Introduction to Neural Networks in Computer Vision

Danna Gurari
University of Colorado Boulder
Fall 2021

https://home.cs.colorado.edu/~DrG/Courses/RecentAdvancesInComputerVision/AboutCourse.html

Review

- Last week:
  - Computer vision: origins
  - What makes computer vision hard?
  - Research in computer vision
  - Course logistics

- Assignments (Canvas)
  - Ranking of topics for student-led lectures due tomorrow (Thursday)
  - New reading assignment out due next Wednesday

- Questions?

Today’s Topics

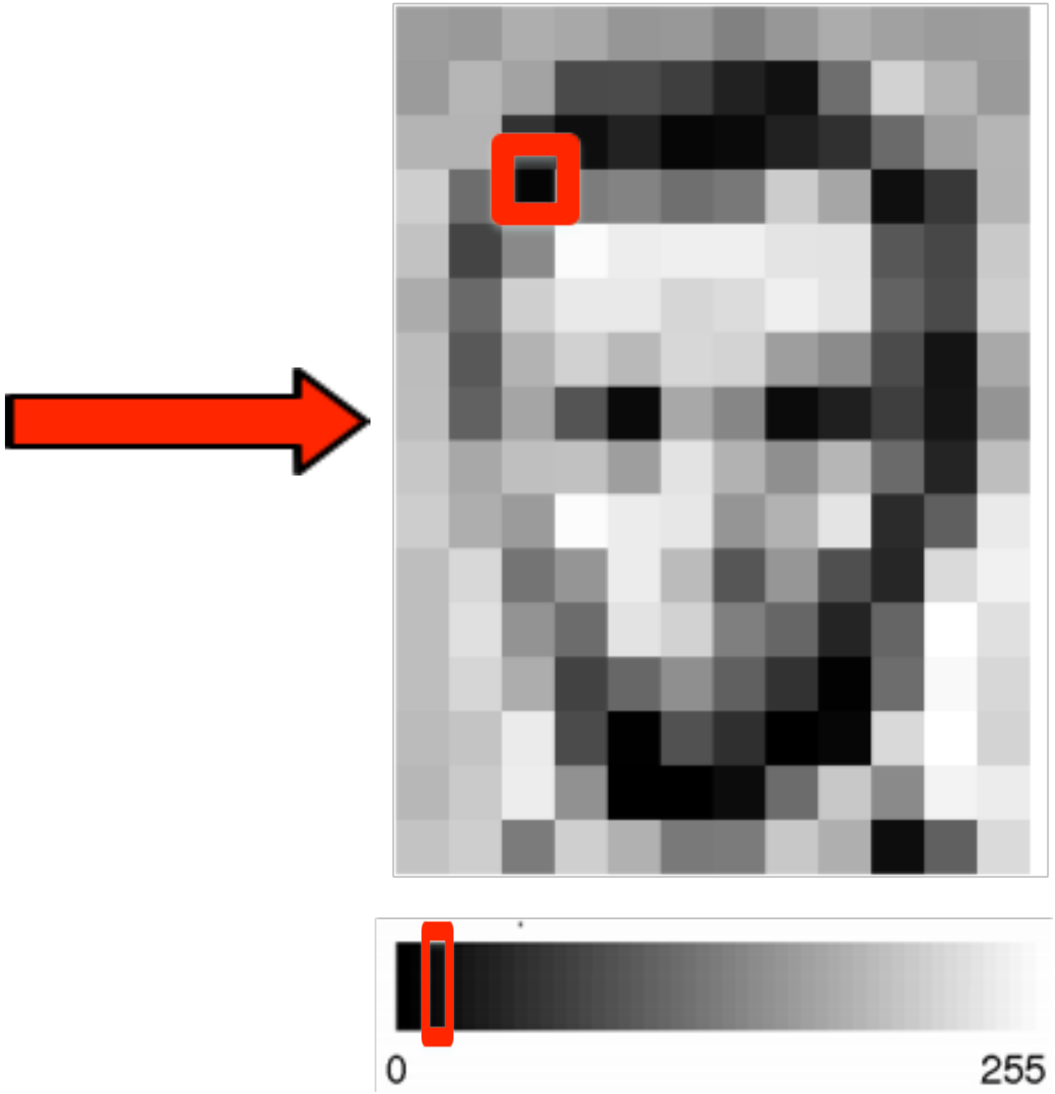
- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Background of machine learning and neural networks
- Training deep neural networks: hardware & software

Today's Topics

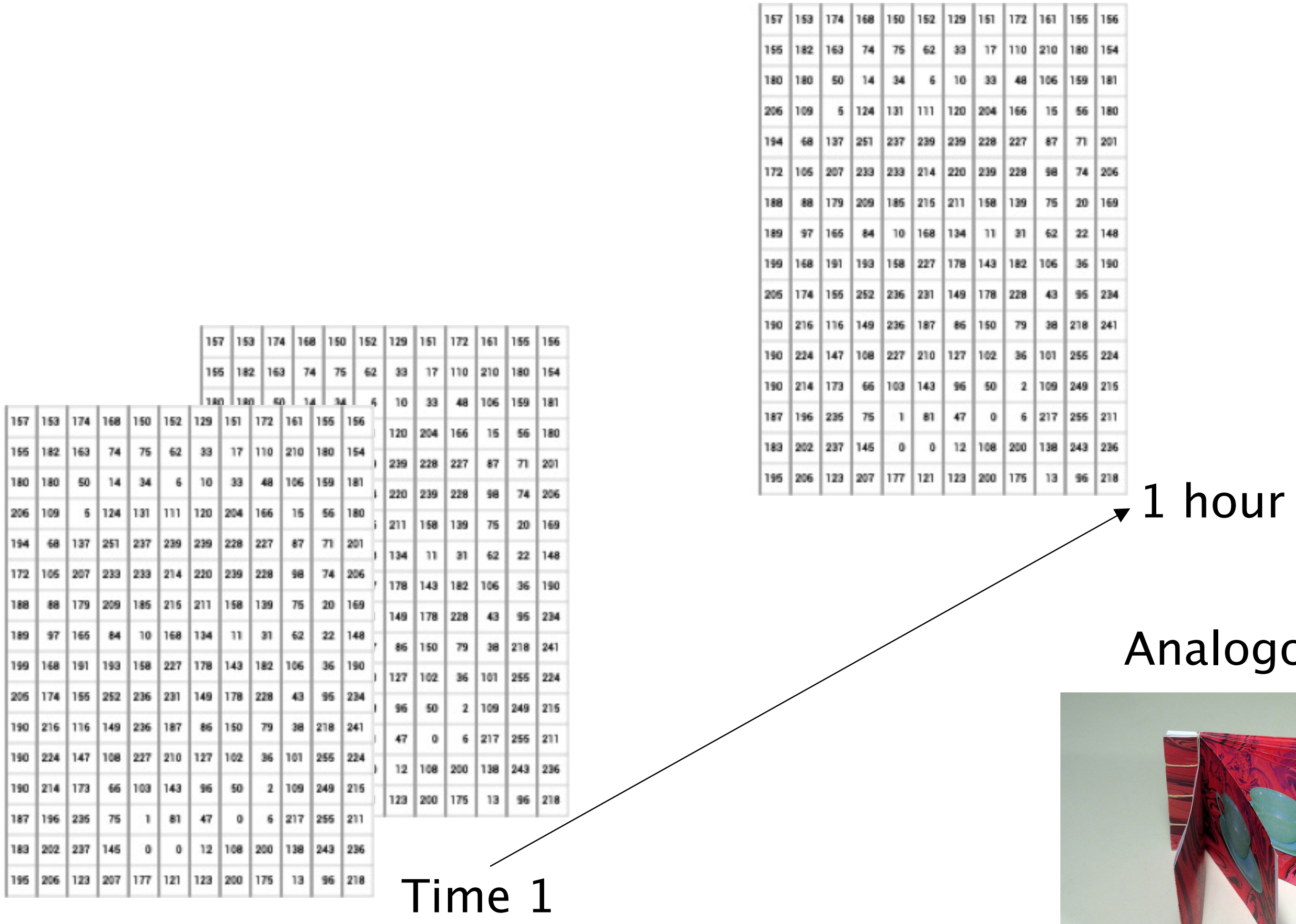
- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Background of machine learning and neural networks
- Training deep neural networks: hardware & software

Recall What a Machine Observes: Digital Image

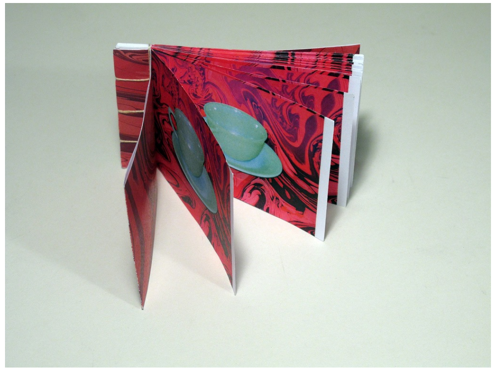
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	[illegible]	5	[illegible]	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	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

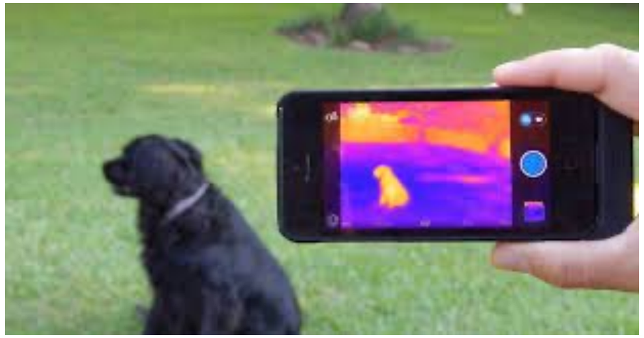


Analogous to:

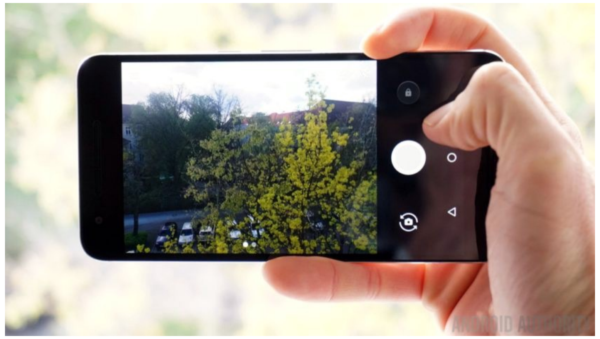


Many Ways to Create Digital Images and Videos

Ultrasound



Infrared



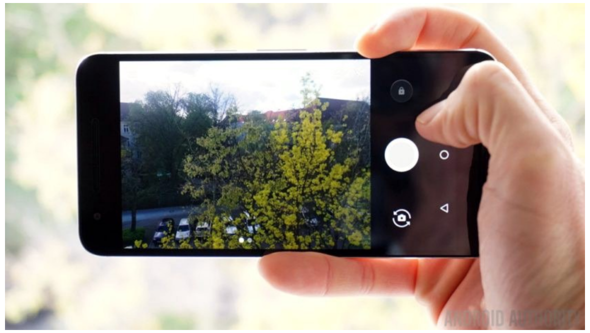
Visible

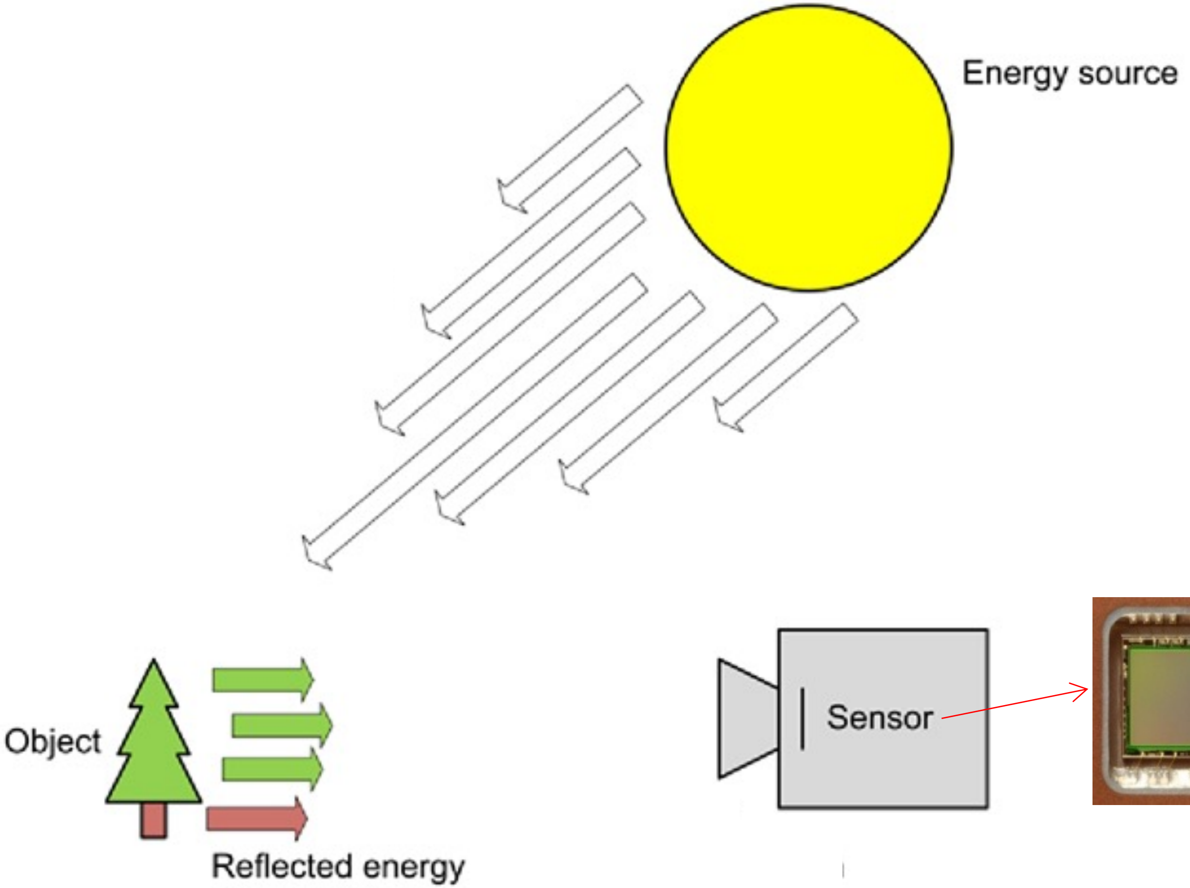
X-ray

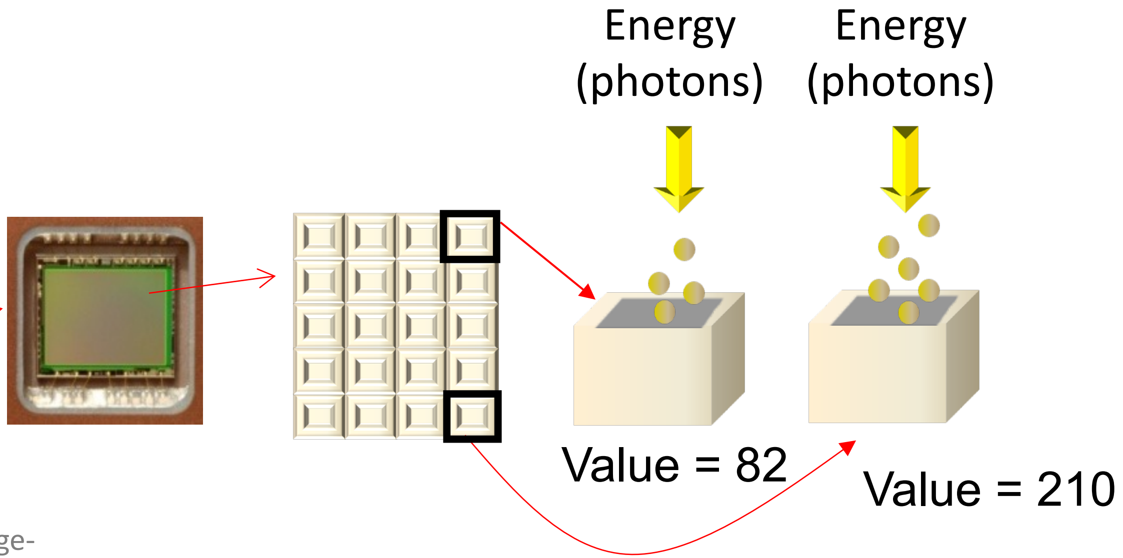
Microscopy

Many Ways to Create Digital Images and Videos

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



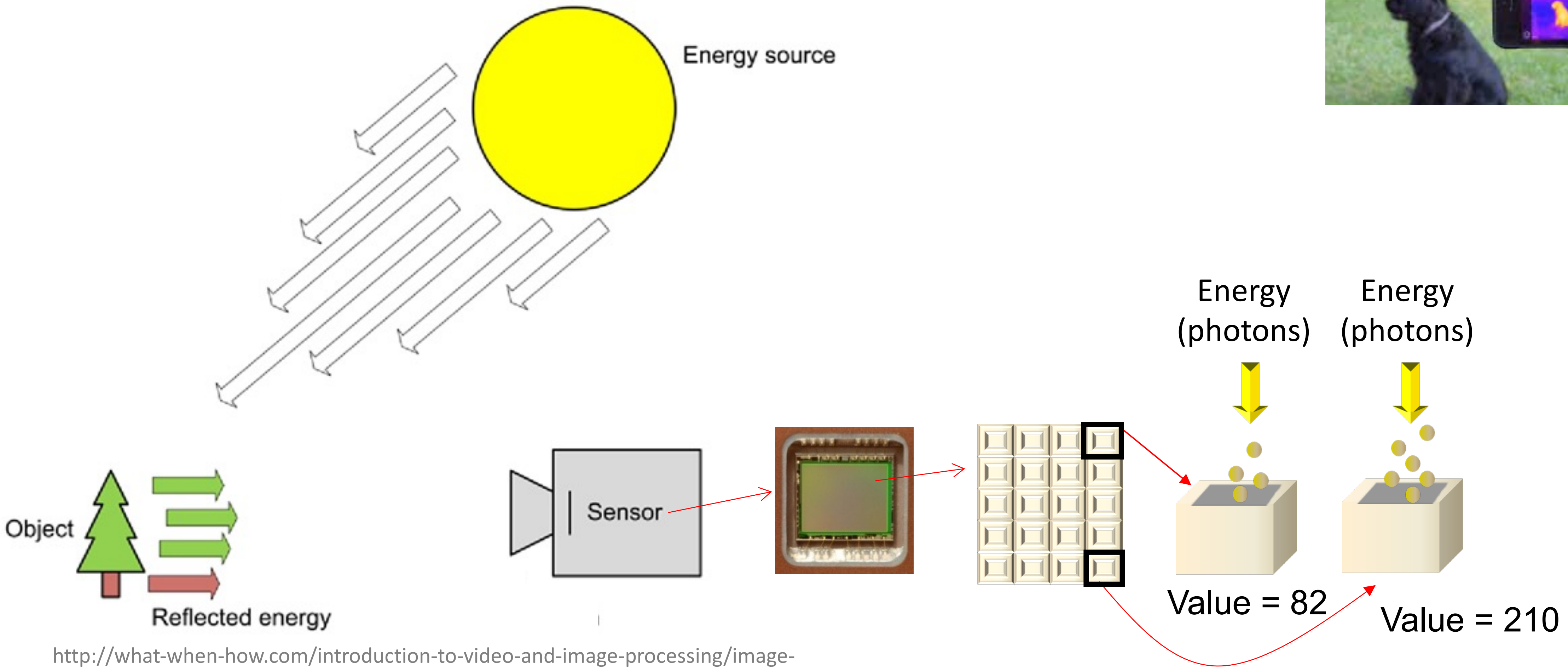




http://what-when-how.com/introduction-to-video-and-image-processing/image-acquisition-introduction-to-video-and-image-processing-part-1/

Many Ways to Create Digital Images and Videos

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

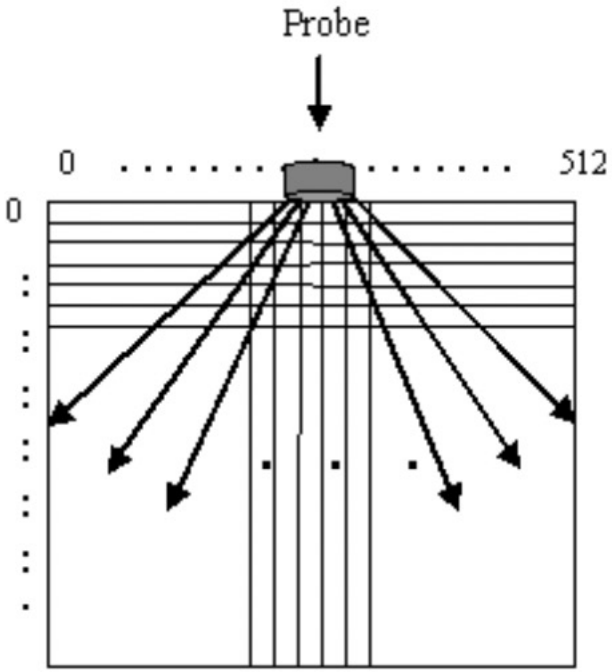


http://what-when-how.com/introduction-to-video-and-image-processing/image-acquisition-introduction-to-video-and-image-processing-part-1/

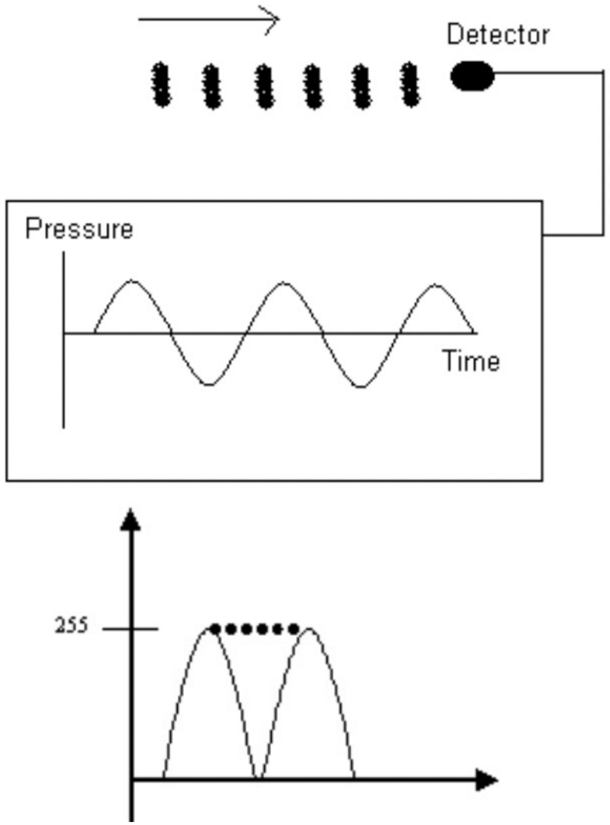
Many Ways to Create Digital Images and Videos

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

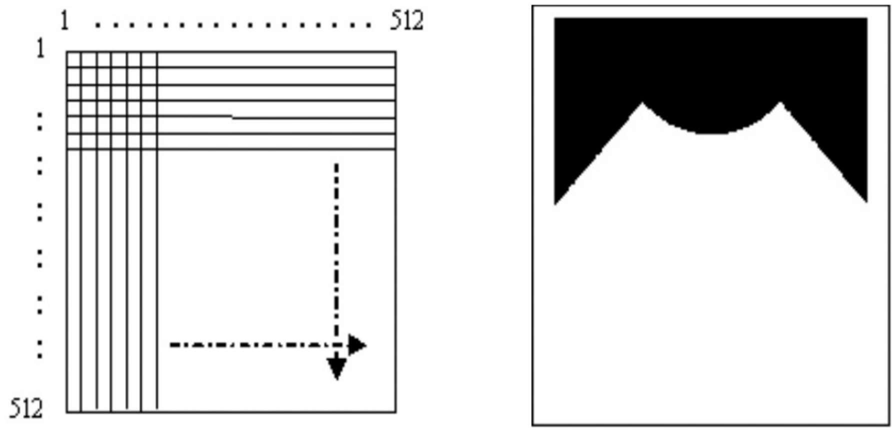
1. Sound wave generation



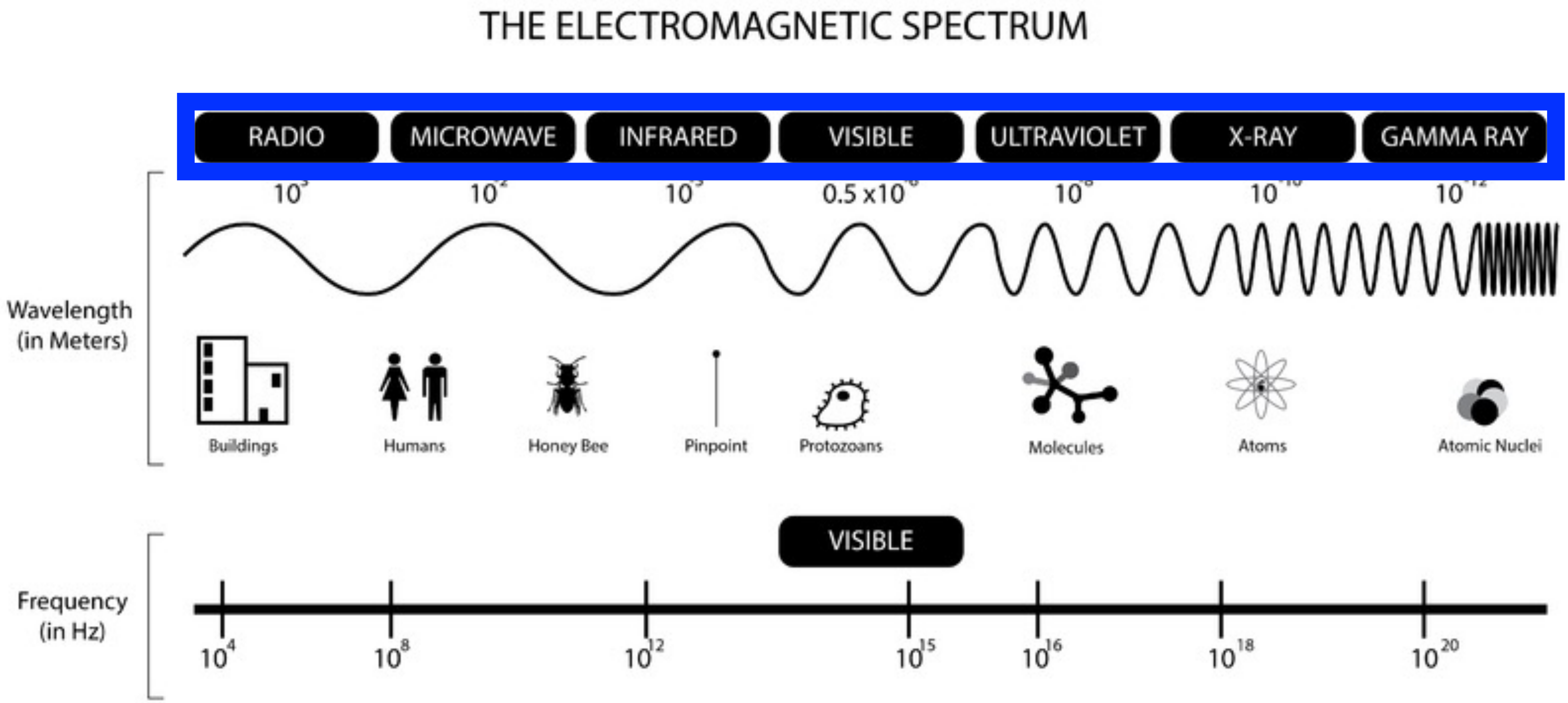
2. For each reflected sound wave, (a) record and (b) digitize to pixel values



3. Convert digitization to image



Many Ways to Create Digital Images and Videos

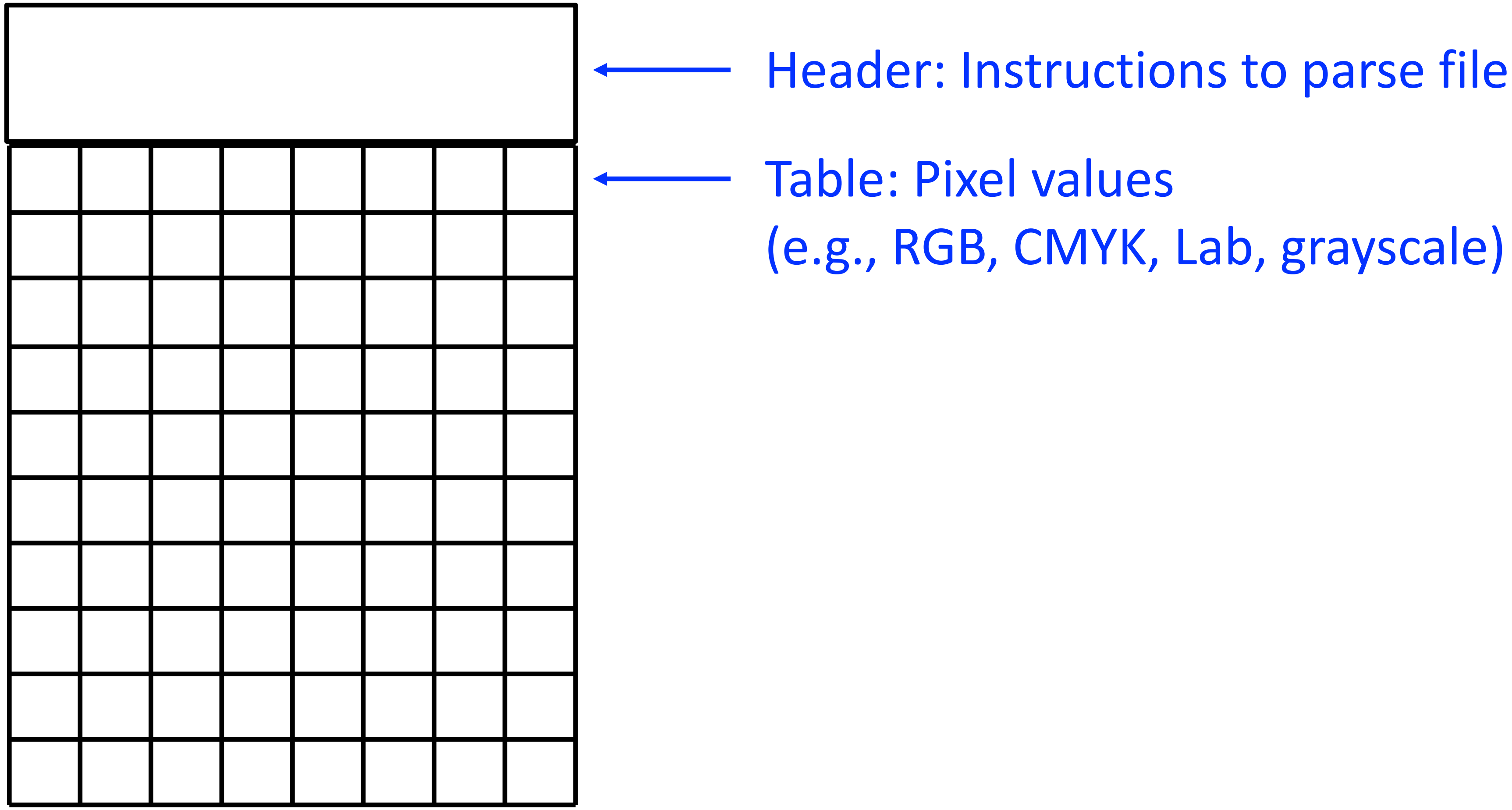


My Focus in My Career

- 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)



Scale of Vision Acquisition

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

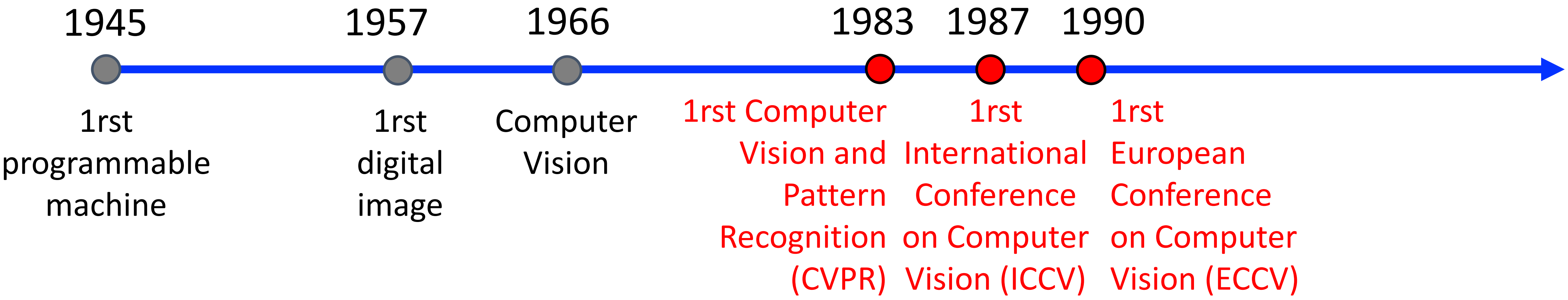
- > 85% of internet data in the form of images and videos [2]

[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

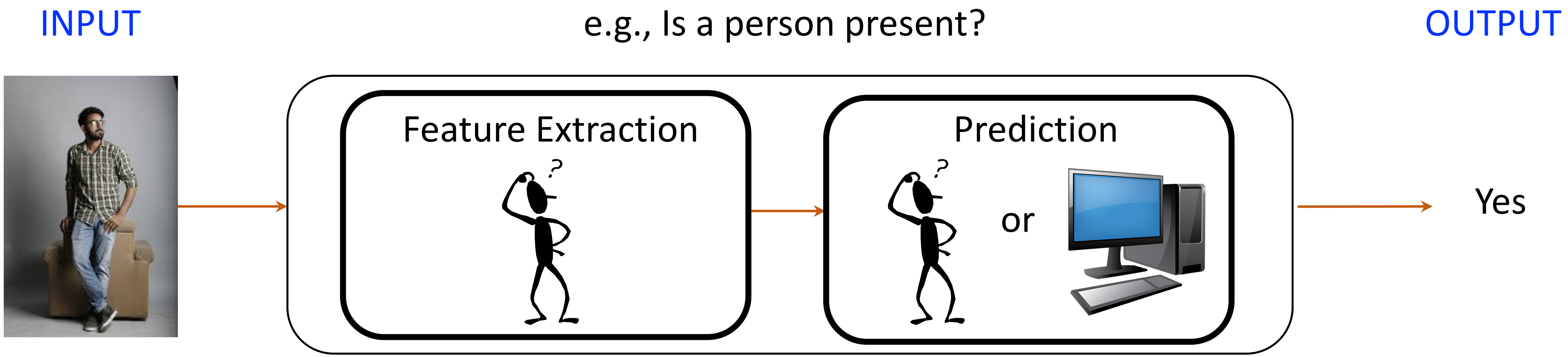
- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Background of machine learning and neural networks
- Training deep neural networks: hardware & software

Recall: Emergence of Research Community



Original Status Quo for Designing Algorithms: Handcrafted Rules

- An engineer manually designs rules to interpret an image



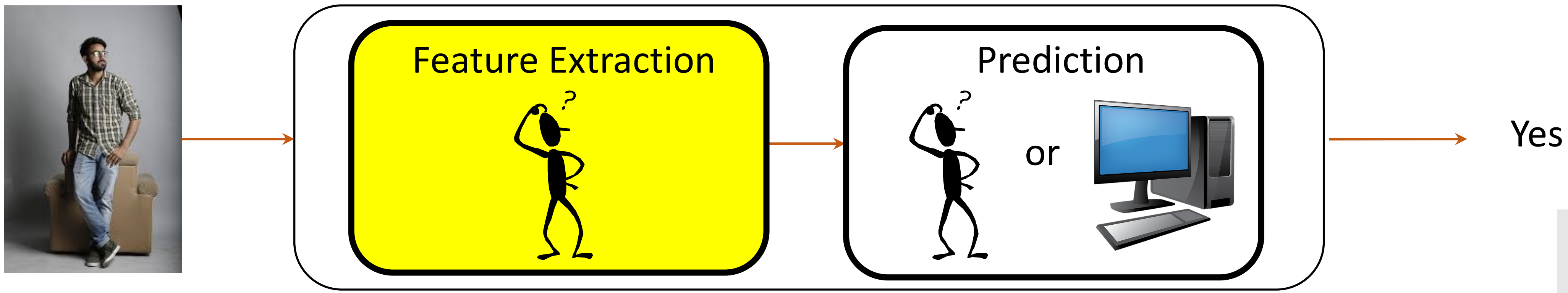
Original Status Quo for Designing Algorithms: Handcrafted Rules

- An engineer manually designs rules to interpret an image

INPUT

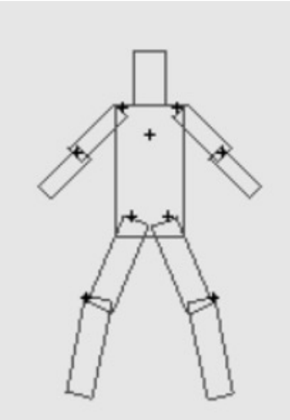
e.g., Is a person present?

OUTPUT



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

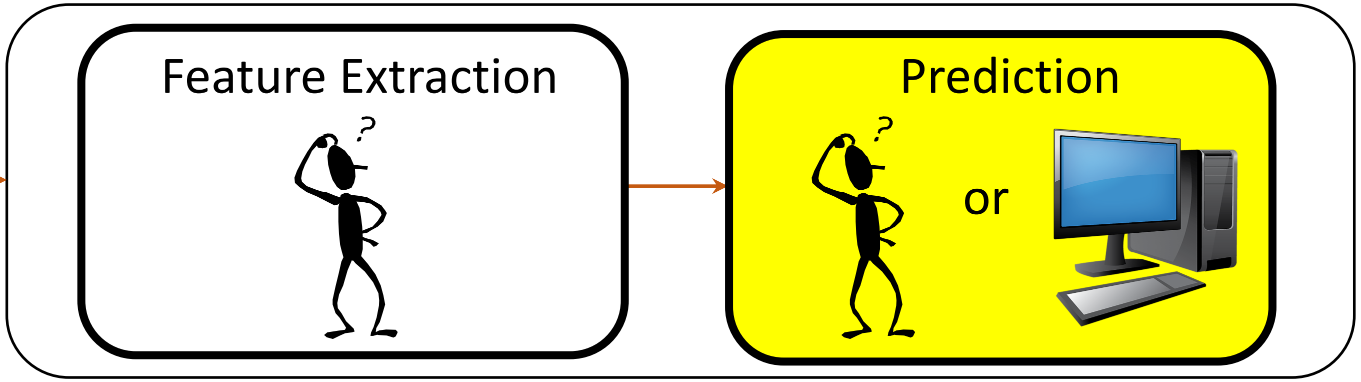
Original Status Quo for Designing Algorithms: Handcrafted Rules

- An engineer manually designs rules to interpret an image

INPUT

e.g., Is a person present?

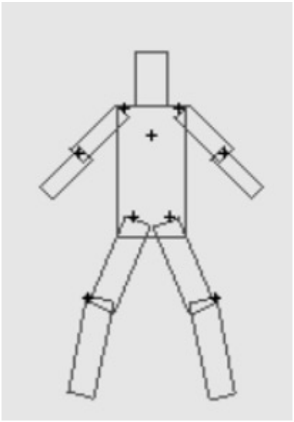
OUTPUT



Yes

What rules should be used to predict yes/no?

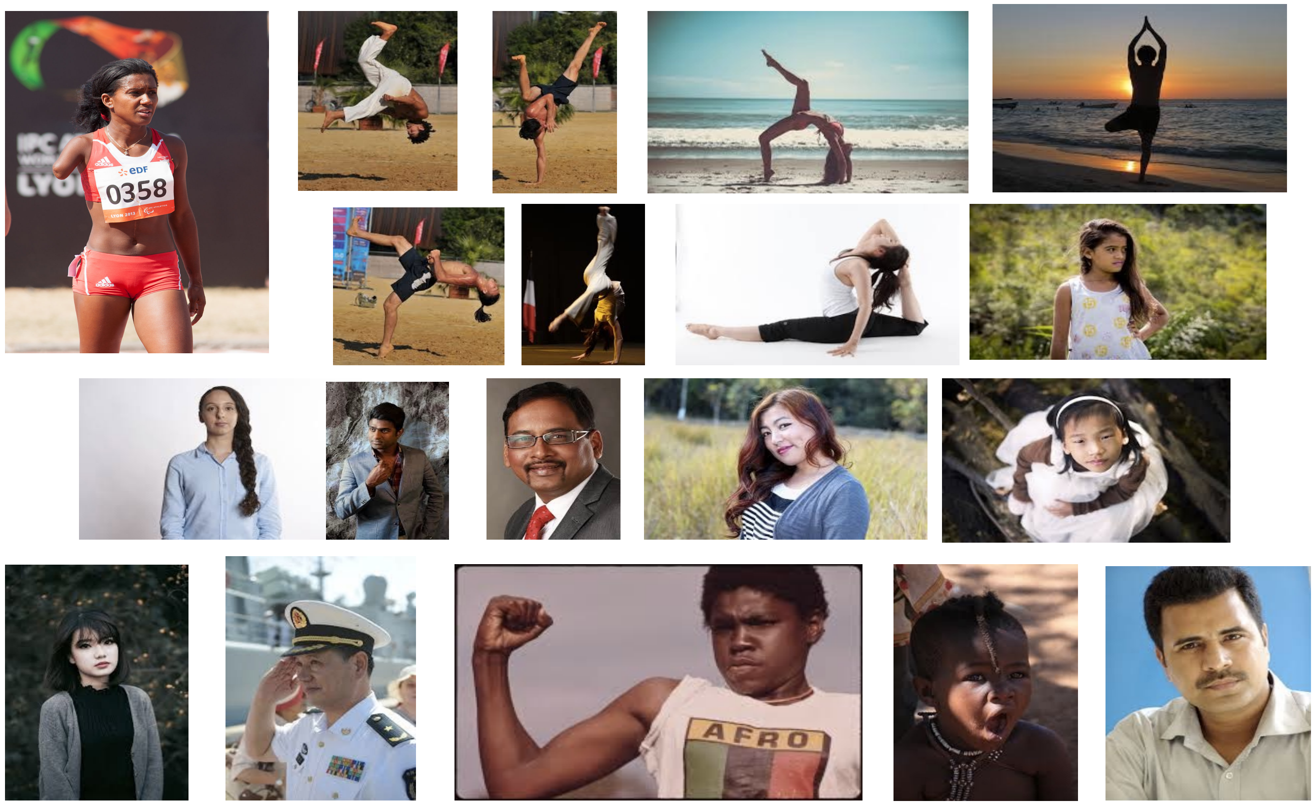
e.g., choose threshold for number of model parts detected or develop machine learning model that predicts from handcrafted features



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

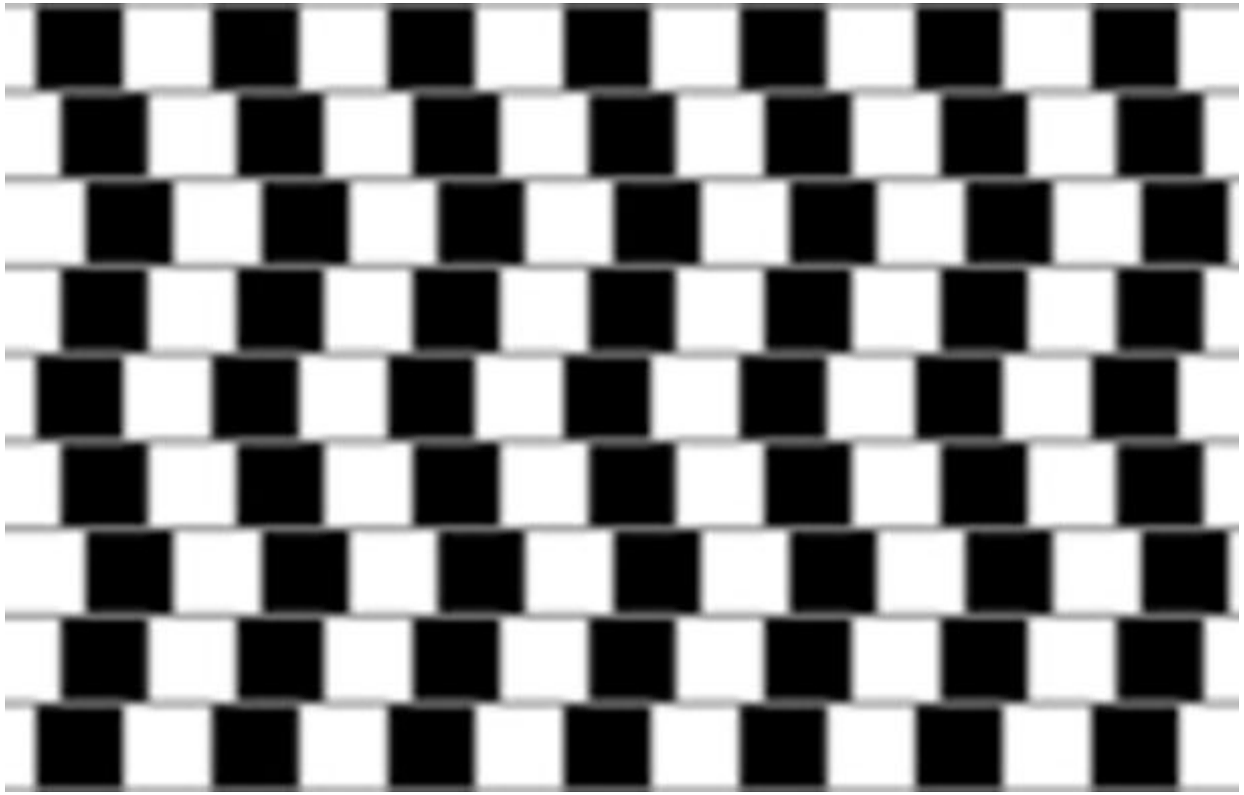
Limitations of Handcrafted Rules

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



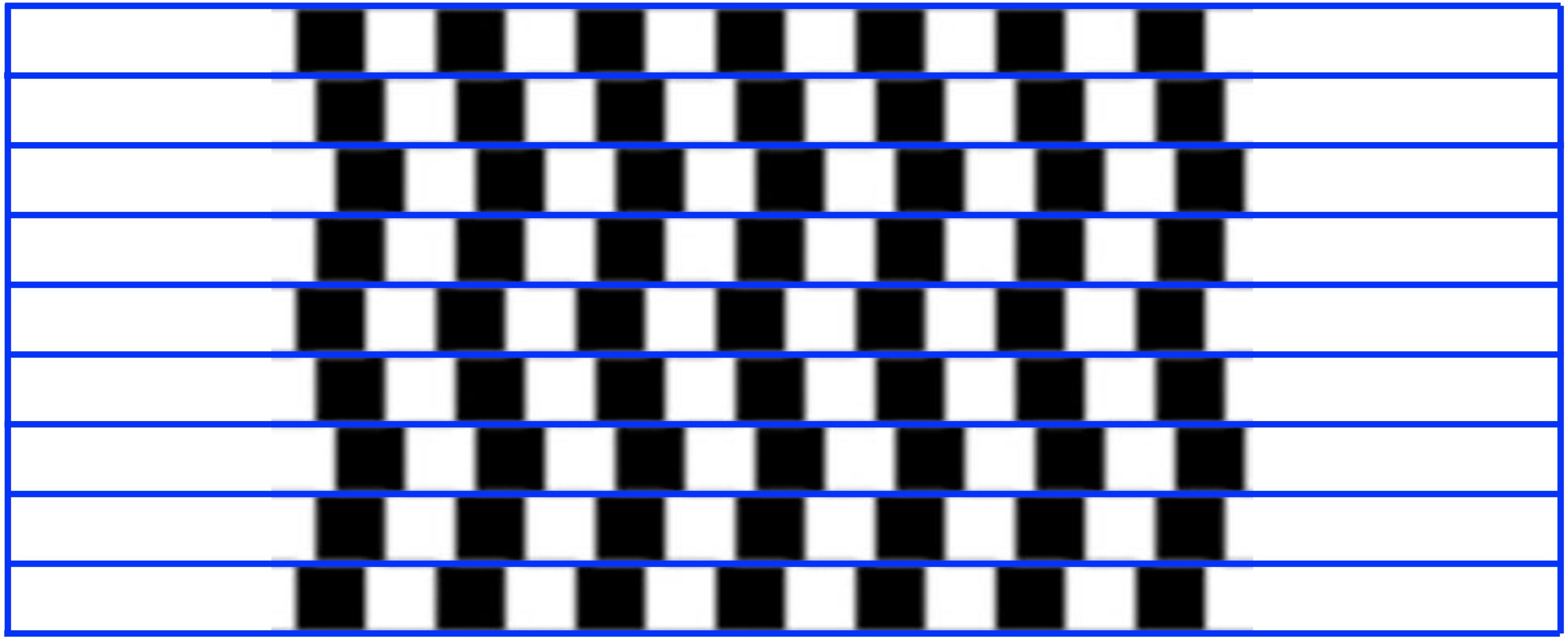
Limitations of Handcrafted Rules

e.g., are these lines parallel?



Limitations of Handcrafted Rules

e.g., are these lines parallel?



Limitations of Handcrafted Rules

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

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

Computer Vision Revolution: Catalyst

1945 — 1957 — 1966 — 1983 — 1987 — 1990 — 1998 — 2009

Computer Vision

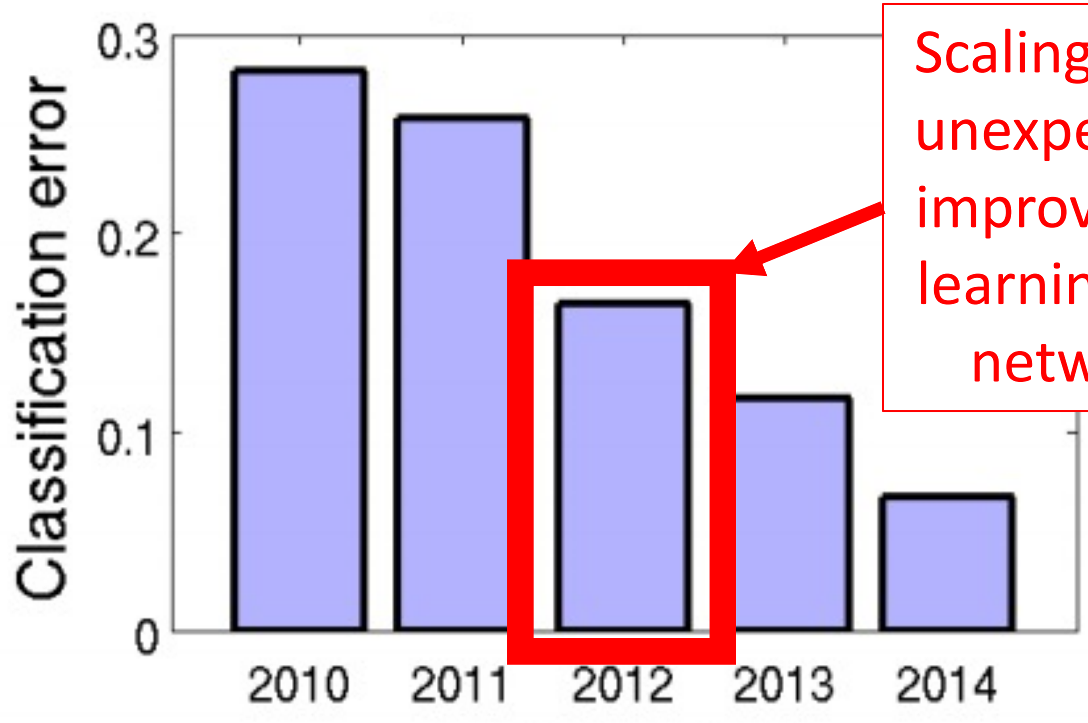
CVPR ICCV ECCV MNIST ImageNet

Large-scale dataset for recognizing objects contained in 3,200,000 images

J. Deng, W. Dong, R. Socher, L. Li, K. Li and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. 2009.

Computer Vision Revolution: Catalyst

Progress of models on ImageNet

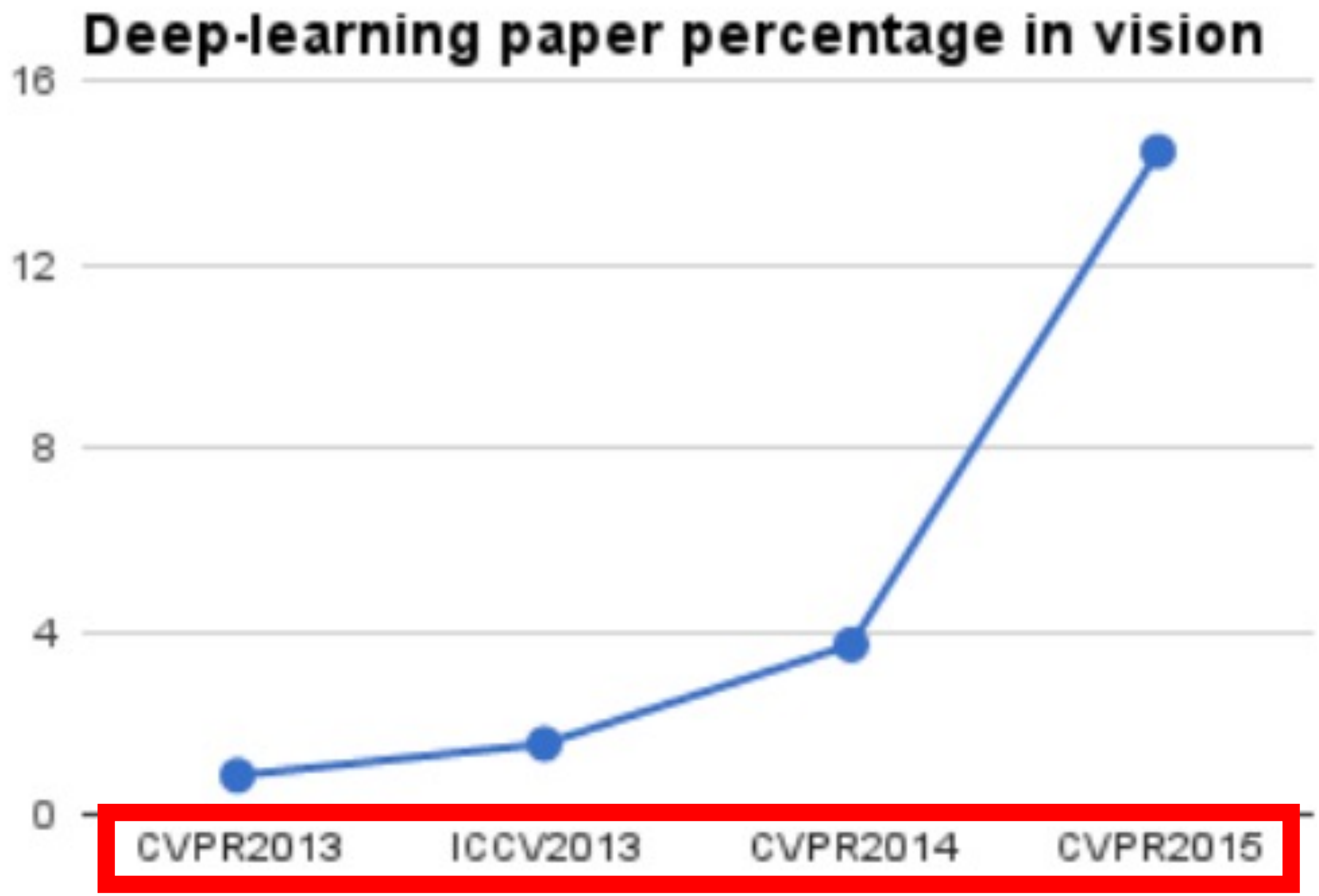


Scaling up the dataset led to unexpected, unprecedented improvements from a "deep learning" model (i.e., neural network) called AlexNet

(15.3% error, 10.8 percentage points less than that of the runner up)

Olga Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. IJCV 2015.
Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Neural Networks. NIPS 2012.

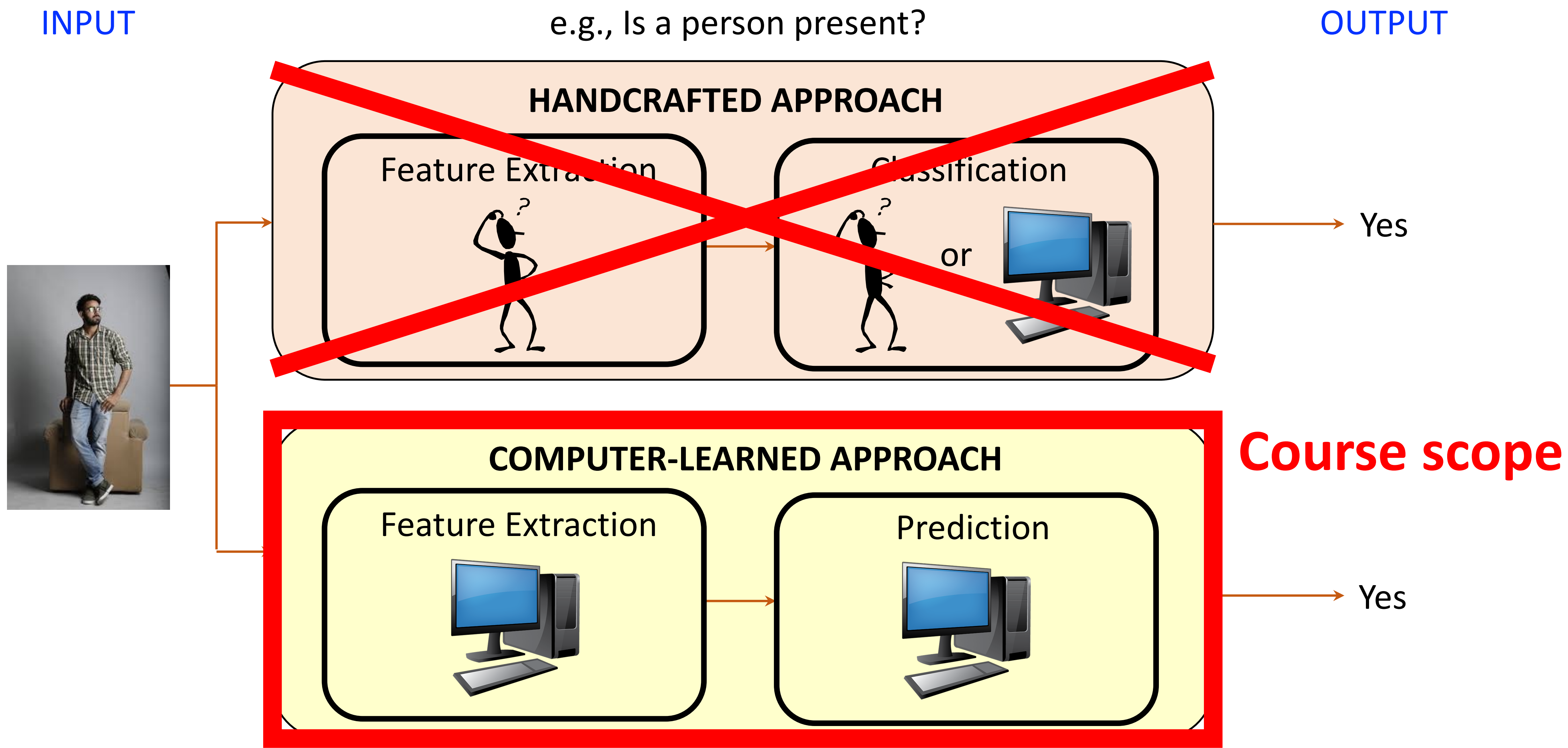
New Status Quo for Designing Algorithms: Neural Networks



Inspired, many more researchers in the computer vision community focused on neural networks and discovered they succeed for many more vision problems!

Slide Credit: https://www.slideshare.net/xavigiro/saliency-prediction-using-deep-learning-techniques

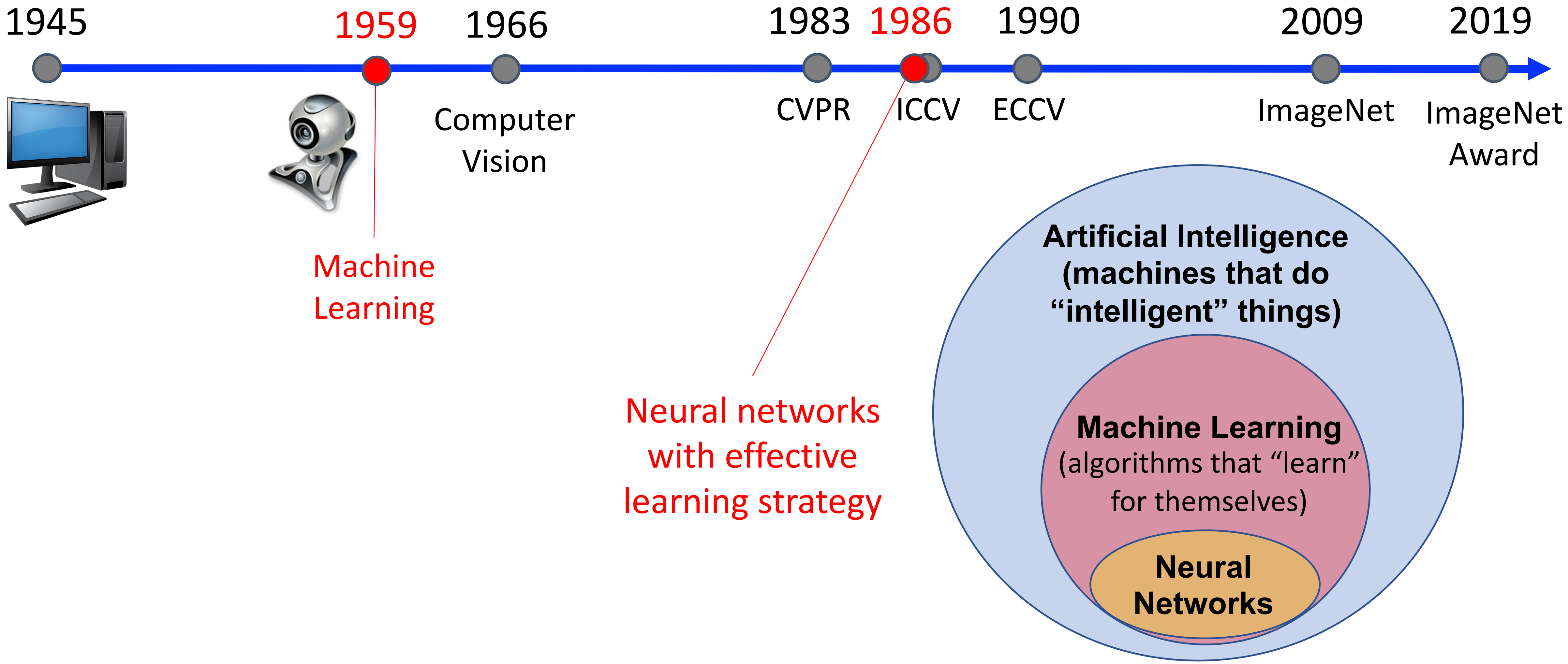
New Status Quo for Designing Algorithms: Neural Networks



Today's Topics

- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Background of neural networks
- Training deep neural networks: hardware & software

Origins of 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/dont-touch-hot-stove-clipart-73647.html

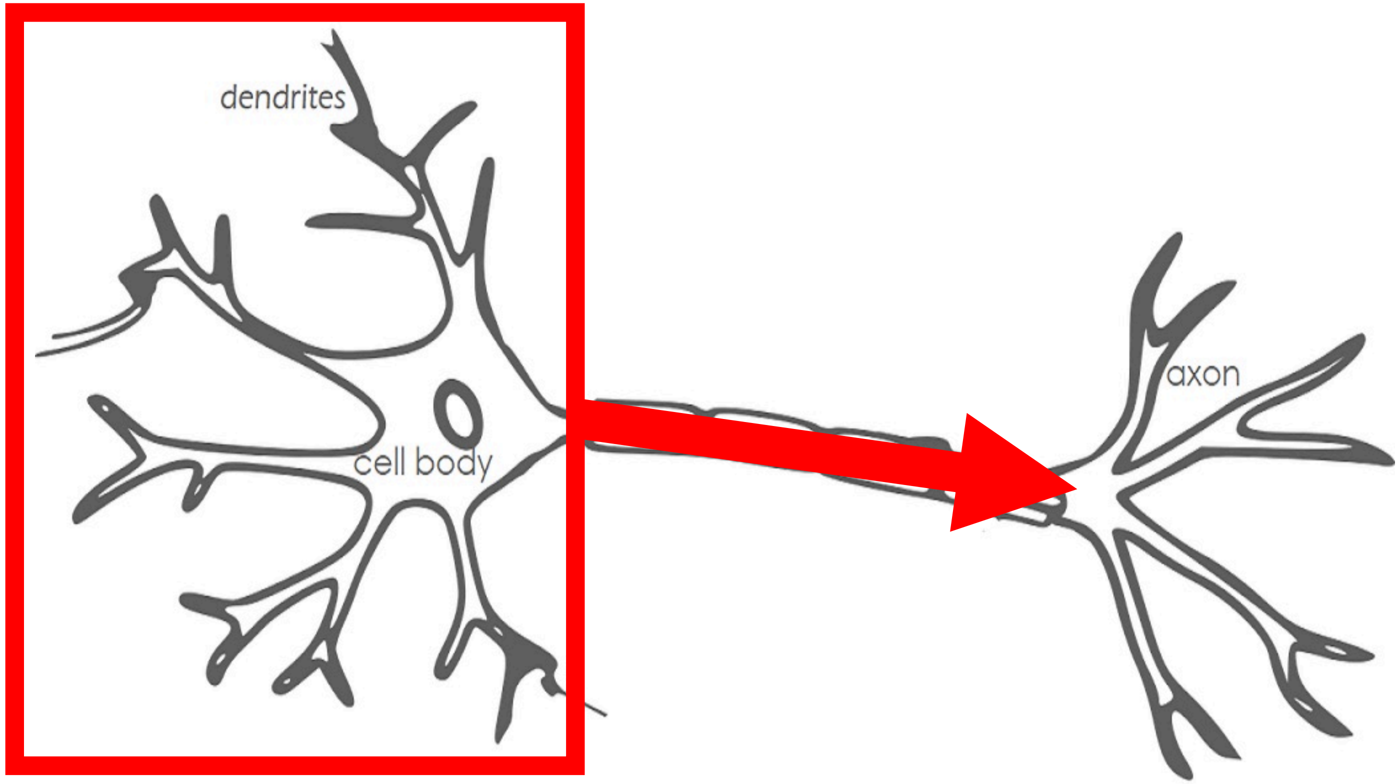
“loud”

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



- When the input signals exceed a certain threshold within a short period of time, a neuron "fires"
- Neuron "firing" (outputs signal) is an "all-or-none" process

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

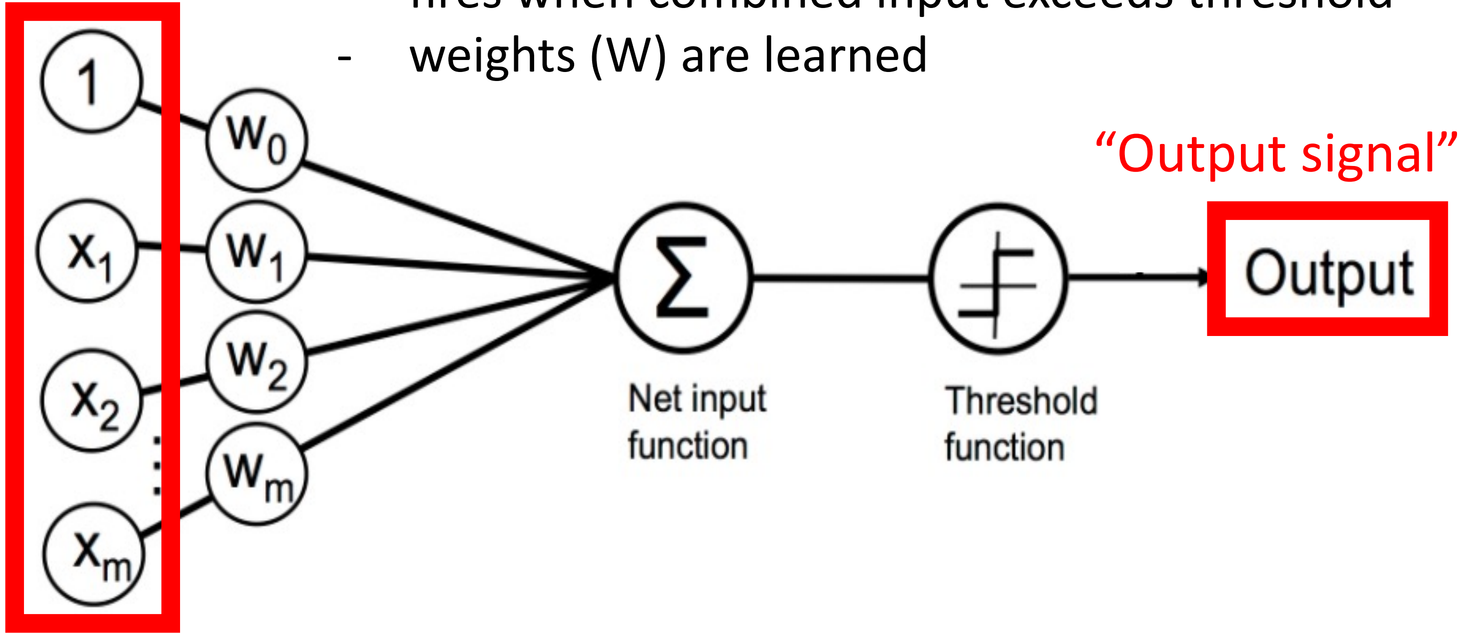
Perceptron (Artificial Neuron)

- fires when combined input exceeds threshold
- weights (W) are learned

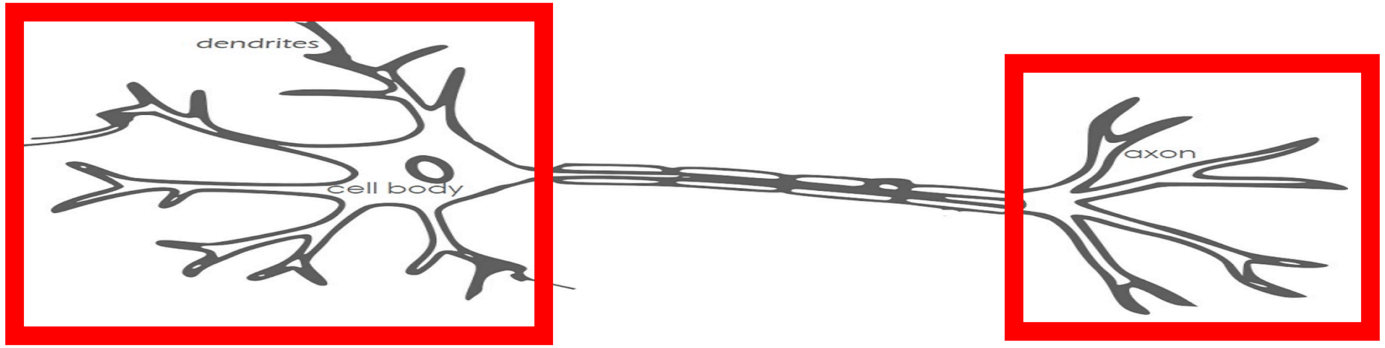
"Input signals"

"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)

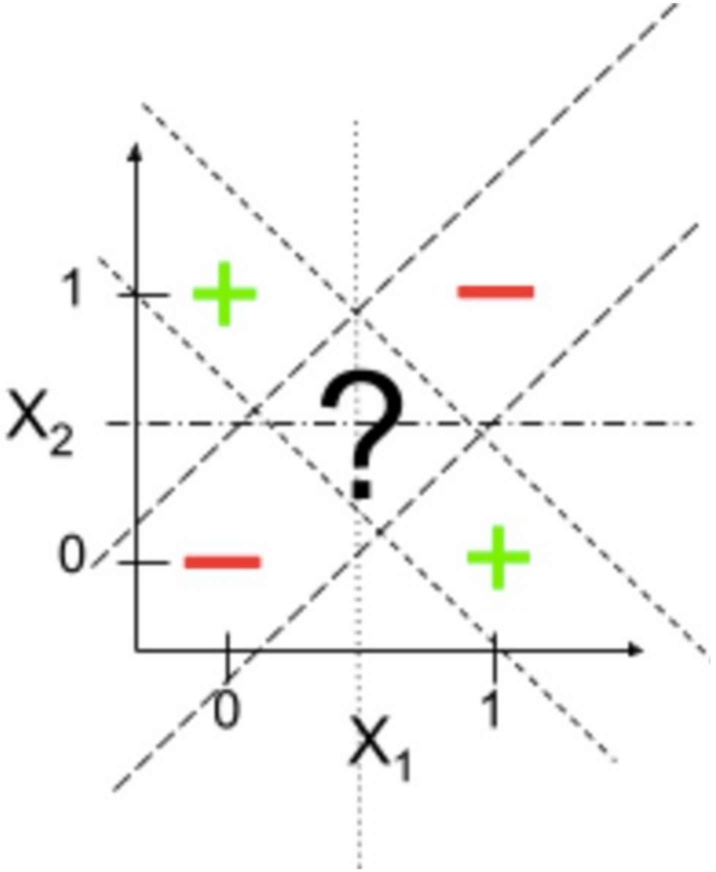
XOR = "Exclusive Or"
- Input: two binary values x_1 and x_2
- Output:
  - 1, when exactly one input equals 1
  - 0, otherwise

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)

Cannot solve XOR problem and so separate 1s from 0s with a perceptron (linear function):

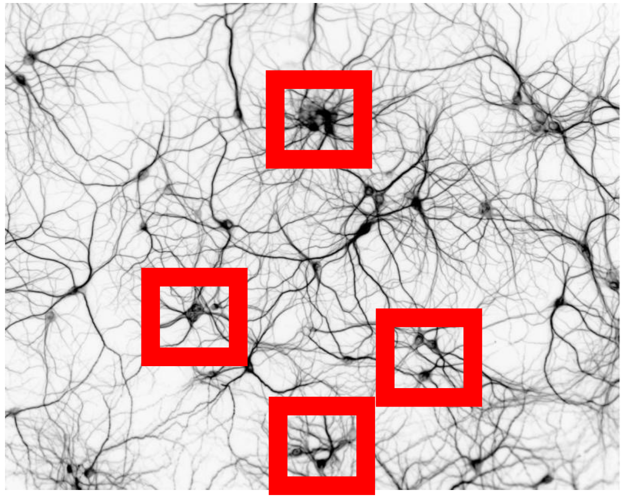


x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

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

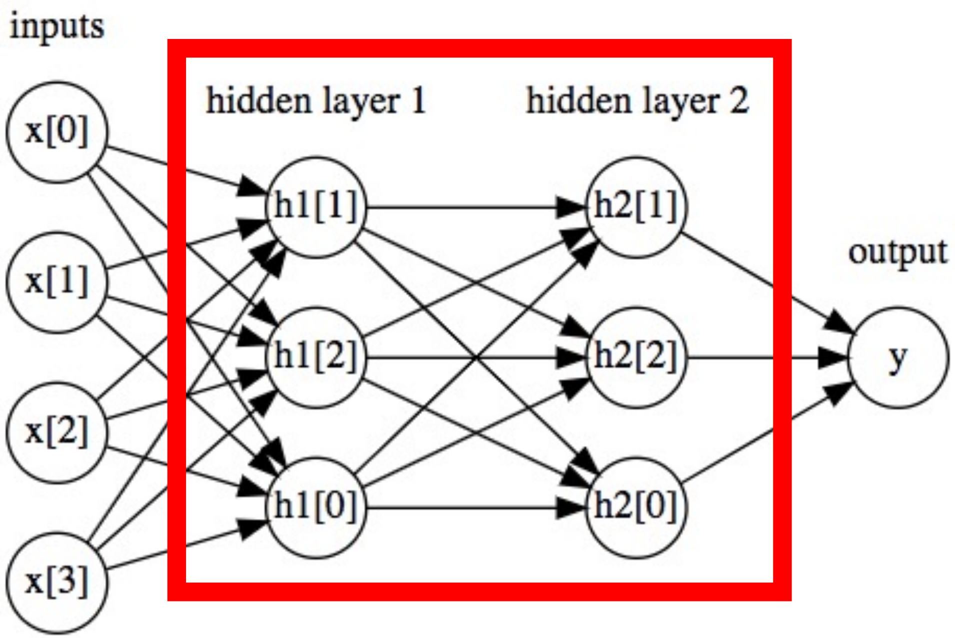
Solution to Overcome Limitation: Neural Networks (Connected Neurons)

Biological Neural Network:



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

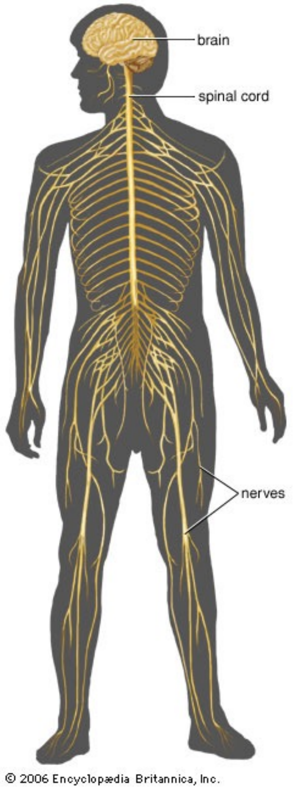
Artificial Neural Network:



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

Inspiration: Animal's Computing Machinery

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

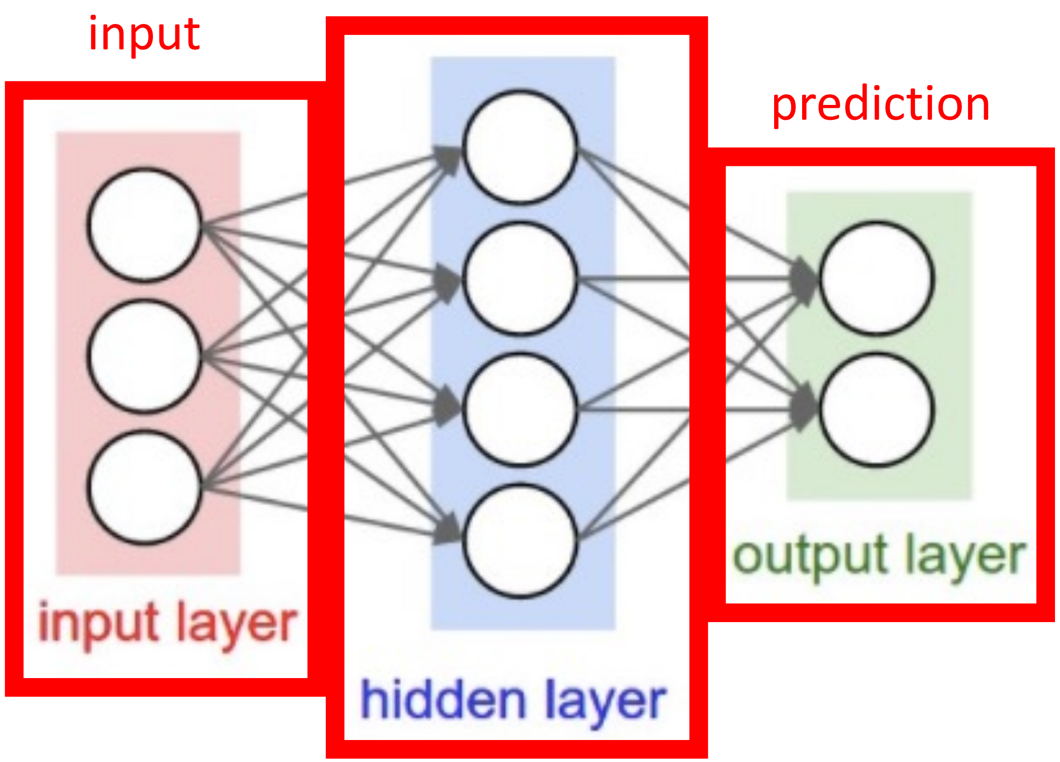


https://www.britannica.com/science/human-nervous-system

Nematode worm: 302 neurons

Human: ~100,000,000,000 neurons

Neural Network

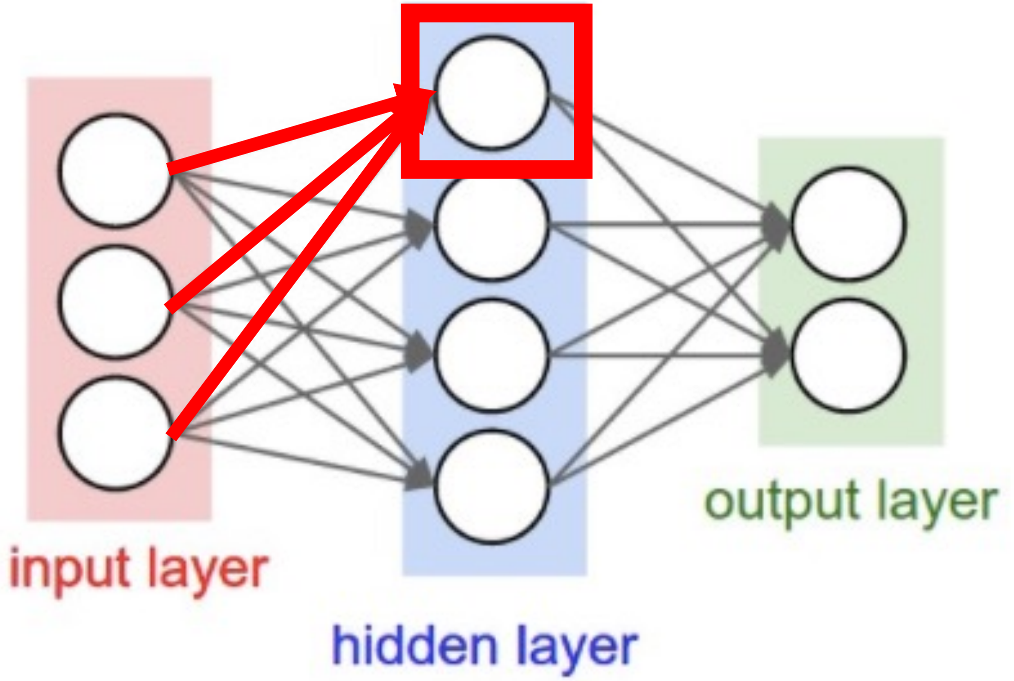


"hidden layer" uses outputs of units (i.e., neurons) and provides them as inputs to other units (i.e., neurons)

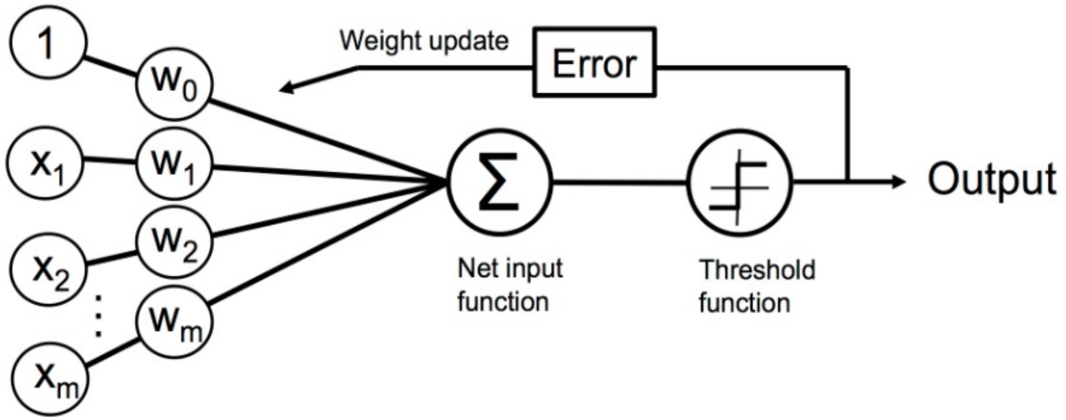
- Also called "multilayer perceptron"
- This is a 2-layer "feed-forward" neural network (i.e., count number of hidden layers plus output layer and exclude input layer)

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

Neural Network



- How does this relate to a perceptron?

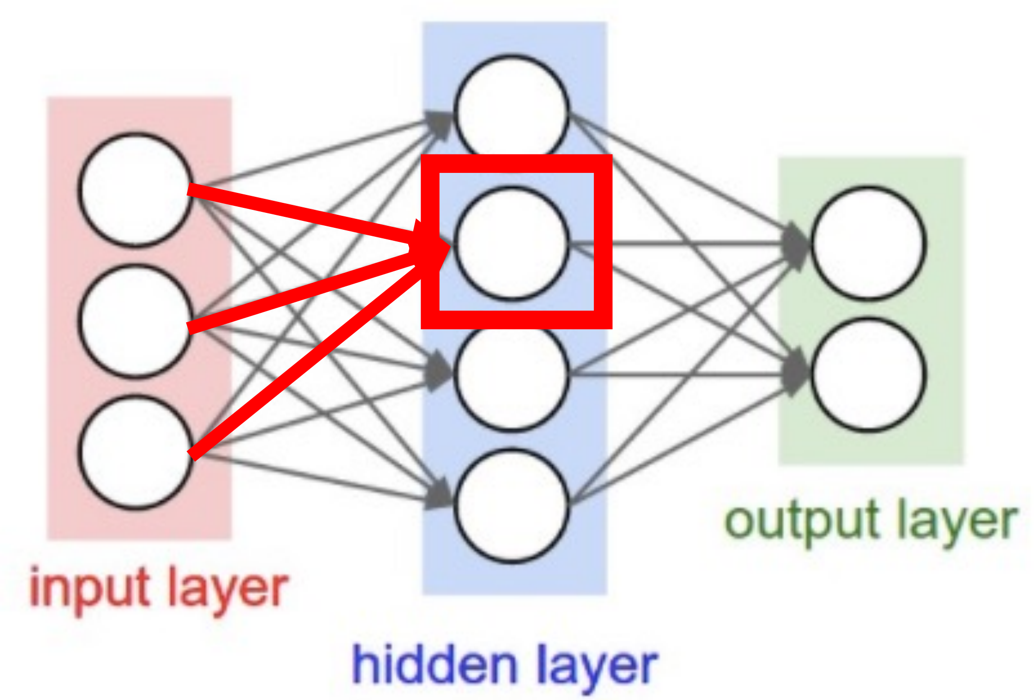


- Unit: takes as input a weighted sum and applies a function to the input

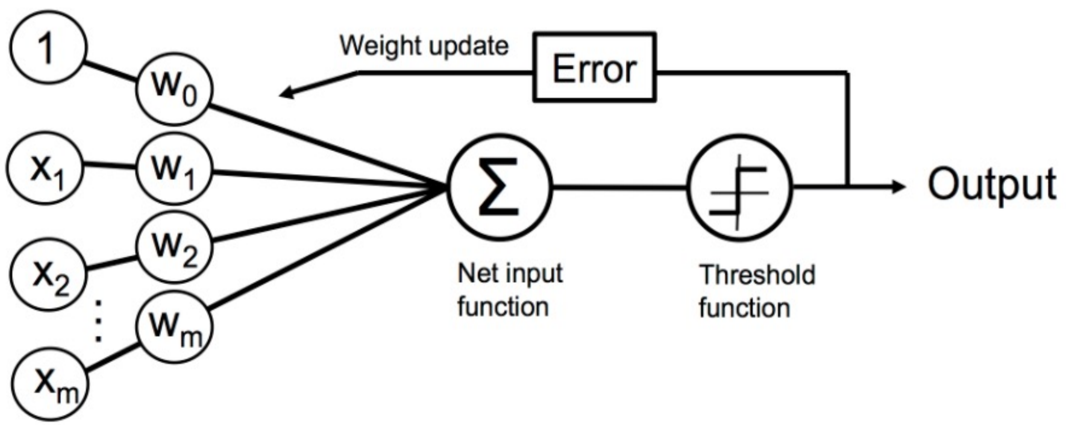
Python Machine Learning; Raschka & Mirjalili

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

Neural Network



- How does this relate to a perceptron?

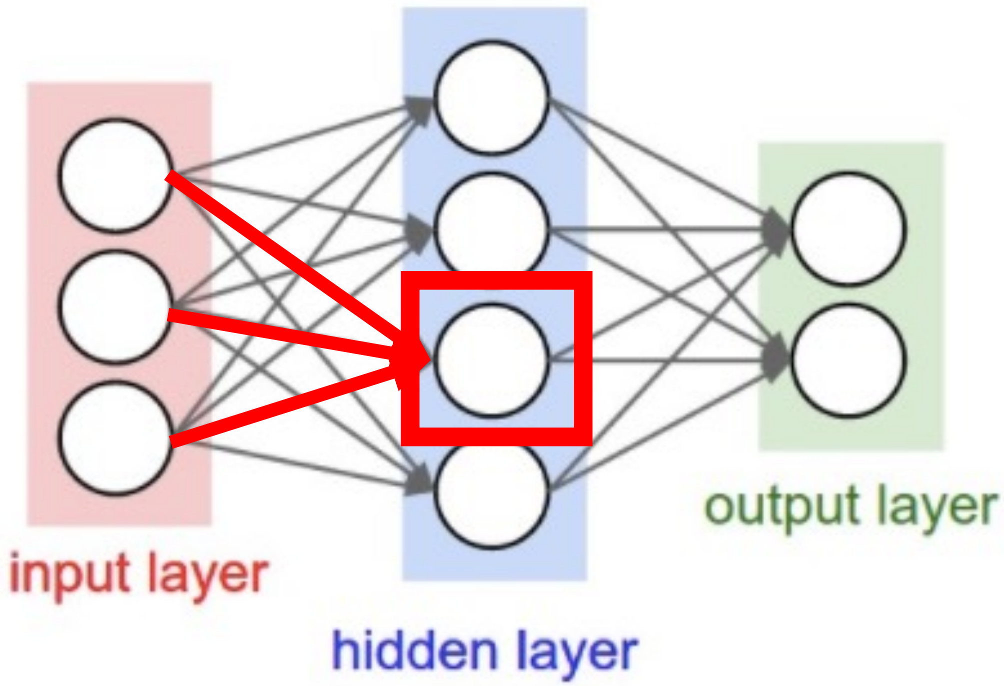


- Unit: takes as input a weighted sum and applies a function to the input

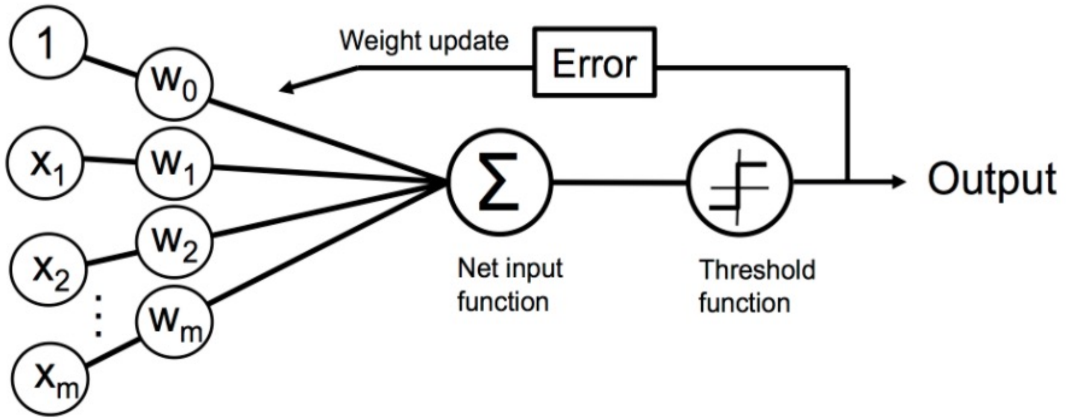
Python Machine Learning; Raschka & Mirjalili

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

Neural Network



- How does this relate to a perceptron?

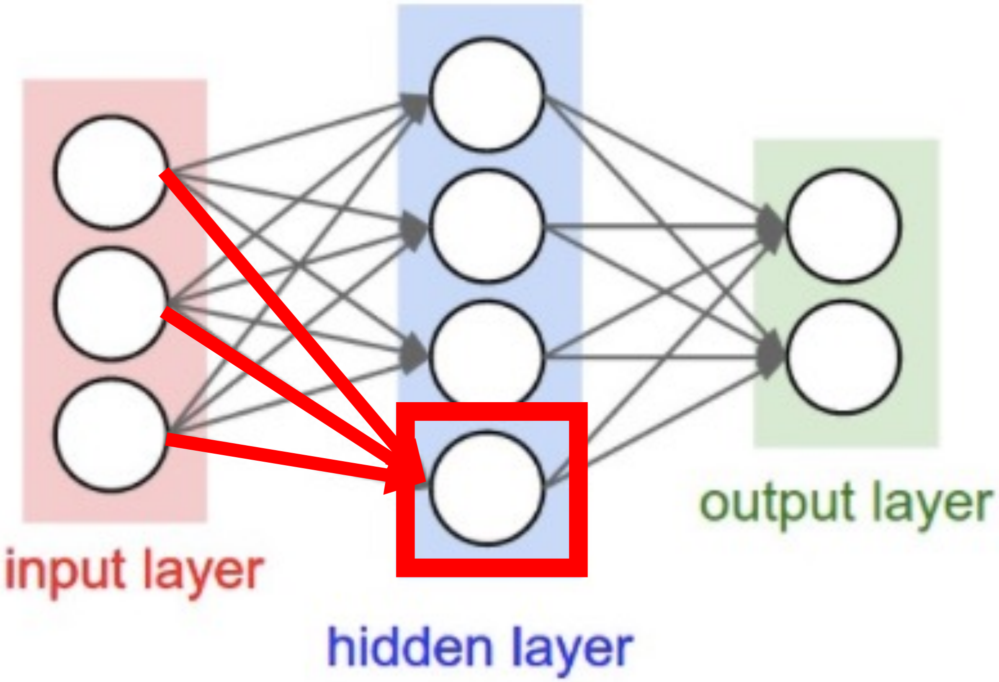


- Unit: takes as input a weighted sum and applies a function to the input

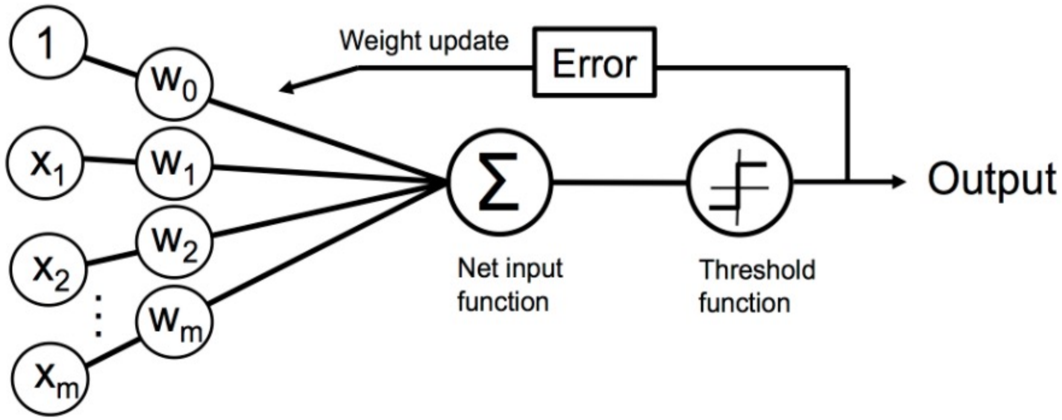
Python Machine Learning; Raschka & Mirjalili

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

Neural Network



- How does this relate to a perceptron?

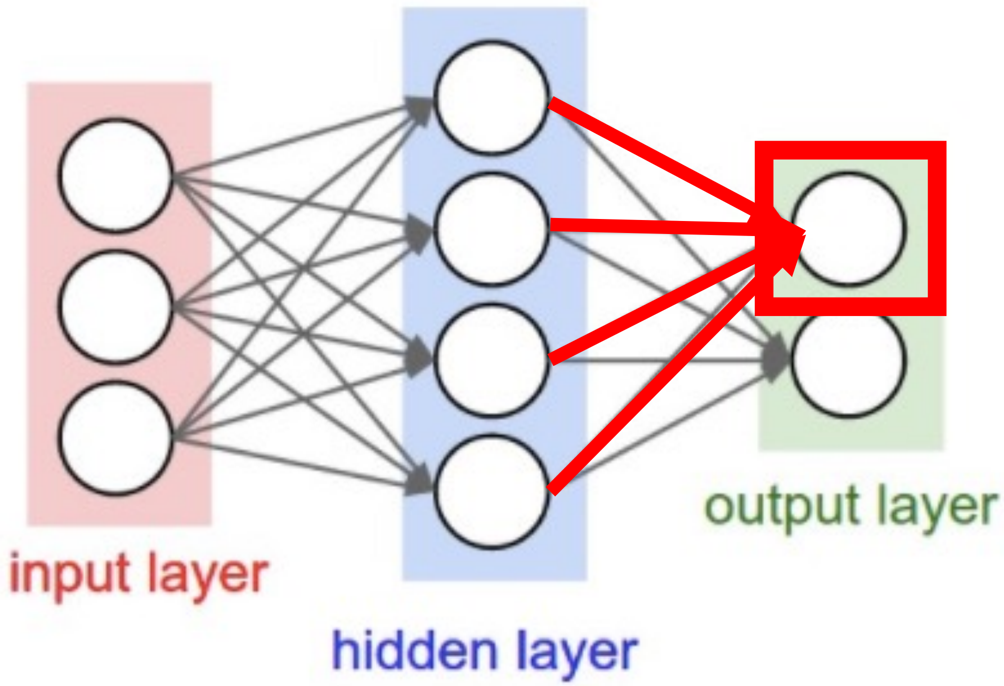


- Unit: takes as input a weighted sum and applies a function to the input

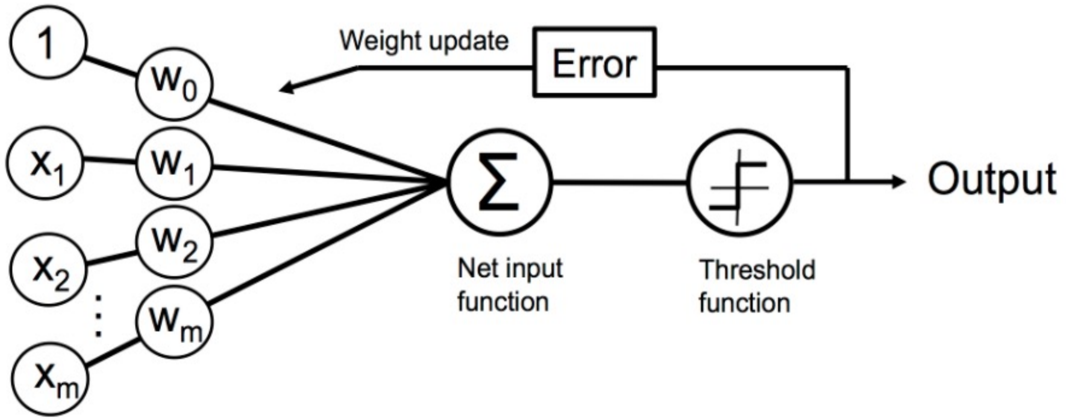
Python Machine Learning; Raschka & Mirjalili

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

Neural Network



- How does this relate to a perceptron?

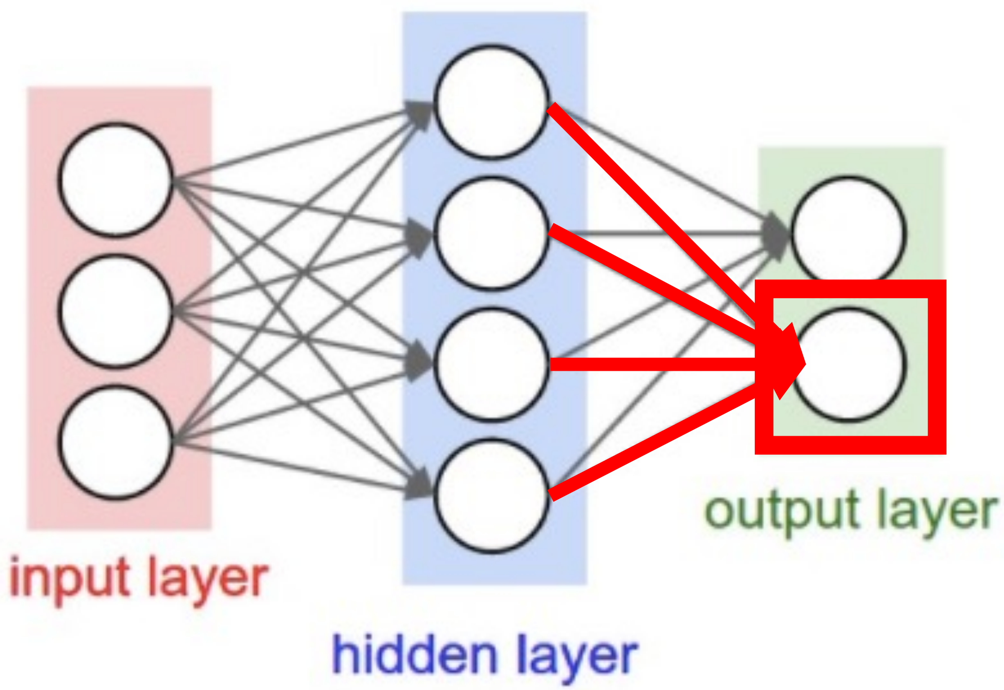


- Unit: takes as input a weighted sum and applies a function to the input

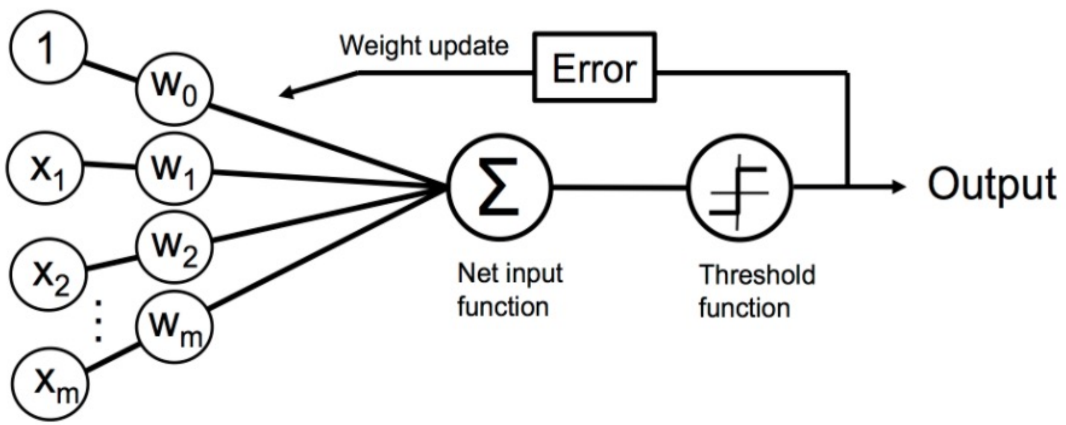
Python Machine Learning; Raschka & Mirjalili

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

Neural Network



- How does this relate to a perceptron?

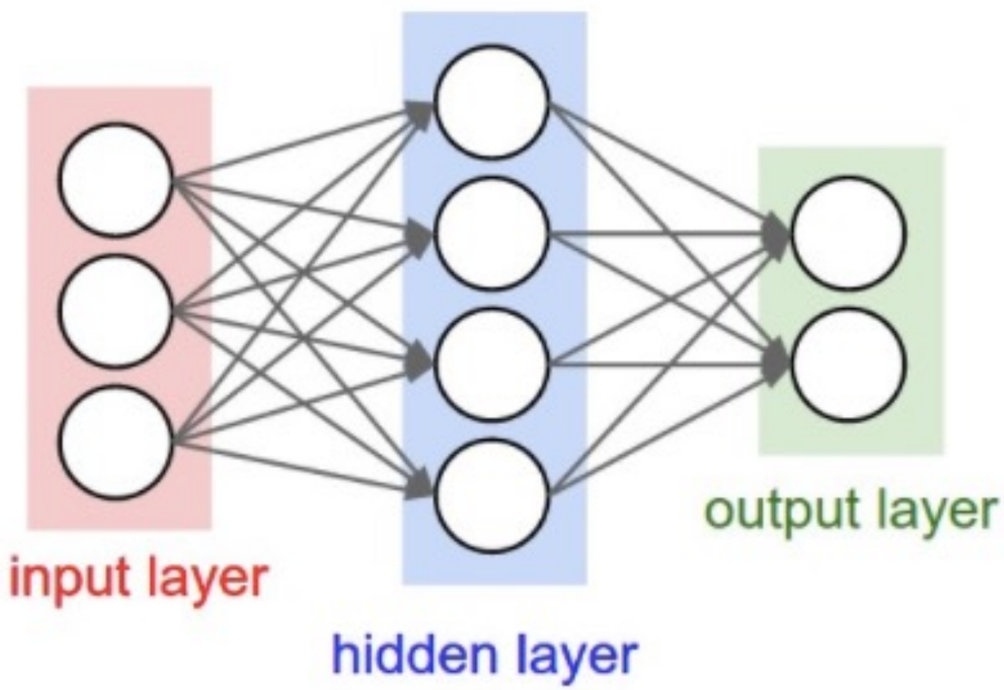


- Unit: takes as input a weighted sum and applies a function to the input

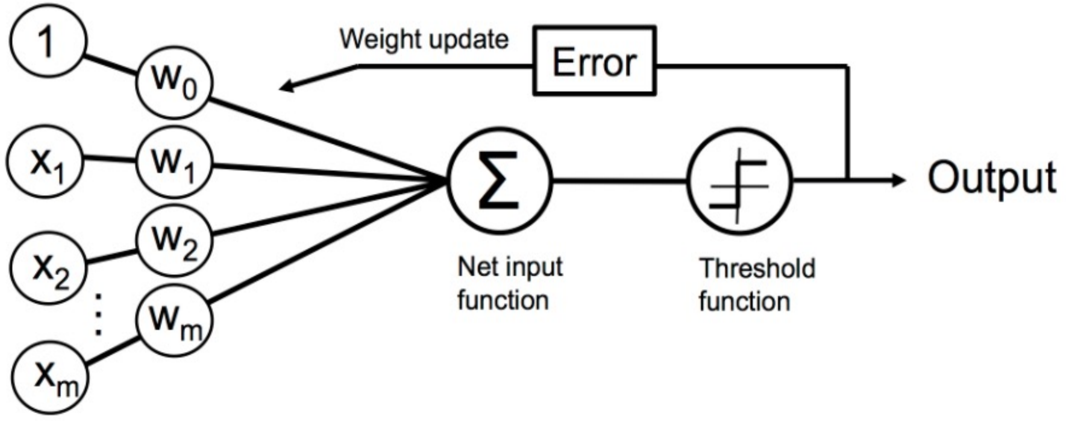
Python Machine Learning; Raschka & Mirjalili

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

Neural Network



- How does this relate to a perceptron?

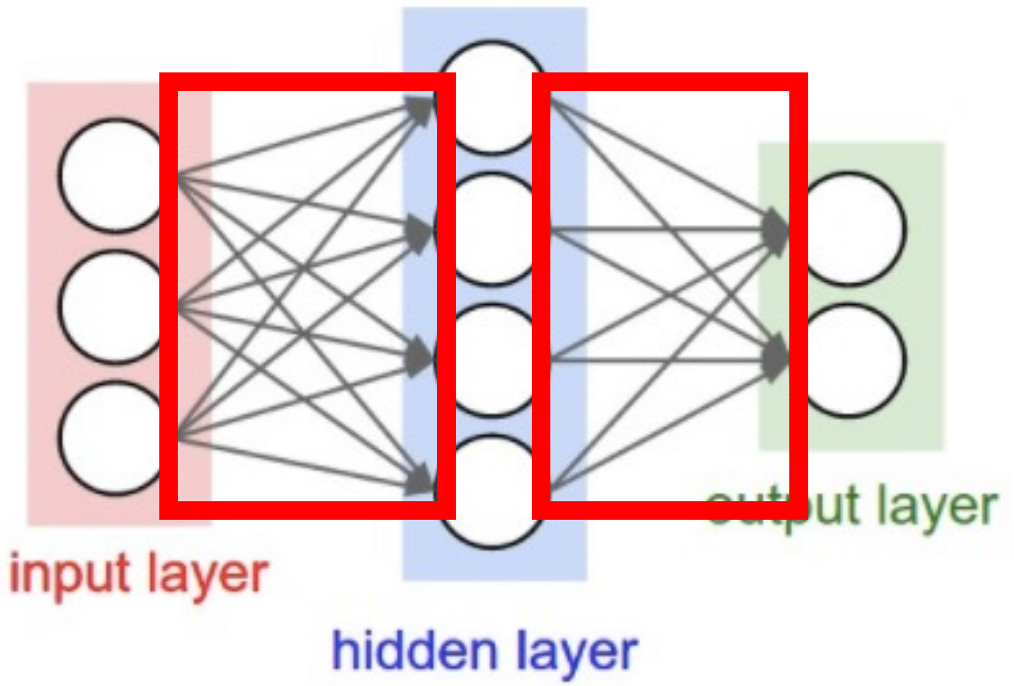


- **Training goal: learn model parameters**

Python Machine Learning; Raschka & Mirjalili

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

Neural Network

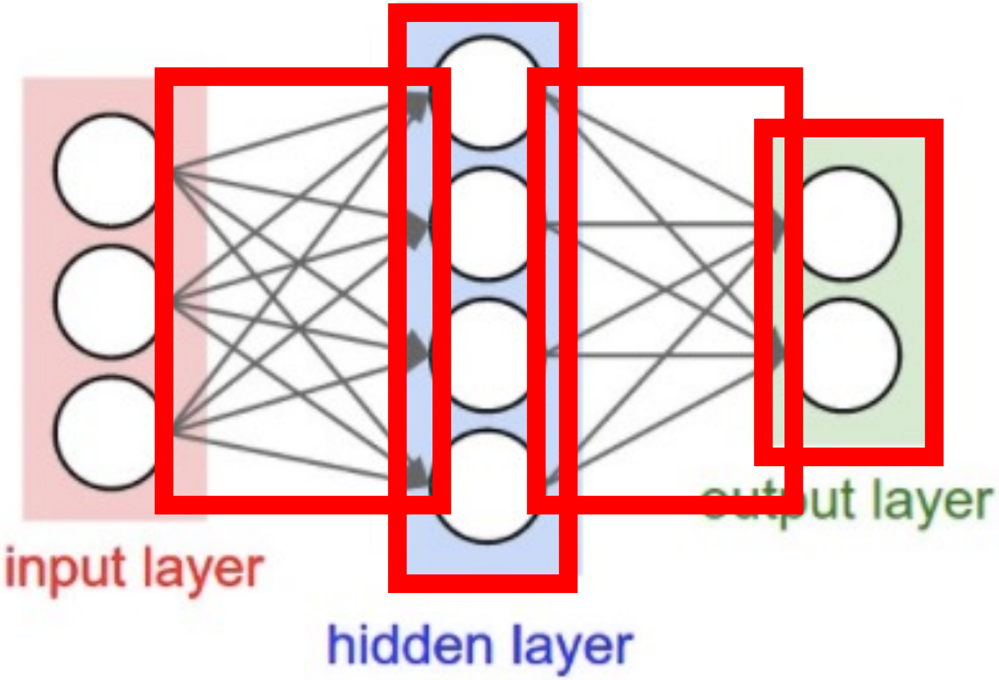


How many weights are in this model?

- 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

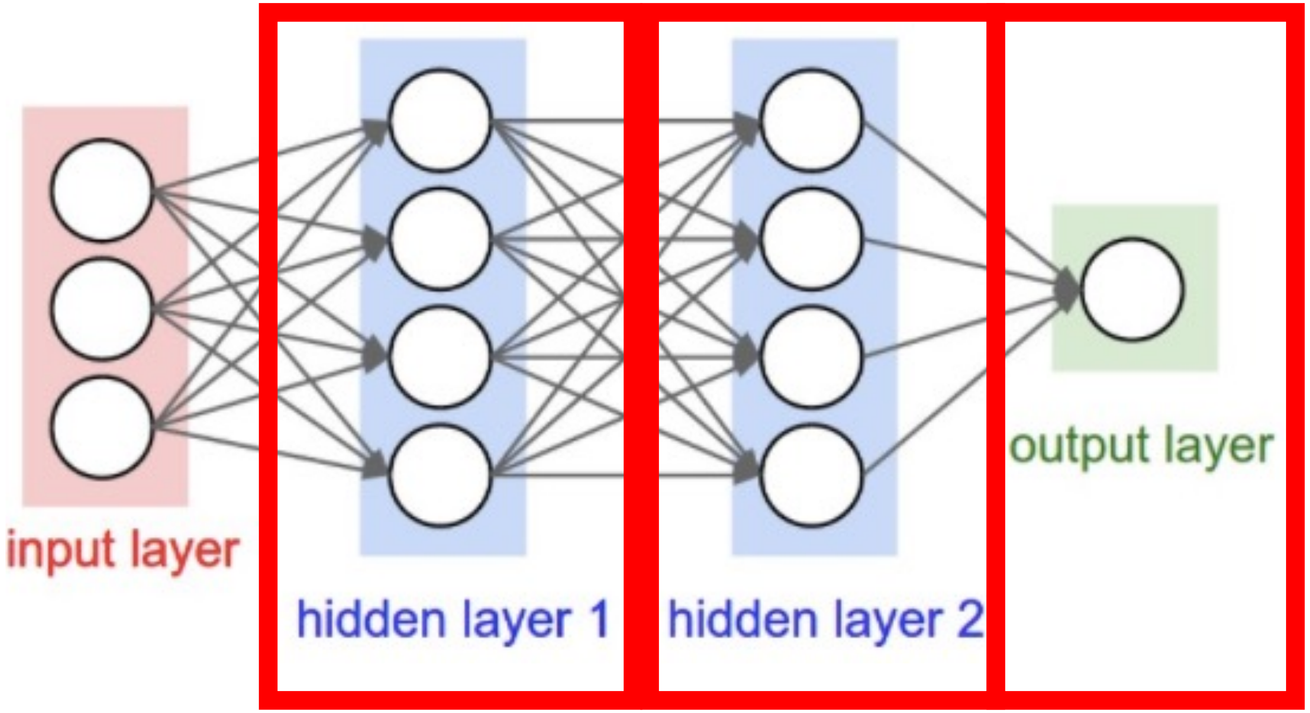


How many parameters are there to learn?

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

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

Neural Network

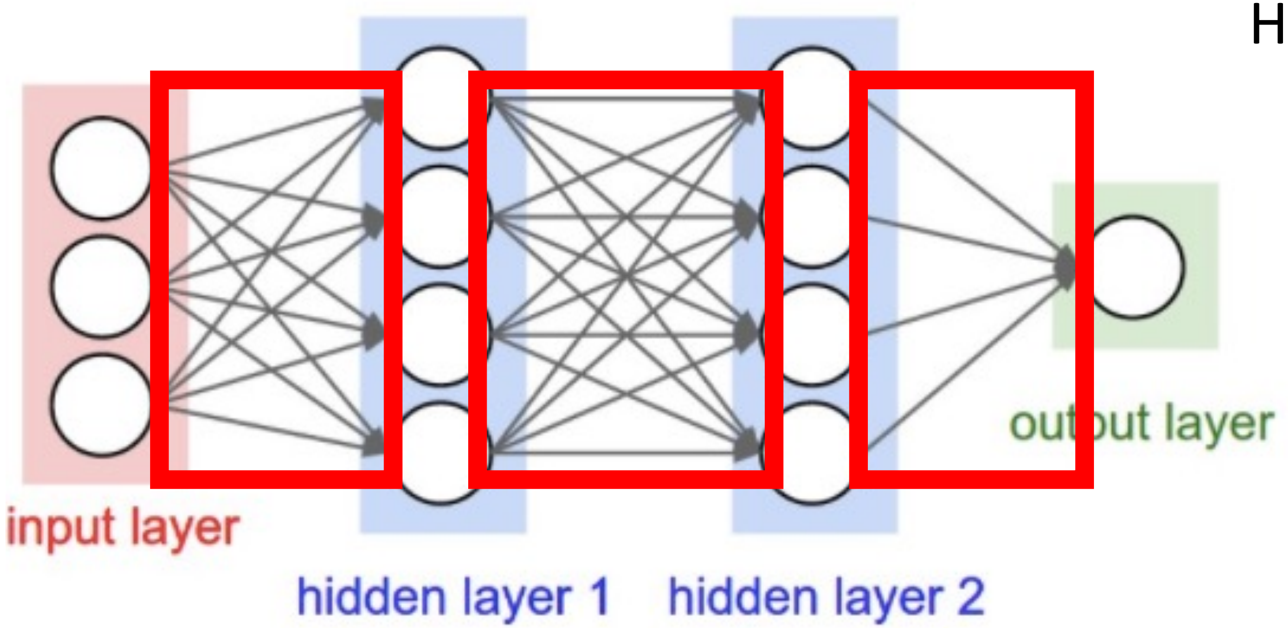


How many layers are in this network?

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

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

Neural Network

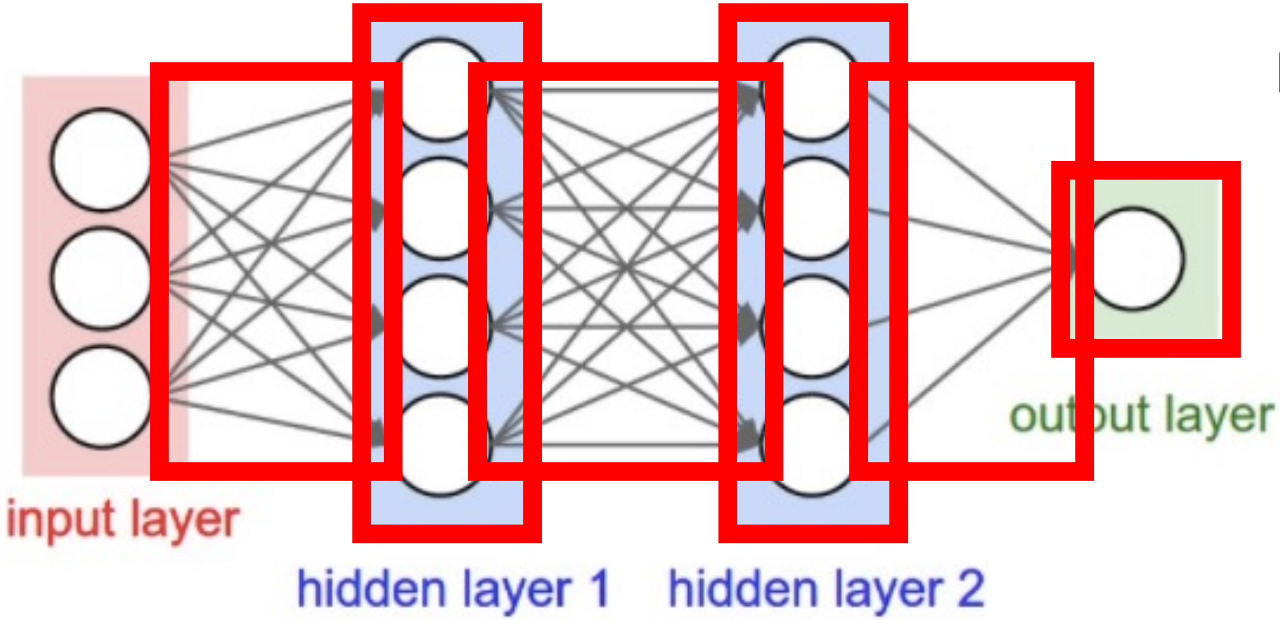


How many weights are in this model?

- 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?

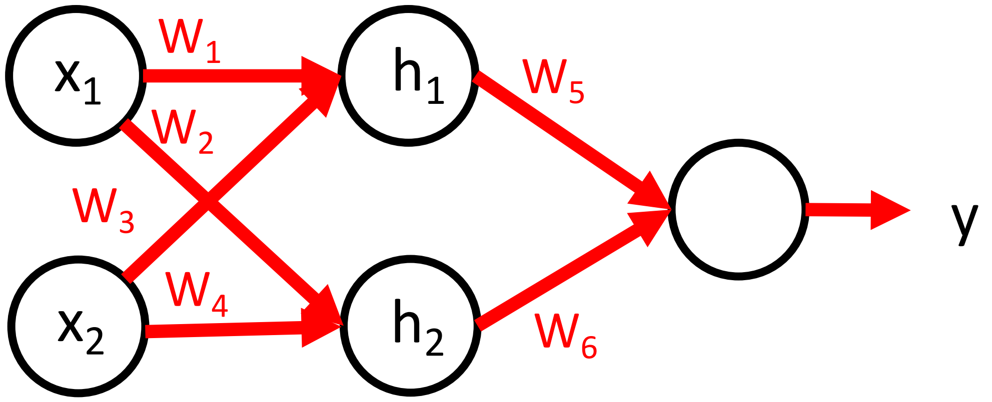
- Number of weights:
  - 32
- Number of biases:
  - 4 + 4 + 1 = 9
- Total
  - 41

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.,



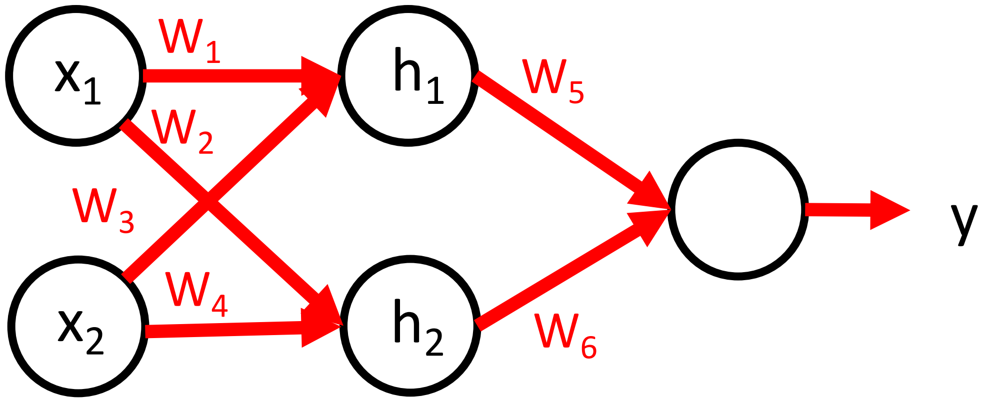
- What is function for h_1?
  - $h_1 = w_1x_1 + w_3x_2 + b_1$
- What is function for h_2?
  - $h_2 = w_2x_1 + w_4x_2 + b_2$
- What is function for y?
  - $y = h_1w_5 + h_2w_6 + b_3$
  - $y = (w_1x_1 + w_3x_2 + b_1)w_5 + (w_2x_1 + w_4x_2 + b_2)w_6 + b_3$
  - $\boxed{y = w_1w_5x_1 + w_3w_5x_2 + w_5b_1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

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.,



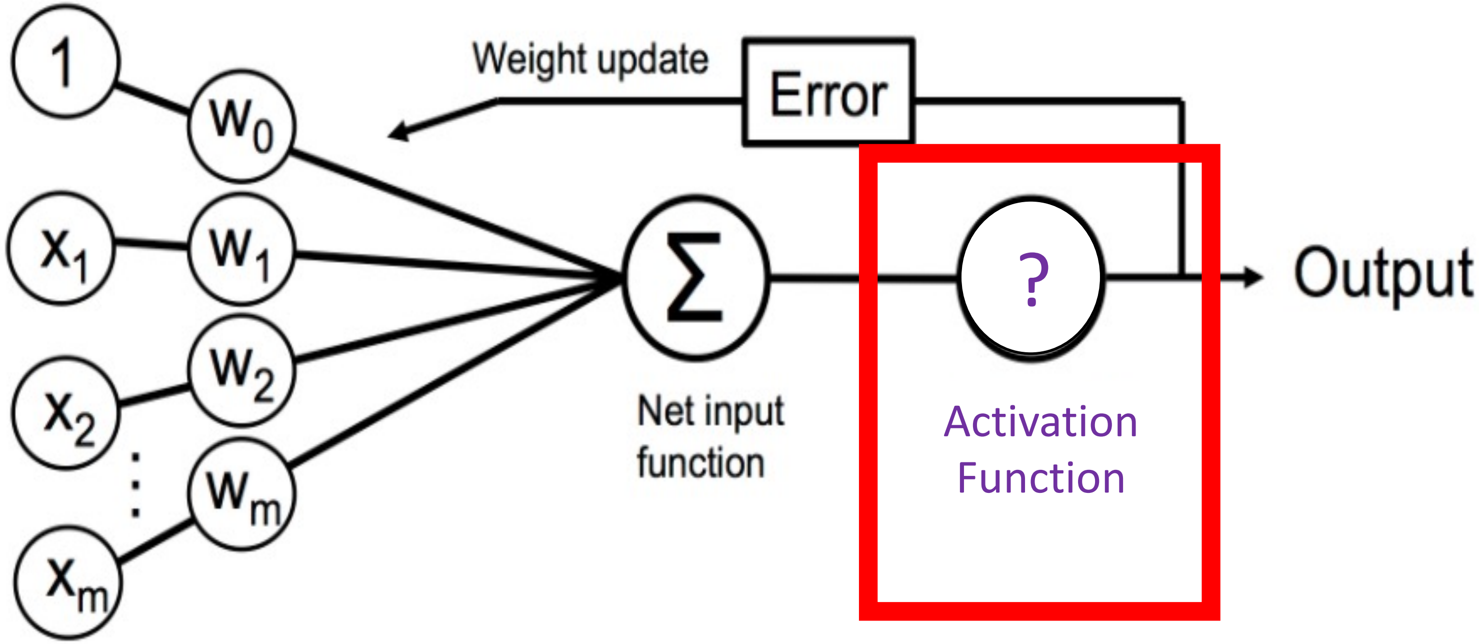
- What is function for h_1?
  - $h_1 = w_1x_1 + w_3x_2 + b_1$
- What is function for h_2?
  - $h_2 = w_2x_1 + w_4x_2 + b_2$
- What is function for y?
  - $y = \underbrace{h_1w_5} + \underbrace{h_2w_6} + b_3$

Constant x linear function = linear function

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

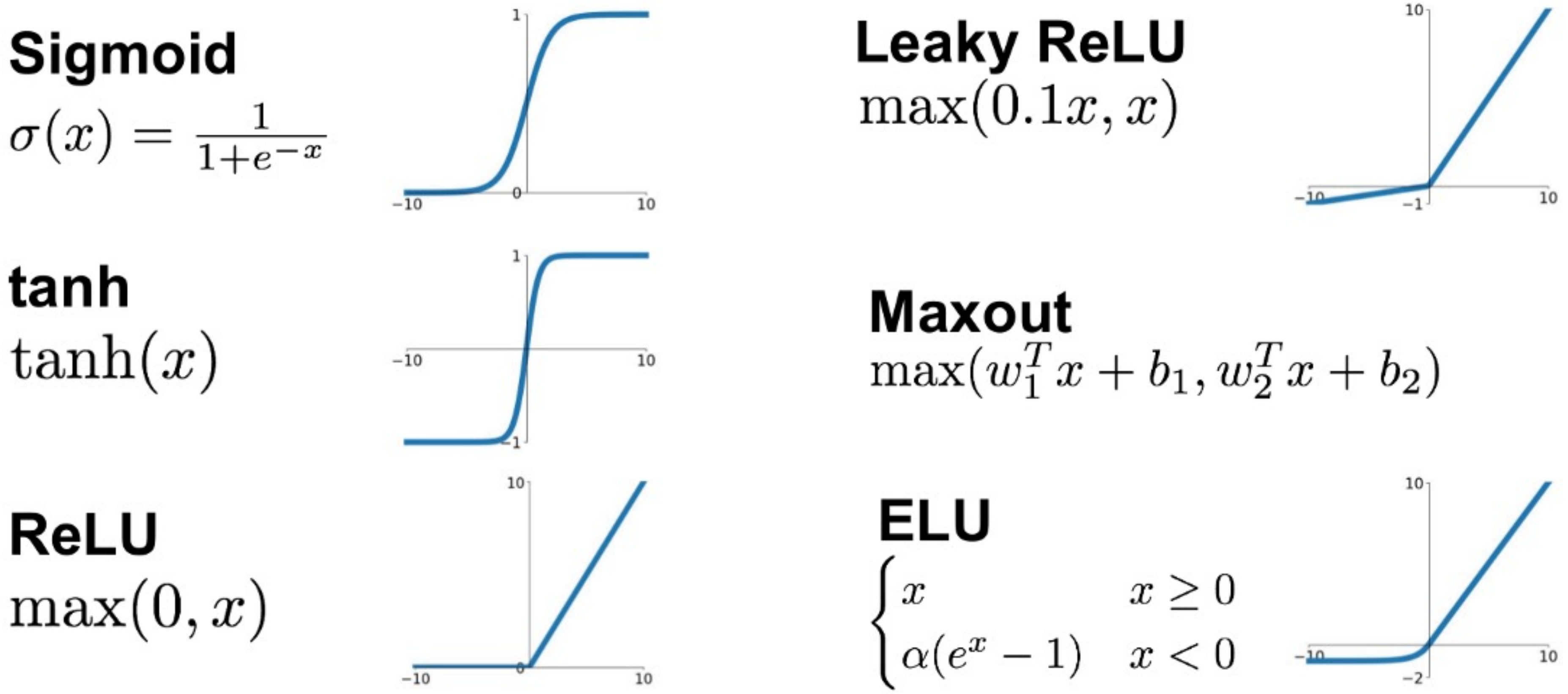
Solution to Model Non-Linear Functions: Non-Linear Activation Functions

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



Python Machine Learning; Raschka & Mirjalili

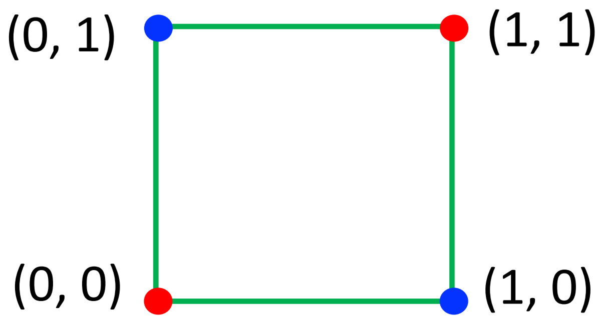
Solution to Model Non-Linear Functions: Non-Linear Activation Functions

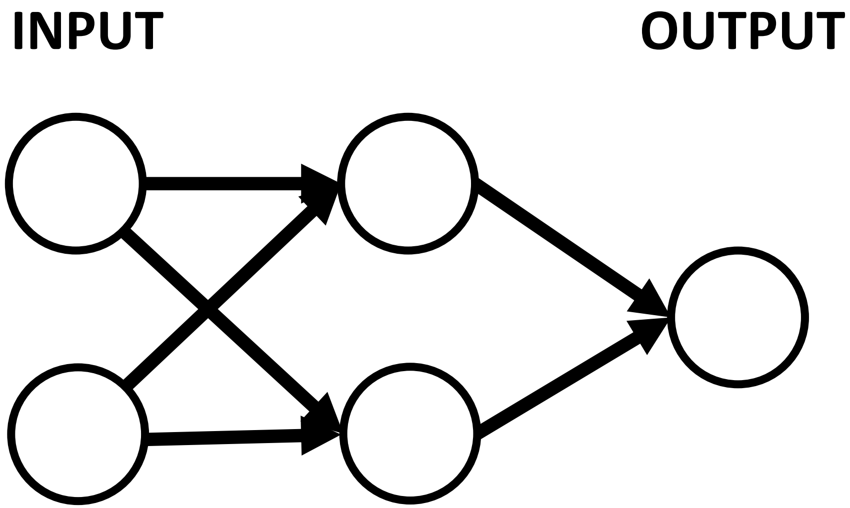


Source: https://www.linkedin.com/pulse/activation-functions-neural-networks-leonardo-calderon-j-/

Non-Linear Example: Revisiting XOR problem

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

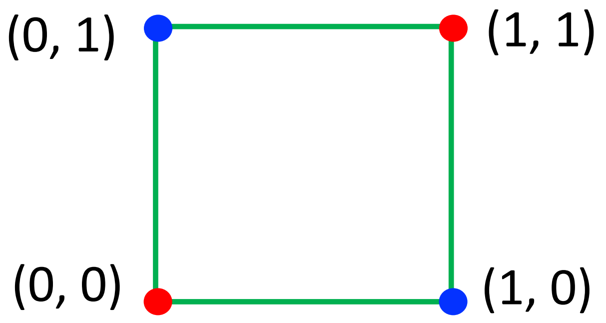




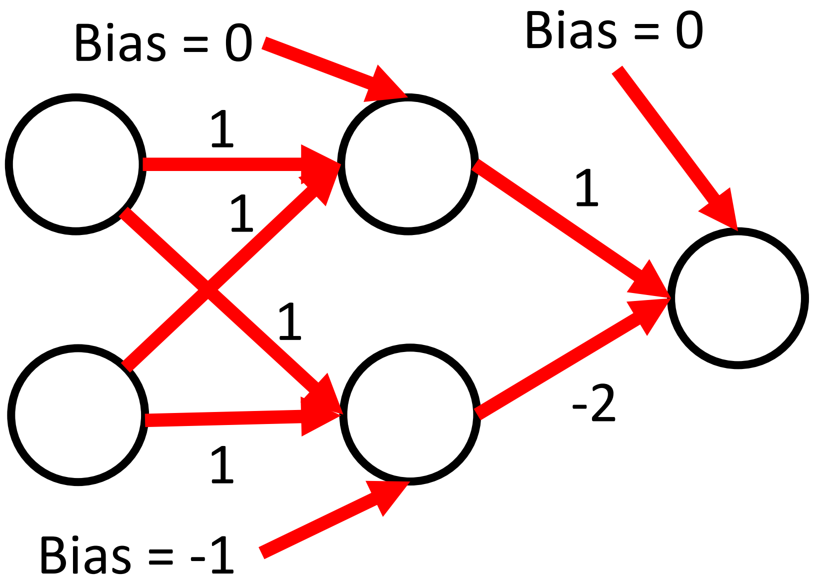
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



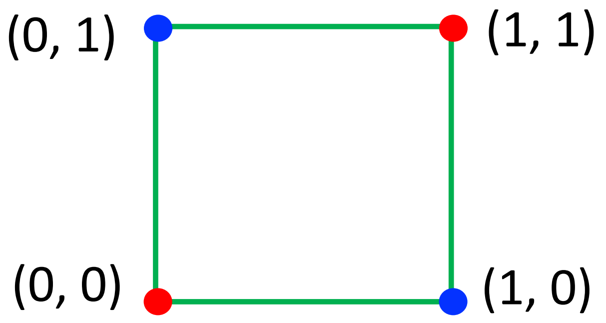
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



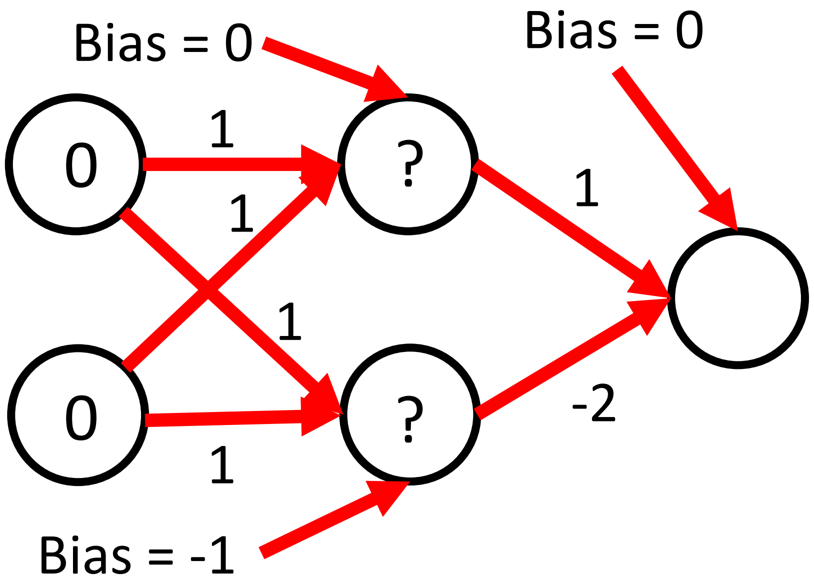
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



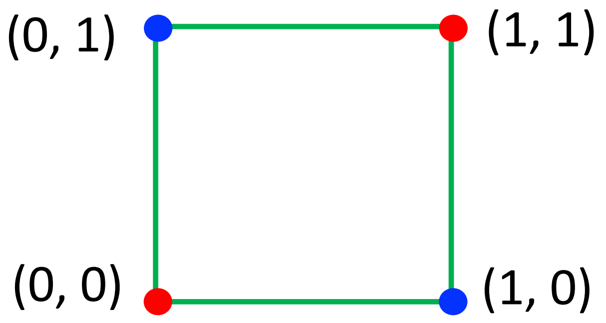
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



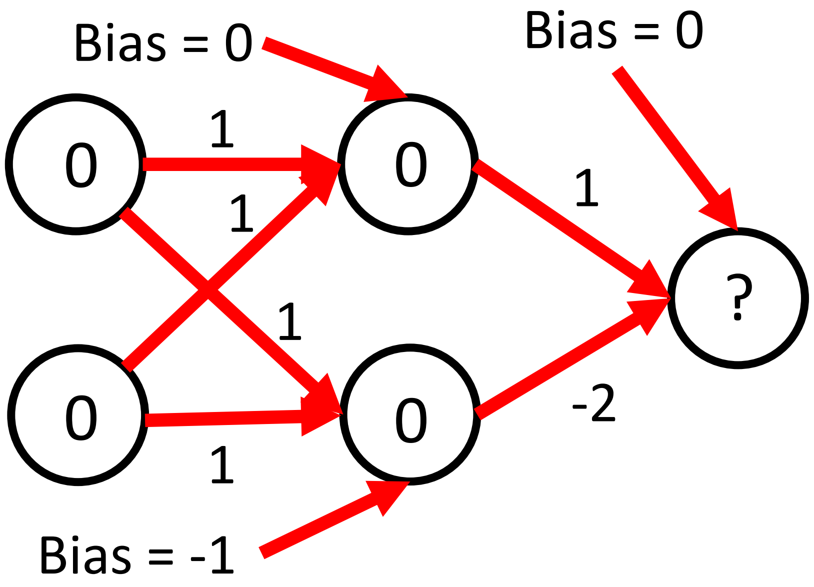
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



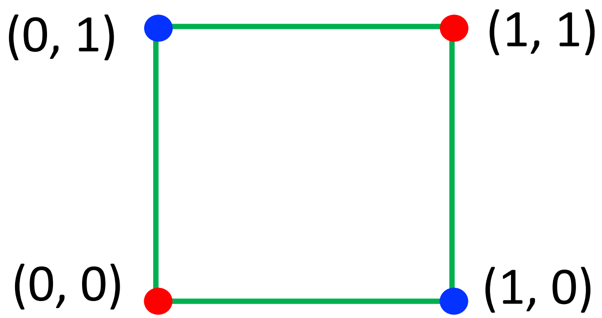
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



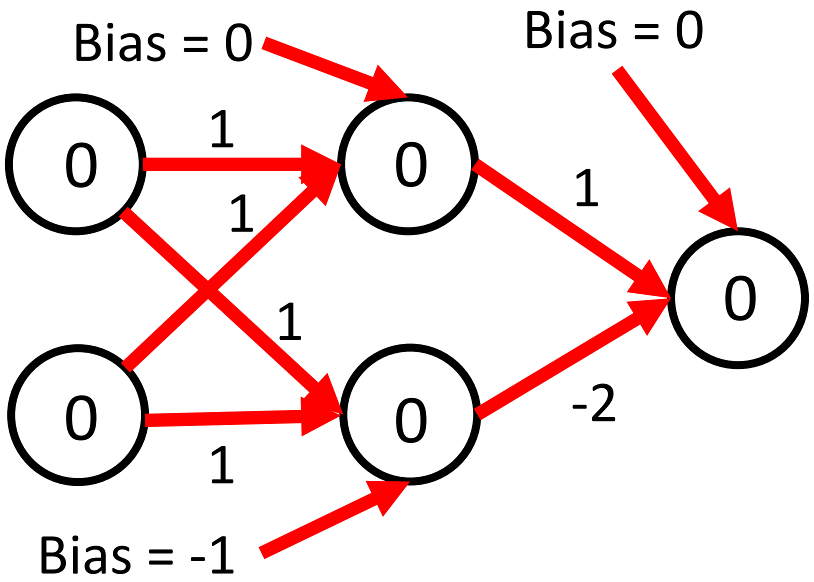
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



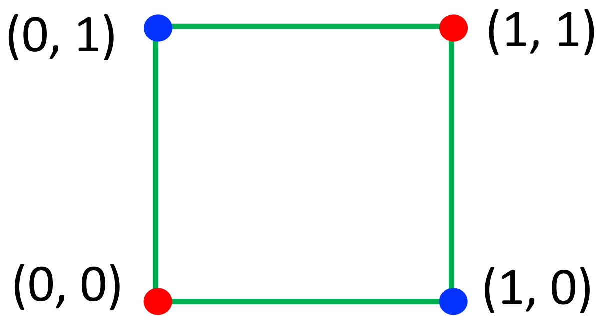
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



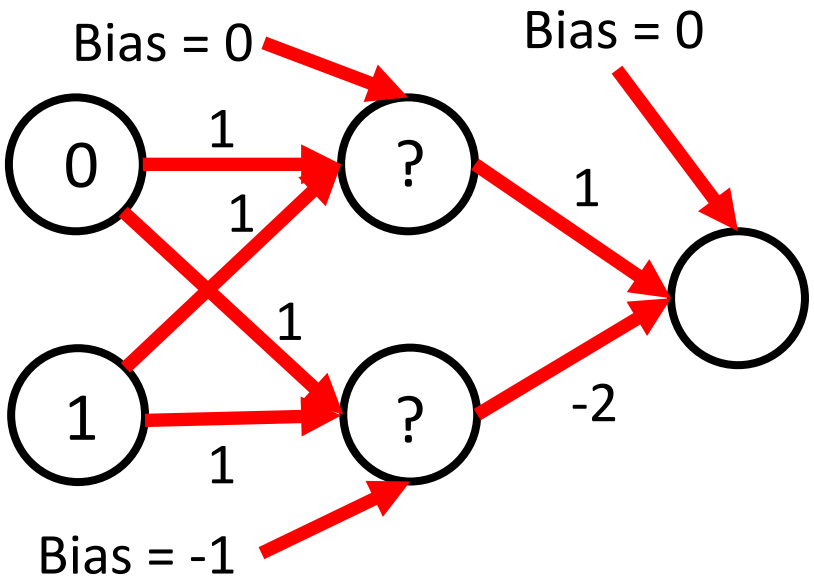
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



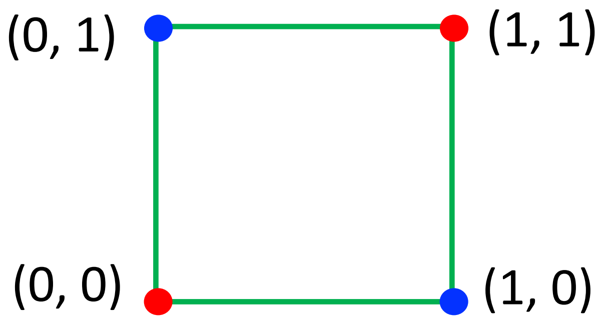
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



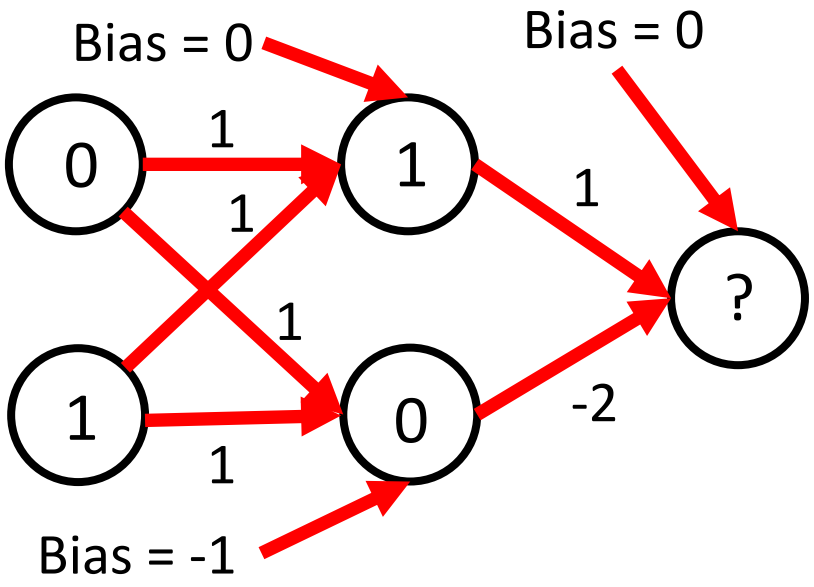
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



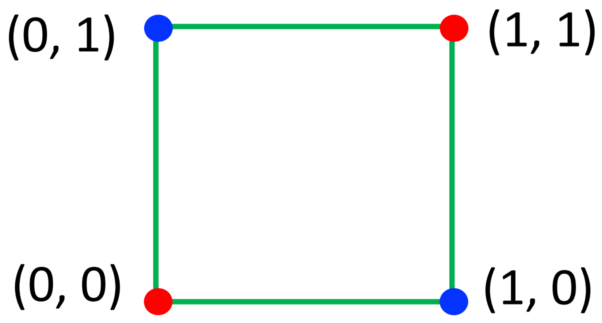
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



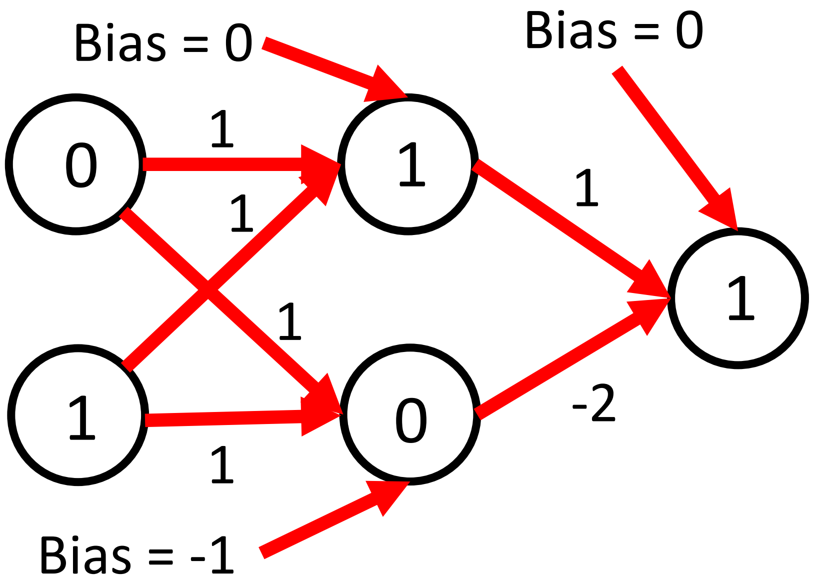
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



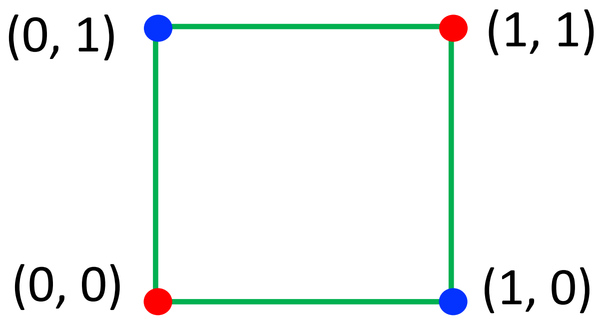
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



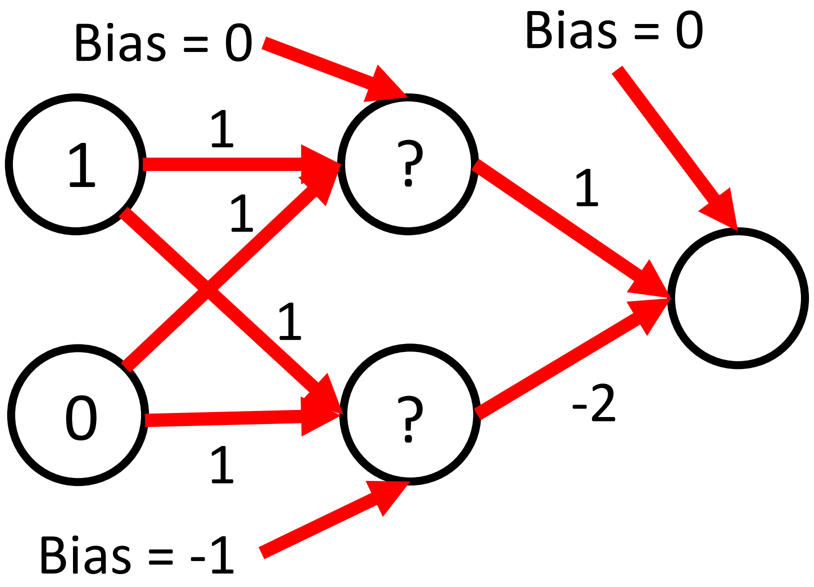
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



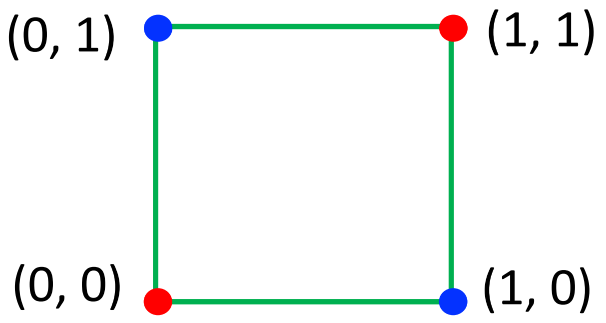
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



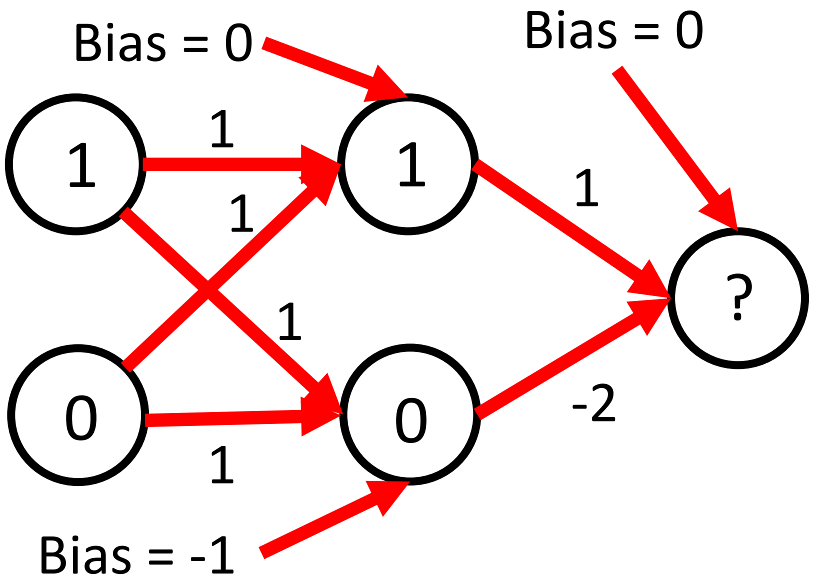
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



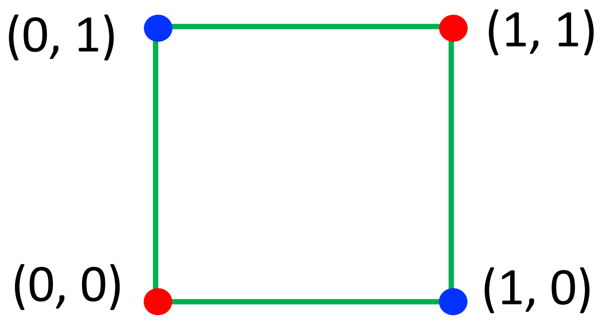
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



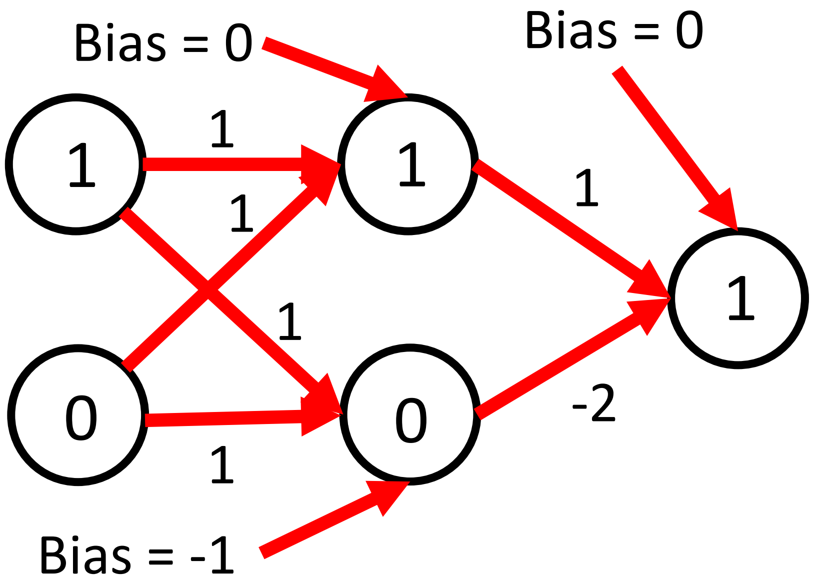
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



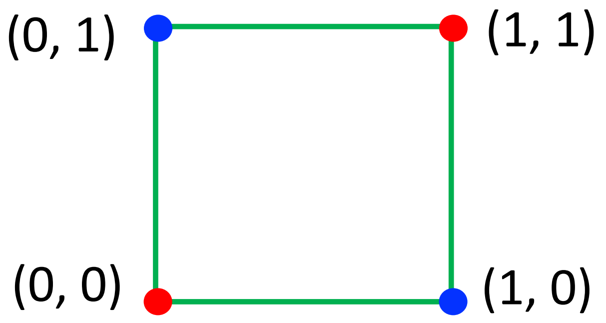
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



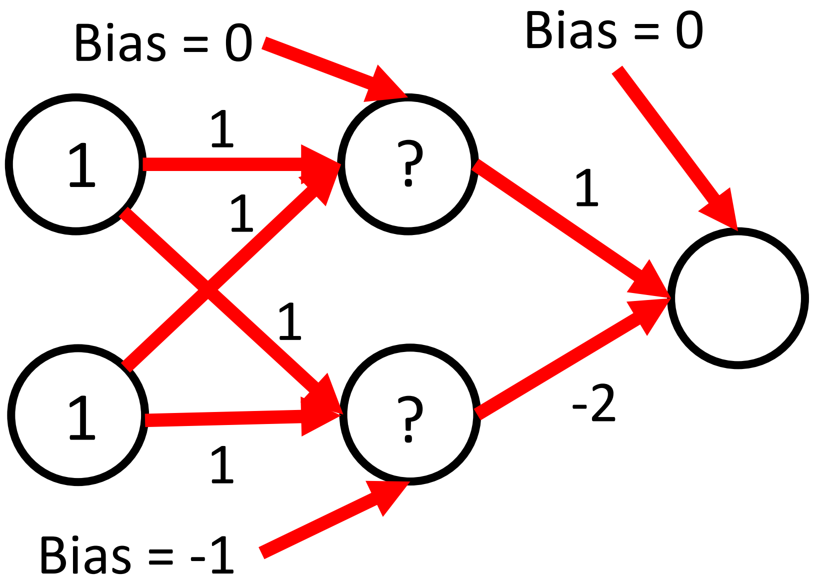
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



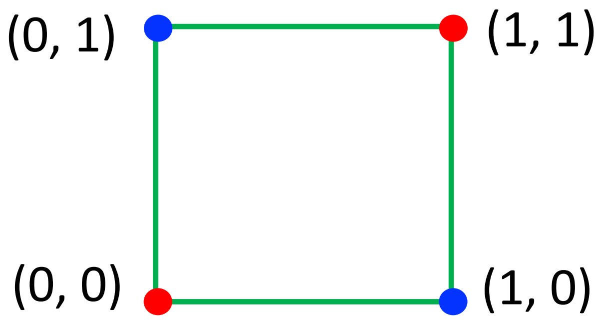
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



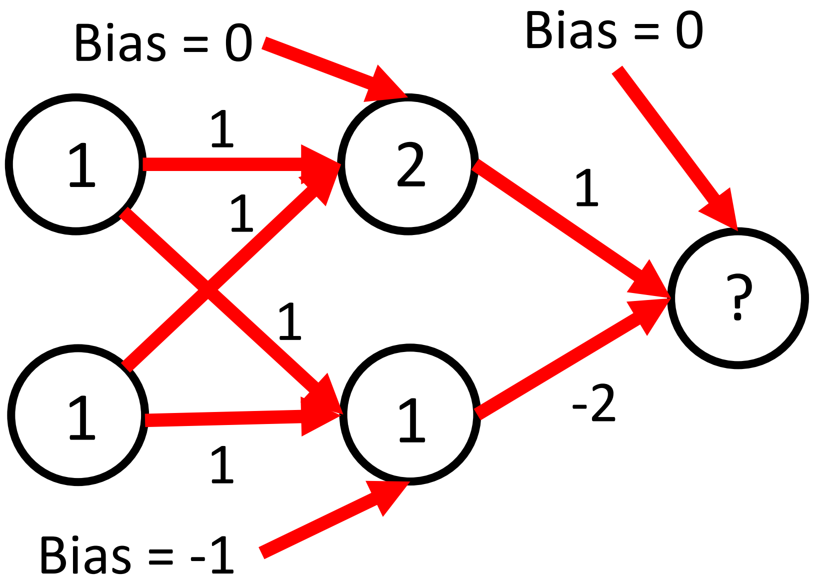
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
$\boxed{1}$	$\boxed{1}$	0

Non-Linear Example: Revisiting XOR problem

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



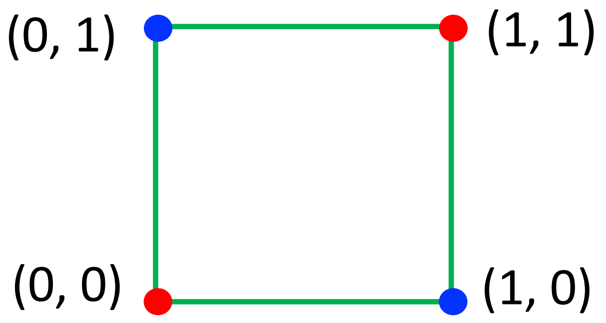
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



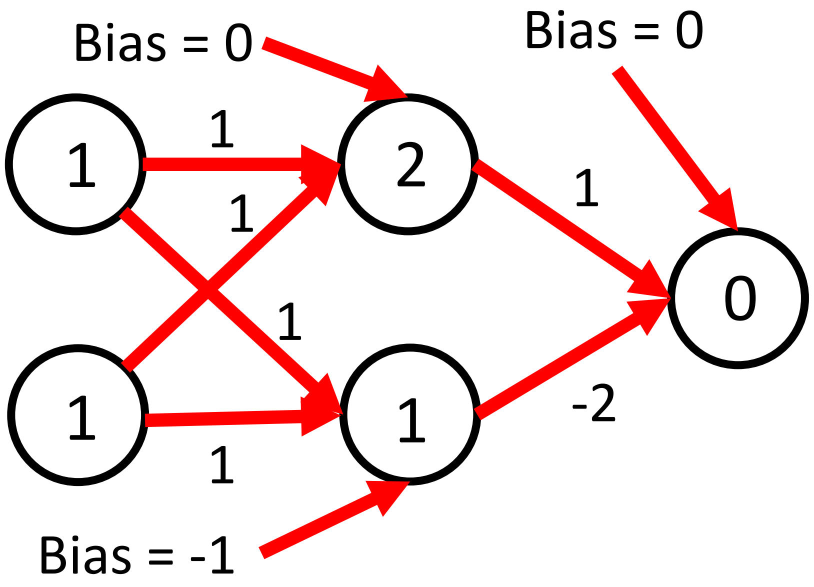
INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Non-Linear Example: Revisiting XOR problem

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



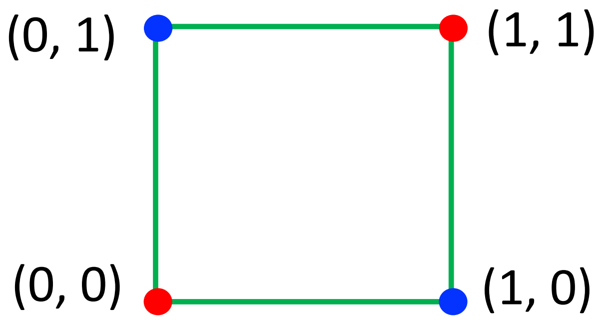
- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:



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

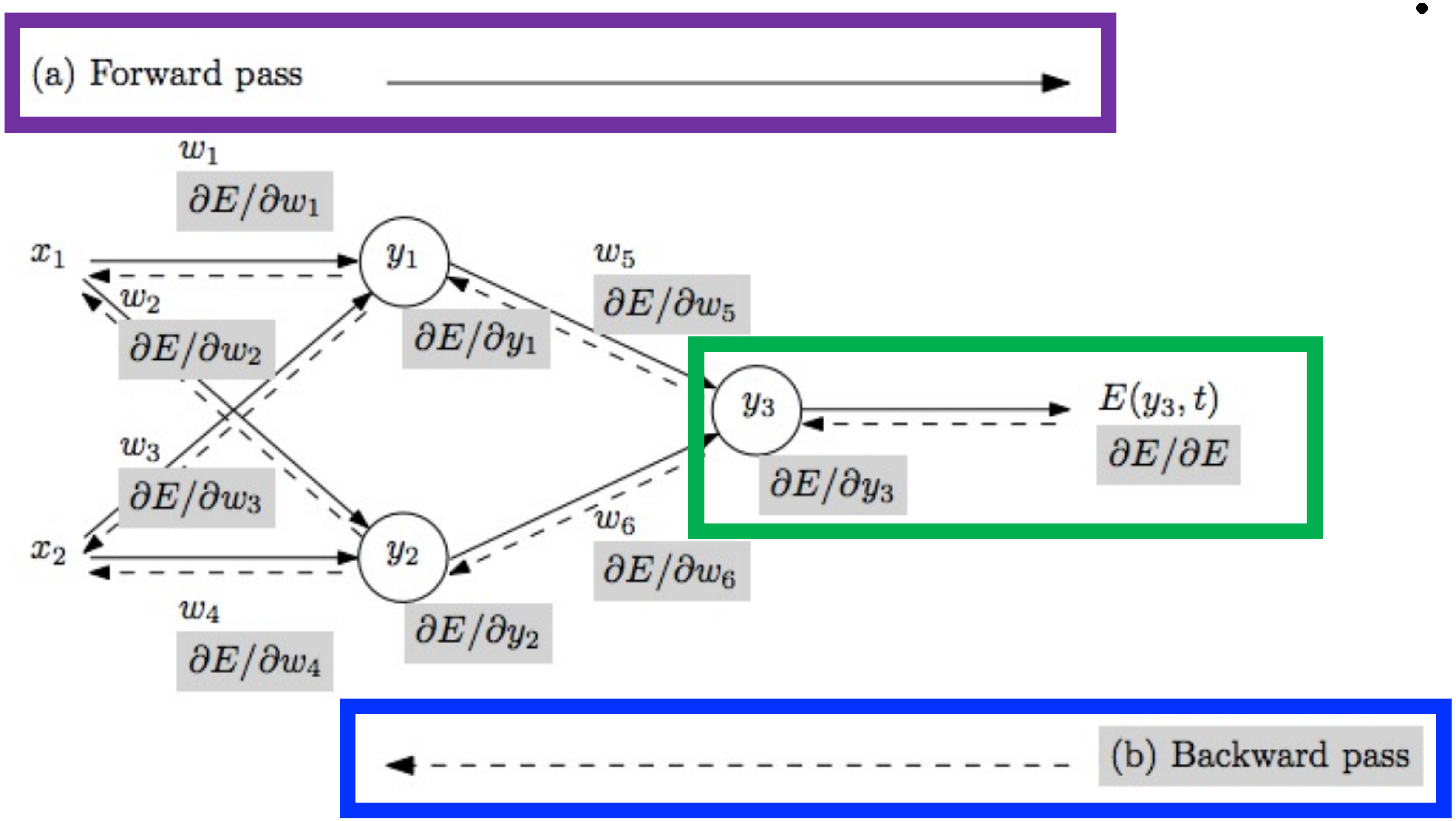
- Non-linear function: separate 1s from 0s:



- 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!

How Neural Networks Learn

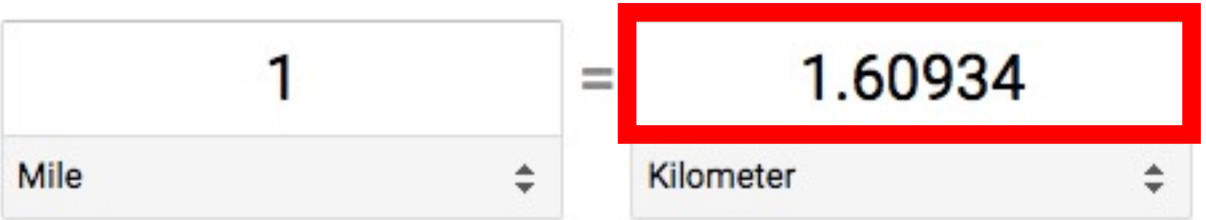


- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model 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

How Neural Networks Learn: 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

How Neural Networks Learn: Intuition

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

\$10 → Shekels = dollars x constant

How Neural Networks Learn: 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

How Neural Networks Learn: Intuition

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

$10 → Shekels = dollars x constant

How Neural Networks Learn: 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

How Neural Networks Learn: Intuition

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

$10 ⟶ Shekels = dollars x constant

How Neural Networks Learn: 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

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

How Neural Networks Learn: Gradient Descent

- Approach: solve mathematical problems by updating estimates of the solution via an iterative process to "optimize" a function
  - e.g., minimize or maximize an objective function f(x) by altering x

Analogy
Hiking to the bottom of a mountain range… blindfolded (or for a person who is blind)!

- When **minimizing** the objective function, it also is often called interchangeably the **cost function**, **loss function**, or **error function**.

Today’s Topics

- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Background of machine learning and neural networks
- Training deep neural networks: hardware & software

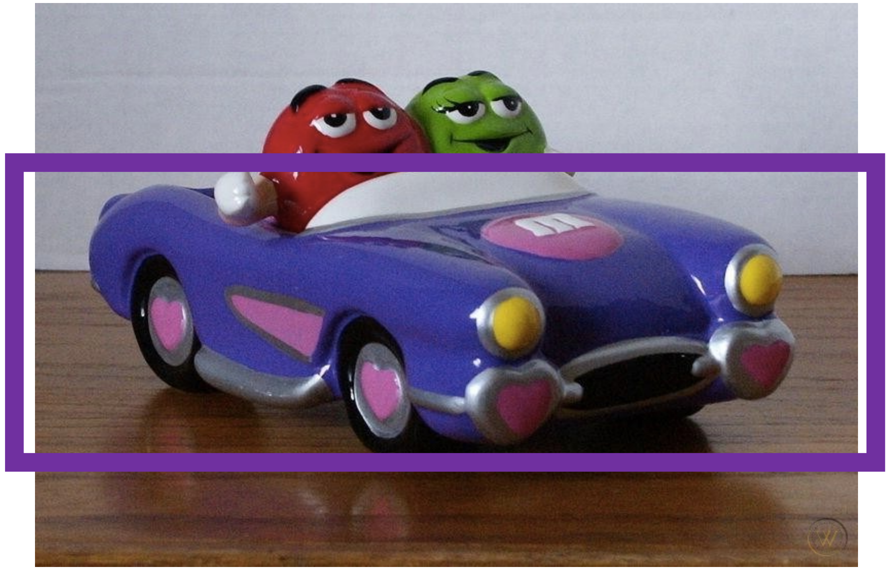
Neural Networks: Key Ingredients for Success

An **algorithm** learns from **data**
on a **processor** the patterns that
will be used to make a prediction

Analogous to a Love Story of Partnering Up and Road Tripping Somewhere

Key Challenge: How Long Does Learning Take?

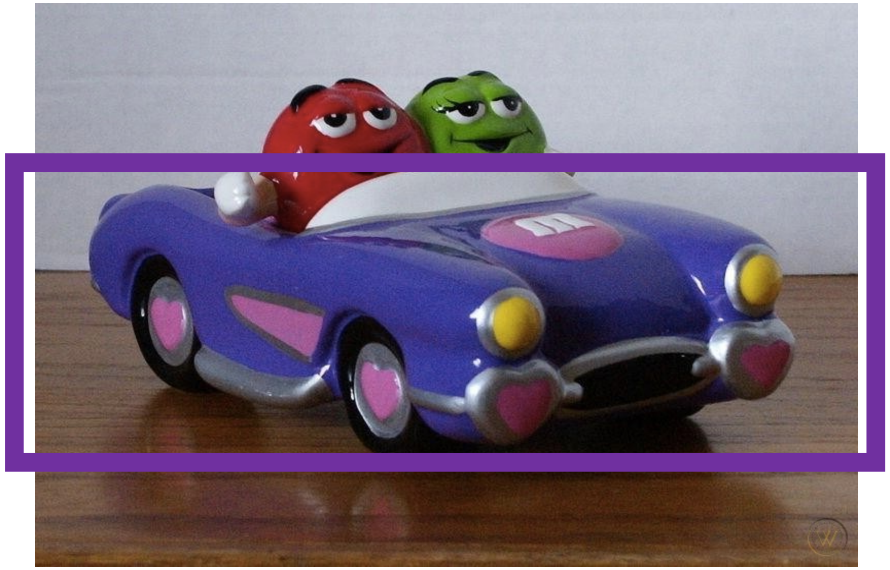
An **algorithm** learns from **data**
on a **processor** the patterns that
will be used to make a prediction



Analogous to a Love Story of Partnering Up and Road Tripping Somewhere

Key Challenge: How Long Does Learning Take?

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



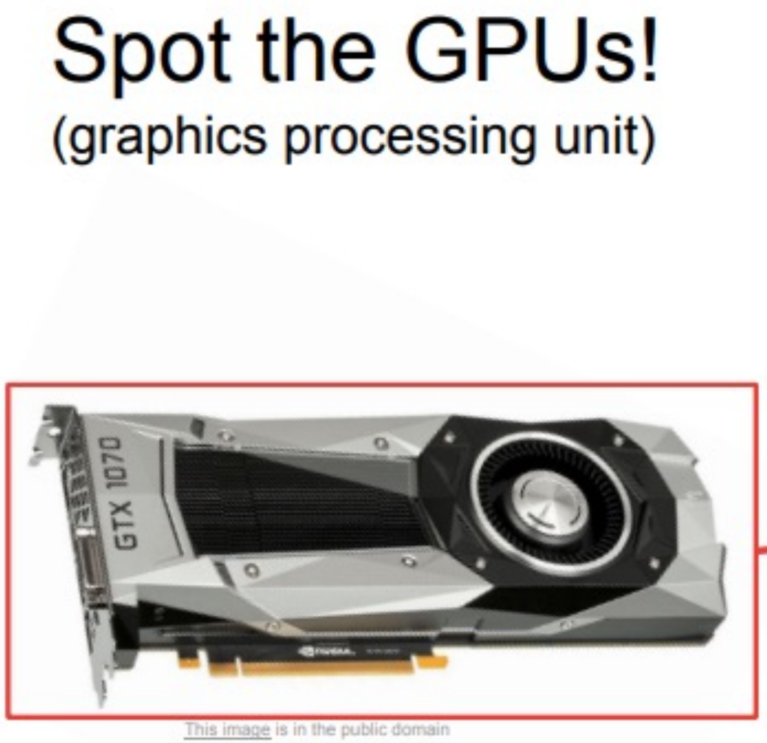
Hardware: CPU versus GPU

Spot the CPU!

(central processing unit)

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

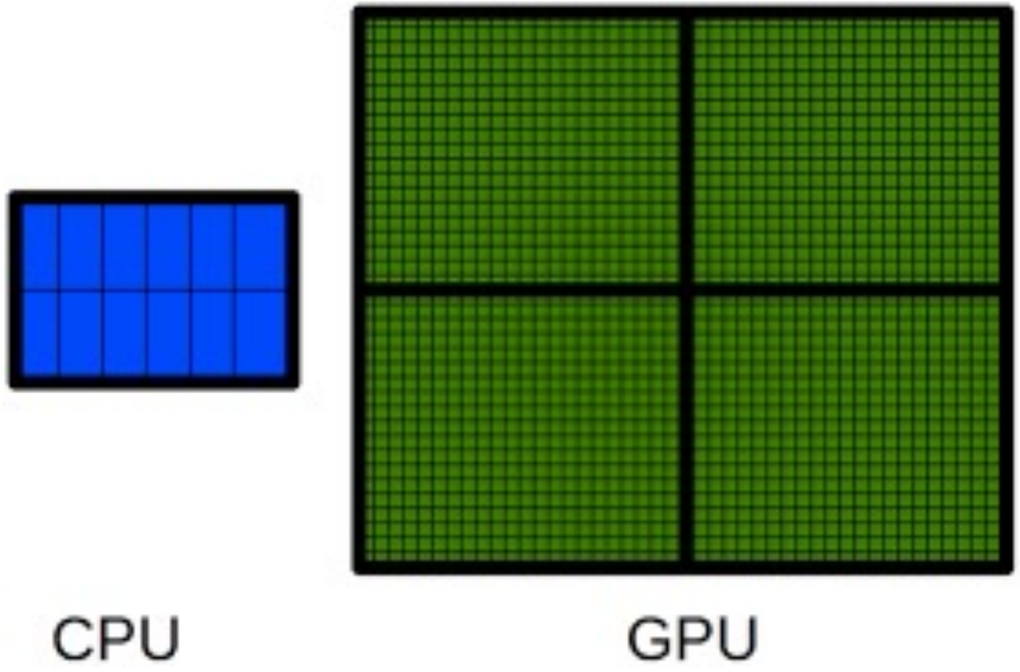
Hardware: CPU versus GPU



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

Hardware: Training Models with GPUs

Model is here

Data is here

If you aren't careful, training can bottleneck on reading data and transferring to GPU!

Solutions:

- Read all data into RAM
- Use SSD instead of HDD
- Use multiple CPU threads to prefetch data

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

(TPU = Tensor Processing Unit)

Hardware: CPU versus GPU

	Cores	Clock Speed	Memory	Price	Speed
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	System RAM	$339	~540 GFLOPs FP32
GPU (NVIDIA GTX 1080 Ti)	3584	1.6 GHz	11 GB GDDR5 X	$699	~11.4 TFLOPs FP32
TPU NVIDIA TITAN V	5120 CUDA, 640 Tensor	1.5 GHz	12GB HBM2	$2999	~14 TFLOPs FP32 ~112 TFLOP FP16
TPU Google Cloud TPU	?	?	64 GB HBM	$6.50 per hour	~180 TFLOP

CPU: Fewer cores, but each core is much faster and much more capable; great at sequential tasks

GPU: More cores, but each core is much slower and "dumber"; great for parallel tasks

TPU: Specialized hardware for deep learning

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

GPU Clusters (Google Cloud’s TPU Servers)

https://www.extremetech.com/extreme/249499-google-takes-swipe-nvidia-powerful-new-learning-capable-cloud-tpu

GPU Machines: Rent Versus Buy?

Rent from Cloud (Microsoft Azure):

ND6

6 vCPU 112 GiB RAM 1X P40 GPU

STARTING FROM

$1,511.10

/per month

POWERED BY

NVIDIA

+ Add to estimate

Buy:

Basic

2x RTX 2080 Ti

2-Way NVLink

Intel i9-9820X (10 cores, 3.30 GHz)

2x RTX 2080 Ti (11 GB VRAM)

64 GB RAM

2 TB SSD

4 TB HDD

Starting at

$7,059

Customize

Rise of “Deep Learning” Open Source Platforms

Motivation:

Can run on GPUs:

OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]

Simplifies using popular neural network architectures:

Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)

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

Rise of "Deep Learning" Open Source Platforms

torch (Collobert et al.) — 2011

theano (Bastien et al.) — 2012

Caffe (Jia et al.) — 2013

2014

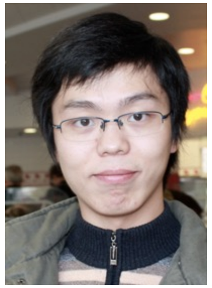
TensorFlow™ (Abadi et al.) — 2015

2016

PYTORCH (Paszke et al.) — 2017

idiap RESEARCH INSTITUTE

Université de Montréal



Berkeley UNIVERSITY OF CALIFORNIA

Google

f

Popular Options Today

Rise of “Deep Learning” Open Source Platforms

Software	Creator	Software license[a]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)	Actively Developed
roNNie.ai	Kevin Lok	MIT license	Yes	Linux, macOS, Windows	Python	Python			Yes		Yes	Yes	Yes			
BigDL	Jason Dai	Apache 2.0	Yes	Apache Spark	Scala	Scala, Python			No		Yes	Yes	Yes			
Caffe	Berkeley Vision and Learning Center	BSD	Yes	Linux, macOS, Windows[2]	C++	Python, MATLAB, C++	Yes	Under development[3]	Yes	Yes	Yes[4]	Yes	Yes	No	?	
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python (Keras), Kotlin	Yes	On roadmap[5]	Yes[6][7]	Computational Graph	Yes[8]	Yes	Yes	Yes	Yes[9]	
Chainer	Preferred Networks	MIT license	Yes	Linux, macOS, Windows		Python	No	No[10][11]	Yes	Yes	Yes	Yes	Yes			
Darknet	Joseph Redmon	Public Domain	Yes	Cross-Platform	C	C, Python	Yes	No[12]	Yes	Yes						
Dlib	Davis King	Boost Software License	Yes	Cross-Platform	C++	C++	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	
DataMelt (DMelt)	S.Chekanov	Freemium	Yes	Cross-Platform	Java	Java	No	No	No	No	No	No	No	No	No	
DyNet	Carnegie Mellon University	Apache 2.0	Yes	Linux, macOS, Windows		C++, Python		No[13]	Yes	Yes	Yes					
Intel Data Analytics Acceleration Library	Intel	Apache License 2.0	Yes	Linux, macOS, Windows on Intel CPU[14]	C++, Python, Java	C++, Python, Java[14]	Yes	No	No	Yes	No		Yes		Yes	
Intel Math Kernel Library	Intel	Proprietary	No	Linux, macOS, Windows on Intel CPU[15]		C[16]	Yes[17]	No	No	Yes	No	Yes[18]	Yes[18]		No	
Keras	François Chollet	MIT license	Yes	Linux, macOS, Windows	Python	Python, R	Only if using Theano as backend	Can use Theano or Tensorflow as backends	Yes	Yes	Yes[19]	Yes	Yes	Yes	Yes[20]	
MATLAB + Neural Network Toolbox	MathWorks	Proprietary	No	Linux, macOS, Windows	C, C++, Java, MATLAB	MATLAB	No	No	Train with Parallel Computing Toolbox and generate CUDA code with GPU Coder[21]	No	Yes[22][23]	Yes[22]	Yes[22]	No	With Parallel Computing Toolbox[24]	
Microsoft Cognitive Toolkit	Microsoft Research	MIT license[25]	Yes	Windows, Linux[26] (macOS via Docker on roadmap)	C++	Python (Keras), C++, Command line,[27] BrainScript[28] (.NET on roadmap[29])	Yes[30]	No	Yes	Yes	Yes[31]	Yes[32]	Yes[32]	No[33]	Yes[34]	
Apache MXNet	Apache Software Foundation	Apache 2.0	Yes	Linux, macOS, Windows,[35][36] AWS, Android,[37] iOS, JavaScript[38]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Yes	On roadmap[39]	Yes	Yes[40]	Yes[41]	Yes	Yes	Yes	Yes[42]	
Neural Designer	Artelnics	Proprietary	No	Linux, macOS, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?	
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	Yes	?	?	No	No	No	?	
PaddlePaddle	Baidu	Apache License	Yes	Linux, macOS, Windows	C++, Python	Python	No	Yes	Yes	Yes	Yes	Yes	Yes	?	Yes	
PlaidML	Vertex.AI	AGPL3	Yes	Linux, macOS, Windows	C++, Python	Keras, Python, C++, C	No	Yes	Yes	Yes		Yes	Yes	?	Yes	
PyTorch	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan	BSD	Yes	Linux, macOS, Windows	Python, C, CUDA	Python	Yes	Via separately maintained package[43][44][45]	Yes	Yes	Yes	Yes	Yes		Yes	
Apache SINGA	Apache Incubator	Apache 2.0	Yes	Linux, macOS, Windows	C++	Python, C++, Java	No	No	Yes	?	Yes	Yes	Yes	Yes	Yes	
TensorFlow	Google Brain team	Apache 2.0	Yes	Linux, macOS, Windows,[46] Android	C++, Python, CUDA	Python (Keras), C/C++, Java, Go, R[47], Julia, Swift	No	On roadmap[48] but already with SYCL[49] support	Yes	Yes[50]	Yes[51]	Yes	Yes	Yes	Yes	
TensorLayer	Hao Dong	Apache 2.0	Yes	Linux, macOS, Windows,[52] Android	C++, Python,	Python	No	On roadmap[48] but already with SYCL[49] support	Yes	Yes[53]	Yes[54]	Yes	Yes	Yes	Yes	
Theano	Université de Montréal	BSD	Yes	Cross-platform	Python	Python (Keras)	Yes	Under development[55]	Yes	Yes[56][57]	Through Lasagne's model zoo[58]	Yes	Yes	Yes	Yes[59]	No
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	BSD	Yes	Linux, macOS, Windows,[60] Android,[61] iOS	C, Lua	Lua, LuaJIT,[62] C, utility library for C++/OpenCL[63]	Yes	Third party implementations[64][65]	Yes[66][67]	Through Twitter's Autograd[68]	Yes[69]	Yes	Yes	Yes	Yes[70]	
Wolfram Mathematica	Wolfram Research	Proprietary	No	Windows, macOS, Linux, Cloud computing	C++, Wolfram Language, CUDA	Wolfram Language	Yes	No	Yes	Yes	Yes[71]	Yes	Yes	Yes	Under Development	
VerAI	VerAI	Proprietary	No	Linux, Web-based	C++,Python, Go, Angular	Graphical user interface, cli	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	

Excellent comparison: https://skymind.ai/wiki/comparison-frameworks-dl4j-tensorflow-pytorch

Excellent comparison: https://arxiv.org/pdf/1511.06435.pdf

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

Today’s Topics

- Ways of seeing: image and video acquisition
- Evolution of computer vision (before versus after 2012)
- Background of machine learning and neural networks
- Training deep neural networks: hardware & software


The End