

# Deep Learning – 101

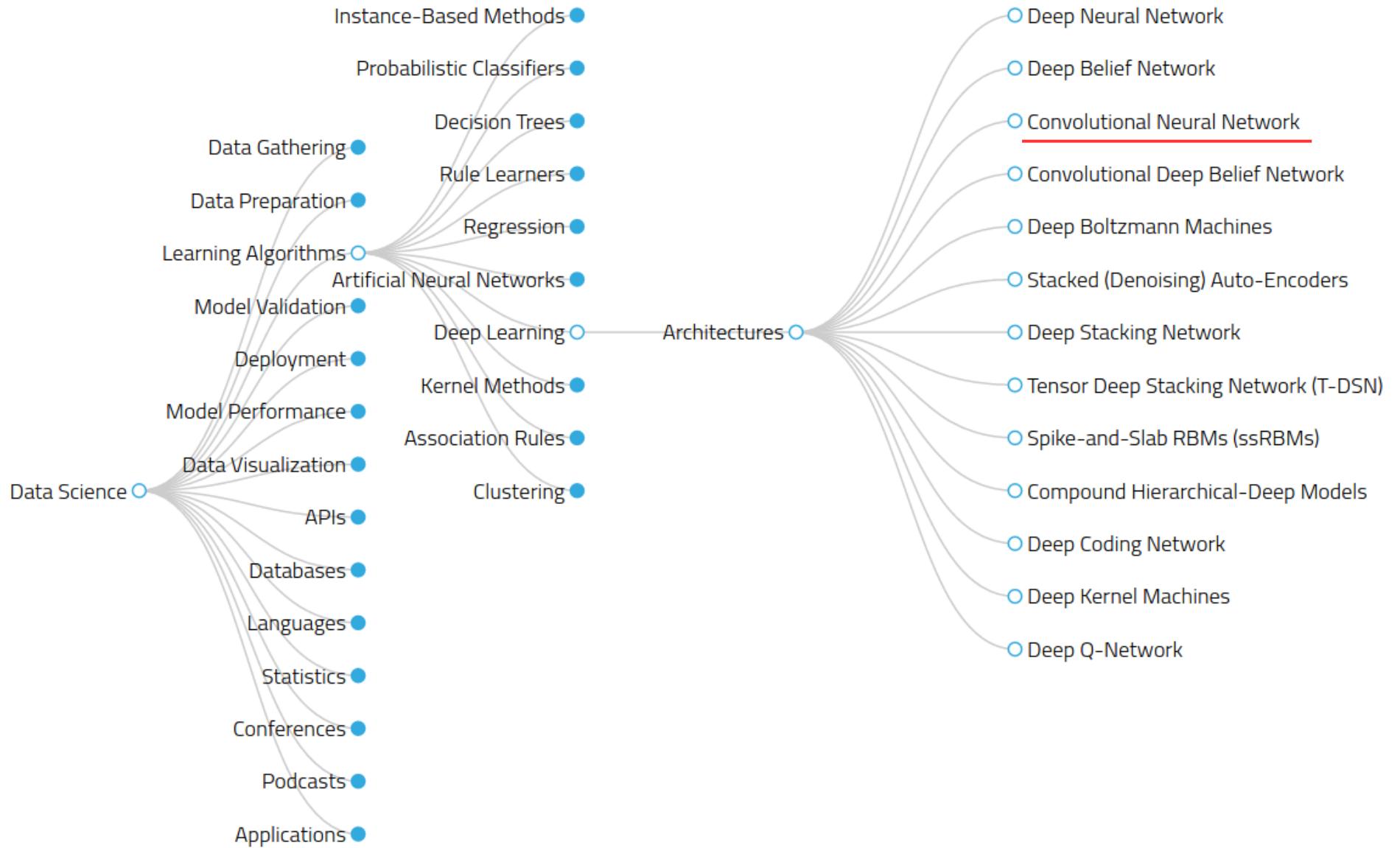
Arun Aniyan  
SKA SA / Rhodes University  
[arun@ska.ac.za](mailto:arun@ska.ac.za)



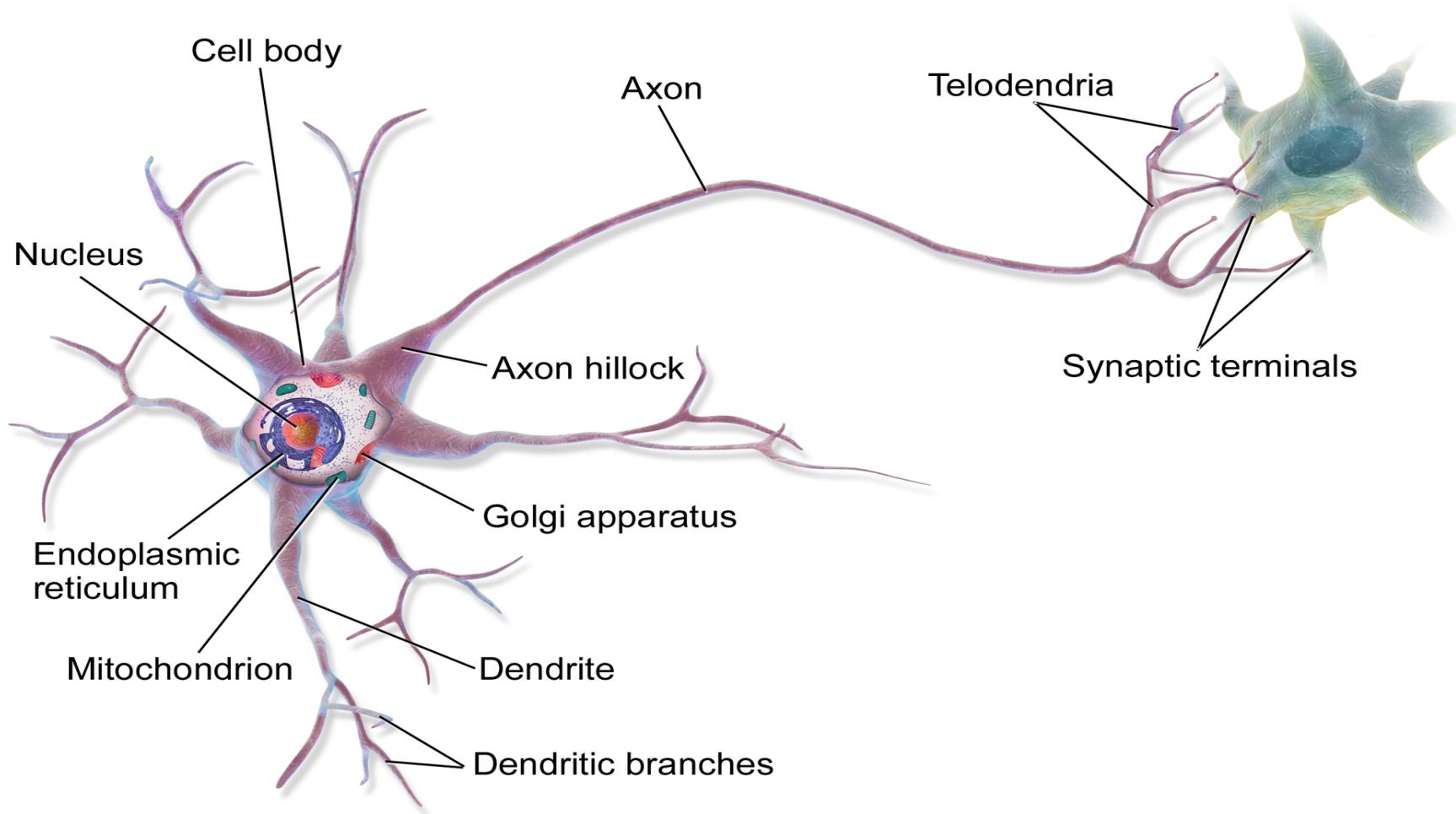
**WE NEED TO GO**

**DEEPER**

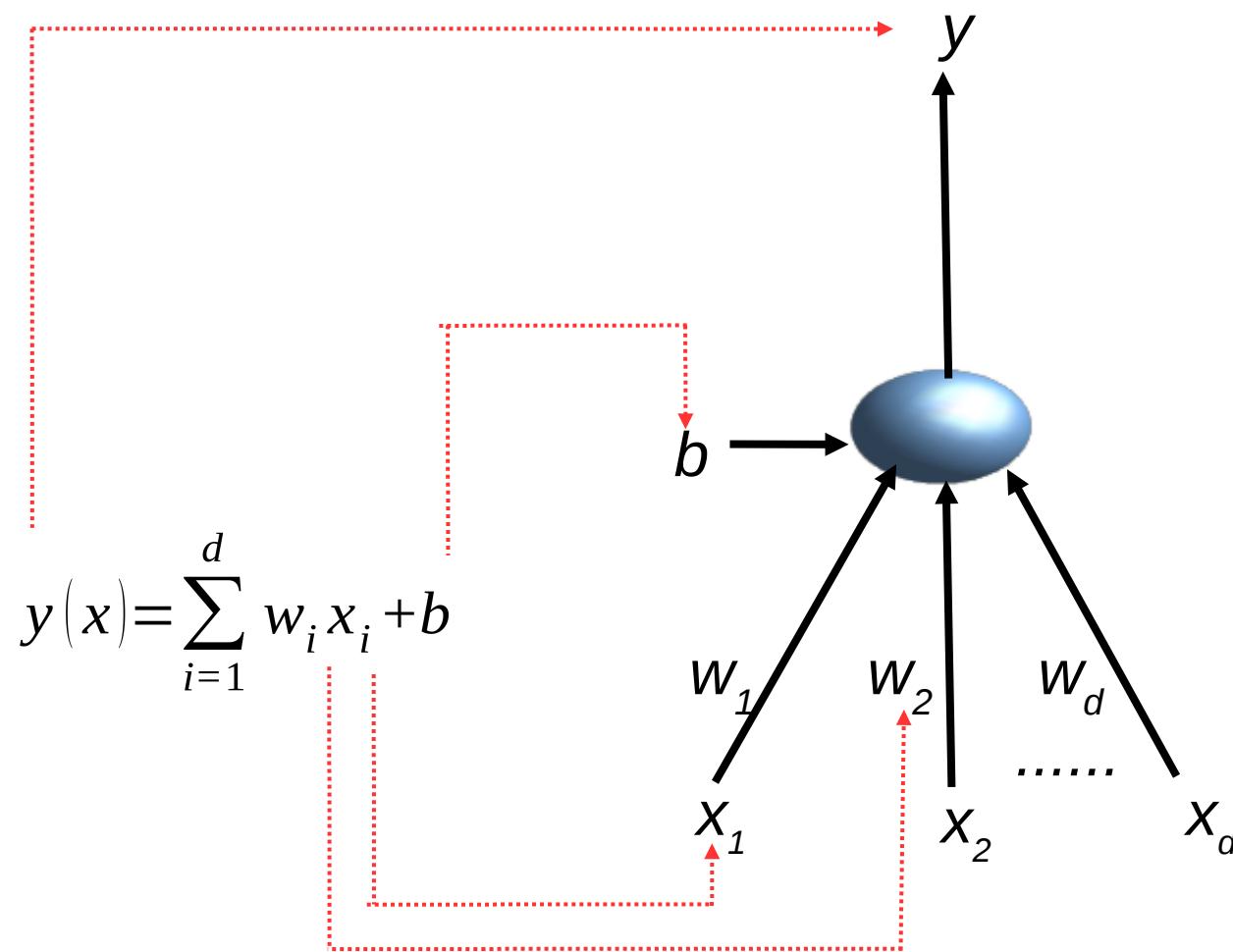
# Deep Learning Ontology



# Neural Network



# Neural Network



# Neural Network

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

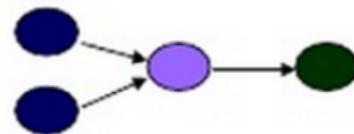


Dot product of two very large matrices

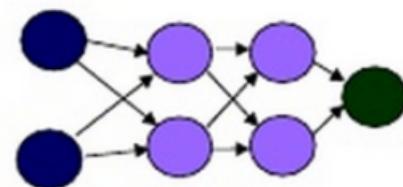
```
layerInput = self.weights[i].dot(input.T[i])
```

# Neural Network

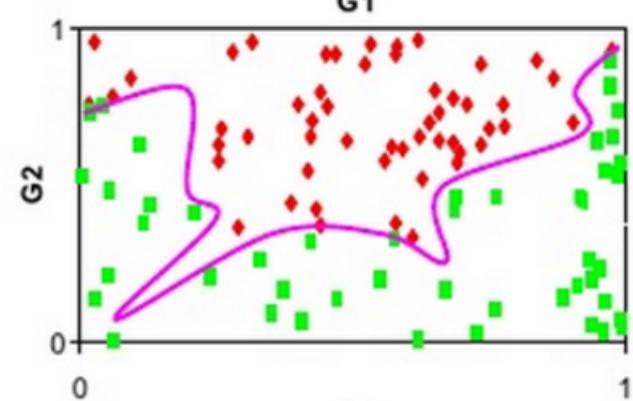
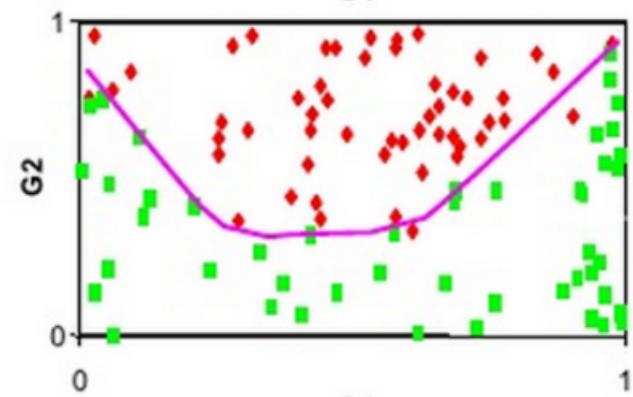
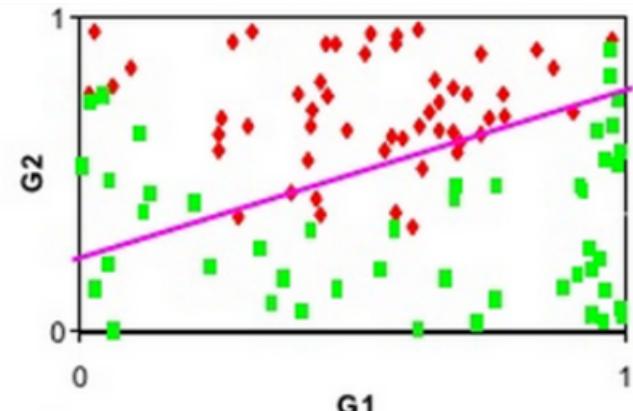
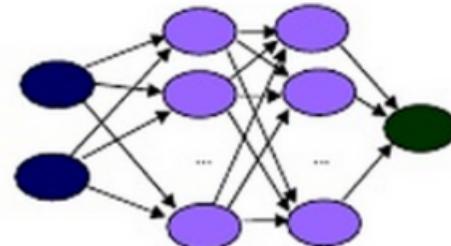
1 neuron



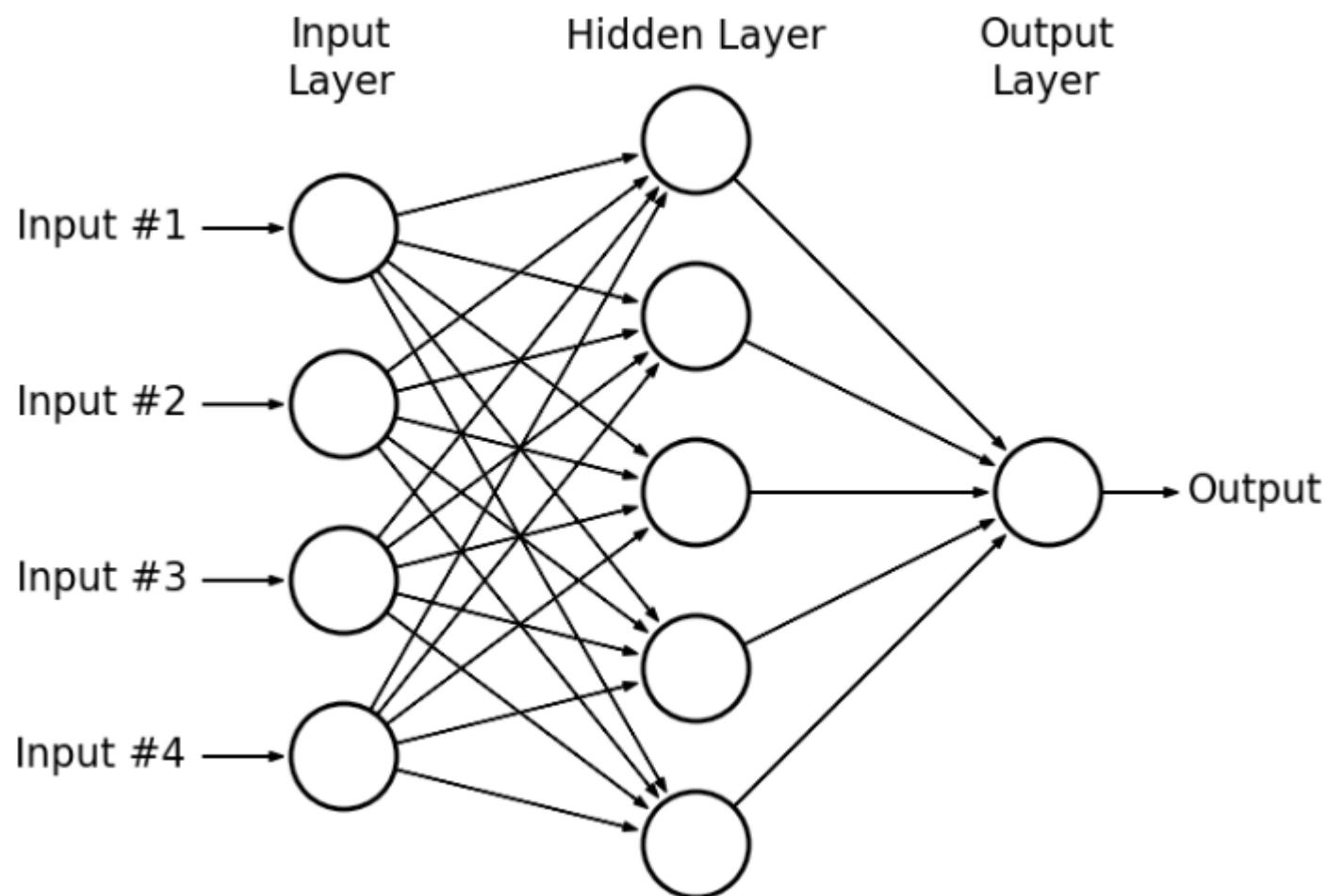
2 + 2 neurons



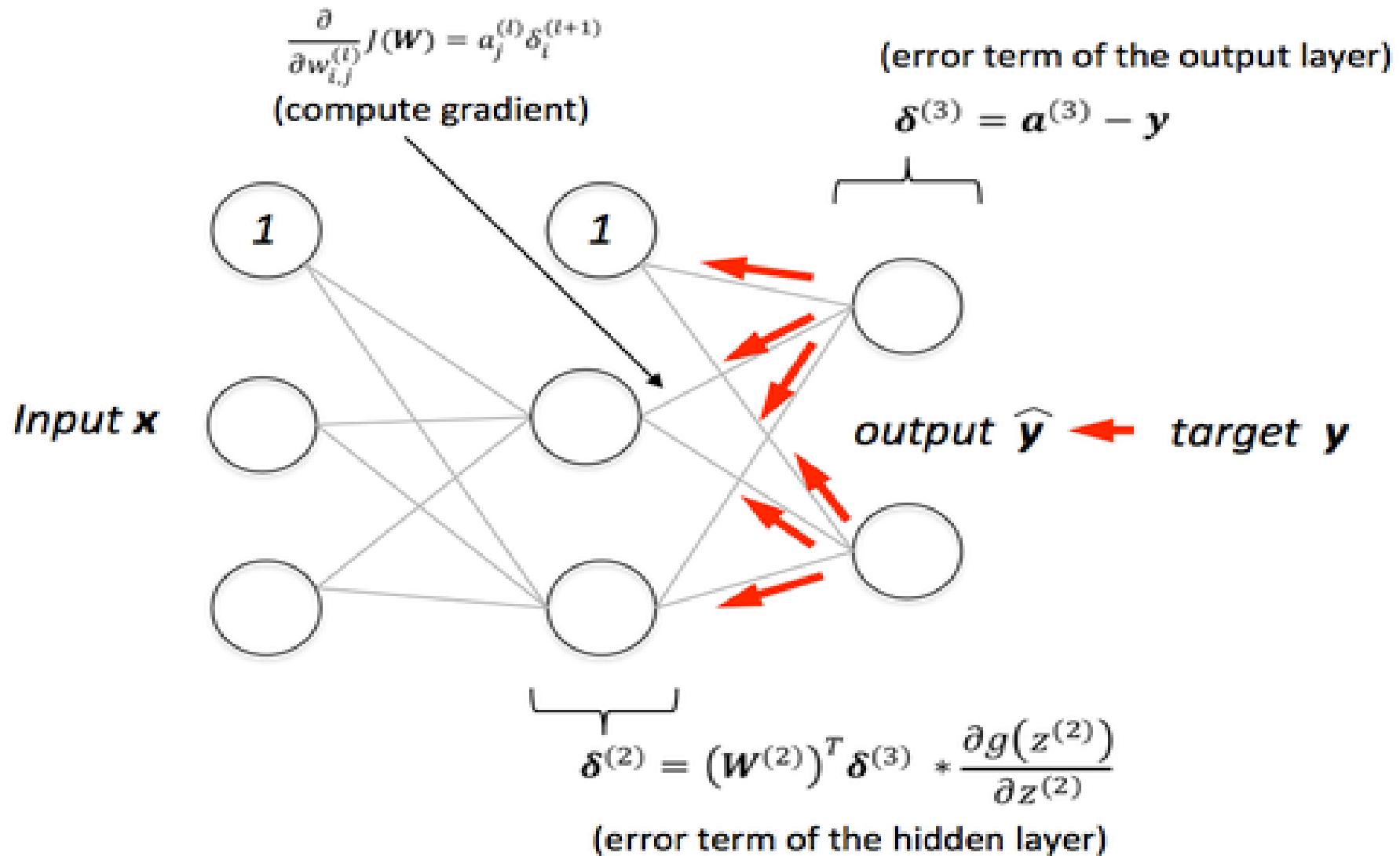
10 + 10 neurons



# 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<sup>1</sup>

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  $i^{\text{th}}$  weight

$x_i$  = the  $i^{\text{th}}$  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

# Neural Network

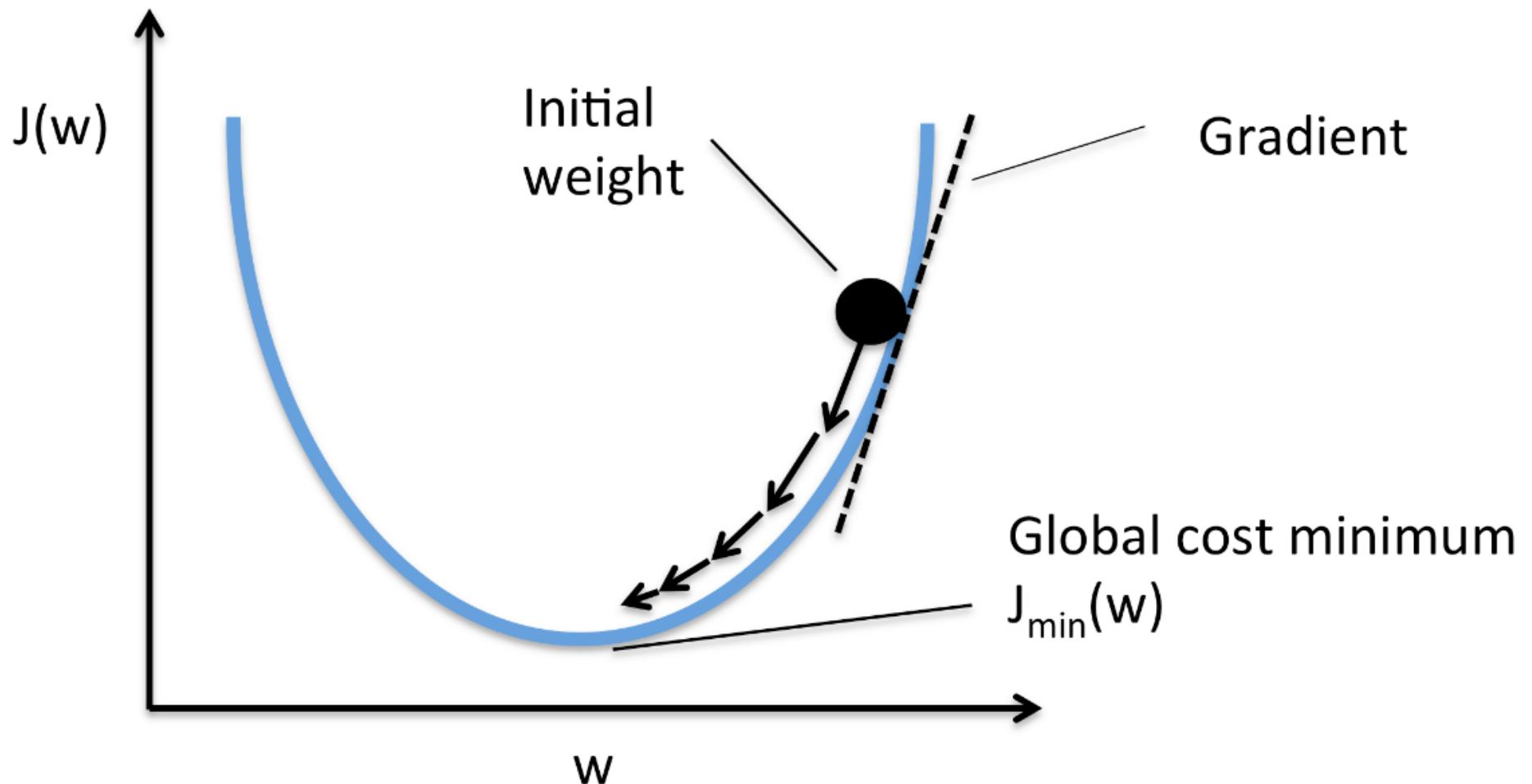
## Backpropagation

$$\Delta w_i = \alpha(t - y)x_i$$

The diagram illustrates the formula for updating a weight  $\Delta w_i$ . The formula is  $\Delta w_i = \alpha(t - y)x_i$ . Arrows point from the terms in the formula to their respective definitions: 'Input' points to  $x_i$ , 'Output' points to  $y$ , 'Target' points to  $t$ , 'Learning Rate' points to  $\alpha$ , and 'Updated Weight' points to  $\Delta w_i$ .

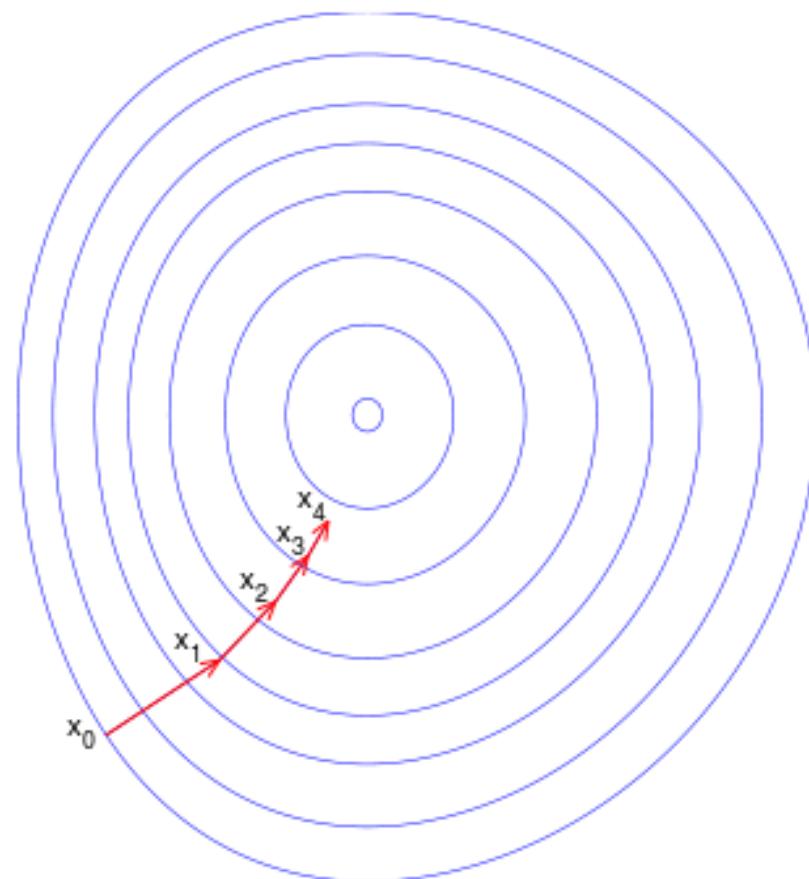
# Neural Network

Backpropagation



# Neural Network

## Backpropagation

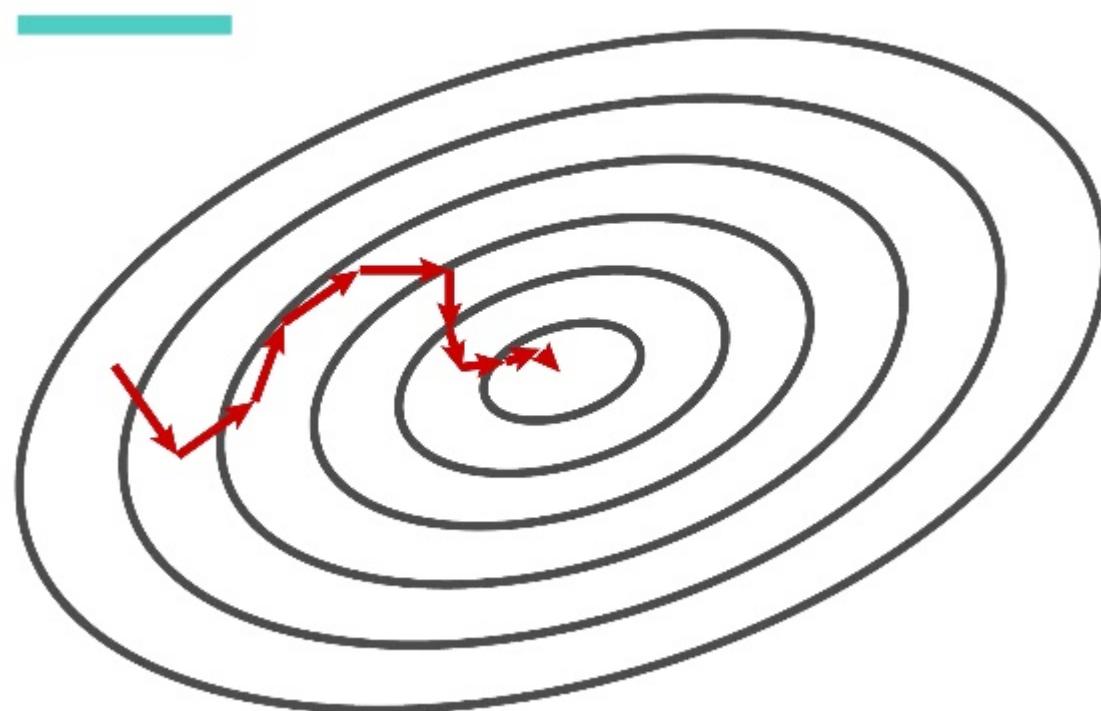


Gradient Descent

# Neural Network

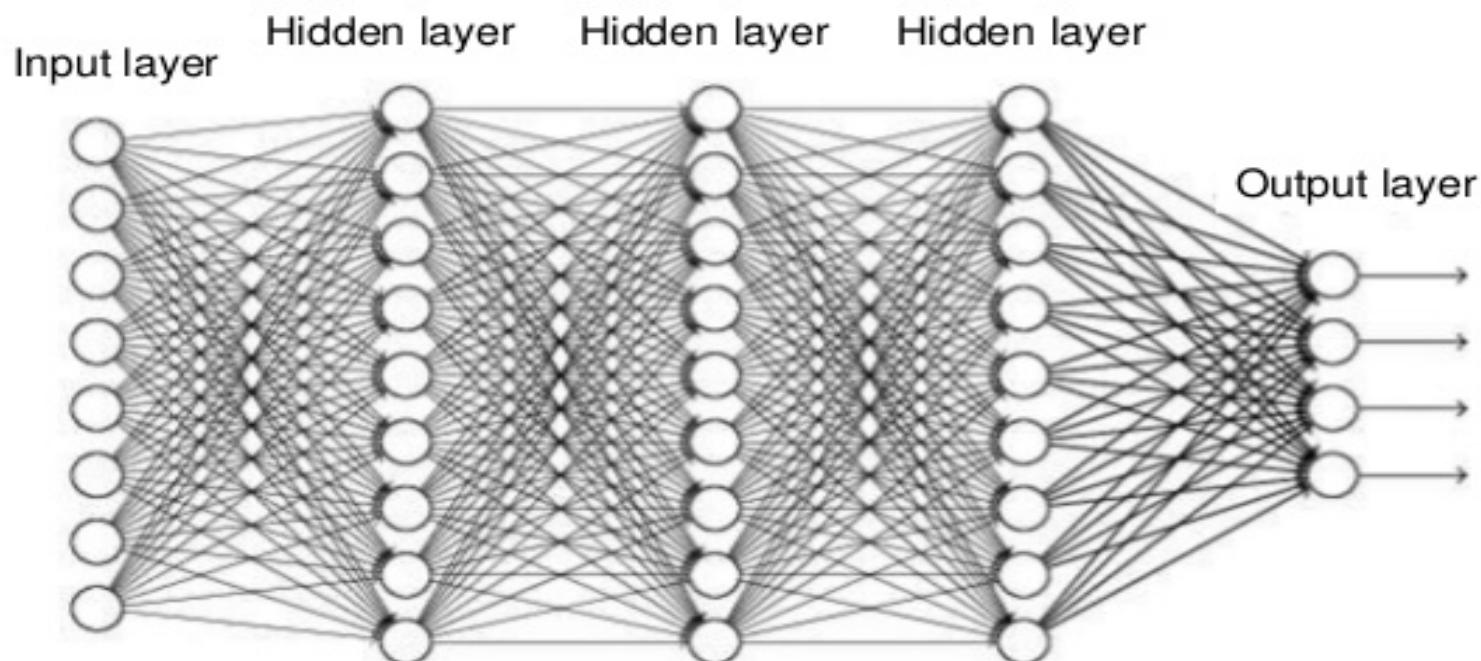
## Backpropagation

### Stochastic Gradient Descent



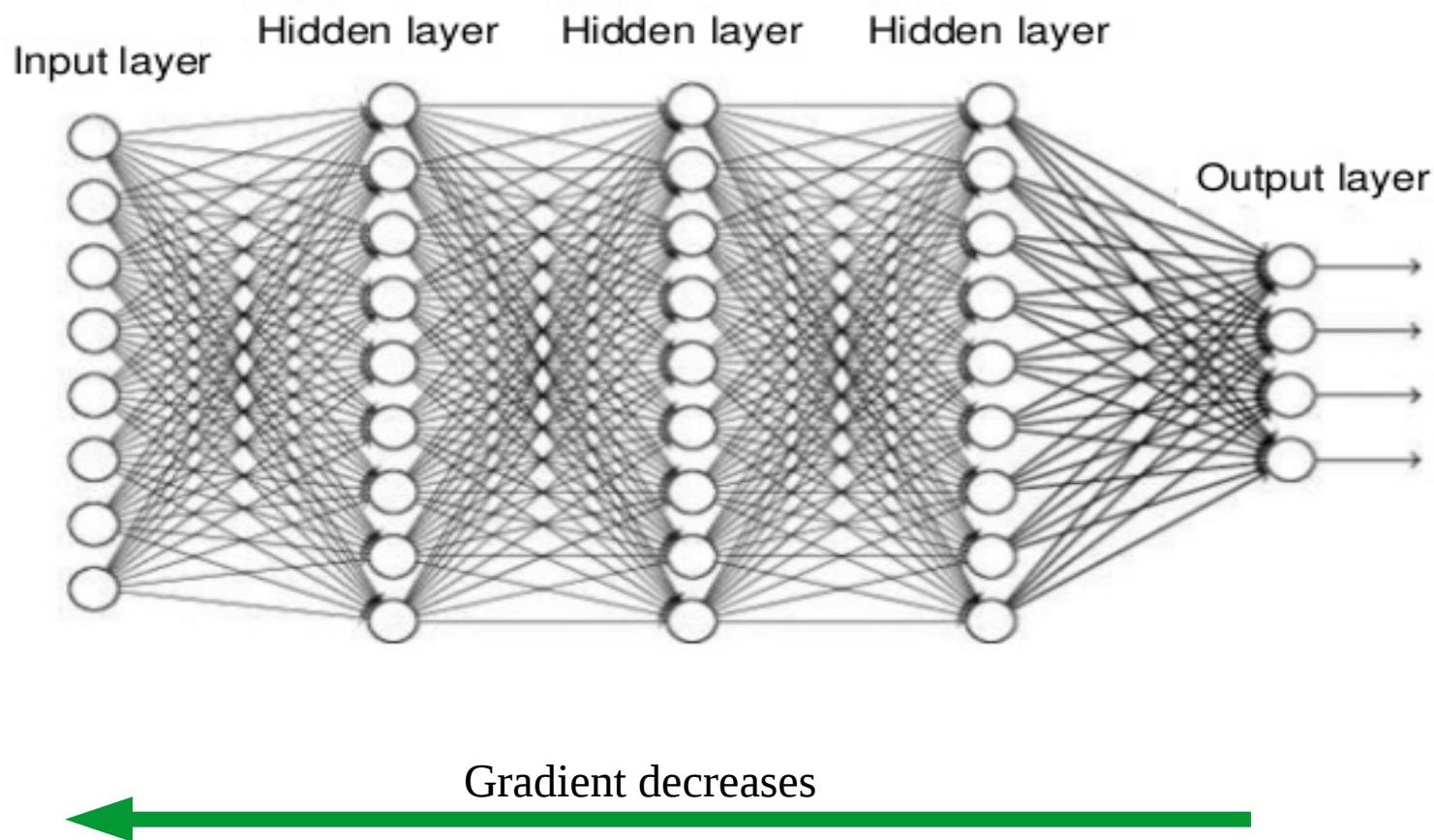
# Neural Network

## Backpropagation



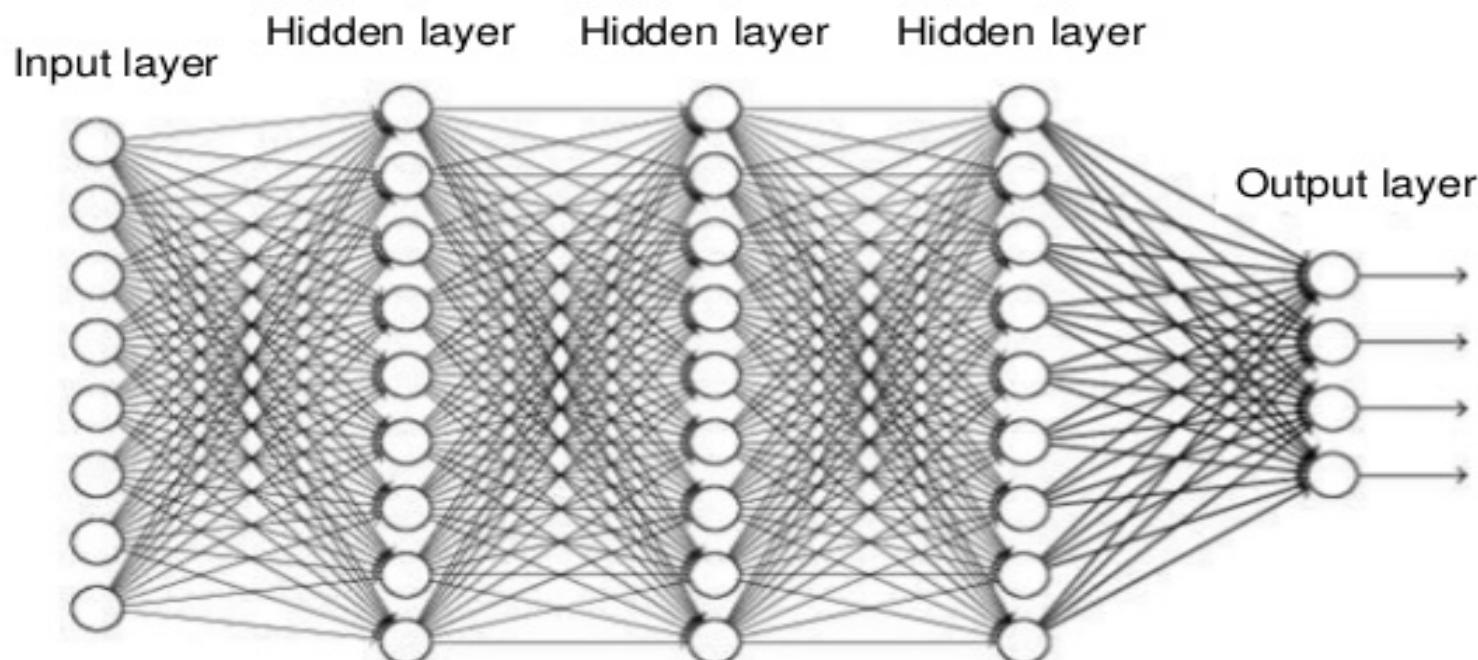
# Neural Network

## Backpropagation



# 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

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

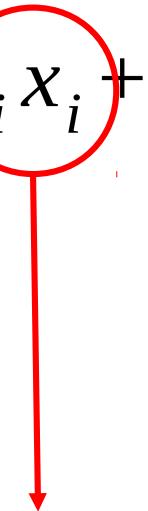


Dot product of two very large matrices

```
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

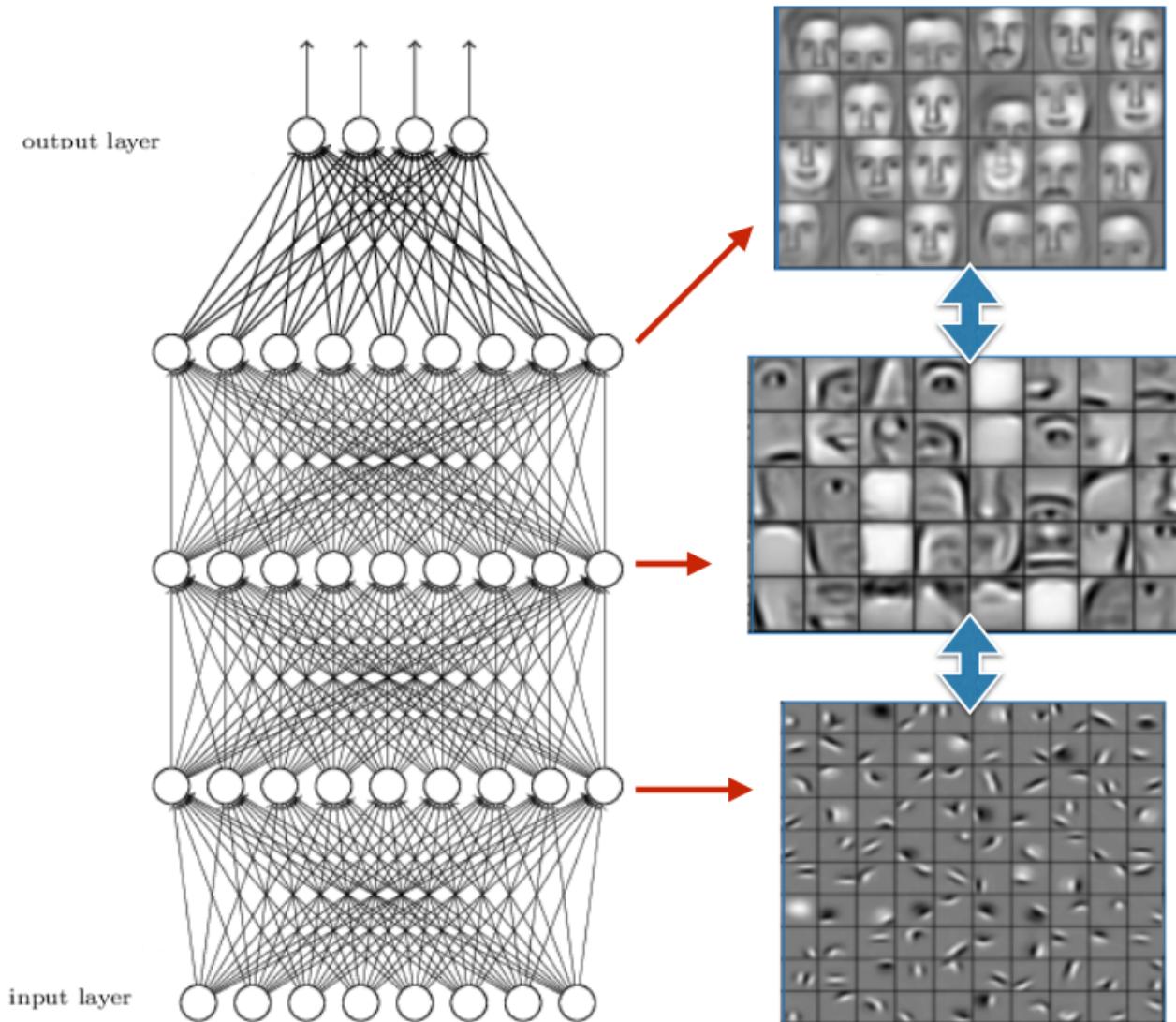
## Motivation

$$\begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} \quad \mathbf{I}$$
$$\begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array} \quad \mathbf{K}$$
$$\begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 3 & 4 & 1 \\ \hline 1 & 2 & 4 & 3 & 3 \\ \hline 1 & 2 & 3 & 4 & 1 \\ \hline 1 & 3 & 3 & 1 & 1 \\ \hline 3 & 3 & 1 & 1 & 0 \\ \hline \end{array} \quad \mathbf{I} * \mathbf{K}$$

The diagram illustrates the convolution operation between input tensor  $\mathbf{I}$  and kernel  $\mathbf{K}$ . The input  $\mathbf{I}$  is a 7x8 matrix of binary values. The kernel  $\mathbf{K}$  is a 3x3 matrix. The result of the convolution is shown in the output tensor  $\mathbf{I} * \mathbf{K}$ , which is a 5x5 matrix of integer values. The convolution step is highlighted with blue dotted lines and a multiplication symbol (\*). The resulting value in the output matrix is highlighted with a green box.



# Convolutional Neural Network



Hierarchy of Features

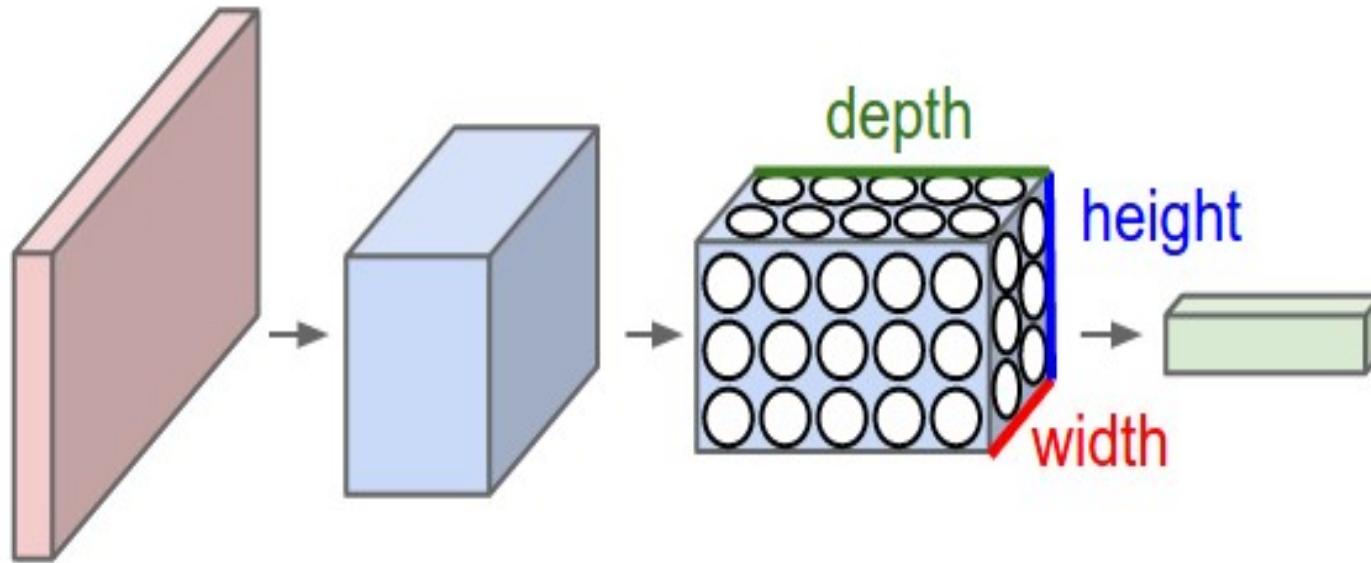
# Convolutional Neural Network



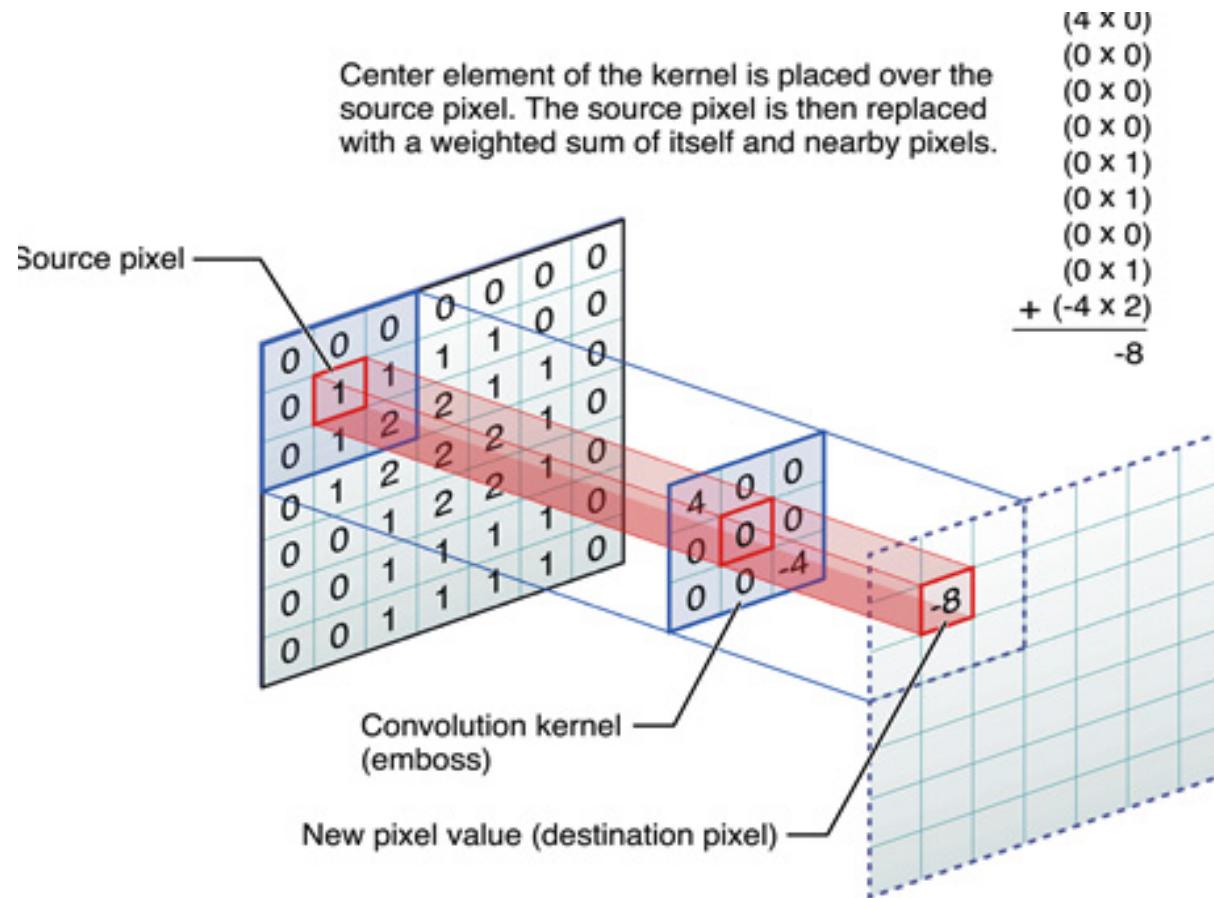
Yann LeCun - ConvNets (1996)



# Convolutional Neural Network



# Convolutional Neural Network



Canny Kernel

# Convolutional Neural Network

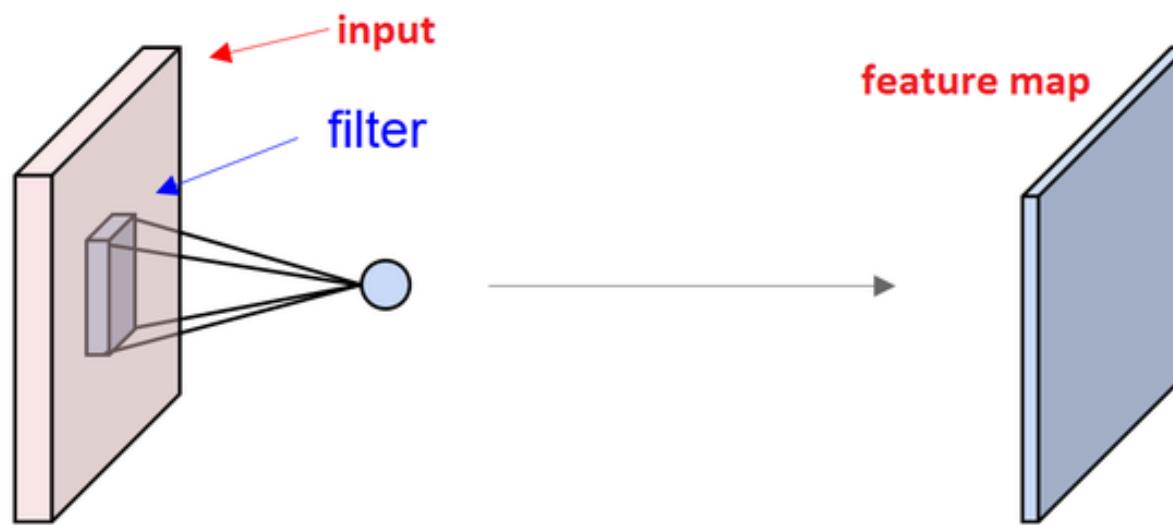
1 x1	1 x0	1 x1	0	0
0 x0	1 x1	1 x0	1	0
0 x1	0 x0	1 x1	1	1
0	0	1	1	0
0	1	1	0	0

Image

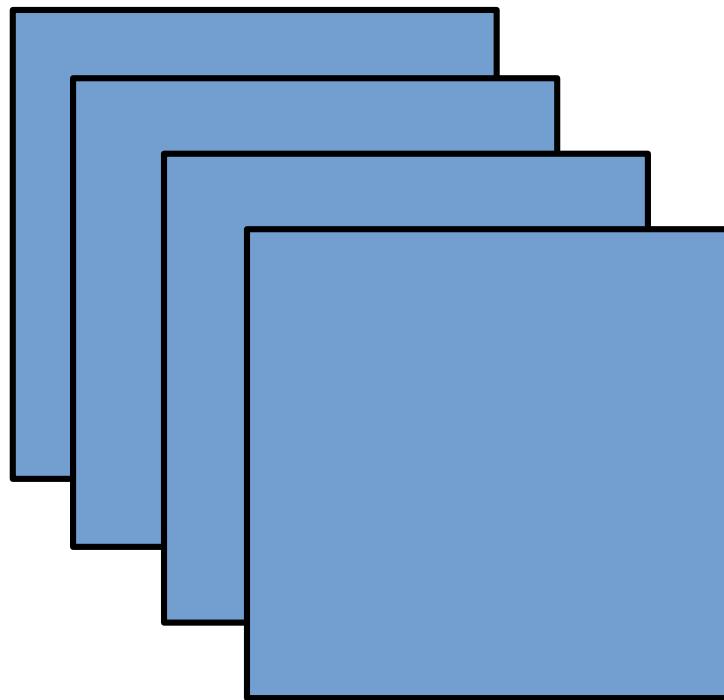
4		

Convolved  
Feature

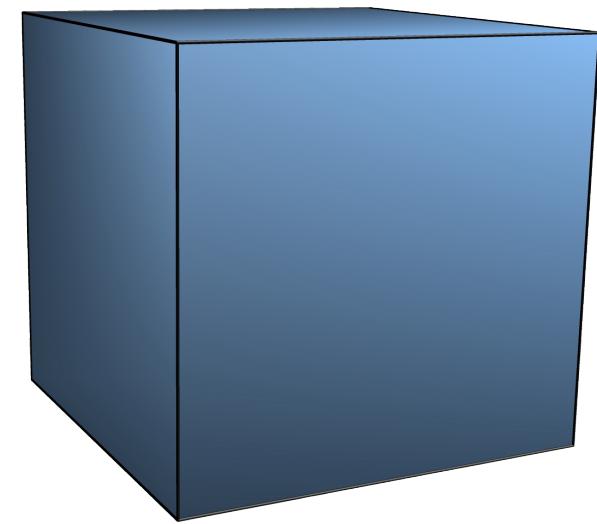
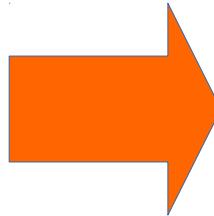
# Convolutional Neural Network



# Convolutional Neural Network



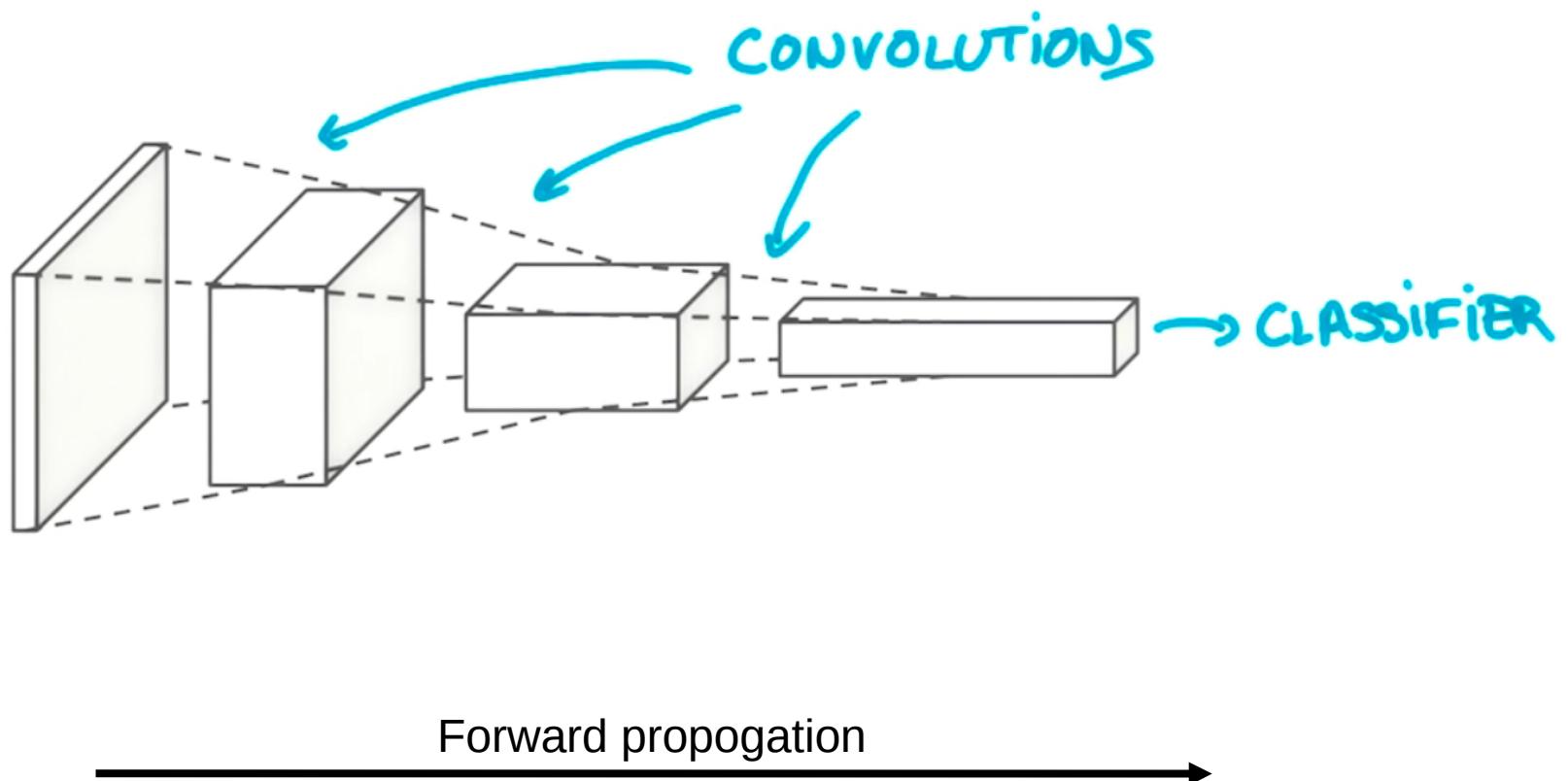
4 feature maps with size 32x32



Collectively represented as  $4 \times 32 \times 32$

# Convolutional Neural Network

CONVOLUTIONAL PYRAMID



# Convolutional Neural Network

CNNs have three main components :

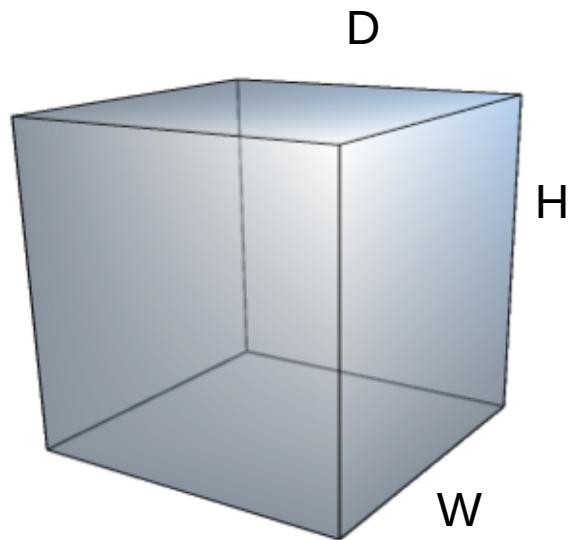
1) *Convolutional Layers*

2) *Pooling layer*

3) *Fully Connected layer*

## *Spatial Size of Output Volume*

$$W \times H \times D$$



## ***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

W = 5 (5x5)

F = 3 (3x3)

P = 0

S = 1

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

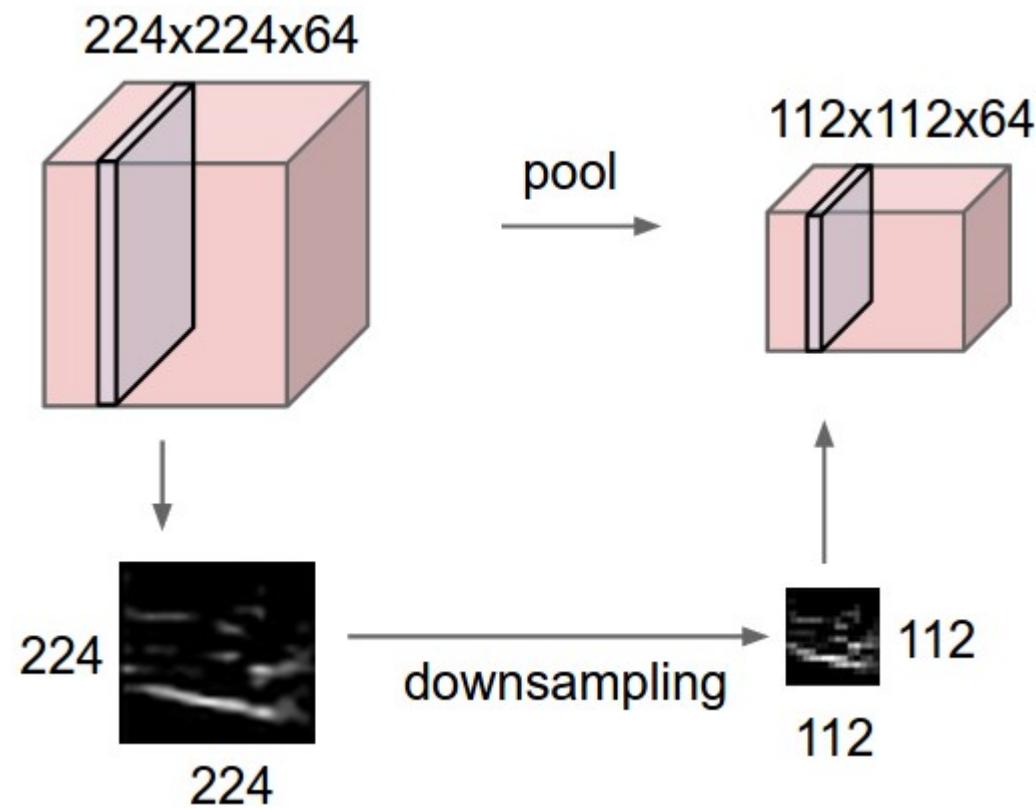
4		

Convolved  
Feature

**sz = 3**

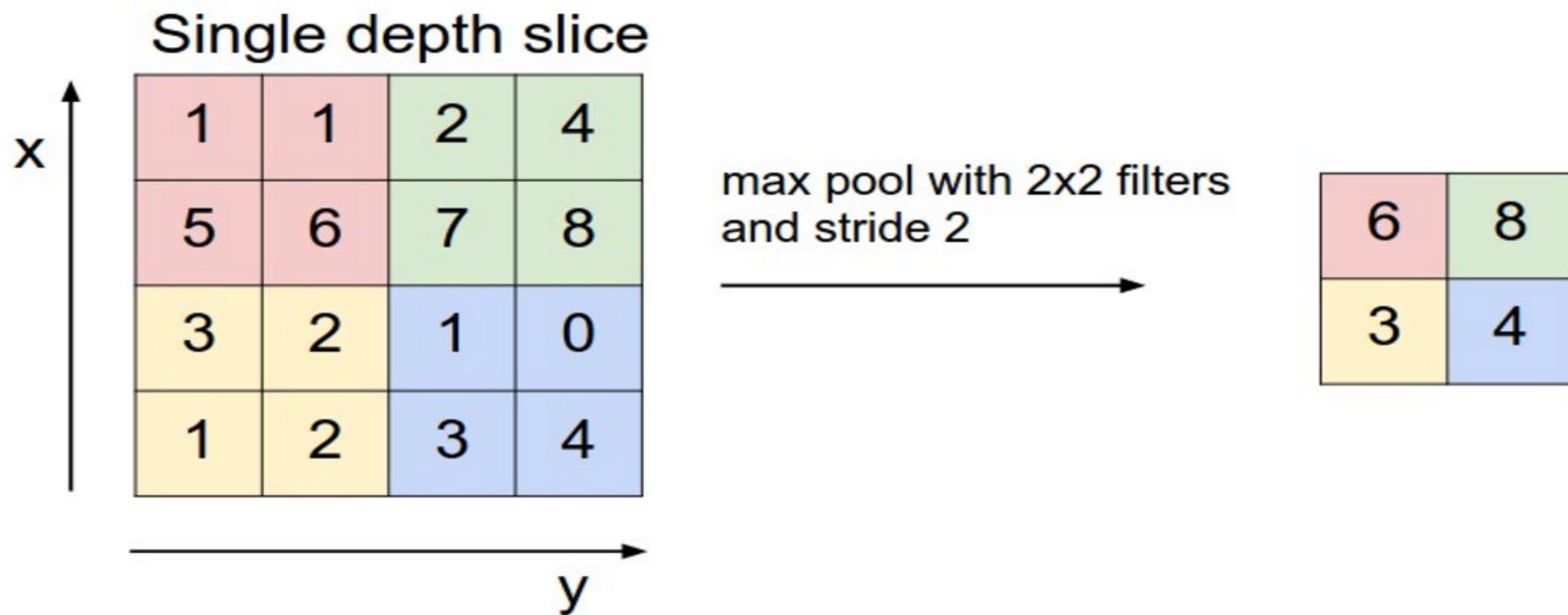
# ***Pooling***

Reduces *spatial size* of input by subsampling

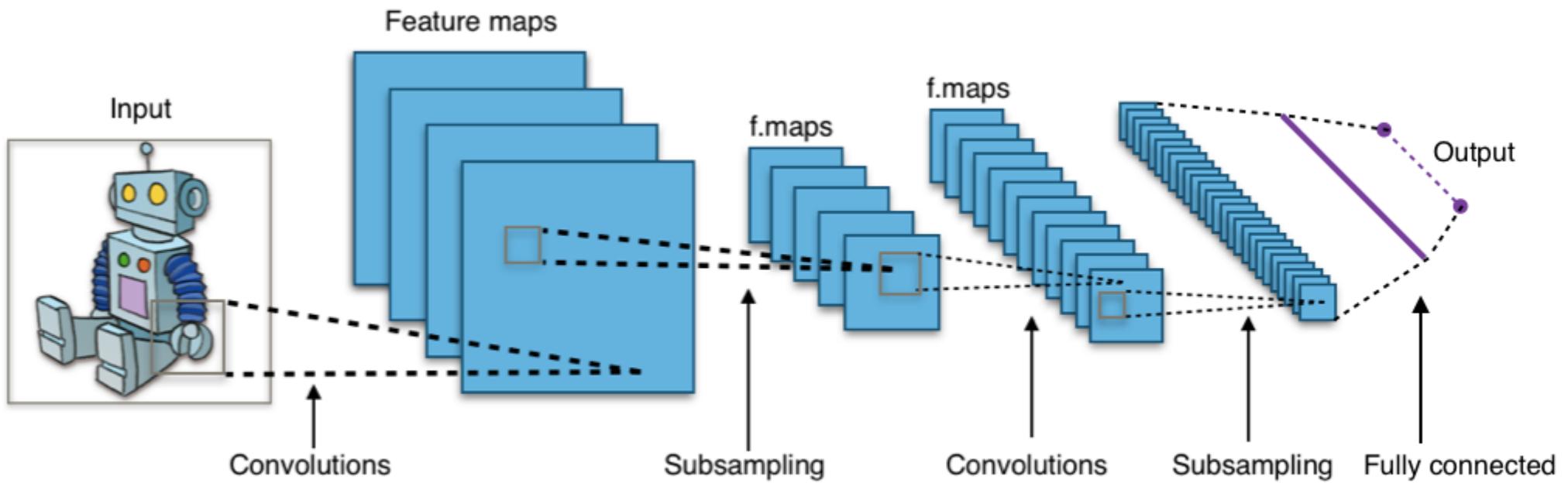


# ***Pooling***

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 : Output at one layer is 7x7x512

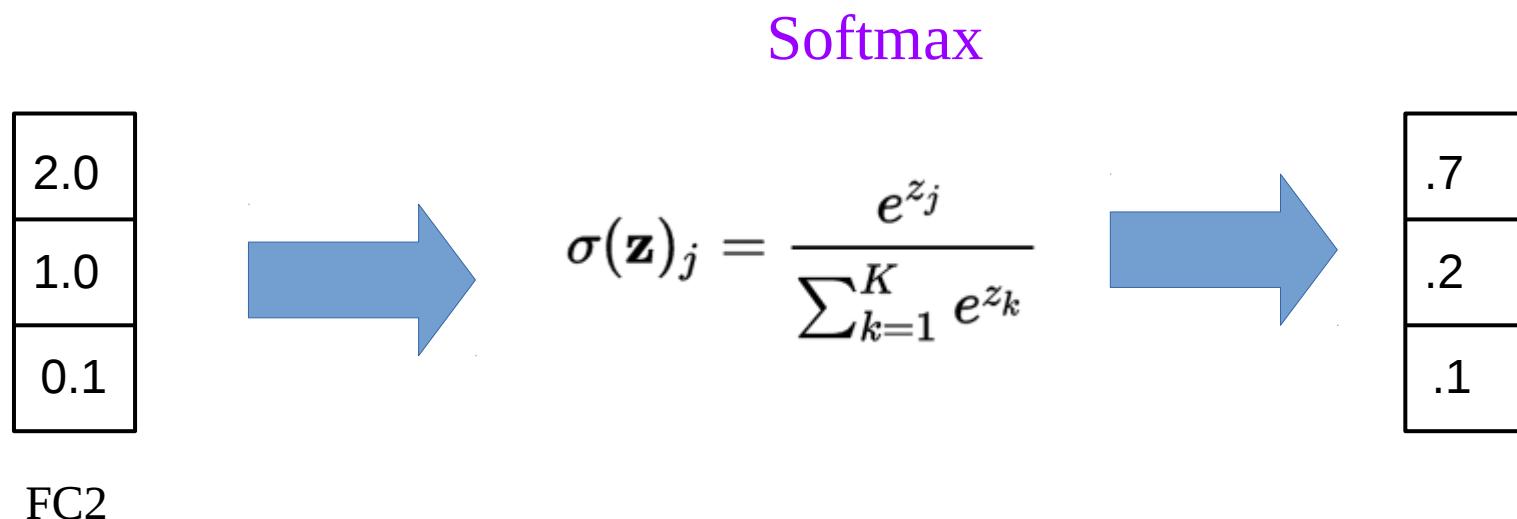
We need to input them into a NN of 1024 inputs

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

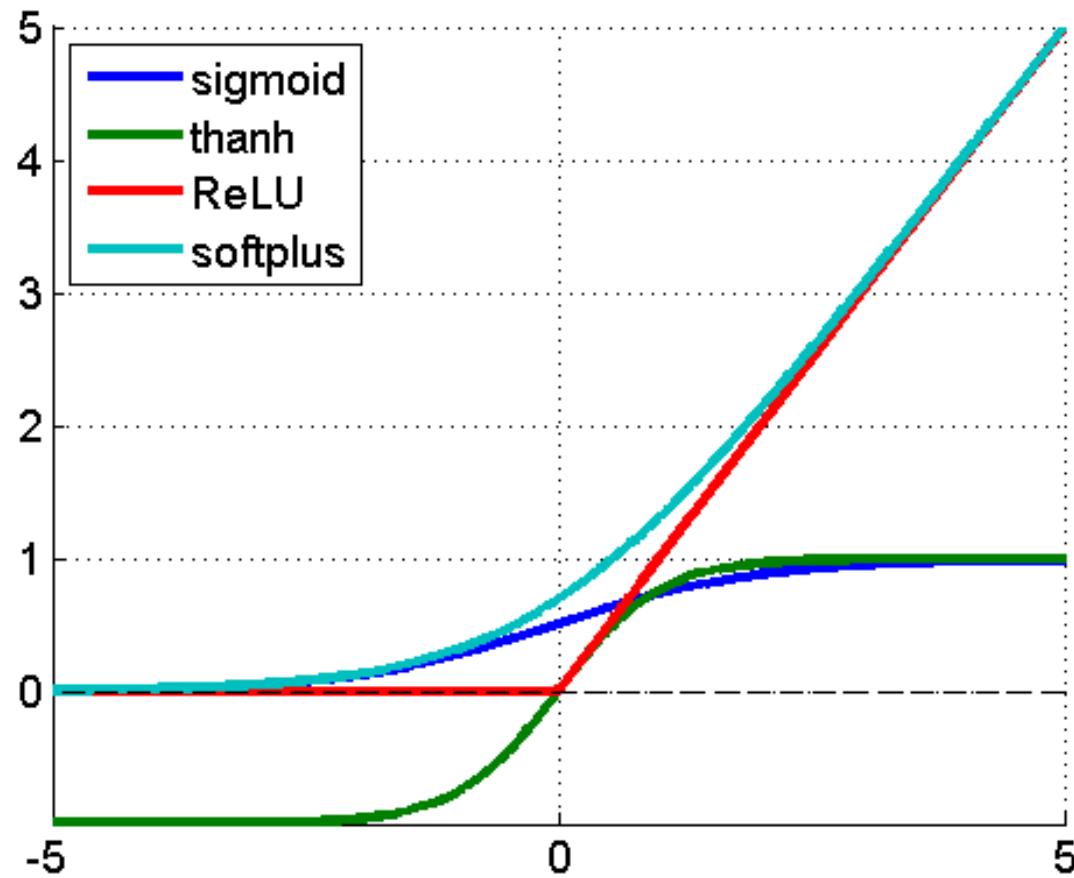
$$SZ = \frac{(W - F + 2P)}{S} + 1$$

**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

0 0 0 0 0 0 0 0 0 0  
1 1 1 1 1 1 1 1 1 1  
2 2 2 2 2 2 2 2 2 2  
3 3 3 3 3 3 3 3 3 3  
4 4 4 4 4 4 4 4 4 4  
5 5 5 5 5 5 5 5 5 5  
6 6 6 6 6 6 6 6 6 6  
7 7 7 7 7 7 7 7 7 7  
8 8 8 8 8 8 8 8 8 8  
9 9 9 9 9 9 9 9 9 9