CS 4803 / 7643: Deep Learning

Topics:

- Generative Adversarial Networks (GANs)
- Closing time

Dhruv Batra
Georgia Tech

Administrativia

- Last class today

- Project submission
  - Due: 11/24, 11:59pm
  - Last deliverable in the class
  - ~~Can't use late days~~ 8 free late days
  - https://www.cc.gatech.edu/classes/AY2021/cs7643_fall/

Generative Adversarial Networks (GAN)

Types of Learning

- Supervised learning $x \to \boxed{y}$ $L(\hat{y}, y)$
  - Learning from a "teacher"
  - Training data includes desired outputs

- Reinforcement learning $r(s_t, \boxed{a_t}, s_{t+1})$
  - Learning to act under evaluative feedback (rewards)

- Unsupervised learning
  - Discover structure in data
  - Training data does not include desired outputs

Taxonomy of Generative Models

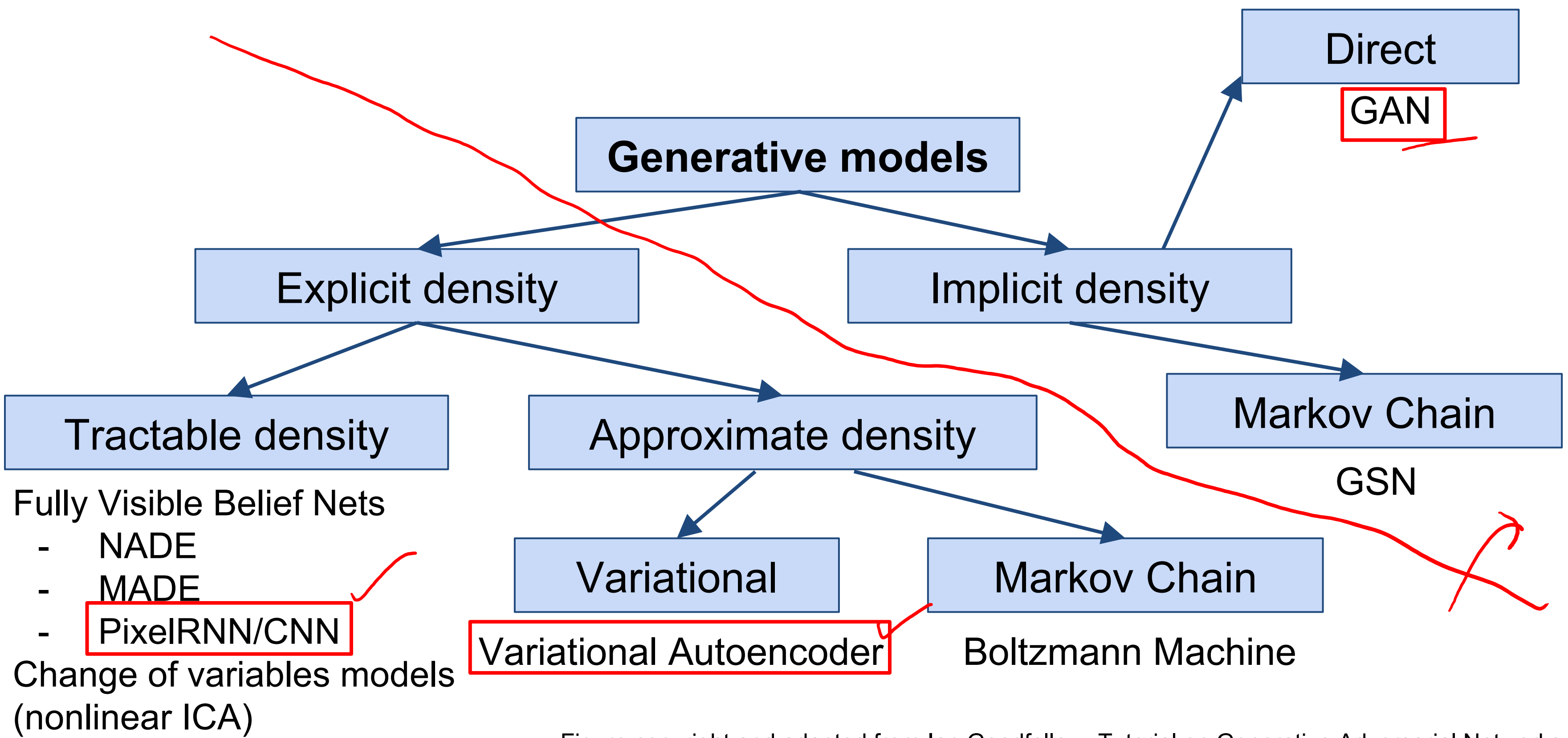


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$\boxed{p_\theta(x)} = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

$P_\theta(\vec{x})$

VAEs define intractable density function with latent **z**:

$$\boxed{p_\theta(x)} = \int p_\theta(z) p_\theta(x|z) dz$$

$P_\theta(\vec{x}, \vec{z})$

unobserved / latent

Cannot optimize directly, derive and optimize lower bound on likelihood instead

So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent **z**:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead

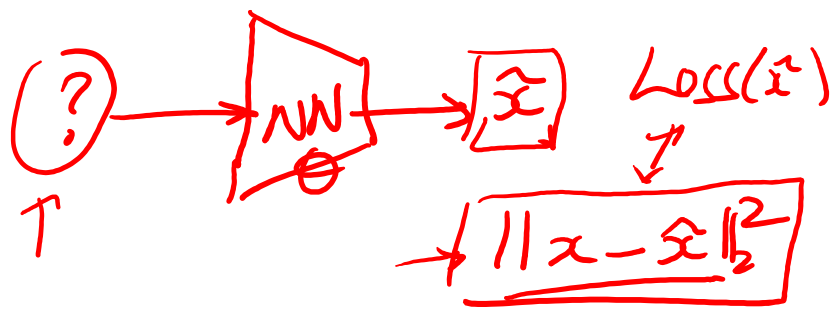
What if we give up on explicitly modeling density, and just want ability to sample?

$\tilde{x} \sim \boxed{P_\theta(x)}$

$\sim \underline{P_{data}(x)}$

$D = \{\vec{x}\}$

$\tilde{x} \sim \boxed{\text{Black box}}$



So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent **z**:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead

What if we give up on explicitly modeling density, and just want ability to sample?

GANs: don't work with any explicit density function!

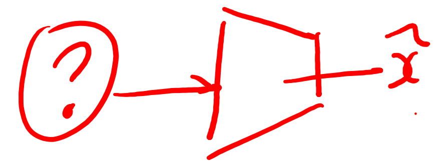
$[\underline{P_\theta(\vec{x})}]$ ×

$\underline{P_\theta(x_i | x_j)}$ ×

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Generative Adversarial Networks (GANs)

GANs are a combination of the following ideas:



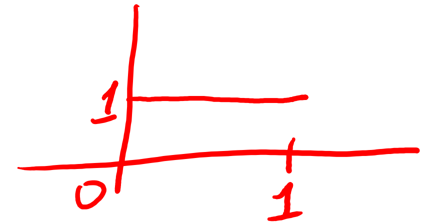
1. Learning to Sample
   - (High-level) Connection to Inverse Transform Sampling

2. Adversarial Training

$L(\, , \,)$

Easy Interview Question

- I give you $\boxed{u \sim U(0,1)}$
- Use u to produce a sample $x \sim \text{Bern}(\theta)$



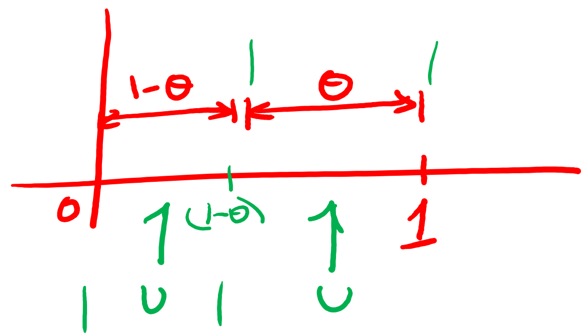
$Pr(u \in (a,b)) = b - a$

$0 \le a \le b \le 1$

$Pr(x=1) = \theta$

$Pr(x=0) = 1-\theta$

```
def Unif2Bern(u)
    if u > (1-θ)
        ret 1
    else
        ret 0
```

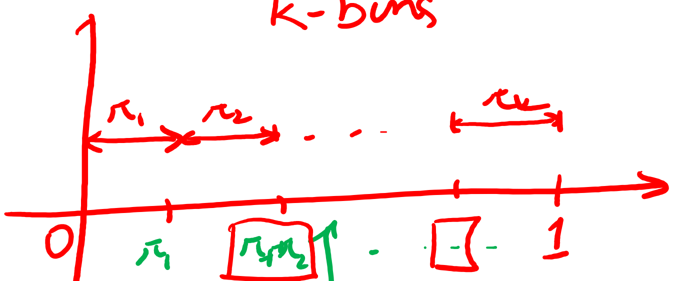


Slightly Harder Interview Question

- I give you $u \sim U(0,1)$
- Use u to produce a sample $x \sim \text{Cat}(\vec{\pi})$

k-way $\begin{bmatrix} \pi_1 \\ \vdots \\ \pi_k \end{bmatrix}$ $\quad \pi_i \geq 0 \quad \sum_{i=1}^{k} \pi_i = 1$

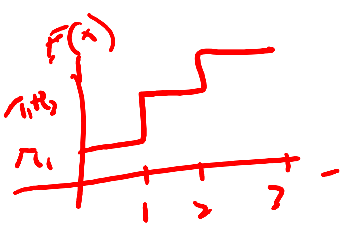
k-bins



Given u
return j s.t. $\boxed{\sum_{i=1}^{j-1} \pi_i} \leq u < \boxed{\sum_{i=1}^{j} \pi_i}$ ← ?

CDF of x

$Pr(X \leq t)$



$\boxed{F^{-1}(u)}$

Slightly Harder Interview Question

- I give you $u \sim U(0,1)$

- Use u to produce a sample $x \sim Cat(\pi)$

Harder Interview Question

- I give you $\boxed{u \sim U(0,1)}$

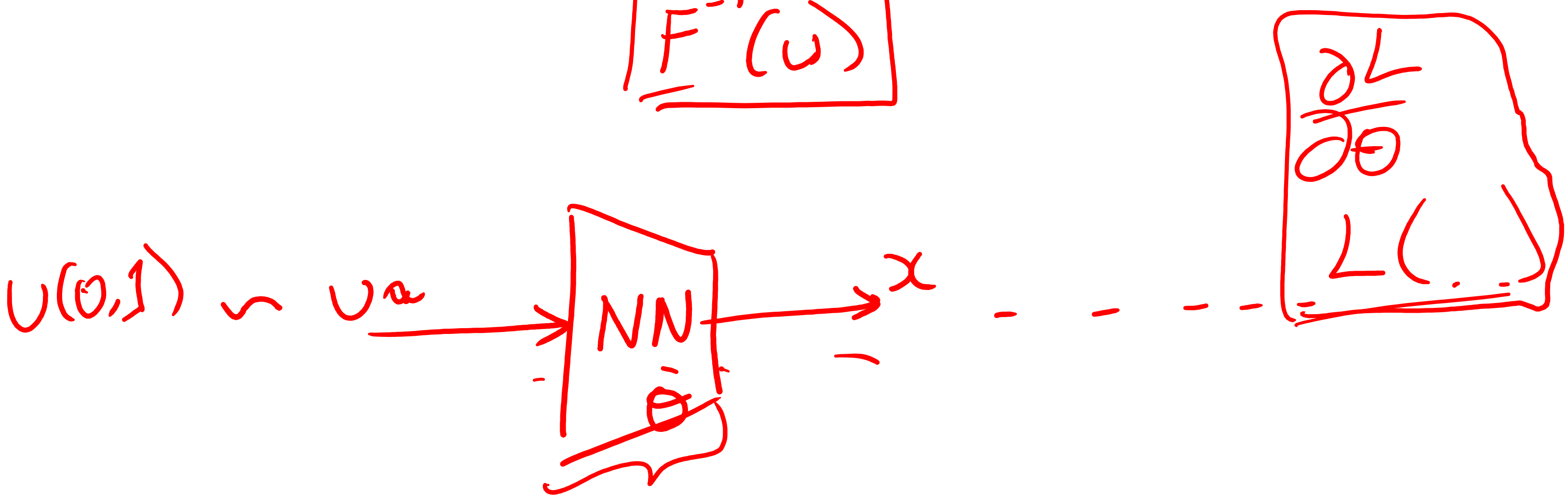
- Use u to produce a sample $\boxed{x \sim F_X(x)}$

return $\boxed{F^{-1}(u)}$

Harder Interview Question

- I give you u ~ U(0,1)
- Use u to produce a sample x ~ $F_X(x)$

$F^{-1}(u)$



Generative Adversarial Networks

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

Generative Adversarial Networks

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

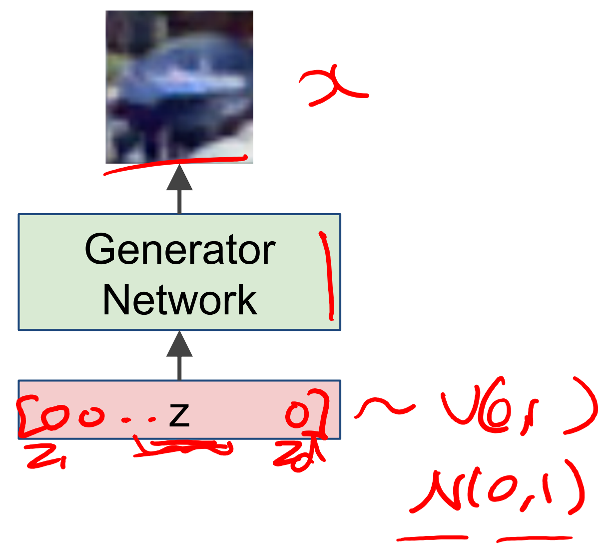
Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

A: A neural network!

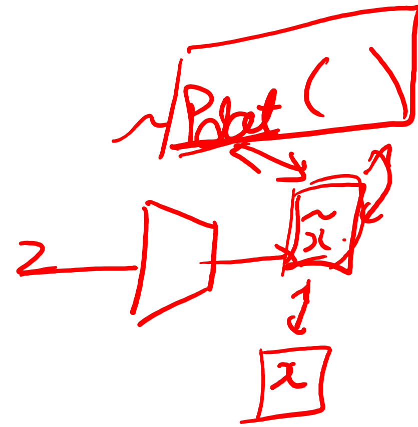
Output: Sample from training distribution



Input: Random noise

Generative Adversarial Networks (GANs)

GANs are a combination of the following ideas:



1. Learning to Sample
   - Connection to Inverse Transform Sampling

2. Adversarial Training

L(,)

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

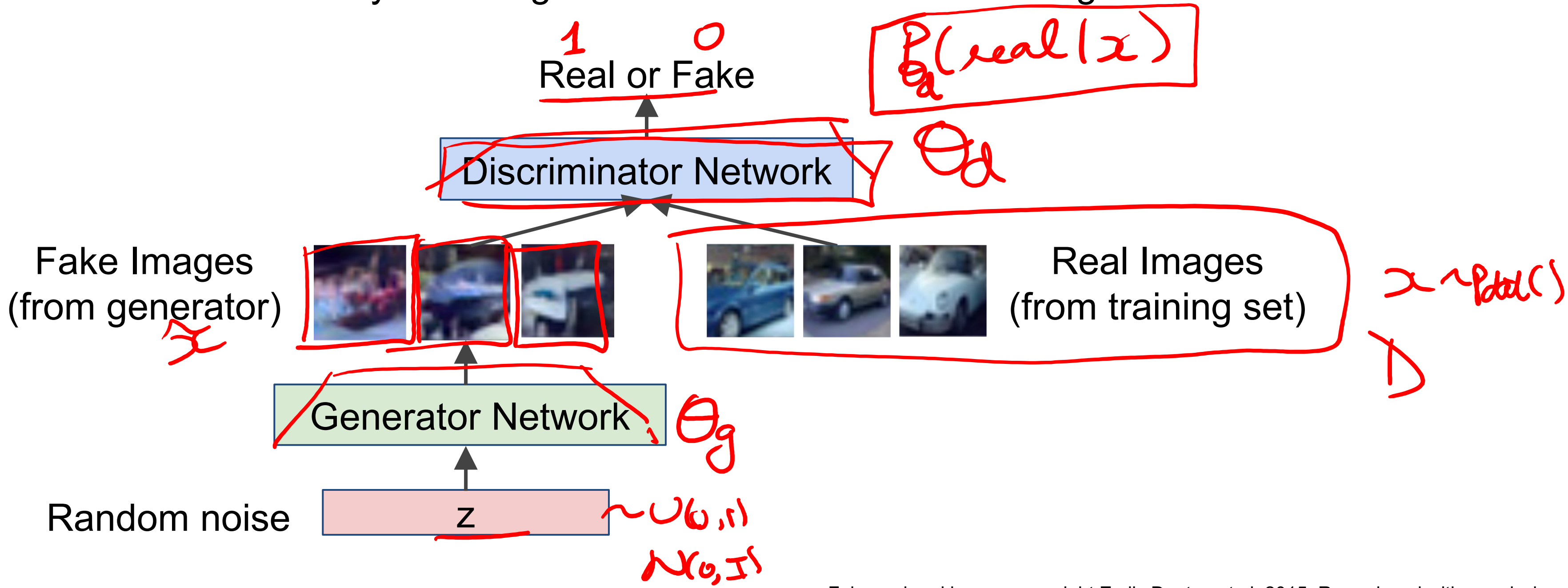
Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

generate

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

↑ $P_{\theta_d}(real|x)$ $P(x)$ $P(y|x)$

Minimax objective function:

$$\boxed{\min_{\theta_g}} \boxed{\max_{\theta_d}} \left[\mathbb{E}_{\boxed{x \sim p_{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{\boxed{z \sim p(z)}} \log(1 - D_{\theta_d}(\overbrace{G_{\theta_g}(z)}^{x})) \right]$$

$U(0,1)$

$N(0,I)$

$P(real|x)$

$\approx \frac{1}{N_1} \sum_{x_i} \log(\quad)$

$\approx \frac{1}{N_2} \sum_{z_i} (\quad)$

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

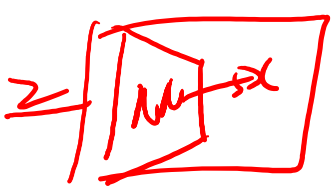
θd

$$\boxed{\min_{\theta_g}}\boxed{\max_{\theta_d}}\left[\mathbb{E}_{x\sim p_{data}}\log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z\sim p(z)}\log(1-\underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}})\right]$$

θg

Saddle point

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)



Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

assume θ_g is fixed

$$\boxed{\max_{\theta_d}} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

$$\boxed{\frac{\partial L_1}{\partial \theta_d}}$$

2. **Gradient descent** on generator

assume θ_d is fixed

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

$$\boxed{\frac{\partial L_2}{\partial \theta_g}}$$

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

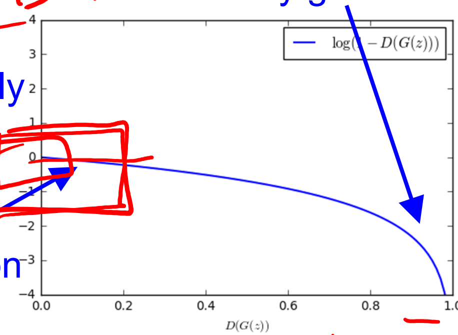
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Gradient signal dominated by region where sample is already good

$\log(1 - D(\cdot))$



$D(\cdot)$

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

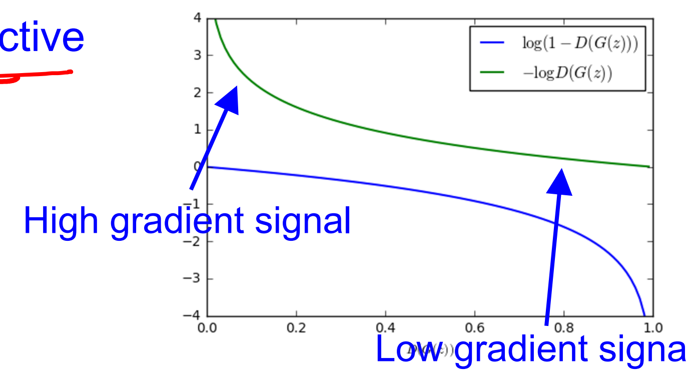
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

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

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.



Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Aside: Jointly training two networks is challenging, can be unstable. Choosing objectives with better loss landscapes helps training, is an active area of research.

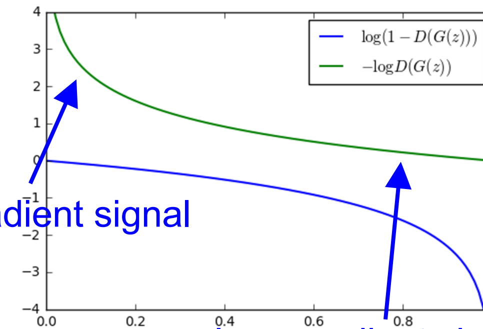
Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

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



High gradient signal

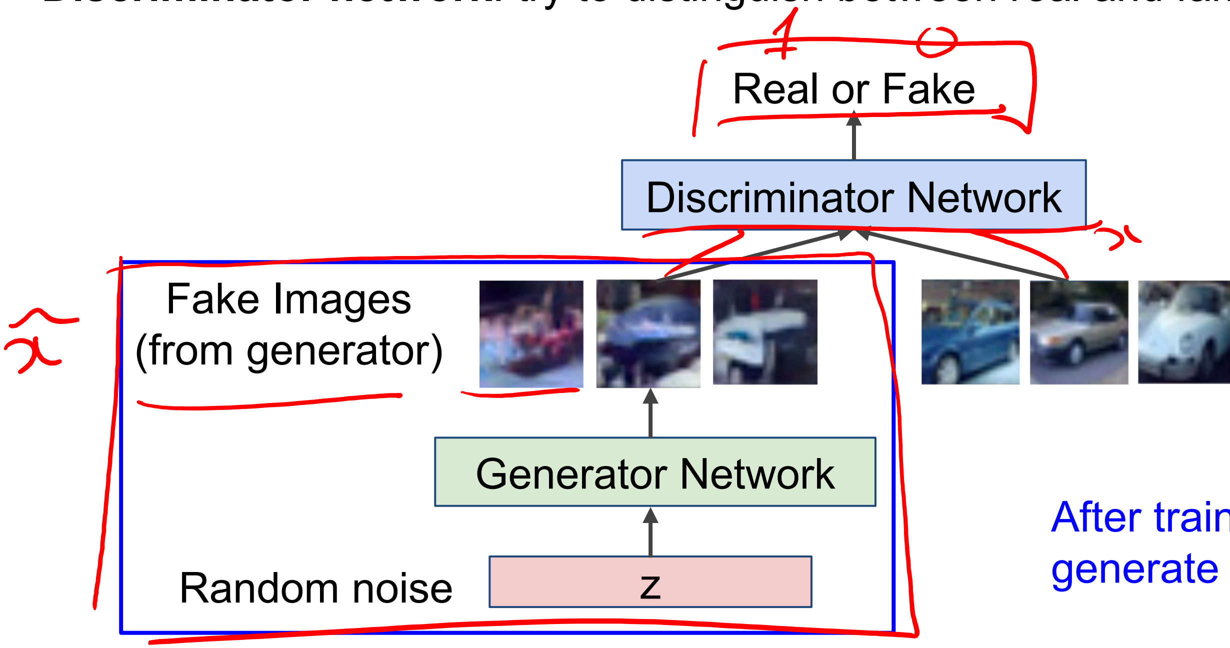
Low gradient signal

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.

Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



Real Images
(from training set)

After training, use generator network to generate new images

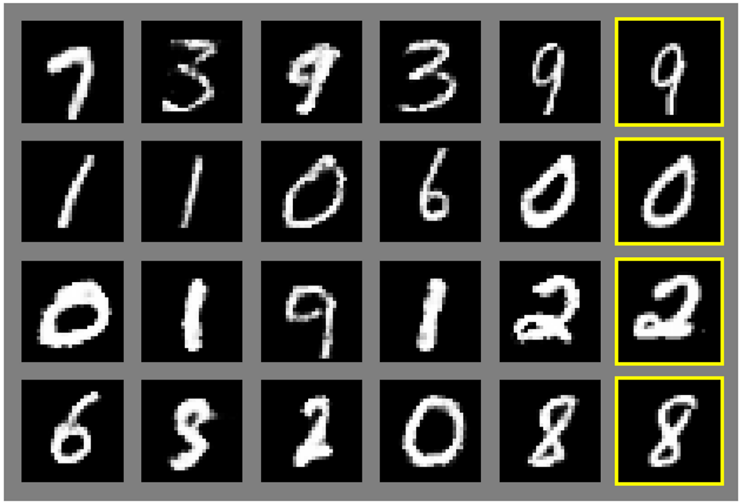
GANs

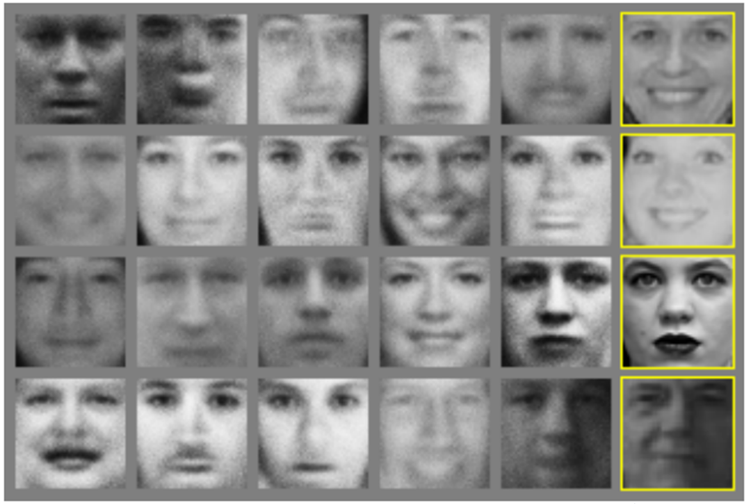
- Demo
  - https://poloclub.github.io/ganlab/

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generative Adversarial Nets

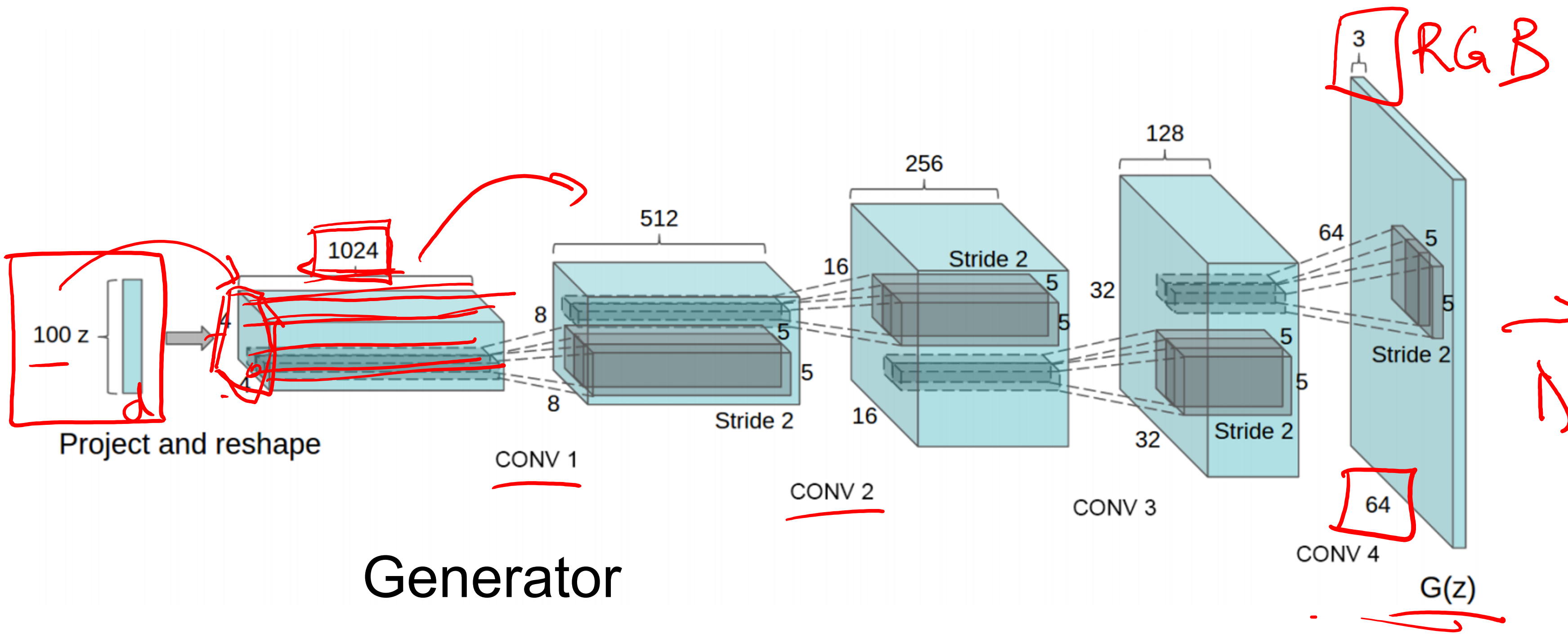
Generated samples





Nearest neighbor from training set

Generative Adversarial Nets: Convolutional Architectures



Generator

Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

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

$\vec{z}_1$ $\vec{z}_2$

Results over the years

The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation, 2018.
https://arxiv.org/abs/1802.07228

BigGAN

Large Scale GAN Training for High Fidelity Natural Image Synthesis
Andrew Brock, Jeff Donahue, Karen Simonyan https://arxiv.org/abs/1809.11096

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).

Large Scale GAN Training for High Fidelity Natural Image Synthesis
Andrew Brock, Jeff Donahue, Karen Simonyan https://arxiv.org/abs/1809.11096

BigGAN

https://gist.github.com/phillipi/d2921f2d4726d7e3cdac7a4780c6050a

Explosion of GANs

"The GAN Zoo"

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

https://github.com/hindupuravinash/the-gan-zoo

Also see https://paperswithcode.com/task/image-generation/latest

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

Plan for Today

- Generative Adversarial Networks (GANs)
- Closing the loop

So what *is* Deep (Machine) Learning?

- A few different ideas:

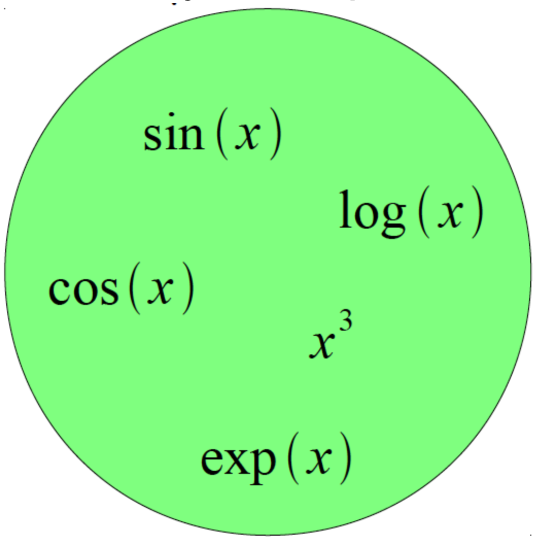
- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extraction

- Distributed Representations
  - No single neuron "encodes" everything
  - Groups of neurons work together

Building A Complicated Function

Given a library of simple functions

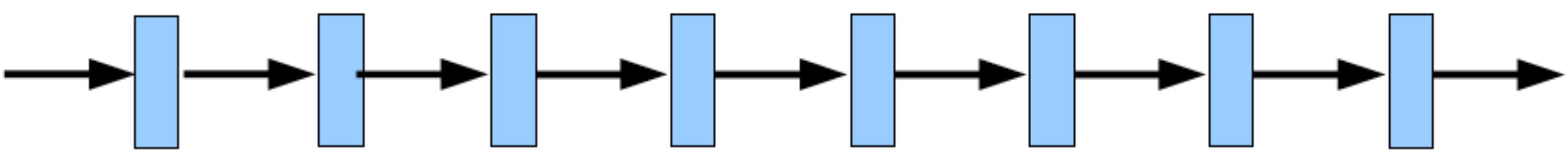


Compose into a complicate function

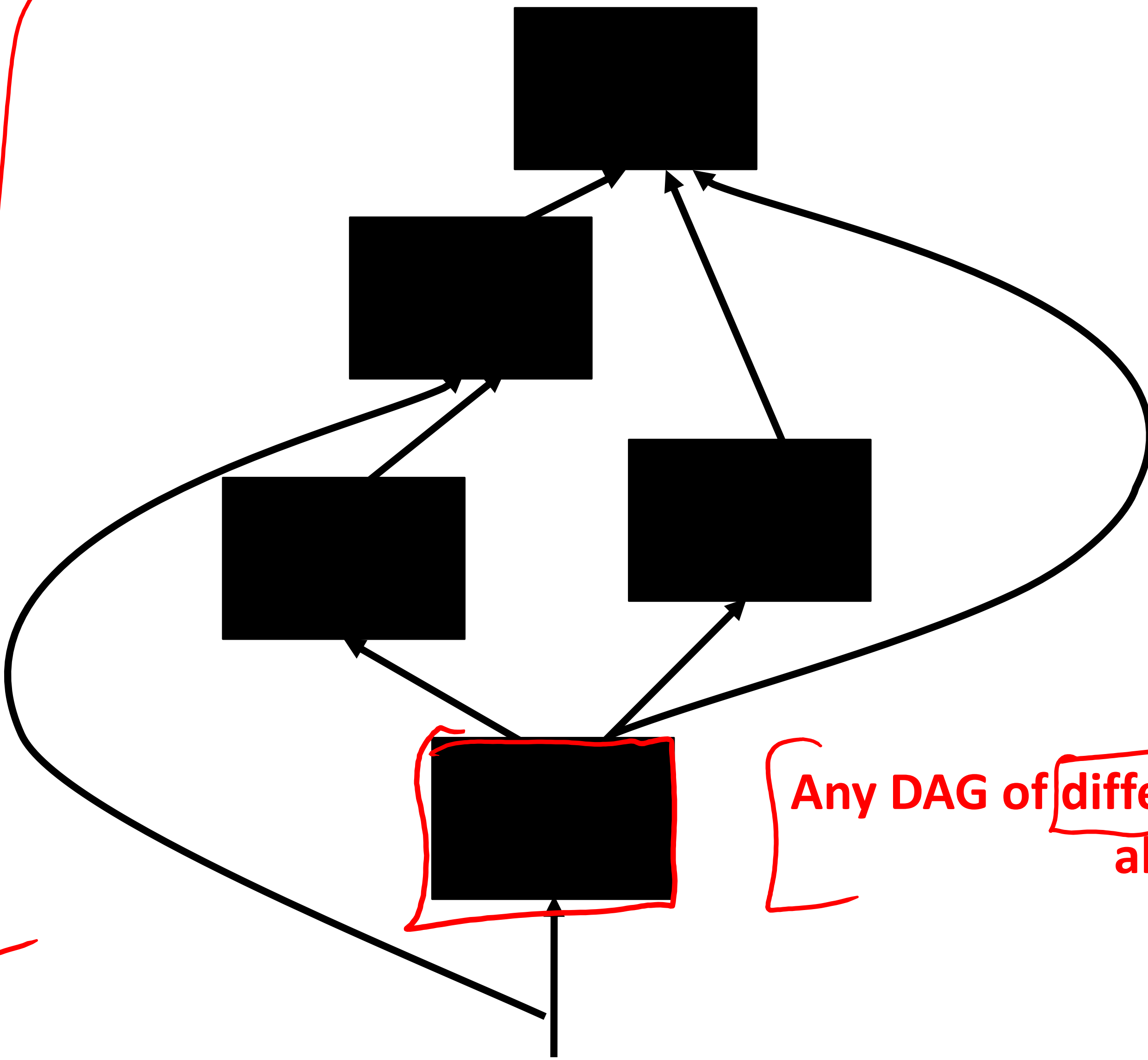
Idea 2: Compositions

- Deep Learning
- Grammar models
- Scattering transforms…

$$f(x) = g_1(g_2(\ldots(g_n(x)\ldots))$$



Differentiable Computation Graph



Any DAG of differentialble modules is allowed!

So what *is* Deep (Machine) Learning?

- A few different ideas:

- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extraction

- Distributed Representations
  - No single neuron “encodes” everything
  - Groups of neurons work together

"Shallow" vs Deep Learning

- "Shallow" models

hand-crafted Feature Extractor → "Simple" Trainable Classifier →

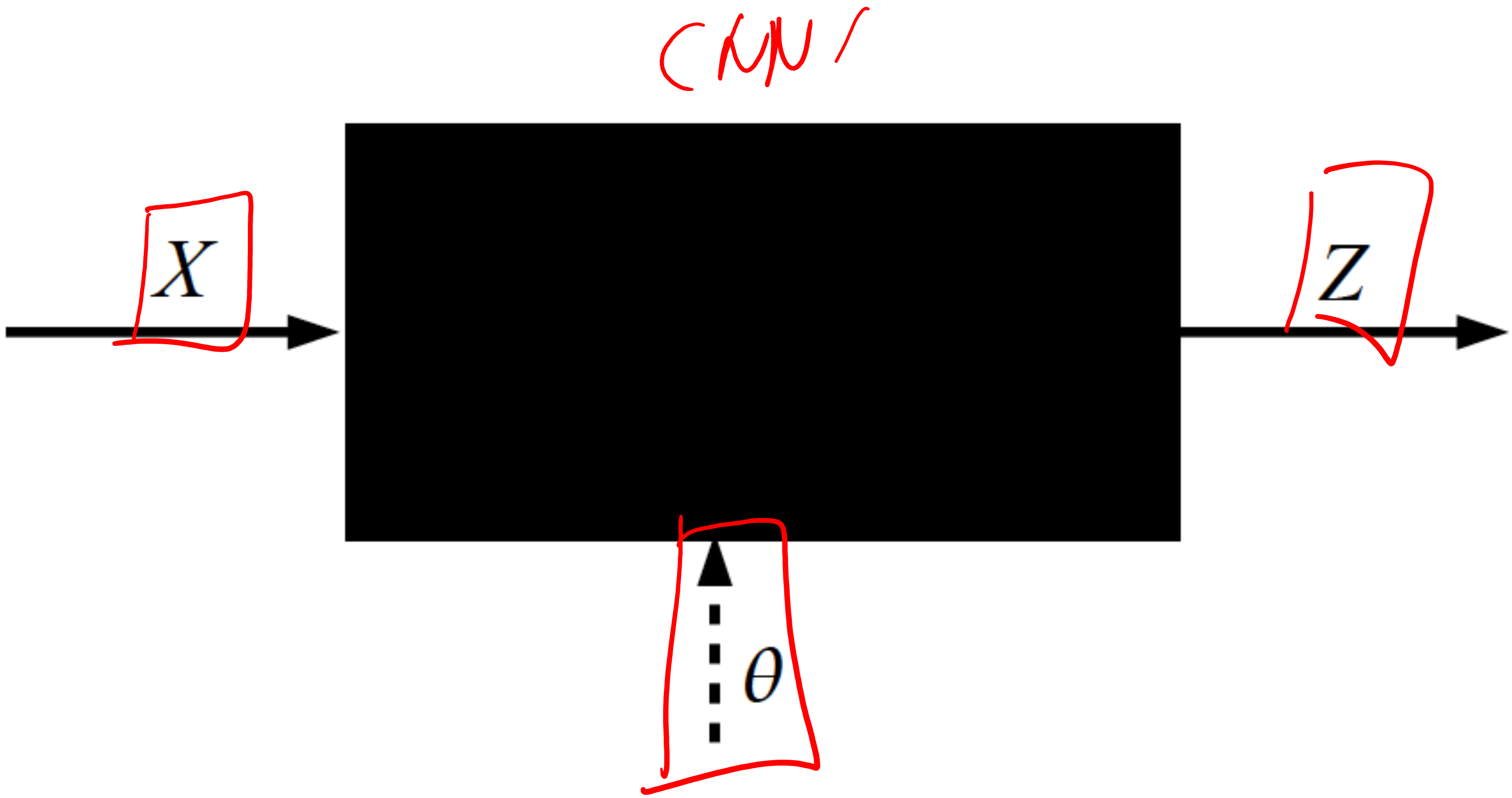
fixed | learned

- Deep models

Trainable Feature-Transform / Classifier → ... → Trainable Feature-Transform / Classifier → Trainable Feature-Transform / Classifier →

Learned Internal Representations

Key Computation: Forward-Prop



Key Computation: Back-Prop

$$\boxed{\frac{\partial L}{\partial X}} \longleftarrow \left\{ \boxed{\frac{\partial Z}{\partial X}}, \boxed{\frac{\partial Z}{\partial \theta}} \right\} \longleftarrow \frac{\partial L}{\partial Z}$$

$$\boxed{\frac{\partial L}{\partial \theta}}$$

So what *is* Deep (Machine) Learning?

- A few different ideas:

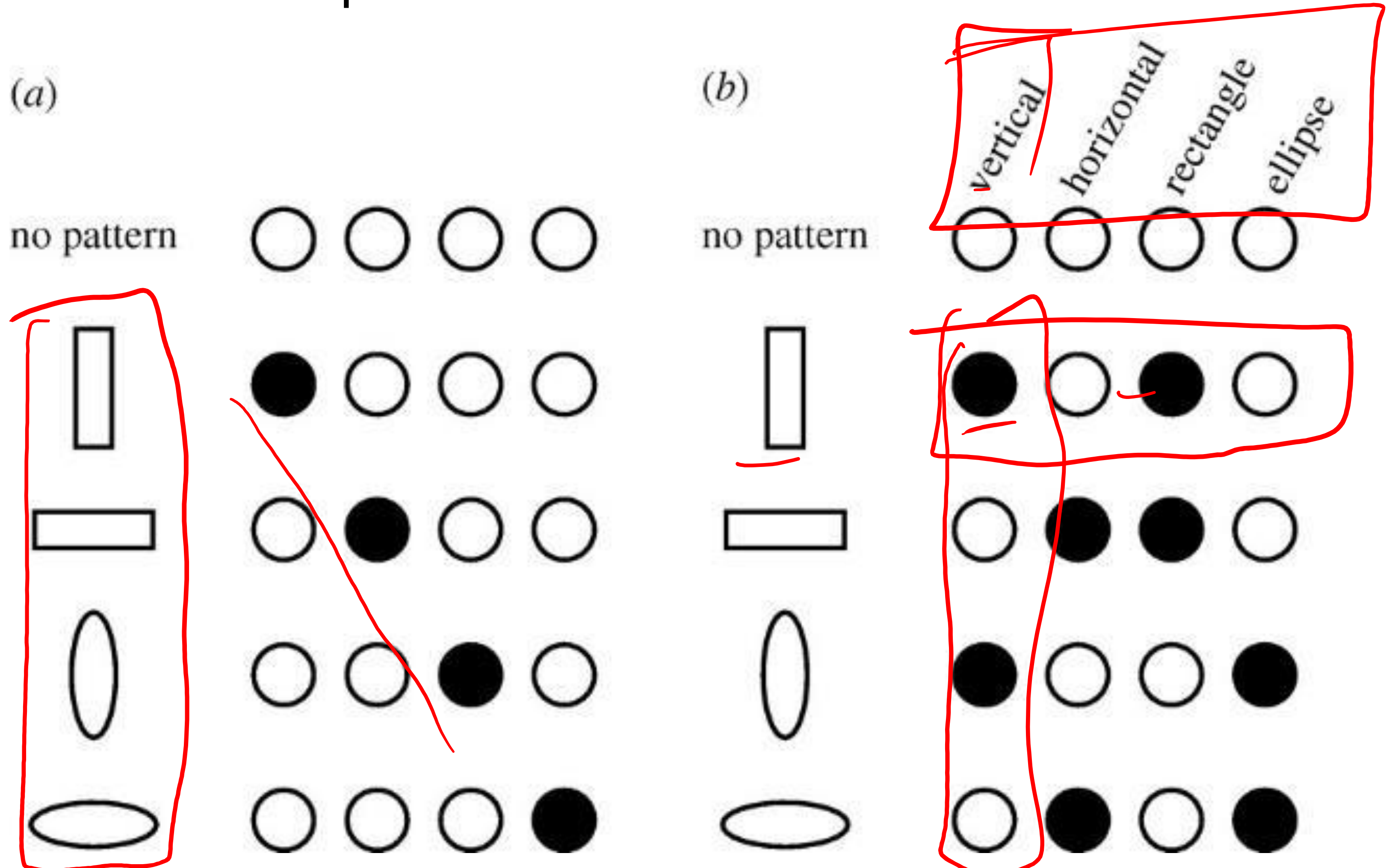
- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extraction

- Distributed Representations
  - No single neuron “encodes” everything
  - Groups of neurons work together

Distributed Representations Toy Example

- Can we interpret each dimension?



Power of distributed representations!

Local ● ● ○ ● = VR + HR + HE = ?

Distributed ● ● ○ ● = $\boxed{V + H + E}$ ≈ ○

$\boxed{Z} \sim p(\cancel{x} z | x)$

$\sim U(,)$

What is this class about?

What is this class about?

- Introduction to Deep Learning

- Goal:
  - After finishing this class, you should be ready to get started on your first DL research project.
    - CNNs
    - RNNs
    - Deep Reinforcement Learning
    - Generative Models (VAEs, GANs)

What did we learn?

- Background & Basics
  - Neural Networks, Backprop, Optimization (SGD)
- Module 1: Convolutional Neural Networks (CNNs)
  - Architectures, Pre-training, Fine-tuning
  - Visualizations, Fooling CNSS, Adversarial examples
  - Different tasks: detection CNNs, segmentation CNNs
- Module 2: Recurrent Neural Networks (RNNs)
  - Difficulty of learning; "Vanilla" RNNs, LSTMs, GRU
  - RNNs for Sequence-to-Sequence (machine translation & image captioning, VQA, Visual Dialog)
- Module 3: Deep Reinforcement Learning
  - Overview, policy gradients
  - Optimizing Neural Sequence Models for goal-driven rewards
- Module 4: Deep Unsupervised Learning
  - Variational Inference
  - Variational Auto Encoders (VAEs)
  - GANs, Adversarial Learning

Arxiv Fire Hose

$\sim p(x|\boxed{z})$

Deep Learning papers

PhD Student

Cornell University Library

arXiv.org

Feedback

http://b.gatech.edu/cios

Thanks!

(We hope your future learnings are deep)