AIR5101 / CIE6021

Generative AI

Lecture 2: Variational Autoencoder

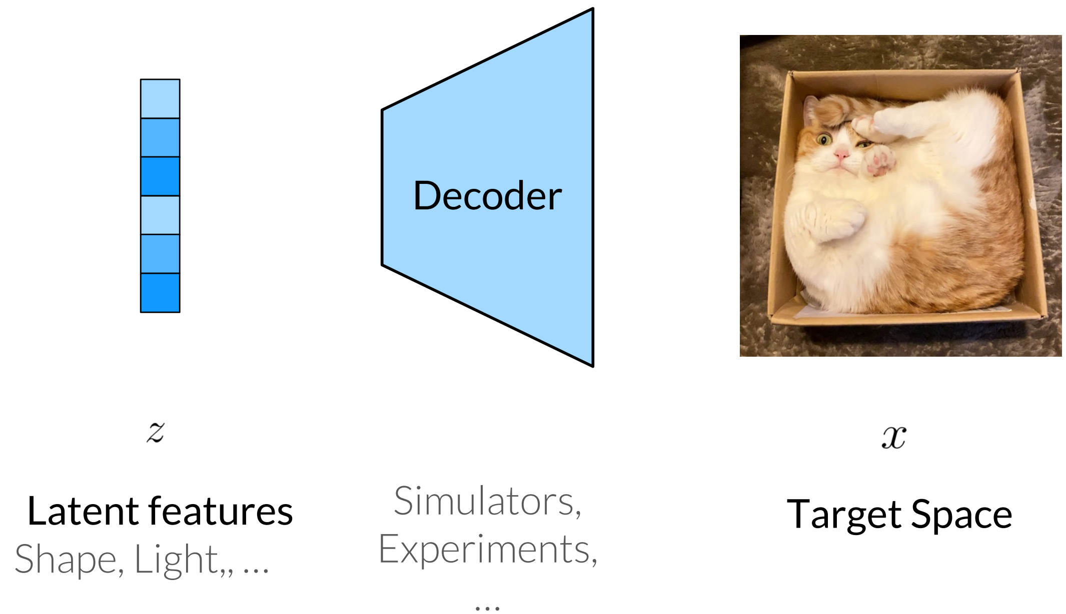
Instructor: Zhen Liu

Spring 2025, CUHK-Shenzhen

Some announcements

- First assignment to be released by mid of next week
- Part of the assignment: write a trainable VAE on toy datasets with PyTorch
  - Start early to see if you are comfortable in writing Python and learning PyTorch
- Find your project teammates and brainstorm ideas
- Paper list for presentations to be released by next week

Latent Features



Why Latent Spaces?

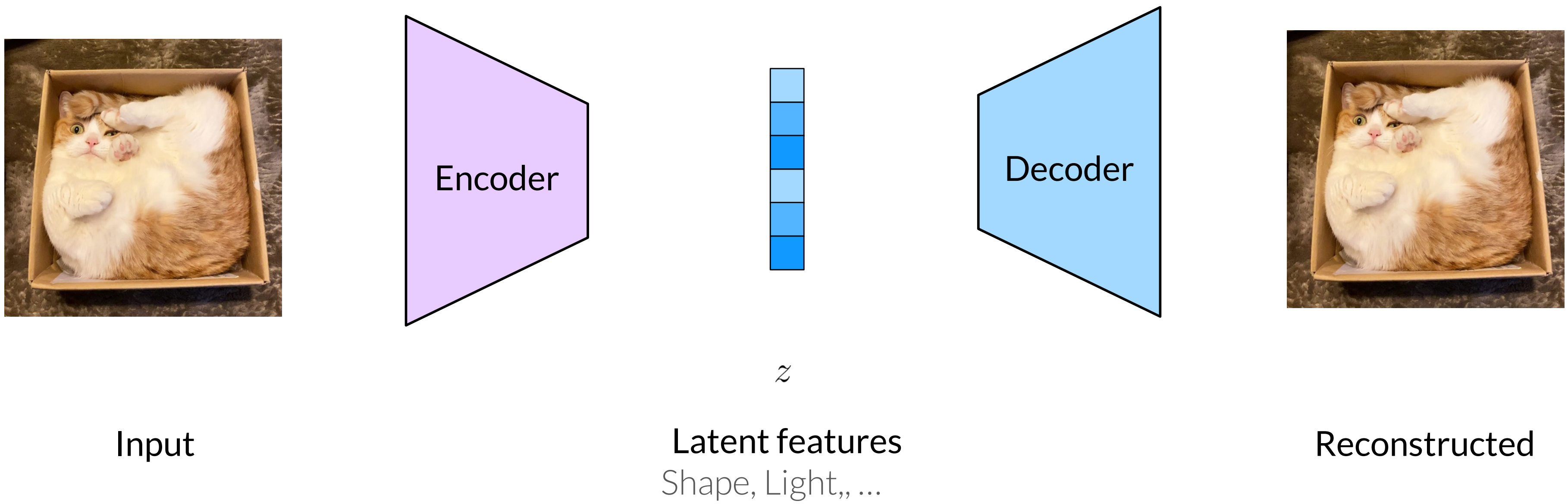
Manifold Hypothesis:

High-dimensional real-world data like images and videos
live in a low-dimensional manifold

Natural to learn distributions on this manifold instead.

Autoencoder

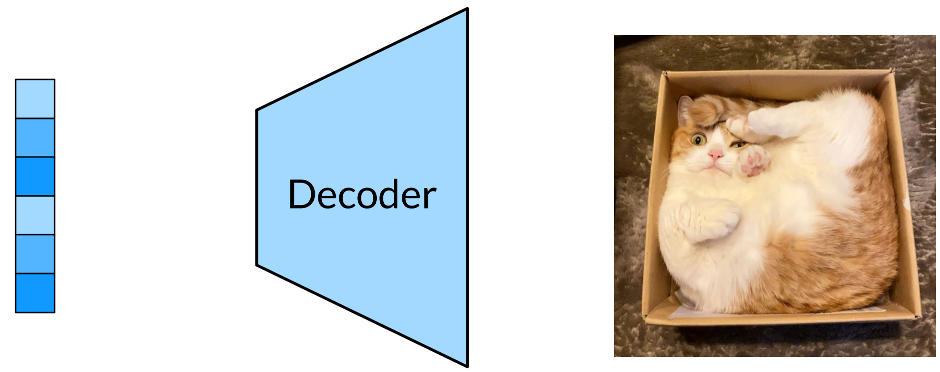
Decoder: only see finite number of latent features



Issue: how to sample in the latent space?

Instead, study the problem under a probabilistic framework

Latent Variable Model



$z \sim p(z)$	$p_\theta(x\|z)$	$p_\theta(x)$
Latent features Shape, Pose, …	Parametrized Conditional Prob	Target Distribution

Q: where does the latent feature come from?

Recall: Maximum Likelihood Estimation

Optimize the “distance” between data and model

With “distance” as KL divergence,

$$
\begin{aligned}
& \min_{\theta} D_{\mathrm{KL}}(p_{\mathcal{D}} \| p_{\theta}) \\
= & \min_{\theta} \mathop{\mathbb{E}}_{x \sim p_{\mathcal{D}}} [\log p_{\mathcal{D}}(x) - \log p_{\theta}(x)] \qquad \text{(Definition of KL divergence)} \\
= & \max_{\theta} \mathop{\mathbb{E}}_{x \sim p_{\mathcal{D}}} \log p_{\theta}(x)
\end{aligned}
$$

MLE for Latent Variable Model

Naïve attempt:

$$
\begin{aligned}
&\max_\theta \mathop{\mathbb{E}}_{x\sim p_\mathcal{D}} \log p_\theta(x) \\
&= \max_\theta \mathop{\mathbb{E}}_{x\sim p_\mathcal{D}} \log \int_z \; p_\theta(x|z)\; p(z)\; dz \\
&\geq \max_\theta \mathop{\mathbb{E}}_{x\sim p_\mathcal{D}} \int_z \log p_\theta(x|z) p(z) dz \qquad \text{(Jensen's Inequality)} \\
&= \max_\theta \mathop{\mathbb{E}}_{x\sim p_\mathcal{D}, z\sim p(z)} \log p_\theta(x|z)
\end{aligned}
$$

Very bad estimate. What is the gap?

MLE for Latent Variable Model

Decompose the log-likelihood:

$$
\begin{aligned}
&\log p_\theta(x) \\
&= \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} && \text{(Bayes' Rule)} \\
&= \log p_\theta(x|z) + \log p(z) - \log p_\theta(z|x)
\end{aligned}
$$

MLE for Latent Variable Model

Decompose the log-likelihood:

$$
\begin{aligned}
&\log p_\theta(x) \\
&= \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} && \text{(Bayes' Rule)} \\
&= \log p_\theta(x|z) + \log p(z) - \log p_\theta(z|x)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{p(z)} \log p_\theta(x) \\
&= \mathop{\mathbb{E}}_{p(z)} \log p_\theta(x|z) + \mathop{\mathbb{E}}_{p(z)} \left[\log p(z) - \log p_\theta(z|x)\right] \\
&= \mathop{\mathbb{E}}_{p(z)} \log p_\theta(x|z) + \boxed{D_{\mathrm{KL}}\Big(p(z) \| p_\theta(z|x)\Big)} && \text{Huge gap!}
\end{aligned}
$$

Proposal Distribution

Instead, we sample with a proposal distribution $q(z)$

$$\mathop{\mathbb{E}}_{q(z)} \log p_\theta(x)$$

$$= \mathop{\mathbb{E}}_{q(z)} \Big[\log p_\theta(x|z) + \log p(z) - \log p_\theta(z|x) \Big]$$

$$= \mathop{\mathbb{E}}_{q(z)} \Big[\log p_\theta(x|z) \Big] + \big(\log p(z) - \log q(z)\big) + \big(\log q(z) - \log p_\theta(z|x)\big)\Big]$$

$$= \mathop{\mathbb{E}}_{q(z)} \Big[\log p_\theta(x|z) \Big] - \underbrace{D_{\mathrm{KL}}\Big(q(z) \| p(z)\Big)}_{\text{“distance” from prior}} + \underbrace{D_{\mathrm{KL}}\Big(q(z) \| p_\theta(z|x)\Big)}_{\text{Intractable}}$$

Proposal Distribution

$$\mathop{\mathbb{E}}_{q(z)} \log p_\theta(x)$$

$$= \underbrace{\mathop{\mathbb{E}}_{q(z)} \Big[\log p_\theta(x|z)\Big] - D_{\mathrm{KL}}\Big(q(z) \| p(z)\Big)}_{\text{Evidence Lower Bound (ELBO)}} + \underbrace{D_{\mathrm{KL}}\Big(q(z) \| p_\theta(z|x)\Big)}_{\text{Non-negative (Property of KL)}}$$

$$\geq \mathop{\mathbb{E}}_{q(z)} \Big[\log p_\theta(x|z)\Big] - D_{\mathrm{KL}}\Big(q(z) \| p(z)\Big)$$

Lower bound of log-likelihood

$\Rightarrow$ Something we can optimize

Encoder

Notice: we can pick any $q(z)$

The intractable $D_{\mathrm{KL}}\Big(q(z) \| p_\theta(z|x)\Big)$ implies we should pick

$$q_\phi(z|x)$$

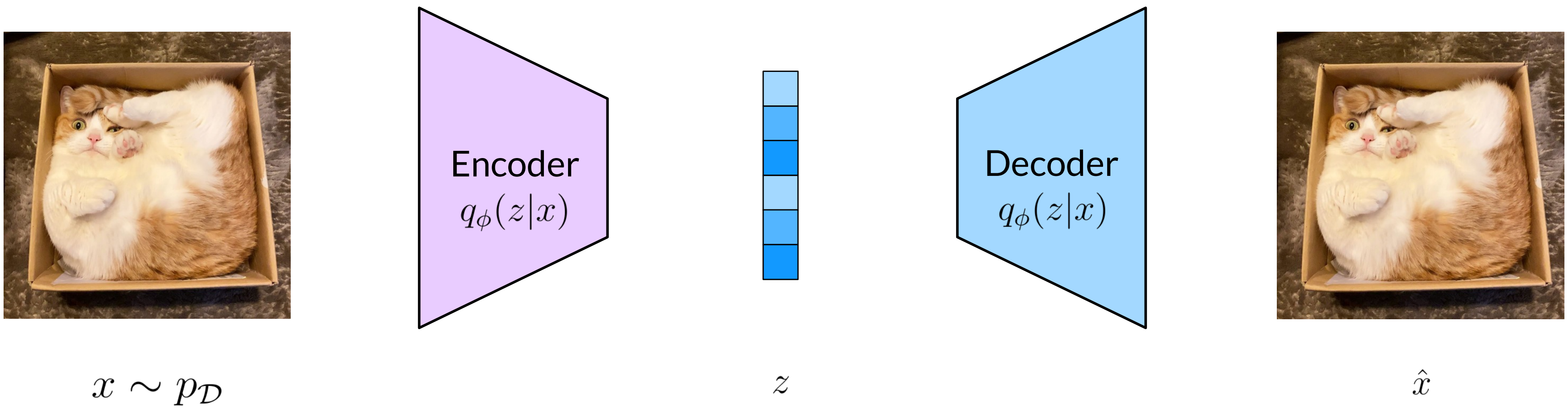
Encoding inputs into latents

With this encoder, our log-likelihood becomes

$$\underbrace{\mathop{\mathbb{E}}_{q_\theta(z|x)}\Big[\log p_\theta(x|z)\Big] - D_{\mathrm{KL}}\Big(q_\theta(z|x) \| p(z)\Big)}_{\text{ELBO with our encoder}} + D_{\mathrm{KL}}\Big(q_\theta(z|x) \| p_\theta(z|x)\Big)$$

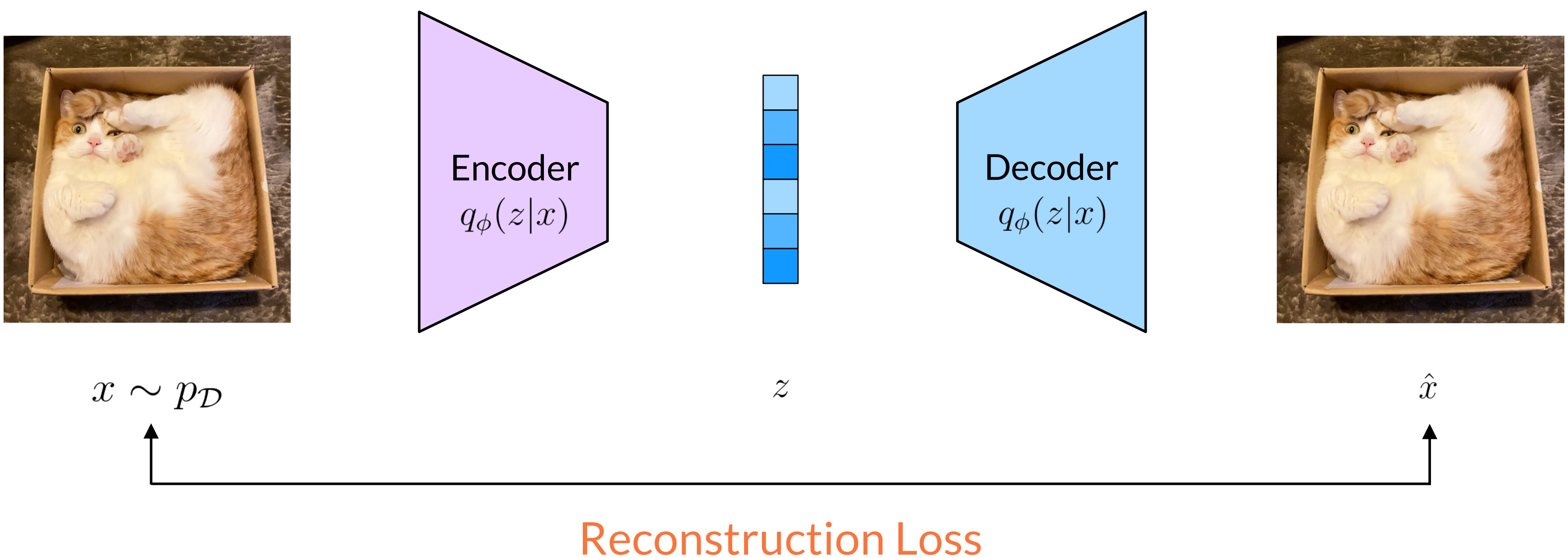
Variational Autoencoder (VAE)

$$\mathop{\mathbb{E}}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] - D_{\mathrm{KL}}\Big(q_\phi(z|x)\|p(z)\Big)$$



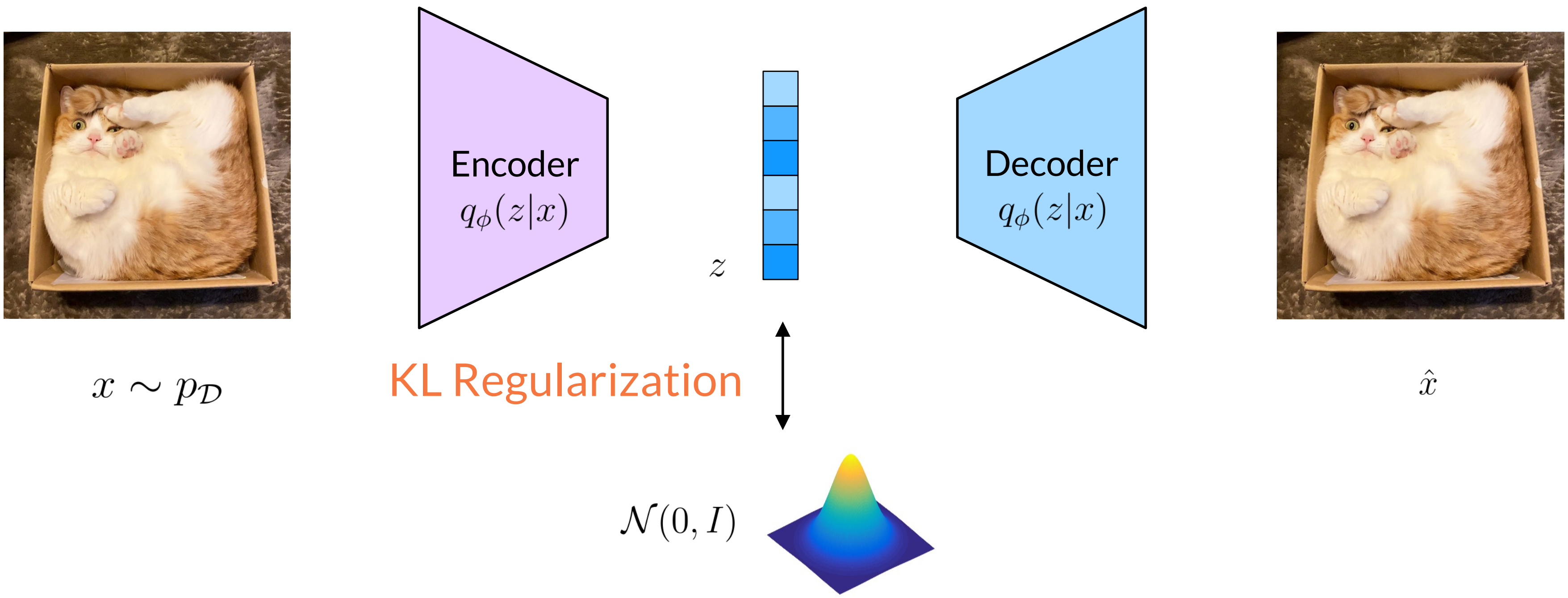
Variational Autoencoder (VAE)

$$\boxed{\mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} \Big[\log p_\theta(x|z) \Big]} - D_{\mathrm{KL}}\Big(q_\phi(z|x) \| p(z) \Big)$$



Variational Autoencoder (VAE)

$$\mathbb{E}_{z \sim q_\phi(z|x)} \Big[\log p_\theta(x|z) \Big] - \boxed{D_{\mathrm{KL}} \Big(q_\phi(z|x) \| p(z) \Big)}$$



Choice of Decoder

$$\mathbb{E}_{z \sim q_\phi(z|x)} \left[\boxed{\log p_\theta(x|z)} \right] - D_{\mathrm{KL}}\Big(q_\phi(z|x) \| p(z)\Big)$$

Typical choice: isotropic Gaussian

$$\log p_\theta(z|x) \propto \|z - g_\theta(x)\|^2 / (2\sigma_o^2)$$

Pick some fixed standard deviation ⇒ scaled L2 loss

Choice of Encoder

$$\mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} \Big[\log p_\theta(x|z) \Big] - D_{\mathrm{KL}} \Big(q_\phi(z|x) \| p(z) \Big)$$

We should be able to easily sample $q_\phi(z|x)$

Natural choice: Gaussian

Encoder now outputs $\quad f_\phi(x) \rightarrow \mu_\phi(x), \sigma_\phi(x)$

Analytical form of KL regularization (left as exercise)

$$D_{\mathrm{KL}} \Big(\mathcal{N}(\mu_\theta(x), \sigma^2_\theta(x)) \;\Big\|\; \mathcal{N}(0, I) \Big)$$

Reparametrization Trick

$$\mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x|z) \right]$$

How to use backprop to optimize the encoder?

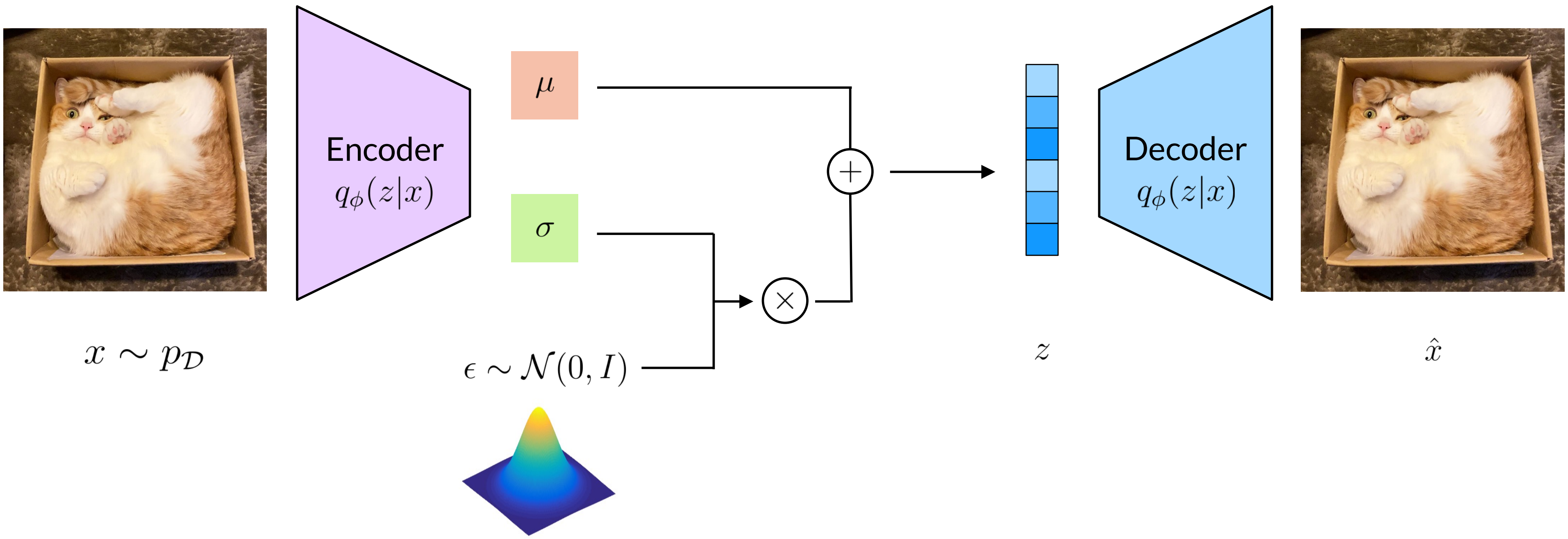
$$z \sim q_\phi(z|x)$$

$$||$$

$$z = \mu_\theta(x) + \sigma_\theta(x)\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

Stochasticity isolated to fixed distribution of epsilon

Reparametrization Trick



Full Training Loss

ELBO for a single datapoint

$$\mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} \Big[\log p_\theta(x|z) \Big] - D_{\mathrm{KL}}\Big(q_\phi(z|x) \| p(z) \Big)$$

ELBO for the whole dataset

$$\mathop{\mathbb{E}}_{x \sim p_{\mathcal{D}}} \left[\mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} \Big[\log p_\theta(x|z) \Big] - D_{\mathrm{KL}}\Big(q_\phi(z|x) \| p(z) \Big) \right]$$

Generation with VAEs

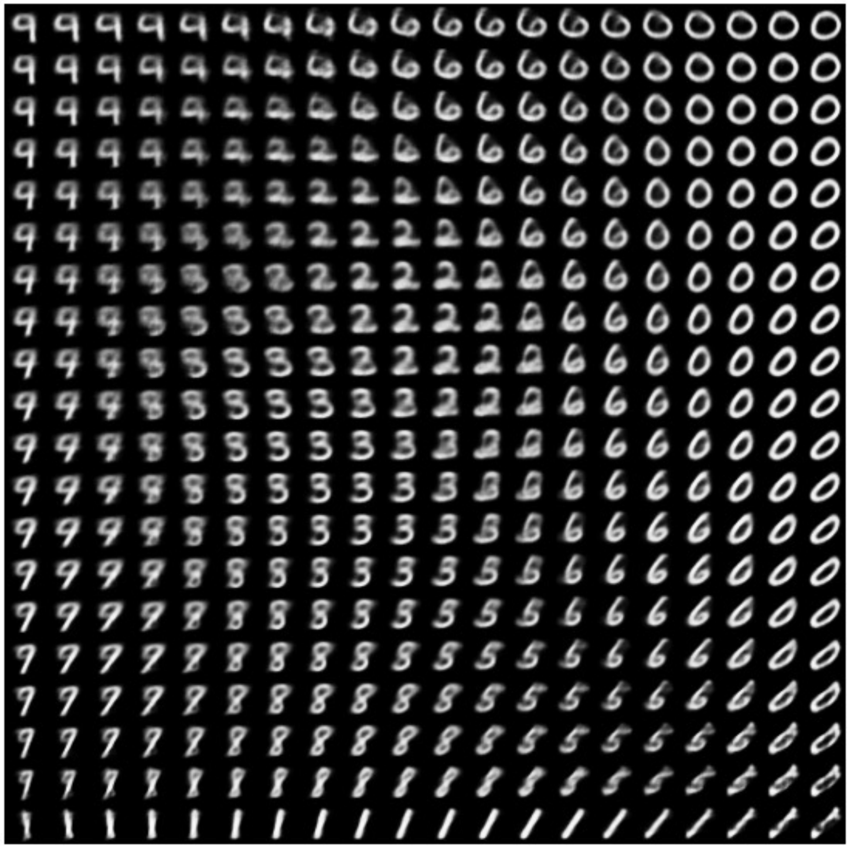
Step 1: Sample from standard Gaussian $z \sim \mathcal{N}(0, I)$

Why it makes sense? $q_\phi(z|x)$ is close to $\mathcal{N}(0, I)$ due to the KL regularization

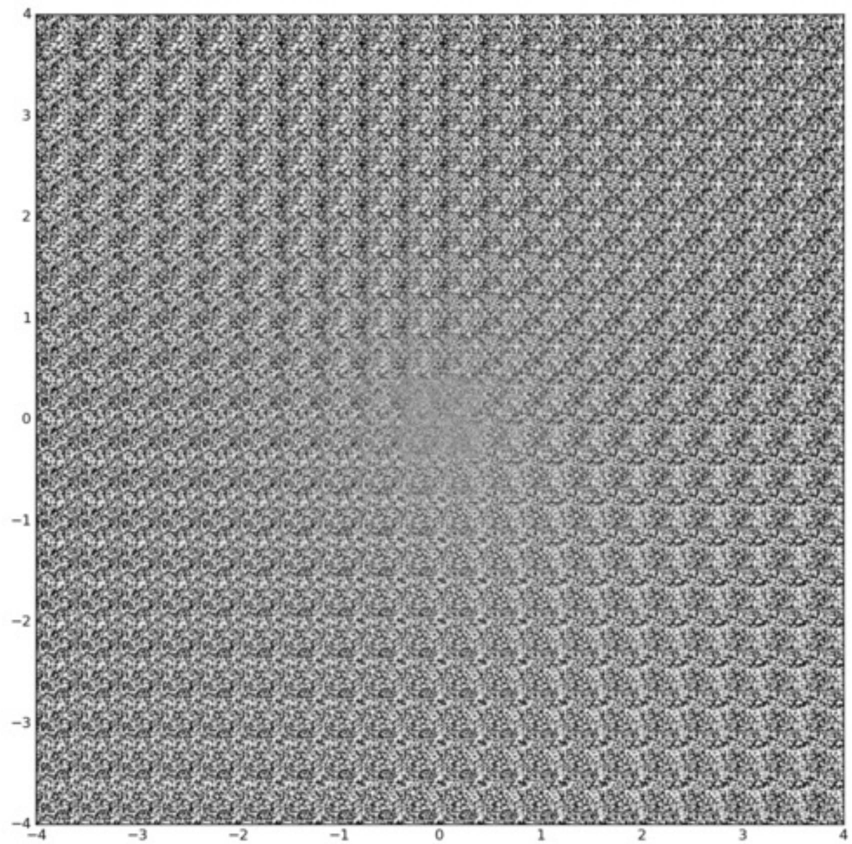
Step 2: Sample the output with $p_\theta(x|z)$

In practice, set the standard deviation of this decoder distribution to zero

2D Latent Space of MNIST

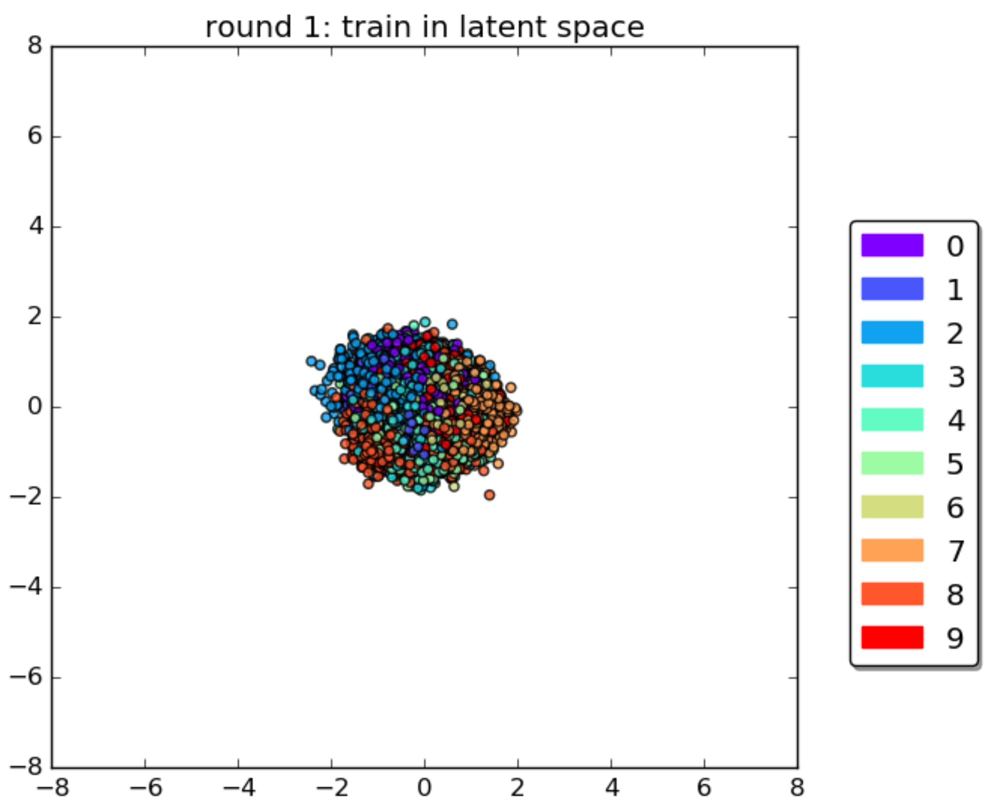


2D Latent Space of MNIST



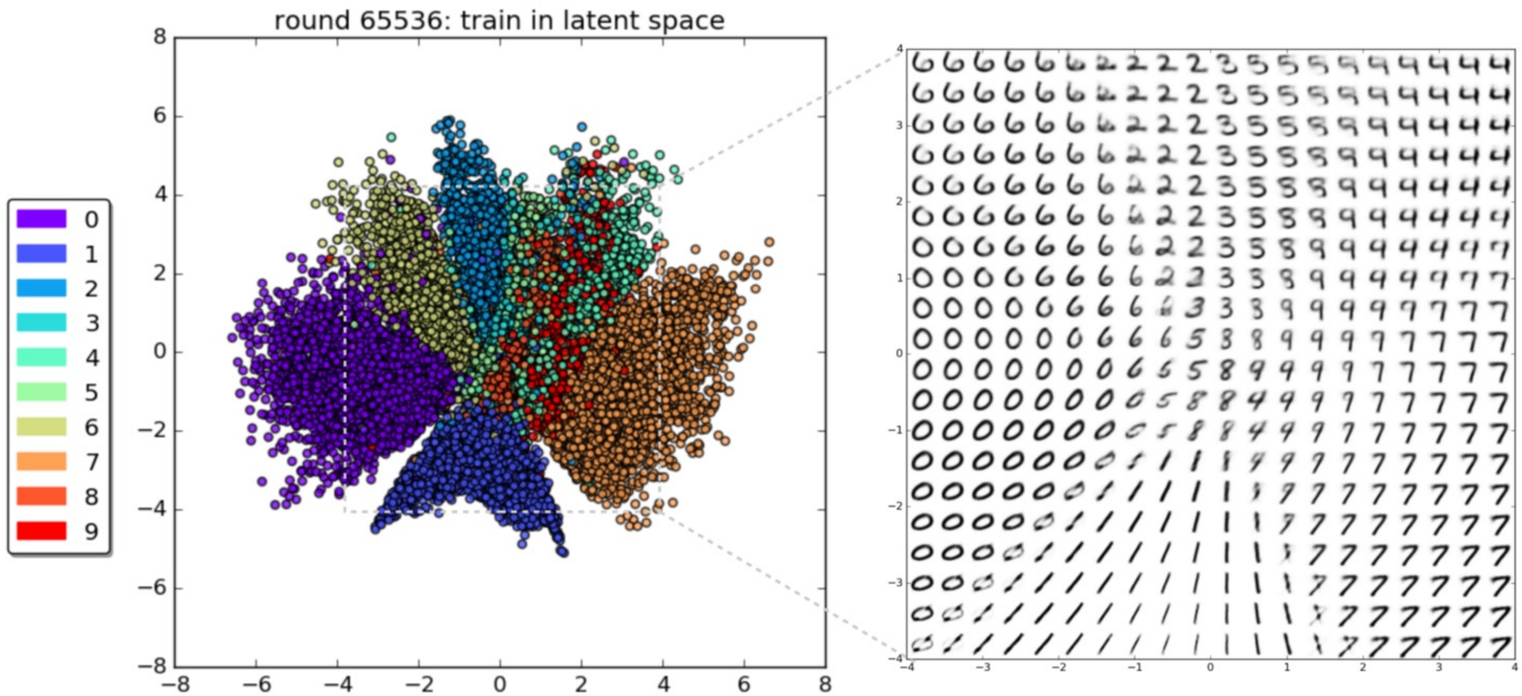
https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

2D Latent Space of MNIST



https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

2D Latent Space of MNIST



https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

VAE as Expectation-Maximization

Instead of gradient descent on theta and phi

Can use EM in simple cases

E step (Expectation): $$q^{(t)} = p_{\theta^{(t)}}(z|x)$$

M step (Maximization): $$\theta^{(t+1)} = \arg\max_{\theta} \mathop{\mathbb{E}}_{\substack{x \sim p_{\mathcal{D}} \\ z \sim p_{\theta^{(t)}}(z|x)}} \left[\log p_{\theta^{(t)}}(x, z) \right]$$

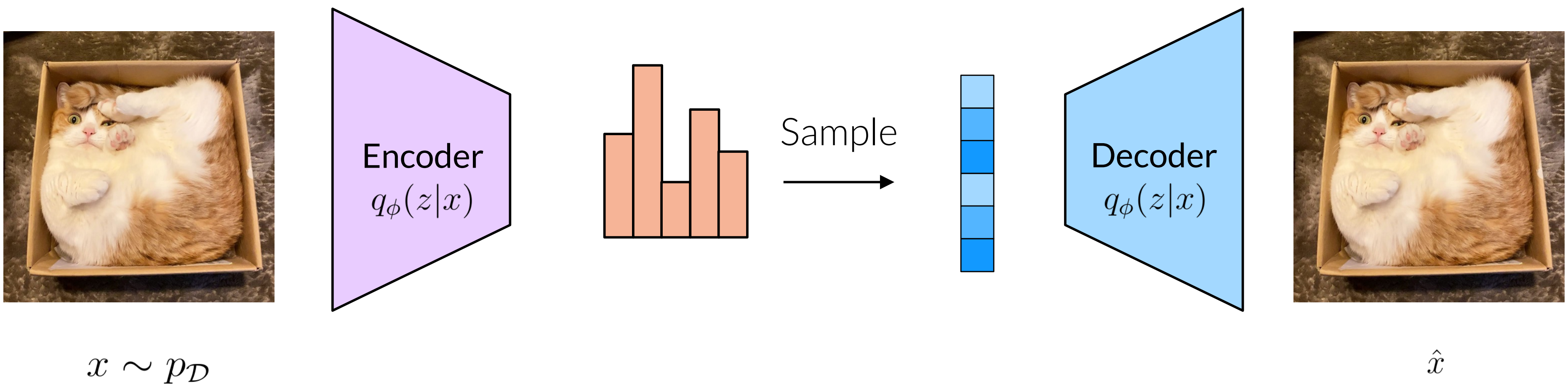
Useful if $q^{(t)}$ has analytical forms:
mixture of Gaussians, K-means, etc.

Failure of VAE

Blurry reconstructed images

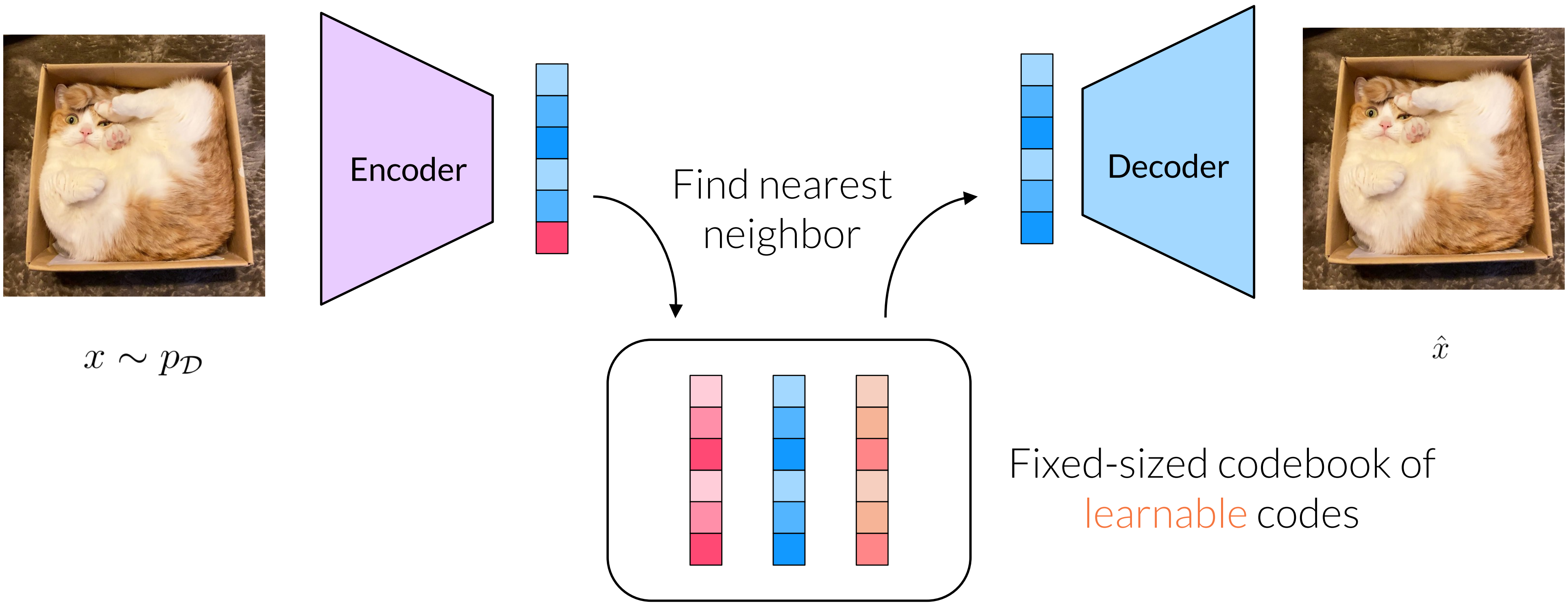
One significant limitation of vanilla VAE:
- Gaussian encoder and decoder

Proposal: Discretized Latent Space



How to discretize the latent space?

Vector Quantization



Training VQ-VAE

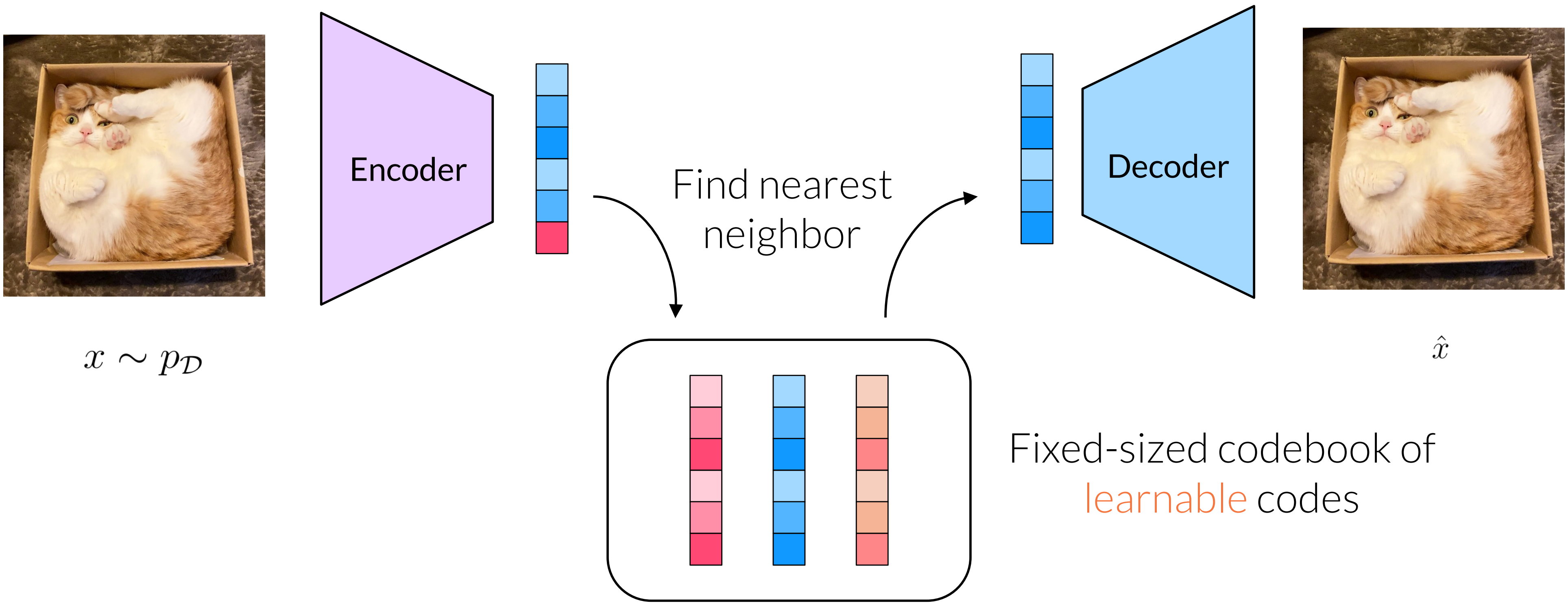
$$\mathcal{L} = \underbrace{\|x - \hat{x}\|^2}_{\text{Reconstruction Loss}} + \beta \underbrace{\|z_e(x) - \text{sg}[z_q(x)]\|^2}_{\text{Commitment Loss}} + \underbrace{\|\text{sg}[z_e(x)] - z_q(x)\|^2}_{\text{Codebook Loss}},$$

sg = stop-gradient

- Commitment loss: encoded feature move towards quantized feature
- Codebook loss: quantized feature towards encoded feature

Exercise: prove that the KL term in ELBO is a constant (with a uniform prior)

Repametrization Trick for VQ-VAE?

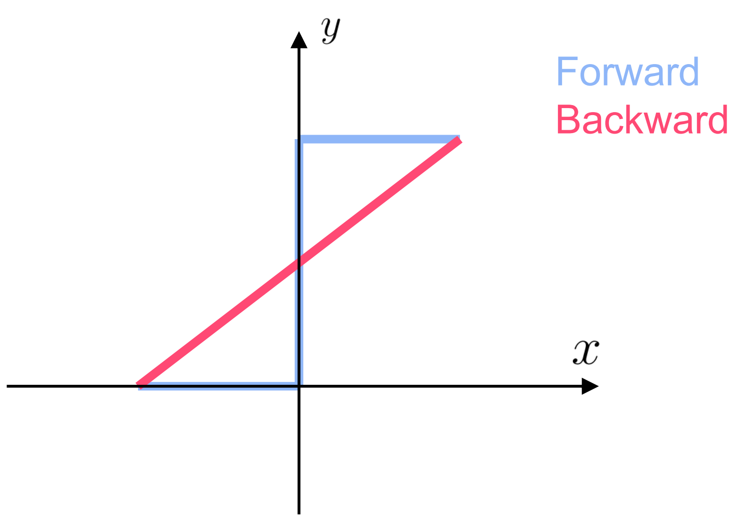


Non-differentiable operation!

Straight-through Estimator

Forward: thresholding function

Backward: pretend that it is an identity function (or some other proxy functions)



Straight-through Estimator (STE)

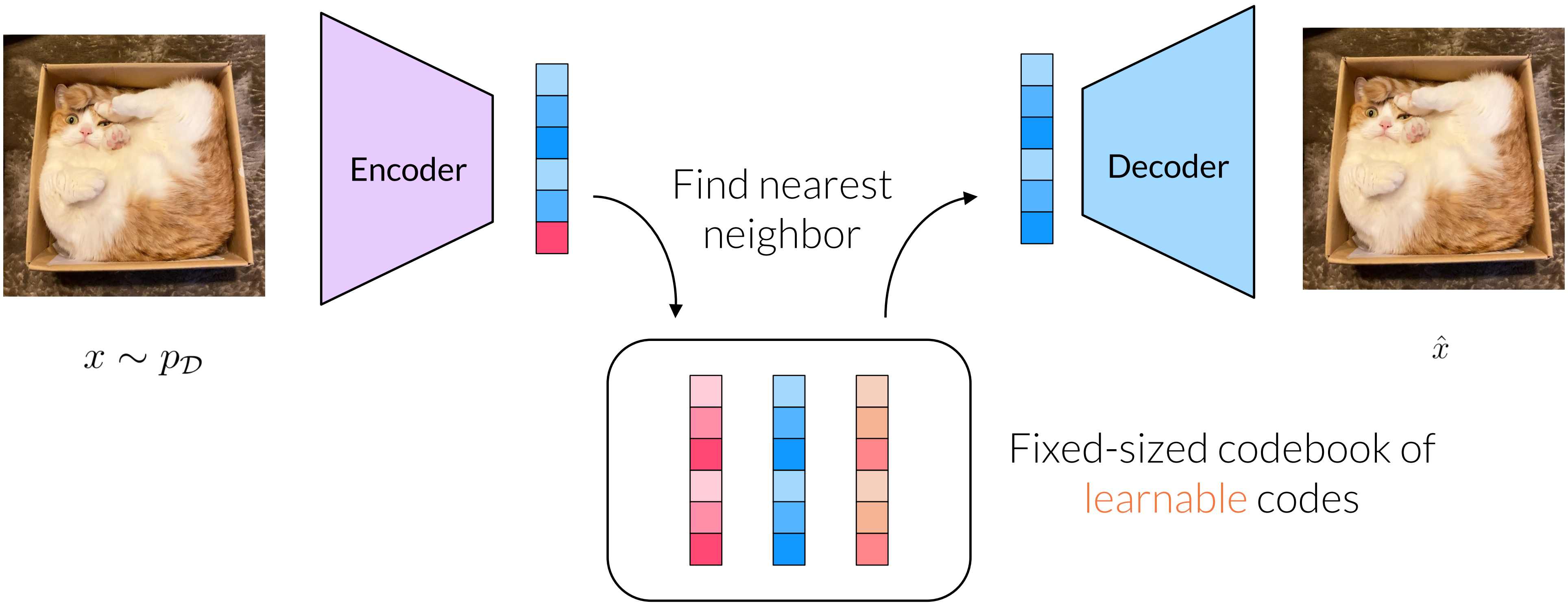
Sampling from categorical distribution = "thresholding" on softmax values

Implementation to use softmax gradients for argmax:

$$\text{stop-grad}\Big(\underbrace{\text{argmax}_k(x_{1:K})}_{\text{Output of forward}} - \text{softmax}_{k,\tau}(x_{1:K})\Big) + \underbrace{\text{softmax}_{k,\tau}(x_{1:K})}_{\text{Gradient for backward}}$$

Can we do better?

Repametrization Trick for VQ-VAE?



How to construct the “epsilon”?

Gumbel-Max Trick

Suppose that we have a categorical distribution $p_\theta(i)$ to pick indices

Define

$$i^* = \arg\max_i \Big(\log p_\theta(i) \underbrace{- \log \big(- \log \epsilon_i \big)}_{\text{Gumbel distribution}} \Big), \quad \epsilon_i \sim \mathrm{Uniform}(0,1)$$

Exercise: show that

$$i^* \sim p_\theta(i)$$

Gumbel-Softmax

Argmax is not differentiable either

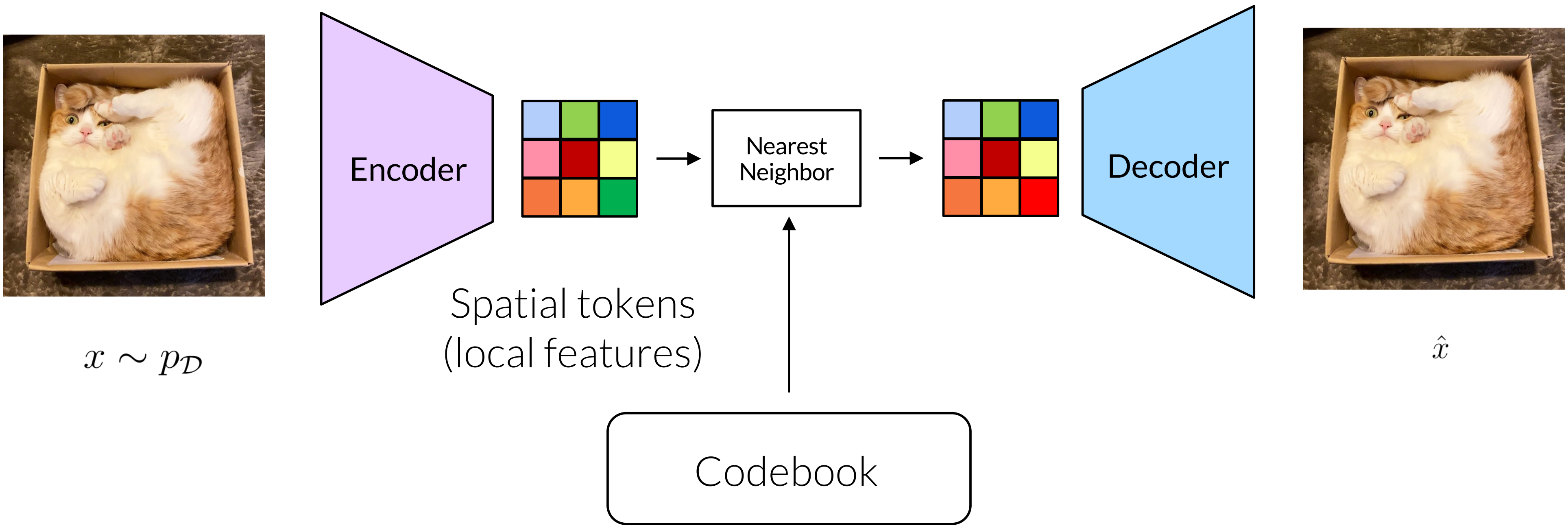
⇒ Replace argmax with softmax

$$i^* = \operatorname*{softmax}_{i,\tau}\Big(\log p_\theta(i) - \log\big(-\log \epsilon_i\big)\Big), \quad \epsilon_i \sim \mathrm{Uniform}(0,1)$$

where

$$\mathrm{softmax}_{i,\tau}(x_i) = \frac{\exp(x_i/\tau)}{\sum_i \exp(x_i/\tau)}$$

VQ-VAE as Tokenizers



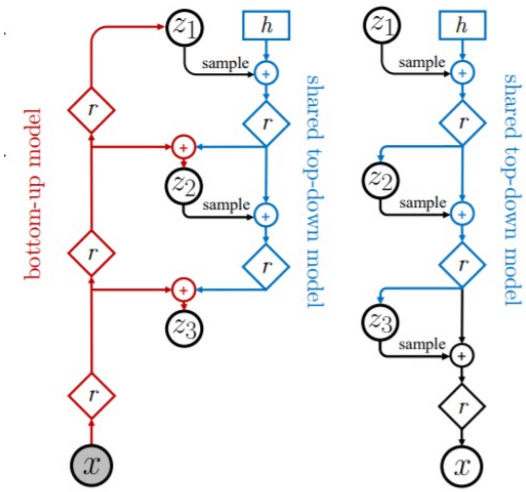
⇒ **Autoregressive modeling on tokens** (*e.g.*, VQ-VAE-2 and some recent work)

Alternative Solution: Hierarchical VAEs

VAEs on latents

Harder (than vanilla VAEs) to train

Will see its connections to other models



(a) Bidirectional Encoder (b) Generative Model

Figure 2: The neural networks implementing an encoder $q(\boldsymbol{z}|\boldsymbol{x})$ and generative model $p(\boldsymbol{x}, \boldsymbol{z})$ for a 3-group hierarchical VAE. ◇r denotes residual neural networks, ⊕ denotes feature combination (e.g., concatenation), and $\boxed{h}$ is a trainable parameter.

NVAE: A Deep Hierarchical Variational Autoencoder. Vahdat et al. NeurIPS 2020.