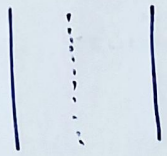
Neuro-computing an introduction

Parking a vechile.



Move left-right until parked properly/perfectly.

↑ The above learning is how our neural network learns.

if labelled data → supervised

if learns by getting feedback + reinforcement learning.

* Fundamental idea behind neural network :→

The primary task of ANN or NN model by to simulate the biological neural network with electronic circuit.

Biological vs ANN.

- BNNs :→
  - primary task is to control various function.
  - The brain consist of billions of neuron connected by synapses
  - It proces the information in a massively parallel manner.
  - It can learn and adapt quickly with minimal ~~effor~~ effor.

→ ANN :-

- → Inspired by BNN.
- → Uses Mathematical Models (neurons) connected in layers.
- → learns by adjusting weight using technique like back propagation.

* ANN are also called as connectionist model / PDP?

Connection list model because knowledge is stored in the connection (weight) between artifical neuron.

PDP (Parallel distributed processing) :-

Because computation occur in parallel across many neuron, similar to how brain proces information.

Purpose of ANN :-

→ To achieve human like performance.

* Similarity b/w ANN & BNN:

→ Gets I/P via synaptic connections.
→ Accumulated input is transformed to a single output.
→ output is transmitted through axon
→ if IP > 0, the neuron fires. (ie if crosses some threshold)
→ Total o/p ← Firing rate.

BNN	ANN.
Synapse (recieve signal)	input connection (weights).
Soma (proces input)	Weighted sum function
Axon (transmit o/p)	o/p neuron activation
threshold based firing	Activation function (RELU, sigmoid) etc.
firing rate control o/p.	Output Magnitude from activation function.

* ANN :→

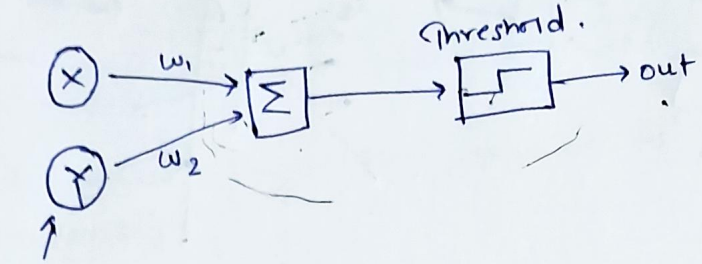
Massively parallel interconnect network, of simple procesing elements which are intended to interact with the object of real word in the same way as biological system.

NN model an extreme simplification of human neural system.

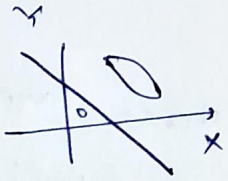
computional elements (neuron/nodes/procesor) are analogus to neuron of Biological system.

* Characteristics.
  - learn from example
  - Generdlize from previous example to new one.
  - abstract essential characteristics from inputs.

Perceptron :- A single neuron connected by weights to a set of inputs.



→ 2 - I/P perceptron.



if $w_1 x + w_2 y > \theta \longrightarrow o/p = 1$

↑ threshold.

$w_1 x + w_2 x = \theta \longrightarrow$ seperating line.

* Learning rule :→

present a set of input pattern, adjust the weights until the desired output occur for each of them.

$w_i(t+1) = w_i(t) + \Delta i ;$

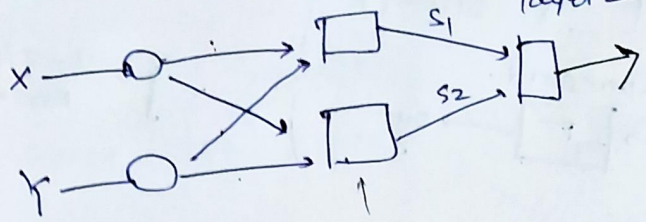
$\Delta i = \eta \delta x_i ;$

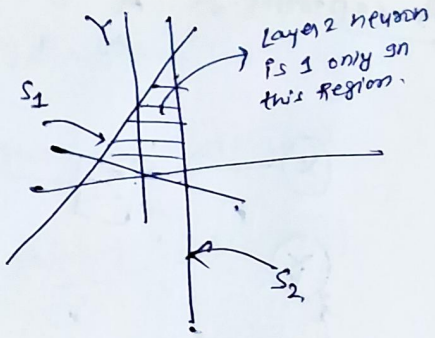
$\delta = T - A \quad \{Target - Actual\}.$

$\eta \rightarrow$ learning rate.

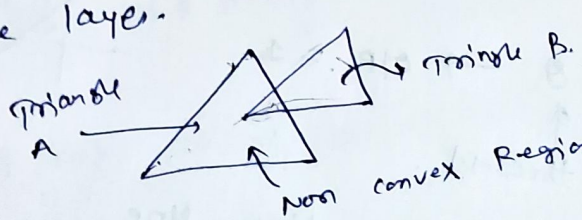
* if the set of pattern are linearly seperable then single layer perceptron is enough.

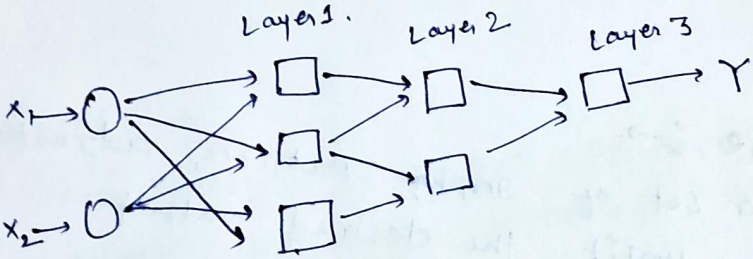
* Two layers: layer 1. layer 2





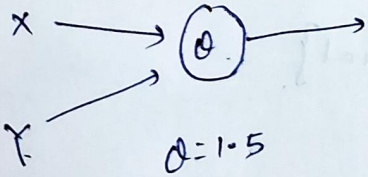
* Three layer.





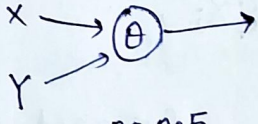
MNN

* Boolean function:-



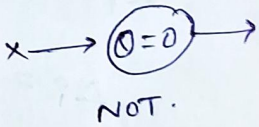
θ = 1.5

And



θ = 0.5

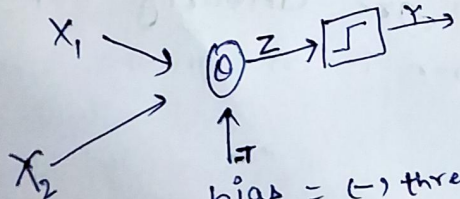
OR



NOT.

$$Z = \sum_i w_i x_i - T$$

$$Y = \begin{cases} 1 & \text{if } Z \geq 0 \\ 0 & \text{else.} \end{cases}$$



bias = (-) threshold.

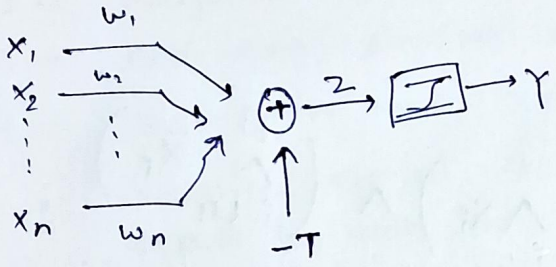
* Soft Perceptron (logistic).

$$Z = \sum_i w_i x_i - T$$

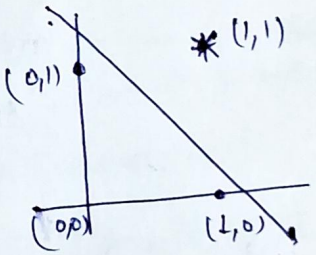
$$Y = \frac{1}{1+\exp(-z)} = \left(\frac{1}{1+e^{-z}}\right) \rightarrow \text{Sigmoid Activation.}$$

A squashing function instead of a threshold at output.

Activation function :- The function that act on weighted combination of input (and threshold).

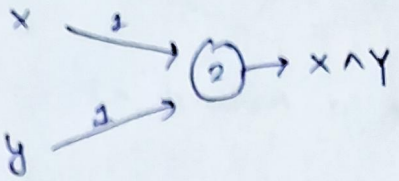


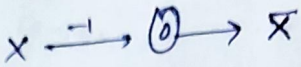
* In general one neuron is enough for linear function. (i.e data will be linearly seperable).

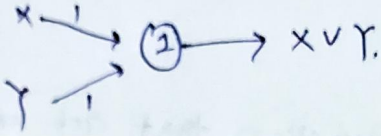


And is linearly seperable.

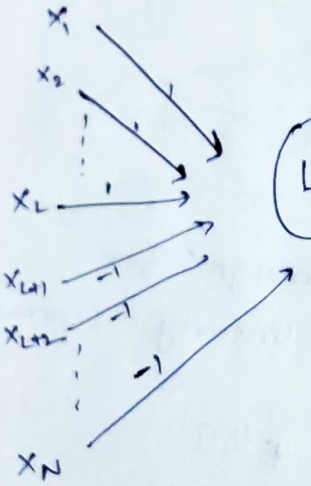
* Perceptron as a Boolean gate :-





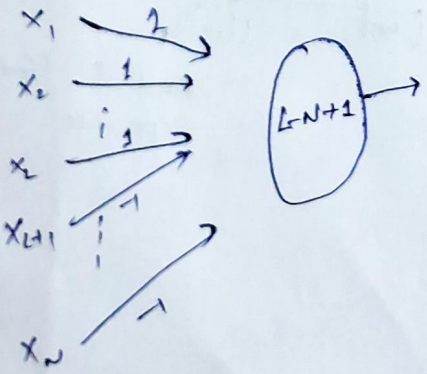


A perceptron can model any simple binary Boolean gate.



$$\left(\bigwedge_{i=1}^{L} x_i\right) \wedge \left(\bigwedge_{i=L+1}^{N} \bar{x}_i\right)$$

will fire only if all x_1 to x_L are 1 and x_{L+1} to x_N are 0.



$$\left(\bigvee_{i=1}^{L} x_i\right) \vee \left(\bigvee_{i=L+1}^{N} \bar{x}_i\right)$$

if any from x_1 to x_L are 1
or
any from x_{L+1} to x_N are 0.

for atleast 1 < at place of 1 in threshold put 1.

ANN (MLP)

→ A NN consist of layer of artifical neuron and connection b/w them.

each connection is associate with weight.

Training done to get right weight and bias so that error get minimized.

* NN Learning:→

1. → Input fed into I/p layer.
2. → Weighted o/p fed into hidden layer.
3. → Activation function applied.
4. → repeat to next layers.
5. → Weighted o/p of last hidden layer are inputs to units making up the o/p layer.

* A Network containing 2 hidden layers is called 3-layer neural network.

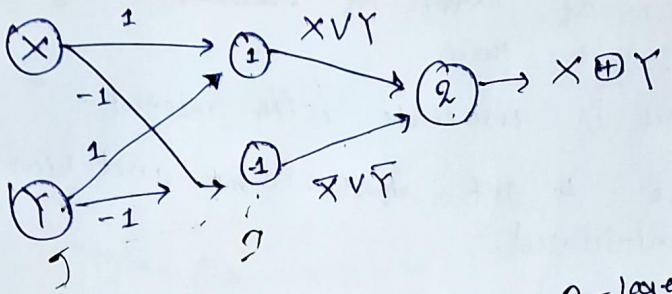
* Decision boundary:-

→ (depend on activation function)
↳ on 2-hidden layer.

0- hidden layer → linear classifier.
1- hidden layer → boundary of convex region. → open or closed.
2- hidden " → combination of convex region.

$z = \sum(w_i \cdot x_i) + b$ → decision boundary $z = 0$.

XOR



↳ can be computed using MLP, 2-layer NN.

* MLPs can compute any boolean function.

* Input to node:

$$I_i = \sum_{\text{All connected node.}} \text{Weight} * \text{o/p of previous node.}$$

output of I^{th} node = $f(I_i)$

↑ Activation function.

* Weight updation:-

$$E = \frac{1}{2} \sum (t_i - o_i)^2$$ → for learning correct weight reduce E as much as possible.

(1) Gradient decent technique:-

$$\Delta w_{ji} \alpha -\frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial I_j} \cdot \frac{\partial I_j}{\partial w_{ji}}$$

$$= \eta \delta_j o_i$$

$$\delta_j = \frac{-\partial E}{\partial I_j} = \frac{-\partial E}{\partial o_j} \frac{\partial o_j}{\partial I_j} = -\frac{\partial E}{\partial o_j} f'(I_j).$$

For nodes on o/p layer.

$$\Delta w_{Ji} = \eta \left(-\frac{\partial E}{\partial O_J}\right) f'(I_J) O_i .$$

For hidden layer.

$$\frac{\partial E}{\partial O_J} = \sum_K \frac{\partial E}{\partial I_K} \frac{\partial I_K}{\partial O_J} = \sum_K \frac{\partial E}{\partial I_K} \frac{\partial \sum_J w_{KJ} O_i}{\partial O_J}$$

$$= \sum_K \frac{\partial E}{\partial I_K} w_{KJ}$$

$$= \sum_K (-\delta) w_{KJ} .$$

$$\Delta w_{Ji} = \eta \left(\sum_K \delta_K w_{KJ}\right) f'(I_J) O_i$$

if $O_J \rightarrow$ sigmoid.

then $f'(O_J) = O_J (1 - O_J)$

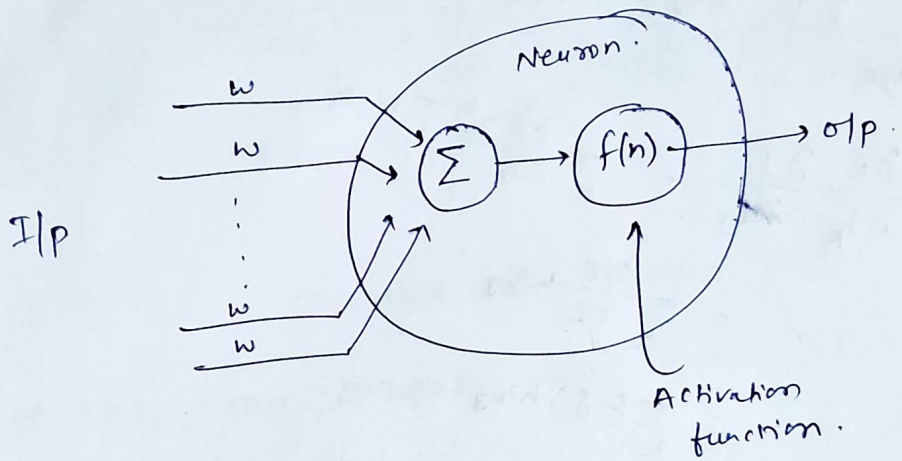
NOTE :→.

$$\Delta w_{Ji} = \begin{cases} \eta \left(-\frac{\partial E}{\partial O_J}\right) O_J (1 - O_J) O_i & \rightarrow \text{o/p layer.} \\ \eta \left(\sum_K \delta_K w_{KJ}\right) O_J (1 - O_J) O_i & \rightarrow \text{hidden layer.} \end{cases}$$

large value of η will lead to oscilation.

So to just increase the momentum without oscilation add bias $\alpha \Delta w_{Ji}(t)$.

Artificial Neuron.



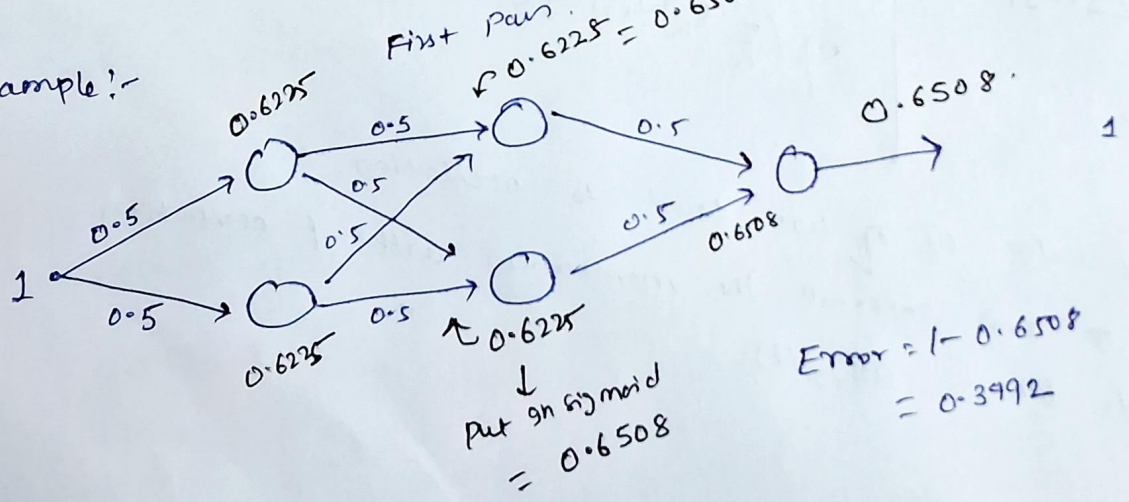
* Back- propagation:-

[Gradient of neuron (G) = Slop of activation / transfer function.

* [Σ{(weight of neuron to the next neurons) * (Gradient of next neuron)}].

[Gradient of o/p :- slope of transfer function * error.]

example:-



CNN
(Convolution Neural network)

Neural nfw with ~~atleast~~ convolution operation instead of Matrix Multiplication in at least one of the layers.

Architecture :→

CNN is a deep learning model designed for image processing and pattern recognisation. It extracts spatial hierarchies of feature from an image using convolutional layers.

A CNN consist of four types of layer :.

1 → Input layer

takes an image (e.g RGB of size 224 *224 *3)

No of channel depend on the image type.

(1 for grayscale, 3 for RGB).

2 → Convolutional Layer (feature extraction) :→

- Applies filter/kernal (small matrices like 3x3 or 5x5) to detect feature.
- Captures edge, texture, shapes and patterns.
- Output :- A feature ~~Mapped~~ (transformed representation of the image).
- Op- Size = $\frac{(I - F + 2P)}{S} + 1$

I → Input size. P → padding
F → filter size S → stride.

ex:- I = 32×32
f = 3×3
Stride = 1
padding = 0

o/p feature maps = 30 × 30

3→ Activation function:- (non-linearity).
Typically ReLU (Rectified Linear unit) is used

$$f(x) = \max(0, x)$$

- introduces non-linearity in model, allowing it to learn complex feature.

4→ Pooling layer (Down-sampling)

→ Reduces feature map size while preserving important information.

Types:-
- Max pooling → Takes the maximum value in small window (e.g 2×2).
- Average pooling → Takes the average value.

ex:- 32 × 32 feature map
↓ after 2×2 pooling.
16×16

5→ fully connected layer (classification) :→
- flatten the feature map into a 1D vector.
- uses dense fully connected layers to make prediction.

ex:- if predicting cat or dog → final layer has 2 neuron.

6 → Output layer (softmax / sigmoid) →

- Softmax → used for multi-class classification.
- Sigmoid → used for binary class classification.

* Convolution :-

It is a fundamental mathematical operation used in image processing, signal processing, and deep learning CNNs. It helps to extract feature, enhance image and perform transformation like smoothning, sharpening and edge detection.

$$y(m,n) = \sum_i \sum_j x(i,j) \cdot h(m-i, n-j)$$

$x(m,n)$ → I/P image

$h(m,n)$ → kernal/mask (filter)

$y(m,n)$ → feature map.

each value in the output is a weighted sum of near by input values, with weights given by kernal.

example :-

$$x(m,n) = \begin{pmatrix} 4, & 5, & 6 \\ 7, & 8, & 9 \end{pmatrix}$$

$$y[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2)\, h[n_1 - k_1, n_2 - k_2]$$

$$h(m,n) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

perform convolution b/w x and h.

$$x(m,n) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}_{2\times 3} \end{array}$$

$x(0,0) = 4$
$x(0,1) = 5$
$x(0,2) = 6$
$x(1,0) = 7$
$x(1,1) = 8$
$x(2,2) = 9$

$$h(m,n) = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{c} 0 \end{array} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}_{3\times 1} \end{array}$$

$h(0,0) = 1$
$h(1,1) = 1$
$h(2,1) = 1$

dim of o/p :- $\dim(x) + \dim(h) - 1$

$= 4 \times 3$

$$y(m,n) = \begin{pmatrix} y(0,0) & y(0,1) & y(0,2) \\ y(1,0) & y(1,1) & y(1,2) \\ y(2,0) & y(2,1) & y(2,2) \\ y(3,0) & y(3,1) & y(3,2) \end{pmatrix}$$

$$y(m,n) = \begin{bmatrix} 4 & 5 & 6 \\ 11 & 13 & 15 \\ 11 & 13 & 15 \\ 7 & 8 & 9 \end{bmatrix}$$

done zero padding when needed.

Ex:-2:

$$x(m,n) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$h(m,n) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$y(m,n) = \begin{pmatrix} 1 \\ 5 \\ 12 \\ 11 \\ 7 \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 1 & 2 & 3 \\ 5 & 7 & 6 \\ 12 & 15 & 9 \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 1 & 3 & 3 & 0 \\ 5 & 12 & 6 & 0 \\ 12 & 27 & 9 & 0 \end{pmatrix}$$

$$: \begin{pmatrix} 1 & 3 & 3 & 5 & 3 \\ 5 & 12 & 6 & 16 & 9 \\ 12 & 27 & 15 & 33 & 18 \\ 11 & 24 & & & \\ 7 & 15 & & & \end{pmatrix}$$

I/p matrix $x(m,n)$ and $h(m,n)$

$$x(m,n) = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}_{2\times 3}$$

$$h(m,n) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}_{3\times 1} \leftarrow h(-m,-n)$$

dimension of resultant matrix = { no of rows (m,n) + no of rows $h(m,n)$ − 1 ; no of column of $x(m,n)$ + no of column of $h(m,n)$ − 1 }

op = 4×3.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 6 \\ 11 & 13 & 15 \\ 11 & 13 & 15 \\ 7 & 8 & 9 \end{bmatrix}$$

Q:2 → $x(m,n) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ $h(m,n) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ ↓ $h(-m,-n)$

op dim = (3+3−1) × (3+2−1)

= 5×4.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 5 & 3 & 0 \\ 5 & 12 & 14 & 9 & \\ 12 & 27 & 33 & 18 & \\ 11 & 24 & 28 & 15 & \\ 7 & 15 & 17 & 9 & \end{bmatrix}$$

Q:8: $x(m,n) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ $h(m,n) = (3,4,5)$

Linear convolution

o/p size : $(3+1-1) * (3+3-1)$

$= 3 \times 5.$ $h[m,-n) = (5,4,3)$

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

$\Rightarrow$ [5 ... 14]

$= \begin{bmatrix} 3 & 10 & 22 & 22 & 15 \\ 12 & 31 & 48 & 49 & 80 \\ 21 & 52 & 94 & 76 & 45 \end{bmatrix}$

* Co-relation:

↳ $h(m,n)$ ← as it is, perform same above operation.

* Determine - co-relation b/w two matrices:.

$x_1[m,n] = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$ $x_2[m,n] = \begin{bmatrix} 1 & 5 \\ 2 & 3 \end{bmatrix}$ $\Rightarrow x_2[-m,-n] = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix}$

Step 1 :- rotate one matrix by 180° i.e $x_2[-m,-n)$

Step 2 :- Now perform linear convolution b/w $x_1[m,n]$ and $x_2[-m,-n]$.

2a :- formation of block matrix

→ equal to number of columns :-

Here we have 2 columns in X_1 hence.

2 block matrix will be there.

H_0 & H_1.

2b :- H_0 will be formed from first row of X_1.

i.e 3 1

$$\begin{bmatrix} 3 & 0 \\ 1 & 3 \\ 0 & 1 \end{bmatrix}$$

2b :- H_1 will be formed from 2nd row of $X_1[m,n]$

2, 4

$$H_1 = \begin{bmatrix} 2 & 0 \\ 4 & 2 \\ 0 & 4 \end{bmatrix}$$

Step 3 :- formation of block - toplitz matrix.

$$A = \begin{bmatrix} H_0 & 0 \\ H_1 & H_0 \\ 0 & H_1 \end{bmatrix}$$

No of zero to be appended in block matrix A depend on the number of rows of $X_2[m,n]$.

Now substitute the value of H_0 & H_1.

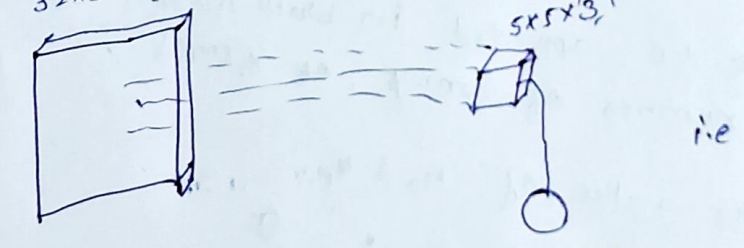
$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix} * \begin{bmatrix} 3 \\ 2 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 \\ 9 \\ 2 \\ 10 \\ 22 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} H_0 & 0 \\ H_1 & H_0 \\ 0 & H_1 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 \\ 4 & 2 & 1 & 3 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix} * \begin{bmatrix} 3 \\ 2 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 \\ 9 \\ 2 \\ 21 \\ 24 \\ 9 \\ 10 \\ 22 \\ 4 \end{bmatrix}$$

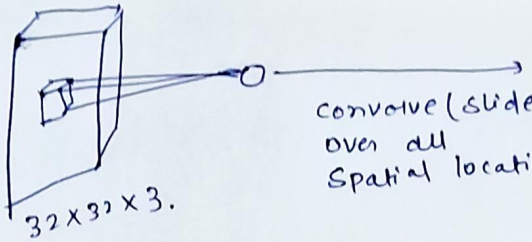
$$Y[m,n] = 3\times3 = \begin{bmatrix} 9 & 9 & 2 \\ 21 & 24 & 9 \\ 10 & 22 & 4 \end{bmatrix}$$

* Convolution layer :-



Result of taking dot product b/w small chunk & filter.
i.e $5*5*3 = 75$-dimensional dot product + 1 bias.

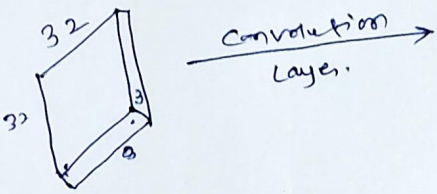
$w^T x + b$



28 x 28 x 1.

Activation map.

if we have 6, 5x5 filters then we'll get seperate activation maps.



will stack up all those to get a new image of size 28 x 28 x 6