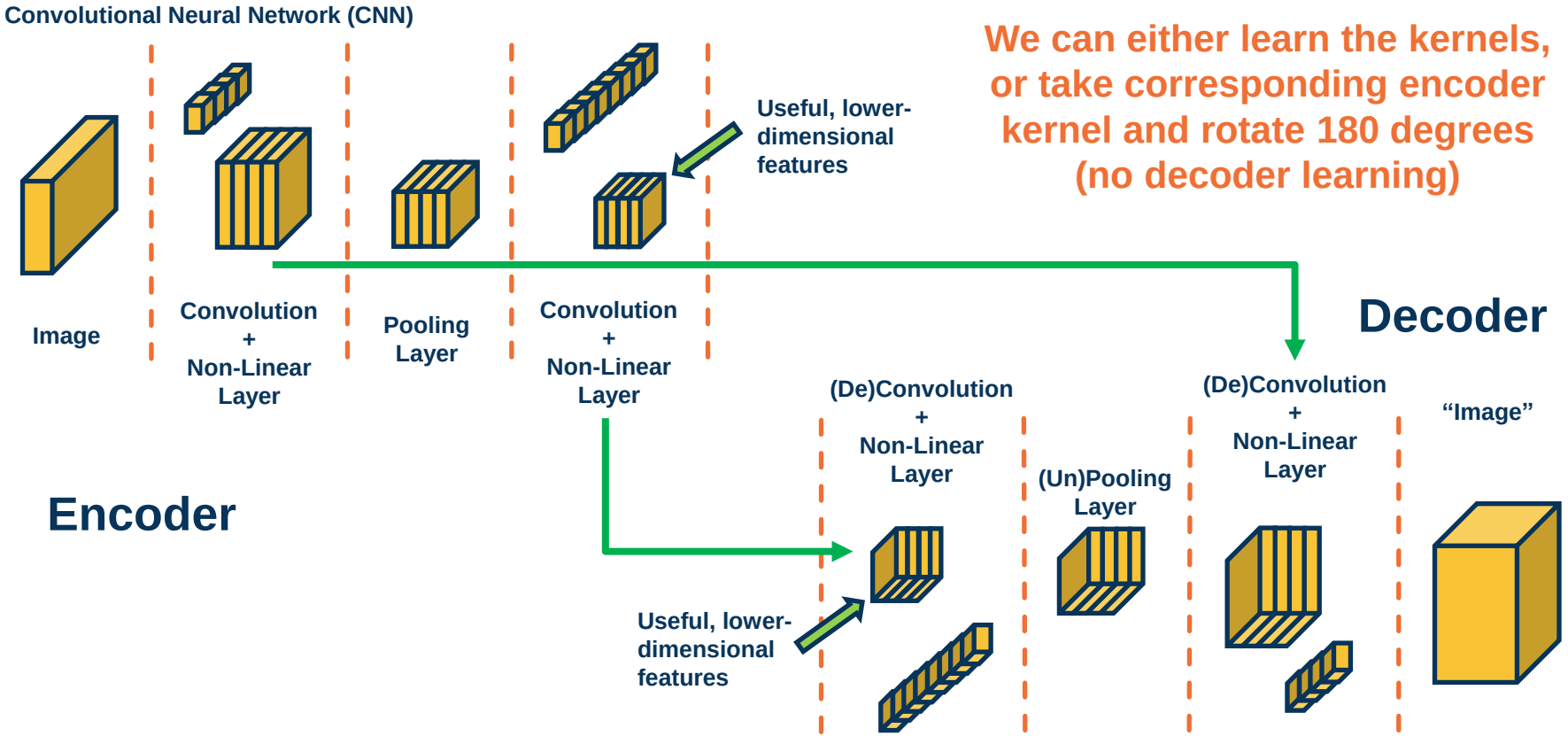
Topics:

- Generative Adversarial Networks

CS 4644-DL / 7643-A
ZSOLT KIRA

- **Assignment 3**
  - Due **March 8th 11:59pm EST**
- **Projects**
  - Project proposal due **March 14th**
- Next Meta office hours 03/14 3pm ET on machine translation

W9: Mar 3	Generative Models (Part I): Generative Adversarial Networks	• Generative Adversarial Networks ;
W9: Mar 5	Project Planning Session	
W10: Mar 10	Generative Models (Part II): Diffusion Models PS3/HW3 due Mar 8th 11:59pm (grace period Mar 10th), PS4/HW4 out (due Mar 30th)	
W10: Mar 12	Variational Autoencoders (VAEs) Project Check-in Due March 14th 11:59pm (grace period until March 16th)	• Tutorial on Variational Autoencoders
W11: Mar 17	Spring Break	
W11: Mar 19	Spring Break	

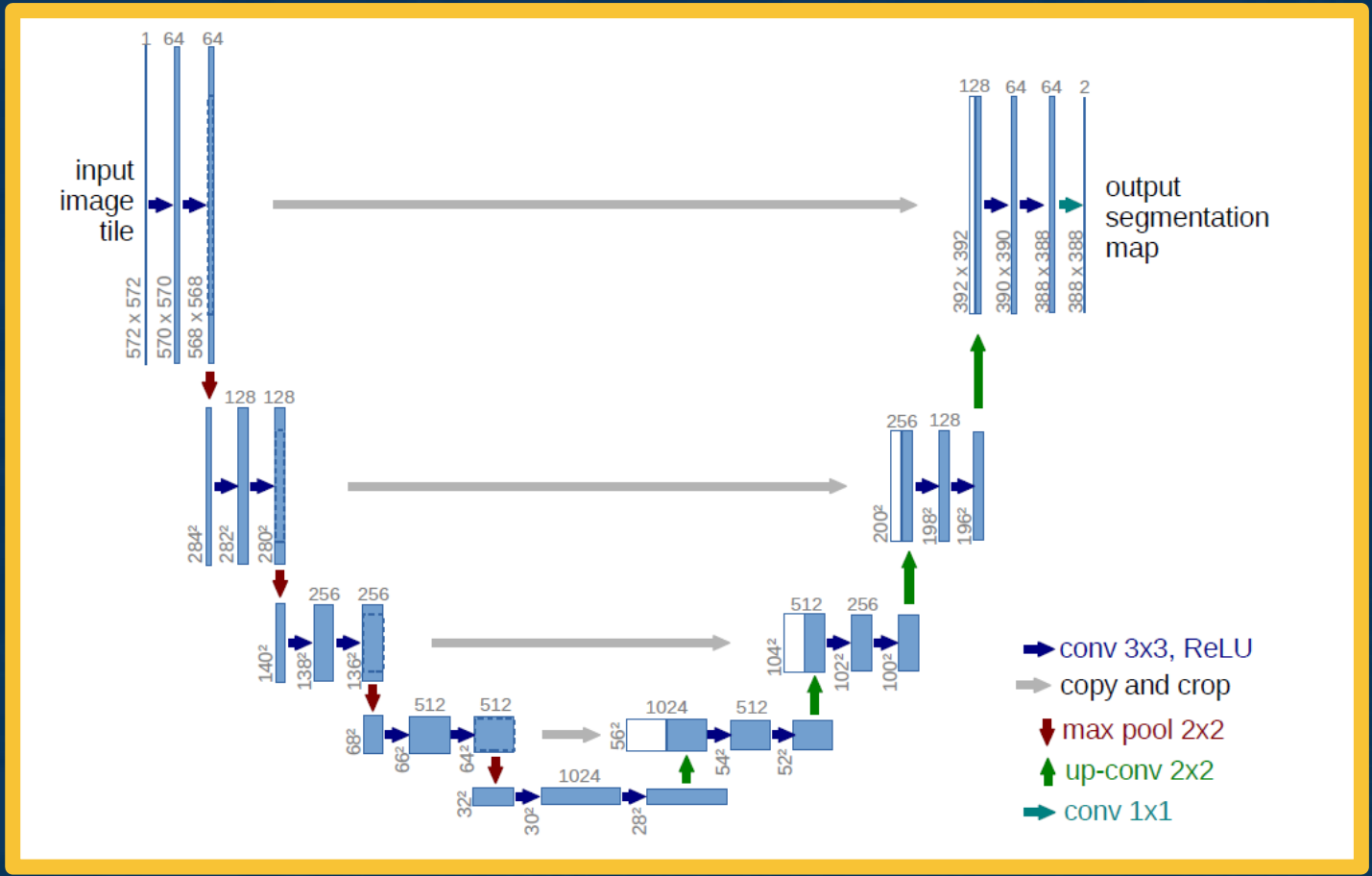


Symmetry in Encoder/Decoder

Georgia Tech

U-Net

You can have skip connections to bypass bottleneck!

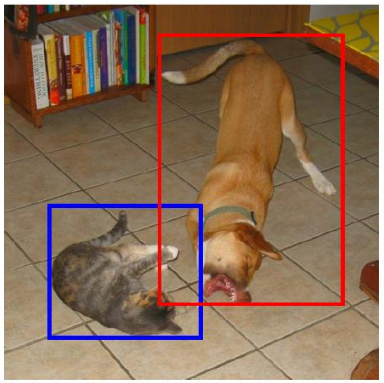


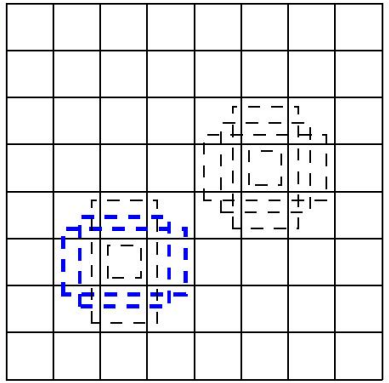
Ronneberger, et al., "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015

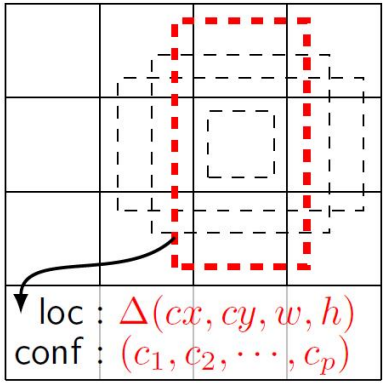
Georgia Tech

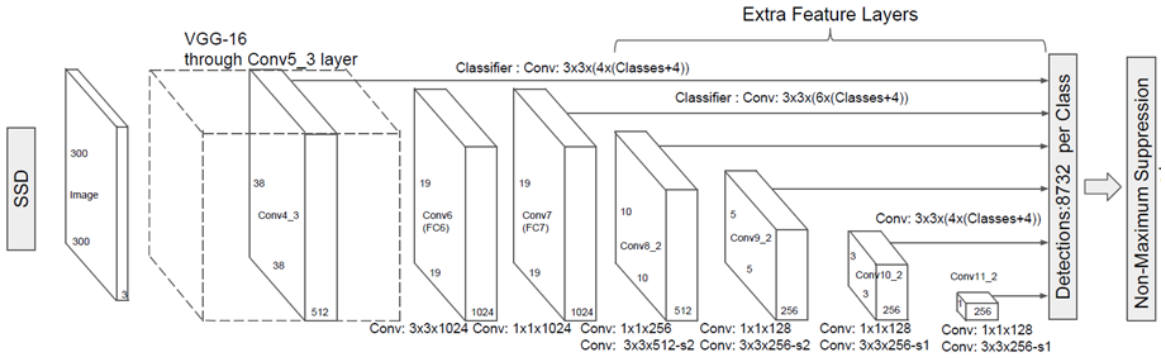
Single-shot detectors use an idea of **grids** as anchors, with different scales and aspect ratios around them

- Various tricks used to increase the resolution (decrease subsampling ratio)







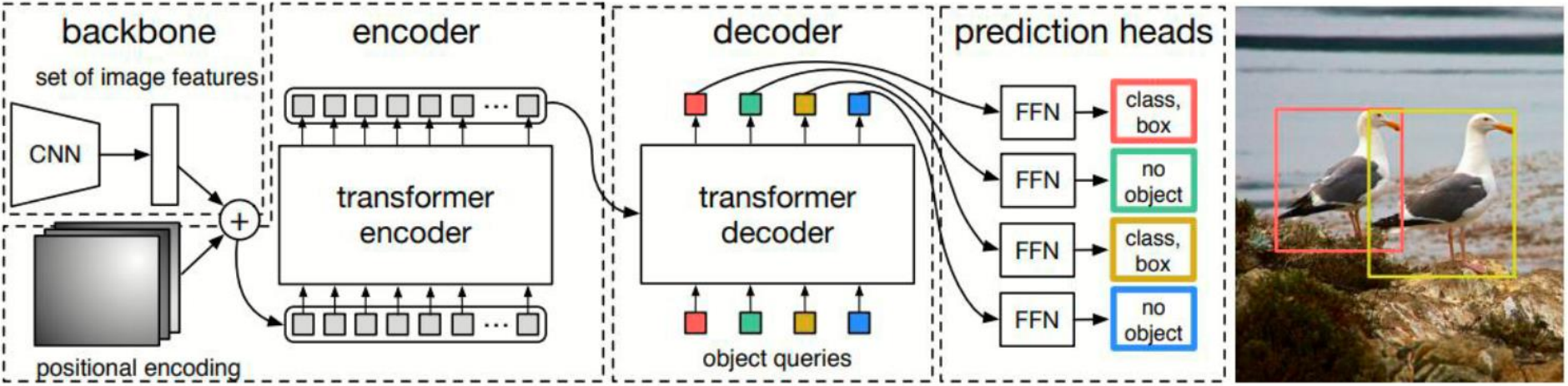


Liu, et al., "SSD: Single Shot MultiBox Detector", 2015

Single-Shot Detector (SSD)

Georgia Tech

DEtector TRansformer - DETR overview



Generative Models: Introduction

Supervised Learning

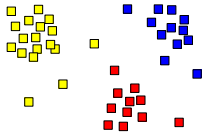
- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, $P(y|x)$
- e.g. classification

Sheep
Dog
Cat
Lion
Giraffe

Less Labels

Unsupervised Learning

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

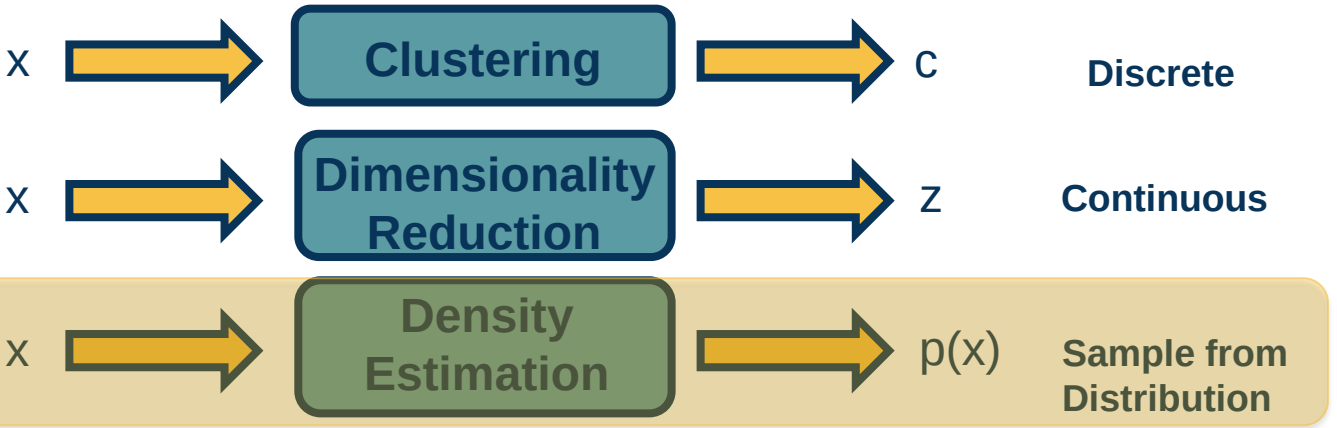


Spectrum of Low-Labeled Learning

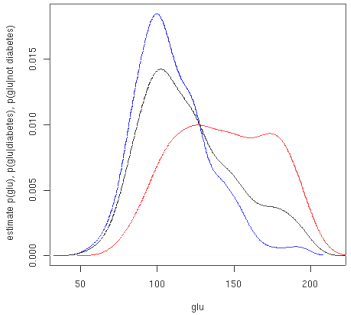
Georgia Tech

Supervised Learning

Unsupervised Learning



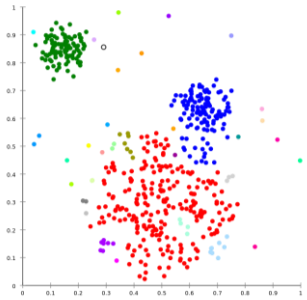
Traditional unsupervised learning methods:



Density estimation

Modeling $P(x)$

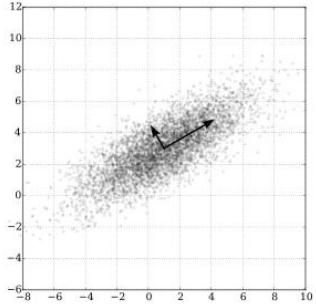
Deep Generative Models



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

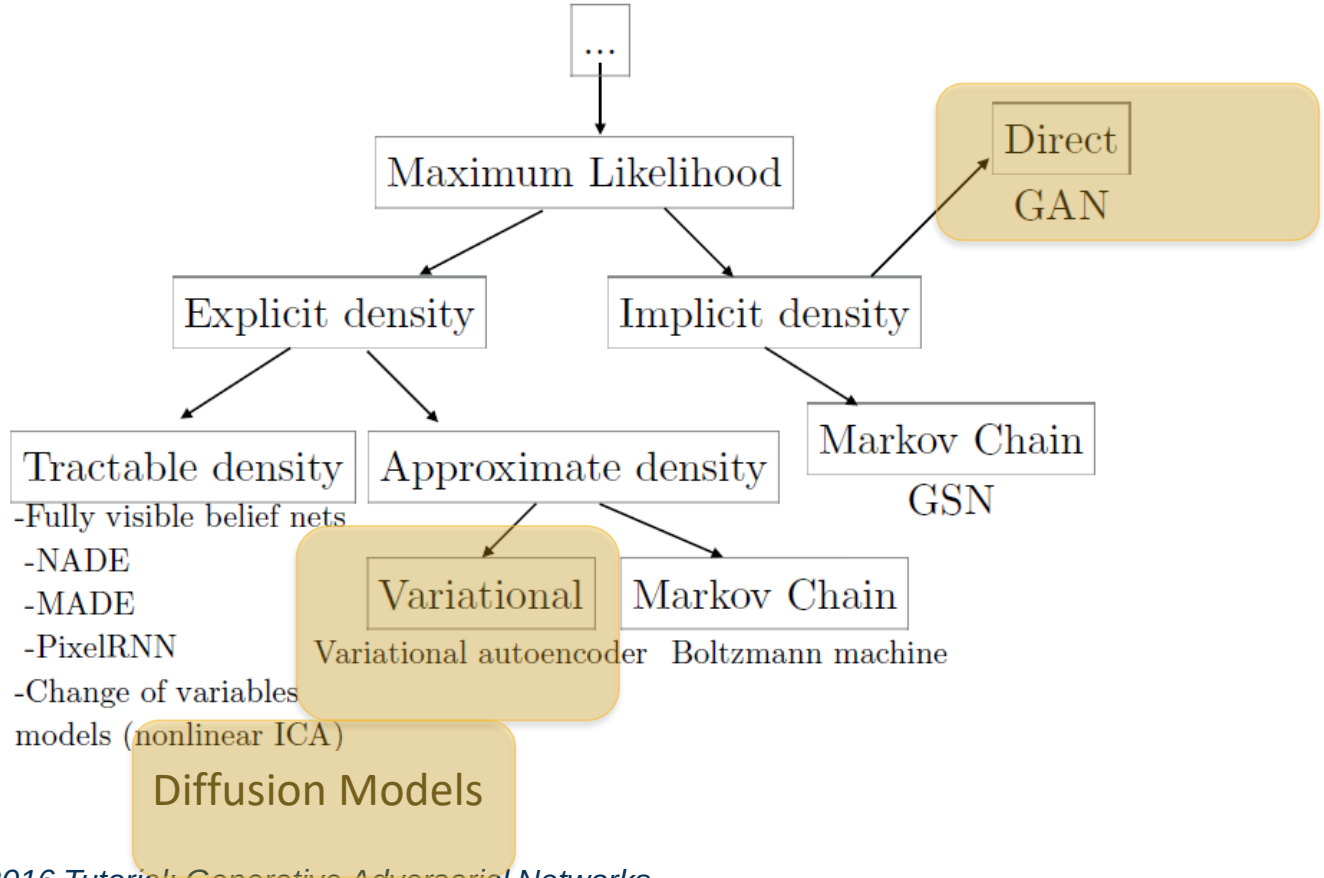
- Discriminative models model $P(y|x)$
  - Example: Model this via neural network, SVM, etc.

- 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 $\boldsymbol{\mu}, \boldsymbol{\sigma}$

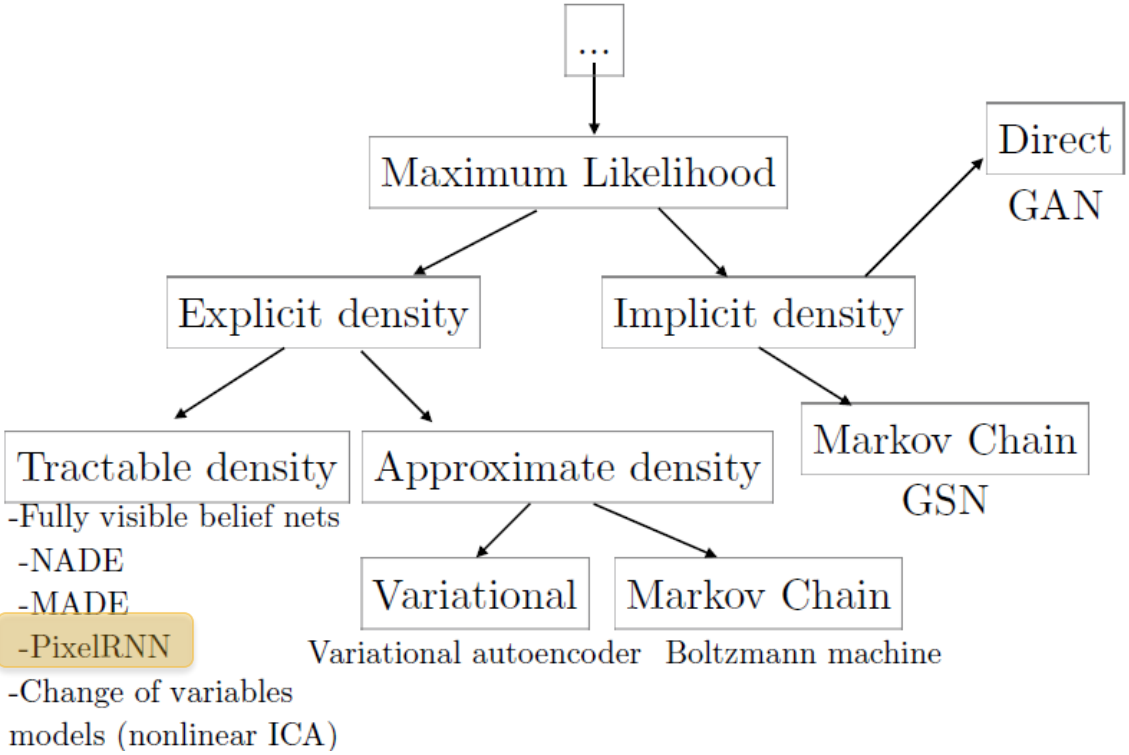
Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

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!

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

Same as language modeling!

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

i: next word; $w_{i-1}, \ldots, w_1$: history

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

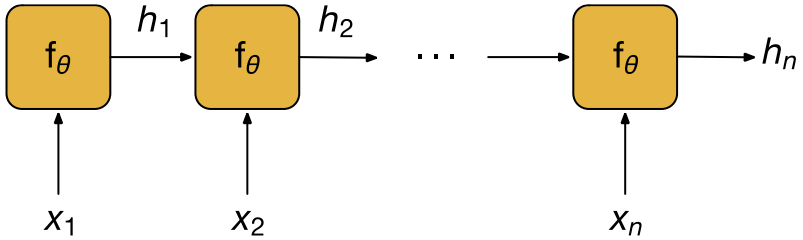
Oord et al., Pixel Recurrent Neural Networks

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

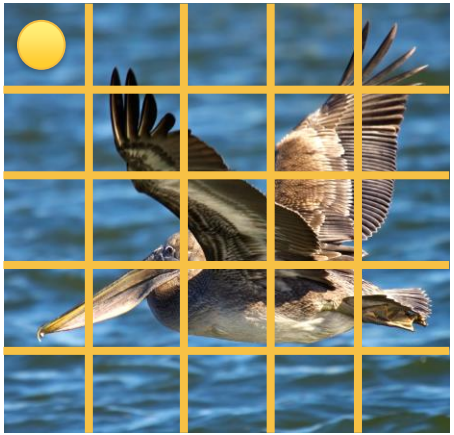
$$\mathrm{p}(\boldsymbol{s}) = \mathrm{p}(w_1, w_2, \ldots, w_n) = \prod_i \mathrm{p}(w_i \mid w_{i-1}, \ldots, w_1)$$

i: next word; $w_{i-1}, \ldots, w_1$: history

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



Language Models as an RNN



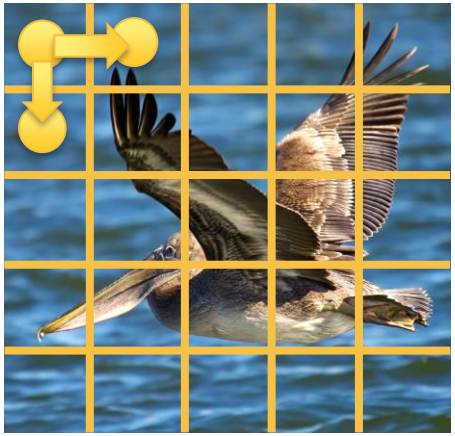
$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

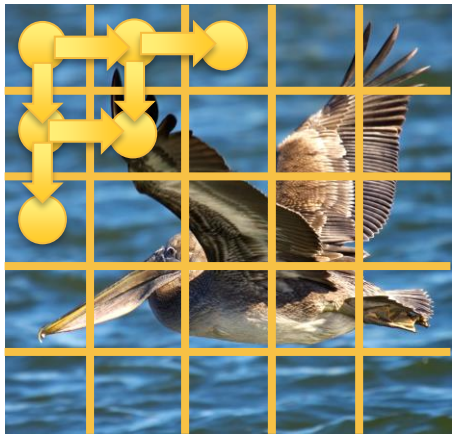
$$p(x) = p(x_1) \prod_{i=2}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

1. Choose ordering (upper left, top to bottom, left to right.

Separate out pixel 1

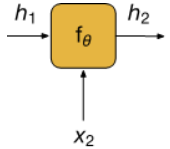
Oord et al., Pixel Recurrent Neural Networks





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

- Model this as RNN with parameters

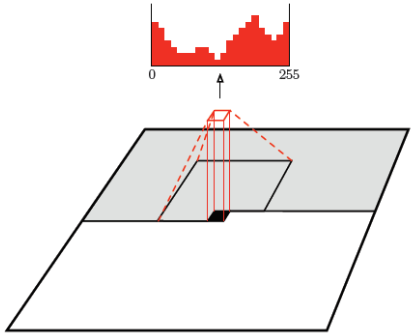


- Training:
  - We can train similar to language models:
  - Maximum likelihood approach

Downsides?

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

Oord et al., Pixel Recurrent Neural Networks



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!
  - Because of spatial locality in images
- 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)

Georgia Tech

Geyser

Hartebeest

Grey whale

Tiger

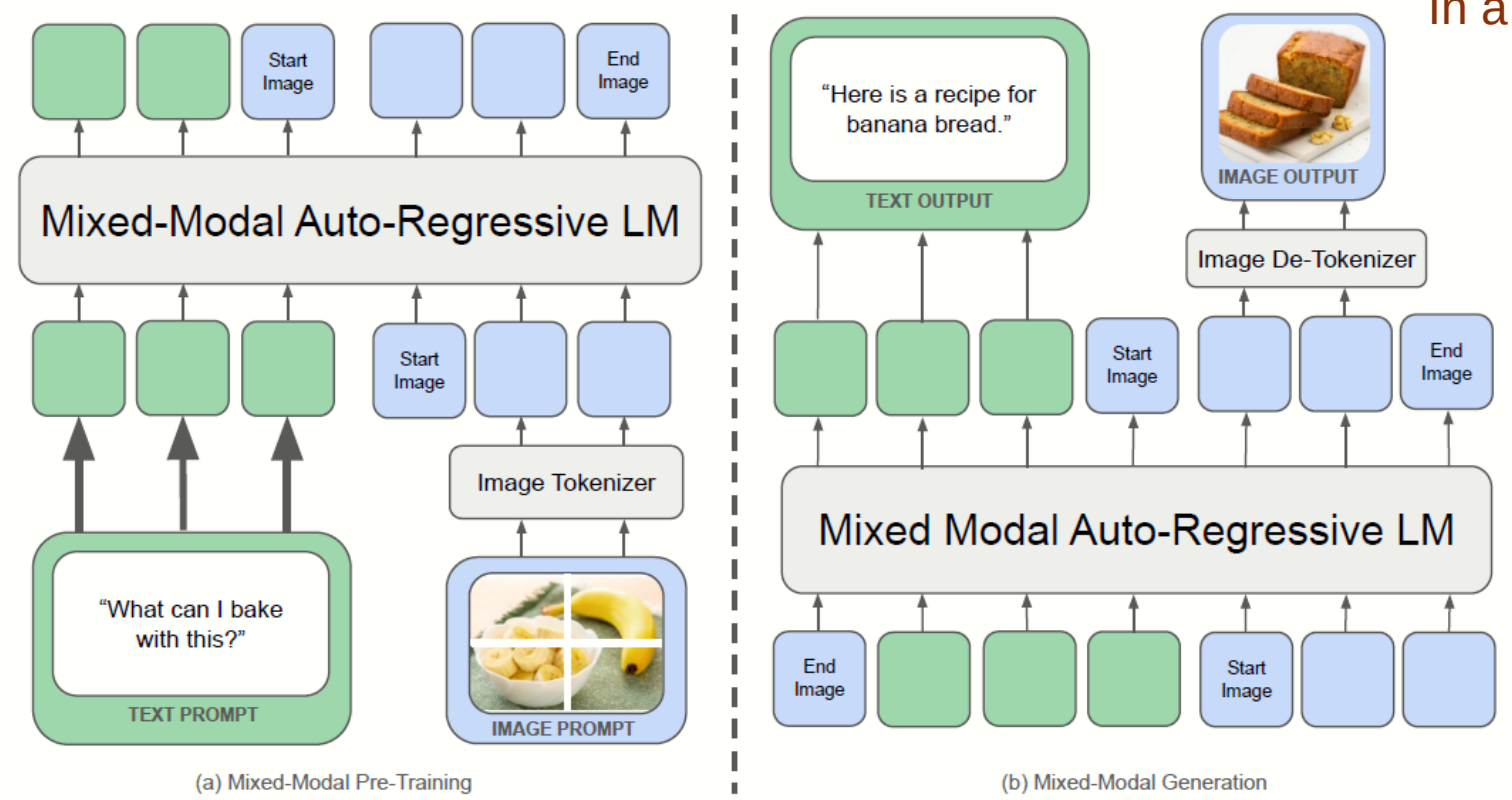
Can we update this to modern times?

Oord et al., Conditional Image Generation with PixelCNN Decoders

Example Images (PixelCNN)

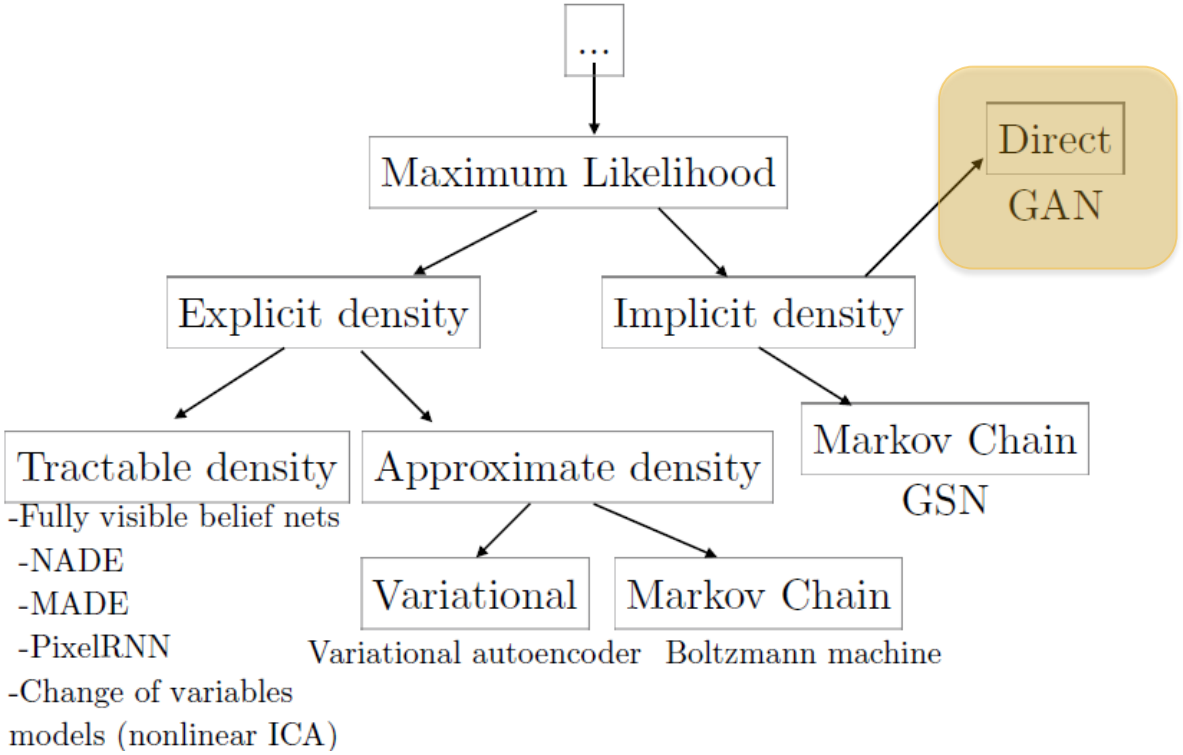
Georgia Tech

In a few weeks



Chameleon: Mixed-Modal Early-Fusion Foundation Models

Generative Adversarial Networks (GANs)



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

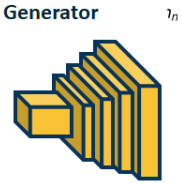
- *Implicit* generative models do not actually learn an explicit model for $\boldsymbol{p(x)}$

- Instead, learn to *generate samples* from $\boldsymbol{p(x)}$
  - Learn good feature representations
  - Perform data augmentation
  - Learn world models (a simulator!) for reinforcement learning

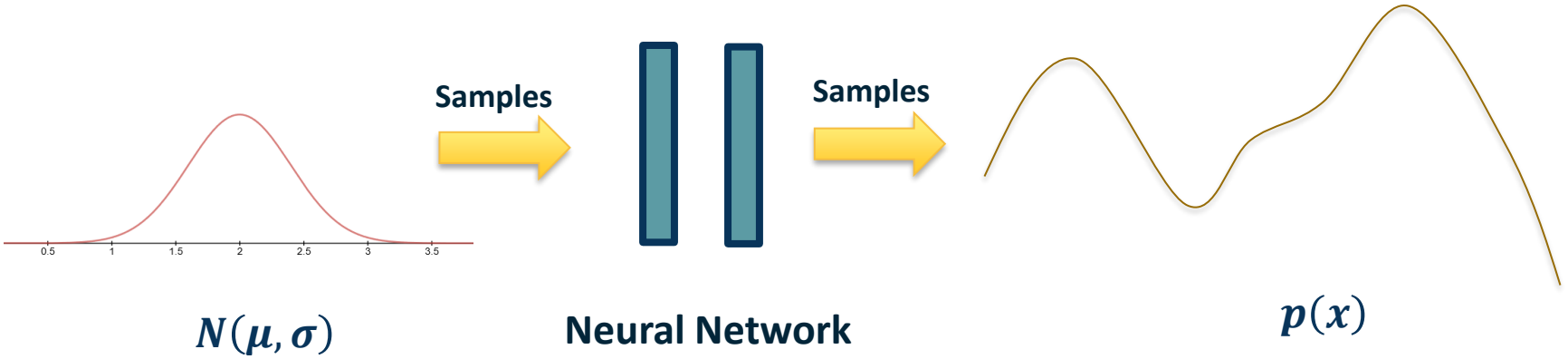
- How?
  - **Decode architecture**
  - **Learn to sample** from a neural network output

What architecture lets us generate images?

How do we generate a different image every time?

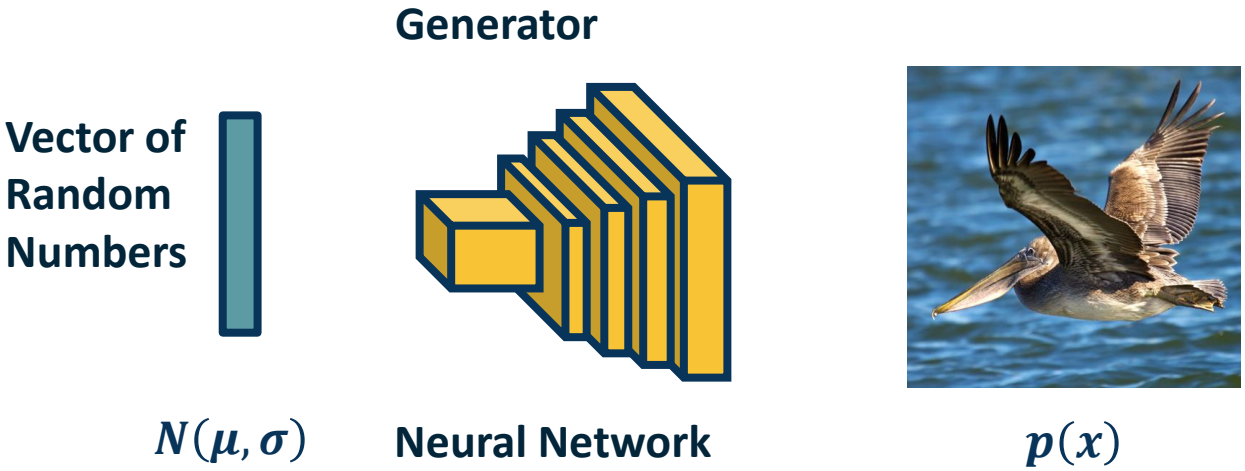


- We would like to *sample* from $\boldsymbol{p(x)}$ using a neural network
- **Idea:**
  - Sample from a simple distribution (Gaussian)
  - Transform the sample to $\boldsymbol{p(x)}$



Learning to Sample

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

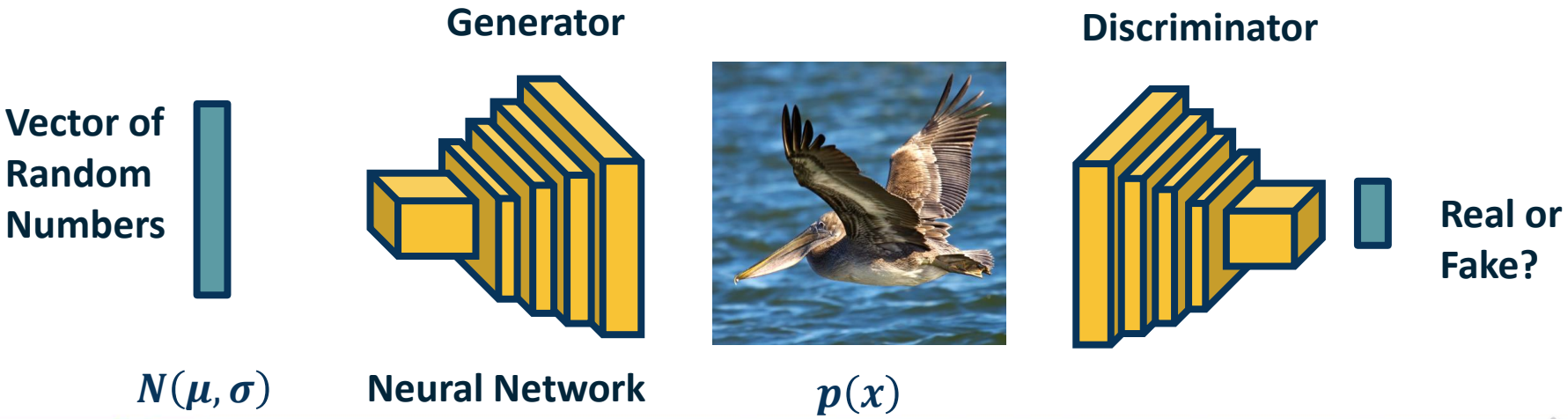


How do we train this (loss)?

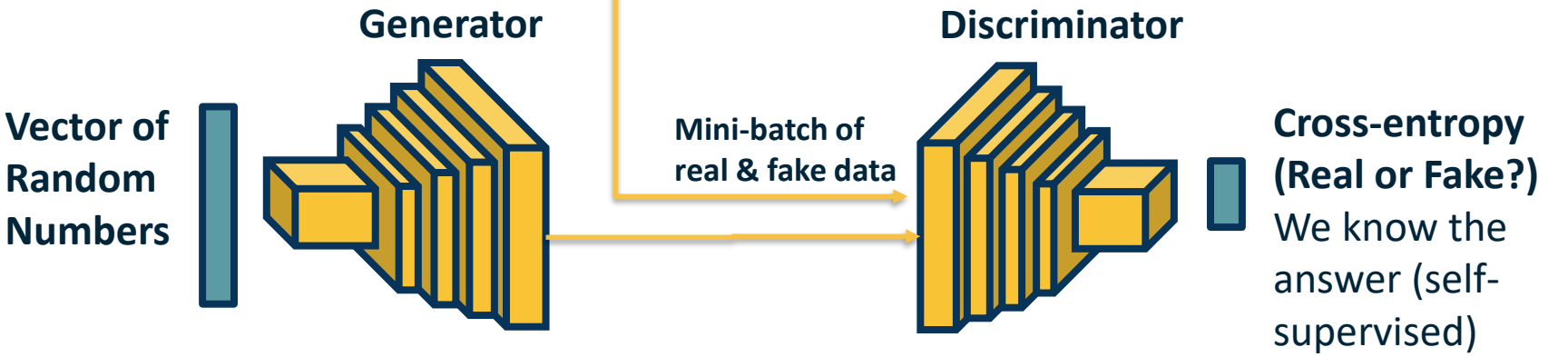
- Instead, learn to *generate samples* from $\boldsymbol{p(x)}$

- How?
  - **Adversarial training** that uses one network's predictions to train the other (dynamic loss function!)
  - **Lots of tricks** to make the optimization more stable

- **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
    - Can be seen as a dynamic (adversarial) loss!



- **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)?

- Since we have two networks competing, this is a mini-max two player game
  - Ties to game theory
  - Not clear what (even local) Nash equilibria are for this game

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Mini-max Two Player Game

- Since we have two networks competing, this is a mini-max two player game
  - Ties to game theory
  - Not clear what (even local) Nash equilibria are for this game

- The full mini-max objective is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Sample from real **Sample from fake**

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- where $D(x)$ is the discriminator outputs probability ([0,1]) of **real** image
- x is a **real image** and $G(z)$ is a **generated** image

- The discriminator wants to **maximize** this:
  - $D(x)$ is pushed up (to 1) because x is a real image
  - $1 - D\big(G(z)\big)$ is also pushed up to 1 (so that $\mathrm{D}\big(\mathrm{G}(\mathrm{z})\big)$ is pushed down to 0)
  - In other words, discriminator wants to classify real images as real (1) and fake images as fake (0)

Discriminator Perspective

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- where $D(x)$ is the discriminator outputs probability ([0,1]) of **real** image
- x is a **real image** and $G(z)$ is a **generated** image

- The generator wants to **minimize** this:
  - First term: G(..) doesn't appear in it!
  - $1 - D\big(G(z)\big)$ is pushed down to 0 (so that $\mathrm{D}\big(\mathrm{G}(\mathrm{z})\big)$ is pushed up to 1)
  - This means that the generator is **fooling** the discriminator, i.e. succeeding at generating images that the discriminator can't discriminate from real

Generator Perspective

- Since we have two networks competing, this is a mini-max two player game
  - Ties to game theory
  - Not clear what (even local) Nash equilibria are for this game

- The full mini-max objective is:

Sample from fake

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Generator *minimizes*

How well discriminator does (0 for fake)

  - where $D(x)$ is the discriminator outputs probability ([0,1]) of **real** image
  - x is a **real image** and $G(z)$ is a **generated** image

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

- Since we have two networks competing, this is a mini-max two player game
  - Ties to game theory
  - Not clear what (even local) Nash equilibria are for this game

- The full mini-max objective is:

Sample from real **Sample from fake**

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Discriminator *maximizes*

How well discriminator does (1 for real)

How well discriminator does (0 for fake)

  - where $D(x)$ is the discriminator outputs probability ([0,1]) of **real** image
  - x is a **real image** and $G(z)$ is a **generated** image

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generator

Discriminator

Vector of Random Numbers

Mini-batch of real & fake data

Cross-entropy (Real or Fake?)
We know the answer (self-supervised)

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

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

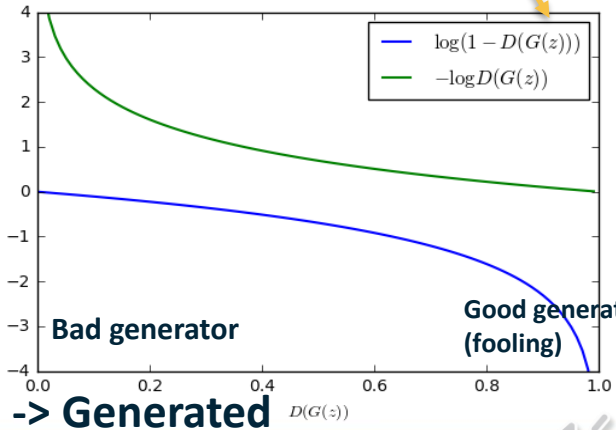
Generative Adversarial Networks (GANs)

Georgia Tech

- The generator part of the objective does not have good gradient properties

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

  - High gradient when $D(G(z))$ is high (that is, discriminator is wrong)
  - We want it to improve when samples are *bad* (discriminator is right)



- Alternative objective, ***maximize***:

$$\max_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Plot from CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Converting to Max-Max Game

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

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

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

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

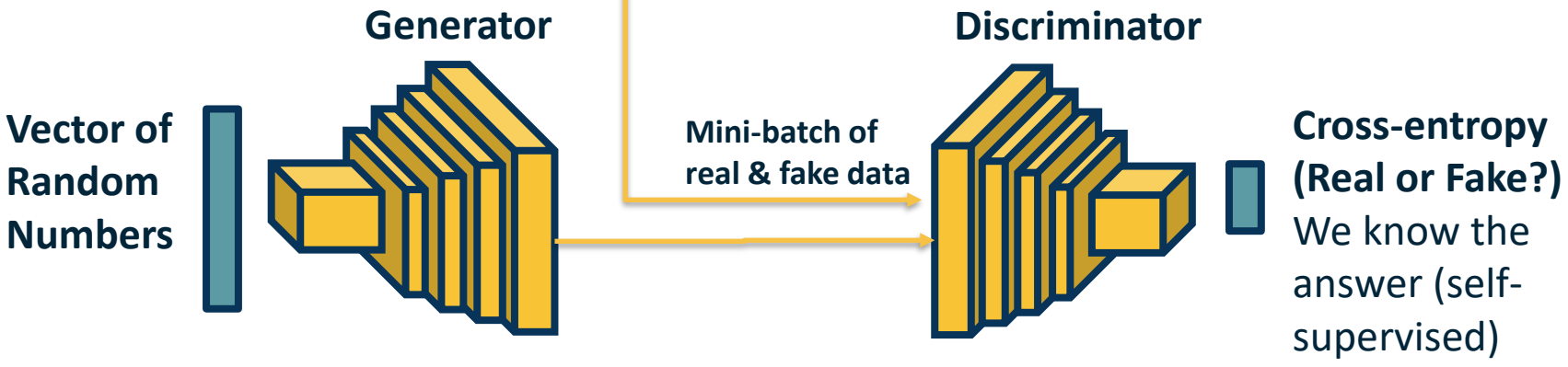
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

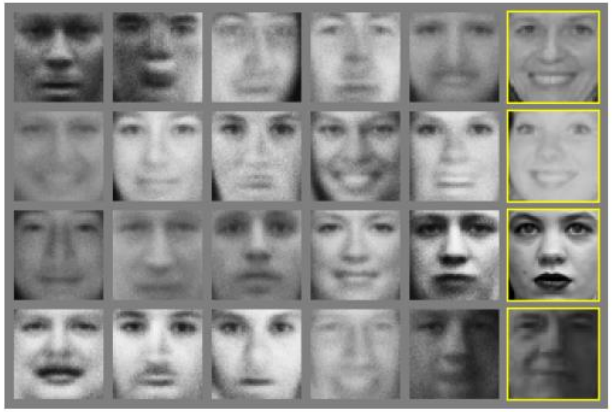
Goodfellow, NeurIPS 2016 Generative Adversarial Nets

Final Algorithm

- At the end, we have:
  - An *implicit* generative model!
  - Features from discriminator

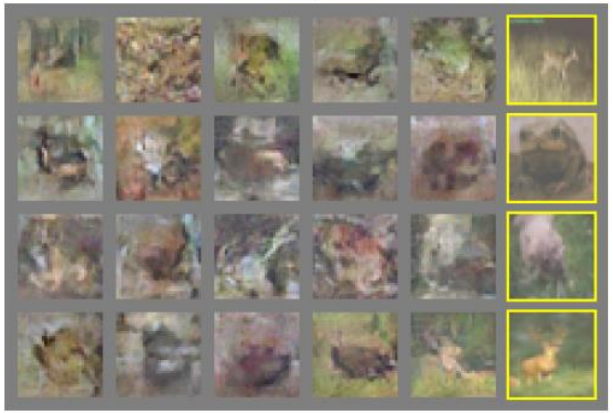


a)



b)

c)



d)

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

Early Results

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

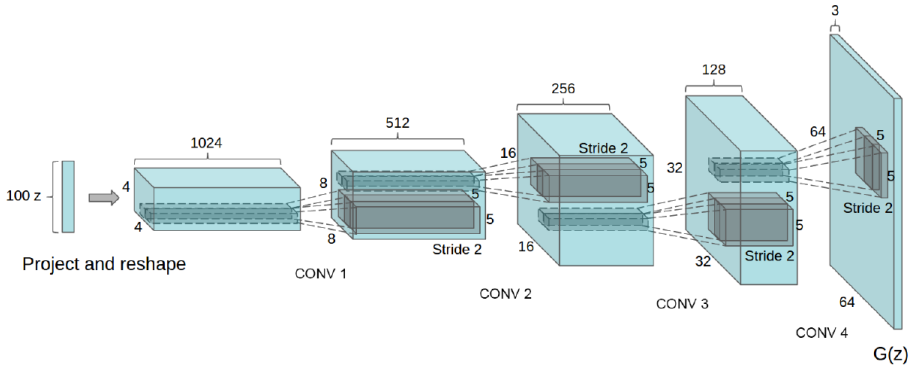
- Advancements include:
  - More stable architectures
  - Regularization methods to improve optimization
  - Progressive growing/training and scaling

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

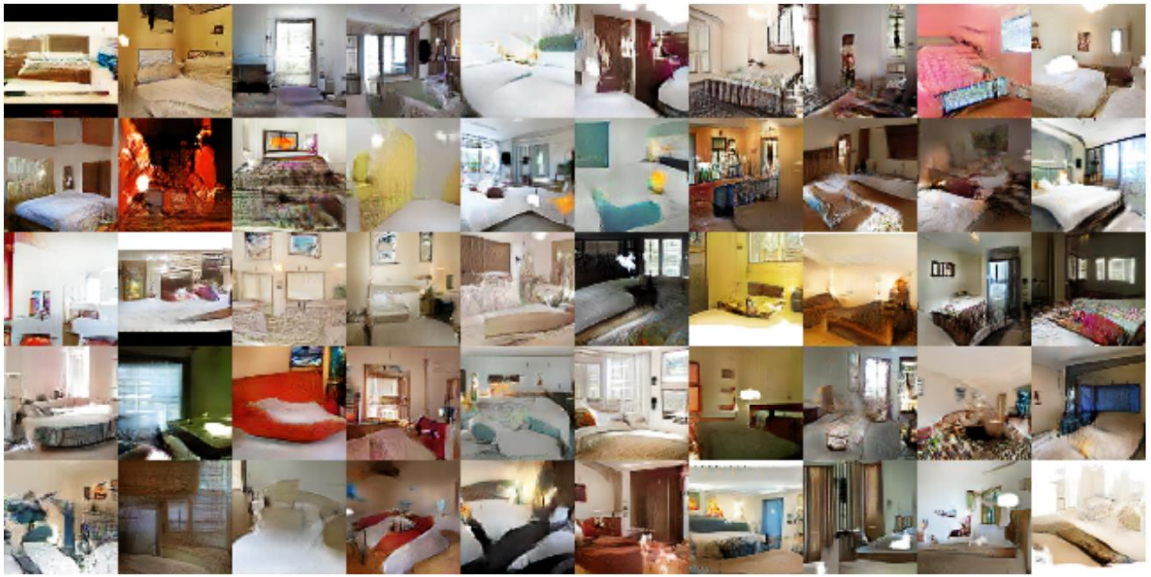
- Several theoretically-motivated regularization methods
  - Simple example: Add noise to real samples!

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

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

Generative Adversarial Nets: Convolutional Architectures

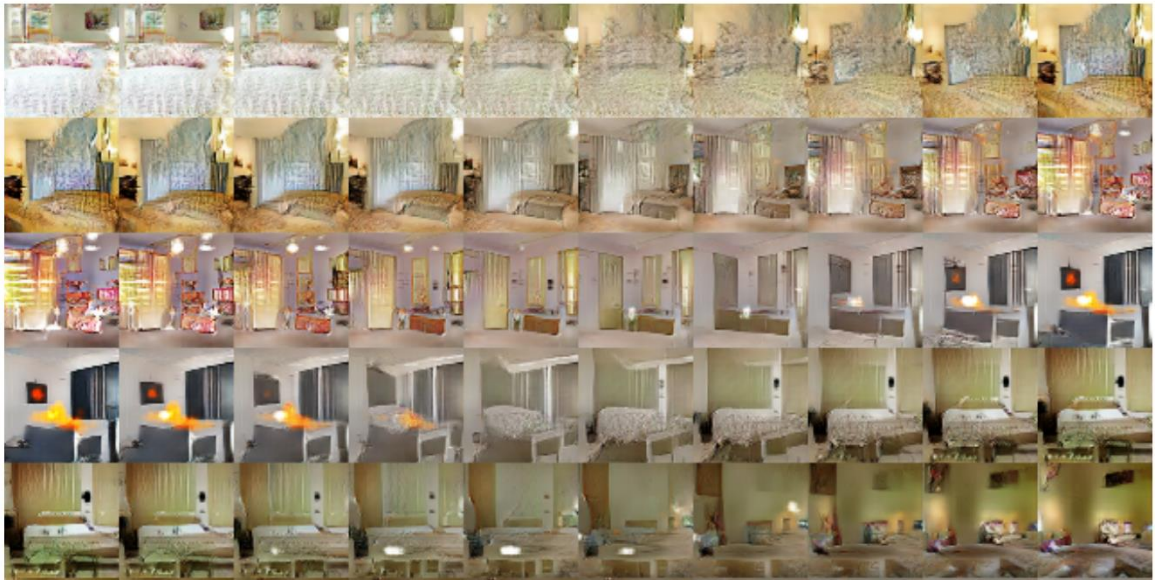
Samples from the model look much better!



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

Georgia Tech

(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

Georgia Tech

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

Video Generation

Georgia Tech

- 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

Comparison of Methods