

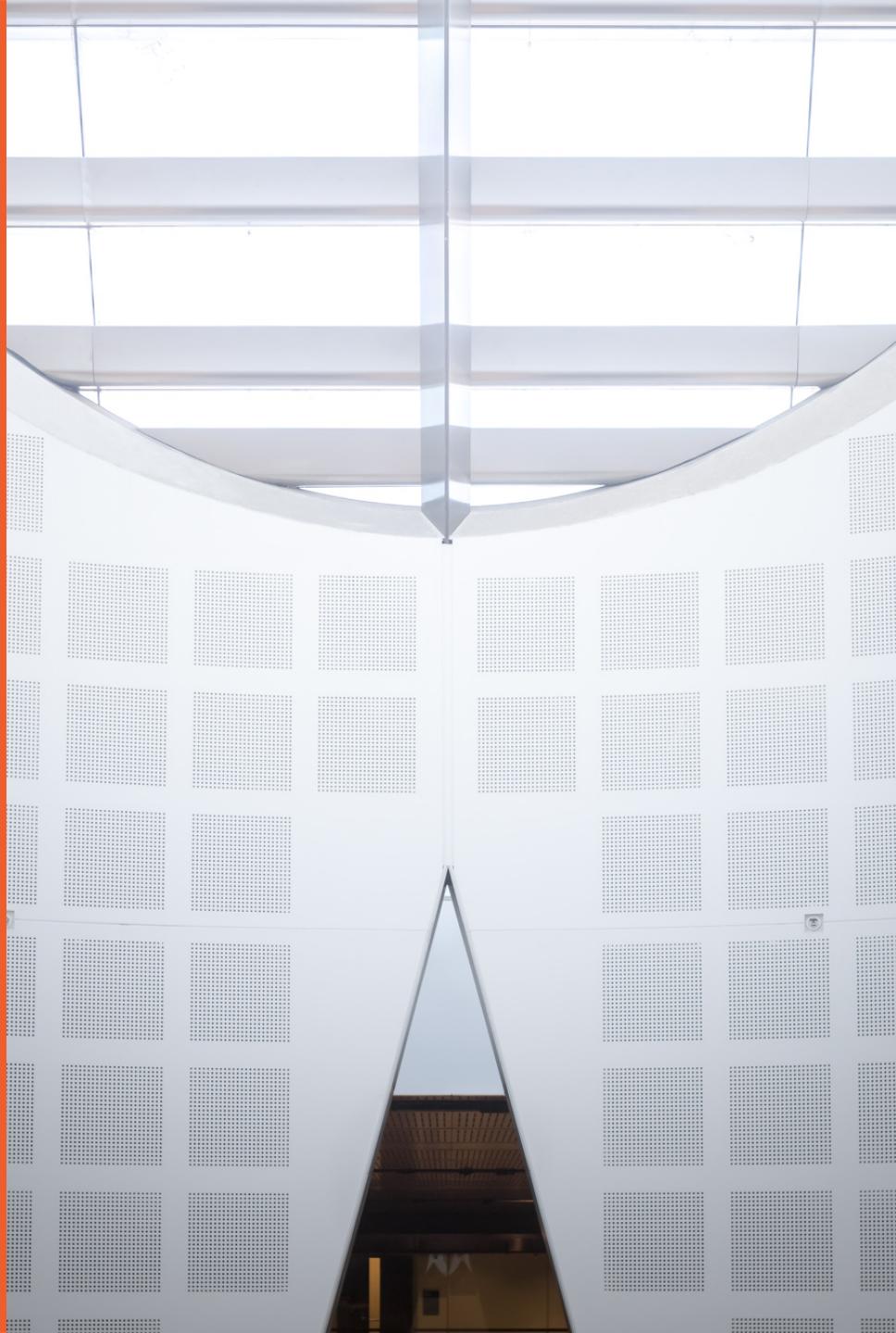
# Generative Models

Dr Chang Xu

School of Computer Science



THE UNIVERSITY OF  
SYDNEY



# Generative Modeling

- Density Estimation



- Sample Generation



# Why study generative model?

- Realistic generation tasks



Image credit to [Jun-Yan Zhu et al. 2017]



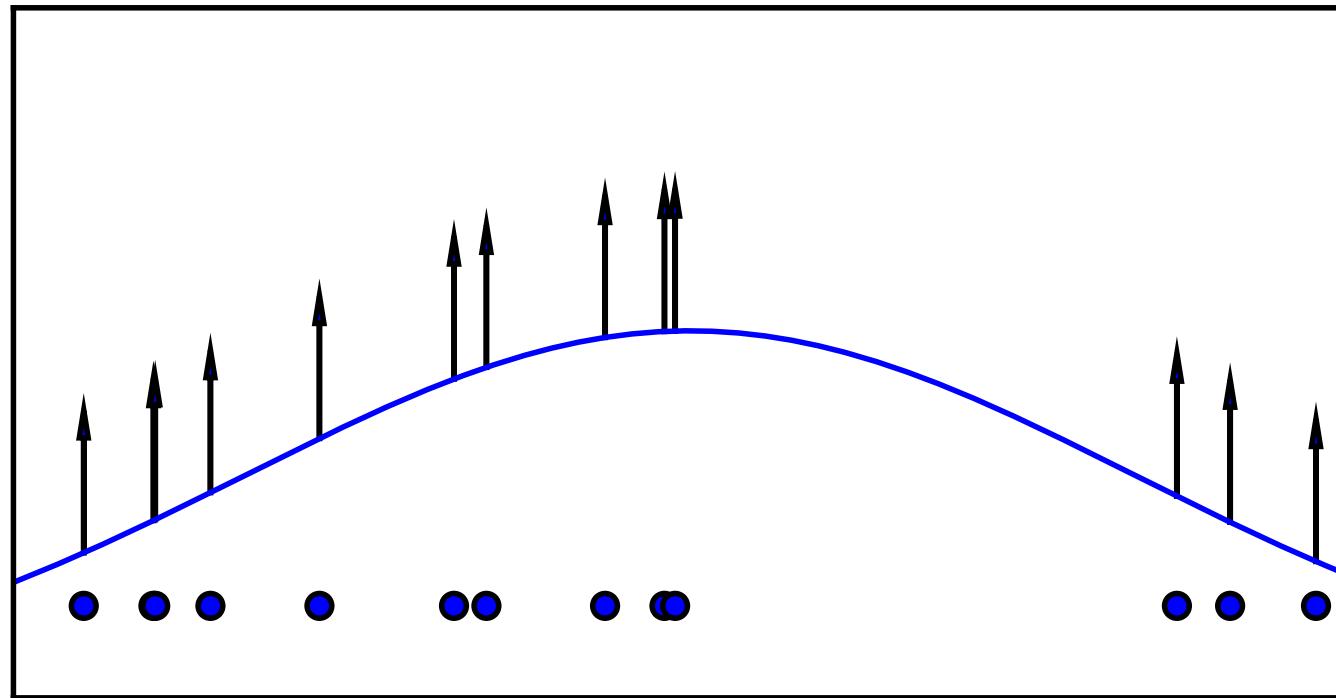
The figure consists of two side-by-side satellite images of a residential area. The left image is labeled 'input' at the bottom center and shows a dense cluster of houses, trees, and streets. The right image is labeled 'output' at the bottom center and shows the same area but with all buildings and trees removed, resulting in a clean, geometric grid of streets.

Image credit to [Phillip Isola et al. 2017]

- Simulate possible futures for planning (e.g. Stock Market Prediction)
  - Training generative models can also enable inference of latent representations that can be useful as general features

# Introduction

- Maximum Likelihood



$$\theta^* = \arg \max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(x \mid \theta)$$

# **Generative Adversarial Networks**

# Generative Adversarial Networks

0.6551  
(RNG) →



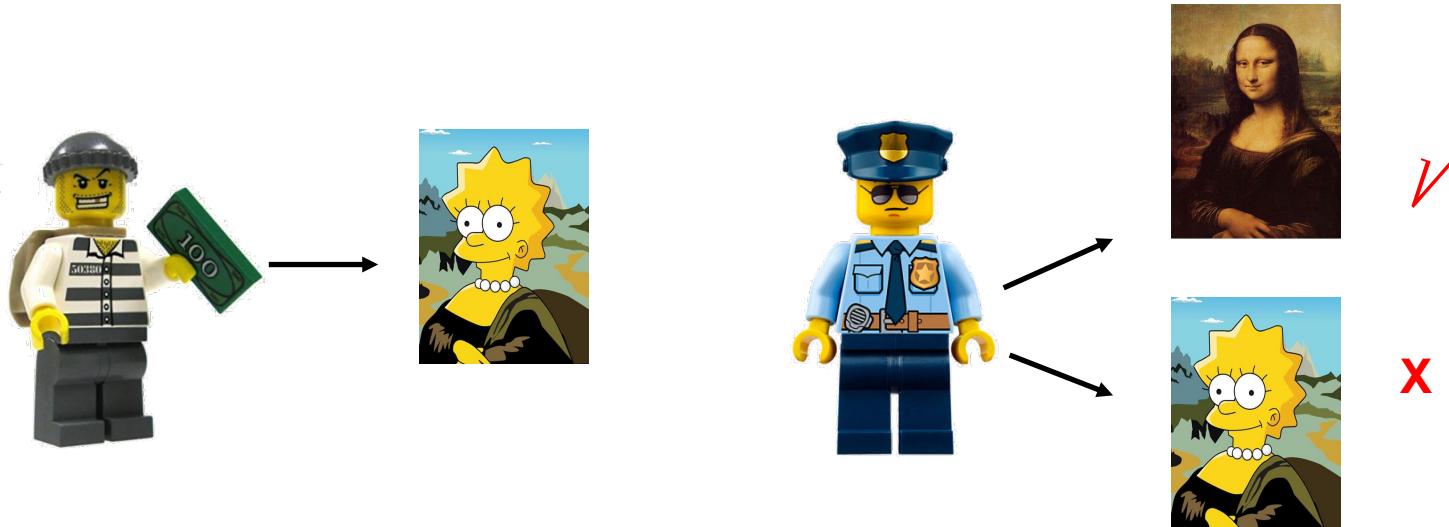
# Generative Adversarial Networks

- A **counterfeiter-police game** between two components: a generator **G** and a discriminator **D**
- **G**: counterfeiter, trying to fool police with fake currency
- **D**: policy, trying to detect the counterfeit currency
- Competition drives both to improve, until counterfeits are *indistinguishable* from genuine currency



# Generative Adversarial Networks

- A **two-player game** between two components: a generator **G** and a discriminator **D**
- **G**: try to fool the discriminator by generating real-looking images
- **D**: try to distinguish between real and fake images



# Generative Adversarial Networks

- A **two-player game** between two components: a generator **G** and a discriminator **D**
- **G**: try to fool the discriminator by generating real-looking images
- **D**: try to distinguish between real and fake images

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

D predicting that  
real data is genuine

D predicting that  
G's generated data is fake

Discriminator outputs likelihood in (0,1) of real image.

# Generative Adversarial Networks

- A **two-player game** between two components: a generator **G** and a discriminator **D**
- **G**: try to fool the discriminator by generating real-looking images
- **D**: try to distinguish between real and fake images

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

D predicting that  
real data is genuine

D predicting that  
G's generated data is fake

D's goal: maximize objective such that  $D(x)$  is close to 1 (real) and  $D(G(z))$  is close to 0 (fake)

G's goal: minimize objective such that  $D(G(z))$  is close to 1 (discriminator is fooled into thinking generated  $G(z)$  is real)

# Generative Adversarial Networks

- A **two-player game** between two components: a generator **G** and a discriminator **D**
- **G**: try to fool the discriminator by generating real-looking images
- **D**: try to distinguish between real and fake images

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

D predicting that  
real data is genuine

D predicting that  
G's generated data is fake

Q: What can we use to represent D and G?

A: Neural networks!

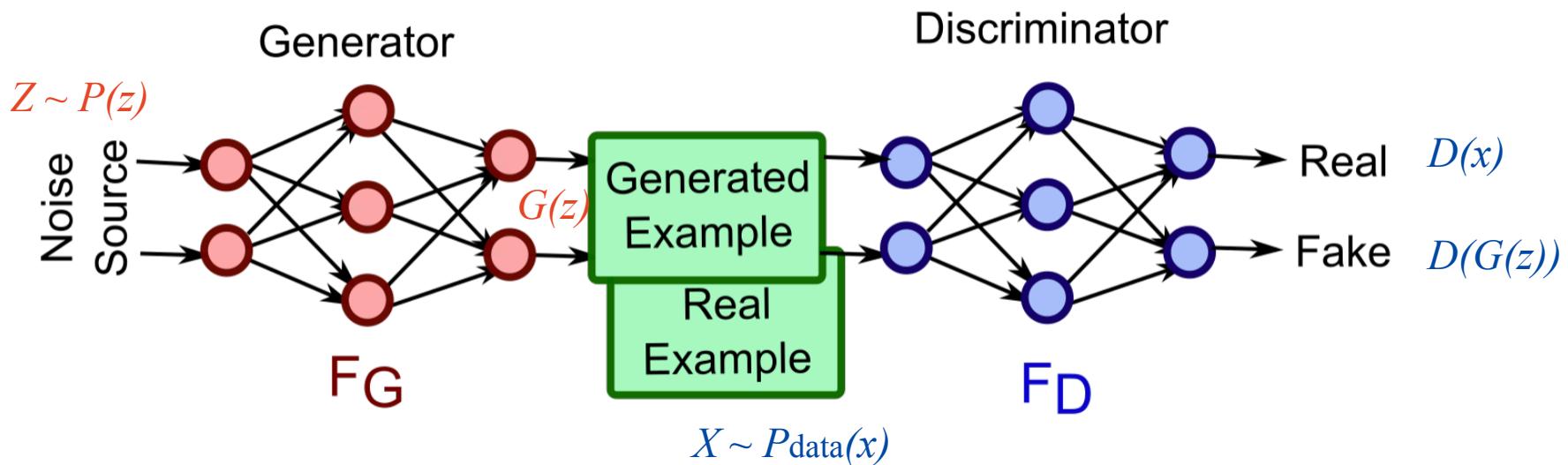
# Generative Adversarial Networks

- A **two-player game** between two components: a generator **G** and a discriminator **D**

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

D predicting that  
real data is genuine

D predicting that  
G's generated data is fake



# Generative Adversarial Networks

- A **two-player game** between two components: a generator **G** and a discriminator **D**

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

- Alternate between:

1. Gradient ascent on **D**

$$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. Gradient descent on **G**

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

# Algorithm

---

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator,  $k$ , is a hyperparameter. We used  $k = 1$ , the least expensive option, in our experiments.

---

**for** number of training iterations **do**

**for**  $k$  steps **do**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Sample minibatch of  $m$  examples  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  from data generating distribution  $p_{\text{data}}(\mathbf{x})$ .
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[ \log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) \right].$$

**end for**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(\mathbf{z}^{(i)}))).$$

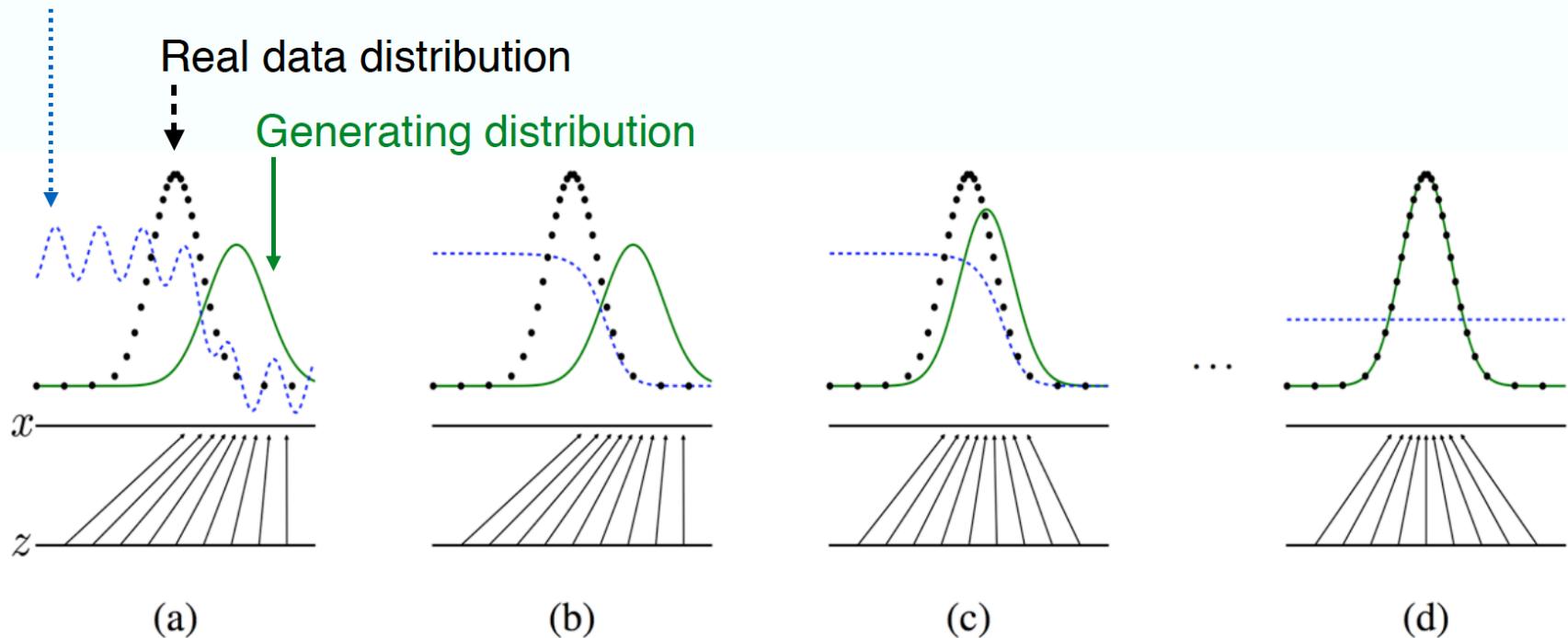
**end for**

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

---

# A simple example

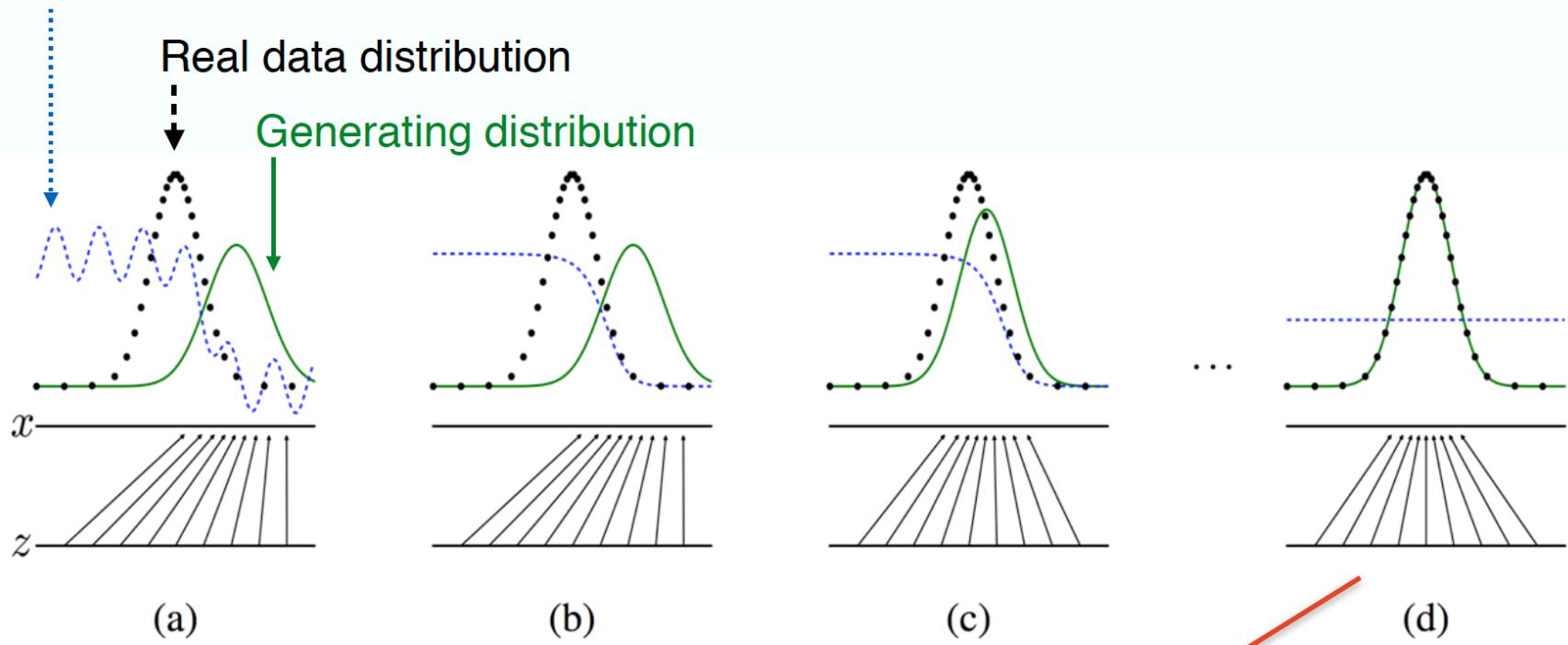
## Discriminative distribution



Generative adversarial nets are trained by simultaneously updating the discriminative distribution (blue line) so that it discriminates between samples from the data generating distribution (black, dotted line)  $p_x$  from those of the generative distribution  $p_g$  (green line). The lower horizontal line is the domain from which  $z$  is sampled, in this case uniformly. The horizontal line above is part of the domain of  $x$ . The upward arrows show how the mapping  $x = G(z)$  imposes the non-uniform distribution  $p_g$  on transformed samples.  $G$  contracts in regions of high density and expands in regions of low density of  $p_g$ .

# A simple example

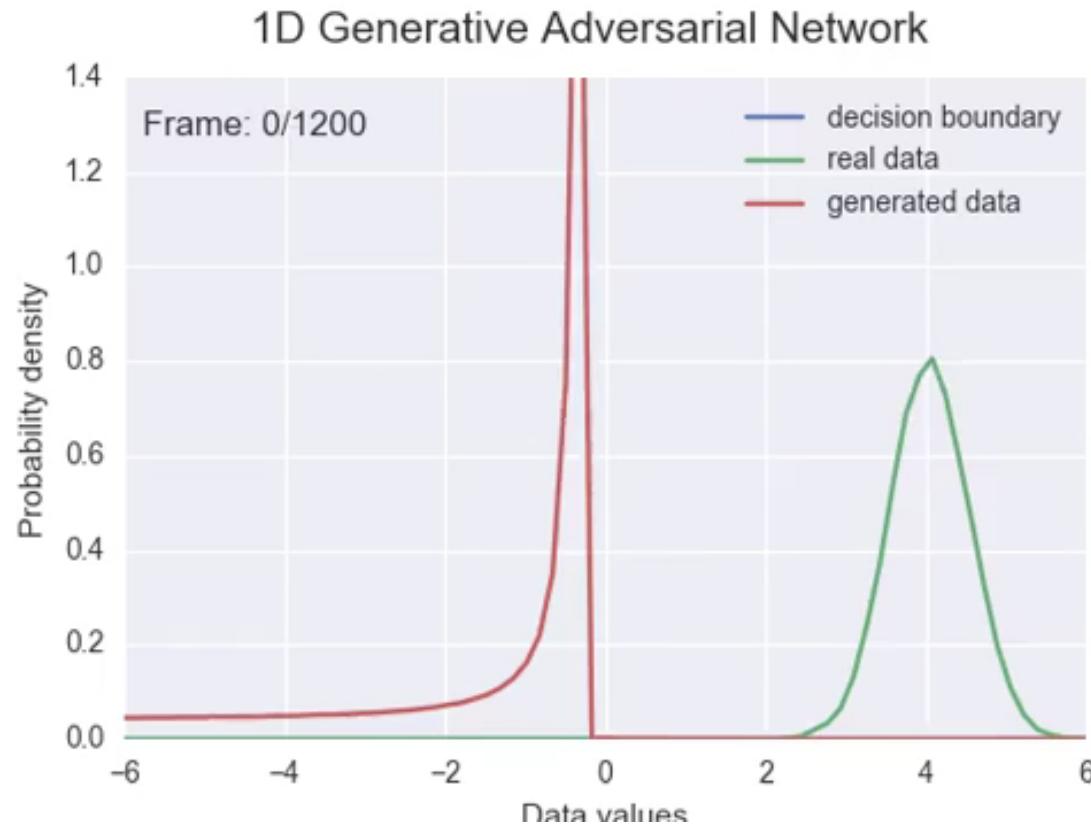
## Discriminative distribution



After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because  $p_g = p_{data}$ . The discriminator is unable to differentiate between the two distributions, i.e.  $D(x) = 1/2$ .

# A simple example

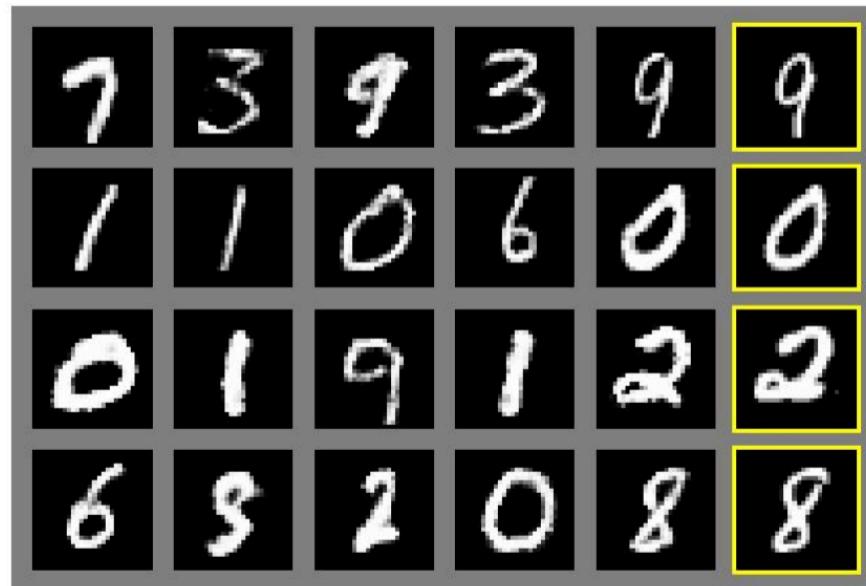
When the data distribution is a 1-D Gaussian distribution



<http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/>

# Generative Adversarial Networks

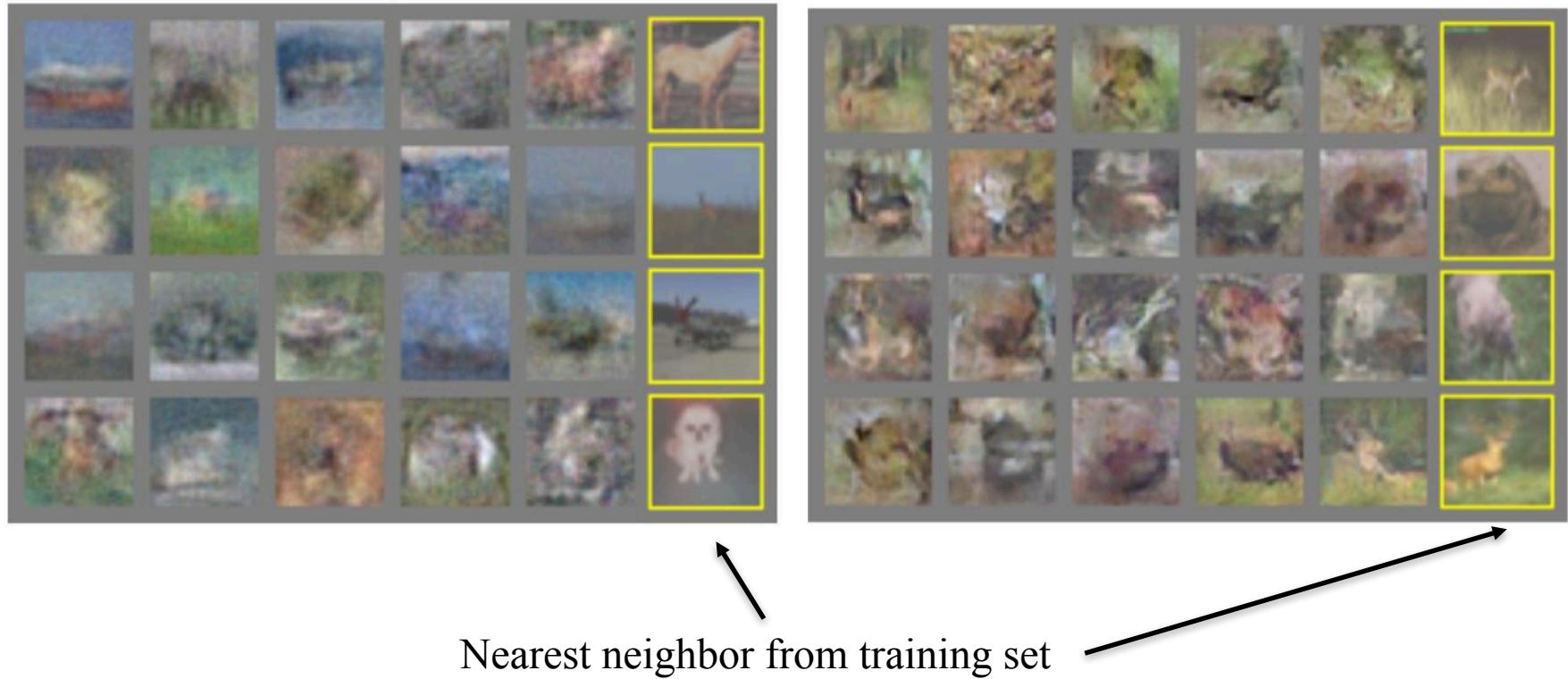
- Generated samples



Nearest neighbor from training set

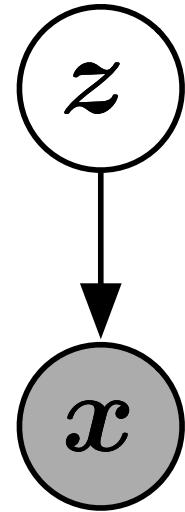
# Generative Adversarial Networks

- Generated samples (CIFAR-10)



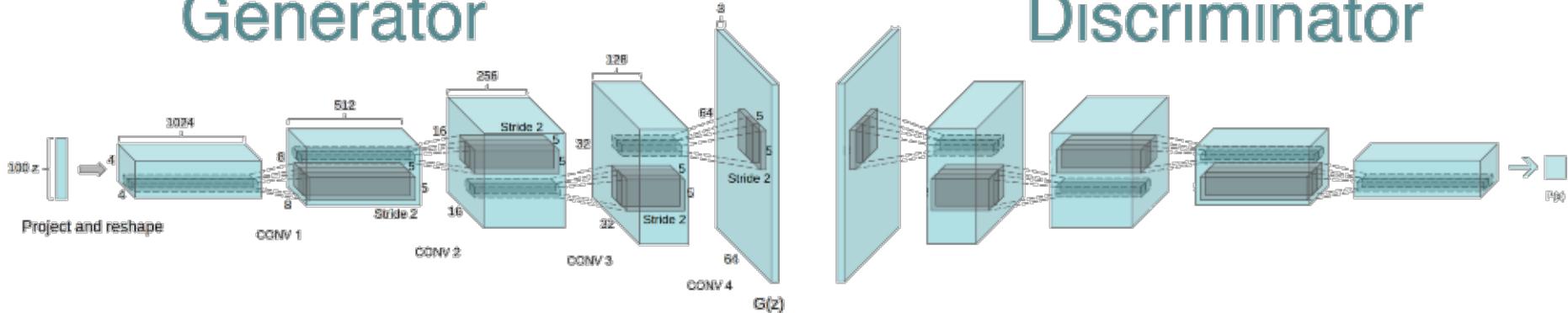
# Generative Adversarial Networks

- $\mathbf{x} = G(\mathbf{z}; \theta^{(G)})$
- It is only required that,  $G$  is differentiable
- So, having training data from real data distribution  $p_r$ , what we want is a generative model that can draw samples from generator's distribution  $p_g$ , where  $p_r \approx p_g$
- Don't need to write a formula for  $p_g$  just learn to draw sample directly.



# Deep Convolutional GANs (DCGANs)

Generator



Discriminator

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

# Deep Convolutional GANs (DCGANs)

Samples from the model look much better!



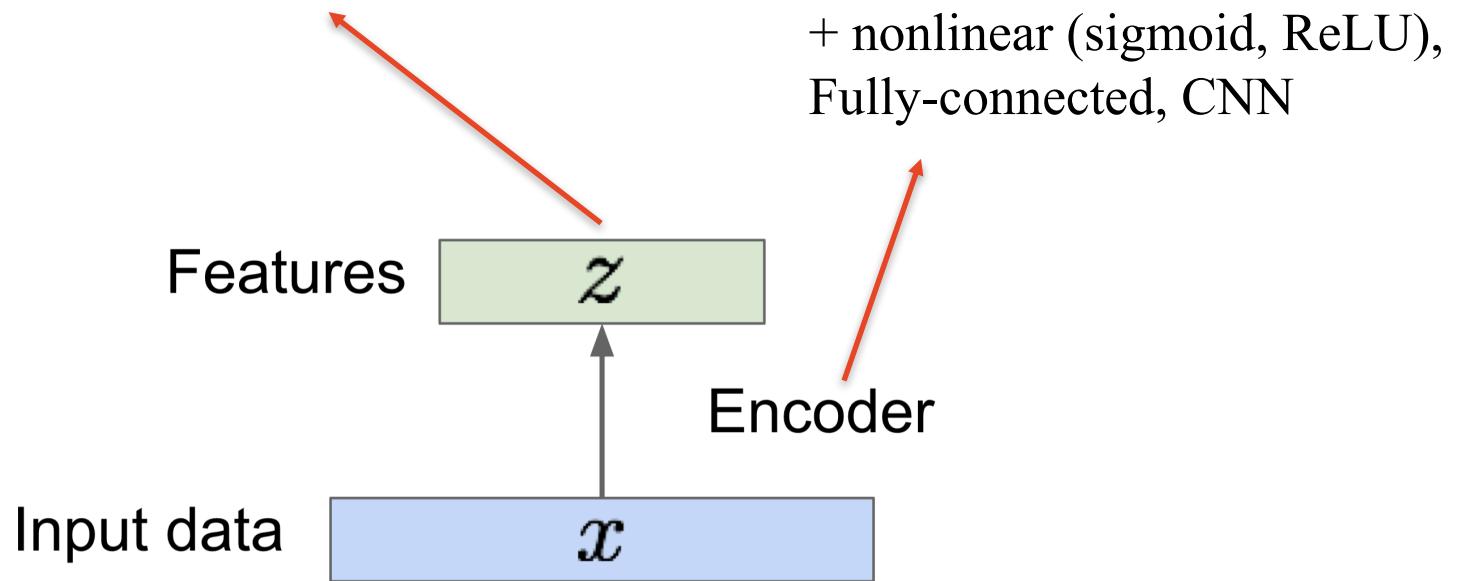
Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

# **Variational Auto-Encoder**

# Recap: Autoencoder

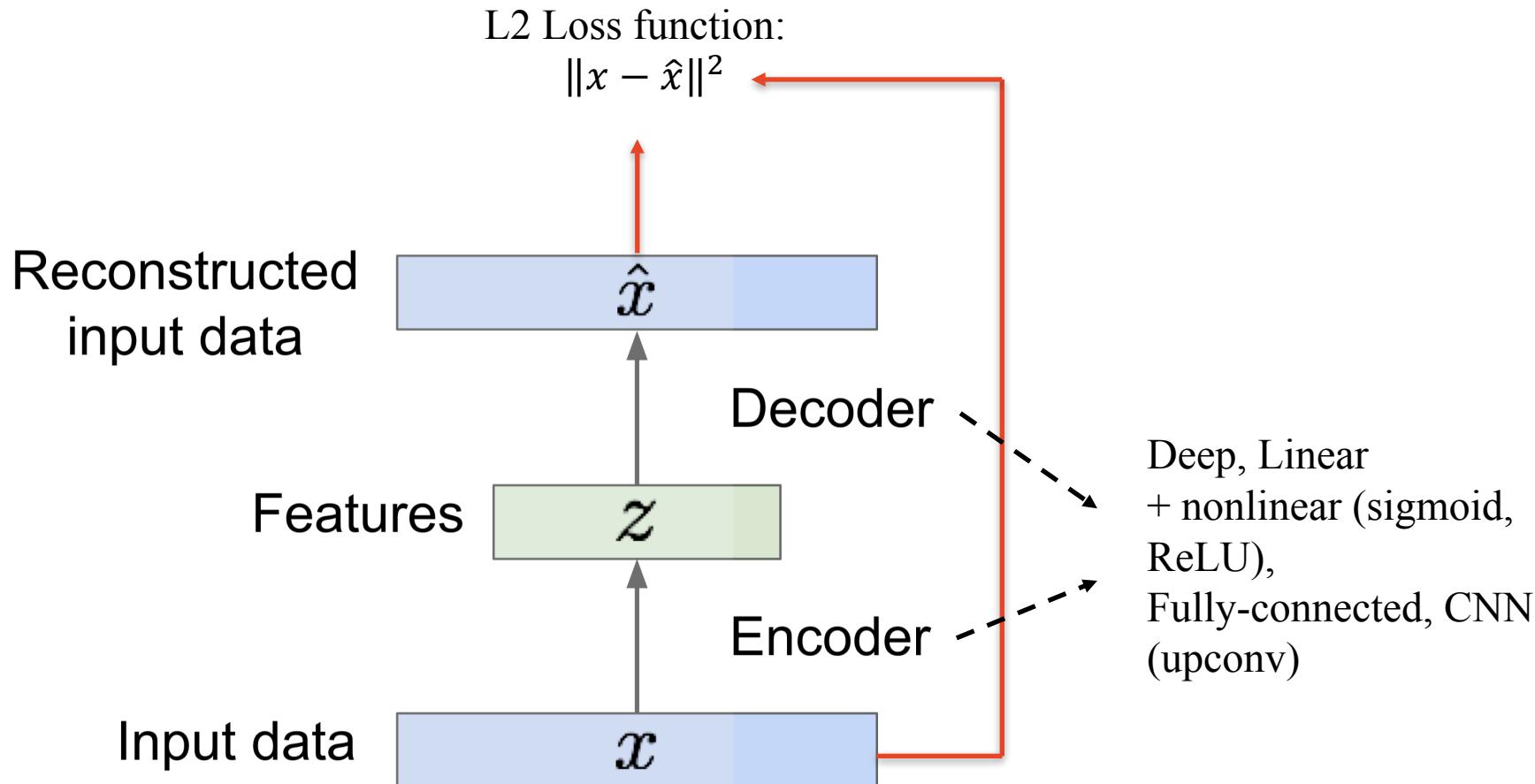
- Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

$z$  usually smaller than  $x$   
(dimension reduction)

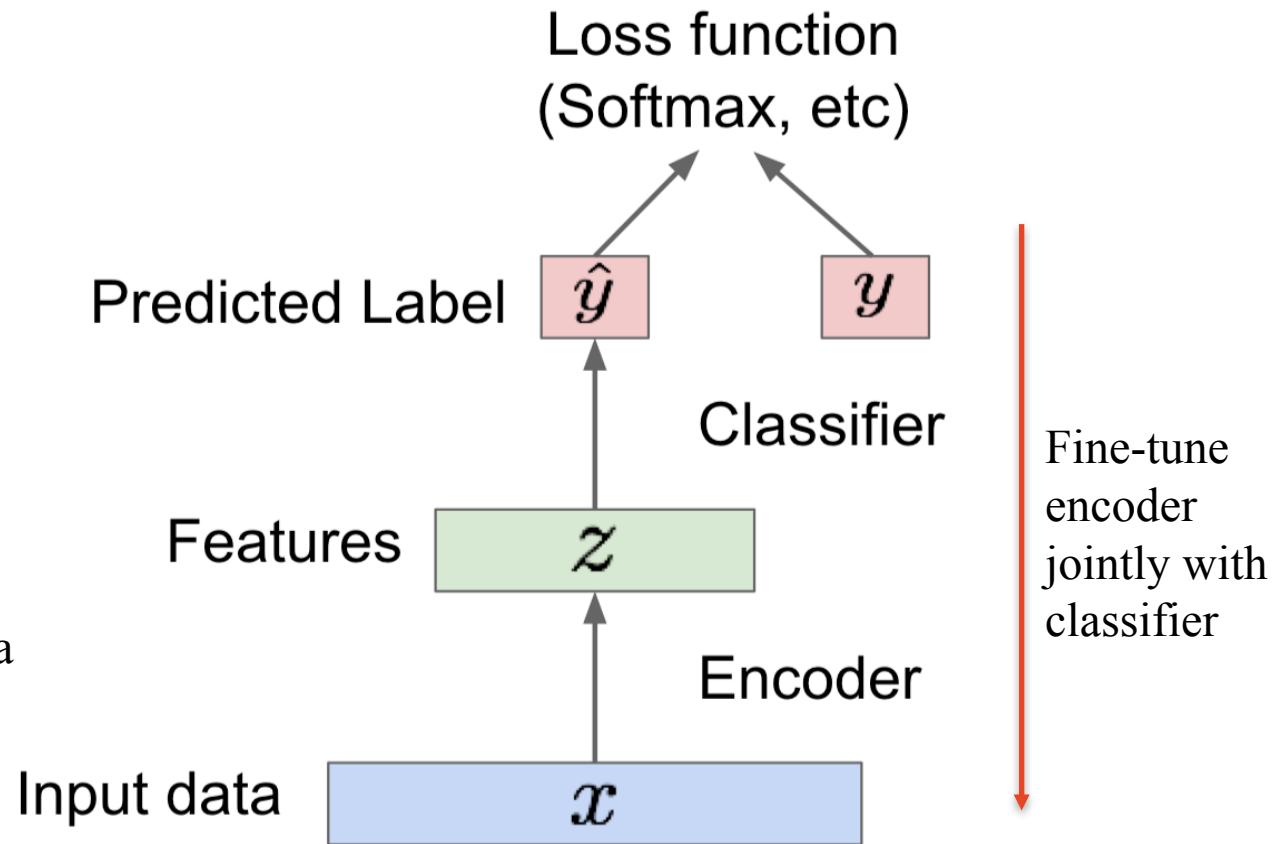


# Recap: Autoencoder

- Train such that features can be used to reconstruct original data  
“Autoencoding” – encoding itself



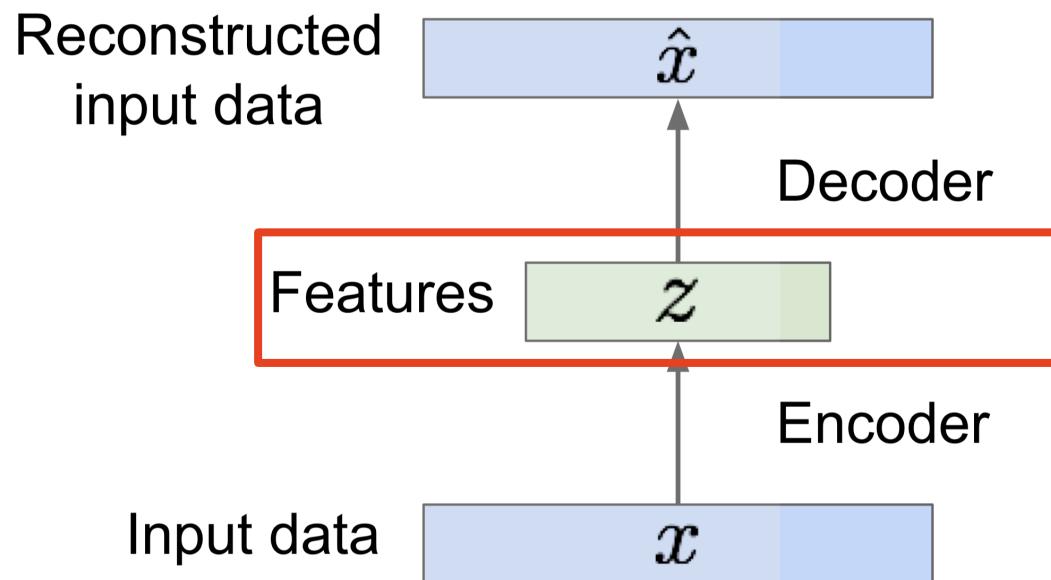
# Recap: Autoencoder



## Recap: Autoencoder

Autoencoders can reconstruct data, and can learn features to initialize a supervised model. Features capture factors of variation in training data.

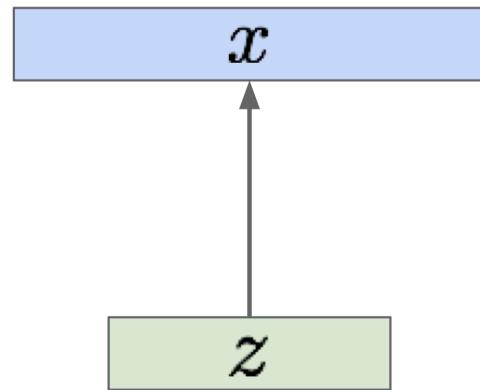
Can we **generate new images** from an autoencoder?



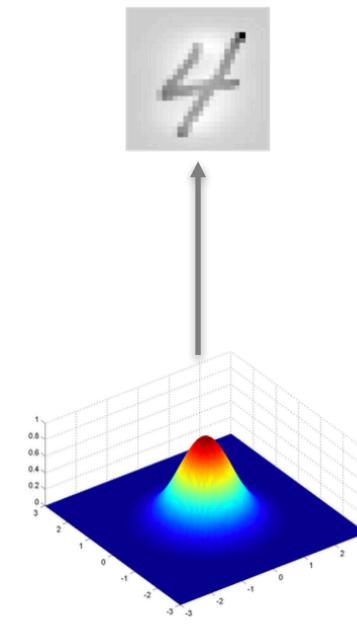
# Variational Autoencoders

- Probabilistic spin on autoencoders - will let us sample from the model to generate data!
- Assume training data  $\{x^{(i)}\}_{i=1}^N$  is generated from underlying unobserved (latent) representation  $z$

Sample from  
true conditional  
 $p_{\theta^*}(x|z^{(i)})$

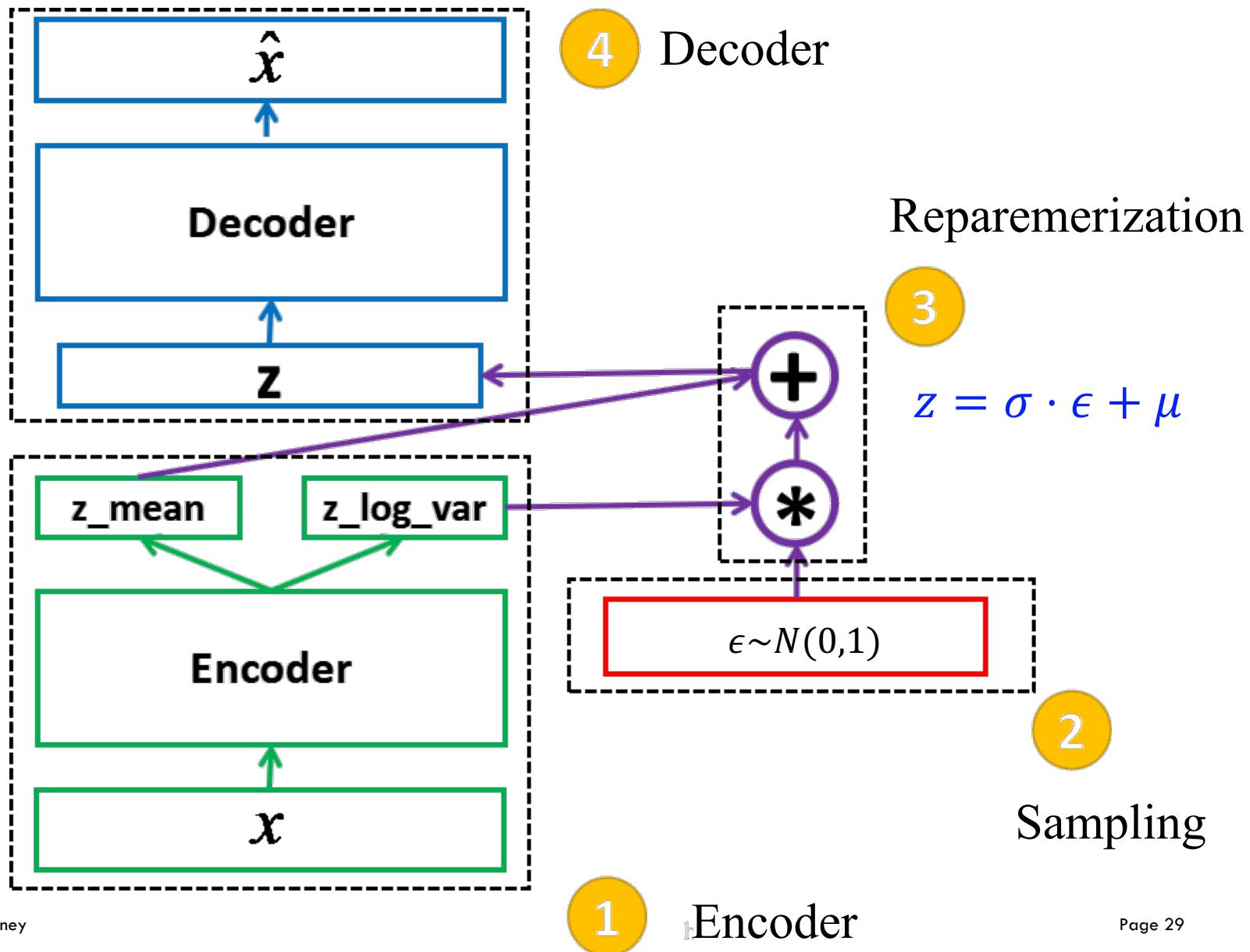


Sample from  
true prior  $p_{\theta^*}(z)$

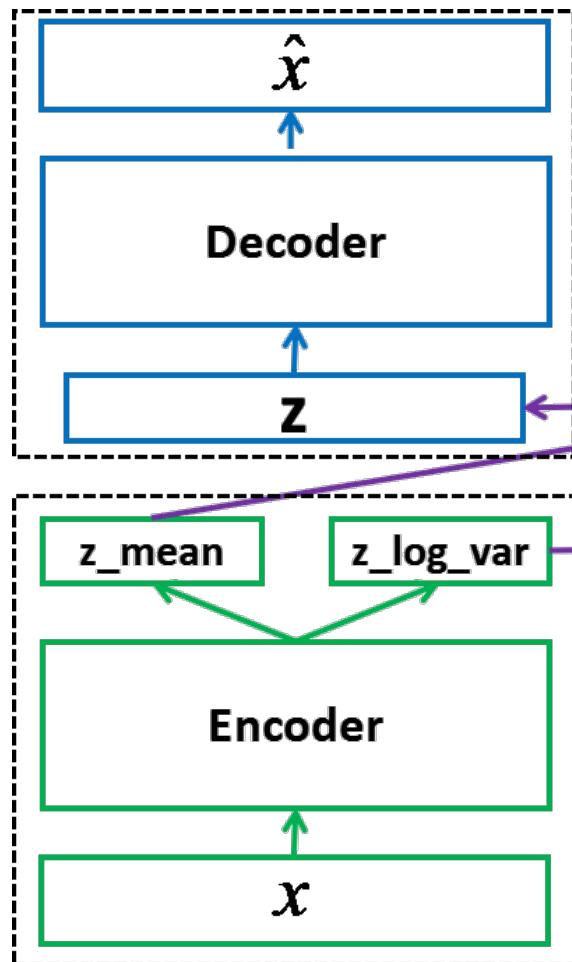


Choose prior  $p(z)$  to  
be simple, e.g.  
Gaussian.  
Reasonable for  
latent attributes, e.g.  
pose.

# Variational Autoencoders



# Variational Autoencoders



4

Cross Entropy

$$\sum_i^n -[x_i \log(\hat{x}_i) + (1 - x_i) \log(1 - \hat{x}_i)]$$

3

MSE  $\sum_i^n (x_i - \hat{x}_i)^2$

KL Divergence

$$-0.5(1 + \log \sigma^2 - \mu^2 - \sigma^2)$$

1

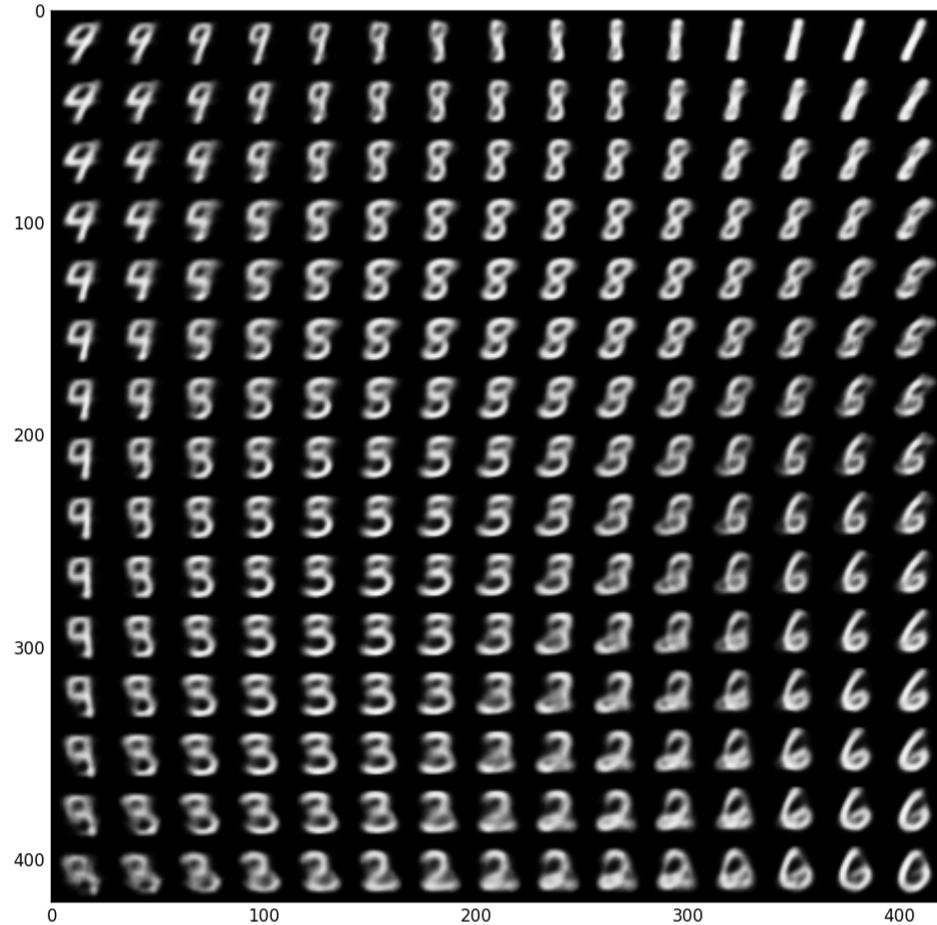
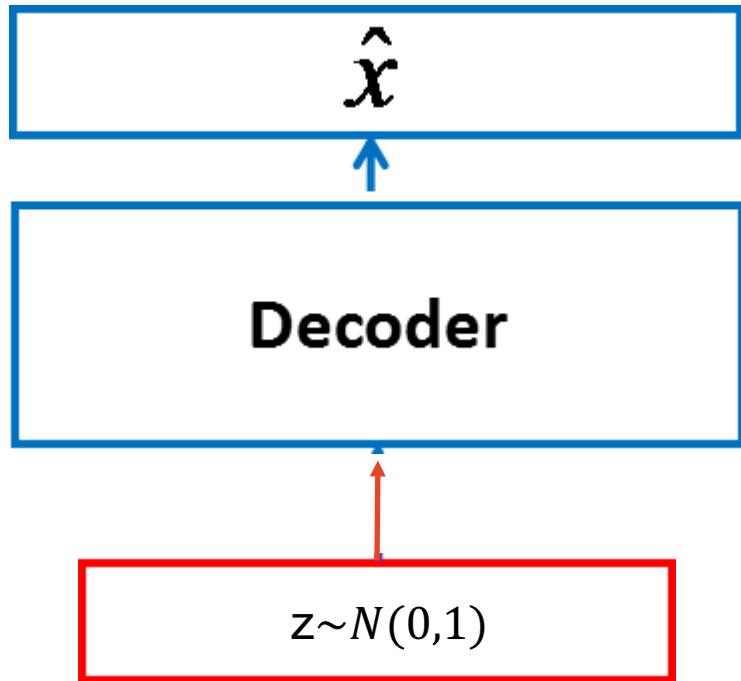
Min CrossEntropy + KLDivergence

h

# Variational Autoencoders

Generating Data:

Use decoder network.  
Sample z from prior.

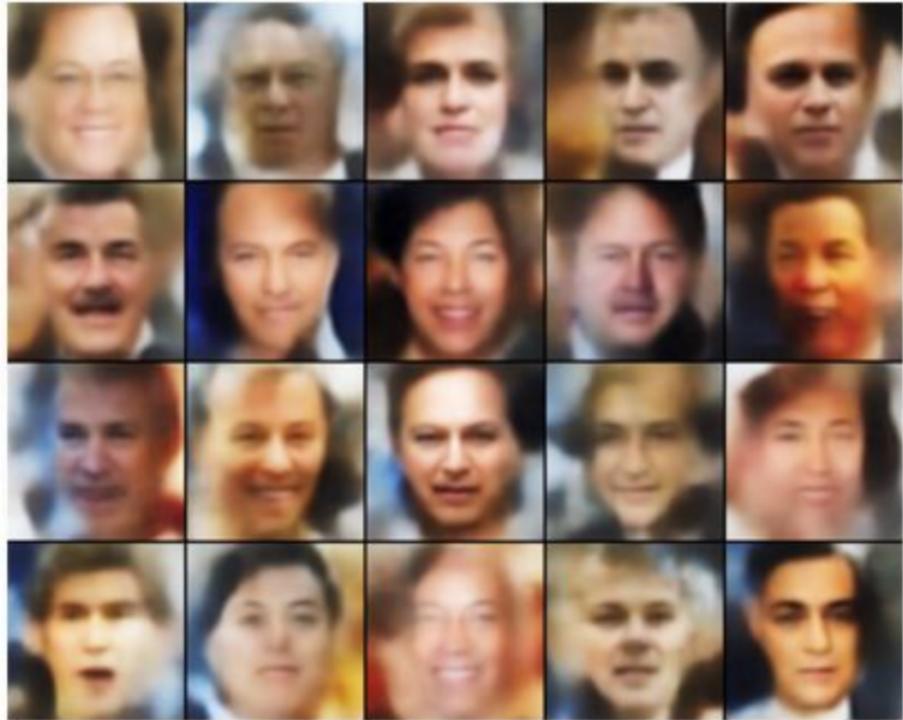


# Variational Autoencoders

Generating Data:



32x32 CIFAR-10



Labeled Faces in the Wild

## **Part II**