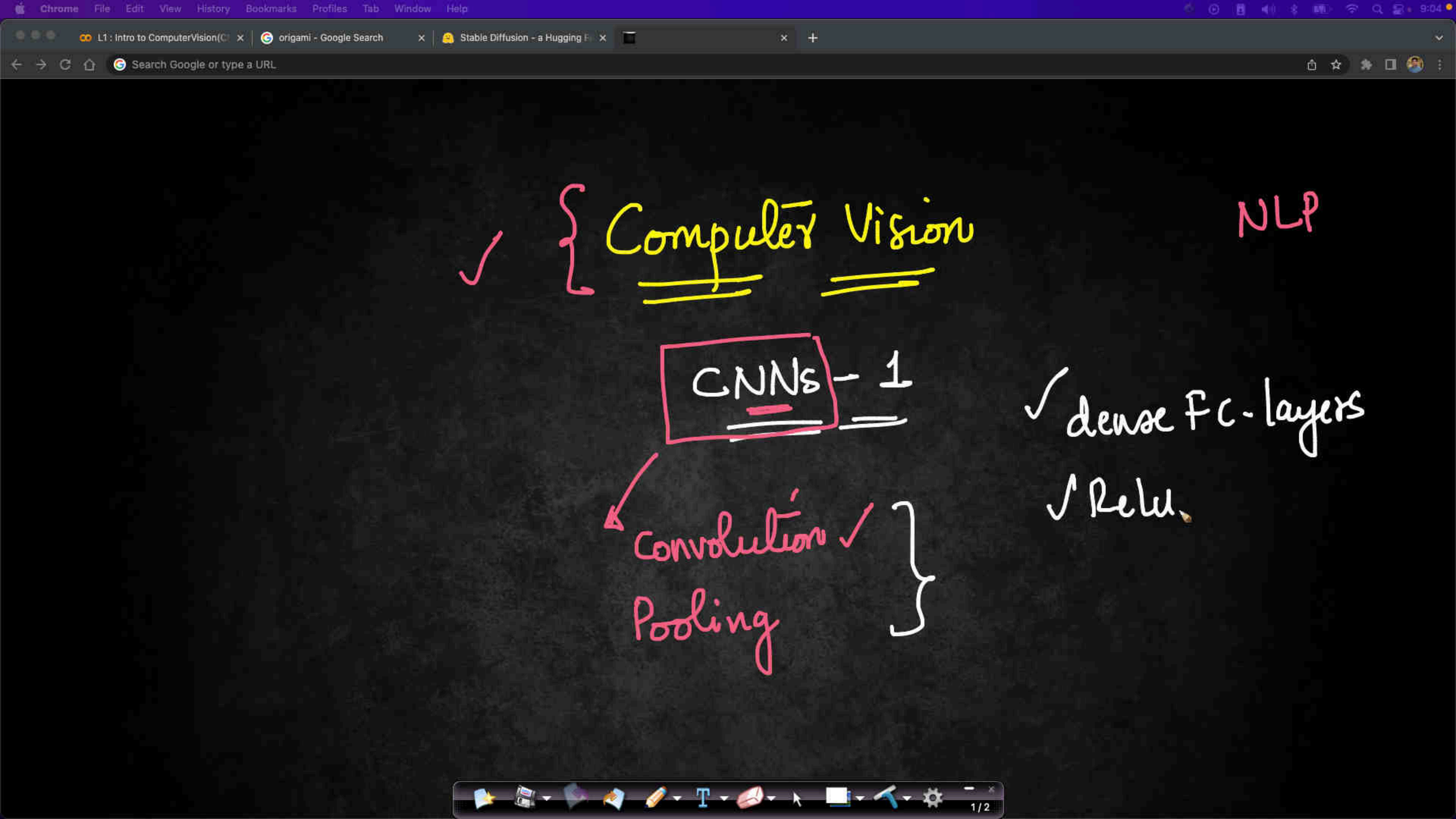
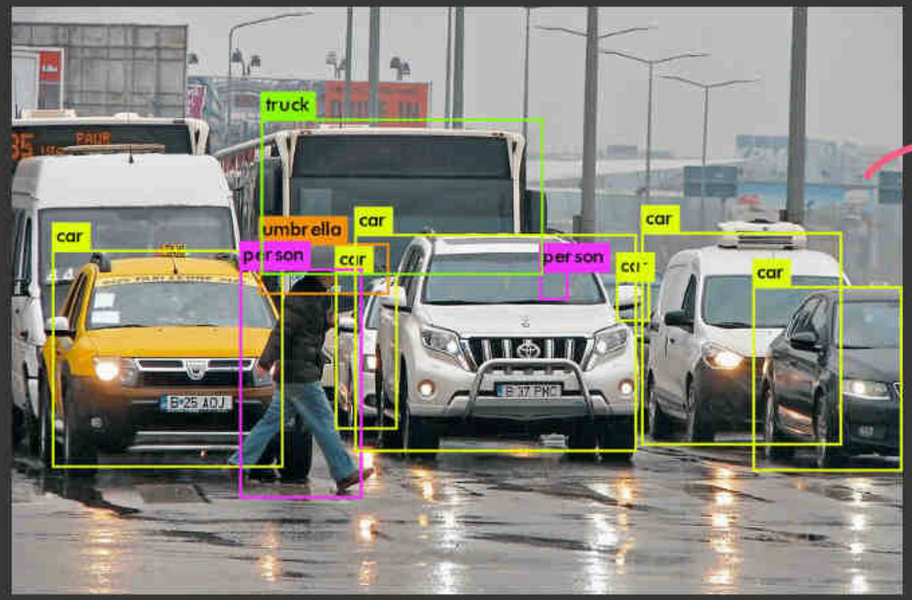

Chrome File Edit View History Bookmarks Profiles Tab Window Help
L1 : Intro to ComputerVision(C
origami - Google Search
Stable Diffusion - a Hugging F
Search Google or type a URL
Computer Vision
NLP
CNNs - 1
dense Fc-layers
Relu
Convolution
Pooling
1 / 2

- Detect all the **Objects** in the Image -



bounding box

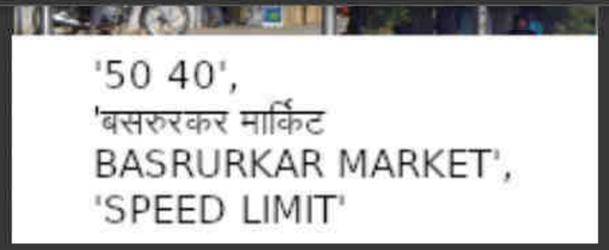
- Extract **Text** from the Image -

- Extract **Text** from the Image -

'50 40',
'बसरुरकर मार्किट
BASRURKAR MARKET',
'SPEED LIMIT'

CNN + RNN
(NLP)

- Write one-line describing the Image(known as **Image Captioning**) -

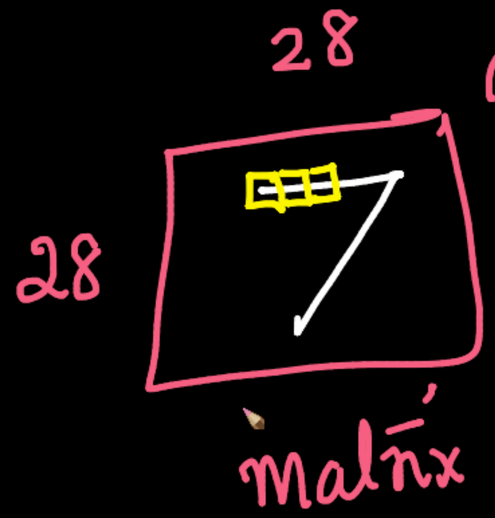


- Write one-line describing the Image(known as **Image Captioning**) -

NLP
CNN

These problems might be easy for humans as we are trained with 500,000,000 years of image data, but What about Computers?

Before even answering this question, let's investigate



grayscale Image

{ 0 - 255
{ B W

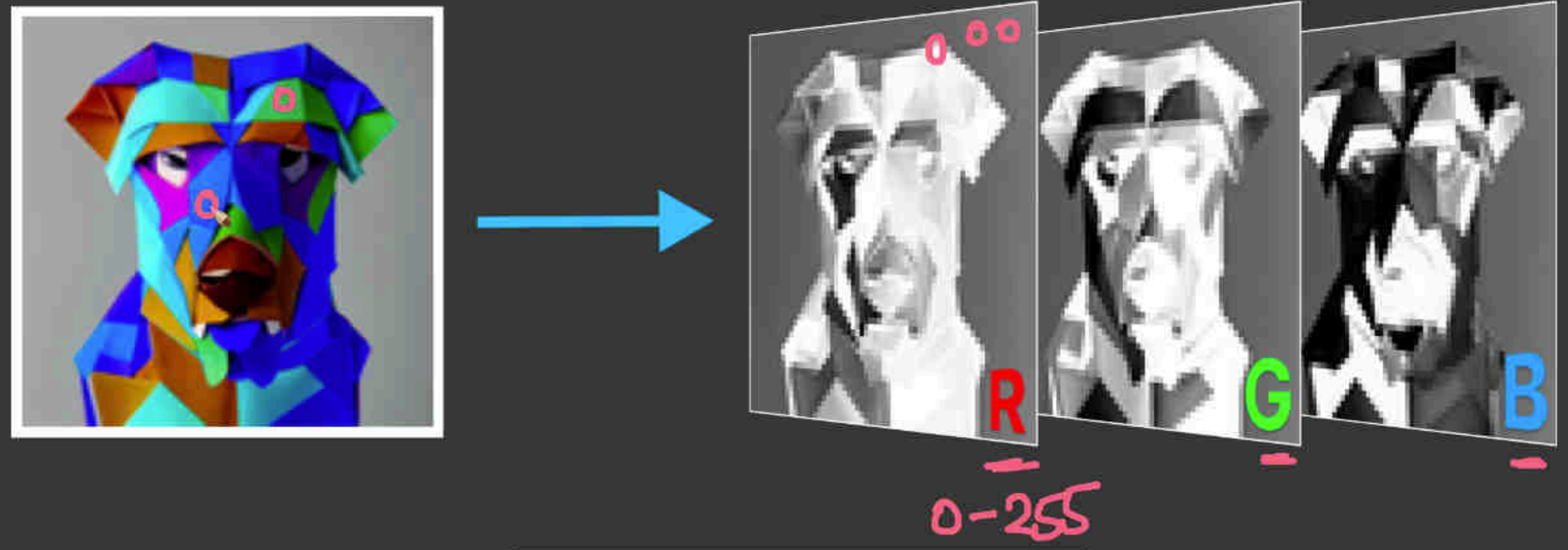
8 bit numbers

{ 0 1 - - -
{ - - - - -

{ 0 - 255
{ W B

What a Computer actually sees?

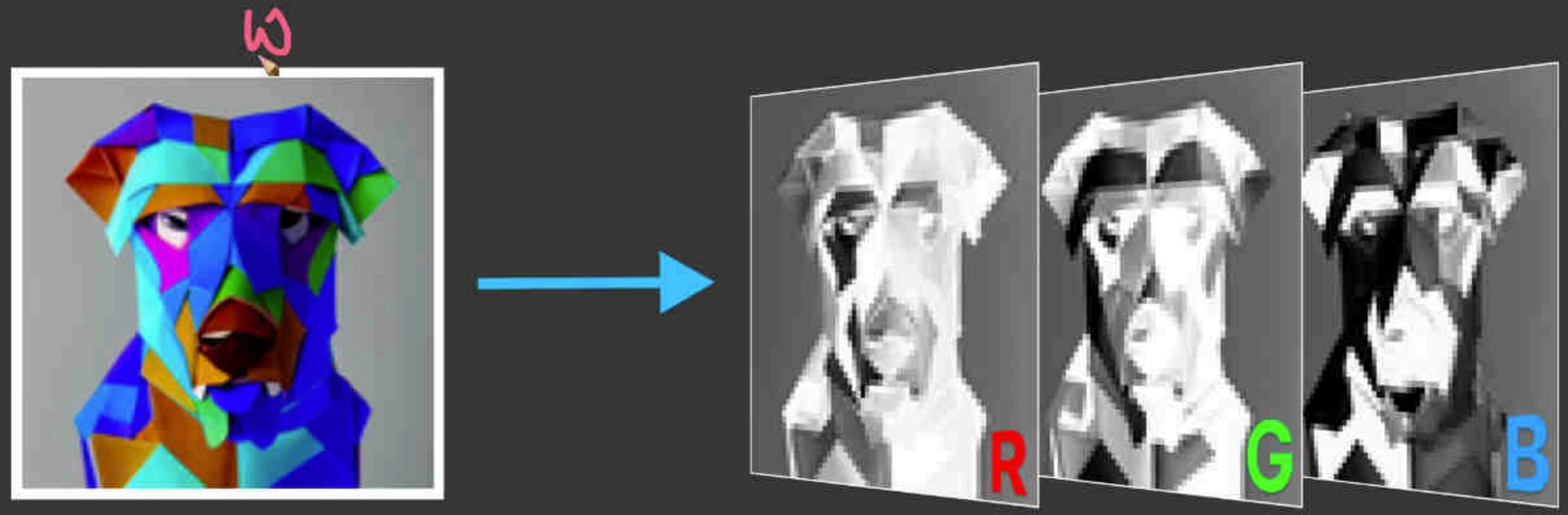
- A **Digital Image** is a matrix of size (H,W,C), comprising of numbers (also known as pixel values) typically ranging from 0-255. Here H,W,C denote Height, Width and No. of Channels in a image.
- For a Grayscale image, # channels is 1, while for Colored Image its 3.



158	176	246	246	251	241	235	242	254	249	244	253	248	255	127	0
159	172	243	247	249	239	240	251	255	185	220	255	249	244	27	4
160	168	239	248	247	250	253	252	246	109	247	250	255	160	4	28
161	164	237	248	248	249	249	255	199	15	234	255	254	97	27	3
162	163	235	250	248	249	246	255	122	0	188	255	195	24	0	4

What a Computer actually sees?

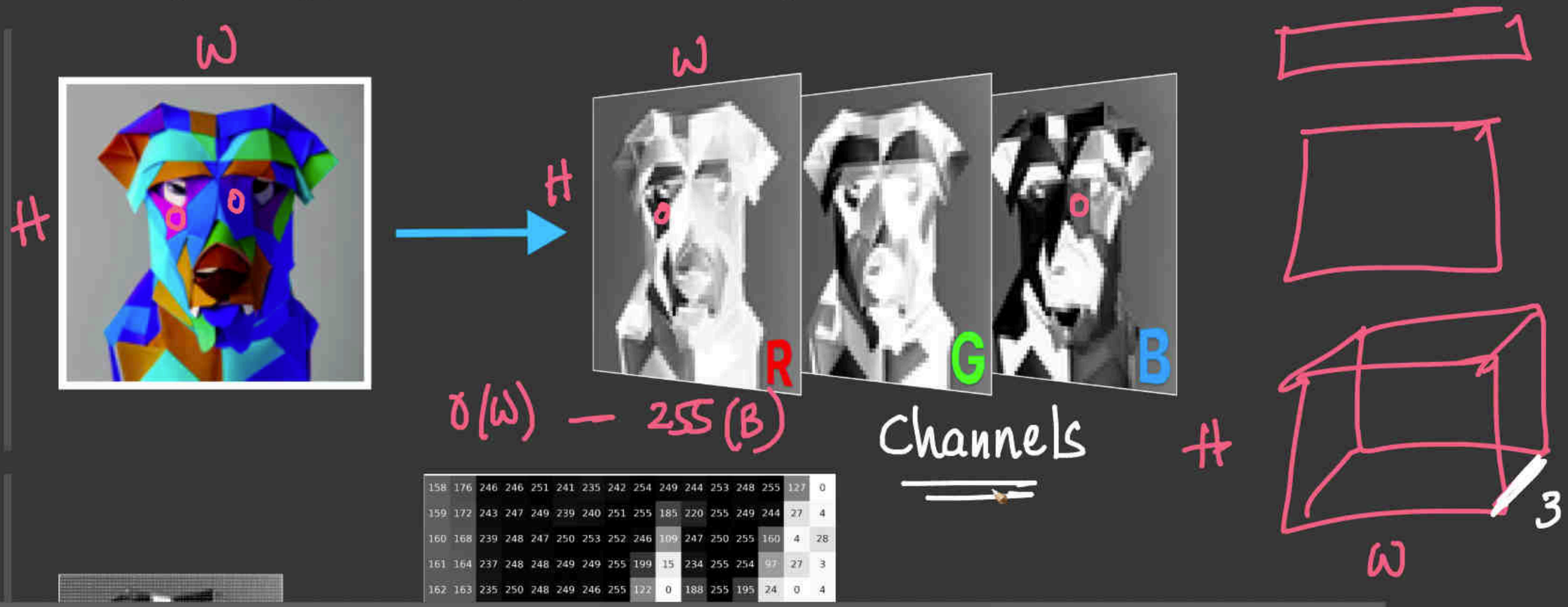
- A **Digital Image** is a matrix of size (H,W,C), comprising of numbers (also known as pixel values) typically ranging from 0-255. Here H,W,C denote Height, Width and No. of Channels in a image.
- For a Grayscale image, # channels is 1, while for Colored Image its 3.



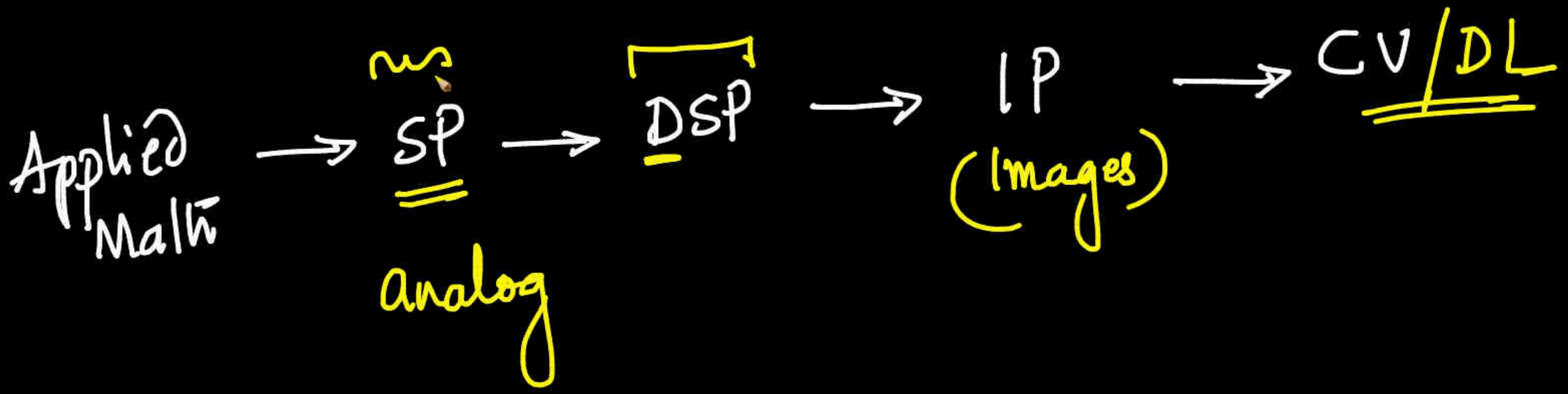
158	176	246	246	251	241	235	242	254	249	244	253	248	255	127	0
159	172	243	247	249	239	240	251	255	185	220	255	249	244	27	4
160	168	239	248	247	250	253	252	246	109	247	250	255	160	4	28
161	164	237	248	248	249	249	255	199	15	234	255	254	97	27	3
162	163	235	250	248	249	246	255	122	0	188	255	195	24	0	4

What a Computer actually sees?

- A **Digital Image** is a matrix of size (H,W,C), comprising of numbers (also known as pixel values) typically ranging from 0-255. Here H,W,C denote Height, Width and No. of Channels in a image.
- For a Grayscale image, # channels is 1, while for Colored Image its 3.

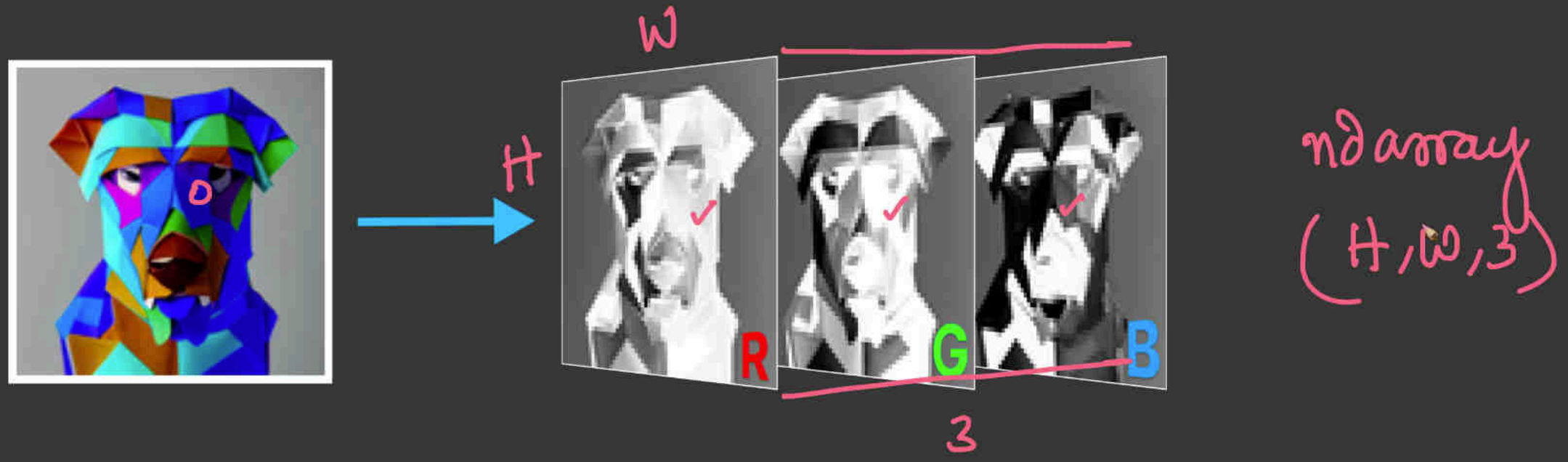


158	176	246	246	251	241	235	242	254	249	244	253	248	255	127	0
159	172	243	247	249	239	240	251	255	185	220	255	249	244	27	4
160	168	239	248	247	250	253	252	246	109	247	250	255	160	4	28
161	164	237	248	248	249	249	255	199	15	234	255	254	97	27	3
162	163	235	250	248	249	246	255	122	0	188	255	195	24	0	4

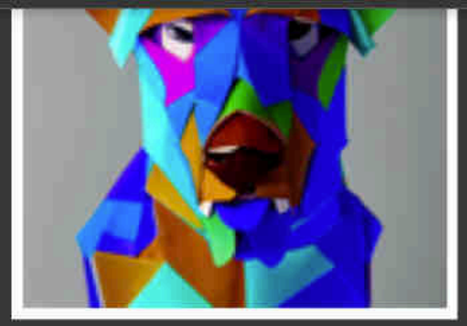


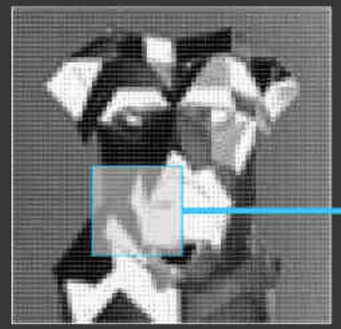
What a Computer actually sees?

- A **Digital Image** is a matrix of size (H,W,C), comprising of numbers (also known as pixel values) typically ranging from 0-255. Here H,W,C denote Height, Width and No. of Channels in a image.
- For a Grayscale image, # channels is 1, while for Colored Image its 3.



158	176	246	246	251	241	235	242	254	249	244	253	248	255	127	0
159	172	243	247	249	239	240	251	255	185	220	255	249	244	27	4
160	168	239	248	247	250	253	252	246	109	247	250	255	160	4	28
161	164	237	248	248	249	249	255	199	15	234	255	254	97	27	3
162	163	235	250	248	249	246	255	122	0	188	255	195	24	0	4

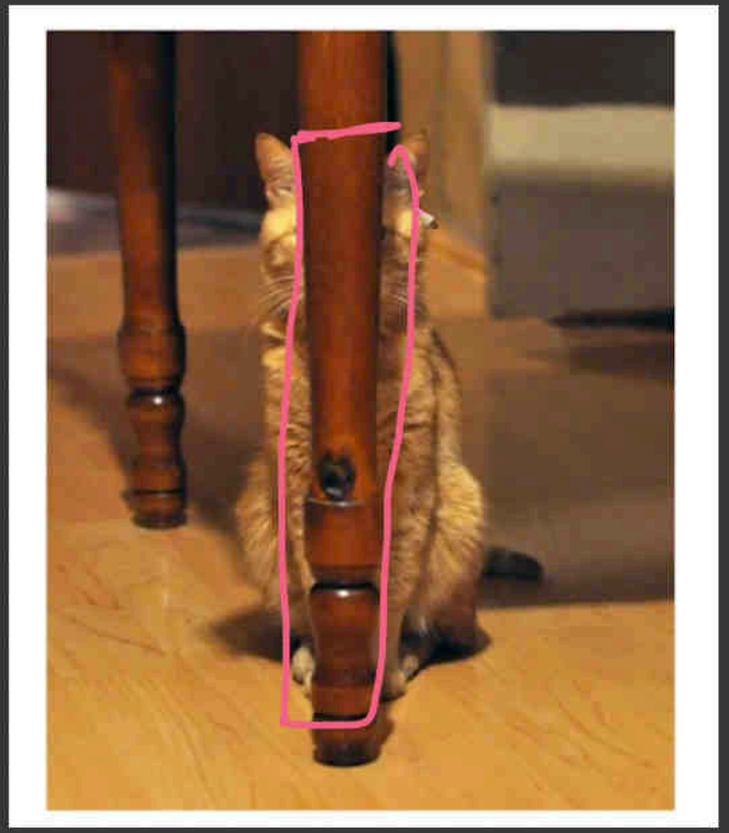




158	176	246	246	251	241	235	242	254	249	244	253	248	255	127	0
159	172	243	247	249	239	240	251	255	185	220	255	249	244	27	4
160	168	239	248	247	250	253	252	246	109	247	250	255	160	4	28
161	164	237	248	248	249	249	255	199	15	234	255	254	97	27	3
162	163	235	250	248	249	246	255	122	0	188	255	195	24	0	4
162	162	233	252	249	249	251	250	44	0	139	255	62	0	8	6
163	158	228	254	249	246	255	188	0	0	93	185	0	0	0	0
161	165	236	252	249	246	255	190	0	0	38	68	13	50	78	87
160	224	253	247	249	248	249	251	58	0	12	25	55	86	100	67
207	255	251	249	255	247	247	255	189	0	8	32	0	0	0	0
255	251	255	145	144	255	244	248	253	58	0	7	12	12	9	5
255	248	251	46	0	192	255	241	255	112	0	3	1	3	3	3
248	255	205	3	0	22	229	250	255	167	0	8	1	4	3	2
243	255	154	0	12	0	66	251	253	209	5	12	10	5	5	3
245	255	182	16	0	7	0	116	255	232	30	0	3	5	1	5
250	252	227	155	25	7	2	0	169	255	57	8	34	4	1	4

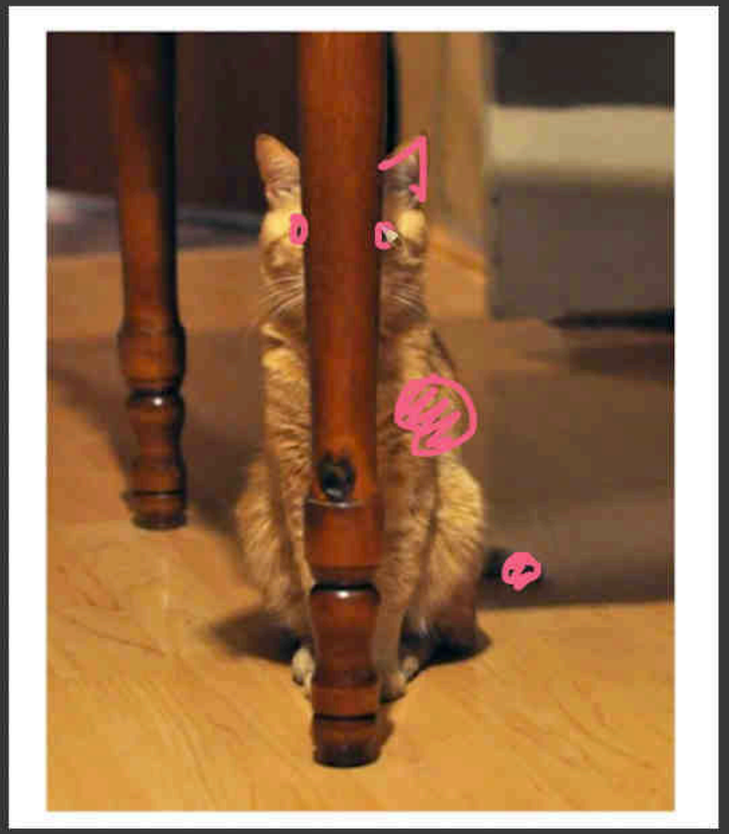
Q. Is CV Easy?

- Occlusion



- Illumination variability

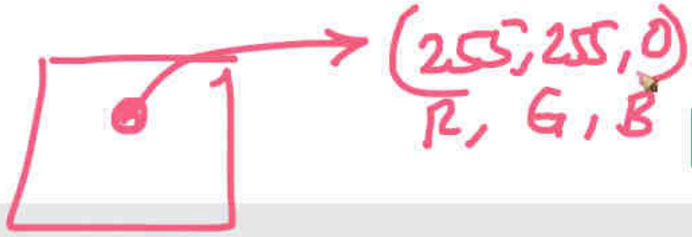
- Occlusion



- Illumination variability

Colors RGB

‹ Previous



Next ›

RGB Calculator

rgb(255, 255, 0)

#ffff00

hsl(60, 100%, 50%)

R: 255 — 255

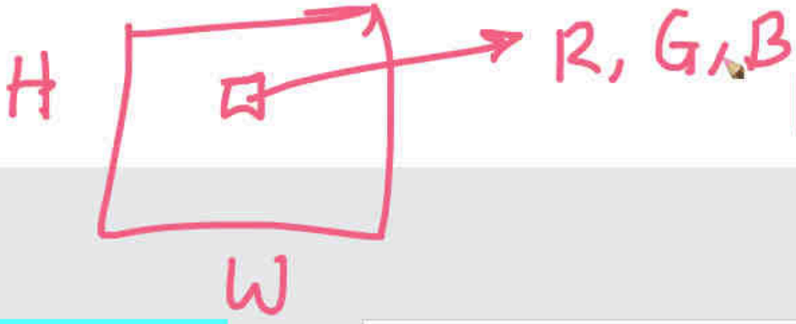
G: 255 — 255

B: 0 — 0

Use this color in our Color Picker

Colors RGB

‹ Previous



Next ›

RGB Calculator

rgb(0, 255, 255)

#00ffff

hsl(180, 100%, 50%)

R: 0 (0)

G: 255 (255)

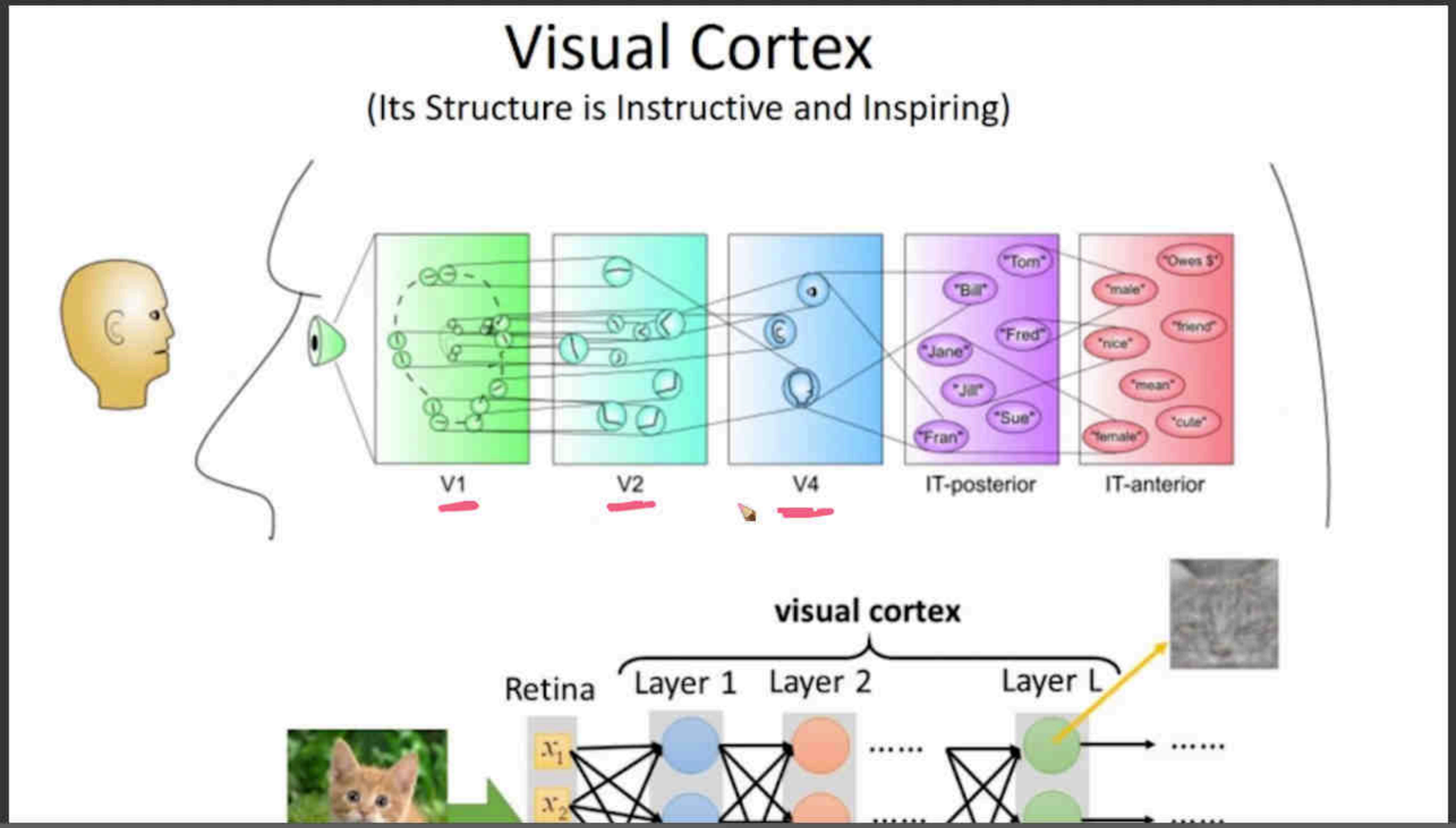
B: 255 (255)

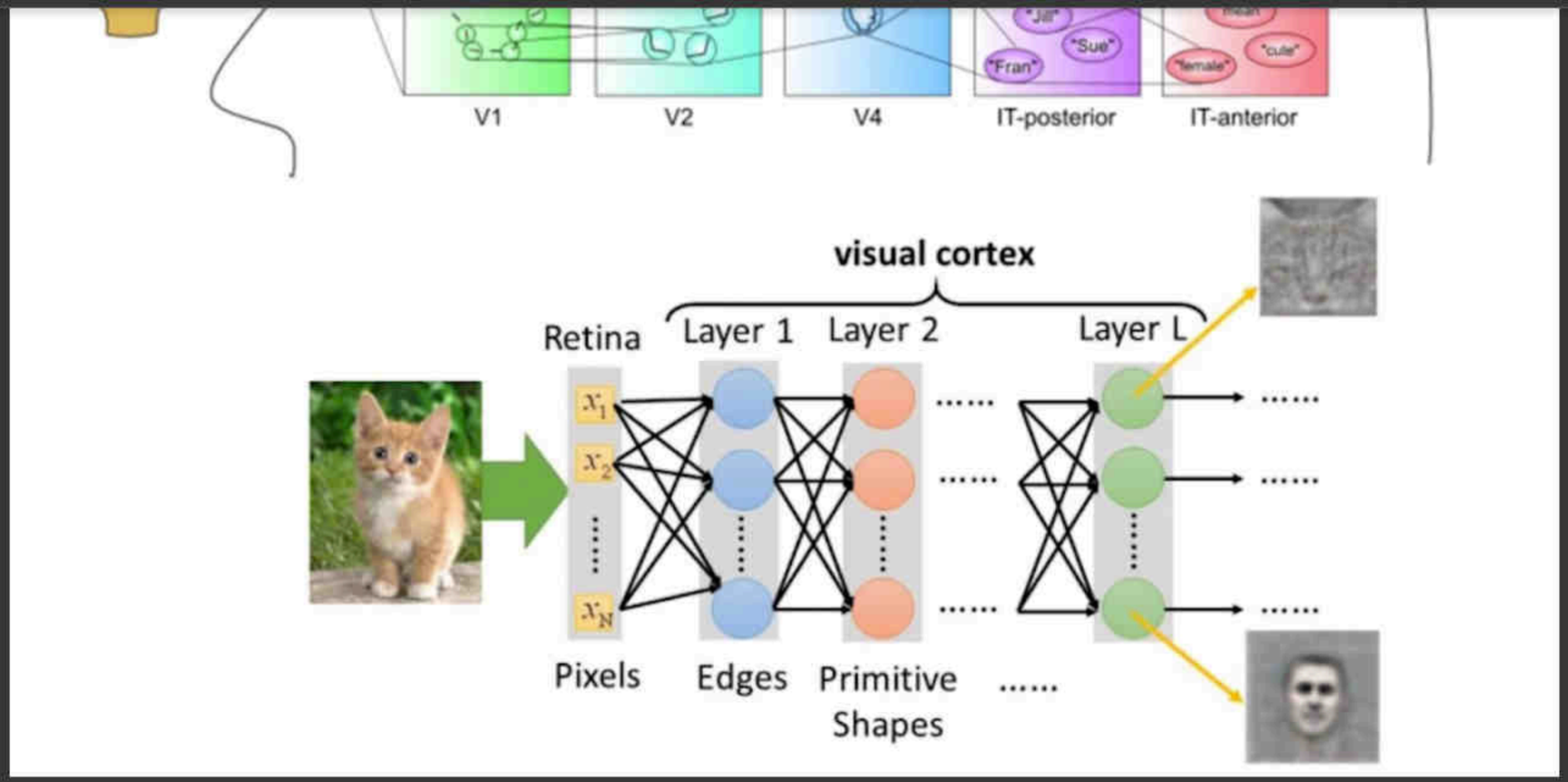
Use this color in our Color Picker

CNNs → late 1998's

↳ Yann LeCun

Our Visual Cortex system is arranged in layers and as information passes from the eyes to deeper parts of the brain, higher and higher order representations are formed. Since a DNN also works in a similar way, using it for analyzing Images makes perfect sense.





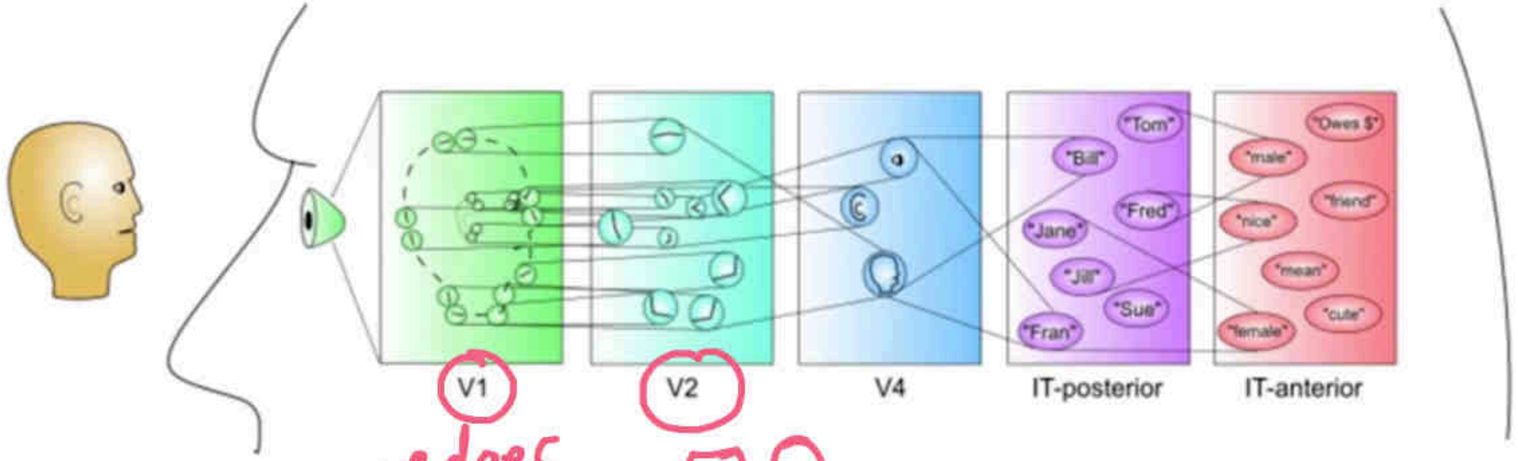
~ approximation

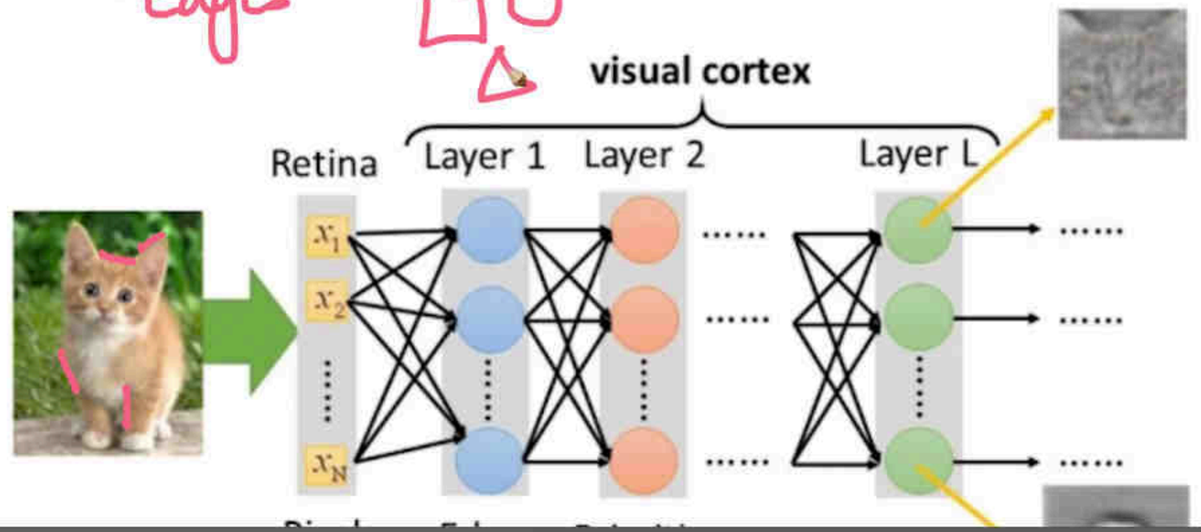
But what about the architecture of DNN, will Vanilla NN or MLP be a suitable fit?

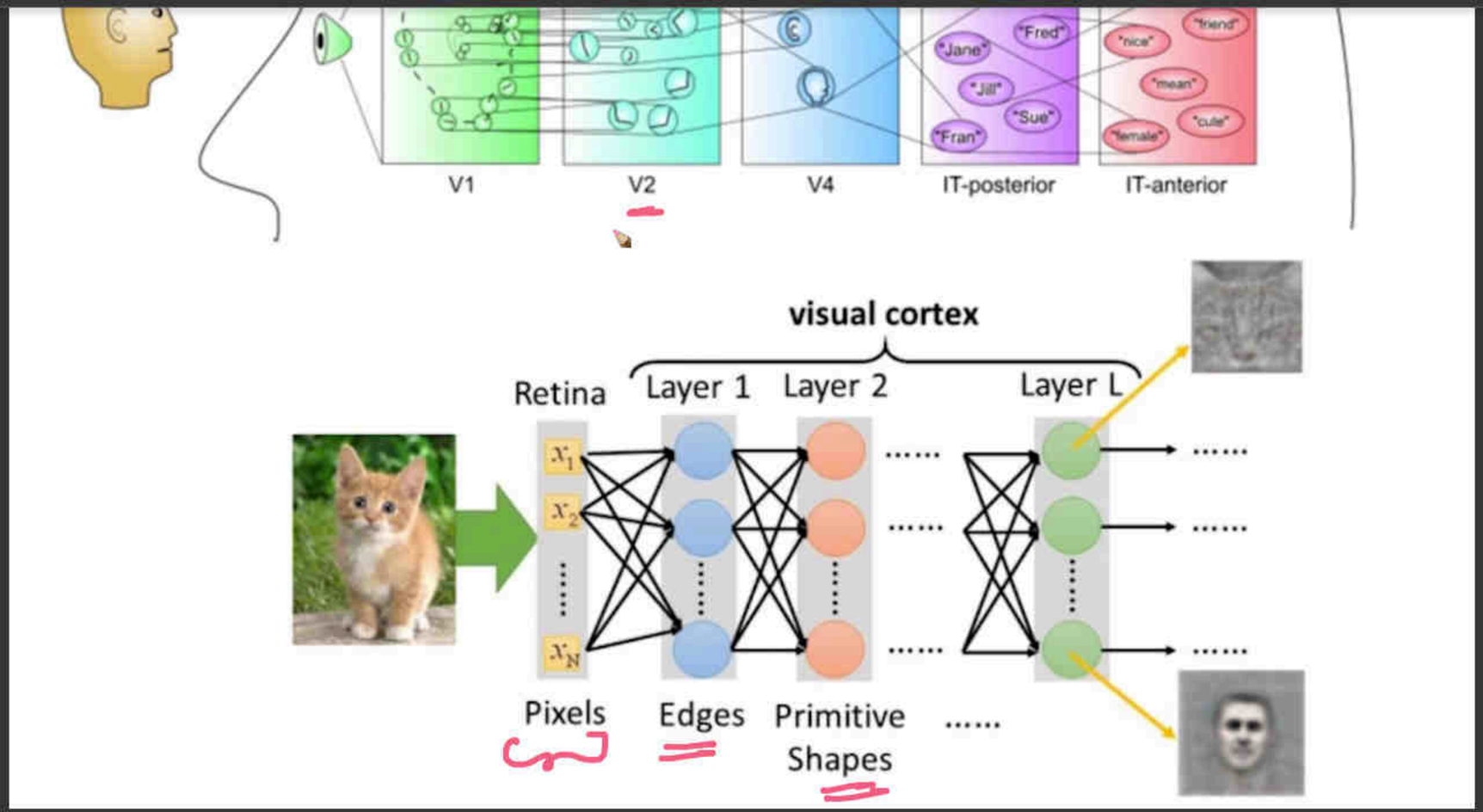
We will answer this question in Part 1 of the module!

Visual Cortex

(Its Structure is Instructive and Inspiring)





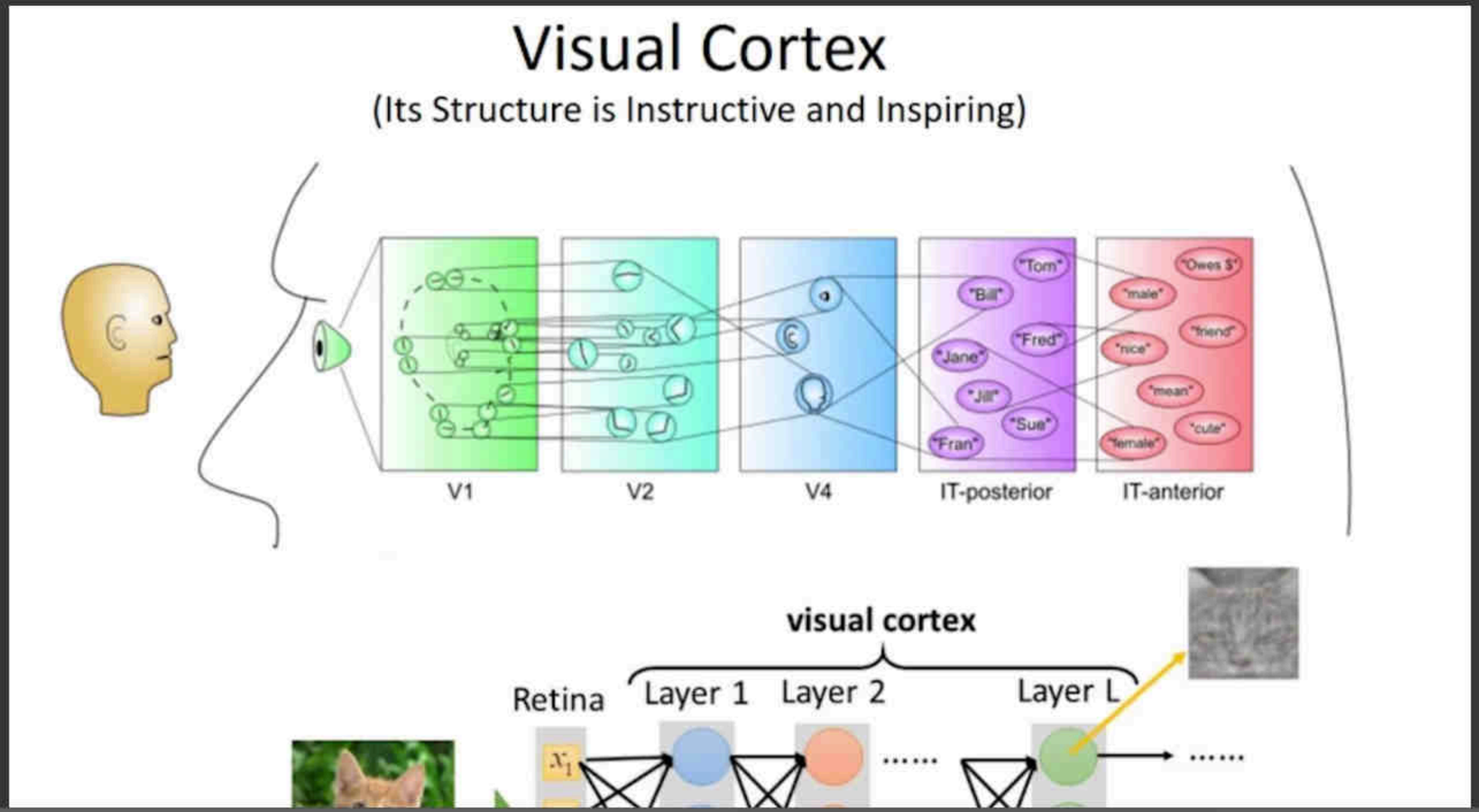


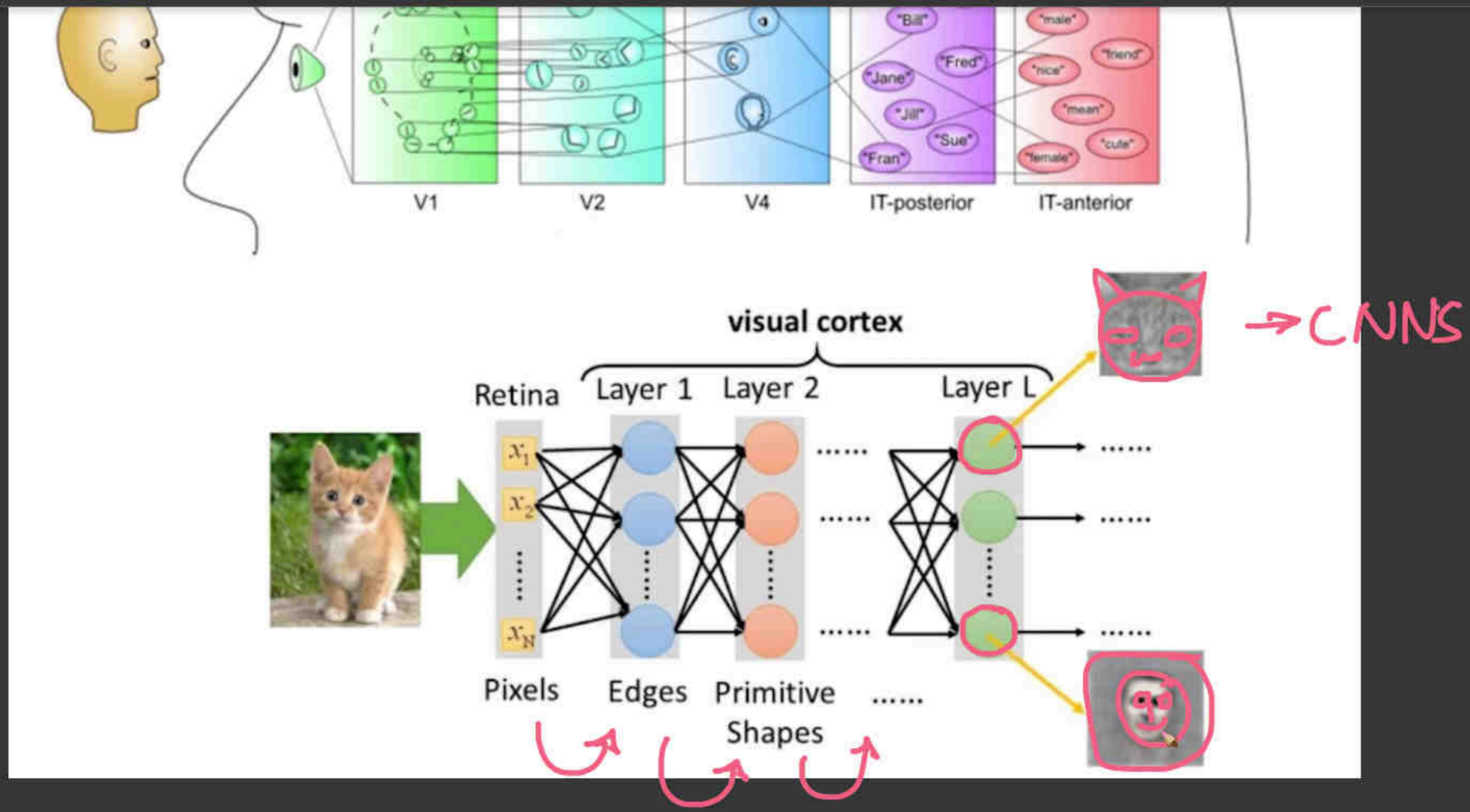
But what about the architecture of DNN, will Vanilla NN or MLP be a suitable fit?

We will answer this question in Part 1 of the module!

In order to understand why DNN works, we need to account for how visual cortex functions!

Our Visual Cortex system is arranged in layers and as information passes from the eyes to deeper parts of the brain, higher and higher order representations are formed. Since a DNN also works in a similar way, using it for analyzing Images makes perfect sense.





But what about the architecture of DNN, will Vanilla NN or MLP be a suitable fit?

We will answer this question in Part 1 of the module!

History:

1950's — 1990's → IP & classical CV

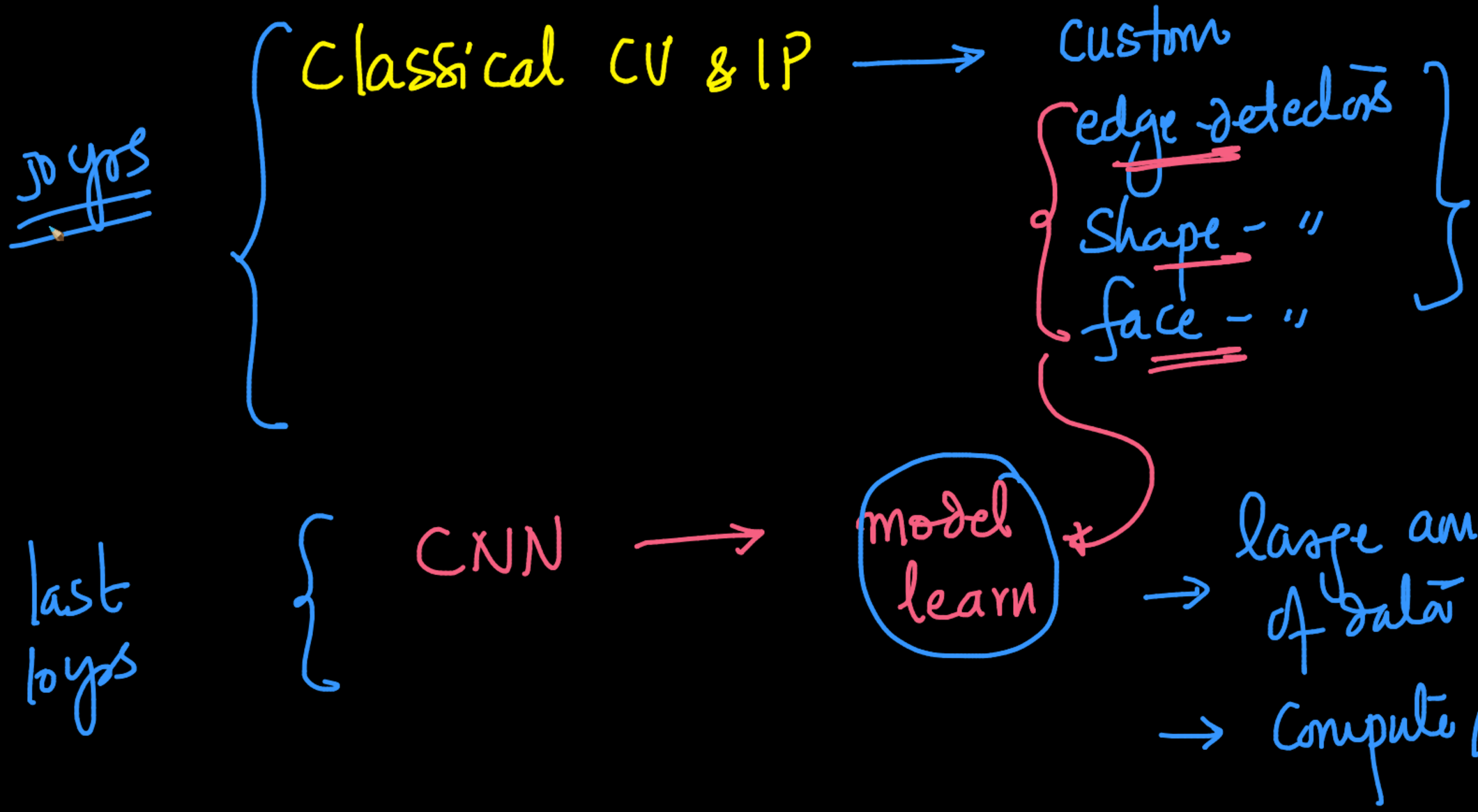
- shape detection
- edge-detection
- face detection

90's → CNN (ignored)

2012 → AlexNet (CNN, scaled, GPUs)

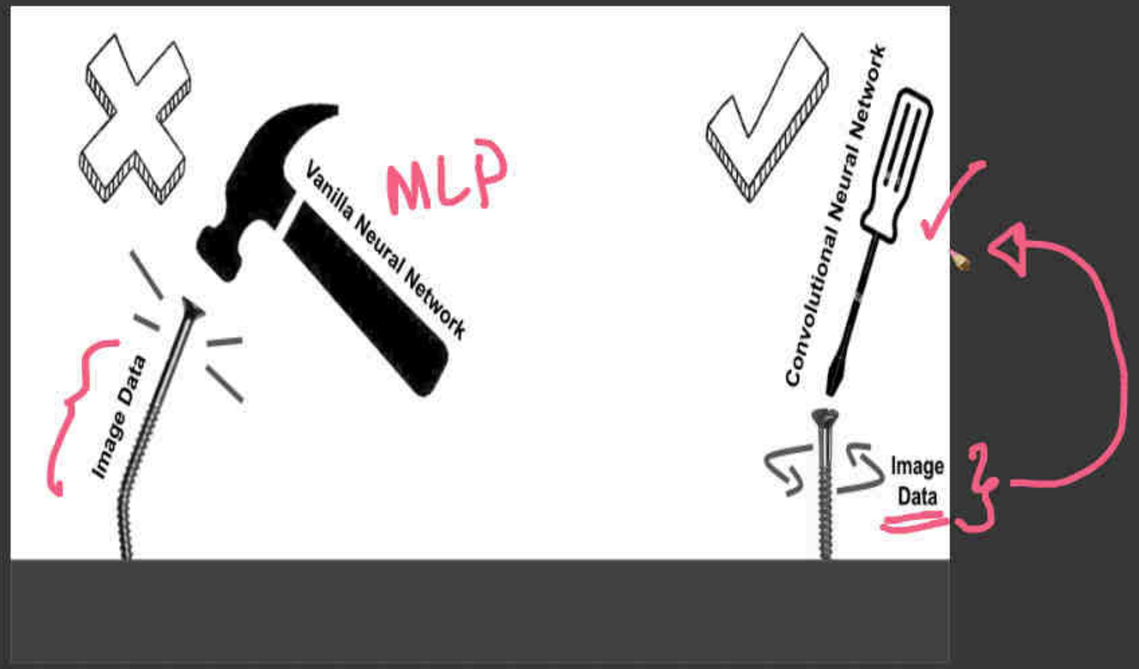
datasets

{ CNNs ✓



We will answer this question in Part 1 of the module!

PART 1: Hammer and Screw



In the Previous lectures we learnt about ANNs and where to use them.

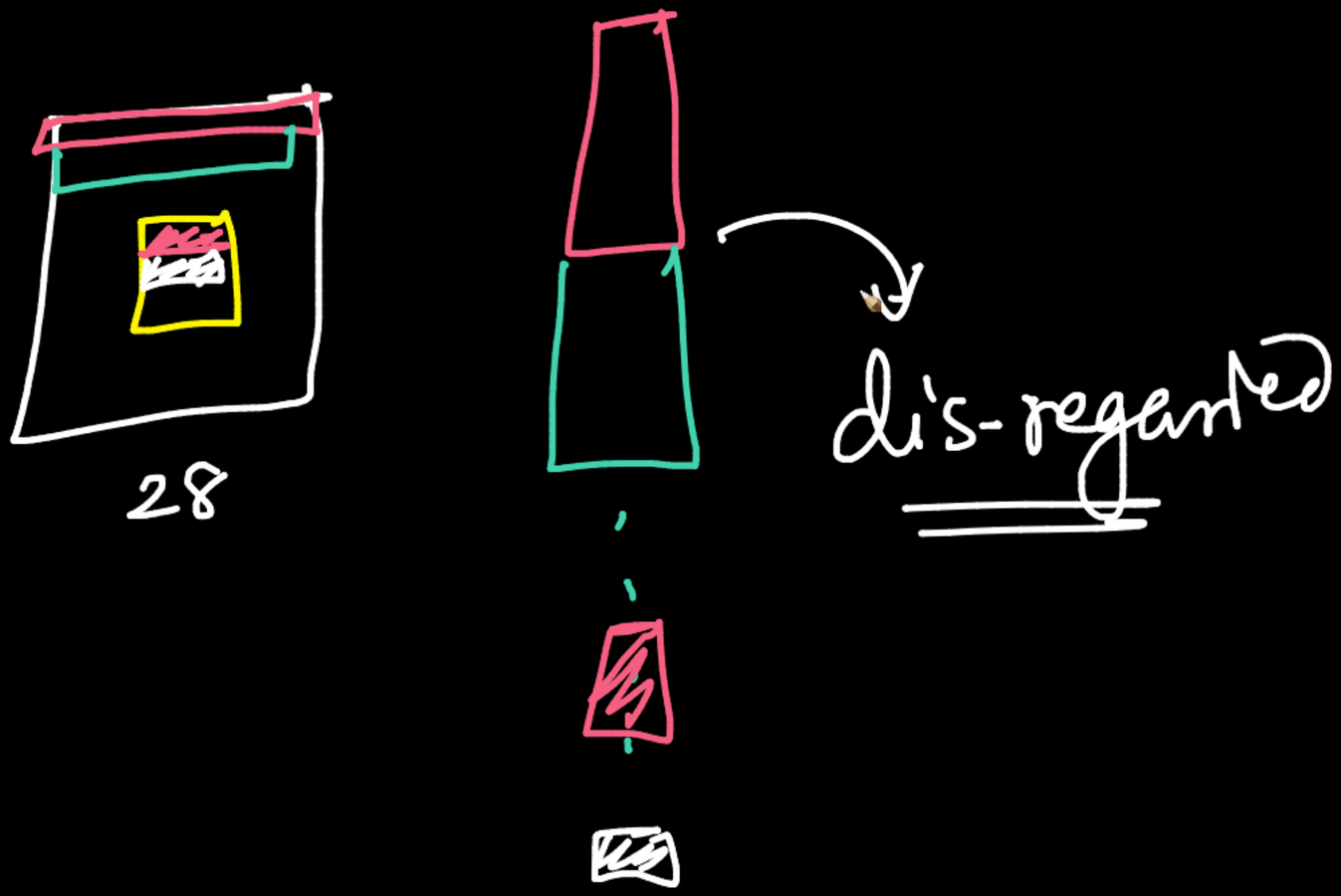
Now let's experiment what happens if we try to fit a Vanilla Neural Network (Hammer) to a very different kind of data, aka Images (Screw) and why this hammering of screw doesn't make sense!

CNN

↑ internals ↓ (code)

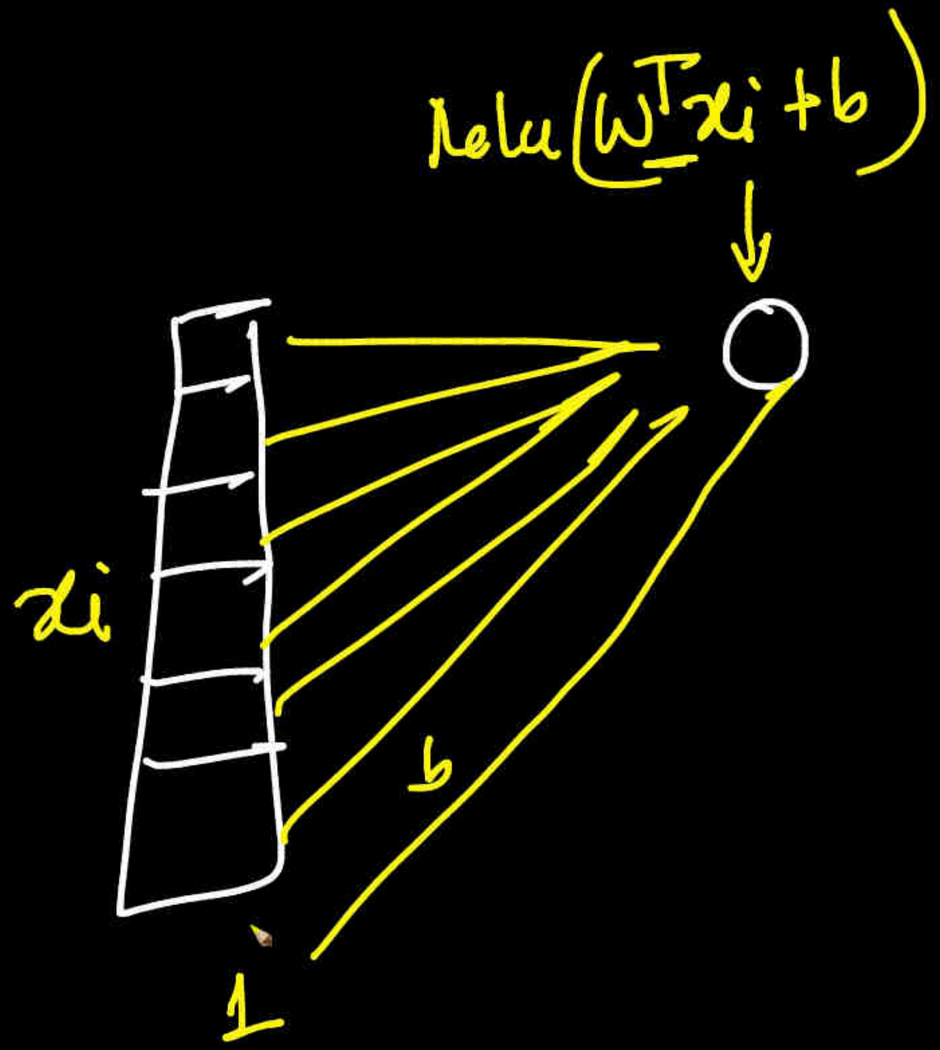
CNN

Convolution

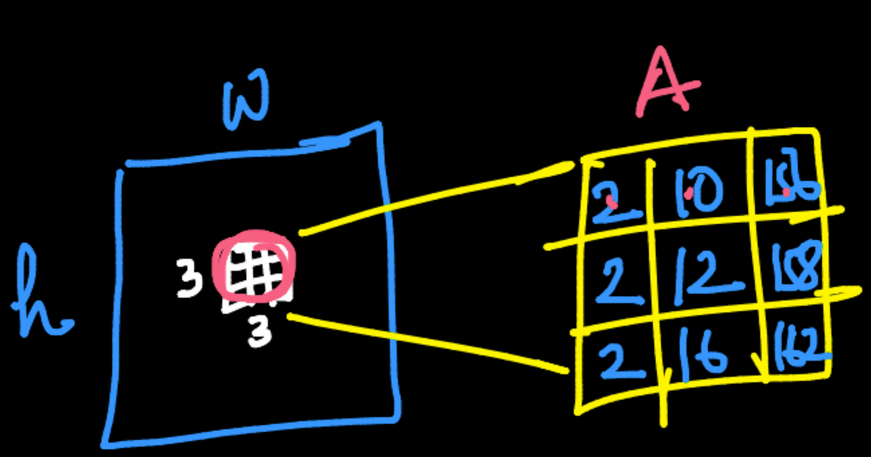


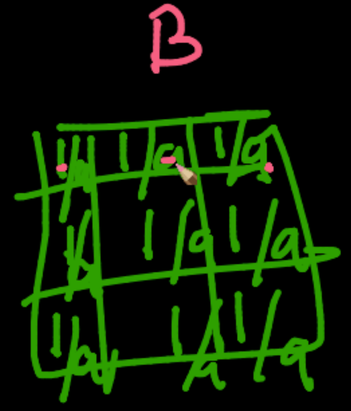
Spatial neighbhood

MLP



grayscale!





average

$$= \frac{2}{9} + \frac{10}{9} + \frac{156}{9} + \frac{2}{9} + \frac{12}{9} + \dots$$

component wise mul

$A \odot B$

Convolution { component-wise matrix mul + addition

↓

treating the spatial neighborhood

Classical IP

Neuroscience

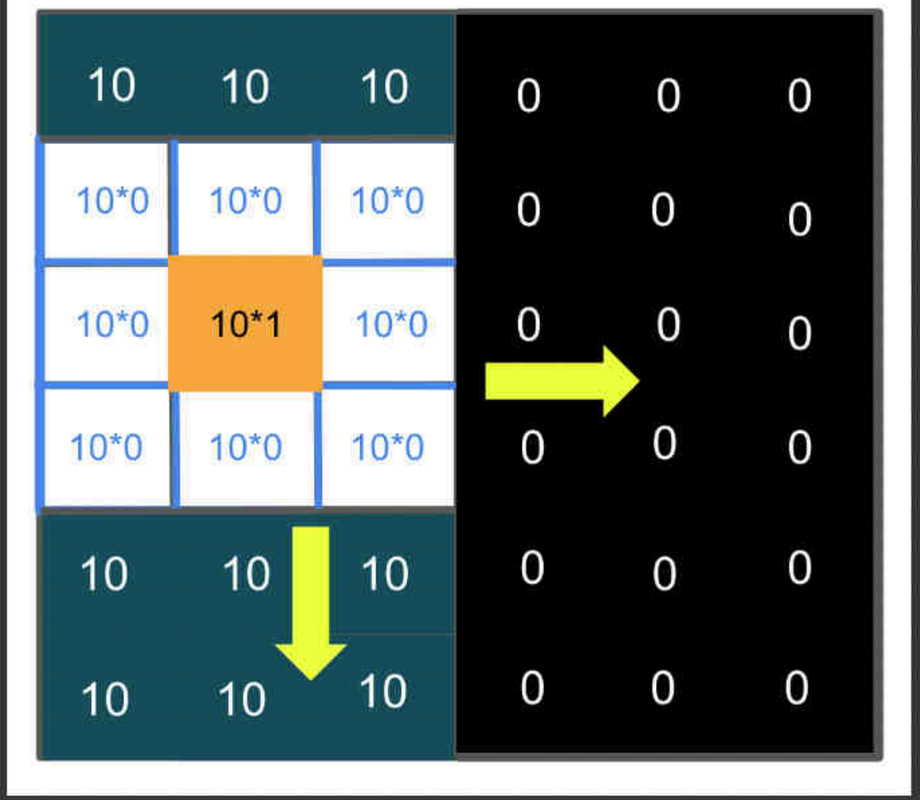
pixels → edges → shapes

↓

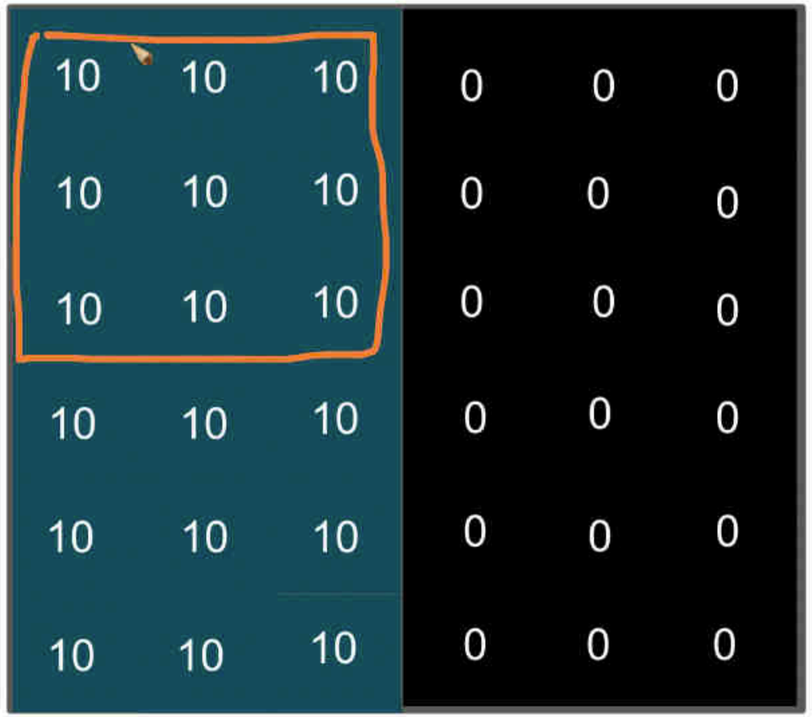
objects

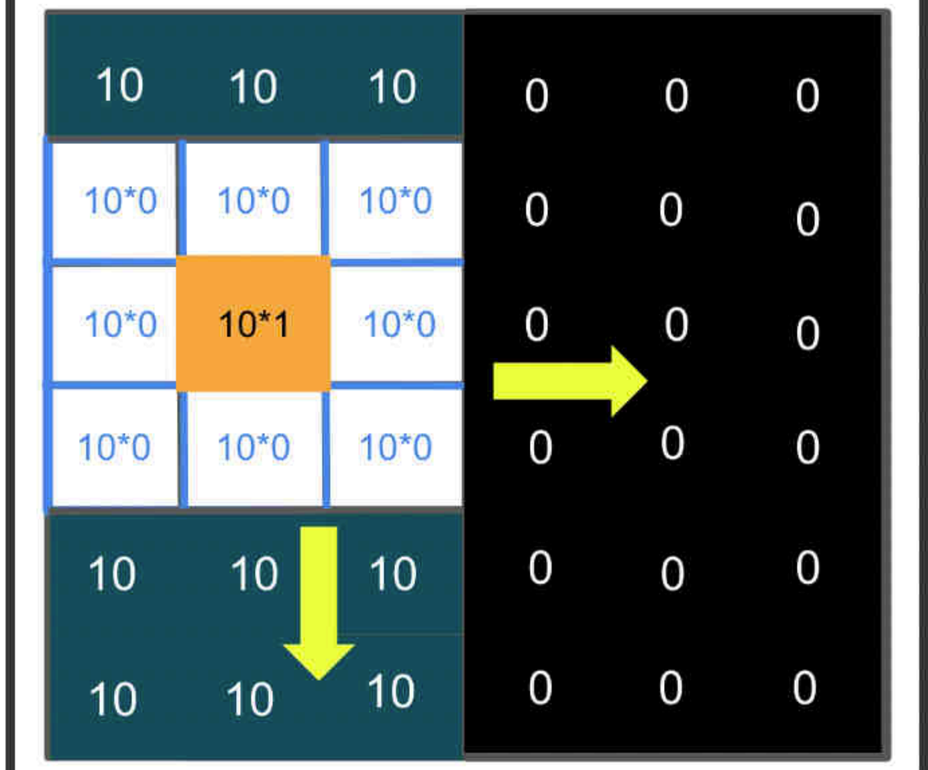
↓

faces/cats..



- A kernel can be visualized as a small matrix that slides across, from left-to-right and top-to-bottom, of a larger image.
- At each pixel in the input image, the neighborhood of the image is convolved with the kernel and the output stored.
- Inshort, a kernel find relations within the neighbouring pixels in its area iteratively and once the finishes sliding over the image, it is able to correlate certain features in a image

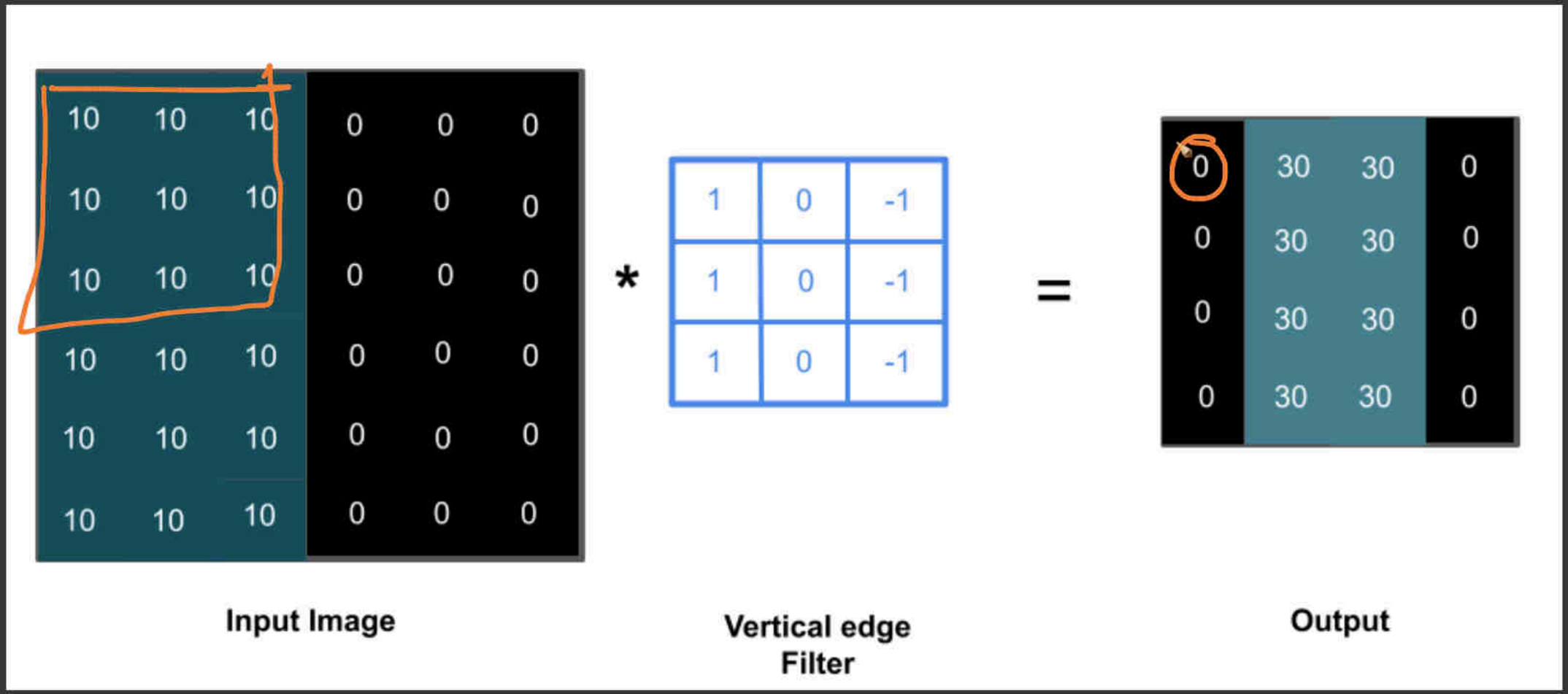




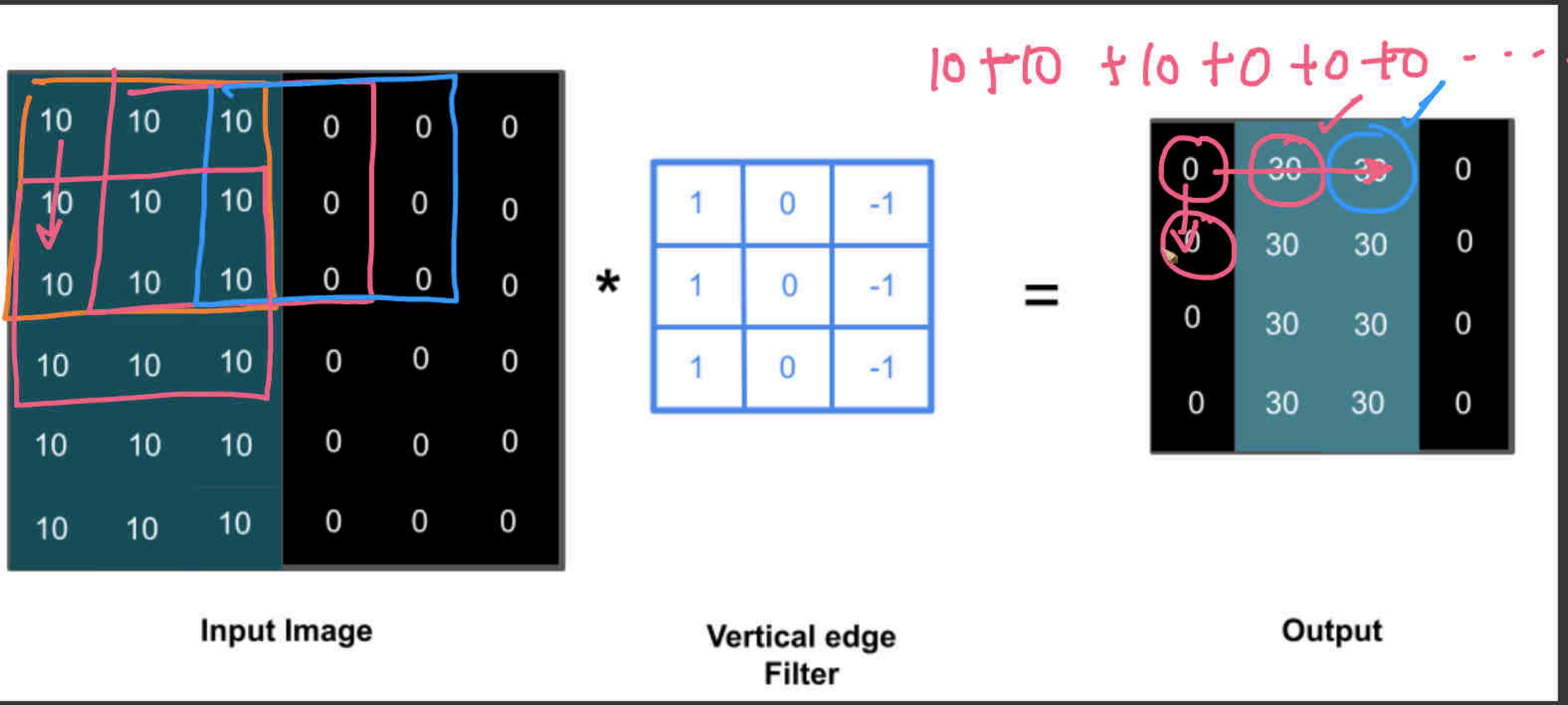
Conv-matrix
filter
kernel

- A kernel can be visualized as a small matrix that slides across, from left-to-right and top-to-bottom, of a larger image.
- At each pixel in the input image, the neighborhood of the image is convolved with the kernel and the output stored.
- Inshort, a kernel find relations within the neighbouring pixels in its area iteratively and once the finishes sliding over the image, it is able to correlate certain features in a image

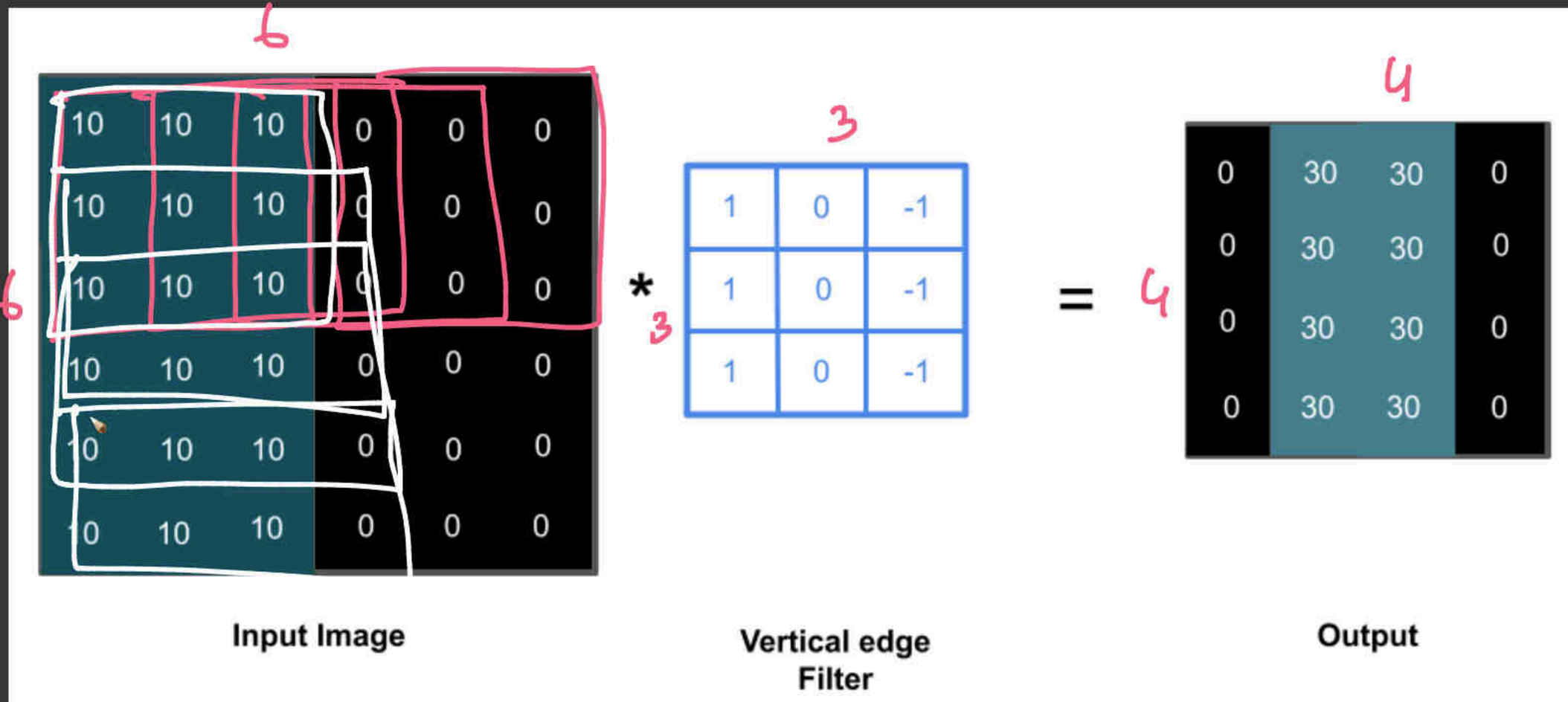
- Let's now convolve the kernel on the Image to see what we get as output:



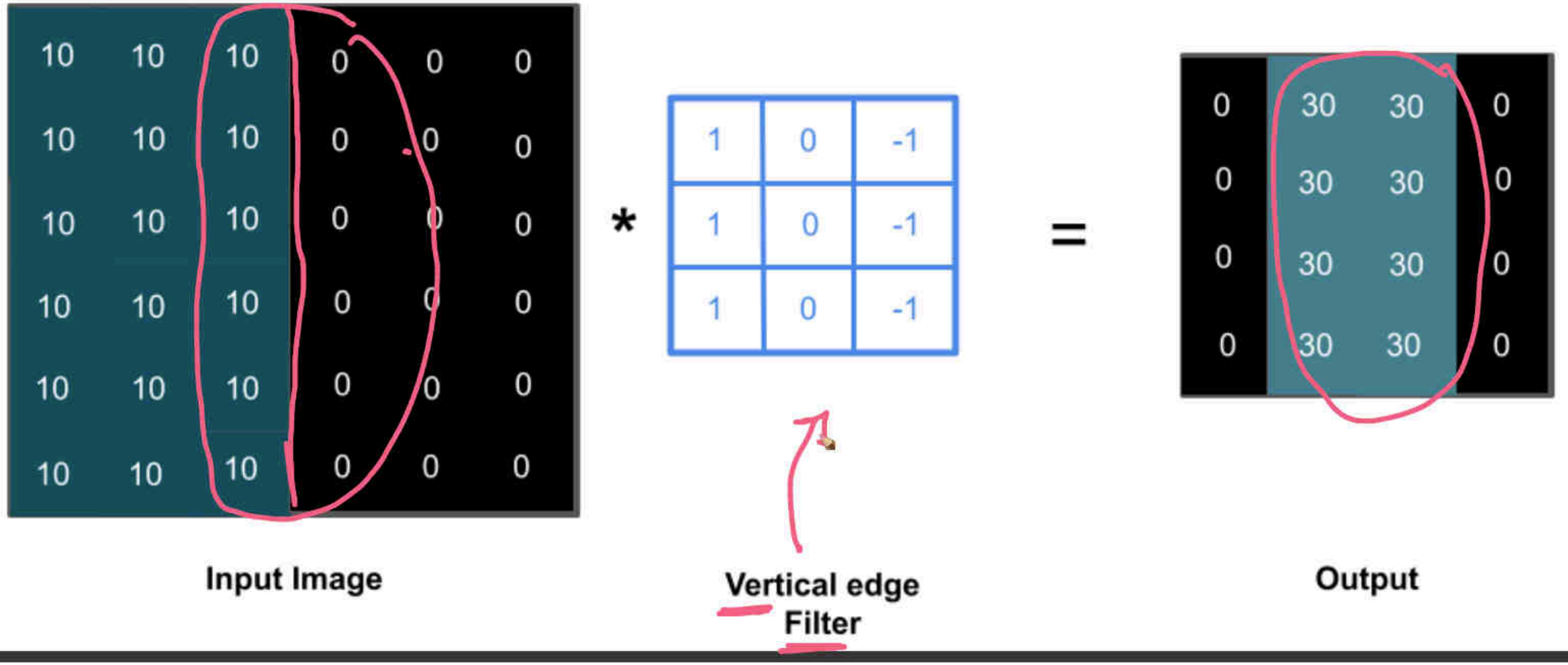
- Let's now convolve the kernel on the Image to see what we get as output:



- Let's now convolve the kernel on the Image to see what we get as output:

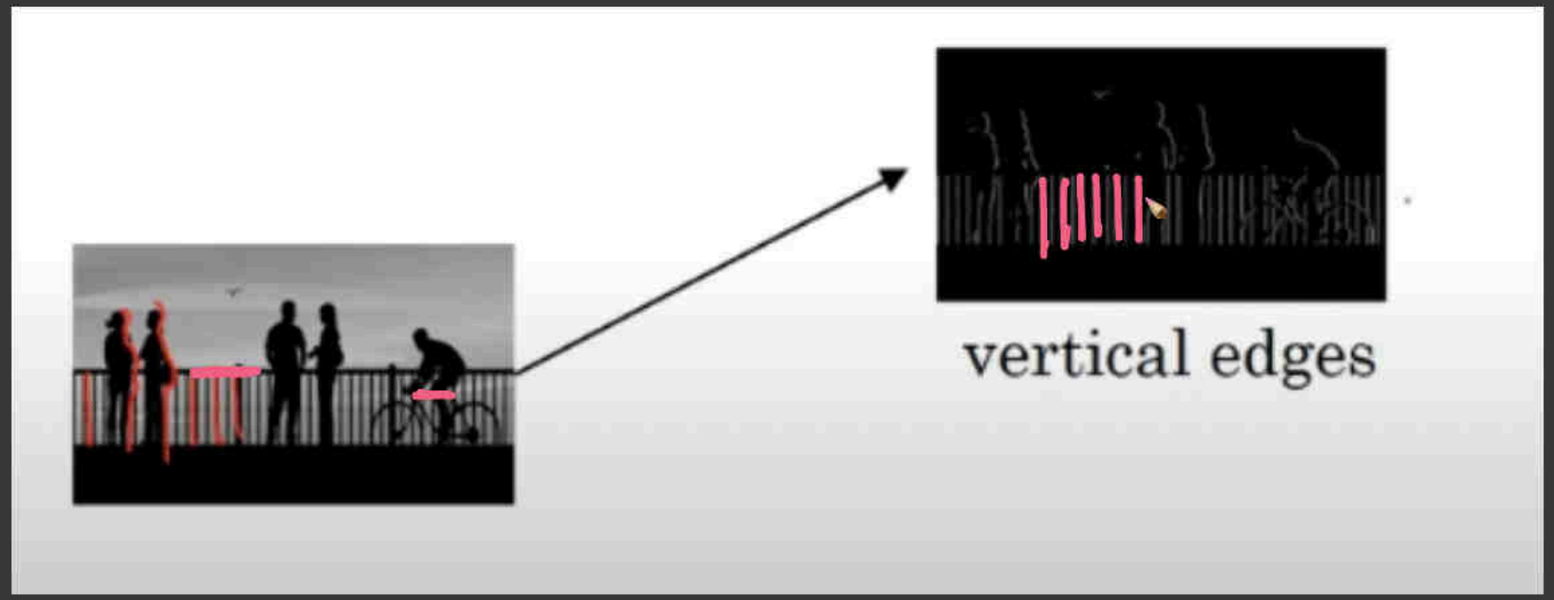


- Let's now convolve the kernel on the Image to see what we get as output:



correlating the pixels between the car and the road.

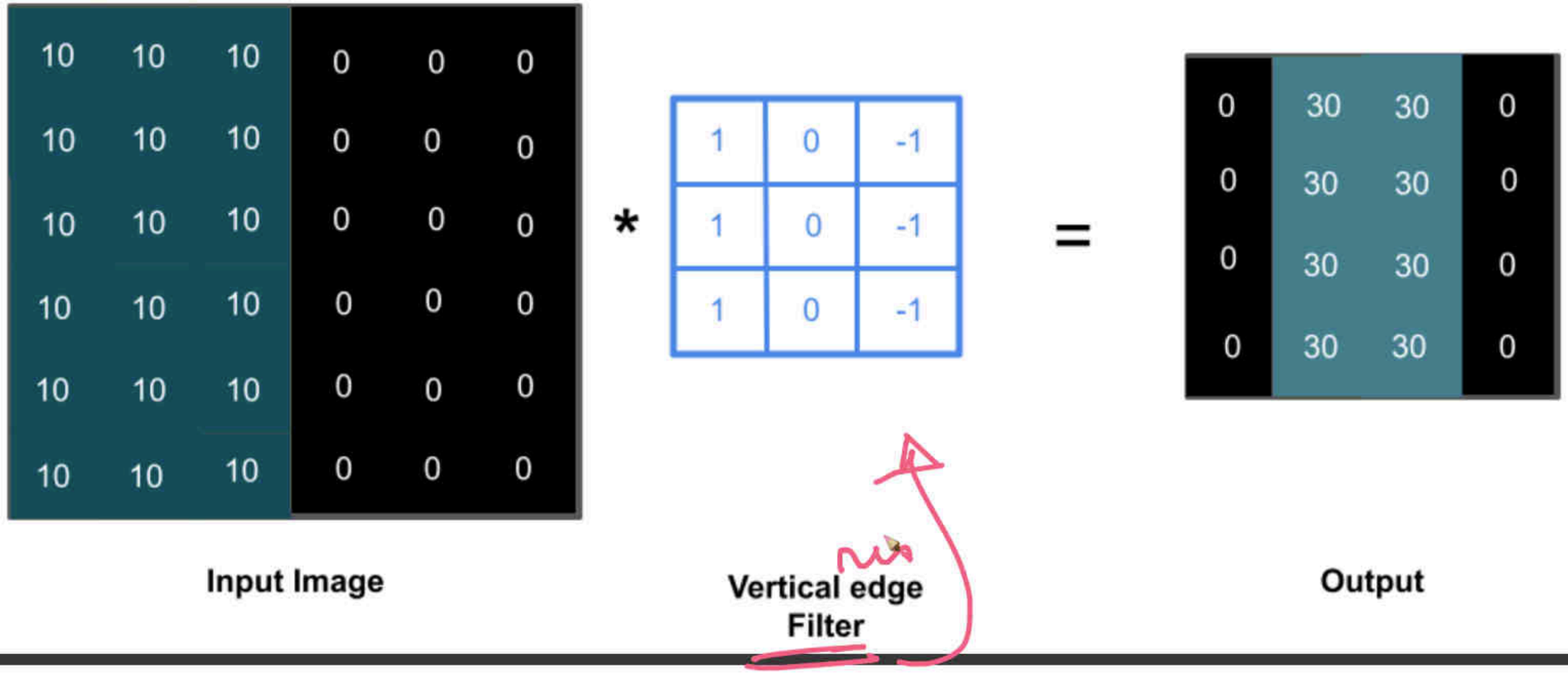
Let's see how can we make Computer to detect vertical edges ?

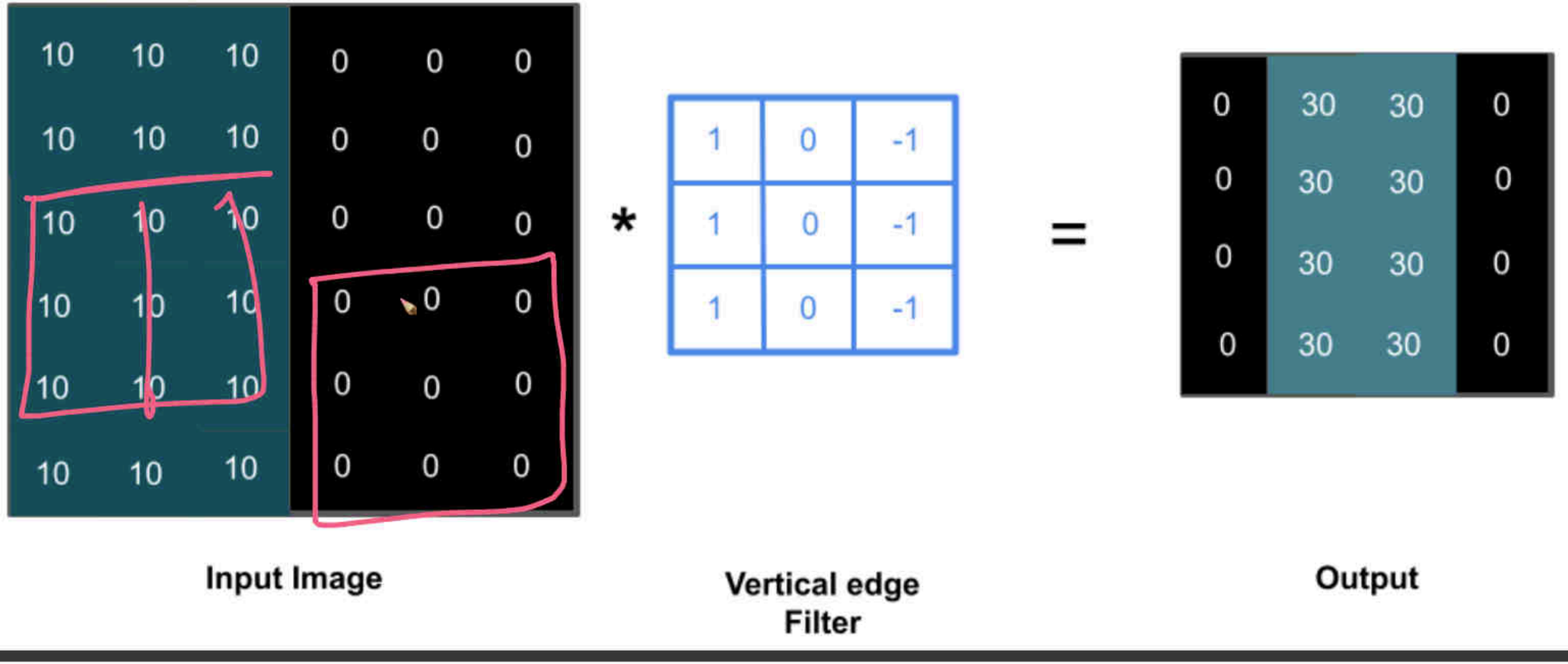


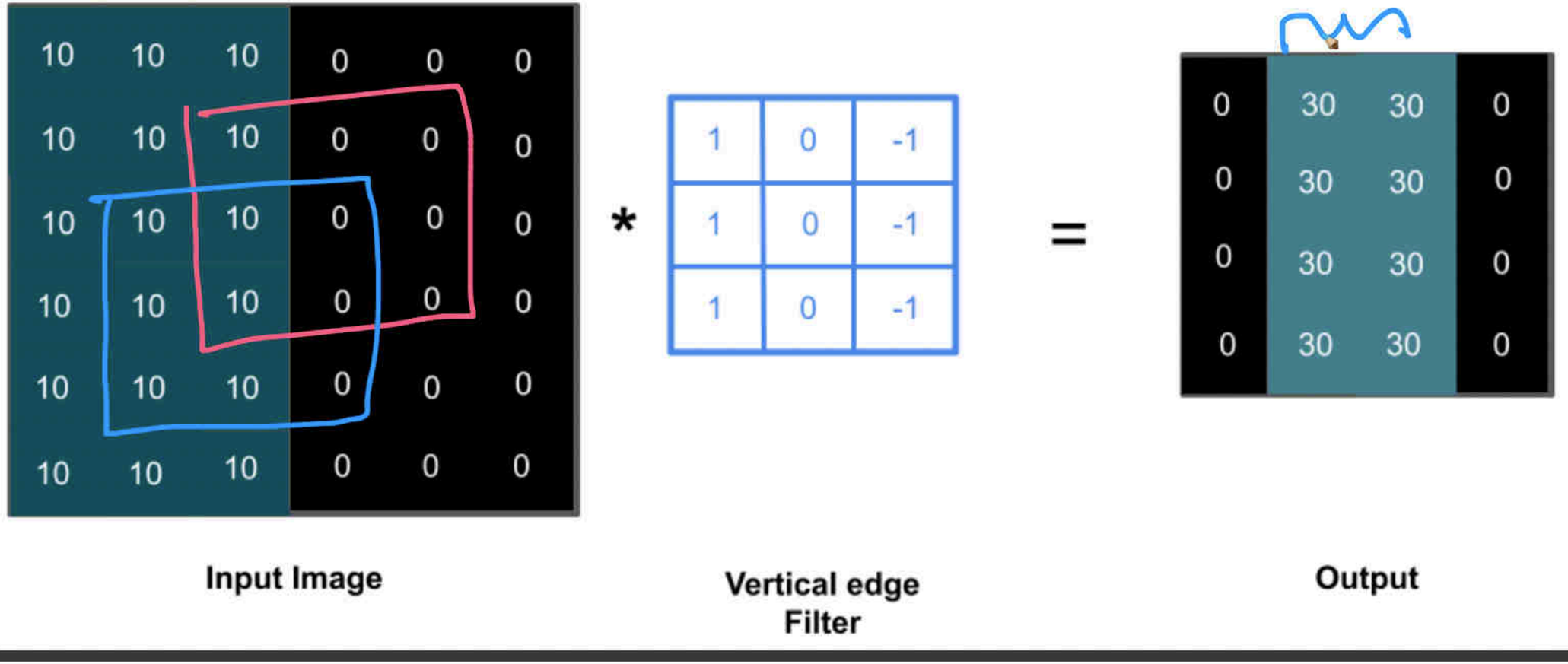
By using APPROPRIATE kernels/filters.

Let's take an example of 1-channel Input Image and a given 3x3 kernel -

- Here **10 denotes a bright pixel while 0 denotes a dark one**

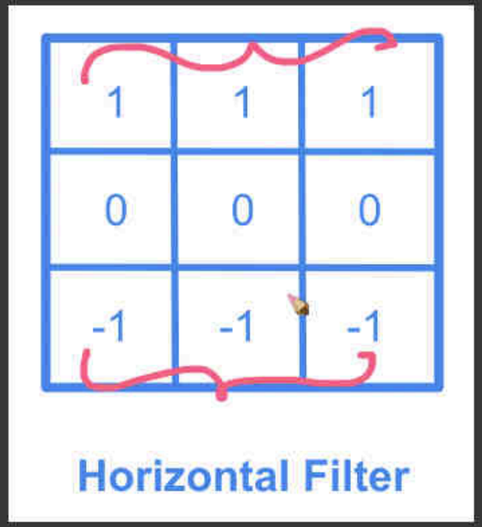






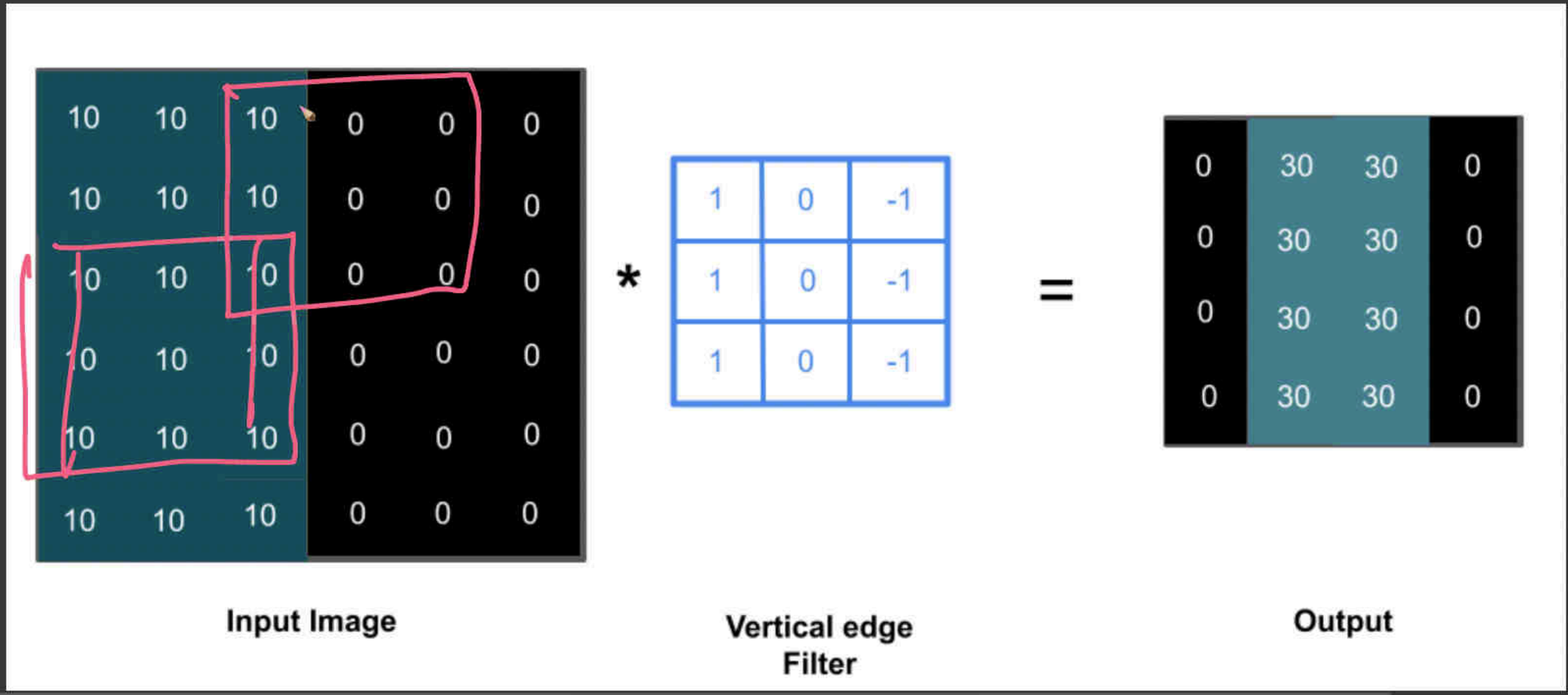
- Notice **after some computing**, the **output matrix was able to detect the vertical line** (highlighted in blue) which separated the bright and dark object.
- Also this **shifting of kernel/filter help make the image translation invariant** as **no matter where the vertical line is**, the **output matrix will have the same pixel values for the vertical lines.**

Q. What kind of filter do you think we can use for detecting horizontal edges?



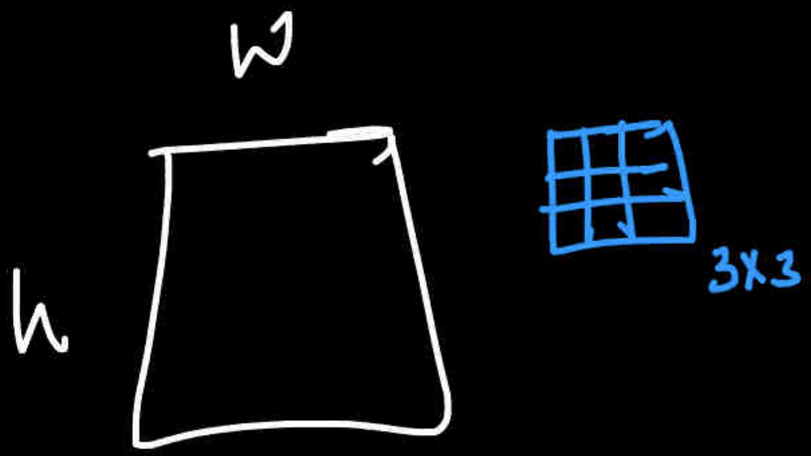
Q. How can we apply both horizontal and vertical edge detector on the same image matrix?

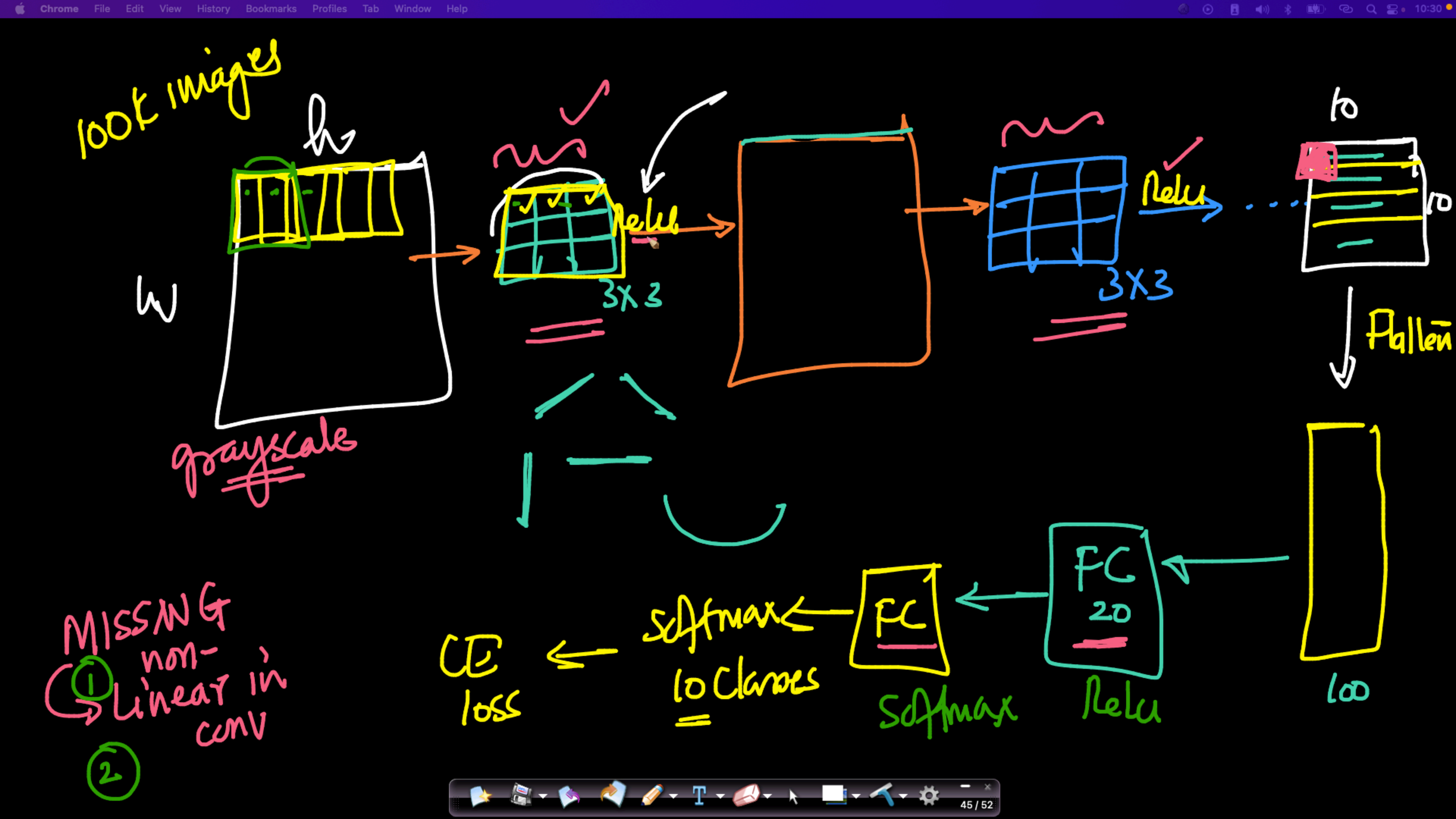
- We can clearly **see the vertical line that separates the two**
- Let's now convolve the kernel on the Image to see what we get as output:

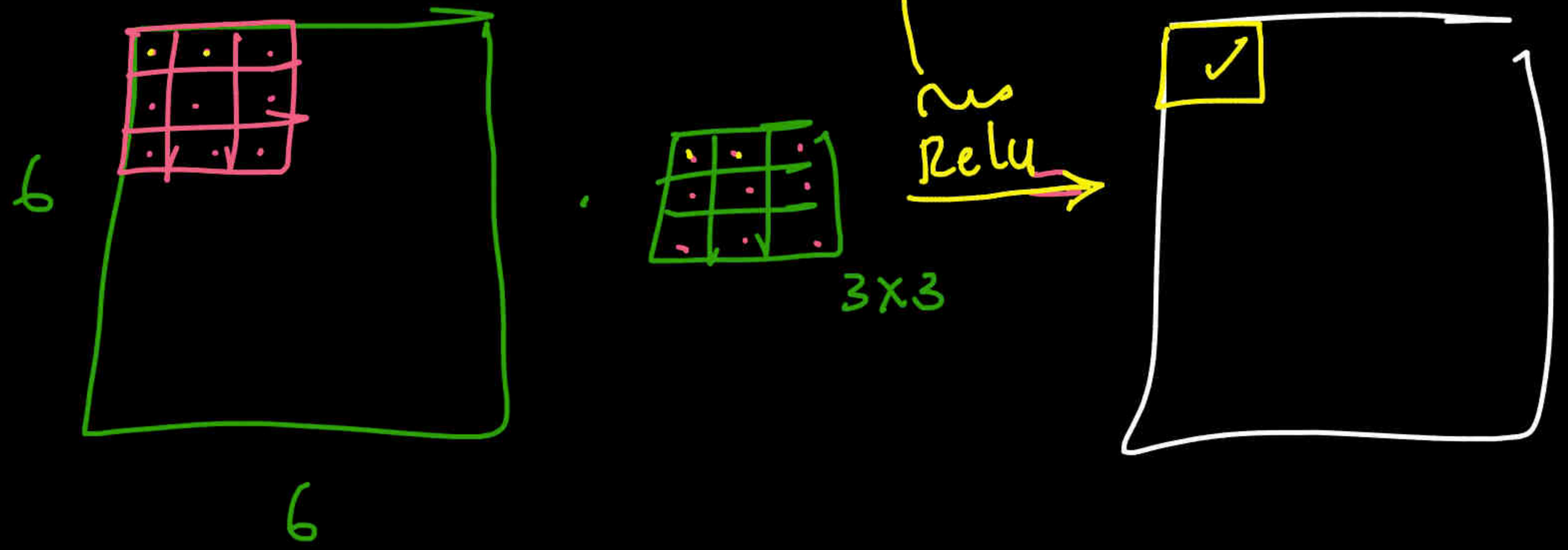


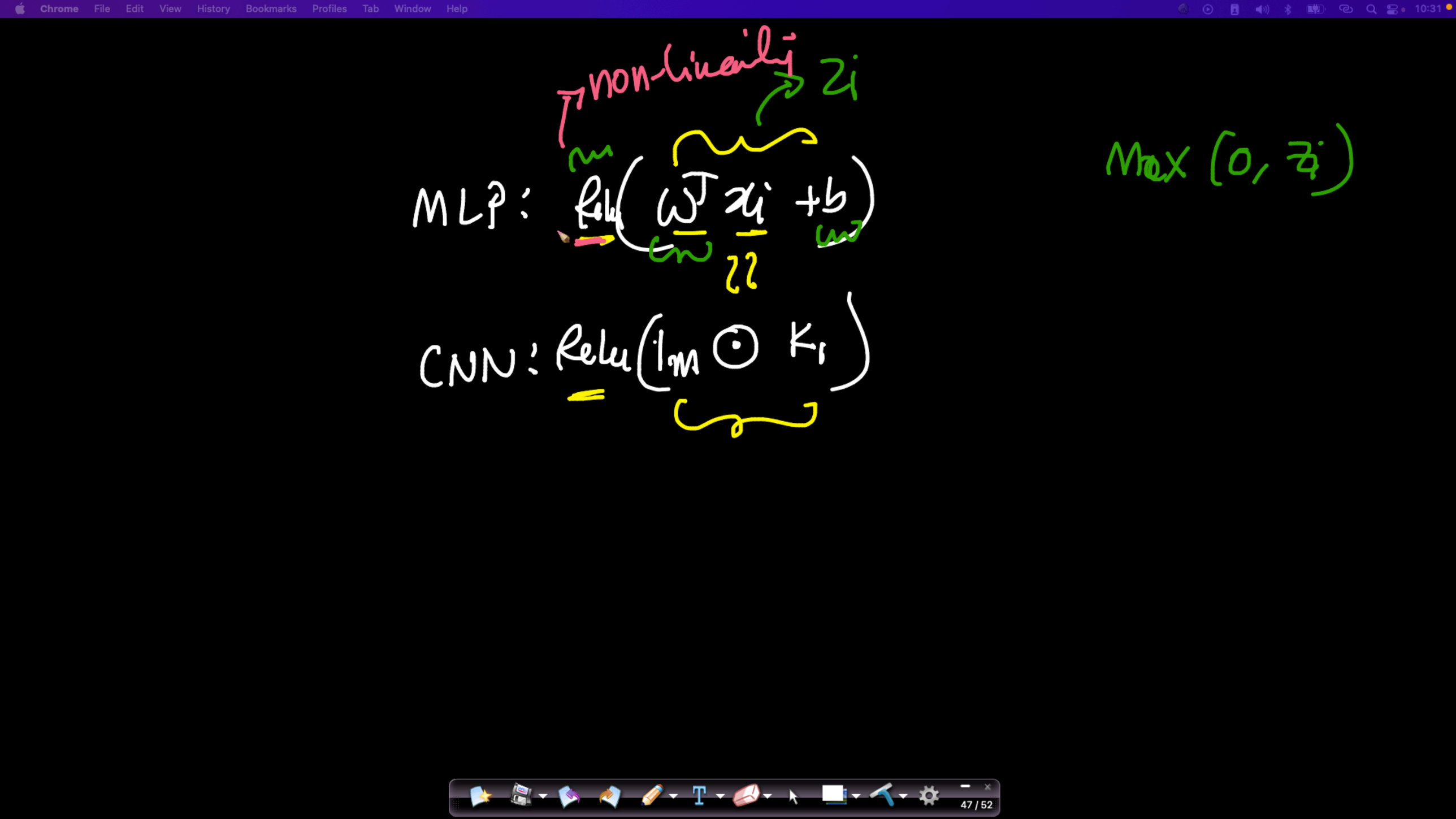
Convolution → Spatial Coherence

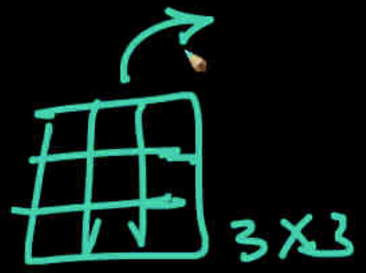
→ edge detection (IP)



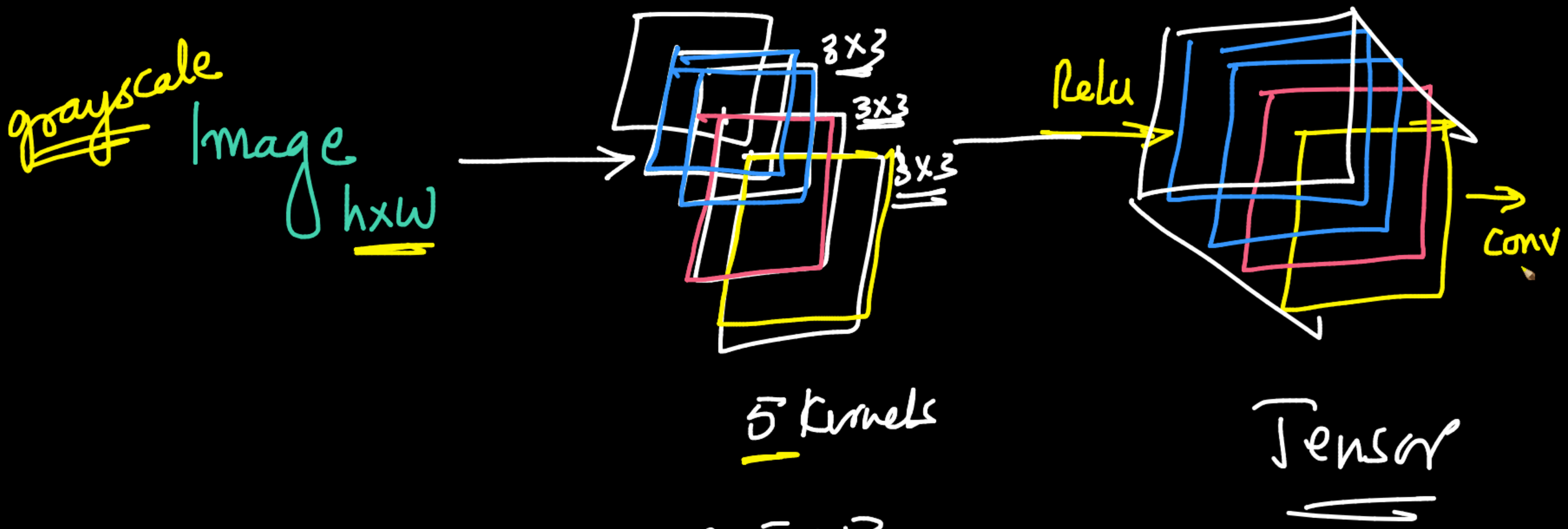

100k images
h
w
grayscale
3X3
relu
3X3
relu
10
Pallen
100
FC
20
relu
FC
softmax
10 Classes
softmax
CE
loss
MISSING
non-
linear in
conv
1
2

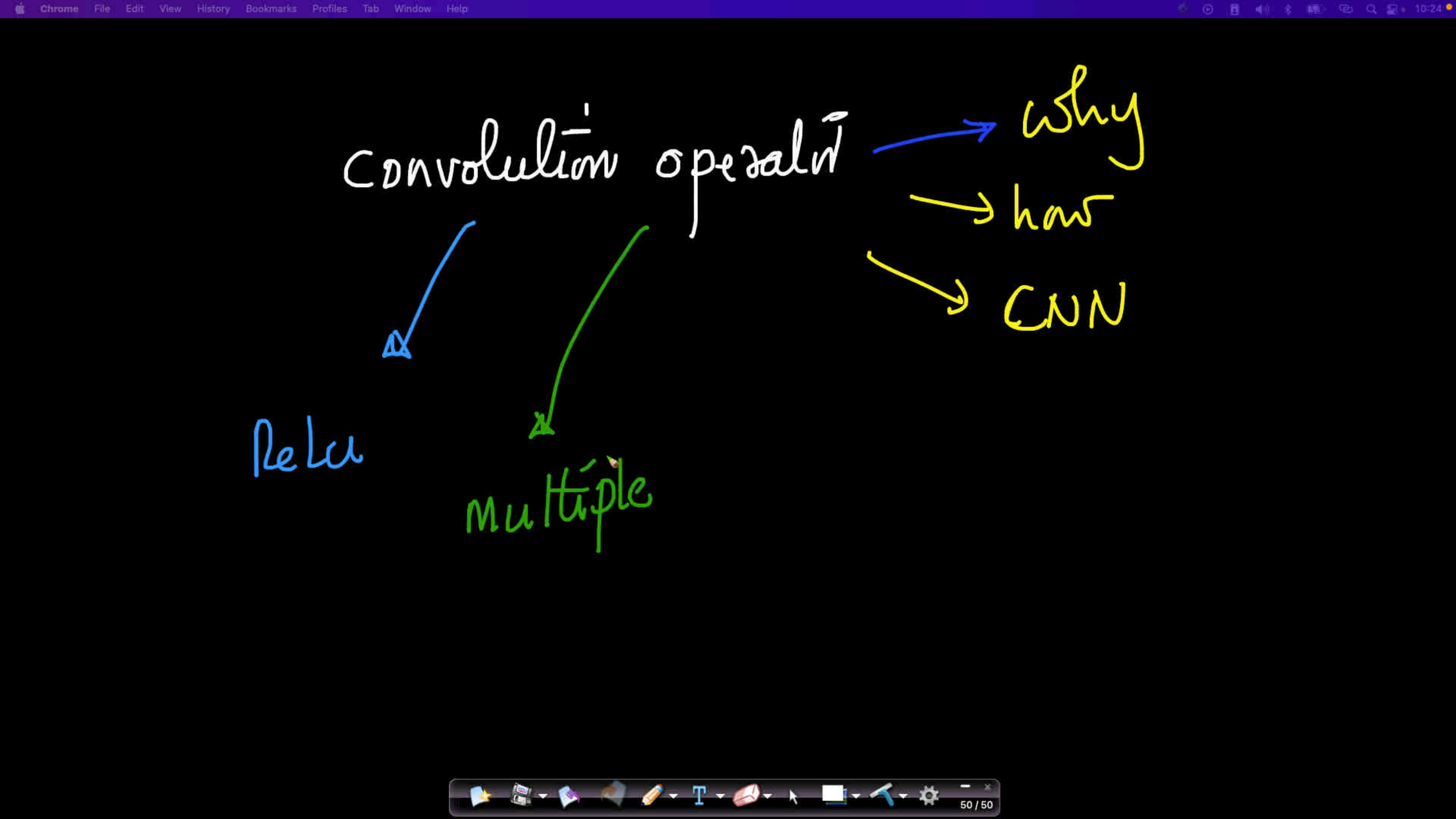


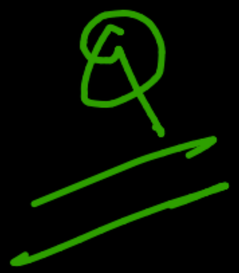

non-linearity
Zi
MLP: Relu(wT xi + b)
Max (0, Zi)
CNN: Relu(Im ⊙ K1)



Pixels → edges → shapes → objects

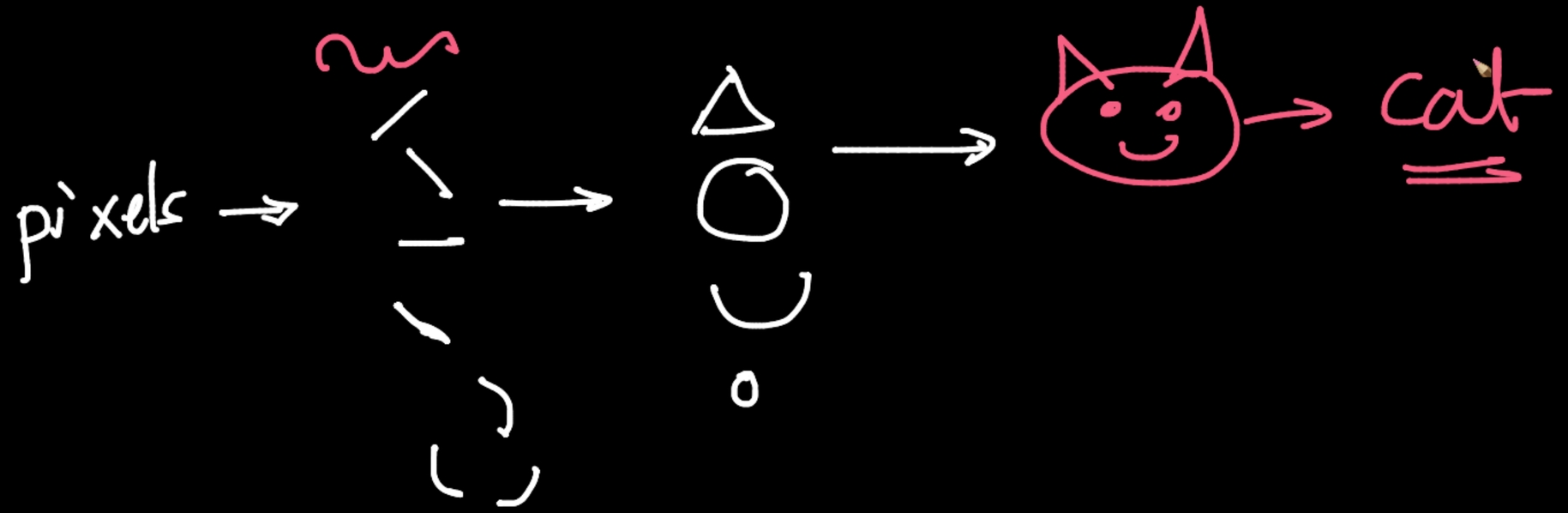

grayscale Image
hxw
3x3
3x3
3x3
Relu
Conv
5 Kernels
9x5=45
Tensor


Convolution operation
why
how
CNN
ReLu
Multiple



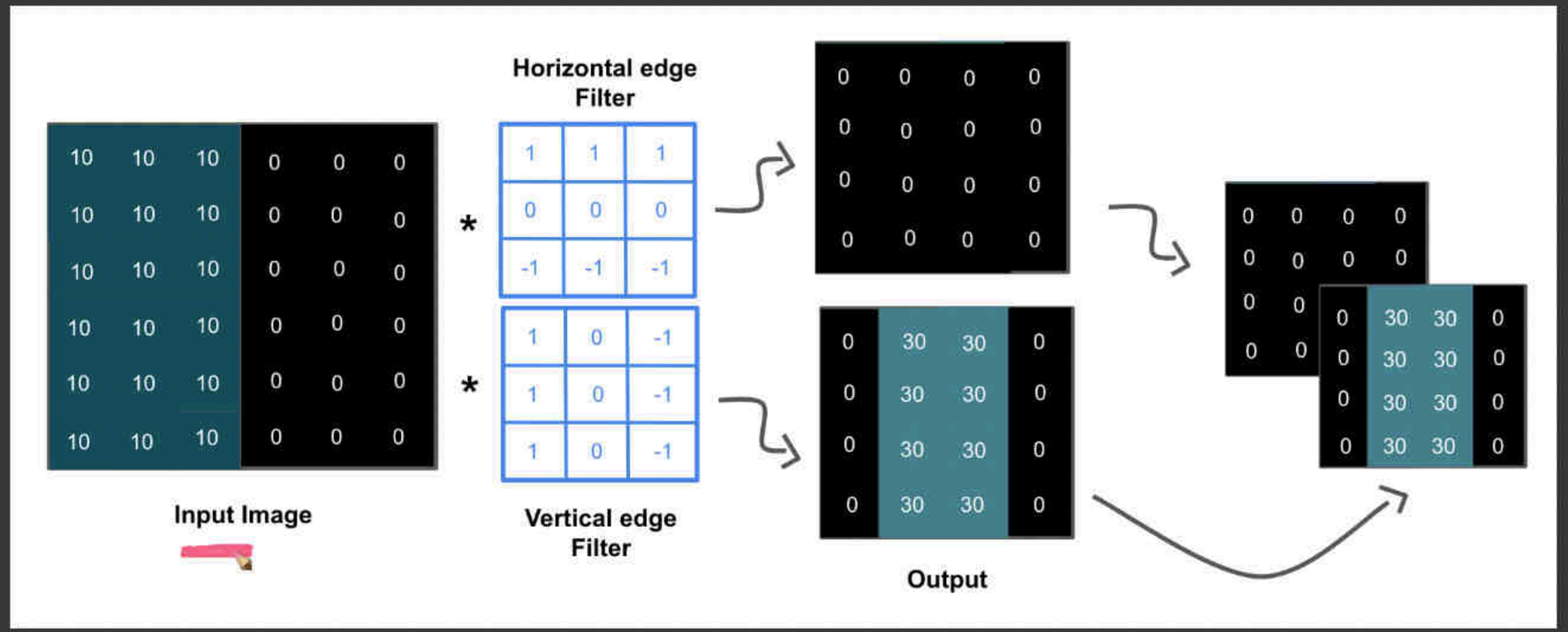
D_{TR} = 10k images → 5k: cat → 1

→ 5k: no cat → 0



Q. How can we apply both horizontal and vertical edge detector on the same image matrix?

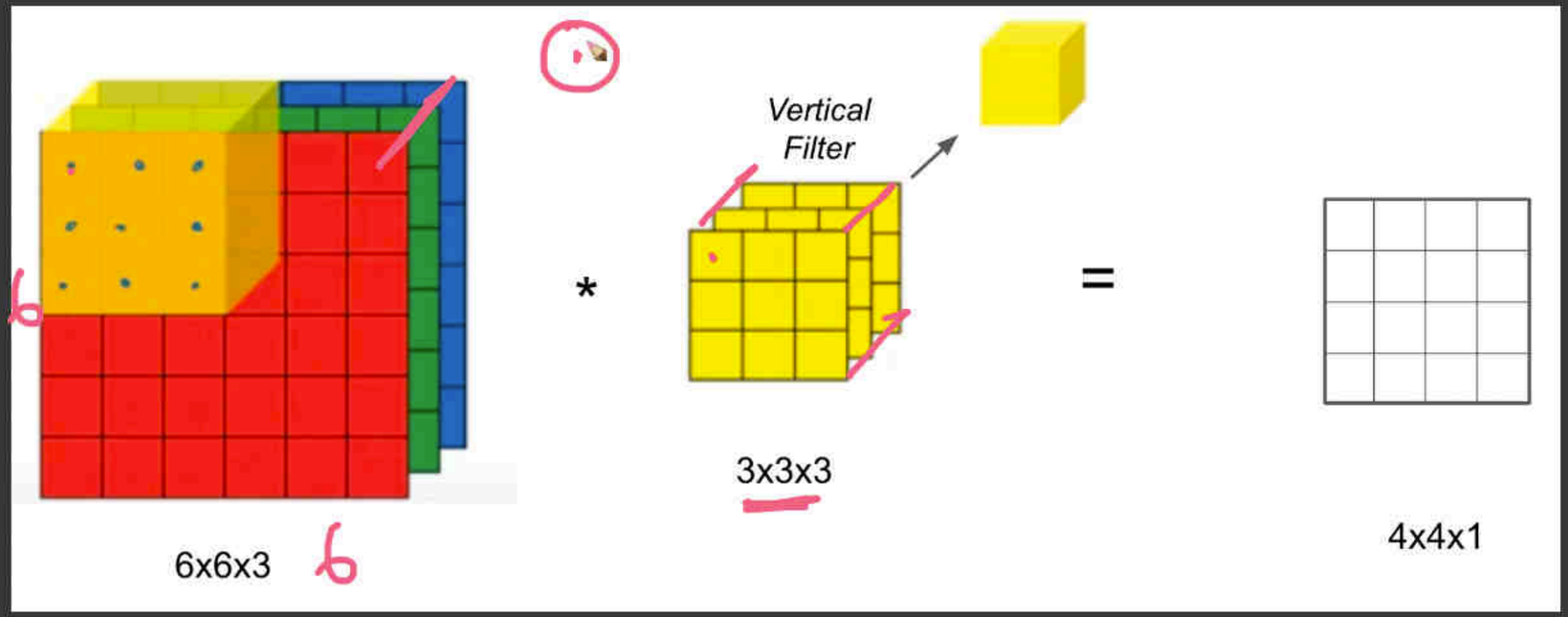
- By stacking Output from both the filters applied separately to the image!
- Notice that there is now a third Dimension to the Output Image, which is equal to the number of filters.



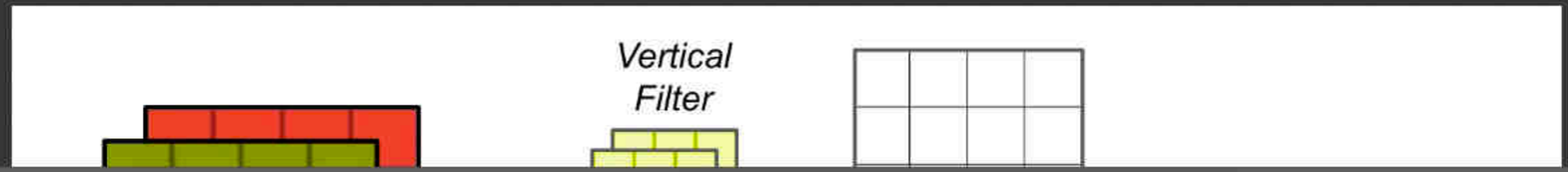
Q. What happens when we have a RGB Image as Input?

CNN!

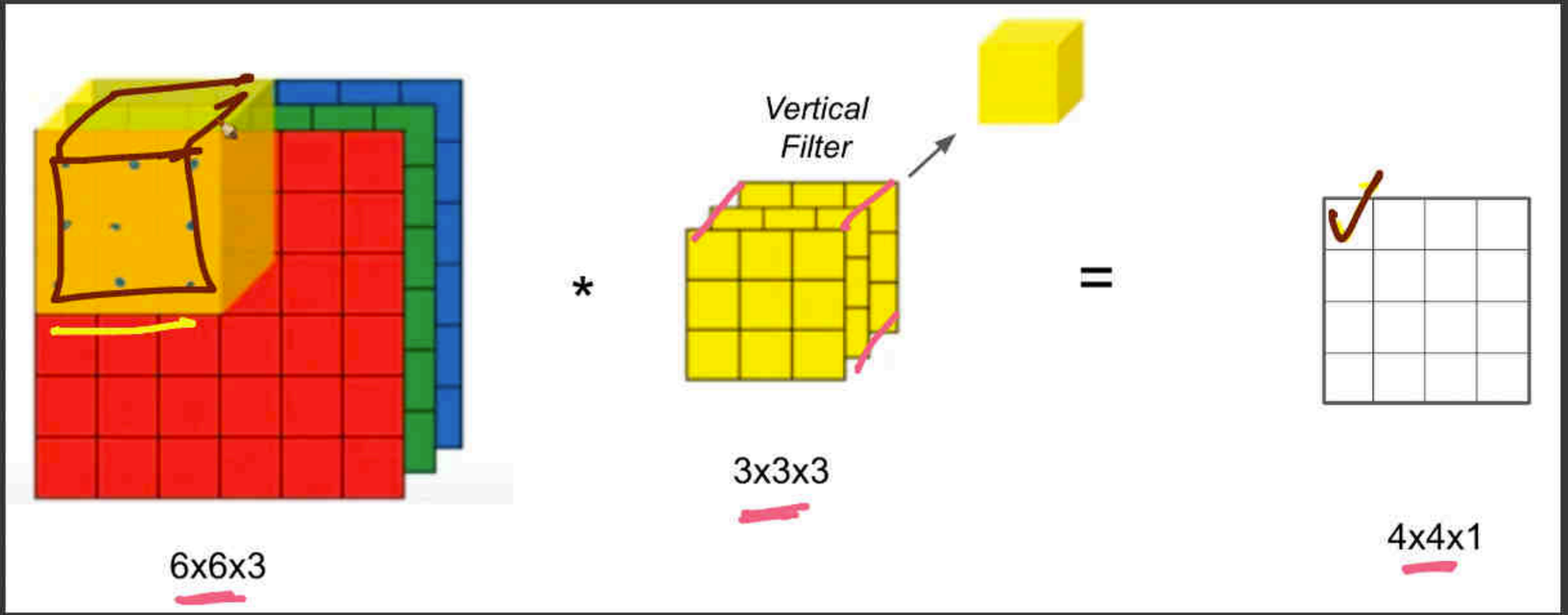
- If the Input Image has more than 1 channel, each Kernel will have a depth associated with it (which is equal to the number of channels in Input Image) and the Convolution will be performed over the entire **VOLUME** of the input image as show here:



- In the case of 2 filters, we get an output dimension of 4x4x2.

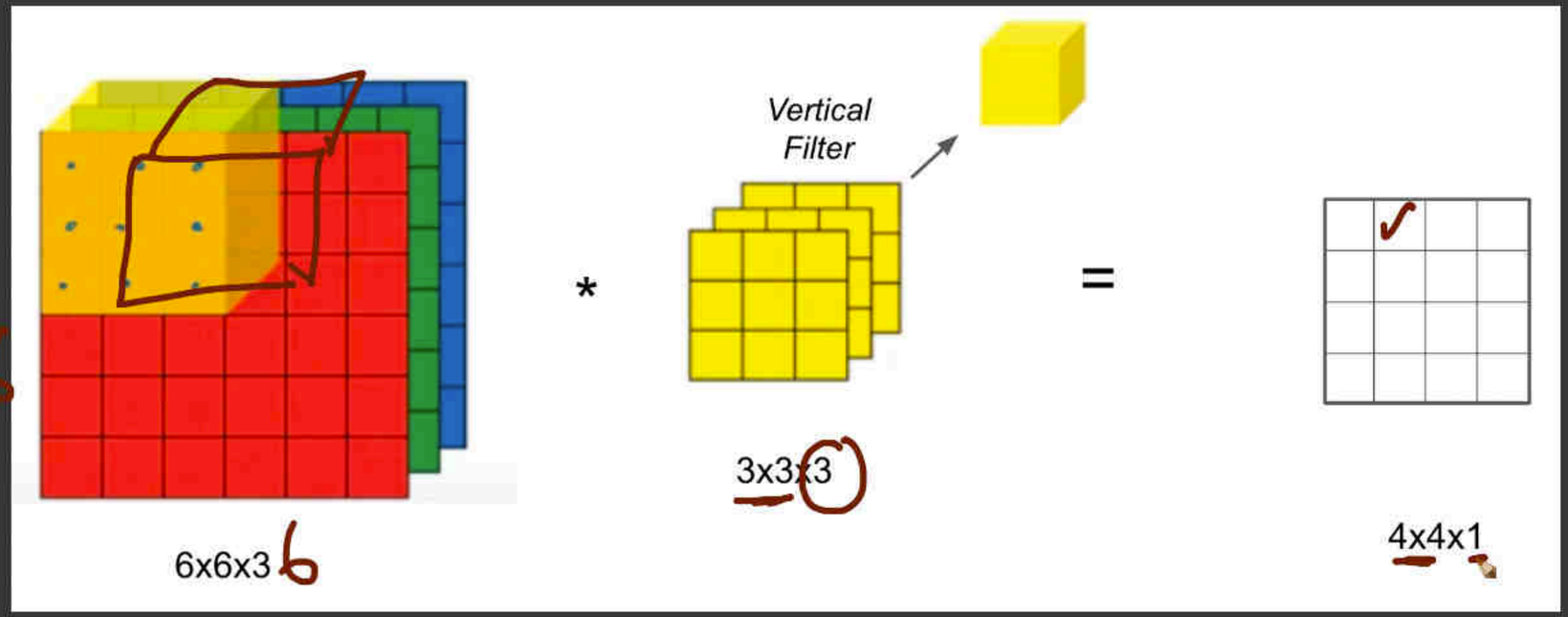


- If the Input Image has more than 1 channel, each Kernel will have a depth associated with it (which is equal to the number of channels in Input Image) and the Convolution will be performed over the entire **VOLUME** of the input image as show here:

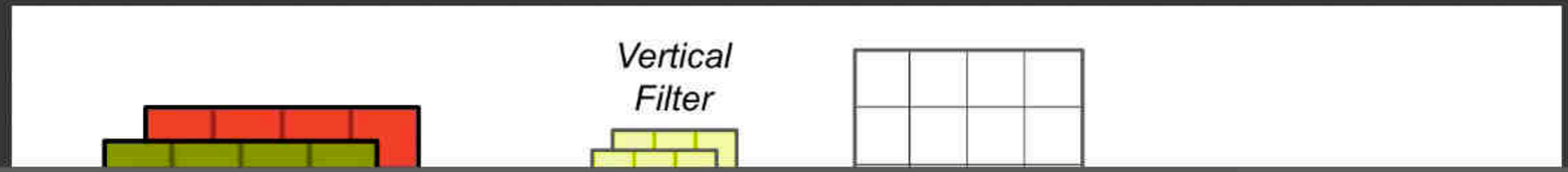


- In the case of 2 filters, we get an output dimension of 4x4x2.

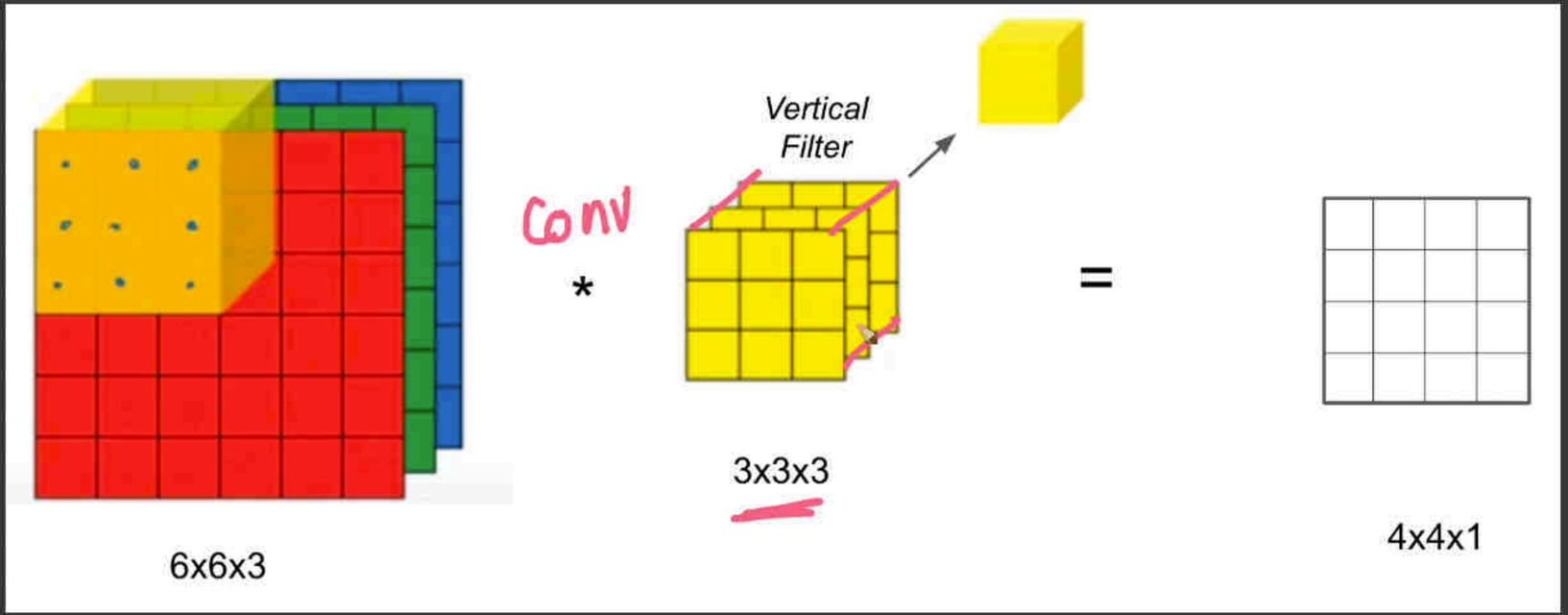
- If the Input Image has more than 1 channel, each Kernel will have a depth associated with it (which is equal to the number of channels in Input Image) and the Convolution will be performed over the entire **VOLUME** of the input image as show here:



- In the case of 2 filters, we get an output dimension of 4x4x2.



- If the Input Image has more than 1 channel, each Kernel will have a depth associated with it (which is equal to the number of channels in Input Image) and the Convolution will be performed over the entire **VOLUME** of the input image as show here:

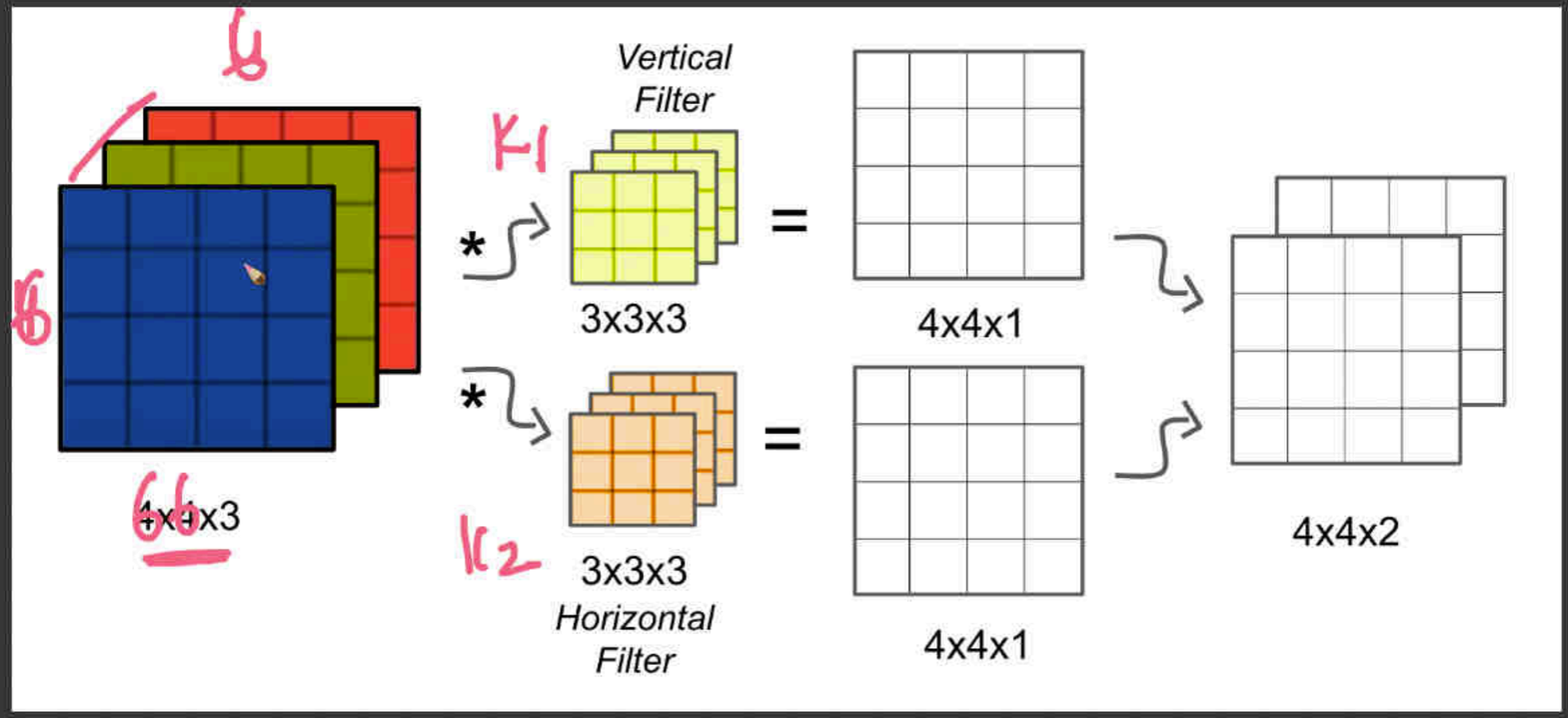


- In the case of 2 filters, we get an output dimension of 4x4x2.

6x6x3

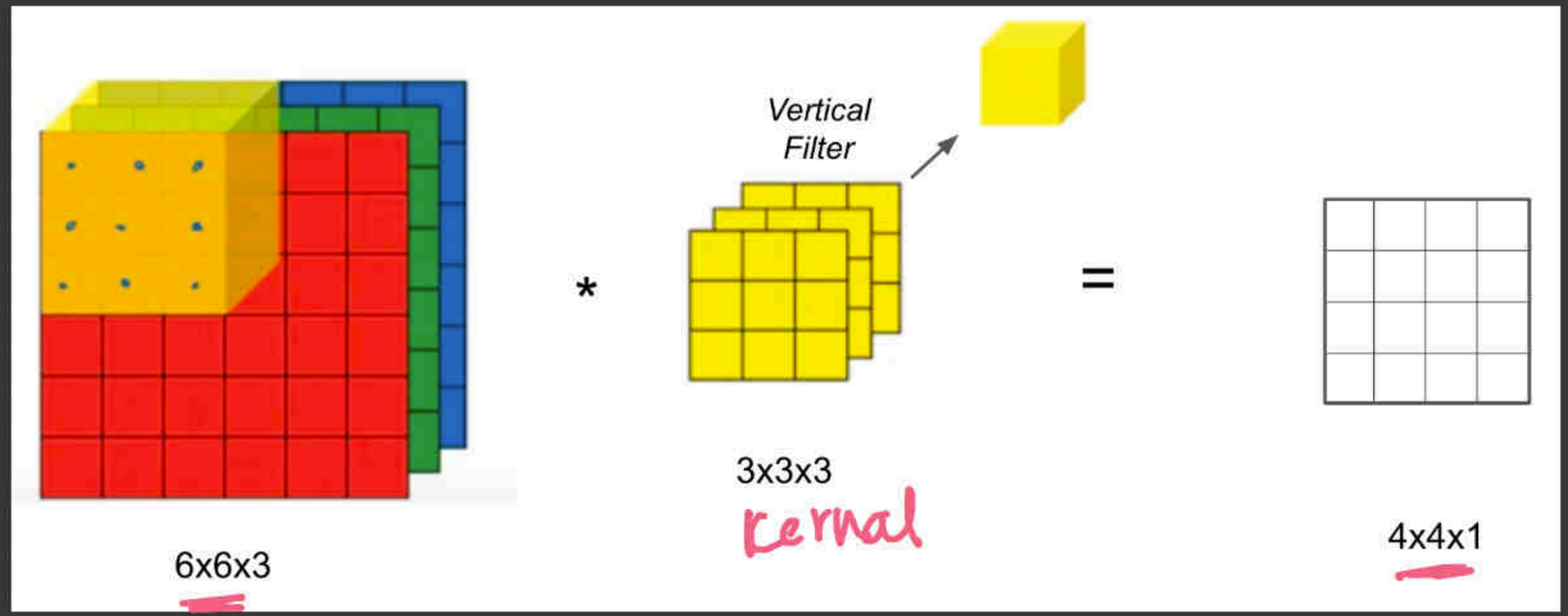
4x4x1

- In the case of 2 filters, we get an output dimension of 4x4x2.

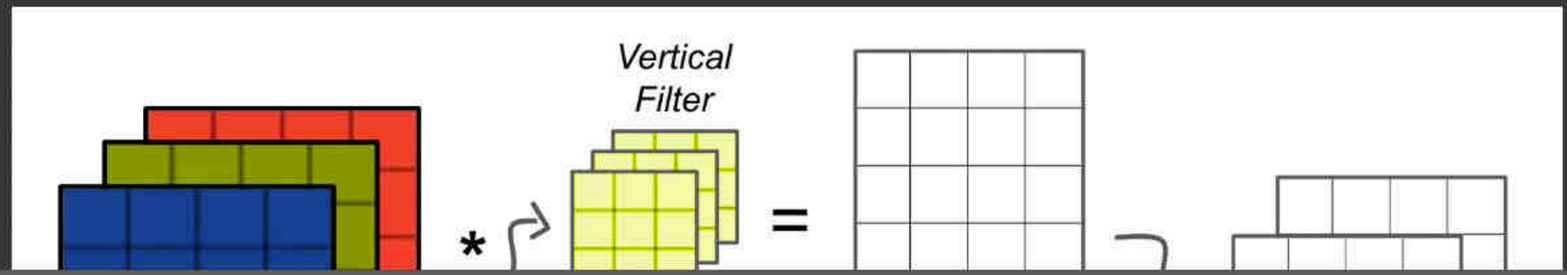


Q. Are these two Kernels enough to know?

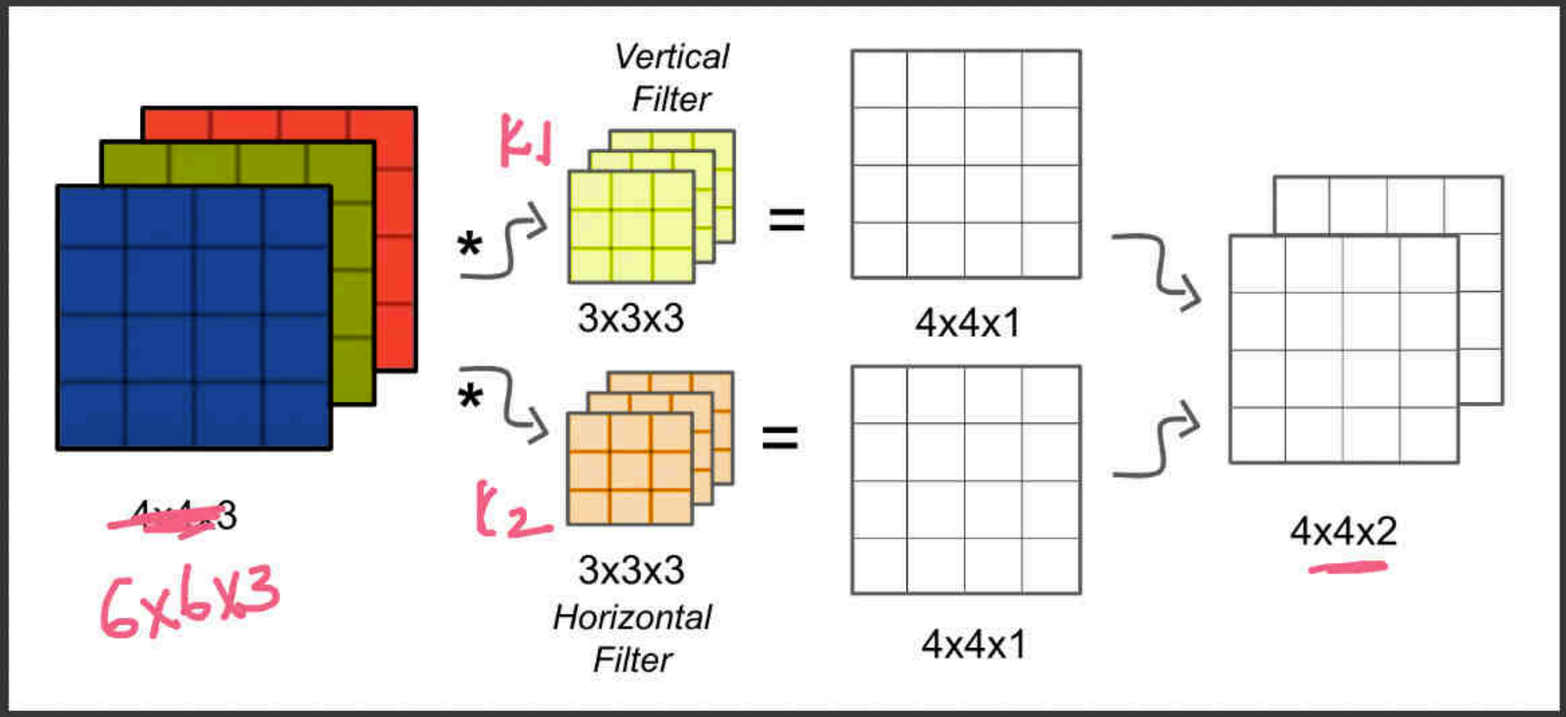
Input Image) and the Convolution will be performed over the entire **VOLUME** of the input image as show here:

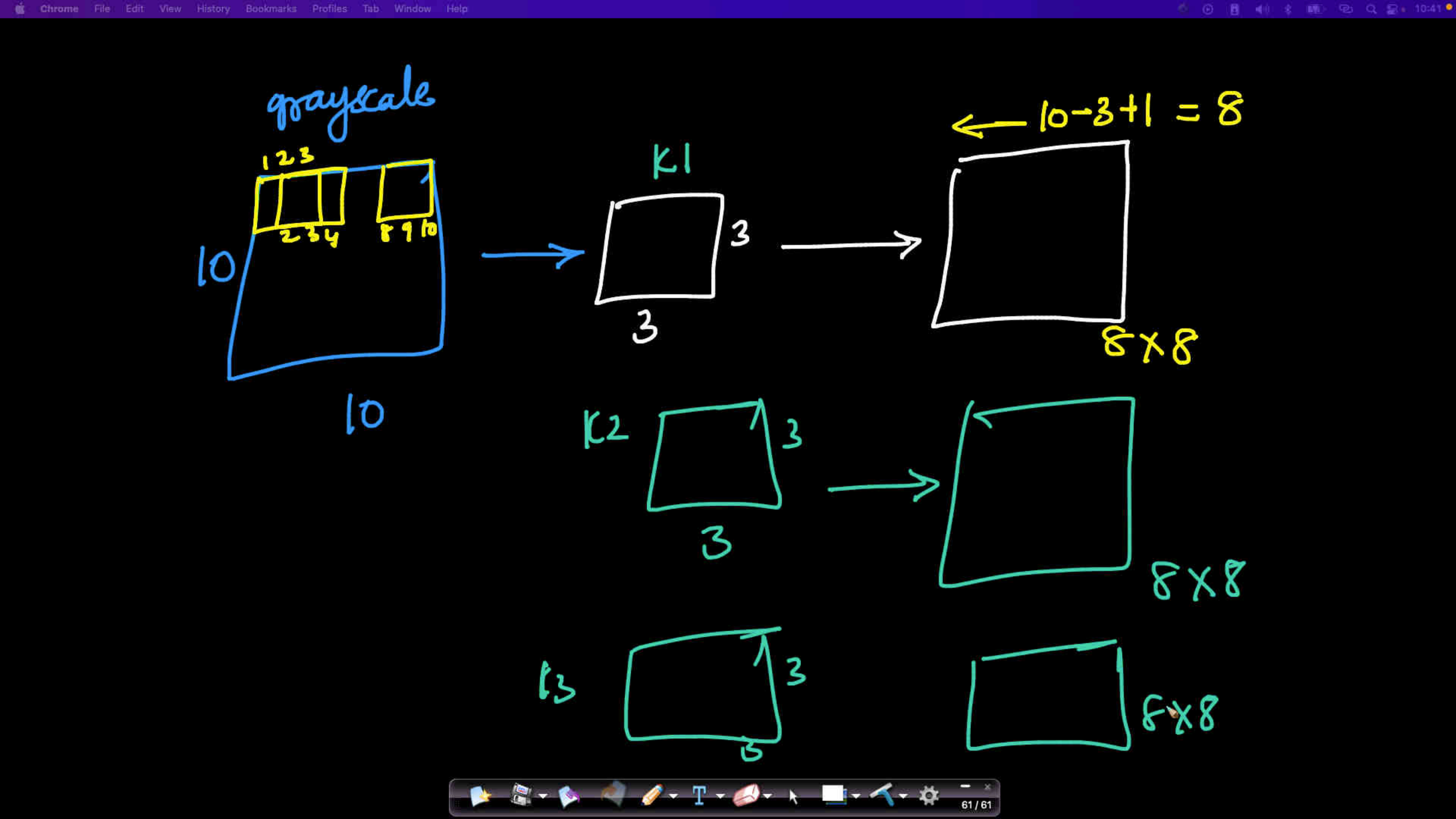


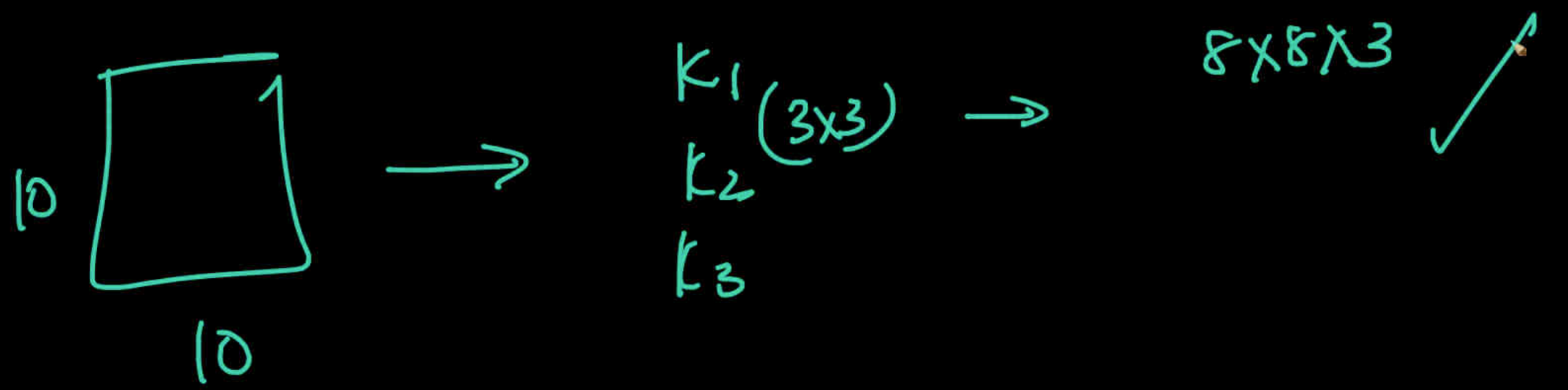
- In the case of 2 filters, we get an output dimension of 4x4x2.

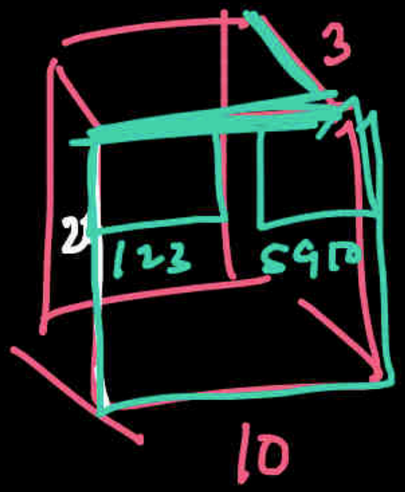


- In the case of 2 filters, we get an output dimension of 4x4x2.

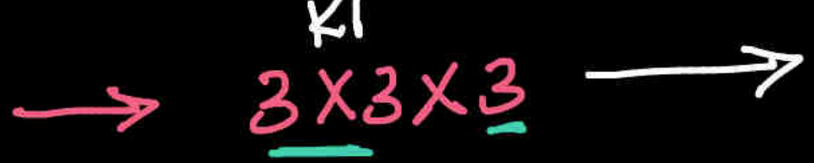


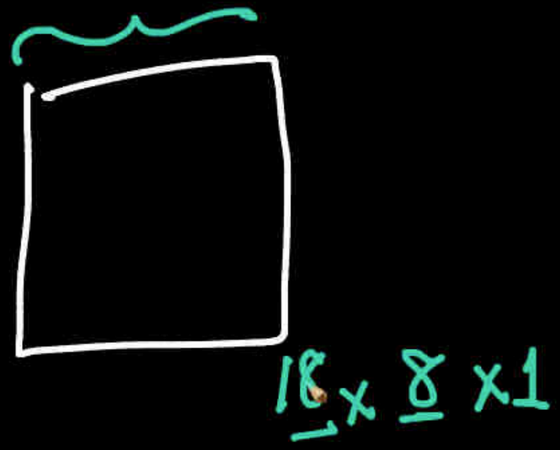

grayscale
1 2 3
2 3 4
8 9 10
10
10
K1
3
3
10-3+1 = 8
8x8
K2
3
3
8x8
K3
3
3
8x8

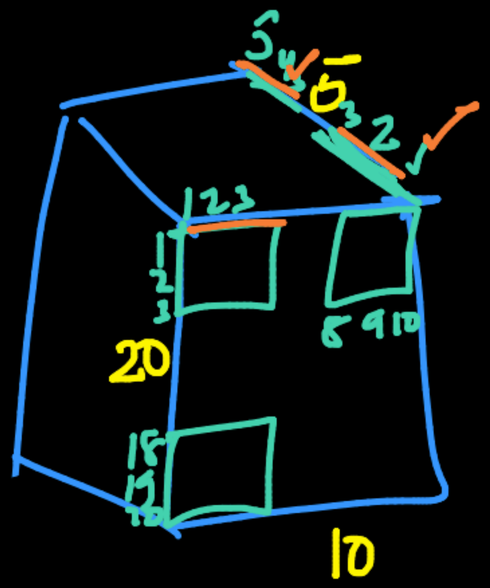

10
10
K1
K2
K3
(3x3)
8x8x3



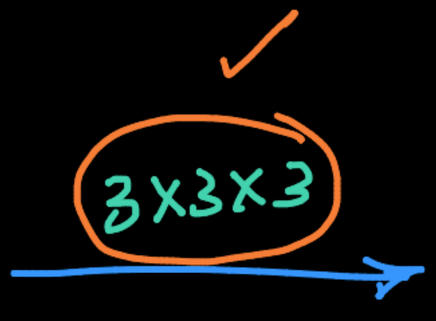
20 X 10 X 3

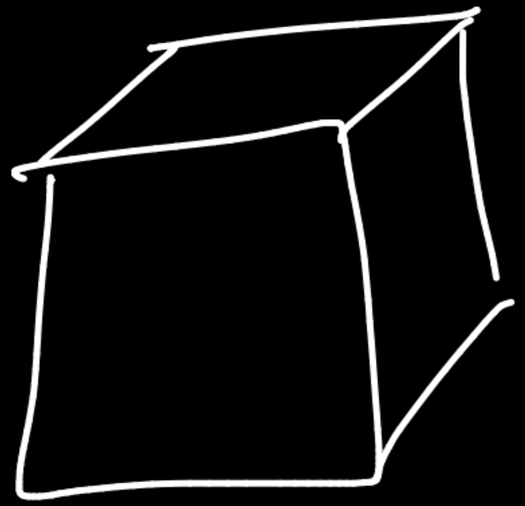






Input

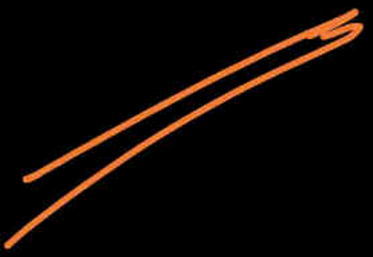




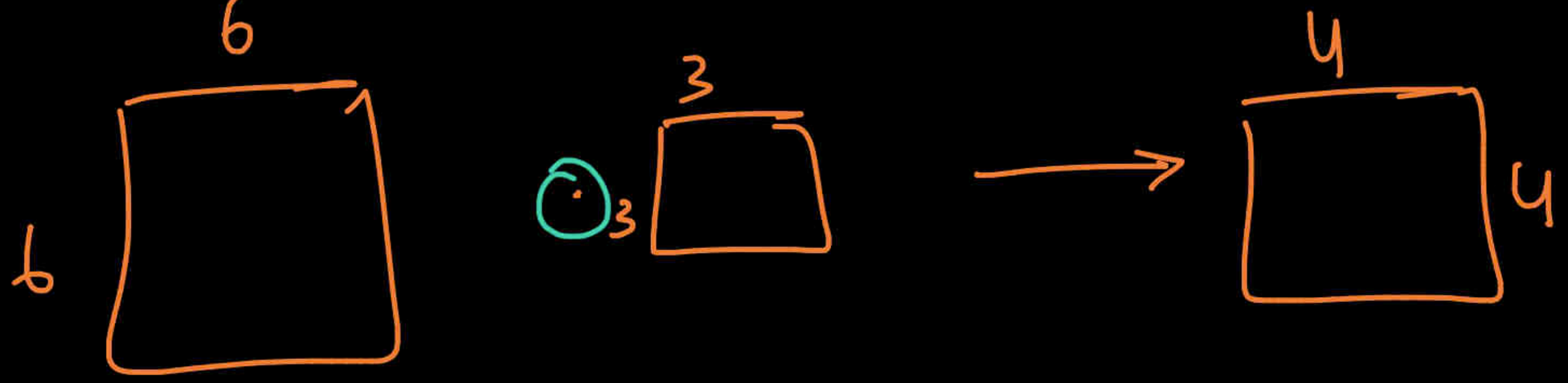
18 x 8 x 3

assignment → Conv

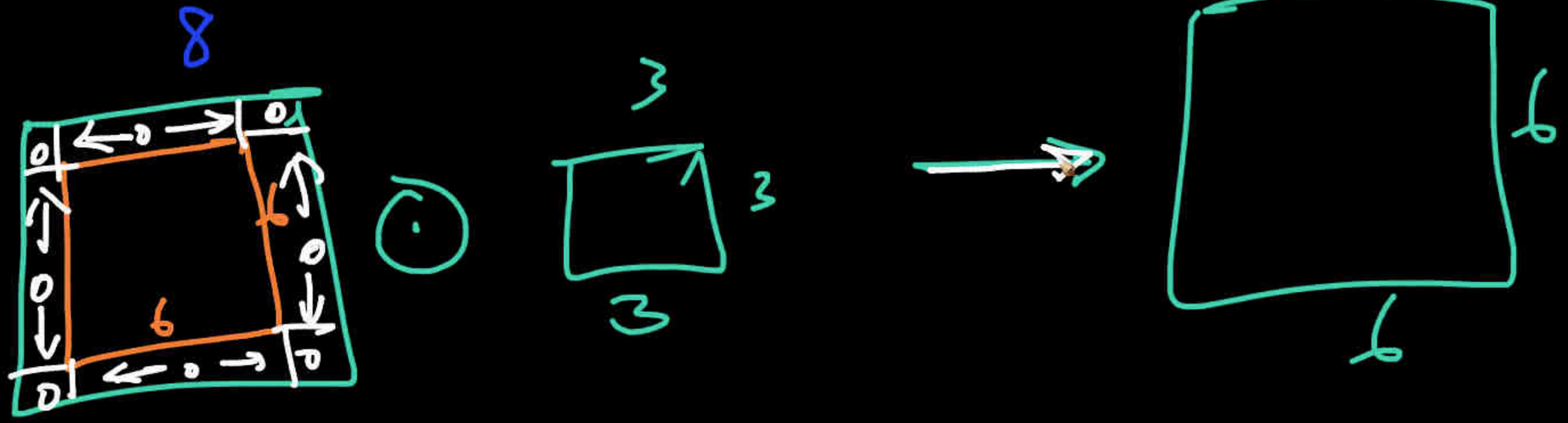
ndarrays

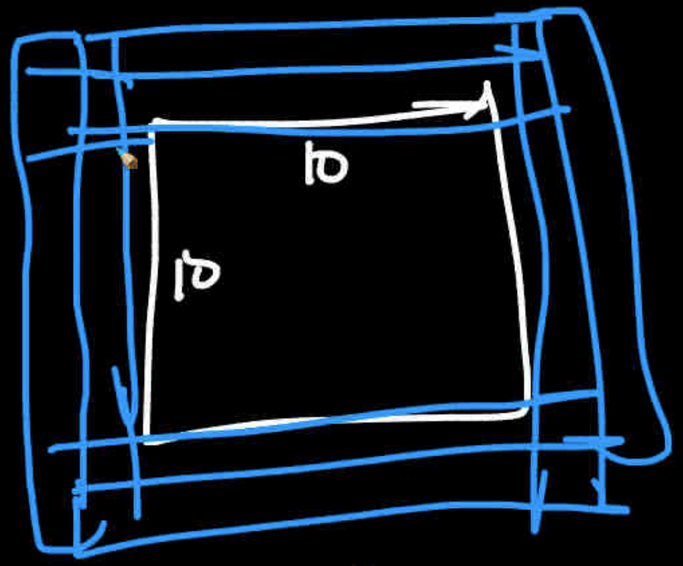


Padding



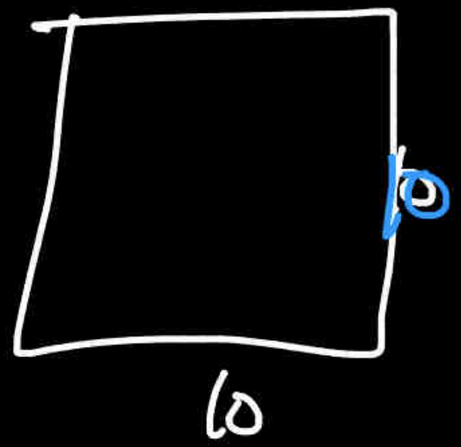
Padding

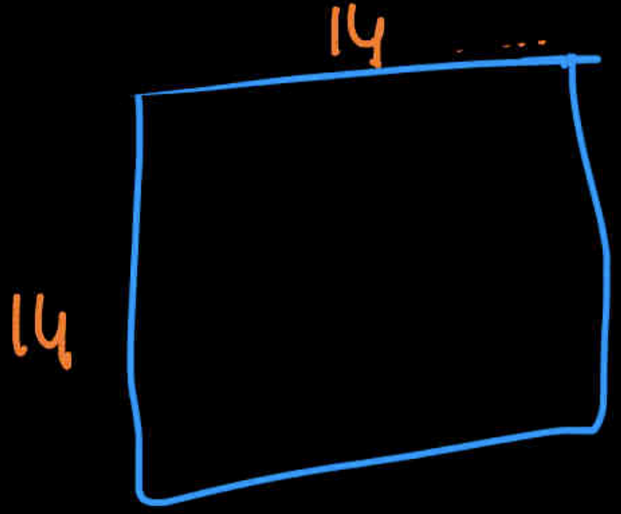




1

5 x 5

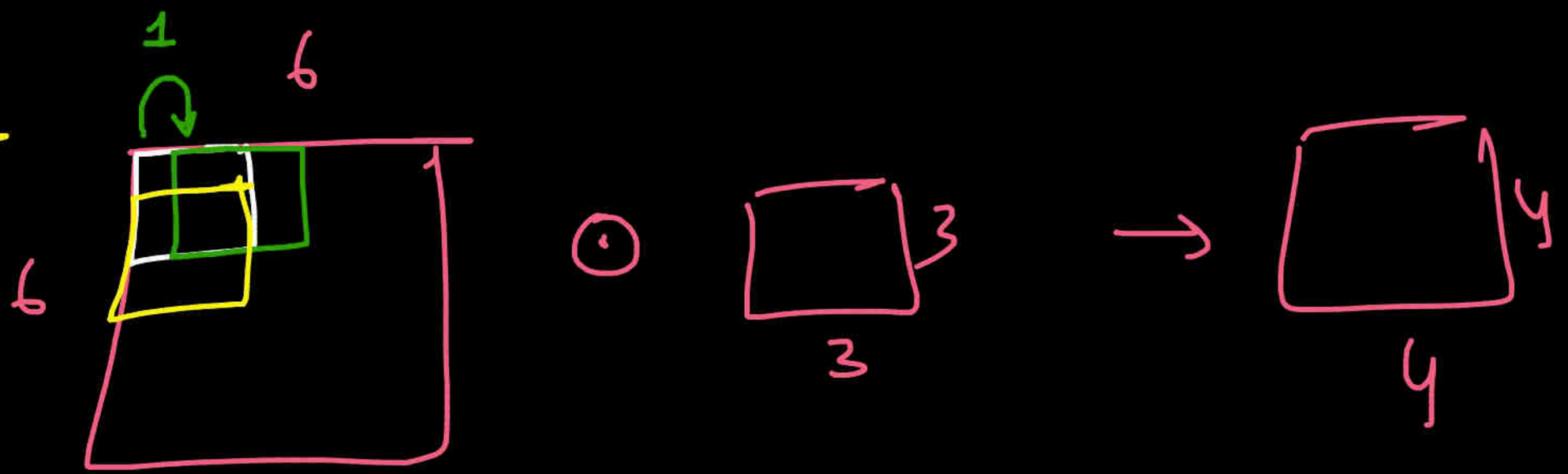


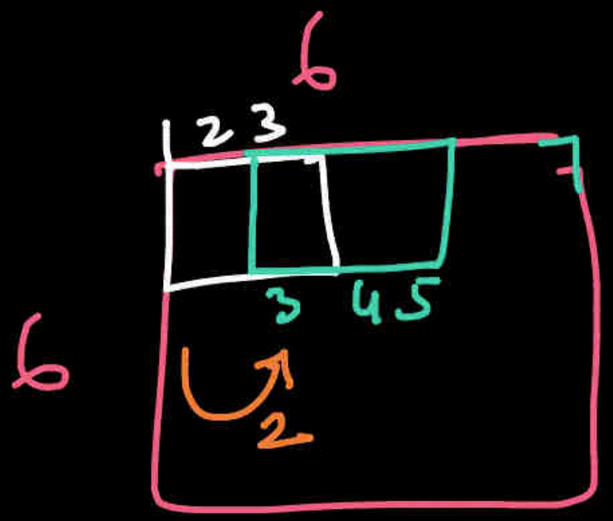


67 / 67

Stride:

default = 1

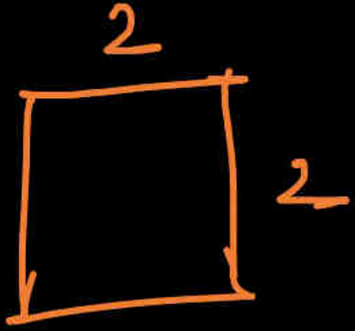




3 x 3

relu

stride = 2



smaller output

→ CV

→ Visual cortex inspiration (pixels → edges → shapes ↓ obj)

→ Classical IP → edge detection (Conv = cell-wise matrix mul + add)

→ learn Conv. layers (Relu, multiple)

→ padding, strides

Next

→ Pooling

→ appended → MLP

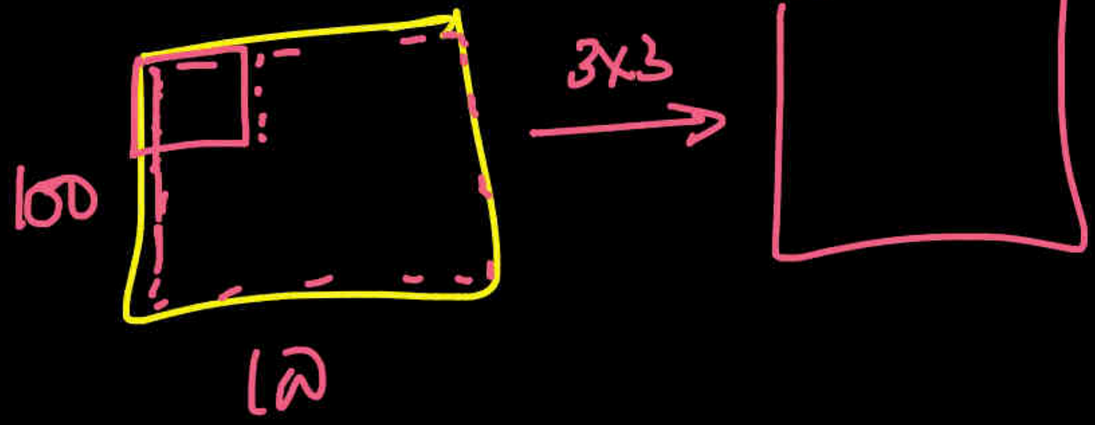
→ CNN

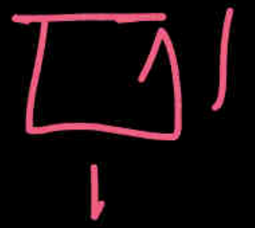
CNN:

Shape of Kernels
Kernels
stride
padding

} hyper-params

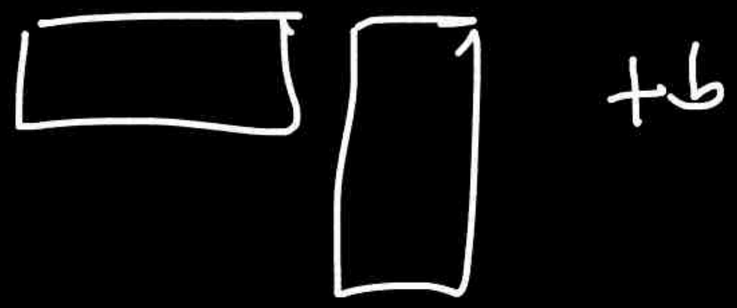
100 layers CNN

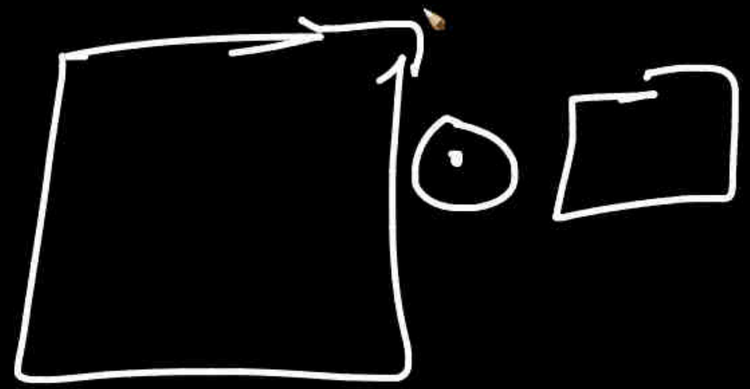


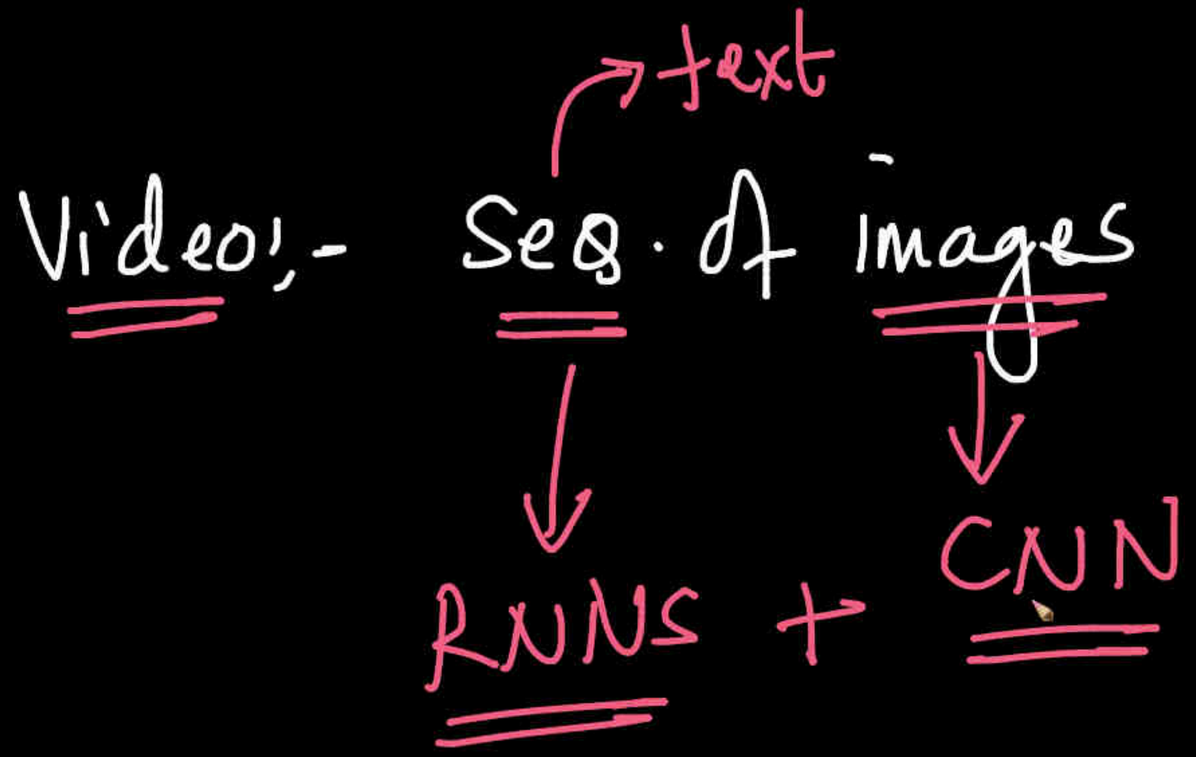


$w^T x_i + b$

$\text{Im} \odot K$






text
Video:- Seq. of images
RNNs + CNN

For faster generation and forthcoming API access you can try DreamStudio Beta

A dog made of origami. Highly detailed. Photograph. 4k. Colorful

Generate image

Advanced options

Share to community