Code The Pixels

Image Processing Workshop

Electronics and Robotics Club, IIT Bombay

PLANNED TIMELINE

Images

OpenCV

CNN

Activity

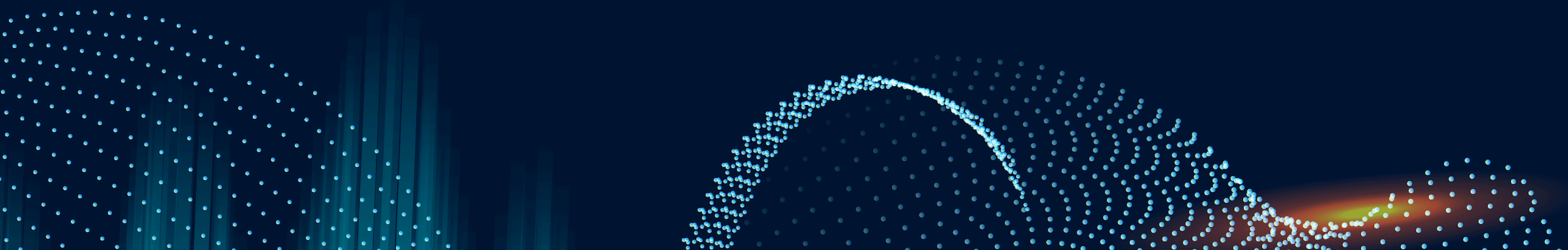
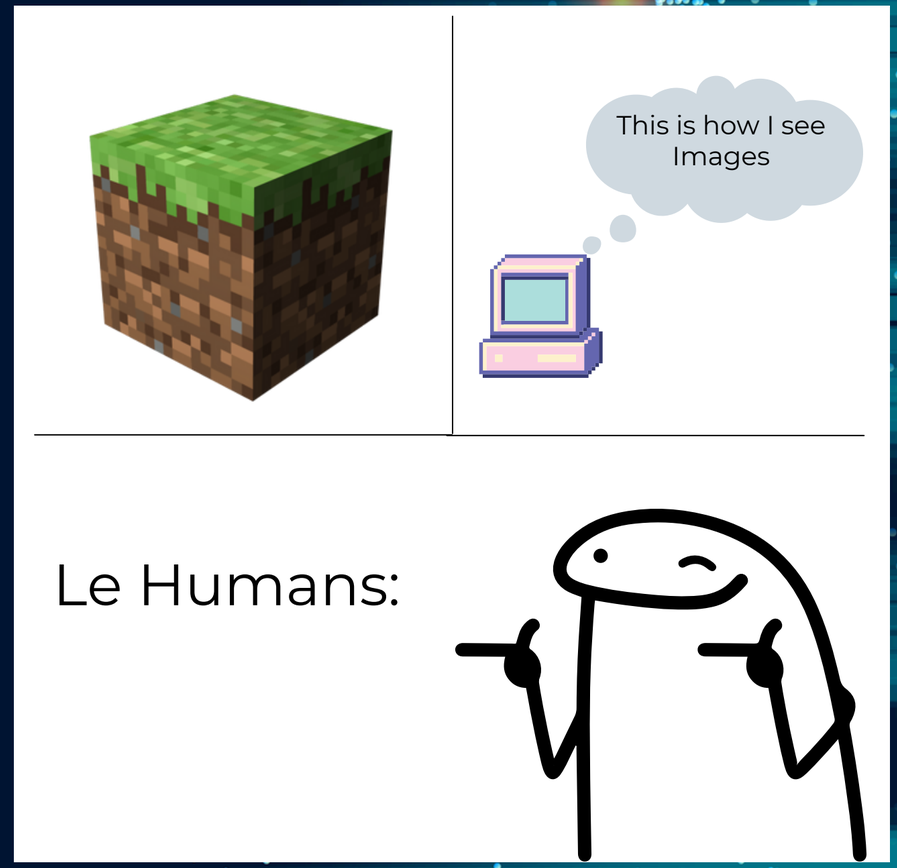
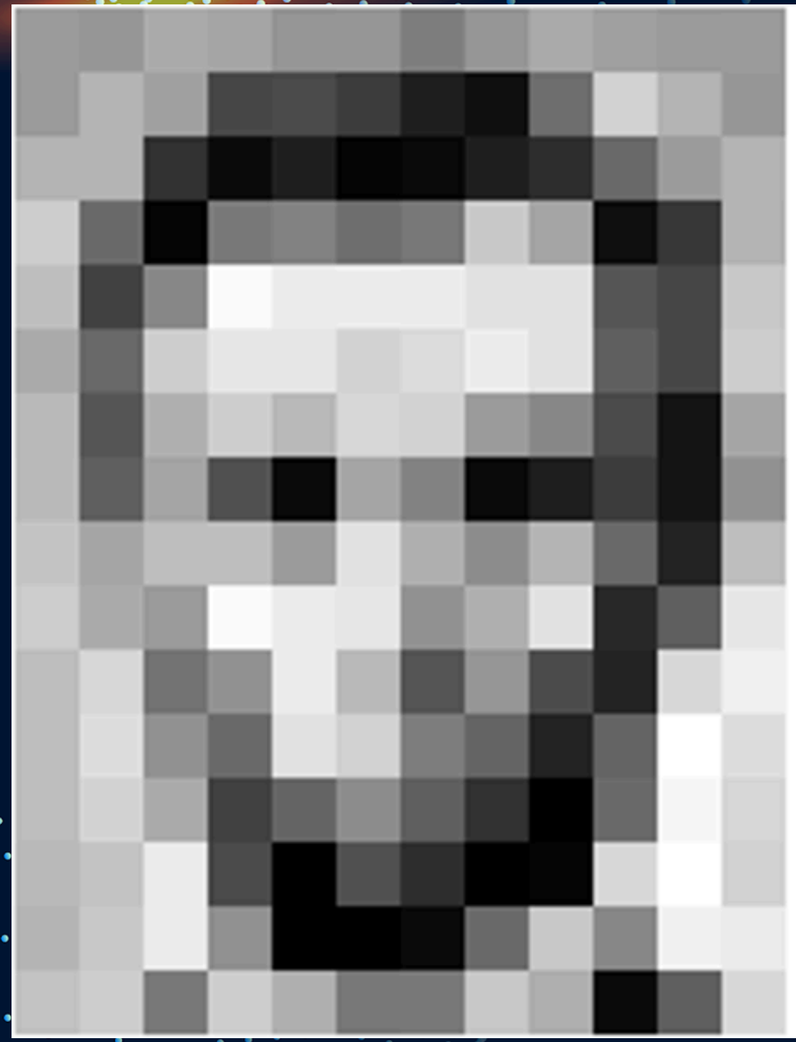


Image as seen by computers

- They are divided into small blocks known as pixels (short for picture elements)
- Each pixel has an intensity value
- More pixels, more variations and so better quality

Now you know why the megapixels of camera matters!

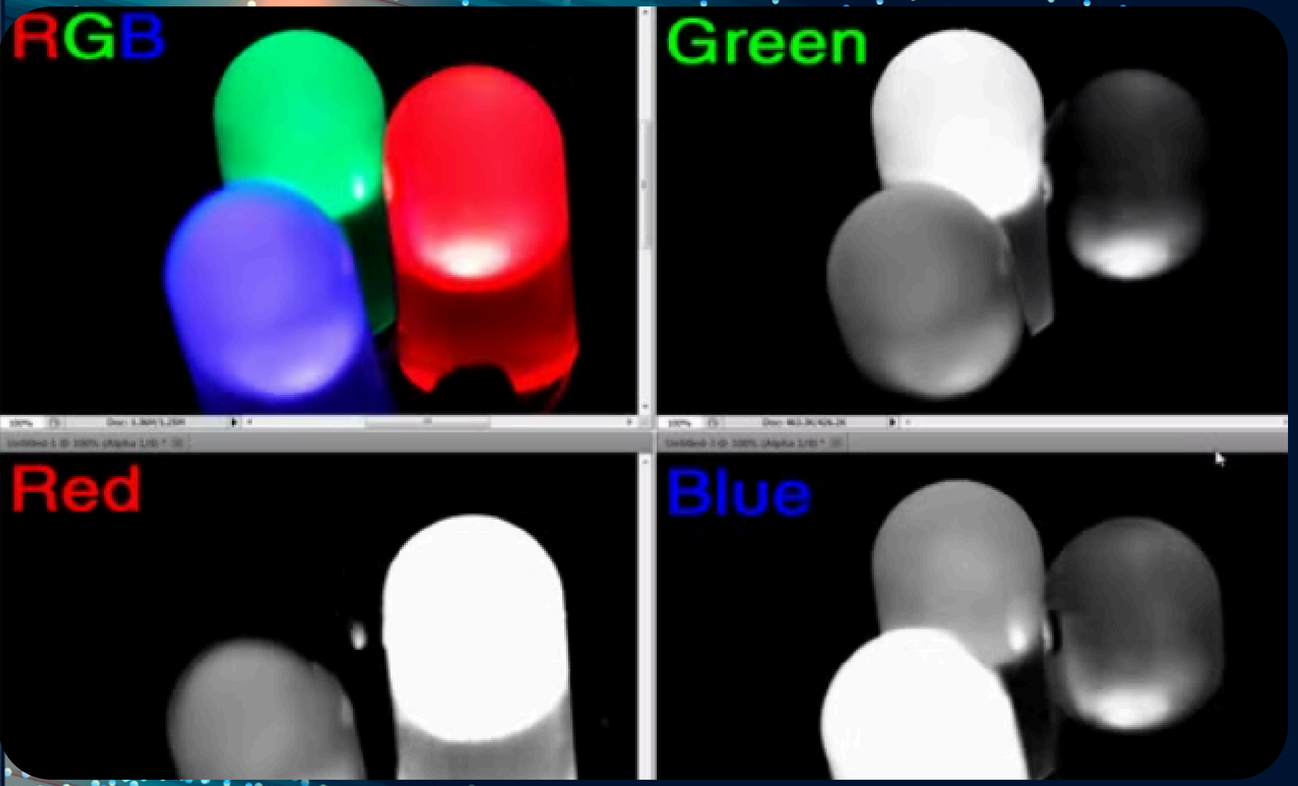




157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

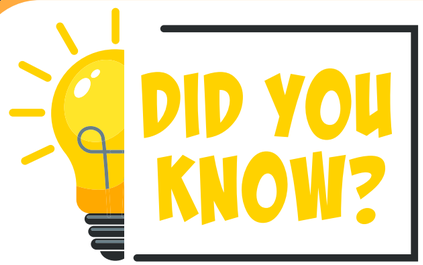
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Colors



Imagine the Color of every Pixel to be a vector, with basis as the colors Red, Green and Blue respectively.

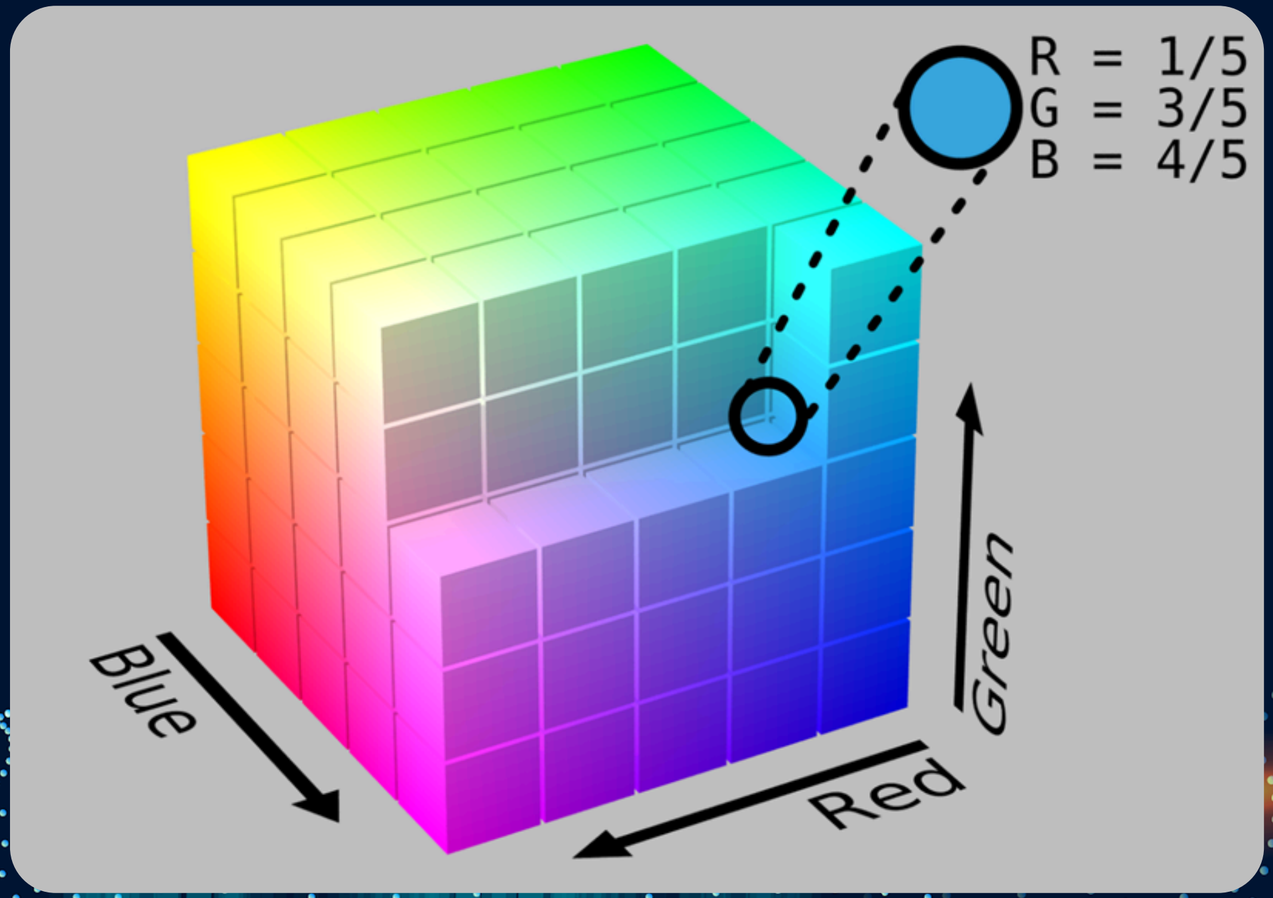
Hence any image can be represented as a collection of 3 greyscale images

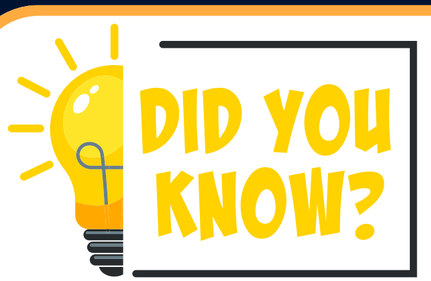


Human eye is more sensitive to green color

Color Formats

RGB

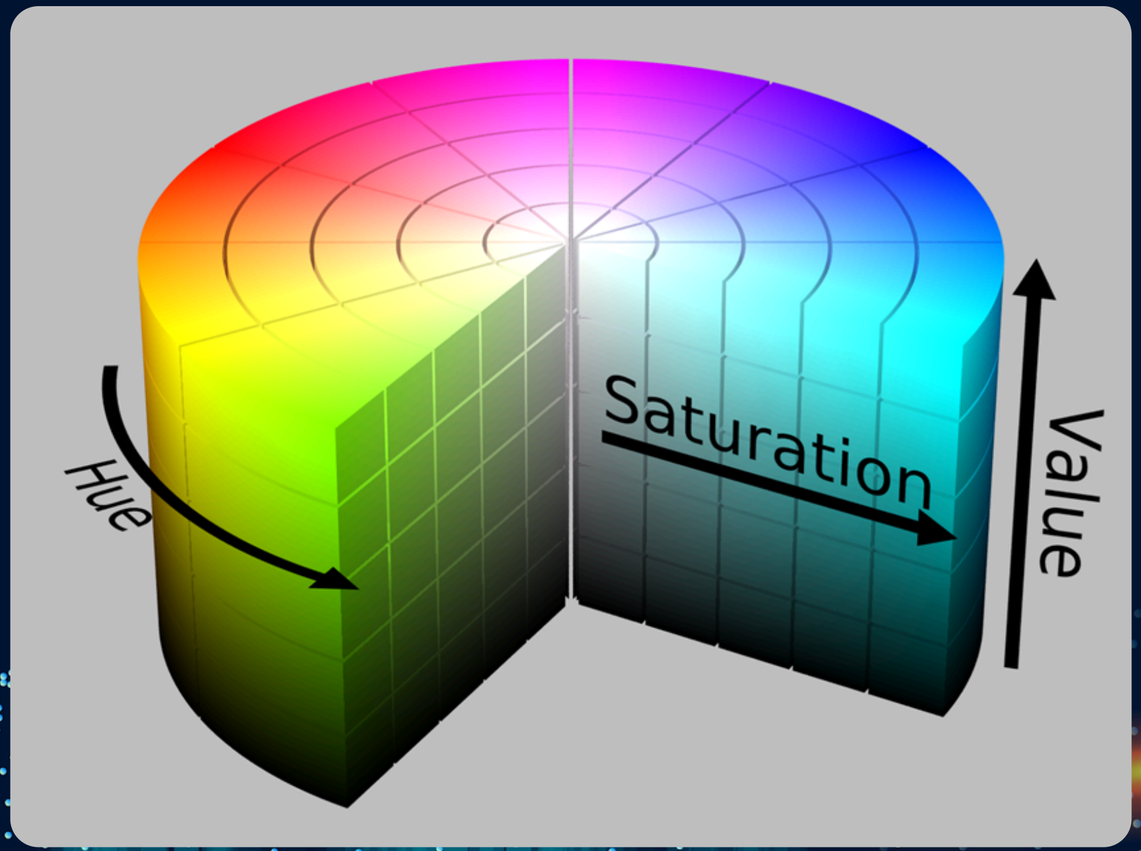




OpenCV uses BGR format instead

Color Formats

HSV



HSV format - Hue, Saturation, Value format

The arcane formulas for transforming between the formats.

The R,G,B values are divided by 255 to change the range from 0..255 to 0..1:

R' = R/255

G' = G/255

B' = B/255

Cmax = max(R', G', B')

Cmin = min(R', G', B')

Δ = Cmax - Cmin

Hue calculation:

$$H = \begin{cases} 60° \times \left(\frac{G'-B'}{\Delta} mod 6\right) & , Cmax = R' \\ 60° \times \left(\frac{B'-R'}{\Delta} + 2\right) & , Cmax = G' \\ 60° \times \left(\frac{R'-G'}{\Delta} + 4\right) & , Cmax = B' \end{cases}$$

Saturation calculation:

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

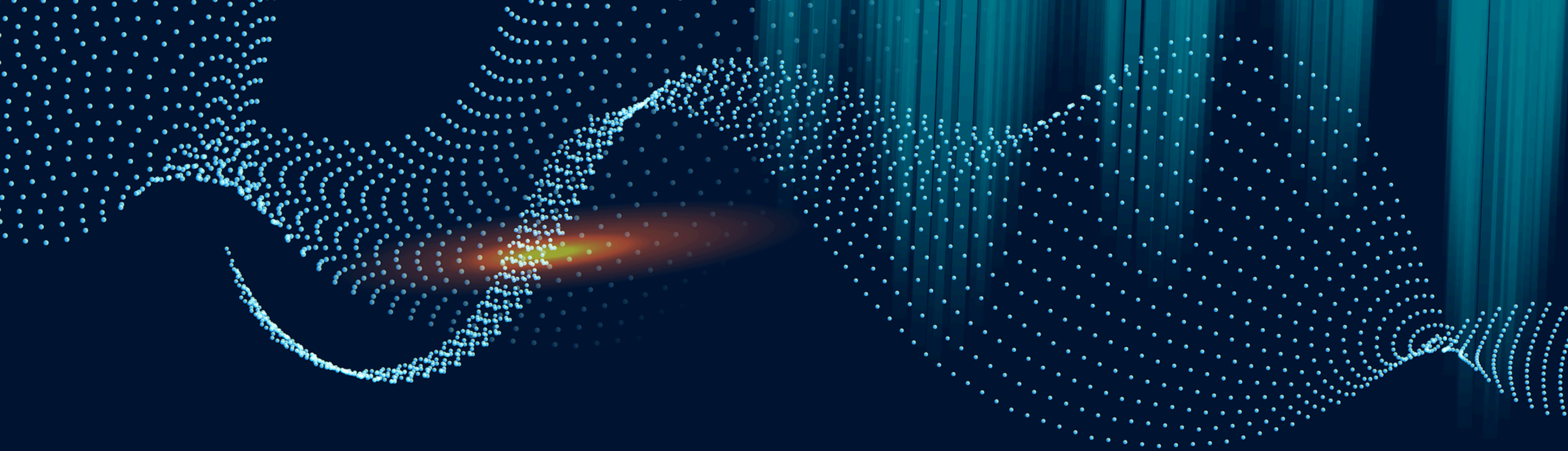
Value calculation: V = Cmax

This is just back end math, so do not worry bout it!

OpenCV

It's hands-on time, folks! Let’s get our mouse a little messy!

bit.ly/erc-imageproc



Thresholding

THRESH_BINARY Python: cv.THRESH_BINARY	$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV Python: cv.THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC Python: cv.THRESH_TRUNC	$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_TOZERO Python: cv.THRESH_TOZERO	$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV Python: cv.THRESH_TOZERO_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$

Necessarily converts the Greyscale image to a Binary format

cv2.threshold(src, thresh, maxval, type)

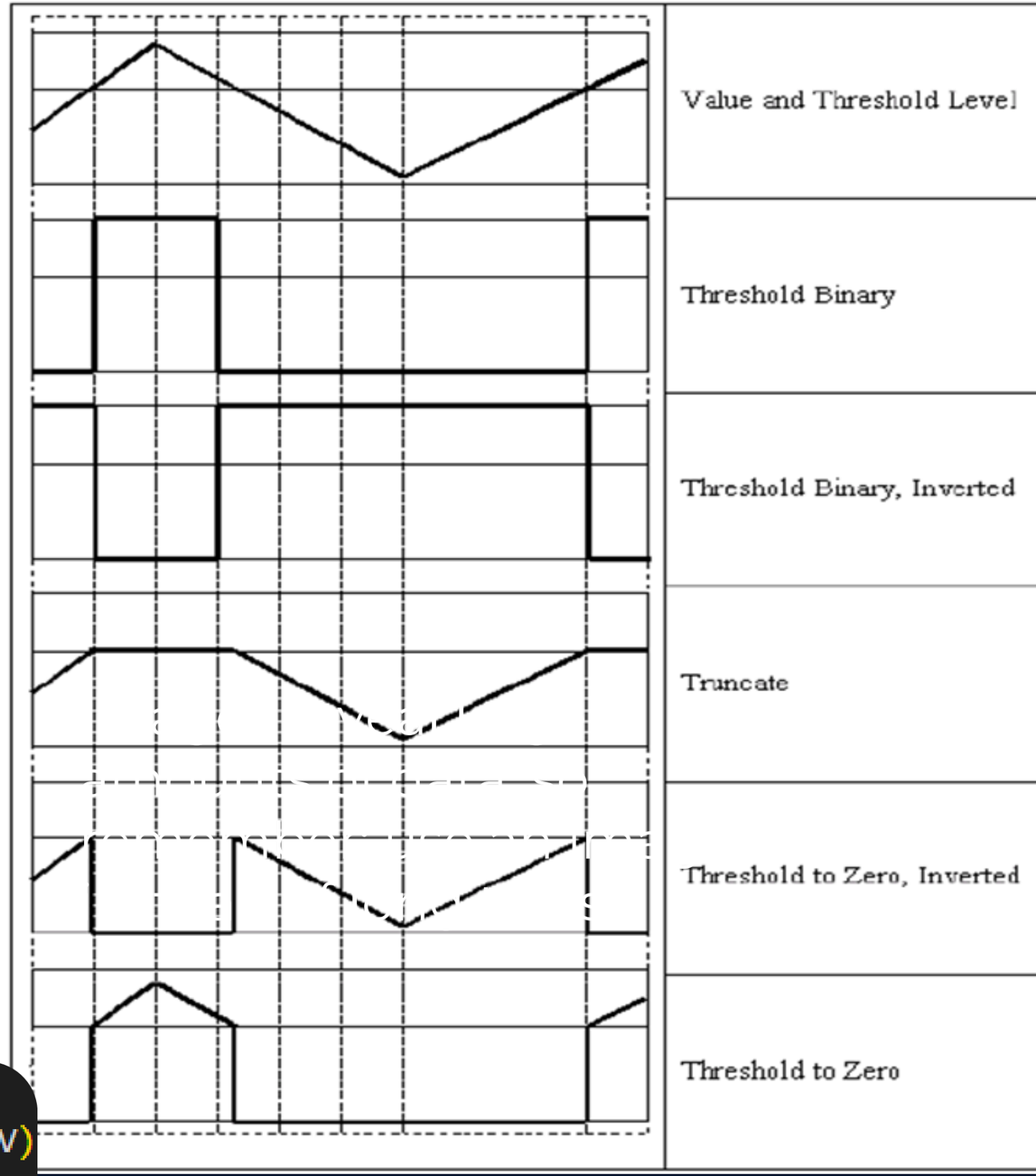
src:
The (grayscale) image to be thresholded

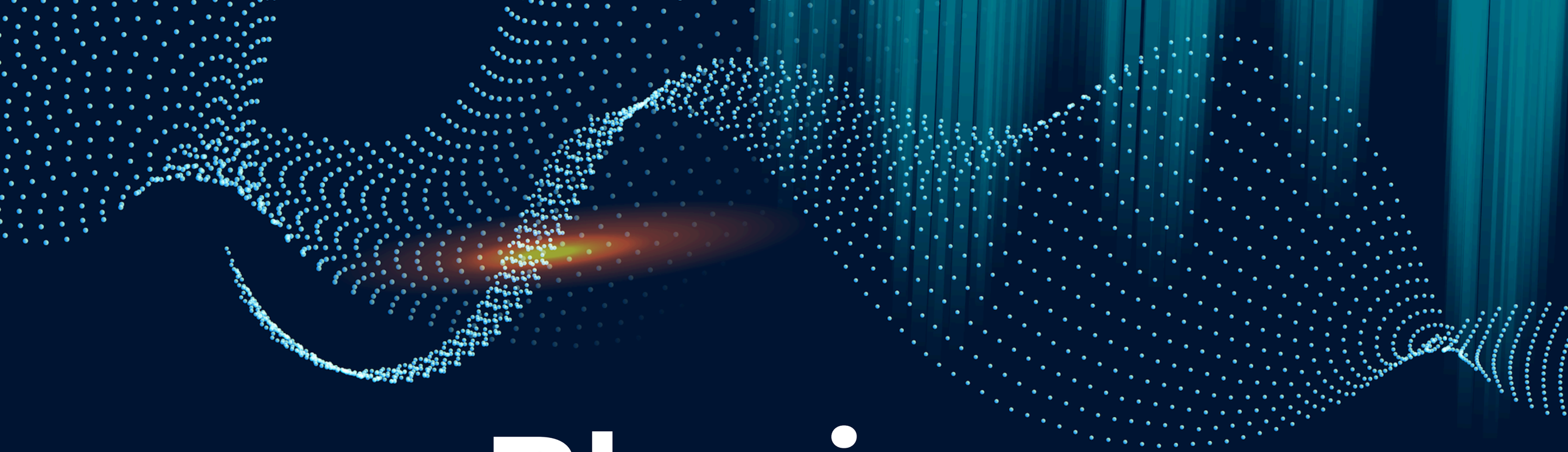
thresh:
The threshold to be used

maxval:
The maximum value to be used in THRESH_BINARY

type:
Type of thresholding to be used

```
ret, thresh1 = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(gray, 127, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(gray, 127, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(gray, 127, 255, cv2.THRESH_TOZERO_INV)
```

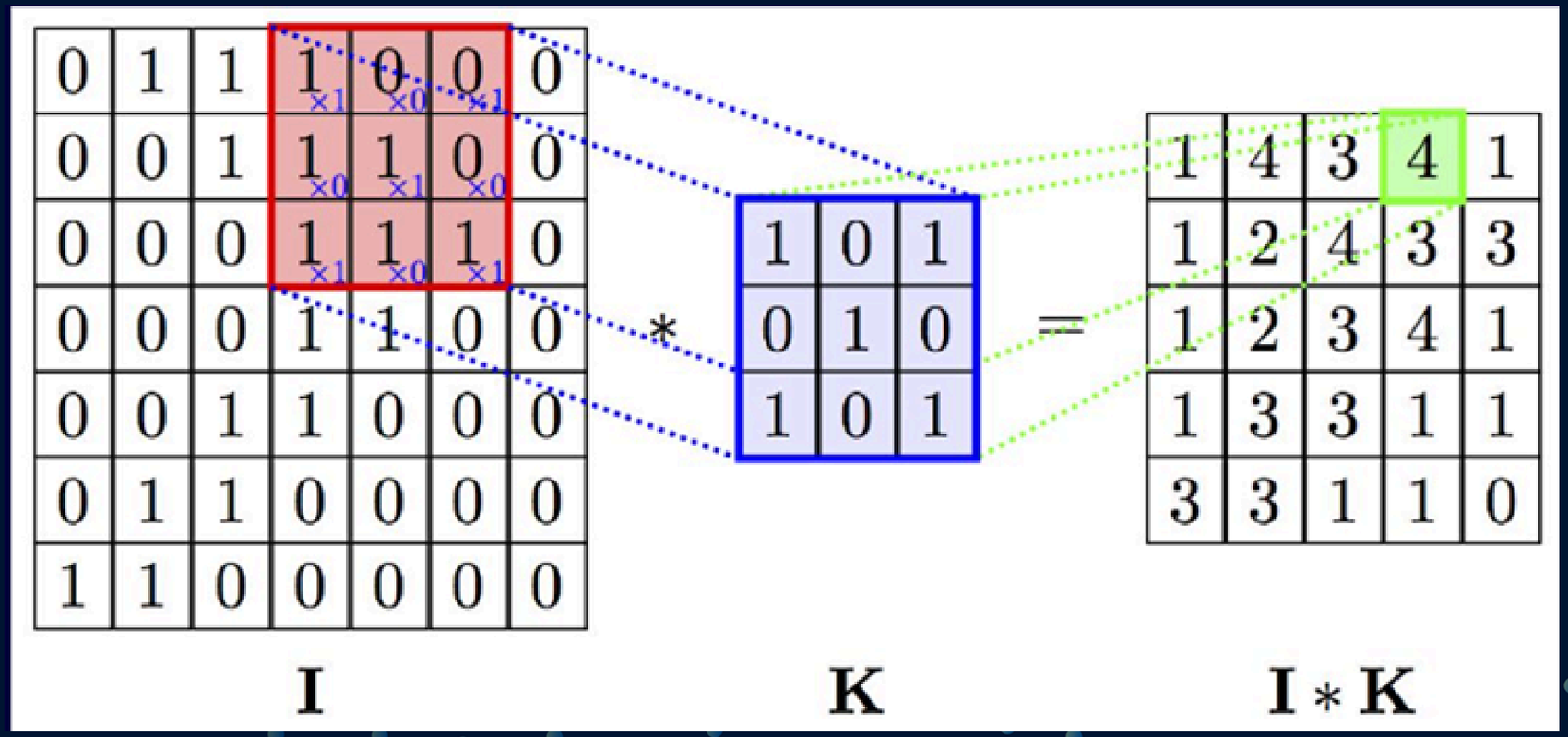




Blurring

Kernel and Convolution

Kernel is nothing more than a Convolution Matrix



feels like you could’nt resist upon thinking about me Huhhh

Q. Let the Operand be of the order m*n and the kernel be of the order p*q determine the order of the convolved matrix.

Ans. `(m-p+1)*(n-q+1)`

Averaging Blur(Box Filter)

- The center of a n×n kernel is replaced by the average of the n^2 pixels in it.
- The kernel size is always an odd integer.

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian Blur(Gaussian Filter)

- The value in the kernel form a 2-D Gaussian distribution

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Median Blur (Median Filter)

- **The center of a n×n kernel is replaced by the median of the n^2 pixels in it.**

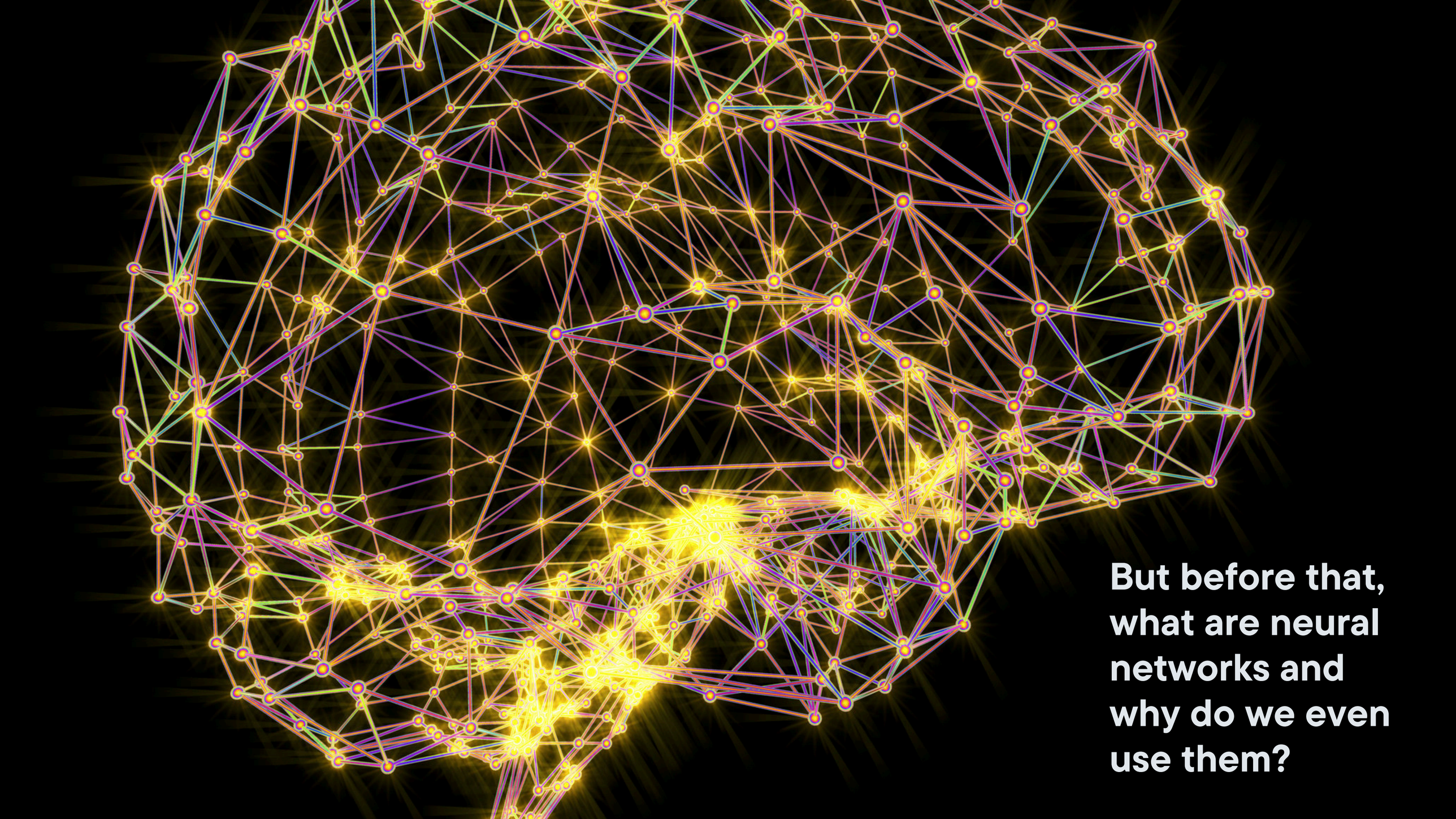
Highly useful in reducing noise from images.

In the above filters, a pixel is set to a calculated value that might not belong to any surrounding pixels. In this method, we set a pixel to value which is present in its neighbourhood. This reduces noise very effectively.

But I am an IITB student, where's the ML-AI

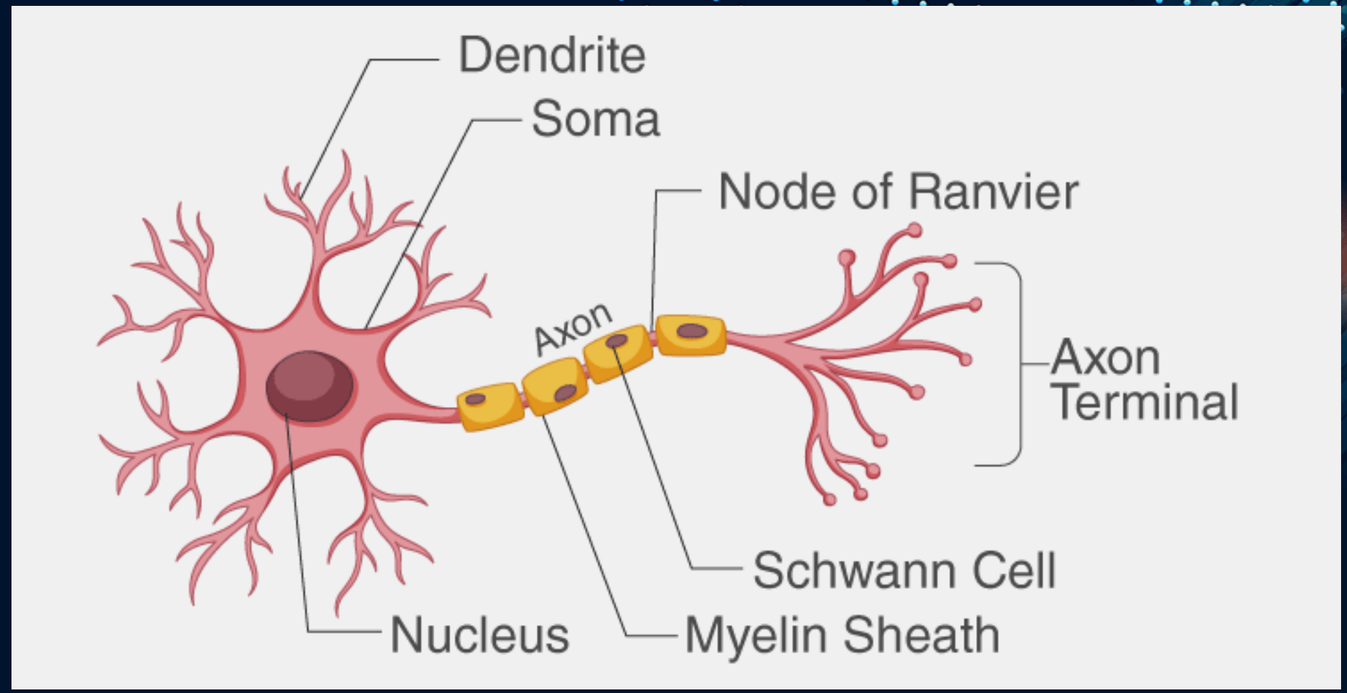
😟😩😭

Okay so let's start with the (machine) learning part of things



But before that, what are neural networks and why do we even use them?

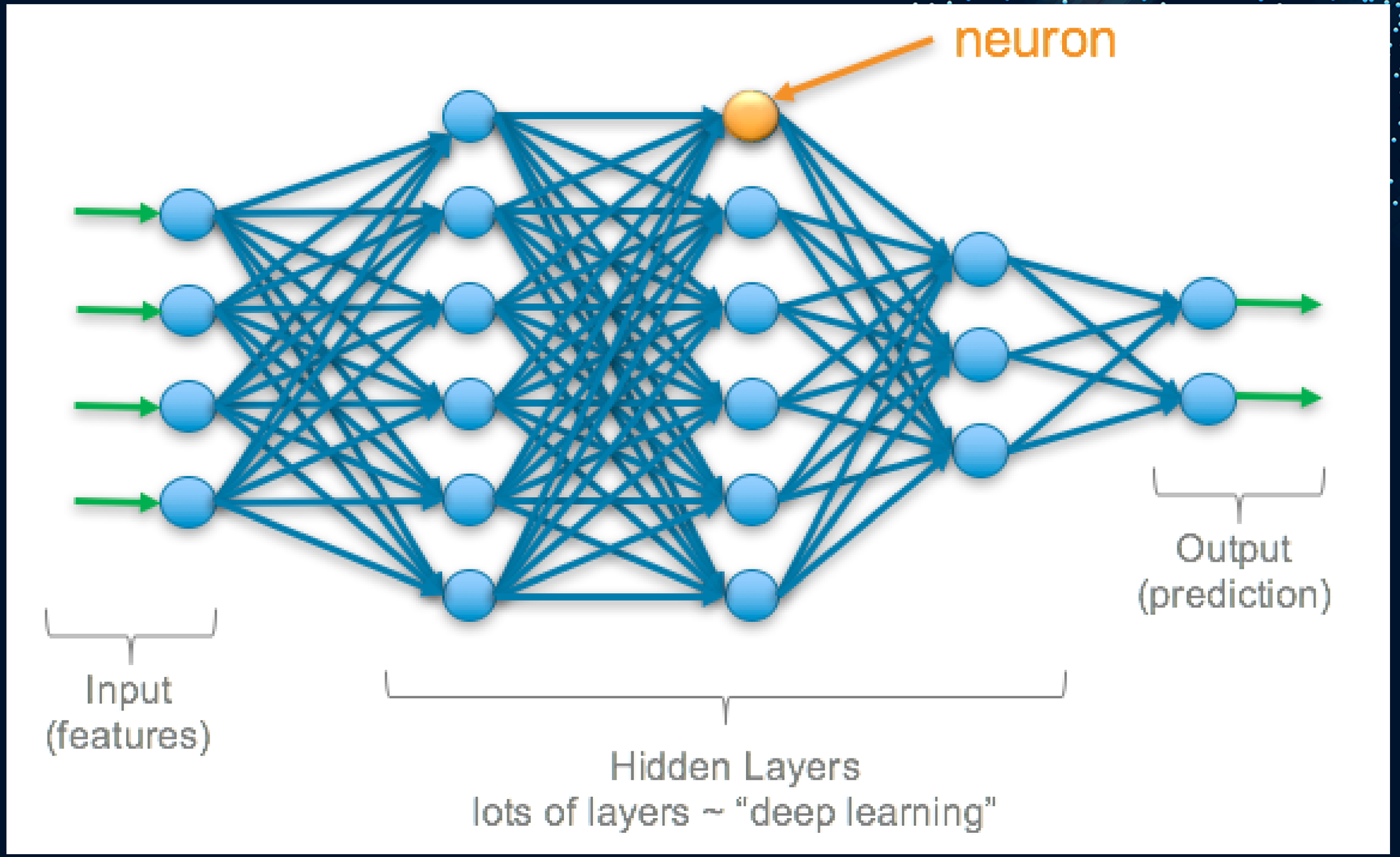
Let us first revise some Class 10 Biology!

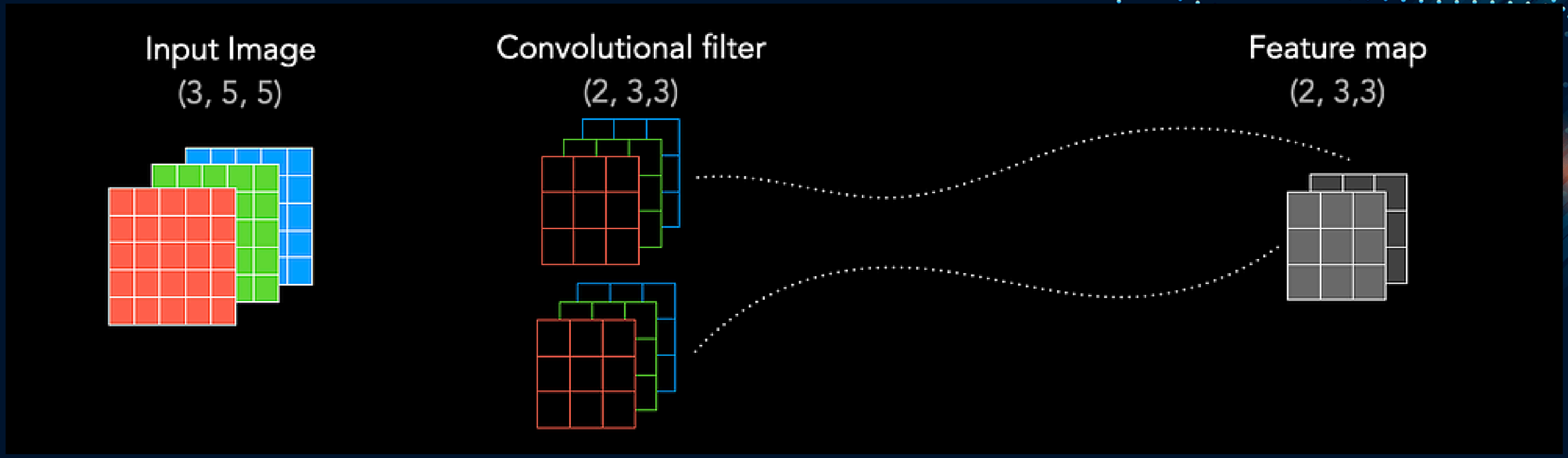


But how does your brain identify anything at all?

You dont't even have a brain bruh

Alright, now let’s understand the need and architecture of a Neural Network!


neuron
Output
(prediction)
Input
(features)
Hidden Layers
lots of layers ~ “deep learning”

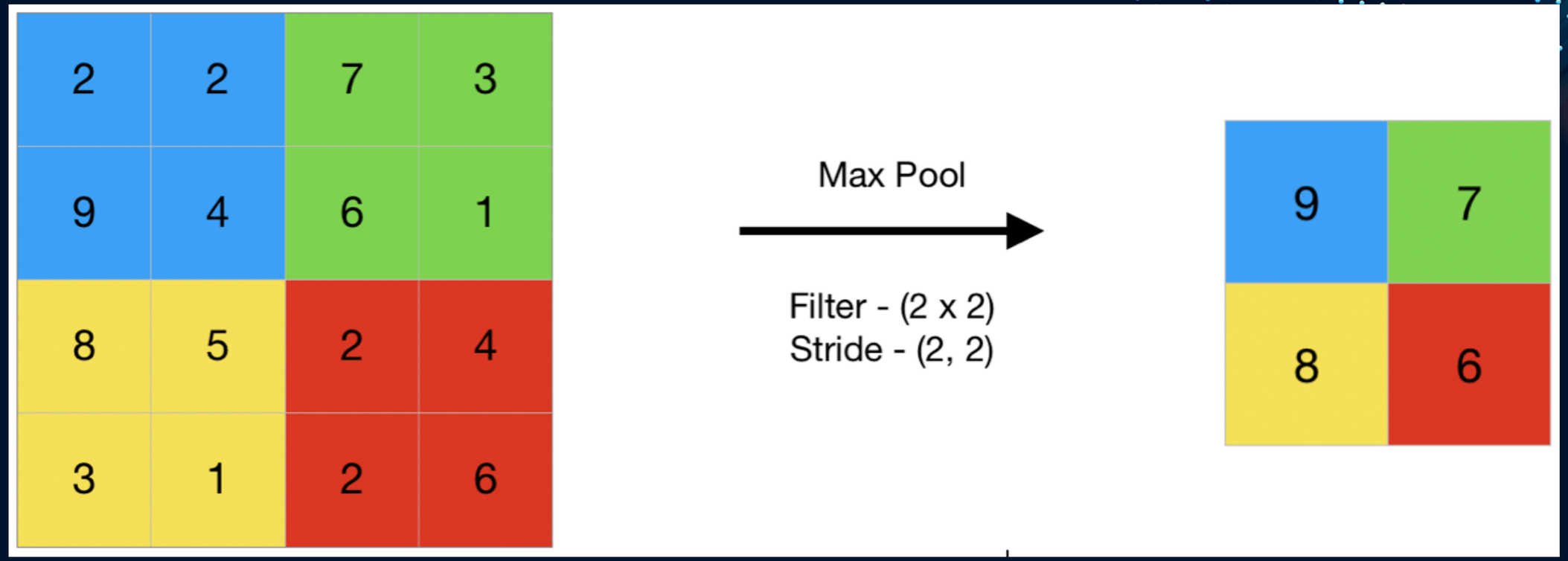

Input Image
(3, 5, 5)
Convolutional filter
(2, 3,3)
Feature map
(2, 3,3)

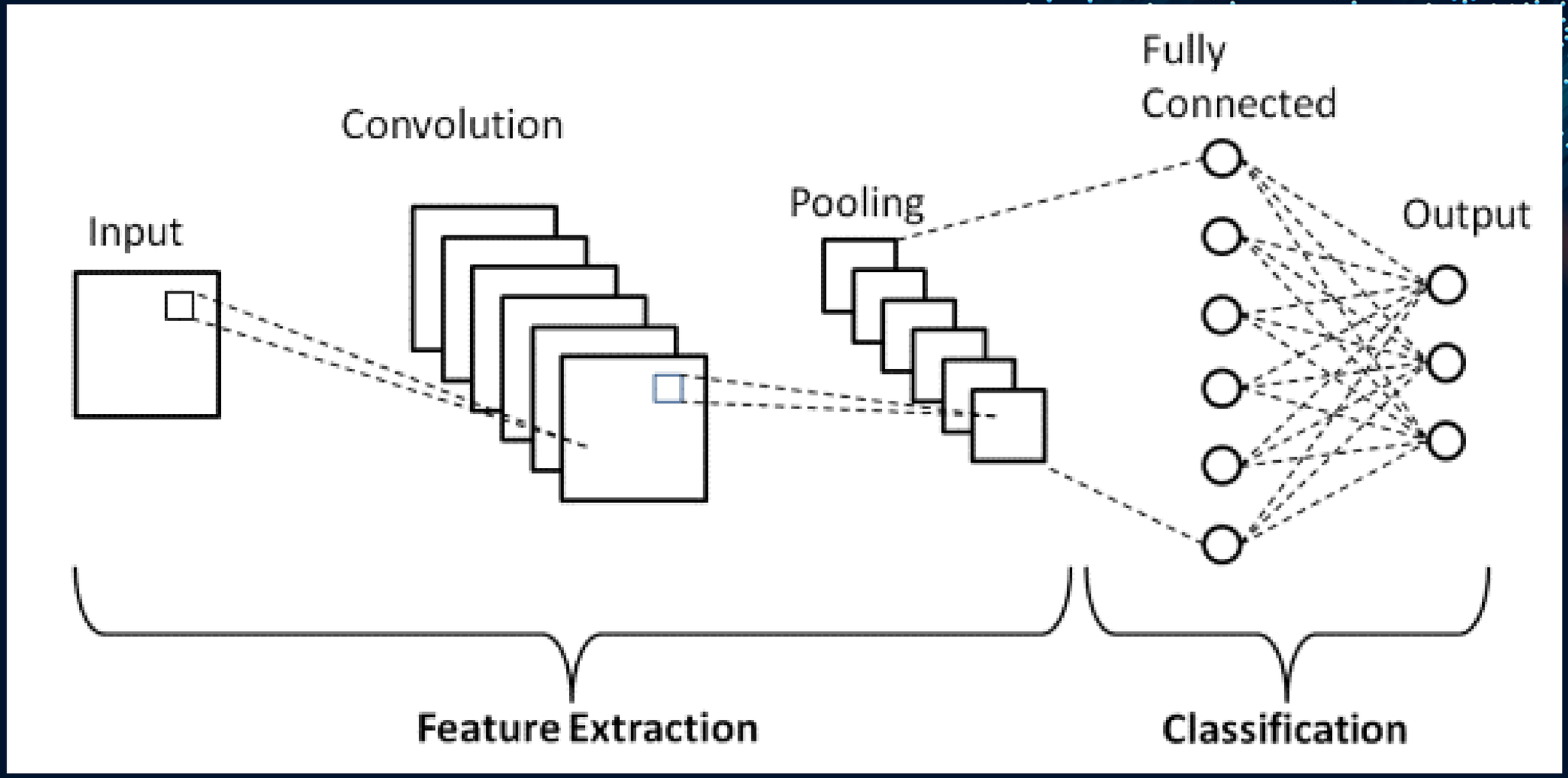
0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

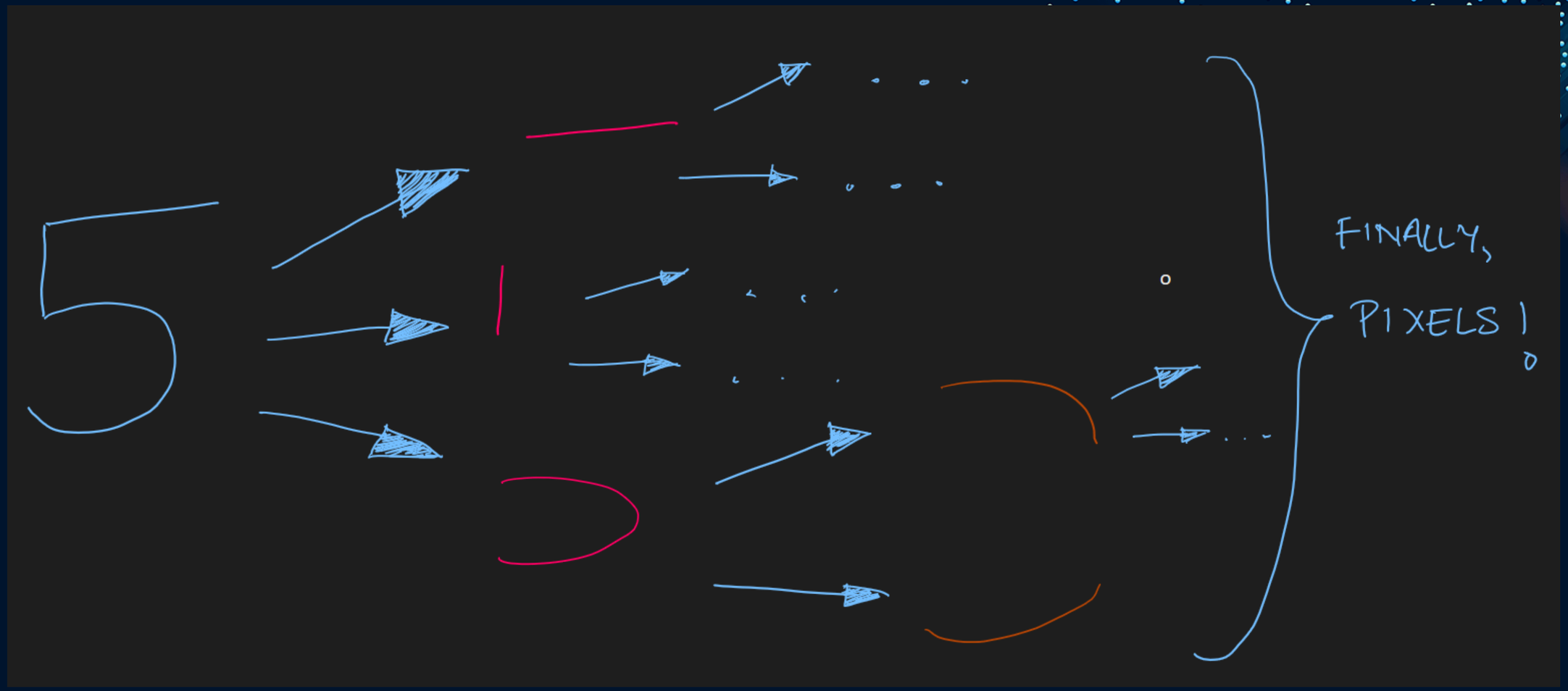
0	-1	0
-1	5	-1
0	-1	0

114	328	-26	470	158
53	266	-61	-30	344


2 2 7 3
9 4 6 1
8 5 2 4
3 1 2 6
Max Pool
Filter - (2 x 2)
Stride - (2, 2)
9 7
8 6


Fully
Connected
Convolution
Pooling
Input
Output
Feature Extraction
Classification

And now, for the most awaited part-**THE PROJECT!**


FINALLY,
PIXELS!