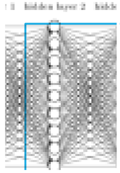
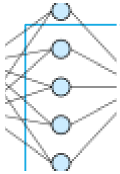
DEEP LEARNING

Dr.S.Lovelyn Rose
Associate Professor
Dept. of CSE
PSG CT, Cbe.

AGENDA

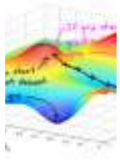


Deep Learning – ?

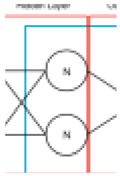


Artificial Neural Networks

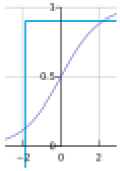
Gradient Descent



Stochastic Gradient Descent



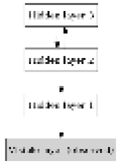
Backpropagation



Activation Functions

Concurrent Neural Network – Eg.

Restricted Boltzmann Machine



Deep Belief Network

Recurrent Neural Net

Recursive Neural Tensor Net

Software Libraries

WITH AND WITHOUT DEEP LEARNING

Features

- Crest
- Hoof
- Muscular legs
- Muzzle
- Lush tail

Do I explicitly state the features?

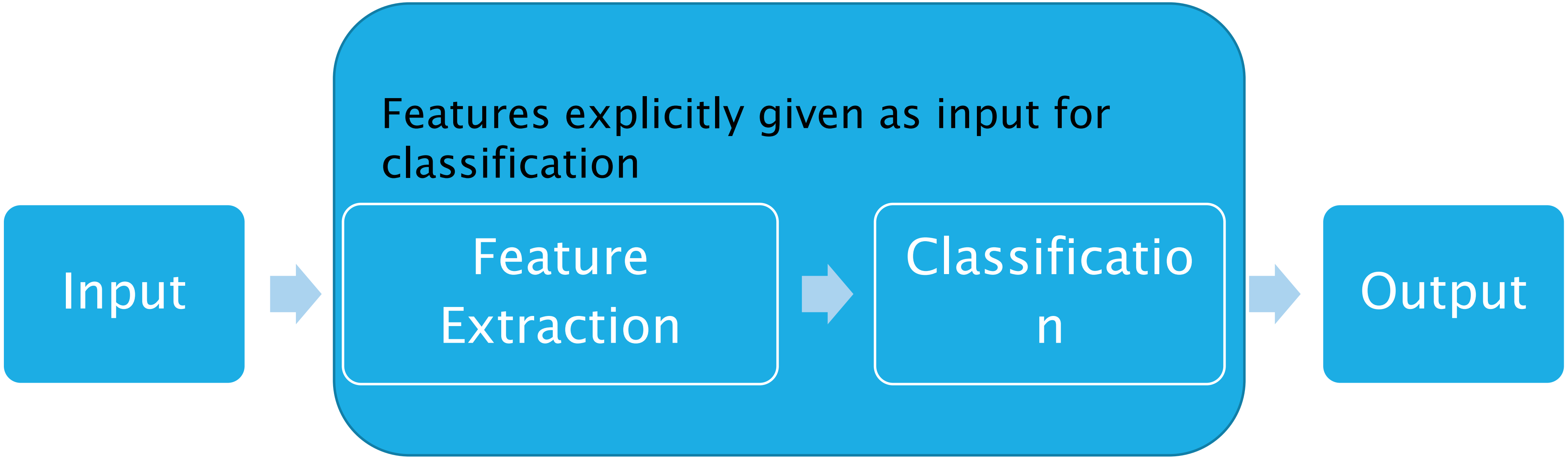
If your answer is YES

Traditional Learning

Do I explicitly state the features?

If your answer is NO

Deep Learning



Features explicitly given as input for classification

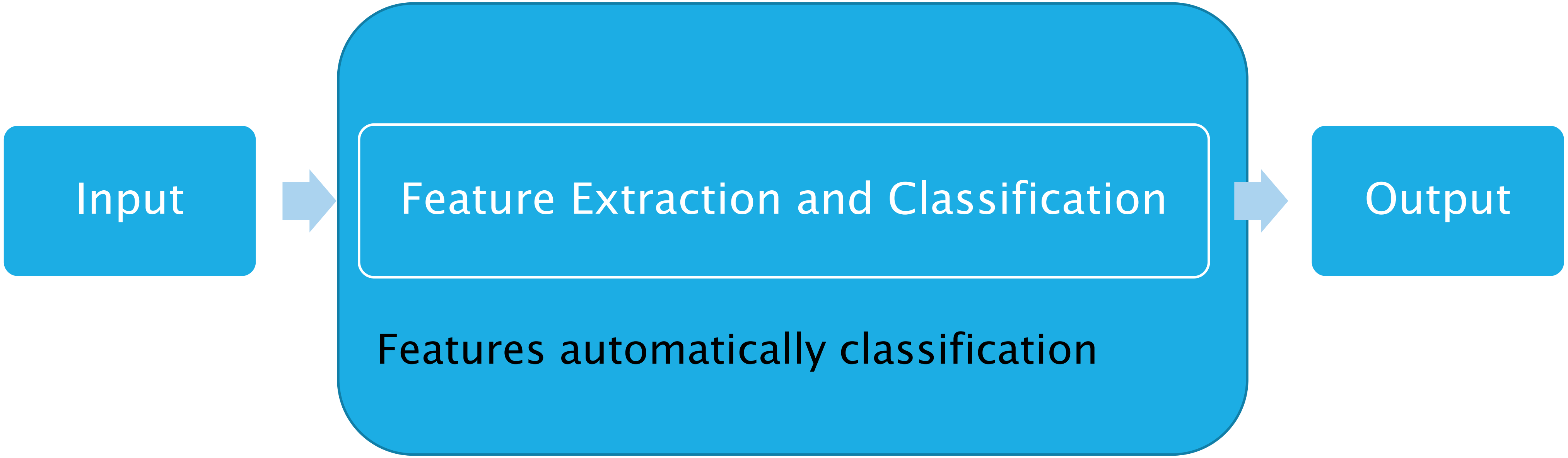
Input

Feature Extraction

Classificatio
n

Output

Traditional Learning vs Deep Learning



Input

Feature Extraction and Classification

Features automatically classification

Output

A FEASIBILITY STUDY

Processing Power

Very parallel

Fast

Cheap

Performance benchmark

PC CPU – 1 –3 Gflops/sec average

GPU – 100 Gflops/sec average

Training Data

Deep learning requires lot of

- Processing power
- Training data

ARTIFICIAL NEURAL NETWORKS

MCCULLOCH AND PITTS NETWORK

Linear threshold gate

Input : x_1, x_2, … x_n

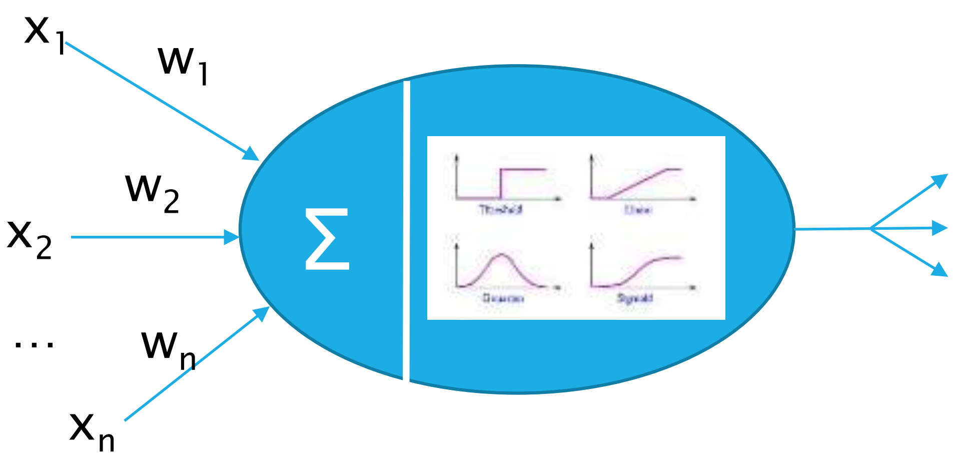
Output : Y – binary

Weights : w_1, w_2, … w_n

Weights Normalized : (0,1), (–0.5,0.5), (–1,1)

Calculate Net Input : $I = \sum_{k=1}^{n} x_k \ * wk$

Activation Function = $f(I) = \begin{cases} 1, if\ I\ \geq T \\ 0, if\ I < T \end{cases}$ – binary step function

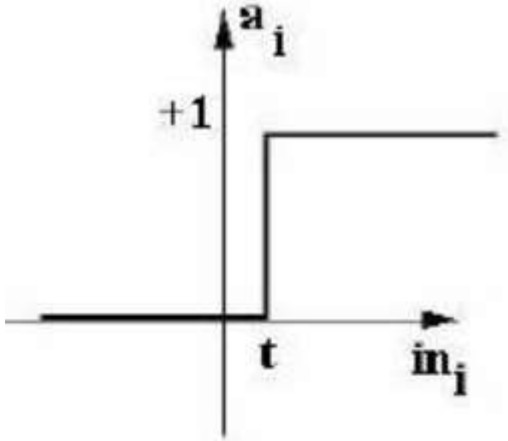


ACTIVATION FUNCTIONS

Heaviside Step function

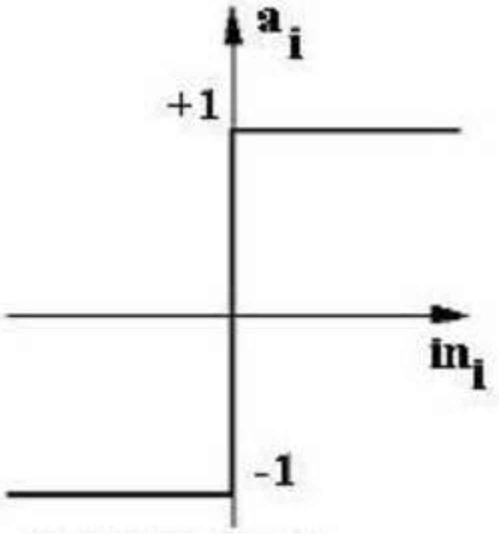
Output either 0 or 1

$$f(I) = \begin{cases} 0, & I < 0 \\ 1, & I \geq 0 \end{cases}$$



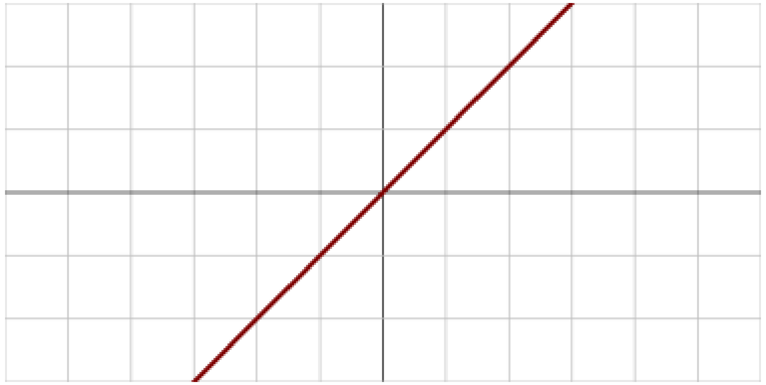
Sign function

- Output either −1 or 1
- $f(I) = \begin{cases} -1, & I < 0 \\ 1, & I \geq 0 \end{cases}$



Linear function

- Input is the output
- $f(I) = I$



SIGMOID FUNCTION

Sigmoid Function

- Mathematical function with S shaped curve(sigmoid

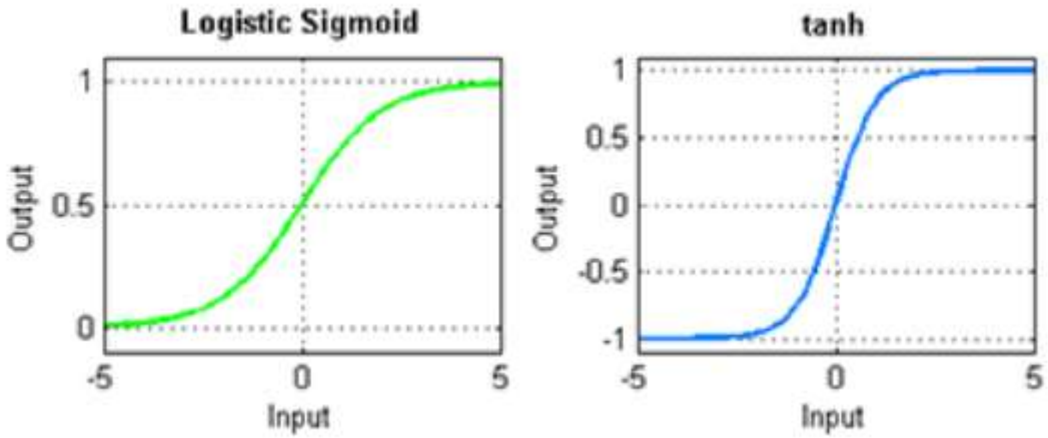
Types of Sigmoid functions

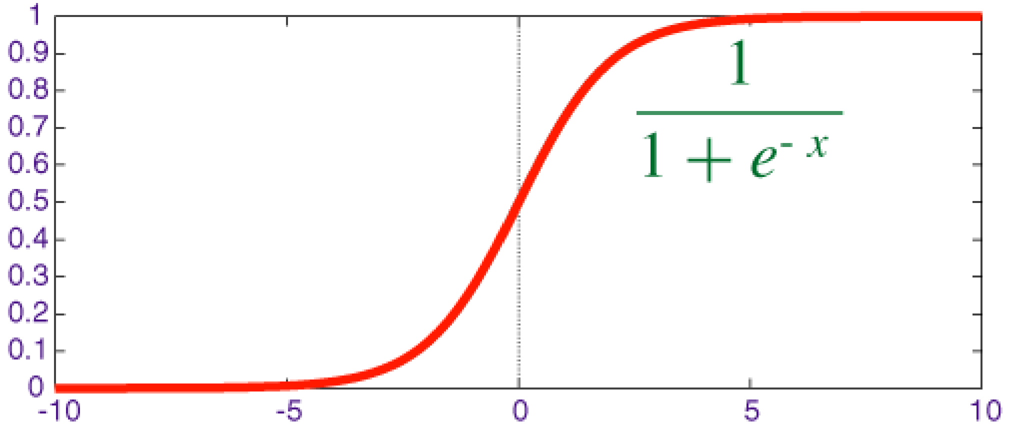
Logistic function

- Output between 0 and 1
- f(I) $= \frac{1}{1+e^{-I}}$

Hyperbolic tangent

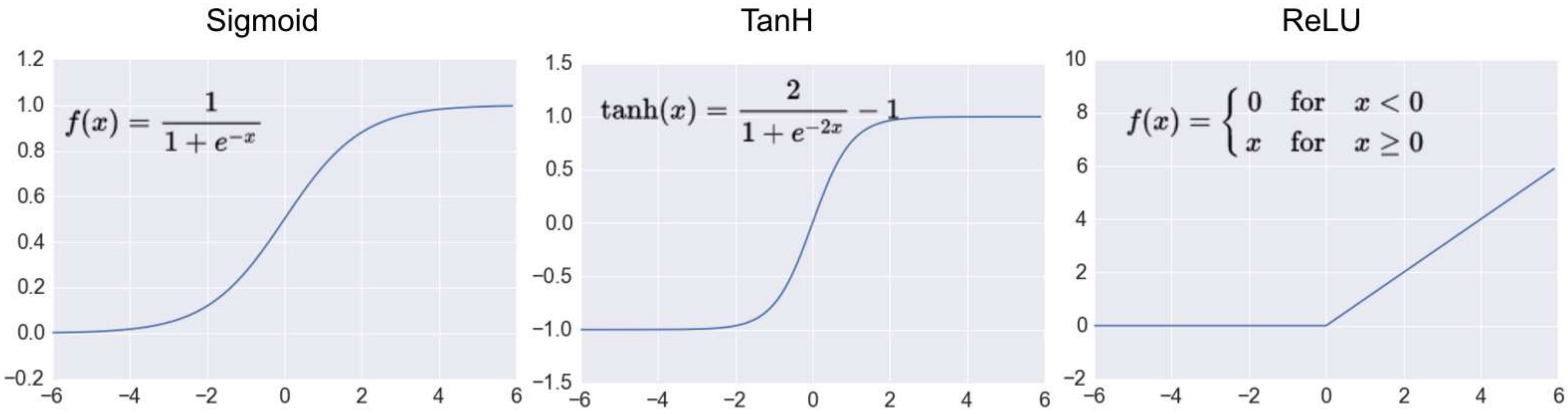
- Output between −1 and 1
- f(I) $= \frac{e^{2I}-1}{e^{2I}+1}$



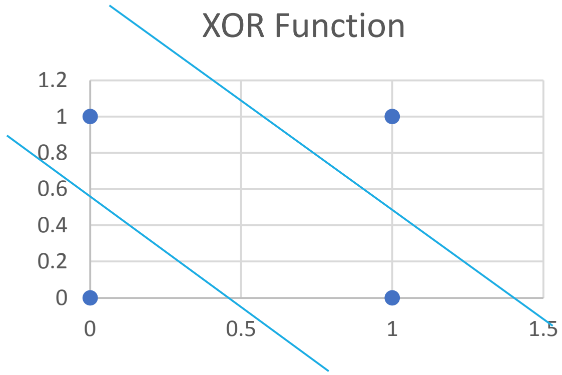


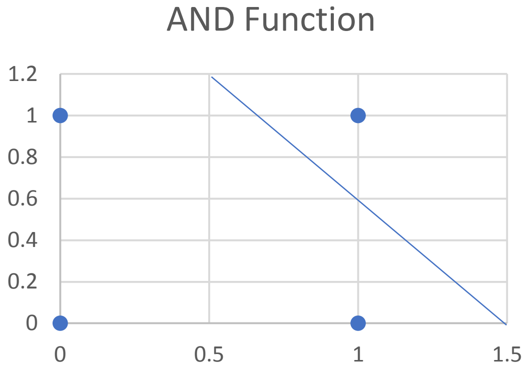
RECTIFIED LINEAR UNIT (RELU)

Commonly used in deep learning
Empirically good result



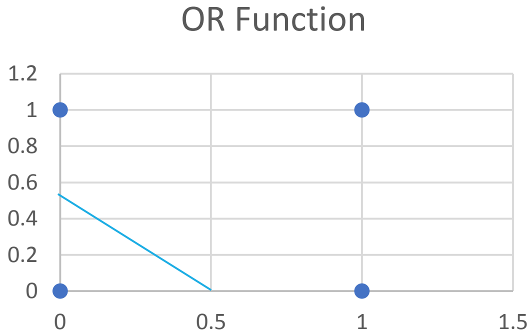
LINEARLY SEPARABLE





X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	1



X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

Right of line : Output should fire

EXAMPLE – AND FUNCTION

Let $w_i = 1 \; \forall \, i$

Net input (1,1) : $1*1 + 1*1 = 2$

Net input (1,0) : $1*1 + 0*1 = 1$

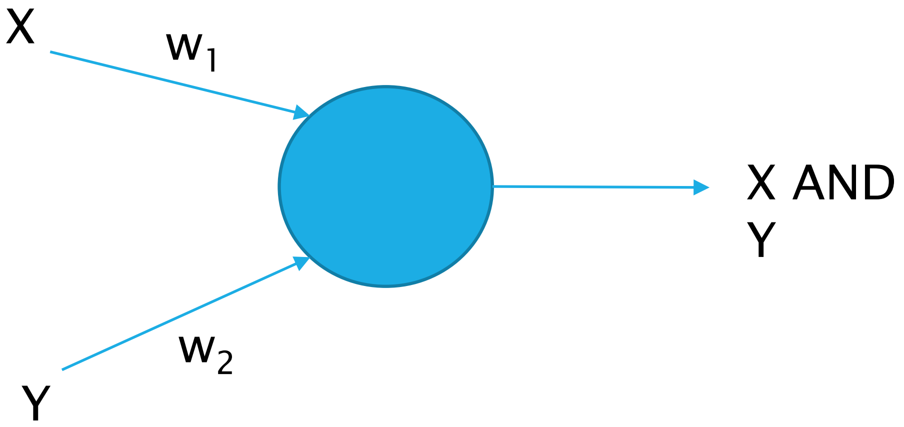
Net input (0,1) : $0*1 + 1*1 = 1$

Net input (0,0) : $0*1 + 0*1 = 0$

Set $T = 2$

- Output : 1, if T ≥ 2
- Output : 0, if T < 2

Remember we designed the network



QUESTION TIME

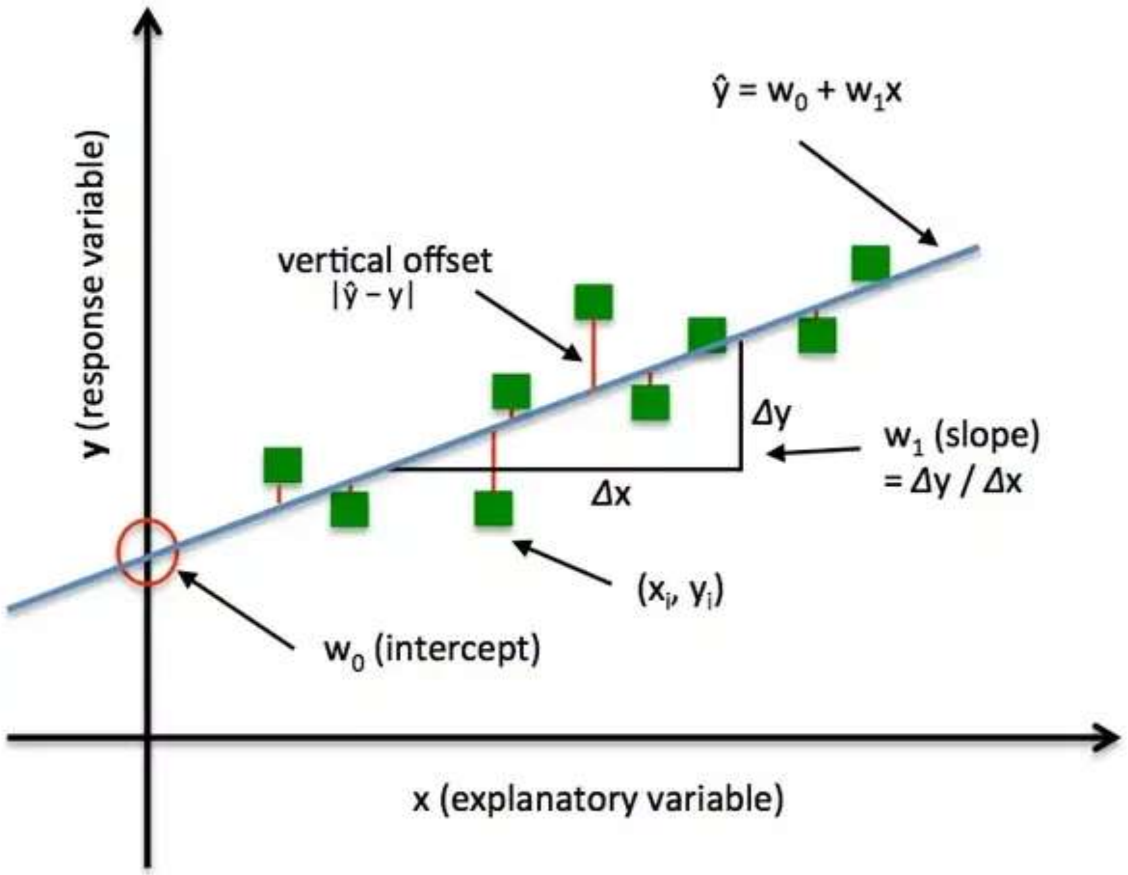
What changes do you need to make to implement an OR function

LINE – SOME MATHEMATICS

$$y = w_0.x_0 + w_1.x_1 + \ldots + w_n.x_n$$

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

$$= w^T x$$



INTRODUCING BIAS

Classifier tries to find line to separate classes

$$y=mx+c$$

In training phase neural network tries to learn appropriate weights to draw the line

If there is no y intercept, any line we draw will pass through origin

So introduce y-intercept as bias

If we set the bias to 1 then why does it make difference to the fit now that every line will now go through (0,1) instead of (0,0)

by multiplying a bias by a weight, you can shift it by an arbitrary amount

BIAS AND THE SIGMOID FUNCTION

$(x_1,x_2) = (0.2, 0.5)$

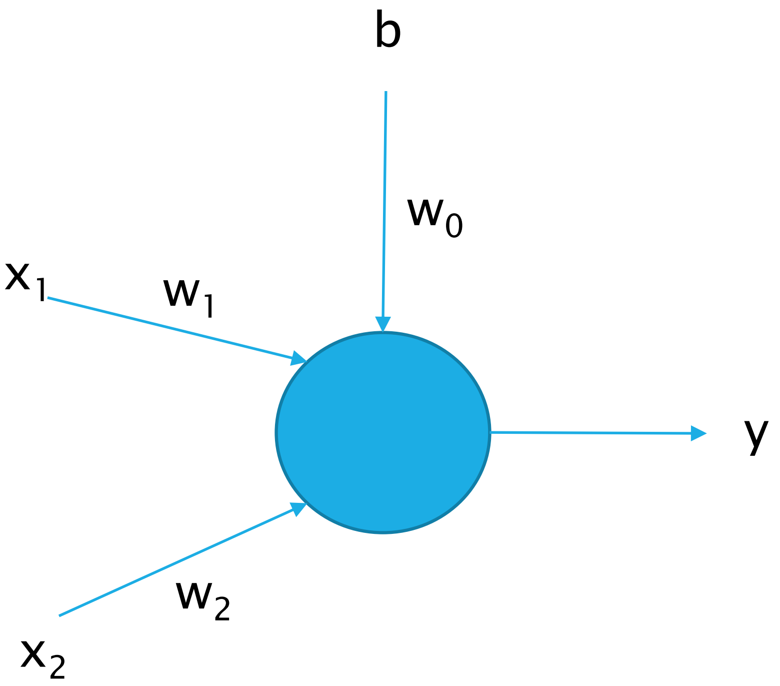
$(w_0,w_1,w_2) = (0.4, 0.3, 0.5)$

$b = 1$

Net input : I = 1*0.4 + 0.2*0.3 + 0.5*0.5

= 0.71

$$y = \frac{1}{1+e^{-I}} = \frac{1}{1+e^{-0.71}} = 0.67$$



PERCEPTRON

Binary classifier

Let input x = (x_1, x_2, ... , x_n) where each x_i = 0 or 1.
And let output y = 0 or 1.

Algorithm is repeat forever:

- Given input x = (x_1, x_2, ... , x_n).
- Perceptron produces output y.
- The correct output is **O**.
- If we had wrong answer, change w_i's and T, otherwise do nothing.

Single layer perceptron solves only linearly separable problems

HOW TO CHANGE w_i

y – Predicted output – 0.67

O – Actual output – 1

Change w_i based on (O–y)

Use error function to connect actual and predicted output

ERROR FUNCTION/LOSS FUNCTION/COST FUNCTION

Given data set with ‘n’ input

Error = Mean Square Error

C $= \frac{1}{n}\sum_{i=1}^{n}(oi - yi)^2$

C(w,b) $= \frac{1}{n}\sum_{i=1}^{n}(oi - (w^T x + b))^2$

$= \frac{1}{n}\sum_{x}(o - y)$

w – collection of weights in the network

b – collection of bias

o – vector of actual(desired) output

y – vector of predicted output when x is input

– dependent on w, b

Why error function?

Why not do it directly?

– error functions help to figure how small changes in w and b change the number of samples correctly classified

OBJECTIVE

- Minimize error function
- Minimize $C(w,b)$
- Perfect if the $C(w,b) = 0$
- Good if $C(w,b) \approx 0$
- i.e Find weights and bias such that $C(w,b) \approx 0$
- To do this use “Gradient Descent”

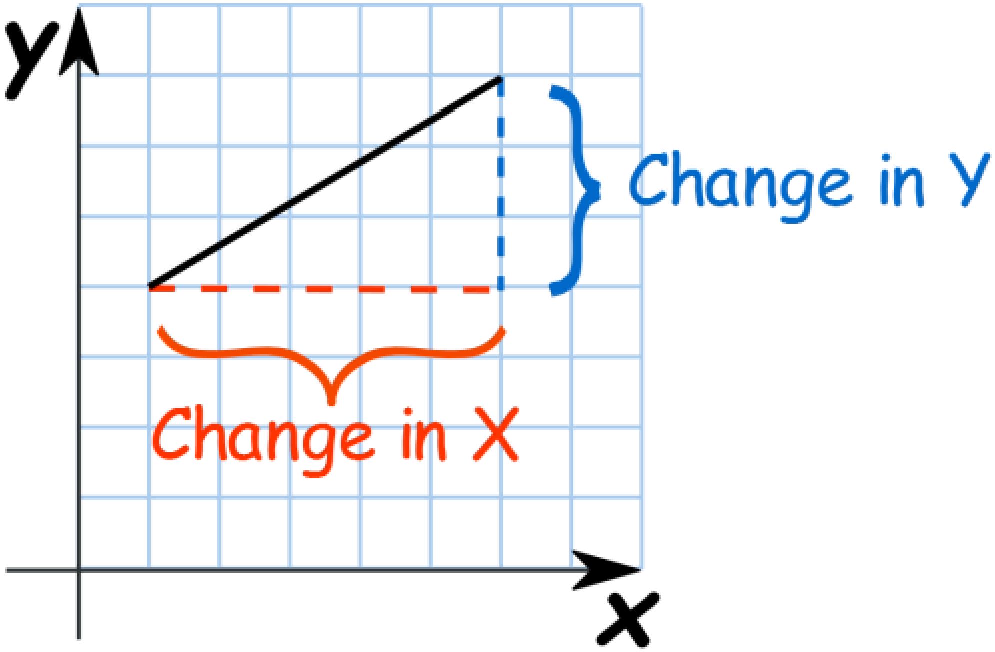
GRADIENT DESCENT

Or slope

How steep is the line

What is the change in y when x changes

Why not just call it slope?

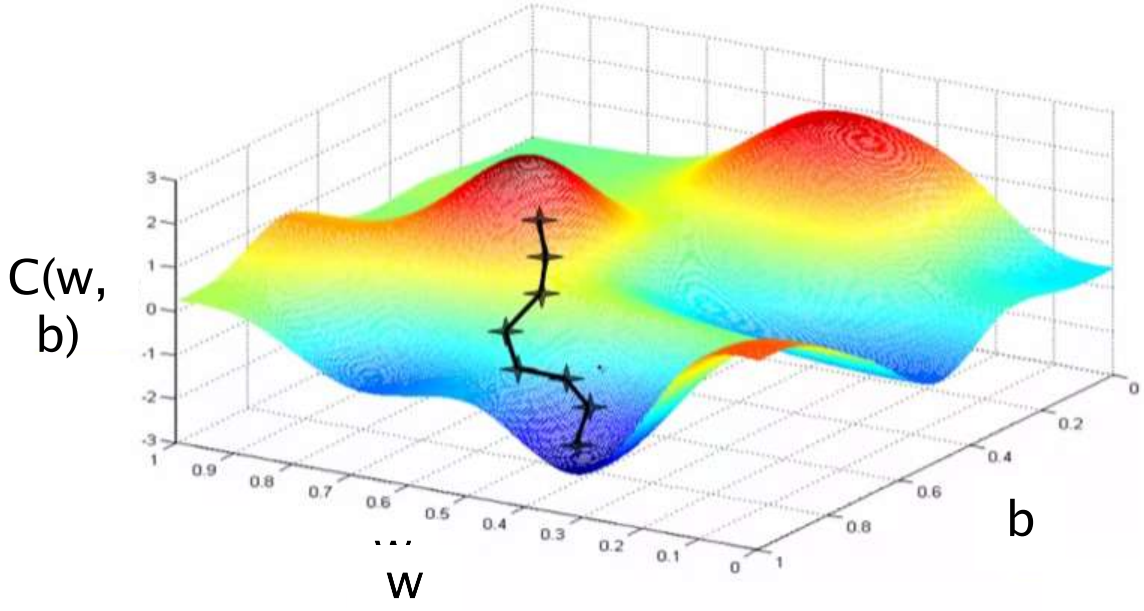


GRADIENT

GRADIENT

– involves more than 2 parameters

– here w, b and C(w,b)



Gradient – Vector calculus

- Gives direction in which a given function increases the maximum

BACKGROUND

If F is a function with variable x

Rate of change of F w.r.t x is $\frac{dF}{dx}$

i.e how much to move in the x direction

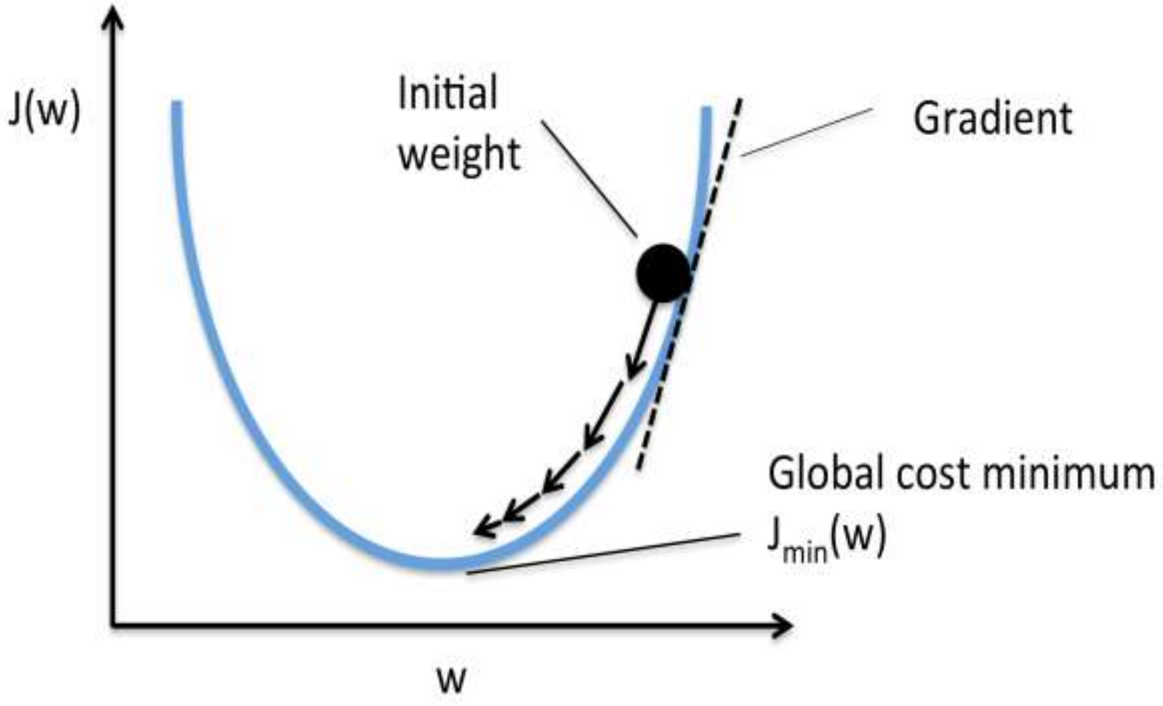
i.e to the right

If C has variables w, b

Direction to move give by ($\frac{\partial C}{\partial w}, \frac{\partial C}{\partial b}$)

These are partial derivatives (i.e keep other variables constant)

This gives direction of movement in the direction of w and b



GRADIENT DESCENT

We need to move in the direction where C will reach the minimum(like ball rolling down a hill)

The gradient given by the partial derivatives show the direction of maximum increase of the function C

Negative of gradient gives direction in which the function decreases the most

So keep moving in that direction till the function decreases no more

i.e. minima

But it can get stuck in local minima since the direction and movement depends on initial position

HOW MUCH TO MOVE

New_W = old_W + $\frac{\partial C}{\partial w}$

New_b = old_b + $\frac{\partial C}{\partial b}$

Or

New_W = old_W + $l \frac{\partial C}{\partial w}$

‘l’ is the learning rate that decides how big a step to take towards the minima

STOCHASTIC GRADIENT DESCENT

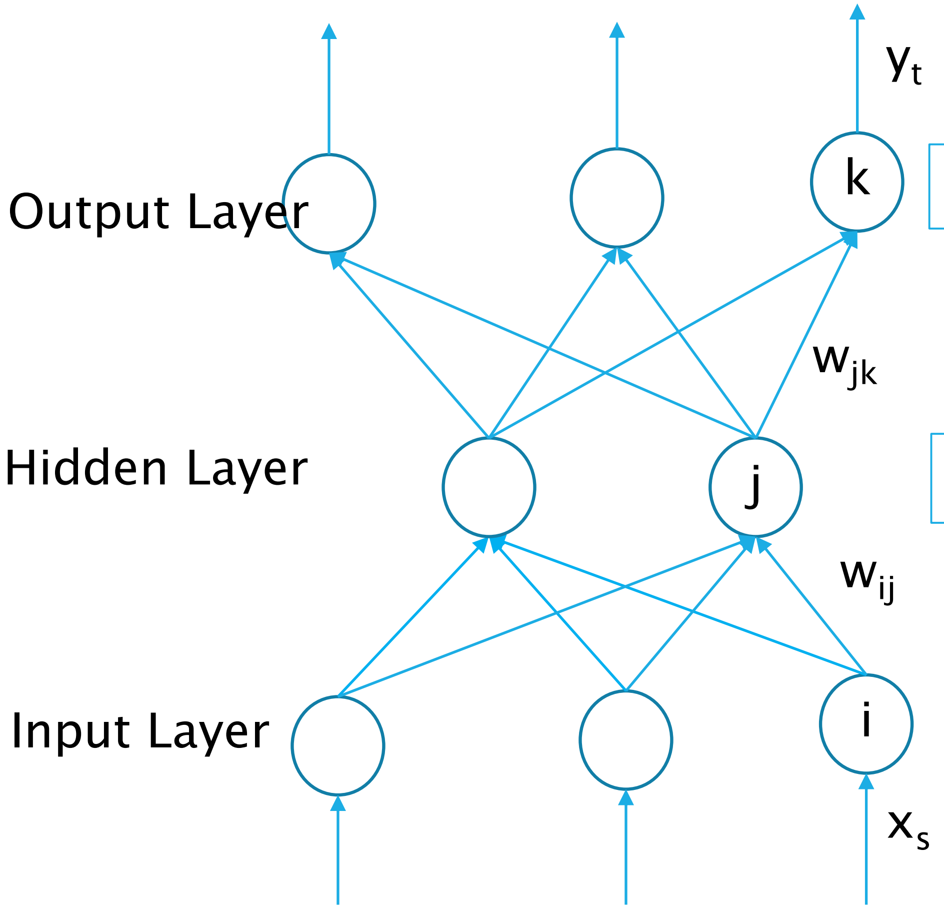
Instead of taking the entire training set before performing a move along minima, SGD takes a random set of samples before making a move.

Variations

1. Take 1 training sample at a time

2. Batch SGD – take a random set of subsamples

BACKPROPAGATION



$Err_k = o_k(1-o_k)(y_k-o_k)$

$Err_j = h_j(1-h_j)\sum_k Err_k w_{jk}$

i – node in input layer
j – node in hidden layer
k – node in output layer
w_{ij} – weight between nodes ‘i’ and ‘j’
w_{jk} – weight between nodes ‘j’ and ‘k’
X – input vector
x_s – input value of input unit ‘s’
y_k – predicted output of output unit ‘k’
o_k – actual output of output unit ‘k’
h_j – output of hidden unit ‘j’

Weight Updation

$$w_{ij} = w_{ij} + (l)Err_j o_i$$

– Do the same for bias also

TYPES OF NEURAL NETWORKS

Feed forward network

- Information goes only in one direction

Single layer perceptron

- information goes from input → output
- Only 1 output layer
- Nodes in output layer are processing elements – with activation function

Multilayer perceptron(MLP)

- Multiple layers of nodes in a directed graph
- Can distinguish non-linearly separable data
- Nodes in hidden and output layers – processing elements – with nonlinear activation functions
- Input → hidden → output

COMMON DEEP LEARNING NETWORKS WITH APPLICATIONS

Extract patterns from unlabeled data

- Restricted Boltzmann Machine
- Autoencoder

Classification from labelled data

- Recurrent neural net
- Recursive neural tensor network

Image recognition

- Deep belief network
- Convolutional neural network

Object recognition

- Convolutional neural network
- Recursive neural tensor network

Speech Recognition/Time series analysis

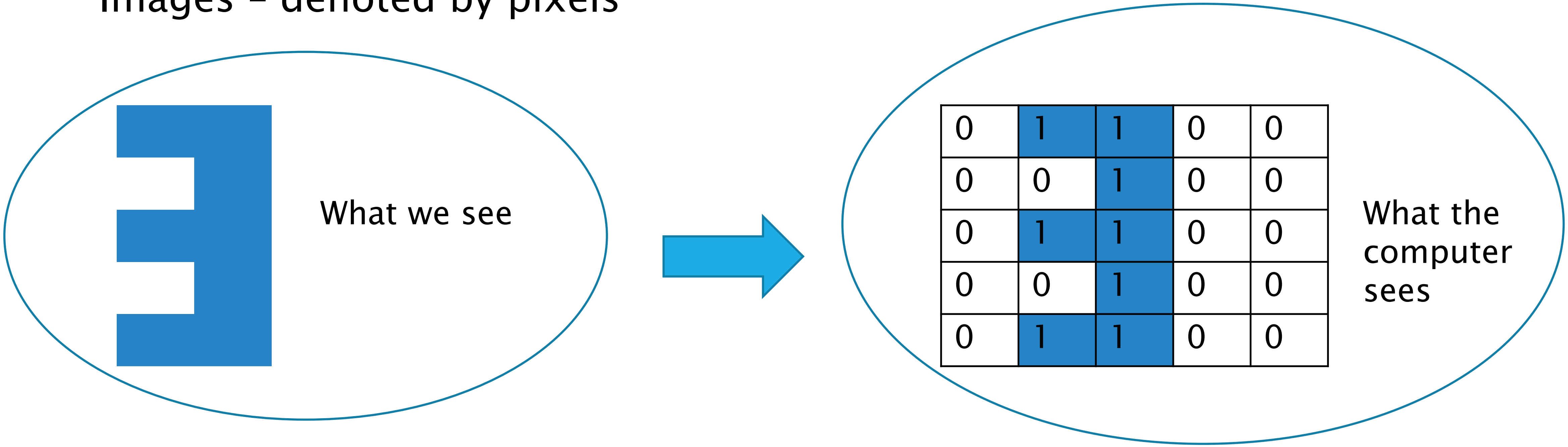
- Recurrent neural net

CONVOLUTIONAL NEURAL NETWORK

IMAGE CLASSIFICATION – A SIMPLE EXAMPLE

Initial use case – image classification

Images – denoted by pixels



What we see

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

What the computer sees

DETECT

Apply filter/kernel/feature detector

0	1	1
0	0	1
0	1	1

Receptive Field : size of filter in input image

Perform element-wise multiplication of filter and receptive field(blue) and add result

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

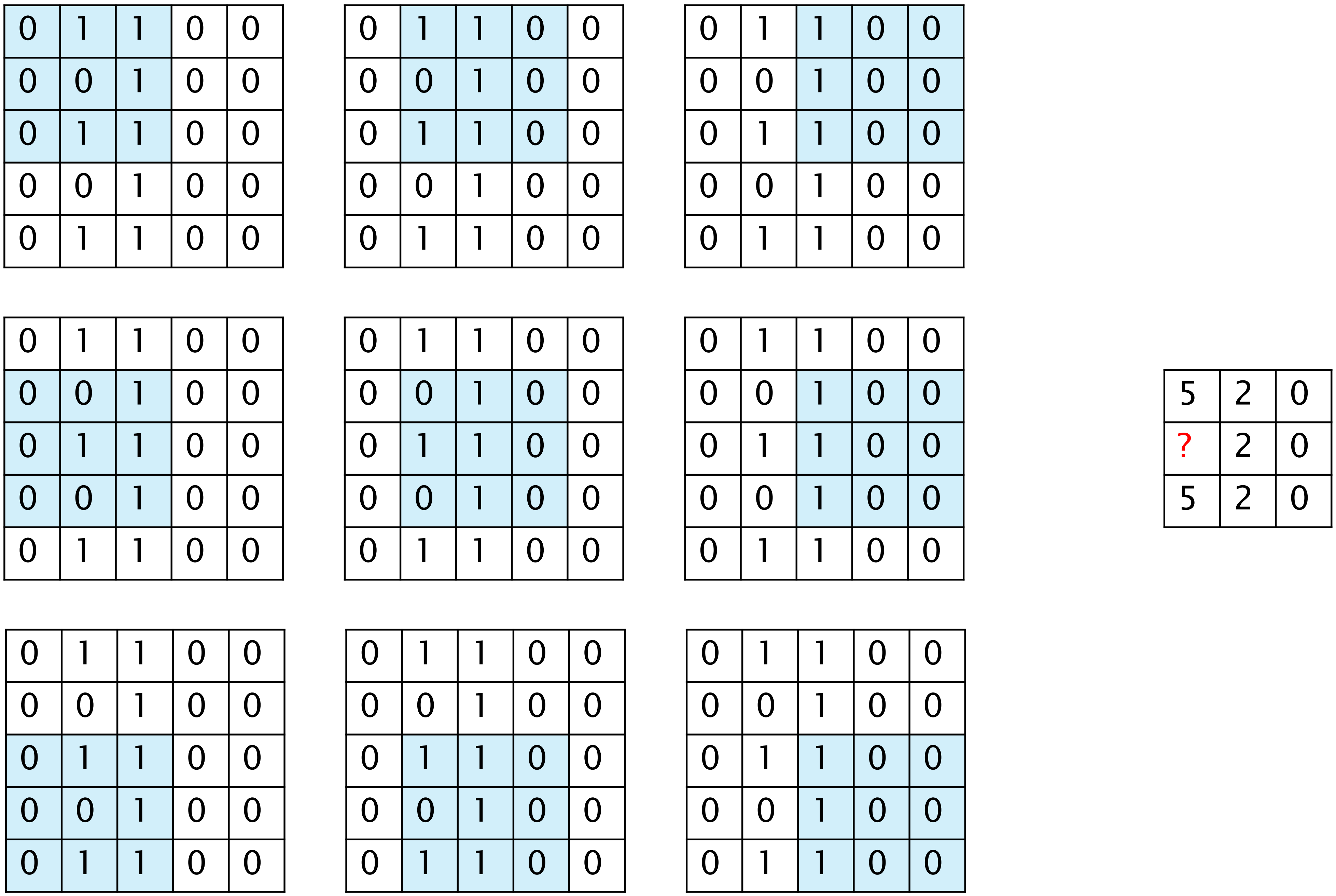
apply

0	1	1
0	0	1
0	1	1

as

0*0	1*1	1*1	0	0
0*0	0*0	1*1	0	0
0*0	1*1	1*1	0	0
0	0	1	0	0
0	1	1	0	0

Result = 5



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

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

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

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

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

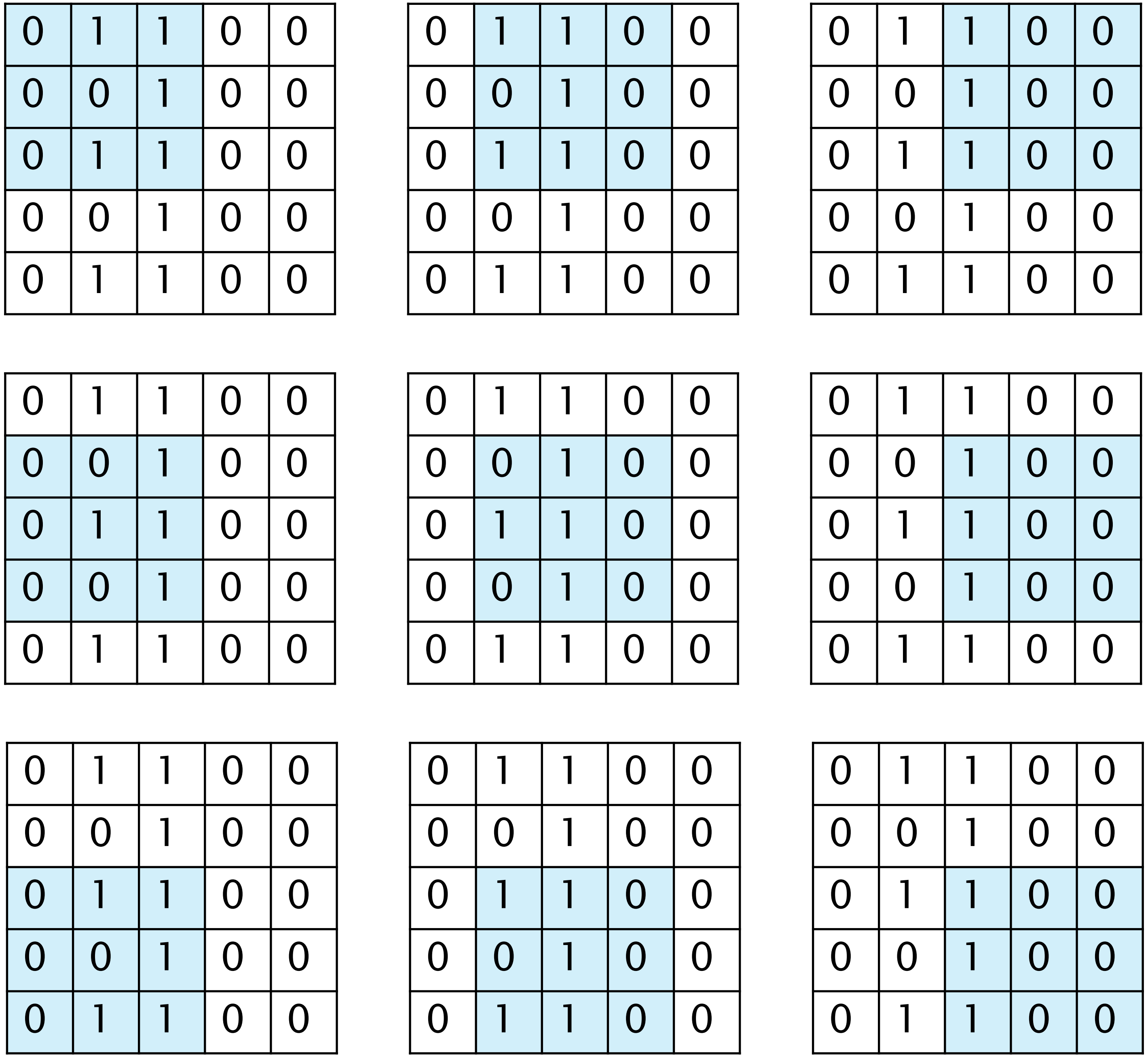
0	1	1	0	0
0	0	1	0	0
0	1	1	0	0
0	0	1	0	0
0	1	1	0	0

5	2	0
?	2	0
5	2	0

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

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

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



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

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

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

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

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

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

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

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

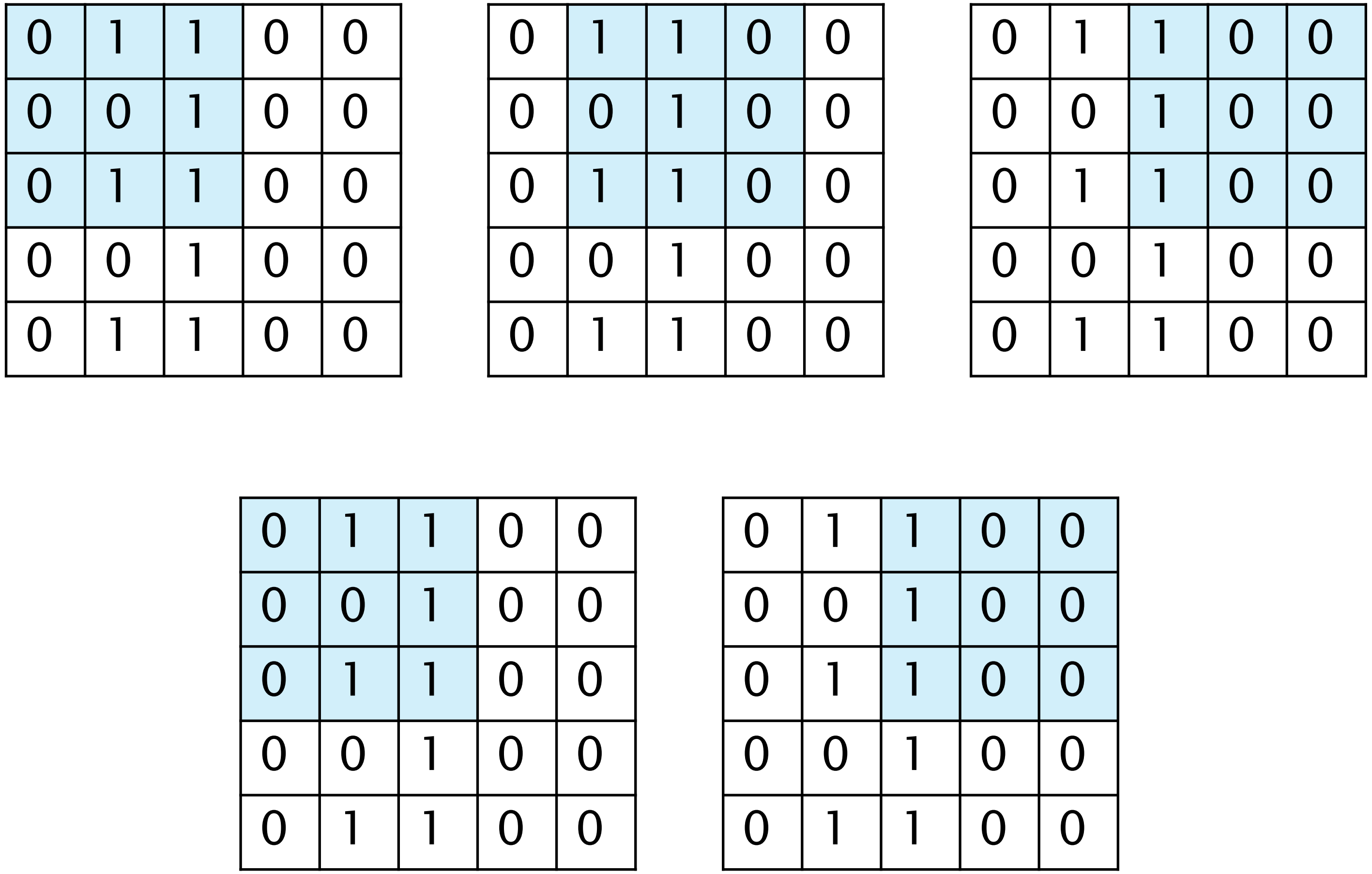
0	1	1	0	0
0	0	1	0	0
0	1	1	0	0
0	0	1	0	0
0	1	1	0	0

5	2	0
3	2	0
5	2	0

Locations with value = sum of the values in kernel

perfectly match the given image

STRIDE



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

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

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

Stride : 1
Move 1 column to right at each step

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

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

Stride : 2
Move 2 columns to right at each step

ZERO PADDING

Size of output = 3x3

If the input matrix is a 10x10 matrix then the output matrix size = ?

ZERO PADDING

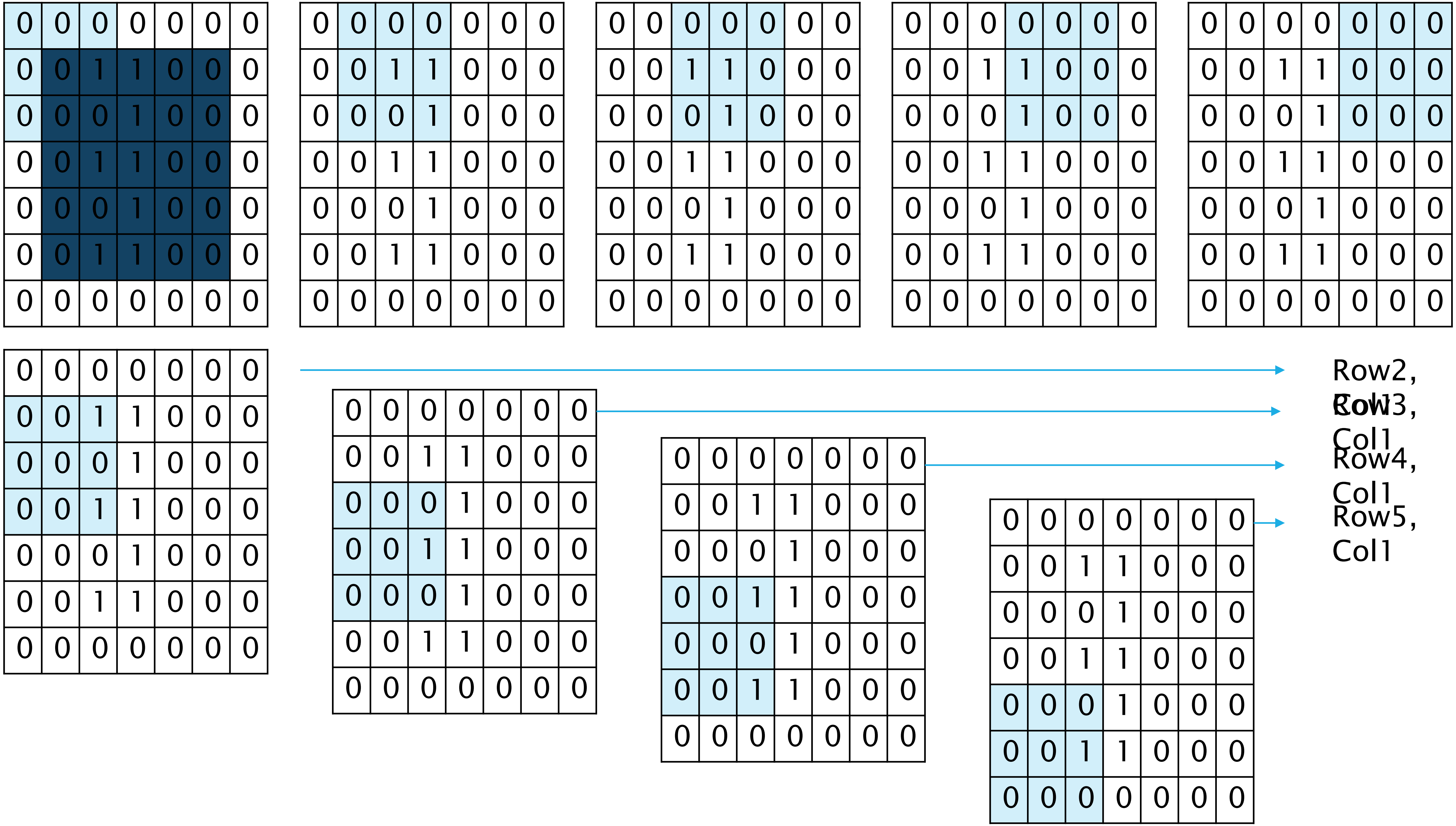
Size of output = 3x3

If the input matrix is a 10x10 matrix then the output matrix size = 7x7

What if Size of output matrix should be equal to the size of the input matrix?

The actual first element should be at the middle when the filter is first applied.

So pad top, left etc with ‘0’, such that filter can be applied with every actual element



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

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

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

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

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

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

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

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

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

WHAT’S ALL THIS GOT TO DO WITH CNN?

Matrix after applying filter on different receptive fields = convolved matrix

Also called activation map/feature map

In our example is the feature map when 9 filters are applied.

5	2	0
3	2	0
5	2	0

Or example is analogous to the working of convolution layer.

Input to convolution layer : image

Output : Value after applying filter

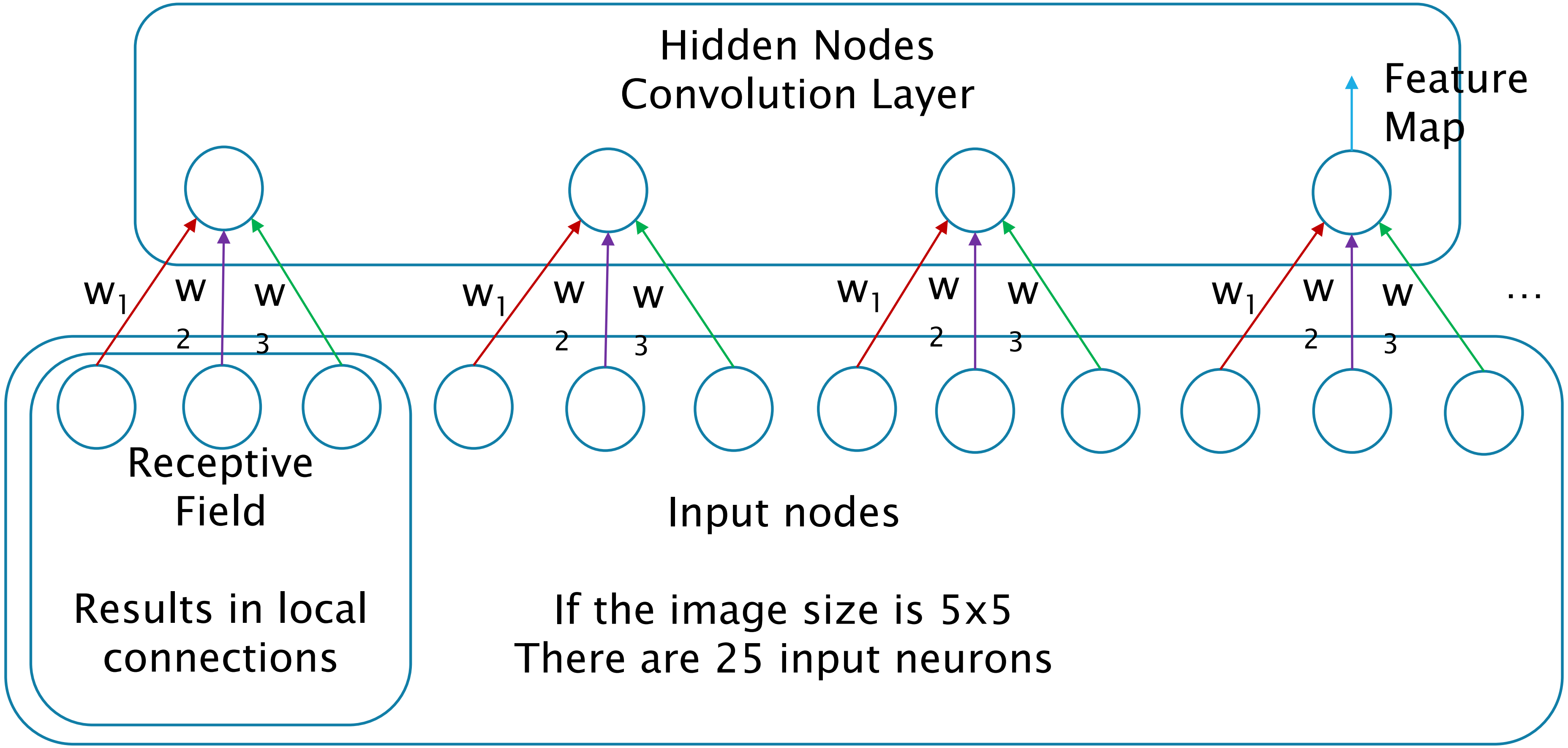
BUT…

Deep learning does not require the programmer to provide the feature

– i.e. in our case – the filter

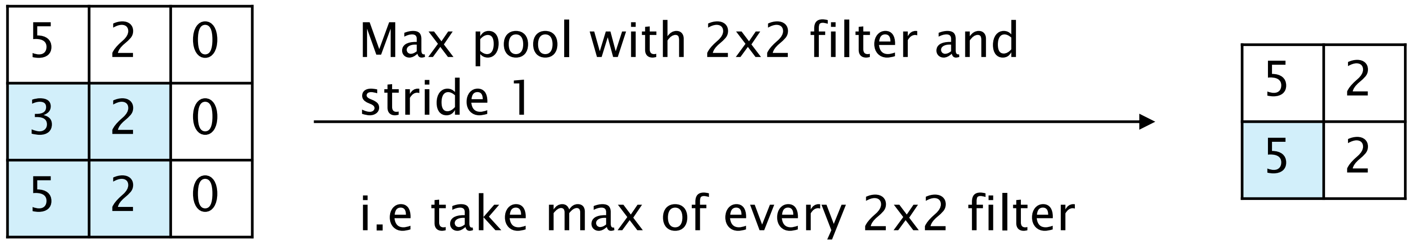
So

– the filter needs to be learnt by the neural network model and not given as input



Filter : w1, w2, w3
Training Phase : The weights are learnt using backpropagation
Activation Function(non-linear) : ReLu or tanh
Output of conv layer – given as input to activation function

MAX POOLING

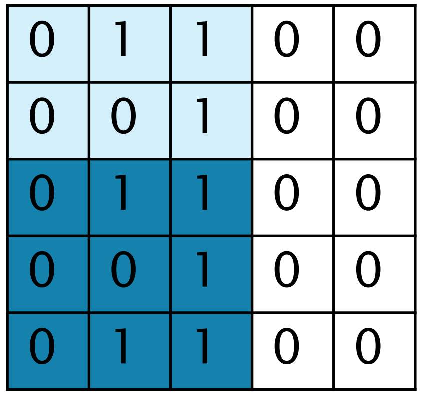


5	2	0
3	2	0
5	2	0

Max pool with 2x2 filter and stride 1

i.e take max of every 2x2 filter

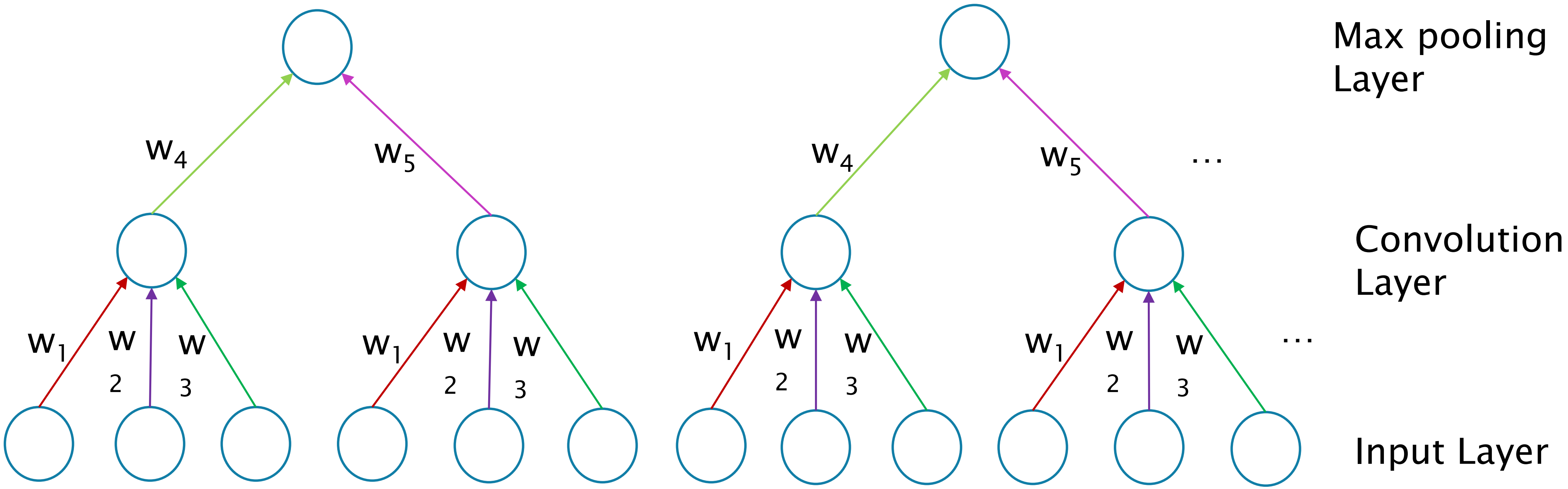
5	2
5	2



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

This results in downsampling and detecting more abstract features
While the filter in the conv layer might detect edges
This layer may detect more abstract things like objects

It has actually found the similarity between the upper and lower portions of the image



Max pooling Layer

Convolution Layer

Input Layer

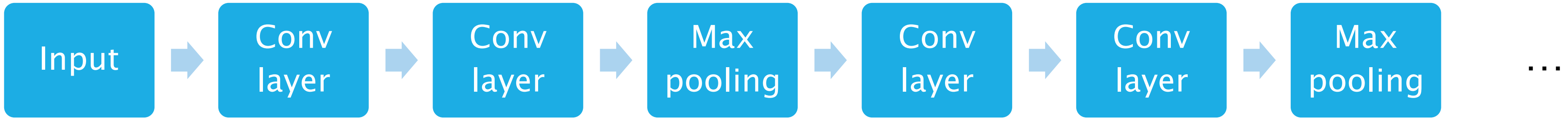
This layer may or may not be present in a CNN
Average pooling is also considered
But max pooling found to give better results empirically

A PARTIAL CNN WITH MANY HIDDEN LAYERS

Has multiple hidden layers of convolution layers and max pooling layers

Different filters applied in different layers

Example Partial CNN



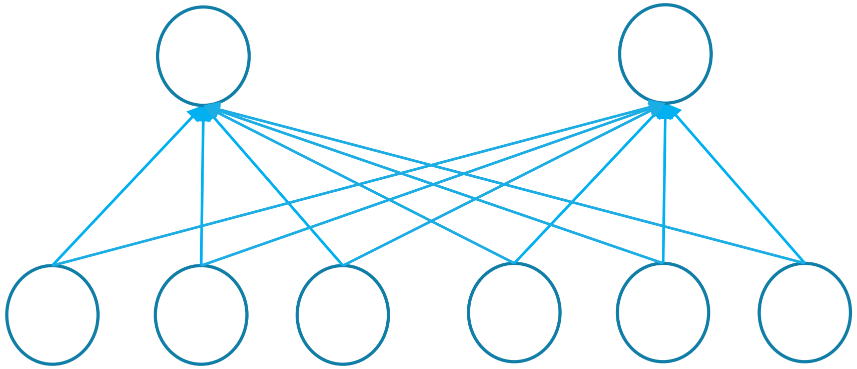
FULLY CONNECTED LAYER

Last layer

Neurons in this layer are fully connected to all neurons in the previous layer

They can have different weights and bias as normal ANN

The weights are multiplied with the output of the previous layer and bias is added – which is input to the activation function



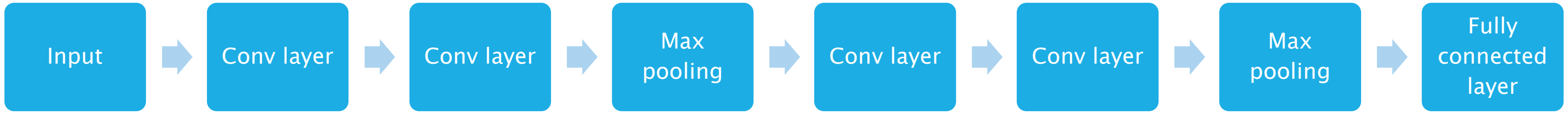
OUTPUT

Usually multiple neurons – based on the number of classes

 – each outputs probability with which a class occurs

If differentiate horse or not

 2 output neurons – with probability with which the image is/is not a horse



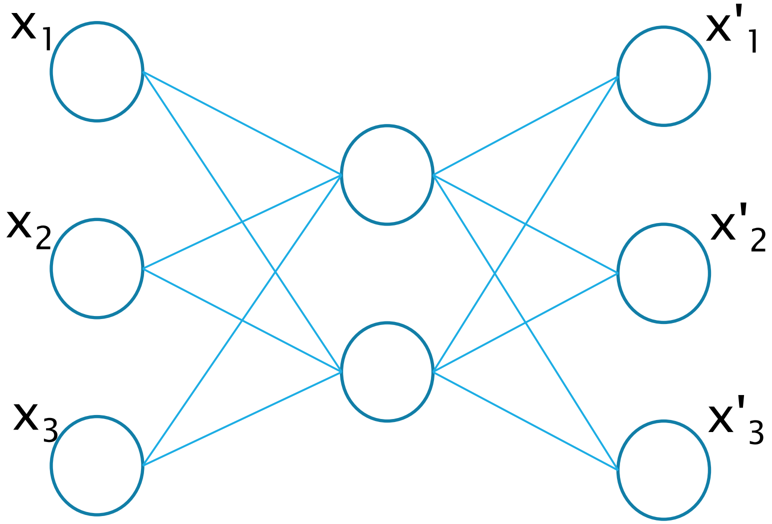
OUTLINE OF OTHER NETWORKS

AUTOENCODER

Unsupervised learning technique

Uses backpropagation

Tries to give output that is very similar to the input



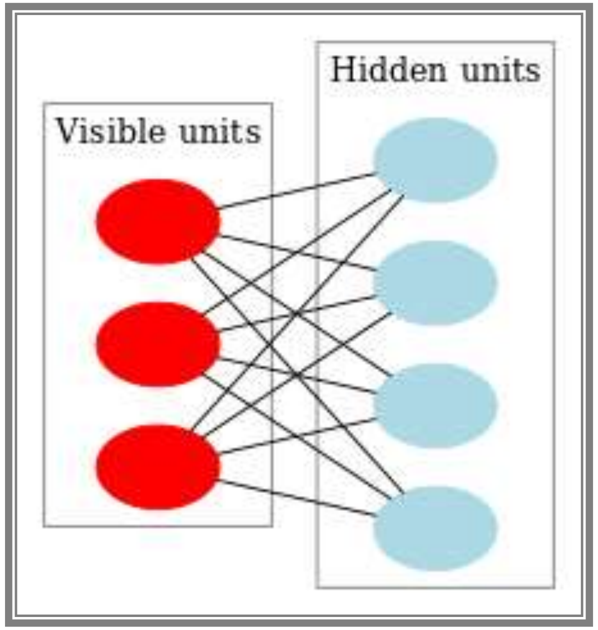
RESTRICTED BOLTZMANN MACHINE

Shallow 2 layer net

Connections

- Fully connected
  - Every node in visible layer connected to every node in hidden layer
- Undirected
- Restricted – no connection between nodes in a layer

Training : Uses contrastive divergence or approximate gradient descent



Action

Heist

DATA – TO – VISIBLE UNIT

User 1: (MI= 1, F&F = 1, Die Hard = 1, Italian = 0, Oceans = 0, Bank Job = 0).

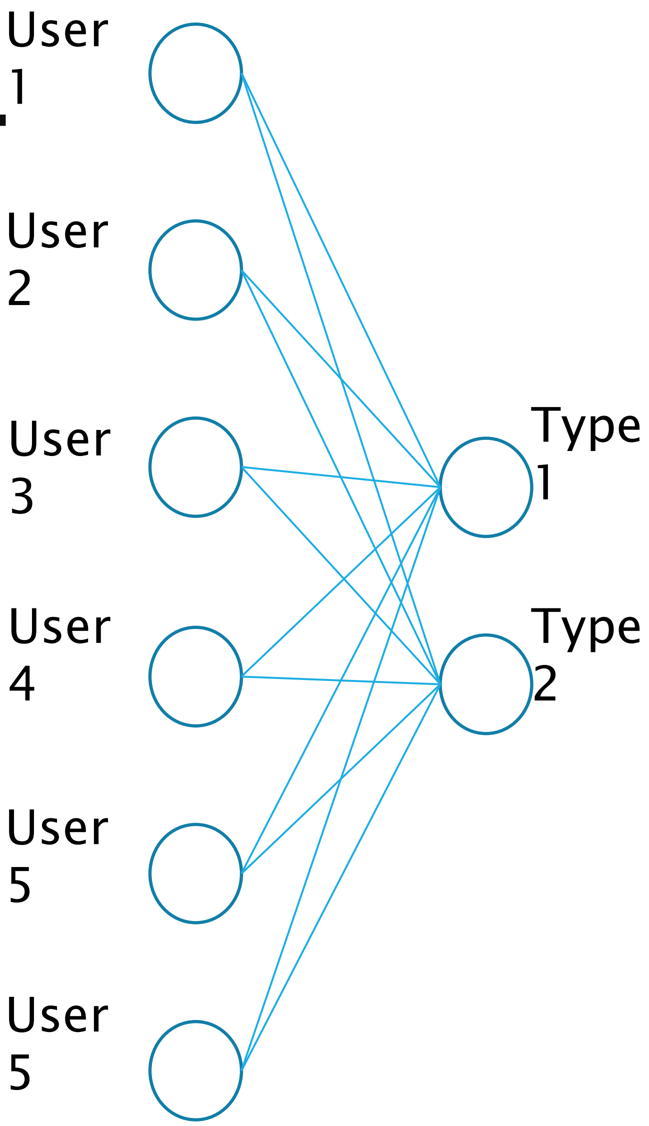
User 2: (MI = 1, F&F = 0, Die Hard = 1, Italian = 0, Oceans = 0, Bank Job = 0).

User 3: (MI = 1, F&F = 1, Die Hard = 1, Italian = 0, Oceans = 0, Bank Job = 0).

User 4: (MI = 0, F&F = 0, Die Hard = 1, Italian = 1, Oceans = 1, Bank Job = 0).

User 5: (MI = 0, F&F = 0, Die Hard = 1, Italian = 1, Oceans = 1, Bank Job = 0).

User 6: (MI = 0, F&F = 0, Die Hard = 1, Italian = 1, Oceans = 1, Bank Job = 0).



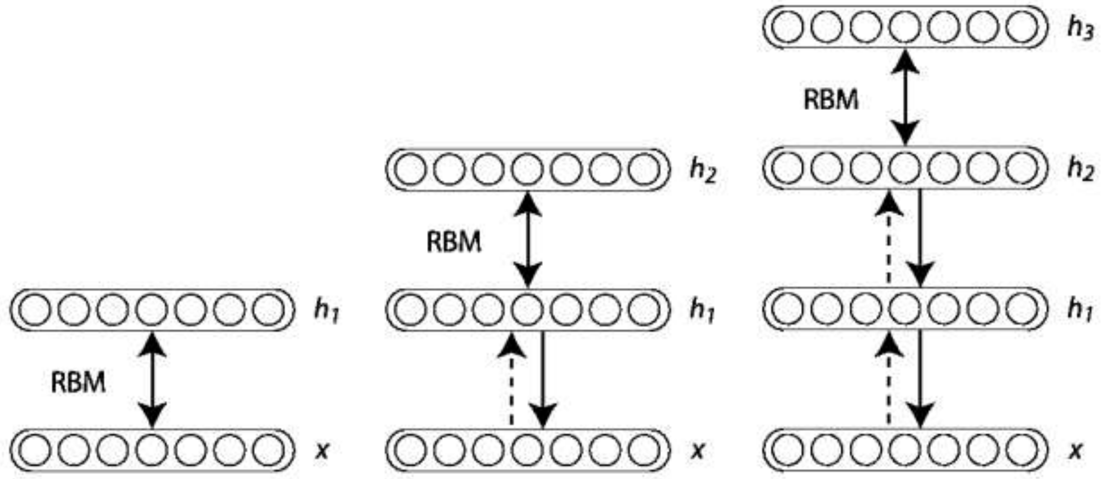
HIDDEN UNIT – TYPE OF MOVIE INTERESTED IN

- User 1: (MI = 1, F&F = 1, Die Hard = 1, Italian = 0, Oceans = 0, Bank Job = 0). Likes action movies
- User 2: (MI = 1, F&F = 0, Die Hard = 1, Italian = 0, Oceans = 0, Bank Job = 0). Prefers action movies, but doesn’t like F&F.
- User 3: (MI = 1, F&F = 1, Die Hard = 1, Italian = 0, Oceans = 0, Bank Job = 0). Likes action movies.
- User 4: (MI = 0, F&F = 0, Die Hard = 1, Italian = 1, Oceans = 1, Bank Job = 0). Generally likes heist movies.
- User 5: (MI = 0, F&F = 0, Die Hard = 1, Italian = 1, Oceans = 1, Bank Job = 0). Generally likes heist movies.
- User 6: (MI = 0, F&F = 0, Die Hard = 0, Italian = 1, Oceans = 1, Bank Job = 1). Likes heist movies.

Based on the type liked the corresponding hidden unit is activated more

DEEP BELIEF NETWORK

Multiple RBMs/MLPs



RECURRENT NEURAL NET

Used when input seen as a sequence of data rather than as a bag of elements

Activation Function : tanh or ReLu

Learning weights : Back Propagation Through Time(BPTT)

Problem : Vanishing Gradient, Exploding Gradient

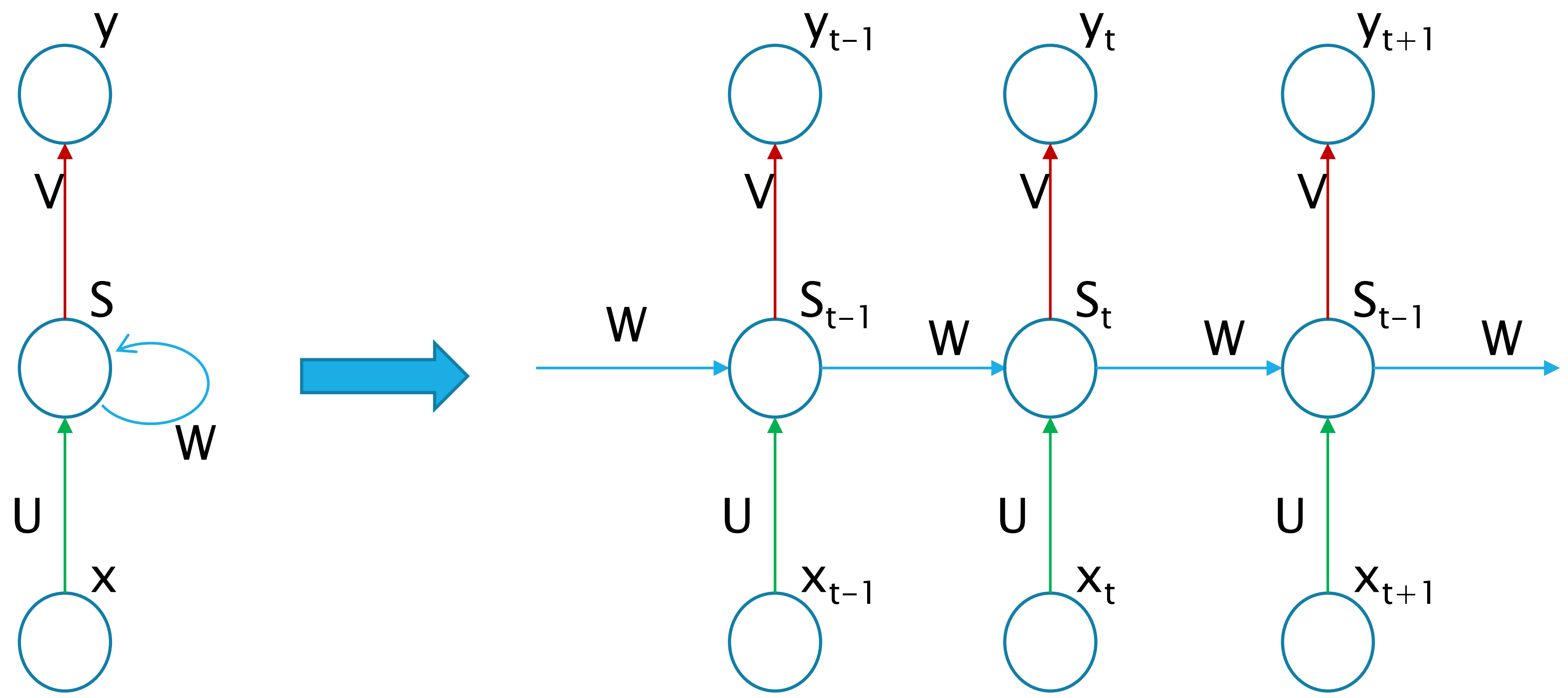
Solutions : Long Short Term Memory(LSTM), Gated Recurrent Units(GRU)

Applications : NLP, Machine Translation, Speech Recognition

RECURRENT NEURAL NET

Has feedback loop

Tries to store previous states in memory – but not able to retain long term information



PARAMETERS IN THE NETWORK

x_t – input at time unit ‘t’

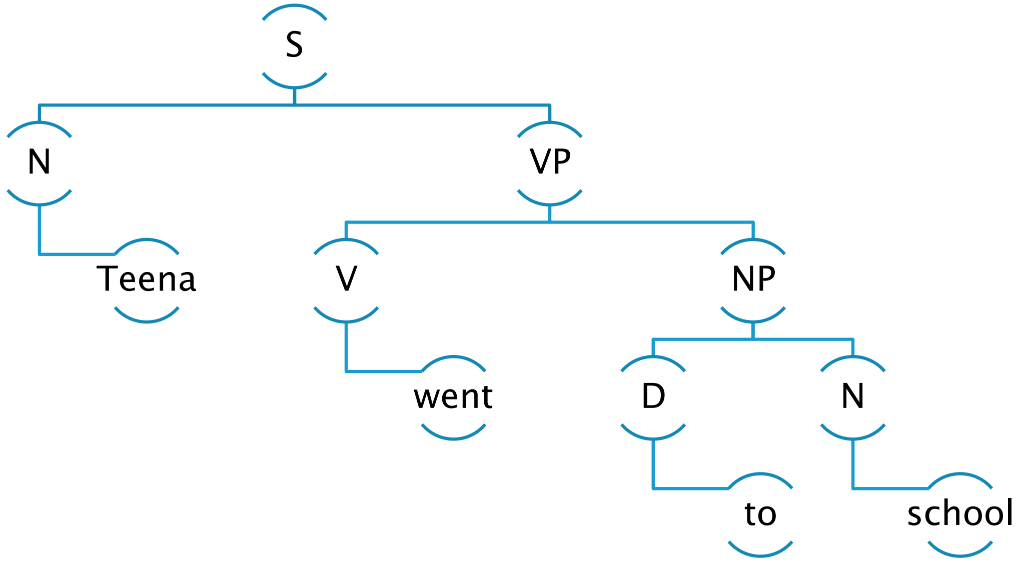
s_t – hidden state at time unit ‘t’

– input to s_t is $(u.x_t + w.\ s_{t-1})$ which is passed through activation function

s_{-1} = all zeros – initially

y_t – output at time unit ‘t’

$= \text{softmax}(v.s_t)$



RECURSIVE NEURAL TENSOR NET

Analyze data with hierarchical structure

Binary tree construction

Vector representation – given as input to leaves

Application : NLP

REFERENCES

http://blog.echen.me/2011/07/18/introduction-to-restricted-boltzmann-machines/

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

http://cs231n.github.io/convolutional-networks/

http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

http://colah.github.io/posts/2014-07-Understanding-Convolutions/

https://talks.pgaleone.eu/Tensorflow%20&%20CNN%20for%20object%20detectio

REFERENCES

https://github.com/adeshpande3/Tensorflow-Programs-and-Tutorials/blob/master/Convolutional%20Neural%20Networks.ipynb

http://neuralnetworksanddeeplearning.com/chap1.html

https://betterexplained.com/articles/vector-calculus-understanding-the-gradient/

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

TO ACCESS THE PPT

https://www.slideshare.net/lovelynrose/deep-learning-simplified