

Generative model

Generative model

Given training data, generate new samples from same distribution



Training data $\sim p_{\text{data}}(x)$



Generated samples $\sim p_{\text{model}}(x)$

Want to learn $p_{\text{model}}(x)$ similar to $p_{\text{data}}(x)$

Addresses density estimation, a core problem in unsupervised learning

Several flavors:

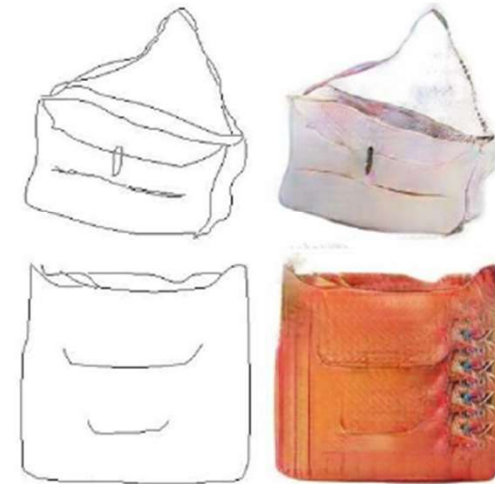
- Explicit density estimation: explicitly define and solve for $p_{\text{model}}(x)$
- Implicit density estimation: learn model that can sample from $p_{\text{model}}(x)$ w/o explicitly defining it

Explicit: distribution assumption (VAE)

Implicit: no distribution assumption (GAN)

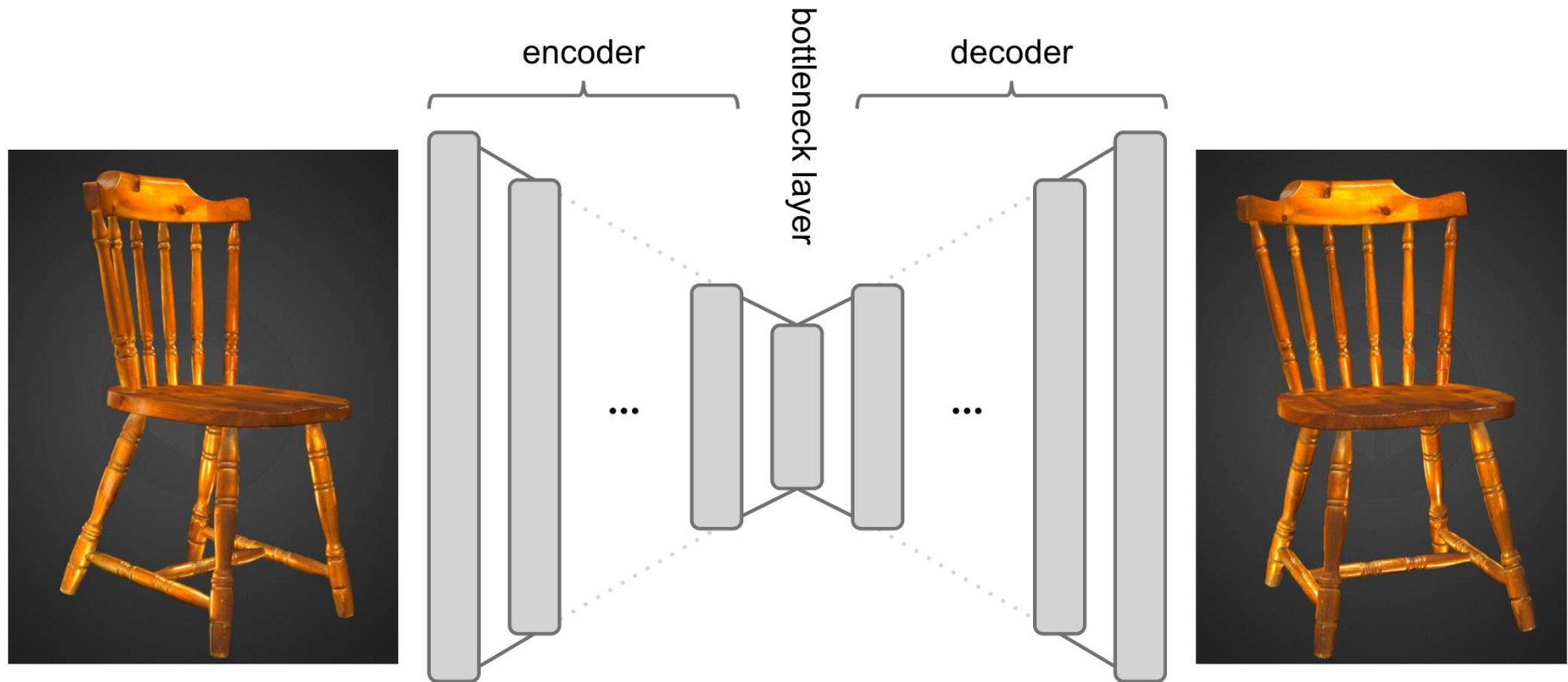
Generative model

- Realistic samples for artwork, super-resolution, colorization, etc.



- Generative models of time-series data can be used for simulation and planning (reinforcement learning applications!)
- Training generative models can also enable inference of latent representations that can be useful as general features

Autoencoder



- Input → encoder (learning grammar)
- latent space (contains essential information of input data)
- Decoder (learning translation) → reconstruct the original data
- Compare between input and output → loss function

Autoencoder

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

$\mathbf{z}$ usually smaller than $\mathbf{x}$
(dimensionality reduction)

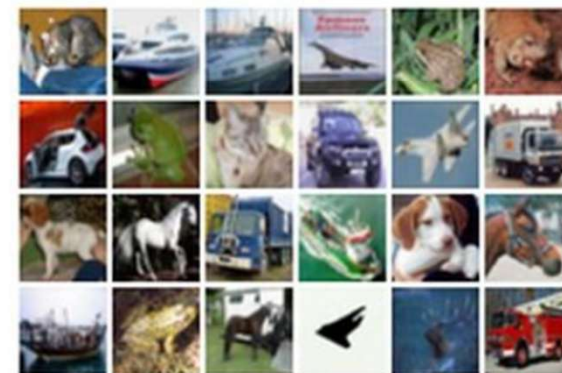
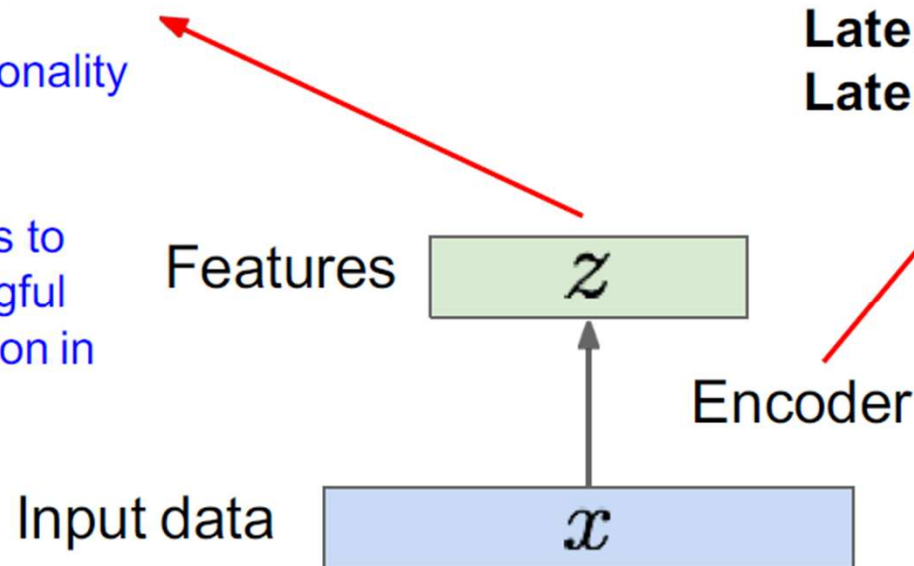
Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data

Originally: Linear + nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN

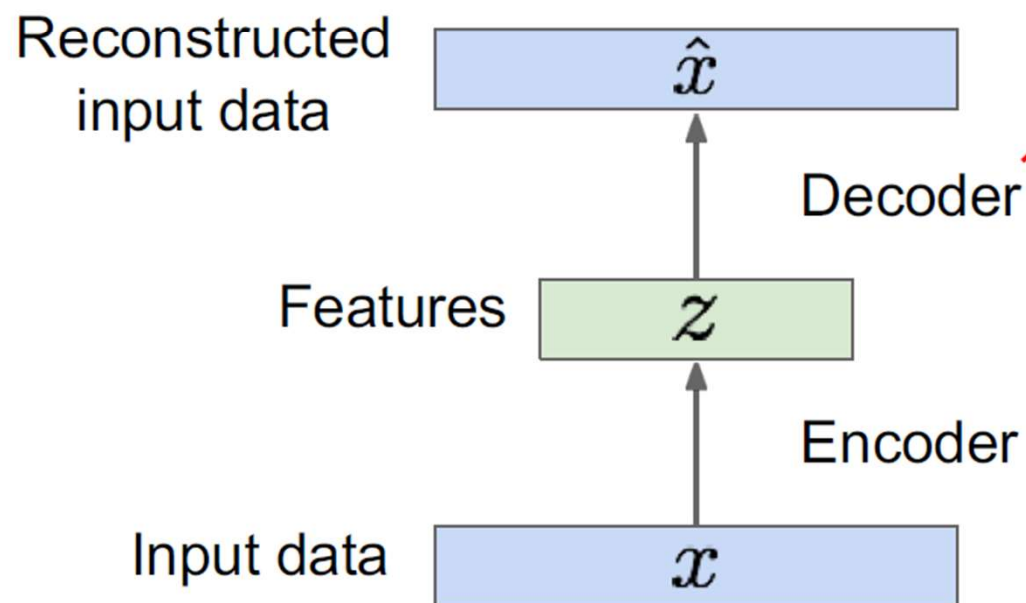


Autoencoder

How to learn this feature representation?

Train such that features can be used to reconstruct original data

“Autoencoding” - encoding itself



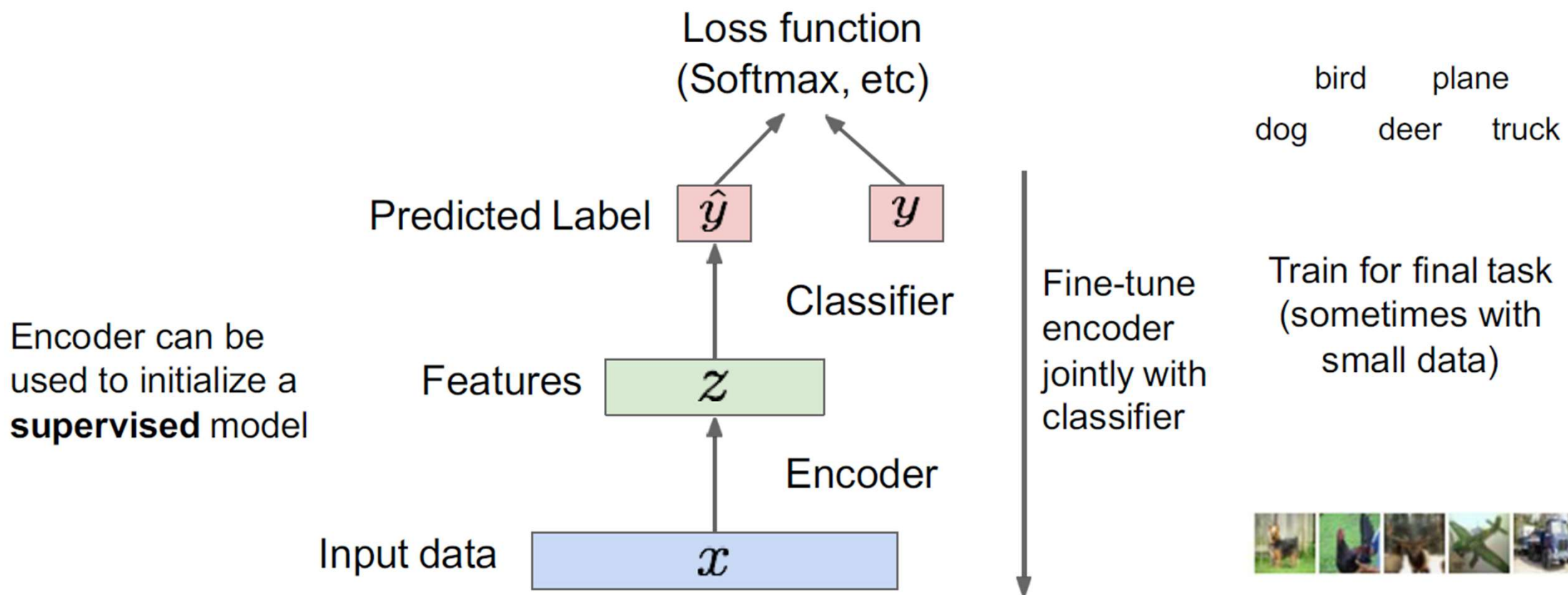
Originally: Linear + nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN (upconv)



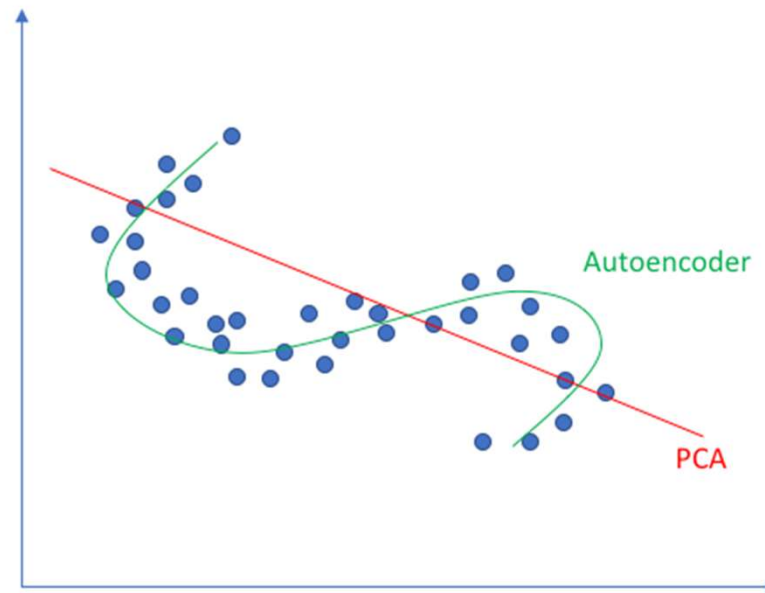
Autoencoder



-Classifier or regression model from “trained” encoder

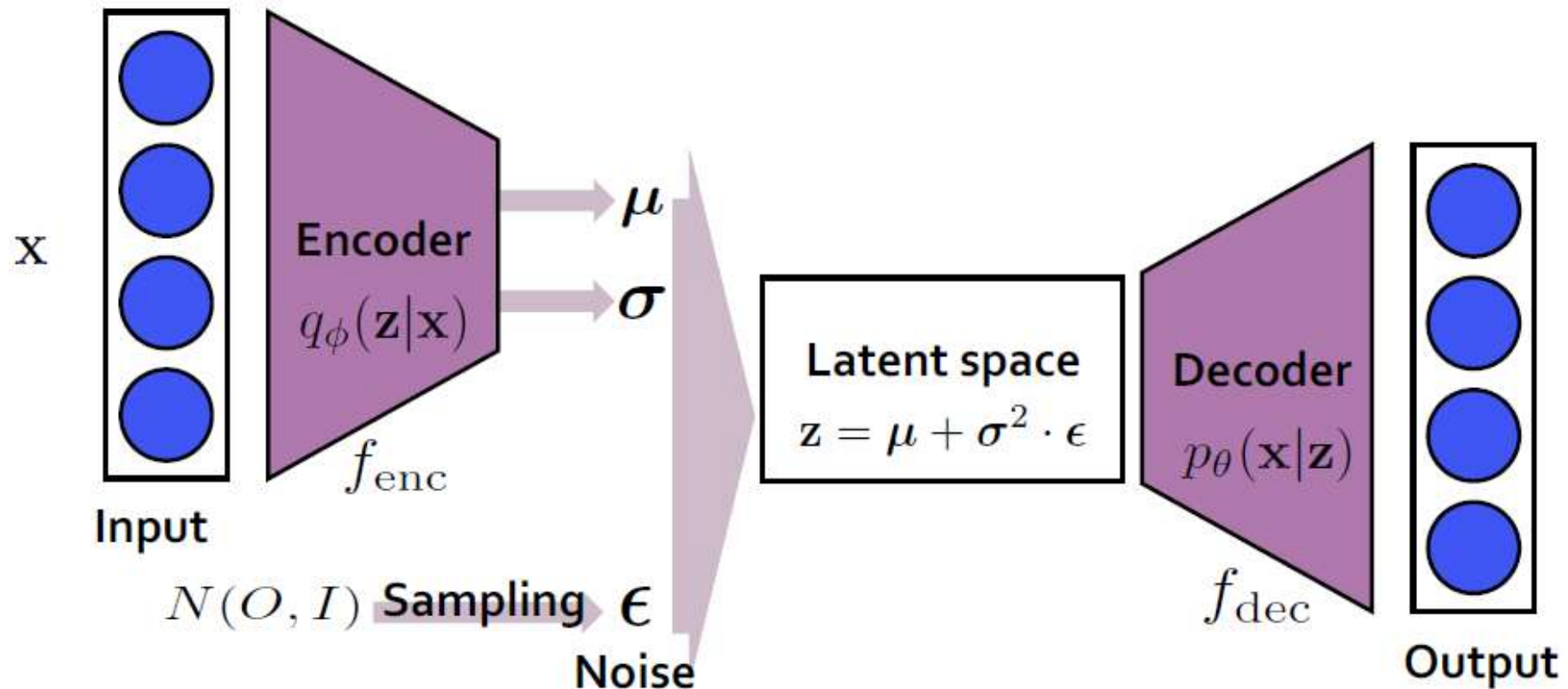
Autoencoder

Linear vs nonlinear dimensionality reduction



-Overfitting problem

Variational autoencoder (VAE)

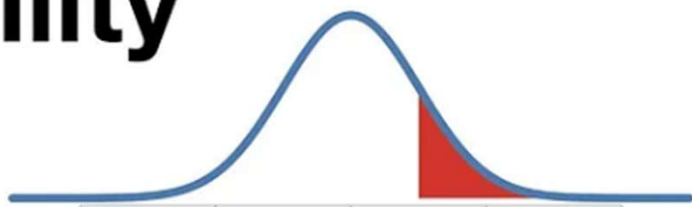


- After encoding $\rightarrow$ sampling from a certain distribution in the latent space
- $\rightarrow$ Decoder $\rightarrow$ reconstruct the data

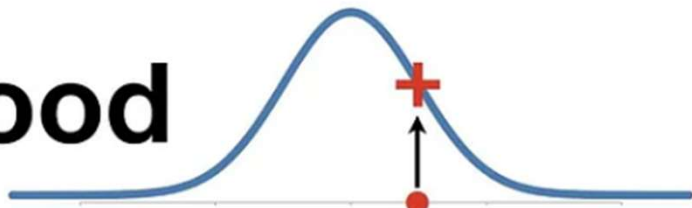
Variational autoencoder (VAE)

Probability

Vs



Likelihood



-Bayes' theorem

LIKELIHOOD

The probability of "B" being True, given "A" is True

PRIOR

The probability "A" being True. This is the knowledge.

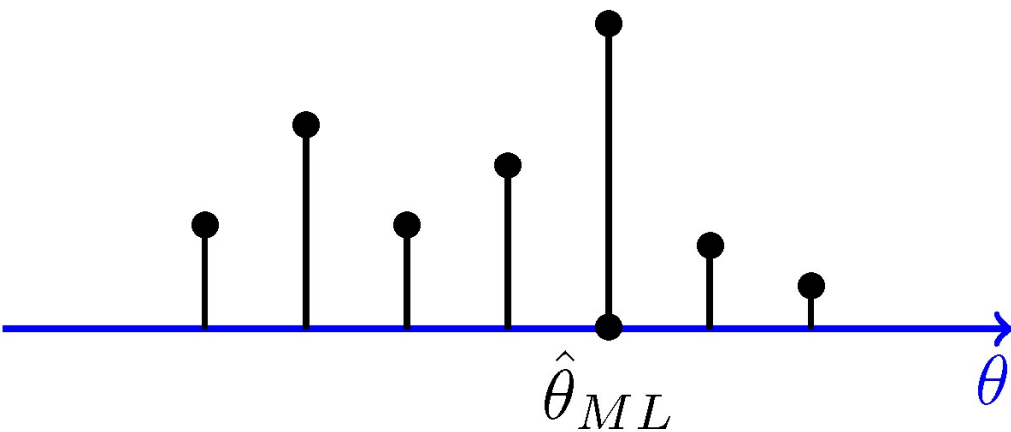
$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

POSTERIOR
The probability of "A" being True, given "B" is True

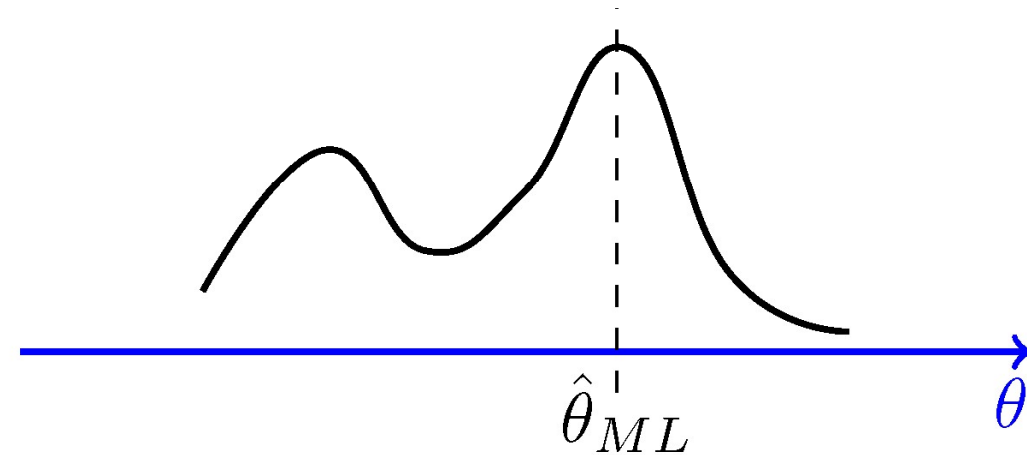
MARGINALIZATION
The probability "B" being True.

Variational autoencoder (VAE)

$$L(x_1, x_2, \dots, x_n; \theta)$$



$$L(x_1, x_2, \dots, x_n; \theta)$$



- MLE (Maximum likelihood estimation)
 - find a variable that makes the likelihood maximum
 - Most likely to happen!

Variational autoencoder (VAE)

Data likelihood: $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$

Posterior density also intractable: $p_{\theta}(z|x) = p_{\theta}(x|z)p_{\theta}(z)/p_{\theta}(x)$

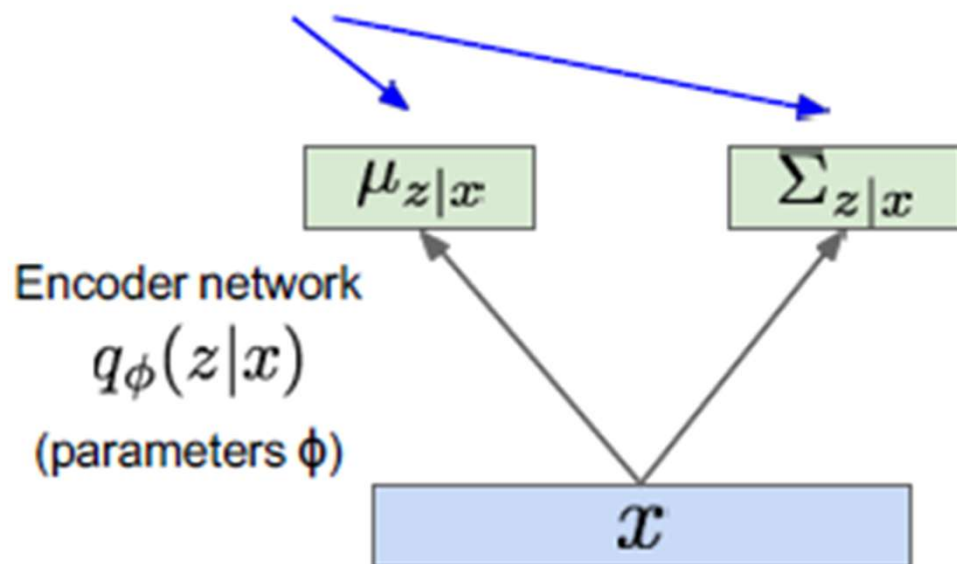
Solution: In addition to decoder network modeling $p_{\theta}(x|z)$, define additional encoder network $q_{\phi}(z|x)$ that approximates $p_{\theta}(z|x)$

Will see that this allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize

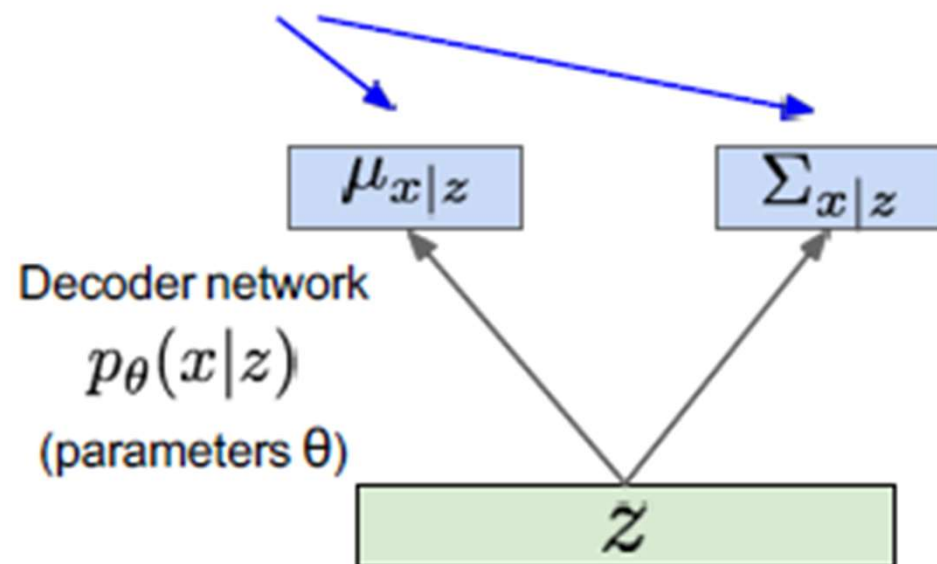
Variational autoencoder (VAE)

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic

Mean and (diagonal) covariance of $\mathbf{z} | \mathbf{x}$



Mean and (diagonal) covariance of $\mathbf{x} | \mathbf{z}$

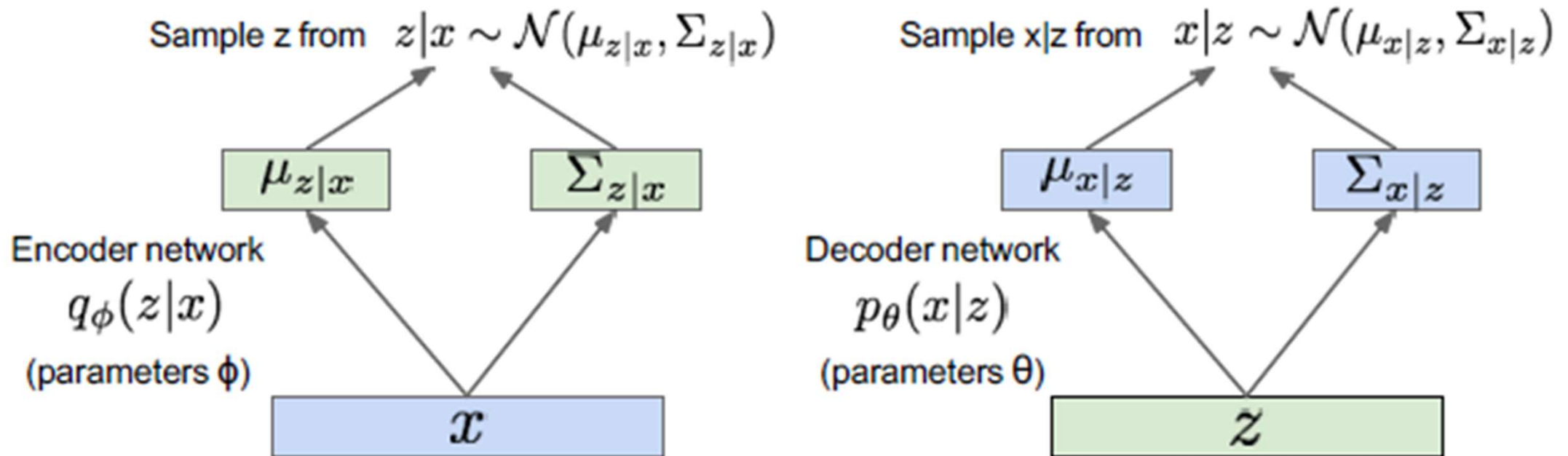


- $P(\mathbf{z})$: Gaussian distribution

- $P(\mathbf{x}|\mathbf{z})$: don't know the distribution $\rightarrow$ sampling from $\mathbf{Z} \rightarrow$ decoder network

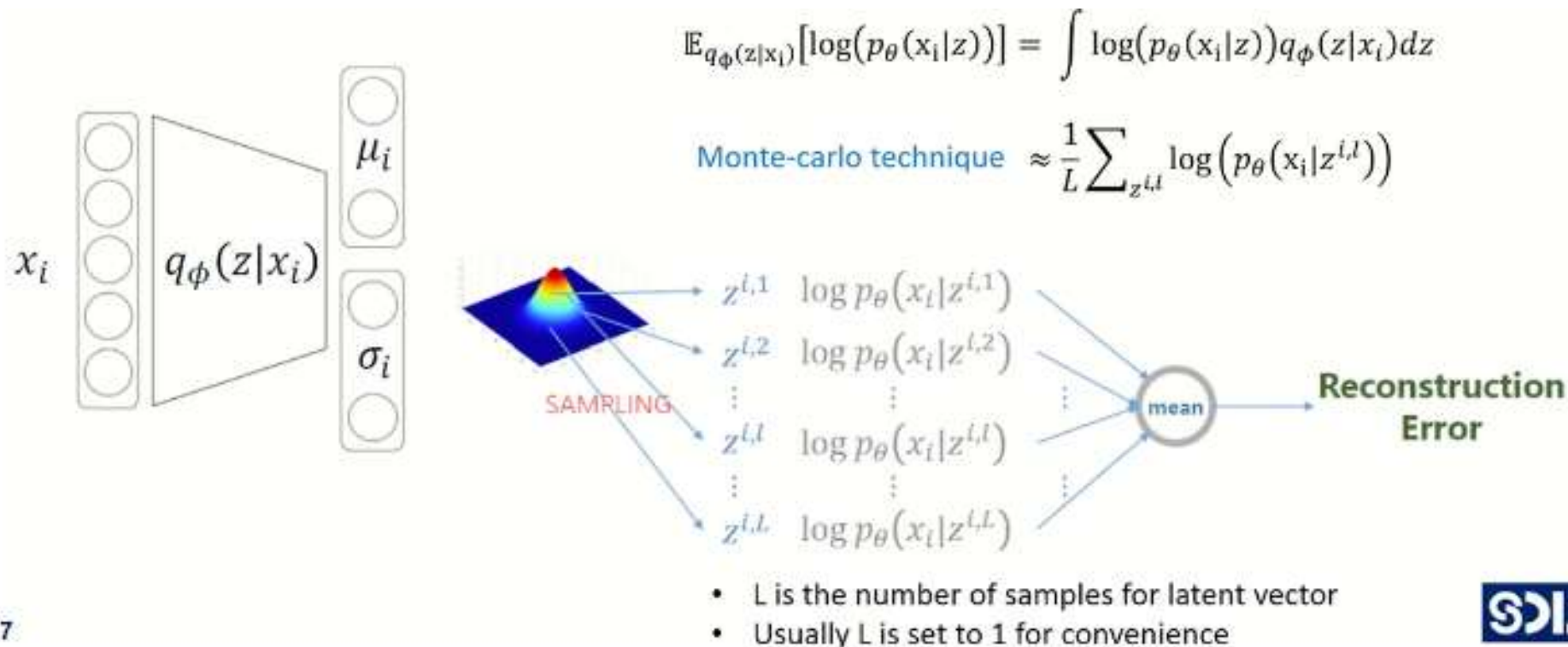
Variational autoencoder (VAE)

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



- Encoder network: approximate Z from x (Mean and Cov)
- Sampling from Gaussian distribution (z)
- Decoder network: using sampled data (this is an input now) $\rightarrow$ approximate x' (Mean and Cov)

Monte Carlo simulation



-Impossible to integrate from the distribution (z) → Monte Carlo simulation for the sampling

Variational autoencoder (VAE)

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

↑
Decoder network gives $p_{\theta}(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

↑
This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

↑
 $p_{\theta}(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always ≥ 0 .

-MLE (Maximum likelihood estimation) $\rightarrow$ find theta $\rightarrow$ maximize log P
(P: real, Q: approximated)

Variational autoencoder (VAE)

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

Decoder network gives $p_{\theta}(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_{\theta}(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always ≥ 0 .

- Reconstruction error: compare between predicted x' and real x
- x generated from z

Variational autoencoder (VAE)

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

↑
Decoder network gives $p_{\theta}(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

↑
This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

↑
 $p_{\theta}(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always ≥ 0 .

- D(KL): KL-divergence: measure the difference between two PDF
- Distribution from encoder: $q(z|x)$ vs noise sampling from Gaussian: $z \sim p(z)$

Variational autoencoder (VAE)

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

↑
Decoder network gives $p_{\theta}(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

↑
This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

↑
 $p_{\theta}(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always ≥ 0 .

Variational autoencoder (VAE)

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\&= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\&= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}\end{aligned}$$

Tractable lower bound which we can take
gradient of and optimize! ($p_{\theta}(x|z)$ differentiable,
KL term differentiable)

Variational autoencoder (VAE)

$$\begin{aligned}\log p_{\theta}(\mathbf{x}) &= \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}) d\mathbf{z} \\&= \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\&= \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x})} \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\&= \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} - KL(q_{\phi}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + KL(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\theta}(\mathbf{z}|\mathbf{x})) \\&= \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} - KL(q_{\phi}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + KL(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\theta}(\mathbf{z}|\mathbf{x})) \\&\geq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z} - KL(q_{\phi}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \longleftarrow \text{Evidence lower bound (ELBO)}\end{aligned}$$

Reconstruction error Regularization error

Variational autoencoder (VAE)

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\begin{aligned}
 \log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\
 \text{Reconstruct the input data} &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z) q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)}) q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\
 &= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\
 &= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right]}_{\mathcal{L}(x^{(i)}, \theta, \phi)} - \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\geq 0} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}
 \end{aligned}$$

Make approximate posterior distribution close to prior

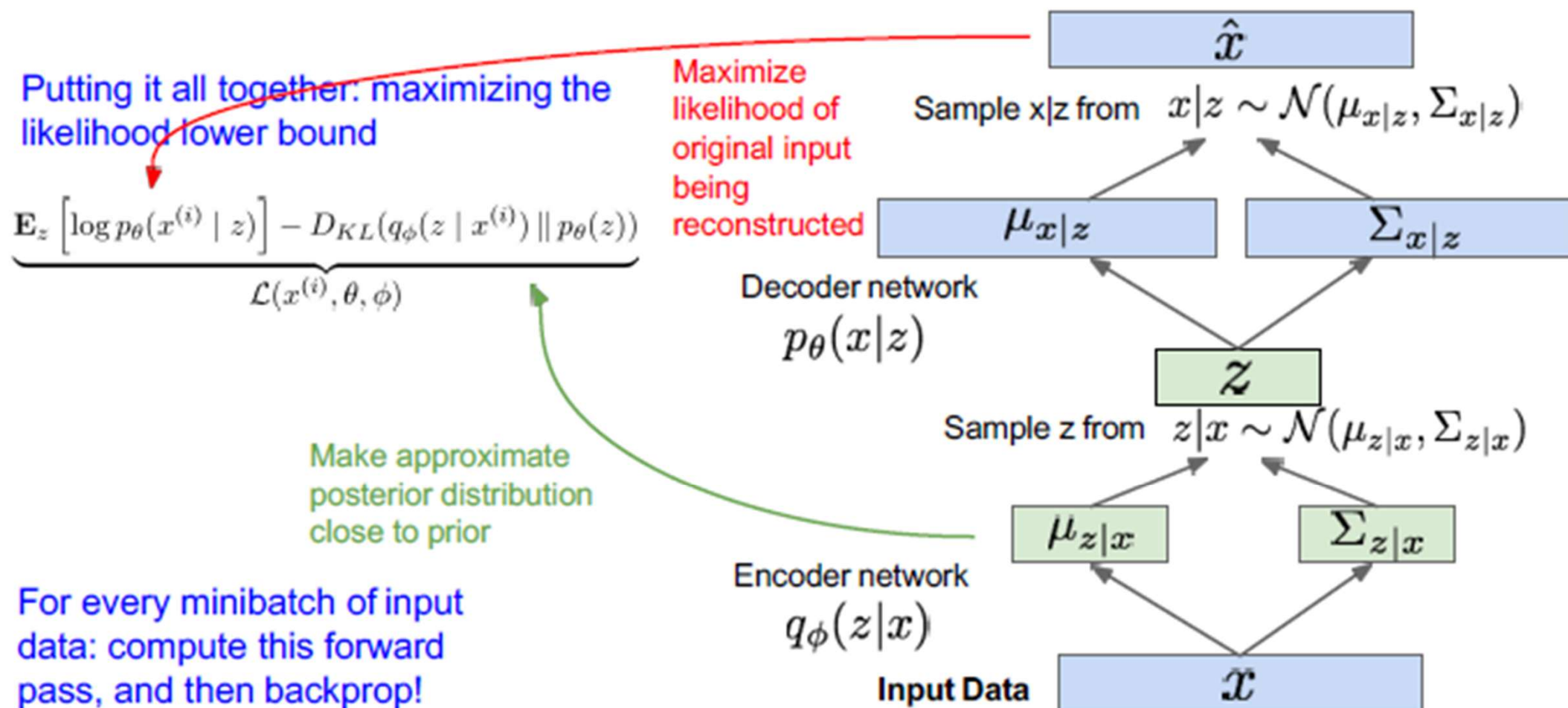
$$\log p_{\theta}(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound ("ELBO")

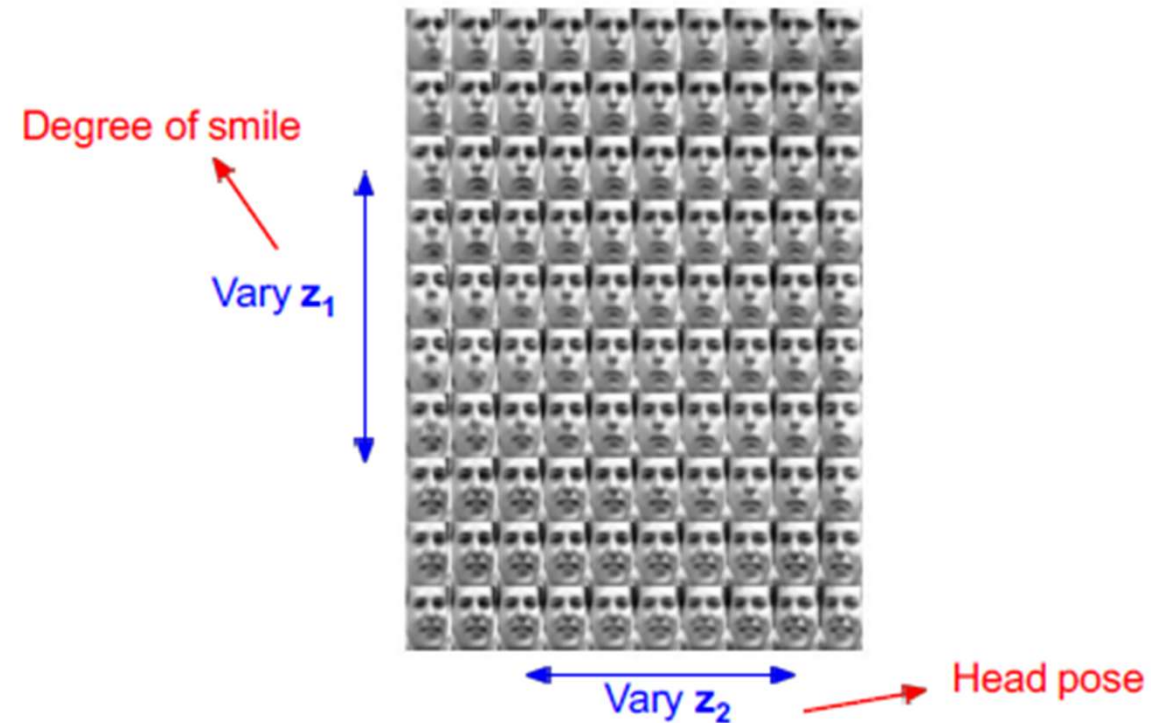
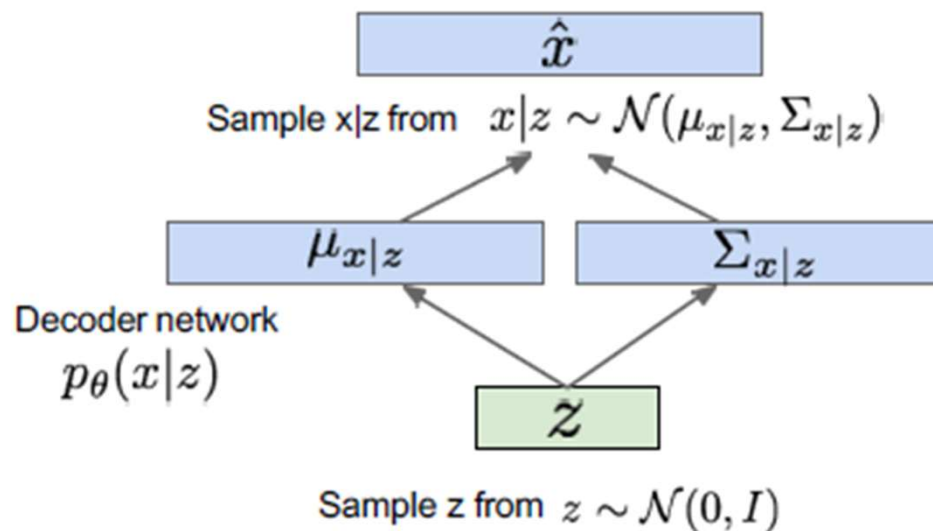
$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^N \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational autoencoder (VAE)



Variational autoencoder (VAE)



- Latent space contains essential features of input data
- Each dimension $\rightarrow$ diagonal to each other (latent variables)

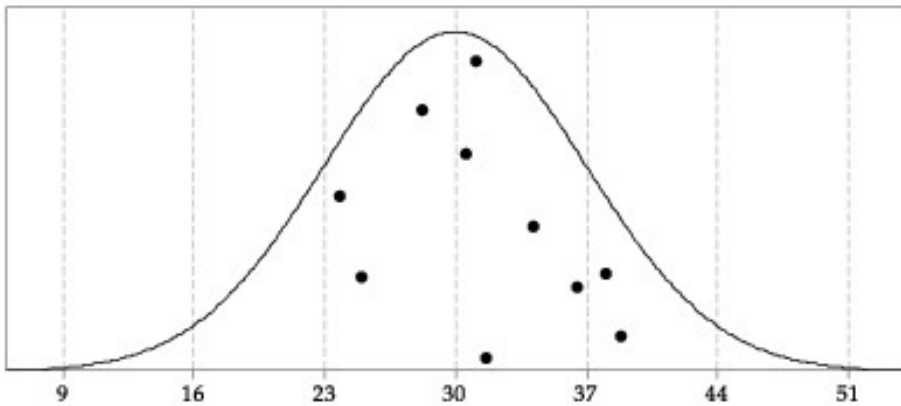
Variational autoencoder (VAE)

- Fast
- But, z does not always follow the gaussian distribution
→ x' is blurry

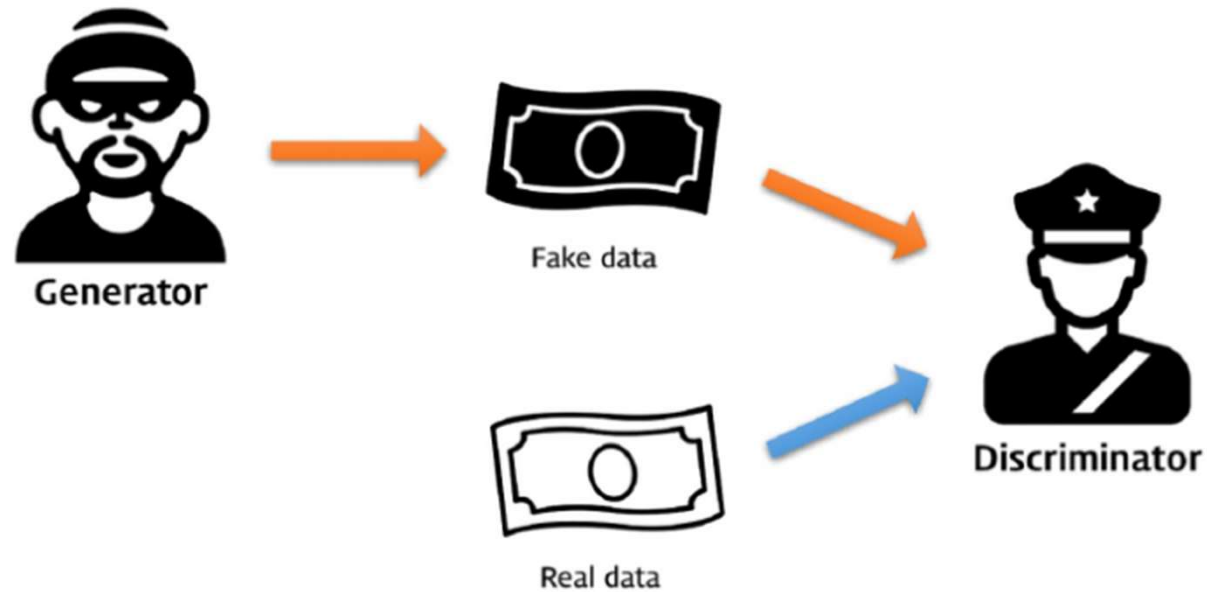


Generative Adversarial Network (GAN)

Implicit approach: do not assume a certain distribution
Just random sampling from gaussian distribution



Generative Adversarial Network (GAN)



Generator: trying to make a fake (simulated) data

Discriminator: trying to distinguish between fake and real data

→ Their ability will be converged (50:50)

Generative Adversarial Network (GAN)

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

Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

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

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\substack{\text{Discriminator output} \\ \text{for real data } x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\substack{\text{Discriminator output for} \\ \text{generated fake data } G(z)}}) \right]$$

- Discriminator (θ_d) wants to **maximize objective** such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real)

Generative Adversarial Network (GAN)

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

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

Generative Adversarial Network (GAN)

Putting it together: GAN training algorithm

K sampling

for number of training iterations do epoch

for k steps do

• Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$. P(z): random sampling from gaussian

• Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$. Read data sampling

• Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

Train Discriminator first

end for

• Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.

• Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$
 train generator next

end for

Generative Adversarial Network (GAN)

Loss function: BCR

Binary cross entropy (only read or fake)

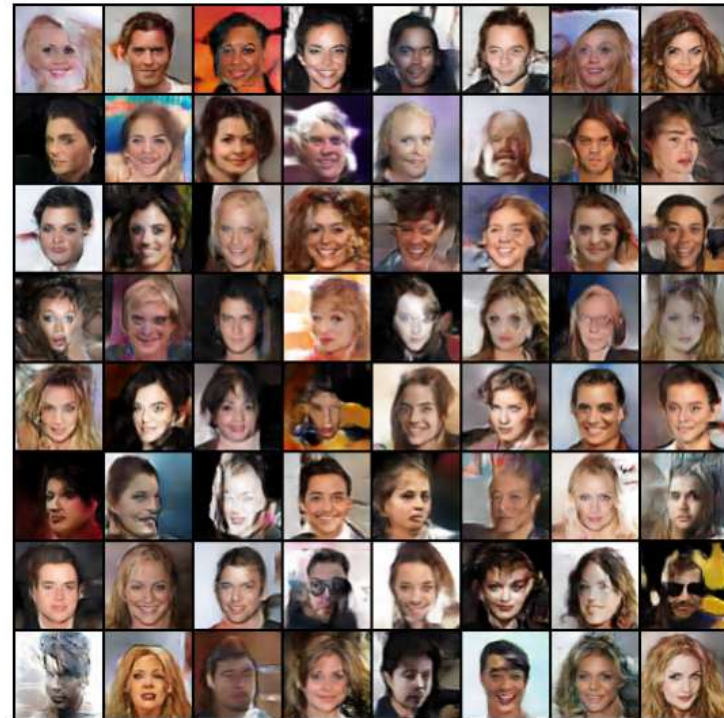
$$\text{BCE} = -\frac{1}{n} \sum_{i=1}^n (y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

Generative Adversarial Network (GAN)

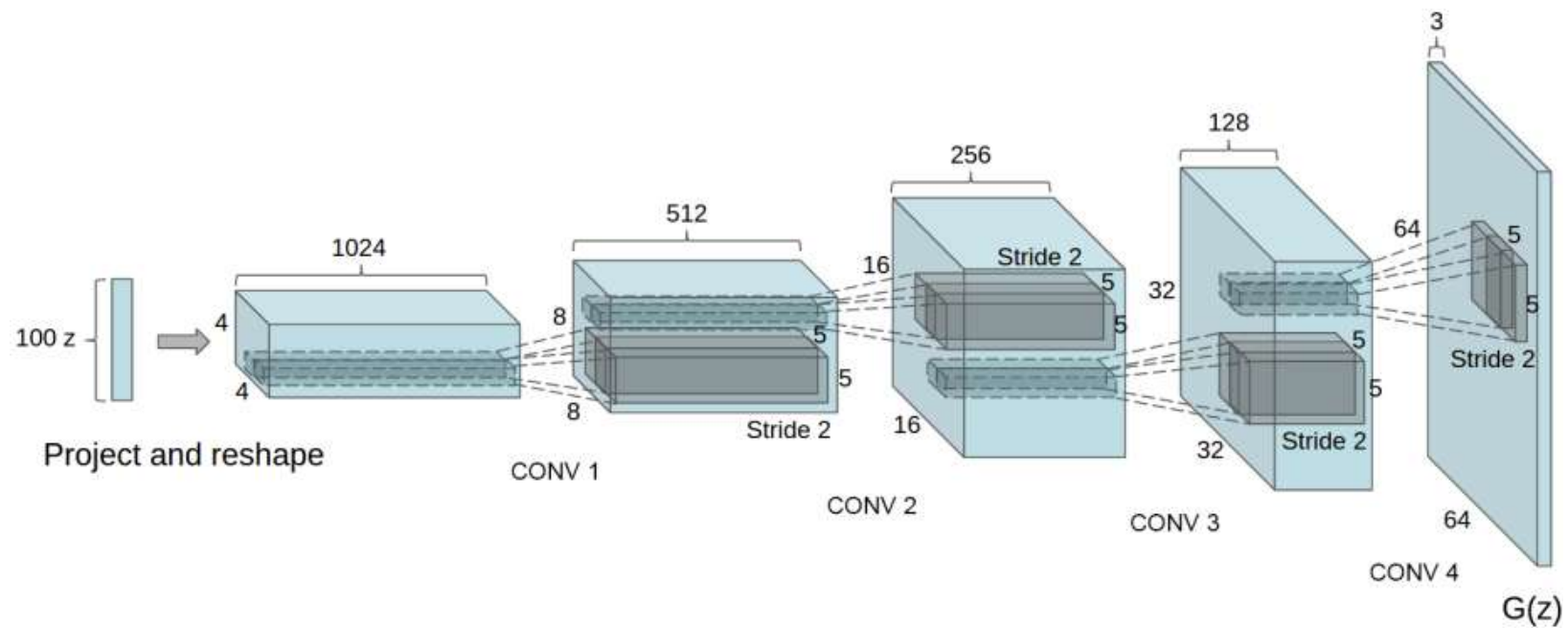
Real Images



Fake Images



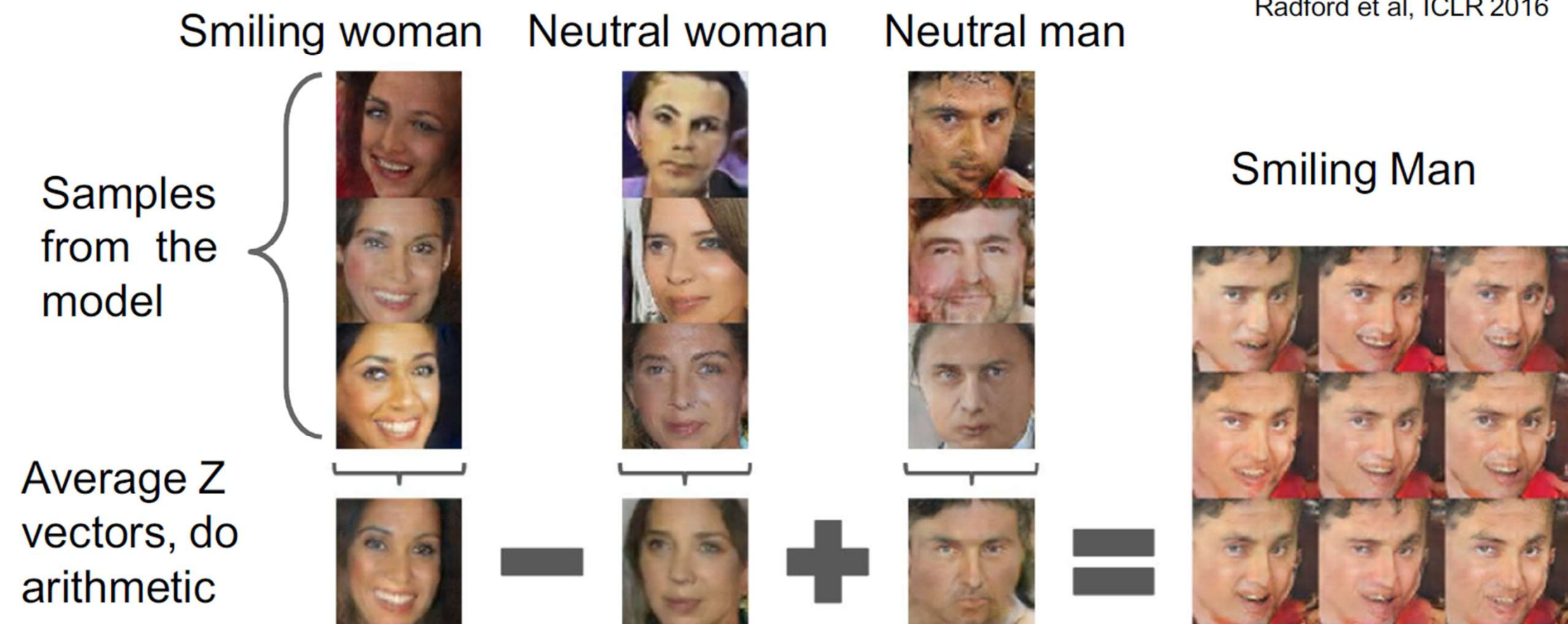
Deep Convolutional Generative Adversarial Networks (DCGAN)



DCGAN

Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016



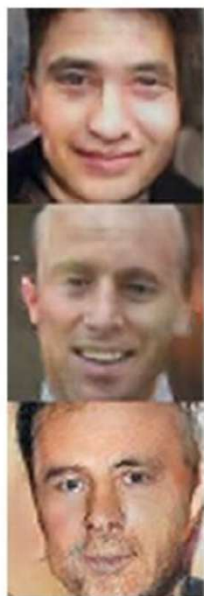
DCGAN

Radford et al,
ICLR 2016

Glasses man



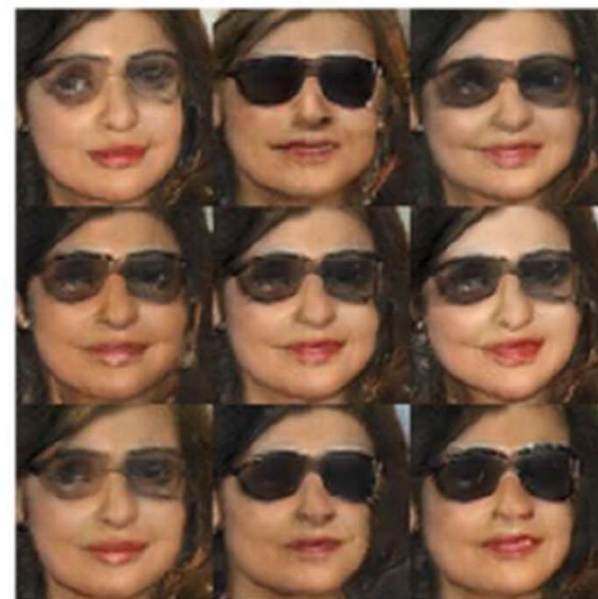
No glasses man



No glasses woman



Woman with glasses



DCGAN

2017: Year of the GAN

Better training and generation



(a) Church outdoor.

(b) Dining room.



(c) Kitchen.

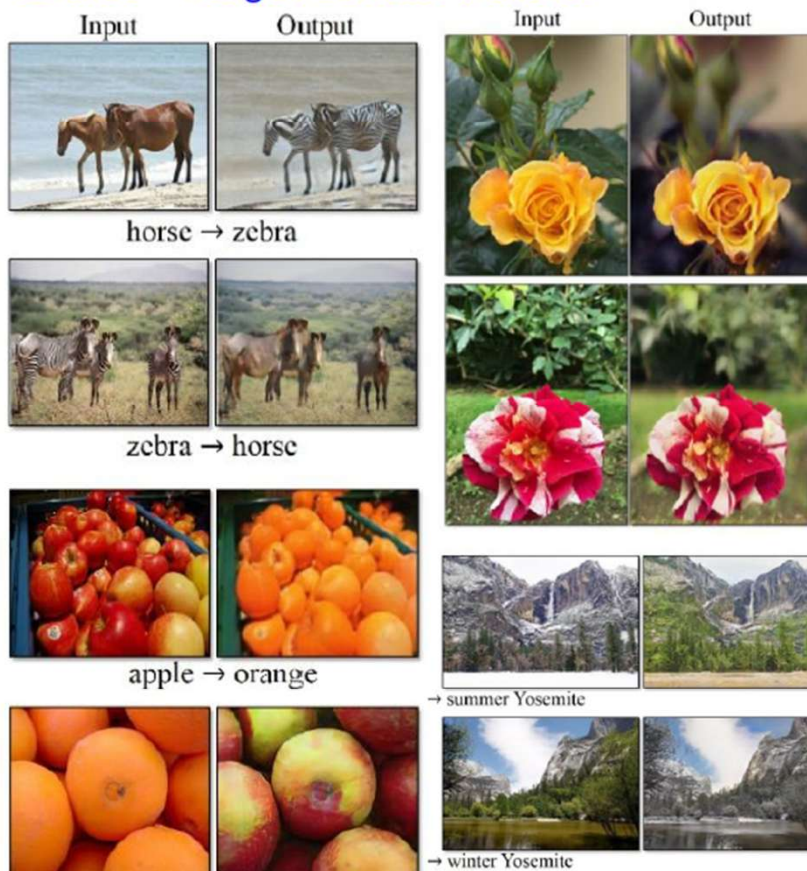
(d) Conference room.

LSGAN. Mao et al. 2017.



BEGAN. Bertholet et al. 2017.

Source->Target domain transfer



CycleGAN. Zhu et al. 2017.

Text -> Image Synthesis

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.



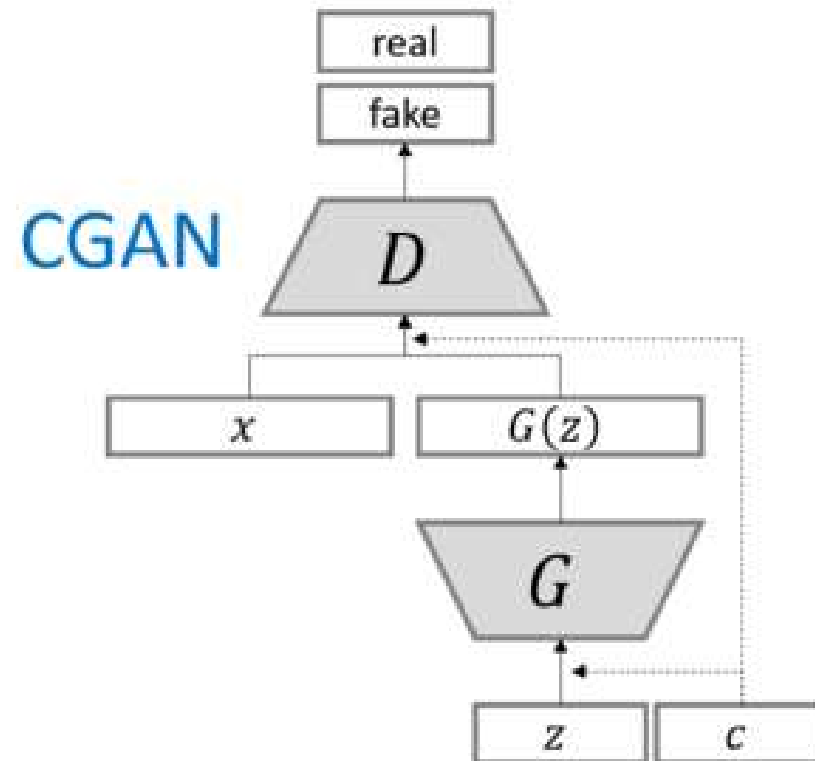
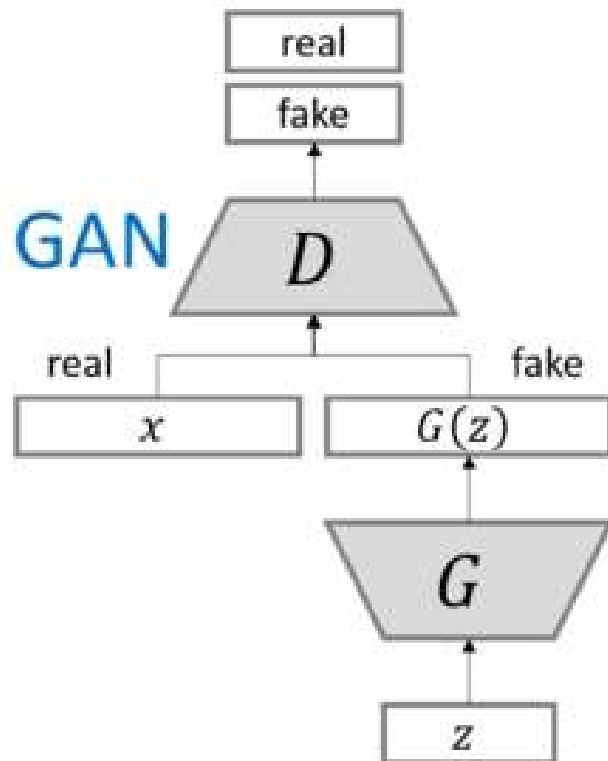
Reed et al. 2017.

Many GAN applications



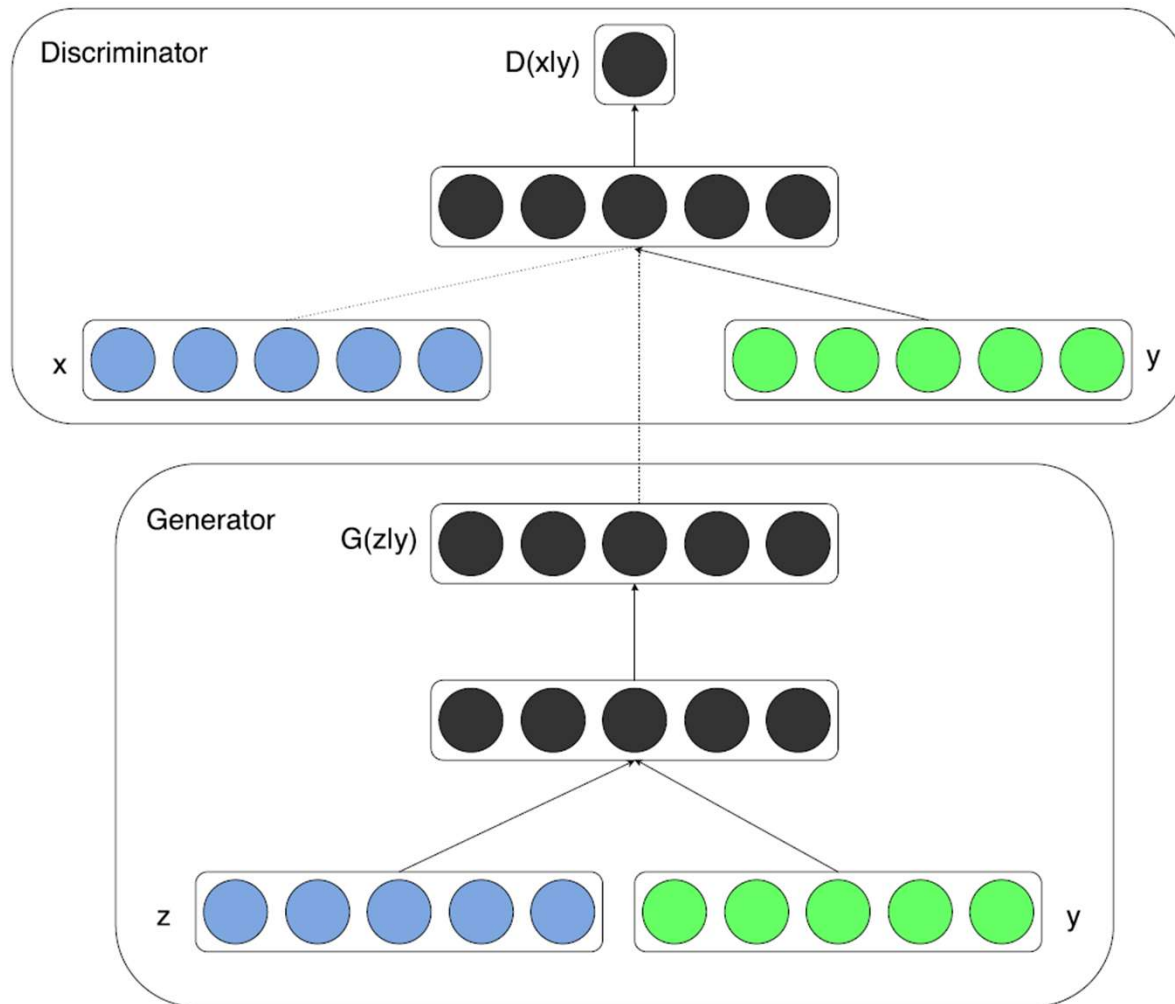
Pix2pix. Isola 2017. Many examples at <https://phillipi.github.io/pix2pix/>

Conditional GAN



- Given condition (c) $\rightarrow$ GAN
- $\rightarrow$ GAN for each condition

Conditional GAN



- y: condition vector
- Concatenate with z (latent space)
- Concatenate with discriminator input (x)
- $D(x|y)$ will be generated for each condition (y)

Conditional GAN

y: condition vector

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

...

[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]



Cycle GAN

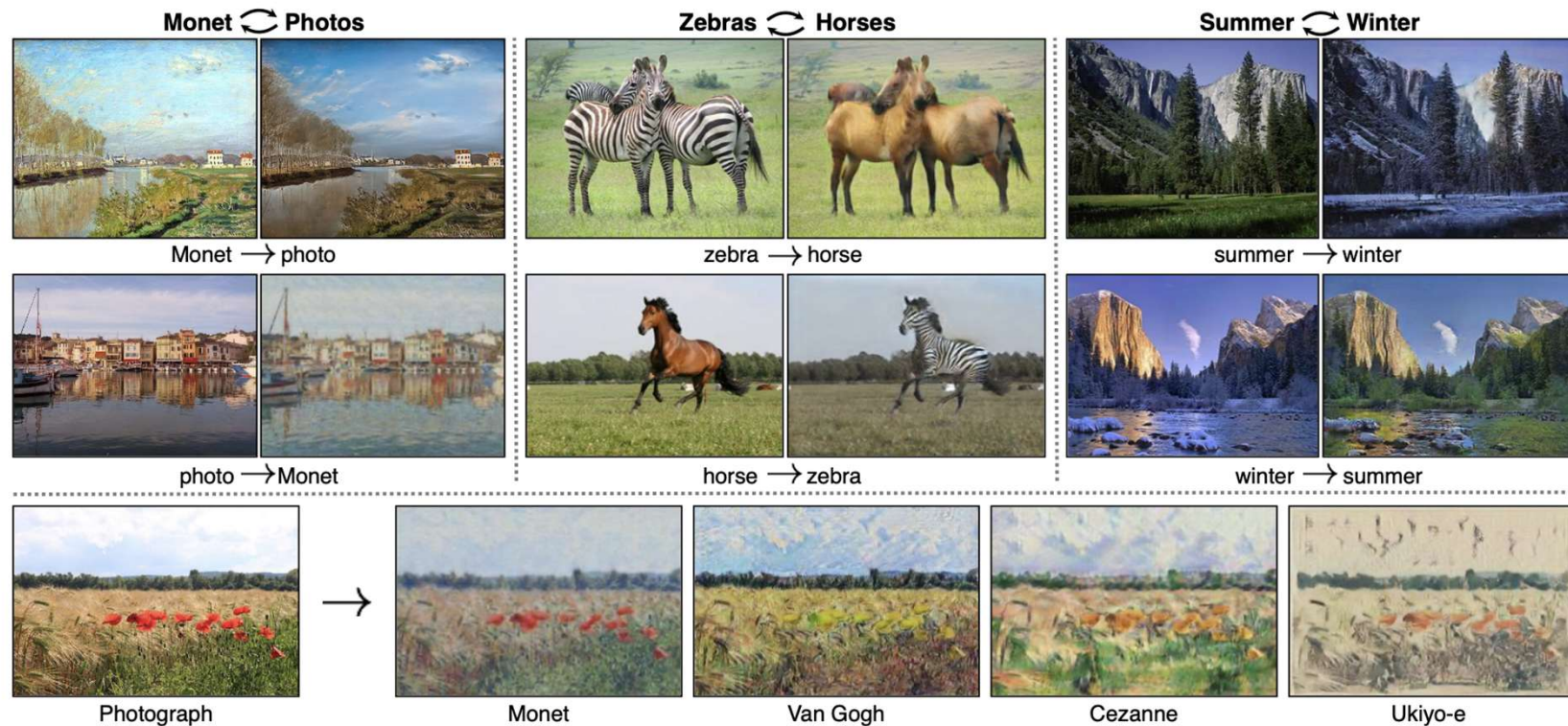
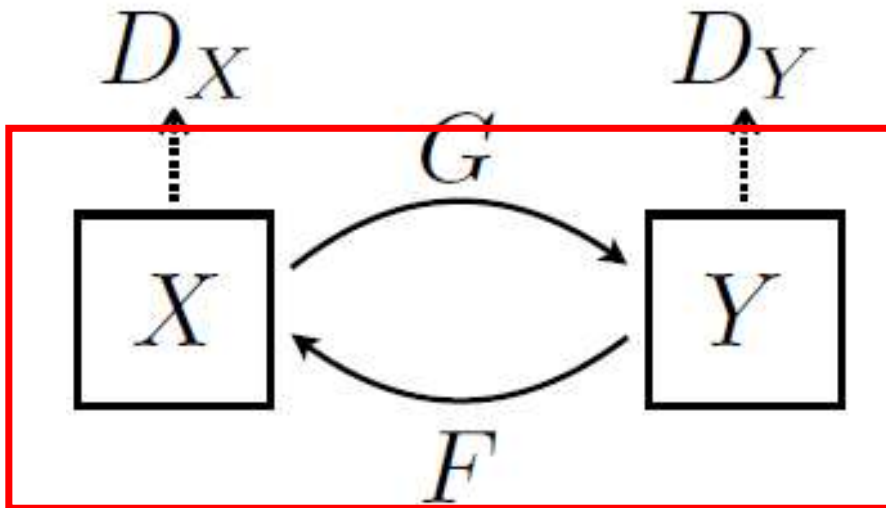


Figure 1: Given any two unordered image collections X and Y , our algorithm learns to automatically “translate” an image from one into the other and vice versa: (*left*) Monet paintings and landscape photos from Flickr; (*center*) zebras and horses from ImageNet; (*right*) summer and winter Yosemite photos from Flickr. Example application (*bottom*): using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

-Style transfer (ex: zebra → horse)

Cycle GAN



(a)

*Loss function

-G: $x \rightarrow y$

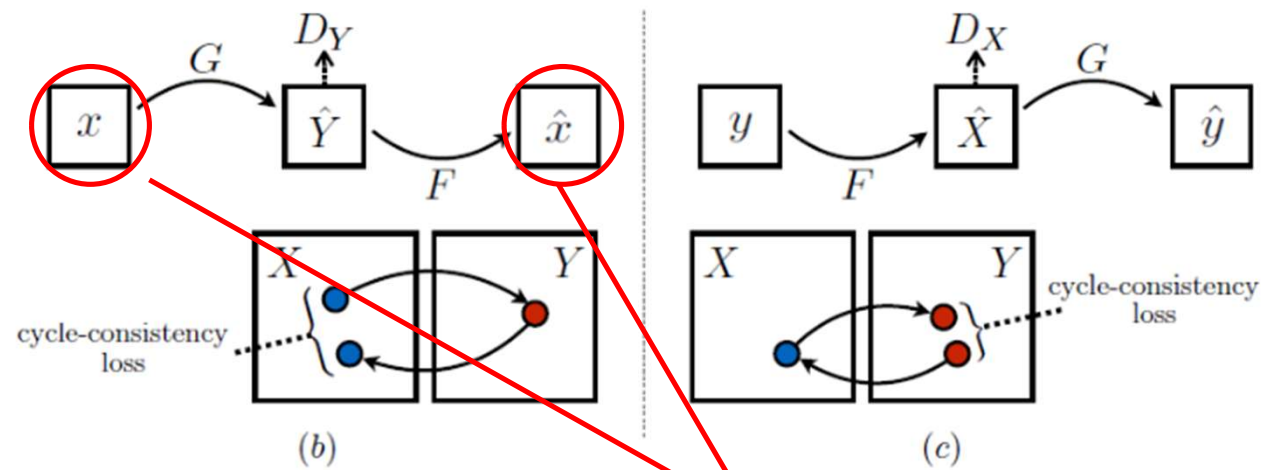
$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)} [\log D_Y(y)] \\ + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log(1 - D_Y(G(x)))]$$

*-F: $y \rightarrow x$

$$\min_F \max_{D_X} \mathcal{L}_{GAN}(F, D_X, X, Y)$$

Cycle GAN

Consistency loss



$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(F(y)) - y\|_1]$$

Cycle GAN

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = & \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ & + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ & + \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

Cycle GAN

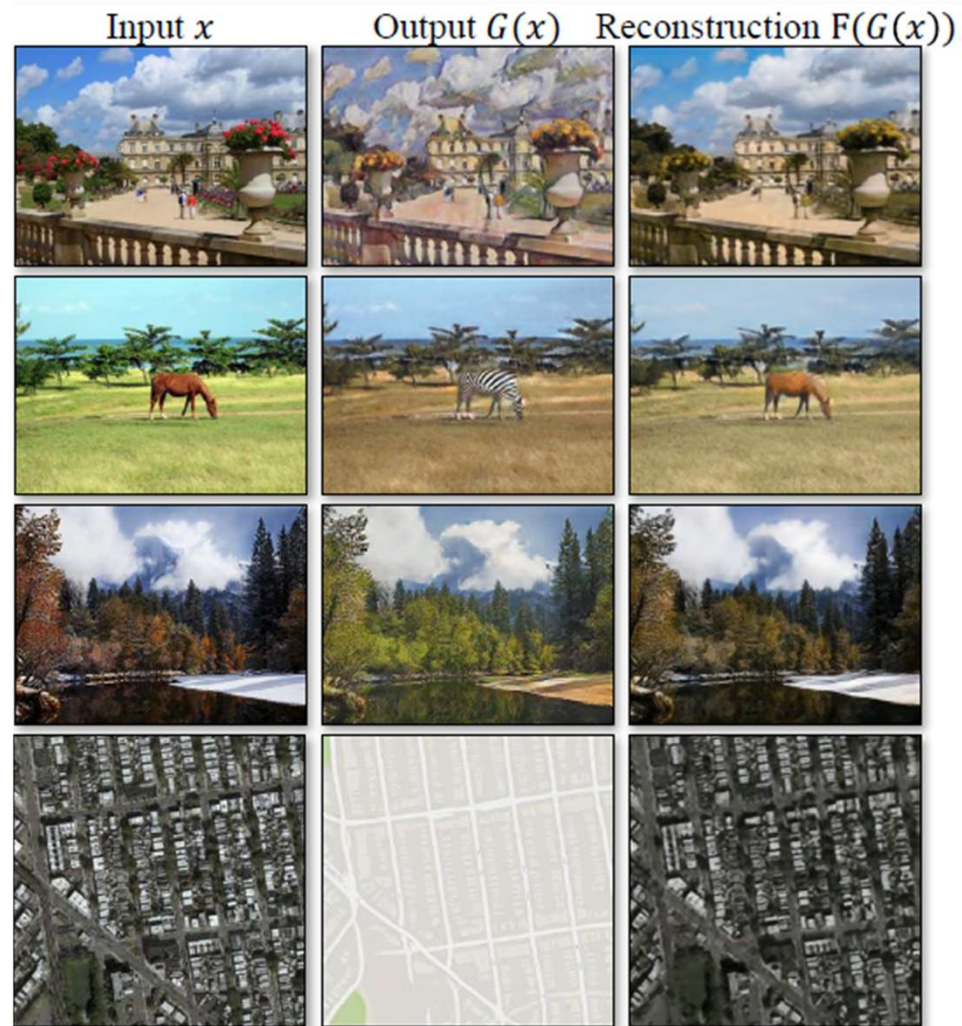
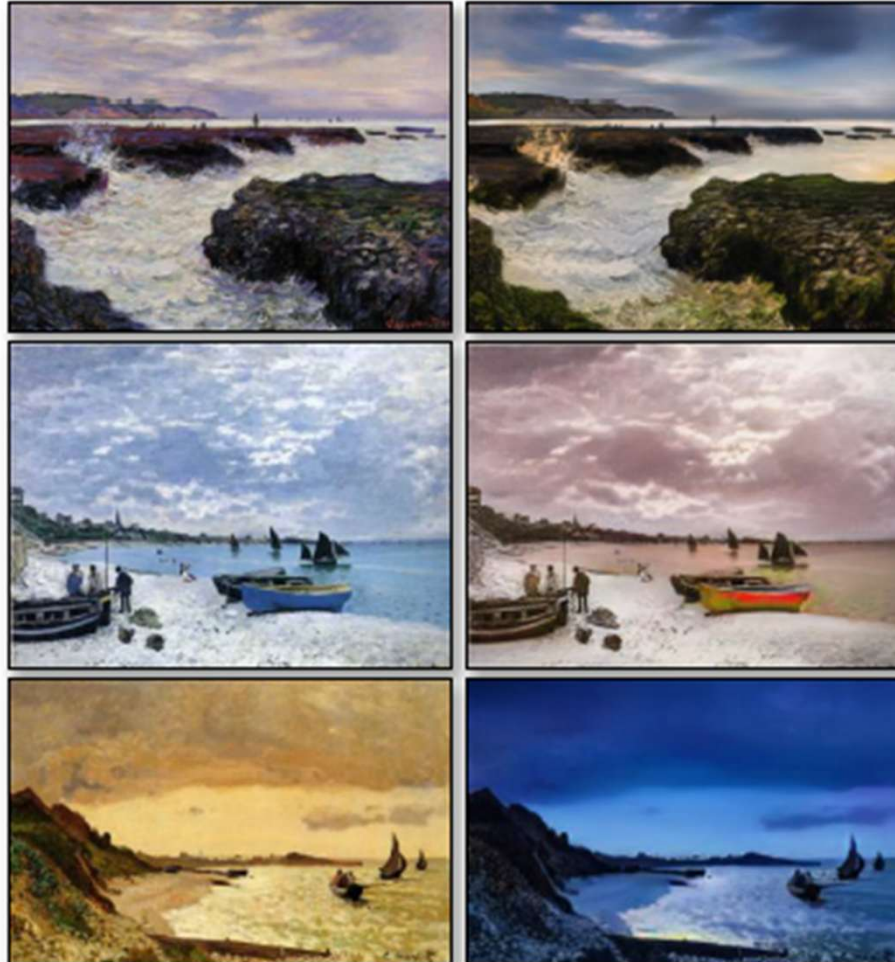


Figure 4: The input images x , output images $G(x)$ and the reconstructed images $F(G(x))$ from various experiments. From top to bottom: photo $\leftrightarrow$ Cezanne, horses $\leftrightarrow$ zebras, winter $\rightarrow$ summer Yosemite, aerial photos $\leftrightarrow$ Google maps.

Cycle GAN (generate different style)

Input

CycleGAN



Cycle GAN (limitation)

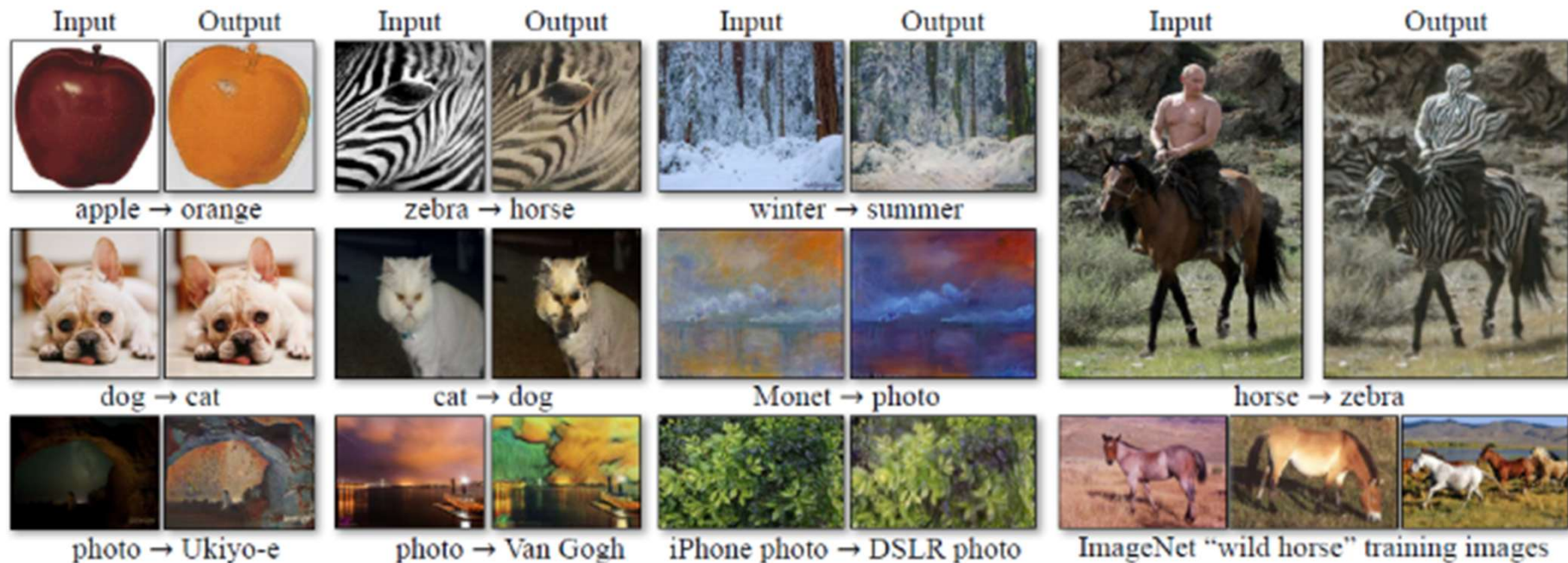


Figure 17: Typical failure cases of our method. Left: in the task of dog→cat transfiguration, CycleGAN can only make minimal changes to the input. Right: CycleGAN also fails in this horse → zebra example as our model has not seen images of horseback riding during training. Please see our [website](#) for more comprehensive results.

- cannot change the “original” shape (or data structure)
- look at the zebra

