Chapter-60

Convolutional Neural Nets

* Convolution Neural Nets will be useful in Visual tasks (object Recognition)

Ex- given a Image and asking is there any car?., tiger? etc.,

* Some neurons in the visual cortex that fire when presented lines at specific angle.

V_1: primary visual cortex → edges (which detects)

lly we have V_2, V_3.

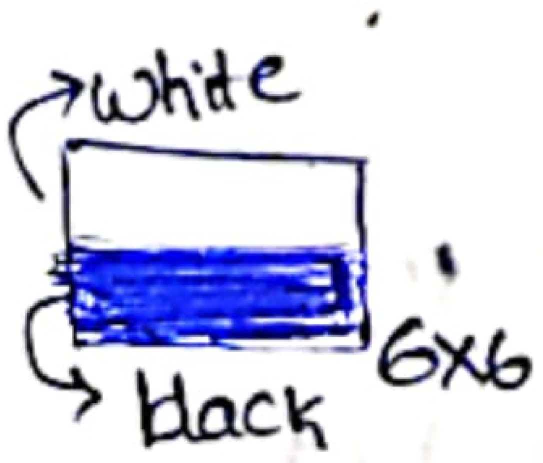
60.2 Convolution : Edge Detection on Images:-

* Edge detection is V_1 (what primary visual cortex) will do

① lets assume we have an Image
(Consider a grayscale Image (0 to 255) or (0 to 1)

So we do as follows:

① We do Convolution between input image and Sobel edge detector / Kernel / filter / mask / Operator.



2D

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

6X6

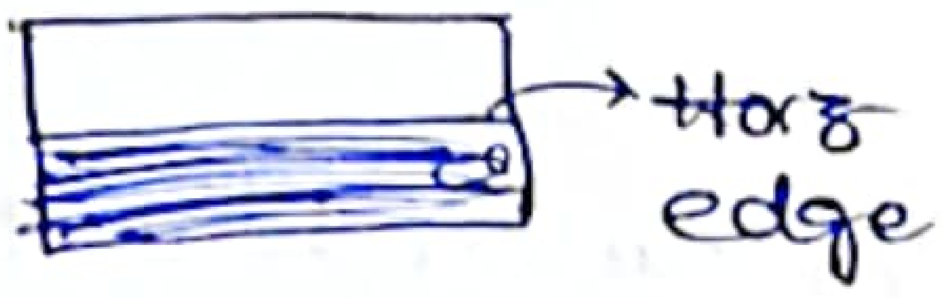
$*$

2D

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

3X3

So what Convolution will do is



$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix}}_{\substack{6\times 6 \\ n\times n}} * \underbrace{\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}}_{\substack{3\times 3 \\ k\times k}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ -1020 & -1020 & -1020 & -1020 \\ -1020 & -1020 & -1020 & -1020 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\substack{4\times 4 \\ (n-k+1 \times n-k+1)}}$$

$$\begin{bmatrix} 0(+1) & 0(+2) & 0(+1) \\ 0(0) & 0(0) & 0(0) \\ 0(-1) & 0(-2) & 0(-1) \end{bmatrix}$$

$0+0+0+0+0+0+0+0+0=0$

$$\begin{bmatrix} 0(+1) & 0(+2) & 0(+1) \\ 0(0) & 0(0) & 0(0) \\ 255(-1) & 255(-2) & 255(-1) \end{bmatrix}$$

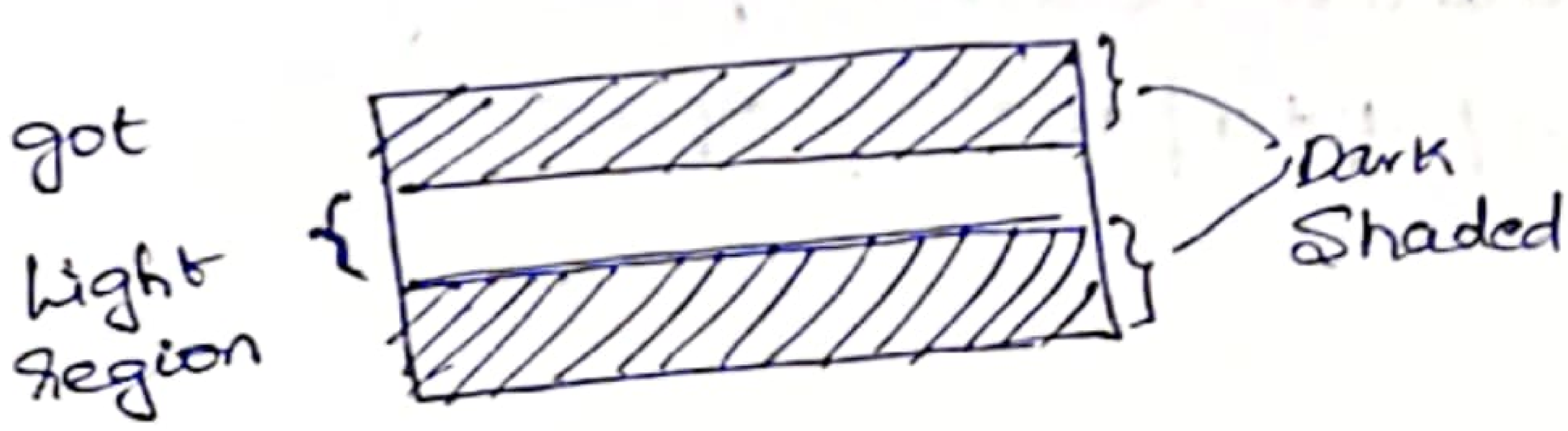
$0+0+0+0+0+0+(-255)+(-510)+(-255)$
$= -1020$

$$\begin{bmatrix} 255(+1) & 255(+2) & 255(+1) \\ 0(255) & 0(255) & 0(255) \\ -1(255) & -2(255) & -1(255) \end{bmatrix}$$

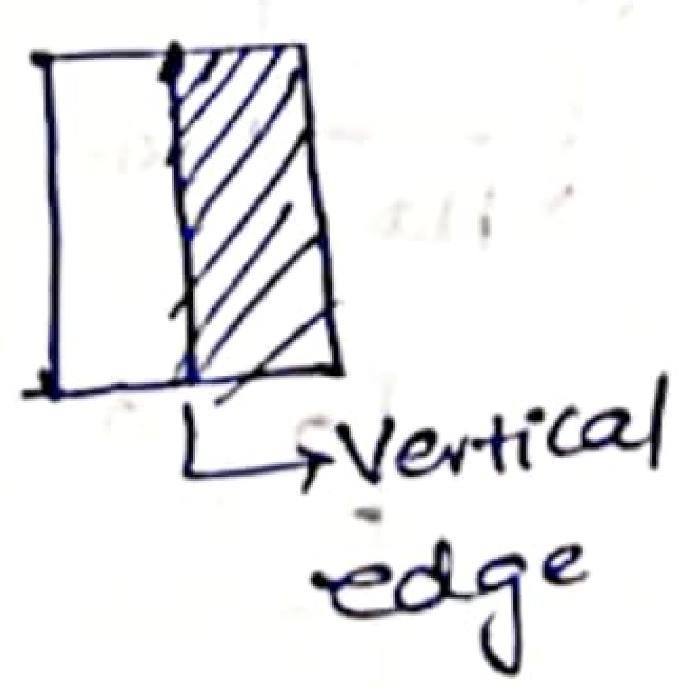
$\cancel{255}+\cancel{510}+\cancel{255}+0+0+0+$
$-\cancel{255}+(-\cancel{510})+(-\cancel{255})$
$=0$

$\square_{n\times n} * \square_{k\times k} = \square_{(n-k+1,\, n-k+1)}$

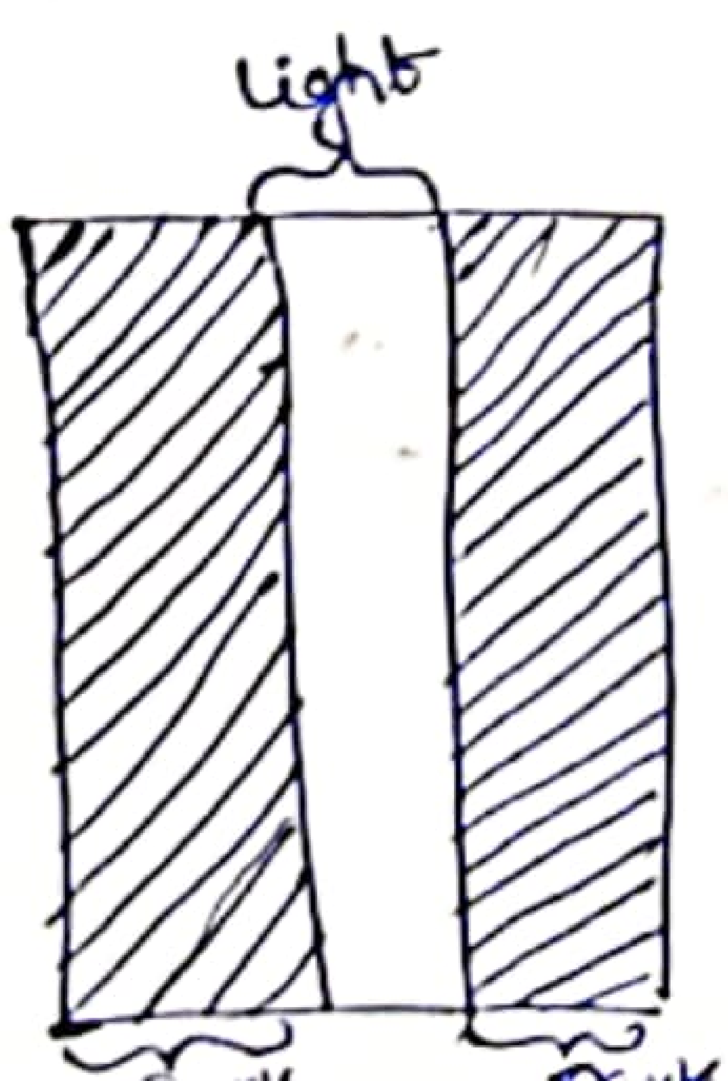
So we got



Similarly if we do for Vertical edge detection



we get



Convolution 6×6 → ...

Ex:- 6×6 → 8×8

60.3 Convolution: padding and Strides:

Padding

Ex:-

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

Horizontal edge 1 →

Horz. edge 2 →

*

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

K×K

6×6 → 8×8

p = 1

and we done here Zero padding.

$$\Rightarrow n \times n \xrightarrow{P=1} (n+2)\times(n+2) \Rightarrow (n+2\times P)\times(n+(2\times P))$$

$$n \times n \xrightarrow[\text{Padd } P]{K\times K} (n-K+2p+1)\times(n-K+2p+1)$$

By padding we Observe some interesting effects on the Operations.

* By padding we can get another edge

② Another type of padding is Same Value padding.

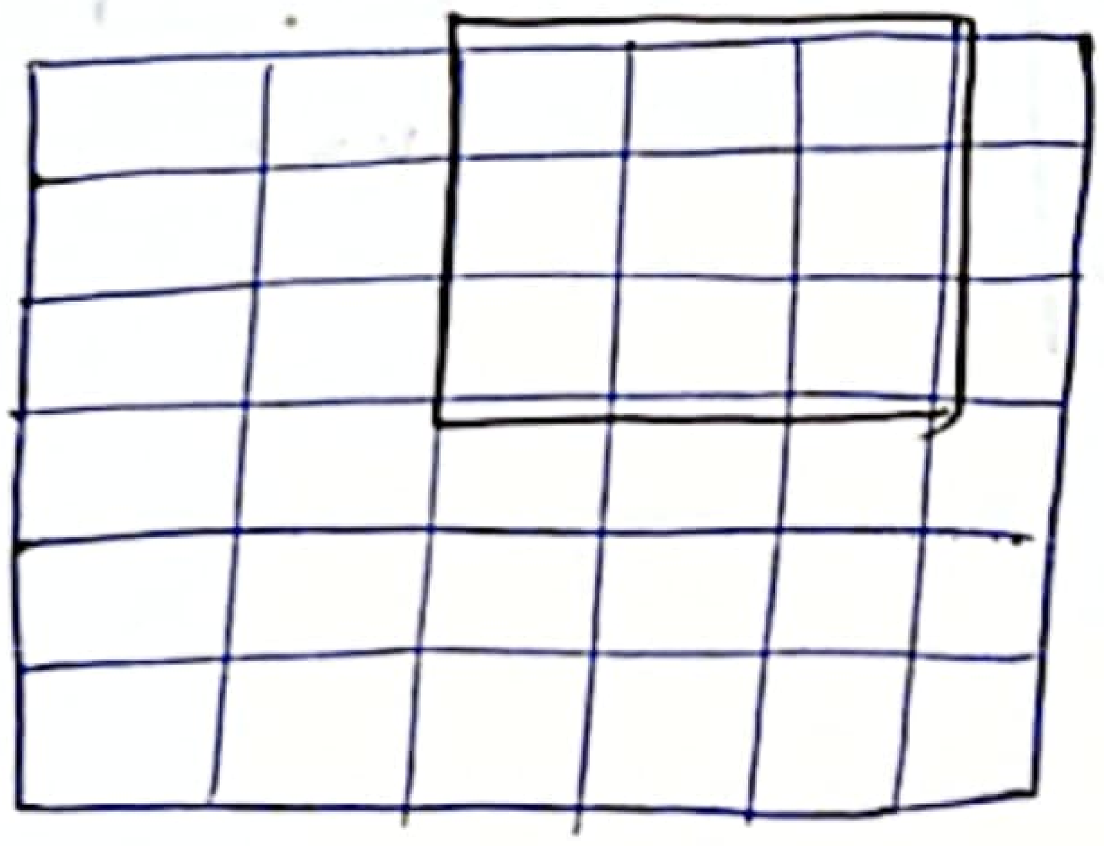
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

But when it comes to here It get confuse

Strides

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

Shifted by 1 = Stride of 1



Shifted by 2 = Stride of 2

$$n \times n \xrightarrow[K=K]{S=S} \left(\frac{n-K}{S}+1\right) \times \left(\frac{n-K}{S}+1\right)$$

Ex: $$6 \times 6 \xrightarrow[K=3]{S=2} \left(\left\lfloor \frac{3}{2} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{3}{2} \right\rfloor + 1\right)$$

$$6 \times 6 \xrightarrow[K=3]{S=2} 2 \times 2 \quad (O/P)$$

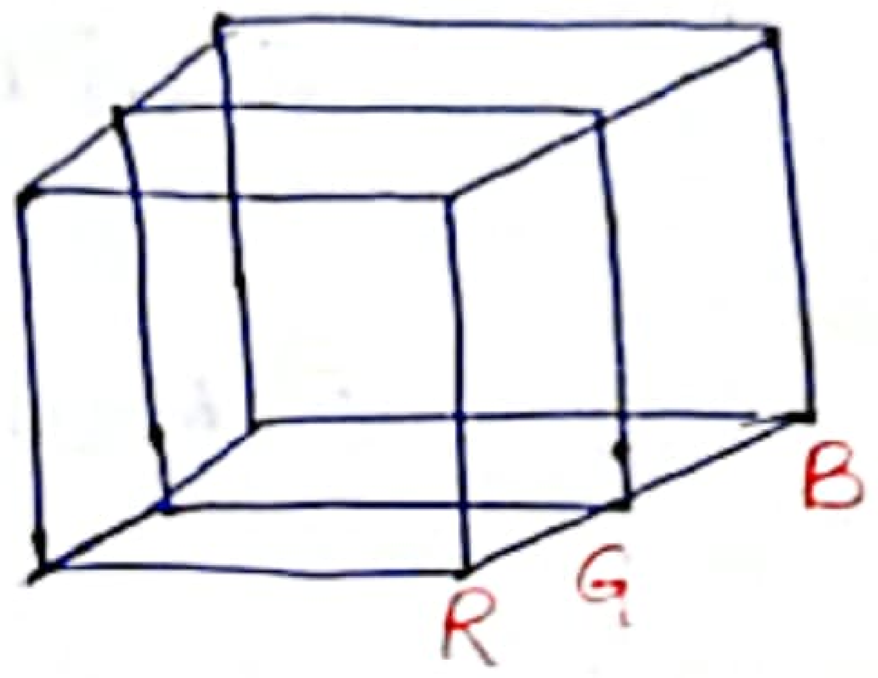
* $$n \times n \xrightarrow[\text{Padding } P]{K \times K} (n-K+2P+1) \times (n-K+2P+1)$$

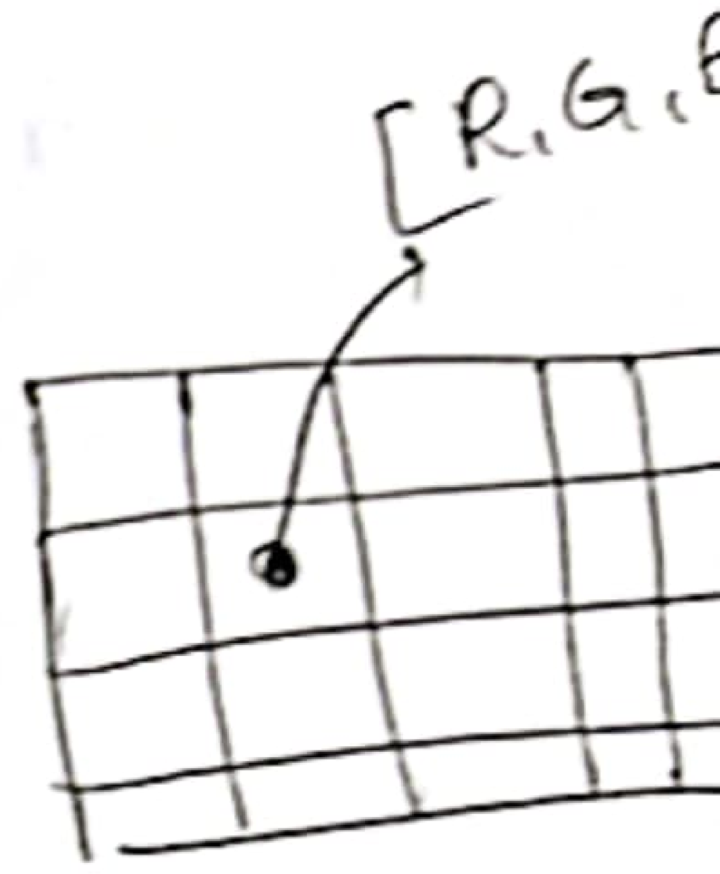
$$n \times n \xrightarrow[P,\, S]{K \times K} \left(\left\lfloor \frac{n-K+2P}{S} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{n-K+2P}{S} \right\rfloor + 1\right)$$

6.4 Convolution Over RGB Images :-

As we know each Image/pixel would be Represented using Three primary colours Red, Green, Blue.

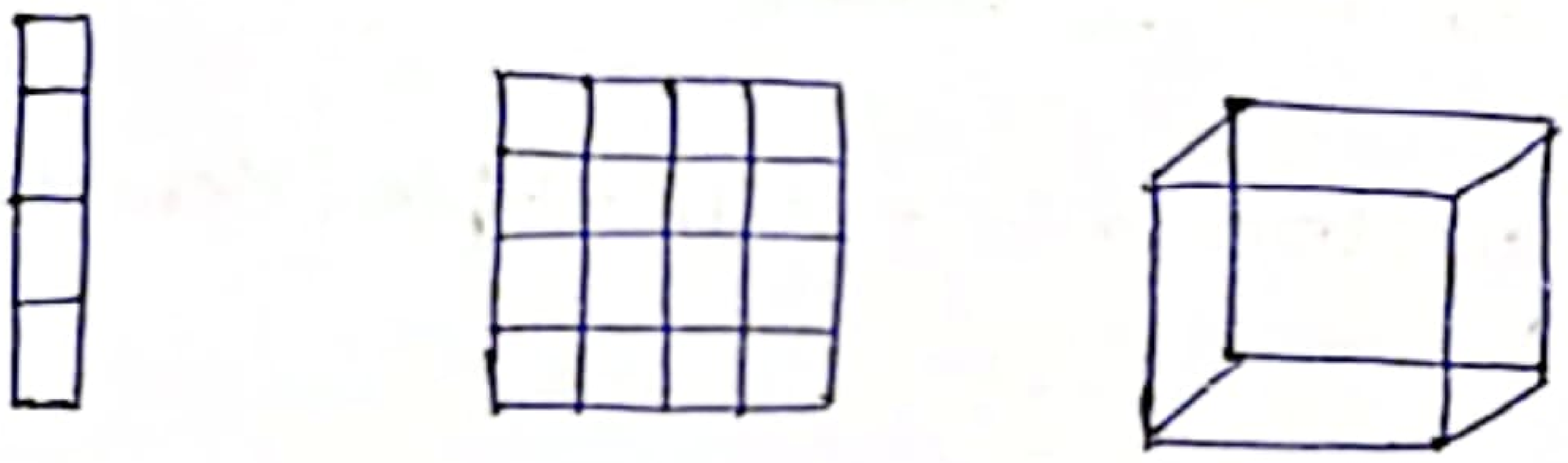
→ To Represent an image R, G, B's will be layered as a matrix to form as a Image





each R, G, B will be a Channel.

→ each dimension would be Represented as a Tensor

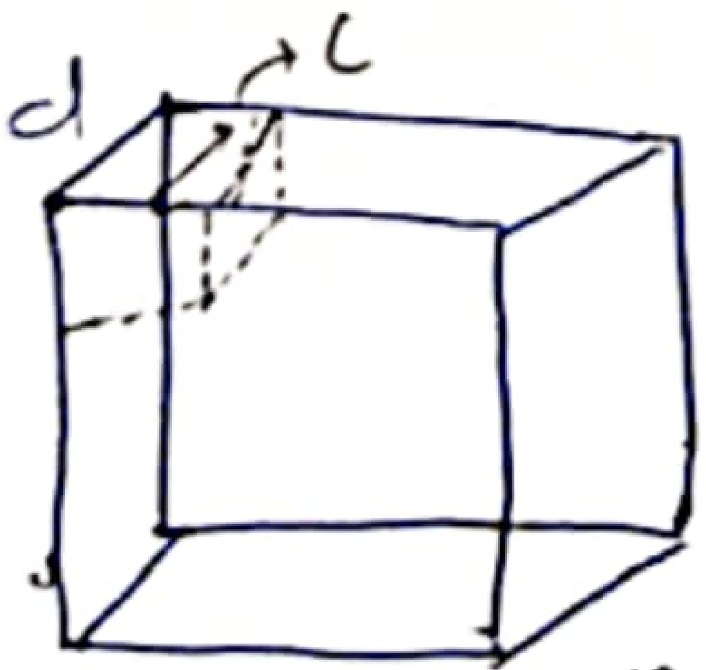


1D-Tensor 2D-Tensor 3D-Tensor.

each 3D tensor → $n \times m \times c$ → Channels

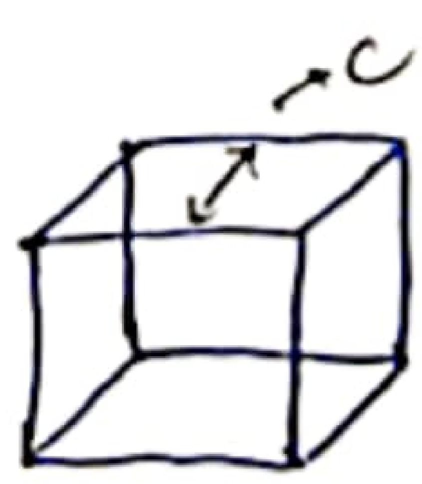
+ as we seen Convolution in 2D matrix → Componentwise mul & add.

↳ when we do convolution over 3D tensors. (Colour Images)



conv

$*$



→ $(n-K+1) \times (n-K+1) \times 1$ (1 Chan

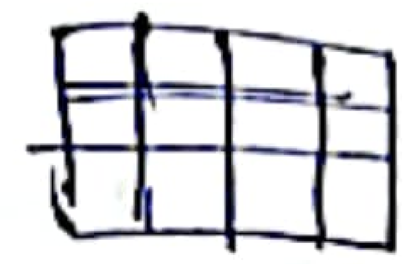
$K \times K \times c$

$n \times m \times c$

$l \times w \times d$

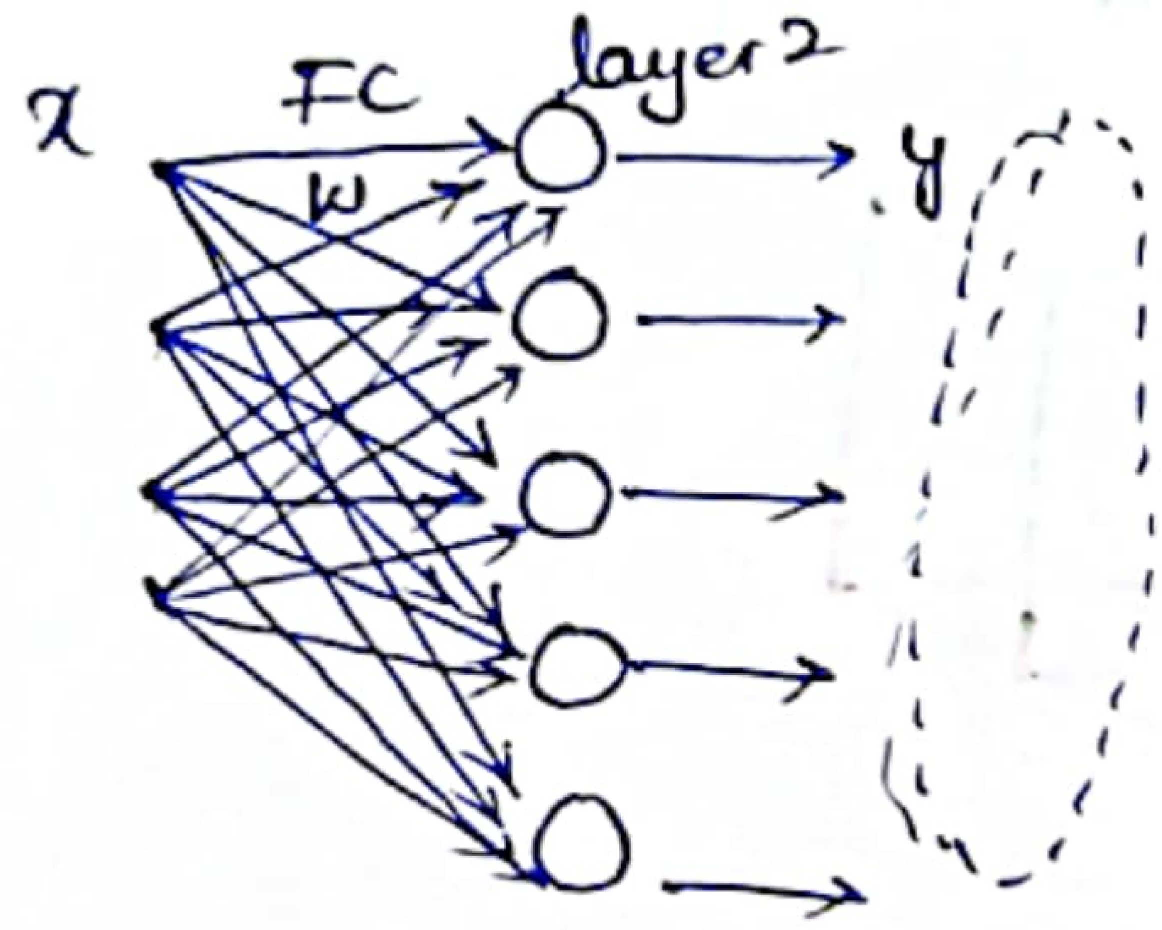
These c's should be same.

c → no of chanels
— R, G, B (depth)

O/p we get a two dimensional Image 

60.5 Convolution layer

* As we seen MLP layer. where we have inputs/ previous layer O/p's.



So as we are doing

① $w^T x = z$ (we are finding z)

② $ReLU(z) = y$ (applying ReLU on top of it).

* Convolution layers are biologically inspired.

as we seen in edge detectors

1 edge detector → 1 Kernel

lly for multiple edge detectors → multiple Kernels.

* As we are using sobel in classical Image processing while in CNN we learn Kernel matrix as of back propagation.

As in MLP ! ~~we~~ learn weights w's → $w \cdot x$.

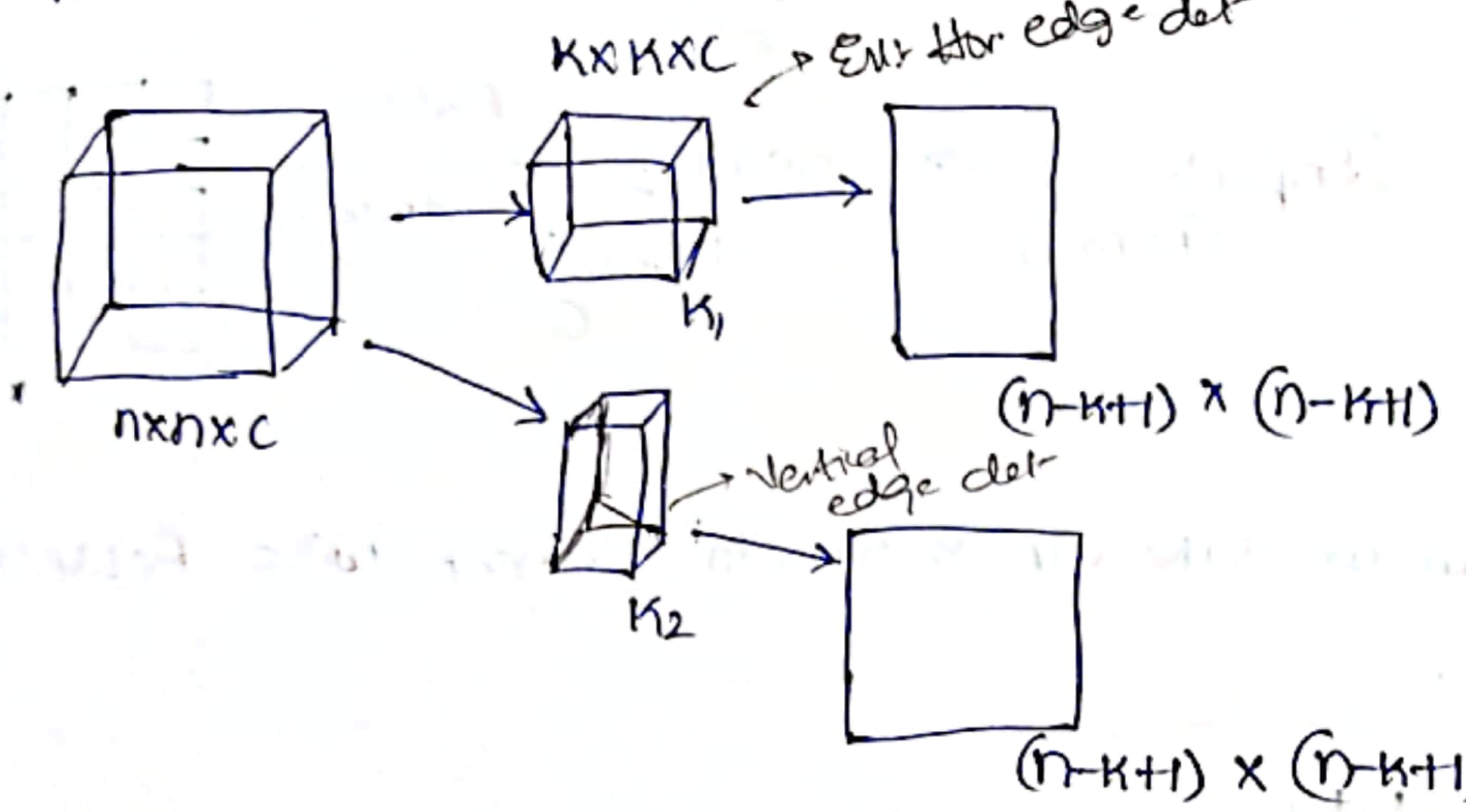
where in CNN! we learn Kernel matrices → $I/p * w_{k \times k}$ Kernel

---

Convolution → 2D matrix → Component wise mul and add

conv → 3D tensor → " " "

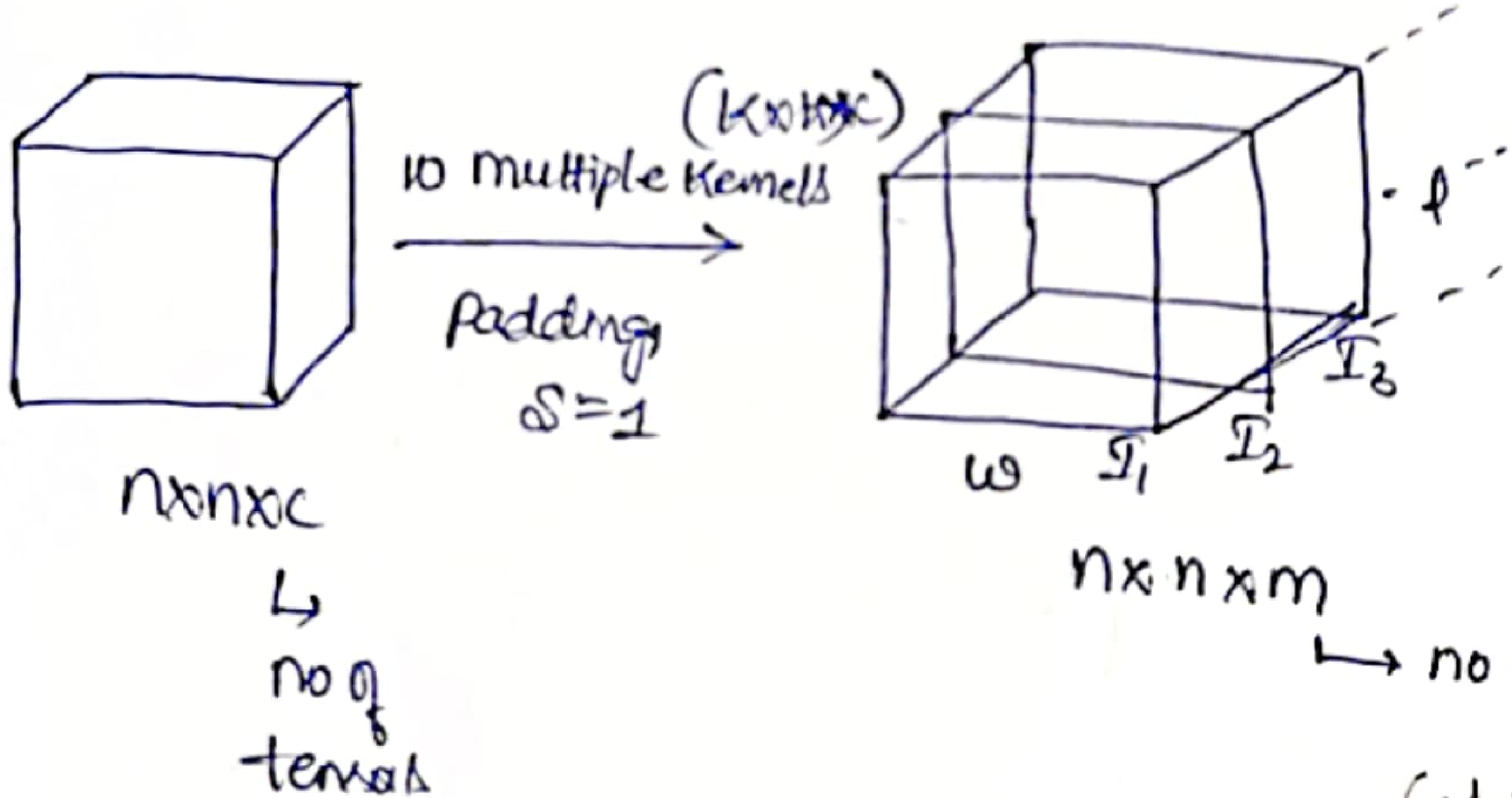
* for ~~multi~~ an input we may get multiple Kernels



so for each convolution layer

for an input n×n×c

we get Kernel ~~map~~ o/p of n×n×m (where m is no of Kernels)



for an 3D, RGB Image
C=3

(which is hyperparameter)
what padding, S are also hyperparameter

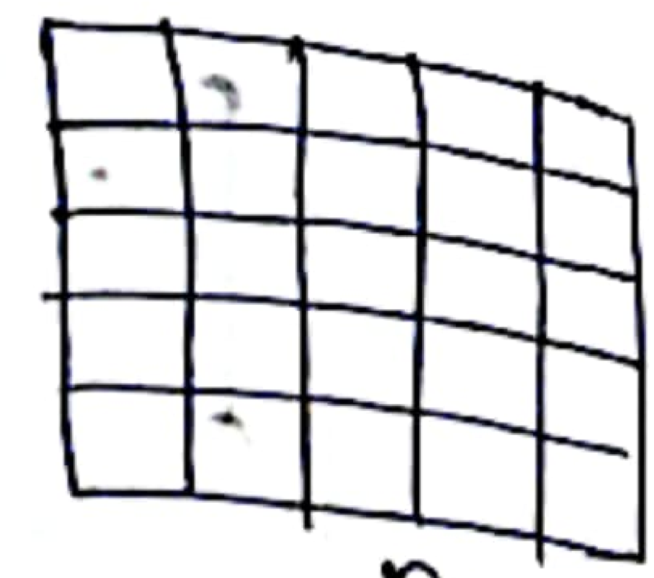
so l, w (length of o/p, & width of o/p) ⟶ Padding, Strides & Kernel.

d ⟶ (m) # of Kernels.

so for one layer of Convolution layer we have 3 hyper-parameters k, # Kernels, padding, & Strides.

The O/P is generated as follows:

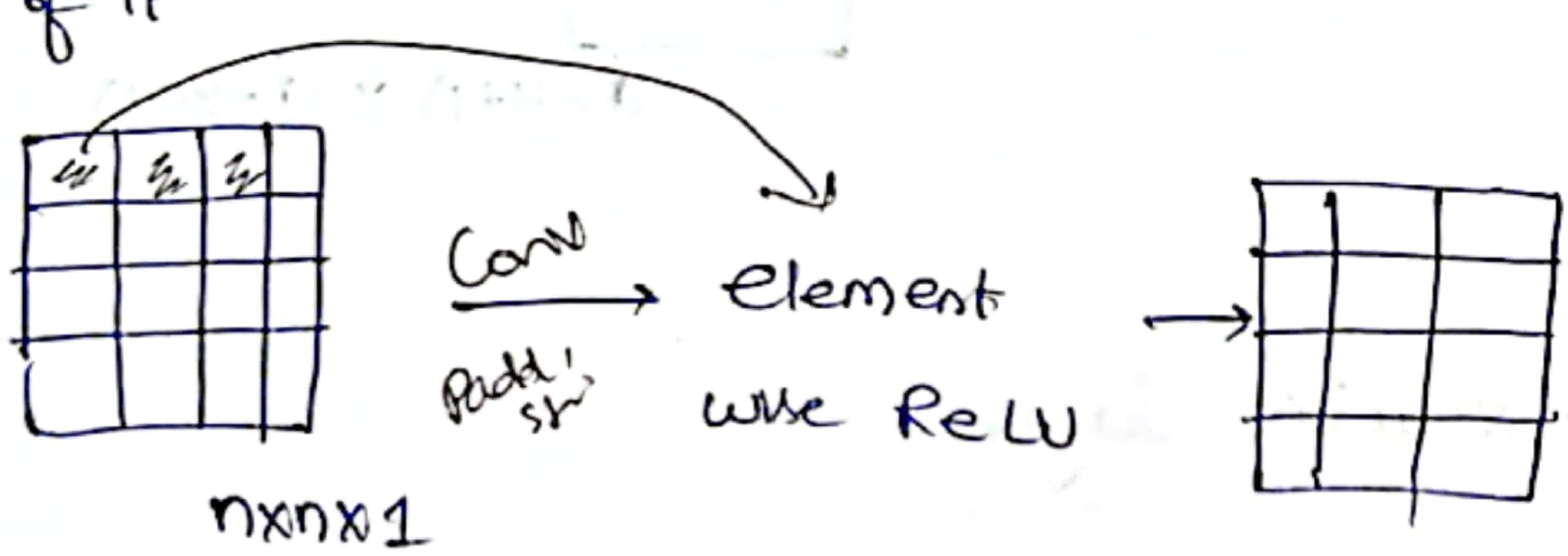
<u>Step 1:-</u> Input $(n \times n \times c)$ $*$ $\underbrace{\text{Kernel}_1}_{}$ $(K \times K \times C)$ $\xrightarrow[\text{\& strides = 1}]{\text{padding}}$ (m kernels)



$n \times n \times 1$

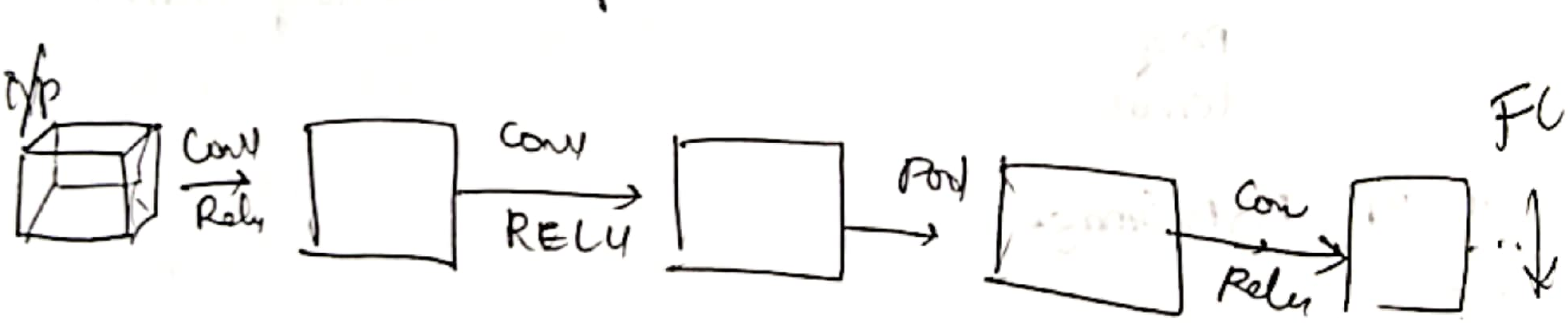
$(n \times n \times m)$

<u>Step 2:-</u> now we take O/P & we do element wise ReLU on top of it



$n \times n \times c \longleftarrow n \times n \times 1$

These size may not be same it varies depends on padding & striding



we do apply multiple layers of Conv layers

606 max pooling :-

* Max pooling is one of the layer which we add in our model.
* Max pooling makes the model location-Invariant, Scale-Invariant, Rotational-Invariant

To understand max pooling

Imagine we have 4x4 Image

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

$\xrightarrow[S=2]{\text{max pooling } K=2}$

6	8
3	4

Now if we do max pooling with Kernel/filter = 2 & strides=2

ly if we do with $\xrightarrow[\text{max pooling}]{K=2 \ \& \ S=1}$

6	7	8
6	6	8
3	3	4

If we apply another level of max pooling

6	8
3	4

$\xrightarrow[2\times 2]{\text{max pooling}}$

8

1x1

and this max value (8) will be the max value among all the values of Convolution matrix input.

This is why we can say that it is location Invariance.

So we can easily detect Image ~~if we even~~ irrespective of its location as the Image seeks max activation value.

max-pooling is extremely used in convnets.

→ we can also do Avg/mean pooling where rather than max value it considers Avg/mean value.

60.7 CNN Training: Optimization

→ As we seen in MLP:

when ever we are applying / using backpropagation which is the key idea in MLP for all algorithms like SGD, Adagrad, Adam. etc all are differentiable.

that means

$$x \longrightarrow \hat{y} \rightarrow \mathcal{L}(y, \hat{y}) \Rightarrow \frac{\partial \mathcal{L}}{\partial w}, \quad \nabla_w \mathcal{L}.$$

lly In case of

Conv layer ⟶ we do convolution ; ReLU

(on matrices)

→ which do

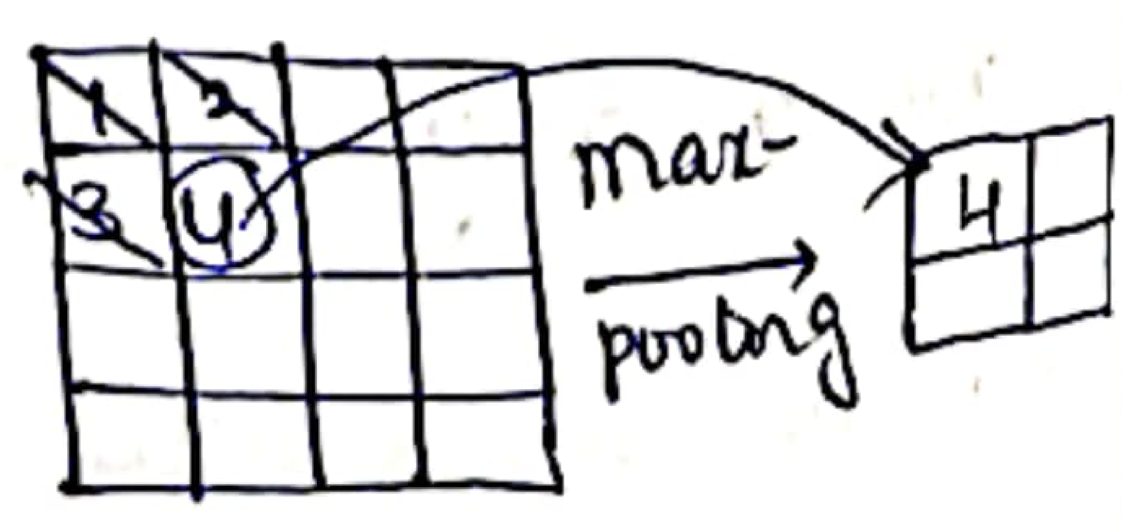
Elementwise + addition → diff
MUL to

lly for max pooling

we do differentiation as given by

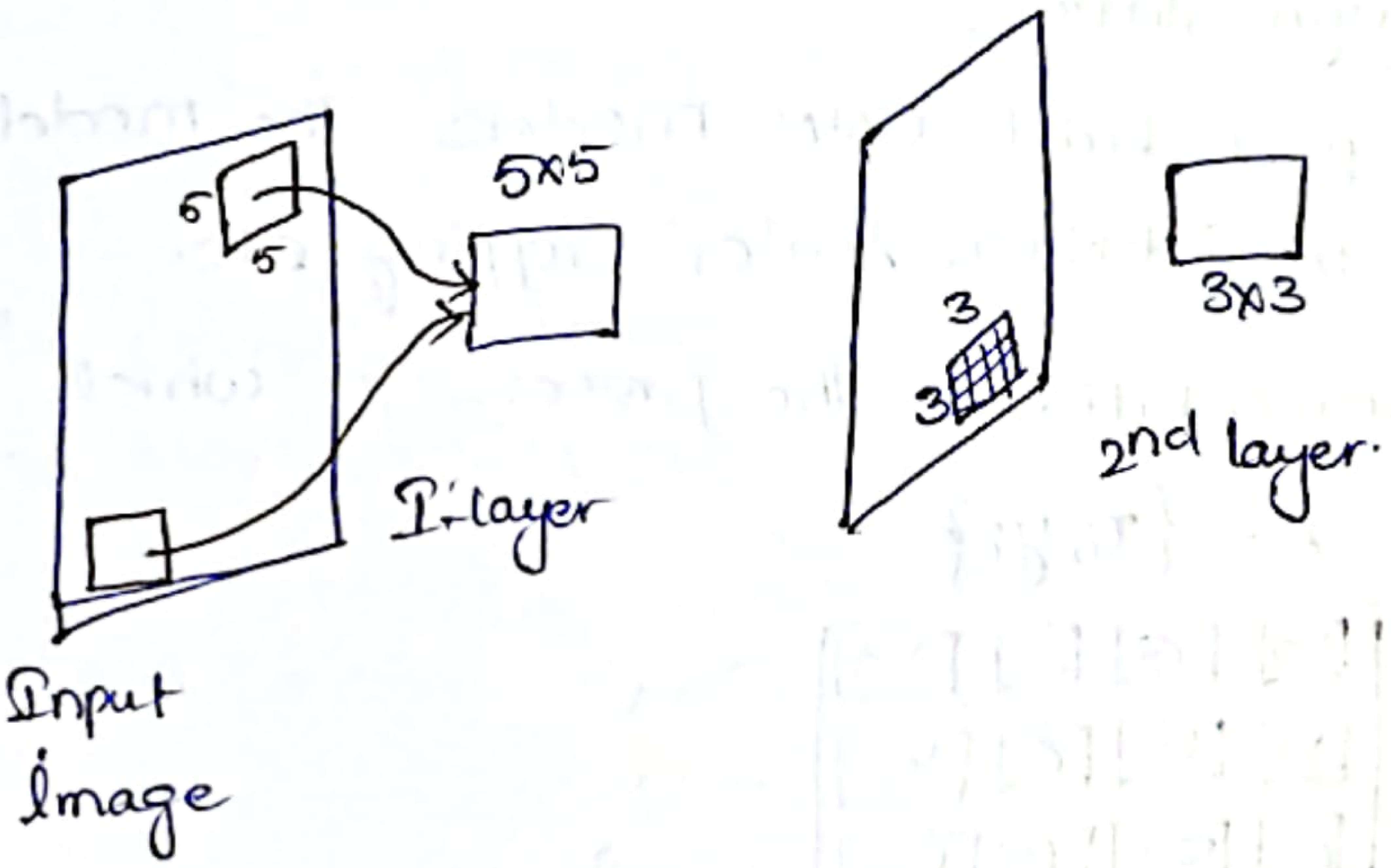
max value $\xrightarrow{\text{diff}}$ 1

non-max value ⟶ 0

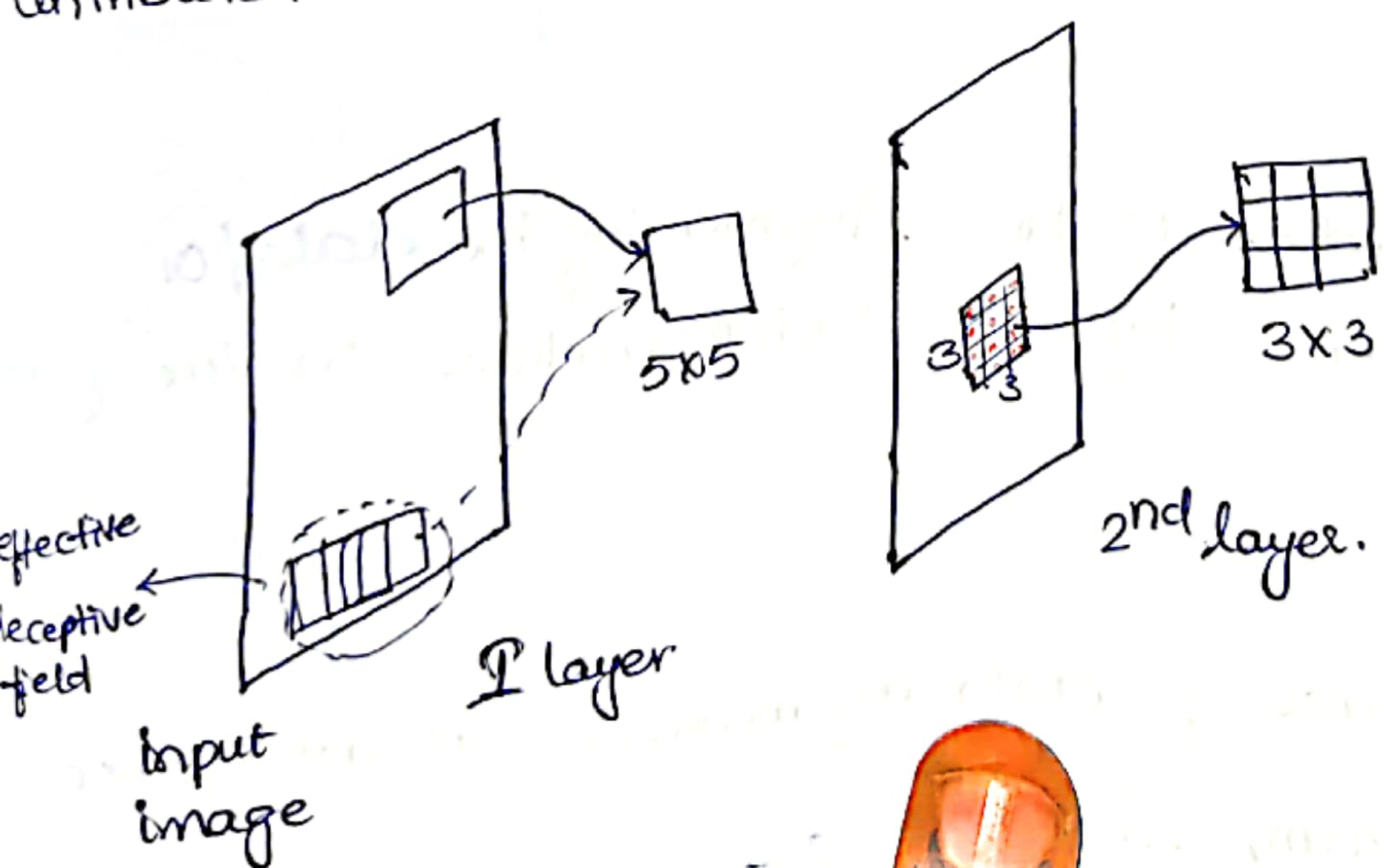


608 Receptive fields and Effective Receptive fields :-

Receptive field is the Region of Image of which the convolution of Kernel at a given time t. is been Convoluted with



where as effective Receptive field would be when we have 2, 3 --- layers (multi layer CNN) the effective Region of the Original Image, whose pixels directly or indirectly Contributed to get O/p Kernel



60.9 Example CNN: LeNet [1998]:-

← LeNet printout →

60.10 ImageNet Dataset

← Wikipedia →

60.11 Data Augmentation:-

while we apply or Build CNN models The model needs to be Robust to Rotation, Scale, Cropping etc.

So Data Augmentation is The process in which for a given Data $D = \{x_i, y_i\}$.

So tries to do all possible Operations like flip, hor-shift, Vertical shift, Rotation, zoom, shear etc., such that the model become Robust and easy to detect the image.

→ So the core idea is to Augmenting The data / or adding the data by doing such Operations on the given image.

* So the main use of data augmentation is invariance to rotation, zoom, shift etc.,

and we can also transform a small datasets → larger datasets.