17th March, 2023

1/9

DSML : Computer Vision

Introduction to Computer Vision : CNNs

Class starts @ 9:05 pm.

About me :

Name : Dhruv. Jawali.

Qualifications :

* B.Tech (CSA) from NIT Goa.
* PhD, Indian Institute of Science.
  (learning Filters, Filterbanks, wavelets and Multiscale Representations.)

Work Experience :

* Software engineer, Samsung Research.
* Deep learning Researcher, Neuropixel.ai.
* ML Engineer and Instructor @ Scaler.

Interests : ♥

- Coding, C.V. Research (3D)
- Problem Solving
- Teaching.

Agenda for today:

* Introduction to the field.
* Inspiration for CNNs: Neuroscience!
* Traditional Neural Networks vs. CNNs.
* Deep Dive:
  - → Convolution.
  - → Pooling
  - → CNN architectures.

Computer Vision as a field of study

Applied Math. → Signal processing. → Image Processing. → Computer Vision

Applied field.

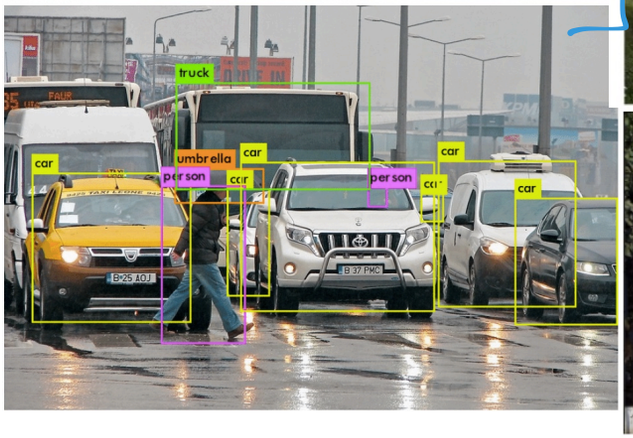
Typical problems in Computer Vision.

A brief history of Computer Vision

1956	"Artificial Intelligence"
1959	Neuropsychology: Discover vision is hierarchical
1960s	First computer vision project (MIT summer project)
1970s	"AI winter"
1990s	Rudimentary neural networks
2001	Face detection (Viola & Jones)
2012	AlexNet (Krizhevsky et al.) VGG, GoogLeNet, ResNet, SENet...

Lack of computational resources.

↑ SGD GPU.



+ NLP

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

Classification → Detection → Segmentation (semantic.) → Image Generation → Generation with multimodal prompts.

Matching → Captioning.

Increasing order of Difficulty

How does a computer "see"?

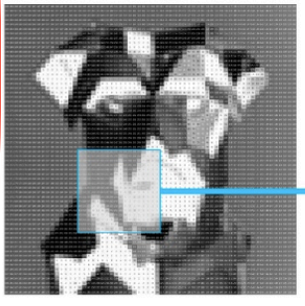
0 - 255 → values in pixels.

↳ Represented?

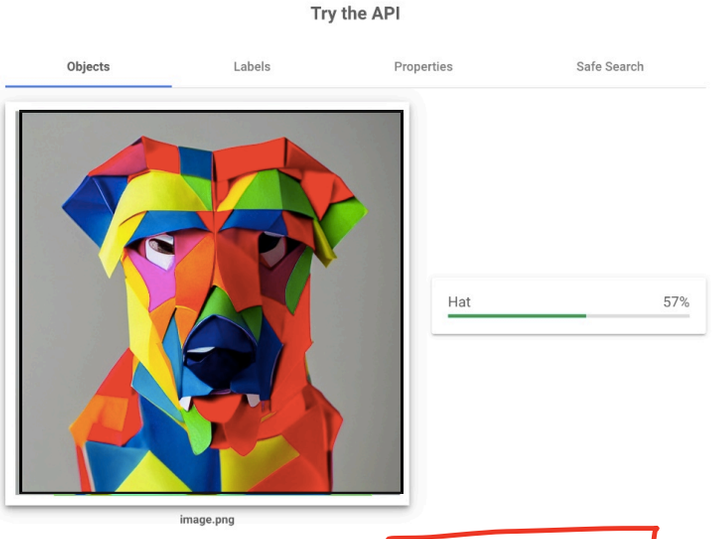
Pixel → entry.

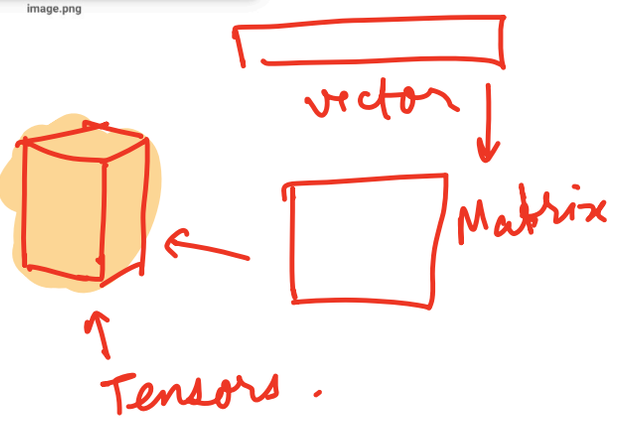
2-D array / 3-D array

grayscale / color



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





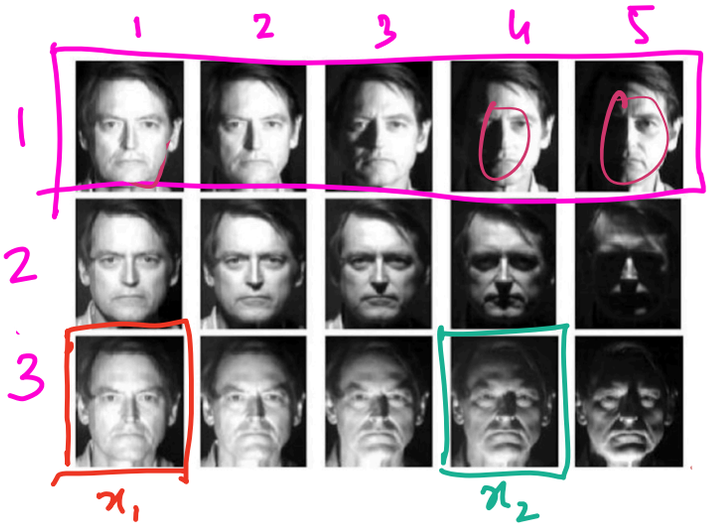
↳ All color images are 3-D arrays.

(height, width, channels).

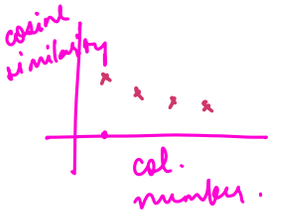
$\rightarrow (256)^3$

What makes a computer vision task difficult?

↳ occlusions (loss from 3D → 2D).



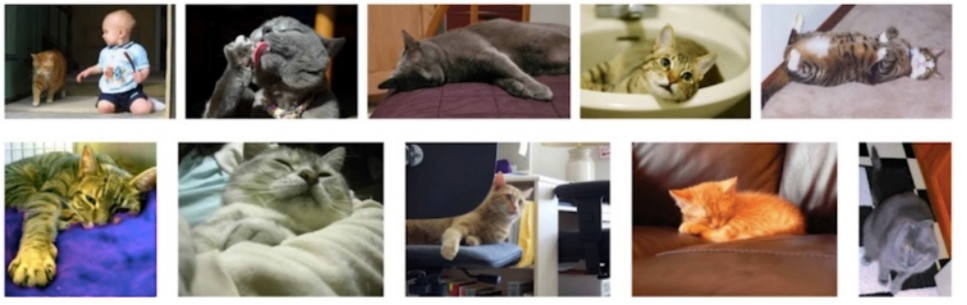
(1,1) (1,2)



→ $f(x_1, x_2)$

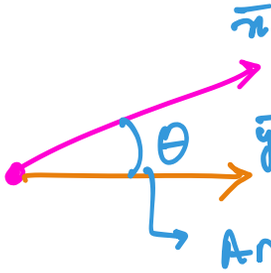
↳ 1

"How to get a similarity measure so that all



→ Different "deformation" images are pointing to the same object?"

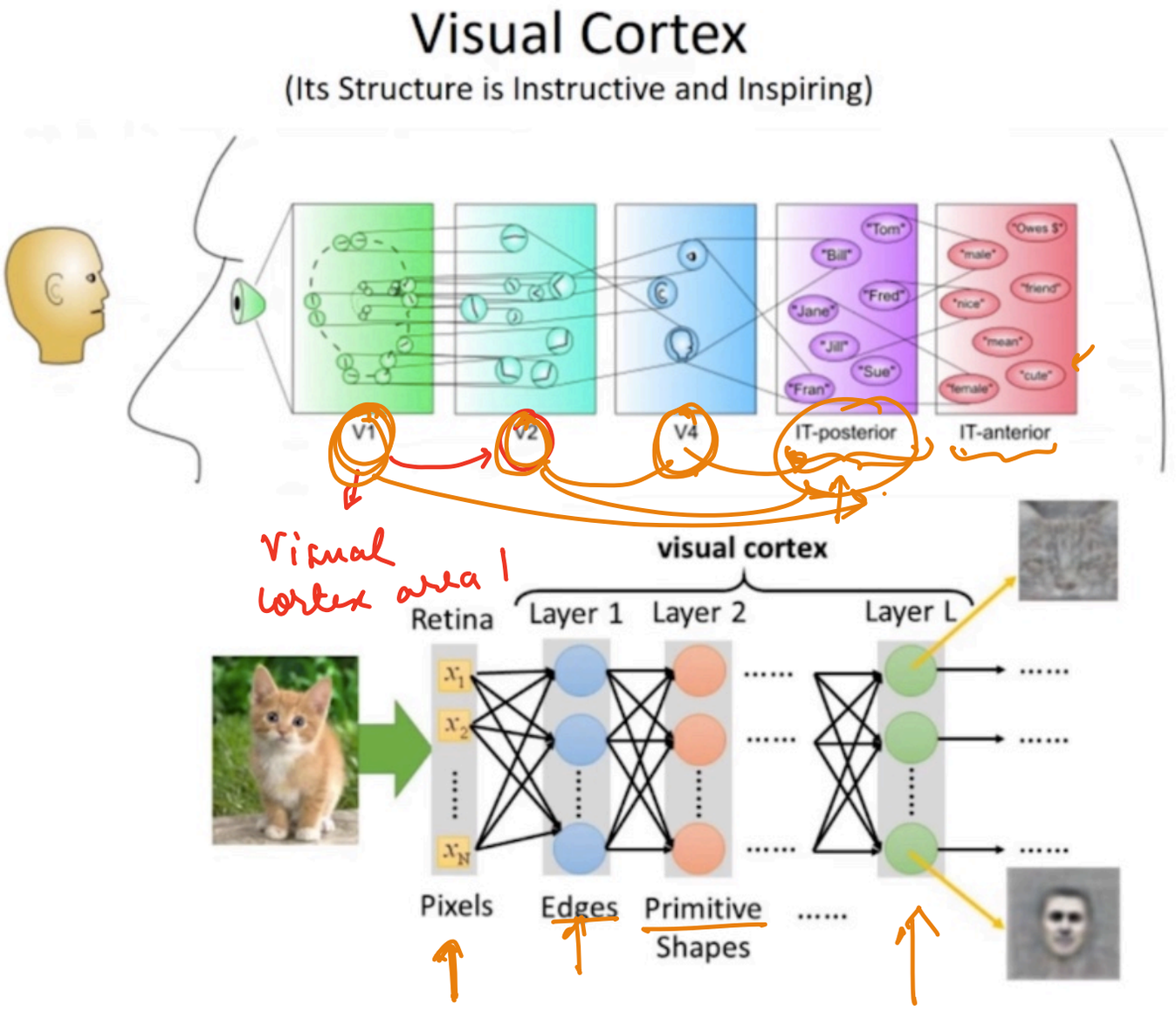
Math:



} Similarity measure.

$$\cos(\theta) = \frac{\bar{x}^T \bar{y}}{\|\bar{x}\| \cdot \|\bar{y}\|}$$

Inspiration: The visual cortex:



Deep dive: Using Neural Networks for Classification.

flatten_1_input InputLayer	input: output:	[(None, 128, 128, 3)]	[(None, 128, 128, 3)]

↓

flatten_1 Flatten	input: output:	(None, 128, 128, 3)	(None, 49152)

↓

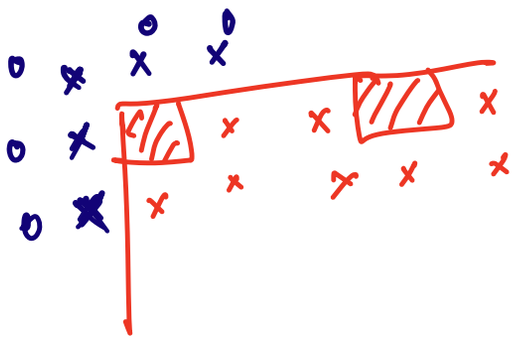
dense Dense	input: output:	(None, 49152)	(None, 1024)

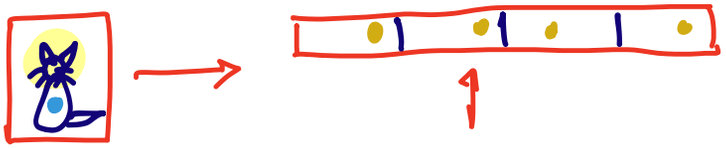
↓

dense_1 Dense	input: output:	(None, 1024)	(None, 256)

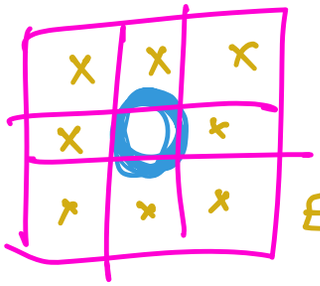
↓

dense_2 Dense	input: output:	(None, 256)	(None, 10)





I am going to lose positional information if I flatten the image.



← neighbourhood of a pixel.

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

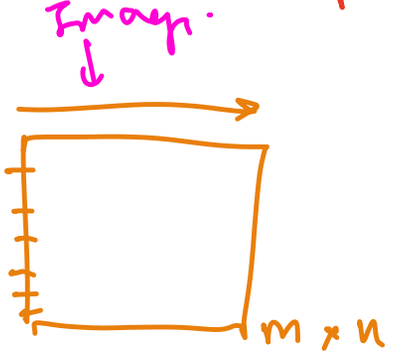
kernel.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

3×3

6×6.

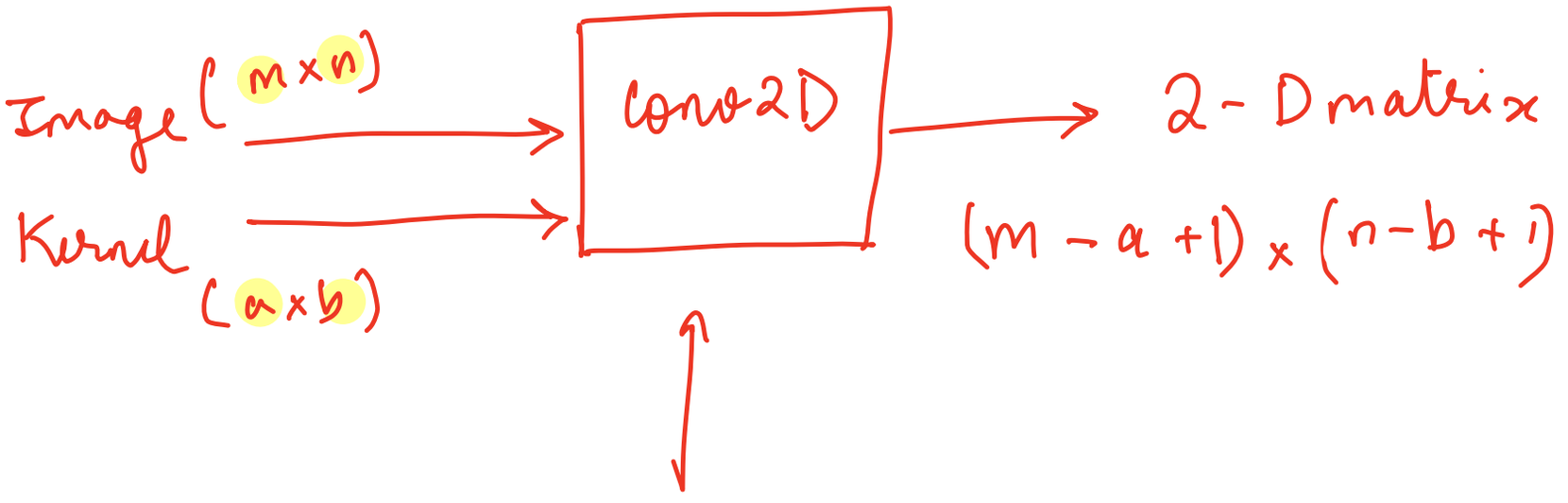
No. of shifts : 4 horizontal,
4 vertical.



$m - a + 1$ (vertical).

$n - b + 1$ (horizontal).

Convolution (2D)



"VALID / CONVOLUTION"

2 more types: (a) SAME TYPE CONV.

(b) FULL TYPE CONV.

kernel

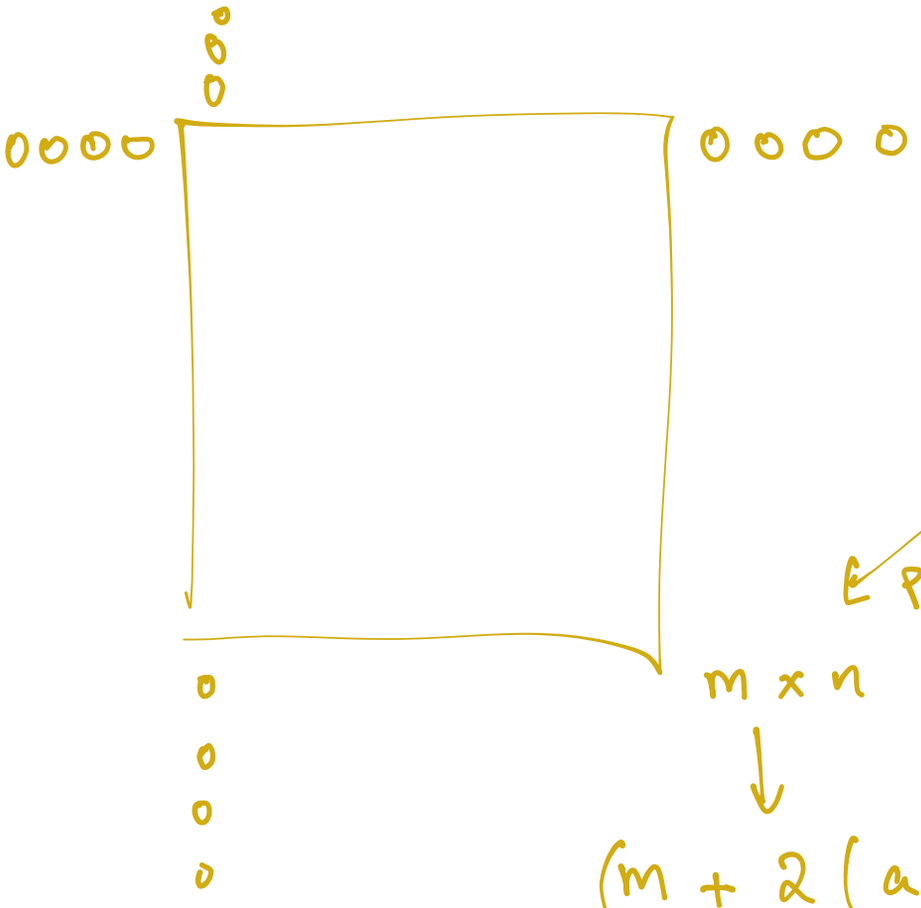
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

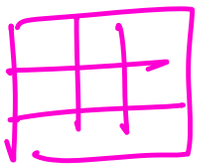
3×3 — 2×2

5×5 — 4×4

a×b — (a-1), (b-1)

2 rows,
2 cols.





a x b.

(a - 1) x (b - 1)

pad.

m + (a - 1)

m x n

(m + 2(a-1)) x (n + 2(b-1))

"FULL TYPE"

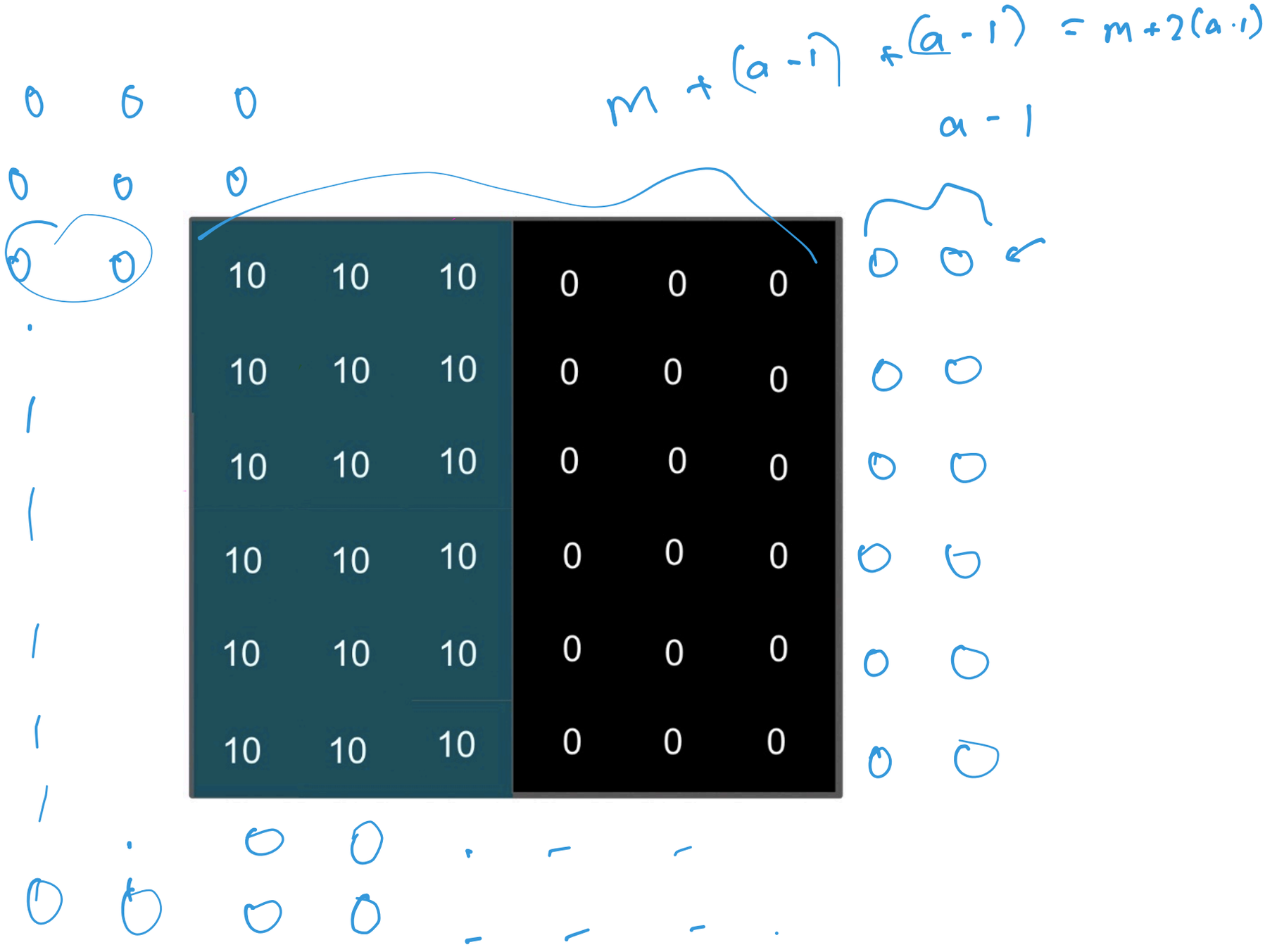
Image (m x n) →

Kernel (a x b) →

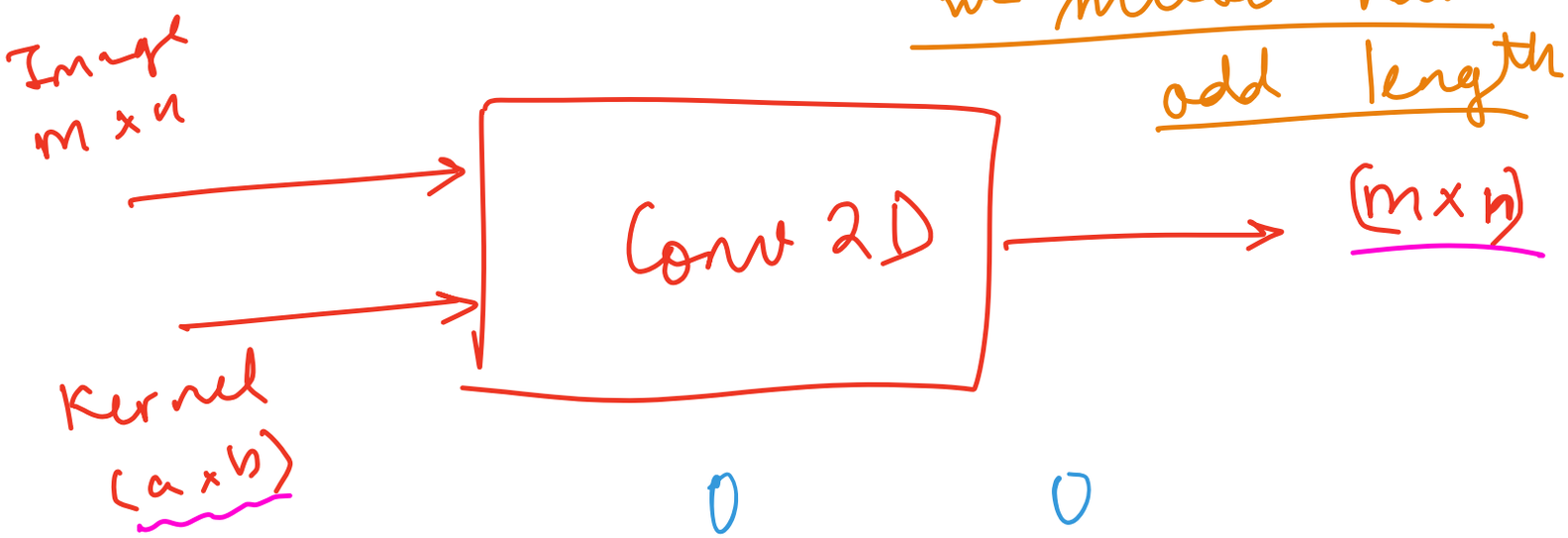
Conv2D → 2-D matrix

(m + a - 1) x (n + b - 1)

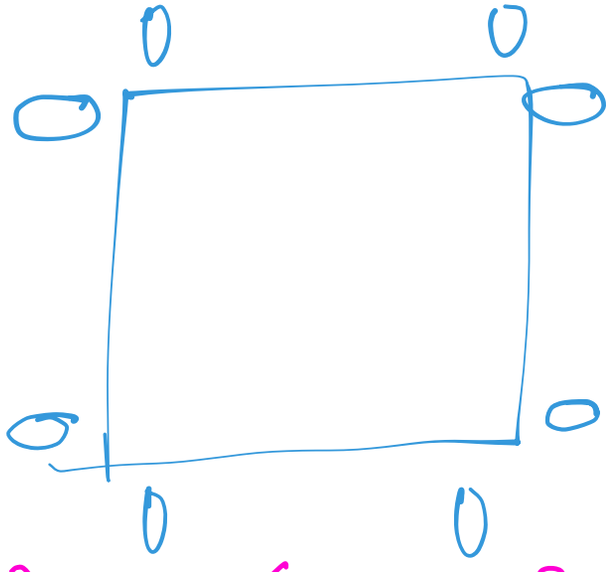
n


m + (a-1) + (a-1) = m + 2(a-1)
a - 1
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

"Same" convolution ← To do same convolution, we must have odd length!



(p, q) ← padding



$m + 2p - (a-1) = m$

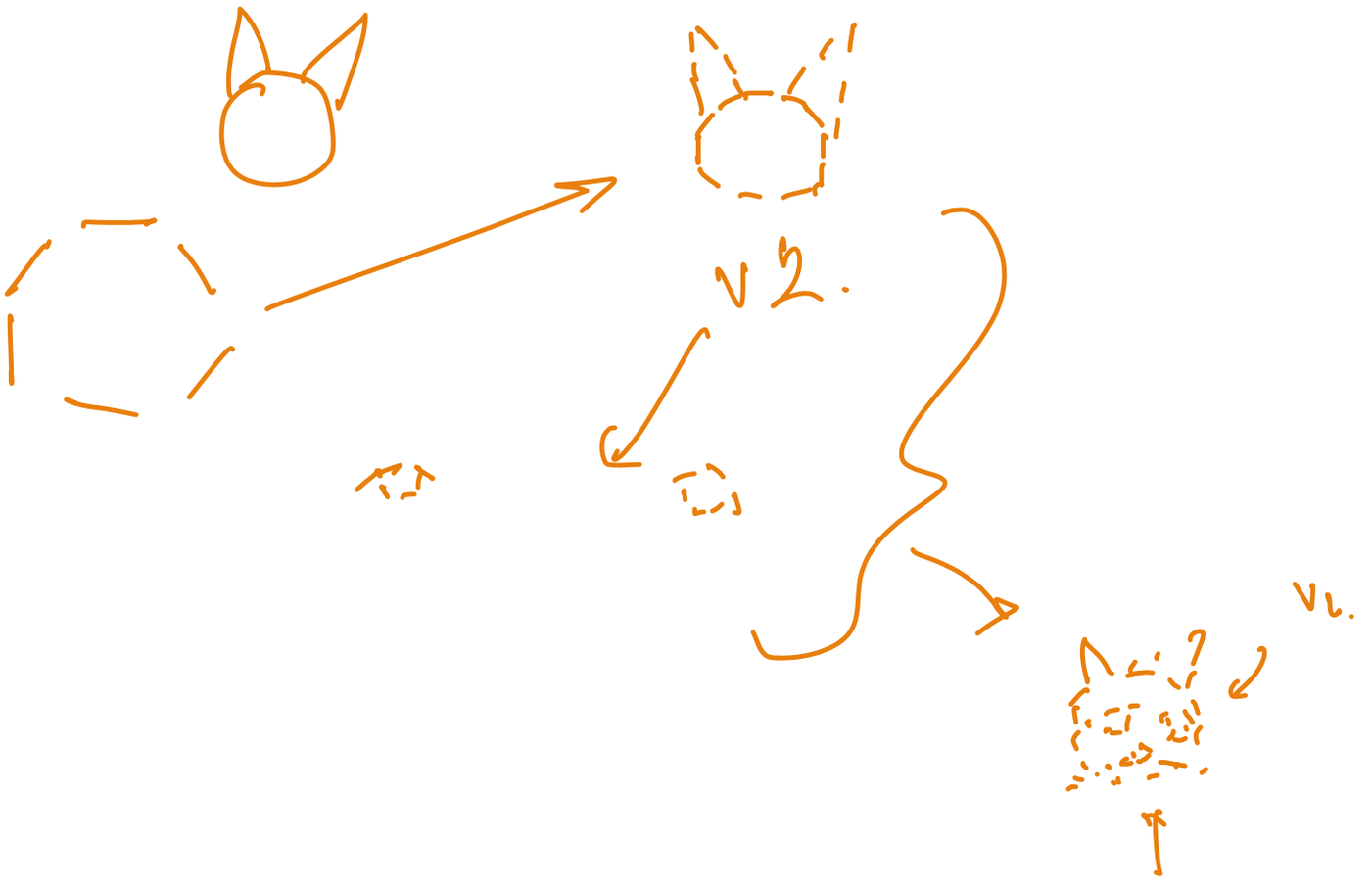
$p = \left(\frac{a-1}{2}\right)$

$m + 2q - (b-1) = m$

$q = \left(\frac{b-1}{2}\right)$

Summary:

1. The CNN architecture is inspired from the Visual Cortex.
2. We use CNNs because flattening results in loss of positional info.
3. The solution is to use the "Convolution" operation.
4. The 3 main types are: ($m \times n \rightarrow$ Image, $a \times b \rightarrow$ Kernel)
   - (i) Valid — $m - a + 1 \times n - b + 1$
   - (ii) Same — $m \times n$
   - (iii) Full — $m + a - 1 \times n + b - 1$.

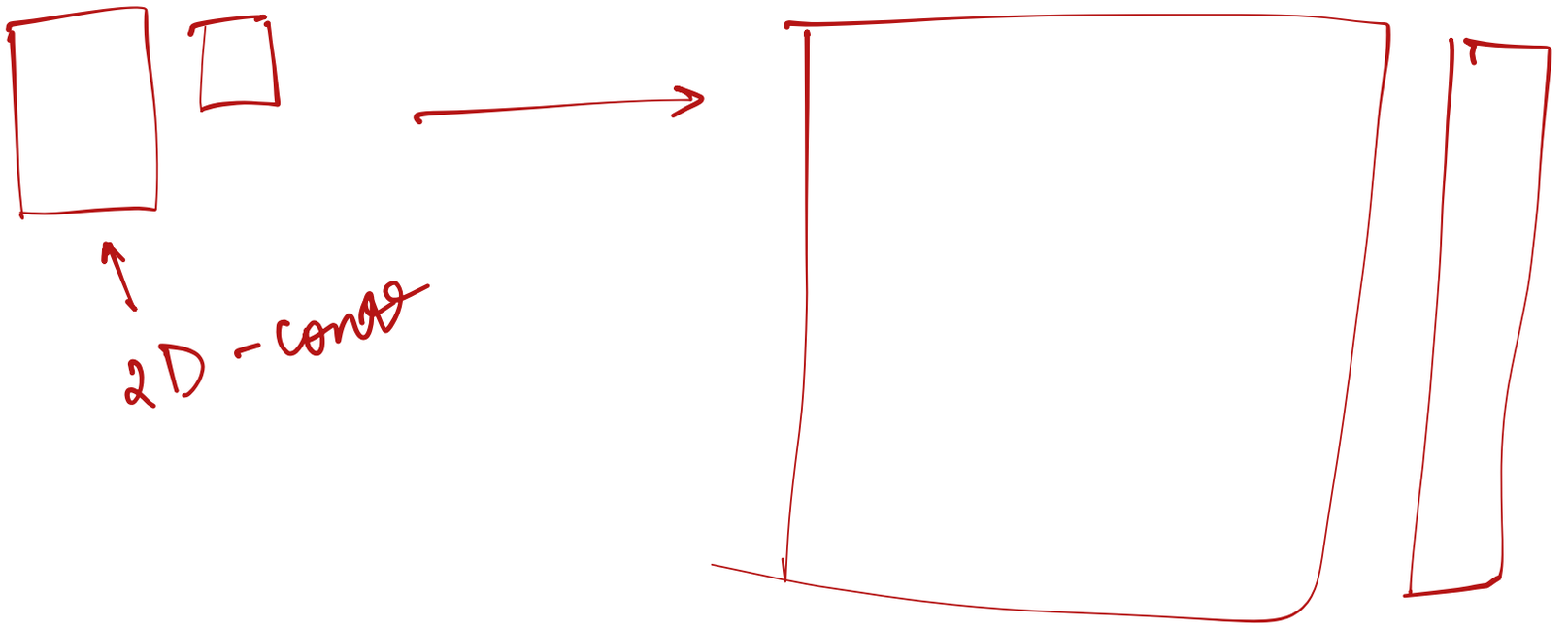

v2.
v1.

Topics :

(a) Vision Transformer.

(b) (Cross encoders) (NLP)

(c) (Bi Encoders)



$$\begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 \end{bmatrix}$$

Toeplitz matrix.

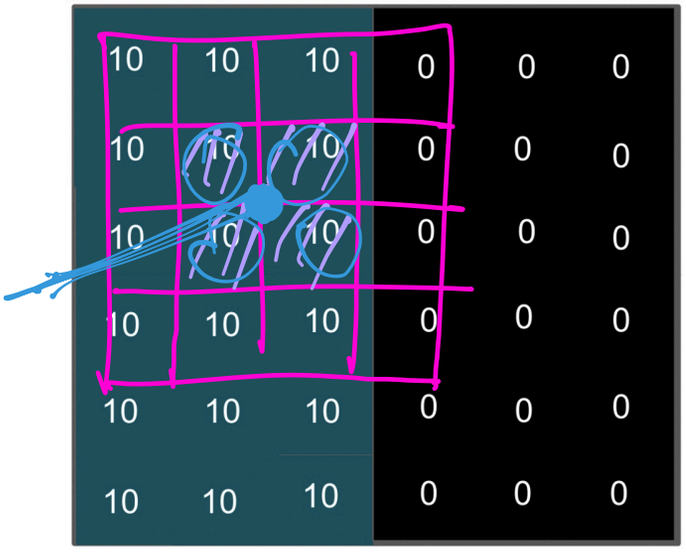
$$W^T b \equiv x_1 * k_1$$

```
import numpy as np
from scipy.signal import convolve2d
```

```
a = np.random.randn(640, 480)
b = np.random.randn(4, 4)
c = convolve2d(a, b)

print(c.shape)
```

```
(643, 483)
```


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