Data Visualization Group
Northstar Middle School

Denny Lee

About Me

- Sr Staff Developer Advocate, Databricks
- Senior Director of Data Science Engineering at SAP Concur
- Principal Program Manager at Microsoft for Azure Cosmos DB, Project Isotope (Azure HDInsight), SQL Server

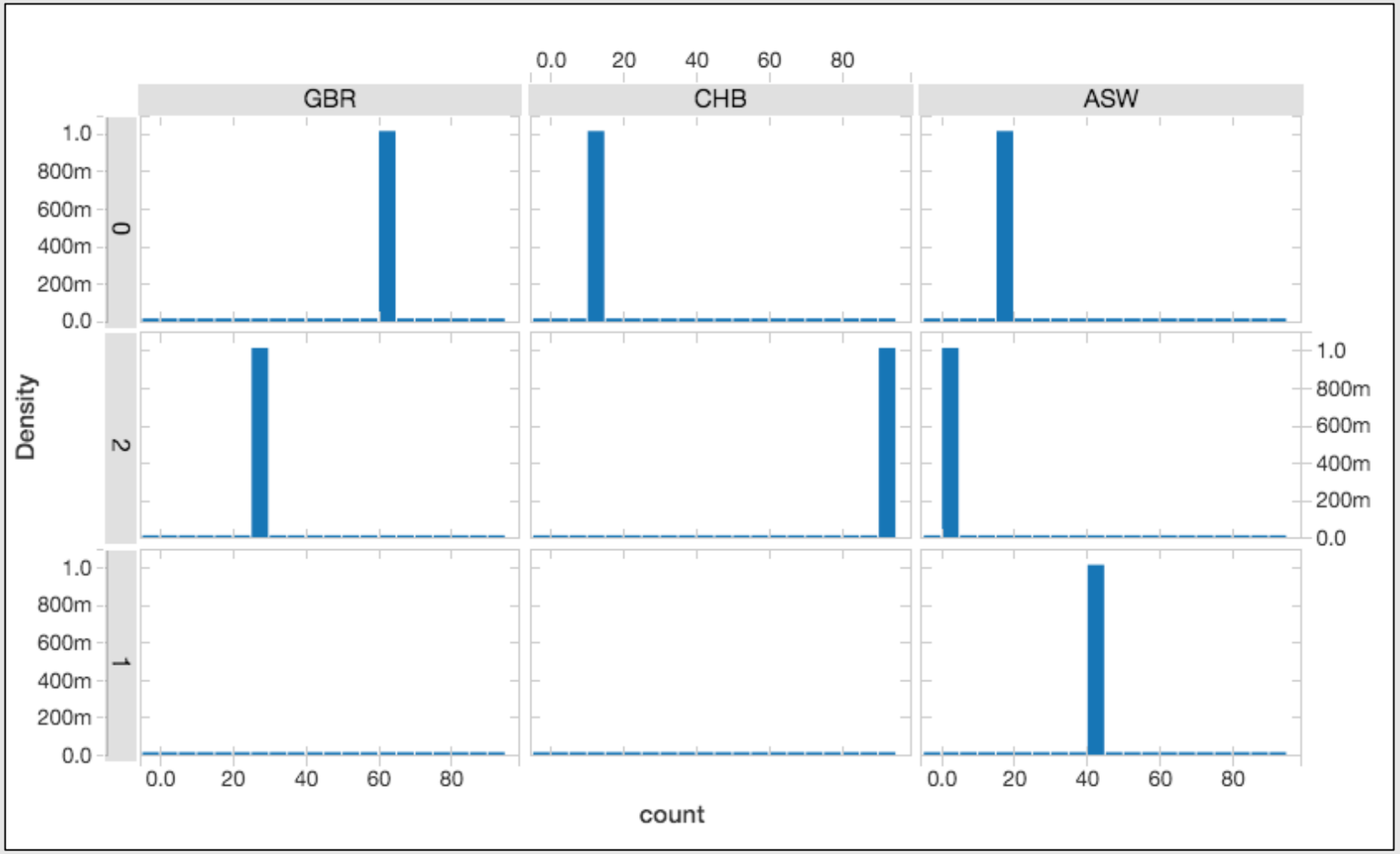
Denny surprised that he's still awake

Session Goals

- What is Deep Learning?
- Deep Learning Applications – including data visualization
- Biology vs scientific model
- Demos!
- Maths (completely optional)

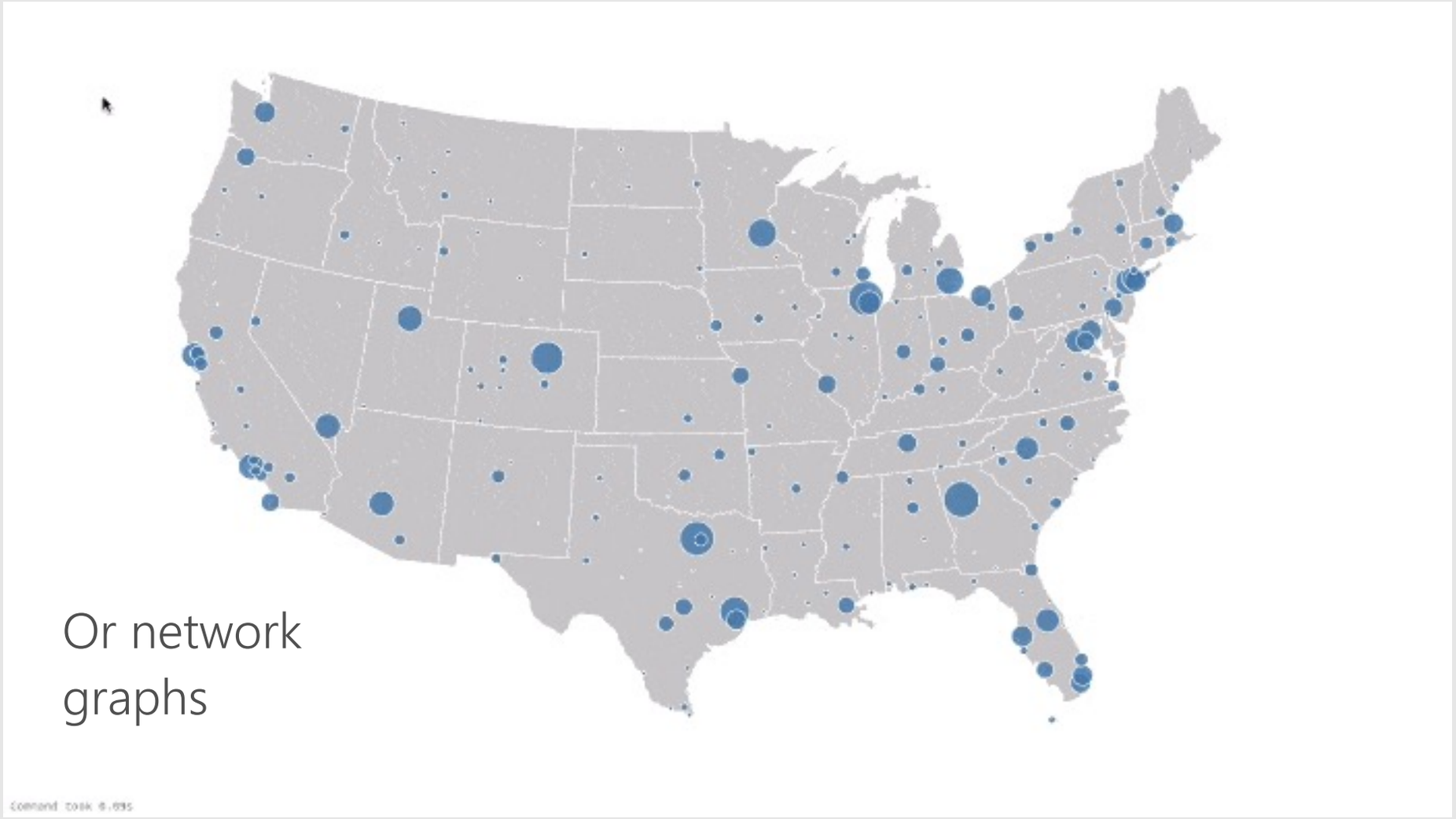
What is Deep Learning?


WAIT, ISN'T THIS
A VISUALIZATION SESSION?
imgflip.com



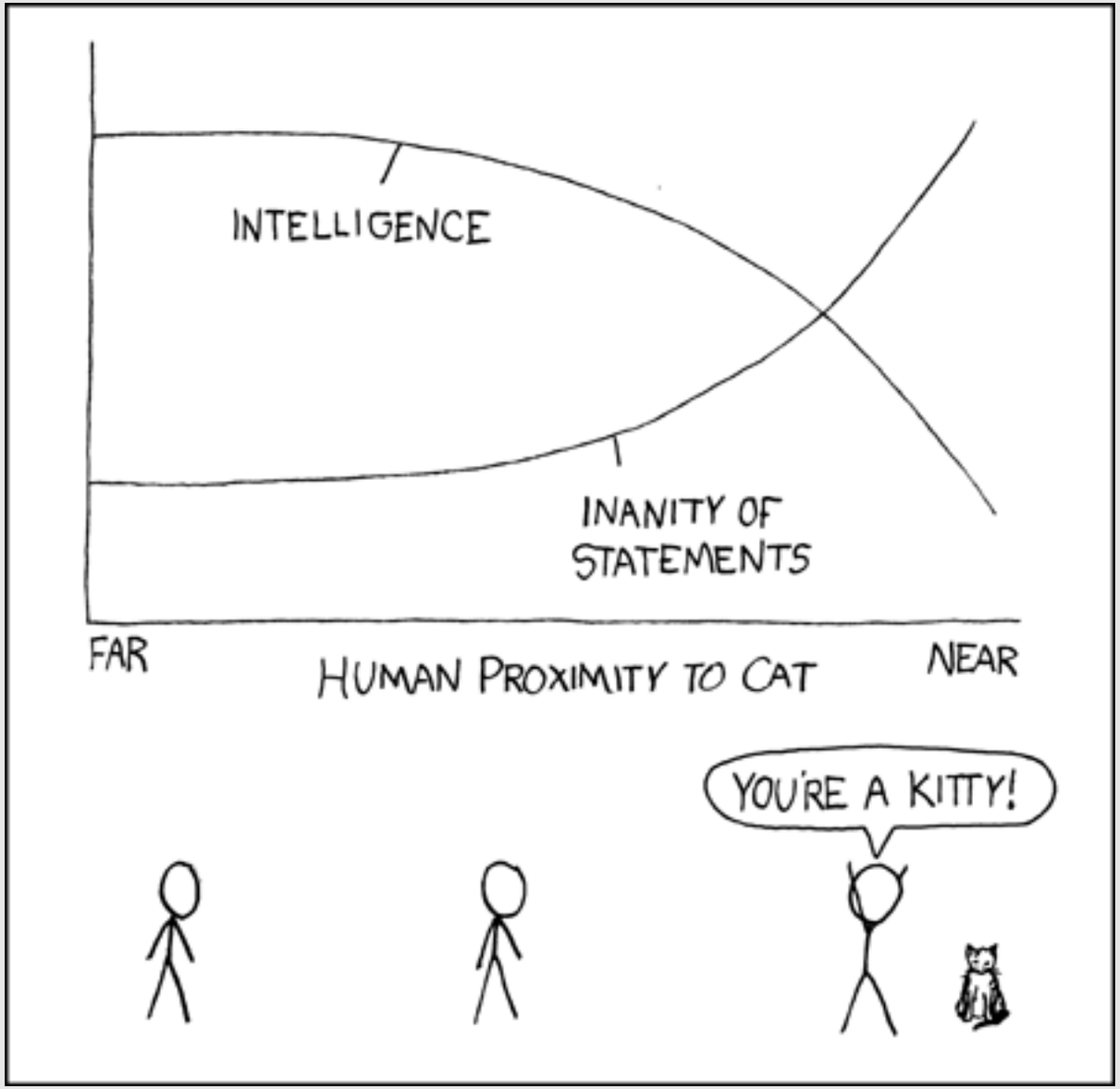
Data visualization can mean a lot of things

.. like graphs!



Or network graphs

Or even more charts …



xkcd comic 231

Or even worse …

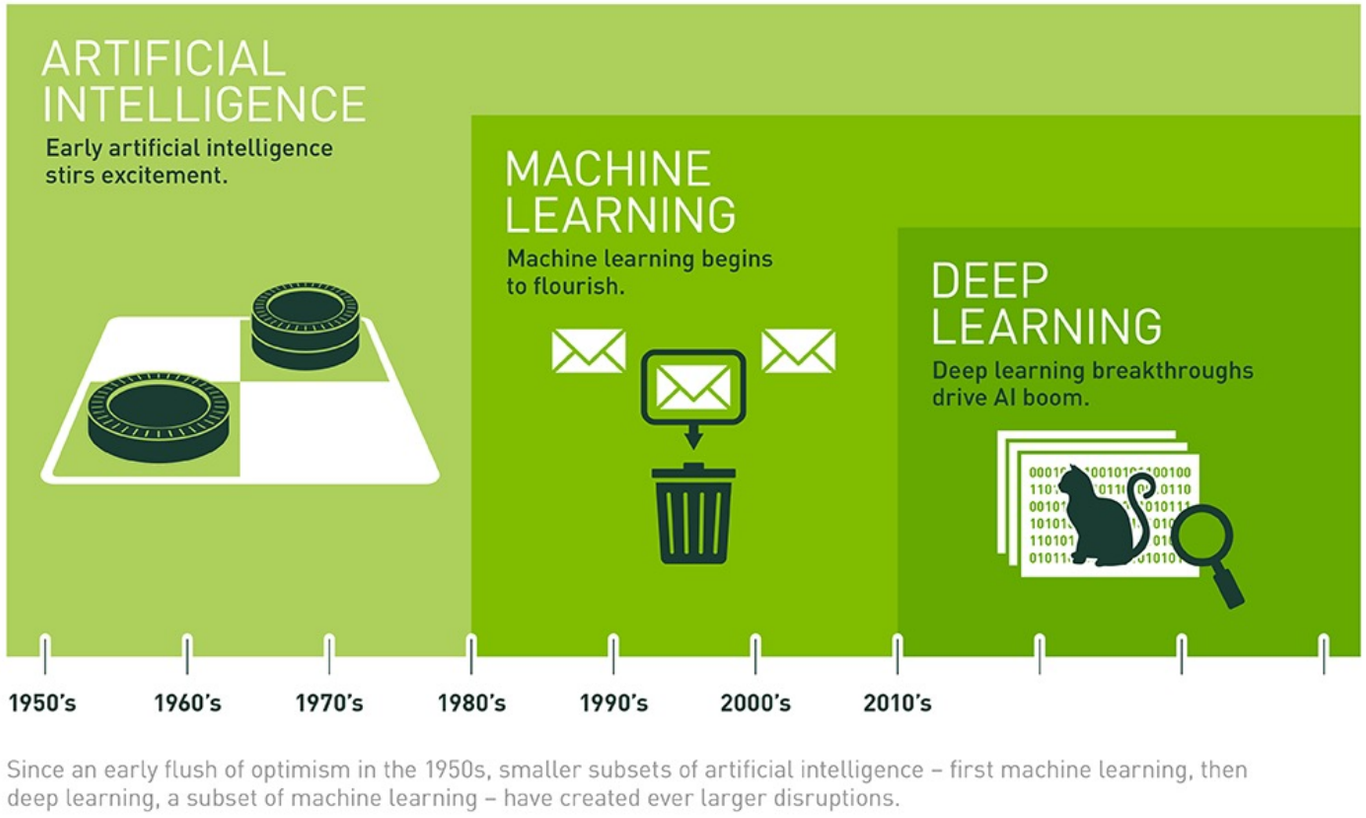
But today, we're talking about the ability to interpret visual data

source: http://bit.ly/2jcquh5

Deep Learning

The next step in AI

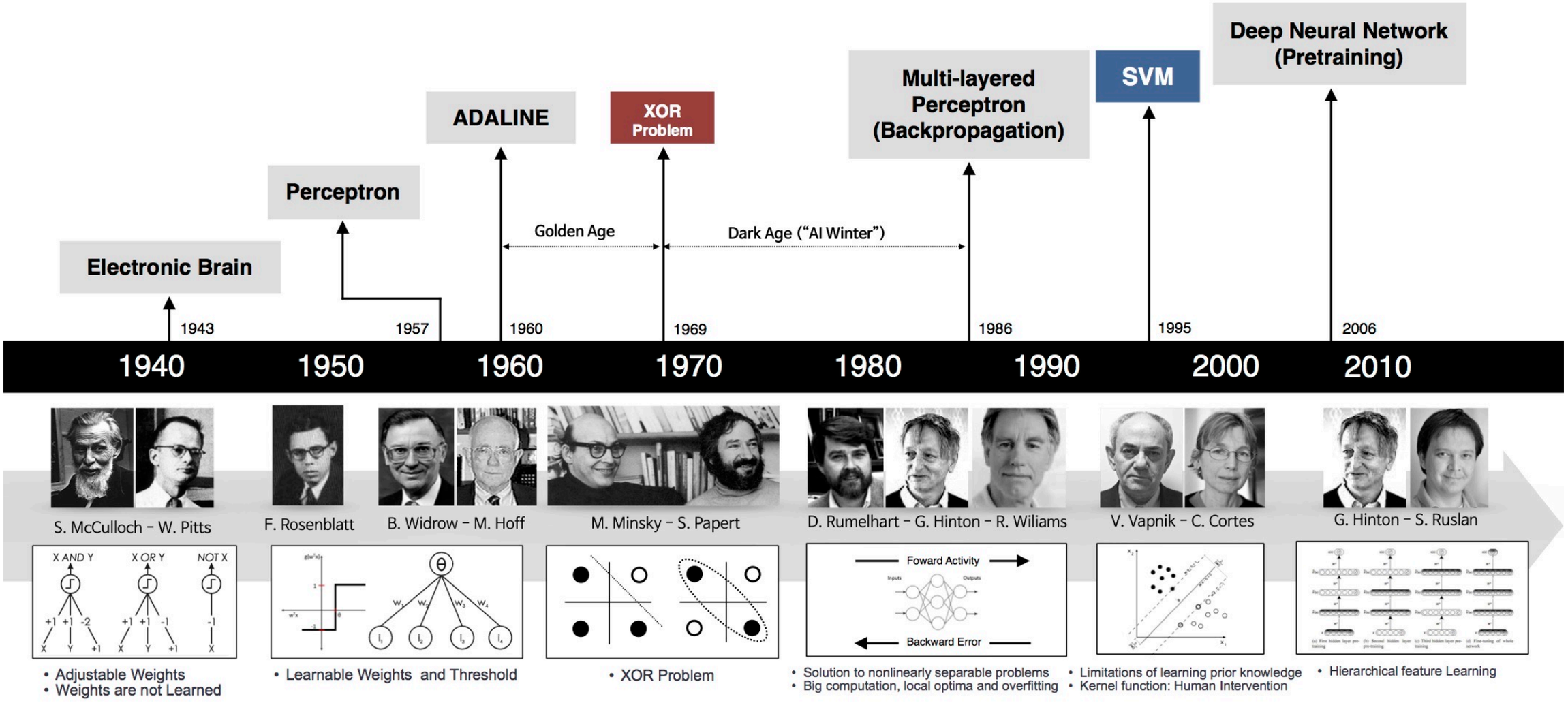
Learning… Lots of learning…



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Source: http://bit.ly/2suufGJ

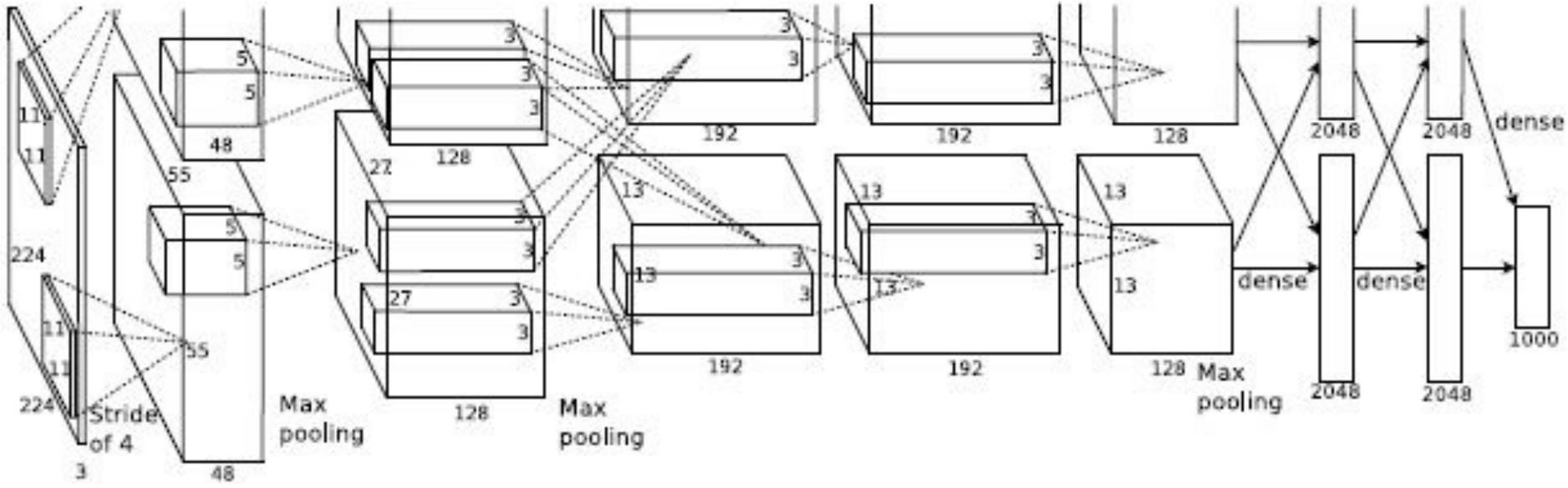
History of AI research



Source: http://bit.ly/2EILhpu

Deep learning

- Vast and sophisticated ANN structure
- Tens, hundreds, thousands… of hidden layers
- Ability to not only map the input to the output but recognize what inputs are needed for the task

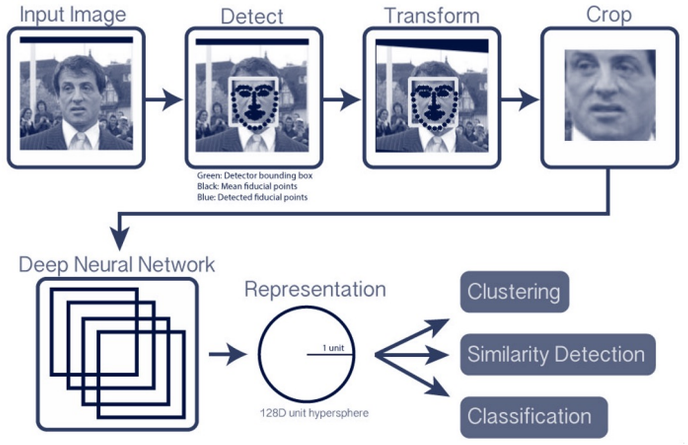


Example: AlexNet (ImageNet 2012 winner); Source: http://bit.ly/2tNa3Ps

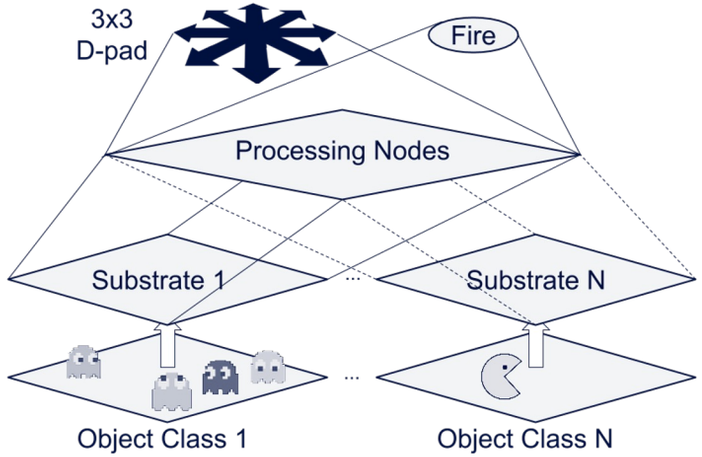
Deep Learning Applications

Applications of neural networks

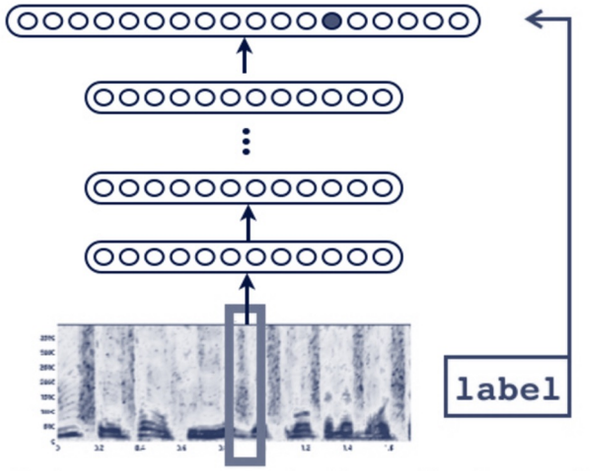
Object Recognition: Facial



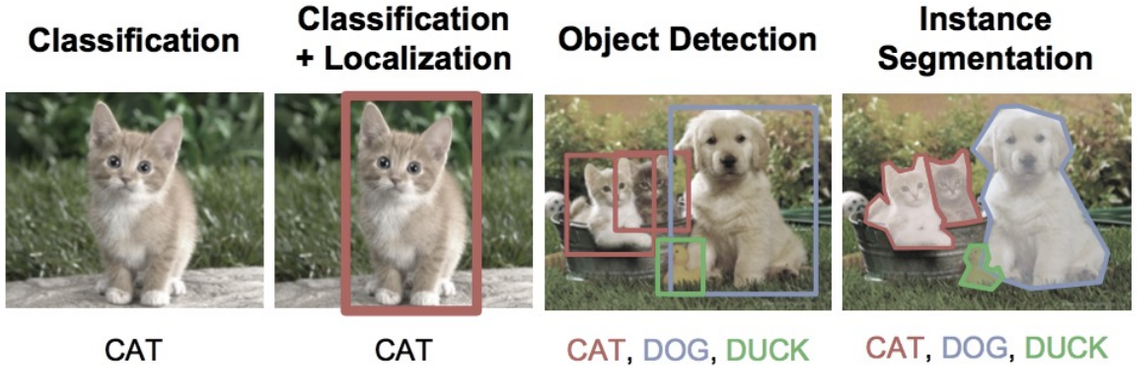
Game Playing



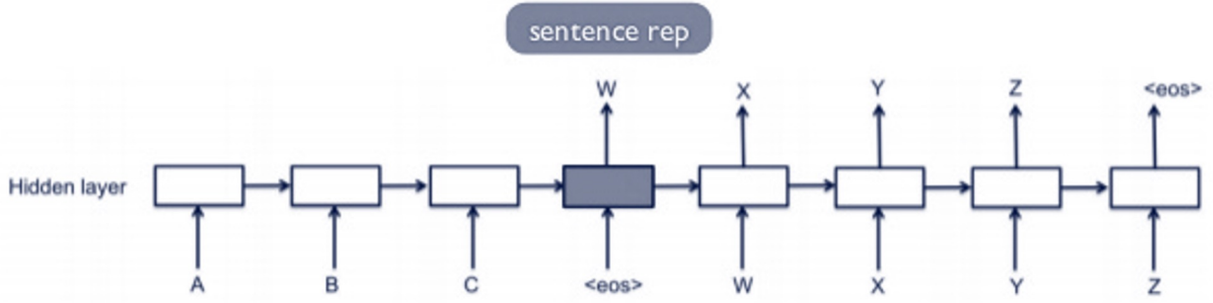
Object Recognition: Speech



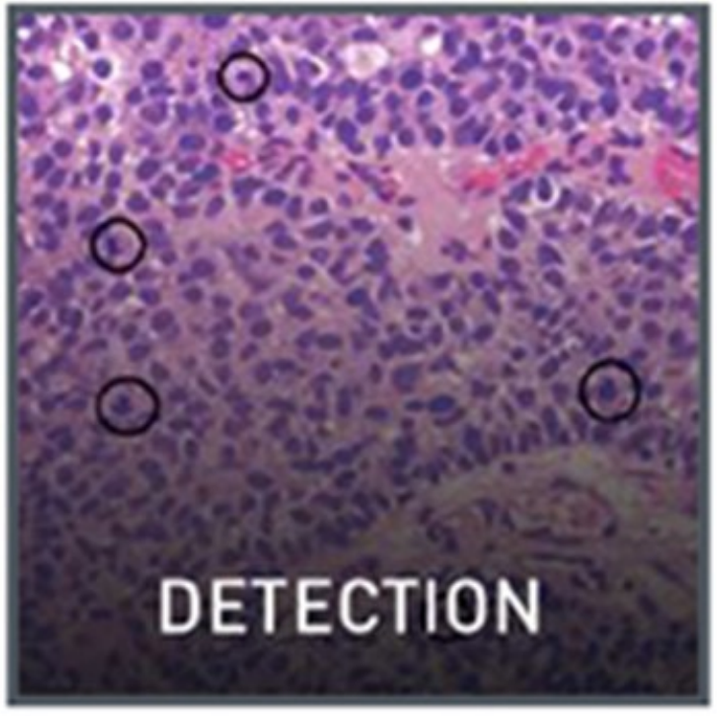
Object Classification

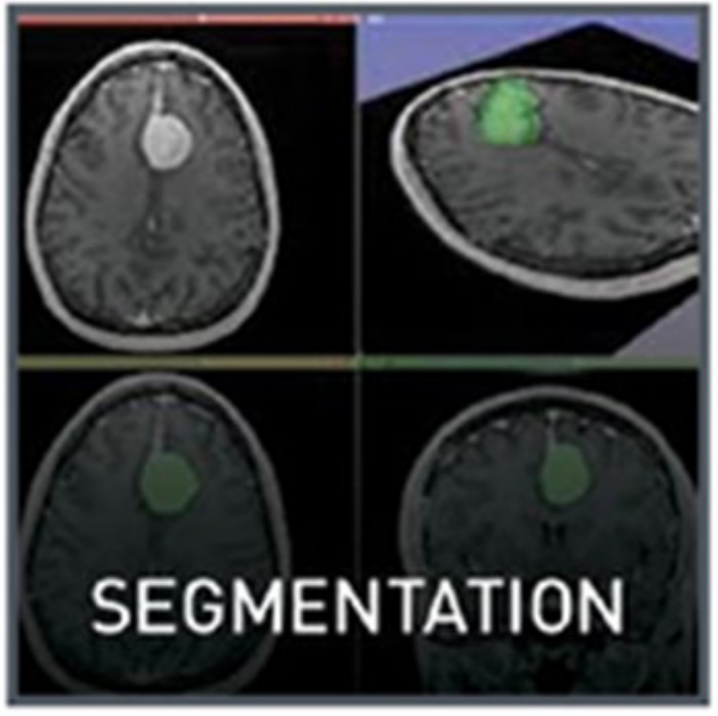


Language Translation



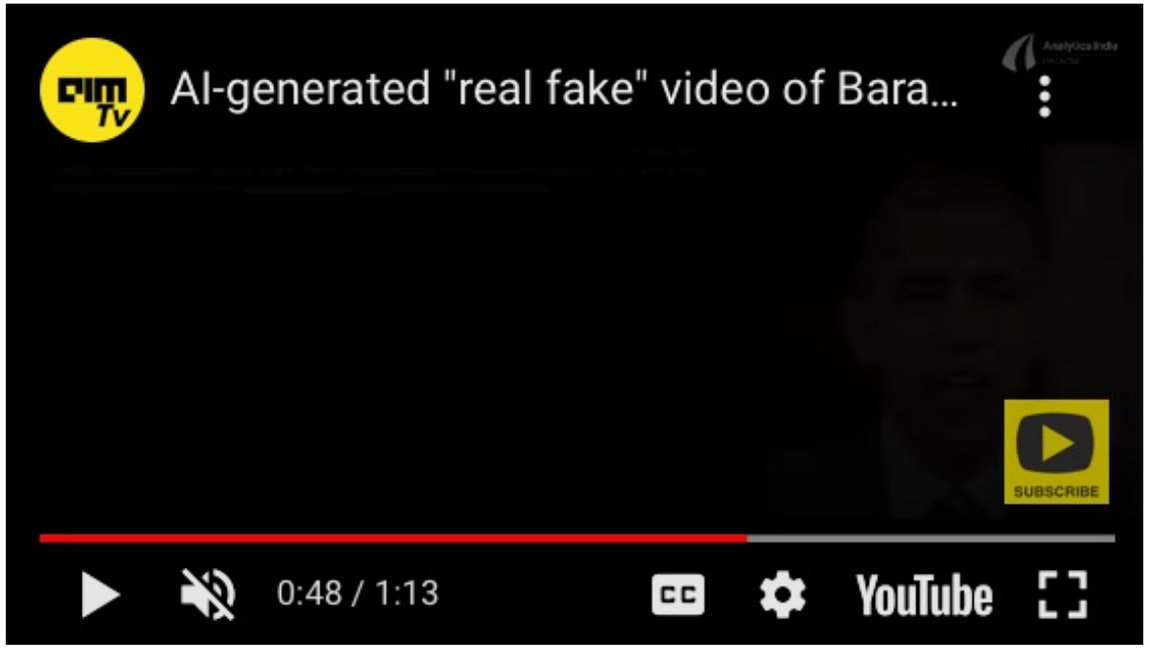
Healthcare





Source: http://bit.ly/2BXF5sR

Faking videos

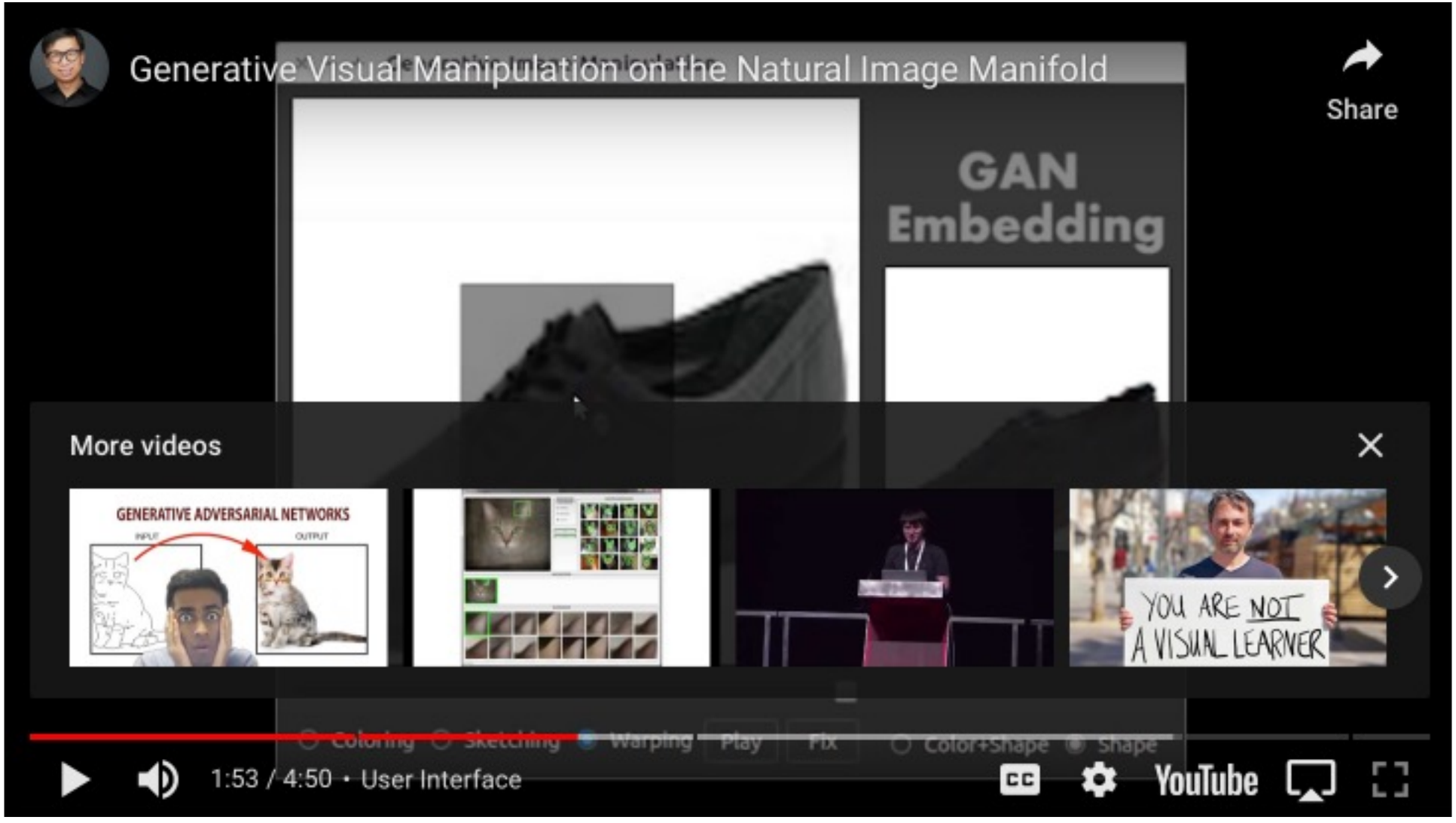


Source: https://youtu.be/dkoi7sZvWiU

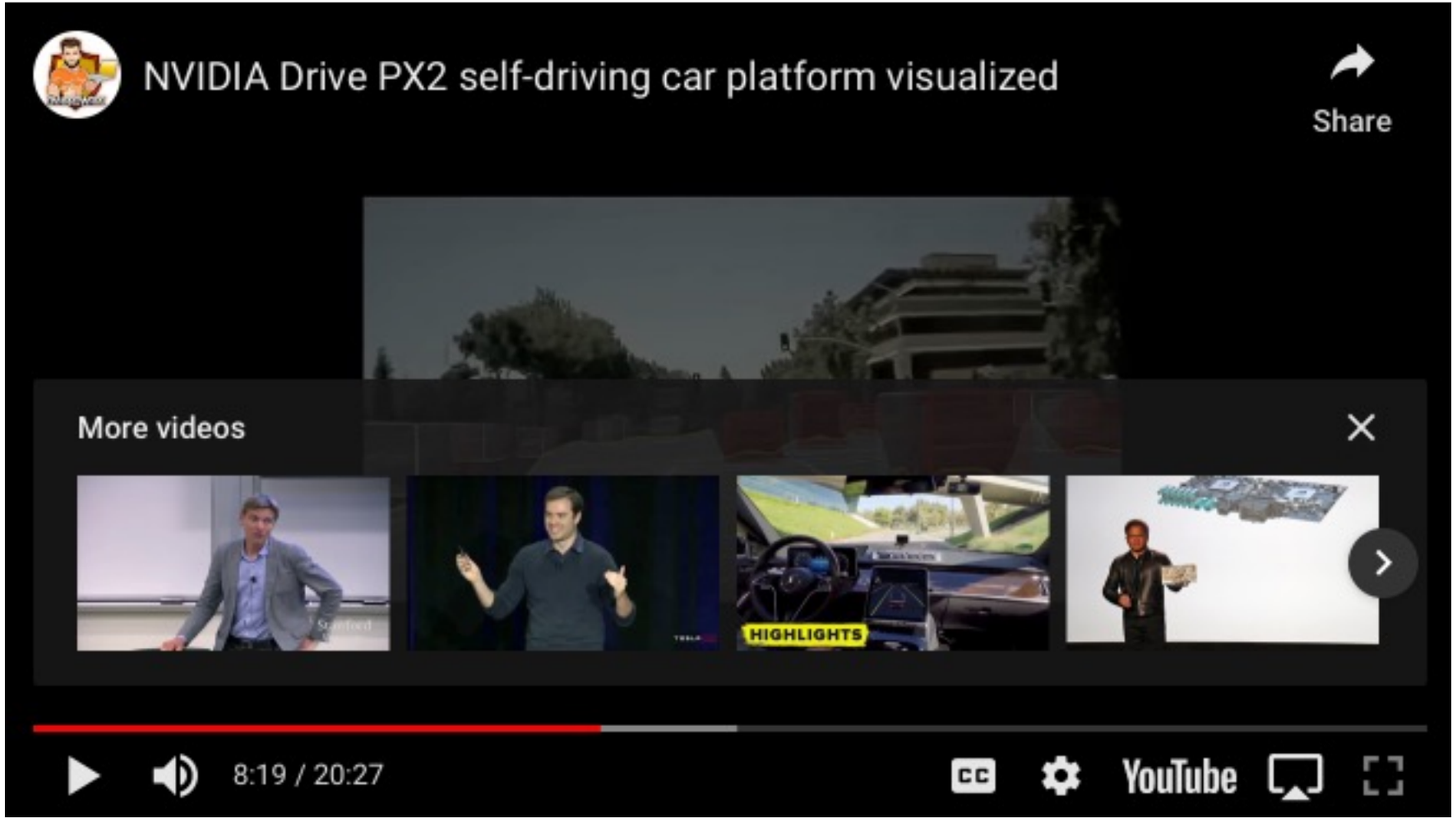
Source: https://youtu.be/JzgOfISLNjk

Drawing support

https://www.youtube.com/watch?v=9c4z6YsBGQ0

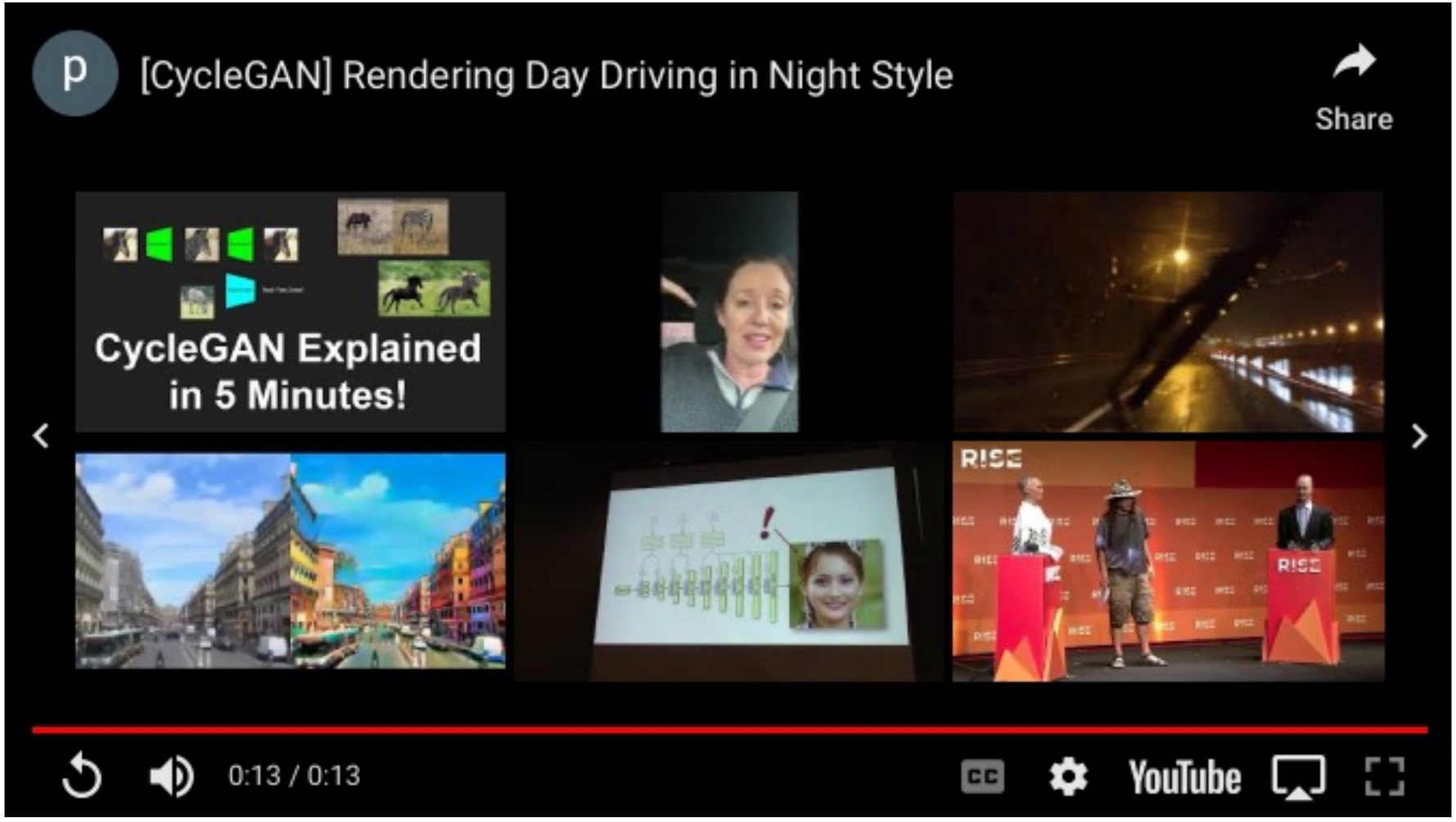


Self-driving cars



Source: https://youtu.be/URmxzxYlmtg?t=6m30s

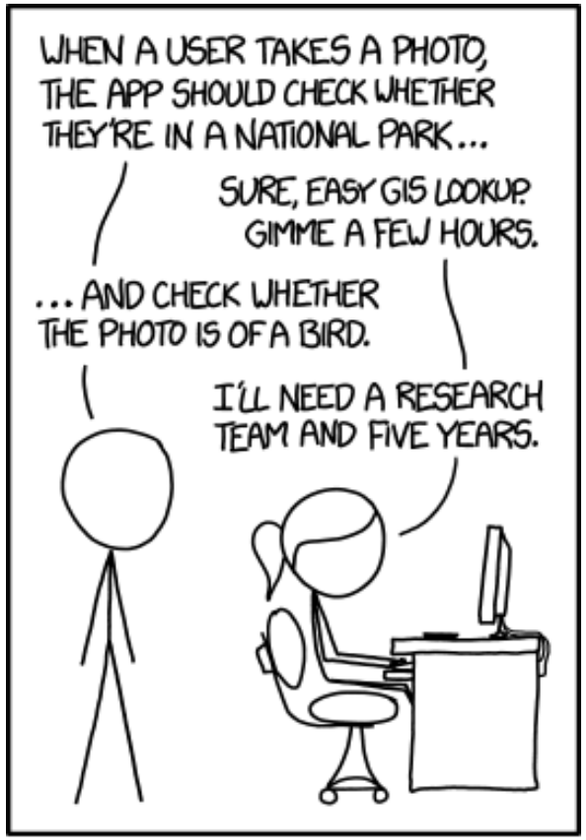
Turning the day into night



Source: https://youtu.be/N7KbfWodXJE

Recognizing Objects

xkcd #1425 (View original here)



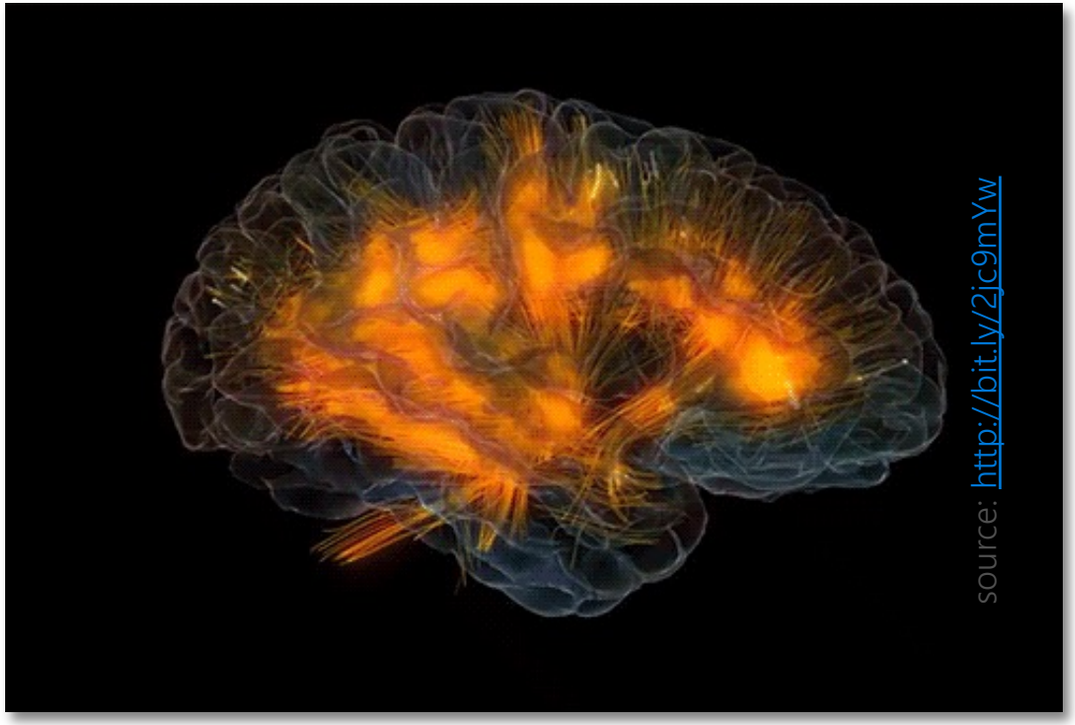
Really good explanation of convolution
http://bit.ly/2suZGkk

source: http://bit.ly/2jjzFyt

What are artificial neural networks?

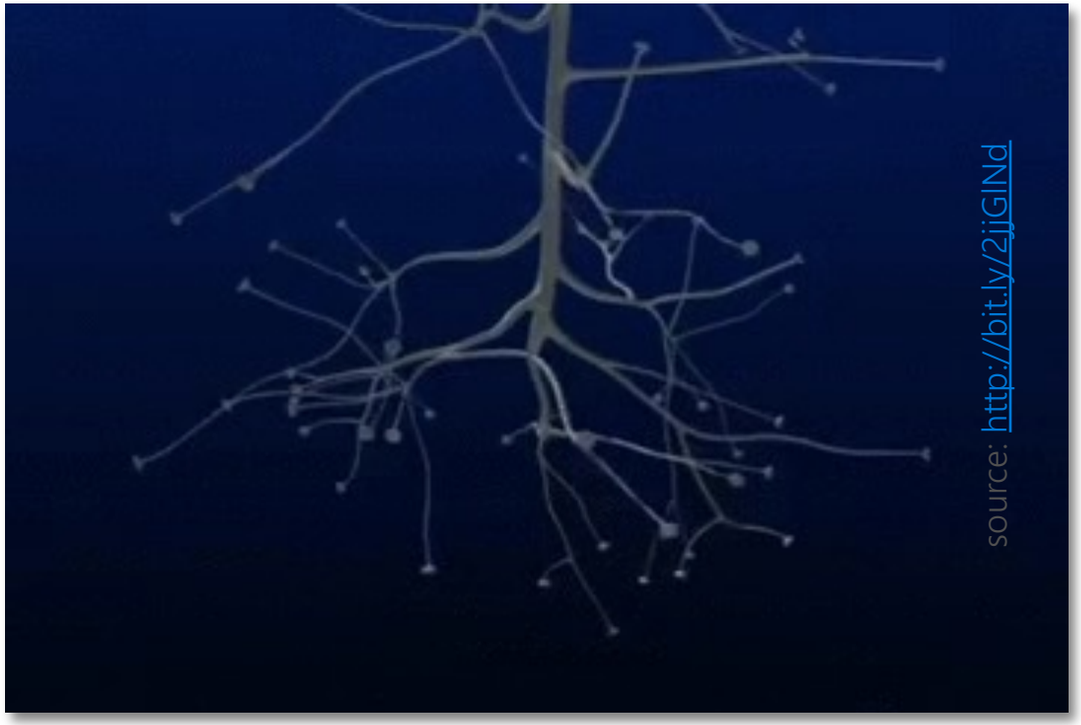
An imitation of human's brain

- Dense net of simple structures
- Around 100 billion neurons
- Each connected to ~10k other neurons
- 10^{15} synaptic connections



A neuron at work

- Receive signal
- Process it
- Propagate the signal (or not)

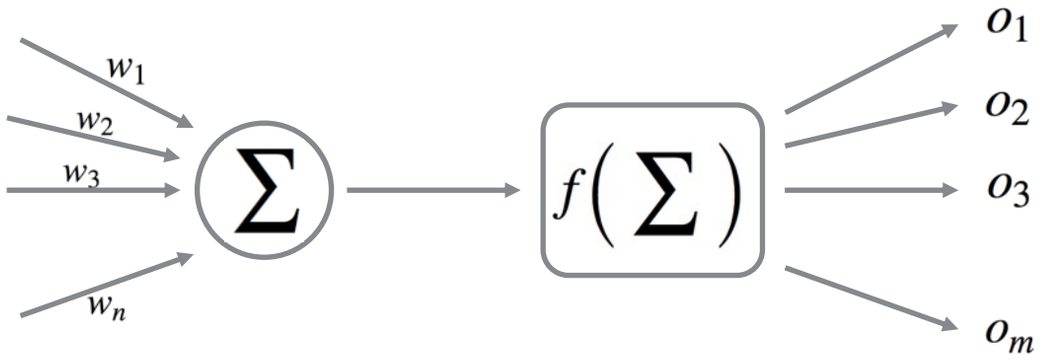


Conceptual mathematical model

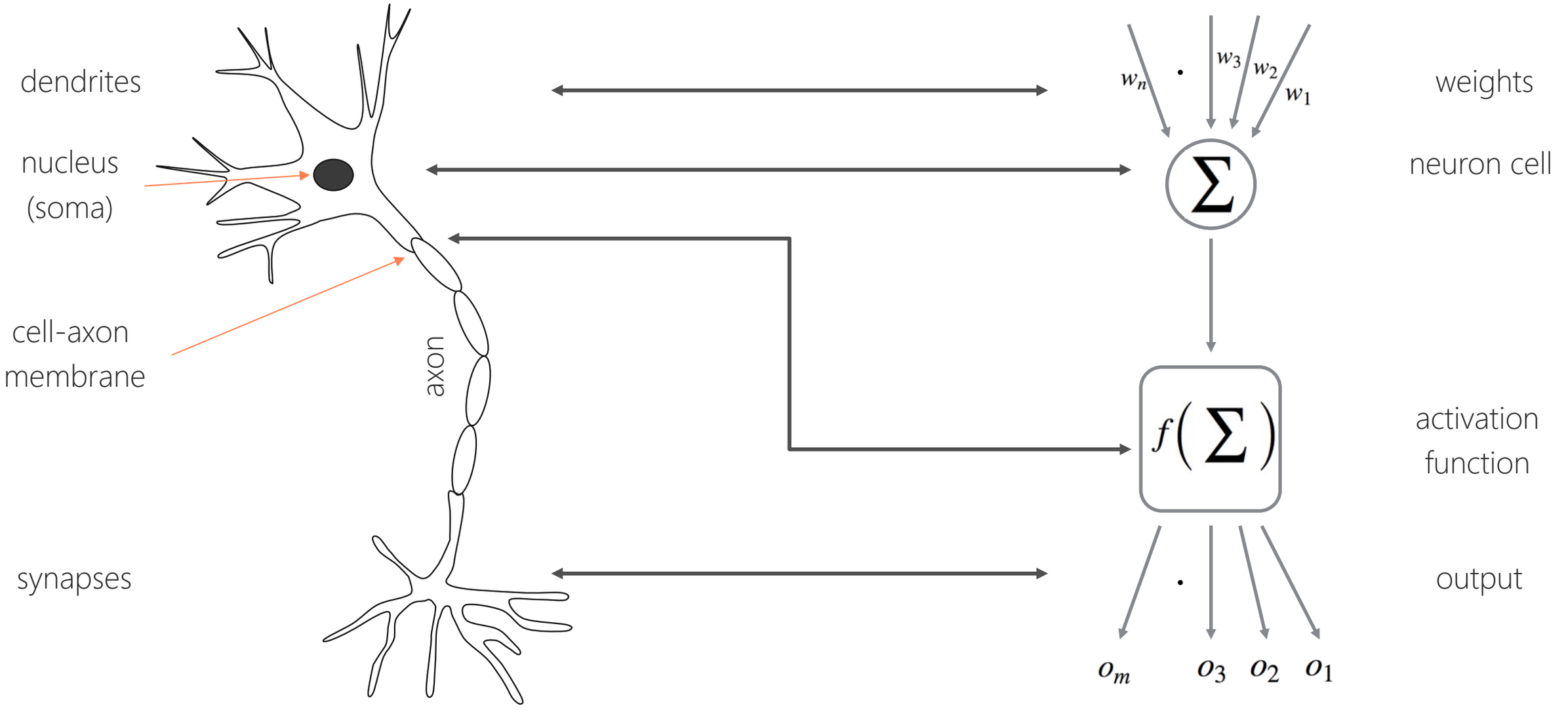
- Receives input from *n* sources
- Computes weighted sum

$$h_1 = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$$

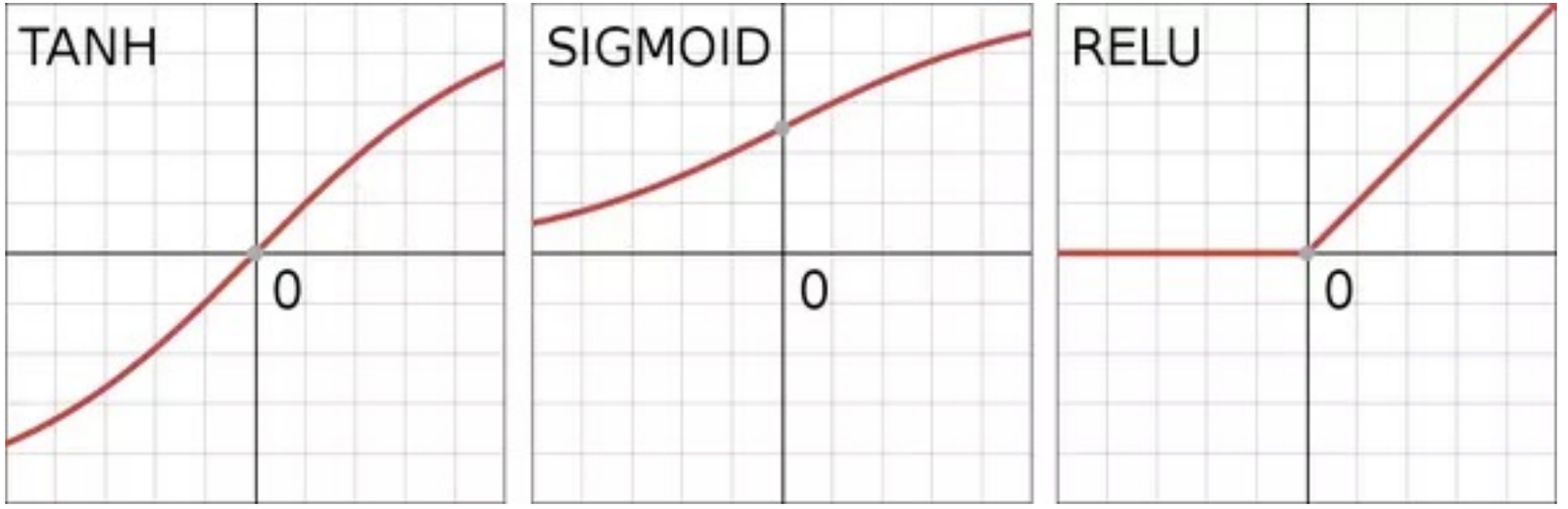
- Passes through an *activation function*
- Sends the signal to *m* succeeding neurons



Parallels

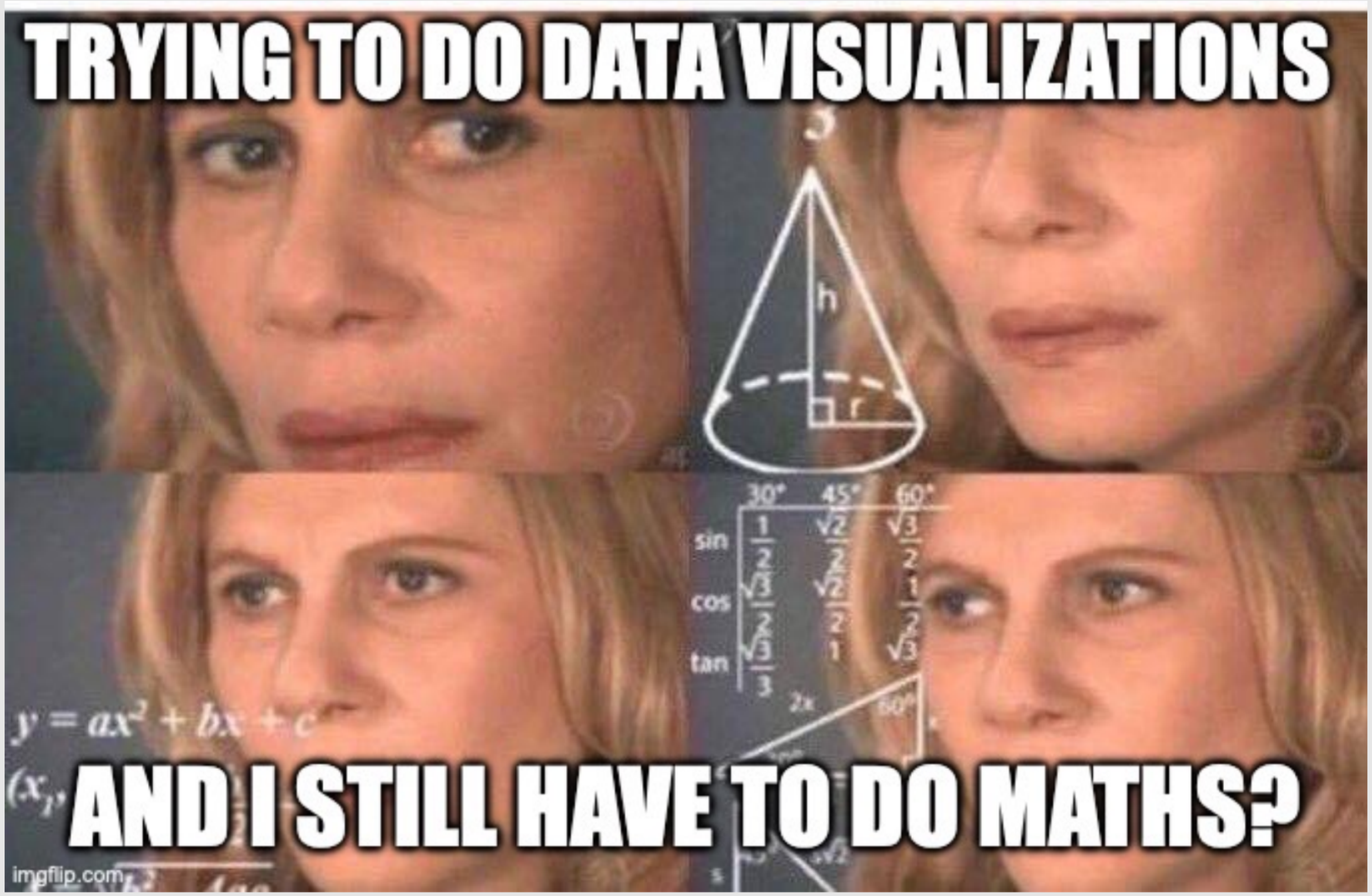

dendrites
nucleus
(soma)
cell-axon
membrane
axon
synapses
w_n
w_3
w_2
w_1
Σ
$f(\Sigma)$
o_m
o_3
o_2
o_1
weights
neuron cell
activation
function
output

Activation functions

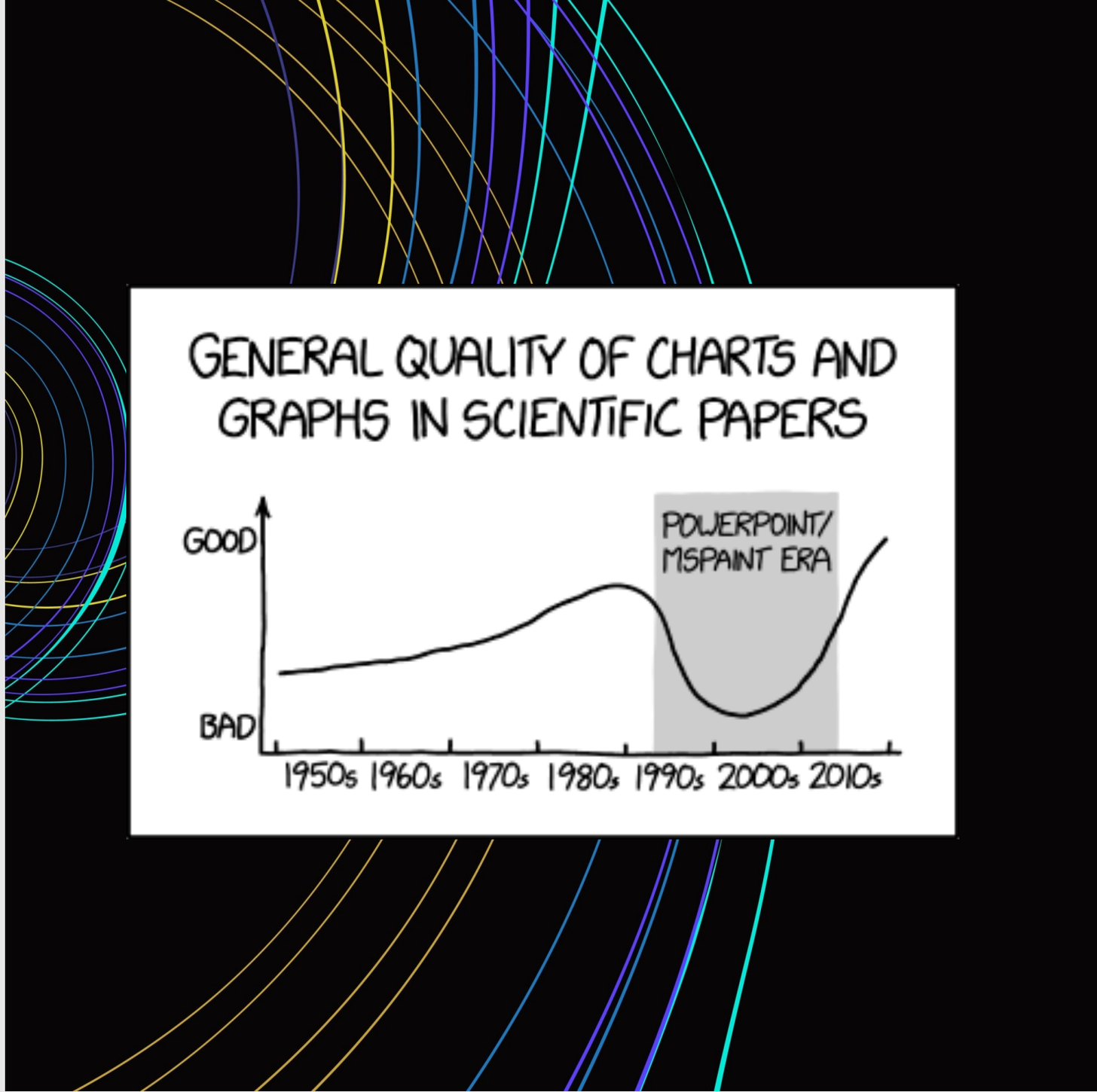


- Bias (threshold) activation function was proposed first
- Sigmoid and tanh introduce non-linearity with different codomains
- ReLu is the latest and greatest due to backprop (more later)

Source: http://bit.ly/2tKhbMS


TRYING TO DO DATA VISUALIZATIONS
h
r
30°
45°
60°
sin
cos
tan
$y = ax^2 + bx + c$
AND I STILL HAVE TO DO MATHS?
imgflip.com

Don’t worry, its demo time

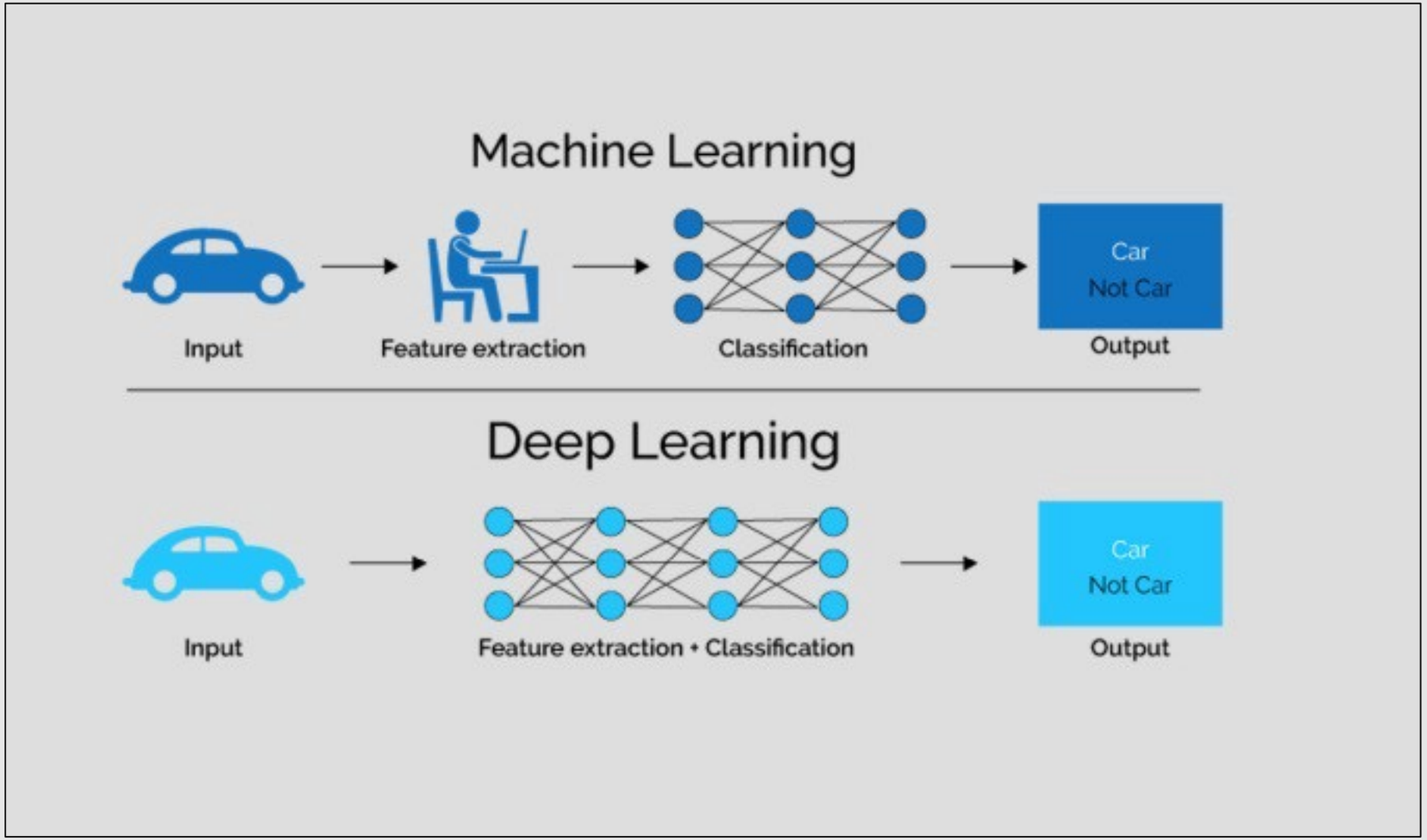


MNIST Keras Demo

Try it out at:

- Download the notebooks at Keras MNIST CNN notebooks: CPU | GPU
- The notebooks are for Databricks Community Edition (free) so work with your parents and teachers to get access

MNIST Keras Demo



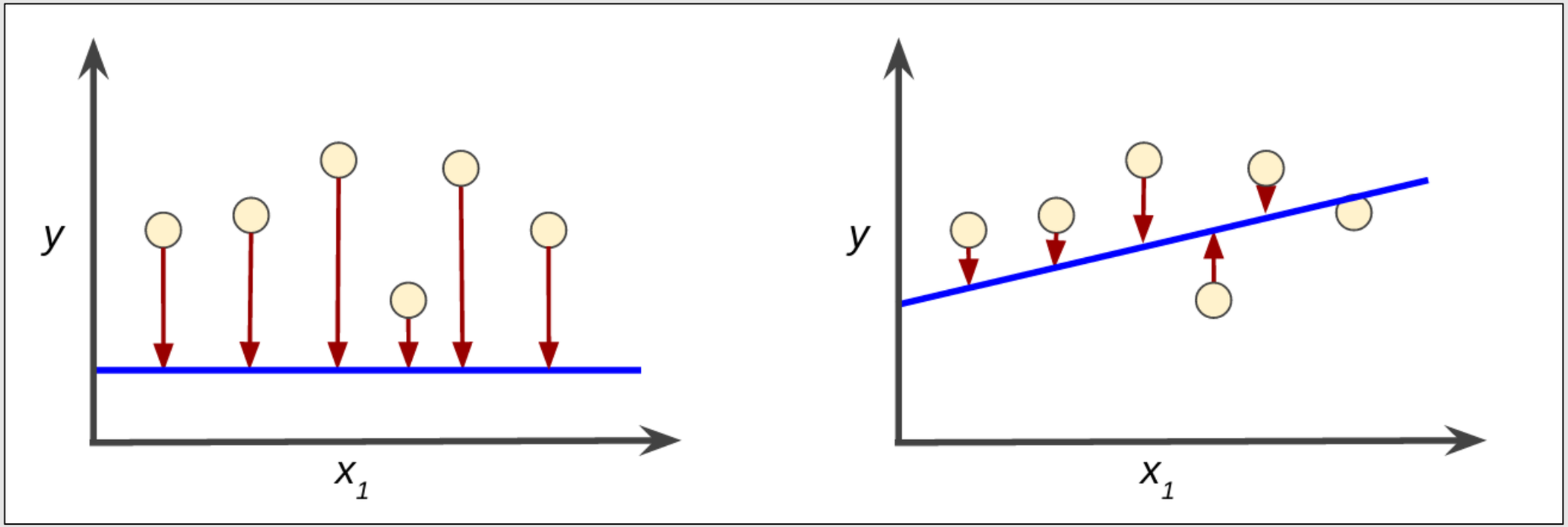
Source: [What is Training Data its types and why it is important?]

MNIST Keras Demo

More data so we can better train the model

Source: [ConvNetJS MNIST demo]

MNIST Keras Demo



High loss in the left model; low loss in the right model.
Source: Descending into ML: Training and Loss

MNIST Keras Demo

Test Accuracy: More green, less red

Source: [ConvNetJS MNIST demo]

ConvNetJS MNIST demo

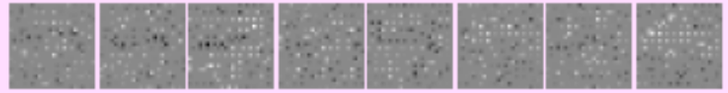
Weights:

Weight Gradients:

relu (24x24x8)
max activation: 4.22335, min: 0
max gradient: 0.00031, min: -0.00037

Activations:

Activation Gradients:



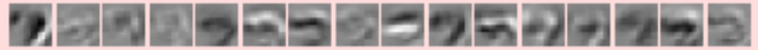
pool (12x12x8)
pooling size 2x2, stride 2
max activation: 4.22335, min: 0
max gradient: 0.00031, min: -0.00037

Activations:

Activation Gradients:

conv (12x12x16)
filter size 5x5x8, stride 1
max activation: 6.32598, min: -7.38596
max gradient: 0.00051, min: -0.00036
parameters: 16x5x5x8+16 = 3216

Activations:



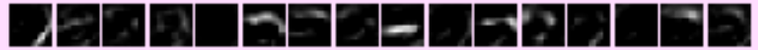
Activation Gradients:

Weights:

Weight Gradients:

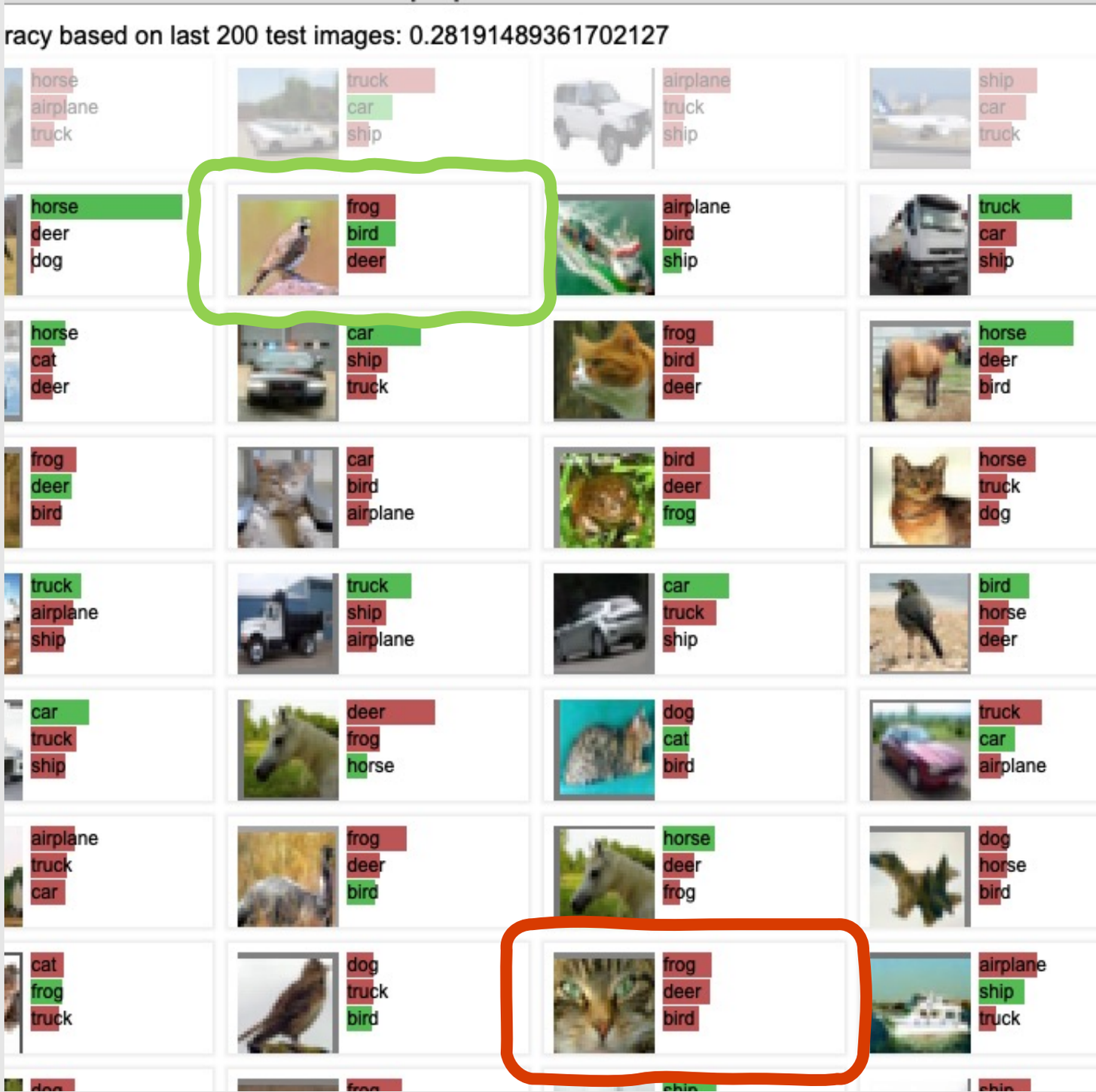
relu (12x12x16)
max activation: 6.32598, min: 0
max gradient: 0.00051, min: -0.00036

Activations:



Activation Gradients:

ConvNetJS CIFAR-10 demo



Deep Visualization Toolbox

Source: https://youtu.be/AgkfIQ4IGaM?t=118

Object Detection

Image AI: Video Object Detection, Tracking and Analysis utilizing YOLOv3 model

Source: Coffee Object Detection Demo

DALL·E 2

is a new AI system that can create realistic images and art from a description in natural language.

An astronaut
riding a horse in a
photorealistic style

https://openai.com/dall-e-2/

Want to know more about DALL·E 2?

How DALL·E 2 actually works?

- Link text and visual semantics via CLIP
  - *The insight that makes CLIP so powerful is the bright idea to train models to not just identify which category (of a pre-defined list of options) an image belongs to, but to identify the caption of each image from a list of random captions.*[1]

- Generating Images from Visual Semantics by doing the inverse with GLIDE via a diffusion model

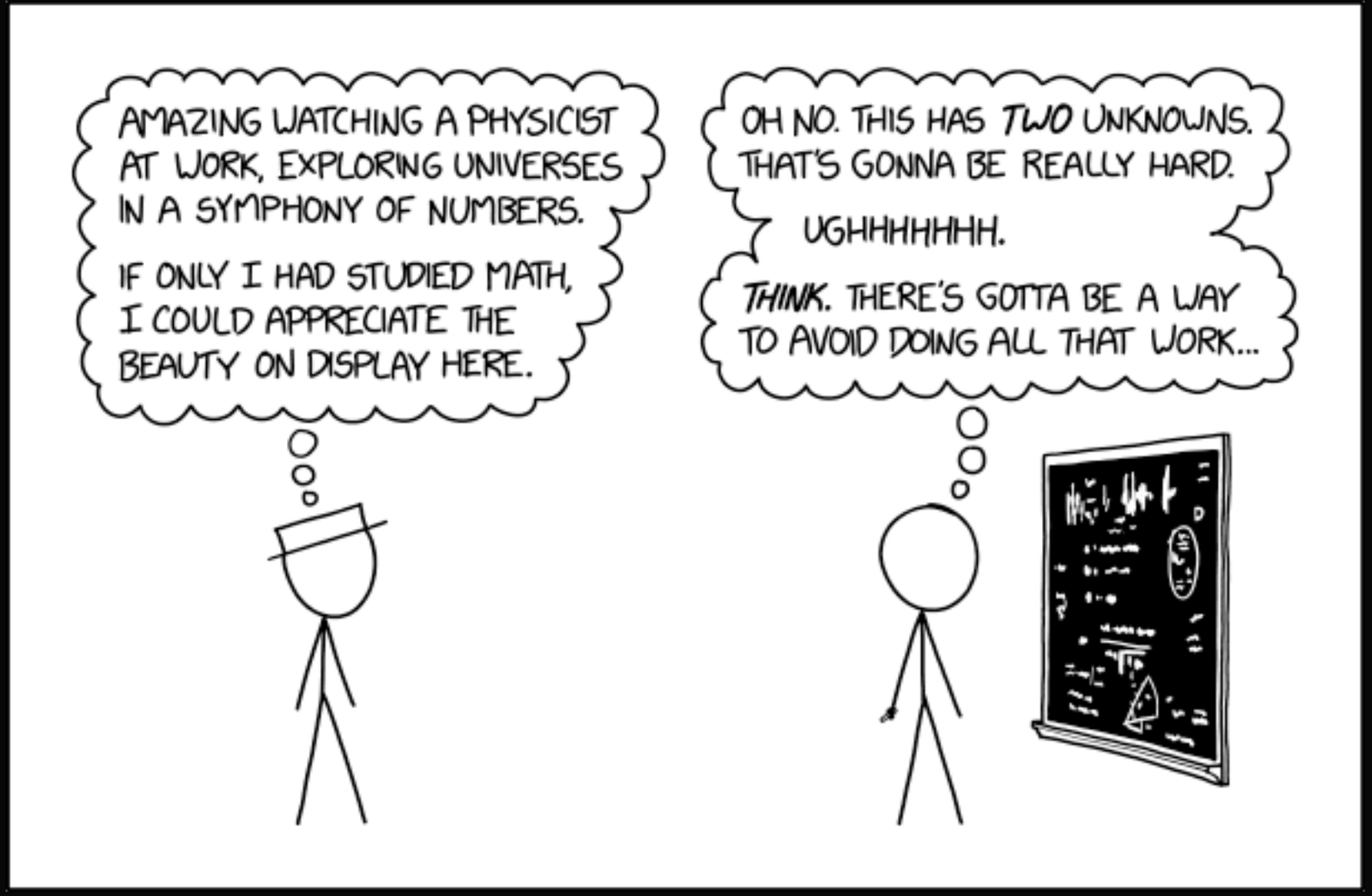
Want to know more about DALL·E 2?

What are diffusion models?

- *Diffusion Models learn to generate data by reversing a gradual noising process.*
- Think of this as unscrambling a Rubik's cube after scrambling it one step at at time (after forgetting about the steps) or
- *More practically, you could train a neural network to go from a state of disorder to a state of less disorder.*

[1] DALL·E 2.0, Explained

Want to do more math?



References

These slides: Data Visualization – Northstar Middle School
Keras MNIST Notebooks: GPU | CPU

Neural networks introduction

Artificial Neural Network

- Organized into layers of neurons
- Typically 3 or more: input, hidden and output
- A black-box model

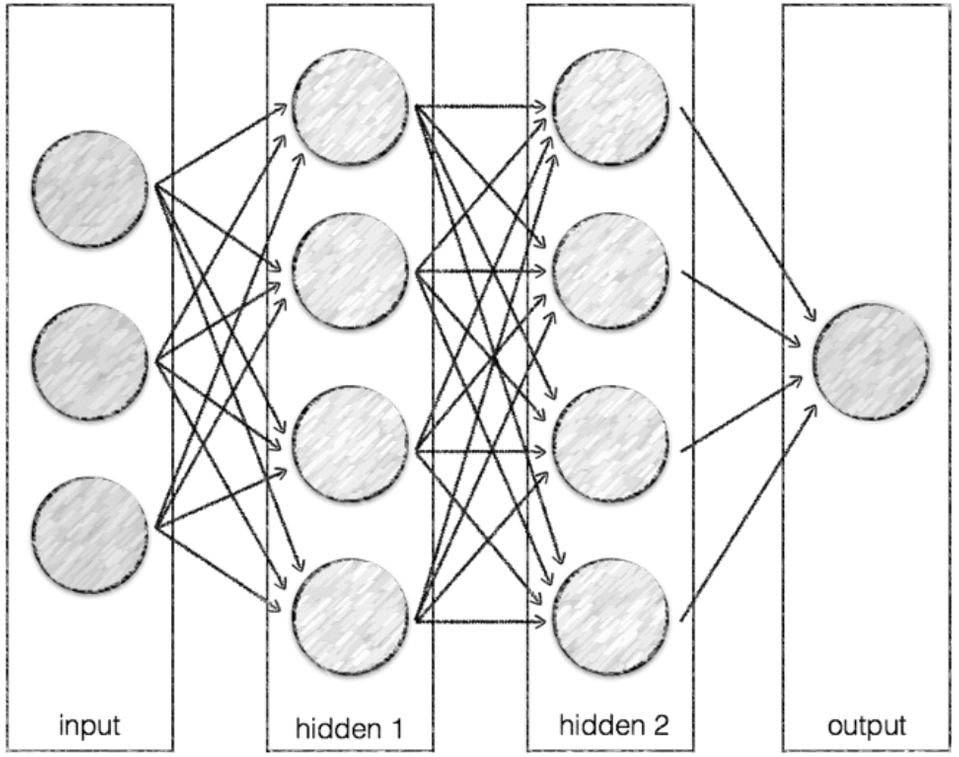
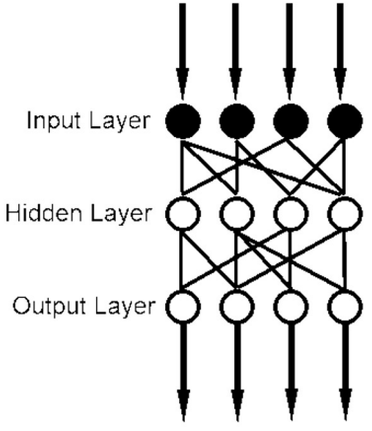


Image sources linked

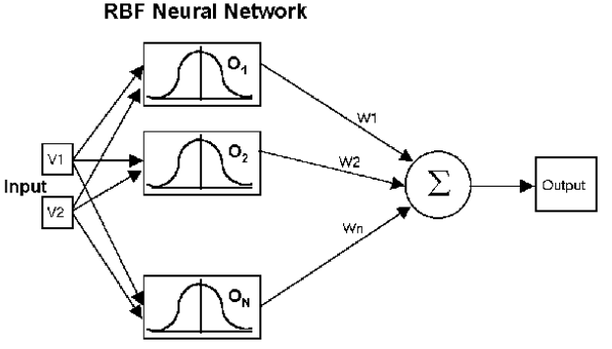
Types of neural networks

Some examples

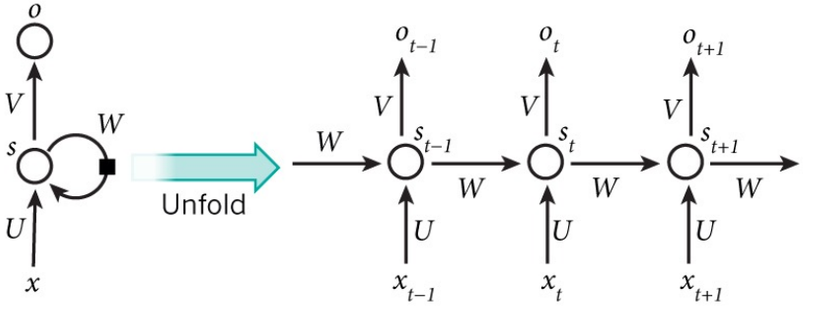
Single- or multi-layer
feedforward networks

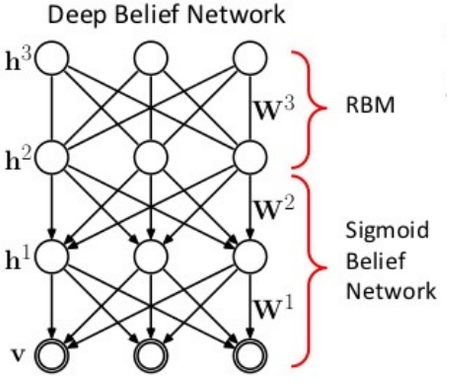


Radial Basis Function



Recurrent Neural Networks





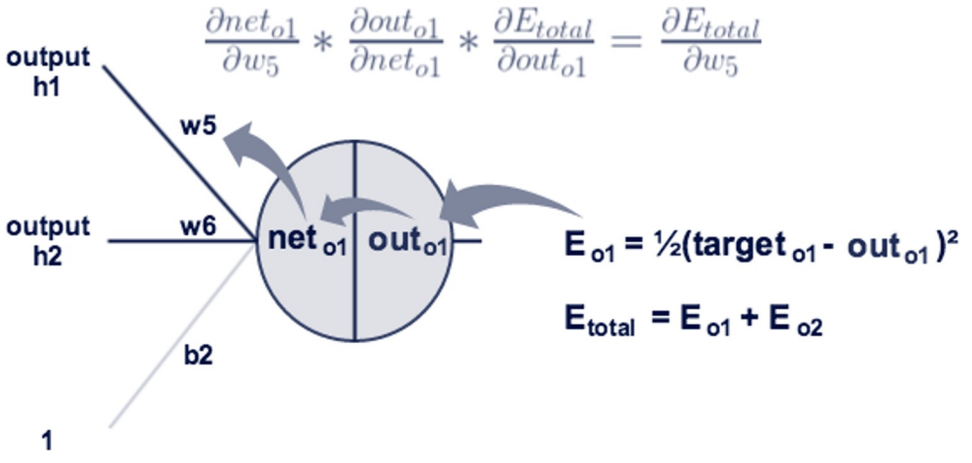
Deep belief networks

Image sources linked

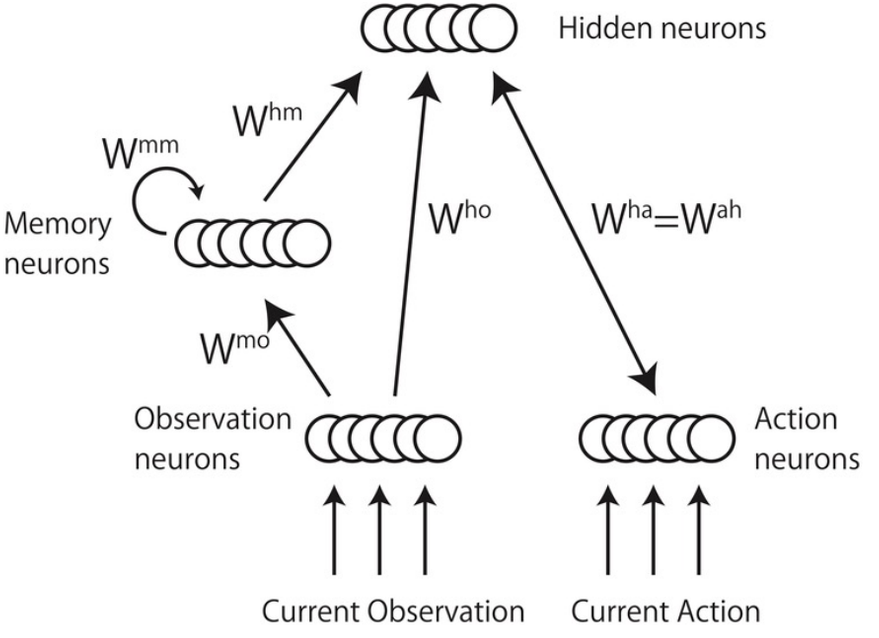
Learning

Some examples

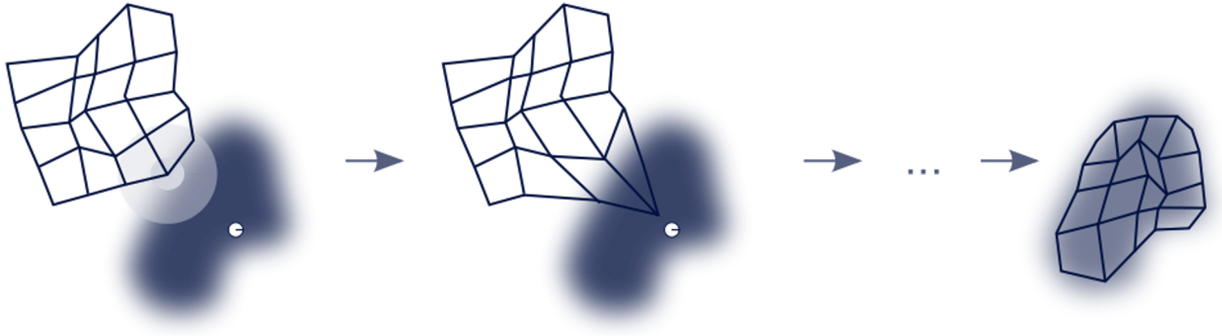
Backpropagation



Memory based learning



Self-organizing maps (competitive learning)



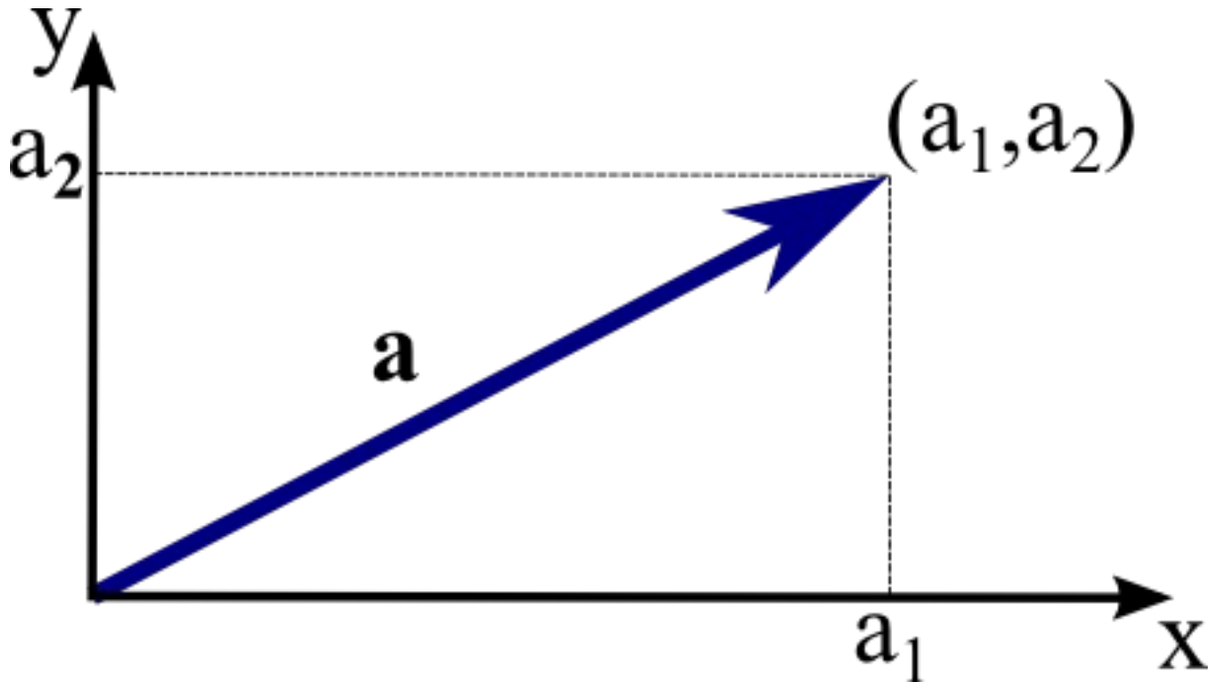
Introduction to tensors

Scalars, vectors, matrices and tensors

Scalar 2.01

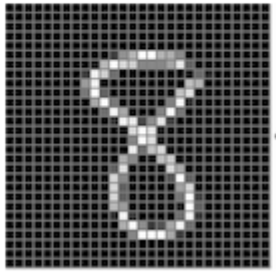
Scalars, vectors, matrices and tensors

Vector



Scalars, vectors, matrices and tensors

Matrix



28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Scalars, vectors, matrices and tensors

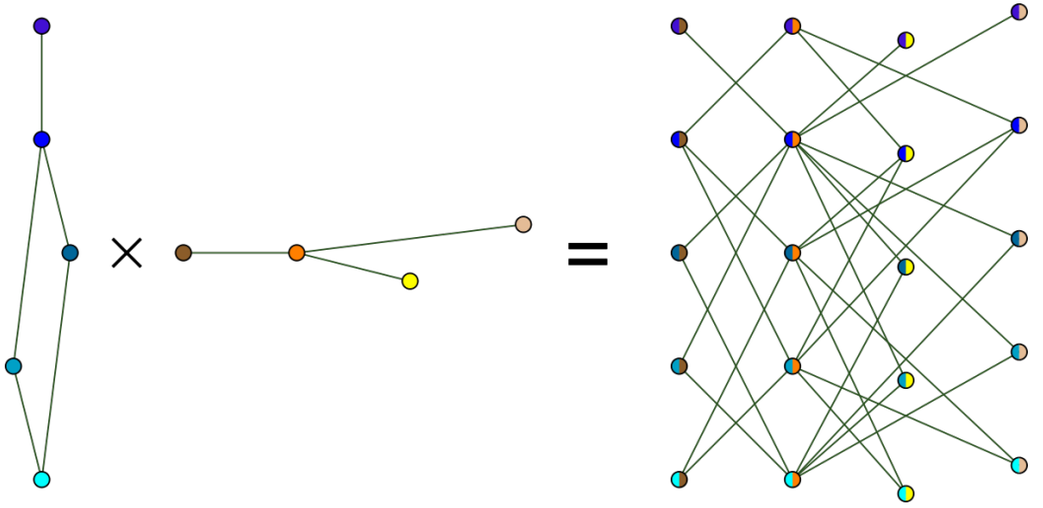
Tensor (3D)

Scalars, vectors, matrices and tensors

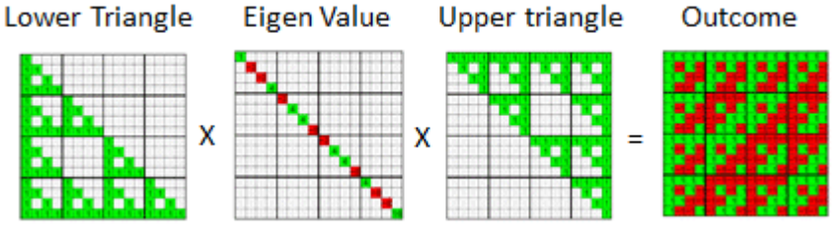
Tensor (4D)

Tensor operations

- Scaling
- Addition and subtraction
- Multiplication
- Contraction
- Tensor product
- Reshaping
- Transposition
- Eigenvectors and eigenvalues
- Cholesky decomposition
- …



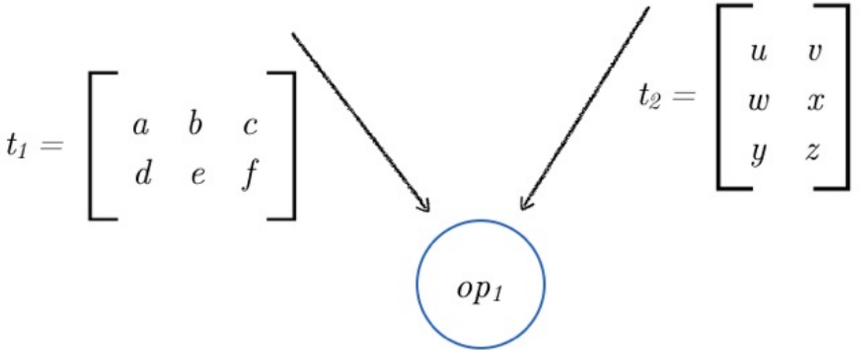
Source: http://bit.ly/2F7svG0



Source: http://bit.ly/2o8dyvy

Concept

- t_1 is a 2x3 matrix
- t_2 is a 3x2 matrix
- The op_1 is a multiplication operator

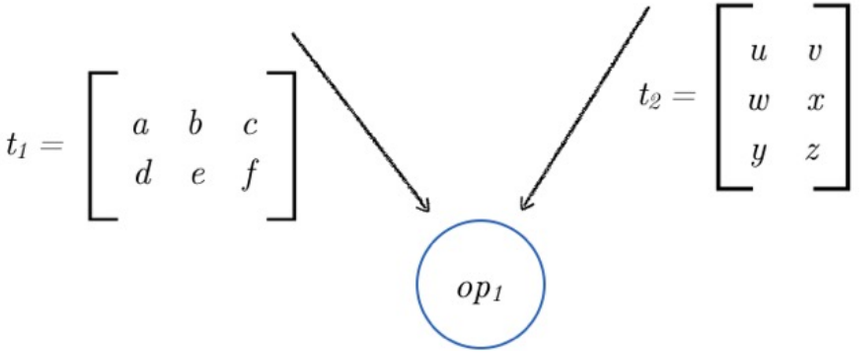


- The above can be represented in a matrix form

$$out = t_1 \underset{op\_1}{*} t_2 = \begin{bmatrix} au + bw + dy & av + bx + cz \\ du + ew + fy & dv + ex + fz \end{bmatrix}$$

Tensor product

- t_1 is a 2x3 matrix
- t_2 is a 3x2 matrix
- The op_1 is a tensor product operator



- The above can be represented in a matrix form

$$out = t_1 \underbrace{\otimes}_{op\_1} t_2 = \begin{bmatrix} at_2 & bt_2 & ct_2 \\ dt_2 & et_2 & ft_2 \end{bmatrix} = \begin{bmatrix} au & av & bu & bv & cu & cv \\ aw & ax & bw & bx & cw & cx \\ ay & az & by & bz & cy & cz \\ du & dv & eu & bv & fu & fv \\ dw & dx & ew & bx & fw & fx \\ dy & dz & qy & bz & fy & fz \end{bmatrix}$$

Neural network as a graph

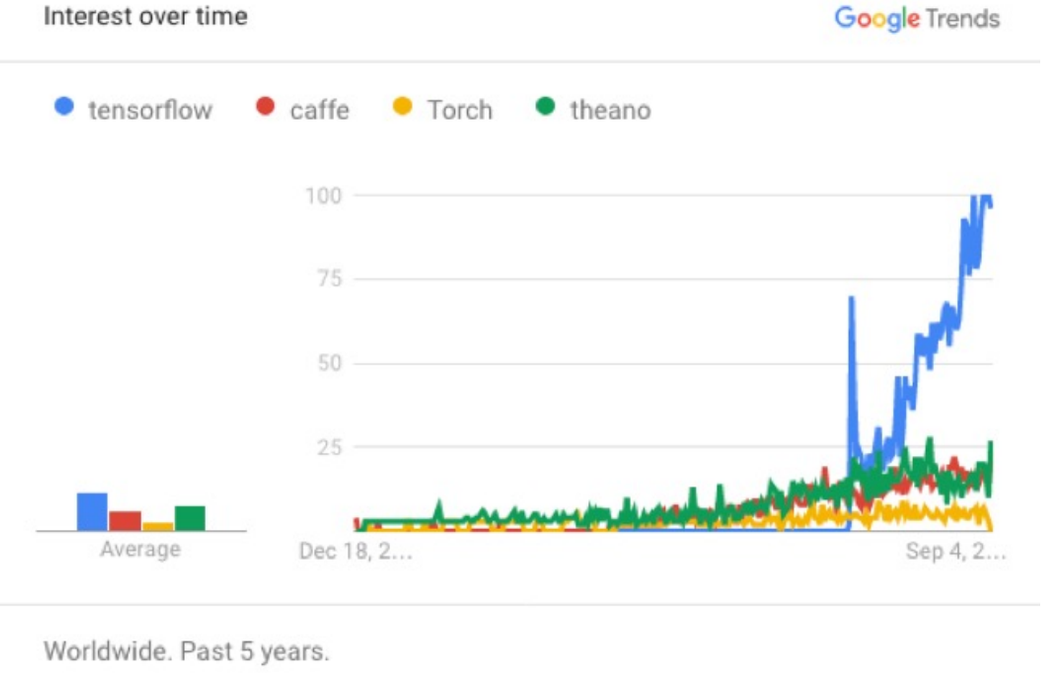
TensorFlow

Technology from Google

source: http://bit.ly/2itbqJT

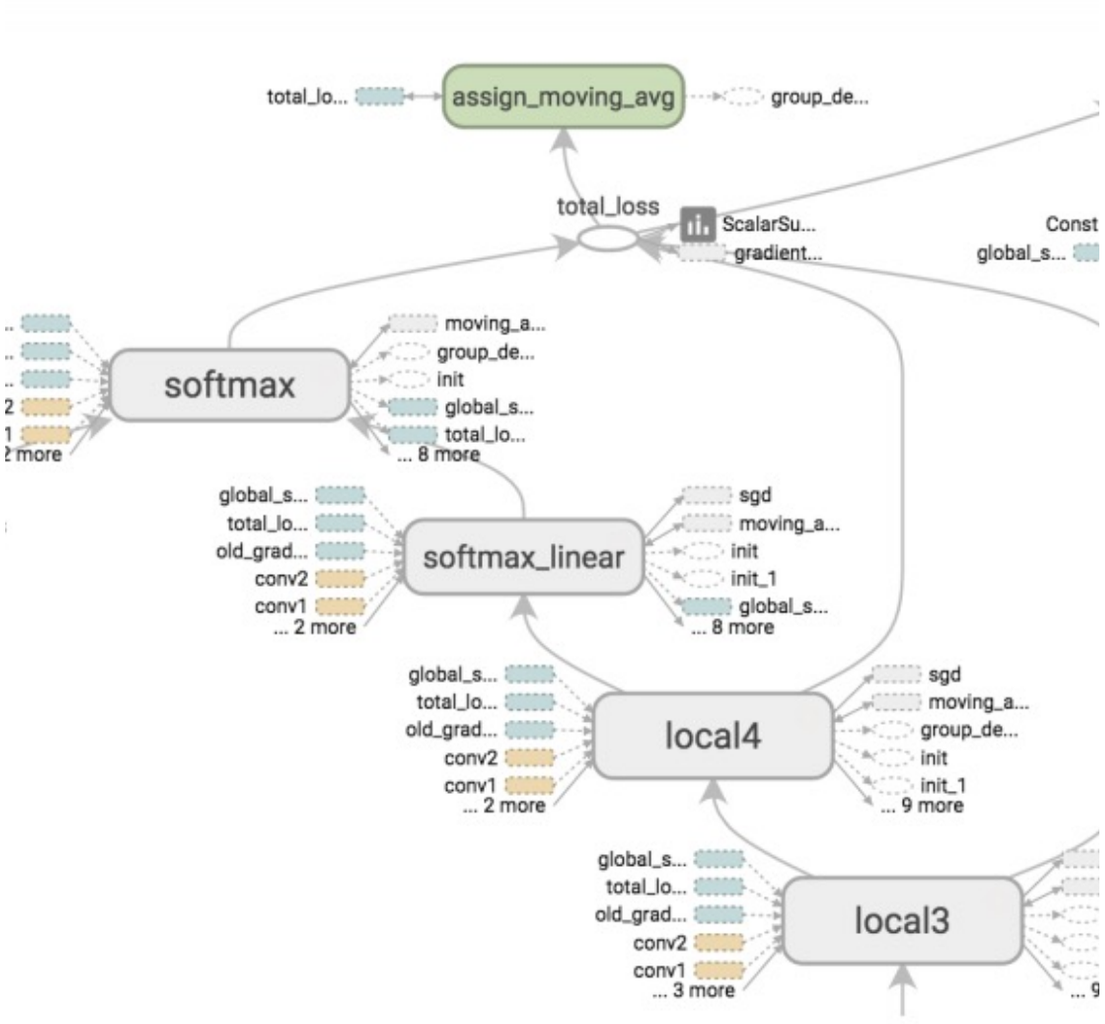
What is TensorFlow

- Open source software library for numerical computation using data flow graphs
- Based loosely on neural networks
- Built in C++ with a Python interface
- Quickly gained high popularity
- Supports GPU computations



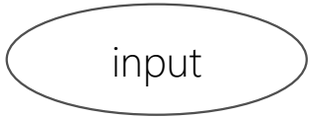
Data Flow Graph

- Nodes represent mathematical operations
- Edges are multidimensional arrays (tensors)
- Thus, TensorFlow is a flow of multidimensional arrays (tensors) between the mathematical operations (nodes)
- i.e. flow of *tensors*



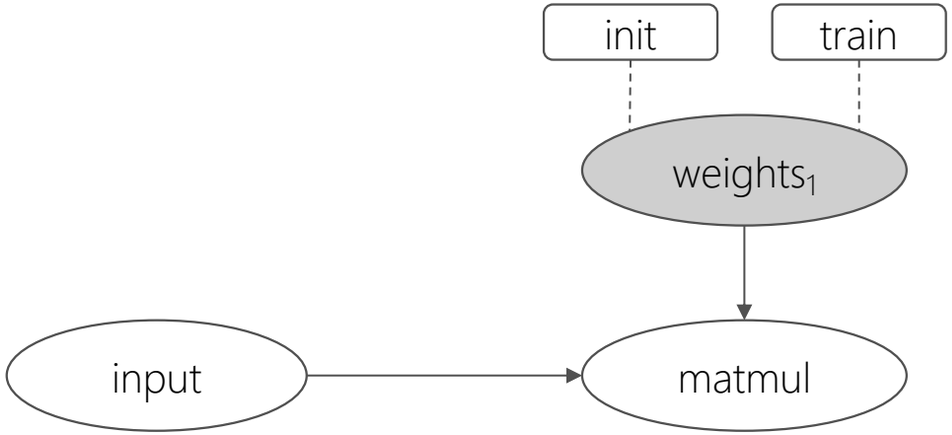
Step by step DAG: input

$$y = x$$



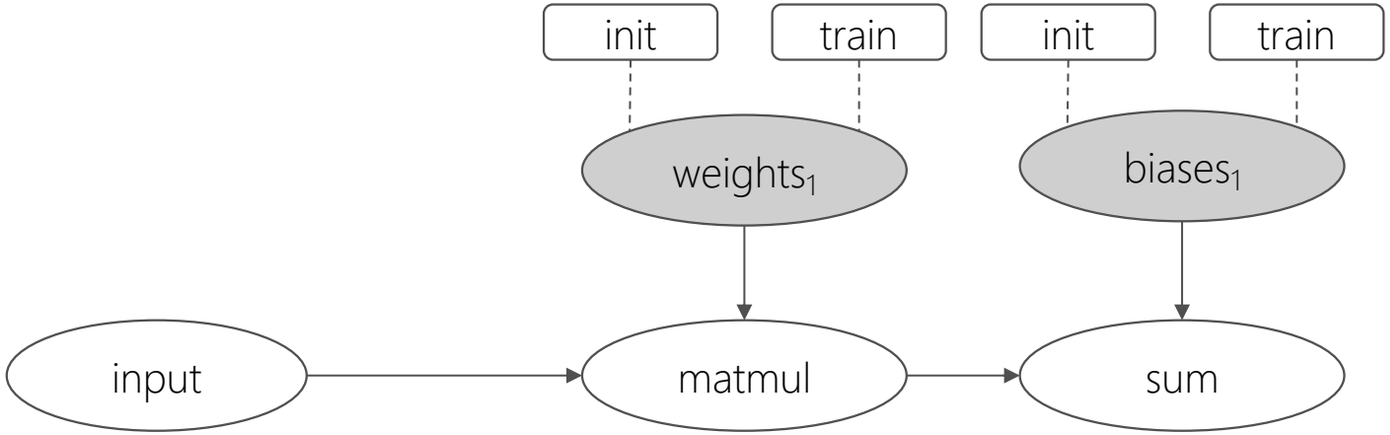
Step by step DAG: input

$$y = wx$$



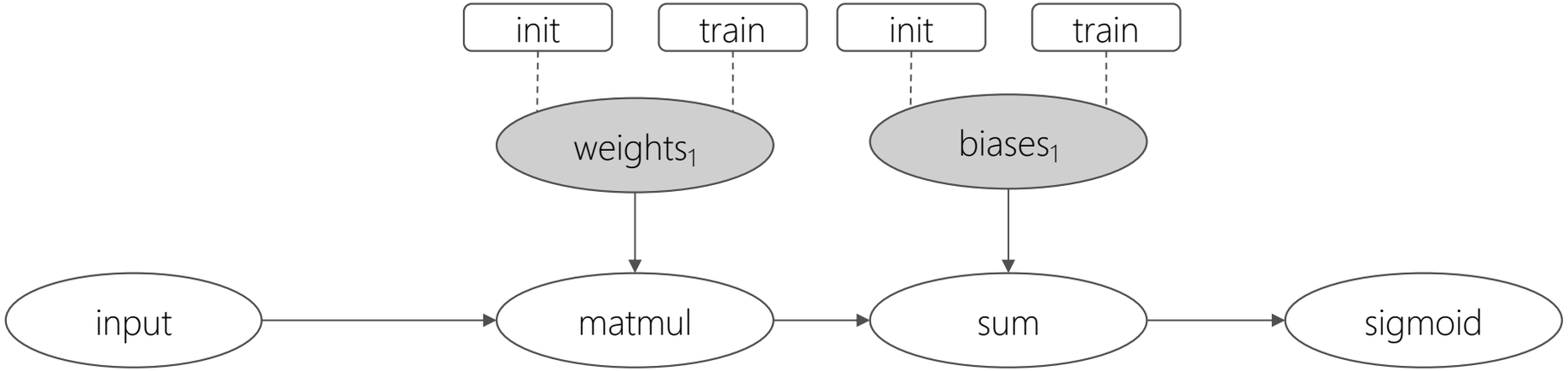
Step by step DAG: bias addition

$$y = wx + b$$



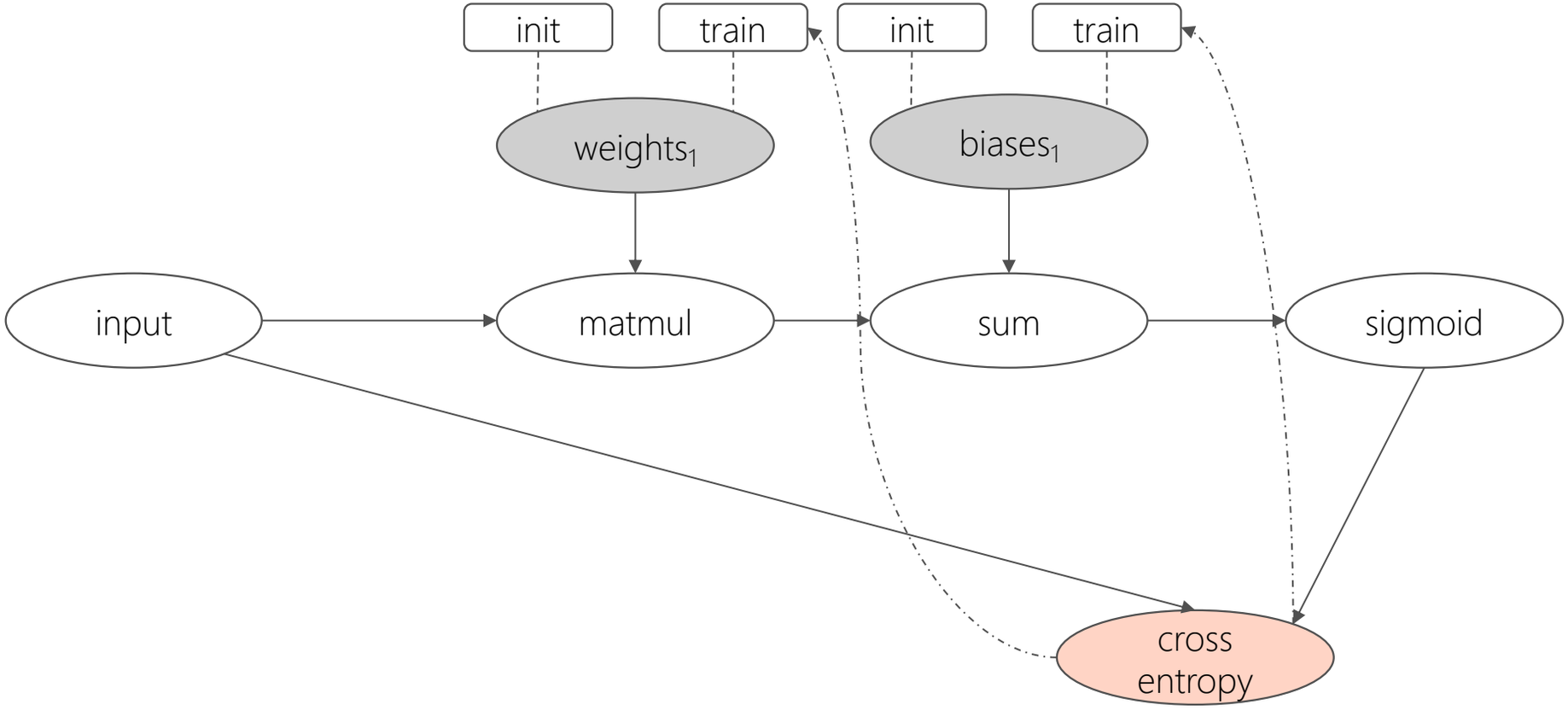
Step by step DAG: activation function

$$y = \frac{1}{1 + e^{-wx-b}}$$



Step by step DAG: cross entropy

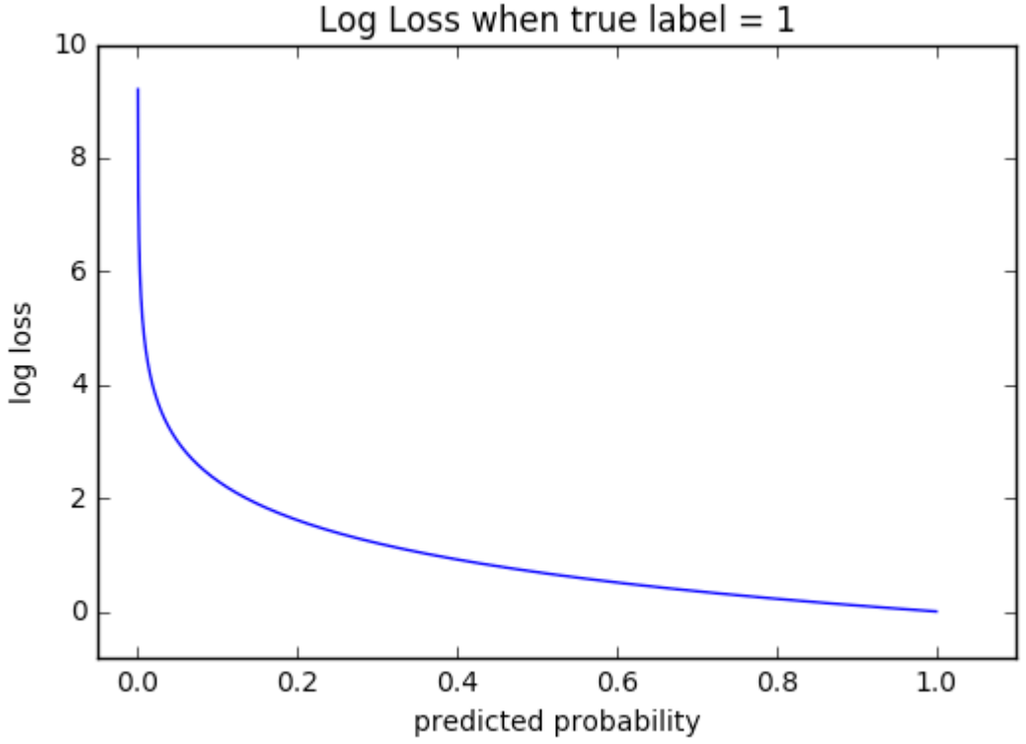
$$y = \frac{1}{1 + e^{-wx-b}}$$



$$loss = -(\check{y}\log(y) + (1-\check{y})\log(1-y))$$

Cross entropy

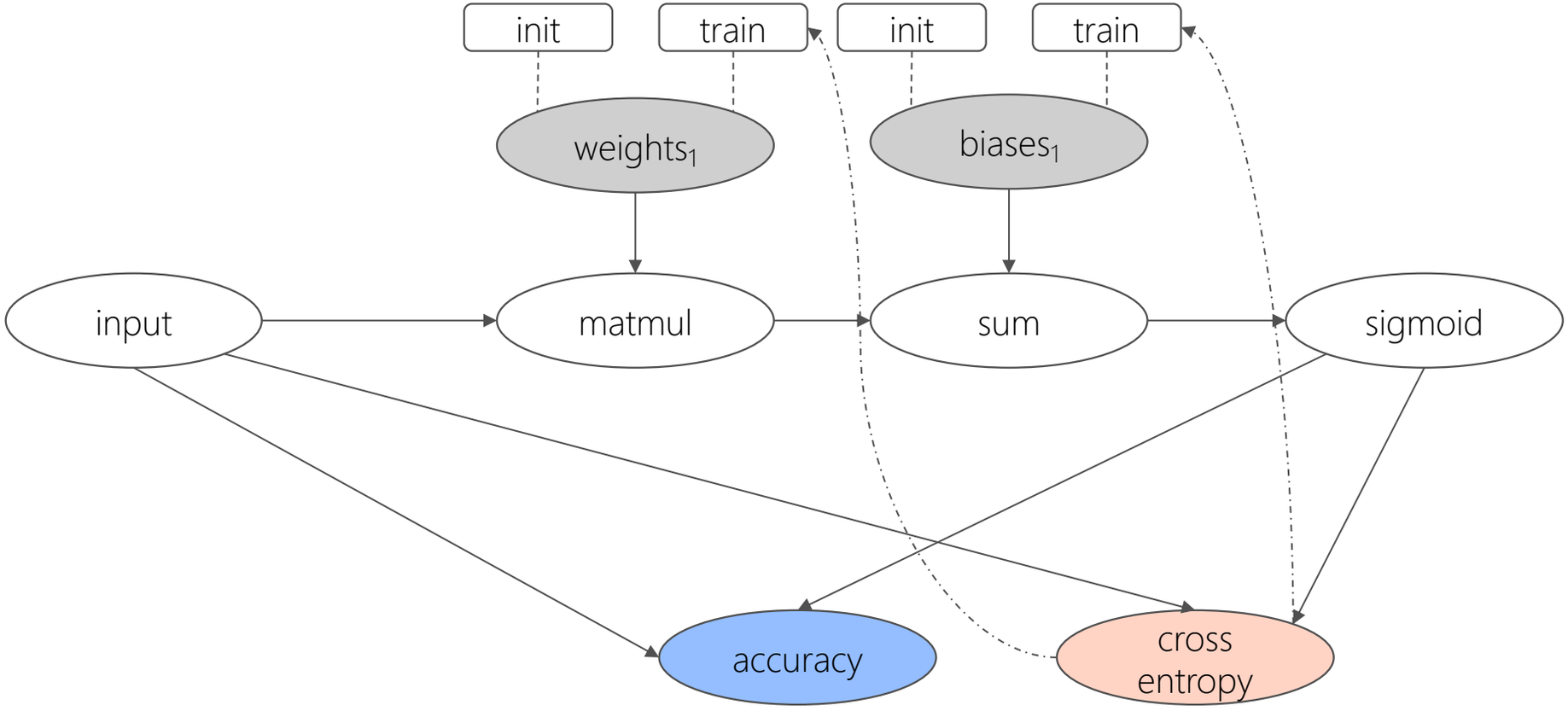
$$loss = -(\check{y}\log(y) + (1-\check{y})\log(1-y))$$



Source: http://bit.ly/2HhbJoG

Step by step DAG: accuracy

$$y = \frac{1}{1 + e^{-wx-b}}$$

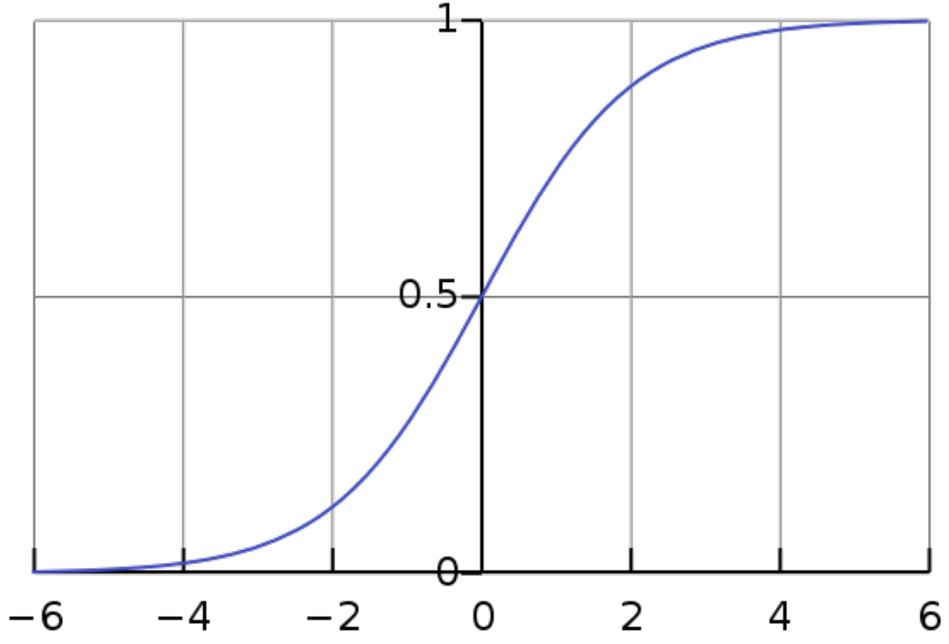


$$loss = -(\check{y}\log(y) + (1-\check{y})\log(1-y))$$

Activation functions

Sigmoid Function

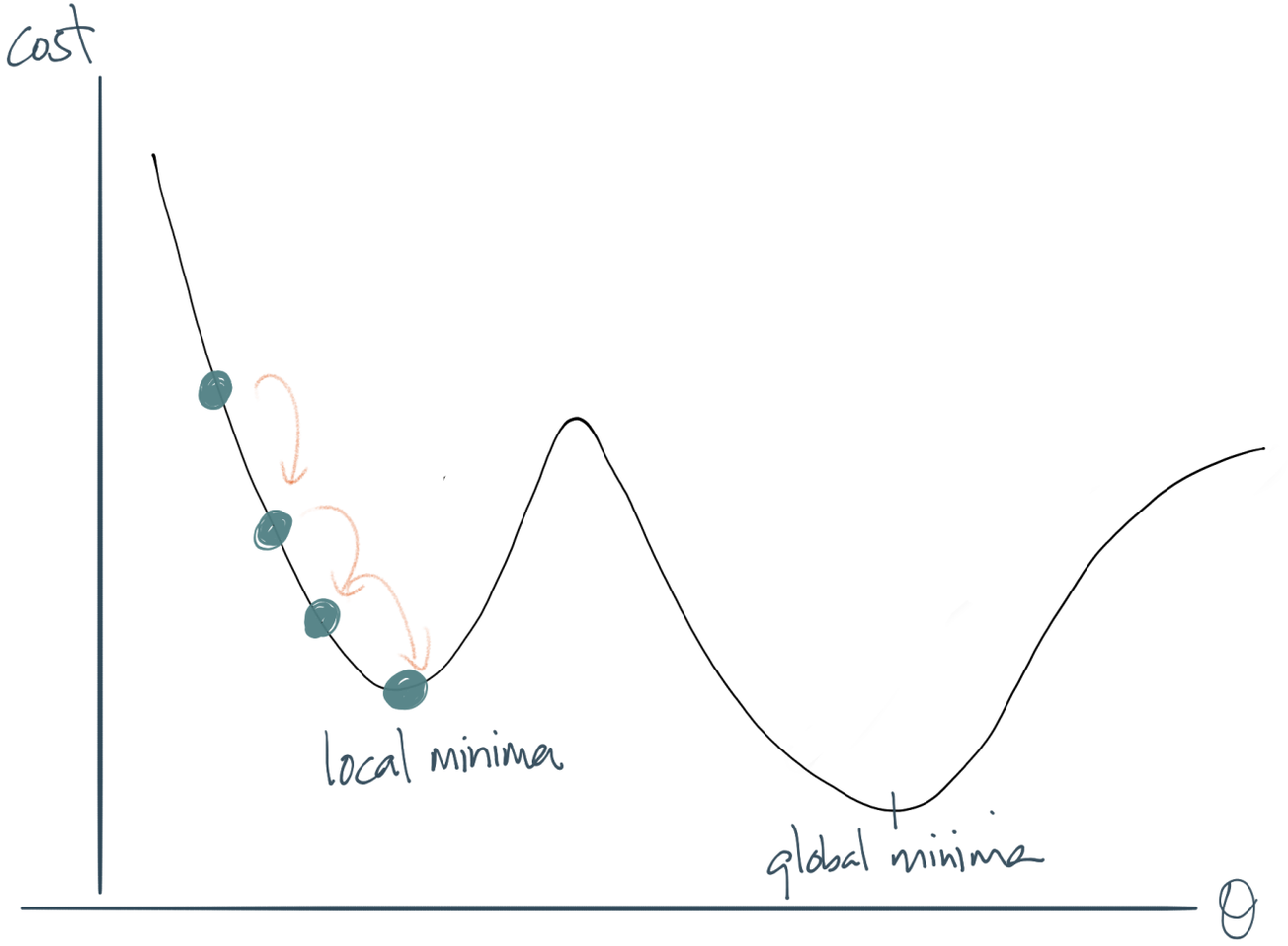
$$f(x) = \sigma(z) \equiv \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$



Source: http://bit.ly/2GgMbGW

- Sigmoid non-linearity squashes real numbers between [0, 1]
- *Historically* nice interpretation of neuron firing rate (i.e. not firing at all (0) to fully-saturated firing (1)).

Gradient Descent

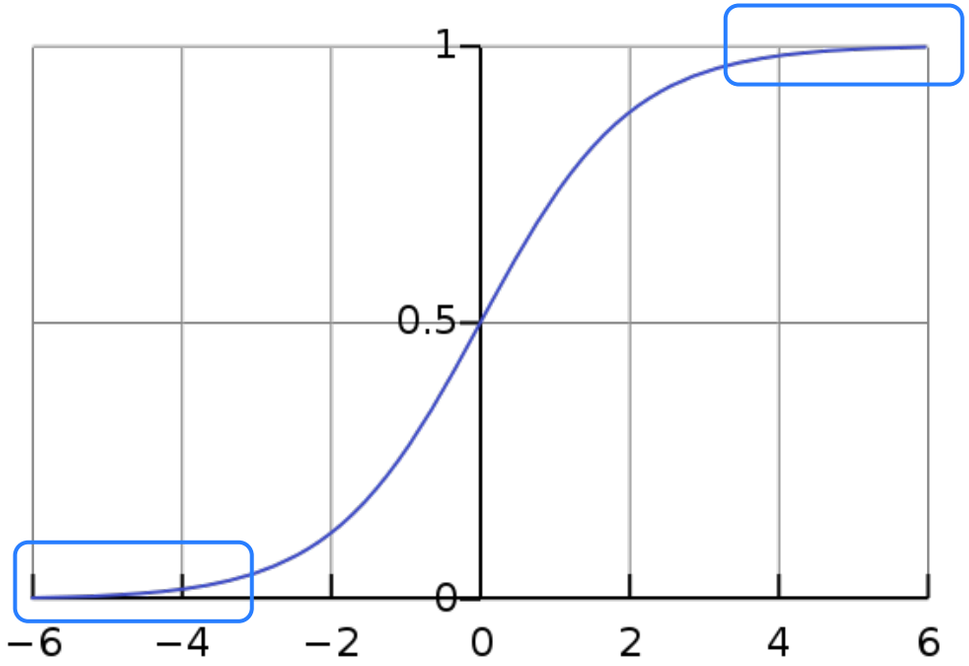


Cost function: what you want to minimize

Gradient descent: finding minimum of a function of multiple variables

Parameters like step size and learning rate are important to escape local minima

Sigmoid function (continued)



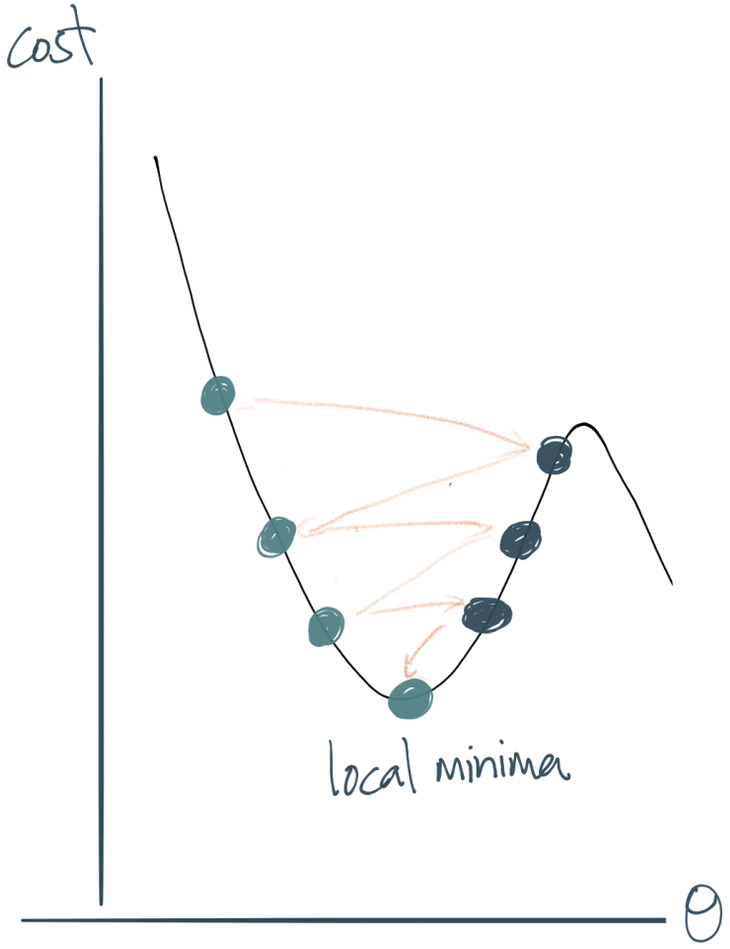
Source: http://bit.ly/2GgMbGW

Sigmoid function is not used as much b/c:

- Saturate and kill gradients: really large values are too close to 0 or 1, gradients are close to 0 (i.e. parallel to x-axis) thus stopping back propagation

- Output is not zero-centered: During gradient descent, if all values are positive then during back propagation the weights will become all positive or all negative creating zig zagging dynamics.

Sigmoid function (continued)

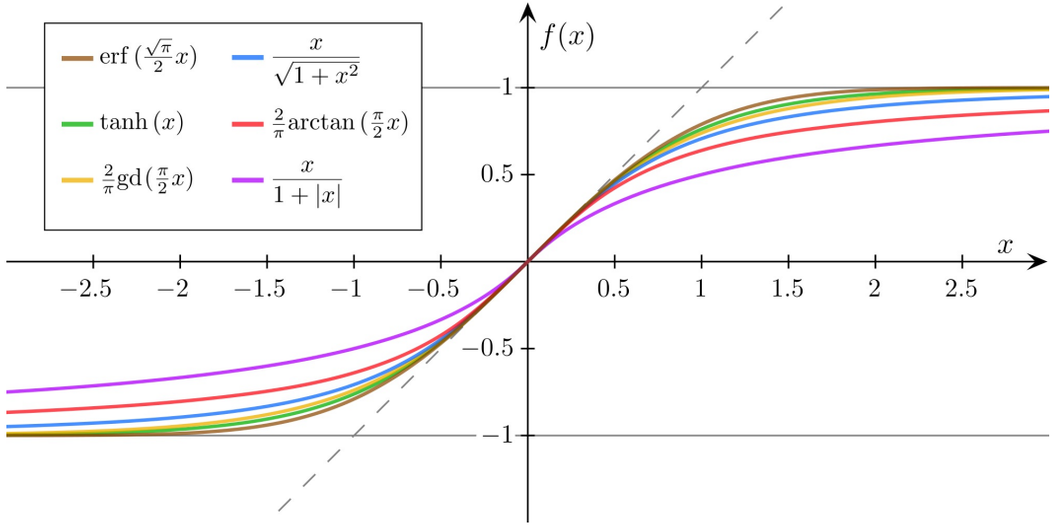


Sigmoid function is not used as much b/c:

- Saturate and kill gradients: really large values are too close to 0 or 1, gradients are close to 0 (i.e. parallel to x-axis) thus stopping back propagation

- Output is not zero-centered: During gradient descent, if all values are positive then during back propagation the weights will become all positive or all negative creating zig zagging dynamics.

Tanh function

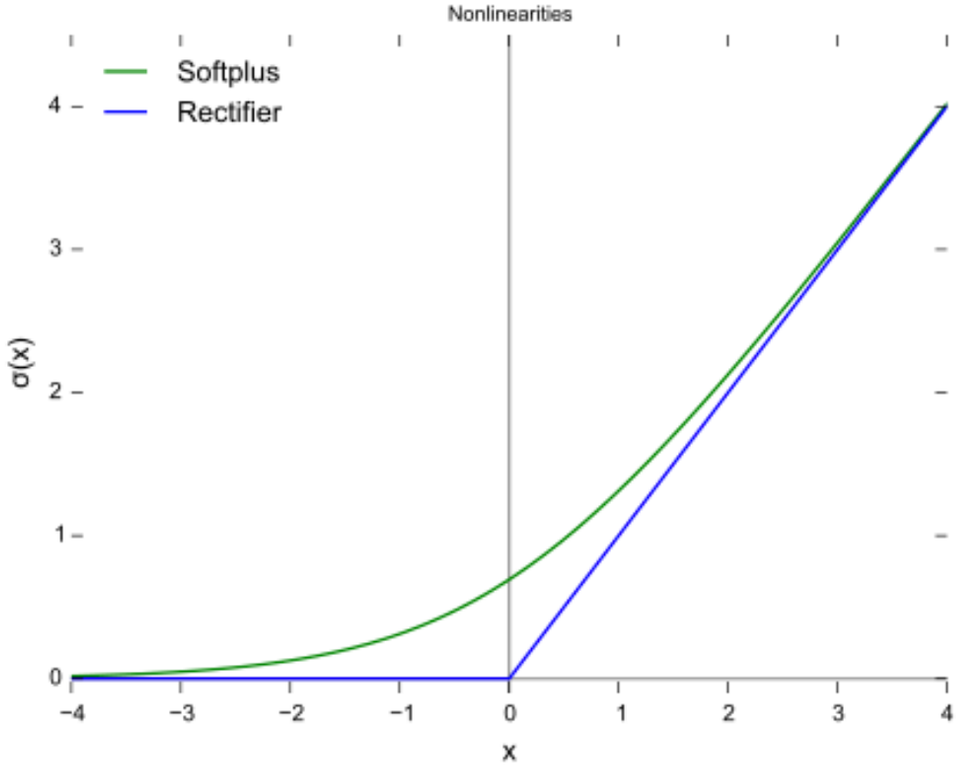


Source: http://bit.ly/2C4y89z

Tanh function squashes real numbers [-1, 1]

- Same problem as sigmoid that its activations saturate thus killing gradients.
- But it is zero-centered minimizing the zig zagging dynamics during gradient descent.
- Currently preferred *sigmoid nonlinearity*

ReLU: Rectifier Linear Unit



CC0, http://bit.ly/2EwRnds

ReLU's activation is thresholded at zero

$$f(z) = \max(0, u)$$

- Quite popular over the last few years
- Speeds up SGD convergence
- It is easier to implement due to simpler mathematical functions
- Sensitive to high learning rate during training resulting in "dead" neurons (i.e. neurons that will not activate across the entire dataset).

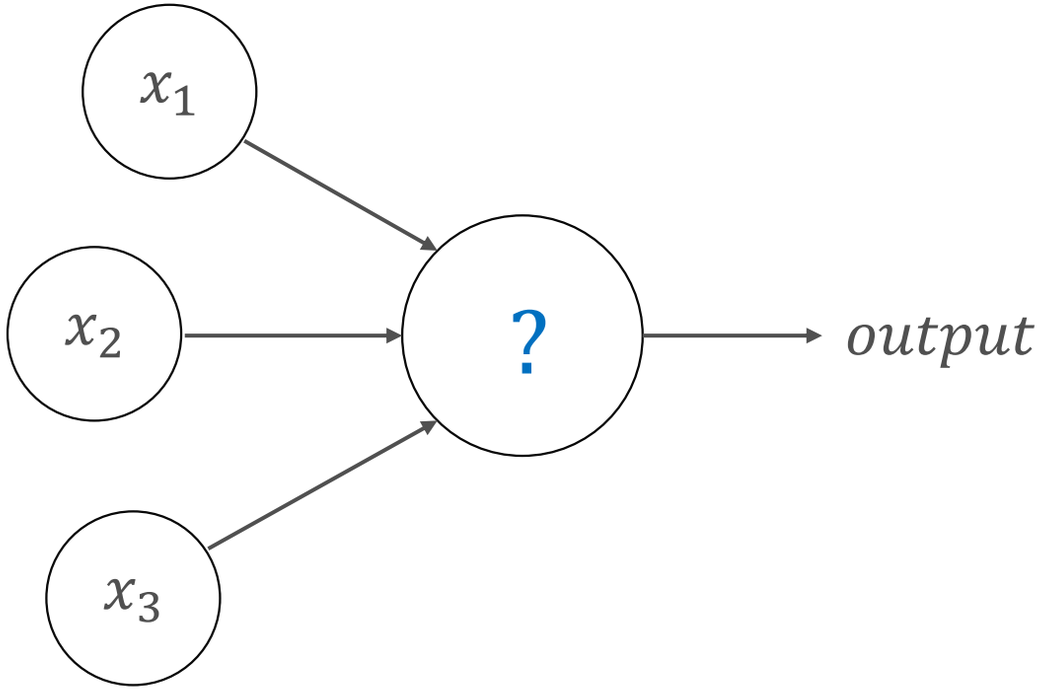
The Immense Power of Simple Structures

Resources:

- Andrej Karpathy's CS231N Neural Networks
- Steve Miller's How to build a neural network
- Michael Nielsen's Neural Networks and Deep Learning, Chapter 1

Perceptron

Simplified (binary) artificial neuron



$Do\ I\ snowboard\ at\ Whistler\ this\ weekend?$

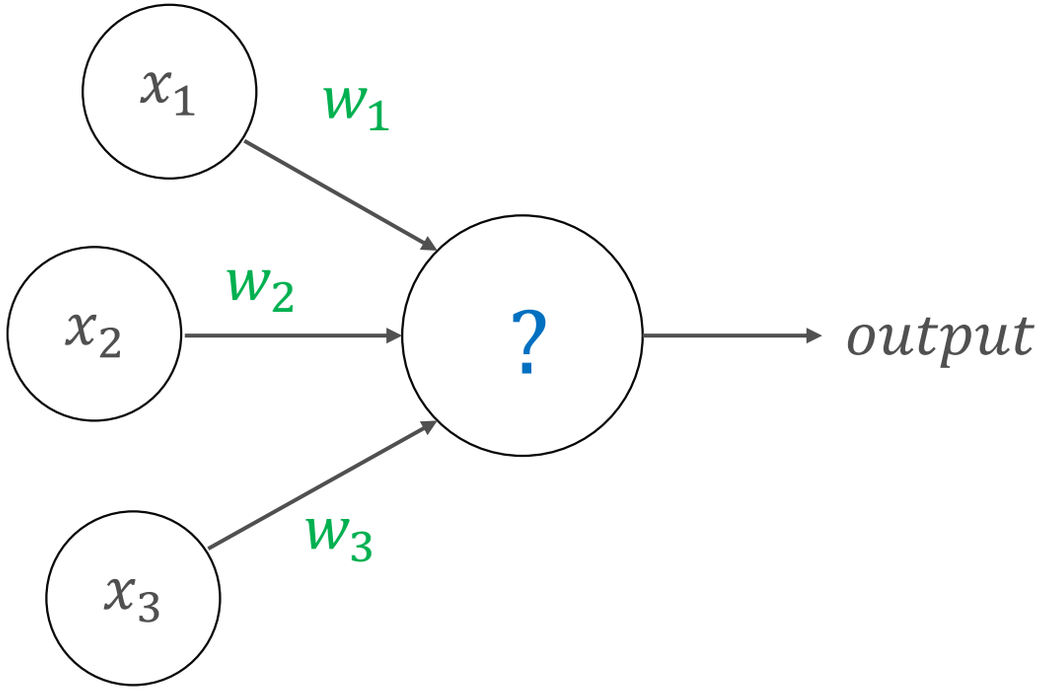
$x_1 \rightarrow Is\ the\ weather\ good?$

$x_2 \rightarrow Is\ the\ powder\ good?$

$x_3 \rightarrow Am\ I\ in\ the\ mood\ to\ drive?$

Perceptron

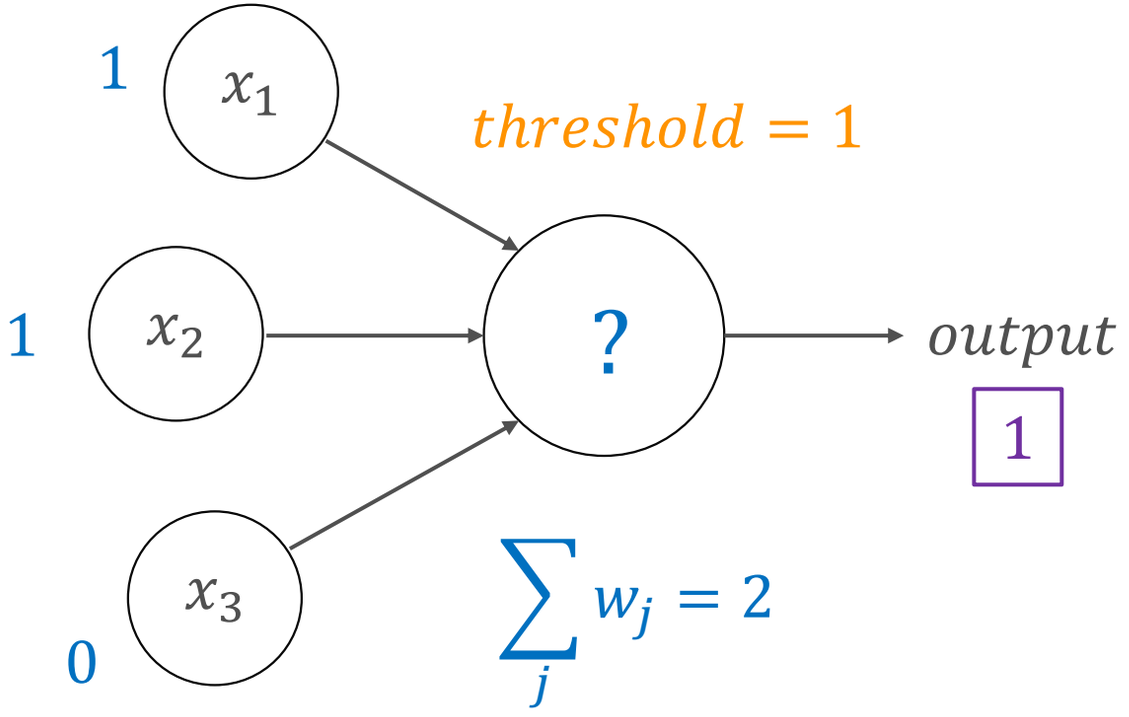
Simplified (binary) artificial neuron with weights



$$output = \begin{cases} 0, & \sum_{j=0}^{n} w_j x_j \leq threshold \\ 1, & \sum_{j=0}^{n} w_j x_j > threshold \end{cases}$$

Perceptron

Simplified (binary) artificial neuron; *no weights*



$Do\ I\ snowboard\ at\ Whistler\ this\ weekend?$

$x_1 = 1\ (good\ weather)$

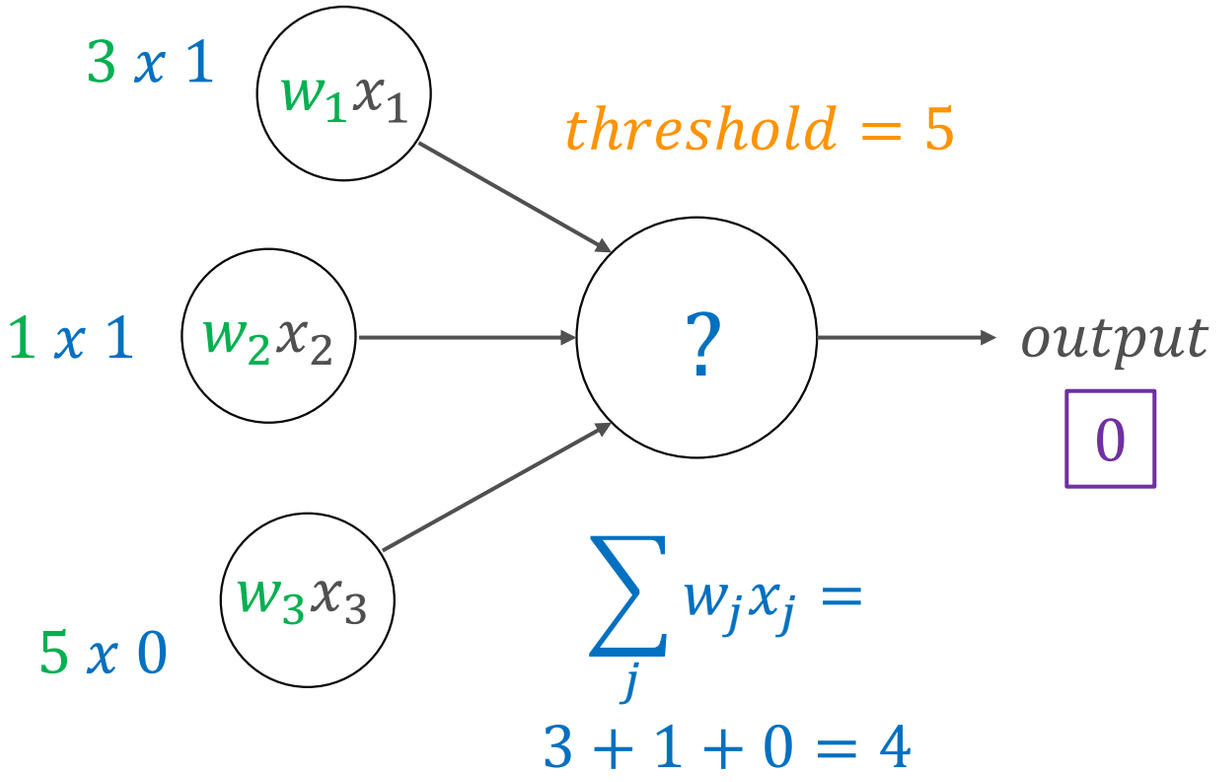
$x_2 = 1\ (a\ lot\ of\ powder)$

$x_3 = 0\ (driving\ sucks)$

Perceptron

Simplified (binary) artificial neuron; *add weights*

Persona: Après-ski'er



Do I snowboard at Whistler this weekend?

$x_1 = 1$ (*good weather*)
$w_1 = 3$

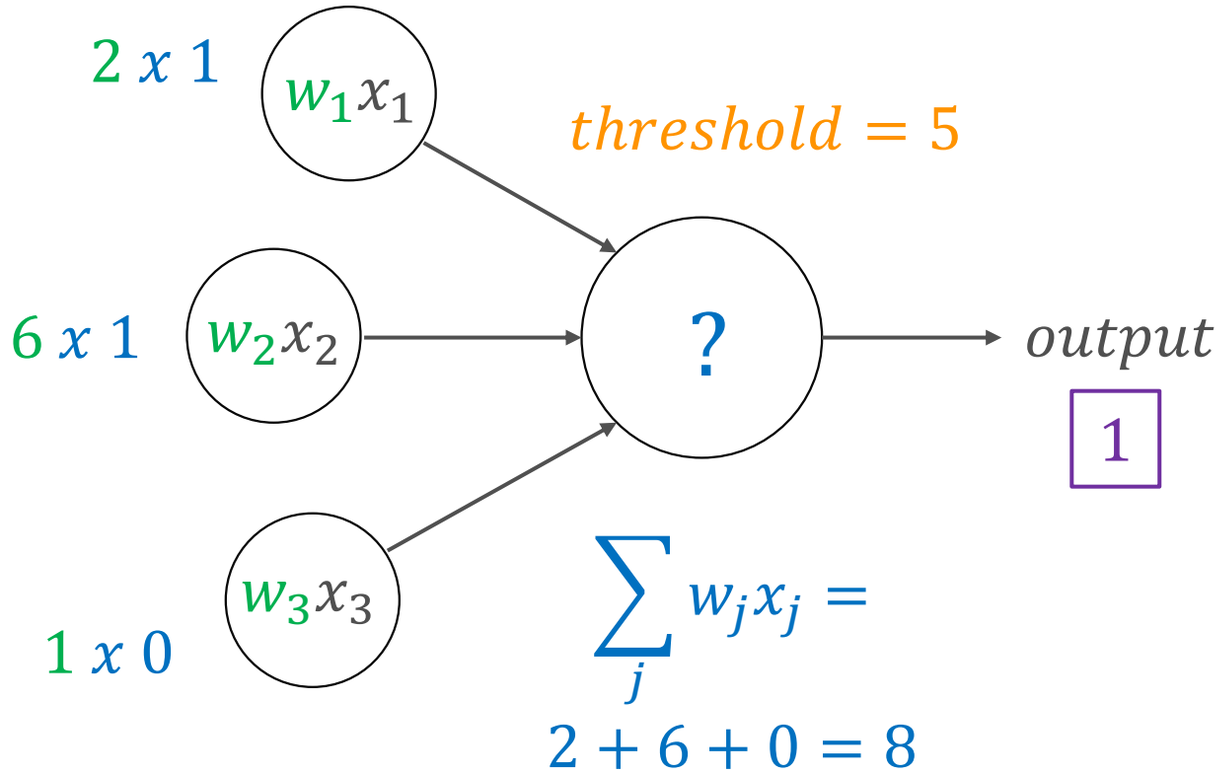
$x_2 = 1$ (*a lot of powder*)
$w_2 = 1$

$x_3 = 0$ (*driving sucks*)
$w_3 = 5$

Perceptron

Simplified (binary) artificial neuron; *add weights*

Persona: Shredder



Do I snowboard at Whistler this weekend?

$x_1 = 1$ (*good weather*)
$w_1 = 2$

$x_2 = 1$ (*a lot of powder*)
$w_2 = 6$

$x_3 = 0$ (*driving sucks*)
$w_3 = 1$

Introducing Bias

Perceptron needs to take into account of bias

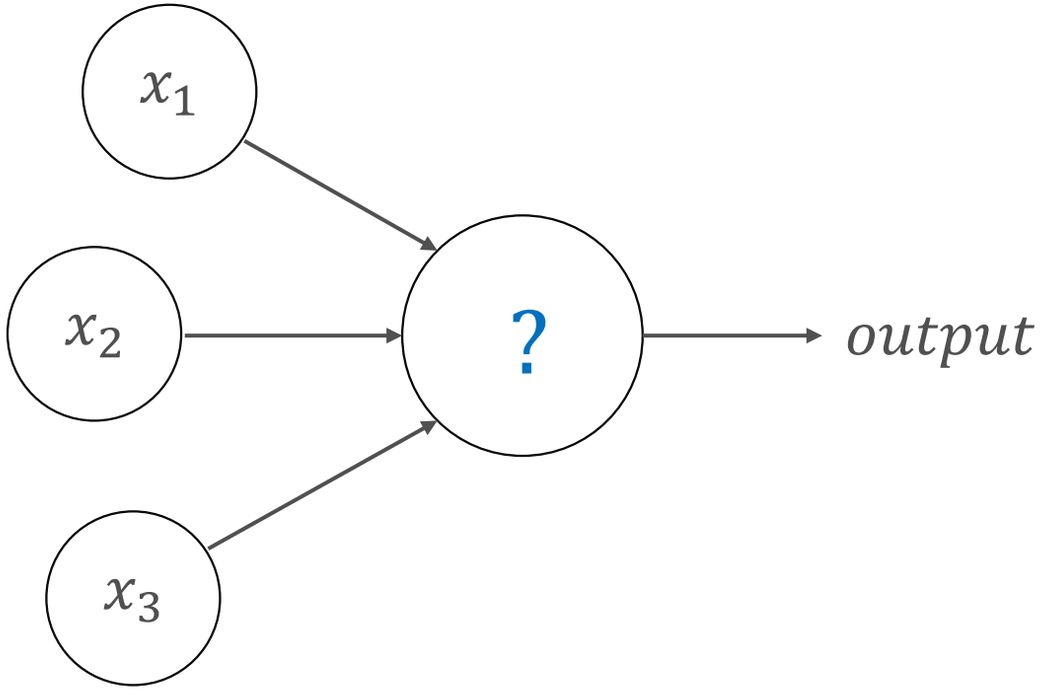
$$output = \begin{cases} 0, & wx + b \leq 0 \\ 1, & wx + b > 0 \end{cases}$$

where b is how easy it is to get the perceptron to fire

e.g. Shredder has a strong positive bias to go to Whistler
while Après-Ski'er bias is not as strong

Sigmoid Neuron

The more common artificial neuron

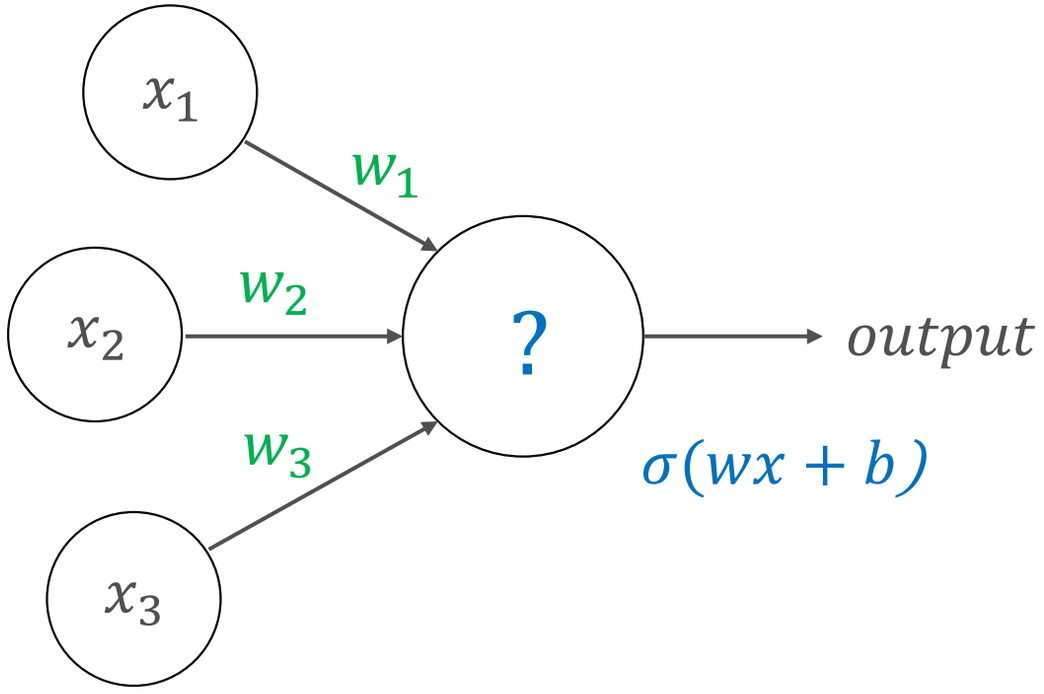


Instead of [0, 1], now (0…1)

Where output is defined by $\sigma(wx + b)$

Sigmoid Neuron

The more common artificial neuron

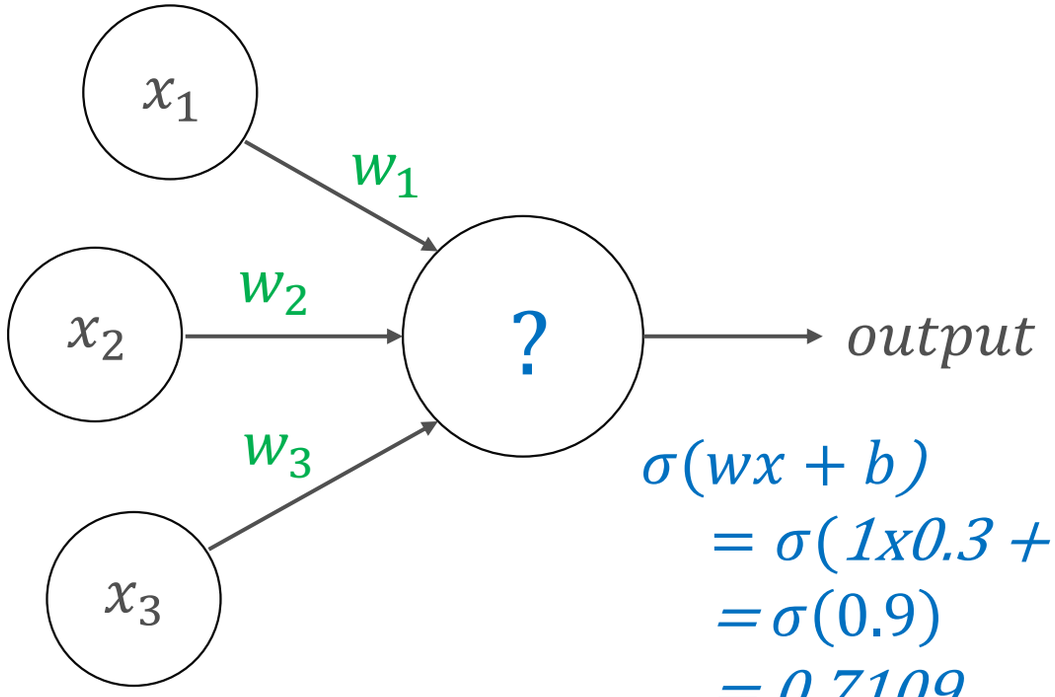


Instead of [0, 1], now (0…1)

Where output is defined by $\sigma(wx + b)$

Sigmoid Neuron

Persona: Shredder



$\sigma(wx + b)$
$= \sigma(1x0.3 + 1x0.6 + 0x\ 0.1)$
$= \sigma(0.9)$
$= 0.7109$

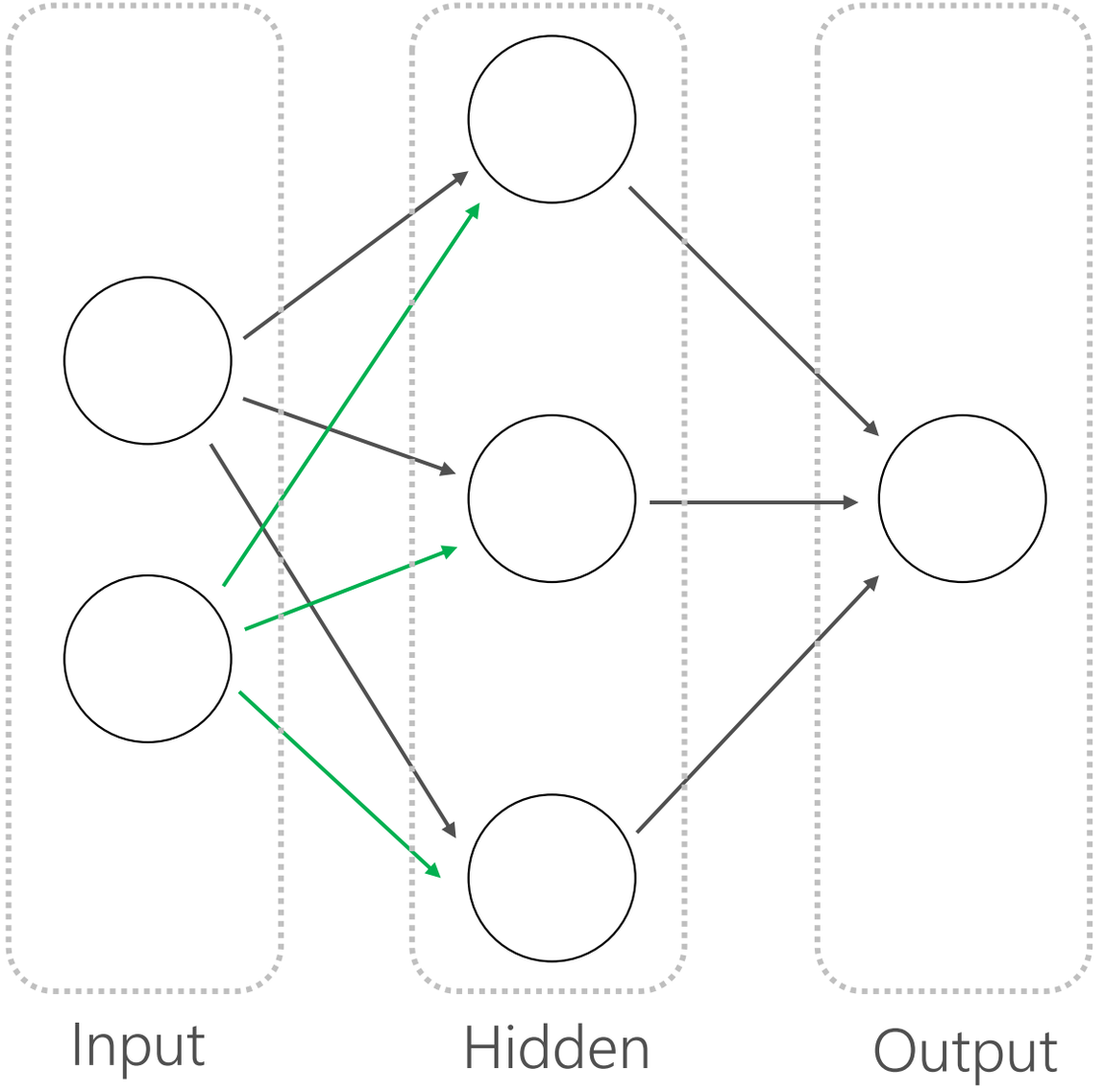
Do I snowboard at Whistler this weekend?

$x_1 = 1$ $(good\ weather)$
$w_1 = 0.3$

$x_2 = 1$ $(a\ lot\ of\ powder)$
$w_2 = 0.6$

$x_3 = 0$ $(driving\ sucks)$
$w_3 = 0.1$

Simplified Two-Layer ANN

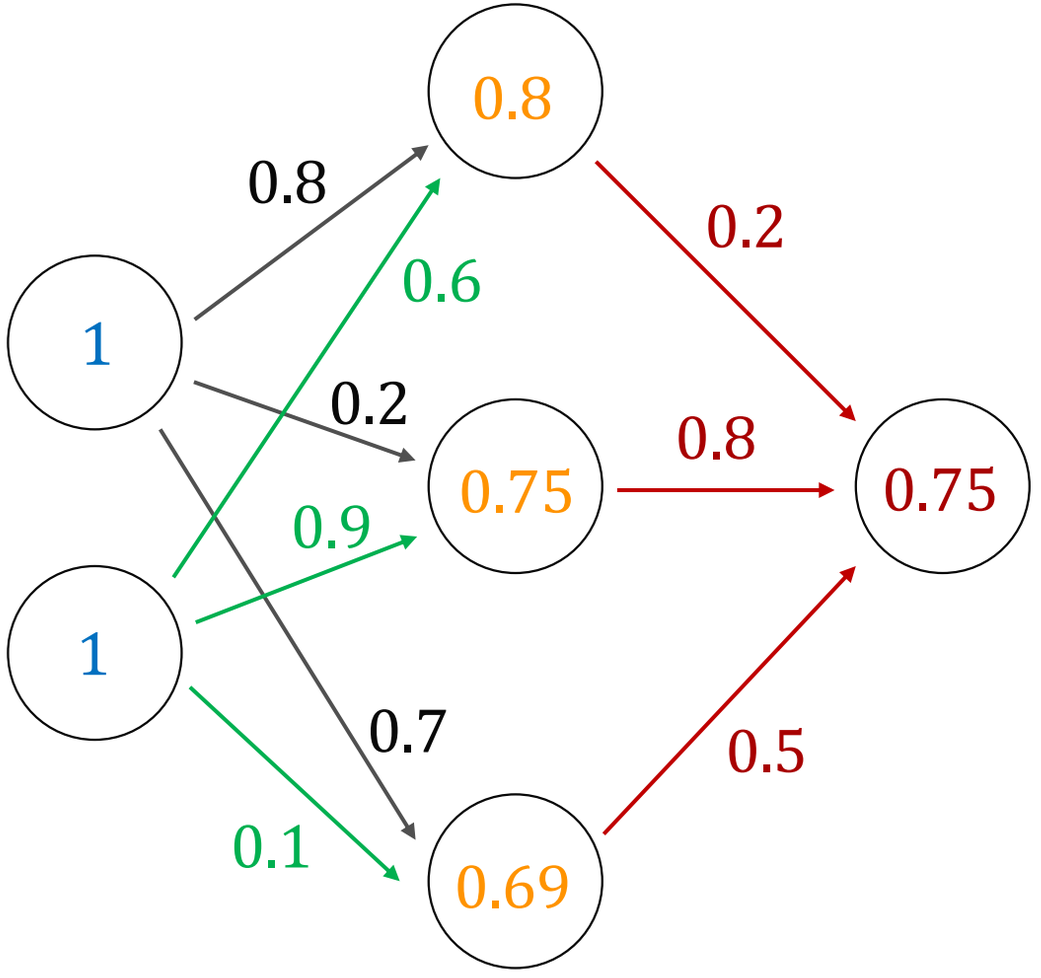


$Do\ I\ snowboard\ at\ Whistler\ this\ weekend?$

$x_1 = Après\ Ski'er$
$x_2 = Shredder$

$h_1 = weather$
$h_2 = powder$
$h_3 = driving$

Simplified Two-Layer ANN



$h_1 = \sigma(1x0.8 + 1x0.6) = 0.80$
$h_2 = \sigma(1x0.2 + 1x0.9) = 0.75$
$h_3 = \sigma(1x0.7 + 1x0.1) = 0.69$

$output$
$= \sigma(0.2x0.8 +$
$0.8x0.75 +$
$0.5x0.69)$
$= \sigma(1.105) = 0.75$

Great References

Andrej Karparthy's ConvNetJS MNIST Demo

What is back propagation in neural networks?

CS231n: Convolutional Neural Networks for Visual Recognition
Syllabus and Slides | Course Notes | YouTube

- With particular focus on CS231n: Lecture 7: Convolution Neural Networks

Neural Networks and Deep Learning

TensorFlow

Great References

Deep Visualization Toolbox

Back Propagation with TensorFlow

TensorFrames: Google TensorFlow with Apache Spark

Integrating deep learning libraries with Apache Spark

Upcoming webinar: Build, Scale, and Deploy Deep Learning Pipelines with Ease

Microsoft

Q&A