Introduction to Neural Networks in Computer Vision

Danna Gurari
University of Colorado Boulder
Fall 2023

Review

- Last week:
  - Computer vision: origins
  - What makes computer vision hard?
  - Research in computer vision
  - Course logistics
- Assignments (Canvas)
  - New reading assignments coming out today due the next two weeks
- Questions?

Today’s Topics

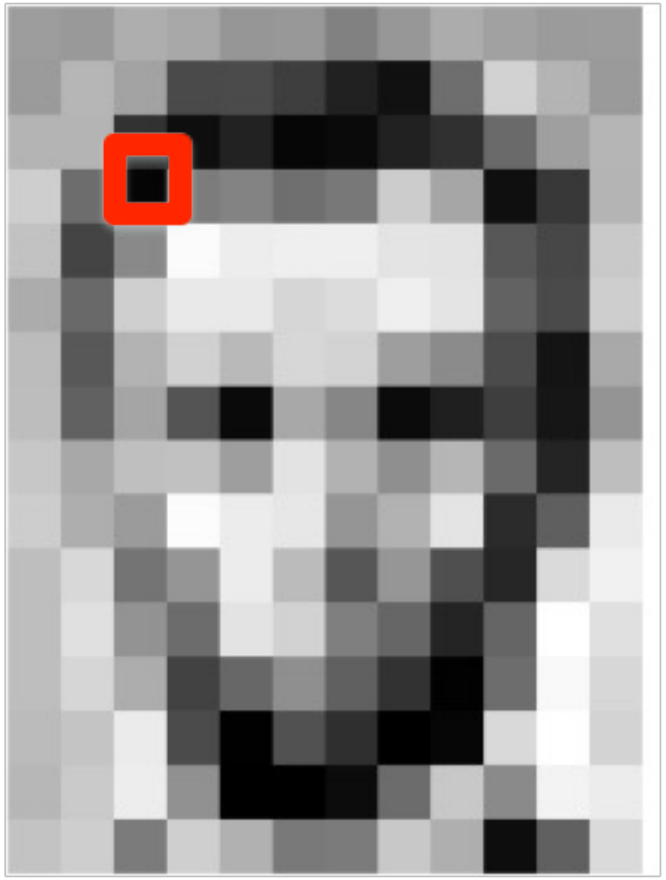
- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Fundamentals of a neural network architecture
- Training deep neural networks

Today’s Topics

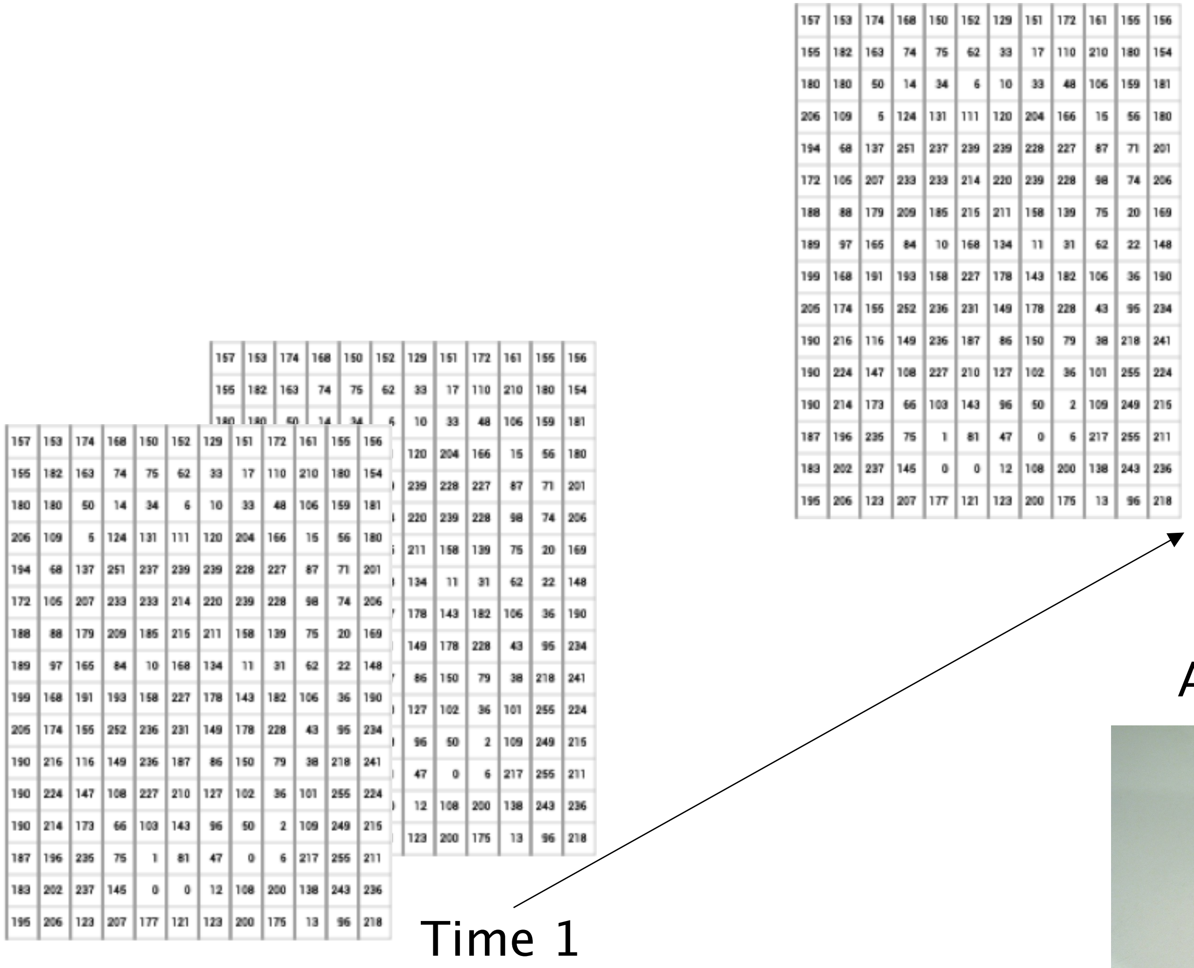
- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Fundamentals of a neural network architecture
- Training deep neural networks

Recall What a Machine Observes: Digital Image

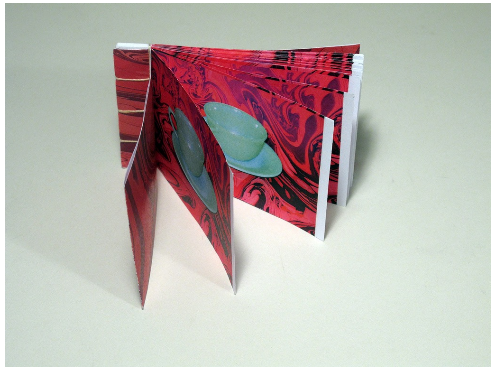
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	[illegible]	5	[illegible]	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



Recall What a Machine Observes: Digital Video

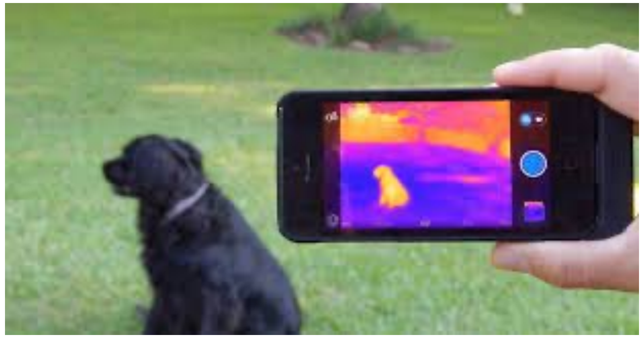


Analogous to:

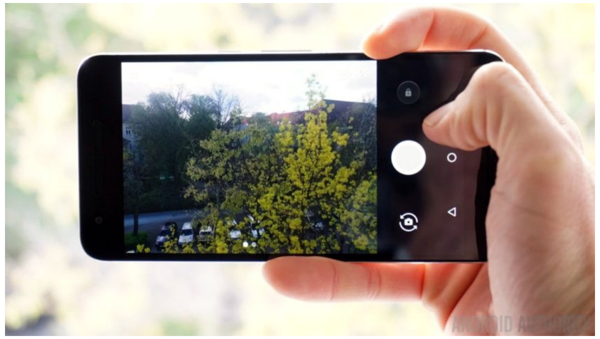


Many Ways to Create Digital Images and Videos

Ultrasound



Infrared



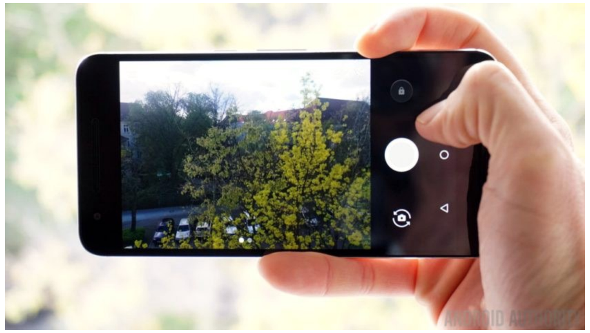
Visible

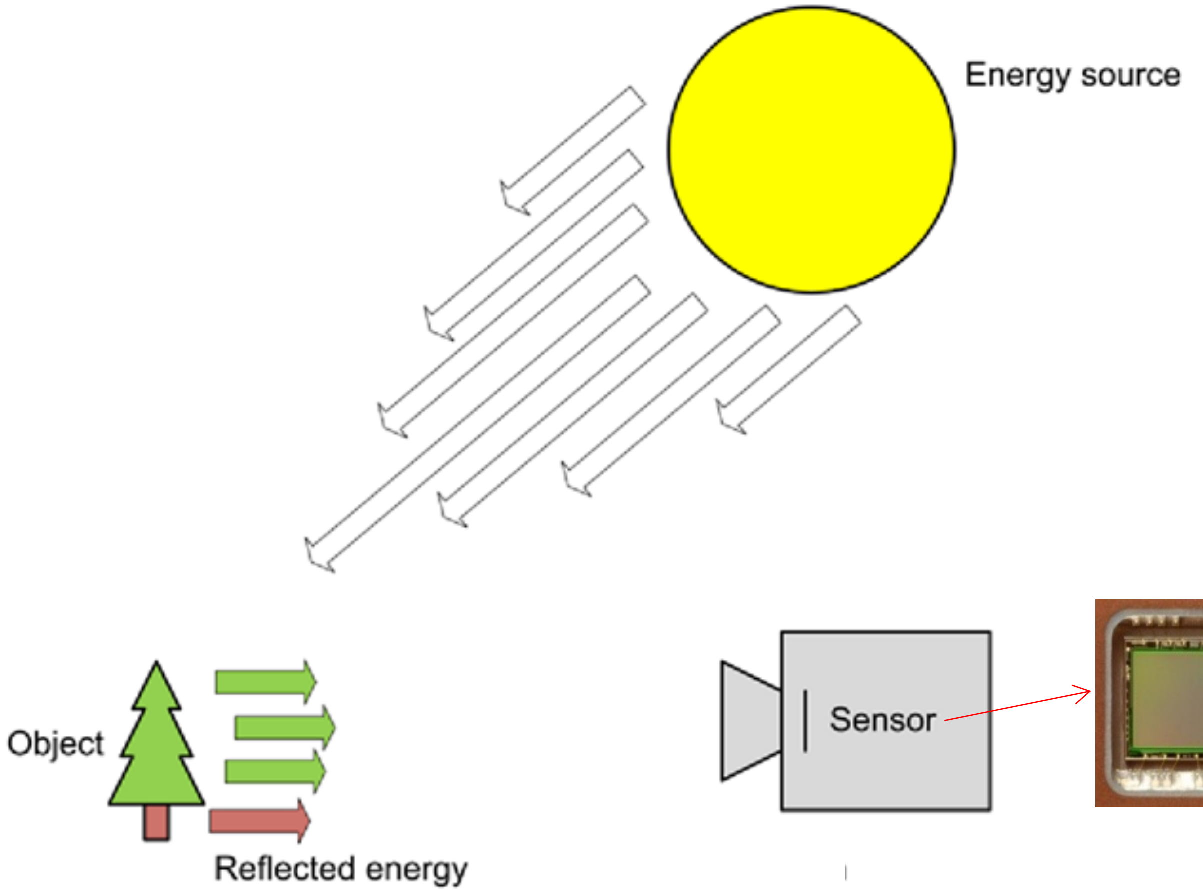
X-ray

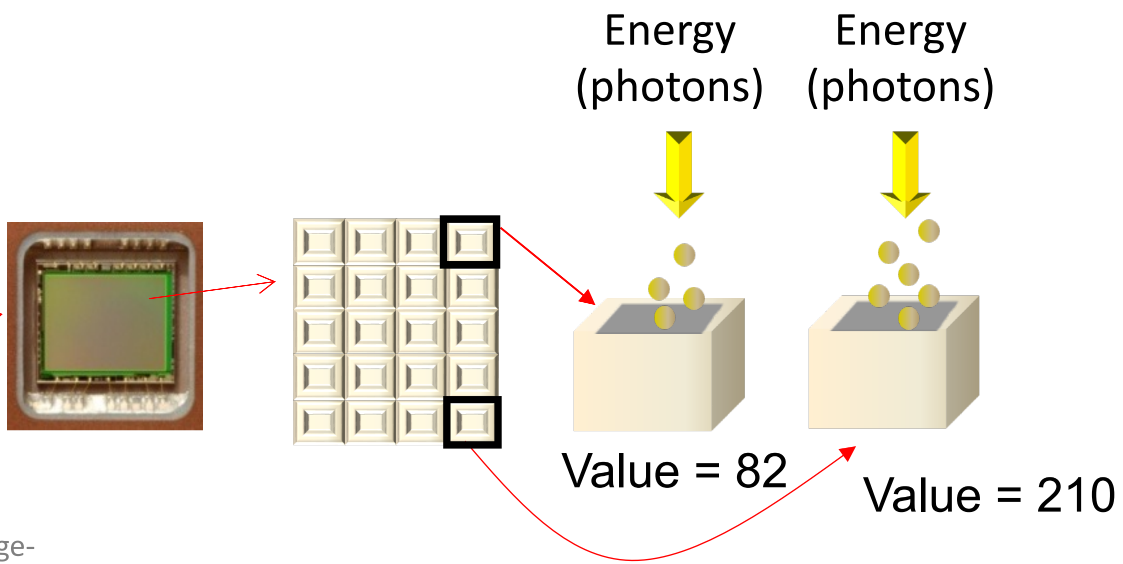
Microscopy

Many Ways to Create Digital Images and Videos

e.g., seeing what is visible to the naked human eye



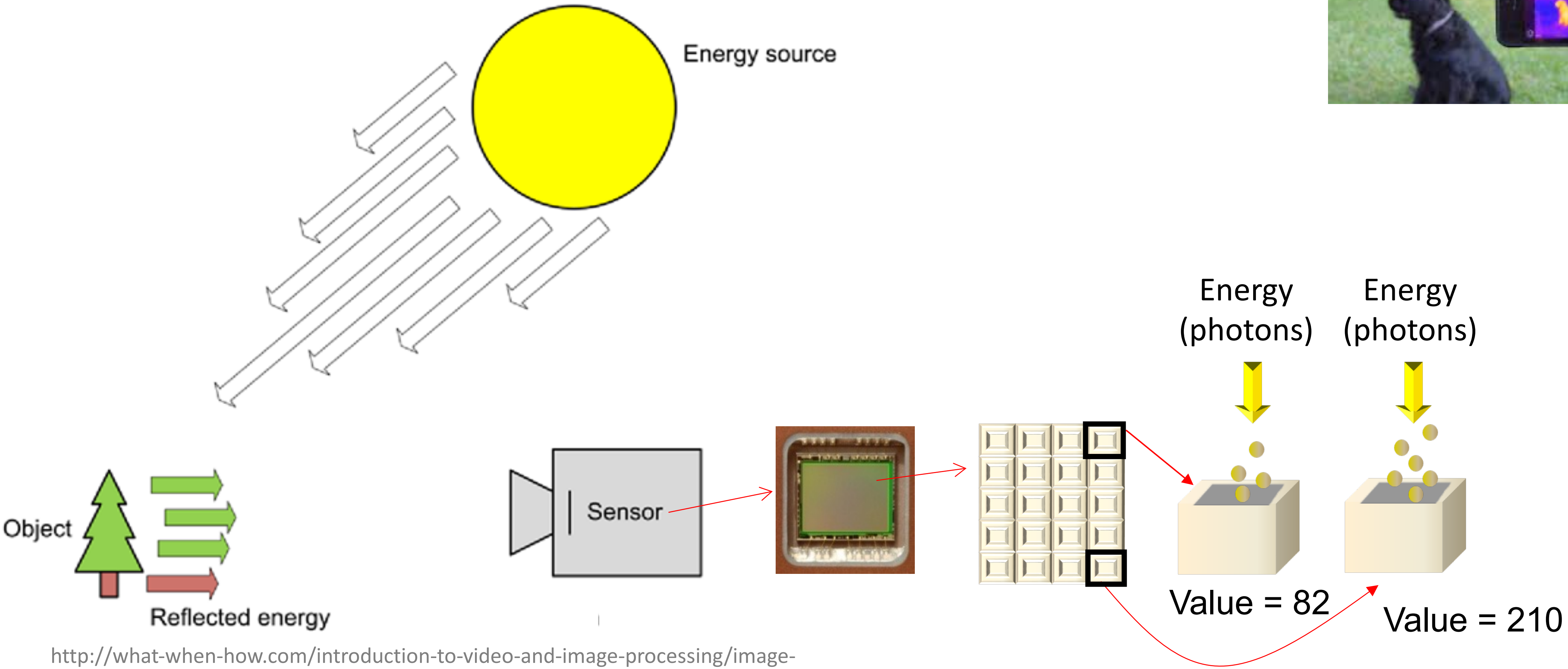




http://what-when-how.com/introduction-to-video-and-image-processing/image-acquisition-introduction-to-video-and-image-processing-part-1/

Many Ways to Create Digital Images and Videos

e.g., seeing what is invisible to the naked human eye with infrared

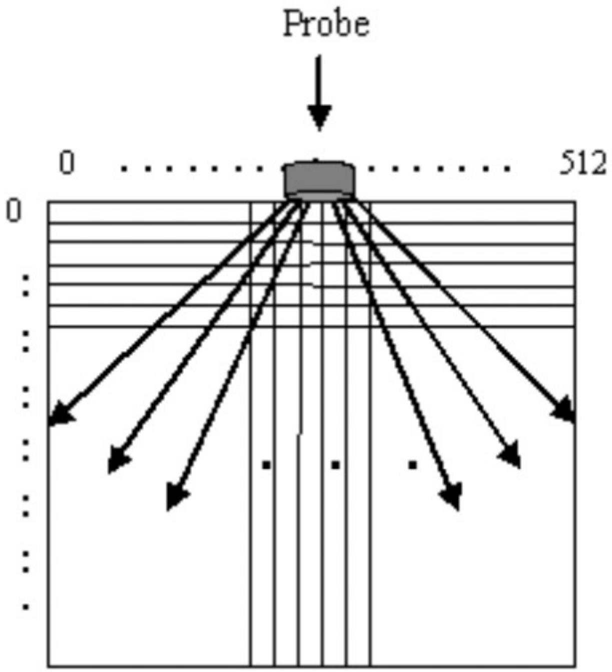


http://what-when-how.com/introduction-to-video-and-image-processing/image-acquisition-introduction-to-video-and-image-processing-part-1/

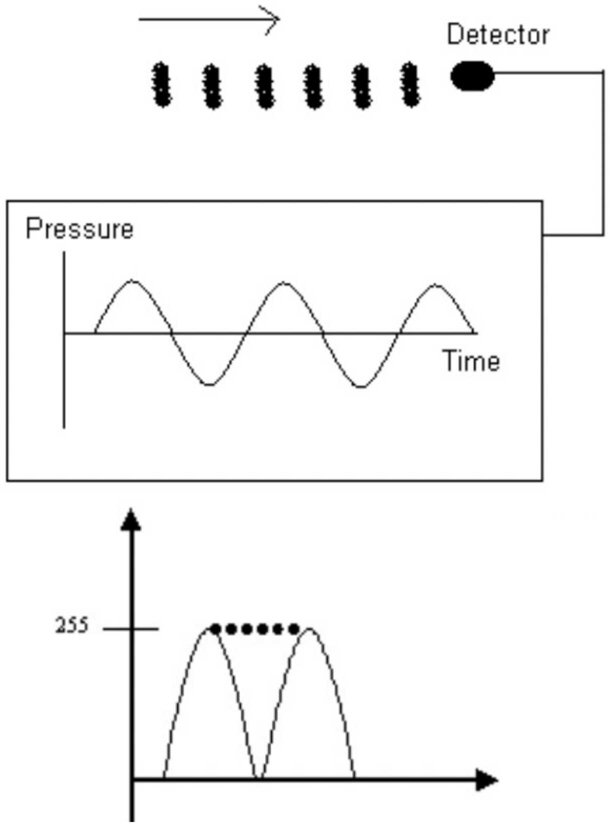
Many Ways to Create Digital Images and Videos

e.g., seeing what is invisible to the naked human eye with sound

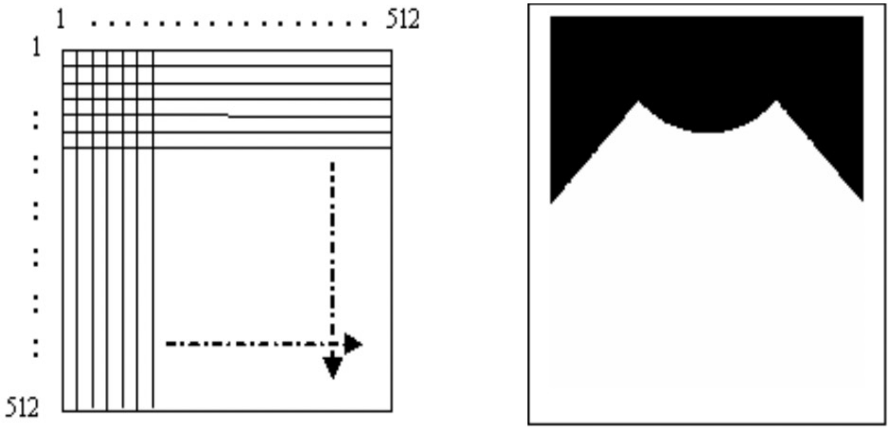
1. Sound wave generation



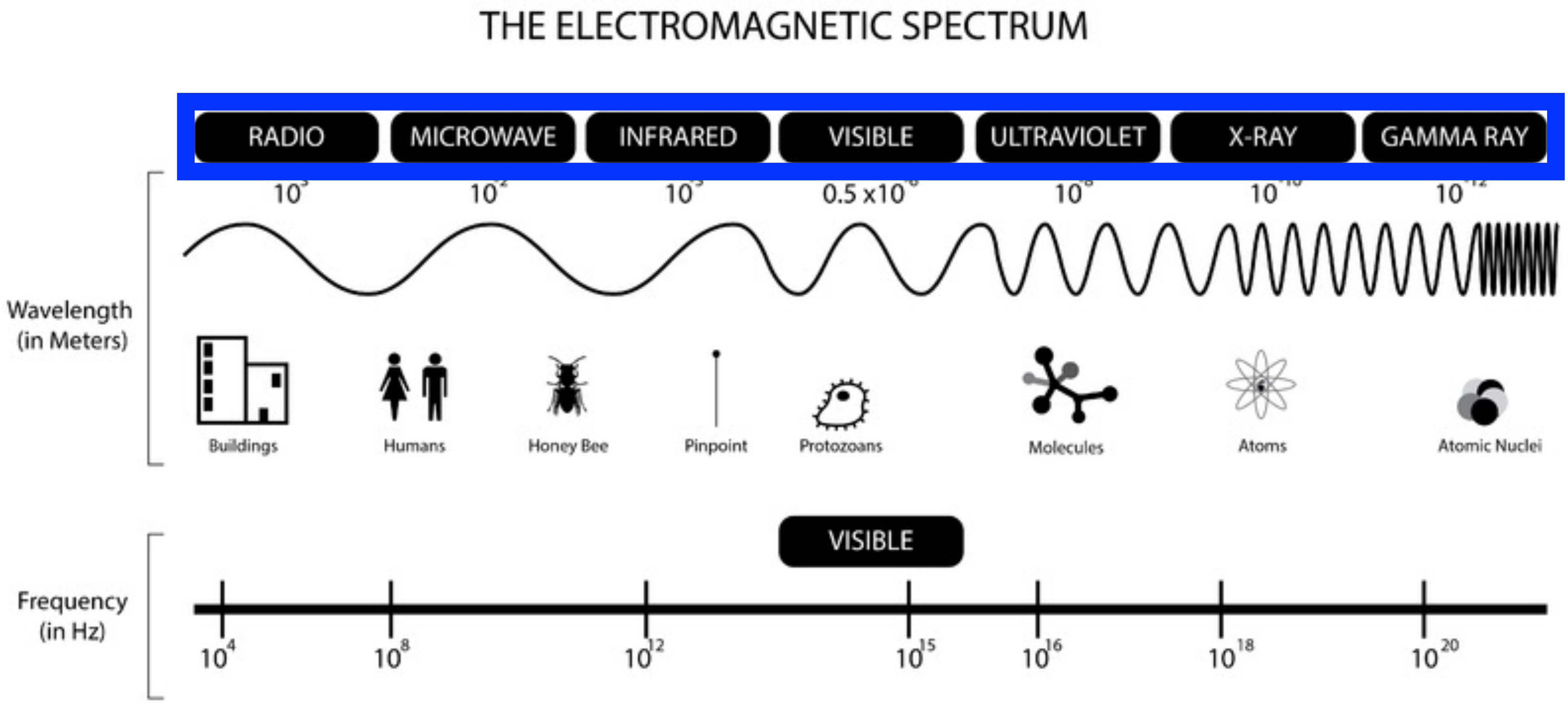
2. For each reflected sound wave, (a) record and (b) digitize to pixel values



3. Convert digitization to image



Many Ways to Create Digital Images and Videos

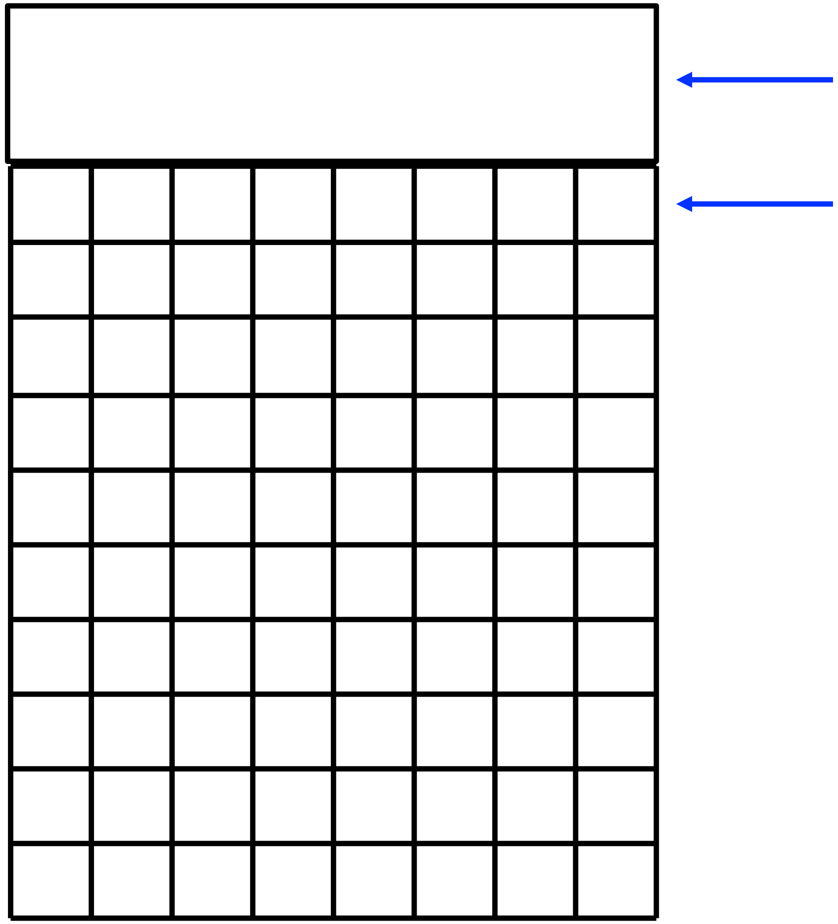


My Focus in My Career

- 2004-2005: Washington University - Ultrasound
- 2005-2007: Raytheon (NPOESS) - Satellite
- 2007-2010: Boulder Imaging - Visible & Infrared
- 2010-2015: Boston University - Microscopy
- 2015-Present: Many more types!

Many Ways to Record Digital Visual Data

e.g., Roughly, can think of file formats as headers followed by pixel values (e.g., jpg, png)



Header: Instructions to parse file

Table: Pixel values
(e.g., RGB, CMYK, Lab, grayscale)

Scale of Vision Acquisition

- 5.8B cameras owned by 4B people with 89% taking pictures resulting in over 1 trillion pictures [2014 statistics] [1]

- > 85% of internet data in the form of images and videos [2]

[1] https://communities-dominate.blogs.com/brands/2014/08/camera-stats-world-has-48b-cameras-by-4b-unique-camera-owners-88-of-them-use-cameraphone-to-take-pic.html
[2] https://sevenshinestudios.wordpress.com/computer-vision-and-deep-learning/

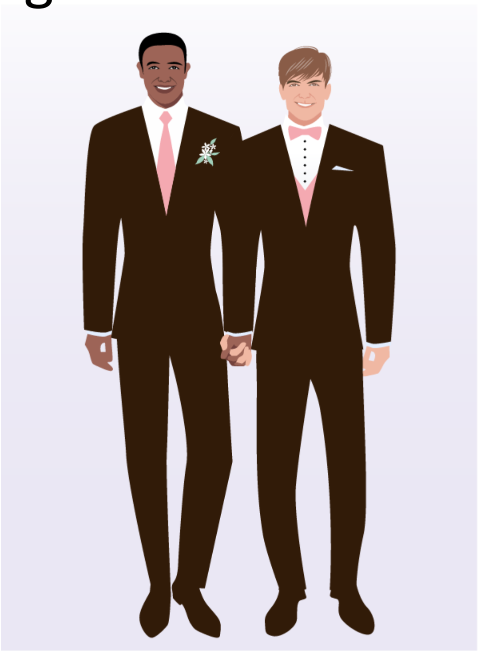
Today’s Topics

- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Fundamentals of a neural network architecture
- Training deep neural networks

Status Quo Until 2012

Algorithm Dataset

Algorithm Dataset



Algorithm Dataset

Algorithm

Dataset

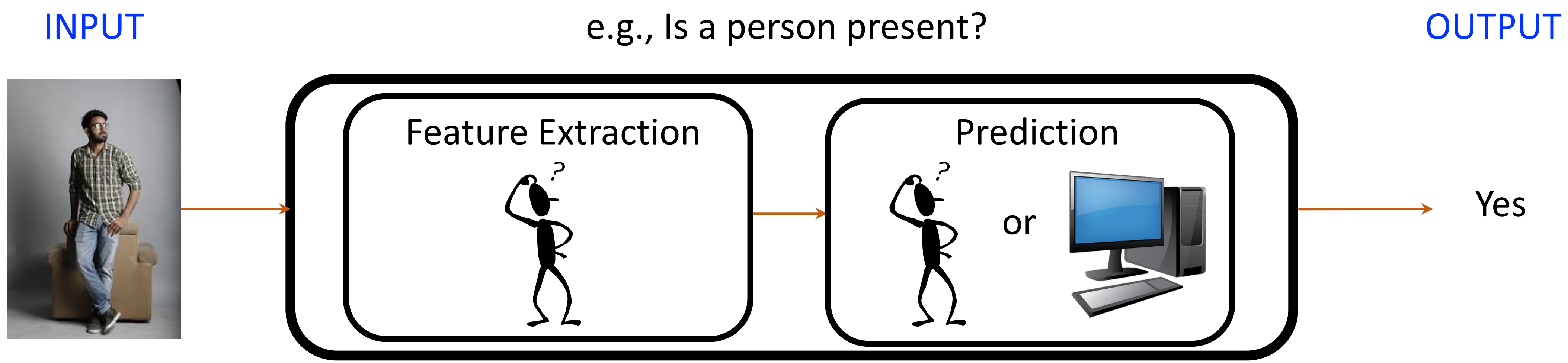
Datasets tended to be relatively small (e.g., 10s or 100s of examples)

Status Quo Until 2012: Datasets

- Authors created datasets primarily with their cameras, purchasing from companies, or downloading images from the Internet
- What’s wrong with this approach?
  - Unable to perform “fair” comparison between algorithms
  - Lacks a community around a shared goal

Status Quo Until 2012: Algorithms

- An engineer manually designs methods to interpret an image



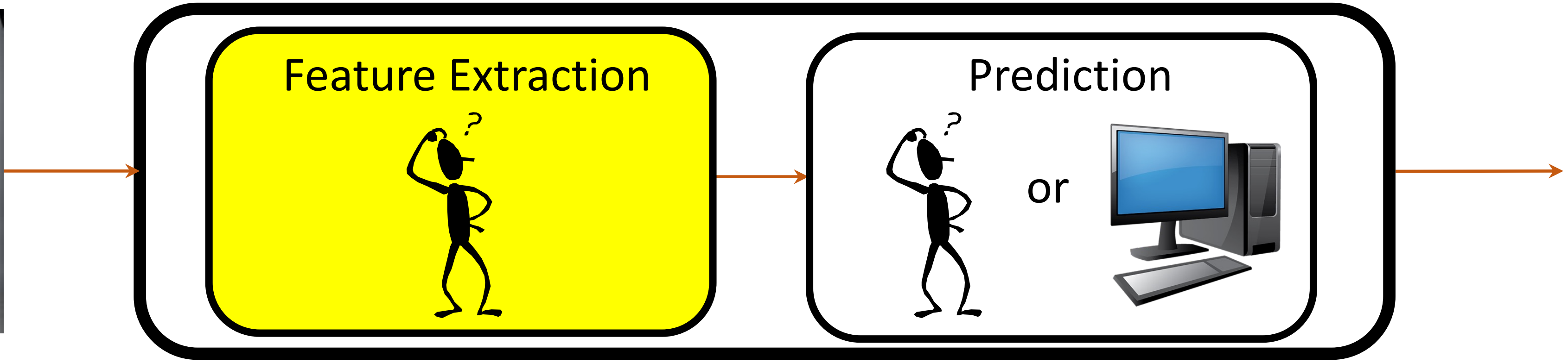
Status Quo Until 2012: Algorithms

- An engineer manually designs methods to interpret an image

INPUT

e.g., Is a person present?

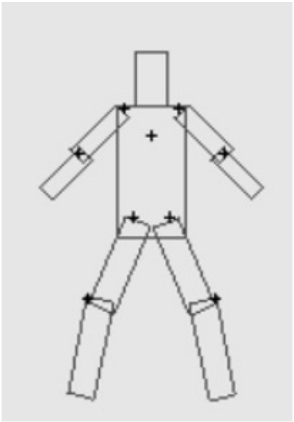
OUTPUT



Yes

What features would help predict yes/no?

e.g., corners, lines, and model of expected body parts as connected shapes



e.g., Pedro F Felzenszwalb and Daniel P Huttenlocher, IJCV 2004

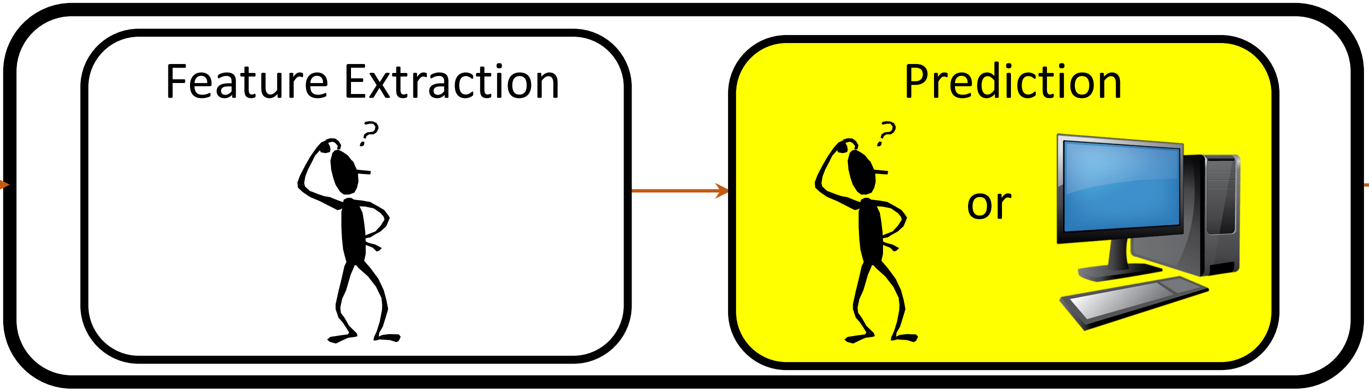
Status Quo Until 2012: Algorithms

- An engineer manually designs methods to interpret an image

INPUT

e.g., Is a person present?

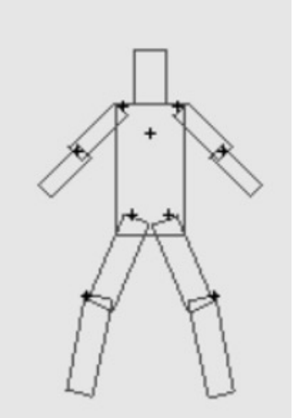
OUTPUT



Yes

What rules should be used to predict yes/no?

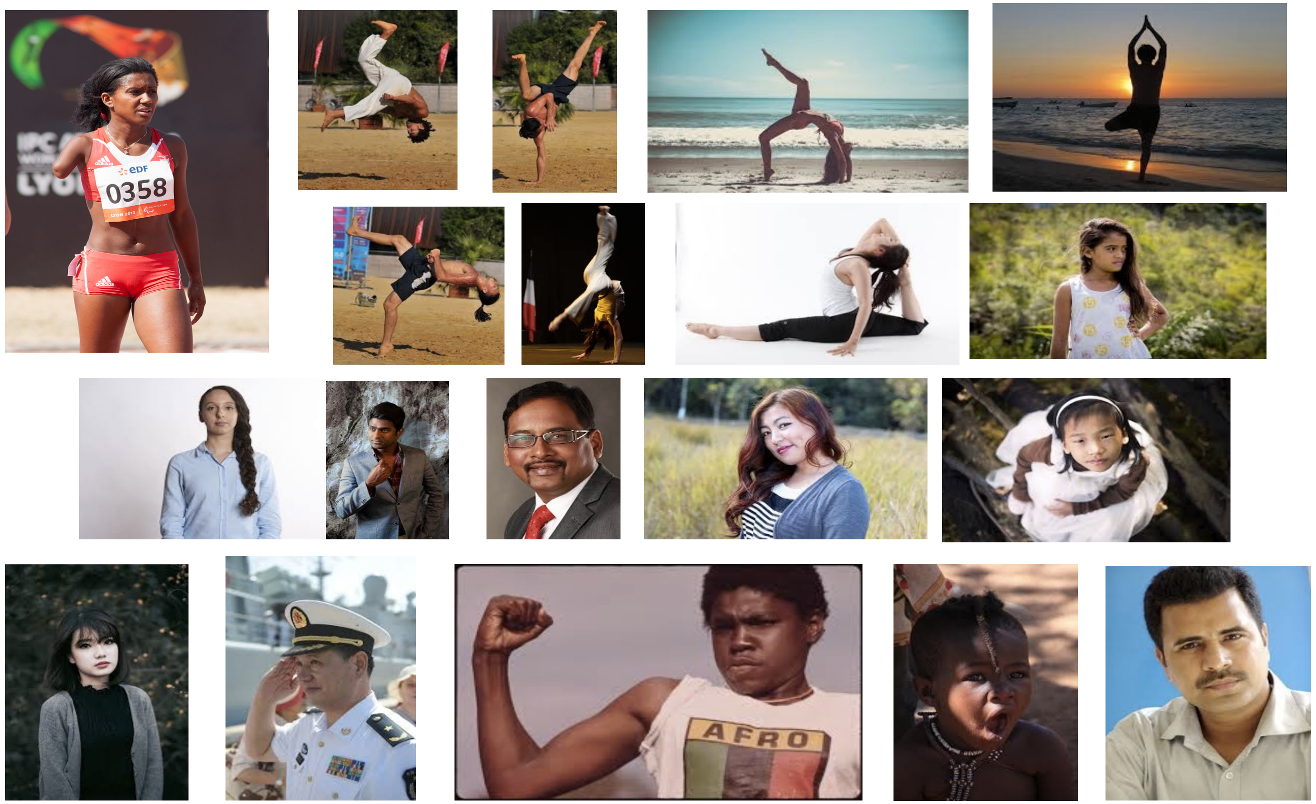
e.g., choose threshold for number of model parts detected or develop machine learning model that predicts from handcrafted features



e.g., Pedro F Felzenszwalb and Daniel P Huttenlocher, IJCV 2004

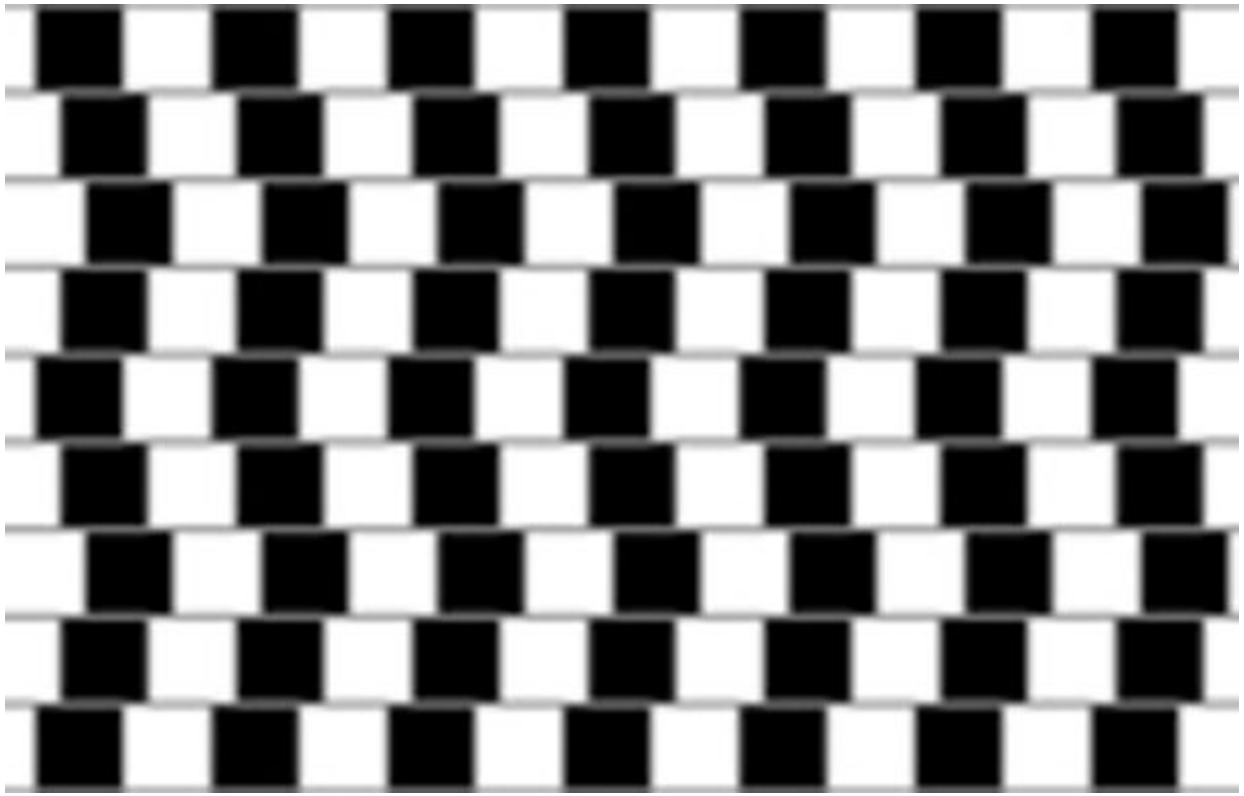
Limitations of Handcrafted Methods

- Challenging for engineers to design effective features (and rules) for ALL examples (for every computer vision problem)!



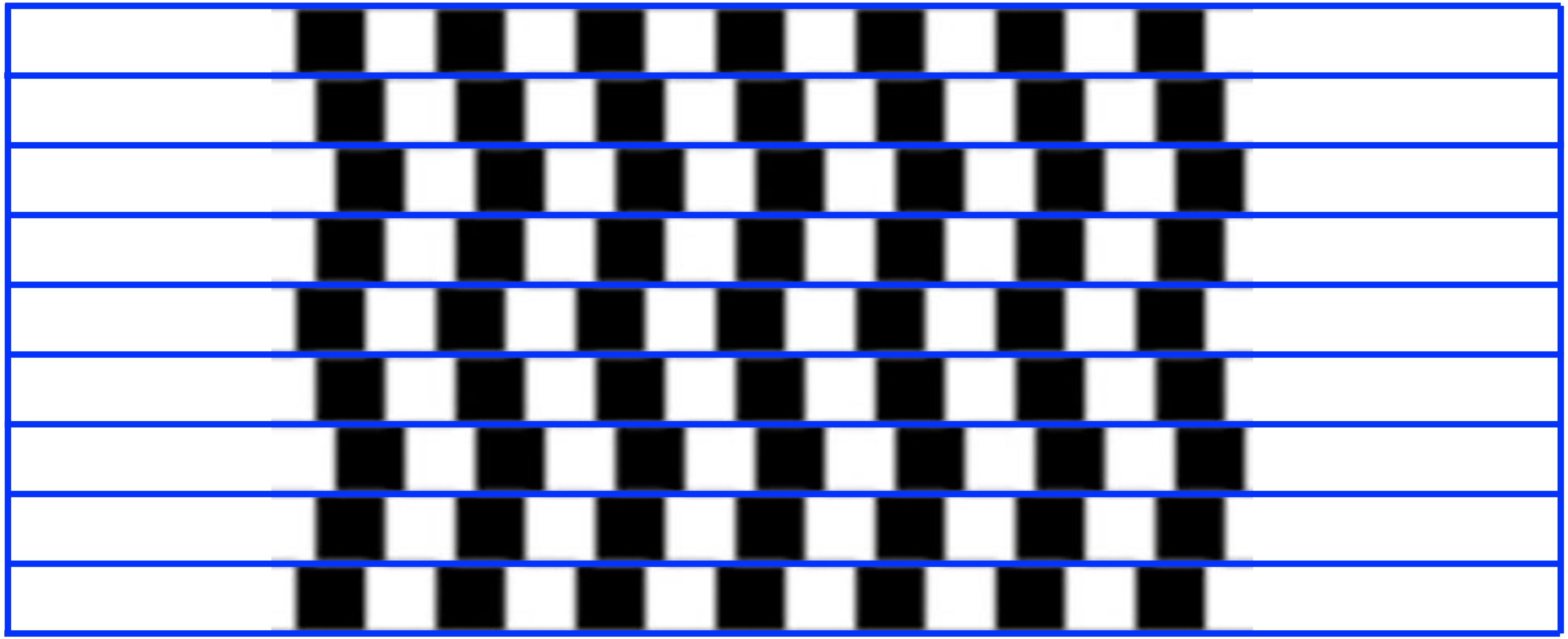
Limitations of Handcrafted Methods

e.g., are these lines parallel?



Limitations of Handcrafted Methods

e.g., are these lines parallel?

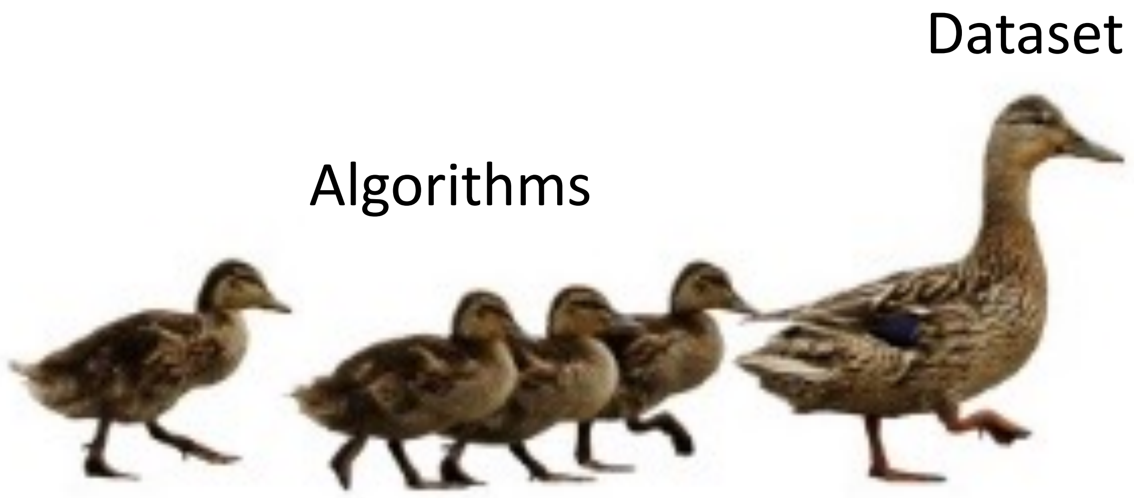


Limitations of Handcrafted Methods

1. It is hard to hand-craft a complete set of methods

2. We, as humans, may not devise the best rules for a machine since our brains (unconsciously) pre-process the data we sense

Status Quo Since 2012

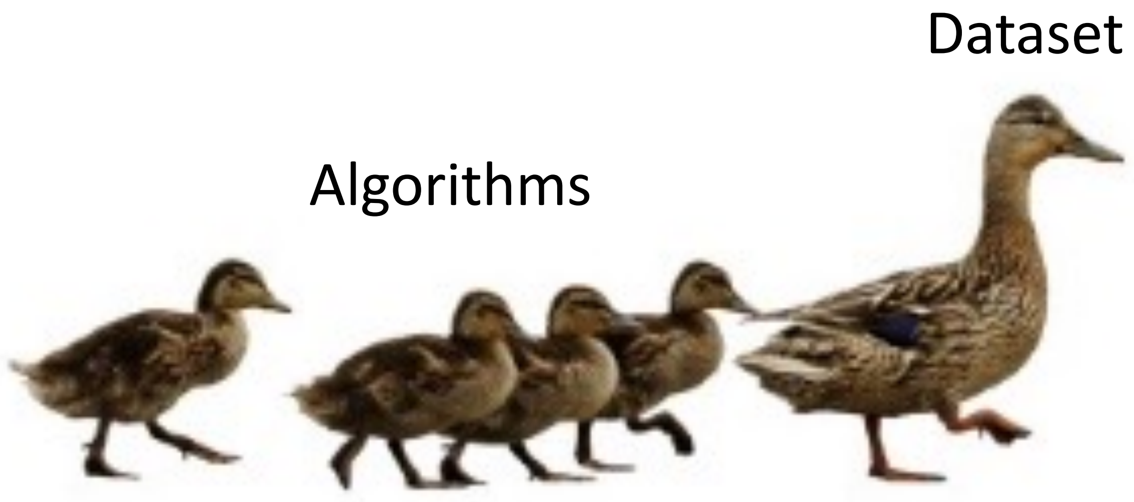


Datasets tend to be large (e.g., thousands to billions of examples)

Image Source: http://larryzitnick.org/Talks/CVPR15_Dataset.pptx

Status Quo Since 2012

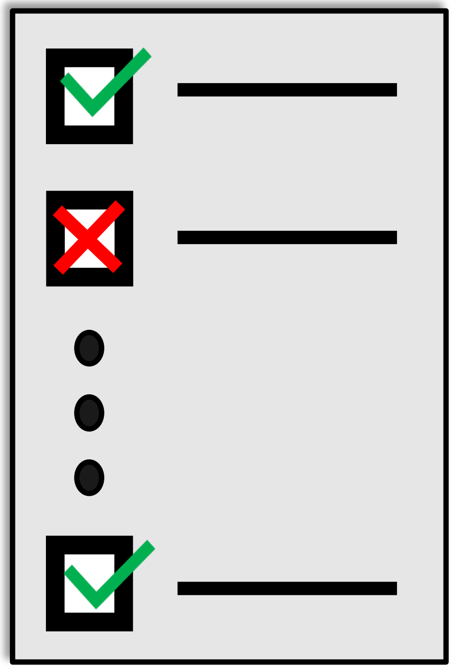
What do you think prompted this shift to large-scale datasets?



Datasets tend to be large (e.g., thousands to billions of examples)

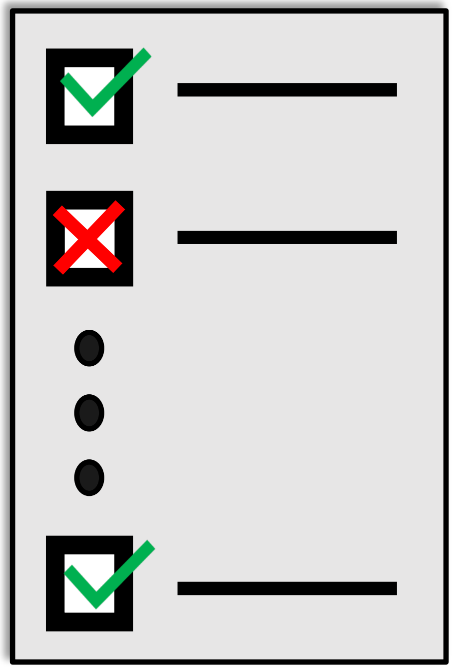
Image Source: http://larryzitnick.org/Talks/CVPR15_Dataset.pptx

Research Since 2012: Dataset Challenges



Create AI Challenges (Tests) Paired With
Public Leaderboards to Track Progress

Research Since 2012: Dataset Challenges



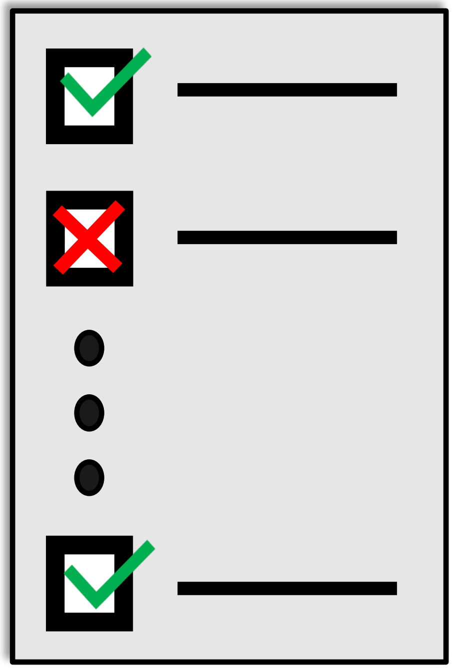
Key components:

1. Publicly-shared test examples without ground truth answers for evaluation

2. Metrics for evaluating algorithm predictions, implemented in an evaluation server

3. Publicly-shared examples with “ground truth” answers to support training and validation

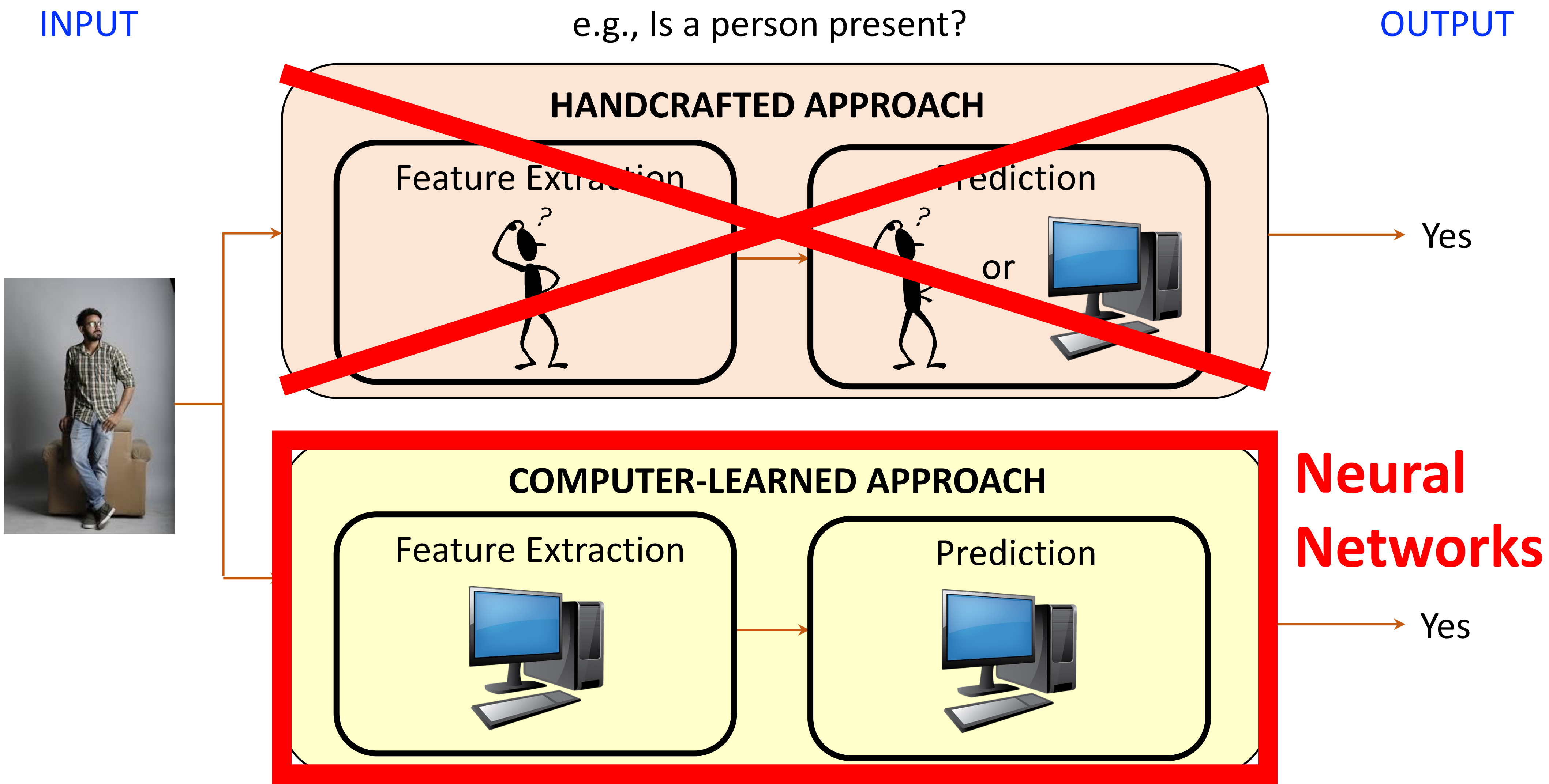
Research Since 2012: Dataset Challenges



Many public dataset challenges and datasets:

- Google Dataset Search
- Kaggle
- Amazon's AWS datasets
- UC Irvine Machine Learning Repository
- Quora.com
- Reddit
- Dataportals.org
- Opendatamonitor.eu
- Quandl.com

Research Since 2012: Algorithms



Today’s Topics

- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Fundamentals of a neural network architecture
- Training deep neural networks

Inspiration: Animal’s Computing Machinery

Neuron

- basic unit in the nervous system for receiving, processing, and transmitting information; e.g., messages such as…

“hot”

https://www.clipart.email/clipart/don t-touch-hot-stove-clipart-73647.html

“loud”

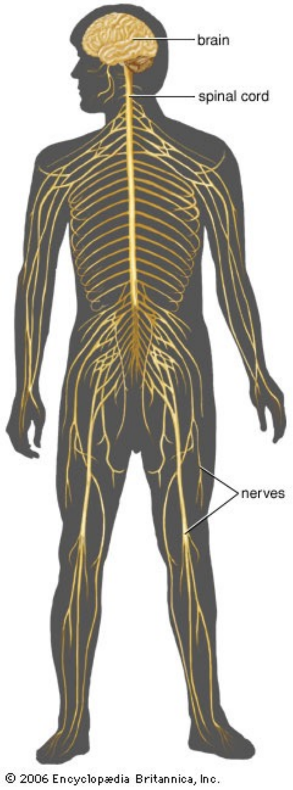
https://kisselpaso.com/if-the-sun-city-music-fest-gets-too-loud-there-is-a-phone-number-you-can-call-to-complain/

“spicy”

https://www.babycenter.com/404_when-can-my-baby-eat-spicy-foods_1368539.bc

Inspiration: Animal's Computing Machinery

https://en.wikipedia.org/wiki/Nematode#/media/File:CelegansGoldsteinLabUNC.jpg

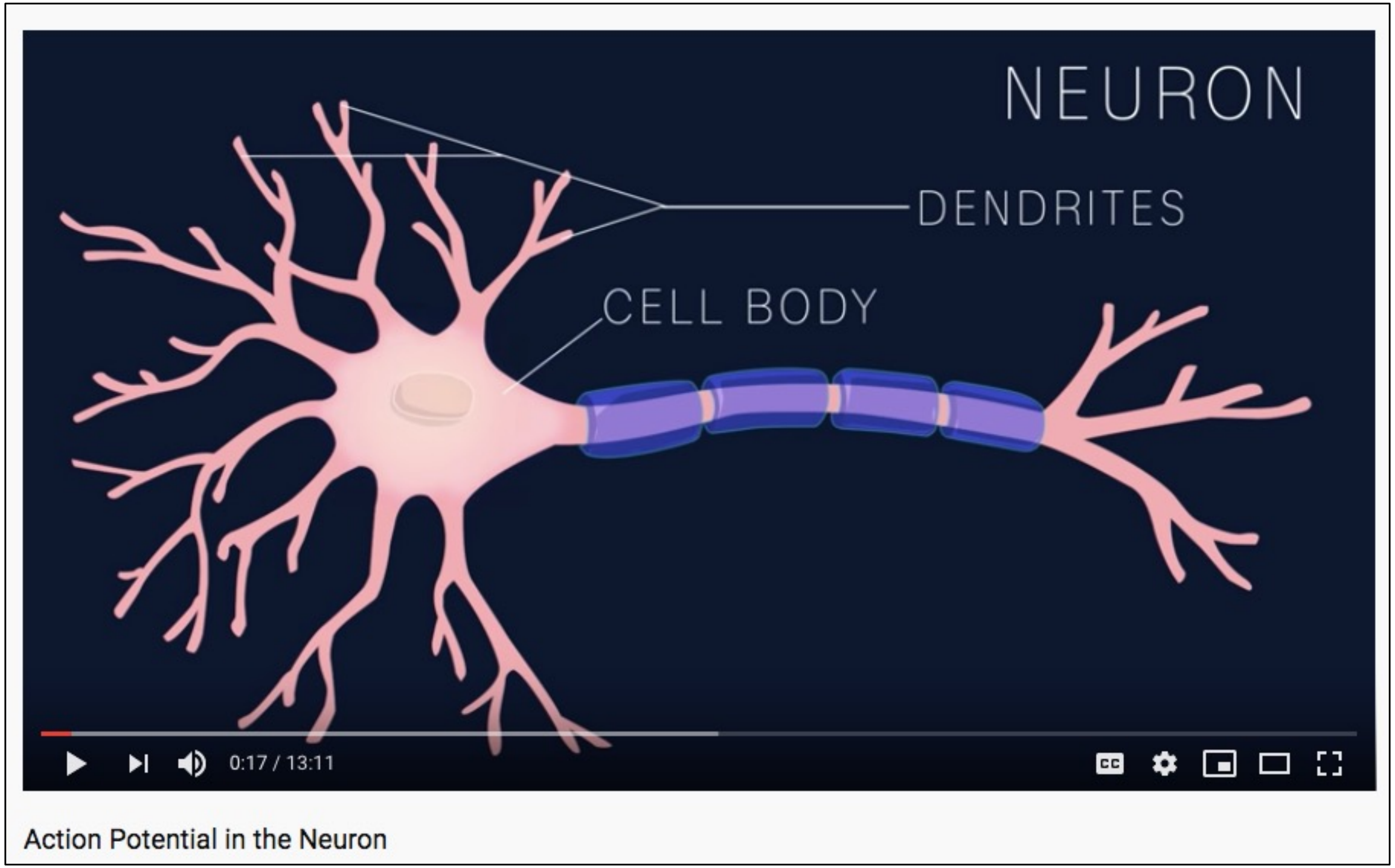


https://www.britannica.com/science/human-nervous-system

Nematode worm: 302 neurons

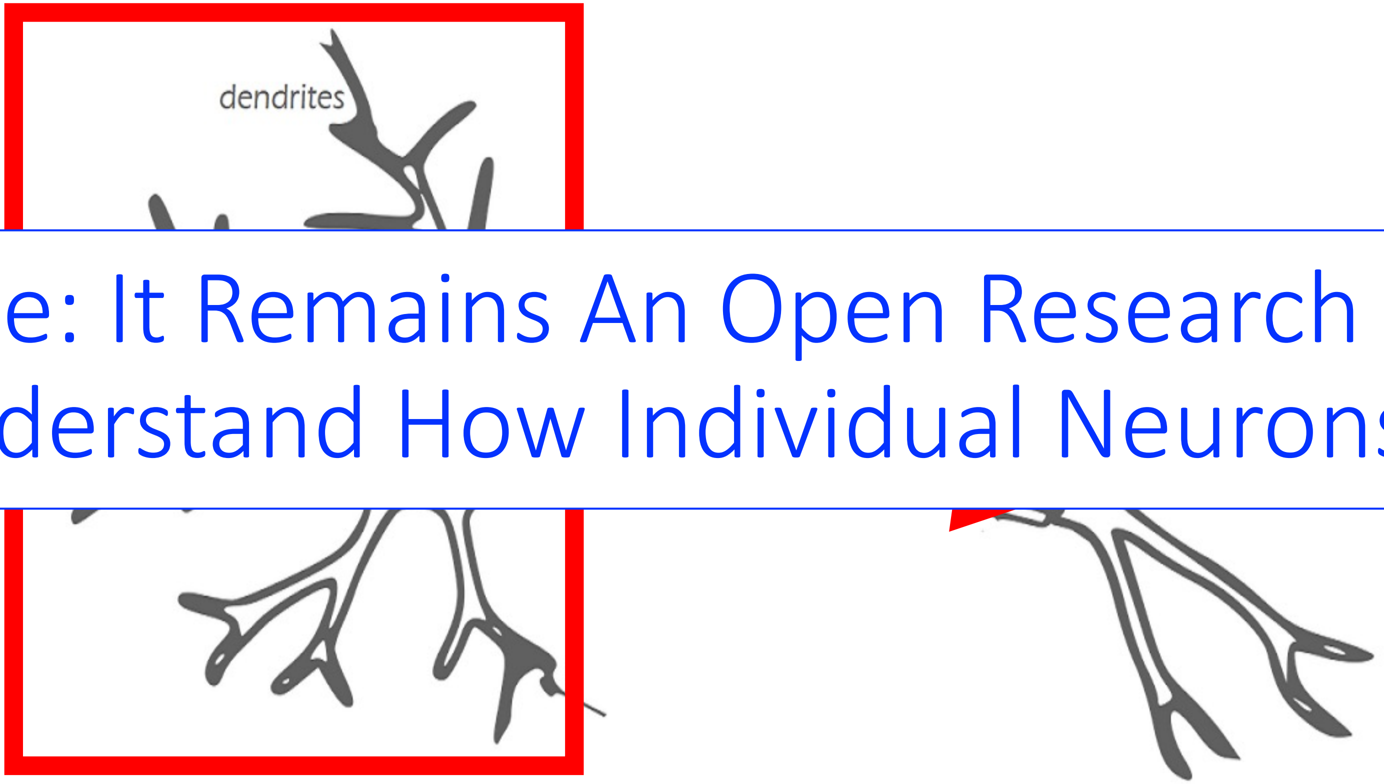
Human: ~100,000,000,000 neurons

Inspiration: Animal’s Computing Machinery



Demo (0-1:14): https://www.youtube.com/watch?v=oa6rvUJlg7o

Inspiration: Basic Understanding of Neurons



Sidenote: It Remains An Open Research Problem to Understand How Individual Neurons Work

- When the input signals exceed a certain threshold within a short period of time, a neuron “fires”
- Neuron “firing” is an “all-or-none” process, where either a signal is sent or nothing happens

Image Source: https://becominghuman.ai/introduction-to-neural-networks-bd042ebf2653

Origins of Neural Networks: Artificial Neurons

1945

1957

Perceptron

1966

Computer Vision

1983

CVPR

1987

ICCV

1990

ECCV

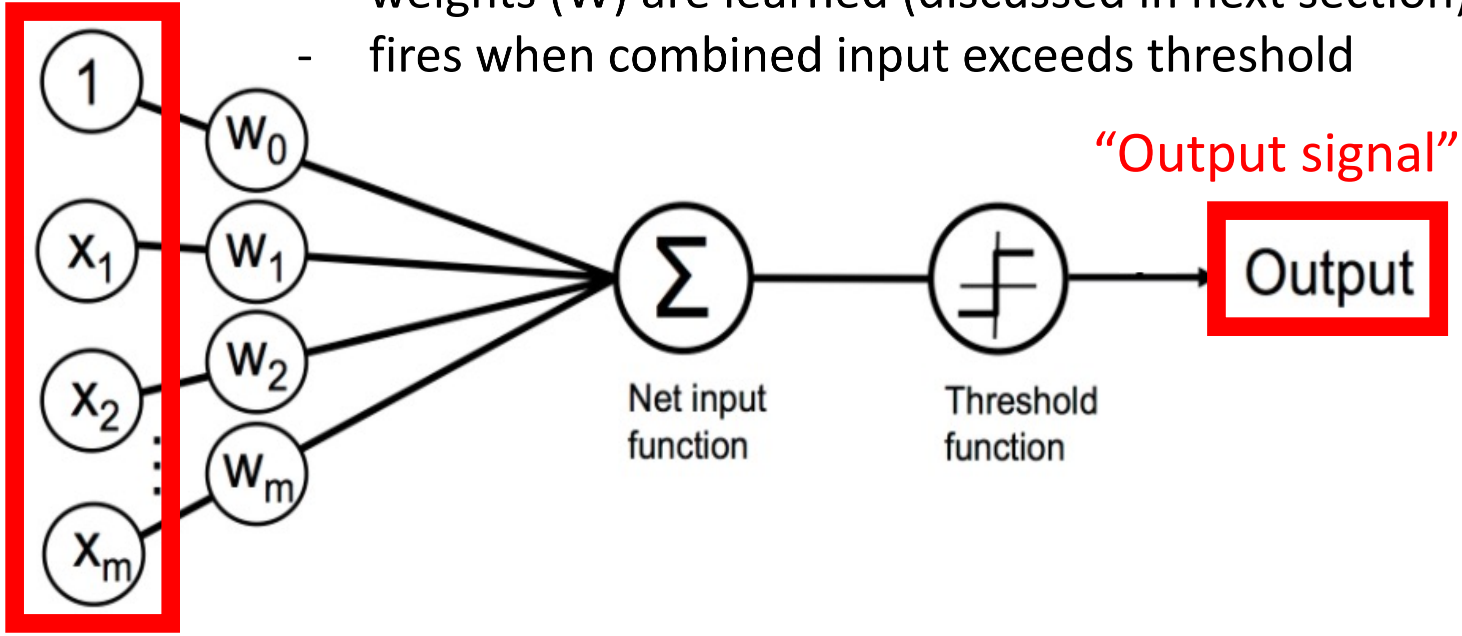
Perceptron (Artificial Neuron)

- weights (W) are learned (discussed in next section)
- fires when combined input exceeds threshold

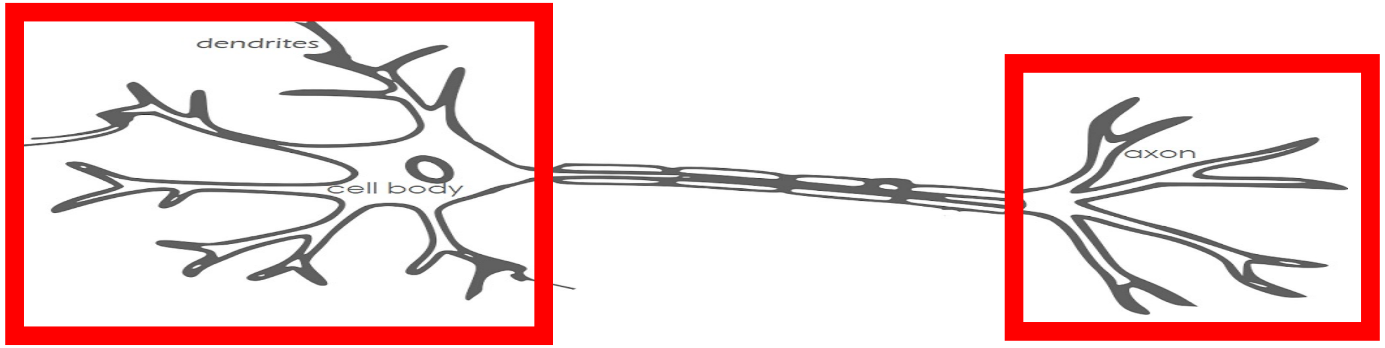
"Input signals"

"Output signal"

Artificial Neuron:



Biological Neuron:



Python Machine Learning; Raschka & Mirjalili

Image Source: https://becominghuman.ai/introduction-to-neural-networks-bd042ebf2653

Rise of Perceptron (Artificial Neuron)

Frank Rosenblatt
(Psychologist)

“[The perceptron is] the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence…. [It] is expected to be finished in about a year at a cost of $100,000.”

1958 New York Times article: https://www.nytimes.com/1958/07/08/archives/new-navy-device-learns-by-doing-psychologist-shows-embryo-of.html

https://en.wikipedia.org/wiki/Frank_Rosenblatt

Fall of Perceptron (Artificial Neuron)

XOR = "Exclusive Or"
- Input: two binary values x_1 and x_2
- Output:
    - 1, when exactly one input equals 1
    - 0, otherwise

x_1	x_2	x_1 XOR x_2
0	0	?
0	1	?
1	0	?
1	1	?

Marvin Minsky and Seymore Papert, Perceptrons, MIT Press, 1969

Fall of Perceptron (Artificial Neuron)

XOR = “Exclusive Or”
- Input: two binary values x_1 and x_2
- Output:
    - 1, when exactly one input equals 1
    - 0, otherwise

x_1	x_2	x_1 XOR x_2
0	0	?
0	1	?
1	0	?
1	1	?

Marvin Minsky and Seymore Papert, Perceptrons, MIT Press, 1969

Fall of Perceptron (Artificial Neuron)

XOR = "Exclusive Or"
- Input: two binary values x_1 and x_2
- Output:
  - 1, when exactly one input equals 1
  - 0, otherwise

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	?
1	0	?
1	1	?

Marvin Minsky and Seymore Papert, Perceptrons, MIT Press, 1969

Fall of Perceptron (Artificial Neuron)

XOR = “Exclusive Or”

- Input: two binary values x_1 and x_2
- Output:
  - 1, when exactly one input equals 1
  - 0, otherwise

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	?
1	1	?

Marvin Minsky and Seymore Papert, Perceptrons, MIT Press, 1969

Fall of Perceptron (Artificial Neuron)

XOR = “Exclusive Or”
- Input: two binary values x_1 and x_2
- Output:
  - 1, when exactly one input equals 1
  - 0, otherwise

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	?

Marvin Minsky and Seymore Papert, Perceptrons, MIT Press, 1969

Fall of Perceptron (Artificial Neuron)

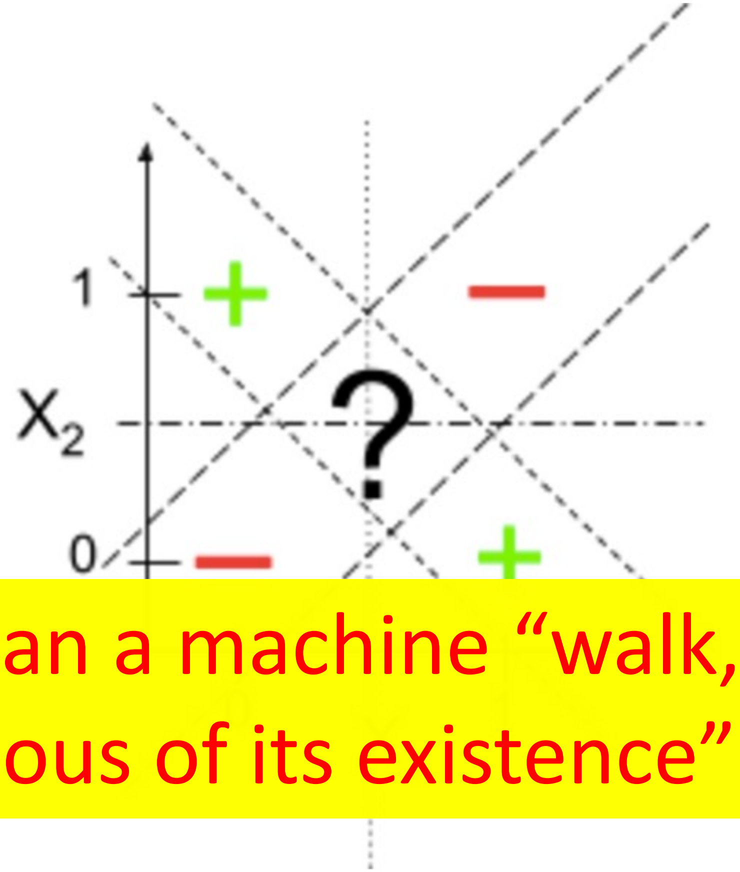
XOR = "Exclusive Or"
- Input: two binary values x_1 and x_2
- Output:
  - 1, when exactly one input equals 1
  - 0, otherwise

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Marvin Minsky and Seymore Papert, Perceptrons, MIT Press, 1969

Fall of Perceptron (Artificial Neuron)

A Perceptron cannot solve XOR problem and so separate 1s from 0s (it's a linear function):



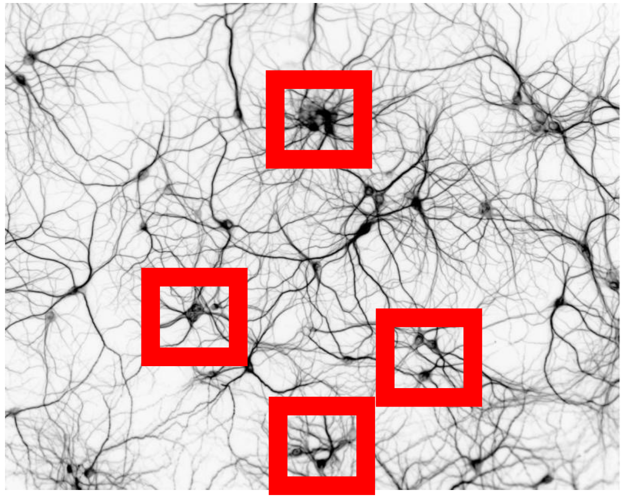
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

How can a machine "walk, talk, see, write, reproduce itself and be conscious of its existence" when it can't solve the XOR problem?

Marvin Minsky and Seymore Papert, Perceptrons, MIT Press, 1969

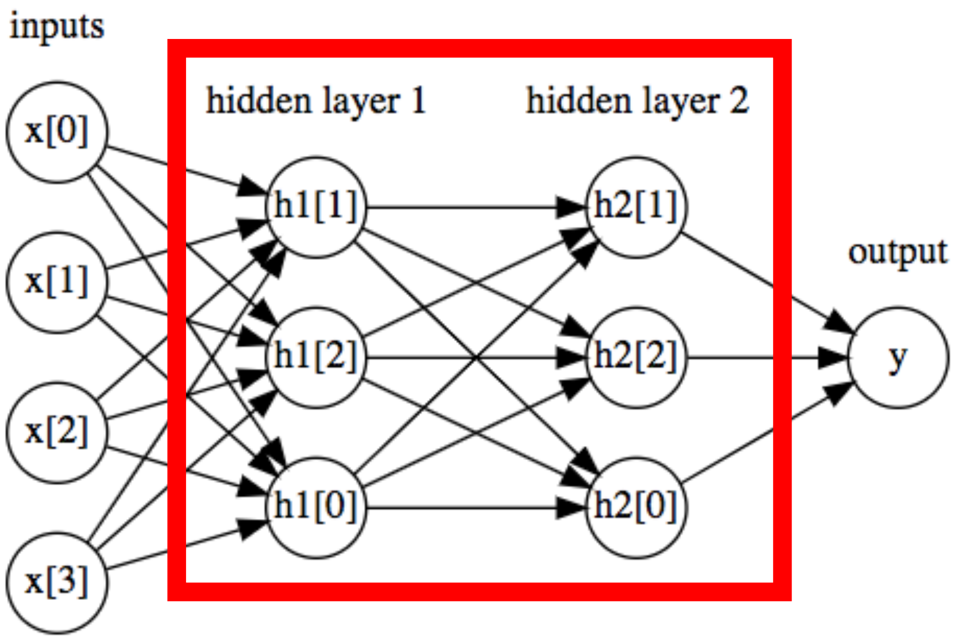
Idea: Use Connected Neurons (i.e., Neural Networks) to Transform Input into Features Useful for Prediction

Biological Neural Network:



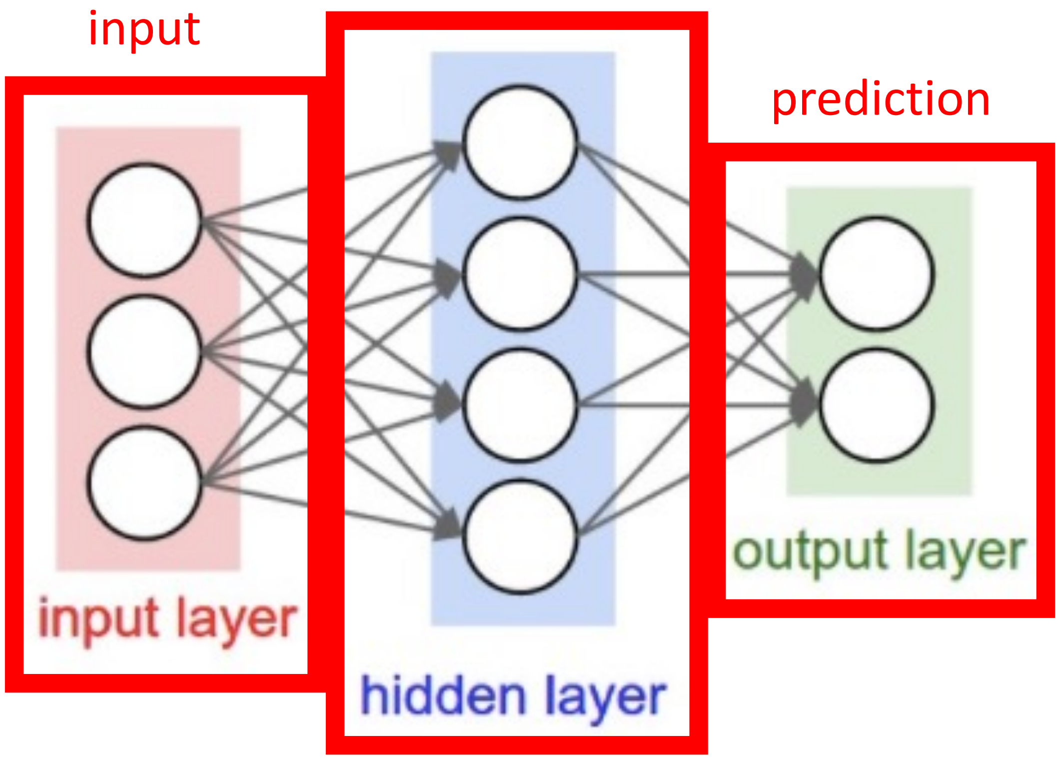
http://www.rzagabe.com/2014/11/03/an-introduction-to-artificial-neural-networks.html

Artificial Neural Network:



https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb

Neural Network

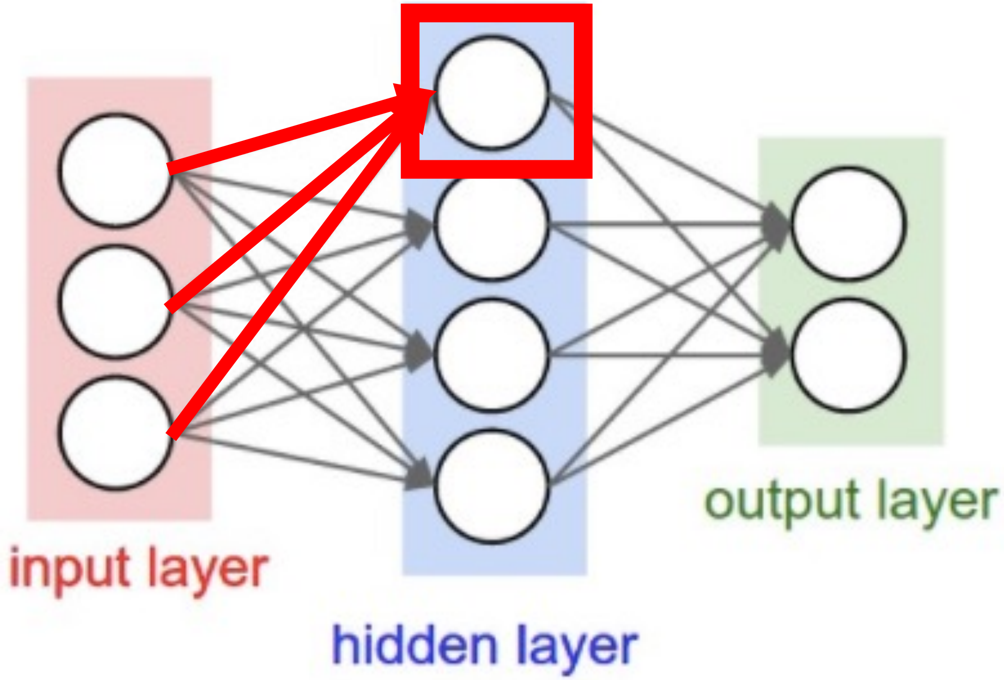


“hidden layer” uses outputs of units (i.e., neurons) and provides them as inputs to other units (i.e., neurons)

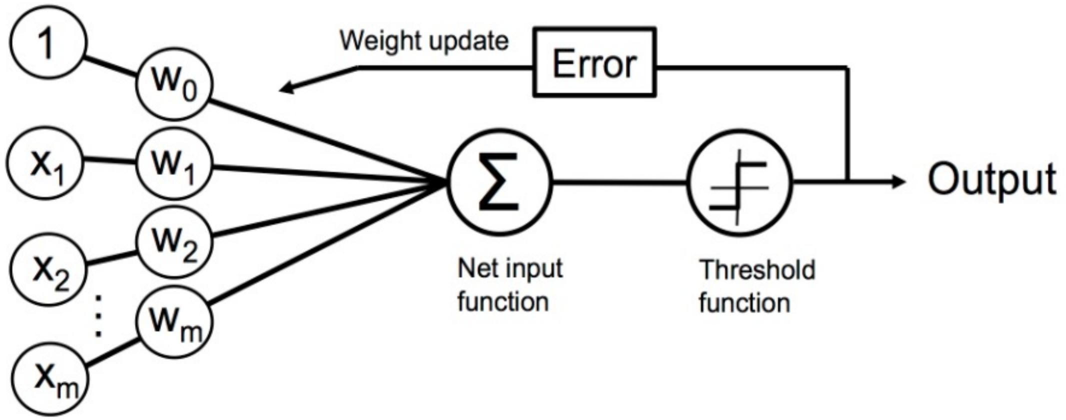
- Also called “multilayer perceptron”
- This is a 2-layer “feed-forward” neural network (i.e., count number of hidden layers plus output layer and exclude input layer)

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

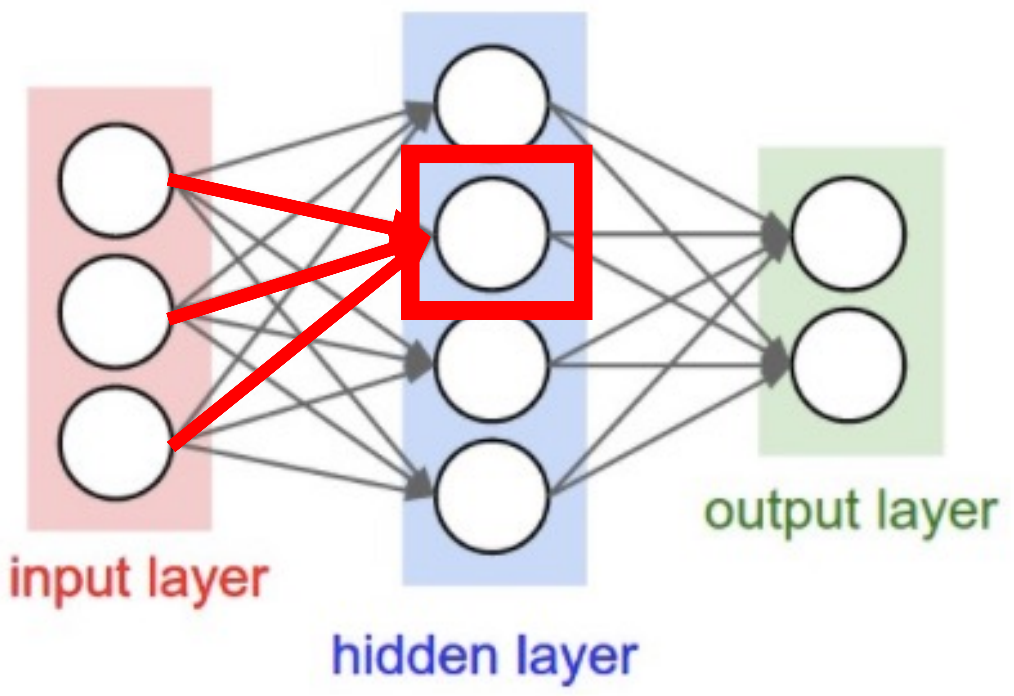


- Unit: takes as input a weighted sum and applies a function to the input

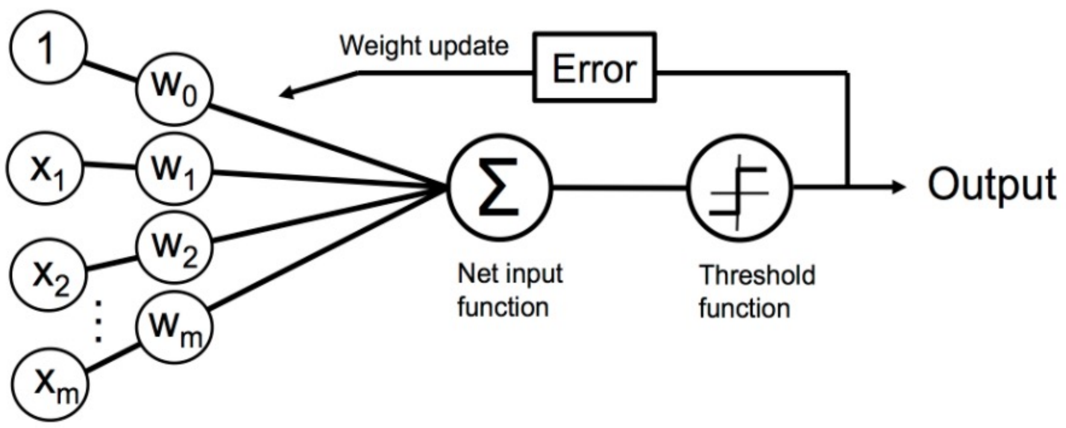
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

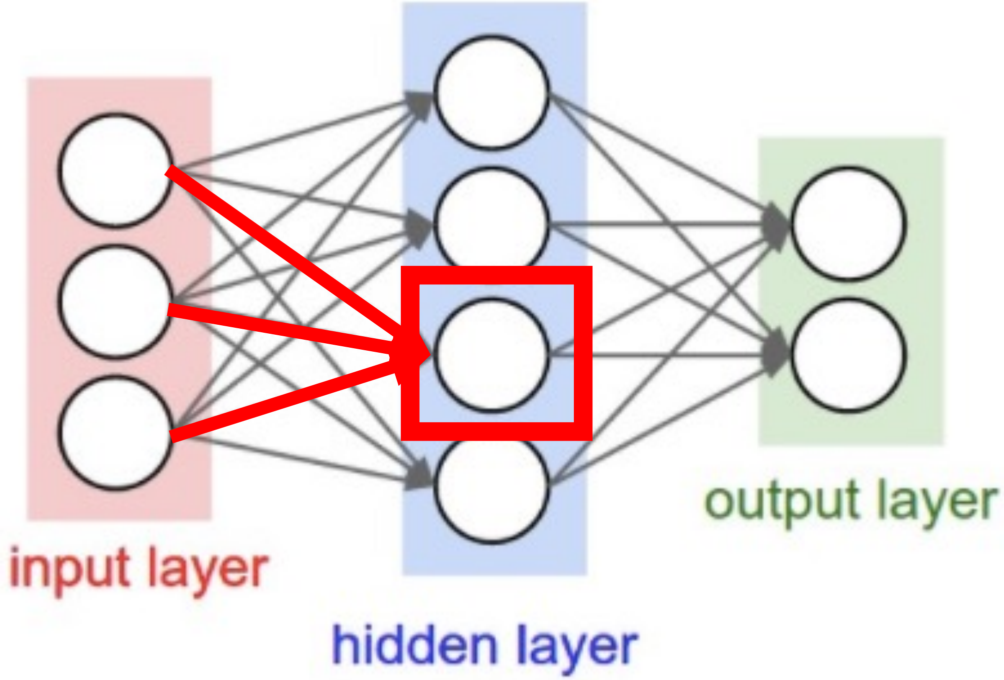


- Unit: takes as input a weighted sum and applies a function to the input

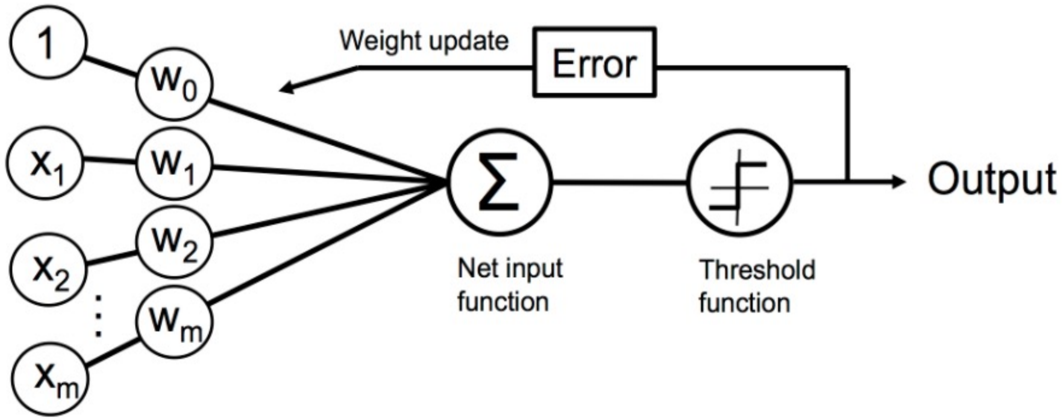
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

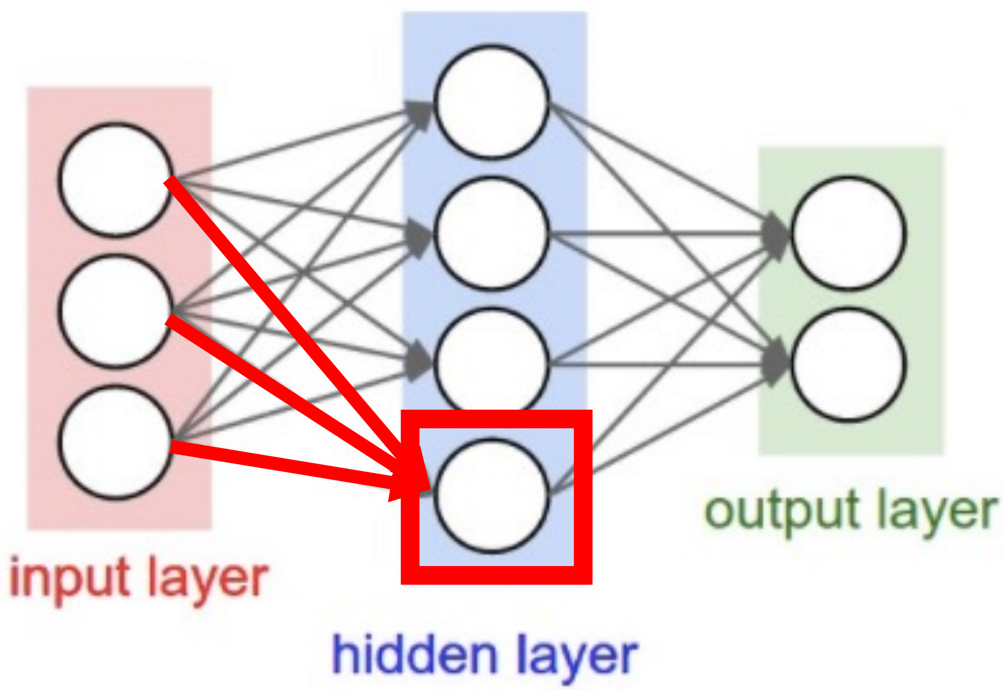


- Unit: takes as input a weighted sum and applies a function to the input

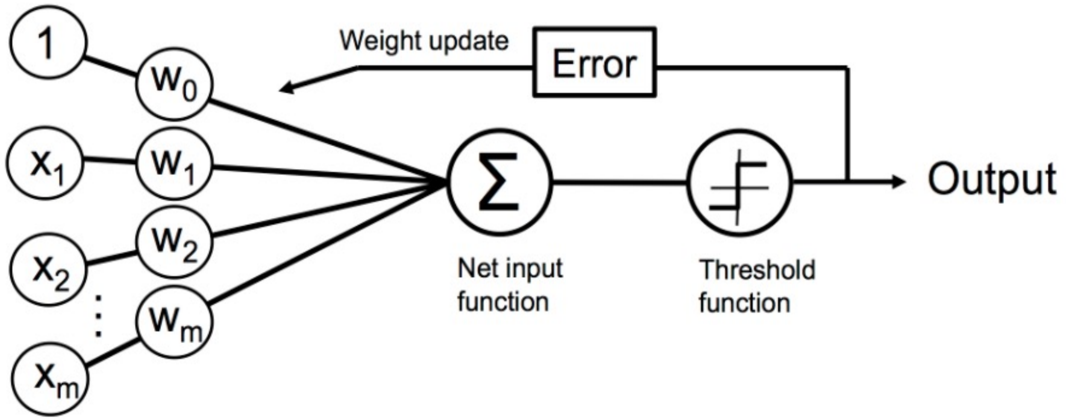
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

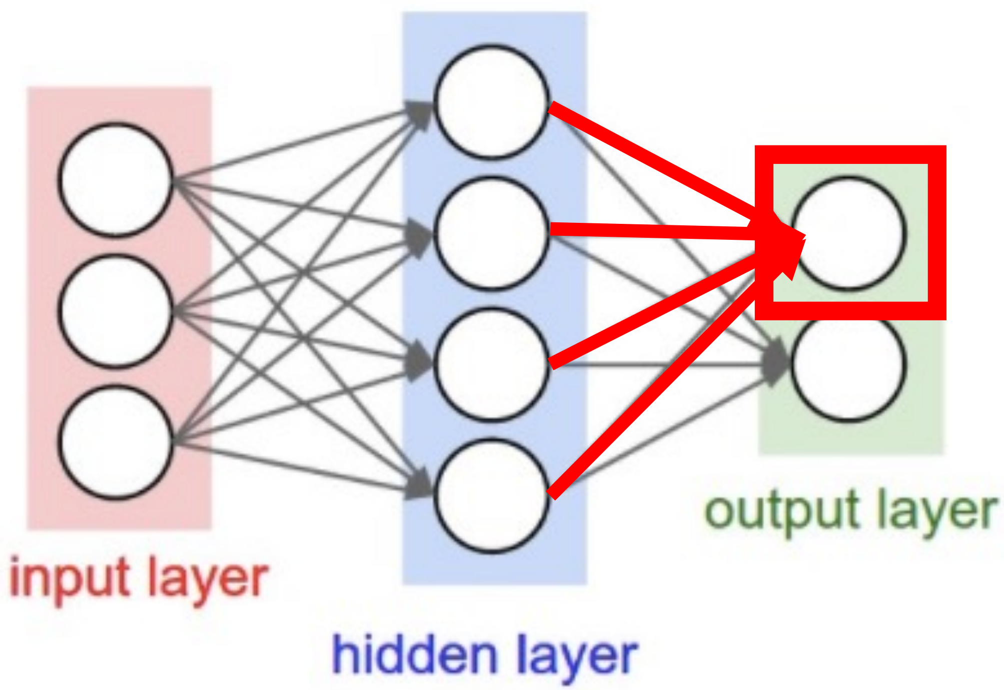


- Unit: takes as input a weighted sum and applies a function to the input

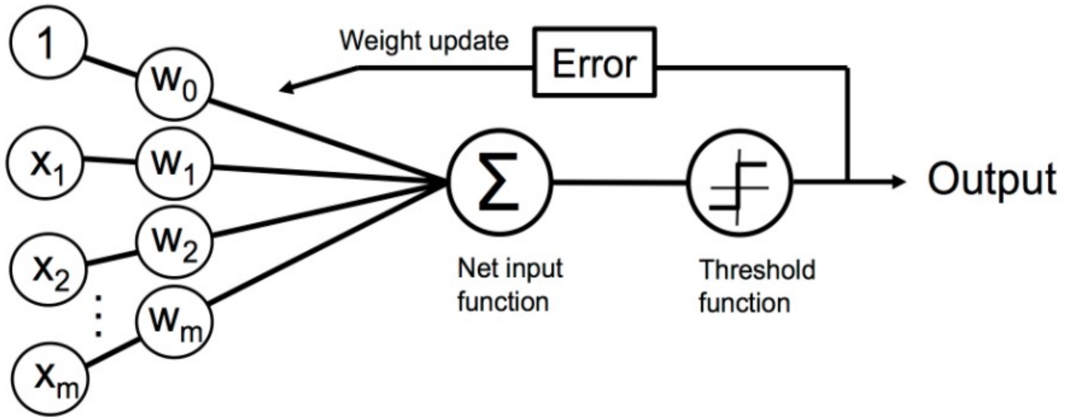
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

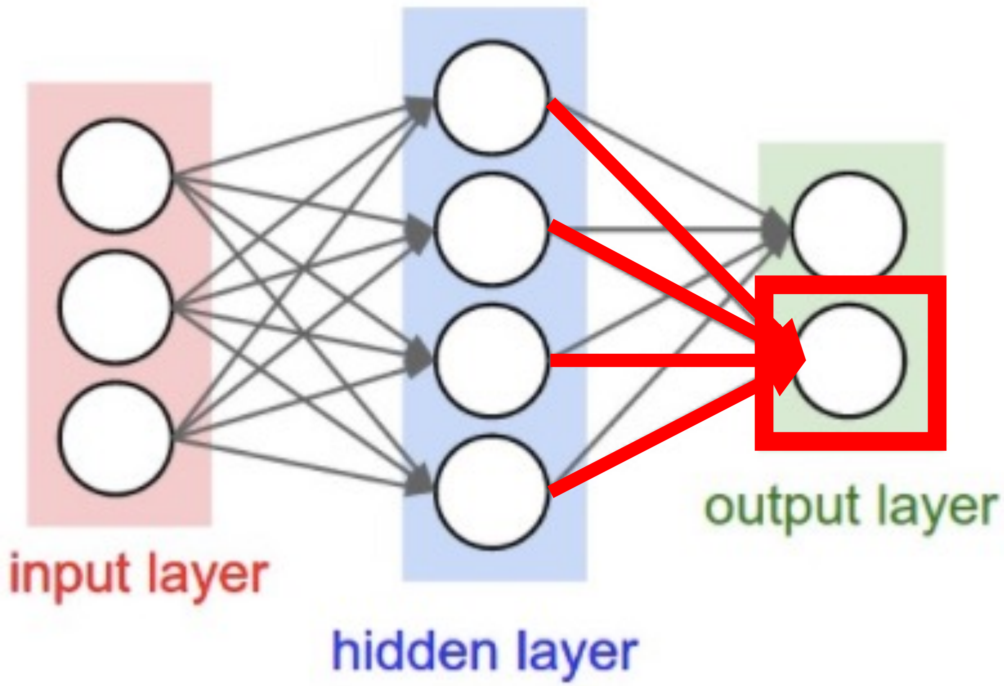


- Unit: takes as input a weighted sum and applies a function to the input

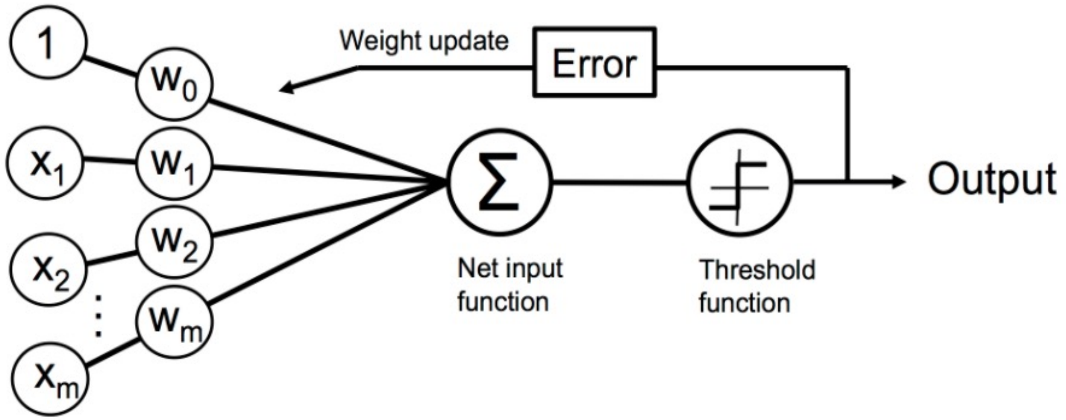
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

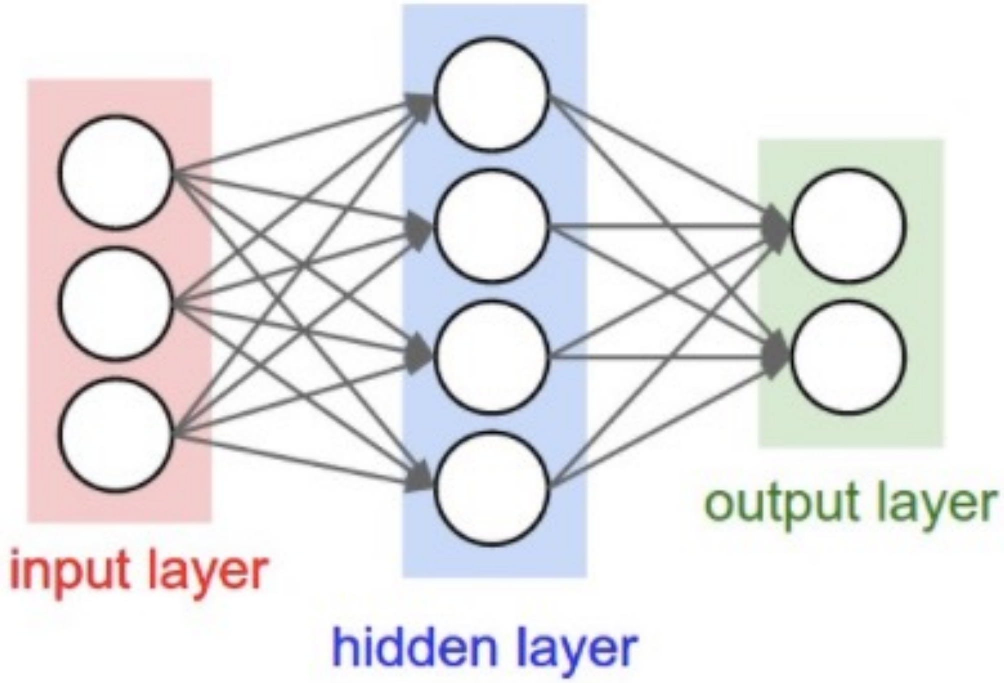


- Unit: takes as input a weighted sum and applies a function to the input

Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

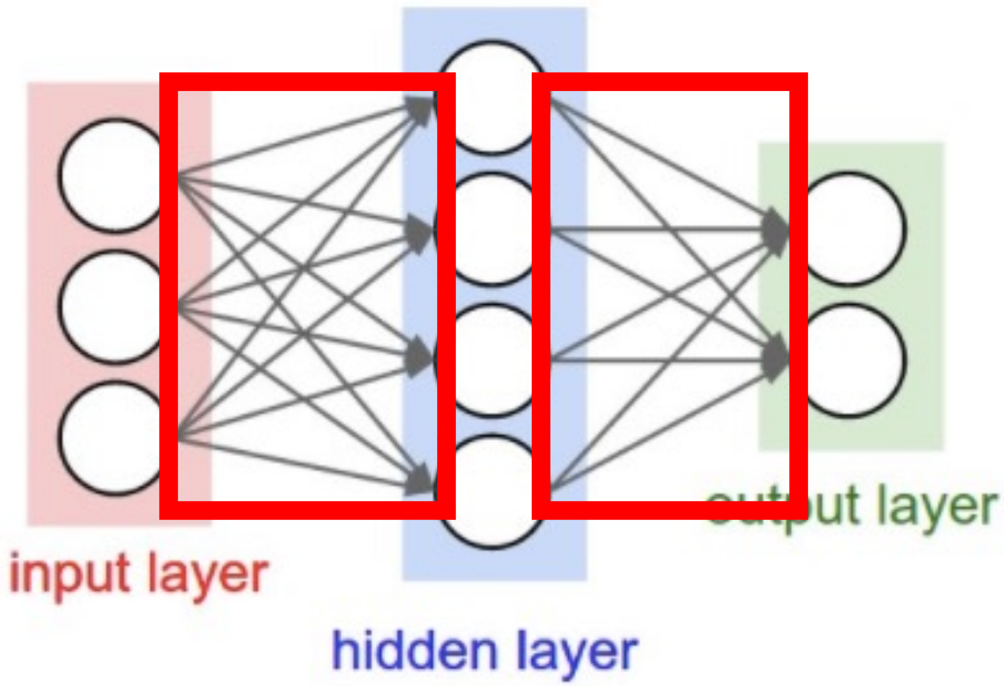
Neural Network



- **Training goal: learn model parameters**
- Layers are called "hidden" because algorithm decides how to use each layer to produce its output

http://cs231n.github.io/neural-networks-1/

Neural Network

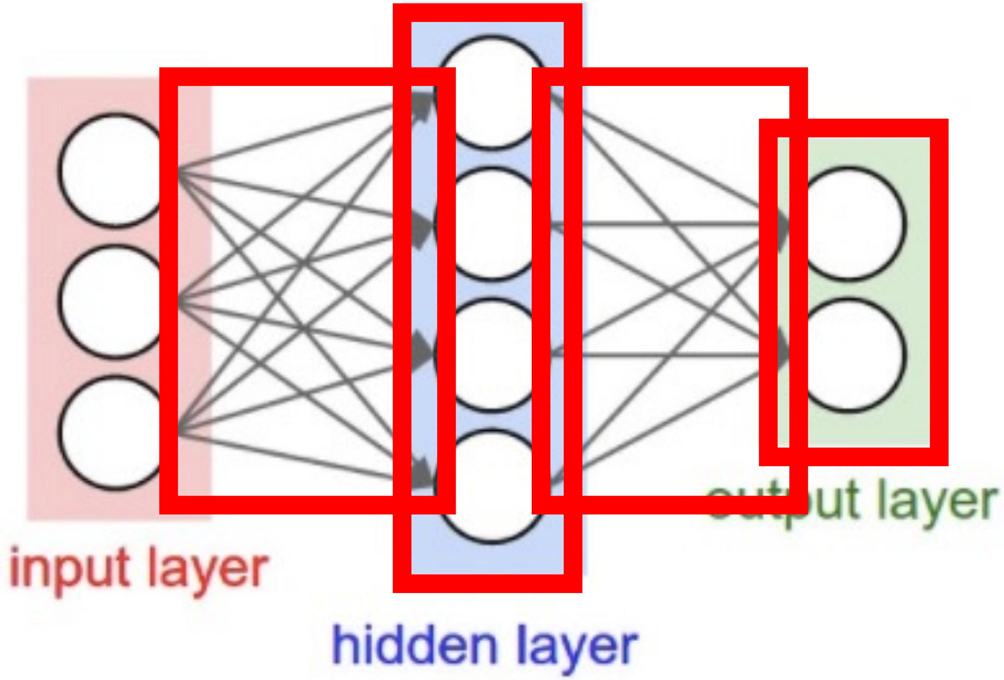


How many weights are in this model?

- Input to Hidden Layer:
  - 3x4 = 12
- Hidden Layer to Output Layer
  - 4x2 = 8
- Total:
  - 12 + 8 = 20

http://cs231n.github.io/neural-networks-1/

Neural Network

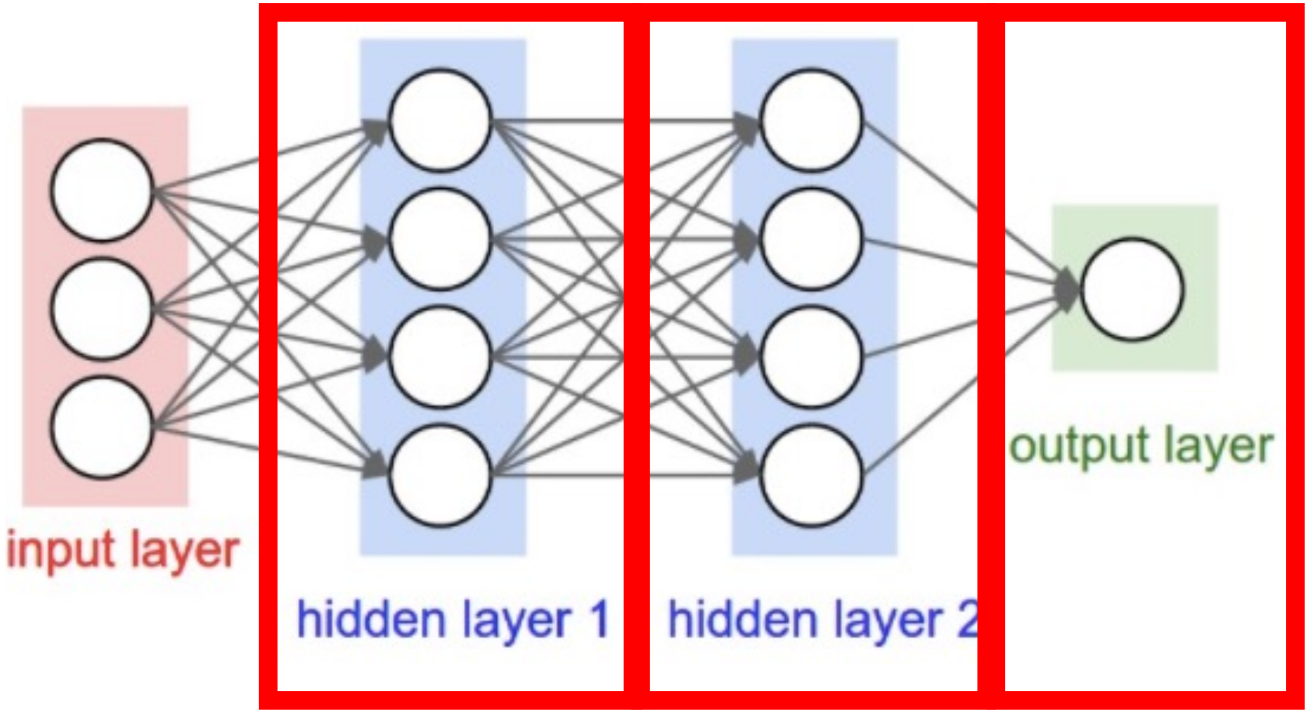


How many parameters are there to learn?

- Number of weights:
  - 20
- Number of biases:
  - 4 + 2 = 6
- Total:
  - 26

http://cs231n.github.io/neural-networks-1/

Neural Network

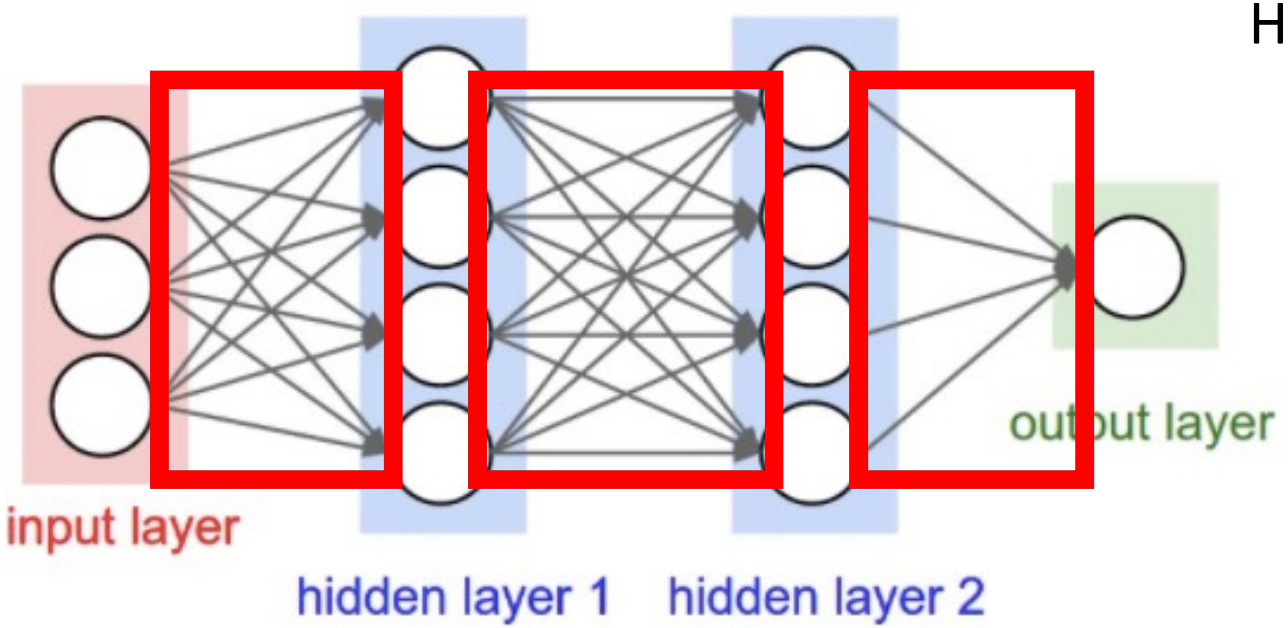


How many layers are in this network?

- 3 (number of hidden layers plus output layer; input layer excluded when counting)

http://cs231n.github.io/neural-networks-1/

Neural Network

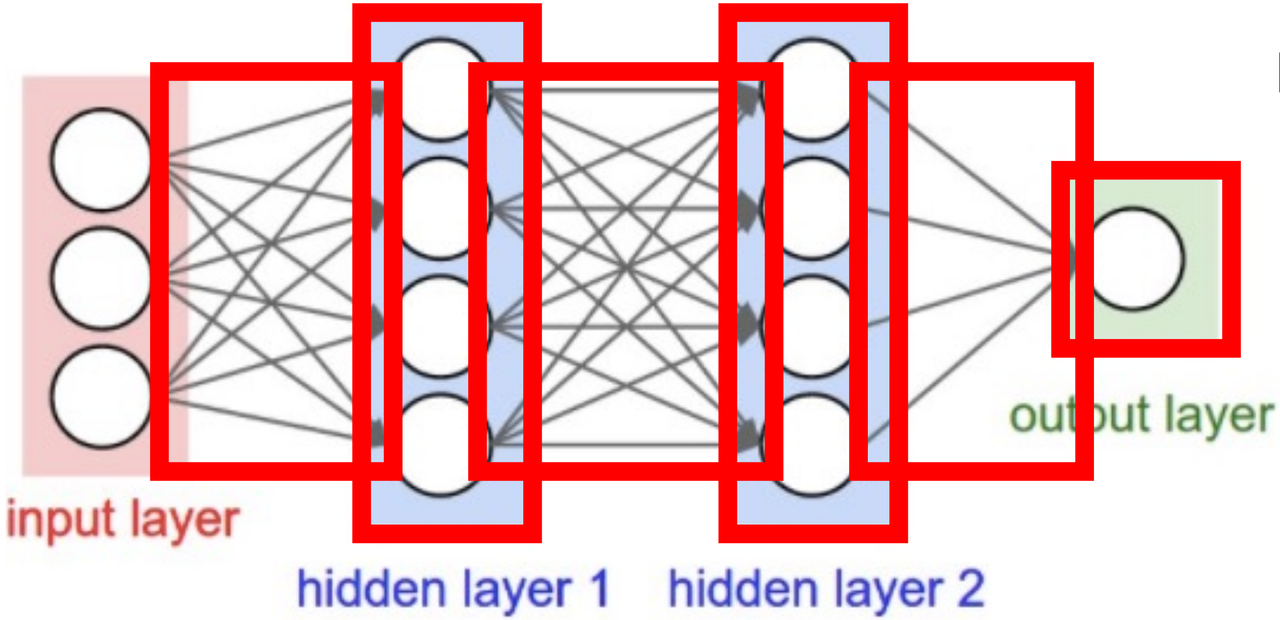


How many weights are in this model?

- Input to Hidden Layer 1:
  - 3x4 = 12
- Hidden Layer 1 to Hidden Layer 2:
  - 4x4 = 16
- Hidden Layer 2 to Output Layer
  - 4x1 = 4
- Total:
  - 12 + 16 + 4 = 32

http://cs231n.github.io/neural-networks-1/

Neural Network



How many parameters are there to learn?

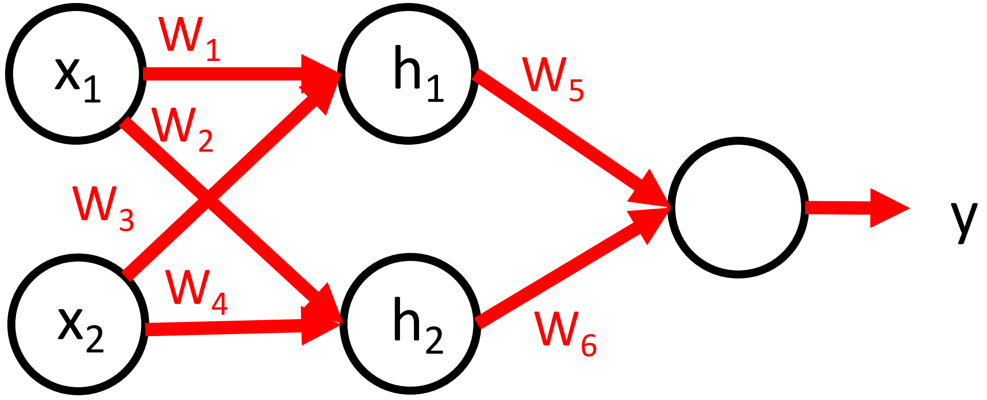
- Number of weights:
  - 32
- Number of biases:
  - 4 + 4 + 1 = 9
- Total
  - 41

http://cs231n.github.io/neural-networks-1/

Hidden Layers Alone Are NOT Enough to Model Non-Linear Functions

Key Observation: feedforward networks are just functions chained together

e.g.,



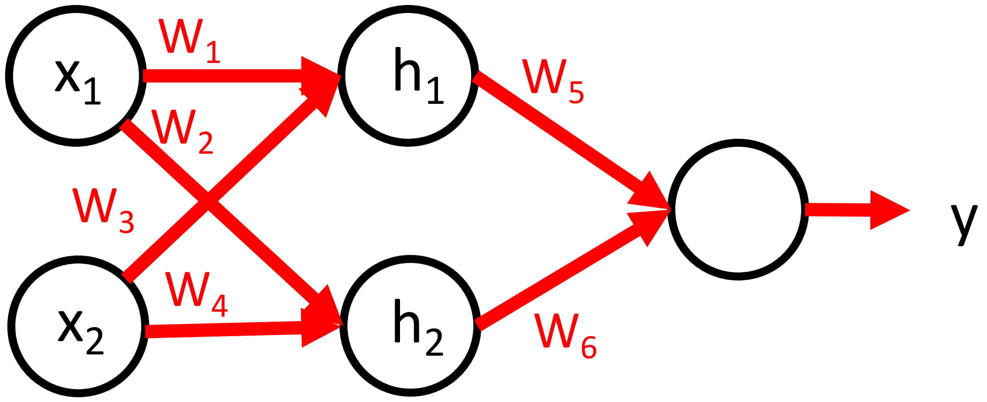
- What is function for h_1?
  - $h_1 = w_1x_1 + w_3x_2 + b_1$
- What is function for h_2?
  - $h_2 = w_2x_1 + w_4x_2 + b_2$
- What is function for y?
  - $y = h_1w_5 + h_2w_6 + b_3$
  - $y = (w_1x_1 + w_3x_2 + b_1)w_5 + (w_2x_1 + w_4x_2 + b_2)w_6 + b_3$
  - $\boxed{y = w_1w_5x_1 + w_3w_5x_2 + w_5b_1 + w_2w_6x_1 + w_4w_6x_2 + w_6b_2 + b_3}$

A chain of LINEAR functions at any depth is still a LINEAR function!

Hidden Layers Alone Are NOT Enough to Model Non-Linear Functions

Key Observation: feedforward networks are just functions chained together

e.g.,



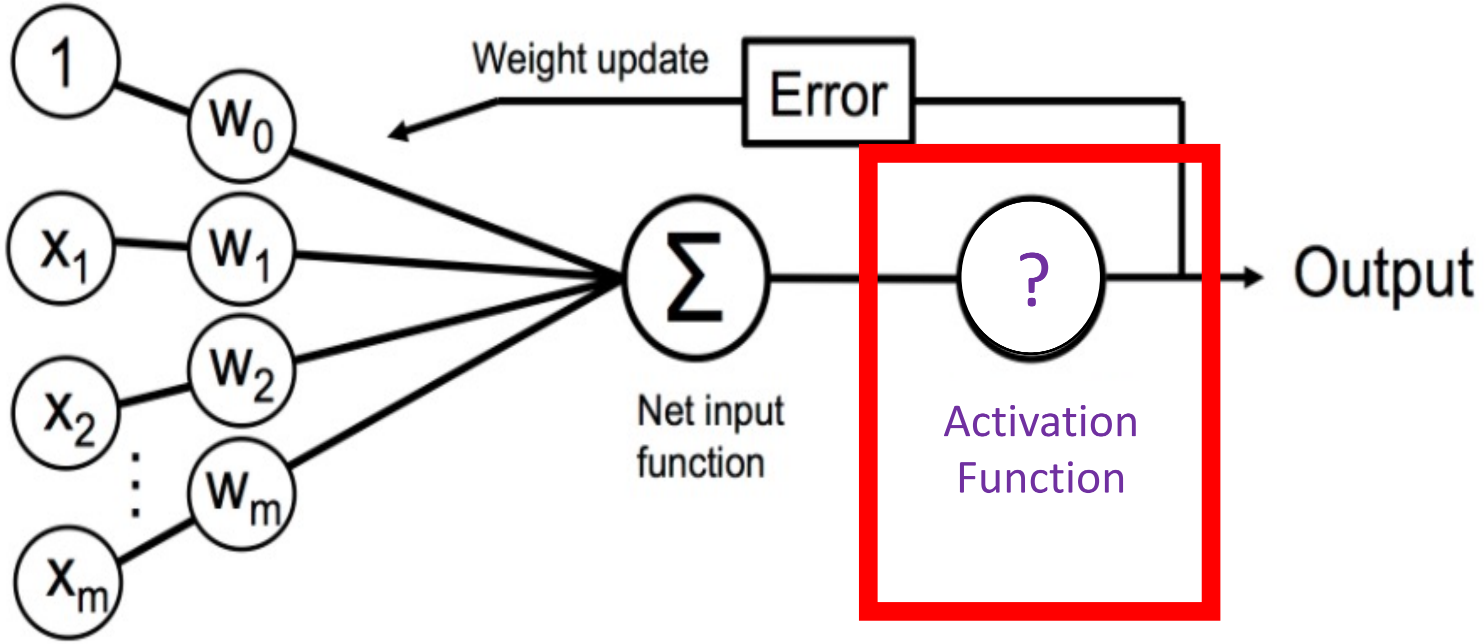
- What is function for h_1?
  - $h_1 = w_1x_1 + w_3x_2 + b_1$
- What is function for h_2?
  - $h_2 = w_2x_1 + w_4x_2 + b_2$
- What is function for y?
  - $y = h_1w_5 + h_2w_6 + b_3$

Constant x linear function = linear function

A chain of LINEAR functions at any depth is still a LINEAR function!

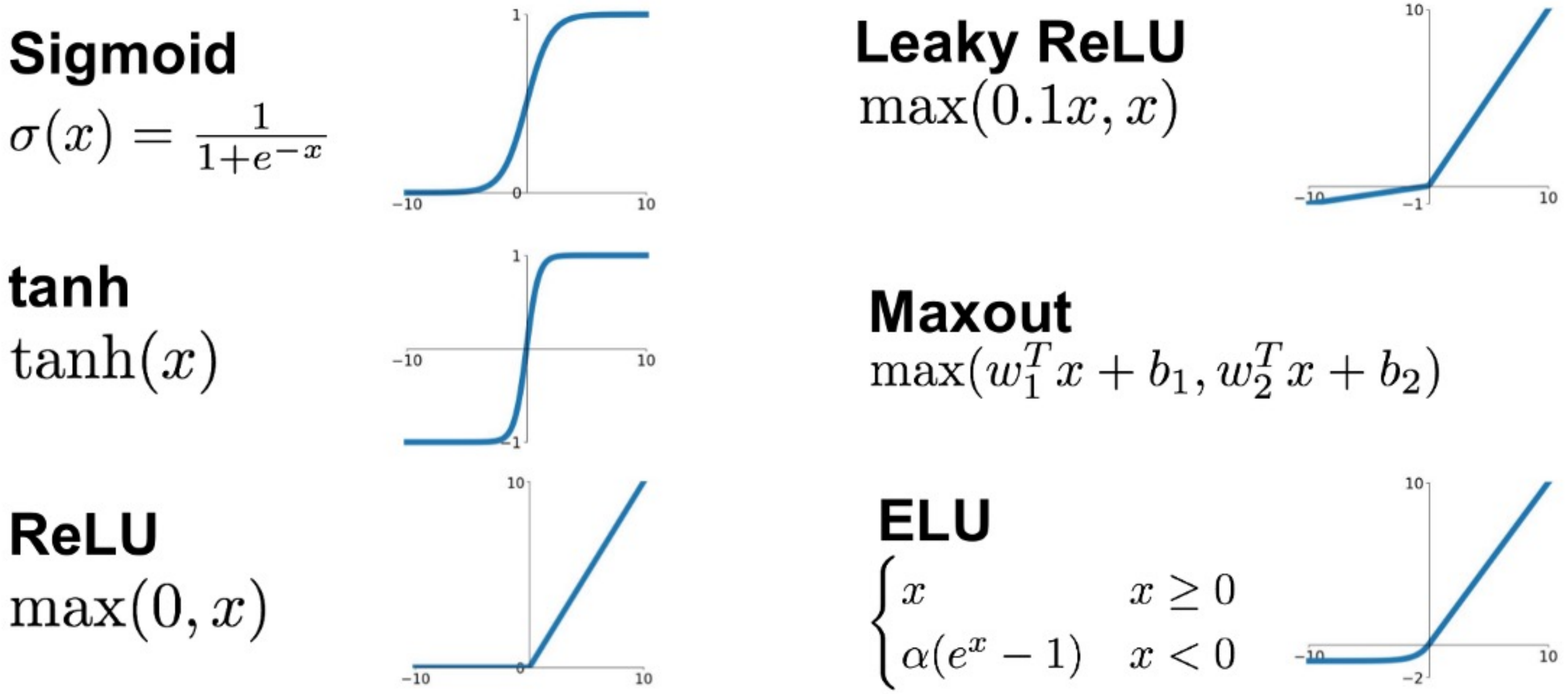
Need to Use Non-Linear Activation Functions

- Each unit applies a non-linear “activation” function to the weighted input to mimic a neuron firing



Python Machine Learning; Raschka & Mirjalili

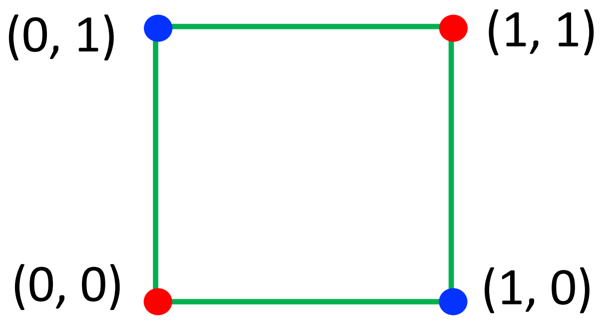
Need to Use Non-Linear Activation Functions

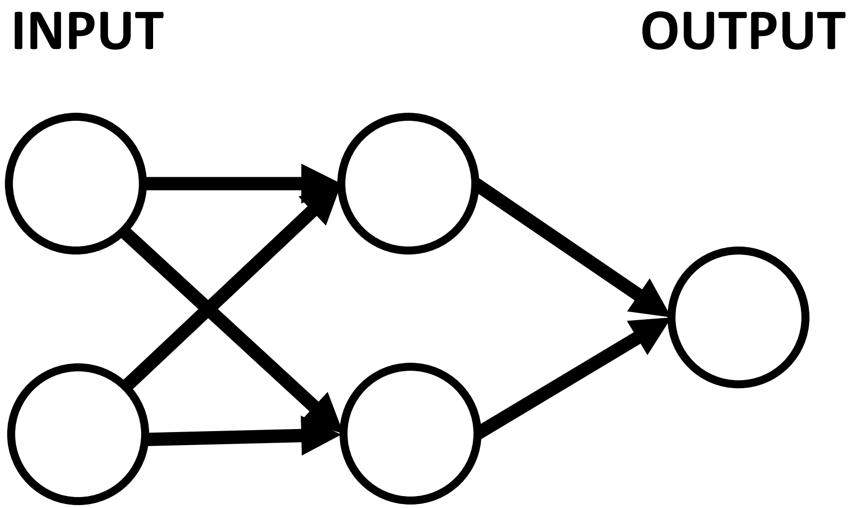


Source: https://www.linkedin.com/pulse/activation-functions-neural-networks-leonardo-calderon-j-/

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:

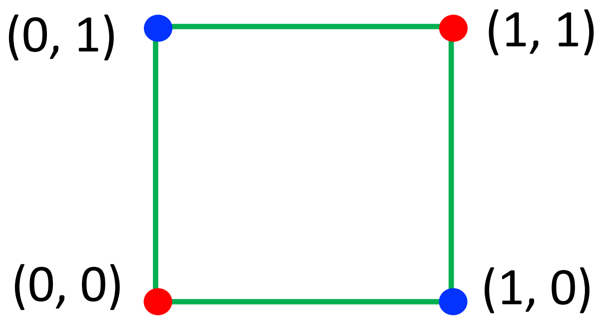




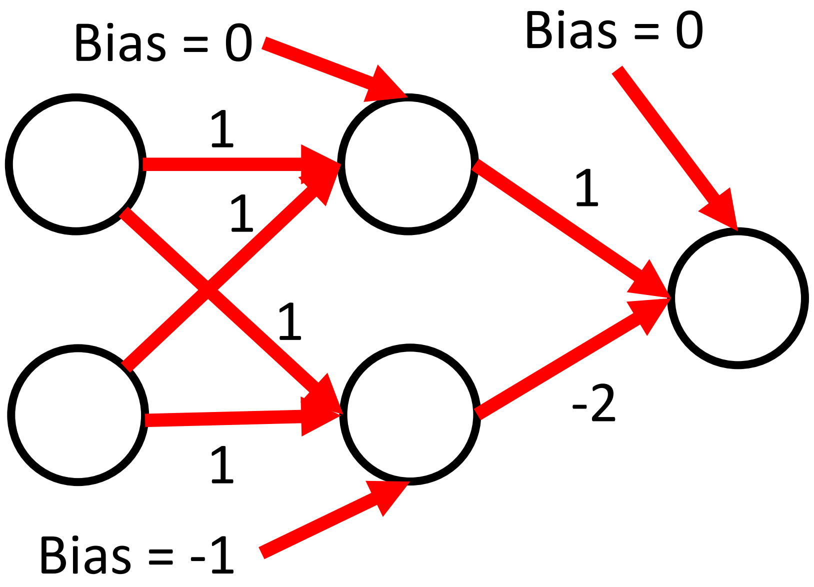
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



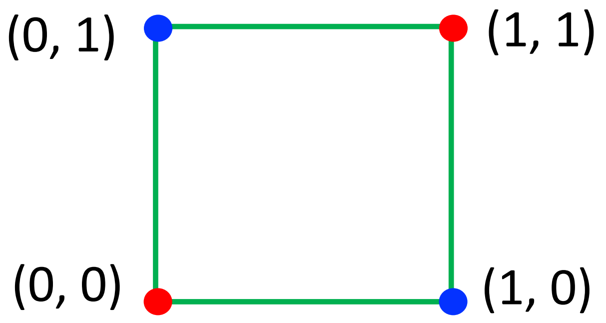
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



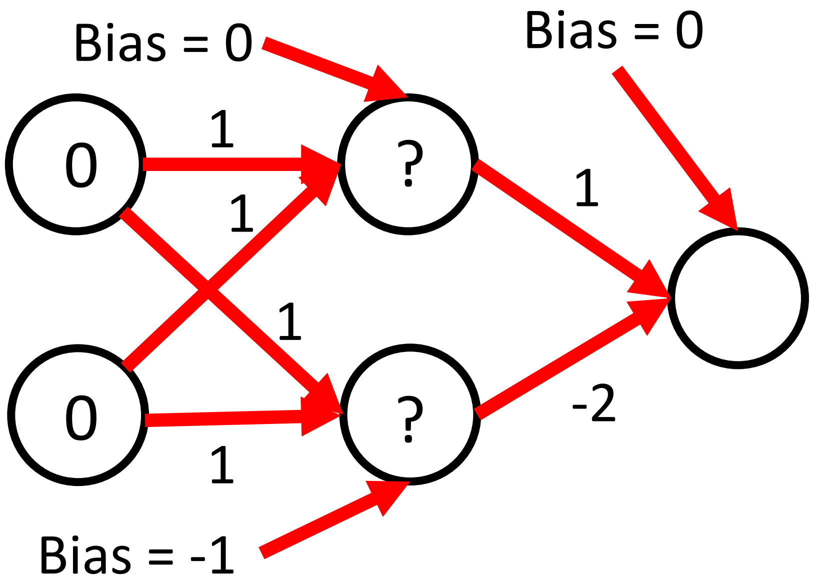
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



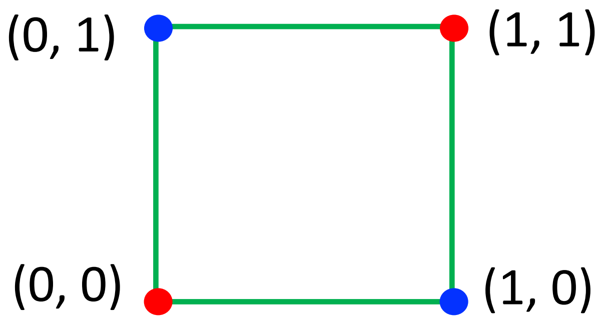
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



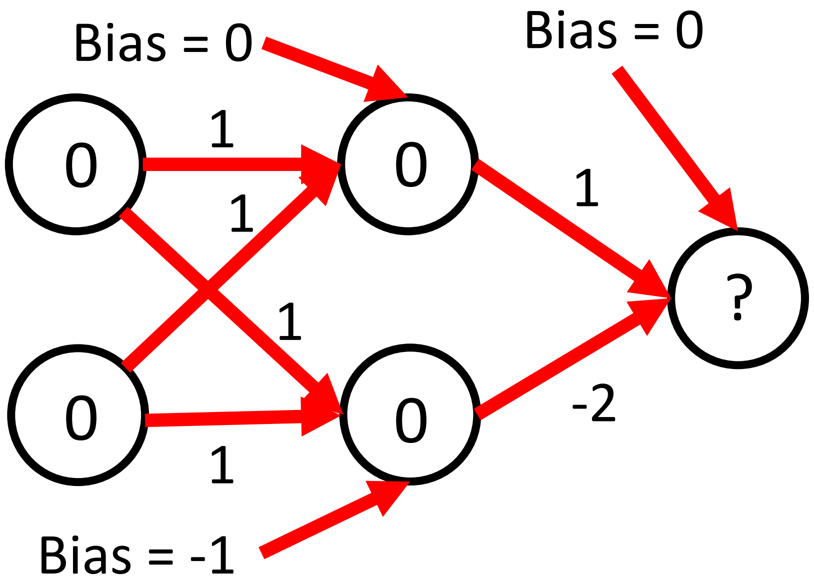
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



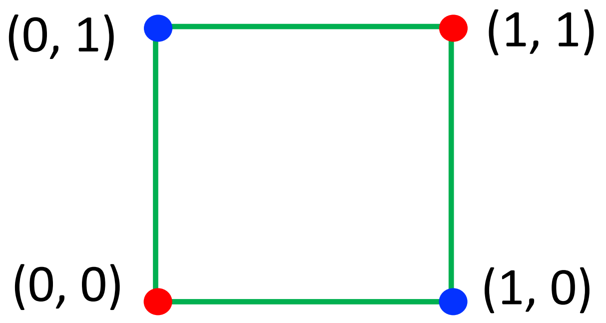
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



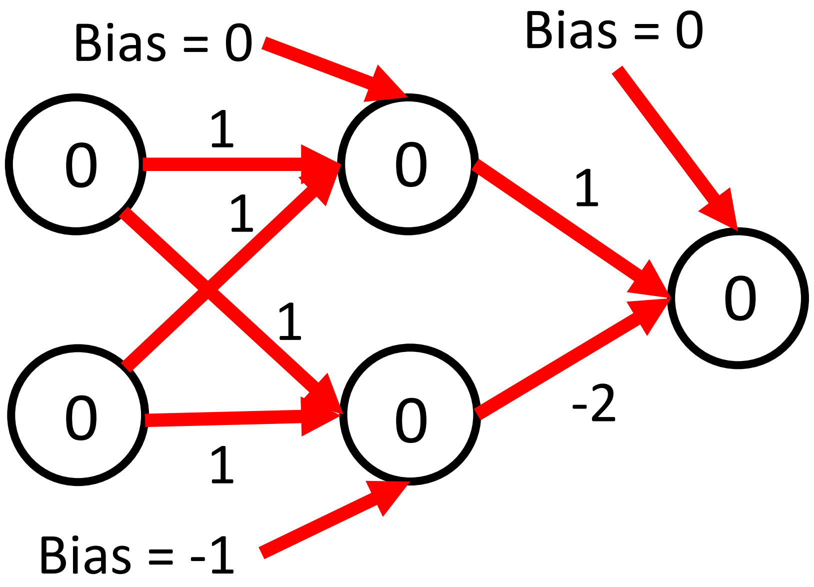
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



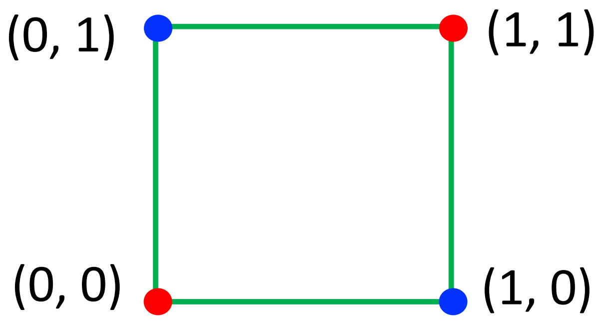
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



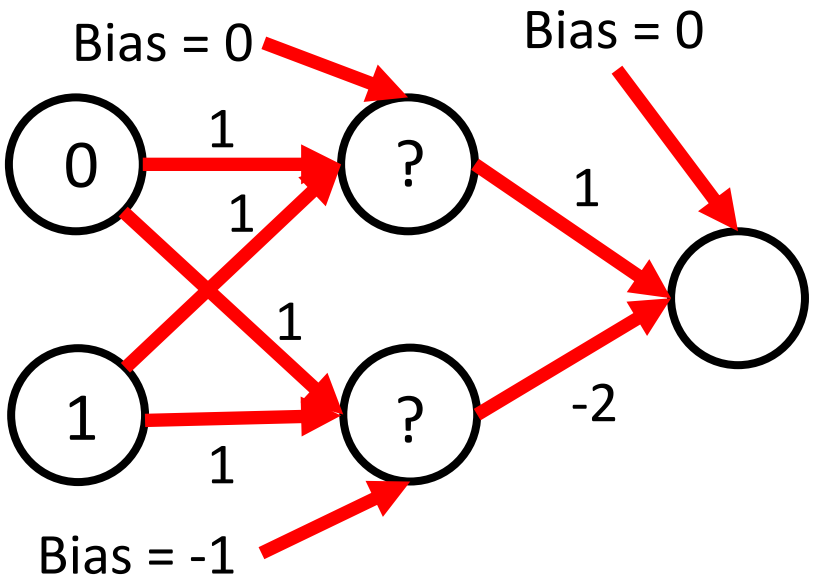
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



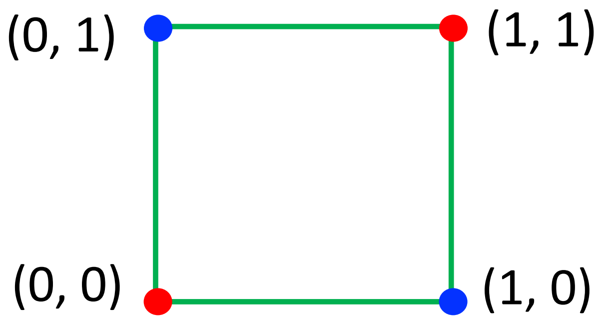
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



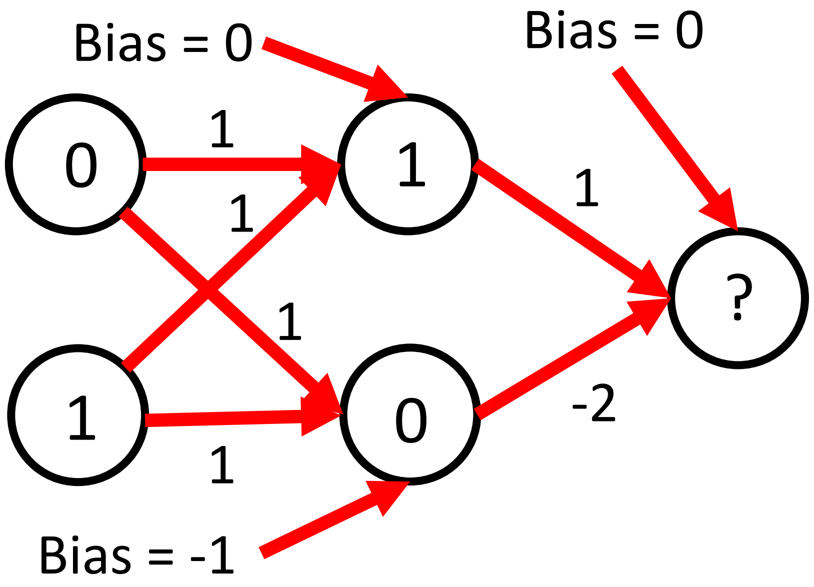
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



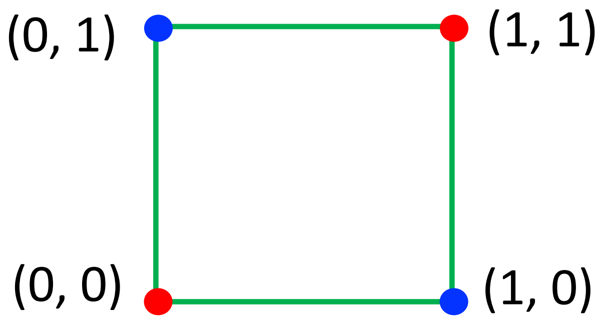
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



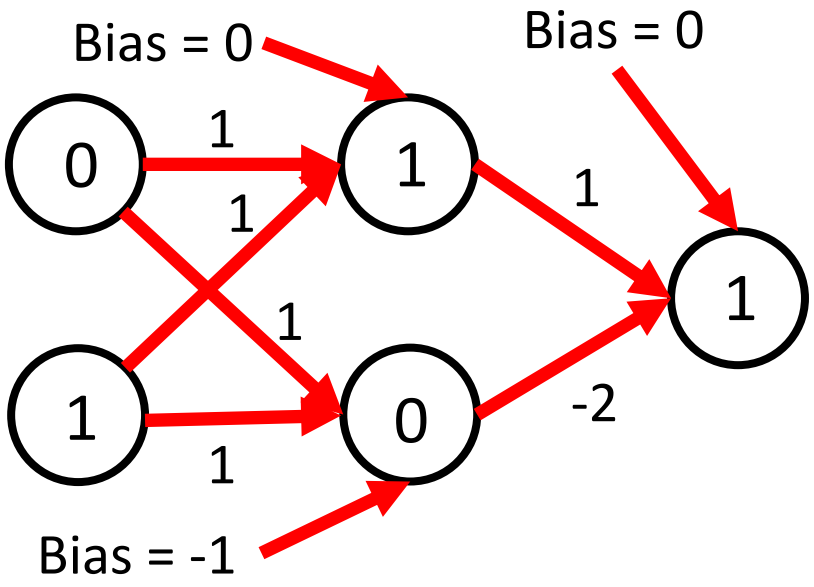
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



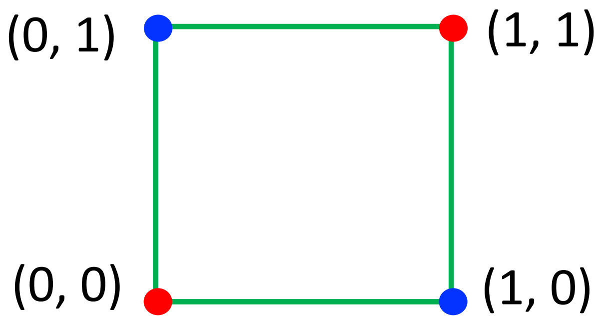
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



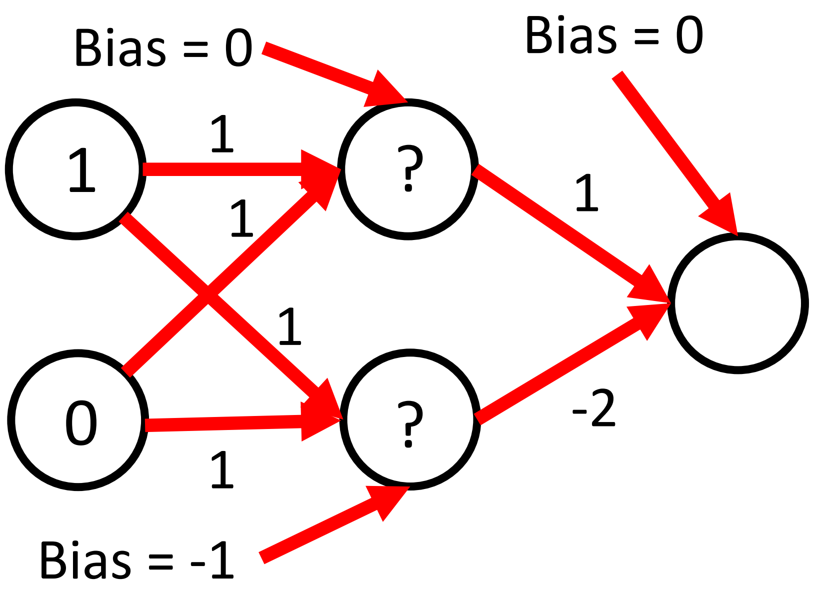
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



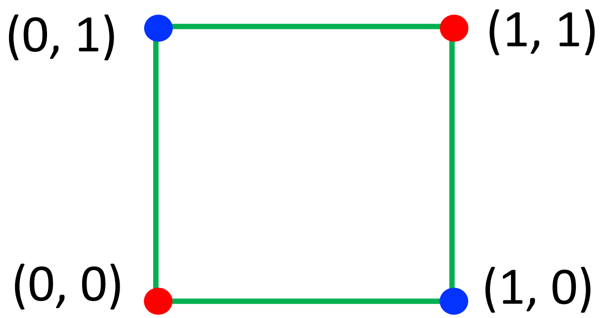
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



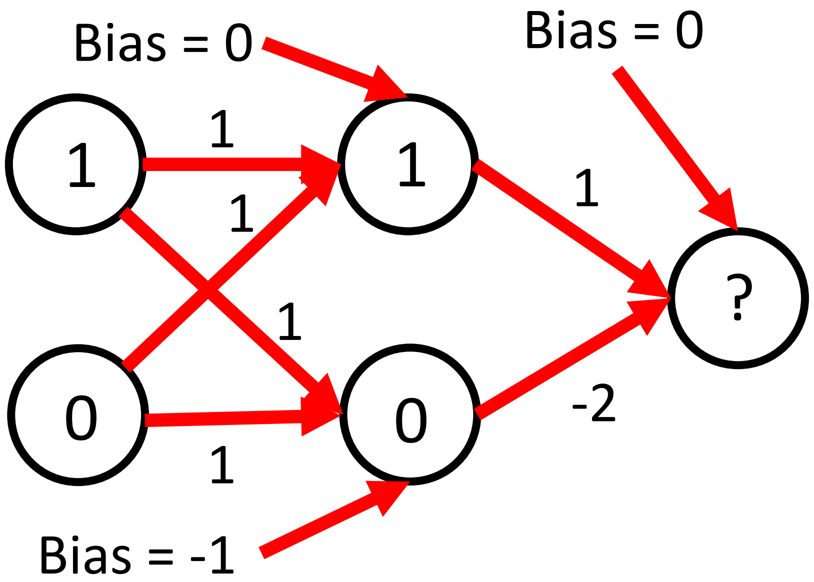
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



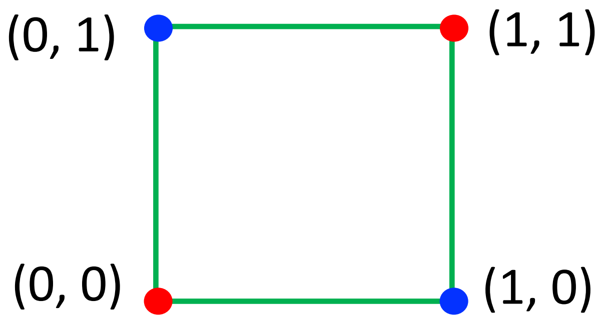
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



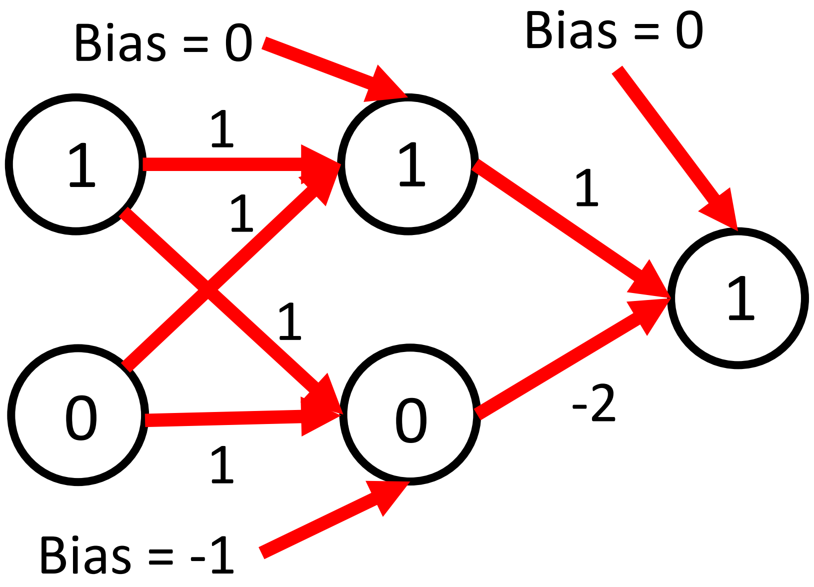
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



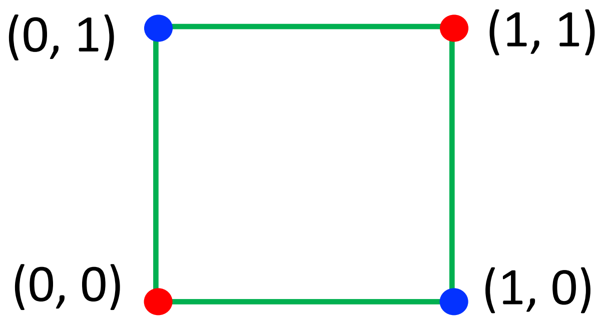
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



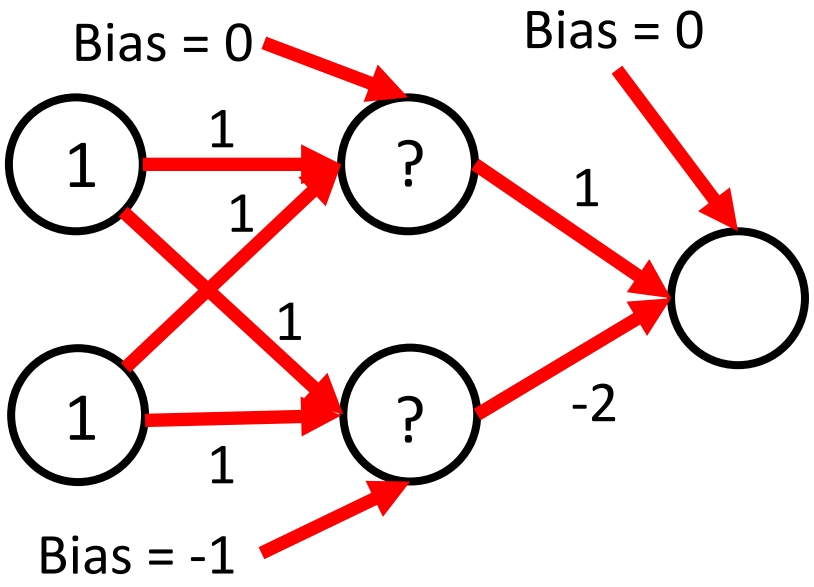
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



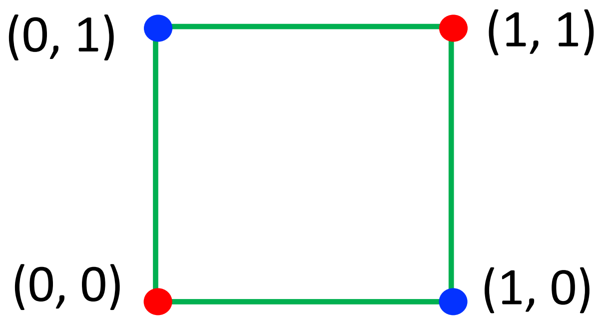
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



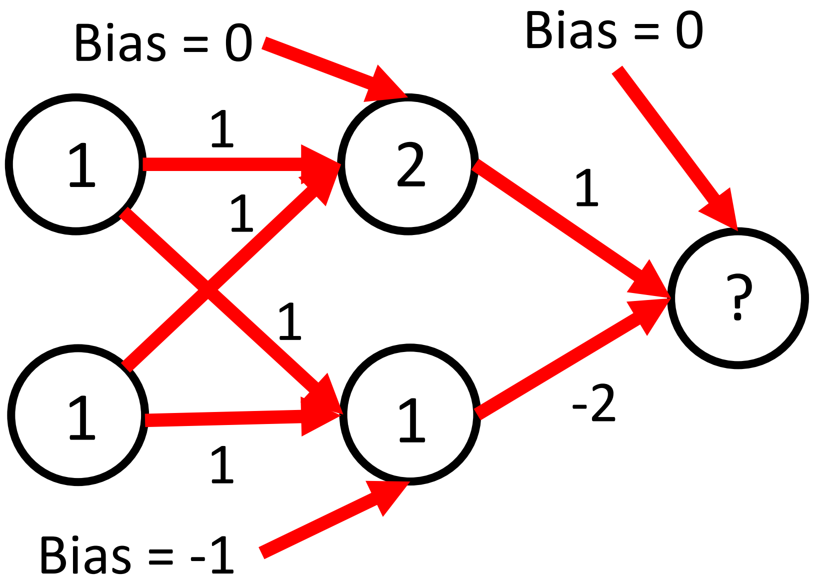
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



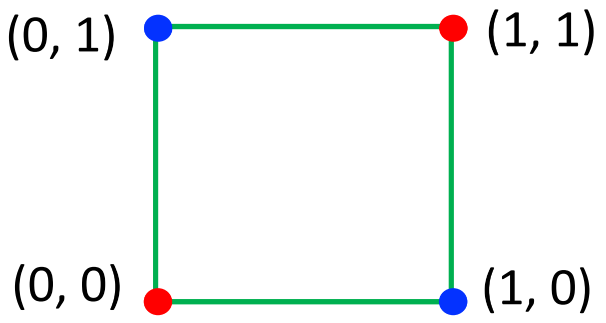
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



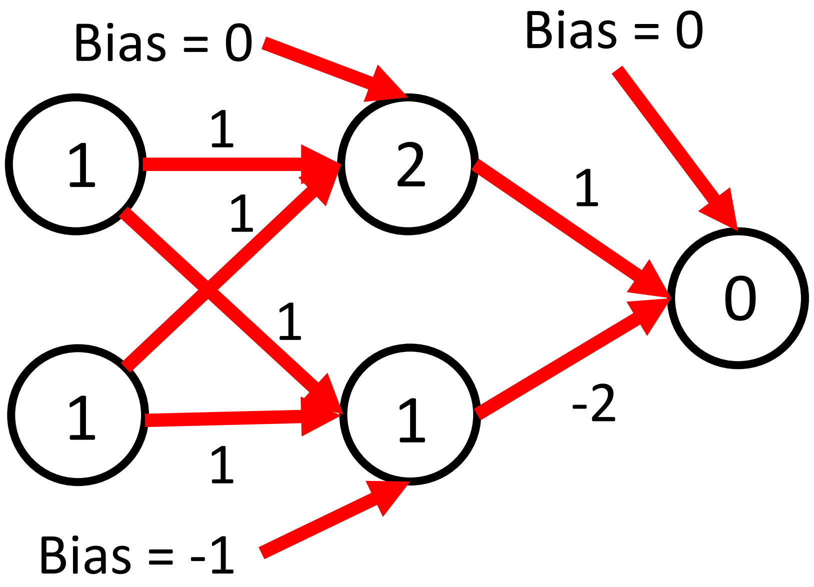
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
$\boxed{1}$	$\boxed{1}$	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



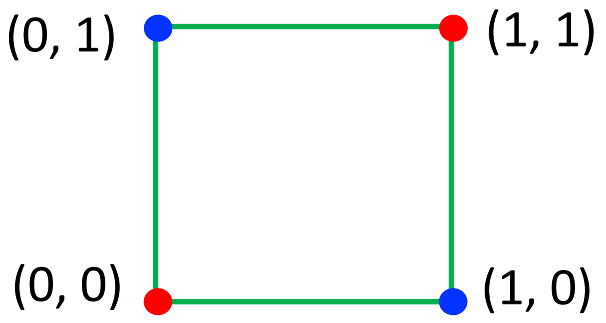
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:



- Approach: Use ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with this model:

Neural networks can solve XOR problem...
and so model non-linear functions!

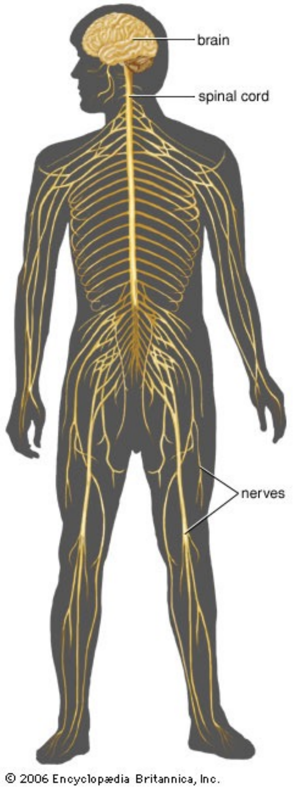
Today’s Topics

- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Fundamentals of a neural network architecture
- Training deep neural networks

Modern Neural Networks Are Huge, Matching or Exceeding Number of Neurons in a Human

https://en.wikipedia.org/wiki/Nematode#/media/File:CeleгansGoldsteinLabUNC.jpg

Nematode worm: 302 neurons

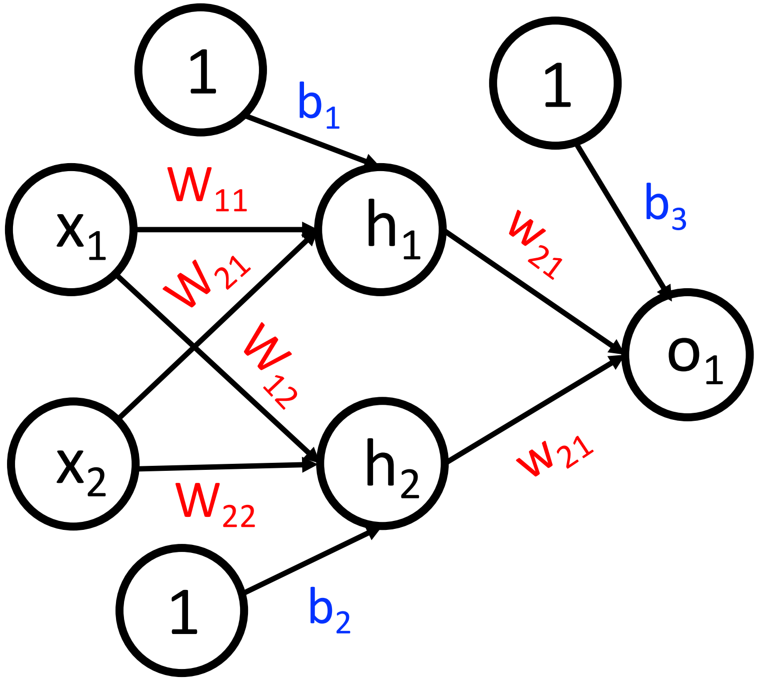


https://www.britannica.com/science/human-nervous-system

Human: ~100,000,000,000 neurons

Training for Neural Networks

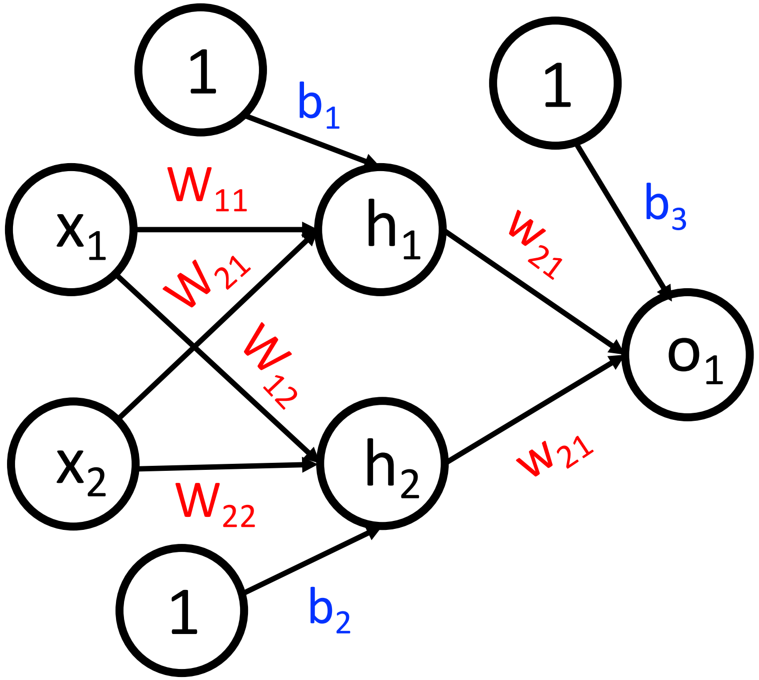
- Learn model parameters that minimize an objective function using gradient descent; e.g., weights and biases:



- Gradient descent was introduced in an 1847 publication and is a scalable way to train nonlinear models on "big data"

http://cs231n.github.io/neural-networks-1/

Objective Functions: A Specified (Measurable) Goal for Trained Models



e.g., make as small as possible the squared error between predictions and ground truth (aka, L2 loss, quadratic loss)

Mean taken over *n* instances

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

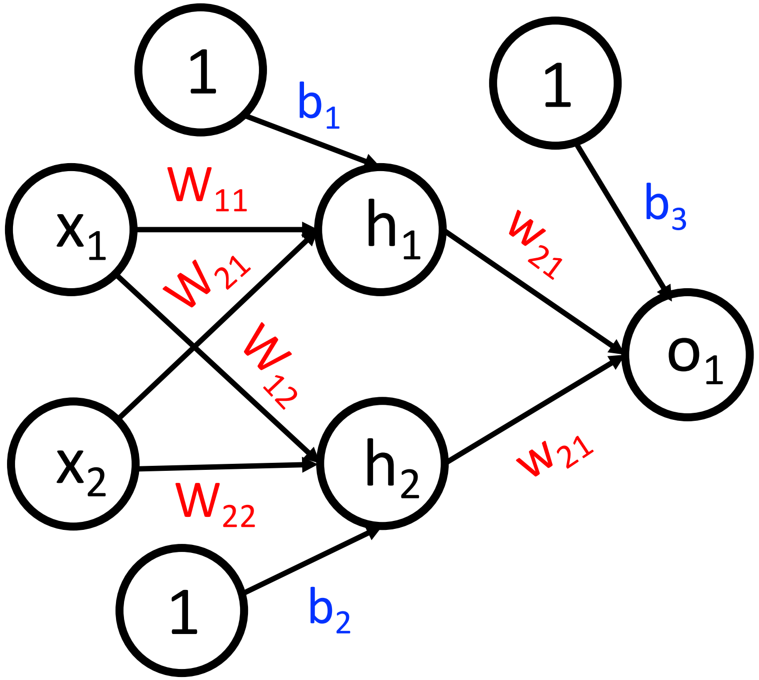
True value

Predicted value

What is the range of possible values?

- Minimum: 0
  - i.e., all correct predictions
- Maximum: Infinity
  - i.e., incorrect predictions

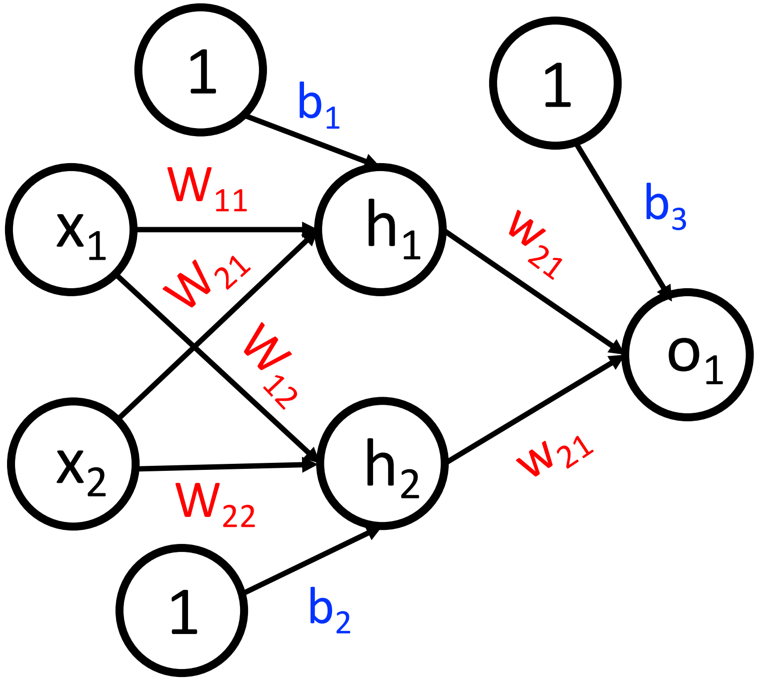
Objective Functions: A Specified (Measurable) Goal for Trained Models



MANY objective functions exist!

Training for Neural Networks

- Learn model parameters that minimize an objective function using $\boxed{\text{gradient descent}}$
  e.g., weights and biases:

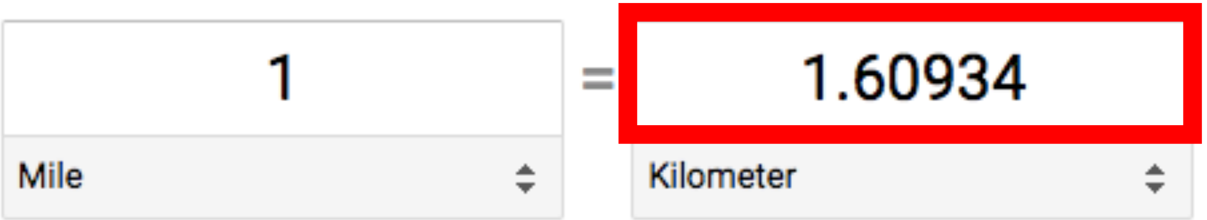


- Gradient descent was introduced in an 1847 publication and is a scalable way to train nonlinear models on "big data"

http://cs231n.github.io/neural-networks-1/

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn linear model for converting kilometers to miles when only observing the input “miles” and output “kilometers”



Miles → Kilometers = miles x constant → Kilometers

Gradient Descent: Intuition

- Repeat:
  1. $\boxed{\text{Guess}}$
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$\$10 \longrightarrow \boxed{\text{Shekels = dollars x constant}}$

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

\$10 → Shekels = dollars x constant → Error = Guess - Correct

Gradient Descent: Intuition

- Repeat:
  1. $\boxed{\text{Guess}}$
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$$\$10 \longrightarrow \boxed{\text{Shekels = dollars x constant}}$$

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant → Error = Guess - Correct

Gradient Descent: Intuition

- Repeat:
  1. $\boxed{\text{Guess}}$
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$\$10 \longrightarrow \boxed{\text{Shekels = dollars x constant}}$

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. $\boxed{\text{Calculate error}}$
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant → Error = Guess - Correct

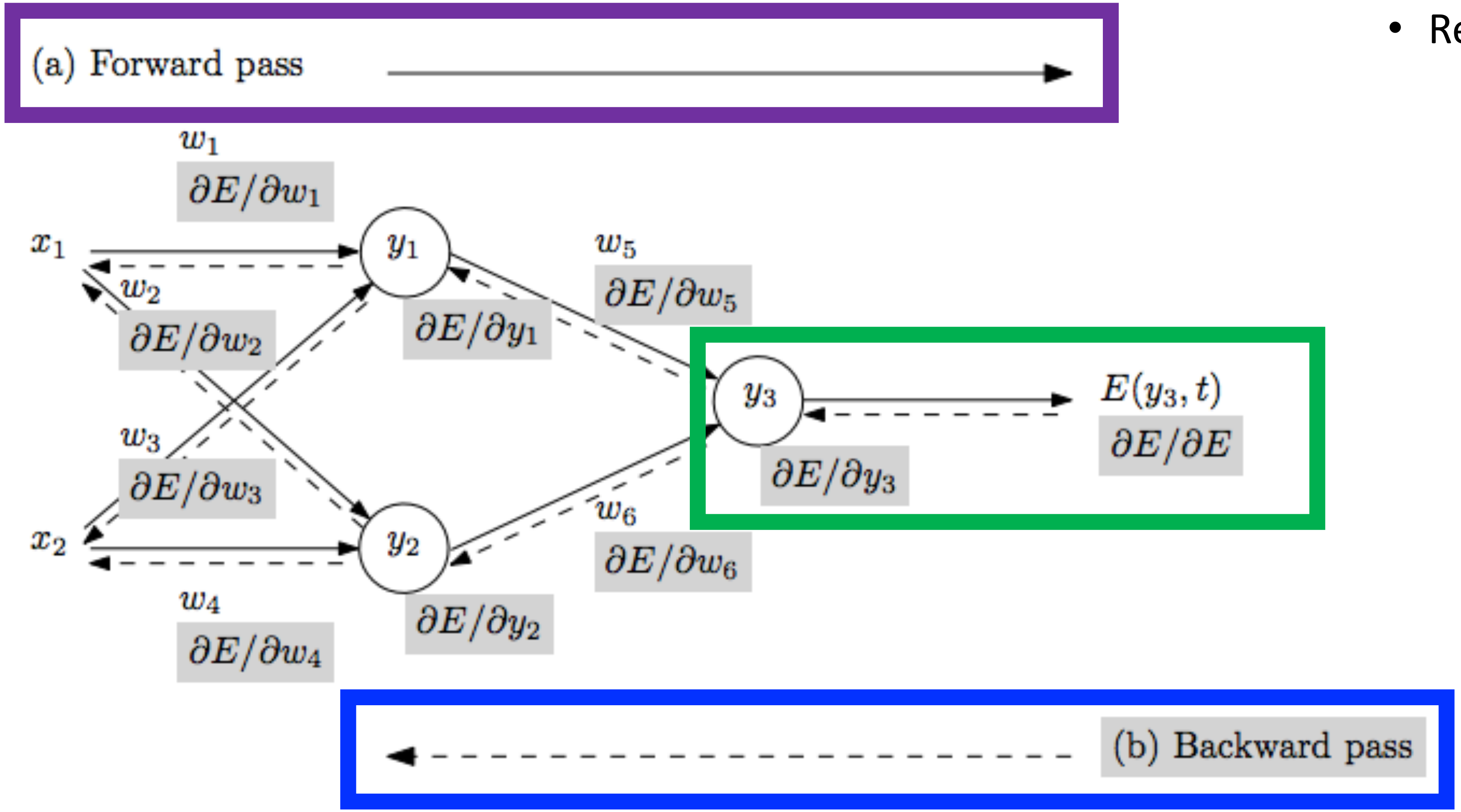
- Idea: iteratively adjust constant (i.e., model parameter) to try to reduce the error

Gradient Descent: Intuition

- Iteratively search for model parameters (e.g., weights and biases) that solve optimization problem (i.e., minimize or maximize an objective function)

Analogy: hiking to the bottom of a mountain range… blind or blindfolded!

Gradient Descent: Implementation



- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure dissatisfaction with a model's predictions on training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Gradient Descent: Implementation

Key challenge: calculating gradients

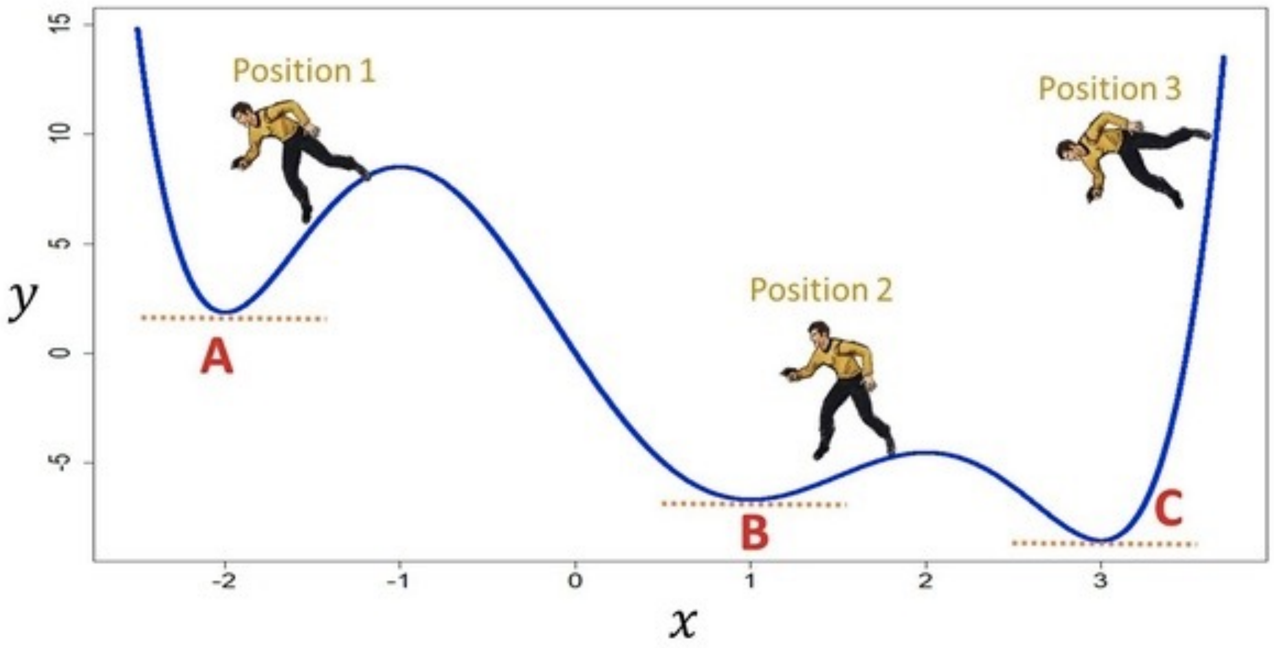
Solution: backpropagation

- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure dissatisfaction with a model's predictions on training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

D. Rulhart, G. Hinton, and R. Williams, Learning Internal Representations by Error Propagation, 1986.

Backpropagation Basics: Employs Calculus

- Idea: use derivatives!
  - Derivatives tells us how to change the input x to make a small change to the output f(x)
  - Gradient is a vector that indicates how f(**x**) changes as each function variable changes (i.e., partial derivatives)
- Gradient descent:
  - Iteratively take steps in the opposite direction of the gradient to minimize the function



Which letter(s) are the global minima?

Which letter(s) are local minima?

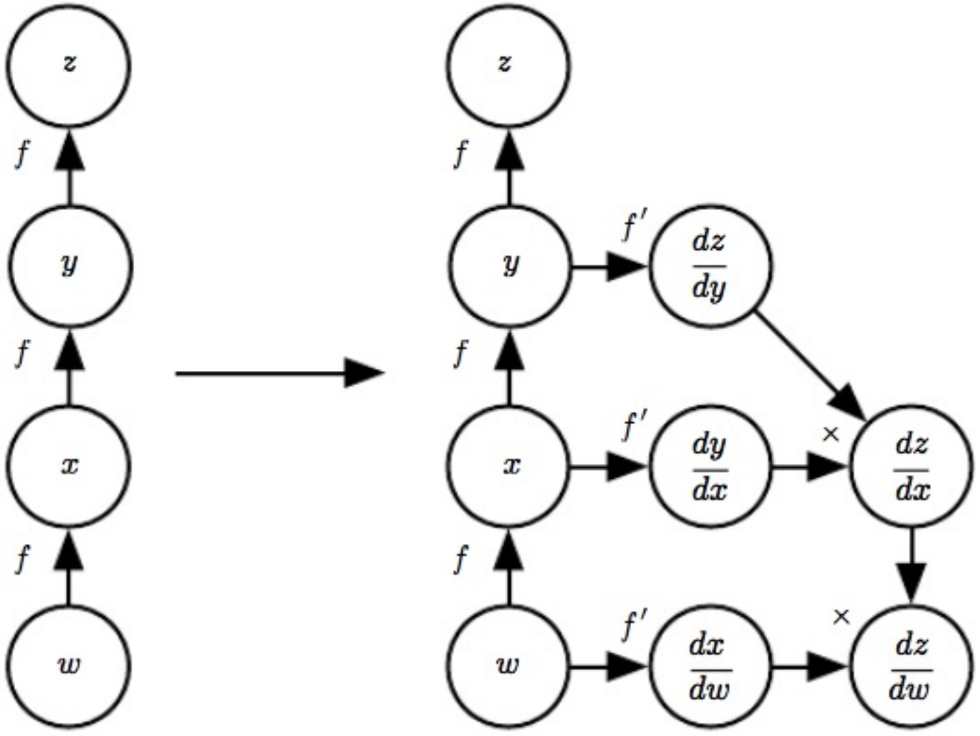
Backpropagation Basics: Chain Rule

- Idea: compute gradient on objective function to decide how to adjust each model parameter to get closer to solving the optimization problem
- Key observation: networks are functions connected in a chain

$$x = f(w),\ y = f(x),\ z = f(y)$$

Can use chain rule of calculus (and so compute from top to bottom where derivatives on the top are used to compute derivatives at the bottom);

e.g., $\dfrac{dz}{dx} = \dfrac{dz}{dy}\dfrac{dy}{dx}$



Ian Goodfellow, Yoshua Bengio, and Aaron Courville; Deep Learning, 2016.

Gradient Descent: Implementation

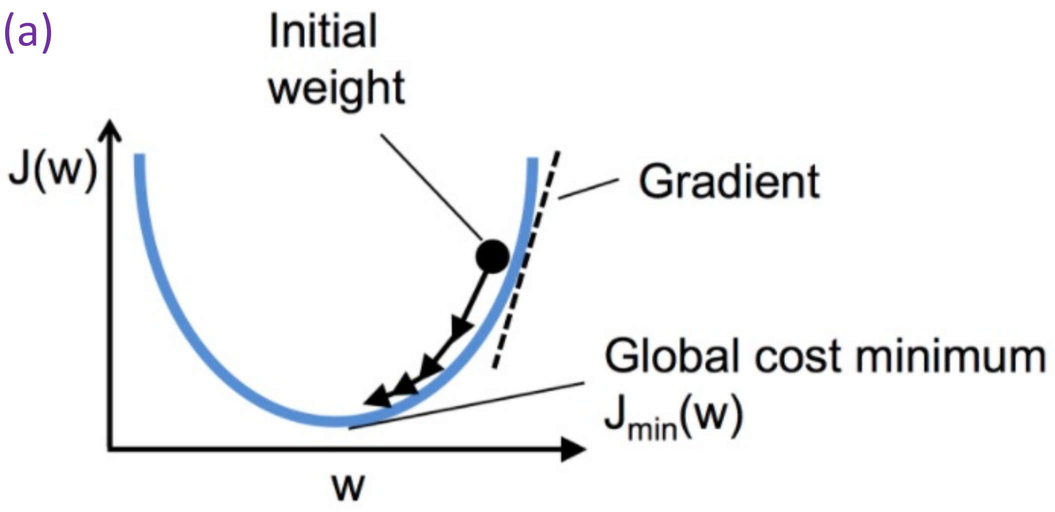
- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure dissatisfaction with a model's predictions on training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

D. Rulhart, G. Hinton, and R. Williams, Learning Internal Representations by Error Propagation, 1986.

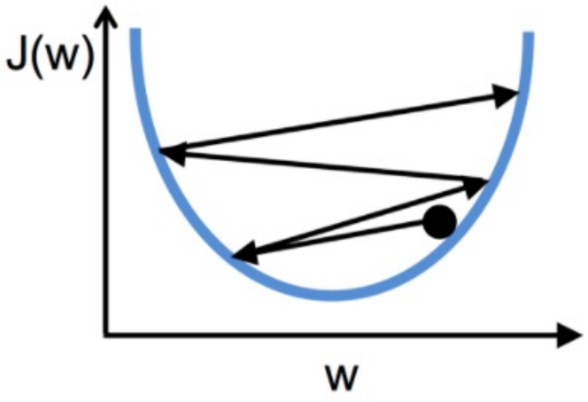
Gradient Descent: How Much to Update?

- Step size = learning rate
  - (a) When learning rate is too small, convergence to good solution will be slow
  - (b) When learning rate is too large, convergence to a good solution is not possible

(a)



(b)



- Many ways to use the gradients

https://github.com/rasbt/python-machine-learning-book-2nd-edition/blob/master/code/ch02/ch02.ipynb

Gradient Descent: Implementation

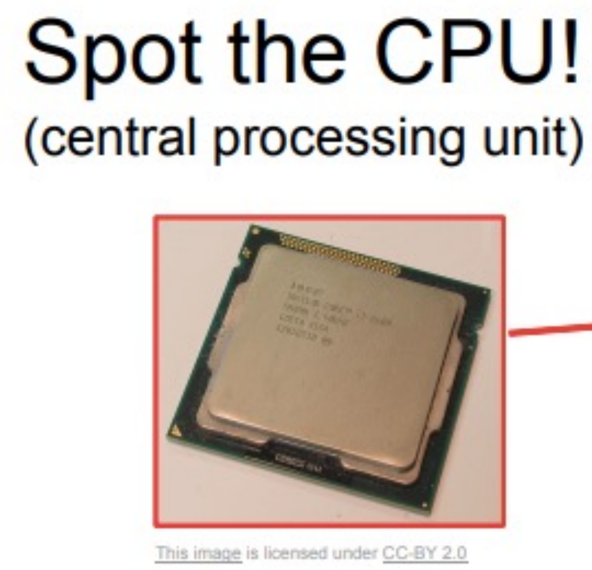
For excellent step-by-step tutorial, watch this video:

https://www.youtube.com/watch?v=VMj-3S1tku0

Critical Foundation for Training: Hardware

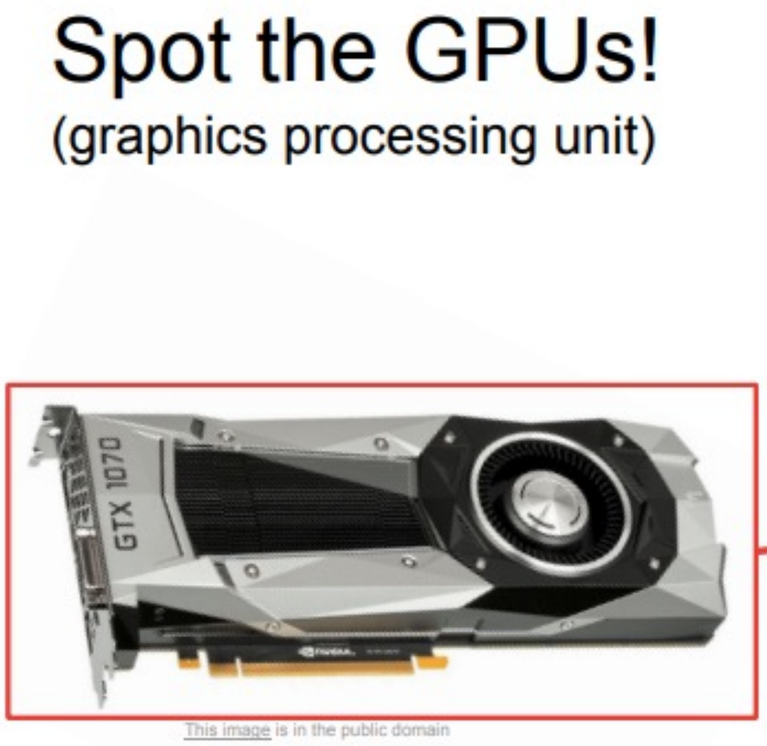
Idea: Train Algorithms Using
GPUs (think Porsche) Instead of CPUs (think Golf Cart)

Hardware: CPU versus GPU



http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture08.pdf

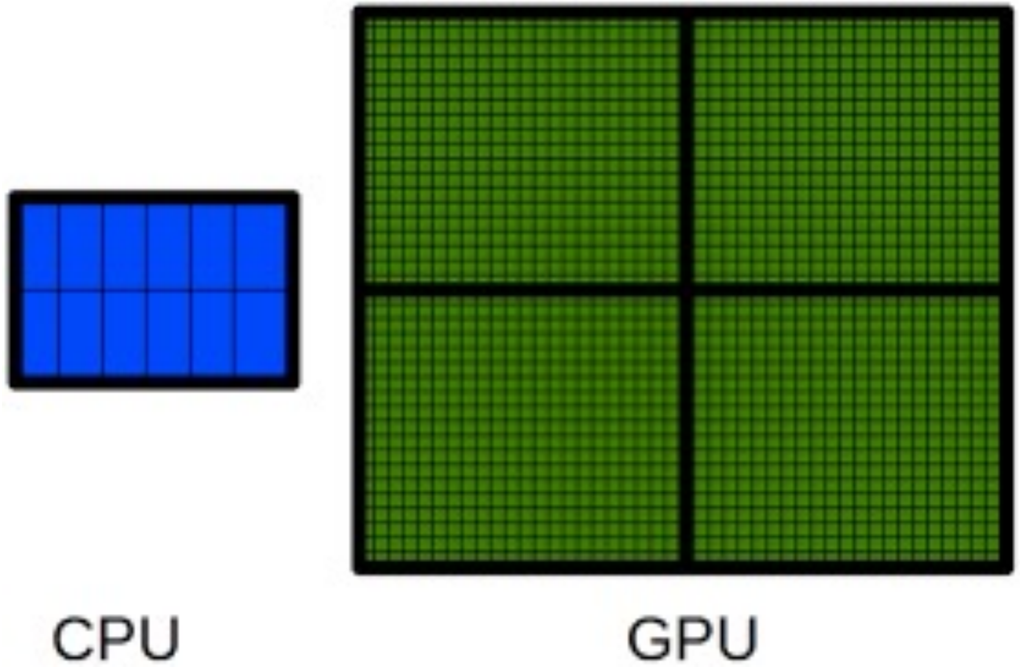
Hardware: CPU versus GPU



http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture08.pdf

Hardware: CPU versus GPU

- Graphical Processing Units: accelerates computational workloads due to MANY more processing cores



https://www.researchgate.net/figure/The-main-difference-between-CPUs-and-GPUs-is-related-to-the-number-of-available-cores-A_fig7_273383346

GPU Machines: Rent Versus Buy?

Rent from Cloud

(e.g., Microsoft Azure):

Instance	Core(s)	RAM	Temporary storage	GPU	Pay as you go with AHB
ND96asr A100 v4	96	900 GiB	6,500 GiB	8x A100 (NVlink)	$27.197/hour

Buy:

Lambda Bare Metal

- 4-8x NVIDIA A100 SXM4 GPUs
- Install in your Datacenter or Lambda Colo
- Customize CPU, RAM, Storage & Network
- Delivered in 2-4 weeks

Starting at

$ **89,283.00**

Rise of “Deep Learning” Open Source Platforms

torch (Collobert et al.) — 2011

theano (Bastien et al.) — 2012

Caffe (Jia et al.) — 2013

2014

Keras

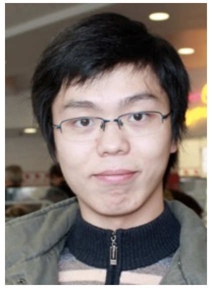
TensorFlow™ (Abadi et al.) — 2015

2016

PYTORCH (Paszke et al.) — 2017

Université de Montréal

idiap RESEARCH INSTITUTE



Berkeley UNIVERSITY OF CALIFORNIA

Google

Popular Options Today

Today’s Topics

- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Fundamentals of a neural network architecture
- Training deep neural networks


The End