香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

COMP 5212
Machine Learning
Lecture 21

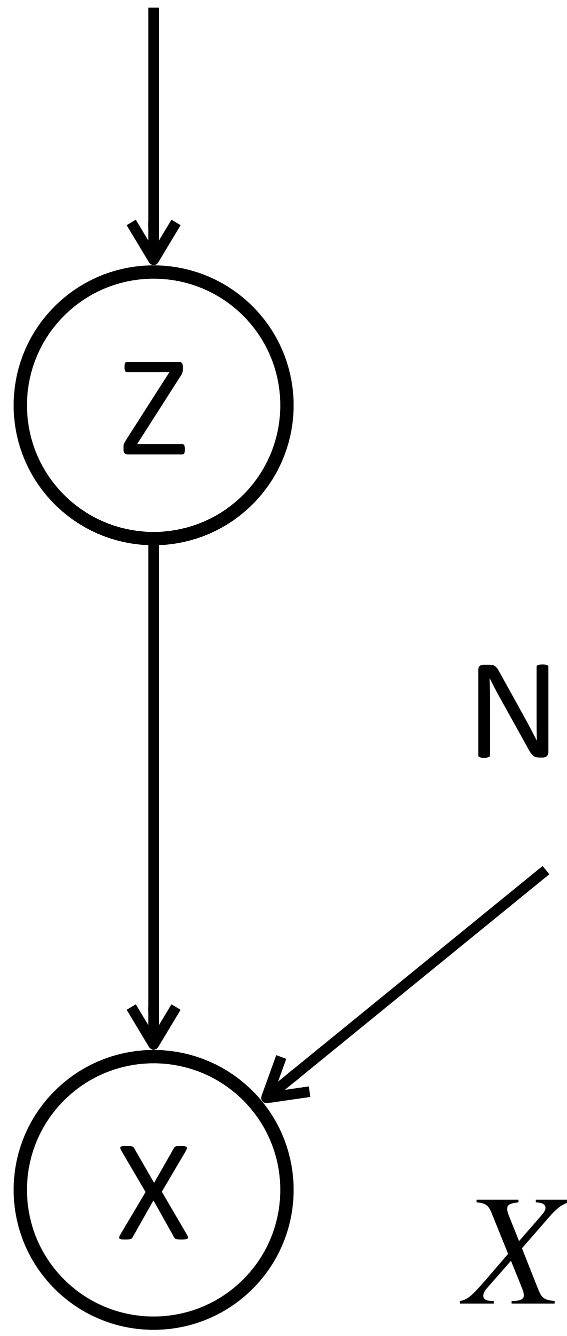
# Variational Autoencoders

Junxian He
Apr 24, 2024

# Recap: VAE is a Generative Model

p(z) is a normal distribution in most cases

p(z)




Neural Networks

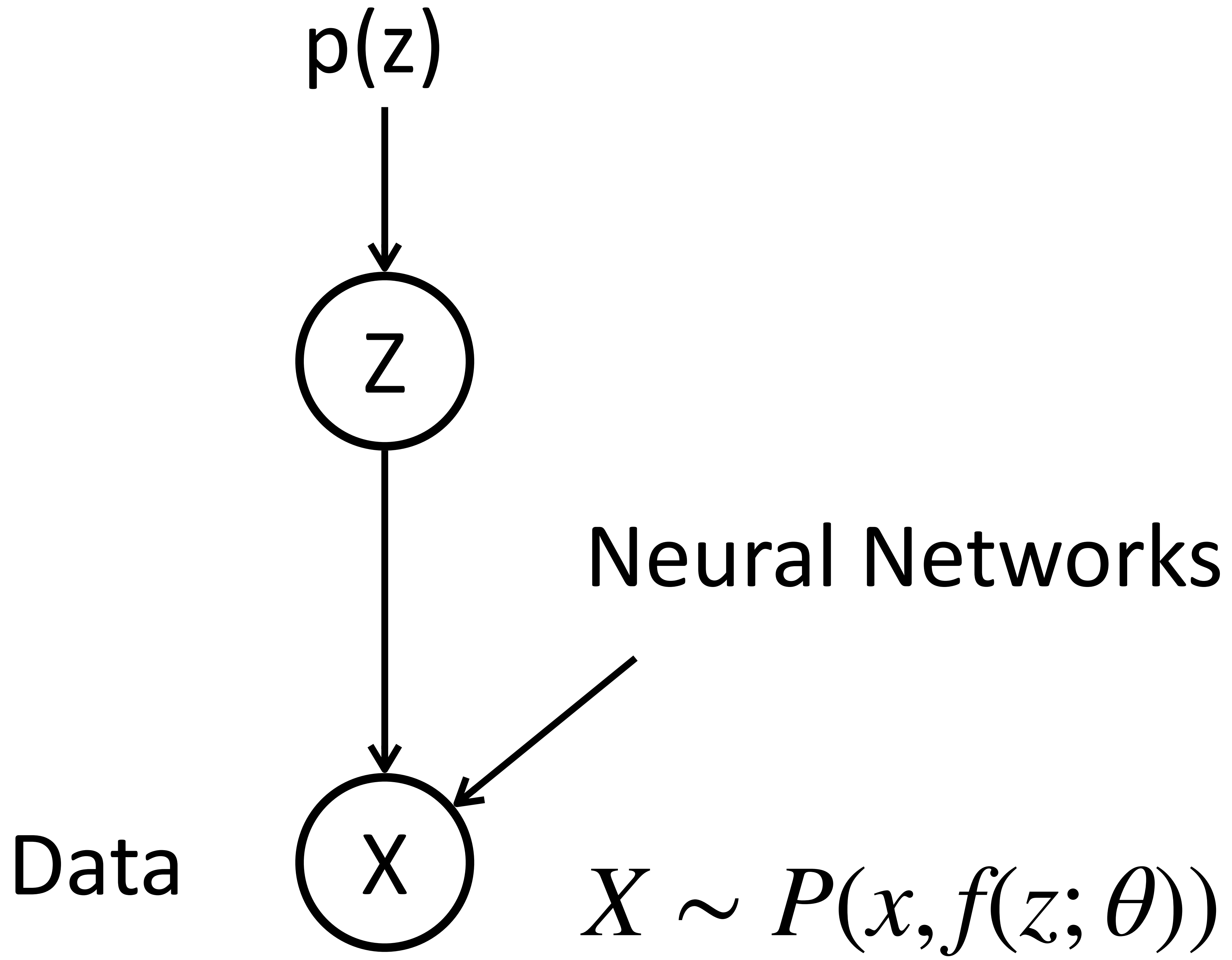
Data

$$X \sim P(x, f(z;\theta))$$

$f$ is a neural network taking Z as input

This graphical representation is similar to GMM

# Recap: Training

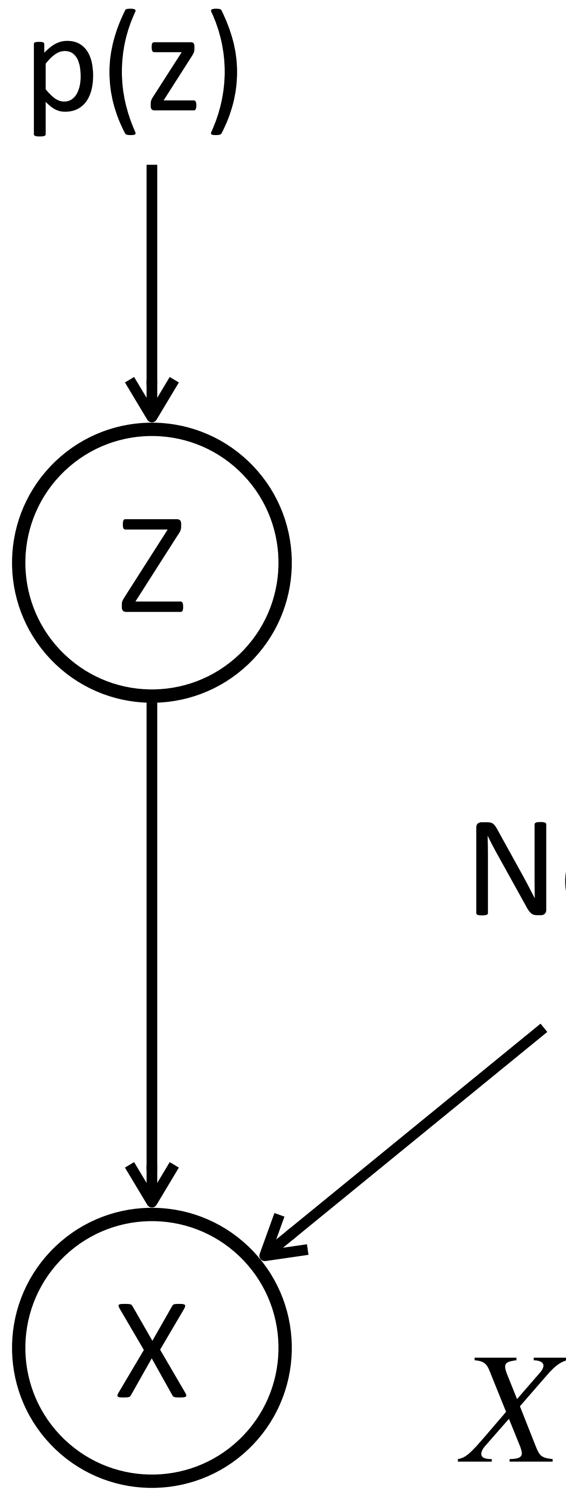



$$X \sim P(x, f(z;\theta))$$

How to train the model? Can we do MLE?

Intractable P(X), EM algorithm?

# Recap: Let's try EM




Neural Networks

$$X \sim P(x, f(z;\theta))$$

E-Step: compute P(z|x)

$$Q(z) = P(z|x) \propto P(z)P(x|z)$$

M-Step: the ELBO objective

$$\text{argmax}_\theta \sum_z Q(z) \log p(x, z; \theta) = \text{argmax}_\theta \mathbb{E}_{z \sim Q(z)} \log p(x, z; \theta)$$

In most cases, we cannot do the sum, and cannot easily sample from Q(z) either

# Recap: Approximate Posterior

We need an easy-to-sample distribution to approximate P(z|x)

$q(z \mid x; \phi)$ to approximate $p(z \mid x; \theta)$

$\phi$ is the parameter for the approximate function, $\theta$ is the generative model parameter

How to train $q(z \mid x; \phi)$, what would be the loss to find $\phi$?

# Recap: ELBO

$$\mathrm{ELBO}(x; Q, \theta) = \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}$$

What is $\mathrm{argmax}_{Q(z)}$ELBO$(x; Q, \theta)$?

ELBO is maximized when Q(z) is equal to p(z|x)

Therefore, we can approximate the true posterior by maximizing ELBO:

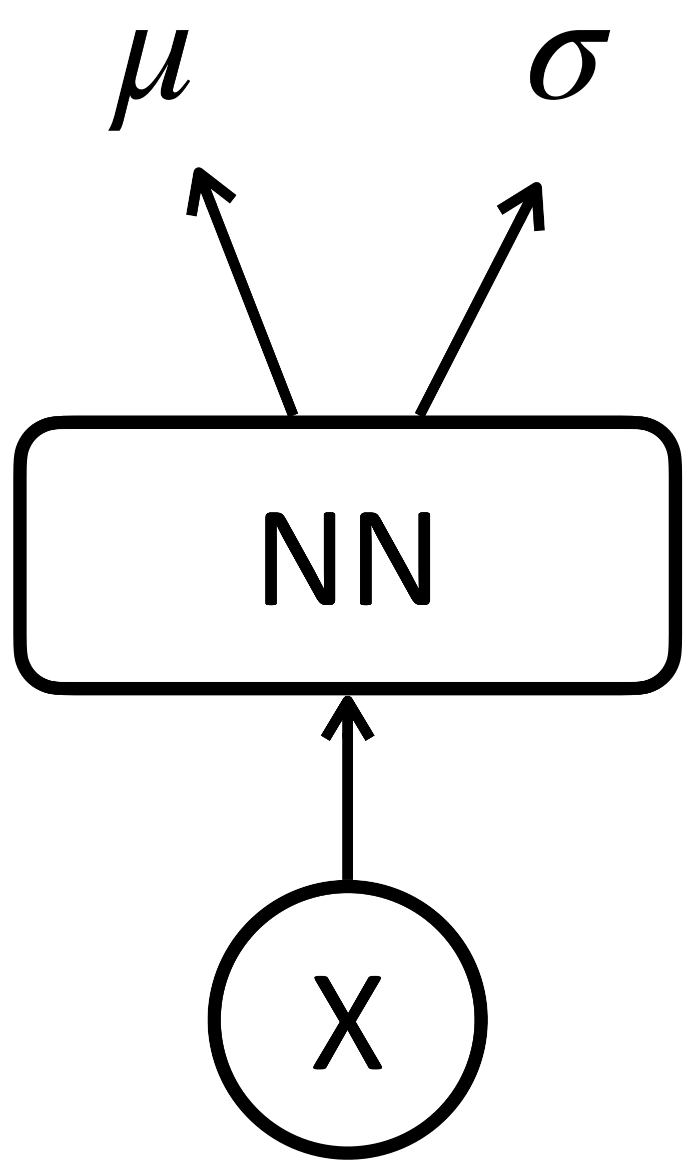
$$\mathrm{argmax}_{\phi} \sum_z q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Variational Inference

# A Common Choice for $q(z \mid x; \phi)$

$$q(z \mid x; \phi) = N(\mu, \sigma^2)$$

$$\mu, \sigma = g(x; \phi)$$




Inference model/network

# Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x;\phi) \log \frac{p(x,z;\theta)}{q(z \mid x;\phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x;\phi) \log \frac{p(x,z;\theta)}{q(z \mid x;\phi)}$$

Same objective, different parameters to optimize

Because we use approximate rather than exact posterior, it is also called Variational EM

# Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_z q(z|x;\phi) \log \frac{p(x,z;\theta)}{q(z|x;\phi)}$$

Can we do gradient descent over $\phi$?

M-Step:

$$\text{argmax}_{\theta} \sum_z q(z|x;\phi) \log \frac{p(x,z;\theta)}{q(z|x;\phi)}$$

We use MC sampling to approximate expectation and use gradient descent to optimize $\theta$

# Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

First, we cannot do sum, but we can sample $z_i$ from $q(z \mid x; \phi)$, which depends on $\phi$, how do we propagate gradients to $\phi$?

Try to express z as a deterministic function $z = g_{\phi}(\epsilon, x)$, where $\epsilon$ is an auxiliary random variable

$$z \sim N(\mu, \sigma^2) \longrightarrow z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim N(0,1)$$

Can you verify z in this equation is Gaussian?

# Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

For every gradient step (assuming batch size=1):

1. Randomly sample $\epsilon^{(i)} \sim N(0,1)$
2. Obtain z sample as $z^{(i)} = \mu + \sigma \odot \epsilon^{(i)}$
3. Perform gradient descent w.r.t. $\log \frac{p(x, z^{(i)}; \theta)}{q(z^{(i)} \mid x; \phi)}$

We can now propagate gradients from z to $\phi$

# Reparameterization Trick

VAE is a class of models

What kind of $q(z \mid x; \phi)$ allows for such a reparameterization trick?

1. Tractable inverse CDF. In this case, let $\boldsymbol{\epsilon} \sim \mathcal{U}(\mathbf{0}, \mathbf{I})$, and let $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ be the inverse CDF of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. Examples: Exponential, Cauchy, Logistic, Rayleigh, Pareto, Weibull, Reciprocal, Gompertz, Gumbel and Erlang distributions.
2. Analogous to the Gaussian example, for any "location-scale" family of distributions we can choose the standard distribution (with location $= 0$, scale $= 1$) as the auxiliary variable $\boldsymbol{\epsilon}$, and let $g(.) = \text{location} + \text{scale} \cdot \boldsymbol{\epsilon}$. Examples: Laplace, Elliptical, Student's t, Logistic, Uniform, Triangular and Gaussian distributions.
3. Composition: It is often possible to express random variables as different transformations of auxiliary variables. Examples: Log-Normal (exponentiation of normally distributed variable), Gamma (a sum over exponentially distributed variables), Dirichlet (weighted sum of Gamma variates), Beta, Chi-Squared, and F distributions.

Kingma et al. Auto-Encoding Variational Bayes

# ELBO

$$\sum_z q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)} = \mathbb{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z \mid x)]$$

ELBO is implemented with the following form:

$$\underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))}_{\text{KL Regularizer}}$$

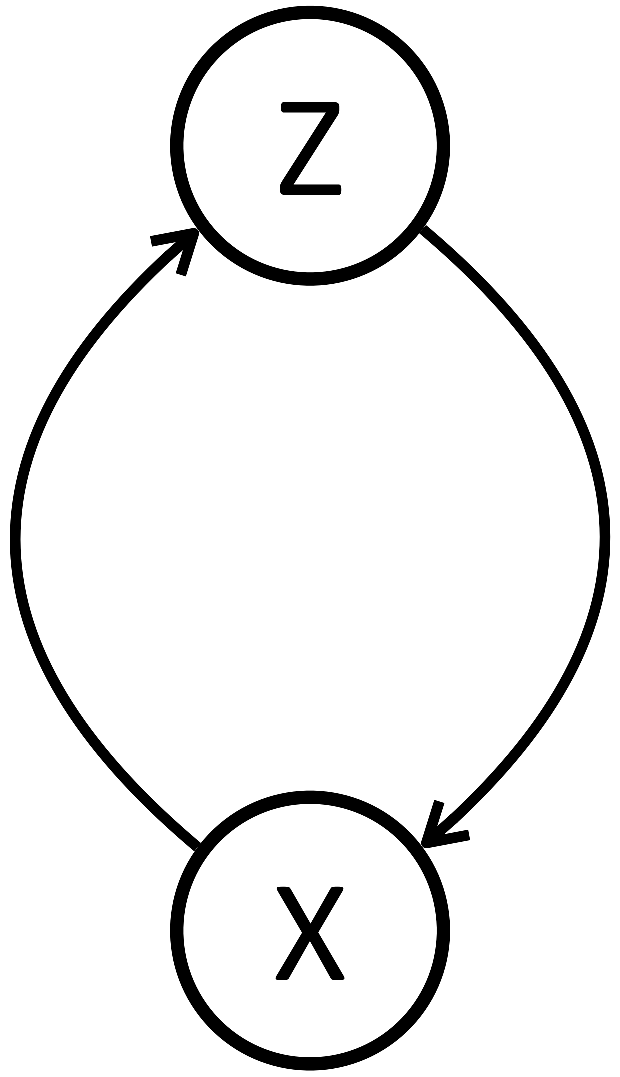
Autoencoder

# ELBO

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Autoencoder Loss

q(z|x) and p(z) are both Gaussian, there is a closed-form for this




This is why it is called variational "autoencoder"

# ELBO

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

$$\begin{aligned}\int q_{\boldsymbol{\theta}}(\mathbf{z}) \log p(\mathbf{z})\, d\mathbf{z} &= \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})\, d\mathbf{z} \\ &= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2)\end{aligned}$$

J is the dimensionality of z

$$\begin{aligned}\int q_{\boldsymbol{\theta}}(\mathbf{z}) \log q_{\boldsymbol{\theta}}(\mathbf{z})\, d\mathbf{z} &= \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, d\mathbf{z} \\ &= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma_j^2)\end{aligned}$$

$$\begin{aligned}-D_{KL}((q_{\boldsymbol{\phi}}(\mathbf{z})\|p_{\boldsymbol{\theta}}(\mathbf{z})) &= \int q_{\boldsymbol{\theta}}(\mathbf{z}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\theta}}(\mathbf{z})\right)\, d\mathbf{z} \\ &= \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2\right)\end{aligned}$$

# Training VAEs

E-Step:

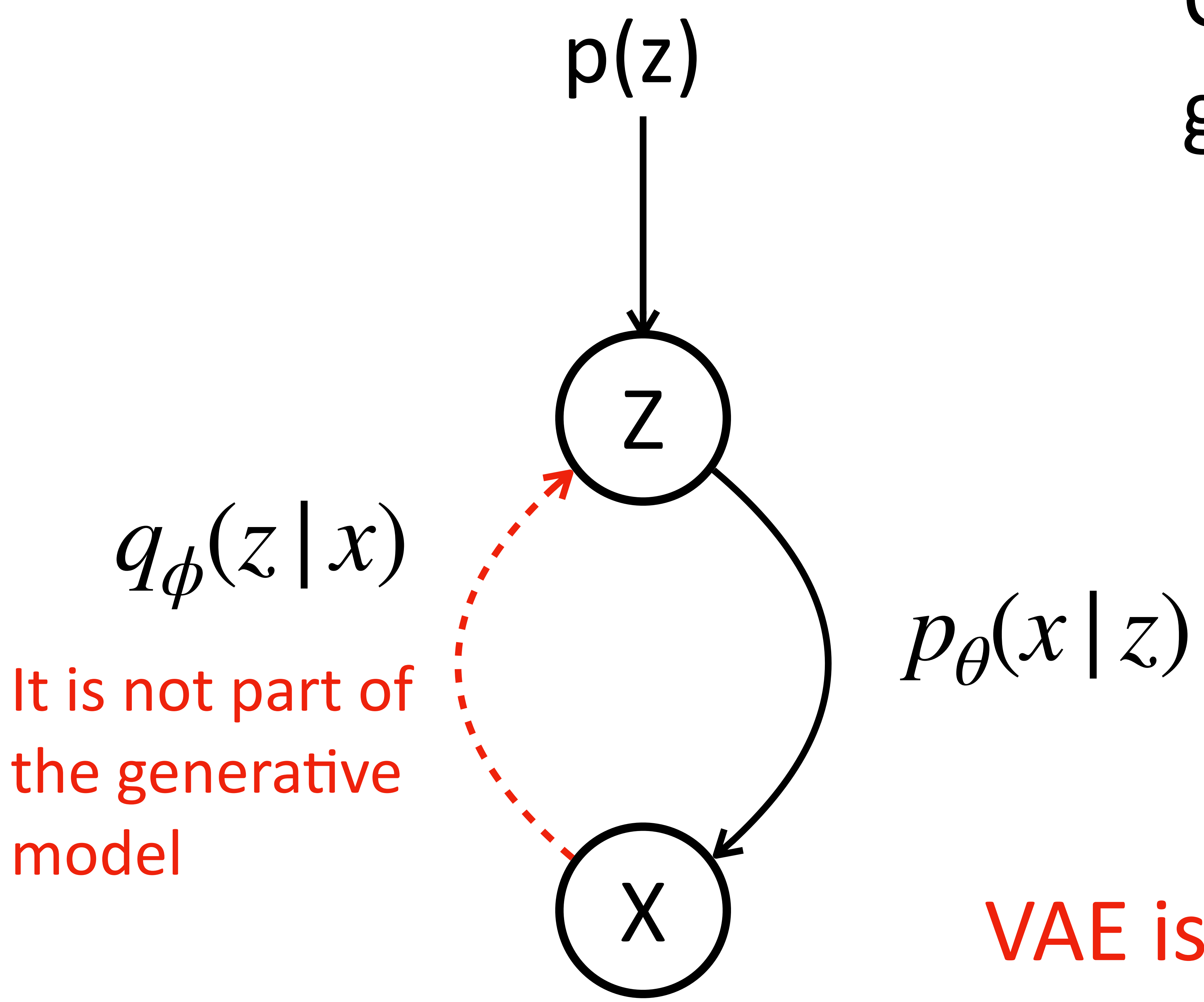
$$\text{argmax}_\phi \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

M-Step:

$$\text{argmax}_\theta \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Intuitively we hope to approximate p(z|x) with q(z|x) accurately in the E-step, to approximate the true EM algorithm

# Review VAE




Only the right (black) part defines the generative model, and the distribution

$p_\theta(x|z)$: generative network/decoder

$q_\phi(z|x)$: inference network/encoder

VAE is a name to represent both the model p(x) and the inference network that is used to train the model, but do not mix them together

# Training VAEs

**Algorithm 1** Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
**repeat**
  $\mathbf{X}^M \leftarrow$ Random minibatch of $M$ datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
**until** convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
**return** $\boldsymbol{\theta}, \boldsymbol{\phi}$

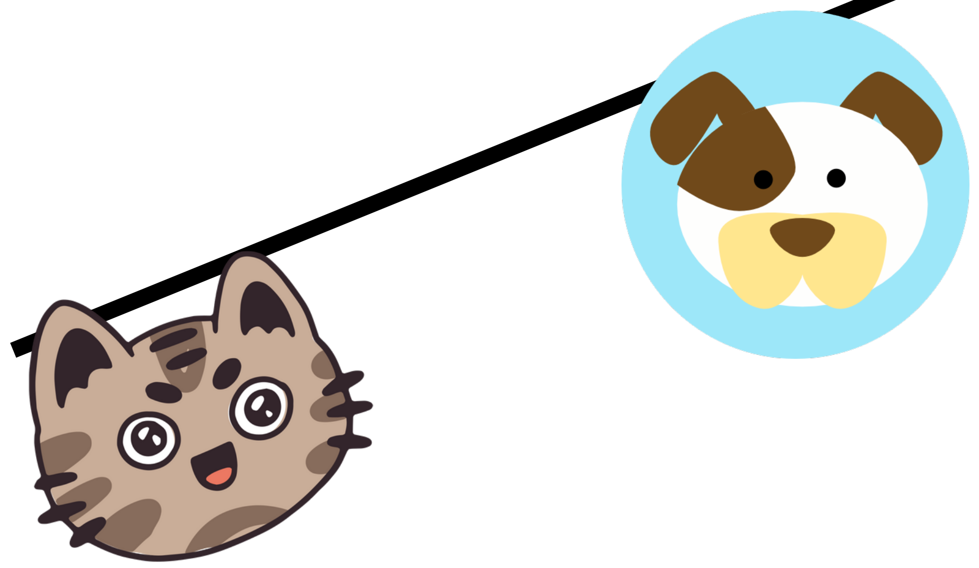
End-to-end, because the objectives are the same (ELBO)

VAE training is optimizing ELBO with gradient descent

# Recap: EM is Hill Climbing

$\log p(x; \theta)$ Only related to $\theta$, no $z$

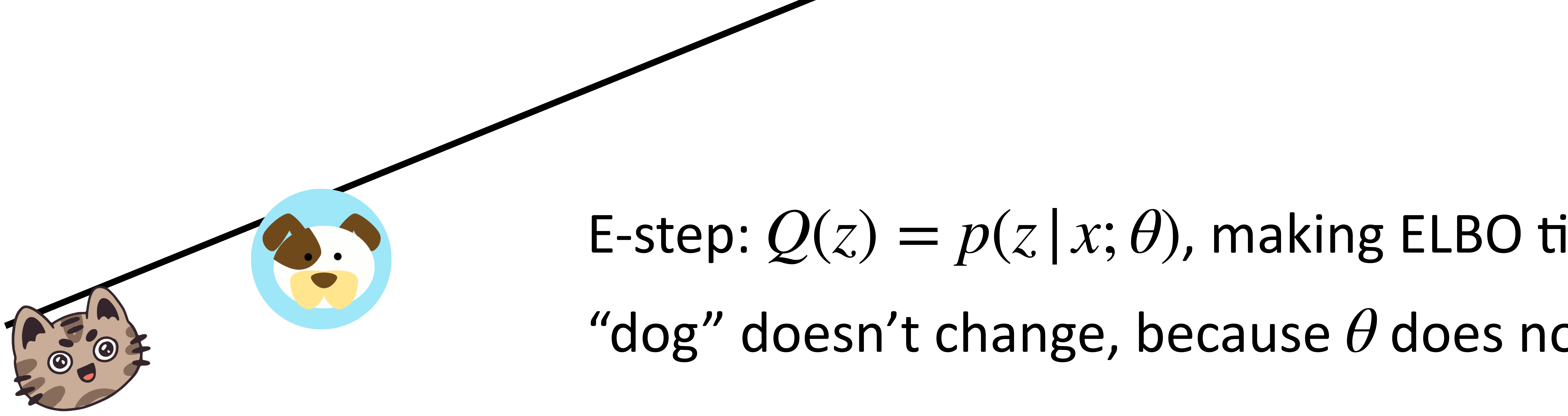
ELBO

# Recap: EM is Hill Climbing

$\log p(x;\theta)$

ELBO

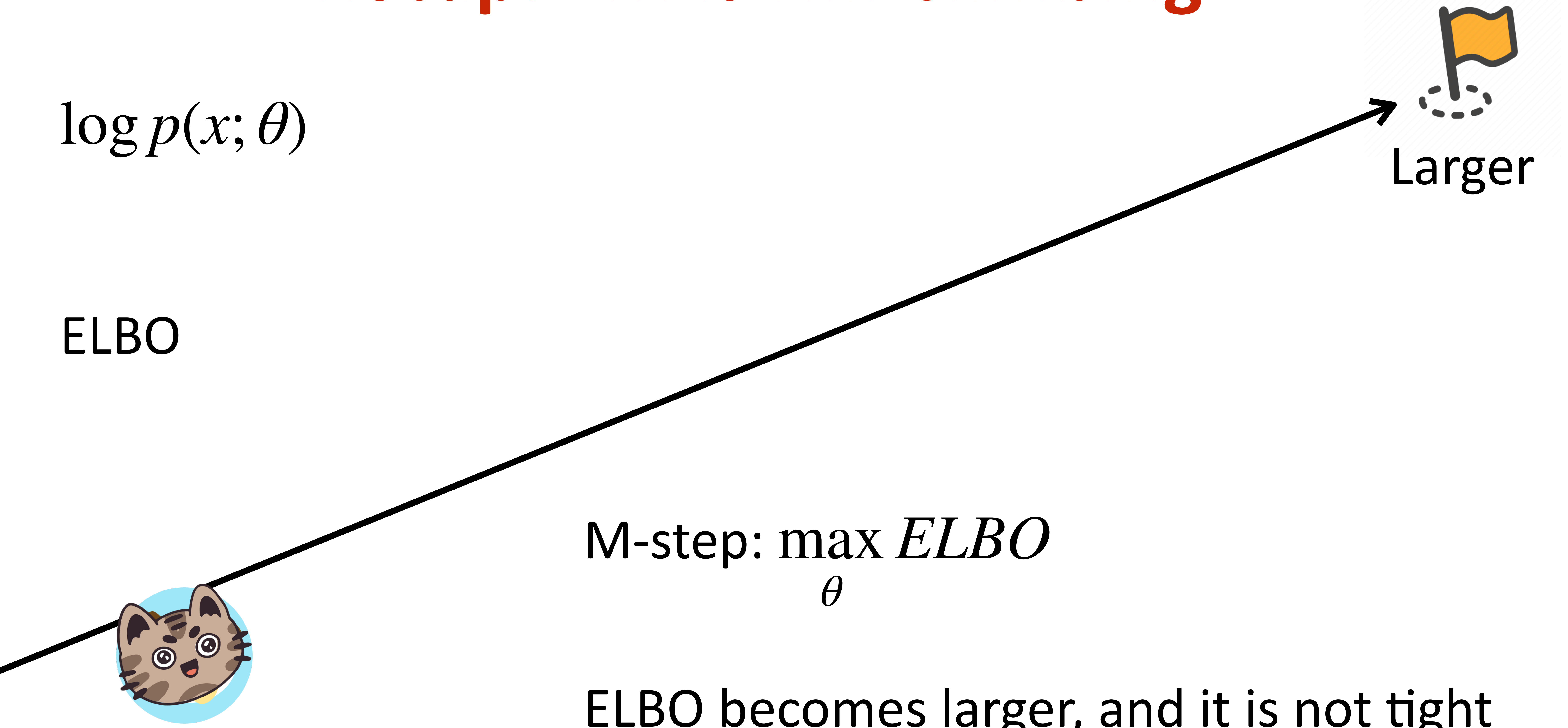
E-step: $Q(z) = p(z \mid x;\theta)$, making ELBO tight

"dog" doesn't change, because $\theta$ does not change

# Recap: EM is Hill Climbing

$\log p(x;\theta)$

ELBO




M-step: $\max_{\theta} ELBO$

ELBO becomes larger, and it is not tight anymore because posterior changes

# Is VAE training still Hill Climbing?

It is not, because q(z|x) may not be accurate to approximate p(z|x)

In VAE training, there is no guarantee that log p(x) is monotonically increasing

It just works in many cases

E-Step:

$$\text{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

According to EM, $\phi$ should be optimized to convergence to have a good approximation for p(z|x) before conducting the M-step, but VAE does not

# The Posterior Collapse Issue

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

In practice, it is often found that after training, $q_\phi(z|x) = p(z)$ and z and x becomes independent (especially in applications of NLP)

Z does not affect x, the model degenerates to a generative model without latent variables

Researchers commonly blame that the KL regularizer is too strong for this and use a weight $0 < \lambda < 1$ to control it:

Reconstruction Loss - $\lambda$ * KL regularizer

This is not a lower-bound of log p(x) anymore and it breaks MLE, but what is wrong with MLE?

# Is VAE training still Hill Climbing?

E-Step:

$$\text{argmax}_\phi \; \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

According to EM, $\phi$ should be optimized to convergence to have a good approximation for p(z|x) before conducting the M-step, but VAE does not

Can we make it closer to EM to have good guarantees?

# VAE training that is Closer to EM

At every iteration, perform multiple gradient updates of $\phi$ (E-step) before performing one step of $\theta$ (M-step)

Published as a conference paper at ICLR 2019

## LAGGING INFERENCE NETWORKS AND POSTERIOR COLLAPSE IN VARIATIONAL AUTOENCODERS

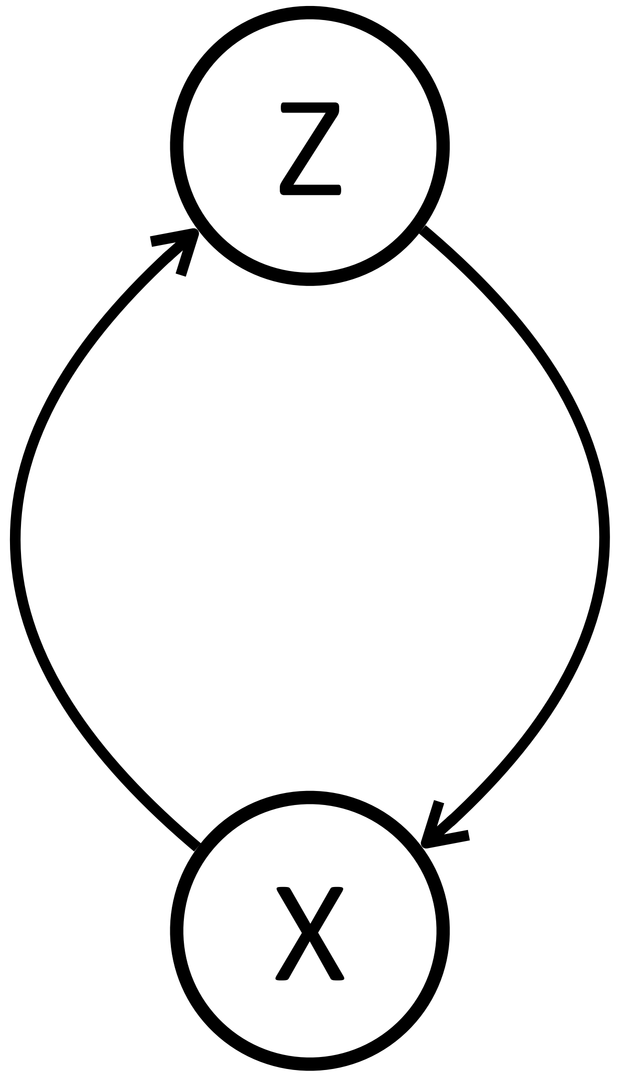
**Junxian He, Daniel Spokoyny, Graham Neubig**
Carnegie Mellon University
{junxianh,dspokoyn,gneubig}@cs.cmu.edu

**Taylor Berg-Kirkpatrick**
University of California San Diego
tberg@eng.ucsd.edu

# AutoEncoders

VAE: $$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

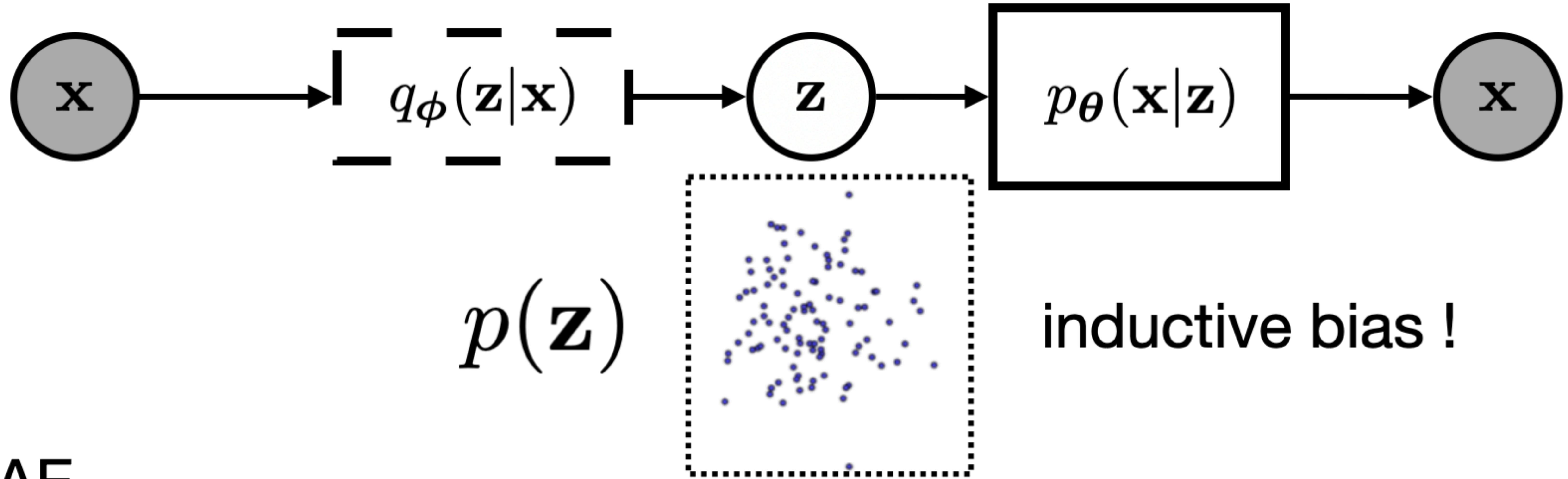
AE: $$\log p_\theta(x \mid q(x))$$



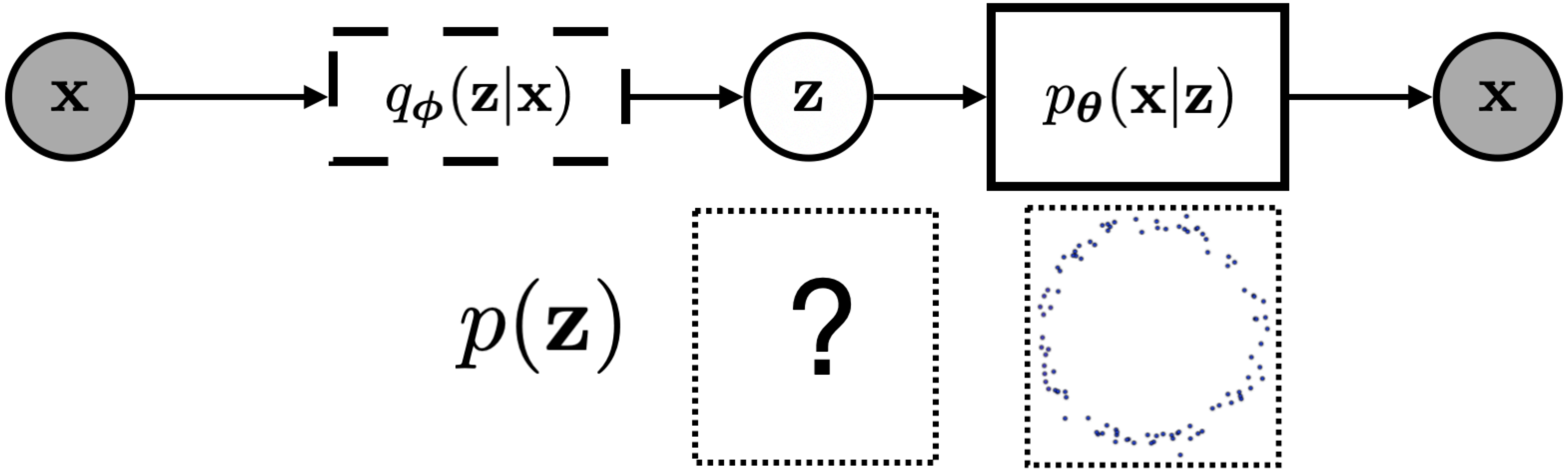

1. Can we generate X samples from an autoencoder?
2. Can we approximate p(x) given x with an autoencoder?
3. What is the difference between the representation space from AE and VAE?

# VAE v.s. AE

VAE




AE

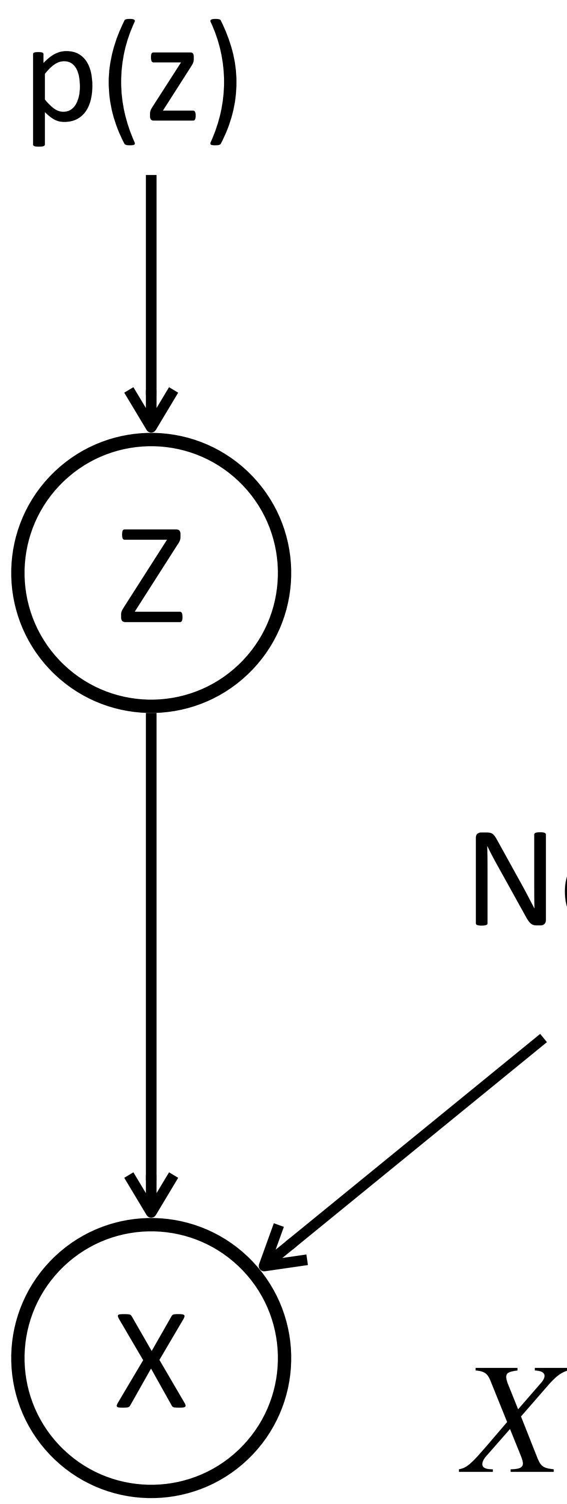


**Generative Adversarial Nets**

**Ian J. Goodfellow, Jean Pouget-Abadie,[*] Mehdi Mirza, Bing Xu, David Warde-Farley,**
**Sherjil Ozair,[†] Aaron Courville, Yoshua Bengio[‡]**
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

# Generative Adversarial Networks

# The GAN Model




Neural Networks

$$X = G(z)$$

The same as the VAE model, except that x is a deterministic function of z, but it can be a distribution as well

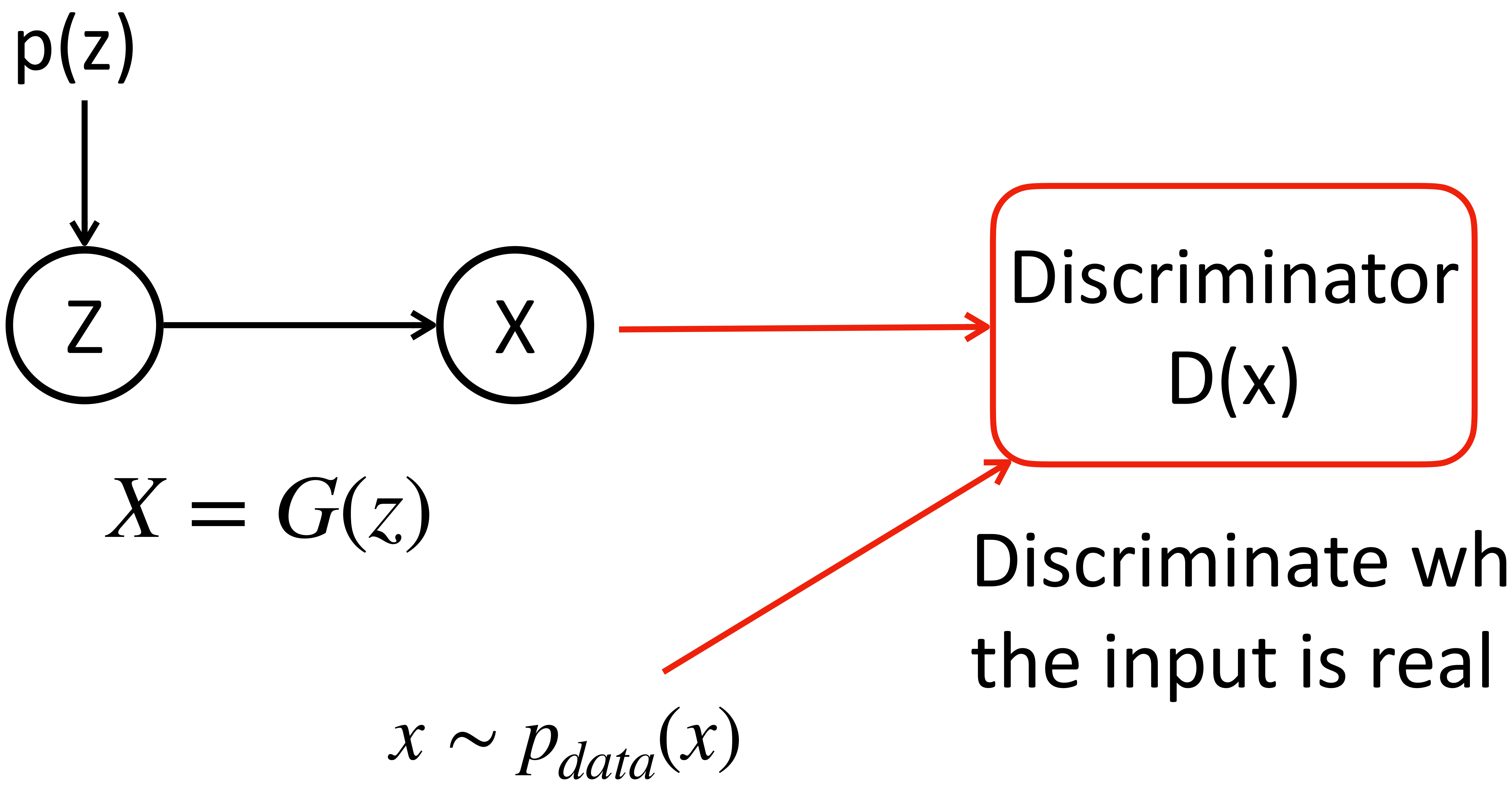
Can VAE use a deterministic x = G(z)?

Sometimes we call GANs *implicit* generative models

You can draw samples, but hard to evaluate p(x)

# Training GANs

Computation Graph




Discriminate whether the input is real or fake

1. Generator is trained to produce realistic examples to fool the discriminator
2. Discriminator is trained to discriminate real and fake examples

# Training GANs

1. Generator is trained to produce realistic examples to fool the discriminator
2. Discriminator is trained to discriminate real and fake examples

The two objectives are against each other

Adversarial Game

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Classification loss

G(z) is trained to minimize the probability of G(z) recognized as “fake” by D

D(x) is trained with a standard classification loss

# Training GANs

1. GAN is a new algorithm to train a common generative model (VAE as well)

2. GAN training is not MLE