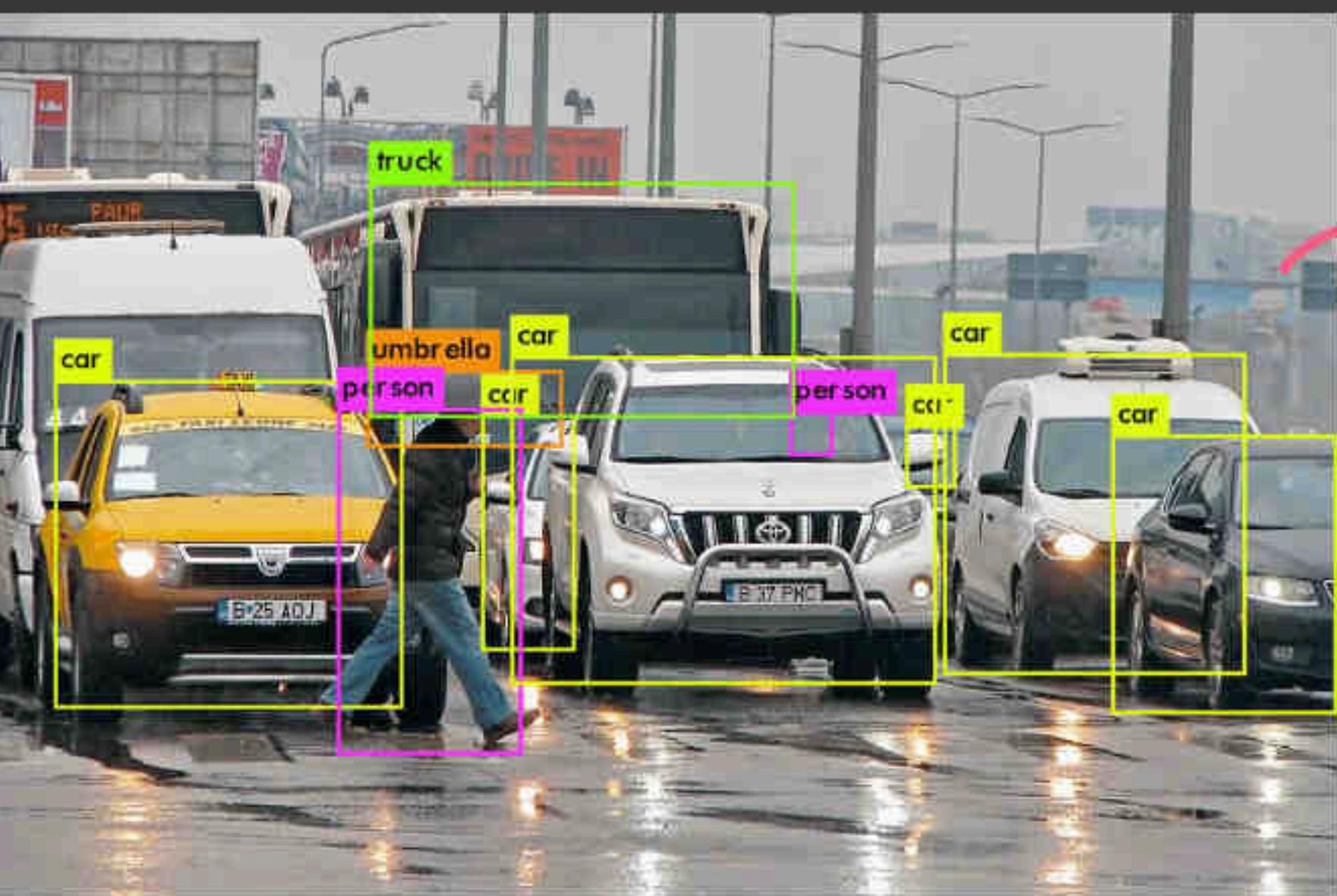


+ Code + Text Changes will not be saved

Connect



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



bounding box

- Extract **Text** from the Image -



L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=bVsL9QWM0Nb7

+ Code + Text Changes will not be saved

Connect

Changes will not be saved

• Extract **Text** from the Image -



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

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

CNN + RNN
(NLP)

3 / 4

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=bVsL9QWM0Nb7

+ Code + Text Changes will not be saved

Connect

Changes will not be saved

+ Code + Text

Changes will not be saved

Search icon

{x} icon

File icon

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

a train traveling down a track next to a forest.

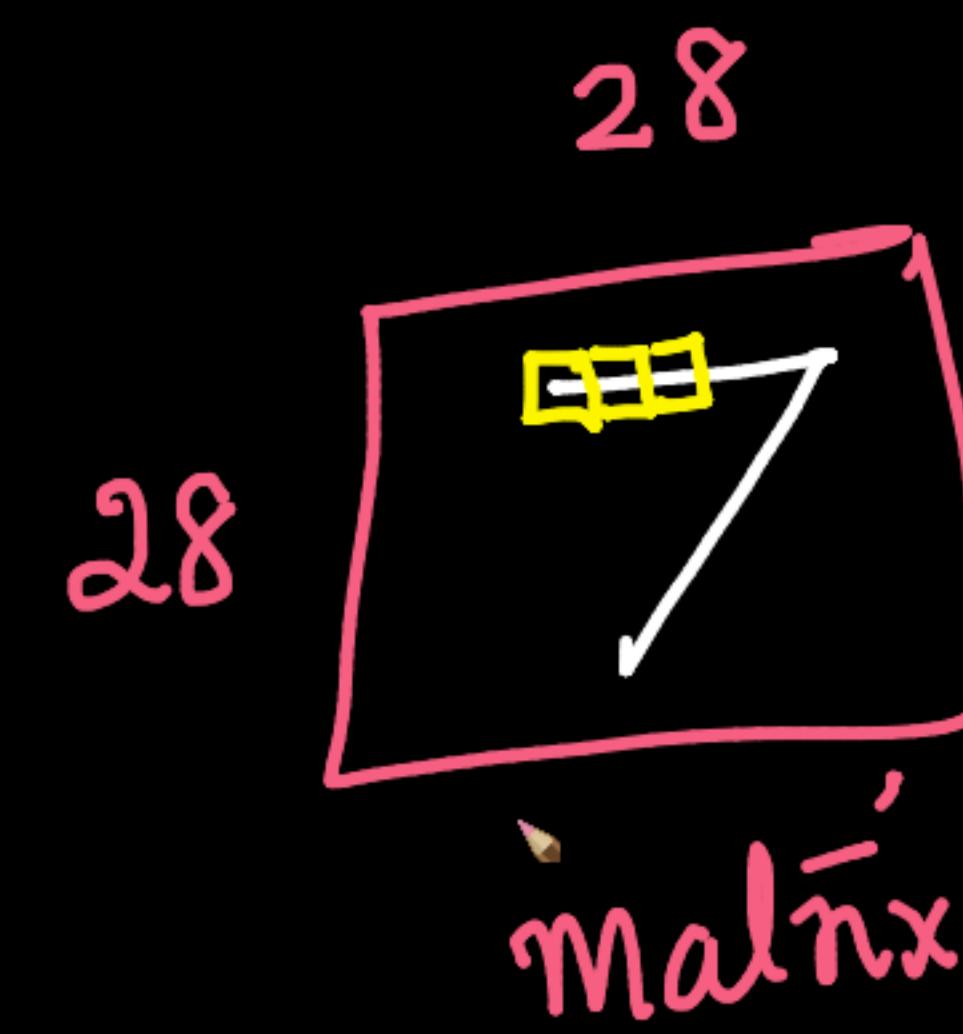


NLP

CNN

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

4 / 5



gray scale Image

$$\left\{ \begin{array}{l} 0 - 255 \\ B \quad W \end{array} \right.$$

8 bit numbers

$$\left\{ \begin{array}{l} 0 - 255 \\ W \quad B \end{array} \right.$$

$$\left\{ \begin{array}{l} 011\dots \\ \dots \\ \dots \end{array} \right.$$

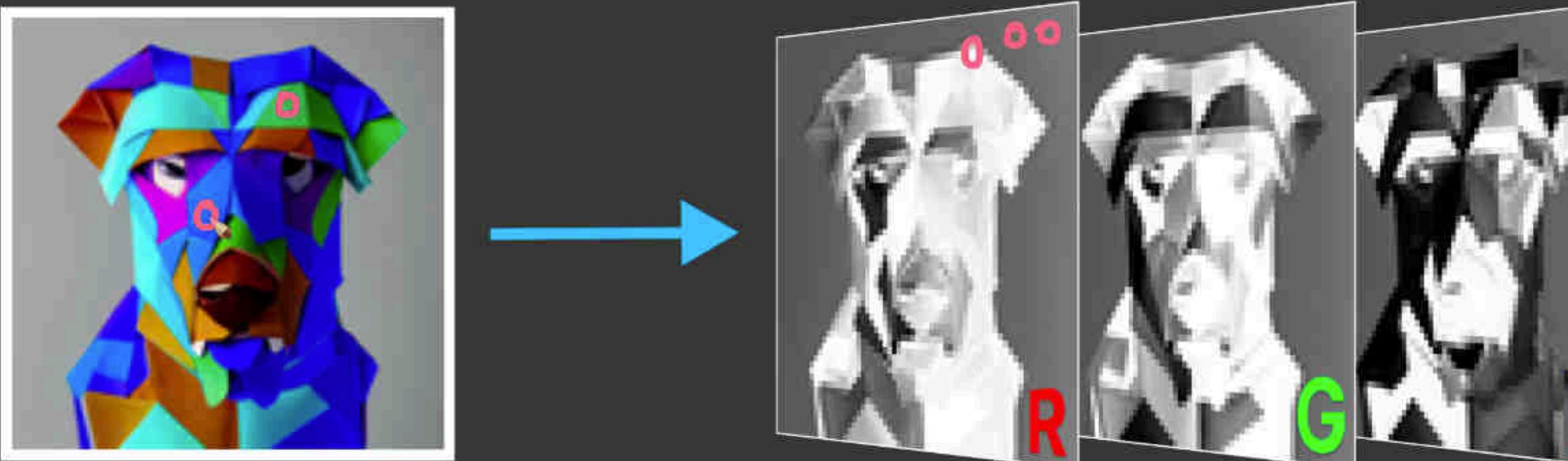
+ Code + Text Changes will not be saved

Connect



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.



0-255

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

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=bVsL9QWM0Nb7

+ Code + Text Changes will not be saved Connect |  

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

7/8

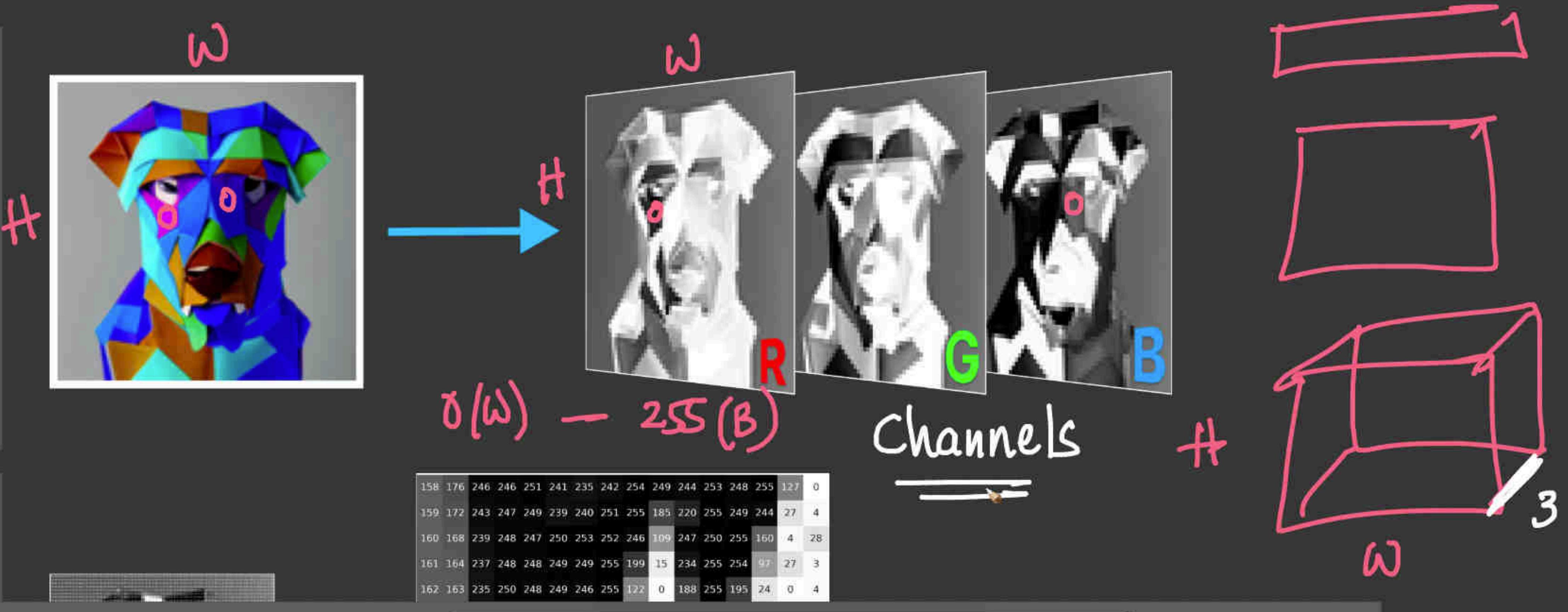
colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=bVsL9QWM0Nb7

+ Code + Text Changes will not be saved

Connect  

What a Computer actually sees?

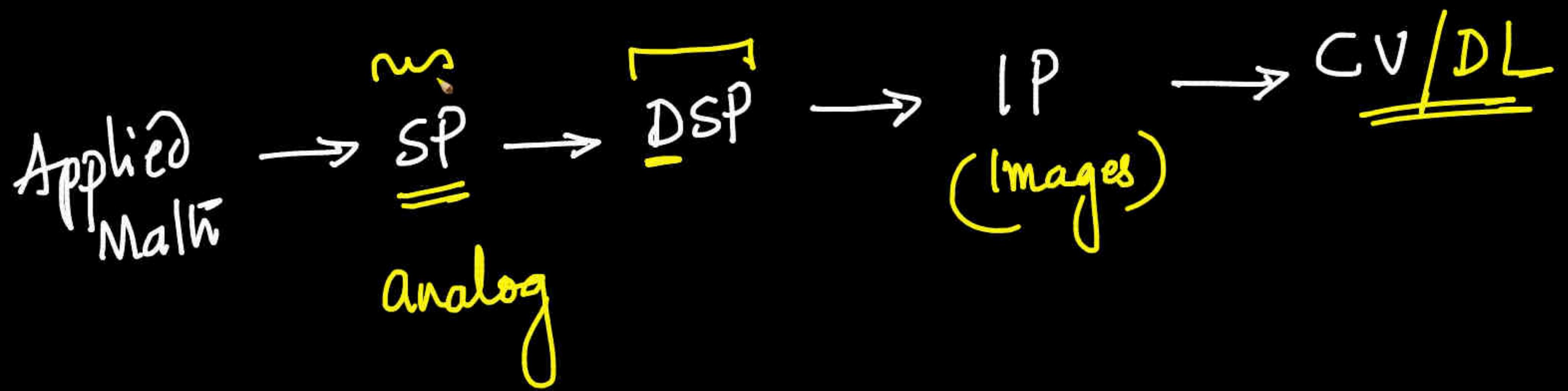
- A **Digital Image** is a matrix of size (H, W, C) , comprising of numbers (also known as pixel values) typically ranging from 0-255. Here H, W, C denote Height, Width and No. of Channels in a image.
- For a Grayscale image, # channels is 1, while for Colored Image its 3.



$\delta(w) = 255(B)$

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

8 / 8

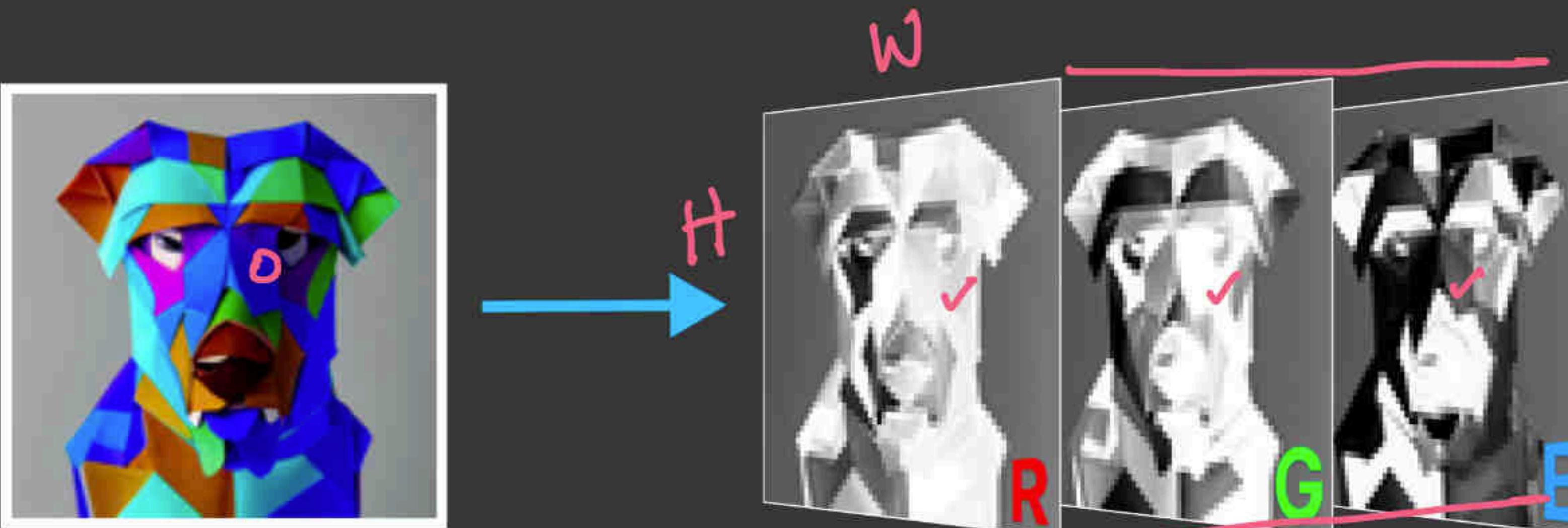


+ Code + Text Changes will not be saved

Connect  

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.



ndarray
($H, W, 3$)

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

L1: Intro to Computer Vision(C) origami - Google Search Stable Diffusion - a Hugging Face colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=bVsL9QWM0Nb7

+ Code + Text Changes will not be saved

Connect

Changes will not be saved

{x}

Q Is CV Easy?

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

11 / 11

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=bVsL9QWM0Nb7

+ Code + Text Changes will not be saved

Connect

≡ OCCULTATION

🔍

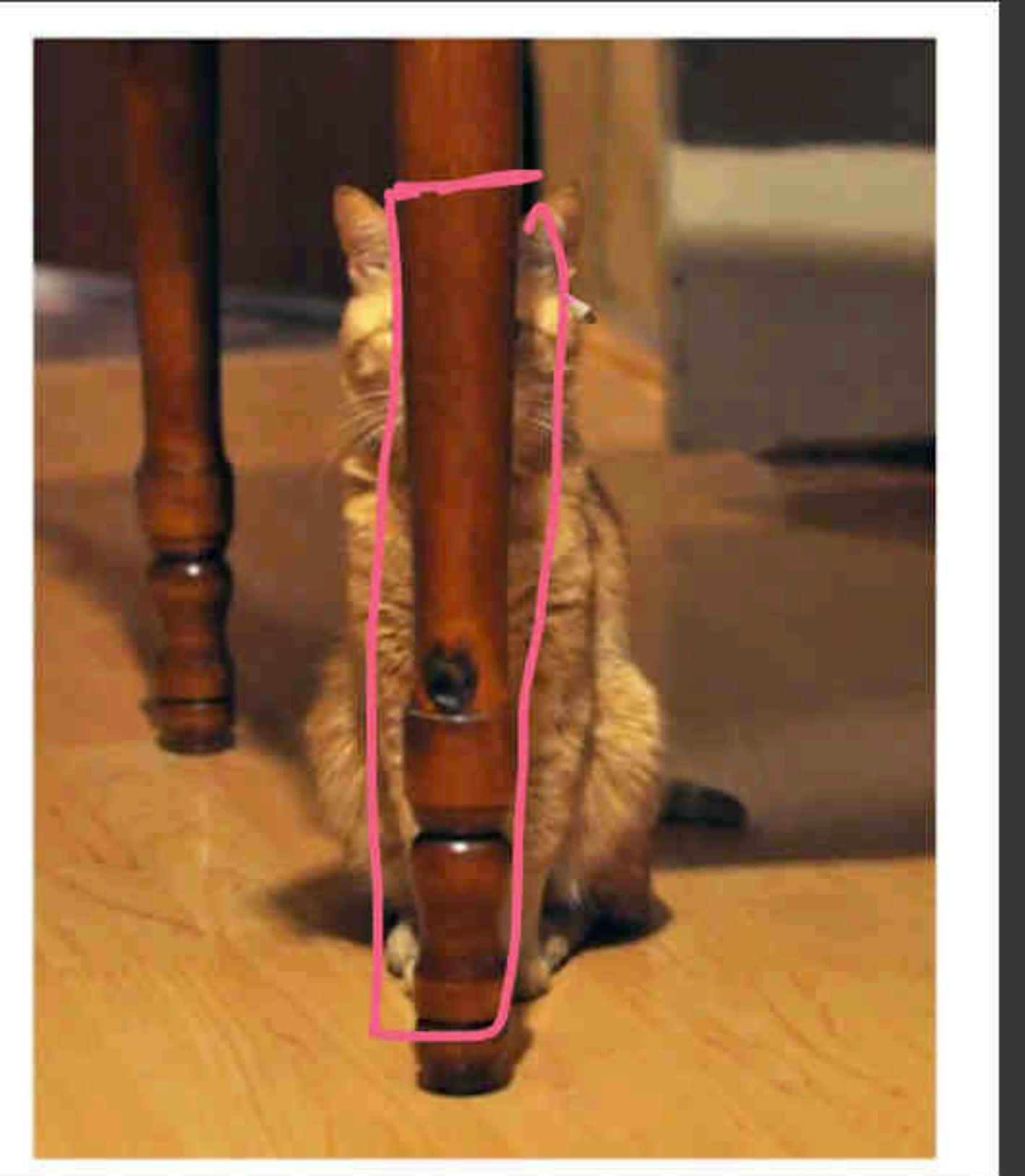
{x}

📁

• Illumination variability

◀ ▶

12 / 12

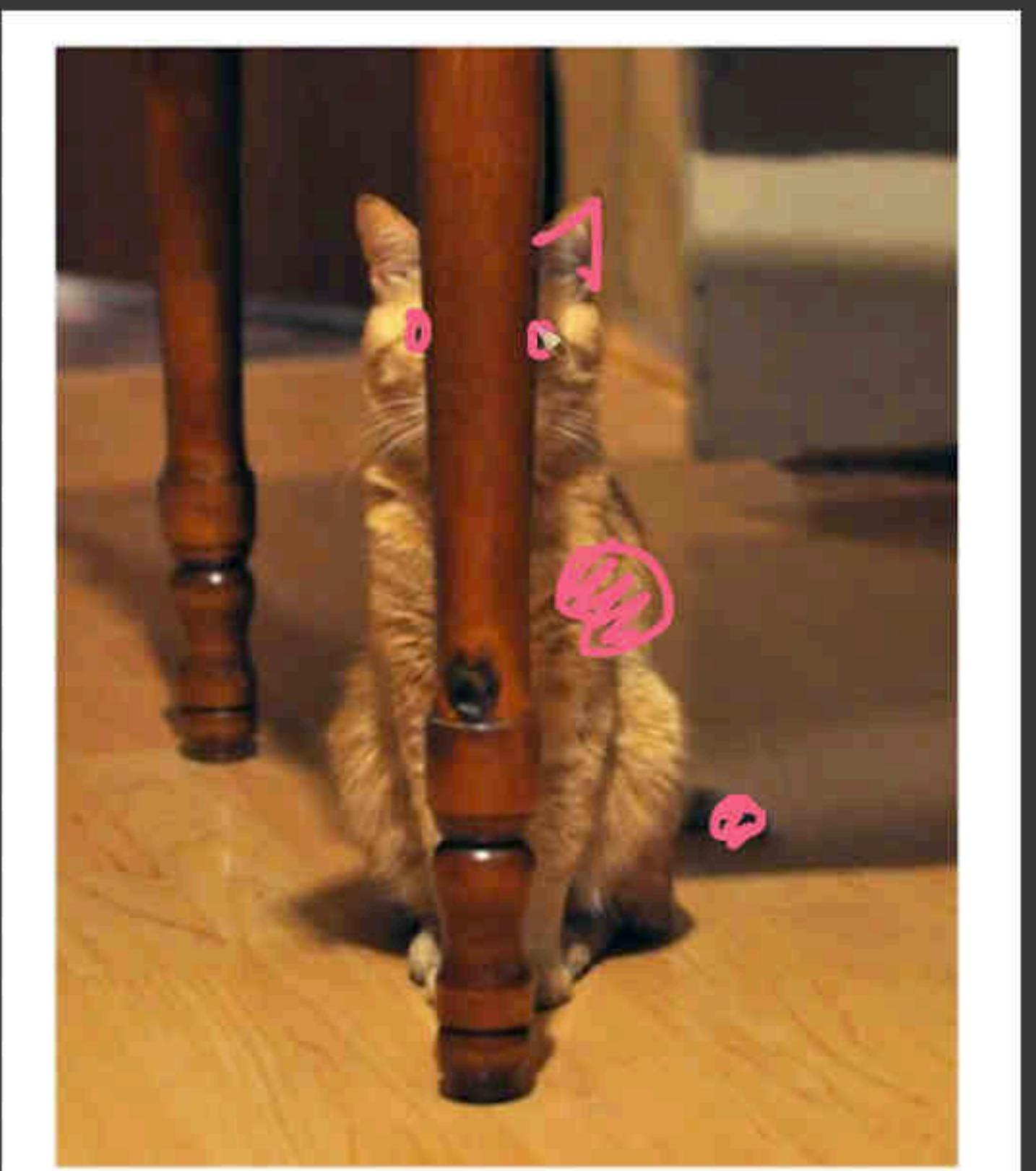


Cat
Table

The image shows a ginger cat standing on a light-colored wooden floor, leaning its front paws against a dark brown wooden chair leg. A pink rectangular box highlights the area where the cat's body meets the chair leg, specifically the right side of the cat's torso and the left side of the chair leg. This highlights a point of occlusion in the scene. To the right of the image, the word "Cat" is written in pink cursive, and below it, the word "Table" is also written in pink cursive.

+ Code + Text Changes will not be saved

Connect



- Illumination variability



L1: intro to ComputerVision(C) x origami - Google Search x Stable Diffusion - a Hugging Face Model x Colors RGB x +

w3schools.com/colors/colors_rgb.asp

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP BOOTSTRAP HOW TO W3.CSS C C++ C# REACT R JQUERY DJANGO TYPESCRIPT

Colors Tutorial

Colors HOME Color Names Color Values Color Groups Color Shades Color Picker Color Mixer Color Converter

Color RGB

Color HEX Color HSL Color HWB Color CMYK Color NCol Color Gradient Color Theory Color Wheels Color currentcolor Color Hues Color Schemes

Color Palettes Color Brands Color W3.CSS Color Metro UI Color Win8 Color Flat UI Color Psychology

Color Schemes

Colors Monochromatic

Colors RGB

Colors HSL

Colors CMYK

Colors Hex

Colors NColor

Colors Gradient

Colors Theory

Colors Wheels

Colors CurrentColor

Colors Hues

Colors Schemes

Colors Palettes

Colors Brands

Colors W3.CSS

Colors Metro UI

Colors Win8

Colors Flat UI

Colors Psychology

Colors Schemes

Colors Monochromatic

Colors RGB

◀ Previous Next ▶

RGB Calculator

rgb(255, 255, 0)
#FFFF00
hsl(60, 100%, 50%)

R: 255 G: 255 B: 0

Use this color in our Color Picker

STAY 5 NIGHTS PAY FOR 3 GAME ON DUBAI

We just launched W3Schools videos

NEW

My First Reading

Waiting for www.google.com...

Colors Tutorial

Colors HOME

Color Names

Color Values

Color Groups

Color Shades

Color Picker

Color Mixer

Color Converter

Color RGB

Color HEX

Color HSL

Color HWB

Color CMYK

Color NCol

Color Gradient

Color Theory

Color Wheels

Color currentcolor

Color Hues

Color Schemes

Color Palettes

Color Brands

Color W3.CSS

Color Metro UI

Color Win8

Color Flat UI

Color Psychology

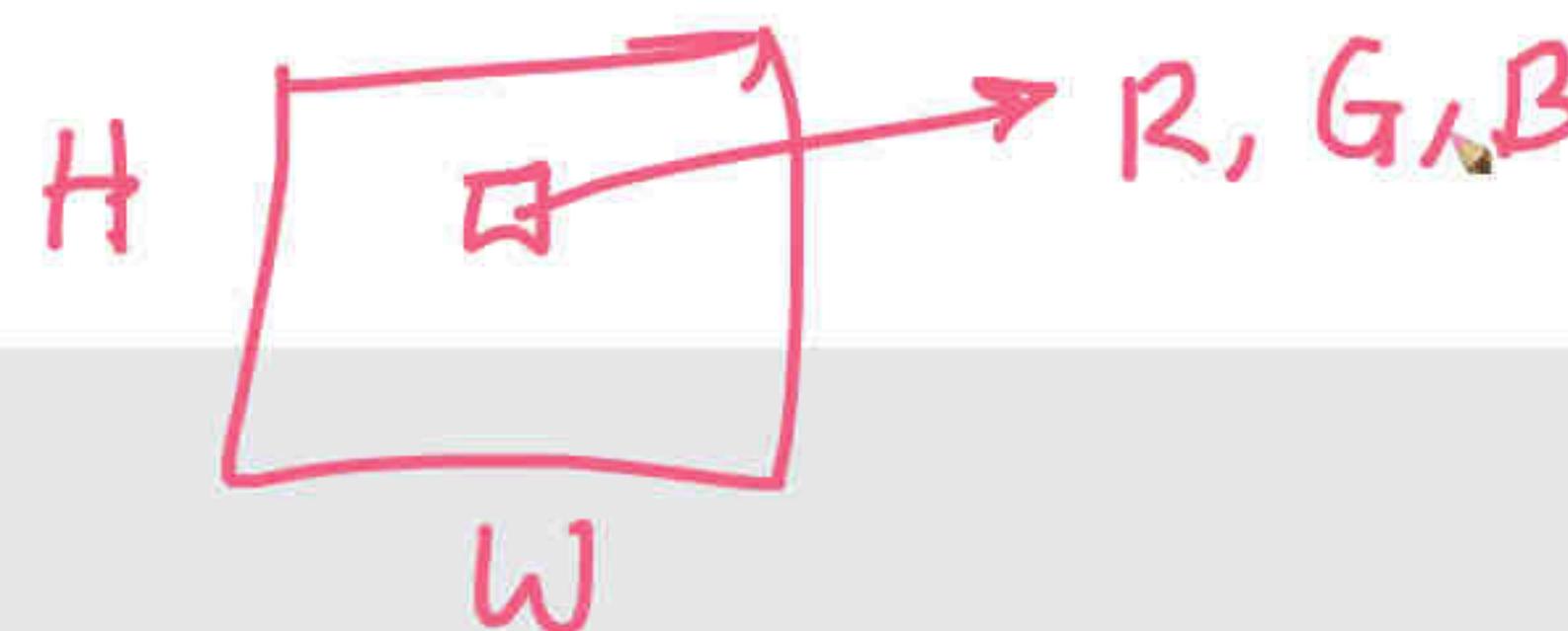
Color Schemes

Colors Monochromatic

Colors Analogous

Colors RGB

< Previous



Next >

RGB Calculator



rgb(0, 255, 255)

#00ffff

hsl(180, 100%, 50%)

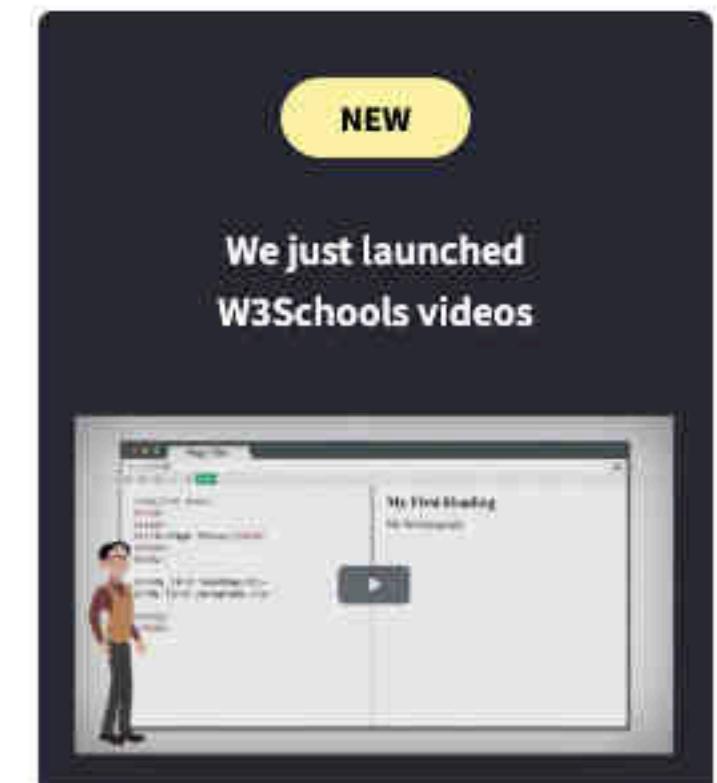
R: 0

G: 255

B: 255

255

255

[Use this color in our Color Picker](#)

CNNs → late 1990's

↳ Yann LeCun

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face Colors RGB colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=bVsL9QWM0Nb7

+ Code + Text Changes will not be saved

Connect

Changes will not be saved

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.

Visual Cortex

(Its Structure is Instructive and Inspiring)

The diagram illustrates the visual pathway from the eye to the brain. Light enters the eye and is processed by the Retina. The information then flows through several layers of the visual cortex: Layer 1, Layer 2, and Layer L. Layer L is shown as a network of nodes, some of which are labeled with names like "Tom", "Bill", "Jane", "Fred", "Fran", "Sue", "Drew S", "male", "female", "nice", "mean", "friend", and "cute". This represents how the visual system forms increasingly complex and semantic representations of the input image.

This diagram shows a comparison between the biological visual cortex and a deep learning model. At the bottom, a photograph of a kitten is shown being processed by a series of layers. The first layer is the Retina, followed by Layer 1, Layer 2, and so on up to Layer L. Each layer is represented by a set of nodes connected by arrows, forming a neural network structure. Above this, the term "visual cortex" is written above a bracket that spans from the Retina to Layer L. A yellow arrow points from the text "visual cortex" to the top of the Layer L node set, emphasizing the structural similarity between the two systems.

visual cortex

Retina Layer 1 Layer 2 Layer L

x_1 x_2

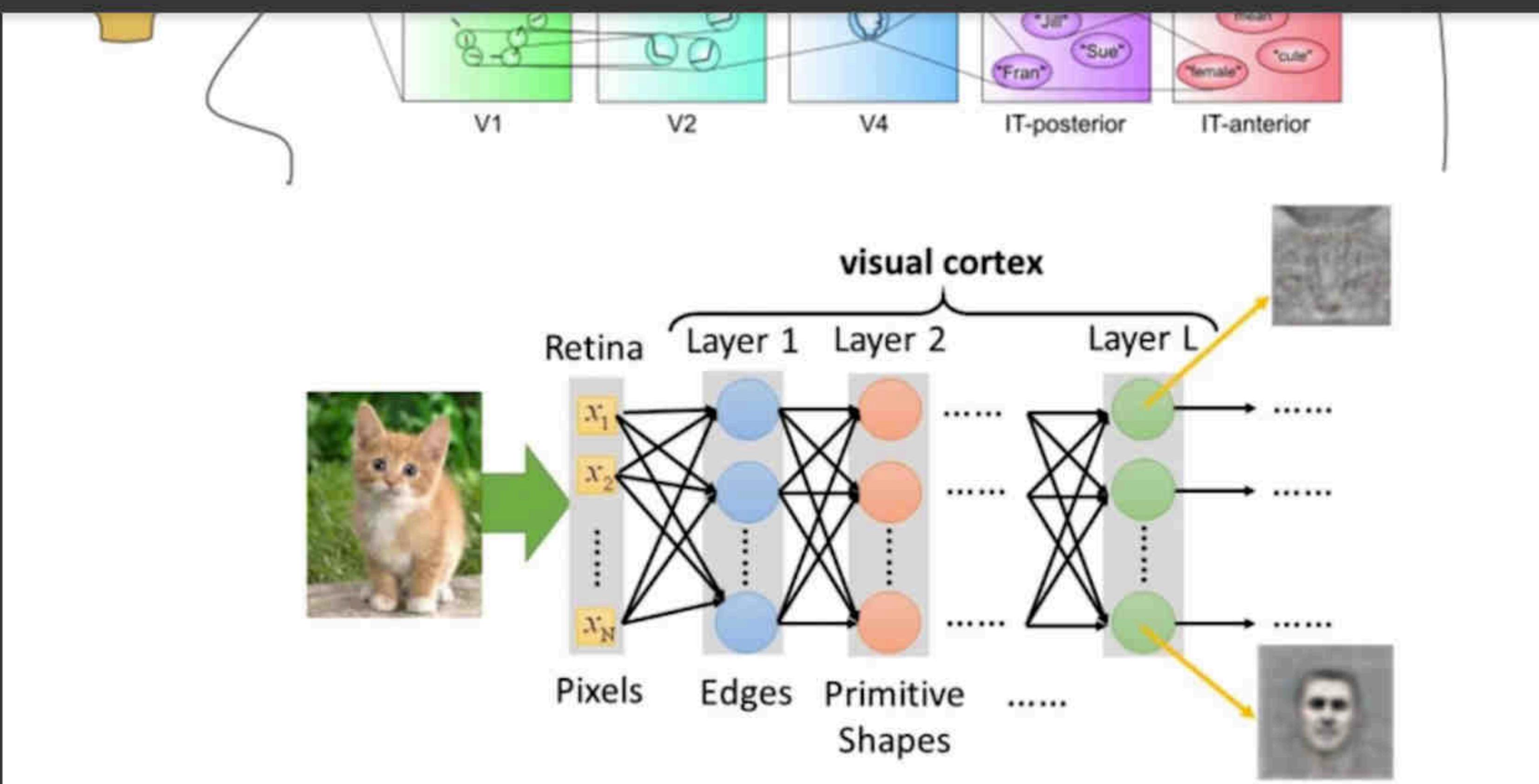
17 / 17

+ Code

+ Text

Changes will not be saved

Connect



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!

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face Colors RGB

colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=bVsL9QWM0Nb7

+ Code + Text Changes will not be saved

Connect

VISUAL Cortex
(Its Structure is Instructive and Inspiring)

{x}

edges

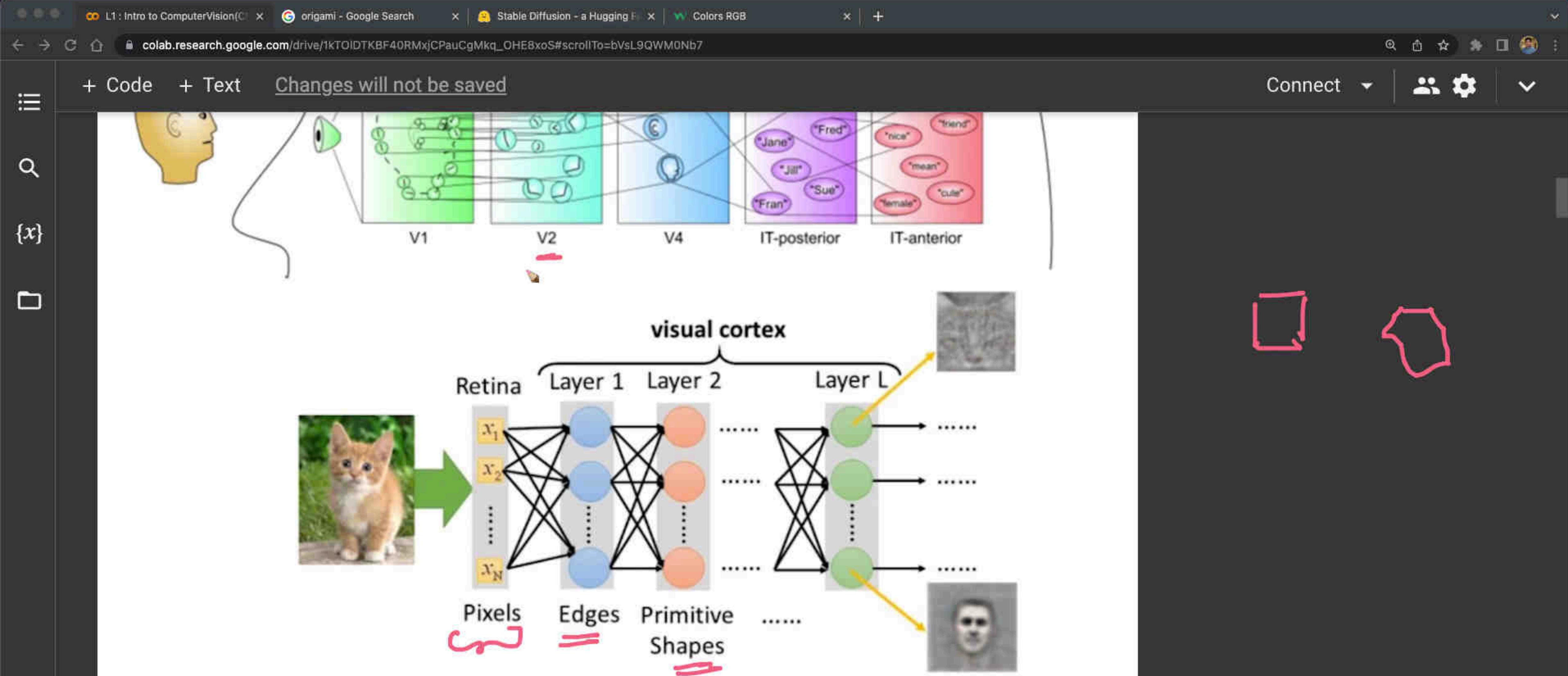
visual cortex

Retina Layer 1 Layer 2 Layer L

x_1 x_2 \dots x_N

Diagram illustrating the visual pathway from the retina to the visual cortex. A green arrow points from a photograph of a kitten to the Retina layer, which contains input nodes labeled x_1, x_2, \dots, x_N . These nodes are connected to the first layer of neurons (Layer 1), which are colored blue. Layer 1 is connected to Layer 2, which is colored orange. This pattern continues through Layer L, where neurons are colored green. The final output is shown as a grayscale image of a cat's face.

19 / 19



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!

+ Code + Text Changes will not be saved

Connect

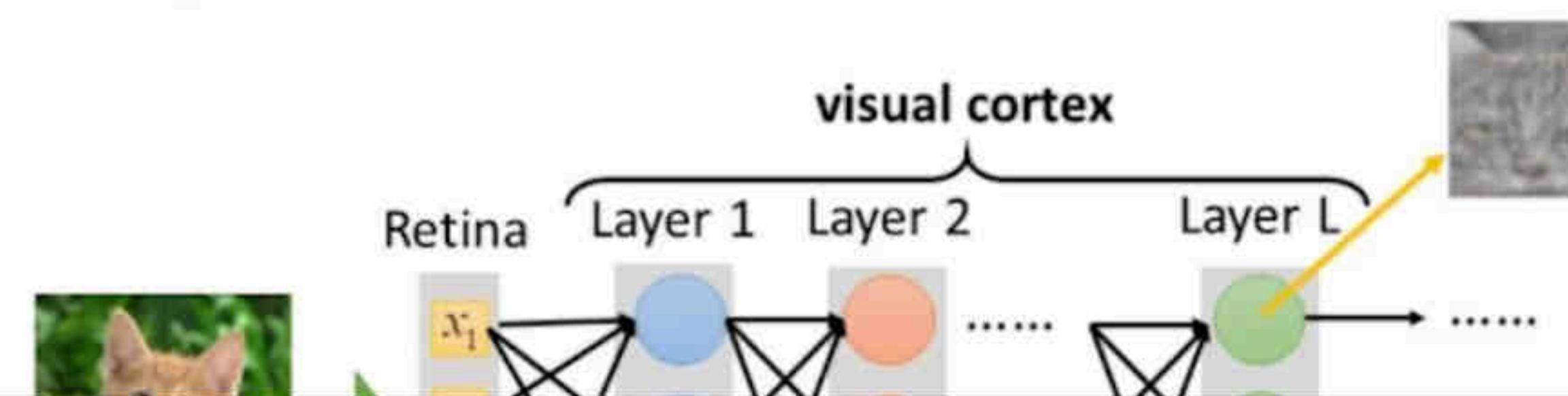
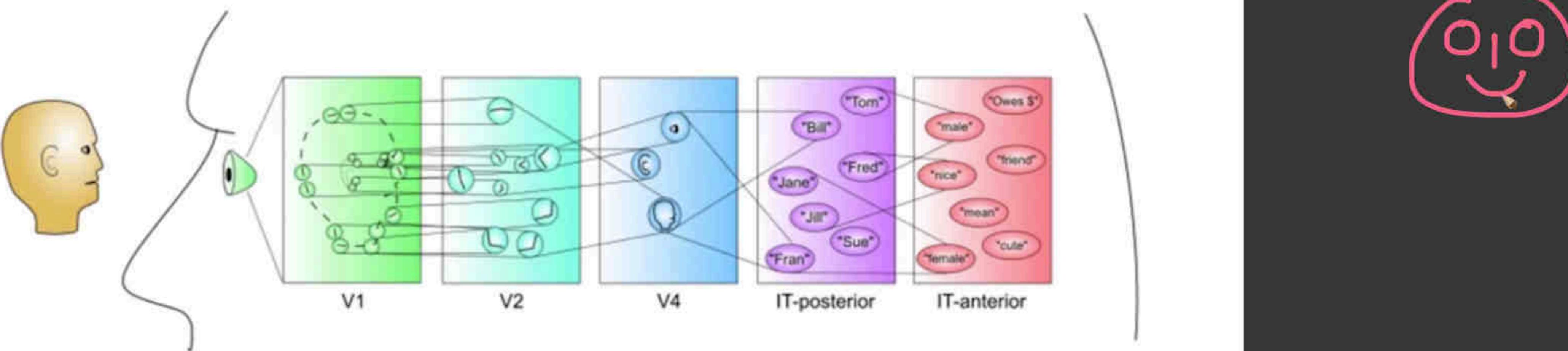


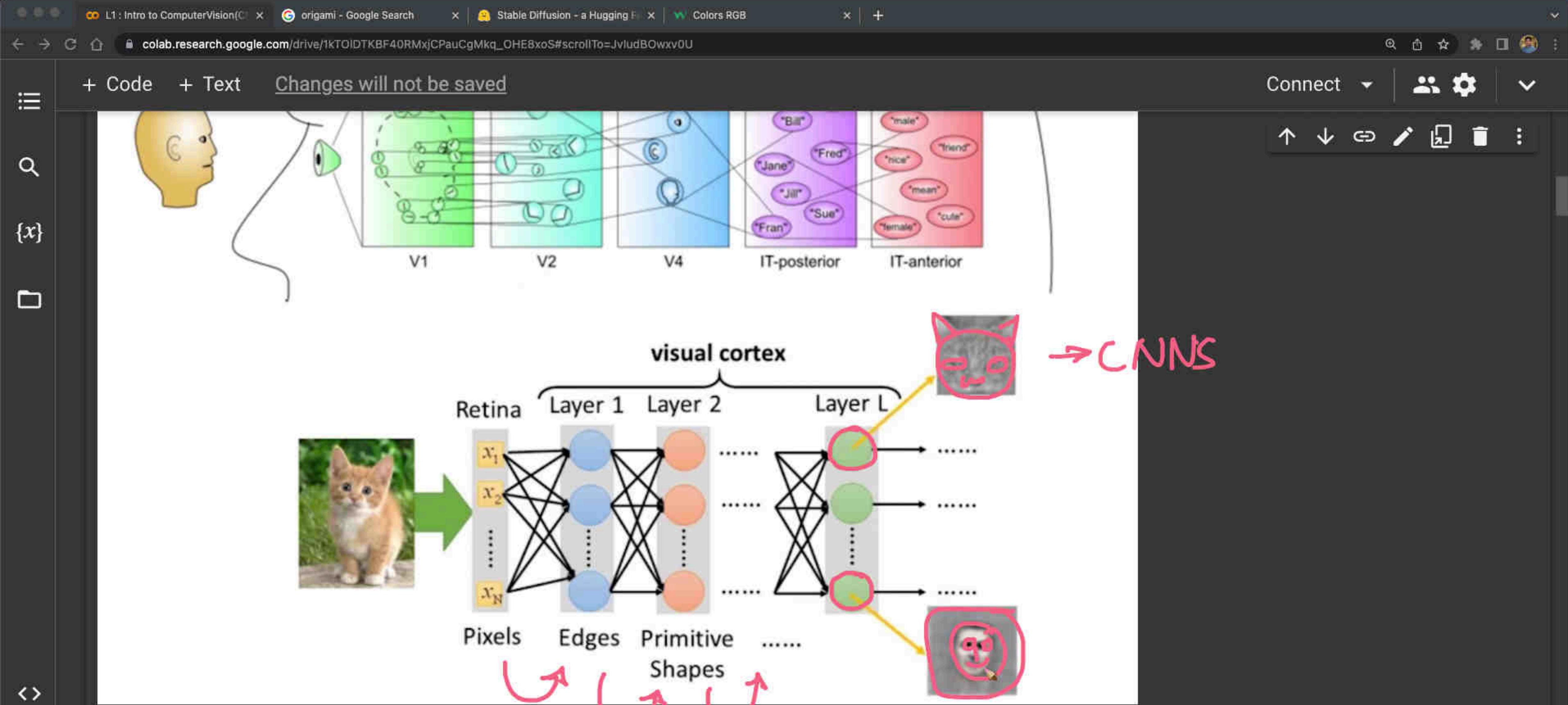
In order to understand why DNN works, we need to account for how visual cortex functions!

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.

Visual Cortex

(Its Structure is Instructive and Inspiring)

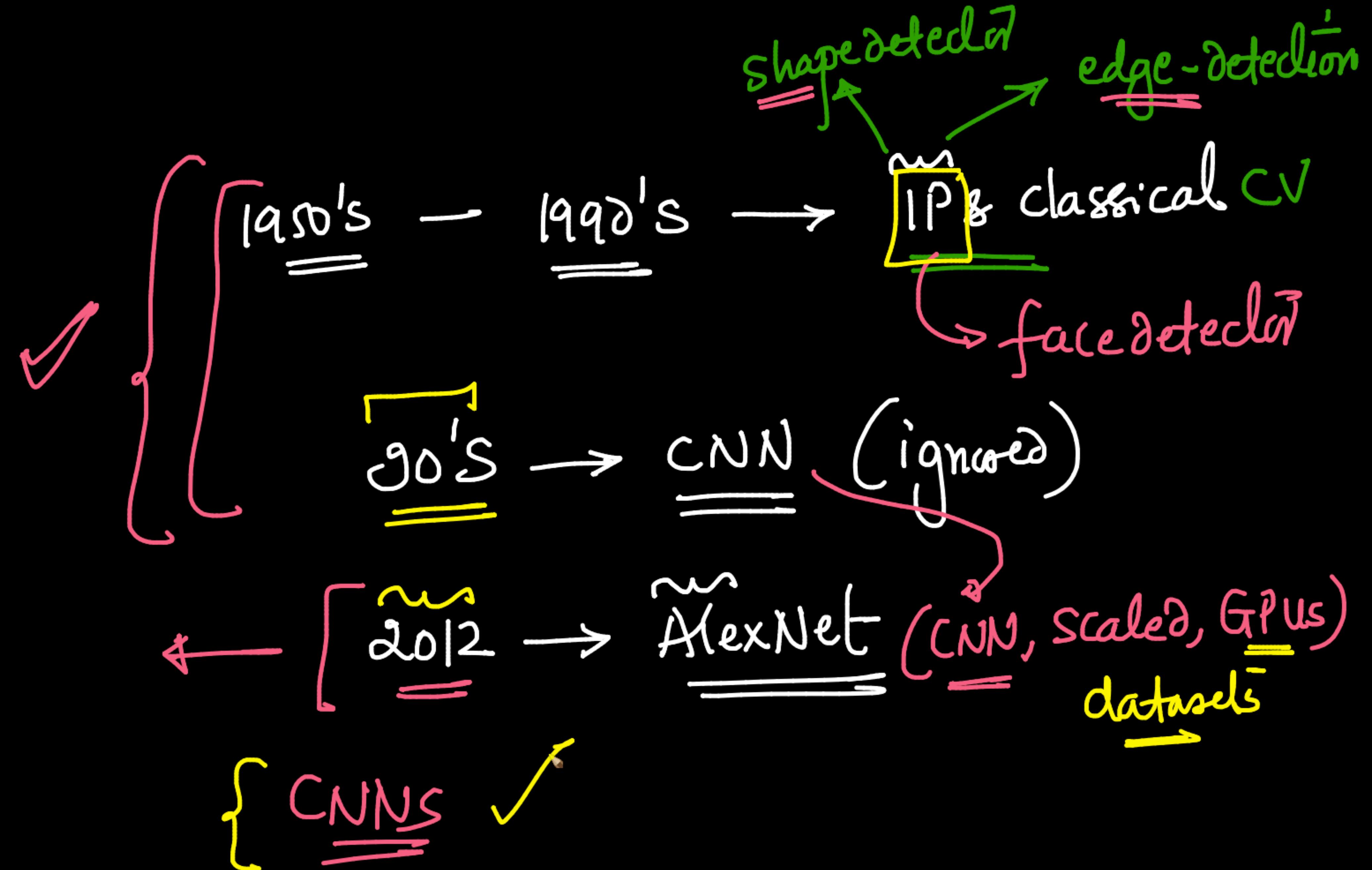


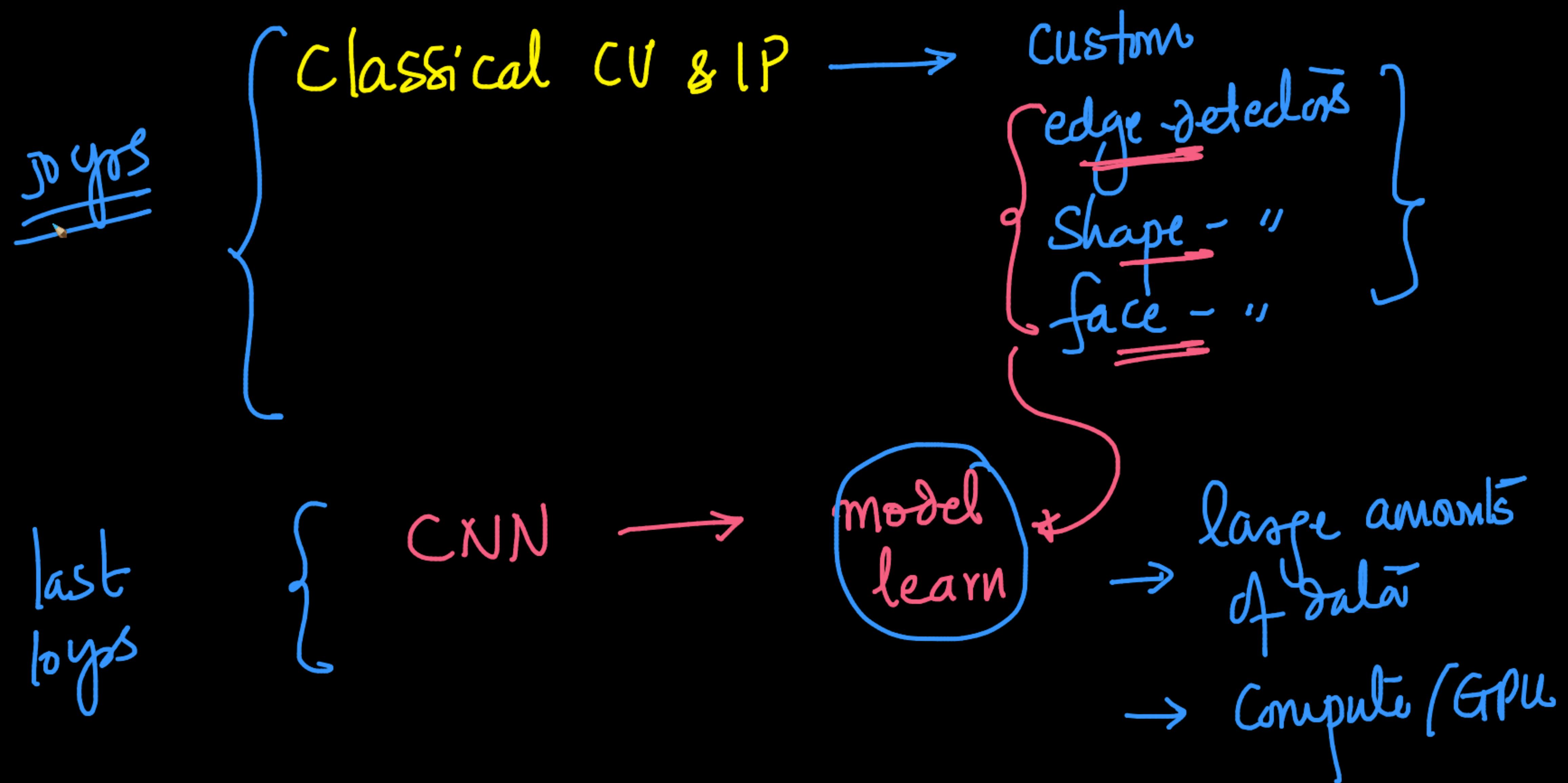


But what about the architecture of DNN, will Vanilla NN or MLP be a suitable fit?

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

History:





L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Colors RGB
colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=JvludBOWxv0U

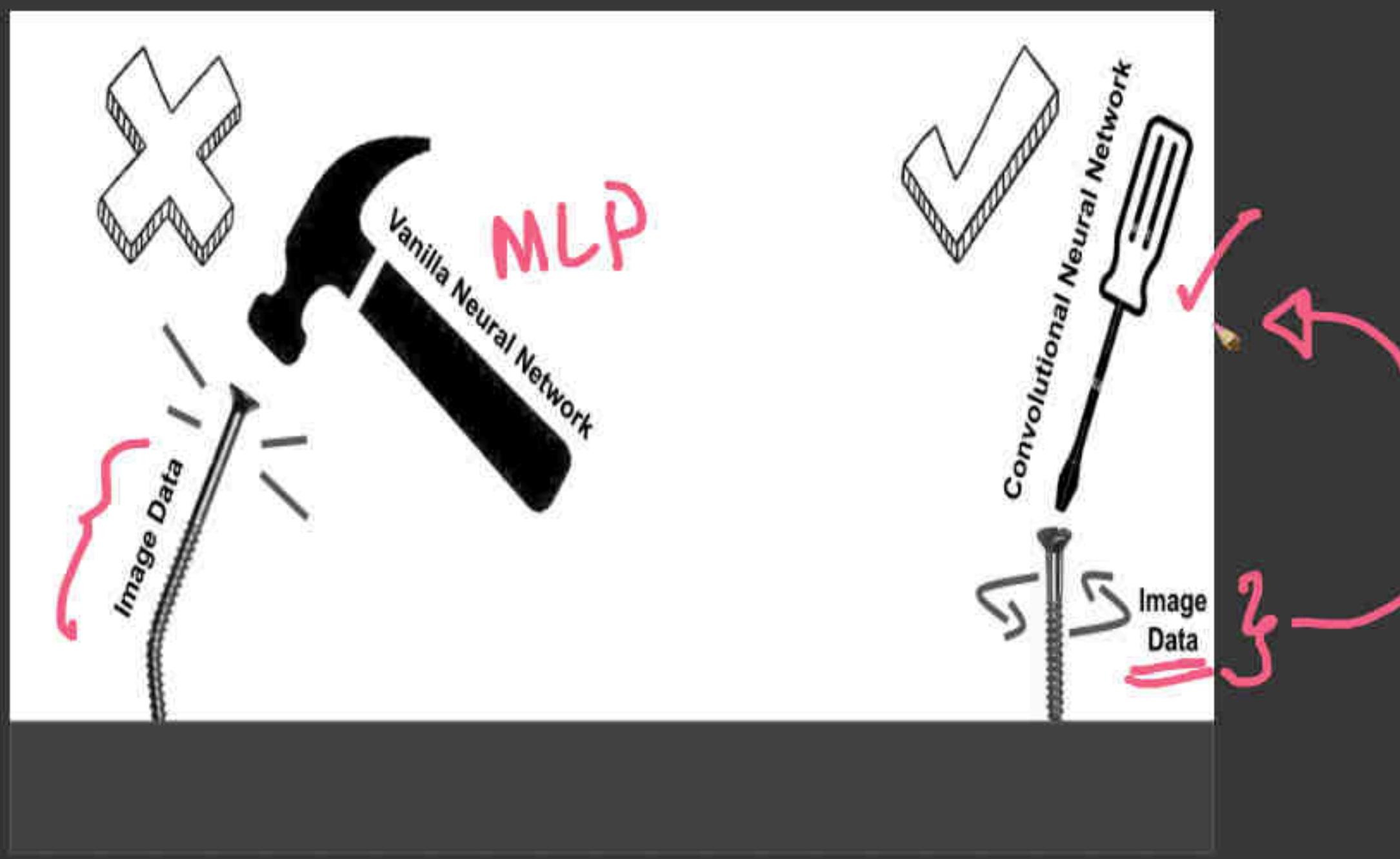
+ Code + Text Changes will not be saved

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

Connect ⚙️

↑ ↓ ⌂ 🖊 🖍 🖞 :

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

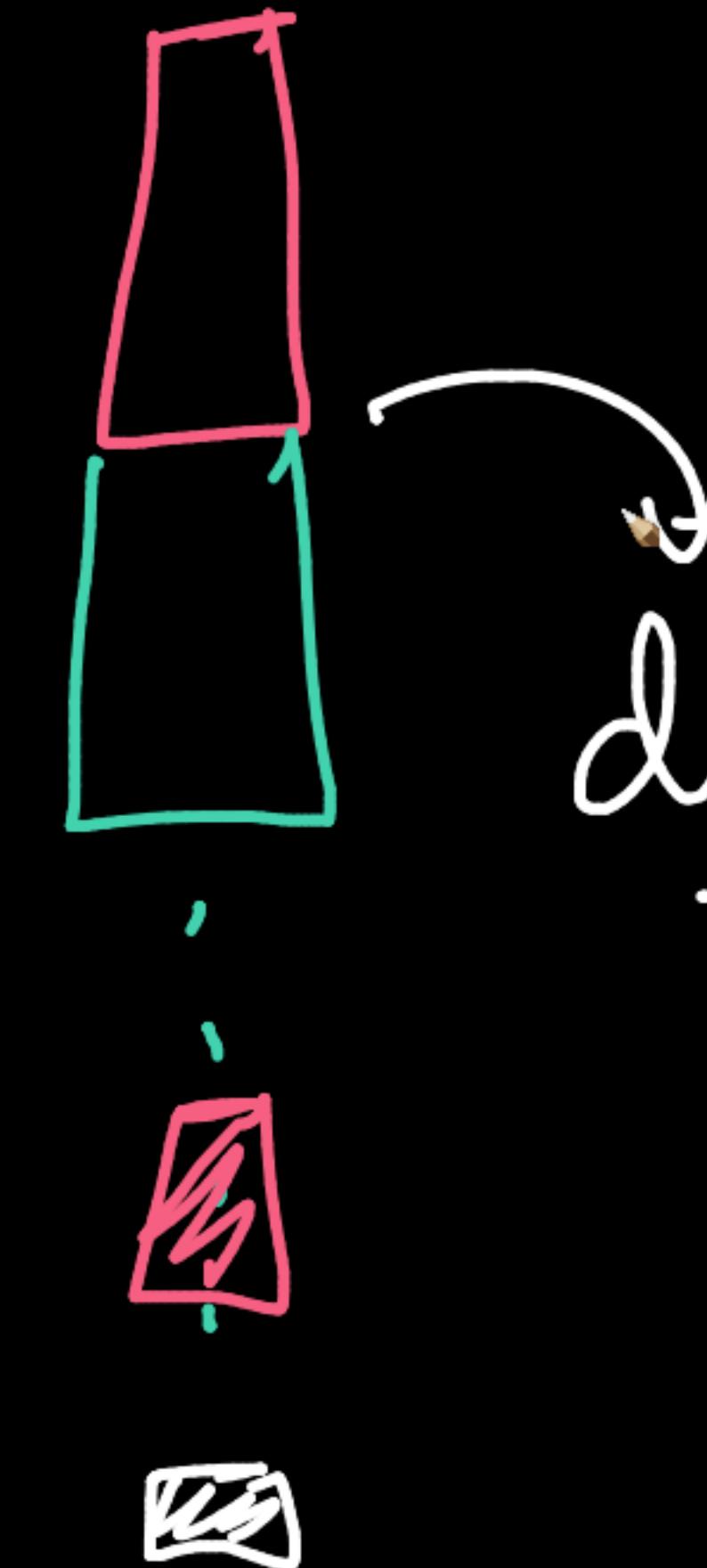
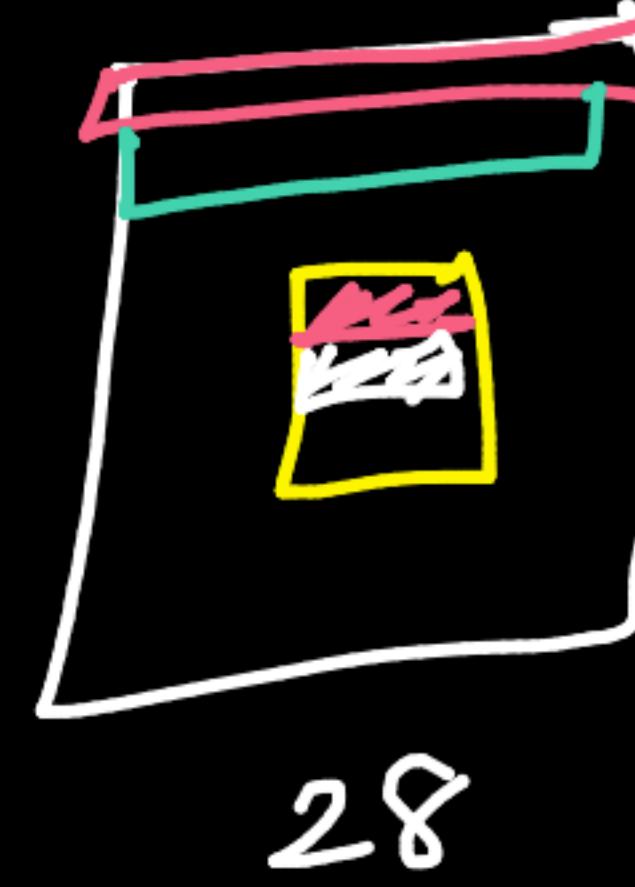
↑
internals
↓
(code)

CNN
==

Convolution
==

Spatial
neighborhood

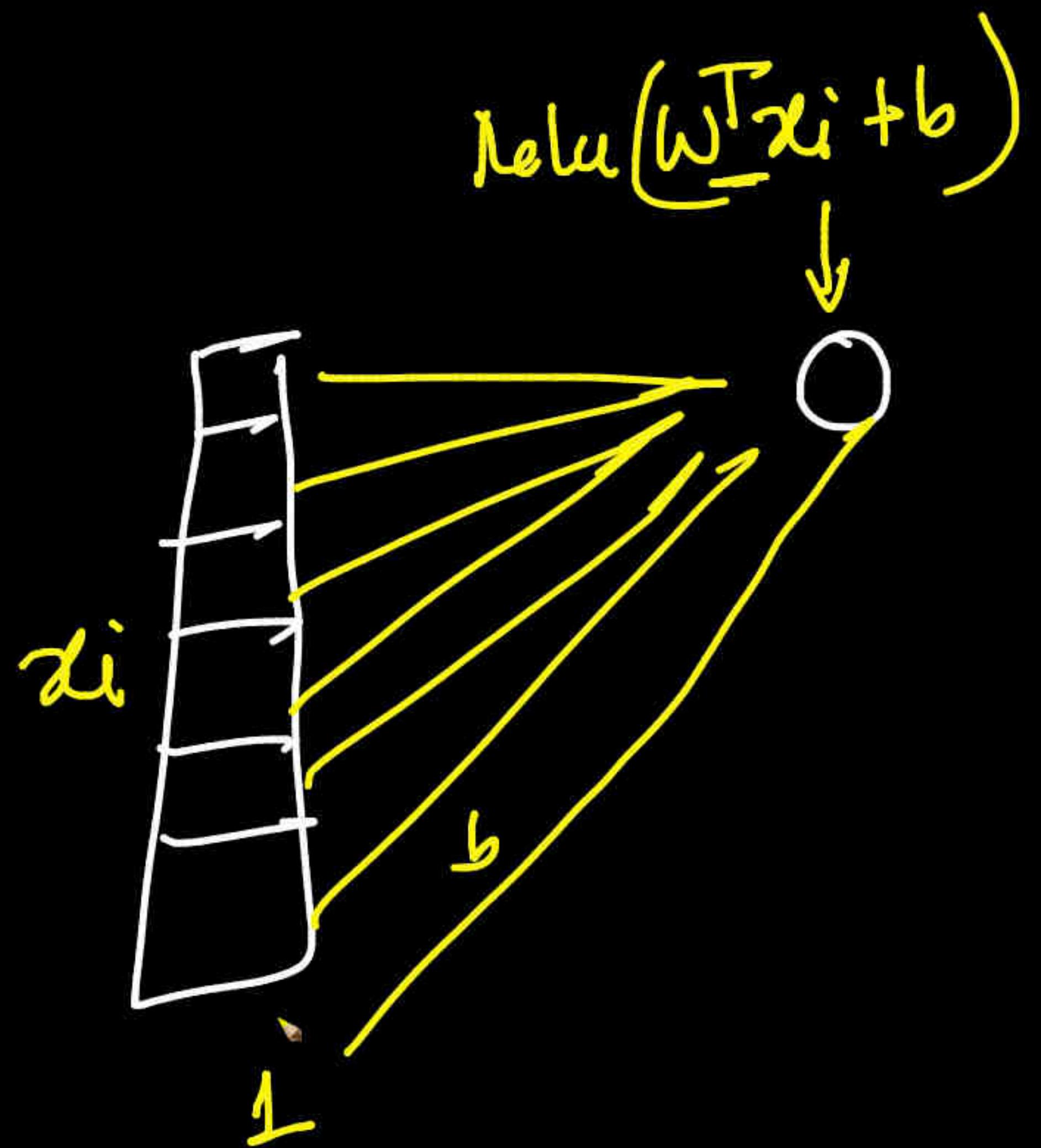
28



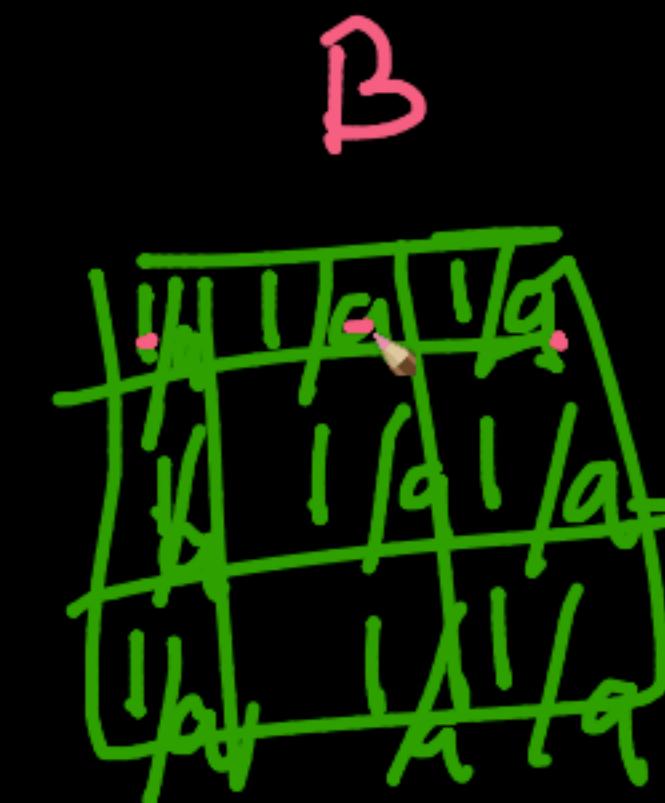
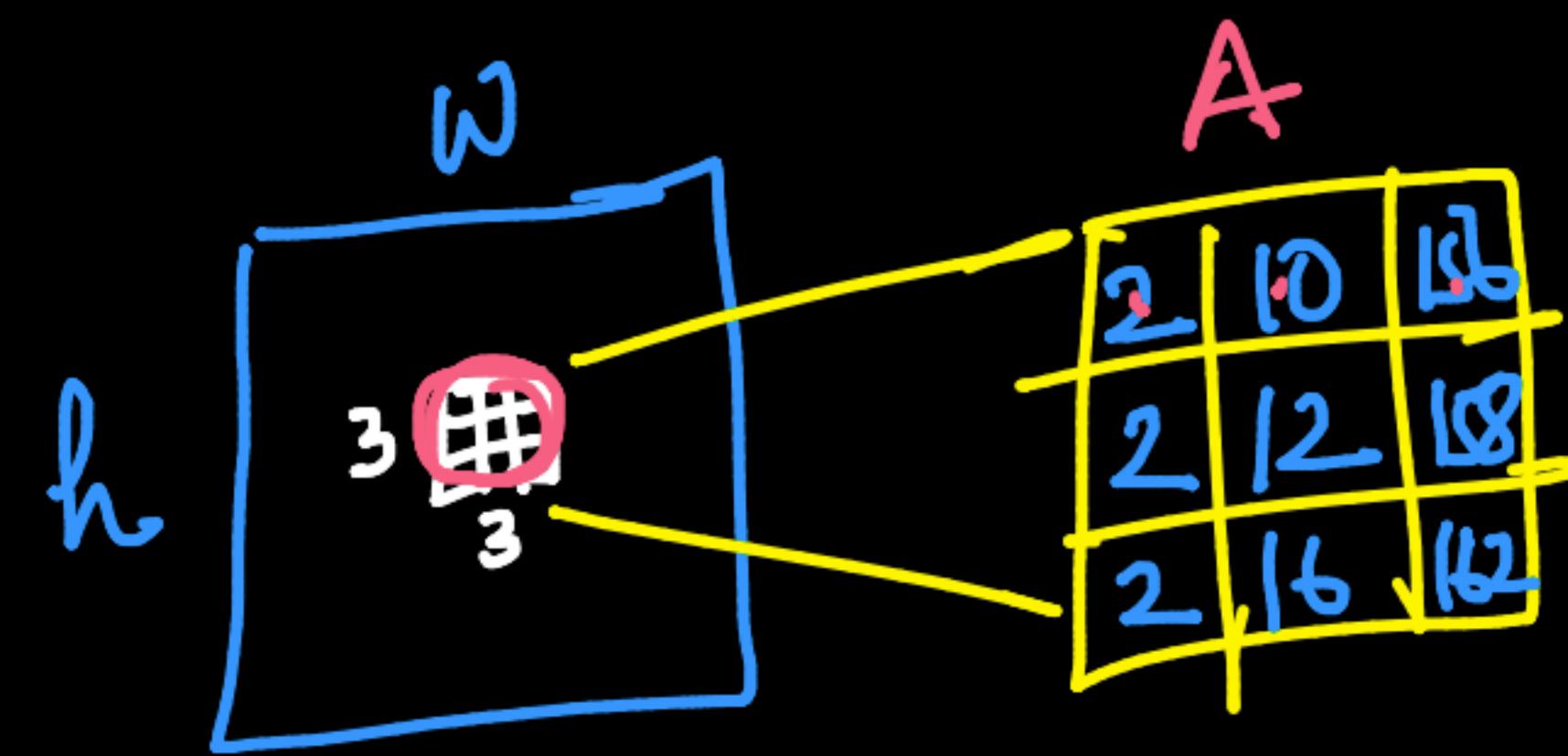
dis-regard

MLP's
==

MLP



grayscale!



average \rightarrow

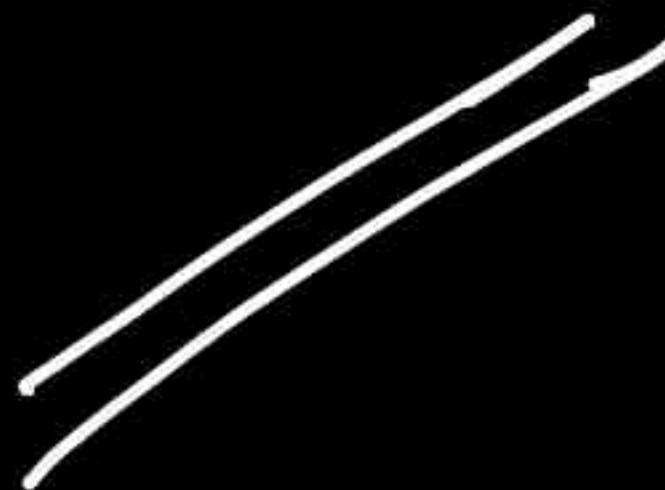
$$\begin{aligned} &= \frac{2}{9} + \frac{10}{9} + \frac{16}{9} \\ &+ \frac{2}{9} + \frac{12}{9} + \dots \end{aligned}$$

component wise mul

$A \odot B$

Convolution { component-wise mul matrix + addition

↓
treating the
Spatial neighborhood



classical IP

Neuroscience

pixels → edges → shapes



objects

faces/cats-

The diagram illustrates a convolution operation. On the left, a 5x3 input image is shown with values 10 in the first four rows and 0 in the last row. On the right, a 3x3 kernel is shown with values 10, 10, 10 in the top row, 10*0, 10*0, 10*0 in the middle row, and 10*0, 10*1, 10*0 in the bottom row. A yellow arrow points from the bottom-left element of the kernel to the bottom-right element of the input image, indicating the receptive field of that output unit. The output layer on the far right shows values 0, 0, 0 in its first three columns. To the right of the output layer is a small image of a face with a green bounding box around its eyes and forehead.

- 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

The diagram illustrates the convolution process between an input image and a kernel. On the left, the input image is a 7x3 grid of value 10, with a 3x3 kernel highlighted by an orange border. To its right is the resulting 4x3 output matrix, where each element is 0. On the right, the kernel is shown as a 3x3 matrix with values 10, 10, 10 in the top row, and 10*0, 10*0, 10*0 in the middle row. The bottom row has 10*0, 10*1, and 10*0. A yellow arrow points from the bottom-left element of the kernel to the bottom-right element of the output matrix, indicating the receptive field of that output unit. Handwritten annotations in orange identify the 'Conv-Matrix' (input), 'filler' (padding zeros), and 'Kernel' (the 3x3 matrix).

- 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

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face Colors RGB

colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=JvludBOWxv0U

+ Code + Text Changes will not be saved Connect |

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

$\{x\}$

$\begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$

Input Image

Vertical edge Filter

Output

34 / 34

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face Colors RGB

colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=JvludBOWxv0U

+ Code + Text Changes will not be saved

Connect

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

$\{x\}$

$\boxed{\begin{array}{ccc} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{array}} * \boxed{\begin{array}{ccc} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{array}} = \boxed{\begin{array}{ccc} 0 & 30 & 0 \\ 30 & 30 & 0 \\ 0 & 30 & 0 \end{array}}$

$10 + 10 + 10 + 10 + 10 = 50$

Input Image

Vertical edge Filter

Output

35 / 35

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face Colors RGB

colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=JvludBOWxv0U

+ Code + Text Changes will not be saved

Connect

Changes will not be saved

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

Input Image

Vertical edge Filter

Output

Handwritten annotations:
Input Image: A 6x6 grid of values. The first four columns are labeled with a red '6' at the top left. The last two columns are labeled with a red '0'. The first four rows have a red border. The last two rows are labeled with a red '0'. The values in the grid are: Row 1: [10, 10, 10, 0, 0, 0]. Row 2: [10, 10, 10, 0, 0, 0]. Row 3: [10, 10, 10, 0, 0, 0]. Row 4: [10, 10, 10, 0, 0, 0]. Row 5: [0, 10, 10, 10, 0, 0]. Row 6: [0, 10, 10, 10, 0, 0].
Vertical edge Filter: A 3x3 grid of values. The first two columns are labeled with a red '3' at the top left. The last column is labeled with a red '-1'. The values in the grid are: Row 1: [1, 0, -1]. Row 2: [1, 0, -1]. Row 3: [1, 0, -1].
Output: A 4x4 grid of values. The first three columns are labeled with a red '4' at the top right. The last column is labeled with a red '0'. The values in the grid are: Row 1: [0, 30, 30, 0]. Row 2: [0, 30, 30, 0]. Row 3: [0, 30, 30, 0]. Row 4: [0, 30, 30, 0].

36 / 36

L1: Intro to ComputerVision(C) x origami - Google Search x Stable Diffusion - a Hugging Face Model x Colors RGB x + colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=JvludBOWxv0U

+ Code + Text Changes will not be saved Connect |  

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

{x}

Input Image

Vertical edge Filter

Output

* =

Diagram illustrating the convolution process. An input image of size 7x6 is multiplied by a vertical edge filter of size 3x3. The result is an output image of size 5x4.

Input Image Data:

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Vertical edge Filter Data:

1	0	-1
1	0	-1
1	0	-1

Output Image Data:

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Annotations: A red circle highlights the rightmost column of the input image. A red arrow points from the label "Vertical edge Filter" to the filter matrix. Another red circle highlights the rightmost column of the output image.

L1: Intro to ComputerVision(C)

origami - Google Search

Stable Diffusion - a Hugging Face Model

Colors RGB

colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=JvludBOWxv0U

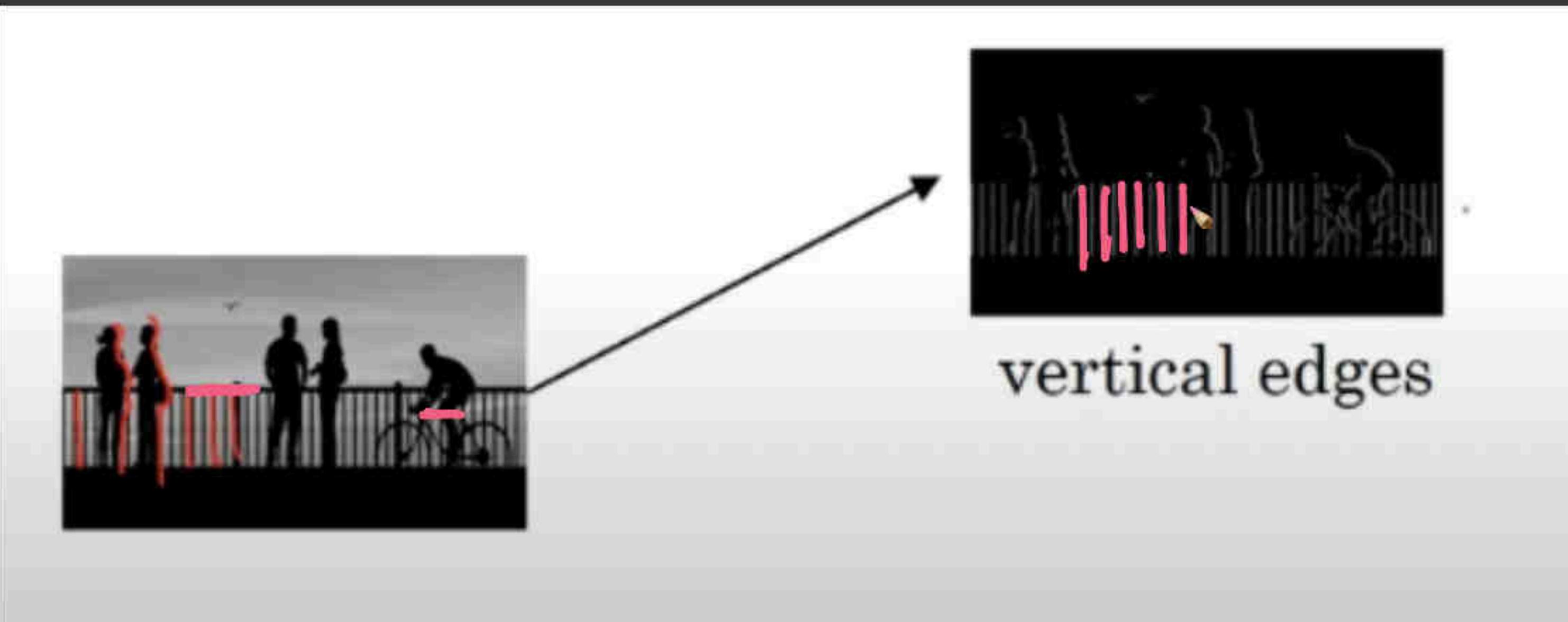
+ Code + Text Changes will not be saved

Connect



We have an image of a car on a road, the kernel when sliding over the image will be able to identify the edges (shape) of a car, by 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

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Face Colors RGB

colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=JvludBOWxv0U

+ Code + Text Changes will not be saved

Connect

Changes will not be saved

Input Image

Output

Vertical edge Filter

39 / 39

The diagram illustrates a convolution operation. On the left, a 6x3 input image matrix $\{x\}$ is shown, where each row contains three values: 10, 10, 10. To its right is a 3x3 filter matrix with values 1, 0, -1 in each row. An asterisk (*) indicates the multiplication of the input by the filter. An equals sign (=) indicates the result. The output matrix on the right has a central teal block of size 3x3 containing the value 30, surrounded by a black border. A red arrow points from the text "Vertical edge Filter" to the filter matrix, highlighting its role in detecting vertical edges.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

L1: Intro to Computer Vision(C) origami - Google Search Stable Diffusion - a Hugging Face Colors RGB

colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=JvludBOWxv0U

+ Code + Text Changes will not be saved

Connect

Changes will not be saved

{x}

Input Image

Vertical edge Filter

Output

40 / 40

The diagram illustrates the convolution process. An input image of size 5x3 with values 10 and 0 is multiplied by a vertical edge filter of size 3x3 with values 1, 0, -1. The result is an output image of size 3x3 with values 30 and 0.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

{x}

10	10	10	10	0	0	0
10	10	10	10	0	0	0
10	10	10	10	0	0	0
10	10	10	10	0	0	0
10	10	10	10	0	0	0

Input Image

*

1	0	-1
1	0	-1
1	0	-1

Vertical edge
Filter

=

0	30	30	0	0
0	30	30	0	0
0	30	30	0	0
0	30	30	0	0
0	30	30	0	0

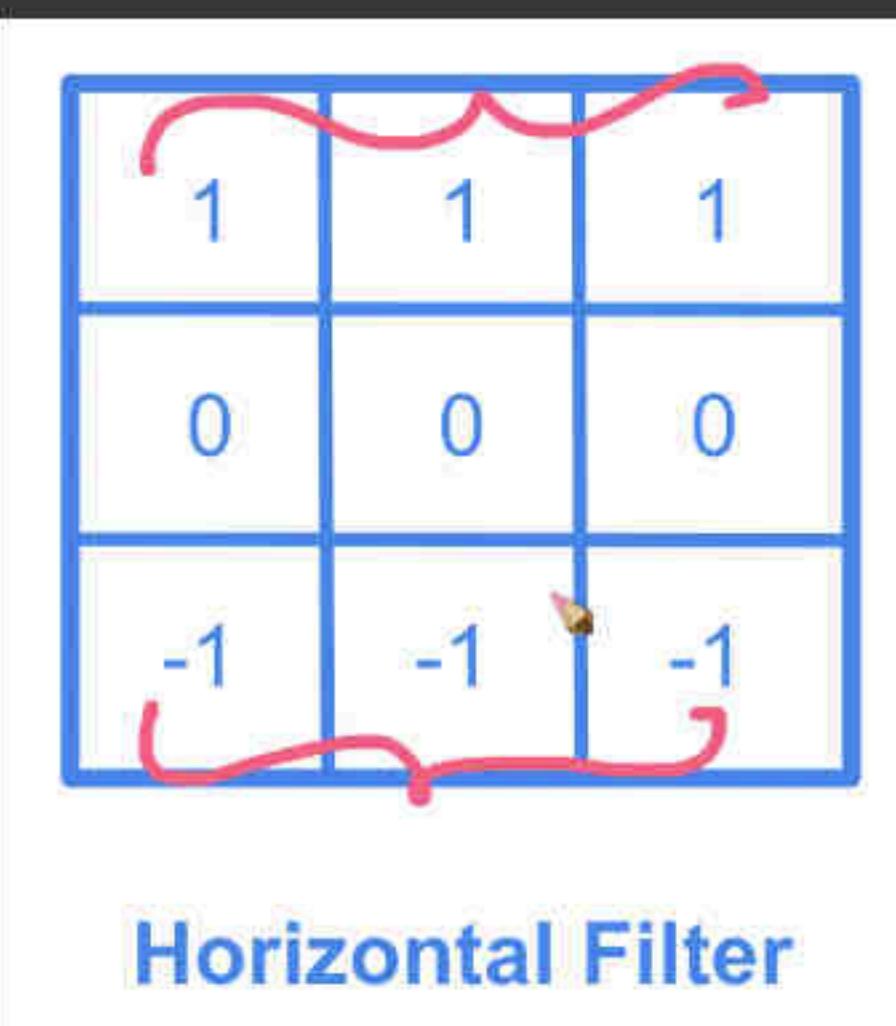
Output

+ Code + Text Changes will not be saved

Connect  

- 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?



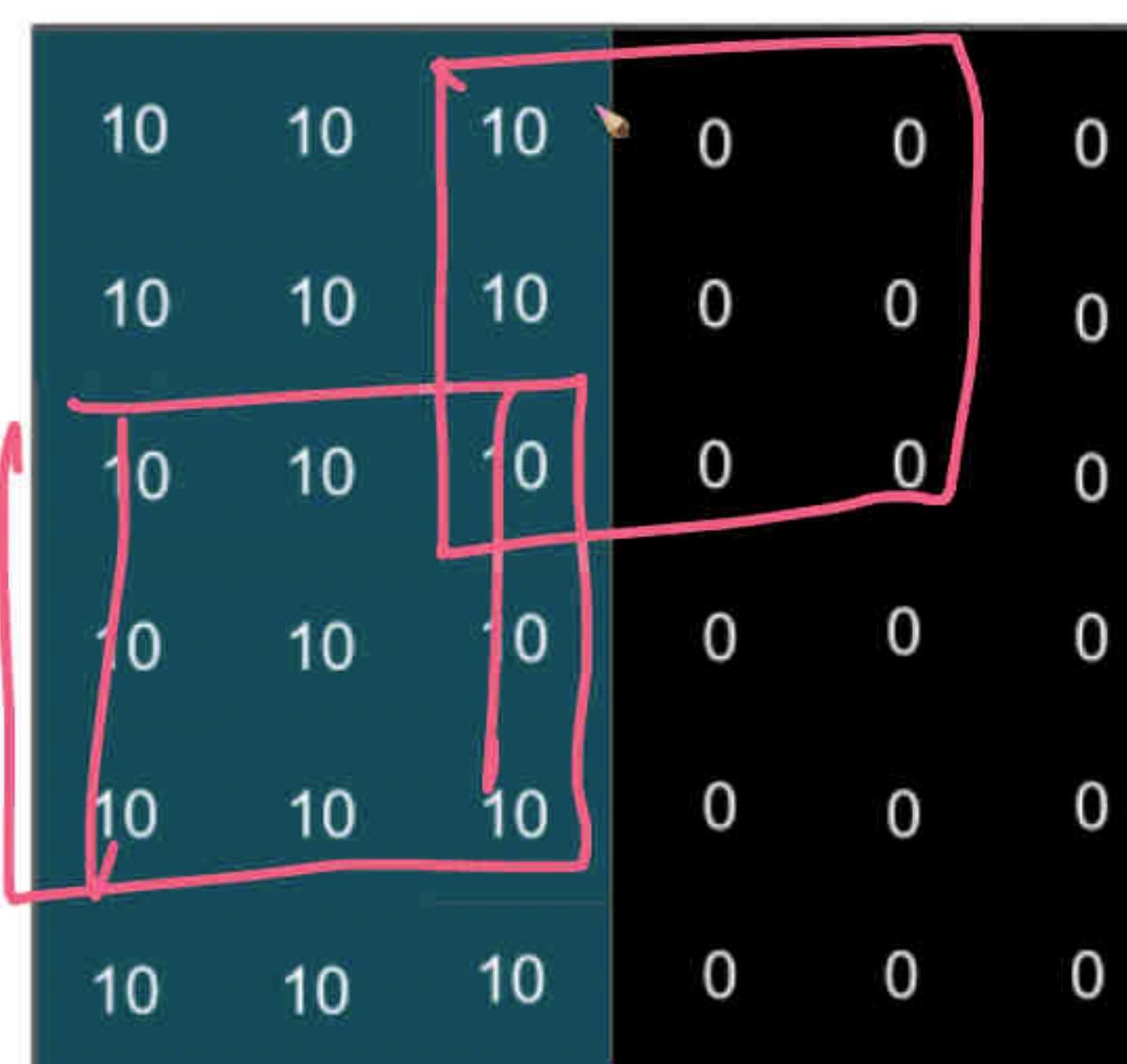
Q. How can we apply both horizontal and vertical edge detector on the same image matrix?

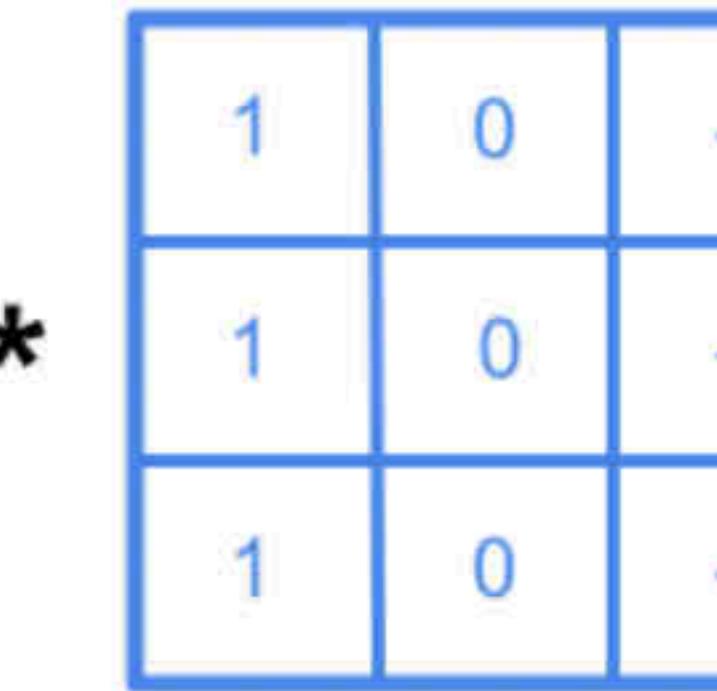
L1: Intro to Computer Vision(C) origami - Google Search Stable Diffusion - a Hugging Face Colors RGB colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=JvludBOWxv0U

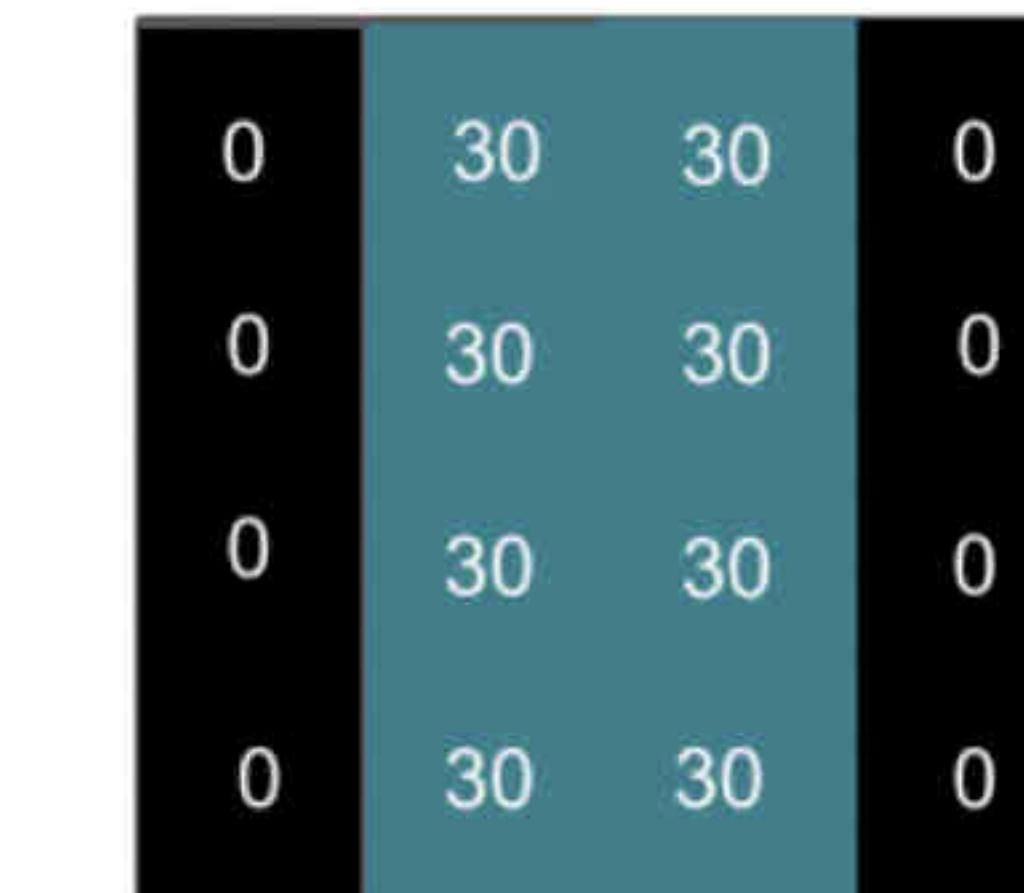
+ Code + Text Changes will not be saved Connect |  

- 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:

$\{x\}$

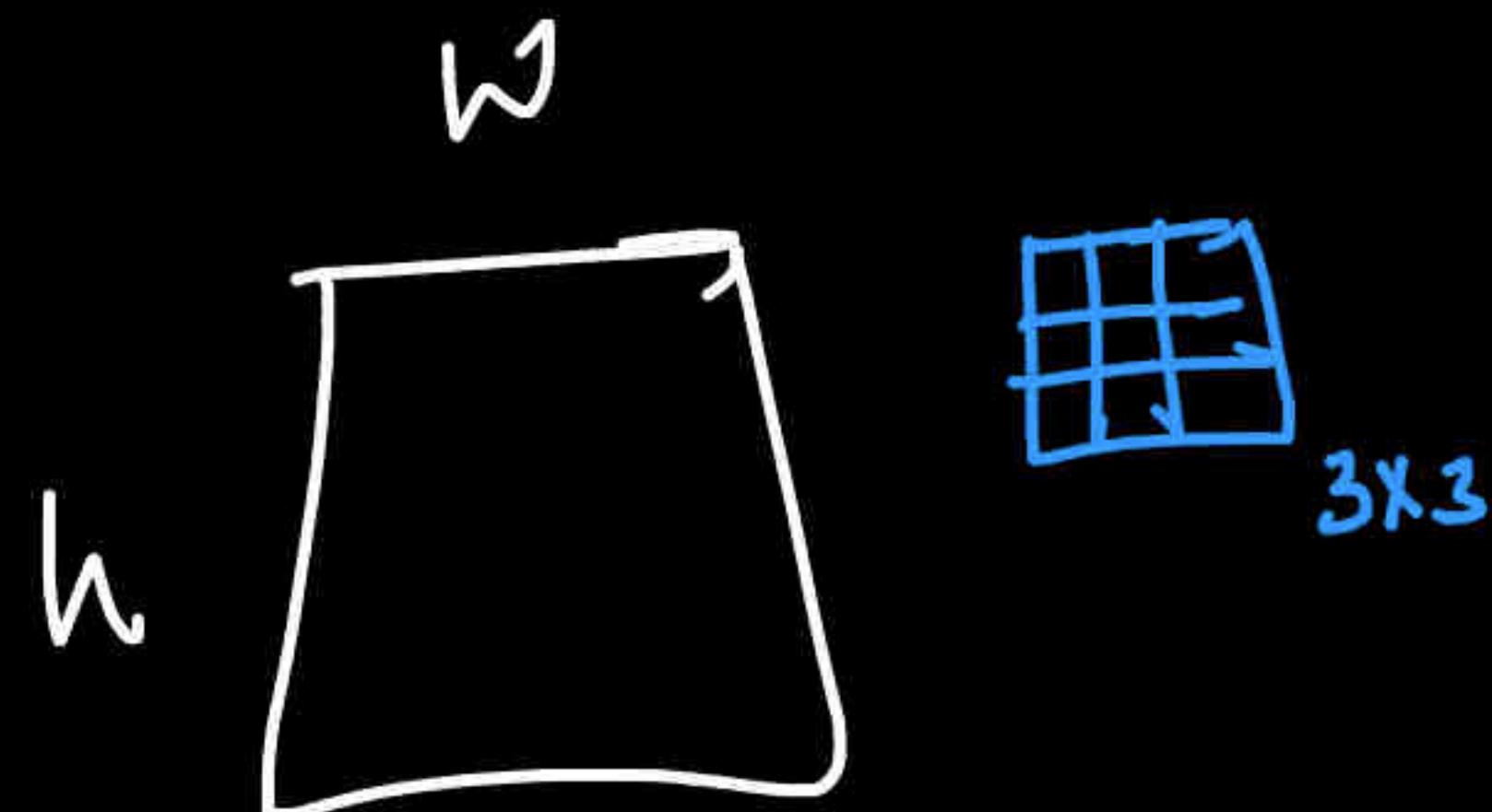

Input Image

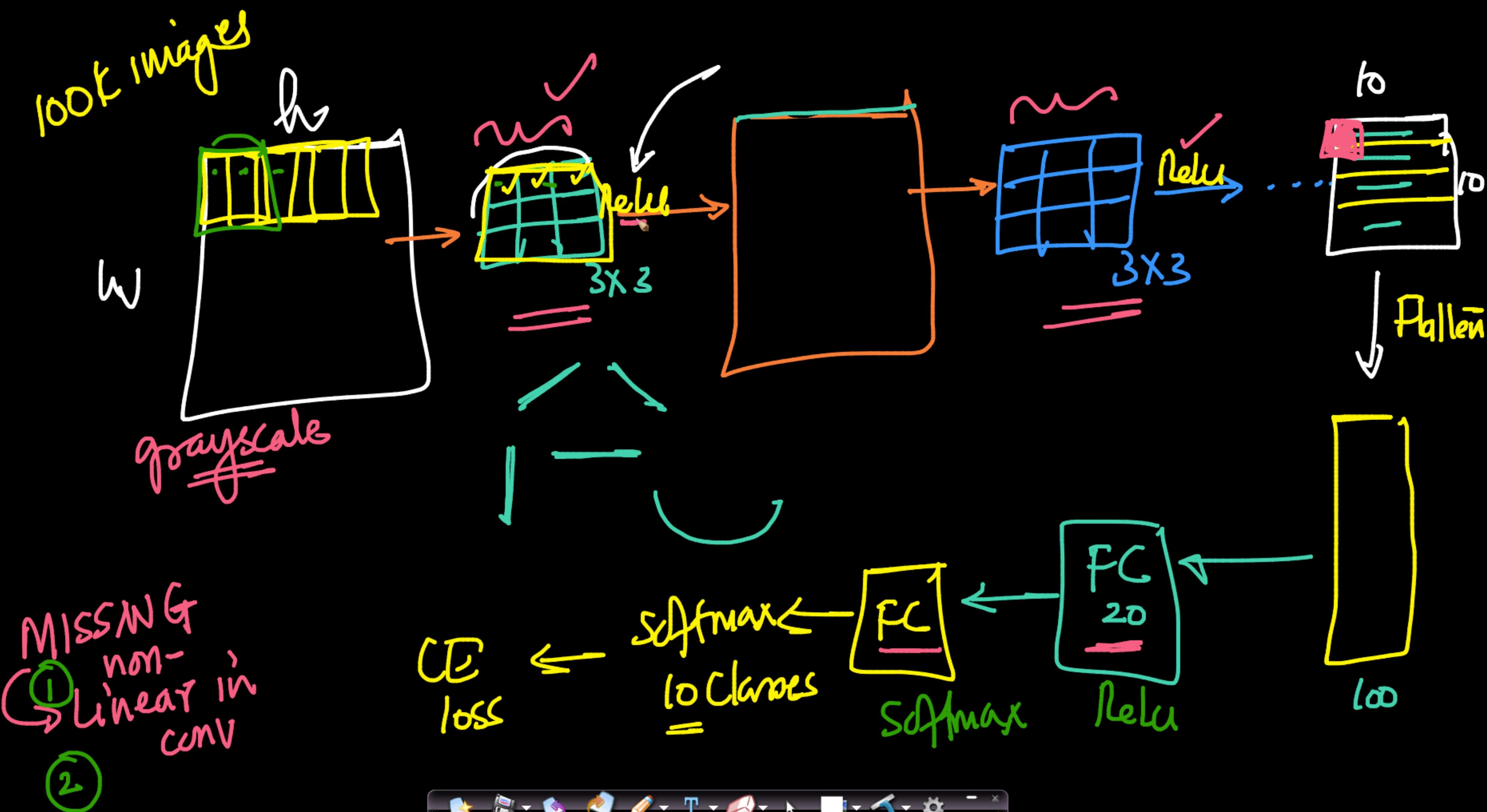
$*$ 
Vertical edge Filter

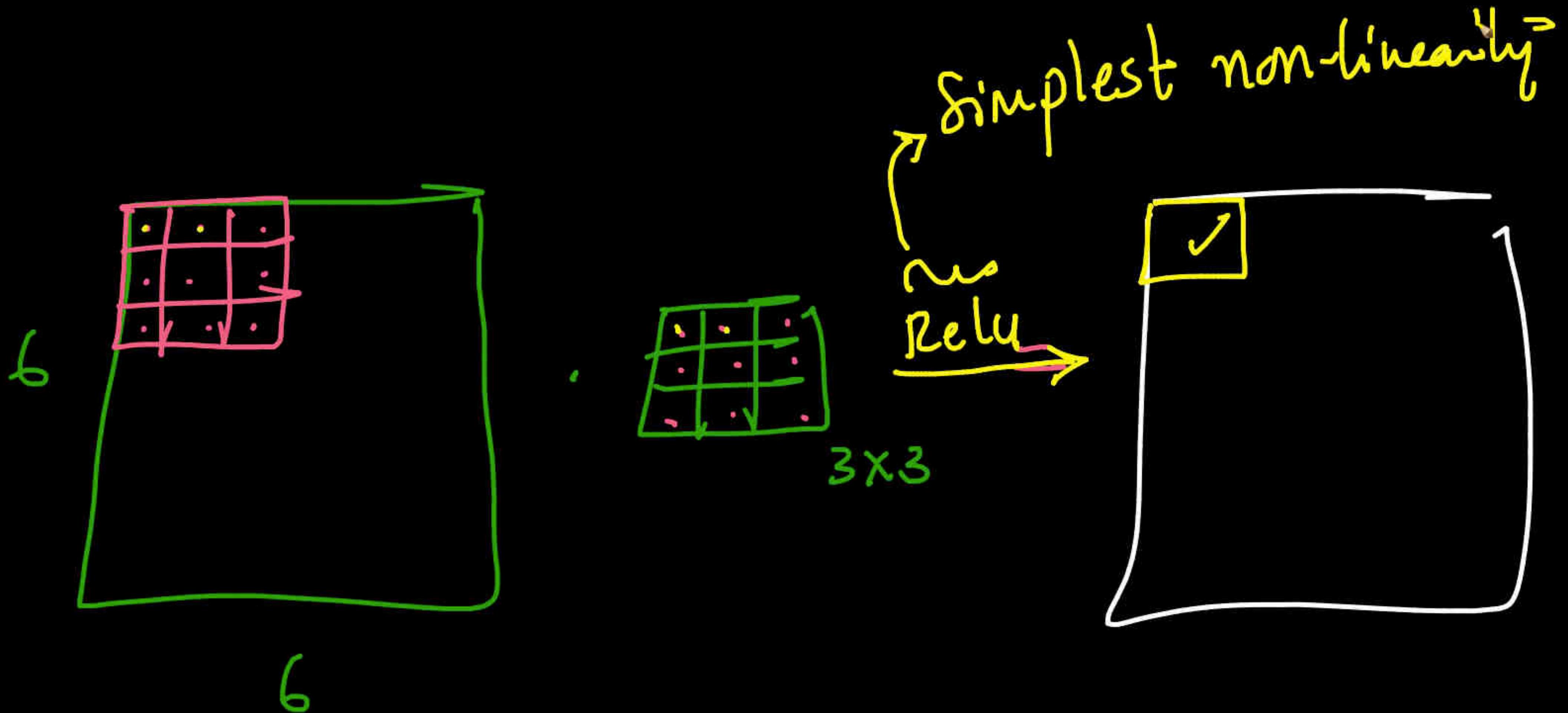
$=$ 
Output

Convolution \rightarrow spatial coherence

\rightarrow edge detection (IP)





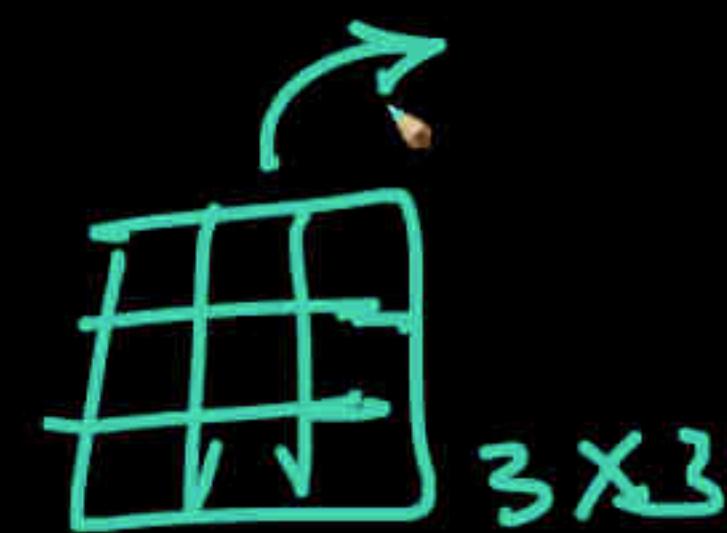


non-linearity $\rightarrow z_i$

$$\text{MLP: } f_{\text{ReLU}}(\underbrace{\omega^T}_{\text{w}} \underbrace{x_i}_{z_i} + b)$$

$$\text{Max}(0, z_i)$$

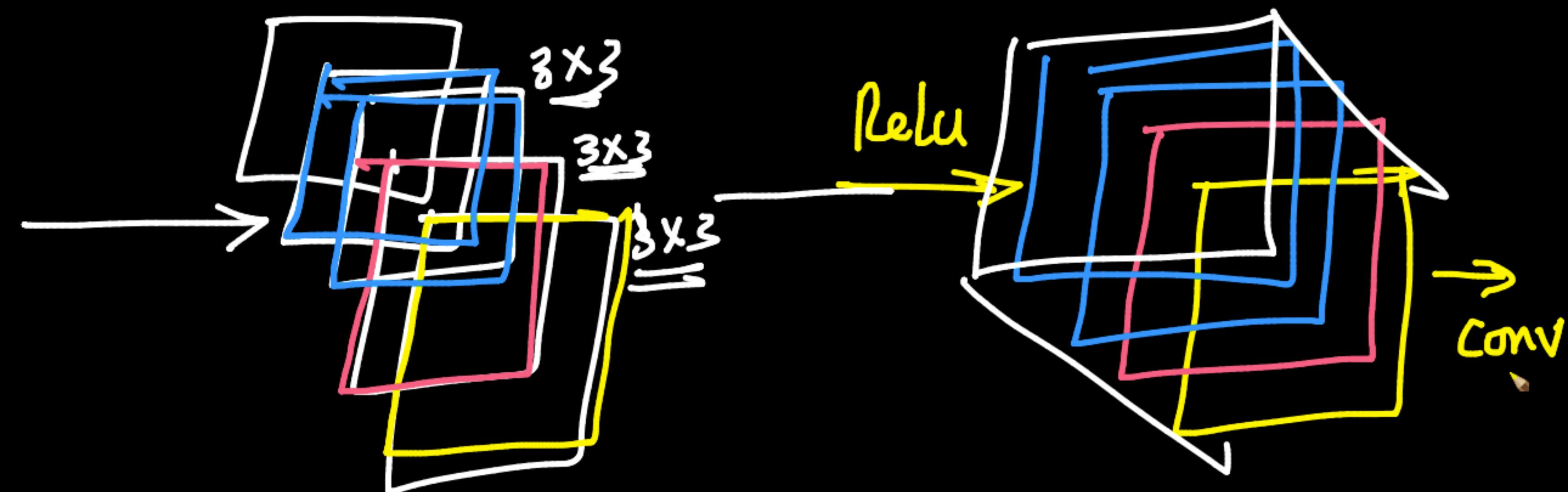
CNN: $f_{\text{ReLU}}(\underbrace{i_m \odot k_i}_{\text{w}})$



Pixels → edges → shapes → objects...



grayscale
Image
 $h \times w$

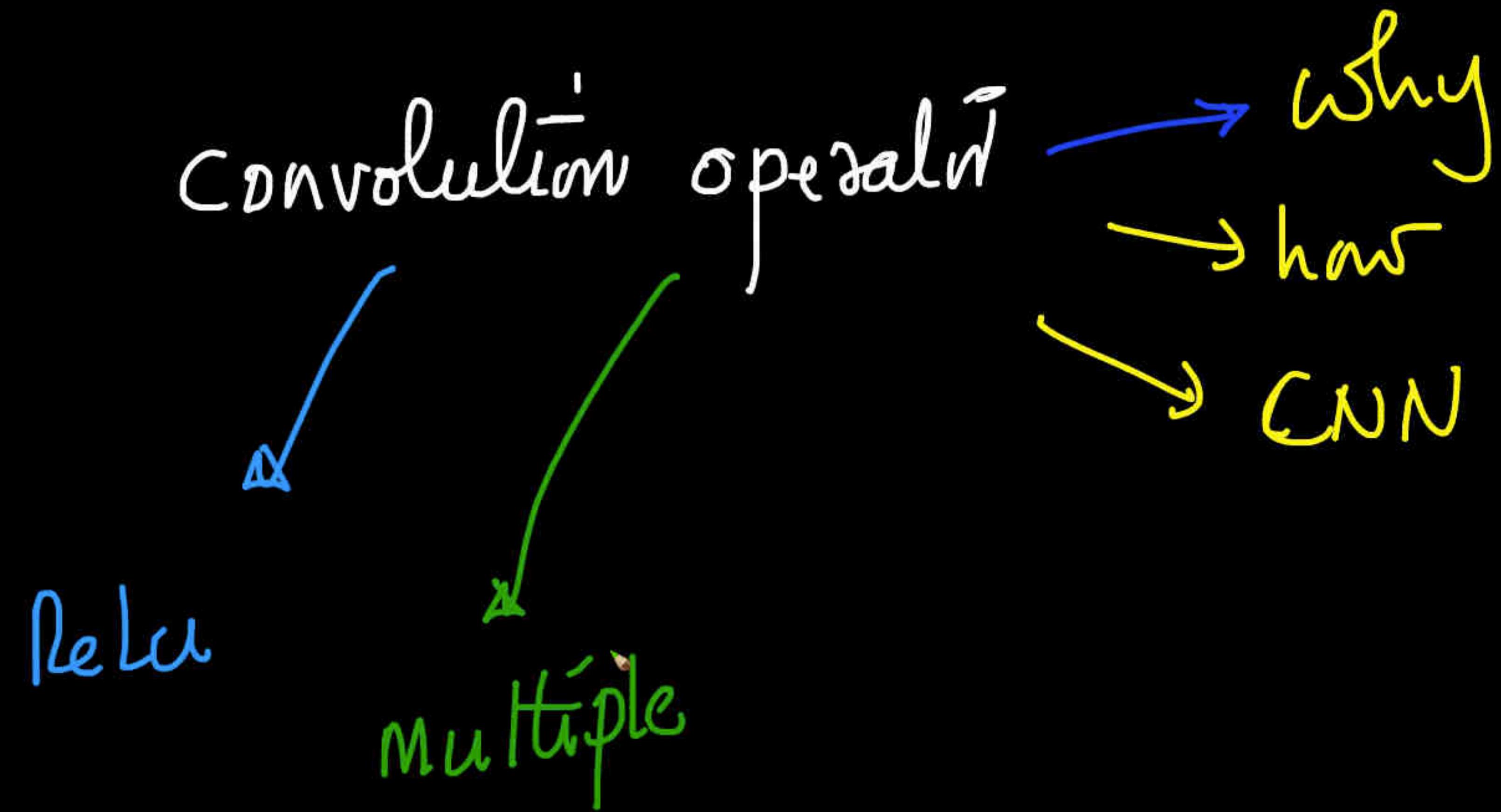


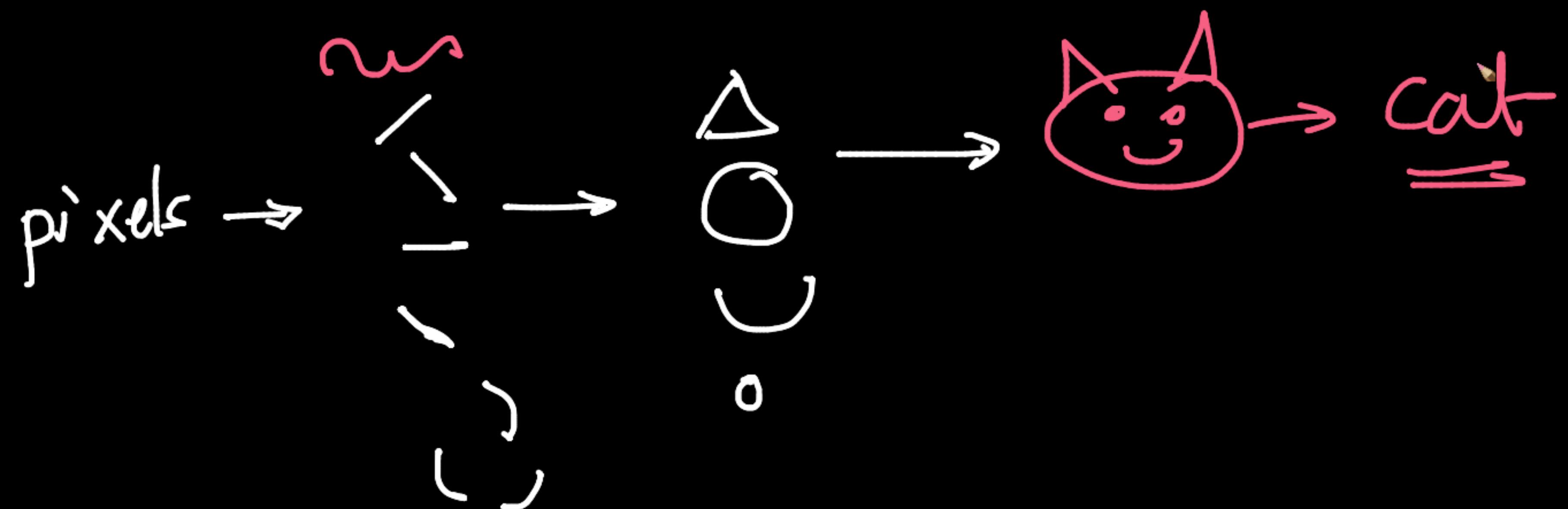
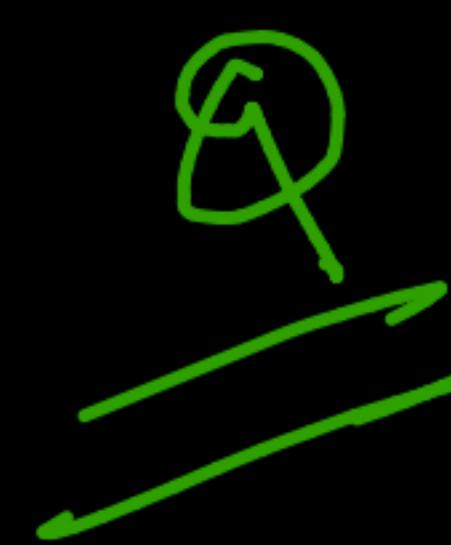
5 kernels

$$9 \times 5 = 45$$

45

Tensor
Tensor

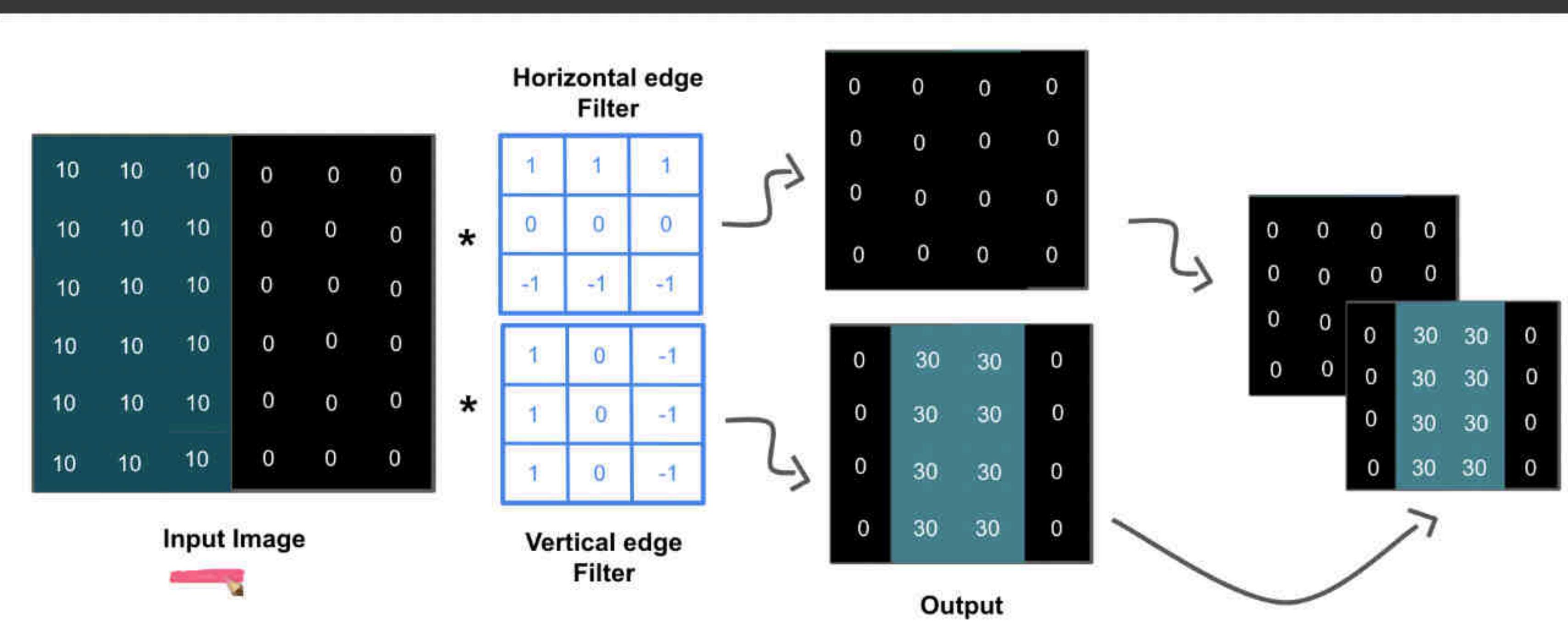




+ Code + Text

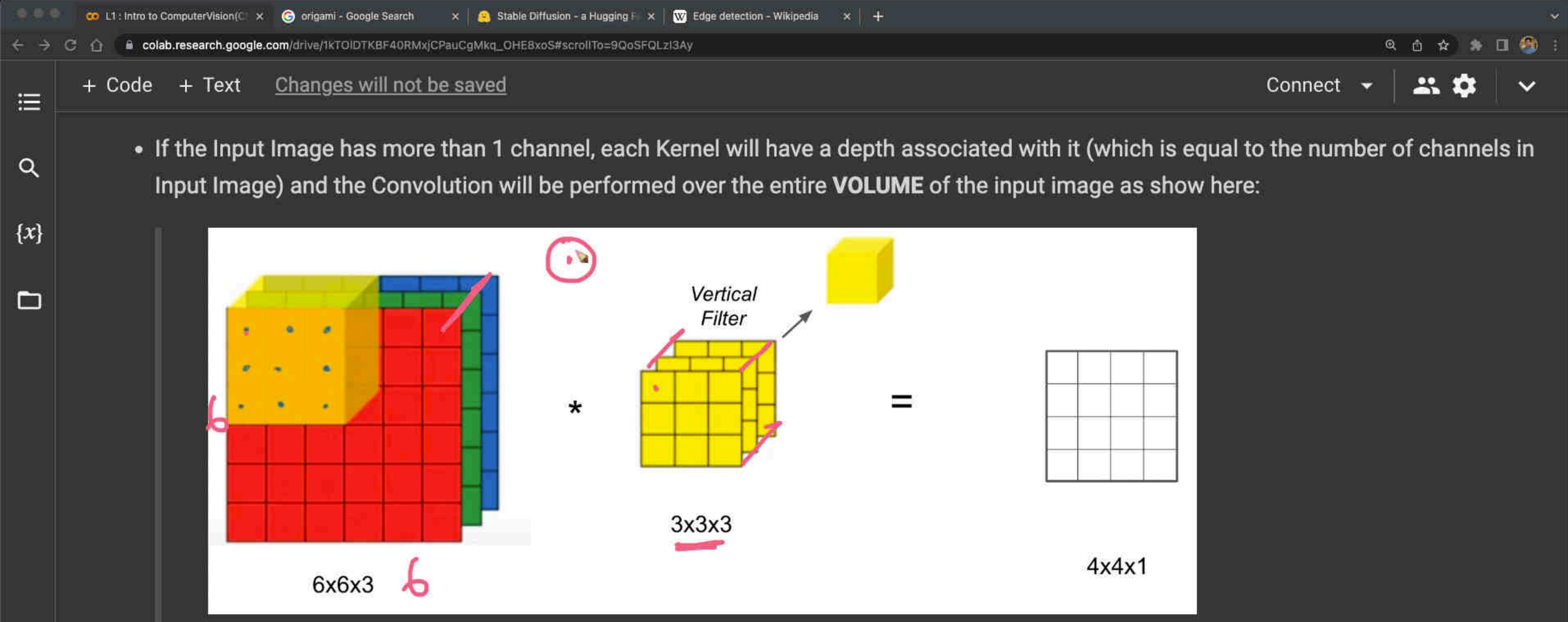
Q. How can we apply both horizontal and vertical edge detector on the same image matrix?

- 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.

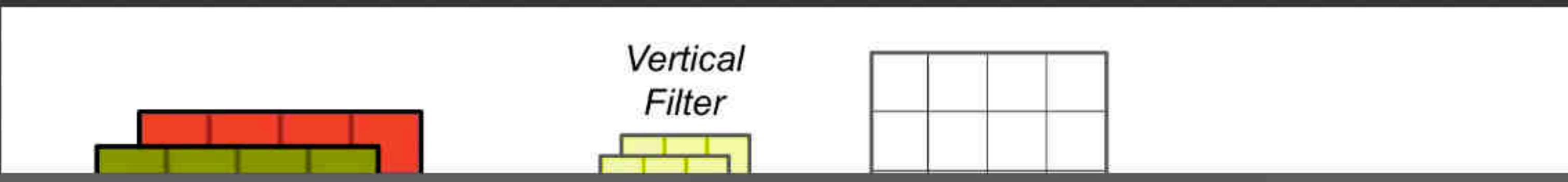


Q. What happens when we have a RGB Image as Input?

CNN!



- In the case of 2 filters, we get an output dimension of $4 \times 4 \times 2$.



L1: Intro to Computer Vision(C) origami - Google Search Stable Diffusion - a Hugging Edge detection - Wikipedia colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=9QoSfQLzI3AY

+ Code + Text Changes will not be saved

Connect

{x}

• 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:

The diagram shows a 3D input volume $6 \times 6 \times 3$ (red, green, blue) being convolved with a $3 \times 3 \times 3$ filter (yellow). The output is a 2D feature map $4 \times 4 \times 1$ (yellow cube). A red checkmark highlights the result.

$6 \times 6 \times 3$

$3 \times 3 \times 3$

$4 \times 4 \times 1$

Vertical Filter

*

=

• In the case of 2 filters, we get an output dimension of $4 \times 4 \times 2$.

The diagram shows a 2D input volume (red and green) being convolved with a $3 \times 3 \times 2$ filter (yellow). The output is a 2D feature map $4 \times 4 \times 2$ (two yellow rectangles).

Vertical Filter

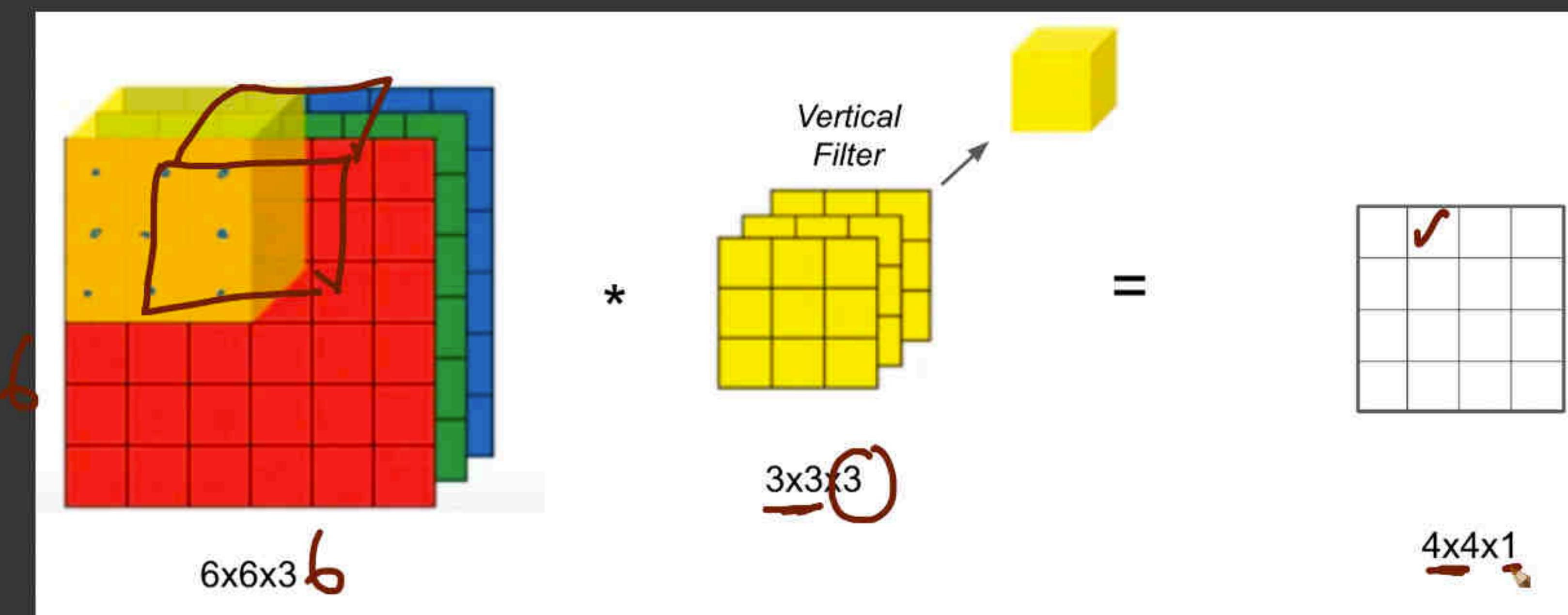


+ Code + Text Changes will not be saved

Connect



- 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 shown here:



- In the case of 2 filters, we get an output dimension of $4 \times 4 \times 2$.



L1: Intro to Computer Vision(C) origami - Google Search Stable Diffusion - a Hugging Edge detection - Wikipedia colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=9QoSfQLzI3AY

+ Code + Text Changes will not be saved

Connect |

• 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 shown here:

The diagram shows a 3D input volume of size $6 \times 6 \times 3$. A handwritten note 'Conv' with a pink asterisk (*) is placed next to the volume. A 'Vertical Filter' of size $3 \times 3 \times 3$ is applied to the volume. The result is a single yellow cube representing the output feature map of size $4 \times 4 \times 1$.

6x6x3

Conv *

Vertical Filter

$3 \times 3 \times 3$

=

4x4x1

• In the case of 2 filters, we get an output dimension of $4 \times 4 \times 2$.

This diagram shows a 3D input volume with two horizontal planes of red and green pixels. Two 'Vertical Filter' blocks are shown, each producing a separate output channel. The final output is a 3D volume of size $4 \times 4 \times 2$, represented by a grid of four columns and two rows of squares.

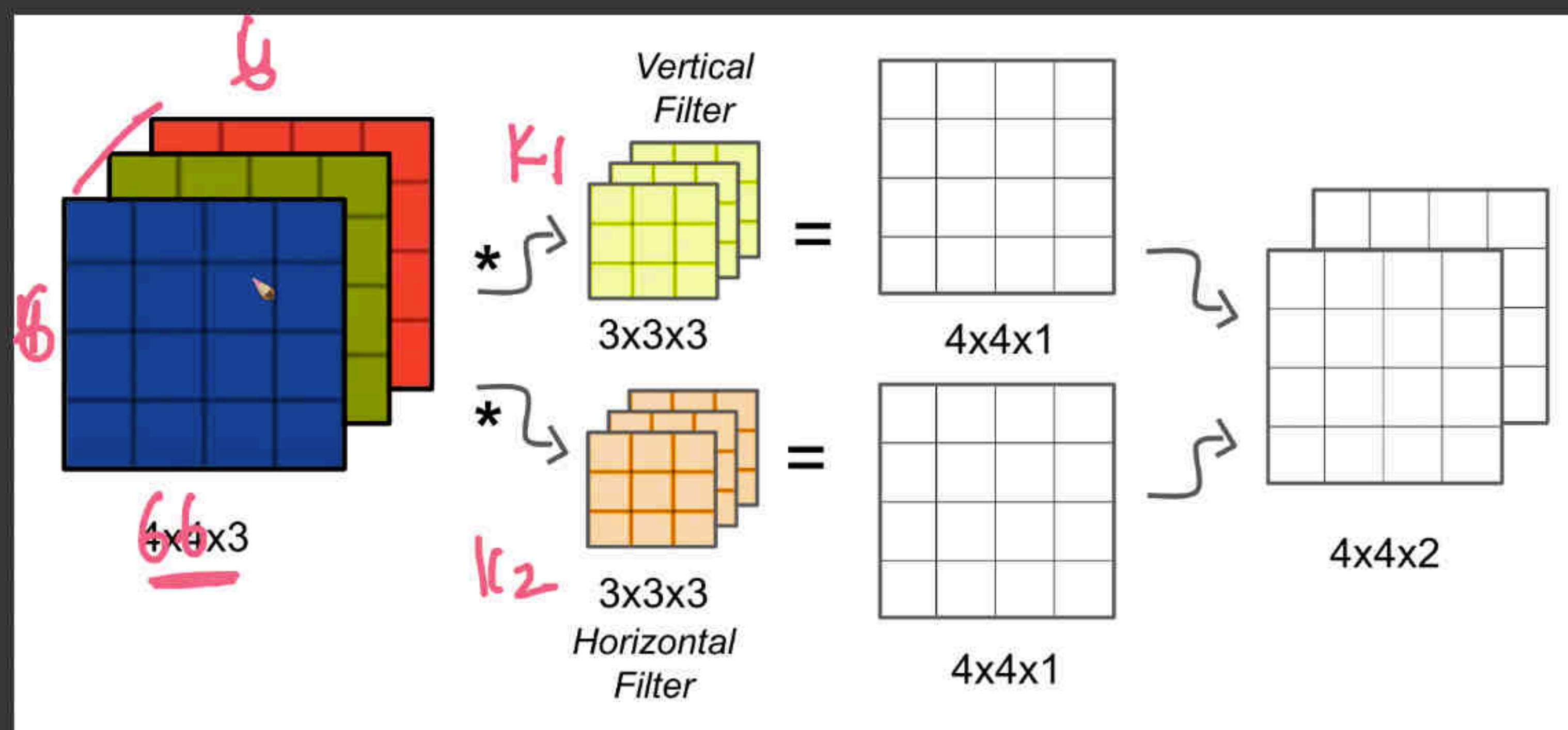
Vertical Filter

$4 \times 4 \times 2$

57 / 57



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



Q. Are these two Kernels enough to know?

Chrome File Edit View History Bookmarks Profiles Tab Window Help

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Edge detection - Wikipedia

colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=9QoSfQLzI3AY

+ Code + Text Changes will not be saved

Connect

Changes will not be saved

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

$\{x\}$

$6 \times 6 \times 3$

\ast

Vertical Filter

$3 \times 3 \times 3$

Kernal

$=$

$4 \times 4 \times 1$

A hand-drawn pink checkmark is present on the right side of the diagram.

- In the case of 2 filters, we get an output dimension of $4 \times 4 \times 2$.

\ast

Vertical Filter

$=$

$4 \times 4 \times 2$

Page-Footer: 59 / 59

L1: Intro to ComputerVision(C) origami - Google Search Stable Diffusion - a Hugging Edge detection - Wikipedia

colab.research.google.com/drive/1kTOIDTKBF40RMxjCPauCgMkq_OHE8xoS#scrollTo=9QoSfQLzI3AY

+ Code + Text Changes will not be saved

Connect

Changes will not be saved

3x3x3

6x6x3

4x4x1

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

Vertical Filter \mathbf{k}_1

\ast

3x3x3

4x4x1

Horizontal Filter \mathbf{k}_2

\ast

3x3x3

4x4x1

4x4x2

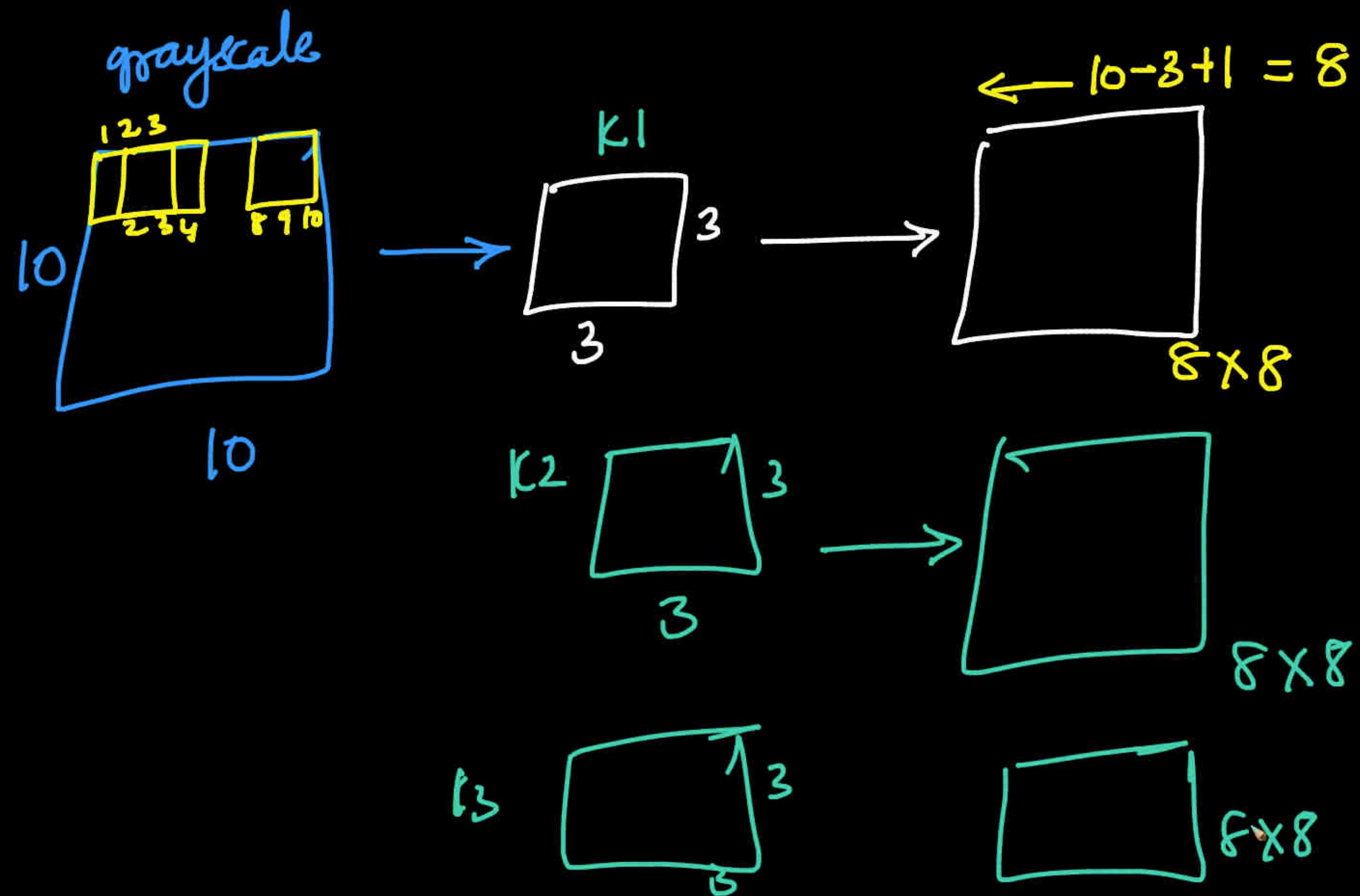
6x6x3

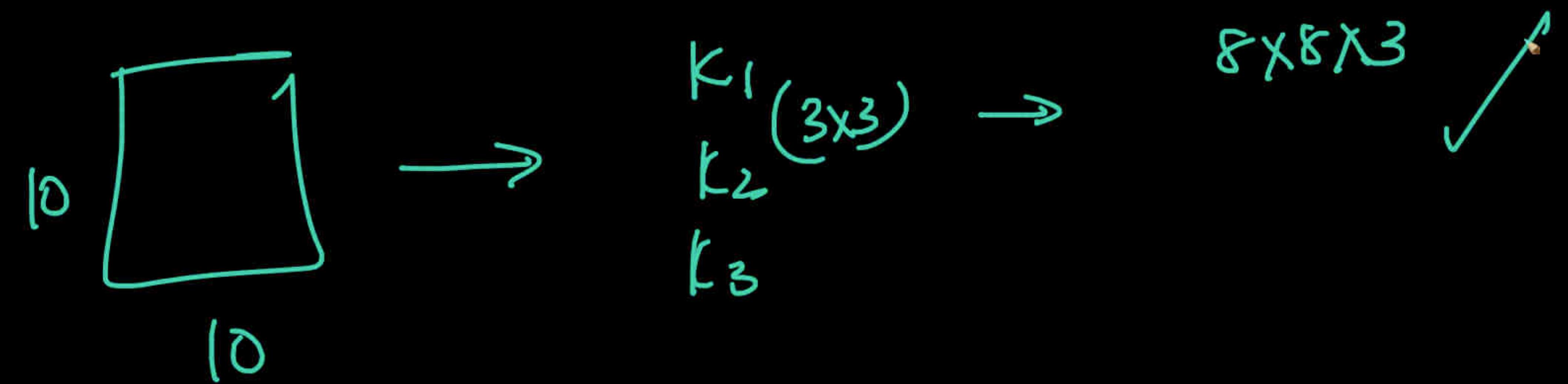
4x4x3

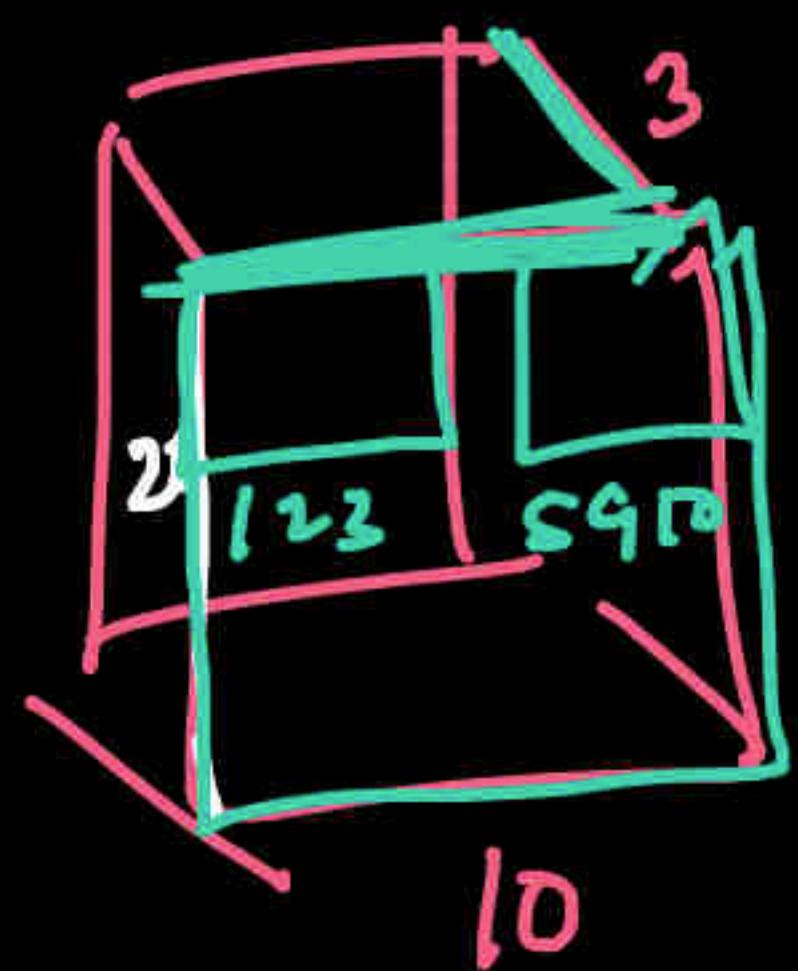
✓

60 / 60

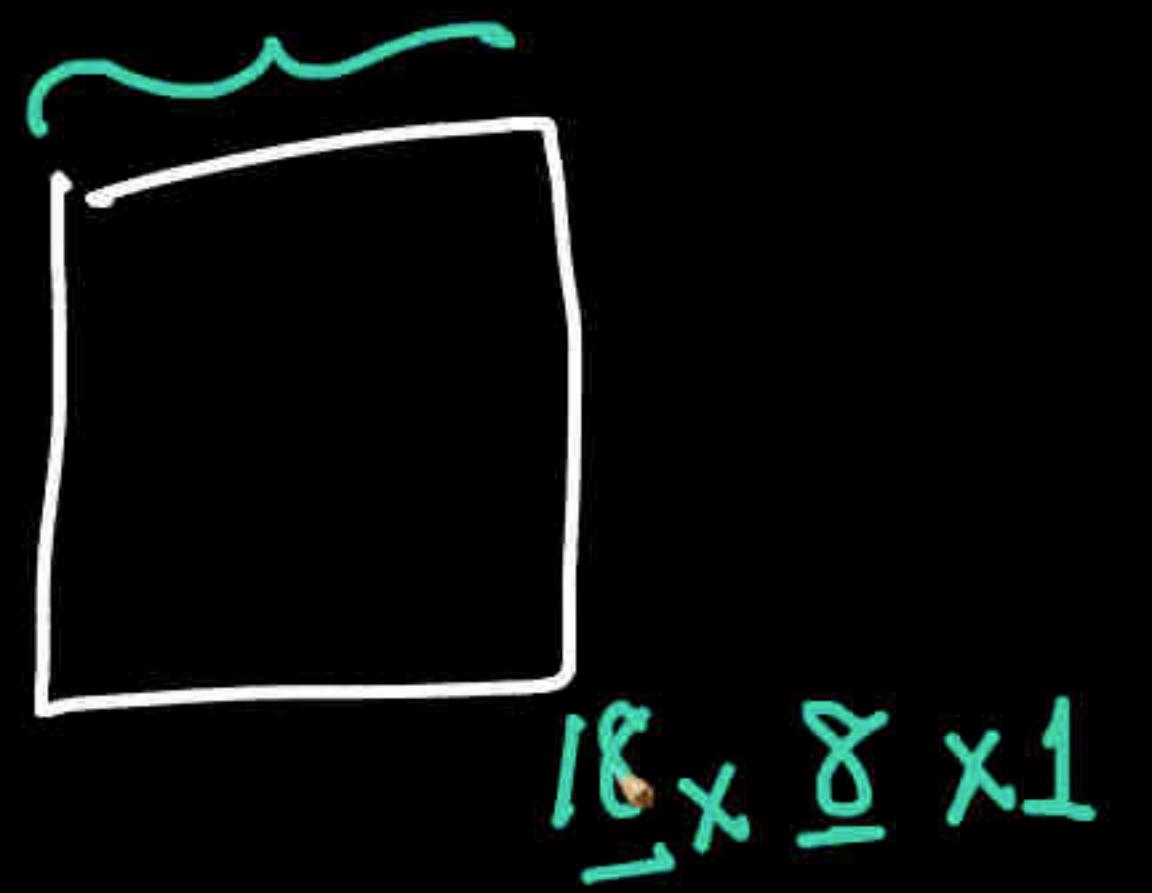
The diagram illustrates the convolution process. It starts with a 6x6x3 input tensor (labeled 6x6x3) and applies two filters: a vertical filter \mathbf{k}_1 and a horizontal filter \mathbf{k}_2 , both of size 3x3x3. The vertical filter produces a 4x4x1 output tensor (labeled 4x4x1), and the horizontal filter also produces a 4x4x1 output tensor. These two outputs are then combined to form a final output tensor of size 4x4x2 (labeled 4x4x2). A large pink checkmark is present on the right side of the diagram.



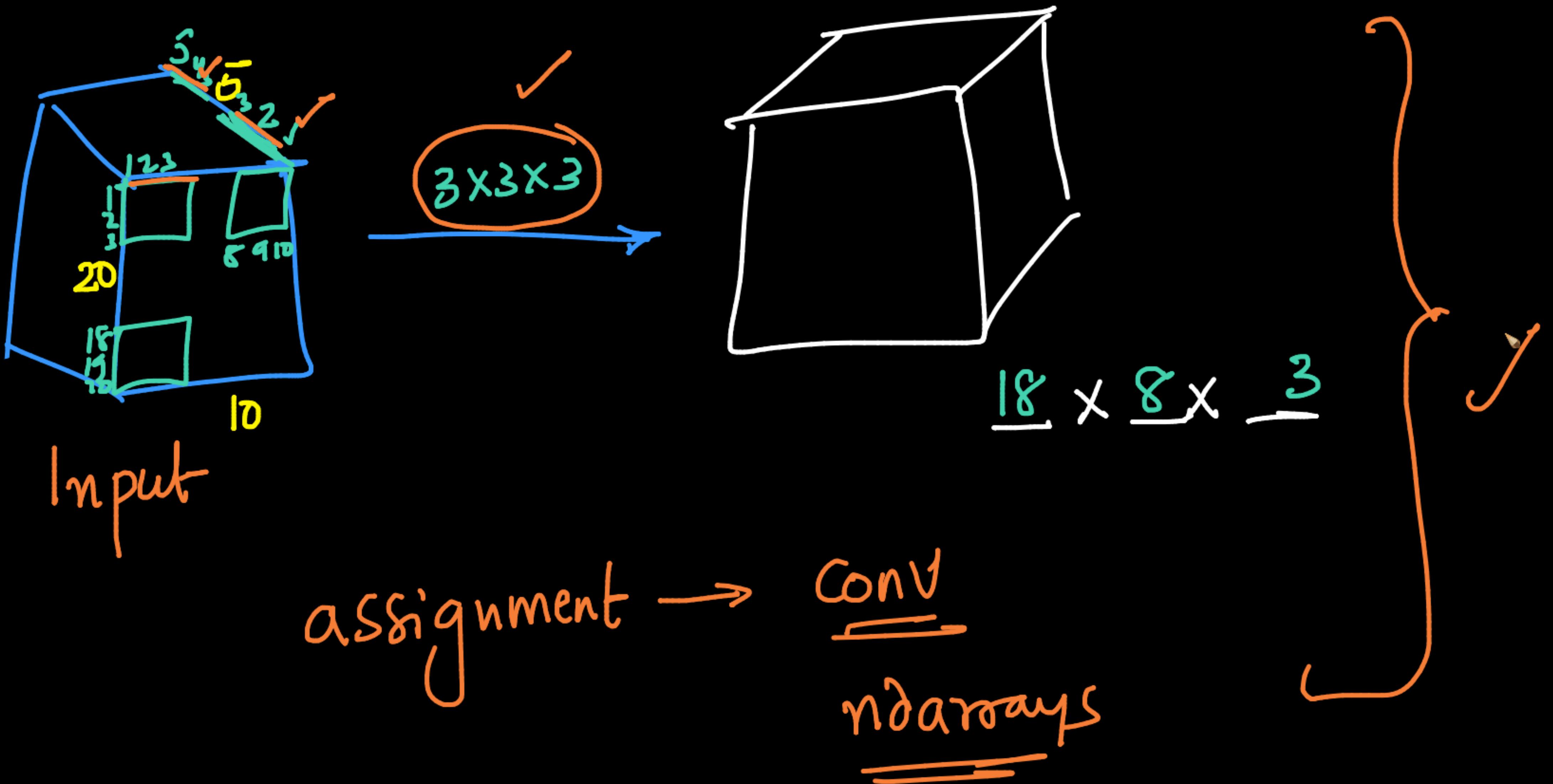




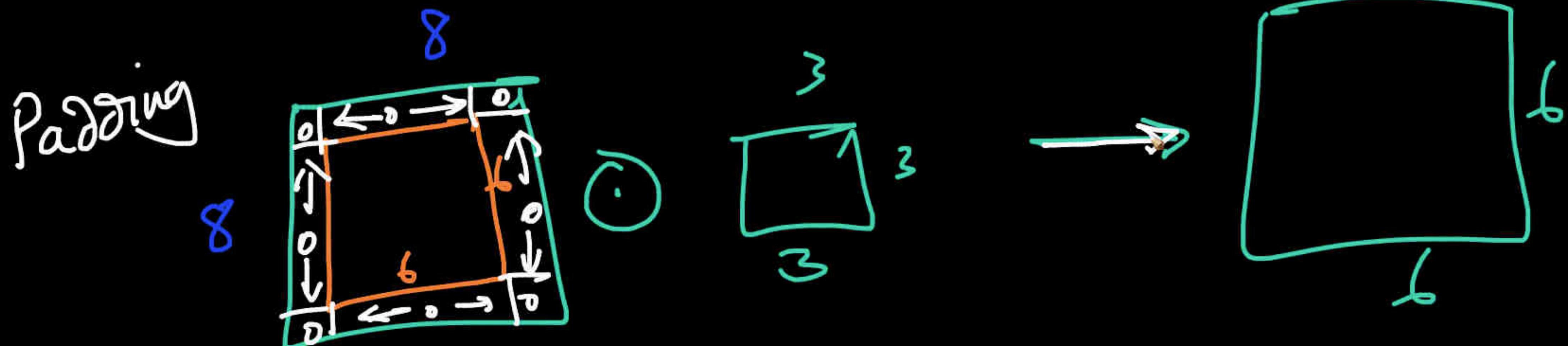
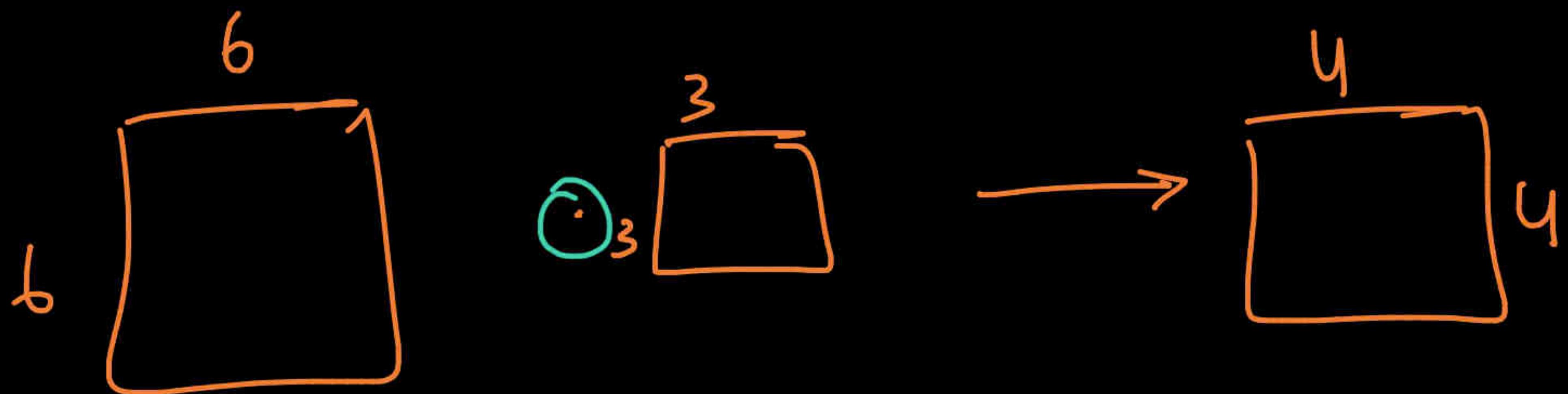
→ $3 \times 3 \times 3$

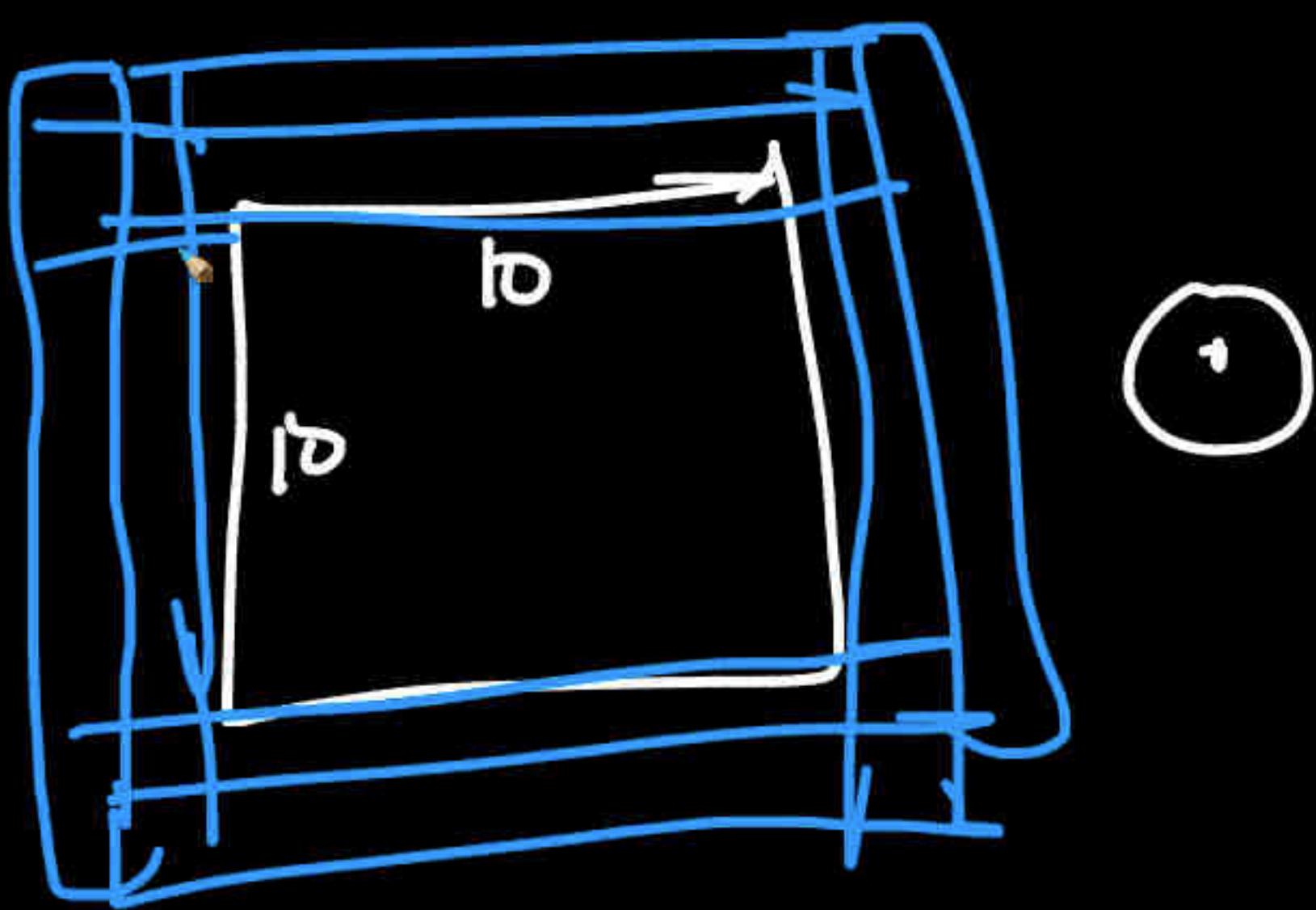


$20 \times 10 \times 3$

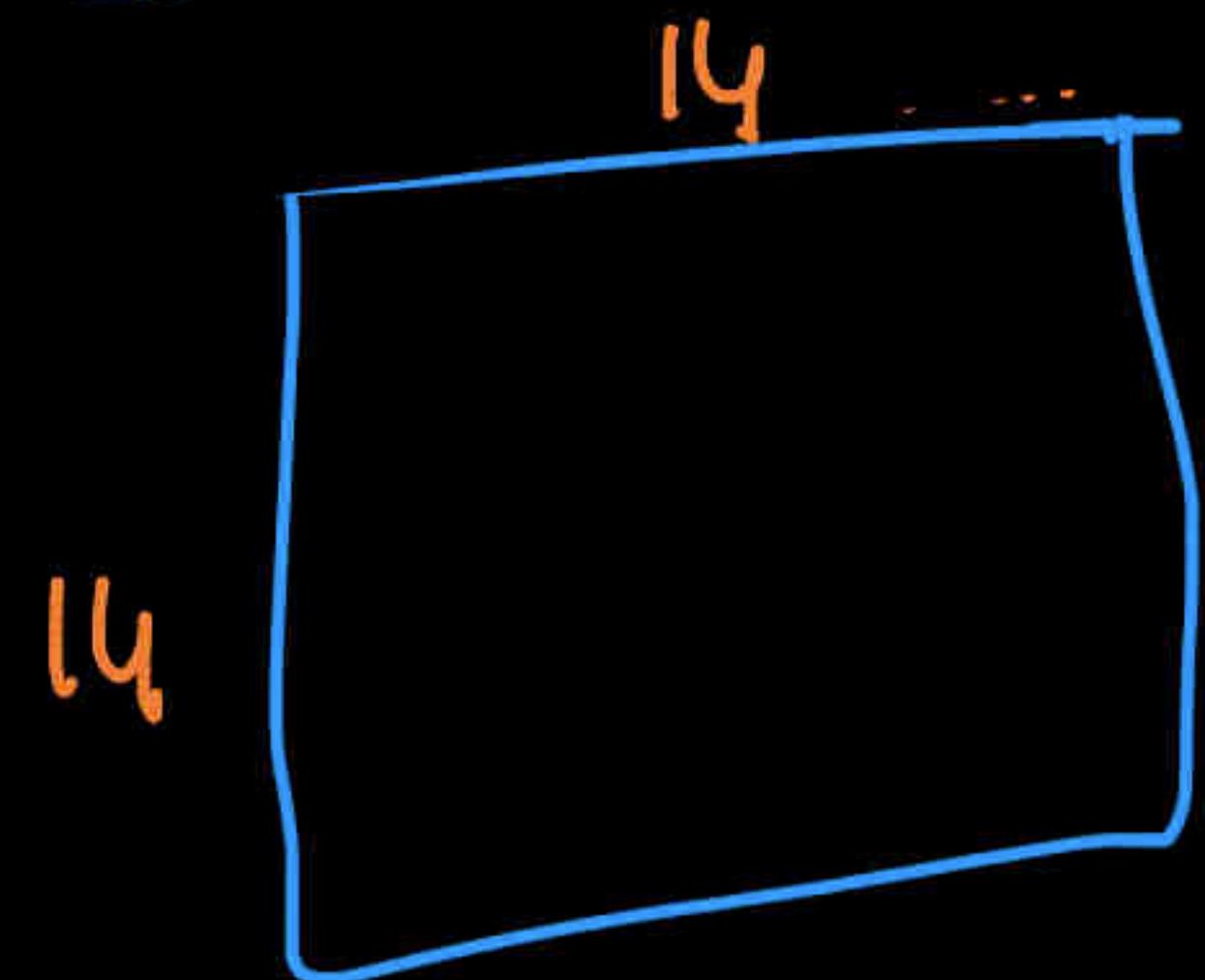
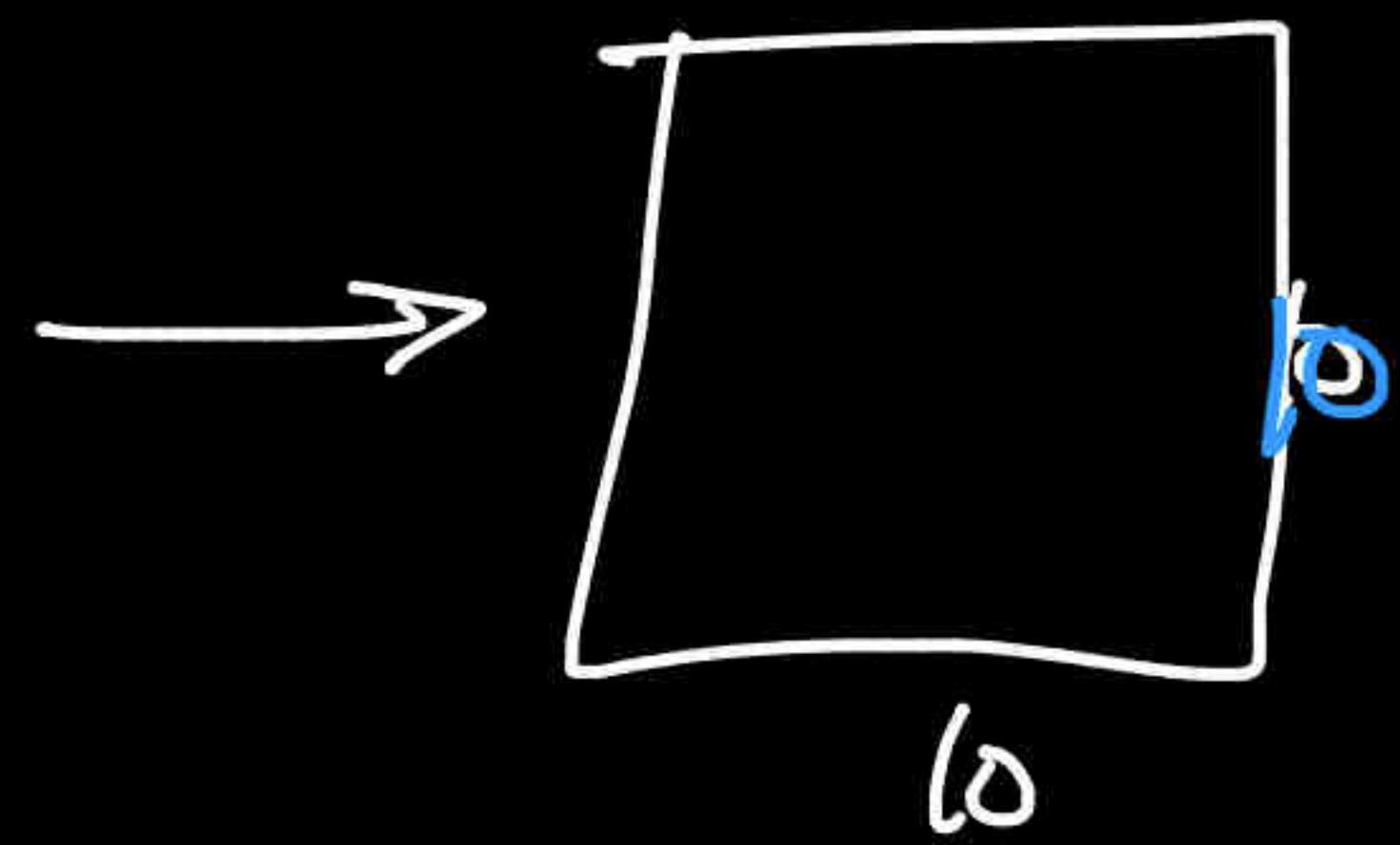


Padding

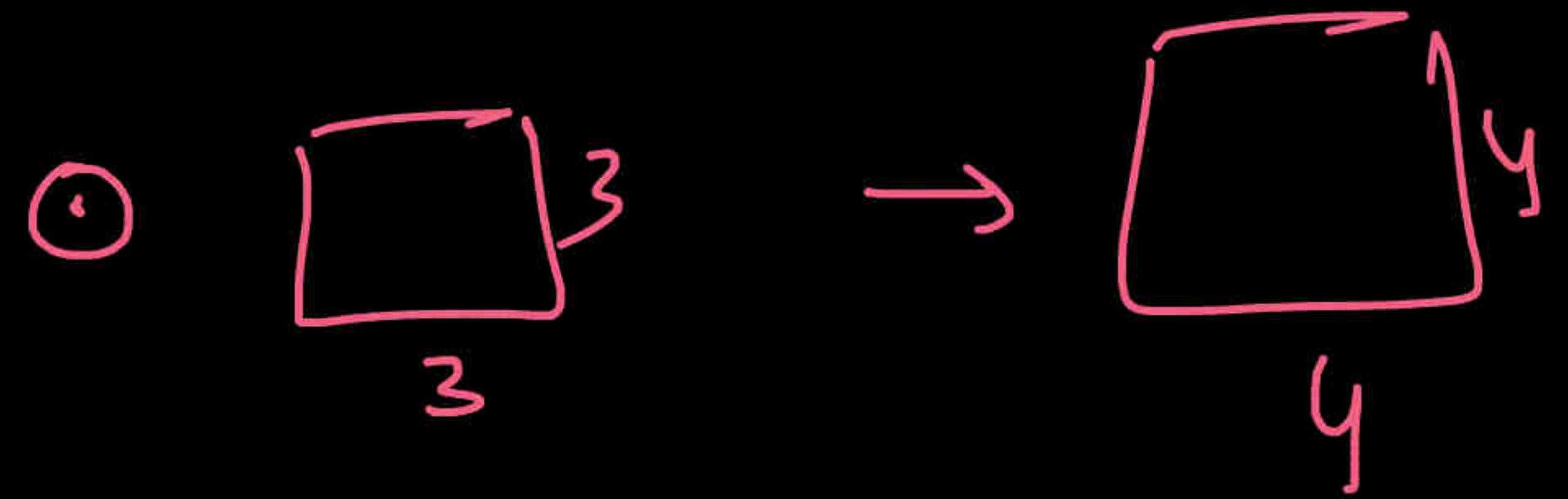
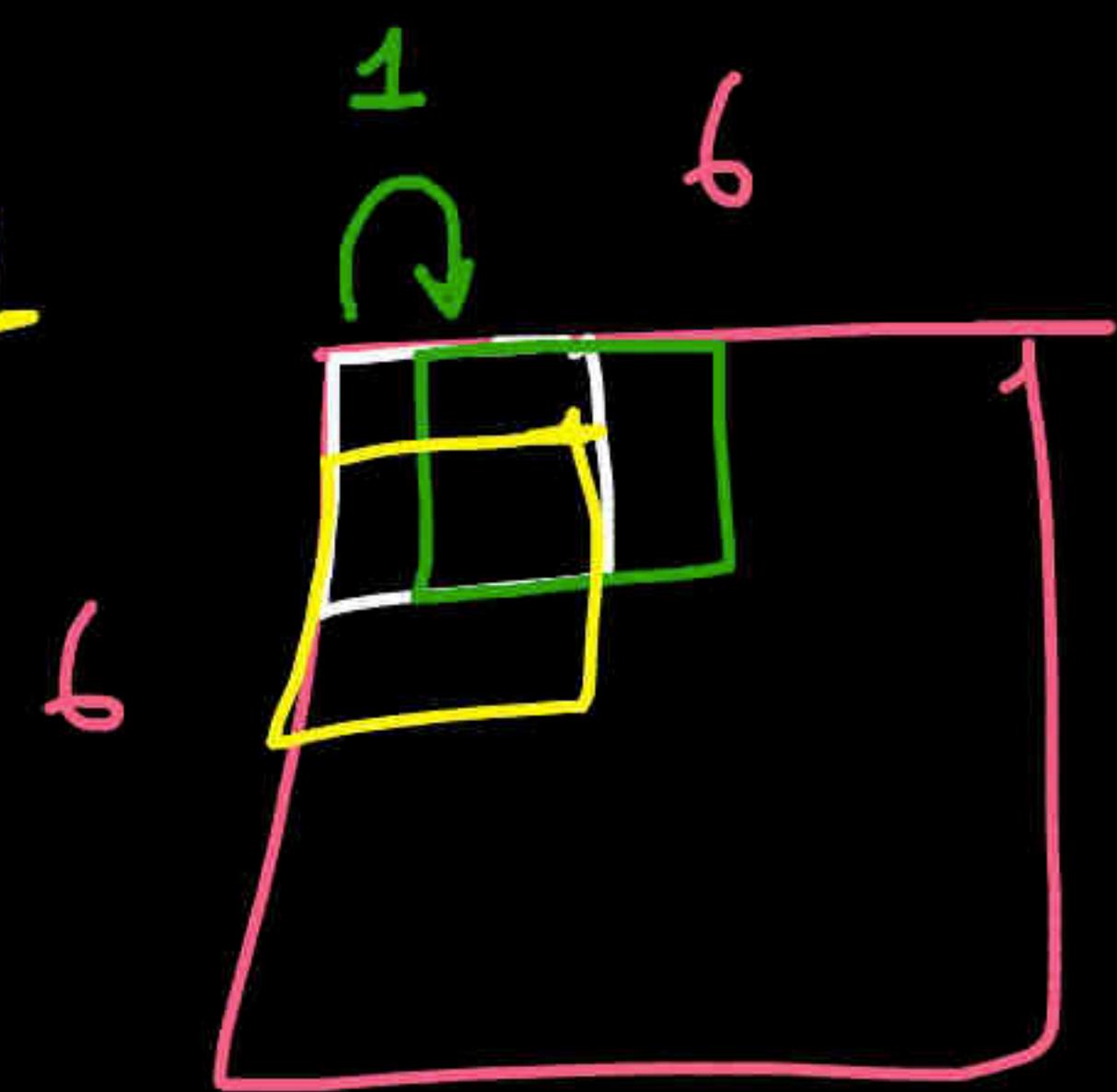


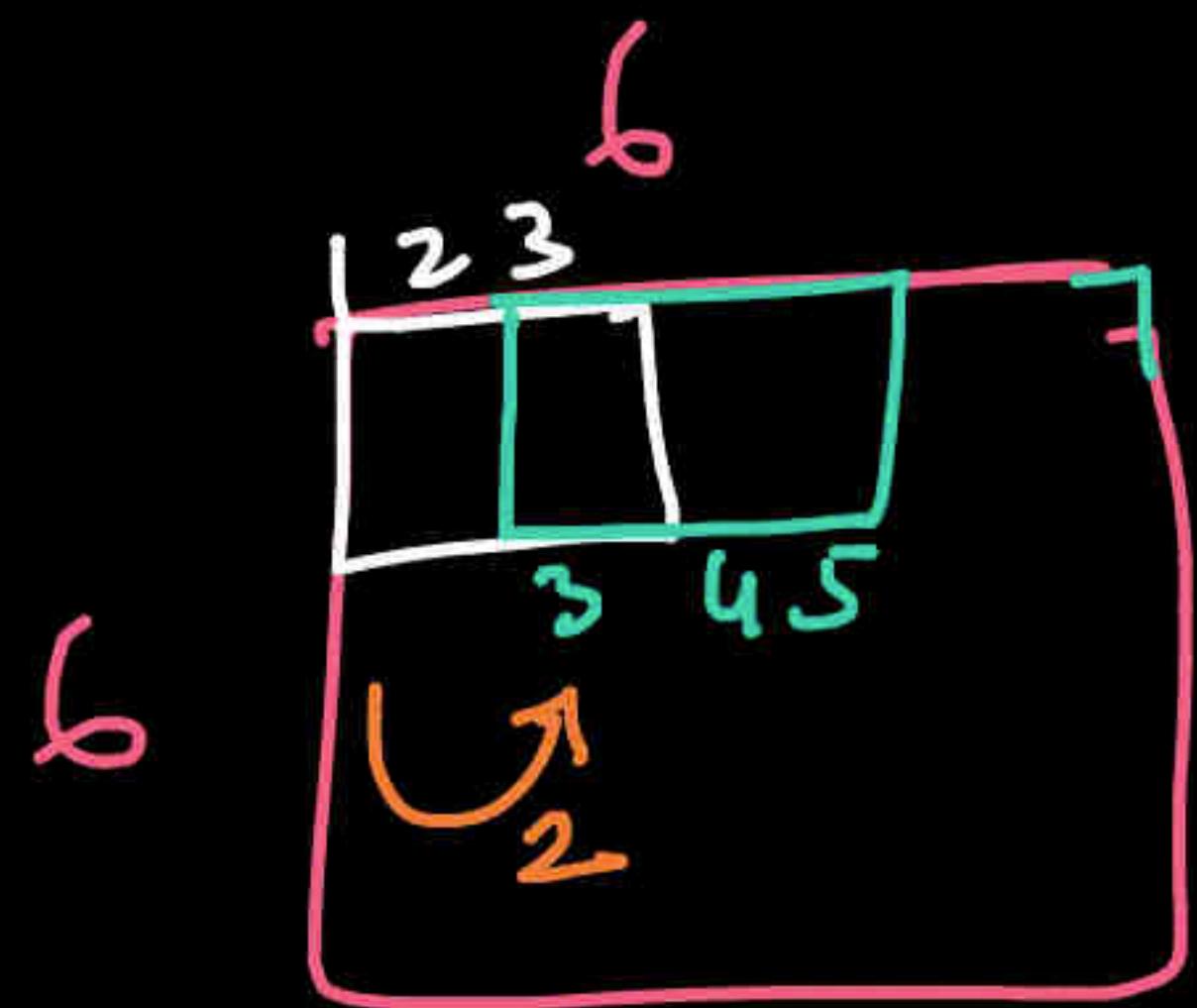


5×5



Stride:
default = 1

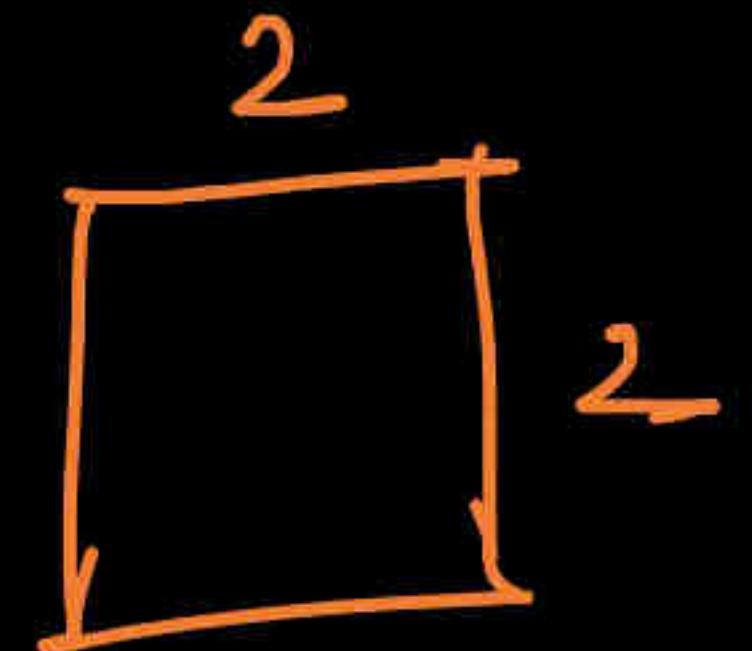




3×3

ReLU

Stride



smaller
output

→ CV

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

→ Classical IP → edge detections (CONV = Matrix mul
+ add
cell-wise)

→ CONV. layers (ReLU, multiple)

→ padding, strides

Next
→ Pooling
→ approach → MLP
→ CNN

CNN!

shape of kernels

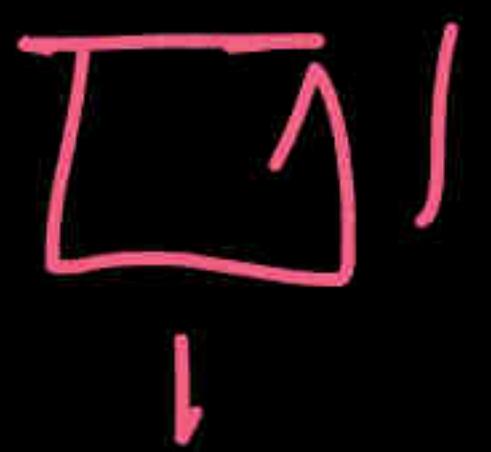
kernels

stride

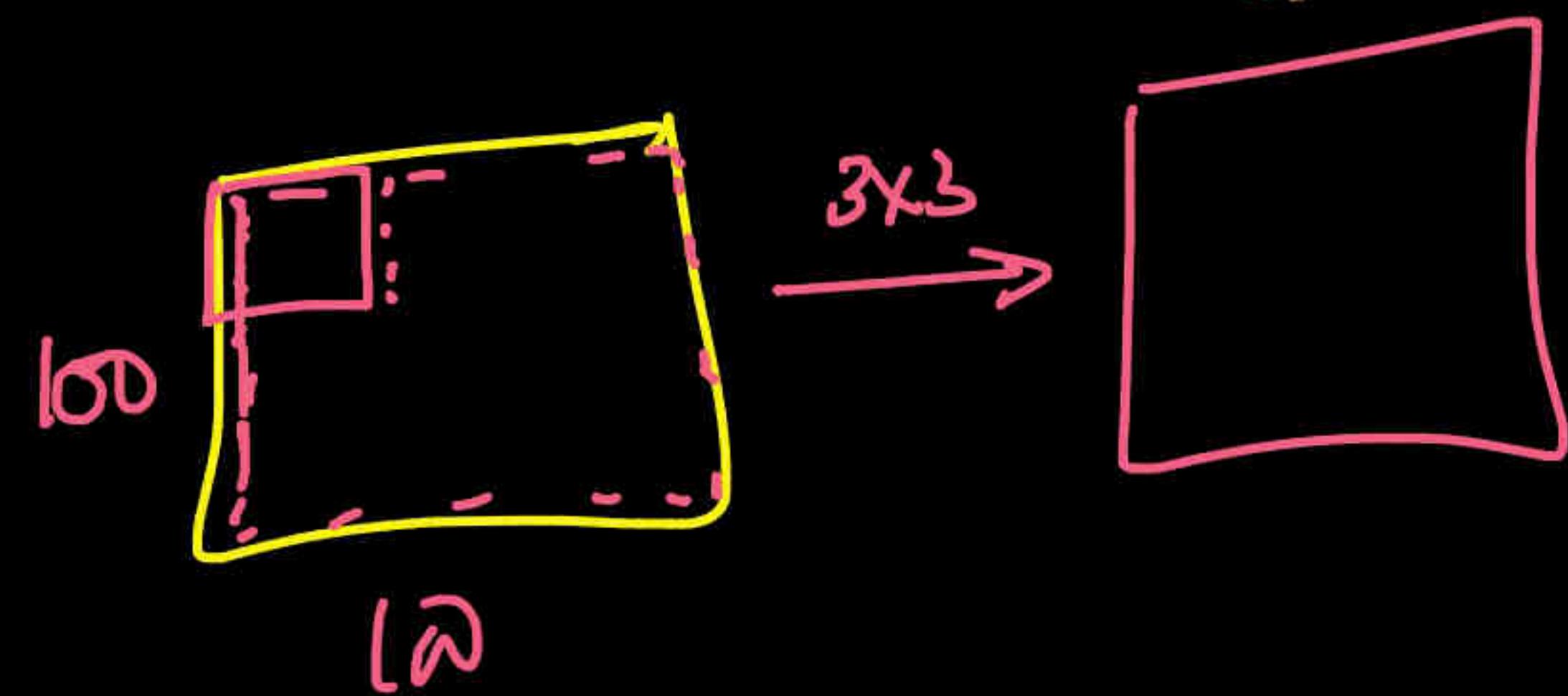
padding



hyper-params

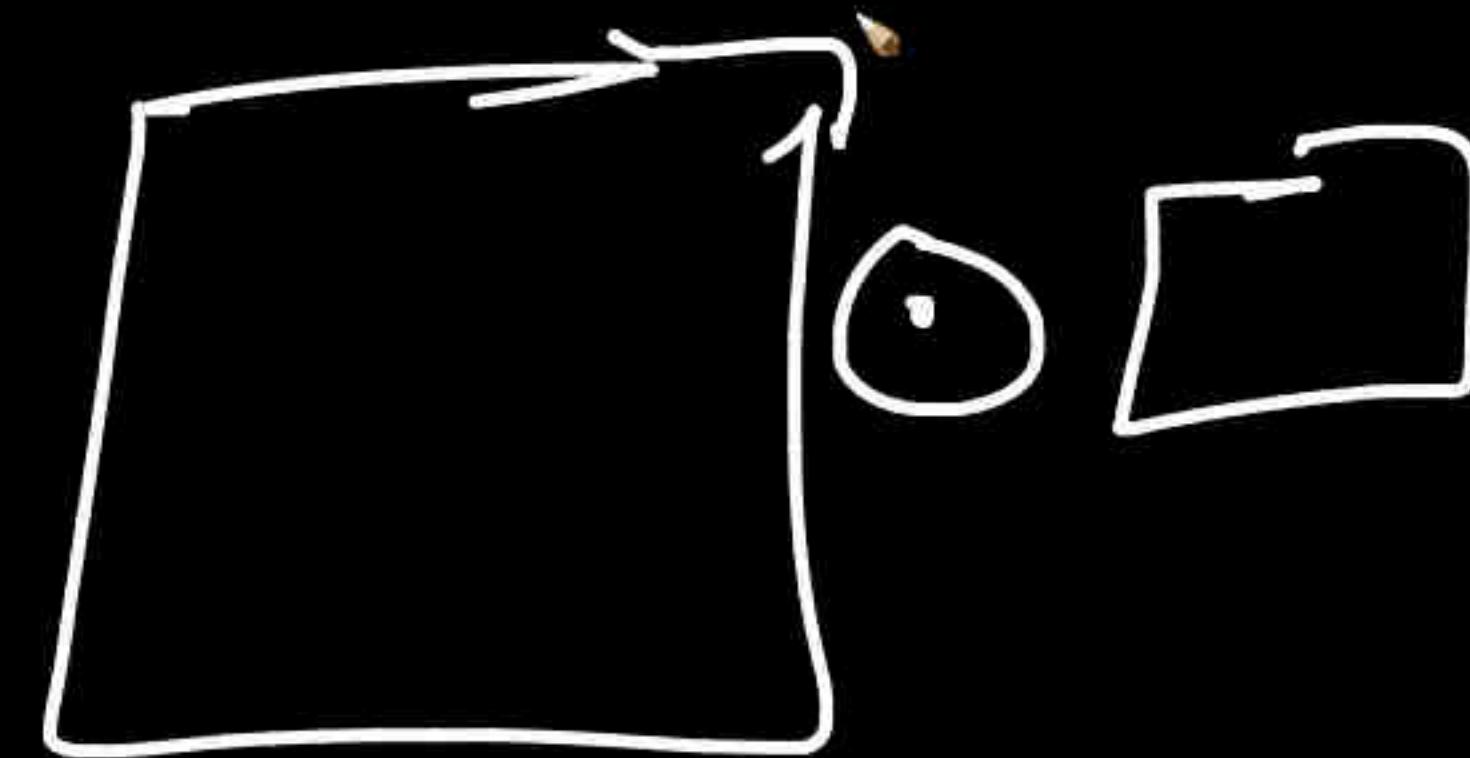
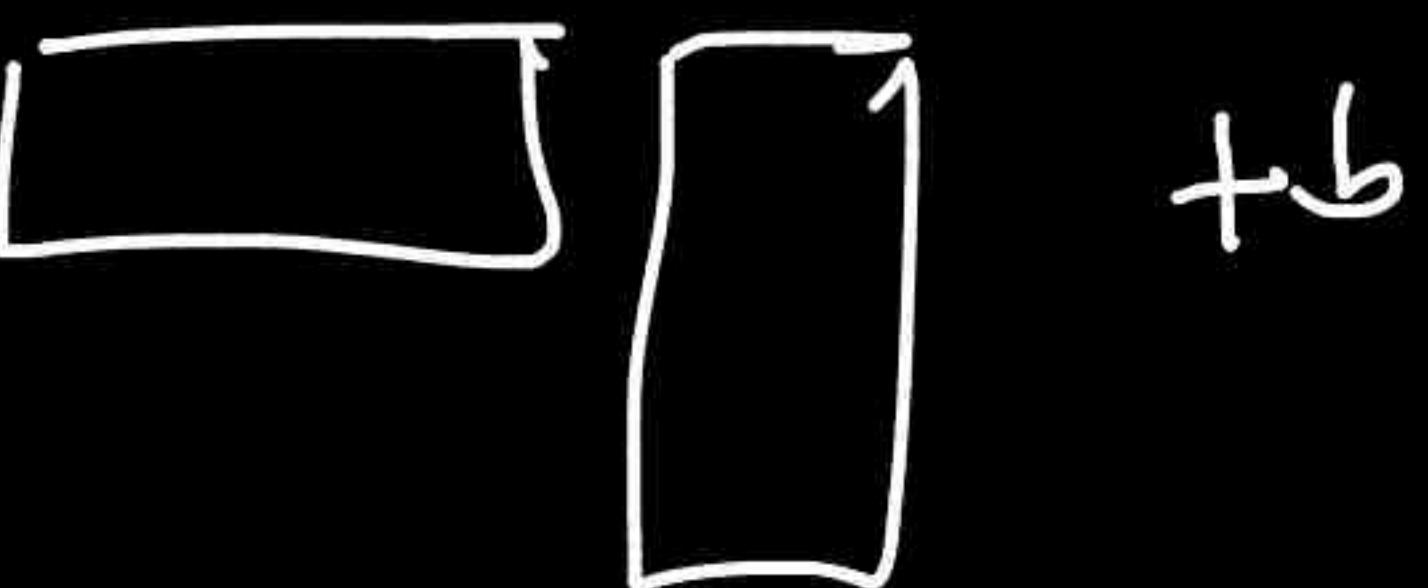


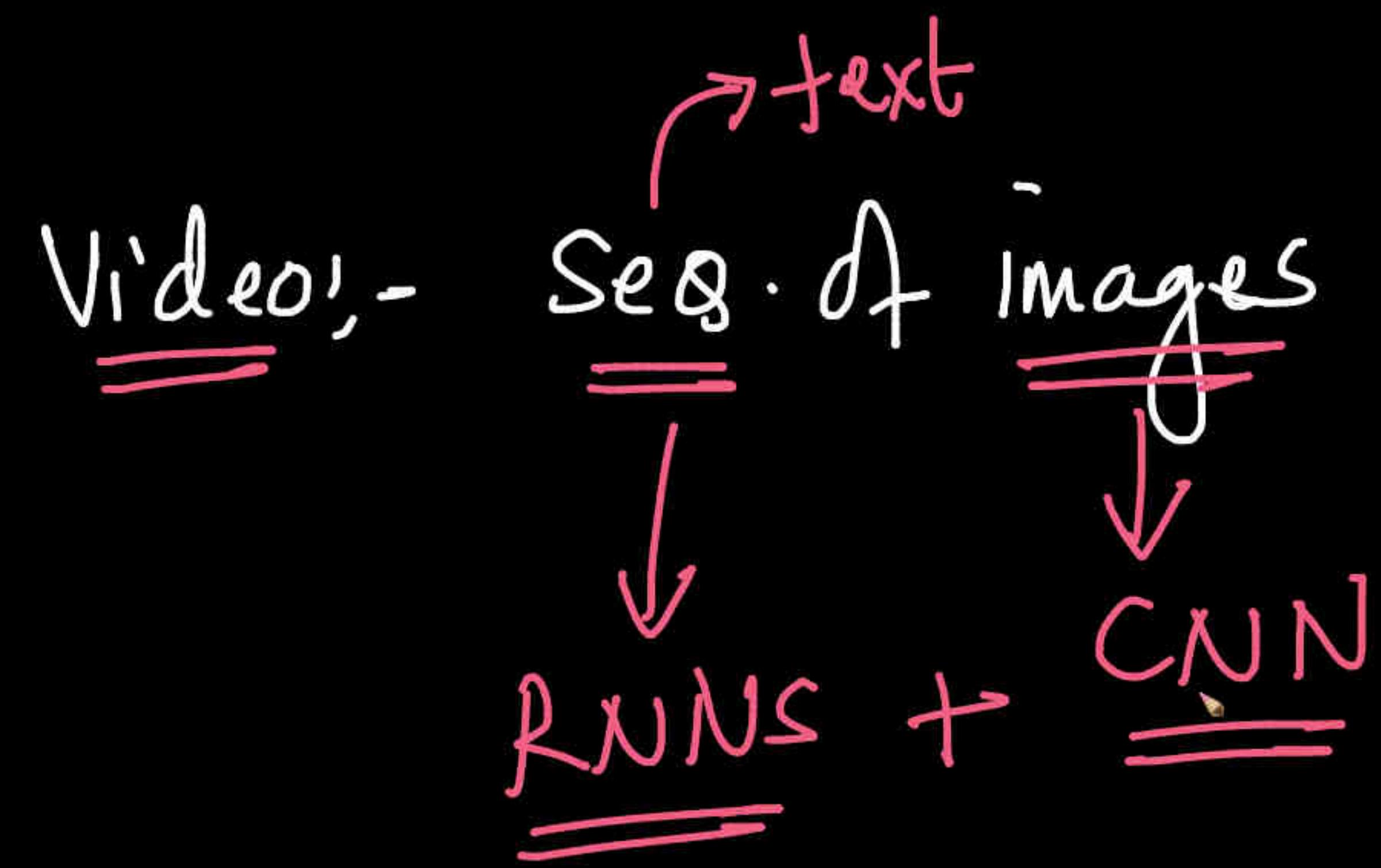
100 layers CNN



✓ { $w^T x_i + b$

l_m ⊙ k





Chrome File Edit View History Bookmarks Profiles Tab Window Help

∞ L1: intro to ComputerVision(C) × origami - Google Search × 🐾 Stable Diffusion - a Hugging Face Model × Edge detection - Wikipedia × +

huggingface.co/spaces/stabilityai/stable-diffusion

For faster generation and forthcoming API access you can try [DreamStudio Beta](#)

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

Generate image

76 / 76

Advanced options Share to community