

# **Variational AutoEncoder (VAE), Generative Adversarial Networks (GAN)**

Shikui Tu

Department of Computer Science and  
Engineering, Shanghai Jiao Tong University

2019-05-14

---

# Auto-Encoding Variational Bayes

---

**Diederik P. Kingma**

Machine Learning Group  
Universiteit van Amsterdam  
dpkingma@gmail.com

**Max Welling**

Machine Learning Group  
Universiteit van Amsterdam  
welling.max@gmail.com

## Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

## Auto-Encoding Variational Bayes

<https://arxiv.org> › stat ▾

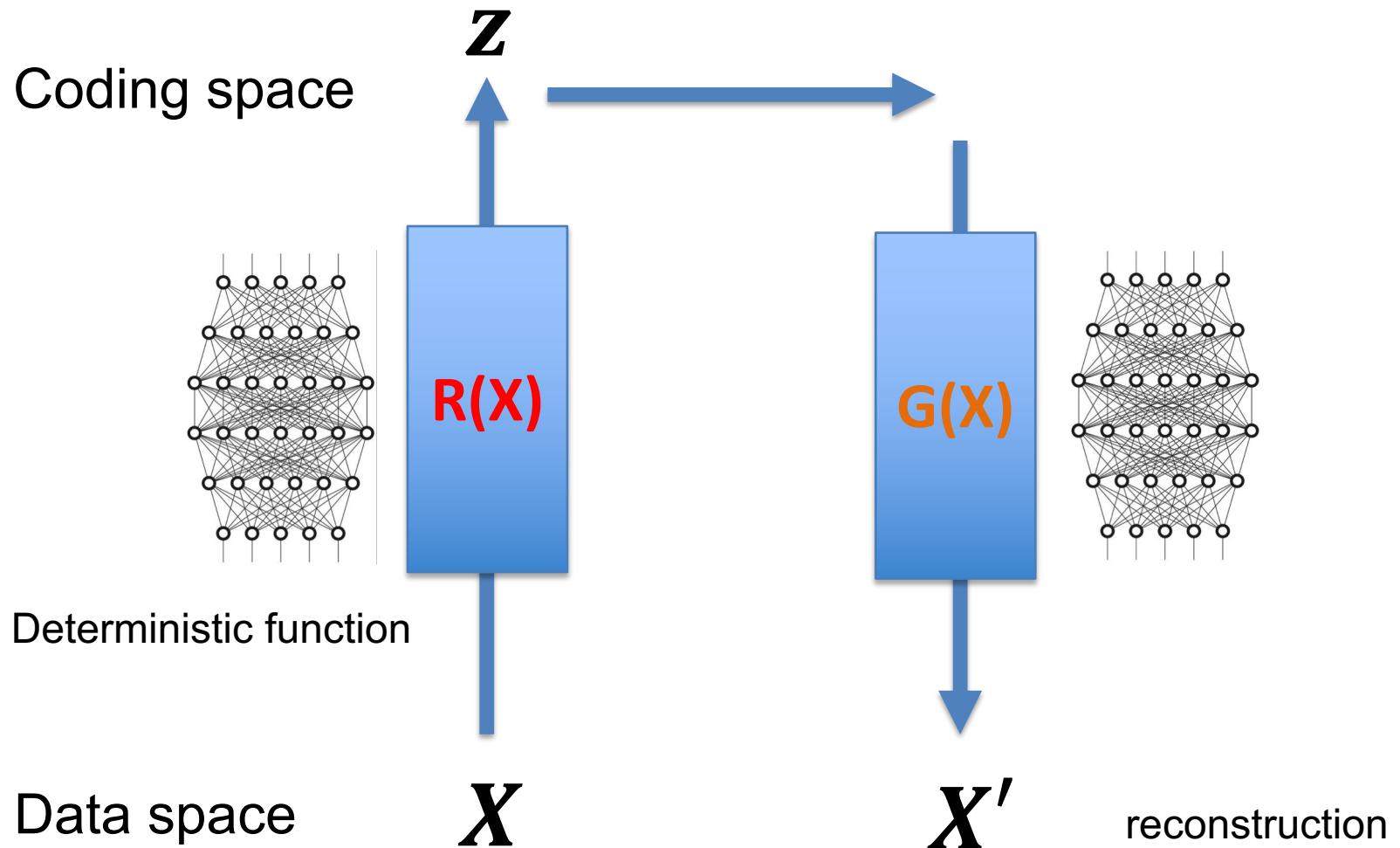
by DP Kingma - 2013 - Cited by 4621 - Related articles

May 1, 2014 - We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability ...

# Outline

- Related concepts
  - Autoencoder (AE)
  - Generative models
  - EM algorithm
  - Variational lower bound
- Variational Autoencoder (VAE)
- Generative Adversarial Networks (GAN)
- VAE vs. GAN

# Autoencoder (AE)

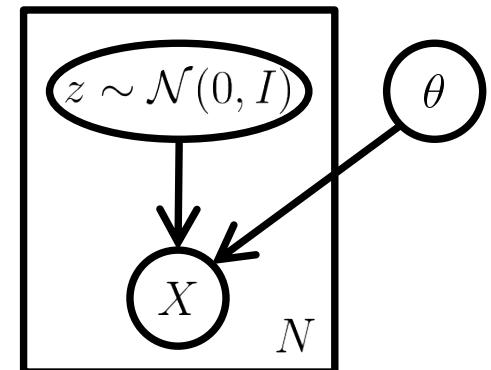


# Generative models

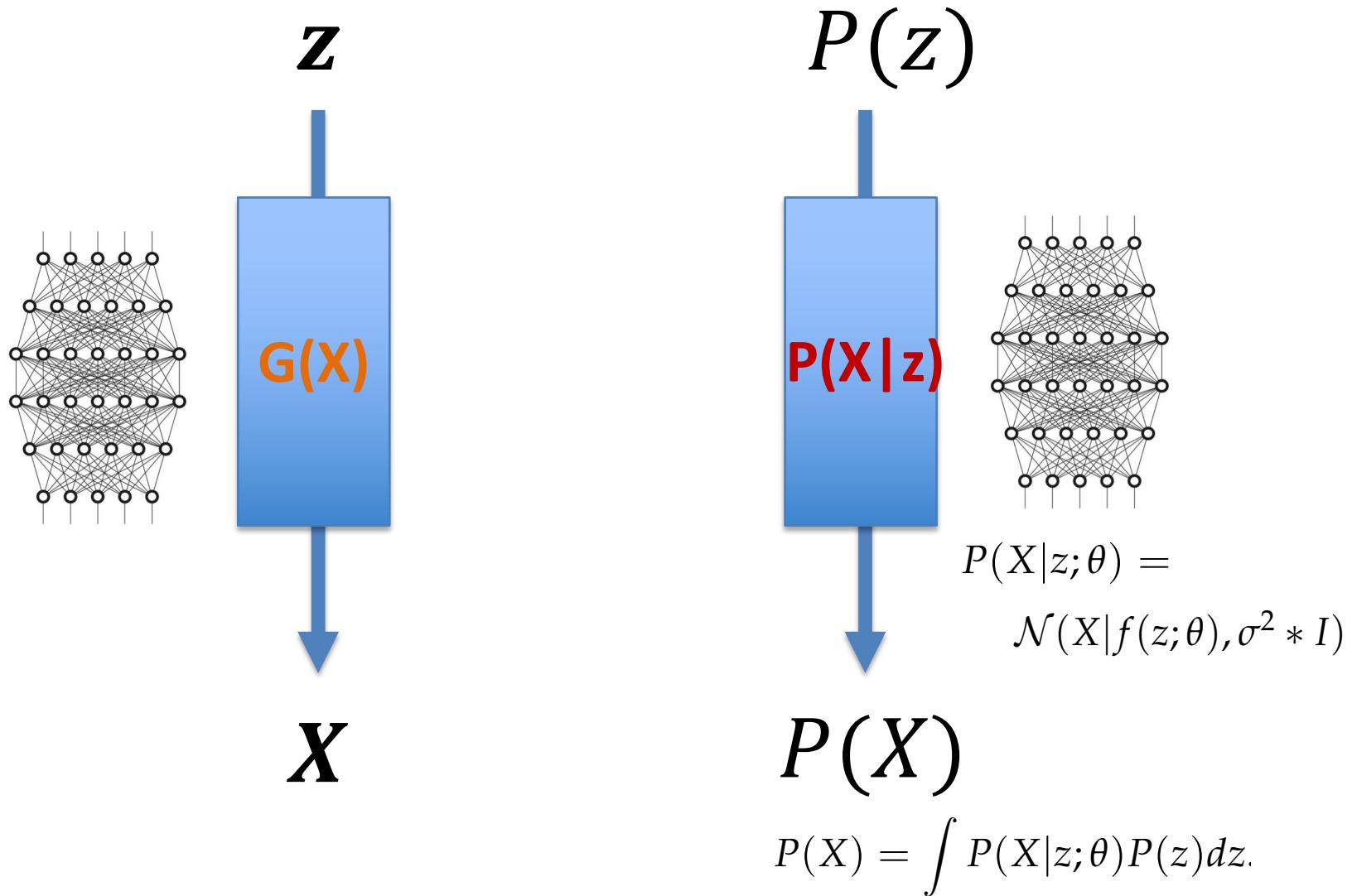
- **Task:** Given a dataset of images  $\{X_1, X_2, \dots\}$  can we learn the distribution of  $X$ ?
- Typically generative models implies modelling  $P(X)$ .
  - **Very limited, given an image the model outputs a probability**
- **More Interested** in models which we can **sample** from.
  - Can generate random examples that follow the distribution of  $P(X)$ .

# Latent variable models

- Modeling  $P(X|z; \theta)$  explicitly
- $z \sim P(z)$ , which we can sample from, such as a Gaussian distribution
- Then,  $P(X) = \int P(X|z; \theta)P(z)dz.$
- Maximum likelihood
  - Find  $\theta$  to maximize the distribution  $P(X)$



# From deterministic to probabilistic



# One way to max $P(X)$

- Approximate with samples of  $z$

$$P(X) \approx \frac{1}{n} \sum_{i=0}^n P(X|z_i)$$

- Problems
  - Need many samples of  $z$  and most of the  $P(X|z) \approx 0$
  - Not computationally tractable
- Need to learn a distribution  $Q(z)$ , where  $z \sim Q(z)$  generates  $P(X|z) \gg 0$ .

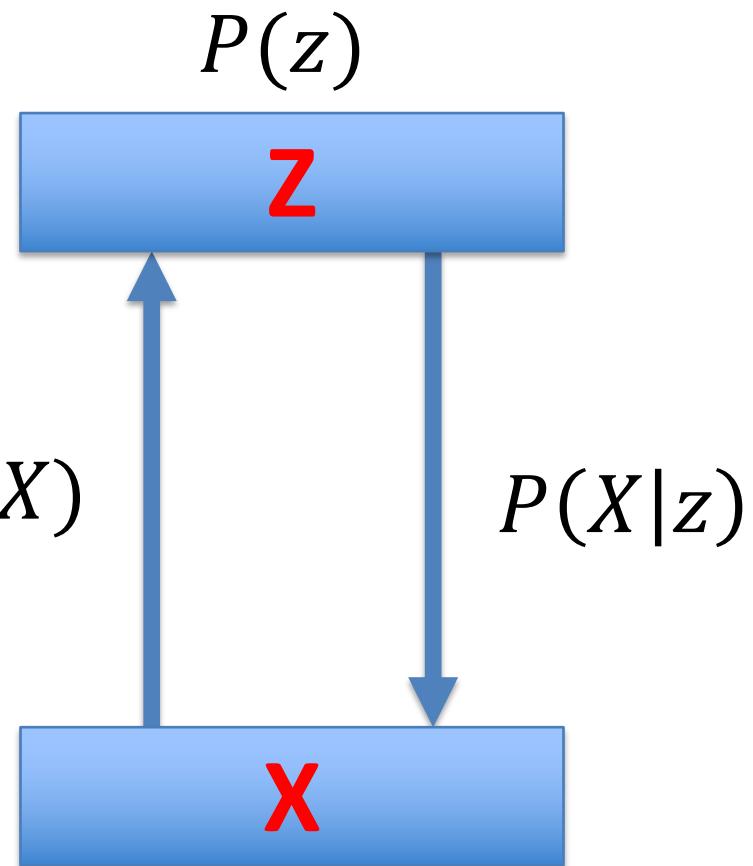
# Recall EM algorithm

**E-Step:** Compute

$$P^{old}(z|X) = \frac{P(X|z)P(z)}{P(X)}$$

**M-Step:** Update  $\Theta$  by

$$\max_{\Theta} \int P^{old}(z|X) \ln P(X|z)P(z)$$



$$P(X) = P(X|\Theta)$$

# Lower bound of likelihood

$$E[f(x)] \geq f(E[x])$$

Jensen's Inequality due to convexity

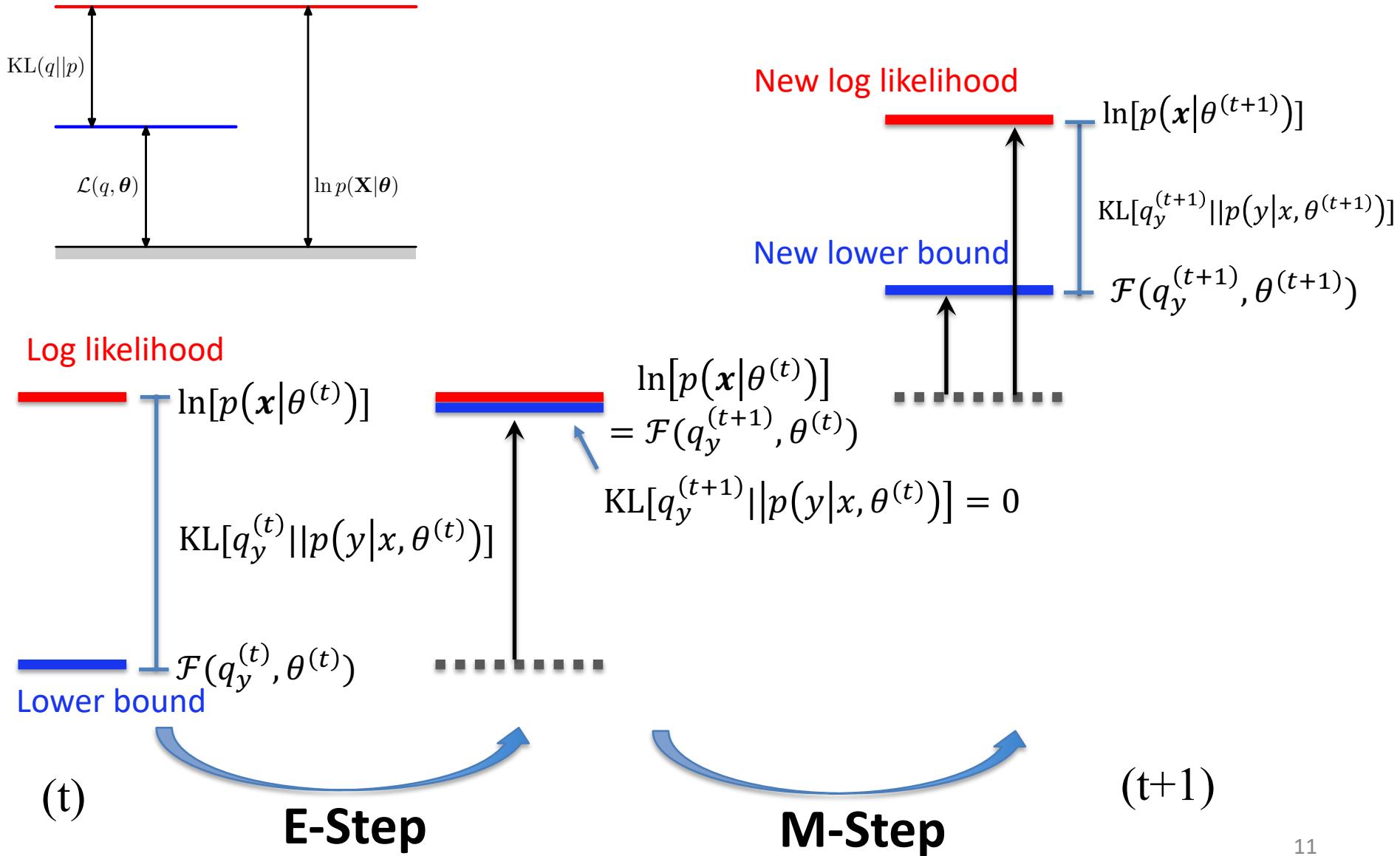
$$\log(P(\mathbf{x}|\theta)) = \log\left(\sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}|\theta)\right)$$

$$\begin{aligned}\log(P(\mathbf{x}|\theta)) &= \log\left(\sum_{\mathbf{y}} q(\mathbf{y}) \frac{P(\mathbf{x}, \mathbf{y}|\theta)}{q(\mathbf{y})}\right) \\ &\geq E_q[\log\left(\frac{P(\mathbf{x}, \mathbf{y}|\theta)}{q(\mathbf{y})}\right)] \\ &\geq E_q[\log\left(\frac{P(\mathbf{y}|\mathbf{x}, \theta)P(\mathbf{x}|\theta)}{q(\mathbf{y})}\right)] \\ &\geq E_q[\log(P(\mathbf{x}|\theta))] - E_q[\log\left(\frac{q(\mathbf{y})}{P(\mathbf{y}|\mathbf{x}, \theta)}\right)] \\ &\geq E_q[\log(P(\mathbf{x}|\theta))] - KL(q(\mathbf{y}) \| P(\mathbf{y}|\mathbf{x}, \theta)) \\ &\geq \log(P(\mathbf{x}|\theta)) - KL(q(\mathbf{y}) \| P(\mathbf{y}|\mathbf{x}, \theta))\end{aligned}$$

---

$$\begin{aligned}\log(P(\mathbf{x}|\theta)) &\geq E_q[\log\left(\frac{P(\mathbf{x}, \mathbf{y}|\theta)}{q(\mathbf{y})}\right)] \\ &\geq E_q[\log(P(\mathbf{x}, \mathbf{y}|\theta))] - E_q[\log(q(\mathbf{y}))] \\ &\geq E_q[\log(P(\mathbf{x}, \mathbf{y}|\theta))] + H(q(\mathbf{y}))\end{aligned}$$

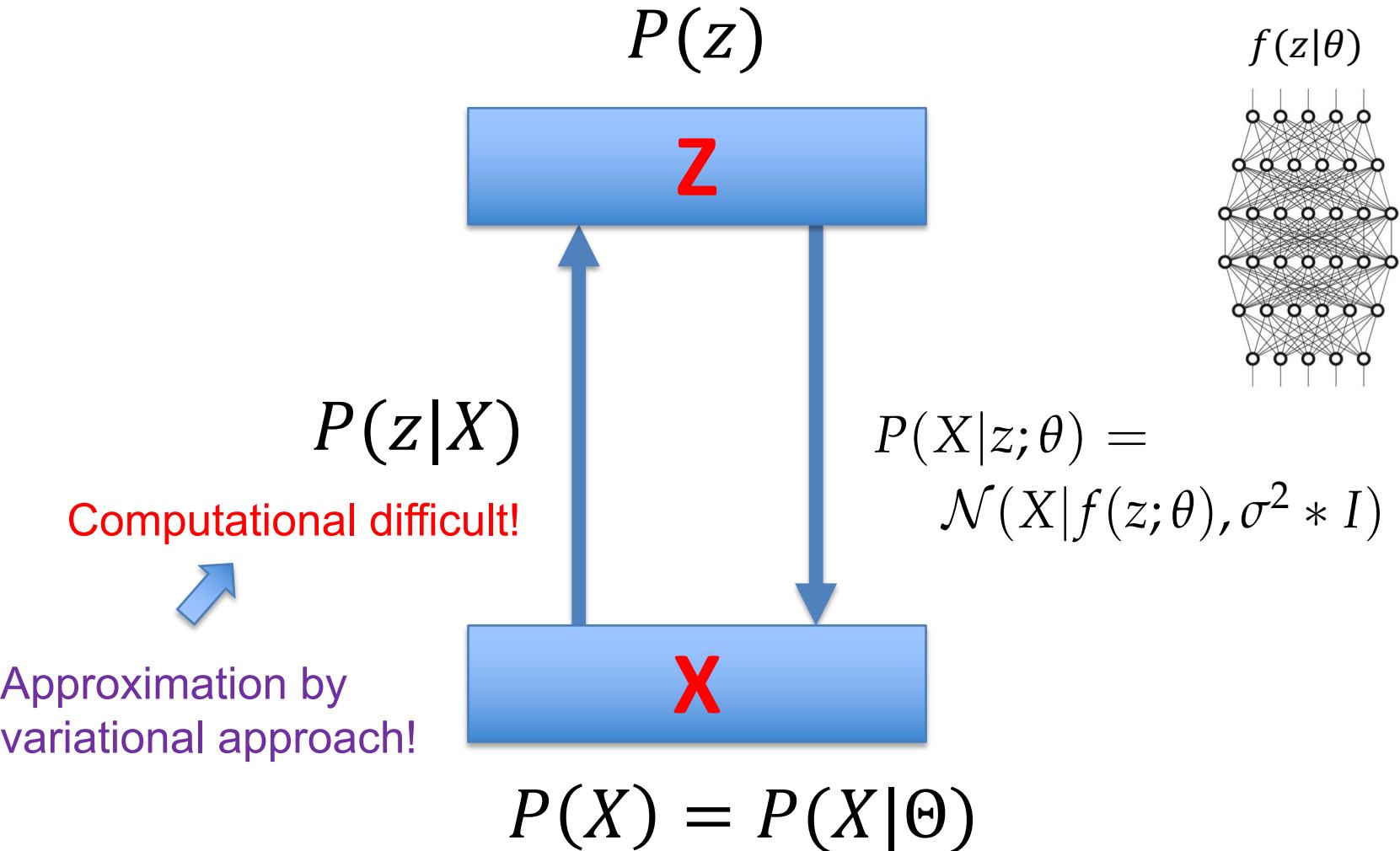
# EM never decreases the likelihood



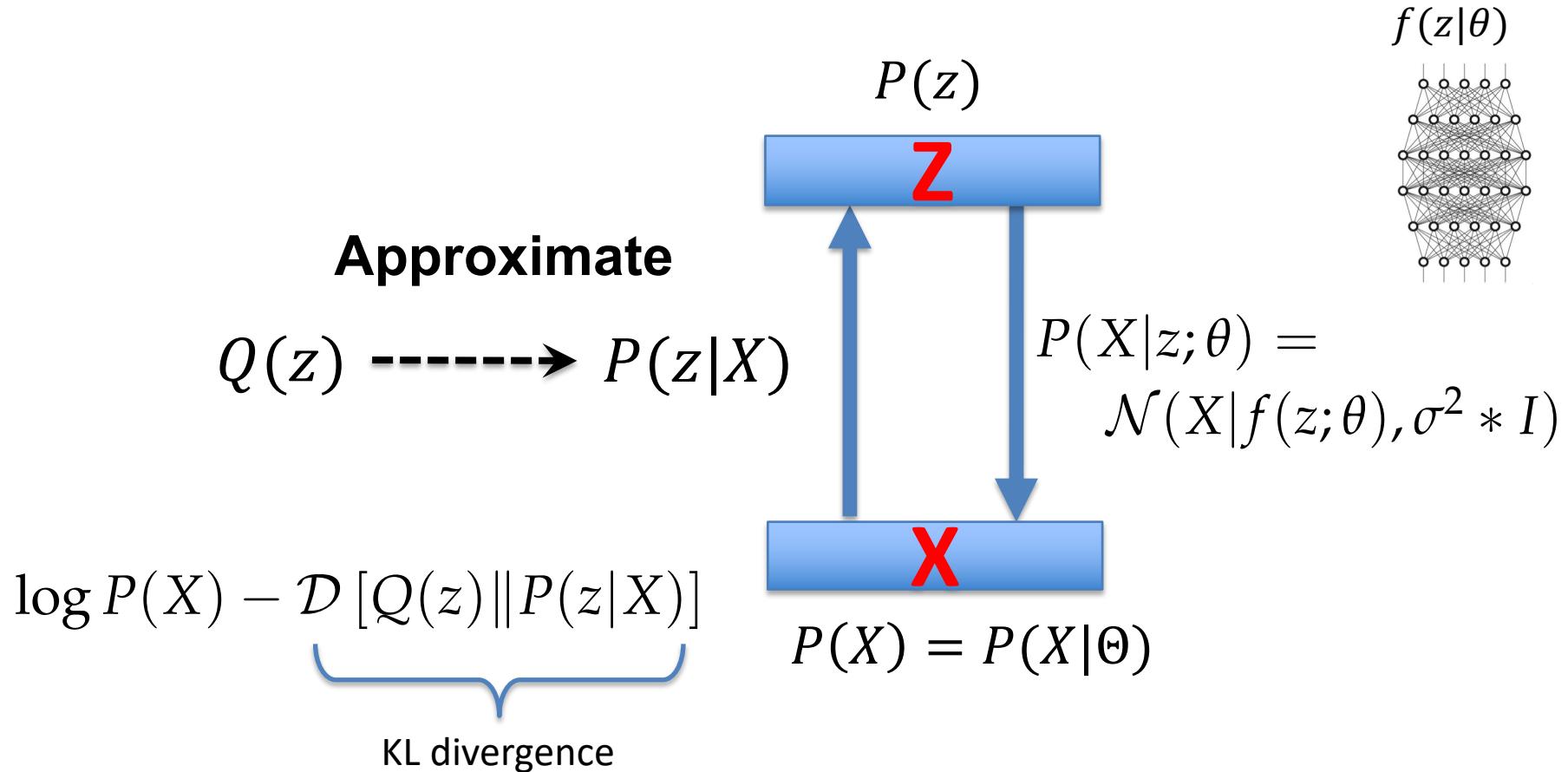
# Outline

- Related concepts
- **Variational Autoencoder (VAE)**
  - Theory and algorithm
  - Applications
  - Conditional VAE
- Generative Adversarial Networks (GAN)
- VAE vs. GAN

# Variational approximation



# Variational approximation



# Which form of $Q(z)$ is better?

$$\log P(X) - \mathcal{D}[Q(z) \| P(z|X)]$$

Note that  $X$  is fixed, and  $Q$  can be *any* distribution, not just a distribution which does a good job mapping  $X$  to the  $z$ 's that can produce  $X$ . Since we're interested in inferring  $P(X)$ , it makes sense to construct a  $Q$  which *does depend on  $X$* , and in particular, one which makes  $\mathcal{D}[Q(z) \| P(z|X)]$  small:

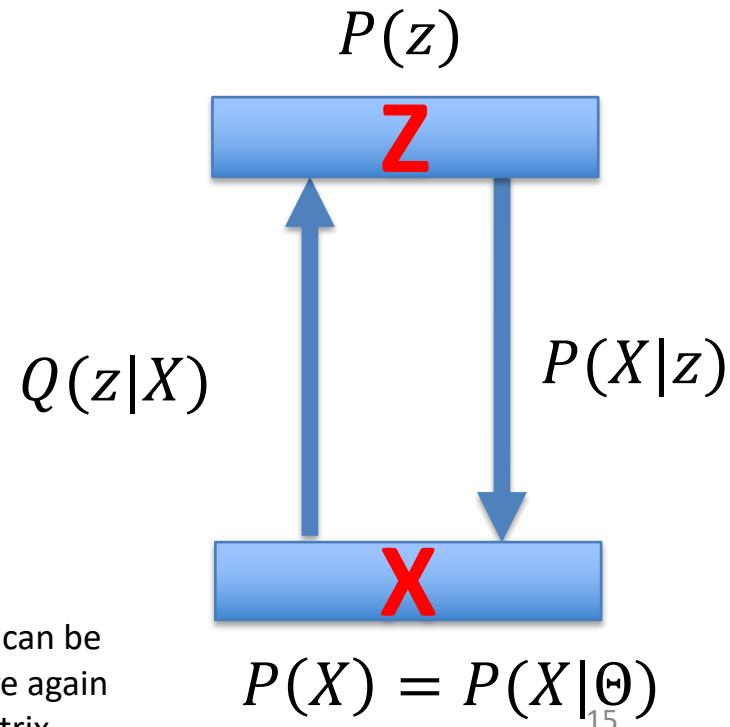
$$\log P(X) - \mathcal{D}[Q(z|X) \| P(z|X)]$$

This term will become small, if  $Q$  is **high-capacity**, making  $Q$  produce  $z$  that can reproduce a given  $X$ .



$$Q(z|X) = \mathcal{N}(z|\mu(X; \vartheta), \Sigma(X; \vartheta))$$

where  $\mu$  and  $\Sigma$  are arbitrary deterministic functions with parameters  $\vartheta$  that can be learned from data (we will omit  $\vartheta$  in later equations). In practice,  $\mu$  and  $\Sigma$  are again implemented via neural networks, and  $\Sigma$  is constrained to be a diagonal matrix.



$$P(X) = P(X|\Theta)$$

# The objective function

$$\log P(X) - \mathcal{D}[Q(z)\|P(z|X)] = E_{z\sim Q} [\log P(X|z)] - \mathcal{D}[Q(z)\|P(z)]$$

$$\log P(X) - \mathcal{D}[Q(z|X)\|P(z|X)] = E_{z\sim Q} [\log P(X|z)] - \mathcal{D}[Q(z|X)\|P(z)]$$

Derivations:

- Definition of KL divergence:

$$\mathcal{D}[Q(z)\|P(z|X)] = E_{z\sim Q} [\log Q(z) - \log P(z|X)]$$

- Apply **Bayes Rule** on  $P(z|X)$  and substitute into the equation above.

- $P(z|X) = P(X|z) P(z) / P(X)$

- $\log(P(z|X)) = \log P(X|z) + \log P(z) - \log P(X)$

- $P(X)$  does not depend on  $z$ , can take it outside of  $E_{z\sim Q}$

$$\mathcal{D}[Q(z)\|P(z|X)] = E_{z\sim Q} [\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X)$$

# Rearrange the terms

$$\mathcal{D}[Q(z) \| P(z|X)] = E_{z \sim Q} [\log Q(z) - \log P(X|z)] - \log P(z) + \log P(X)$$

Rearrange the terms:

$$E_{z \sim Q} [\log Q(z) - \log P(z)] = D [Q(z) \| P(z)]$$

$$\log P(X) - \mathcal{D}[Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - \mathcal{D}[Q(z) \| P(z)]$$

# Why is this important?

$$\log P(X) - \mathcal{D}[Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - \mathcal{D}[Q(z) \| P(z)]$$

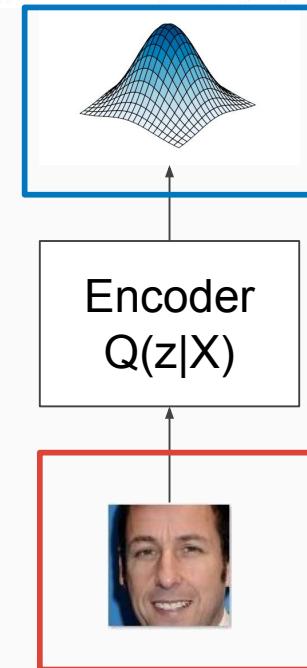
- Recall we want to **maximize  $P(X)$**  with respect to  $\theta$ , **which we cannot do.**
- KL divergence is always  $> 0$ .
- $\log P(X) > \log P(X) - D[Q(z) \| P(z|X)]$ .
- Maximize the **lower bound** instead.

# Choosing appropriate $Q$

$$\log P(X) - \mathcal{D}[Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - \mathcal{D}[Q(z) \| P(z)]$$

- $Q(z)$  or  $Q(z|X)$ ?
- Model  $Q(z|X)$  with a neural network.
- Assume  $Q(z|X)$  to be Gaussian,  $N(\mu, c \cdot I)$ 
  - Neural network **outputs** the **mean  $\mu$** , and **diagonal covariance matrix  $c \cdot I$** .
  - **Input: Image, Output: Distribution**

Let's call  $Q(z|X)$  the **Encoder**.



$$Q(z|X) = \mathcal{N}(z|\mu(X; \theta), \Sigma(X; \theta))$$

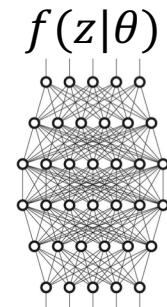
# The loss function: 1<sup>st</sup> term

$$\log P(X) - \mathcal{D}[Q(z|X)\|P(z|X)] = \boxed{E_{z \sim Q} [\log P(X|z)]} - \mathcal{D}[Q(z|X)\|P(z)]$$

The first term:

- Model  $P(X|z)$  with a neural network, let  $f(z)$  be the network output.
- Assume  $P(X|z)$  to be i.i.d. Gaussian
  - $X = f(z) + \eta$ , where  $\eta \sim N(0, I)$  \*Think Linear Regression\*
  - Simplifies to an  $L_2$  loss:  $\|X - f(z)\|^2$

$$P(X|z; \theta) = \mathcal{N}(X|f(z; \theta), \sigma^2 * I)$$



# Optimization by stochastic gradient descent

We could use sampling to estimate  $E_{z \sim Q} [\log P(X|z)]$ , but getting a good estimate would require passing many samples of  $z$  through  $f$ , which would be expensive. Hence, as is standard in stochastic gradient descent, we take one sample of  $z$  and treat  $P(X|z)$  for that  $z$  as an approximation of  $E_{z \sim Q} [\log P(X|z)]$ .

Actually, we want to optimize

$$\begin{aligned} E_{X \sim D} [\log P(X) - \mathcal{D} [Q(z|X) \| P(z|X)]] &= \\ E_{X \sim D} [E_{z \sim Q} [\log P(X|z)] - \mathcal{D} [Q(z|X) \| P(z)]] \end{aligned}$$

# The loss function: 2<sup>nd</sup> term

$$\log P(X) - \mathcal{D}[Q(z|X)\|P(z|X)] = E_{z \sim Q} [\log P(X|z)] - \boxed{\mathcal{D}[Q(z|X)\|P(z)]}$$

Assume  $P(z) = G(z|0, I)$ . Then, the second term is a KL divergence between two multivariate Gaussians.

$$\begin{aligned}\mathcal{D}[\mathcal{N}(\mu(X), \Sigma(X))\|\mathcal{N}(0, I)] &= \\ &\frac{1}{2} \left( \text{tr}(\Sigma(X)) + (\mu(X))^\top (\mu(X)) - k - \log \det(\Sigma(X)) \right)\end{aligned}$$

where  $k$  is the dimensionality of the distribution.

---

KL divergence between two multivariate Gaussians:

$$\begin{aligned}\mathcal{D}[\mathcal{N}(\mu_0, \Sigma_0)\|\mathcal{N}(\mu_1, \Sigma_1)] &= \\ &\frac{1}{2} \left( \text{tr}(\Sigma_1^{-1} \Sigma_0) + (\mu_1 - \mu_0)^\top \Sigma_1^{-1} (\mu_1 - \mu_0) - k + \log \left( \frac{\det \Sigma_1}{\det \Sigma_0} \right) \right)\end{aligned}$$

# The loss function: all together

$$\log P(X) - \mathcal{D}[Q(z) \| P(z|X)] = \boxed{E_{z \sim Q} [\log P(X|z)]} - \boxed{\mathcal{D}[Q(z) \| P(z)]}$$

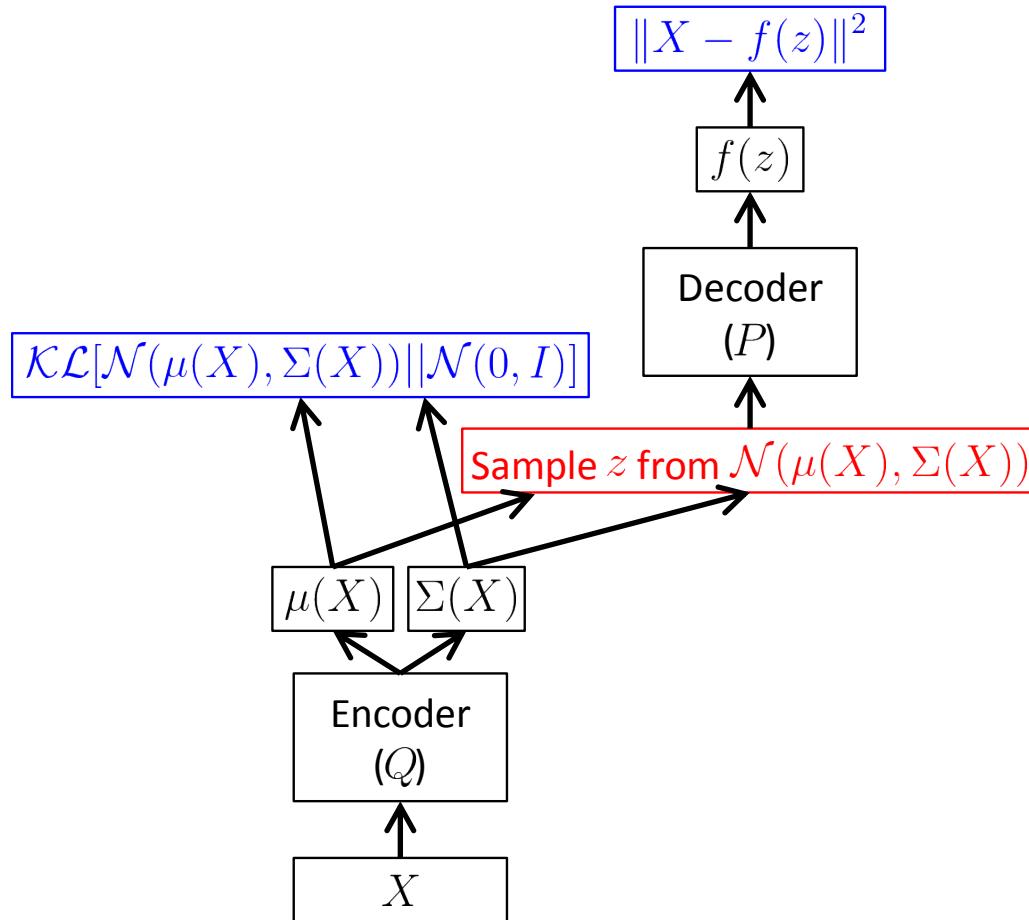
Putting it all together:  $E_{z \sim Q(z|X)} \log P(X|z) \propto ||X - f(z)||^2$

$$L = ||X - f(z)||^2 - \lambda \cdot \mathcal{D}[Q(z) \| P(z)]$$

, given a  $(X, z)$  pair.



# The whole architecture

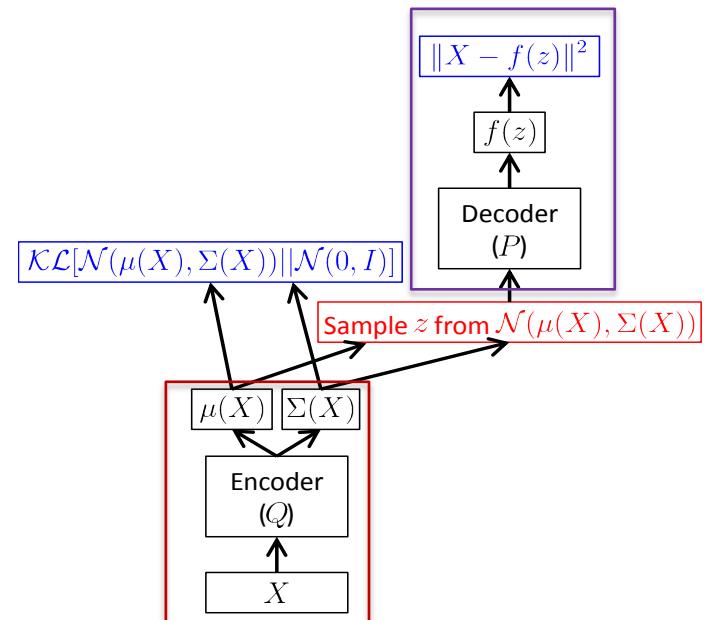


# How to train the model?

- Training the **Decoder** is easy, by standard backpropagation.



- How to train the **Encoder**?
  - How to apply gradient descent through samples?  
Not trivial.



# Reparameterization trick

- How to effectively backpropagate through the  $z$  samples to the Encoder?

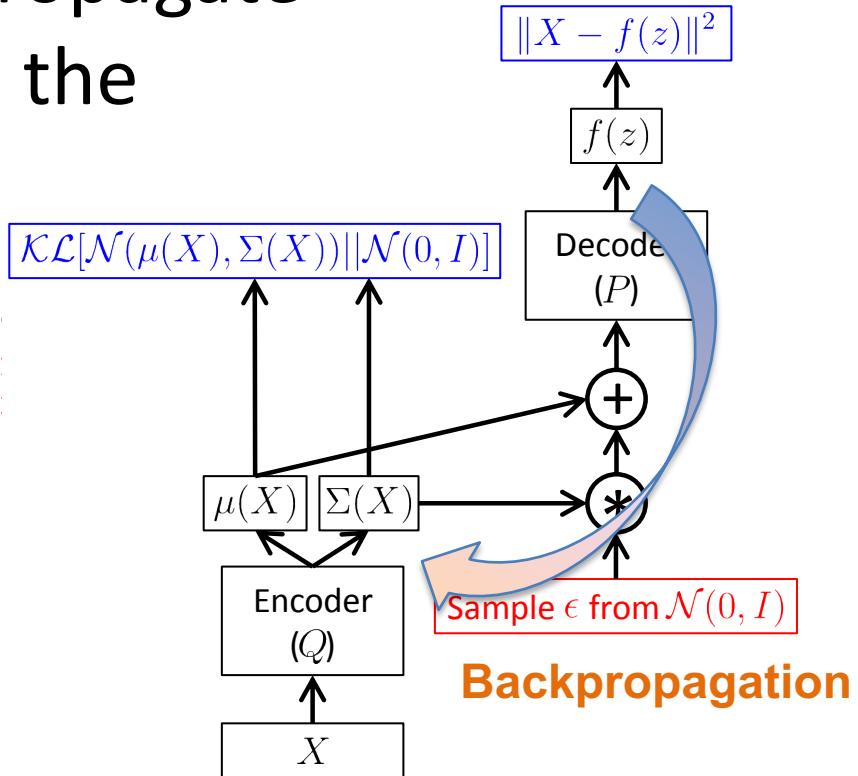
Reparameterization

$$z \sim \mathcal{N}(z|\mu, \sigma^2)$$

$\Updownarrow$  **Equivalent!**

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

$$E_{X \sim D} \left[ E_{\epsilon \sim \mathcal{N}(0,I)} [\log P(X|z = \mu(X) + \Sigma^{1/2}(X) * \epsilon)] - \mathcal{D}[Q(z|X) \| P(z)] \right]$$



Now, we can easily backpropagate the loss to the Encoder.

# The whole training process

Given a dataset of examples  $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots\}$

Initialize parameters for Encoder and Decoder

**Repeat till convergence:**

$\mathbf{X}^M \leftarrow$  Random minibatch of  $M$  examples from  $\mathbf{X}$

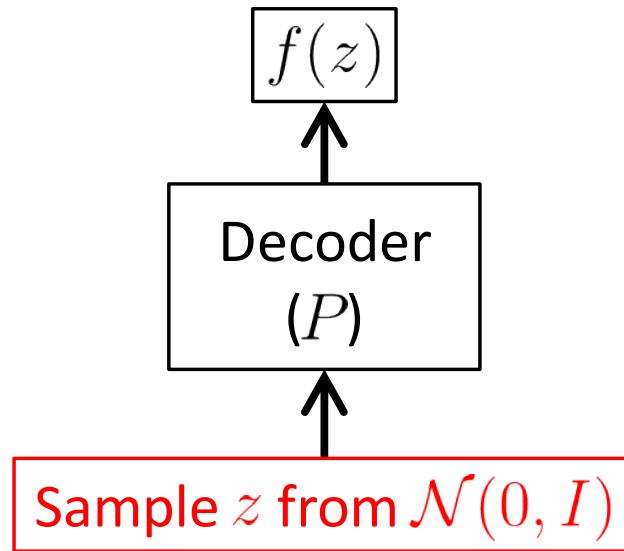
$\varepsilon \leftarrow$  Sample  $M$  noise vectors from  $N(0, I)$

Compute  $L(\mathbf{X}^M, \varepsilon, \theta)$  (i.e. run a forward pass in the neural network)

Gradient descent on  $L$  to updated Encoder and Decoder.

# Testing the learned model

At test time, when we want to generate new samples, we simply input values of  $z \sim \mathcal{N}(0, I)$  into the decoder. That is, we remove the “encoder,” including the multiplication and addition operations that would change the distribution of  $z$ .



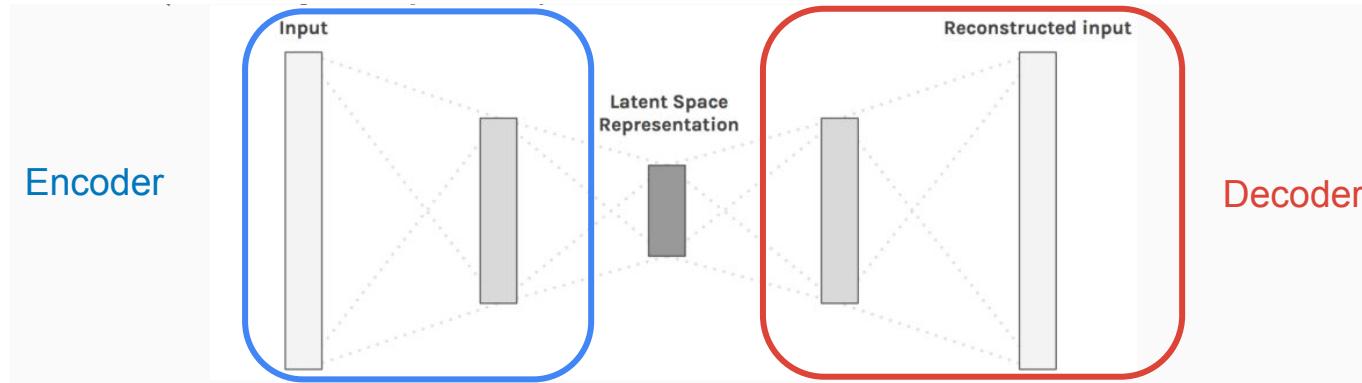
No good quantitative metric to evaluate the generated samples (e.g., images), and it relies on human visual inspection usually.

# Outline

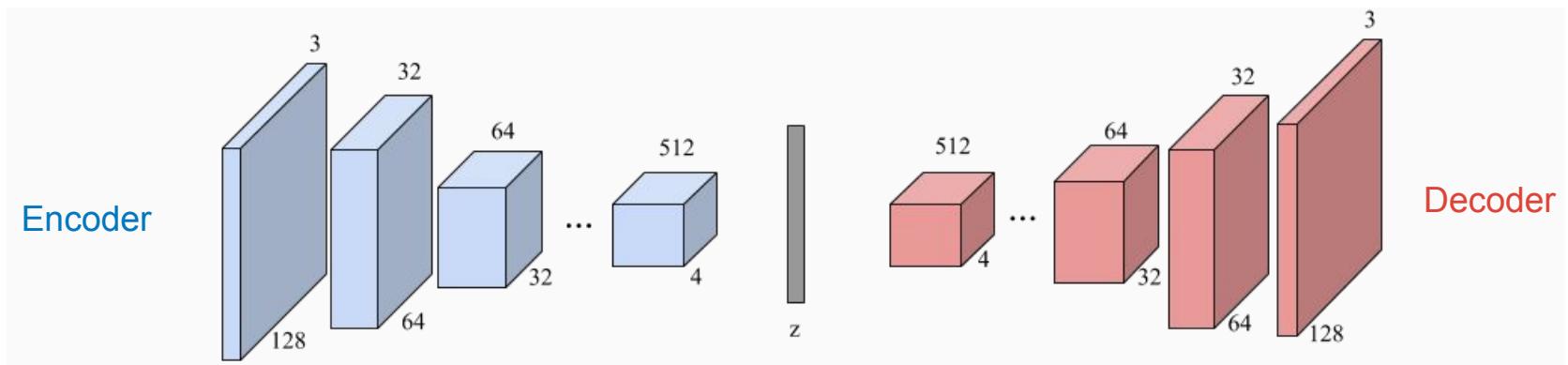
- Related concepts
- **Variational Autoencoder (VAE)**
  - Theory and algorithm
  - Applications
  - Conditional VAE
- Generative Adversarial Networks (GAN)
- VAE vs. GAN

# Common VAE architecture

- Fully connected network



- Convolution and deconvolution network



# Disentangle latent factors by VAE



Visualization of learned data manifold for generative models with two-dimensional latent space.  
[http://www.dpkingma.com/sgvb\\_mnist\\_demo/demo.html](http://www.dpkingma.com/sgvb_mnist_demo/demo.html)

# Different dimensionalities of latent space

Random samples from learned generative models of MNIST for different dimensionalities of latent space.



(a) 2-D latent space

(b) 5-D latent space

(c) 10-D latent space

(d) 20-D latent space

# Control latent factors for generation

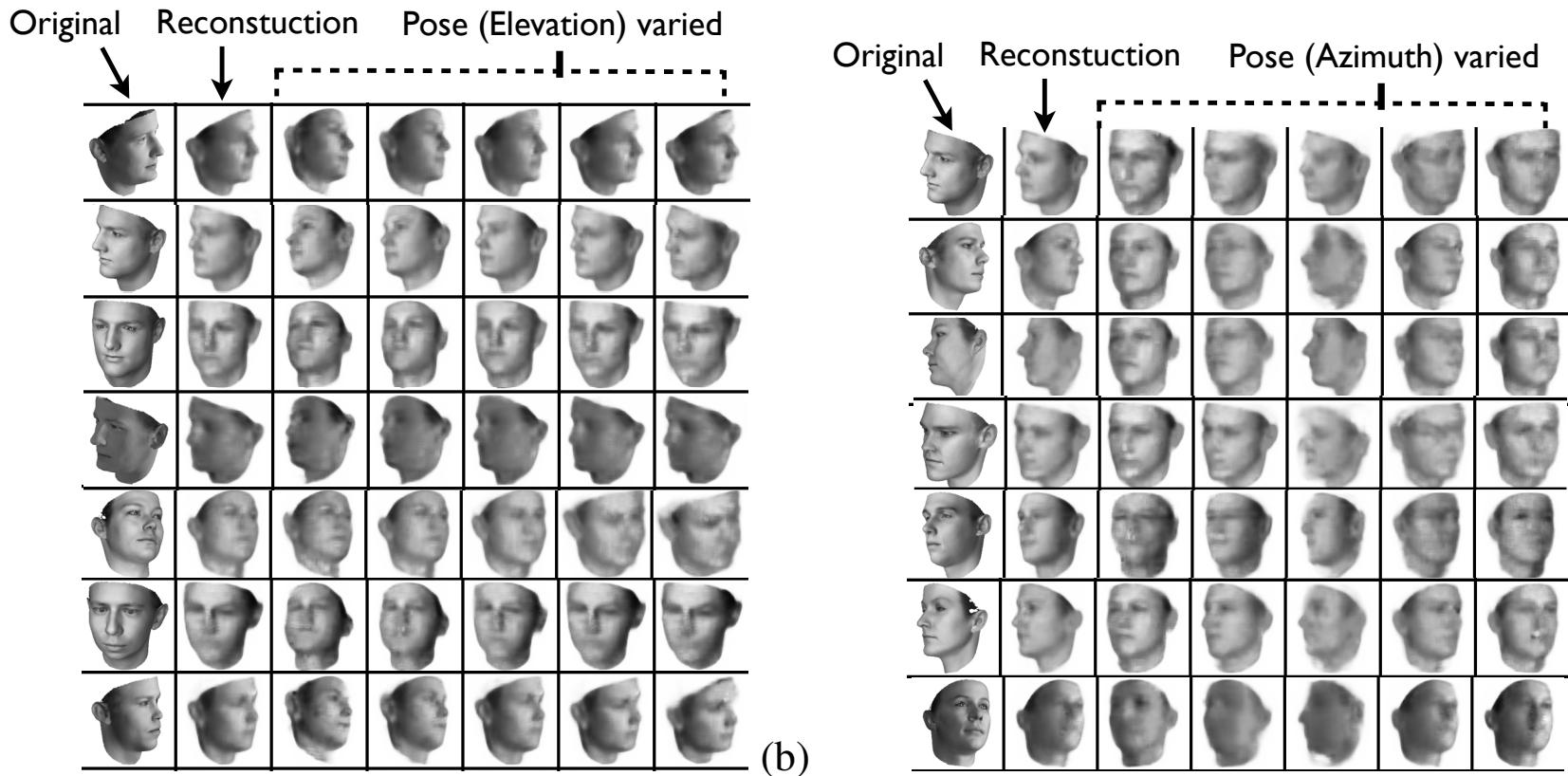


Figure 4: **Manipulating pose variables:** Qualitative results showing the generalization capability of the learned DC-IGN decoder to rerender a single input image with different pose directions. **(a)** We change the latent  $z_{elevation}$  smoothly from -15 to 15, leaving all 199 other latents unchanged. **(b)** We change the latent  $z_{azimuth}$  smoothly from -15 to 15, leaving all 199 other latents unchanged.

# Manipulating rotation

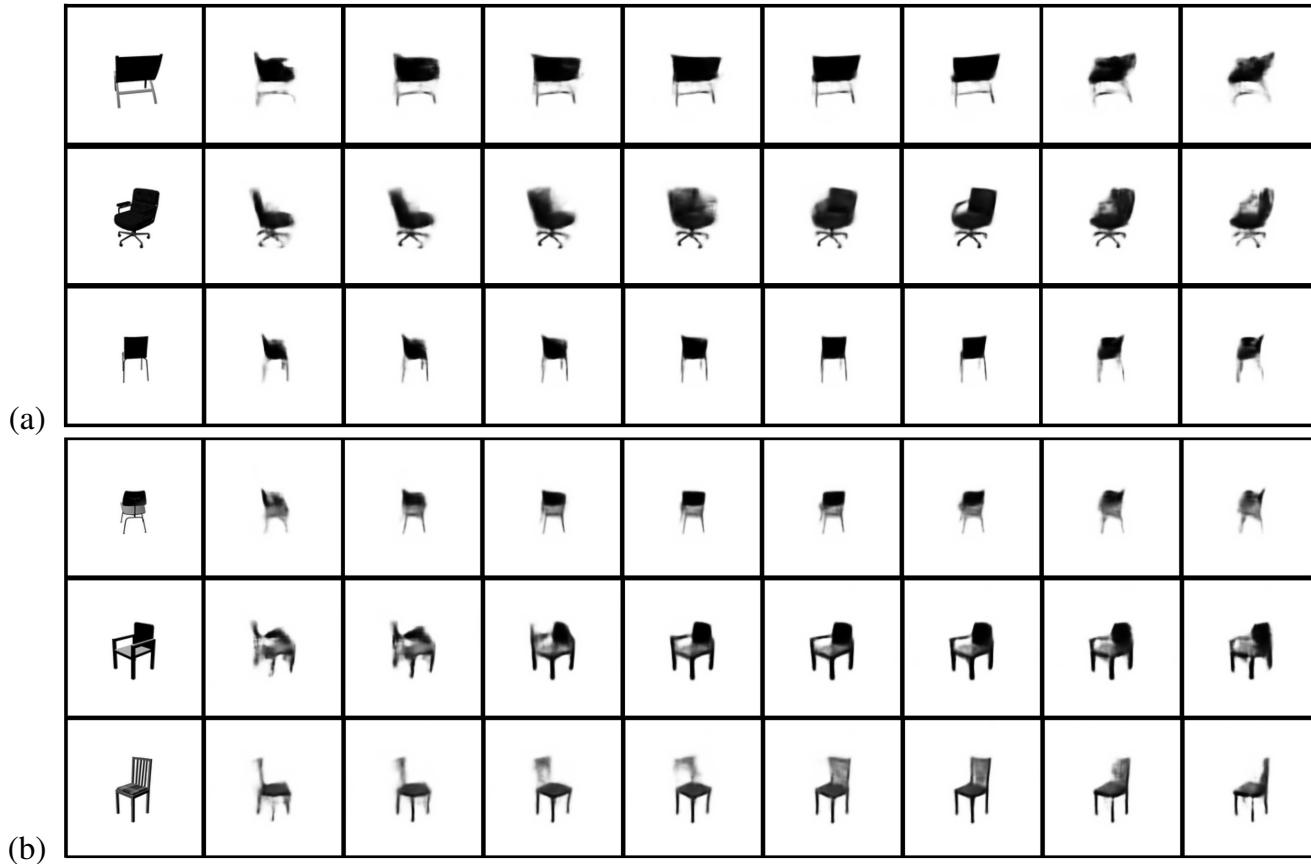


Figure 7: **Manipulating rotation:** Each row was generated by encoding the input image (leftmost) with the encoder, then changing the value of a single latent and putting this modified encoding through the decoder. The network has never seen these chairs before at any orientation. **(a)** Some positive examples. Note that the DC-IGN is making a conjecture about any components of the chair it cannot see; in particular, it guesses that the chair in the top row has arms, because it can't see that it doesn't. **(b)** Examples in which the network extrapolates to new viewpoints less accurately.

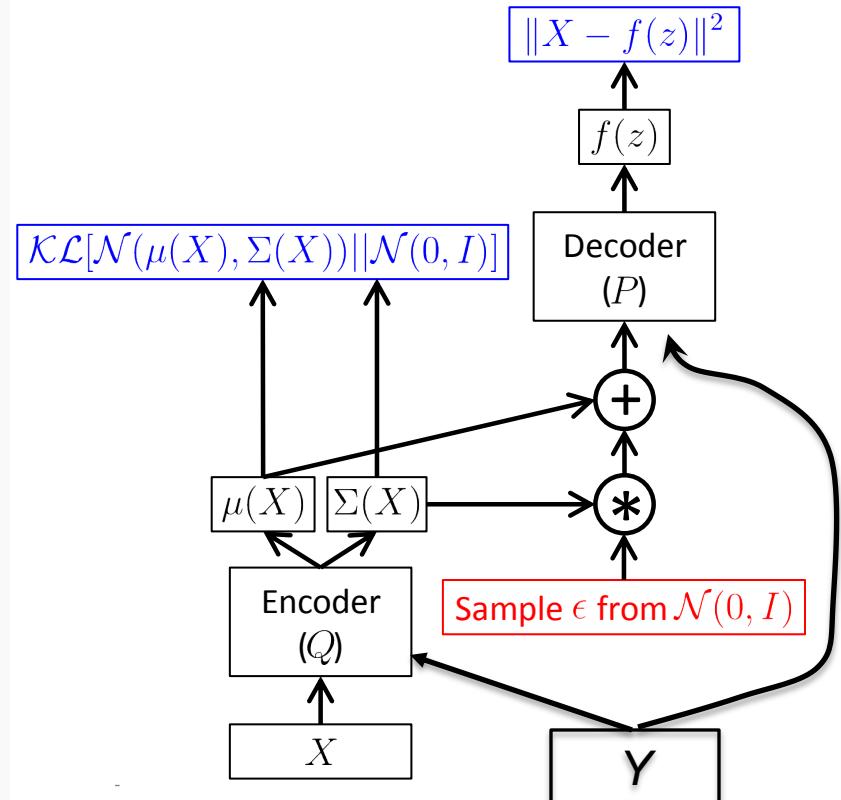
# Outline

- Related concepts
- **Variational Autoencoder (VAE)**
  - Theory and algorithm
  - Applications
  - Conditional VAE
- Generative Adversarial Networks (GAN)
- VAE vs. GAN

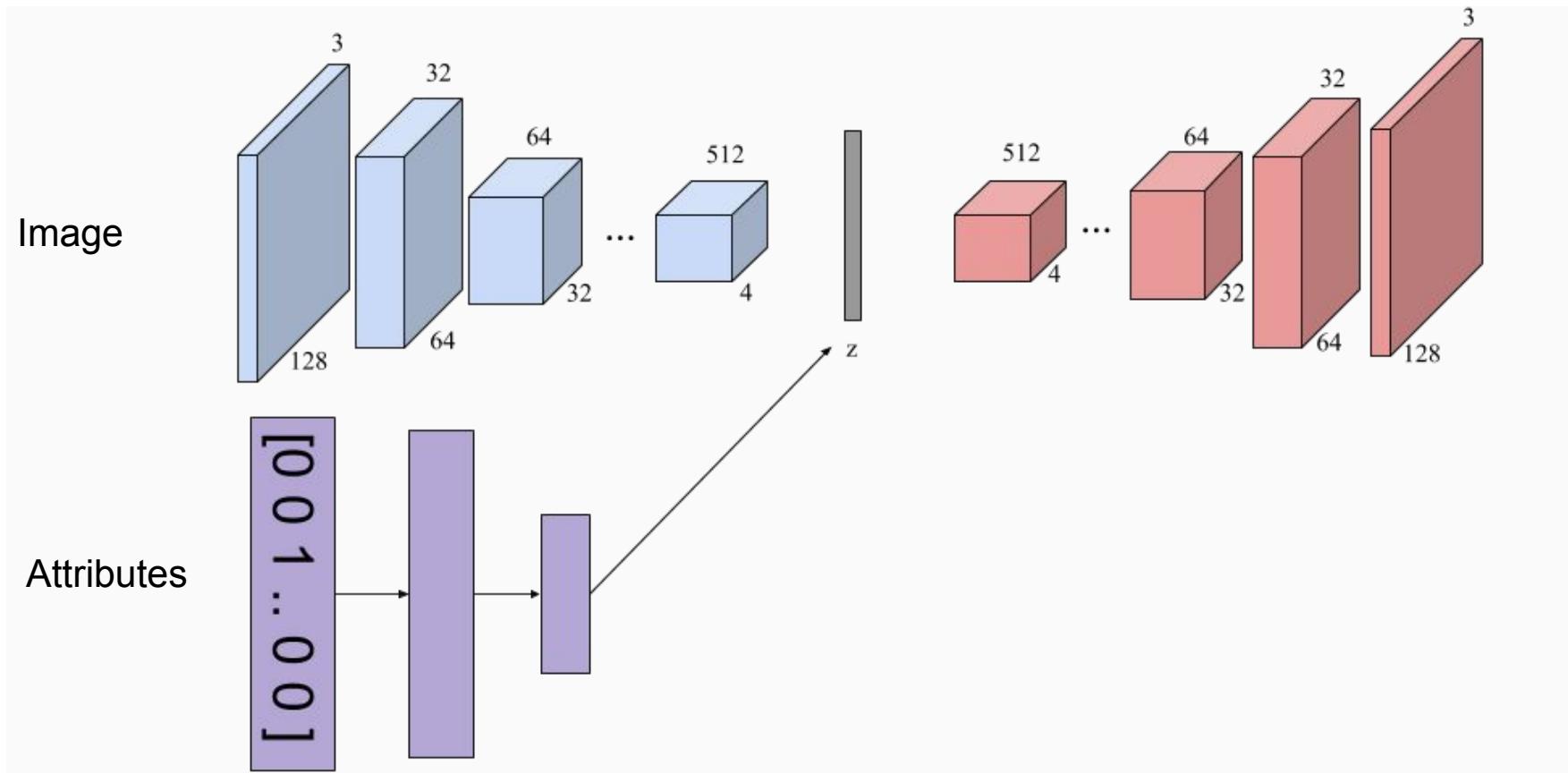
# Conditional VAE

What if we have **labels**? (e.g.  
digit labels or attributes) Or  
other inputs we wish to  
condition on (**Y**).

- **NONE** of the derivation changes.
- Replace all **P(X|z)** with **P(X|z,Y)**.
- Replace all **Q(z|X)** with **Q(z|X,Y)**.
- Go through the same KL divergence  
procedure, to get the same lower bound.

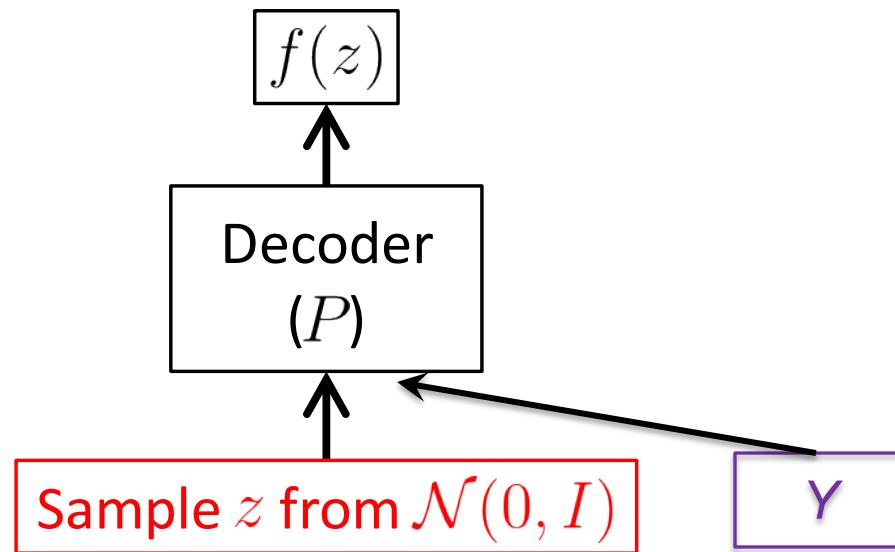


# Common architecture for CVAE

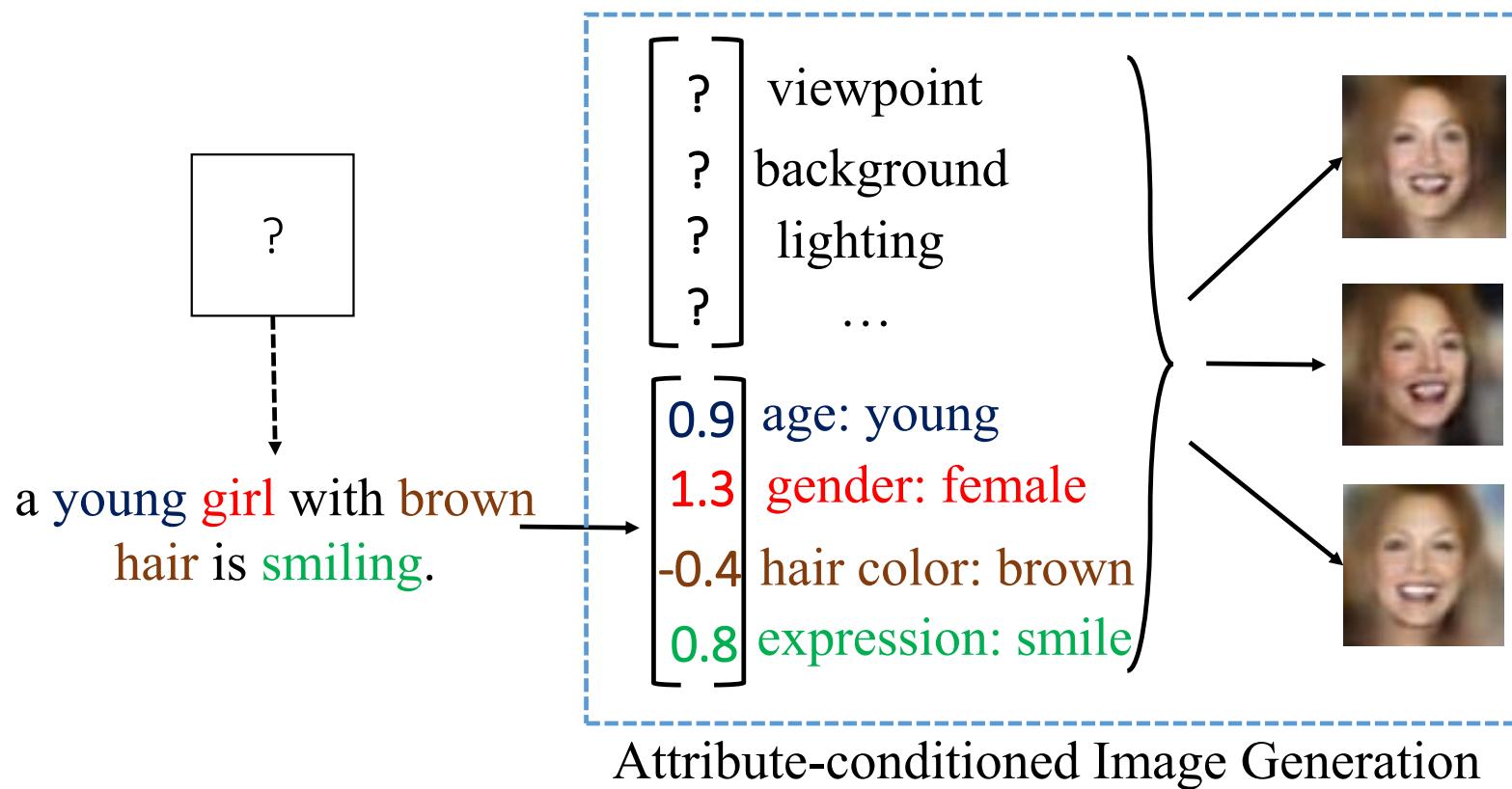


# Testing CVAE

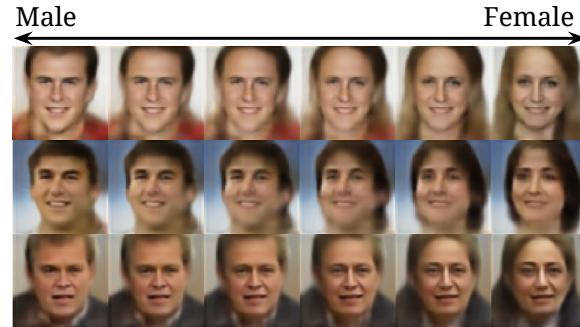
- Again, remove the Encoder as test time
- Sample  $z \sim N(0, I)$  and input a desired  $Y$  to the Decoder.



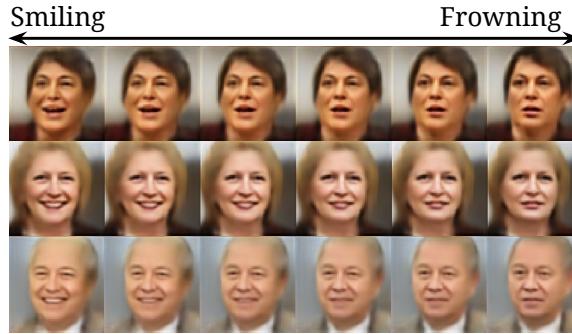
# Conditioned image generation



# Attribute-conditioned image progression



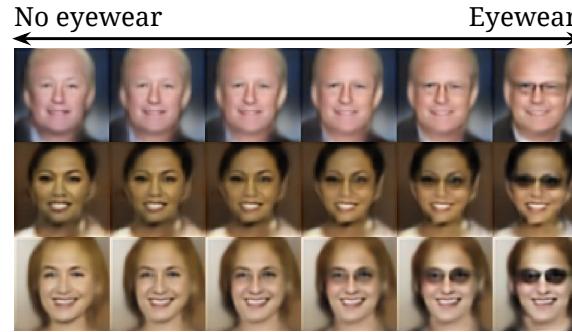
(a) progression on gender



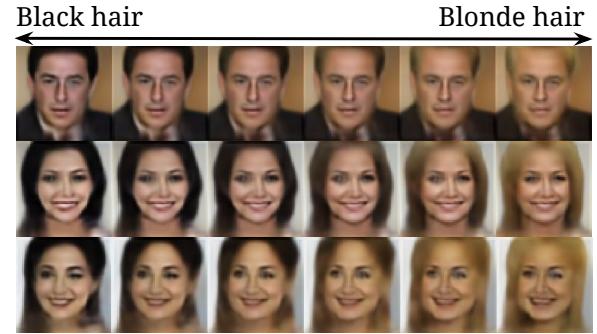
(c) progression on expression



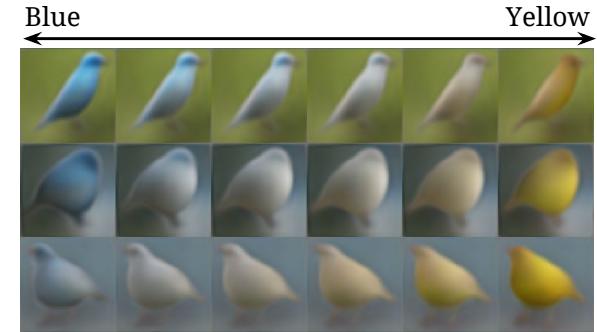
(b) progression on age



(d) progression on eyewear



(e) progression on hair color



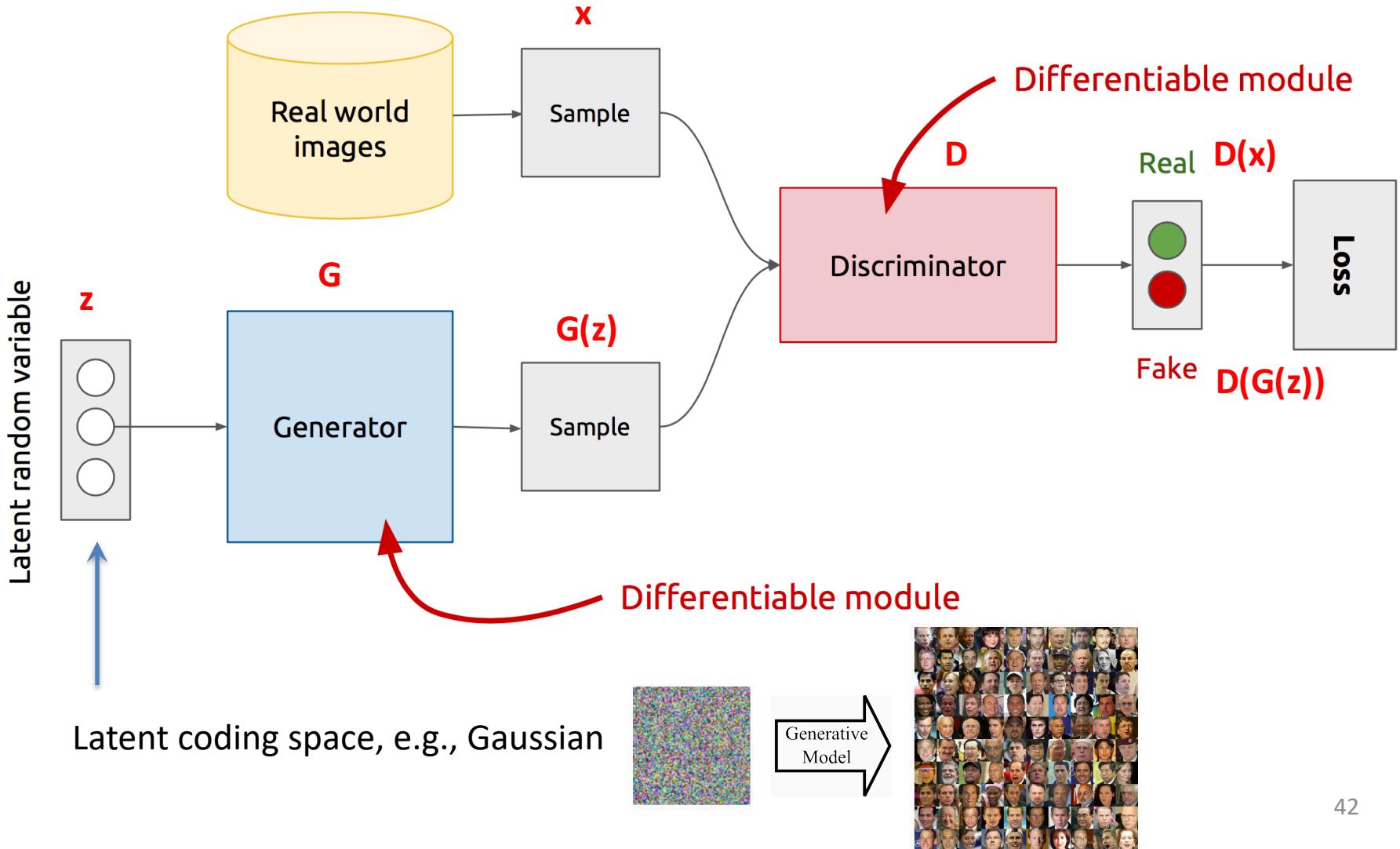
(f) progression on primary color

**Fig. 4.** Attribute-conditioned image progression. The visualization is organized into six attribute groups (e.g., “gender”, “age”, “facial expression”, “eyewear”, “hair color” and “primary color (blue vs. yellow)”). Within each group, the images are generated from  $p_\theta(x|y, z)$  with  $z \sim \mathcal{N}(0, I)$  and  $y = [y_\alpha, y_{rest}]$ , where  $y_\alpha = (1-\alpha) \cdot y_{min} + \alpha \cdot y_{max}$ . Here,  $y_{min}$  and  $y_{max}$  stands for the minimum and maximum attribute value respectively in the dataset along the corresponding dimension.

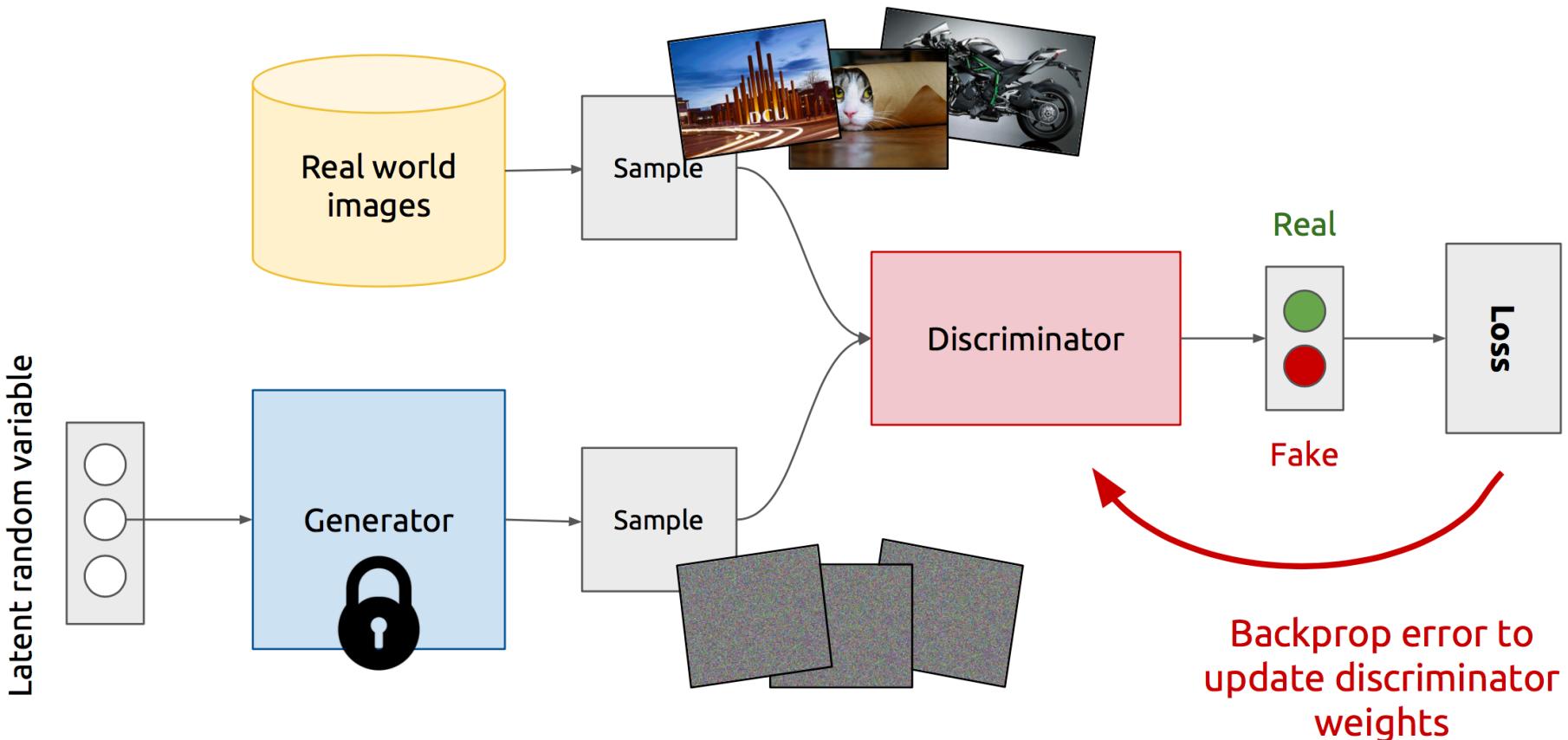
# Outline

- Related concepts
- Variational Autoencoder (VAE)
- **Generative Adversarial Networks (GAN)**
- VAE vs. GAN

# Generative Adversarial Networks (GAN)

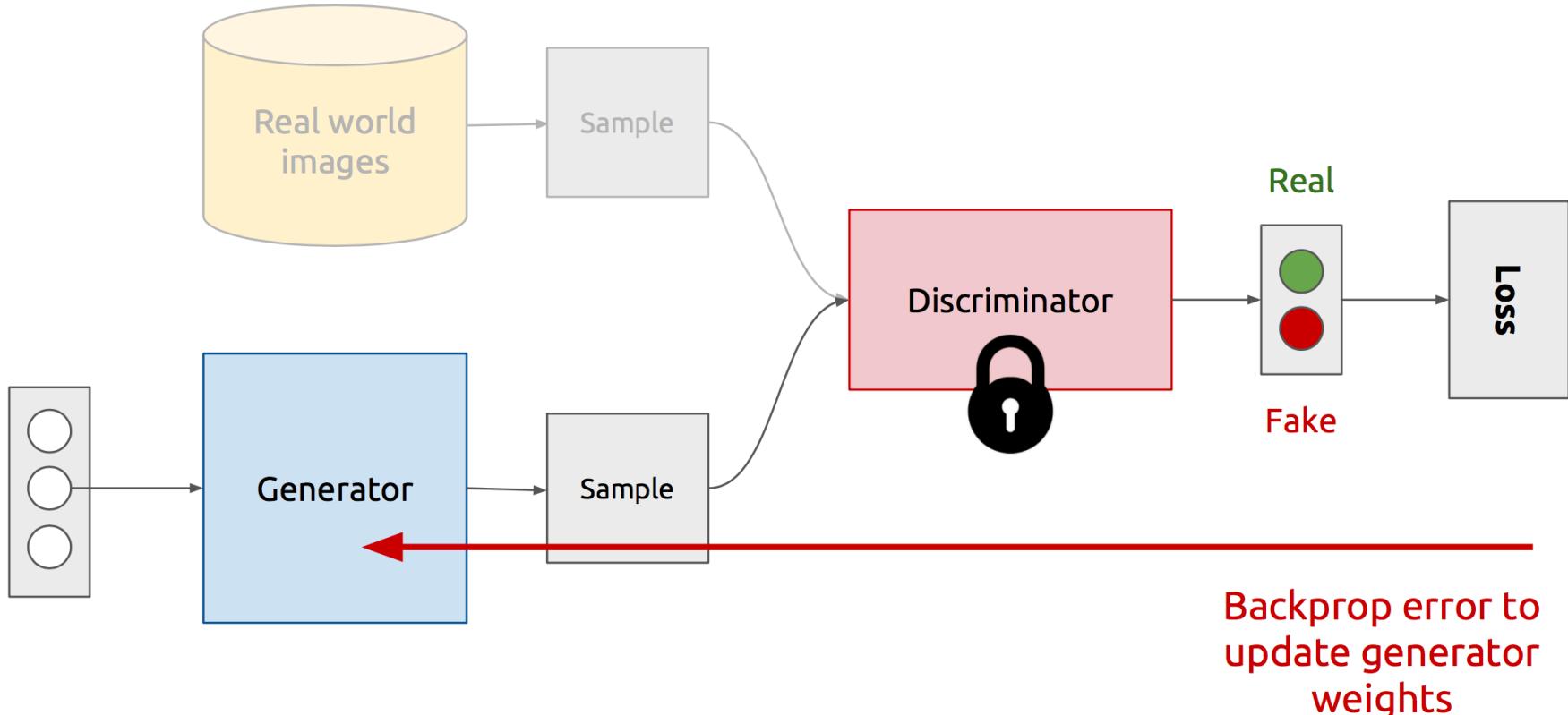


# Training discriminator



# Training generator

Latent random variable



# Manipulating latent codes by InfoGAN



(a) Varying  $c_1$  on InfoGAN (Digit type)



(b) Varying  $c_1$  on regular GAN (No clear meaning)



(c) Varying  $c_2$  from  $-2$  to  $2$  on InfoGAN (Rotation)

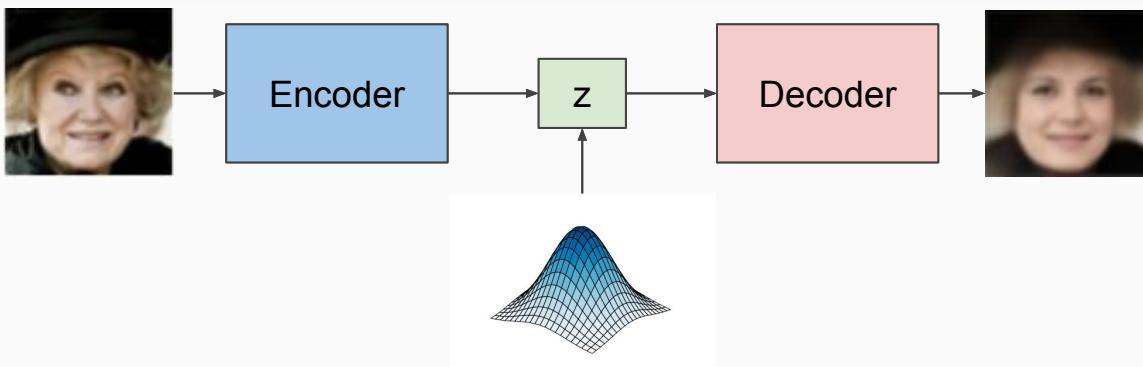


(d) Varying  $c_3$  from  $-2$  to  $2$  on InfoGAN (Width)

# Outline

- Related concepts
- Variational Autoencoder (VAE)
- Generative Adversarial Networks (GAN)
- **VAE vs. GAN**

# VAE vs. GAN



VAE

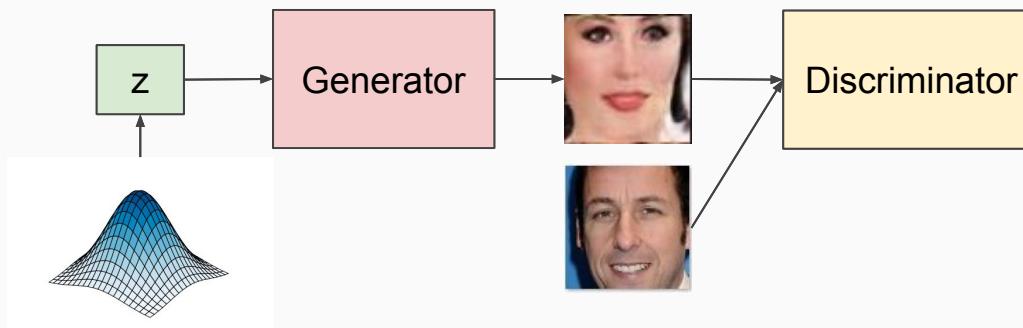
- ✓ : Given an **X** **easy** to find **z**.
- ✓ : Interpretable probability  $P(X)$

**X:** Usually outputs blurry Images

## GAN

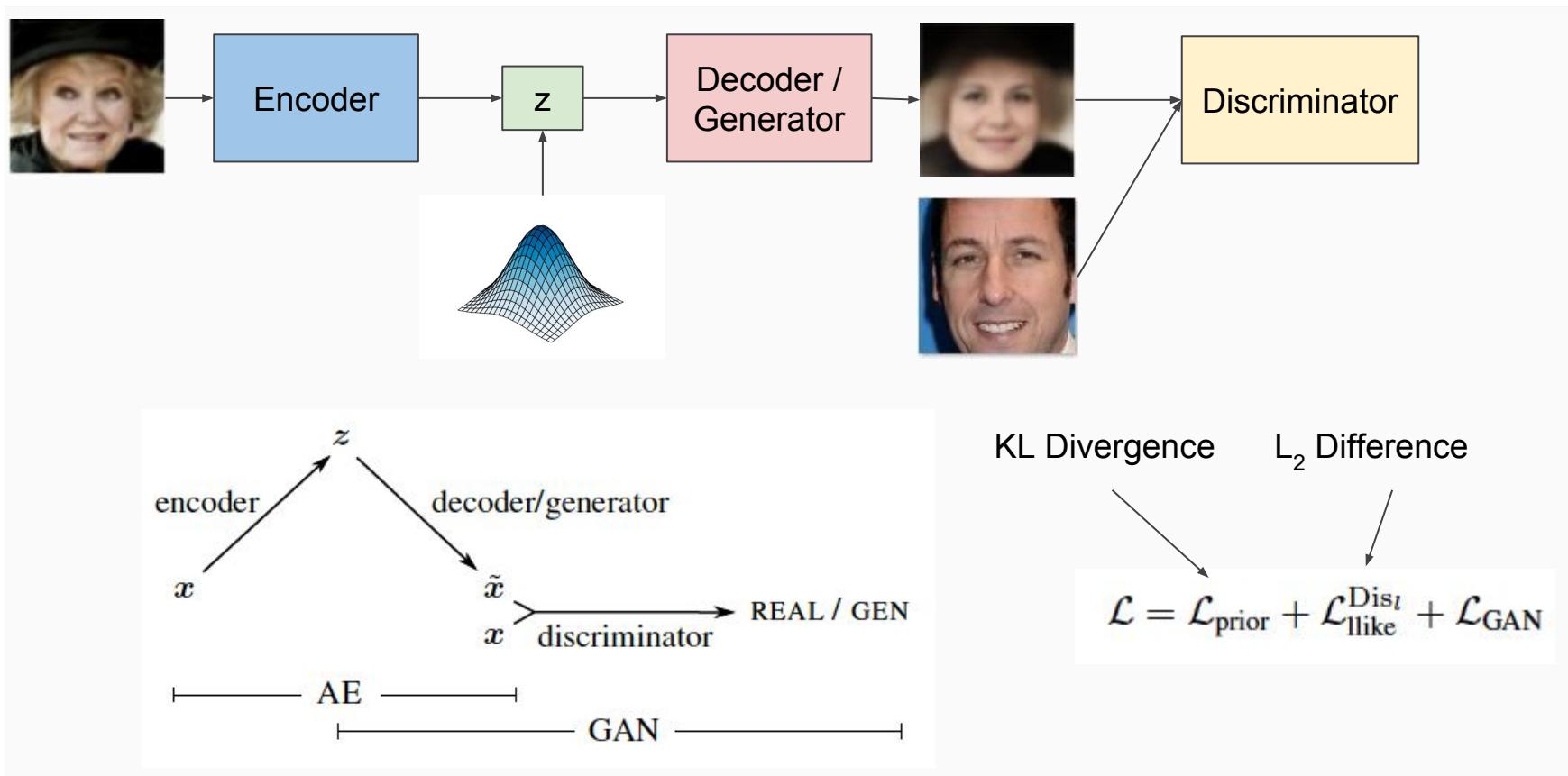
✓ : Very sharp images

**X:** Given an **X** **difficult** to find **z**. (Need to backprop.)



✓ / X: No explicit  $P(X)$ .

# GAN + VAE: combination



# Results by VAE vs. GAN

VAE



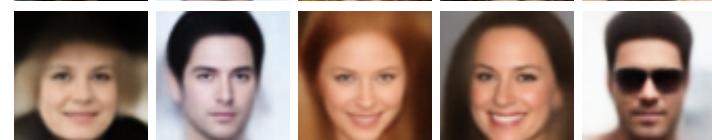
Input



$\text{VAE}_{\text{Dis}_l}$



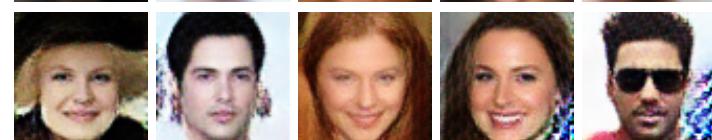
VAE



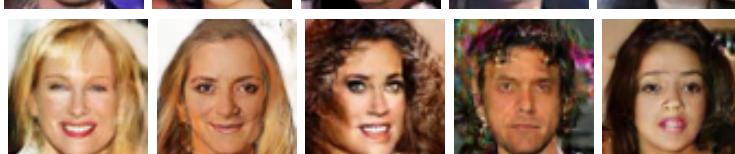
VAE/GAN



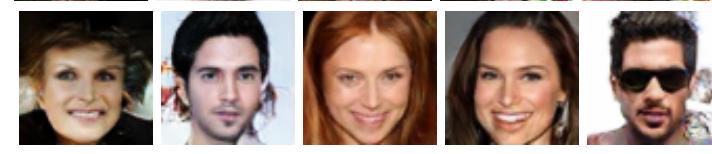
$\text{VAE}_{\text{Dis}_l}$



GAN



VAE/GAN



Samples from different generative models

Reconstructions from different autoencoders

$\text{VAE}_{\text{Dis}_l}$  : Train a GAN first, then use the discriminator of GAN to train a VAE

VAE/GAN: GAN and VAE trained together

Thank you!