

CS769 Advanced NLP

# Latent Variable Models

Junjie Hu



Slides adapted from Graham

<https://junjiehu.github.io/cs769-spring23/>

# Goal for Today

- **Variational Auto-encoder (VAE)**
  1. Maximize Evidence Lower Bound (ELBO)
  2. Reparametrization trick
  3. Stabilizing Training
- **Discrete Latent Variable Models (LVM)**
  1. Enumerate
  2. Sampling
  3. Gumbel-softmax
- **LVM Applications in NLP**

# Discriminative vs. Generative Models

- **Discriminative model:** calculate the probability of output given input  $P(Y|X)$
- **Generative model:** calculate the probability of a variable  $P(X)$ , or multiple variables  $P(X,Y)$
- Which of the following models are discriminative vs. generative?
  - Standard BiLSTM POS tagger
  - Language model

# Types of Variables

- Observed vs. Latent:
  - **Observed:** something that we can see from our data, e.g. X or Y
  - **Latent:** a variable that we assume exists, but we aren't given the value
- Deterministic vs. Random:
  - **Deterministic:** variables that are calculated directly according to some deterministic function
  - **Random (stochastic):** variables that obey a probability distribution, and may take any of several (or infinite) values

# Quiz: What Types of Variables?

- In the an attentional sequence-to-sequence model using MLE/teacher forcing, are the following variables observed or latent? deterministic or random?
  - The input word ids **f**
  - The encoder hidden states **h**
  - The attention values **a**
  - The output word ids **e**

# Latent Variable Models

- A vector of latent variables  $\mathbf{z}$  in a high-dim space  $\mathcal{Z}$  which we can sample from some PDF  $p(\mathbf{z})$  over  $\mathcal{Z}$
- For every observed  $\mathbf{x}$  in our dataset, there is a  $\mathbf{z}$  that causes the model to generate  $\mathbf{x}$
- A latent variable model (LVM) is a probability distribution over two sets of variables  $\mathbf{x}, \mathbf{z}$ :

$$p(\mathbf{x}|\mathbf{z}; \theta)$$

# Maximum Likelihood Framework

- Maximize the probability of each  $\mathbf{x}$  in the training set under the generative process according to:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z}; \theta)p(\mathbf{z})d\mathbf{z}$$

# Why LVM?

- Intuitively, latent variable  $\mathbf{z}$  enables the model to first decide which property to generate before it assigns values to the output  $\mathbf{x}$
- Example:
  - Sample a LV  $z$  from the set  $[0, \dots, 9]$  before generating an image for the digit

$$z=2 \rightarrow x= \img alt="A handwritten digit '2' on a black background." data-bbox="518 612 588 705"/>$$

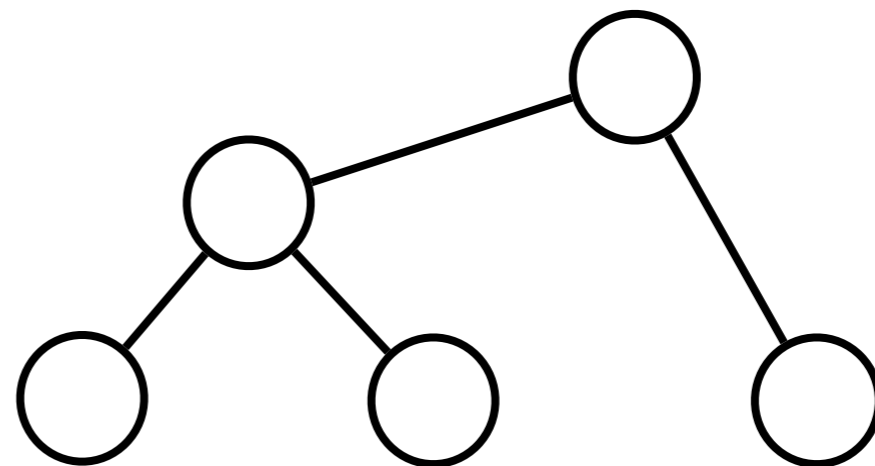
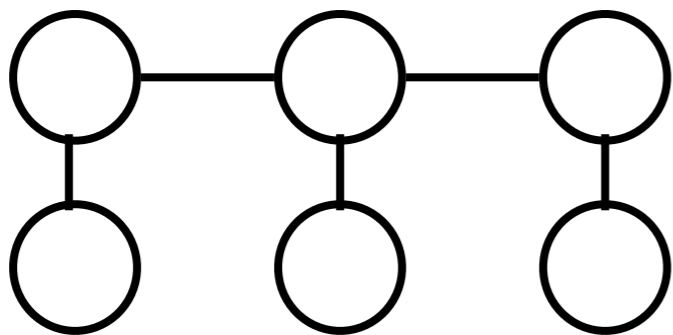
- Sample a sentiment from {positive, negative} before generating a positive/negative movie review.

$$z=\text{"negative"} \rightarrow x= \text{"This is a bad movie."}$$



# What Types of Latent Variables?

- Latent continuous vector (e.g. variational auto-encoder)
- Latent discrete vector (e.g. topic model)
- Latent structure (e.g. HMM or tree-structured model)

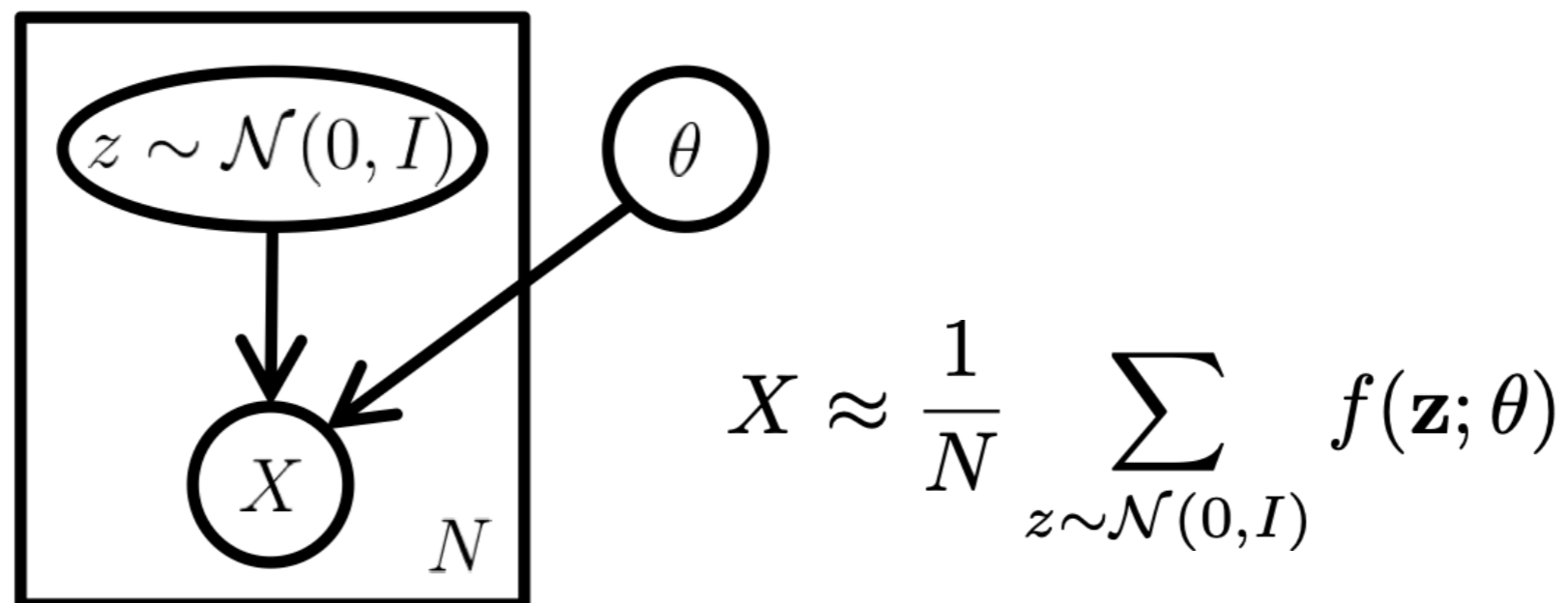


# Variational Auto-encoders

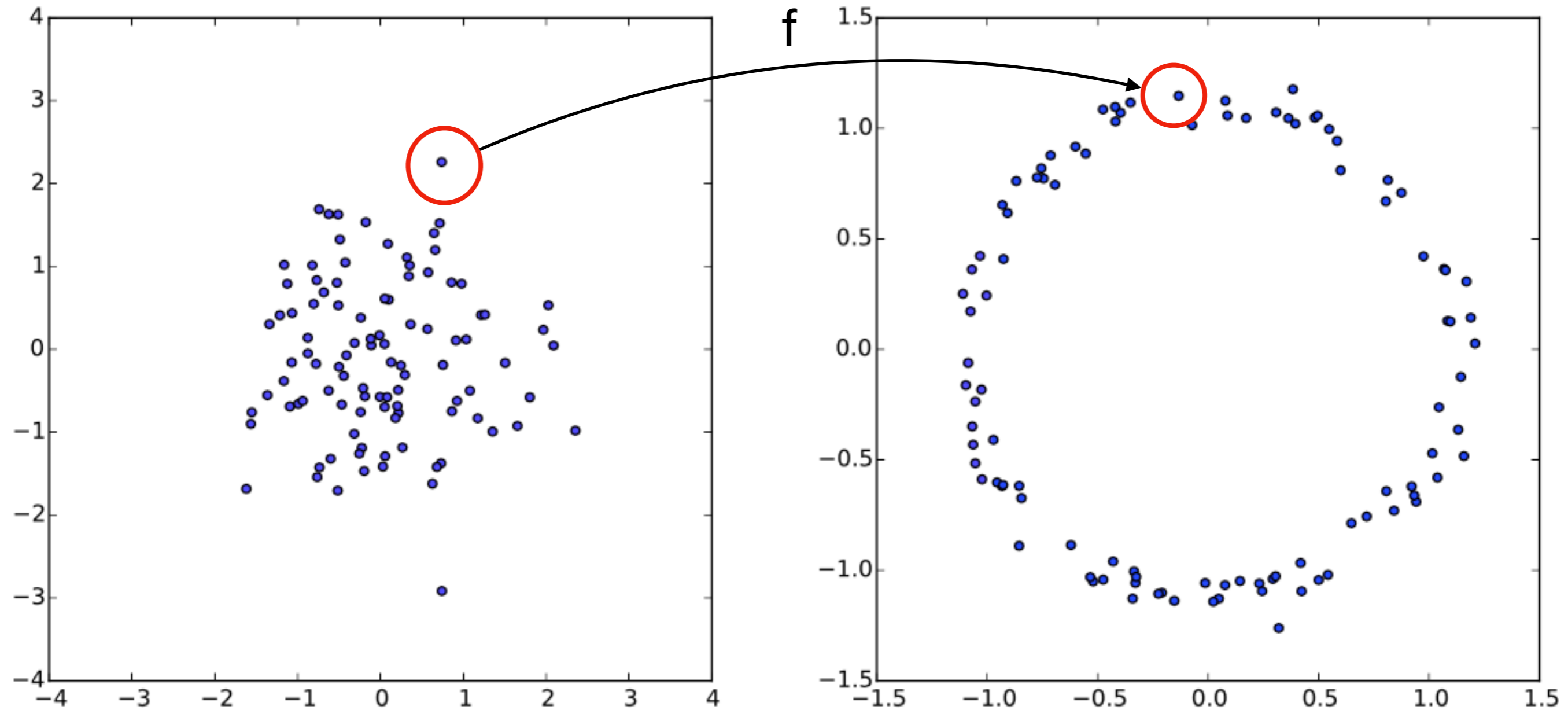
(Kingma and Welling 2014)

# VAE as a Graphical Model

- We have an observed datapoint  $X$  in our dataset
- We have a latent variable  $\mathbf{z}$  sampled from a Gaussian
- We have a deterministic function  $f(\mathbf{z}; \theta)$  that maps  $\mathbf{z}$  to the data space  $\mathcal{X}$ . If  $\mathbf{z}$  is a random variable (r.v.) in  $\mathcal{Z}$ , then  $f(\mathbf{z}; \theta)$  is also a r.v. in  $\mathcal{X}$
- If we repeat this sampling  $N$  times, we hope to approximate  $X$  by  $f(\mathbf{z}; \theta)$



# An Example (Goersch 2016)

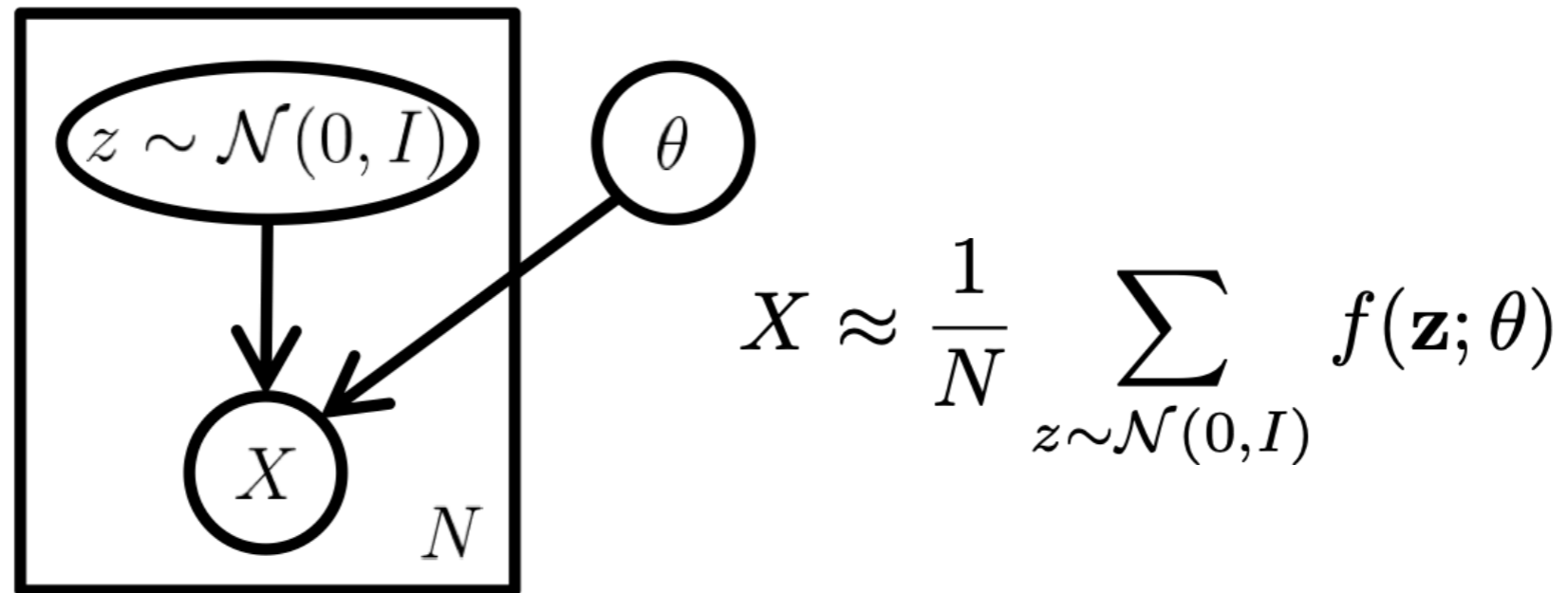


**$\mathbf{z}$**

$$f(\mathbf{z}) = \mathbf{z}/10 + \mathbf{z}/\|\mathbf{z}\|$$

**$\mathbf{x}$**

# A probabilistic perspective on Variational Auto-Encoder



- For each datapoint  $i$  :
  - Draw latent variables  $z_i \sim p(z)$  (prior)
  - Draw data point  $x_i \sim p_\theta(x|z)$
- Joint probability distribution over data and latent variables:
$$p(x, z) = p(z)p_\theta(x|z)$$
- Similar to Naive Bayes, HMM: only the prior differs

# What is Our Loss Function?

- We would like to maximize the corpus log likelihood

$$\log P(\mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} \log P(\mathbf{x}; \theta)$$

- For a single example, the marginal likelihood is

$$P(\mathbf{x}; \theta) = \int P(\mathbf{x} | \mathbf{z}; \theta) P(\mathbf{z}) d\mathbf{z}$$

- We can approximate this by sampling  $\mathbf{z}$ s then summing

$$P(\mathbf{x}; \theta) \approx \sum_{\mathbf{z} \in S(\mathbf{x})} P(\mathbf{x} | \mathbf{z}; \theta) \quad \text{where} \quad S(\mathbf{x}) := \{\mathbf{z}'; \mathbf{z}' \sim P(\mathbf{z})\}$$

# Variational Inference

Two tasks of interest:

- Learn the parameters  $\theta$  of  $p_\theta(x|z)$
- **Inference** over  $z$  with the *posterior* distribution:  $p_\theta(z|x)$  given input  $x$ , what are its latent factors?

$$p_\theta(z|x) = \frac{p_\theta(x|z)p(z)}{p(x)}$$

$$p(x) = \int p(z)p_\theta(x|z)dz \quad \leftarrow \text{intractable}$$

- However, as  $p(x)$  is intractable  $p_\theta(z|x)$  is also intractable.
- Solution: **Variational inference** approximates the posterior  $p_\theta(z|x)$  with a family of distributions  $q_\phi(z|x)$

# Variational Inference

- Variational inference approximates the true posterior  $p_{\theta}(z|x)$  with a family of distributions  $q_{\phi}(z|x)$

$$\text{minimize : } \text{KL}(q_{\phi}(z|x) || p_{\theta}(z|x))$$

- One other target is to maximize the log-data likelihood which can be rewritten as:

$$\log p(x) = \text{ELBO} + \text{KL}(q_{\phi}(z|x) || p_{\theta}(z|x))$$

- Combining the above two, this is equivalent to maximize the **Evidence Lower Bound (ELBO)**

$$\text{ELBO} = \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - \text{KL}(q_{\phi}(z|x) || p(z))$$

$$\text{KL}(q || p) \geq 0 \Rightarrow \log p(x) \geq \text{ELBO}$$



# Variational Inference

- Variational inference approximates the true posterior  $p_{\theta}(z|x)$  with a family of distributions  $q_{\phi}(z|x)$

$$\text{minimize : } \text{KL}(q_{\phi}(z|x) || p_{\theta}(z|x))$$

- One other target is to maximize the log-data likelihood which can be rewritten as:

$$\log p(x) = \text{ELBO} + \text{KL}(q_{\phi}(z|x) || p_{\theta}(z|x))$$

- **Evidence Lower Bound (ELBO)**

$$\text{ELBO} = \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - \text{KL}(q_{\phi}(z|x) || p(z))$$

$$\text{KL}(q || p) \geq 0 \Rightarrow \log p(x) \geq \text{ELBO}$$

# Variational Inference

- Variational inference approximates the true posterior  $p_\theta(z|x)$  with a family of distributions  $q_\phi(z|x)$

$$\text{minimize : } \text{KL}(q_\phi(z|x) || p_\theta(z|x))$$

$$\text{maximize: } \log p(x) = \text{ELBO} + \text{KL}(q_\phi(z|x) || p_\theta(z|x))$$

- Combining the above two objectives, it is equivalent to maximize ELBO

$$\text{maximize : ELBO}$$

# Variational Auto-Encoders

$$\log p_{\theta}(\mathbf{x}) \geq \text{ELBO}$$

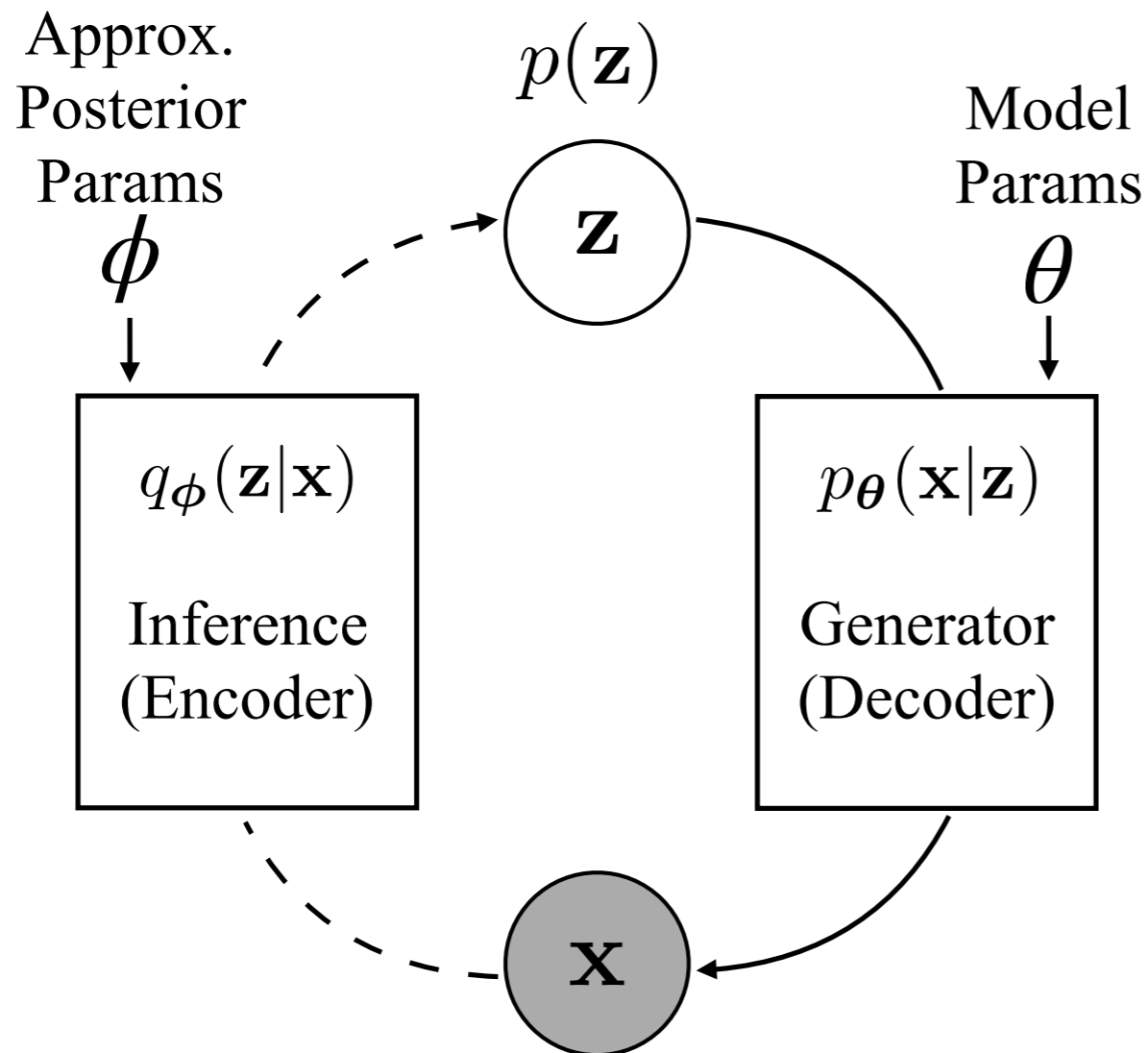
$$\underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))}_{\text{KL Regularizer}}$$

The inequality holds for any  $q_{\phi}(z|x)$ , but the lower bound is tight only if  $p_{\theta}(z|x) = q_{\phi}(z|x)$

$p_{\theta}(z|x)$  is intractable

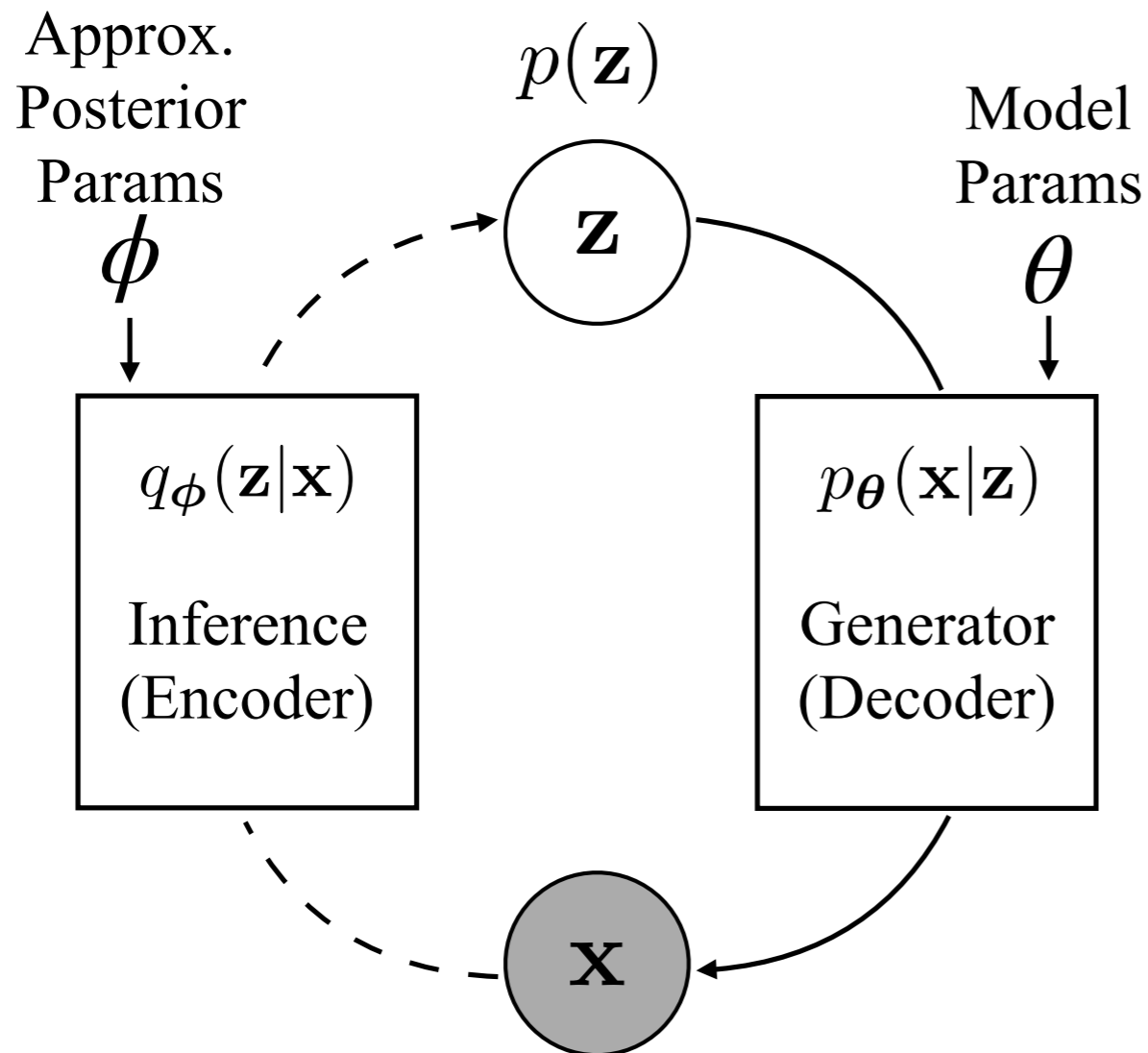
# Variational Autoencoders

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



# Variational Autoencoders

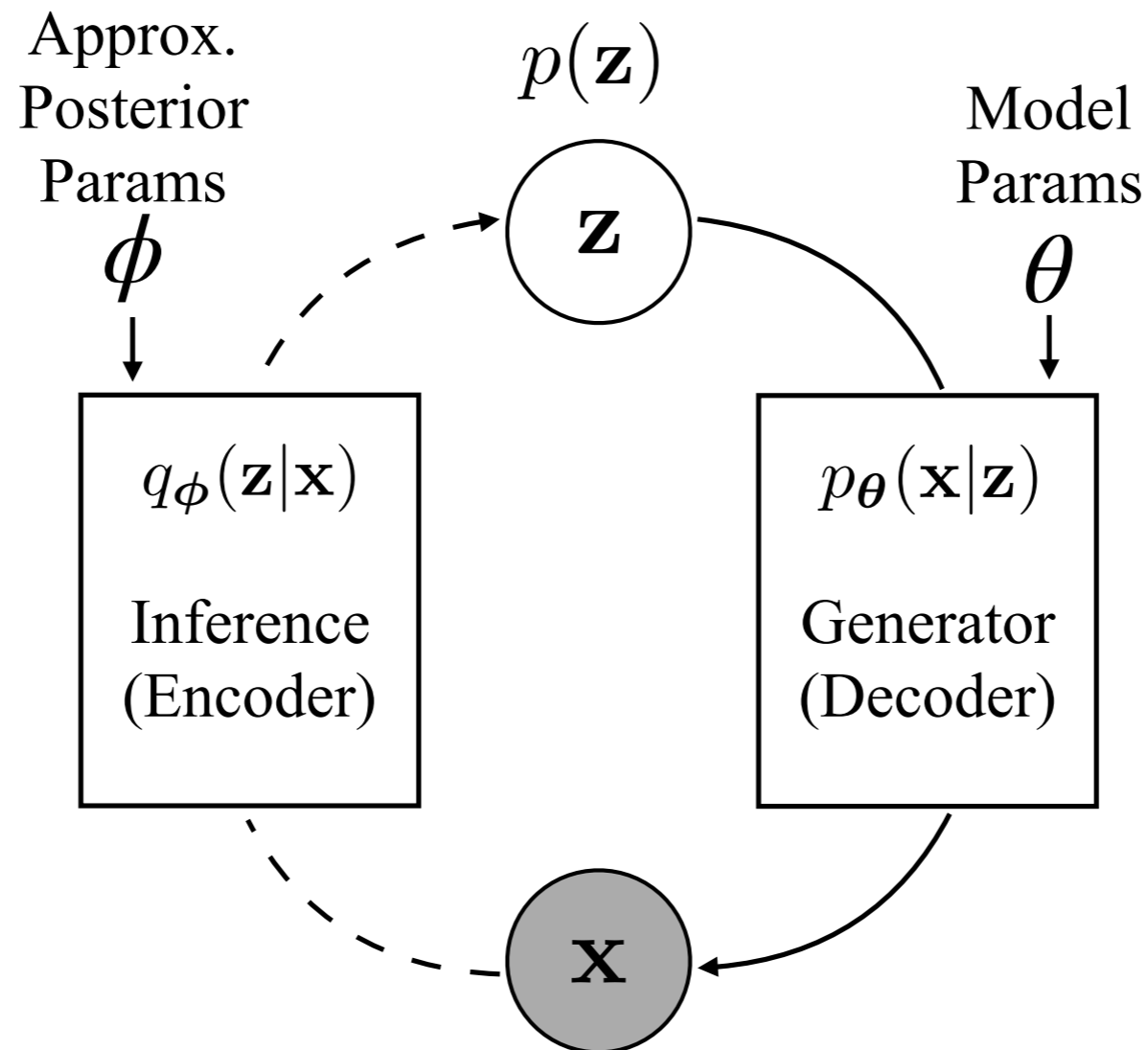
$$\log p_{\theta}(\mathbf{x}) \geq \underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))}_{\text{KL Regularizer}}$$



Regularized Autoencoder

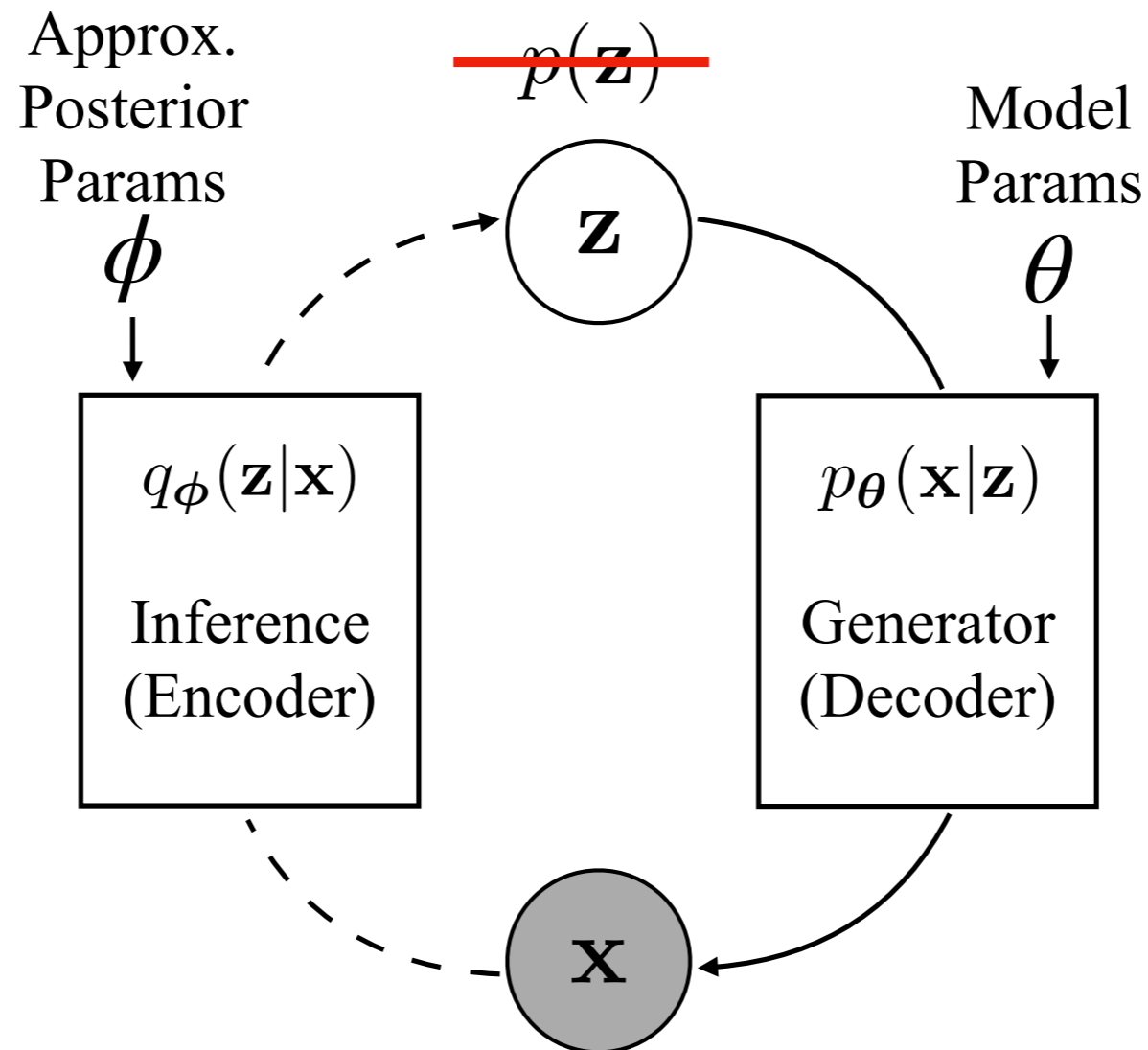
# Why prior ?

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



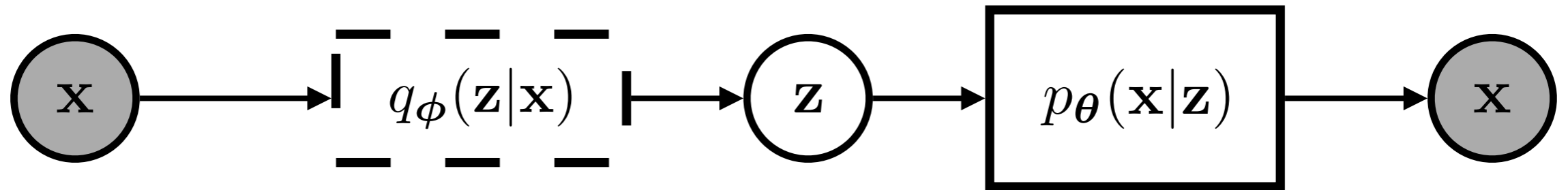
# Why prior ?

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



# VAE vs. AE

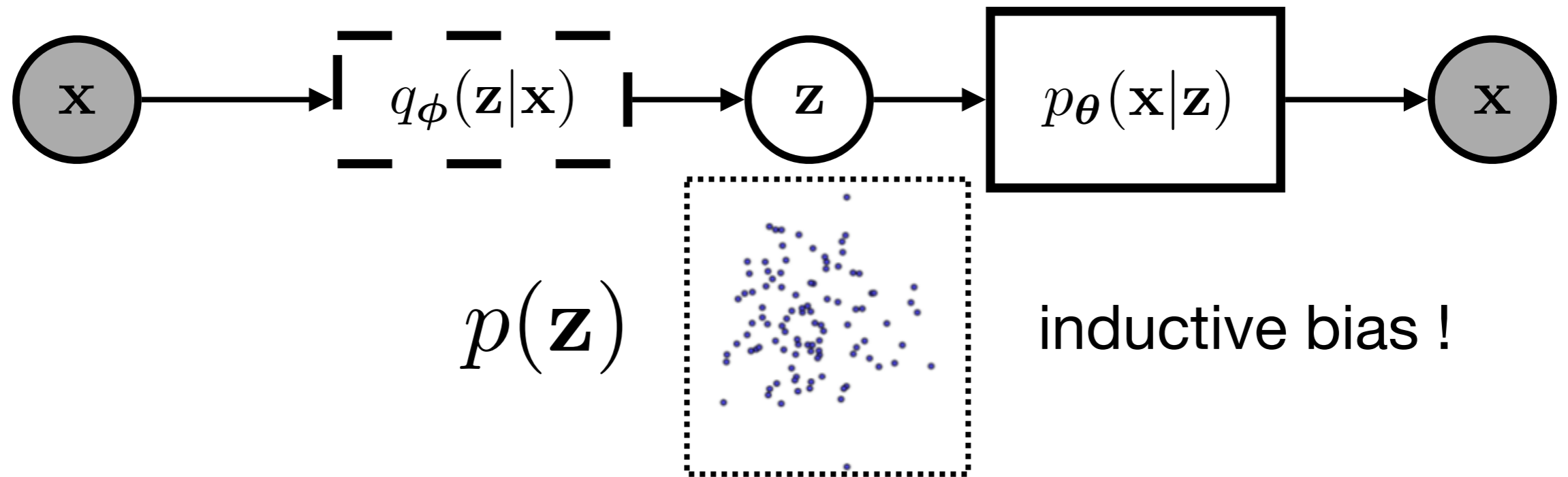
VAE





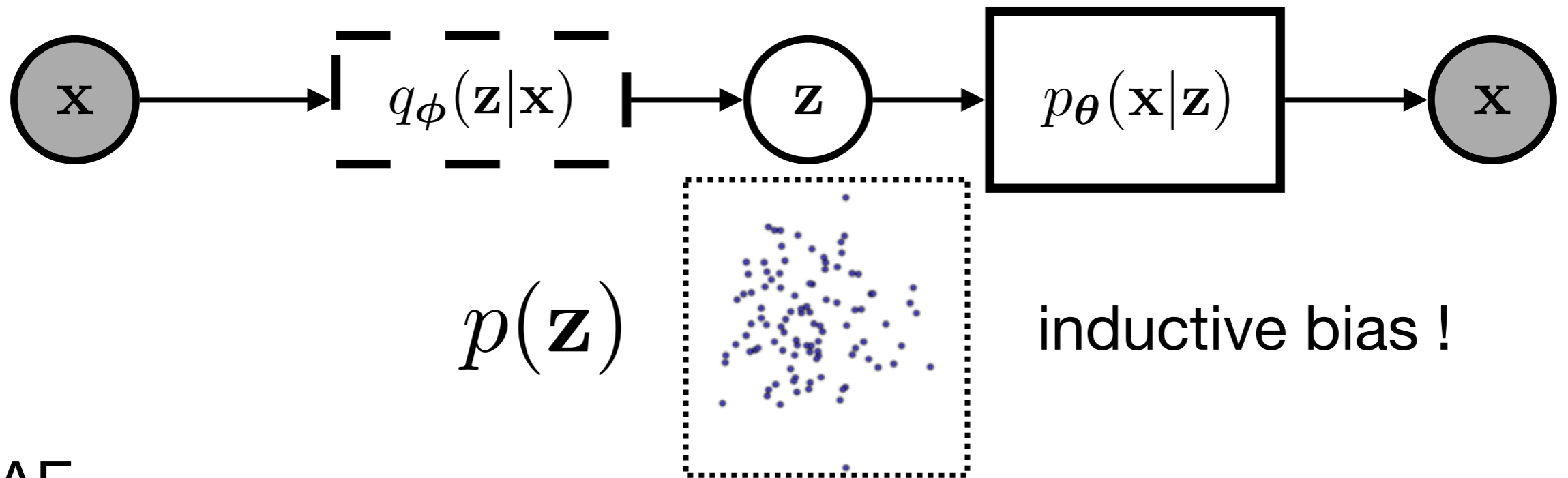
# VAE vs. AE

VAE

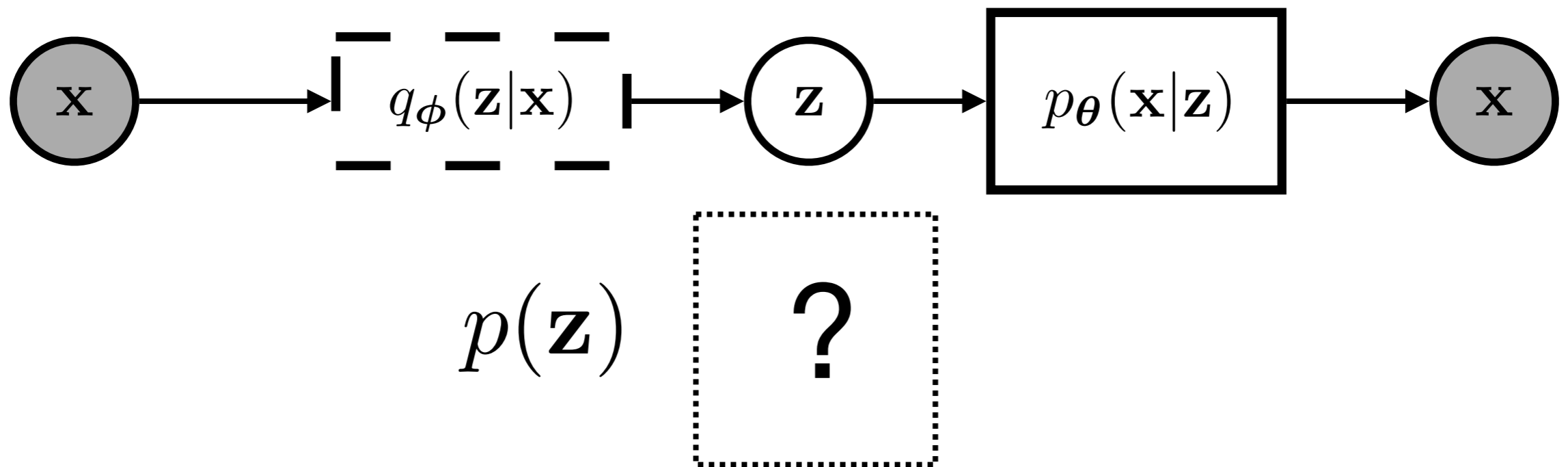


# VAE vs. AE

VAE

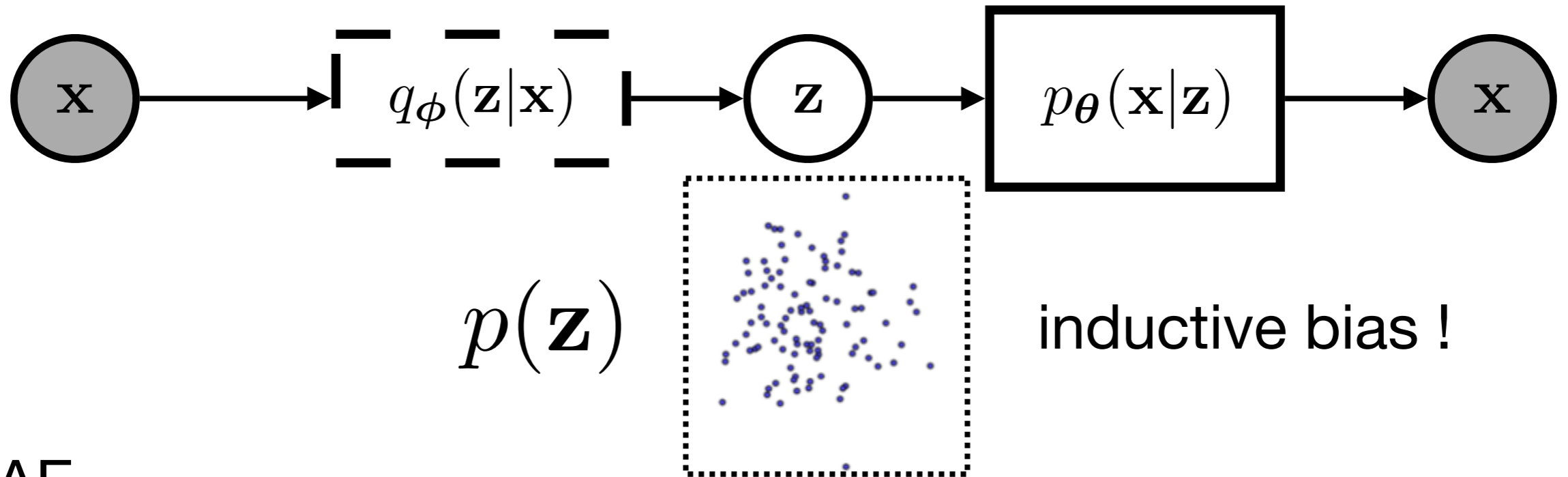


AE

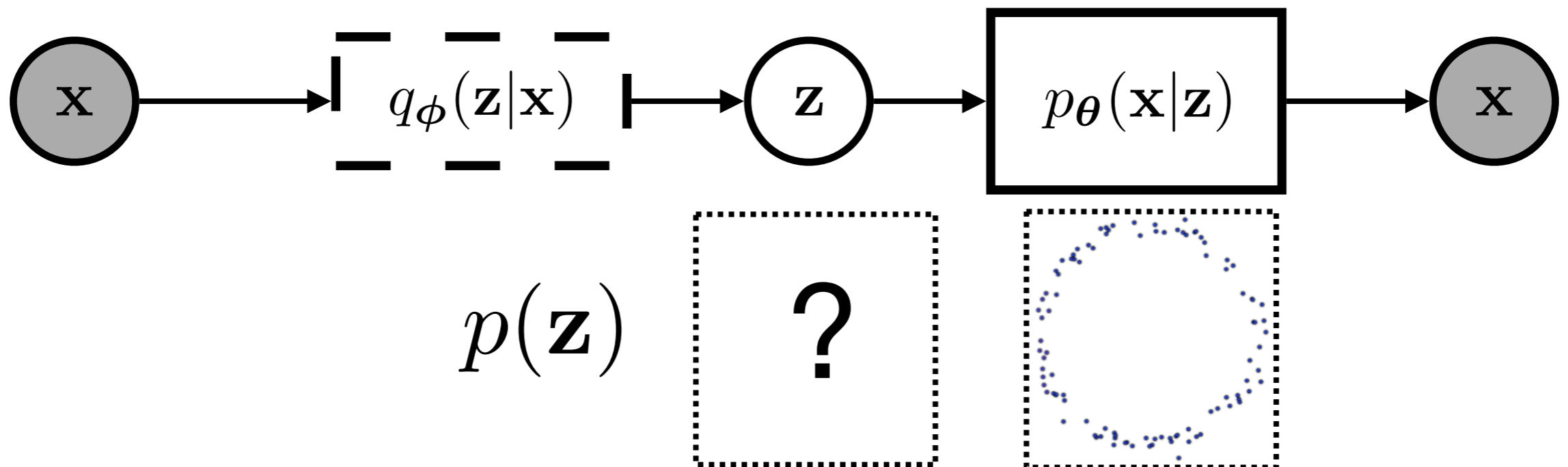


# VAE vs. AE

VAE

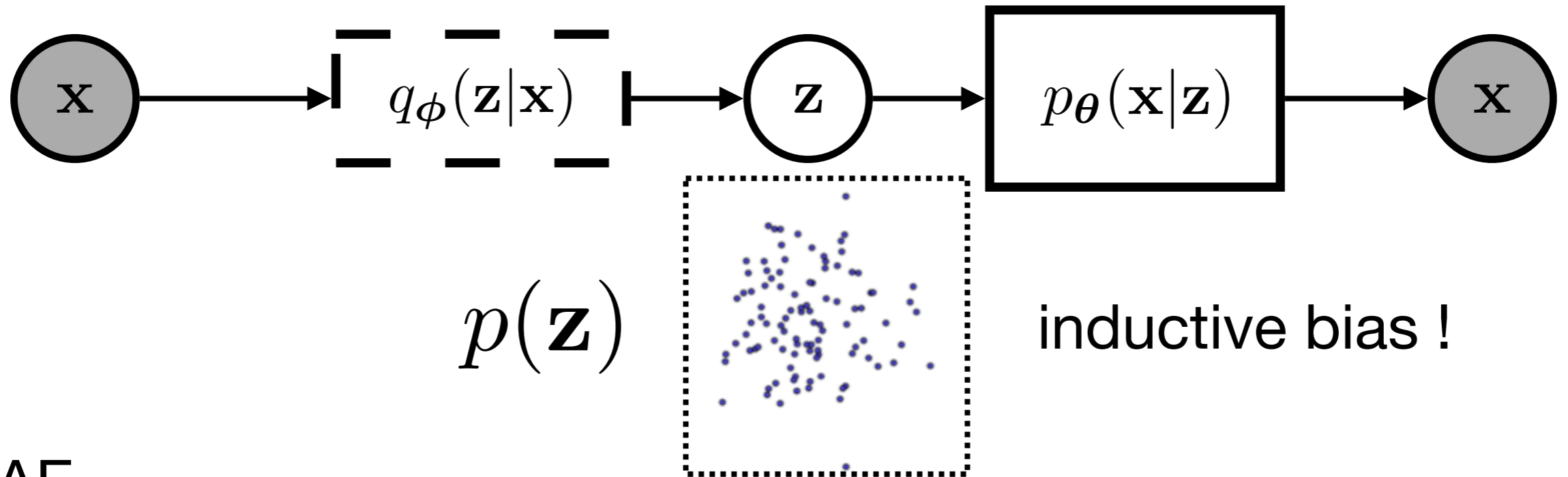


AE

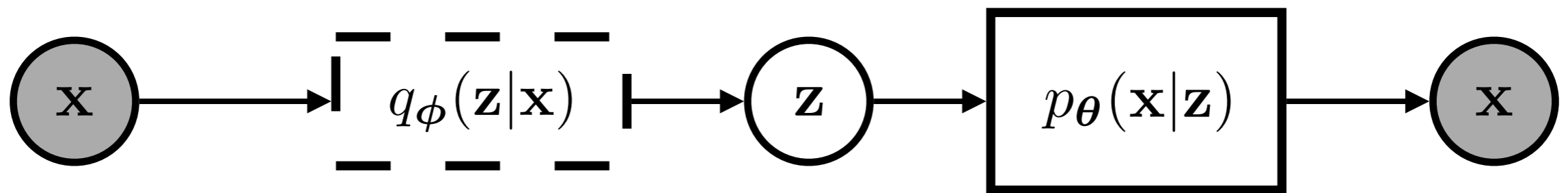


# VAE vs. AE

VAE



AE

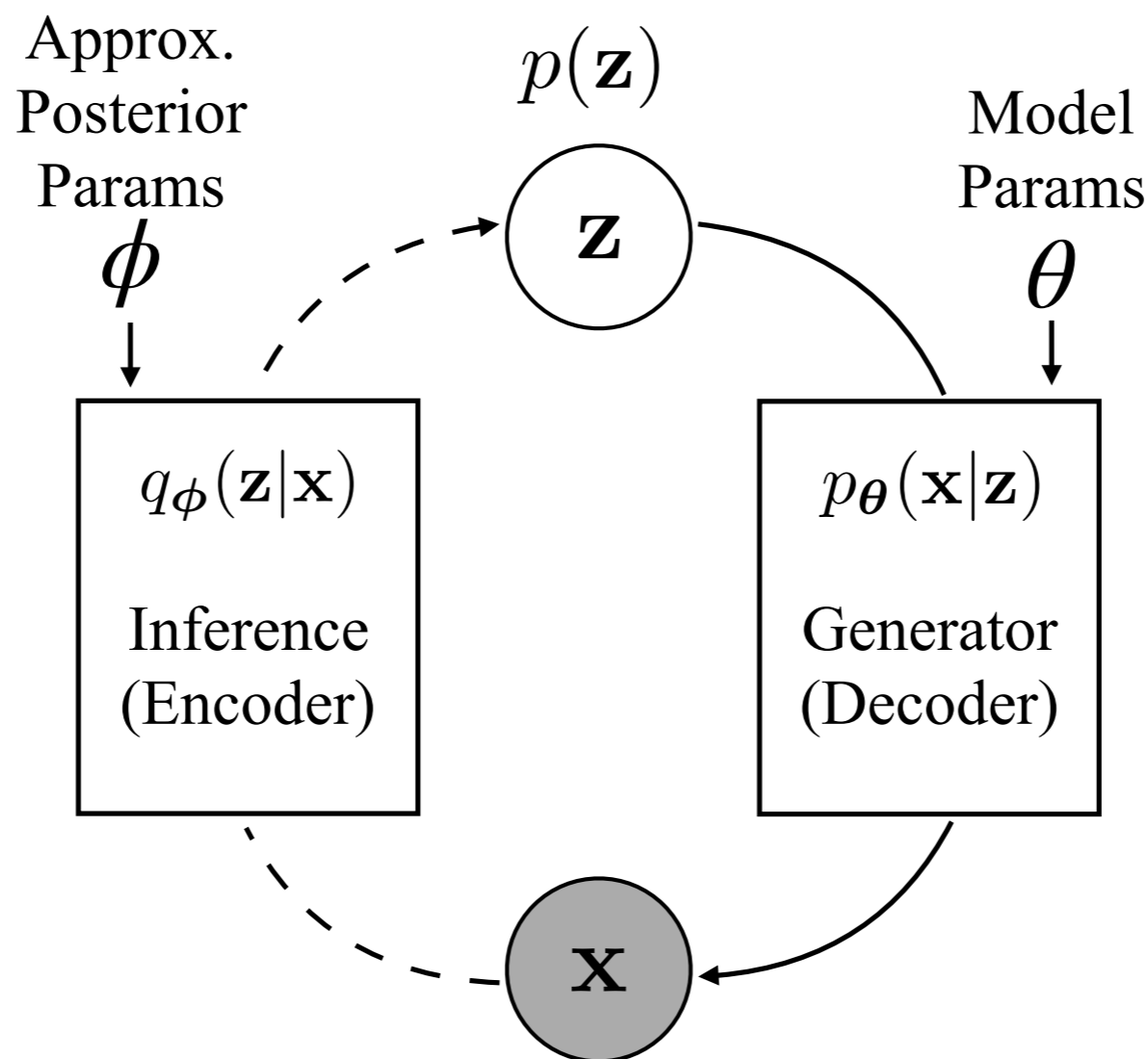


AE is not generative model:

- (1) Can't sample new data from AE
- (2) Can't compute the log likelihood of data  $x$

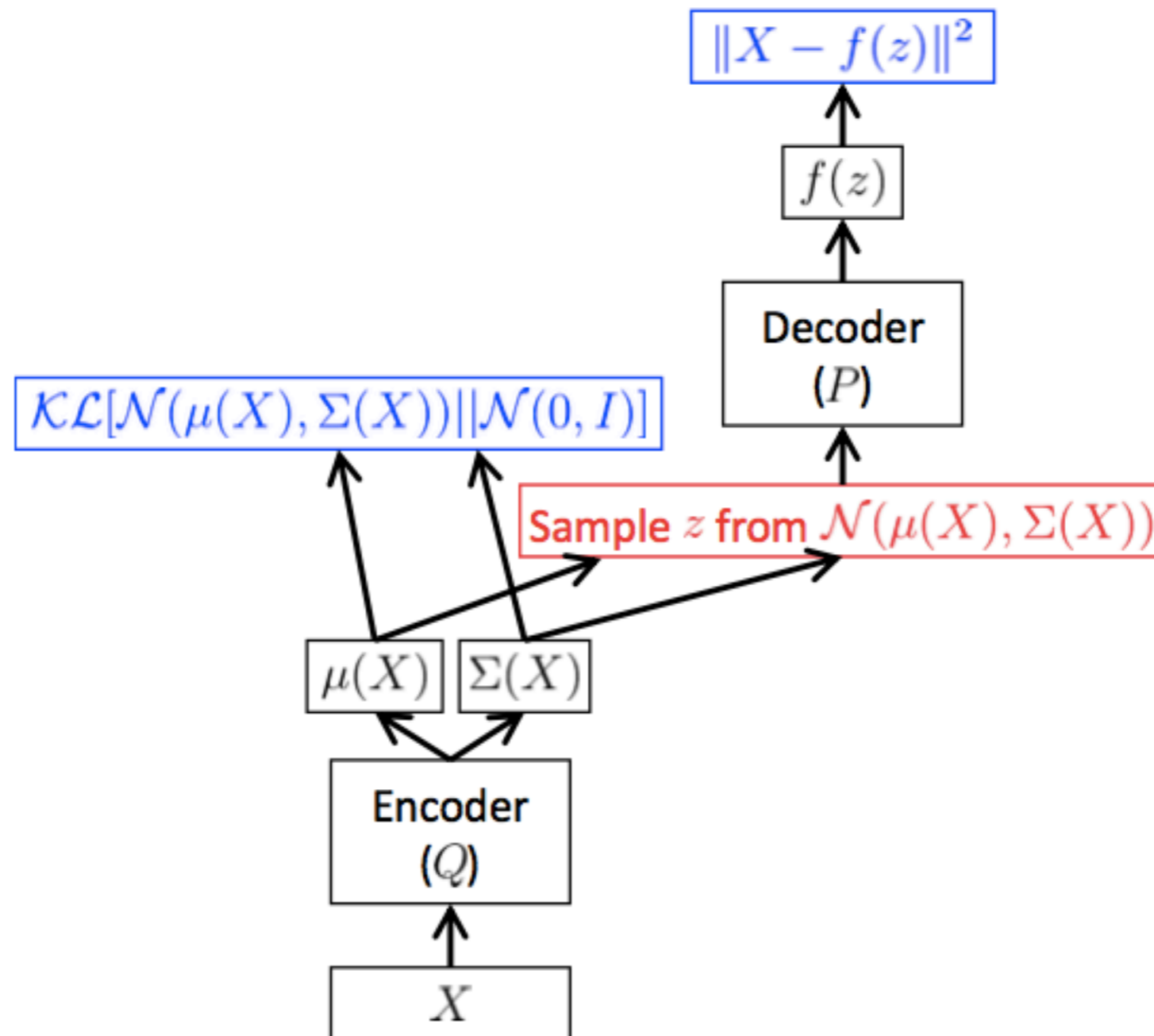
# Training of VAE

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



# Problem!

## Sampling Breaks Backprop



# Solution: Re-parameterization Trick

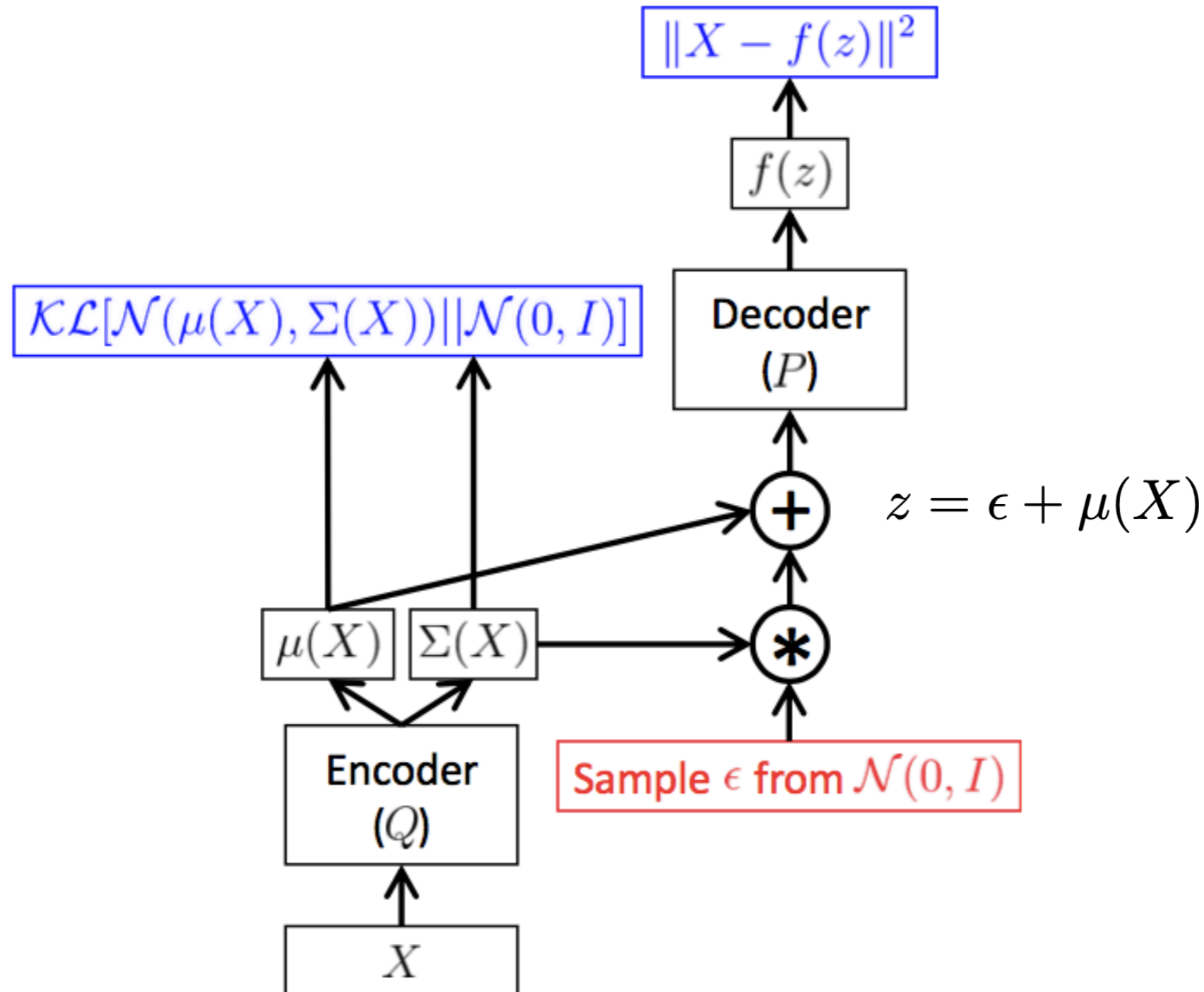


Figure Credit: Doersch (2016)

# An Example: Generating Sentences w/ Variational Autoencoders



# Generating from Language Models

- **Remember:** using ancestral sampling, we can generate from a normal language model

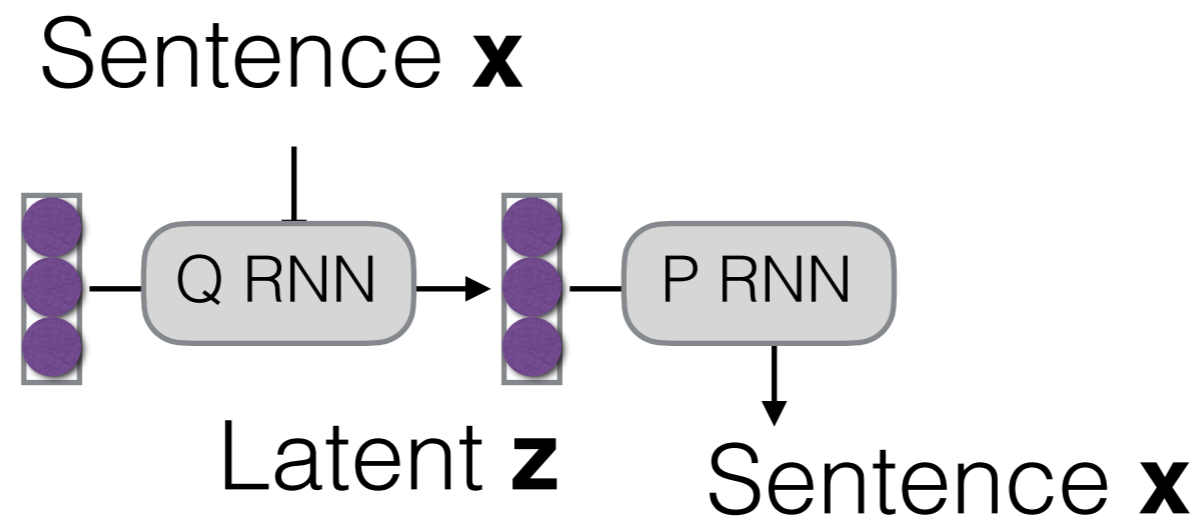
```
while  $x_{j-1} \neq \text{"</s>"}$ :  
   $x_j \sim P(x_j \mid x_1, \dots, x_{j-1})$ 
```

- We can also generate conditioned on something  $P(\mathbf{y} \mid \mathbf{x})$  (e.g. translation, image captioning)

```
while  $y_{j-1} \neq \text{"</s>"}$ :  
   $y_j \sim P(y_j \mid X, y_1, \dots, y_{j-1})$ 
```

# Generating Sentences from a Continuous Space (Bowman et al. 2015)

- The VAE-based approach is conditional language model that conditions on a latent variable  $\mathbf{z}$
- Like an encoder-decoder, but latent representation is latent variable, input and output are identical



# Motivation for Latent Variables

- Allows for a **consistent latent space** of sentences?
  - e.g. interpolation between two sentences

## Standard encoder-decoder

---

**i went to the store to buy some groceries .**  
*i store to buy some groceries .*  
*i were to buy any groceries .*  
*horses are to buy any groceries .*  
*horses are to buy any animal .*  
*horses the favorite any animal .*  
*horses the favorite favorite animal .*  
**horses are my favorite animal .**

---

## VAE

---

**“ i want to talk to you . ”**  
*“i want to be with you . ”*  
*“i do n’t want to be with you . ”*  
*i do n’t want to be with you .*  
**she did n’t want to be with him .**

---

**he was silent for a long moment .**  
*he was silent for a moment .*  
*it was quiet for a moment .*  
*it was dark and cold .*  
*there was a pause .*  
**it was my turn .**

---

- **More robust to noise?** VAE can be viewed as standard model + regularization.

# Difficulties in Training

- Of the two components in the VAE objective, the KL divergence term is much easier to learn!

$$\underbrace{\mathbb{E}_{z \sim Q(z|x)} [\log P(x | z)]}_{\text{Requires good generative model}} - \underbrace{\mathcal{KL}[Q(z | x) || P(z)]}_{\text{Just need to set the mean/variance of Q to be same as P}}$$

Requires good  
generative model

Just need to  
set the mean/variance  
of Q to be same as P

- Results in the model learning to rely solely on decoder and ignore latent variable ( $P(x|z) = P(x)$ )  
-> **Posterior Collapse**

# Solution 1:

## KL Divergence Annealing

- Basic idea: Multiply KL term by a constant  $\lambda$  starting at zero, then gradually increase to 1
- Result: model can learn to use  $\mathbf{z}$  before getting penalized

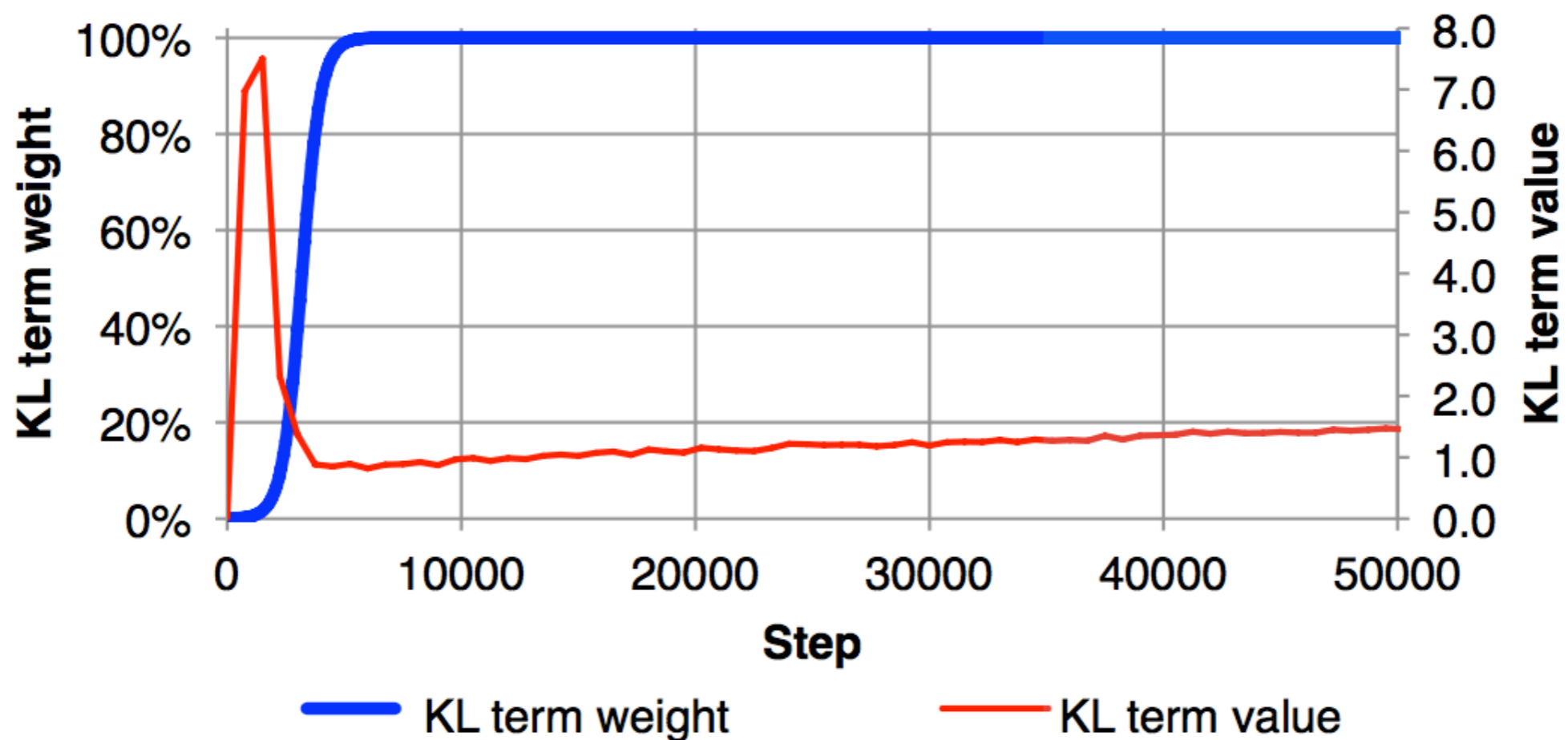


Figure Credit: Bowman et al. (2017)

# Solution 2:

## Free bits / KL thresholding

- Free bits replaces the KL term in ELBO with a hinge loss that maximize each component of the original KL with a constant:

$$\sum_i \max[\lambda, D_{\text{KL}}(q_\phi(z_i|x) || p(z_i))]$$

- $\lambda$  : Target rate

# Solution 3: Weaken the Decoder

- But theoretically still problematic: it can be shown that the optimal strategy is to ignore  $\mathbf{z}$  when it is not necessary (Chen et al. 2017)
- Solution: weaken decoder  $P(\mathbf{x}|\mathbf{z})$  so using  $\mathbf{z}$  is essential
  - Use word dropout to occasionally skip inputting previous word in  $\mathbf{x}$  (Bowman et al. 2015)
  - Use a convolutional decoder w/ limited context (Yang et al. 2017)

# Solution 4: Aggressive Inference Network Learning

$$\begin{array}{ccc} \max_{\theta, \phi} & \underbrace{\log p_{\theta}(\mathbf{x})}_{\text{marginal log data likelihood}} & - \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})|p_{\theta}(\mathbf{z}|\mathbf{x}))}_{\text{agreement between approximate and model posteriors}} \\ & & \downarrow \\ \max_{\theta} \max_{\phi} & \underbrace{\log p_{\theta}(\mathbf{x})}_{\text{marginal log data likelihood}} & - \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})|p_{\theta}(\mathbf{z}|\mathbf{x}))}_{\text{agreement between approximate and model posteriors}} \end{array}$$

(He et al. 2019)



# Handling Discrete Latent Variables

# Discrete Latent Variables?

- Many variables are better treated as discrete
  - Part-of-speech of a word
  - Class of a question
  - Writer traits (left-handed or right-handed, etc.)
- How do we handle these?

# Method 1: Enumeration

- For discrete variables, our integral is a sum

$$P(\mathbf{x}; \theta) = \sum_{\mathbf{z}} P(\mathbf{x} | \mathbf{z}; \theta) P(\mathbf{z})$$

- If the number of possible configurations for  $\mathbf{z}$  is small, we can just sum over all of them

# Method 2: Sampling

- Randomly sample a subset of configurations of  $\mathbf{z}$  and optimize with respect to this subset
- Various flavors:
  - Minimum risk training
  - Maximize ELBO loss
- Score function gradient estimator - Policy Gradient Method
  - Unbiased estimator but high variance - need to control variance

# Method 3: Reparameterization

(Maddison et al. 2017, Jang et al. 2017)

- Reparameterization also possible for discrete variables!

## Original Categorical Sampling Method:

$$\hat{z} = \text{cat-sample}(P(\mathbf{z} | \mathbf{x}))$$

## Reparameterized Method

$$\hat{z} = \text{argmax}(\log P(\mathbf{z} | \mathbf{x}) + \text{Gumbel}(0,1))$$

where the Gumbel distribution is

$$\text{Gumbel}(0, 1) = -\log(-\log(\text{Uniform}(0,1)))$$

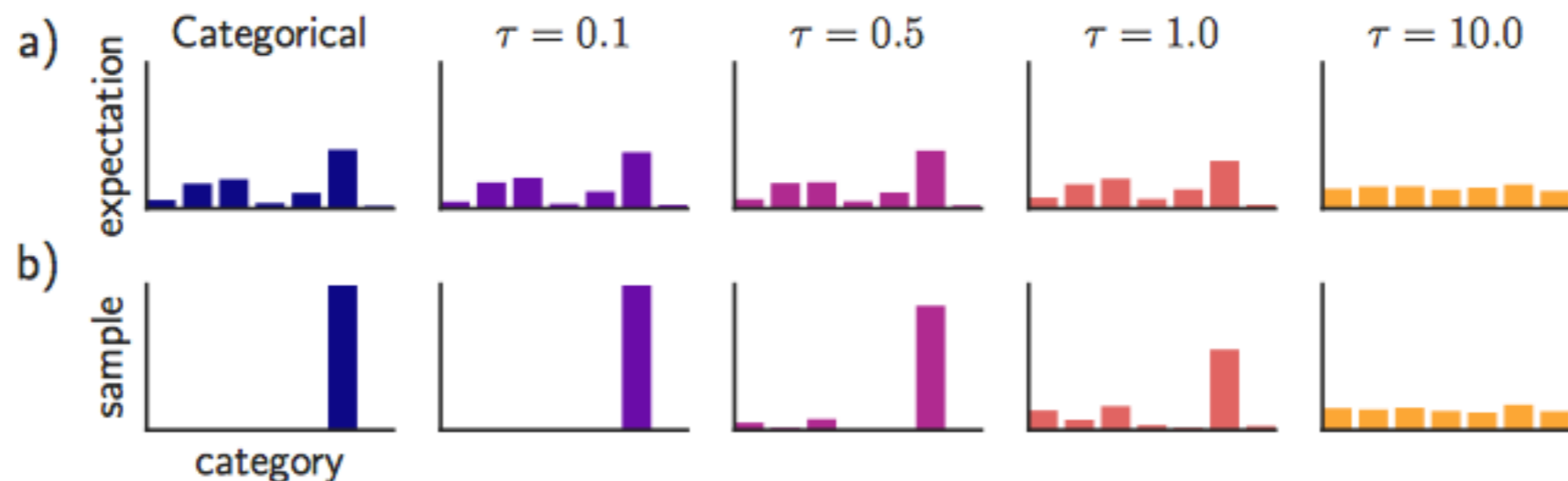
- Backprop is still not possible, due to argmax

# Gumbel-Softmax

- A way to soften the decision and allow for continuous gradients
- Instead of argmax, take softmax with temperature  $\tau$

$$\hat{z} = \text{softmax}((\log P(\mathbf{z} | \mathbf{x}) + \text{Gumbel}(0,1))^{1/\tau})$$

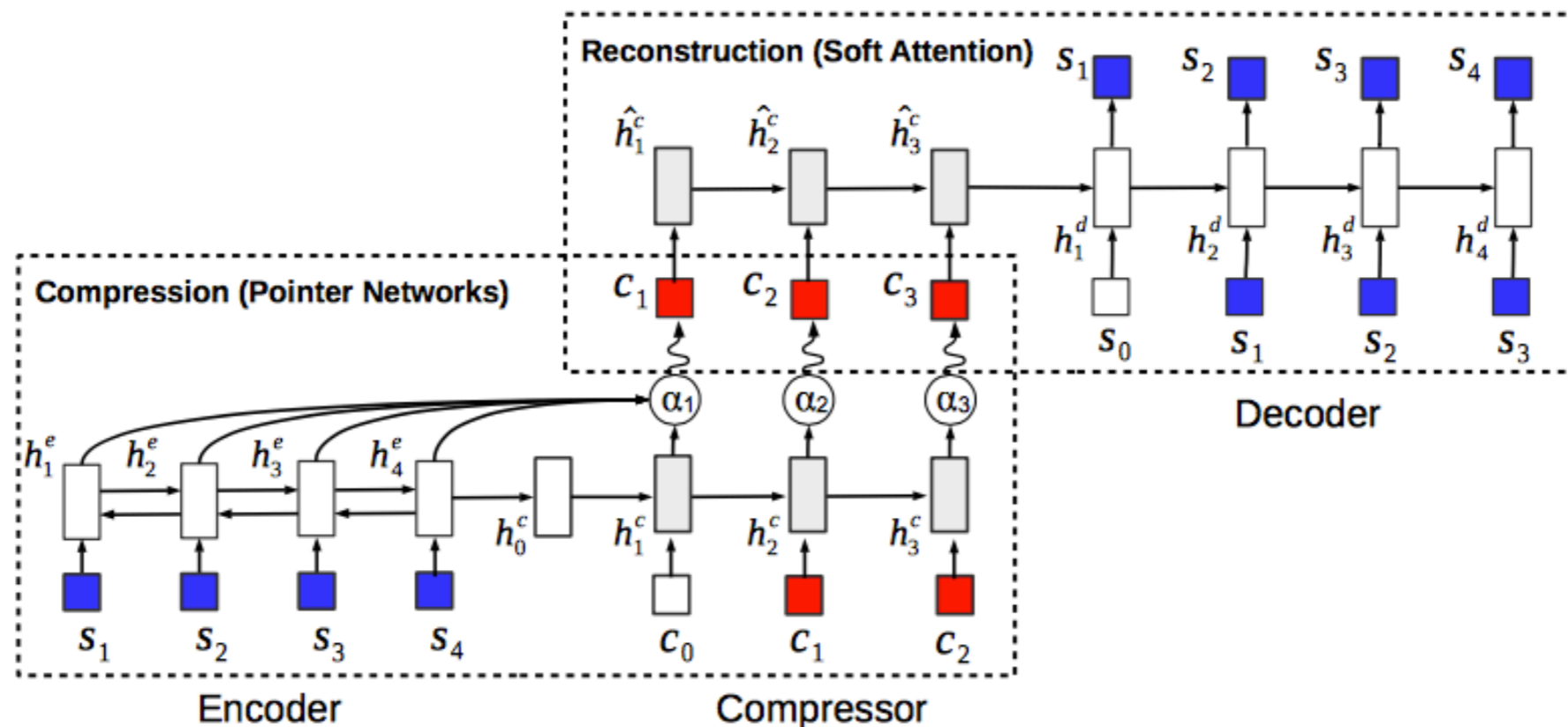
- As  $\tau$  approaches 0, will approach max



# Application Examples in NLP

# Symbol Sequence Latent Variables (Miao and Blunsom 2016)

- Encoder-decoder with a sequence of latent symbols

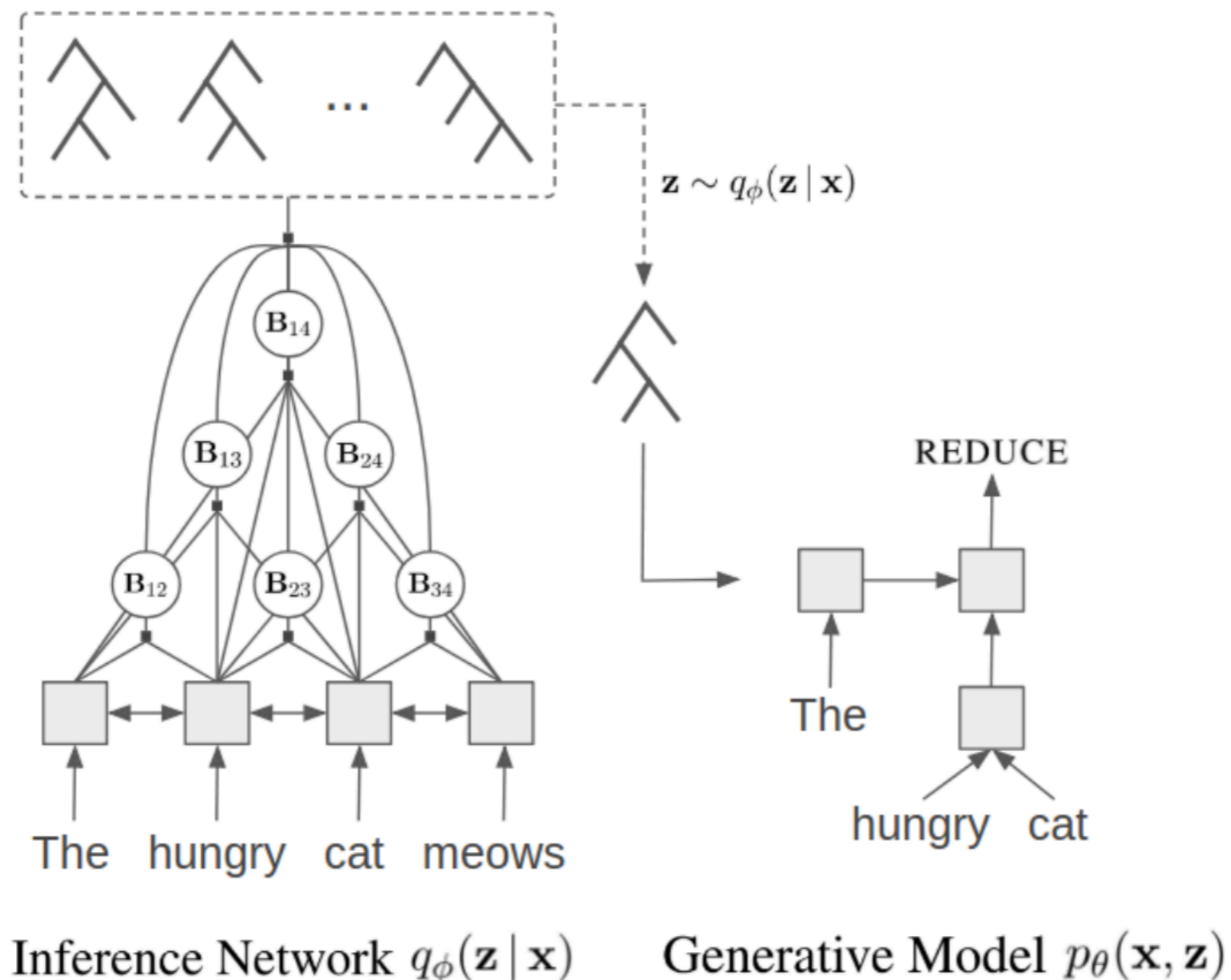


- Summarization in Miao and Blunsom (2016)
- Attempts to “discover” language (e.g. Havrylov and Titov 2017)
  - But things may not be so simple! (Kottur et al. 2017)

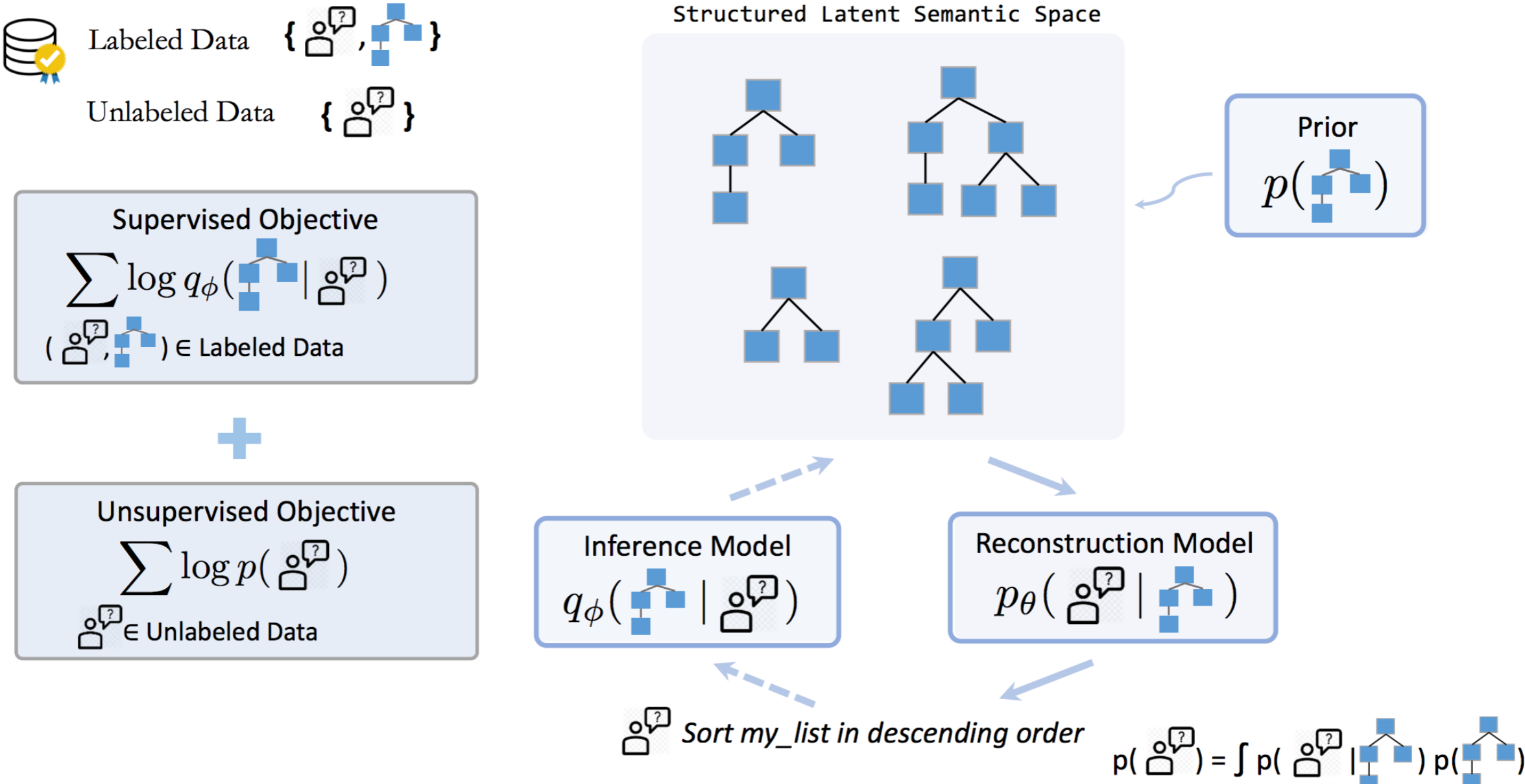


# Unsupervised Recurrent Neural Network Grammars

(Kim et al., 2019)



# STRUCTVAE: Tree-structured Latent Variable Models for Semi-supervised Semantic Parsing (Yin et al. 2018)



Questions?