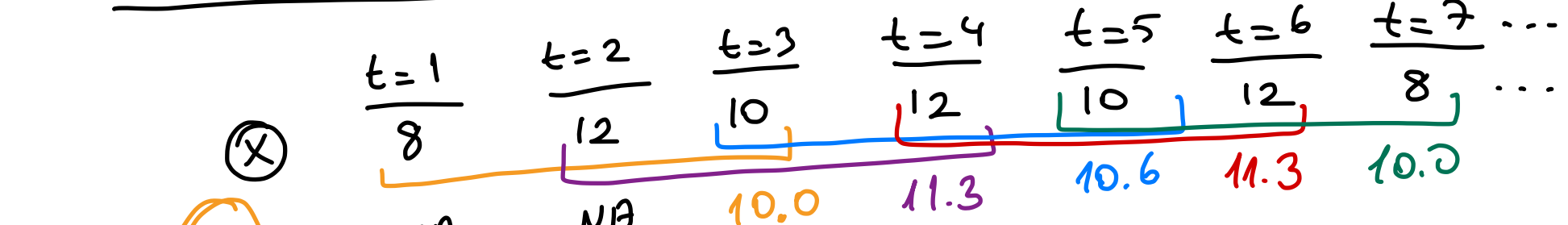
Convolutional Neural Networks (CNNs)

1D Convolution



(X) → 3MA → (y)

we assumed that $w_1 = w_2 = w_3 = \frac{1}{3}$

$$y_3 = \frac{x_1 + x_2 + x_3}{3} \quad \left\{ \frac{1}{3} \cdot x_1 + \frac{1}{3} x_2 + \frac{1}{3} x_3 \right.$$

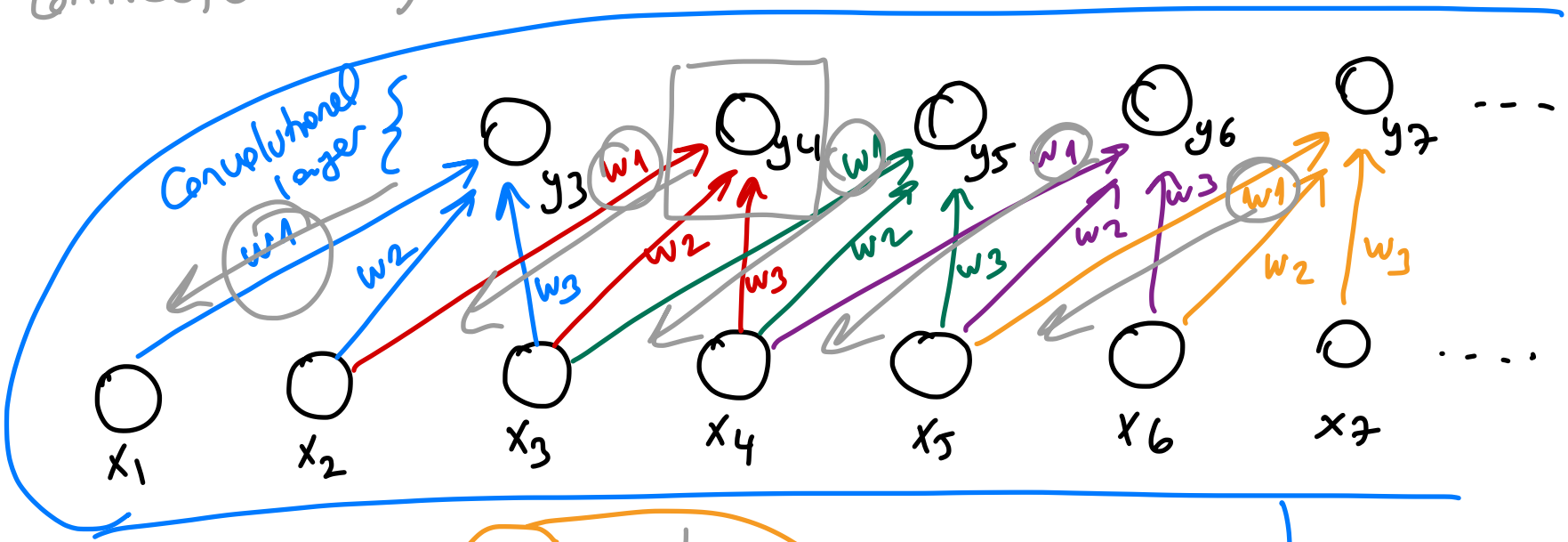
$$y_4 = \frac{x_2 + x_3 + x_4}{3} \quad \left\{ \frac{1}{3} x_2 + \frac{1}{3} x_3 + \frac{1}{3} x_4 \right.$$

$$\vdots$$

$$y_7 = \frac{x_5 + x_6 + x_7}{3} \quad \left\{ \frac{1}{3} x_5 + \frac{1}{3} x_6 + \frac{1}{3} x_7 \right.$$

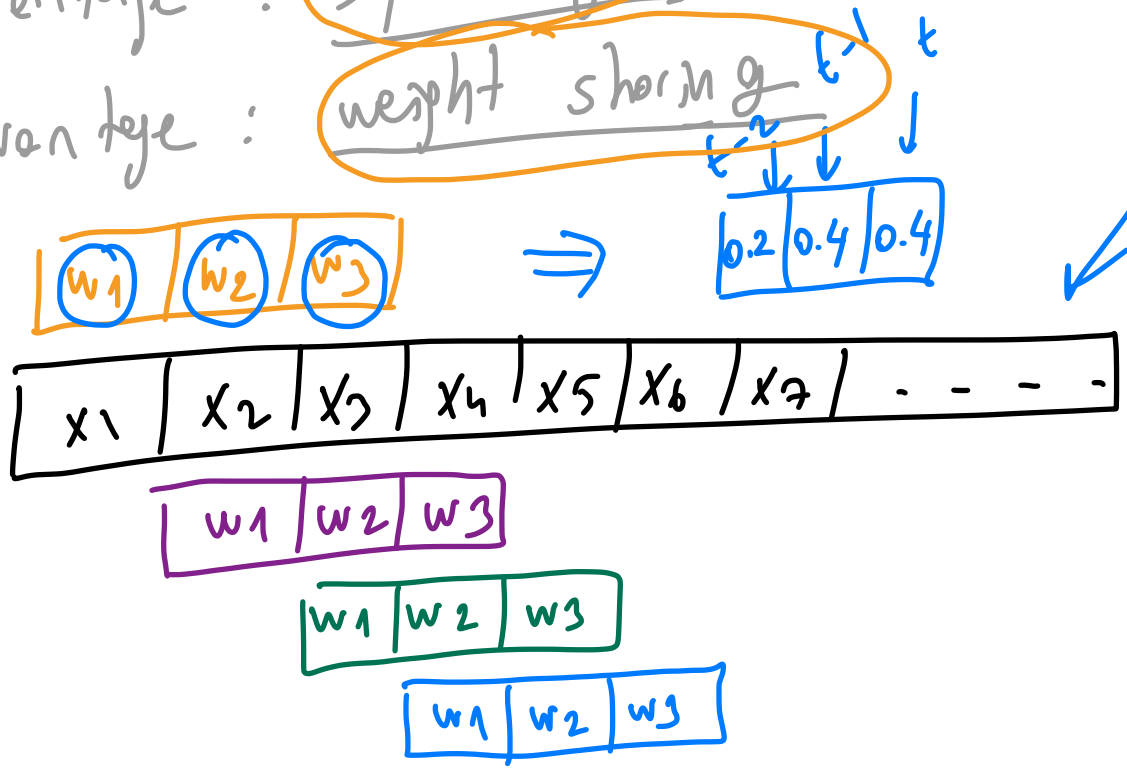
$$y_t = \frac{1}{3} x_{t-2} + \frac{1}{3} x_{t-1} + \frac{1}{3} \cdot x_t$$

fully connected layer = 7 × 5 = 35 connections in total



1st advantage : sparsity

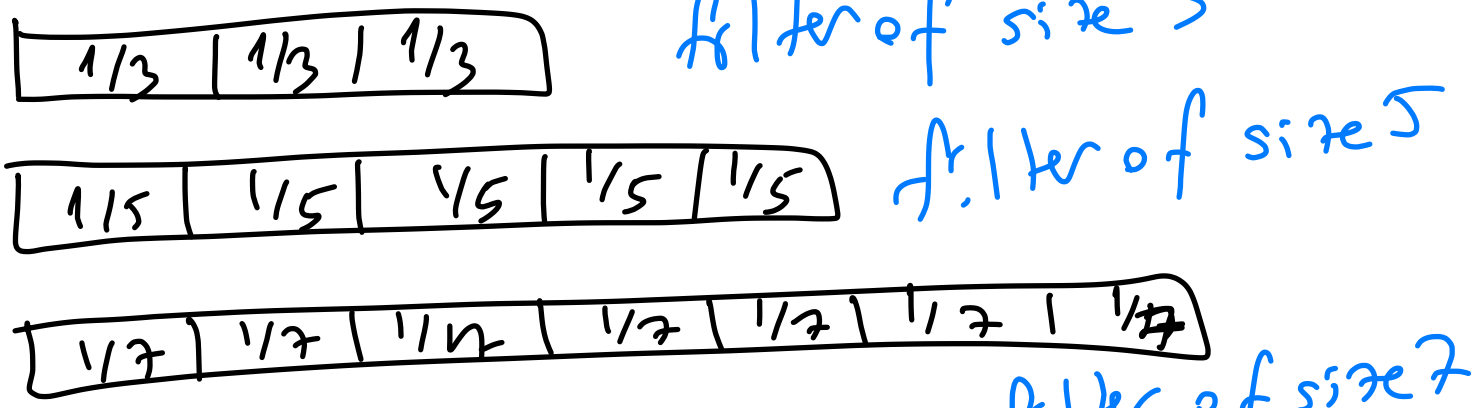
2nd advantage : weight sharing



3MA

5MA

7MA



filter of size 3

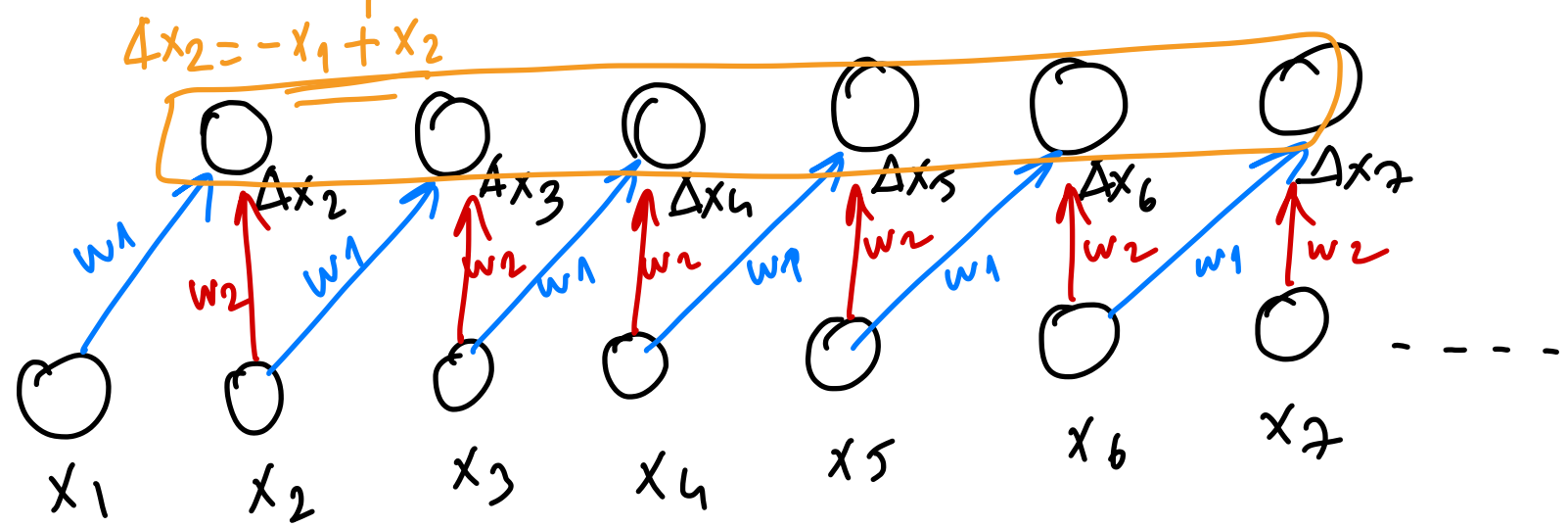
filter of size 5

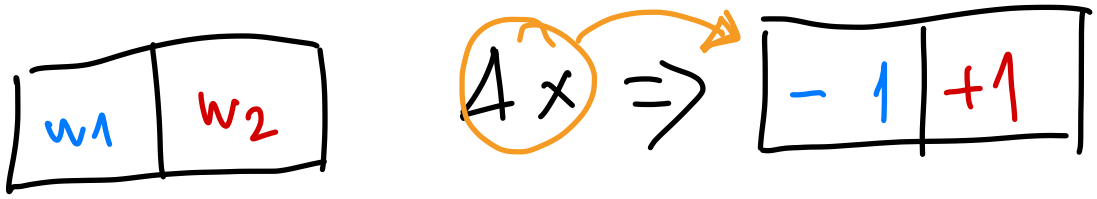
filter of size 7

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	...
x	8	12	10	12	10	12	8	...
1st ← Δx	NA	4	-2	2	-2	2	-4	....
2nd ← ΔΔx	NA	NA	-6	4	-4	4	-6	.....

$$\Delta x = \frac{x_t - x_{t-1}}{t - (t-1)} = x_t - x_{t-1}$$

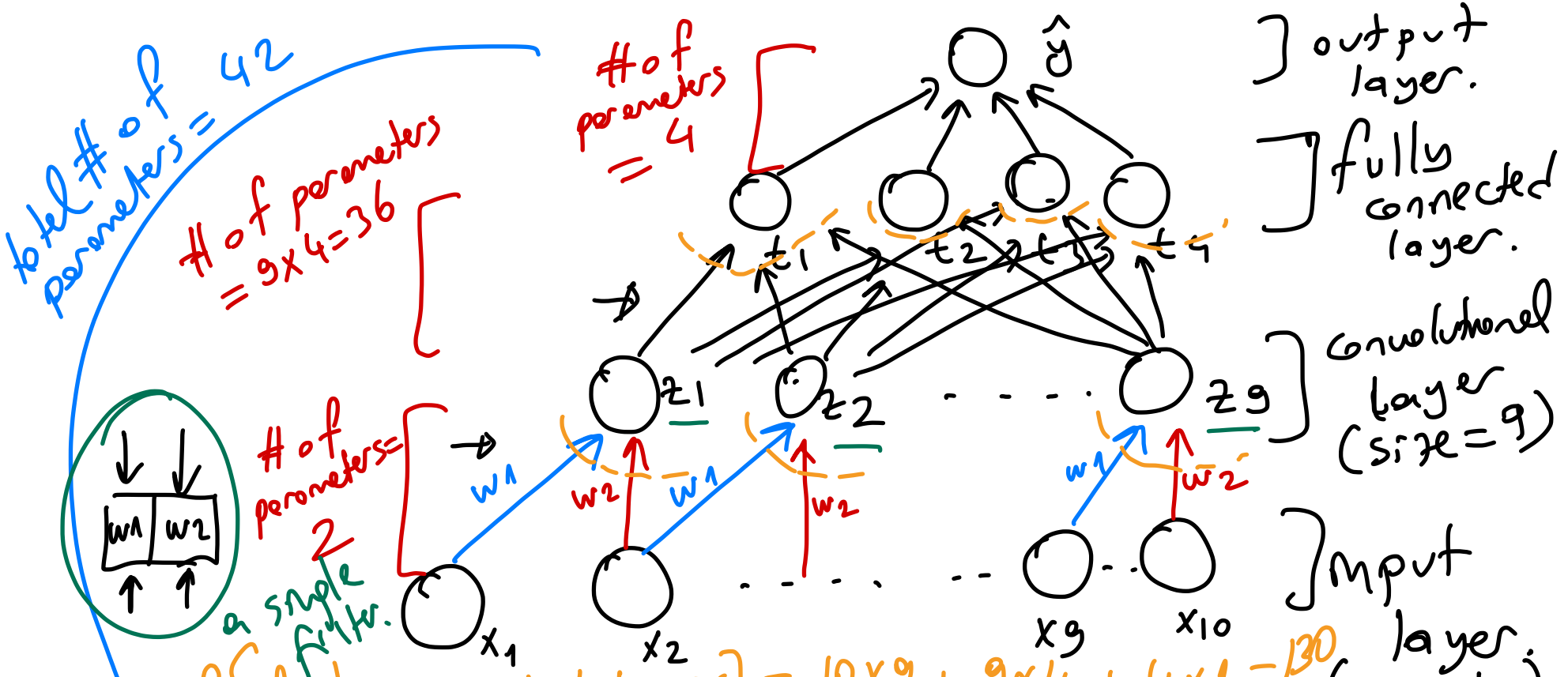
$$\Delta x_t = x_t - x_{t-1}$$



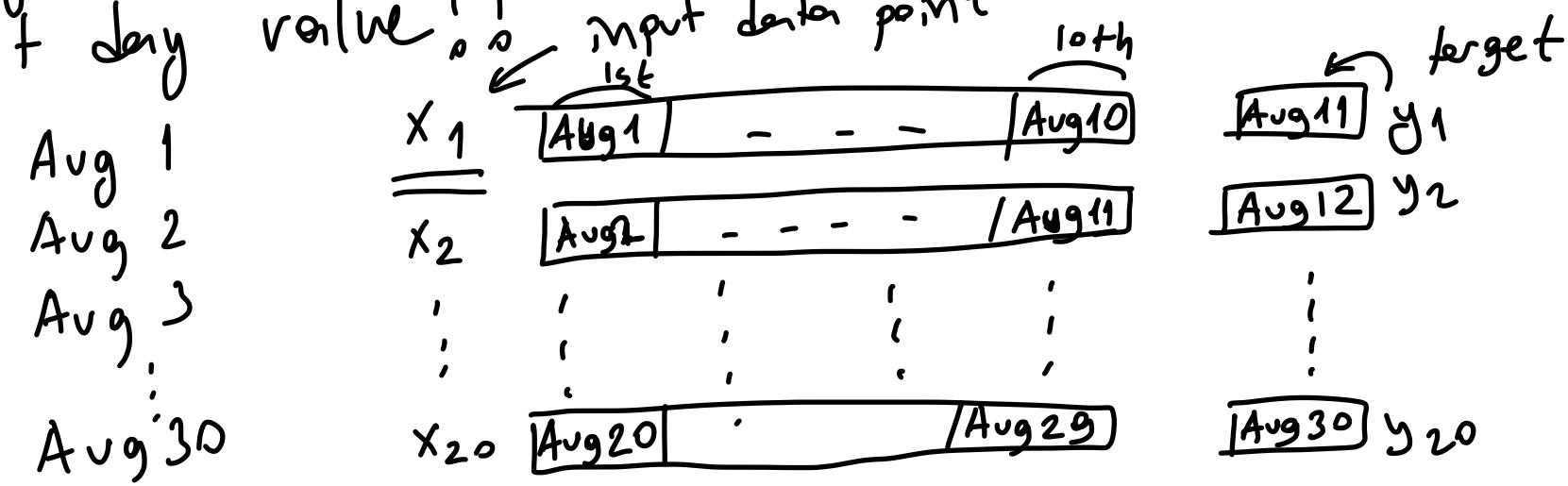


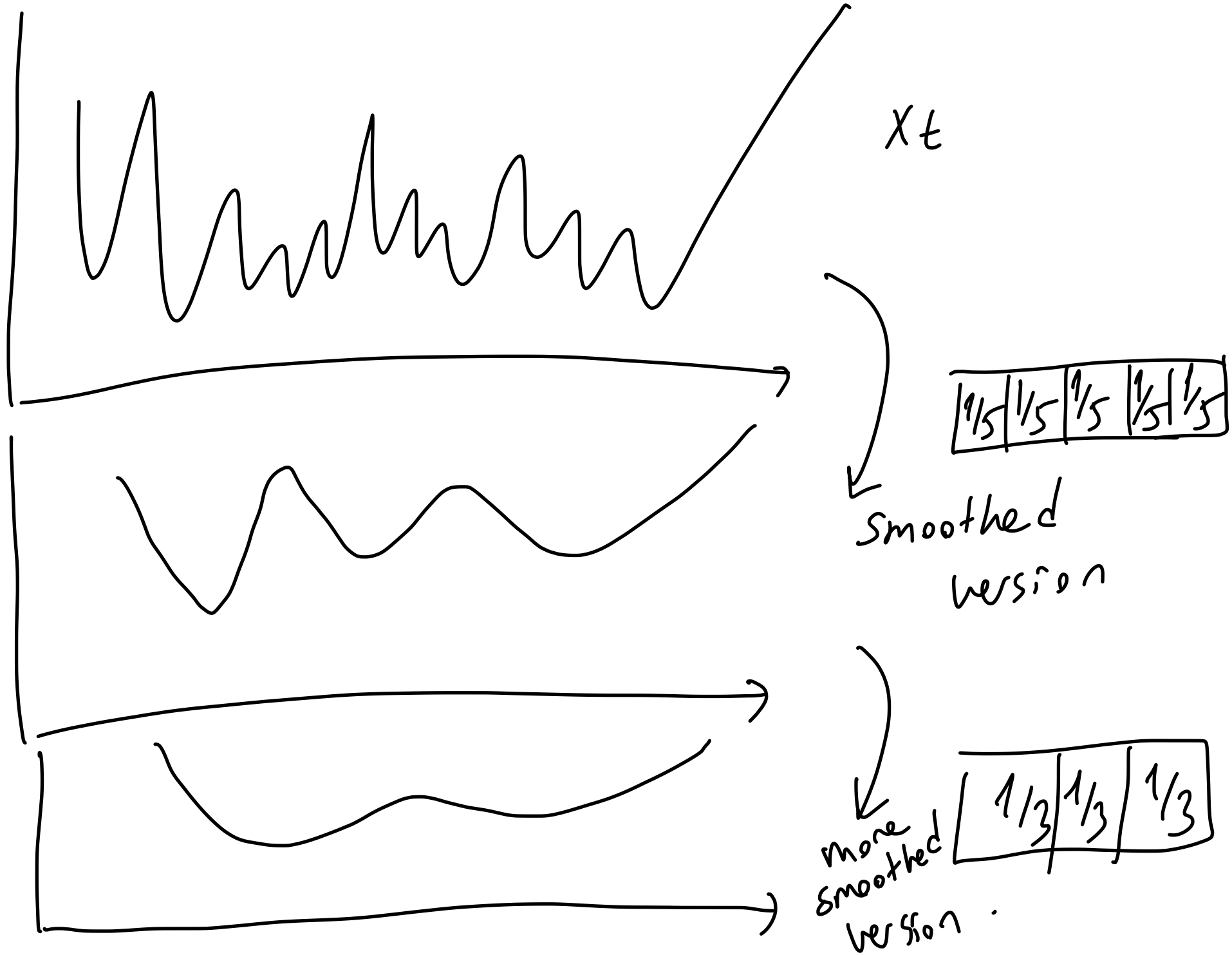
$-0.67 \quad 0.89 \Rightarrow 0.89\, x_t - 0.67\, x_{t-1}$

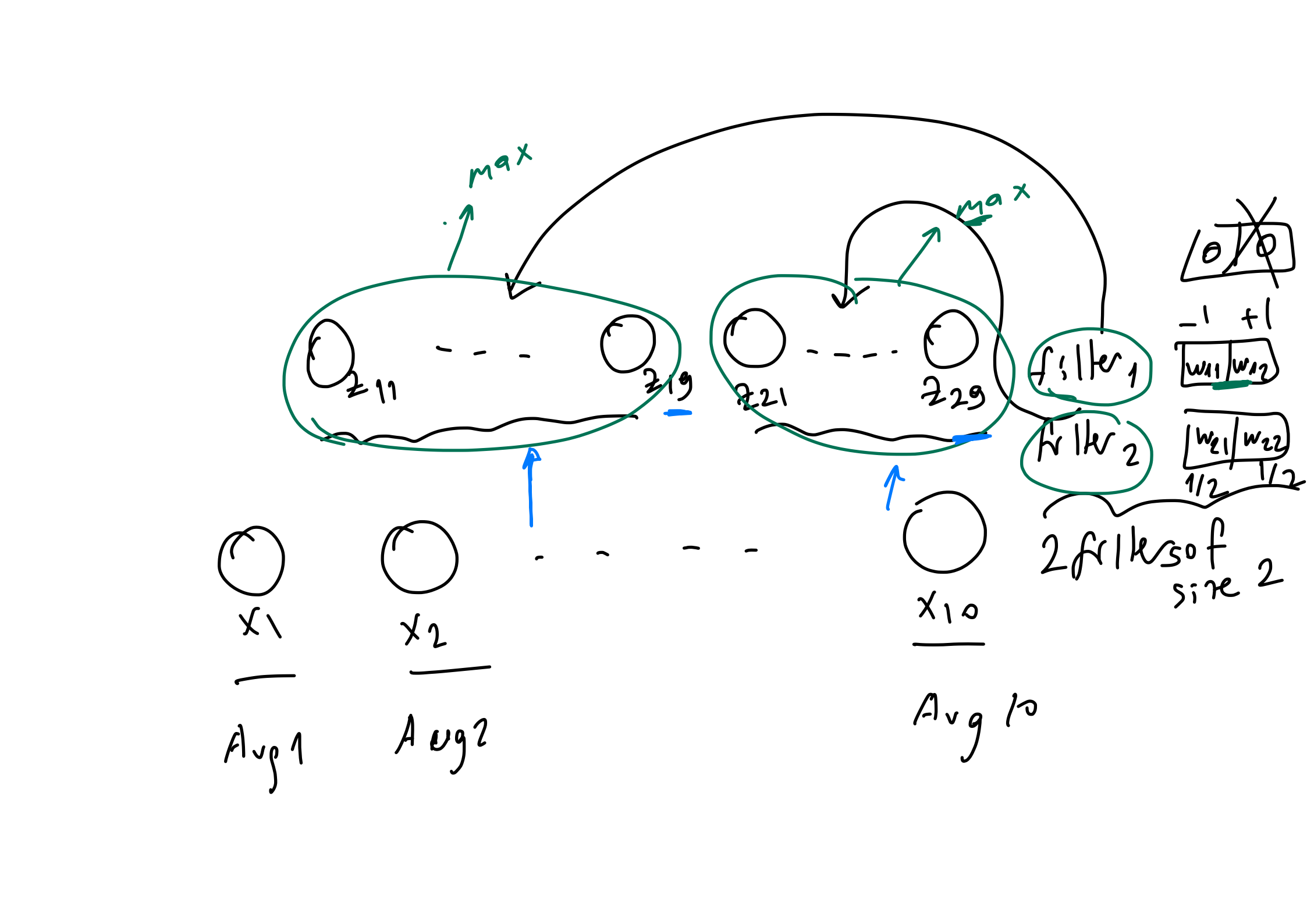
↳ by learning the filters instead of fixing them to constants.



Using last 10 days' USD/TRY rates, predict the next day value!!

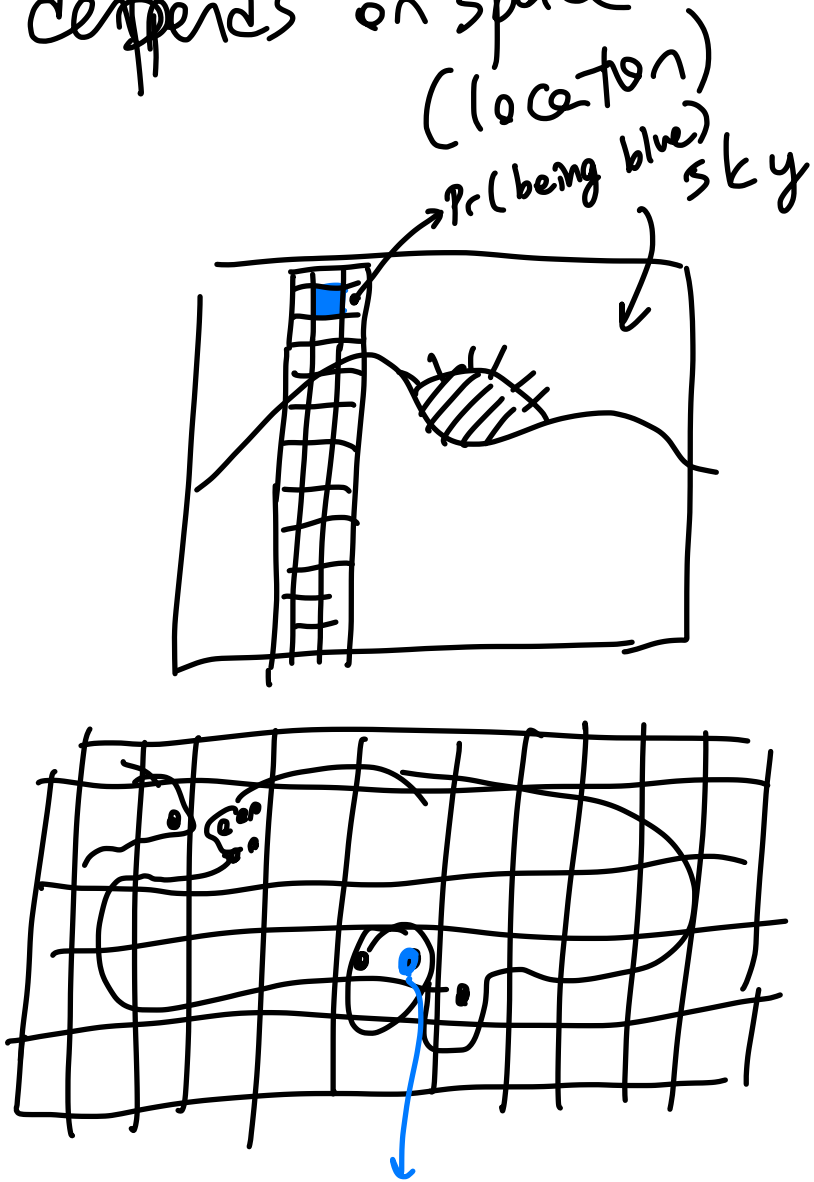



Xt
1/5 1/5 1/5 1/5 1/5
smoothed version
1/3 1/3 1/3
more smoothed version.


max
max
z_{11}
z_{19}
z_{21}
z_{29}
filter 1
filter 2
0 0
-1 +1
w_{11} w_{12}
w_{21} w_{22}
1/2 1/2
2 filters of size 2
x_1
x_2
x_{10}
Avg 1
Avg 2
Avg 10

Spatial

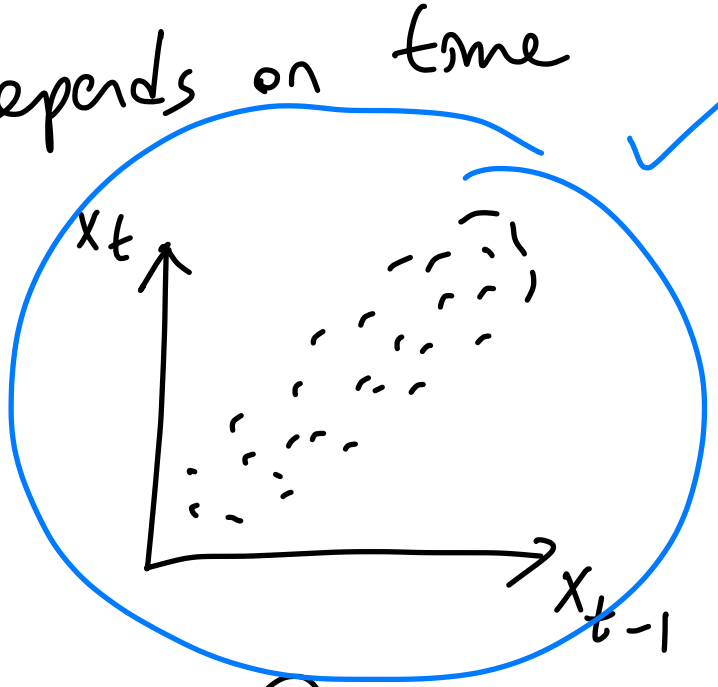
depends on space (location)



temporal

x_t x_{t+1} x_{t+2}

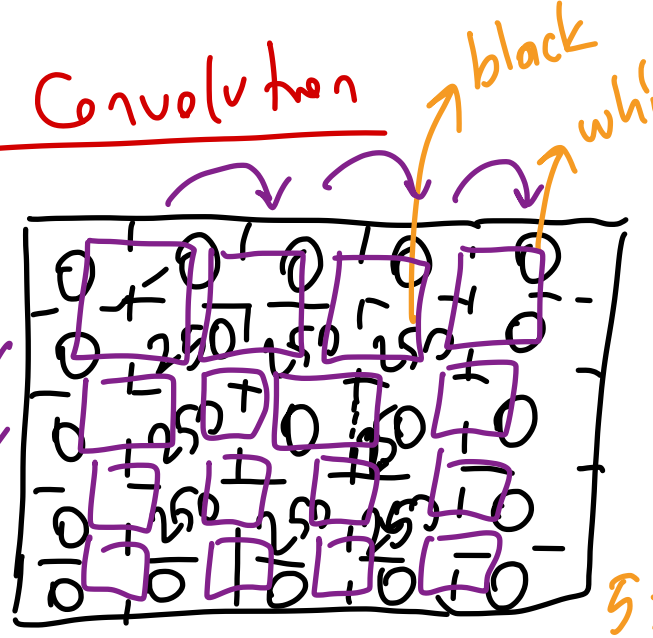
depends on time

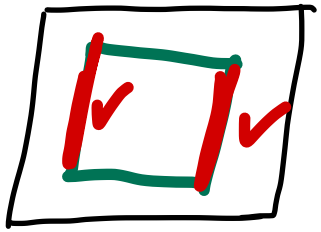


Adana = $38^{\circ}C$ ($35-40^{\circ}C$)

Mersin ⇒

2D Convolution





original picture.

5x5 = 25 pixels

filter of size 2x2

filter 1

-1	+1
-1	+1

filter 2

-1	-1
+1	+1

top left

horizontal lines

looking for vertical lines.

	1	2	3	4	5
1	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
2	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
3					
4					
5	x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

$(-1)x_{11} + (+1).x_{12} +$

$(-1)x_{21} + (+1)x_{22}$

filter 1

250	0	0	-250
500	-250	+250	-500
500	-250	+250	-500
250	0	0	-250

4x4

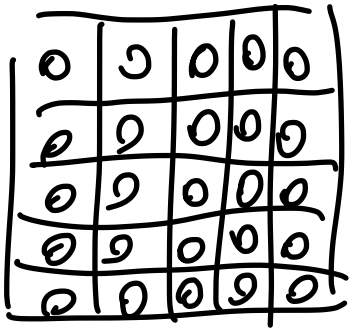
filter 2.

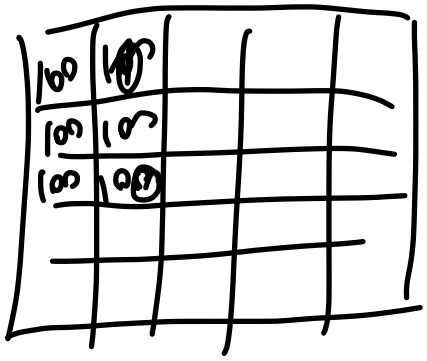
250	500	500	250
0	-250	+250	0
0	-250	+250	0
-250	-500	-500	-250

filter 3

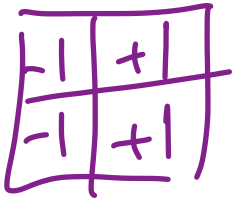
-1	+1
+1	-1

→ dragonal lines.

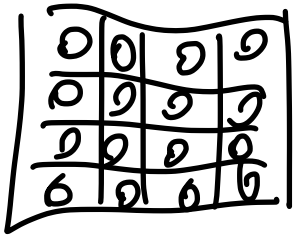


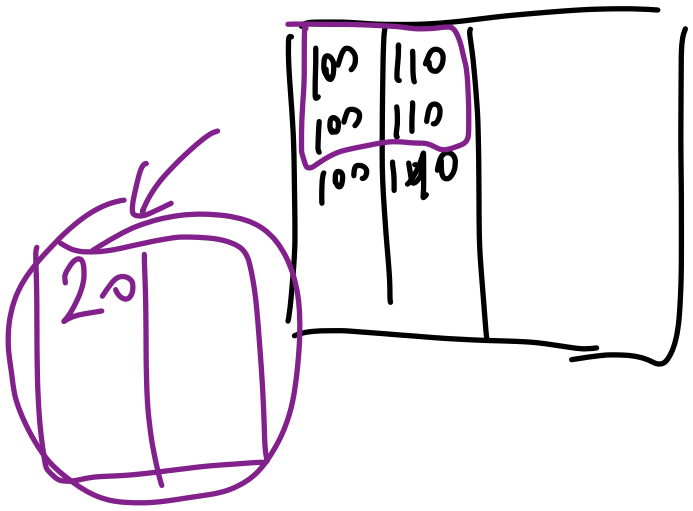


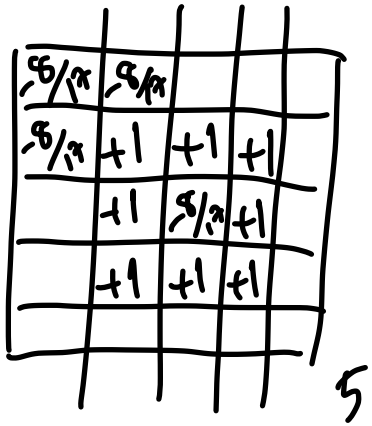
apply filter 1, 2 or 3



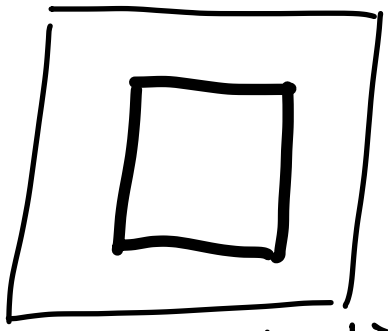
result



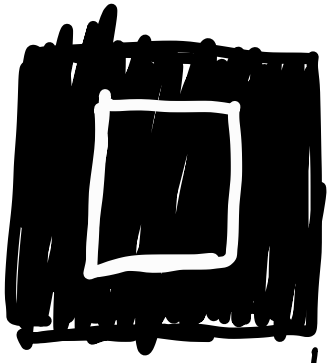




5 x 5



gradient > 0

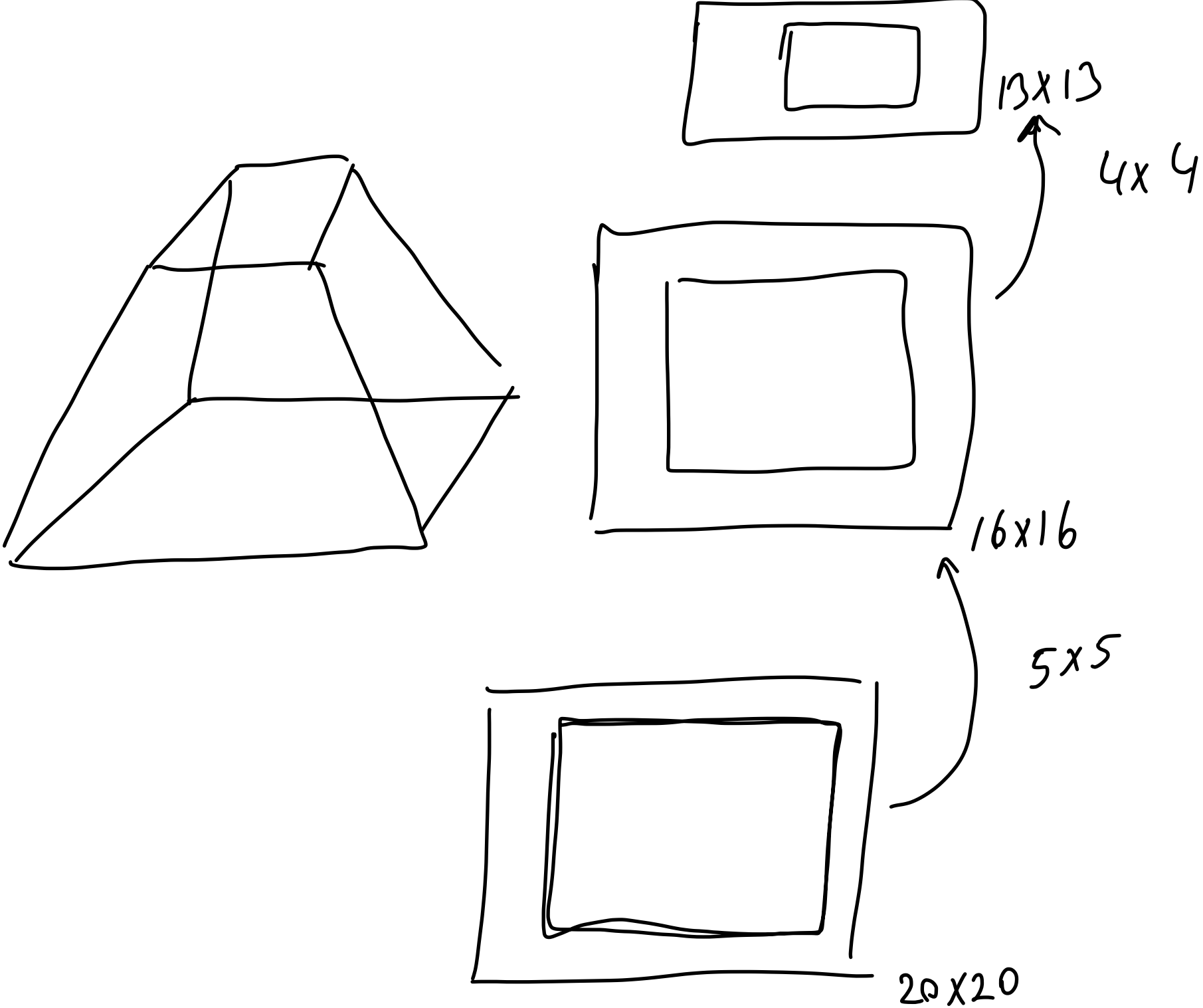


gradient < 0

filter will recognize both of them.

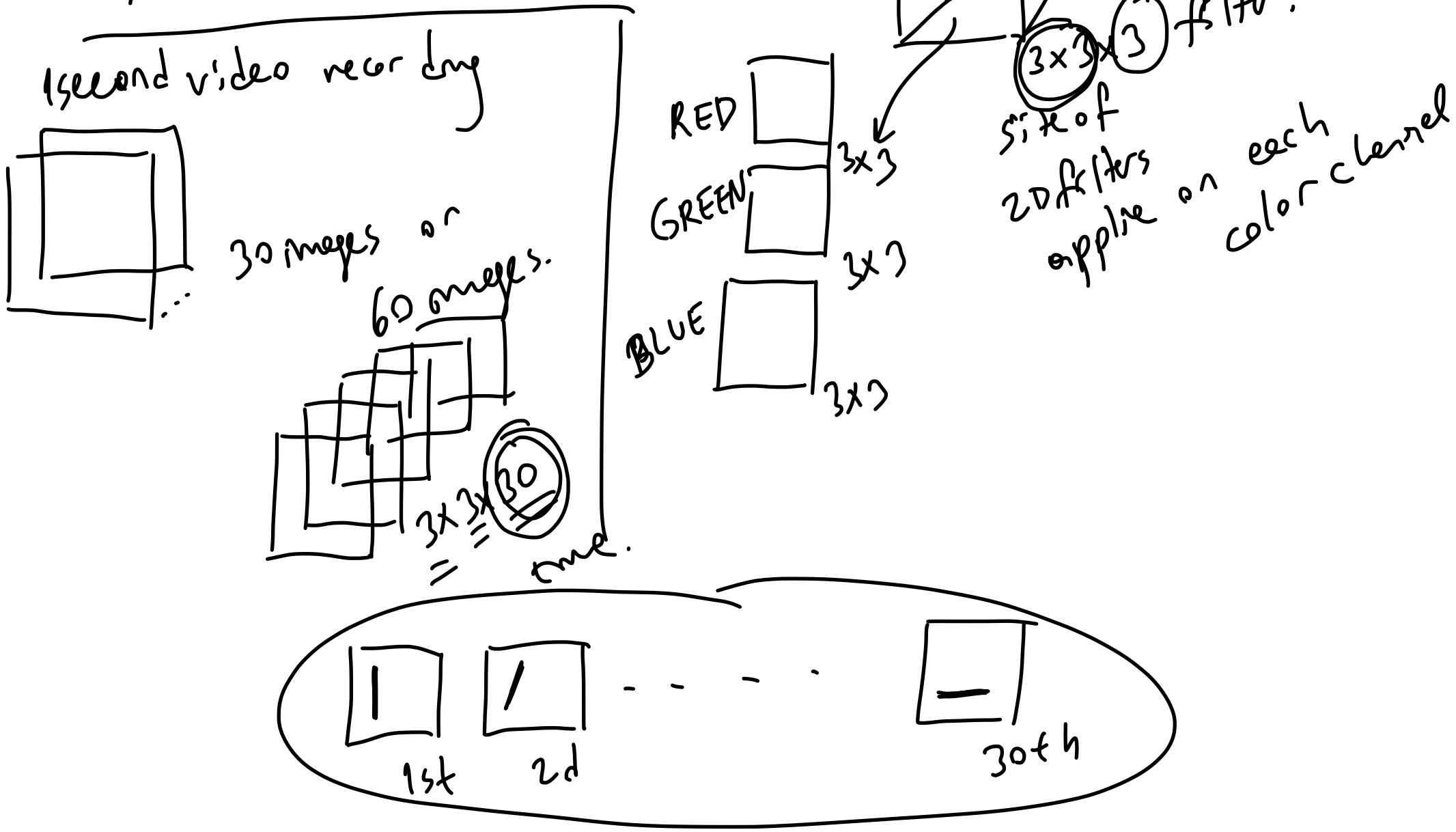
Gabor Filters

- manually designed (hand-crafted) filters.
- they use filters to extract features.
- feed extracted features to SVMs, kNNs, logistic regression ...


13x13
4x4
16x16
5x5
20x20

3D Convolution

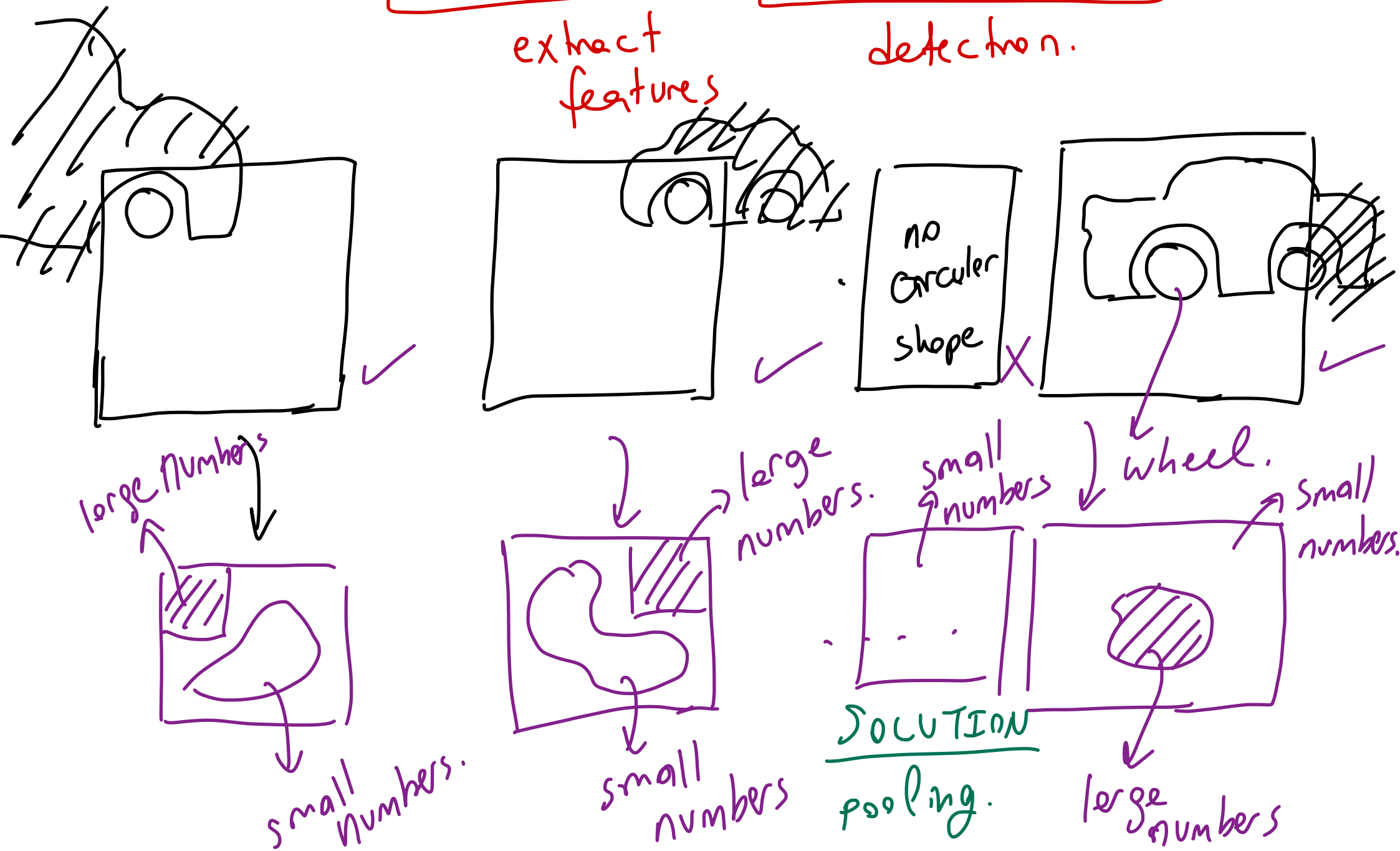
filters will be 3D

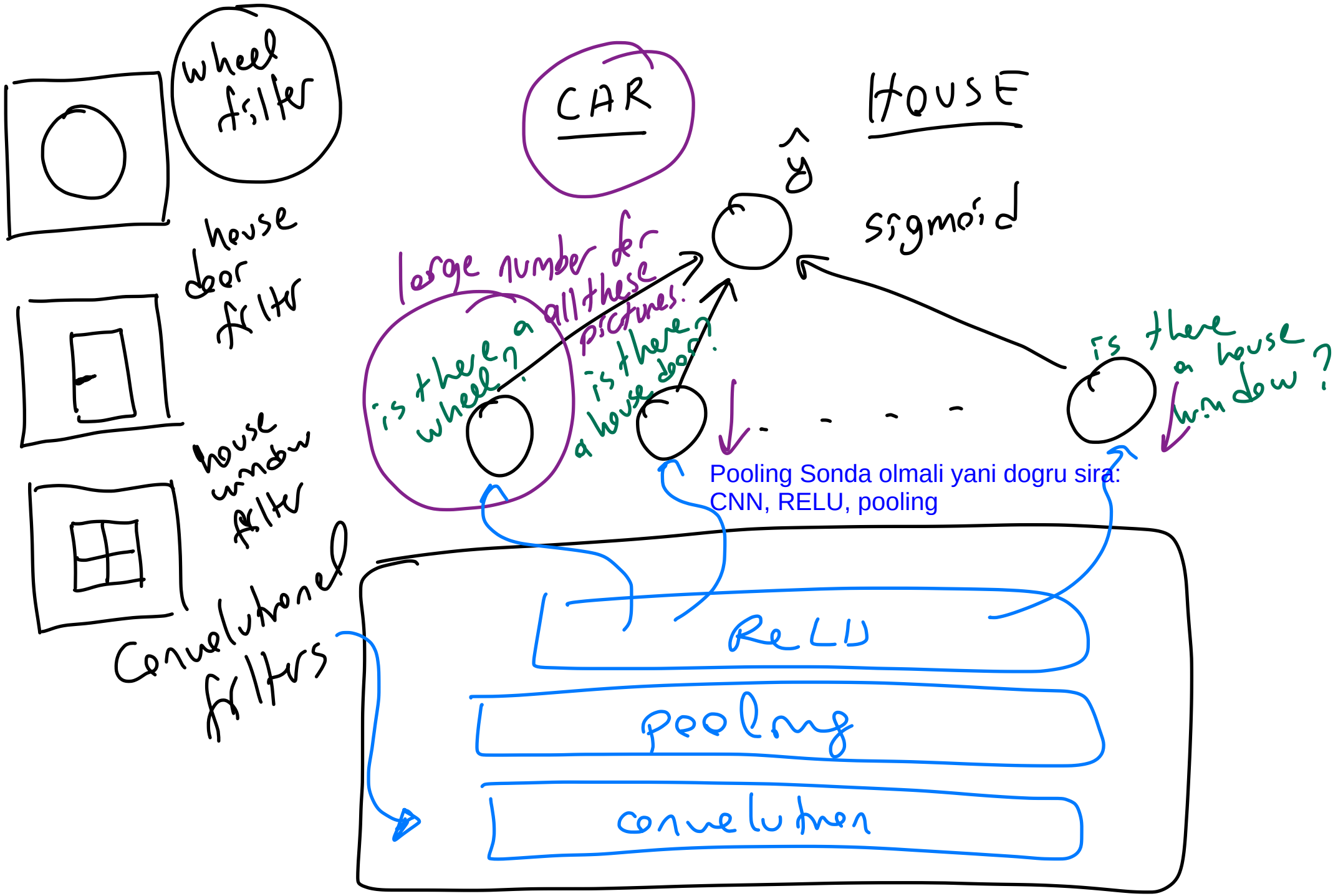


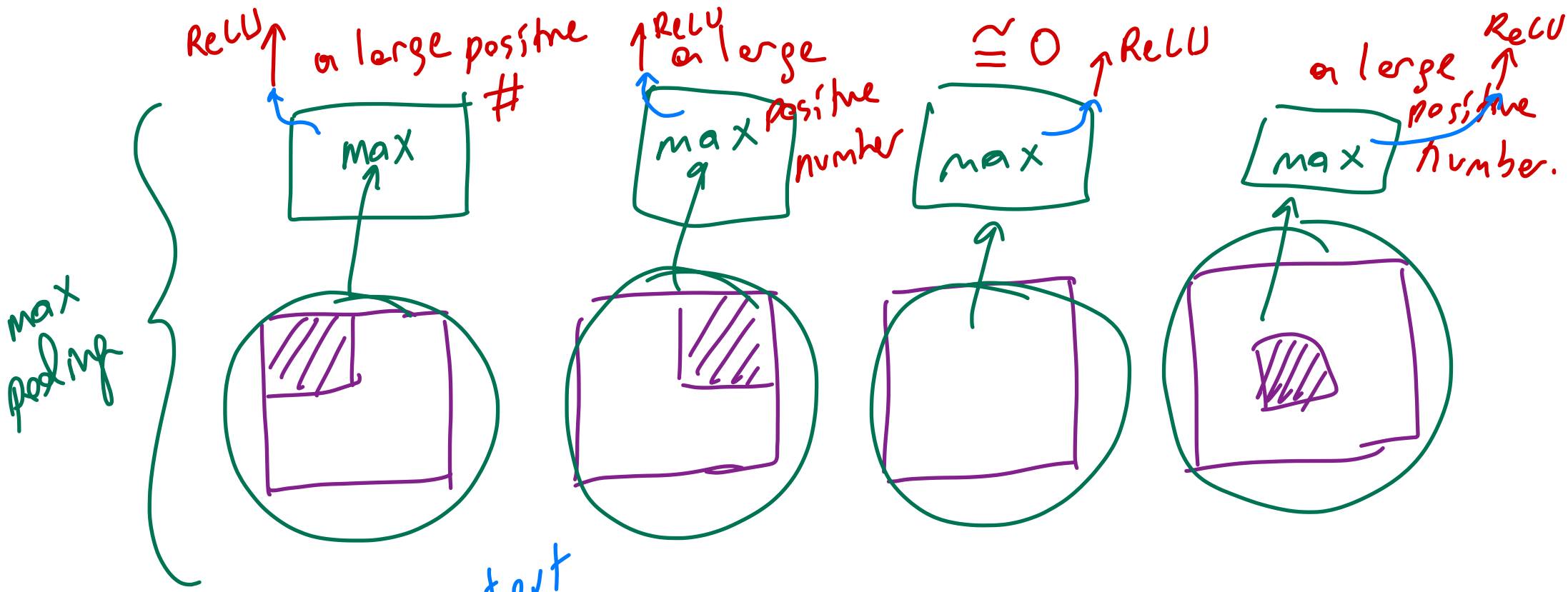
Convolutional → Pooling → ReLU

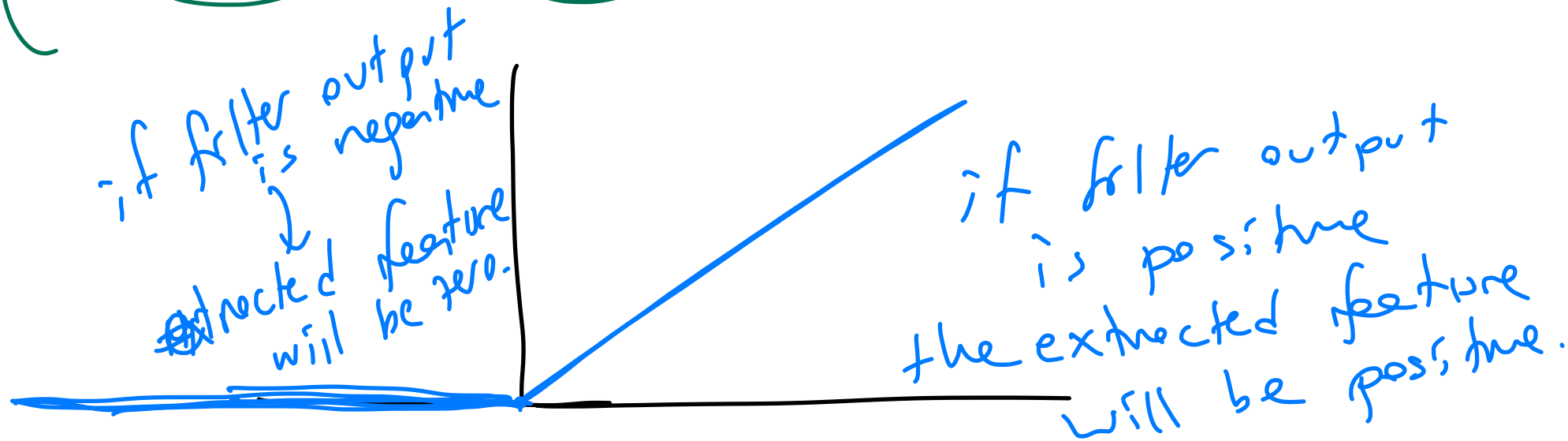
extract features

detection.



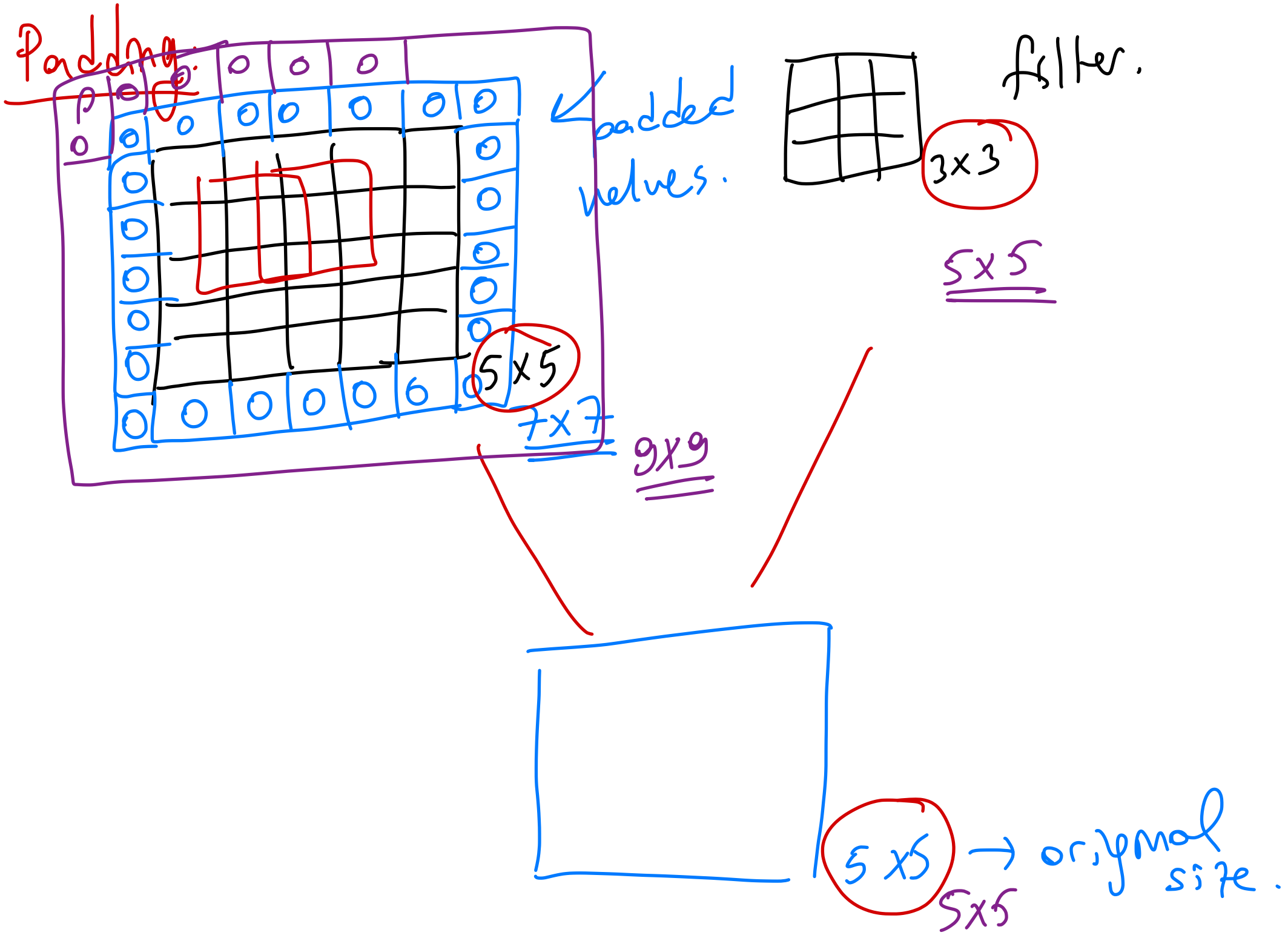

wheel filter
house door filter
house window filter
Convolutional filters
CAR
HOUSE
ŷ
sigmoid
large number of all these pictures.
is there a wheel?
is there a house door?
is there a house window?
Pooling Sonda olmali yani dogru sira:
CNN, RELU, pooling
RELU
pooling
convolution





↳ ReLU ⇒ $(0, \infty)$

Sigmoid ⇒ $(0,1)$ tanh ⇒ $(-1,+1)$


Padding.
padded values.
filter.
3x3
5x5
5x5
7x7
9x9
5x5
5x5
original size.

t=0 t=1 t=2 ... t=7

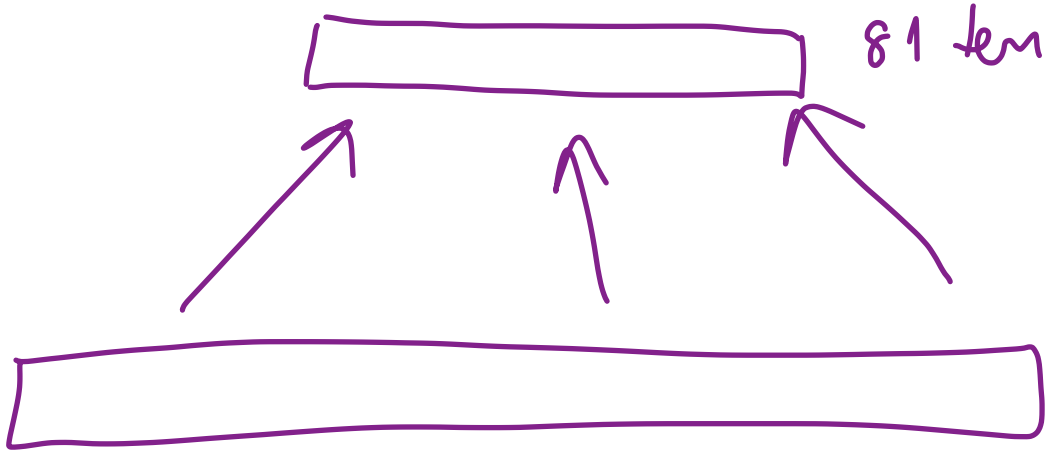
0, 8, 12, 10, 12, 10, 10, 8, ----

Δx 8 4 -2 2 -2 0 -2

$$\text{length}(\Delta x) = \text{length}(x)$$

---

81 cities × 10 last days ⇒ Input data of size 810



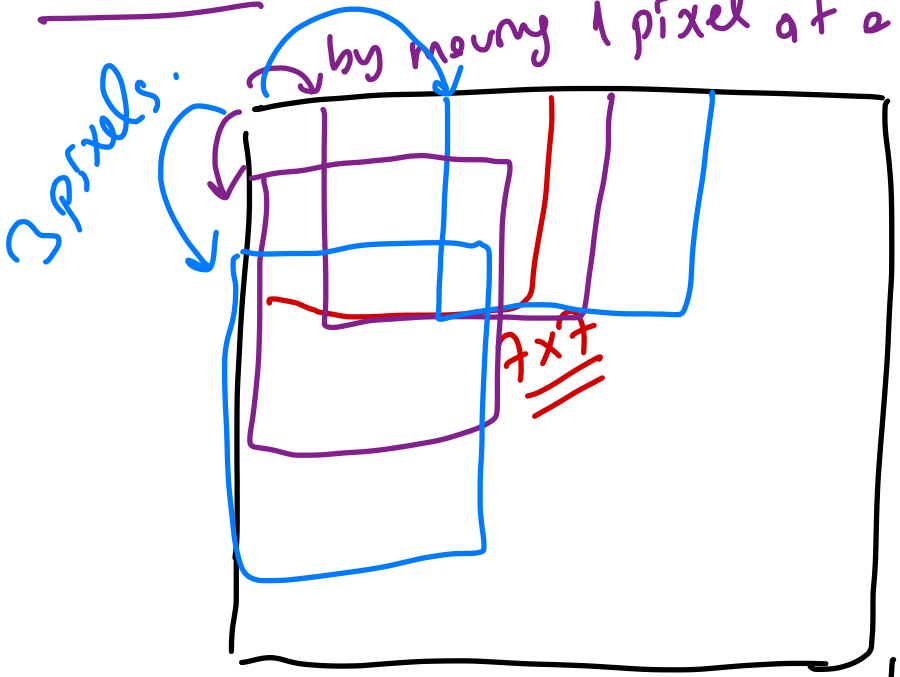
81 temperatures of tomorrow.

810 temperatures

Stride

usually done for computational reasons

by moving 1 pixel at a time. (stride 1)



100 x 100

Stride (3)

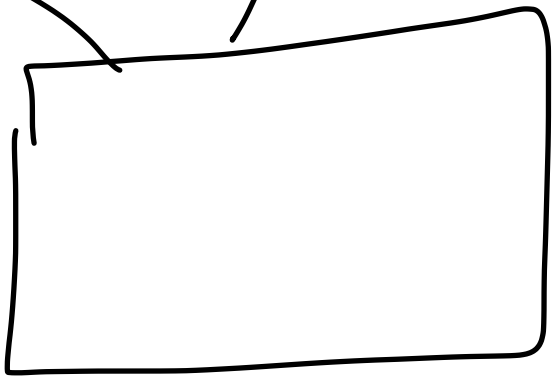
→ 49 parameters.

7x7 filter.

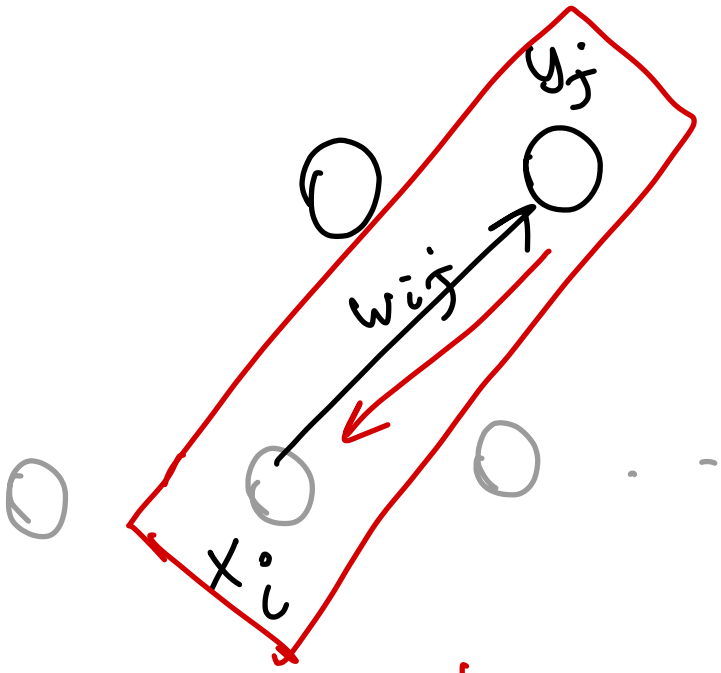
10 filters

⇓

490 parameters.



94 x 94 will be smaller.



$$\frac{\partial \text{loss}}{\partial w_{ij}} =$$

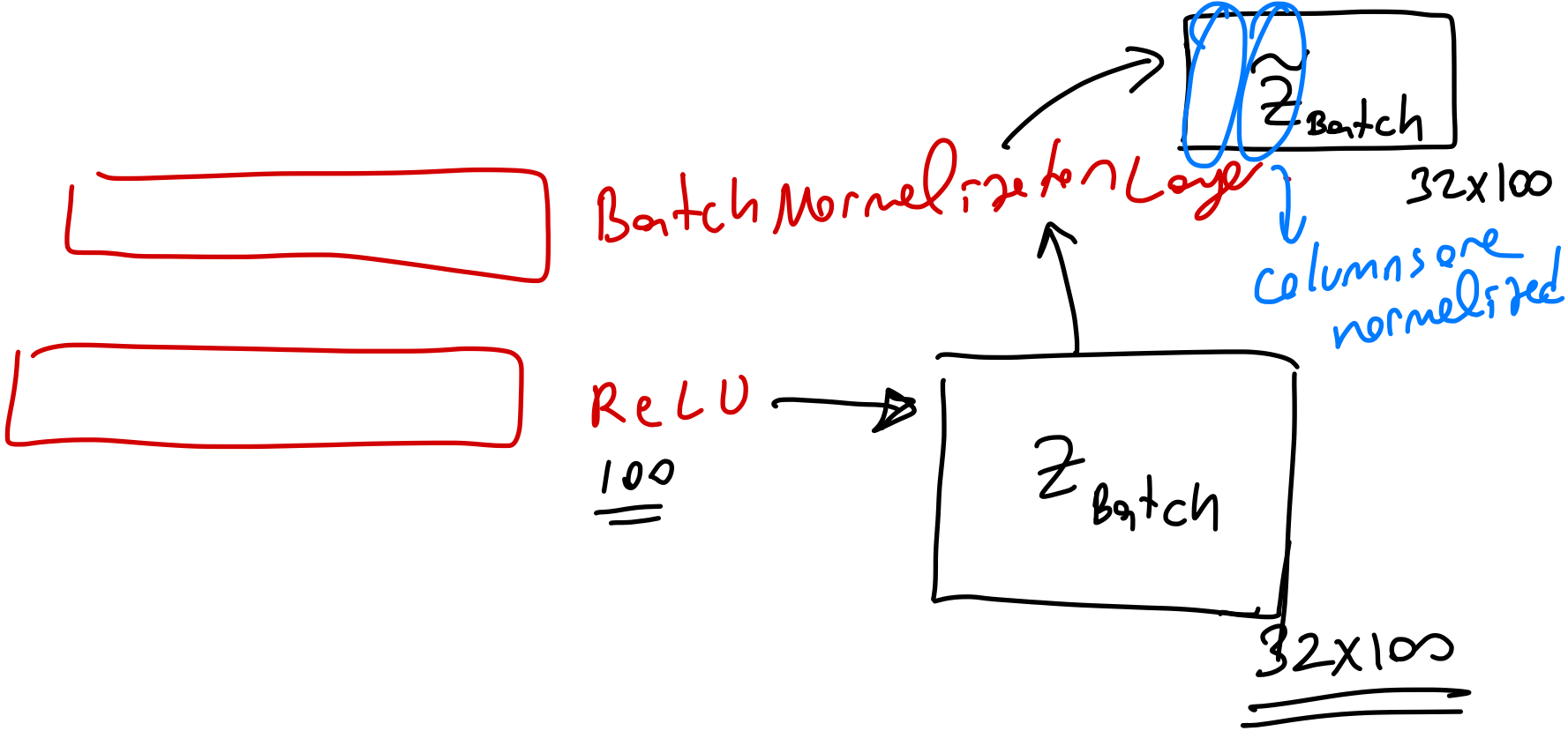
Batch Normalization

- vanishing gradient
- exploding gradient

$$\frac{\partial f(x)}{\partial x} = \frac{\partial f(x)}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial x}$$

$\Rightarrow$ will go to 0

$\Rightarrow$ will go to ∞


Batch Normelizeton Laye
$\tilde{Z}_{Batch}$
32x100
columns are normelized
ReLU
100
Z_{Batch}
32x100