Image: Synthetic faces sampled from NVAE model by Vahdat and Kautz

COMP547

DEEP UNSUPERVISED LEARNING

Lecture #7 – Variational Autoencoders

KOÇ UNIVERSITY

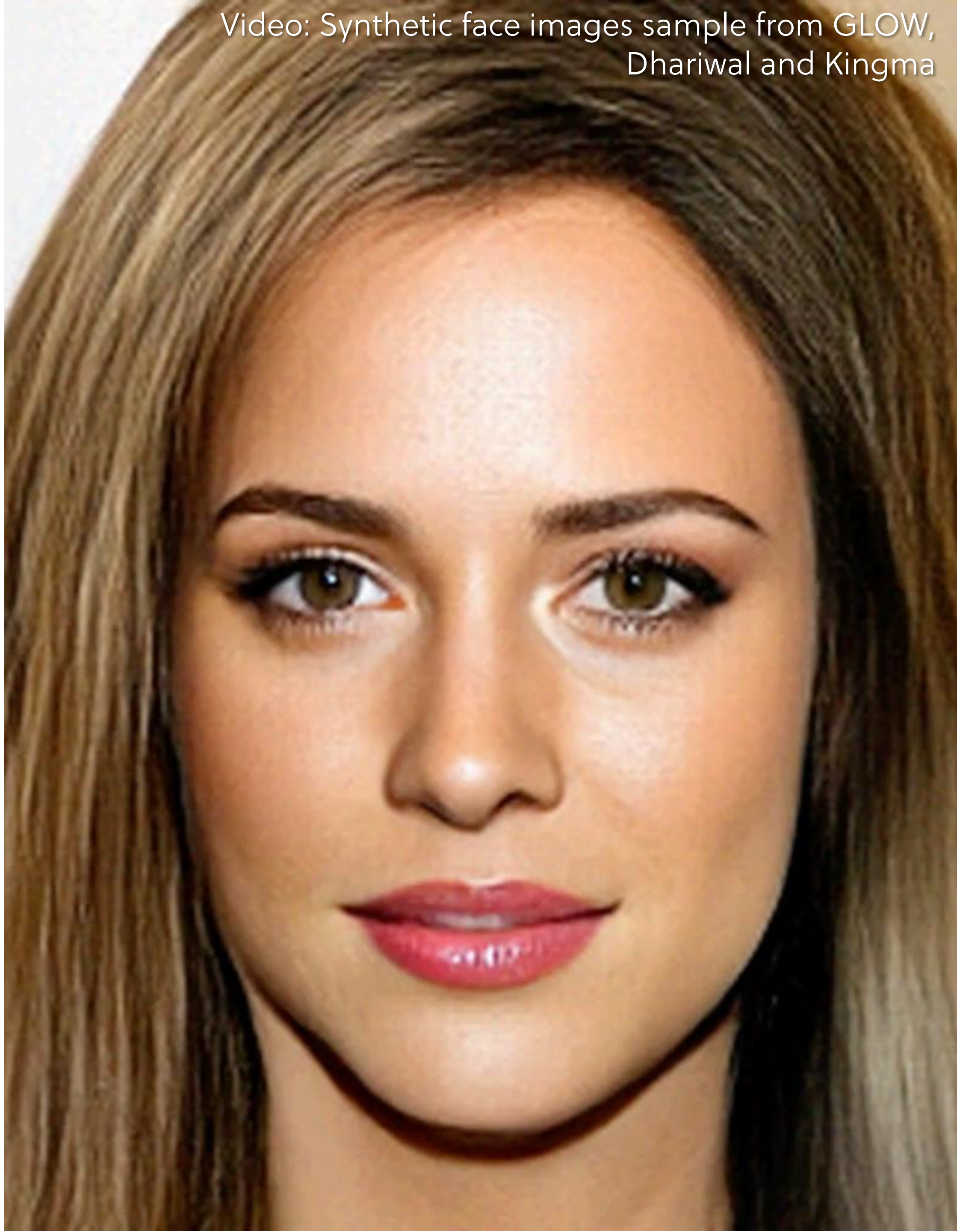
Aykut Erdem // Koç University // Spring 2022

Good news, everyone!

- Assignment 1 is due March 16, 23:59!
- Project proposals are due in one month (April 16)

Previously on COMP547

- Foundations of Flows (1-D)
- 2-D Flows
- N-D Flows
- Dequantization



Lecture overview

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
- Variations
- Related ideas

Disclaimer: Much of the material and slides for this lecture were borrowed from

—Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas’ Berkeley CS294-158 class

—David McAllester's TTIC 31230 class

—Aaron van den Oord's talk on "Neural Discrete Representation Learning"

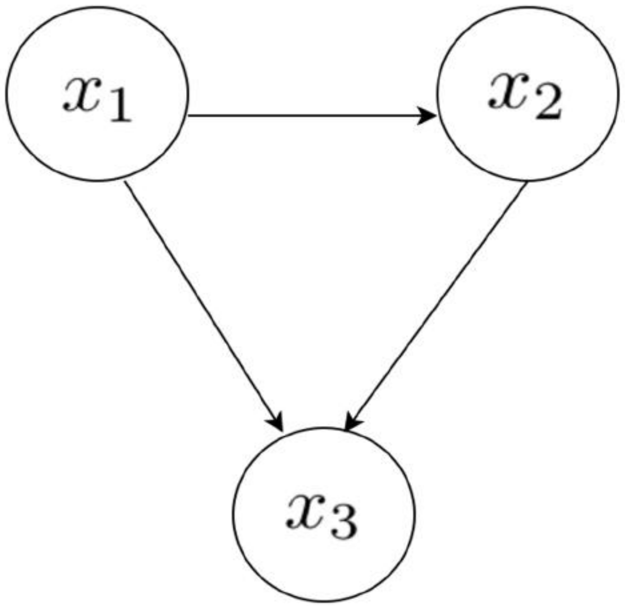
—Jimmy Ba’s UToronto CSC413/2516 class

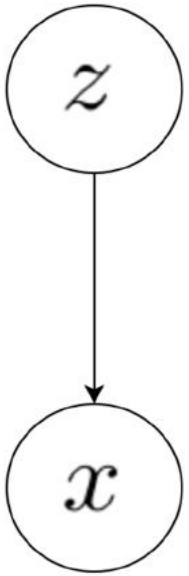
Lecture overview

- **Motivation**
- Training Latent Variable Models (including VAE and IWAE)
- Variations
- Related ideas

Latent Variable Models

- Autoregressive models + Flows
  - All random variables are observed
- Latent Variable Models (LVMs):
  - Some random variables are hidden - we do not get to observe





Why Latent Variable Models?

- Simpler, lower-dimensional representations of data often possible
  - Latent variable models hold the promise of automatically identifying those hidden representations

Obj1 @ (x,y) = Corgi, red & white

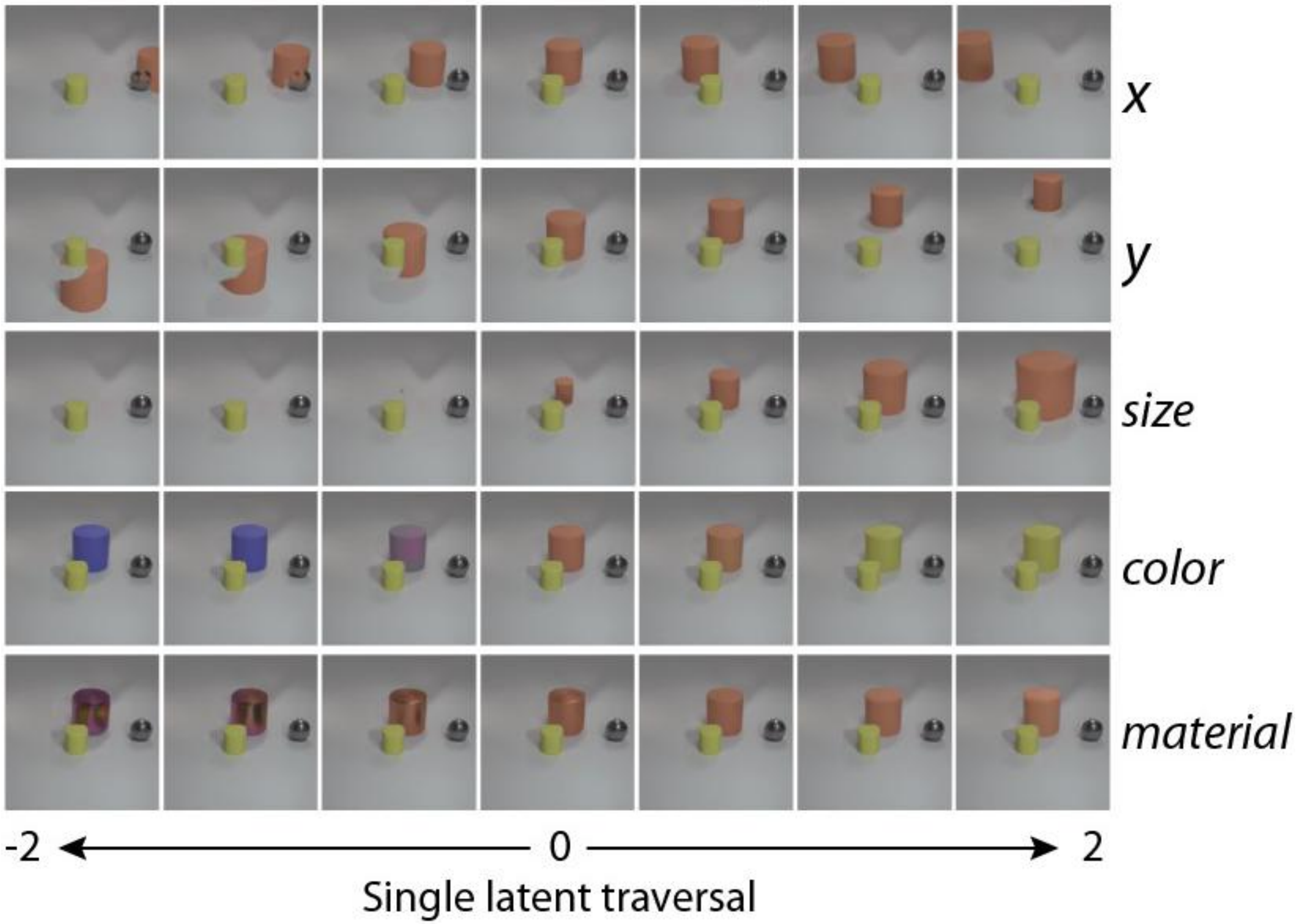
Obj2 @ (x,y) = Corgi, red & white, floppy left ear

Background = Wood bench in a park

Obj3 @ (x,y) = Corgi, Tri-color

Why Latent Variable Models?

- Simpler, lower-dimensional representations of data often possible
  - Latent variable models hold the promise of automatically identifying those hidden representations



IODINE [Greff et al, 2019]

Why Latent Variable Models?

- AR models are slow to sample because all pixels (observation dims) are assumed to be dependent on each other
- We can make part of observation space independent **conditioned on some latent variables**
  - Latent variable models can have faster sampling by exploiting statistical patterns

Latent Variable Models

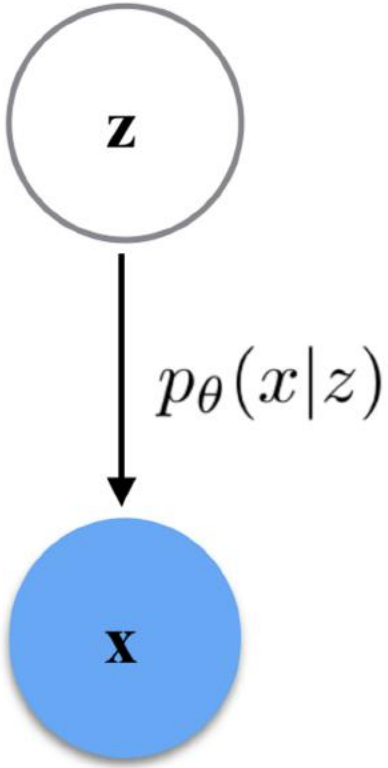
- Sometimes, it's possible to design a latent variable model with an understanding of the causal process that generates data
- In general, we don't know what are the latent variables and how they interact with observations
  - Most popular models make little assumption about what are the latent variables
  - Best way to specify latent variables is still an active area of research

Modifying the latent code in the direction of a smile vector, producing a range of versions of the original, from smiling to sadness (White, 2016)

Example Latent Variable Model

$$z = (z_1, z_2, \cdots, z_K) \sim p(z;\beta) = \prod_{k=1}^{K} \beta_k^{z_k}(1-\beta_k)^{1-z_k}$$

$$x = (x_1, x_2, \cdots, x_L) \sim p_\theta(x|z) \Leftrightarrow \text{Bernoulli}(x_i; \text{DNN}(z))$$



Latent Variable Model

- Sample

$$z \sim p_Z(z)$$

$$x \sim p_\theta(x \mid z)$$

- Evaluate likelihood

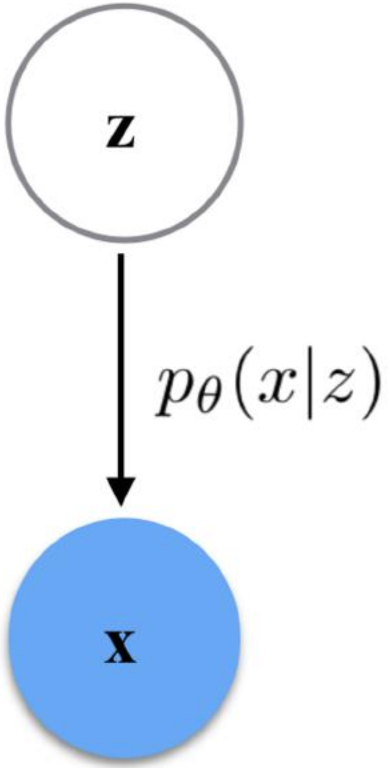
$$p_\theta(x) = \sum_z p_Z(z) p_\theta(x \mid z)$$

- Train

$$\max_\theta \sum_i \log p_\theta(x^{(i)}) = \sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} \mid z)$$

- Representation

$$x \to z$$



Lecture overview

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
  - Objective
    - Exact
    - Prior Sampling
    - Importance Sampling
    - Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)
  - Optimization
    - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
    - Optimizing the VLB/ELBO
- Variations
- Related ideas

Training Latent Variable Models

- Objective:

$$\max_\theta \sum_i \log p_\theta(x^{(i)}) = \sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} \mid z)$$

- **Scenario 1:** z can only take on a small number of values → exact objective tractable

- **Scenario 2:** z can take on an impractical number of values to enumerate → approximate

How about optimizing $p_z(z)$? = "learning the prior" and sometimes done [more later]

Lecture overview

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
  - Objective
    - Exact
    - Prior Sampling
    - Importance Sampling
    - Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)
  - Optimization
    - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
    - Optimizing the VLB/ELBO
- Variations
- Related ideas

Exact Likelihood Objective

Example: mixture of 3 Gaussians, with uniform prior over components

$$p_\theta(x) = \sum_z p_Z(z) p_\theta(x \mid z)$$

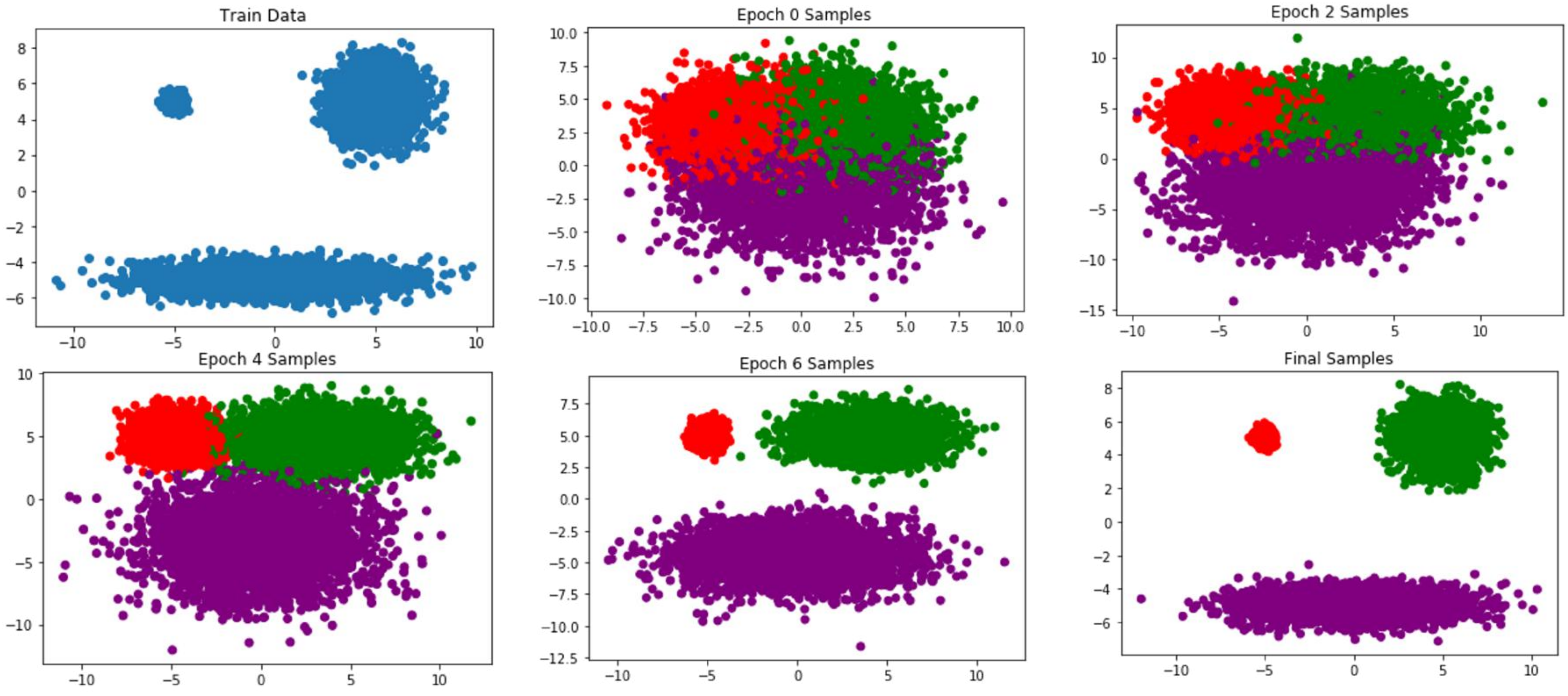
$$p_Z(z = A) = p_Z(z = B) = p_Z(z = C) = \frac{1}{3}$$

$$p_\theta(x \mid z = k) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k) \right)$$

Training objective: $\max_\theta \sum_i \log p_\theta(x^{(i)})$

$$\begin{aligned} = \max_{\mu, \Sigma} \sum_i \log[\quad & \frac{1}{3} \quad \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_A|^{\frac{1}{2}}} \exp\left(-\tfrac{1}{2}(x^{(i)} - \mu_A)^\top \Sigma_A^{-1} (x^{(i)} - \mu_A)\right) \\ & + \frac{1}{3} \quad \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_B|^{\frac{1}{2}}} \exp\left(-\tfrac{1}{2}(x^{(i)} - \mu_B)^\top \Sigma_B^{-1} (x^{(i)} - \mu_B)\right) \\ & + \frac{1}{3} \quad \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_C|^{\frac{1}{2}}} \exp\left(-\tfrac{1}{2}(x^{(i)} - \mu_C)^\top \Sigma_C^{-1} (x^{(i)} - \mu_C)\right)] \end{aligned}$$

2-D mixture of Gaussians



Prior Sampling

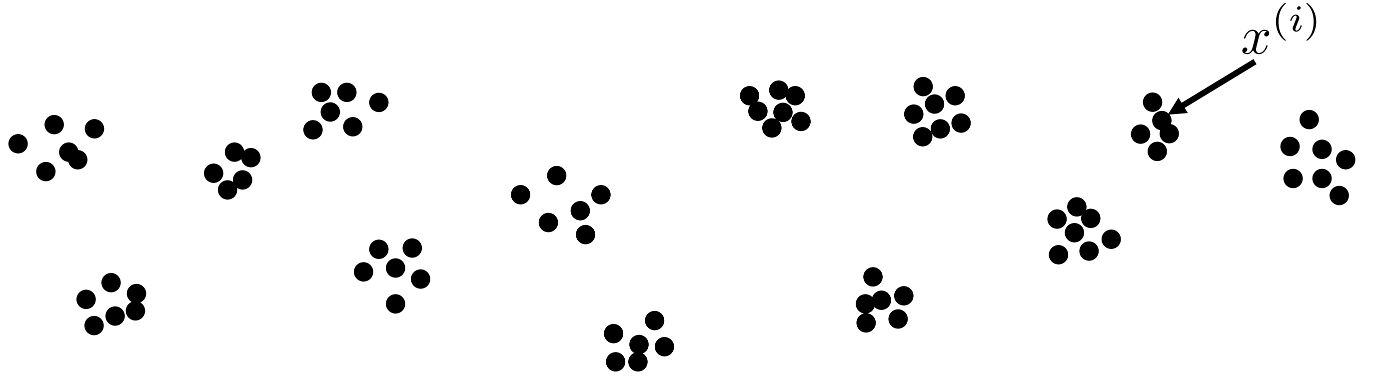
Main idea: if z can take on many values → sample z

$$\sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} \mid z) \approx \sum_i \log \frac{1}{K} \sum_{k=1}^{K} p_\theta(x^{(i)} \mid z_k^{(i)}) \qquad z_k^{(i)} \sim p_Z(z)$$

→ run Stochastic Gradient Descent (SGD) on the approximate objective

Prior Sampling – Example + Challenge

Consider data in N clusters:



Sampling z uniformly results in only in only 1/N terms being useful.

Recall: z might correspond to many high level properties, e.g., 100 high level properties, probably of correct z for given x: 0.5^{100}

Issue: When going to higher dimensional data, it becomes near impossible to be lucky enough that a sampled z is a good match for a data point $x^{(i)}$

Importance Sampling – Motivation

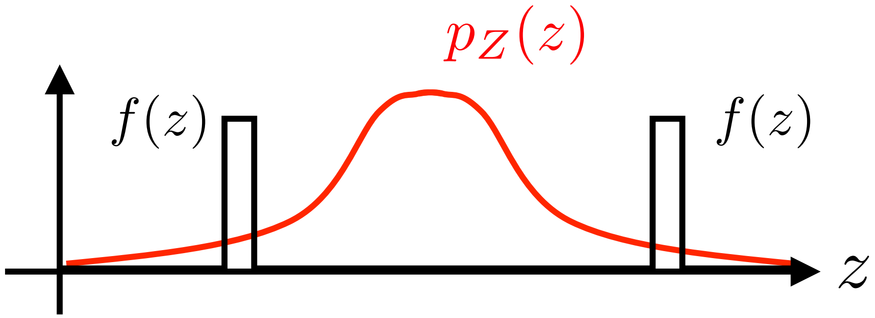
Problem setting:

Want to compute $\mathbb{E}_{z \sim p_Z(z)} [f(z)]$

But: (1) hard to sample from $p_Z(z)$

and/or (2) samples from $p_Z(z)$ are not very informative

Example of (2):



Note: our Latent Variable Model objective is also example of (2)

Importance Sampling – Algorithm

Formulation:

$$
\begin{aligned}
\mathbb{E}_{z\sim p_Z(z)}\left[f(z)\right] &= \sum_z p_Z(z) f(z) \\
&= \sum_z \frac{q(z)}{q(z)} p_Z(z) f(z) \\
&= \mathbb{E}_{z\sim q(z)}\left[\frac{p_Z(z)}{q(z)} f(z)\right] \\
&\approx \frac{1}{K}\sum_{k=1}^{K} \frac{p_Z(z^{(k)})}{q(z^{(k)})} f(z^{(k)}) \quad \text{with} \quad z^{(k)} \sim q(z)
\end{aligned}
$$

→ Can sample from q to compute expectation w.r.t. p

Importance Sampling for Latent Variable Model

Training Objective:

$$\sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} \mid z) \approx \sum_i \log \frac{1}{K} \sum_{k=1}^{K} \frac{p_Z(z_k^{(i)})}{q(z_k^{(i)})} p_\theta(x^{(i)} \mid z_k^{(i)}) \quad \text{with} \quad z_k^{(i)} \sim q(z_k^{(i)}))$$

Good proposal distribution q(z)?

We want samples compatible with $x^{(i)}$

→ How about $q(z) = p_\theta(z \mid x^{(i)}) = \dfrac{p_\theta(x^{(i)} \mid z) p_Z(z)}{p_\theta(x^{(i)})}$

Issue: not clear how to sample from this distribution...

Importance Sampling Proposal Distribution

General Principle of Variational Approach:

We can't directly use p we want

So, instead, we propose a parameterized distribution q we know we can work with easily (in this case, sample from easily), and try to find a parameter setting that makes it as good as possible.

E.g. find $q(z) = \mathcal{N}(z; \mu, \sigma^2)$ as close as possible to

$$p_\theta(z \mid x^{(i)}) = \frac{p_\theta(x^{(i)} \mid z) p_Z(z)}{p_\theta(x^{(i)})}$$

Importance Sampling Proposal Distribution

Variational Approach to Finding q:

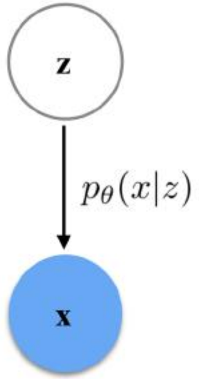
We can't directly use p we want

So, instead, we propose a parameterized distribution q we know we can work with easily (in this case, sample from easily), and try to find a parameter setting that makes it as good as possible.

E.g. find $q(z) = \mathcal{N}(z; \mu, \sigma^2)$ as close as possible to

$$p_\theta(z \mid x^{(i)}) = \frac{p_\theta(x^{(i)} \mid z) p_Z(z)}{p_\theta(x^{(i)})}$$

Importance Sampling Proposal Distribution



$$\min_{q(z)} \mathrm{KL}(q(z) \| p_\theta(z \mid x^{(i)})$$

$$= \min_{q(z)} \mathbb{E}_{z \sim q(z)} \log \left(\frac{q(z)}{p_\theta(z|x^{(i)})} \right)$$

$$= \min_{q(z)} \mathbb{E}_{z \sim q(z)} \log \left(\frac{q(z)}{p_\theta(x^{(i)}|z) p_Z(z) / p_\theta(x^{(i)})} \right)$$

$$= \min_{q(z)} \mathbb{E}_{z \sim q(z)} \left[\log q(z) - \log p_Z(z) - \log p_\theta(x^{(i)} \mid z) \right] + \log p_\theta(x^{(i)})$$

$$= \min_{q(z)} \mathbb{E}_{z \sim q(z)} \left[\log q(z) - \log p_Z(z) - \log p_\theta(x^{(i)} \mid z) \right] + \text{constant independent of z}$$

→ optimize to find q

Note: all needed quantities in the objective readily computable

Amortized Inference

General Idea of Amortization: if same inference problem needs to be solved many times, can we parameterize a neural network to solve it?

Our case: for all $x^{(i)}$ we want to solve:

$$\min_{q(z)} \mathrm{KL}(q(z) \| p_\theta(z \mid x^{(i)}))$$

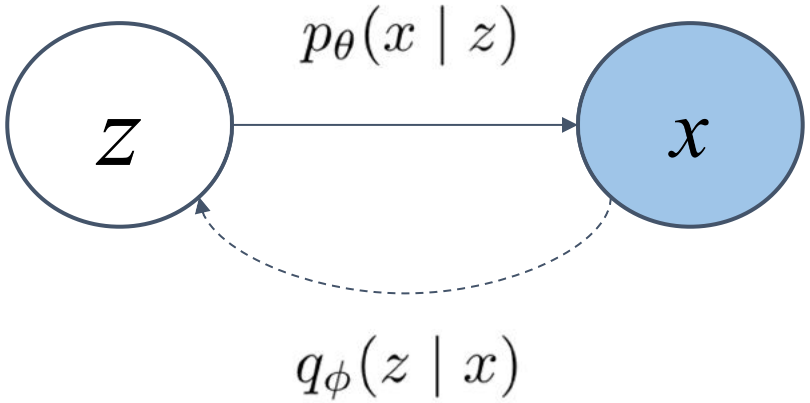
Amortized formulation:

$$\min_{\phi} \sum_i \mathrm{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)}))$$

Trade-off: + : faster, regularization; – : not as precise

Amortized Inference

Amortized formulation: $$\min_{\phi} \sum_i \mathrm{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)}))$$



E.g: $$q_\phi(z \mid x) = \mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x))$$

Equivalently: $z = \mu_\phi(x) + \varepsilon \sigma_\phi(x)$ with $\varepsilon \sim \mathcal{N}(0, I)$

Importance Weighted AutoEncoder (IWAE)

Objective: $$\sum_i \log \frac{1}{K} \sum_{k=1}^{K} \frac{p_Z(z_k^{(i)})}{q(z_k^{(i)})} p_\theta(x^{(i)} \mid z_k^{(i)}) \quad \text{with} \quad z_k^{(i)} \sim q(z_k^{(i)}))$$

And: $$\min_\phi \sum_i \mathrm{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)}))$$

→ maximize term1 – term2

Importance Weighted AutoEncoder (IWAE)

Theorem 1. *For all k, the lower bounds satisfy*

$$\log p(\mathbf{x}) \geq \mathcal{L}_{k+1} \geq \mathcal{L}_k.$$

Moreover, if $p(\mathbf{h}, \mathbf{x})/q(\mathbf{h}|\mathbf{x})$ is bounded, then $\mathcal{L}_k$ approaches $\log p(\mathbf{x})$ as k goes to infinity.

[Burda et al., 2015]

Lecture overview

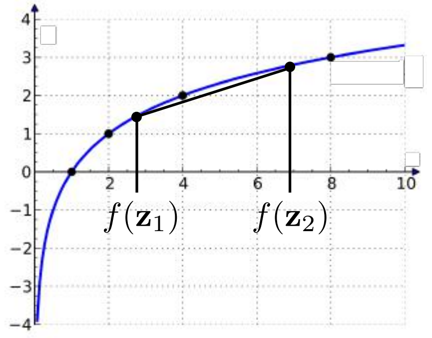
- Motivation
- **Training Latent Variable Models (including VAE and IWAE)**
  - **Objective**
    - Exact
    - Prior Sampling
    - Importance Sampling
    - **Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)**
  - Optimization
    - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
    - Optimizing the VLB/ELBO
- Variations
- Related ideas

VLB: Derivation 1 (Jensen)

Jensen Inequality (for concave functions)

- log is a concave function)

$$\log\left(\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}[f(\mathbf{z})]\right) = \log\left(\sum_{\mathbf{z}} q(\mathbf{z}) f(\mathbf{z})\right) \geq \sum_{\mathbf{z}} q(\mathbf{z}) \log f(\mathbf{z})$$



$$\max_\theta \sum_i \log p_\theta(x^{(i)}) = \max_\theta \sum_i \log\left(\sum_z p_z(z) \cdot p_\theta(x^{(i)}|z)\right)$$

$$\max_\theta \sum_i \log\left(\sum_z \frac{q(z)}{q(z)} p_z(z) \cdot p_\theta(x^{(i)}|z)\right)$$

$$\geq \max_\theta \sum_i \mathbb{E}_{z\sim q(z)} \log \frac{p_z(z)}{q(z)} p_\theta(x^{(i)}|z)$$

$$= \max_{\theta, q} \sum_i \mathbb{E}_{z\sim q(z)} \log p_z(z)$$

$$+ \mathbb{E}_{z\sim q(z)} \log p_\theta(x^{(i)}|z)$$

$$- \mathbb{E}_{z\sim q(z)} q(z)$$

If $q(z) \propto p_z(z) p_\theta(x^{(i)}|z)$

$p_\theta(z|x^{(i)})$,

we have equality

VLB: Variational Lower Bound

ELBO: Evidence Lower Bound

VLB: Derivation 2 (KL)

$$\begin{aligned} D_{\mathrm{KL}}\left[q_x(z) \parallel p(z|x)\right] &= \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z|x)\right] \\ &= \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log \frac{p(z,x)}{p(x)}\right] \\ &= \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z) + \log p(x)\right] \\ &= \underbrace{\mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z)\right]}_{\text{Only this part depends on } z} + \log p(x) \end{aligned}$$

$$\log p(x) = \mathbb{E}_{z\sim q_x(z)}\left[-\log q_x(z) + \log p(z) + \log p(x \mid z)\right] + D_{KL}\left[q_x(z) \| p(z \mid x)\right]$$

Same as with Jensen’s, but now we know the gap = KL

Variational Lower Bound (VLB)

- We now have an objective amenable to stochastic optimization

$$D_{\text{KL}}\left[q_x(z) \,\|\, p(z|x)\right] = \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z)\right] + \log p(x)$$

- Turns out we can get more out of this exercise

$$\begin{aligned}\log p(x) &= -\mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z)\right] + D_{\text{KL}}\left[q_x(z) \,\|\, p(z|x)\right] \\ &= \underbrace{\mathbb{E}_{z\sim q_x(z)}\left[\log p(z) + \log p(x|z) - \log q_x(z)\right]}_{\text{Variational Lower Bound}} + \underbrace{D_{\text{KL}}\left[q_x(z) \,\|\, p(z|x)\right]}_{\geq 0}\end{aligned}$$

  - note: the optimal $q_x(z)$ of VLB is $p(z|x)$, at which point VLB is tight (= $\log p(x)$)

VLB Maximization

- Given a data distribution x ~ p_{data}, we can know train the generative model by maximizing the VLB under data distribution

$$
\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\mathbb{E}_{\mathbf{z}\sim q_x(\mathbf{z})}\left[\log p(\mathbf{z}) + \log p(\mathbf{x}|\mathbf{z}) - \log q_x(\mathbf{z})\right]\right]
$$

$$
\leq \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log p(\mathbf{x})\right]
$$

Lecture overview

- Motivation
- **Training Latent Variable Models (including VAE and IWAE)**
  - Objective
    - Exact
    - Prior Sampling
    - Importance Sampling
    - Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)
  - **Optimization**
    - **Likelihood Ratio Gradients vs. Reparameterization Trick Gradients**
    - **Optimizing the VLB/ELBO**
- Variations
- Related ideas

Likelihood Ratio Gradient

$$\max_\phi \mathbb{E}_{z\sim q_\phi(z)}[f(z)] \qquad z^{(i)} \sim q_\phi(z) \qquad \mathbb{E}_{z\sim q_\phi(z)}[f(z)] \approx \frac{1}{k}\sum_{i=1}^k f(z^{(i)})$$

cannot compute $\nabla_\phi \left(\frac{1}{k}\sum_{i=1}^k f(z^{(i)})\right)$

$$\max_\phi \mathbb{E}_z[q_\phi(z) f(z)]$$

$$\nabla_\phi \left(\max_\phi \mathbb{E}_z[q_\phi(z) f(z)]\right) = \sum_z \nabla_\phi [q_\phi(z) f(z)]$$

We need to turn this into an expectation

$$= \sum_z \frac{q_\phi(z)}{q_\phi(z)} \nabla_\phi [q_\phi(z) f(z)]$$

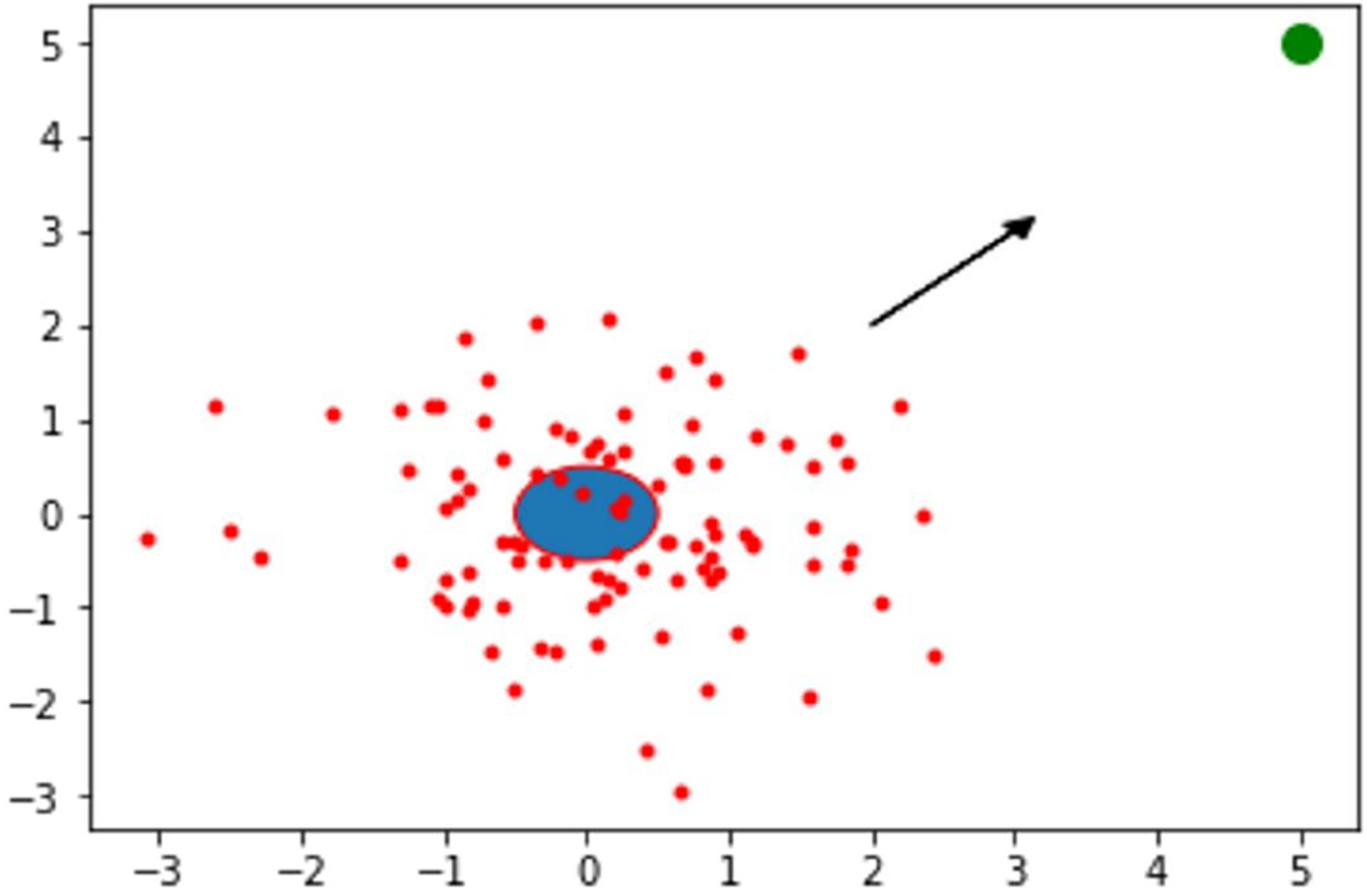
$$= \mathbb{E}_{z\sim q_\phi(z)} \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)} f(z)$$

$$= \mathbb{E}_{z\sim q_\phi(z)}[\nabla_\phi \log q_\phi(z) f(z)] \approx \frac{1}{k}\sum_{i=1}^k \nabla_\phi \log q_\phi(z) f(z)$$

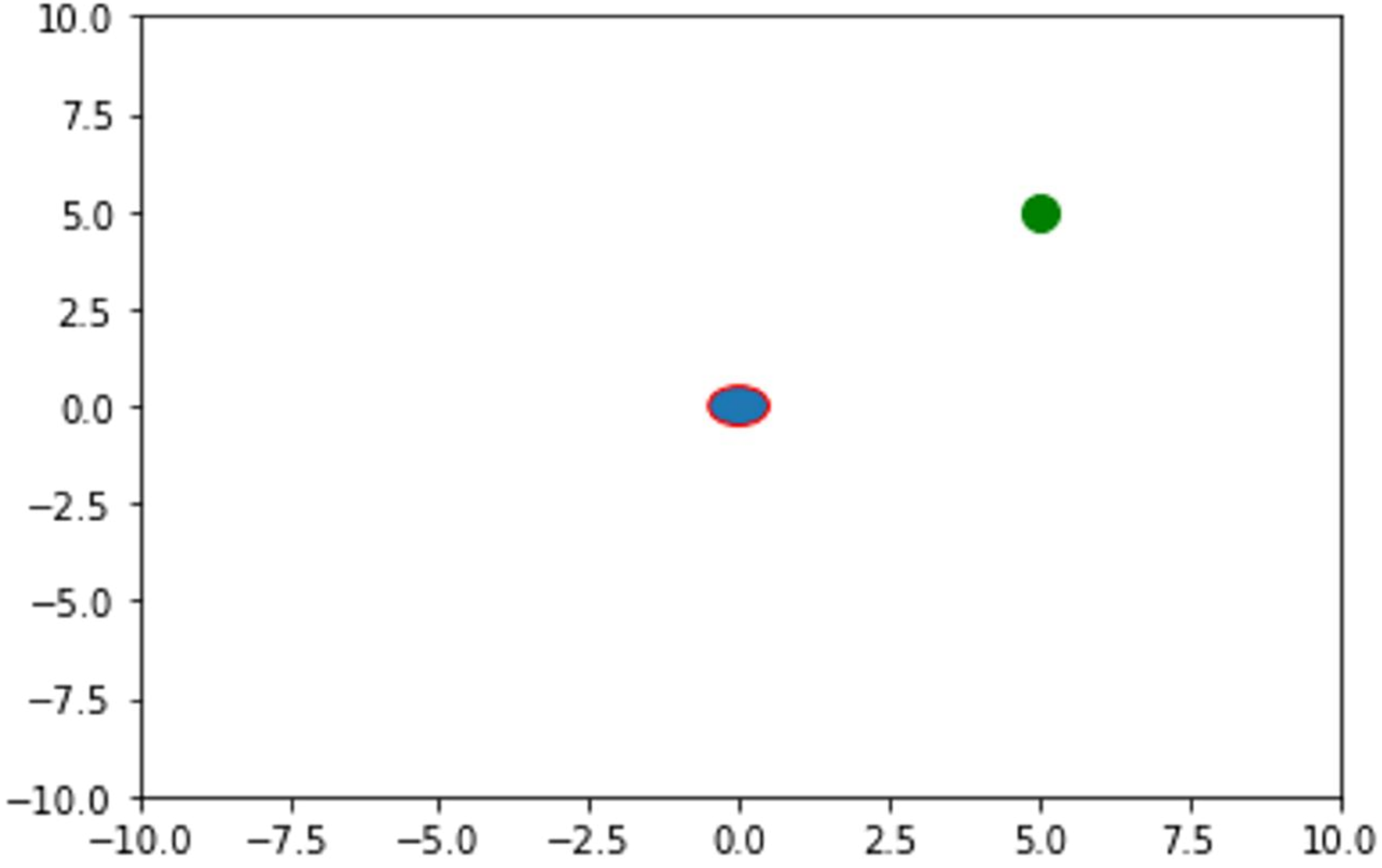
Likelihood Ratio Gradient – Toy Problem

Learn $\mu \in \mathbb{R}^2$ to minimize the objective below to reach the green point

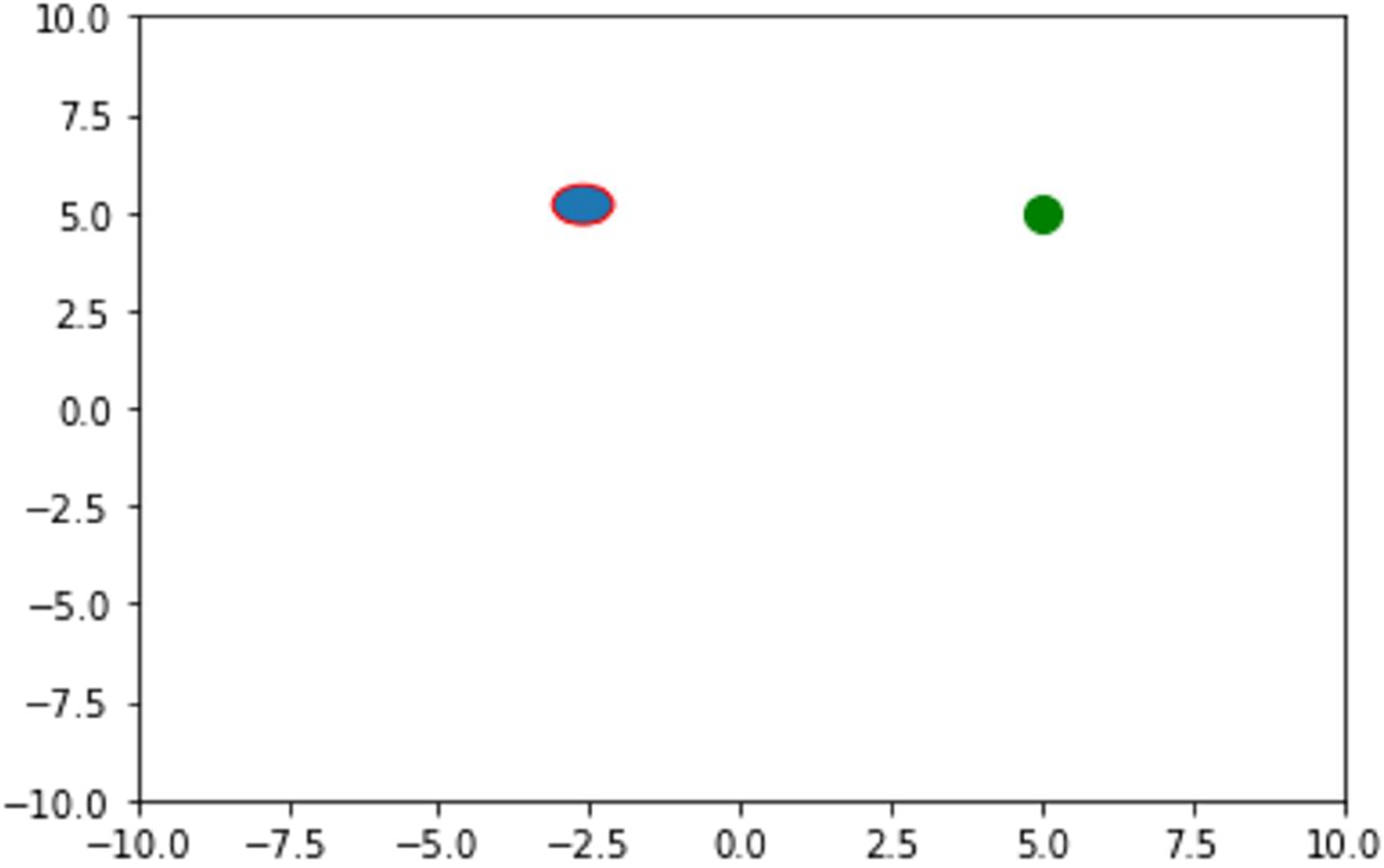
$$\mathcal{L} = \mathbb{E}_{x \sim N(\mu, I)} \left\| x - \begin{bmatrix} 5 \\ 5 \end{bmatrix} \right\|_2^2$$



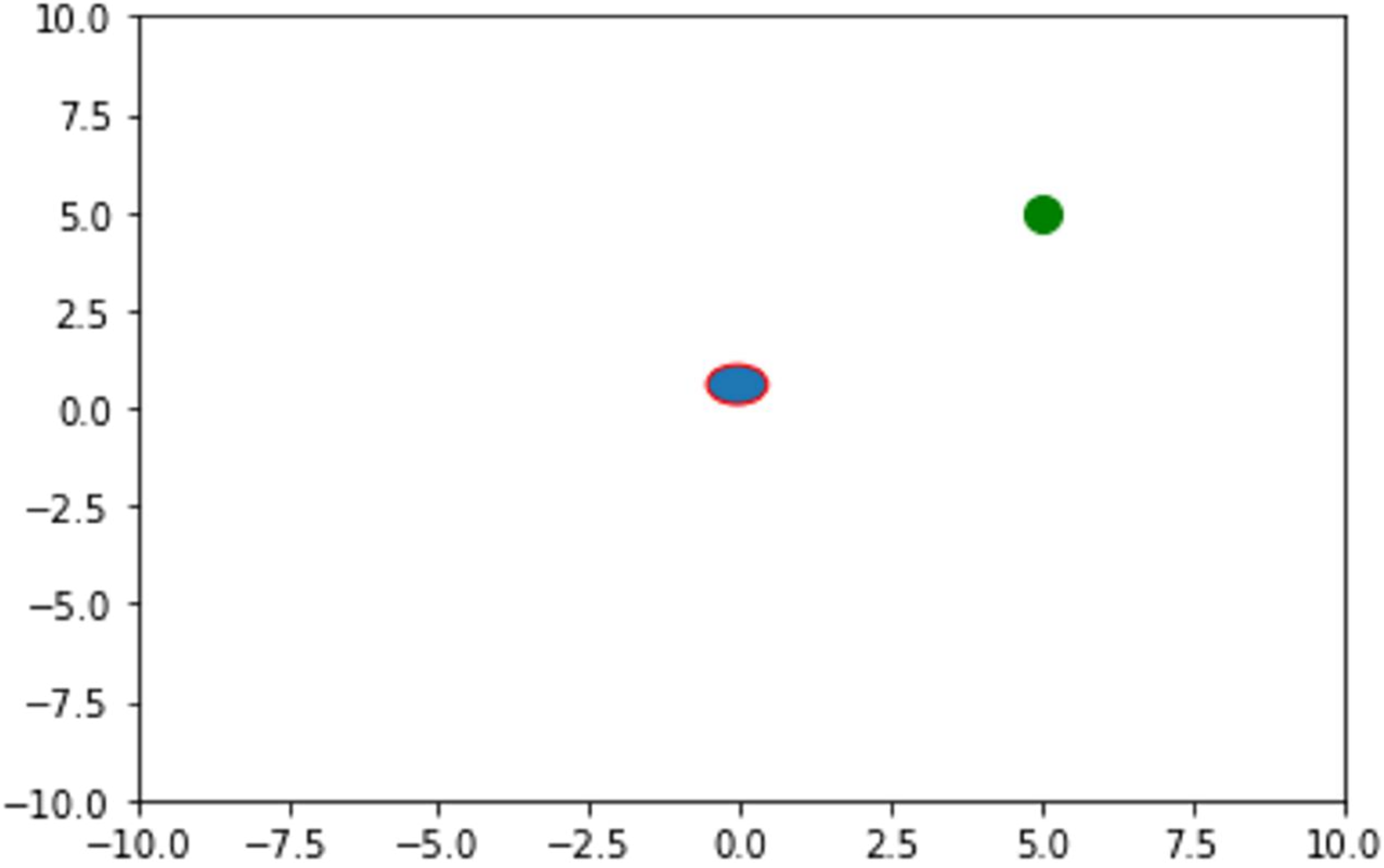
Likelihood Ratio Gradient – Toy Problem



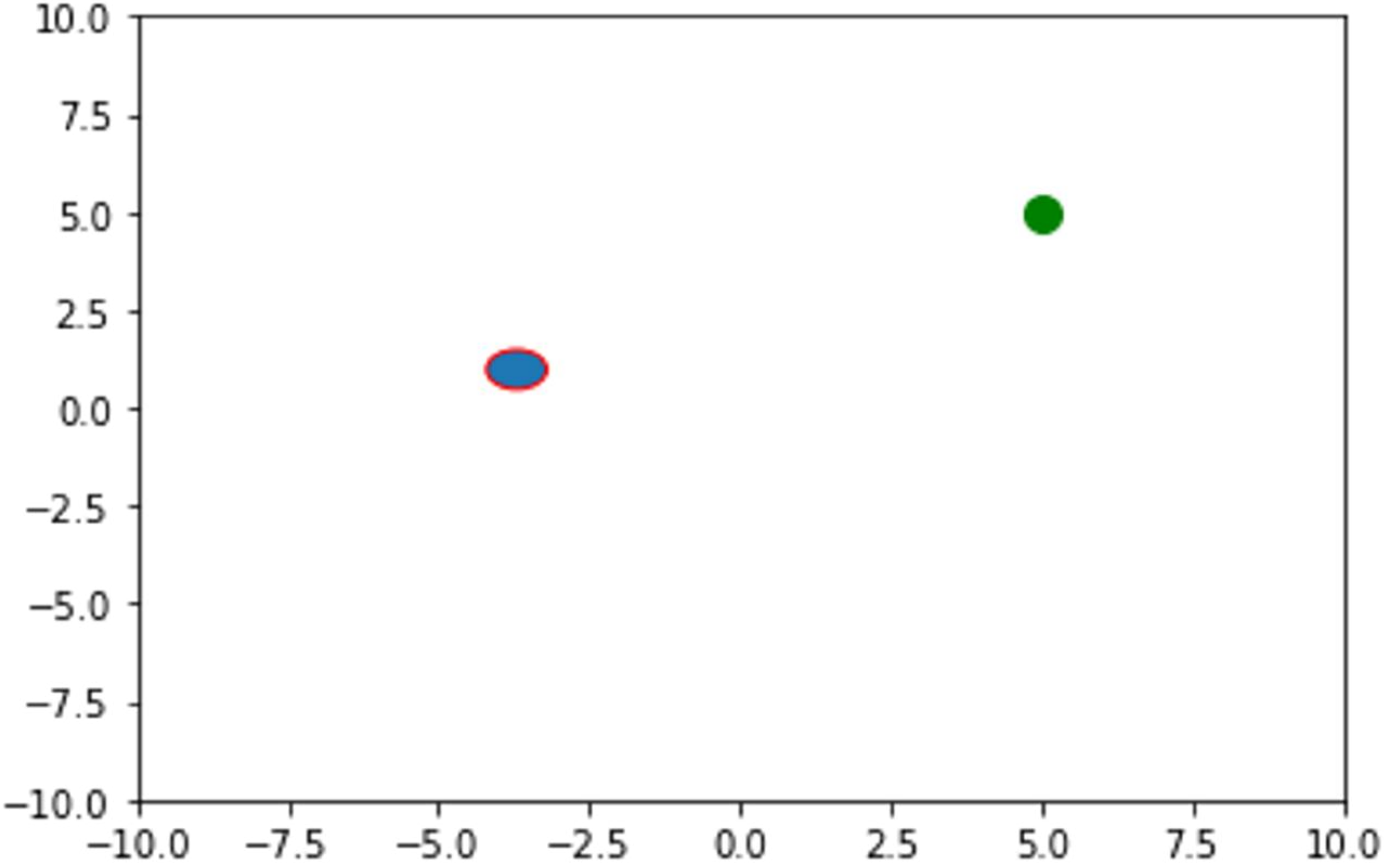
Likelihood Ratio Gradient – Toy Problem



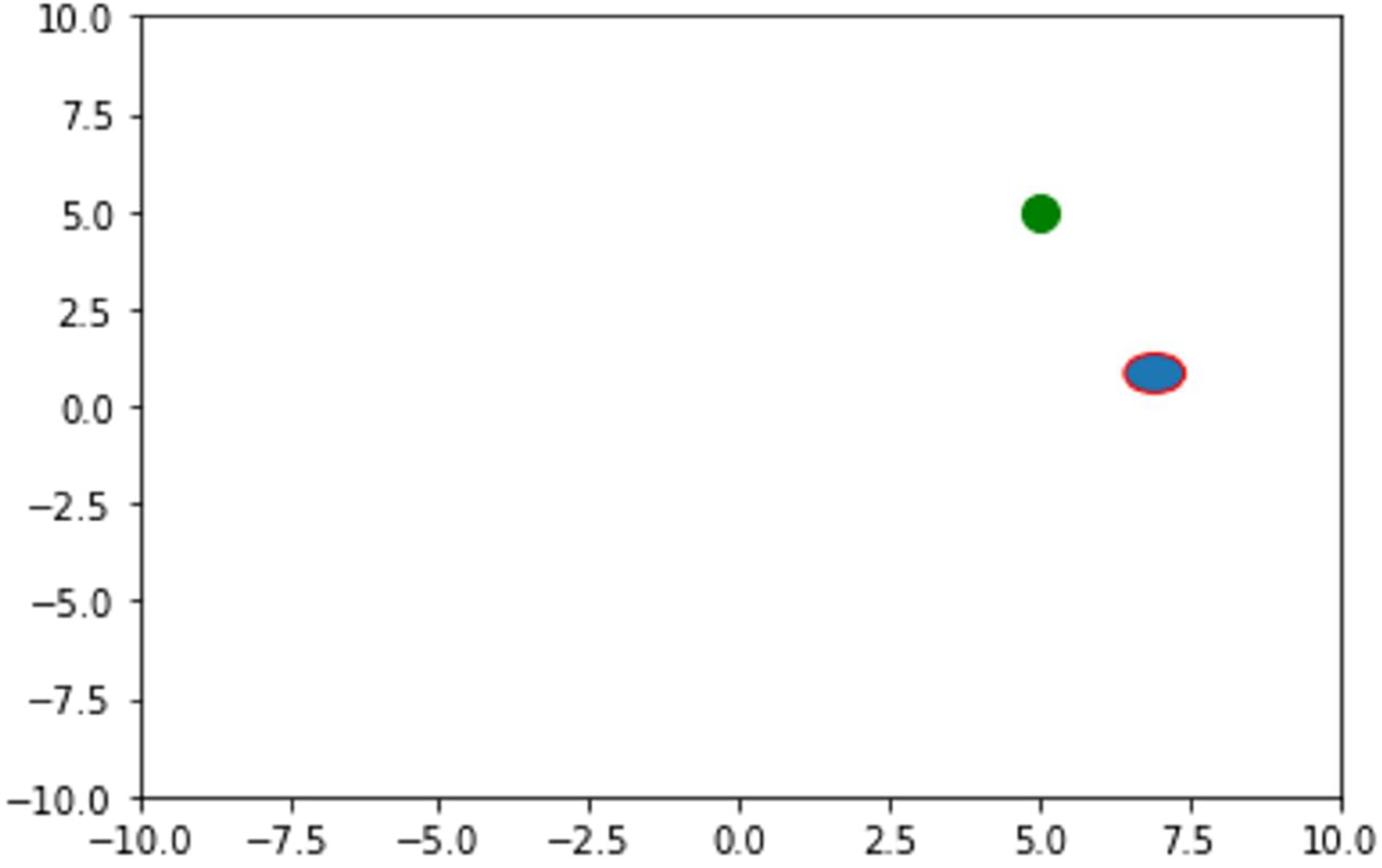
Likelihood Ratio Gradient – Toy Problem



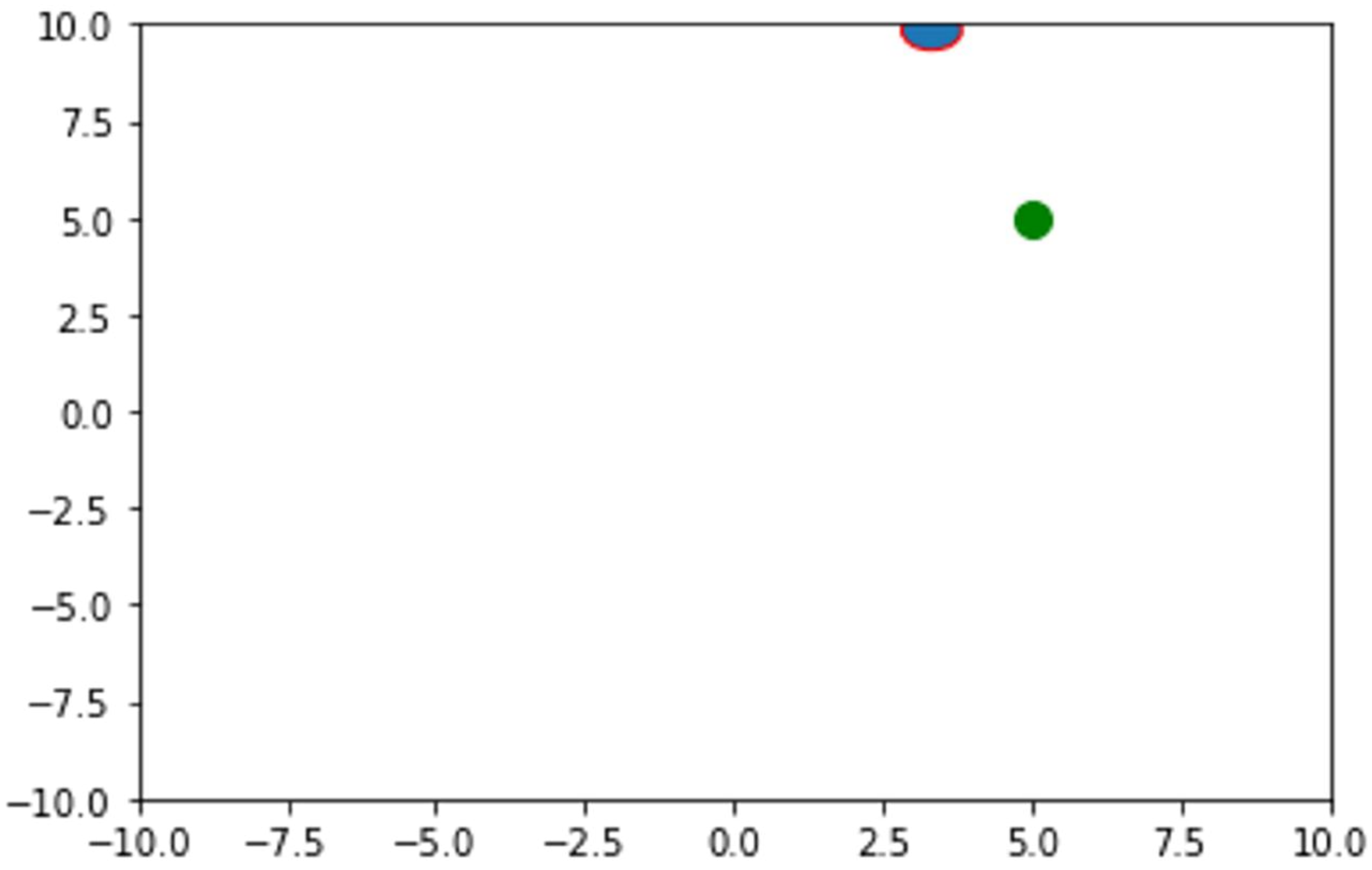
Likelihood Ratio Gradient – Toy Problem



Likelihood Ratio Gradient – Toy Problem



Likelihood Ratio Gradient – Toy Problem



Pathwise Derivative / Reparameterization Trick

$\mathbb{E}_{z\sim q_\phi(z)}[f(z)]$

$$q_\phi(z) = \mathcal{N}(\mu, \sigma^2)$$

$$z = \mu + \epsilon \cdot \sigma \qquad \epsilon \sim \mathcal{N}(0,1)$$

$$= \mathbb{E}_{\underbrace{\epsilon\sim\mathcal{N}(0,1)}_{\text{no } \phi \text{ here}}}[f(\mu + \epsilon\sigma)]$$

$$\approx \frac{1}{k}\sum_{i=1}^{k} f(\mu + \epsilon^{(i)}\sigma)$$

- We can then compute $\nabla_{\mu,\sigma}\left(\frac{1}{k}\sum_{i=1}^{k} f(\mu + \epsilon^{(i)}\sigma)\right)$

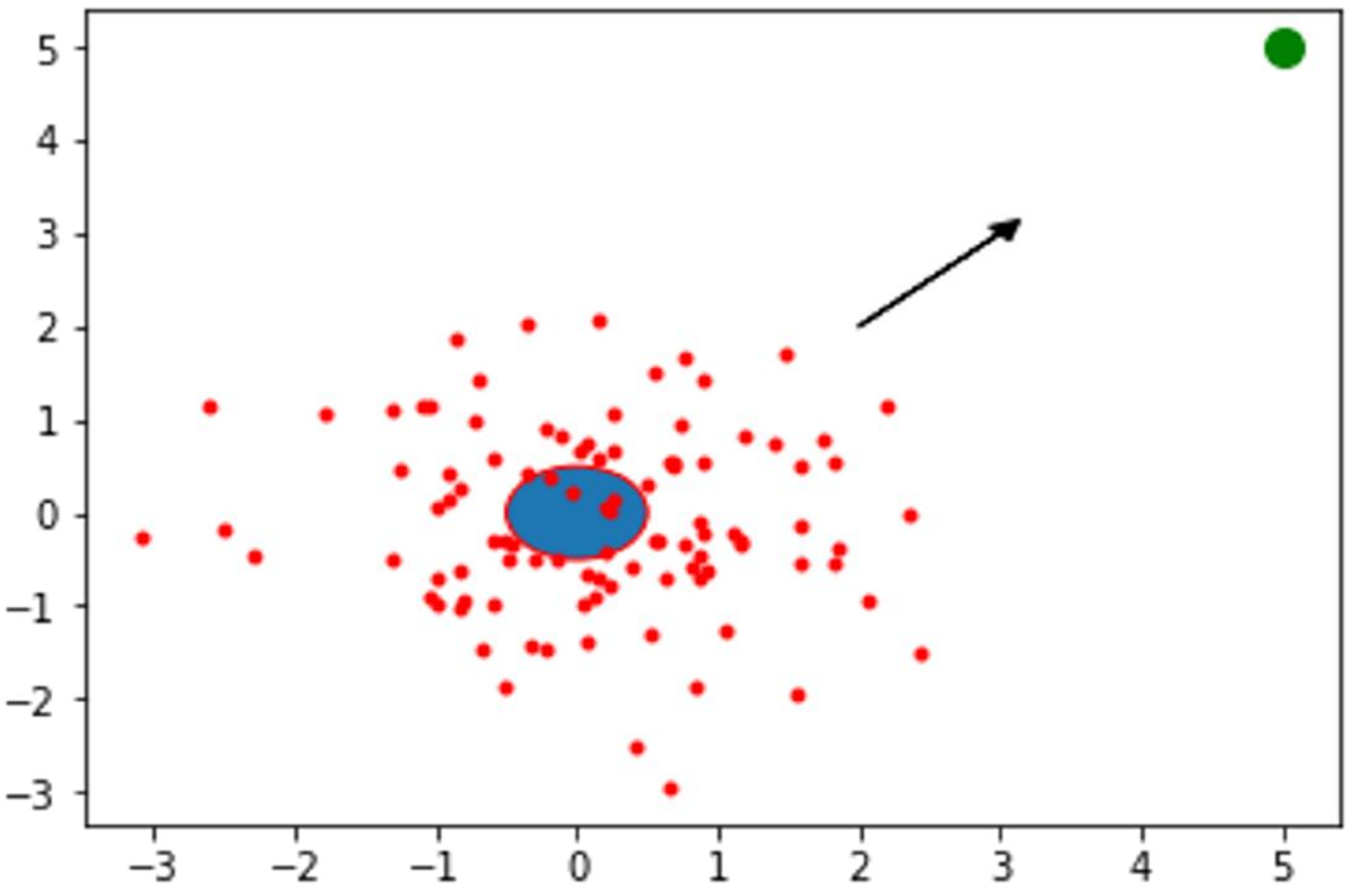
Pathwise Derivative (PD)

- Stochastic gradient possible if z is continuous now (more technical condition?)
  - Common choice: $\epsilon \sim \mathcal{N}, f(\epsilon) = \mu + \sigma\epsilon$
  - Any flow that you just learned!
- Also known as **reparameterization trick**
- Can work with only 1~2 samples

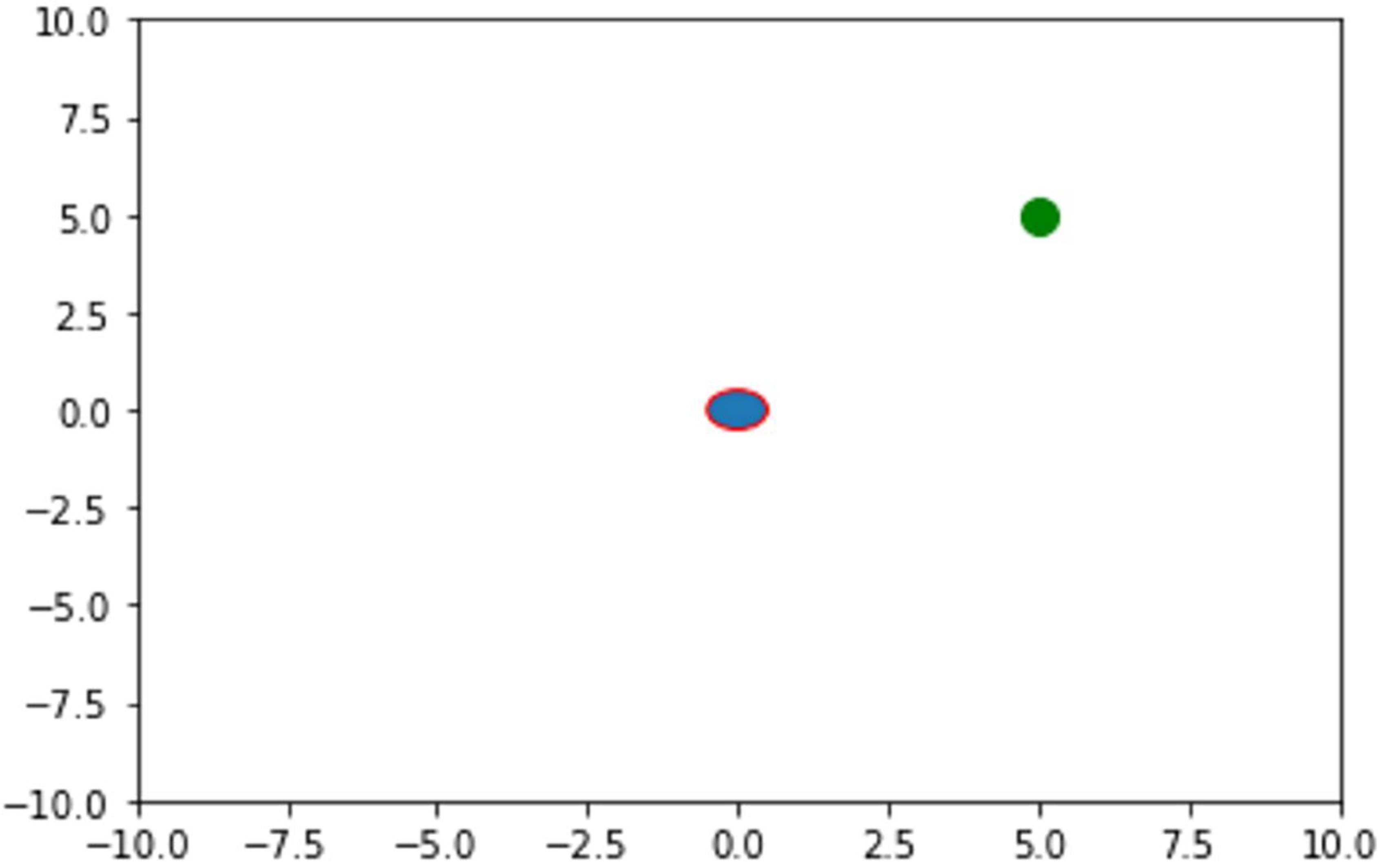
Pathwise Derivative – Toy Problem

Learn $\mu \in \mathbb{R}^2$ to minimize the objective below to reach the green point

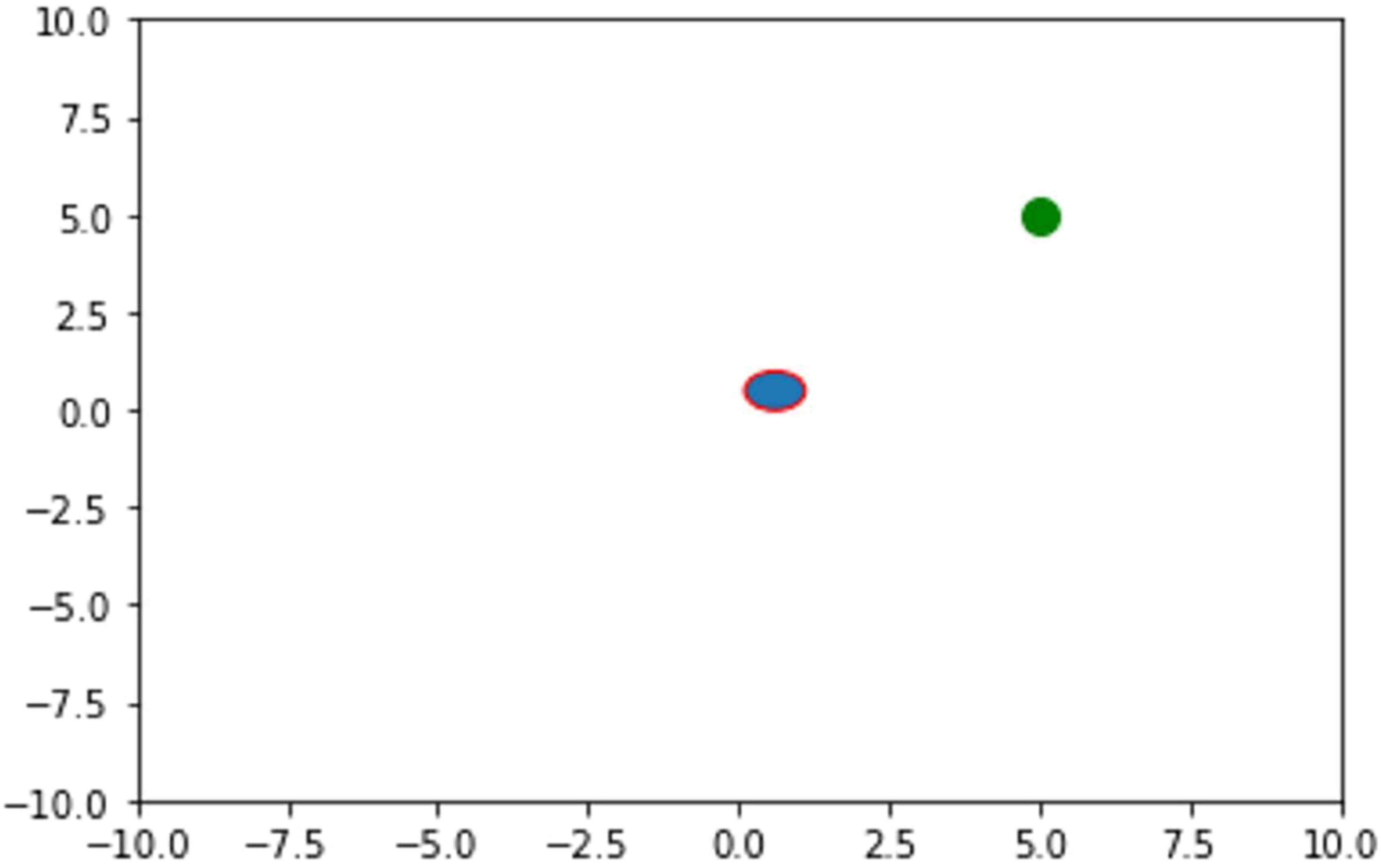
$$\mathcal{L} = \mathbb{E}_{x \sim N(\mu, I)} \left\| x - \begin{bmatrix} 5 \\ 5 \end{bmatrix} \right\|_2^2$$



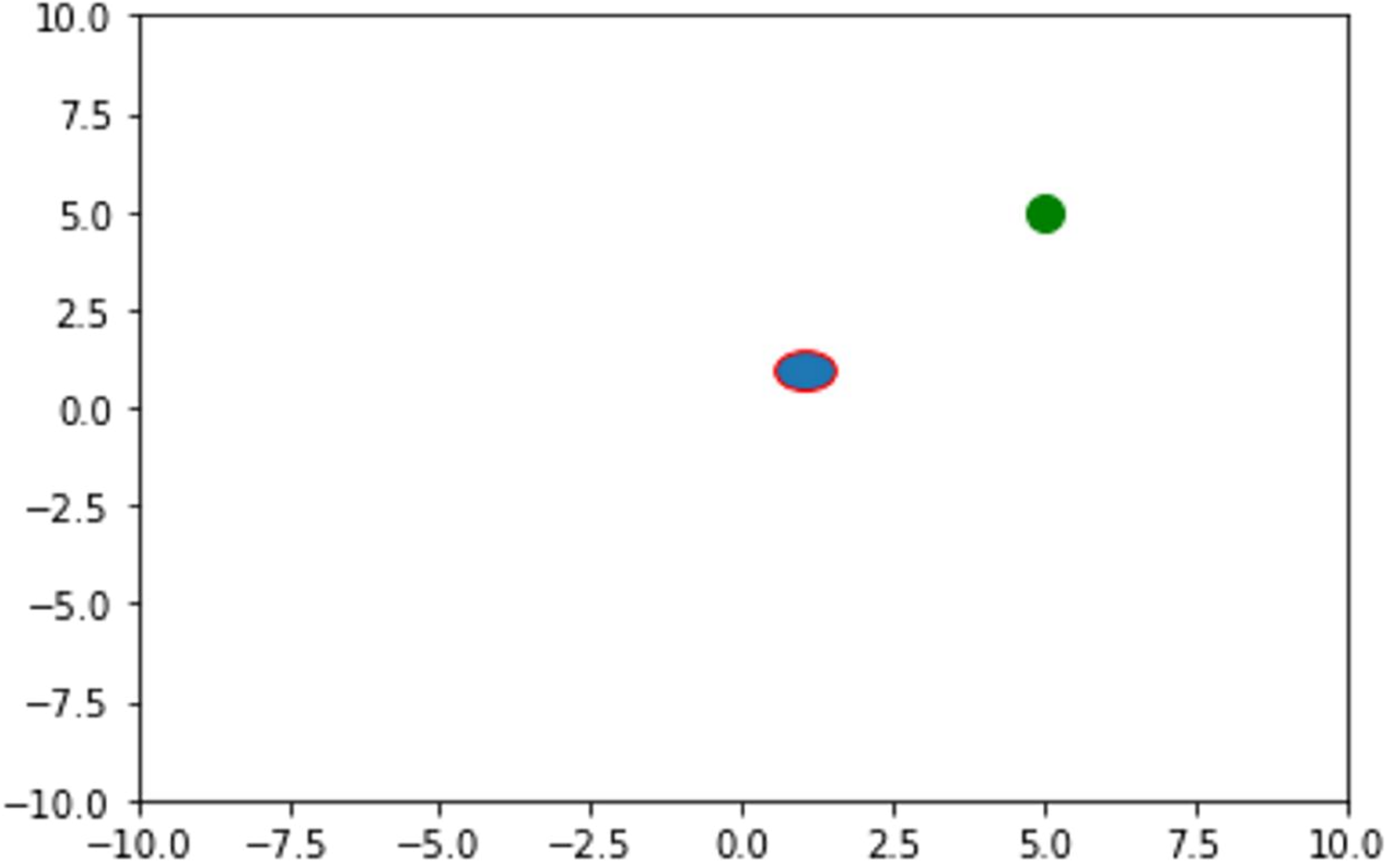
Pathwise Derivative – Toy Problem



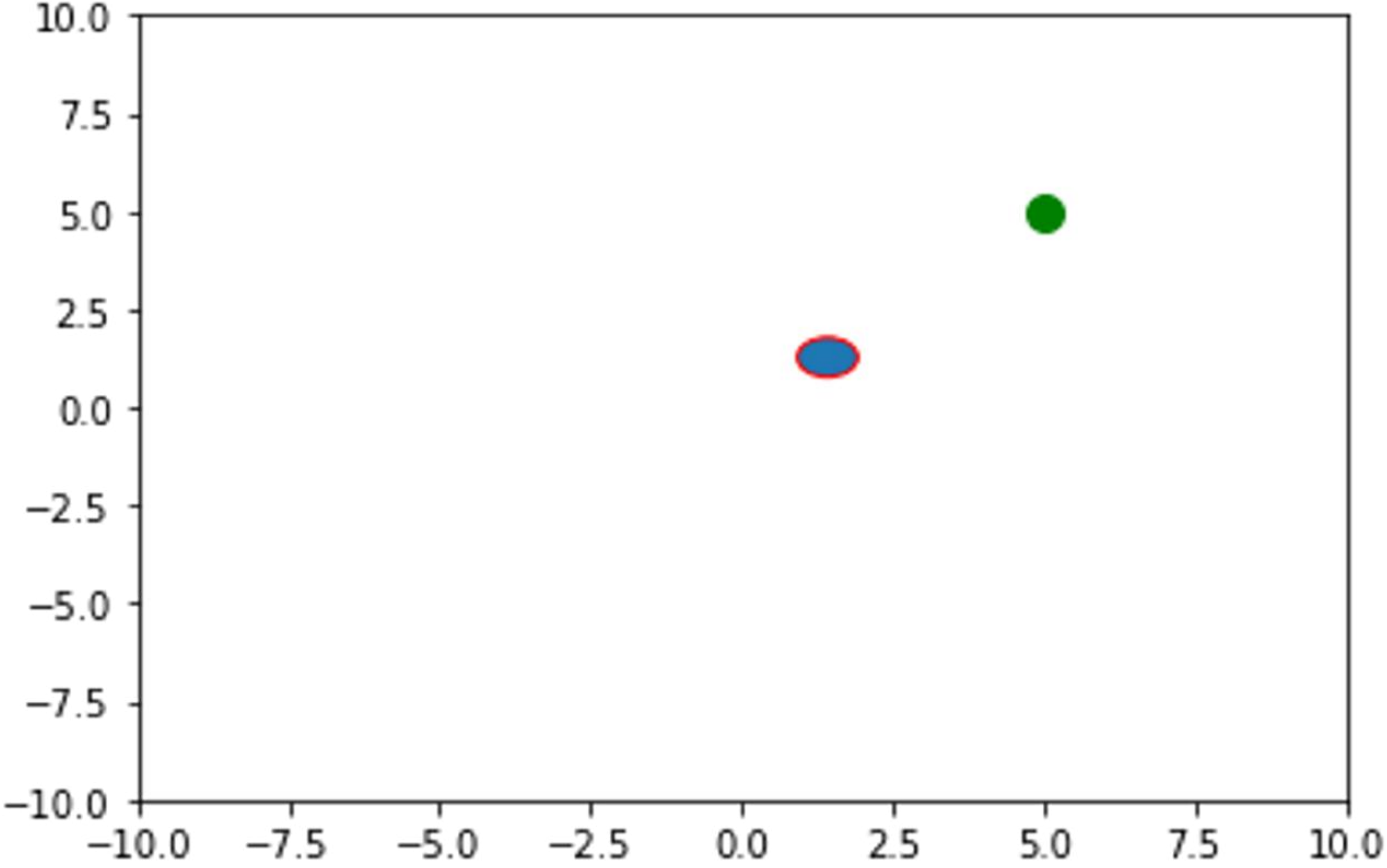
Pathwise Derivative – Toy Problem



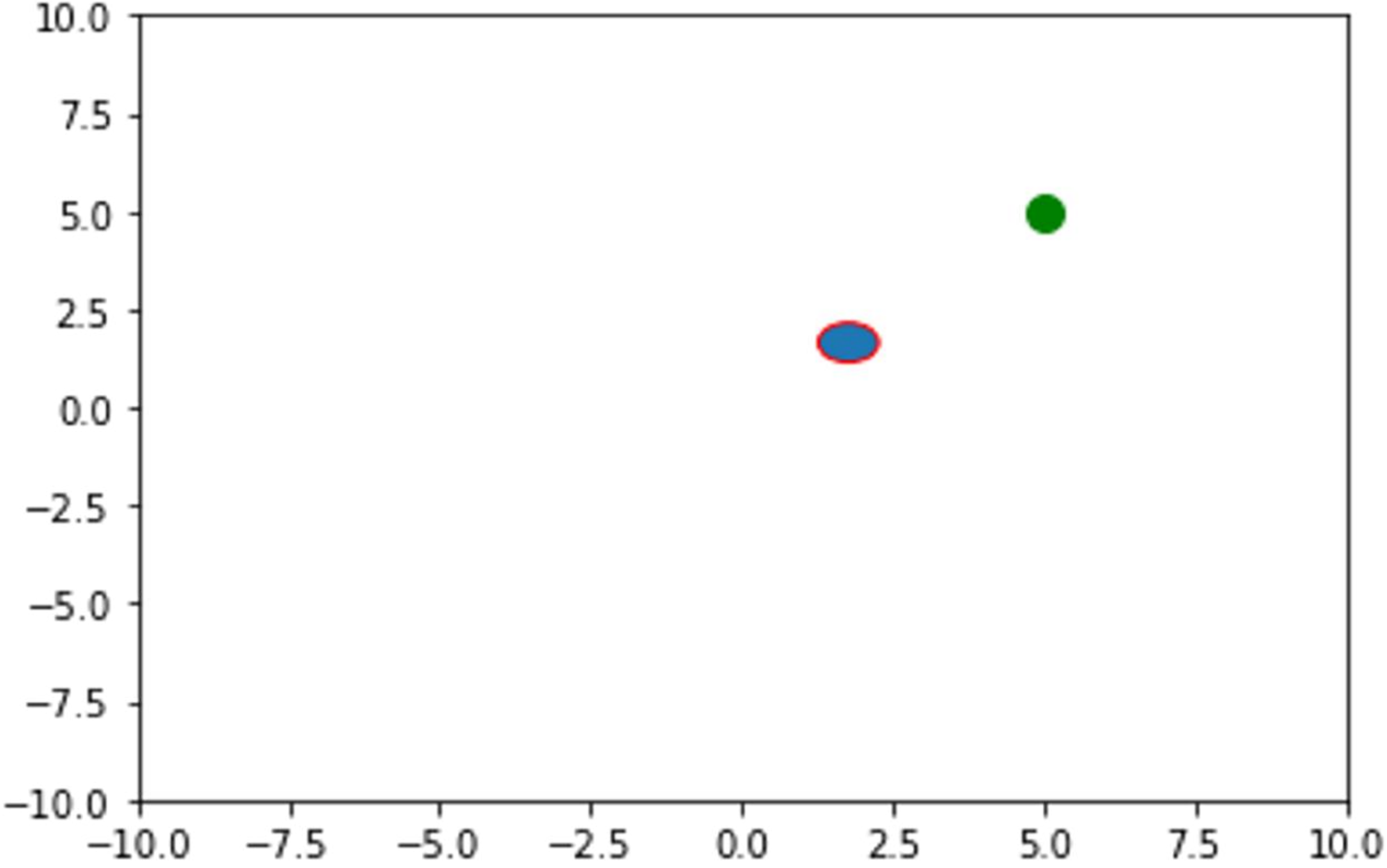
Pathwise Derivative – Toy Problem



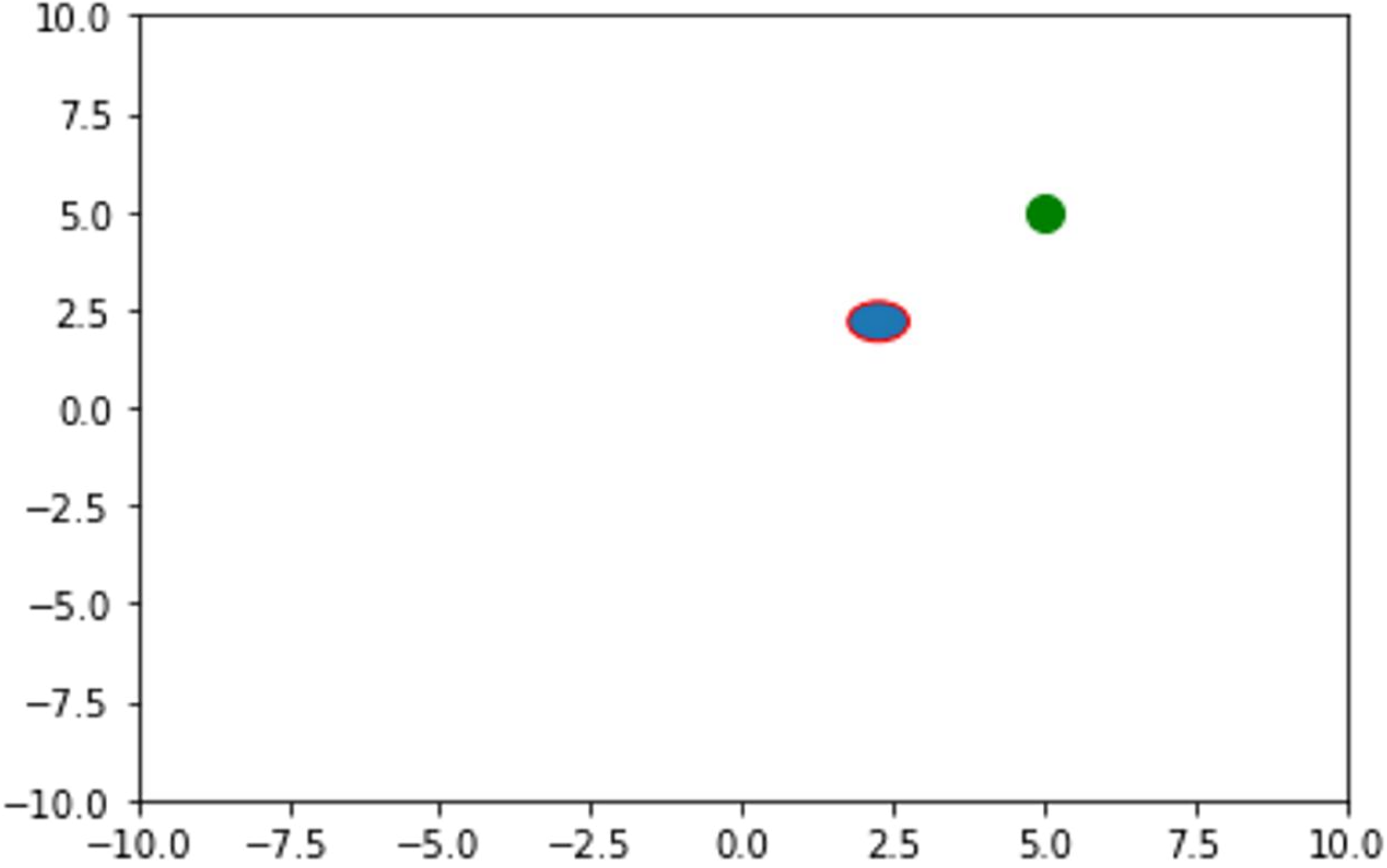
Pathwise Derivative – Toy Problem



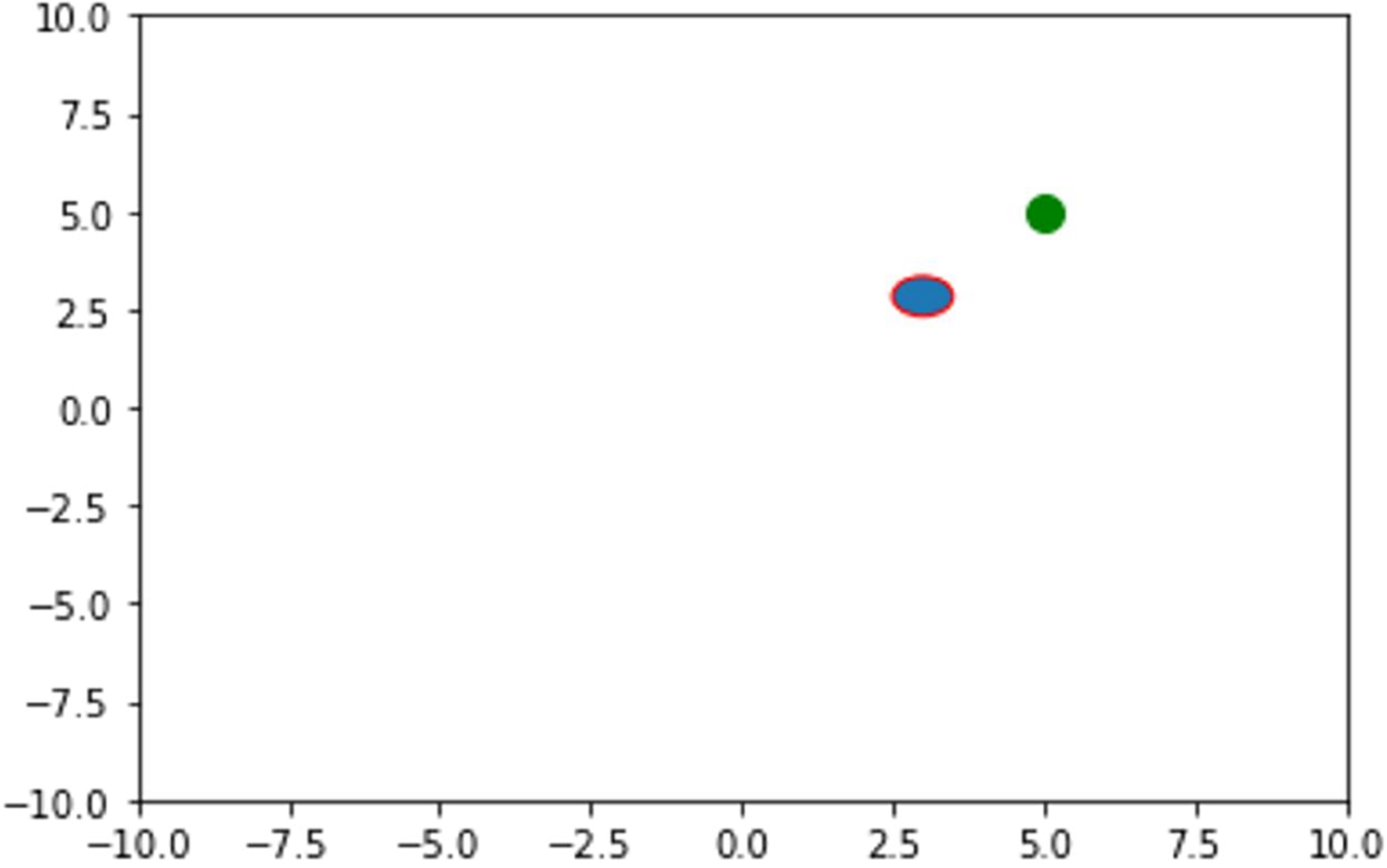
Pathwise Derivative – Toy Problem



Pathwise Derivative – Toy Problem



Pathwise Derivative – Toy Problem



Pathwise Derivative (PD)

One other way to optimize this objective when z is continuous is to cast z as a function of a simple fixed noise such as standard gaussian.

$$z = g(\epsilon, \phi), \epsilon \sim \mathcal{N}(0, I)$$

$$\mathbb{E}_{z \sim q_\phi(\cdot|x)}\left[f(z)\right] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}\left[f(g(\epsilon, \phi))\right]$$

When f is differentiable,

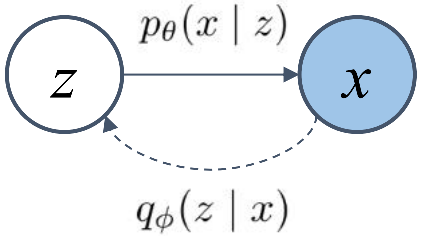
$$\nabla_\phi \mathbb{E}_{z \sim q_\phi(\cdot|x)}\left[f(z)\right] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}\left[\nabla_\phi f(g(\epsilon, \phi))\right]$$

Reparameterizing distributions

- Any distribution in the scale-location family (Laplace, Cauchy, Student's t, etc.) can be reparameterized in the same way.
- Distributions like Gamma and Dirichlet can be parameterized using implicit reparameterization.
- Not every distribution can be reparameterized in a differentiable way.
  - For example, discrete variables are not reparameterizable in this way.
- Deep learning frameworks such as TensorFlow and PyTorch support reparameterization for many continuous distributions, making it easy to propagate gradients through samples.

Slide credit: Andriy Mnih

VAE and Likelihood Ratio Gradient



$$\max_{\theta,\phi} \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log p_z(z) + \log p_\theta(x^{(i)}|z) - \log q_\phi(z|x^{(i)})]$$

$$\nabla_\theta(\cdot) = \nabla_\theta \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)] \approx \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \log p_\theta(x^{(i)}|z^{(k)})$$

$$z^{(k)} \sim q_\phi(z|x^{(i)})$$

$$\nabla_\phi(\cdot) = \mathbb{E}_{z\sim q_\phi(z|x^{(i)})} \nabla_\phi \log q_\phi(z|x^{(i)})[\log p_z(z) + \log p_\theta(x^{(i)}|z) - \log q_\phi(z|x^{(i)})]$$

$$+ \underbrace{\mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[-\nabla_\phi \log q_\phi(z|x^{(i)})]}$$

$$\mathbb{E}_{z\sim q_\phi(z|x^{(i)})}\left[-\frac{\nabla_\phi q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})}\right] = \mathbb{E}_z\left[-q_\phi(z|x^{(i)})\frac{\nabla_\phi q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})}\right]$$

$$= \nabla_\phi \textstyle\sum_z q_\phi(z|x^{(i)}) = 1$$

Likelihood Ratio Estimator

We are interested in $\text{argmax}_\phi \mathbb{E}_{z \sim q_\phi(\cdot|x)} [f(z)]$

How do we compute $\nabla_\phi \mathbb{E}_{z \sim q_\phi(\cdot|x)} [f(z)]$?

$$\nabla_\phi \sum_z q_\phi(z|x) f(z) = \sum_z \nabla_\phi q_\phi(z|x) f(z) = \sum_z \underbrace{\frac{\nabla_\phi q_\phi(z|x)}{q_\phi(z|x)}} f(z) q_\phi(z|x)$$

$$\Rightarrow \nabla_\phi \mathbb{E}_{z \sim q_\phi(\cdot|x)} [f(z)] = \sum_z \left(\nabla_\phi \log q_\phi(z|x) f(z)\right) q_\phi(z|x) = \mathbb{E}_{z \sim q_\phi(\cdot|x)} [\nabla_\phi \log q_\phi(z|x) f(z)]$$

$$\phi \leftarrow \phi + \alpha \nabla_\phi \mathbb{E}_{z \sim q_\phi(\cdot|x)} [\nabla_\phi \log q_\phi(z|x) f(z)]$$

Issue: High variance gradients, needs many samples of z to form a good estimate

VAE and Pathwise Derivative

One other way to optimize this objective when z is continuous is to cast z as a function of a simple fixed noise such as standard gaussian.

$$z = g(\epsilon, \phi), \epsilon \sim \mathcal{N}(0, I)$$

$$\mathbb{E}_{z \sim q_\phi(\cdot|x)}\left[f(z)\right] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}\left[f(g(\epsilon, \phi))\right]$$

When f is differentiable,

$$\nabla_\phi \mathbb{E}_{z \sim q_\phi(\cdot|x)}\left[f(z)\right] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}\left[\nabla_\phi f(g(\epsilon, \phi))\right]$$

PD applied to VI

Variational AutoEncoder

$q_\phi(z|x)$ is modeled as a Gaussian with parameters μ and σ a DNN encoder (parameters ϕ) of x. The DNN decoder $p_\theta(x|z)$ is differentiable.

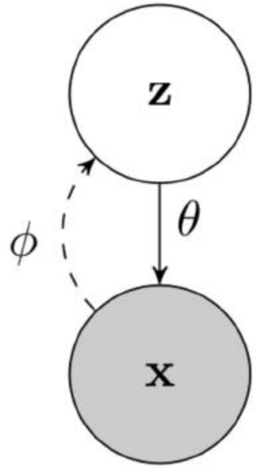
$$\text{Let } z = \Sigma^{1/2}(x;\phi)\epsilon + \mu(x;\phi)$$

$$\begin{aligned}\text{VLB} &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\log p_\theta(x|z) - \log q_\phi(z|x) + \log p(z)\right] \\ &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\log p_\theta(x|z)\right] - KL(q_\phi(z|x)||p(z))\end{aligned}$$

$\nabla_\theta\,[\text{VLB}]$ and $\nabla_\phi\,[\text{VLB}]$ can now be efficiently computed with SGD.

VAE

$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$$

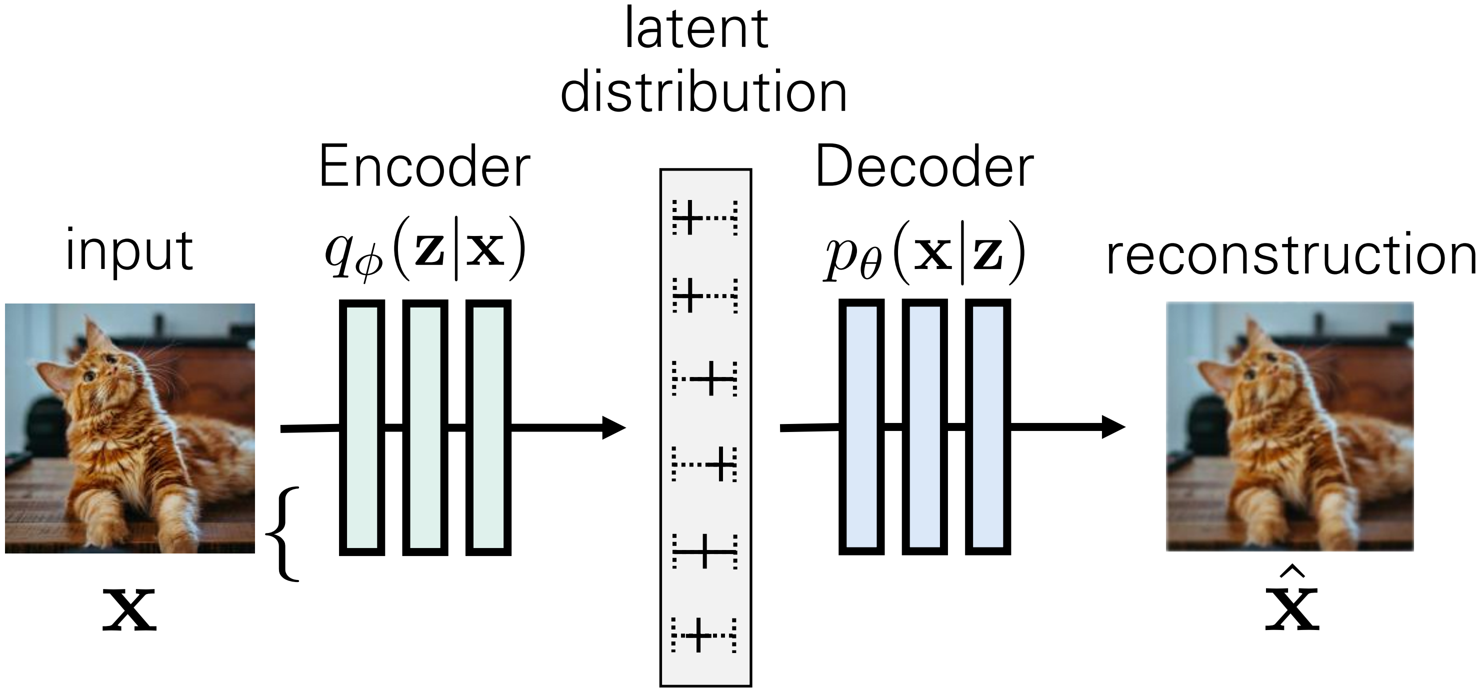


$$p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma_\theta(z))$$

- **Prior:** $p(\mathbf{z}) = \mathcal{N}(0, I)$
- **Likelihood / decoder:**
  - For binary data: $p_\theta(\mathbf{x}|\mathbf{z}) = \mathrm{Bernoulli}(\mathrm{NN}_\theta(\mathbf{z}))$
  - For real-valued data: $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathrm{NN}_\theta(\mathbf{z}), \mathrm{diag}(\mathrm{NN}_\theta(\mathbf{z})))$
- **Variational posterior / encoder:** $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathrm{NN}_\phi(\mathbf{x}), \mathrm{diag}(\mathrm{NN}_\phi(\mathbf{x})))$
- Can also use other types of neural nets (e.g. ConvNets) instead of fully-connected neural networks.

VAEs in a Nutshell

- Assume a known distribution (e.g. Normal distribution) on the latent space, $p(\mathbf{z})$.
- Parametrize the conditional probability $p(\mathbf{x}|\mathbf{z})$ by a neural network.
- Approximate $p(\mathbf{z}|\mathbf{x})$ using a variational distribution $q(\mathbf{z}|\mathbf{x})$, also parametrized by a neural network.



$\mathbf{z} \sim \mathcal{N}(0, I)$ (Kingma and Welling, 2014)

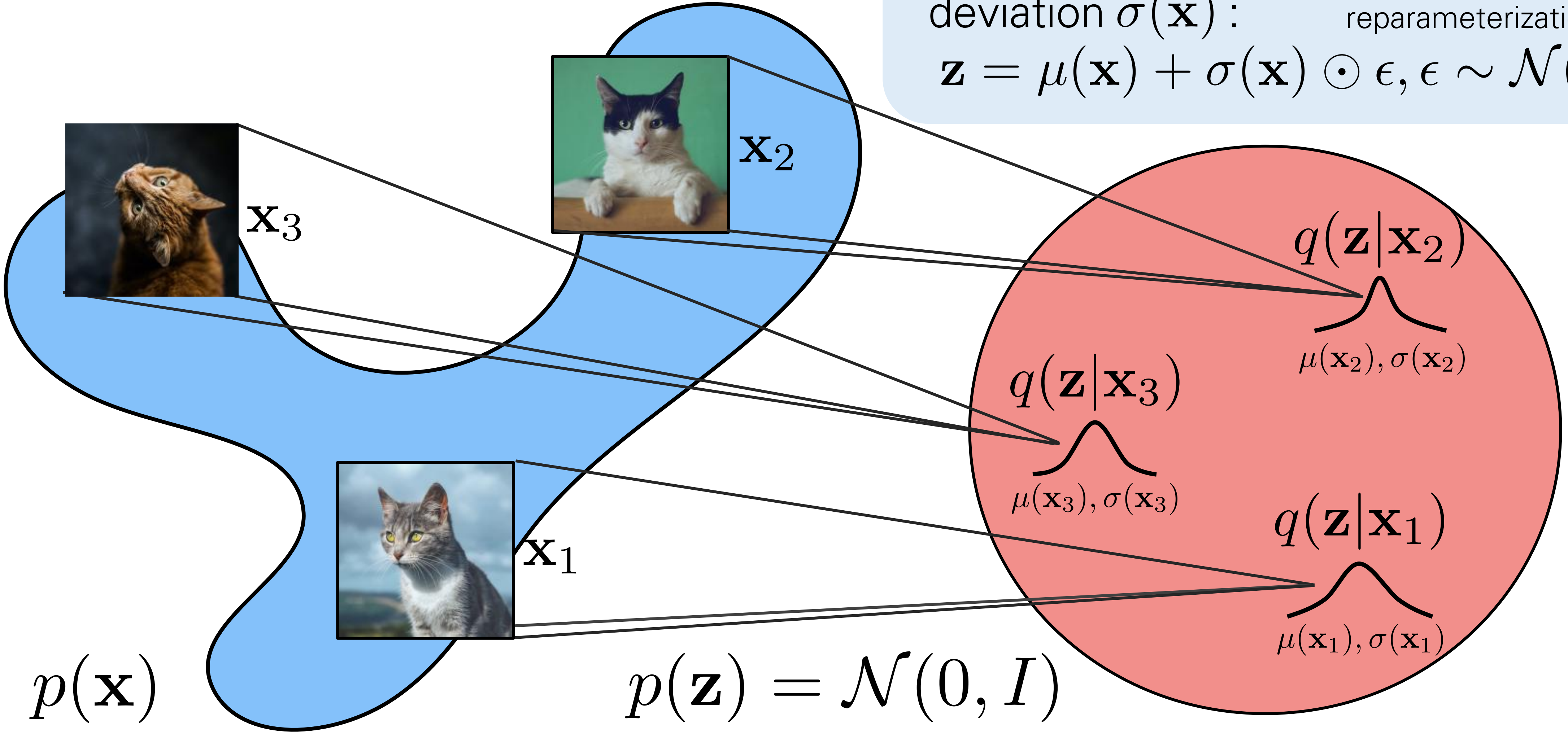
$$\mathcal{L}_{VAE}(\mathbf{x};\theta,\phi) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log p_\theta(\mathbf{x}|\mathbf{z})\right]}_{\text{Reconstruction cost}} + \underbrace{KL\left(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\right)}_{\text{Regularization}}$$

Slide adapted from Loic Matthey, Irina Higgins

Intuition behind VAEs

To get the latent code $\mathbf{z}$, we sample from the mean $\mu(\mathbf{x})$ and the standart deviation $\sigma(\mathbf{x})$: reparameterization trick

$$\mathbf{z} = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \odot \epsilon, \epsilon \sim \mathcal{N}(0, I)$$



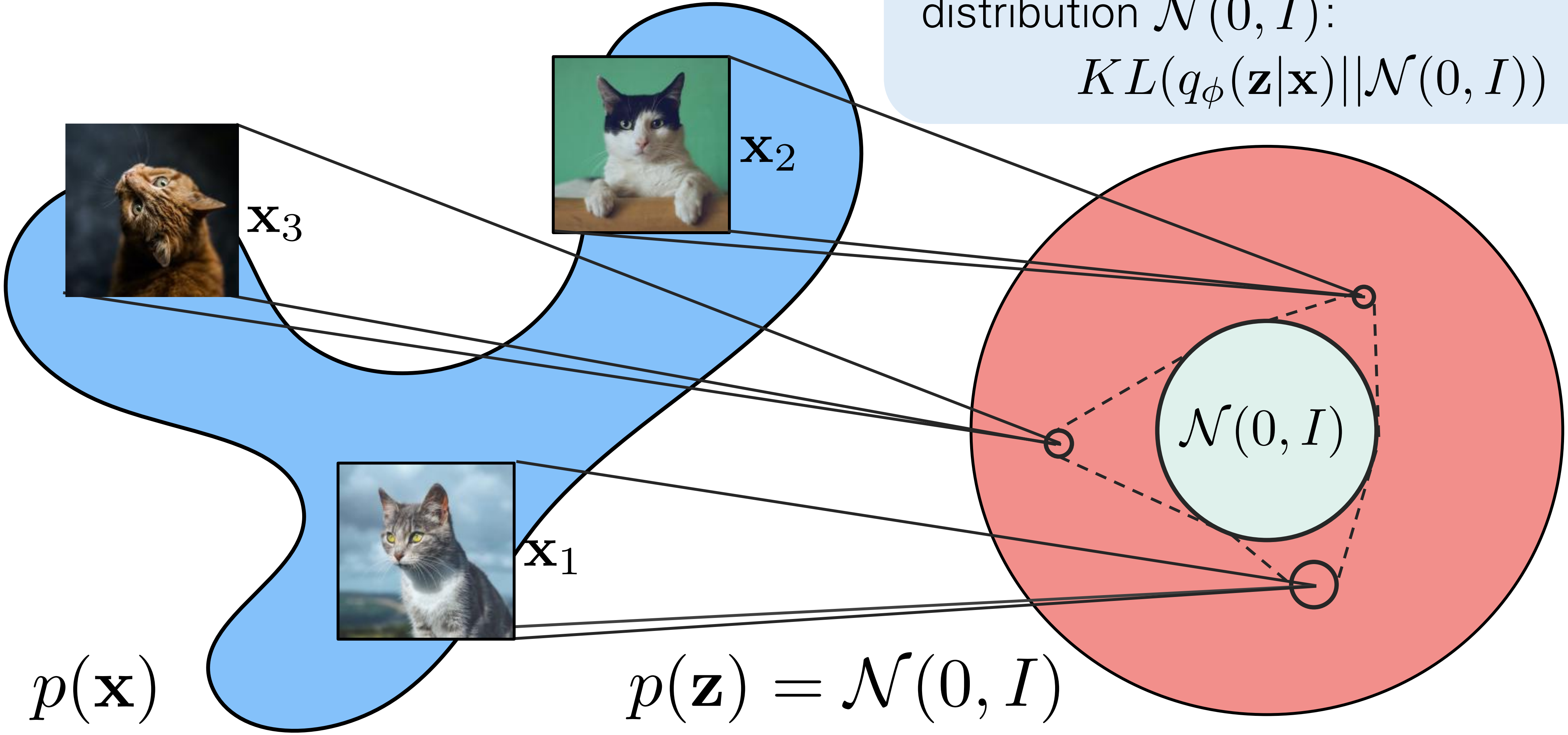
$$\mathcal{L}_{VAE}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log p_\theta(\mathbf{x}|\mathbf{z})\right] + KL\left(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})\right)$$

Slide adapted from Andrea Asperti

Intuition behind VAEs

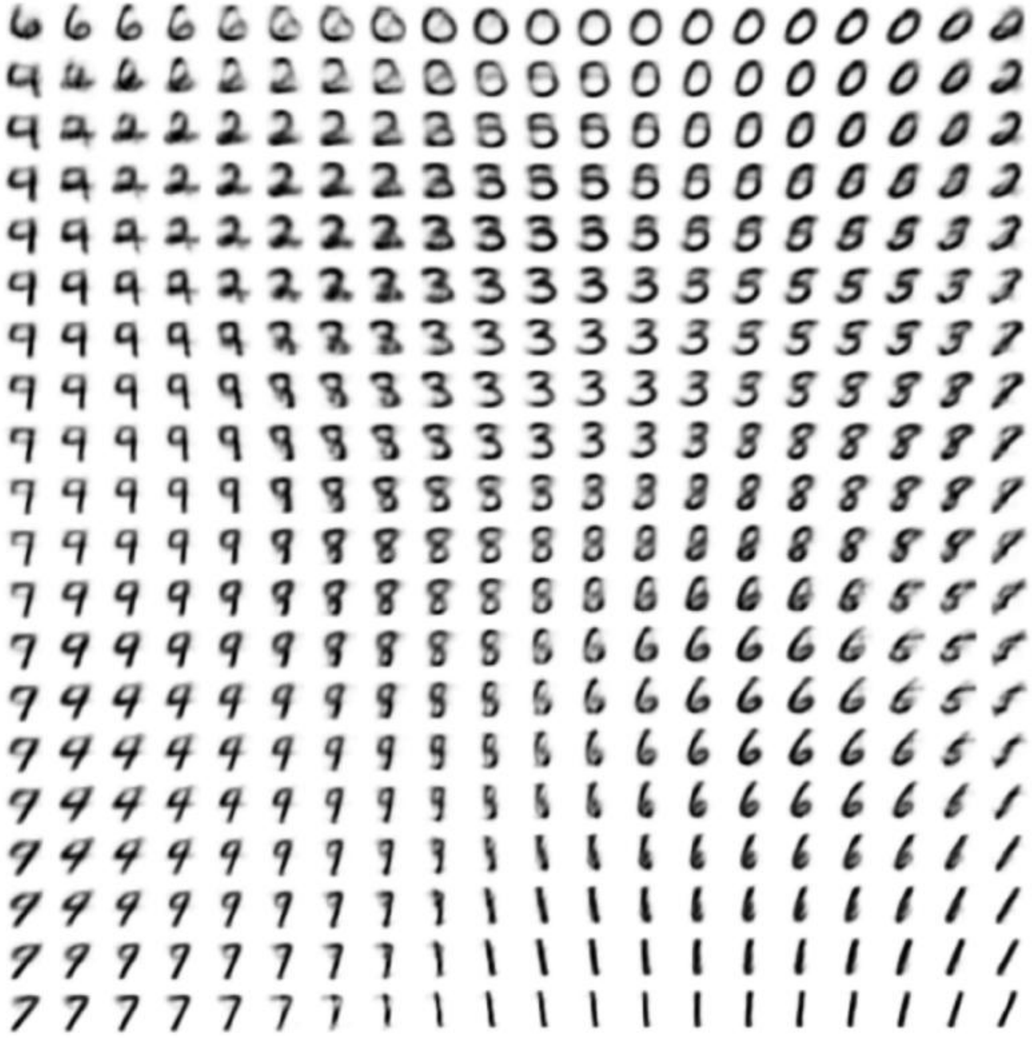
Minimize the Kullback-Leibler distance between each $q(\mathbf{z}|\mathbf{x})$ and the normal distribution $\mathcal{N}(0, I)$:

$$KL(q_\phi(\mathbf{z}|\mathbf{x})||\mathcal{N}(0, I))$$



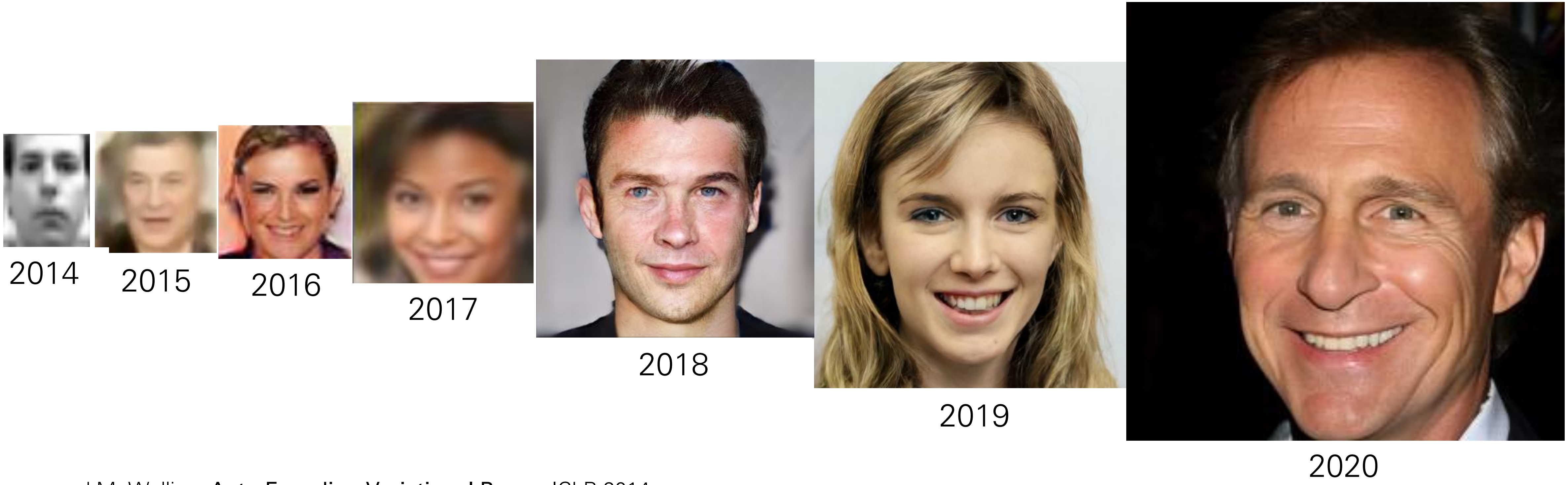
$$\mathcal{L}_{VAE}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log p_\theta(\mathbf{x}|\mathbf{z})\right] + KL(q_\phi(\mathbf{z}|\mathbf{x})||\mathcal{N}(0, I))$$

VAE



Evolutions of VAEs

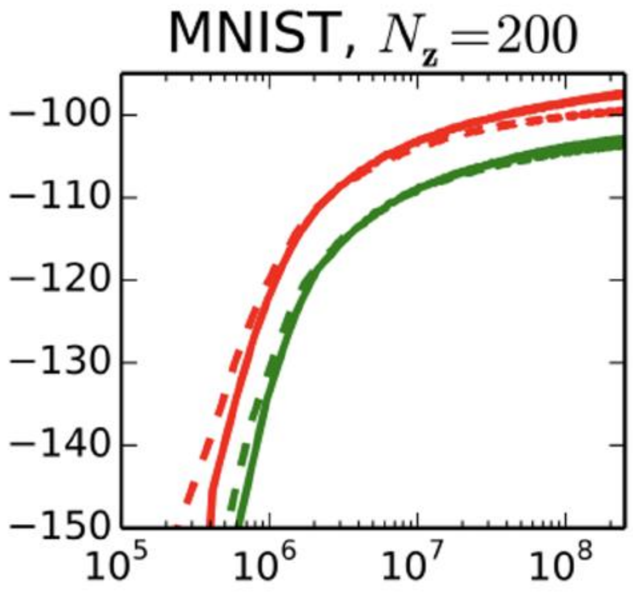
- 6 years of VAE progress on face generation



D.P. Kingma and M. Welling. **Auto-Encoding Variational Bayes**. ICLR 2014.
A. Radford. **Conv/Deconv Variational Autoencoder**. 2015.
A. Boesen L. Larsen, S.K. Sønderby, H. Larochelle, O. Winther. **Autoencoding beyond pixels using a learned similarity metric**. ICML 2016.
L.M. Mescheder, S. Nowozin, A. Geiger. **Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks**. ICML 2017.
H. Huang, Z. Li, R. He, Z. Sun, T. Tan. **IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis**. NeurIPS 2018.
A. Razavi, A. van den Oord, O. Vinyals. **Generating Diverse High-Resolution Images with VQ-VAE-2**. NeurIPS 2019.
A. Vahdat and J. Kautz. **NVAE: A Deep Hierarchical Variational Autoencoder**. NeurIPS 2020.

Compared to AR

- We now have a family of trainable latent variable models!
- But performance is lacking



Model	NLL Test
DBM 2hl [1]:	≈ 84.62
DBN 2hl [2]:	≈ 84.55
NADE [3]:	88.33
EoNADE 2hl (128 orderings) [3]:	85.10
EoNADE-5 2hl (128 orderings) [4]:	84.68
DLGM [5]:	≈ 86.60
DLGM 8 leapfrog steps [6]:	≈ 85.51
DARN 1hl [7]:	≈ 84.13
MADE 2hl (32 masks) [8]:	86.64
DRAW [9]:	≤ 80.97
PixelCNN:	81.30

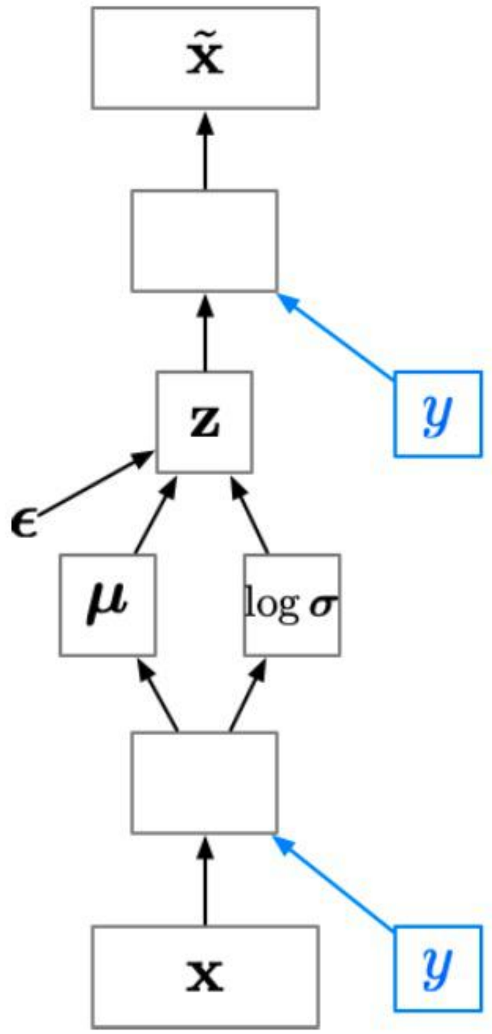
Why is it called an autoencoder?

- We have seen that a variational autoencoder is a latent variable model with Gaussian prior p(z) and approximate posterior q(z|x).
  - Why is it called an "autoencoder"?

$$\underbrace{\log p_\theta(x) \geq \underbrace{\left(E_{z \sim q_x(z)} \log p_\theta(x|z)\right)}_{\text{Reconstruction loss}} \quad - \quad \underbrace{KL(q_\phi(z|x)||p(z))}_{\text{Regularization}}}_{L(\theta,\phi)\text{ - VAE objective}}$$

Class-Conditional VAE

- So far, we haven't used the labels y. A **class-conditional VAE** provides the labels to both the encoder and the decoder.
- Since the latent code z no longer has to model the image category, it can focus on modeling the stylistic features.
- If we're lucky, this lets us **disentangle** style and content. (Note: disentanglement is still a dark art.)
- See Kingma et al., "Semi-supervised learning with deep generative models."

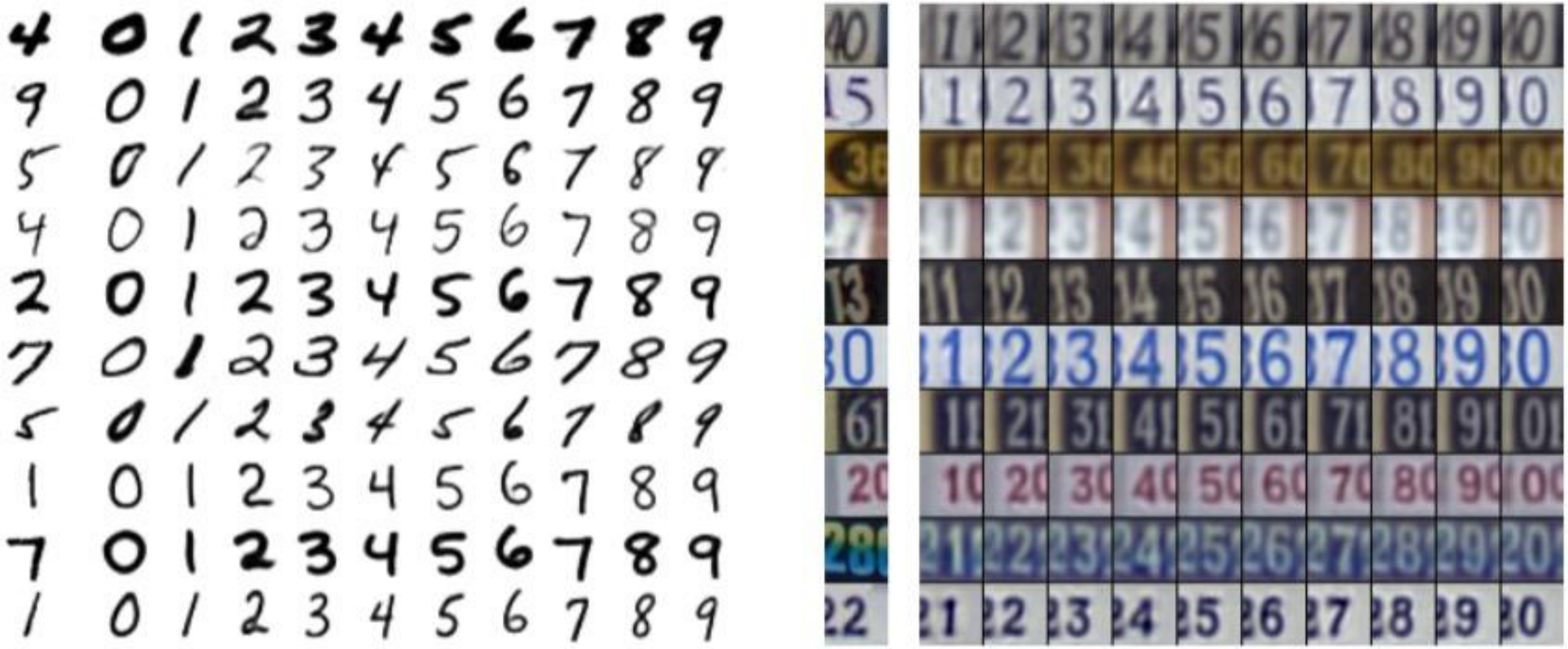


Class-Conditional VAE

- By varying two **latent dimensions** (i.e. dimensions of z) while holding y fixed, we can visualize the **latent space**.

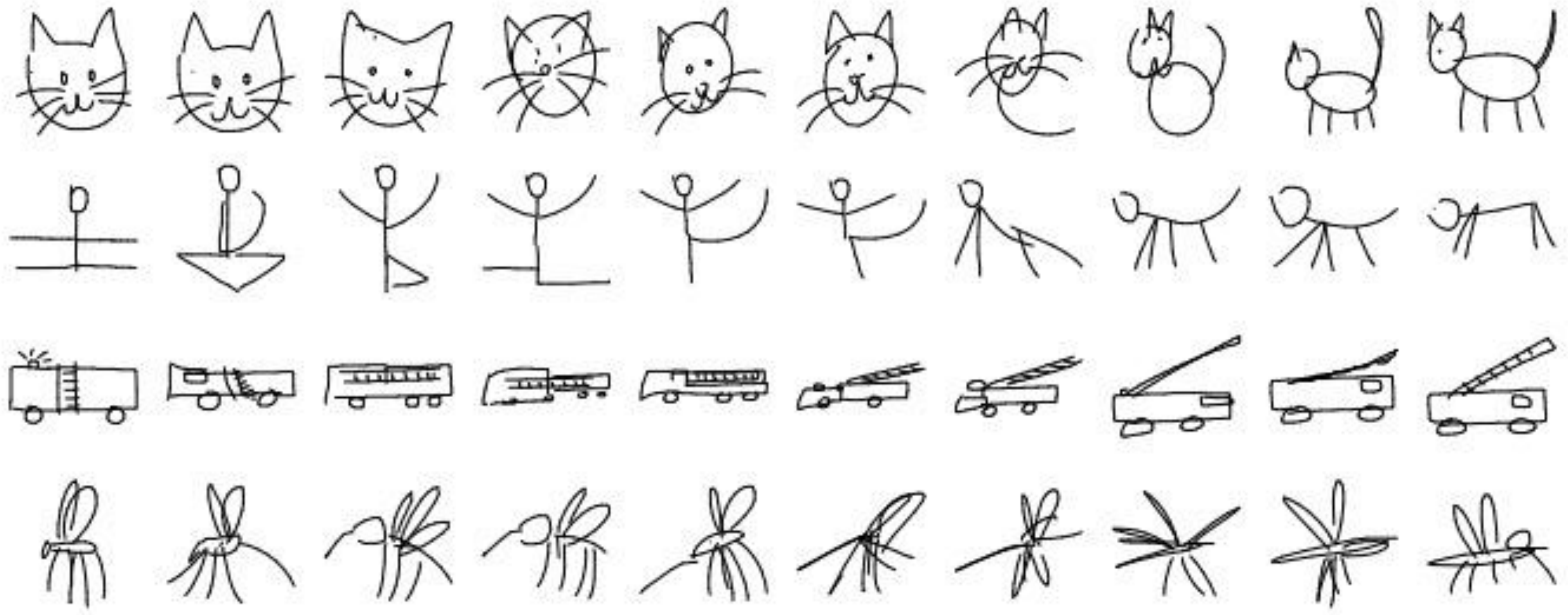
Class-Conditional VAE

- By varying the label y while holding z fixed, we can solve image analogies.



Latent Space Interpolations

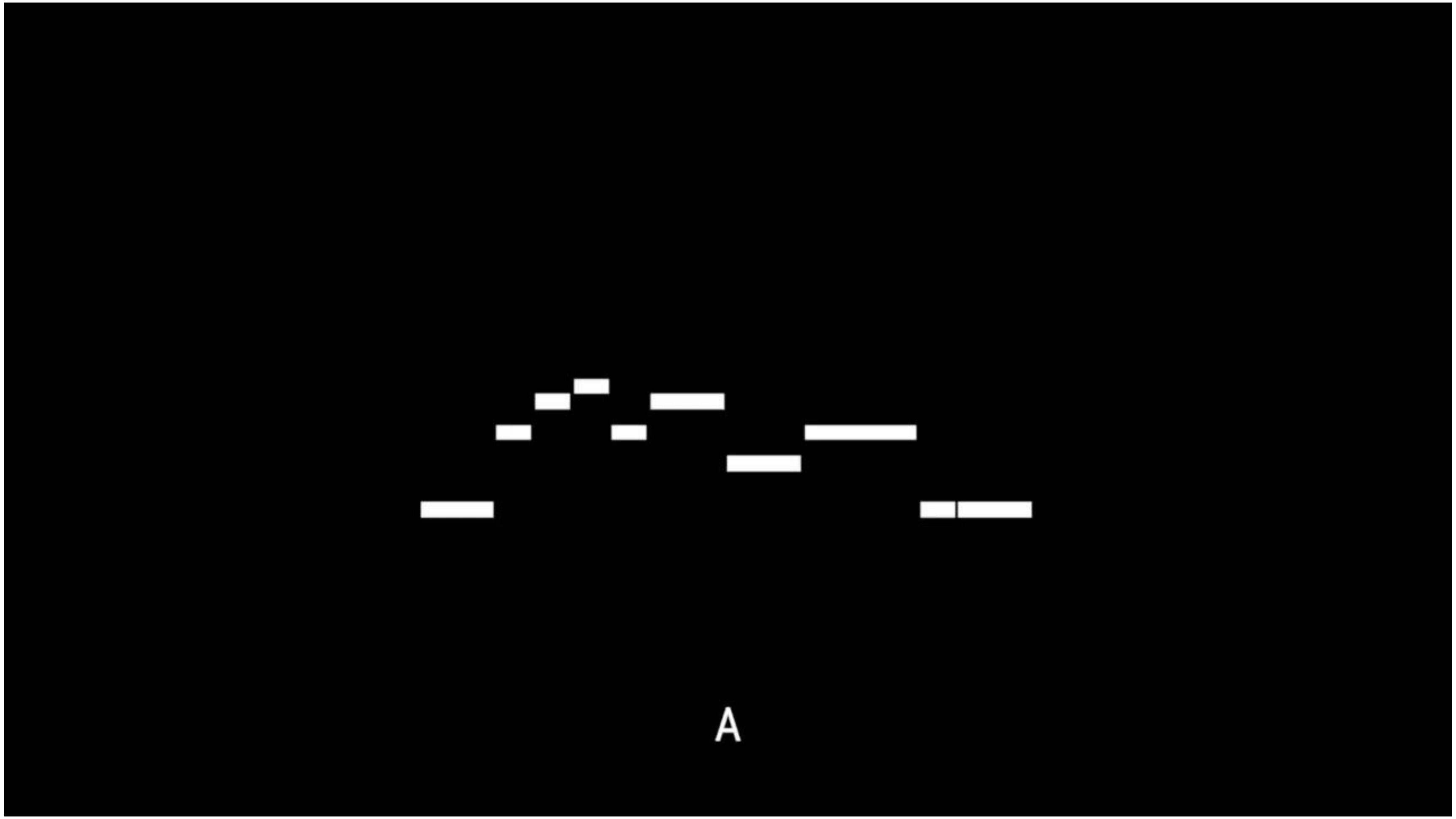
- You can often get interesting results by interpolating between two vectors in the latent space:



Ha and Eck, "A neural representation of sketch drawings"

Latent Space Interpolations

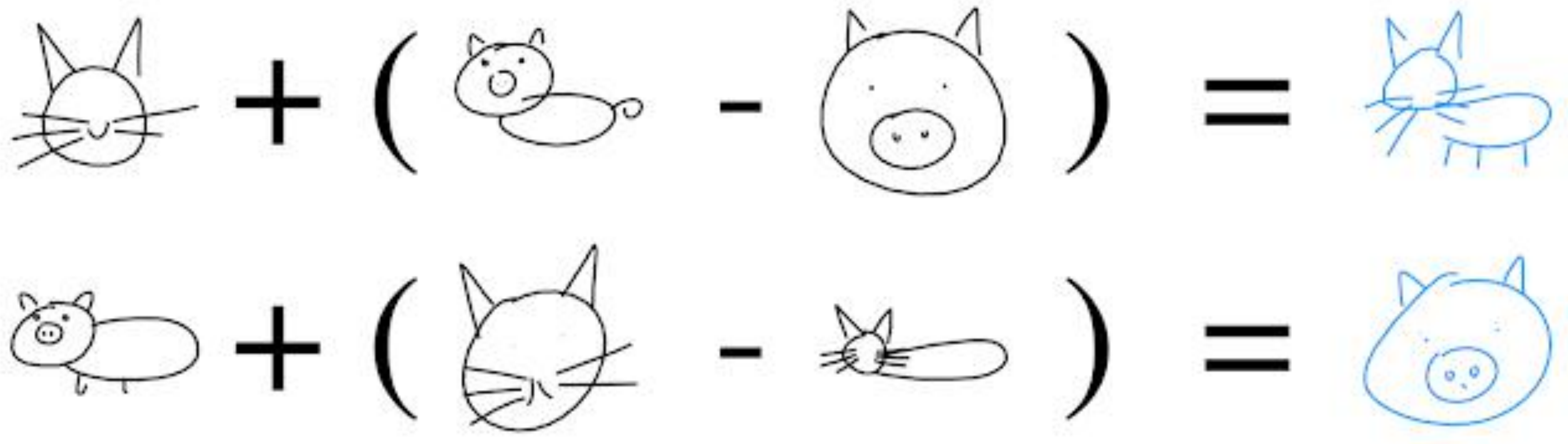
- Latent space interpolation of music:



https://magenta.tensorflow.org/music-vae

Latent Space Arithmetic

- You can even perform vector arithmetic on latent vectors:

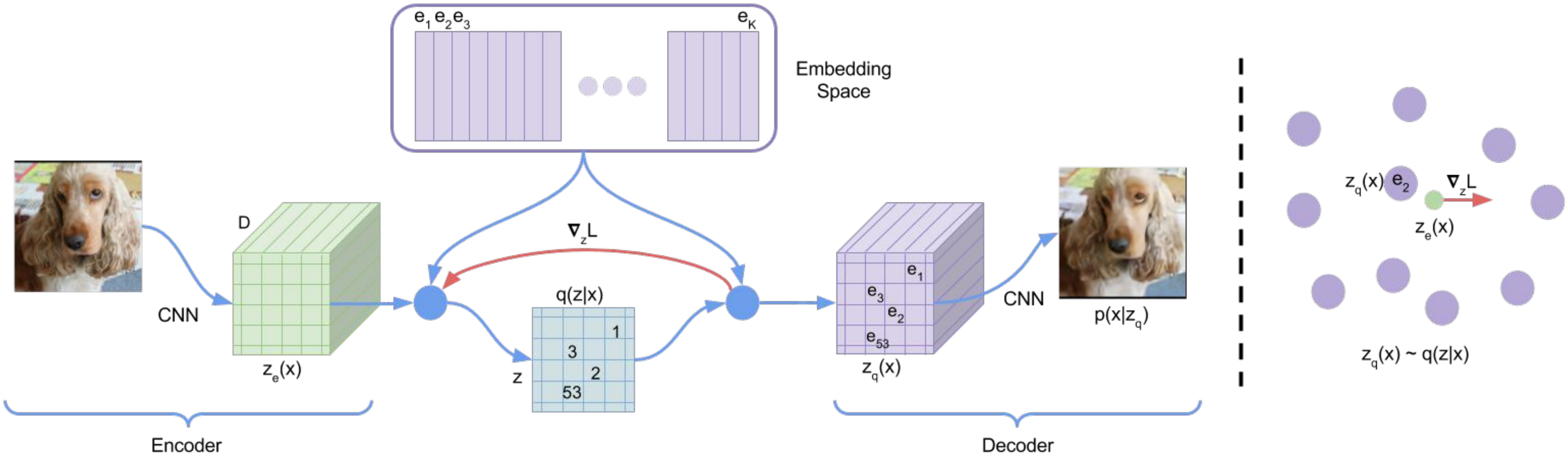


Ha and Eck, "A neural representation of sketch drawings"

Lecture overview

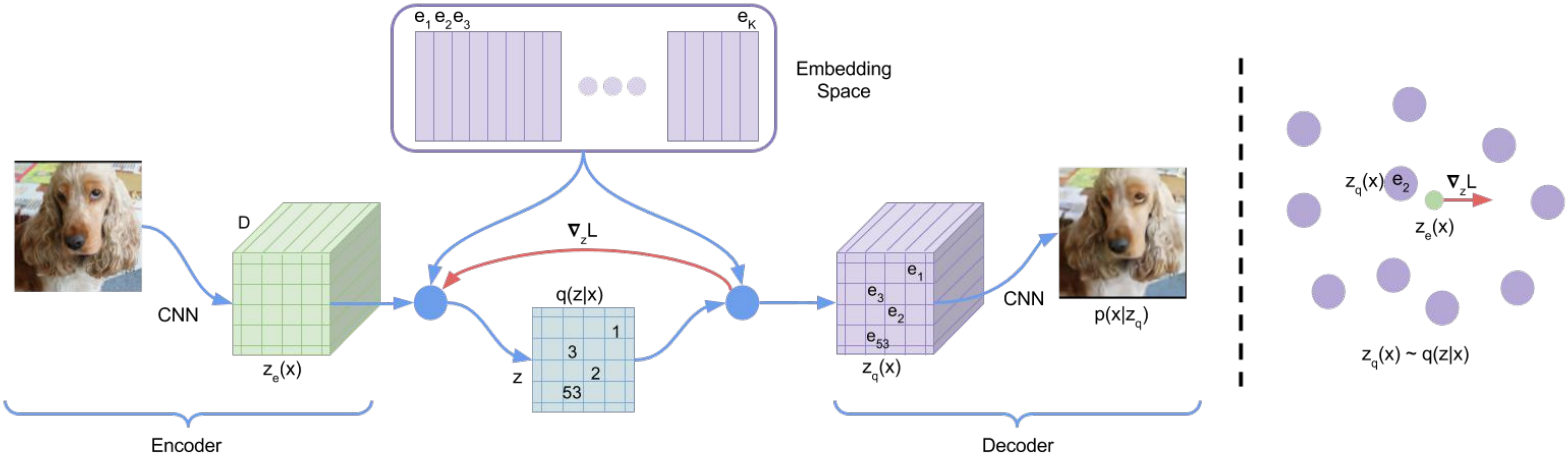
- Motivation
- Training Latent Variable Models (including VAE and IWAE)
- **Variations**
  - **SOTA: VQ-VAE, VQ-VAE 2.0**, NVAE
  - AR + VAE: Variational Lossy AutoEncoder, PixelVAE
  - Disentanglement: Beta VAE
- Related ideas

VQ-VAE



- A latent embedding space $e \in \mathbb{R}^{K \times D}$ where K is the size of the discrete latent space (K-way categorical distribution)
- Encoder output $z_e(x)$ is mapped to discrete latent code z by a nearest neighbour look-up using the shared embedding space e

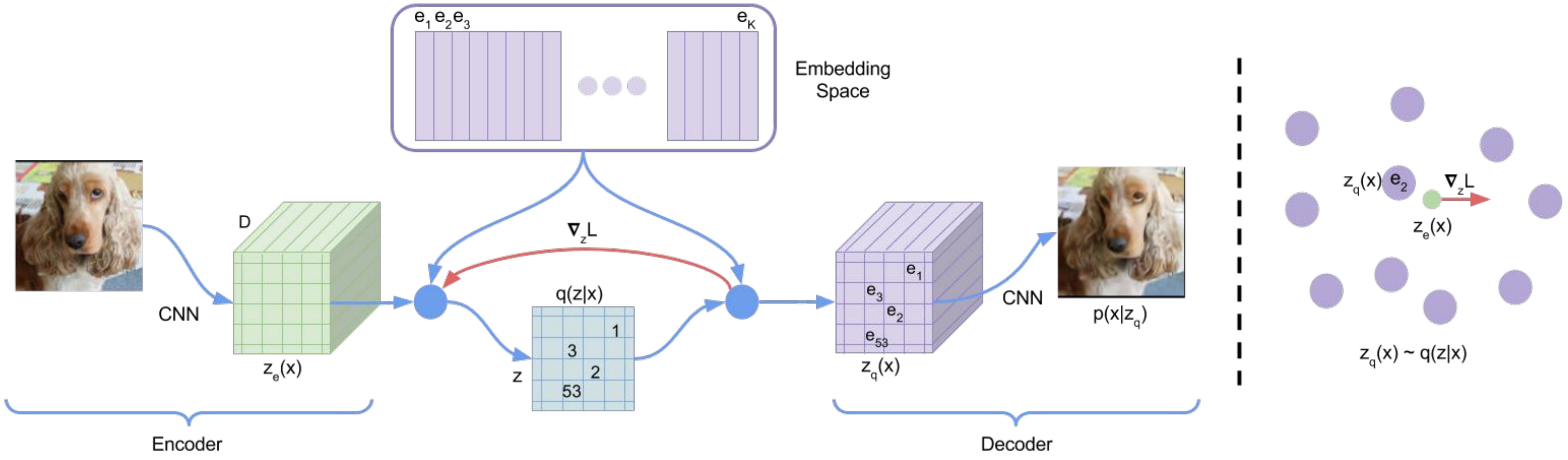
VQ-VAE



- The posterior categorical distribution $q(z|x)$ probabilities are defined as one-hot as:

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

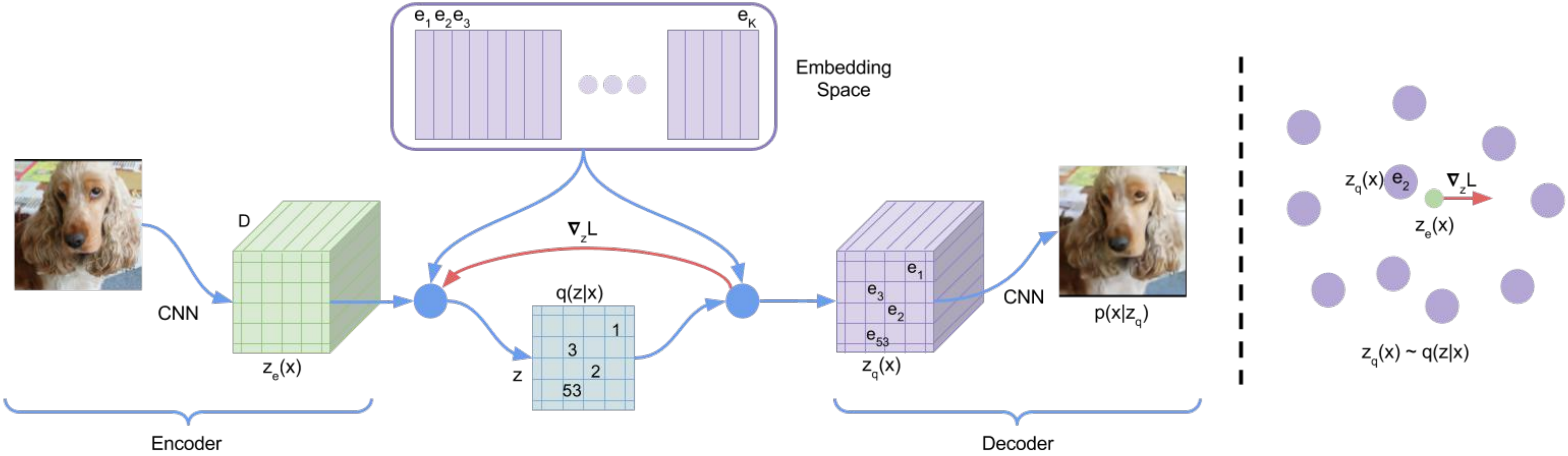
VQ-VAE



- This model can be seen as a VAE in which the proposal distribution $q(z = k|x)$ is deterministic.
- The representation $z_e(x)$ is passed through the discretization bottleneck followed by mapping onto the nearest element of embedding e

VQ-VAE

sg stands for the **stopgradient** operator that is defined as identity at forward computation time and has zero partial derivatives, thus effectively constraining its operand to be a non-updated constant.

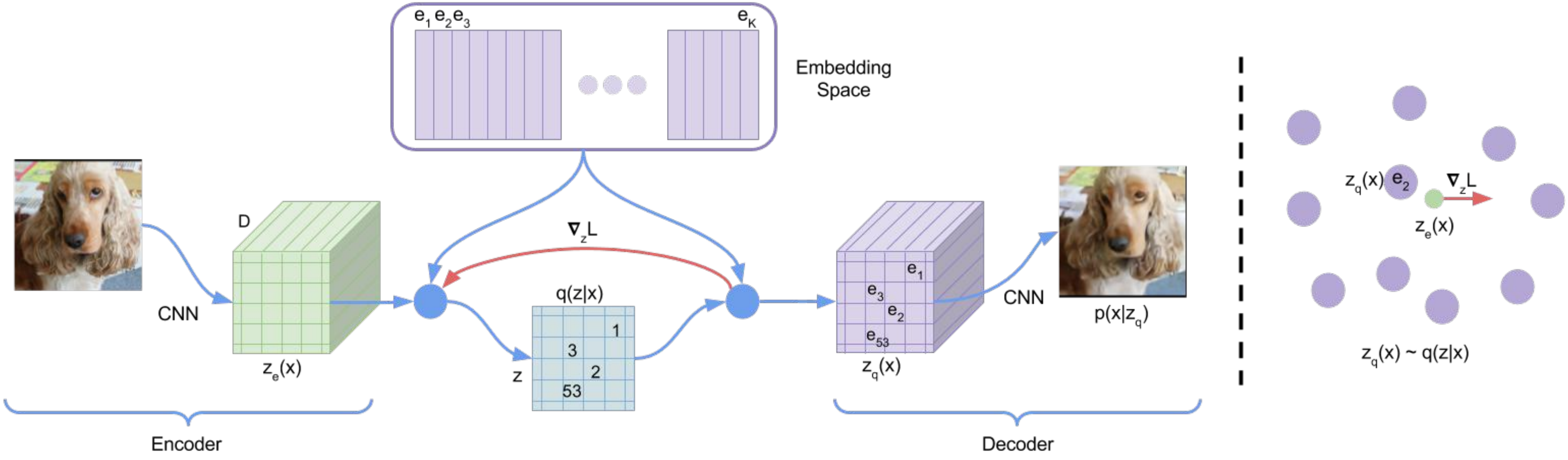


$$z_q(x) = e_k, \quad \text{where} \quad k = \text{argmin}_j \|z_e(x) - e_j\|_2$$

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

VQ-VAE

sg stands for the `stopgradient` operator that is defined as identity at forward computation time and has zero partial derivatives, thus effectively constraining its operand to be a non-updated constant.



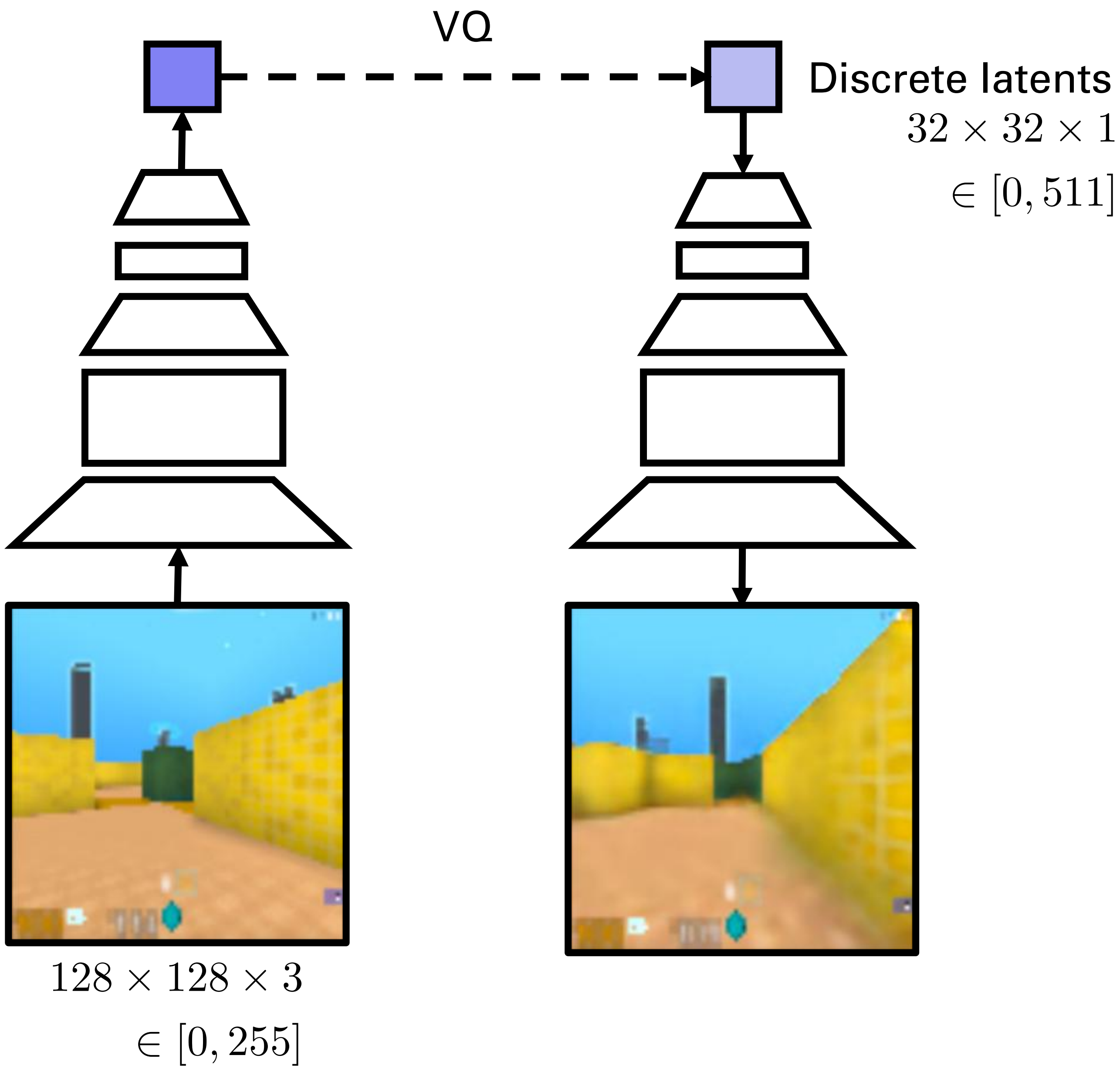
$$L = \log p(x \mid z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \text{sg}[e]\|_2^2$$

The embedding space is learned via Vector Quantization (VQ), whose objective uses the l_2 error to move the embedding vectors e_i towards the encoder outputs $z_e(x)$

VQ-VAE

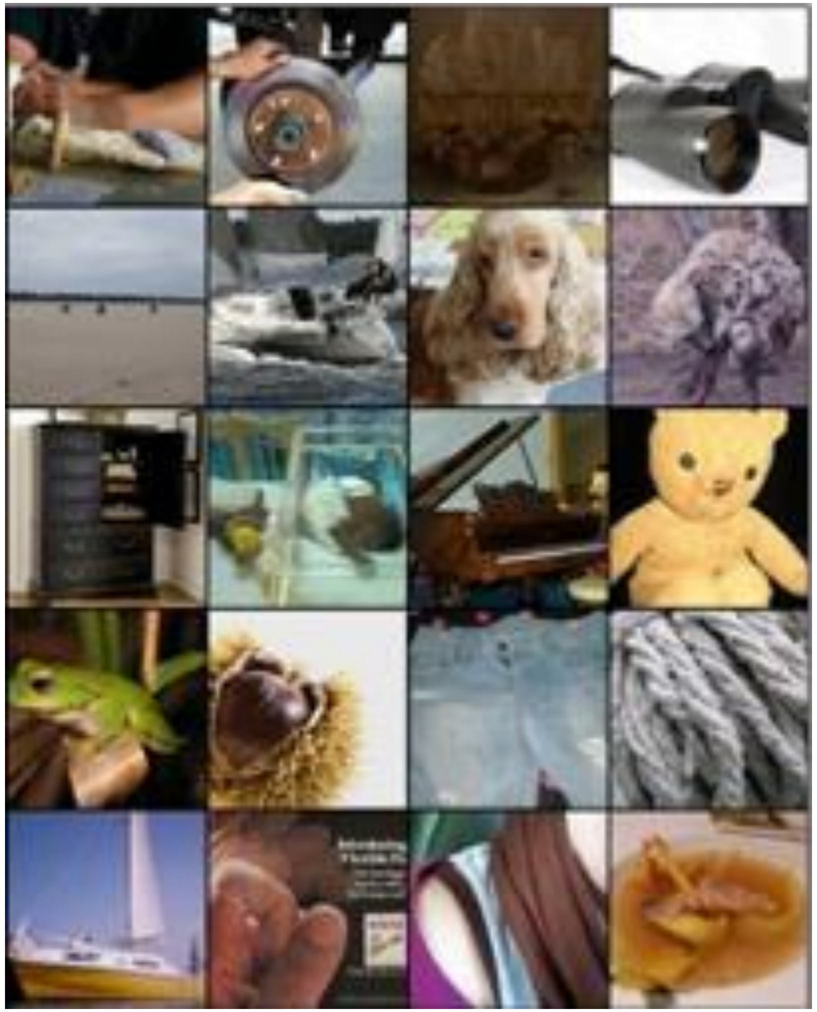
- For speech, image and videos one can respectively extract a 1D, 2D and 3D latent feature spaces

- 32 x 32 latents for ImageNet, or 8 x 8 x 10 for CIFAR10
- 512-dimensional latents for VCTK and LibriSpeech

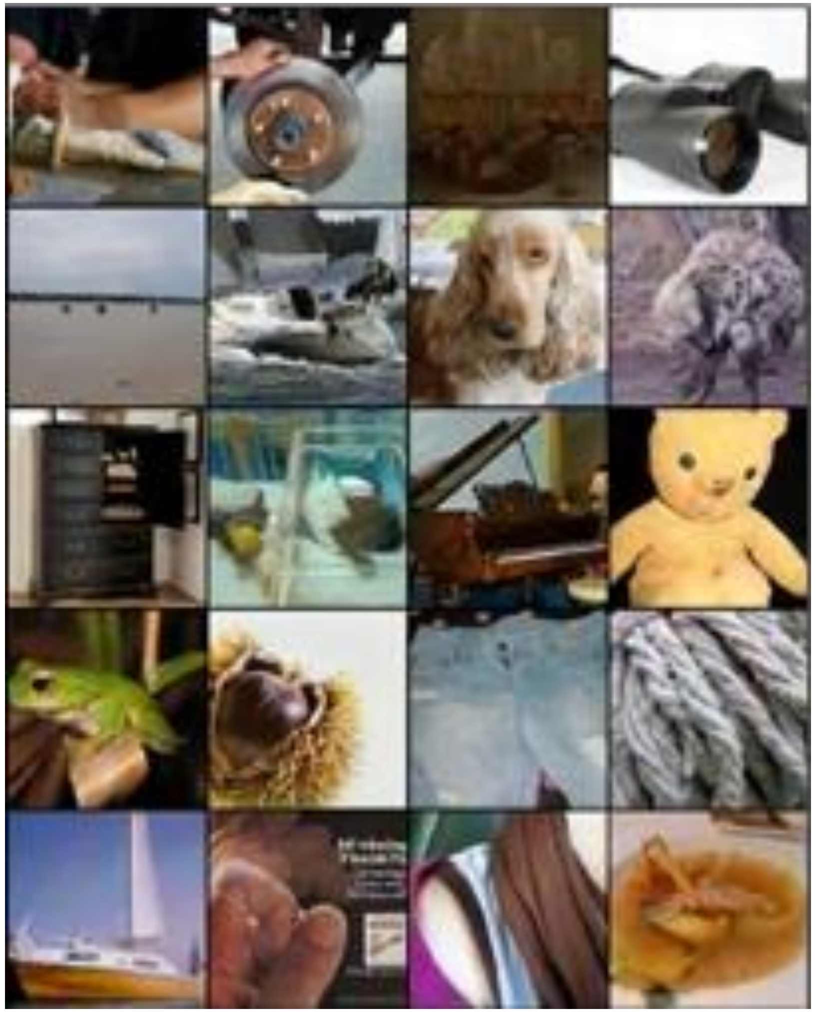


VQ-VAE ImageNet Reconstructions

Original 128 x 128 images

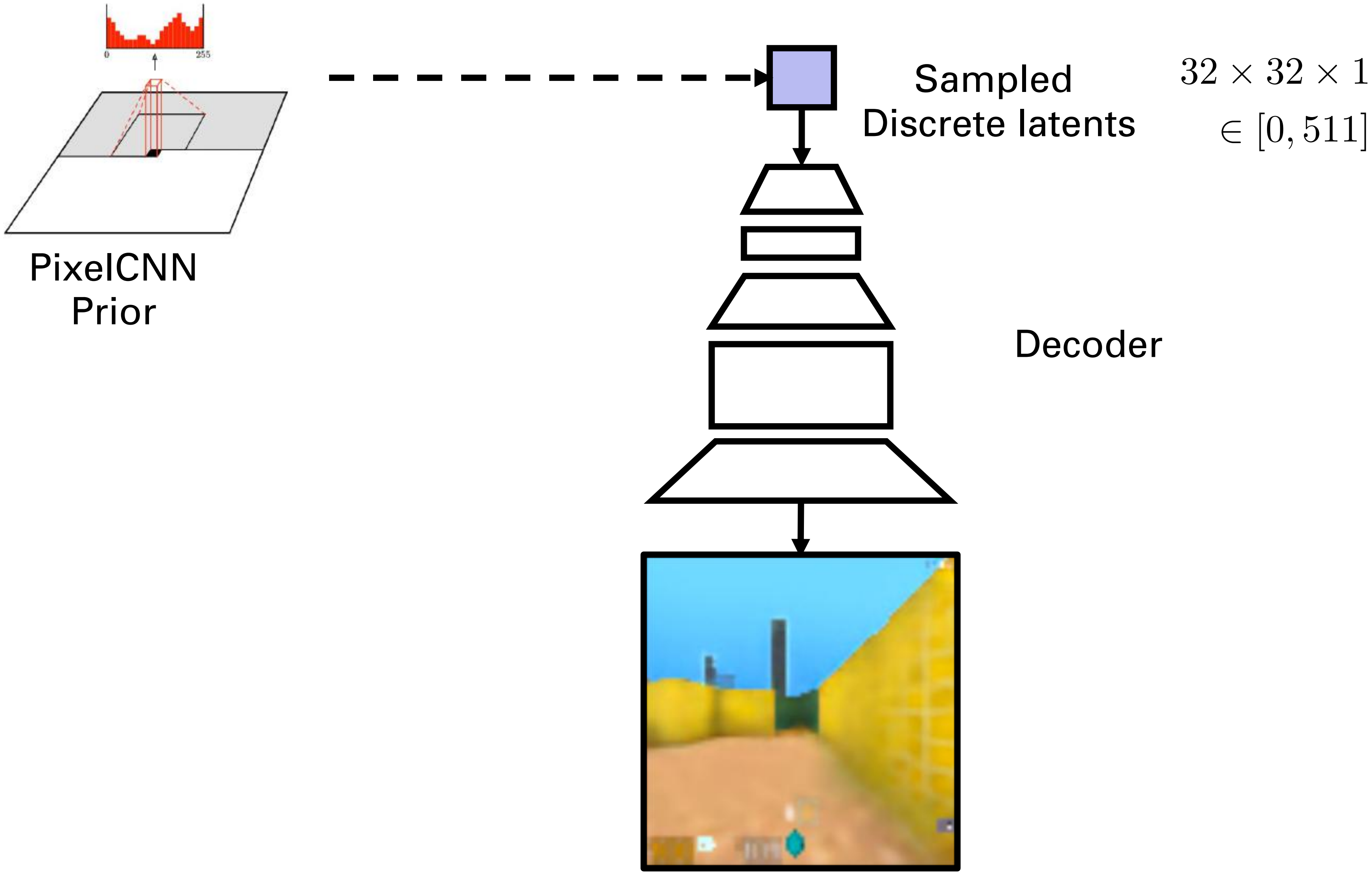


Reconstructions



Slide credit: Aäron van den Oord

VQ-VAE Sampling from prior



Slide credit: Aäron van den Oord

VQ-VAE ImageNet Samples

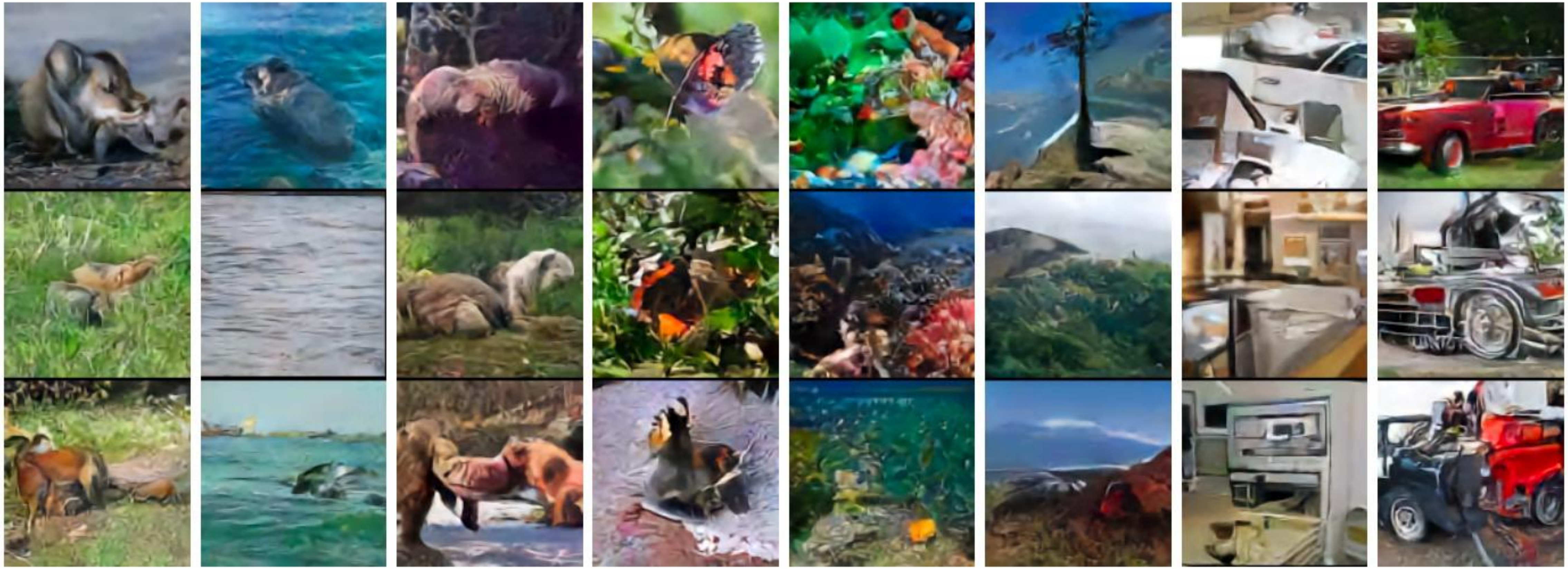
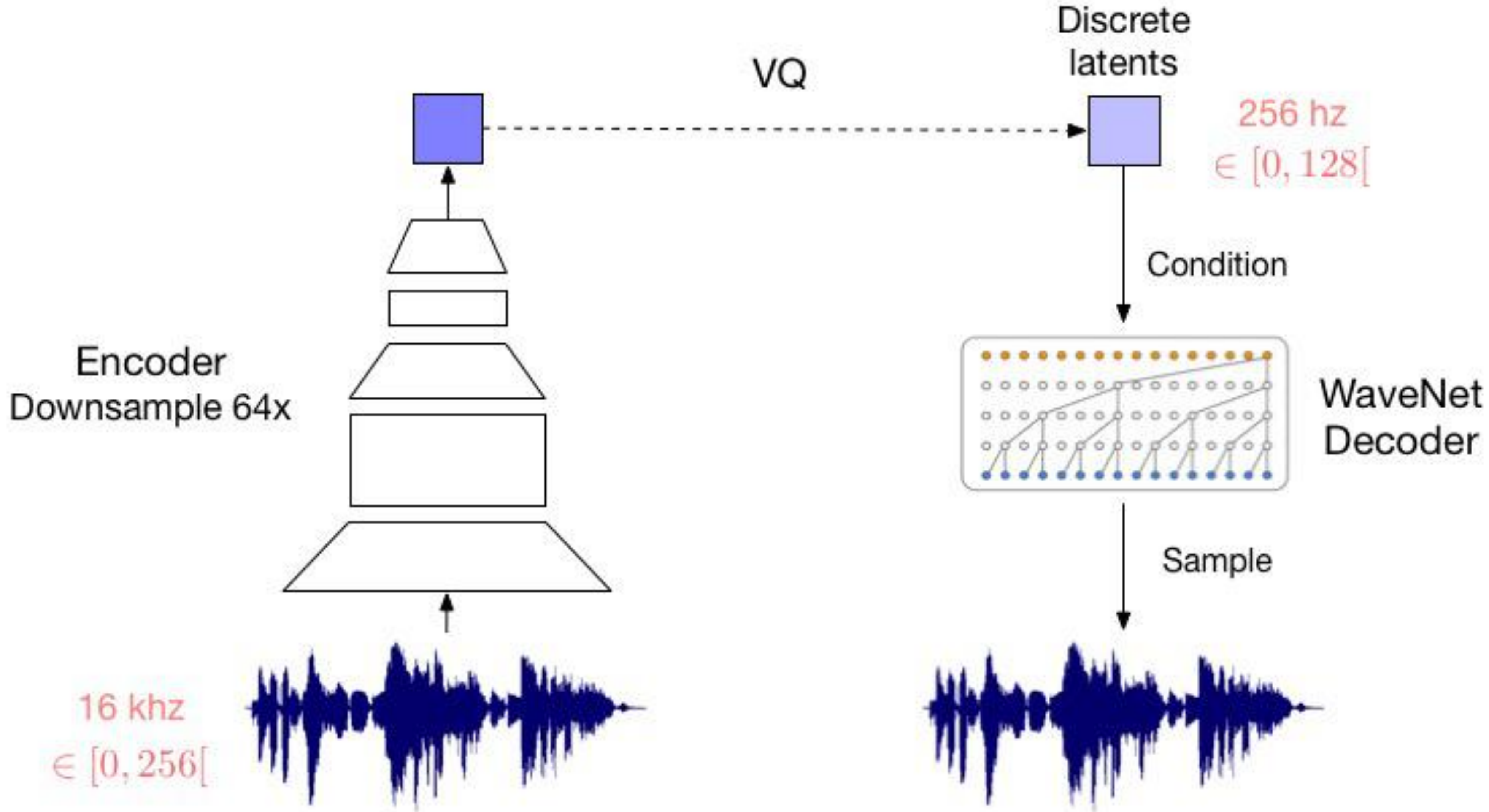
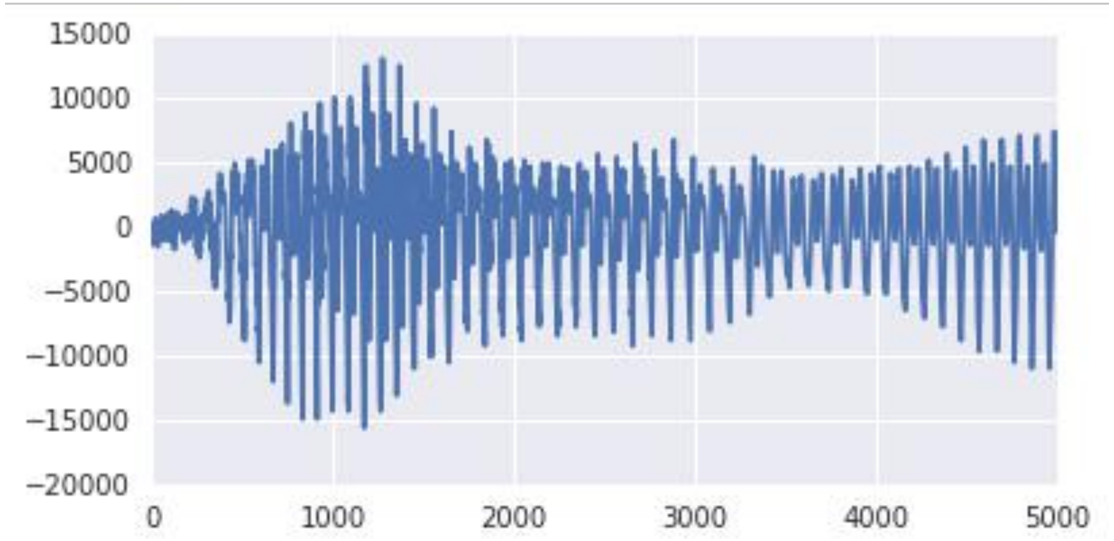


Figure 3: Samples (128x128) from a VQ-VAE with a PixelCNN prior trained on ImageNet images. From left to right: kit fox, gray whale, brown bear, admiral (butterfly), coral reef, alp, microwave, pickup.

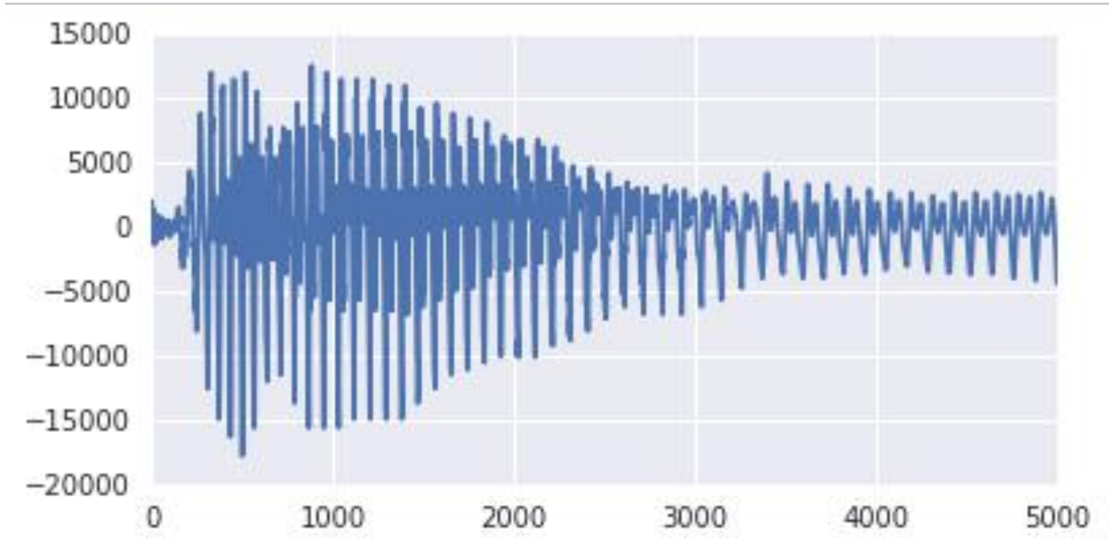
VQ-VQA Speech Modeling



VQ-VQA VCTK Reconstructions

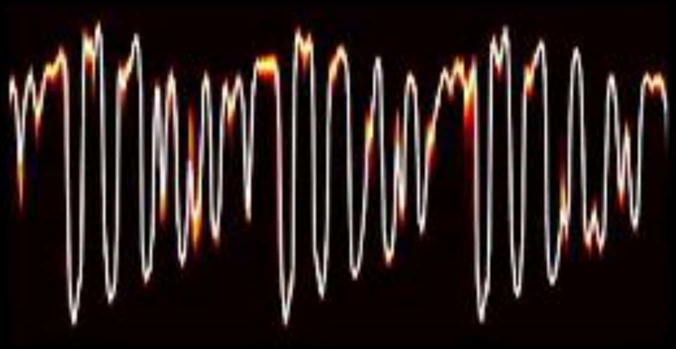


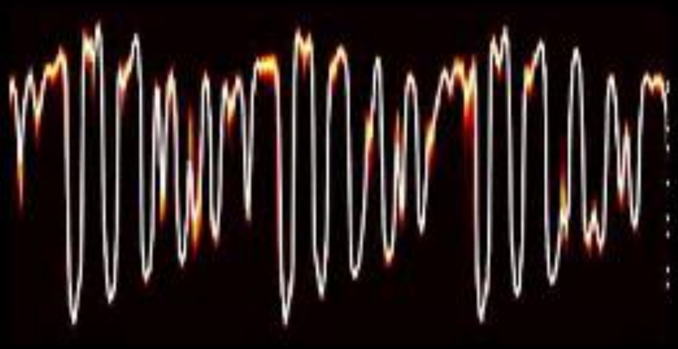
Original



Reconstruction

VQ-VAE Reconstructions





VQ-VAE Sampling from prior

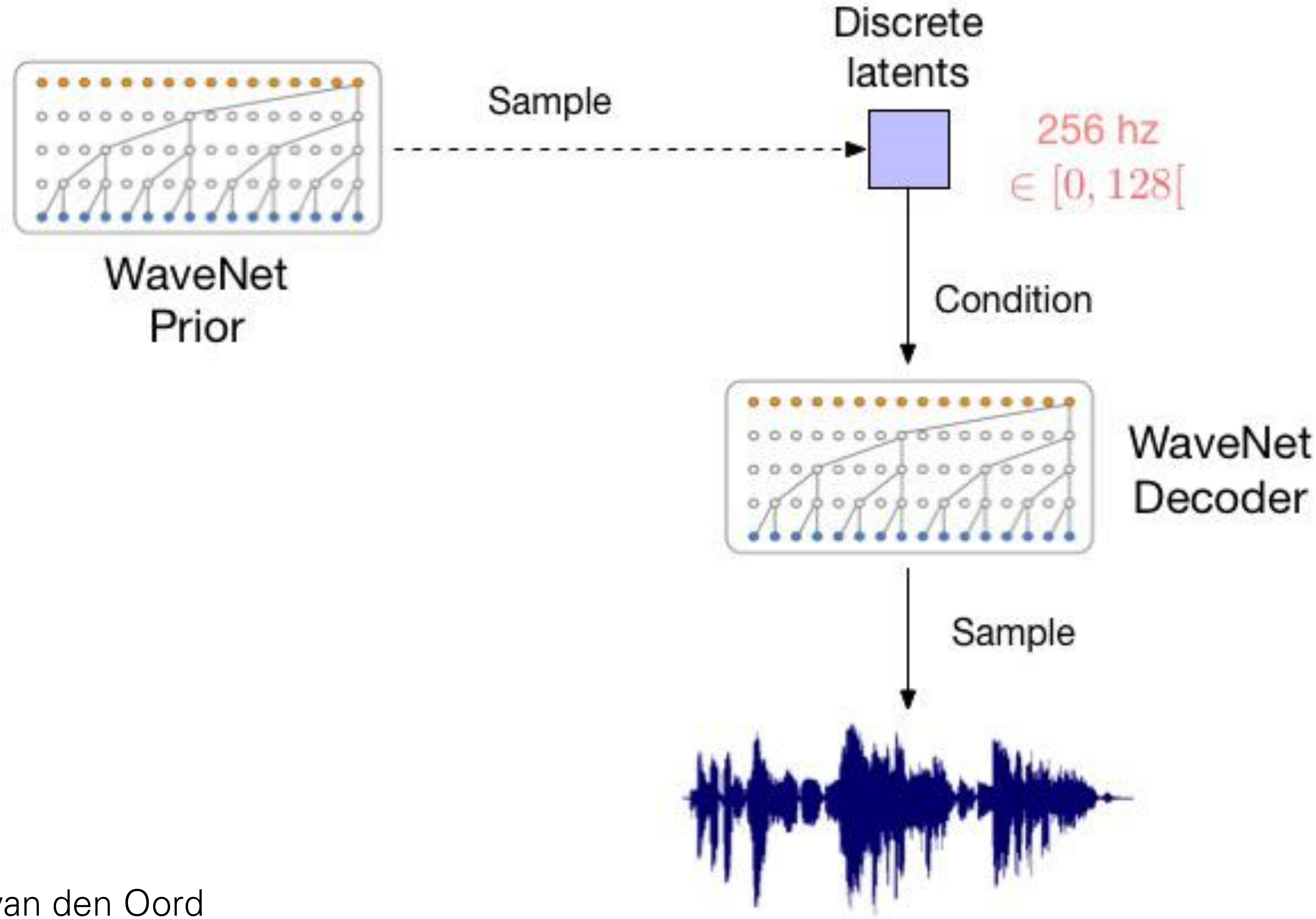
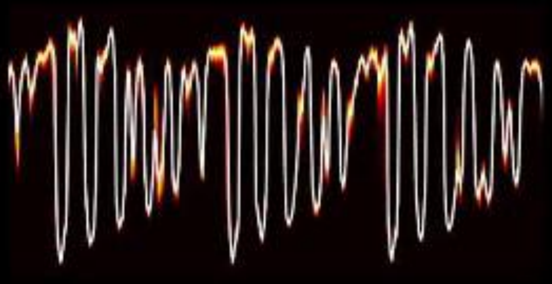
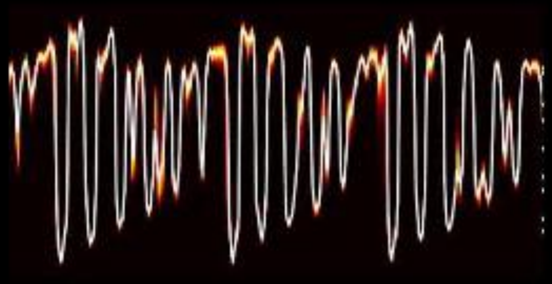
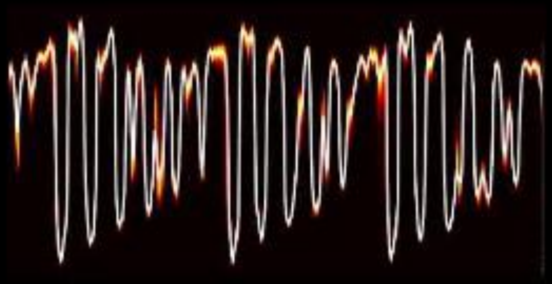


Image credit: Äëron van den Oord

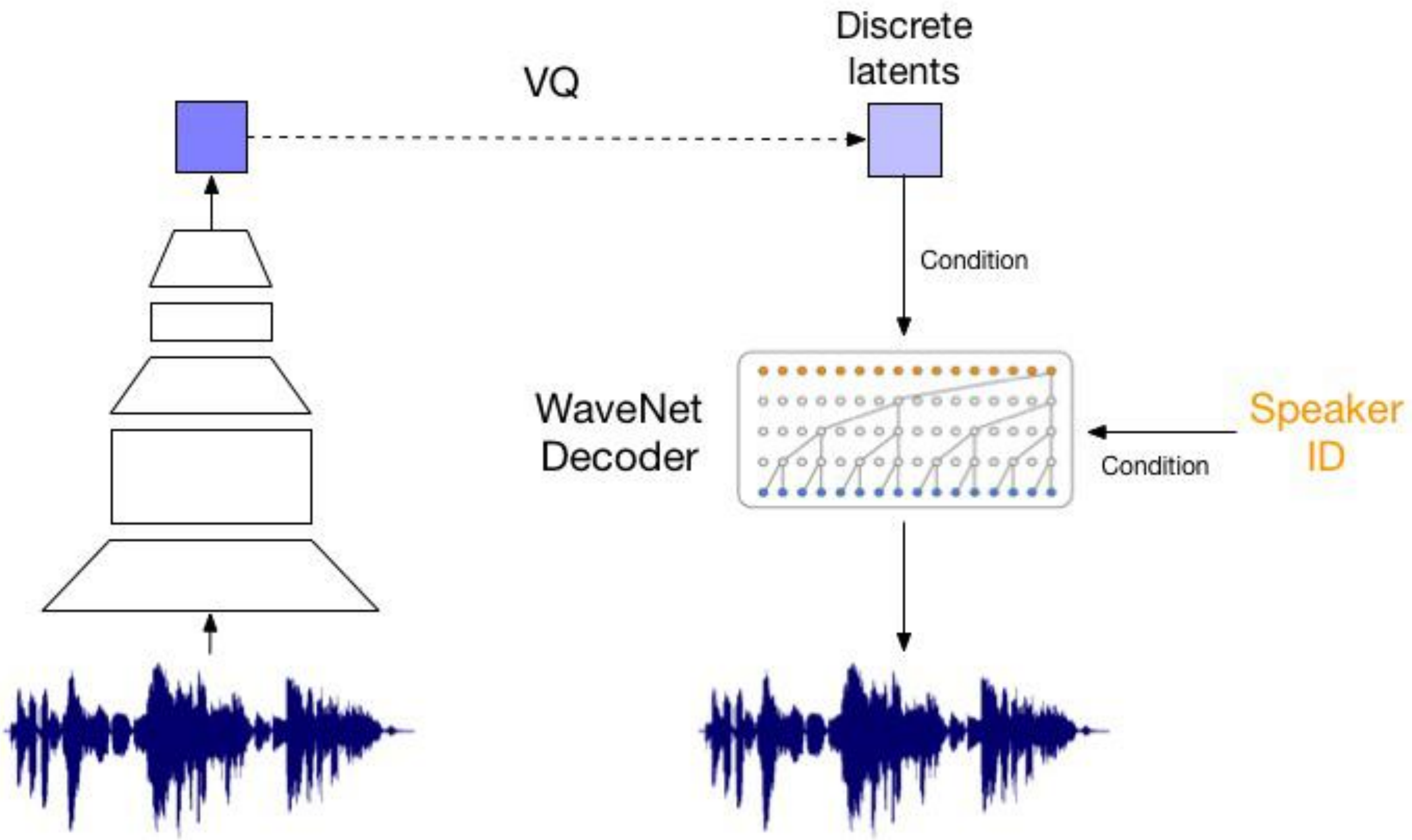
VQ-VAE Samples



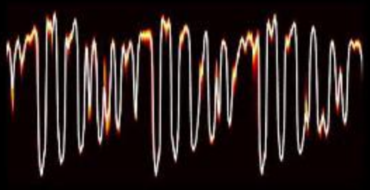


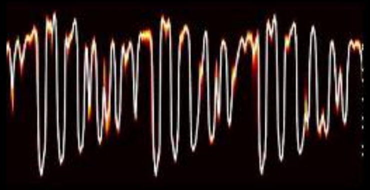


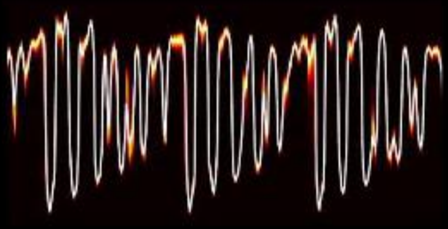
VQ-VAE Voice Style-Transfer

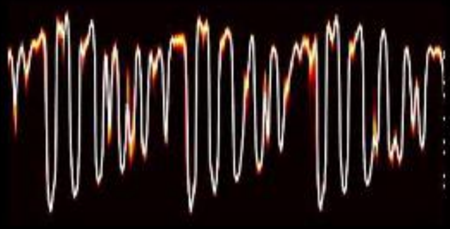


VQ-VAE Voice Style-Transfer Results

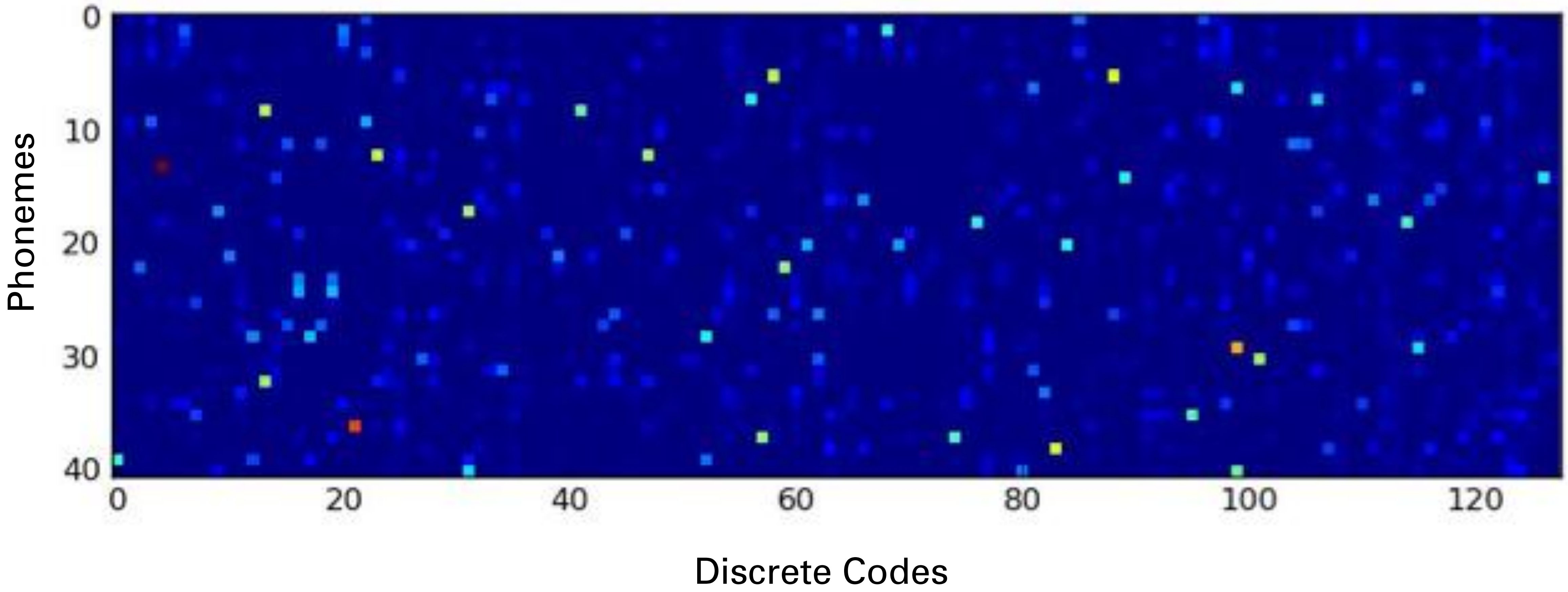








Unsupervised Learning of Phonemes

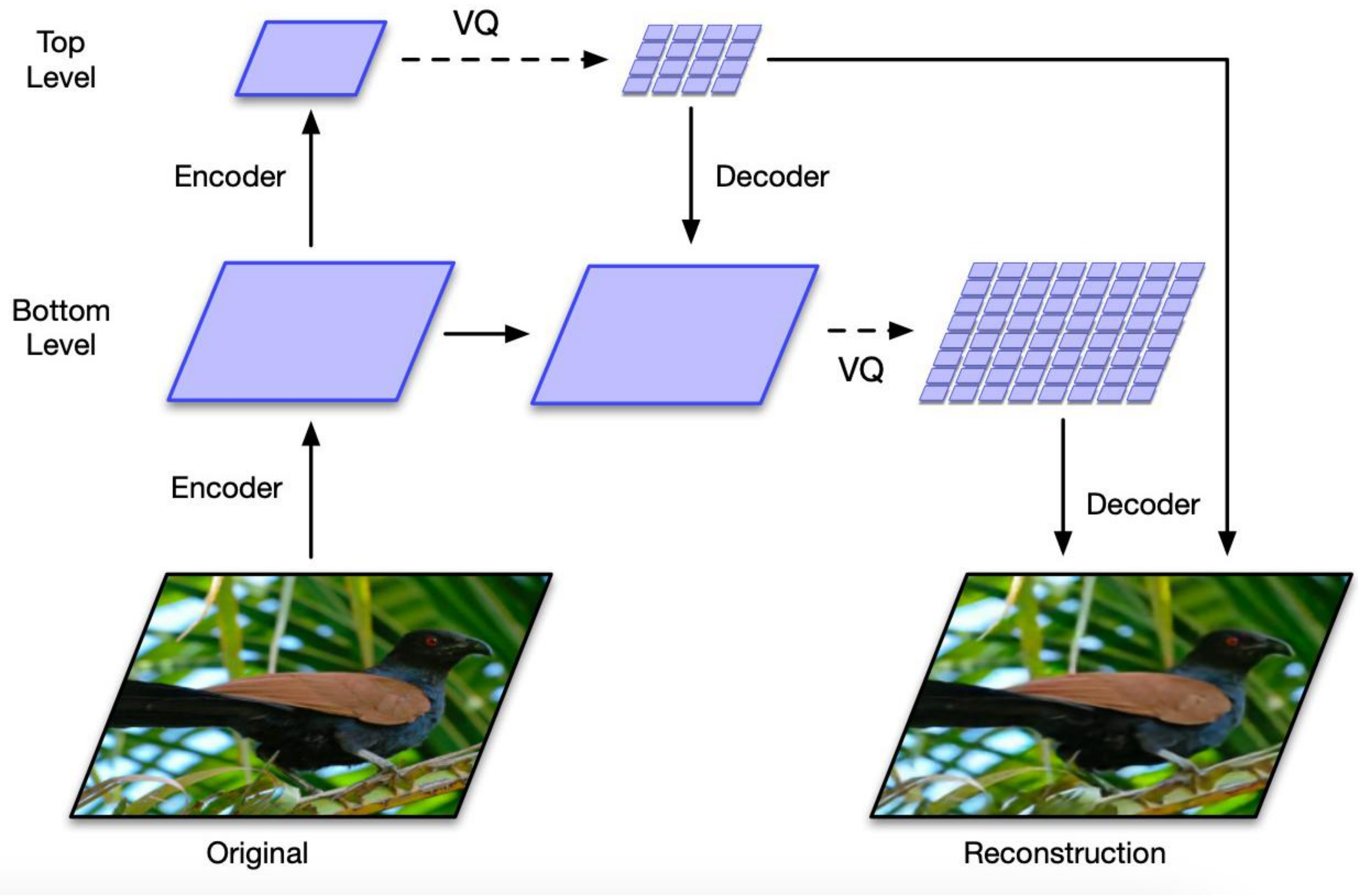


- 41-way classification
- 49.3% accuracy **fully unsupervised**

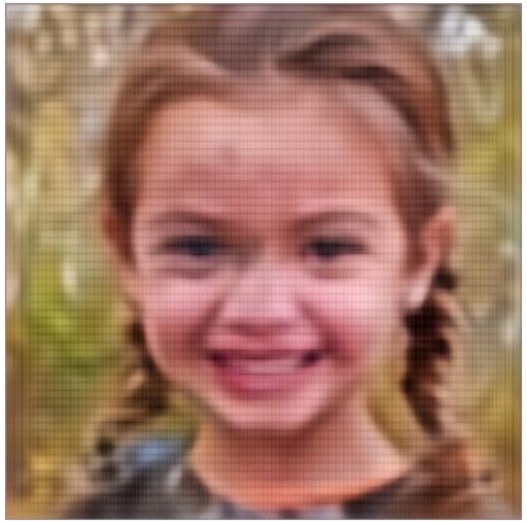
VQ-VAE-2

- The input 256×256 image is compressed to quantized latent maps of size 64×64 and 32×32

- The decoder reconstructs the image from the two latent maps.



VQ-VAE-2 Reconstructions



h_{top}

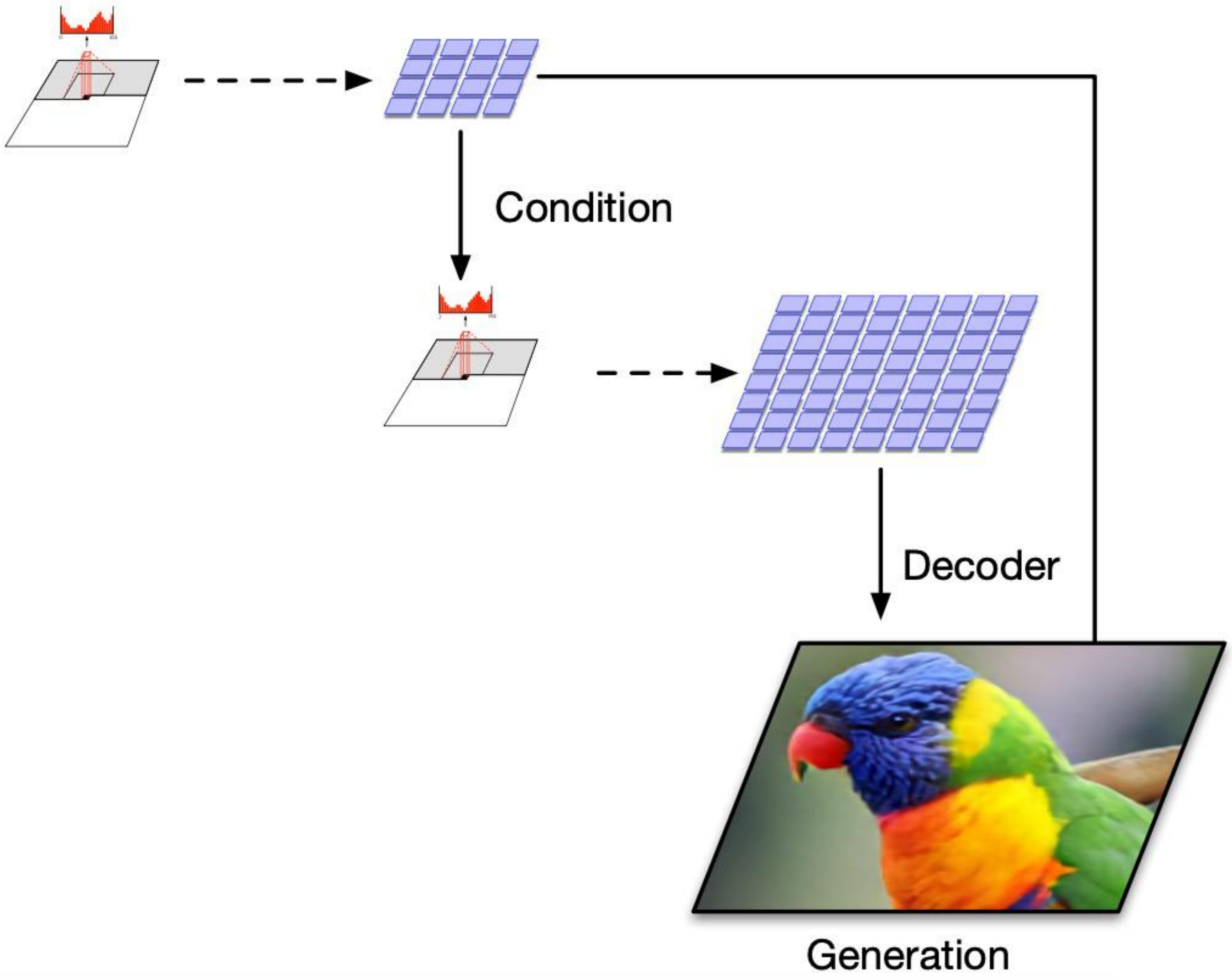
$h_{\text{top}}, h_{\text{middle}}$

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

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.

VQ-VAE-2



VQ-VAE-2 FFHQ-2014 Samples

VQ-VAE ImageNet Samples

VQ-VAE 2.0

BigGAN Deep

VQ-VAE ImageNet Samples

VQ-VAE 2.0

BigGAN Deep

VQ-VAE ImageNet Samples

VQ-VAE 2.0

BigGAN Deep

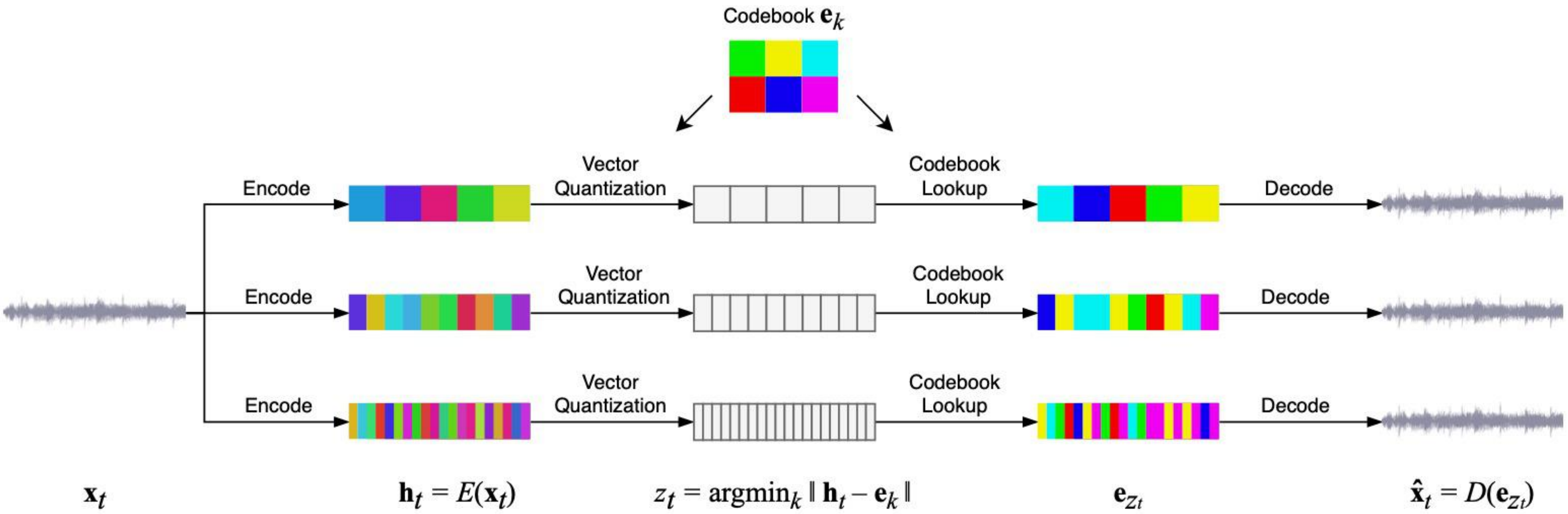
VQ-VAE-2 Quantitative Evaluation

- The VQ-VAE-2 paper reports a classification accuracy score (CAS) for class-conditional image generation.
- Generate image-class pairs from the generative model trained on the ImageNet training data.
- Train an image classifier from the generated pairs and measure its accuracy on the ImageNet test set.

	Top-1 Accuracy	Top-5 Accuracy
BigGAN deep	42.65	65.92
VQ-VAE	54.83	77.59
VQ-VAE after reconstructing	58.74	80.98
Real data	73.09	91.47

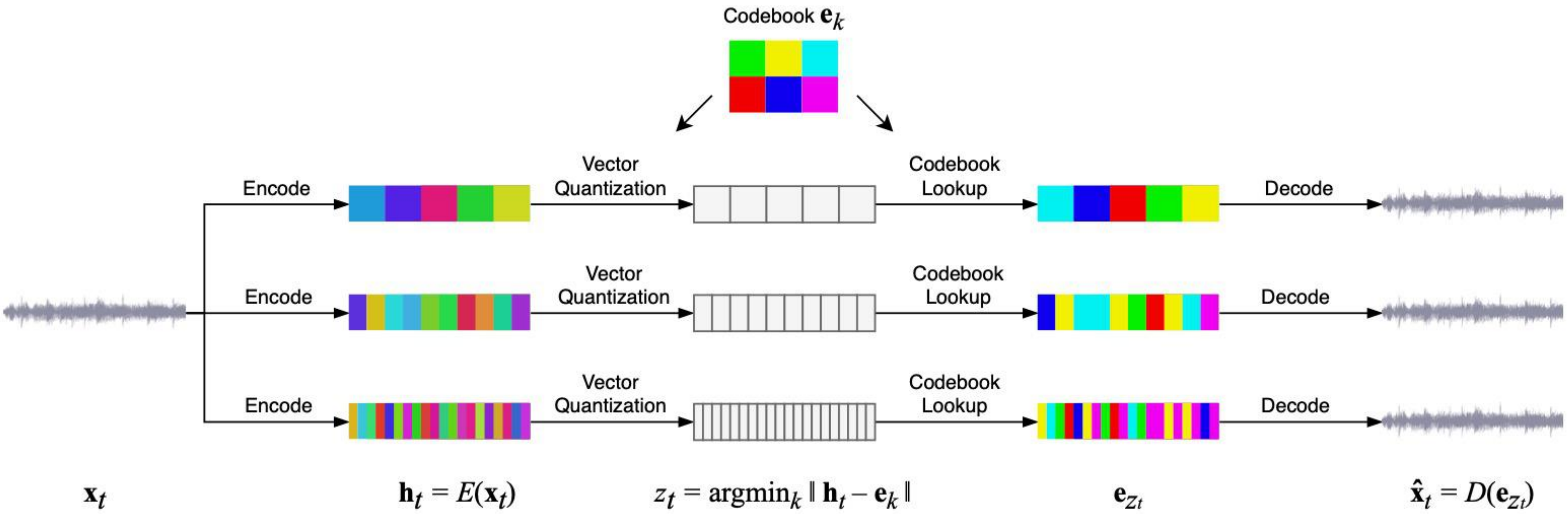
Slide credit: David McAllester

VQ-VAE-2 for Music Generation



- Three separate VQ-VAE models with different temporal resolutions

VQ-VAE-2 for Music Generation



- Three separate VQ-VAE models with different temporal resolutions

OpenAI

API PROJECTS BLOG ABOUT

Jukebox

We're introducing Jukebox, a neural net that generates music, including rudimentary singing, as raw audio in a variety of genres and artist styles. We're releasing the model weights and code, along with a tool to explore the generated samples.

READ PAPER VIEW CODE

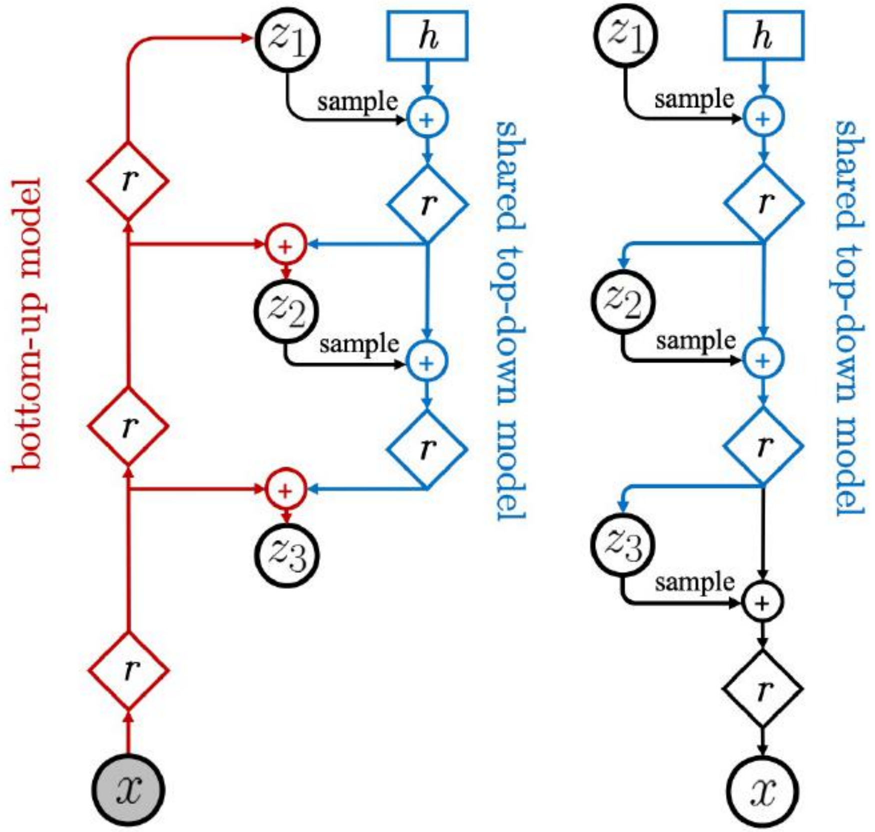
https://openai.com/blog/jukebox/

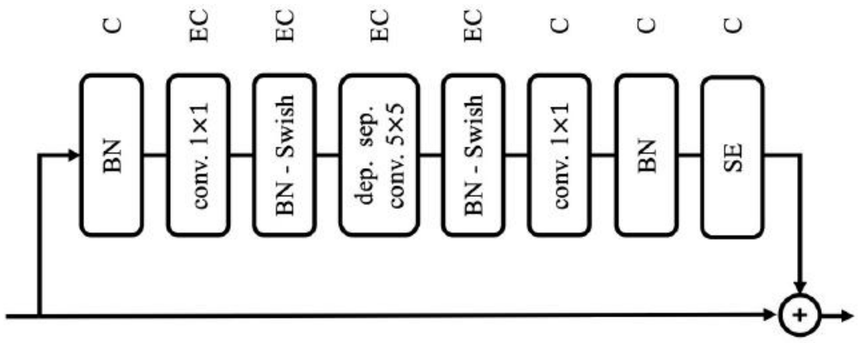
April 30, 2020
12 minute read, 10 day listen

Lecture overview

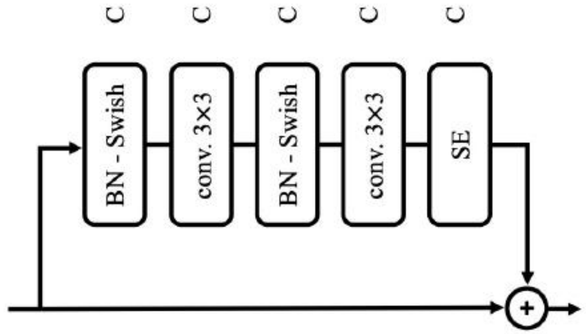
- Motivation
- Training Latent Variable Models (including VAE and IWAE)
- Variations
  - SOTA: VQ-VAE, VQ-VAE 2.0, NVAE
  - AR + VAE: Variational Lossy AutoEncoder, PixelVAE
  - Disentanglement: Beta VAE
- Related ideas

NVAE





(a) Residual Cell for NVAE Generative Model



(b) Residual Cell for NVAE Encoder

$$\mathcal{L}_{\text{VAE}}(\boldsymbol{x}) := \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \text{KL}(q(\boldsymbol{z}_1|\boldsymbol{x})||p(\boldsymbol{z}_1)) - \sum_{l=2}^{L} \mathbb{E}_{q(\boldsymbol{z}_{<l}|\boldsymbol{x})}\left[\text{KL}(q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l})||p(\boldsymbol{z}_l|\boldsymbol{z}_{<l}))\right]$$

$$q(\boldsymbol{z}_{<l}|\boldsymbol{x}) := \prod_{i=1}^{l-1} q(\boldsymbol{z}_i|\boldsymbol{x}, \boldsymbol{z}_{<i})$$

NVAE – Experiments

Lecture overview

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
- **Variations**
  - SOTA: VQ-VAE, VQ-VAE 2.0, NVAE
  - **AR + VAE: Variational Lossy AutoEncoder, PixelVAE**
  - Disentanglement: Beta VAE
- Related ideas

Decoder distribution

- So far all models use simple distribution for $p(x|z)$
- Due to lack of expressivity itself, all entropy is pushed to z and z needs to convey a lot of information

[Ho et al., 2019]

Powerful decoder

- What's the maximum VLB?

$$\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[VLB\right] \leq \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log p_\theta(x)\right]$$
$$\leq \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log p_{\text{data}}(x)\right]$$

- What if $p(x|z) = p_{data}(x)$?

$$\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[VLB\right] = \mathbb{E}_{x\sim p_{\text{data}}(x), z\sim q(z|x)}\left[\log p(x|z) + \log p(z) - \log q(z|x)\right]$$
$$= \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log p_{\text{data}}(x) + \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log p(z) - \log q(z|x)\right]\right]$$

  - $q(z|x)$ would be set to $p(z)$ → z has no information

Powerful decoder

- Having information in z incurs VLB penalty of $KL(q||p)$ which is usually non-zero
- "Ignoring latent code" problems well documented in literature
  - (Fabius & van Amersfoort,2014; Chung et al., 2015; Bowman et al., 2015; Serban et al., 2016; Fraccaro et al., 2016; Xu &Sun, 2016)
  - Many proposed solutions

Weakening models

- Adding dropout in autoregressive conditioning (Bowman et al., 2015)
- PixelCNN with limited receptive field (Chen et al., 2016)
- Constant bit rate. $D_{KL}(q_\phi(z|x)||p_\theta(z)) = c$ (Guu et al., 2017), (Xu & Durrett, 2018), (Davidson et al., 2018)
- Minimum bit rate $D_{KL}(q_\phi(z|x)||p_\theta(z)) \geq \delta$ (Razavi et al., 2019)

Changing training dynamics

- $D_{KL}(q_\phi(z|x)||p_\theta(z))$ warmup (Bowman et al., 2015); (Yang et al., 2017); (Kim et al., 2018); (Gulrajani et al., 2016)
- "Free-bits" (Kingma et al., 2016); (Chen et al., 2016)
- More training updates to q(z|x) (He et al., 2019)

Lecture overview

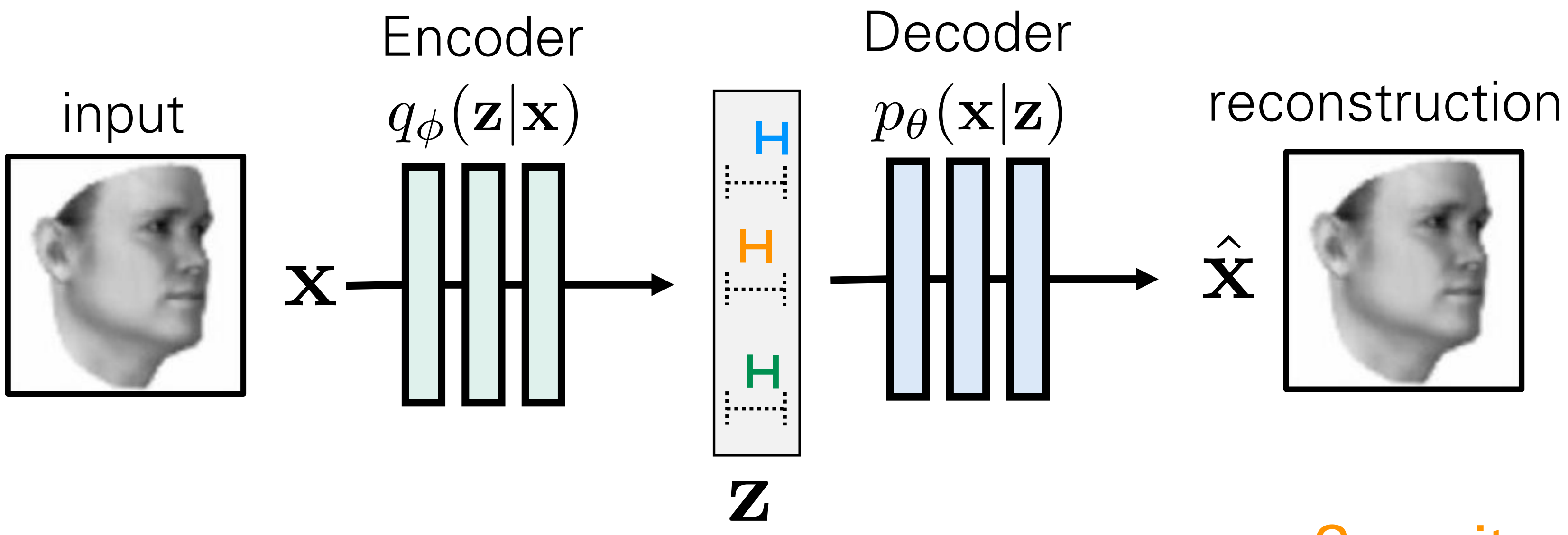
- Motivation
- Training Latent Variable Models (including VAE and IWAE)
- **Variations**
  - SOTA: VQ-VAE, VQ-VAE 2.0, NVAE
  - AR + VAE: Variational Lossy AutoEncoder, PixelVAE
  - **Disentanglement: β-VAE**
- Related ideas

β-VAE

The Beta-VAE objective is identical to the VAE objective when beta = 1

$$\mathcal{L} = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \beta D_{KL}\left(q_\phi(z \mid x) \| p(z)\right)$$

Disentangled Representations with β-VAE



Capacity constraint

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

Reconstruction cost

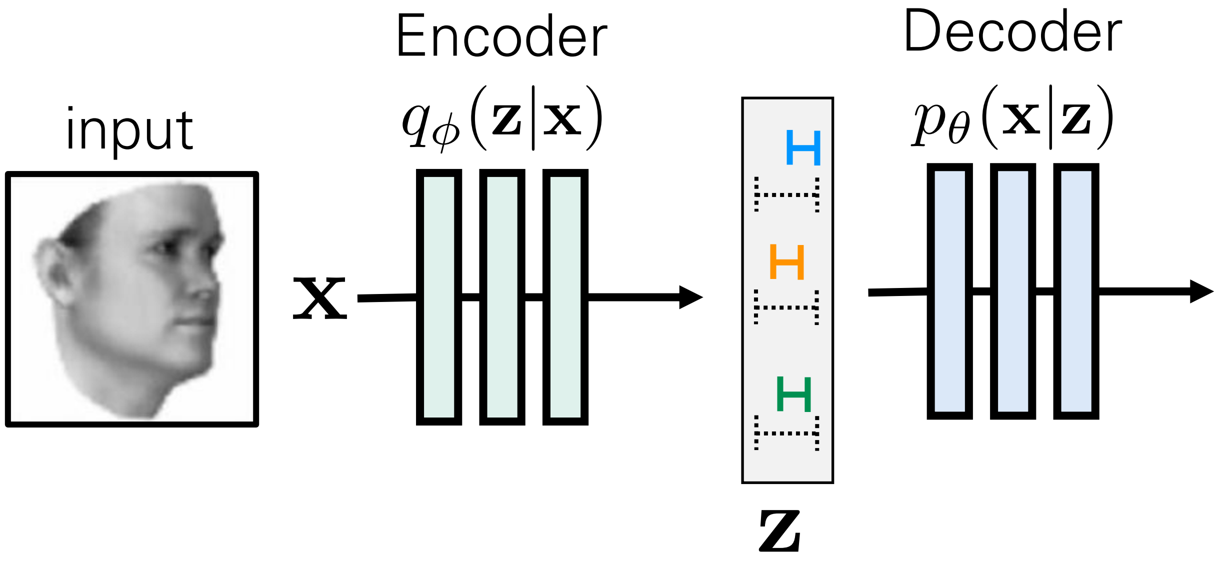
(Higgins et al., 2017)

Slide credit: Loic Matthey, Irina Higgins

Disentangled Representations with β-VAE

- What do individual features learn?

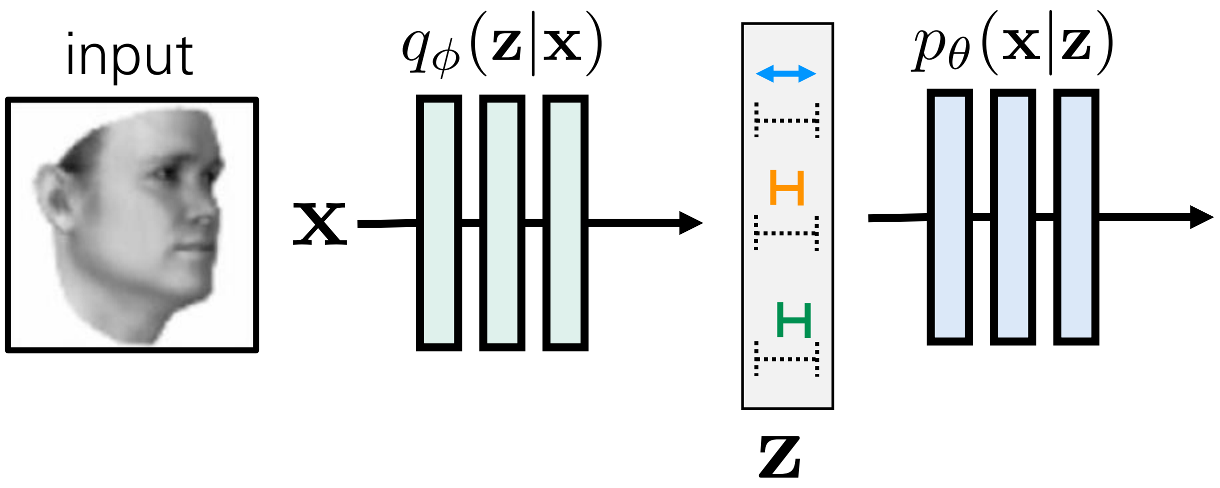
Latent traversal



Disentangled Representations with β-VAE

- What do individual features learn?

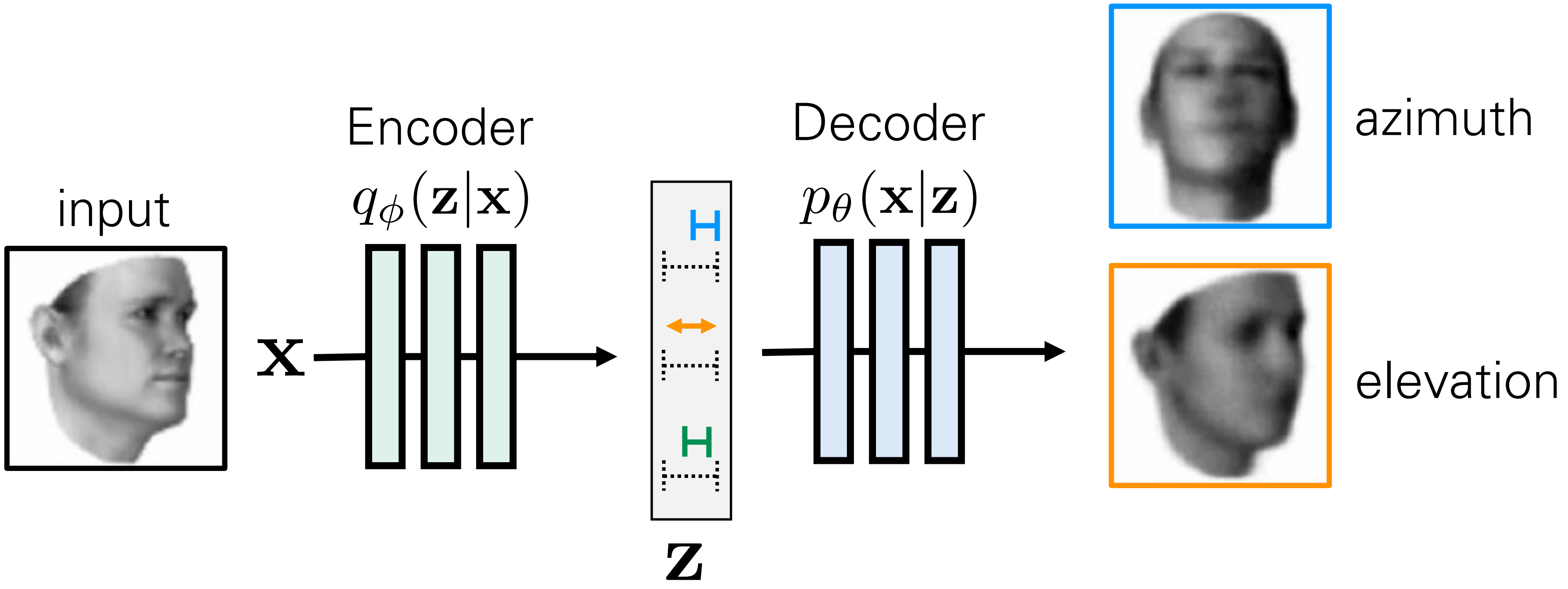
Latent traversal



Disentangled Representations with β-VAE

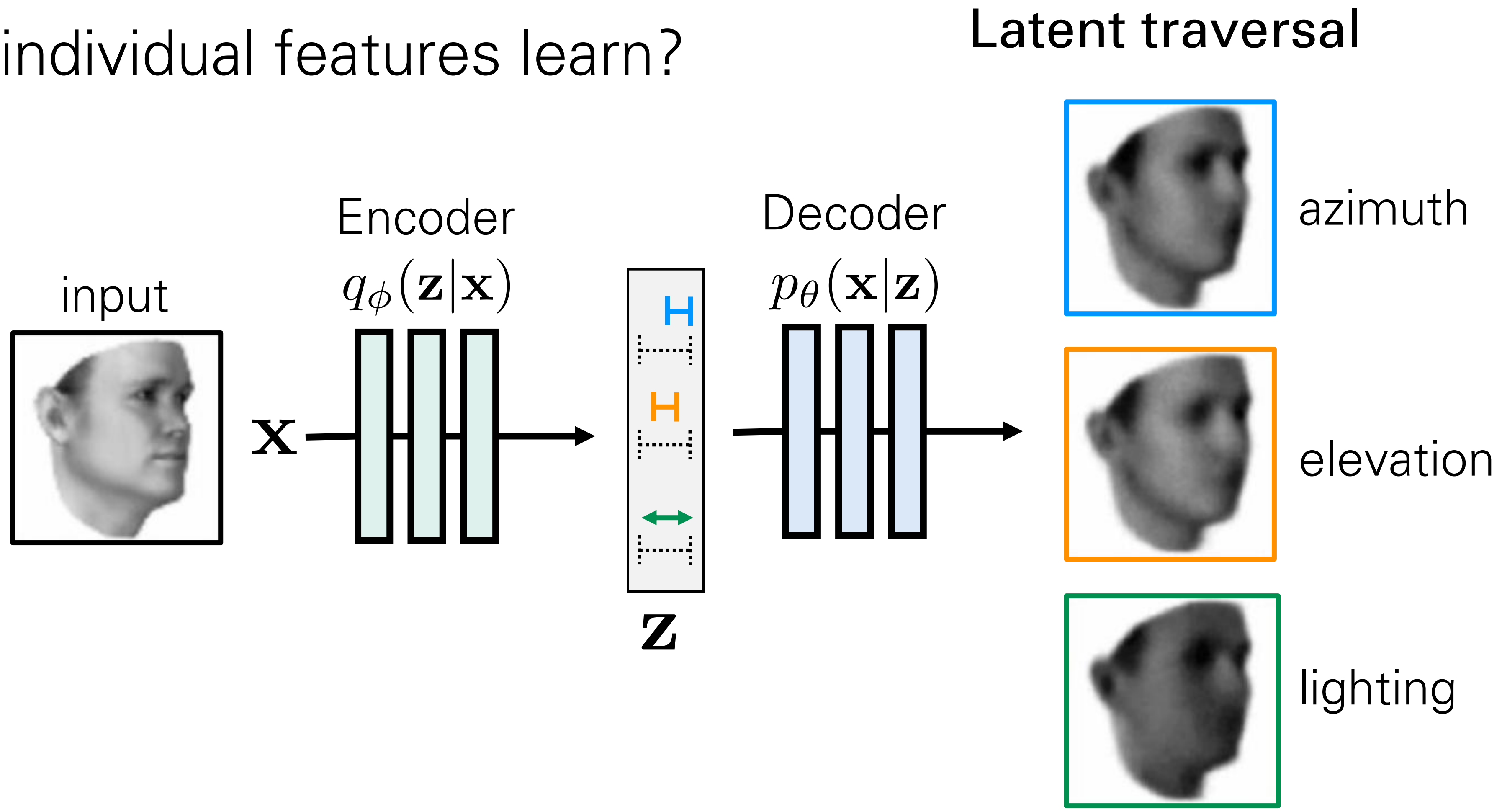
- What do individual features learn?

Latent traversal



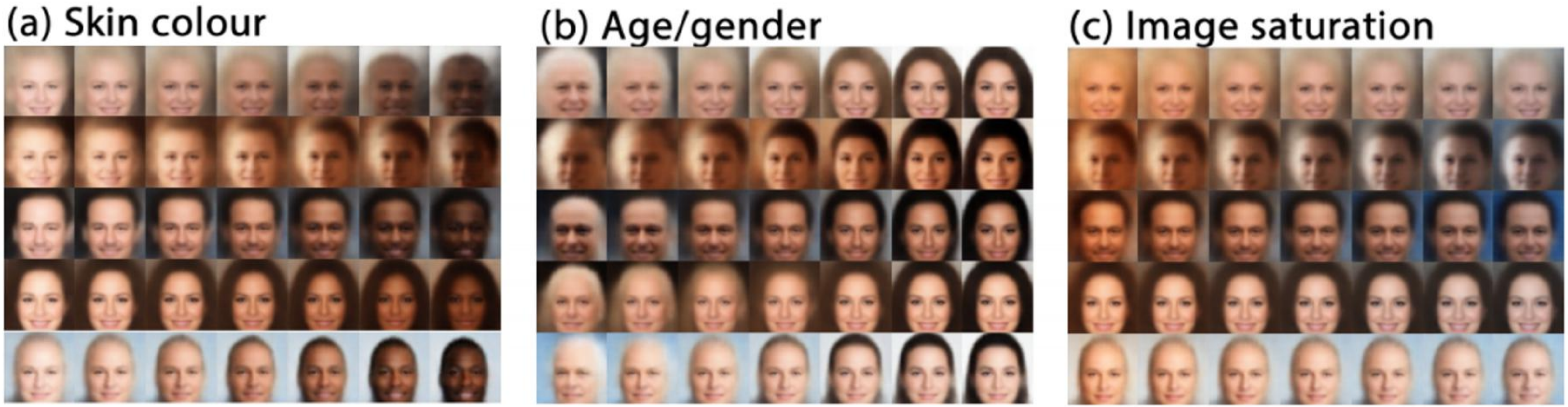
Disentangled Representations with β-VAE

- What do individual features learn?

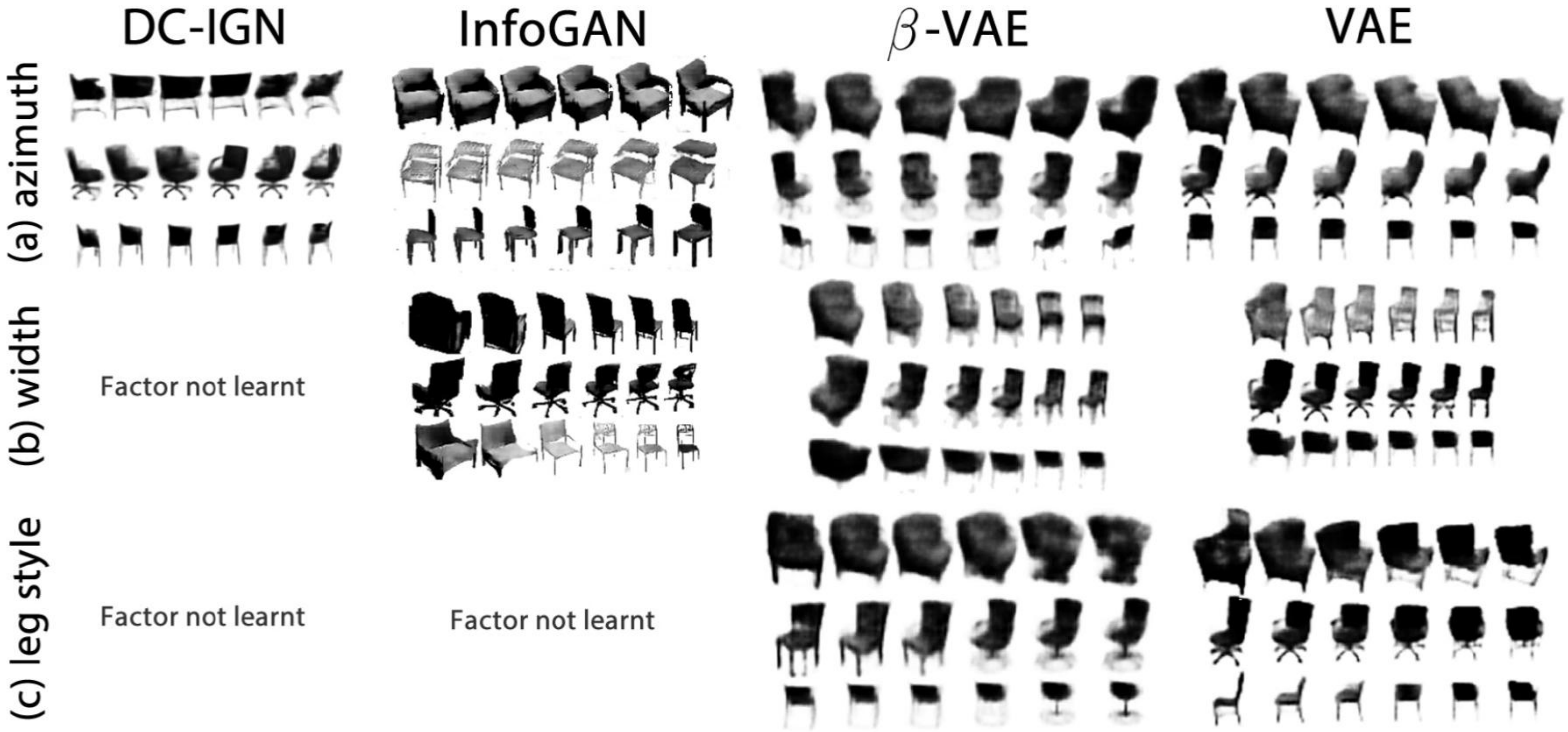


Slide credit: Loic Matthey, Irina Higgins

β-VAE – Experiments



β-VAE – Experiments



Lecture overview

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
- Variations
- **Related ideas**
  - Variational Dequantization (flow++)

Recap: Uniform Dequantization

- **Uniform Dequantization.** Add noise to data.

  $$\mathbf{x} \in \{0, 1, 2, \ldots, 255\}$$

  – We draw noise u uniformly from $[0, 1)^D$

$$\begin{aligned}
\mathbb{E}_{\mathbf{y} \sim p_{\text{data}}} [\log p_{\text{model}}(\mathbf{y})] &= \sum_{\mathbf{x}} P_{\text{data}}(\mathbf{x}) \int_{[0,1)^D} \log p_{\text{model}}(\mathbf{x} + \mathbf{u}) \, d\mathbf{u} \\
&\leq \sum_{\mathbf{x}} P_{\text{data}}(\mathbf{x}) \log \int_{[0,1)^D} p_{\text{model}}(\mathbf{x} + \mathbf{u}) \, d\mathbf{u} \\
&= \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}} [\log P_{\text{model}}(\mathbf{x})]
\end{aligned}$$

[Theis, Oord, Bethge, 2016]

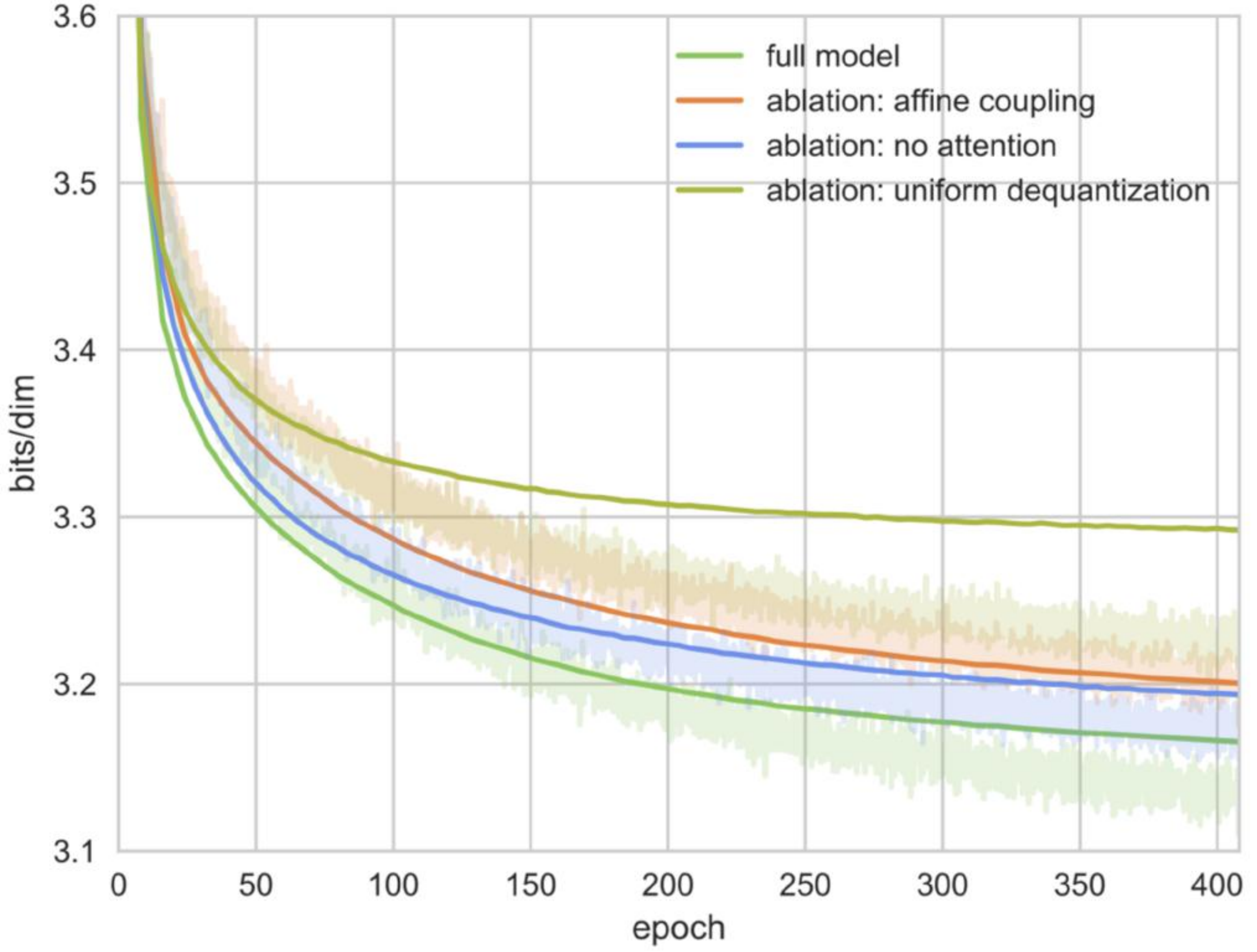
Variational Dequantization

- **Variational Dequantization**. Add a learnable noise q to data.

$$
\begin{aligned}
\mathbb{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_{\text{model}}(\mathbf{x})\right] &= \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log \int_{[0,1)^D} q(\mathbf{u}|\mathbf{x})\frac{p_{\text{model}}(\mathbf{x}+\mathbf{u})}{q(\mathbf{u}|\mathbf{x})}\,d\mathbf{u}\right] \\
&\geq \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\int_{[0,1)^D} q(\mathbf{u}|\mathbf{x})\log\frac{p_{\text{model}}(\mathbf{x}+\mathbf{u})}{q(\mathbf{u}|\mathbf{x})}\,d\mathbf{u}\right] \\
&= \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}}\mathbb{E}_{\mathbf{u}\sim q(\cdot|\mathbf{x})}\left[\log\frac{p_{\text{model}}(\mathbf{x}+\mathbf{u})}{q(\mathbf{u}|\mathbf{x})}\right]
\end{aligned}
$$

[Ho et al., 2019]

Variational Dequantization on CIFAR



[Ho et al., 2019]

Next lecture:
Generative Adversarial
Networks