Topics:

- Variational Autoencoders

CS 4803-DL / 7643-A
ZSOLT KIRA

- **A4 due April 4th (grace until 6th)**

- **Projects!**
  - Make sure to contribute equally with your teammates!!!
  - We will have optional team peer review, and reduce scores if necessary

- **Rest of the semester:**
  - Open to topic suggestions for 04/17
  - Otherwise will cover VLMs

W12: Mar 27	Variational Autoencoders (VAEs)	• Sutton & Bartow Chapter 1 • Survey paper on Deep RL • MDP Notes (courtesy Byron Boots)
W12: Mar 29	Large Language Models (William Held)	• Notes on Q-learning (courtesy Byron Boots)
W13: Apr 3	RL background. PS4/HW4 due Apr 2nd (grace period Apr 4th)	• Policy iteration notes (courtesy Byron Boots) • Policy gradient notes (courtesy Byron Boots)
W13: Apr 5	RL: RL Part 2 - Q-Learning, DQN, Policy Gradient.	
W14: Apr 10	RL: Policy Gradients, REINFORCE, Actor-Critic.	
W14: Apr 12	Visualization and Interpretability	• Understanding Neural Networks Through Deep Visualization • Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization
W15: Apr 17		
W15: Apr 19	Final Project Due April 29 11:59pm (grace period May 1st)	

Back to Generative Models

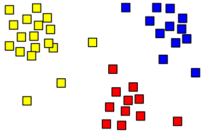
Supervised Learning

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

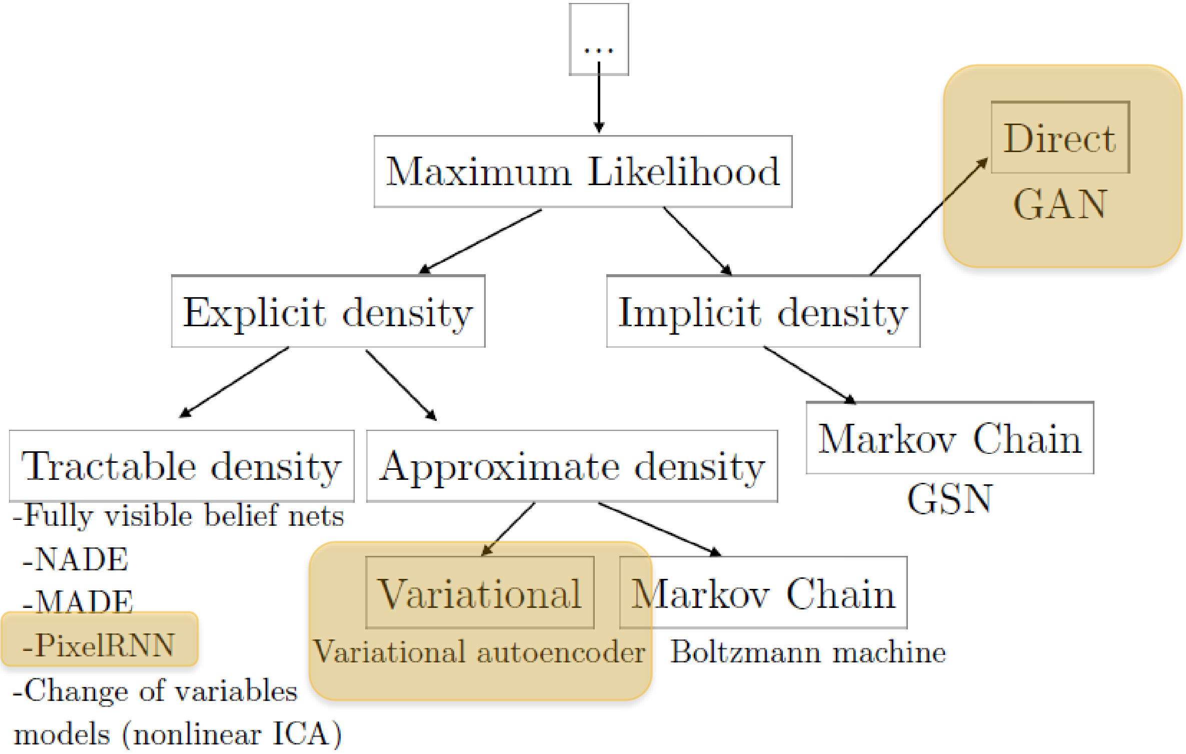
Less Labels

Unsupervised Learning

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



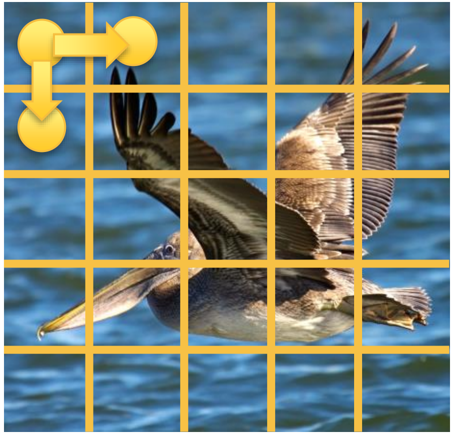
Spectrum of Low-Labeled Learning

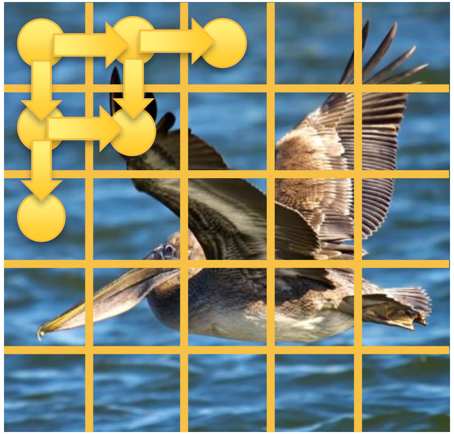


Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

Georgia Tech





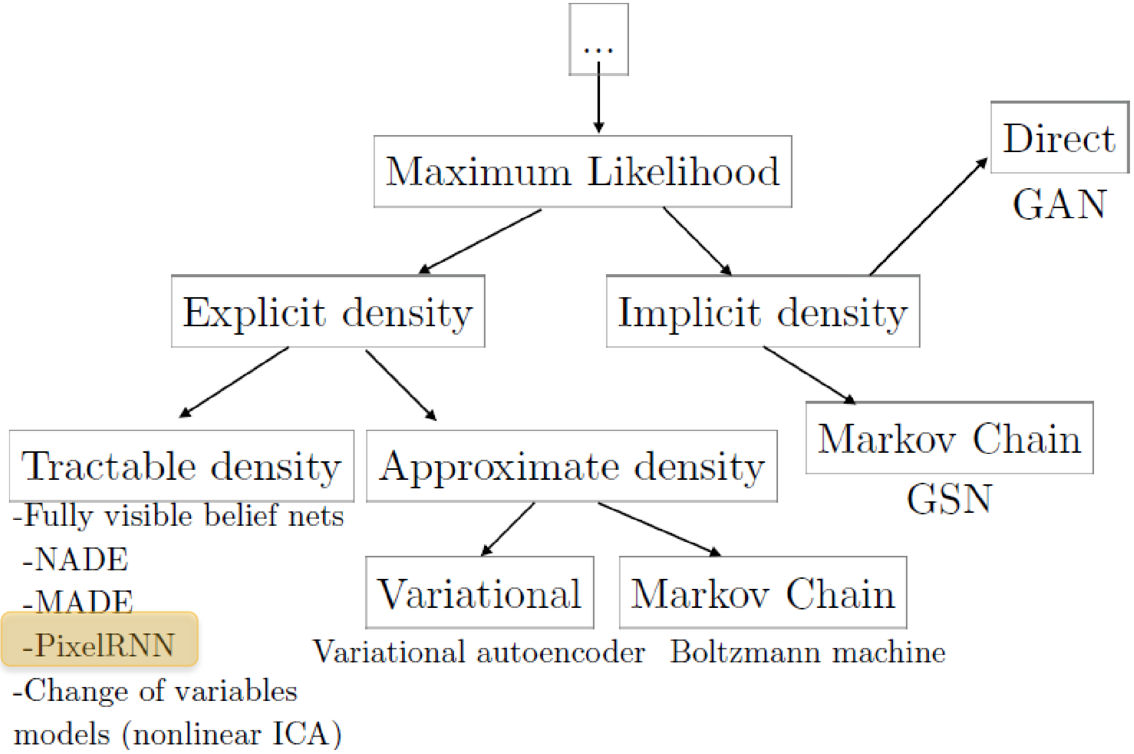
$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1)\prod_{i=1}^{n^2} p(x_i|x_1, \dots, x_{i-1})$$

- 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

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!

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

- 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

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

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

$$= \prod_i \mathrm{p}(w_i \mid w_{i-1}, \ldots, w_1)$$

next word

history

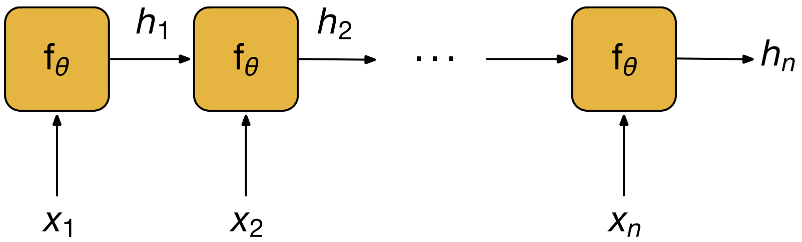
Georgia Tech

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

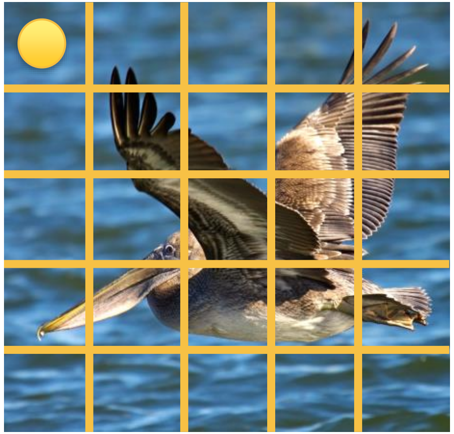
$$p(\boldsymbol{s}) = p(w_1, w_2, \ldots, w_n) = \prod_i p(w_i \mid w_{i-1}, \ldots, w_1)$$

next word — history

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



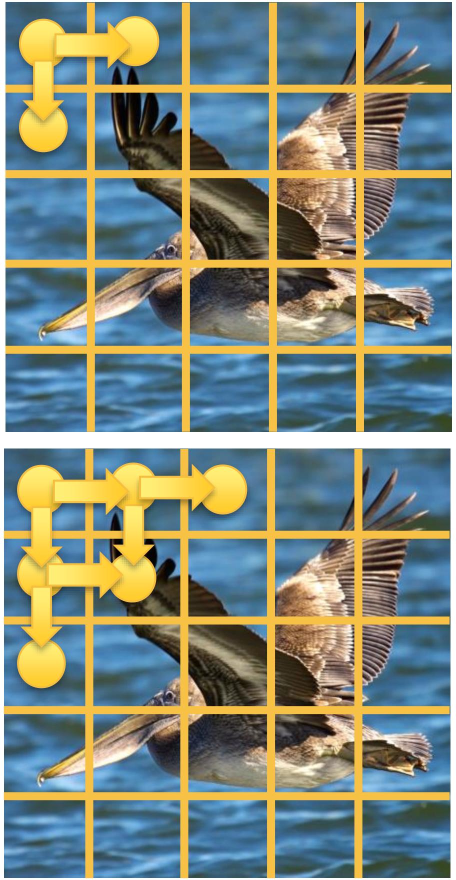
Georgia Tech



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

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

Oord et al., Pixel Recurrent Neural Networks

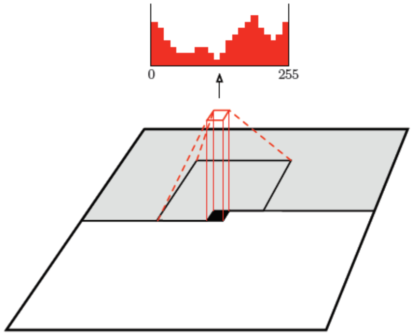


$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1)\prod_{i=1}^{n^2} p(x_i|x_1, \dots, x_{i-1})$$

- 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

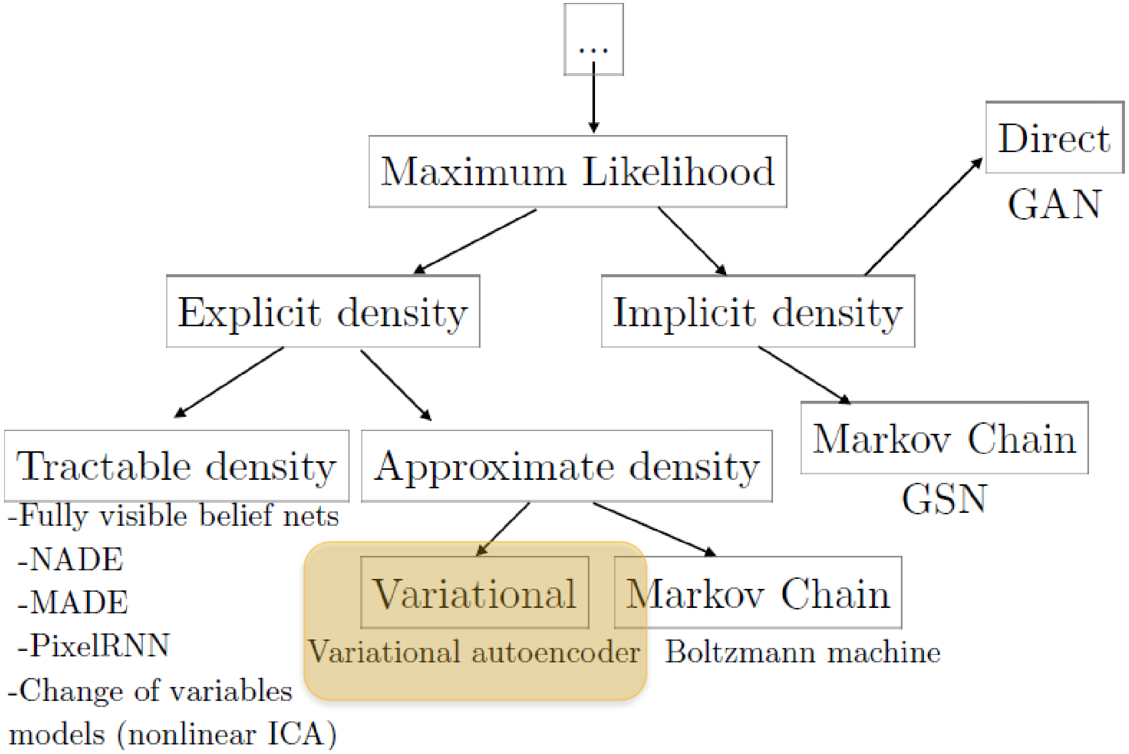
Oord et al., Conditional Image Generation with PixelCNN Decoders

Example Results: Image Completion (PixelRNN)

Oord et al., Conditional Image Generation with PixelCNN Decoders

Example Images (PixelCNN)

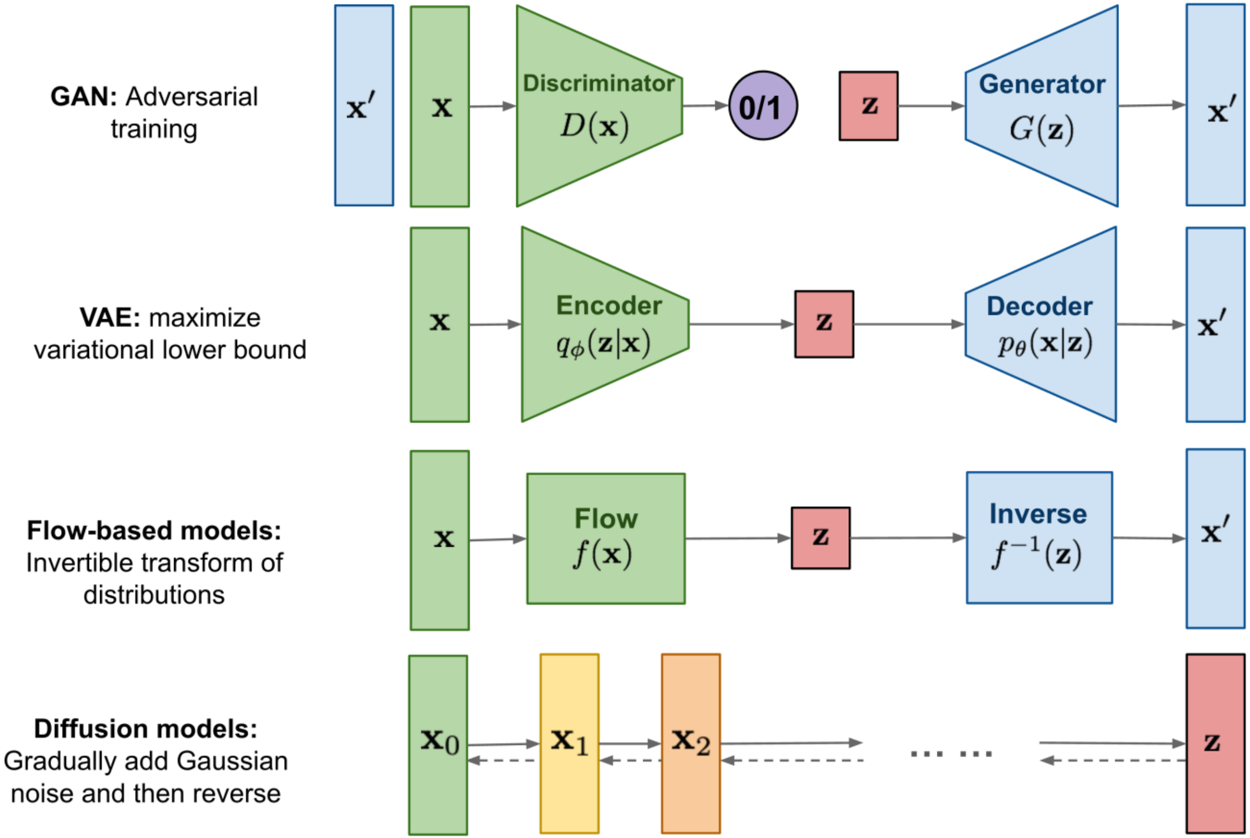
Variational Autoencoders (VAEs)

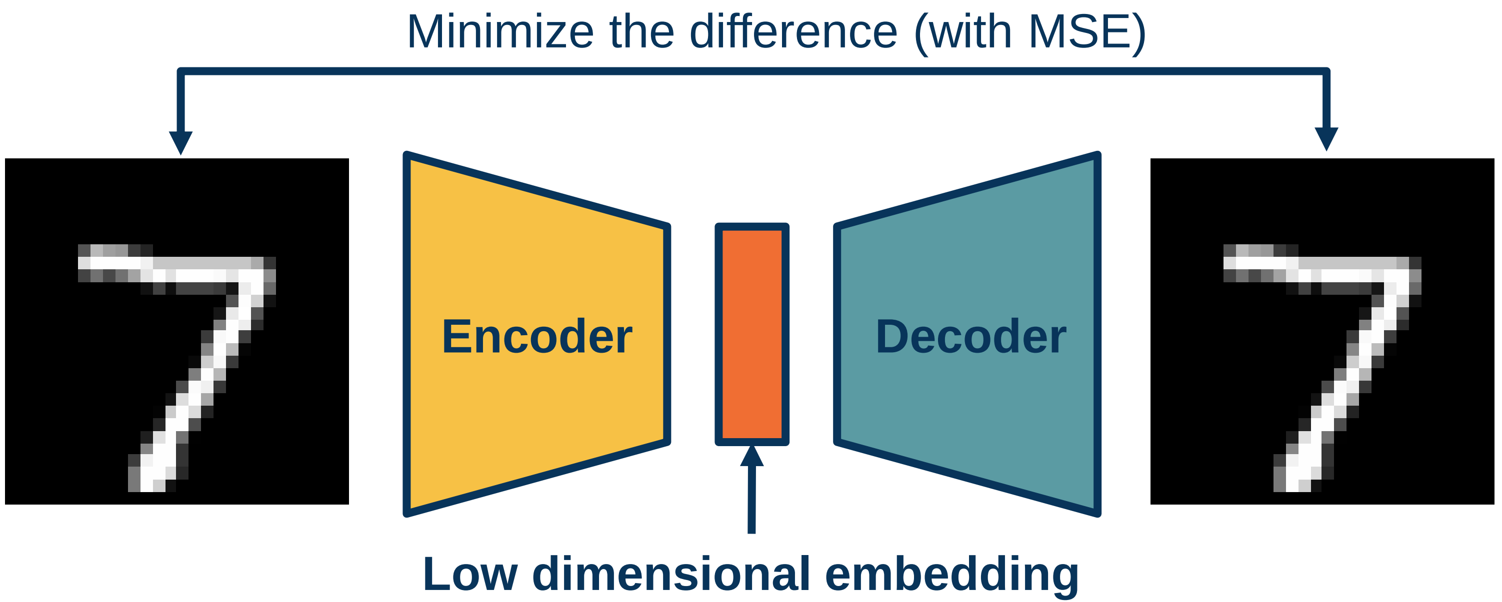


Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

Comparison


GAN: Adversarial training
x'
x
Discriminator
$D(\mathbf{x})$
0/1
z
Generator
$G(\mathbf{z})$
x'
VAE: maximize variational lower bound
x
Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
z
Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
x'
Flow-based models: Invertible transform of distributions
x
Flow
$f(\mathbf{x})$
z
Inverse
$f^{-1}(\mathbf{z})$
x'
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
z

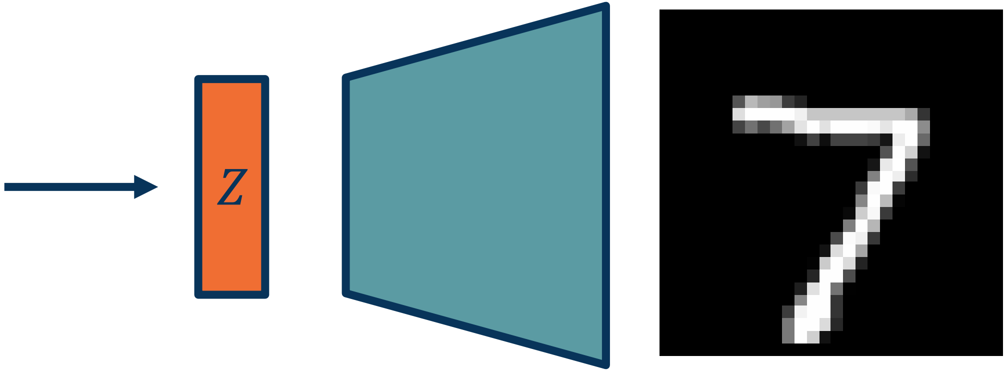


Linear layers with reduced dimension or Conv-2d layers with stride

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

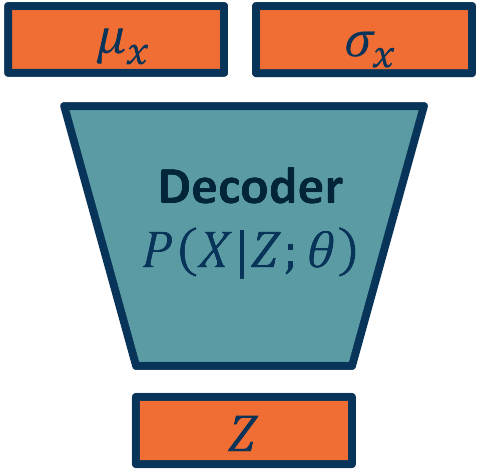
Kingma & Welling, Auto-Encoding Variational Bayes

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

- Just as before, sample Z from simpler distribution

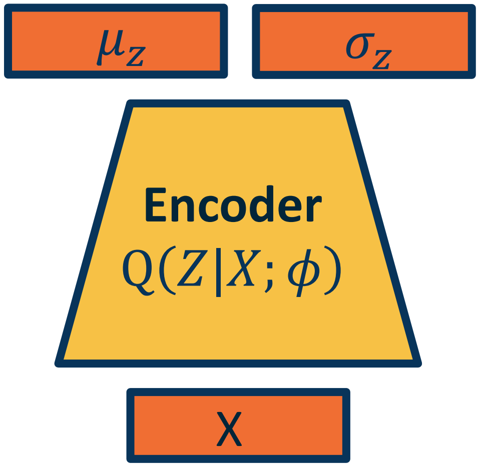
- We can also output parameters of a probability distribution!
  - **Example**: μ, σ of Gaussian distribution
  - For multi-dimensional version output diagonal covariance

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



Variational Autoencoder: Decoder

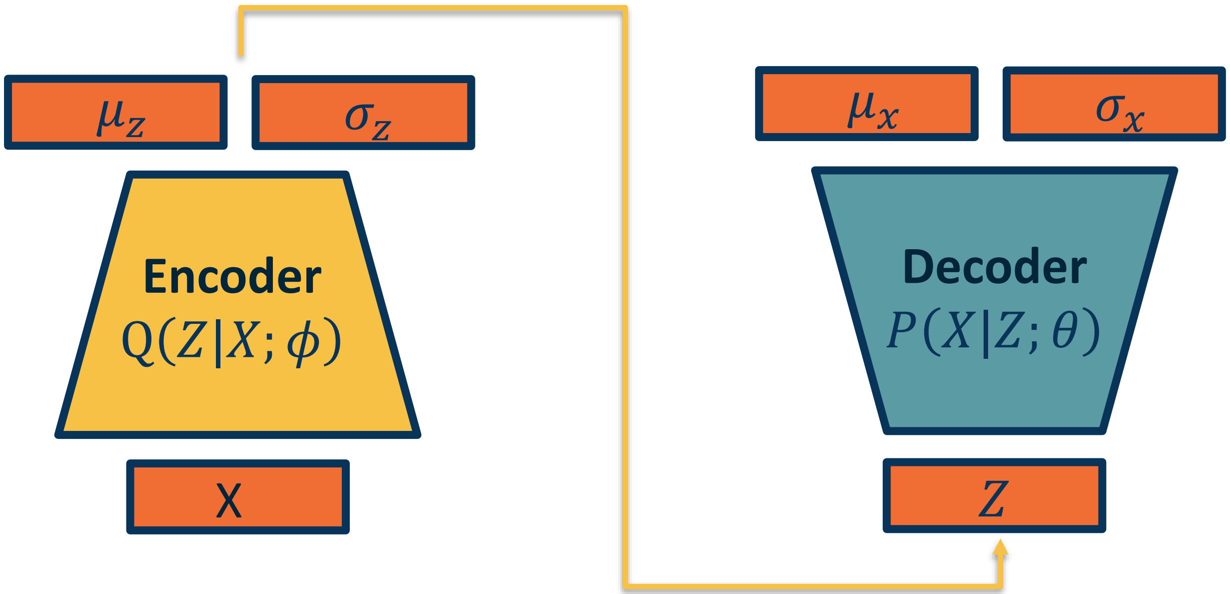
- We can combine the probabilistic view, sampling, autoencoders, and approximate optimization



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

Putting Them Together

- 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 $N(\mu_x, \sigma_x)$



- 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:

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

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

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

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

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

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

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

Maximizing Likelihood

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

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

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

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

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

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 $p_\theta(x|z)$, 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

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

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

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

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

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

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

Variational lower bound ("ELBO")

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

Training: Maximize lower bound

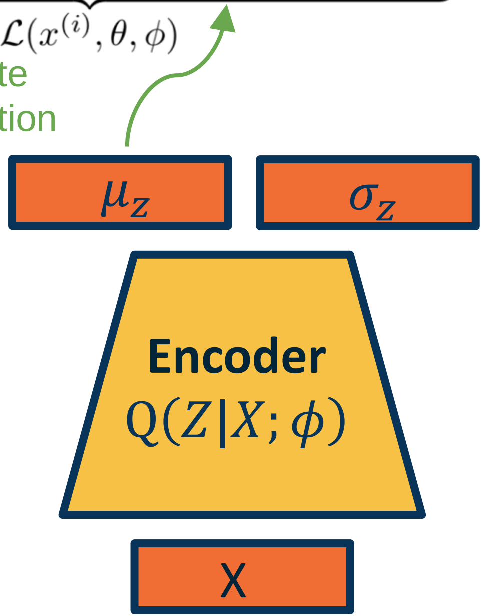
From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

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

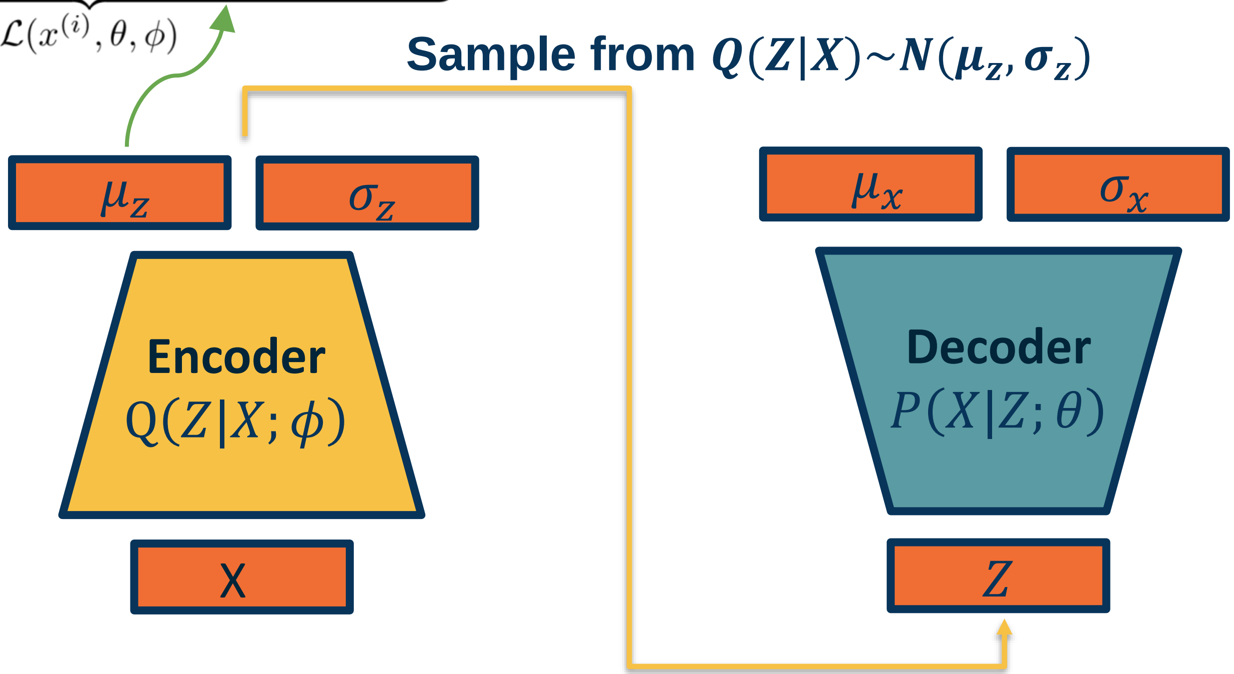


From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Putting it all together: maximizing the likelihood lower bound

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

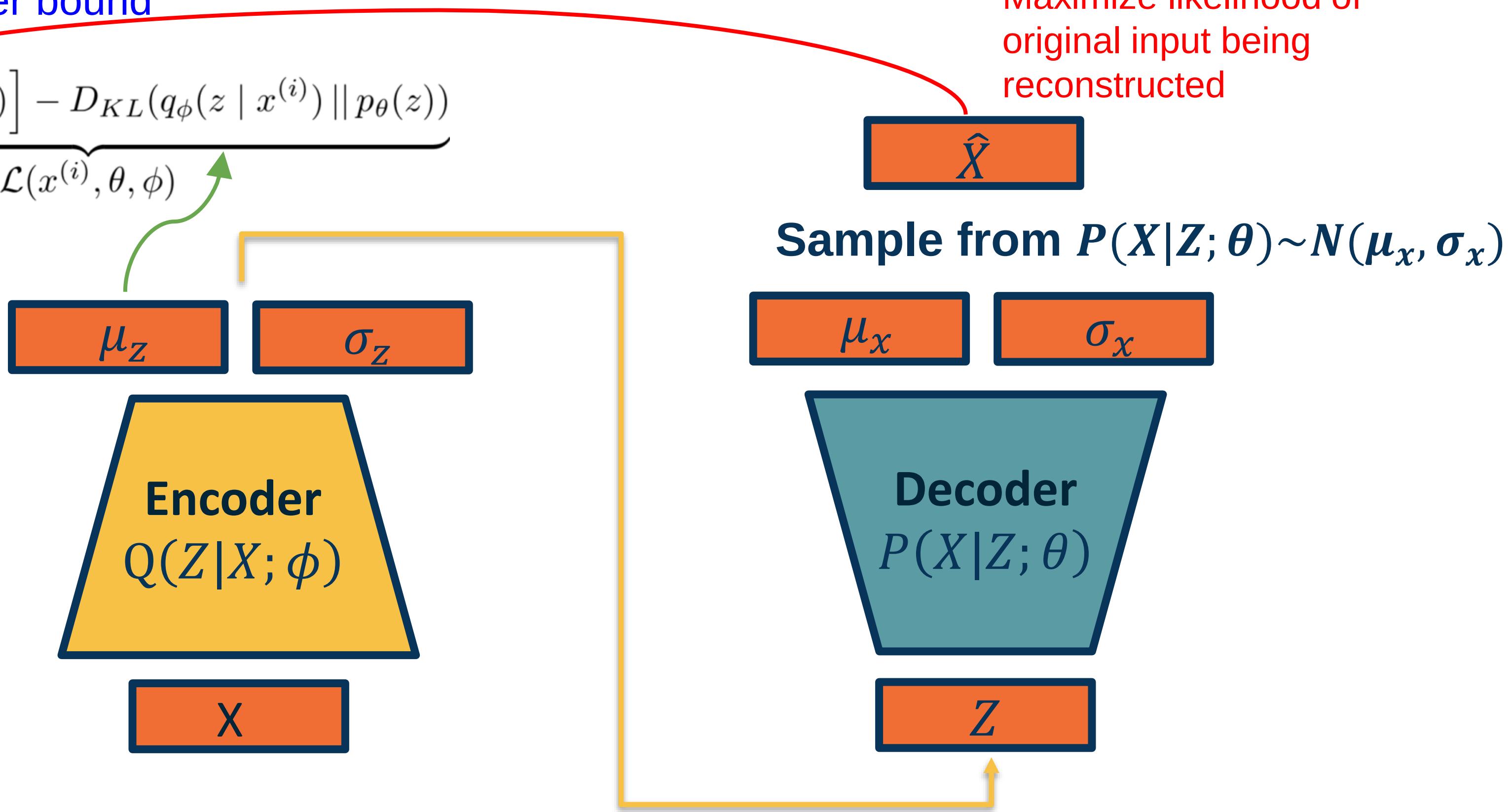
Sample from $Q(Z|X) \sim N(\mu_z, \sigma_z)$



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

Putting it all together: maximizing the likelihood lower bound

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

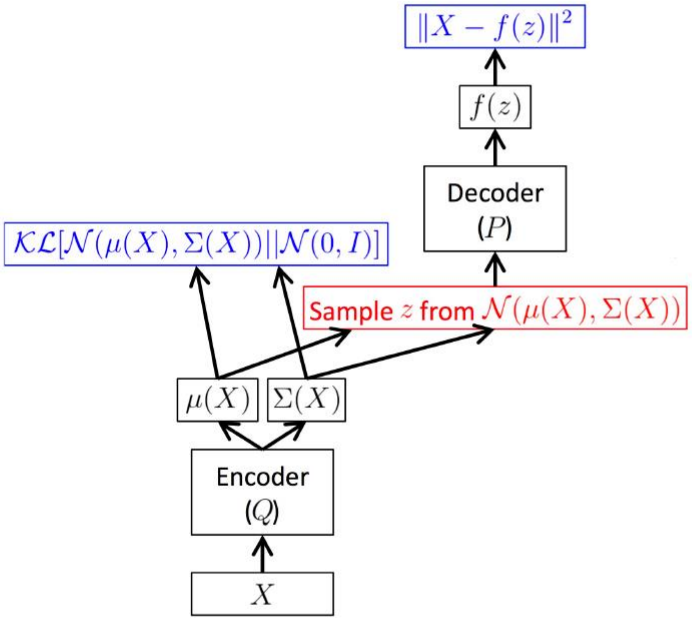


From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

- 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)

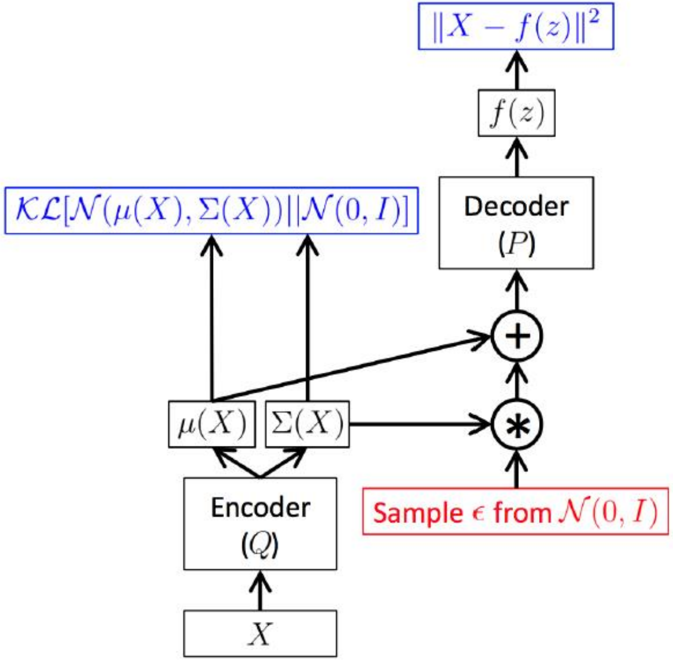


From: Tutorial on Variational Autoencoders https://arxiv.org/abs/1606.05908

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

Problem

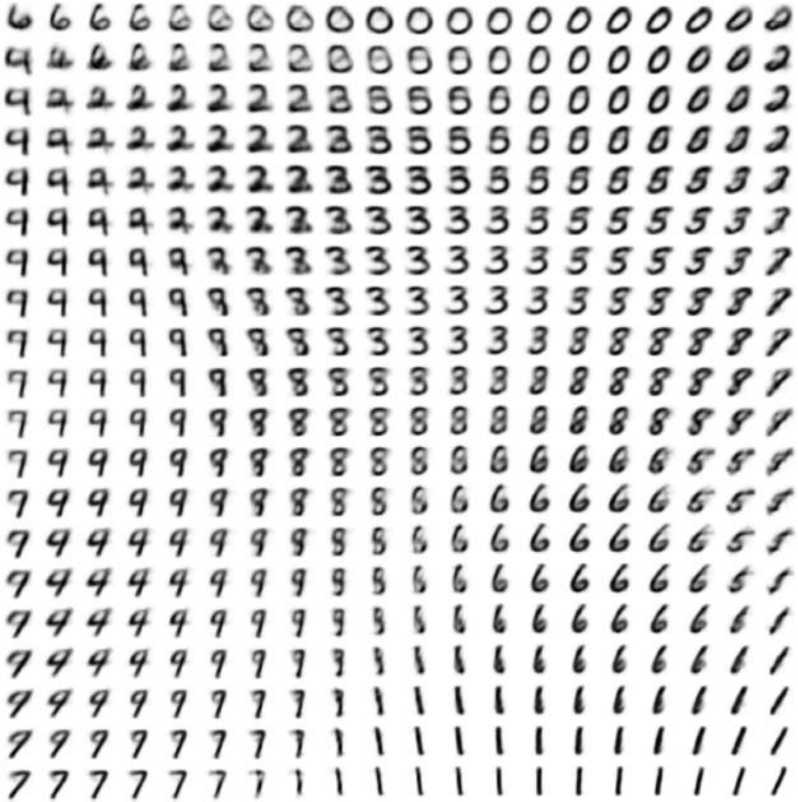
- Solution: make the randomness independent of encoder output, making the encoder deterministic
- Gaussian distribution example:
  - Previously: encoder output = random variable $z \sim N(\mu, \sigma)$
  - Now encoder output = distribution parameter $[\mu, \sigma]$
  - $z = \mu + \epsilon * \sigma, \epsilon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



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)

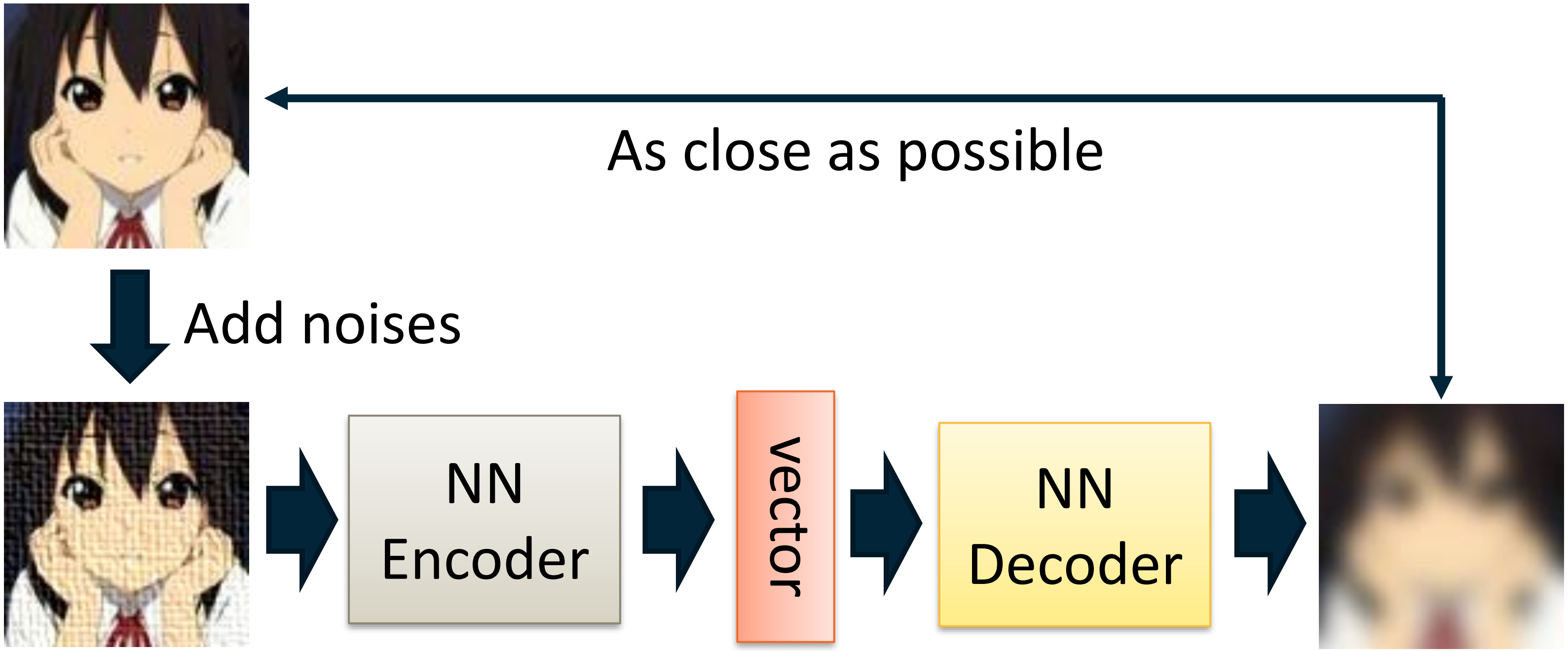
- Samples are often not as competitive as diffusion models or GANs

- 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

De-noising Auto-encoder

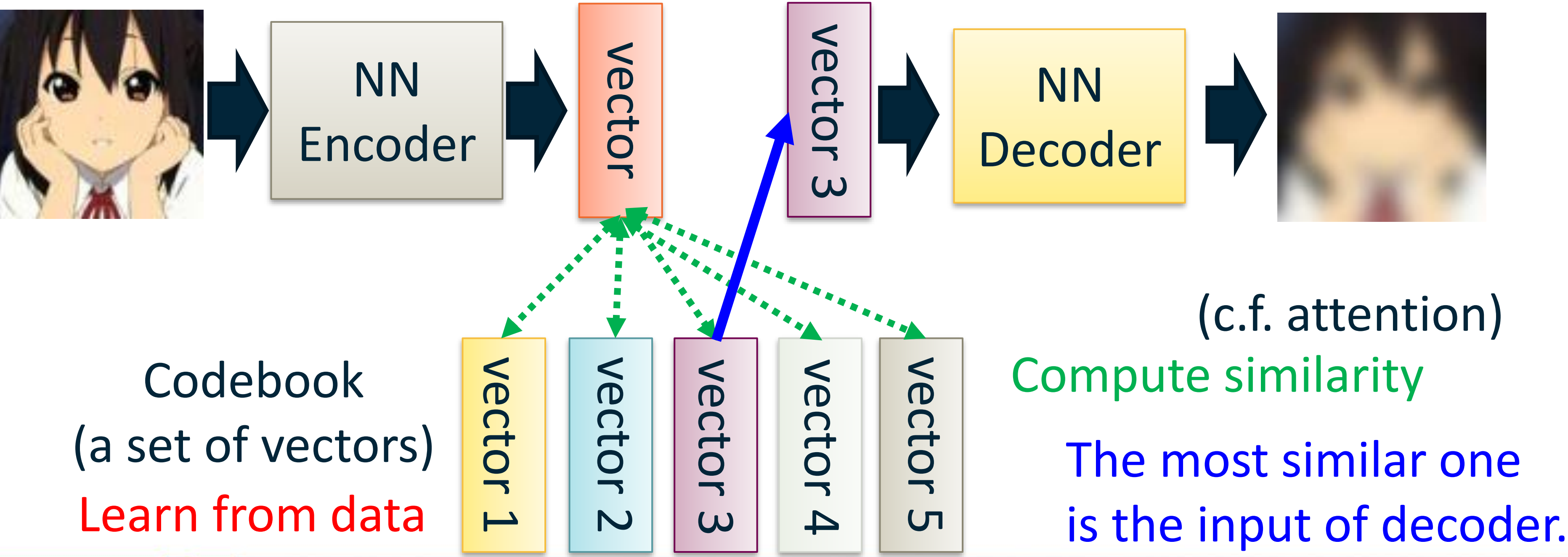


Slide by Hung-yi Lee

Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML,* 2008.

Discrete Representation

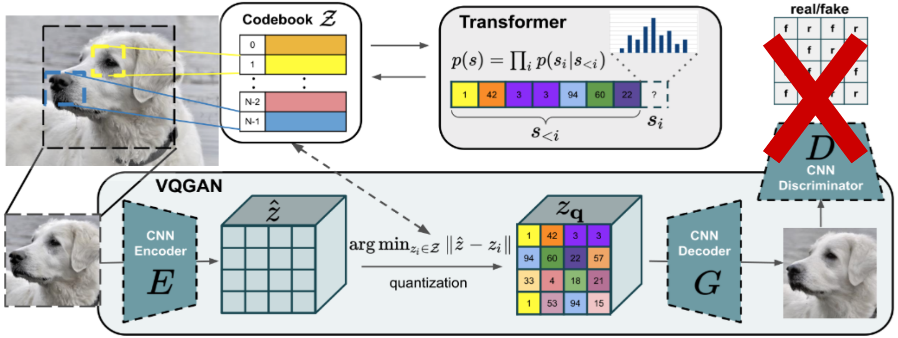
- Vector Quantized Variational Auto-encoder (VQVAE)



VQVAE – Vector Quantized VAE

VQ-VAE + Transformers:

- VQ-VAE to build a codebook (dictionary) of features.
- Transformer to predict those codebook vectors (features) autoregressively, starting from Layer 0.
  - VQVAE sees whole set of features. Decodes it into 64* tokens.
  - Transformer sees previous tokens, outputs probabilities over the next one.



Results used for latent space diffusion!

- 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

Comparison

GAN: Adversarial training

VAE: maximize variational lower bound

Flow-based models: Invertible transform of distributions

Diffusion models: Gradually add Gaussian noise and then reverse