Intro to Computer Vision

Grayscale

30x30x1

h x w

30 x 30 x 3

R G B

Colored image

$\square \times \square \times 3$

40   50

30 x 30 x 3   3D Tensor

2700 x 1   1D Vector

Flattening

3D Tensor → 1D Vector } Input

Shirt, T-shirt

Agenda

→ Simple NN/MLP are not helpful

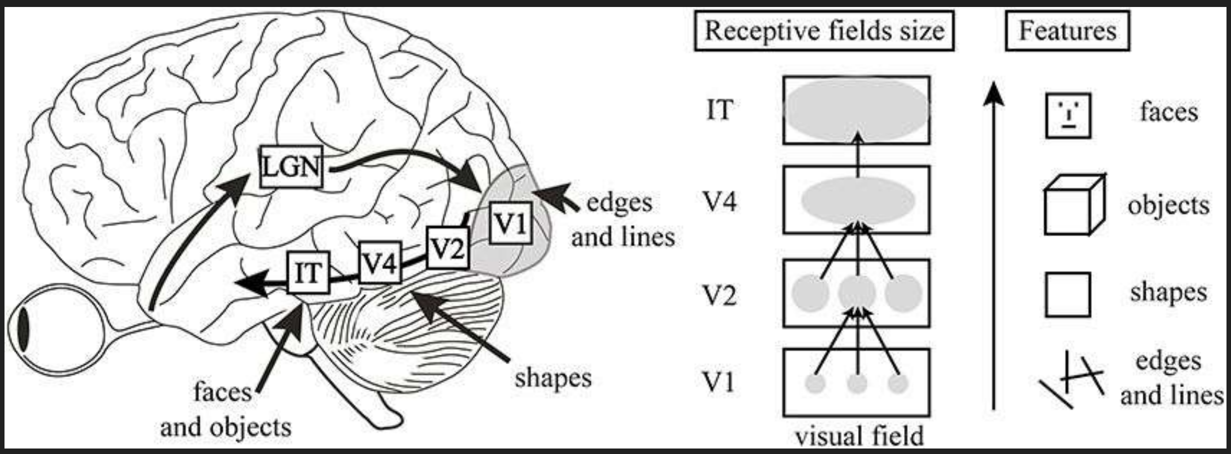
→ CNN
  ↳ Convolution
    ↳ kernel
    ↳ Padding
    ↳ Stride
  ↳ Max Pool

→ CNN can overfit
  ↳ GAP ⇐
  ↳ BN
  ↳ Dropout
  ↳ Reg → L1/L2

$30 \times 30 \times 3 \xrightarrow{\text{flatten}} \underbrace{2700}_{\text{features}}$

$2700\,F \longrightarrow$ C1, C2, C3

Why Deep Neural Nets for Images?



Paragraphs

Sentences

Words

Characters

Softmax

3x3x5

Human Car Bike

Conv. 3x3x10

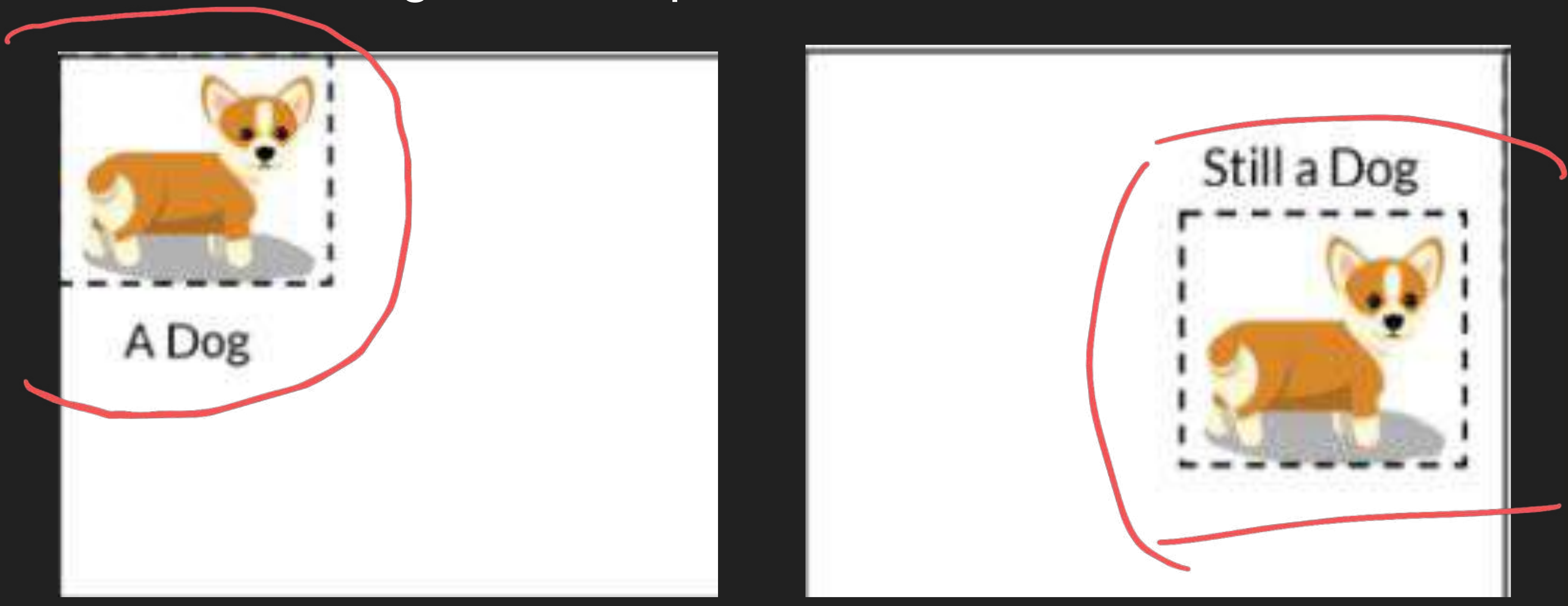
face of a child Leg head of a car

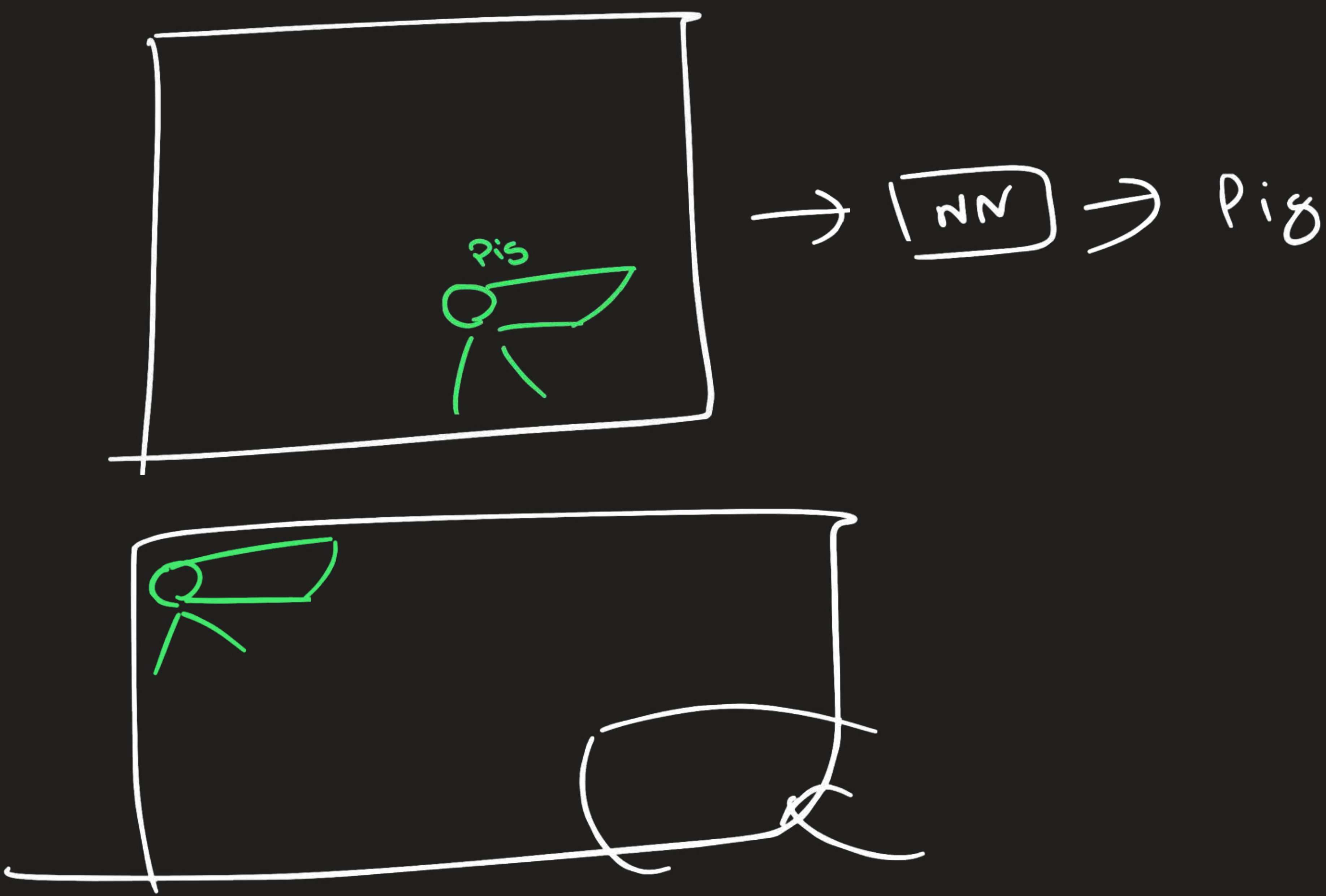
Conv 3x3x10

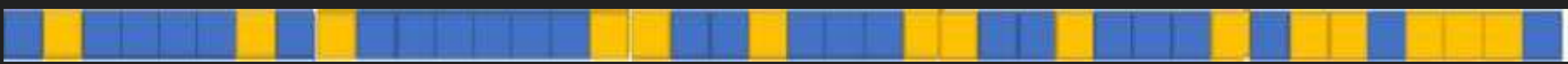
Edges and lines

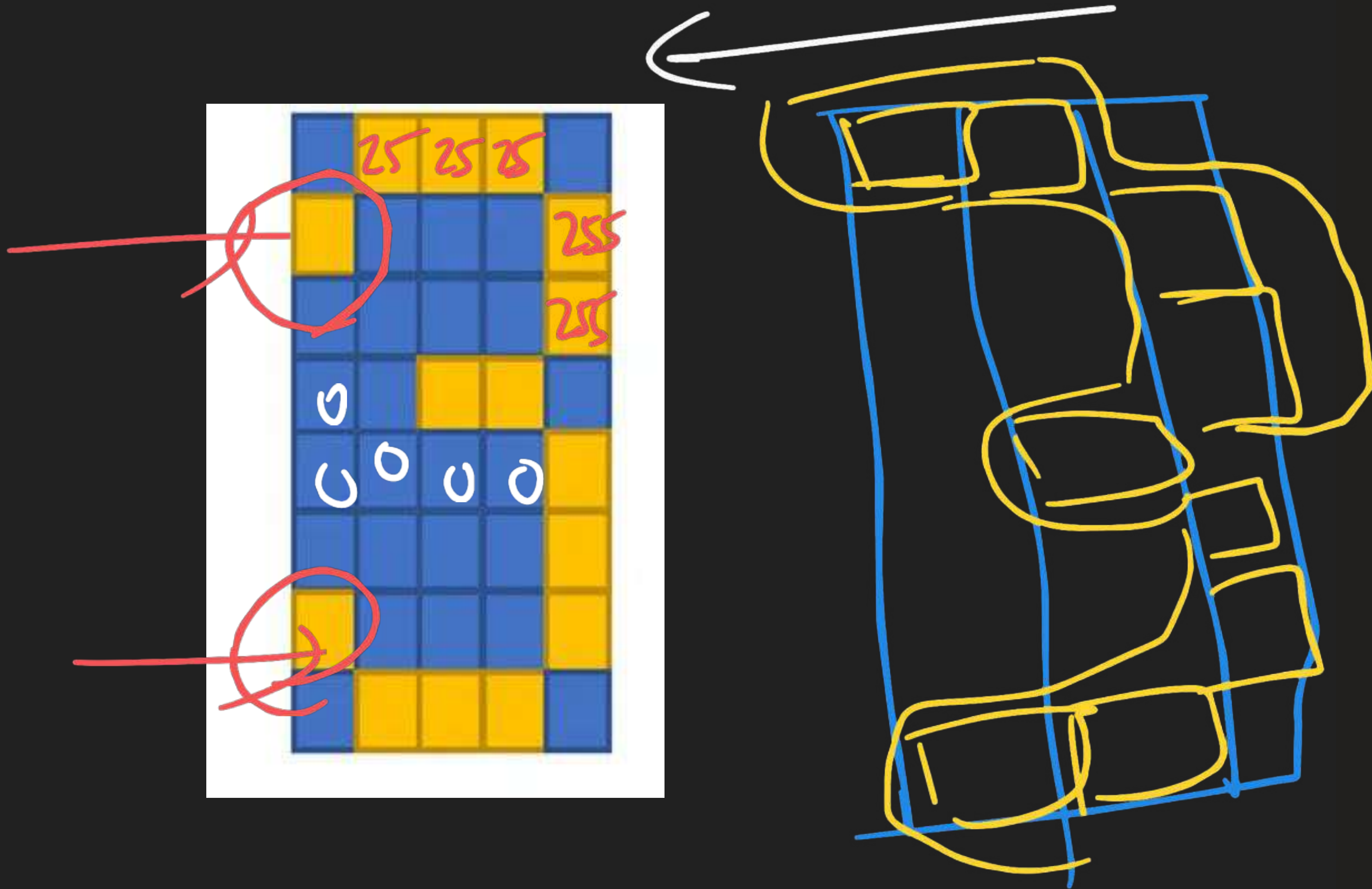
Not translation invariant

Issues of using MLP/Simple NN









✓

Overfit   Dollar ← large no. of parameters

Issues

→ Spatial info completely lost (No translation invariance)

→ Lots of parameters

deep

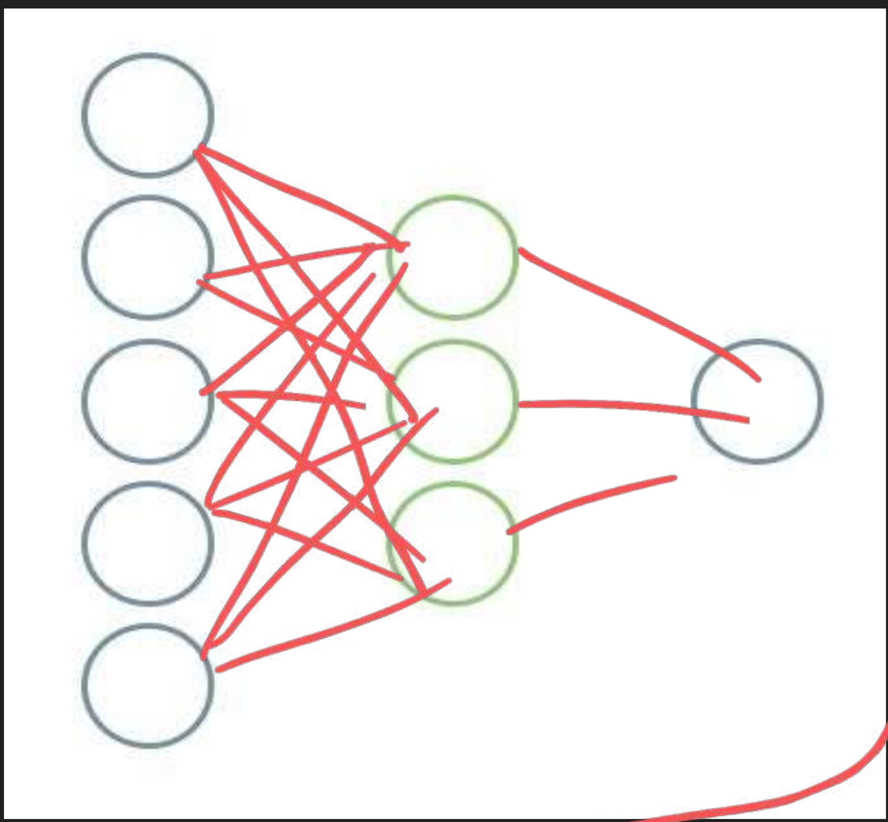
Sparse connection

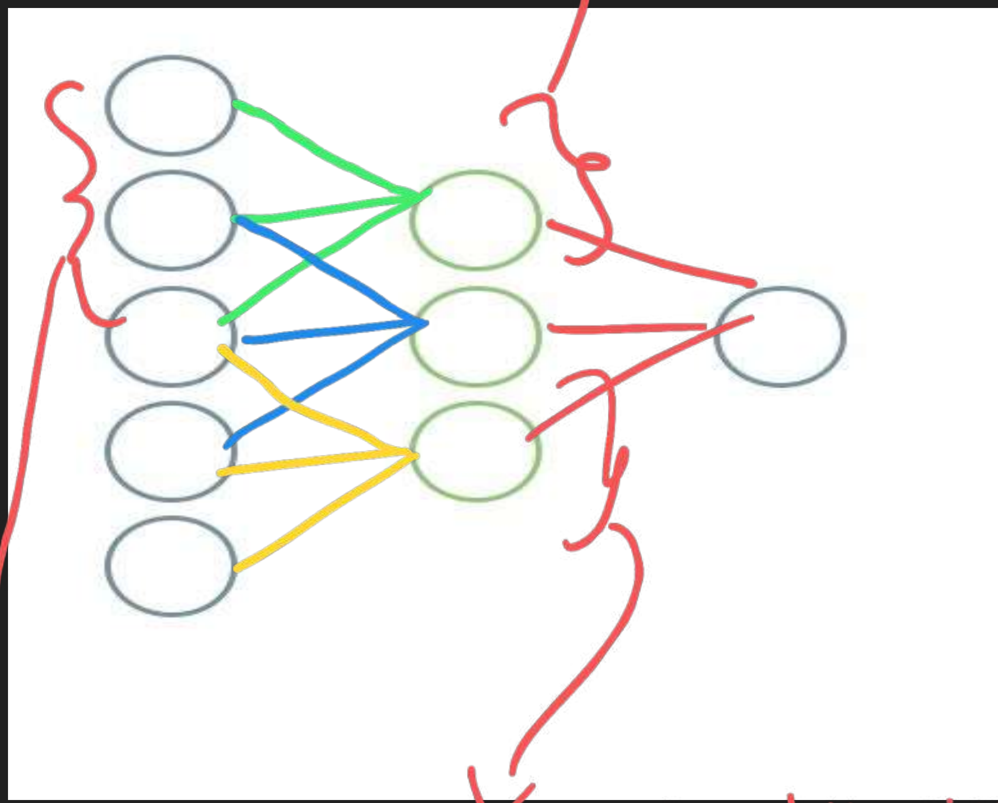
sparse

Improving MLPs

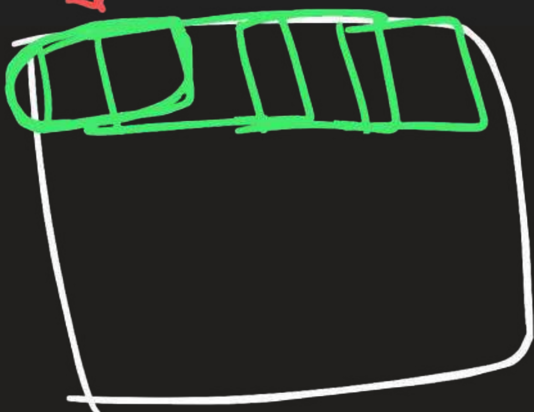
← Enforce locality

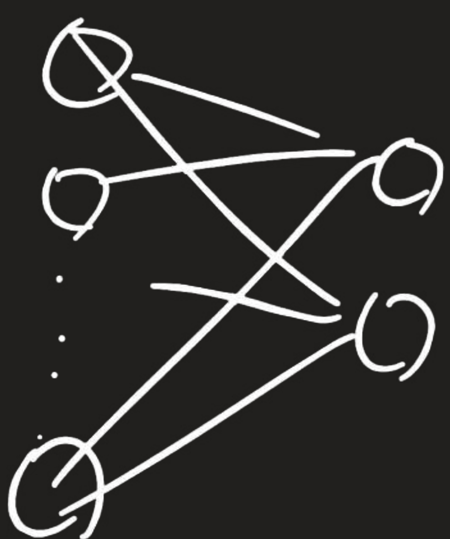
find correlations of brown pixels





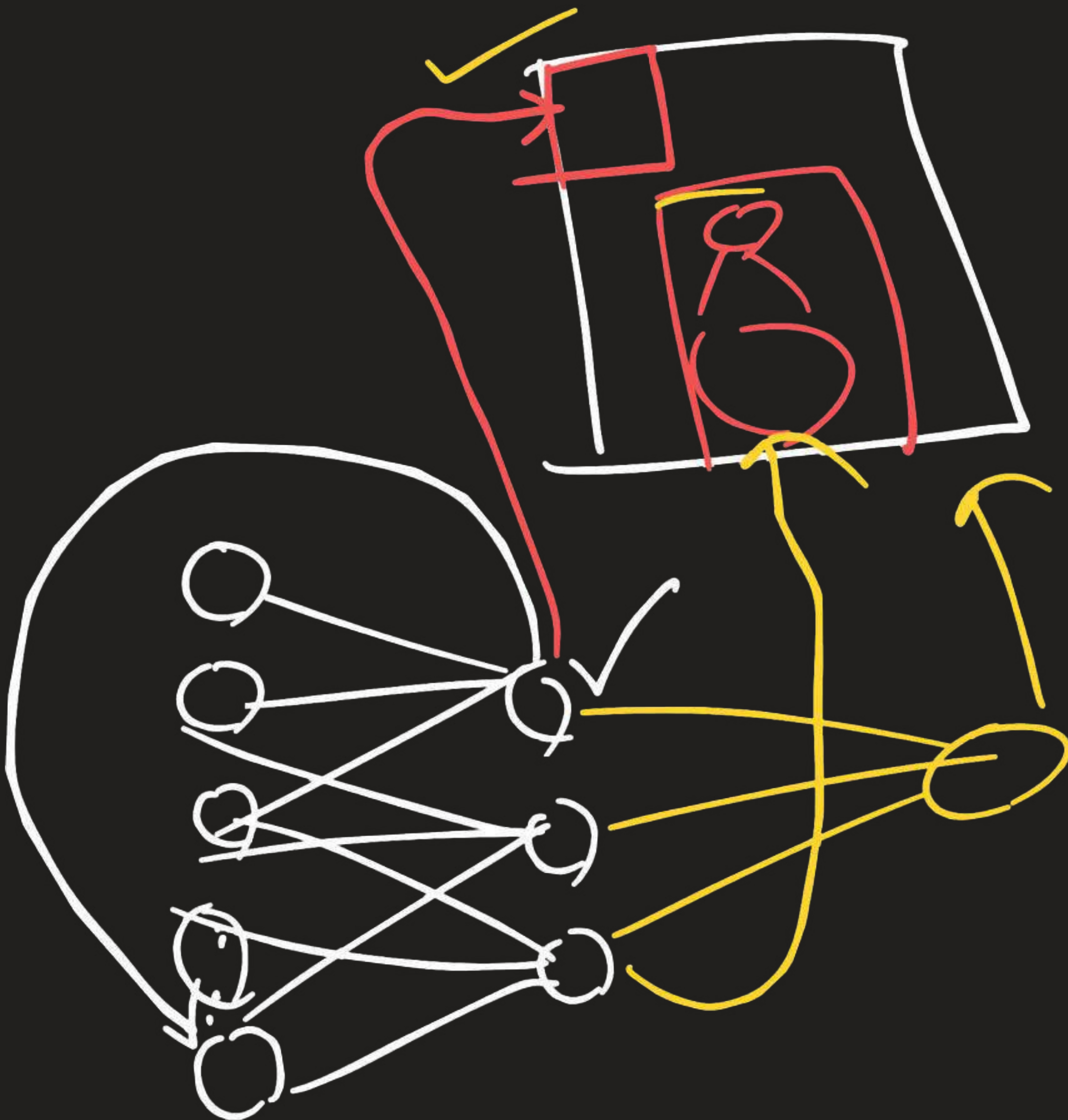
corelation of white pixels





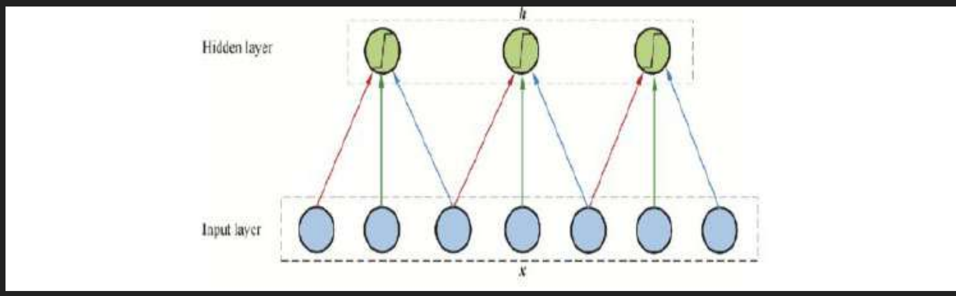
FE

BE

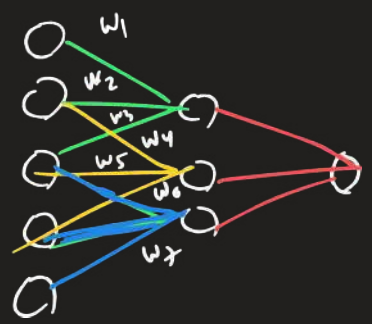


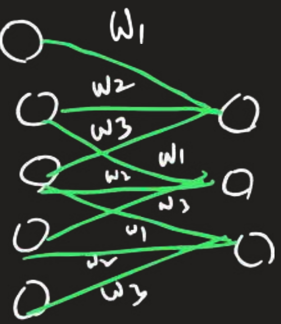
far away, pixels are not correlated

Enforcing stationarity

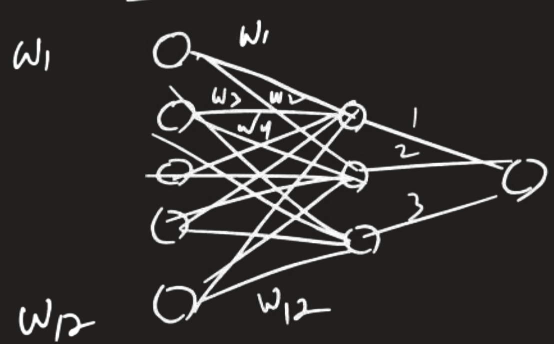


Stationarity

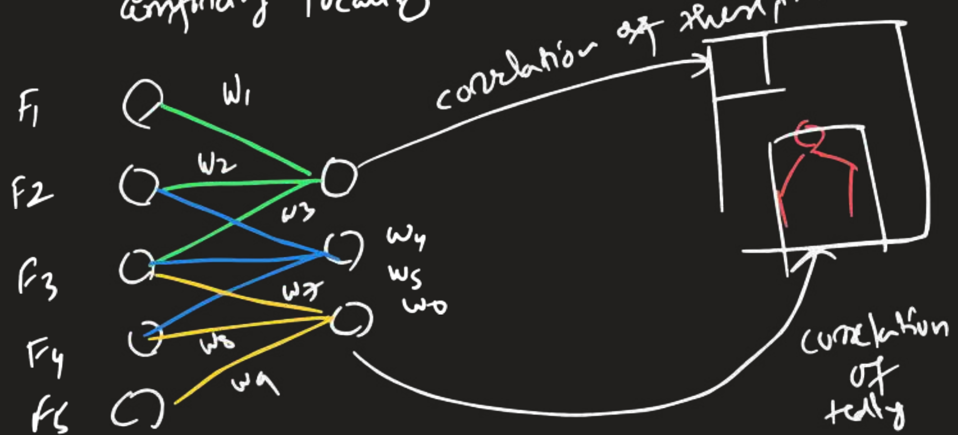




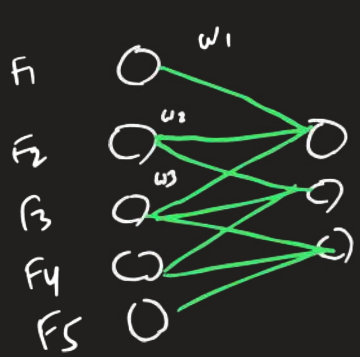
FF/MLP



Enforcing locality

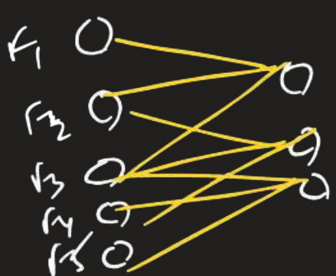


Enforcing stationar

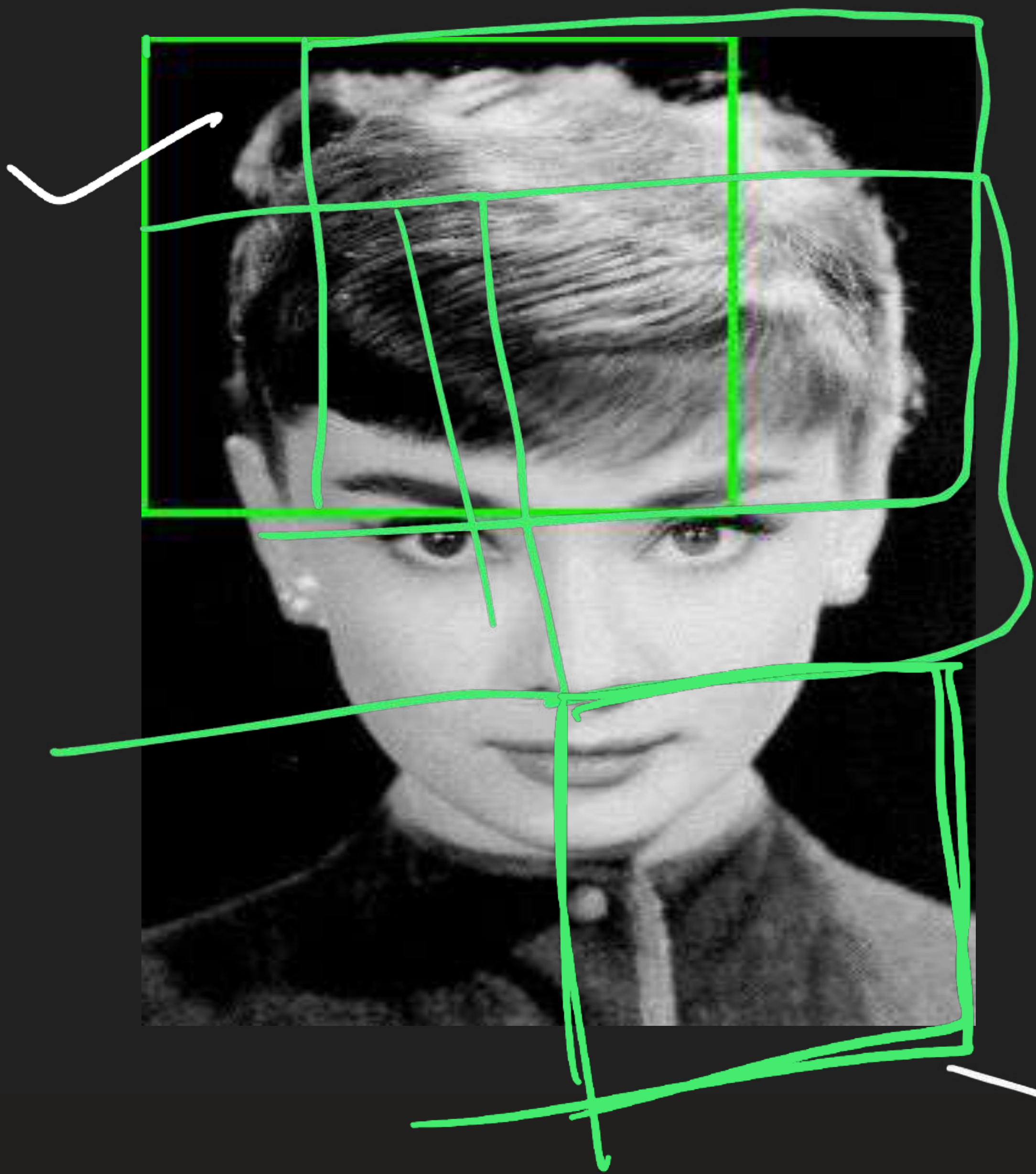


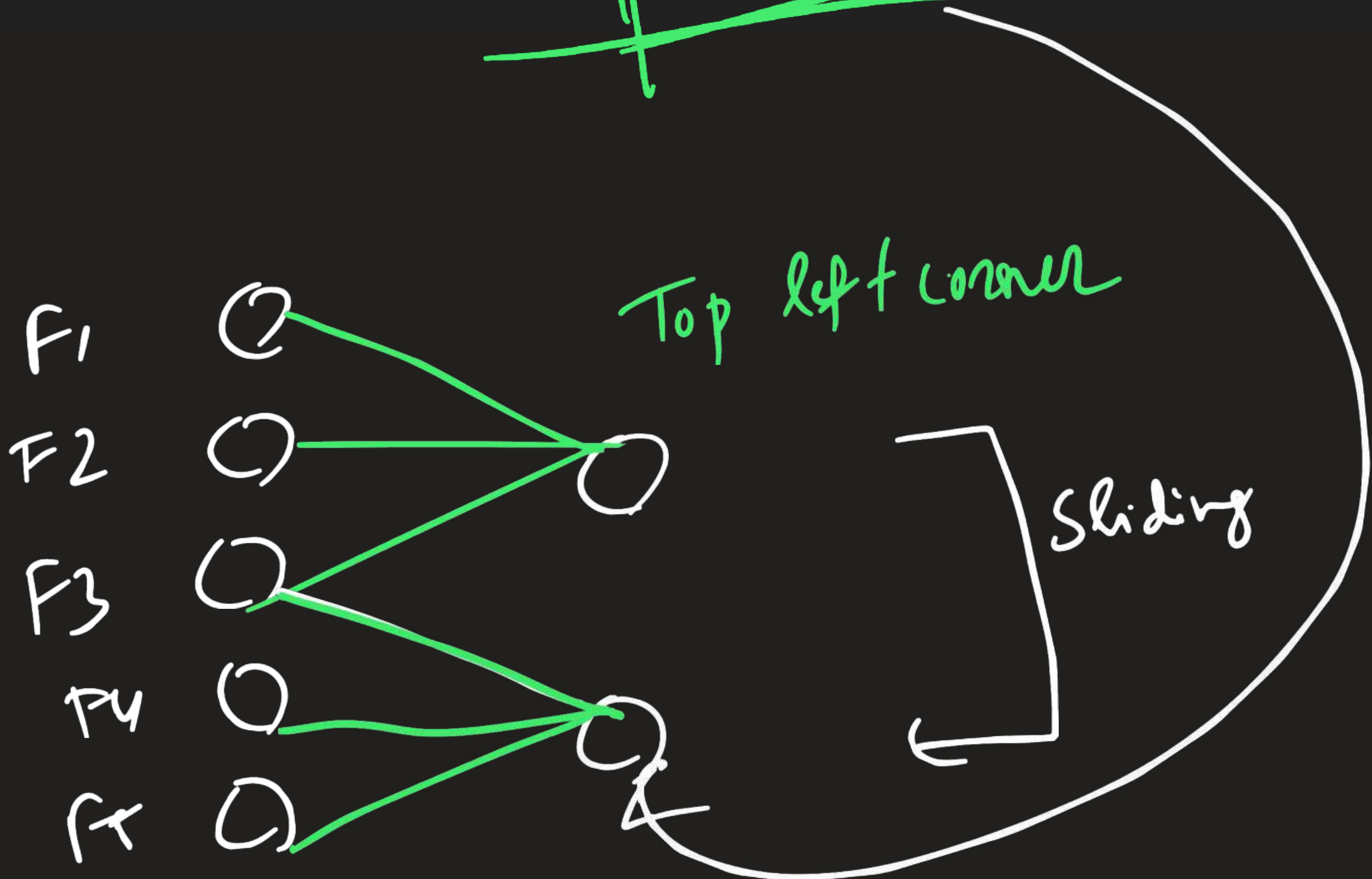
$$\begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix}$$

$$\begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix}$$



$$\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}$$





Grayscale

Convolutions

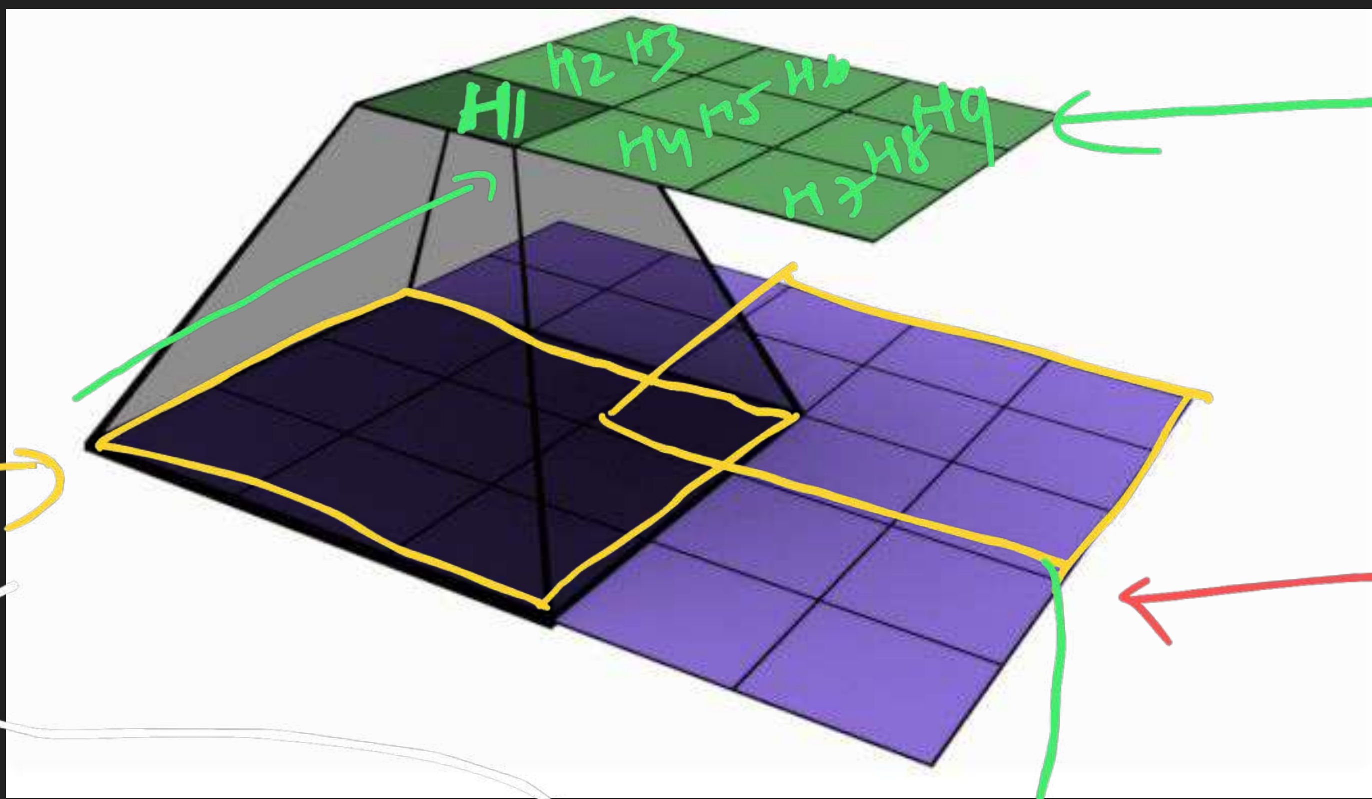
Feature map

Hidden neurons

3x3

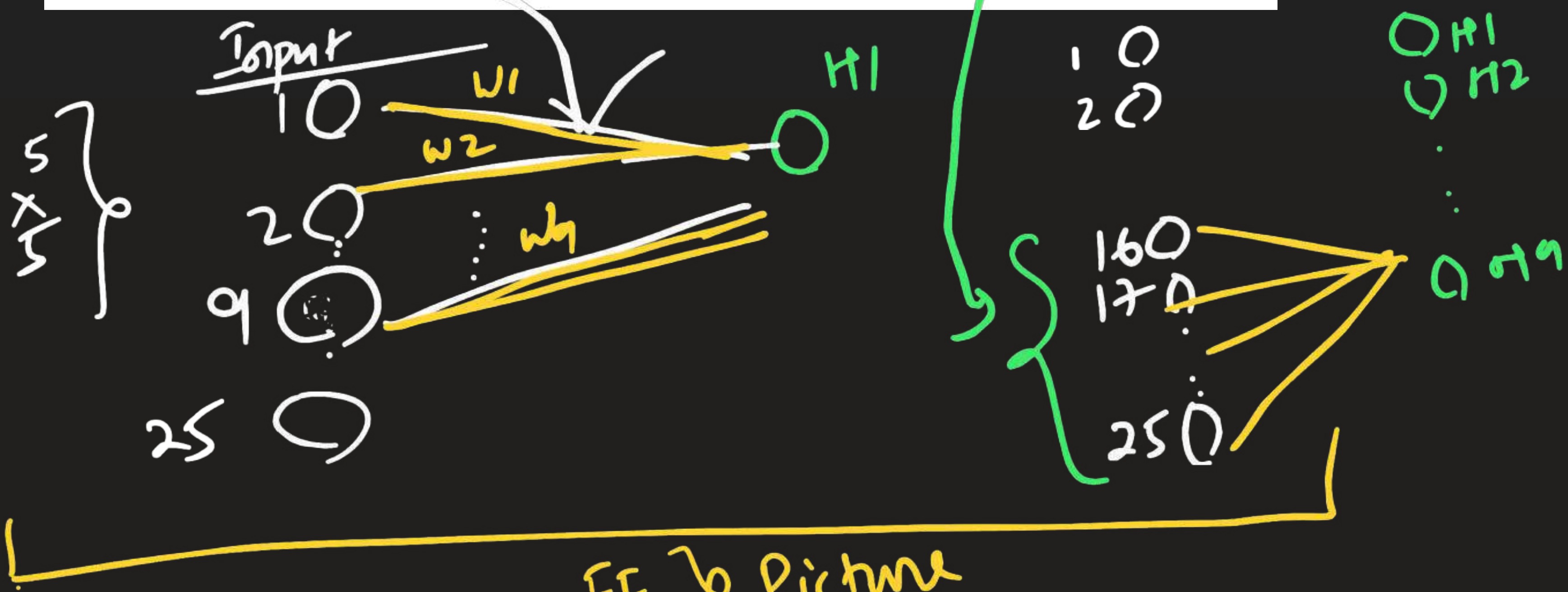
Wt matrix

Kernel



5x5 (x3)

Input

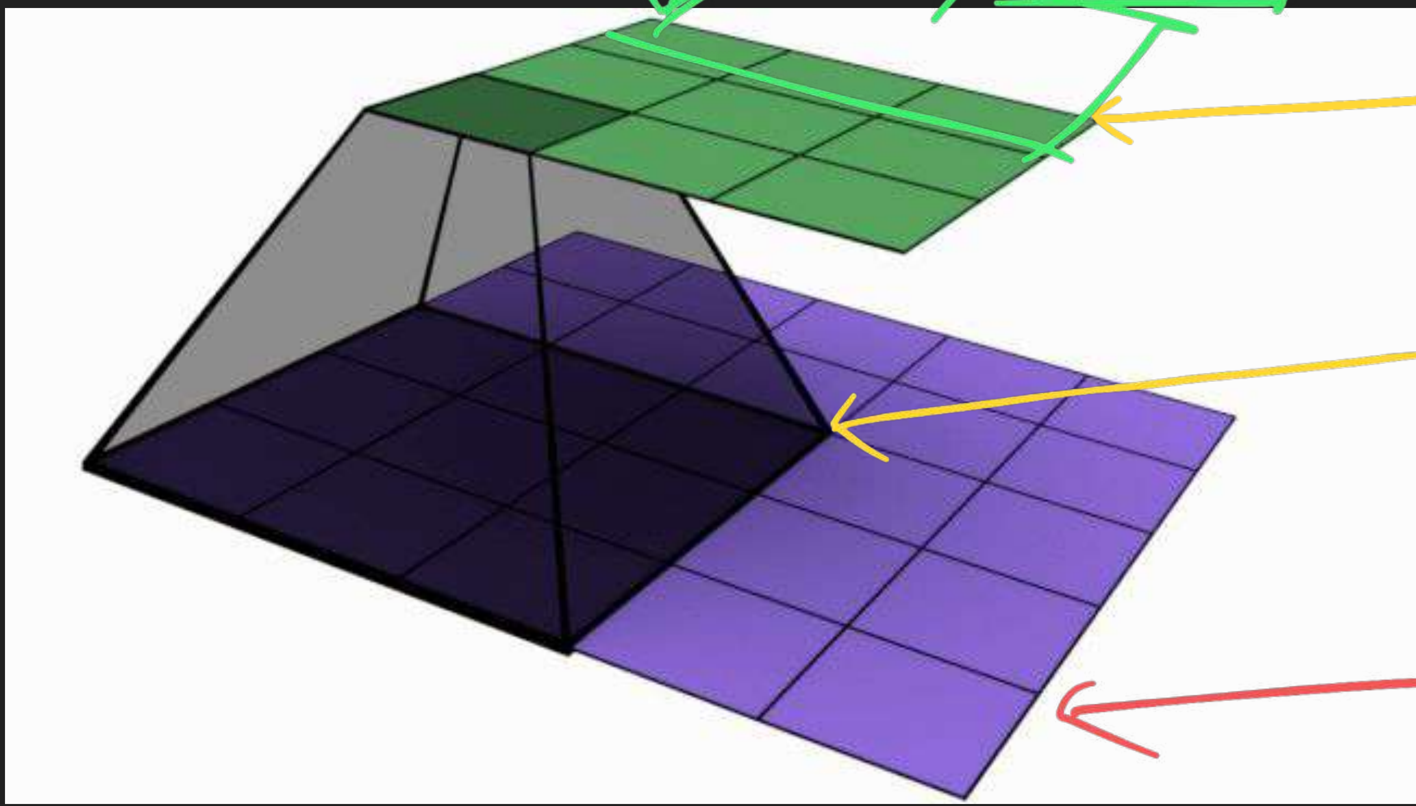


FF } Picture

Convolutions

10:10

Feature Map



3x3

IP
5x5
Fixed

?
Multiple

IP	Kernel	Feature Map
5x5	3x3x10	3x3 x 10

fit this model

K K K K

I/P

5x5x3

K

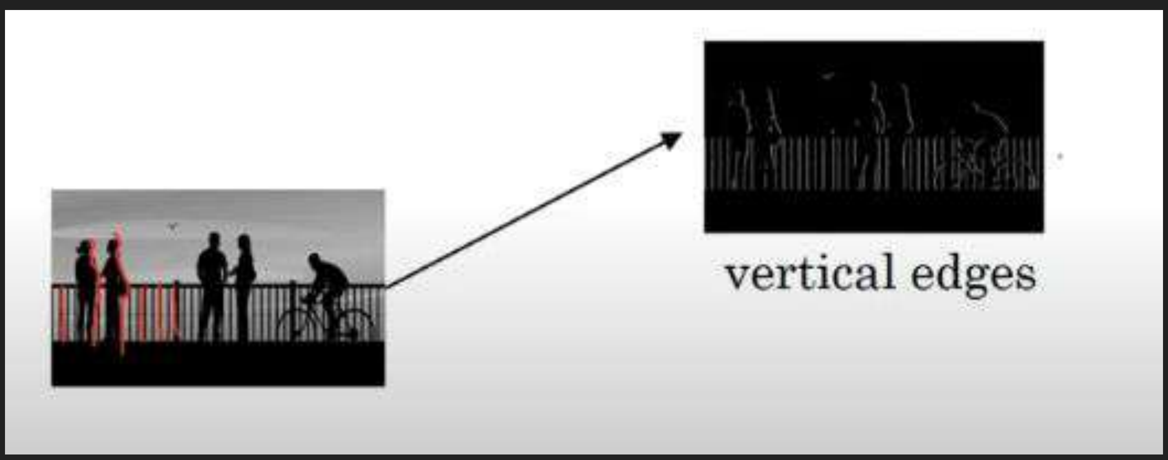
3x3x3

1 kernel

→

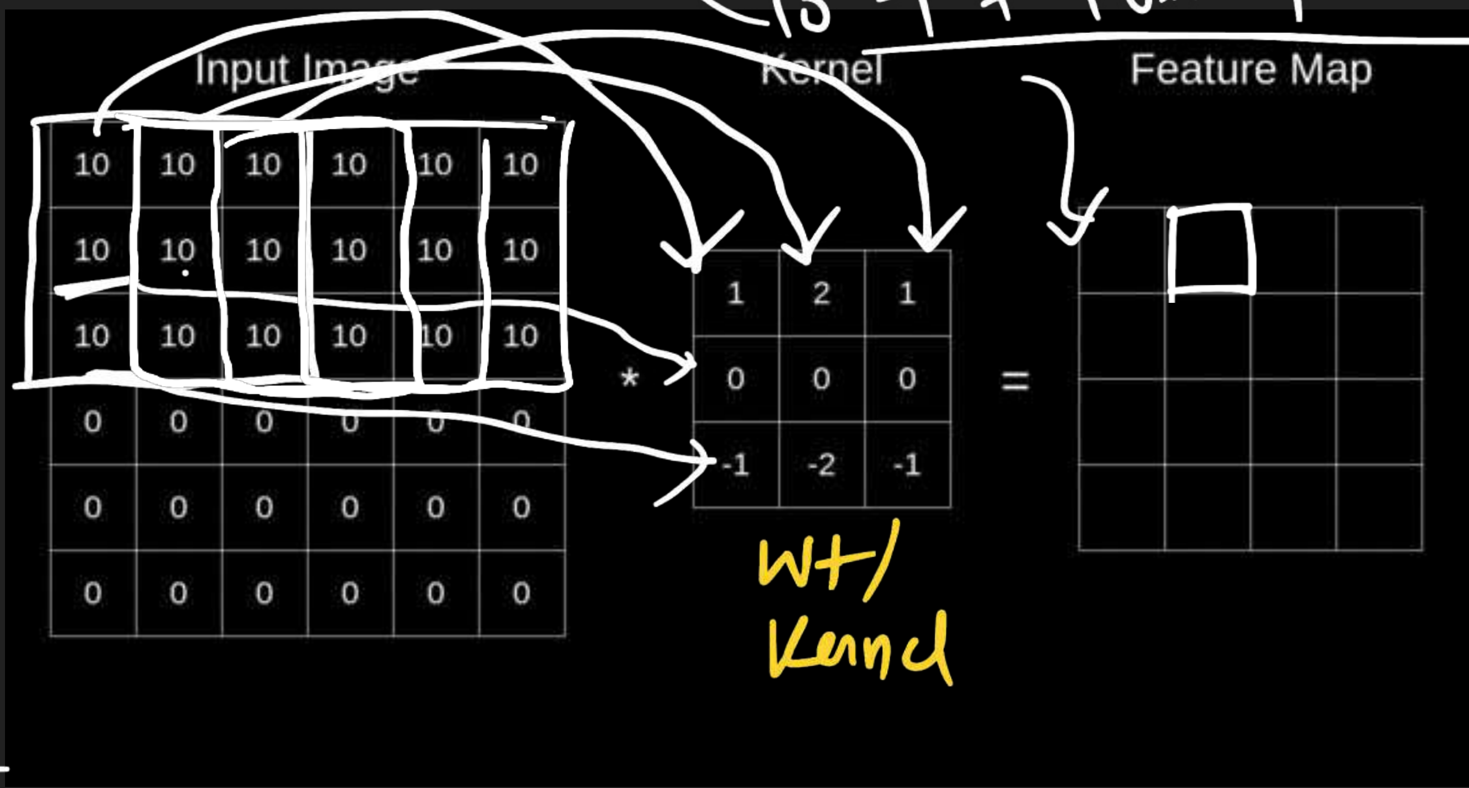
3x3

Edge Detector



Convolution Operation-2D

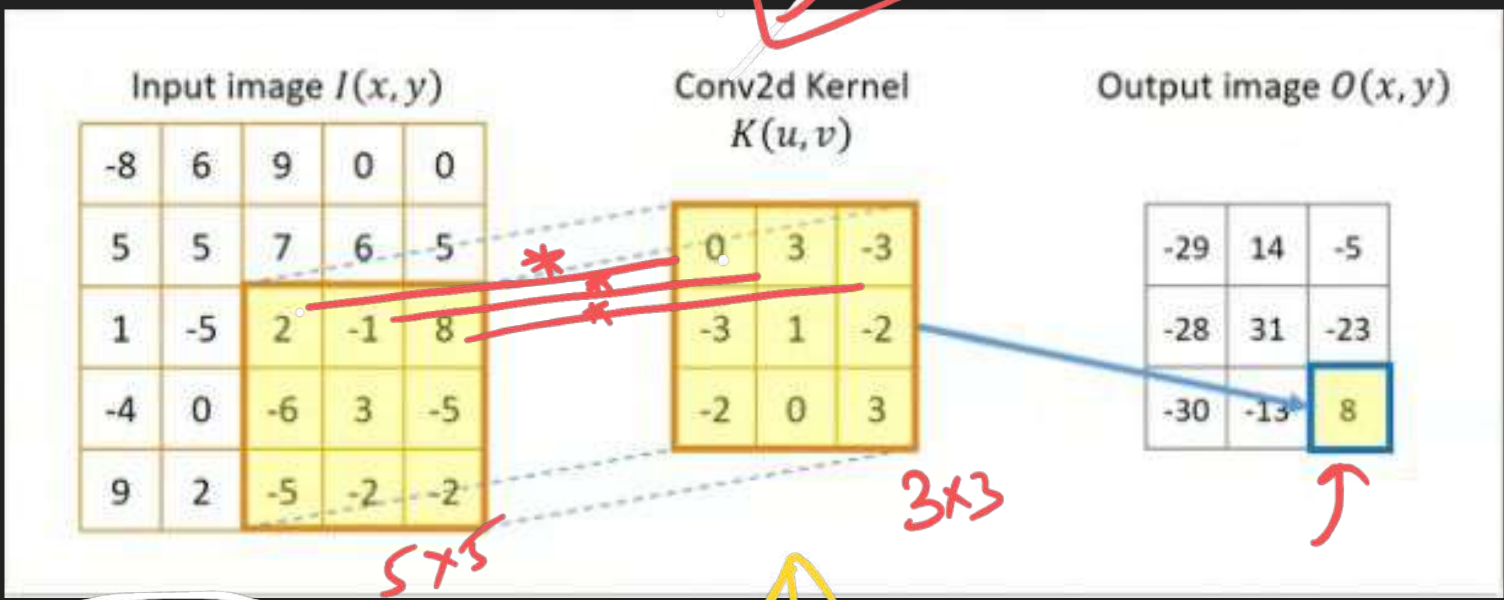
$$10*1 + 10*2 + 10*1$$
$$10*0 + 10*0 + 10*0$$
$$10*-1 + 10*-2 + 10*-1$$



1	2	1
0	0	0
-1	-2	-1

2dConv

$w^* x + b$



Input
5x5
fixed ✓

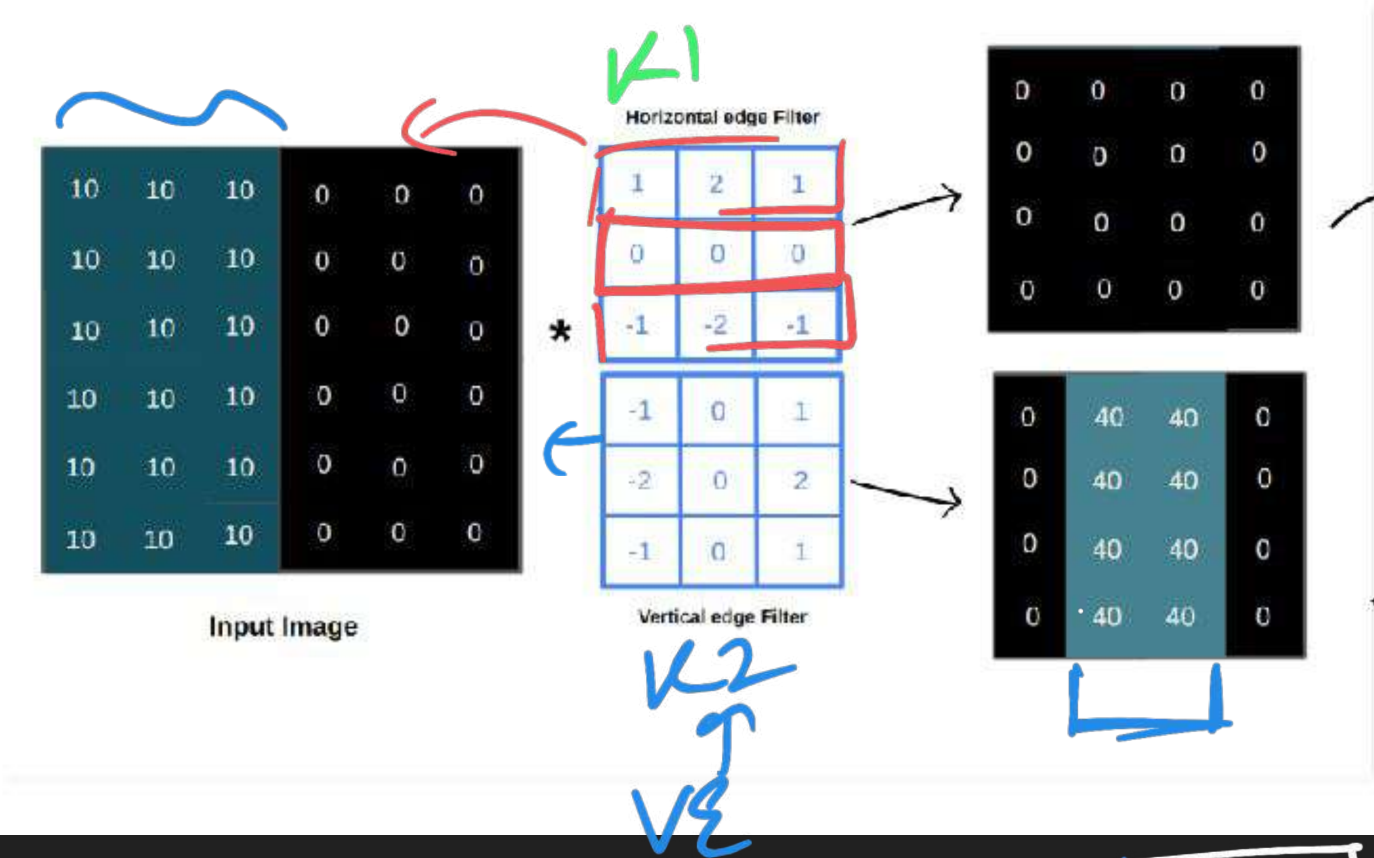
[1 2 3] [4 5 6]

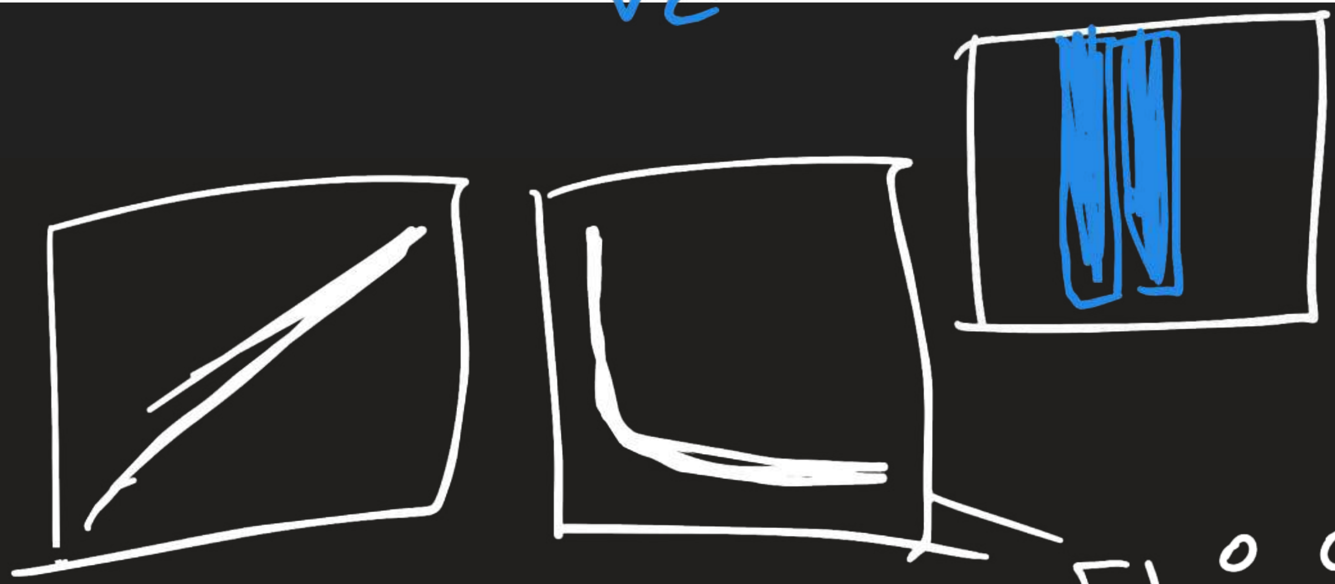
Output
Shift ✓ fixed

OW1 OPW2

✓ Backprop → [~~RW1~~ ~~RW2~~ RW3 ... RW9]

3x3 ✓

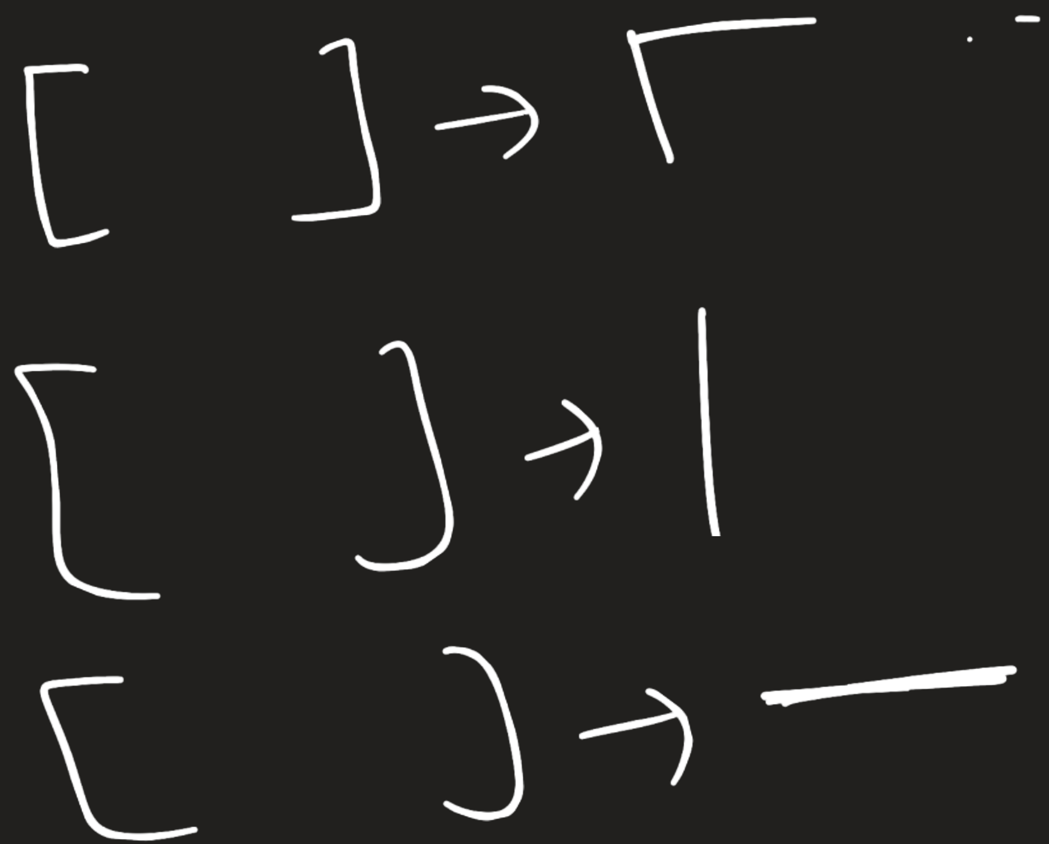




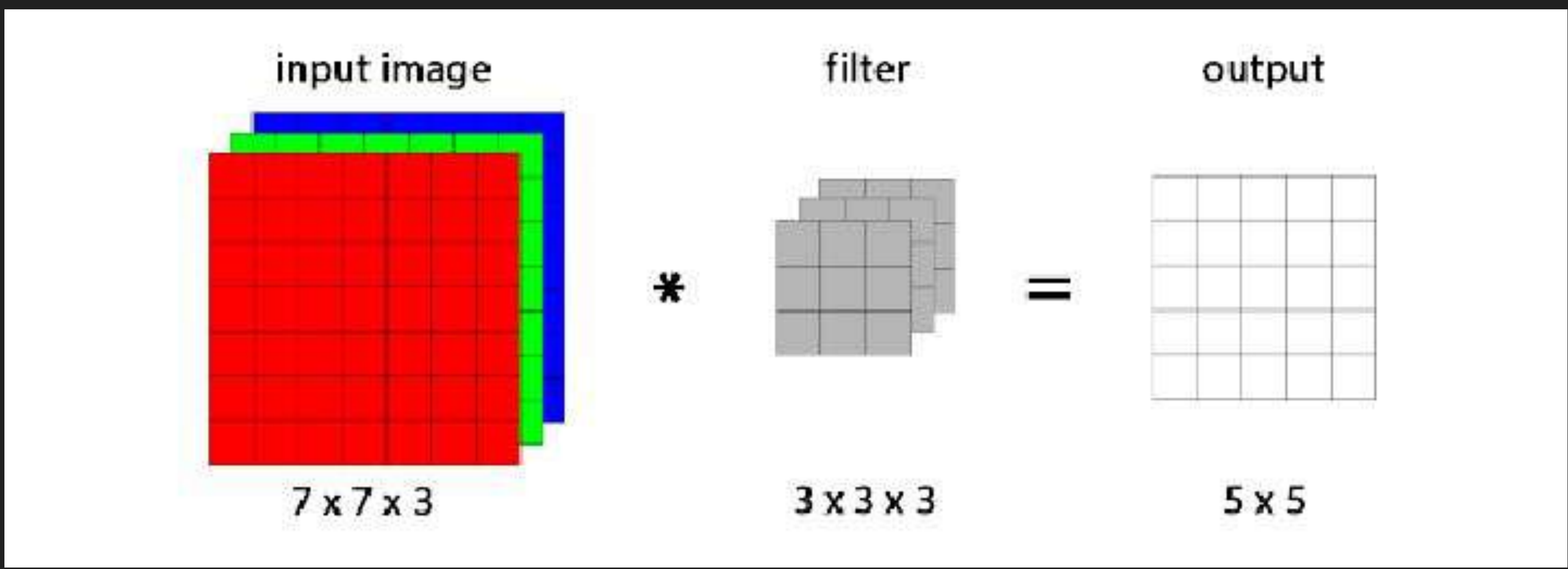
$$\begin{bmatrix} 0 & 0 & 3 \\ 1 & 2 & 0 \\ 1 & -2 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 3 & 4 & 5 \end{bmatrix}$$

1990s



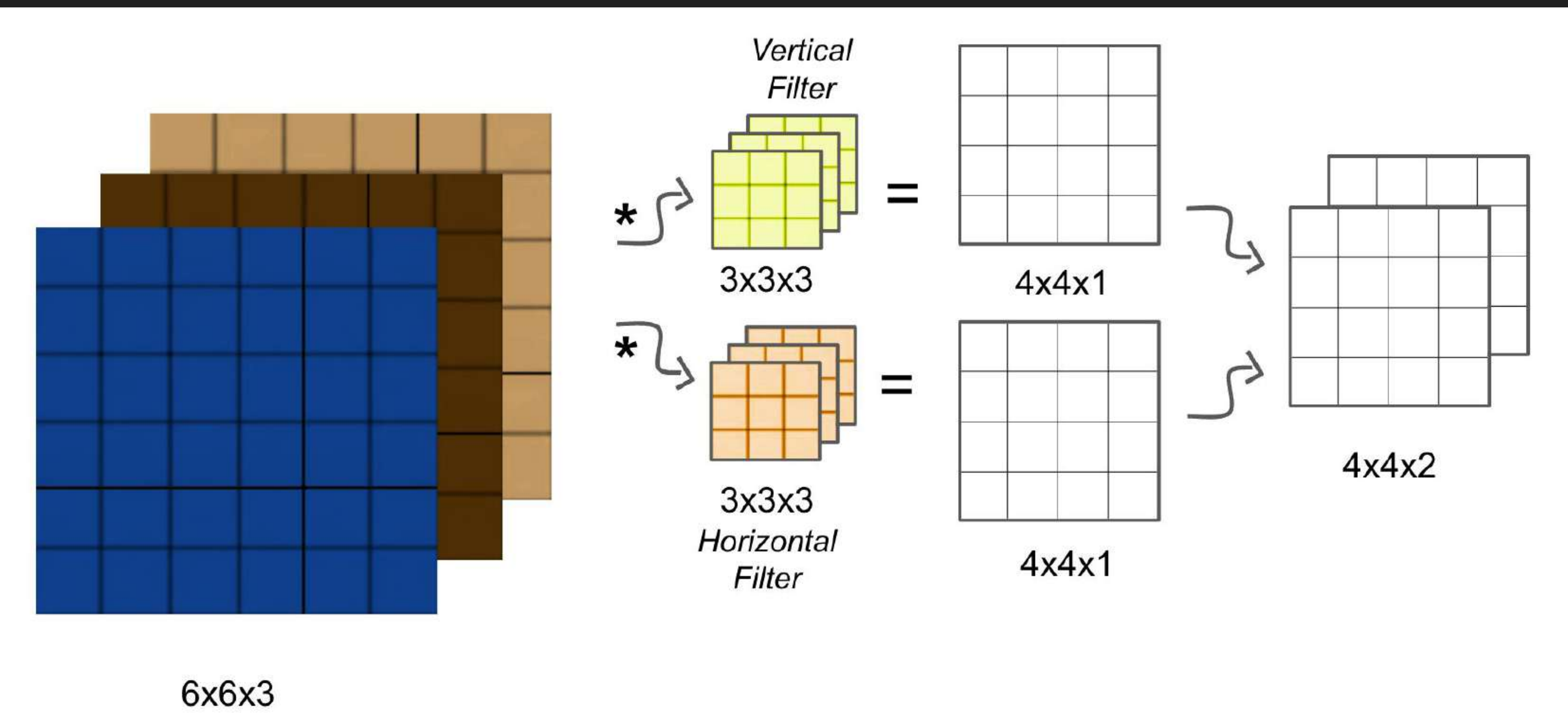
1 kernel



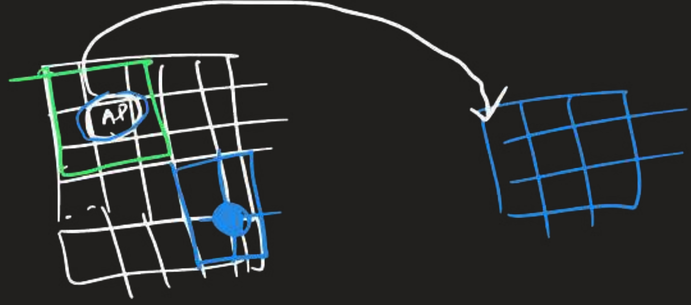
5x5x3 → RGB

3x3x3

3x3
FM



Vertical Filter
3x3x3
4x4x1
3x3x3
Horizontal Filter
4x4x1
4x4x2
6x6x3



5x5x3

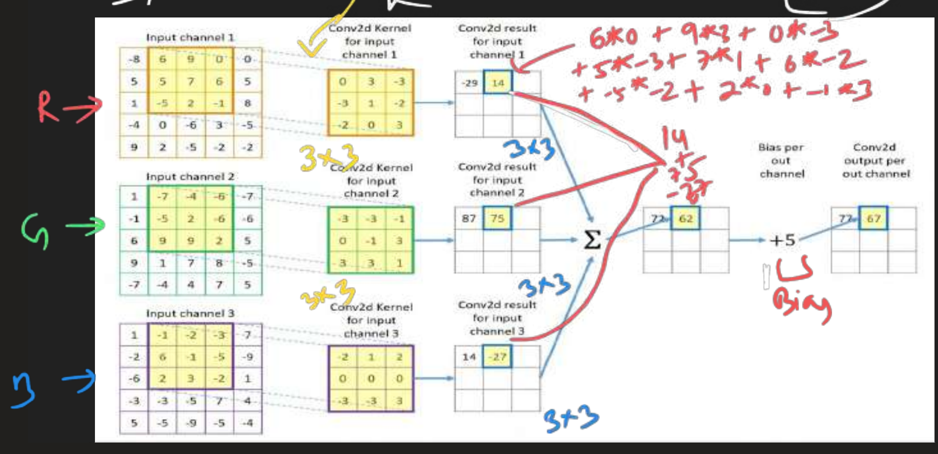
I/P

single kernel

3x3x3 ← each channel of input

K

3x3



5x5x3 $\xrightarrow{\text{Conv}}$ 3x3x3 → 3x3

IM K FM

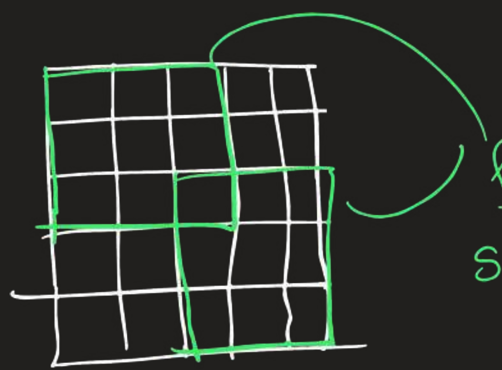
1 ← ve

5x5x3 $\xrightarrow{\text{Conv}}$ 3x3x3 x 10 → 3x3x 10

Single Kern

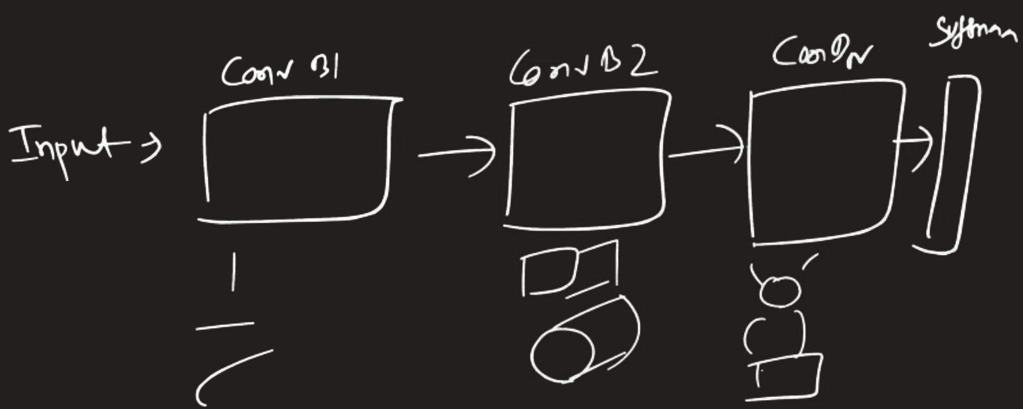
10 such 3d kernels

no. of channels in FM



Parallelized

SIMD ← Single instruction multiple data



5x5x3 → 3x3x3 x10 → 3x3x10

Random

Optimal kernel wt

10 feature maps

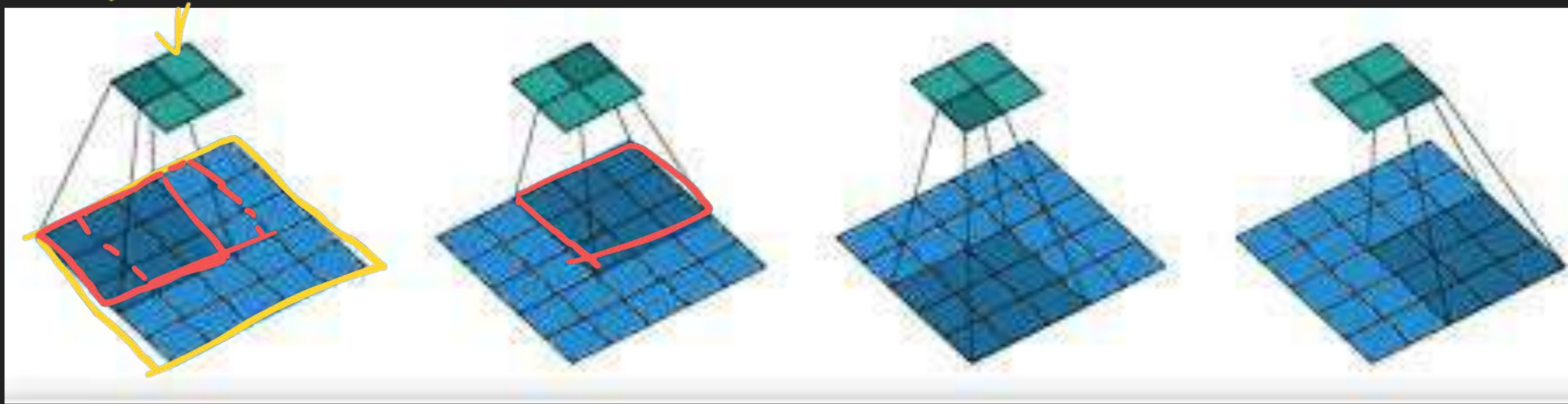
T1
T2
T3

3x3

Strides

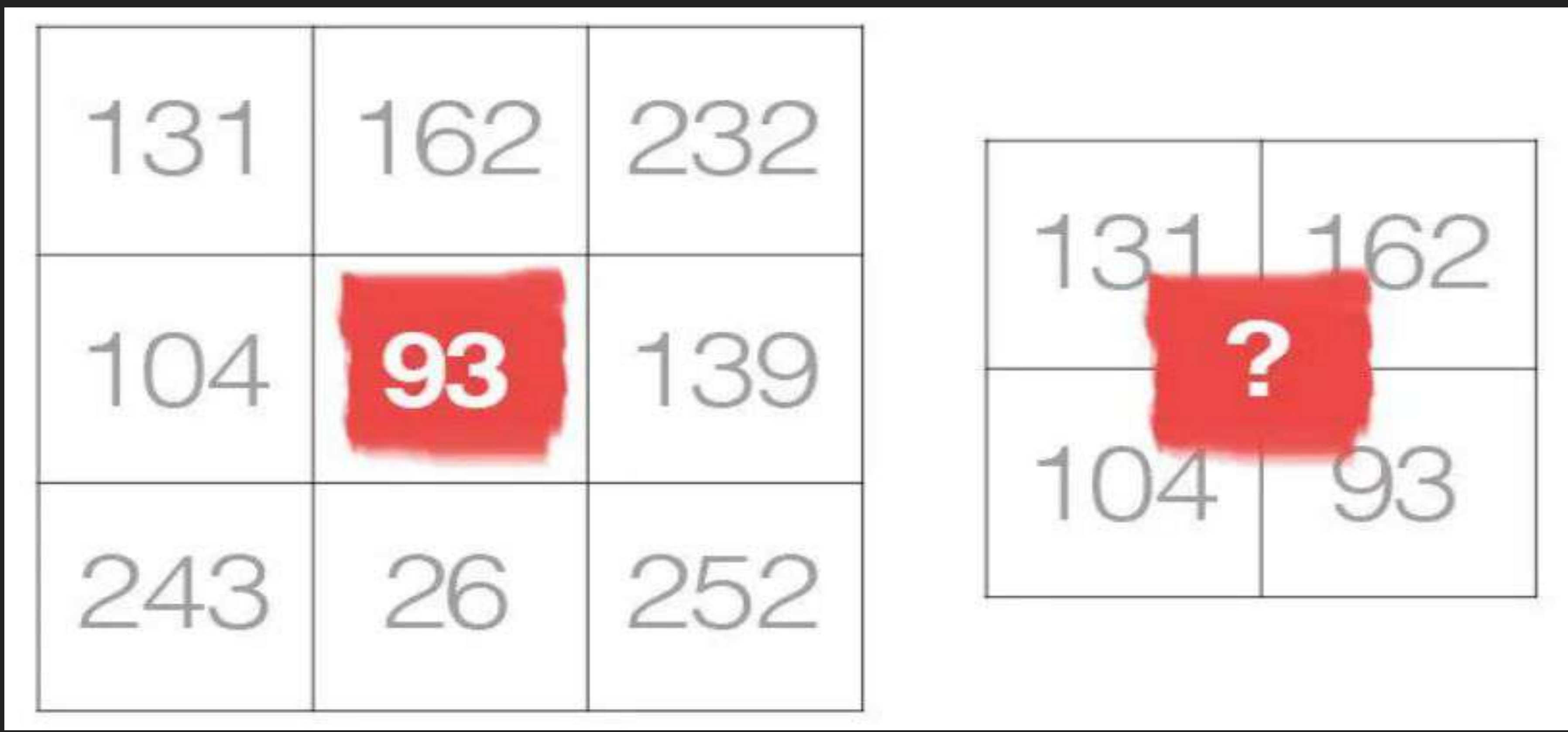
Strid = 2

2x2



We never consider even kernel

3×3 5×5 7×7



Stride = 1 = 1020p

stride = 2 = 720p

stride = 3 = 240p

0

are no longer being misrepresented

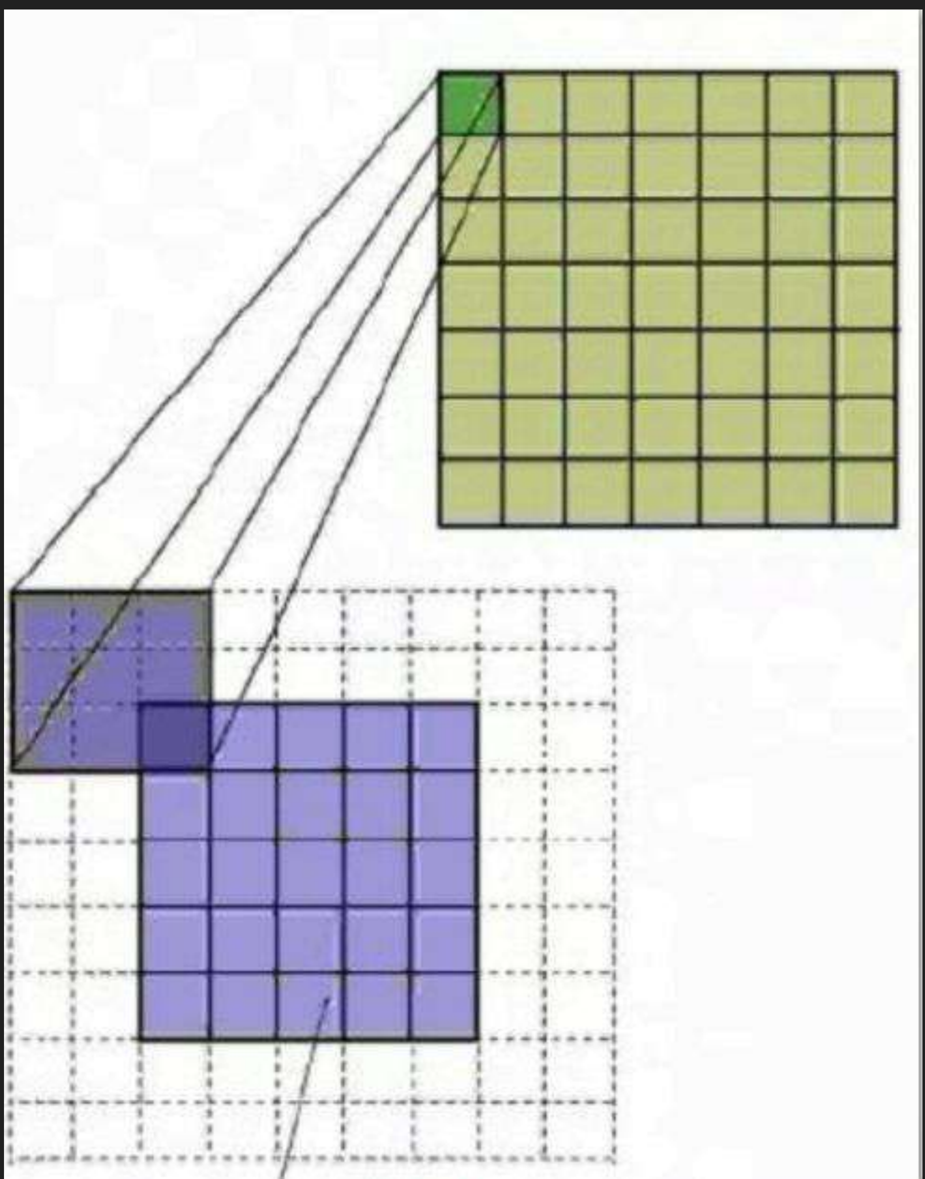
	0	0	0	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0

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

no chance afterwards

Represented

Convolving

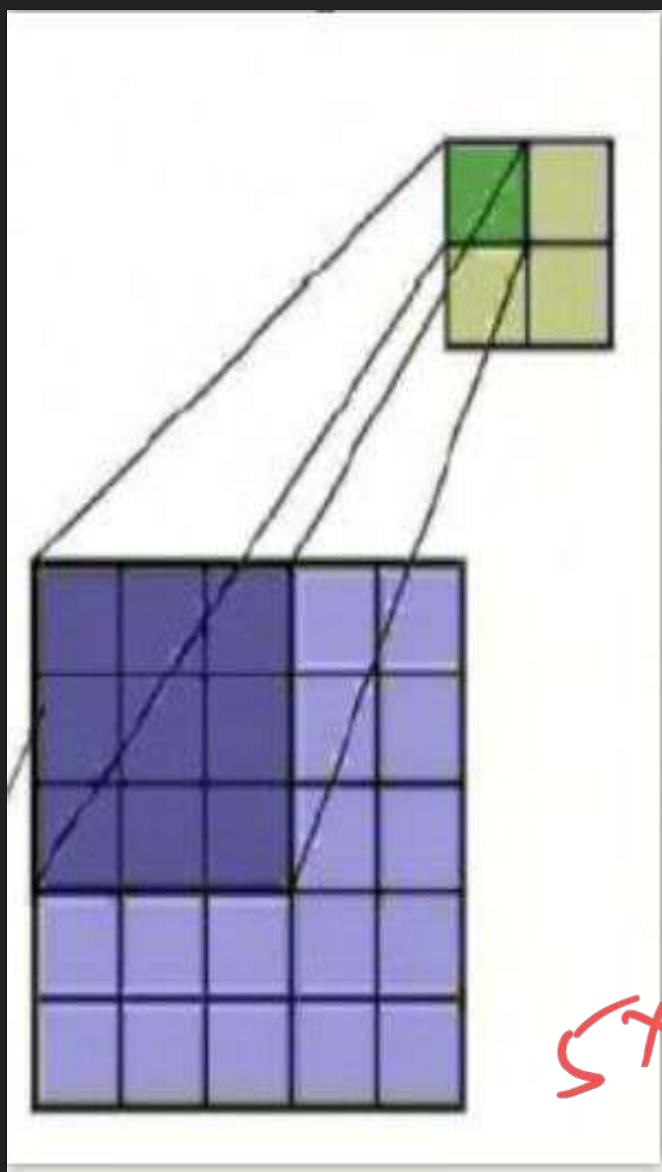


Full Padding

Output FM size > original input

Valid Padding

no padding

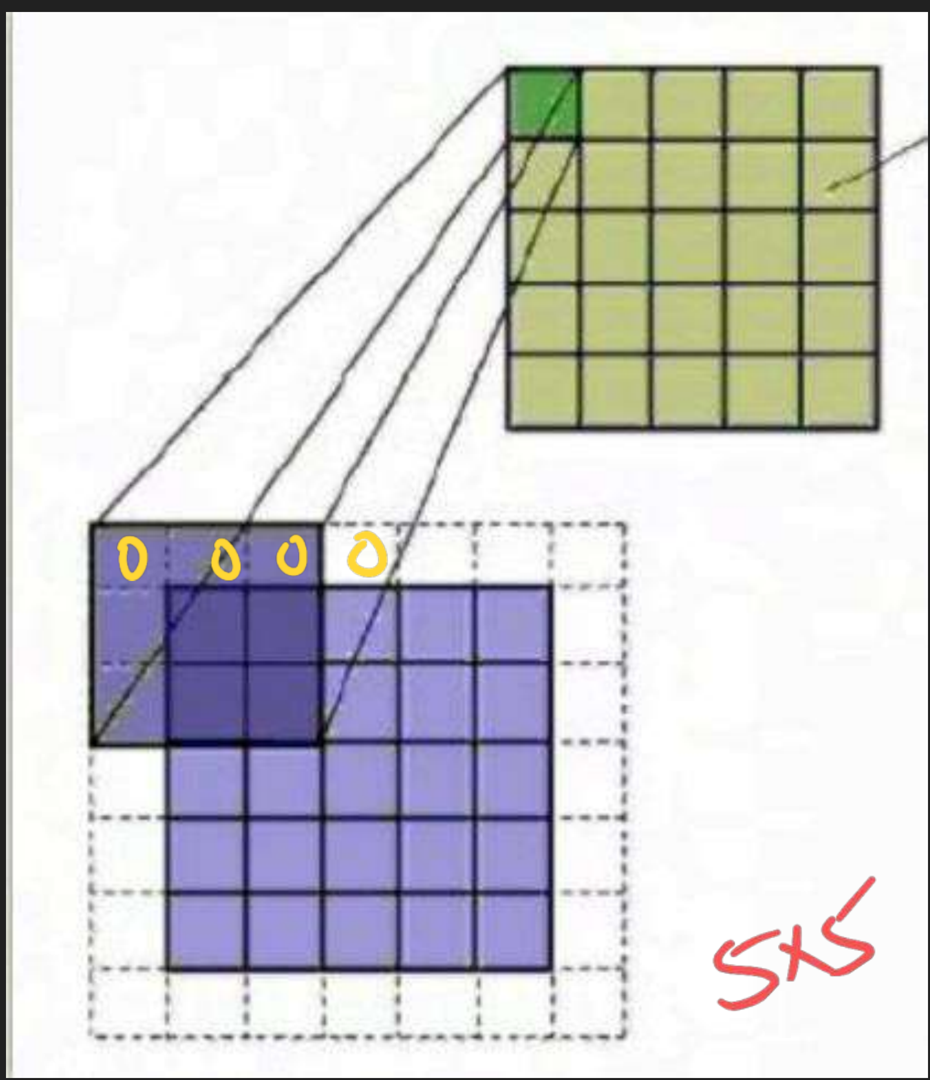


FM < IS

2x2

5x5

Same padding



IS = FMS

input
5x5x3

Kernel
3x3x3

Padding
valid = 0

5 - 3

0

stride = 1

$$output = \frac{input - kernel\_size + 2 * padding}{stride} + 1$$

$\frac{2\ \cancel{5-3}}{1} + 1 = 3$

5x5

~~6x6~~

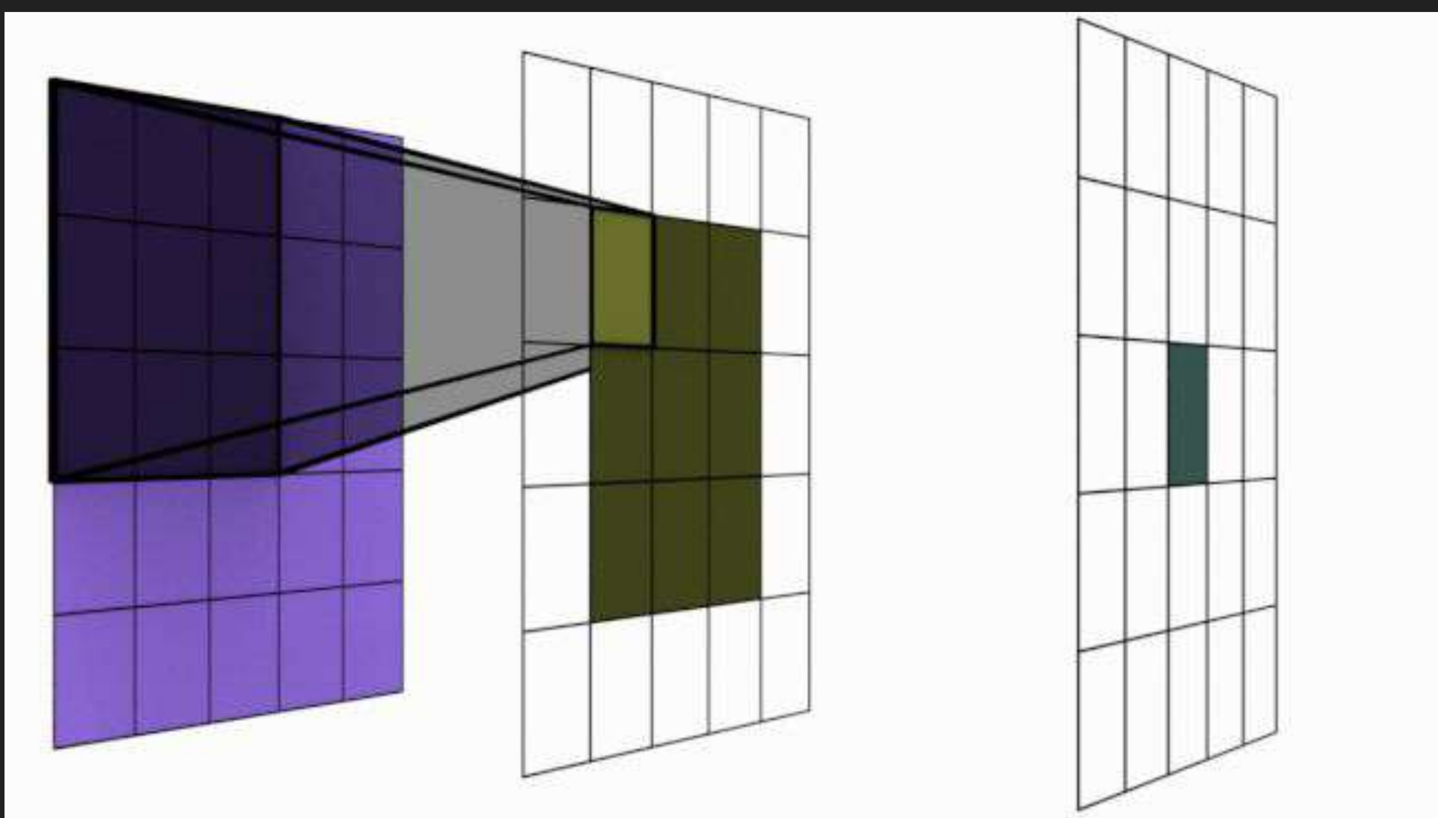
5x5x3 → 3x3x3 → 3x3

Padding = 1

$\frac{I(5) - K(3) + 2 * 1}{1} + 1 = 5$

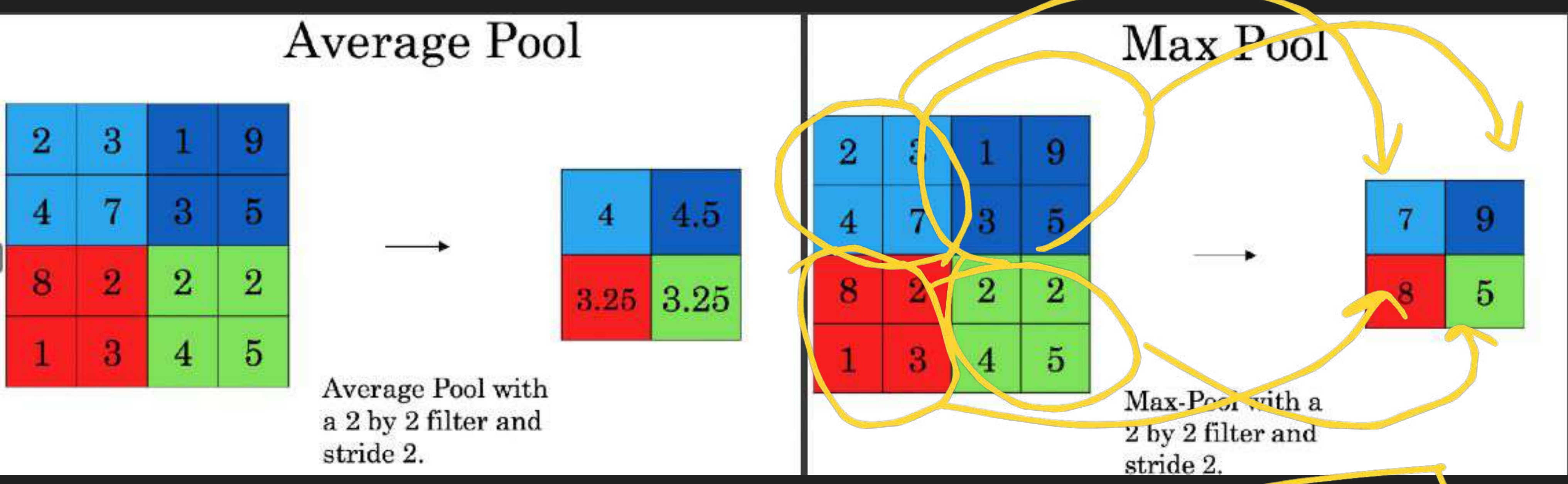
5x5x3 → 3x3x3 → 5x5

~~Vali~~ Same padding

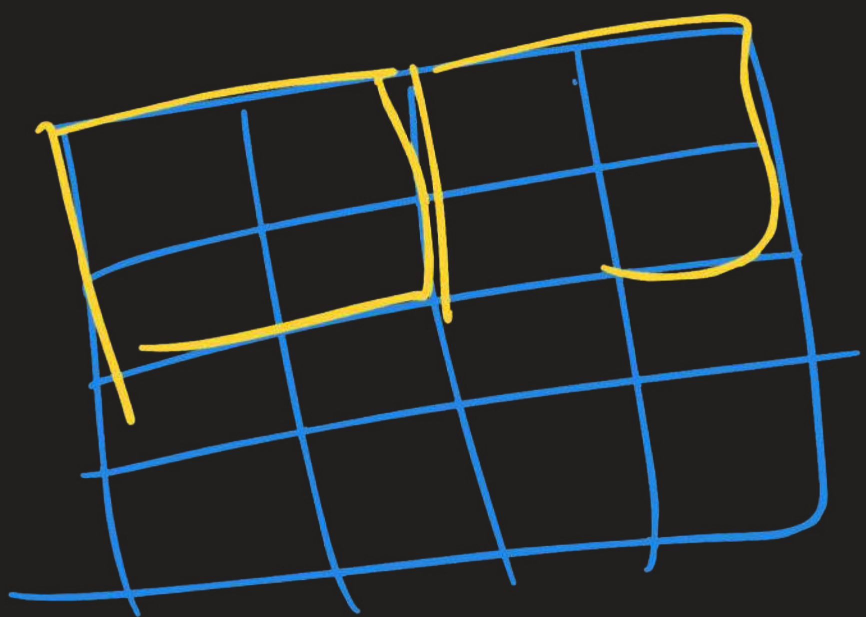


non parametric

(2x2)



Conv2d → Maxpool →



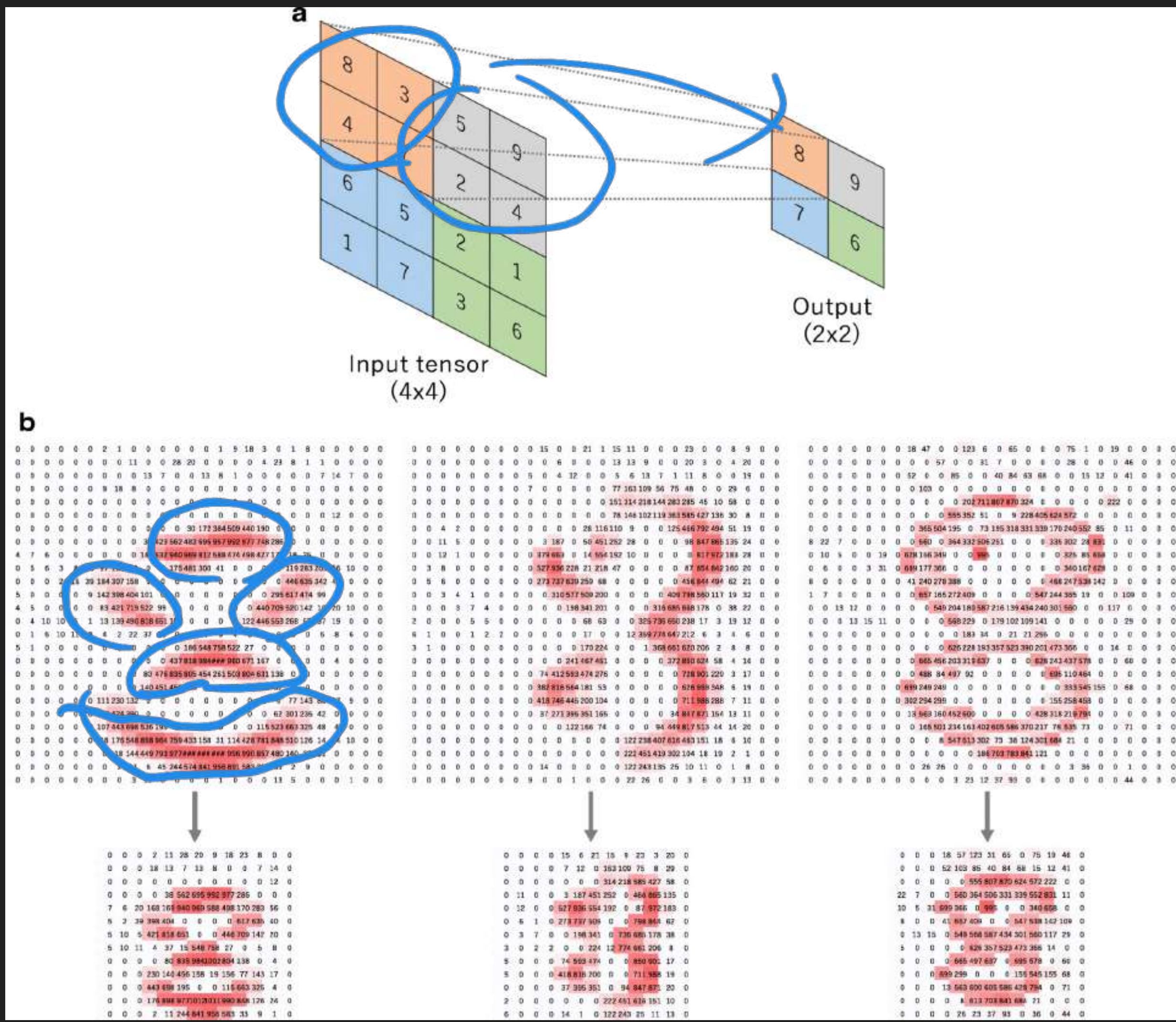
Conv2d + Maxpool

is not used towards last layers

6x6x3 → 3x3x3 → 4x4

avg pooling

2x2



2x2

Strid=2

By default

→ Conv

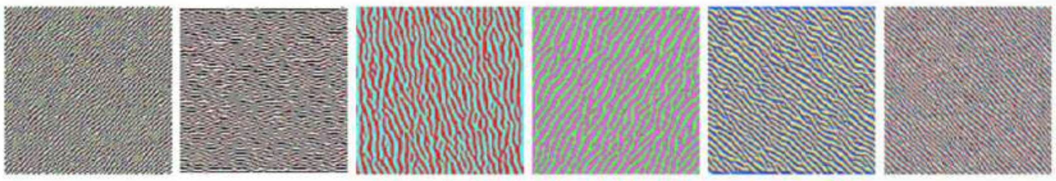
↳ Stride

↳ kernel size

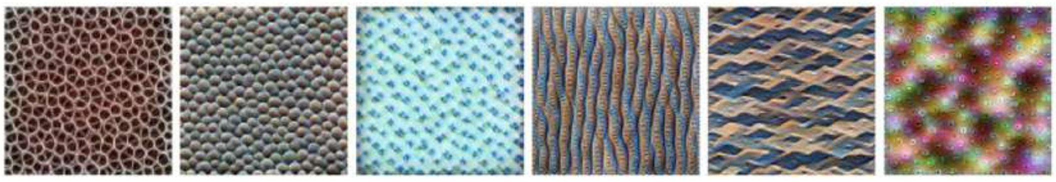
↳ # of filter

→ Padding

→ Maxpool.



Edges (layer conv2d0)



Textures (layer mixed3a)

Patterns (layer mixed4a)

Parts (layers mixed4b & mixed4c)

Objects (layers mixed4d & mixed4e)

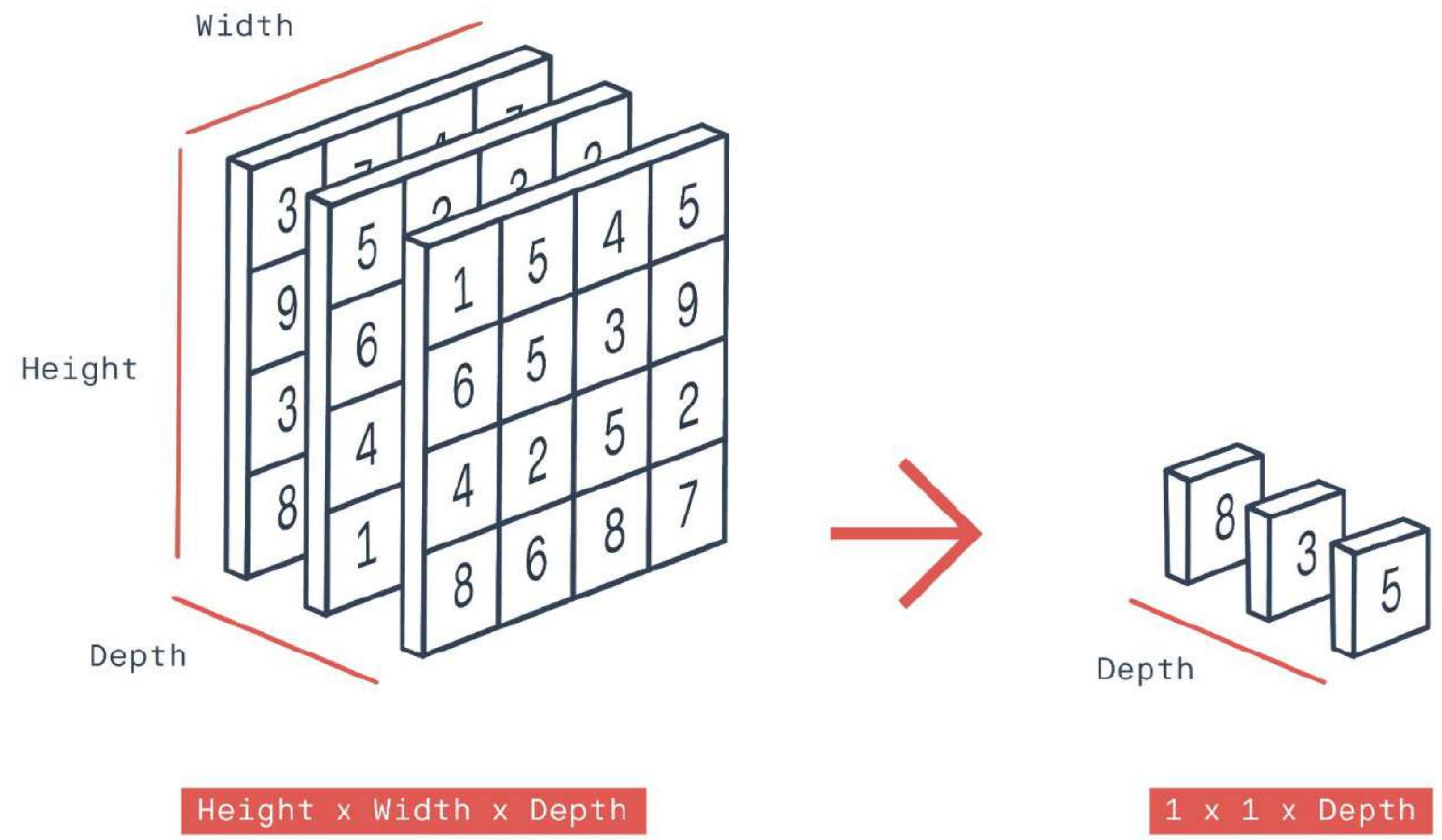


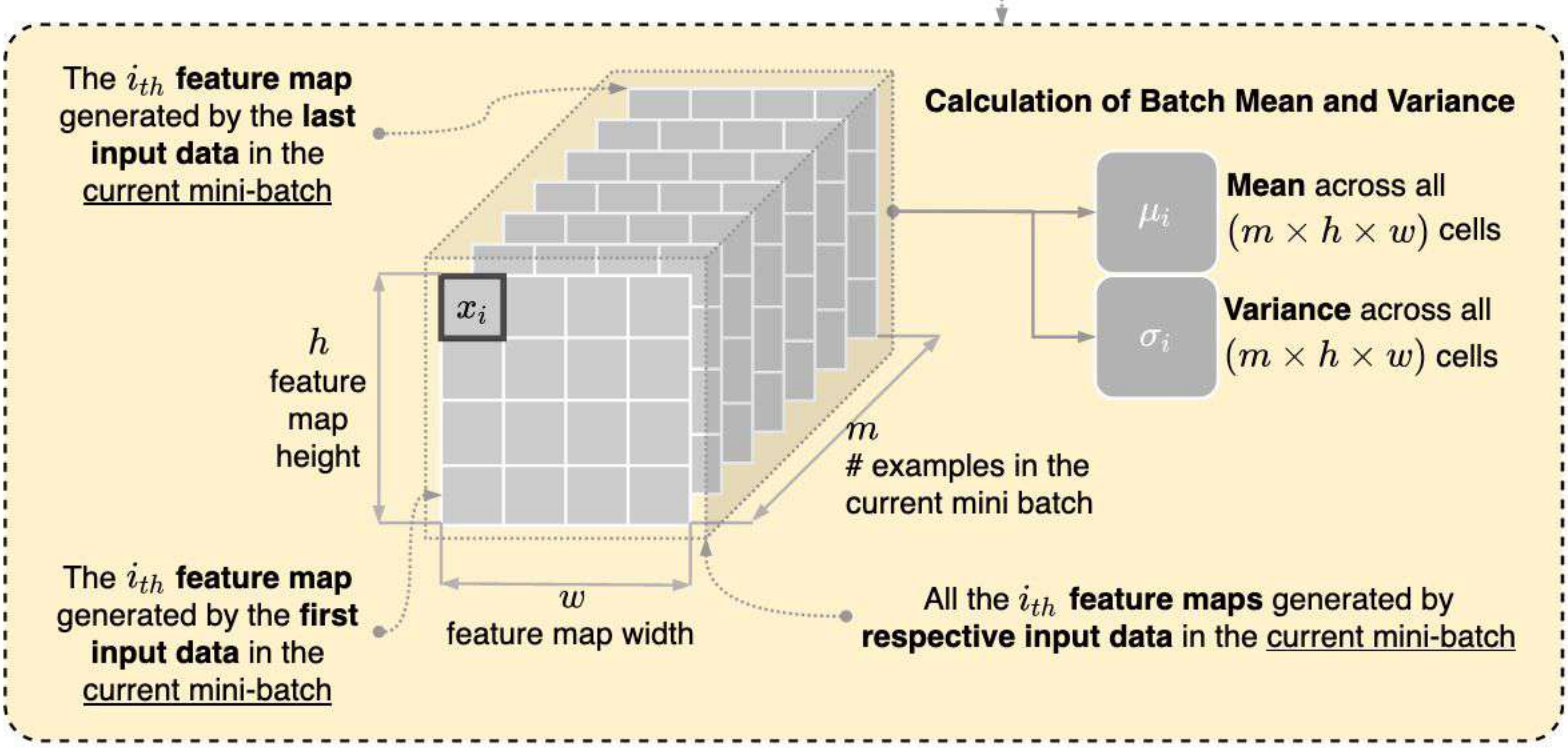
Figure 1. 2D Global average pooling operation. Here 'Depth' = 'Filters'

1. Computing the mean of your minibatch: $\mu_B^{(k)} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_{B_i}^{(k)}$.

2. Computing the variance of your minibatch: $\sigma^{2(k)}_B \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_{B_i}^{(k)} - \mu_B^{(k)})^2$

3. Normalizing the value: $\hat{x}_B^{(k)} \leftarrow \frac{x_B^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma^{2(k)}_B + \epsilon}}$

4. Scaling and shifting: $y_i \leftarrow \gamma \hat{x}_B^{(k)} + \beta$.


Calculation of Batch Mean and Variance
The i_{th} **feature map** generated by the **last input data** in the current mini-batch
x_i
h
feature map height
w
feature map width
m
examples in the current mini batch
The i_{th} **feature map** generated by the **first input data** in the current mini-batch
μ_i
Mean across all $(m \times h \times w)$ cells
σ_i
Variance across all $(m \times h \times w)$ cells
All the i_{th} **feature maps** generated by **respective input data** in the current mini-batch