CIS 522: Lecture 1

History of Deep Learning

Penn Engineering

This course has a nonstandard format

We believe in learning in groups

- Hence we insist that you participate in group (pod)

We believe in learning by doing

- Hence most is in short embedded videos of 5 min
- Followed by doing it

We believe in you being creative

- Hence a heavy focus on the projects

Deep Learning and Society

Machine learning matters:

- Decides which movies you see and books you read
- Which scientific articles you read
- Who are your friends
- If you get credit
- If you get out of prison
- If the police searches you
- **We need to understand what data does!**

Key questions in this course

1. How do we decide which problems to tackle with deep learning?
2. Given a problem setting, how do we determine what model to use?
3. What’s the best way to implement said model?
4. How can we best visualize, explain, and justify our findings?
5. How can neuroscience inspire deep learning?

What we're covering

1. Fundamentals of Deep Learning (Weeks 1-5)
2. Computer Vision (Weeks 6-8)
3. NLP (Weeks 9-10)
4. Reinforcement Learning (Week 11-12)
5. Future DL (Weeks 13)
6. Project work (Remainder)

PYTORCH

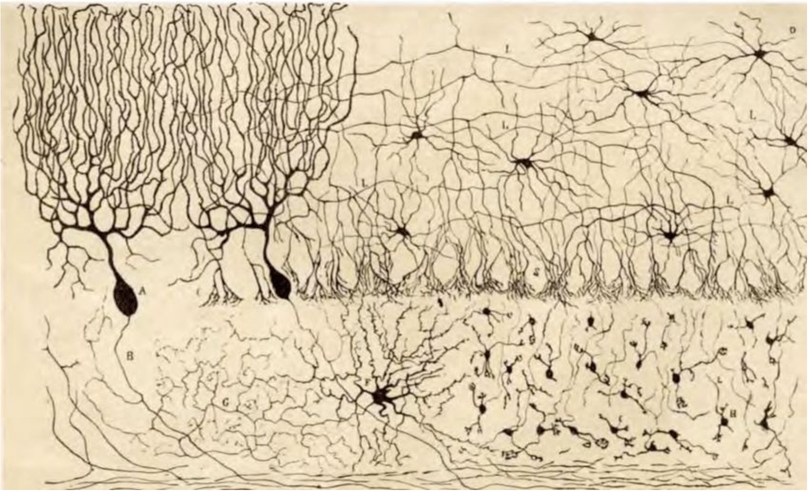
Key questions covered by other courses

1. **CIS 580, 581:** What are the foundations relating vision and computation?
2. **CIS 680:** What is the SOTA* architecture for _ problem domain in computer vision?
3. **CIS 530:** What are the foundations relating natural language and computation?
4. **ESE 546:** How and why does the mathematics behind different architecture work?
5. **CIS 700-001:** What is the SOTA* architecture for _ problem domain in NLP?
6. **STAT 991:** What is the cutting-edge of formal deep learning research?

* SOTA = State of the Art

History of Neural Networks

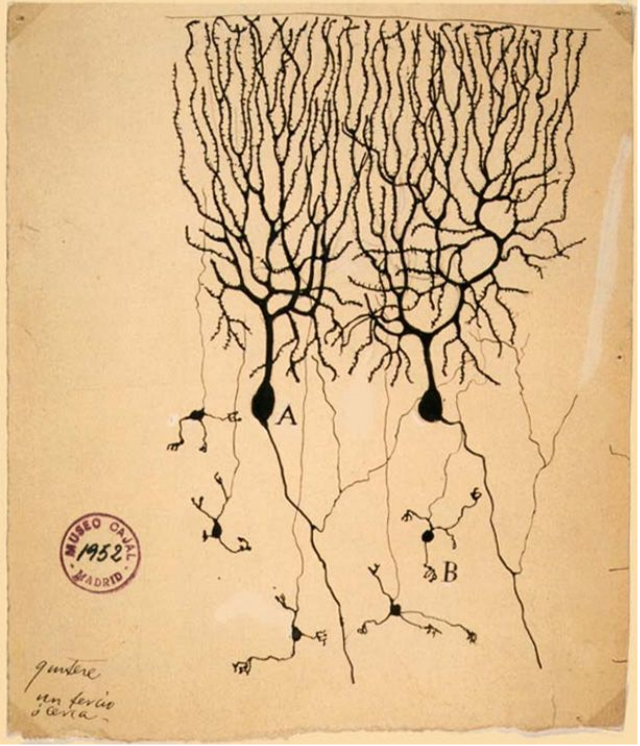
The Neuron Doctrine - Santiago Ramon y Cajal (1888)



Picture taken from *Revista Trimestral de Histología Normal y Patológica by Santiago Ramon y Cajal* from which the Neuron Doctrine originated.

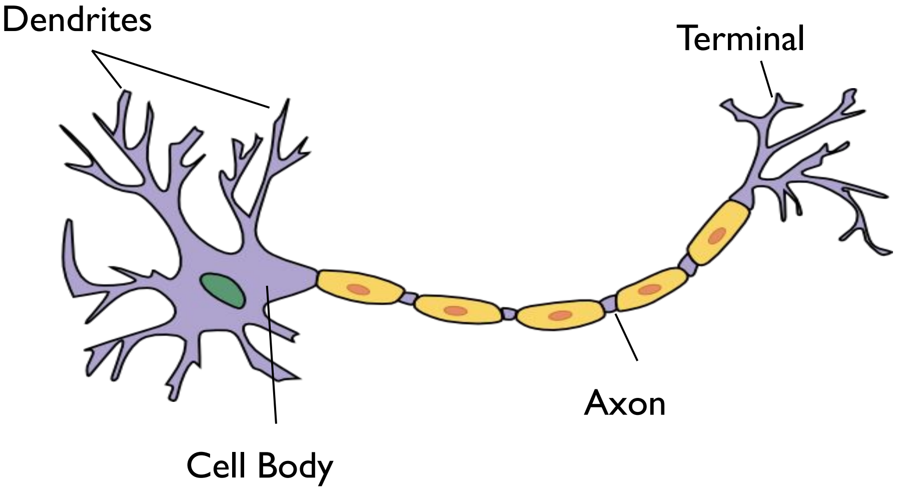
Idea: The nervous system is **not** just one continuous thread-like cell, but rather composed of **multiple individual cells**, later called *neurons* by anatomist H. Waldeyer-Hartz.

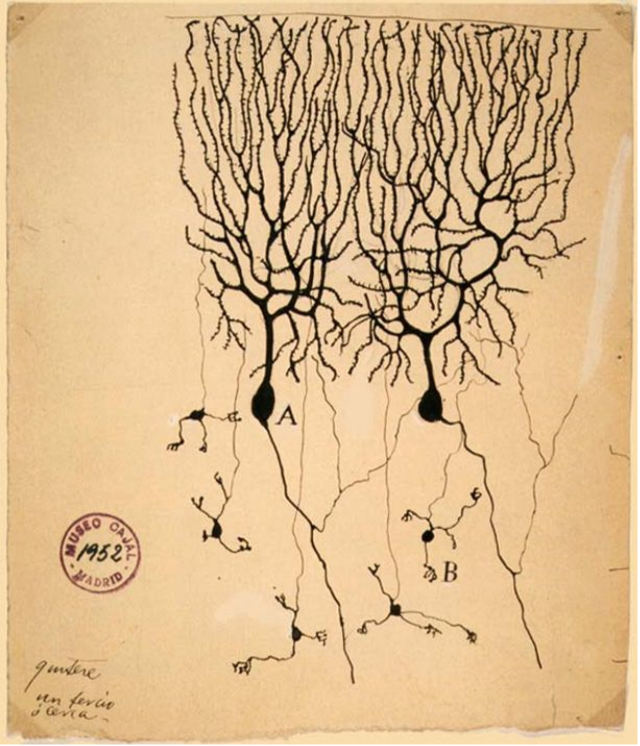
Law of Dynamic Polarization (also Cajal)



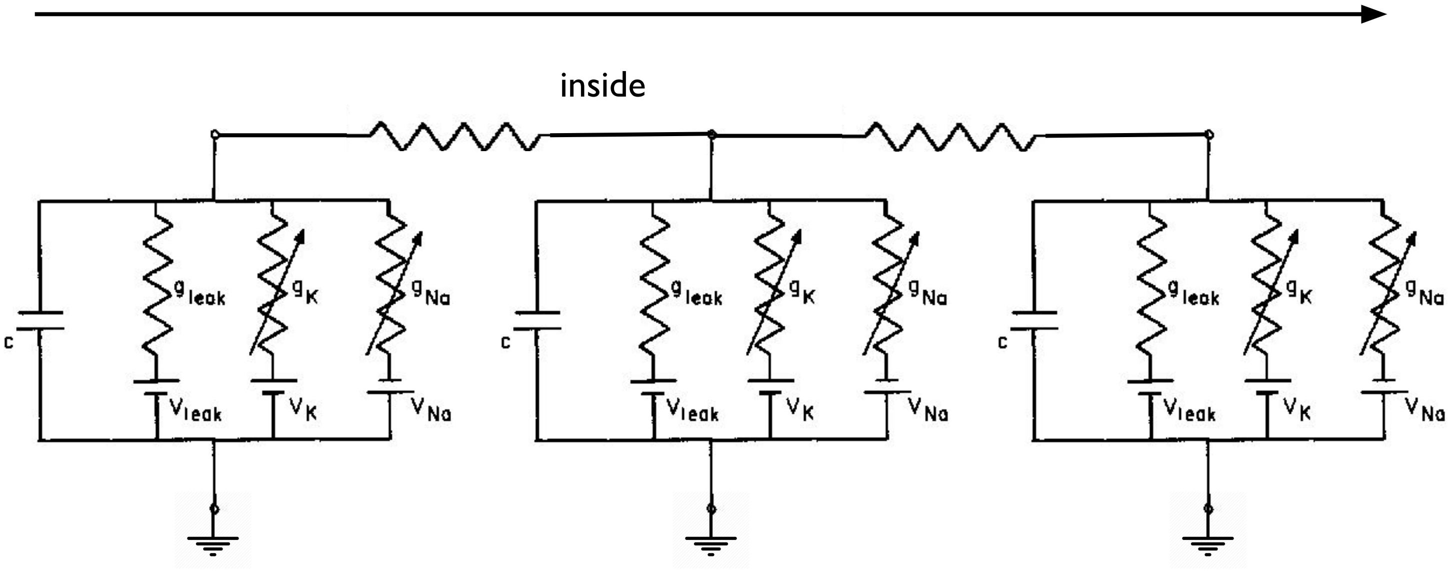
Law of Dynamic Polarization (also Cajal)

<u>Law:</u> Information travels in **one direction,** from the dendrites to the cell body through the axon and to the terminal.





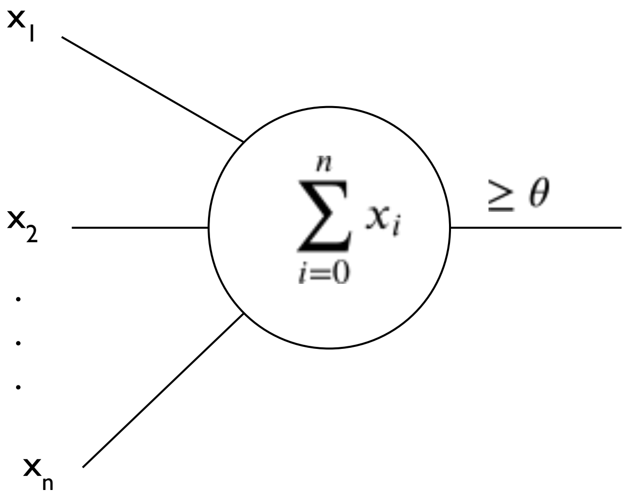
An axon



inside

outside

McCullochs - Pitts
Linear Threshold Unit (LTU), 1943



Inputs:

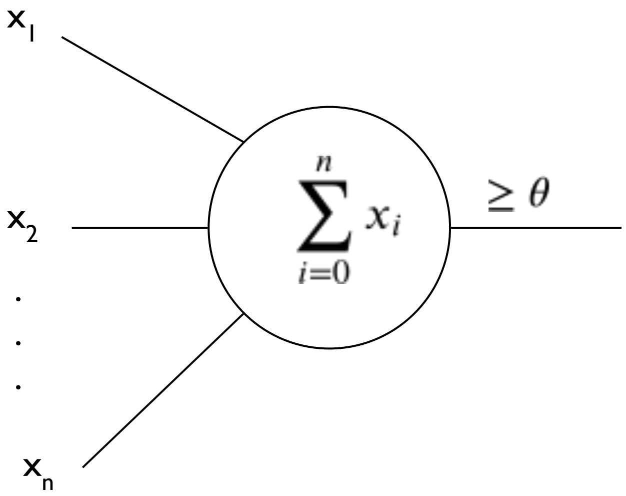
- $x \in \{0, 1\}$
- $\theta \in \mathbb{Z}^*$

Outputs:

- $y \in \{0, 1\}$

Function: $\sum_{i=0}^{n} x_i \geq \theta$

McCullochs - Pitts
Linear Threshold Unit (LTU), 1943



Inputs:

- $x \in \{0, 1\}$
- $\theta \in \mathbb{Z}^*$

Outputs:

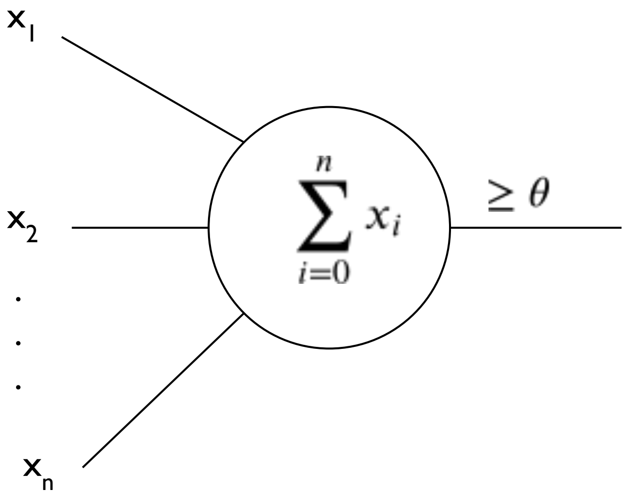
- $y \in \{0, 1\}$

Function: $\sum_{i=0}^{n} x_i \geq \theta$

A sum is a **linear** transformation and this sum is then **thresholded** by theta.

*Hence, **Linear Threshold Unit!***

McCullochs - Pitts
Linear Threshold Unit (LTU), 1943



Inputs:

- $x \in \{0, 1\}$
- $\theta \in \mathbb{Z}^*$

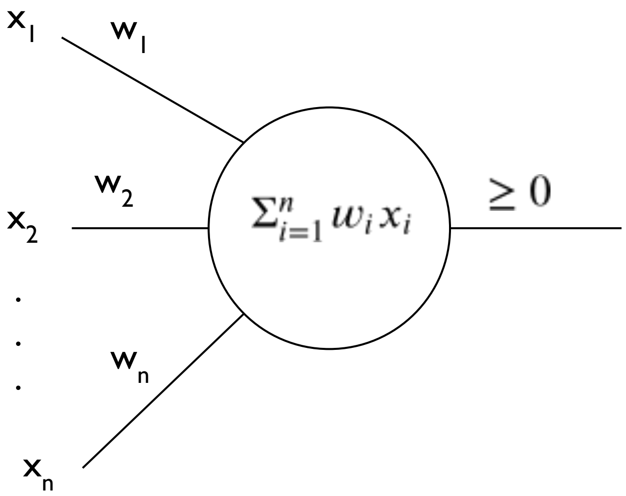
Outputs:

- $y \in \{0, 1\}$

Function: $\sum_{i=0}^{n} x_i \geq \theta$

Note: We do not learn the threshold parameter, it must be specified beforehand.

Rosenblatt’s perceptron (1958)



Inputs:

- $x \in \mathbb{R}^n$
- $w \in \mathbb{R}^n$

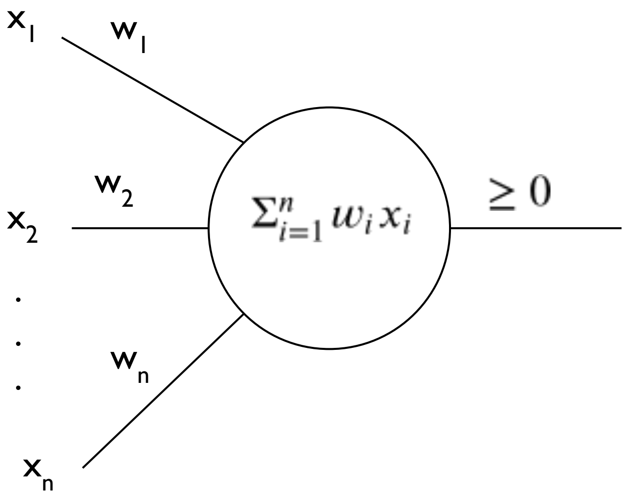
Outputs:

- $y \in \{0, 1\}$

Function:

$$y = x \cdot w \geq 0$$

Rosenblatt’s perceptron (1958)



Inputs:

- $x \in \mathbb{R}^n$
- $w \in \mathbb{R}^n$

Outputs:

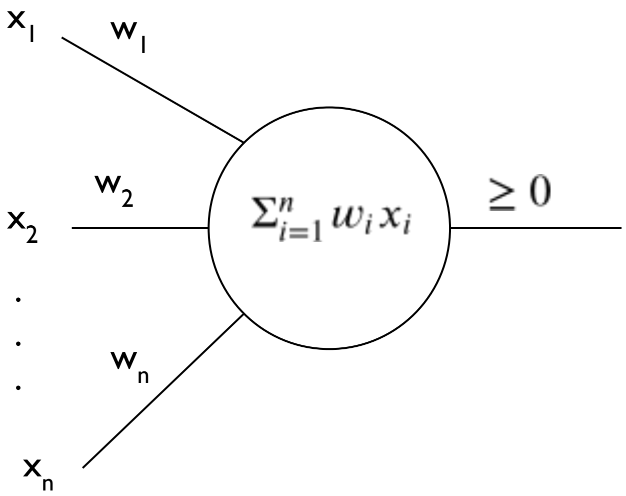
- $y \in \{0, 1\}$

Function:

$y = x \cdot w \geq 0$

What happened to θ*?*

Rosenblatt's perceptron (1958)



Inputs:

- $x \in \mathbb{R}^n$
- $w \in \mathbb{R}^n$

Outputs:

- $y \in \{0, 1\}$

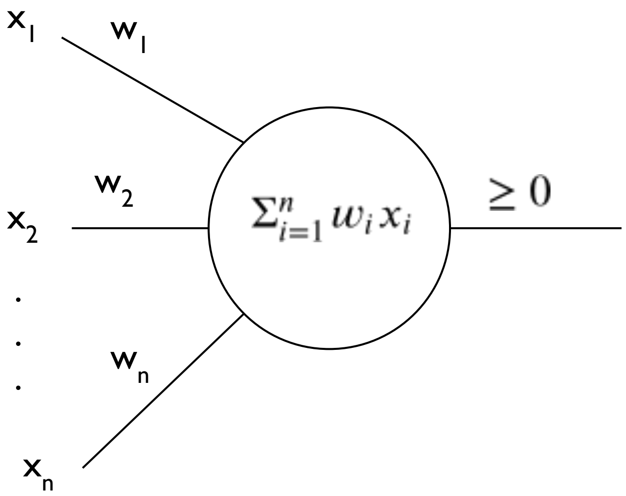
Function:

$y = x \cdot w \geq 0$

What happened to θ?

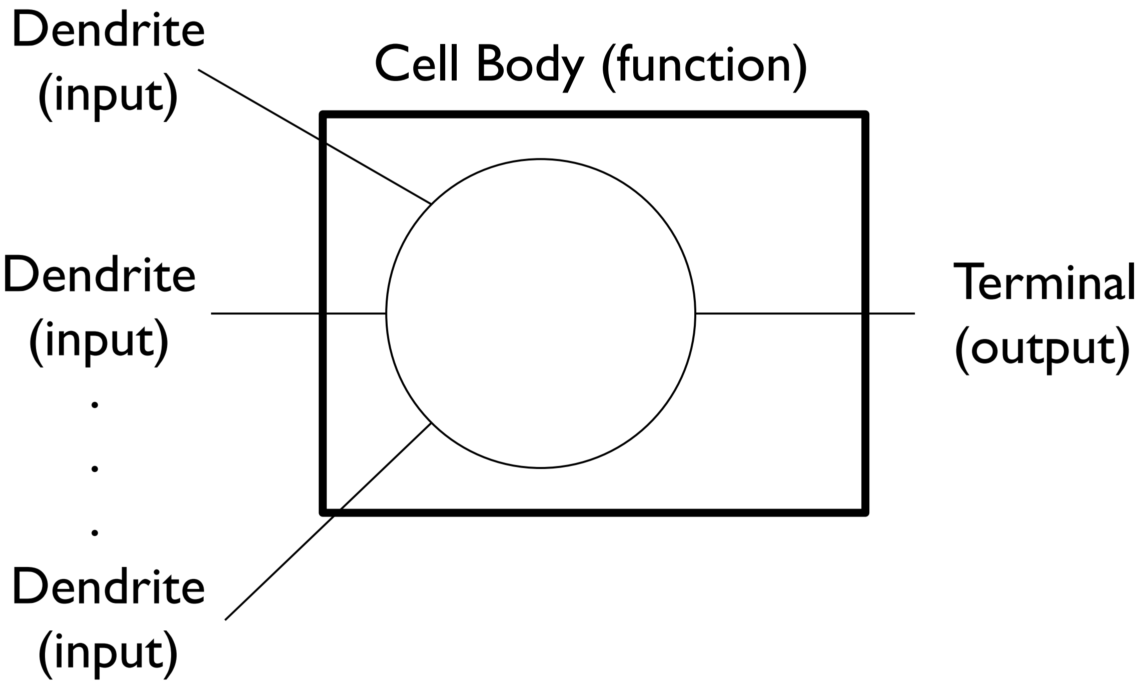
One of x's features is 1 and the corresponding weight is - θ.

Rosenblatt's perceptron (1958)



Seem familiar?

Rosenblatt’s perceptron (1958)



Rosenblatt’s Perceptron - Algorithm

1. Initialize w_0 to a random vector
2. Given training example *(x, y)*
3. Input training example x in to perceptron, denote the output as $\hat{y}$.
4. At iteration *t* denote $w_{t+1} = w_t + \alpha(y - \hat{y})x$ (α is the learning rate)
5. Repeat sets 2-4 for desired number of iterations.

Provably converges to optimal solution

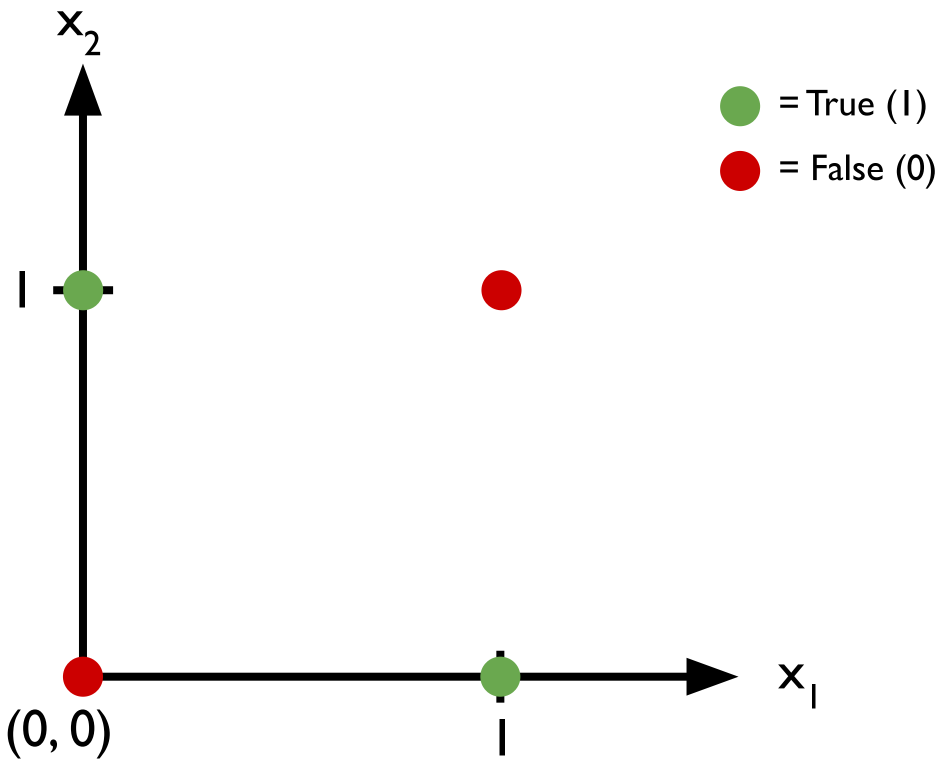
XOR Problem

Problem: Consider the truth table below for the XOR function. Pretty simple right?

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

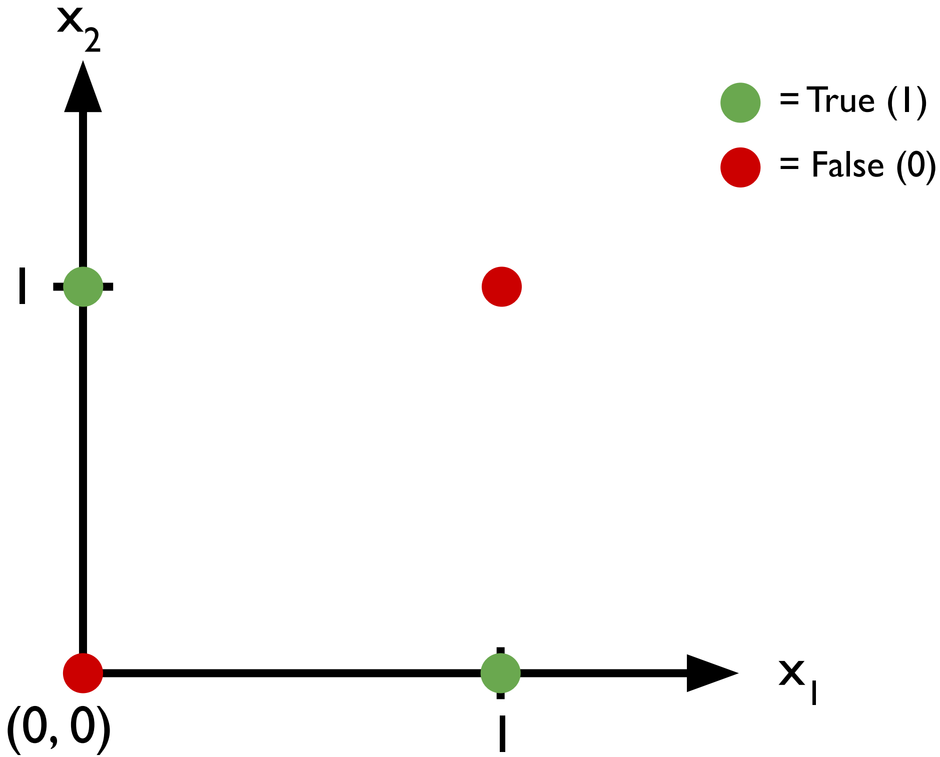
XOR Problem

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0



XOR Problem

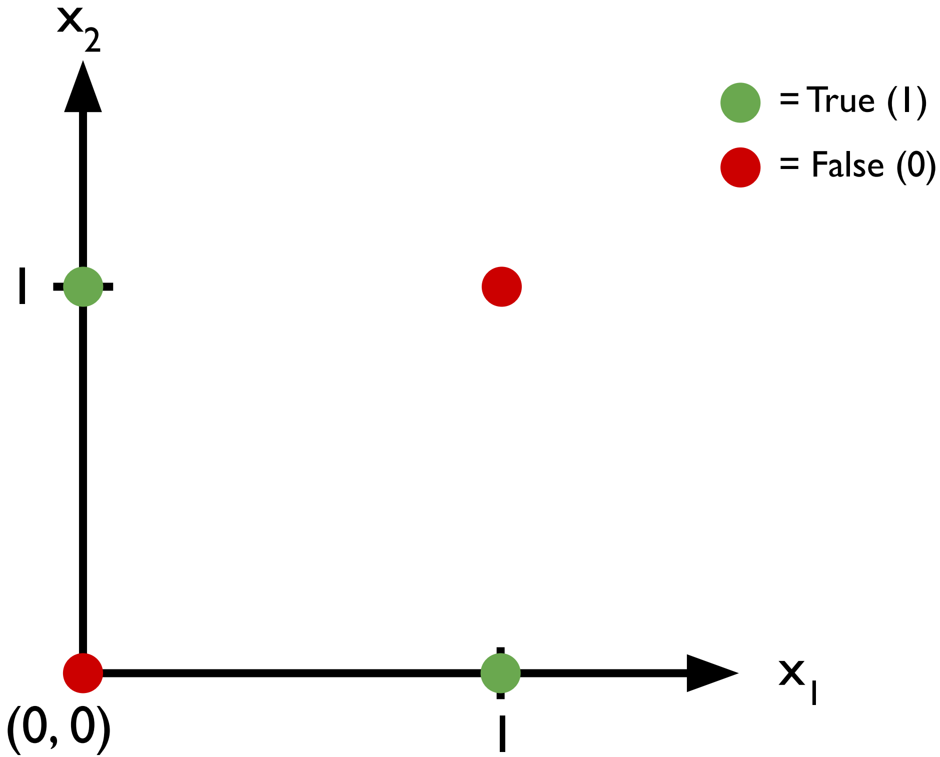
Can you draw a line that separates the positive from negative examples?



XOR Problem

Can you draw a line that separates the positive from negative examples?

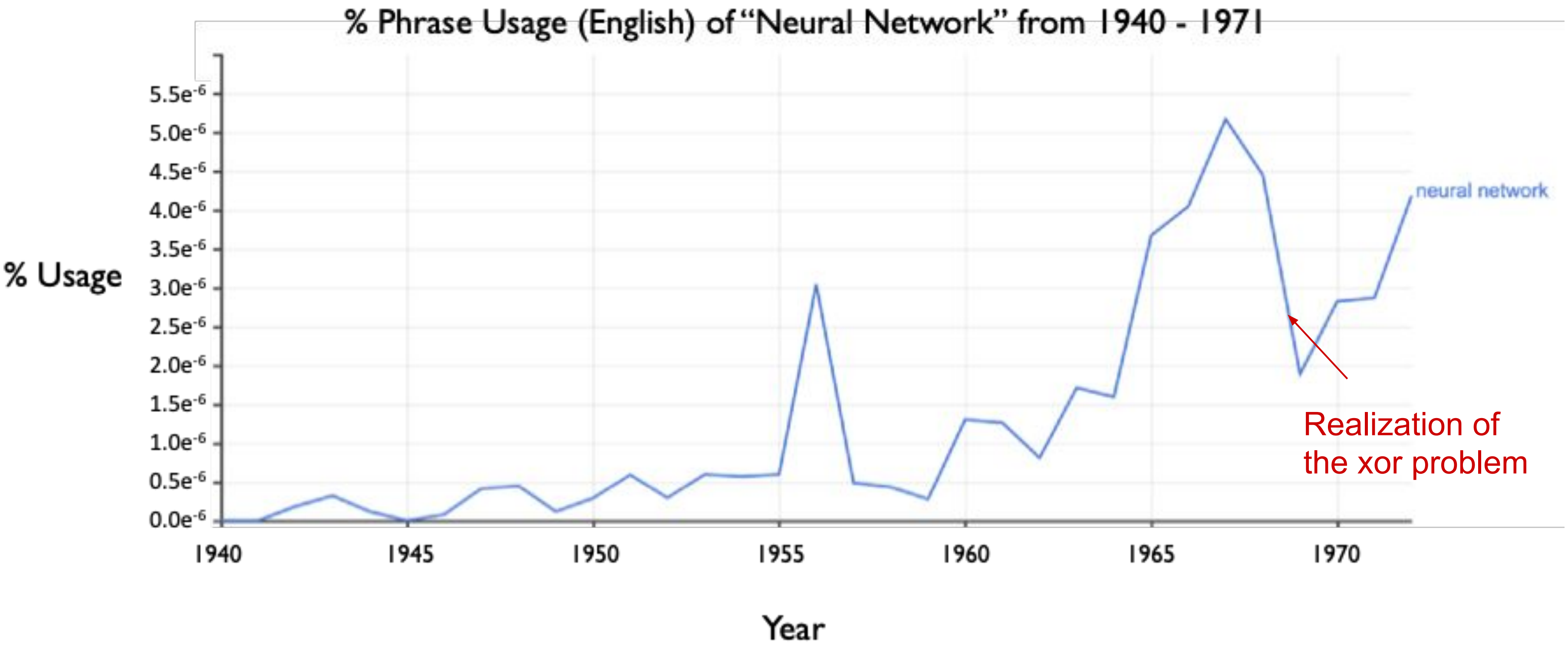
Impossible with the XOR function.



As of 1970, there was one huge problem with neural networks

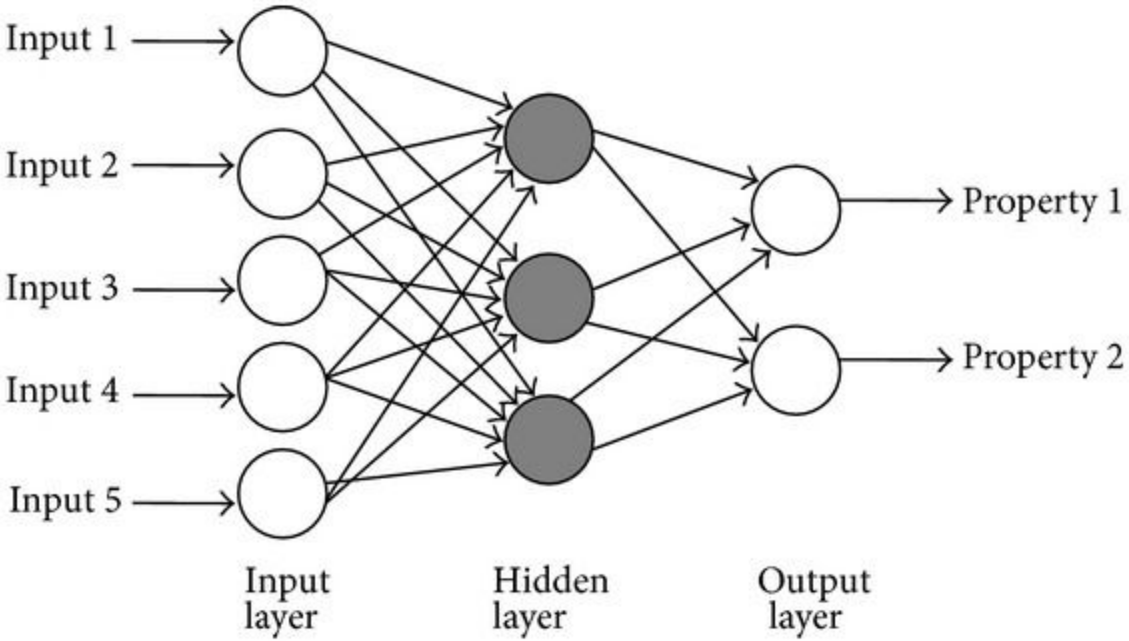
1. Neural networks could not solve any problem that wasn't linearly separable

Winter #1



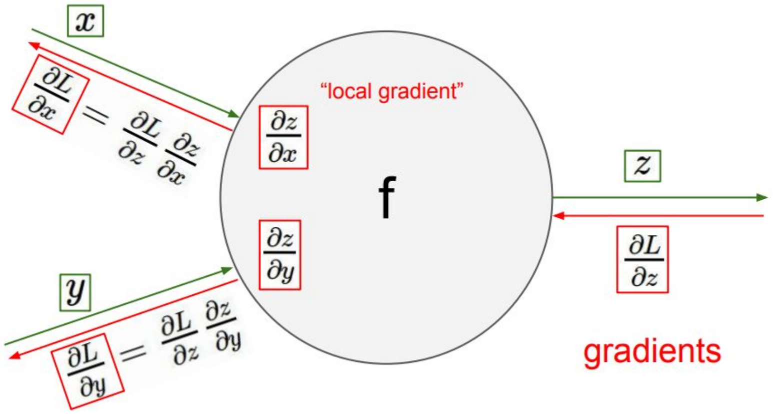
Realization of the xor problem

Multi-layer perceptrons



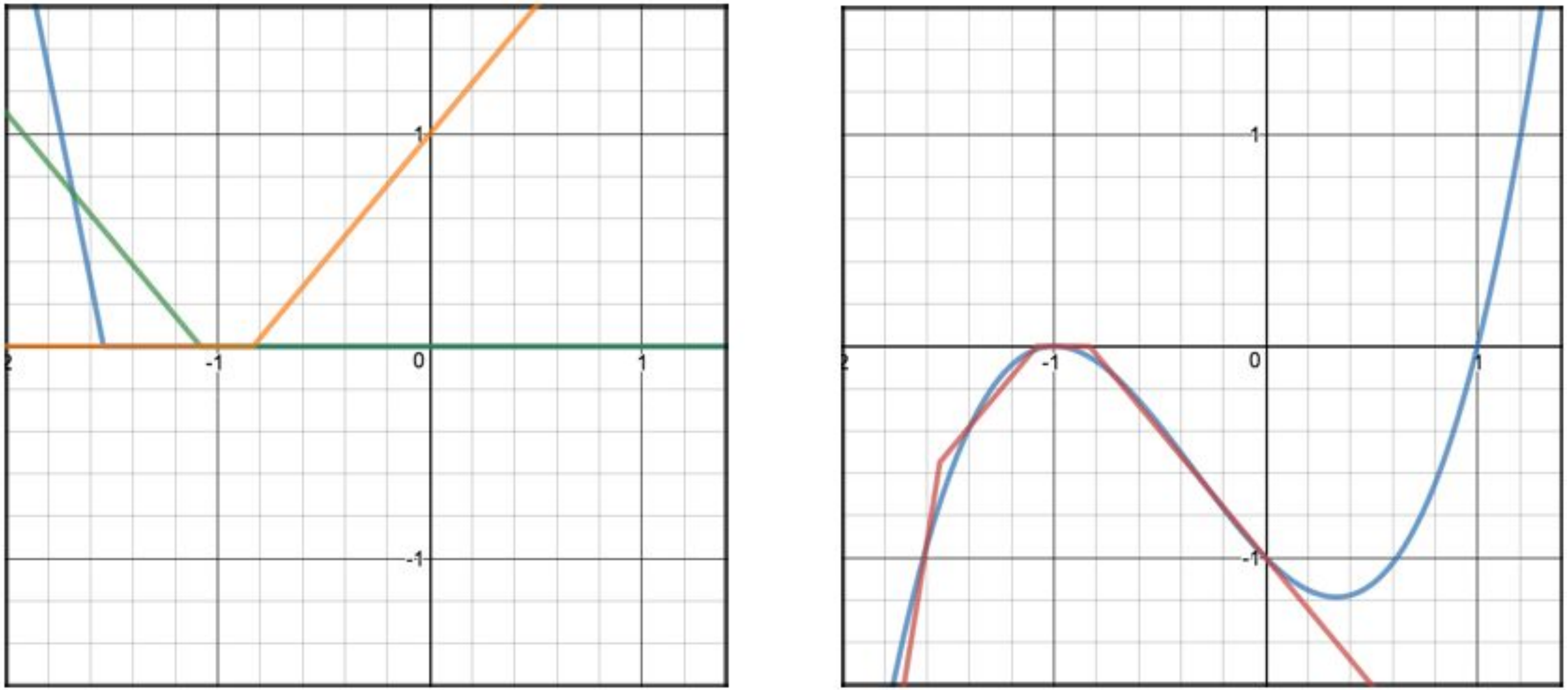
Solution: Differentiation/ Backpropagation

- Using chain rule to calculate partial derivative of cost with respect to every parameter.
- Every float operation performed by a computer at some level involves:
  - Elementary binary operators (+, - , x, /)
  - Elementary functions (sin x, cos x, e^x, etc.)

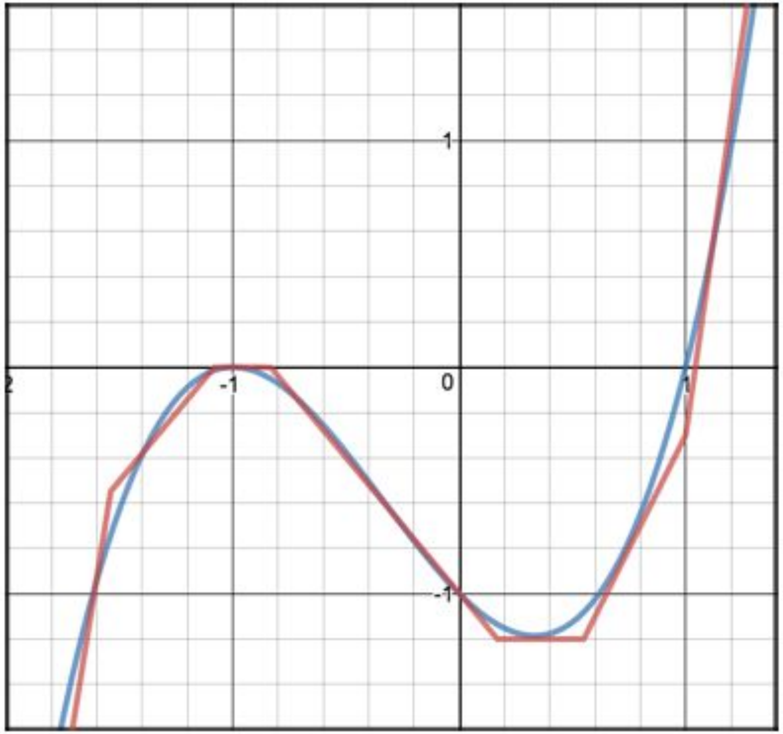


x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

Universal Function Approximator



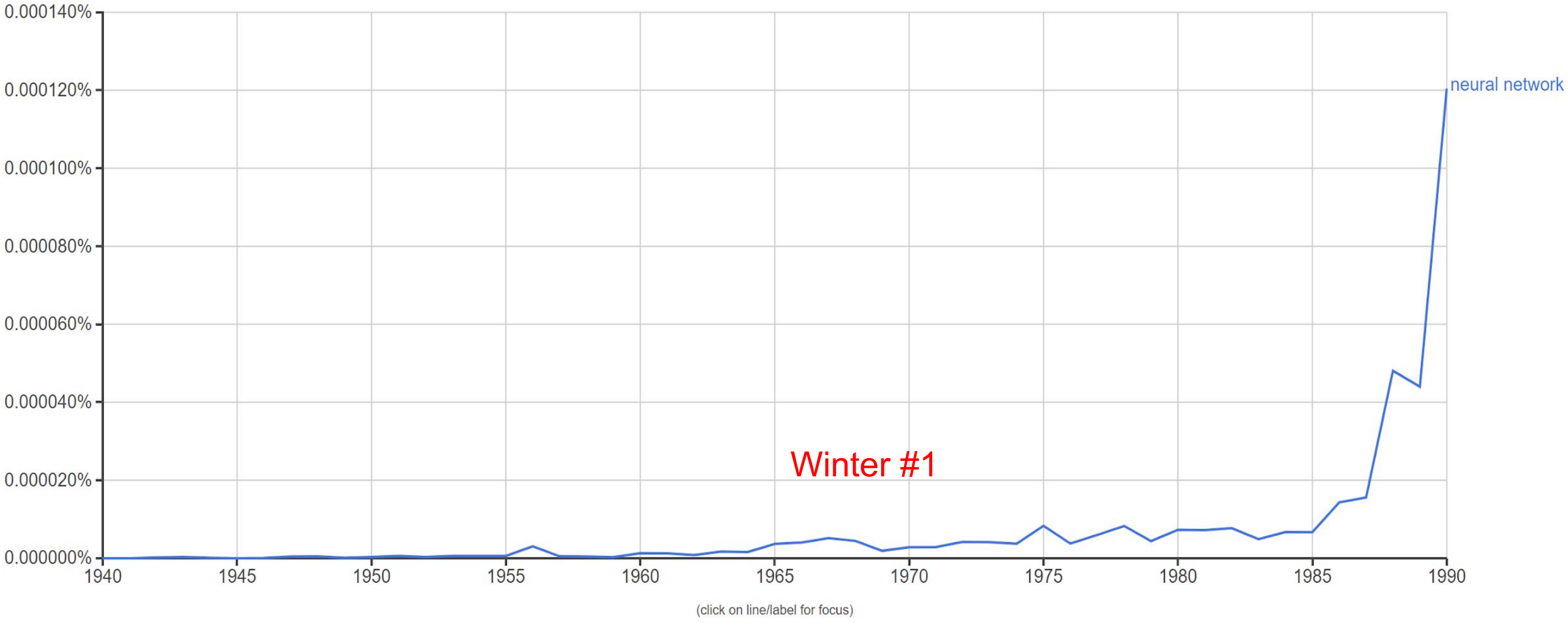
Universal Function Approximator



$$n_1(x) = Relu(-5x - 7.7)$$
$$n_2(x) = Relu(-1.2x - 1.3)$$
$$n_3(x) = Relu(1.2x + 1)$$
$$n_4(x) = Relu(1.2x - .2)$$
$$n_5(x) = Relu(2x - 1.1)$$
$$n_6(x) = Relu(5x - 5)$$

$$\begin{aligned} Z(x) = &-n_1(x) - n_2(x) - n_3(x) \\ &+ n_4(x) + n_5(x) + n_6(x) \end{aligned}$$

Resurgence of Interest in Neural Nets



As of 1986, there were 2 huge problems with neural nets.

1. ~~Neural networks could not solve any problem that wasn't linearly separable~~
    a. Solved by backpropagation and depth.
2. Backpropagation takes forever to converge!
    a. Not enough compute power to run the model
    b. Not enough labeled data to train the neural net

As of 1986, there were 2 huge problems with neural nets.

~~1. They couldn't solve any problem that wasn't linearly separable.~~

   a. Solved by backpropagation and depth.

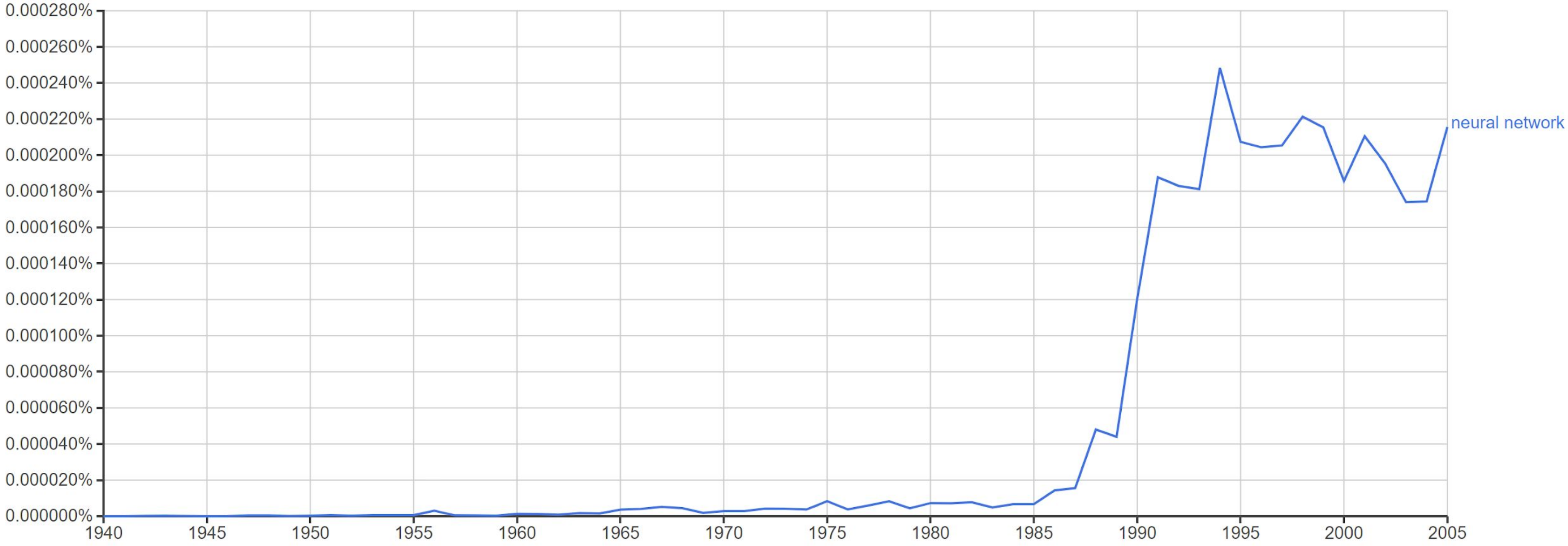
2. Backpropagation takes forever to converge!

   a. Not enough compute power to run the model

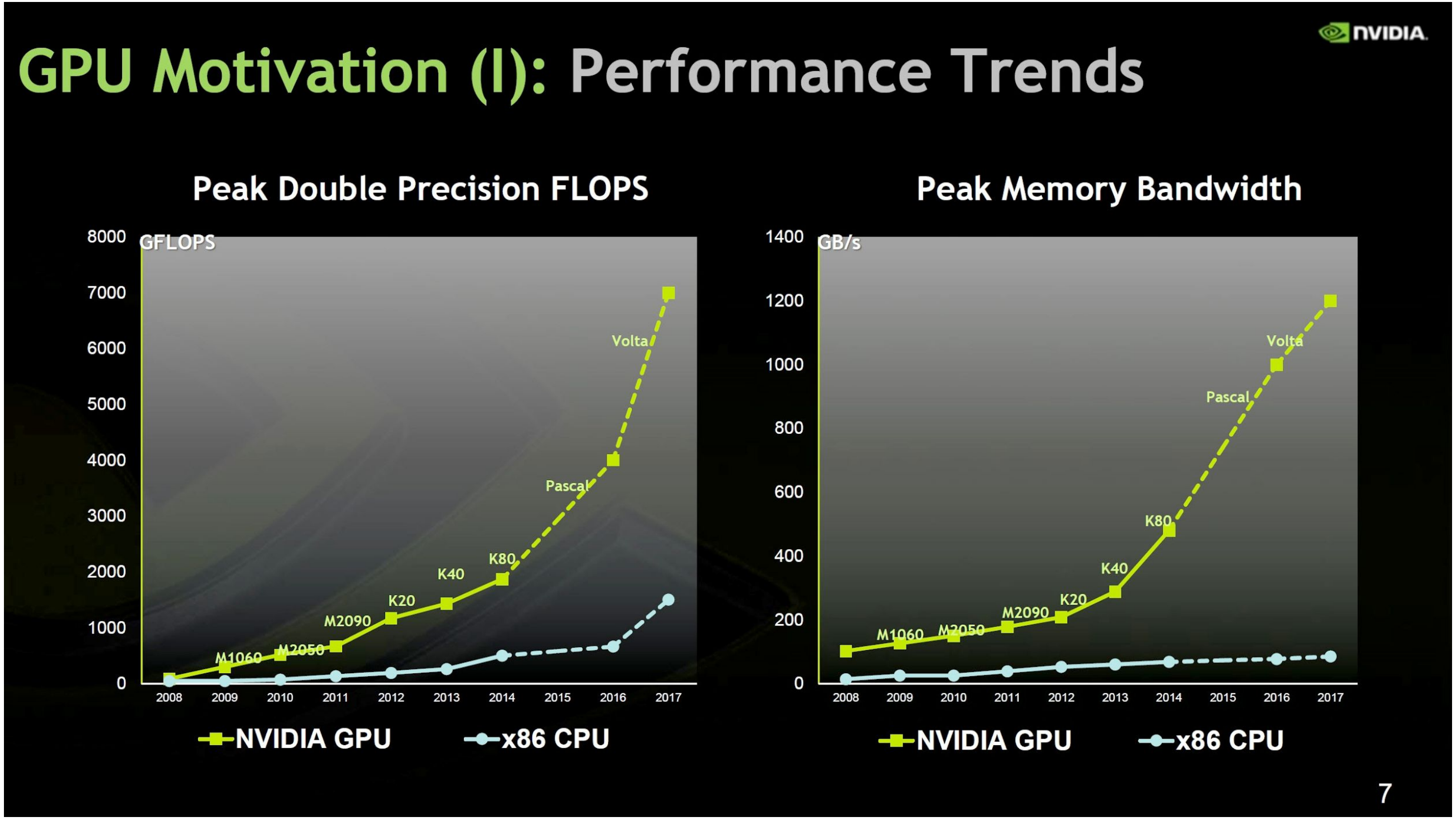
   b. Not enough labeled data to train the neural net

Outclassed by SVM

Winter #2


0.000280%
0.000260%
0.000240%
0.000220%
0.000200%
0.000180%
0.000160%
0.000140%
0.000120%
0.000100%
0.000080%
0.000060%
0.000040%
0.000020%
0.000000%
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
neural network

Part of the solution: the GPU



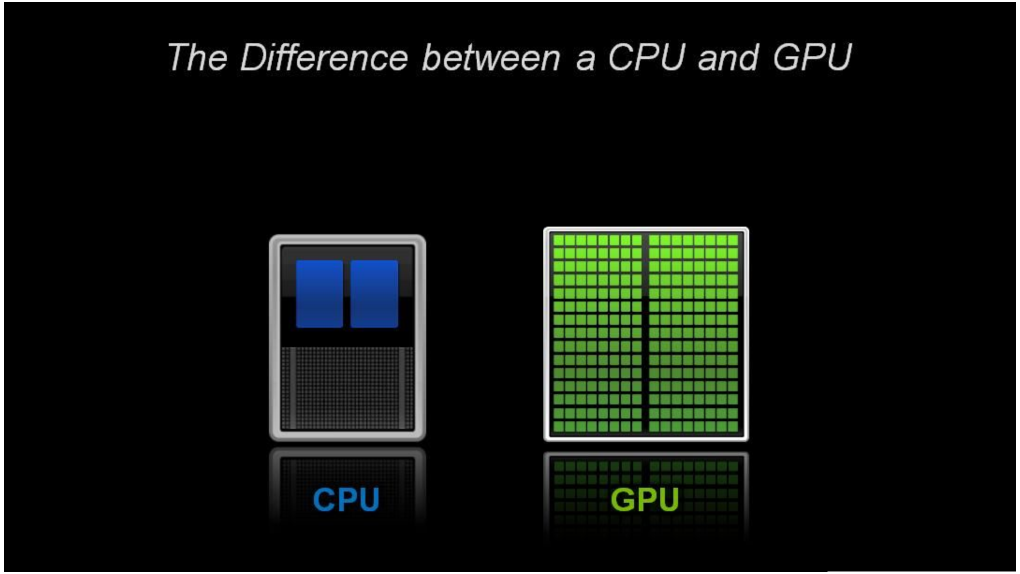
GPU Motivation (I): Performance Trends

Peak Double Precision FLOPS

Peak Memory Bandwidth

Why are GPUs so good at matrix multiplication?

1. Much higher bandwidth than CPUs.
2. Better parallelization.
3. More register memory.



NVIDIA
CUDA

As of 2007, there was one huge problem with neural nets.

1. ~~They couldn't solve any problem that wasn't linearly separable.~~
   a. Solved by backpropagation and depth.
2. Backpropagation takes forever to converge!
   a. ~~Not enough compute power to run the model~~
      i. Solved by GPU
   b. Not enough labeled data to train the neural net

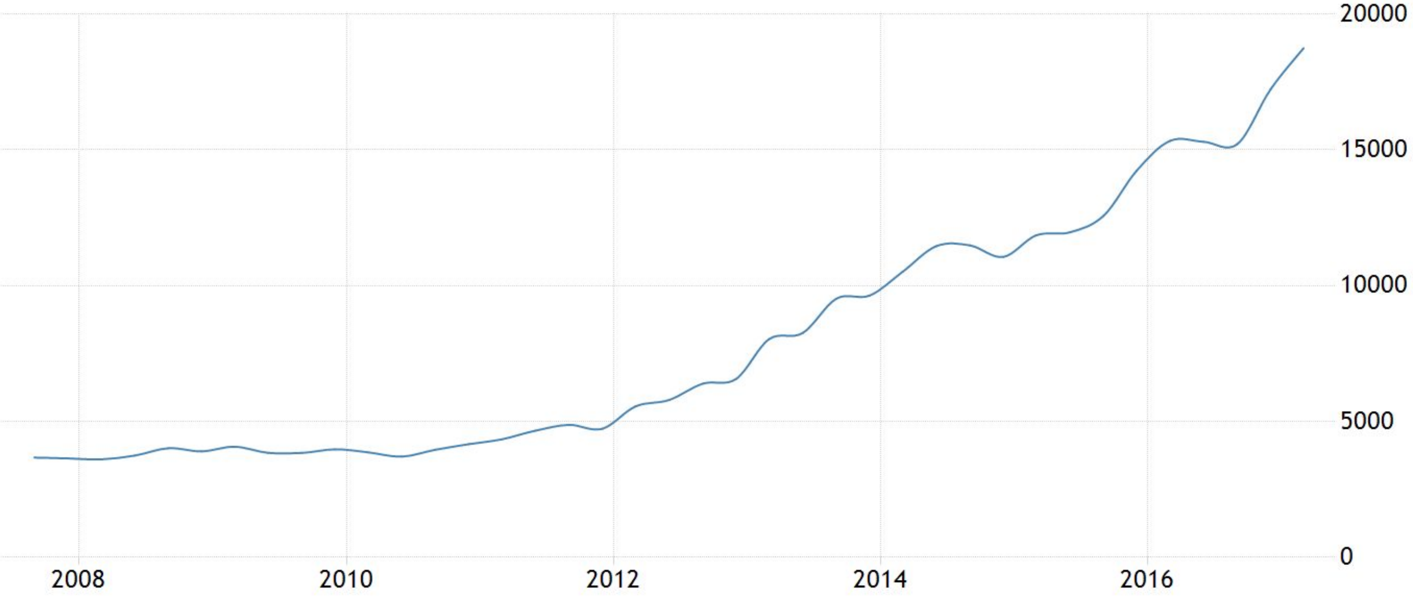
Lots of Data!

- 2005-2012: Pascal Visual Object Classes
  - 20 classes, 27.5k annotations (in most recent)
- 2010: ImageNet
  - 27 categories, 21.3k subcategories, ~1M images with annotations!
- 2014-2017 COCO
  - 80 classes, ~250k images with bounding boxes and segmentations
- 2017-2019 OpenImages
  - 9M images, 16M bounding boxes, 600 classes, 2.8M segmentation masks

And lots of data augmentation (e.g. rotation, crop, add noise, etc)

Who cares if we can’t download it?

Who cares if we can’t download it?



Average household download speed (Kb/s)

SOURCE: TRADINGECONOMICS.COM | AKAMAI

The 2010s: the decade of domain applications

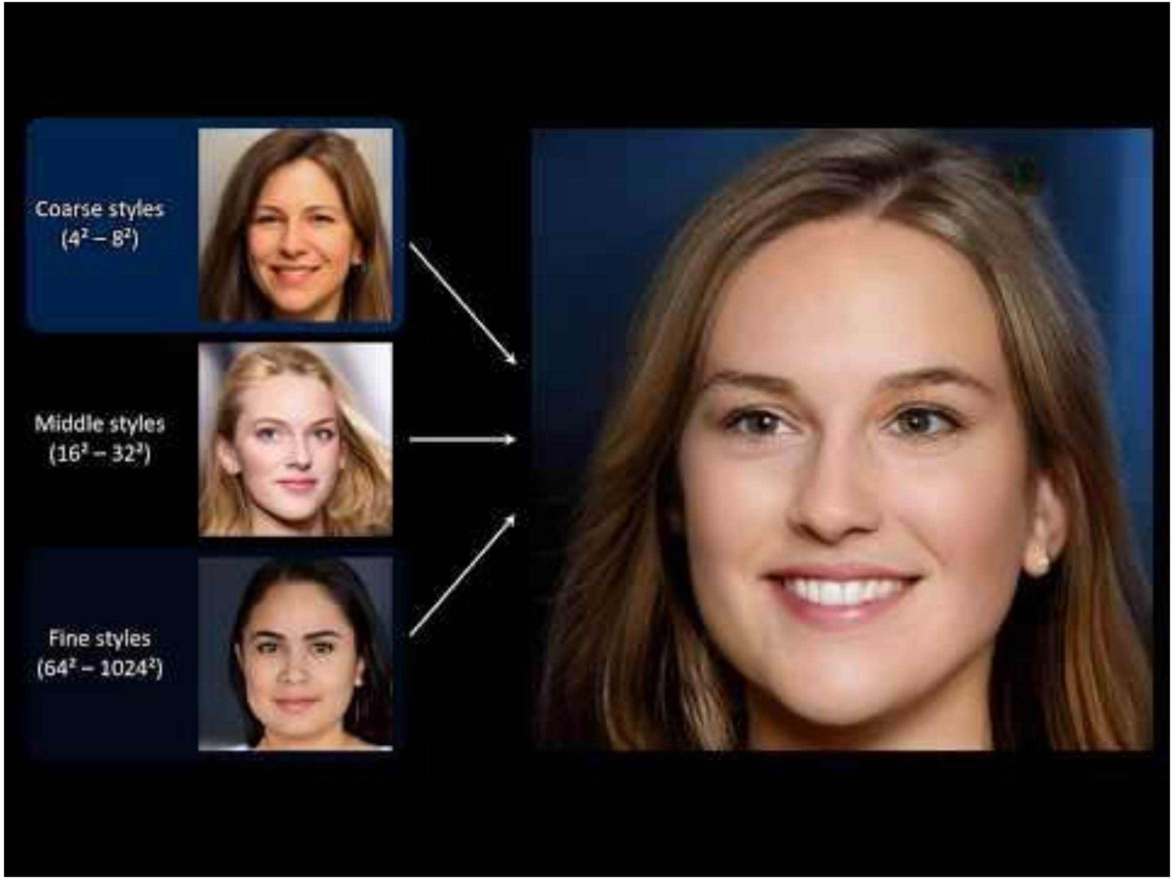
1. ~~They couldn't solve any problem that wasn't linearly separable.~~
2. ~~Backpropagation takes forever to converge!~~
3. Variable-length problems cause gradient problems!
4. Data is rarely labeled!
5. Neural nets are uninterpretable!

using domain-specific architectures

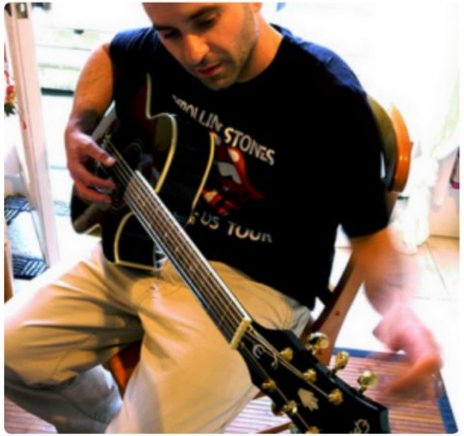
The 2010s: the decade of domain applications

1. ~~They couldn't solve any problem that wasn't linearly separable.~~
2. ~~Backpropagation takes forever to converge!~~
3. Variable-length problems cause gradient problems!
    a. Solved by the forget-gate.
4. Data is rarely labeled!
    a. Addressed by DQN, SOMs.
5. Neural nets don’t use information well
    a. Attention allows the focus on a small number of informative items.
    b. Encoders create a more compact, semantic representation

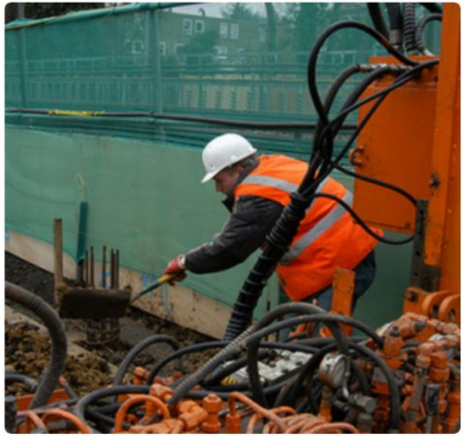
Computer Vision, Style Transfer GANs (2019)



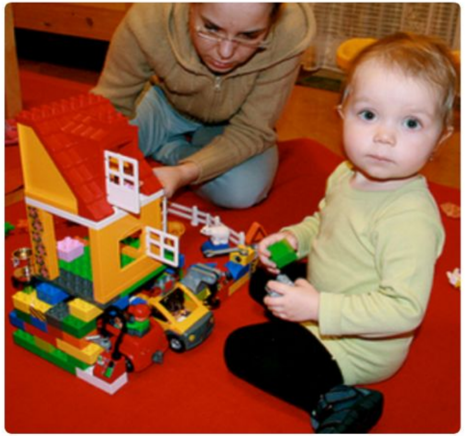
Natural Language Processing/CV - Image Captioning



"man in black shirt is playing guitar."



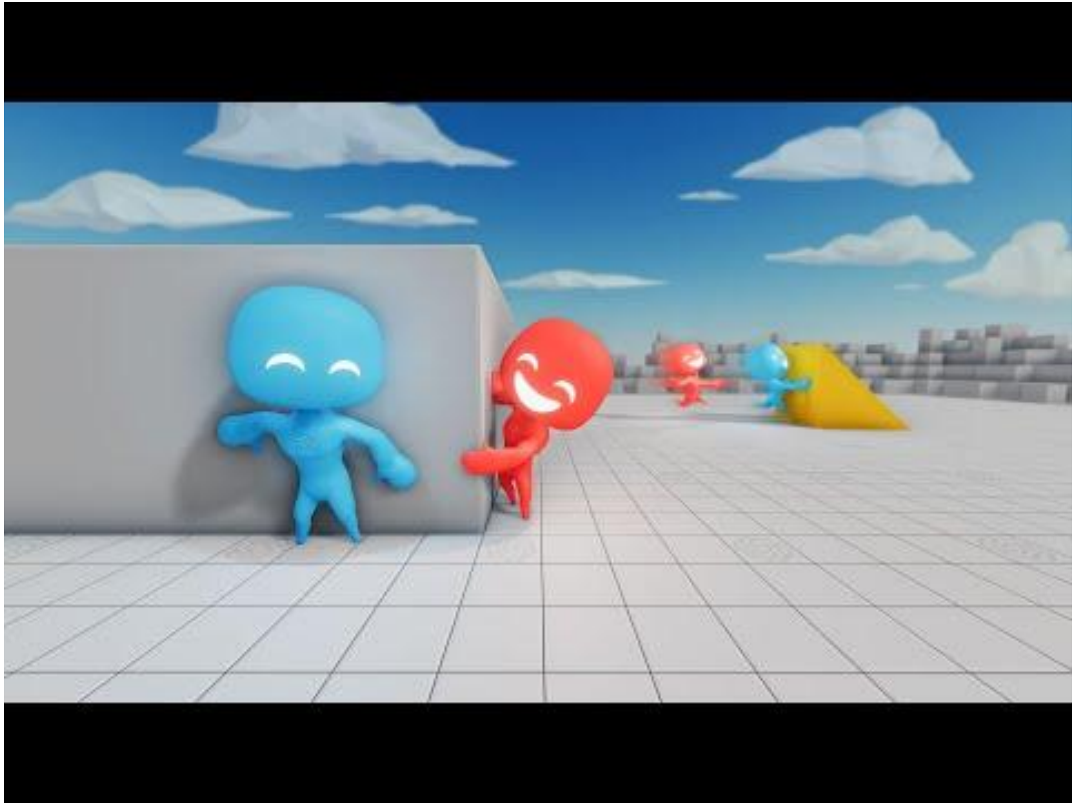
"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."

Transformers/ GPT-3

Deep Reinforcement Learning - Hide and Seek



New decade, old problems

1. Extrapolates poorly when the dataset is too specialized
2. Poor transfer across domains
3. Hard to explain and audit
4. Still too data-hungry
5. Is about the past, not the future
6. And many, many more.

"There is almost as much BS being written about a purported impending AI winter as there is around a purported impending AGI explosion." -- Yann Lecun, FAIR

Societal Implications of DL

Deep Learning in Society

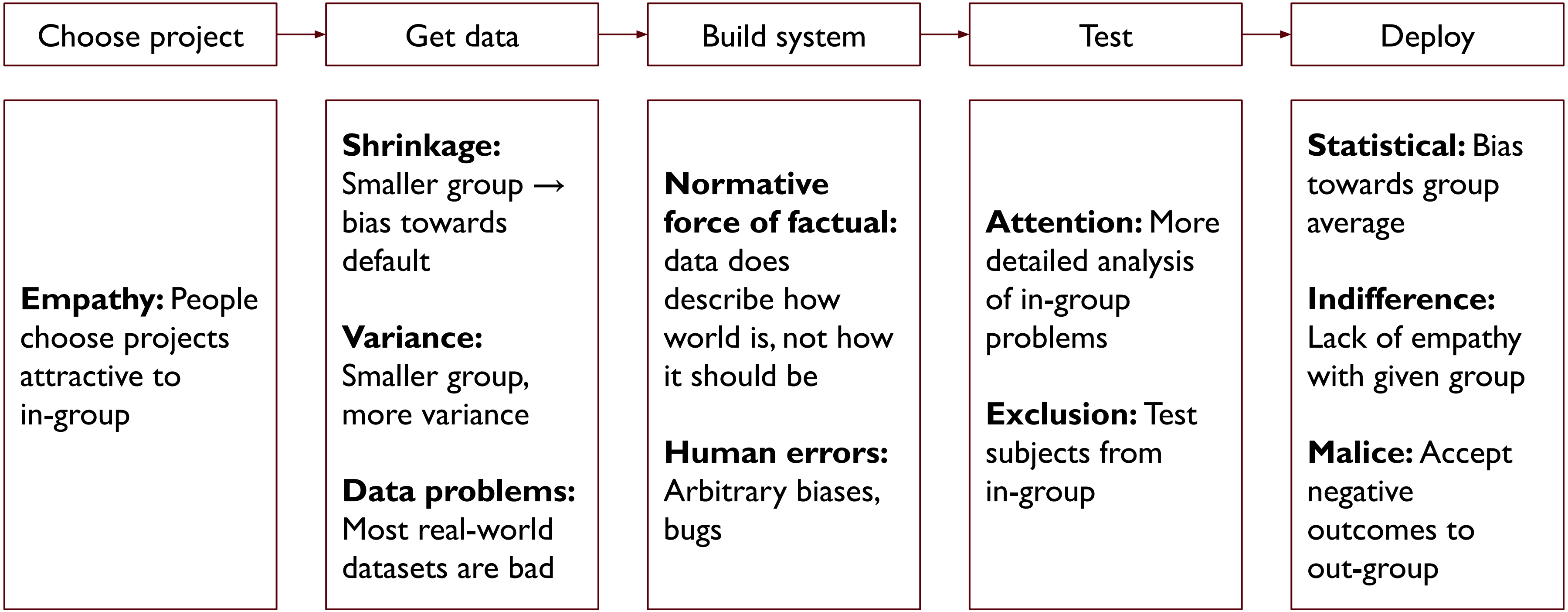
- Credit card applications
- Informs bail, patrol, and criminal sentencing
- Resume screening
- Health risk assessment
- Google search
- Facebook’s news feed
- …

The Ethics of Pothole Detection

Implications of DL - Why should we care?

- You understand the pipeline and the models, others might not
- You have the power to change things and voice potential issues
- You might actually be held accountable for it
- It’s the morally right thing to do (in my opinion)

Where can things go wrong?



Choose project → Get data → Build system → Test → Deploy

Empathy: People choose projects attractive to in-group

Shrinkage: Smaller group → bias towards default

Variance: Smaller group, more variance

Data problems: Most real-world datasets are bad

Normative force of factual: data does describe how world is, not how it should be

Human errors: Arbitrary biases, bugs

Attention: More detailed analysis of in-group problems

Exclusion: Test subjects from in-group

Statistical: Bias towards group average

Indifference: Lack of empathy with given group

Malice: Accept negative outcomes to out-group

DL Ethics in CIS 522

We don't have all the answers - but we have lots of interesting (and difficult) questions! ⟶ About balancing fairness and accuracy, implications on society, trade offs, data, …

Discussions in pods, homework assignments, hopefully guest lecturers

This course has a nonstandard format

Trust us.

The format seems weird.

You will learn more. And be able to solve real problems at end of course

Looking forward

- Do your two worksheets by Monday 1pm each week*
- Prepare questions
    - For your pod
    - Or the next live session
- Give feedback!

*this week will only have one worksheet (W1D1)