Topics:

- Variational Autoencoders

CS 4803-DL / 7643-A
ZSOLT KIRA

- **A4 grades slated for this weekend**

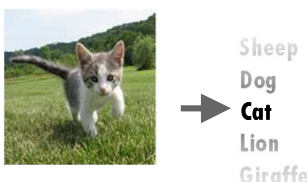
- **Projects!**
  - Due May 1st (May 3th with grace period)
  - Cannot extend due to grade deadlines!

- **CIOS**
  - Please make sure to fill out! Let us know about things you liked and didn’t like in comments so that we can keep or improve!
  - http://b.gatech.edu/cios

Introduction

Supervised Learning

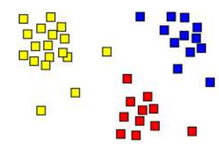
- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, $P(y|x)$
- e.g. classification



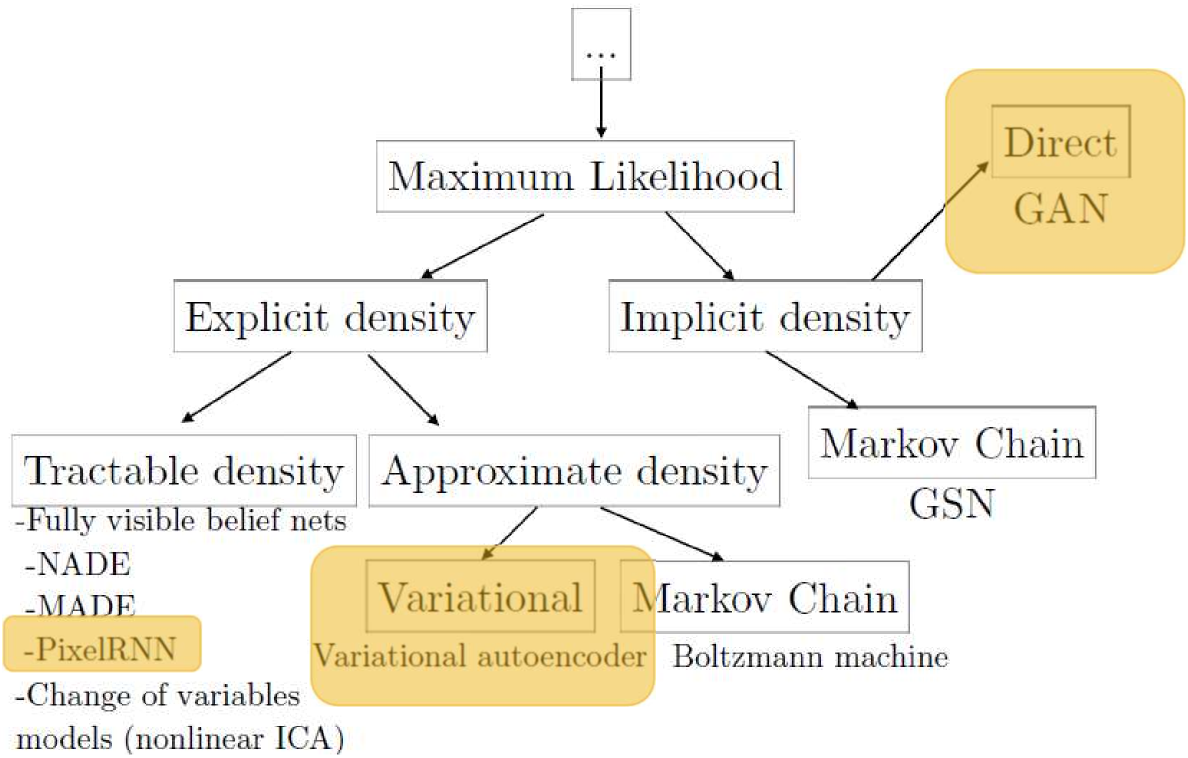
Less Labels

Unsupervised Learning

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

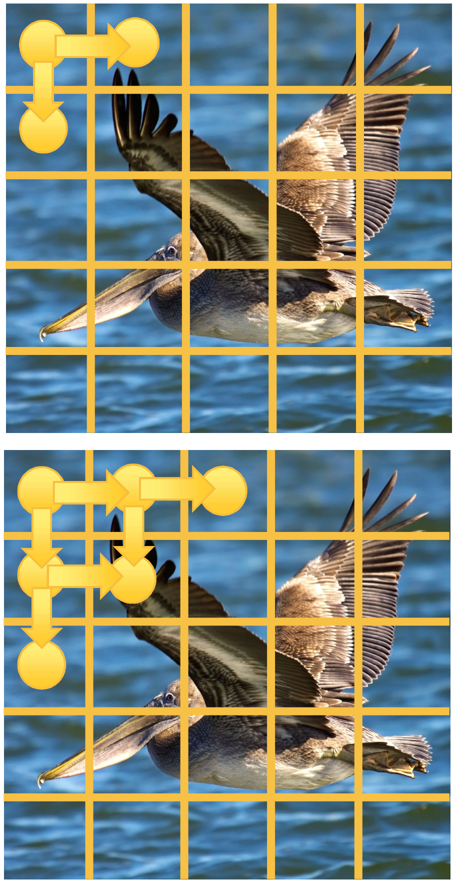


Spectrum of Low-Labeled Learning



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models



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

- Training:
  - We can train similar to language models: Teacher/student forcing
  - Maximum likelihood approach

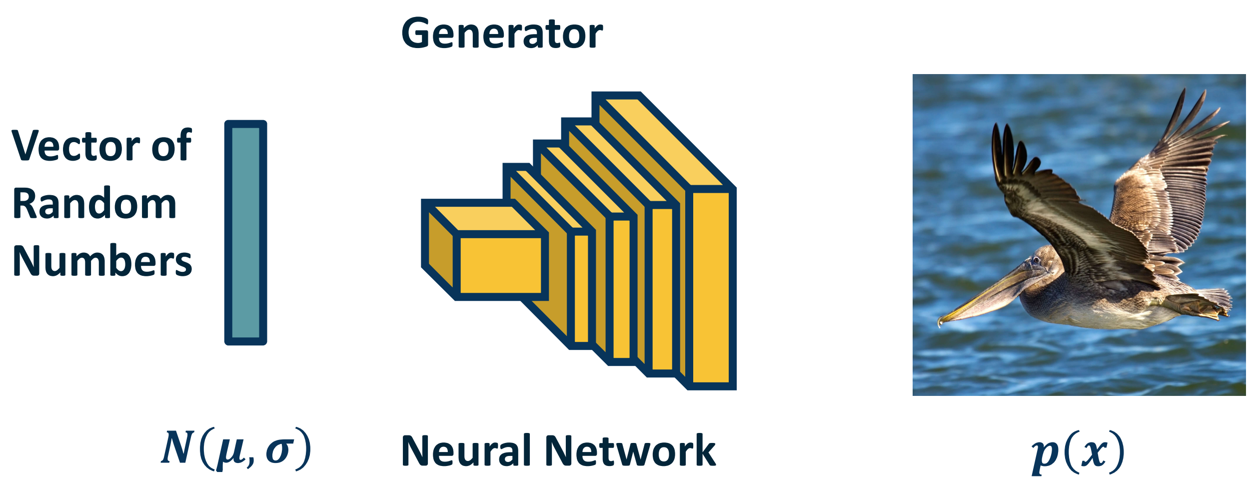
- Downsides:
  - Slow sequential generation process
  - Only considers few context pixels

Oord et al., Pixel Recurrent Neural Networks

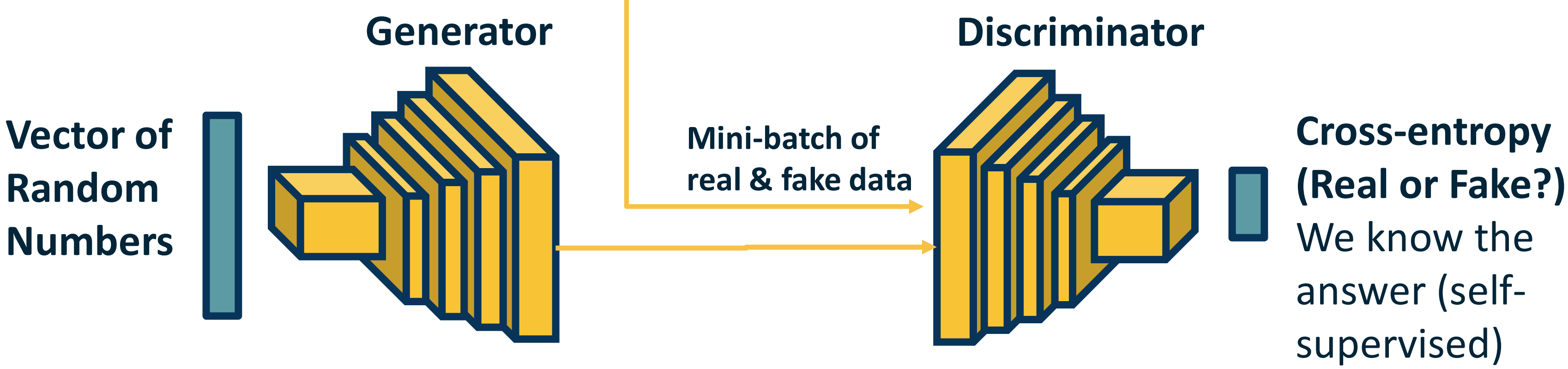
Factorized Models for Images

Generating Images

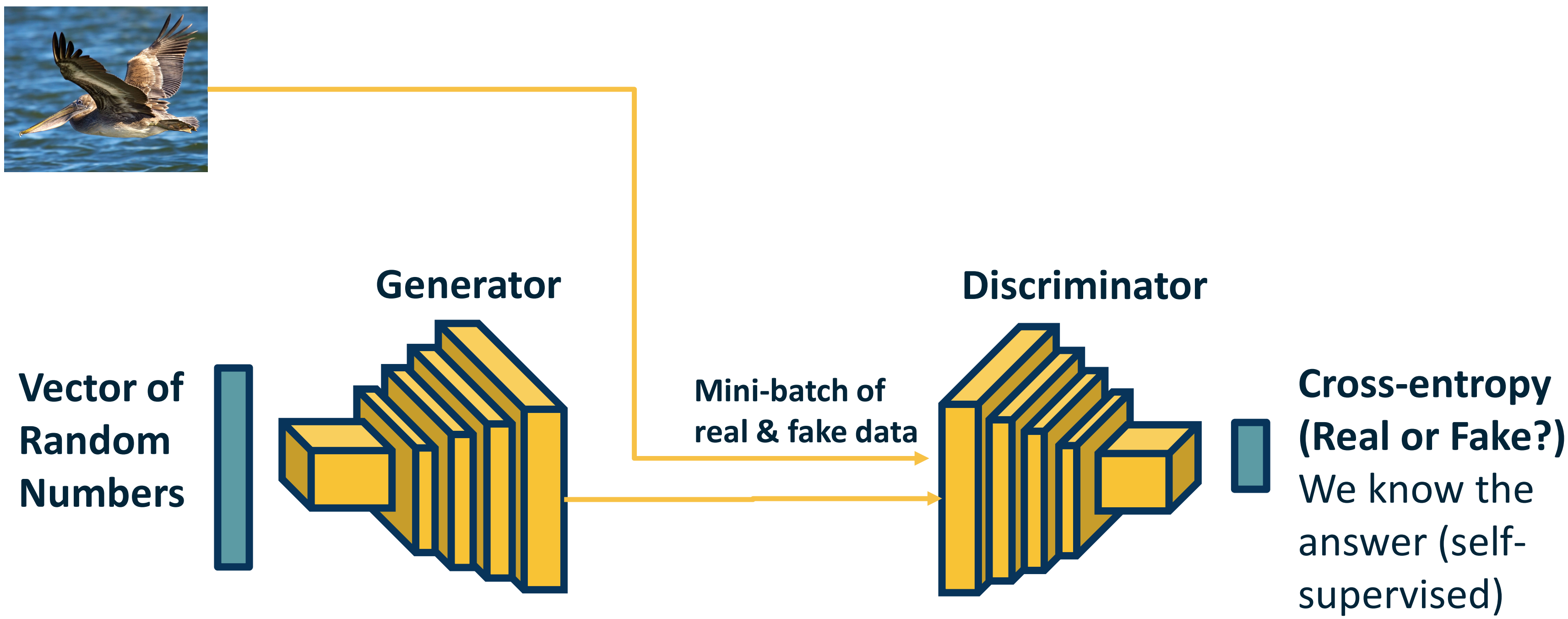
- Input can be a vector with (independent) Gaussian random numbers
- We can use a CNN to generate images!



- **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)?

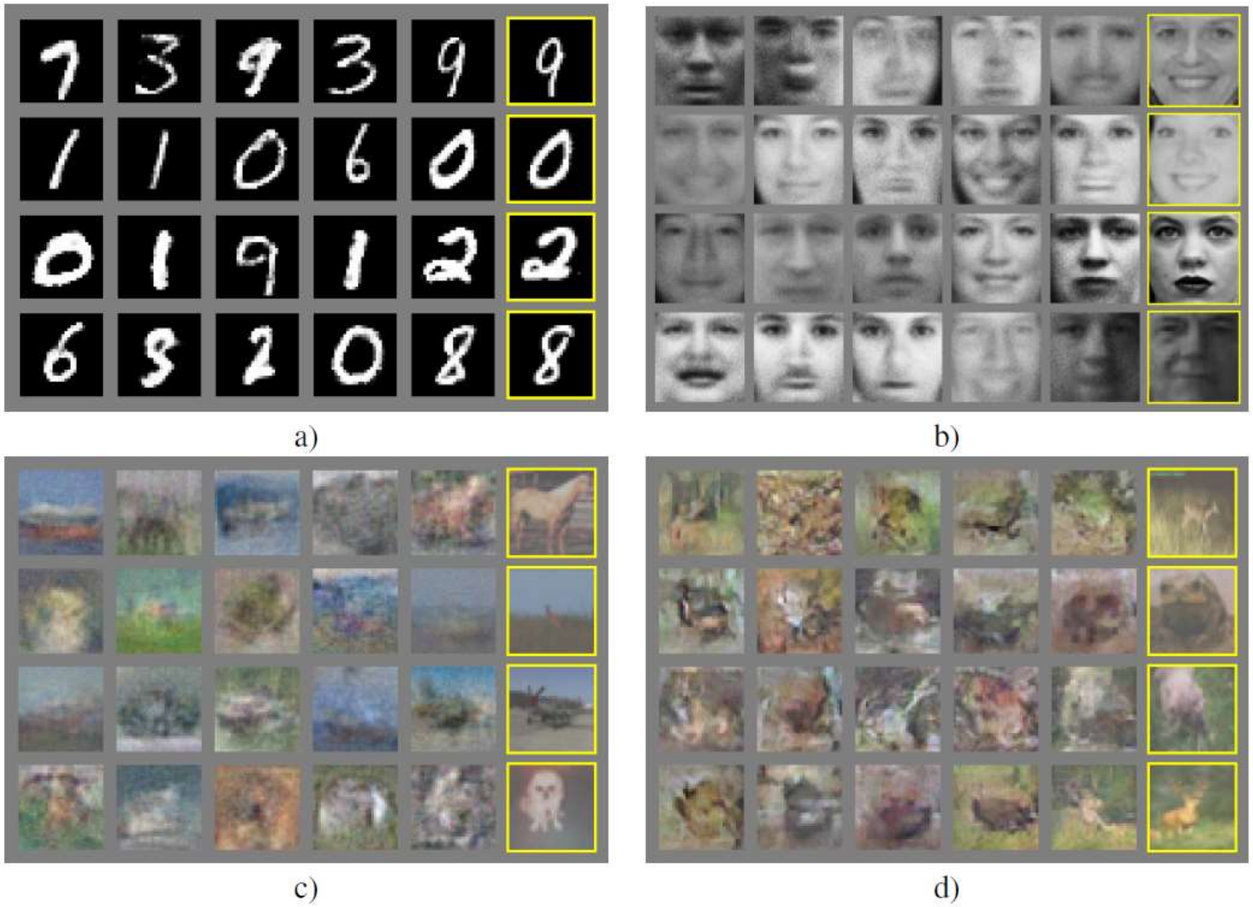


$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

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



- Low-resolution images but look decent!
- Last column are nearest neighbor matches in dataset

Early Results

Georgia Tech

- GANs are very difficult to train due to the mini-max objective

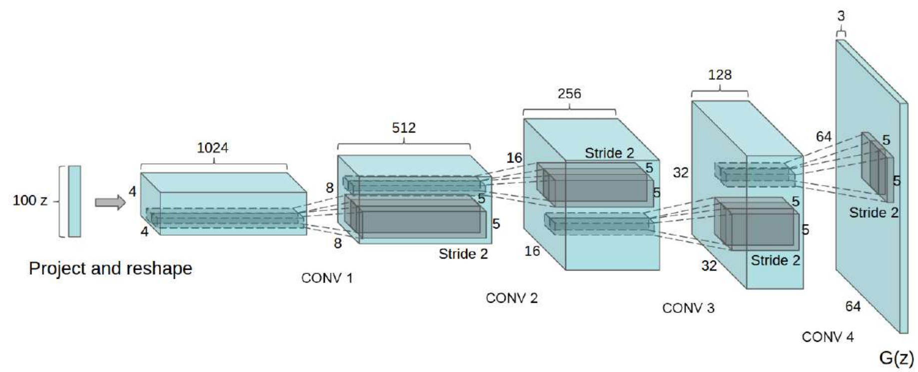
- Advancements include:
  - More stable architectures
  - Regularization methods to improve optimization
  - Progressive growing/training and scaling

Goodfellow, NeurIPS 2016 Generative Adversarial Nets

Difficulty in Training

Architecture guidelines for stable Deep Convolutional GANs

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



Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN

- Training GANs is difficult due to:
  - Minimax objective – For example, what if generator learns to memorize training data (no variety) or only generates part of the distribution?
  - Mode collapse – Capturing only some modes of distribution

- Several theoretically-motivated regularization methods
  - Simple example: Add noise to real samples!

$$\lambda \cdot \mathbb{E}_{x \sim P_{real}, \delta \sim N_d(0, cI)} \left[\|\nabla_{\mathbf{x}} D_\theta(x + \delta)\| - k \right]^2$$

Kodali et al., On Convergence and Stability of GANs (also known as How to Train your DRAGAN)

Regularization

Generative Adversarial Nets: Convolutional Architectures

Samples from the model look much better!

Radford et al, ICLR 2016

Generative Adversarial Nets: Convolutional Architectures

Interpolating between random points in latent space

Radford et al, ICLR 2016

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

Example Generated Images - BigGAN

(a) 128×128 (b) 256×256 (c) 512×512 (d)

Figure 4: Samples from our model with truncation threshold 0.5 (a-c) and an example of class leakage in a partially trained model (d).

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

Failure Examples - BigGAN

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

Video Generation

- A few other examples:
  - Deep nostalgia: https://www.myheritage.com/deep-nostalgia
  - High-resolution outputs: https://compvis.github.io/taming-transformers/

GANs

Don't work with an explicit density function
Take game-theoretic approach: learn to generate from training distribution through 2-player game

Pros:
- Beautiful, state-of-the-art samples!

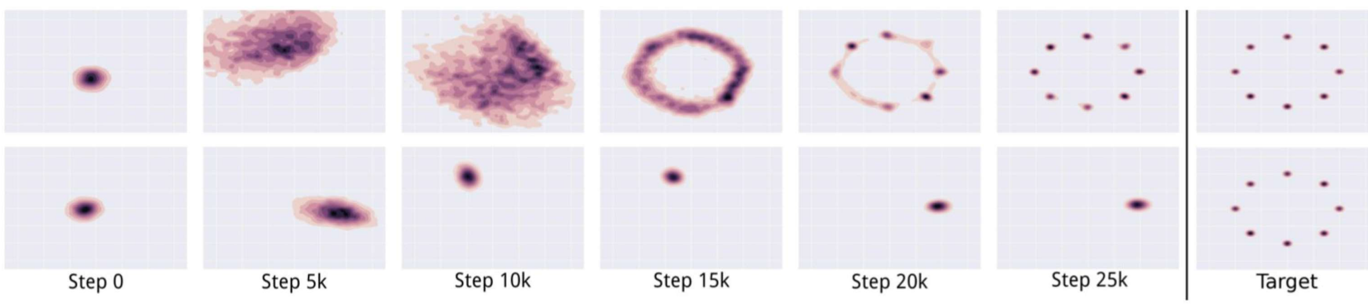
Cons:
- Trickier / more unstable to train
- Can't solve inference queries such as p(x), p(z|x)

Active areas of research:
- Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- Conditional GANs, GANs for all kinds of applications

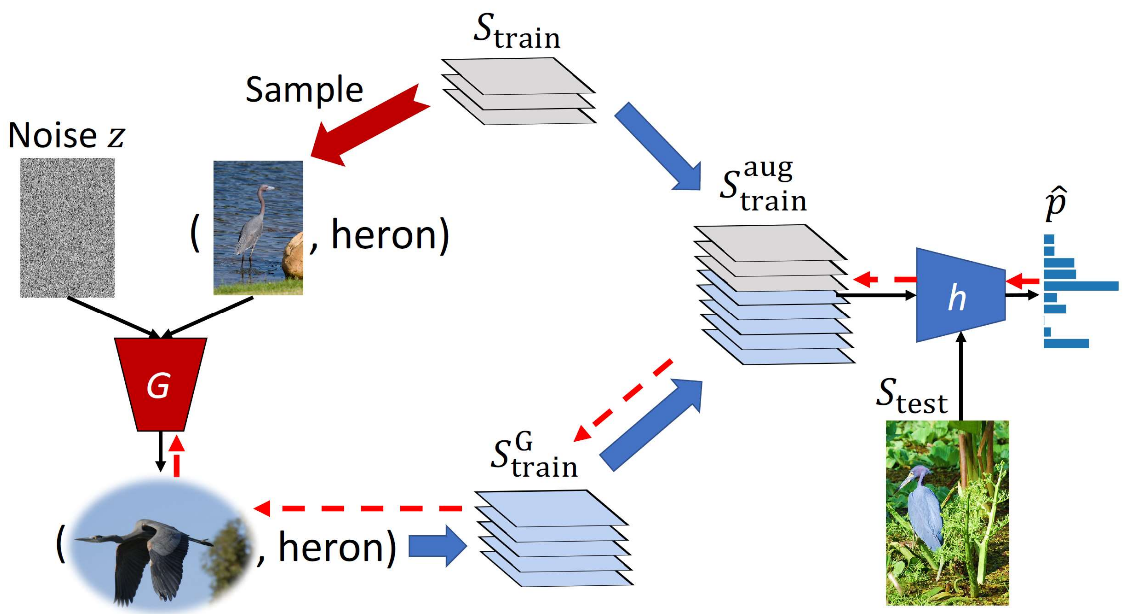
Mode Collapse

- Optimization of GANs is tricky
  - Not guaranteed to find Nash equilibrium

- Large number of methods to combat:
  - Use history of discriminators
  - Regularization
  - Different divergence measures

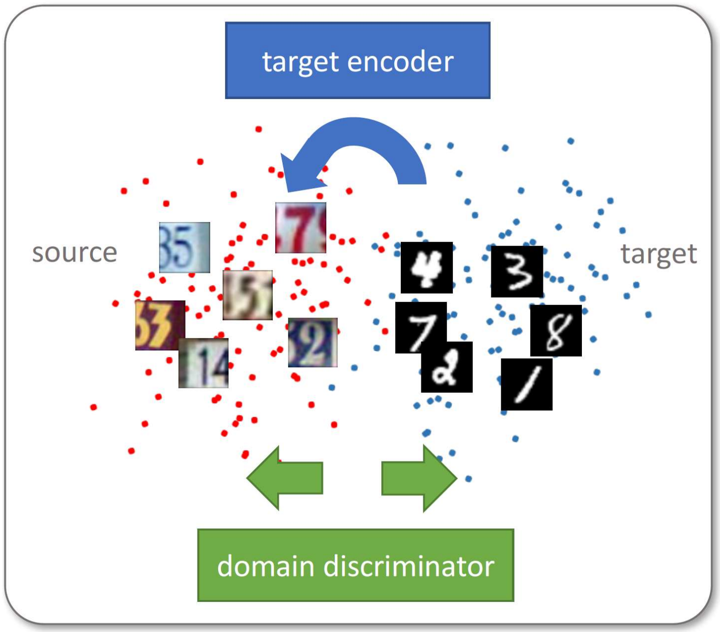


Application: Data Augmentation

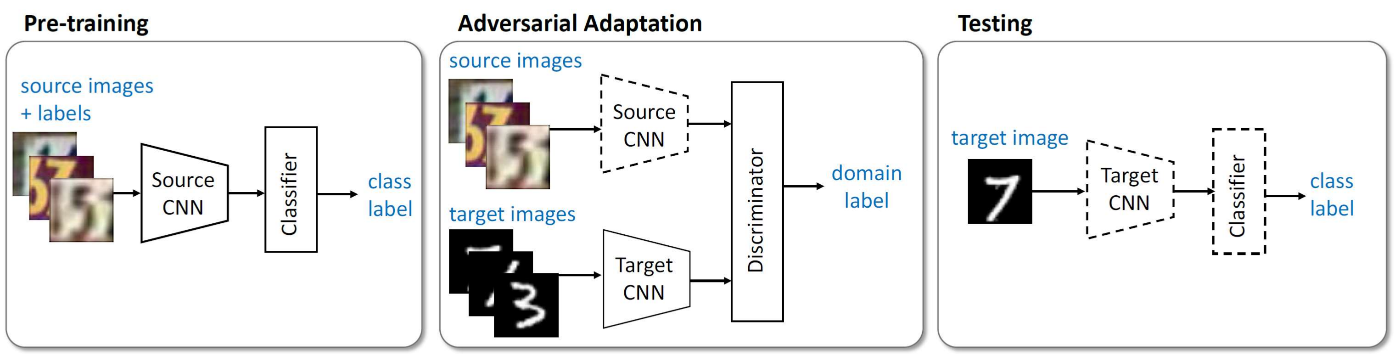


Application: Domain Adaptation

- **Idea:** Train a model on *source* data and adapt to *target* data using unlabeled examples from target



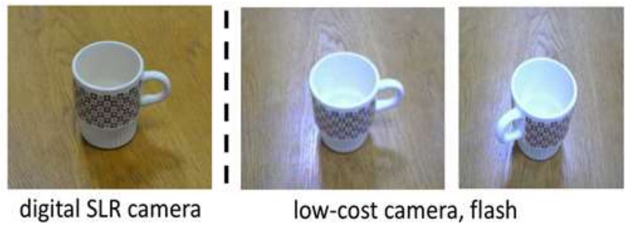
Approach

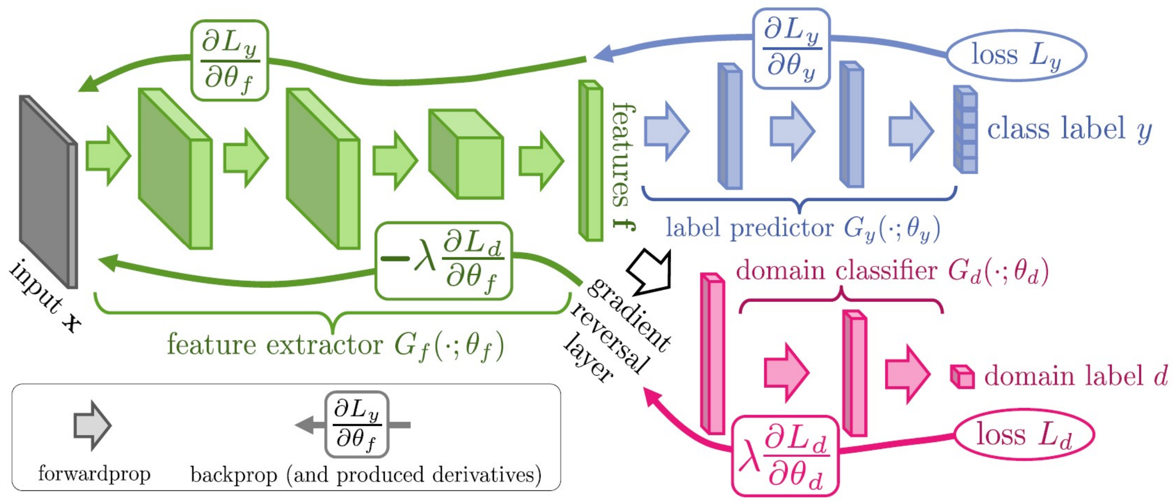


Method	MNIST → USPS	USPS → MNIST	SVHN → MNIST
Source only	0.752 ± 0.016	0.571 ± 0.017	0.601 ± 0.011
Gradient reversal	0.771 ± 0.018	0.730 ± 0.020	0.739 [16]
Domain confusion	0.791 ± 0.005	0.665 ± 0.033	0.681 ± 0.003
CoGAN	0.912 ± 0.008	0.891 ± 0.008	did not converge
ADDA (Ours)	0.894 ± 0.002	0.901 ± 0.008	0.760 ± 0.018

Table 2: Experimental results on unsupervised adaptation among MNIST, USPS, and SVHN.

Aside: Other ways to Align





[Ganin et al. , JMLR 2016]

- Generative Adversarial Networks (GANs) can produce amazing images!

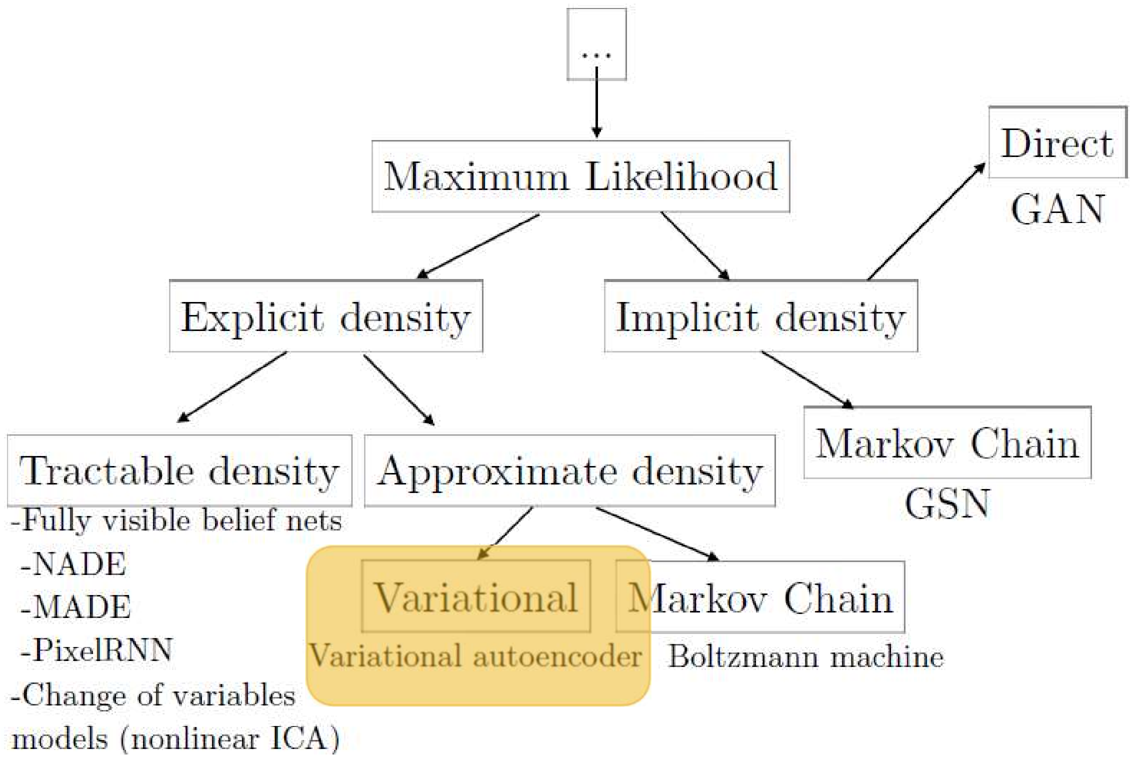
- Several drawbacks
  - High-fidelity generation heavy to train
  - Training can be unstable
  - No explicit model for distribution

- Larger number of extensions:
  - GANs conditioned on labels or other information
  - Adversarial losses for other applications

Summary

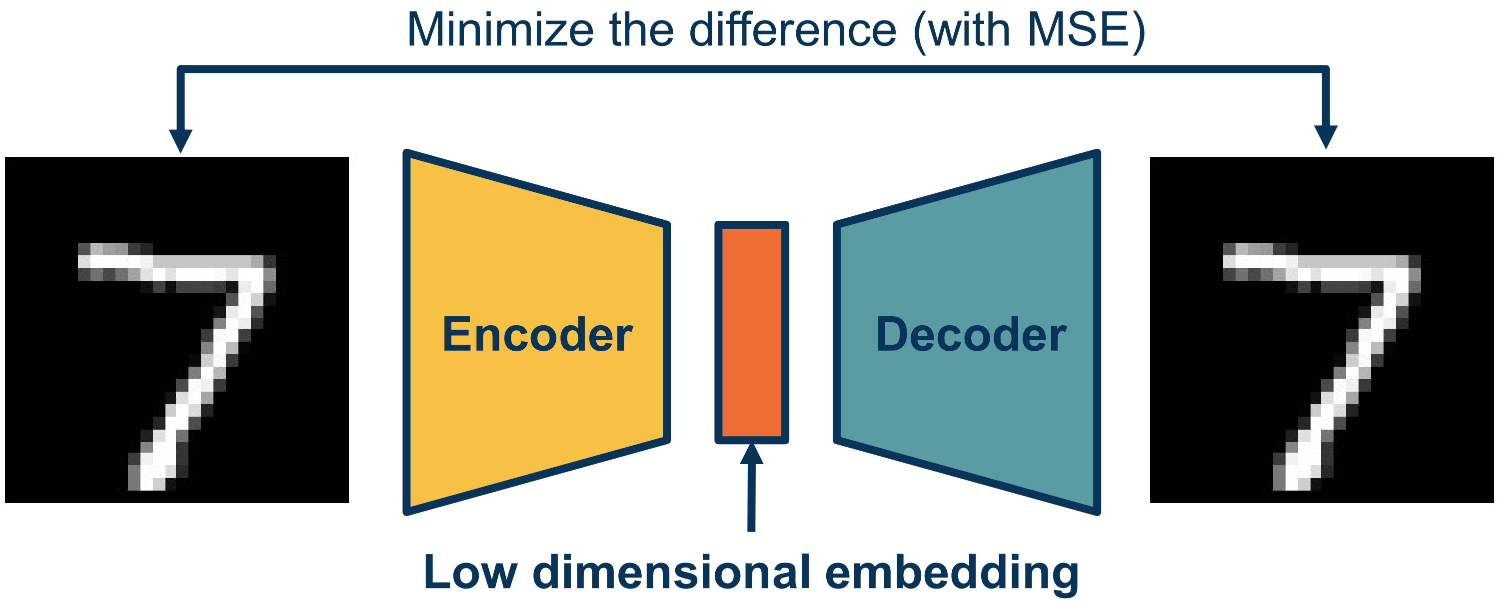
Variational Autoencoders (VAEs)

Georgia Tech



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

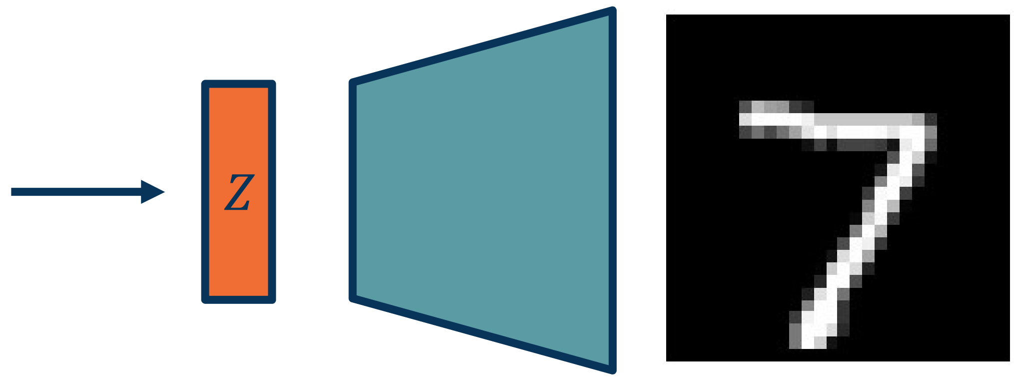


Linear layers with reduced dimension or Conv-2d layers with stride

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

Formalizing the Generative Model

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

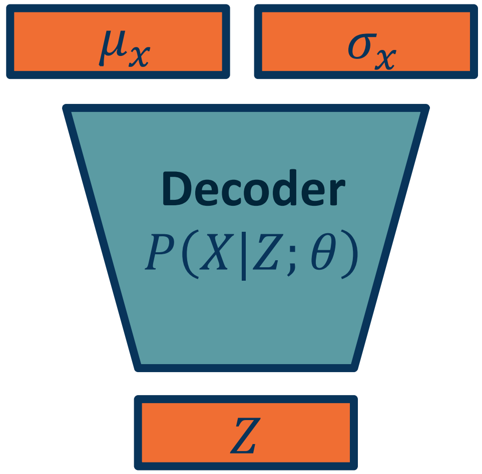


$$P(X) = \int P(X|Z;\theta)P(Z)dZ$$

- 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

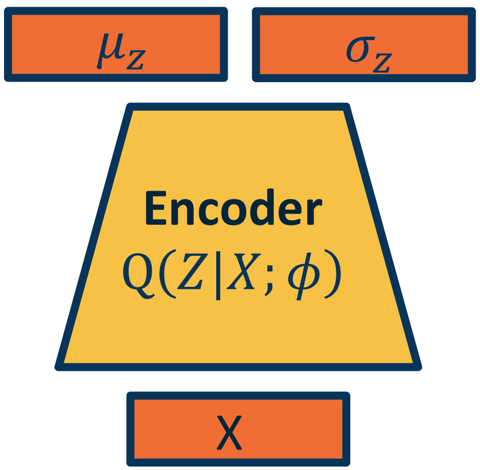
- We can combine the probabilistic view, sampling, autoencoders, and approximate optimization
- Just as before, sample Z from simpler distribution
- We can also output parameters of a probability distribution!
  - **Example**: μ, σ of Gaussian distribution
  - For multi-dimensional version output diagonal covariance
- How can we maximize $P(X) = \int P(X|Z;\theta)P(Z)dZ$



Variational Autoencoder: Decoder

Georgia Tech

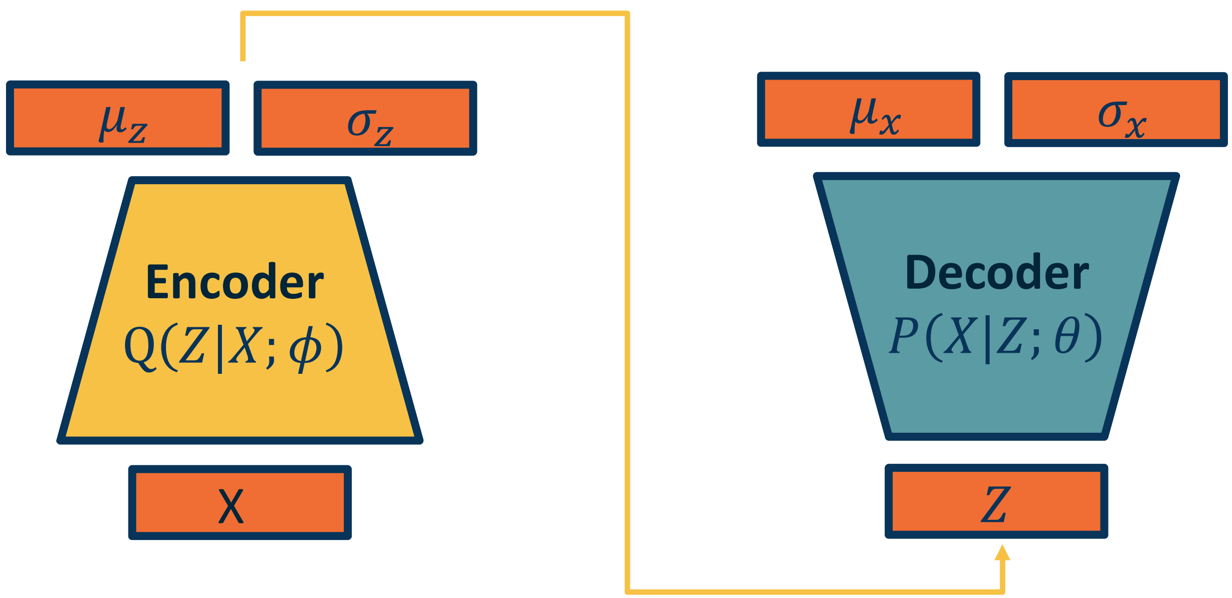
- We can combine the probabilistic view, sampling, autoencoders, and approximate optimization



- 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

- 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

Georgia Tech

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

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

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

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

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

Maximizing Likelihood

Georgia Tech

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

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

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

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

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

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

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

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

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

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

Maximizing Likelihood

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

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

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

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

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

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.

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

Maximizing Likelihood

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

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

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

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

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

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

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

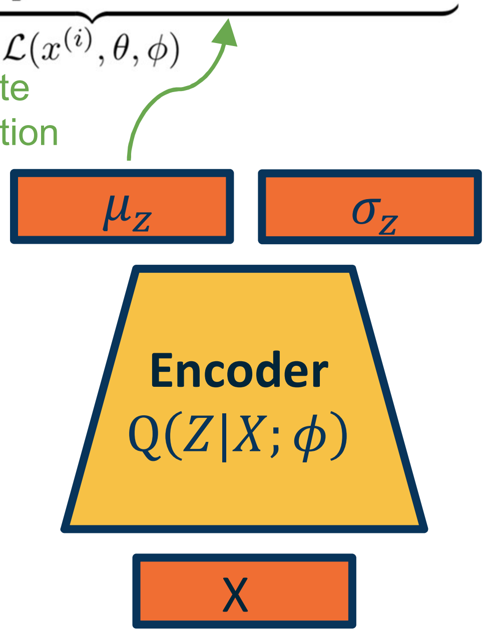
Maximizing Likelihood

Forward and Backward Passes

Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior

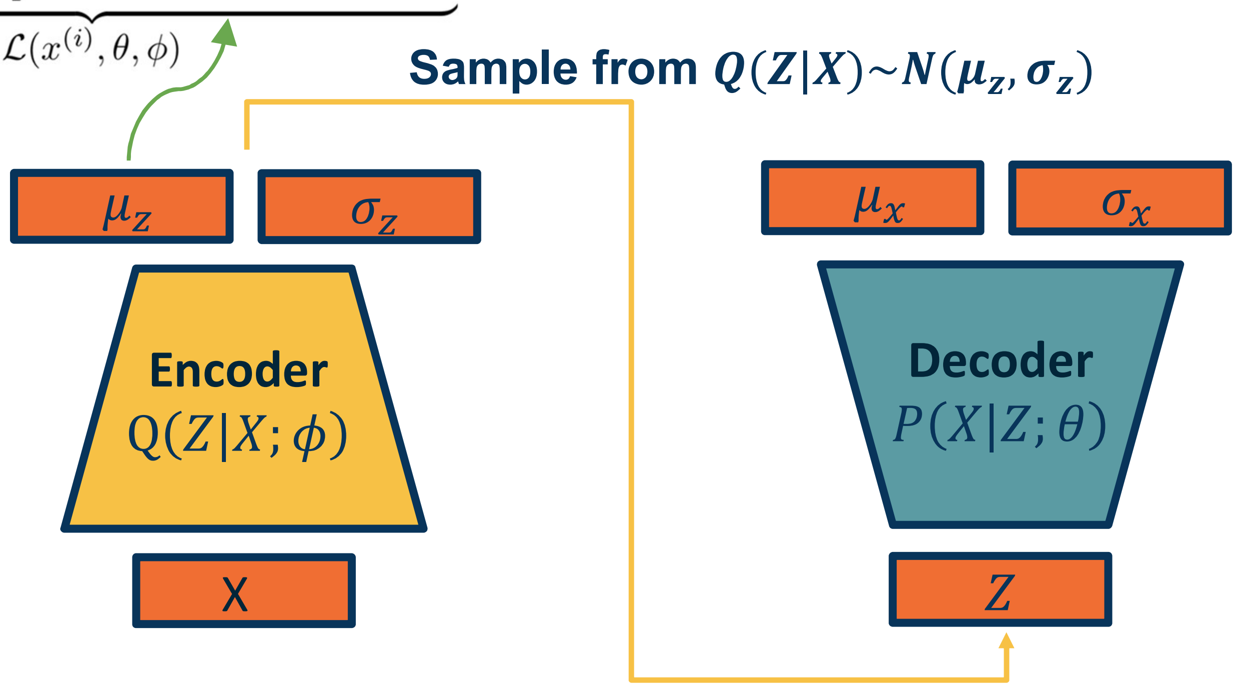


From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Putting it all together: maximizing the likelihood lower bound

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

Sample from $Q(Z|X) \sim N(\mu_z, \sigma_z)$



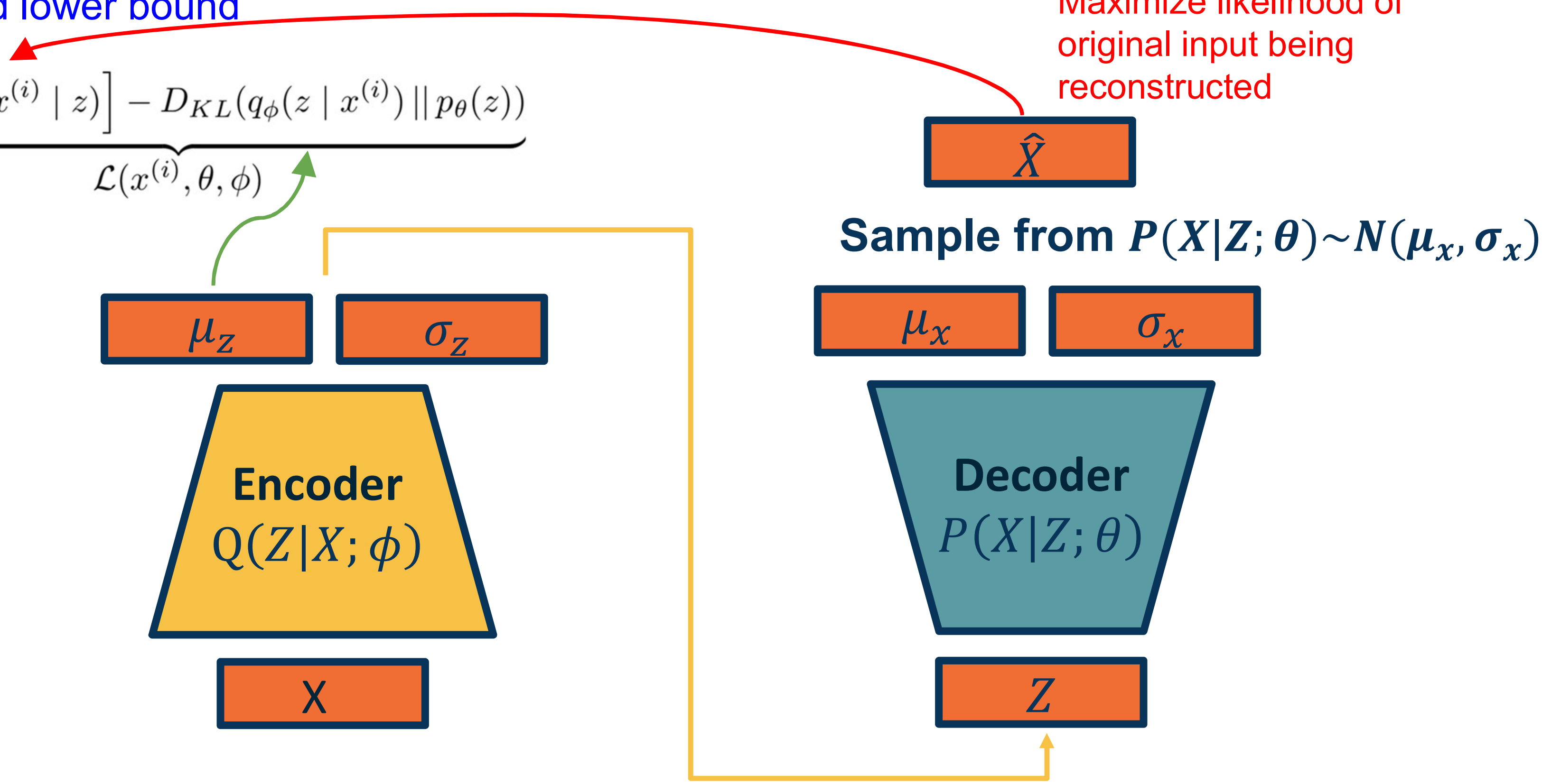
From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Forward and Backward Passes

Georgia Tech

Putting it all together: maximizing the likelihood lower bound

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



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

Forward and Backward Passes

Georgia Tech

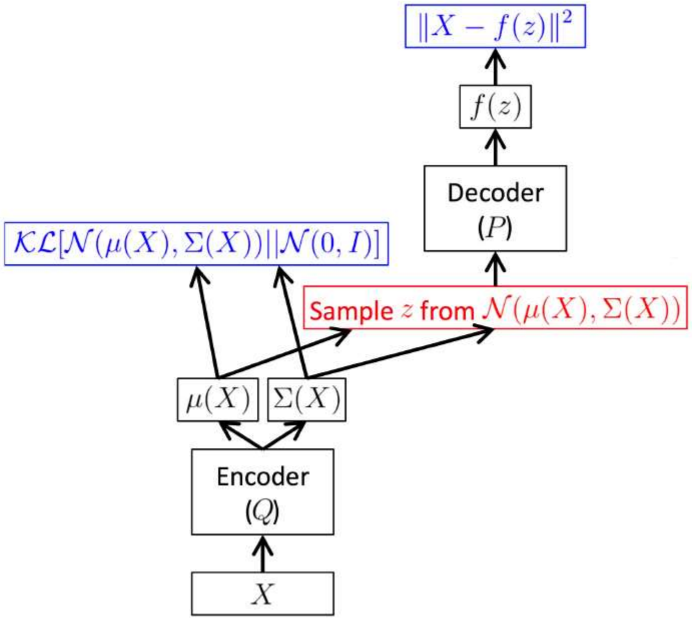
Reparameterization Trick: Problem

- Problem with respect to the VLB: updating ϕ

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

$$= -D_{\text{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})) + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z}$$

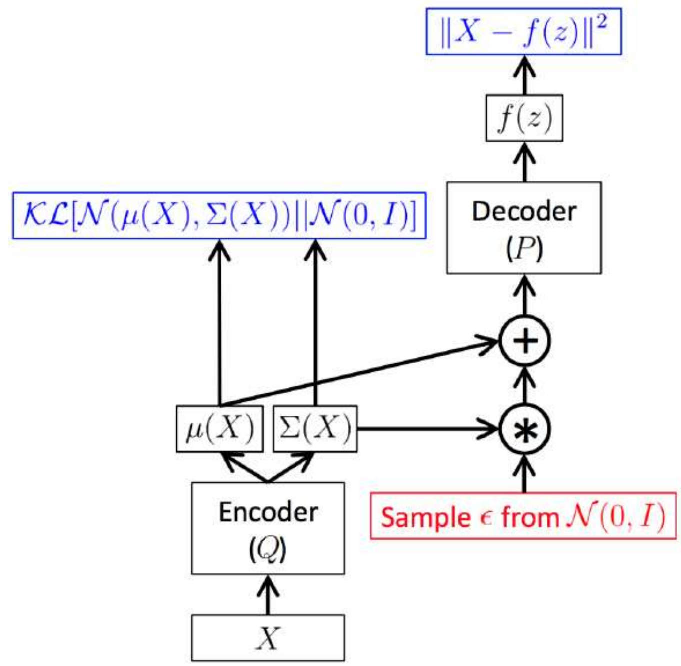
- $Z \sim Q(Z|X;\phi)$: need to differentiate through the sampling process w.r.t ϕ (encoder is probabilistic)



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

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

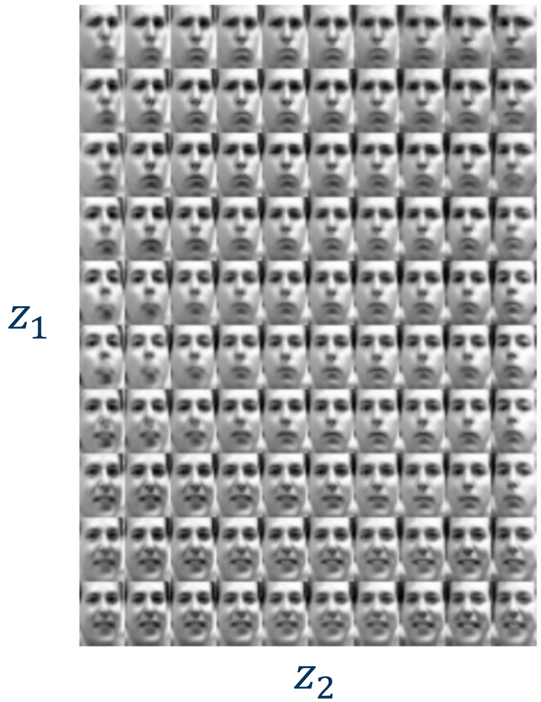
- Solution: make the randomness independent of encoder output, making the encoder deterministic
- Gaussian distribution example:
  - Previously: encoder output = random variable $z \sim N(\mu, \sigma)$
  - Now encoder output = distribution parameter $[\mu, \sigma]$
  - $z = \mu + \epsilon * \sigma, \epsilon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

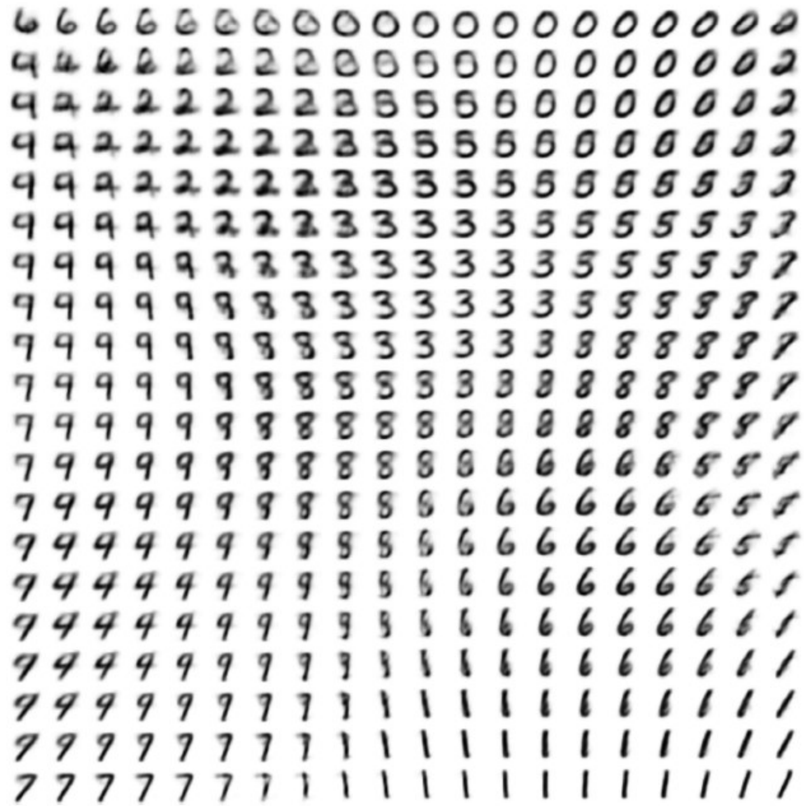


From: Tutorial on Variational Autoencoders
https://arxiv.org/abs/1606.05908

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

Reparameterization Trick: Solution





Kingma & Welling, Auto-Encoding Variational Bayes

Interpretability of Latent Vector

- 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 GANs
- 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

- 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