Topics:

- Variational Autoencoders

CS 4803-DL / 7643-A
ZSOLT KIRA

- **Assignment 3 out**
  - Due **March 14th 11:59pm EST**

- Quiz **March 18th**

- Project milestone
  - Due ~~**March 14**~~ **March 20th 11:59pm EST**

Generative Models: Introduction

Supervised Learning

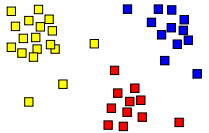
- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, $P(y|x)$
- e.g. classification

Sheep
Dog
Cat
Lion
Giraffe

Less Labels

Unsupervised Learning

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

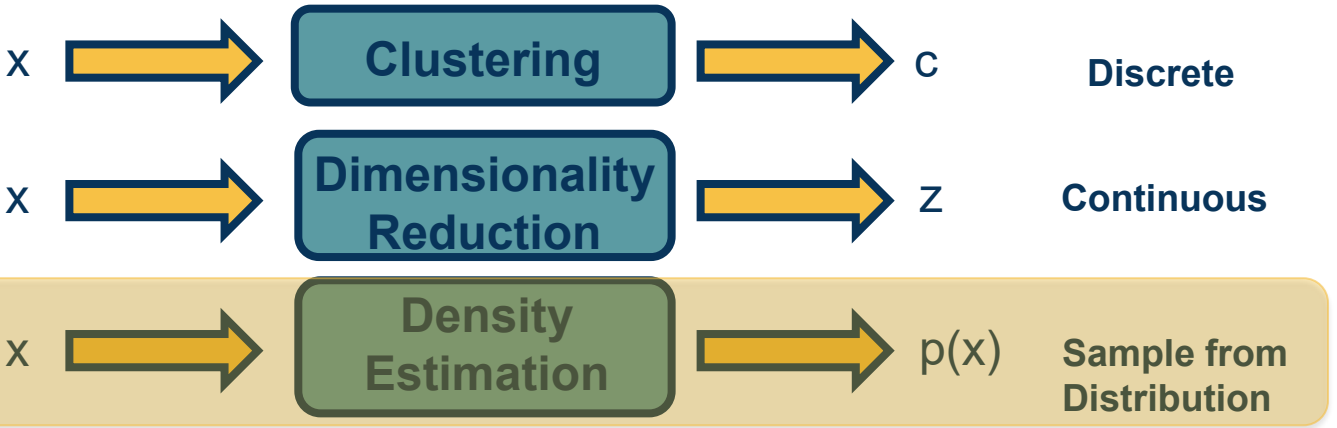


Spectrum of Low-Labeled Learning

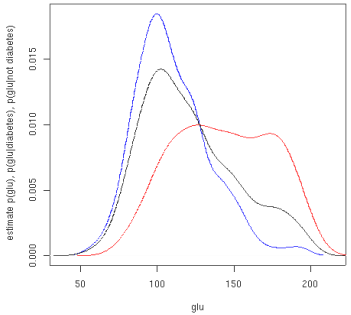
Georgia Tech

Supervised Learning

Unsupervised Learning



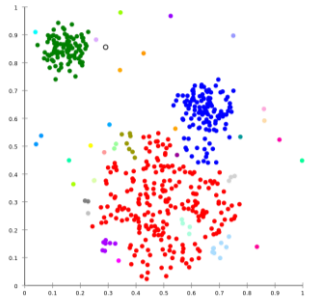
Traditional unsupervised learning methods:



Density estimation

Modeling $P(x)$

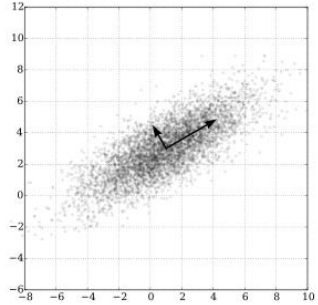
Deep Generative Models



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**

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

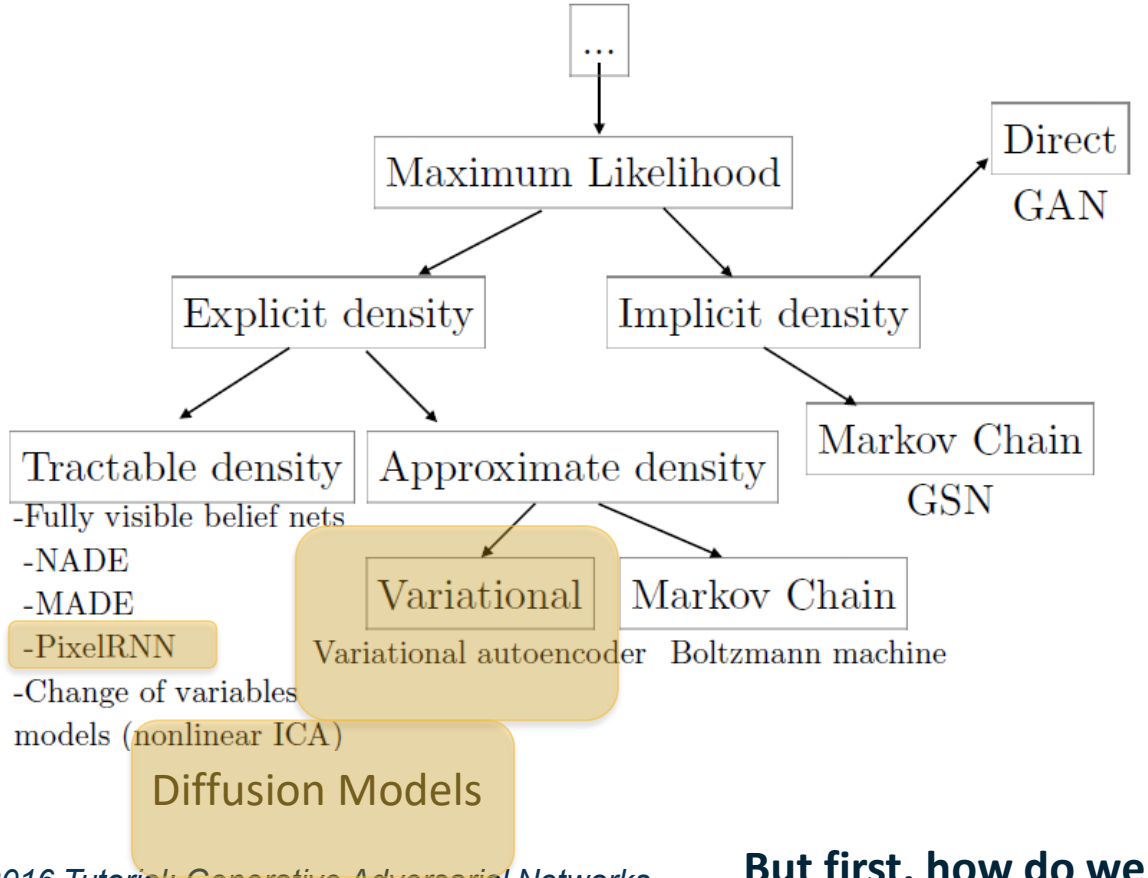
- Discriminative models model $P(y|x)$
  - Example: Model this via neural network, SVM, etc.

- 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

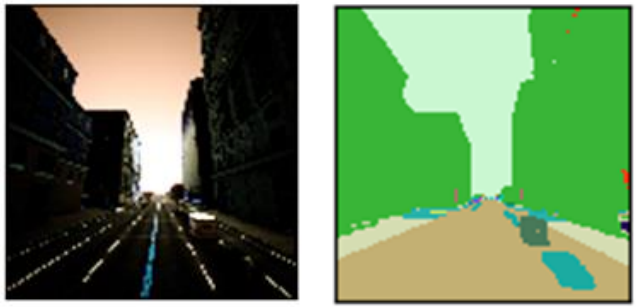
But first, how do we *generate* an image?

Generative Models

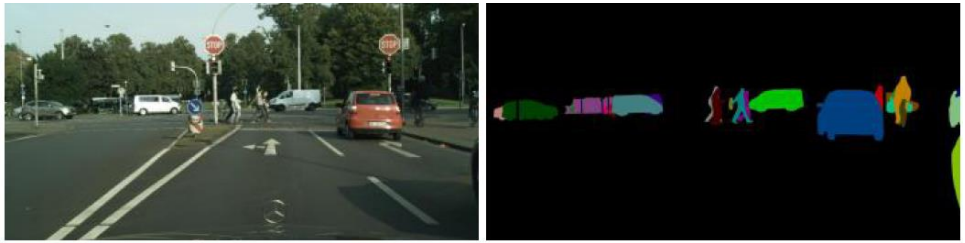
Georgia Tech

Given an image, output another image

- Each output contains class distribution per pixel
- More generally an image-to-image problem



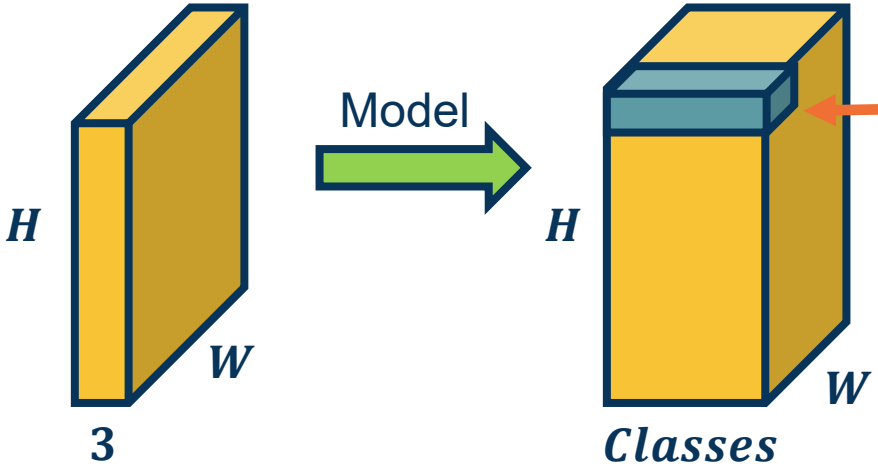
Semantic Segmentation
(Class distribution per pixel)

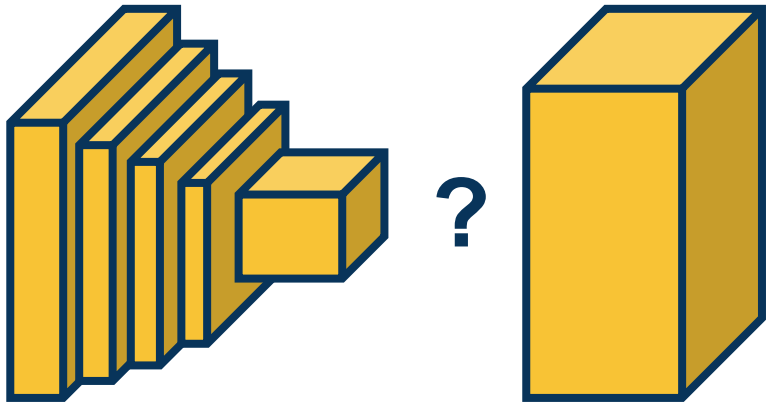


Instance Segmentation
(Class distribution per pixel with unique ID)

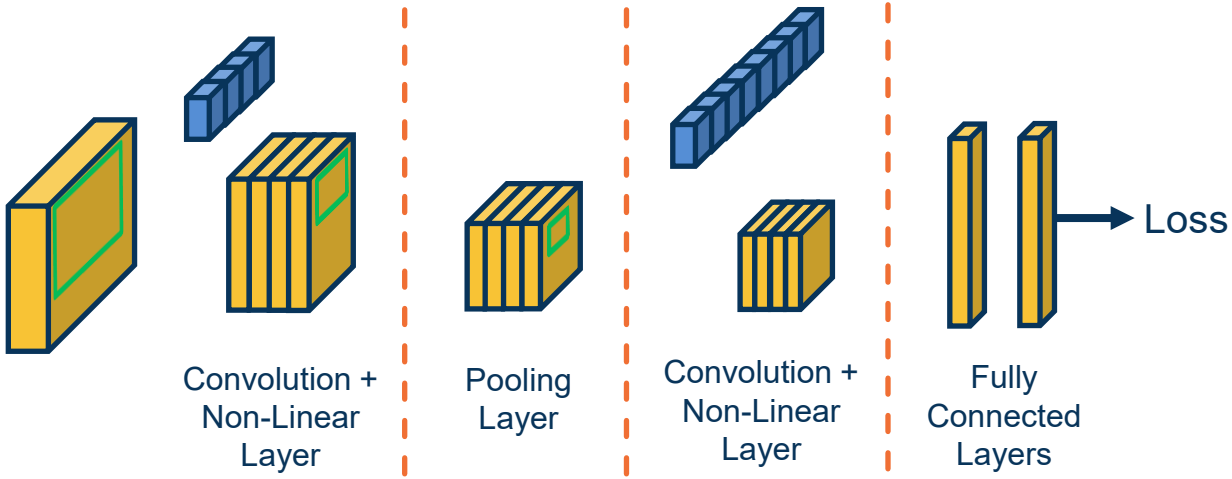
How can we generate images?

Georgia Tech



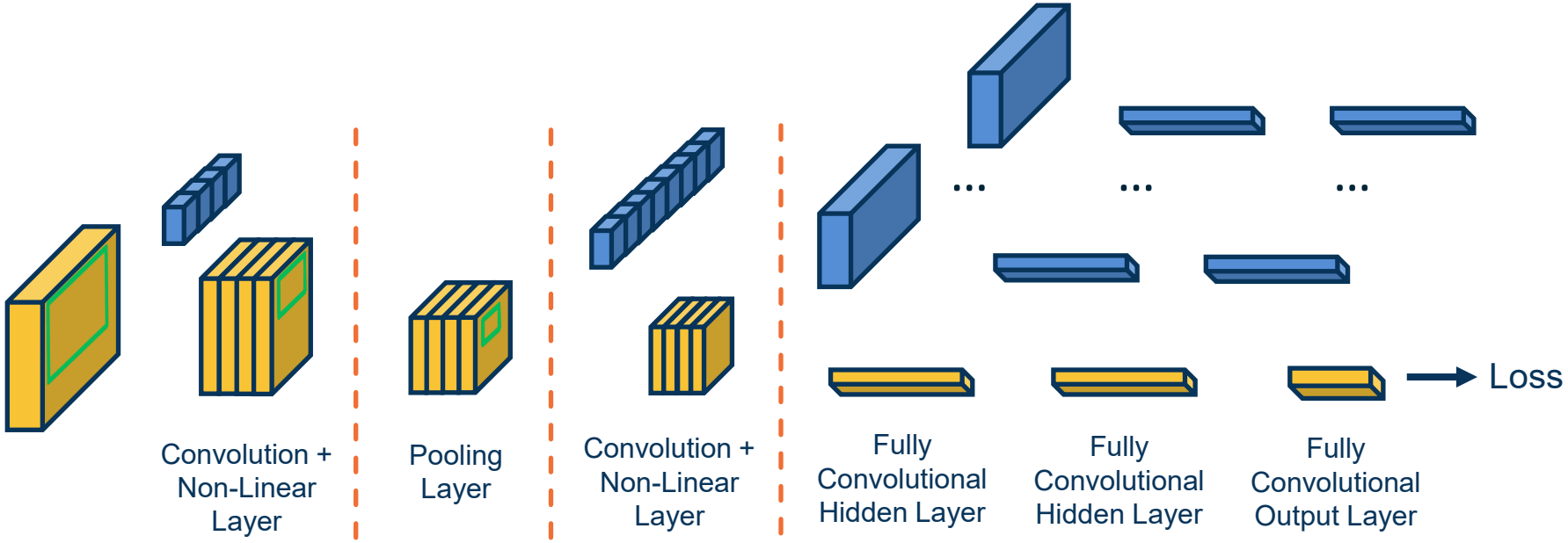


Input & Output



Fully connected layers no longer explicitly retain spatial information (though the network can still learn to do so)

Idea: Convert fully connected layer to convolution!



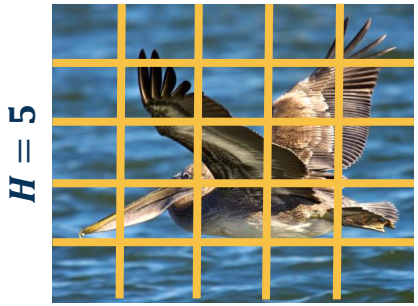
Each kernel has the size of entire input! (output is 1 scalar)

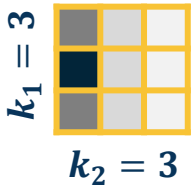
- This is equivalent to Wx+b!
- We have one kernel per output node

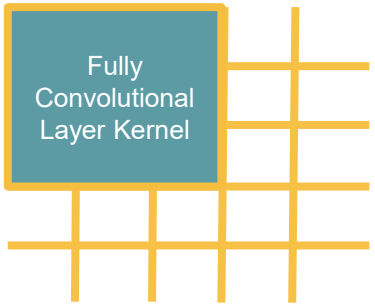
Converting FC Layers to Conv Layers

Georgia Tech

Original:

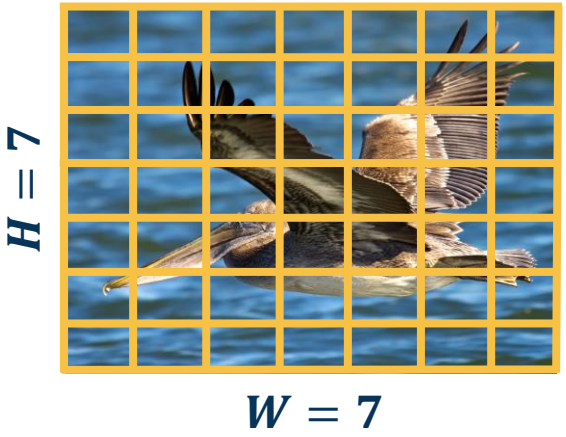


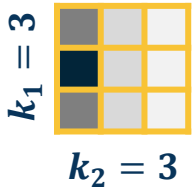


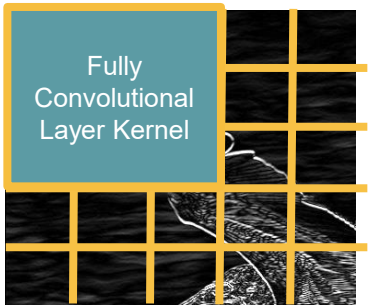


Input Conv Kernel Output

Larger:







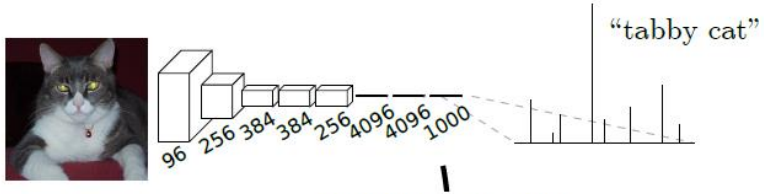
Same Kernel, Larger Input

Georgia Tech

Why does this matter?

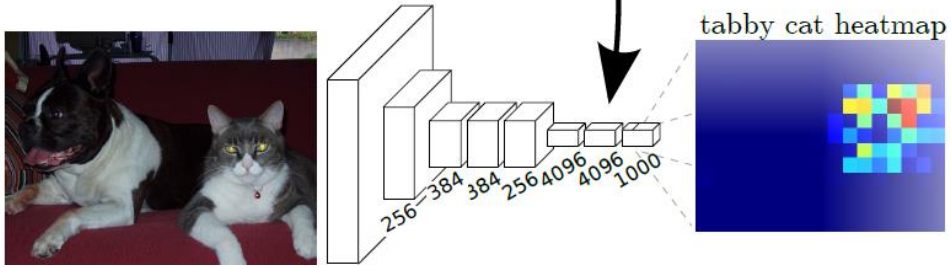
- We can stride the “fully connected” classifier across larger inputs!
- Convolutions work on arbitrary input sizes (because of striding)

Original sized image



convolutionalization

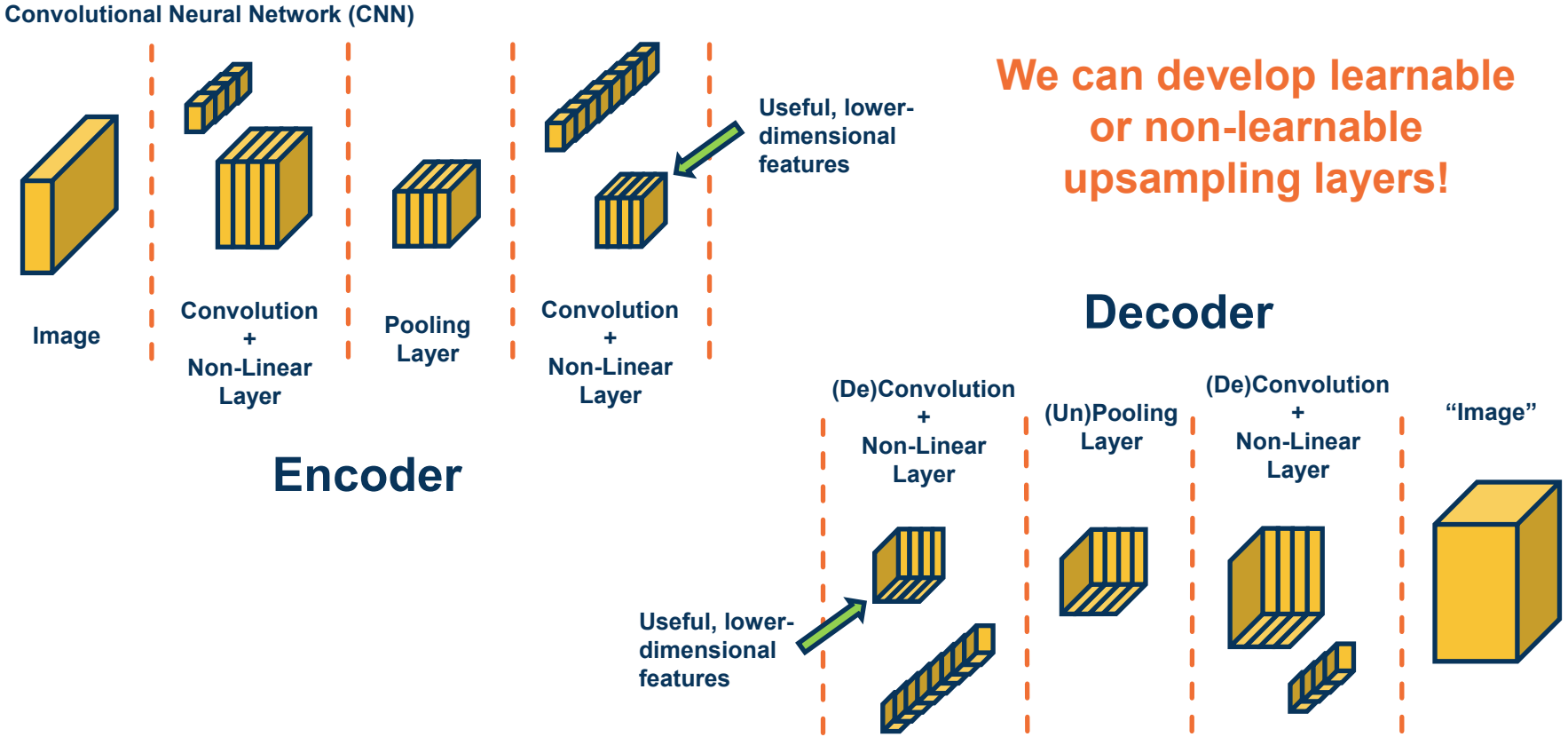
Larger Image



Larger Output Size!

Larger Output Maps

Long, et al., “Fully Convolutional Networks for Semantic Segmentation”, 2015

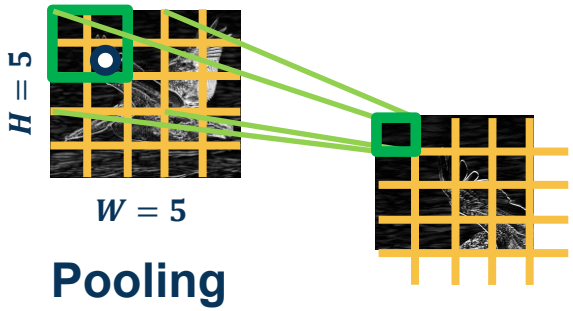


Idea 2: "De"Convolution and UnPooling

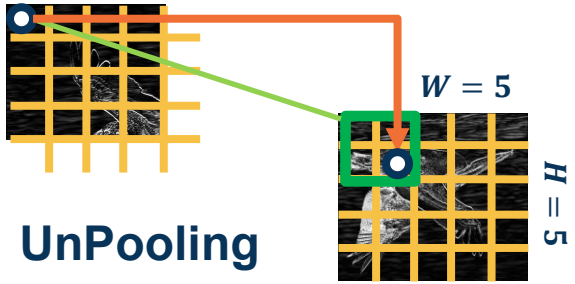
Example : Max pooling

- Stride window across image but perform per-patch **max operation**

$$\mathbf{X}(0:1, 0:1) = \begin{bmatrix} 100 & 150 \\ 100 & 200 \end{bmatrix} \Rightarrow \max(0{:}1, 0{:}1) = 200$$



Copy value to position chosen as max in encoder, fill reset of this window with zeros



Idea: Remember max elements in encoder! Copy value from equivalent position, rest are zeros

$$X = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix}$$

2x2 max pool

$$Y = \begin{bmatrix} 150 & 150 \\ 100 & 110 \end{bmatrix}$$

Encoder

Decoder

$$X = \begin{bmatrix} 300 & 450 \\ 100 & 250 \end{bmatrix}$$

2x2 max unpool

$$Y = \begin{bmatrix} 0 & 300 & - \\ 0 & 0 & - \\ - & - & - \end{bmatrix}$$

Max Unpooling Example (one window)

Georgia Tech

$$X_{enc} = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix}$$

2x2 max pool

$$Y_{enc} = \begin{bmatrix} 150 & 150 \\ 100 & 110 \end{bmatrix}$$

Encoder

Contributions from multiple windows are summed

Decoder

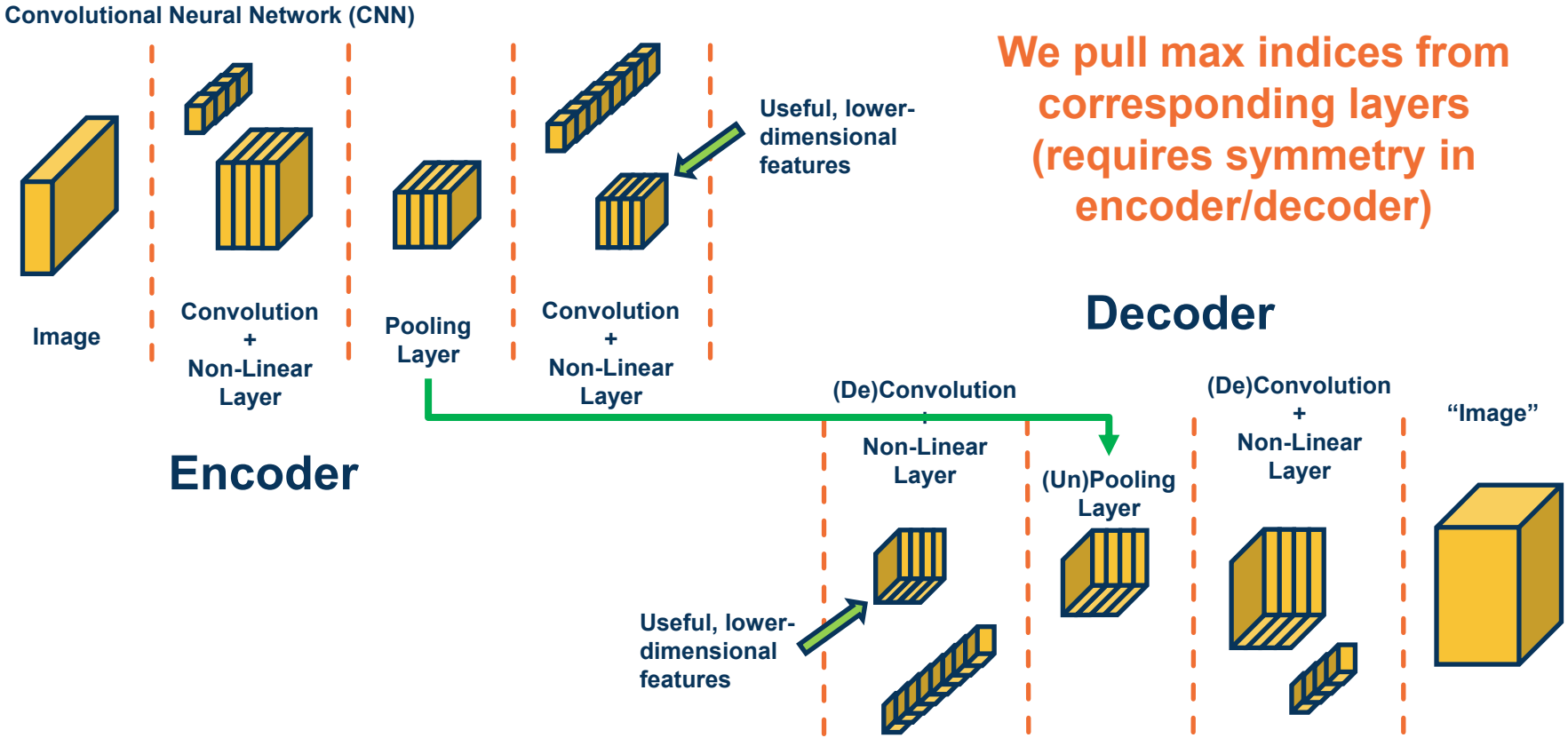
$$X_{dec} = \begin{bmatrix} 300 & 450 \\ 100 & 250 \end{bmatrix}$$

2x2 max unpool

$$Y_{dec} = \begin{bmatrix} 0 & 300+450 & 0 \\ 100 & 0 & 250 \\ 0 & 0 & 0 \end{bmatrix}$$

Max Unpooling Example

Georgia Tech

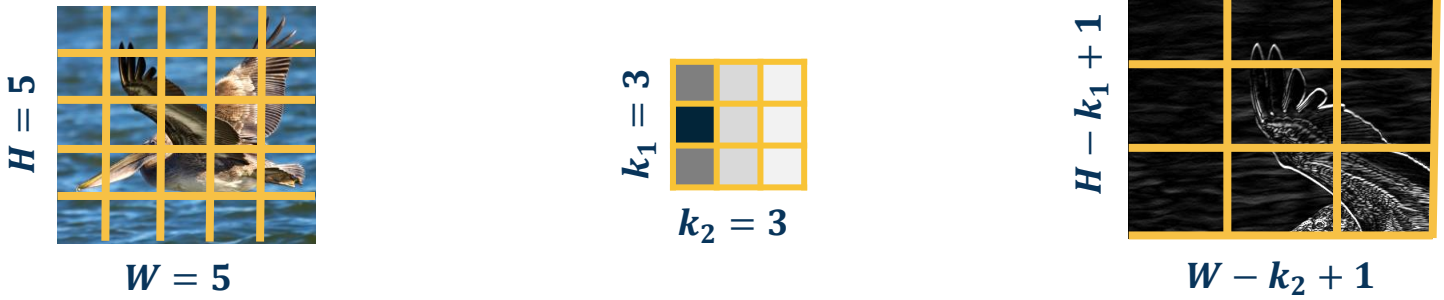


Symmetry in Encoder/Decoder

Georgia Tech

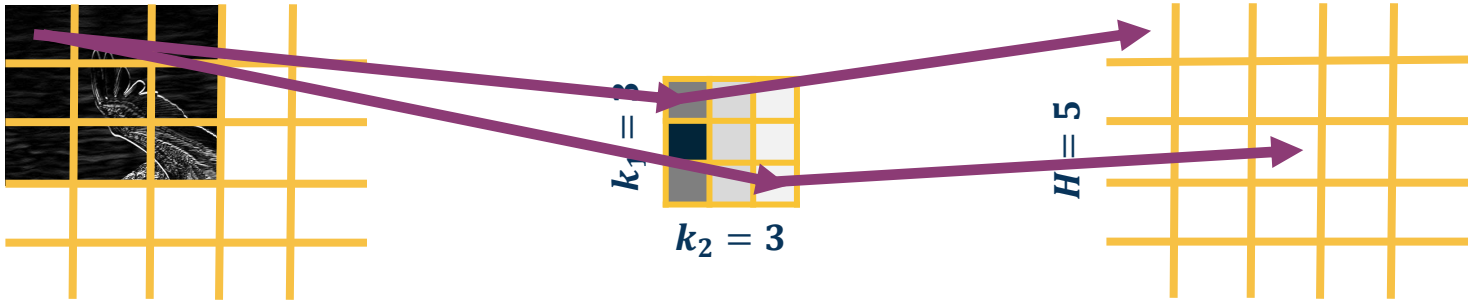
How can we *upsample* using convolutions and learnable kernel?

Normal Convolution



Transposed Convolution (also known as "deconvolution", fractionally strided conv)

Idea: Take each input pixel, multiply by learnable kernel, "stamp" it on output



$$X = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix} \qquad K = \begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix}$$

Contributions from multiple windows are summed

$$\begin{bmatrix} 120 & -120 & 0 & 0 \\ 240 & -240 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

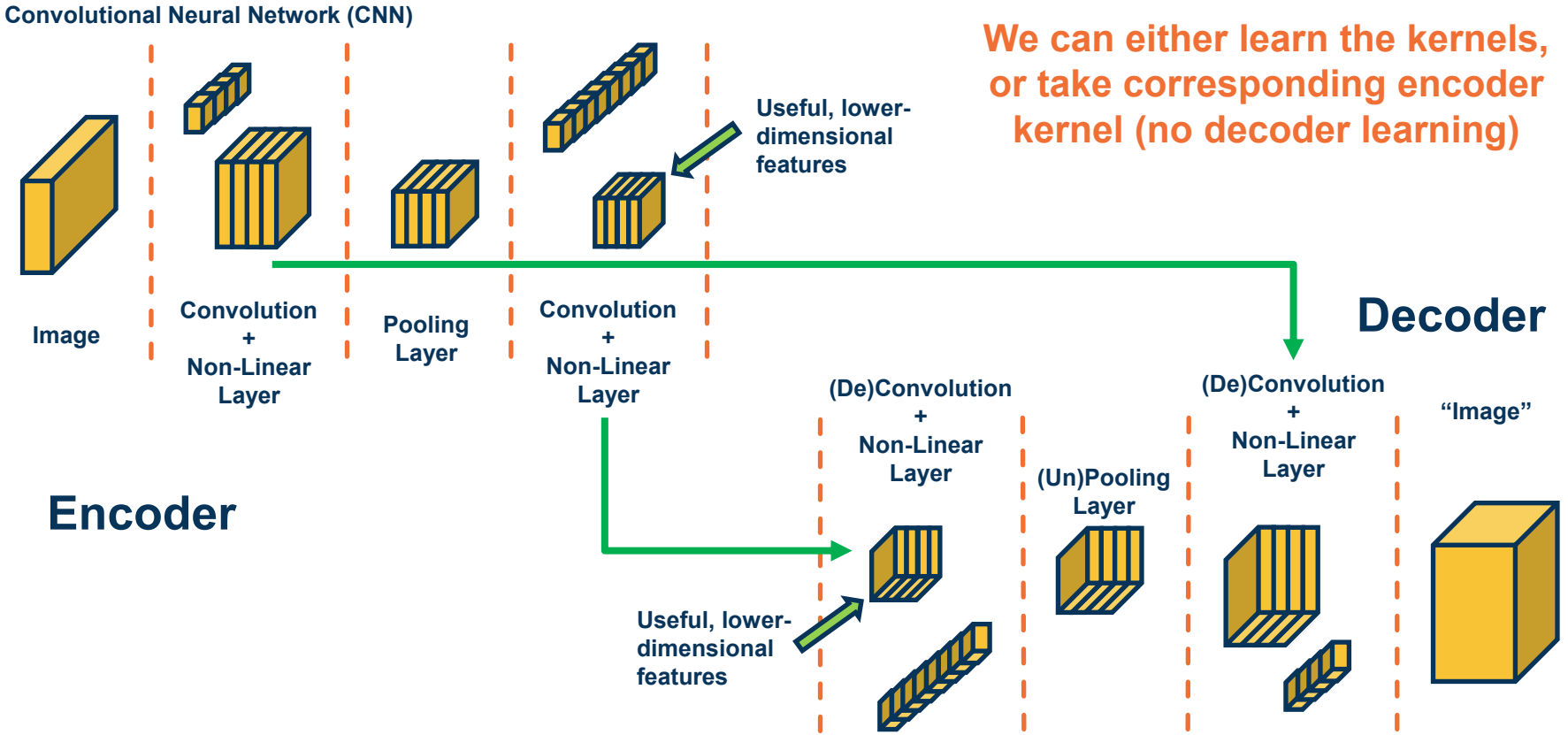
Incorporate
X(0,0)

$$\begin{bmatrix} 120 & -120+150 & -150 & 0 \\ 240 & -240+300 & -300 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Incorporate
X(1,0)

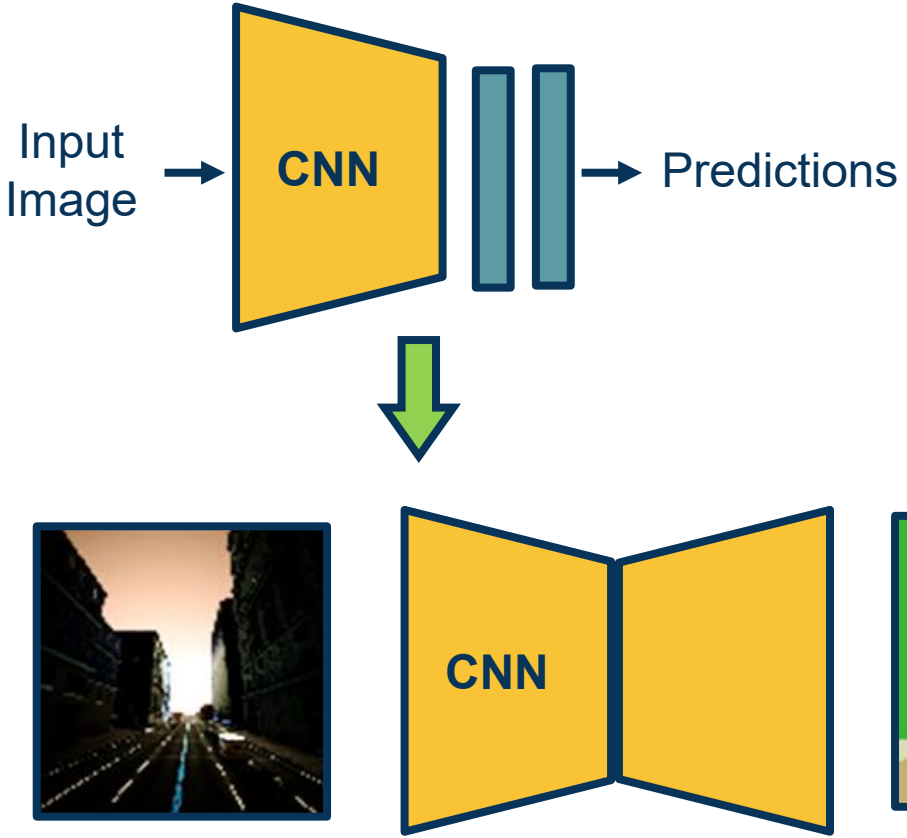
Transposed Convolution Example

Georgia Tech



Symmetry in Encoder/Decoder

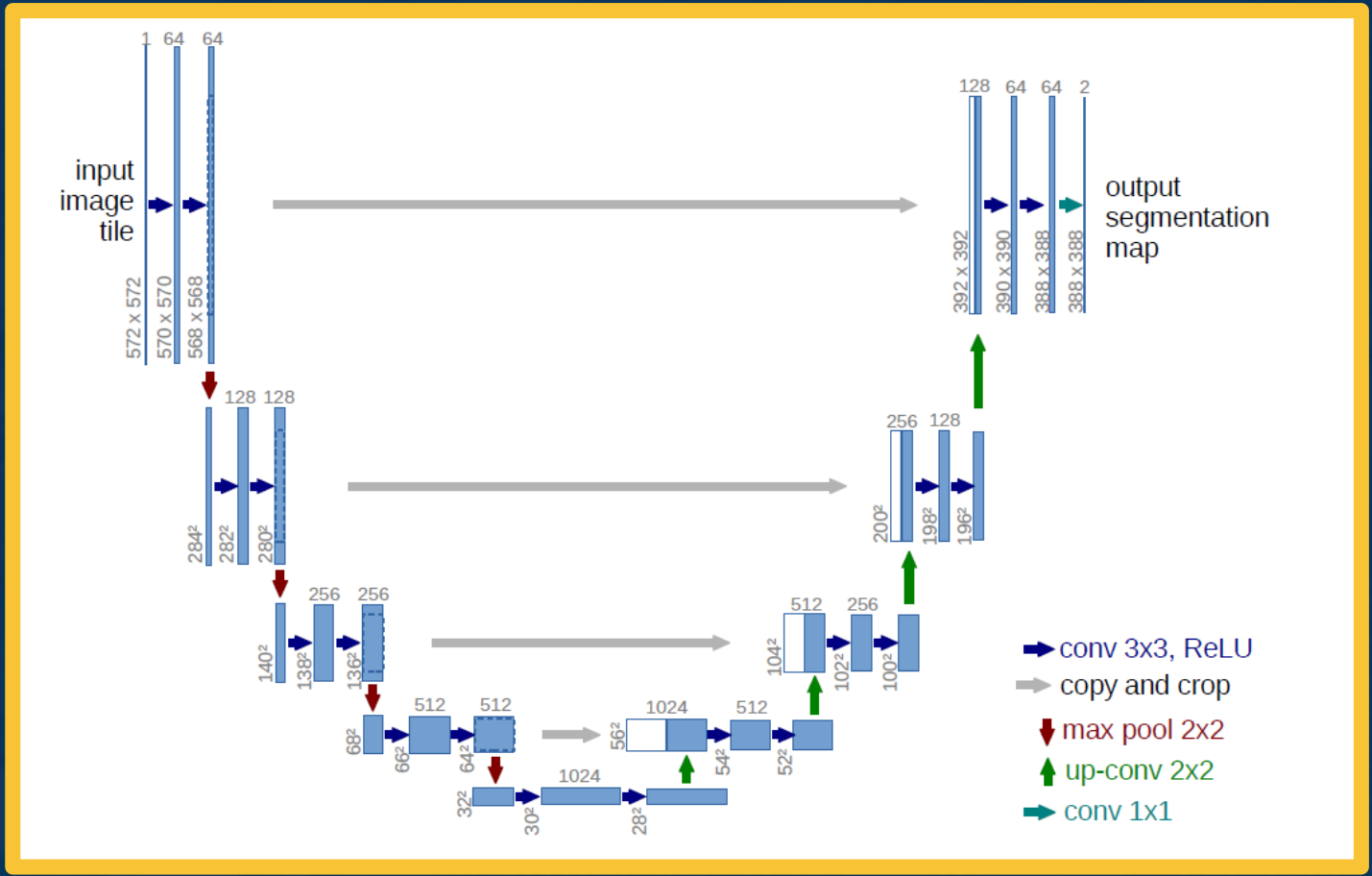
Georgia Tech



We can start with a pre-trained trunk/backbone (e.g. network pretrained on ImageNet)!

U-Net

You can have skip connections to bypass bottleneck!



Ronneberger, et al., "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015

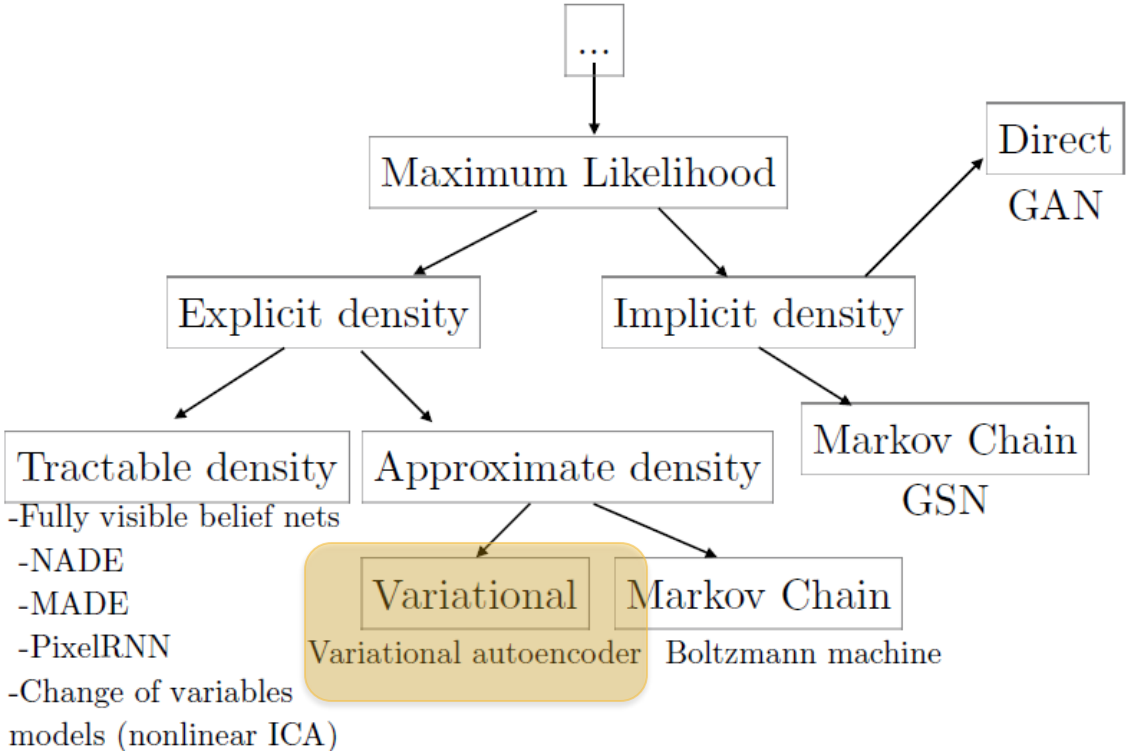
Georgia Tech

Summary

- Various ways to get **image-like outputs,** for example to predict segmentations of input images
- Fully convolutional layers essentially apply the striding idea to the output classifiers, supporting arbitrary input sizes
  - (without output size depending on what the input size is)
- We can have various upsampling layers that actually increase the size
- Encoder/decoder architectures are popular ways to leverage these to perform general image-to-image tasks

Georgia Tech

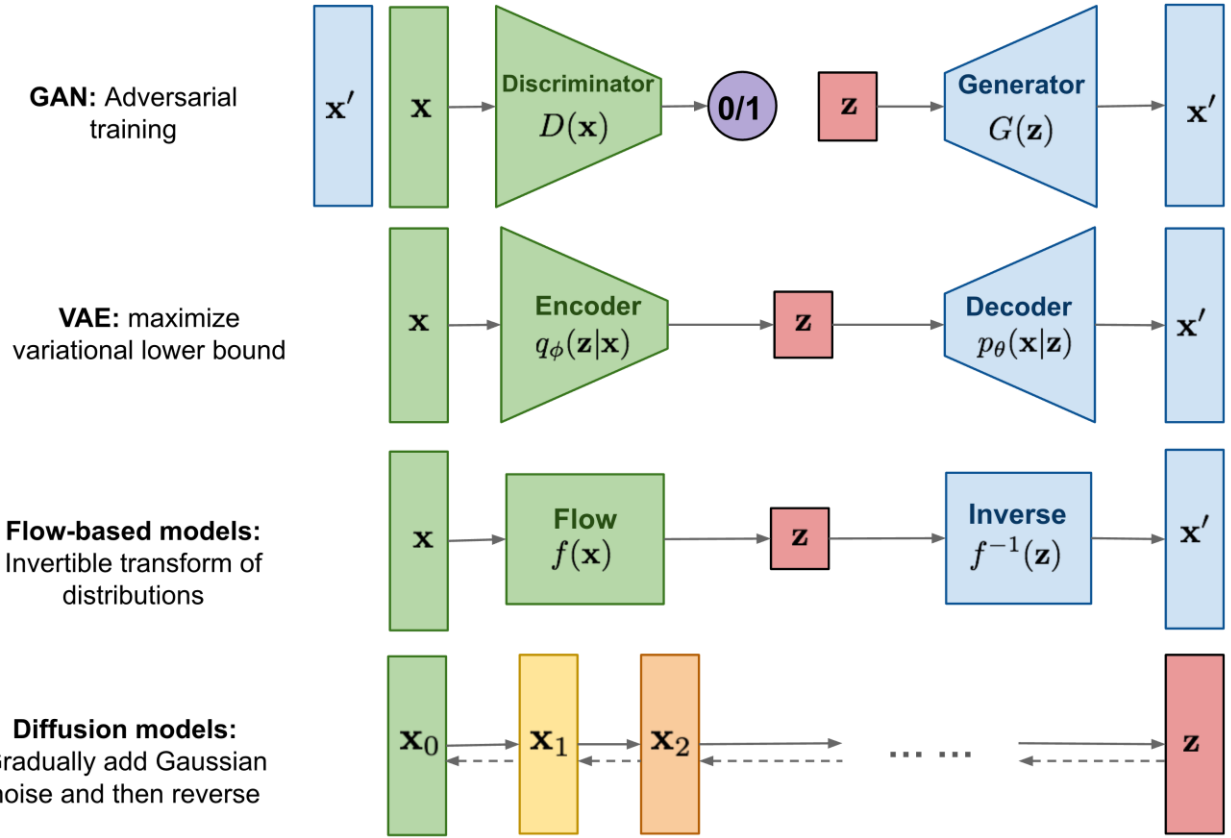
Variational Autoencoders (VAEs)

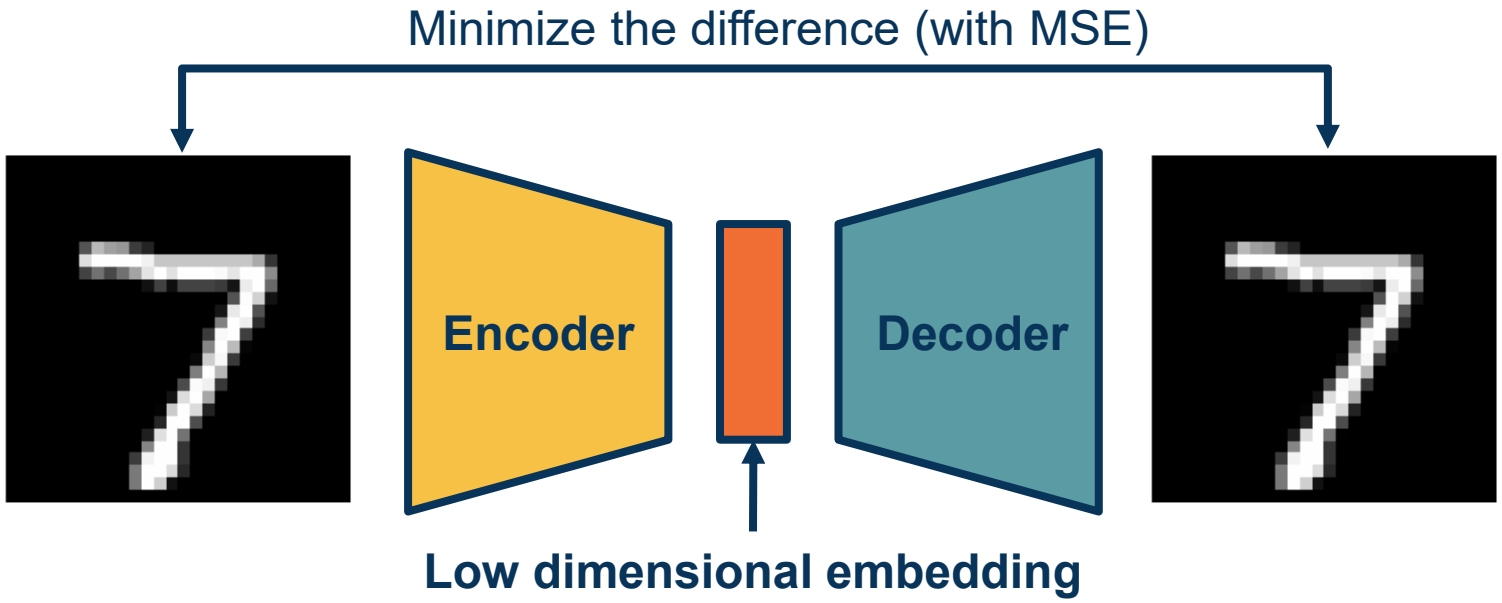


Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

Comparison

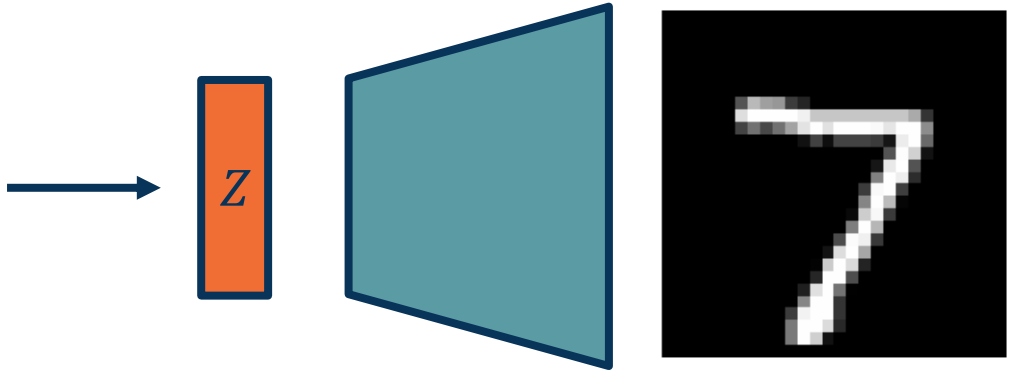

GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Flow-based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



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

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

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



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

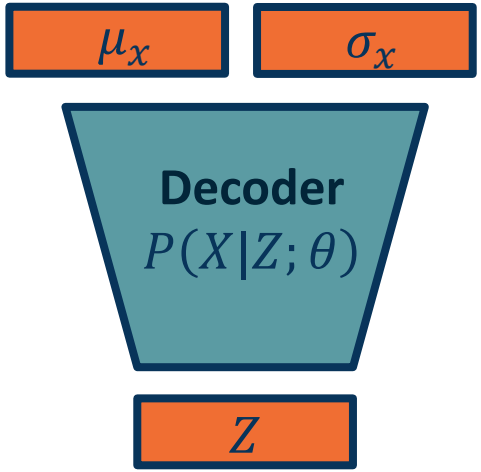
- We cannot maximize this likelihood due to the integral
- Instead we maximize a variational *lower bound* (VLB) that we *can* compute

Kingma & Welling, Auto-Encoding Variational Bayes

Formalizing the Generative Model

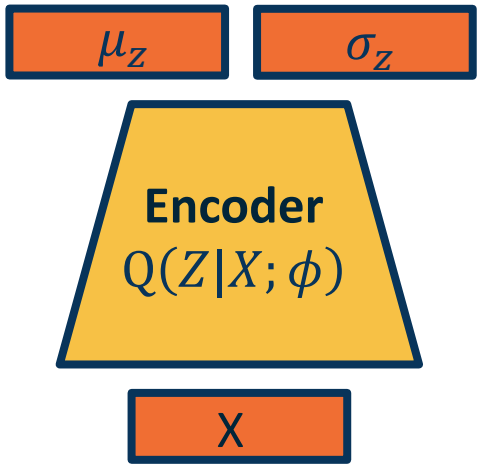
Georgia Tech

- 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

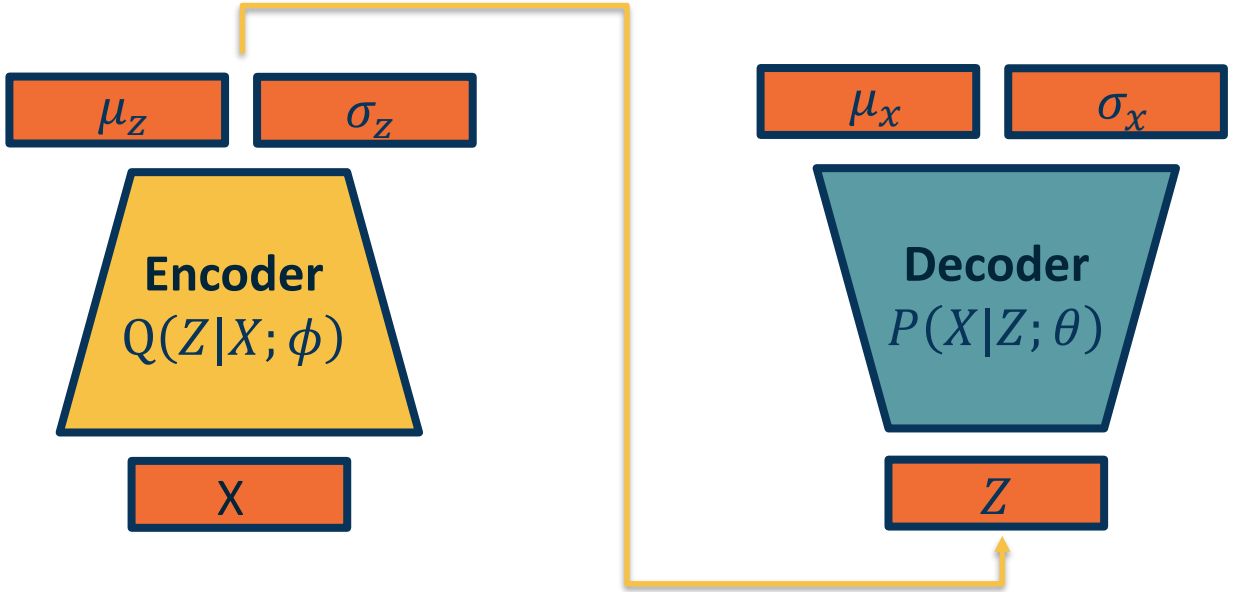
- We can combine the probabilistic view, sampling, autoencoders, and approximate optimization



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

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:

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

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

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

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

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

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

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

Maximizing Likelihood

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

$$KL(a||b) = H_c(a,b) - H(a) = \sum a\,(x)\log a(x) - \sum a(x)\log b(x)$$

Definition of Expectation

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

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

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

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

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

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

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

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

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

Maximizing Likelihood

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

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

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

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

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

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

Georgia Tech

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

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

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

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

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

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

Variational lower bound ("ELBO")

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

Training: Maximize lower bound

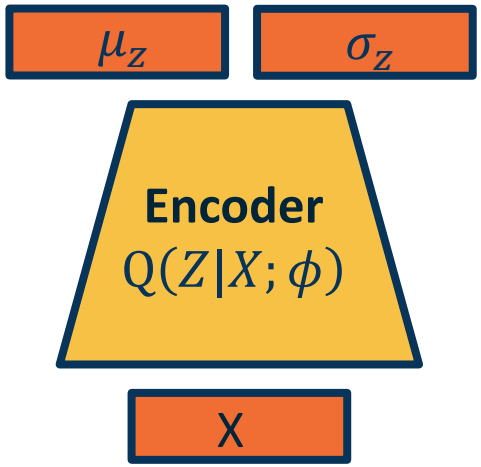
From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Maximizing Likelihood

Putting it all together: maximizing the likelihood lower bound

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

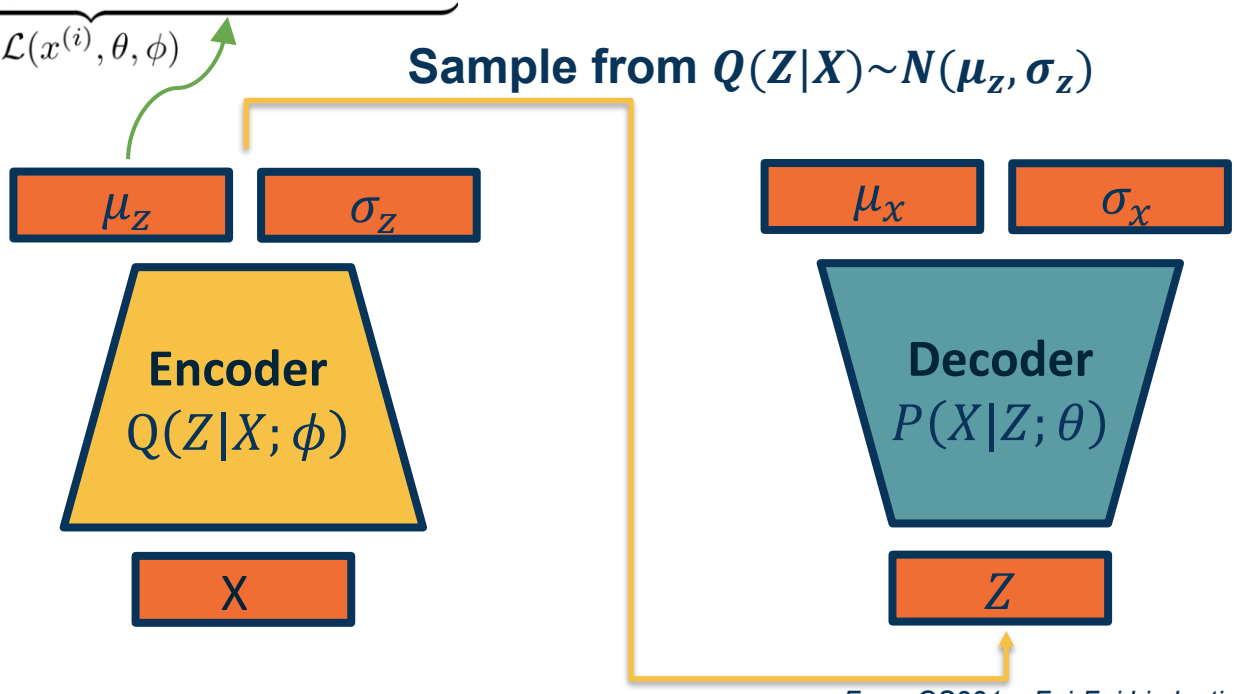
Make approximate posterior distribution close to prior



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

Putting it all together: maximizing the likelihood lower bound

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

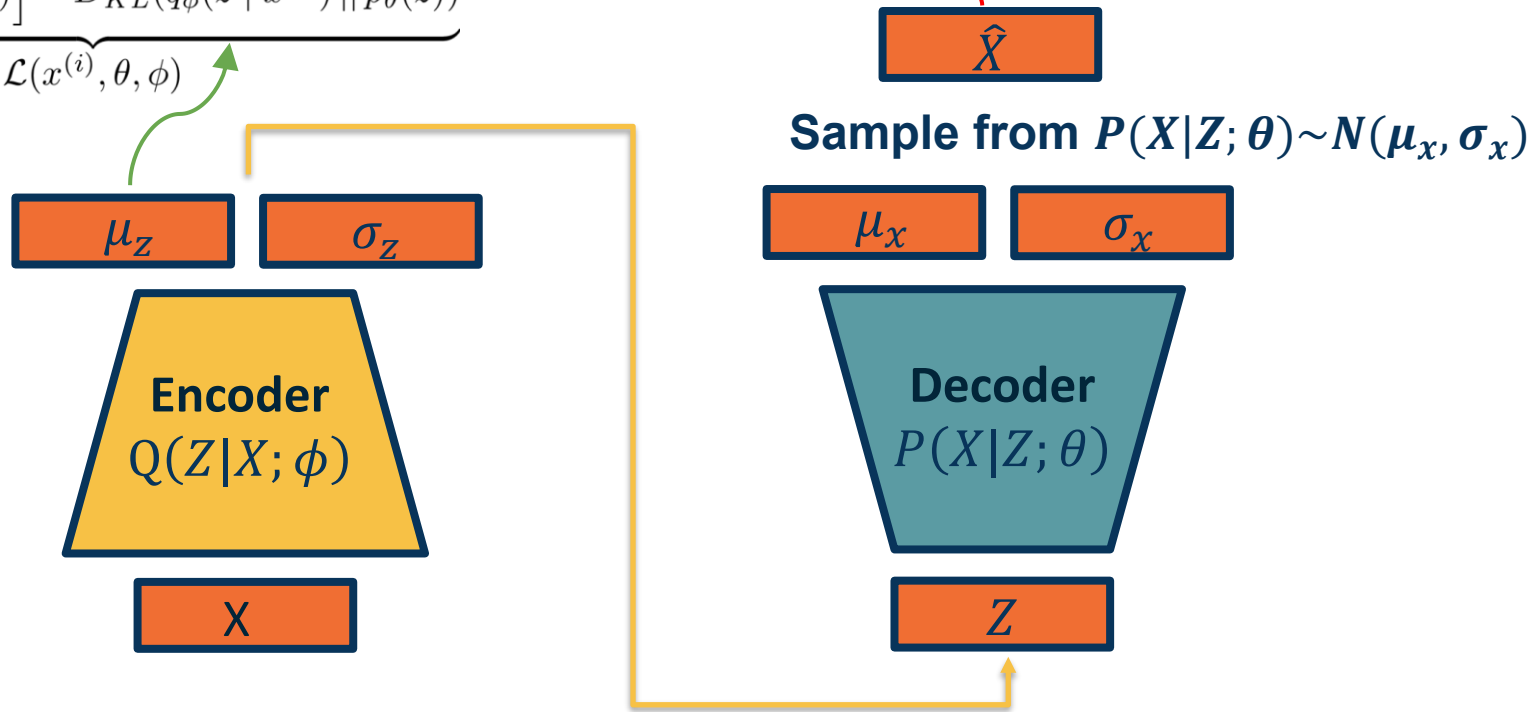


From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed



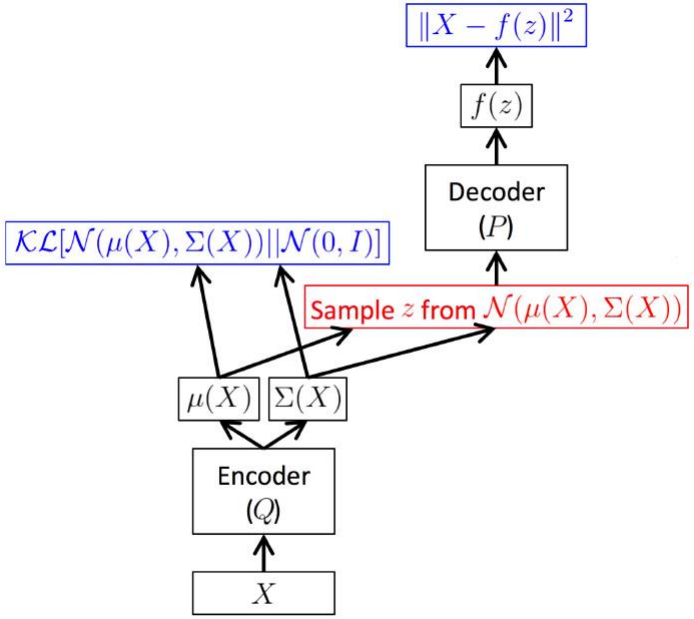
From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Forward and Backward Passes

- 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)

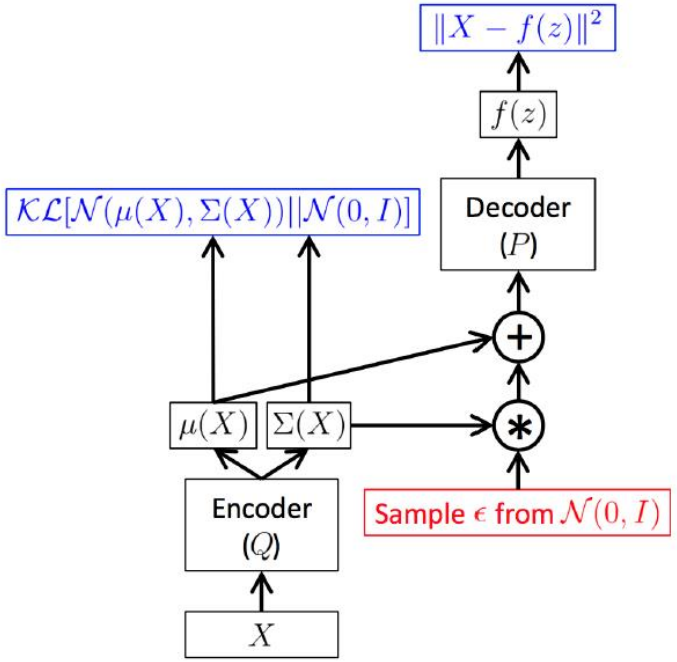


From: Tutorial on Variational Autoencoders
https://arxiv.org/abs/1606.05908

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 $[\mu, \sigma]$
  - $z = \mu + \epsilon * \sigma, \epsilon \sim N(0,1)$



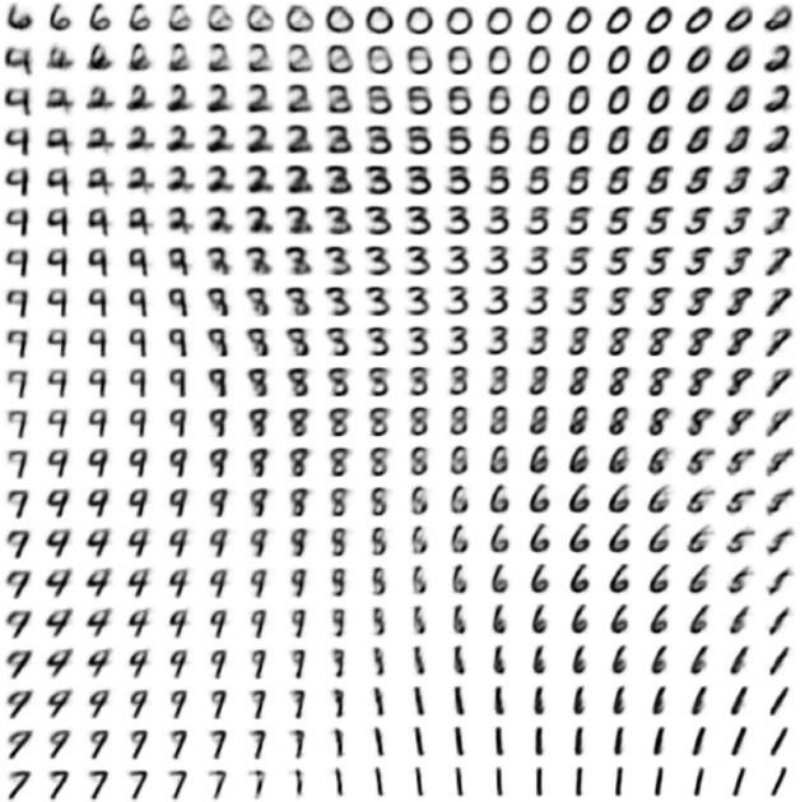
From: Tutorial on Variational Autoencoders
https://arxiv.org/abs/1606.05908

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

Reparameterization Trick: Solution

z_1

z_2



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

De-noising Auto-encoder

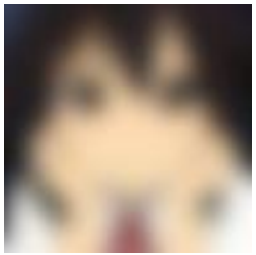
As close as possible

Add noises

NN Encoder

vector

NN Decoder

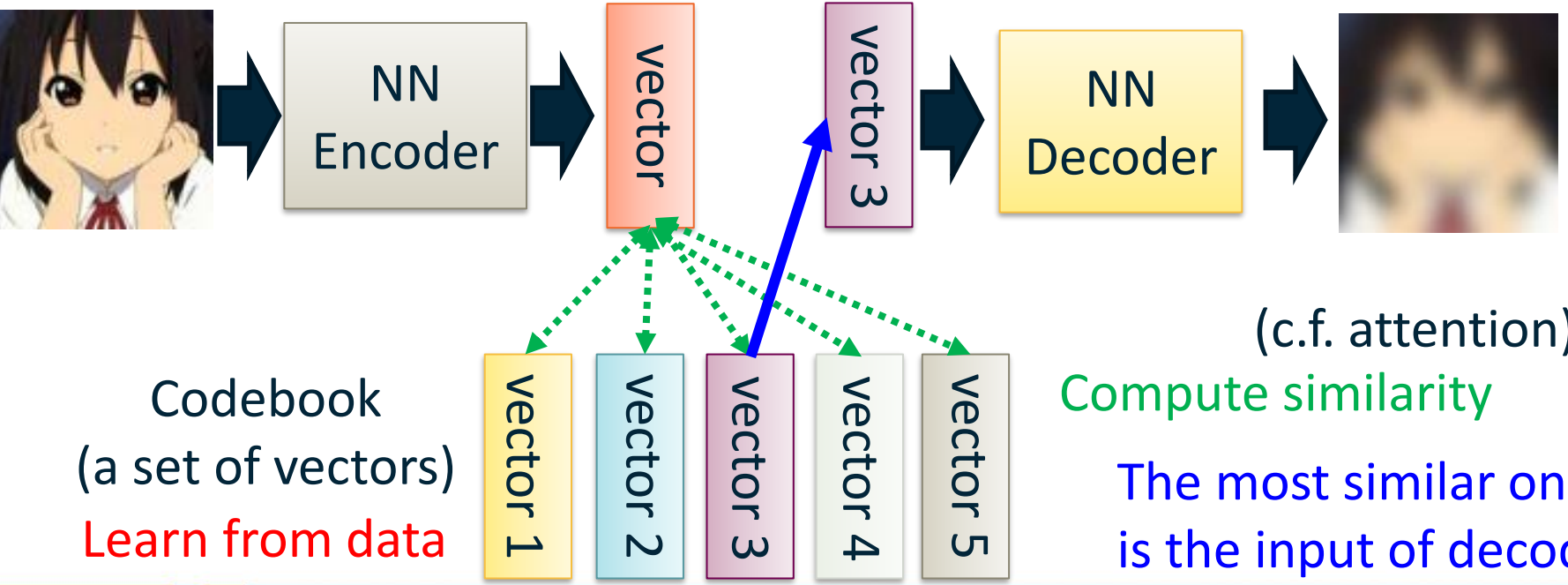


Slide by Hung-yi Lee

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

Discrete Representation

- Vector Quantized Variational Auto-encoder (VQVAE)



Codebook
(a set of vectors)
Learn from data

(c.f. attention)
Compute similarity

The most similar one
is the input of decoder.

- 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

Comparing the different generative models

Q. Which ones are VAEs good at?

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations			
Fast sampling			
High quality samples			

Slide by Krishna & Pratt

Comparing the different generative

VAEs are bad at generating high quality samples

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅		
Fast sampling	✅		
High quality samples	❌		

Slide by Krishna & Pratt

Comparing the different generative models

Q. Which ones are GANs good at?

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅		
Fast sampling	✅		
High quality samples	❌		

Slide by Krishna & Pratt

Comparing the different generative models

GANs suffer from mode collapse

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅	❌	
Fast sampling	✅	✅	
High quality samples	❌	✅	

Slide by Krishna & Pratt

Comparing the different generative models

Q. Which ones are Diffusion models good at?

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅	❌	
Fast sampling	✅	✅	
High quality samples	❌	✅	

Slide by Krishna & Pratt

Comparing the different generative models

Diffusion models are bad at sampling fast.

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✅	❌	✅
Fast sampling	✅	✅	❌
High quality samples	❌	✅	✅

Slide by Krishna & Pratt

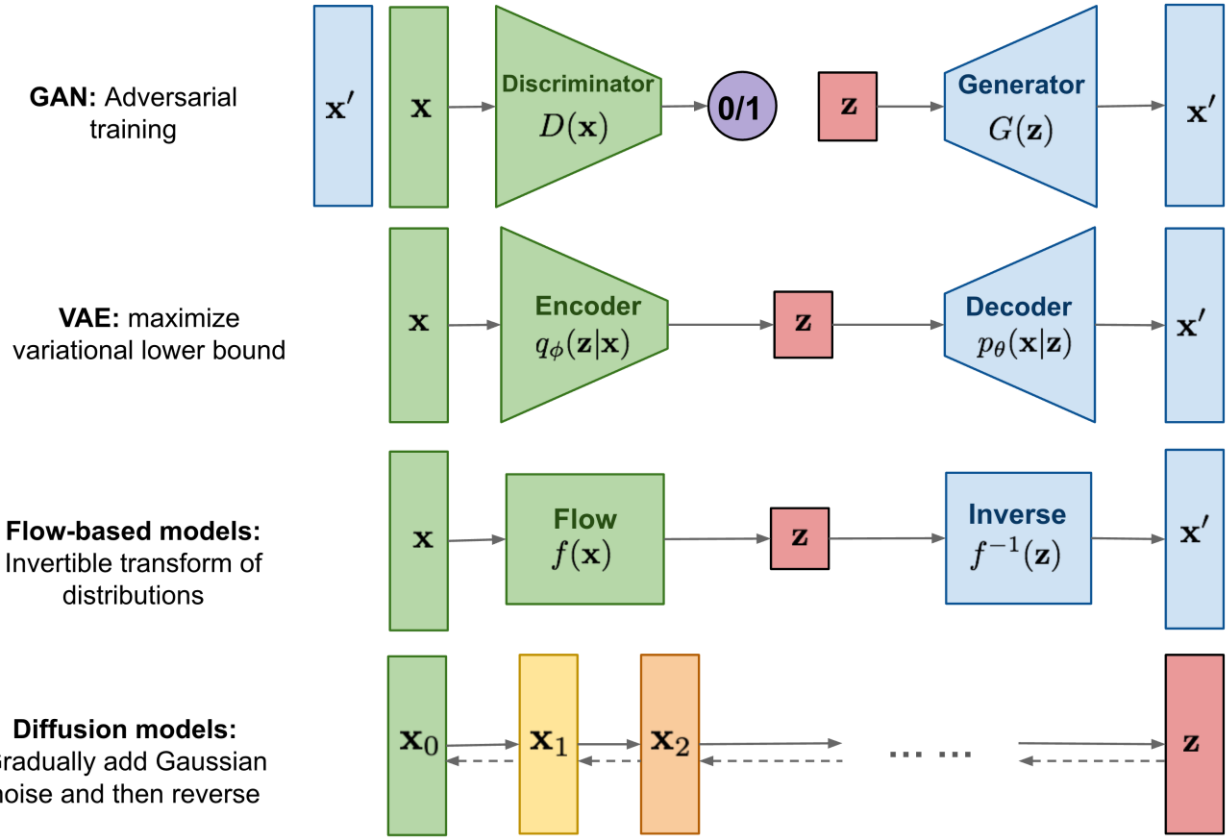
- Several ways to learn *generative* models via deep learning

- **Variational Autoencoders:**
  - Pro: Principled mathematical formulation
  - Pro: Results in disentangled latent representations
  - Con: Approximation inference, results in somewhat lower quality reconstructions
- **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)
- **Diffusion Models**
  - Pro: Great results and diversity!
  - Con: Slow generation (though lots of tricks to address)

Ha & Schmidhuber, World Models, 2018

Overall Summary

Comparison


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