Intro to PyTorch

A Simple and Powerful Deep Learning Framework

Download this slide and workshop from the URL below:

https://tinyurl.com/intro-to-pytorch

https://tinyurl.com/intro-to-pytorch-workshop

Wi-Fi SSID: !(^_^)!Instructor

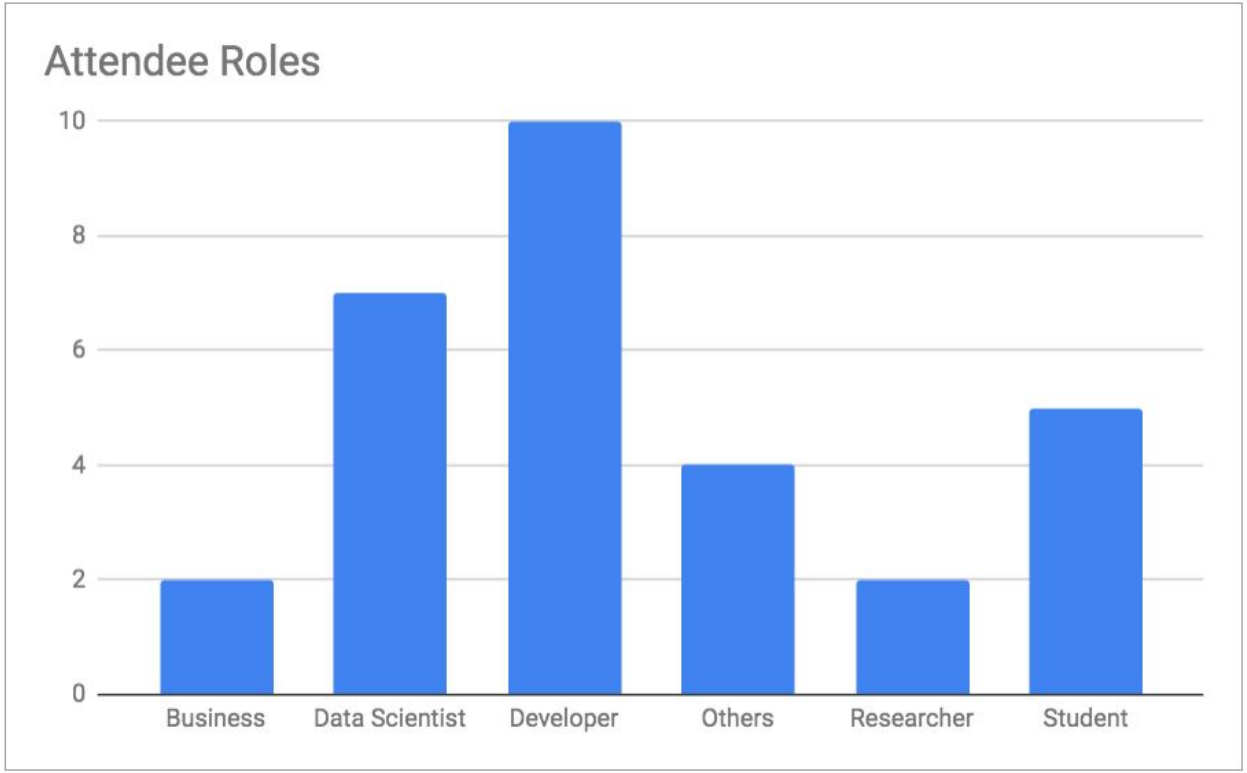
Password: iamprontotools

Hash Tag: #DevCBKK #PyTorchThailand

Feel free to grab any drinks, coffee, and ice cream. Also, please take a photo at the booth :)

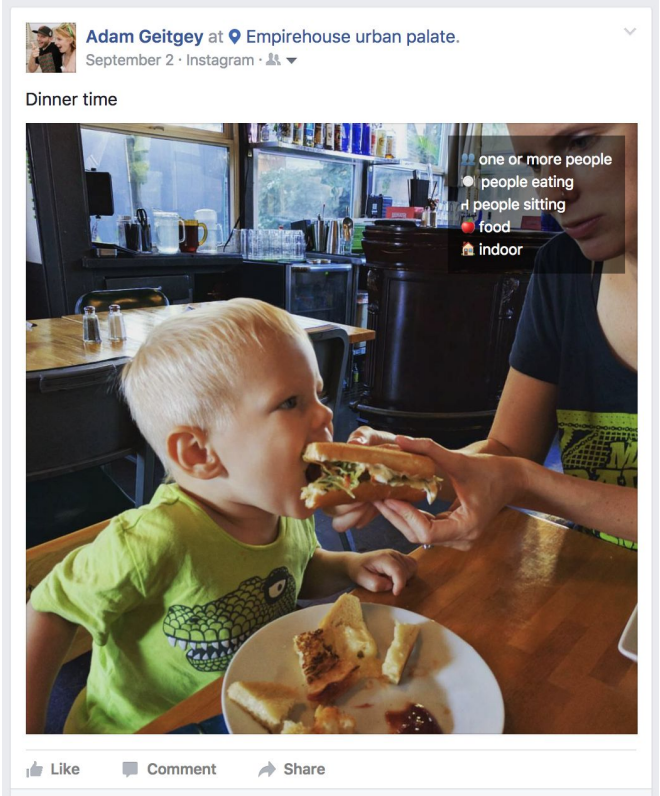
- **Name:** Teera Laiteerapong
- **Position:** Technical Training Lead at TechTalkThai
- **Education:**
  - Computer Engineering (B.Eng.), KMITL
  - Computer Science (M. Eng.), AIT
- **Research Interests:**
  - Machine Learning in Computer Vision
  - Medical Imaging, Civil Infrastructure Imaging

https://github.com/mrteera


Attendee Roles
10
8
6
4
2
0
Business
Data Scientist
Developer
Others
Researcher
Student

Deep Learning

Facebook Automatic Alternative Text



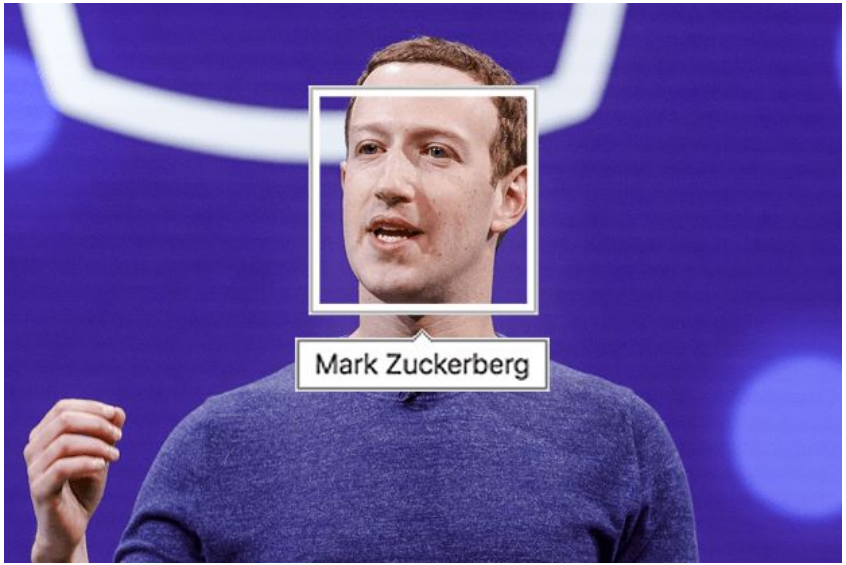
Source: https://github.com/ageitgey/show-facebook-computer-vision-tags,
https://www.theverge.com/2016/4/5/11364914/facebook-automatic-alt-tags-blind-visually-impared

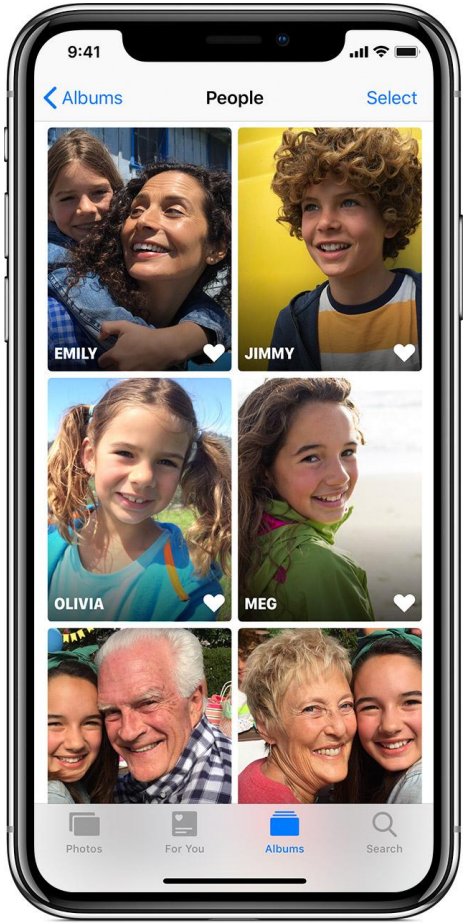
Vision AI for the Blind and Visually Impaired

AiPoly

Source: https://www.aipoly.com/

Face Recognition





Source: https://www.digitaltrends.com/computing/facebook-facial-recognition-privacy/, https://support.apple.com/en-us/HT207103

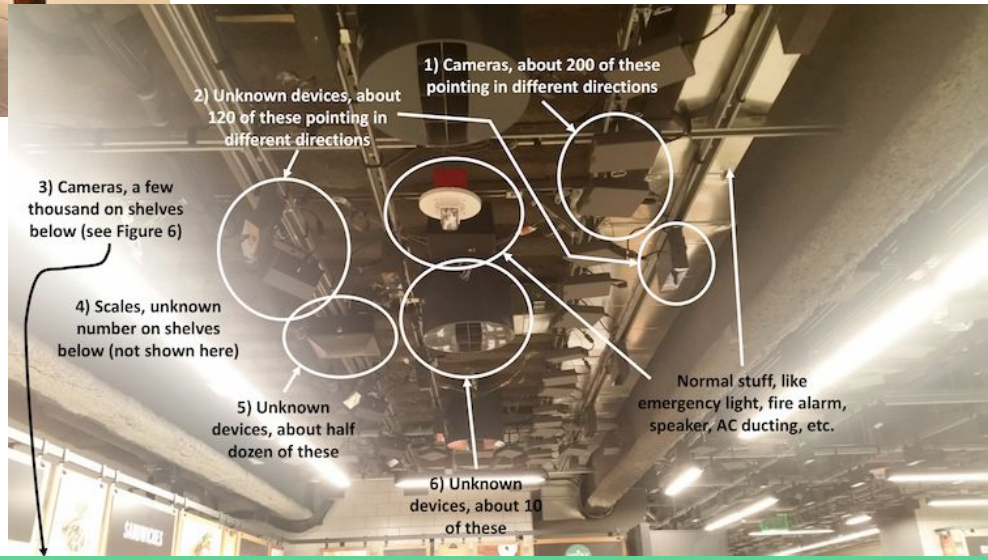
Face Recognition (2)

Source: https://www.omise.co/bringing-artificial-intelligence-out-of-the-lab-introducing-omise-facepay

Face Recognition (3)

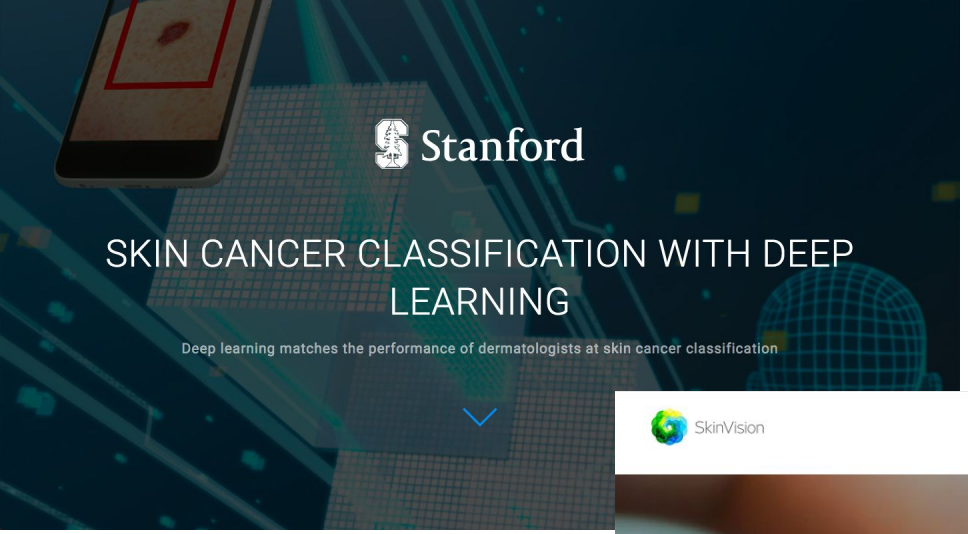
Source: https://techsauce.co/news/7-eleven-face-recognition/

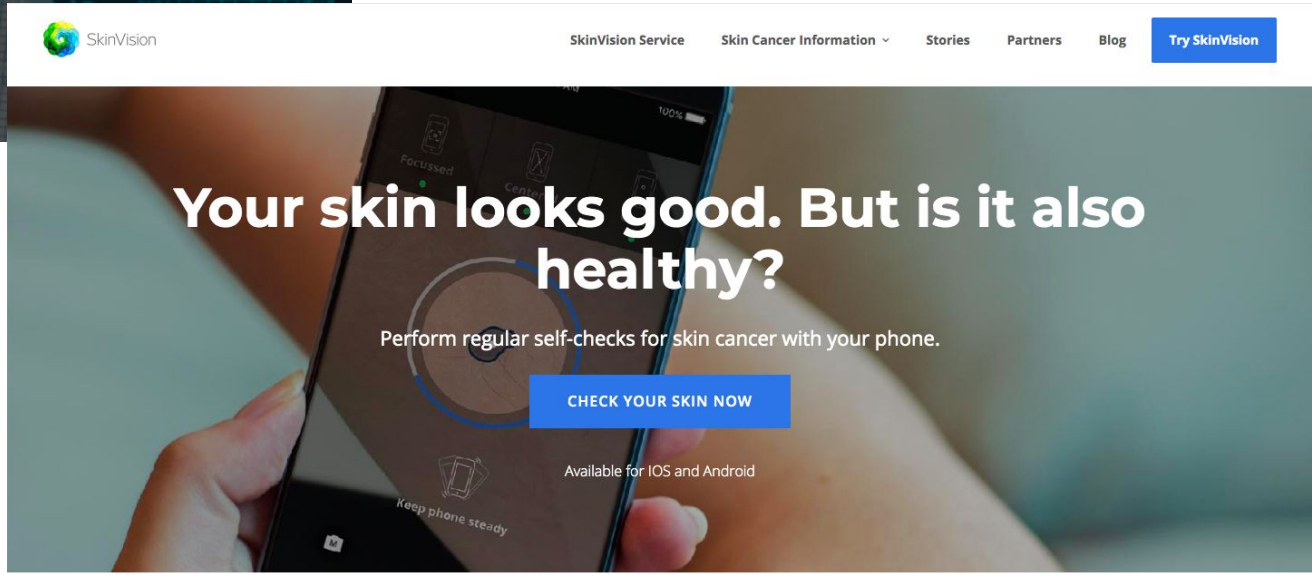
Convenience Store



Source: http://www.clresearch.com/research/detail.cfm?guid=6A608036-3048-78A9-2FB3-4E6295D65919

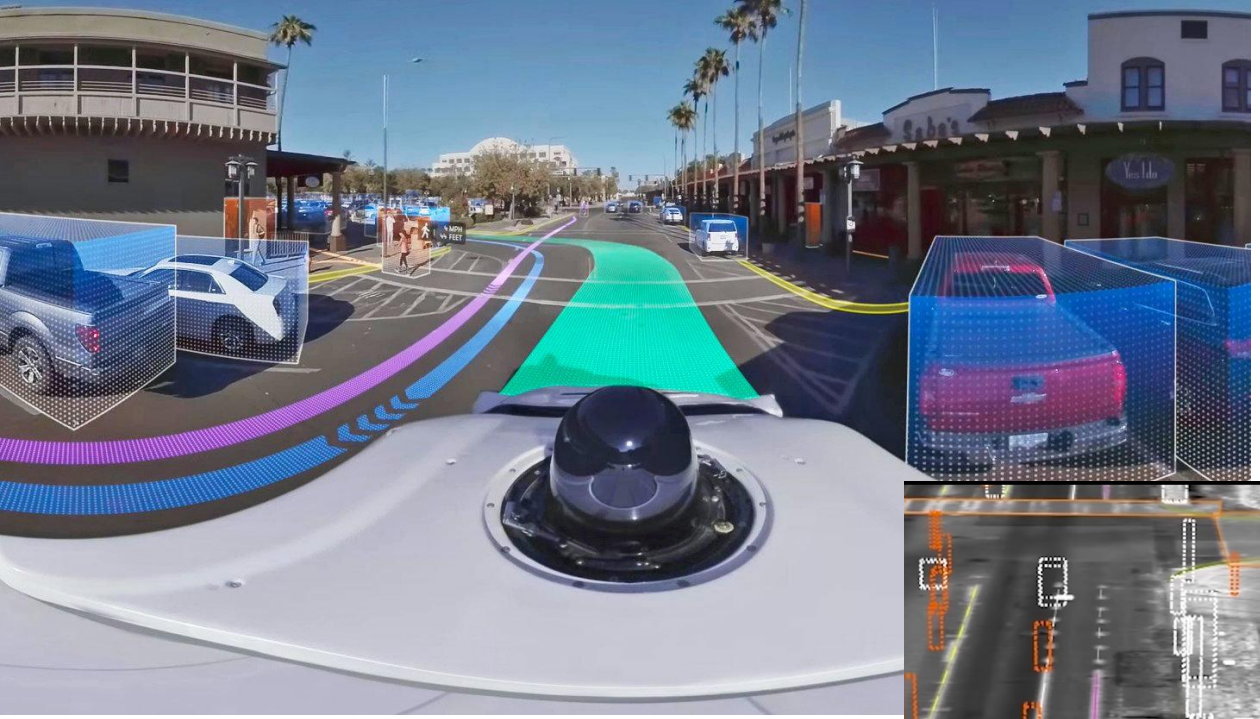
Skin Cancer Classification

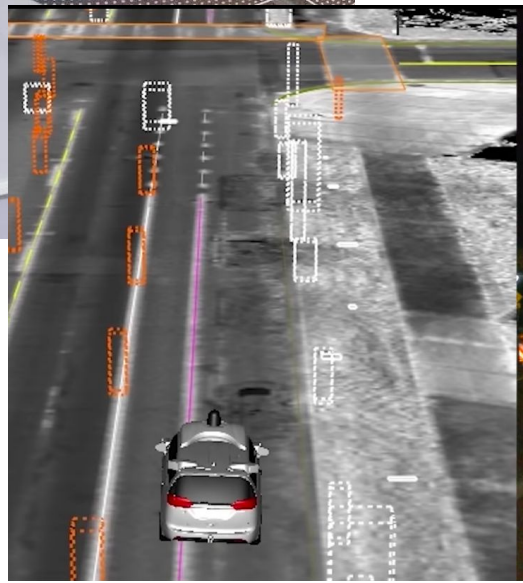


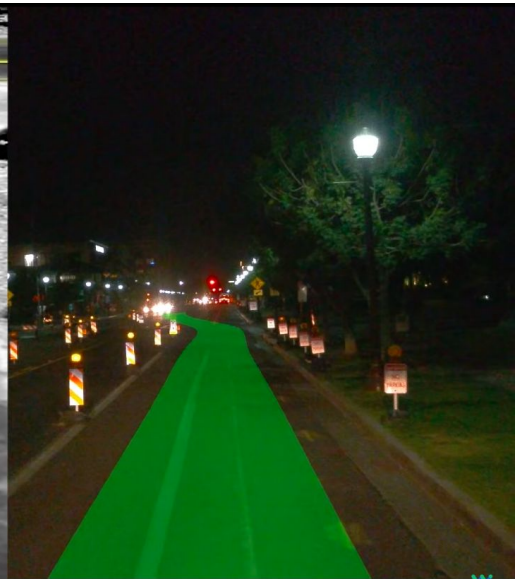


Source: https://cs.stanford.edu/people/esteva/nature/, https://www.skinvision.com

Autonomous Driving

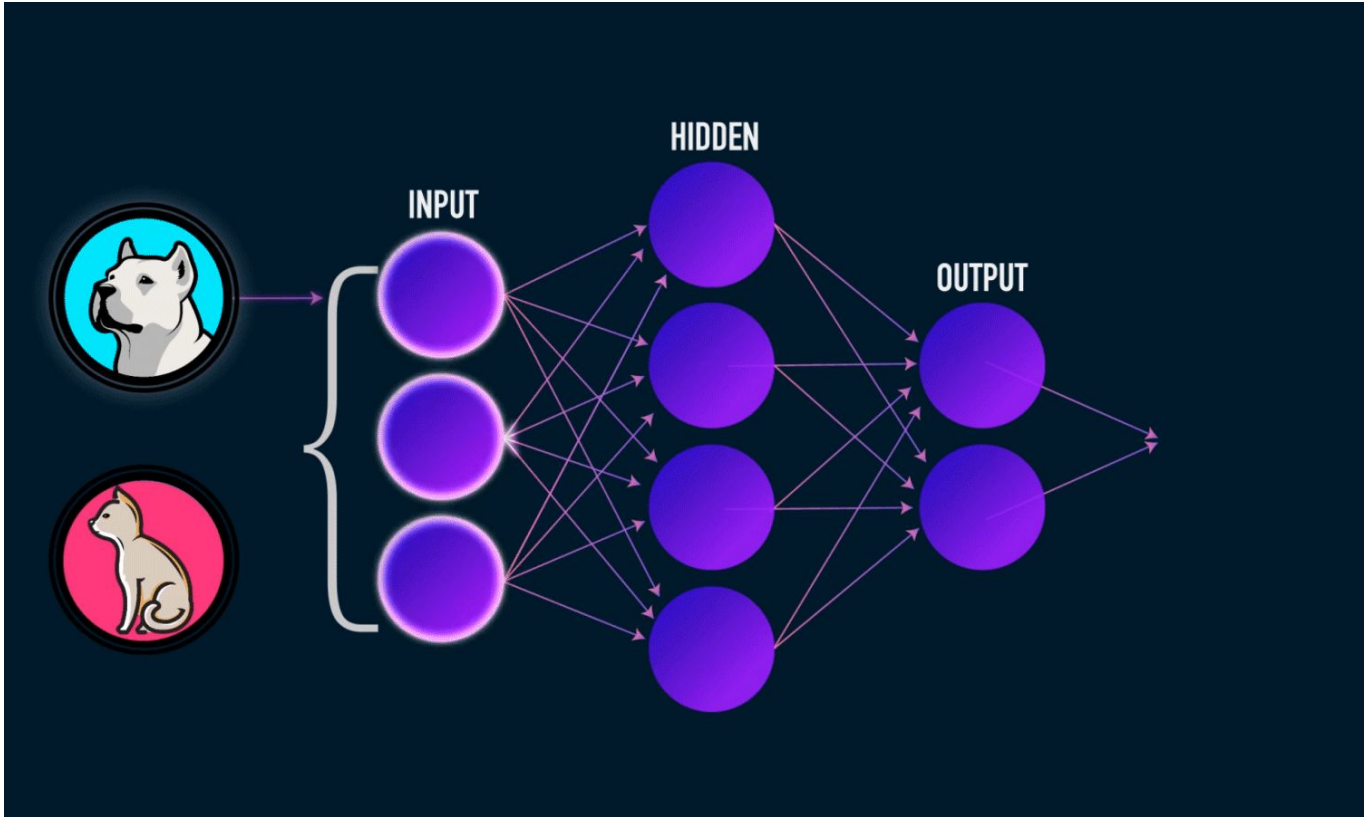






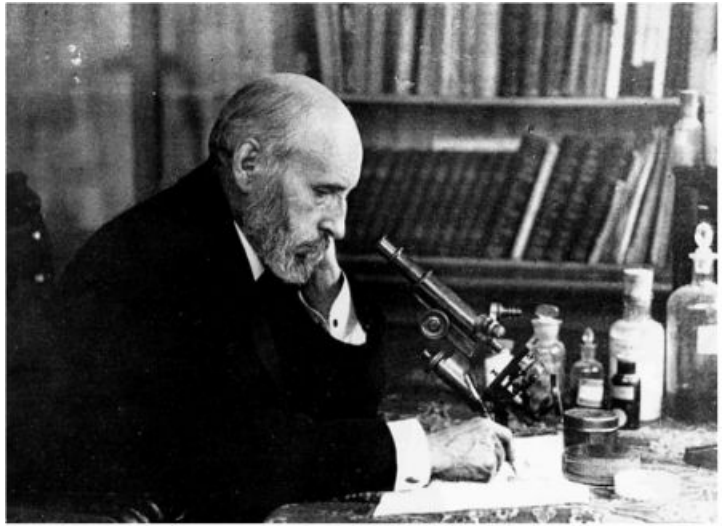
Source: https://www.engadget.com/2018/02/28/waymo-360-degree-vr-self-driving-car-demo/, https://youtu.be/ROAwXEqDk7k

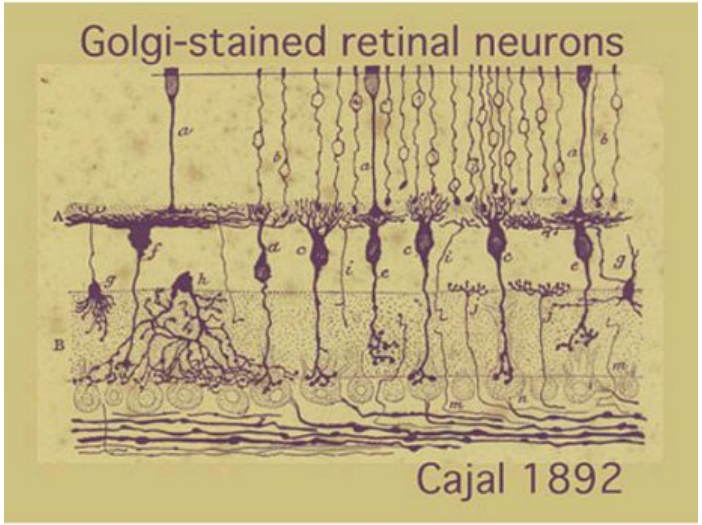
Neural Networks



Source: https://www.analyticsindiamag.com/how-to-create-your-first-artificial-neural-network-in-python/

Inspired from Neuroscience

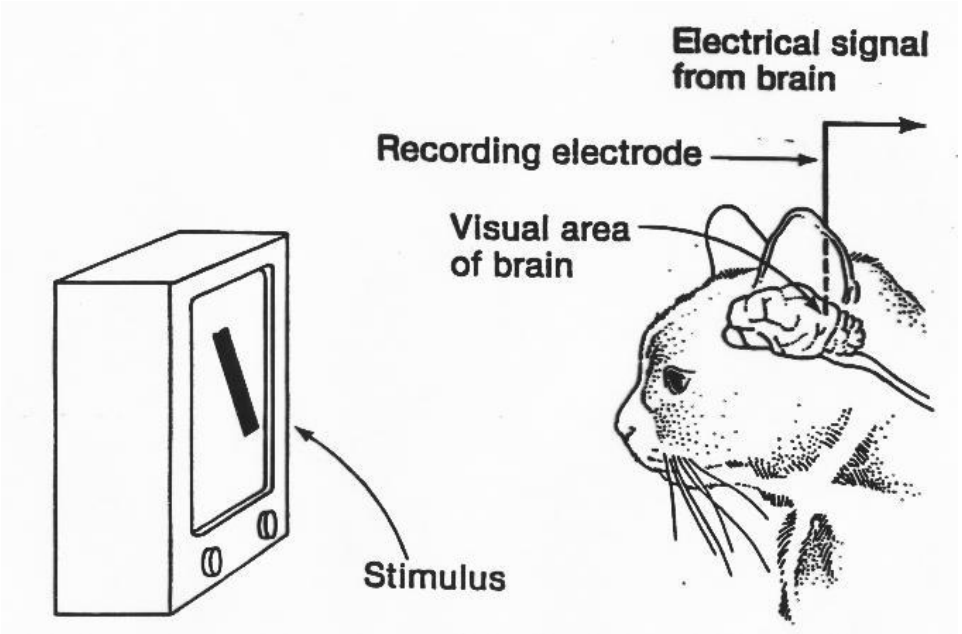




Santiago Ramón y Cajal
Spanish Neuroscientist and Pathologist
(Received Nobel Prize in 1906- joint work with Golgi

- A neuron is a discrete functional unit of the nervous system
- Neurons communicate with each other via specialized junctions, or spaces, between cells -- neural circuits !

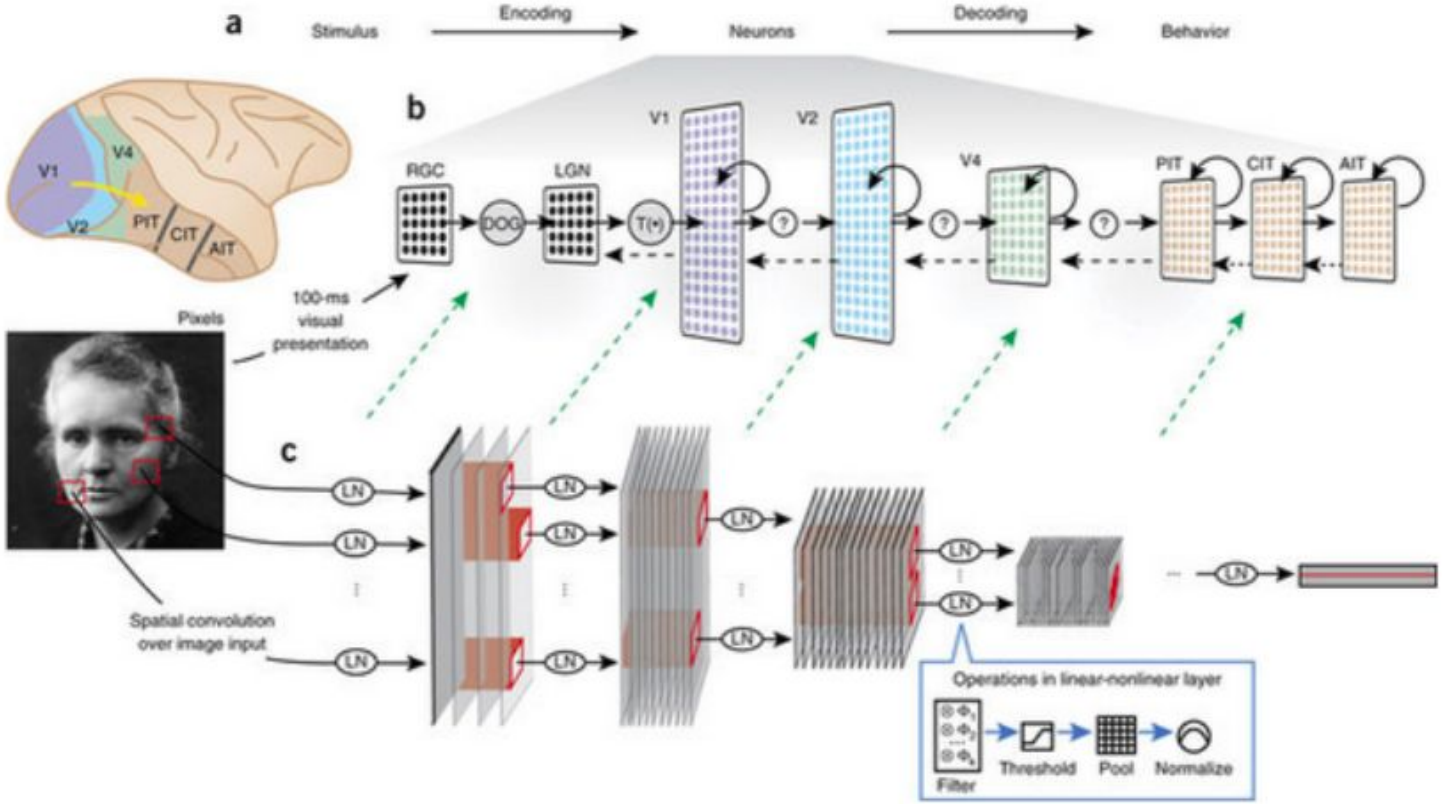
David Hubel and Torsten Wiesel (1950s)



Hubel and Wiesel study how cat's visual cortex react to the different orientation of the line. They earned Nobel Prize for Physiology or Medicine in 1981.

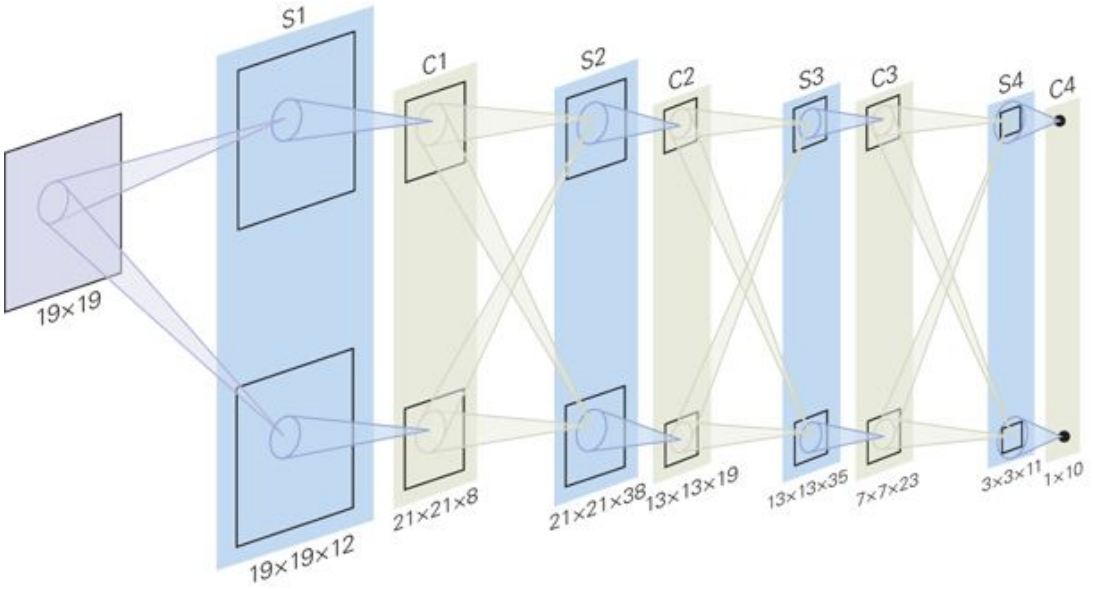
Source: https://goodpsychology.wordpress.com/2013/03/13/235/
http://braintour.harvard.edu/archives/portfolio-items/hubel-and-wiesel

Inspiration from Human Brain



Source: https://neurdiness.wordpress.com/2018/05/17/deep-convolutional-neural-networks-as-models-of-the-visual-system-qa/

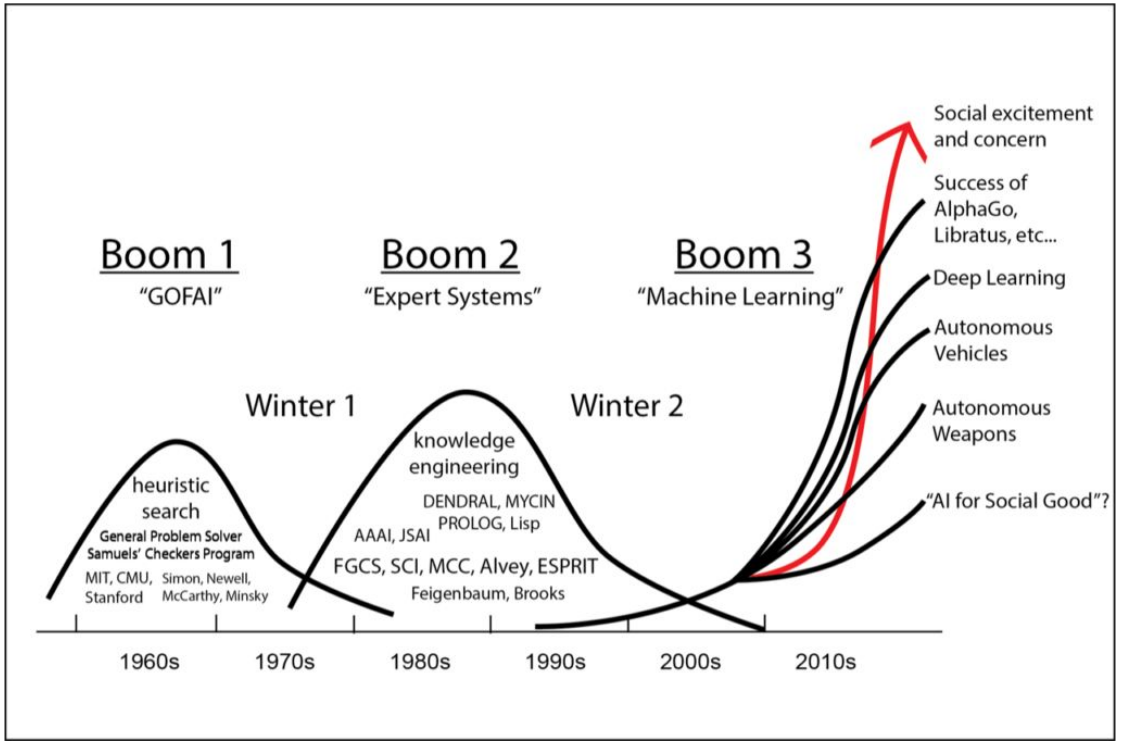
Neocognitron



Kunihiko Fukushima, 1980

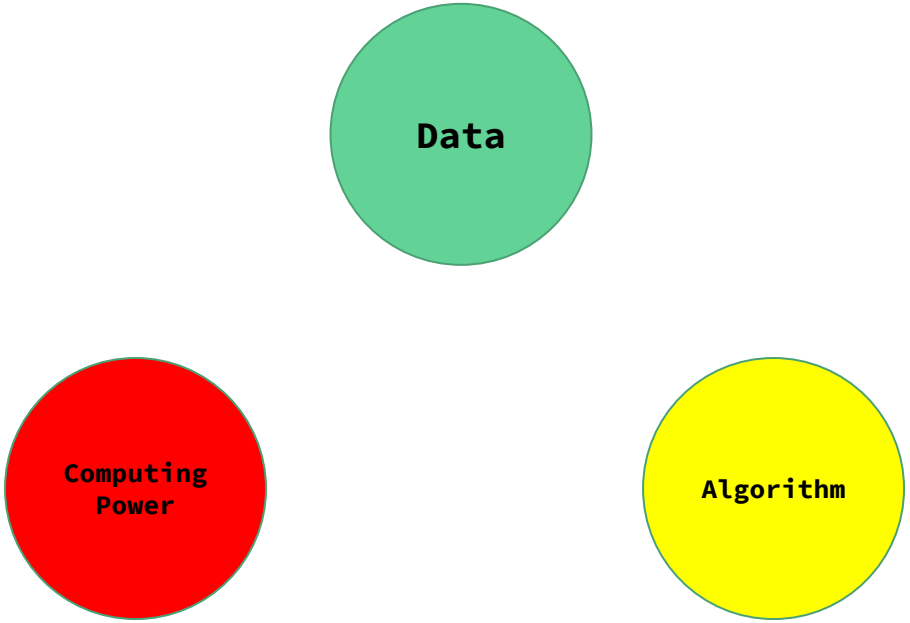
Source: https://neurology.mhmedical.com/book.aspx?bookID=1049, https://github.com/vtkuzmin/Neocognitron/blob/master/ClassNeocognitron.py

AI Winter

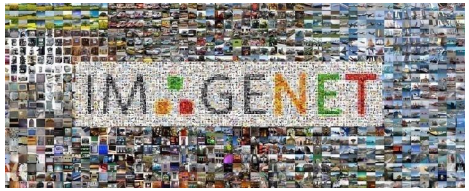


Source: https://www.technologystories.org/ai-evolution/

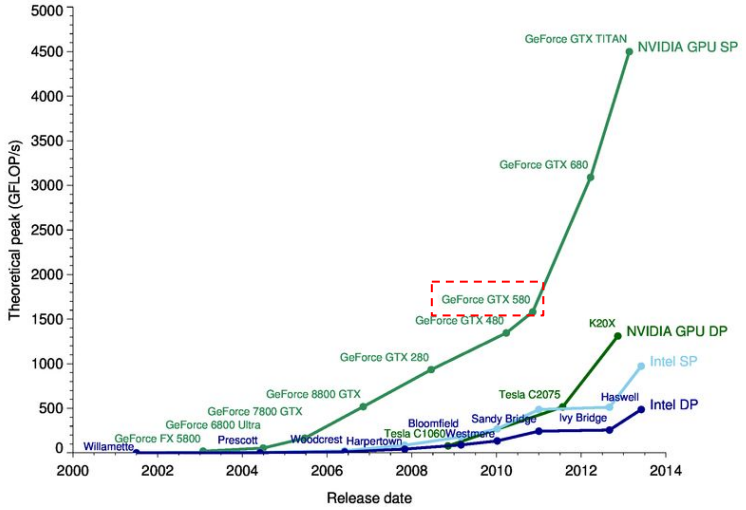
Neural Network Ingredients



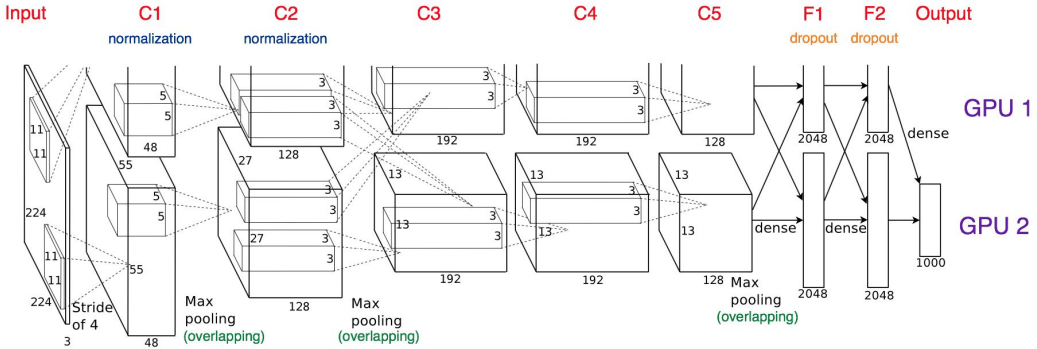
Neural Network Ingredients



14 million annotated images in 2009



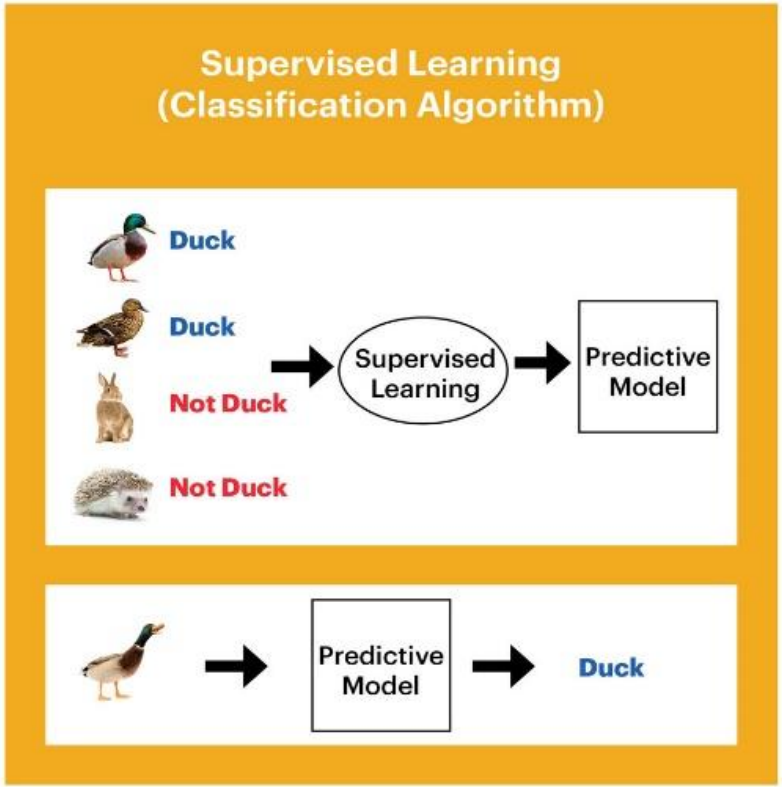
High performance processing unit using GPU.

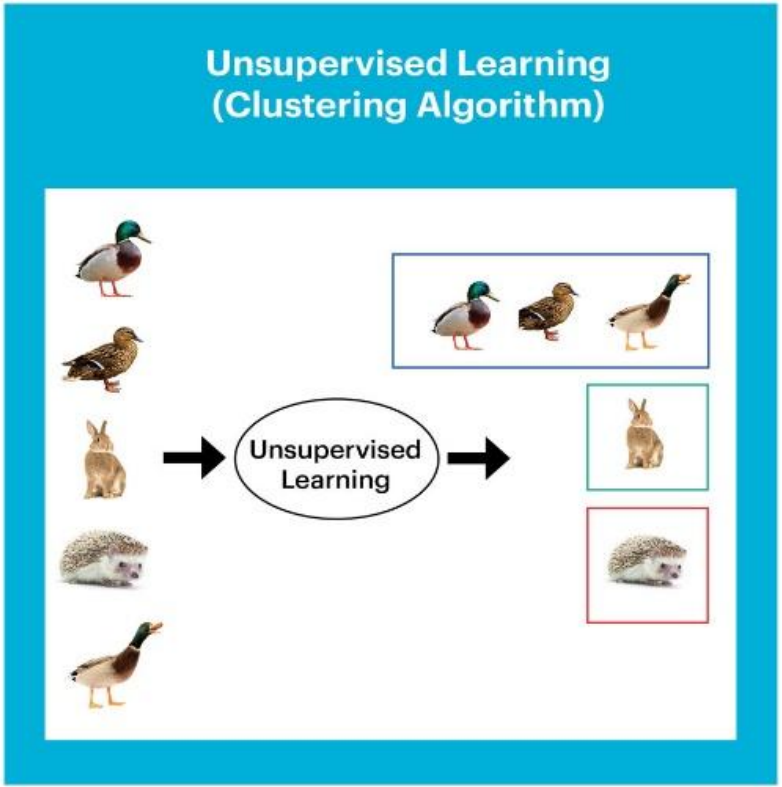


Neural network implementation breakthrough in 2012 by AlexNet.

Source: https://michaelgalloy.com/2013/06/11/cpu-vs-gpu-performance.html

Examples of Learning Algorithm

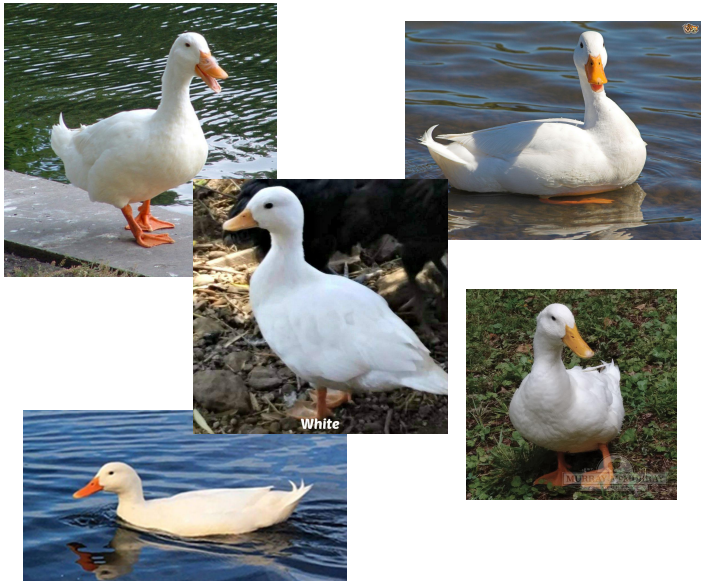




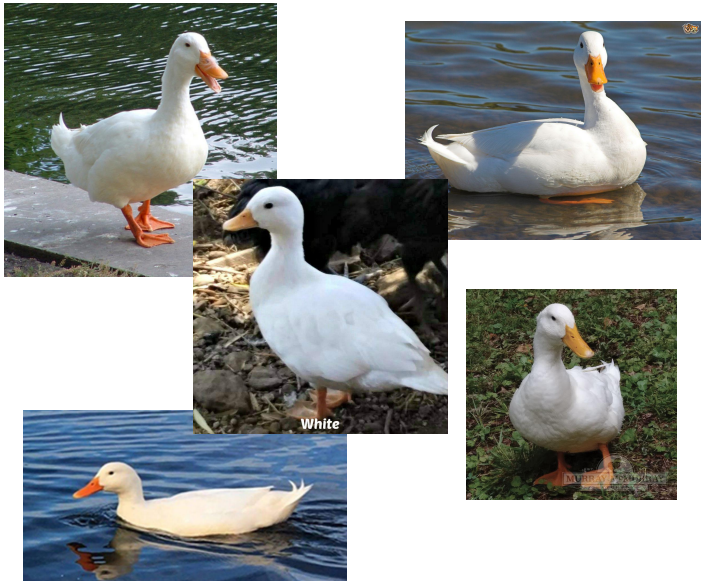
Western Digital.

Source: https://www.datanami.com/2018/09/05/how-to-build-a-better-machine-learning-pipeline/

Classification



Classification



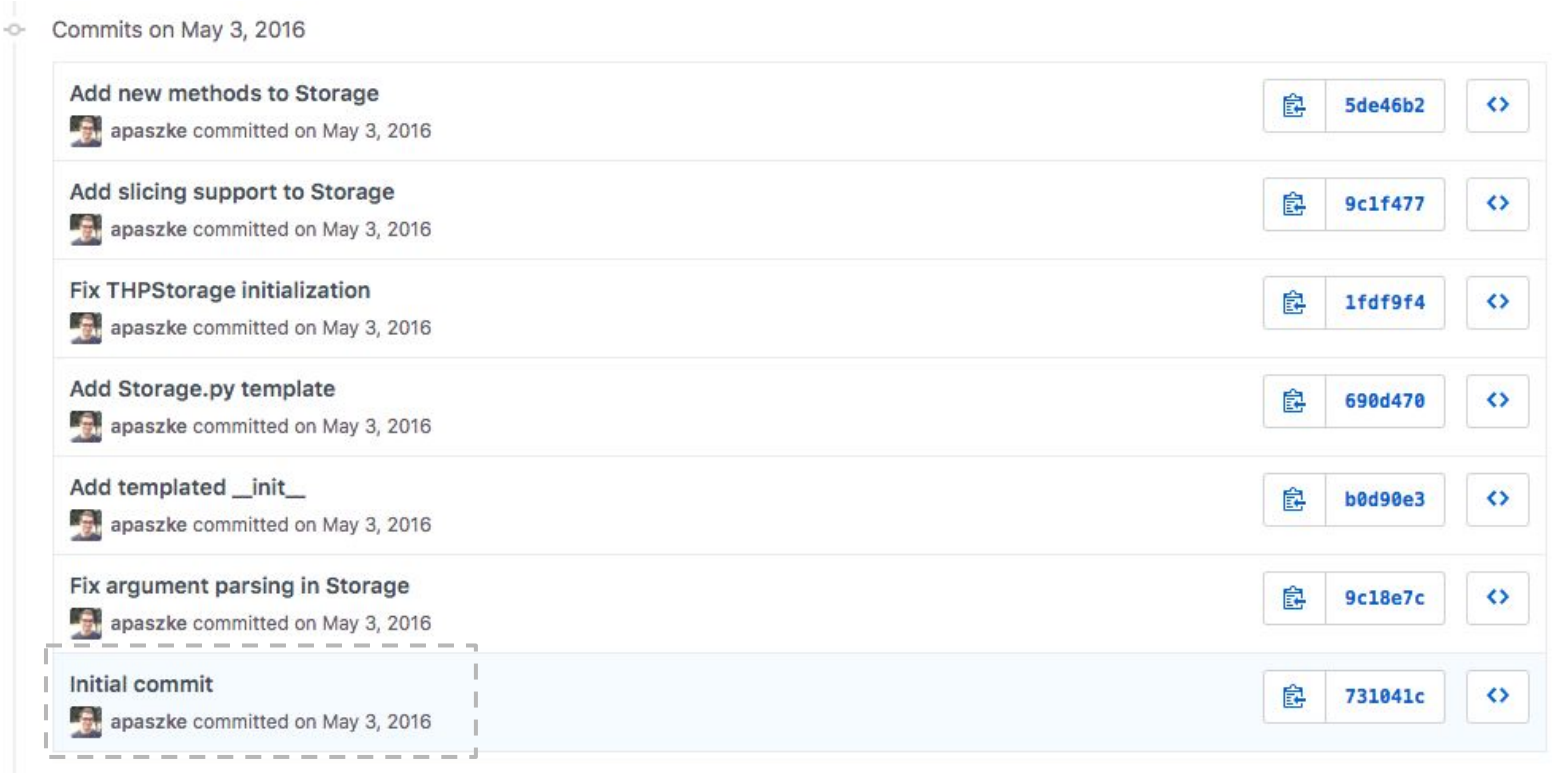
PyTorch

A Simple and Powerful Deep Learning Framework

PyTorch 1.0 & The Evolution of AI Development Frameworks

Source: https://www.youtube.com/watch?v=NKukpCIEH6M

The Origin of PyTorch

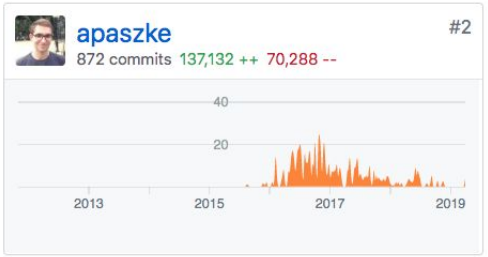


Source: https://github.com/pytorch/pytorch/commit/731041cb6a12b6f46626e16a04ab11523fed1c49

Me and Adam Paszke at PyCon Thailand 2019 :)

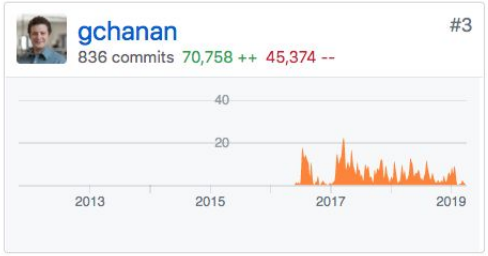
PyTorch Authors

Edward Z. Yang
Research Engineer
Facebook AI Research



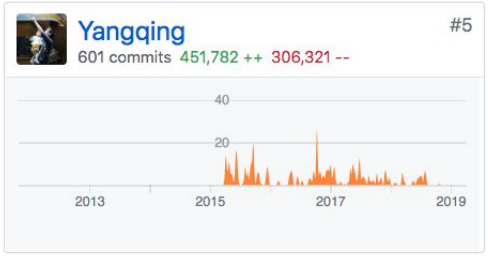
Adam Paszke
Deep Learning Engineer
NVIDIA

Gregory Channan
Technical Lead / Software Engineering Manager
Facebook



Soumith Chintala
Research Engineer
Facebook AI Research

Yangqing Jia
Research Scientist Director
Facebook



Sam Gross
Research Engineer
Facebook AI Research

PyTorch

PyTorch is a Python package that provides two high-level features:

- Tensor computation (like NumPy) with strong GPU acceleration
- Deep neural networks built on a tape-based autograd system

Source: https://github.com/pytorch/pytorch

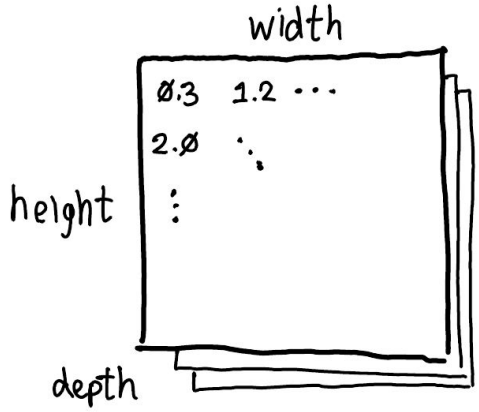
PyTorch

PyTorch is a Python package that provides two high-level features:

- Tensor computation (like NumPy) with strong GPU acceleration
- Deep neural networks built on a tape-based autograd system

Source: https://github.com/pytorch/pytorch

Tensor



sizes	(D, H, W)
strides	(H*W, W, 1)
dtype	float
device	cuda:0
layout	strided

contiguous

Source: http://blog.ezyang.com/2019/05/pytorch-internals/

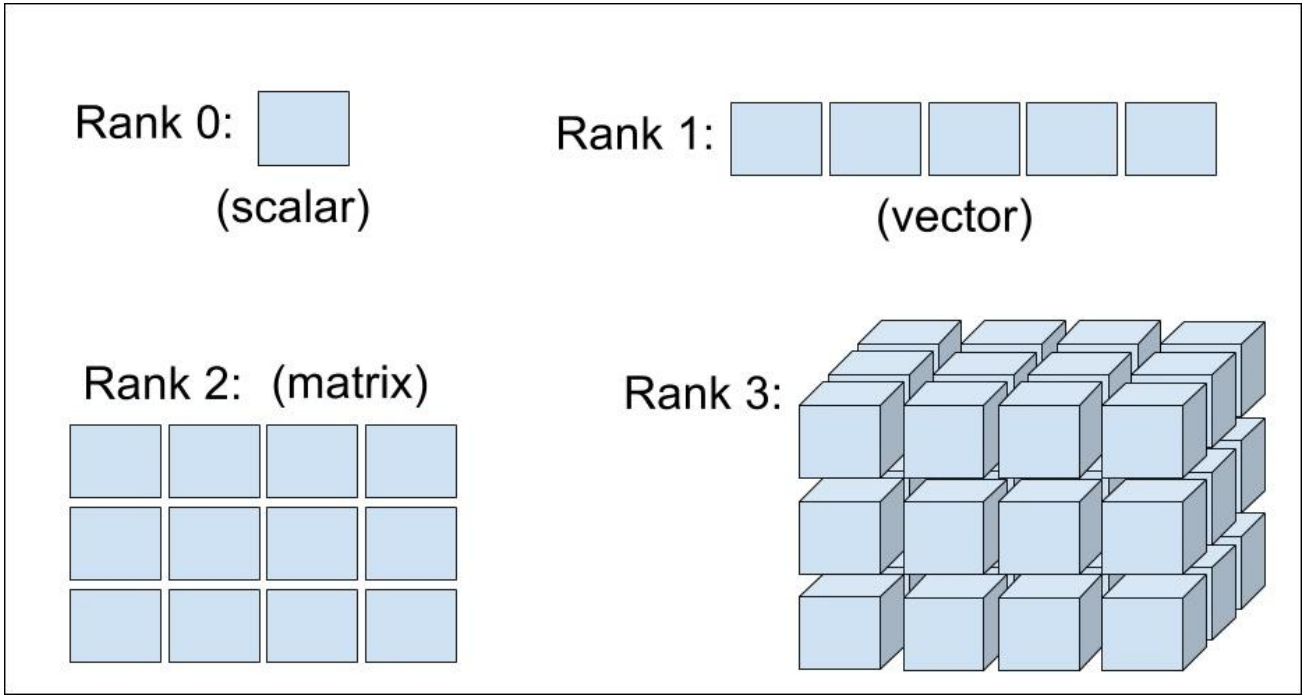
Tensor Supported Data Type

Data type	dtype	CPU tensor	GPU tensor
32-bit floating point	`torch.float32` or `torch.float`	`torch.FloatTensor`	`torch.cuda.FloatTensor`
64-bit floating point	`torch.float64` or `torch.double`	`torch.DoubleTensor`	`torch.cuda.DoubleTensor`
16-bit floating point	`torch.float16` or `torch.half`	`torch.HalfTensor`	`torch.cuda.HalfTensor`
8-bit integer (unsigned)	`torch.uint8`	`torch.ByteTensor`	`torch.cuda.ByteTensor`
8-bit integer (signed)	`torch.int8`	`torch.CharTensor`	`torch.cuda.CharTensor`
16-bit integer (signed)	`torch.int16` or `torch.short`	`torch.ShortTensor`	`torch.cuda.ShortTensor`
32-bit integer (signed)	`torch.int32` or `torch.int`	`torch.IntTensor`	`torch.cuda.IntTensor`
64-bit integer (signed)	`torch.int64` or `torch.long`	`torch.LongTensor`	`torch.cuda.LongTensor`

Source: https://pytorch.org/docs/stable/tensors.html

Tensor

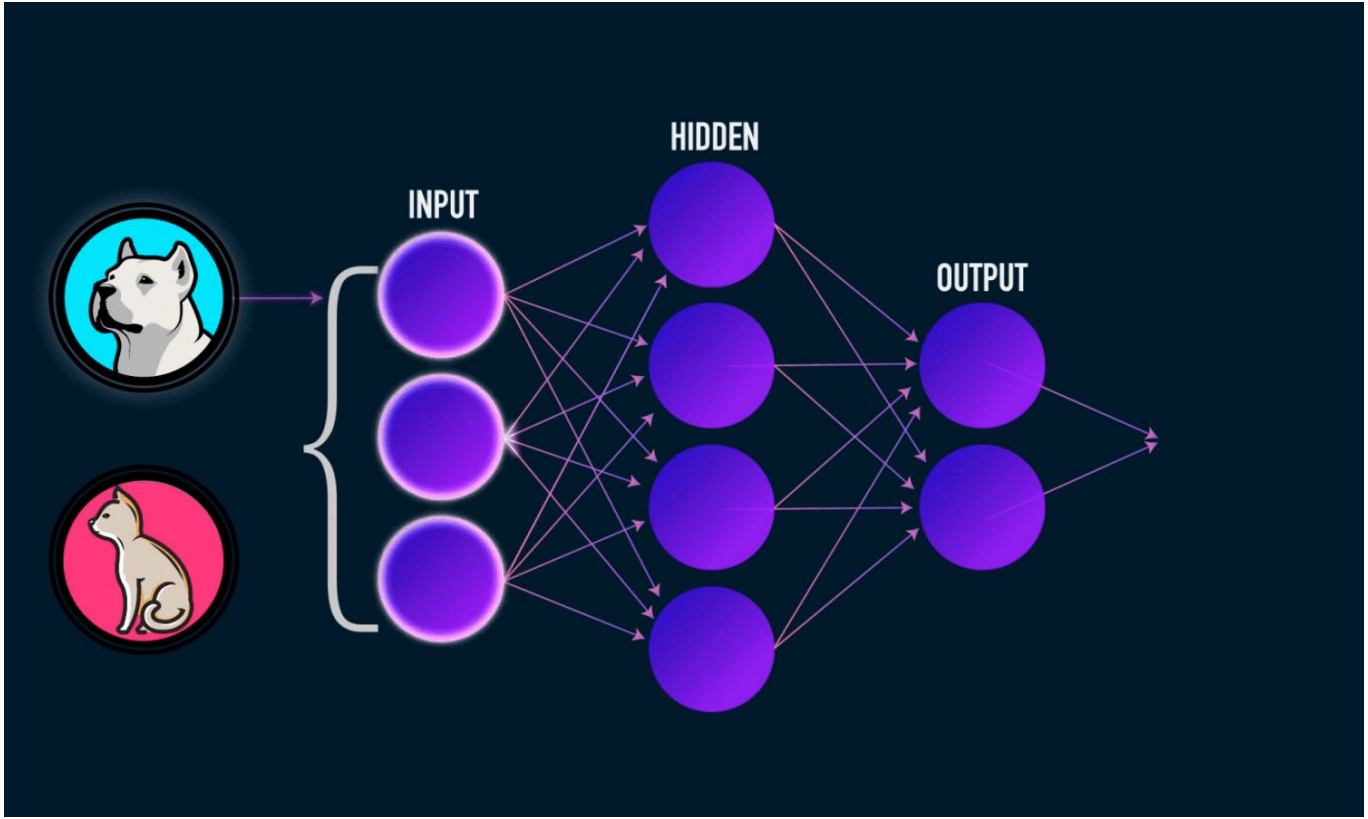
“A **tensor** is a type of **multidimensional array** with certain **transformation properties**”, --John Mangual



Source: https://math.stackexchange.com/a/1134812, https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781787125933/14/ch14lvl1sec85/tensorflow-ranks-and-tensors, https://www.reddit.com/r/math/comments/bztsoz/why_in_machine_learning_is_the_word_tensor_used/

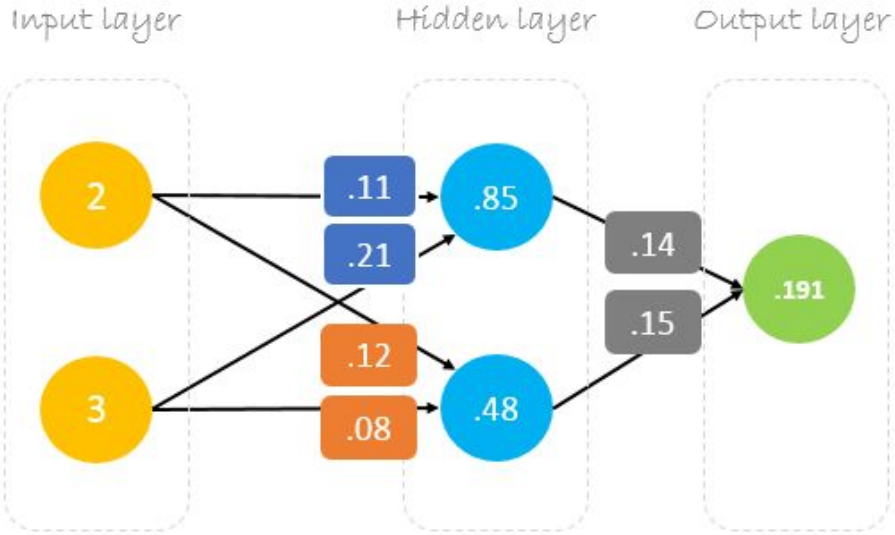
Workshop 1: Tensor in PyTorch

Neural Networks



Source: https://www.analyticsindiamag.com/how-to-create-your-first-artificial-neural-network-in-python/

Tensor as Neural Network Data Structure



Matrix multiplication

$$[2 \quad 3] \cdot \begin{bmatrix} 0.11 & 0.12 \\ 0.21 & 0.08 \end{bmatrix} = [0.85 \quad 0.48] \cdot \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} = [0.191]$$

Details

$2 \times .11 + 3 \times .21 = .85$

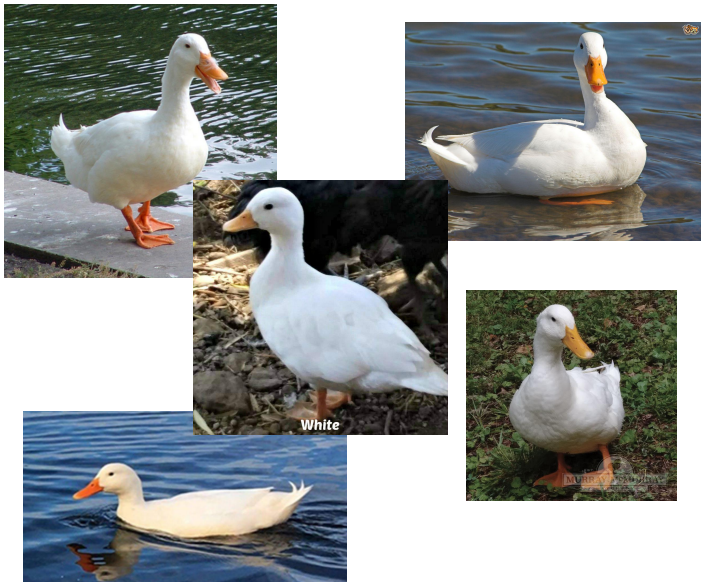
$2 \times .12 + 3 \times .08 = .48$

$.85 \times .14 + .48 \times .15 = .191$

Source: https://hmkcode.github.io/ai/backpropagation-step-by-step/

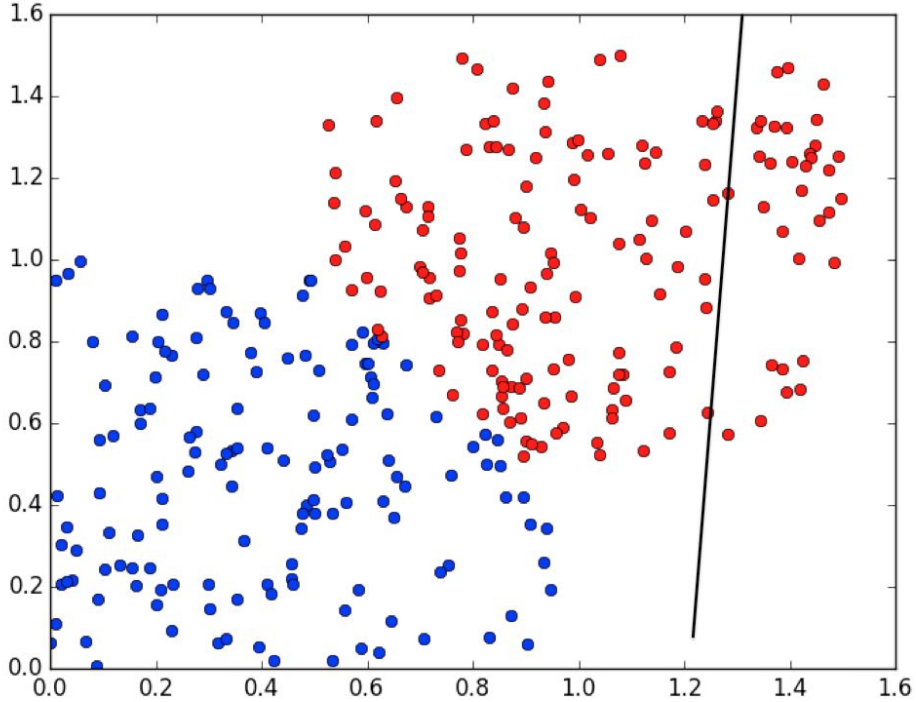
Workshop 2: A Simple Neural Network in PyTorch

Goal: Classification



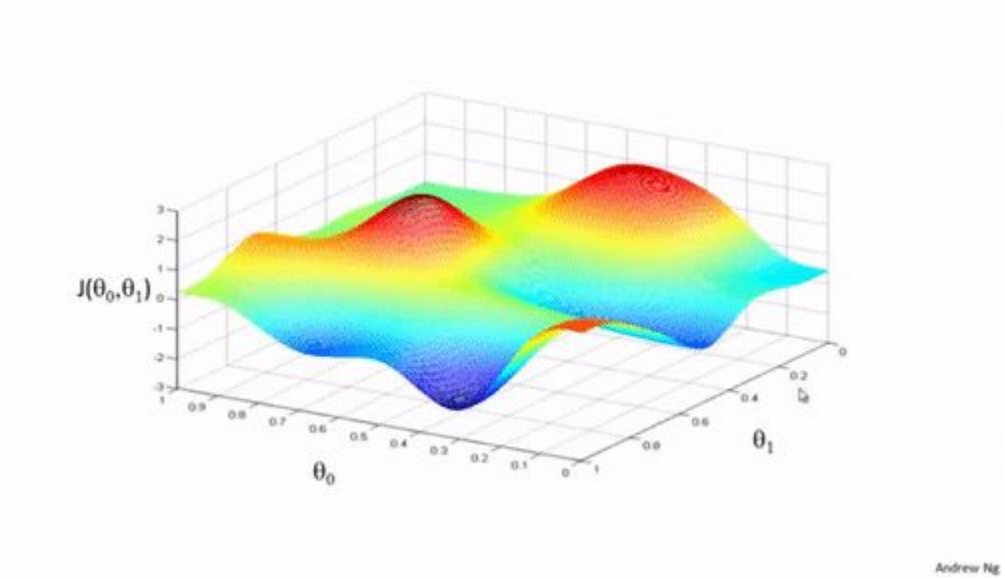
Workshop 3: Implementing PyTorch Neural Network on MNIST Dataset

Classification



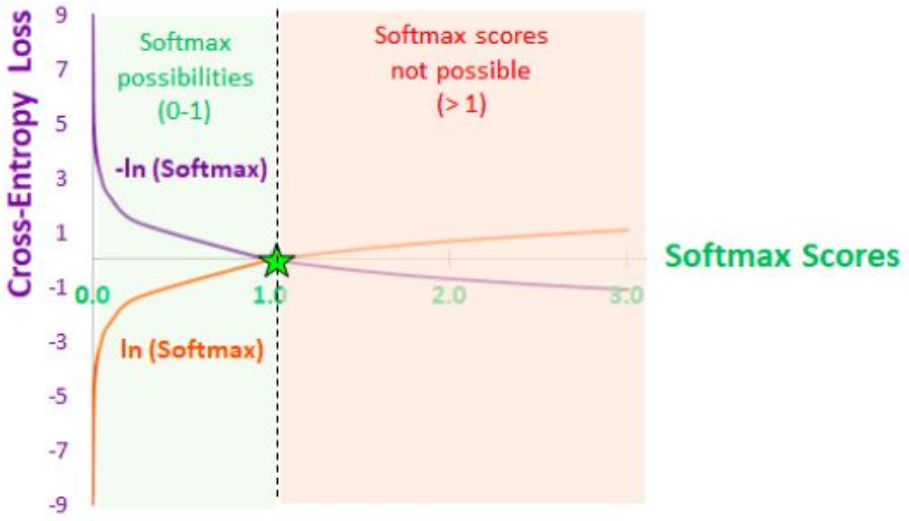
Source: https://medium.com/coinmonks/backpropagation-concept-explained-in-5-levels-of-difficulty-8b220a939db5

Minimize Error by Adjusting Weights

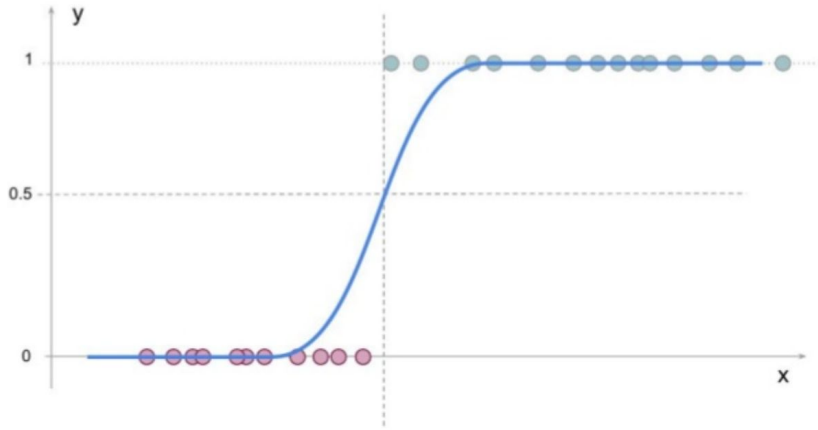


Minimize Error

Error Function and Activation Function



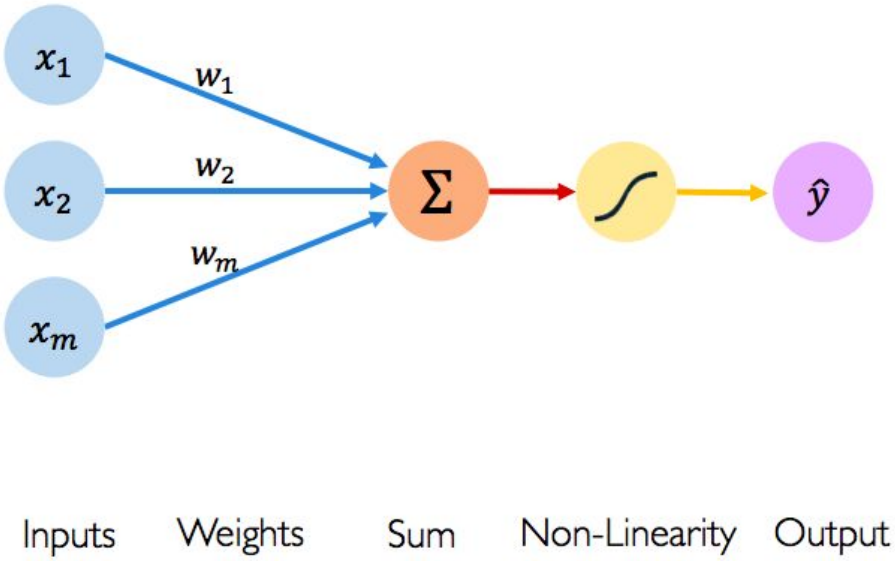
Error Function



Activation Function

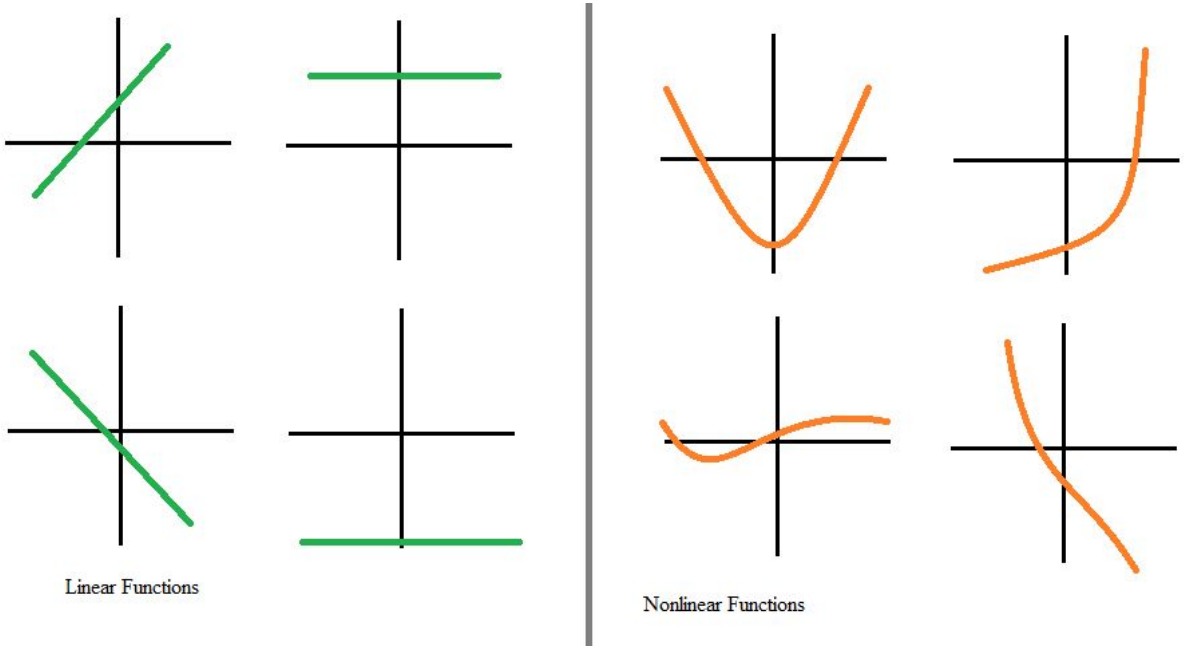
Source: https://www.slideshare.net/xavigiro/the-perceptron-audio-and-vision-d1l2-2017-upc-deep-learning-for-artificial-intelligence

Neural Network with Activation Function



Source: http://introtodeeplearning.com/materials/2019_6S191_L1.pdf

Linear vs Nonlinear



https://playground.tensorflow.org

Source: https://study.com/academy/lesson/how-to-recognize-linear-functions-vs-non-linear-functions.html

Activation Function Example

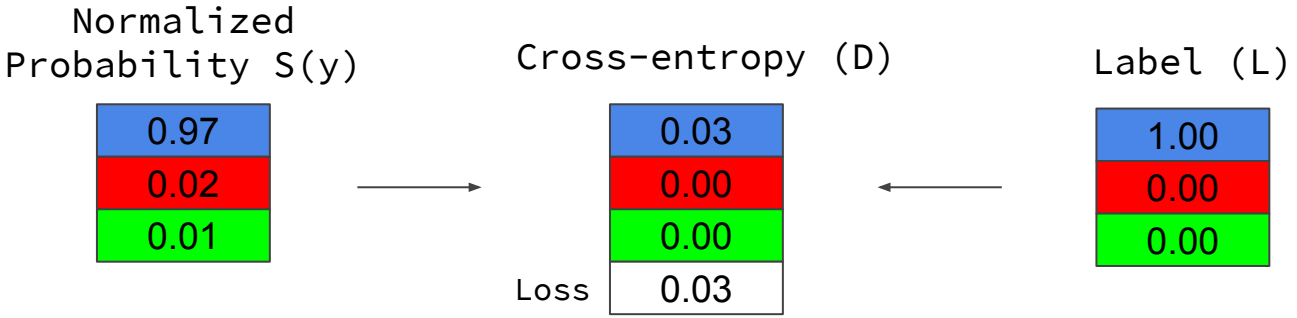
Actual Image

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

	Raw Probability		Unnormalized Probability		Normalized Probability S(y)
Cat	14.2	e^{logit}	1,468,864	normalize	0.97
Dog	10.2		26,903		0.02
Tiger	9.4		12,088		0.01
		Sum	1,507,856	Sum	1.0

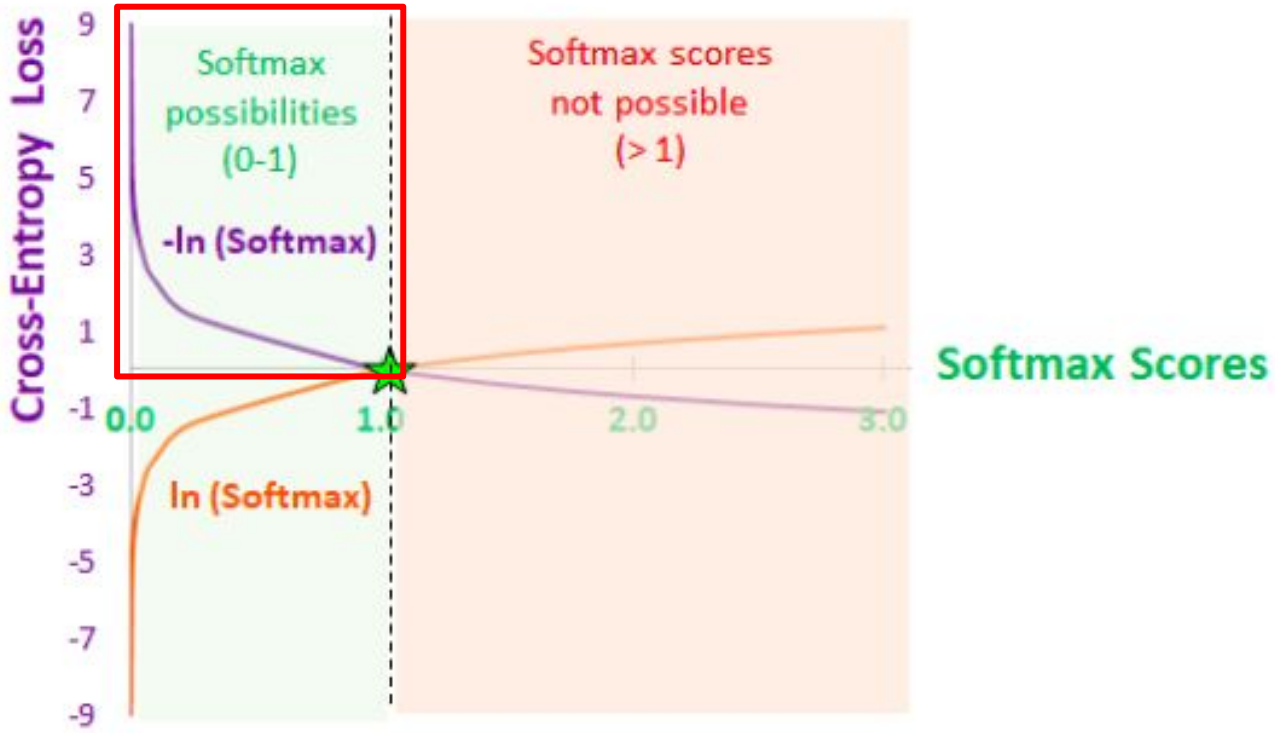
Error Function Example

$$D(S, L) = -\sum_i L_i \log(S_i)$$



Cross-entropy measures distance (D) between S(y) and label (L) for the correct class.

Error Function



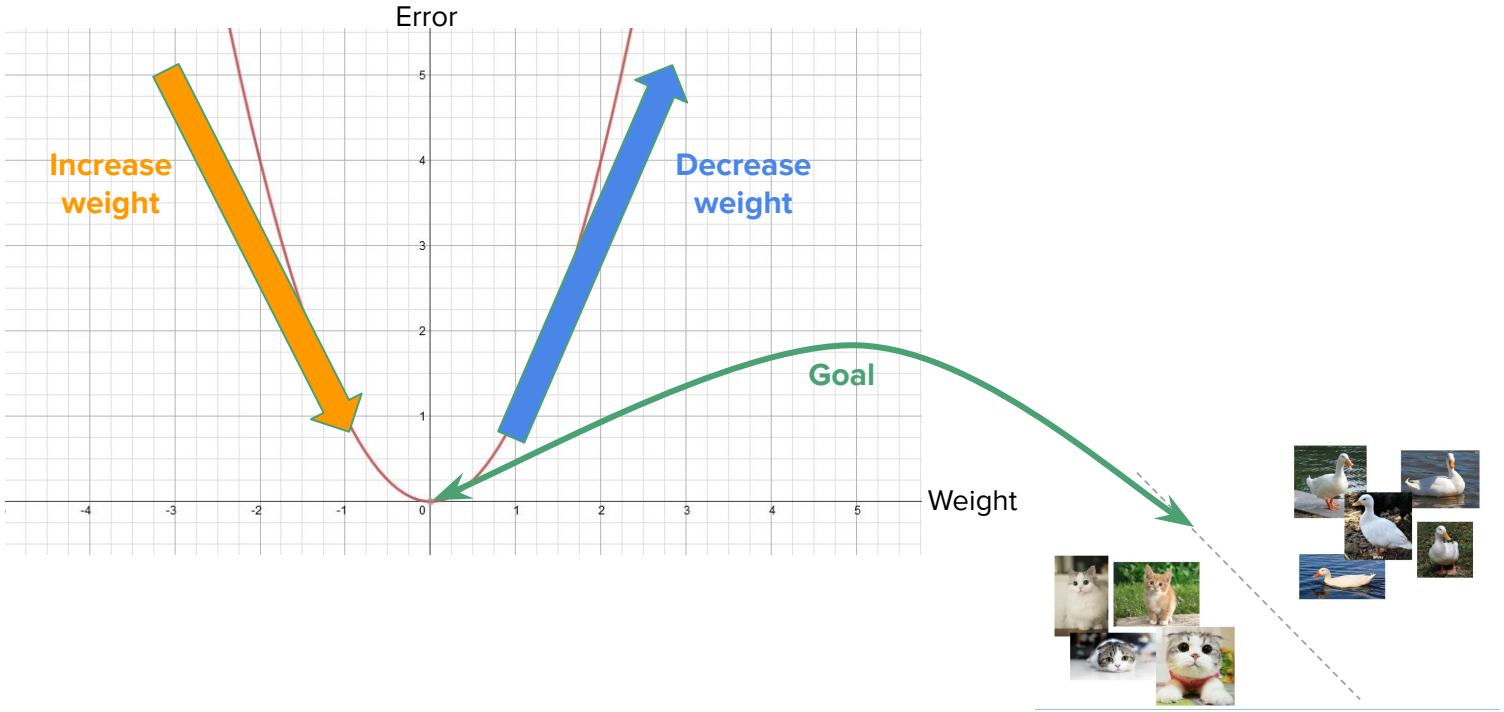
Error Functions

data type	loss	$L(u, a)$
real	quadratic	$(u-a)^2$
real	absolute value	$\lvert u-a \rvert$
real	huber	$\mathbf{huber}(u-a)$
boolean	hinge	$(1-ua)_+$
boolean	logistic	$\log(1+\exp(-au))$
integer	poisson	$\exp(u) - au + a\log a - a$
ordinal	ordinal hinge	$\sum_{a'=1}^{a-1}(1-u+a')_+ + \sum_{a'=a+1}^{d}(1+u-a')_+$
categorical	one-vs-all	$(1-u_a)_+ + \sum_{a' \neq a}(1+u_{a'})_+$
categorical	multinomial logit	$\frac{\exp(u_a)}{\sum_{a'=1}^{d}\exp(u_{a'})}$

https://pytorch.org/docs/stable/nn.html#loss-functions

Source: https://people.orie.cornell.edu/mru8/doc/udell18_big_data.pdf

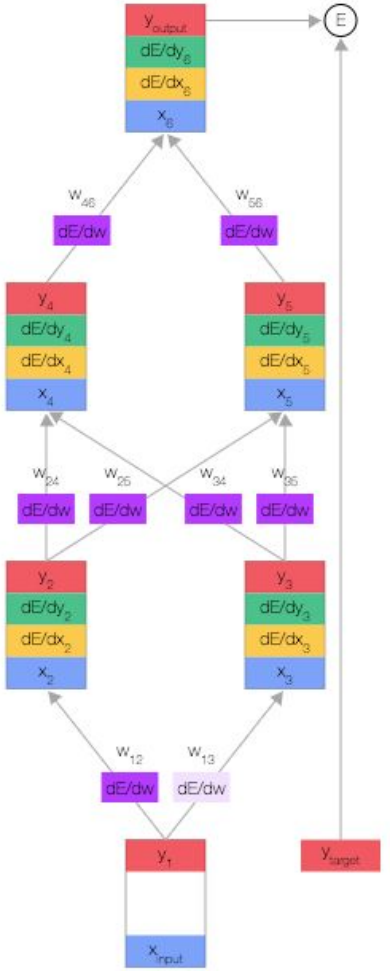
How to Update Weights



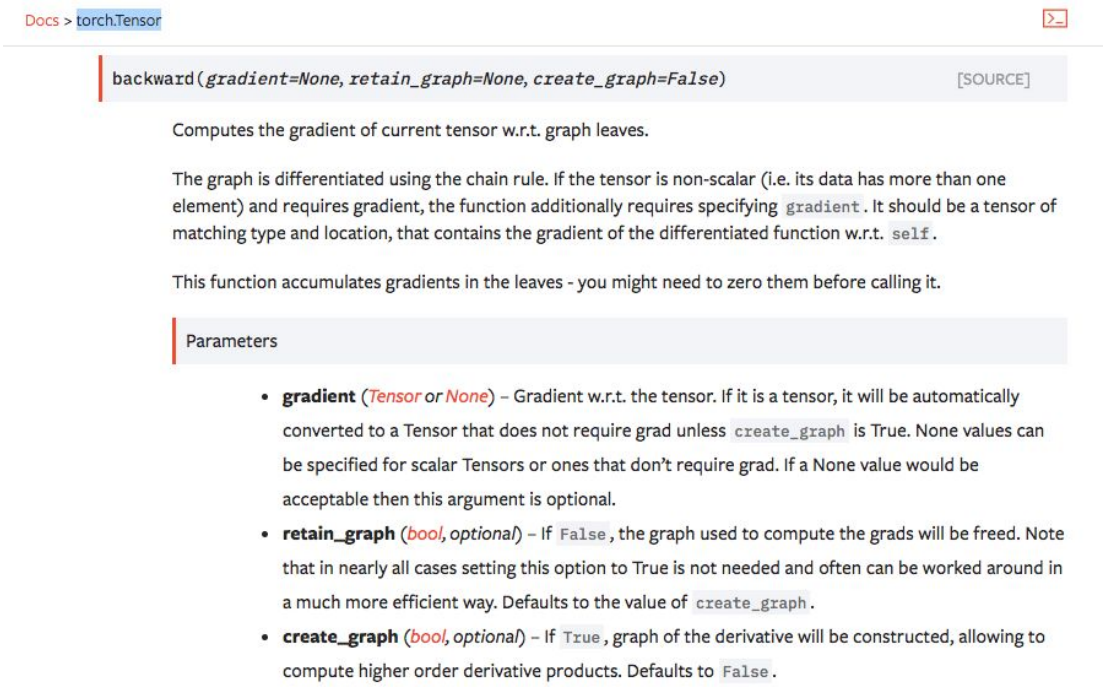
Source: https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

Backpropagation Algorithm

https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/



PyTorch Autograd (Automatic Differentiation)



Docs > torch.Tensor

backward(*gradient=None, retain_graph=None, create_graph=False*) [SOURCE]

Computes the gradient of current tensor w.r.t. graph leaves.

The graph is differentiated using the chain rule. If the tensor is non-scalar (i.e. its data has more than one element) and requires gradient, the function additionally requires specifying `gradient`. It should be a tensor of matching type and location, that contains the gradient of the differentiated function w.r.t. `self`.

This function accumulates gradients in the leaves - you might need to zero them before calling it.

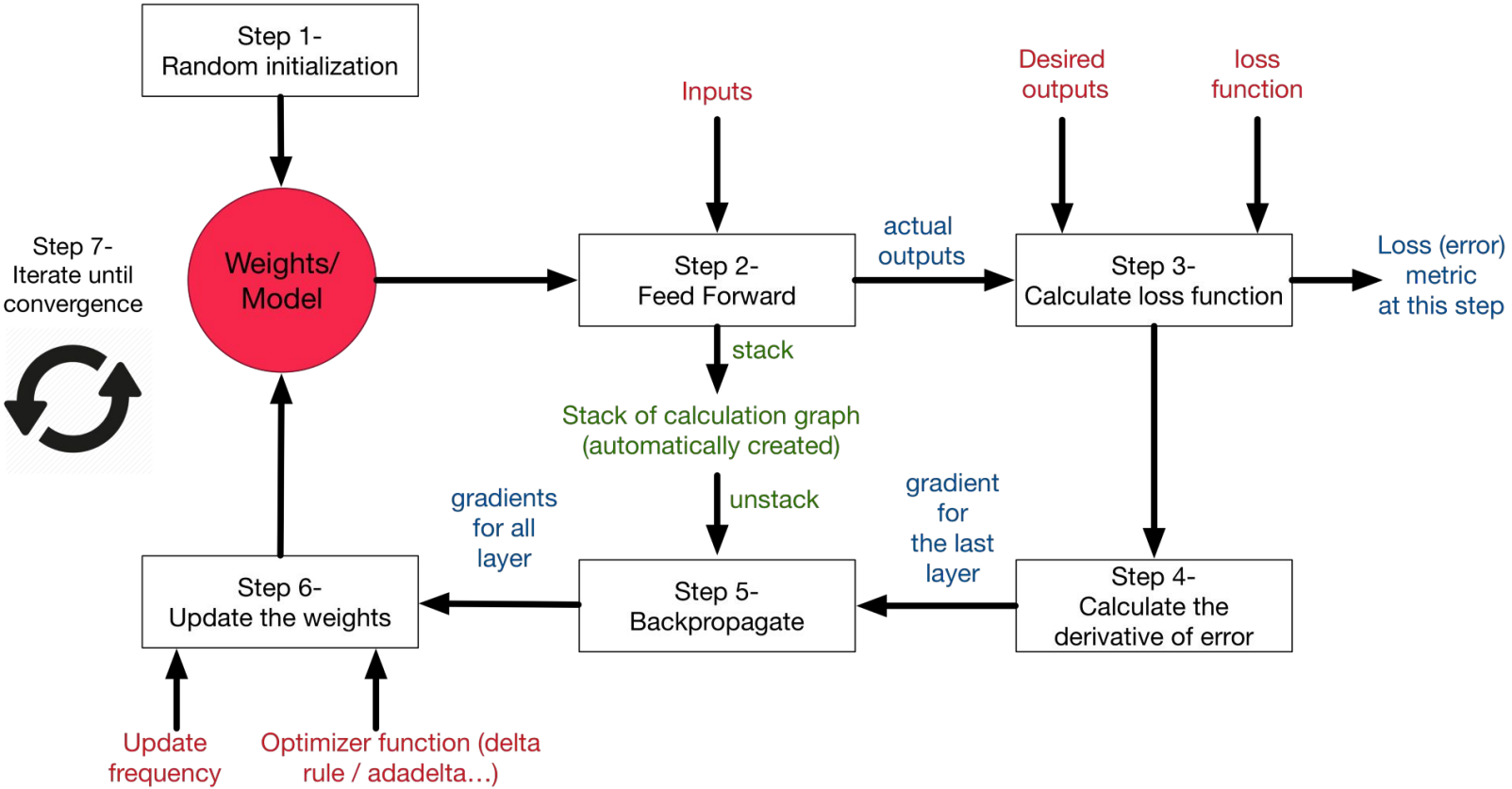
Parameters

- **gradient** (*Tensor* or *None*) – Gradient w.r.t. the tensor. If it is a tensor, it will be automatically converted to a Tensor that does not require grad unless `create_graph` is True. None values can be specified for scalar Tensors or ones that don't require grad. If a None value would be acceptable then this argument is optional.
- **retain_graph** (*bool, optional*) – If `False`, the graph used to compute the grads will be freed. Note that in nearly all cases setting this option to True is not needed and often can be worked around in a much more efficient way. Defaults to the value of `create_graph`.
- **create_graph** (*bool, optional*) – If `True`, graph of the derivative will be constructed, allowing to compute higher order derivative products. Defaults to `False`.

More detail: http://blog.ezyang.com/2019/05/pytorch-internals/
https://rufflewind.com/2016-12-30/reverse-mode-automatic-differentiation

Source: https://pytorch.org/docs/stable/tensors.html?highlight=backward#torch.Tensor.backward

Recap



Source: https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

PyTorch

PyTorch is a Python package that provides two high-level features:

- Tensor computation (like NumPy) with strong GPU acceleration
- Deep neural networks built on a tape-based autograd system

Source: https://github.com/pytorch/pytorch

Workshop 4: Train Neural Network in PyTorch

Workshop 5: Fashion-MNIST Dataset

We would love to hear your feedback :)

https://tinyurl.com/pytorch-feedback