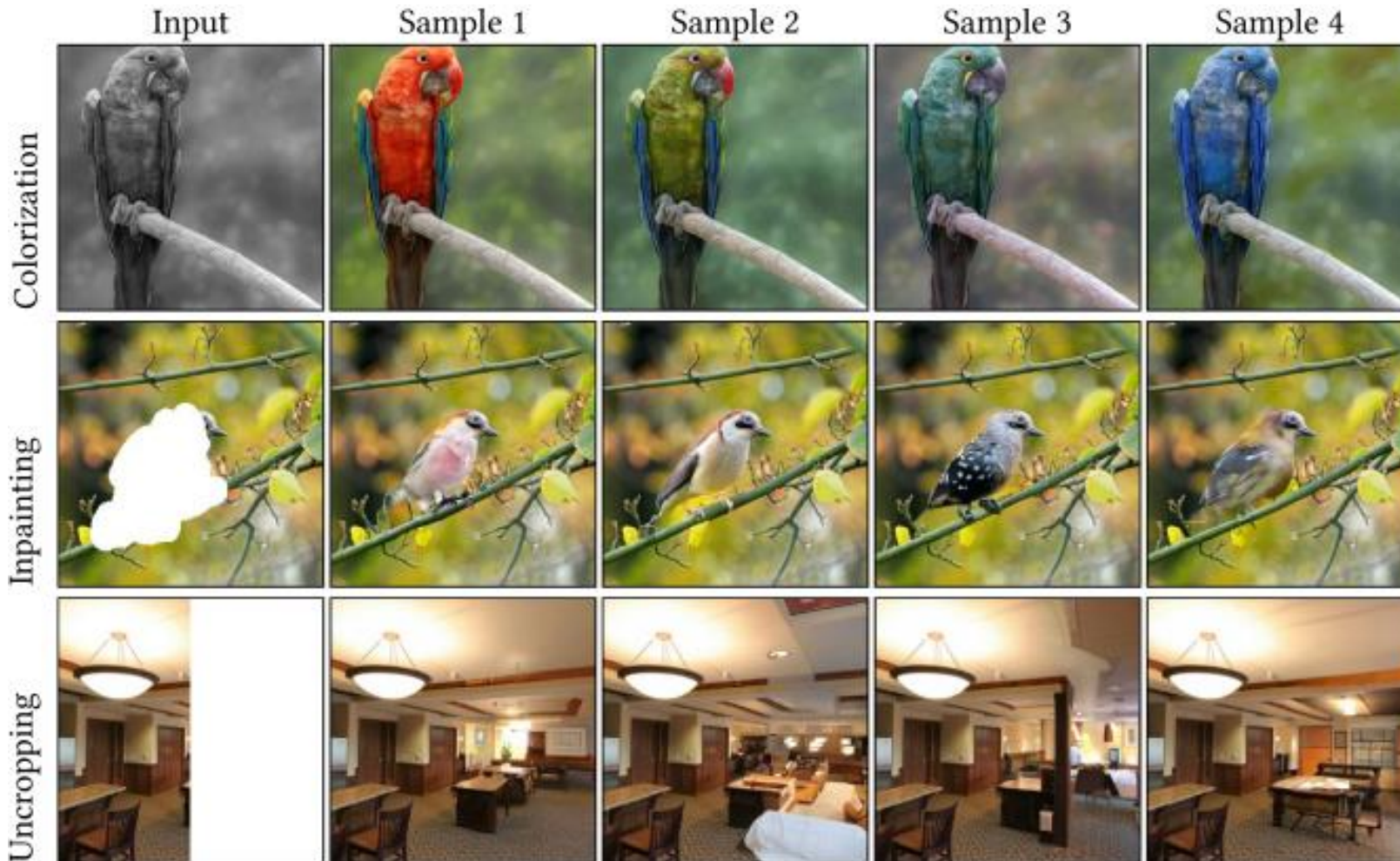
(256 × 256)



(1024 × 1024)



# Image-to-Image Translation Tasks



# Resources:

- [Tutorial on Denoising Diffusion-based Generative Modeling: Foundations and Applications](#), CVPR'22
- [Understanding Diffusion Models: A Unified Perspective](#), Calvin Luo, Google Research
- [What are Diffusion Models?](#) Ari Seff
- <https://lilianweng.github.io/lil-log/2021/07/11/diffusion-models.html>
- Dhariwal, P. and Nichol, A., 2021. “**Diffusion Models Beat GANs on Image Synthesis.**” arXiv e-prints, pp.arXiv-2105.
- A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, “**Glide: Towards photorealistic image generation and editing with text-guided diffusion models,**” arXiv preprint arXiv:2112.10741, 2021.

# Credit:



Yaniv Nikankin



Rafail Fridman



Omer Bar-Tal



Next class/tutorial

Self-supervised learning