28-Jan-2023

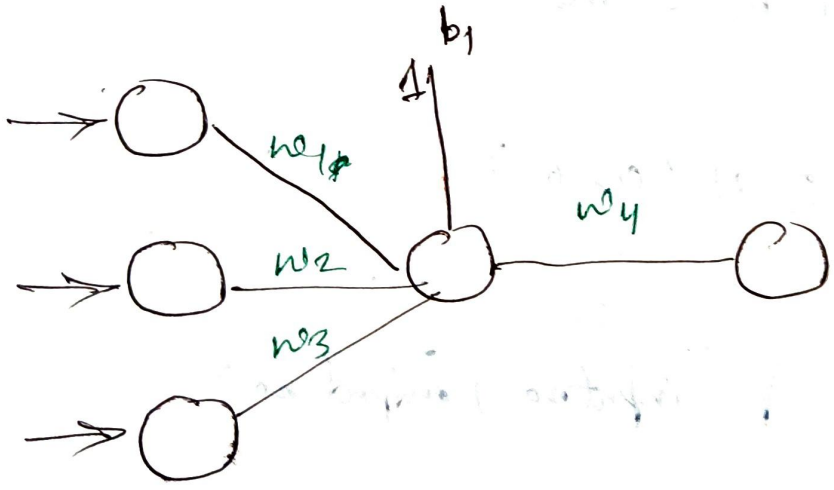
Weight Initialization Techniques

1. Uniform Distribution
2. Xavier / Glorot Initialization
3. Kaiming He Initialization

Key Points / Assumptions

1. Weights should be small.
2. Weights should not be same
   - If weights are same, then it will not be able to perform the task.
3. Weight should have good variance.

NOTE: 1. All the weight initialization techniques and its assumptions is applicable for biased.
2. Weight should be different, because wherever the weight is different that neuron will be activated. If weight will be same then all the neuron gets activated at the same.



input = 3
output = 1

Random Initialize {Huge value}

- Random Initialization can be done using above three techniques.

① Uniform distribution

w_{ij}

i → w.r.t layers

j → w.r.t. layers neurons

$w_{ij} \sim$ Uniform Distribution $\left[\frac{-1}{\sqrt{\text{input no.}}}, \frac{1}{\sqrt{\text{input no.}}} \right]$ — Range

if no. of input = 3

Range = $\left[\frac{-1}{\sqrt{3}}, \frac{1}{\sqrt{3}} \right]$ { w_1, w_2, w_3 will get initialize using this range }

② Xavier / Glorot Initialization

Researchers → Xavier Glorot

(2.1) Xavier Normal Initialization

- weight belongs to normal distribution where mean will be 0 and standard deviation will be σ.

$$w_{ij} \sim N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{(\text{input no.} + \text{output no.})}}$$

2.2 Xavier Uniform

- Weights belongs to uniform distribution.

$W_{ij} \sim$ Uniform distribution $\left[\frac{-\sqrt{6}}{\sqrt{\text{input no.} + \text{output no.}}}, \frac{\sqrt{6}}{\sqrt{\text{input no.} + \text{output no.}}} \right]$ (Range)

3 Kaiming He Initialization

3.1 He Normal

- $W_{ij} \sim N(0, \sigma)$

$$\sigma = \sqrt{\frac{2}{\text{input no.}}}$$

3.2 He Uniform

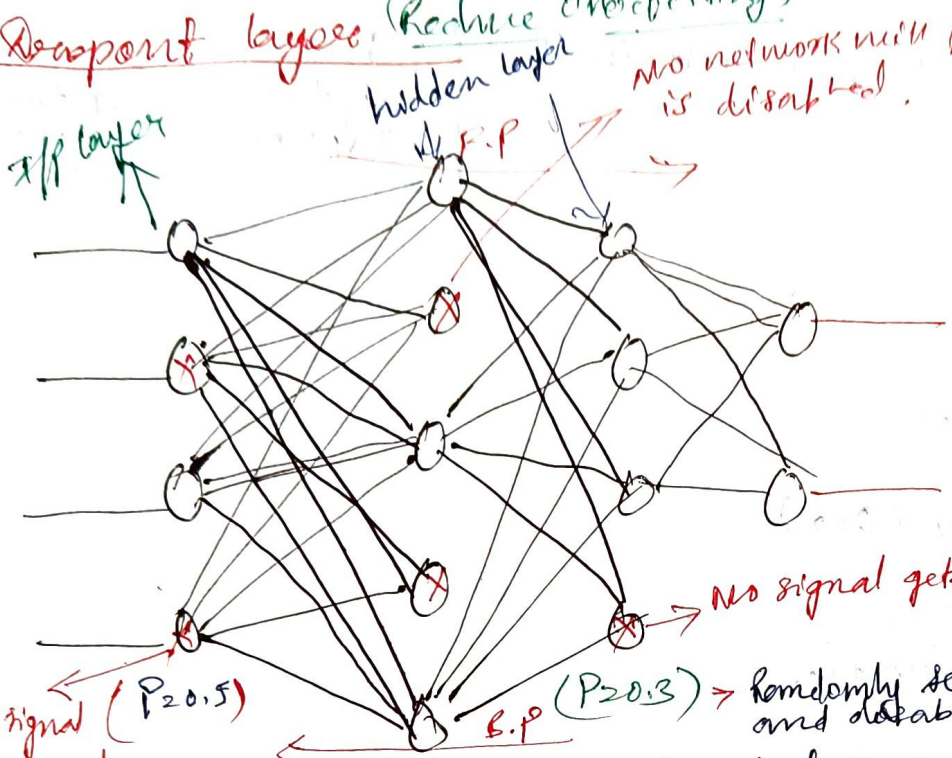
$W_{ij} \sim$ Uniform distribution $\left[-\sqrt{\frac{6}{\text{input no}}}, \sqrt{\frac{6}{\text{input no.}}} \right]$ (Range)

Note: ① In most of the time, we use Xavier/Glorot distribution.

Q. Can the ANN leads to overfitting?

Ans Yes, it also leads to overfitting problem.
To avoid this problem, we use dropout layers.

Dropout layers (Reduce Overfitting)



No network will pass because it is disabled.

No signal gets processed.

No signal processed. ($P \geq 0.5$)

($P \geq 0.3$) → Randomly select 30% of neurons and disable it.

($P \geq 0.5$) → Randomly select 50% of neurons and disabled it.

→ We directly select P value.

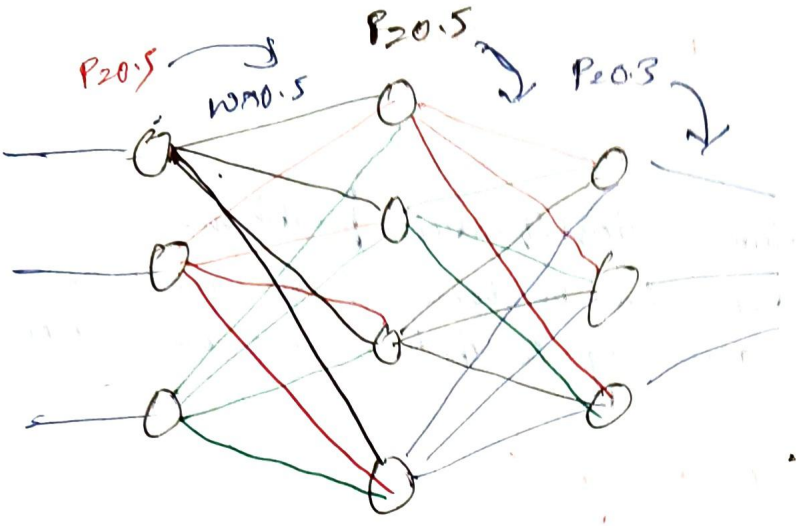
$0 \leq P \leq 1$ { P = Probability }

→ Again F.P happen, and P-value make sure that it will not select the same neurons.

→ In B.P, ~~this~~ weight will not get initialize to disabled neuron.

Q. What will happen to test data?

Ans We only do predictions. Hence, probability and weights gets multiplied.



⇒ P-value can be applied to input layer as well.

⇒ Dropout layer is use for creating a generalize model.

Convolutional Neural Network (CNN):

Q. Why do we use CNN?

① To solve Regression & classification problem.

② Image classification can be done but will not get very good accuracy. So, we use CNN.

{CNN → I/P = Images}

Cerebral cortex and Visual cortex

Visual cortex

- Visual cortex (V1-V5) → {Region of the brain that receives, integrates and process visual information relayed from retinas}.

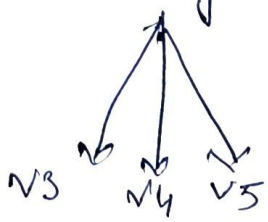
Occipital lobe

- Capture visual information

V1 layer → Primary visual cortex {orientation of edges and lines}

V2 layer → {Responsible for finding differences in colors, complex, pattern, object recognitions}

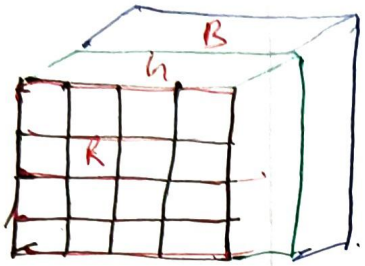
Yolo, RCNN, detection



Visualise the image at visual cortex

RGB Images and Gray Scale Images

> All the colours is a combination three channels RGB.
> For transparency, we use a alpha channel called A.

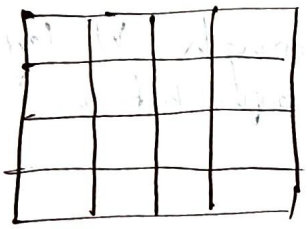


0-255px

4 X 4

RGB

Gray Scale Images

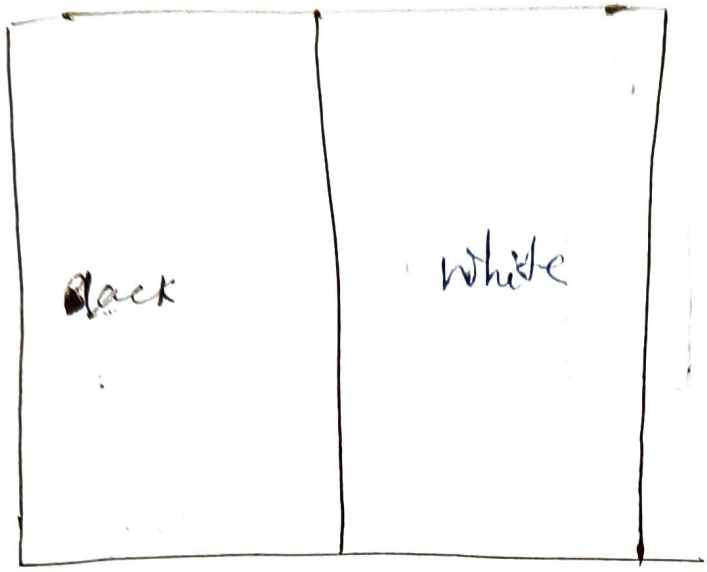


0-255px

255 → White colour
0 → Black colour

Convolutional Operator in CNN

0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255



If this image is pass through VI layer it will detect edges using convolutional network.

Convolutional Operation

→ Perform Min-Max Scaler

Scaling

$\frac{255}{255} = 1$

$\frac{0}{255} = 0$

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

6*6

Filter : Responsible for detecting edges.

F=3

(× multiply)

+1	0	-1
+2	0	-2
+1	0	-1

3*3

→ filter will get placed over images, and multiplication and summation happen for that part.

output

=

0	-4	-4	0
0	-4	-4	0
0	-4	-4	0
0	-4	-4	0

4×4

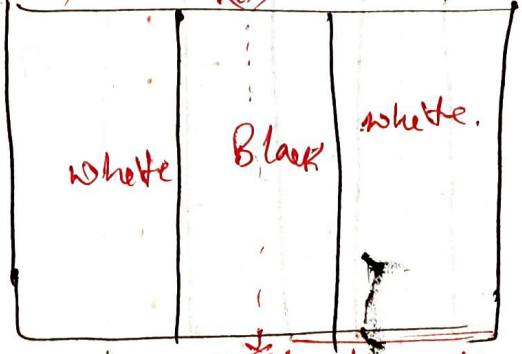
n = image size = 6×6
f = filter size = 3×3
output = n − f + 1 = (6×6) − (3×3) + (1×1) = 4×4

↓ Rescale → Min-Max Scaler (Biggest number is 1 and smallest number is 0)

here
Max = 0
Min = −4

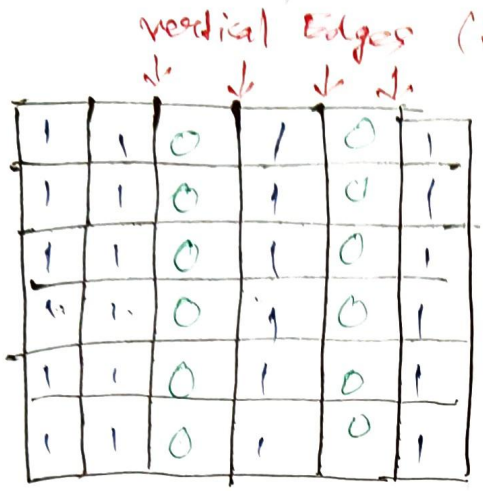
255	0	0	255
255	0	0	255
255	0	0	255
255	0	0	255

output



middle line is edge. → vertical edges filter

⇒ filter is responsible for getting vertical edges.



⇒ For horizontal edges, there need be another filter.

⇒ Like we initialize weights as in ANN, similarly we have to initialize filter.

⇒ For differe images, there will be different filter to ~~at~~ capture edges.

Horizontal edge feature

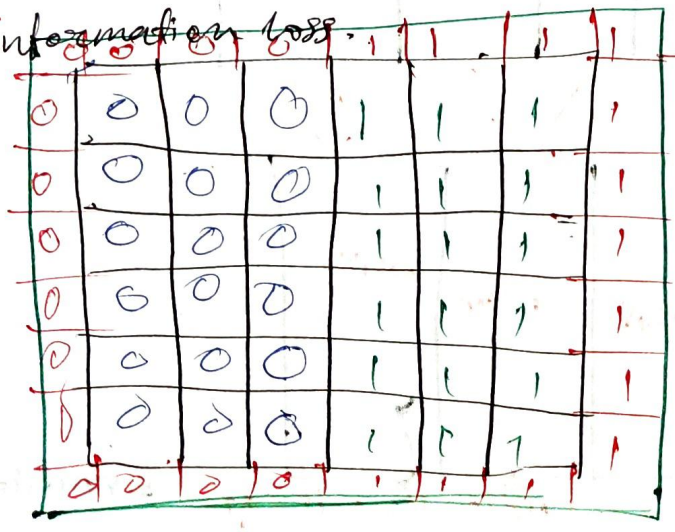
1	2	1
0	0	0
-1	-2	-1

→ fixed filter
→ In Reality we have to initialize numbers.

⇒ We are losing information because after apply filter (3×3) 6×6 images become 4×4. To avoid this we use Padding.

Padding

→ To avoid ~~information~~ loss.



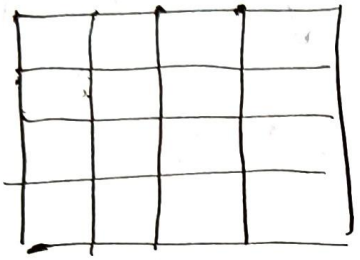
Earlier 6×6
After Padding 8×8

Padding Two Techniques
(i) Add zeros in padding
(ii) Add nearest value in padding

Filter

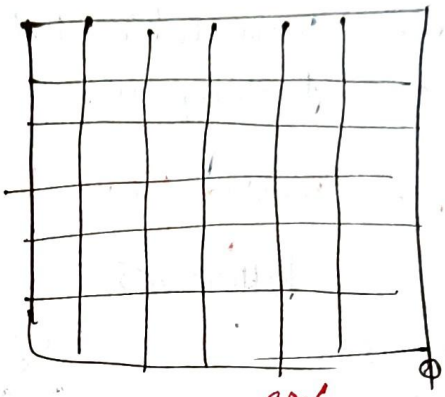
3×3

output



4×4

After Padding



6×6

$\{o/p = n - f + 1\}$

$n - f + 2p + 1 = 6$

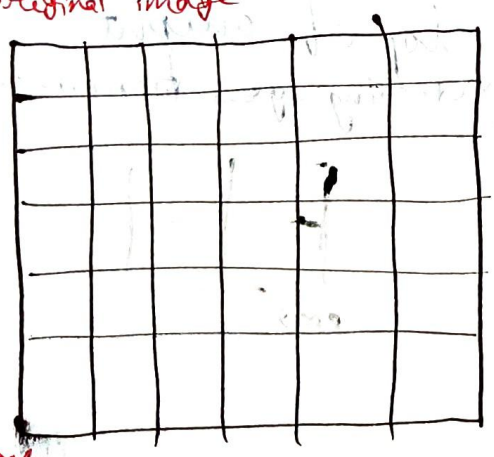
> output should be 6

$6 - 3 + 2 \times p + 1 = 6$

$\boxed{P = 1}$ Padding (Adding another boundary)

Convolution Operation + Relu activation function

original image

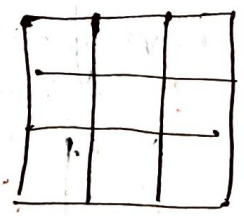


6×6

1st operation

×

filter



3×3

output

0	-4	-4	0
0	-4	-4	0
0	-4	-4	0
0	-4	-4	0

4×4

f_2

f_3 = 

4×4

f_n = 

4×4

convolutional layer

> After applying convolution operation, we apply Relu activation function.

> We have to update filter. To update filter, Relu has been used.

> Convolution network should be able to learn what filter needs to be applied to predict image.

> After convolutional layer, max pooling, min pooling and mean pooling can be applied.

Max Pooling, Mean Pooling and Mean Pooling

filter = 5×5

MNIST

> check image on google.

(28×28×1) → (1 channel)

Black & White Image

→ n| channels

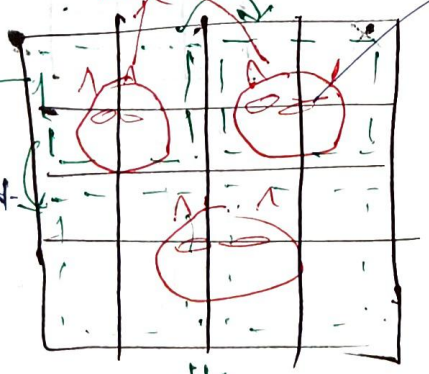
n - f + 1

28 - 5 + 1 = 24

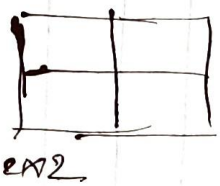
(24×24×1)

> We apply max pooling on top of output.

> Detecting eyes from many faces

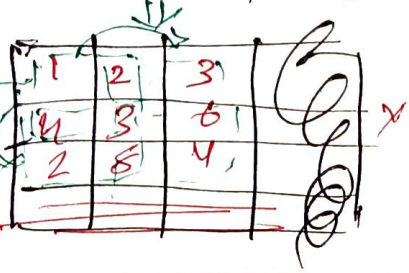


Pooling → Captures the maximum information during jump



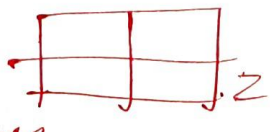
4×4

Pooling



3×3

Max Pooling



2×2
stride = 2

After max pooling

4	6
8	4

> Max pooling is used for location invariance

> In case of min pooling, minimum value will be taken.

> In case of mean pooling, average value will be taken.

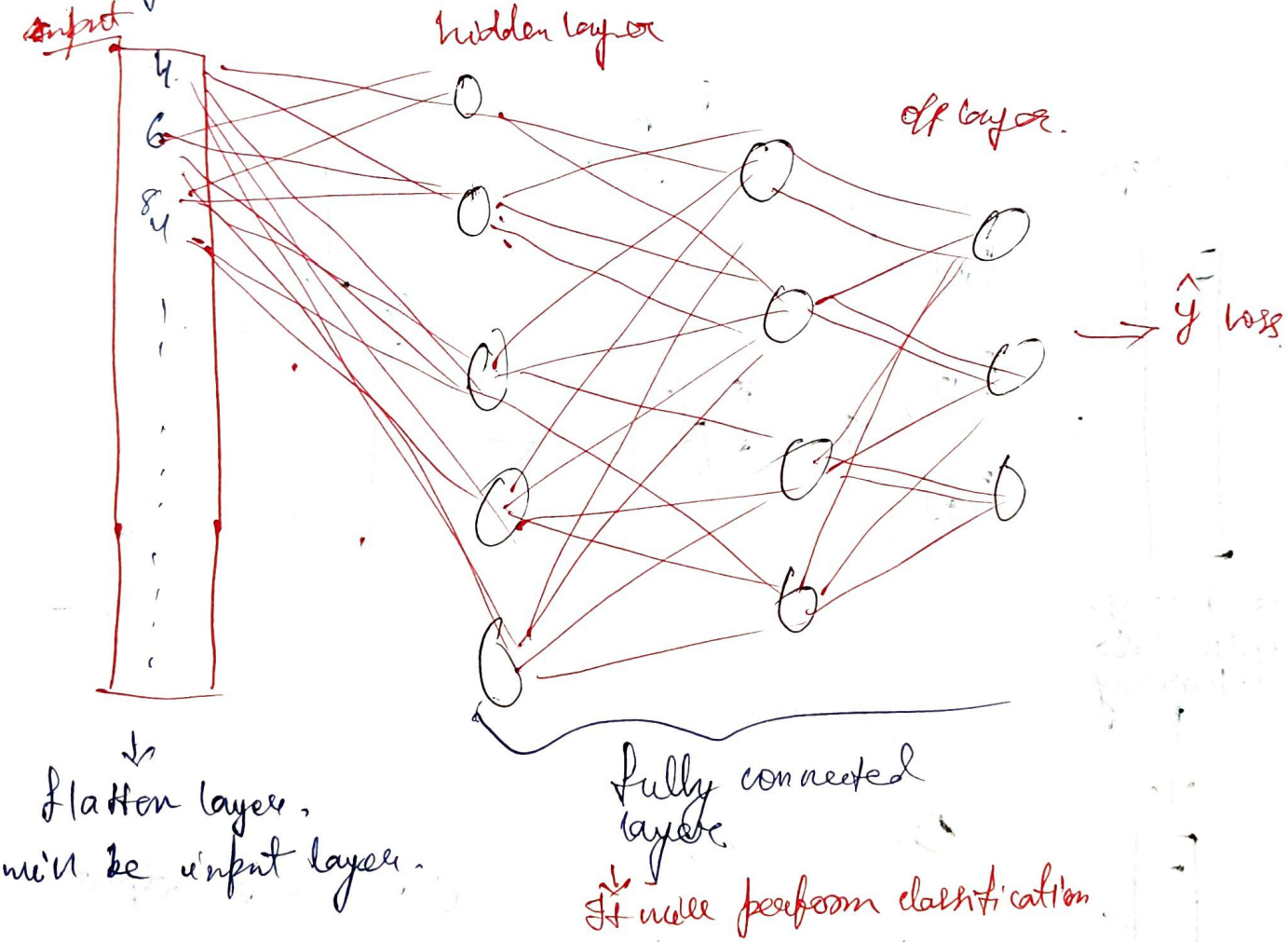
> Stride = 2, because information will be passed if we use stride = 1 then information will not be passed.

Max and Mean Pooling application

> Smoothing, Video noise removal, Photo smoothing.

Flattened

– Elongate the data.



Flatten layer, will be input layer.

fully connected layer

↓

It will perform classification