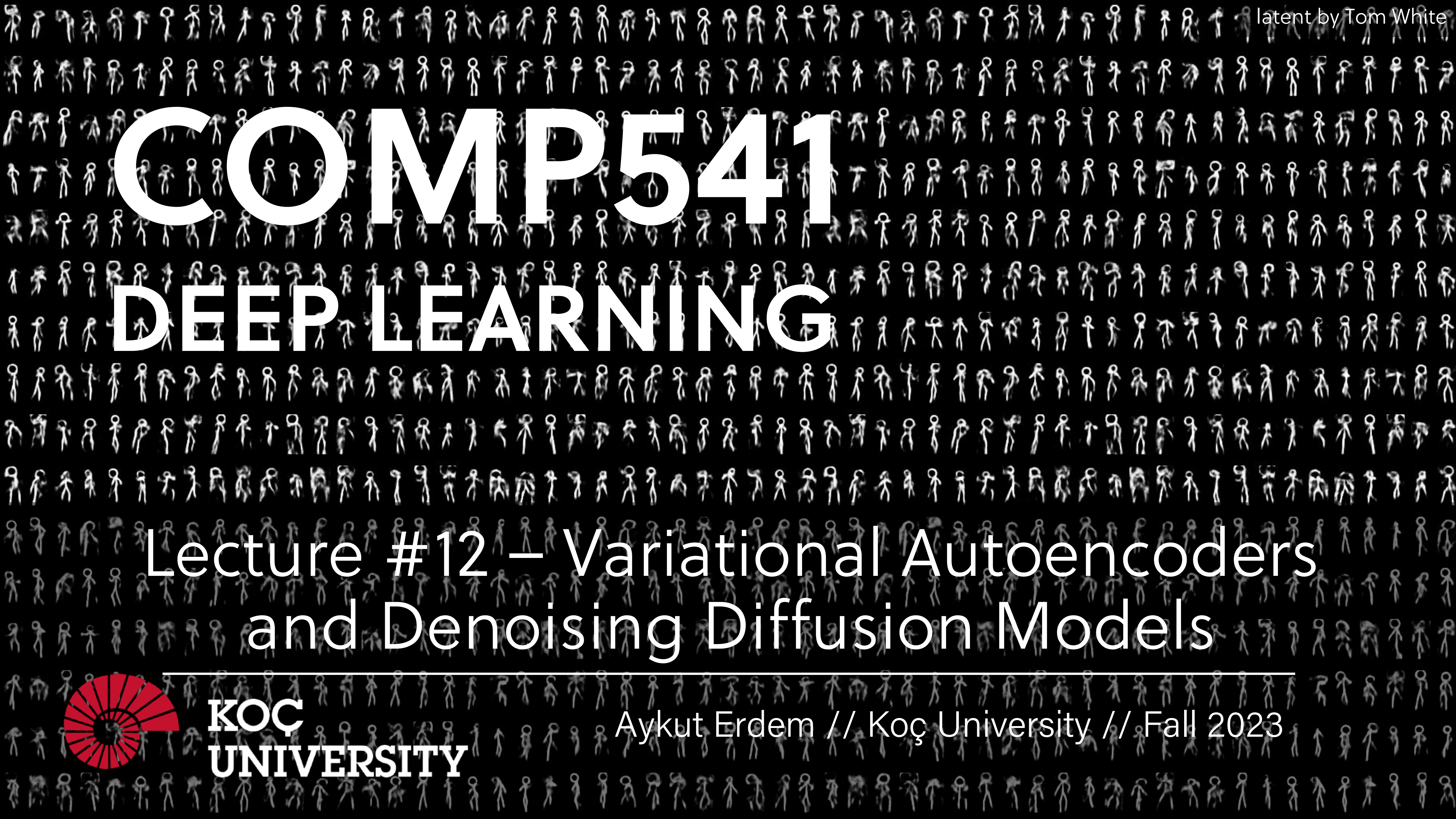


latent by Tom White

COMP541 DEEP LEARNING

Lecture #12 – Variational Autoencoders and Denoising Diffusion Models

KOÇ UNIVERSITY

Aykut Erdem // Koç University // Fall 2023

Previously on COMP541

- generative adversarial networks (GANs)
- normalizing flow models

Artificial faces synthesized by StyleGAN (Nvidia)

Lecture overview

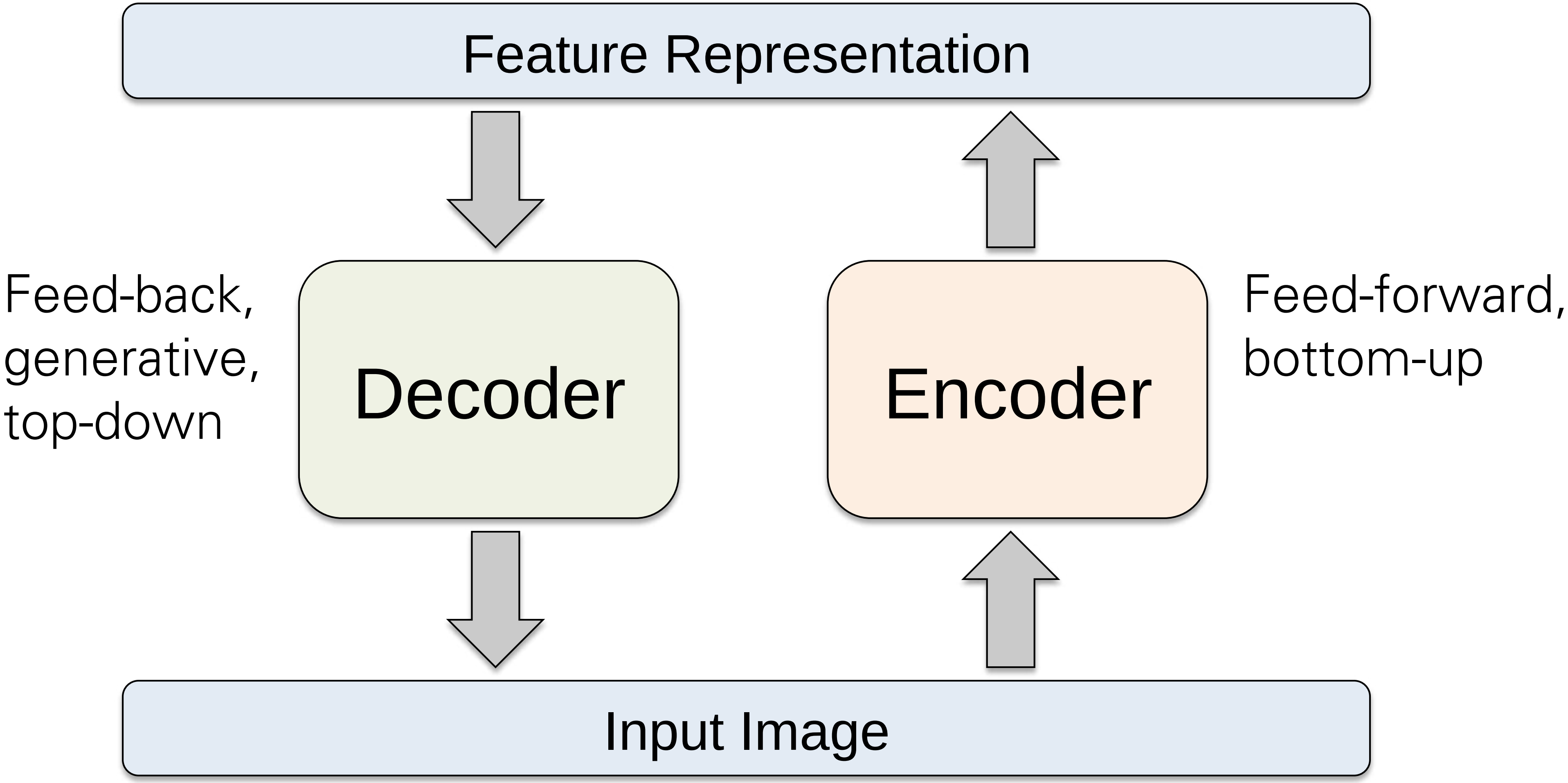
- variational autoencoders (VAEs)
- vector quantized VAEs (VQ-VAEs)
- denoising diffusion models

Disclaimer: Much of the material and slides for this lecture were borrowed from
—Pavlov Protopapas, Mark Glickman and Chris Tanner's Harvard CS109B class
—Andrej Risteski's CMU 10707 class
—David McAllester's TTIC 31230 class
—Andrew Owens's EECS 442/504 class
—Sangwoo Mo's talk titled "Introduction to Diffusion Models"
—Robin Rombach's slides on "Latent Diffusion Models"

Lecture overview

- **variational autoencoders (VAEs)**
- vector quantized VAEs (VQ-VAEs)
- denoising diffusion models

Recap: Autoencoders



- Details of what goes insider the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Parameter space of autoencoder

- Let’s examine the latent space of an AE.
- Is there any separation of the different classes? If the AE learned the “essence” of the MNIST images, similar images should be close to each other.
- Plot the latent space and examine the separation.
- Here we plot the 2 PCA components of the latent space.

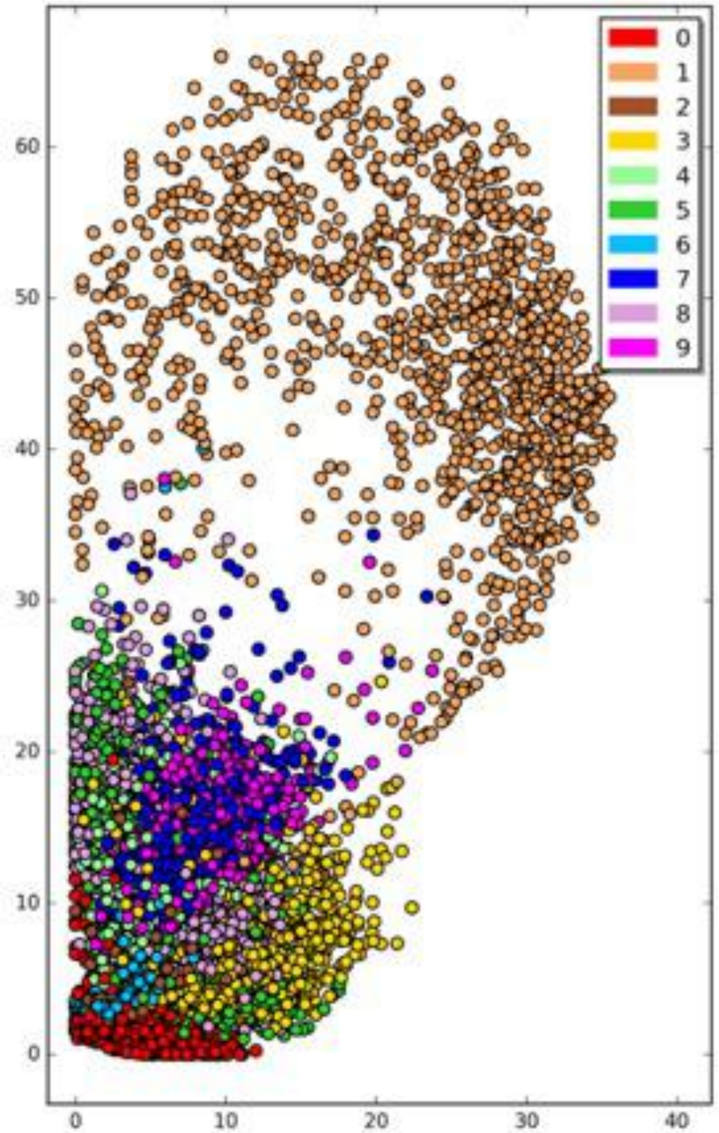
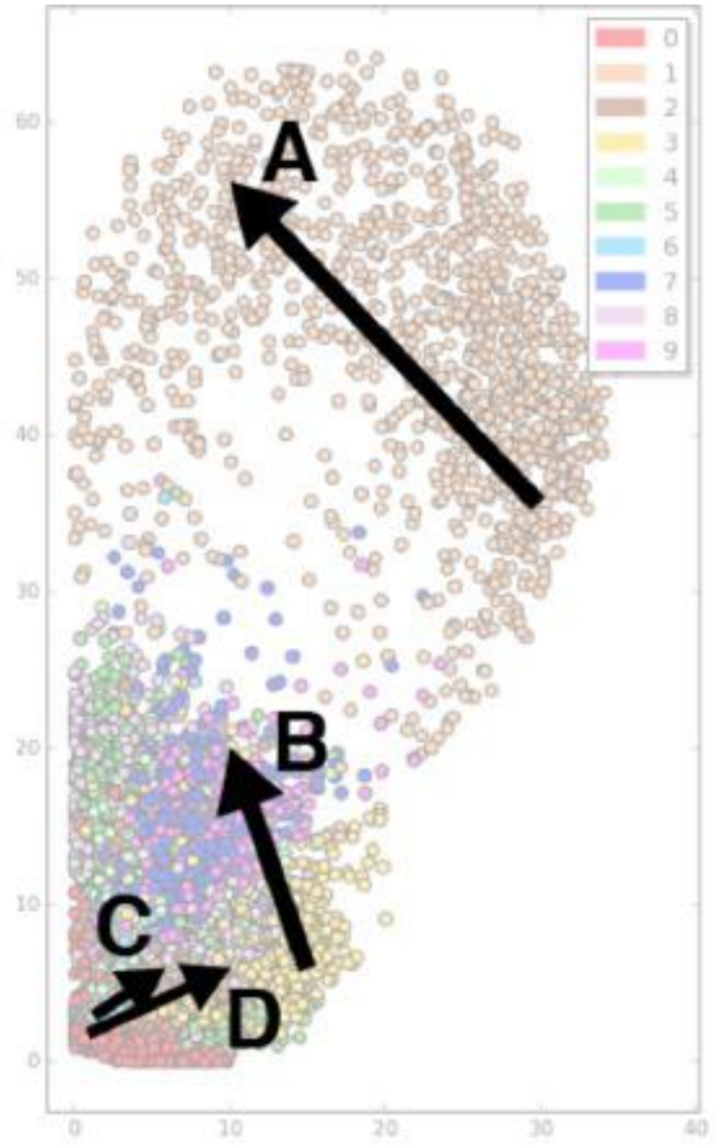


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Traversing the latent space



- We start at the start of the arrows in latent space and then move to end of the arrow in 7 steps.
- For each value of z we use the already trained decoder to produce an image.

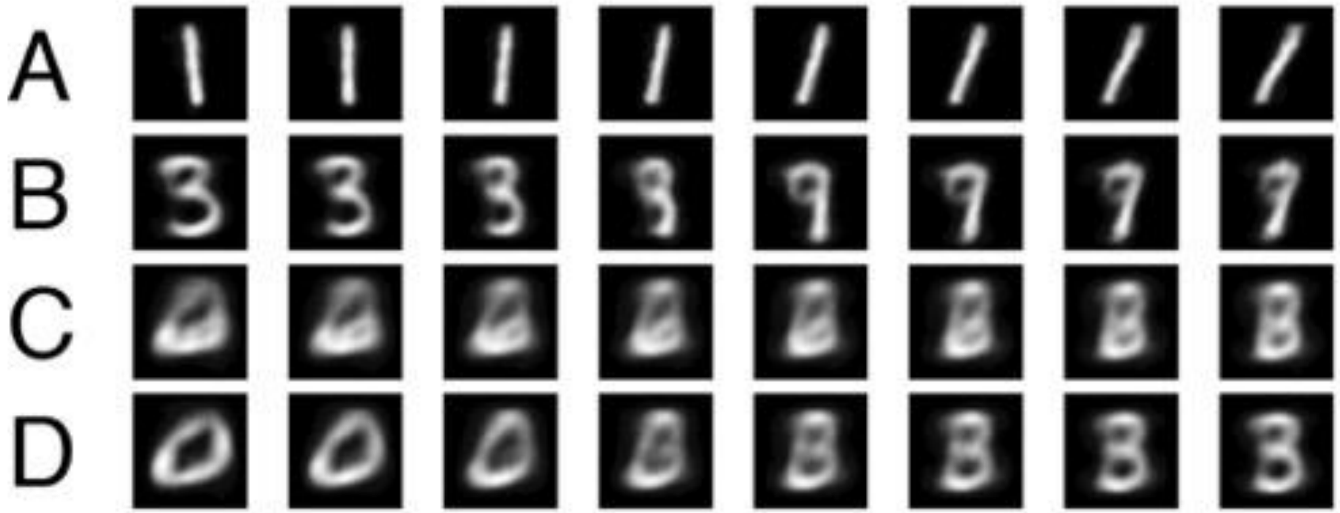


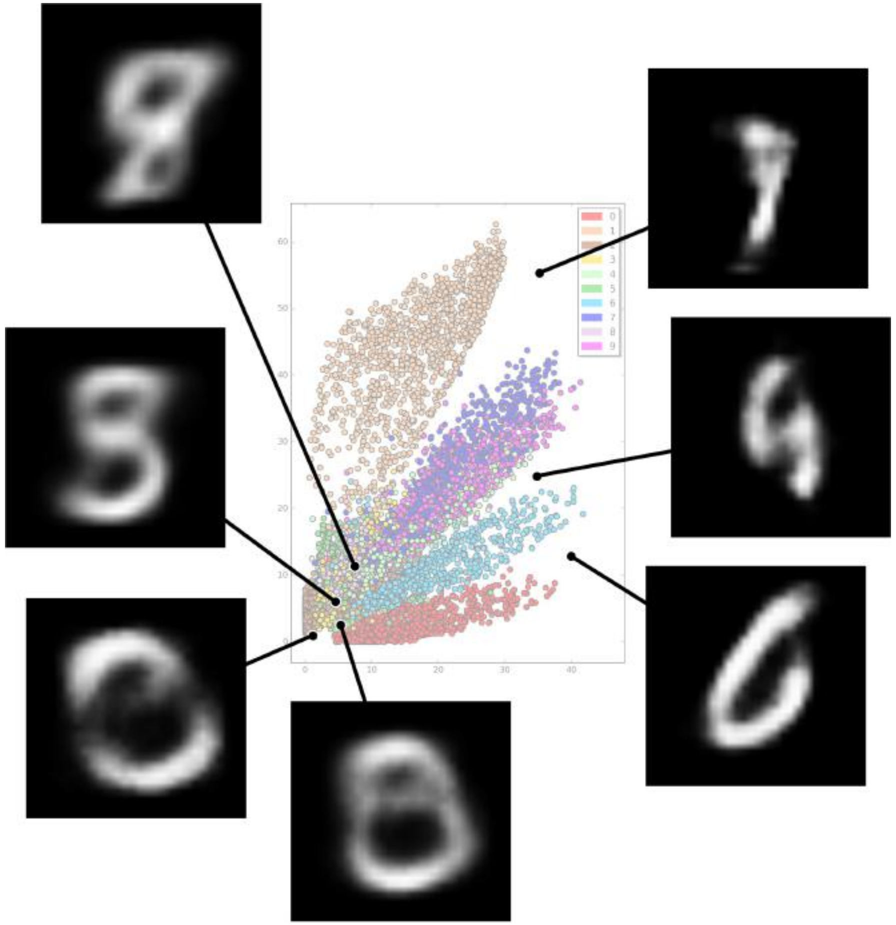
Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Problems with Autoencoders

- Gaps in the latent space

- Discrete latent space

- Separability in the latent space



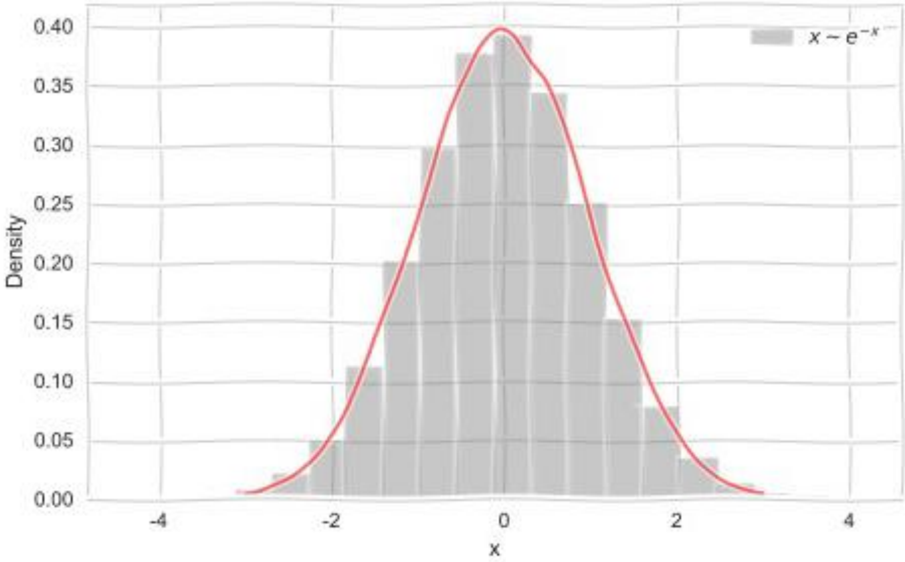
Generative models

- Imagine we want to generate data from a distribution,

$$x \sim p(x)$$

- e.g.

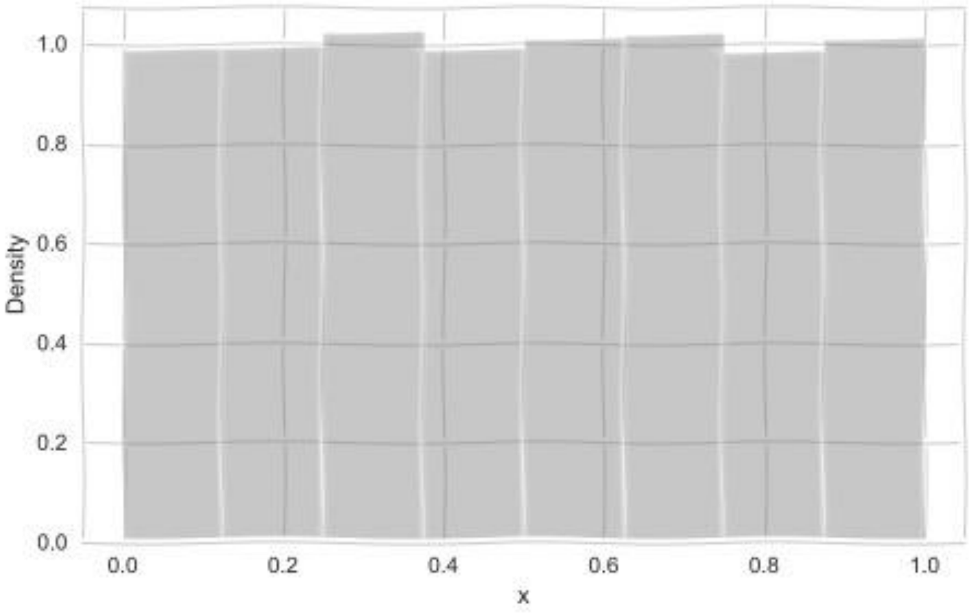
$$x \sim \mathcal{N}(\mu, \sigma)$$



Generative models

- But how do we generate such samples?

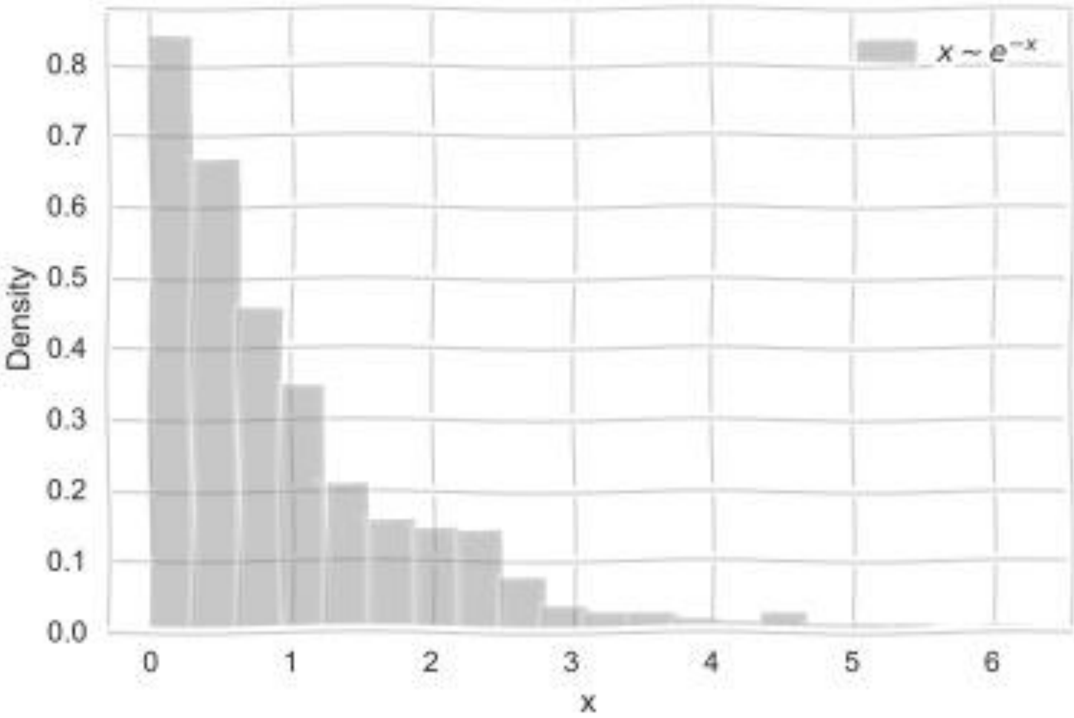
$$z \sim \mathrm{Unif}(0, 1)$$



Generative models

- But how do we generate such samples?

$$z \sim \mathrm{Unif}(0,1) \quad \mathrm{x} = \ln \mathrm{z}$$



Generative models

- In other words we can think that if we choose ***z*∼Uniform** then there is a mapping:

$$x = f(z)$$

such as:

$$x \sim p(x)$$

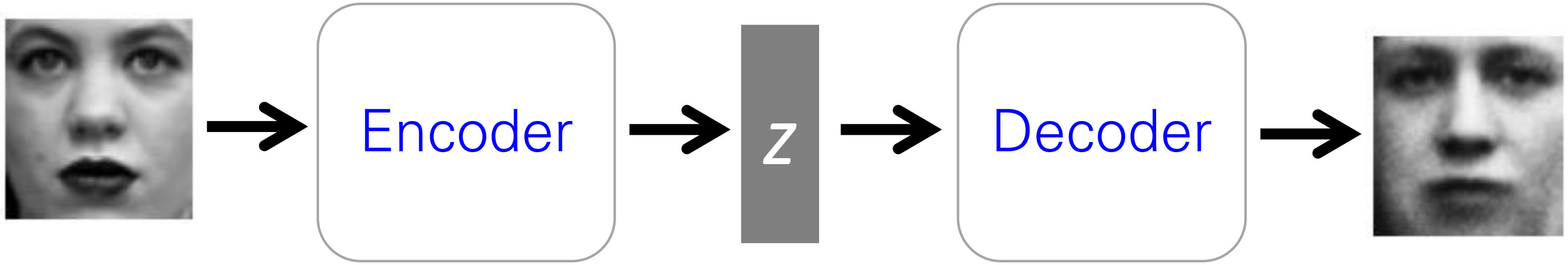
where in general f is some complicated function.

- We already know that **Neural Networks are great in learning complex functions.**

$$\begin{tikzcd} \boxed{z \sim g(z)} \arrow[r] & \boxed{x = f(z)} \arrow[r] & \boxed{x \sim p(x)} \end{tikzcd}$$

Traditional Autoencoders

- In traditional autoencoders, we can think of encoder and decoders as some function mapping.



Encoder

z

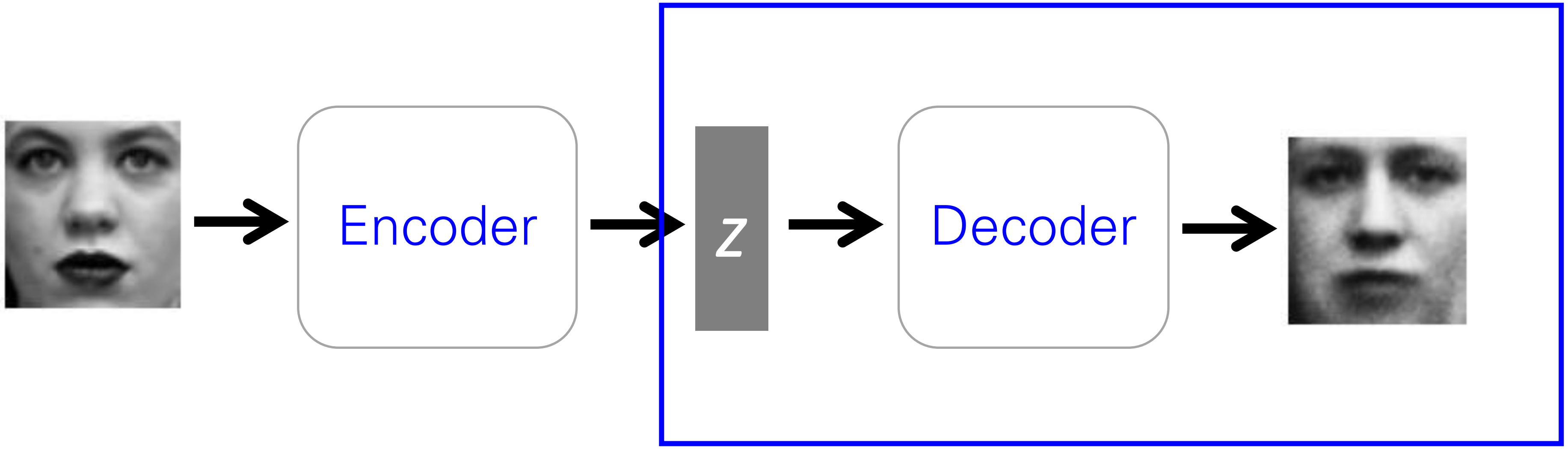
Decoder

$\boxed{z = h(x)}$

$\boxed{\hat{x} = f(z)}$

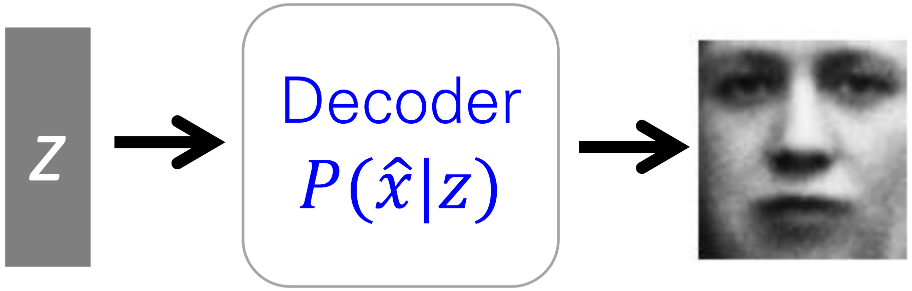
Variational Autoencoders

- To go to variational autoencoders, we need to first add some **stochasticity** and think of it as a probabilistic modeling.



Variational Autoencoders

Sample from g(z)
e.g. Standard
Gaussian

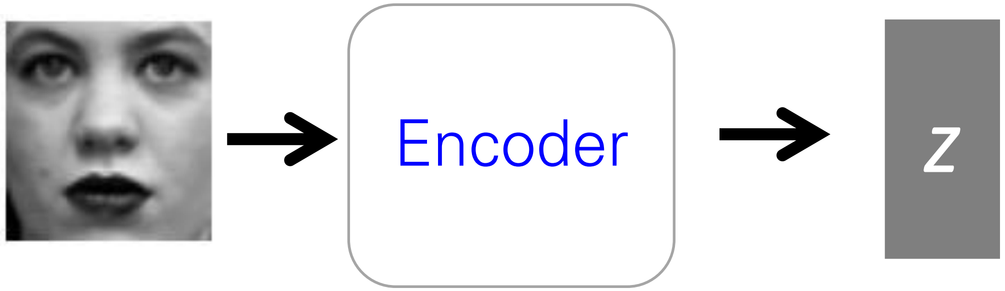


z

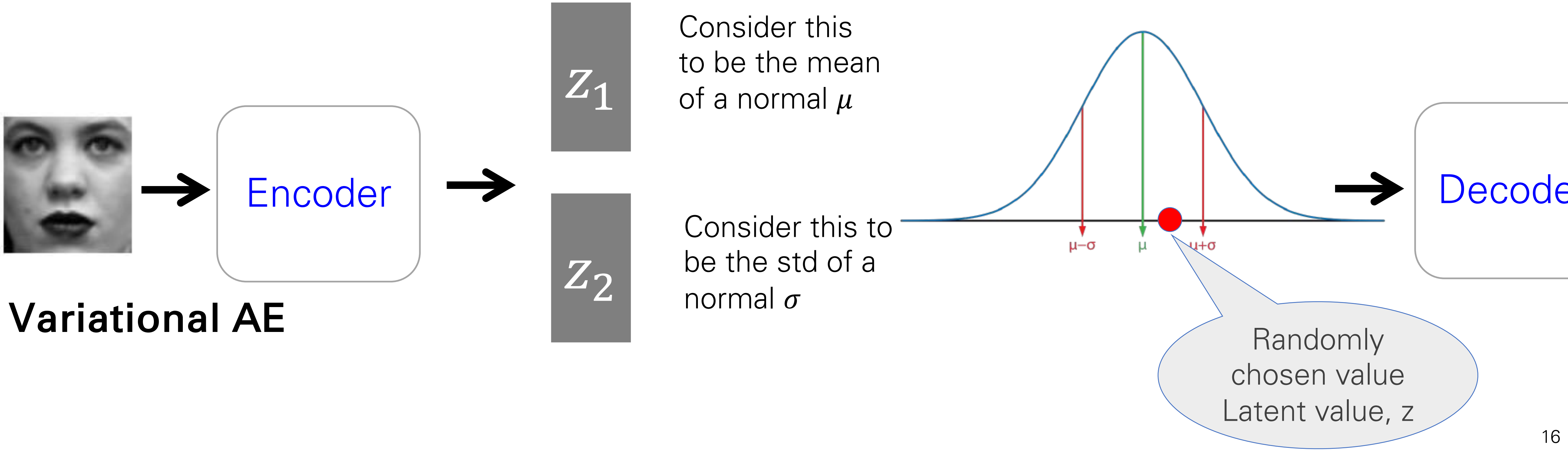
Decoder
$P(\hat{x}|z)$

$\boxed{z \sim g(z)}$ $\boxed{\hat{x} = f(z)}$ $\boxed{\hat{x} \sim P(x|z)}$

Variational Autoencoders



Traditional AE



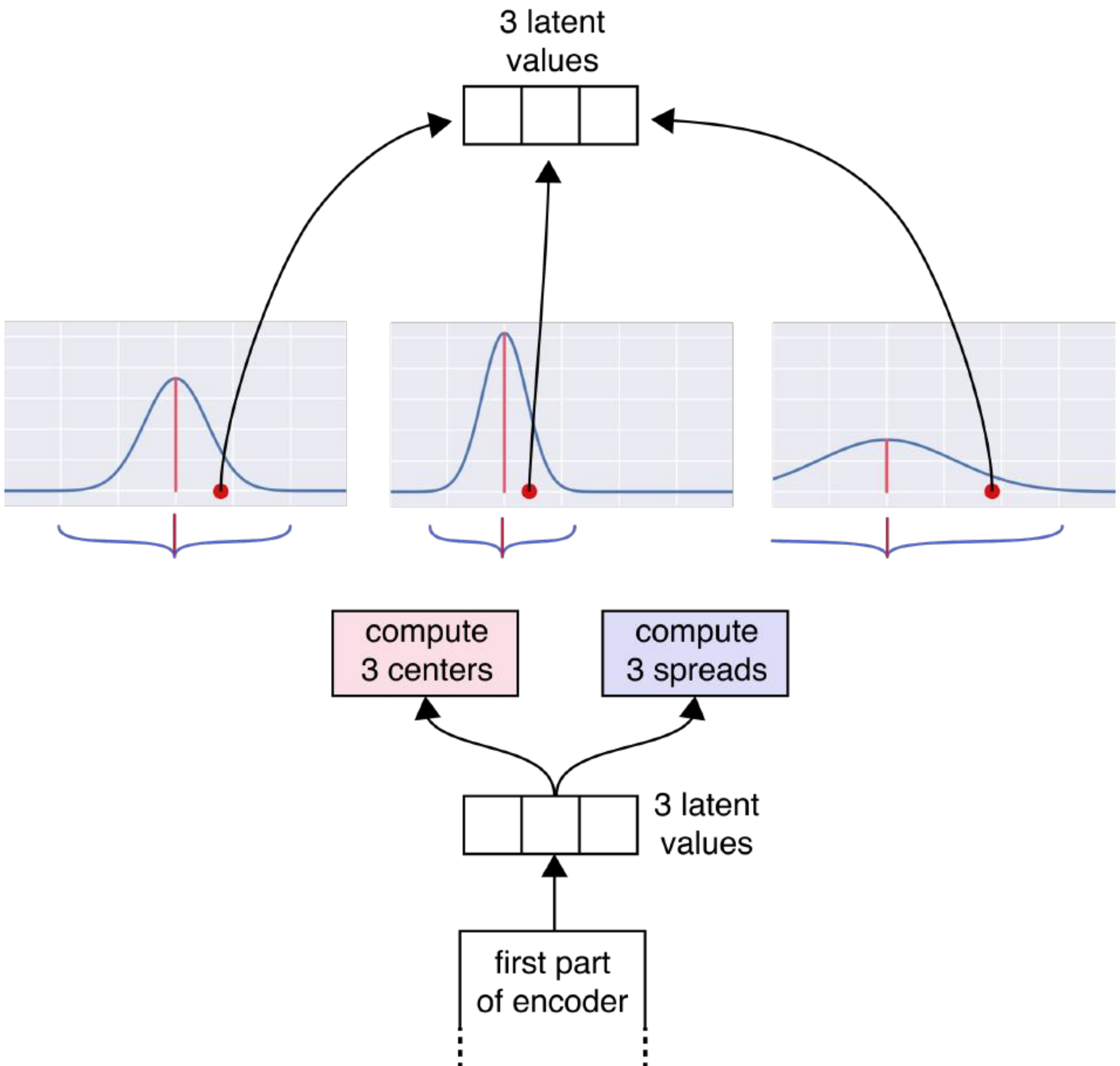
Consider this to be the mean of a normal μ

Consider this to be the std of a normal σ

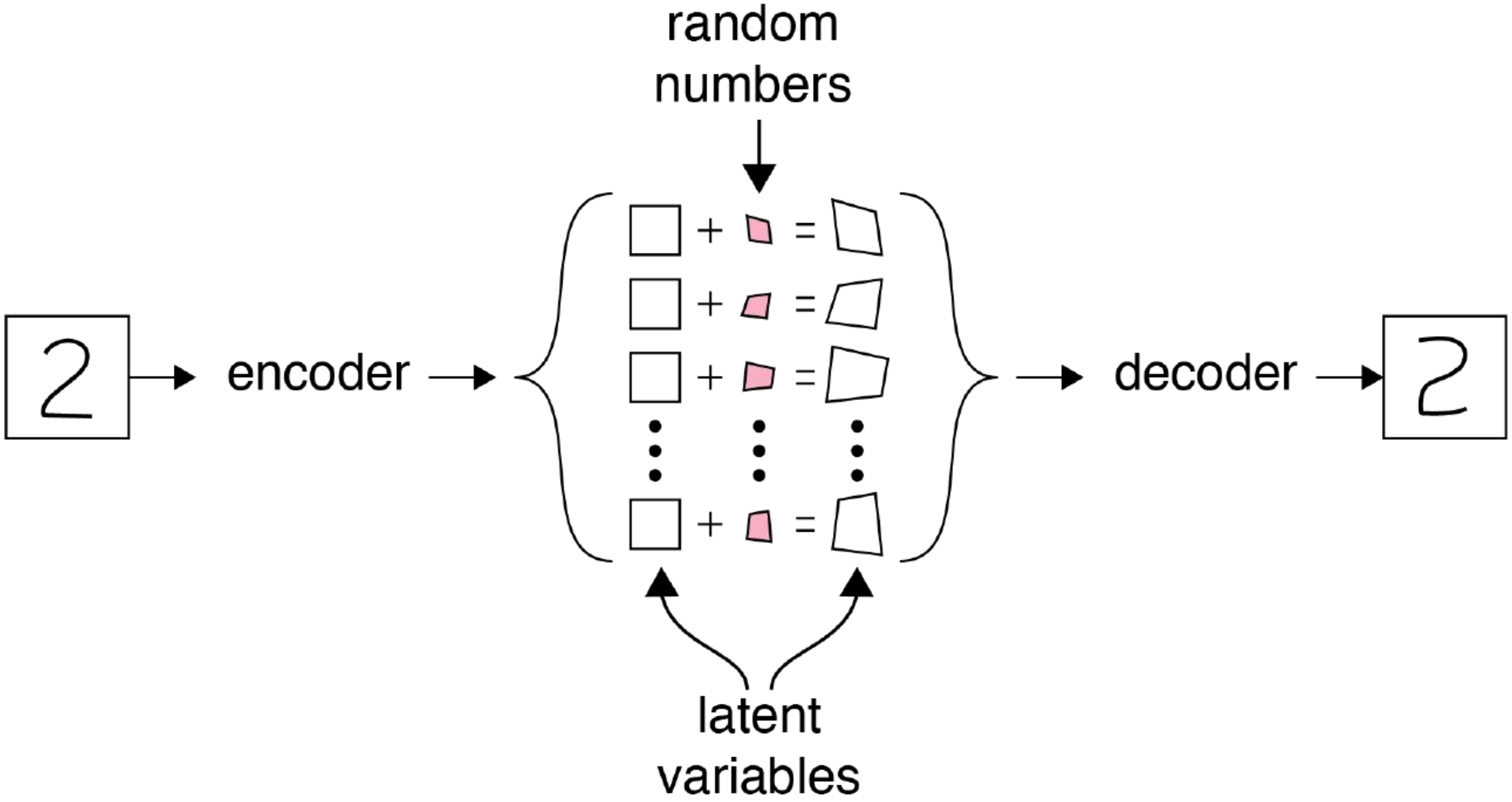
Randomly chosen value Latent value, z

Variational AE

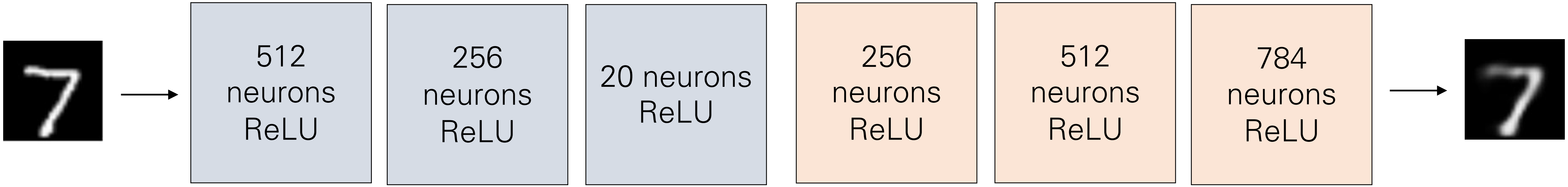
Variational Autoencoders



Variational Autoencoders



Variational Autoencoders



512 neurons ReLU

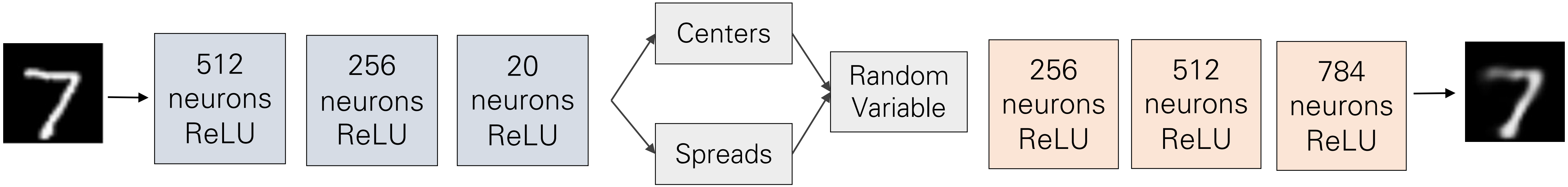
256 neurons ReLU

20 neurons ReLU

256 neurons ReLU

512 neurons ReLU

784 neurons ReLU



512 neurons ReLU

256 neurons ReLU

20 neurons ReLU

Centers

Spreads

Random Variable

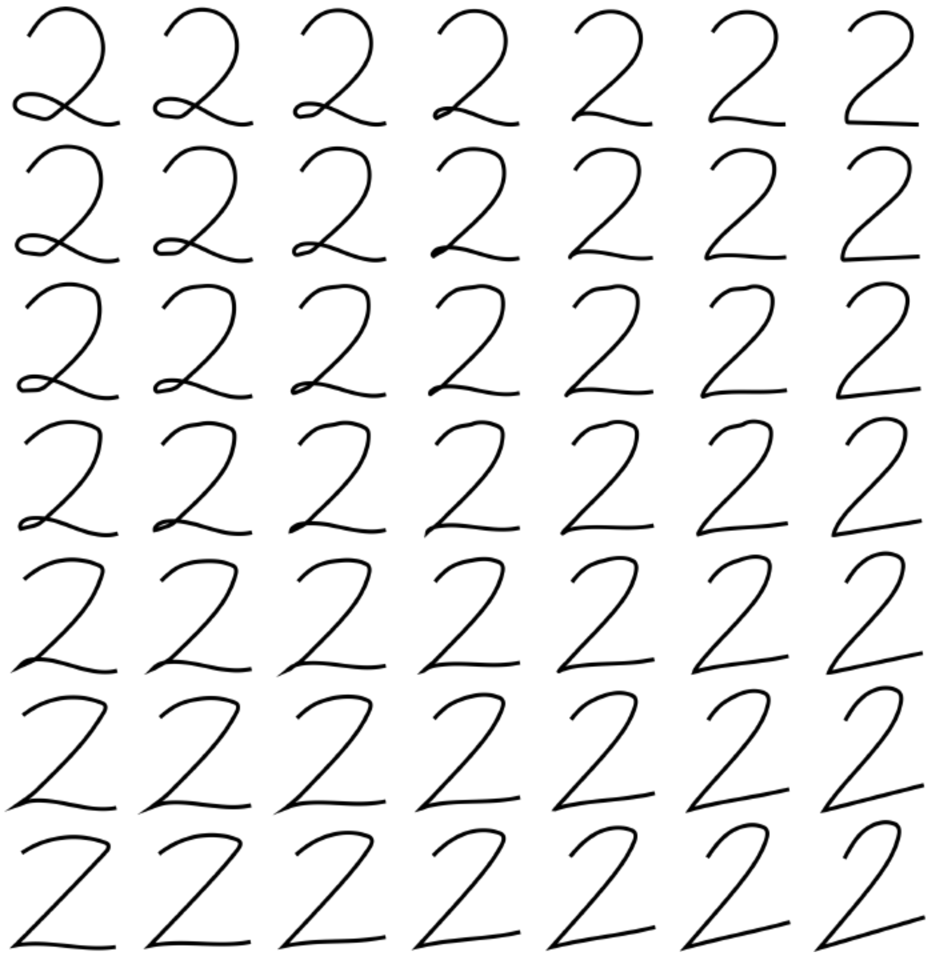
256 neurons ReLU

512 neurons ReLU

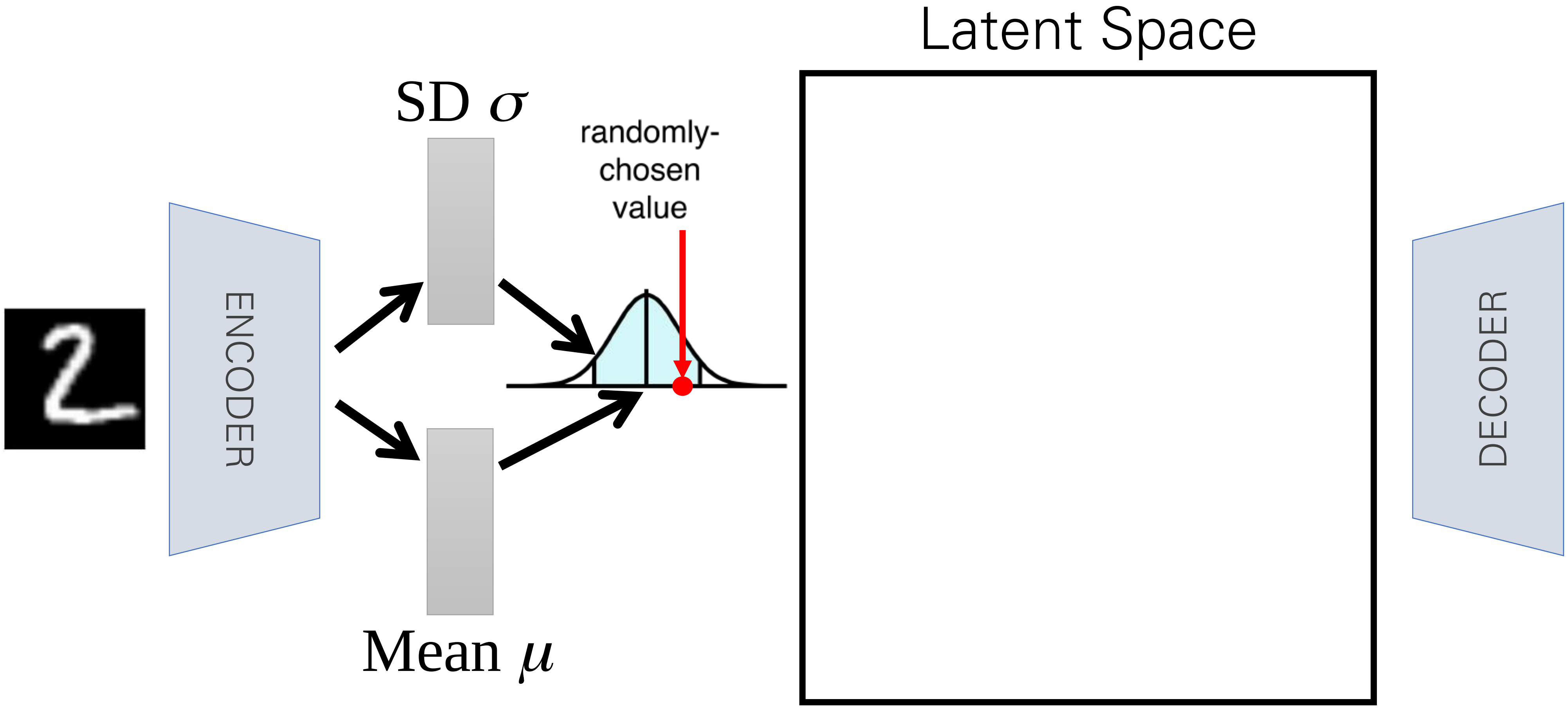
784 neurons ReLU

Separability in Variational Autoencoders

- Separability is not only between classes but we also want similar items in the same class to be near each other.
- For example, there are different ways of writing "2", we want similar styles to end up near each other.
- Let's examine VAE, there is something magic happening once we add stochasticity in the latent space.

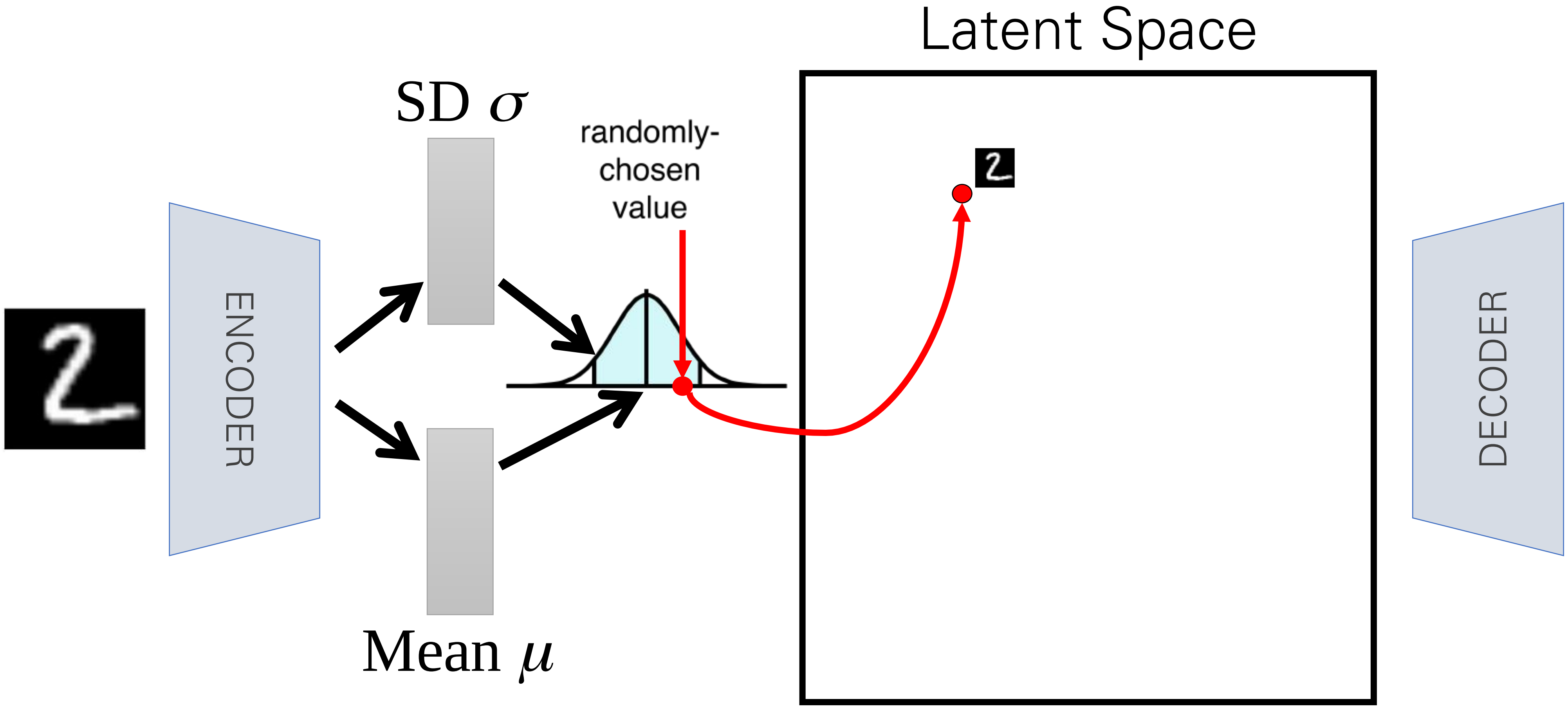


Separability in Variational Autoencoders



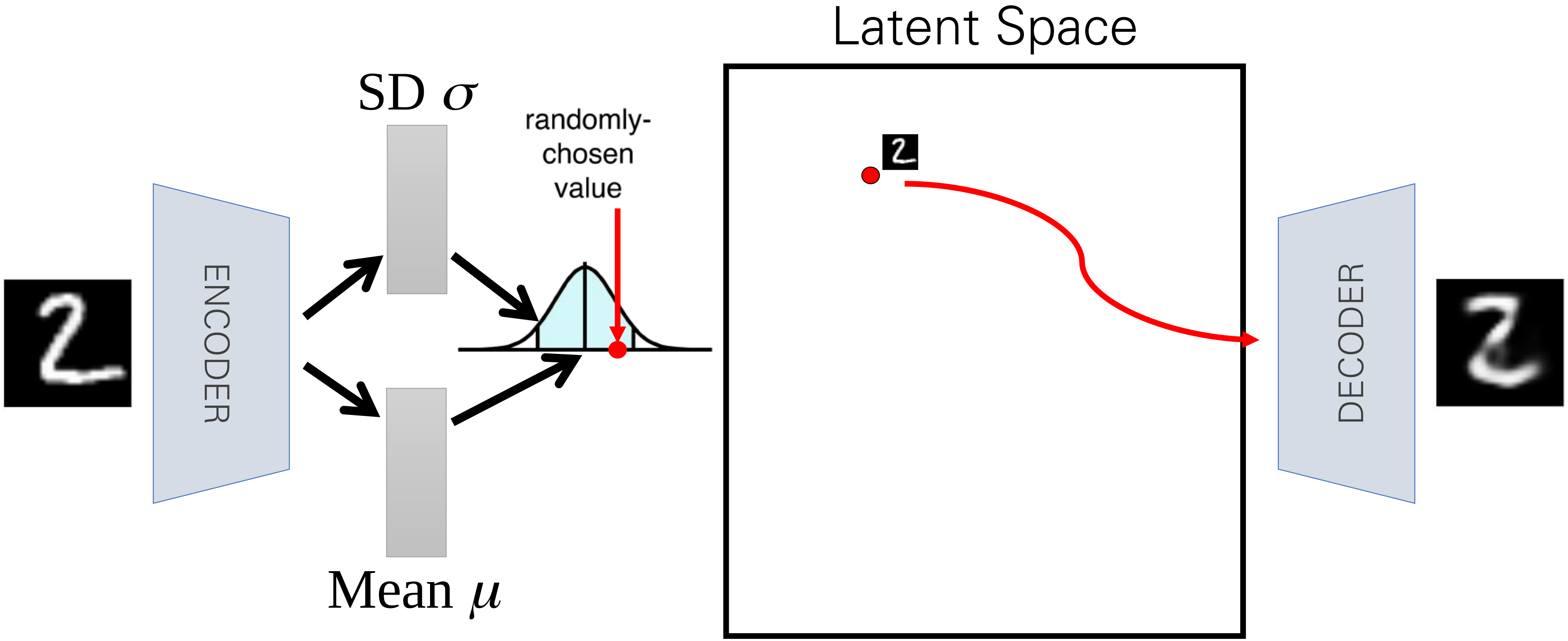
Encode the first sample (a “2”) and find μ_1, σ_1

Separability in Variational Autoencoders



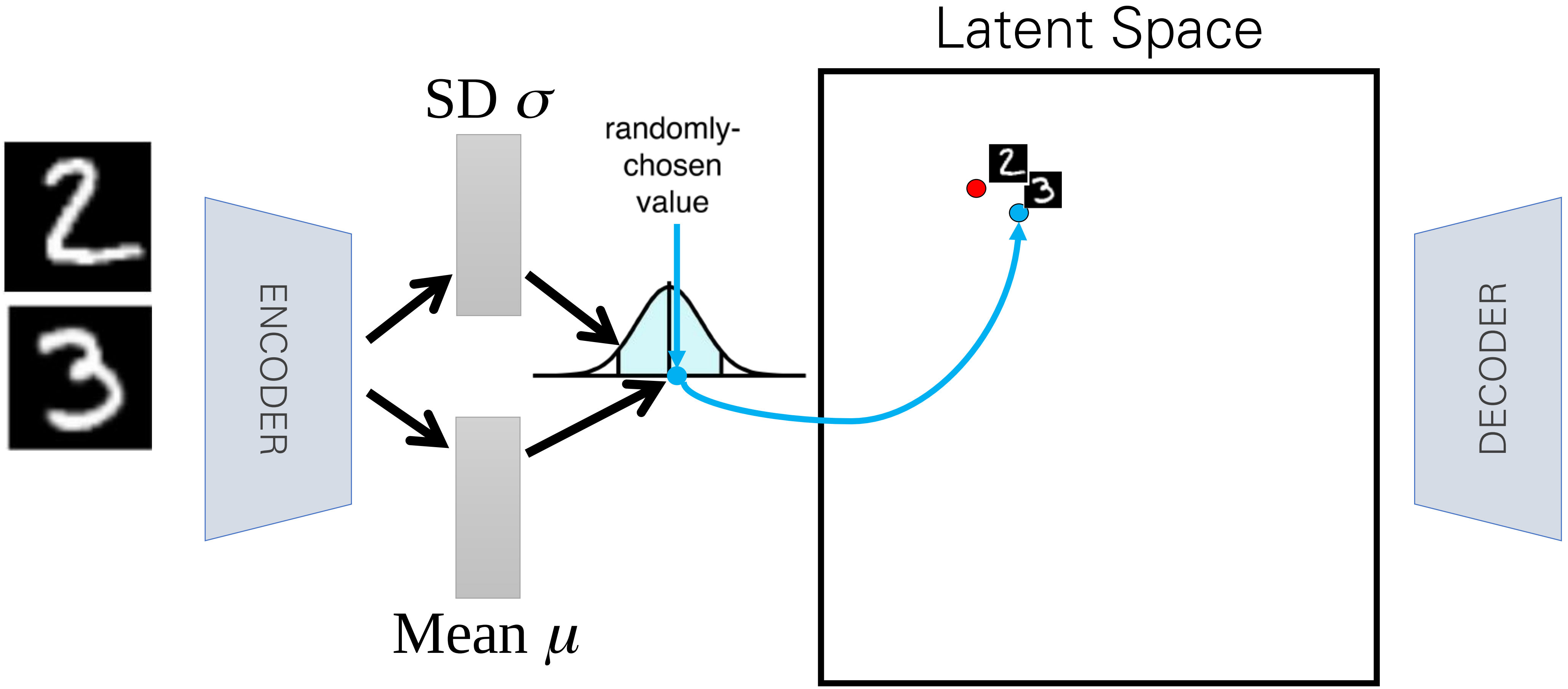
Sample $z_1 \sim N(\mu_1, \sigma_1)$

Blending Latent Variables



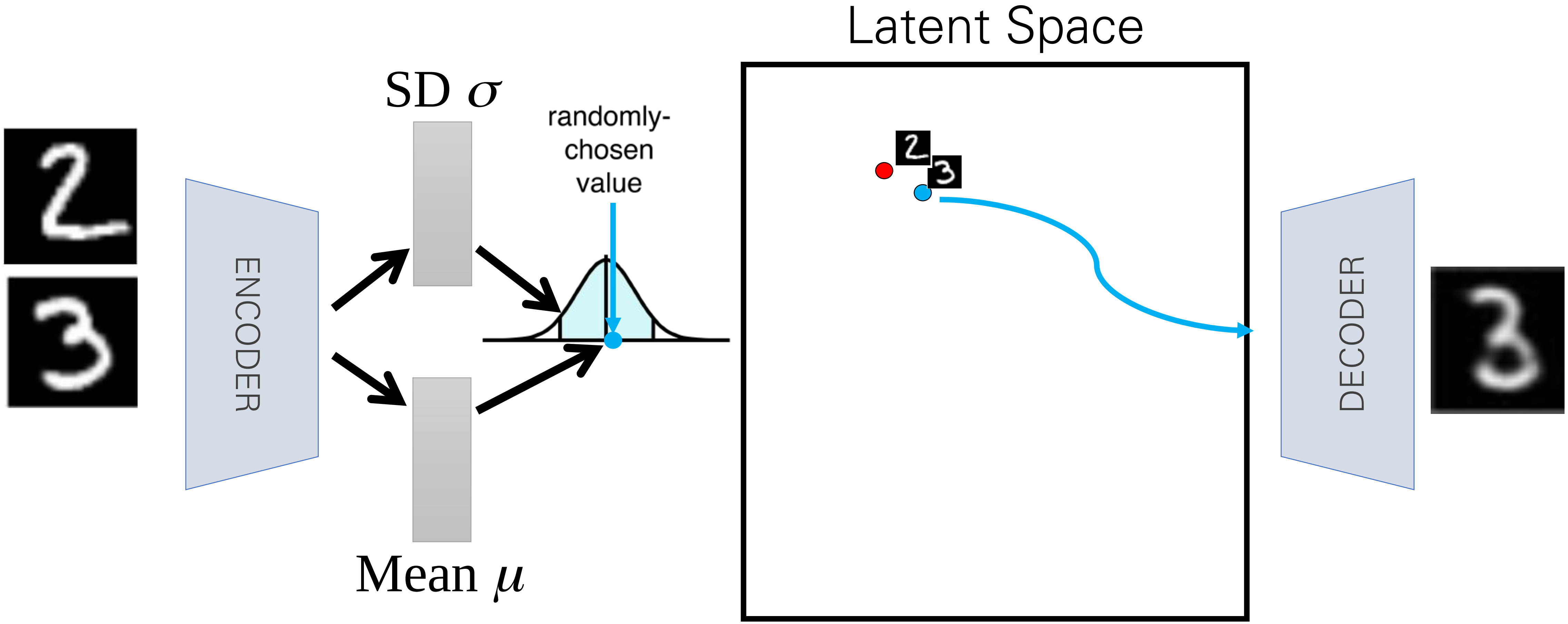
Decode to $\hat{x}_1$

Separability in Variational Autoencoders



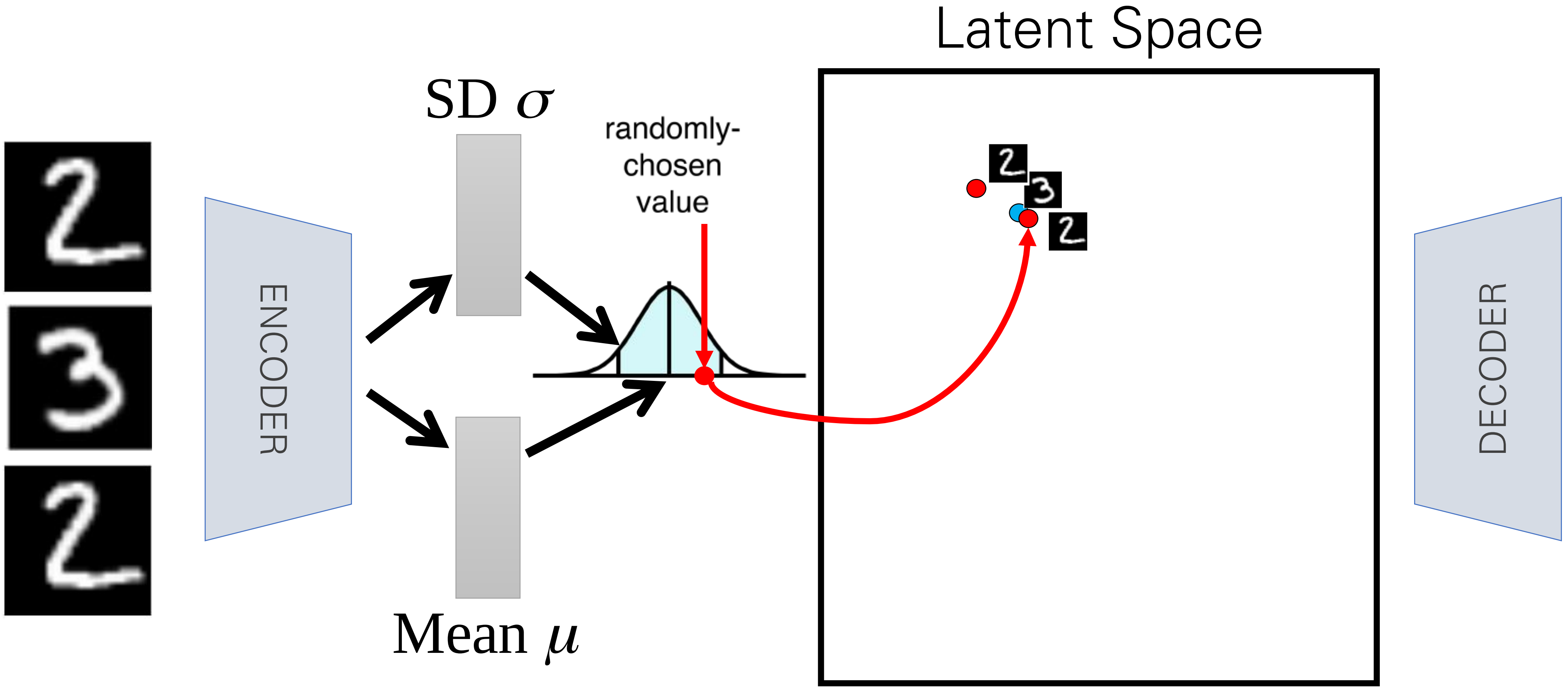
Encode the second sample (a "3") find μ_2, σ_2. Sample $z_2 \sim N(\mu_2, \sigma_2)$

Separability in Variational Autoencoders



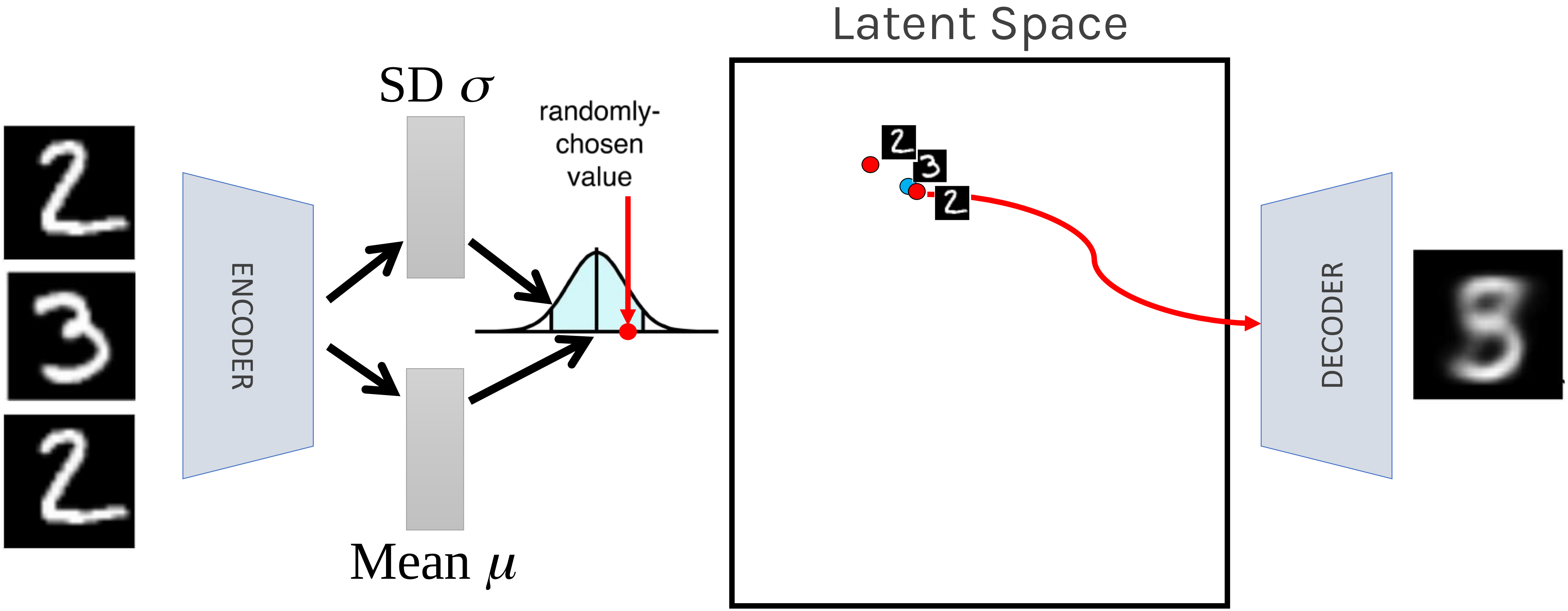
Decode to $\hat{x}_2$

Separability in Variational Autoencoders



Train with the first sample (a "2") again and find μ_1, σ_1. However, $z_1 \sim N(\mu_1, \sigma_1)$ **will not be the same**. It can happen to be close to the "3" in latent space.

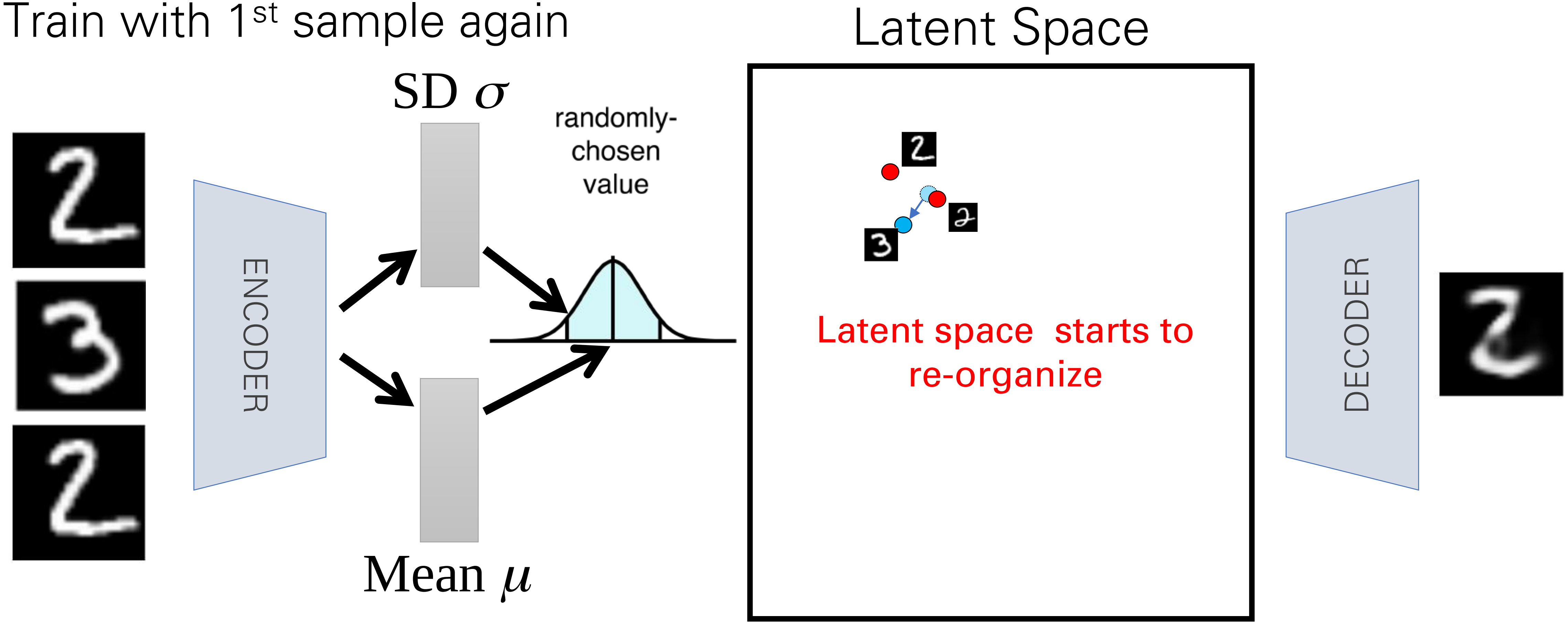
Separability in Variational Autoencoders



Decode to $\hat{x}_1$. Since the decoder only knows how to map from latent space to $\hat{x}$ space, it will return a “3”.

Separability in Variational Autoencoders

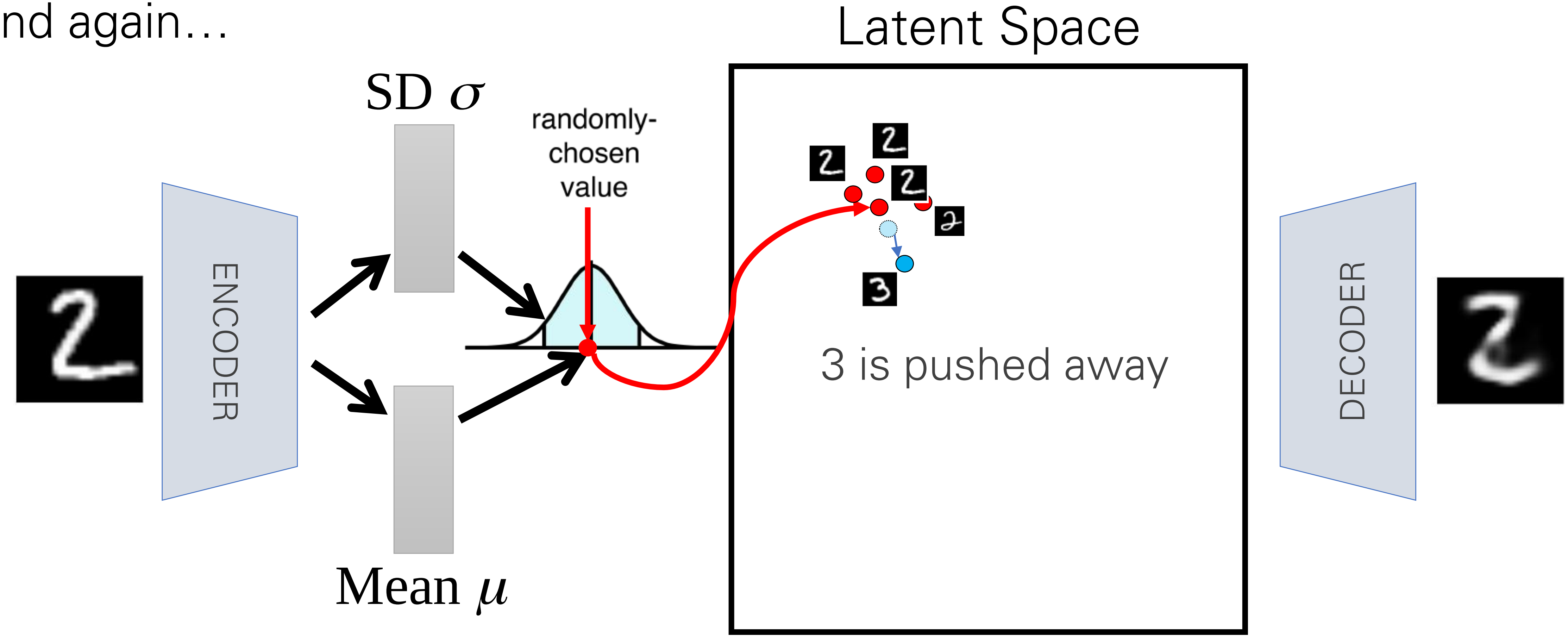
Train with 1st sample again



Latent space starts to re-organize

Separability in Variational Autoencoders

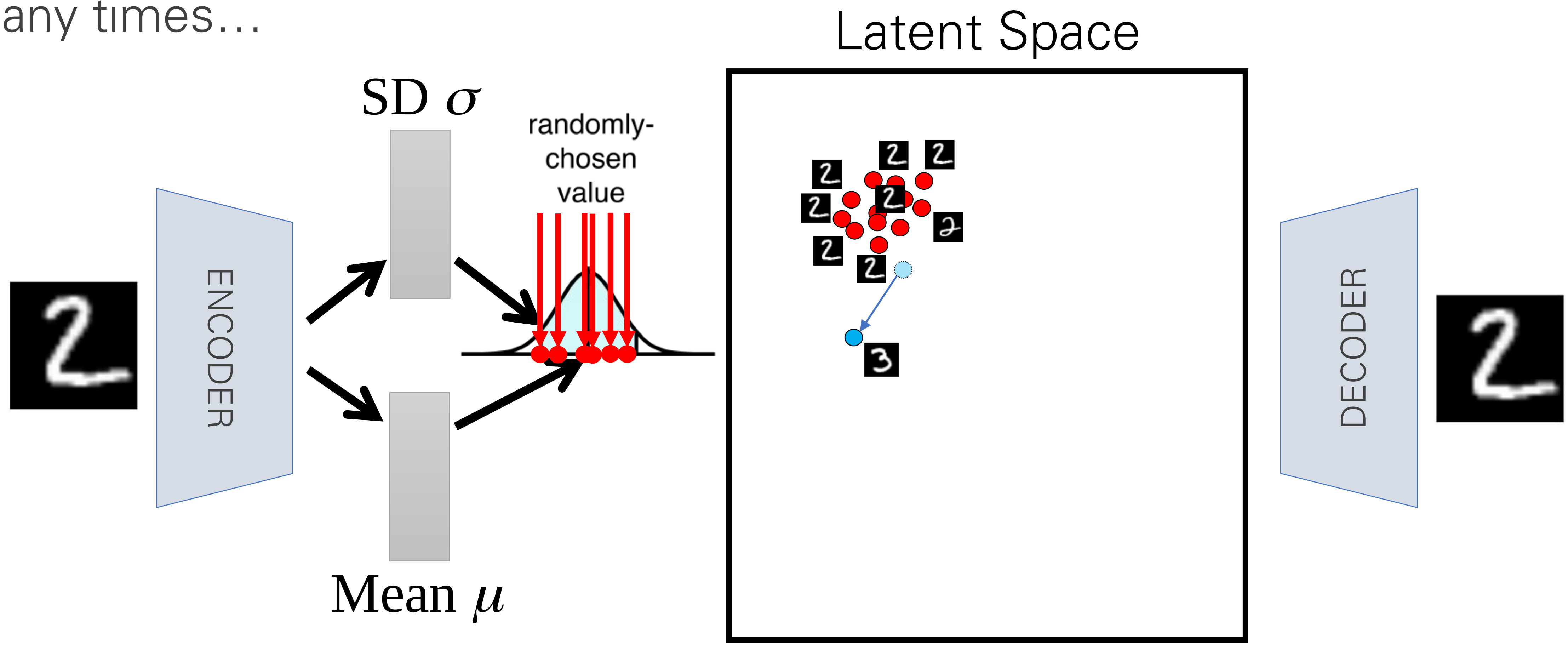
And again…



3 is pushed away

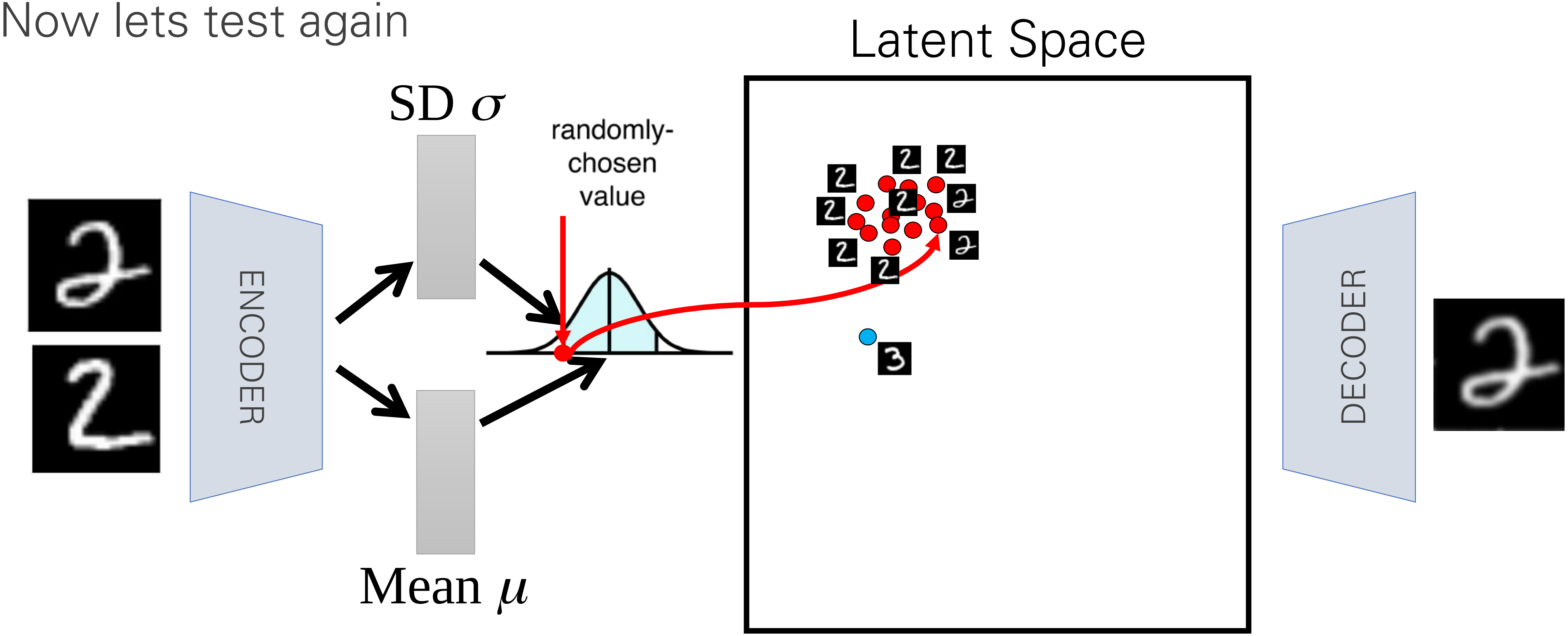
Separability in Variational Autoencoders

Many times…



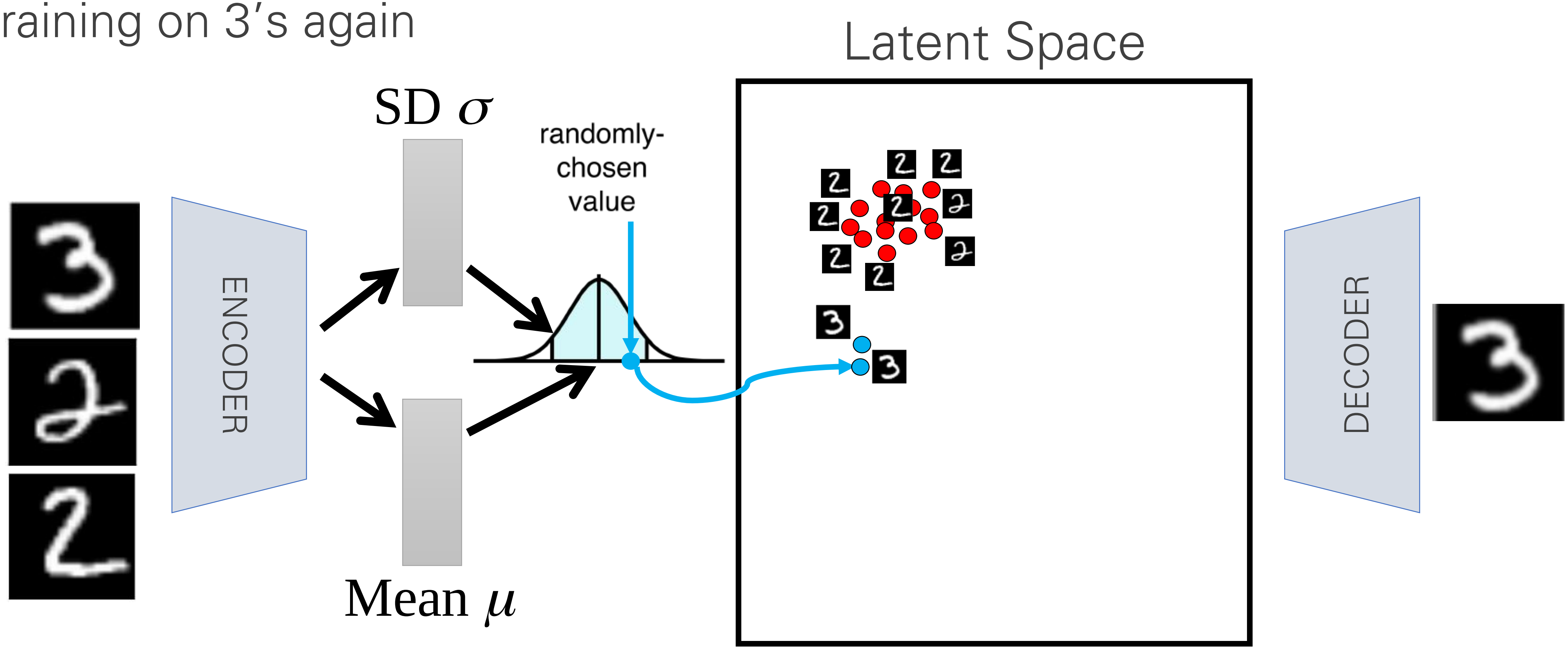
Separability in Variational Autoencoders

Now lets test again



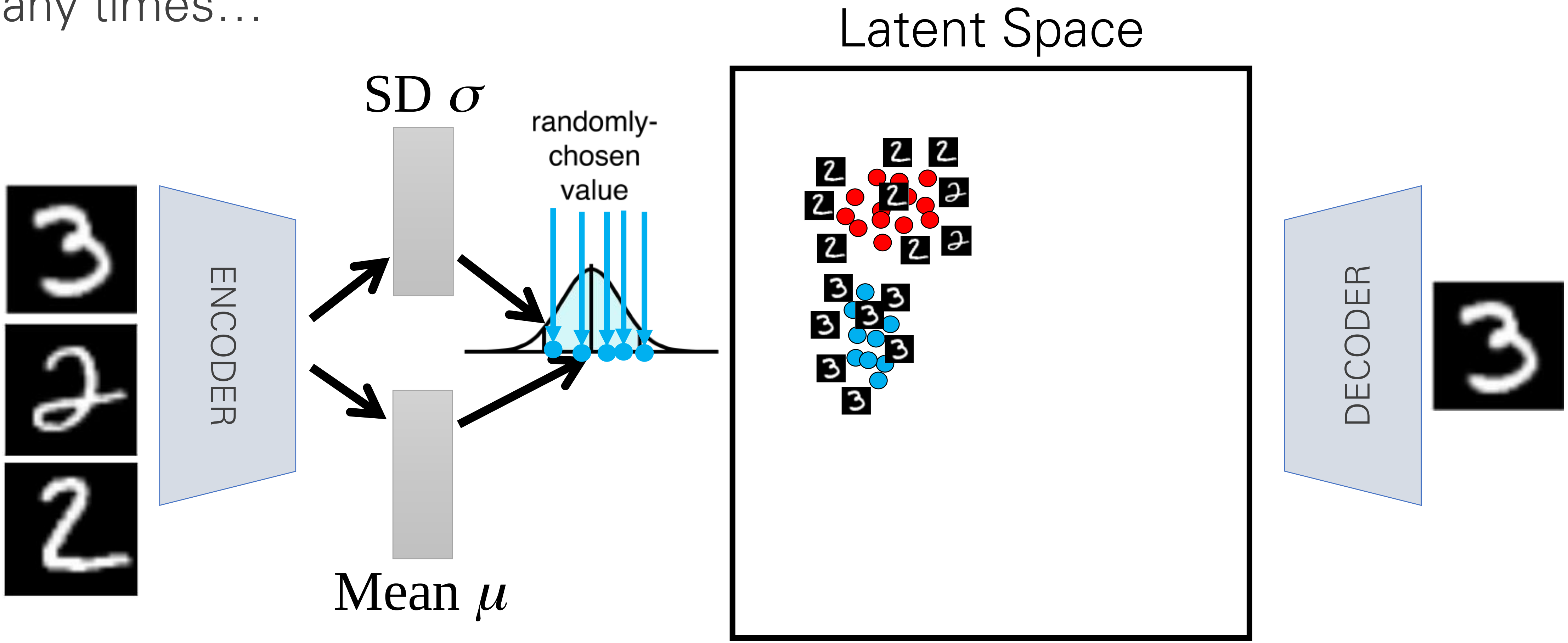
Separability in Variational Autoencoders

Training on 3's again

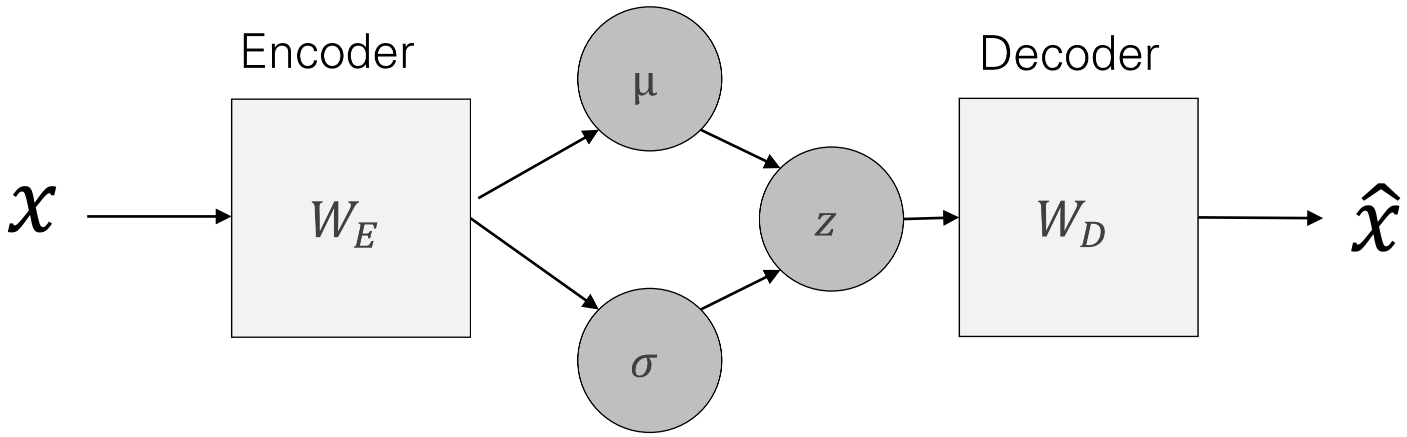


Separability in Variational Autoencoders

Many times…



Training



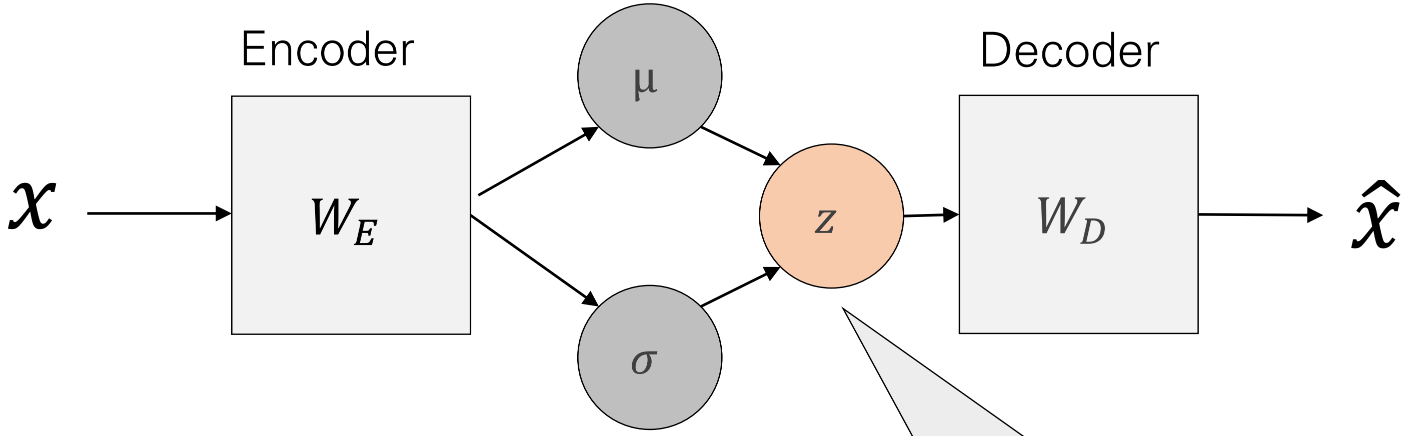
Training means learning W_E and W_D.

- Define a loss function $\mathcal{L}$
- Use stochastic gradient descent (or Adam) to minimize $\mathcal{L}$

The Loss function:

- Reconstruction error: $\mathcal{L}_R = \frac{1}{n}\sum_i (x_i - \hat{x}_i)^2$
- Similarity between the probability of z given x, $\mathrm{p}(z|x)$, and some predefined probability distribution $\mathrm{p(z)}$, which can be computed by Kullback-Leibler divergence (KL): $KL(p(z|x)||p(z))$

Bayesian AE



Parameters of the model (θ is z)

Bayes rule:

$$\mathrm{p}(\theta|D) \propto p(D|\theta)p(\theta)$$

Posterior for our parameters, z is:

$$\mathrm{p}(z|x,\hat{x}) \propto p(\hat{x}|z,x)p(z)$$

Posterior predictive, probability to see $\hat{x}$ given x; **this is INFERENCE**:

$$\mathrm{p}(\hat{x}|x) = \int p(\hat{x}|z,x)p(z|x)dz$$

Decoder: NN

Posterior

Bayesian AE

The posterior, $P(z|x,\hat{x})$, can be sampled with MCMC, i.e. no minimization of Loss function. How?

1. Set the priors, $p(z)$
2. Define the likelihood, $P(\hat{x}|z,x)$
3. Propose a new z* and:
    a. check if $P(z^*|x,\hat{x})/P(z|x,\hat{x})$ >1: accept, z^*
    b. If $P(z^*|x,\hat{x})/P(z|x,\hat{x})$ <1 throw a random coin and accept/reject z^*
4. This will converge to true $P(z|x,\hat{x})$!
5. Calculate $P(\hat{x}|x) = \int P(\hat{x}|z,x)P(z|x)dz$ (Note: this is easily done with sample from z and re-weight given the likelihood)

DOABLE!

Variational AE

Problem: z is the dimensionality of your latent space, which can be too large. In other words this $\int p(\hat{x}|z,x)p(z|x)dz$ becomes intractable.

Instead we turn this into a minimization problem – Variational Calculus

Find a q$(z|x)$ that is similar to $p(z|x)$ by minimizing their difference.

After some math:

Proposal distribution should resemble a Gaussian

Reconstruction Loss

$$-\mathbf{E}_{z\sim q_\phi(z|x)}\log\left(p_\theta\left(x|z\right)\right) + KL\left(q_\phi\left(z|x\right)\|p_\theta(z)\right)$$

Evidence Lower BOund (ELBO)

Variational AE

- The VAE approach: introduce an inference machine $q_\phi(z \mid x)$ that learns to approximate the posterior $p_\theta(z \mid x)$.
  - Define a variational lower bound on the data likelihood: $p_\theta(x) \geq \mathcal{L}(\theta, \phi, x)$

$$
\begin{aligned}
\mathcal{L}(\theta, \phi, x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x)\right] \\
&= \underbrace{-D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| \, p_\theta(z)\right)}_{\textit{regularization term}} + \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]}_{\textit{reconstruction term}}
\end{aligned}
$$

- What is $q_\phi(z \mid x)$?

Slide credit: Aaron Courville

Variational AE: Math Maximum Likelihood?

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} p_\theta(x^{(i)})$$ Maximize likelihood of dataset $\{x^{(i)}\}_{i=1}^{N}$

Kingma and Welling, ICLR 2014

Variational AE: Math Maximum Likelihood?

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} p_\theta(x^{(i)})$$

Maximize likelihood of dataset $\{x^{(i)}\}_{i=1}^{N}$

$$= \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x^{(i)})$$

Maximize log-likelihood instead because sums are nicer

Kingma and Welling, ICLR 2014

Variational AE: Math Maximum Likelihood?

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} p_\theta(x^{(i)})$$

Maximize likelihood of dataset $\{x^{(i)}\}_{i=1}^{N}$

$$= \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x^{(i)})$$

Maximize log-likelihood instead because sums are nicer

$$p_\theta(x^{(i)}) = \int p_\theta(x^{(i)}, z)dz$$

Marginalize joint distribution

Kingma and Welling, ICLR 2014

Variational AE: Math Maximum Likelihood?

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} p_\theta(x^{(i)})$$

Maximize likelihood of dataset $\{x^{(i)}\}_{i=1}^{N}$

$$= \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x^{(i)})$$

Maximize log-likelihood instead because sums are nicer

$$p_\theta(x^{(i)}) = \int p_\theta(x^{(i)}, z)dz = \int p_\theta(x^{(i)} \mid z)p_\theta(z)dz$$

Intractible integral!

Variational AE: Math

$$\log p_\theta(x^{(i)})$$

Variational AE: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} \left[\log p_\theta(x^{(i)}) \right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Variational AE: Math

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}
\end{aligned}$$

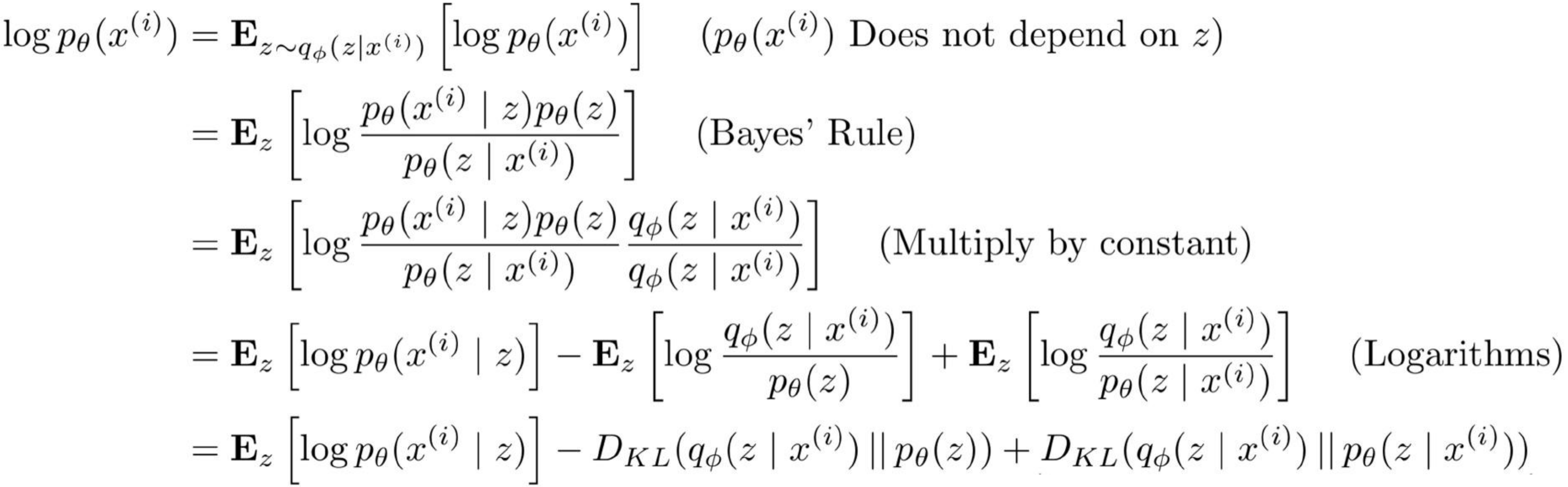
Variational AE: Math

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad (\text{Multiply by constant})
\end{aligned}
$$

Variational AE: Math

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Logarithms})
\end{aligned}
$$

Variational AE: Math



$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

Variational AE: Math

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)} \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)} \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)} \\
&= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)\ \text{“Elbow”}} + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

Variational AE: Math

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)} \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)} \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)} \\
&= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)\ \text{“Elbow”}} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}
\end{aligned}
$$

Variational AE: Math

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)} \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)} \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)} \\
&= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}
\end{aligned}
$$

"Elbow"

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound (elbow)

Variational AE: Math

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Logarithms}) \\
&= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)\ \text{“Elbow”}} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}
\end{aligned}
$$

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound (elbow)

$$\theta^*,\phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

Variational AE: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Reconstruct the input data

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \underbrace{\boxed{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right]} - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

"Elbow"

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

Variational AE: Math

Latent states should follow the prior

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Reconstruct the input data

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \underbrace{\boxed{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right]} - \boxed{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

“Elbow”

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

Variational AE: Math

Latent states should follow the prior

Reconstruct the input data

Sampling with reparam. trick (see paper)

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)}\mid z)\,p_\theta(z)}{p_\theta(z\mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)}\mid z)\,p_\theta(z)}{p_\theta(z\mid x^{(i)})}\,\frac{q_\phi(z\mid x^{(i)})}{q_\phi(z\mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)}\mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z\mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z\mid x^{(i)})}{p_\theta(z\mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \underbrace{\boxed{\mathbf{E}_z\left[\log p_\theta(x^{(i)}\mid z)\right]} - \boxed{D_{KL}(q_\phi(z\mid x^{(i)})\,||\,p_\theta(z))}}_{\mathcal{L}(x^{(i)},\theta,\phi)\ \text{“Elbow”}} + \underbrace{D_{KL}(q_\phi(z\mid x^{(i)})\,||\,p_\theta(z\mid x^{(i)}))}_{\geq 0}$$

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound (elbow)

$$\theta^*,\phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

Variational AE: Math

Latent states should follow the prior

Reconstruct the input data

Everything is Gaussian, closed form solution!

Sampling with reparam. trick (see paper)

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)}\mid z)p_\theta(z)}{p_\theta(z\mid x^{(i)})}\right] \qquad (\text{Bayes' Rule})\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)}\mid z)p_\theta(z)}{p_\theta(z\mid x^{(i)})}\frac{q_\phi(z\mid x^{(i)})}{q_\phi(z\mid x^{(i)})}\right] \qquad (\text{Multiply by constant})\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)}\mid z)\right] - \mathbf{E}_z\left[\log\frac{q_\phi(z\mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log\frac{q_\phi(z\mid x^{(i)})}{p_\theta(z\mid x^{(i)})}\right] \qquad (\text{Logarithms})\\
&= \underbrace{\boxed{\mathbf{E}_z\left[\log p_\theta(x^{(i)}\mid z)\right]} - \boxed{D_{KL}(q_\phi(z\mid x^{(i)})\,||\,p_\theta(z))}}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z\mid x^{(i)})\,||\,p_\theta(z\mid x^{(i)}))}_{\geq 0}
\end{aligned}$$

“Elbow”

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound (elbow)

$$\theta^*,\phi^* = \arg\max_{\theta,\phi}\sum_{i=1}^{N}\mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

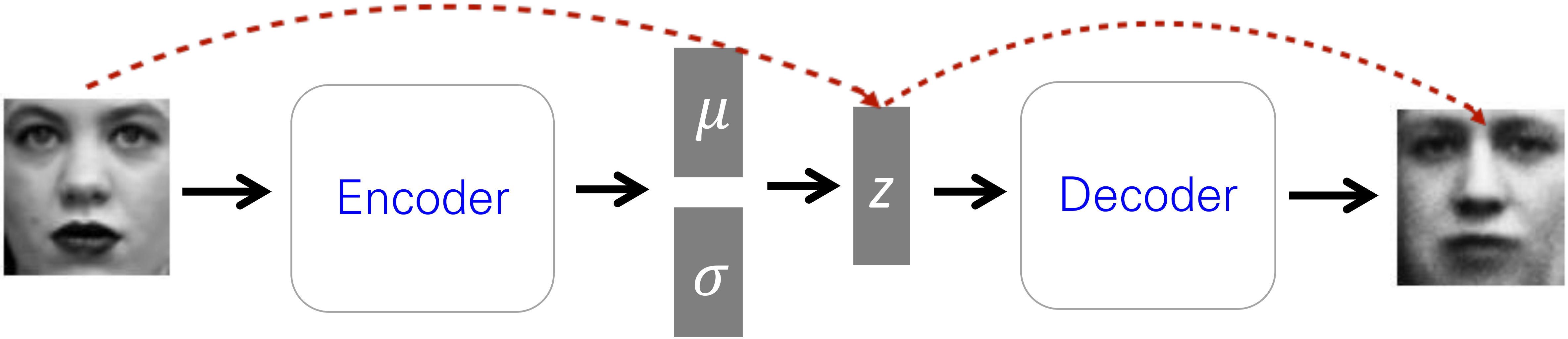
Training VAE

- Apply stochastic gradient descent (SGD)

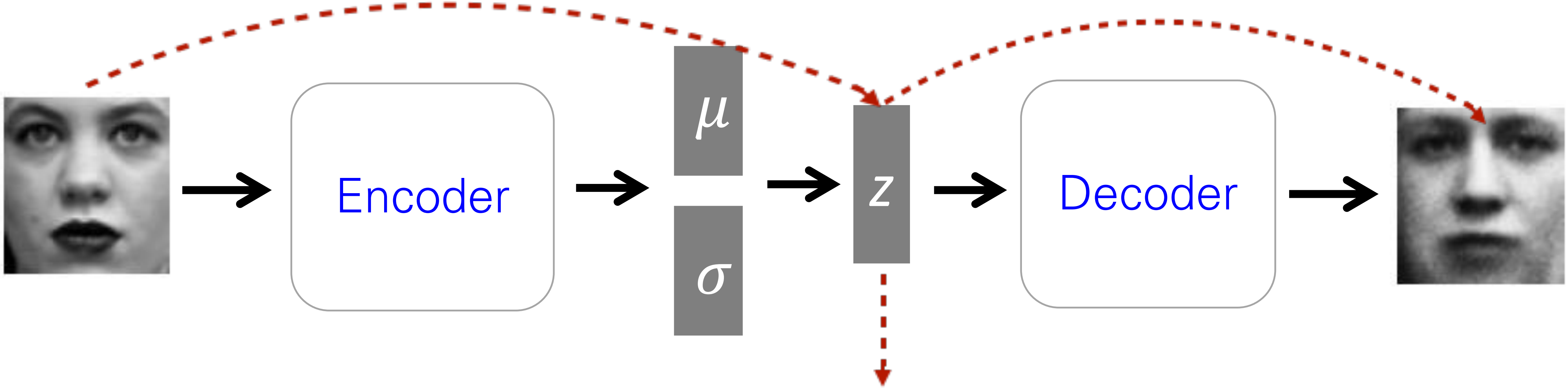
Problem:

- Sampling step not differentiable
- Use a re-parameterization trick
  - Move sampling to input layer, so that the sampling step is independent of the model

Reparametrization Trick

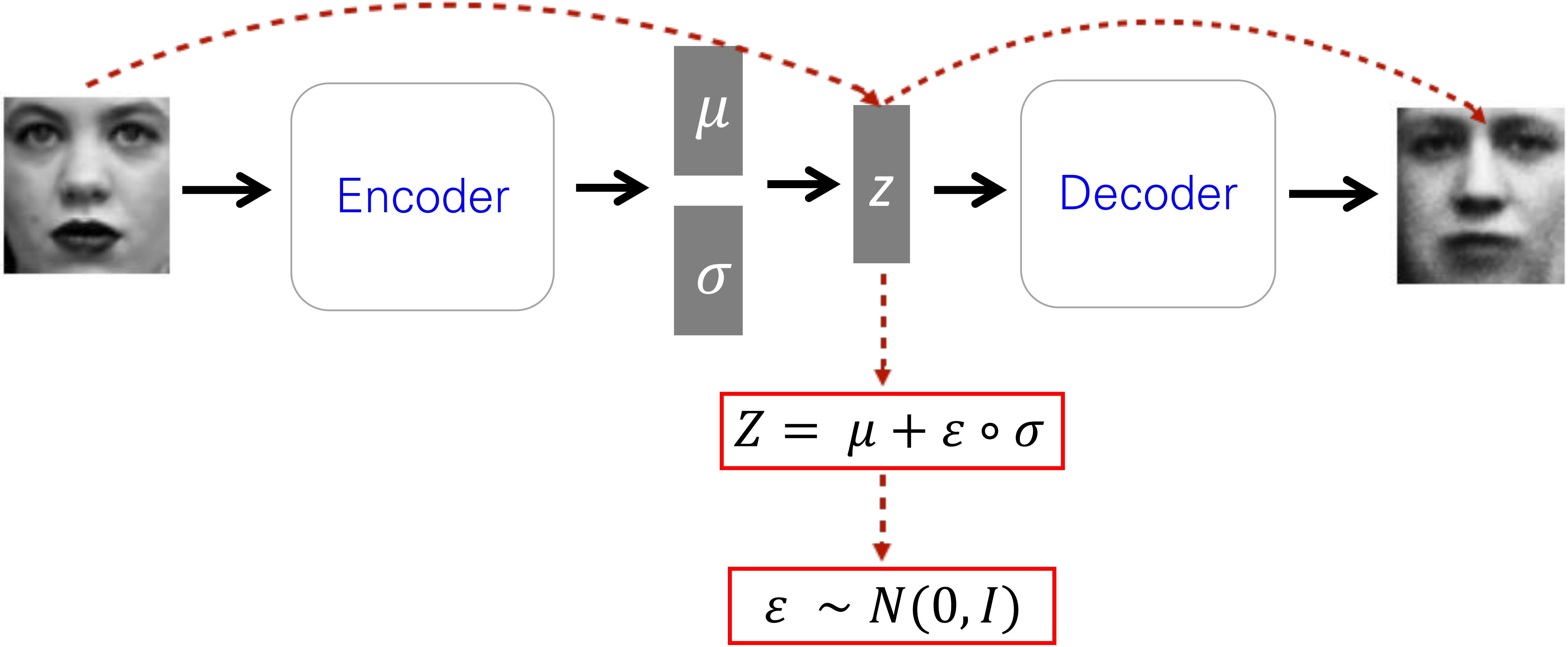


Reparametrization Trick



$$Z = \mu + \varepsilon \circ \sigma$$

Reparametrization Trick



$$Z = \mu + \varepsilon \circ \sigma$$

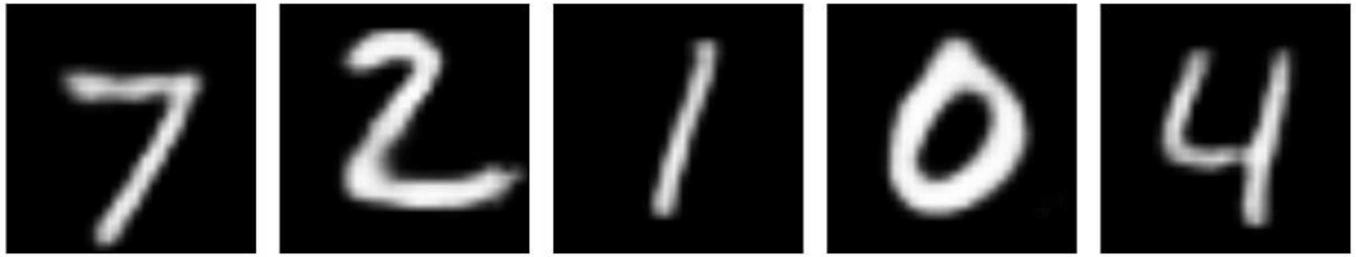
$$\varepsilon \sim N(0, I)$$

Training VAE

Traditional AE:

Input Image:

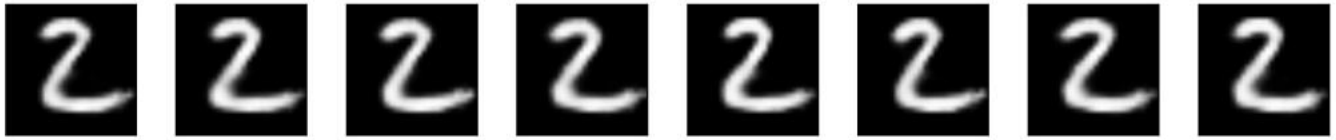
Output Images:



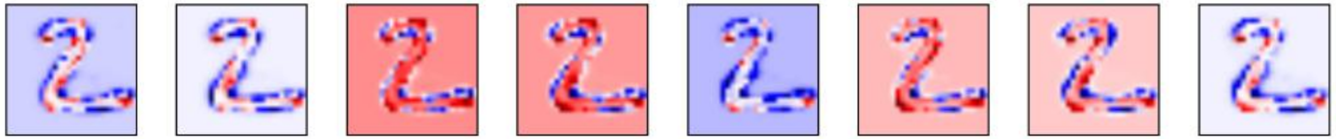
Variational AE:

Input Image:

Output Images:

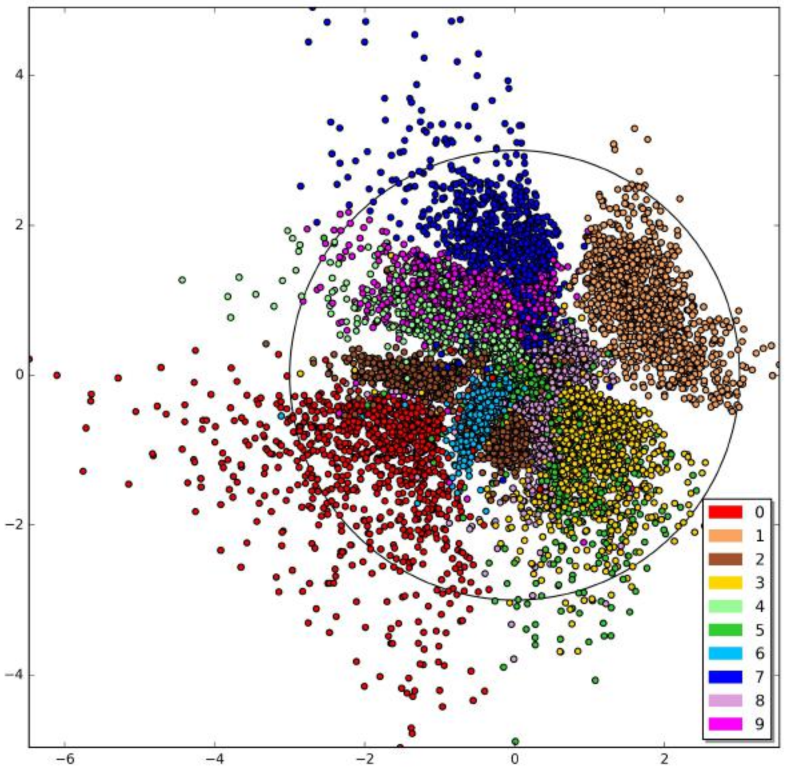


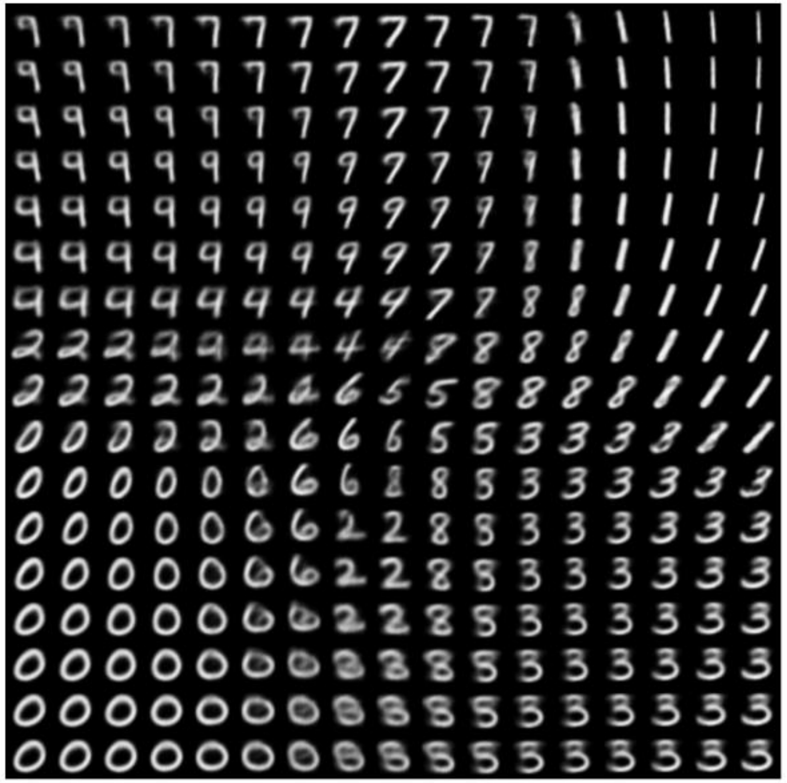
Difference:



Latent space of VAE

- More separable than AE
- Because of the prior N(0,1) everything is center at (0,0) with spread of approx 1.





Desiderata for representations

What do we want out a representation?

Many possible answers here. First, a few uncontroversial desiderata:

- **Interpretability**: if the derived features are semantically meaningful, and interpretable by a human, they can be easily evaluated.
  (e.g. noisy-OR: "features" are diseases a patient has)

  Sparsity of a representation is an important subcase: "explanatory" features for sample can be examined if there are a small number of them.

- **Downstream usability:** the features are "useful" for downstream tasks. Some examples:

  Improving label efficiency: if, for a task, a linear (or otherwise "simple") classifier can be trained on features and it works well, smaller # of labeled samples are needed.

Desiderata for representations

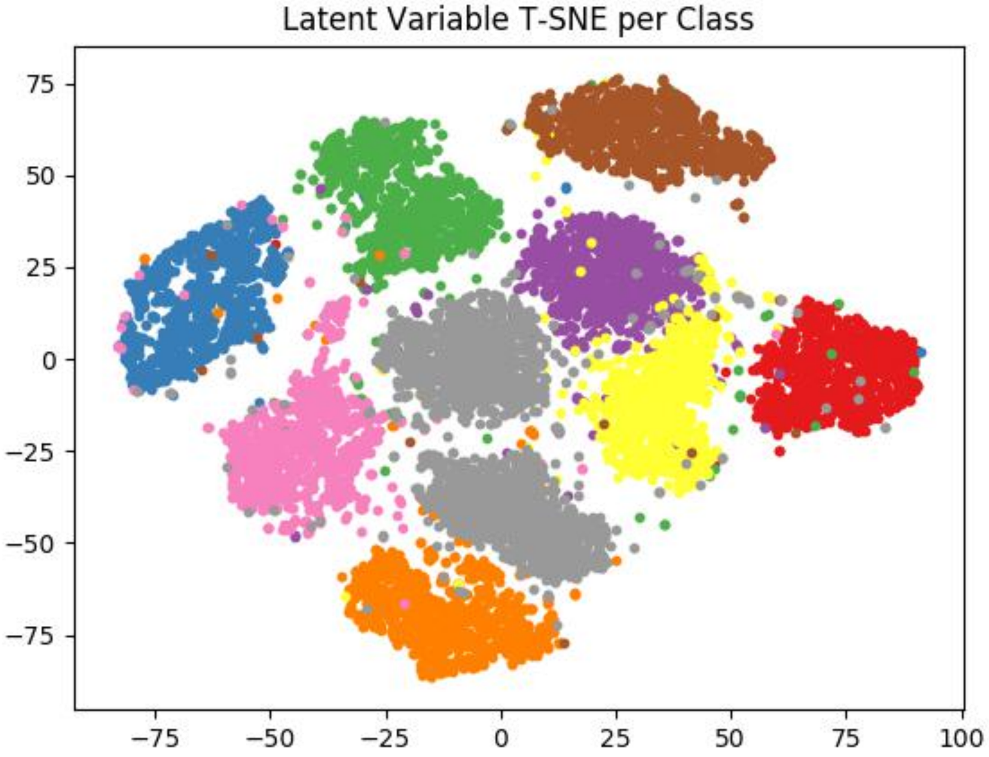
- **Obvious issue:** interpretability and “usefulness” are not easily mathematically expressed. We need some “proxies” that induce such properties.

  This is a lot more contraversial – here we survey some general desiderata, proposed as early as Bengio-Courville-Vincent ’14:

- **Hierarchy/compositionality**: video/images/text/ are expected to have hierarchical structure – depth helps induce such structure.
- **Semantic clusterability**: features of the same ”semantic class” (e.g. images in the same category) are clustered.
- **Linear interpolation**: in representation space, linear interpolations produce meaningful data points (i.e. ”latent space is convex”). Sometimes called manifold flattening.
- **Disentangling**: features capture “independent factors of variation” of data. (Bengio-Courville-Vincent ’14). Has been very popular in modern unsupervised learning, though many potential issues with it.

Semantic clustering

- **Semantic clusterability:** features of the same “semantic class” (e.g. images in the same category) are clustered together.



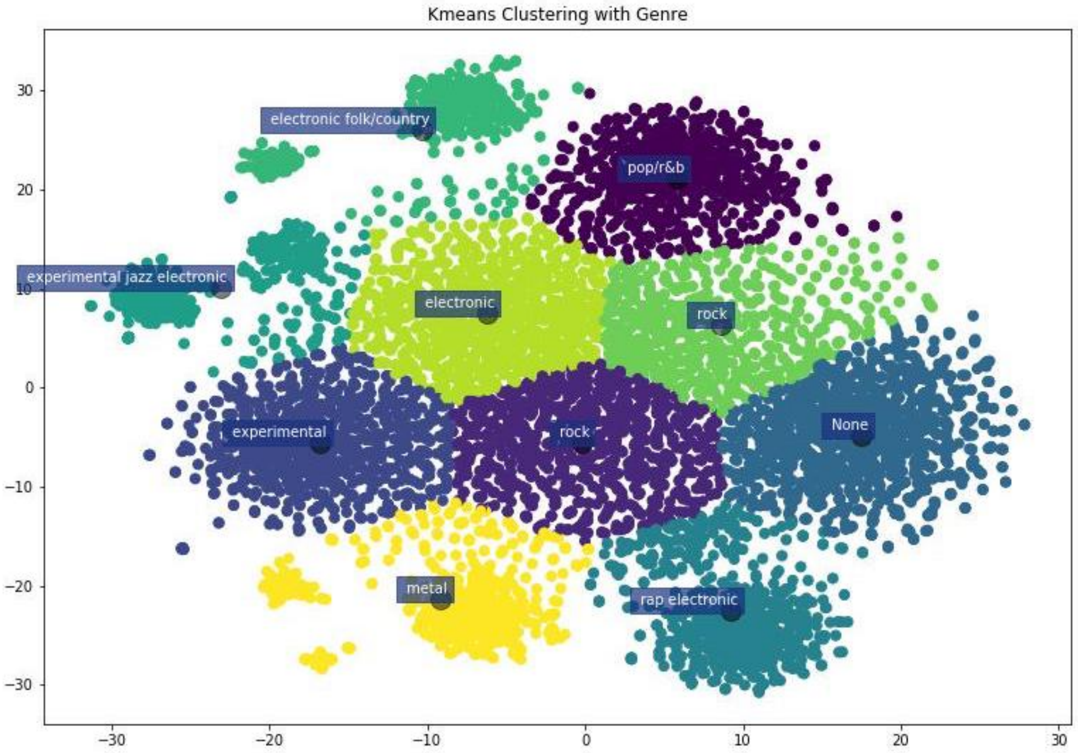
The intuition:

If semantic classes are linearly (or other simple function) separable, and labels on downstream tasks depend linearly on semantic classes – can afford to learn a simple classifier!!

t-SNE projection of VAE-learned features of the 10 MNIST classes.
Image from https://pyro.ai/examples/vae.html

Semantic clustering

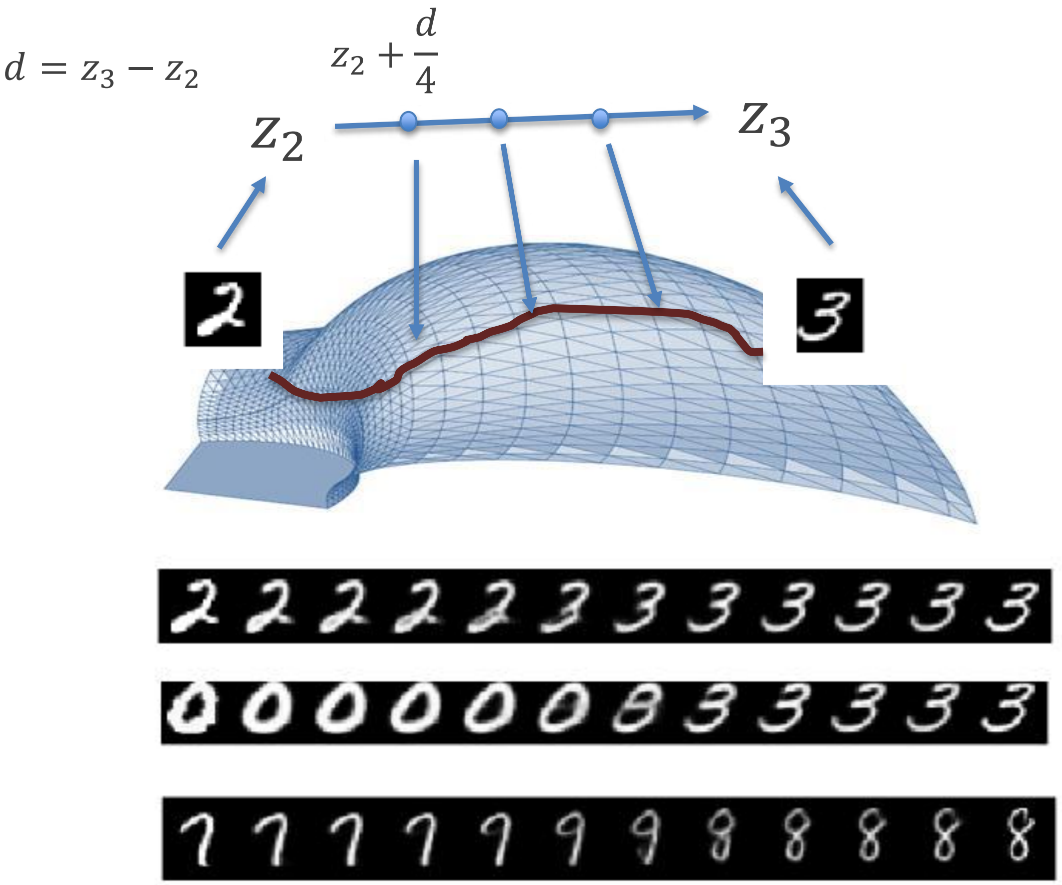
- **Semantic clusterability:** features of the same "semantic class" (e.g. images in the same category) are clustered together.



t-SNE projection of word embeddings for artists (clustered by genre). Image from https://medium.com/free-code-camp/learn-tensorflow-the- word2vec-model-and-the-tsne-algorithm-using-rock-bands-97c99b5dcb3a

Linear interpolation

- **Linear interpolation:** in representation space, linear interpolations produce meaningful data points. (i.e. "latent space is convex")



Interpolations for a VAE trained on MNIST.

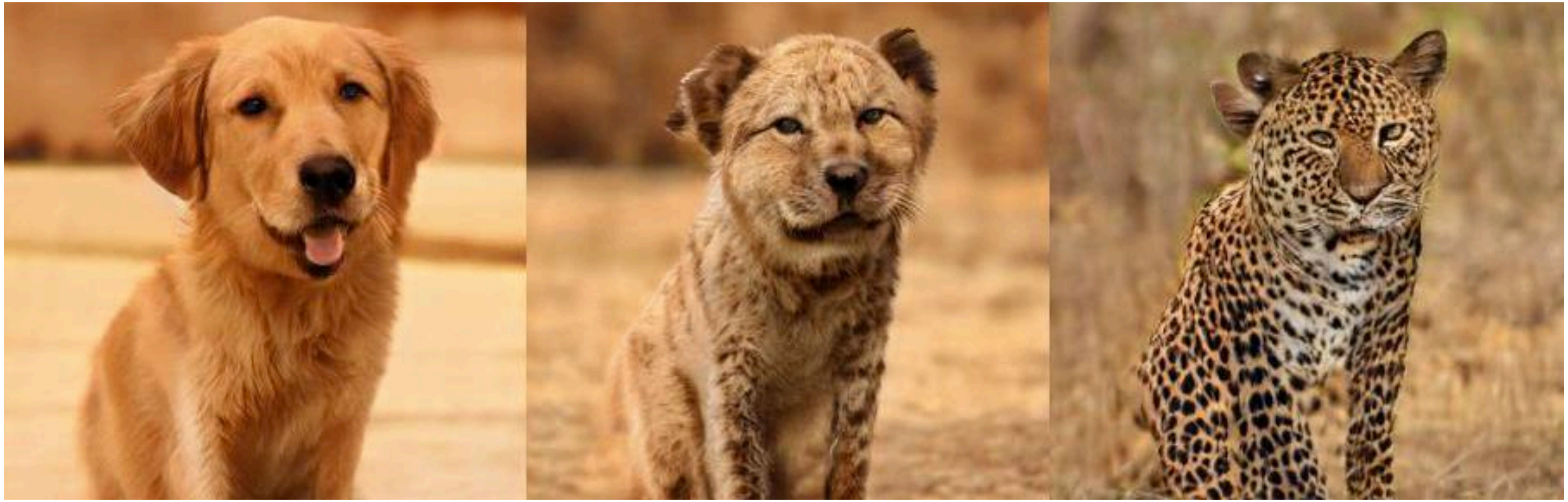
The intuition:

The data manifold is complicated/curved.

The latent variable manifold is a convex set – moving in straight lines keeps us on it.

Linear interpolation

- **Linear interpolation:** in representation space, linear interpolations produce meaningful data points. (i.e. “latent space is convex”)



Interpolations for a BigGAN, image from
https://thegradient.pub/bigganex-a-dive-into- the-latent-space-of-biggan/

Disentangled representations

- **Disentangling**: features capture "independent factors of variation" of data. (Bengio-Courville-Vincent '14).
- For concreteness, let's assume that we have a latent variable model for data with latent variables $\mathbf{z}$, observables $\mathbf{x}$, and joint distribution $p_\theta(\mathbf{z}, \mathbf{x})$
- There are (at least) two ways to formalize this.

Prior disentangling: is a product distribution, i.e. $\boldsymbol{p_\theta}(\mathbf{z}) = \Pi_i \boldsymbol{p_\theta}(\boldsymbol{z_i})$

Classical example: ICA (independent component analysis)

Posterior disentangling: fit a variational posterior $\boldsymbol{q_\theta}$ s.t. $\boldsymbol{q_\theta}(\mathbf{z}|\boldsymbol{x})$ is (on average over $\mathbf{x}$) a product distribution

In other words $\int_x \boldsymbol{q_\theta}(\boldsymbol{z}|\boldsymbol{x})\boldsymbol{p}(\boldsymbol{x})d\boldsymbol{x}$ -- usually called the aggregate posterior – is close to a product distribution.

Disentangled representations

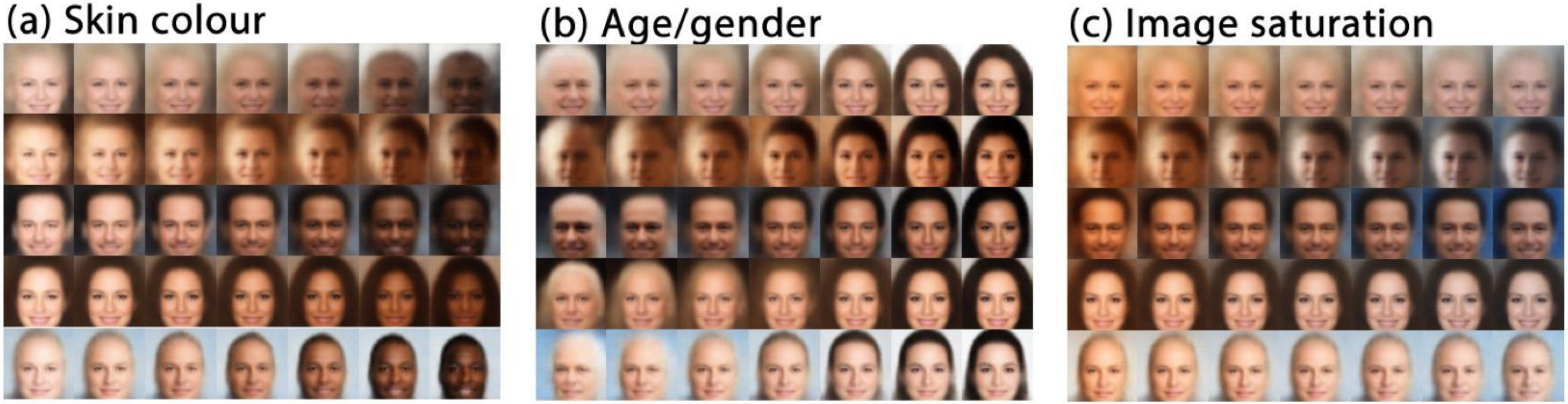


Figure 4: **Latent factors learnt by β-VAE on celebA:** traversal of individual latents demonstrates that β-VAE discovered in an unsupervised manner factors that encode skin colour, transition from an elderly male to younger female, and image saturation.

- Posterior disentangling in β-VAE. To produce plots, infer latent variable for an image, then change a single latent variable gradually.

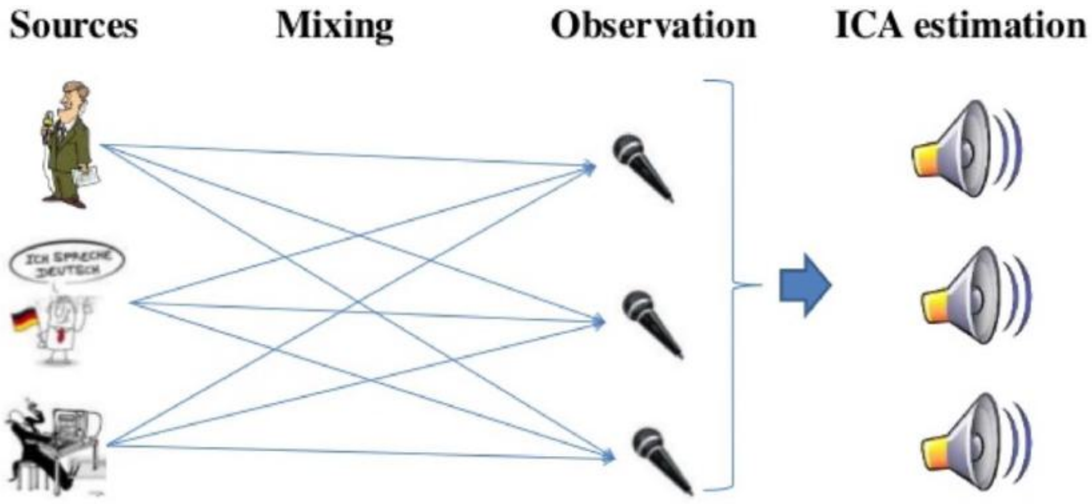
Irina Higgins et al. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. ICLR 2017.

Prior disentangling

- **Prior disentangling:** $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is a product distribution, i.e. $p_{\boldsymbol{\theta}}(\boldsymbol{z}) = \Pi_i p_{\boldsymbol{\theta}}(\boldsymbol{z_i})$

Classical example: ICA (independent component analysis), also called the "cocktail party problem".

Assume data is generated as



If z has an independent, non-Gaussian prior, model is identifiable and efficiently learnable. (See, e.g. Frieze-Jerum-Kannan '96, Anandkumar et al '12)

Other examples: noisy-OR networks (diseases are independent), general Bayesian nets, viewing top variables as z's, GANs, …

Posterior disentanglement in VAEs

- Recall the “regularization” view of the VAEs objective:

$$\underbrace{\Sigma_x \mathbb{E}_{q(h^L|x)} \log p(x|h^L)}_{\text{"Reconstruction" error}} - \underbrace{KL(q(h^L|x)||p(h^L))}_{\text{"Regularization towards prior"}}$$

- Consider a prior which is a product distribution (e.g. standard Gaussian): The KL term implicitly penalizes distributions for which

$$\sum_x KL(q(h^L|x)||p(h^L)) \approx \mathbb{E}_{x\sim p^*} KL(q(h^L|x)||p(h^L))$$

is large – i.e. the aggregated posterior is far from a product distribution

Posterior disentanglement in VAEs

- Recall the “regularization” view of the VAEs objective:

$$\underbrace{\Sigma_x \mathbb{E}_{q(h^L|x)} \log p(x|h^L)}_{\text{"Reconstruction" error}} - \underbrace{KL(q(h^L|x)||p(h^L))}_{\text{"Regularization towards prior"}}$$

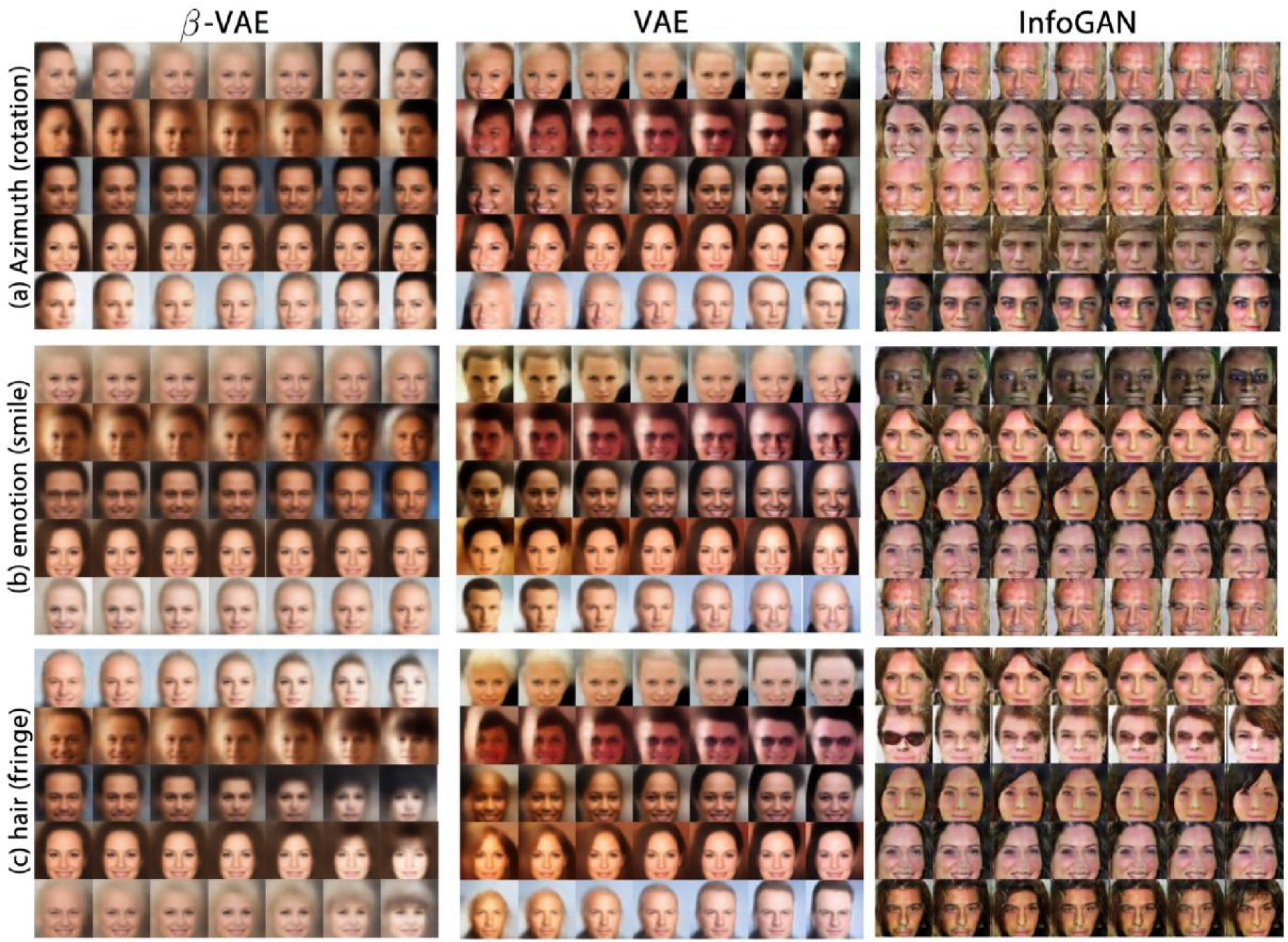
The KL term implicitly penalizes distributions for which

$$\sum_x KL(q(h^L|x)||p(h^L)) \approx \mathbb{E}_{x\sim p^*} KL(q(h^L|x)||p(h^L))$$

The idea of Higgins et al ’17 introduce a “weighting” factor to put more weight on reconstruction or disentanglement:

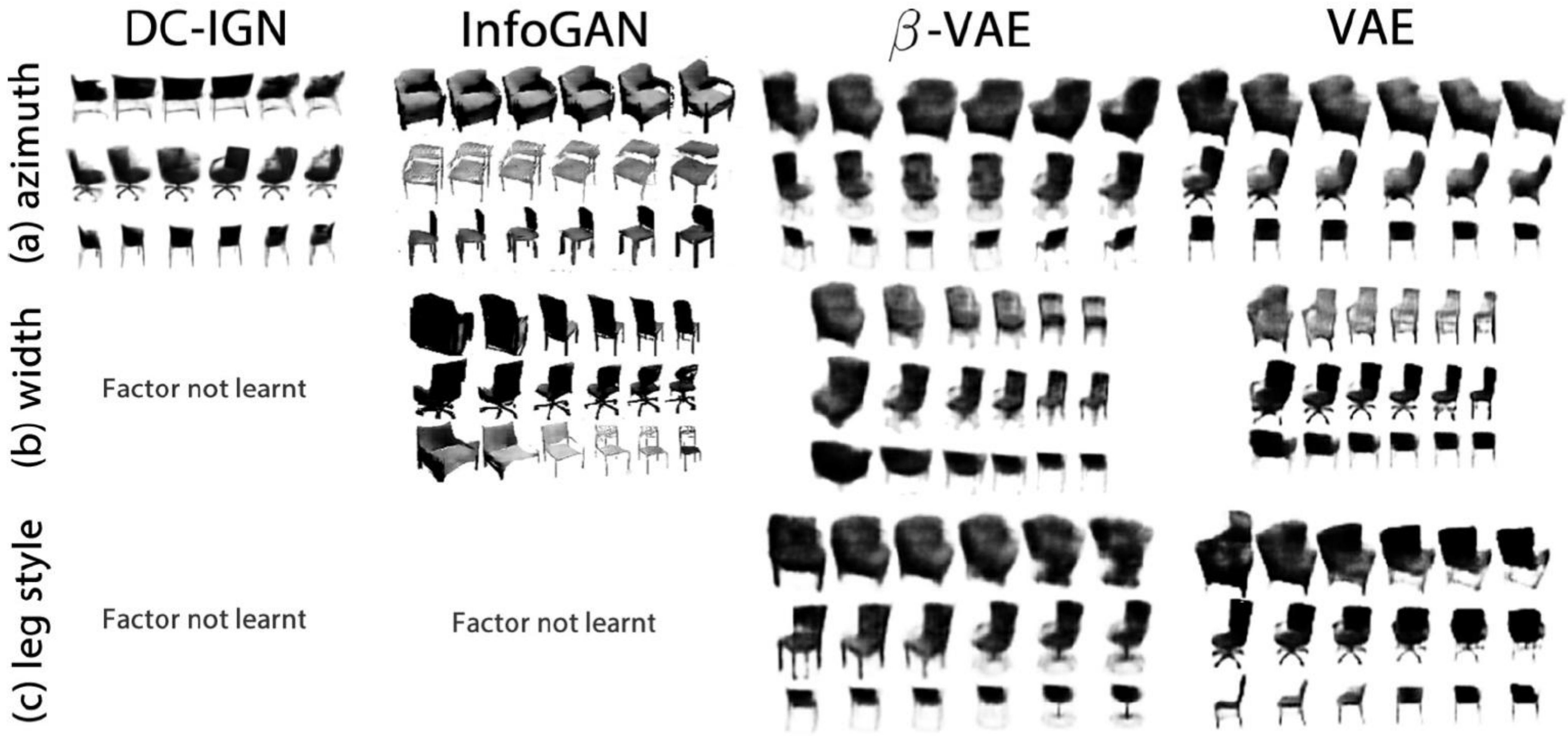
β-VAE objective: $\sum_x \mathbb{E}_{q(h^L|x)} \log p(x|h^L) - \beta KL(q(h^L|x)\|p(h^L))$

Posterior disentanglement in VAEs



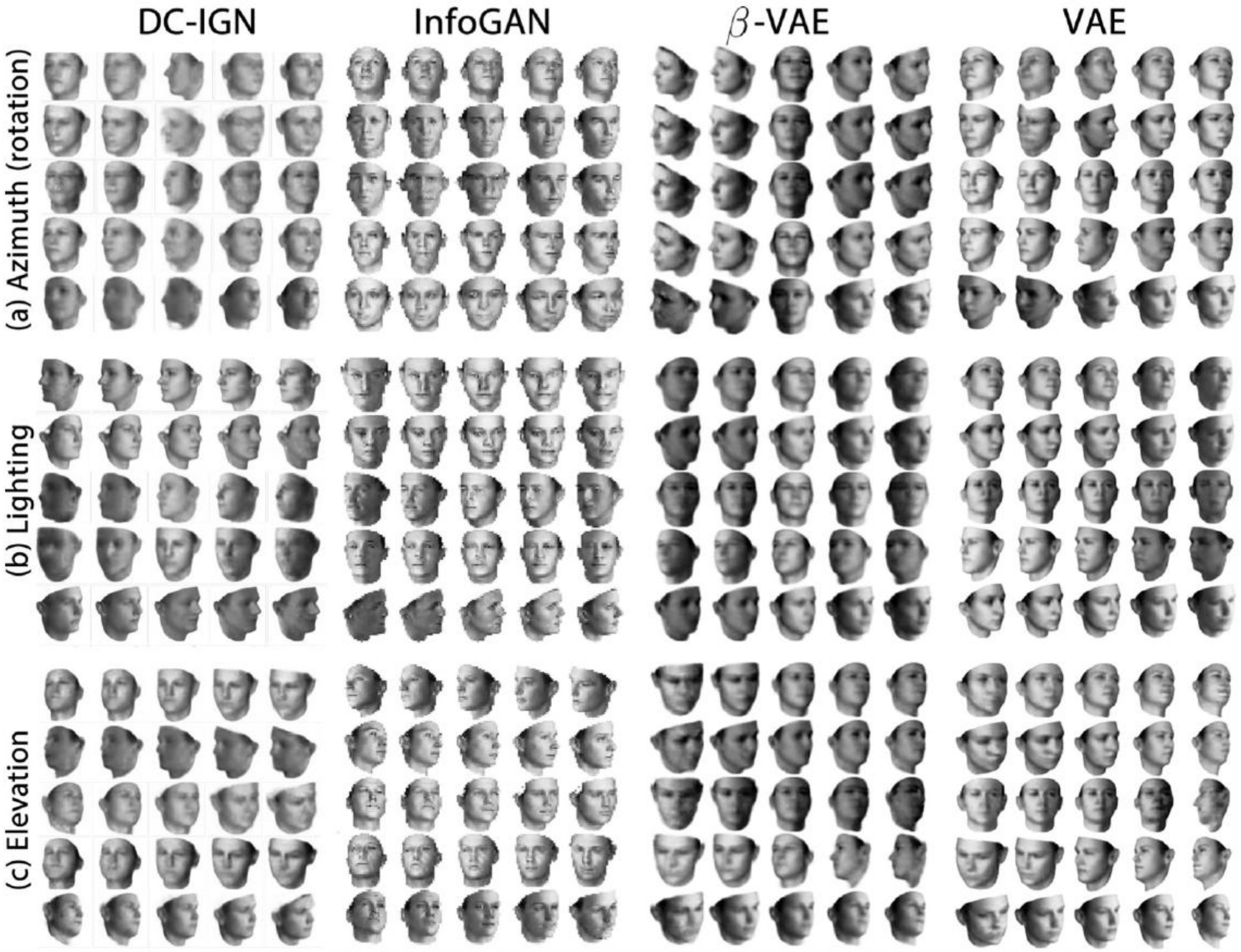
Irina Higgins et al. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. ICLR 2017.

Posterior disentanglement in VAEs



Irina Higgins et al. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. ICLR 2017.

Posterior disentanglement in VAEs



Irina Higgins et al. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. ICLR 2017.

Measuring disentanglement

- Metrics are typically defined assuming access to a dataset with K “ground-truth” variation factors.

BetaVAE metric: based on "linear separability" of factors

```
Generate a **training set** of samples as follows:
  Sample a **batch** of B samples as follows:
    Pick a **ground-truth variation factor k** uniformly at random from [K].
    Generate two sets of “ground truth” latent factors, $\mathbf{v}_1, \mathbf{v}_2 \in R^K$, s.t.
    $(\mathbf{v}_1)_k = (\mathbf{v}_2)_k$ , and other coords are independently, randomly sampled.
    Generate **images** $\mathbf{x}_1, \mathbf{x}_2$ from $\mathbf{v}_1, \mathbf{v}_2$.
    Infer latent vars $\mathbf{z}_1, \mathbf{z}_2$ using model we are evaluating. (e.g. encoder in VAE)
Calculate average $\mathbf{z}_{avg}$ of $| \mathbf{z}_1 - \mathbf{z}_2 |$ in batch, add $(\mathbf{z}_{avg}, k)$ to training set.
Train linear predictor on training set, evaluate it’s test performance.
```

Measuring disentanglement

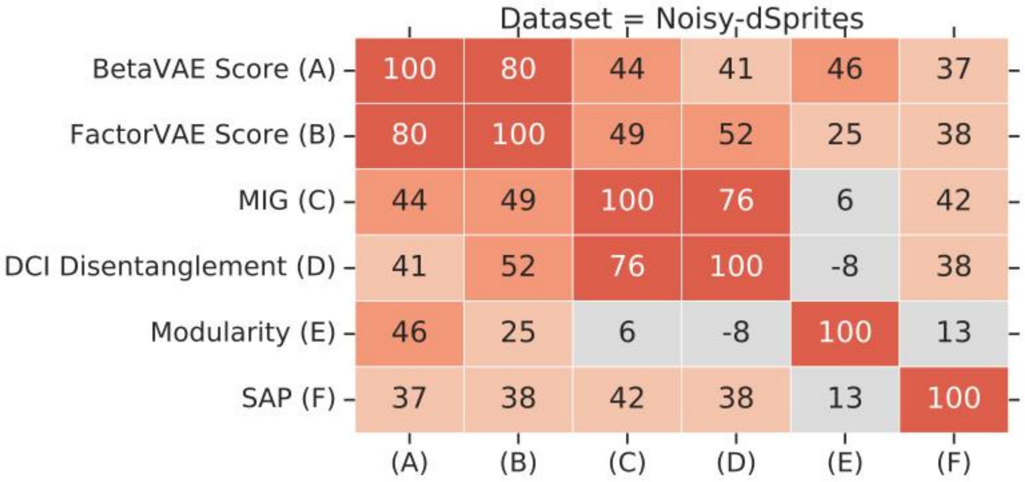
BetaVAE metric: based on "linear separability" of factors

```
Generate a training set of samples as follows:
  Sample a batch of B samples as follows:
    Pick a ground-truth variation factor k uniformly at random from [K].
    Generate two sets of "ground truth" latent factors, $\mathbf{v}_1$, $\mathbf{v}_2 \in R^K$, s.t.
    $(\mathbf{v}_1)_k = (\mathbf{v}_2)_k$ , and other coords are independently, randomly sampled.
    Generate images $\mathbf{x}_1$, $\mathbf{x}_2$ from $\mathbf{v}_1$, $\mathbf{v}_2$.
    Infer latent vars $\mathbf{z}_1$, $\mathbf{z}_2$ using model we are evaluating. (e.g. encoder in VAE)
Calculate average $\mathbf{z}_{avg}$ of $|\mathbf{z}_1 - \mathbf{z}_2|$ in batch, add $(\mathbf{z}_{avg}, k)$ to training set.
Train linear predictor on training set, evaluate it's test performance.
```

- Intuition: averaging should make coords in $\mathbf{z}_{avg}$ different from k smaller, thus linear classifier should "focus" on k.
- Many variants of this exist. (e.g. FactorVAE, mutual information gap, etc.)

Measuring disentanglement

- Locatello et al ’19, “Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations” (Best paper award ar ICML’19): A large-scale study of disentanglement measures, as well as gen. models.

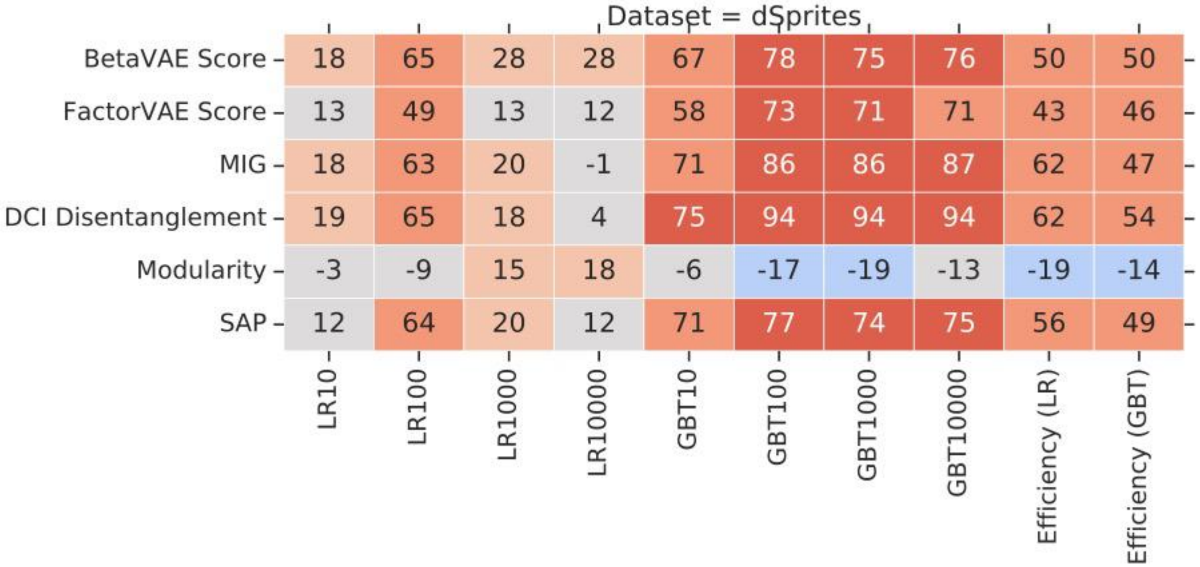


	(A)	(B)	(C)	(D)	(E)	(F)
BetaVAE Score (A)	100	80	44	41	46	37
FactorVAE Score (B)	80	100	49	52	25	38
MIG (C)	44	49	100	76	6	42
DCI Disentanglement (D)	41	52	76	100	-8	38
Modularity (E)	46	25	6	-8	100	13
SAP (F)	37	38	42	38	13	100

Figure 2. Rank correlation of different metrics on Noisy-dSprites. Overall, we observe that all metrics except Modularity seem mildly correlated with the pairs BetaVAE and FactorVAE, and MIG and DCI Disentanglement strongly correlated with each other.

Usefulness of disentanglement?

- Downstream classification task: predict true ground-truth factors (w/ multiclass logistic regression)
- Careful to extrapolate too much – task/setup is a little contrived.



Dataset = dSprites

	LR10	LR100	LR1000	LR10000	GBT10	GBT100	GBT1000	GBT10000	Efficiency (LR)	Efficiency (GBT)
BetaVAE Score	18	65	28	28	67	78	75	76	50	50
FactorVAE Score	13	49	13	12	58	73	71	71	43	46
MIG	18	63	20	-1	71	86	86	87	62	47
DCI Disentanglement	19	65	18	4	75	94	94	94	62	54
Modularity	-3	-9	15	18	-6	-17	-19	-13	-19	-14
SAP	12	64	20	12	71	77	74	75	56	49

Figure 5. Rank correlations between disentanglement metrics and downstream performance (accuracy and efficiency) on dSprites.

Locatello et al. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. ICML 2019.

Usefulness of disentanglement?

- Statistical efficiency measure: average accuracy based on 100 samples divided by the average accuracy based on 10,000 samples

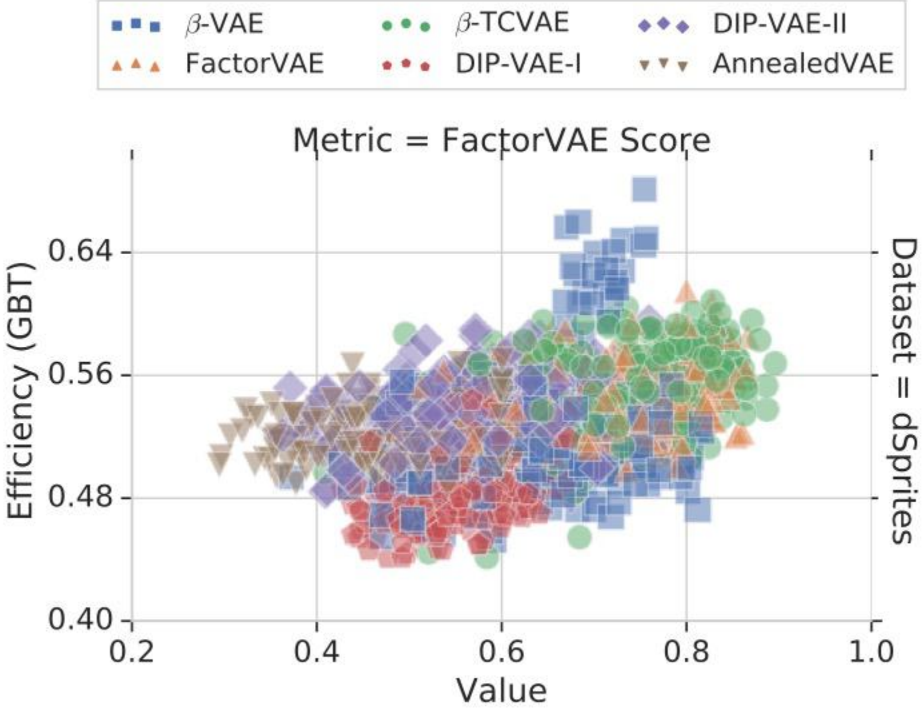


Figure 6. Statistical efficiency of the FactorVAE Score for learning a GBT downstream task on dSprites.

Locatello et al. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. ICML 2019.

Issue of ill-posedness?

- Similar issues plague disentangling that do "flat minima": a model can be re-parametrized, s.t. the distribution over the data is unchanged, but it can be arbitrarily more "entangled".
- Thus, **some kind of inductive bias both on model class and data seems necessary**.
- As a simple example: consider. $\mathbf{z} \sim \mathcal{N}(0, \boldsymbol{I})$, let $\mathbf{z}' = \boldsymbol{U}\mathbf{z}$, for any non-identity orthogonal matrix U.
- Then, under any "intuitive" understanding of entangling, $\mathbf{z}'$ seems entangled with $\mathbf{z}$ – small changes of coordinates of z cause global changes in $\mathbf{z}'$.

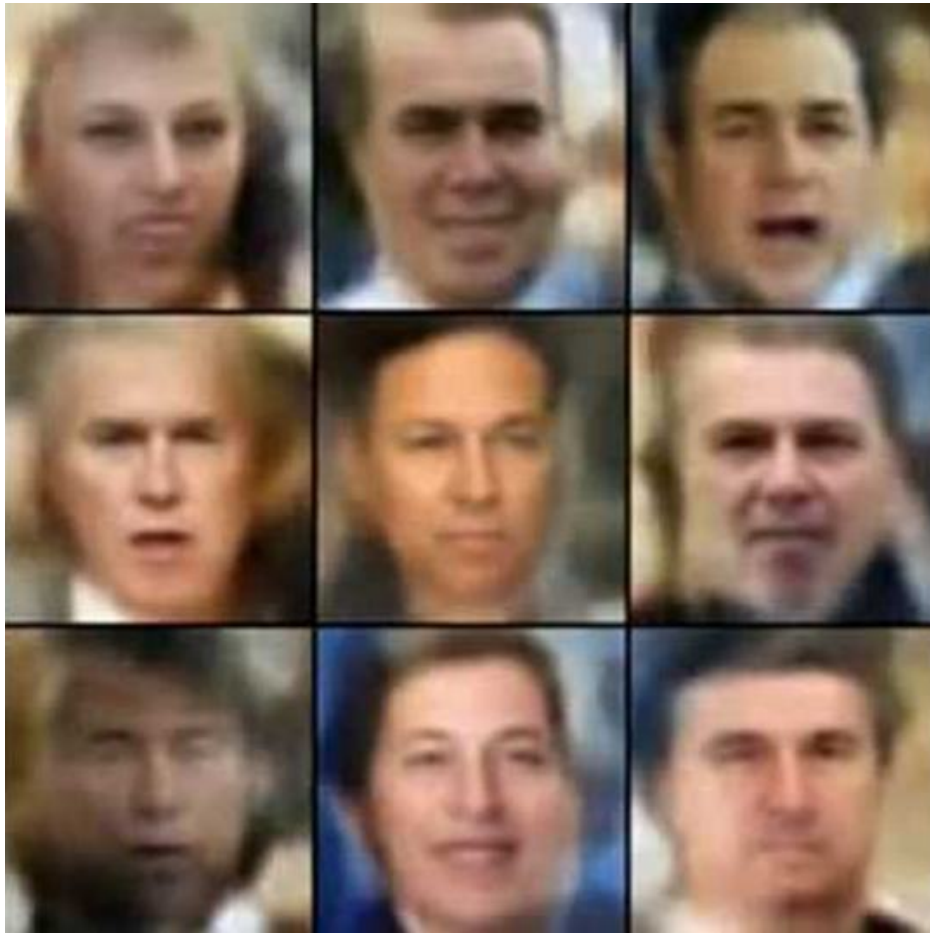
Locatello et al. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. ICML 2019.

Today

- variational autoencoders (VAEs)
- **vector quantized VAEs (VQ-VAEs)**
- denoising diffusion models

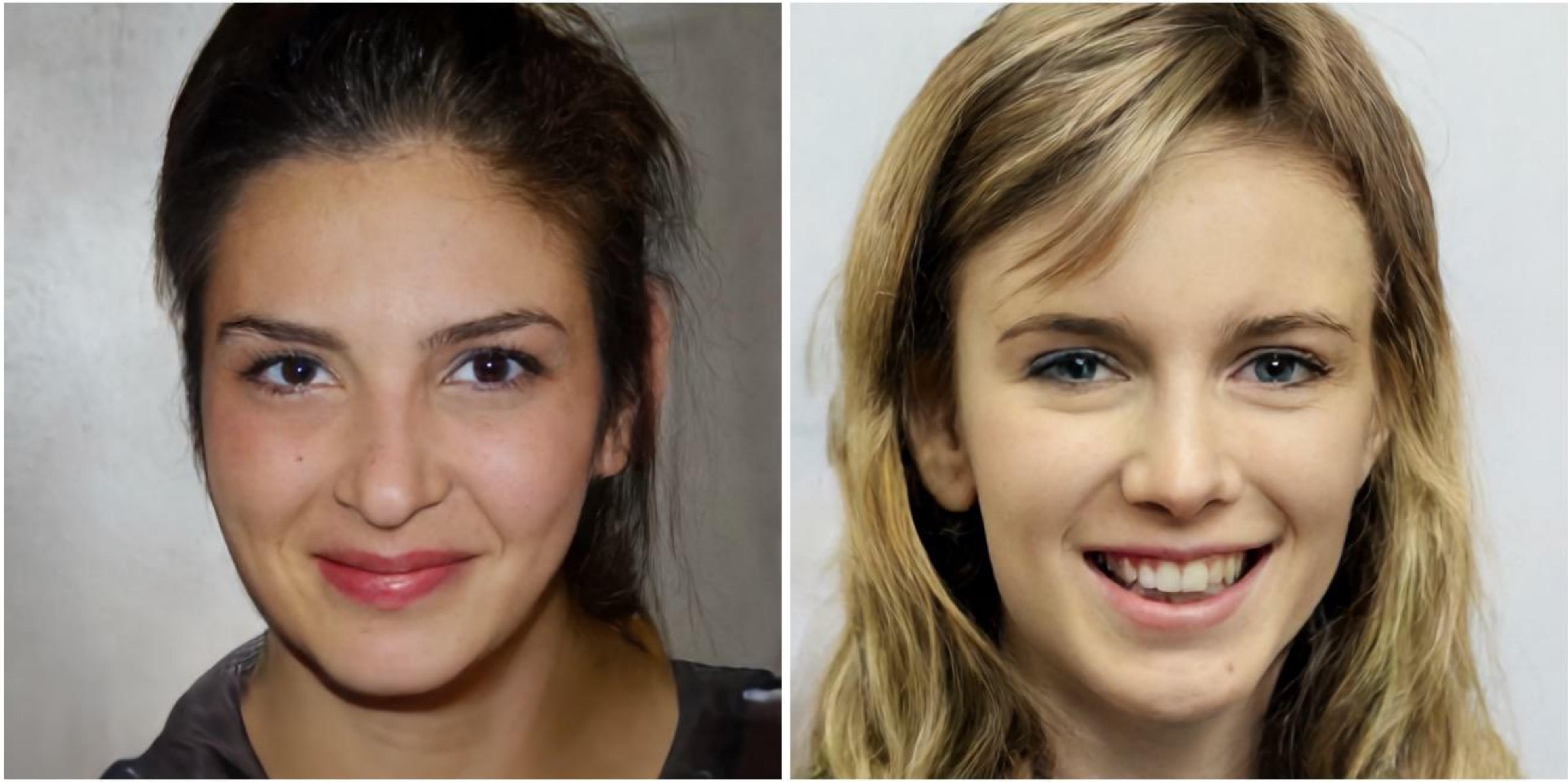
Gaussian VAEs 2013

Sample $z \sim \mathcal{N}(0, I)$ and compute $y_\Phi(z)$



[Alec Radford]

Vector Quantized VAEs (VQ-VAE) 2019



VQ-VAE-2, Razavi et al., NeurIPS 2019

Vector Quantized VAEs (VQ-VAE) 2019

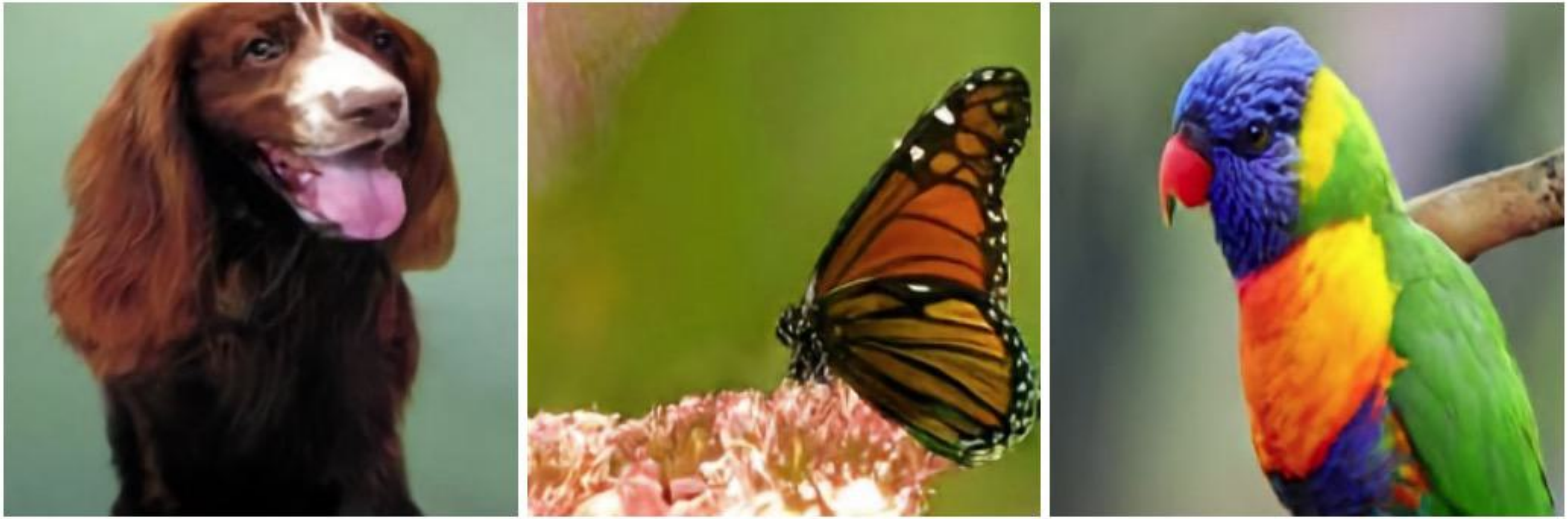
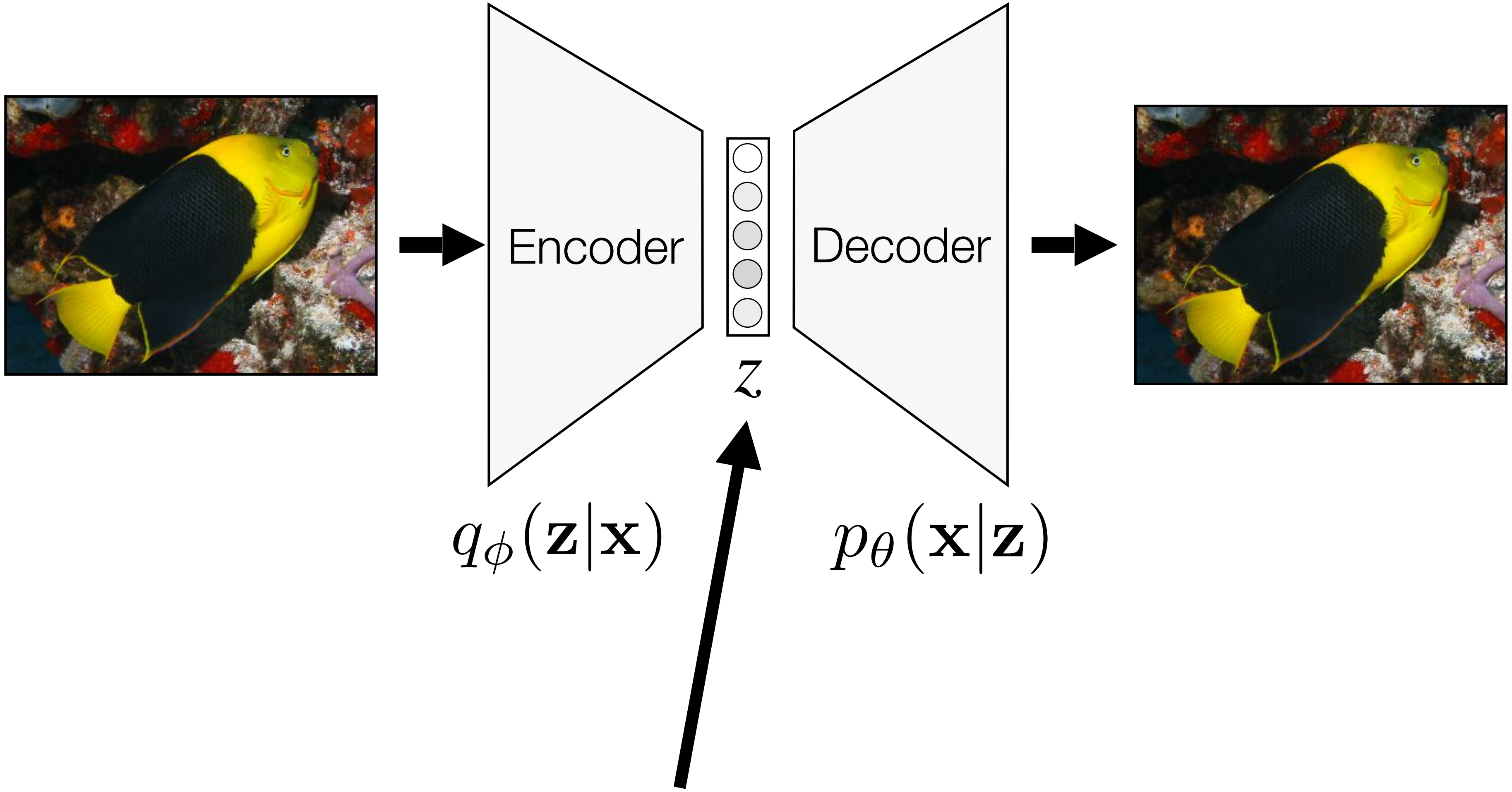


Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

VQ-VAE-2, Razavi et al., NeurIPS 2019

Vector Quantized VAEs


Encoder
Decoder
z
$q_\phi(\mathbf{z}|\mathbf{x})$
$p_\theta(\mathbf{x}|\mathbf{z})$

- replace latent **z** vector with an autoregressive model

Discrete codes (Symbols)

- Autoregressive models meets variational autoencoders
- VAEs usually use a continuous representation for latent code, **z**
- But many events in the world are discrete.
  - e.g. can sometimes describe images concisely as collection of objects
- Key idea: replace Gaussian code with categorical code

Learning

- Two stages:

1. Train the VQ-VAE
2. Learn a better prior, p(**z**)

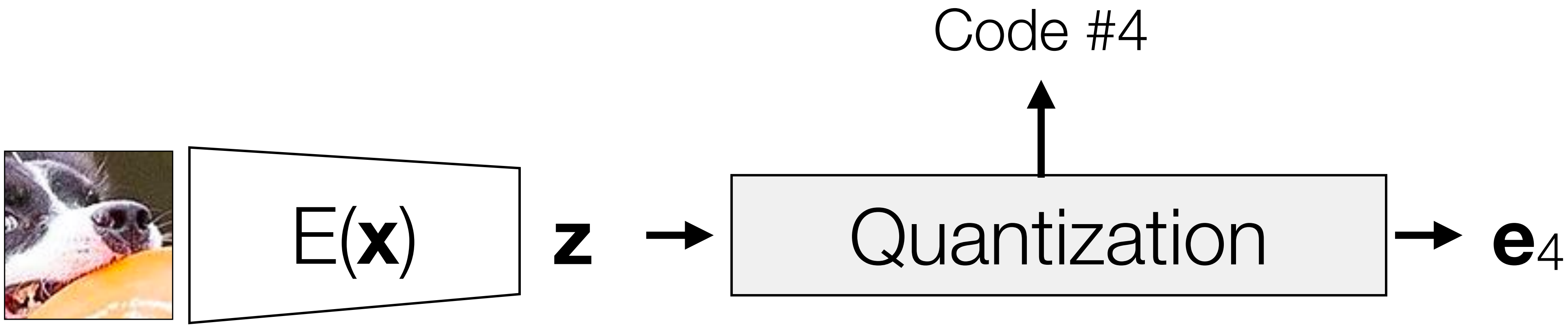
Learning

- Two stages:

1. **Train the VQ-VAE**
2. Learn a better prior, p(**z**)

Vector quantization

- Predict a real-valued vector, like in an ordinary VAE. Then, “snap” it to a nearest neighbor from a codebook



Code #4

E(**x**)

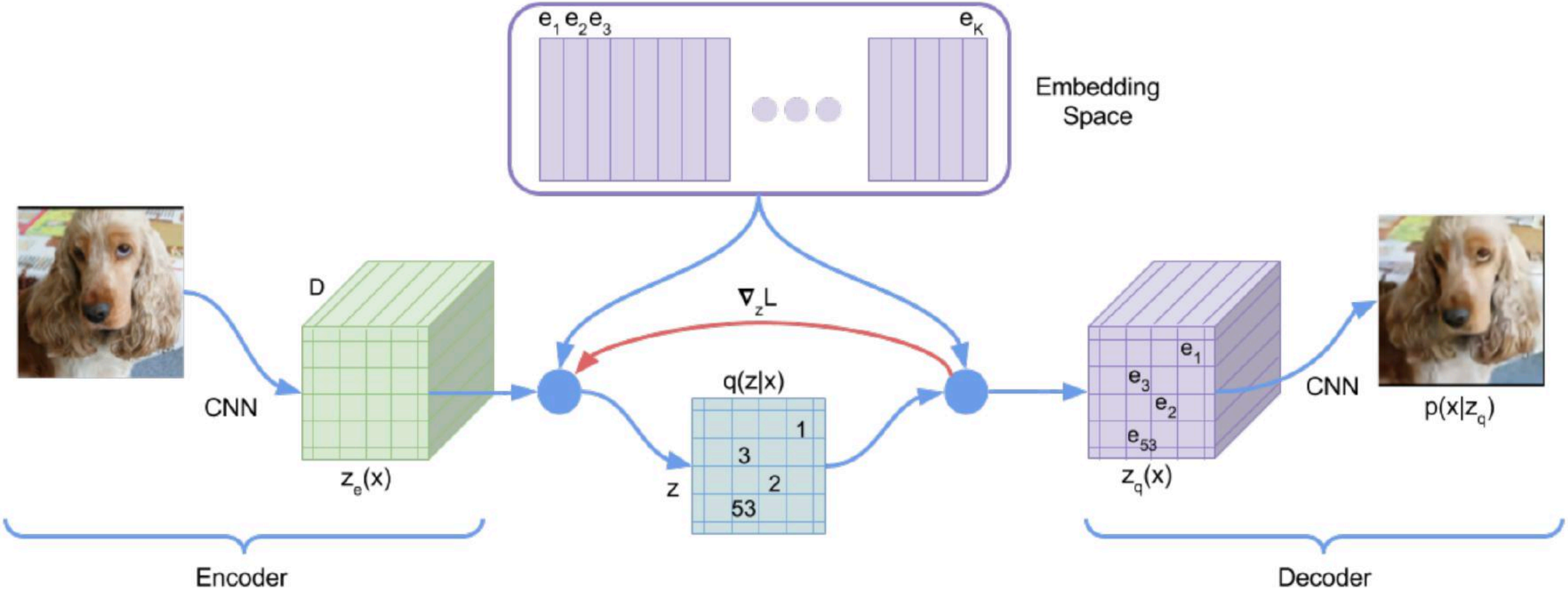
z

Quantization

$\mathbf{e}_4$

$$\text{Quantize}(E(\mathbf{x})) = \mathbf{e}_k \quad \text{where } k = \arg\min_j ||E(\mathbf{x}) - \mathbf{e}_j||$$

VQ-VAE



e_1 e_2 e_3
e_K
Embedding Space
D
$\nabla_z L$
CNN
$z_e(x)$
q(z|x)
1
3
2
53
z
e_1
e_3
e_2
e_{53}
$z_q(x)$
CNN
$p(x|z_q)$
Encoder
Decoder

Training the VQ-VAE

$$\mathcal{L}(\mathbf{x}, D(\mathbf{e})) = ||\mathbf{x} - D(\mathbf{e})||_2^2 + ||sg[E(\mathbf{x})] - \mathbf{e}||_2^2 + \beta||sg[\mathbf{e}] - E(\mathbf{x})||_2^2$$

where:

- **E(x)** is the encoder and **D(e)** is the decoded image
- **e** is the quantized code for image patch **x**
- sg[x] is "stop gradient" a.k.a. "detach" and β is a constant

Backprop through vector quantization

- Hard to run backprop through this:

$$\hat{\mathbf{x}} = D(\text{quantize}(E(\mathbf{x}))$$

- Why?

$$\nabla_u \text{quantize}(\text{u}) = 0$$

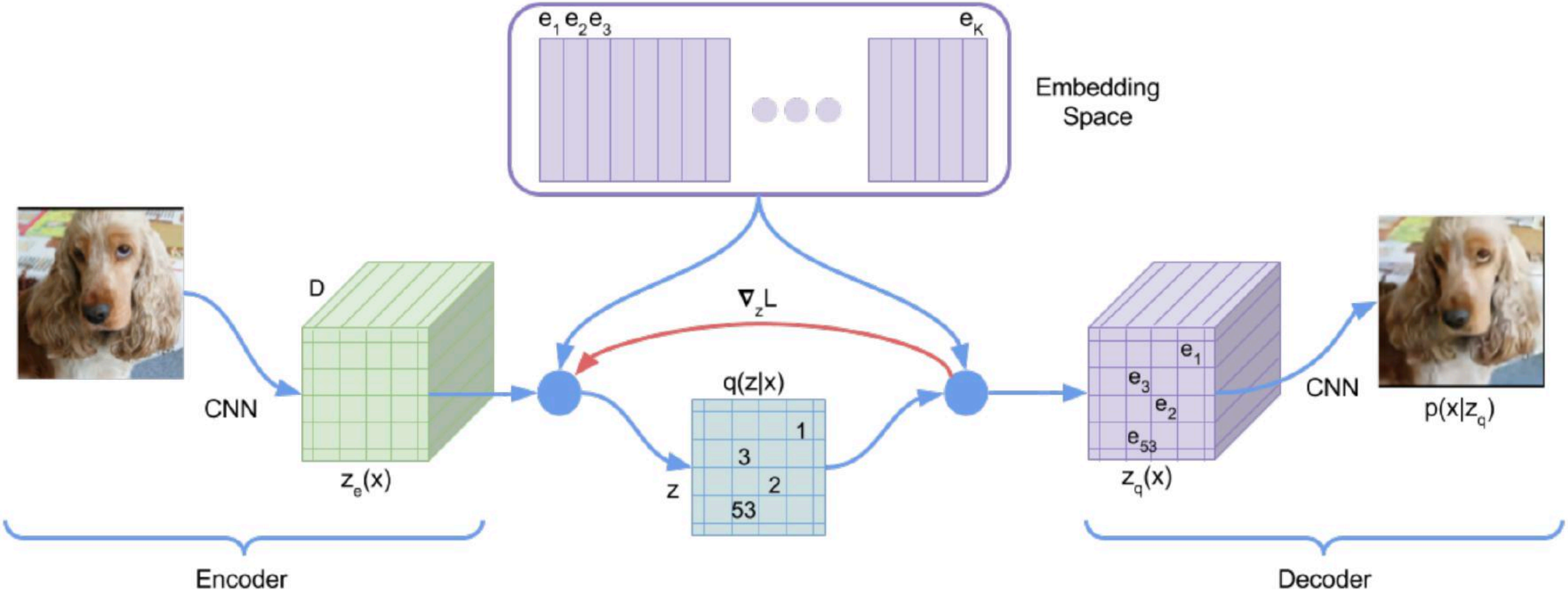
Trick: use the **straight-through estimator**. Pretend $\text{quantize}(\,.\,)$ is the identity during backwards pass!

Learning

- Two stages:

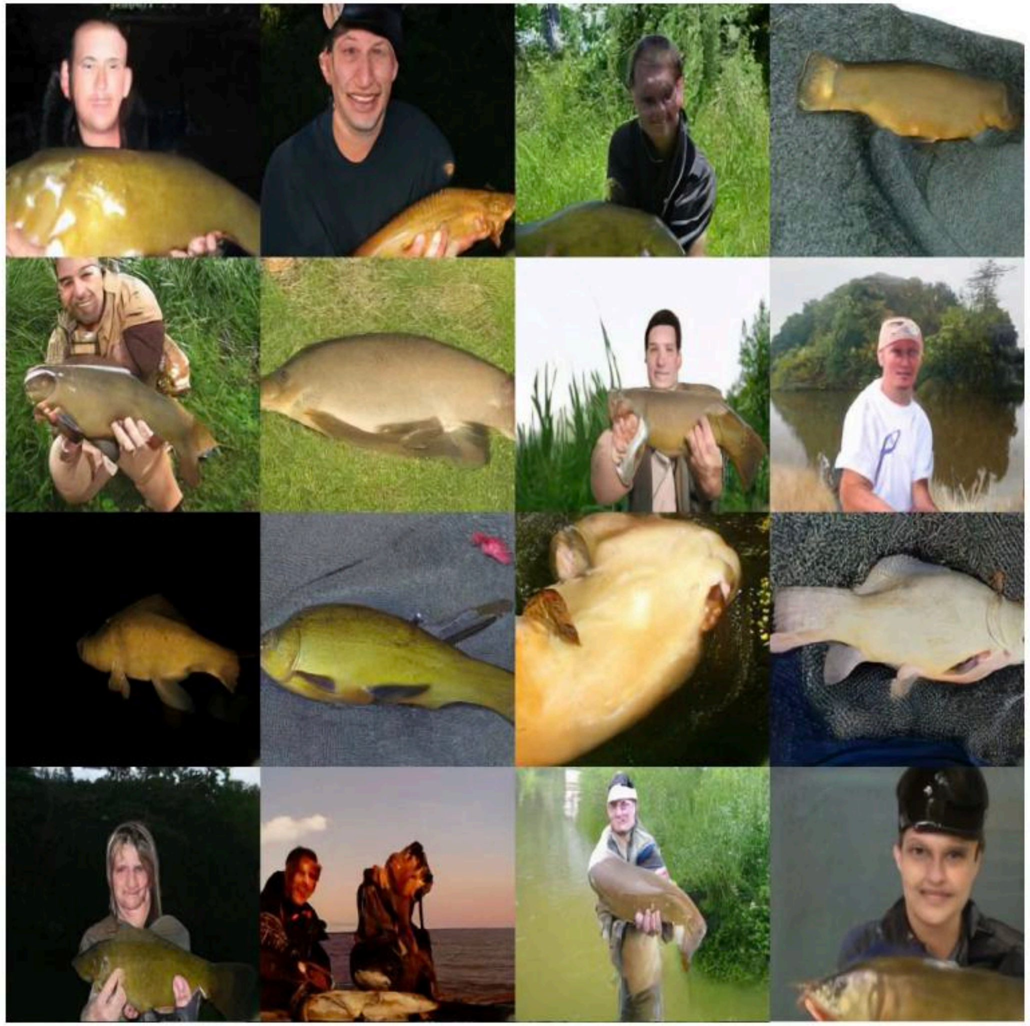
1. Train the VQ-VAE
2. **Learn a better prior, p(z)**

How do we sample **z**?

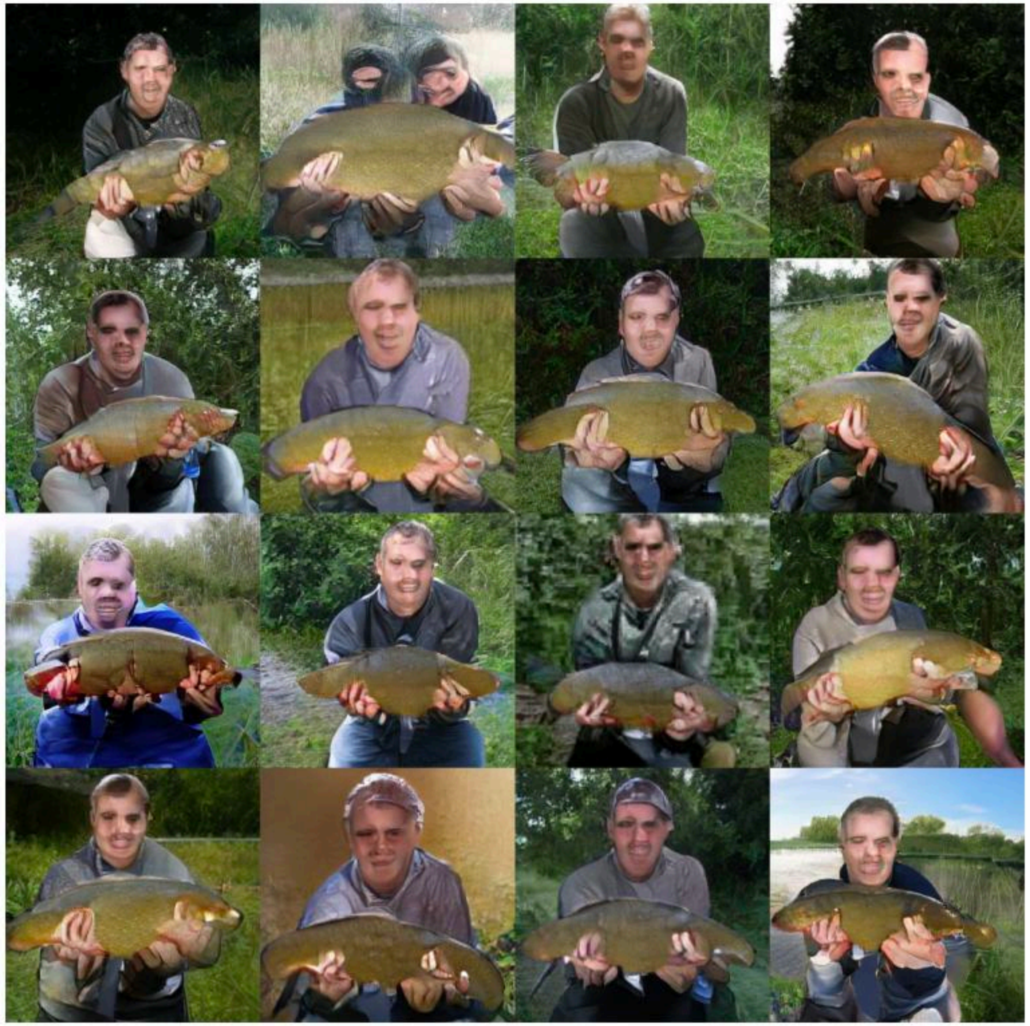


- The grid of codes, **z**, is a lot like a very tiny image.
- Fit an autoregressive model to it!

VQ-VAE vs. BigGAN deep

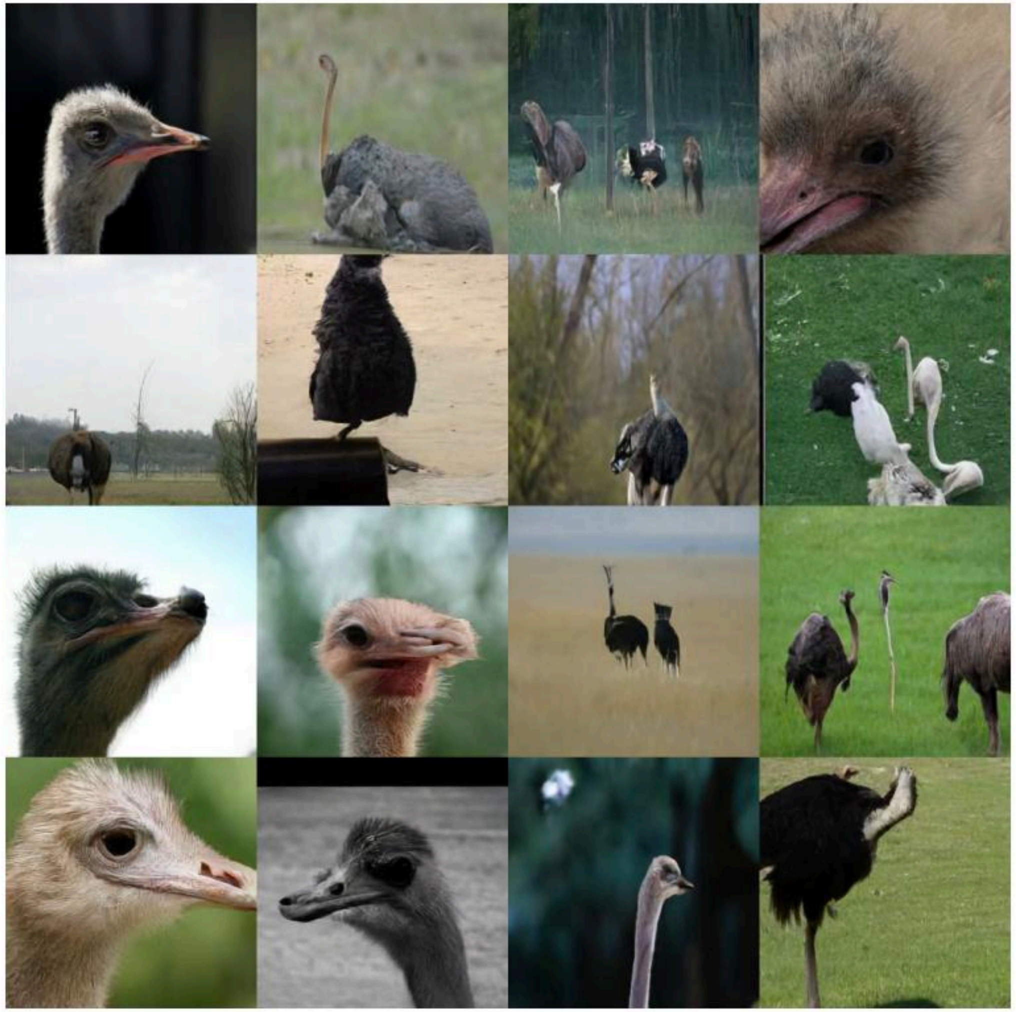


VQ-VAE

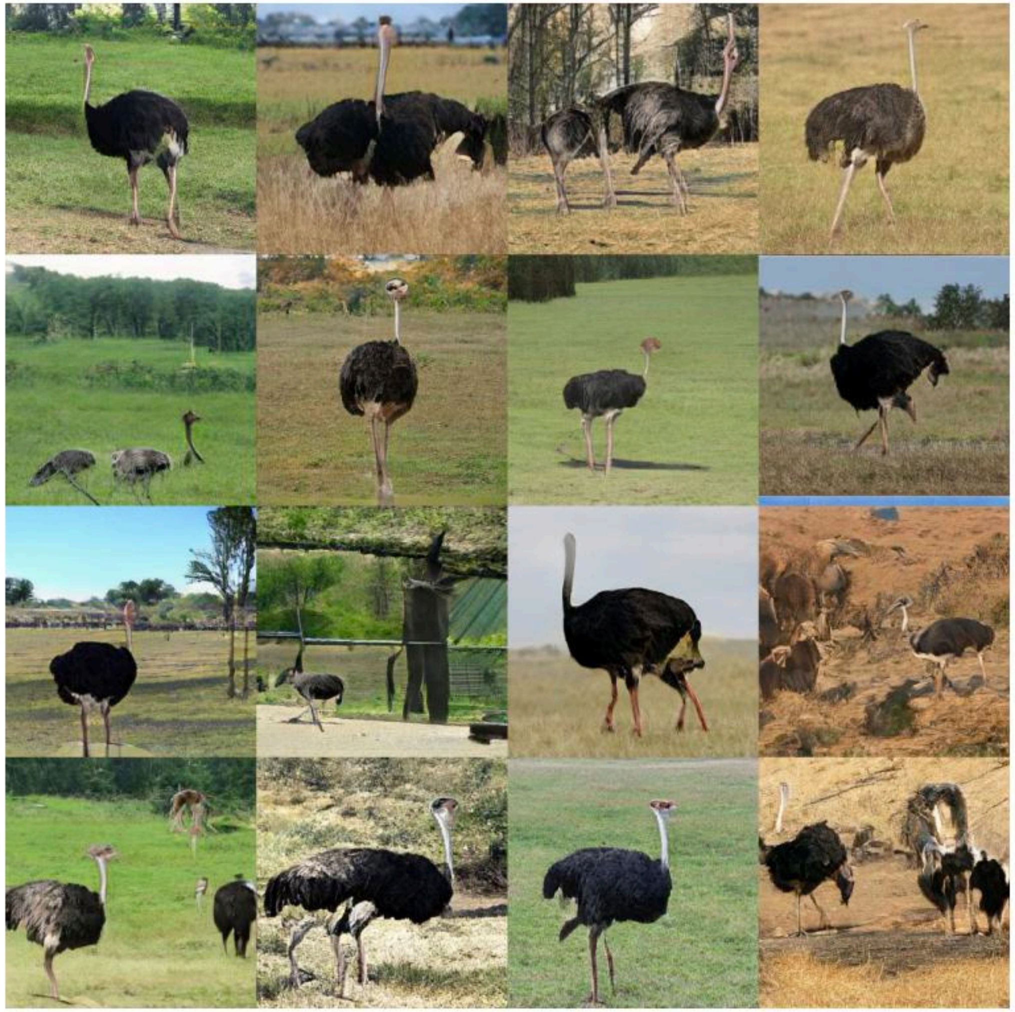


BigGAN deep

VQ-VAE vs. BigGAN deep

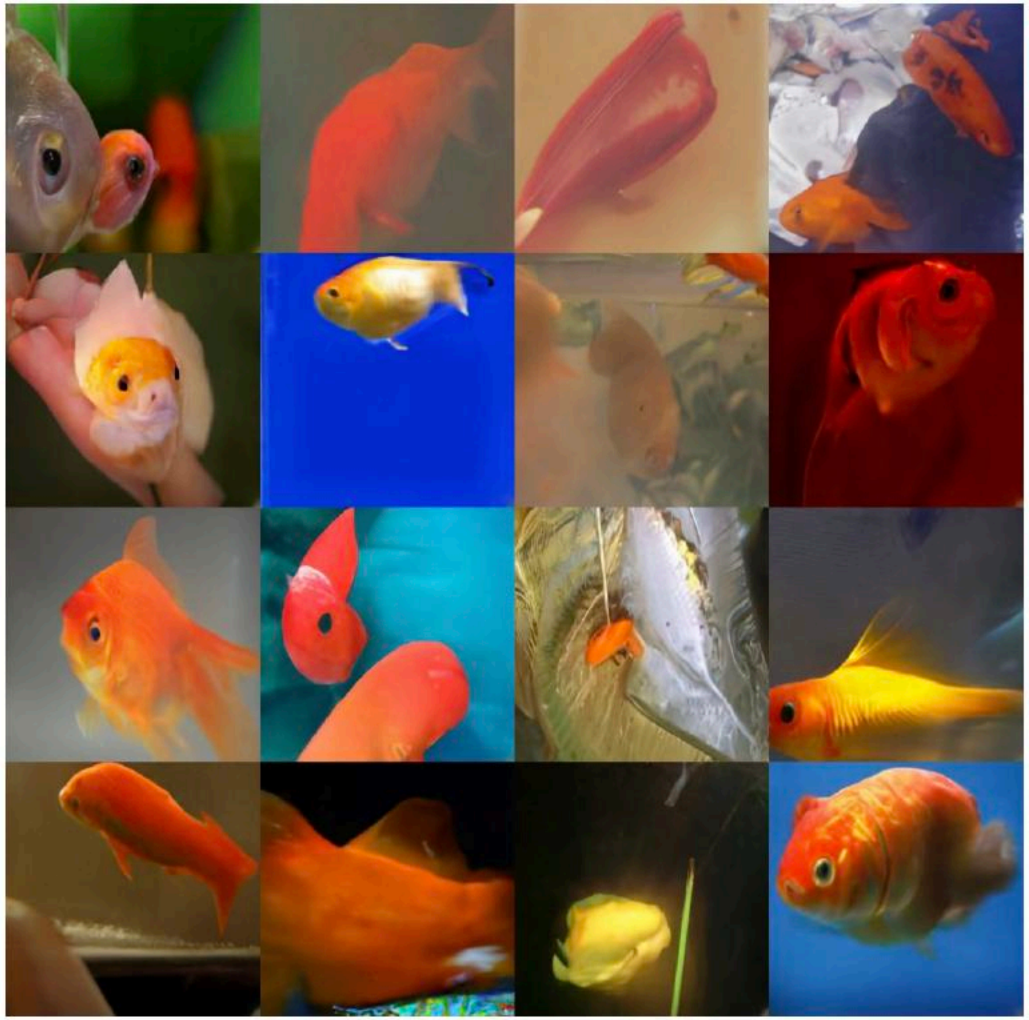


VQ-VAE

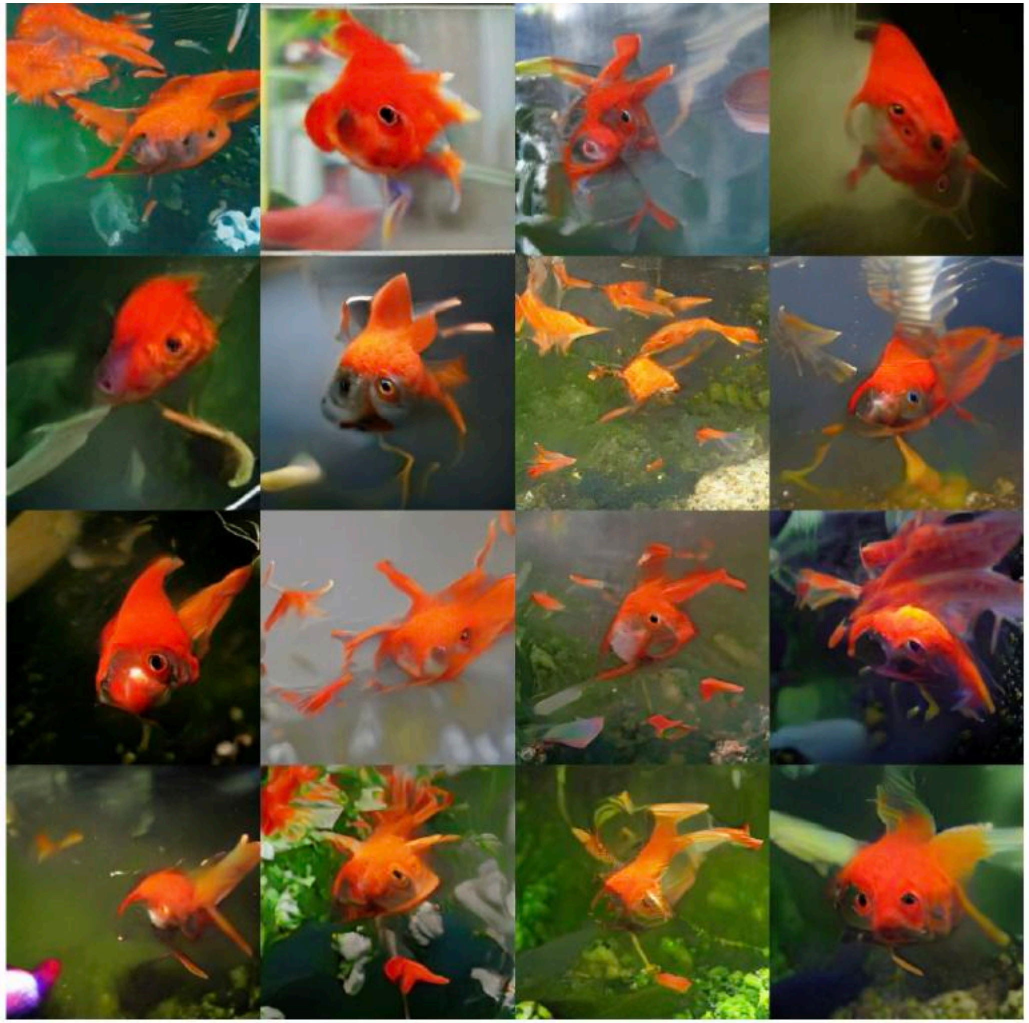


BigGAN deep

VQ-VAE vs. BigGAN deep

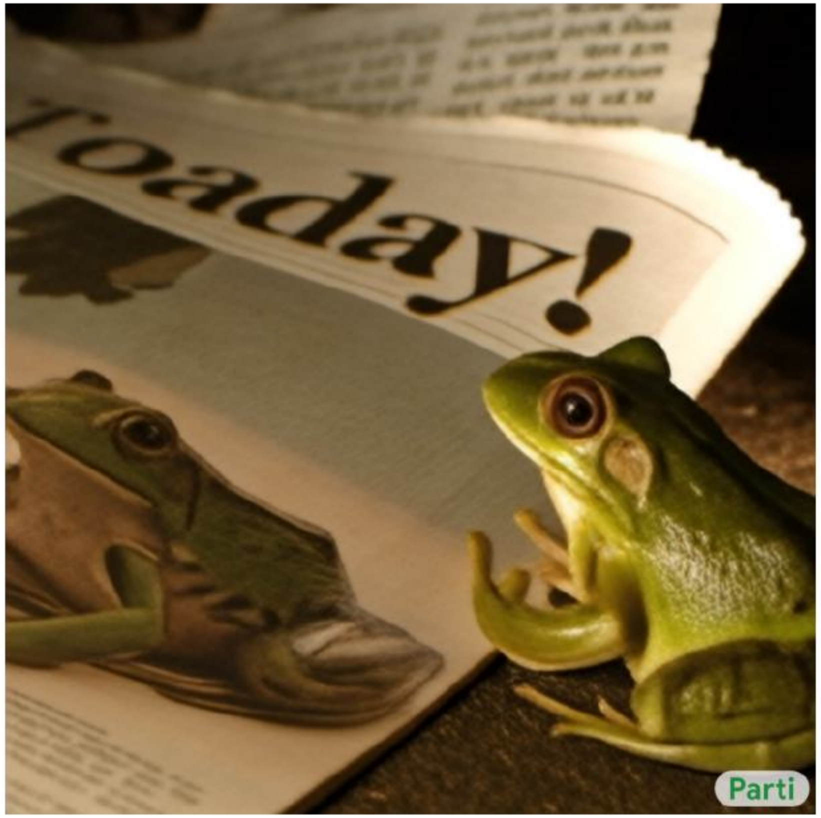


VQ-VAE

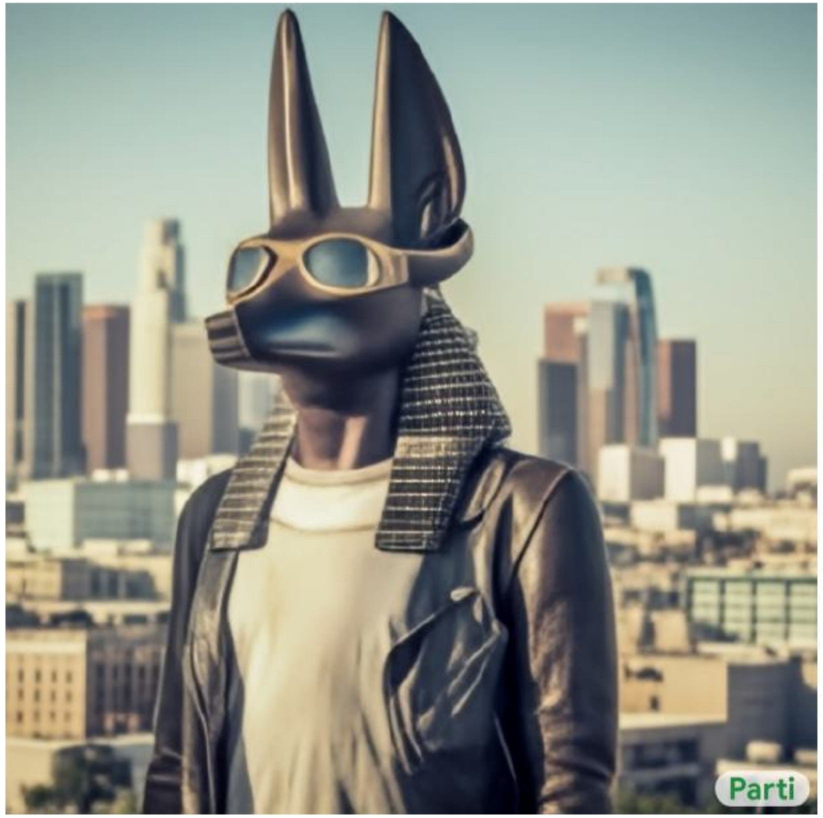


BigGAN deep

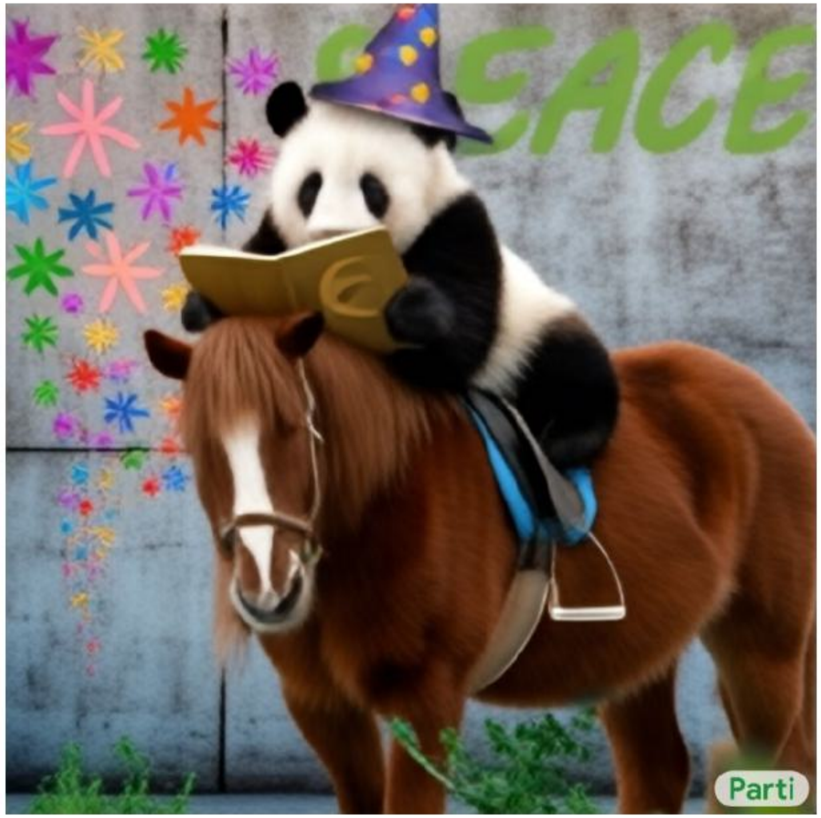
Also good for conditional synthesis



A. *A photo of a frog reading the newspaper named "Toaday" written on it. There is a frog printed on the newspaper too.*



B. *A portrait of a statue of the Egyptian god Anubis wearing aviator goggles, white t-shirt and leather jacket. The city of Los Angeles is in the background. Hi-res DSLR photograph.*



C. *A high-contrast photo of a panda riding a horse. The panda is wearing a wizard hat and is reading a book. The horse is standing on a street against a gray concrete wall. Colorful flowers and the word "PEACE" are painted on the wall. Green grass grows from cracks in the street. DSLR photograph. daytime lighting.*

[Yu et al., "Scaling Autoregressive Models for Content-Rich Text-to-Image Generation", 2022]

Multi-Layer Vector Quantized VAEs

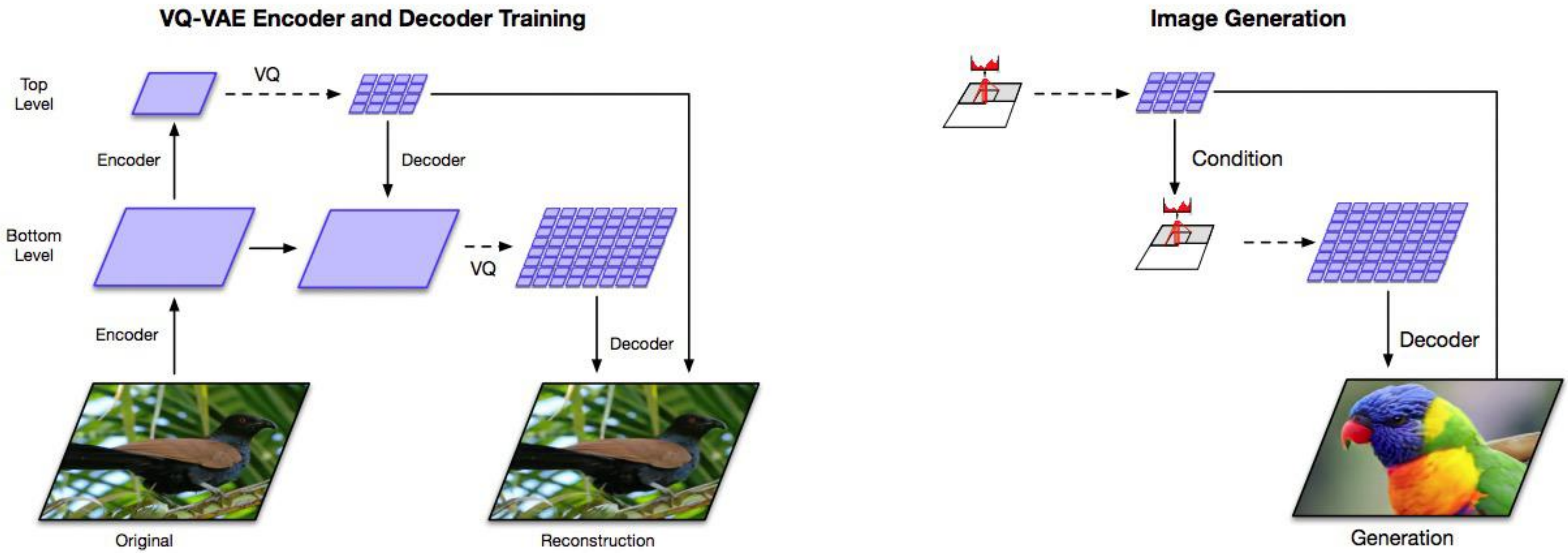
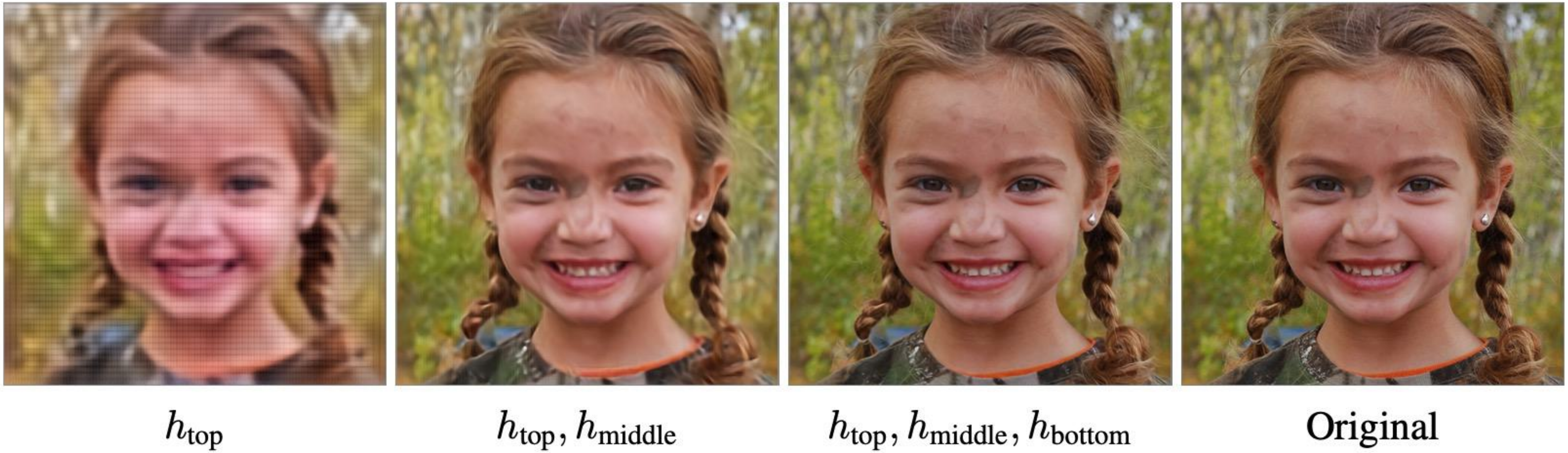


Image Compression



h_{top}

$h_{\text{top}}, h_{\text{middle}}$

$h_{\text{top}}, h_{\text{middle}}, h_{\text{bottom}}$

Original

Figure 3: Reconstructions from a hierarchical VQ-VAE with three latent maps (top, middle, bottom). The rightmost image is the original. Each latent map adds extra detail to the reconstruction. These latent maps are approximately 3072x, 768x, 192x times smaller than the original image (respectively).

Vector Quantization (Emergent Symbols)

- Vector quantization represents a distribution (or density) on vectors with a discrete set of embedded symbols.
- Vector quantization optimizes a rate-distortion tradeoff for vector compression.
- The VQ-VAE uses vector quantization to construct a discrete representation of images and hence a measurable image compression rate-distortion trade-off.

Symbols: A Better Learning Bias

- Do the objects of reality fall into categories?

- If so, shouldn’t a learning architecture be designed to categorize?

- Whole image symbols would yield emergent whole image classification.

Symbols: Improved Interpretability

- Vector quantization shifts interpretation from linear threshold units to the emergent symbols.

- This seems related to the use of t-SNE as a tool in interpretation.

Symbols: Unifying Vision and Language

- Modern language models use word vectors.

- Word vectors are embedded symbols.

- Vector quantization also results in models based on embedded symbols.

Symbols: Addressing the "Forgetting" Problem

- When we learn to ski we do not forget how to ride a bicycle.
- However, when a model is trained on a first task, retraining on a second tasks degrades performance on the first (the model "forgets").
- But embedded symbols can be task specific.
- The embedding of a task-specific symbol will not change when training on a different task.

Symbols: Improved Transfer Learning

- Embedded symbols can be domain specific.

- Separating domain-general parameters from domain-specific parameters may improve transfer between domains.

Today

- variational autoencoders (VAEs)
- vector quantized VAEs (VQ-VAEs)
- **denoising diffusion models**

Observation 1: Diffusion Destroys Structure

- Dye density represents probability density
- **Goal:** Learn structure probability density
- **Observation:** Diffusion destroys structure

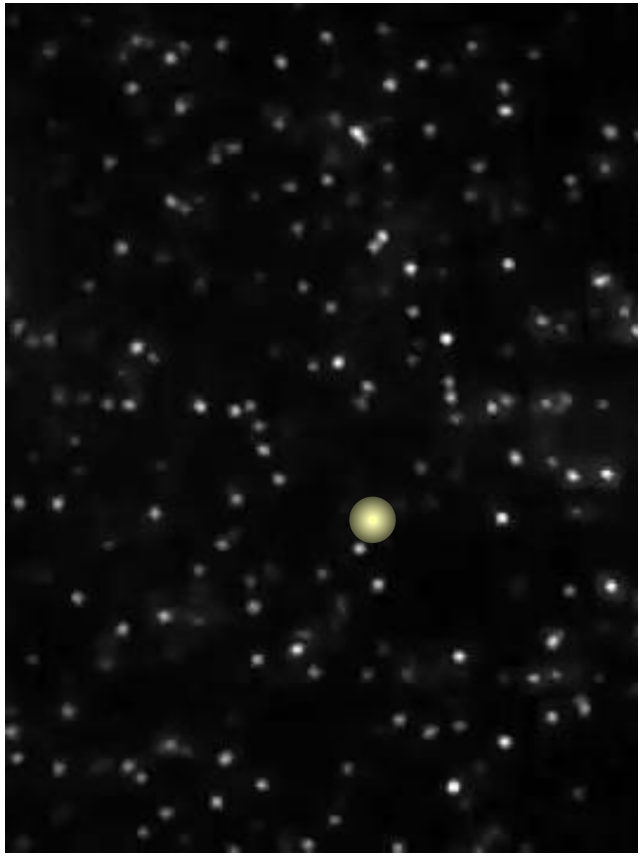
Data distribution → Uniform distribution

Idea: Recover Structure by Reversing Time

- What if we could reverse time?
- Recover data distribution by starting from uniform distribution and running dynamics backwards

Data distribution ← Uniform distribution

Observation 2: Microscopic Diffusion is Time Reversible

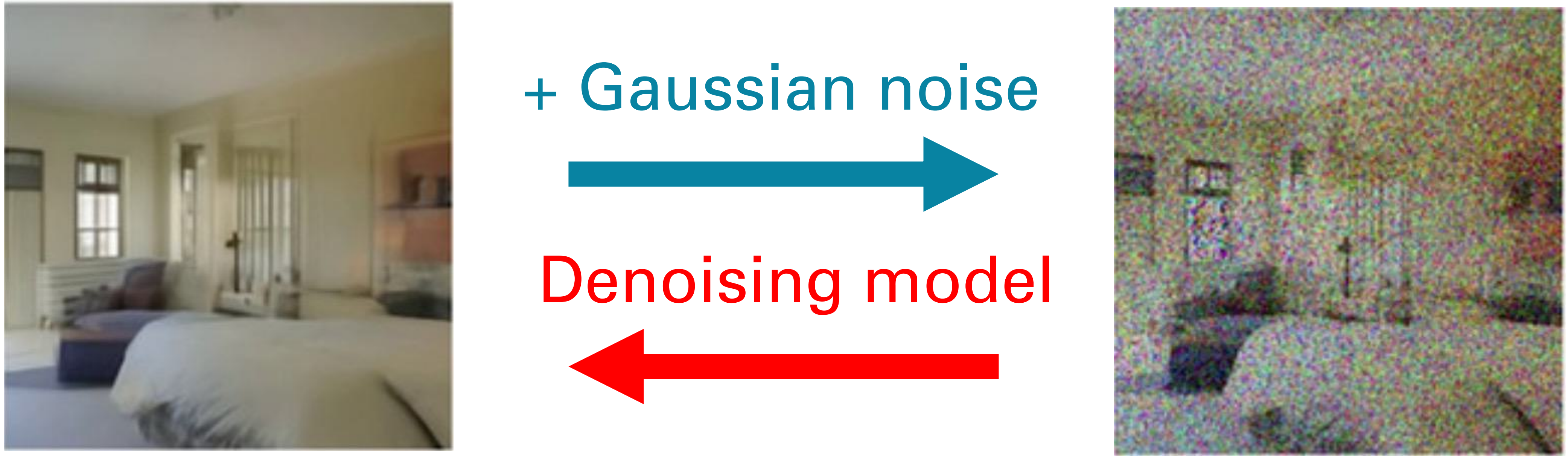


Nanoparticles in water

- Microscopic view
- Brownian motion
- Position updates are small Gaussians
  - Both forwards and backwards in time

Video credit: Rutger Saly

Overview of Diffusion Probabilistic Models



+ Gaussian noise

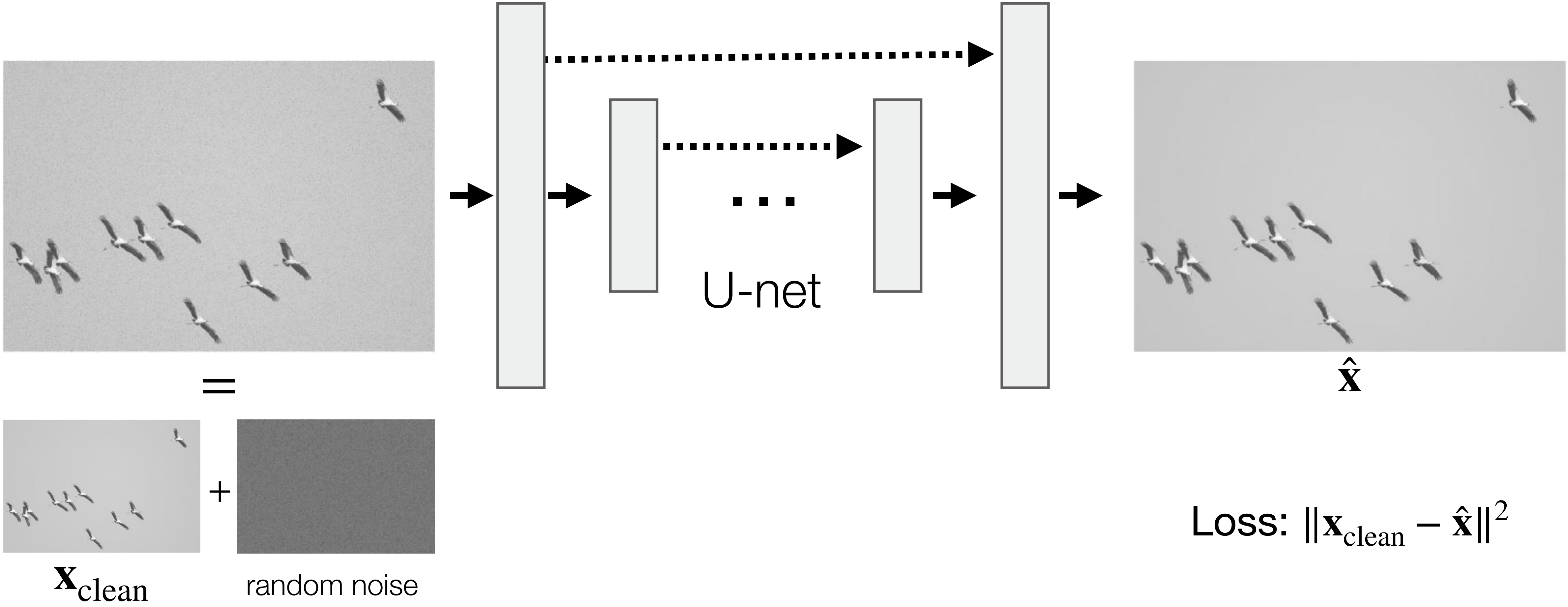
Denoising model

Clean

Noisy

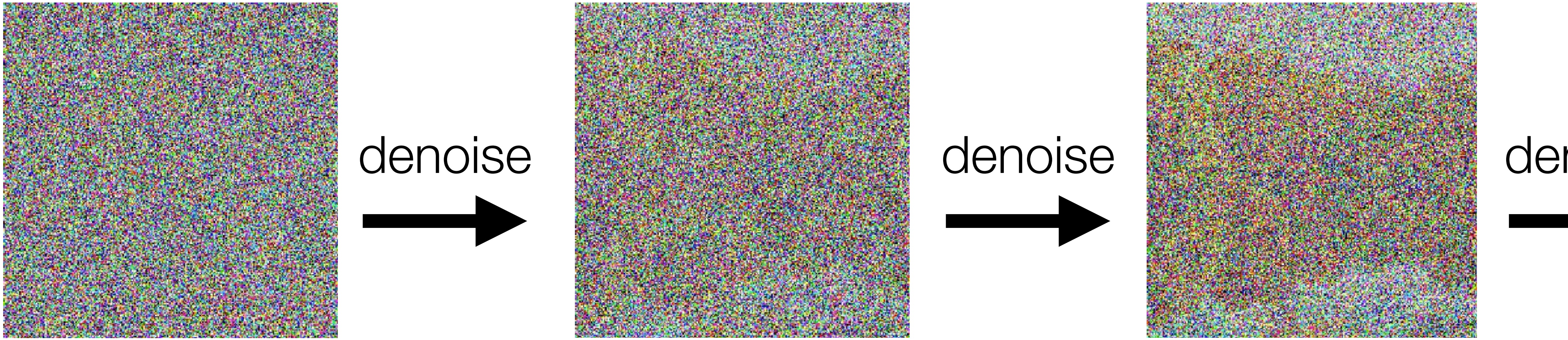
Slide adapted from Kiaming Song

Recall: the denoising problem



Loss: $\|\mathbf{x}_{\text{clean}} - \hat{\mathbf{x}}\|^2$

From noise to an image



denoise

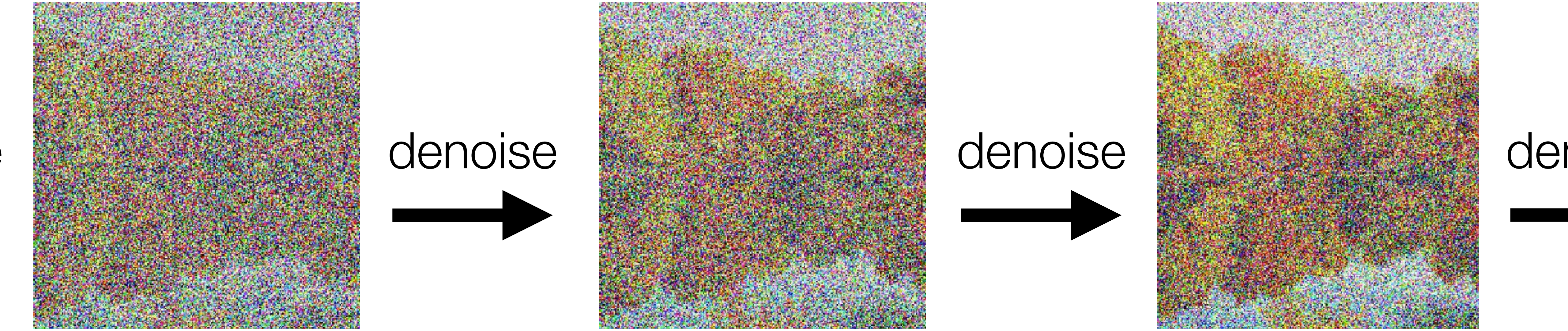
denoise

der

Random noise

Example source: Aditya Ramesh

From noise to an image



e

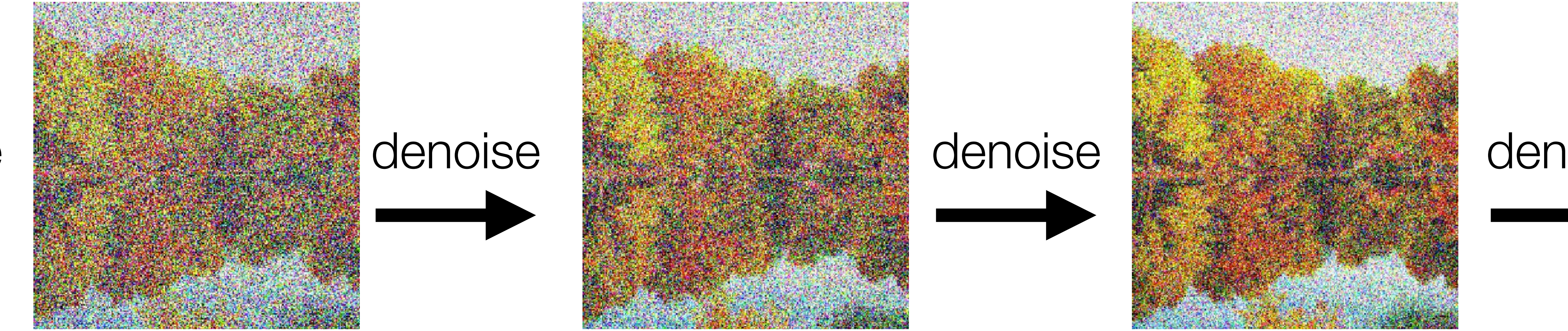
denoise

denoise

den

Example source: Aditya Ramesh

From noise to an image



e

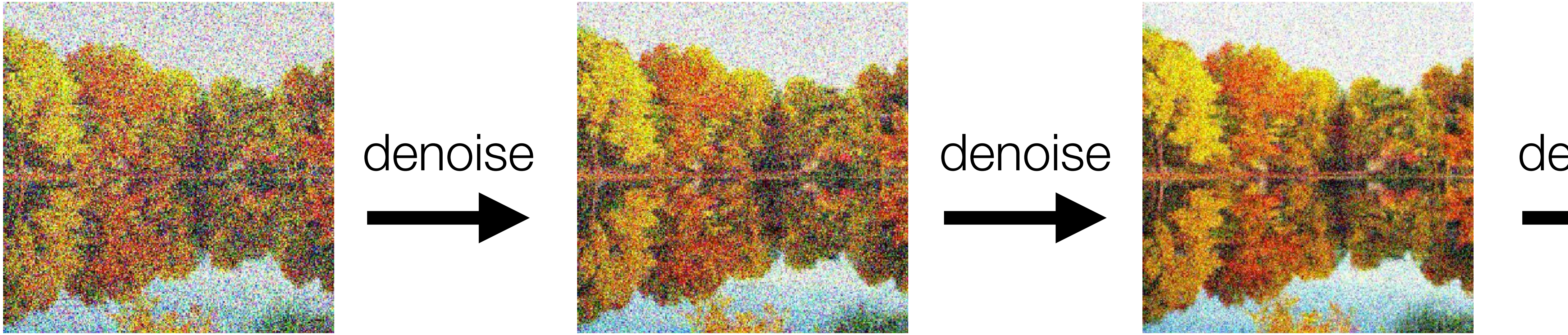
denoise

denoise

den

Example source: Aditya Ramesh

From noise to an image



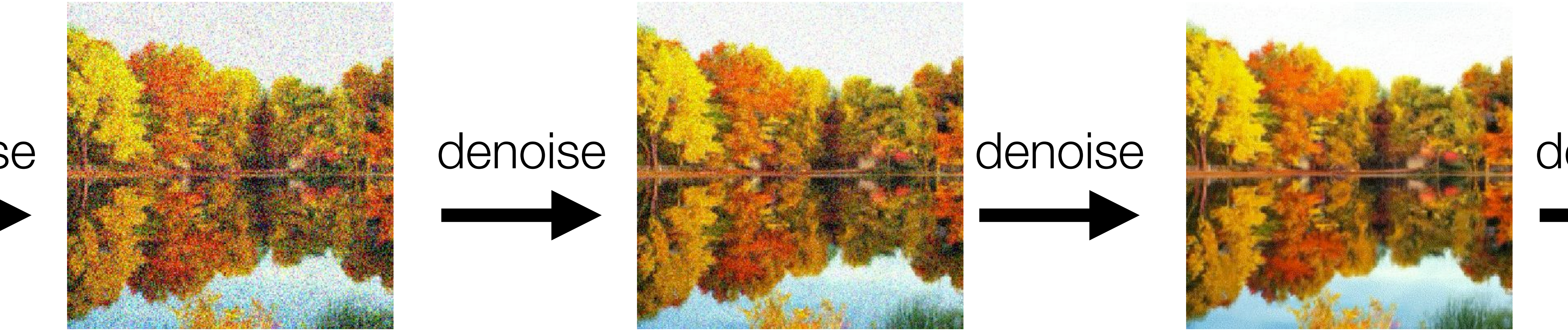
denoise

denoise

de

Example source: Aditya Ramesh

From noise to an image



se

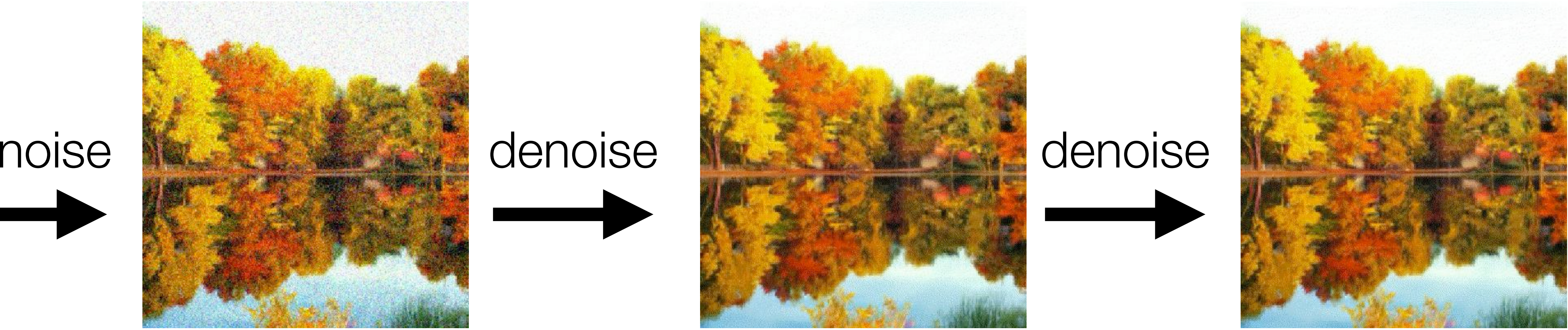
denoise

denoise

de

Example source: Aditya Ramesh

From noise to an image



noise

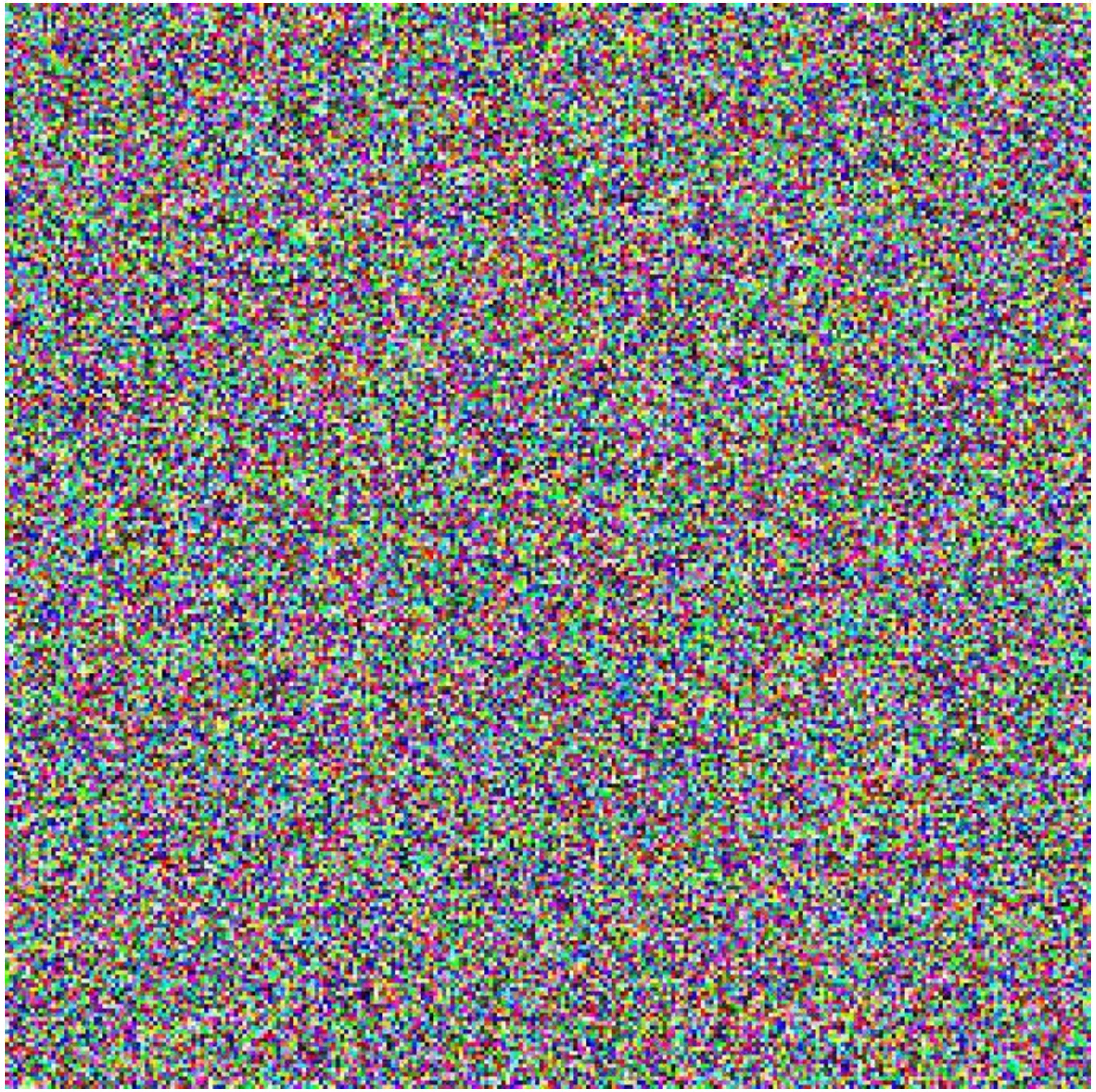
denoise

denoise

Generated image!

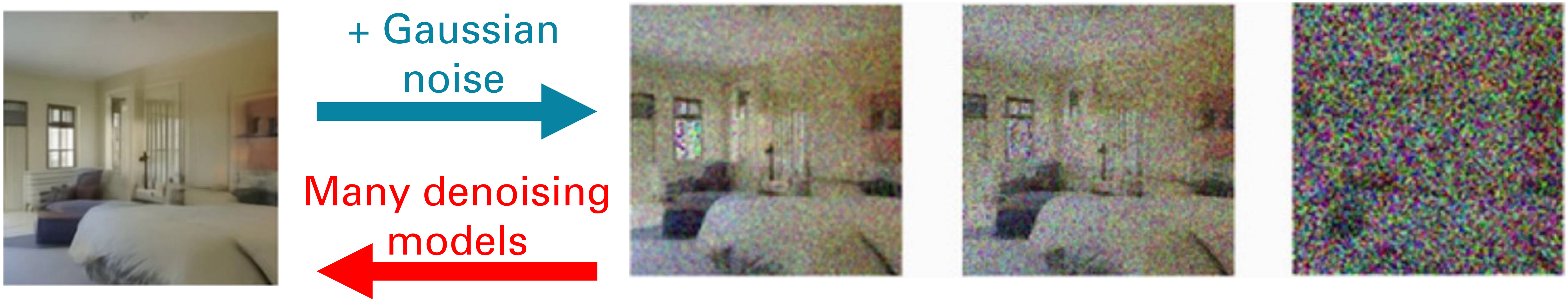
Example source: Aditya Ramesh

From noise to an image



Example source: Aditya Ramesh

Diffusion Probabilistic Models



+ Gaussian noise

Many denoising models

Clean

Different noise levels

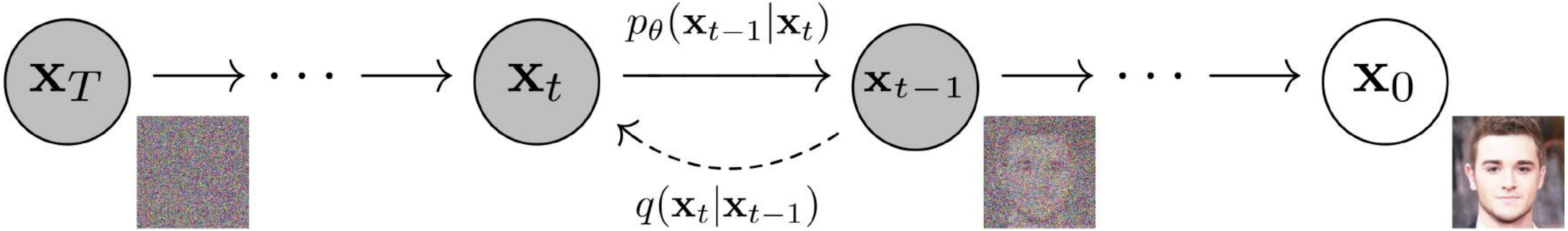
Slide adapted from Kiaming Song

Overview of Diffusion Probabilistic Models

- Destroy all structure in data distribution using diffusion process
- Learn reversal of diffusion process
  - Estimate function for mean and covariance of each step in the reverse diffusion process (binomial rate for binary data)
- Reverse diffusion process is the model of the data

Diffusion Probabilistic Models

- Diffusion model aims to learn the reverse of noise generation procedure
  - **Forward step**: (Iteratively) Add noise to the original sample
    - → The sample $\mathbf{x}_0$ converges to the complete noise $\mathbf{x}_T$ (e.g., $\mathcal{N}(0, I)$)



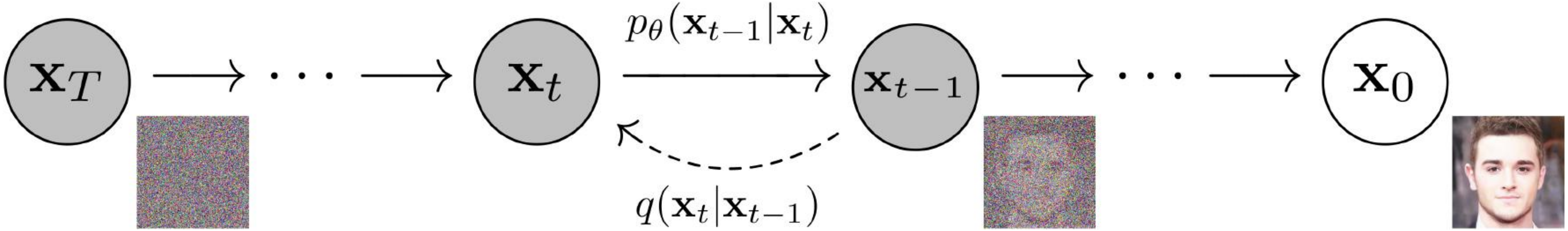
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$

Forward (diffusion) process

Diffusion Probabilistic Models

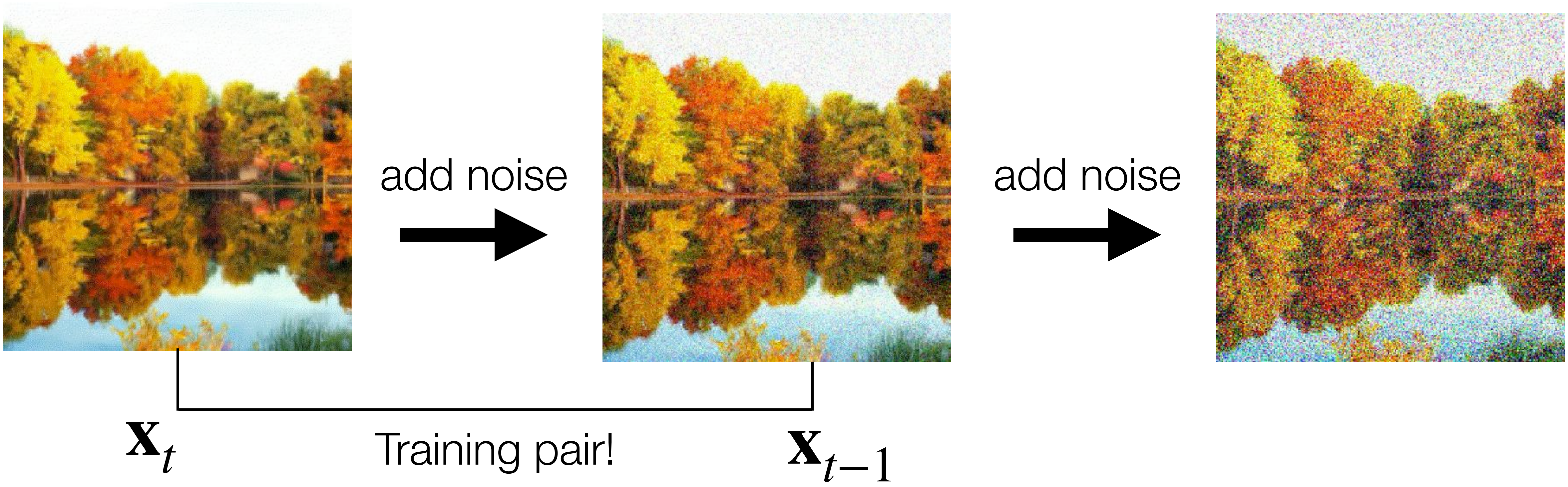
- Diffusion model aims to learn the reverse of noise generation procedure
  - **Forward step**: (Iteratively) Add noise to the original sample
    - → The sample $\mathbf{x}_0$ converges to the complete noise $\mathbf{x}_T$ (e.g., $\mathcal{N}(0, I)$)
  - **Reverse step**: Recover the original sample from the noise
    - → Note that it is the "generation" procedure

Reverse process


$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_T$
$\mathbf{x}_t$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$

Forward (diffusion) process

How do we train this model?



add noise

add noise

$\mathbf{x}_t$

Training pair!

$\mathbf{x}_{t-1}$

- We’ll use a variance schedule, $\beta_1, \beta_2, \ldots, \beta_T$, for $0 < \beta_t < 1$
- Also, we’ll scale the image by a factor $\sqrt{1-\beta_t}$ so that mean goes to 0 over time.

Diffusion Probabilistic Models

- Diffusion model aims to learn the reverse of noise generation procedure
  - **Forward step**: (Iteratively) Add noise to the original sample

    → Technically, it is a product of conditional noise distributions $q(\mathbf{x}_t|\mathbf{x}_{t-1})$

    - Usually, the parameters β_t are fixed (one can jointly learn, but not beneficial)
    - Noise annealing (i.e., reducing noise scale $\beta_t < \beta_{t-1}$) is crucial to the performance

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

Diffusion Probabilistic Models

- Diffusion model aims to learn the reverse of noise generation procedure
  - **Forward step**: (Iteratively) Add noise to the original sample
    - →Technically, it is a product of conditional noise distributions $q(\mathbf{x}_t|\mathbf{x}_{t-1})$
      - Usually, the parameters β_{t} are fixed (one can jointly learn, but not beneficial)
      - Noise annealing (i.e., reducing noise scale $\beta_{\text{t}} < \beta_{\text{t-1}}$) is crucial to the performance

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

  - **Reverse step**: Recover the original sample from the noise
    - →It is also a product of conditional (de)noise distributions $p_\theta(\mathbf{x}_{t=1}|\mathbf{x}_t)$
      - →Use the **learned** parameters: denoiser $\boldsymbol{\mu}_{\theta}$ (main part) and randomness $\boldsymbol{\Sigma}_{\theta}$

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

Diffusion Probabilistic Models

- Diffusion model aims to learn the reverse of noise generation procedure
  - **Forward step**: (Iteratively) Add noise to the original sample
  - **Reverse step**: Recover the original sample from the noise

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

  - **Training:** Minimize variational lower bound of the model

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

Diffusion Probabilistic Models

- Diffusion model aims to learn the reverse of noise generation procedure
  - **Forward step**: (Iteratively) Add noise to the original sample
  - **Reverse step**: Recover the original sample from the noise

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

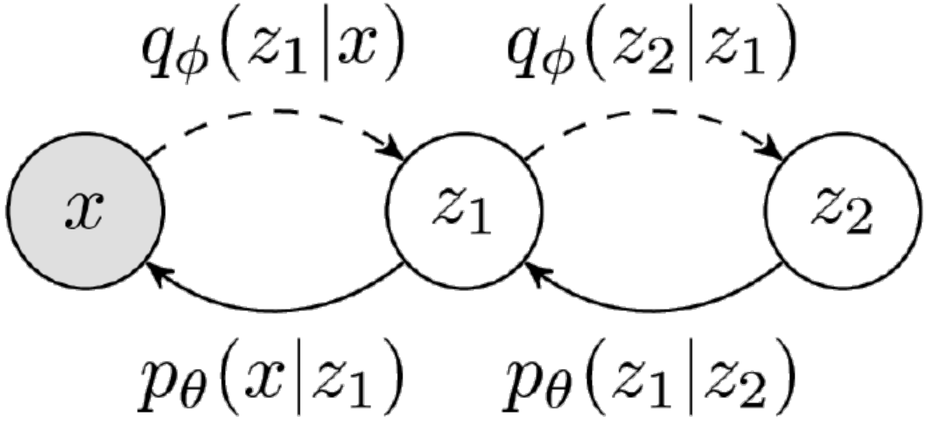
  - **Training:** Minimize variational lower bound of the model

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

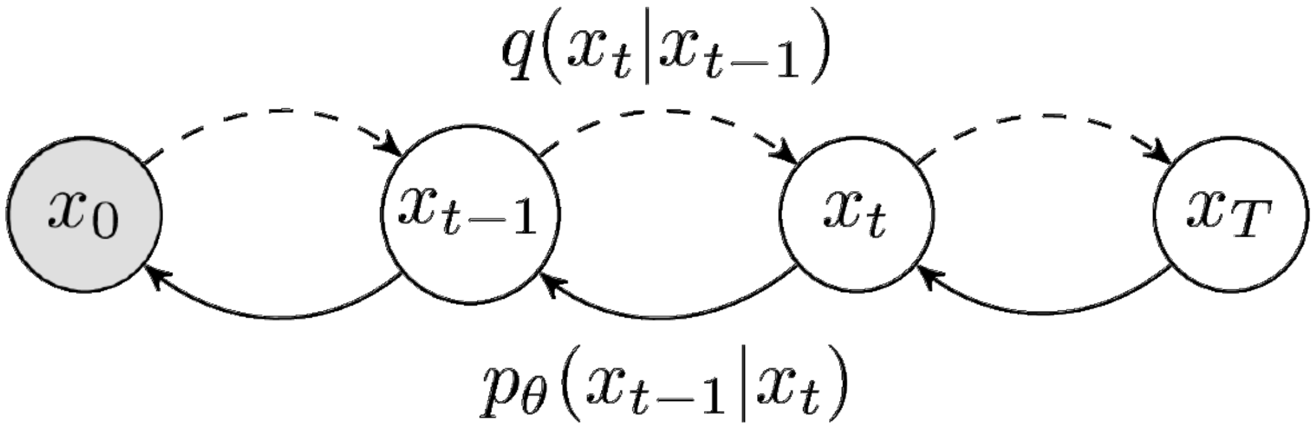
    - It can be decomposed to the **step-wise** losses (for each step *t*)

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

Diffusion Models as a kind of VAE



A Hierarchical VAE



A Diffusion Probabilistic Model

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

Diffusion Probabilistic Models

- Diffusion model aims to learn the reverse of noise generation procedure
  - **Training:** Minimize variational lower bound of the model
    - It can be decomposed to the **step-wise** losses (for each step *t*)

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

  - Here, the true reverse step $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ can be computed as a **closed form** of β_{t}
    - Note that we only define the true forward step

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_{t-1} ; \tilde{\mu}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t^3 \mathbf{I}\right)$$

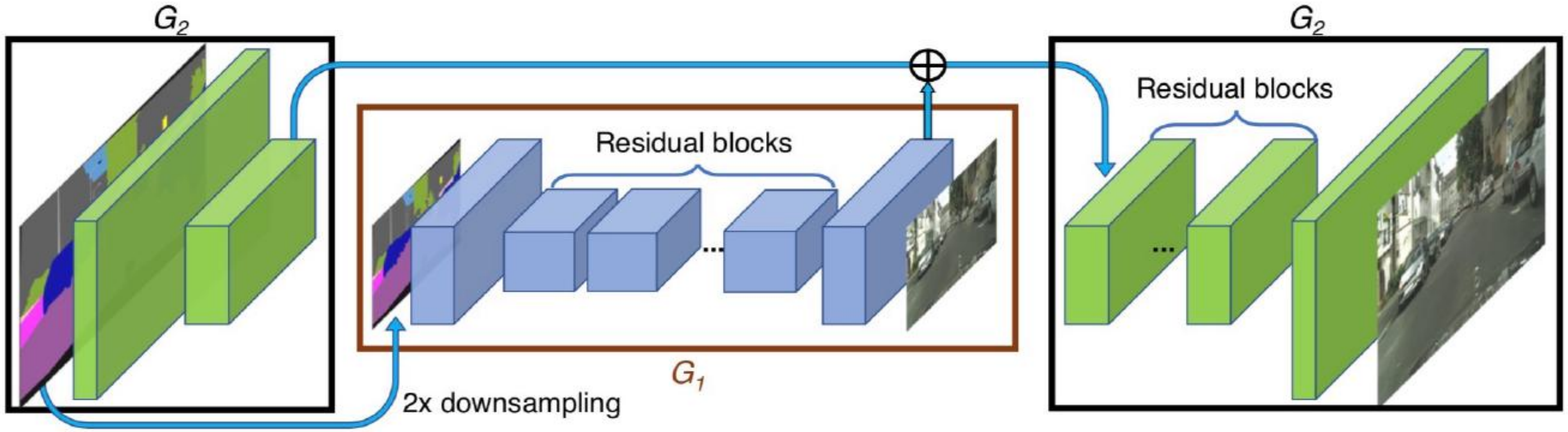
$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right):=\tilde{\beta}_t^1 \mathbf{x}_0+\tilde{\beta}_t^2 \mathbf{x}_t$$

  - Since all distributions above are Gaussian, the KL divergences are tractable

Diffusion Probabilistic Models

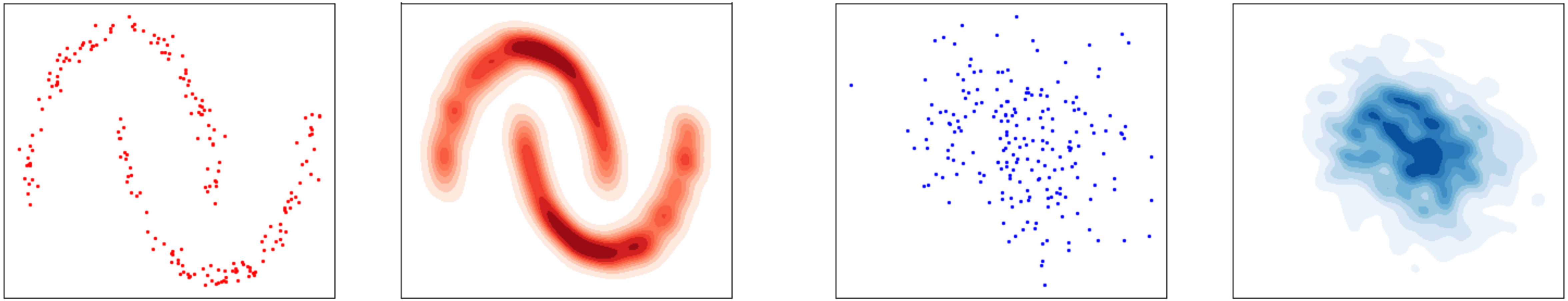
- Diffusion model aims to learn the reverse of noise generation procedure
  - **Network:** Use the image-to-image translation (e.g., U-Net) architectures
    - Recall that input is $\mathbf{x}_t$ and output is $\mathbf{x}_{t-1}$, both are images
    - It is expensive since both input and output are high-dimensional

    - Note that the denoiser μ_θ ($\mathbf{x}_t$, t) shares weights, but conditioned by step t



Diffusion Probabilistic Models

- Diffusion model aims to learn the reverse of noise generation procedure
  - **Sampling:** Draw a random noise $\boldsymbol{x}_{\mathrm{T}}$ then apply the reverse step $p(\mathbf{x}_{t=1}|\mathbf{x}_t)$
    - It often requires the hundreds of reverse steps (very slow)
  - Early and late steps change the high- and low-level attributes, respectively



Diffusion Probabilistic Models – CIFAR10

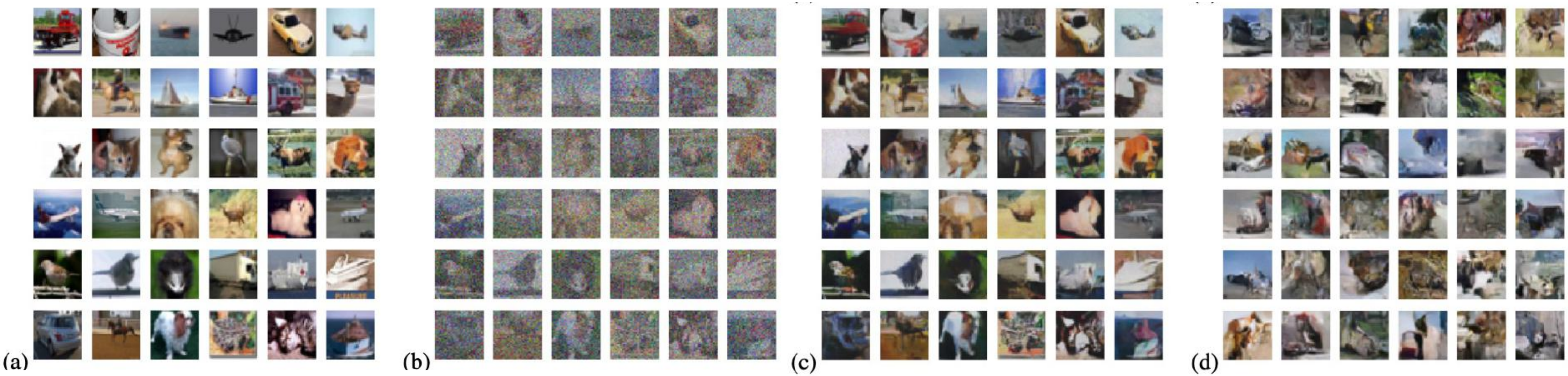


Figure 3. The proposed framework trained on the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset. *(a)* Example holdout data (similar to training data). *(b)* Holdout data corrupted with Gaussian noise of variance 1 (SNR = 1). *(c)* Denoised images, generated by sampling from the posterior distribution over denoised images conditioned on the images in (b). *(d)* Samples generated by the diffusion model.

Denoising Diffusion Probabilistic Model

- DDPM reparametrizes the reverse distributions of diffusion models
  - Key idea: The original reverse step fully creates the denoiser $\mu_\theta(\mathbf{x}_t, t)$ from $\mathbf{x}_t$
    - However, $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ share most information, and thus it is redundant
      - Instead, create the **residual** $\epsilon_\theta(\mathbf{x}_t, t)$ and add to the original $\mathbf{x}_t$
    - Set $\mathbf{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2\mathbf{I}$

Training resembles denoising score matching

Sampling resembles Langevin Dynamics

Initiated the diffusion model boom!

Denoising Diffusion Probabilistic Model

- DDPM reparametrizes the reverse distributions of diffusion models
  - Key idea: The original reverse step fully creates the denoiser $\mu_\theta(\mathbf{x}_t, t)$ from $\mathbf{x}_t$
    - However, $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ share most information, and thus it is redundant
      - Instead, create the **residual** $\epsilon_\theta(\mathbf{x}_t,t)$ and add to the original $\mathbf{x}_t$
    - Set $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2\mathbf{I}$
  - Formally, DDPM reparametrizes the learned reverse distribution as

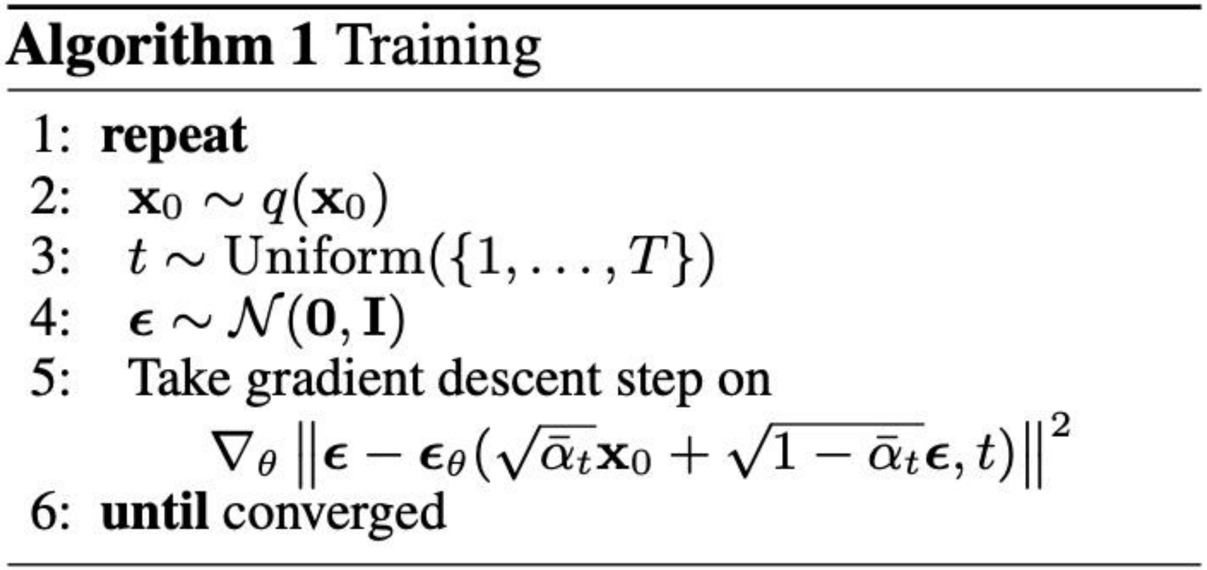
$$\boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right)$$

$$\alpha_t := 1 - \beta_t$$
$$\bar{\alpha}_t := \prod\nolimits_{s=1}^{t} \alpha_s$$

and the step-wise objective L_{t-1} can be reformulated as

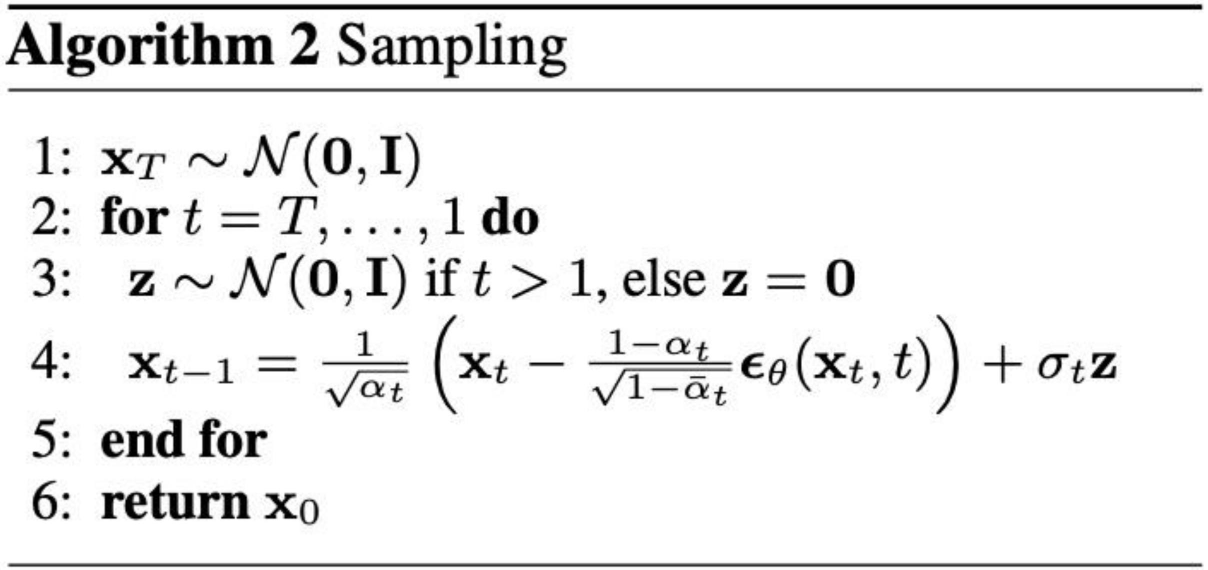
$$\mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right]$$

Denoising Diffusion Probabilistic Model



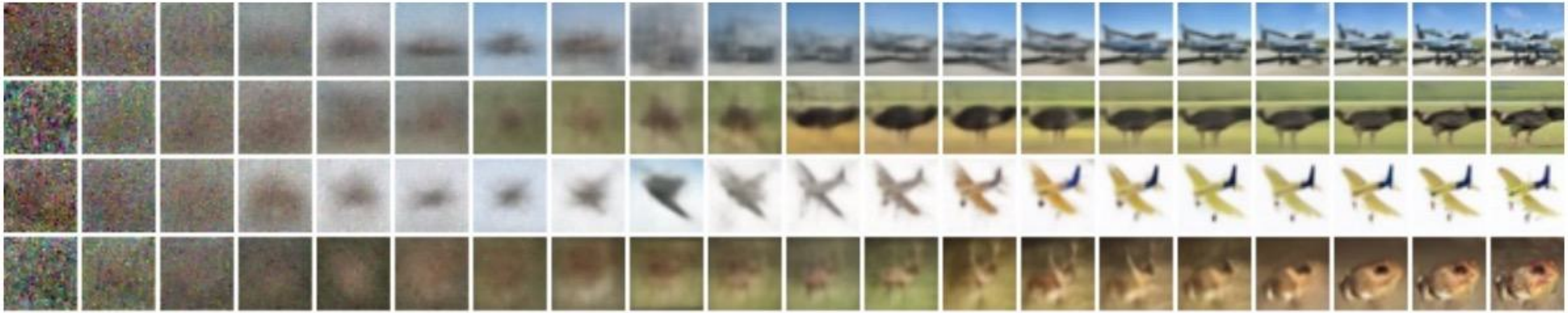
Algorithm 1 Training

```
repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  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
```

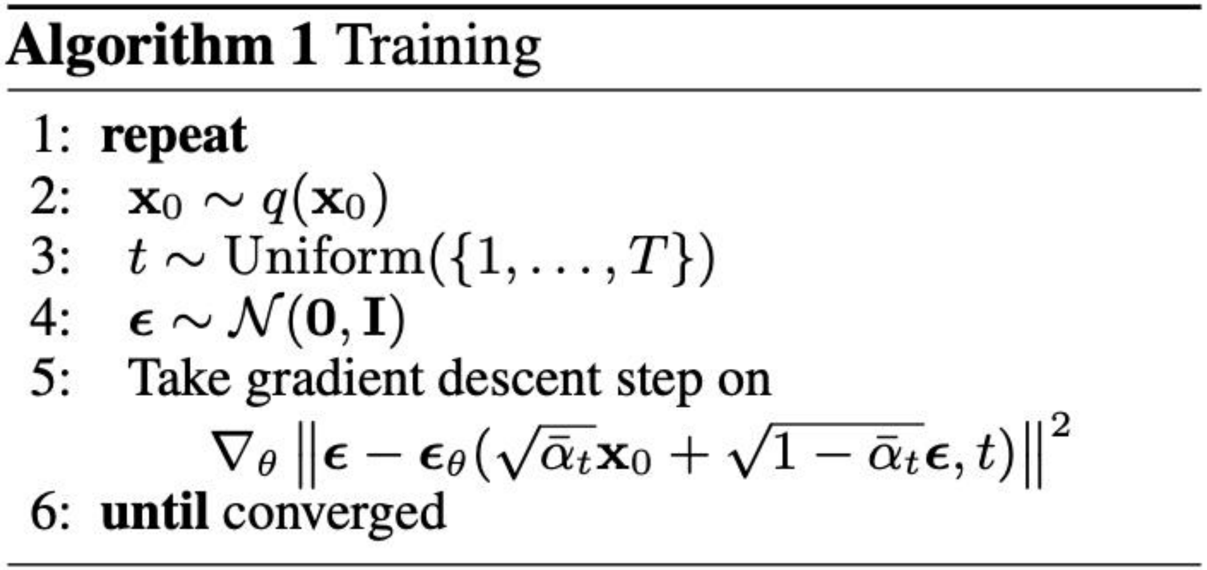


Algorithm 2 Sampling

```
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ 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 $\mathbf{x}_0$
```

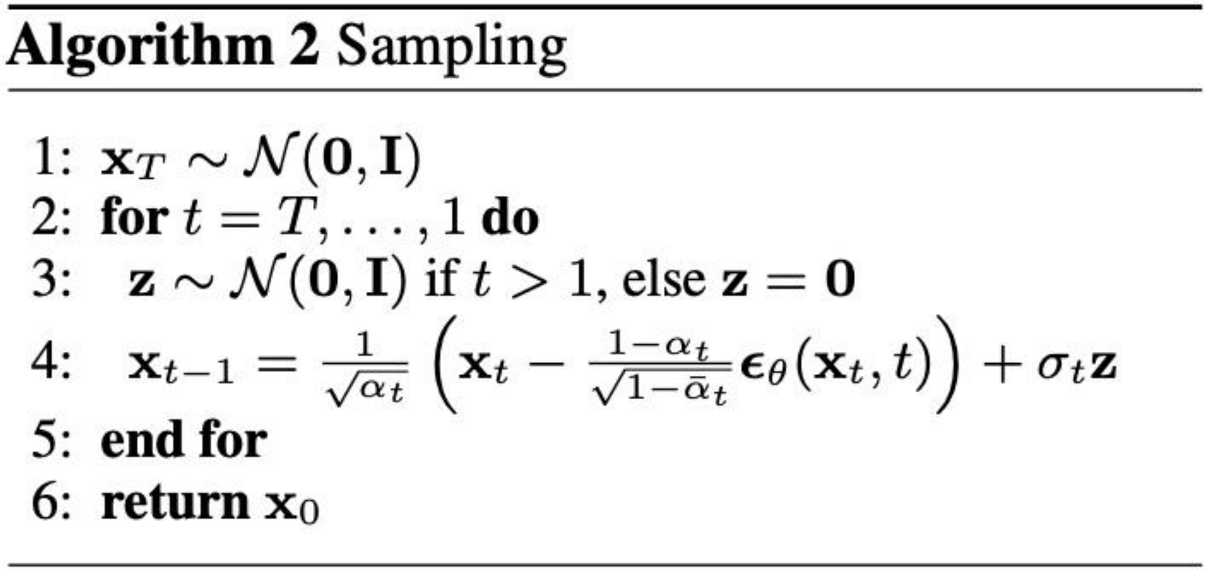


Denoising Diffusion Probabilistic Model



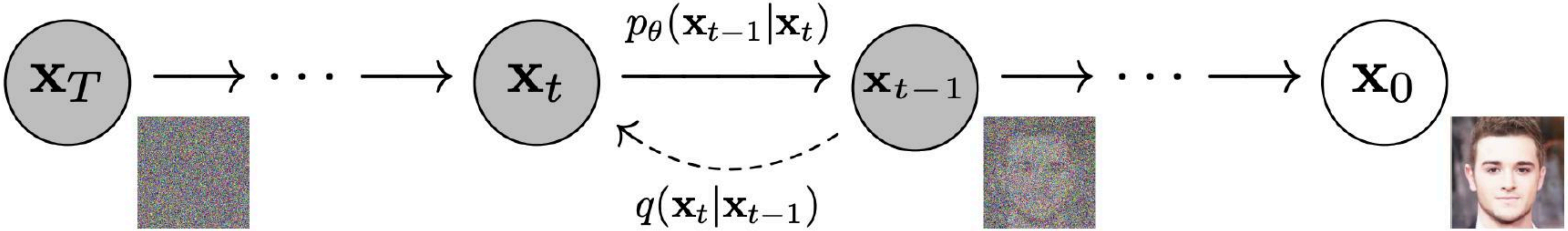
Algorithm 1 Training

```
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
```

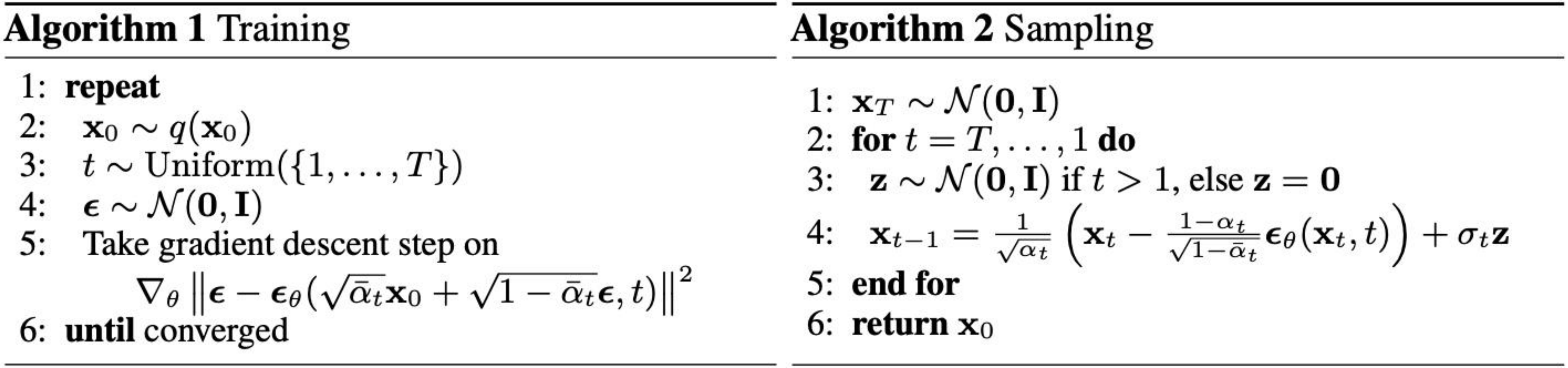


Algorithm 2 Sampling

```
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
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 $\mathbf{x}_0$
```


$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$

Denoising Diffusion Probabilistic Model

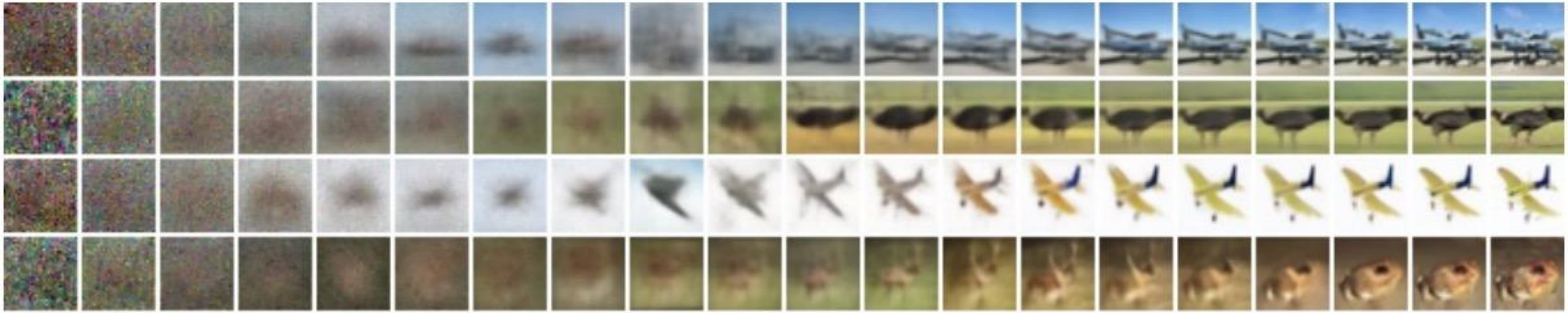


Algorithm 1 Training

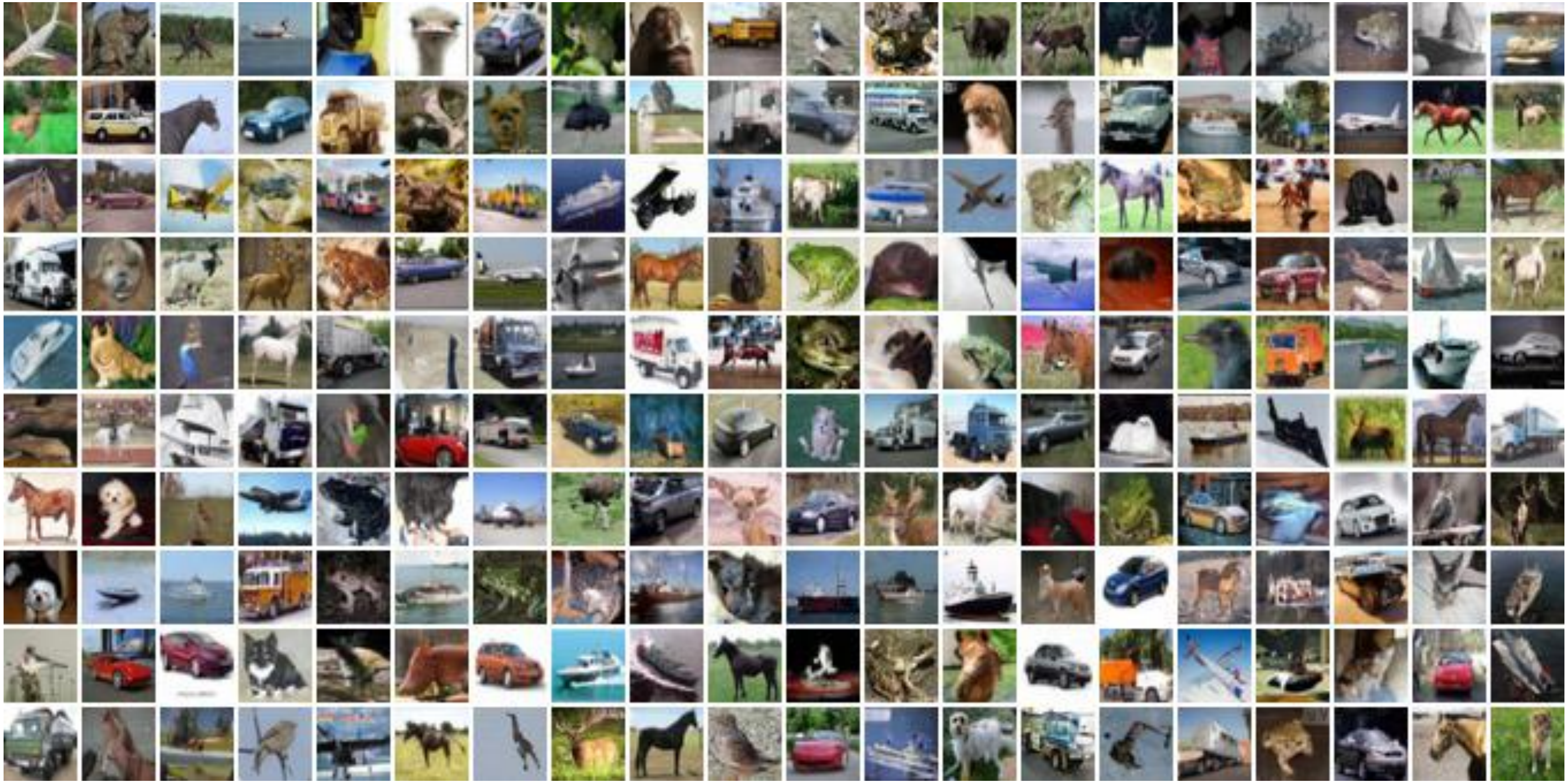
```
repeat
    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
    $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Take gradient descent step on
        $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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



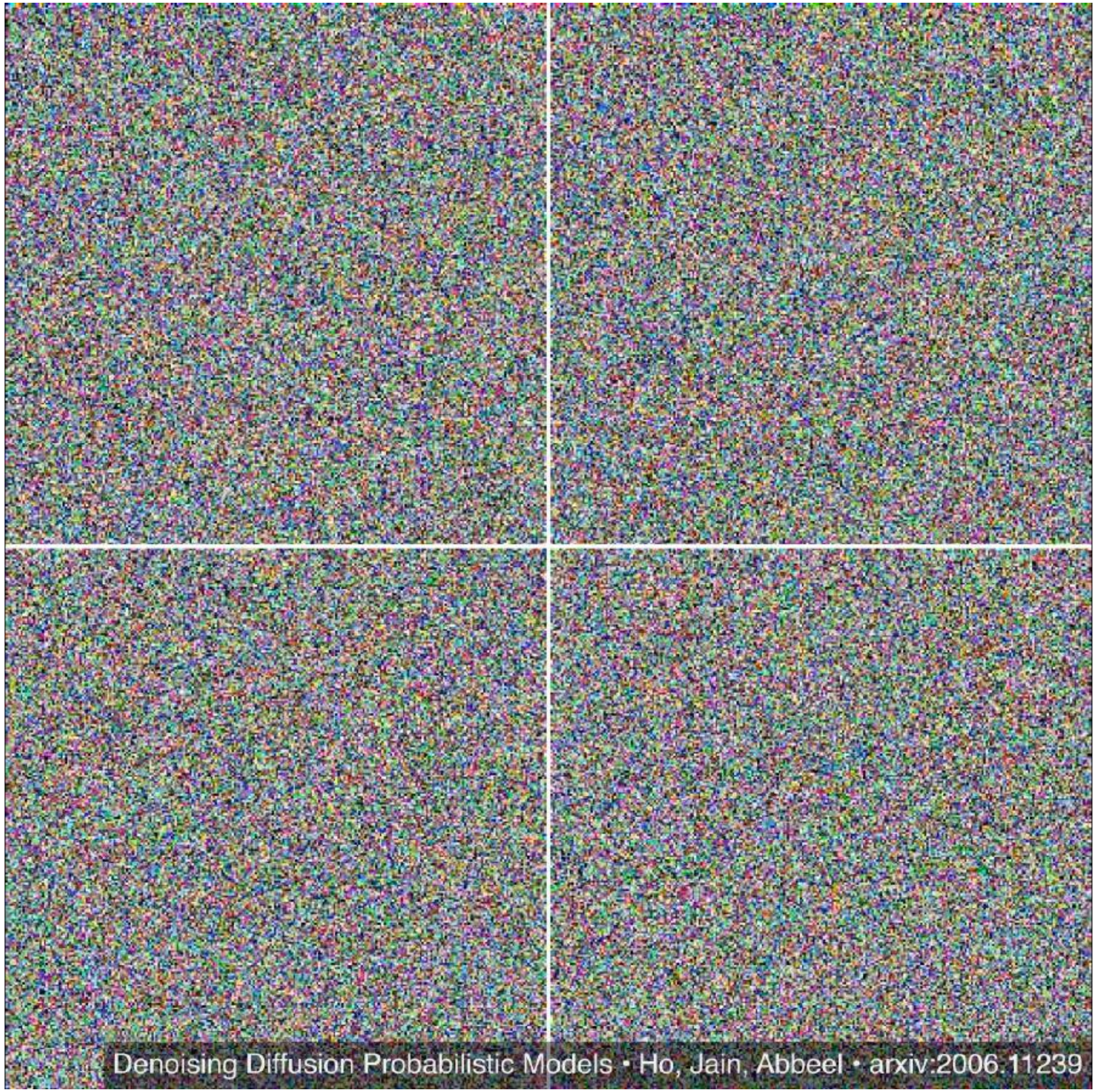
Denoising Diffusion Probabilistic Model



Unconditional CIFAR10 samples. Inception Score=9.46, FID=3.17.

Denoising Diffusion Probabilistic Model

Denoising Diffusion Probabilistic Model



Denoising Diffusion Probabilistic Models • Ho, Jain, Abbeel • arxiv:2006.11239

Denoising Diffusion Implicit Model

- DDIM roughly sketches the final sample, then refine it with the reverse process

- **Motivation**:
  - Diffusion model is slow due to the iterative procedure
  - GAN/VAE creates the sample by one-shot forward operation
  - Can we combine the advantages for fast sampling of diffusion models?

- **Technical spoiler:**
  Instead of naively applying diffusion model upon GAN/VAE,
  DDIM proposes a principled approach of rough sketch + refinement

Denoising Diffusion Implicit Model

- DDIM roughly sketches the final sample, then refine it with the reverse process
  - Key idea:
    - Given $\mathbf{x}_t$, generate the rough sketch $\mathbf{x}_0$ and refine $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$
    - Unlike original diffusion model, it is not a Markovian structure

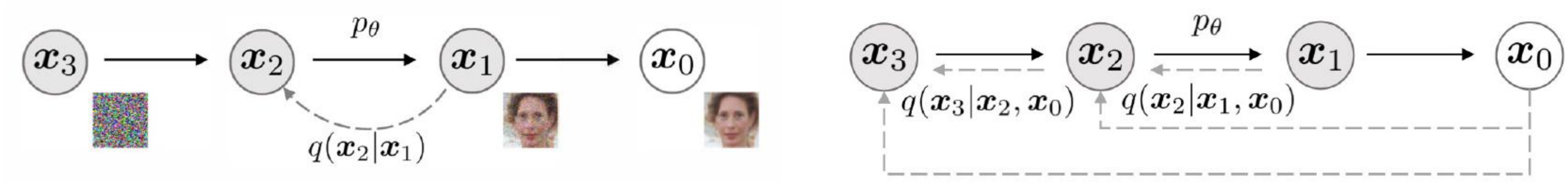
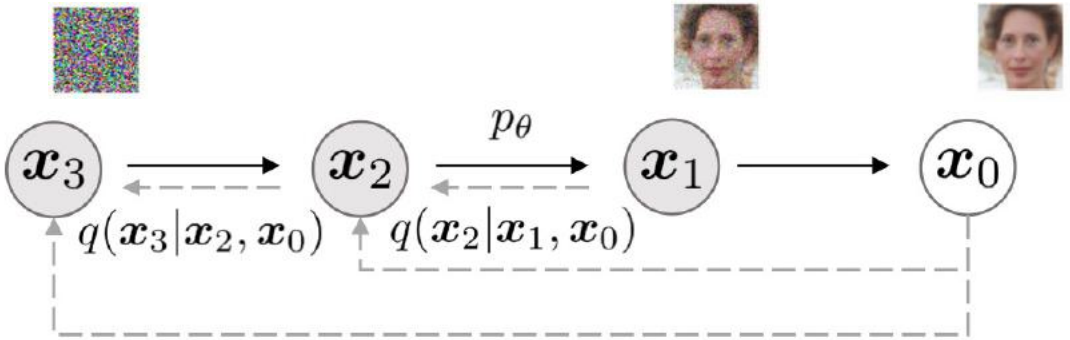


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Denoising Diffusion Implicit Model

- DDIM roughly sketches the final sample, then refine it with the reverse process
  - **Key idea:** Given $\mathbf{x}_t$, generate the rough sketch $\mathbf{x}_0$ and refine $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$



$\boldsymbol{x}_3$
$\boldsymbol{x}_2$
p_θ
$\boldsymbol{x}_1$
$\boldsymbol{x}_0$
$q(\boldsymbol{x}_3|\boldsymbol{x}_2, \boldsymbol{x}_0)$
$q(\boldsymbol{x}_2|\boldsymbol{x}_1, \boldsymbol{x}_0)$

  - **Formulation:** Define the forward distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ as

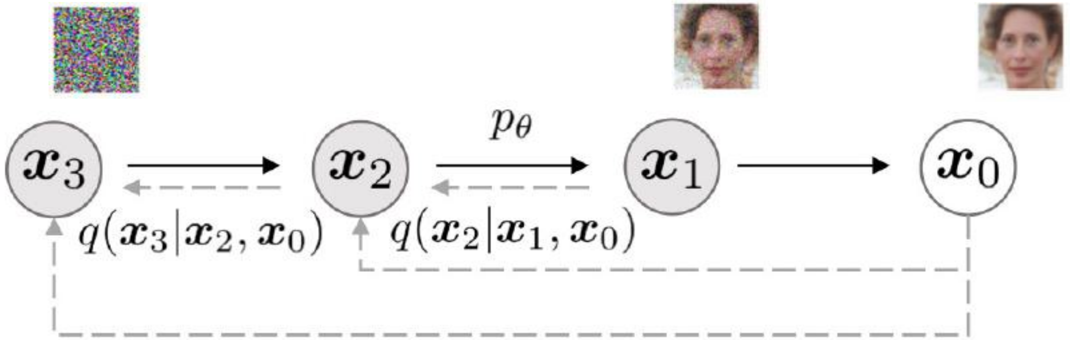
$$q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2 \boldsymbol{I}\right)$$

then, the forward process is derived from Bayes' rule

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

Denoising Diffusion Implicit Model

- DDIM roughly sketches the final sample, then refine it with the reverse process
  - **Key idea:** Given $\mathbf{x}_t$, generate the rough sketch $\mathbf{x}_0$ and refine $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$



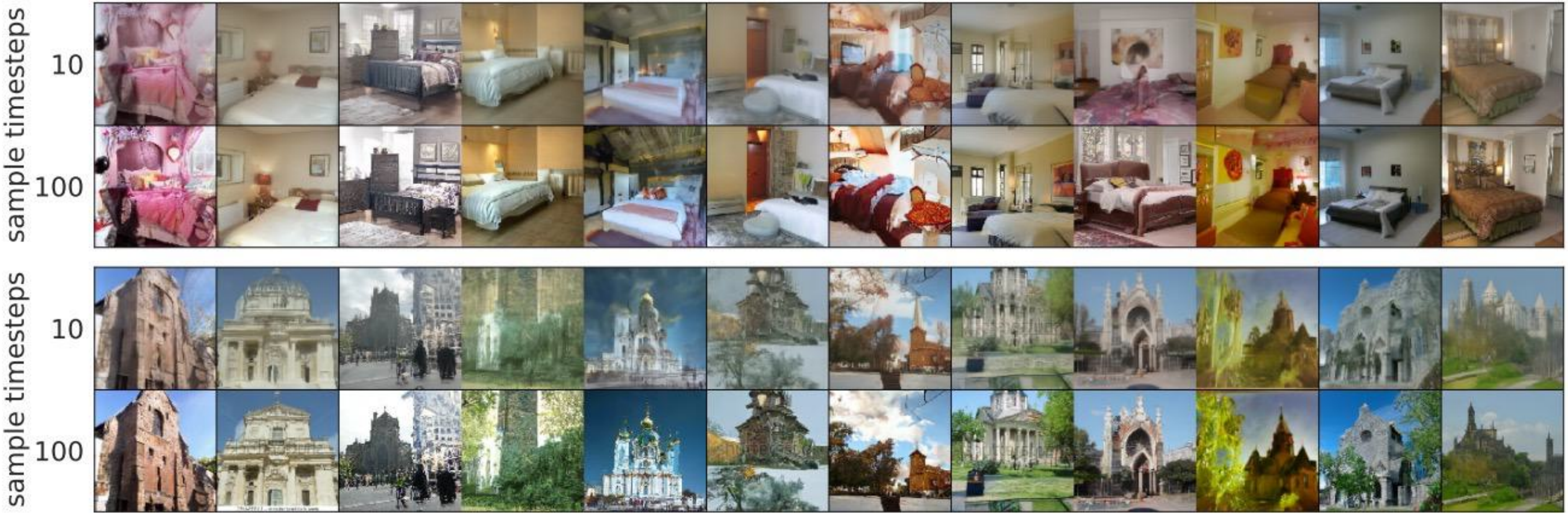
  - **Formulation:** Forward process is $q_\sigma(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \dfrac{q_\sigma(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)\, q_\sigma(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}{q_\sigma(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}$

    and reverse process is $\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\dfrac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\,\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{”predicted } \boldsymbol{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{“direction pointing to } \boldsymbol{x}_t\text{ ”}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$

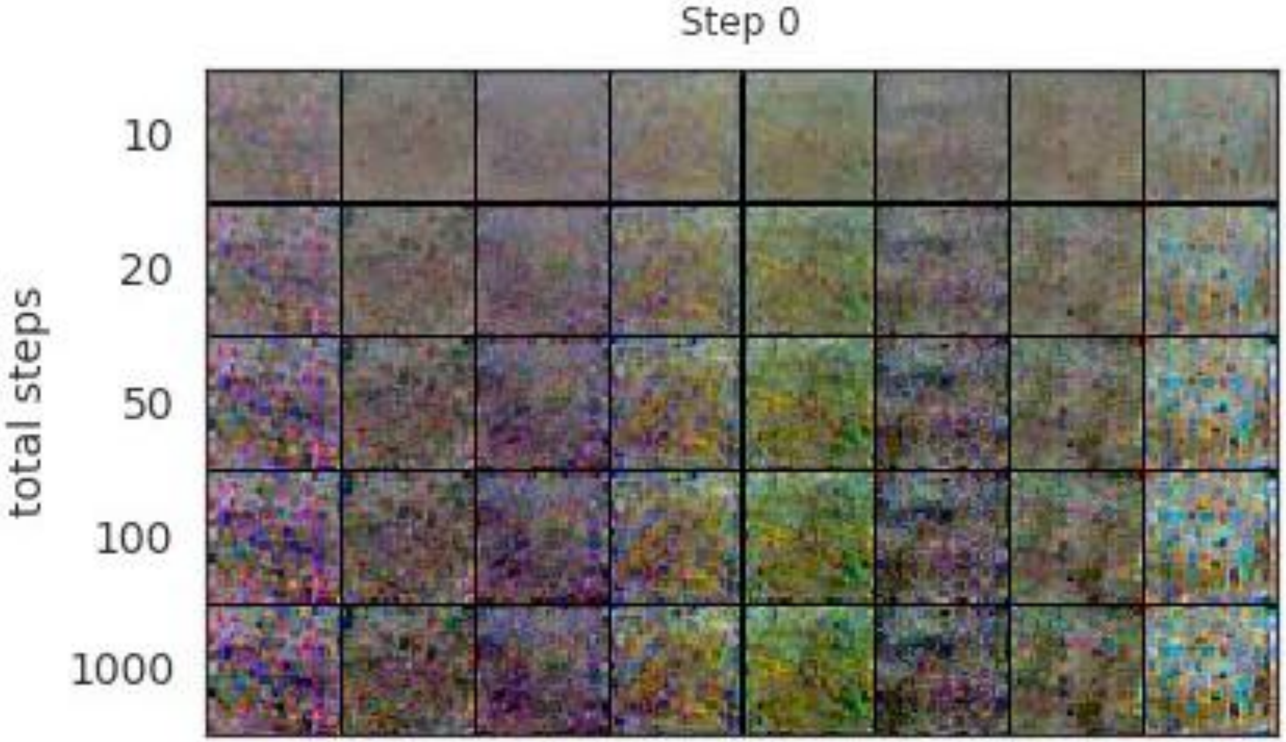
  - **Training:** The variational lower bound of DDIM is identical to the one of DDPM
    - It is surprising since the forward/reverse formulation is totally different

Denoising Diffusion Implicit Model

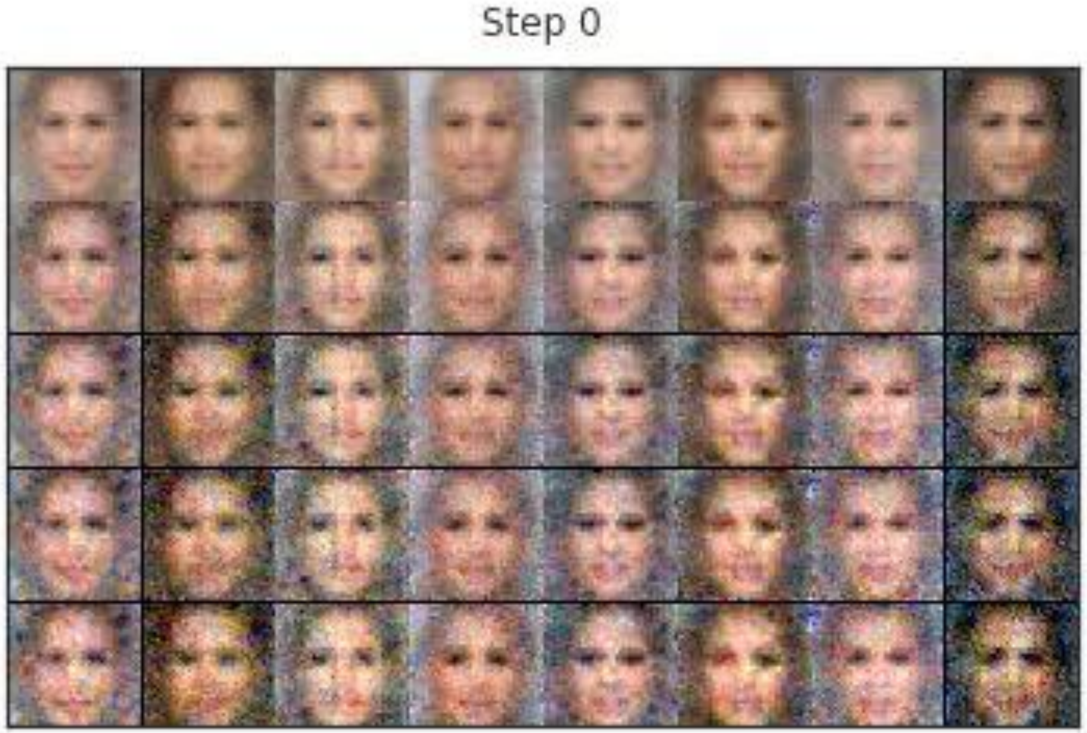
- DDIM significantly reduces the sampling steps of diffusion model
  - Creates the outline of the sample after only 10 steps (DDPM needs hundreds)



Denoising Diffusion Implicit Model

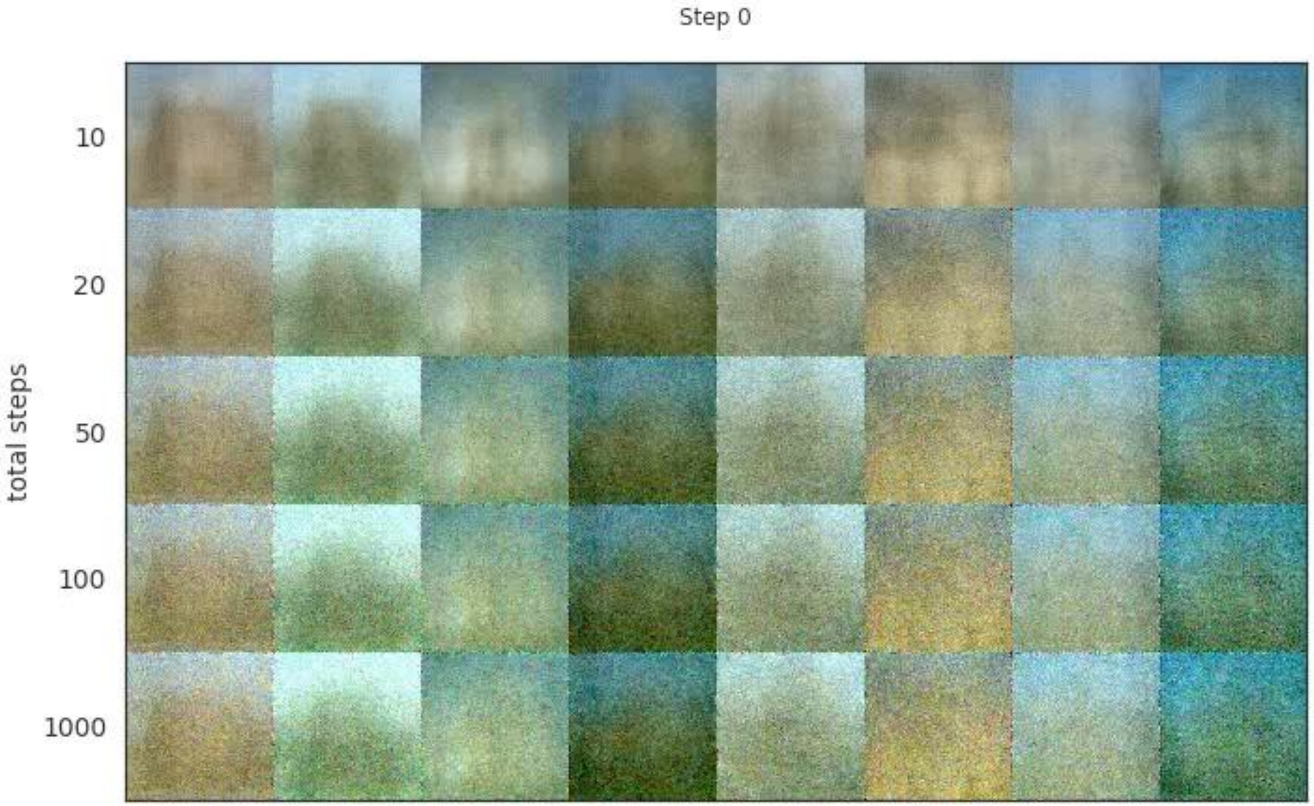


Generating CIFAR10 samples

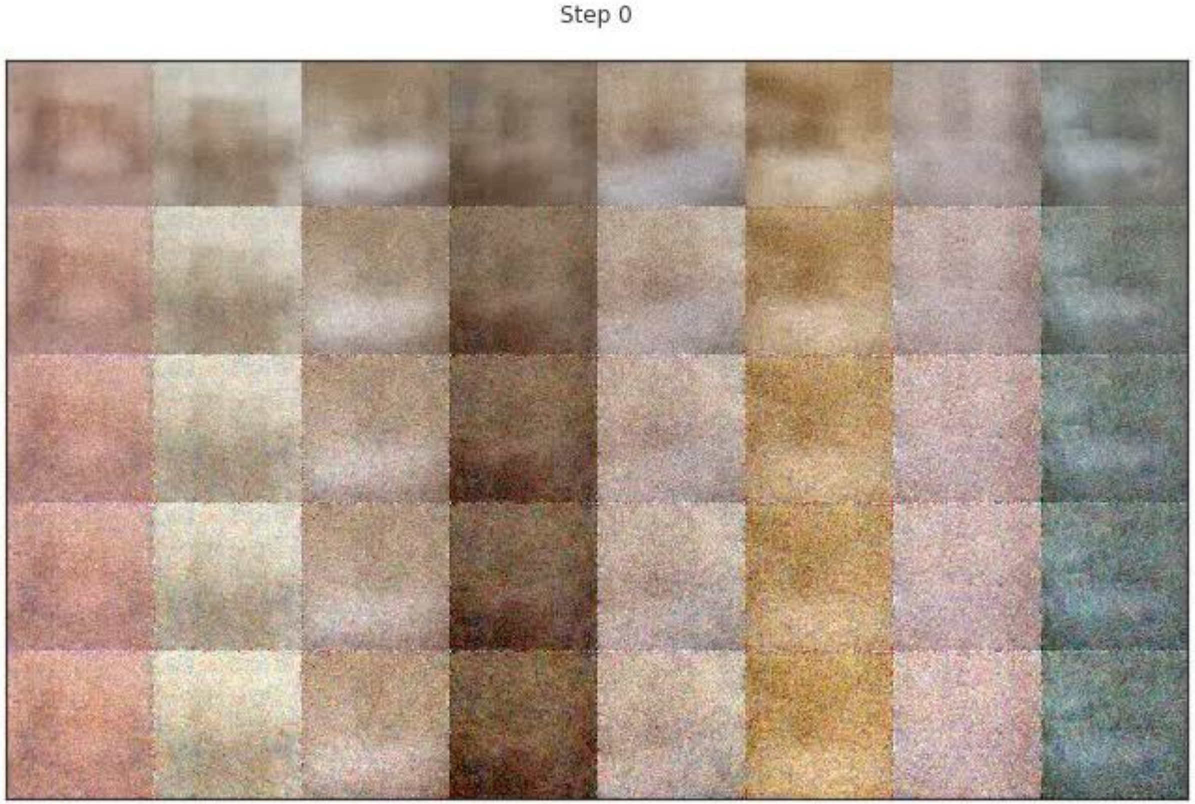


Generating CelebA samples

Denoising Diffusion Implicit Model



Generating LSUN Church samples



Generating LSUN Bedroom samples

Denoising Diffusion Implicit Model

- DDIM significantly reduces the sampling steps of diffusion model
  - Creates the outline of the sample after only 10 steps (DDPM needs hundreds)

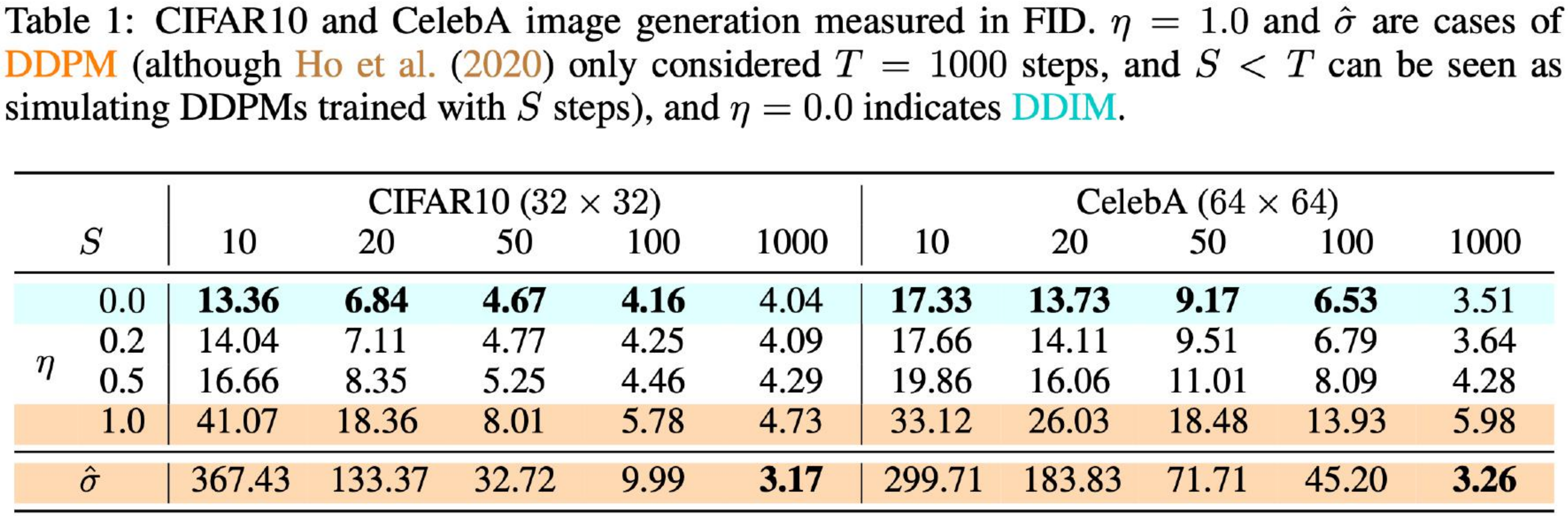


Table 1: CIFAR10 and CelebA image generation measured in FID. $\eta = 1.0$ and $\hat{\sigma}$ are cases of DDPM (although Ho et al. (2020) only considered $T = 1000$ steps, and $S < T$ can be seen as simulating DDPMs trained with S steps), and $\eta = 0.0$ indicates DDIM.

		CIFAR10 (32×32)					CelebA (64×64)				
	S	10	20	50	100	1000	10	20	50	100	1000
η	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**

Diffusion Models for Image Generation

- Beat BigGAN and StyleGAN on generating high-resolution images

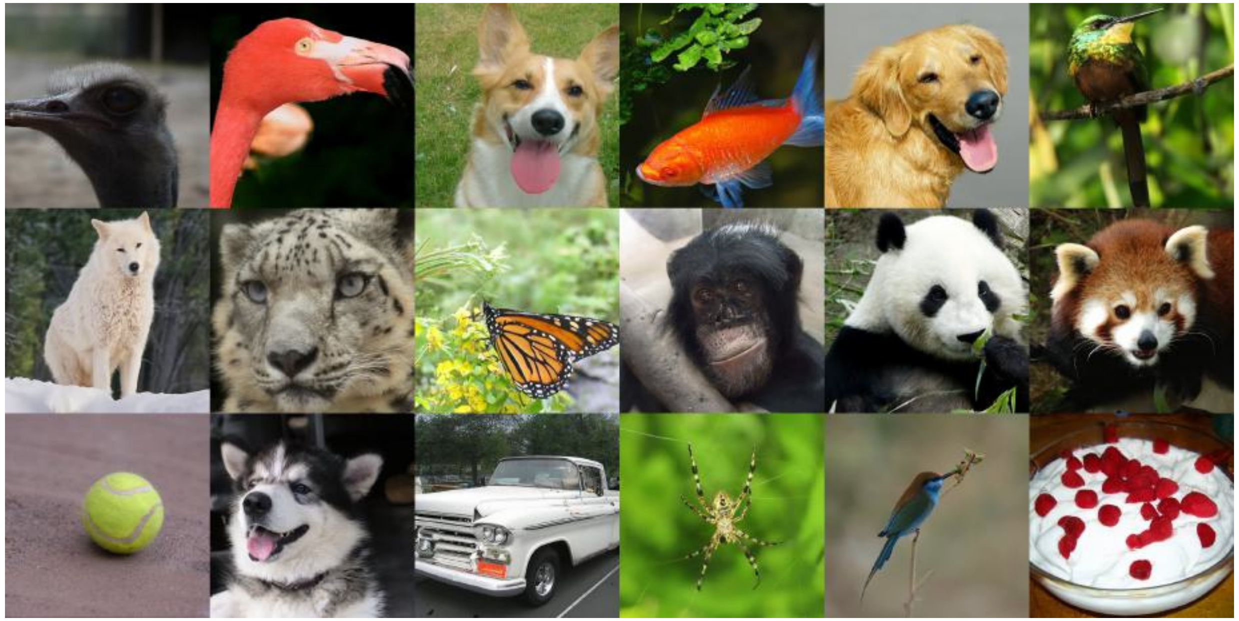
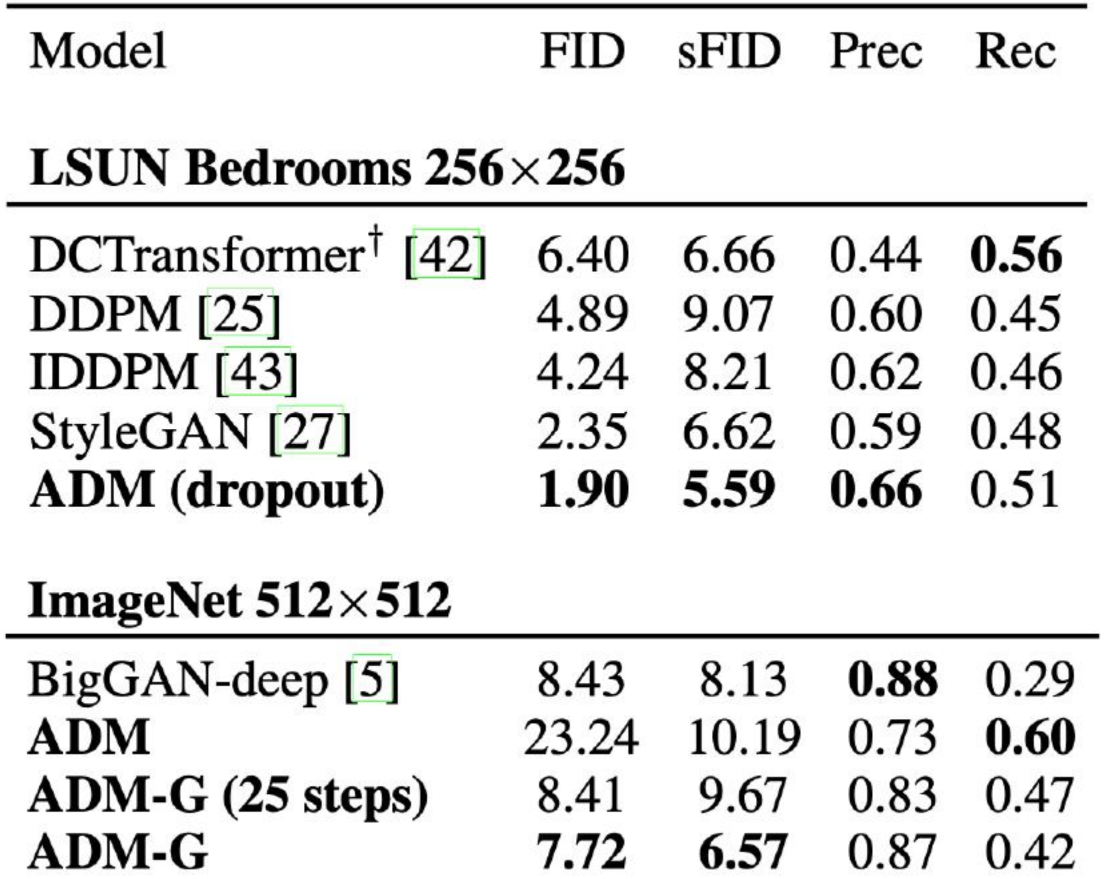


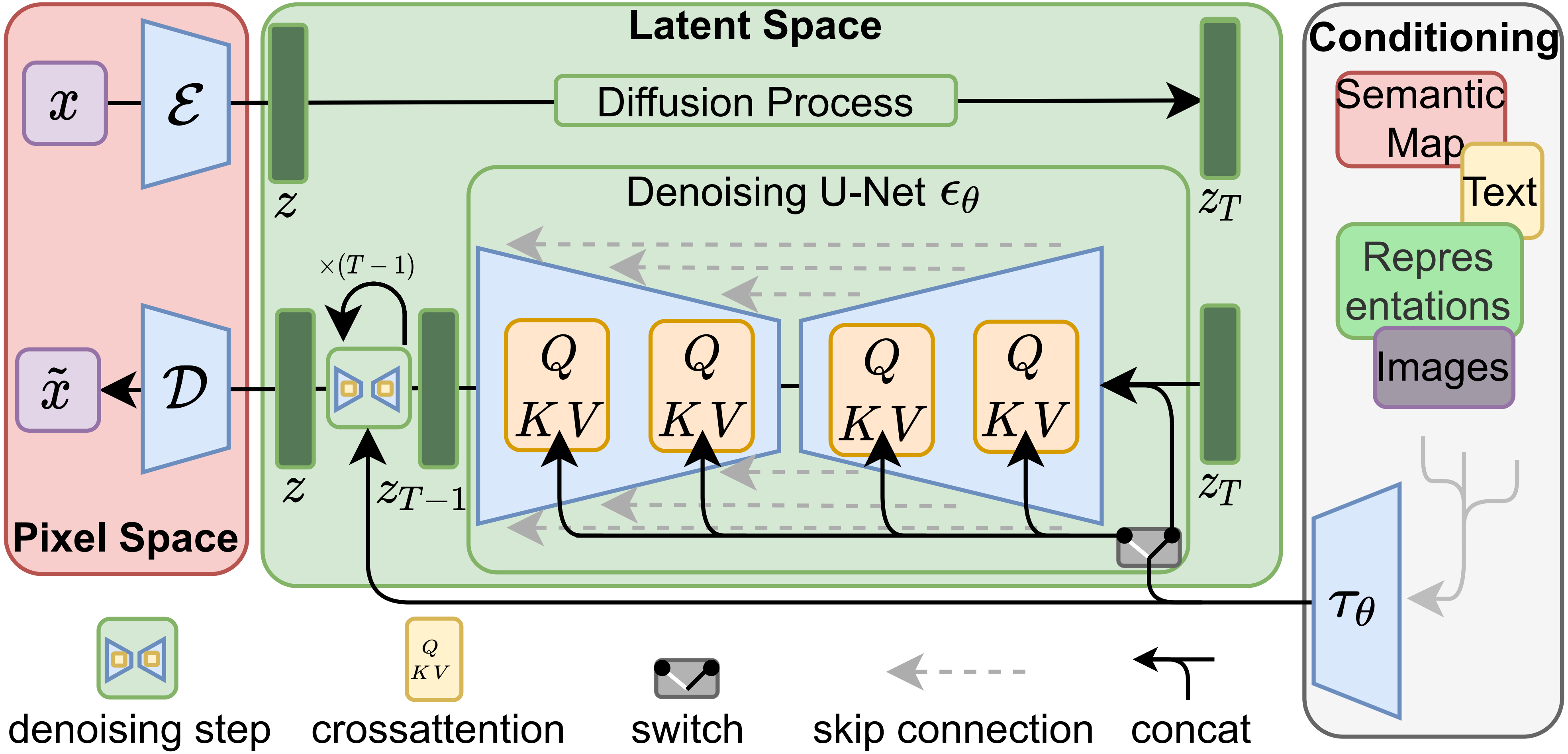
Figure 1: Selected samples from our best ImageNet 512×512 model (FID 3.85)



Model	FID	sFID	Prec	Rec
LSUN Bedrooms 256×256				
DCTransformer† [42]	6.40	6.66	0.44	**0.56**
DDPM [25]	4.89	9.07	0.60	0.45
IDDPM [43]	4.24	8.21	0.62	0.46
StyleGAN [27]	2.35	6.62	0.59	0.48
ADM (dropout)	**1.90**	**5.59**	**0.66**	0.51
ImageNet 512×512				
BigGAN-deep [5]	8.43	8.13	**0.88**	0.29
ADM	23.24	10.19	0.73	**0.60**
ADM-G (25 steps)	8.41	9.67	0.83	0.47
ADM-G	**7.72**	**6.57**	0.87	0.42

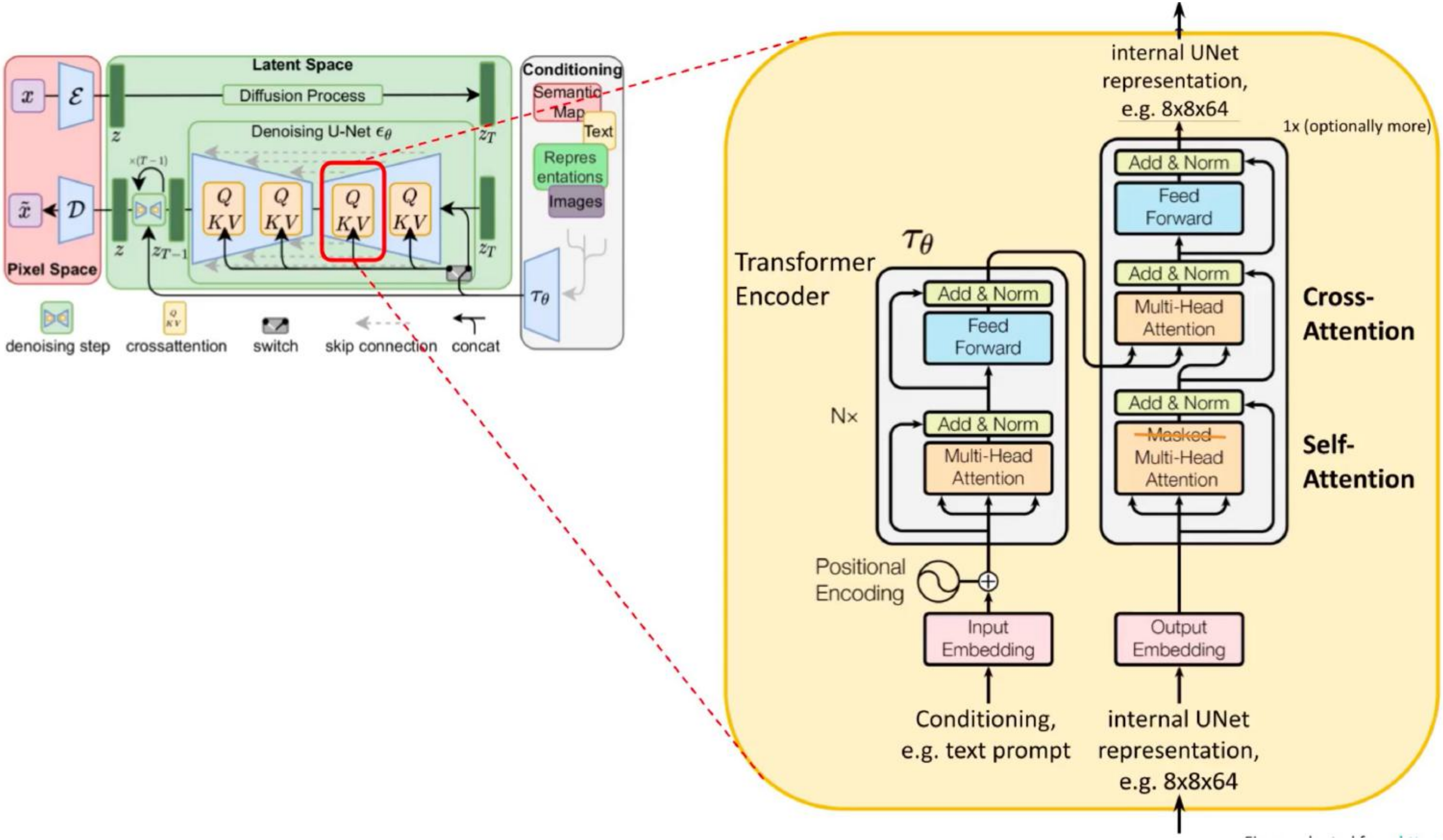
Latent Diffusion Models

Autoencoder with KL or VQ regularization



[Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022]

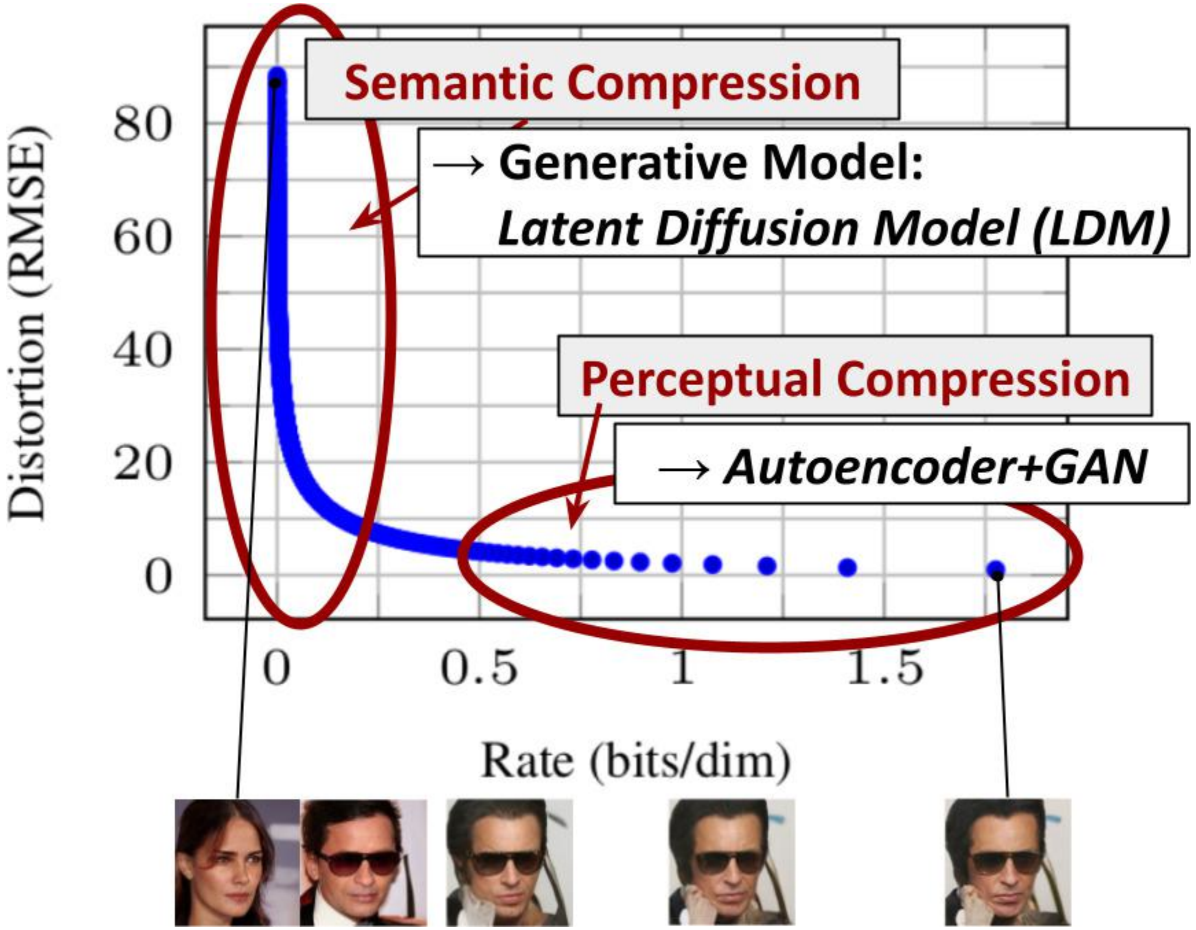
Latent Diffusion Models



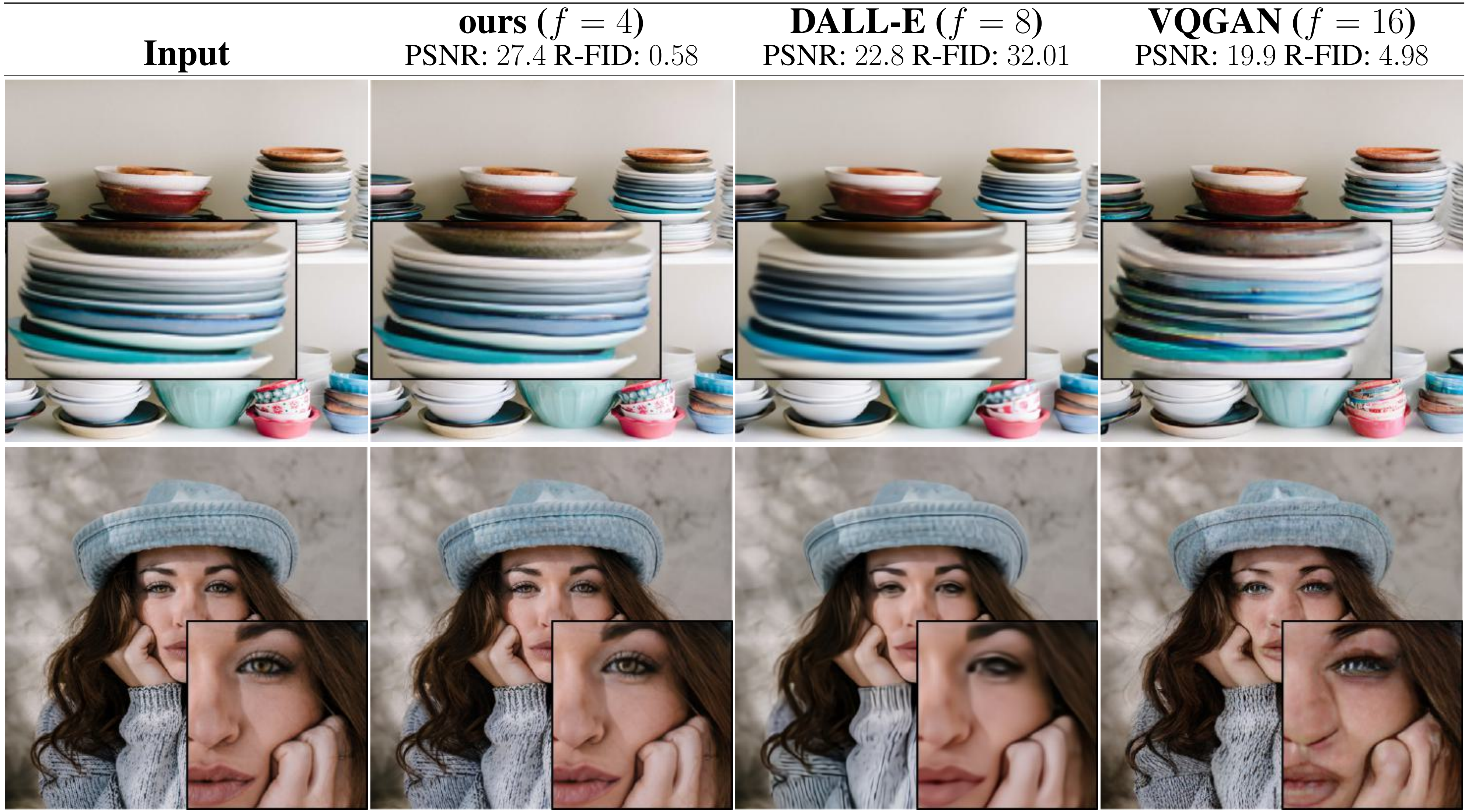
Latent Diffusion Models

- Why latent space?

- find perceptually equivalent space (to pixel space)
- efficient training
- fast sampling
- one-step decoding to image space

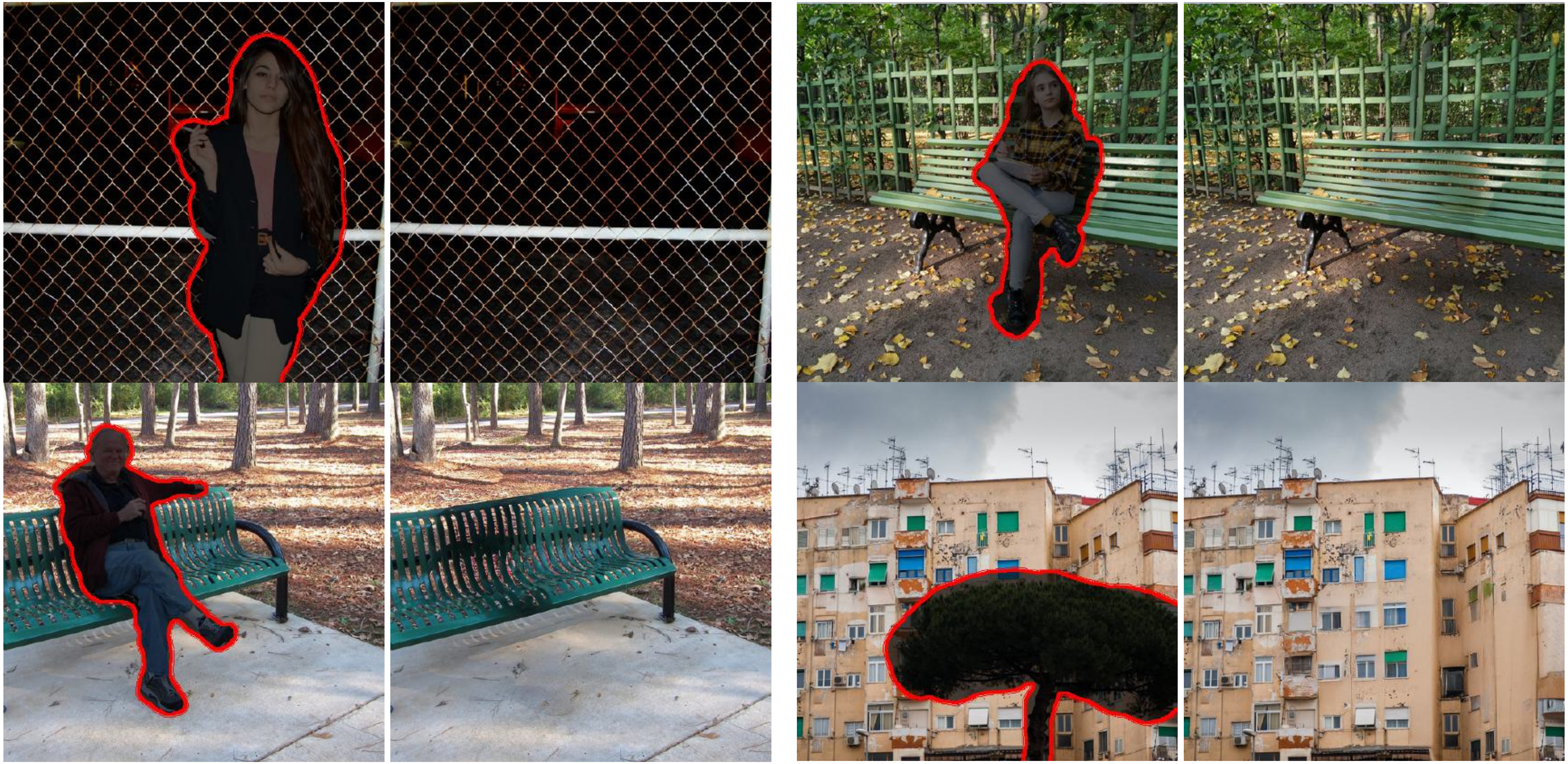


Tuning Compression Ratios


Input
ours ($f = 4$)
PSNR: 27.4 R-FID: 0.58
DALL-E ($f = 8$)
PSNR: 22.8 R-FID: 32.01
VQGAN ($f = 16$)
PSNR: 19.9 R-FID: 4.98

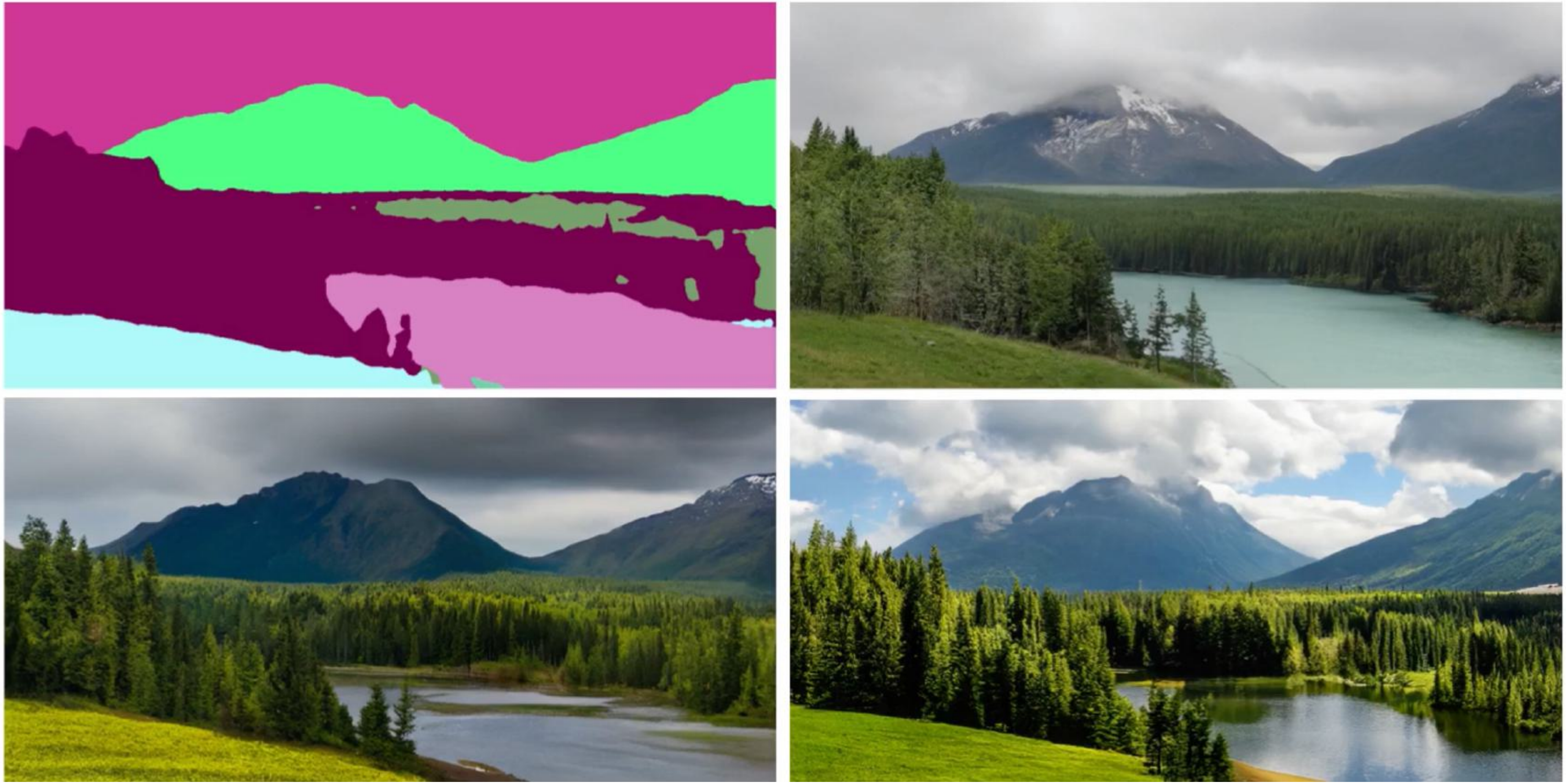
256 x 256　　64 x 64　　32 x 32　　16 x 16

Image Inpainting



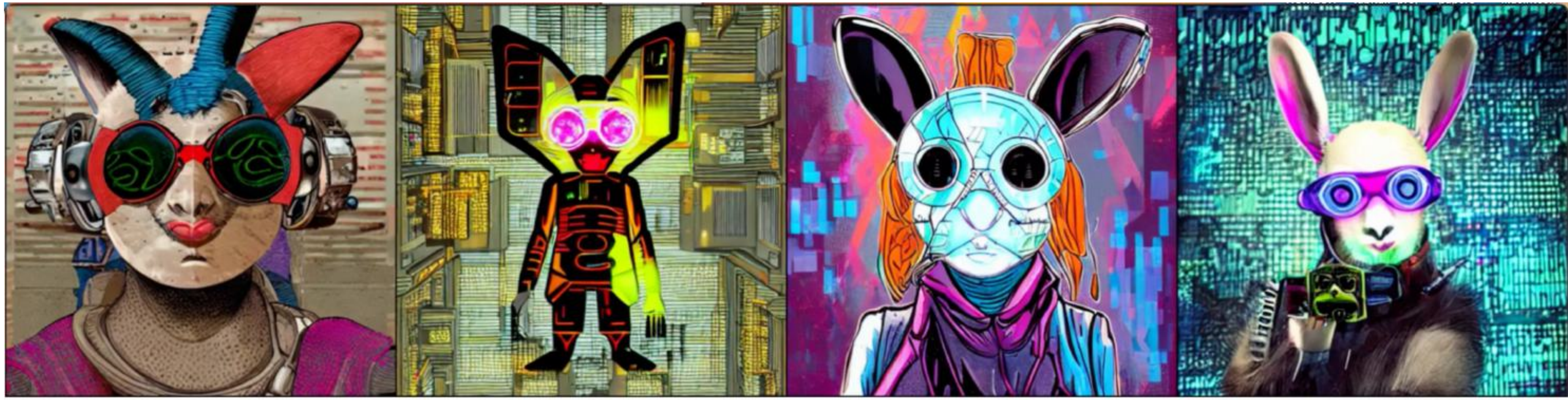
["LDM", Rombach et al., 2022]

Semantic Image Synthesis



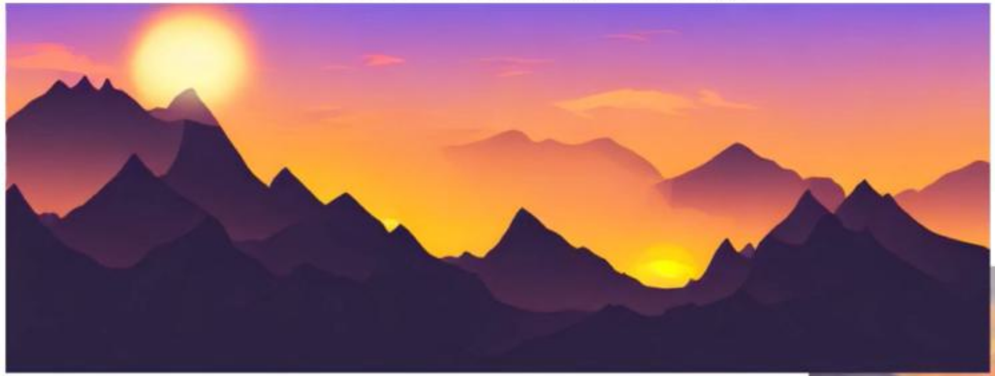
["LDM", Rombach et al., 2022]

Generating Images from Text

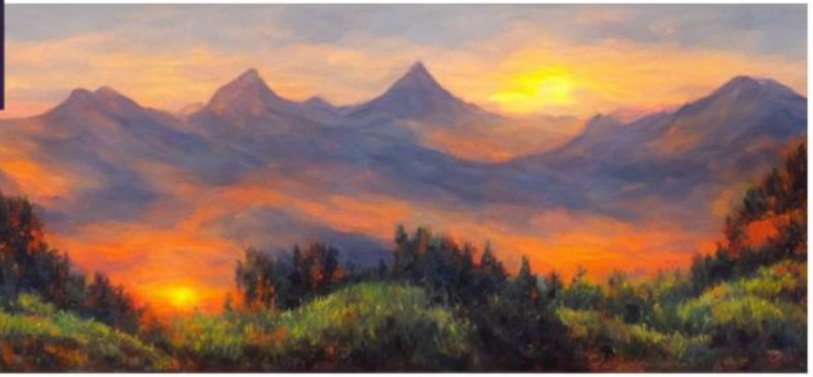


"a portrait of a cyberpunk rabbit, trending on artstation"

"A sunset over a mountain, vector image"



"A sunset over a mountain, oil on canvas"



["LDM", Rombach et al., 2022]

Generating Images from Text

a teddy bear on a skateboard in times square

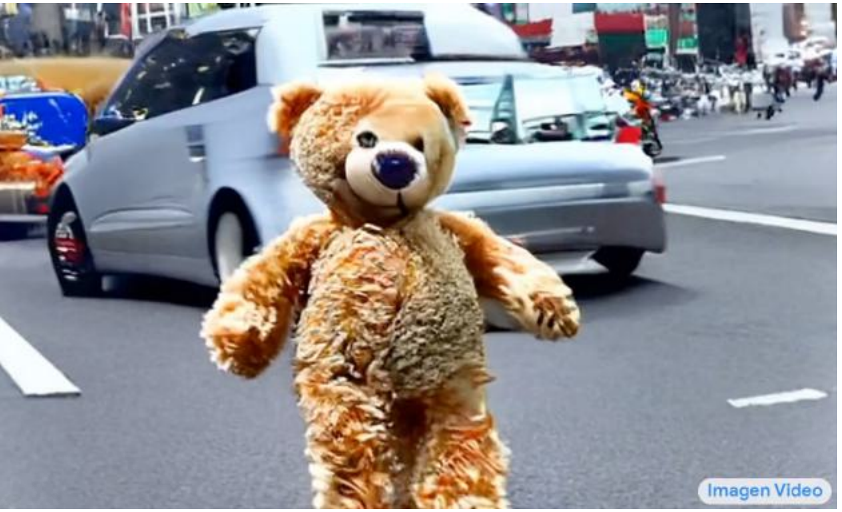
A photo of Michelangelo’s sculpture of David wearing headphones djing

“A sea otter with a pearl earring” by Johannes Vermeer

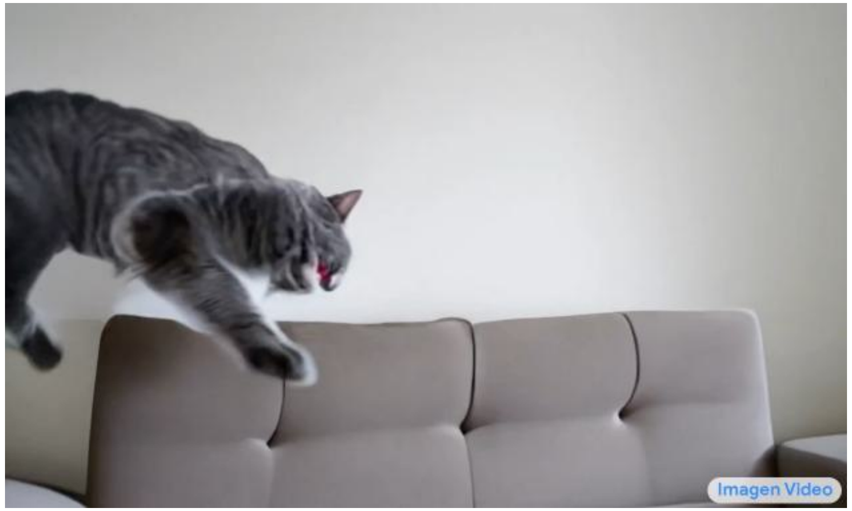
3D render of a cute tropical fish in an aquarium on a dark blue background, digital art

[“DALL-E 2”, Ramesh et al., 2022]

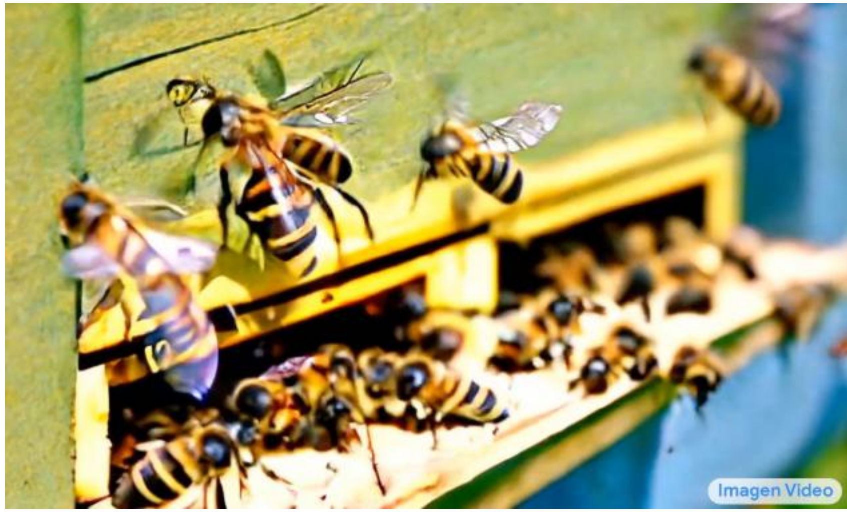
Generating Videos from Text



A teddy bear
running in New York City



A british shorthair
jumping over a coach



A swarm of bees
flying around their hive

[Ho et al., "Imagen Video", 2022]

Image Editing

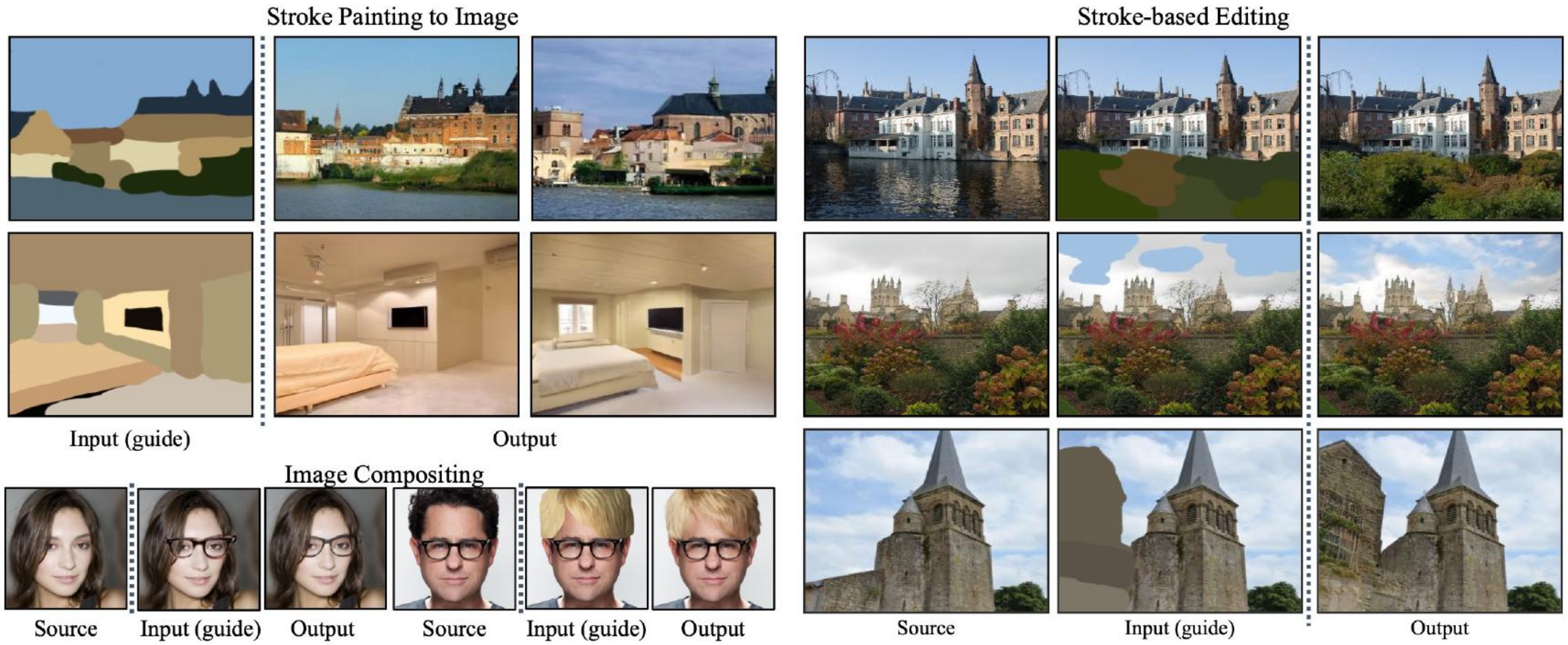
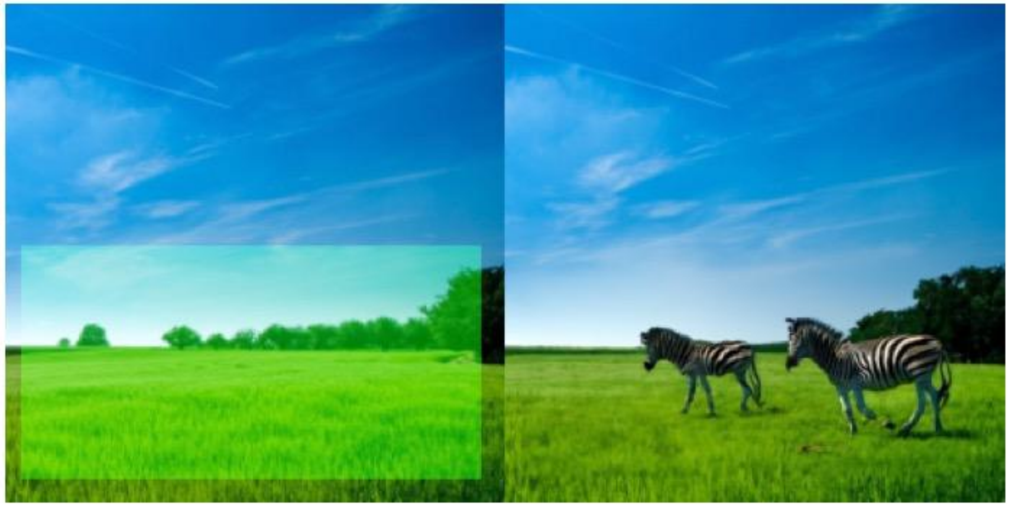
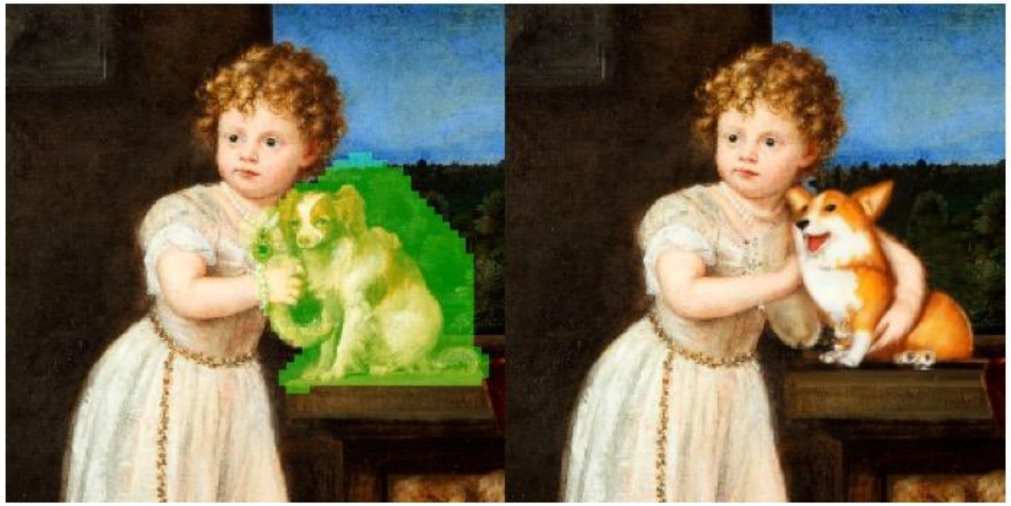


Figure 1: Stochastic Differential Editing (SDEdit) is a **unified** image synthesis and editing framework based on stochastic differential equations. SDEdit allows stroke painting to image, image compositing, and stroke-based editing **without** task-specific model training and loss functions.

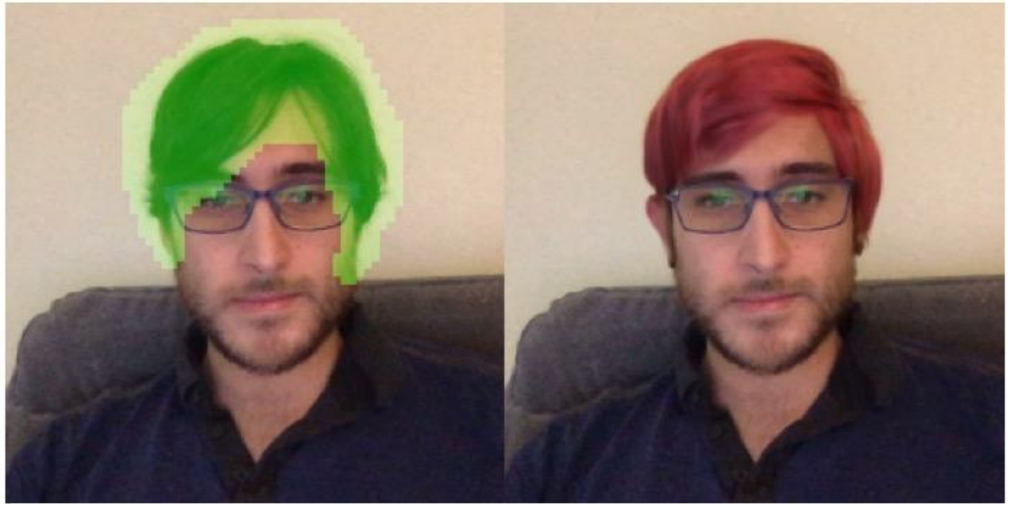
Image Editing



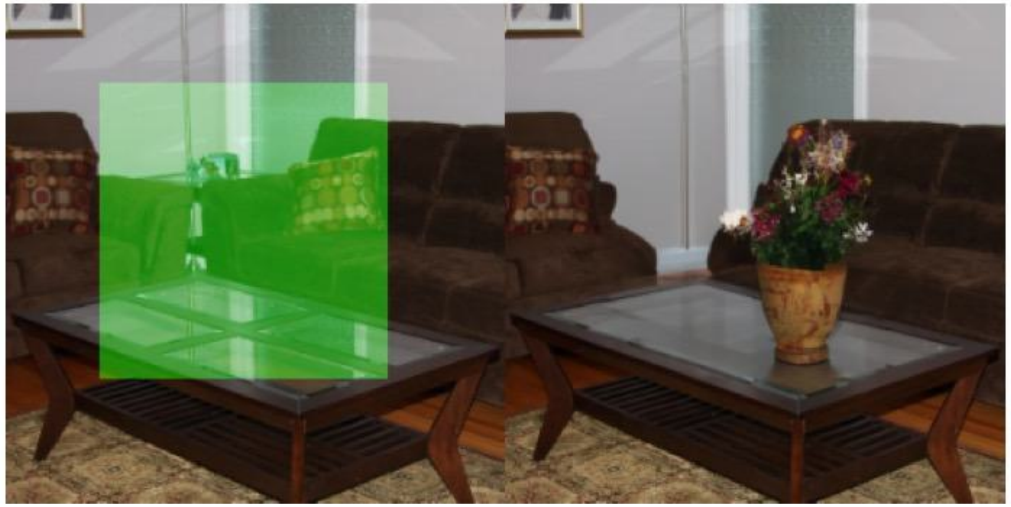
“zebras roaming in the field”



“a girl hugging a corgi on a pedestal”

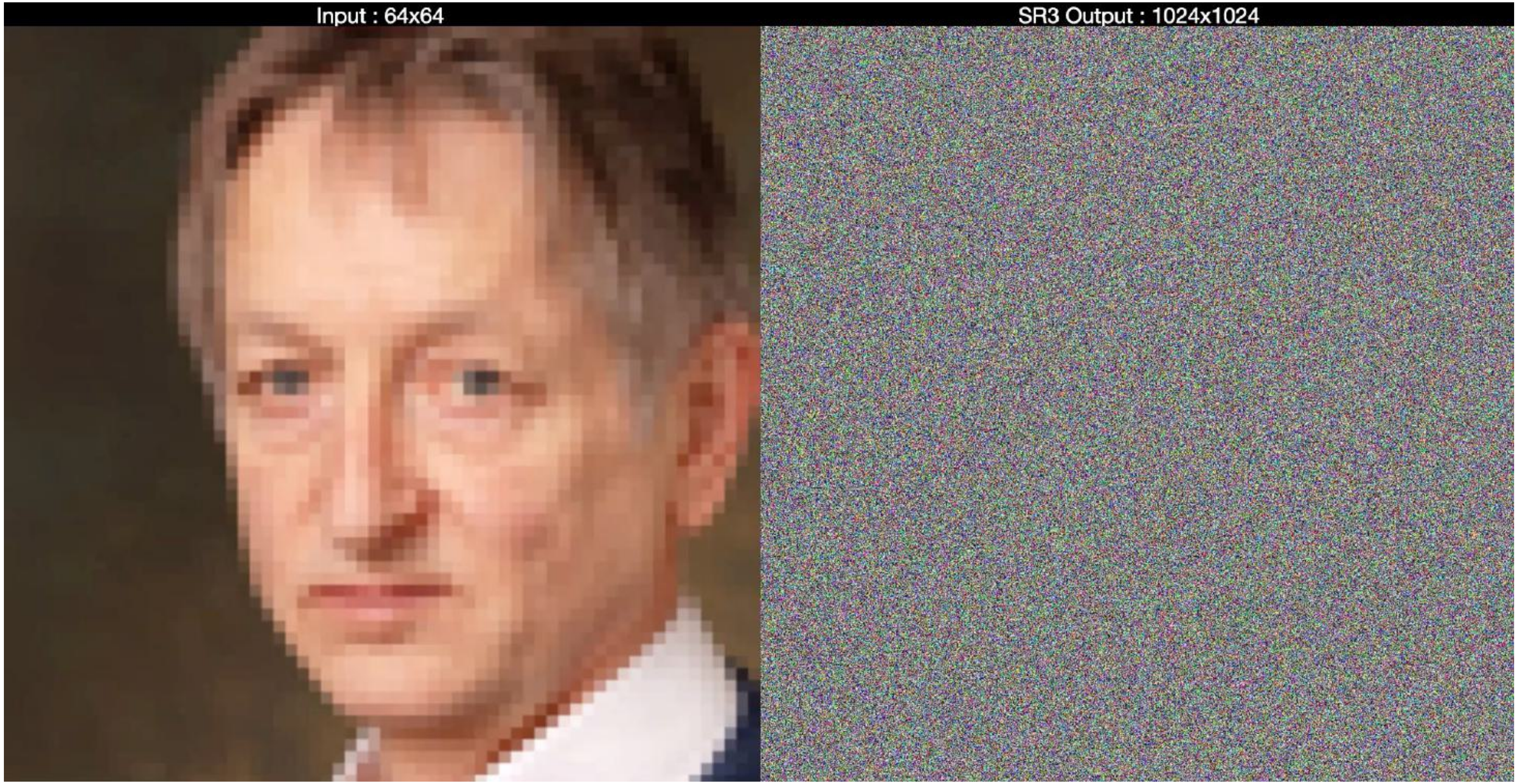


“a man with red hair”



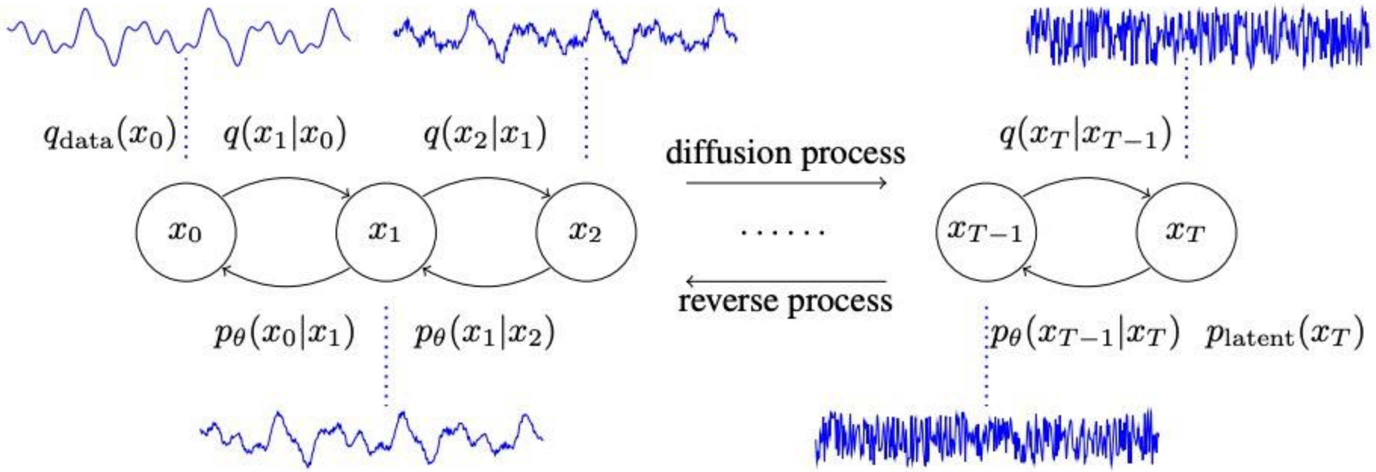
“a vase of flowers”

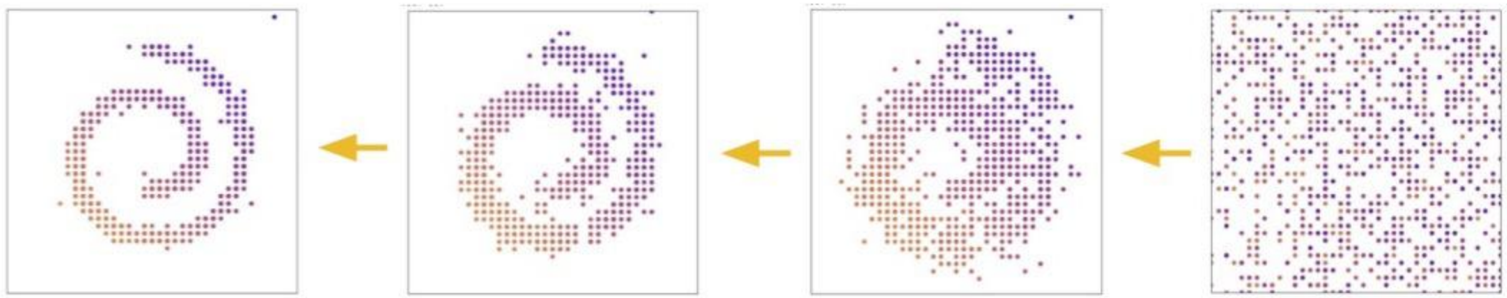
Super Resolution



Results of a SR3 model (64×64 → 512×512), trained on FFHQ, and applied to images outside of the training set.

Diffusion Models are also effective for non-visual domains





generated at t=0

t=T/4

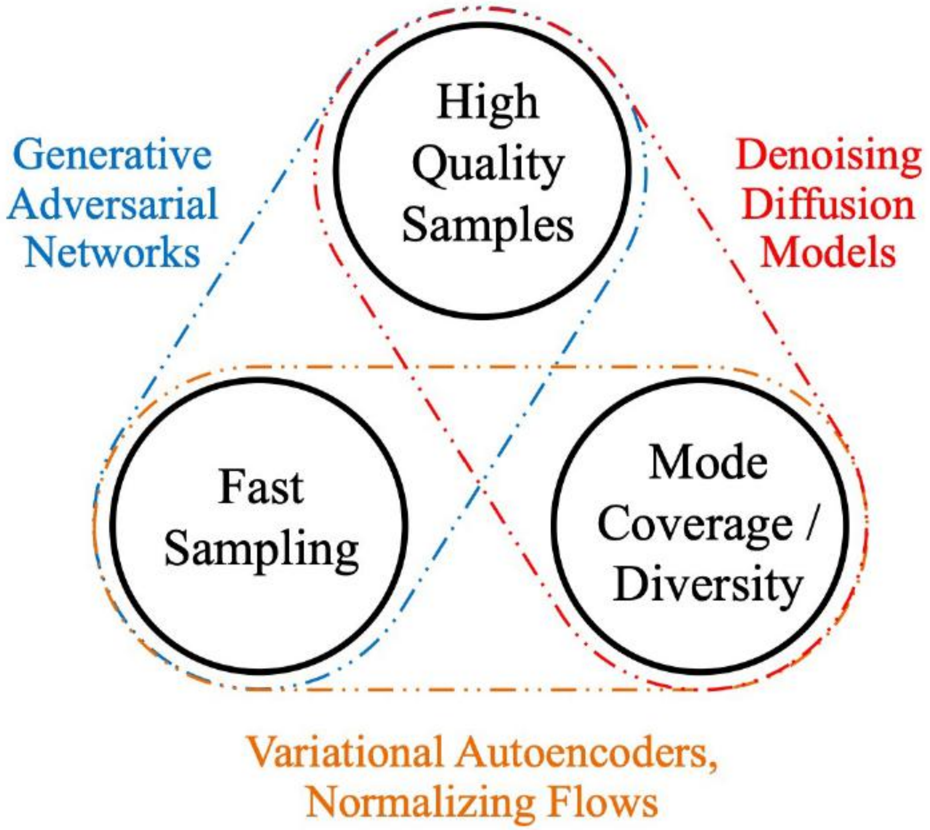
t=T/2

t=T (stationary)

https://github.com/heejkoo/Awesome-Diffusion-Models

Diffusion Model is All We Need?

- Trilemma of generative models: Quality vs. Diversity vs. Speed
  - Diffusion model produces diverse and high-quality samples, but generations is slow


High Quality Samples
Generative Adversarial Networks
Denoising Diffusion Models
Fast Sampling
Mode Coverage / Diversity
Variational Autoencoders, Normalizing Flows

Next lecture:
Self-Supervised Learning