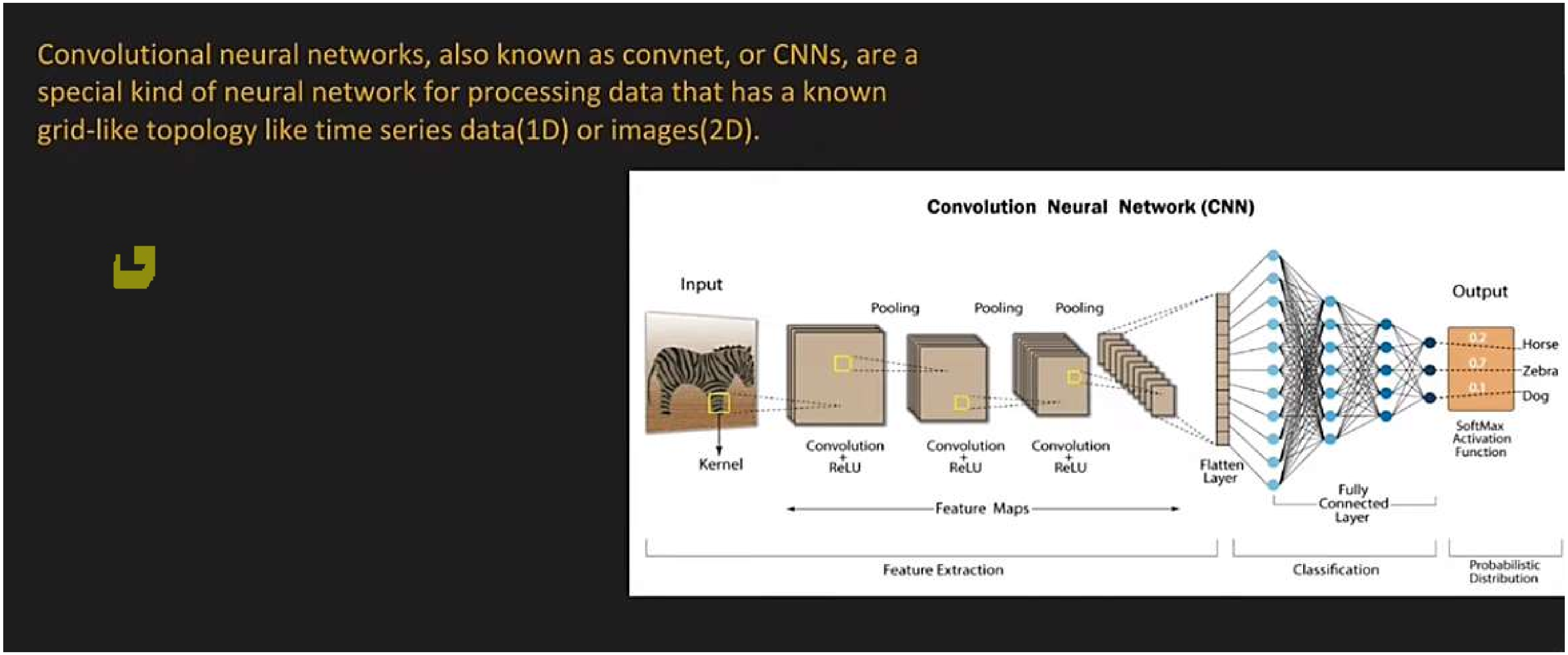
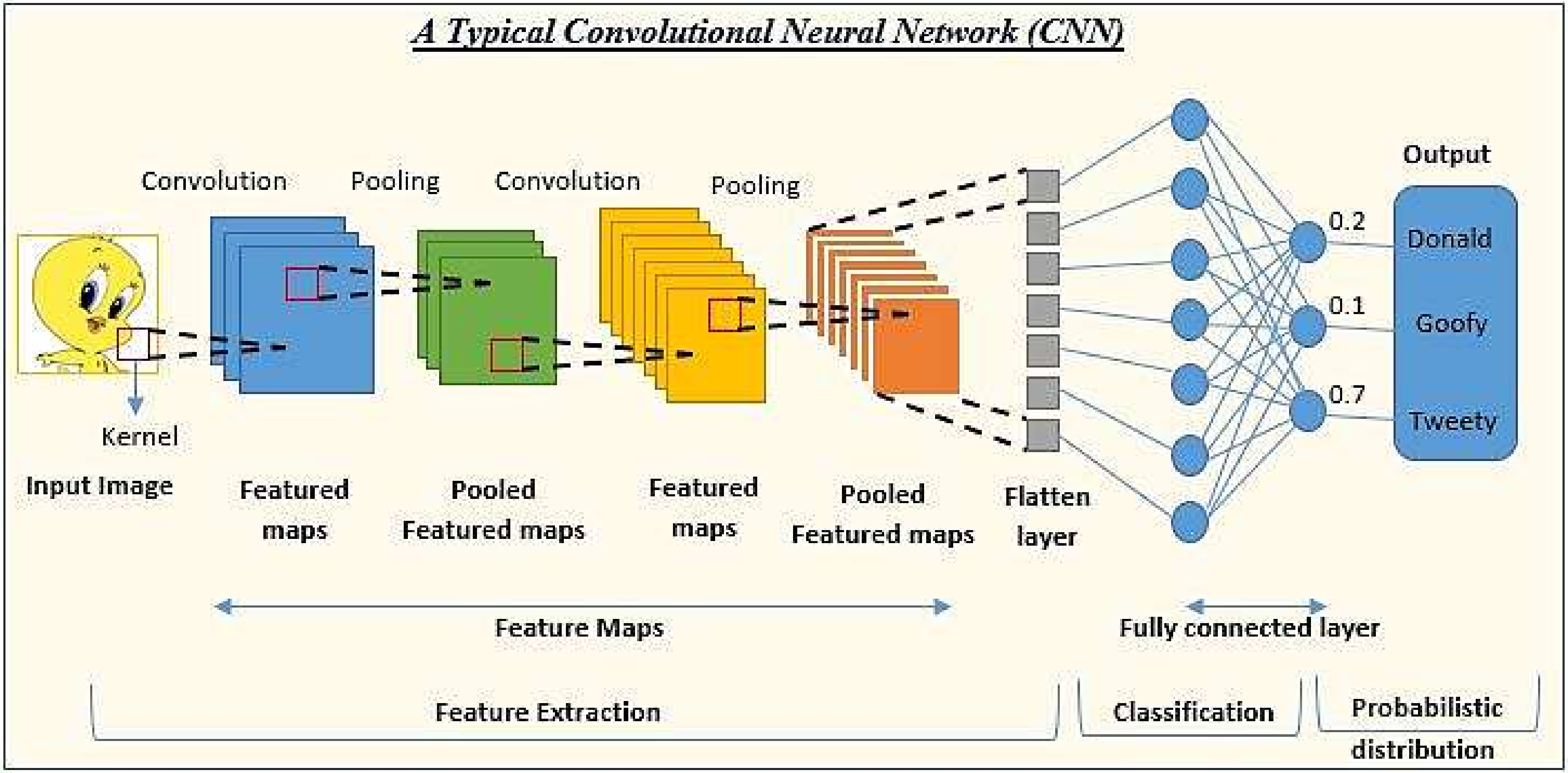
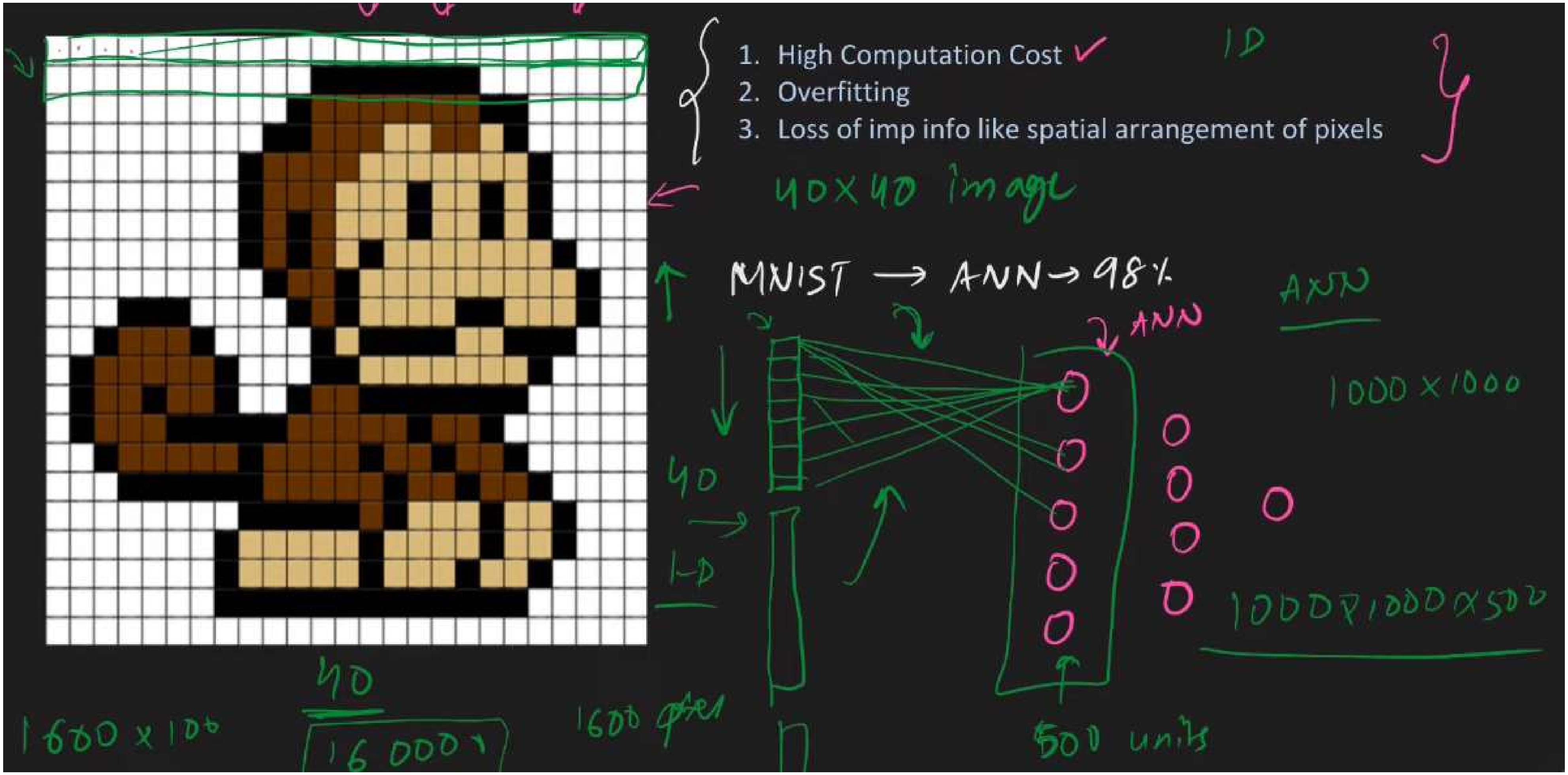
Convolutional Neural Network



Convolutional neural networks, also known as convnet, or CNNs, are a special kind of neural network for processing data that has a known grid-like topology like time series data(1D) or images(2D).

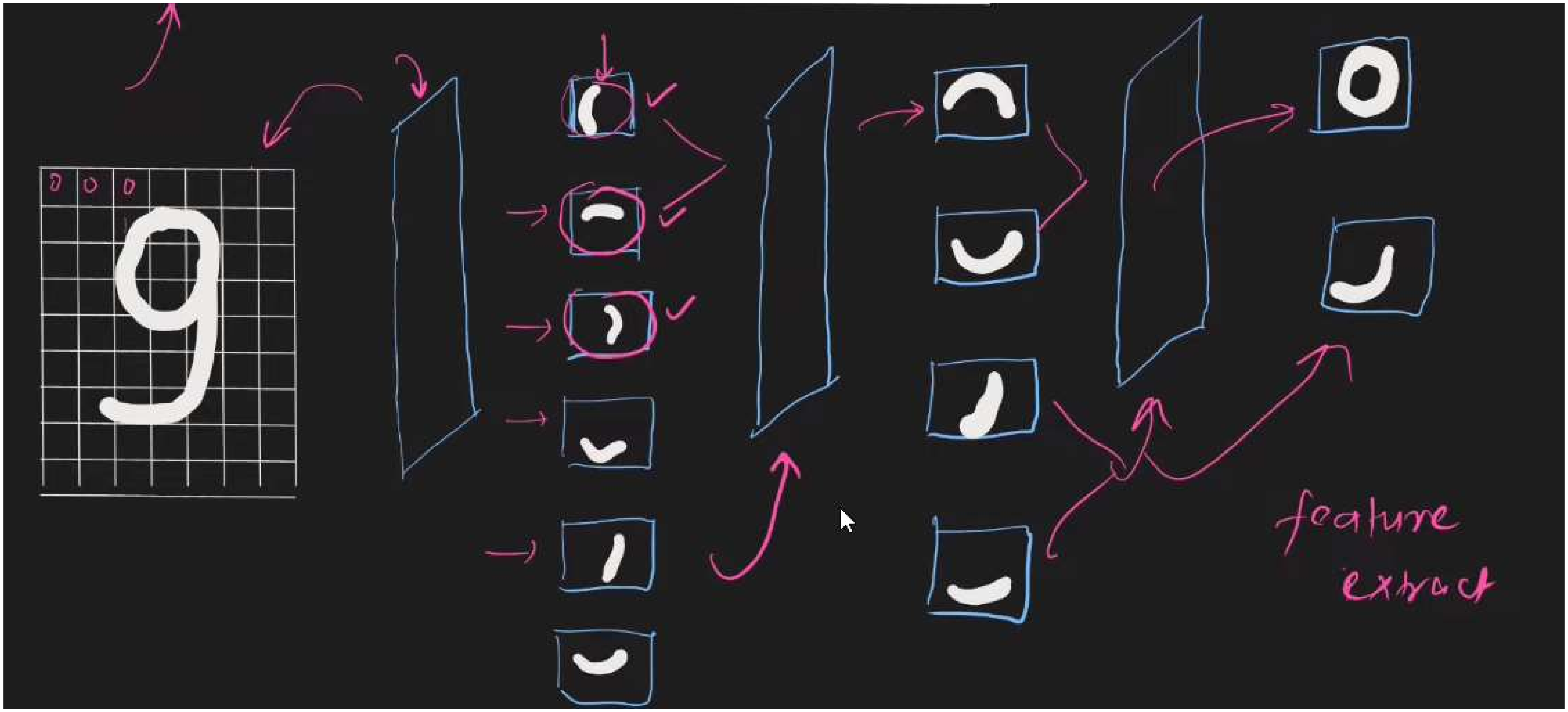

A Typical Convolutional Neural Network (CNN)
Convolution
Pooling
Convolution
Pooling
Output
0.2
0.1
0.7
Donald
Goofy
Tweety
Kernel
Input Image
Featured maps
Pooled Featured maps
Featured maps
Pooled Featured maps
Flatten layer
Feature Maps
Fully connected layer
Feature Extraction
Classification
Probabilistic distribution

Why Not Use ANNs



1. High Computation Cost ✓
2. Overfitting
3. Loss of imp info like spatial arrangement of pixels

Intuition of CNN



Application of CNN

Image Classification

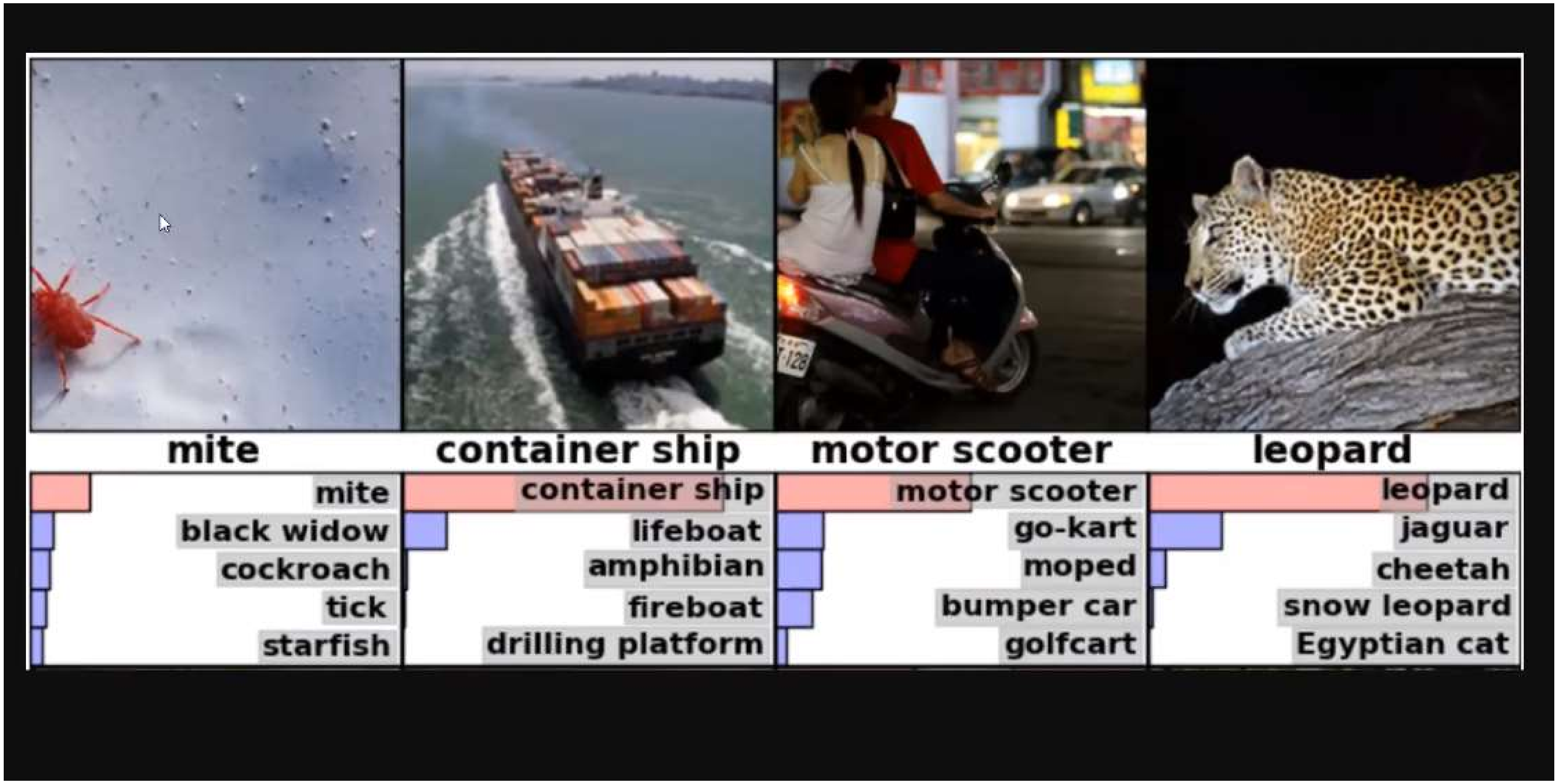
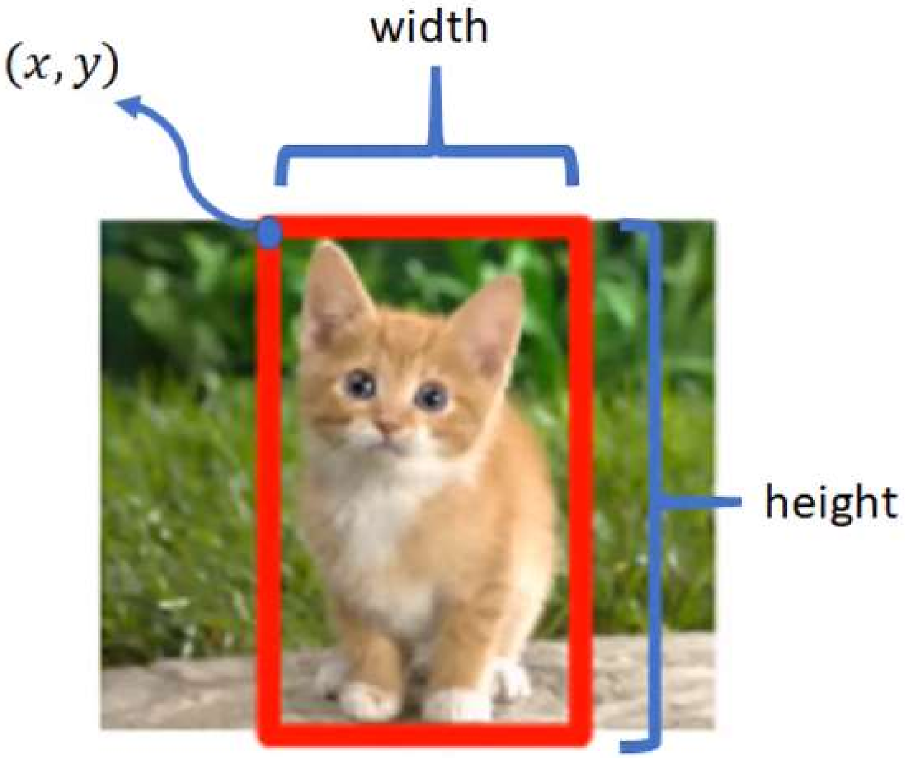
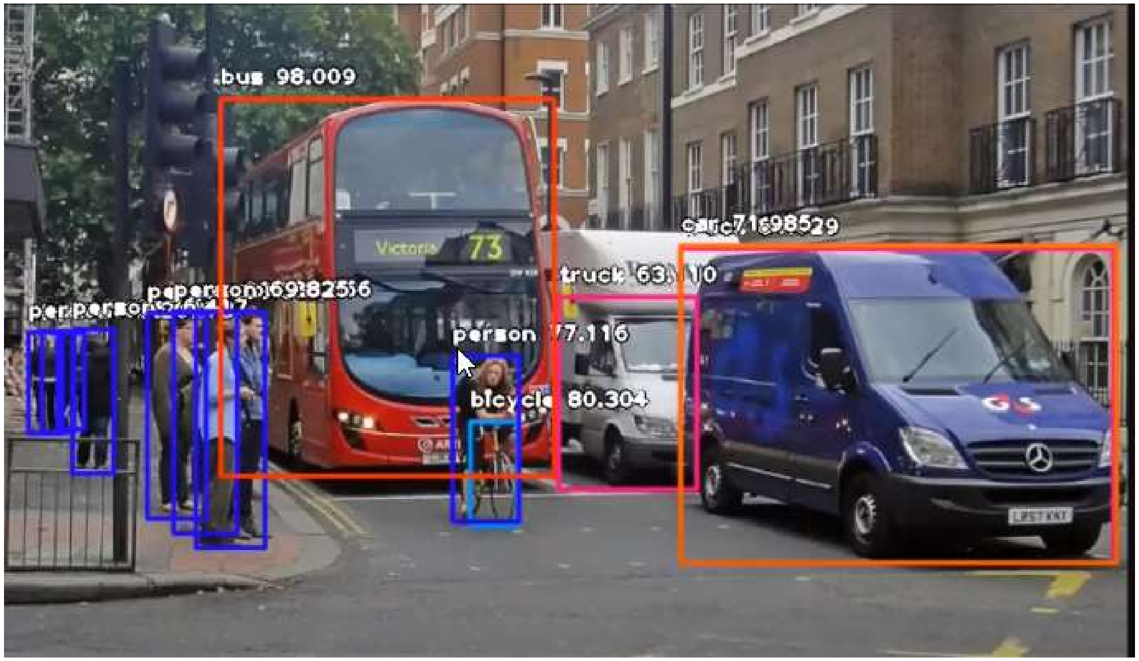


Image Localization



Object Detection



Facial Recognition

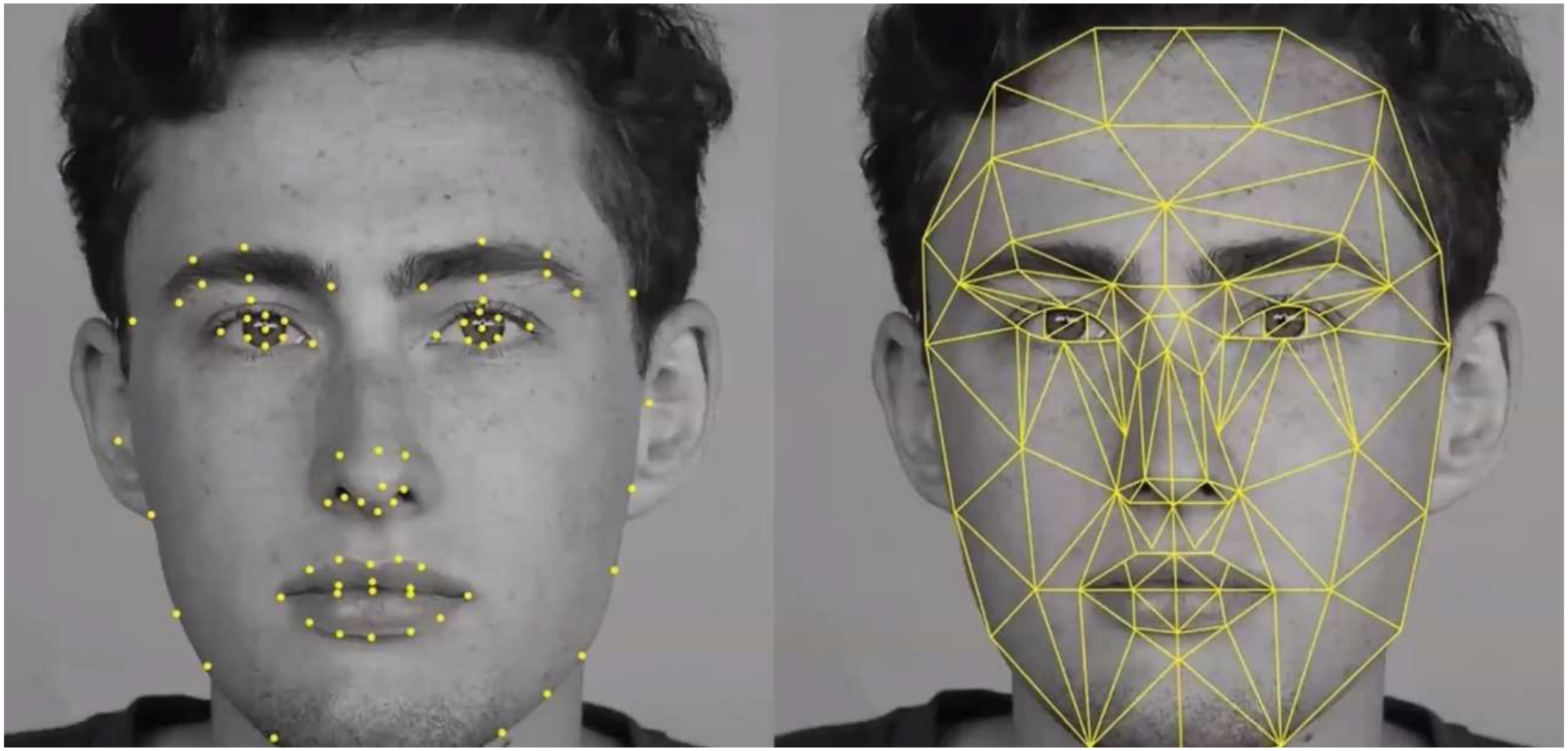
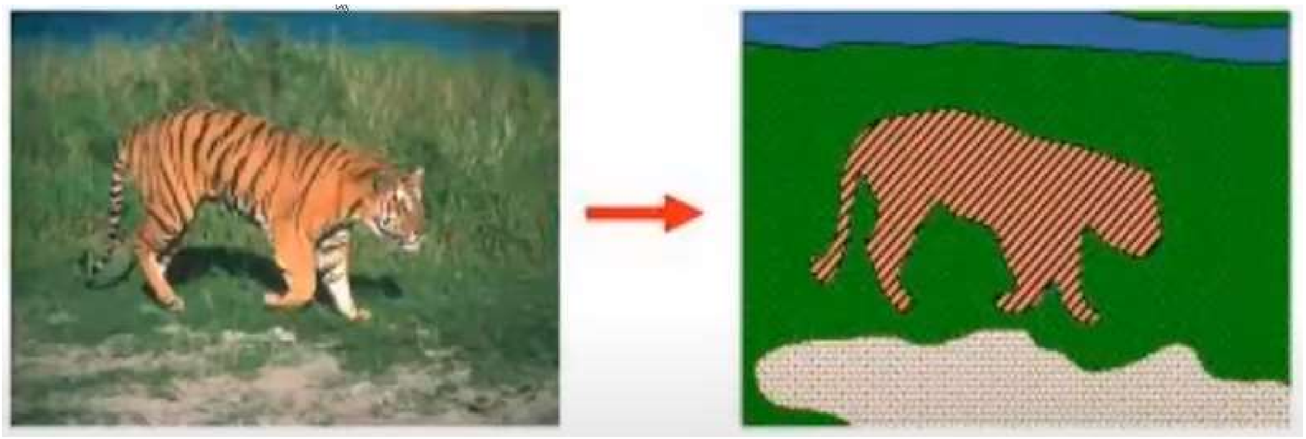
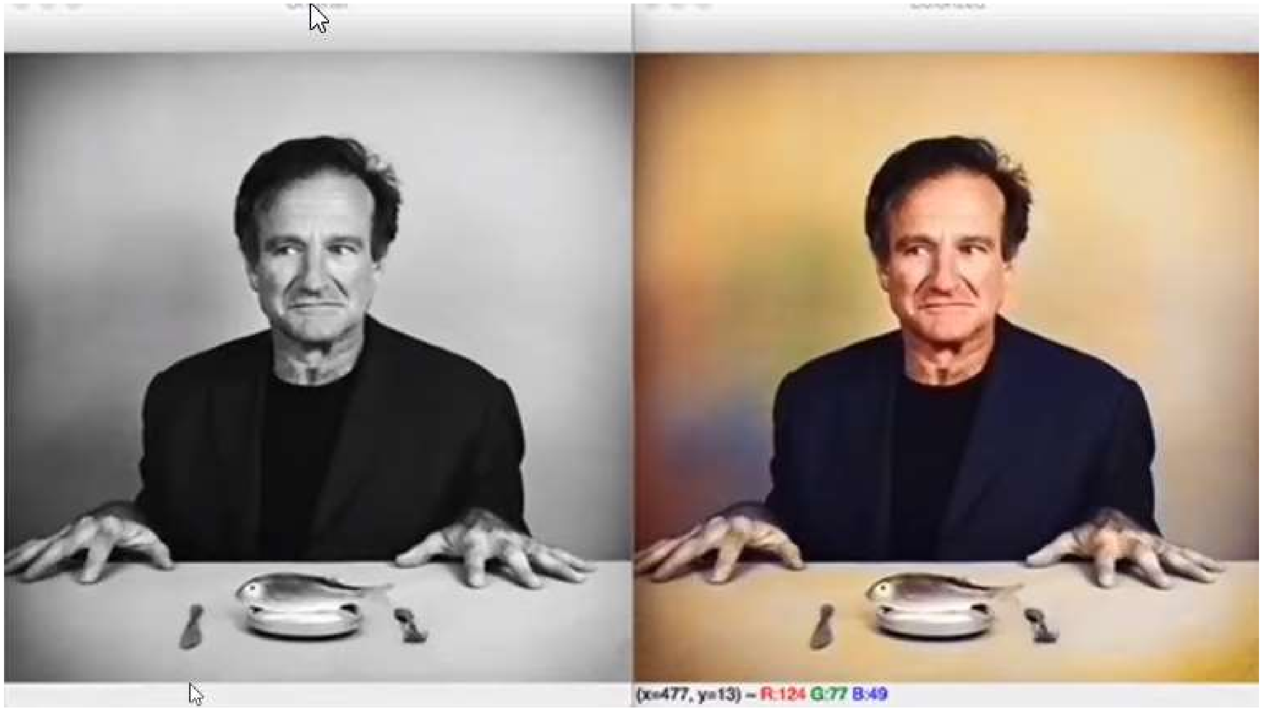


Image Segmentation

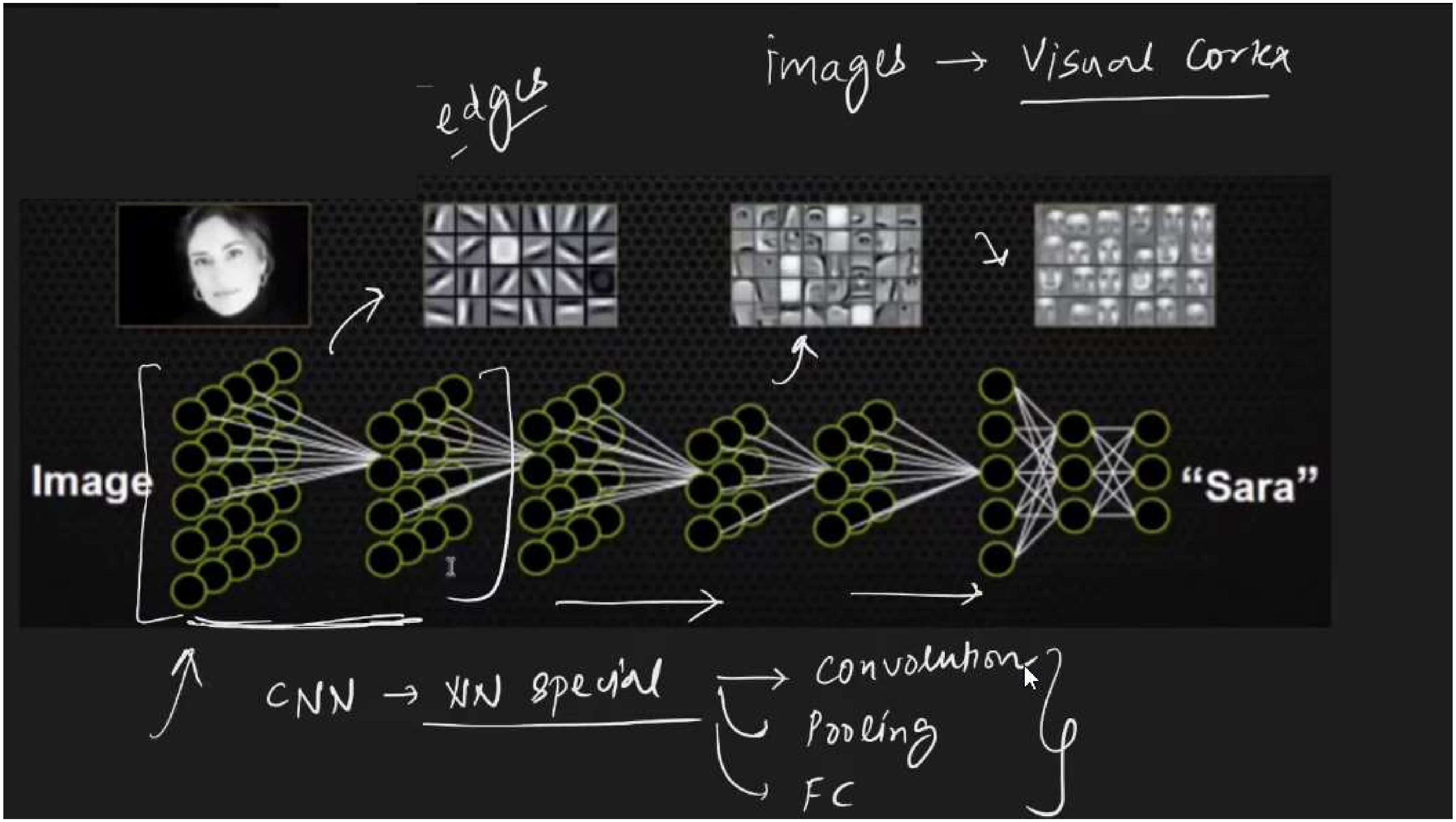




```
(x=477, y=13) ~ R:124 G:77 B:49
```

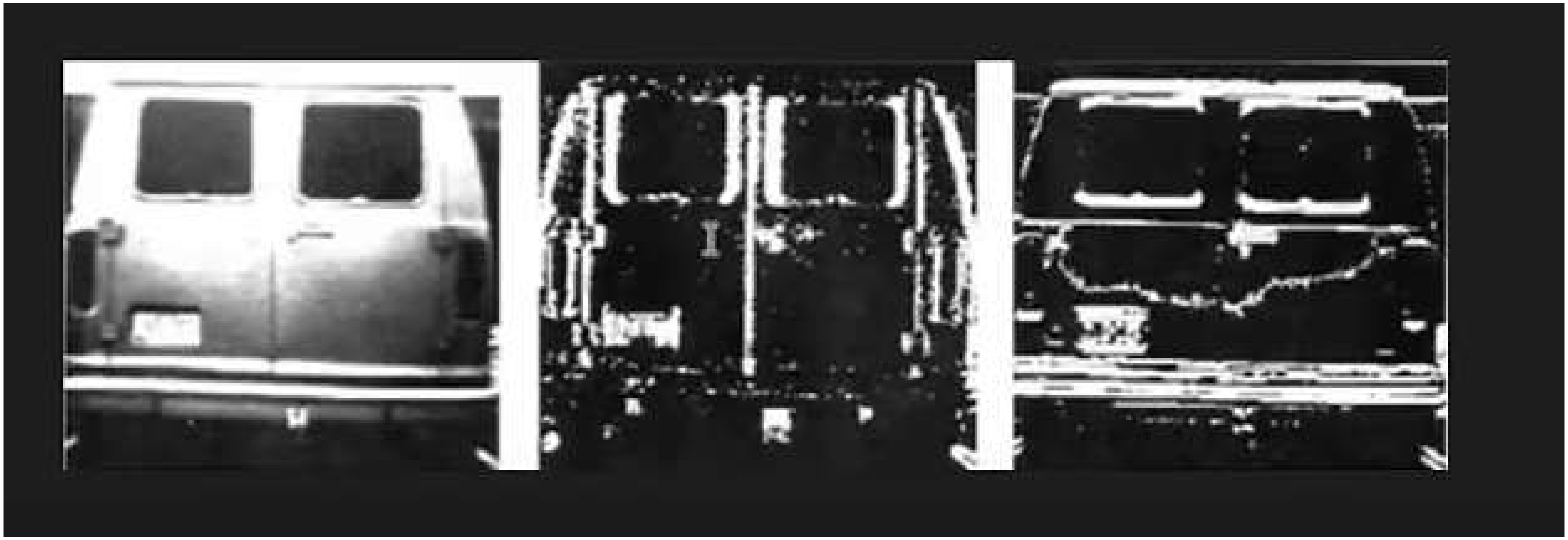
Pose Estimation

images → Visual Cortex



CNN → NN special → convolution
Pooling
FC

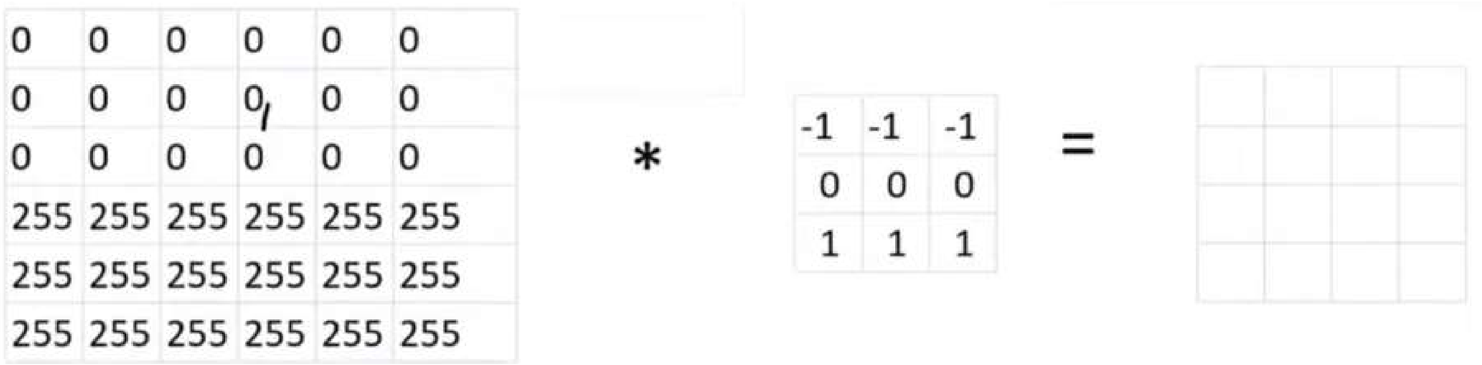
Convolution Operation (Edge Detection)



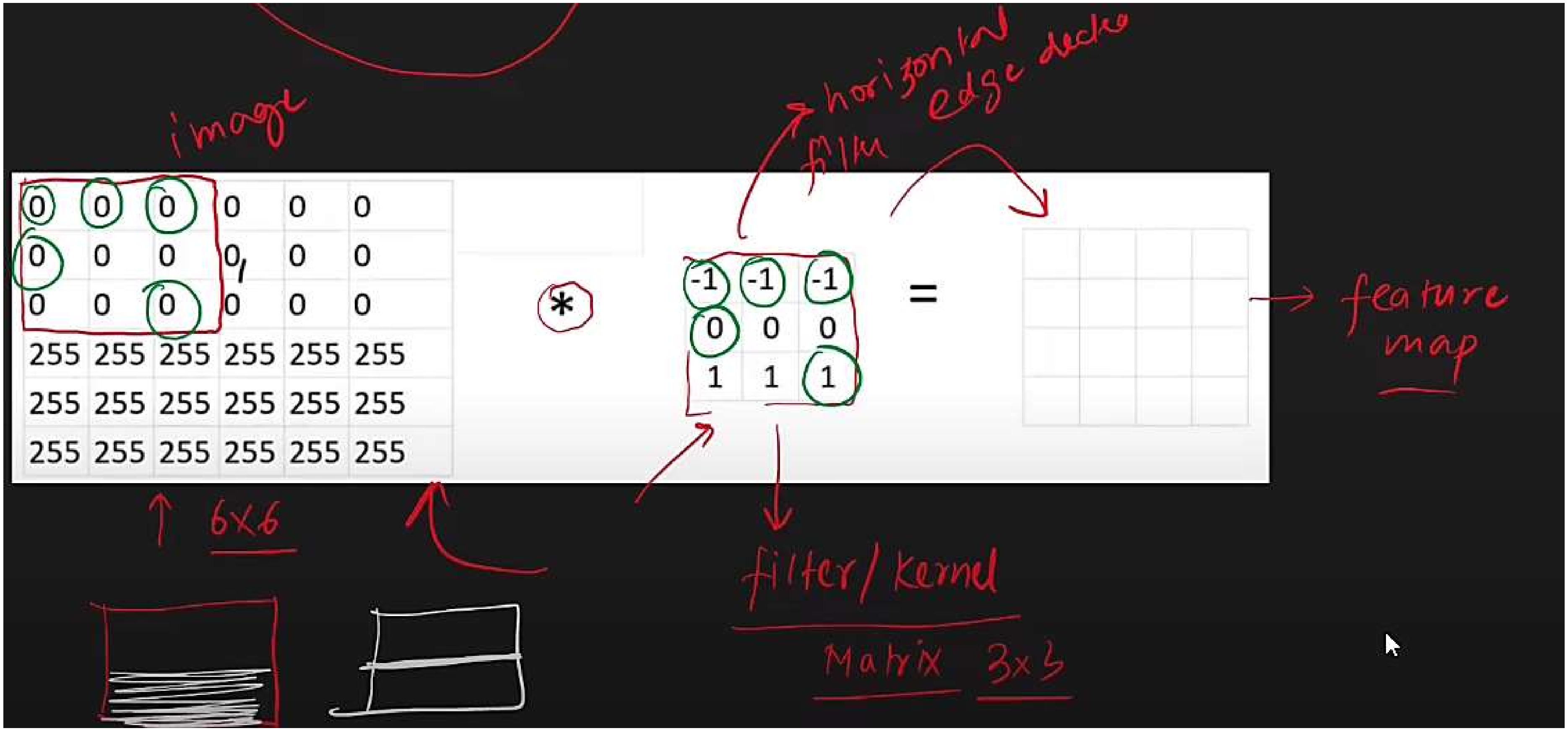
Vertical(Edge Detection)

Horizontal (Edge Detection)

Convolution Operation (Edge Detection) Example



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix} * \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \; & \; & \; & \; \\ \; & \; & \; & \; \\ \; & \; & \; & \; \\ \; & \; & \; & \; \end{bmatrix}$$



image

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

6x6

*

horizontal edge detector filter

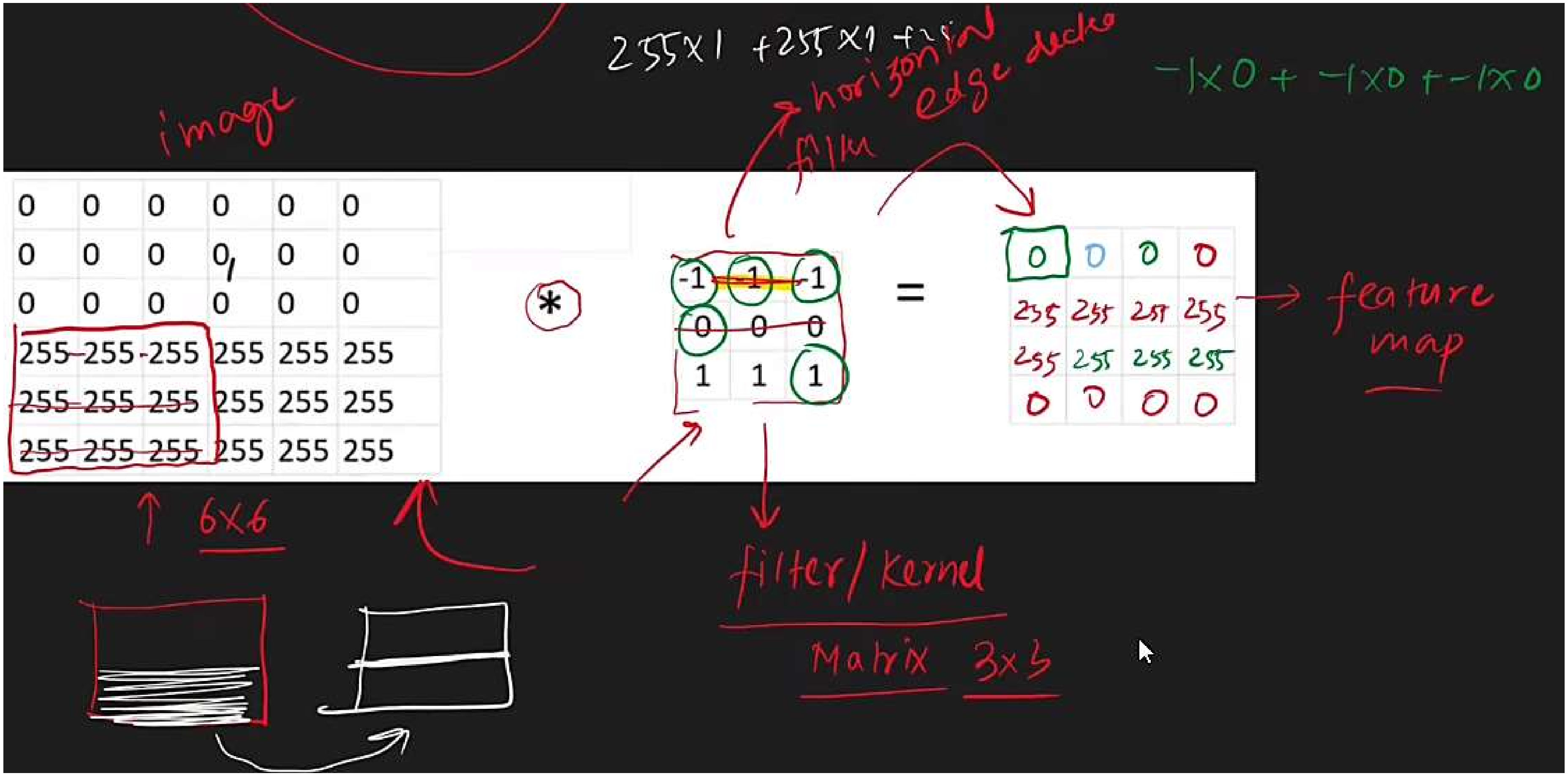
-1	-1	-1
0	0	0
1	1	1

filter/kernel

Matrix 3x3

=

feature map



255x1 +255x1 +...

image

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

6x6

*

horizontal edge detector filter

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

filter/Kernel

Matrix 3x3

=

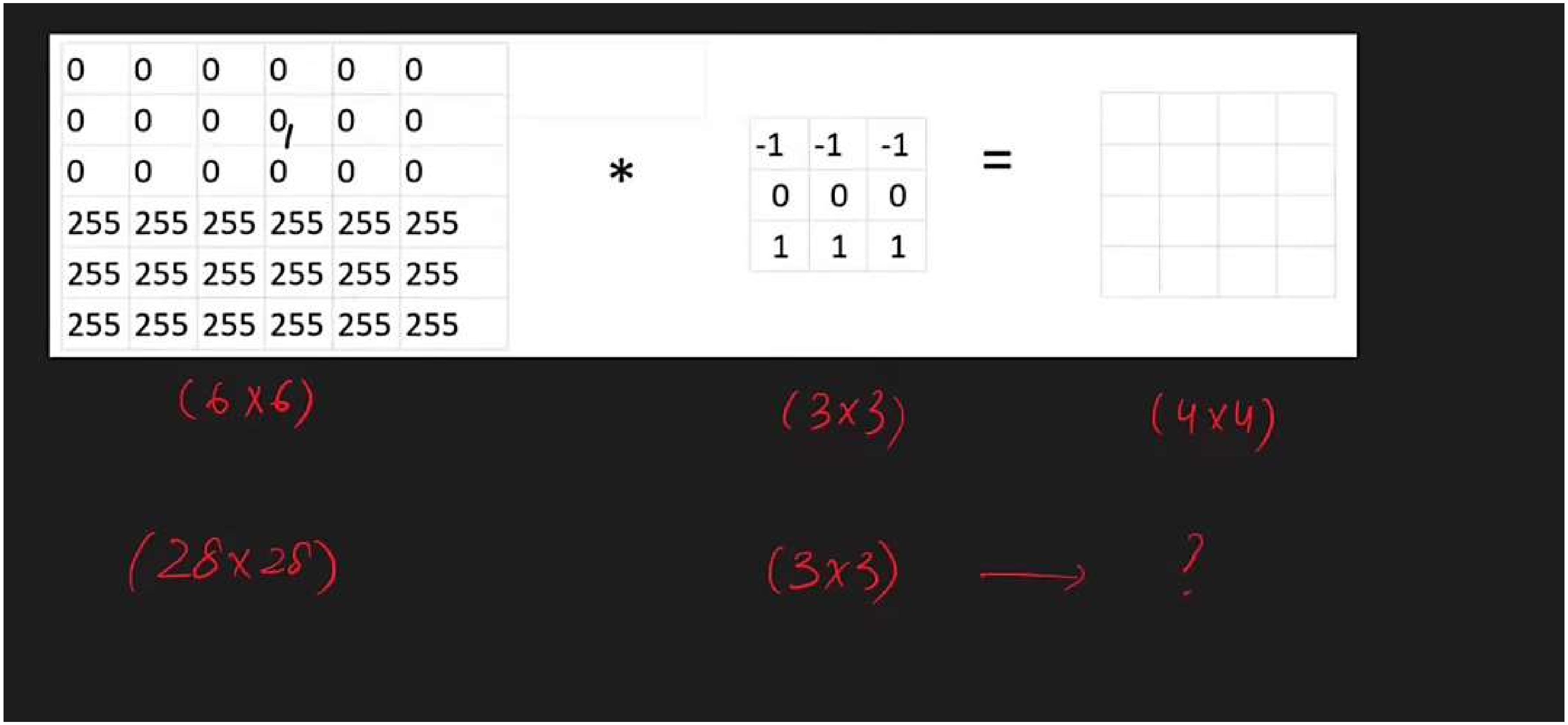
-1x0 + -1x0 + -1x0

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

feature map

Convolution Operation (Edge Detection) Demo

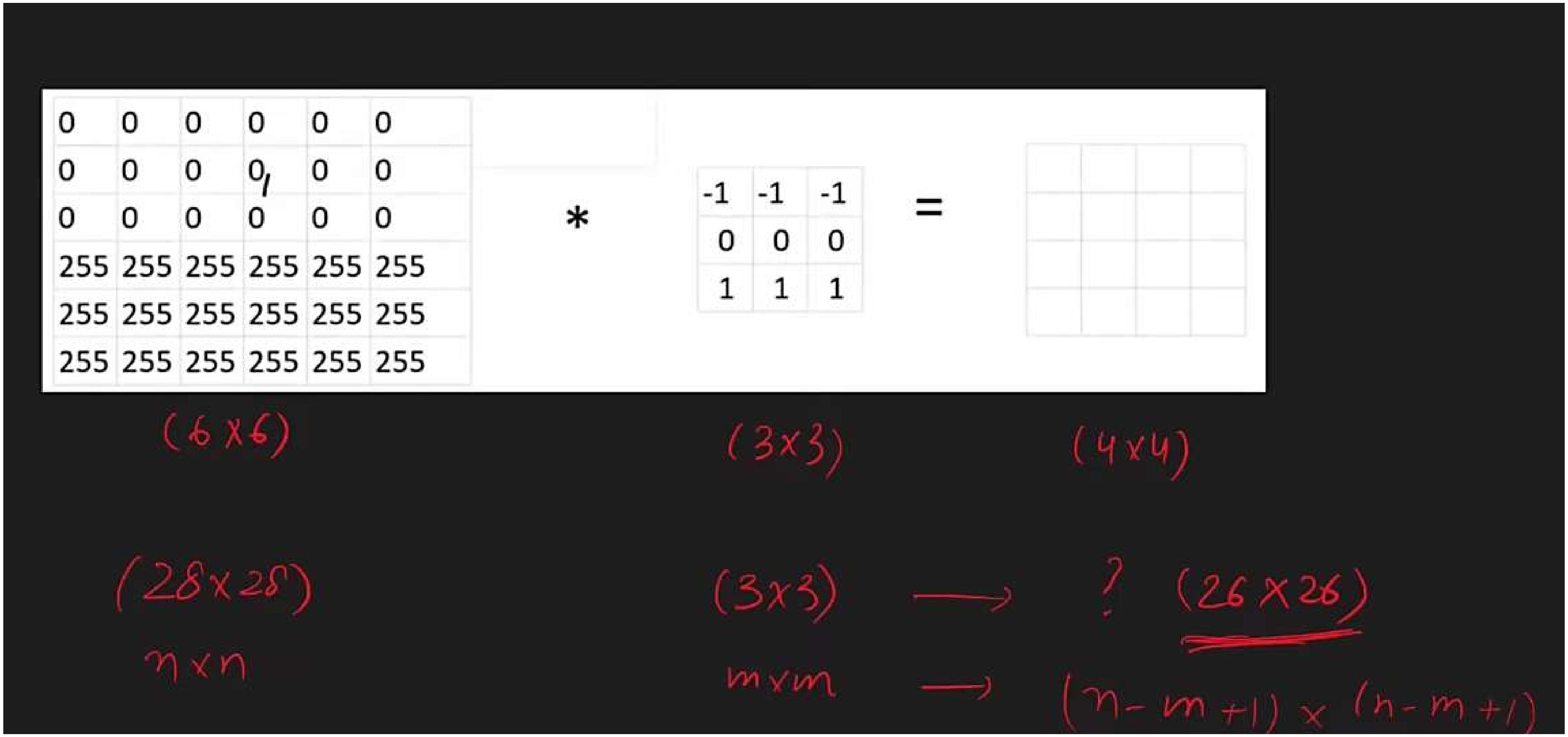
https://deeplizard.com/resource/pavq7noze2



$$\begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 255 & 255 & 255 & 255 & 255 & 255 \\ \hline 255 & 255 & 255 & 255 & 255 & 255 \\ \hline 255 & 255 & 255 & 255 & 255 & 255 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline \; & \; & \; & \; \\ \hline \; & \; & \; & \; \\ \hline \; & \; & \; & \; \\ \hline \; & \; & \; & \; \\ \hline \end{array}$$

(6x6) (3x3) (4x4)

(28x28) (3x3) → ?

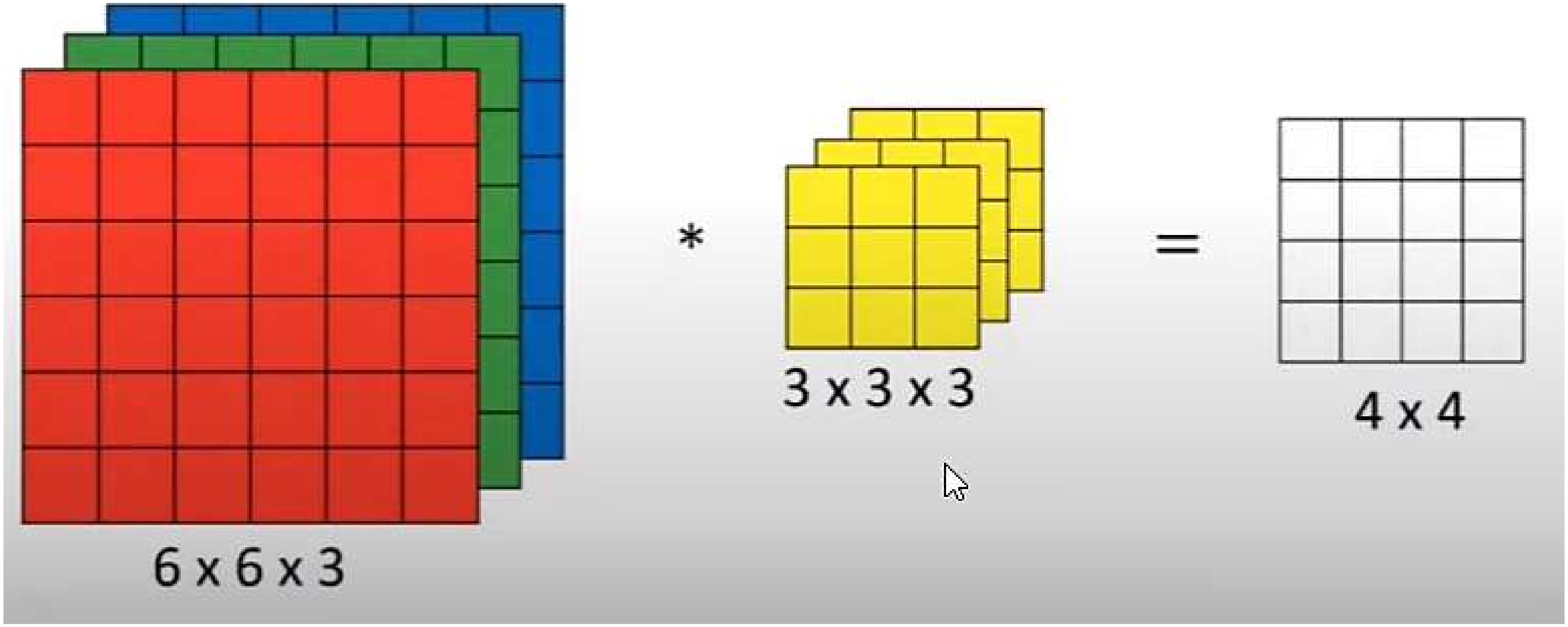


$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix} * \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \ & \ & \ & \ \\ \ & \ & \ & \ \\ \ & \ & \ & \ \\ \ & \ & \ & \ \end{bmatrix}$$

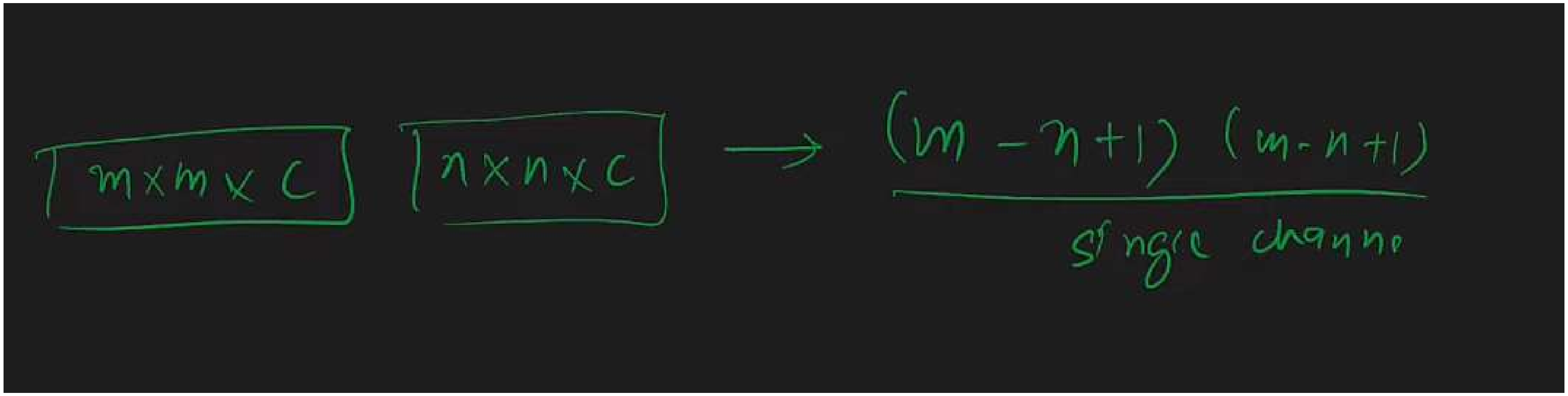
(6×6) (3×3) (4×4)

(28×28) $(3 \times 3) \longrightarrow$? $\underline{\underline{(26 \times 26)}}$

$n \times n$ $m \times m \longrightarrow (n-m+1) \times (n-m+1)$



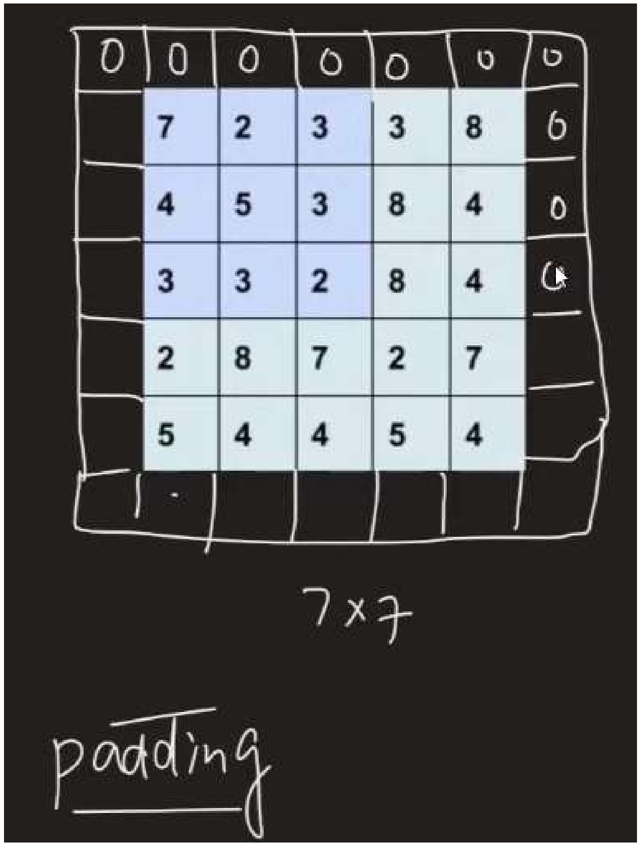
https://medium.com/swlh/convolutional-neural-networks-part-3-convolutions-over-volume-and-the-convnet-layer-91fb7c08e28b



$$\boxed{m \times m \times c} \quad \boxed{n \times n \times c} \longrightarrow \underset{\text{single channe}}{\underline{(m-n+1)\ (m-n+1)}}$$

Padding & Stride

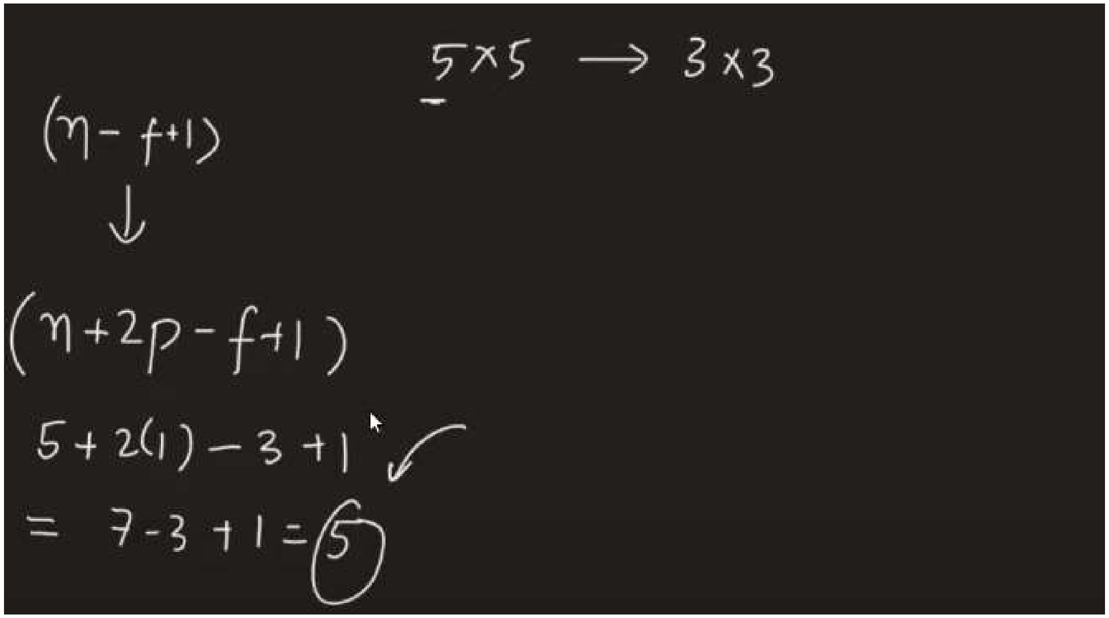
https://medium.com/latinxinai/convolutional-neural-network-from-scratch-6b1c856e1c07



0	0	0	0	0	0	0
	7	2	3	3	8	0
	4	5	3	8	4	0
	3	3	2	8	4	0
	2	8	7	2	7	
	5	4	4	5	4	

7 x 7

padding



$5 \times 5 \longrightarrow 3 \times 3$

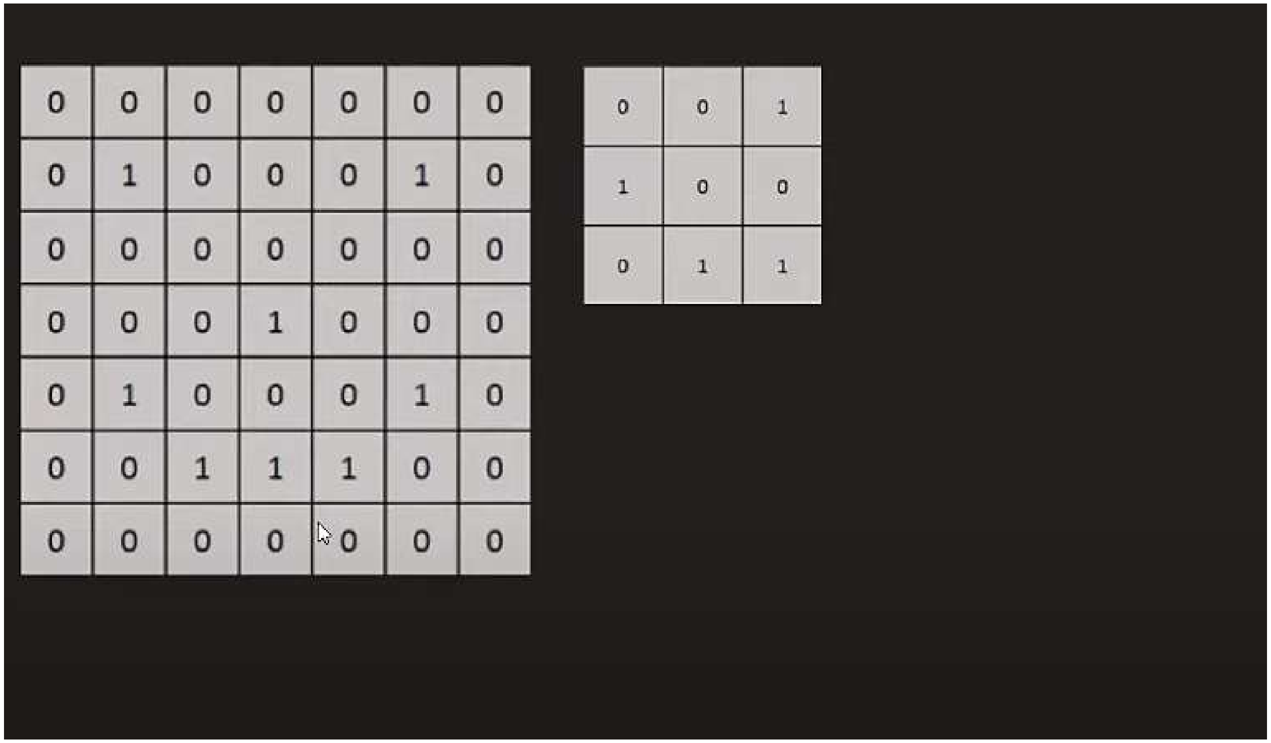
$(n - f + 1)$

$\downarrow$

$(n + 2p - f + 1)$

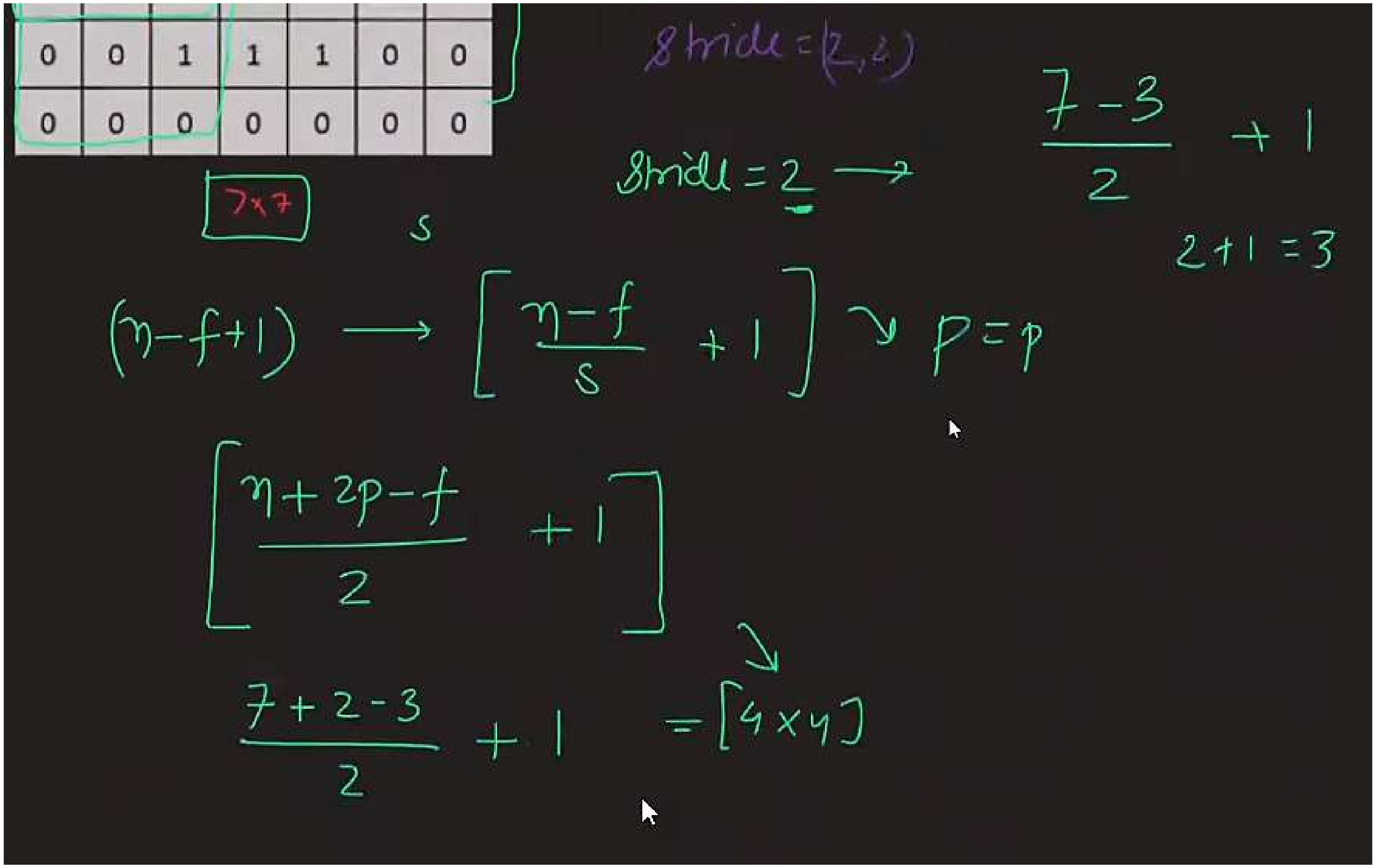
$5 + 2(1) - 3 + 1$

$= 7 - 3 + 1 = \boxed{5}$



0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

0	0	1
1	0	0
0	1	1



0	0	1	1	1	0	0
0	0	0	0	0	0	0

Stride = (2, 2)

Stride = 2 → $\frac{7-3}{2}+1$

$2+1=3$

$\boxed{7\times 7}$

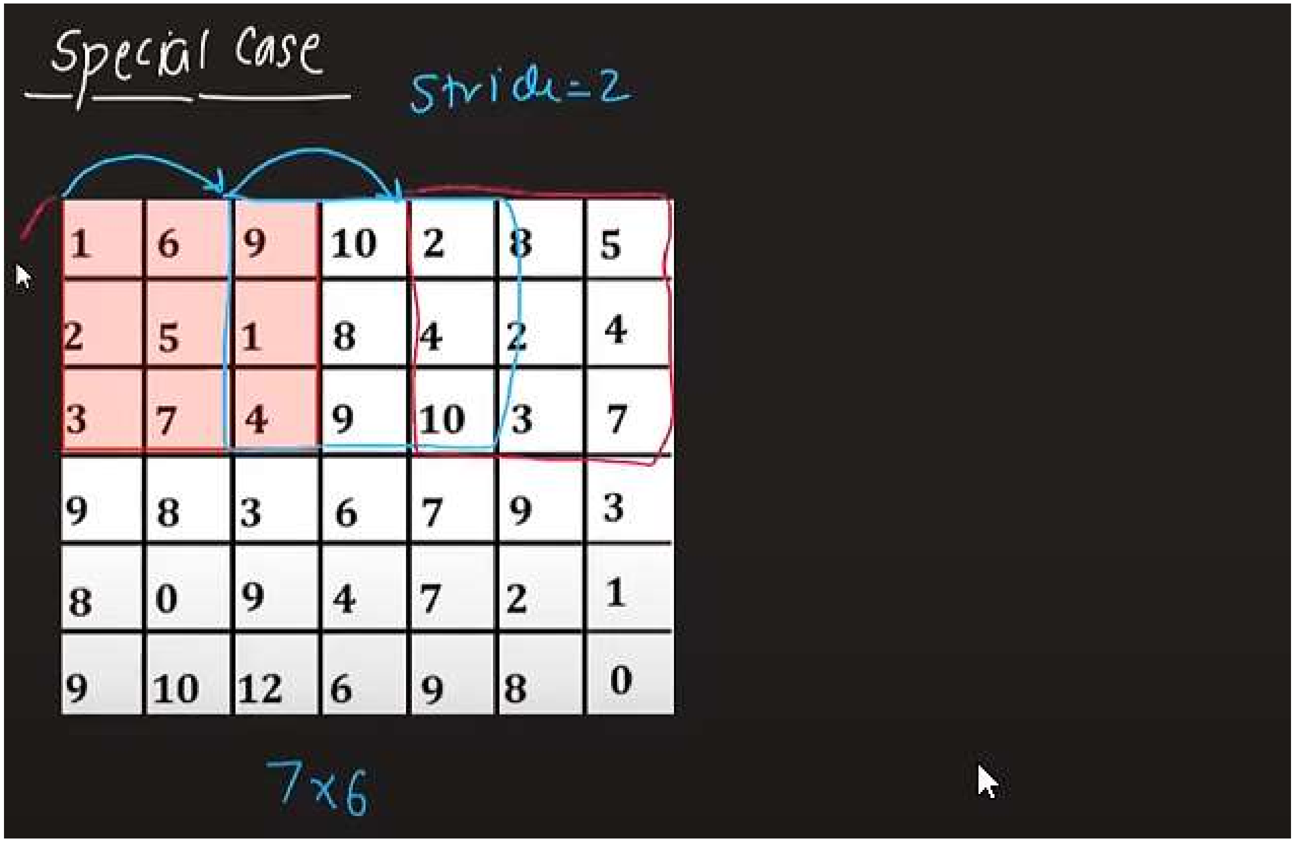
s

$(n-f+1) \longrightarrow \left[\frac{n-f}{s}+1\right] \searrow p=p$

$$\left[\frac{n+2p-f}{2}+1\right]$$

$$\searrow$$

$$\frac{7+2-3}{2}+1 = [4\times 4]$$



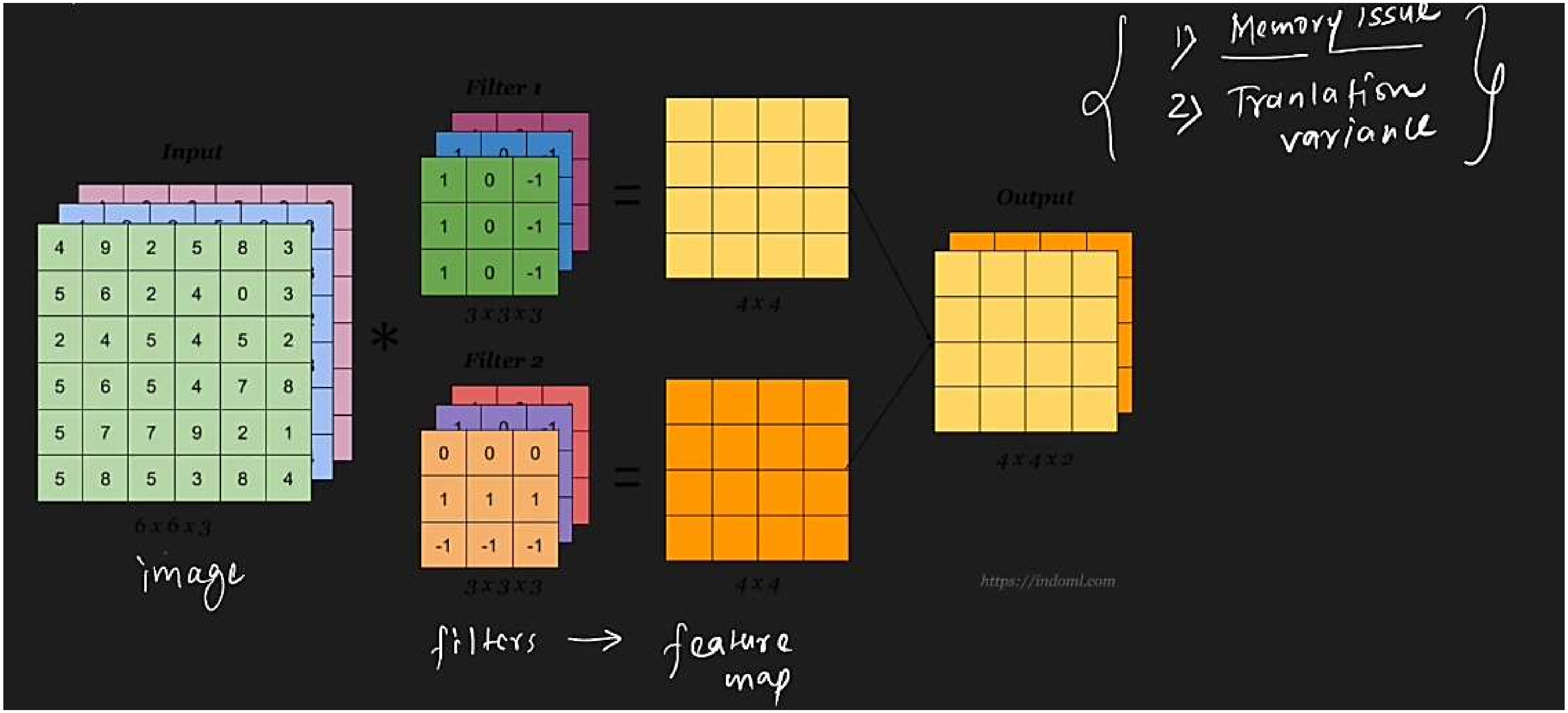
Special case

stride=2

1	6	9	10	2	8	5
2	5	1	8	4	2	4
3	7	4	9	10	3	7
9	8	3	6	7	9	3
8	0	9	4	7	2	1
9	10	12	6	9	8	0

7x6

Max Pooling in CNN



4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

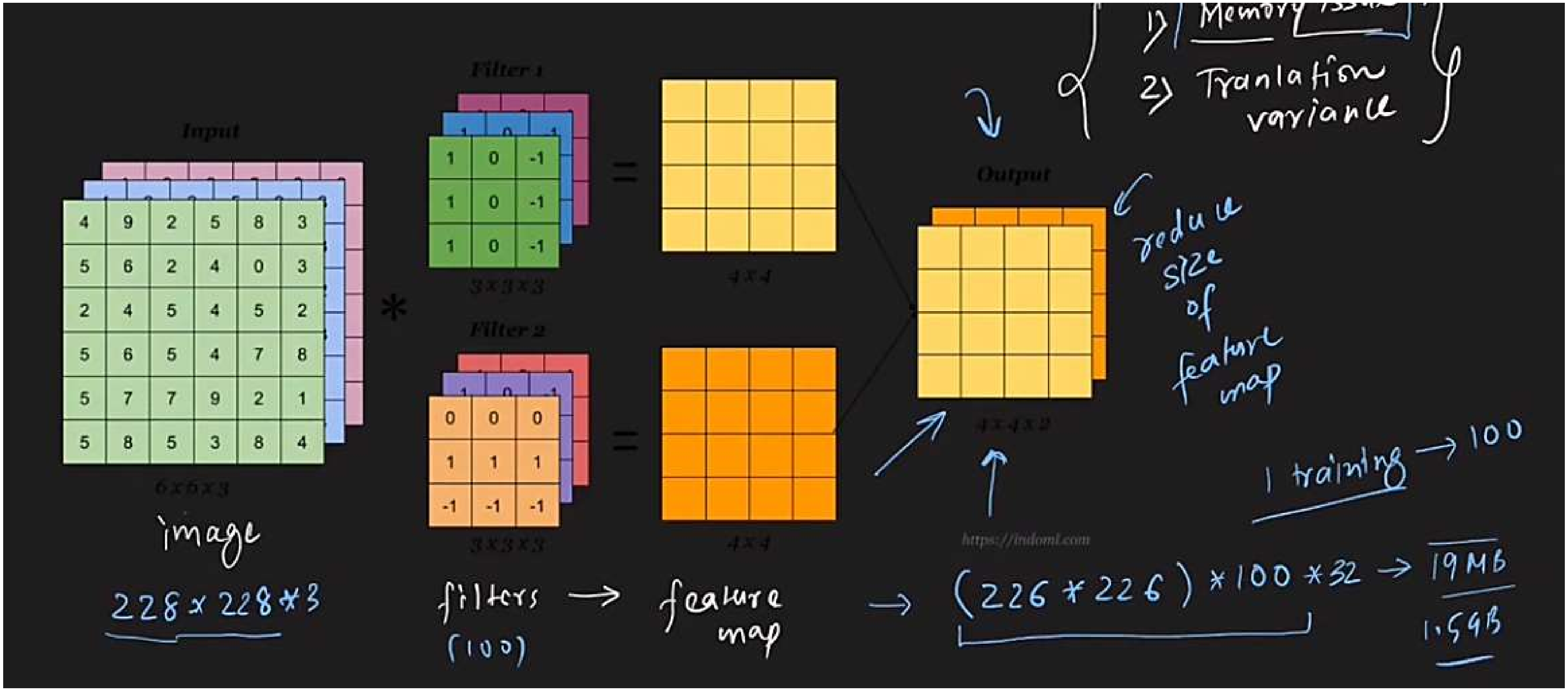
1	0	-1
1	0	-1
1	0	-1

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

filters → feature map

1) Memory issue
2) Tranlation variance

Problem with Convolution

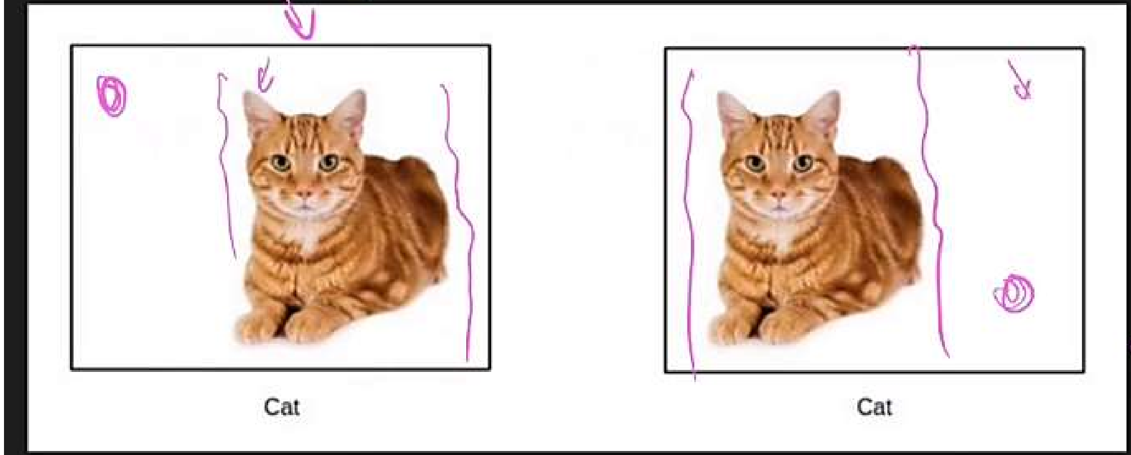


4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

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

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

Translation Variance



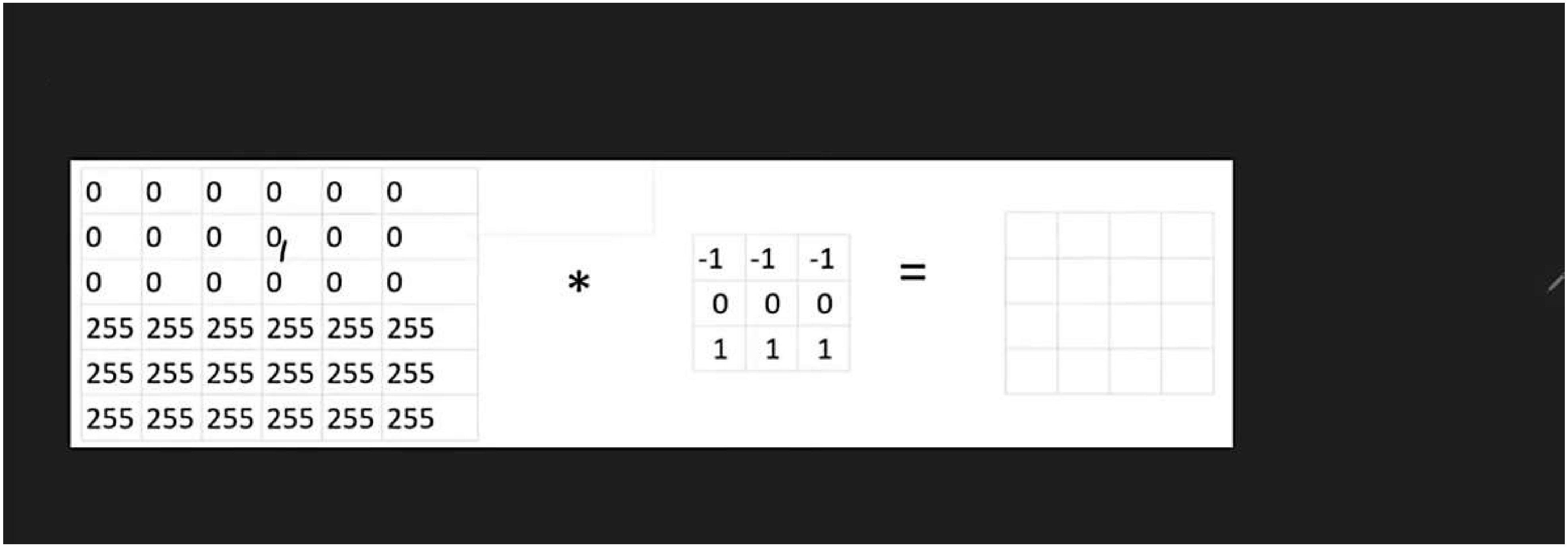
features

{ location dependent }

location

pooling { down sample your feature map }

Pooling

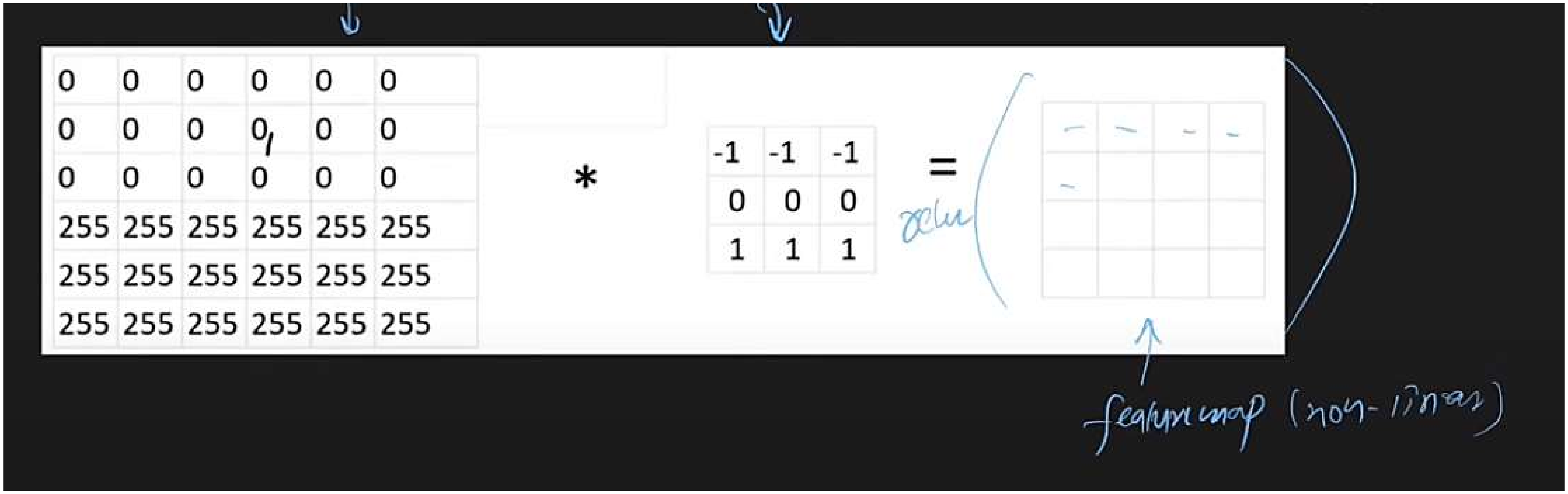


0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255

*

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

=



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

*

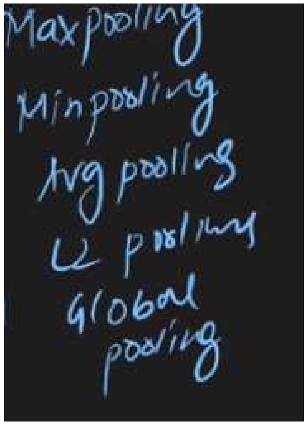
-1	-1	-1
0	0	0
1	1	1

= relu(

–	–	–	–
–			

)

featuremap (non-linear)



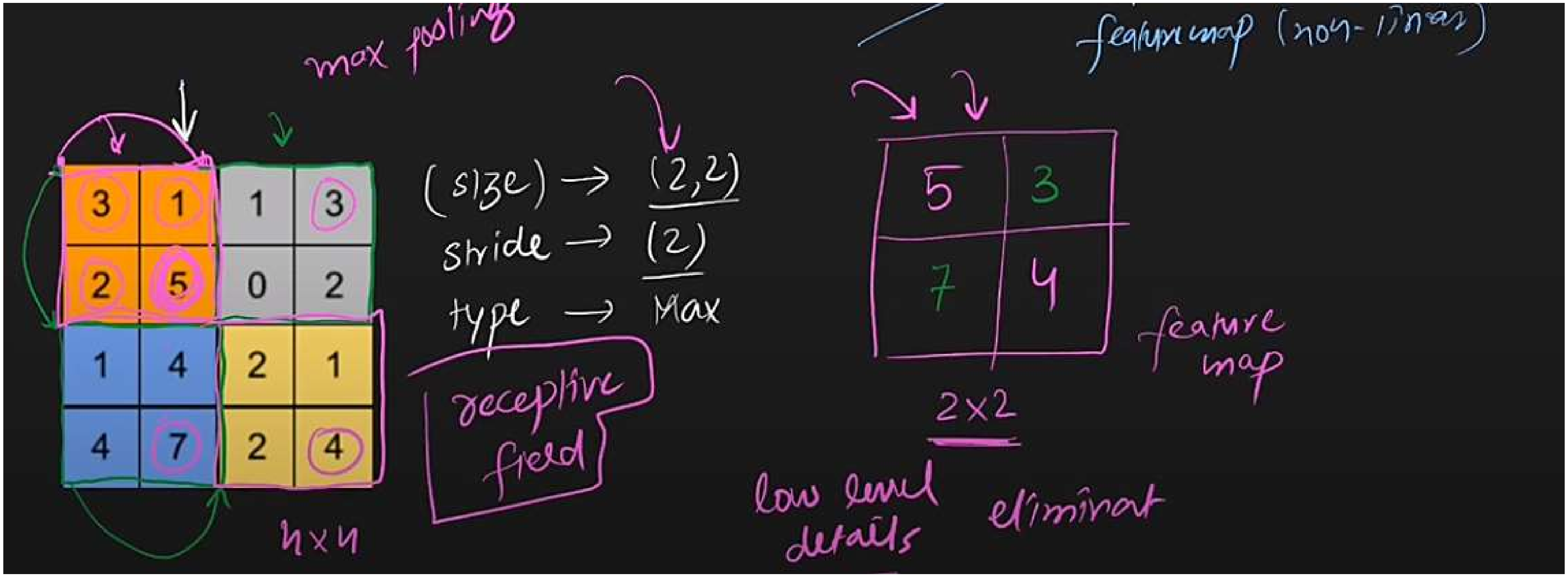
Maxpooling

Minpooling

Avg pooling

L2 pooling

Global pooling



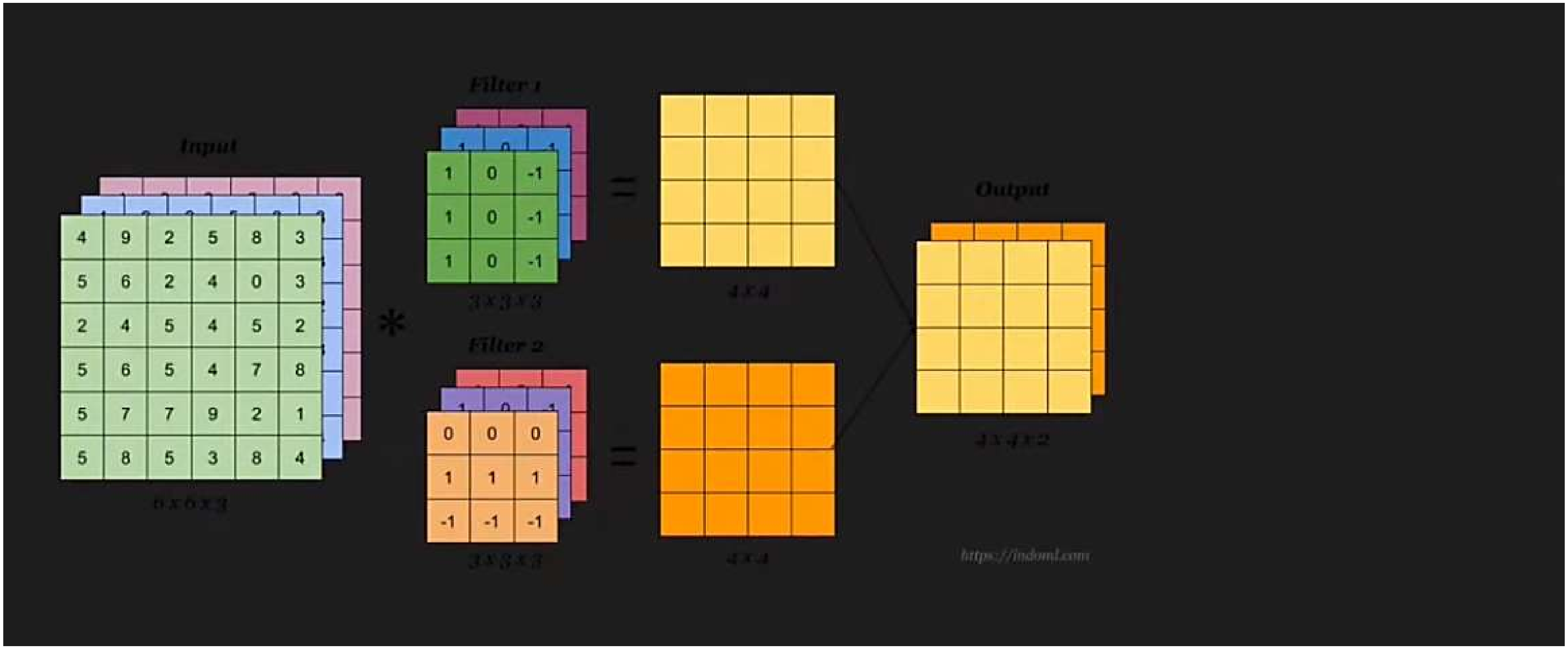
3	1	1	3
2	5	0	2
1	4	2	1
4	7	2	4

(size) → (2,2)
stride → (2)
type → Max

5	3
7	4

https://deeplizard.com/resource/pavq7noze3

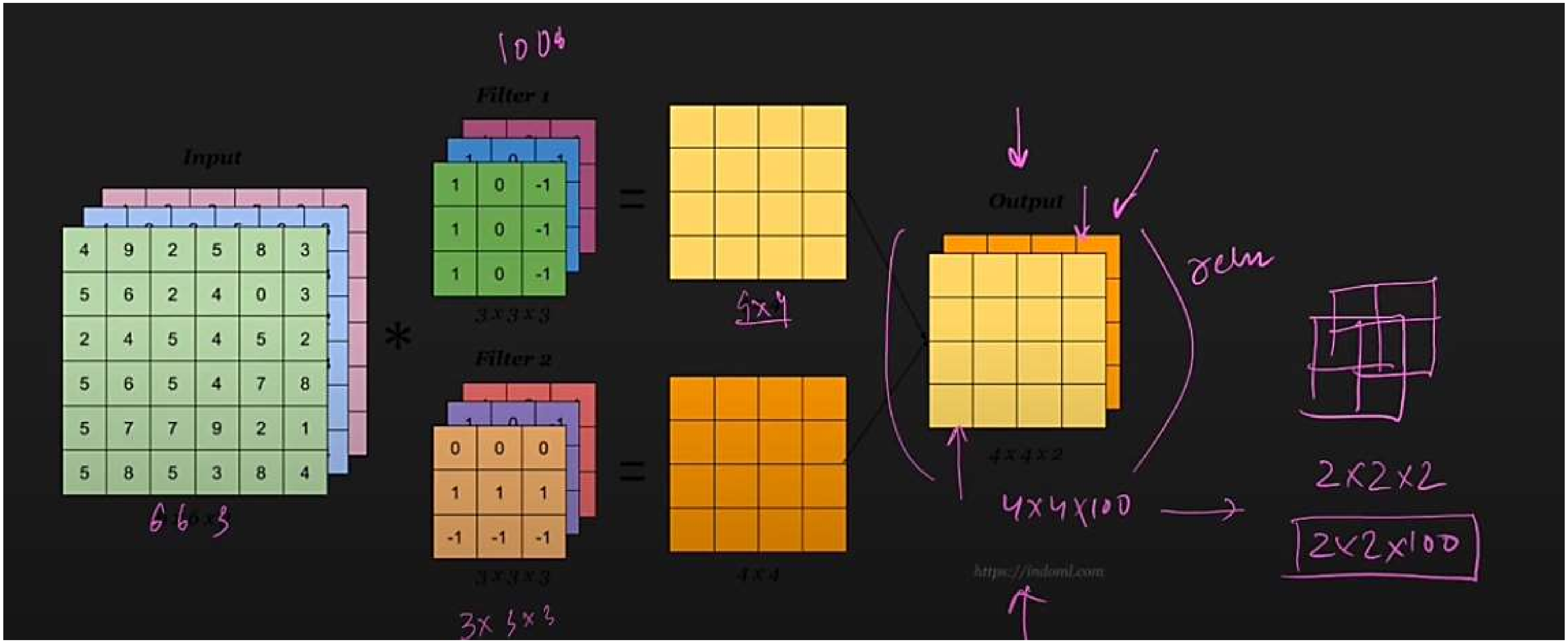
Problem on volumes



4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

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

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

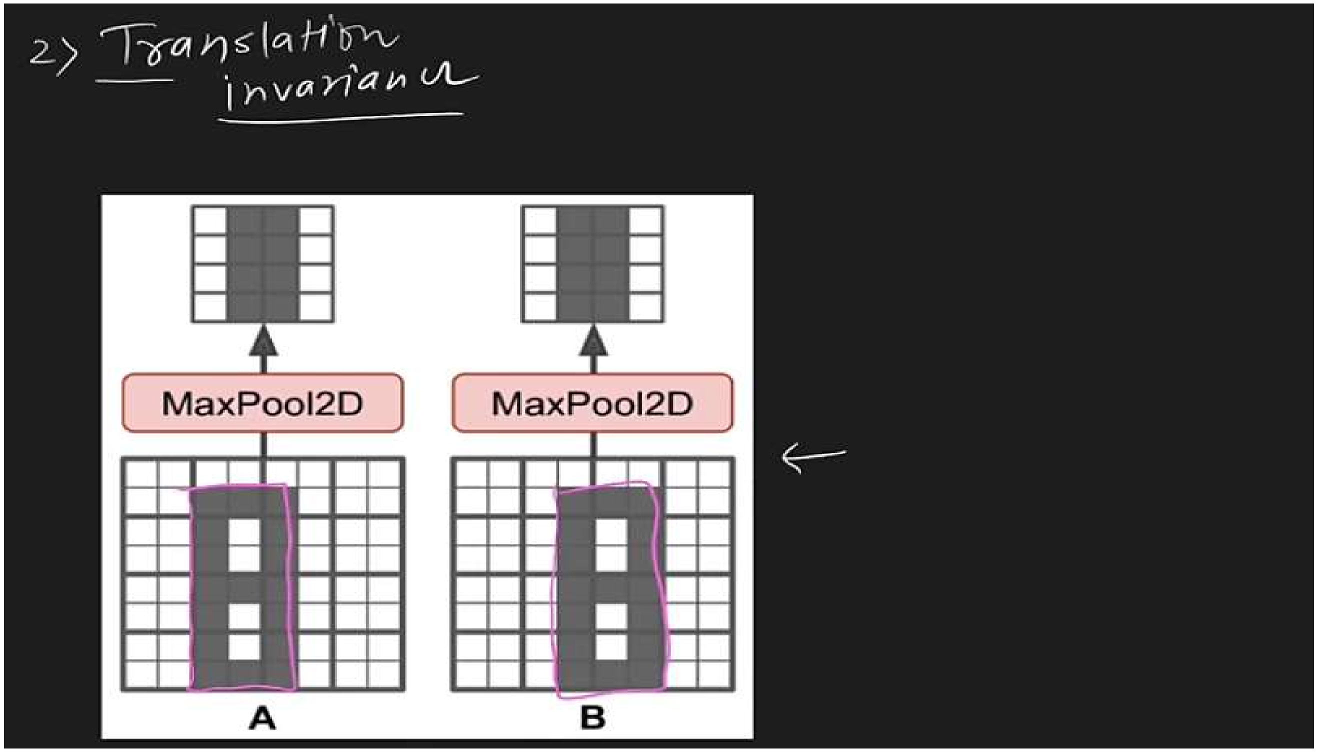


4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

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

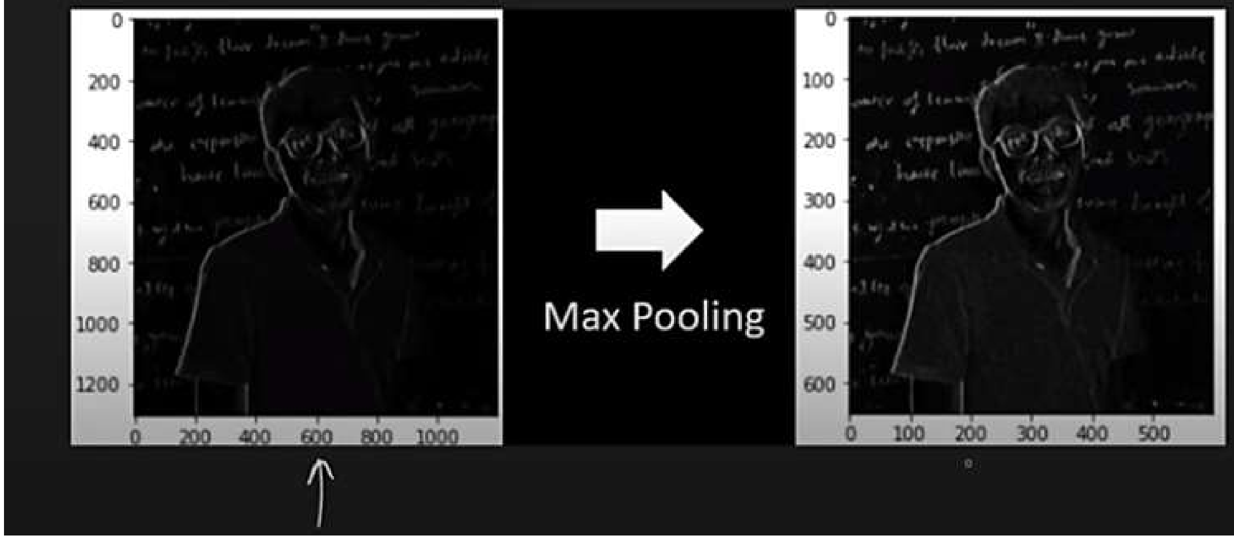
0	0	0
1	1	1
-1	-1	-1

Advantages



2) Translation invariance

3) Enhanced Features
(only in case of Maxpooling)



Disadvantages

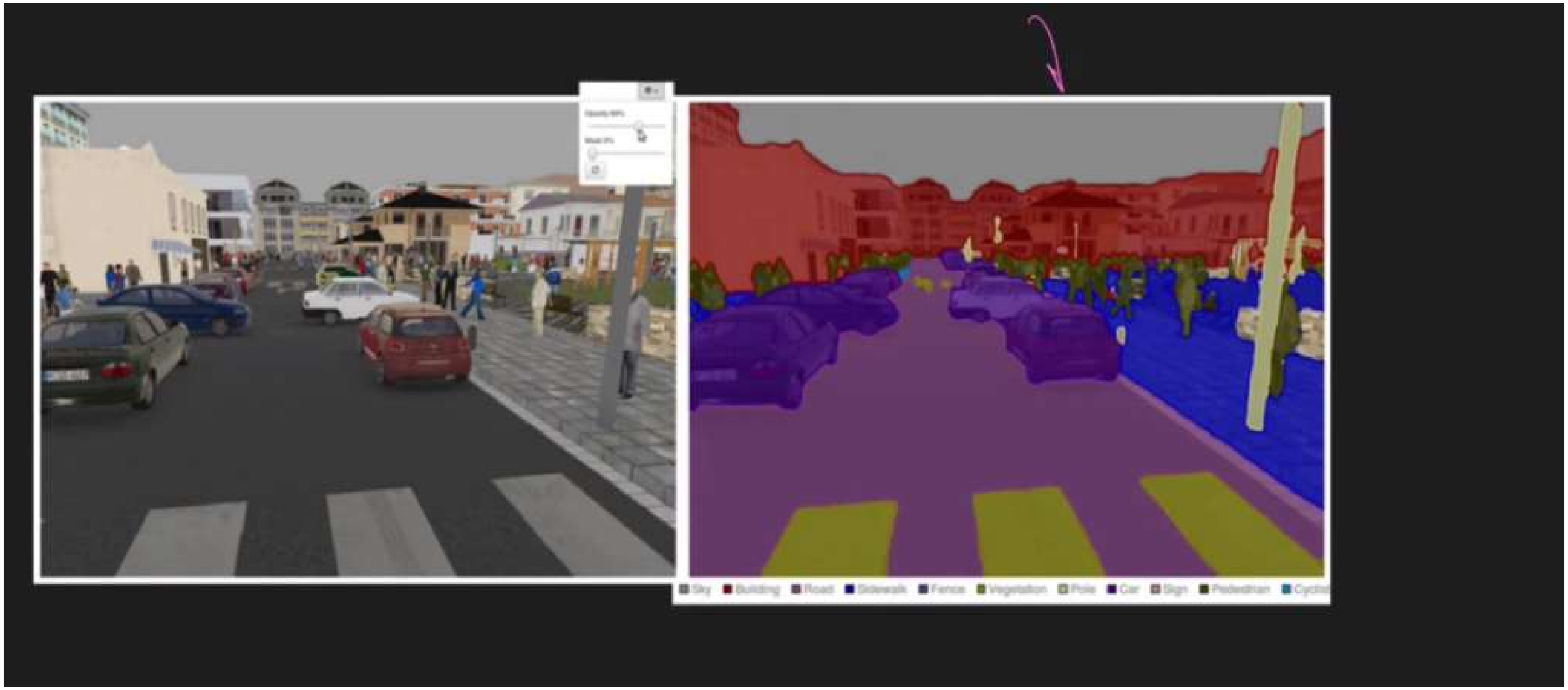
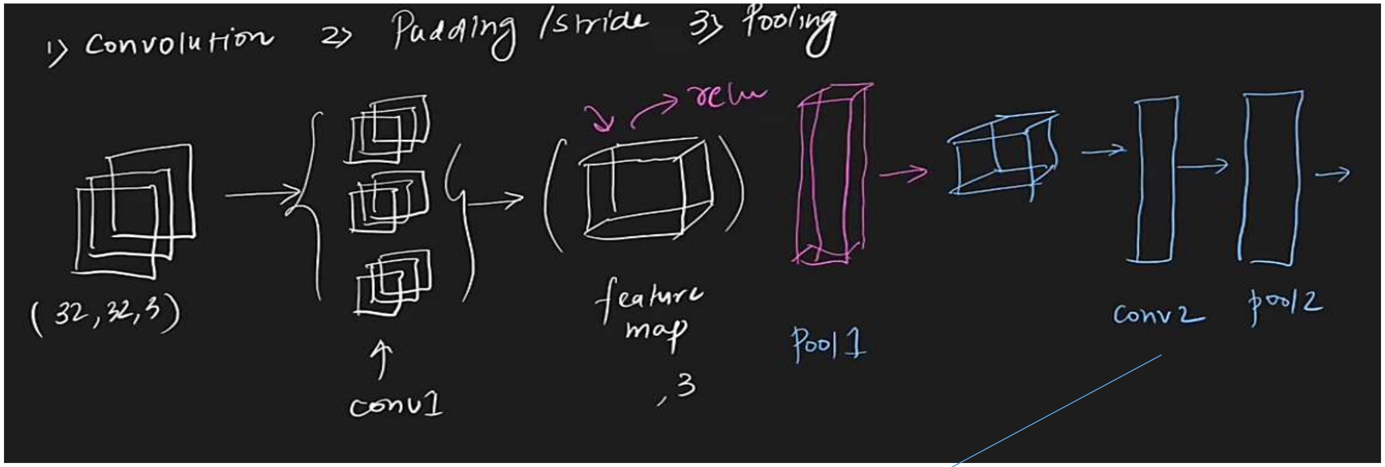
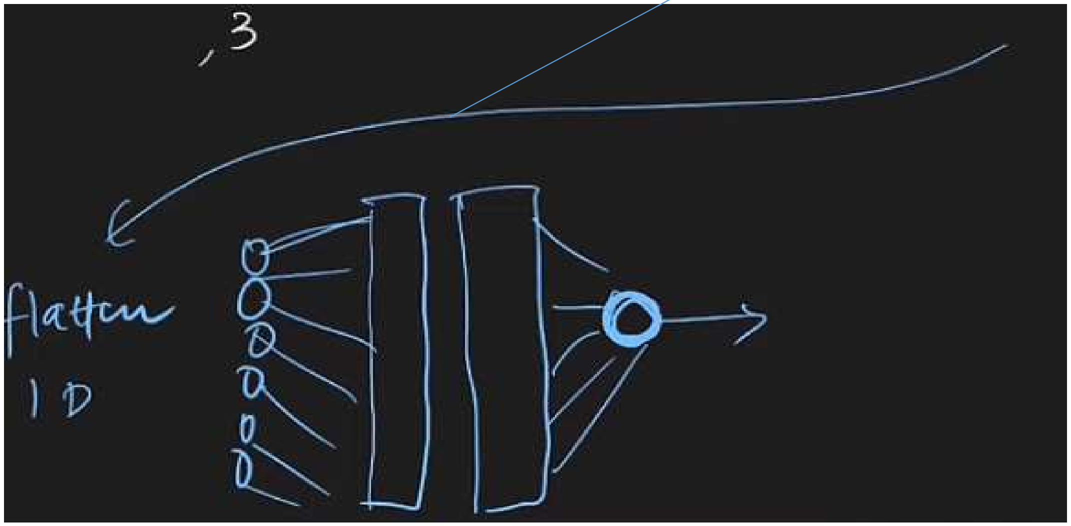
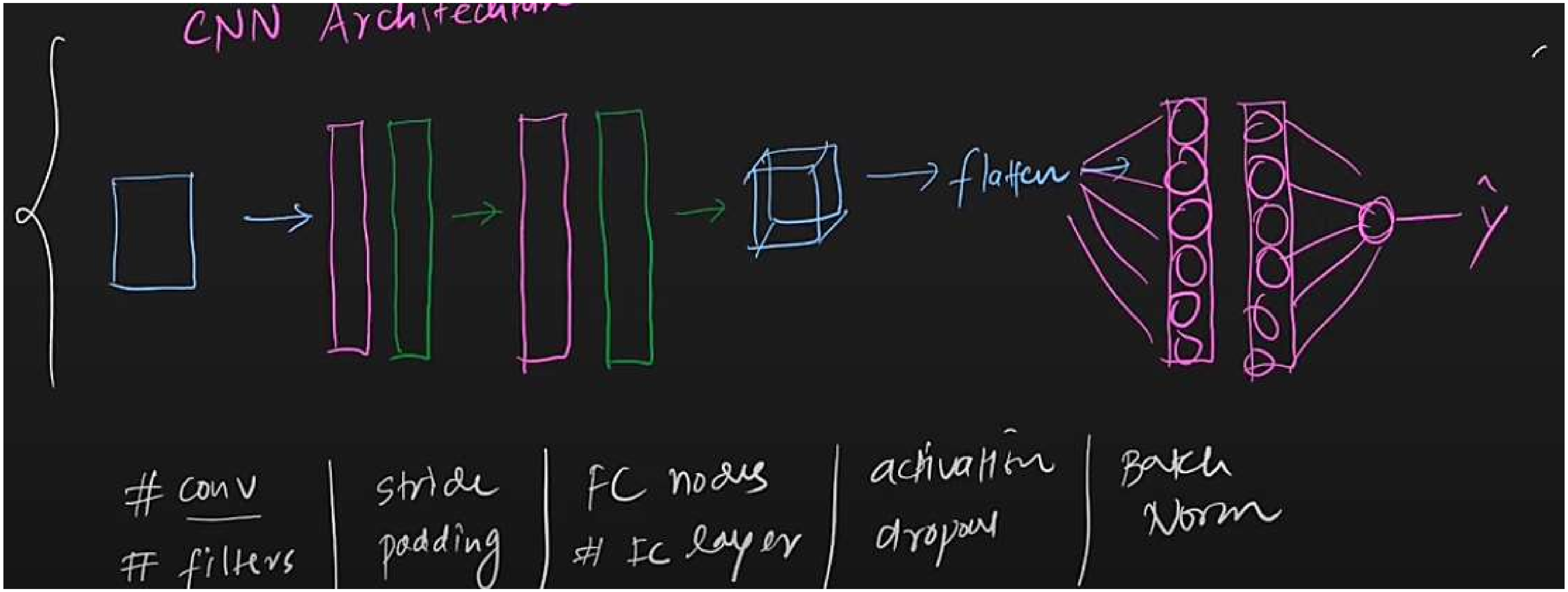


Image Segmentation


1> Convolution
2> Padding /stride
3> Pooling
relu
(32,32,3)
conv1
feature map
,3
Pool 1
conv2
pool2

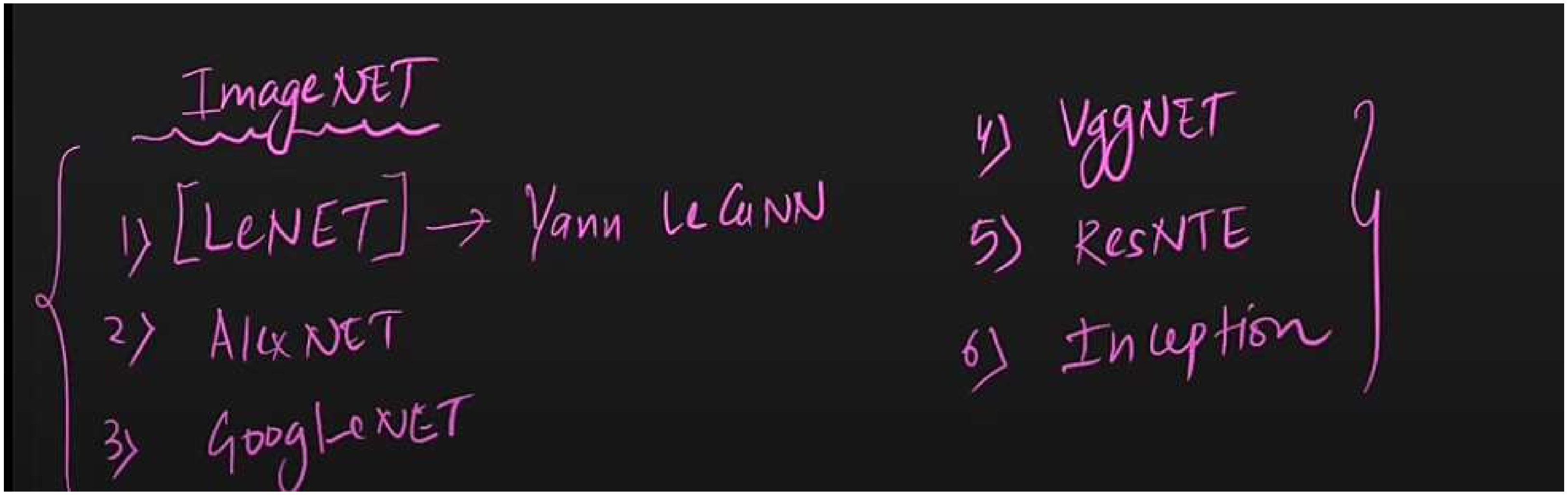

,3
flatten
1D



CNN Architecture

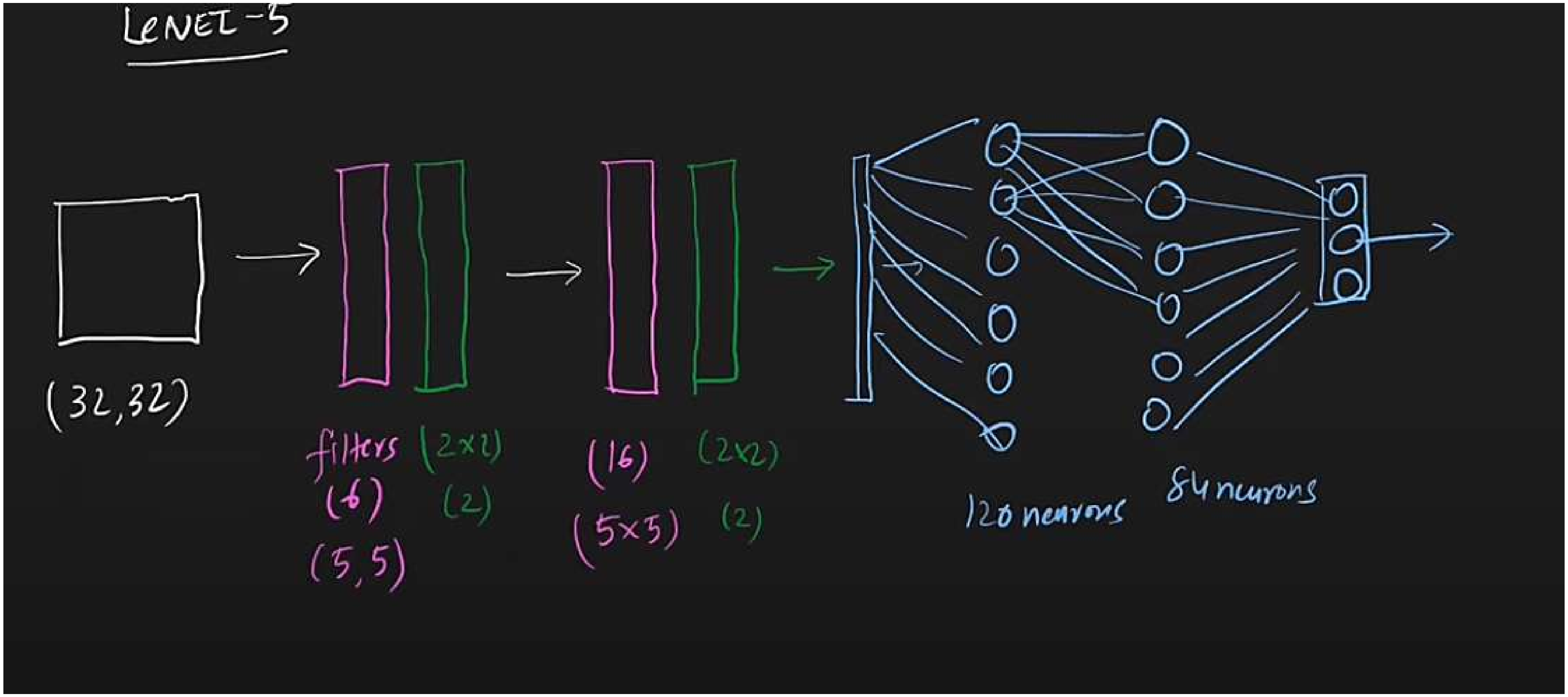
conv | stride | FC nodes | activation | Batch Norm

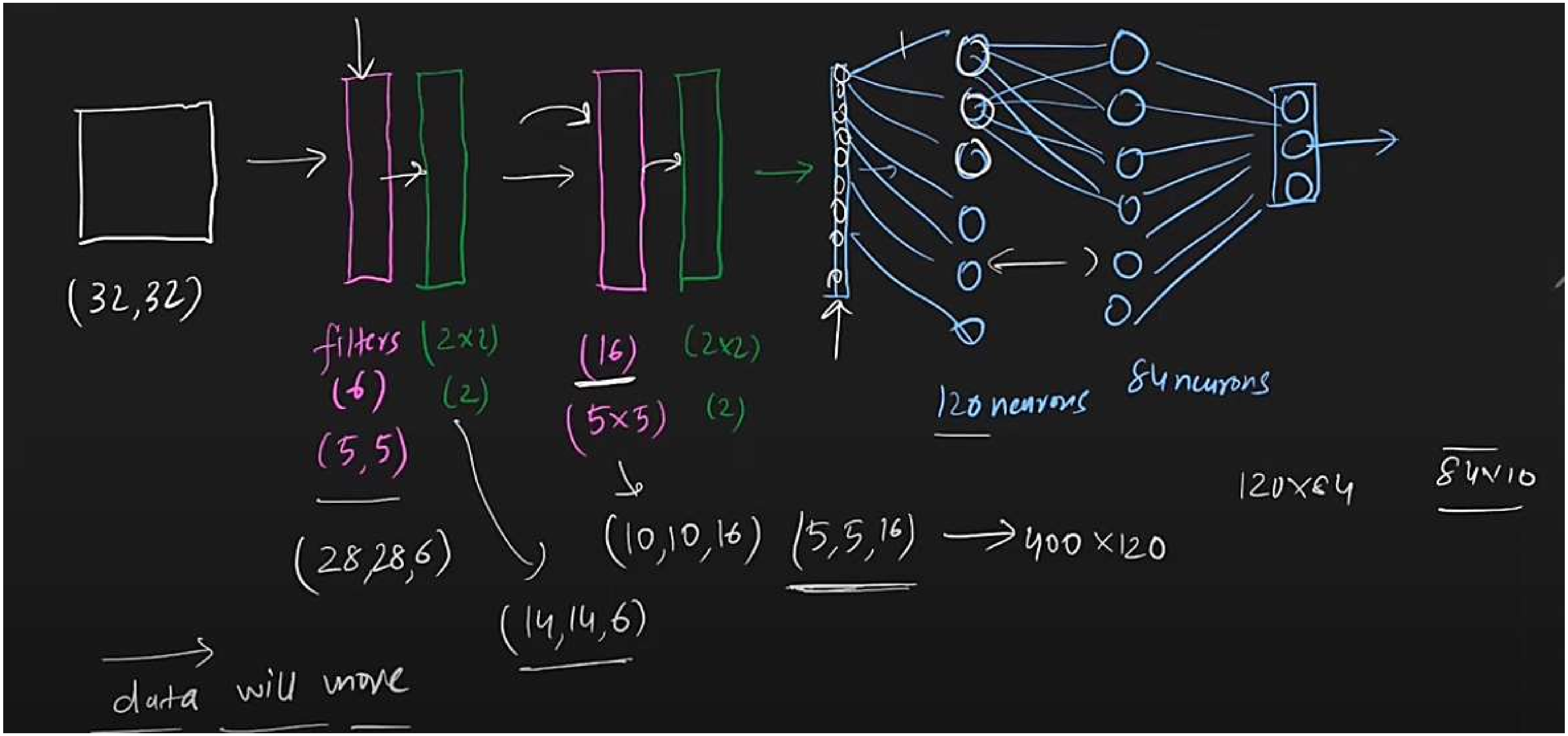
filters | padding | # FC layer | dropout



ImageNET

1) [LeNET] → Yann LeCuNN
2) AlexNET
3) GoogLeNET
4) VggNET
5) ResNTE
6) Inception

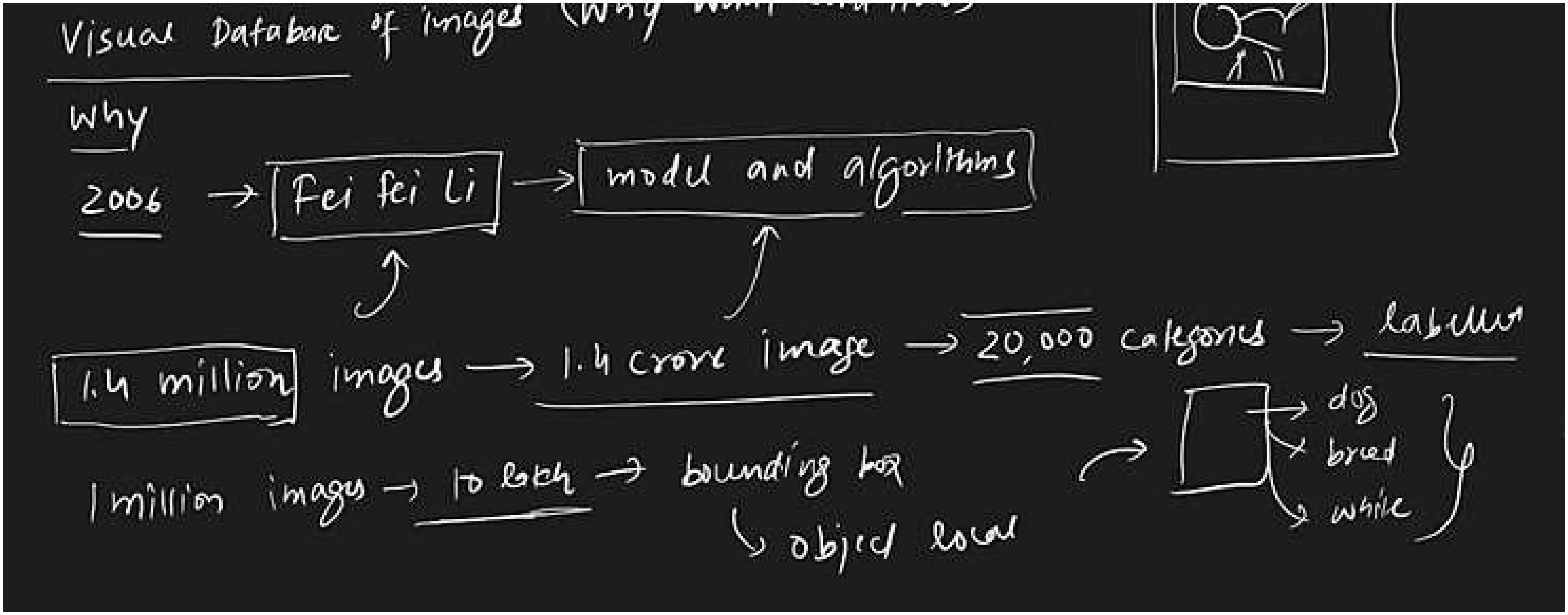

LeNET-5
(32,32)
filters
(6)
(5,5)
(2x2)
(2)
(16)
(5x5)
(2x2)
(2)
120 neurons
84 neurons


(32,32)
filters
(6)
(5,5)
(2x2)
(2)
(28,28,6)
(14,14,6)
(16)
(5x5)
(2x2)
(2)
(10,10,16)
(5,5,16)
400 x120
120 neurons
84 neurons
120x84
84x10
data will move

Pre-trained Models

1- Data Hungry
2- Time

A pre-trained model is a (DL) model that has been trained on a large dataset and can be fine-tuned for a specific task. Pre-trained models are often used as a starting point for developing DL models, as they provide a set of initial weights and biases that can be fine-tuned for a specific task.



Visual Database of images (Why What and How)

Why

2006 → Fei fei Li → model and algorithms

1.4 million images → 1.4 crore image → 20,000 categories → labellin

1 million images → 10 lakh → bounding box

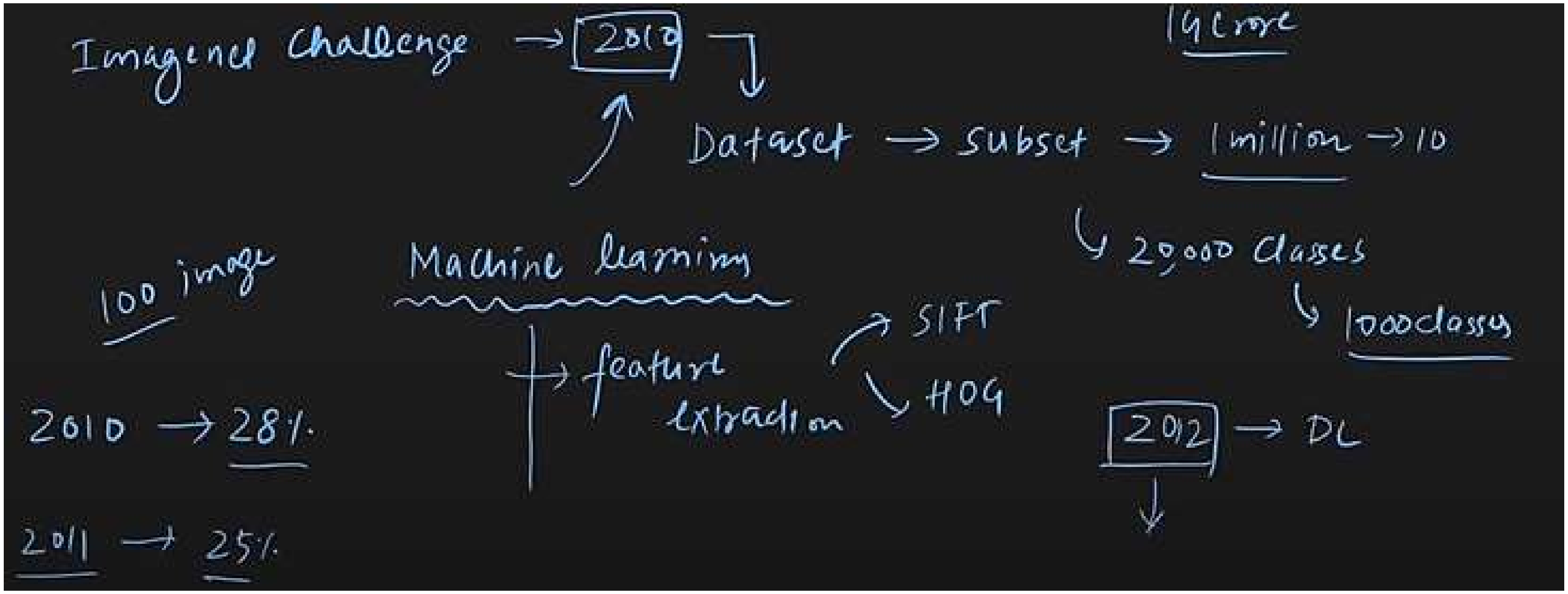
↳ object local

dog

breed

white

How → crowd sourcing → Amazon mechanical Turk



Imagenet Challenge → 2010 → Dataset → subset → 1 million → 10

14 crore

↳ 20,000 classes ↳ 1000 classes

100 image

Machine learning

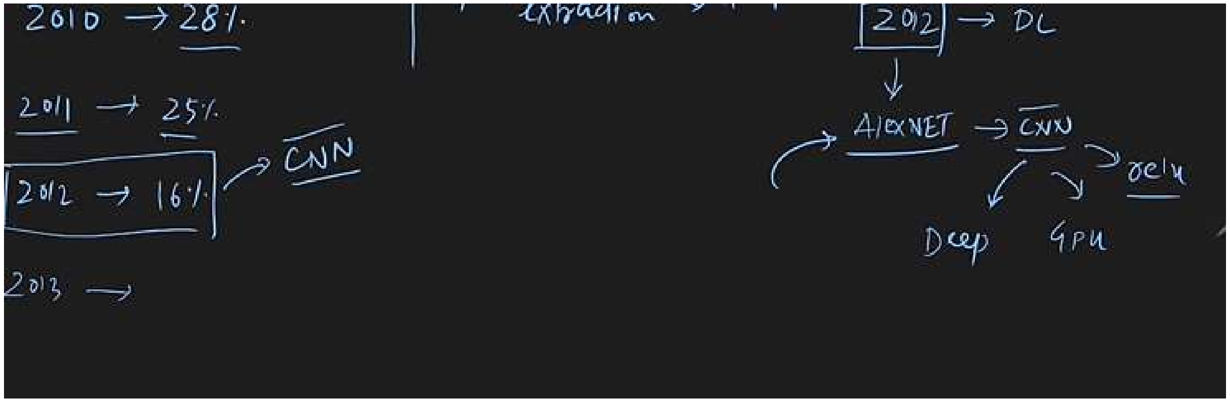
→ feature extraction → SIFT, HOG

2010 → 28%

2011 → 25%

2012 → DL

↓



2010 → 28%

2011 → 25%

2012 → 16% → CNN

2013 →

extraction

2012 → DL

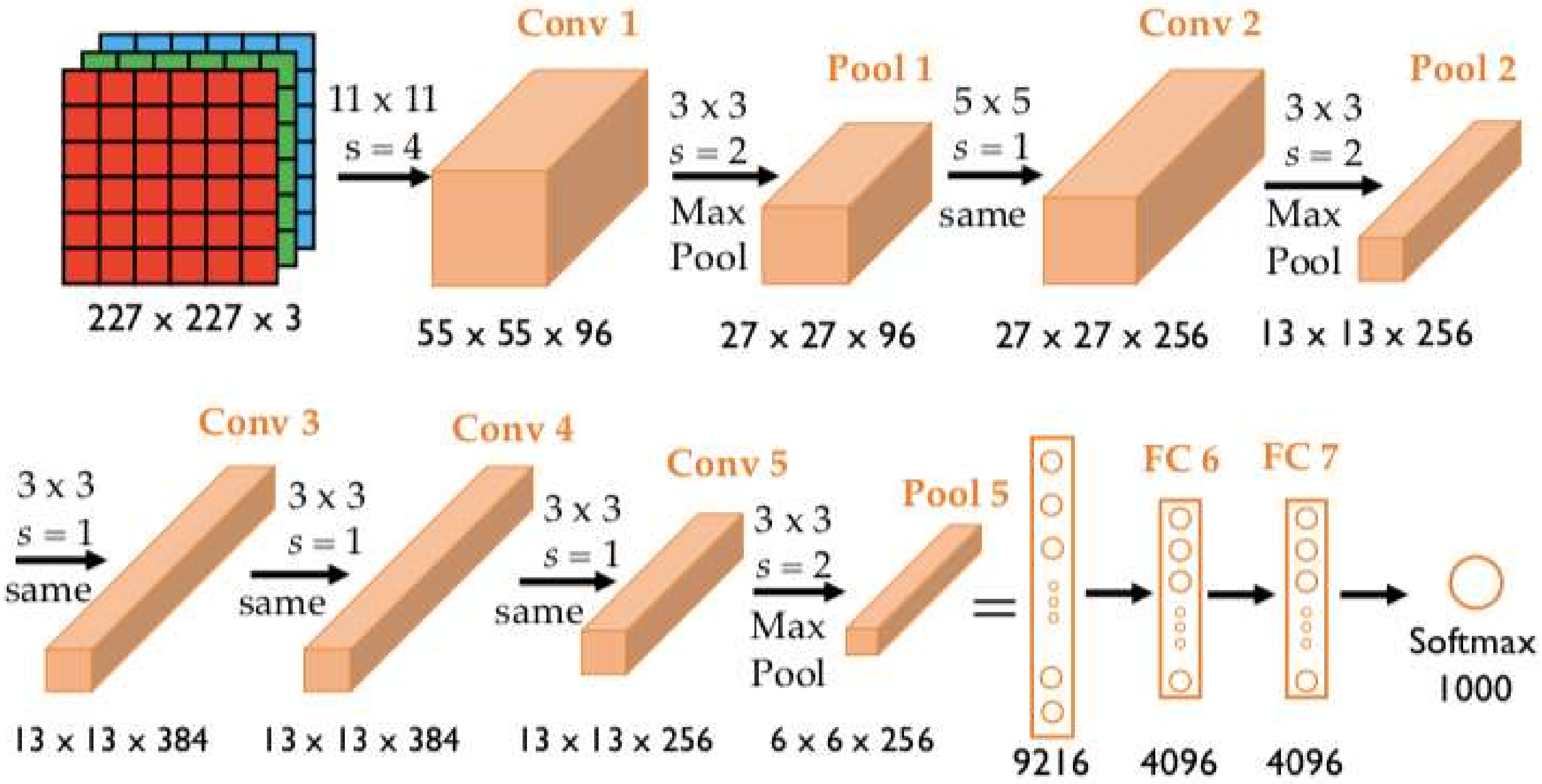
↓

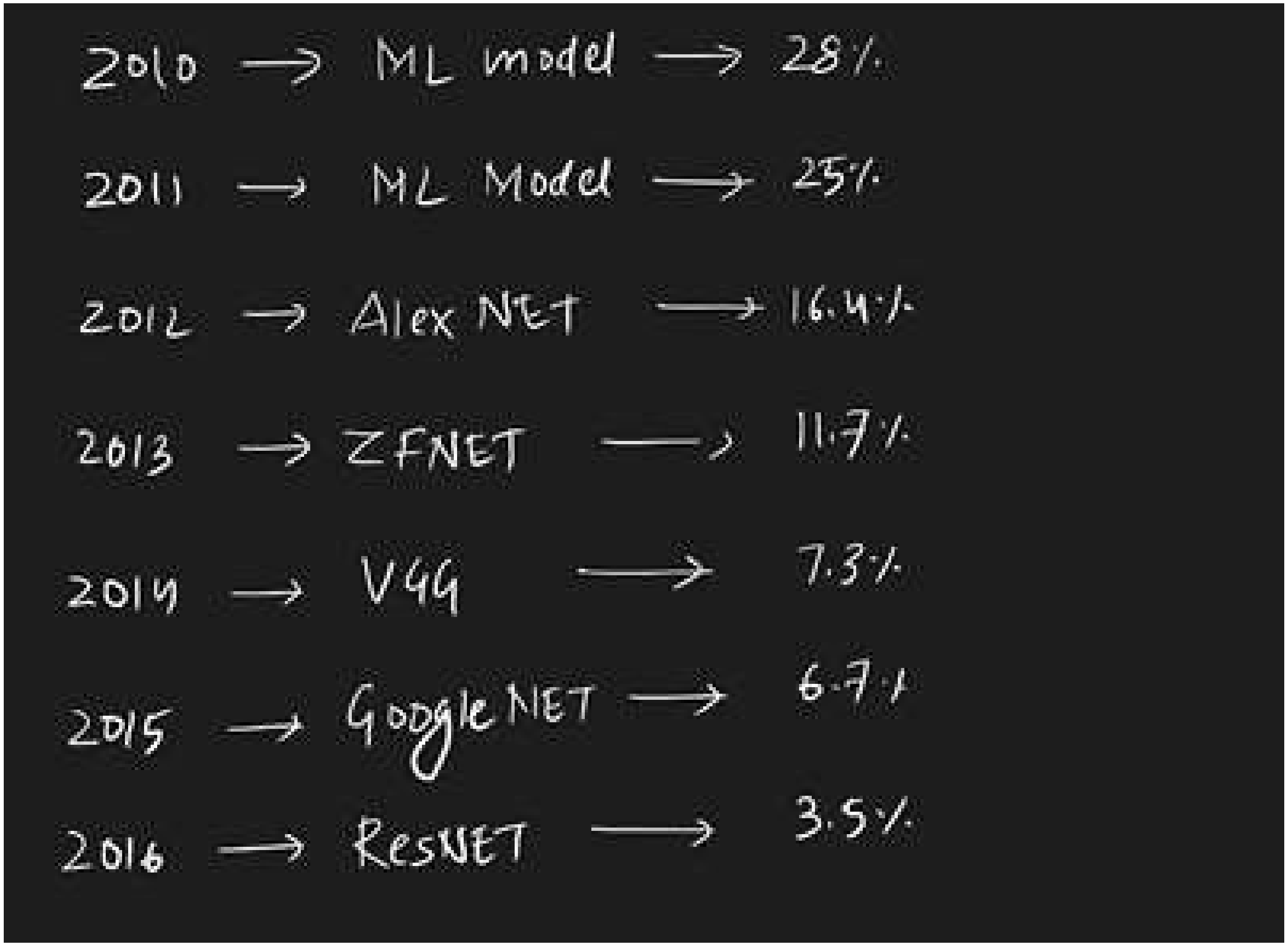
AlexNET → CNN → relu

Deep

GPU

AlexNet


227 x 227 x 3
11 x 11
s = 4
Conv 1
55 x 55 x 96
3 x 3
s = 2
Max
Pool
Pool 1
27 x 27 x 96
5 x 5
s = 1
same
Conv 2
27 x 27 x 256
3 x 3
s = 2
Max
Pool
Pool 2
13 x 13 x 256
3 x 3
s = 1
same
Conv 3
13 x 13 x 384
3 x 3
s = 1
same
Conv 4
13 x 13 x 384
3 x 3
s = 1
same
Conv 5
13 x 13 x 256
3 x 3
s = 2
Max
Pool
Pool 5
6 x 6 x 256
=
9216
FC 6
4096
FC 7
4096
Softmax
1000



2010 → ML model → 28%

2011 → ML Model → 25%

2012 → Alex NET → 16.4%

2013 → ZFNET → 11.7%

2014 → VGG → 7.3%

2015 → GoogleNET → 6.7%

2016 → ResNET → 3.5%