Topics:

- Generative Models
- Pixel CNN
- Variational Autoencoders

CS 4644-DL / 7643-A

ZSOLT KIRA

- **Projects!**
  - Due May 1rd (May 3th with grace period)
  - Cannot extend due to grade deadlines!

- Outline of rest of course:

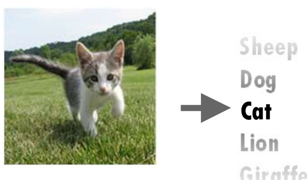
W14: Apr 14	Variational Autoencoders	•Tutorial on Variational Autoencoders
W15: Apr 19	Diffusion Models	
W15: Apr 21	Emerging trends, wrap-up.	

Introduction

Spectrum of Low-Labeled Learning

Supervised Learning

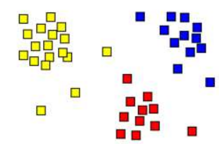
- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, $P(y|x)$
- e.g. classification



Less Labels →

Unsupervised Learning

- Input: $\{X\}$
- Learning output: $P(x)$
- Example: Clustering, density estimation, etc.



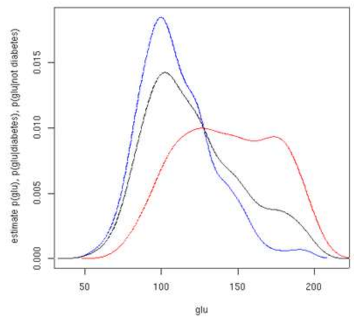
Supervised Learning

x	→	Classification	→ y	Discrete
x	→	Regression	→ y	Continuous

Unsupervised Learning

x	→	Clustering	→ c	Discrete
x	→	Dimensionality Reduction	→ z	Continuous
x	→	Density Estimation	→ p(x)	On simplex

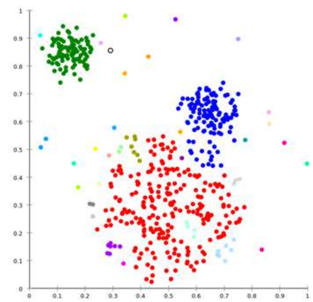
Traditional unsupervised learning methods:



Density estimation

Modeling $P(x)$

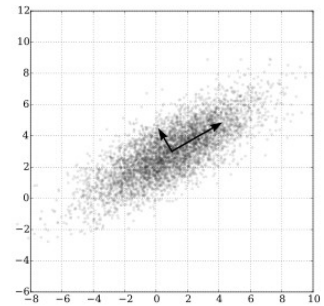
Deep Generative Models



Clustering

Comparing/ Grouping

Metric learning & clustering



Principal Component Analysis

Representation Learning

Almost all deep learning!

Similar in deep learning, but **from neural network/learning perspective**

What to Learn?

Discriminative vs. Generative Models

- Discriminative models model $P(y|x)$
  - Example: Model this via neural network, SVM, etc.

- Generative models model $P(x)$

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Discriminative vs. Generative Models

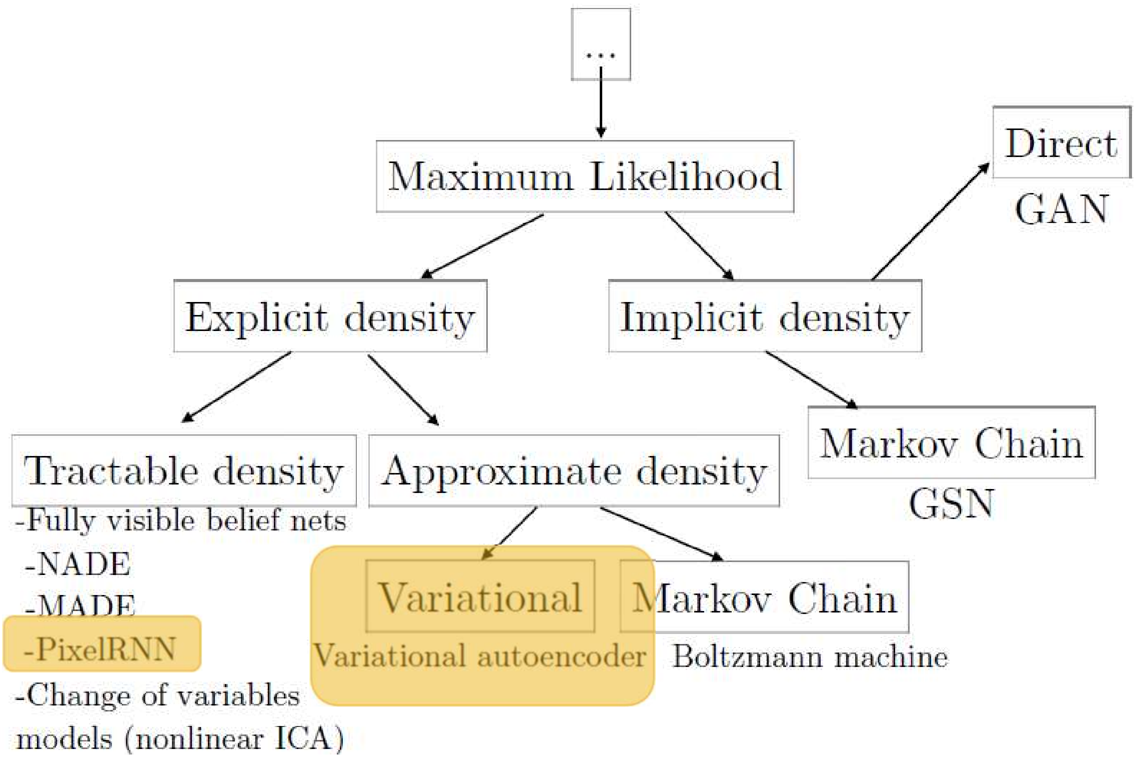
- Discriminative models model $P(y|x)$
  - Example: Model this via neural network, SVM, etc.

- Generative models model $P(x)$
- We can parameterize our model as $P(x, \theta)$ and use maximum likelihood to optimize the parameters given an unlabeled dataset:

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) \\ &= \arg\max_{\boldsymbol{\theta}} \log \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right).\end{aligned}$$

- They are called generative because they can often generate *samples*
  - Example: Multivariate Gaussian with estimated parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}$

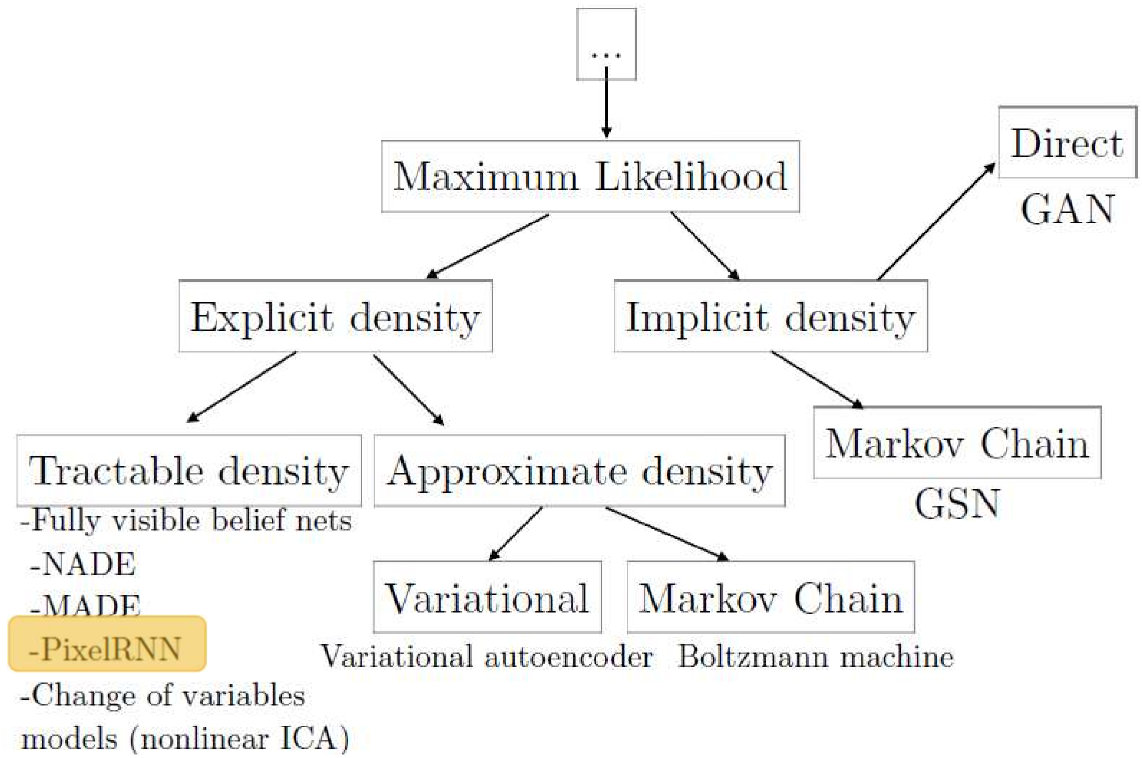
Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

PixelRNN & PixelCNN



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

We can use chain rule to decompose the joint distribution

- Factorizes joint distribution into a product of conditional distributions
  - Similar to Bayesian Network (factorizing a joint distribution)
  - Similar to language models!

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

- Requires some *ordering* of variables (edges in a probabilistic graphical model)
- We can estimate this conditional distribution as a neural network

Oord et al., Pixel Recurrent Neural Networks

Factorizing P(x)

$$\mathrm{p}(\boldsymbol{s}) = \mathrm{p}(w_1, w_2, \ldots, w_n)$$

$$= \mathrm{p}(w_1)\,\mathrm{p}(w_2 \mid w_1)\,\mathrm{p}(w_3 \mid w_1, w_2) \cdots \mathrm{p}(w_n \mid w_{n-1}, \ldots, w_1)$$

$$= \prod_i \mathrm{p}(\underbrace{w_i}_{\text{next word}} \mid \underbrace{w_{i-1}, \ldots, w_1}_{\text{history}})$$

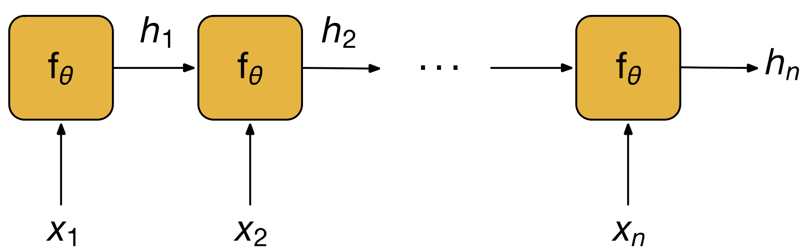
Modeling language as a sequence

Georgia Tech

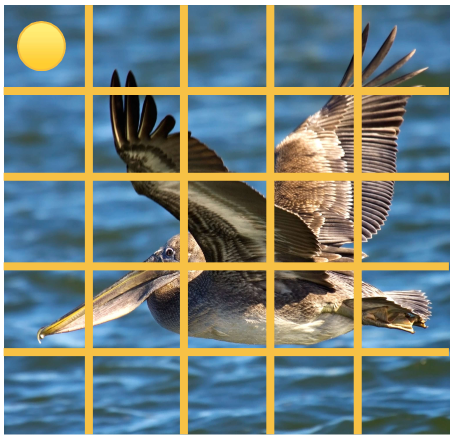
- Language modeling involves estimating a probability distribution over sequences of words.

$$p(\boldsymbol{s}) = p(w_1, w_2, \ldots, w_n) = \prod_i p(\underbrace{w_i}_{\text{next word}} \mid \underbrace{w_{i-1}, \ldots, w_1}_{\text{history}})$$

- RNNs are a family of neural architectures for modeling sequences.



Language Models as an RNN

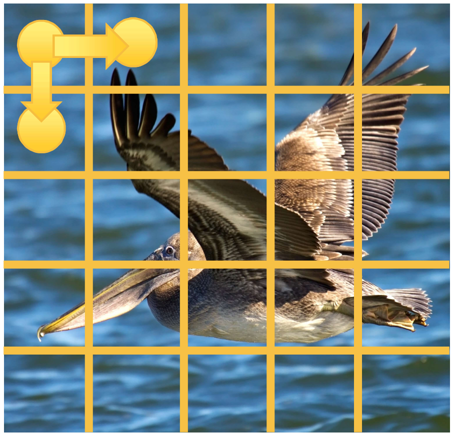


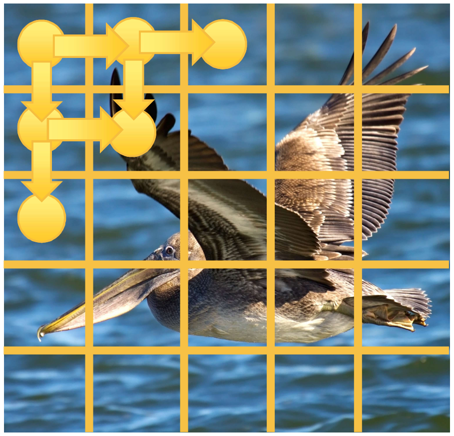
$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1})$$

$$p(x) = p(x_1) \prod_{i=2}^{n^2} p(x_i | x_1, \dots, x_{i-1})$$

Oord et al., Pixel Recurrent Neural Networks

Factorized Models for Images



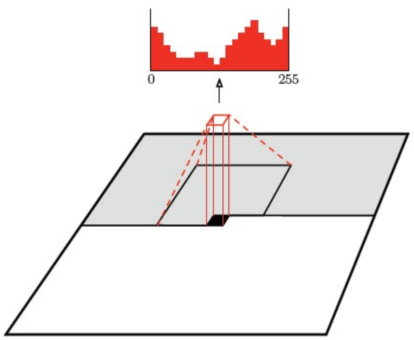


$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1)\prod_{i=1}^{n^2} p(x_i|x_1, \ldots, x_{i-1})$$

- Training:
  - We can train similar to language models: Teacher/student forcing
  - Maximum likelihood approach
- Downsides:
  - Slow sequential generation process
  - Only considers few context pixels

Oord et al., Pixel Recurrent Neural Networks

Factorized Models for Images



1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

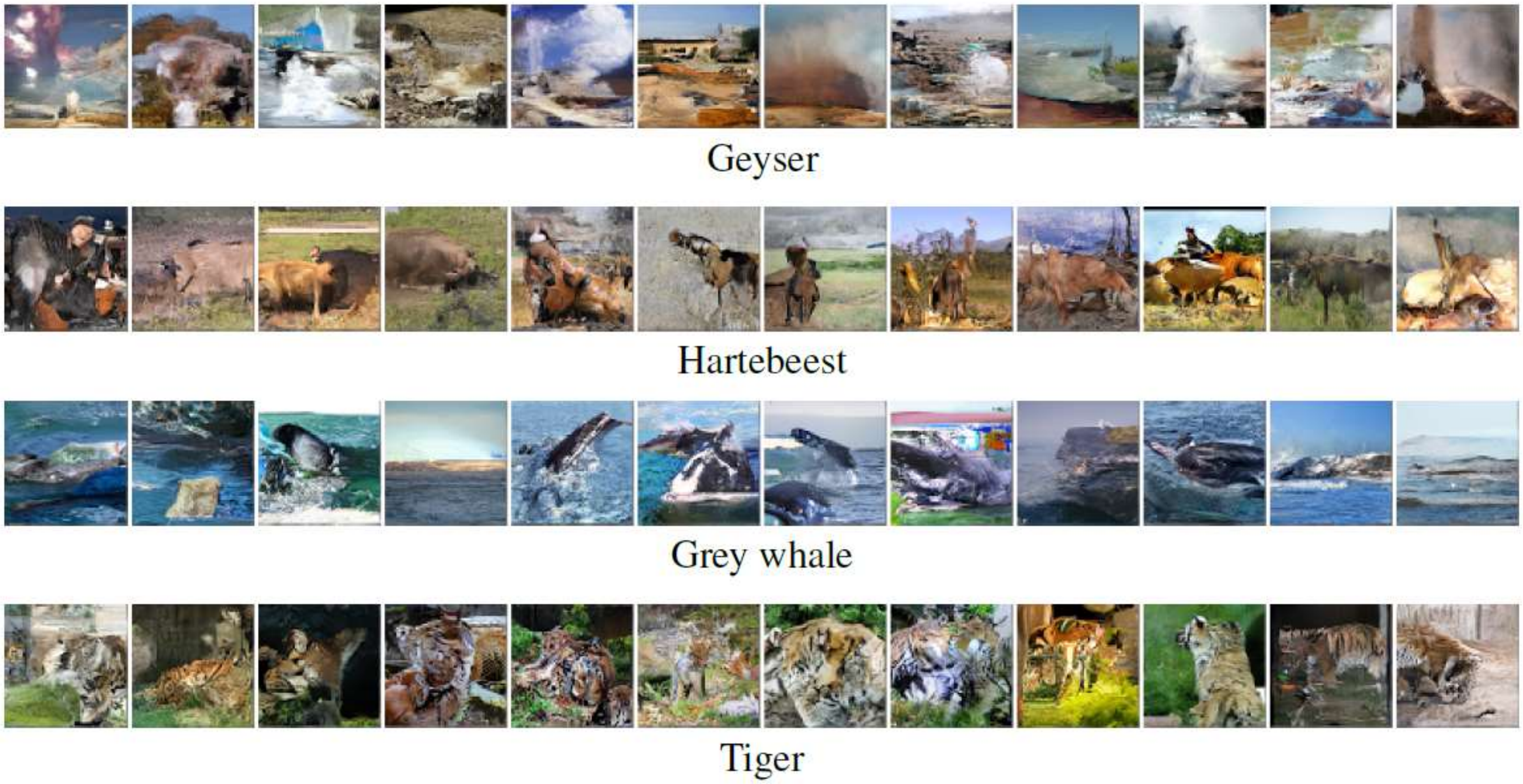
- **Idea:** Represent conditional distribution as a convolution layer!
- Considers larger context (receptive field)
- Practically can be implemented by applying a mask, zeroing out “future” pixels
- Faster training but still slow generation
  - Limited to smaller images

Oord et al., Conditional Image Generation with PixelCNN Decoders

Pixel CNN

Oord et al., *Conditional Image Generation with PixelCNN Decoders*

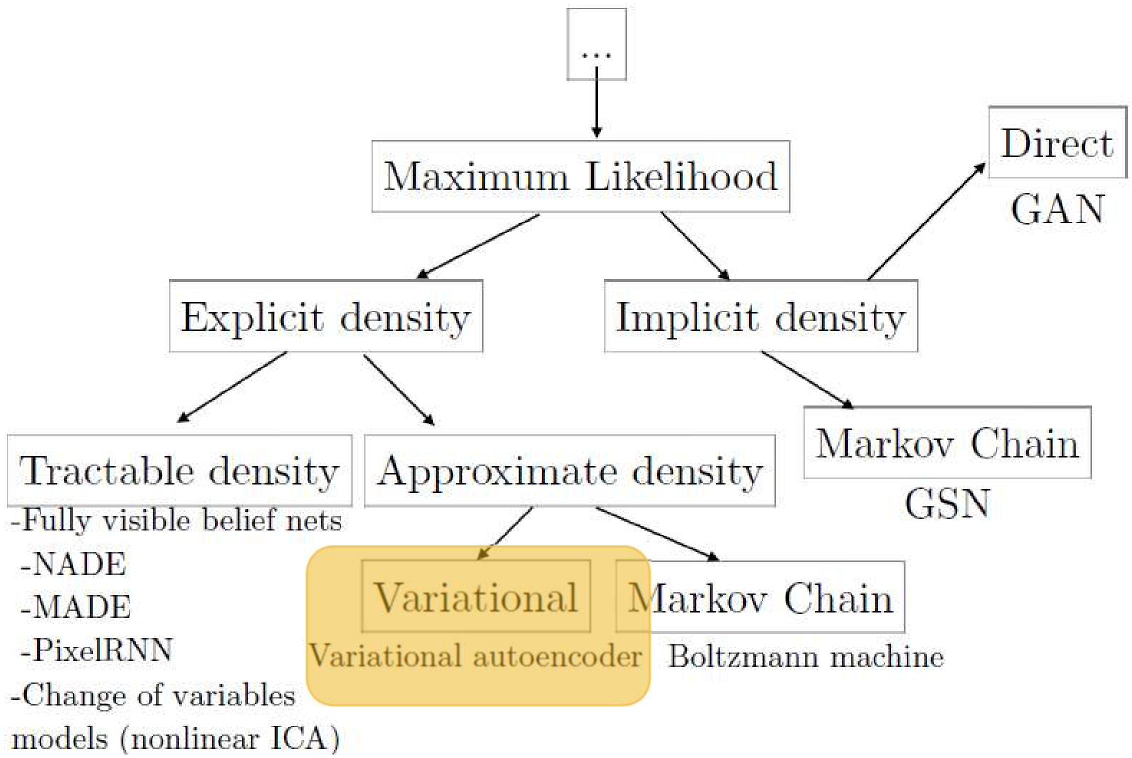
Example Results: Image Completion (PixelRNN)



Oord et al., *Conditional Image Generation with PixelCNN Decoders*

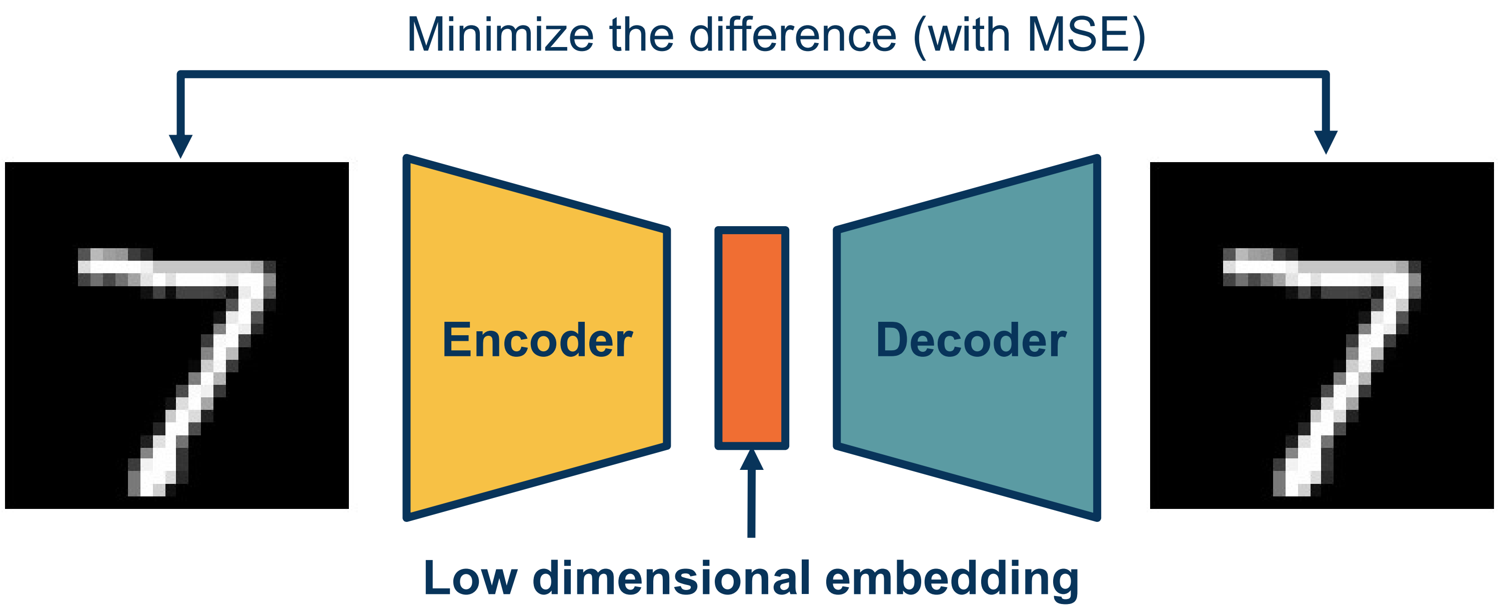
Example Images (PixelCNN)

Variational Autoencoders (VAEs)



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

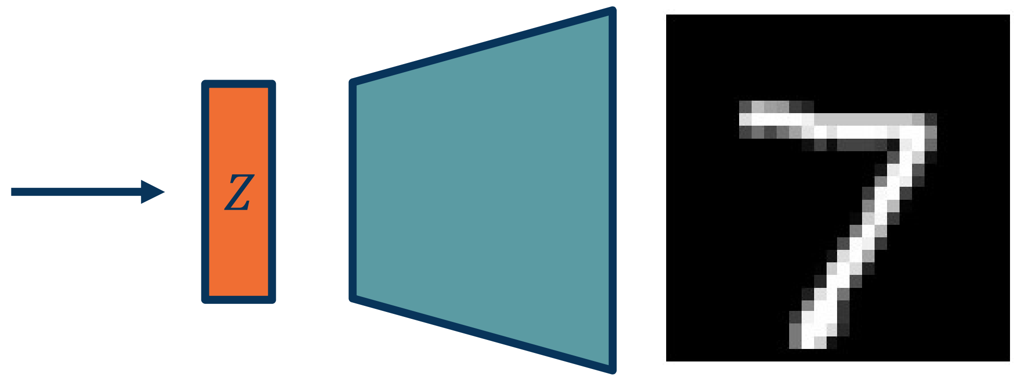


Linear layers with reduced dimension or Conv-2d layers with stride

Linear layers with increasing dimension or Conv-2d layers with bilinear upsampling

Autoencoders

What is this?
Hidden/Latent variables
Factors of variation that produce an image:
(digit, orientation, scale, etc.)



$$P(X) = \int P(X|Z;\theta)P(Z)dZ$$

- We cannot maximize this likelihood due to the integral
- Instead we maximize a variational *lower bound* (VLB) that we *can* compute

Kingma & Welling, Auto-Encoding Variational Bayes

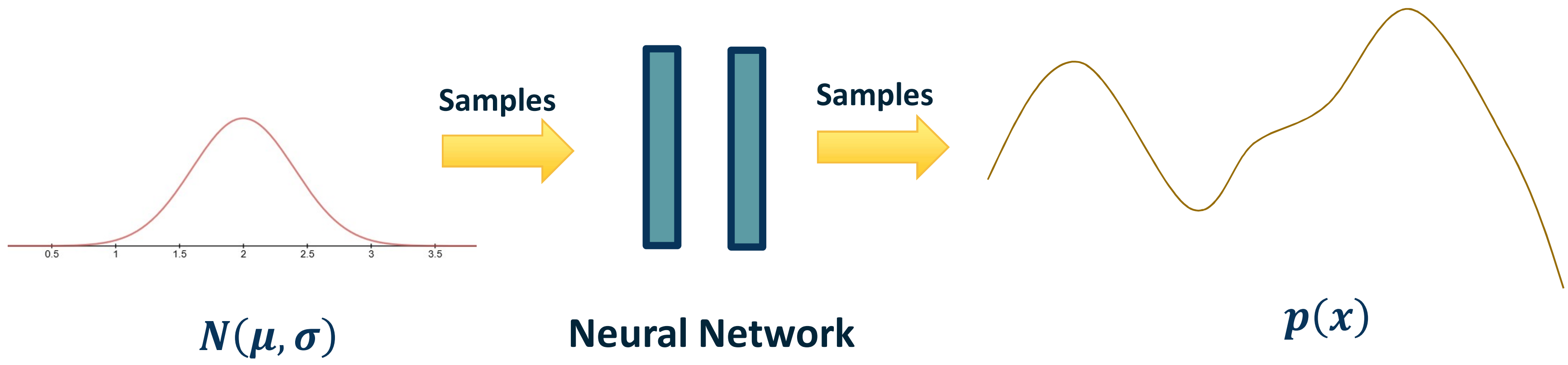
Formalizing the Generative Model

- We can combine the probabilistic view, sampling, autoencoders, and approximate optimization

- Sample Z from simpler distribution

Variational Autoencoder: Decoder

- We would like to *sample* from $p(x)$ using a neural network
- **Idea:**
  - Sample from a simple distribution (Gaussian)
  - Transform the sample to $p(x)$

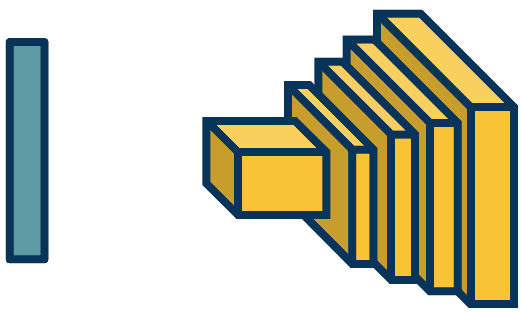


Learning to Sample

- Input can be a vector with (independent) Gaussian random numbers
- We can use a CNN to generate images!

Generator

Vector of Random Numbers



$N(\mu, \sigma)$ Neural Network $p(x)$

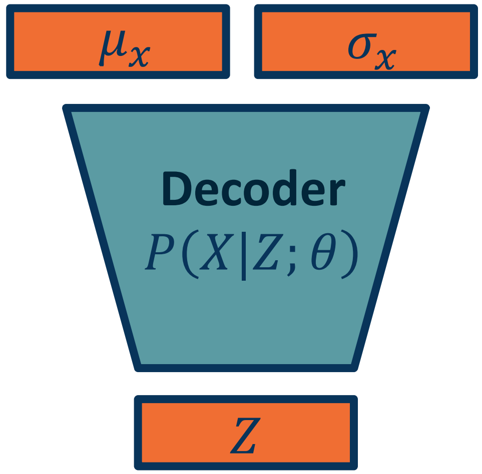
Generating Images

- We can combine the probabilistic view, sampling, autoencoders, and approximate optimization

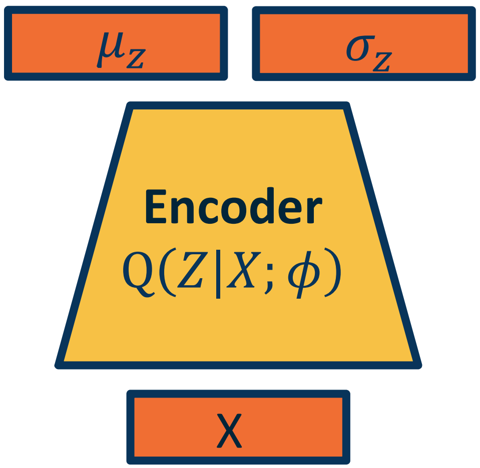
- Assume Z comes from simpler distribution (Normal)

- We can also output parameters of a probability distribution!
  - **Example**: μ, σ of Gaussian distribution
  - For multi-dimensional version output diagonal covariance

- How can we maximize $P(X) = \int P(X|Z;\theta)P(Z)dZ$



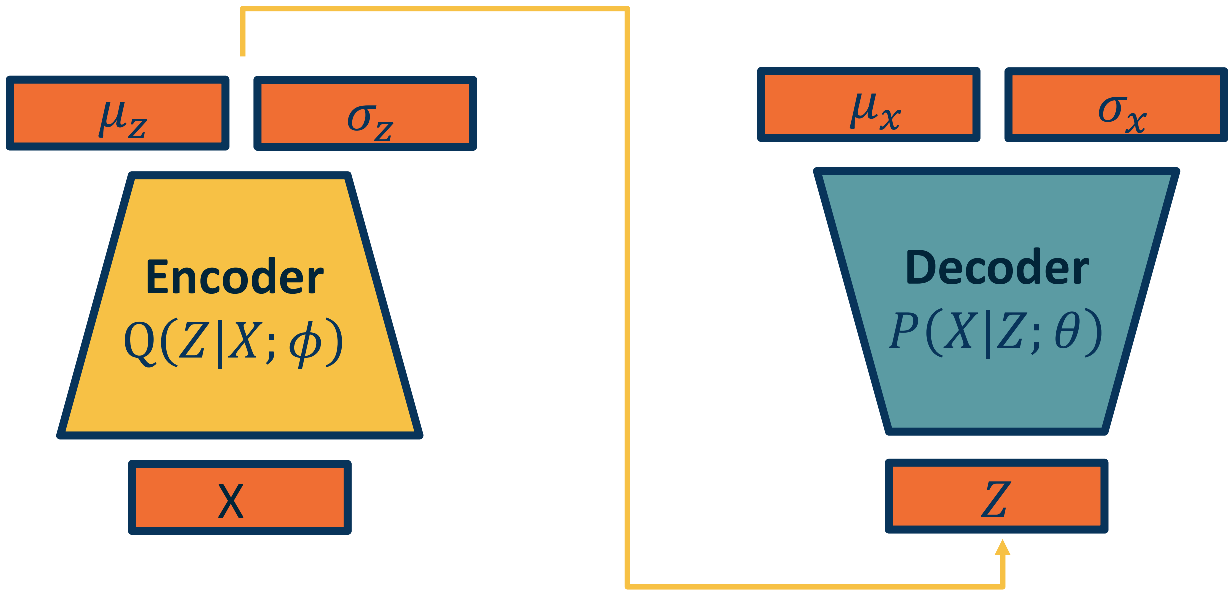
- We can combine the probabilistic view, sampling, autoencoders, and approximate optimization



- Given an image, estimate Z
- Again, output *parameters of a distribution*

Variational Autoencoder: Encoder

- We can tie the encoder and decoder together into a probabilistic autoencoder
  - Given data (X), estimate μ_z, σ_z and sample from $N(\mu_z, \sigma_z)$
  - Given Z, estimate μ_x, σ_x and sample from $N(\mu_x, \sigma_x)$



Putting Them Together

Georgia Tech

- How can we optimize the parameters of the two networks?

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

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

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

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

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

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

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

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

Aside: KL Divergence (distance measure for distributions), always >= 0

$$KL(p||q) = H_c(p,q) - H(p) = \sum p(x)\log q(x) - \sum p(x) \log p(x)$$

Definition of Expectation

$$\mathbb{E}[f] = \mathbb{E}_{x\sim q}[f(x)] = \sum_{x\in\Omega} q(x)f(x)$$

$$KL(a||b) = E[\log\ a(x)] - E[log\ b(x)] = E[log\frac{a(x)}{b(x)}]$$

KL-Divergence

Georgia Tech

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

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

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

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

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

The expectation wrt. **z** (using encoder network) let us write nice KL terms

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

Georgia Tech

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

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

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

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

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

Decoder network gives $p_\theta(x|\mathbf{z})$, 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!

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

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

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

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

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

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

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

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

Variational lower bound ("ELBO")

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

Training: Maximize lower bound

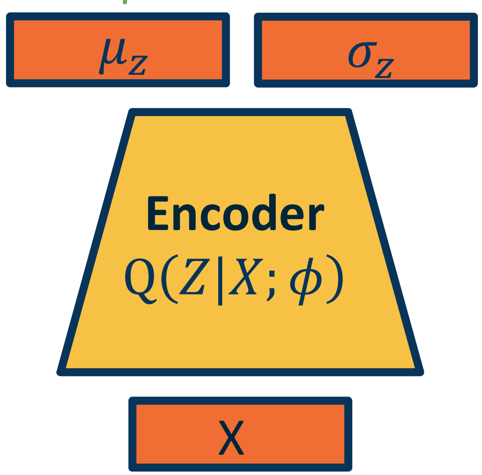
From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

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

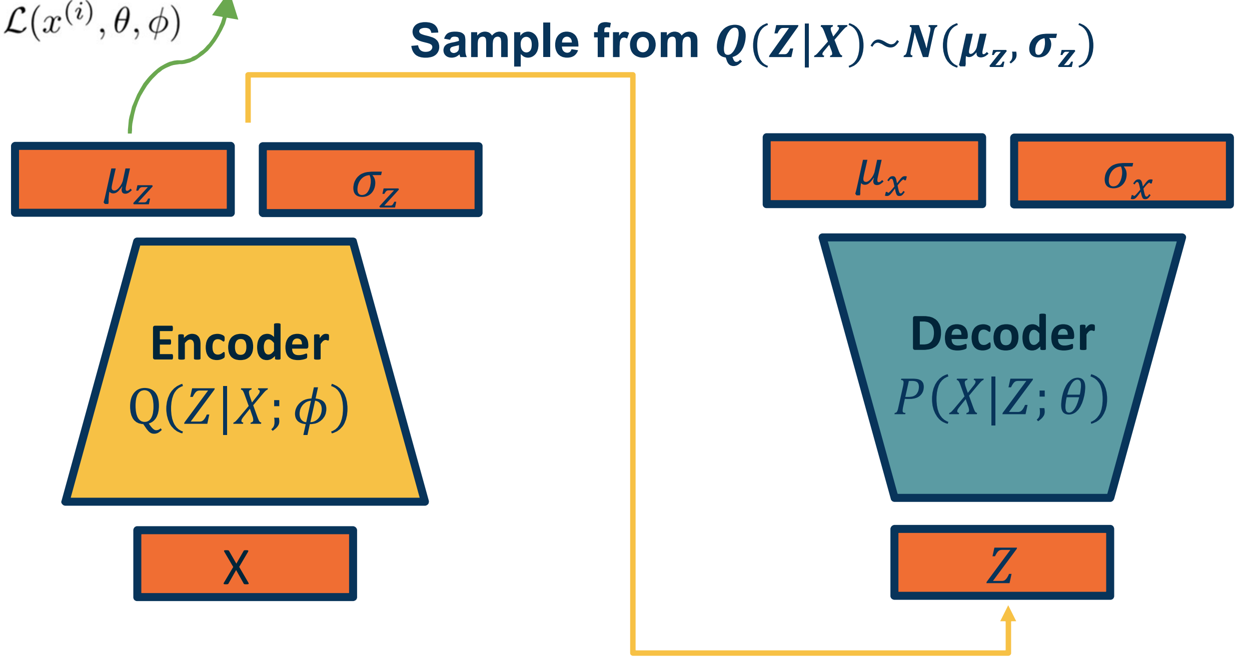


From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Forward and Backward Passes

Putting it all together: maximizing the likelihood lower bound

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



From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

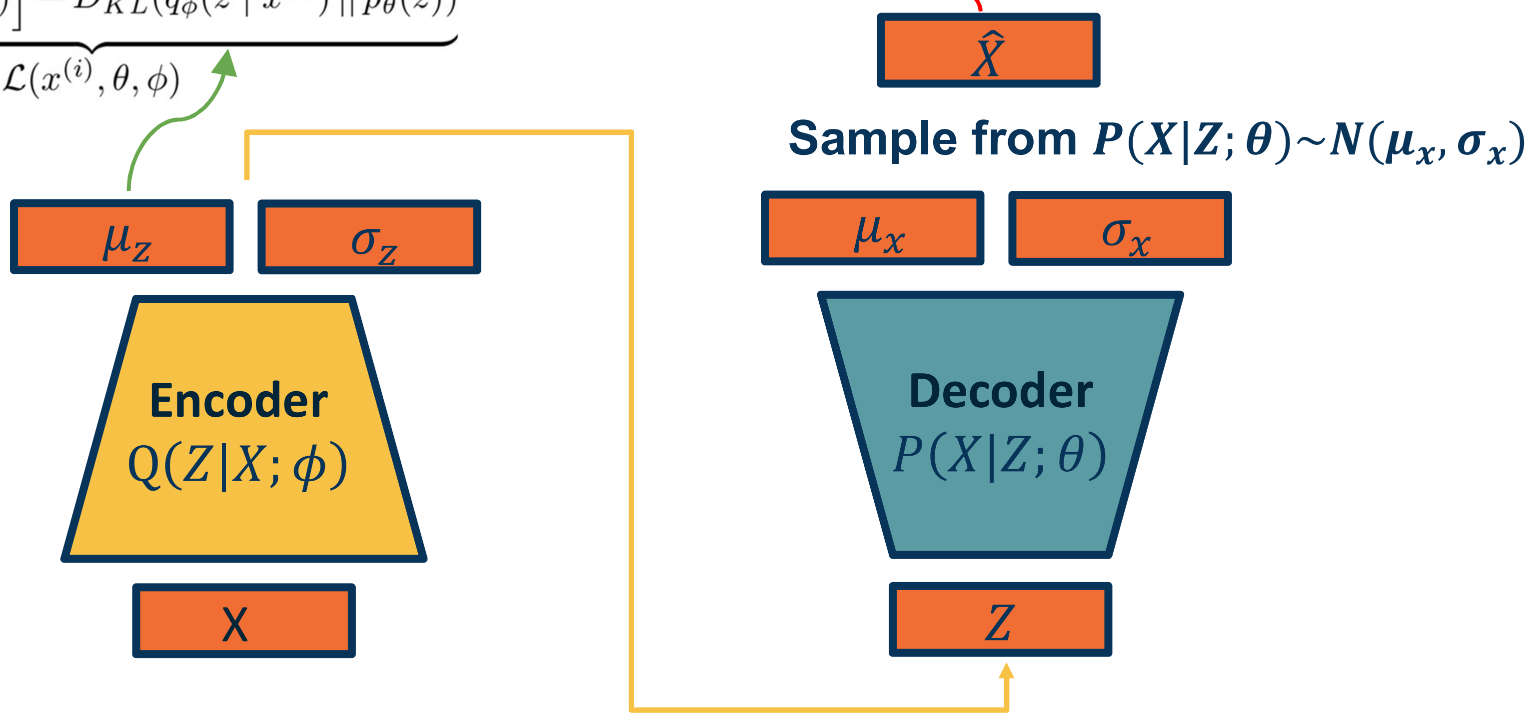
Forward and Backward Passes

Georgia Tech

Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed



From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

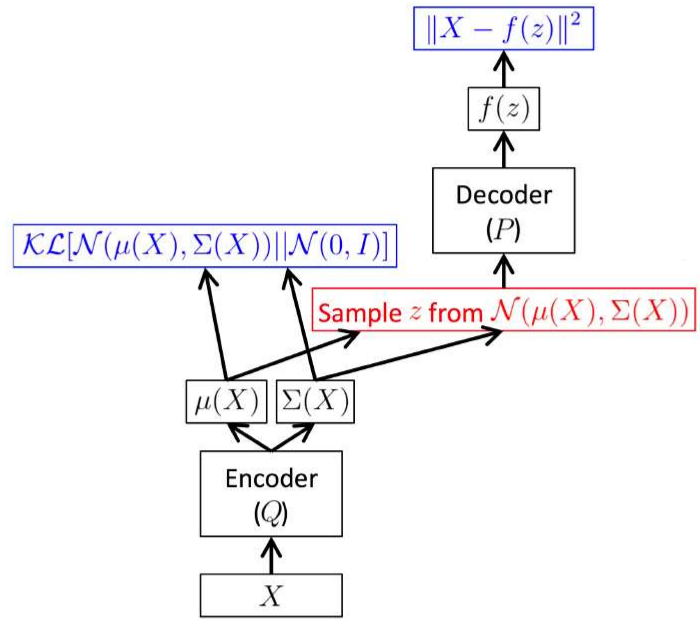
Forward and Backward Passes

Reparameterization Trick: Problem

- Problem with respect to the VLB: updating ϕ

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{z},\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]$$
$$= -D_{\text{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})) + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z}$$

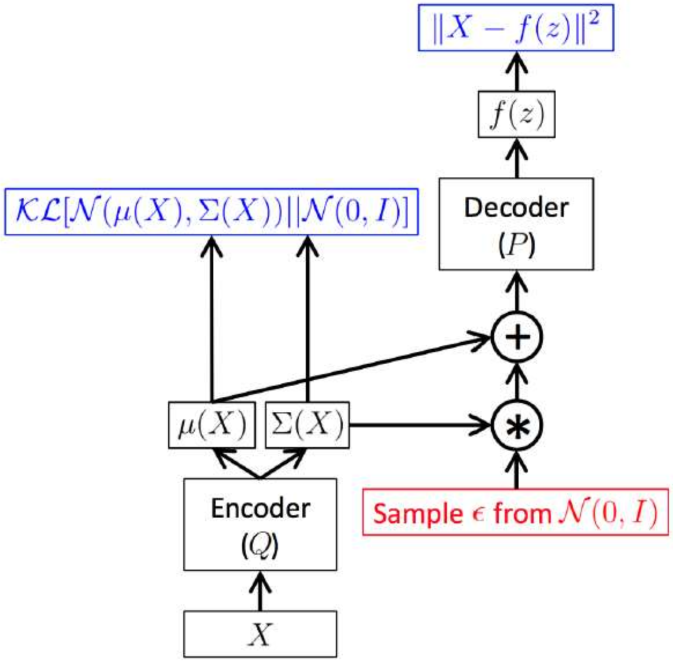
- $Z \sim Q(Z|X;\phi)$: need to differentiate through the sampling process w.r.t ϕ (encoder is probabilistic)



From: Tutorial on Variational Autoencoders
https://arxiv.org/abs/1606.05908

From: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/

- Solution: make the randomness independent of encoder output, making the encoder deterministic
- Gaussian distribution example:
  - Previously: encoder output = random variable $z \sim N(\mu, \sigma)$
  - Now encoder output = distribution parameter $[\mu, \sigma]$
  - $z = \mu + \epsilon * \sigma, \epsilon \sim N(0,1)$



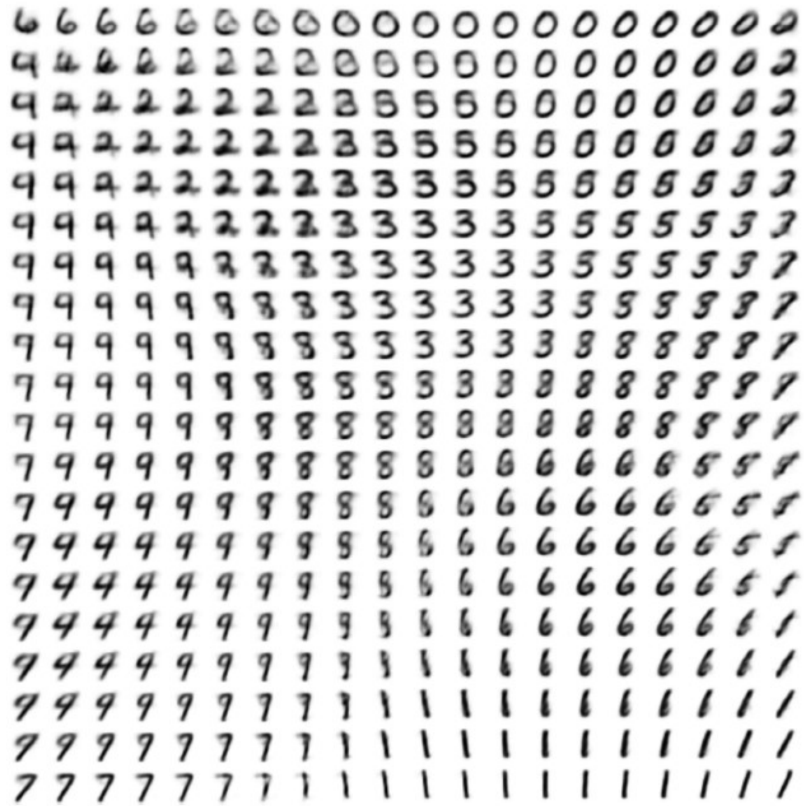
From: Tutorial on Variational Autoencoders
https://arxiv.org/abs/1606.05908

From: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/

Reparameterization Trick: Solution

z_1

z_2



Kingma & Welling, Auto-Encoding Variational Bayes

Interpretability of Latent Vector

Georgia Tech

- Variational Autoencoders (VAEs) provide a principled way to perform approximate maximum likelihood optimization
  - Requires some assumptions (e.g. Gaussian distributions)
- Samples are often not as competitive as other methods (GANs, diffusion)
- Latent features (learned in an unsupervised way!) often good for downstream tasks:
  - Example: World models for reinforcement learning (Ha et al., 2018)

Ha & Schmidhuber, World Models, 2018

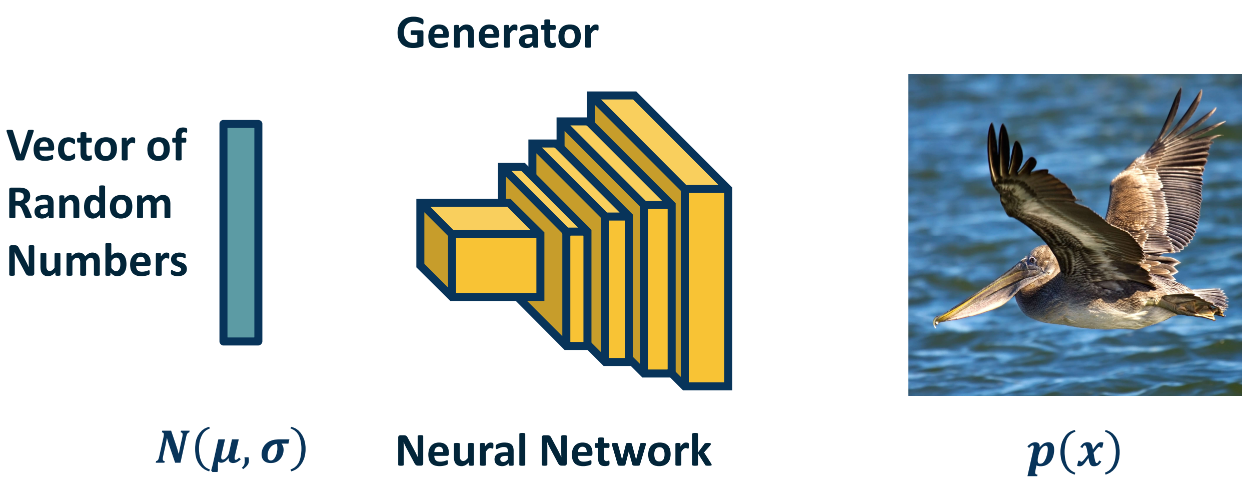
Summary

Georgia Tech

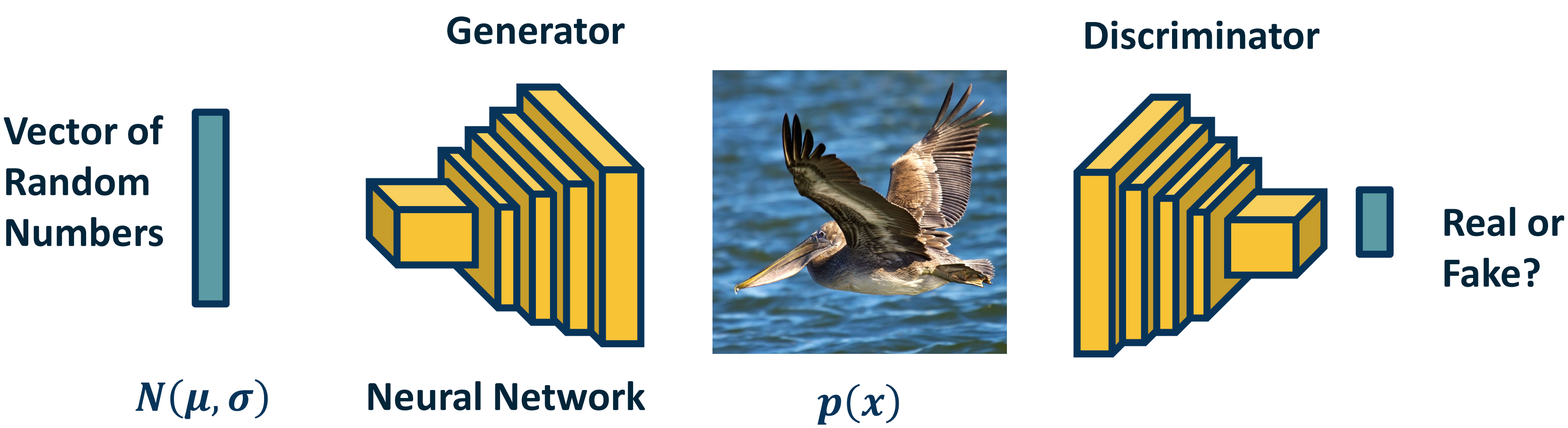
Generative Adversarial Networks (GANs)

Generating Images

- Input can be a vector with (independent) Gaussian random numbers
- We can use a CNN to generate images!

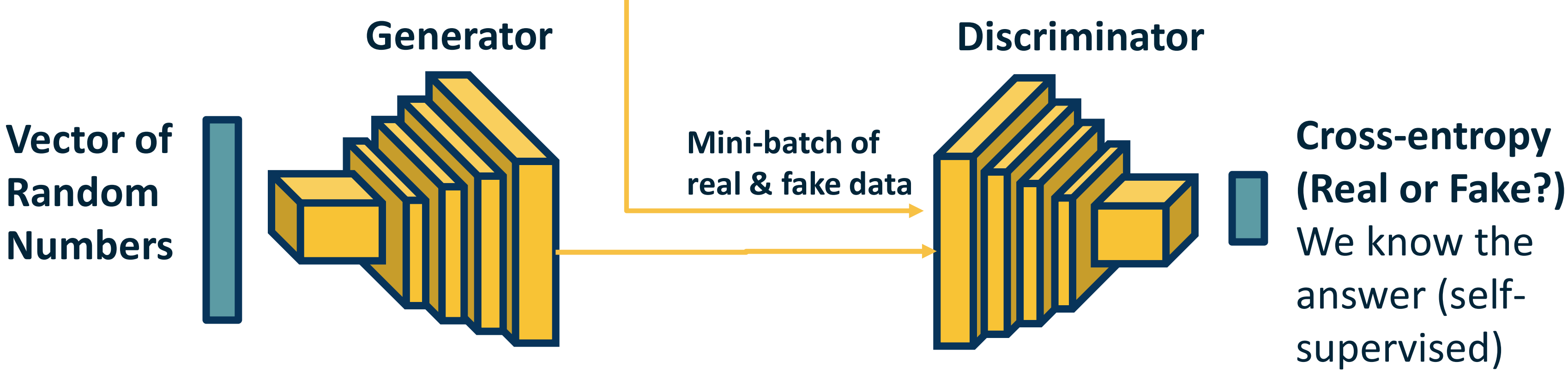


- **Goal:** We would like to generate *realistic* images. How can we drive the network to learn how to do this?
- **Idea:** Have *another* network try to distinguish a real image from a generated (fake) image
  - **Why?** Signal can be used to determine how well it's doing at generation

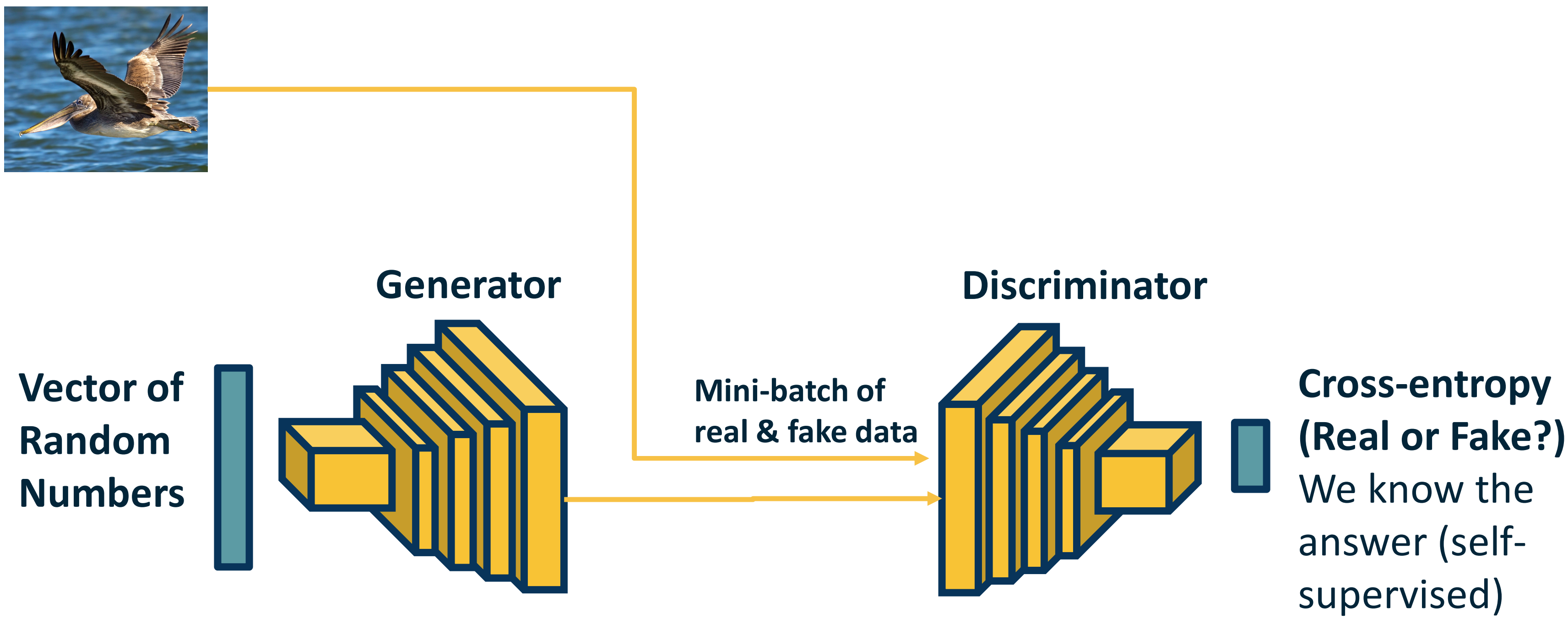


Adversarial Networks

- **Generator:** Update weights to improve realism of generated images
- **Discriminator:** Update weights to better discriminate



Question: What loss functions can we use (for each network)?



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

Generator Loss

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

Discriminator Loss

Brock et al., Large Scale GAN Training for High Fidelity Natural Image Synthesis

Example Generated Images - BigGAN

https://www.youtube.com/watch?v=PCBTZh41Ris

Video Generation

- Several ways to learn *generative* models via deep learning

- **PixelRNN/CNN:**
  - Simple tractable densities we can model via a NN and optimize
  - Slow generation – limited scaling to large complex images
- **Generative Adversarial Networks (GANs):**
  - Pro: Amazing results across many image modalities
  - Con: Unstable/difficult training process, computationally heavy for good results
  - Con: Limited success for discrete distributions (language)
  - Con: Hard to evaluate (implicit model)
- **Variational Autoencoders:**
  - Pro: Principled mathematical formulation
  - Pro: Results in disentangled latent representations
  - Con: Approximation inference, results in somewhat lower quality reconstructions

Ha & Schmidhuber, World Models, 2018

Overall Summary