

---

# Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement  
Learning

# Today's Outline

---

- Unsupervised Learning Landscape
- Autoencoders and Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)

---

# Unsupervised Learning Landscape

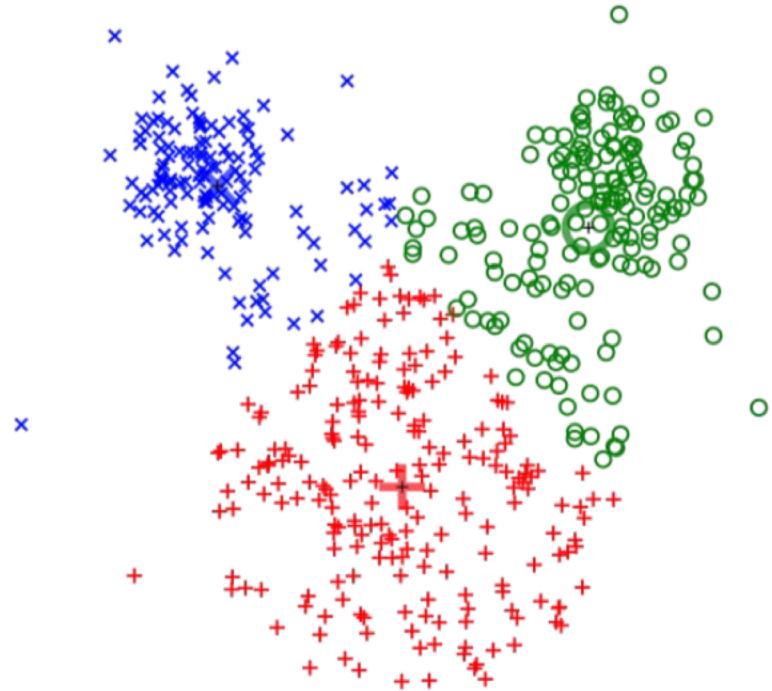
# Unsupervised Learning

---

- Supervised learning
  - Involves feature and label pairs as training data
  - Goal is to find a map from feature to label/value
- Unsupervised learning
  - Involves only feature vectors
    - Example: images
  - Goal is to learn some patterns of data
  - There is no objective measure of success

# Unsupervised Learning Tasks

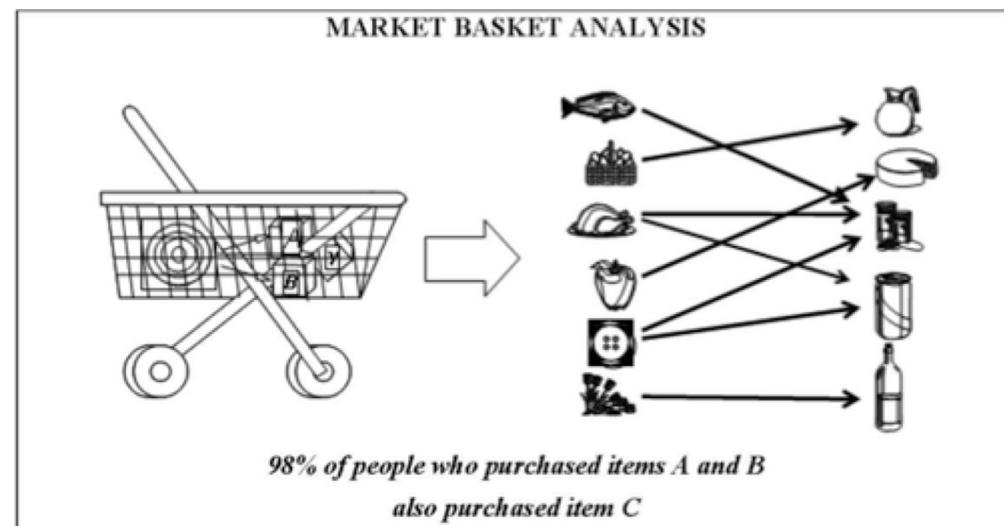
- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



K-means clustering

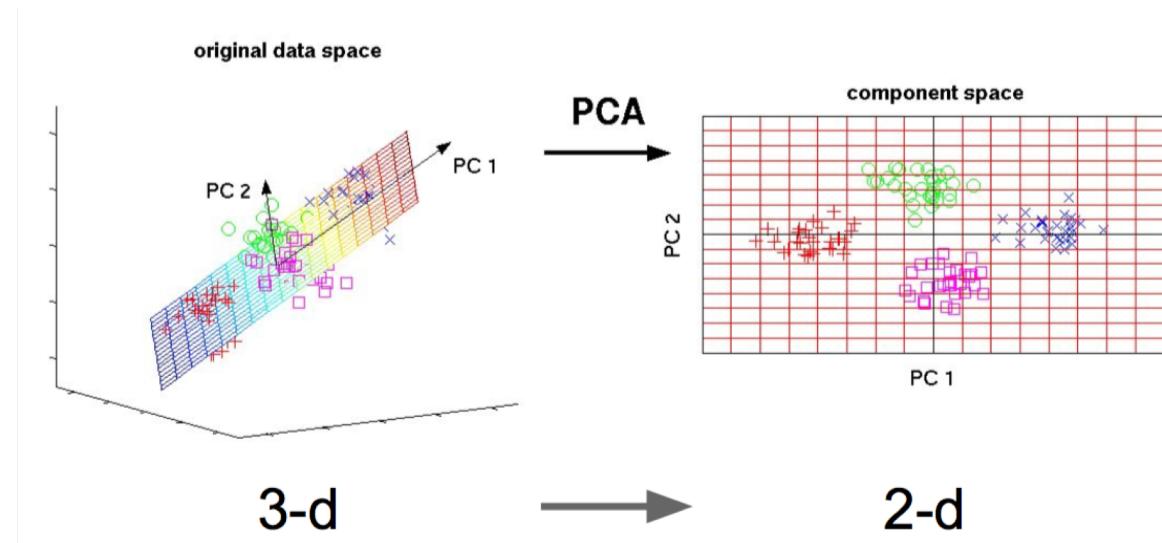
# Unsupervised Learning Tasks

- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



# Unsupervised Learning Tasks

- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



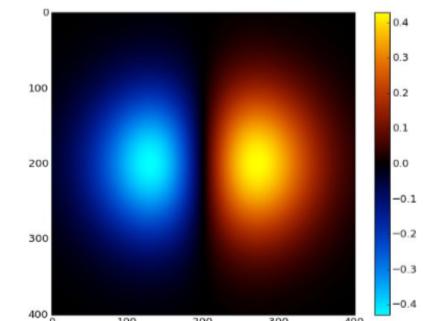
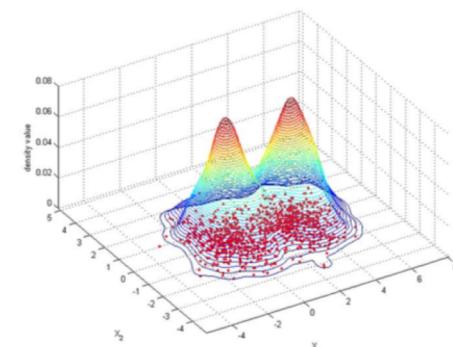
# Unsupervised Learning Tasks

- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation

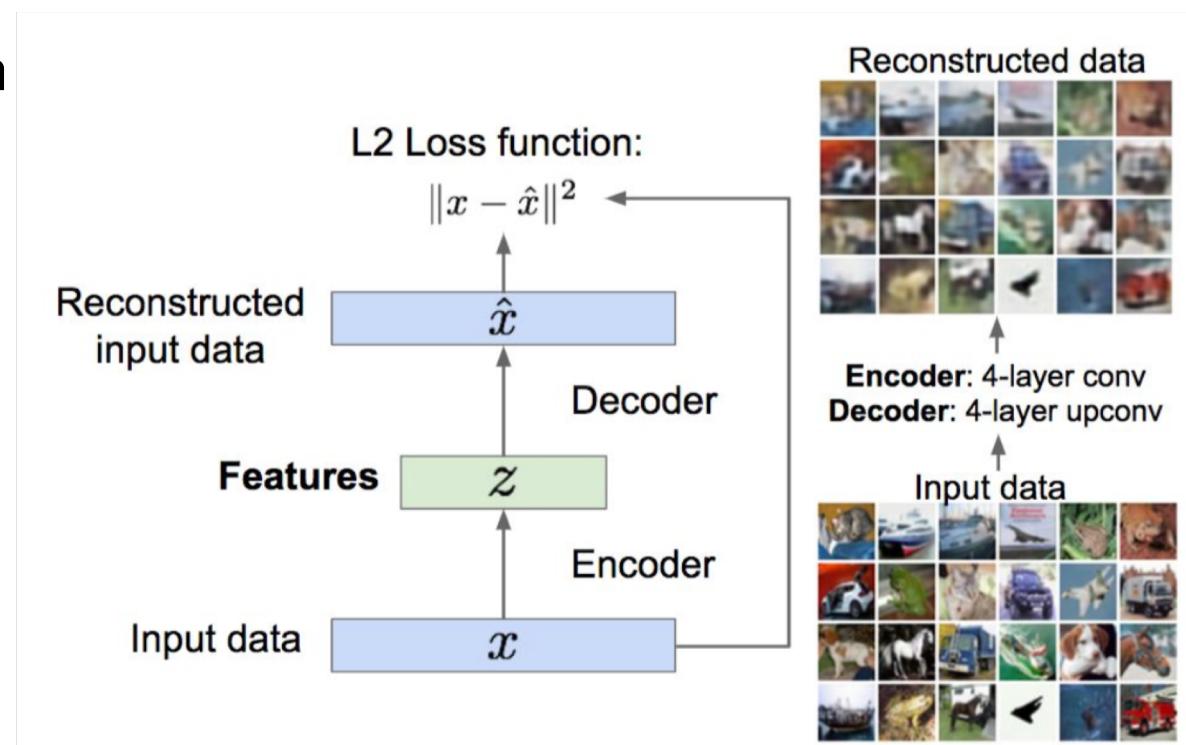


2-d density estimation

<sup>1</sup>Reference: CS231n (Stanford, Spring 17)

# Unsupervised Learning Tasks

- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



# Unsupervised Learning Tasks

- Clustering
  - Association rules
  - Dimensionality reduction
  - Density estimation
  - Embedding
  - Sampling

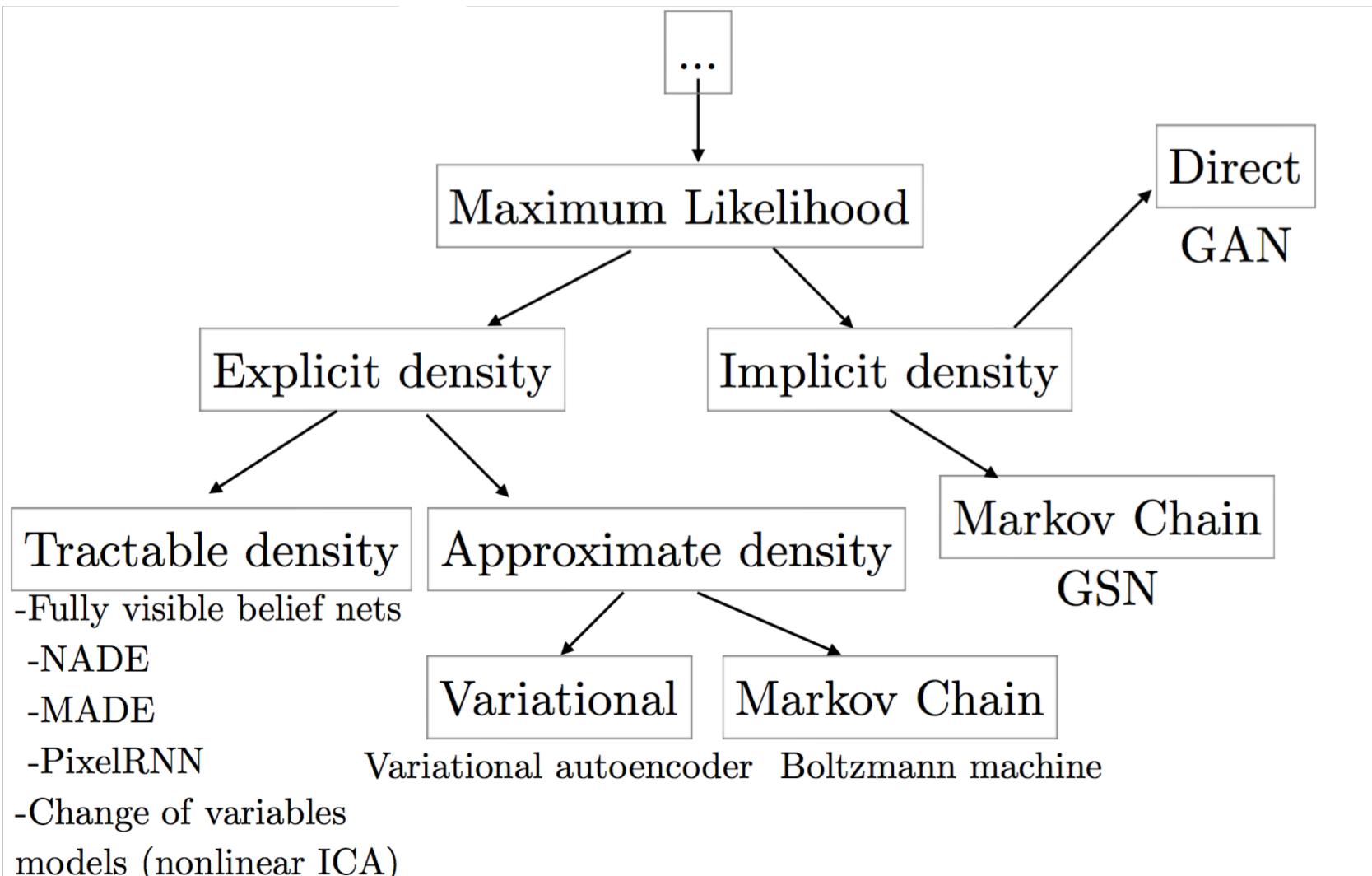


# Learning a Distribution

---

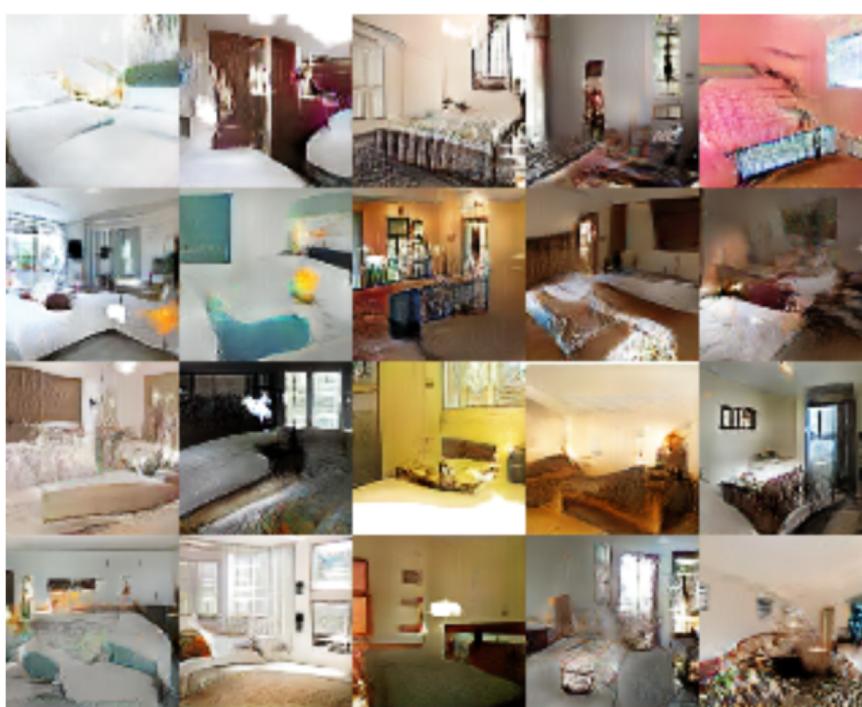
- Given (large amount of) data drawn from  $P_d$ , we want to estimate  $P_m$  such that samples from  $P_m$  are as similar as possible to samples from  $P_d$
- Two approaches:
  - Explicit
    - If we construct  $P_m$  explicitly, we can address all the other tasks mentioned
  - Implicit
    - We can directly generate a sample from  $P_m$  without explicitly defining it!

# Explicit and Implicit Approaches



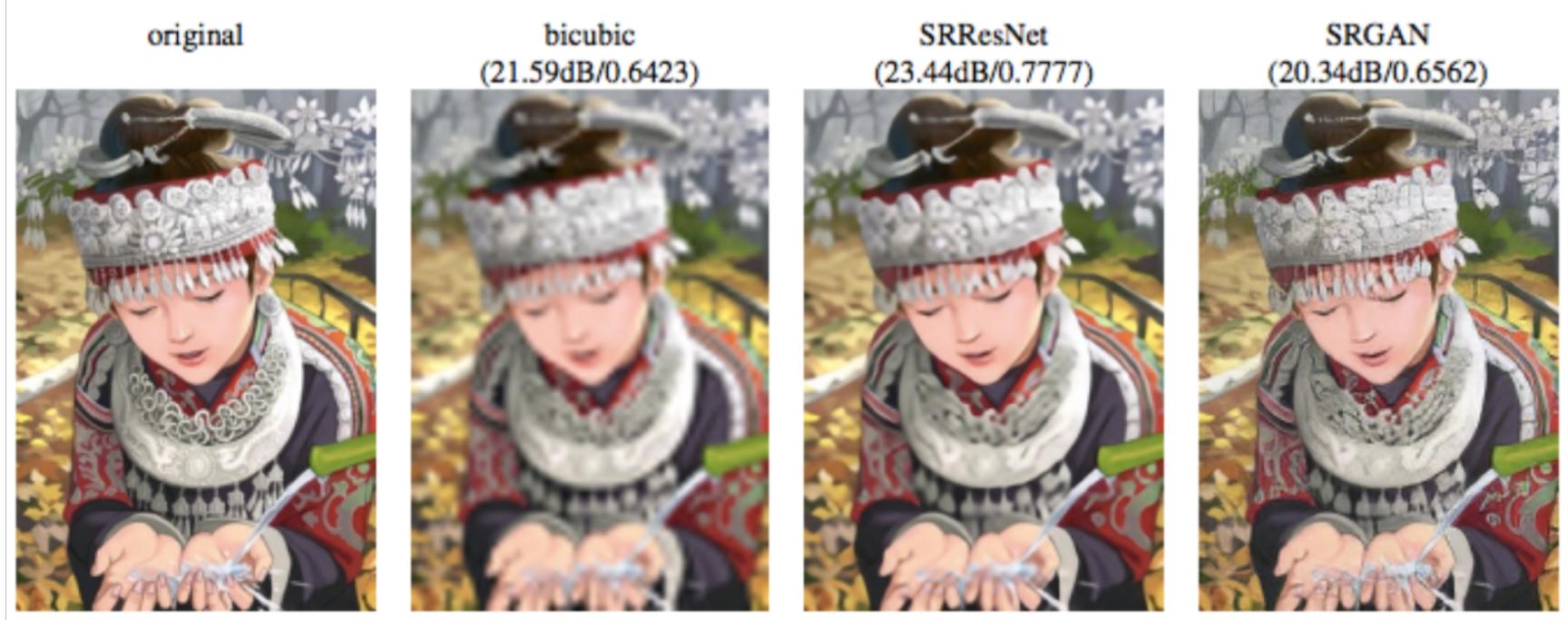
# Explicit and Implicit Approaches

- When would we be okay with an implicit approach
  - Simulate possible futures for planning
  - When samples themselves are useful for other tasks...



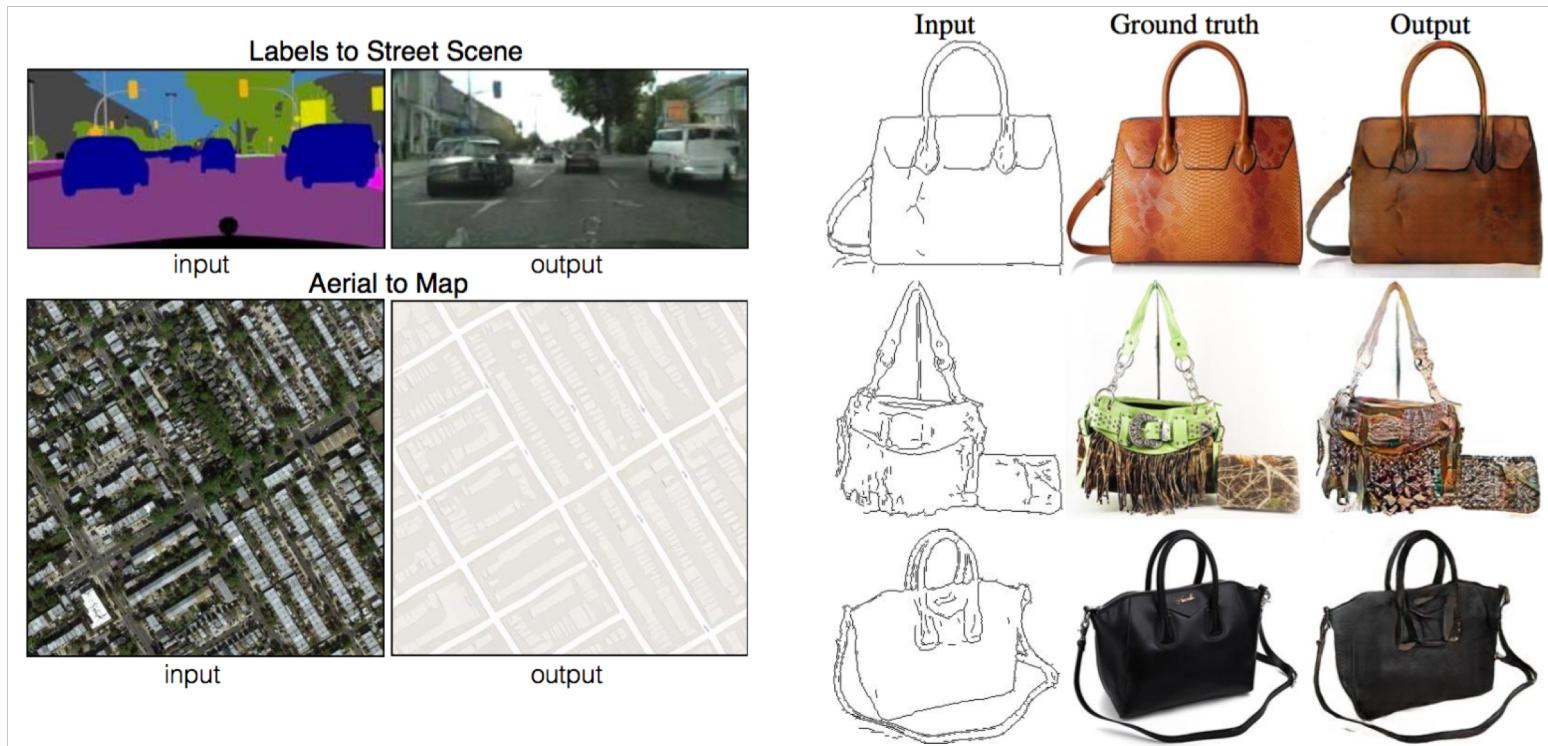
# Explicit and Implicit Approaches

- When would we be okay with an implicit approach
  - Simulate possible futures for planning
  - When samples themselves are useful for other tasks...



# Explicit and Implicit Approaches

- When would we be okay with an implicit approach
  - Simulate possible futures for planning
  - When samples themselves are useful for other tasks...



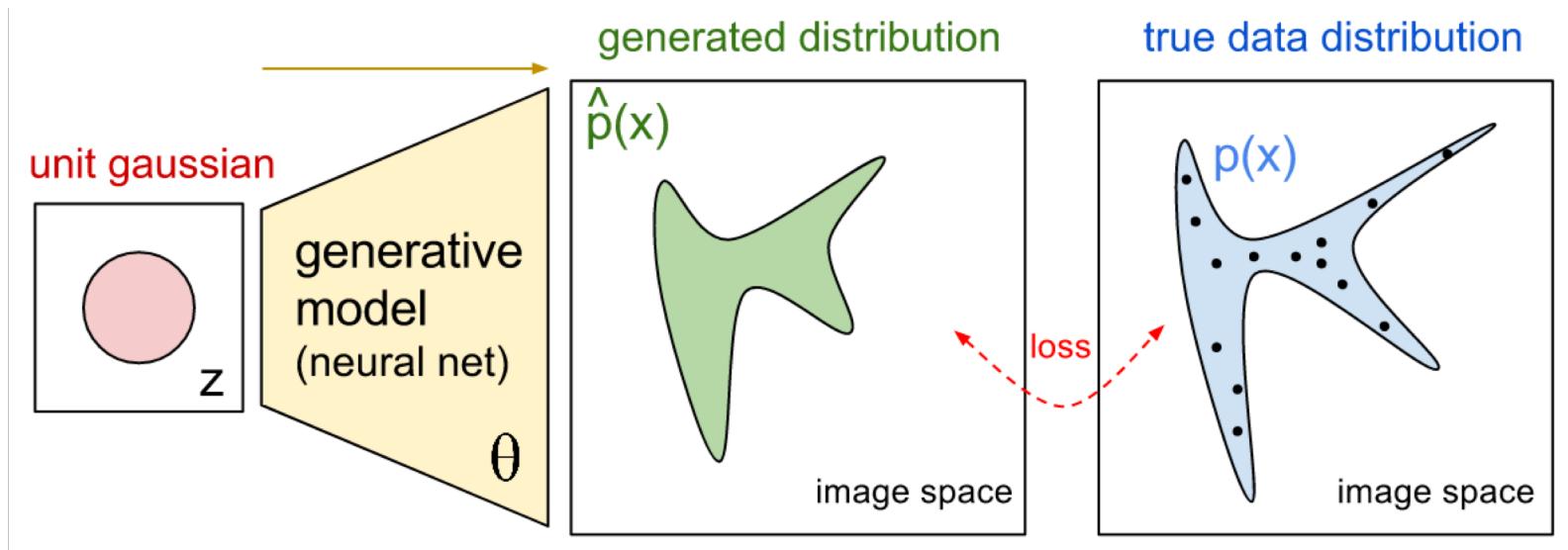
# Explicit and Implicit Approaches

---

- We will look at one model under each approach and work with **image** data
  - Explicit: Variational Autoencoders (VAE)
  - Implicit: Generative Adversarial Networks (GAN)
- Both use **neural networks** as a core object

# More than Memorization

- Either model (VAE or GAN) will essentially build the yellow box below:



---

---

# Questions?

# Today's Outline

---

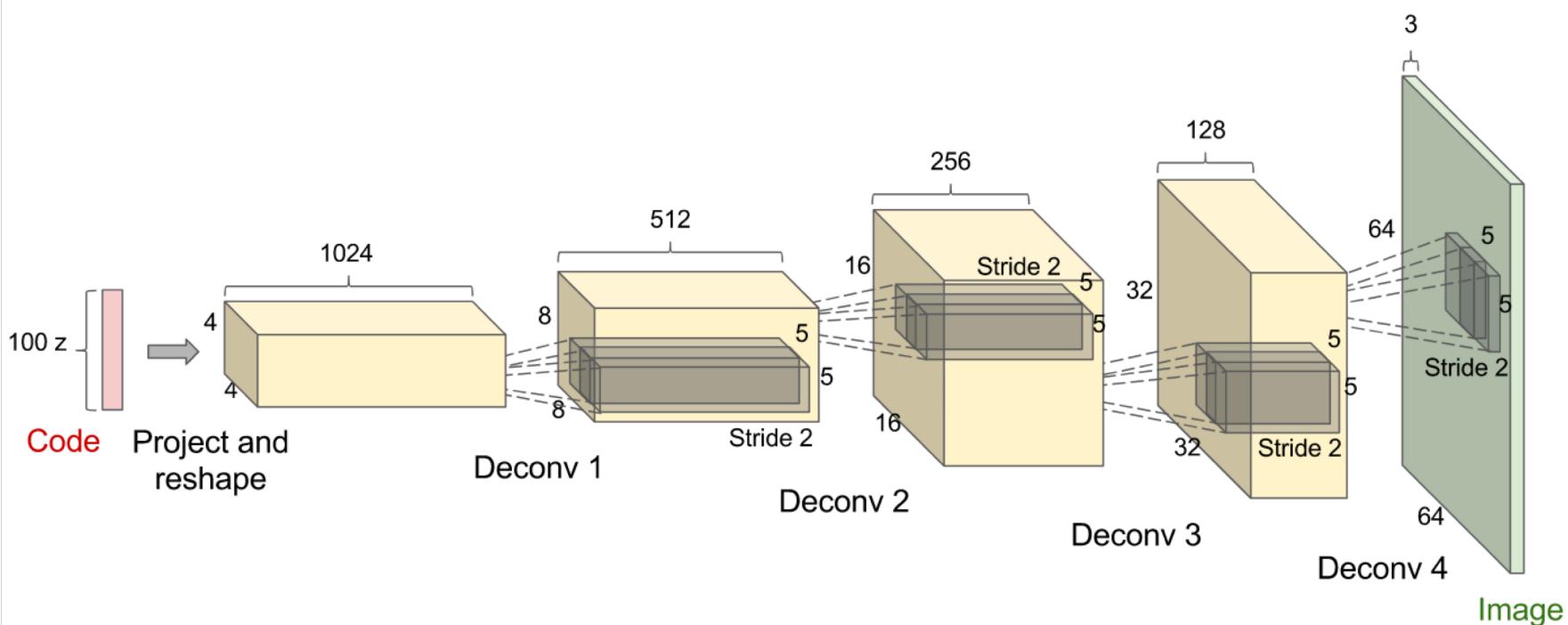
- Unsupervised Learning Landscape
- Autoencoders and Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)

---

# Autoencoders and Variational Autoencoders

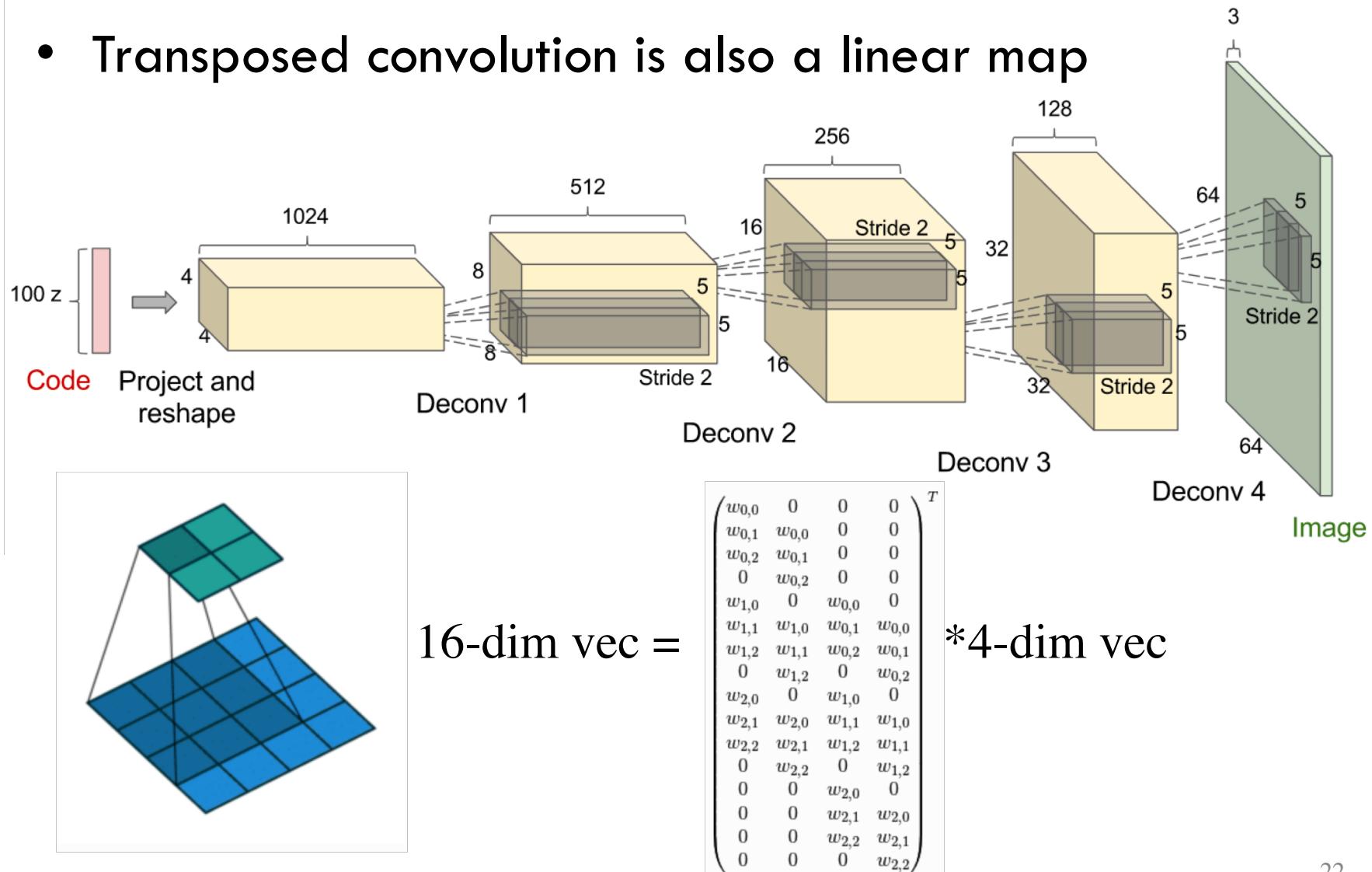
# Neural Net as a Transformation Map

- NN is a function that maps an input to output
- Here is a ~~deconvolutional~~/transposed-convolutional network



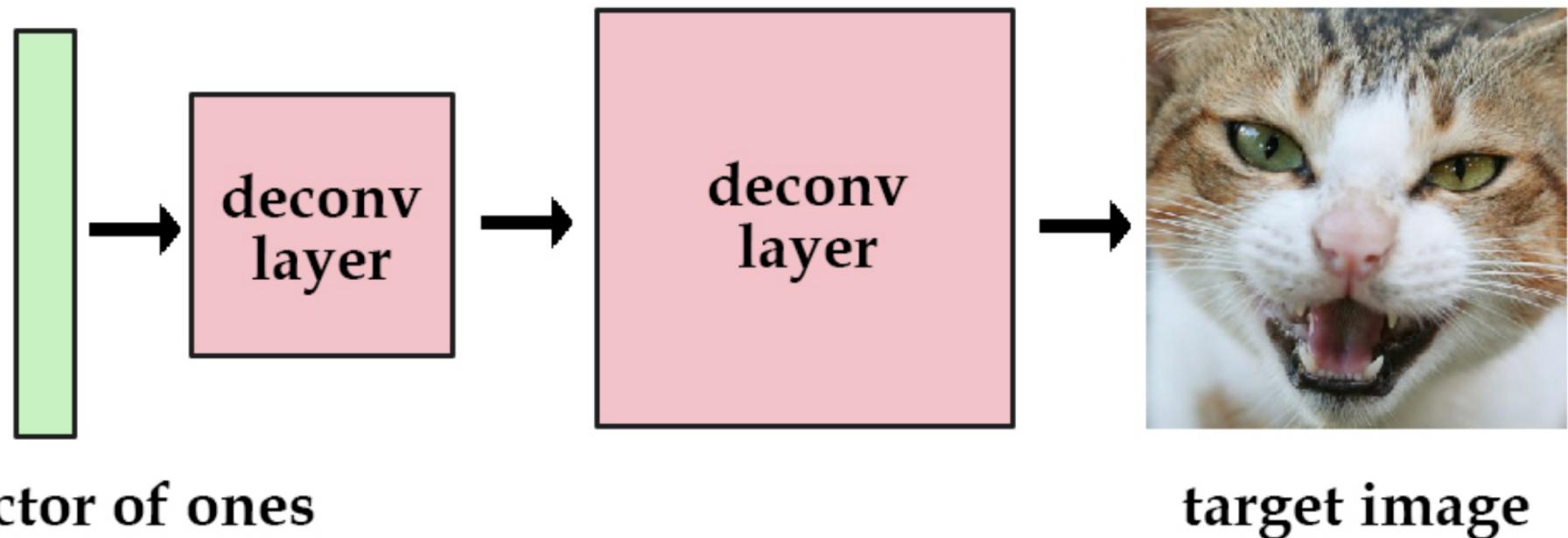
# Neural Net as a Transformation Map

- Transposed convolution is also a linear map



# Transformation from a Single Vector

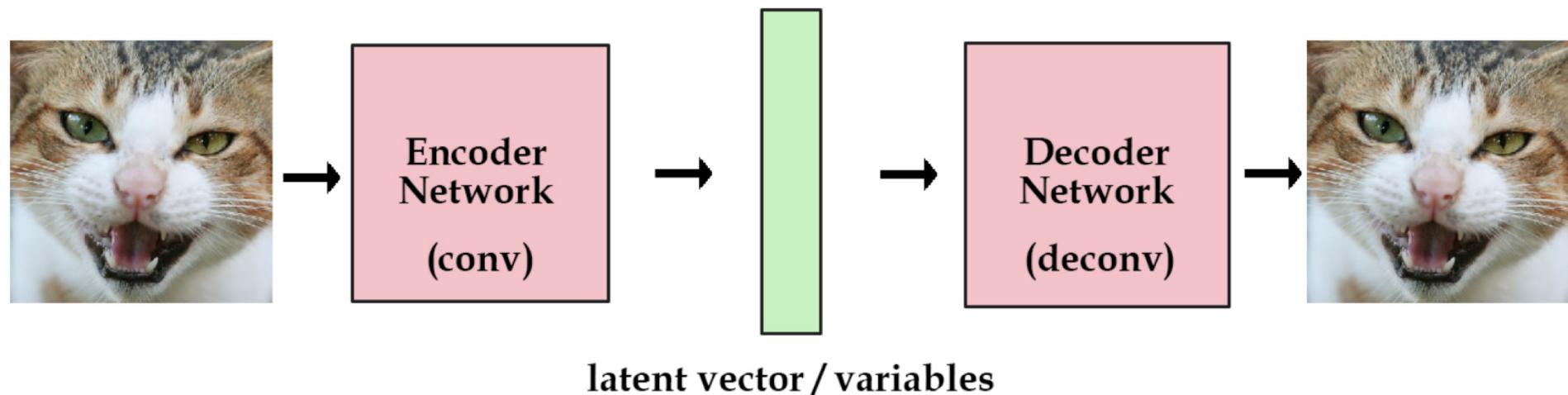
- For example, set inputs to all ones
- Train network to reduce MSE between its output and target image
- Then information related to image is captured in network parameters



<sup>1</sup>Reference: <http://kvfrans.com/variational-autoencoders-explained/>

# Transformation from Multiple Vectors

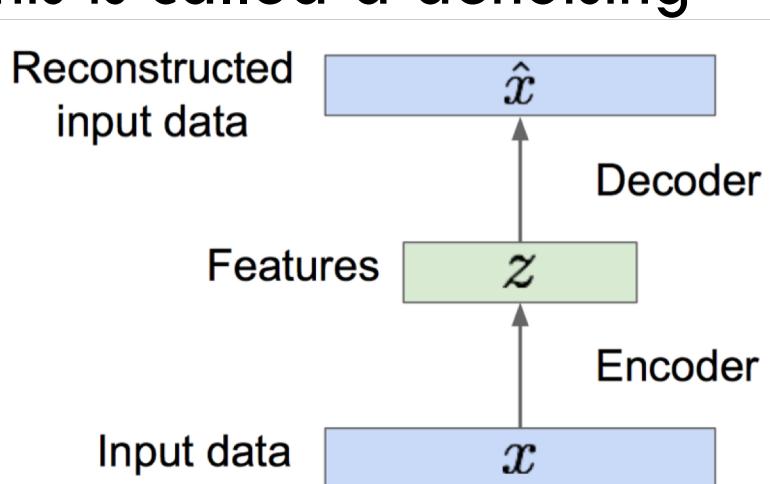
- Do the same with multiple input vectors (e.g., one hot encoded)
- These input vectors are called codes. The network is called a decoder.
- In an autoencoder, we also have an ‘encoder’ that takes original images and ‘codes’ them



<sup>1</sup>Reference: <http://kvfrans.com/variational-autoencoders-explained/>

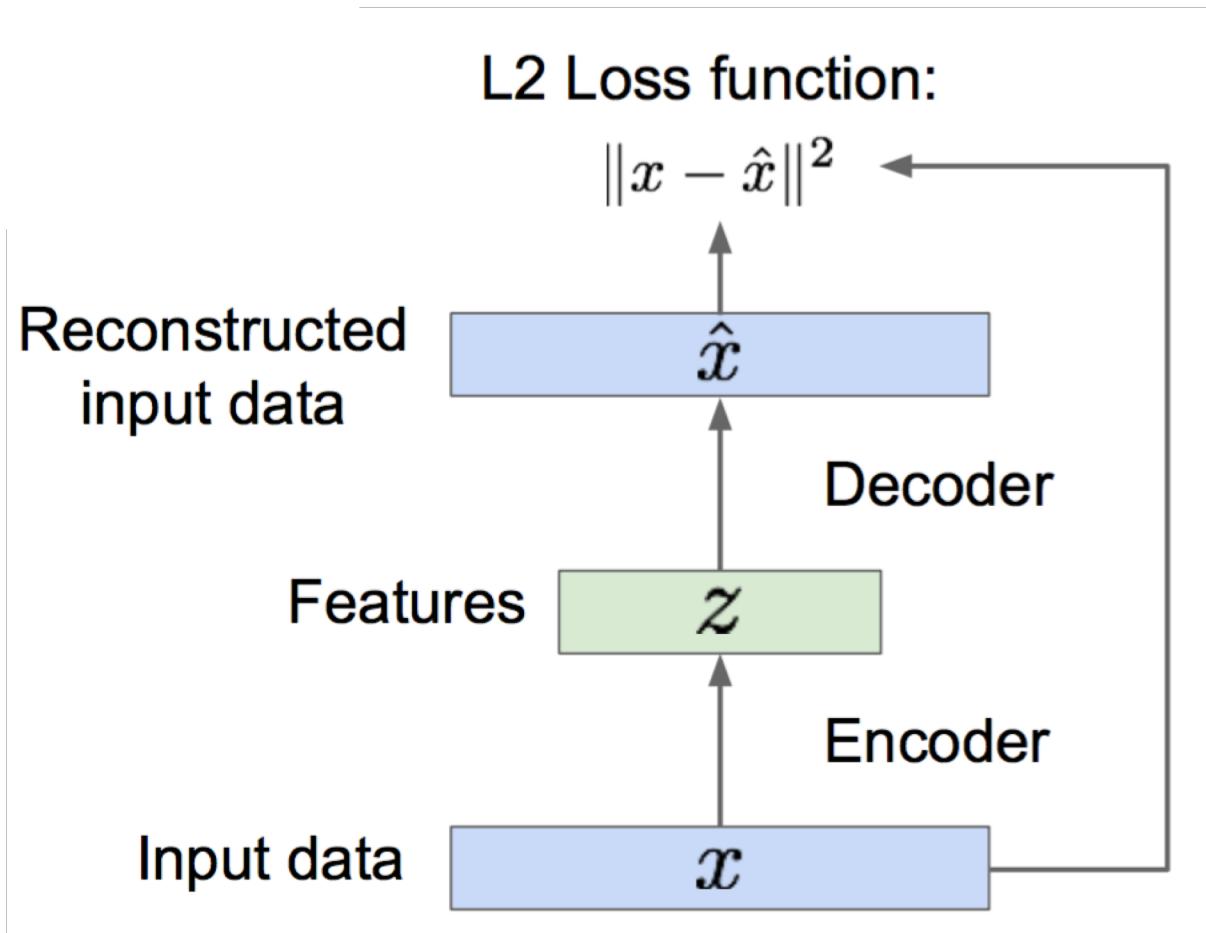
# Autoencoder: The Objective

- Captures information in training data
- The latent variable  $z$  (also called code) can be thought of as embedding
- Keep the dimension of  $z$  smaller than input  $x$ , otherwise we have a trivial solution
  - If we choose a larger dimension, add noise to  $x$  during training (this is called a denoising autoencoder)

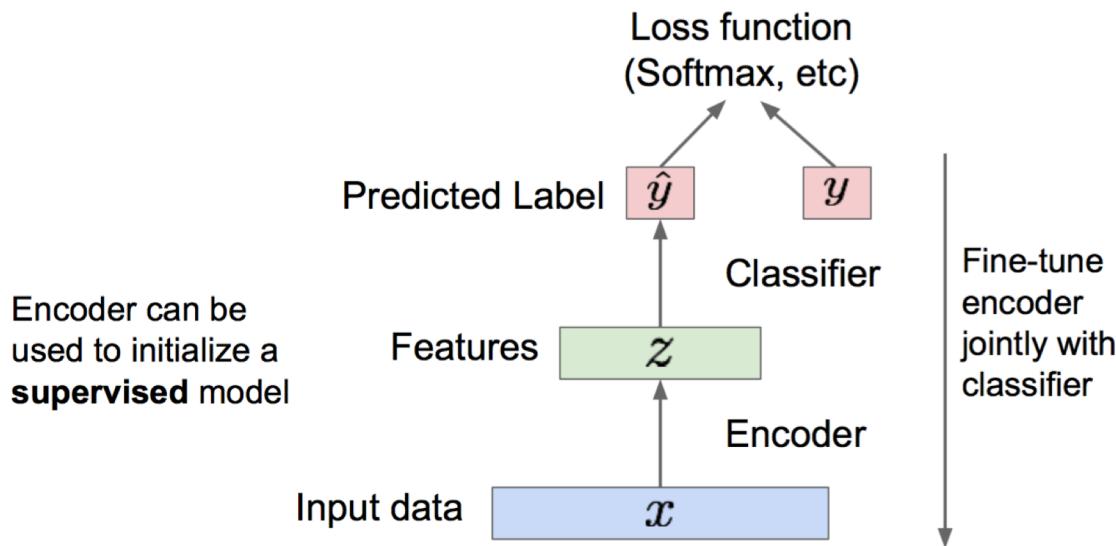


# Autoencoder: The Architecture

- No labels are needed here



# Autoencoder: Uses



- Reduction in dimension achieved by the encoder is useful
  - Just like PCA
  - Captures meaningful variations in the data via the embeddings
- Named ‘autoencoder’ because it attempts to reconstructs original data
- **Cannot generate new samples yet!**

# Variational Autoencoder

---

- Probabilistic extension of autoencoding
- The intuitive idea is to make  $z$  random, and in particular make  $P_z$  a Gaussian
  - If we can manage this, then we can sample from  $P_z$  and generate new images
- Two high level changes needed
  - Architecture
  - Loss function

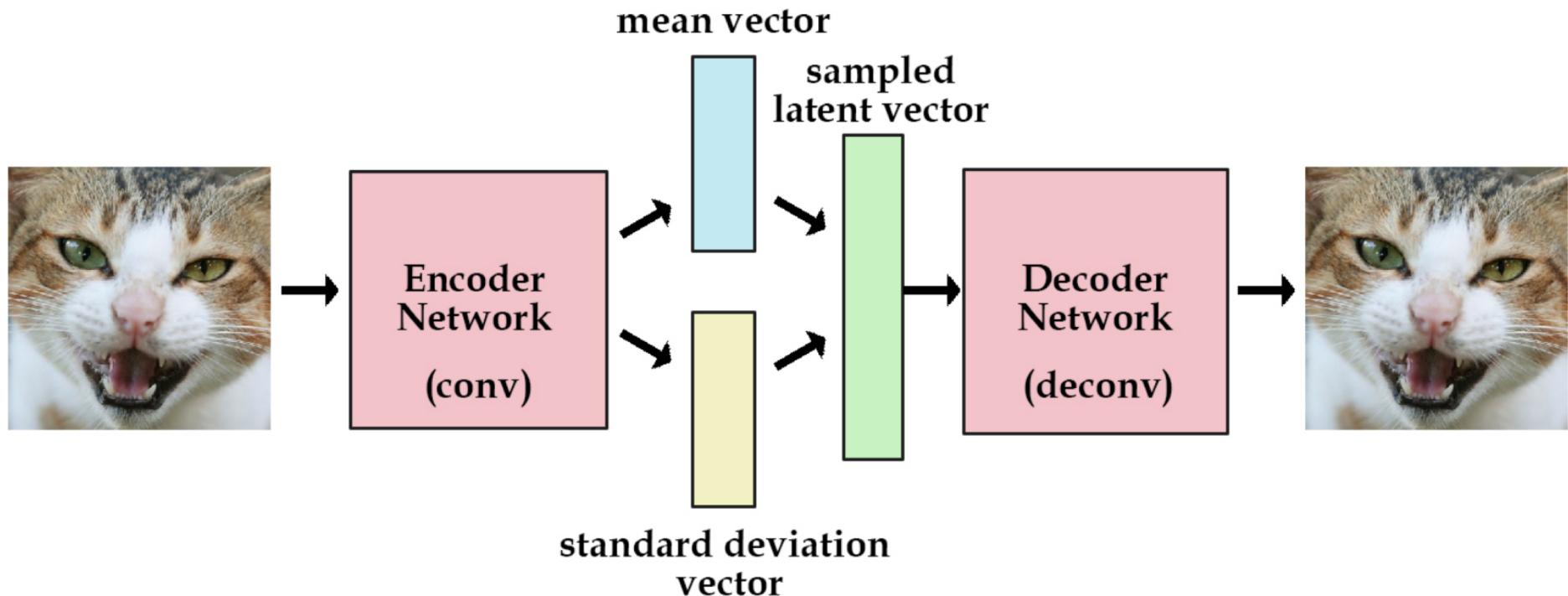
# Variational Autoencoder: Loss

---

- Loss will be sum of two losses
  - Reconstruction loss
  - Latent loss (how far from Gaussian the distribution obtained from encoder is)
    - Measured using KL divergence
    - Encoder generates the mean and covariance of the Gaussian
- We will look at the math behind this shortly

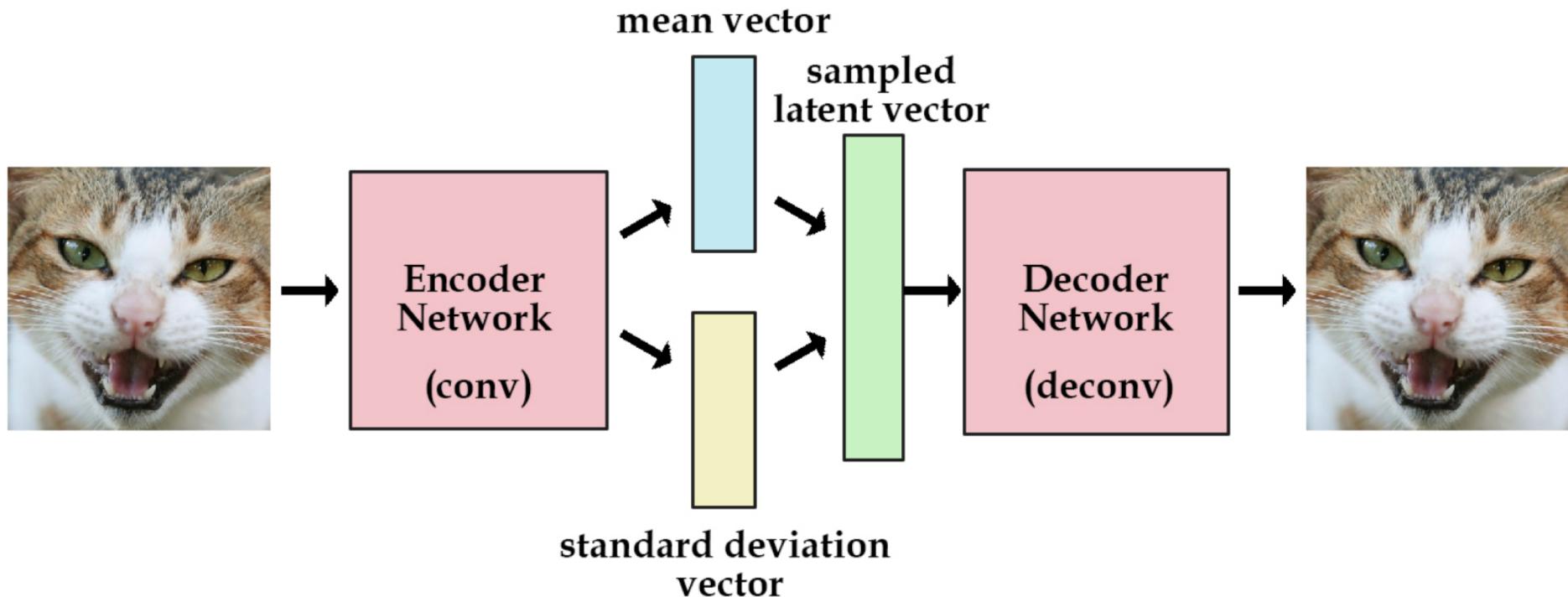
# Variational Autoencoder: Architecture

- Architecture involves a sampling in between



# Variational Autoencoder: Architecture

- Architecture involves a sampling in between
- Can still backprop given realized sample



<sup>1</sup>Reference: <http://kvfrans.com/variational-autoencoders-explained/>

# Variational Autoencoder: Generalization

---

- This sampling allows for generalization
  - Gaussian noise ensures we are not remembering only the training data
- Once we have trained, we can sample from a Gaussian and pass it through the decoder to get a new image

# Variational Autoencoder: Samples

- Experiments on MNIST
  - Samples generated during training (left, center) and original data



# VAE: Derivation

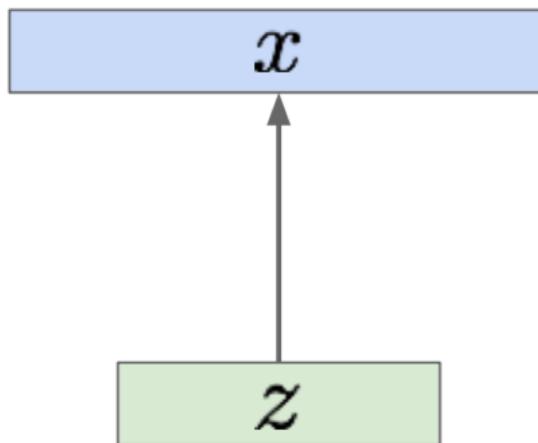
- Assume a model as below
- Variable  $x$  represents image,  $z$  represents the latent variable
- We want to estimate  $\theta^*$

Sample from  
true conditional

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

Sample from  
true prior

$$p_{\theta^*}(z)$$



# VAE: Derivation

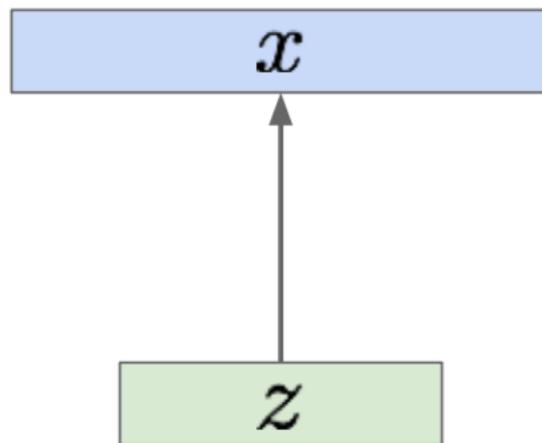
- Let  $P_z$  be Gaussian
- Let  $P(x|z)$  be a neural network: decoder
- We can train by maximizing likelihood of training data  $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$

Sample from  
true conditional

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

Sample from  
true prior

$$p_{\theta^*}(z)$$



# VAE: Derivation

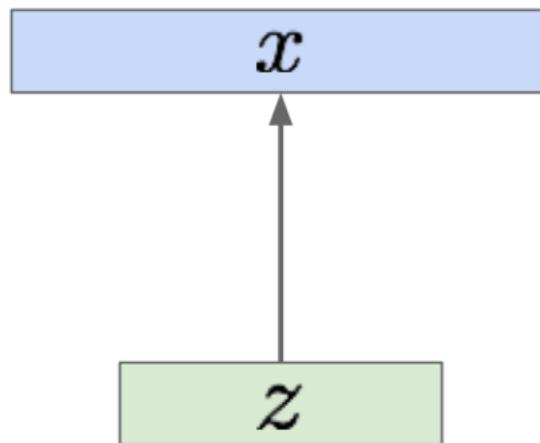
- Let  $P_z$  be Gaussian
- Let  $P(x|z)$  be a neural network: decoder
- We can train by maximizing likelihood of training data  $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$

Sample from  
true conditional

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

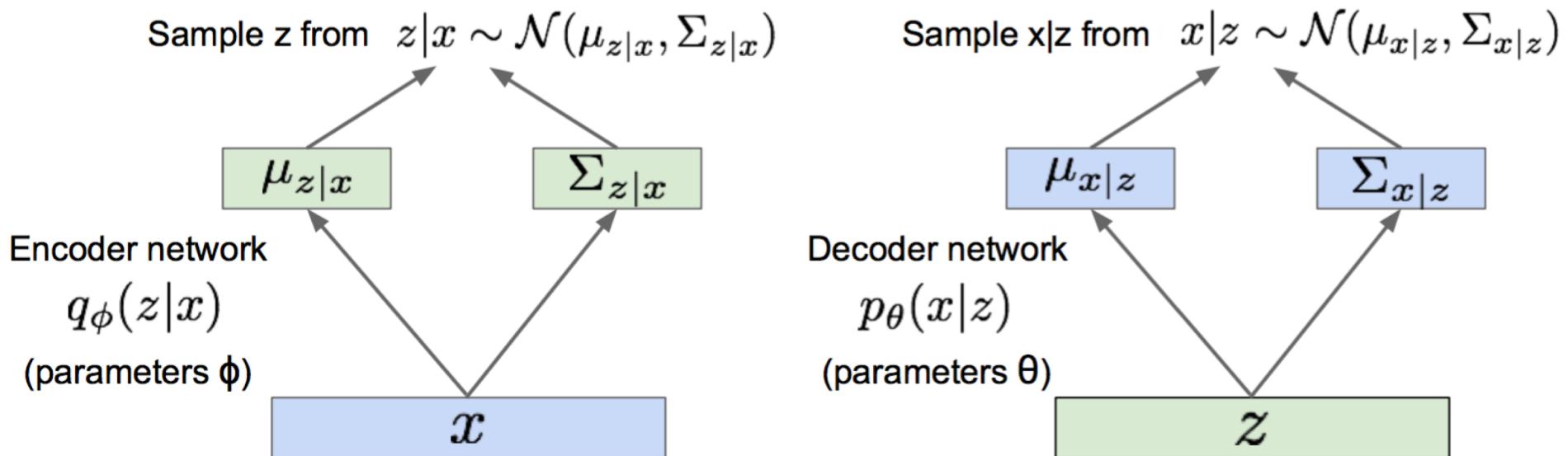
Sample from  
true prior

$$p_{\theta^*}(z)$$



# VAE: Derivation

- We will also make the encoder probabilistic



# Aside: Notion of Information

---

- Information:  $-\log P(x)$
- Entropy:  $-\sum P(x) \log P(x)$
- KL divergence:
  - A notion of dissimilarity between two distributions
  - $D_{KL}(p||q) = \sum P(x) \log \frac{P(x)}{Q(x)}$

# VAE: Derivation

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} [\log p_\theta(x^{(i)})] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

# VAE: Derivation

$$\begin{aligned}\log p_\theta(x^{(i)}) &= \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} [\log p_\theta(x^{(i)})] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\ &= \mathbf{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})\end{aligned}$$

# VAE: Derivation

$$\begin{aligned}\log p_\theta(x^{(i)}) &= \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} [\log p_\theta(x^{(i)})] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\ &= \mathbf{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad (\text{Bayes' Rule}) \\ &= \mathbf{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \right] \quad (\text{Multiply by constant})\end{aligned}$$

# VAE: Derivation

$$\begin{aligned}\log p_\theta(x^{(i)}) &= \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\ &= \mathbf{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad (\text{Bayes' Rule}) \\ &= \mathbf{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \right] \quad (\text{Multiply by constant}) \\ &= \mathbf{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbf{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbf{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad (\text{Logarithms})\end{aligned}$$

# VAE: Derivation

$$\begin{aligned}\log p_\theta(x^{(i)}) &= \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\ &= \mathbf{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad (\text{Bayes' Rule}) \\ &= \mathbf{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \right] \quad (\text{Multiply by constant}) \\ &= \mathbf{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbf{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbf{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad (\text{Logarithms}) \\ &= \mathbf{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z)) + D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z | x^{(i)}))\end{aligned}$$

# VAE: Derivation

- The first two terms constitute a lower bound for the data likelihood that we can maximize tractably

$$= \underbrace{\mathbf{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z | x^{(i)}))}_{> 0}$$

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (“ELBO”)

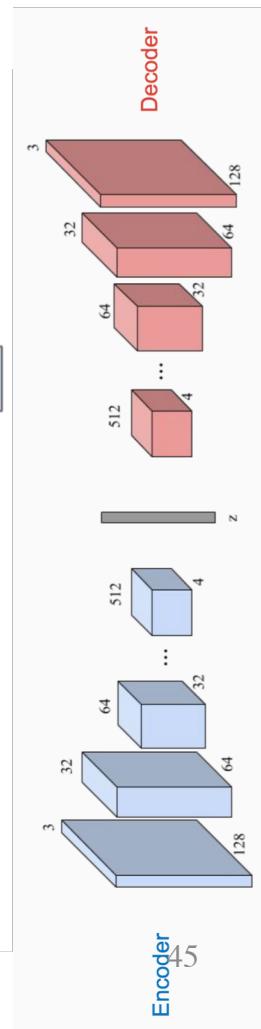
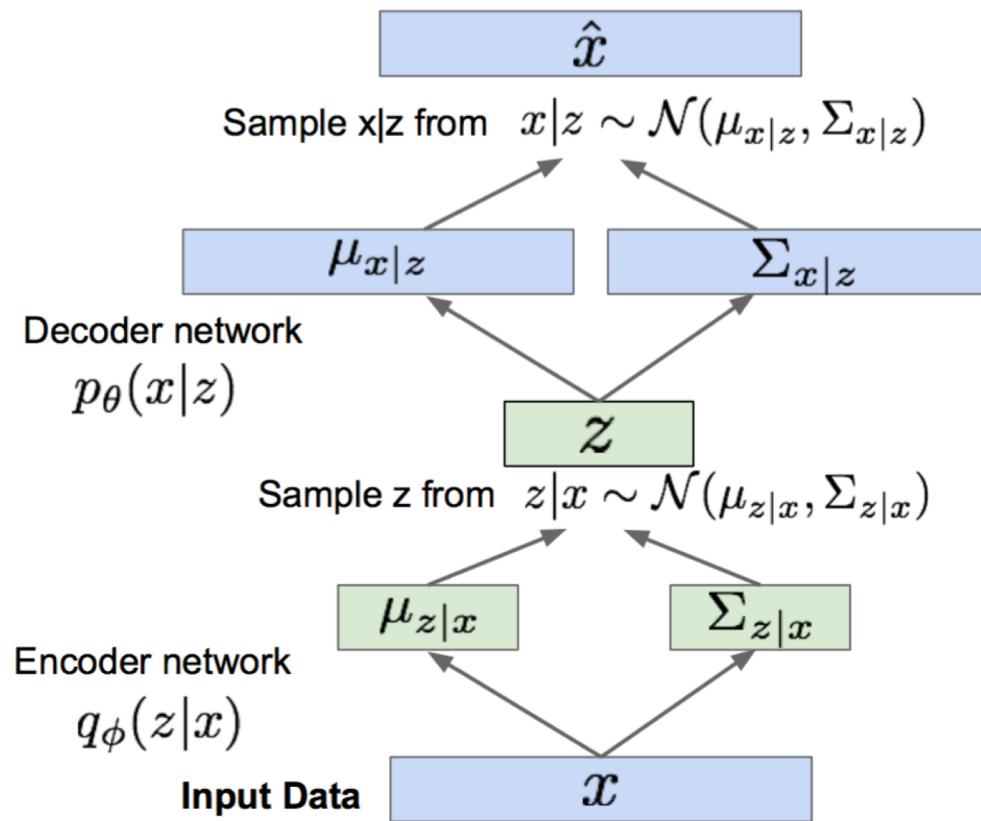
$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^N \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

- The first term of  $\mathcal{L}$  is essentially reconstruction error
- The second term of  $\mathcal{L}$  is making the encoder network close to Gaussian prior

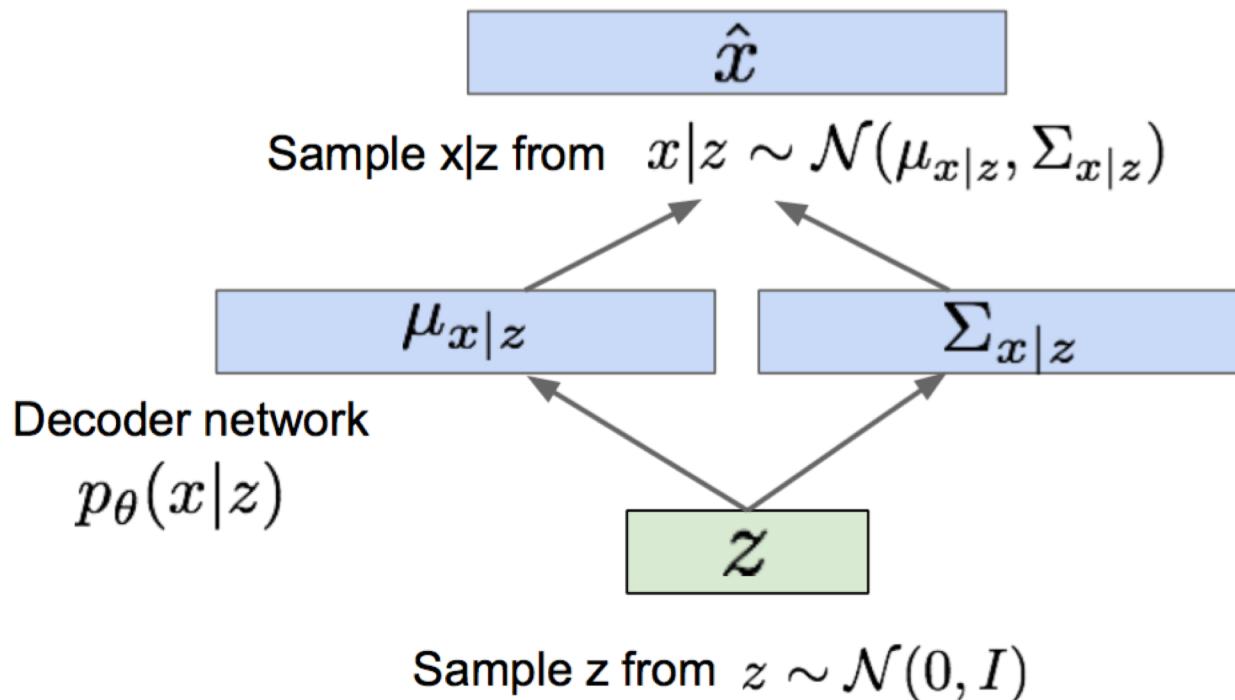
# VAE: Derivation

- In summary,



# VAE: Samples

- We can create new samples!



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

# VAE: Experiments

- Some generated samples



Further reading: <https://arxiv.org/pdf/1606.05908.pdf>

---

---

# Questions?

# Today's Outline

---

- Unsupervised Learning Landscape
- Autoencoders and Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)

---

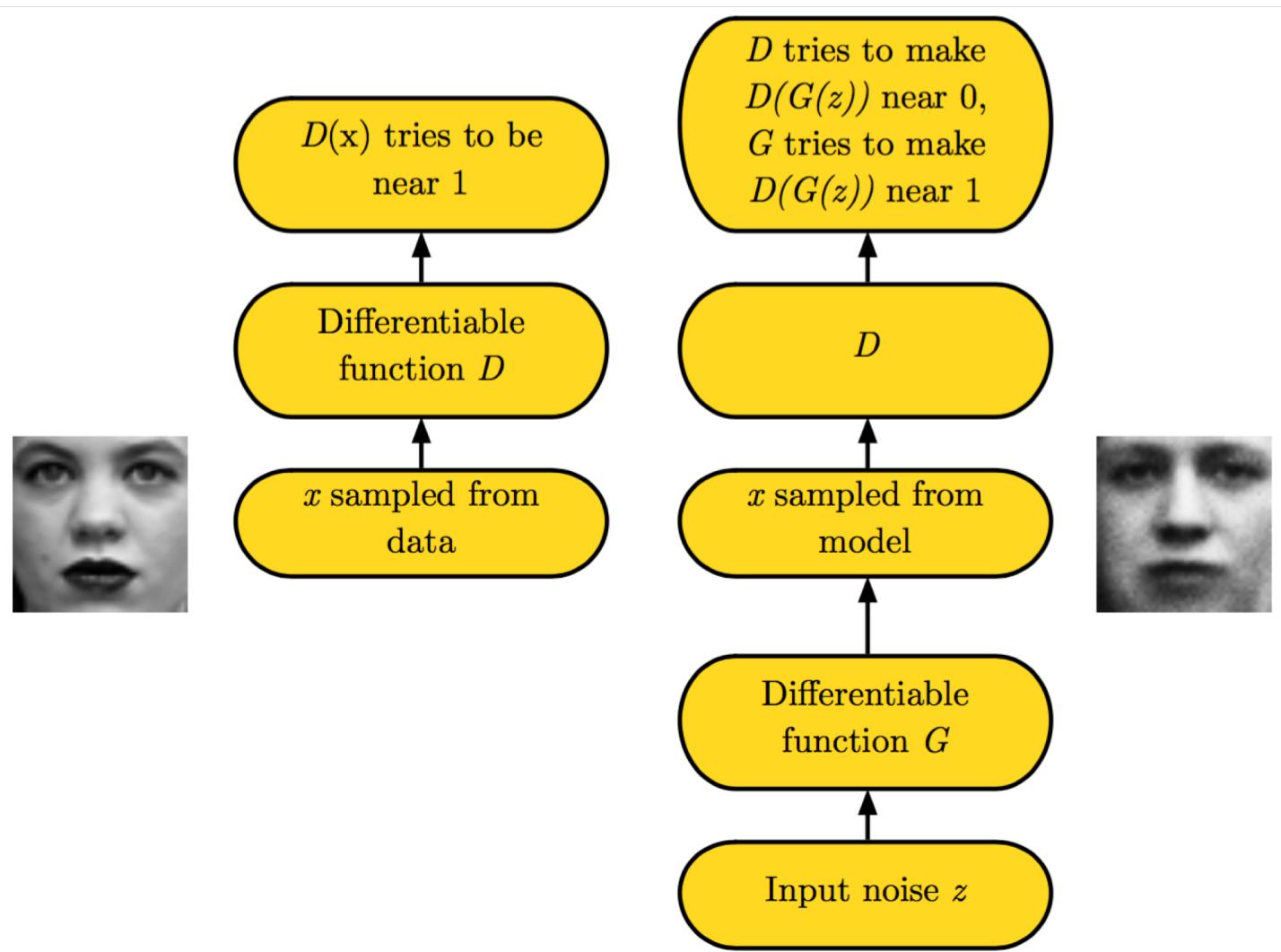
# Generative Adversarial Networks

# GANs: Two Scenarios

---

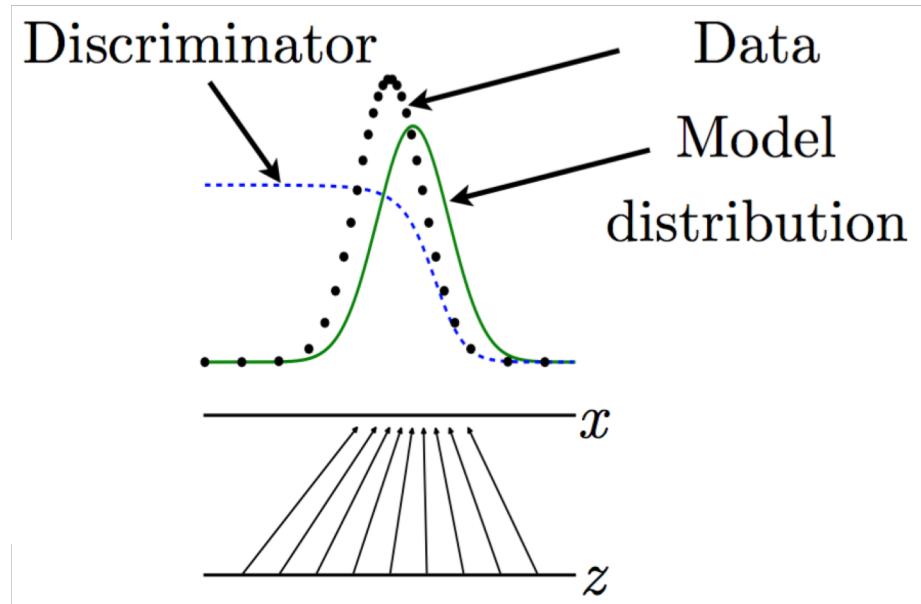
- Overall Idea: Instead of working with an explicit density function, GANs take an ‘adversarial’ or ‘game-theoretic’ approach

# GANs: Two Scenarios



# The Generator and the Discriminator

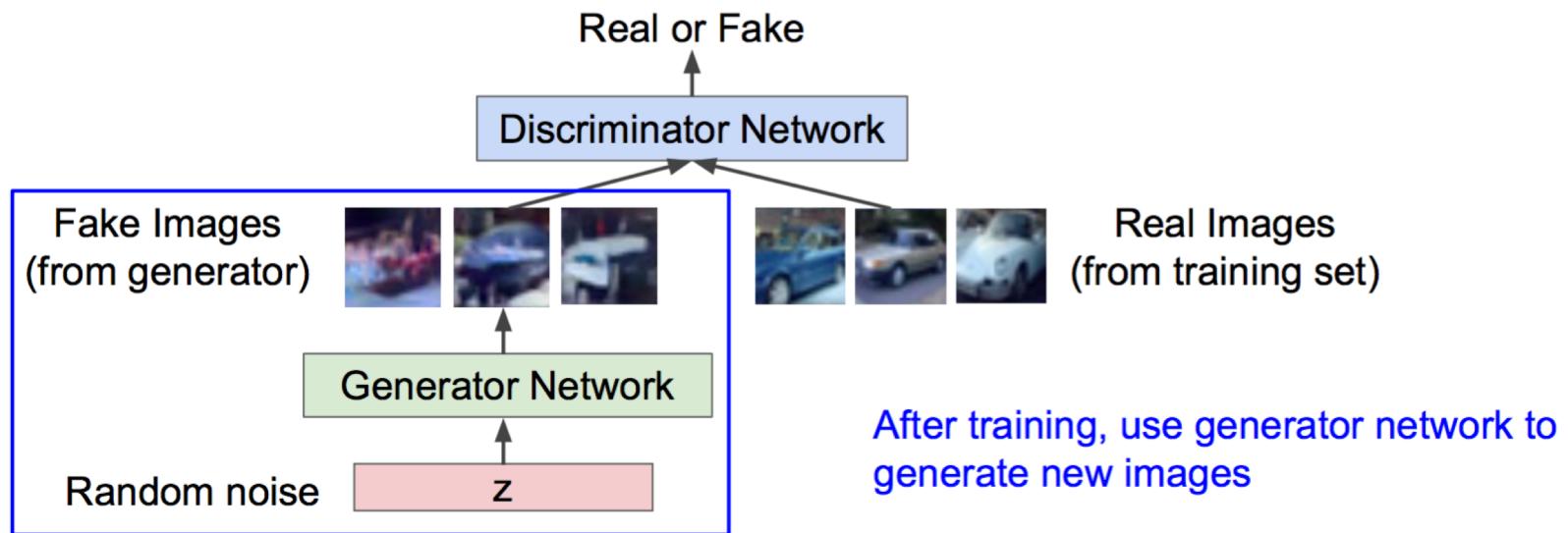
- Assume  $X = G_{\theta_g}(z)$
- Differentiable
- $D_{\theta_d}(X)$  takes values in  $\{0,1\}$



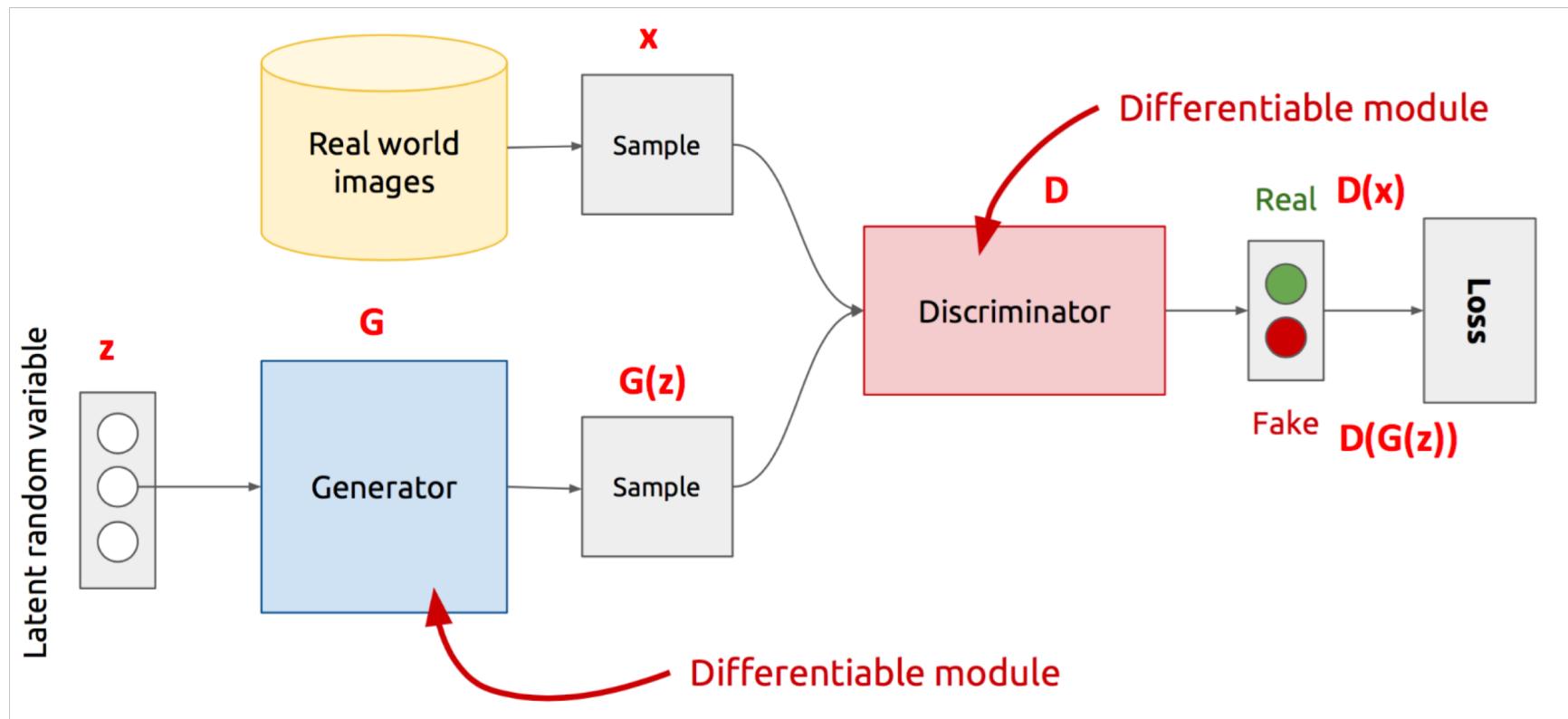
# The Generator and the Discriminator

**Generator network:** try to fool the discriminator by generating real-looking images

**Discriminator network:** try to distinguish between real and fake images



# The Generator and the Discriminator



# The Objectives

---

- The generator and the discriminator are playing a minimax game.
- $J(D) = -E_{P_d} \log D(x) - E_{P_m} \log(1 - D(x))$ 
  - Where  $P_m(x)$  is the derived distribution using  $G(z)$  and  $P_z$
- $J(G) = -J(D)$

# The Objectives

---

- The optimal strategy for the discriminator at equilibrium is

- $$D(x) = \frac{P_d(x)}{P_d(x)+P_m(x)}$$

# The Objectives

---

- The optimal strategy for the discriminator at equilibrium is
  - $D(x) = \frac{P_d(x)}{P_d(x)+P_m(x)}$
- The optimal strategy for the generator is to find parameters such that
  - $P_d = P_m$

# The Training Procedure

---

- Create a minibatch of real data
- Create a minibatch of generated data
- Score the discriminator
- Backprop to update the parameter  $\theta_d$
- Score the generator
- Backprop to update the parameter  $\theta_g$

# The Training Procedure

Minimax objective function:

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

Alternate between:

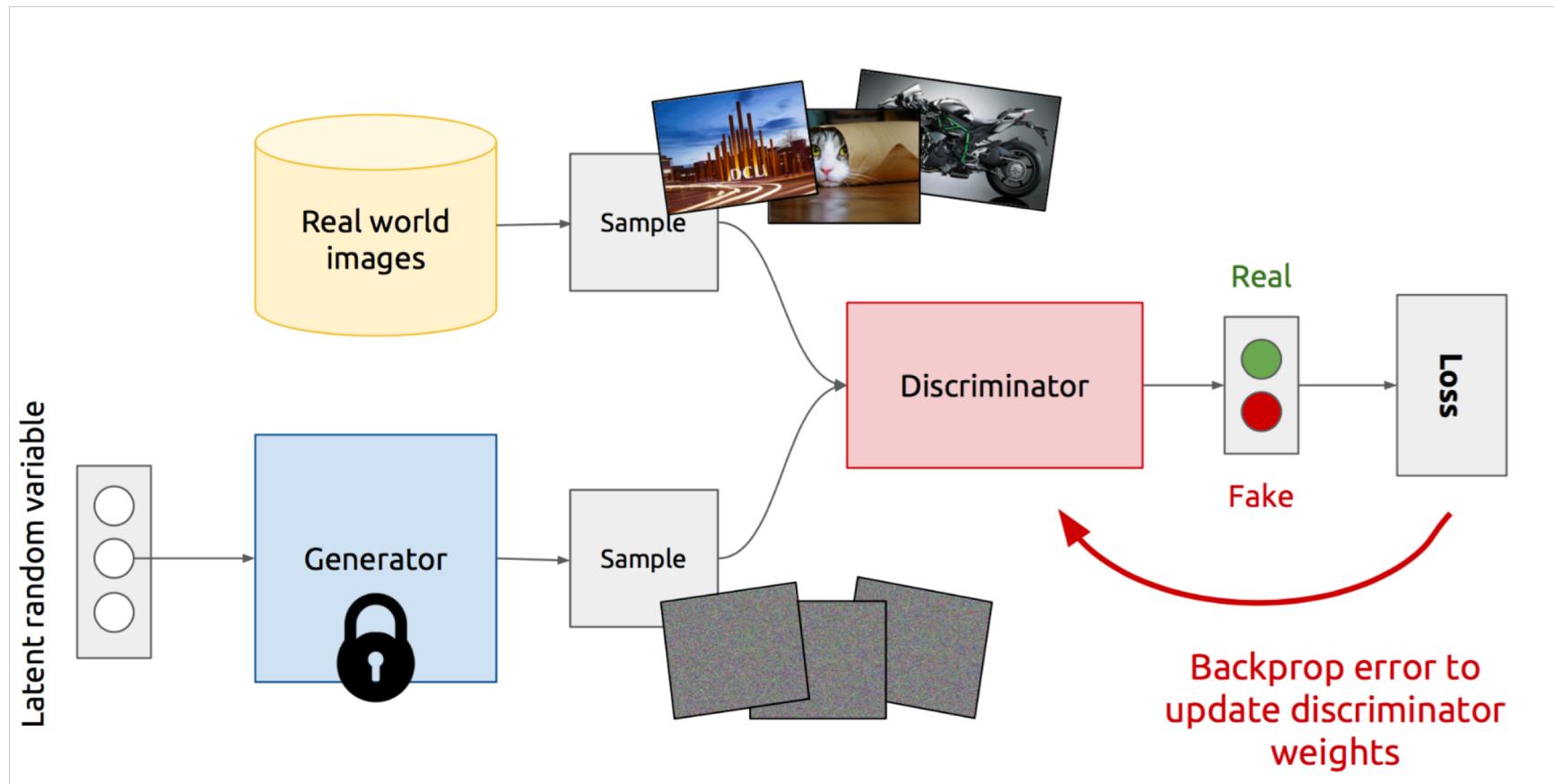
1. **Gradient ascent** on discriminator

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

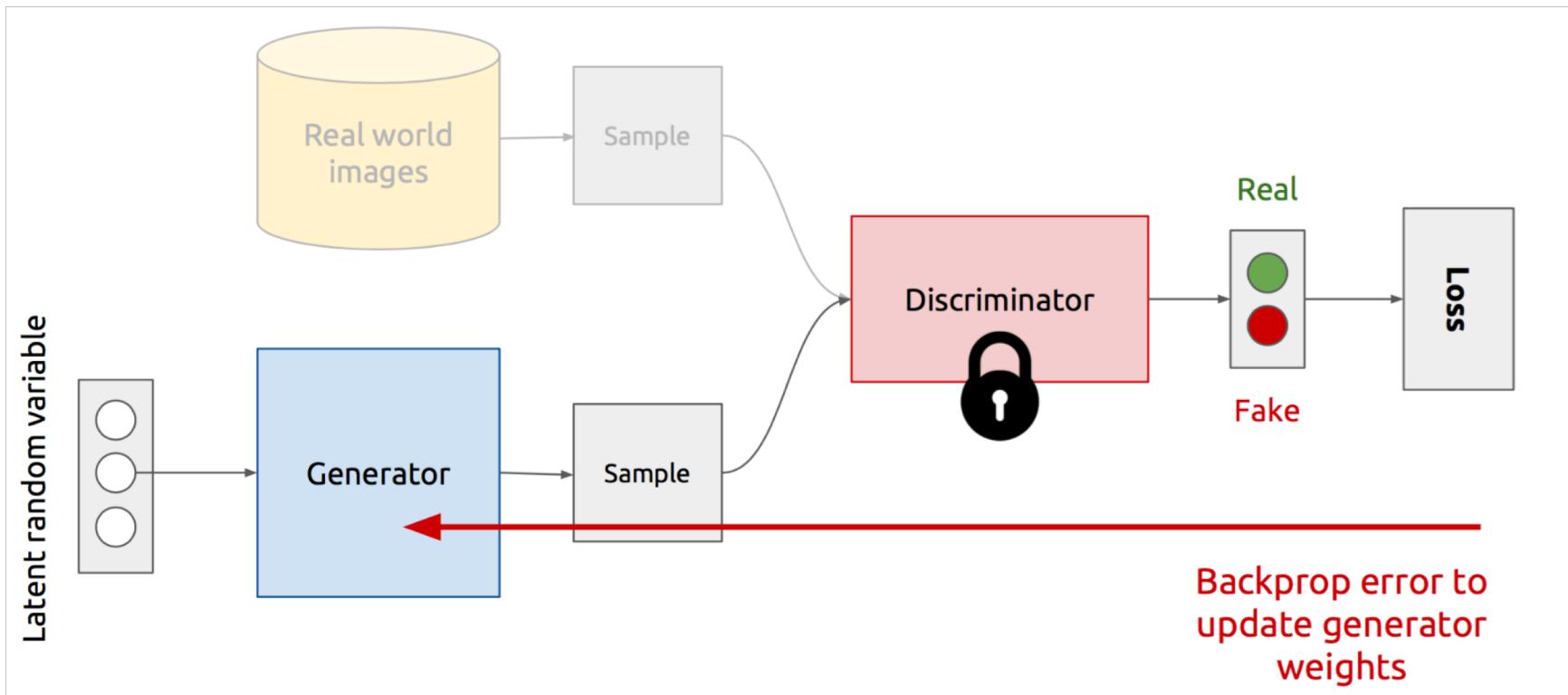
2. **Gradient descent** on generator

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

# The Training Procedure

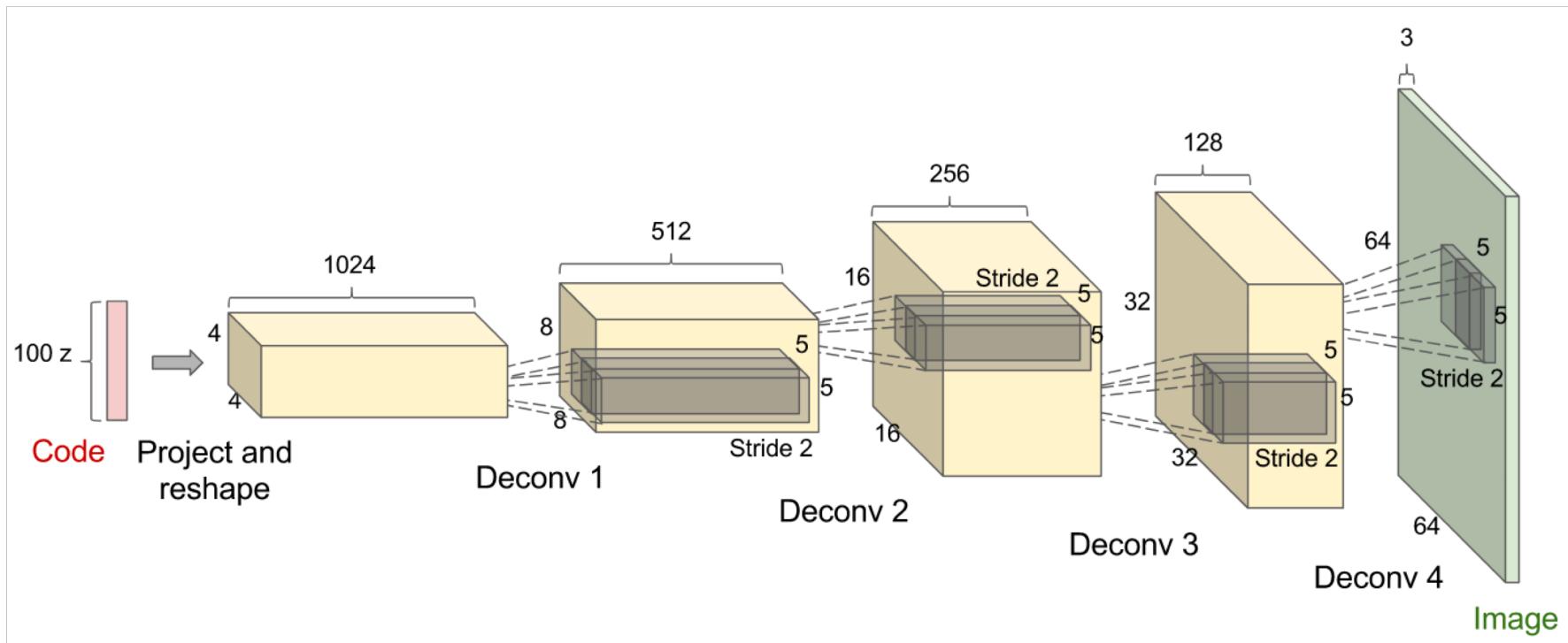


# The Training Procedure



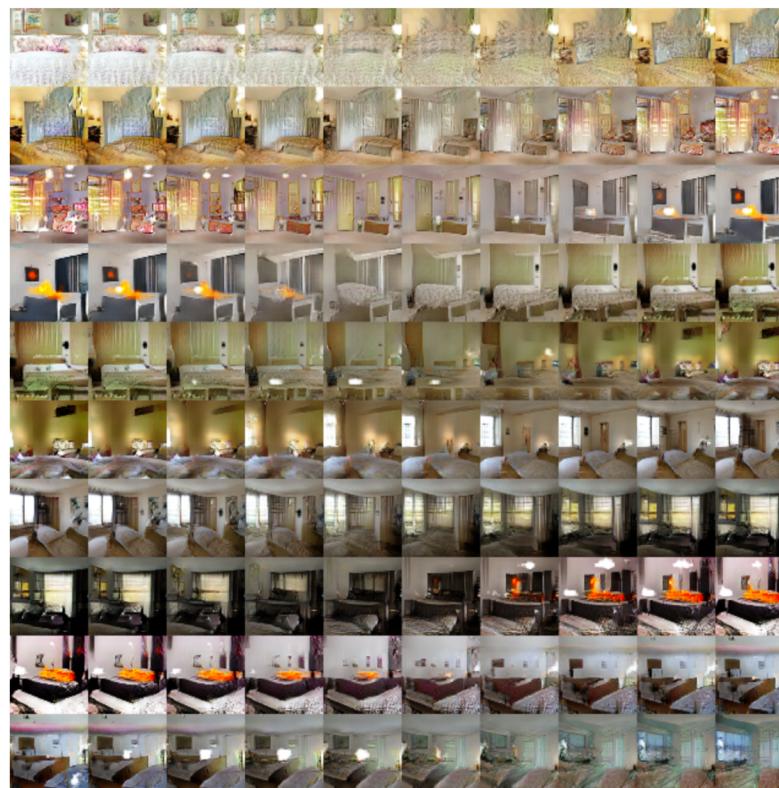
# Example Generator Architecture

- DCGAN

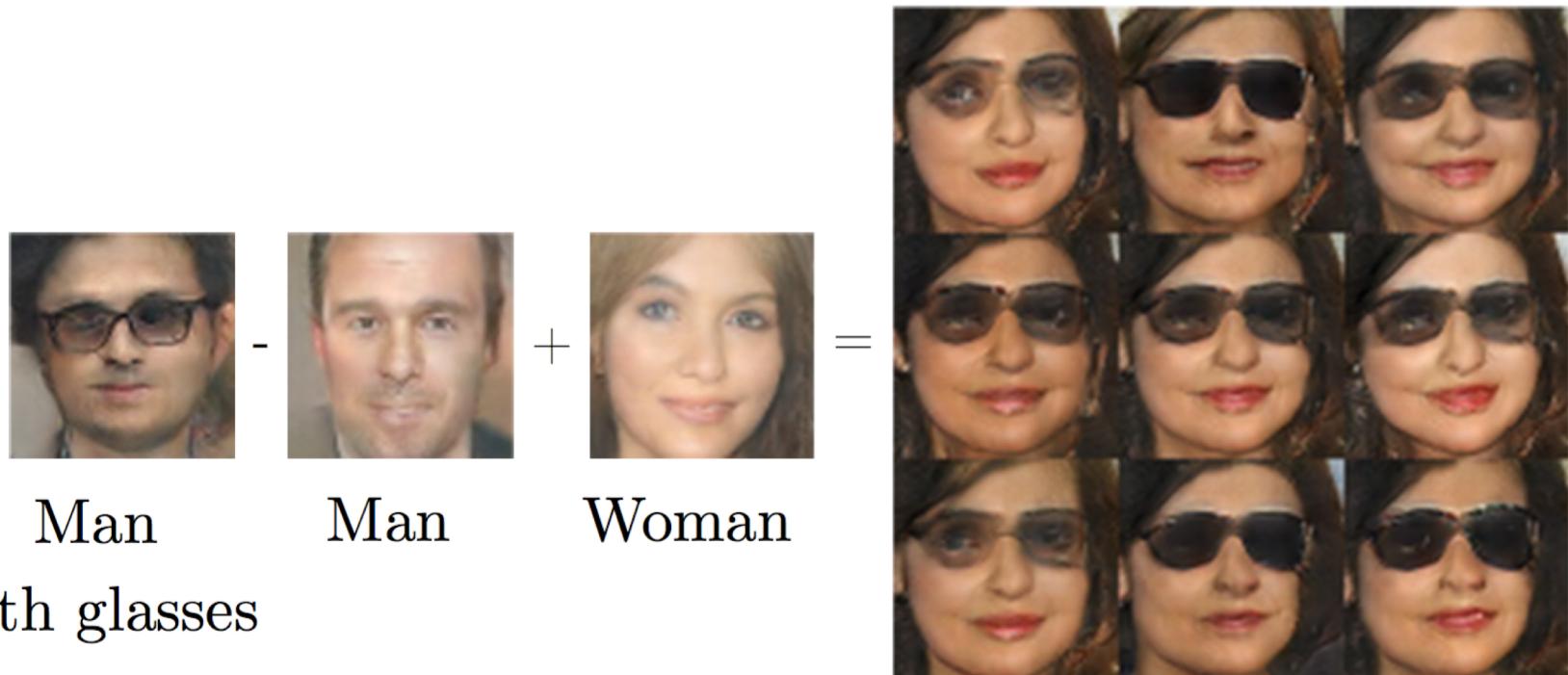


# GAN Properties: Latent Space

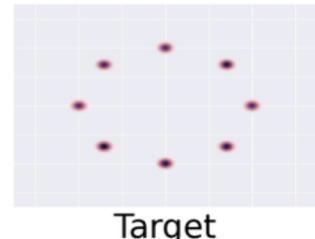
- Consider Deep Convolutional Generative Adversarial Network (DCGAN)
  - You can walk from one point to another in the bedroom latent space (e.g., 6<sup>th</sup> and 10<sup>th</sup> rows)



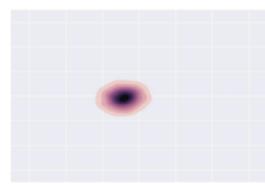
# GAN Properties: Latent Space Arithmetic as a Byproduct



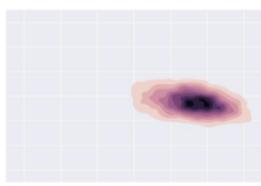
# GAN Properties: Mode Collapse Issue



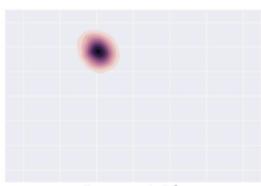
Target



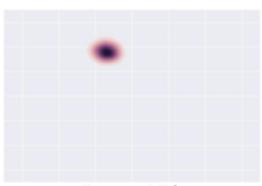
Step 0



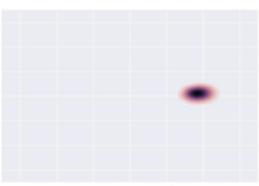
Step 5k



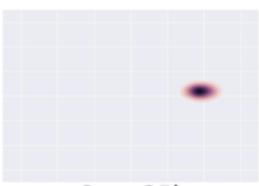
Step 10k



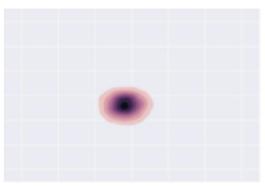
Step 15k



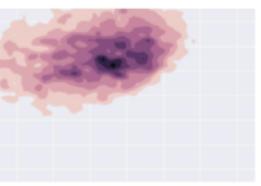
Step 20k



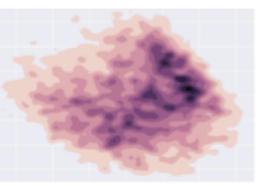
Step 25k



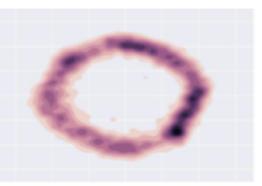
Step 0



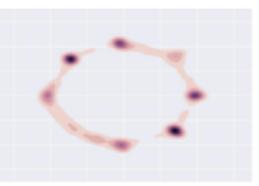
Step 5k



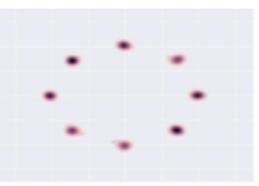
Step 10k



Step 15k



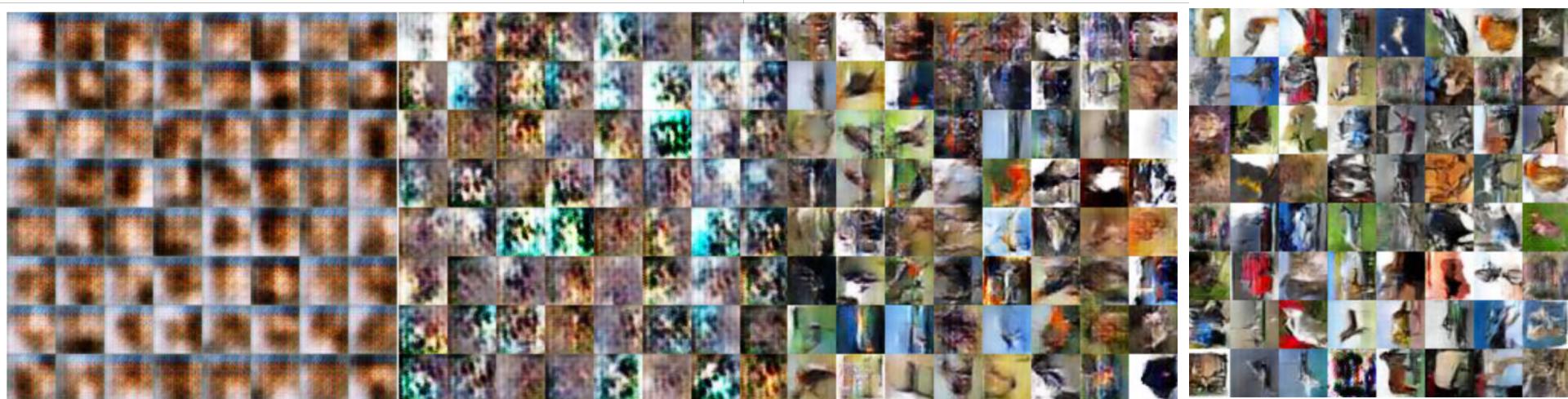
Step 20k



Step 25k

# GAN: Experiments

- Experiments on CIFAR-10 (only generated images below)
  - Code: <https://github.com/kvfrans/generative-adversarial>



---

---

# Questions?

# VAE and GAN

---

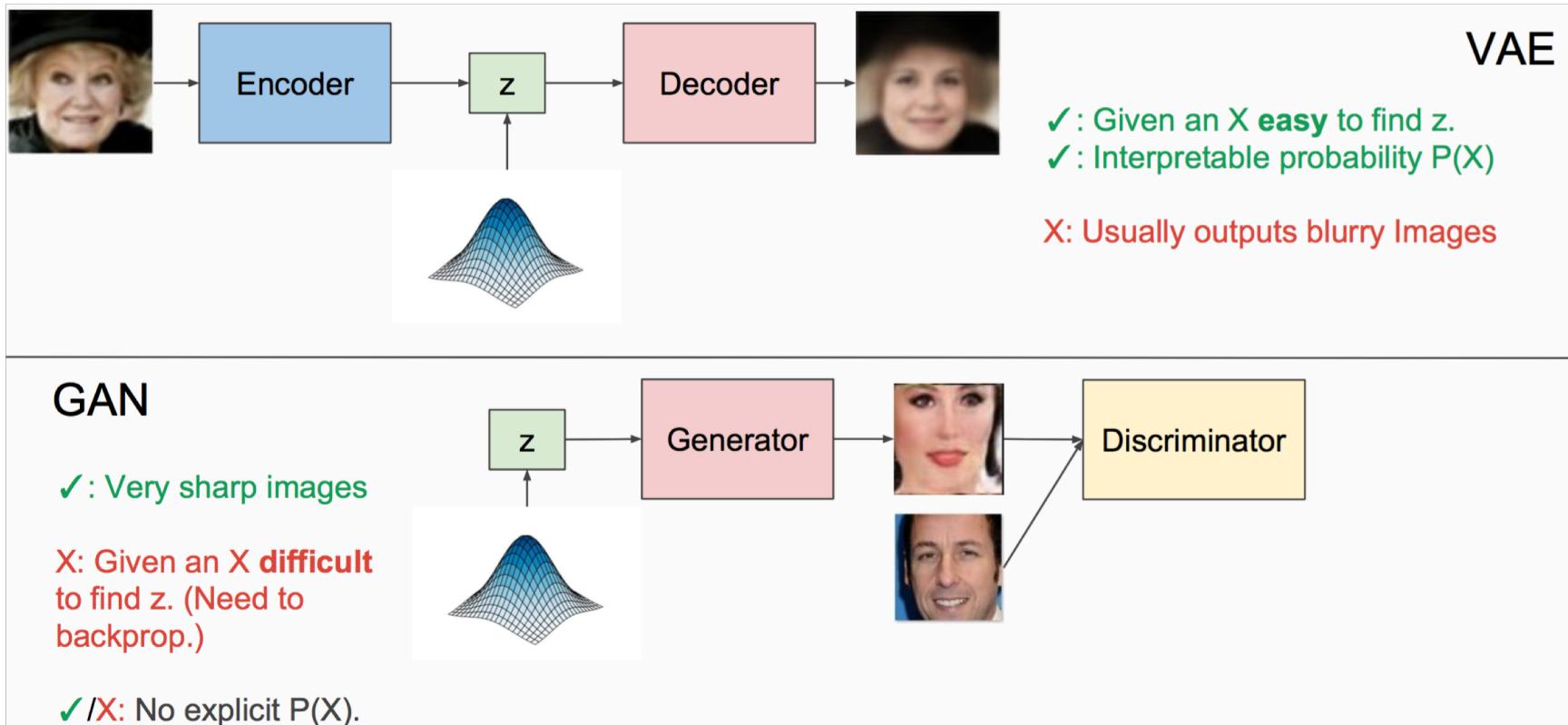
- VAEs
  - Are generative models that use regularized log likelihood to approximate performance score
  - Tend to achieve higher likelihoods of data, but the generated samples don't have real-world properties like sharpness
  - Can compare generated images with original images, which is not possible with GANs
  - Part of graphical models with principled theory

# VAE and GAN

---

- GANs
  - Are generative models that use a supervised learning classifier to approximate performance score
    - No constraint that a ‘bed’ should look like a ‘bed’
  - Try to solve an intractable game, vastly more difficult to train
  - Tend to have sharper image samples
  - Start with latent variables and transform them deterministically
  - There is no Markov chain style of sampling required
  - They are asymptotically consistent (will converge to  $P_d$ ), whereas VAEs are not
  - Many many variations have been proposed in the past 3 years ( $>150!$ )

# VAE and GAN



# Summary

---

- Both models are recent (VAEs from 2013, GANs from 2014) and have initiated very exciting new directions in machine learning and AI
- Useful in many applications such as
  - Image denoising
  - Image Super-resolution
  - Reinforcement learning
  - Generating embeddings
  - Artistic help
- Eventually help the computer understand the world better

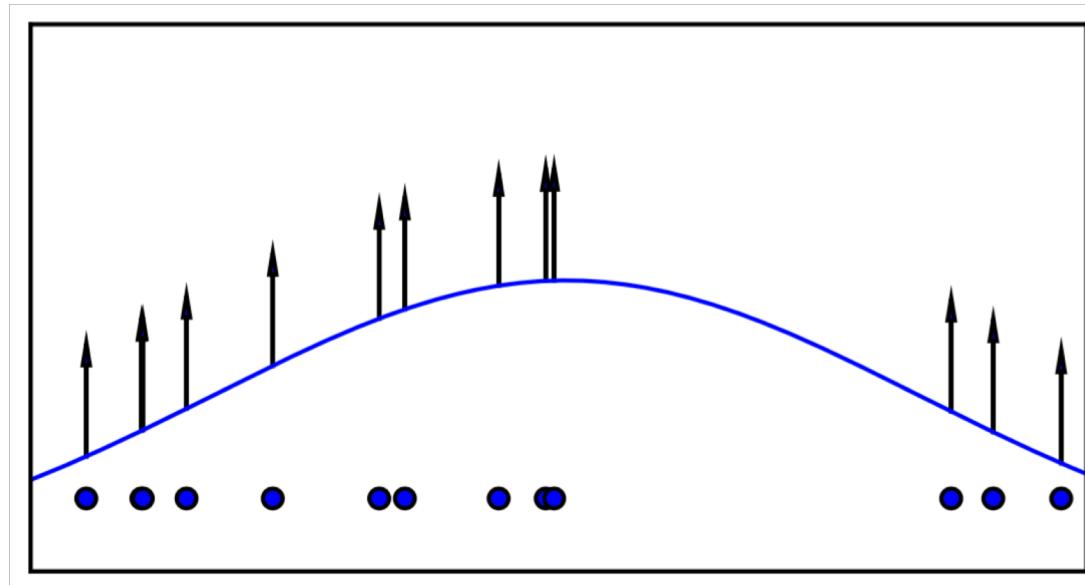
# Appendix

# Sample Exam Questions

---

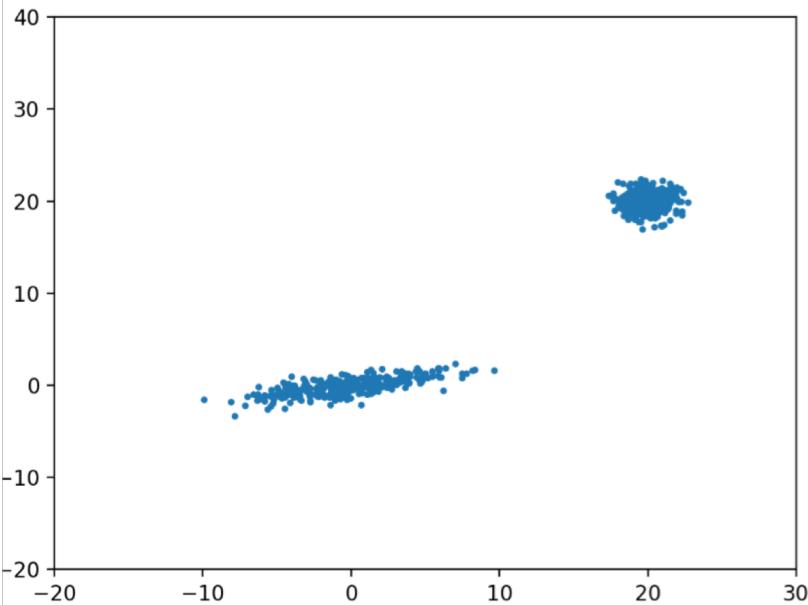
- What are the uses of generative models?
- What is the difference between a regular autoencoder and a variational autoencoder?
- What is the qualitative objective of the discriminator in a GAN? What is the qualitative objective of the generator?
- Describe some differences between a VAE model and a GAN.

# Maximum Likelihood Estimation I



# Maximum Likelihood Estimation II

**Step 1: observe a set of samples**



**Step 2: assume a GMM model**

$$p(x|\theta) = \sum_i \pi_i \mathcal{N}(x|\mu_i, \Sigma_i)$$

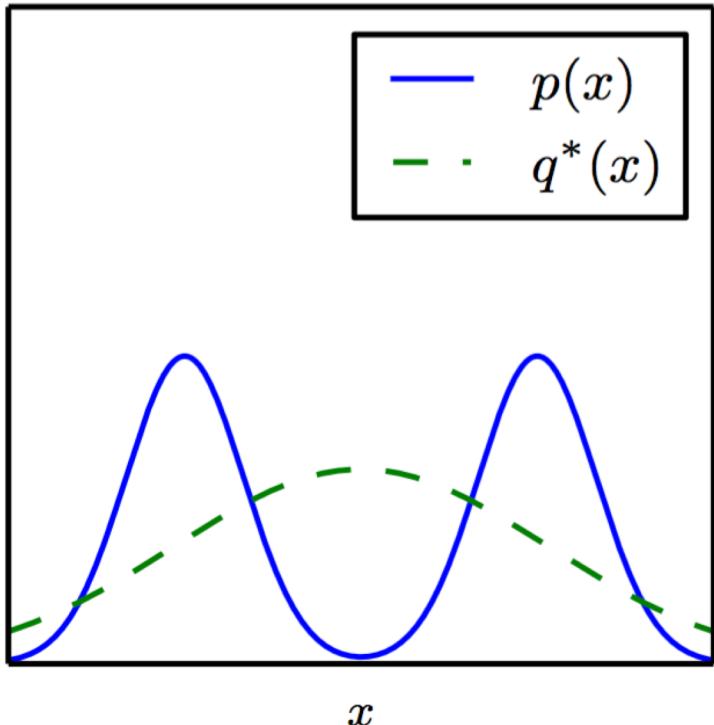
**Step 3: perform maximum likelihood learning**

$$\max_{\theta} \sum_{x^{(j)} \in \text{Dataset}} \log p(\theta|x^{(j)})$$

# KL Divergence

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(p\|q)$$

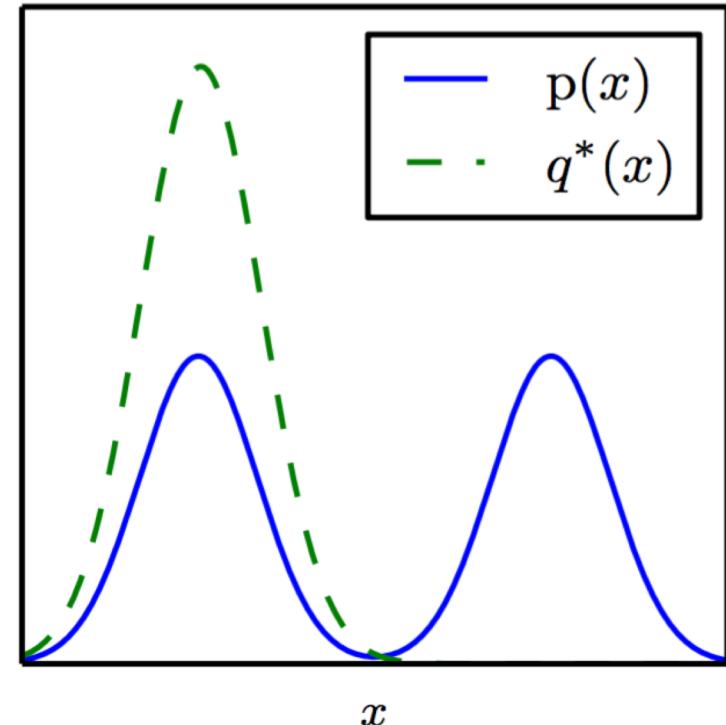
Probability Density



Maximum likelihood

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(q\|p)$$

Probability Density



Reverse KL