Probabilistic Graphical Models & Probabilistic AI

Ben Lengerich

Lecture 18: VAEs, GANs

April 8, 2025

Reading: See course homepage

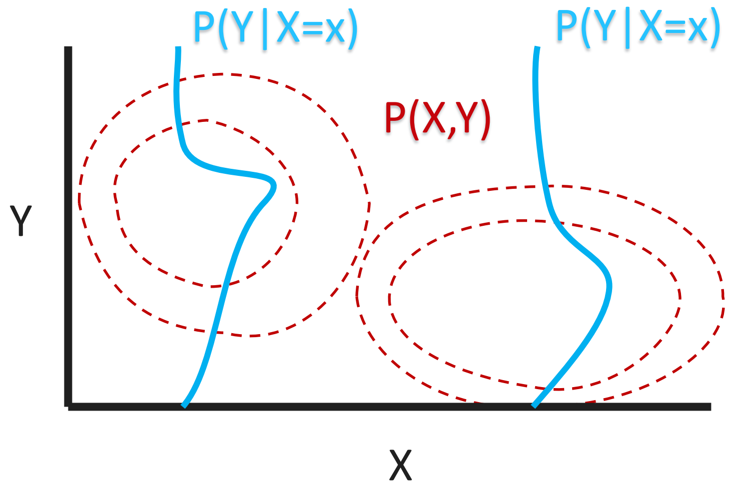
Today

- Deep Generative Models
  - VAEs
  - GANs
  - Diffusion Models

Deep Generative Models

Recall Generative and Discriminative Models

- **Generative**:
  - Models the joint distribution $\mathrm{P}(X, Y)$.
- **Discriminative**:
  - Models the conditional distribution $\mathrm{P}(Y|X)$.



Two paths to P(Y|X)

- **Discriminative**:

Observe X, Y → Learn P(Y|X)

- **Generative**:
  - Learn $\mathrm{P}(X|Y), \mathrm{P(Y)}$
  - Calculate $P(X) = \int_Y P(X,Y)dY$

Observe X, Y → $\mathrm{P(Y|X)} = \frac{\mathrm{P(X|Y)P(Y)}}{P(X)}$

Example Generative Model: Naïve Bayes

Learn $\mathrm{P}(X|Y), \mathrm{P}(\mathrm{Y})$

Observe X, Y

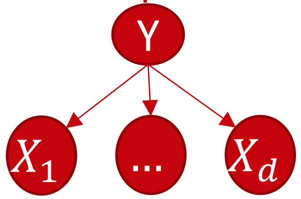
$$\mathrm{P(Y|X)} = \frac{\mathrm{P(X|Y)P(Y)}}{P(X)}$$

- Parameterize:
  - Assume $P(X|Y) = \prod_{j=1}^{d} P(X_j|Y)$,
  - $P(X_j|Y) = N(\mu_{jk}, \sigma_{jk}^2)$

$$P(Y = k) = \frac{\#\ of\ samples\ with\ Y{=}k}{Total\ samples}$$

Conditional independences of features X | Y

Frequency of labels



Example Generative Model: Naïve Bayes

Learn $\mathrm{P}(X|Y), \mathrm{P}(\mathrm{Y})$

Observe X, Y

$$\mathrm{P}(\mathrm{Y}|\mathrm{X}) = \frac{\mathrm{P}(\mathrm{X}|\mathrm{Y})\mathrm{P}(\mathrm{Y})}{P(X)}$$

- Parameterize:
  - Assume $P(X|Y) = \prod_{j=1}^{d} P\left(X_j \middle| Y\right)$,

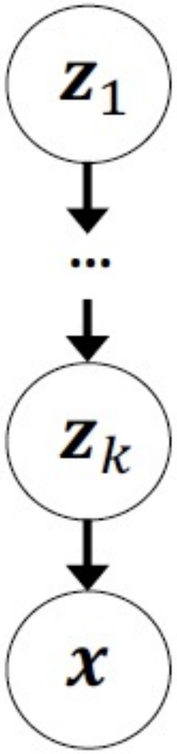
$$P(Y = k) = \frac{\#\ of\ samples\ with\ Y{=}k}{Total\ samples}$$

- Estimate:
  - $\hat{\mu}, \hat{\sigma} = \operatorname{argmax}_{\mu,\sigma} P(X|Y)$
- Calculate $P(Y = 1|X) = \frac{\prod_{j=1}^{d} P\left(X_j \middle| Y = 1\right) P(Y=1)}{P(X)}$

Deep Generative Models

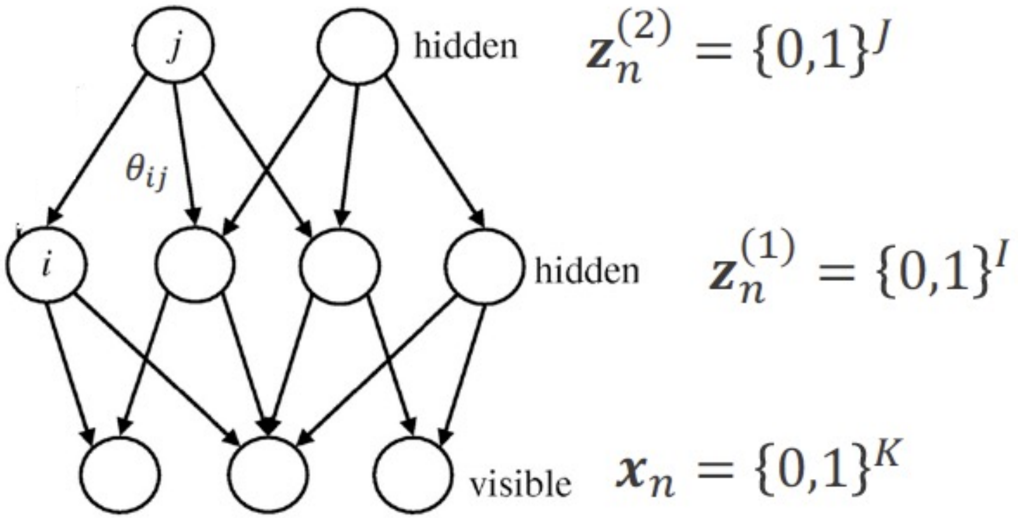
Deep Generative Models

- Define probabilistic distributions overs a set of variables
- "Deep" means multiple layers of hidden variables!



Early forms of deep generative models

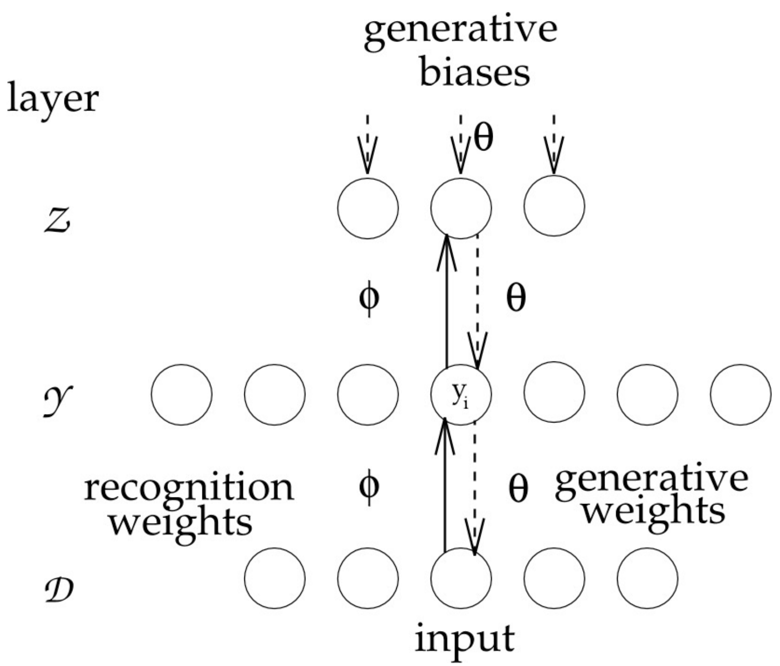
- Hierarchical Bayesian models
  - Sigmoid belief nets [Neal 1992]



$$p\left(x_{kn} = 1 \middle| \boldsymbol{\theta}_k, \mathbf{z}_n^{(1)}\right) = \sigma\left(\boldsymbol{\theta}_k^T \mathbf{z}_n^{(1)}\right)$$
$$p\left(z_{in}^{(1)} = 1 \middle| \boldsymbol{\theta}_i, \mathbf{z}_n^{(2)}\right) = \sigma\left(\boldsymbol{\theta}_i^T \mathbf{z}_n^{(2)}\right)$$

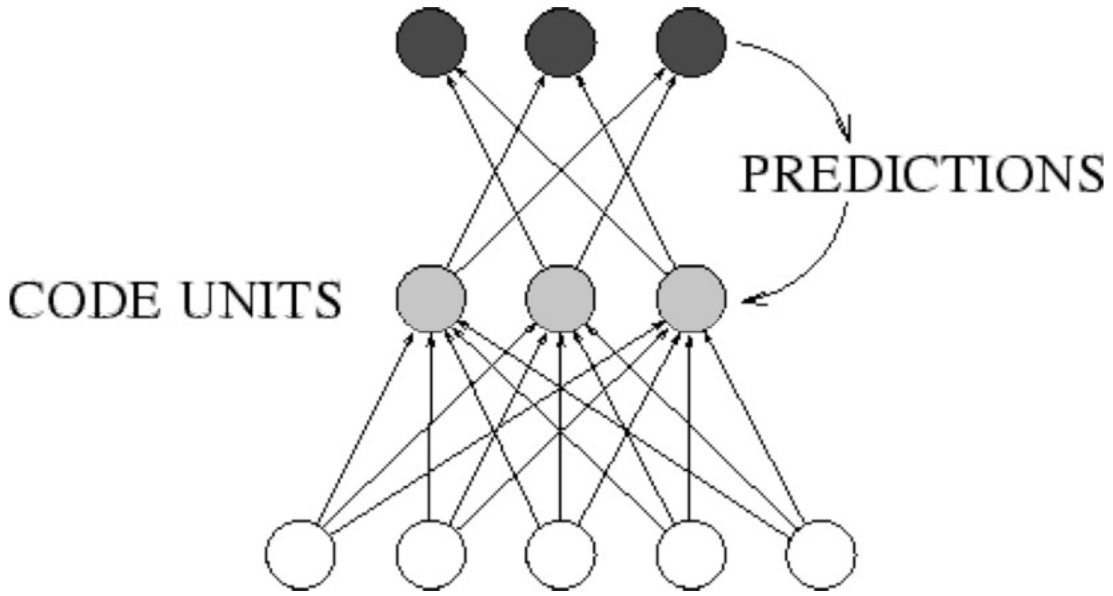
Early forms of deep generative models

- Hierarchical Bayesian models
  - Sigmoid belief nets [Neal 1992]
  - Neural network models
    - Helmoltz machines [Dayan et al., 1995]



Early forms of deep generative models

- Hierarchical Bayesian models
  - Sigmoid belief nets [Neal 1992]
  - Neural network models
    - Helmoltz machines [Dayan et al., 1995]
    - Predictability minimization [Schmidhuber 1995]



[Schmidhuber 1996]

Training DGMs

- Via an EM-style framework
  - Sampling / data augmentation

$$\boldsymbol{z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2\}$$
$$\boldsymbol{z}_1^{new} \sim p(\boldsymbol{z}_1 | \boldsymbol{z}_2, \boldsymbol{x})$$
$$\boldsymbol{z}_2^{new} \sim p(\boldsymbol{z}_2 | \boldsymbol{z}_1^{new}, \boldsymbol{x})$$

  - Variational inference

$$\log p(\boldsymbol{x}) \geq \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z})) := \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$$

$$\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$$

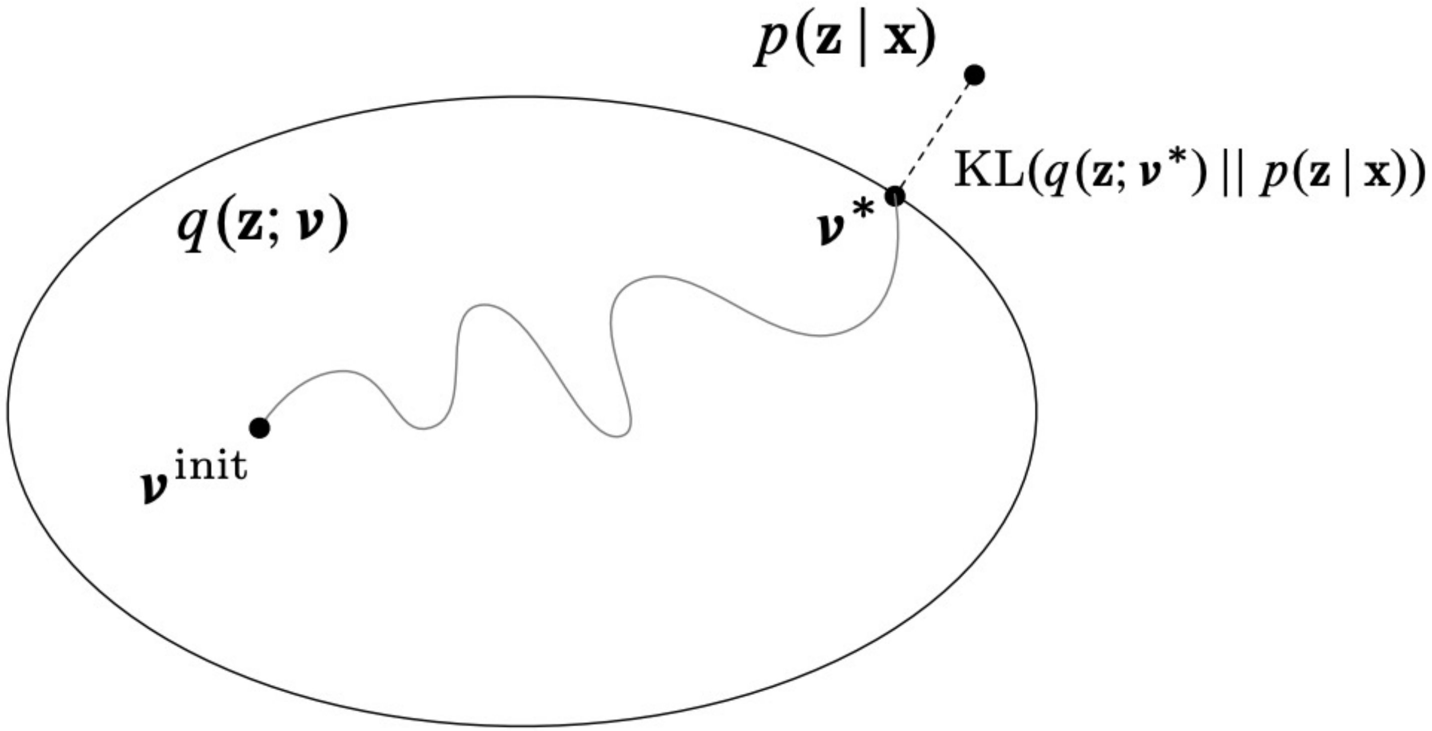
  - Wake sleep

Wake: $\max_{\theta} \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$

Sleep: $\max_{\phi} \mathbb{E}_{p_\theta(x|z)}[\log q_\phi(z|x)]$

Variational Autoencoders (VAEs)

Recall Variational Inference



VI solves **inference** with **optimization.**

Recall EM and the ELBO

$$\log p(x \mid \theta) = E_{z\sim q}[\log p(x, z \mid \theta)] + H(q) + KL(q(z \mid x) \,||\, p(z \mid x, \theta))$$

EM: Let $q_t(z \mid x) = p(z \mid x, \theta_t)$.
Max $p(x \mid \theta)$ by iterating:

$$Q(\theta', \theta_t) = E_{z\sim p(z|\theta_t)}[\log p(x, z \mid \theta')]$$
$$\theta_{t+1} = \text{argmax}_{\theta'} Q(\theta', \theta_t)$$

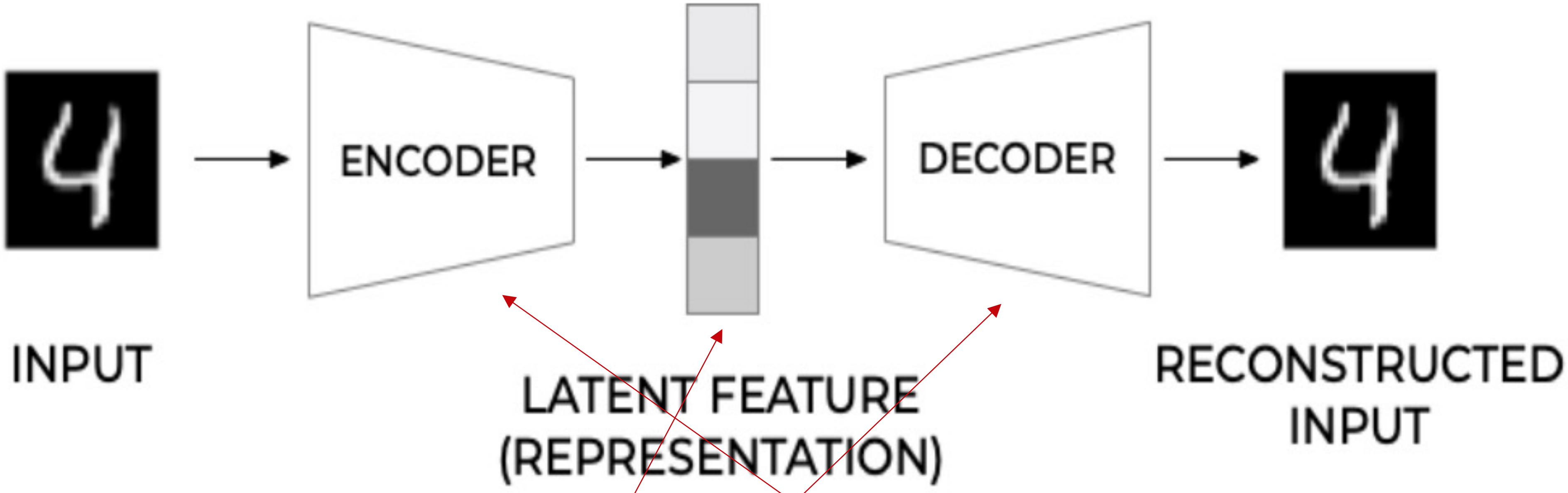
Variational Inference: Let $q(z \mid x)$ be some family that's easier to optimize.

$$\log p(x \mid \theta) \geq \underbrace{E_{z\sim q}[\log p(x, z \mid \theta)] + H(q)}_{\text{"ELBO": Evidence Lower Bound}}$$

equivalently,

$$\text{ELBO} = \log p(x \mid \theta) - KL(q(z \mid x) || p(z, x\theta))$$

Autoencoders



$$\tilde{x} = f(h) = f\big(g(x)\big)$$

[Michelucci 2022]

Variational Autoencoders

- [Kingma & Welling, 2014]
- Use variational inference with an inference model
  - Enjoy similar applicability with wake-sleep algorithm
- Generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, and prior $p(\boldsymbol{z})$
  - Joint distribution $p_\theta(\boldsymbol{x},\boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$
- Inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$

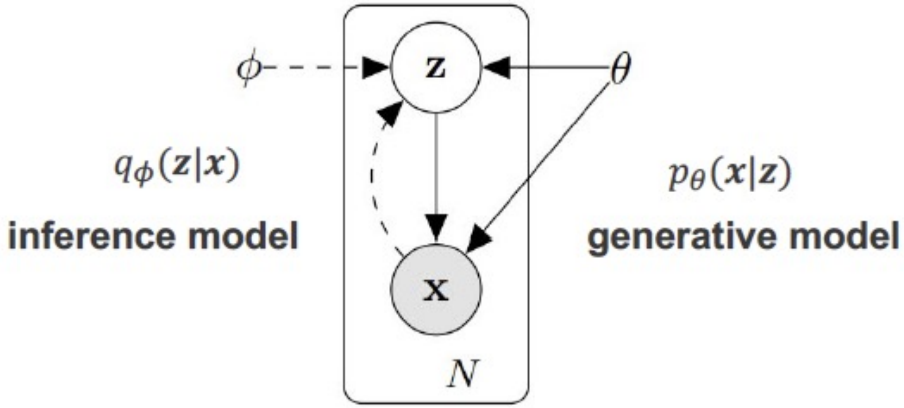
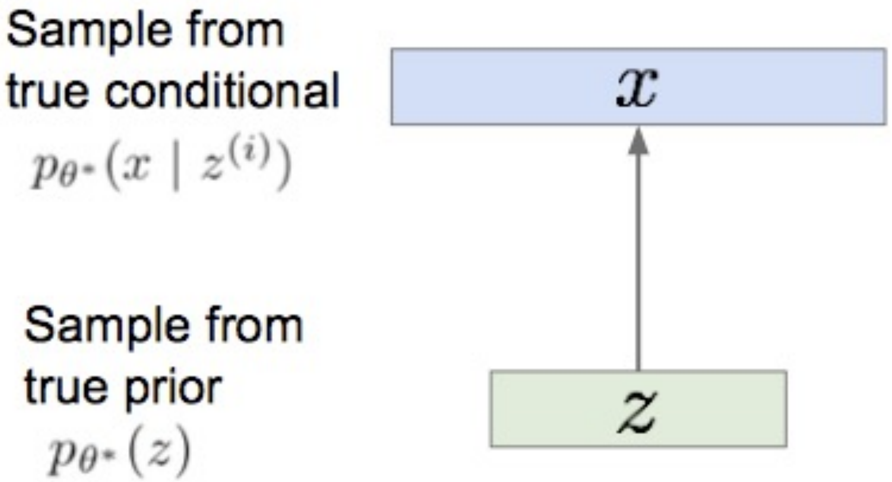


Figure courtesy: Kingma & Welling, 2014

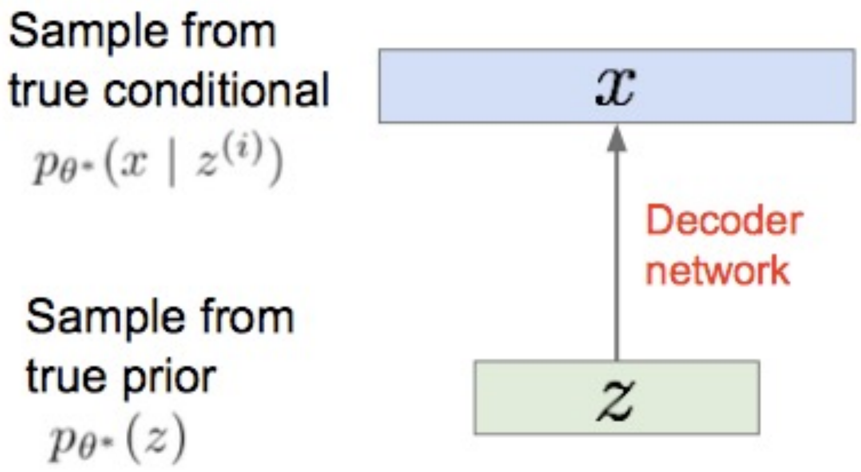
Variational Autoencoders



We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Variational Autoencoders



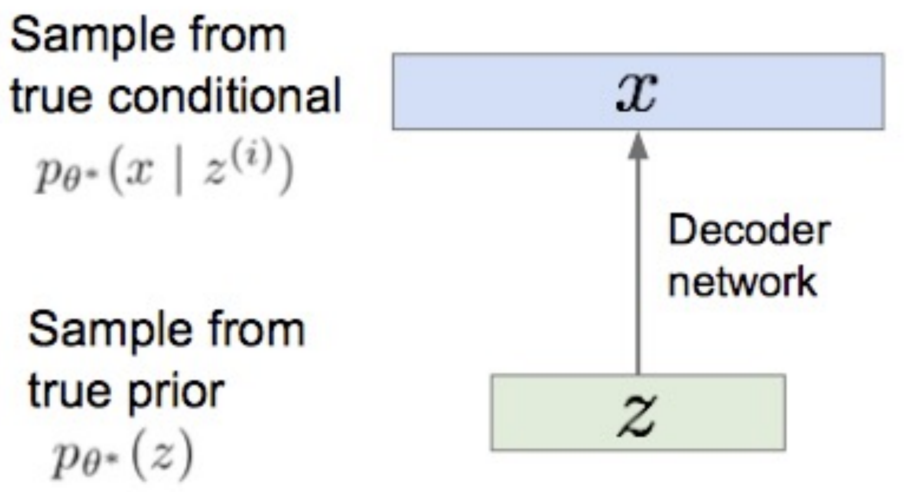
We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Choose prior p(z) to be simple, e.g. Gaussian.

Conditional p(x|z) is complex (generates image) => represent with neural network

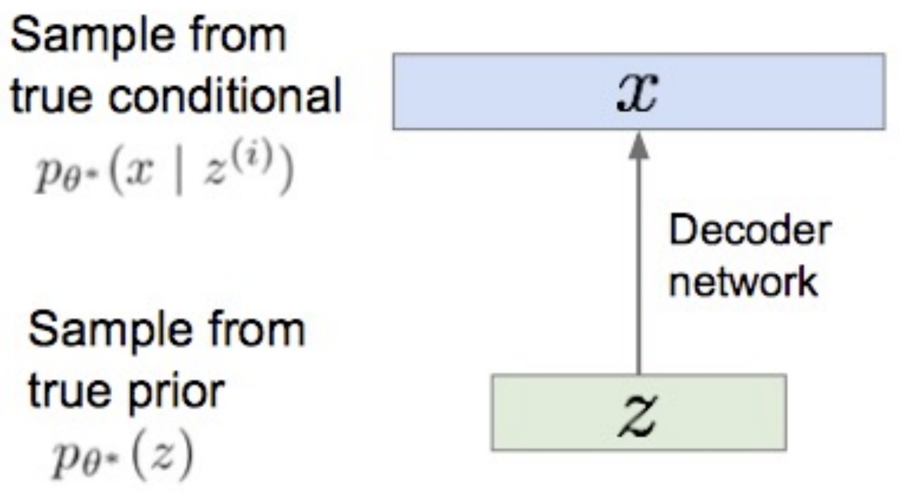
Variational Autoencoders



We want to estimate the true parameters θ^* of this generative model.

How to train the model?

Variational Autoencoders



We want to estimate the true parameters θ^* of this generative model.

How to train the model?

maximize likelihood of training data

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Now with latent z

Variational Autoencoders

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

Variational Autoencoders

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

Intractible to compute p(x|z) for every z!

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

Intractable data likelihood

Variational Autoencoders

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

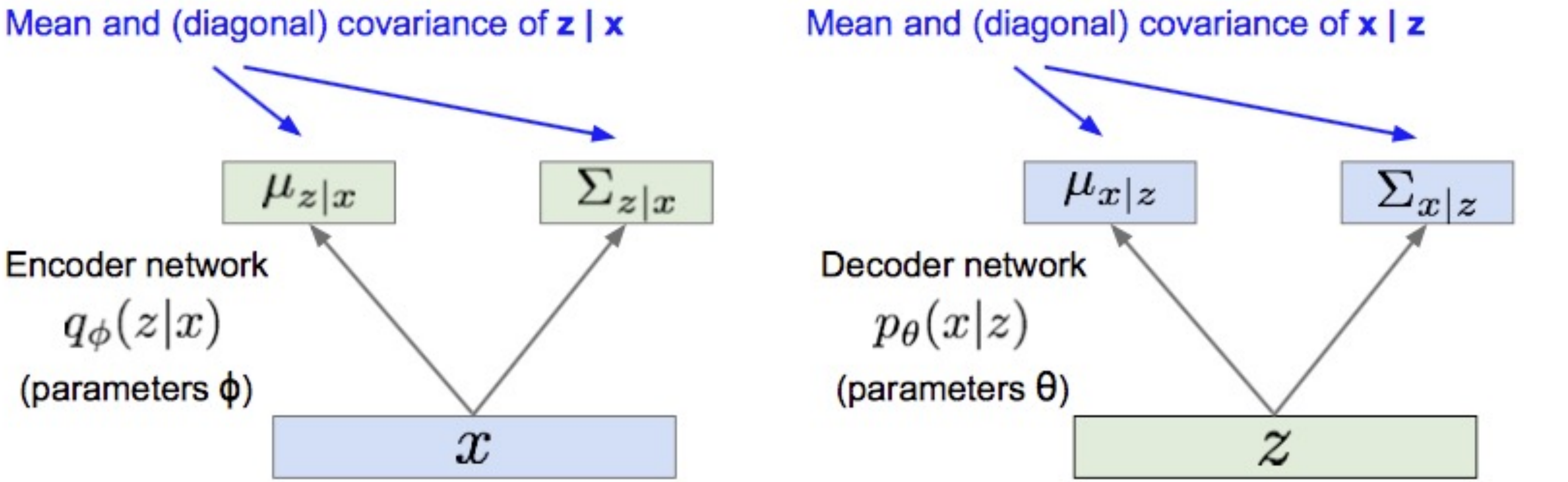
Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$

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 Autoencoders

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



Variational Autoencoders

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

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

Variational Autoencoders

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

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

Variational Autoencoders

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

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

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

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 Autoencoders: Reparameterization Trick

We want to use gradient descent to learn the model's parameters

Given z drawn from $q_\theta(z|x)$, how do we take derivatives of (a function of) z w.r.t. θ?

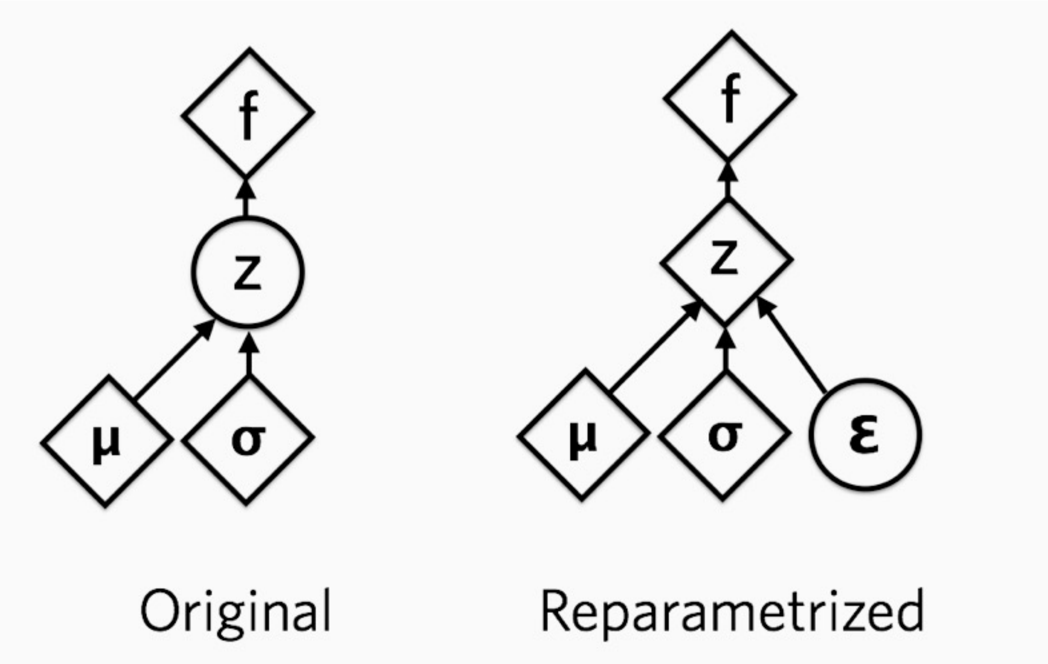
We can reparameterize: $z = \mu + \sigma \odot \epsilon$

$\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\odot$ is element-wise product

Can take derivatives of (functions of) z w.r.t. μ and σ

Output of $q_\theta(z|x)$ is vector of μ's and vector of σ's

Variational Autoencoders: Reparameterization Trick



Variational Autoencoders: Reparameterization Trick

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

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

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

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

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

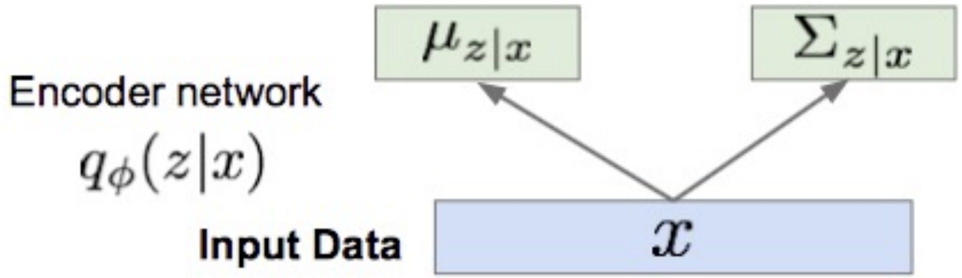
Let's look at computing the bound (forward pass) for a given minibatch of input data

Input Data x

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

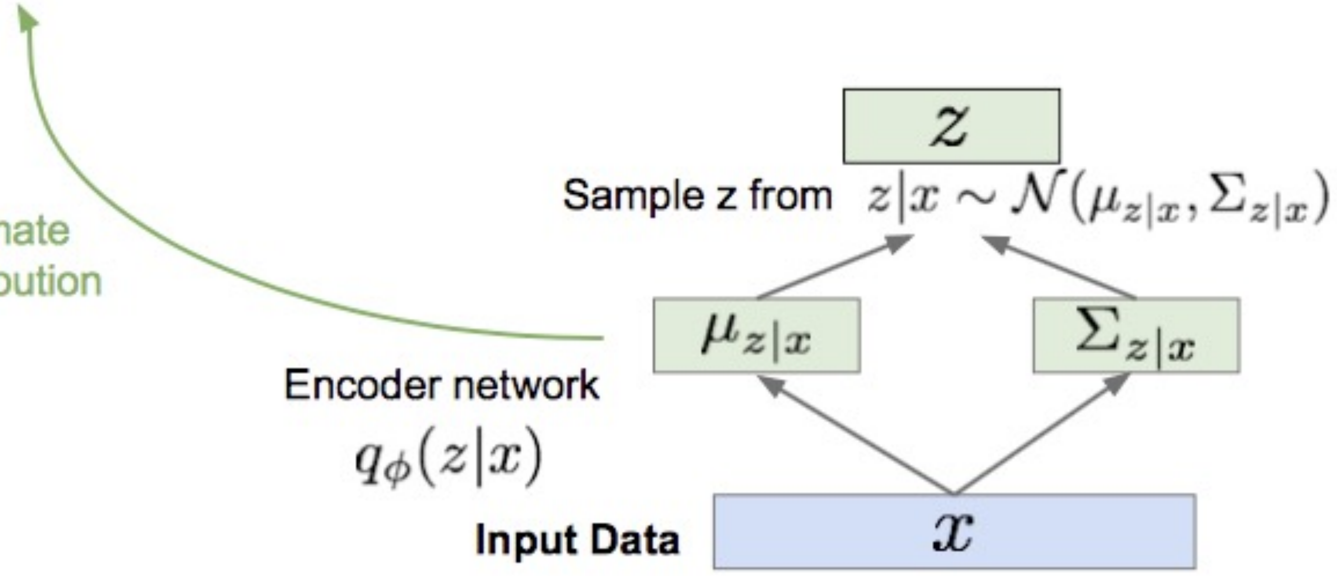


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior

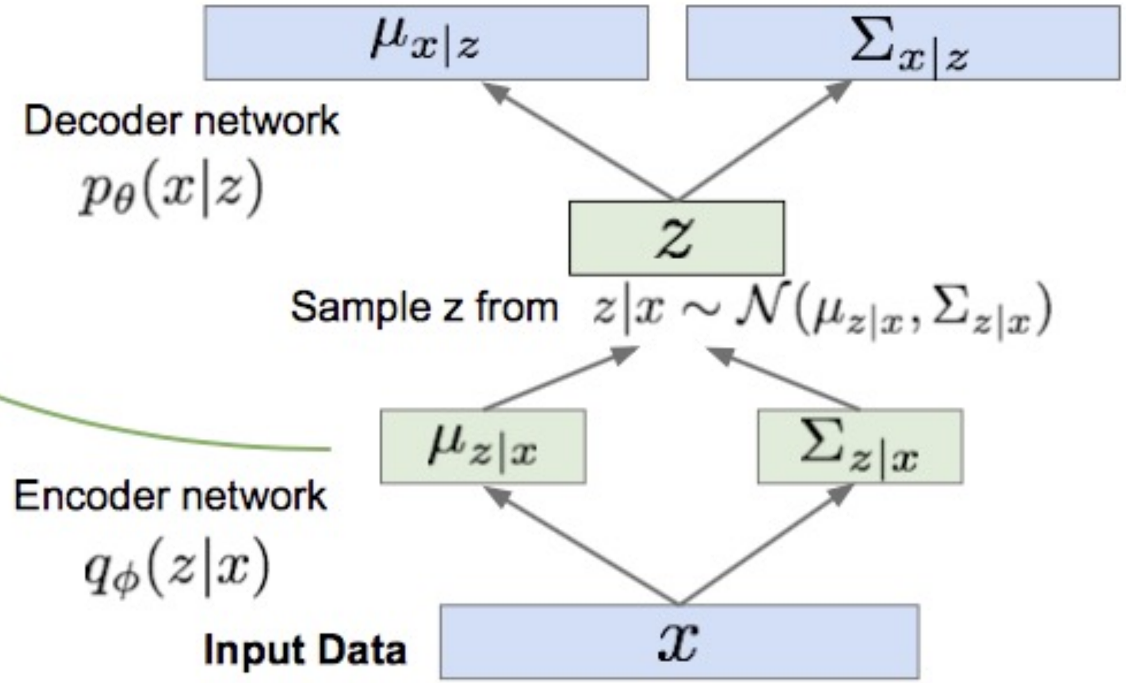


Variational Autoencoders

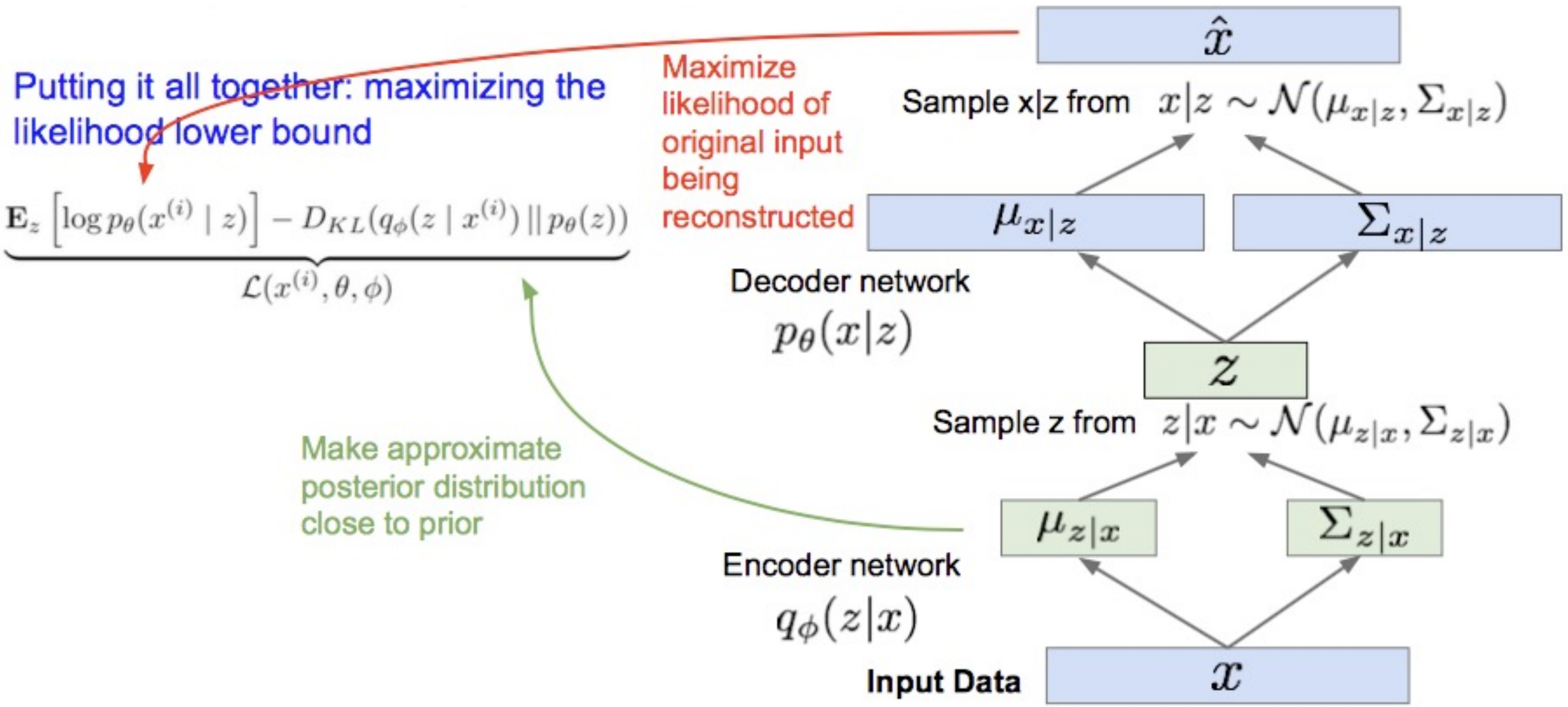
Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior

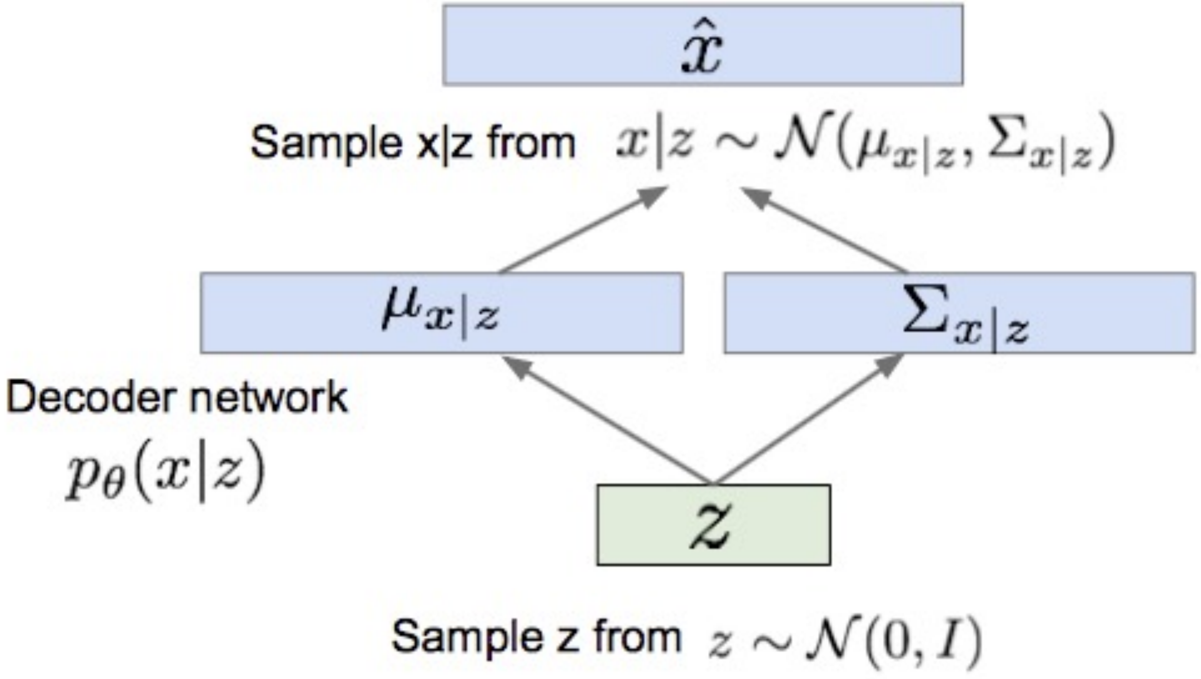


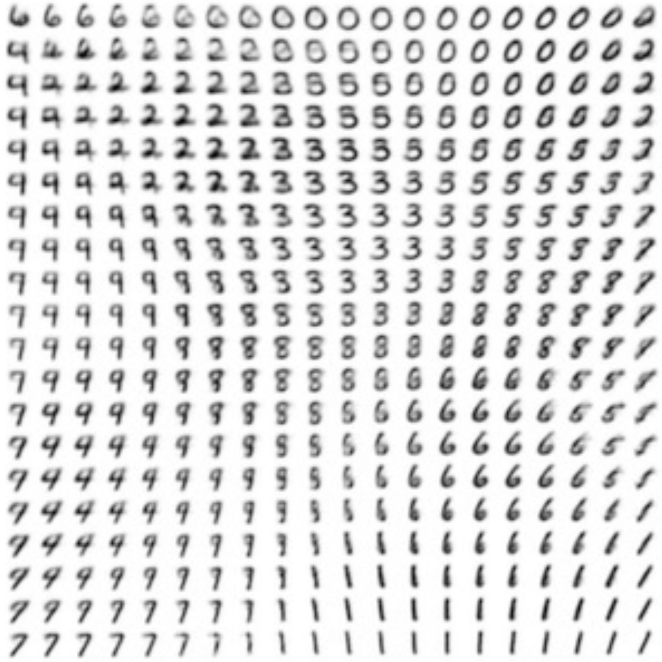
Variational Autoencoders



Variational Autoencoders: Generating

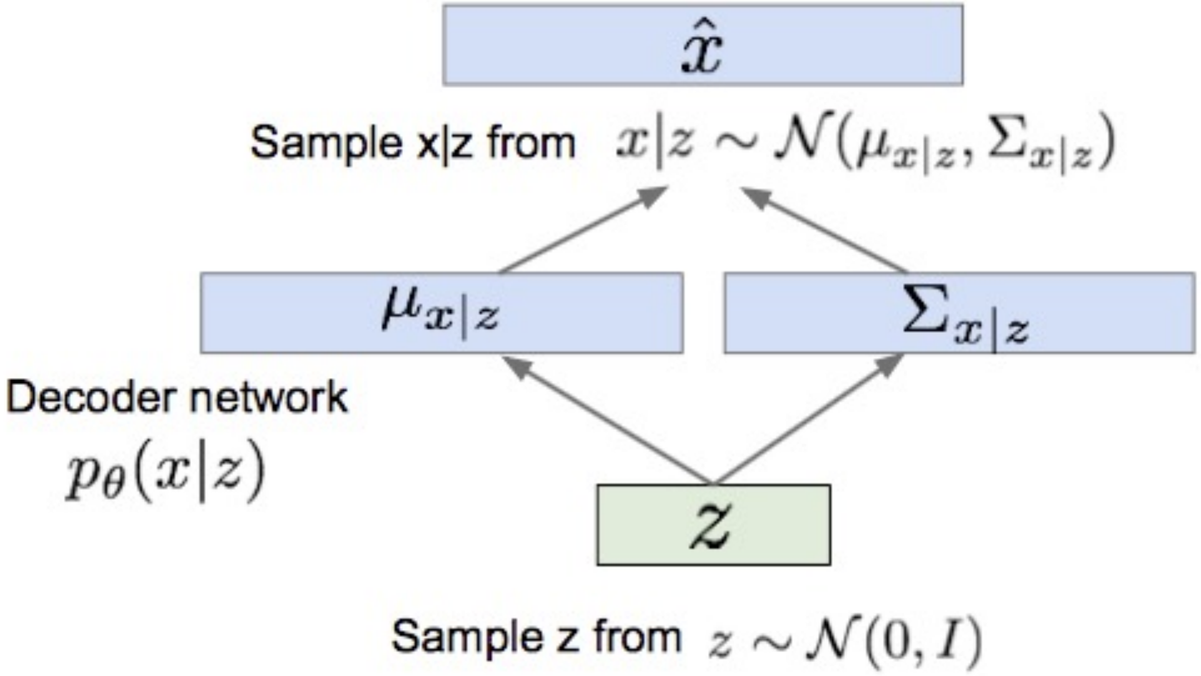
Use decoder network. Now sample z from prior!





Variational Autoencoders: Generating

Use decoder network. Now sample z from prior!



- VAEs tend to generate blurred images due to the mode covering behavior (more later)

Celebrity faces [Radford 2015]

Generative Adversarial Networks (GANs)

Generative Adversarial Nets (GANs)

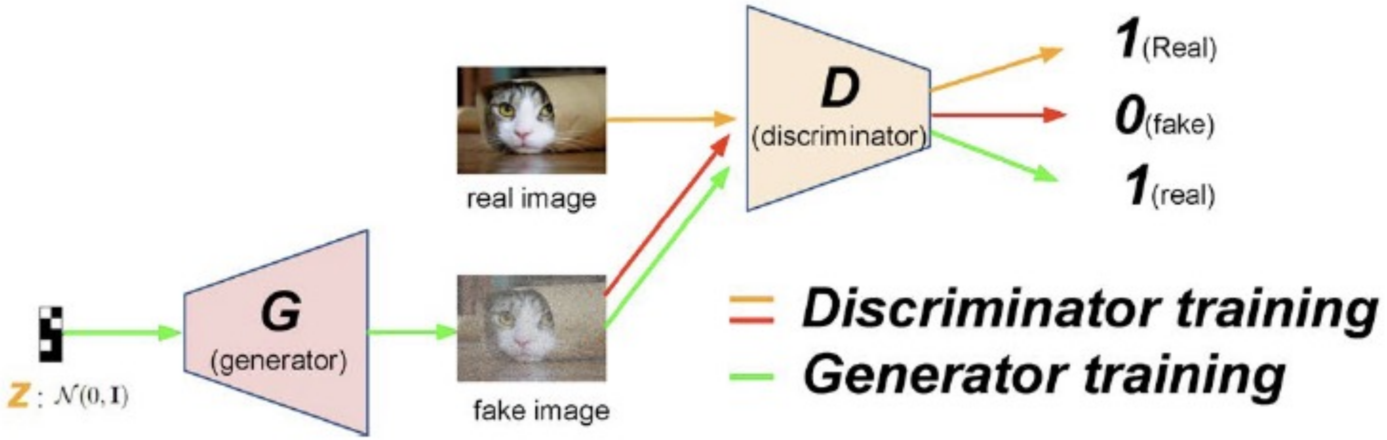
- [Goodfellow et al., 2014]
- Generative model $\boldsymbol{x} = G_\theta(\boldsymbol{z}), \ \ \boldsymbol{z} \sim p(\boldsymbol{z})$
  - Map noise variable $\boldsymbol{z}$ to data space $\boldsymbol{x}$
  - Define an implicit distribution over $\boldsymbol{x}$: $p_{g_\theta}(\boldsymbol{x})$
    - a stochastic process to simulate data $\boldsymbol{x}$
    - Intractable to evaluate likelihood
- Discriminator $D_\phi(\boldsymbol{x})$
  - Output the probability that $\boldsymbol{x}$ came from the data rather than the generator
- No explicit inference model
- No obvious connection to previous models with inference networks like VAEs
  - We will build formal connections between GANs and VAEs later

GANs

- Learning
  - A minimax game between the generator and the discriminator
  - Train D to maximize the probability of assigning the correct label to both training examples and generated samples
  - Train G to fool the discriminator

$$\max_D \mathcal{L}_D = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})} [\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim G(\boldsymbol{z}), \boldsymbol{z} \sim p(\boldsymbol{z})} [\log(1 - D(\boldsymbol{x}))]$$

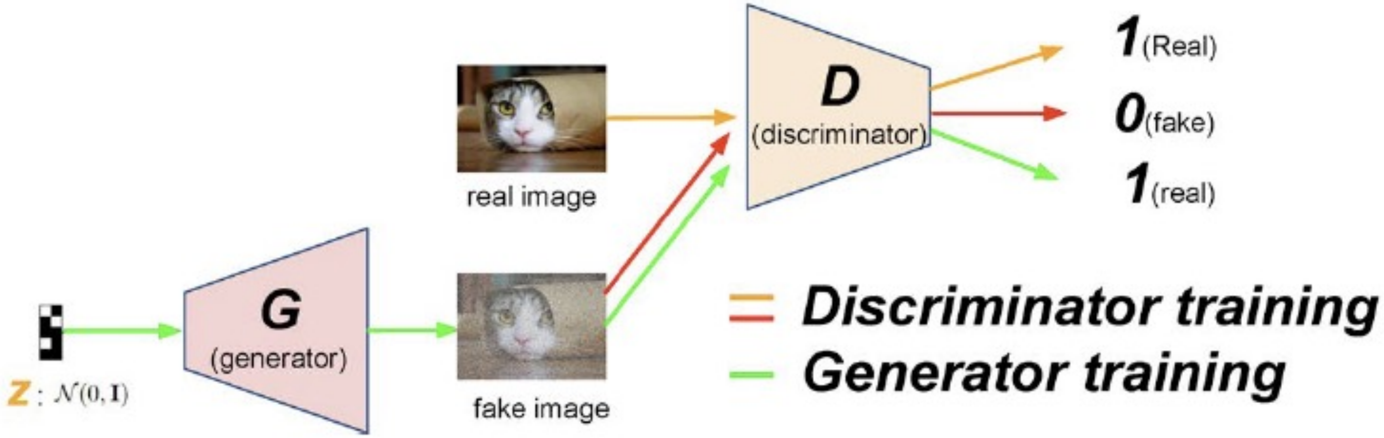
$$\min_G \mathcal{L}_G = \mathbb{E}_{\boldsymbol{x} \sim G(\boldsymbol{z}), \boldsymbol{z} \sim p(\boldsymbol{z})} [\log(1 - D(\boldsymbol{x}))] .$$



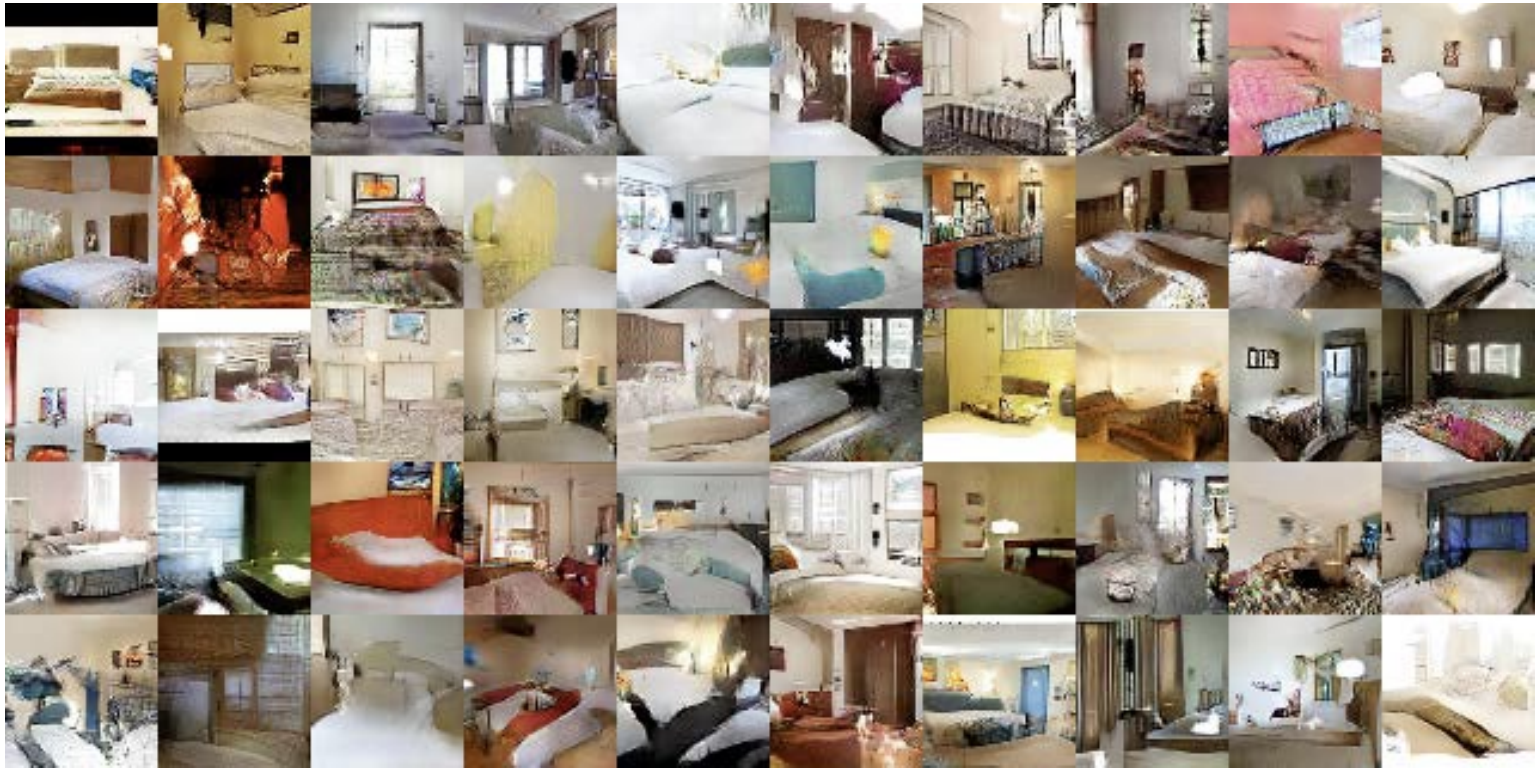
GANs

- Learning
  - A minimax game between the generator and the discriminator
  - Train D to maximize the probability of assigning the correct label to both training examples and generated samples
  - Train G to fool the discriminator

$$\max_D \mathcal{L}_D = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z}), \boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1 - D(\boldsymbol{x}))\right]$$
$$\min_G \mathcal{L}_G = \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z}), \boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1 - D(\boldsymbol{x}))\right].$$



GANs: example results



[Radford et al., 2016]

GANs and VAEs: A Unified View

GANs

- Implicit distribution over $x \sim p_\theta(x \mid y)$

$$p_\theta(\boldsymbol{x}|y) = \begin{cases} p_{g\theta}(\boldsymbol{x}) & y = 0 \\ p_{data}(\boldsymbol{x}) & y = 1. \end{cases}$$

(distribution of generated images)

(distribution of real images)

- $x \sim p_{g_\theta}(x)$ ↔️ $x = G_\theta(z), z \sim p(z \mid y = 0)$
- $x \sim p_{data}(x)$

GANs: Rewrite in Variational-EM format

- The familiar "Variational-EM" format:

$$\max_{\boldsymbol{\phi}} \mathcal{L}_{\phi} = \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=0)}\left[\log(1-D_\phi(\boldsymbol{x}))\right] + \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log D_\phi(\boldsymbol{x})\right]$$

$$\max_{\boldsymbol{\theta}} \mathcal{L}_{\theta} = \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=0)}\left[\log D_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log(1-D_\phi(\boldsymbol{x}))\right]$$

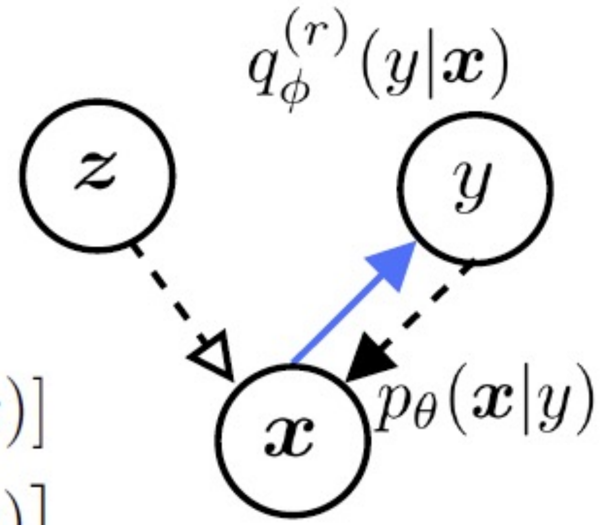
$$= \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=0)}\left[\log D_\phi(\boldsymbol{x})\right]$$

- Implicit distribution over $x \sim p_\theta(x \mid y)$:

$$x = G_\theta(z), z \sim p(z \mid y = 0)$$

- Discriminator distribution $q_\phi(y \mid x)$:

$$q_\phi^r(y \mid x) = q_\phi(1 - y \mid x)$$



$$\max_{\boldsymbol{\phi}} \mathcal{L}_{\phi} = \mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)}\left[\log q_\phi(y|\boldsymbol{x})\right]$$

$$\max_{\boldsymbol{\theta}} \mathcal{L}_{\theta} = \mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)}\left[\log q_\phi^r(y|\boldsymbol{x})\right]$$

Variational EM vs. GANs

- Variational EM
  - Objectives

    $$\max_{\phi}\mathcal{L}_{\phi,\theta} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + KL\left(q_\phi(z|x)||p(z)\right)$$
    $$\max_{\theta}\mathcal{L}_{\phi,\theta} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + KL\left(q_\phi(z|x)||p(z)\right)$$
  - Single objective for both θ and ϕ
  - Extra prior regularization by $p(z)$
- The reconstruction term:
  - Maximize the conditional log-likelihood of x with the generative distribution $p_\theta(x \mid z)$ conditioning on the latent code z inferred by $q_\phi(z \mid x)$
- $p_\theta(x \mid z)$ is the generative model
- $q_\phi(z \mid x)$ is the inference model

- GAN
  - Objectives

    $$\max_{\boldsymbol{\phi}}\mathcal{L}_{\phi} = \mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)}\left[\log q_\phi(y|\boldsymbol{x})\right]$$
    $$\max_{\boldsymbol{\theta}}\mathcal{L}_{\theta} = \mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)}\left[\log q^r_\phi(y|\boldsymbol{x})\right]$$
  - Two objectives
- Maximize the conditional log-likelihood of y with the distribution $q_\phi(y \mid x)$ conditioning on data/generation x from $p_\theta(x \mid y)$
- $q_\phi(y \mid x)$ is the generative model
- $p_\theta(x \mid y)$ is the inference model

GANs vs VAEs: A Symmetry

Hu et al. "Unifying Deep Generative Models"

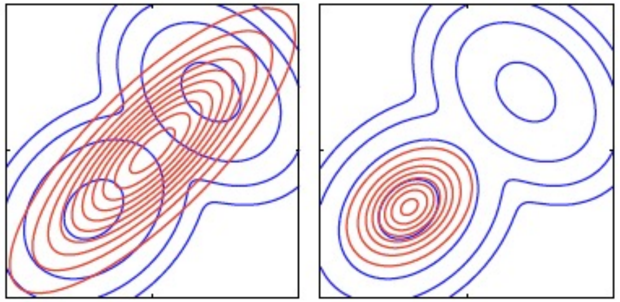
	GANs (InfoGAN)	VAEs
Generative distribution	$p_\theta(\boldsymbol{x}\|y) = \begin{cases} p_{g_\theta}(\boldsymbol{x}) & y=0 \\ p_{data}(\boldsymbol{x}) & y=1. \end{cases}$	$p_\theta(\boldsymbol{x}\|\boldsymbol{z}, y) = \begin{cases} p_\theta(\boldsymbol{x}\|\boldsymbol{z}) & y=0 \\ p_{data}(\boldsymbol{x}) & y=1. \end{cases}$
Discriminator distribution	$q_\phi(y\|\boldsymbol{x})$	$q_*(y\|\boldsymbol{x})$, perfect, degenerated
$\boldsymbol{z}$-inference model	$q_\eta(\boldsymbol{z}\|\boldsymbol{x}, y)$ of InfoGAN	$q_\eta(\boldsymbol{z}\|\boldsymbol{x}, y)$
KLD to minimize	$\min_\theta \mathrm{KL}\left(p_\theta(\boldsymbol{x}\|y) \,\|\|\, q^r(\boldsymbol{x}\|\boldsymbol{z}, y)\right)$ $\sim \min_\theta \mathrm{KL}(P_\theta \,\|\|\, Q)$	$\min_\theta \mathrm{KL}\left(q_\eta(\boldsymbol{z}\|\boldsymbol{x}, y) q_*^r(y\|\boldsymbol{x}) \,\|\|\, p_\theta(\boldsymbol{z}, y\|\boldsymbol{x})\right)$ $\sim \min_\theta \mathrm{KL}(Q \,\|\|\, P_\theta)$

GANs vs VAEs: A Symmetry

Hu et al. "Unifying Deep Generative Models"

	GANs (InfoGAN)	VAEs
KLD to minimize	$\min_\theta \text{KL}\left(p_\theta(\boldsymbol{x}\|y) \,\|\|\, q^r(\boldsymbol{x}\|\boldsymbol{z}, y)\right)$ $\sim \min_\theta \text{KL}(P_\theta \,\|\|\, Q)$	$\min_\theta \text{KL}(q_\eta(\boldsymbol{z}\|\boldsymbol{x}, y) q_*^r(y\|\boldsymbol{x}) \,\|\|\, p_\theta(\boldsymbol{z}, y\|\boldsymbol{x}))$ $\sim \min_\theta \text{KL}(Q \,\|\|\, P_\theta)$

- Asymmetry of KLDs inspires combination of GANs and VAEs
  - GANs: $\min_\theta \text{KL}(P_\theta || Q)$ tends to missing mode
  - VAEs: $\min_\theta \text{KL}(Q || P_\theta)$ tends to cover regions with small values of p_{data}



Mode covering   Mode missing

Diffusion Models

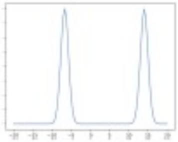
Diffusion Models

- DDPM (Ho-Jain-Abbeel 2020) : Produce new data using a sequence of denoising steps.
  Denoising Diffusion Probabilistic Models

$$x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_{t-1} \leftarrow x_t \leftarrow x_{t+1} \leftarrow \cdots \leftarrow x_{T-1} \leftarrow x_T$$

$x_0 \sim p(x)$

Probability of training/real data (what we want to estimate)



$$x_{t-1} \sim p(x_{t-1}|x_t) = \mathcal{N}(\mu_{\text{denoise}}(x_t, n_t), \sigma^2_{\text{denoise}}(t)\text{I})$$

$$x_{t-1} = a^x_t x_t - b^n_t n_t + \sigma^2_{\text{denoise}}(t)z, \;\; z \sim \mathcal{N}(0, \text{I})$$

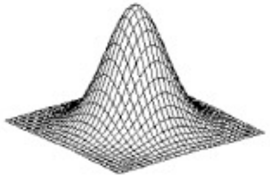
x_{t-1} is produced with a linear combination of the current denoised data x_t, an estimation of the noise n_t and a pure random noise z.

Starting from pure noise $x = x_T$, gradually remove noise to generate intermediate states x_{T-1}, x_{T-2},... until reaching a clean data $x = x_0$, which belongs to the training distribution p(x).

Backward process

$x_T \sim \mathcal{N}(0, \text{I})$

x_T is sampled from a prior (unconditional) probability.



[Slide from Xavier Bresson]

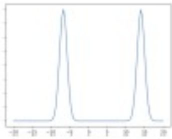
Diffusion Models

- The denoising steps are learned from the forward pass, which consists in adding noise to the original data :

$$x_0 \rightarrow x_1 \rightarrow \cdots \rightarrow x_{t-1} \rightarrow x_t \rightarrow x_{t+1} \rightarrow \cdots \rightarrow x_{T-1} \rightarrow x_T$$

$x_0 \sim p(x)$

Probability of training/real data



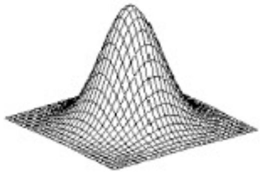
$$x_t \sim p(x_t|x_{t-1}) = \mathcal{N}\left(\mu_{\text{noise}}(x_{t-1}), \sigma^2_{\text{noise}}(t)\mathrm{I}\right)$$

$$x_t = c_t^x x_{t-1} + \sigma^2_{\text{noise}}(t) z, \quad z \sim \mathcal{N}(0, \mathrm{I})$$

x_t is computed with a linear combination of the current noisy data x_{t-1} and a pure random noise z.

$x_T \sim \mathcal{N}(0, \mathrm{I})$

x_T samples from a prior probability



Starting with clean data $x = x_0$ from the training set, add noise to generate intermediate states x_1, x_2,... until reaching a high noise level $x = x_T$, where the original structure is no longer recognizable.

Forward process

[Slide from Xavier Bresson]

Questions?

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON