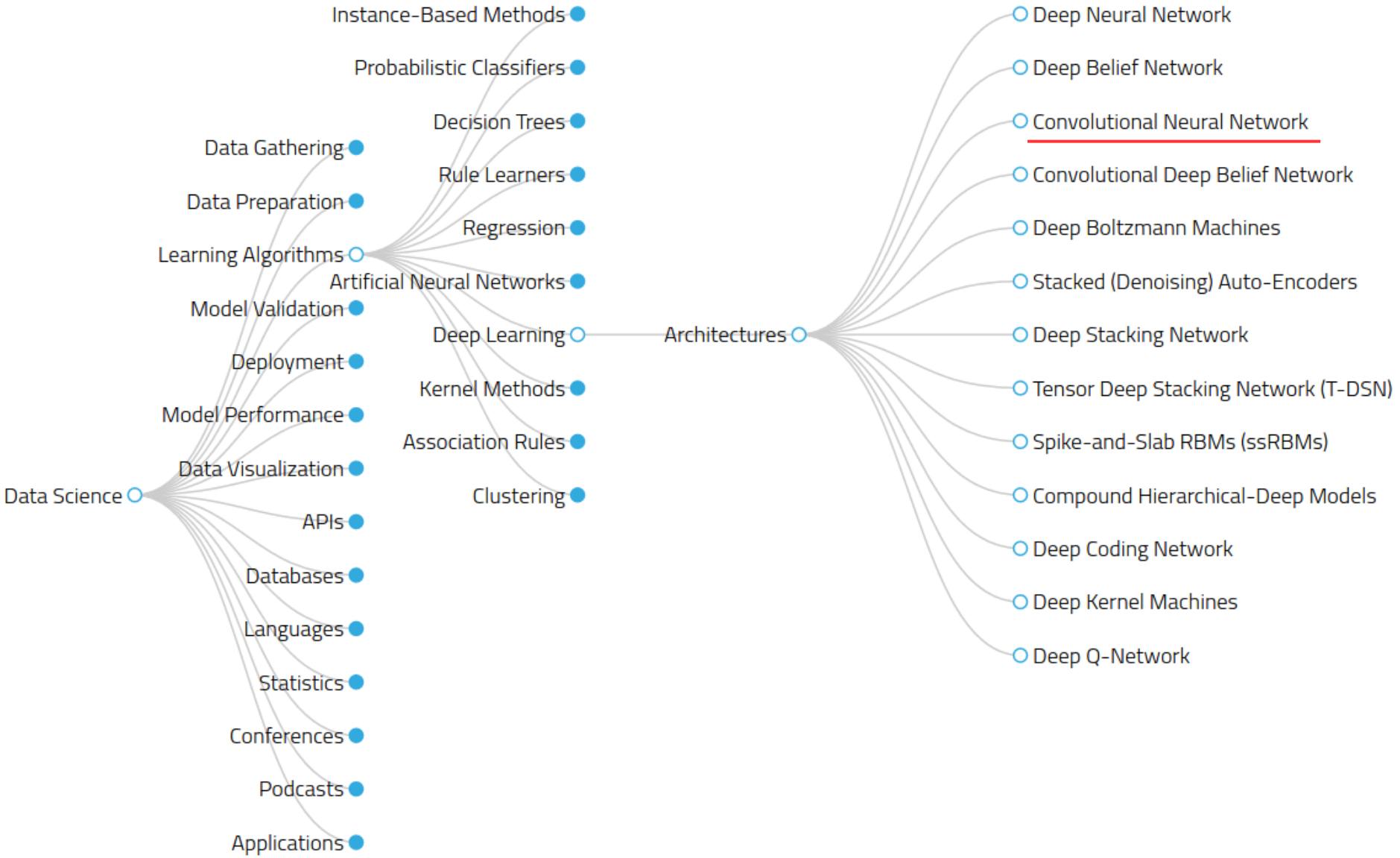
Deep Learning – 101

Arun Aniyan
SKA SA / Rhodes University
arun@ska.ac.za

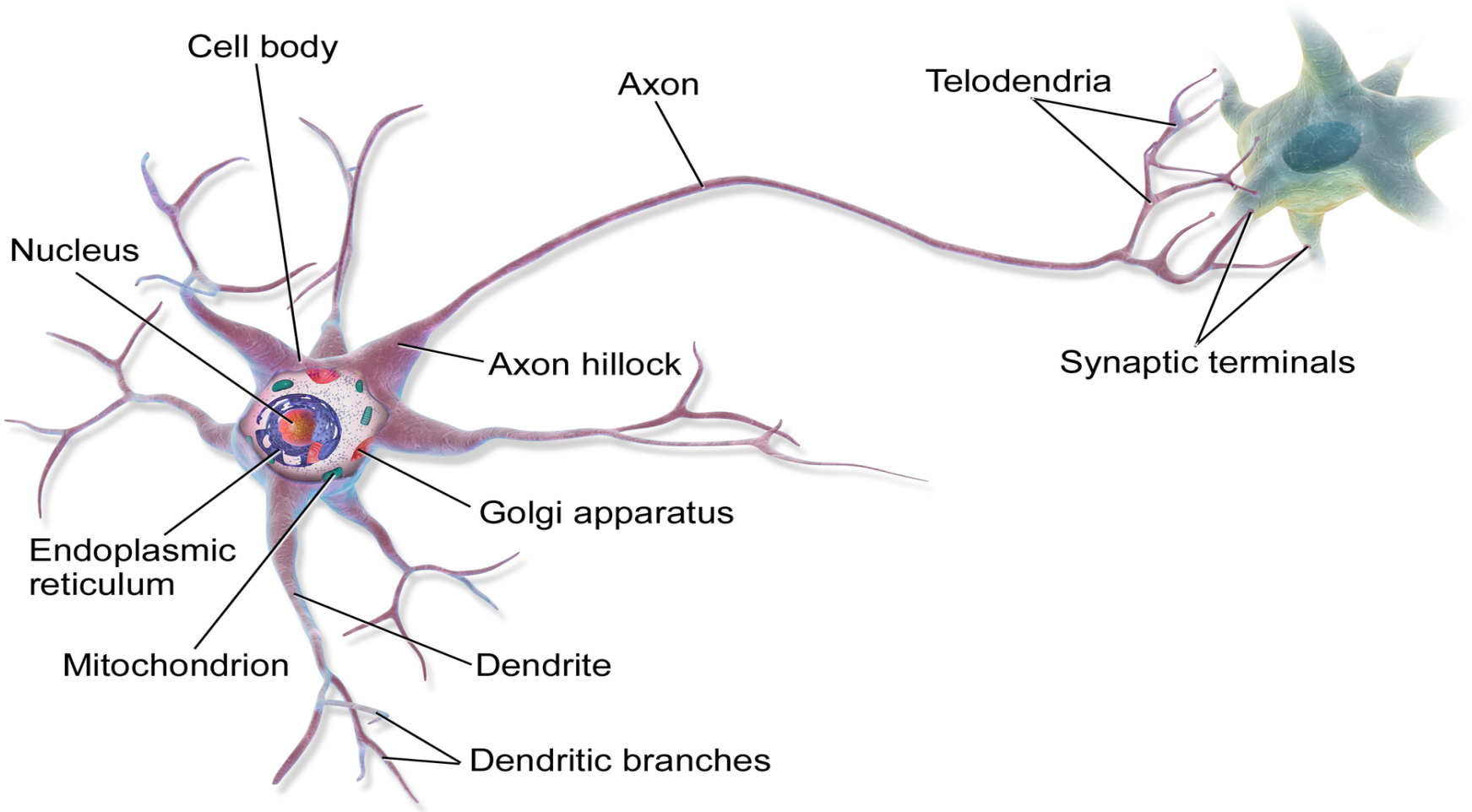

WE NEED TO GO
DEEPER

Deep Learning Ontology

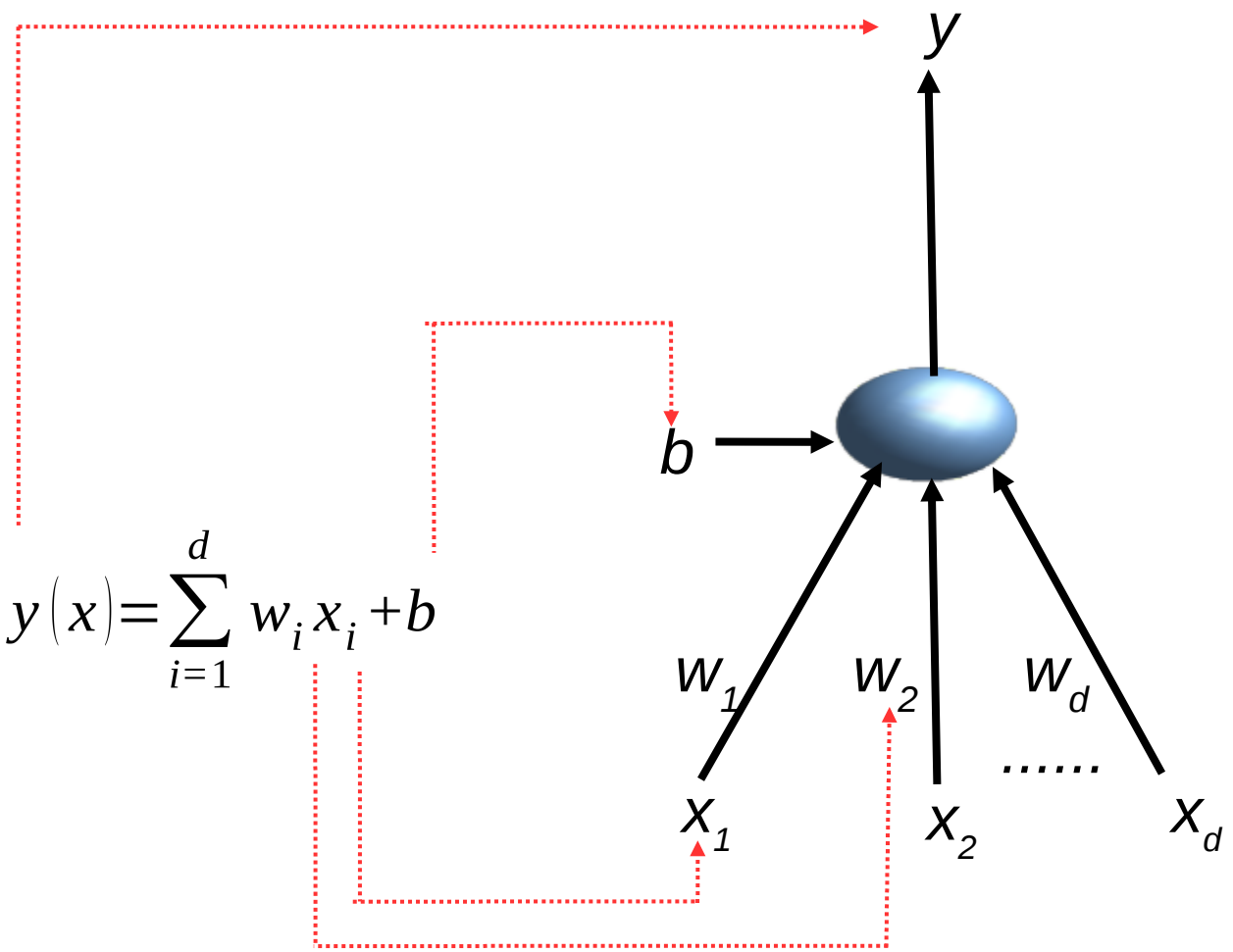


http://collaboratescience.com/WDS/

Neural Network



Neural Network



Neural Network

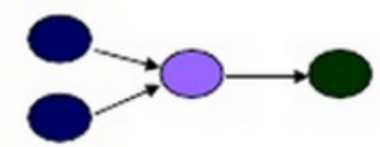
$$y(x) = \sum_{i=1}^{d} w_i x_i + b$$

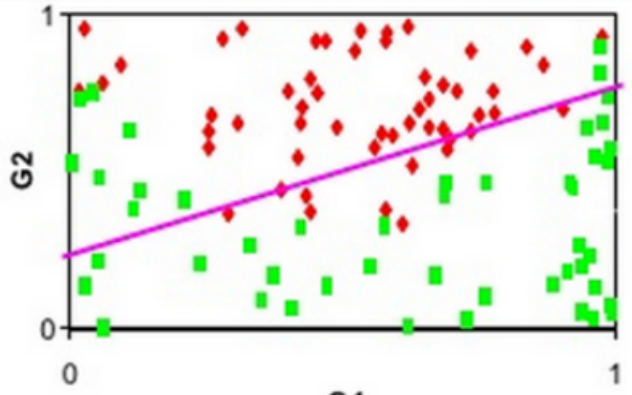
Dot product of two very large matrices

```
layerInput = self.weights[i].dot(input.T[i])
```

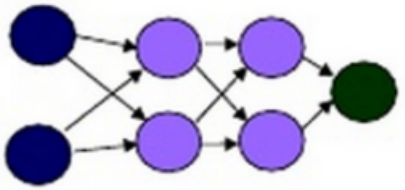
Neural Network

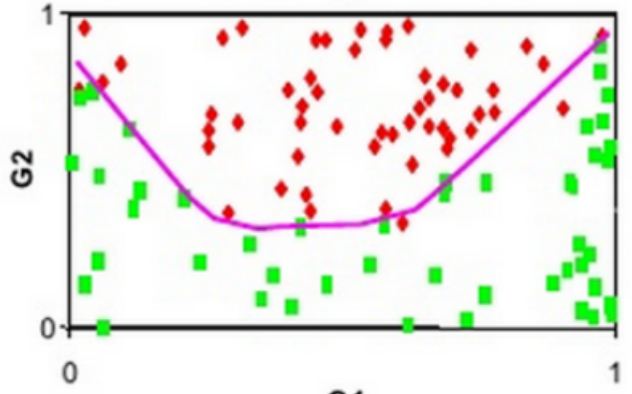
1 neuron



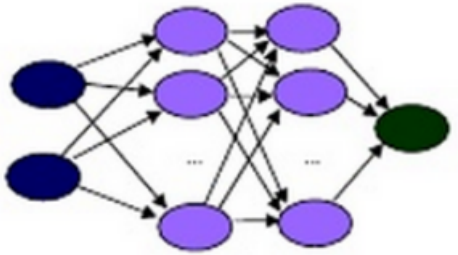


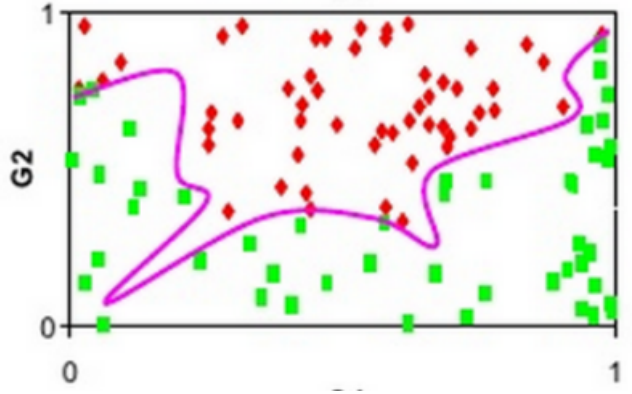
2 +2 neurons



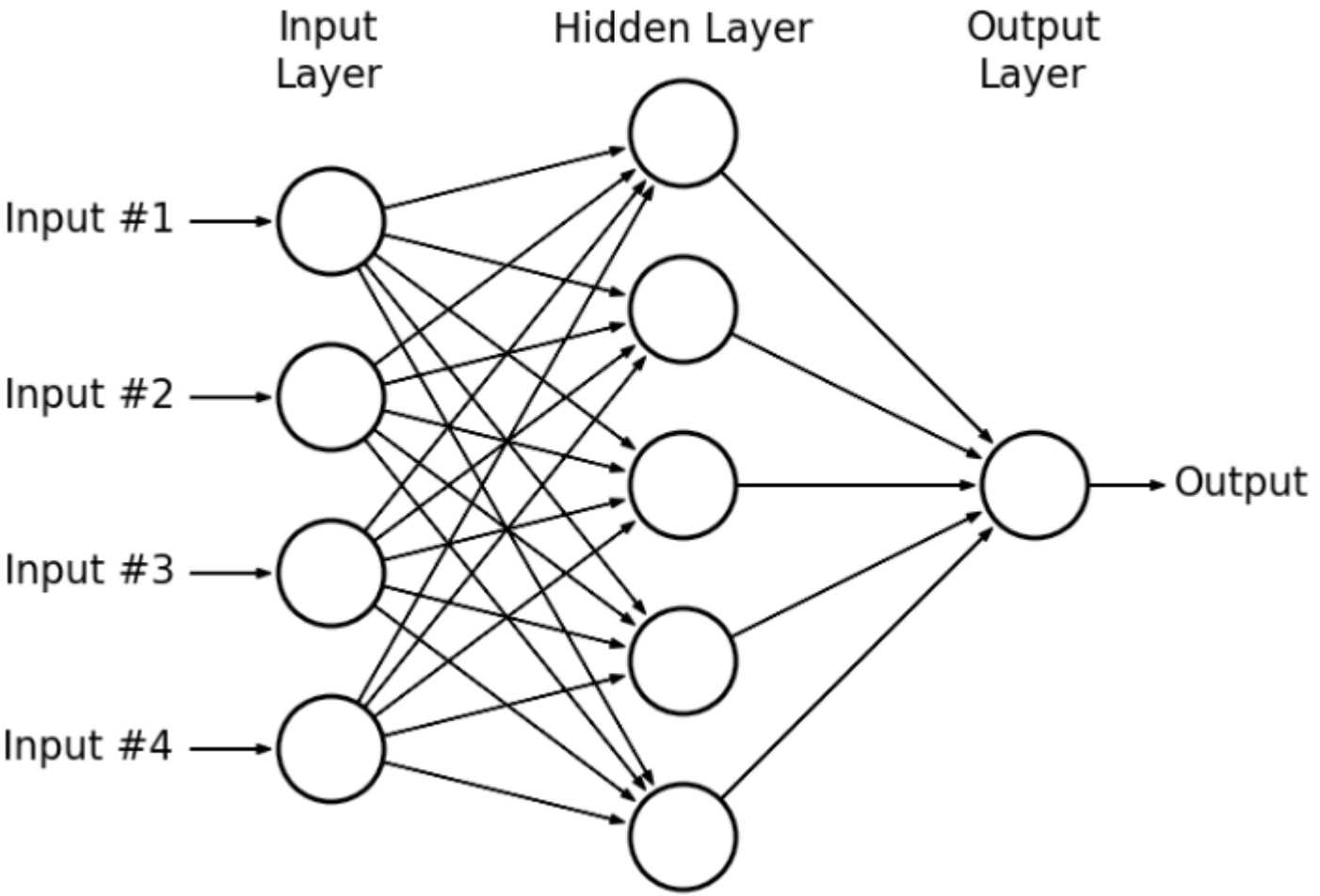


10 +10 neurons

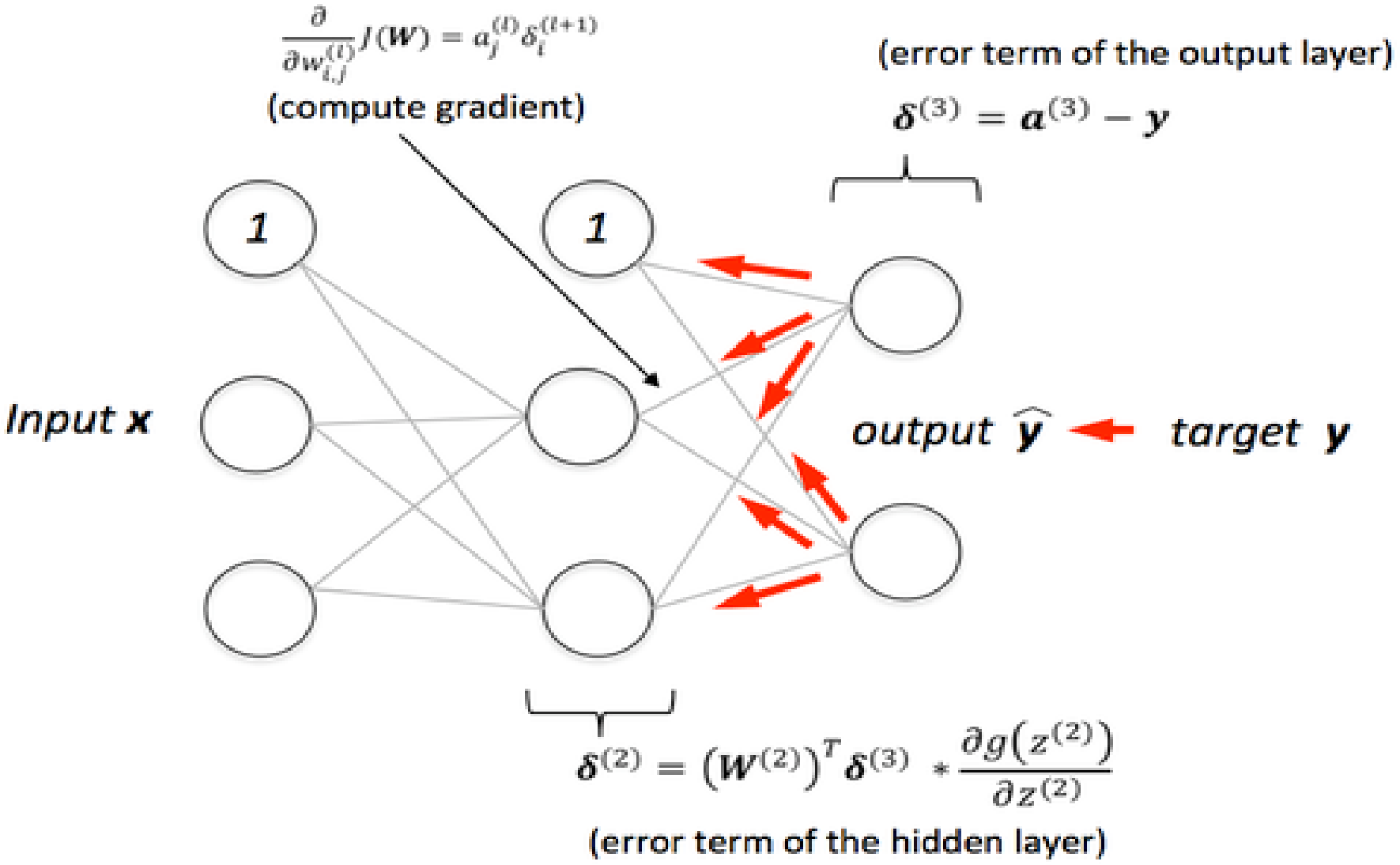




Neural Network



Neural Network



Neural Network

The Error Function

$$E = \frac{1}{2}(t - y)^2,$$

E = the squared error

t = target output

y = actual output of the output neuron

The Actual Output Function

$$y = \sum_{i=1}^{n} w_i x_i$$

n = the number of input units to the neuron

w_i = the i^{th} weight

x_i = the i^{th} input value to the neuron

Neural Network

Backpropagation

$$\frac{\partial E}{\partial w_i} = \frac{dE}{dy}\frac{dy}{d\text{net}}\frac{\partial \text{net}}{\partial w_i}$$

$\frac{\partial E}{\partial w_i}$ = How the error changes when the weights are changed

$\frac{dE}{dy}$ = How the error changes when the output is changed

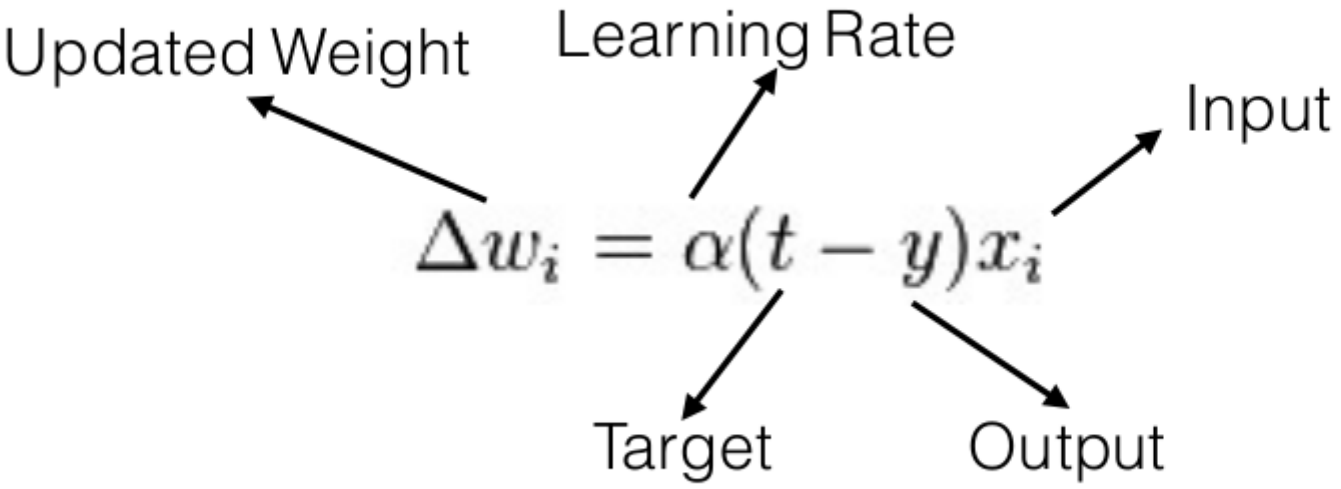
$\frac{dy}{d\text{net}}$ = How the output changes when the weighted sum changes

$\frac{\partial \text{net}}{\partial w_i}$ = How the weighted sum changes as the weights change

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

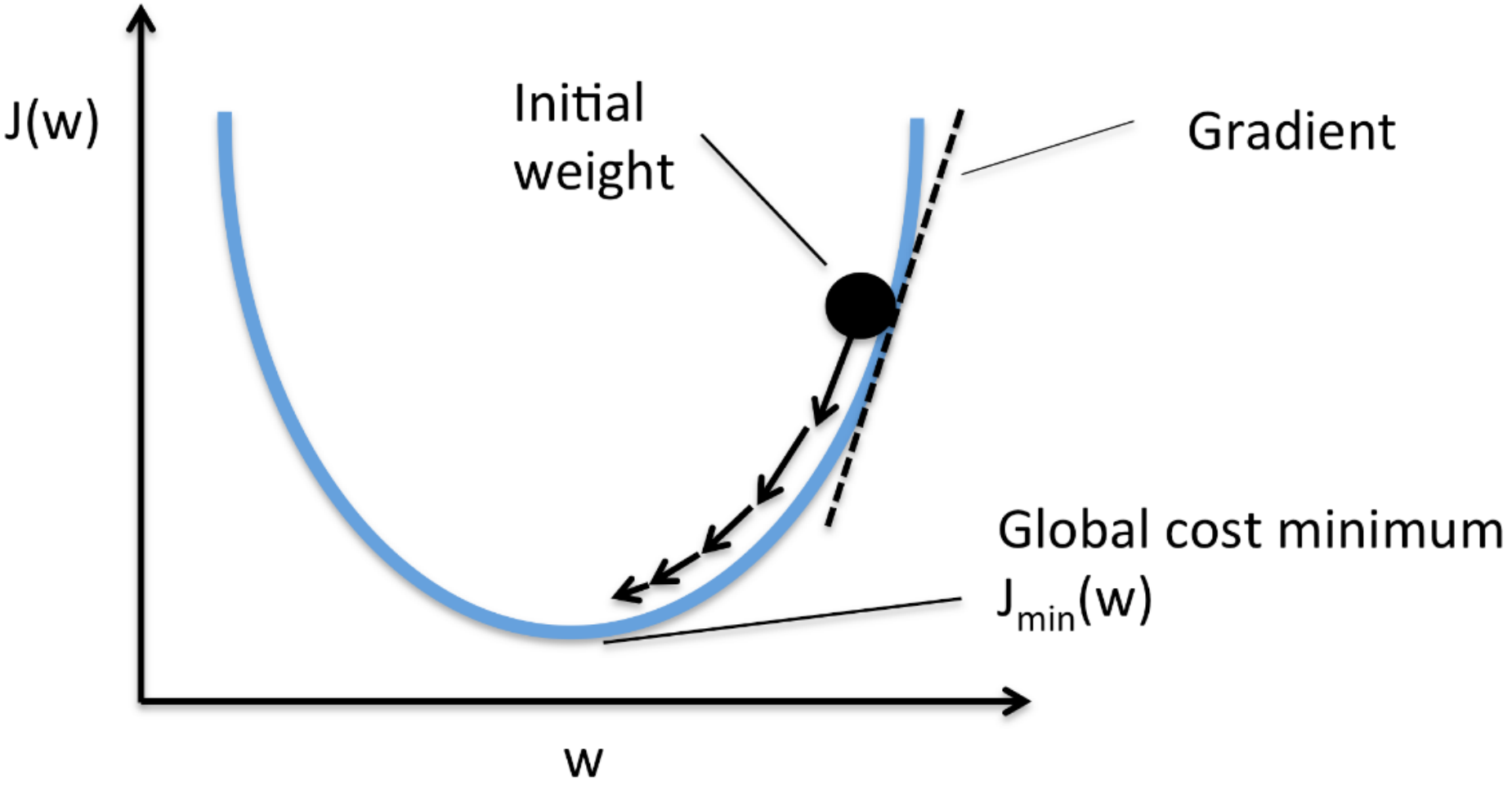
Neural Network

Backpropagation



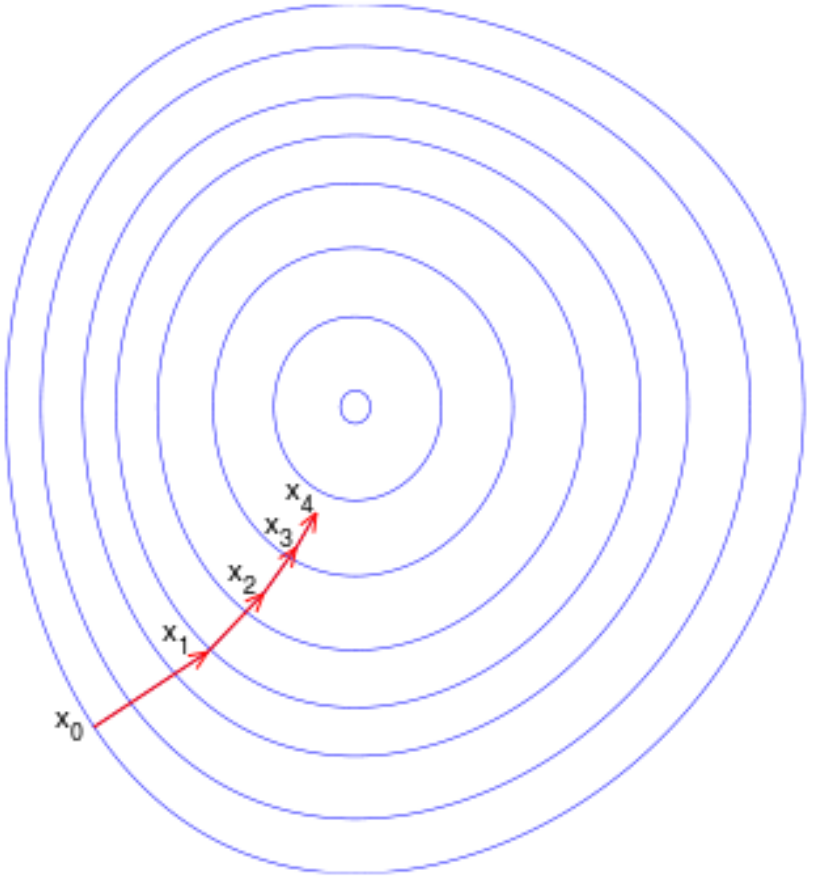
Neural Network

Backpropagation



Neural Network

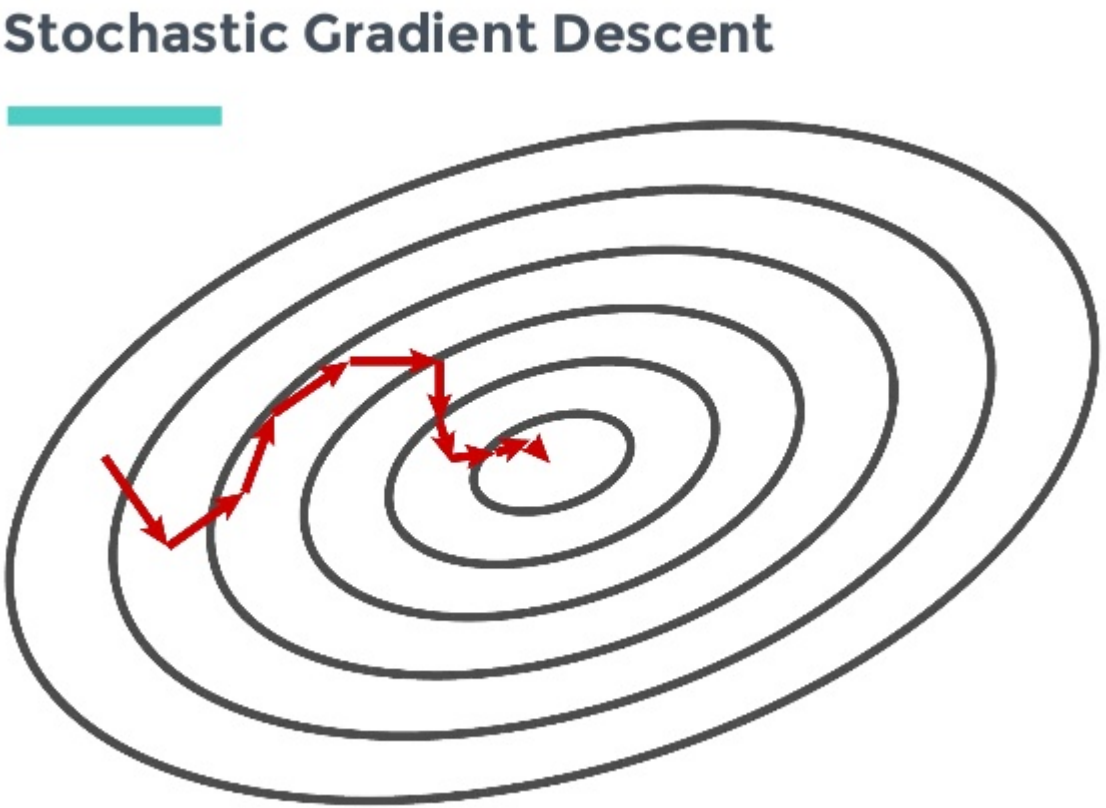
Backpropagation



Gradient Descent

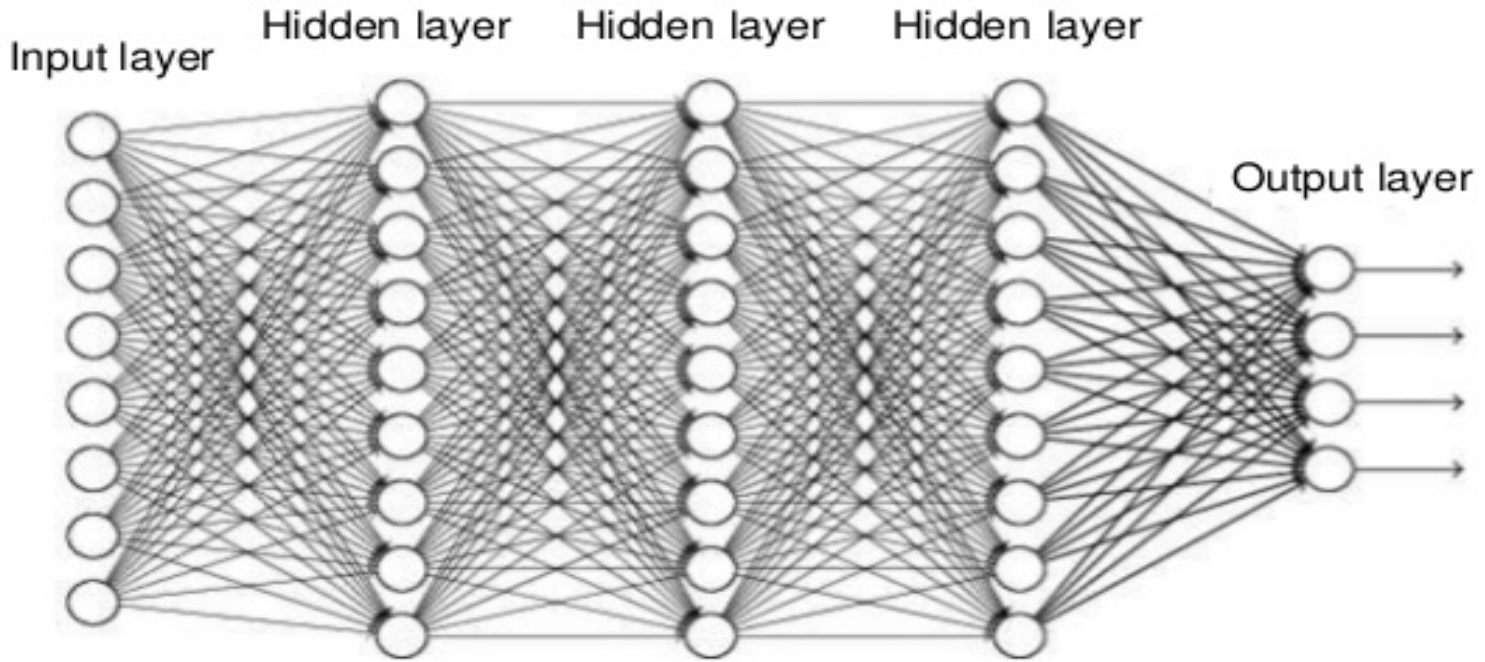
Neural Network

Backpropagation



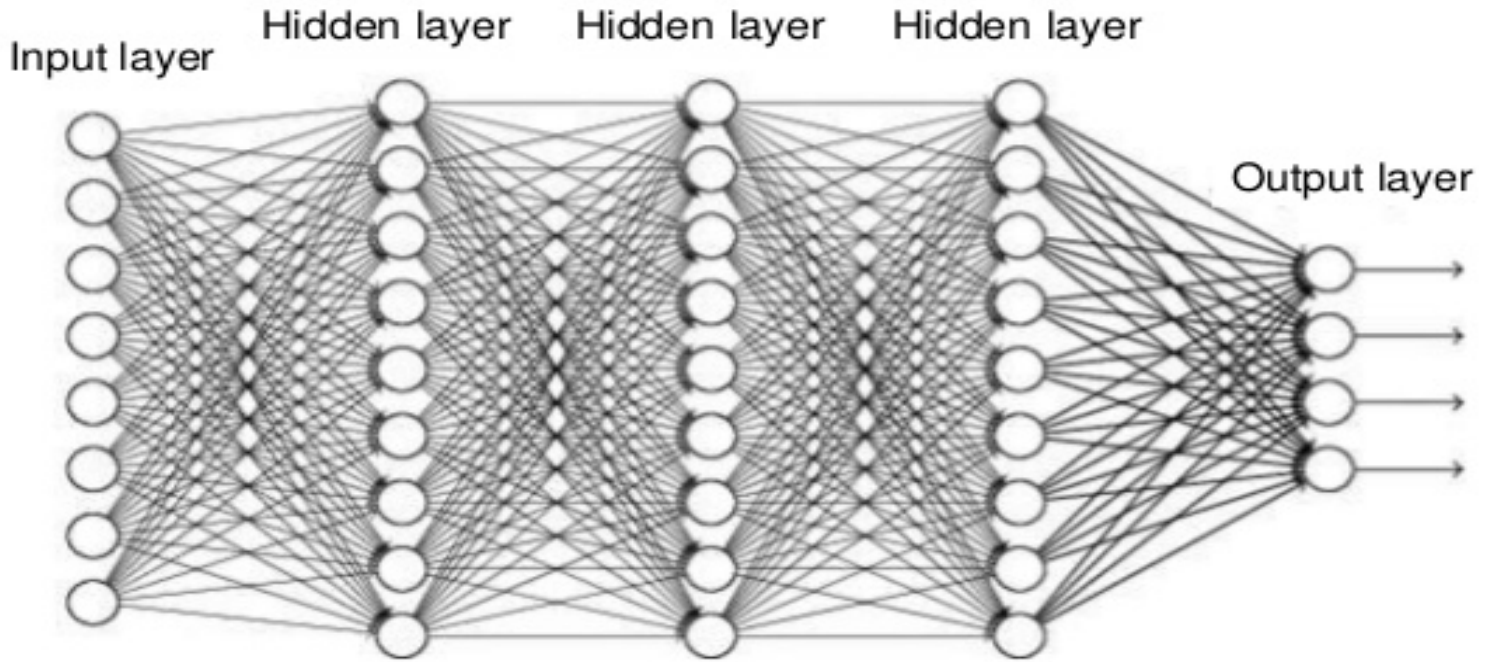
Neural Network

Backpropagation



Neural Network

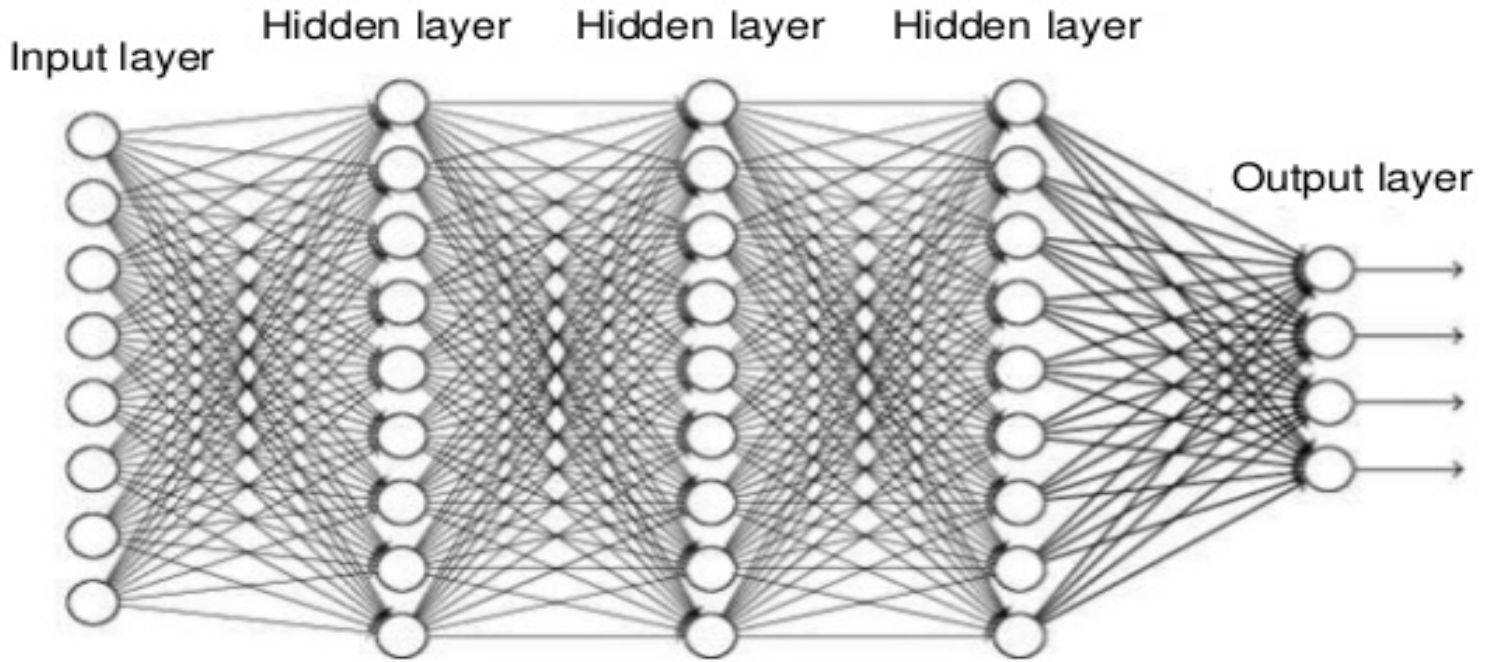
Backpropagation



Gradient decreases

Neural Network

Backpropagation



Gradient decreases

Vanishing Gradient

Neural Network

- Hinton, G. E., Osindero, S., & Teh, Y.W. (2006). A fast learning algorithm for deep belief nets.

Geoffrey Hinton
University of Toronto

Neural Network

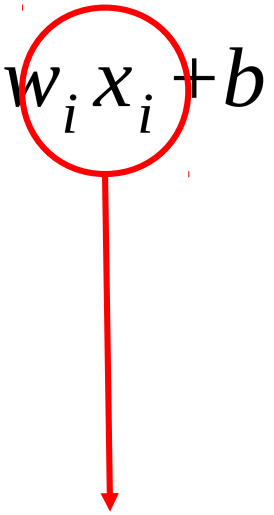
$$y(x) = \sum_{i=1}^{d} w_i x_i + b$$

Dot product of two very large matrices

```
layerInput = self.weights[i].dot(input.T[i])
```

Convolutional Neural Network

$$y(x) = \sum_{i=1}^{d} w_i x_i + b$$



Convolution of two very large matrices

```
y = scipy.signal.convolve(input[i,:],w[i,:])
```

OR

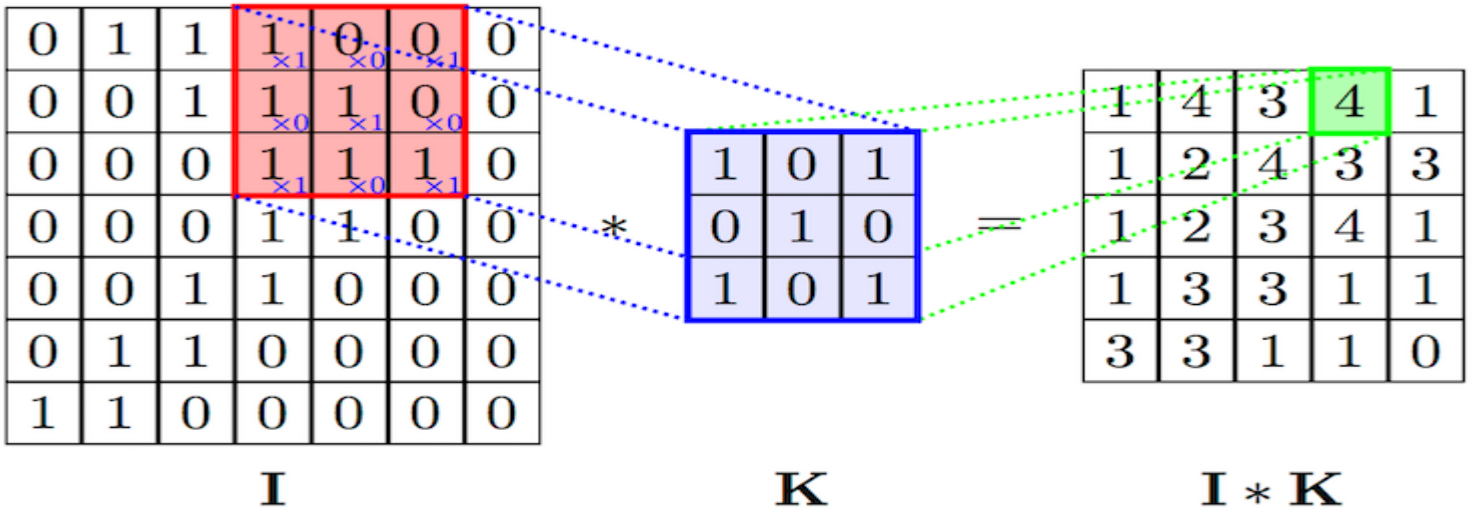
```
y = scipy.signal.convolve2d(input[:,:,i],w[:,:,i,j])
```

Convolutional Neural Network

Motivation

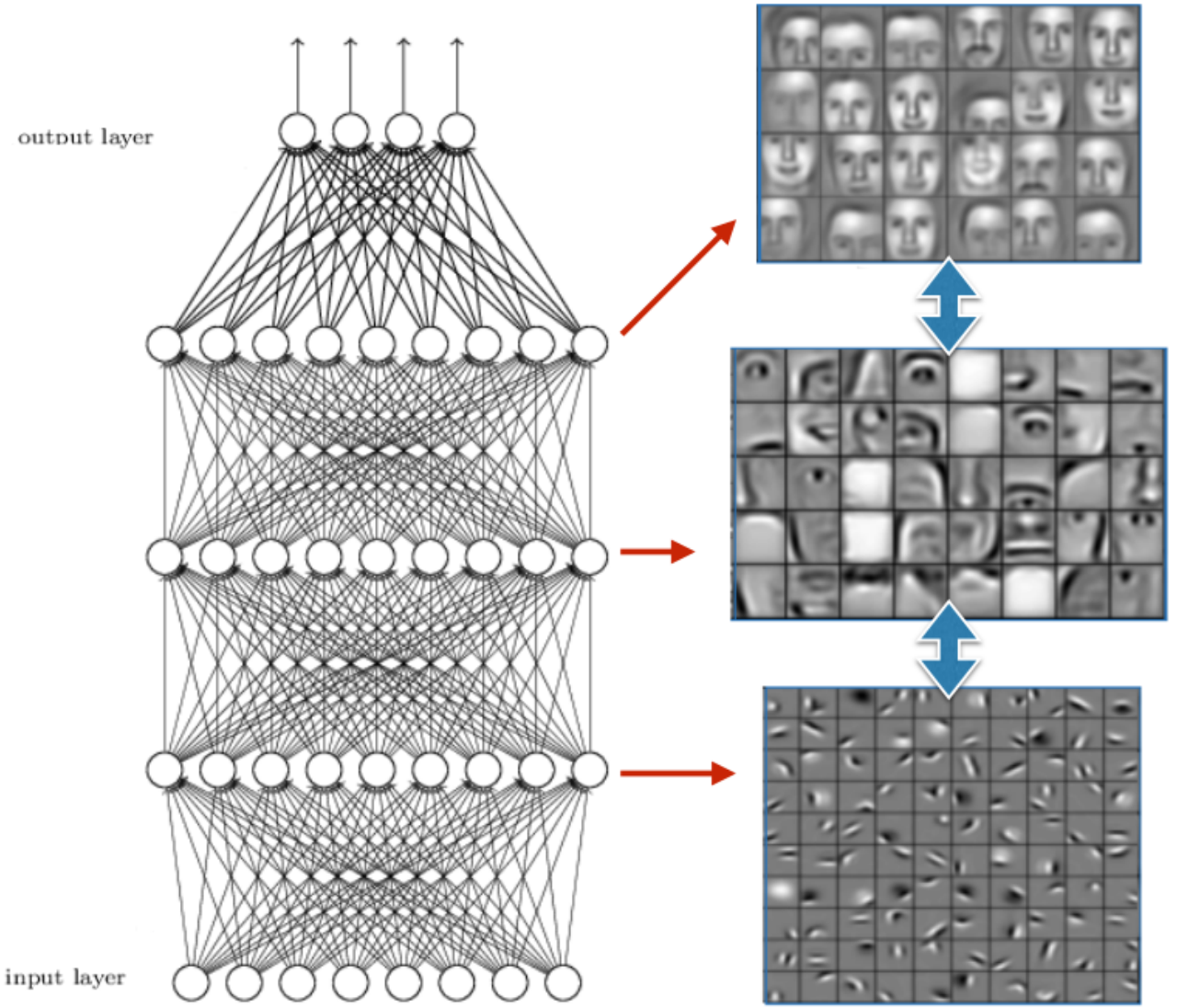
Convolutional Neural Network

Motivation



I **K** **I ∗ K**

Convolutional Neural Network

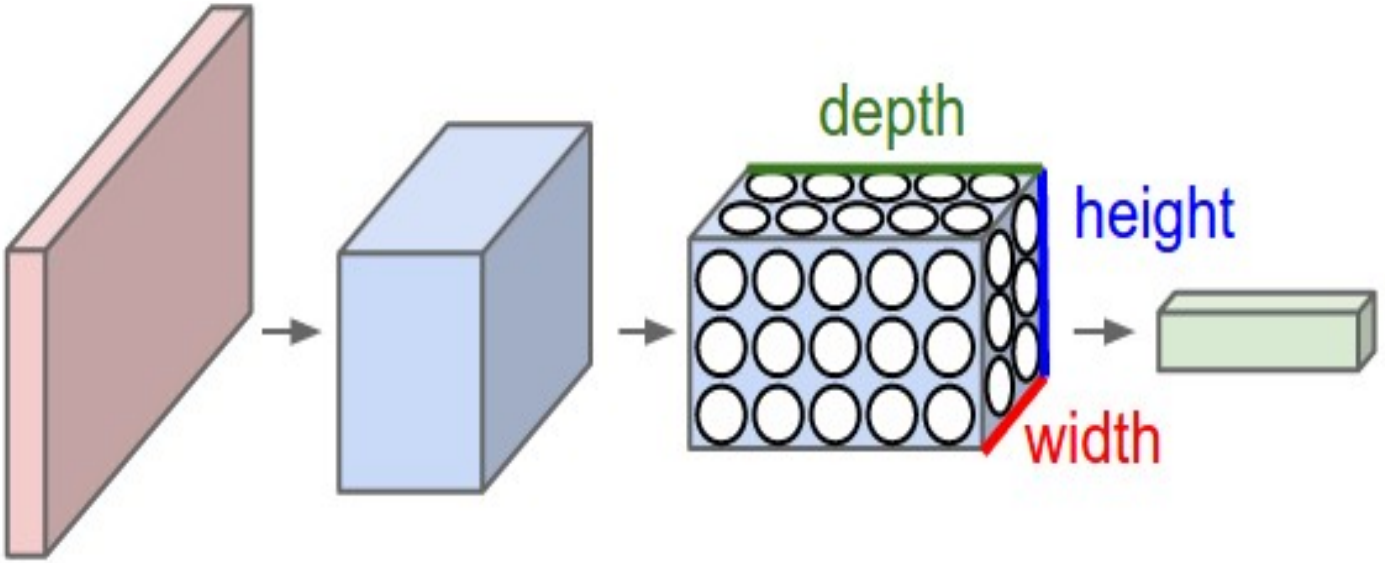


Hierarchy of Features

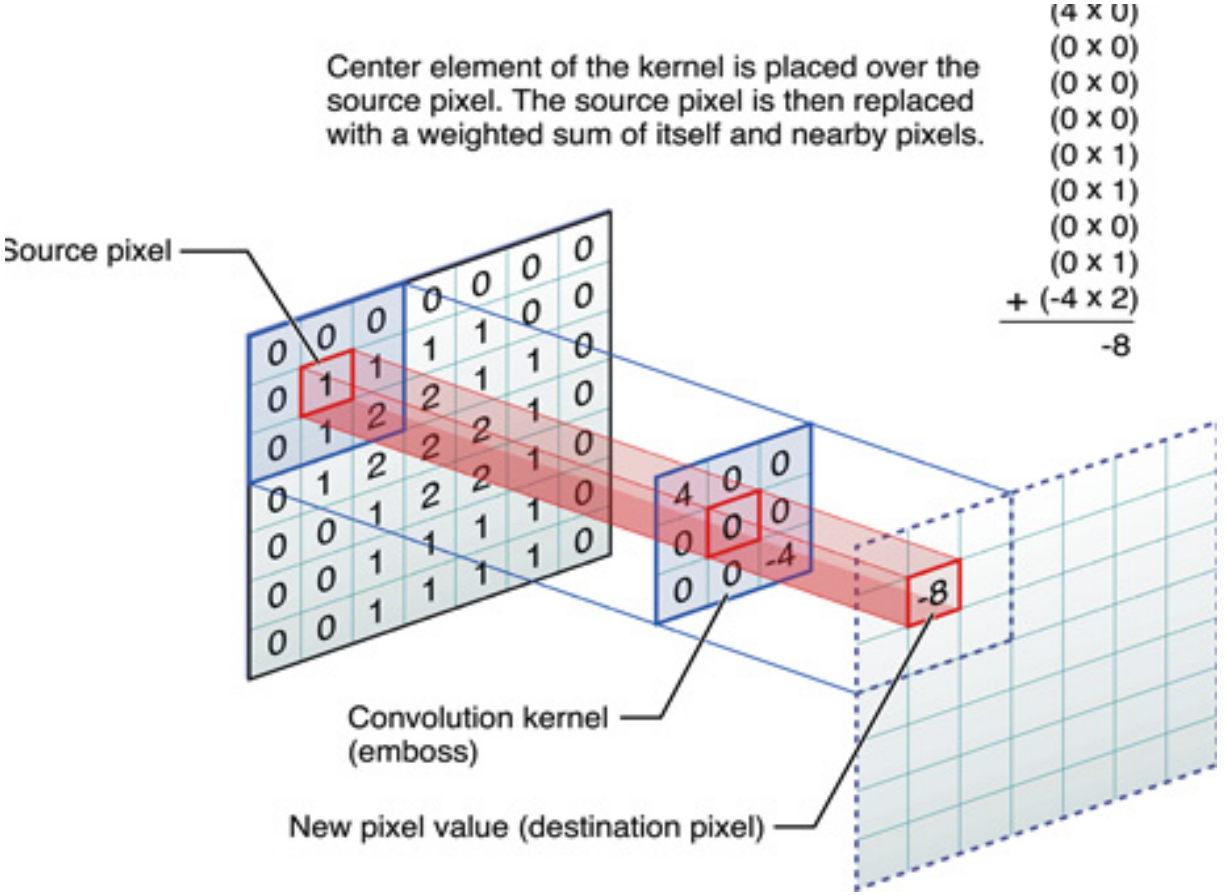
Convolutional Neural Network

Yan LeCun - ConvNets (1996)

Convolutional Neural Network

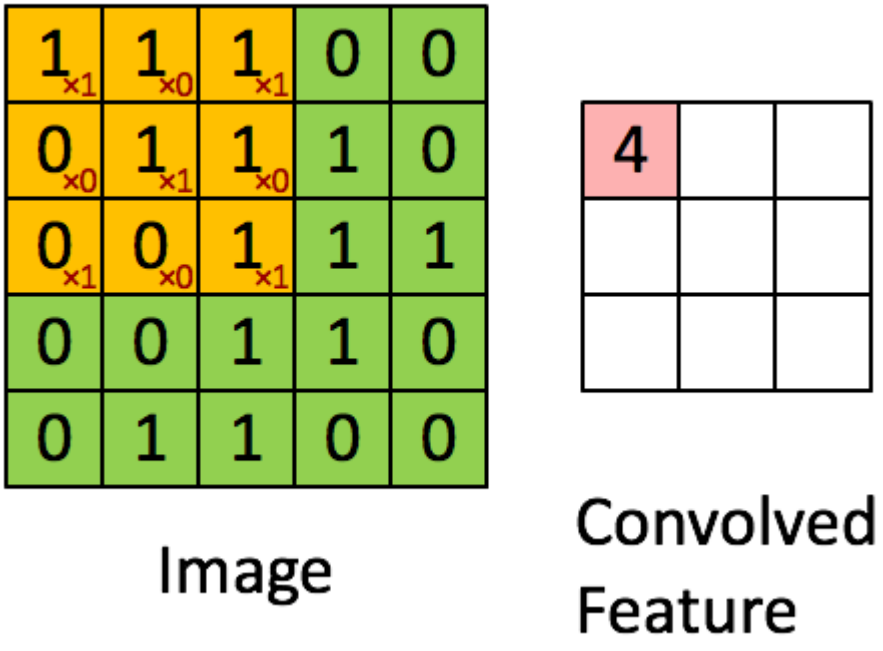


Convolutional Neural Network

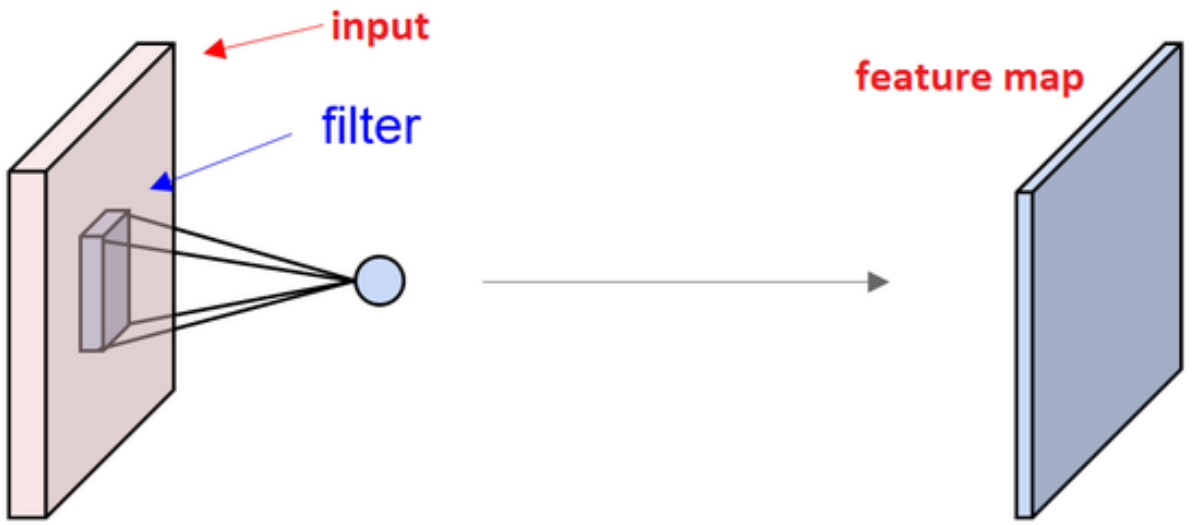


Canny Kernel

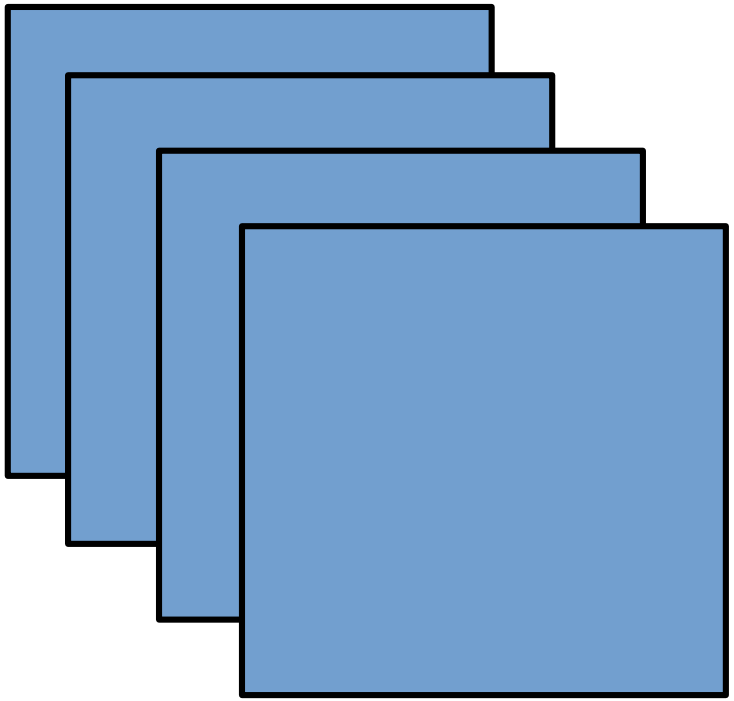
Convolutional Neural Network



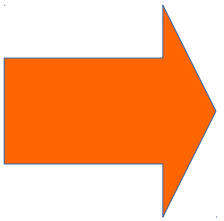
Convolutional Neural Network

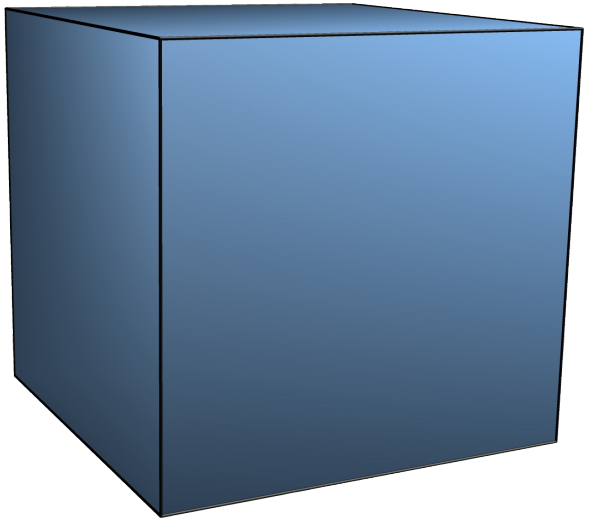


Convolutional Neural Network



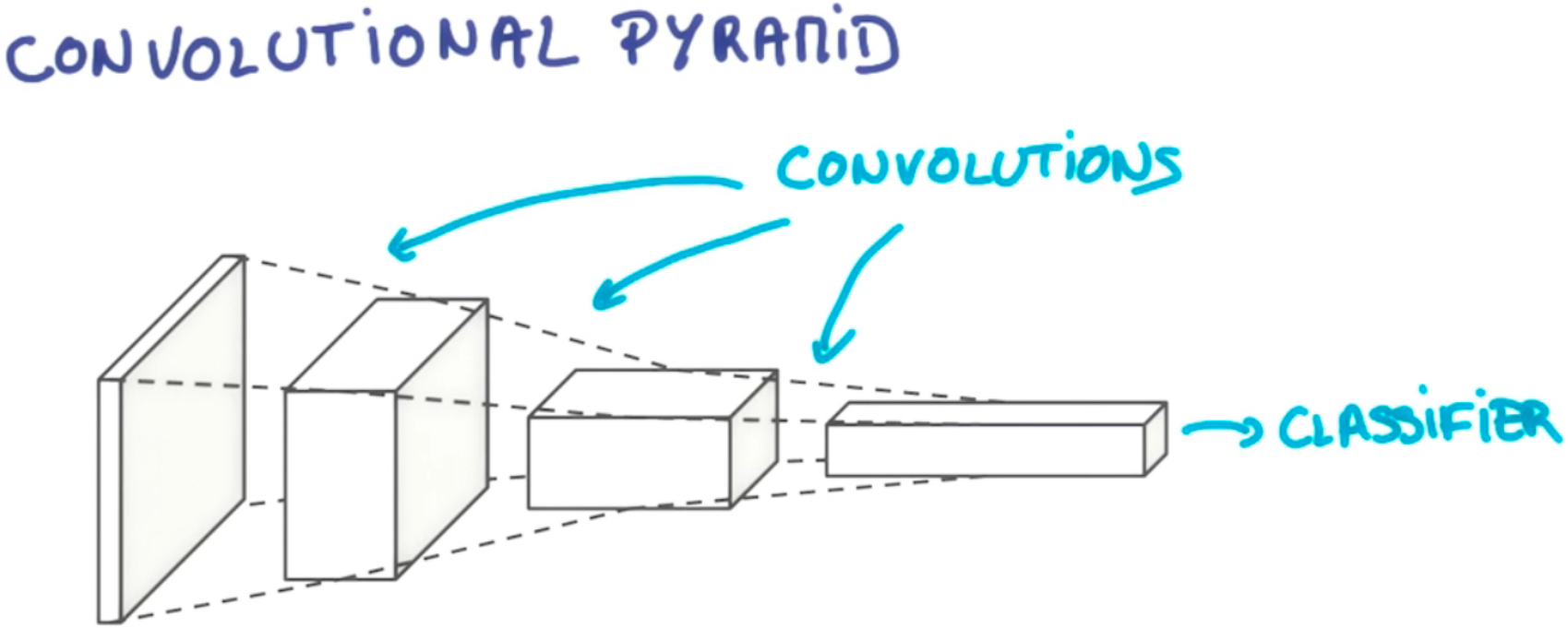
4 feature maps with size 32x32





Collectively represented as 4 x 32 x32

Convolutional Neural Network



Forward propogation

Convolutional Neural Network

CNNs have three main components :

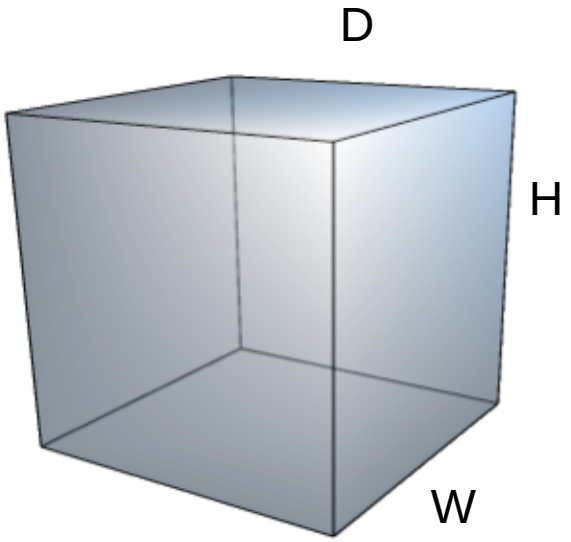
1) ***Convolutional Layers***

2) Pooling layer

3) Fully Connected layer

Spatial Size of Output Volume

W x H x D



Spatial Size of Output Volume

W x H x D

$$SZ = \frac{(W - F + 2P)}{S} + 1$$

W = size of input

F = size of receptive field

P = padding size

S = stride

Spatial Size of Output Volume

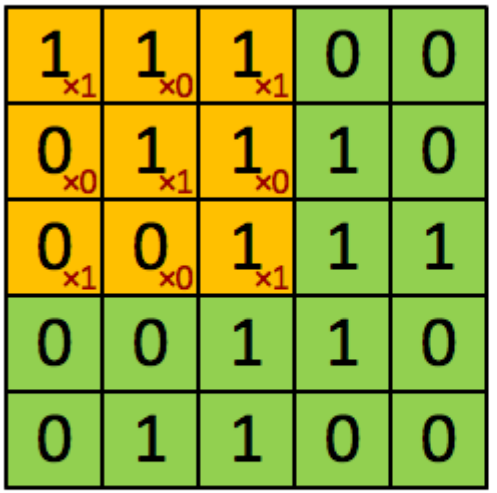
$$SZ = \frac{(W - F + 2P)}{S} + 1$$

W = size of input

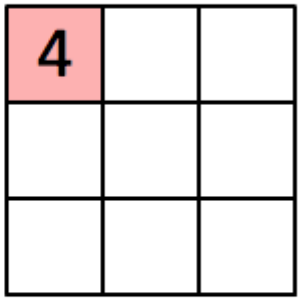
F = size of receptive field

P = padding size

S = stride



Image



Convolved Feature

W = 5 (5x5)

F = 3 (3x3)

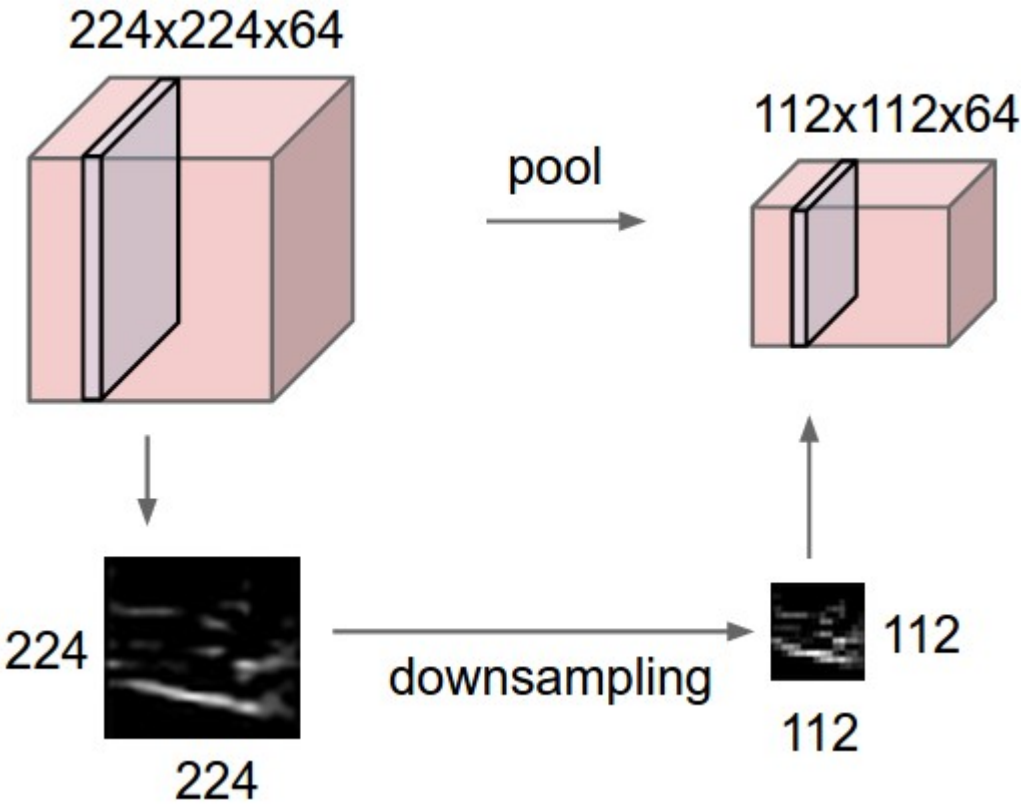
P = 0

S = 1

SZ = 3

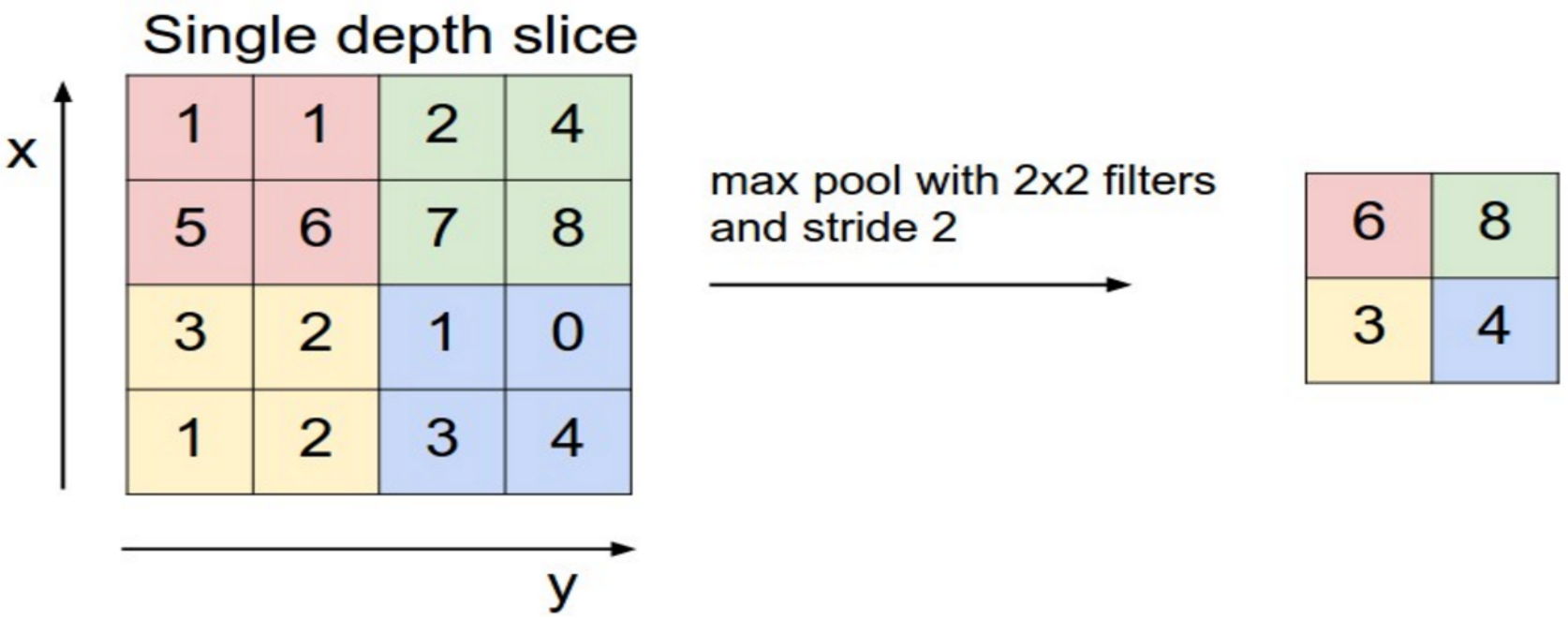
Pooling

Reduces *spatial size* of input by subsampling

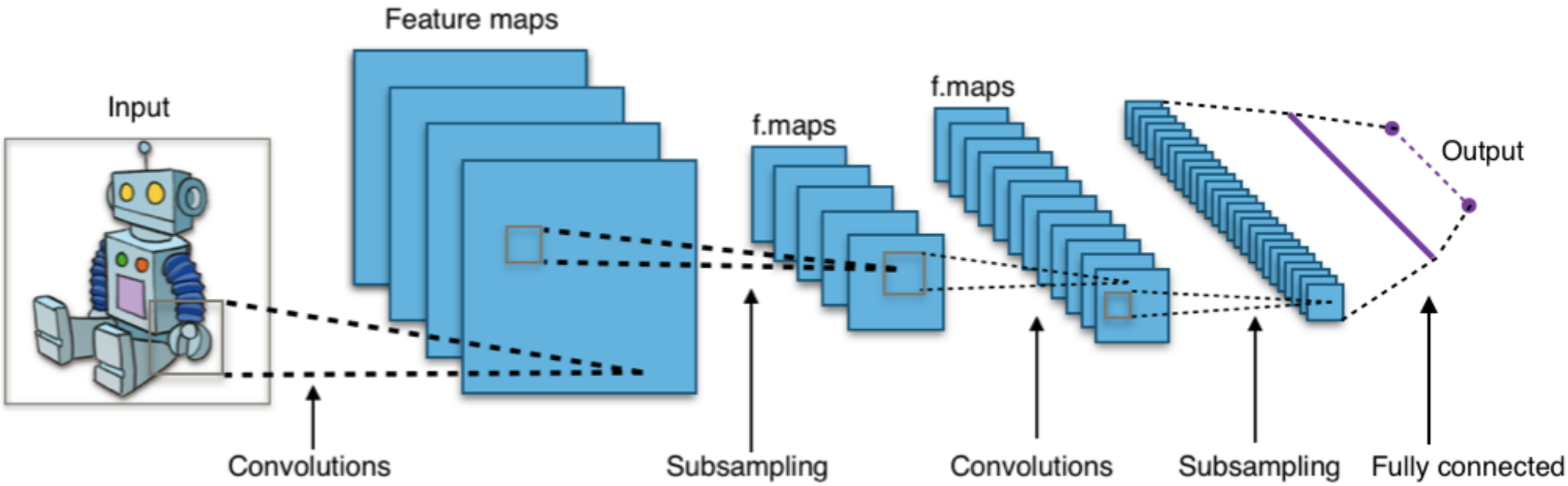


Pooling

Reduces *spatial size* of input by subsampling



Fully Connected Layer



Turn a volume array into a single vector

Fully Connected Layer

Convert spatial connections with volume to a single layer so it can go into a regular NN

Example : Ouput at one layer is 7x7x512

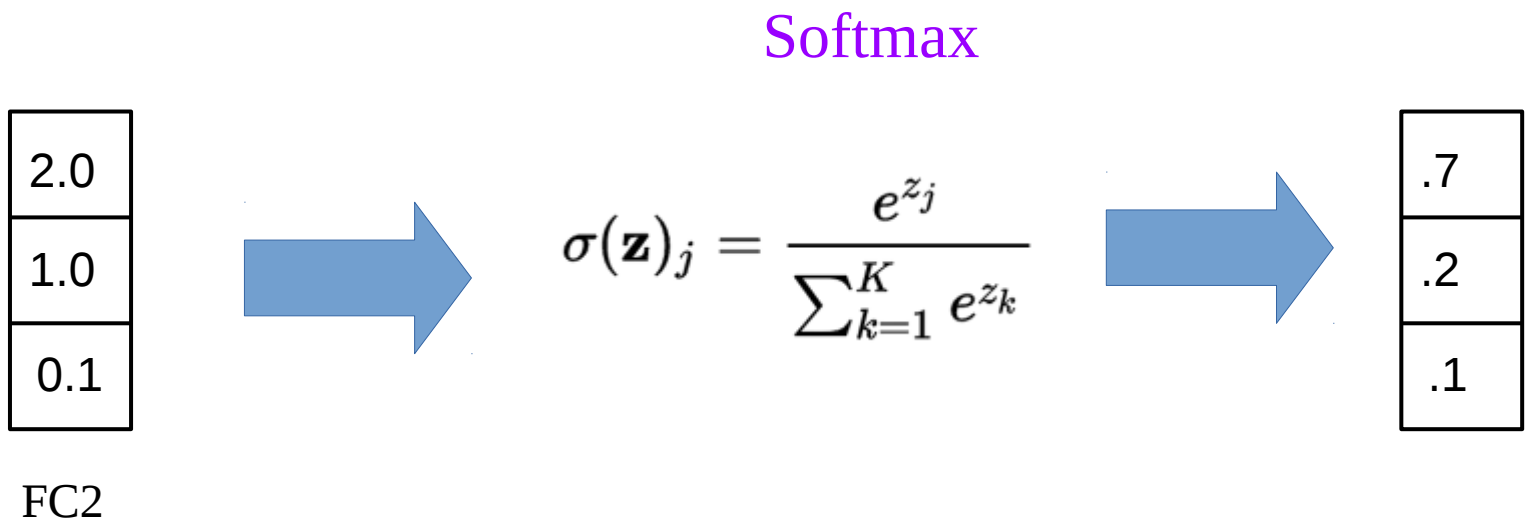
We need to input them into a NN of 1024 inputs

-- Use window of size 7x7 with stride = 1 and pad = 0

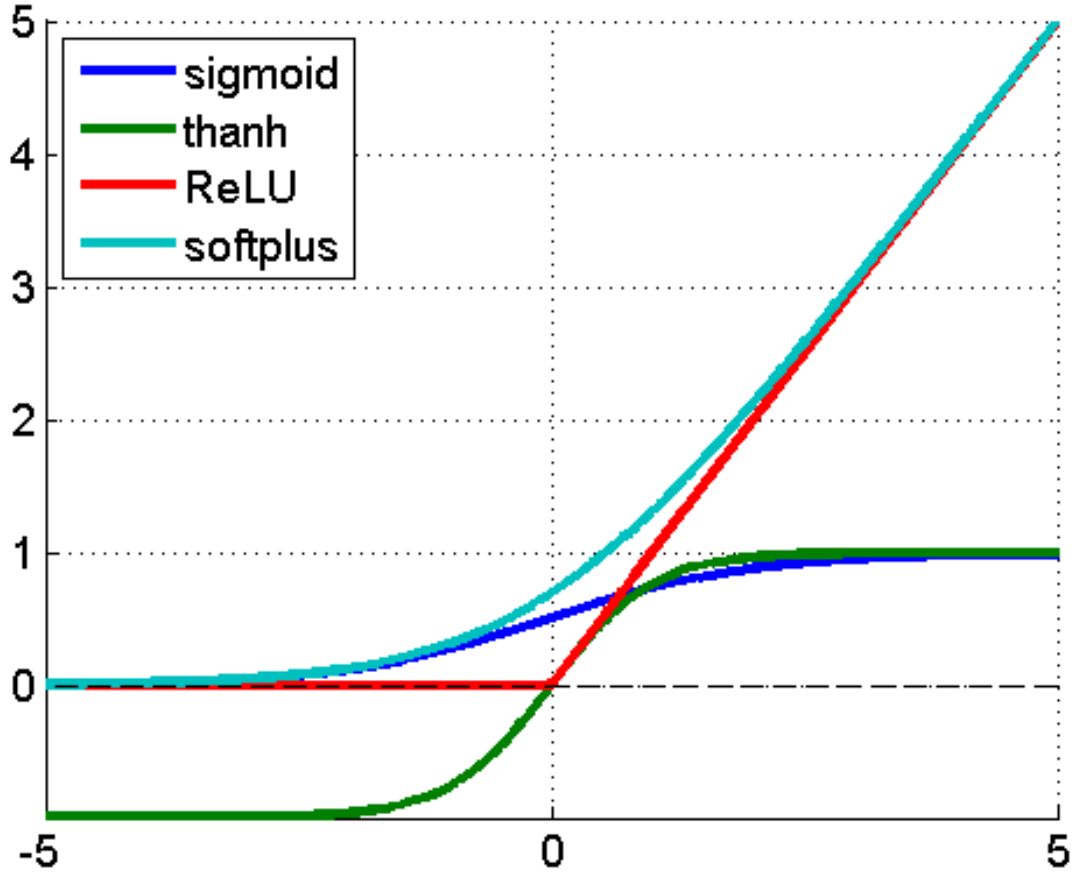
$$SZ = \frac{(W - F + 2P)}{S} + 1$$

SZ = 1x 1x 1024

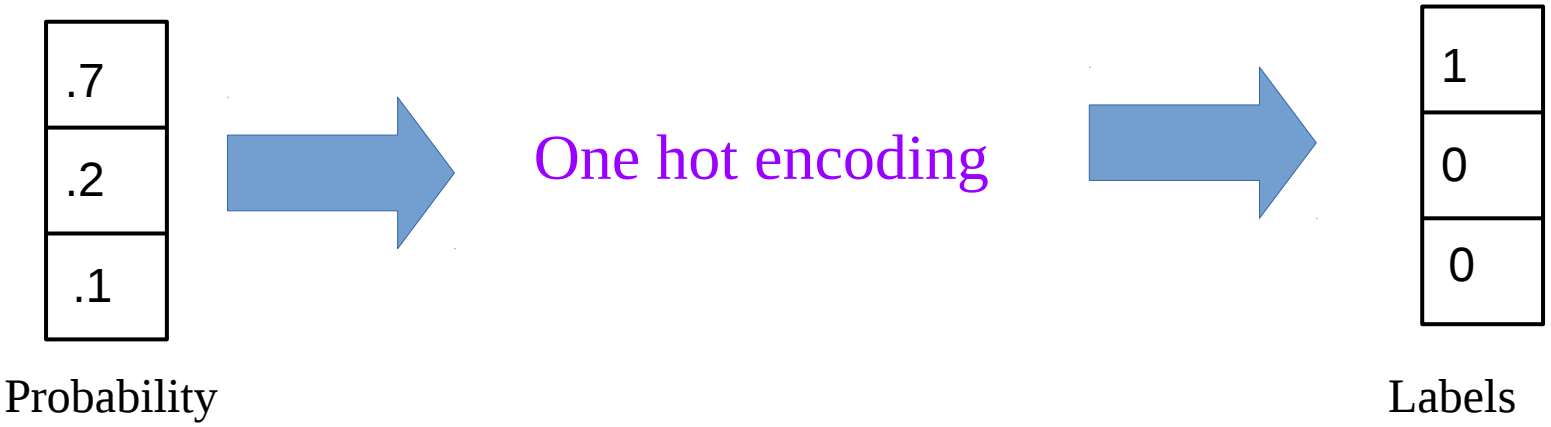
Scores to Probability



Activation Functions



Probability to Class



Data representation in Convnets

N samples, M channels, X,Y

For a single RGB image : (1,3,32,32)

For 10 feature maps with size 20,20 : (1,10,20,20)

Practical – Classify MNIST Dataset