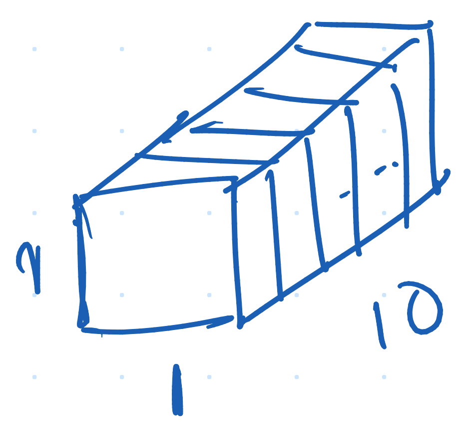
Parameter Calculation in DNNs

19/02/25

batch size = 1

1 x 1 x 1 x 3072

1 x 10 x 1 x 3072 →



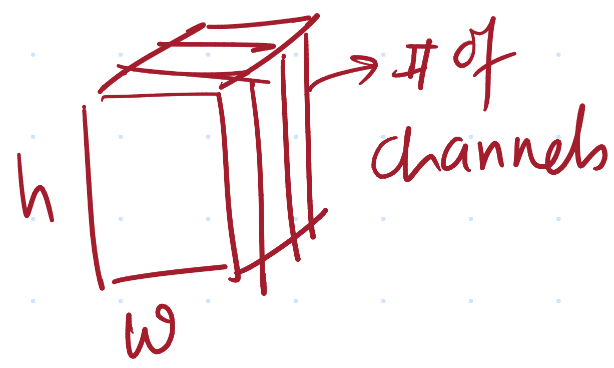
1 x 10 x 1 x 1

↓

output.

Dimension of Convolution block :-

(batch-size x number of channels x height x width.)



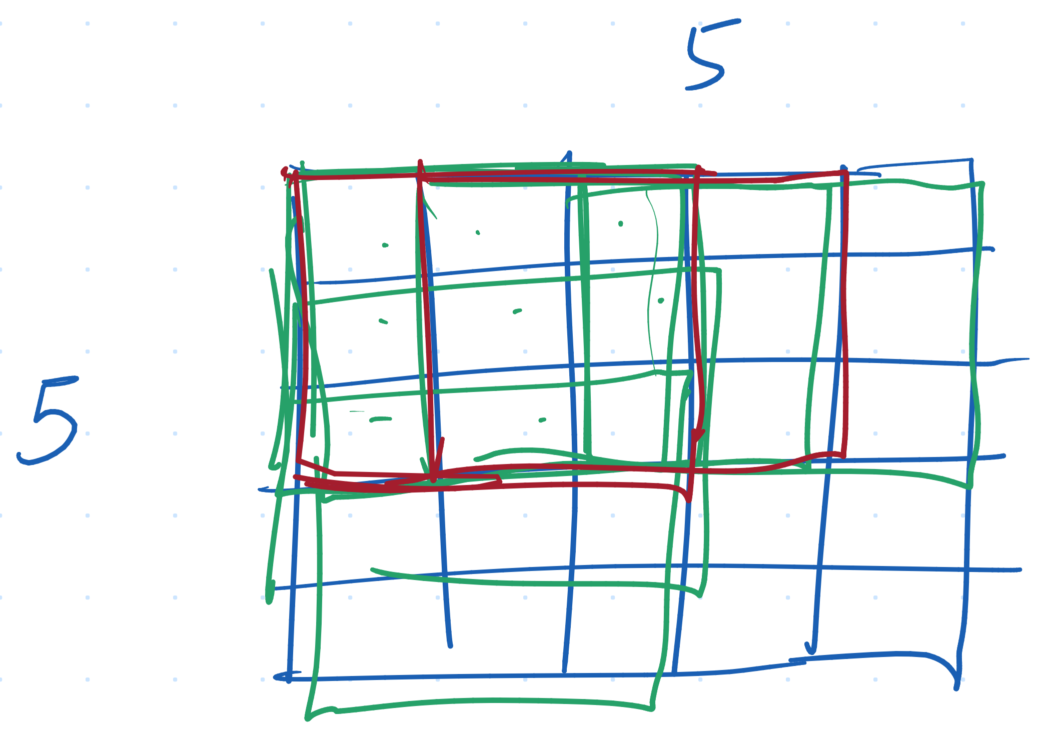
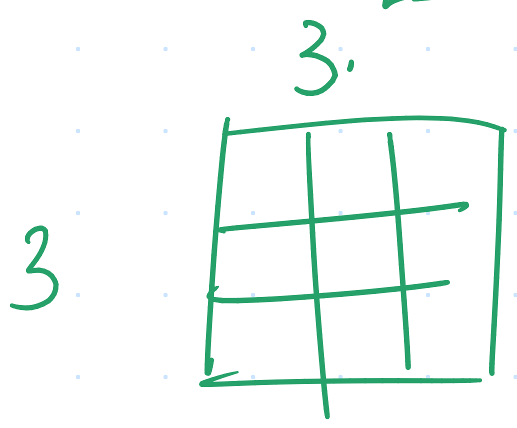


Image.

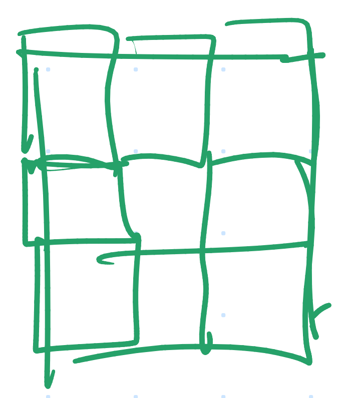
1-channel ⇒ image.

1 X 1 X H X W

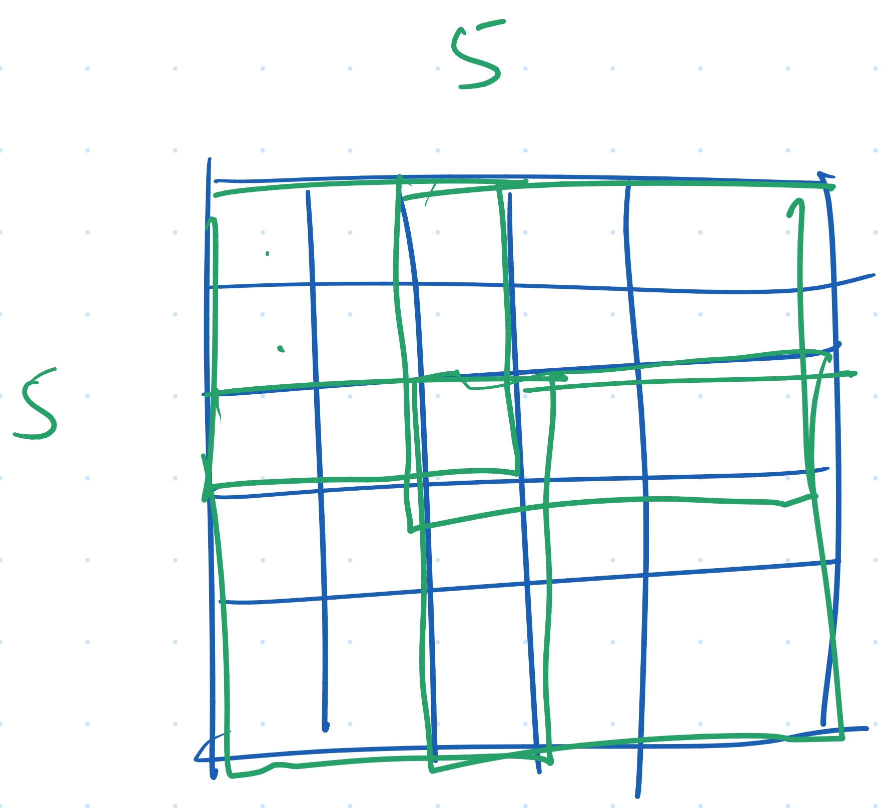


convolution →

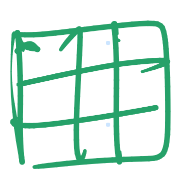
stride = 1



3x3.

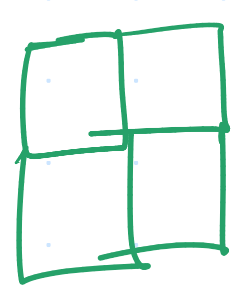


3x3 kernel

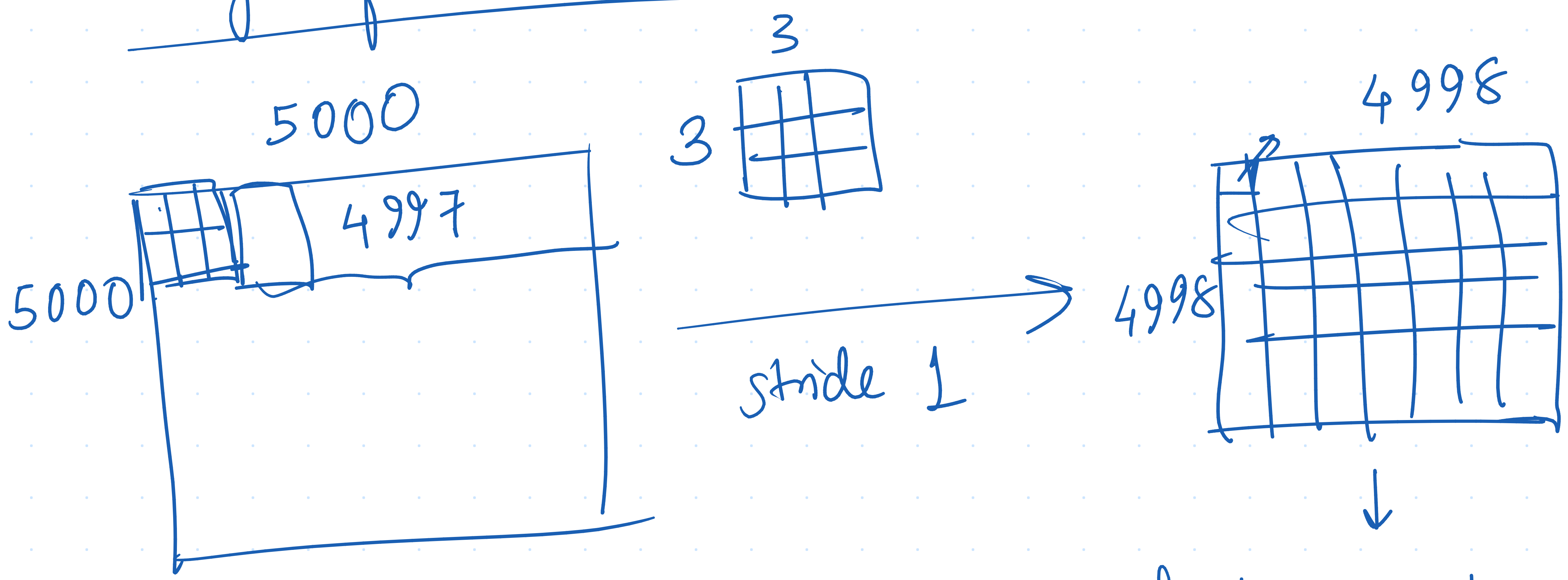


→

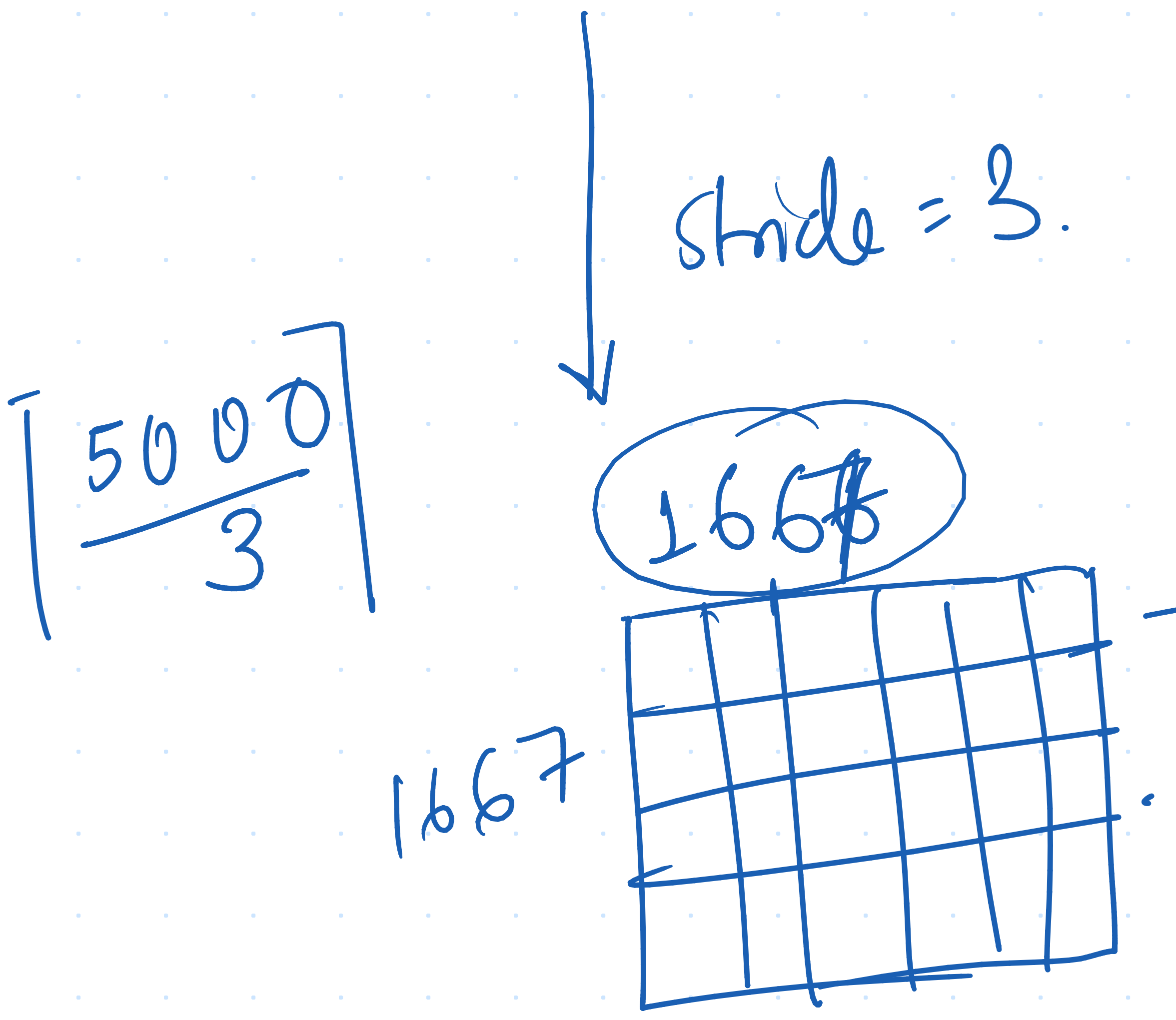
with stride = 2



Usage of strides.

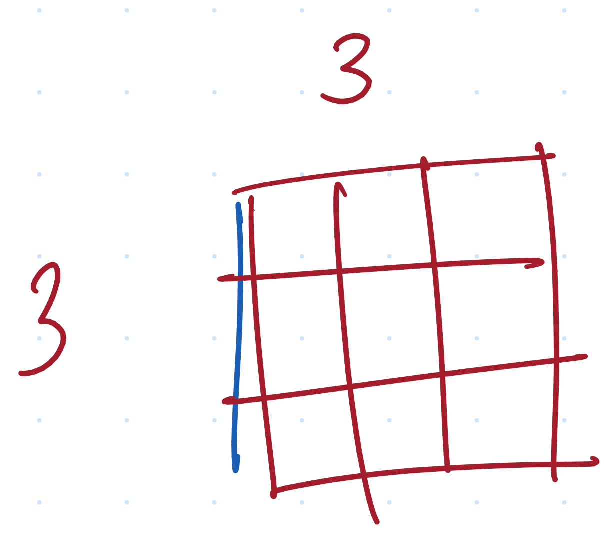


feature map → increase with small stride.



→ feature map size reduces.

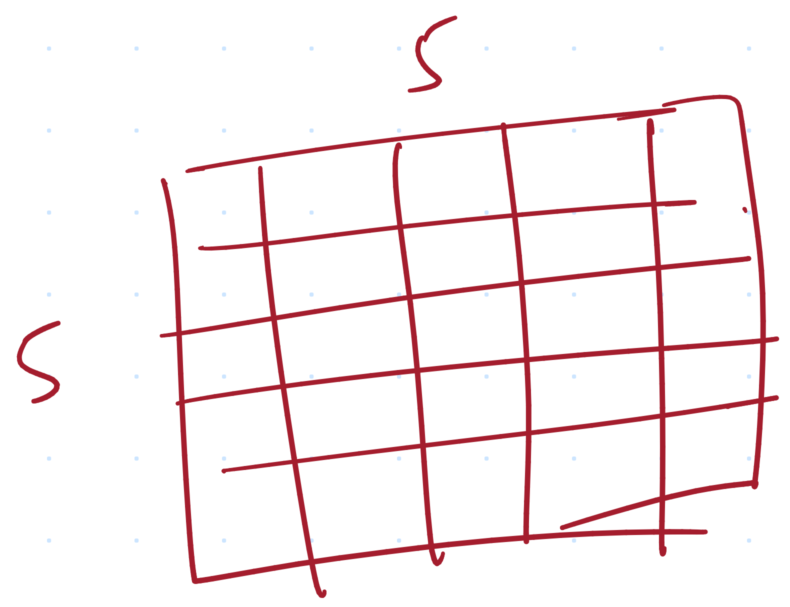
CNN → mainly used to drastically reduce the size of the neural network.



kernel → # params

$3 \times 3 + 1$ → 10

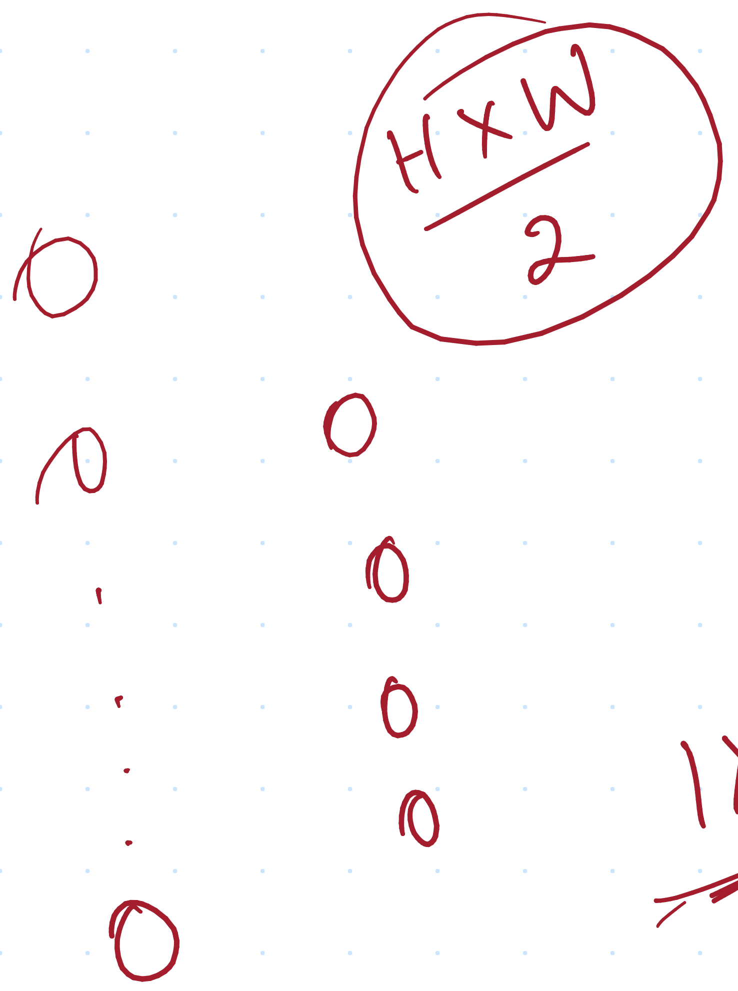
1 → bias.



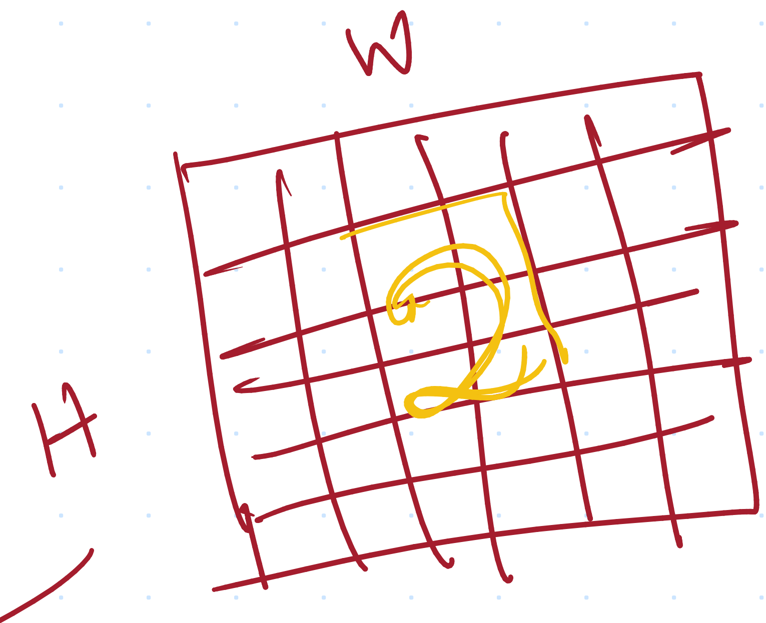
kernel → # params

$5 \times 5 + 1$ → 26

1 → bias.



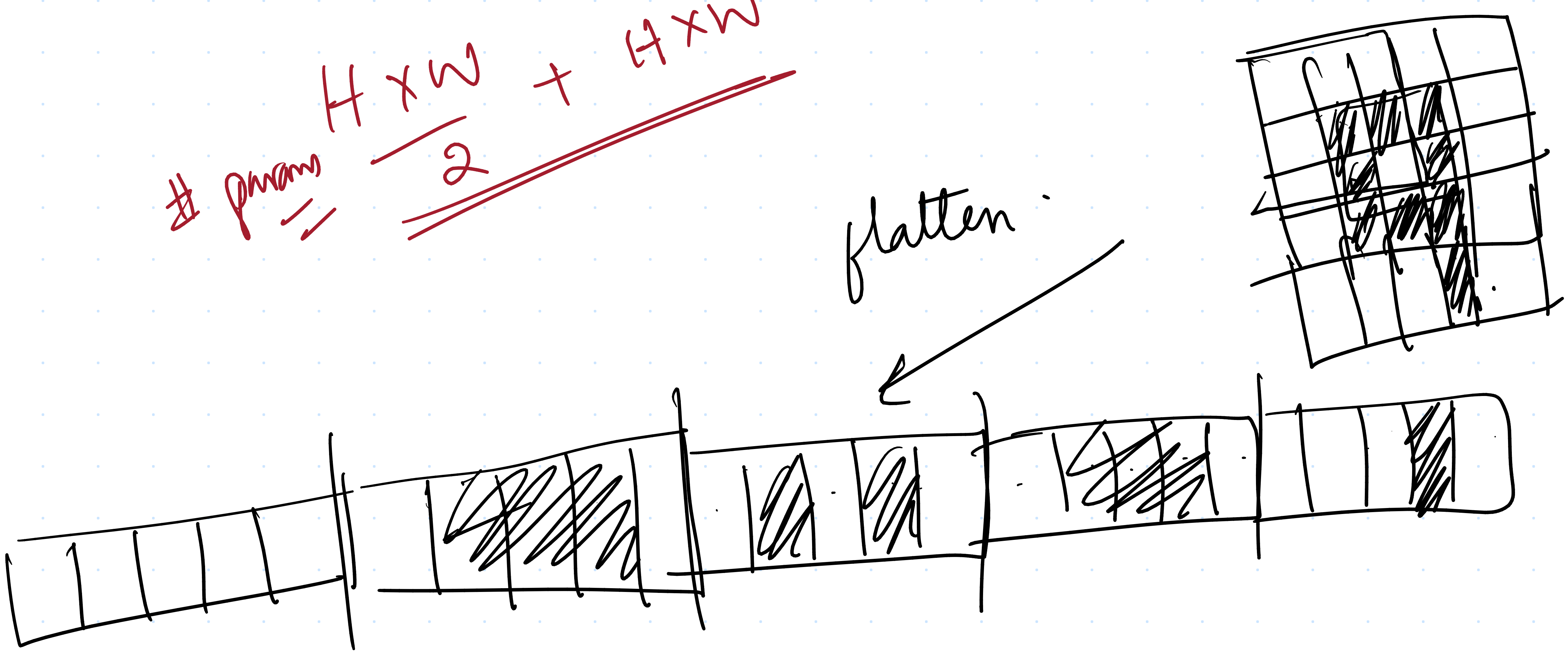
flatten



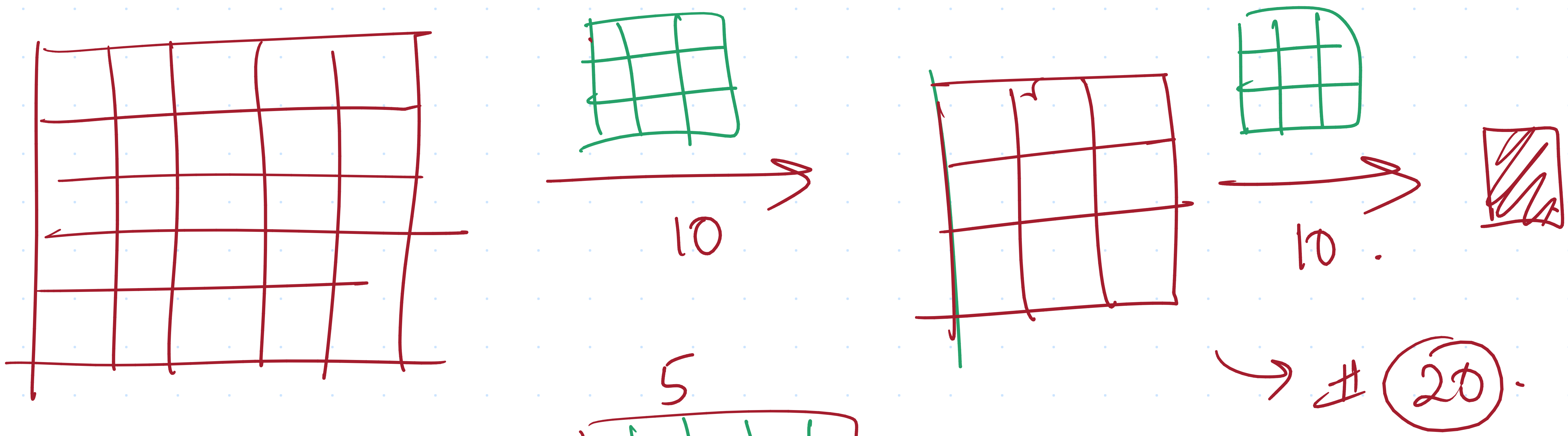
1×1×H×W

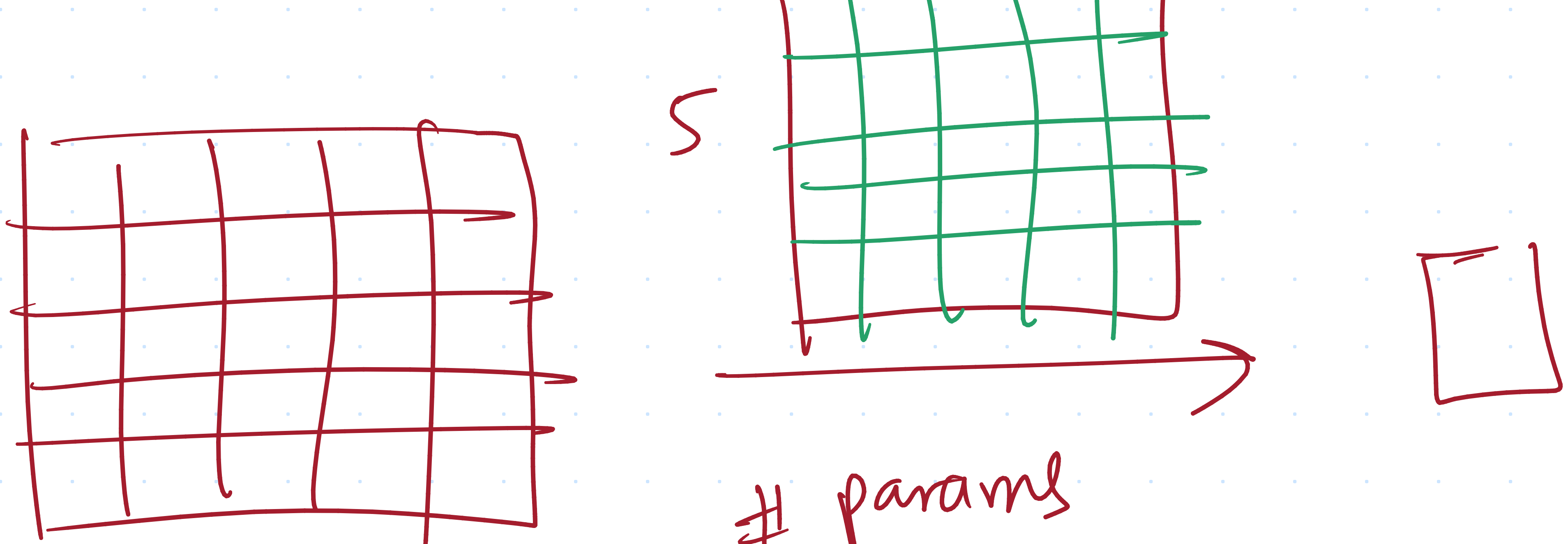
FCL / Dense Layer

params = $\frac{H \times W}{2} + H \times W$



Receptive field.



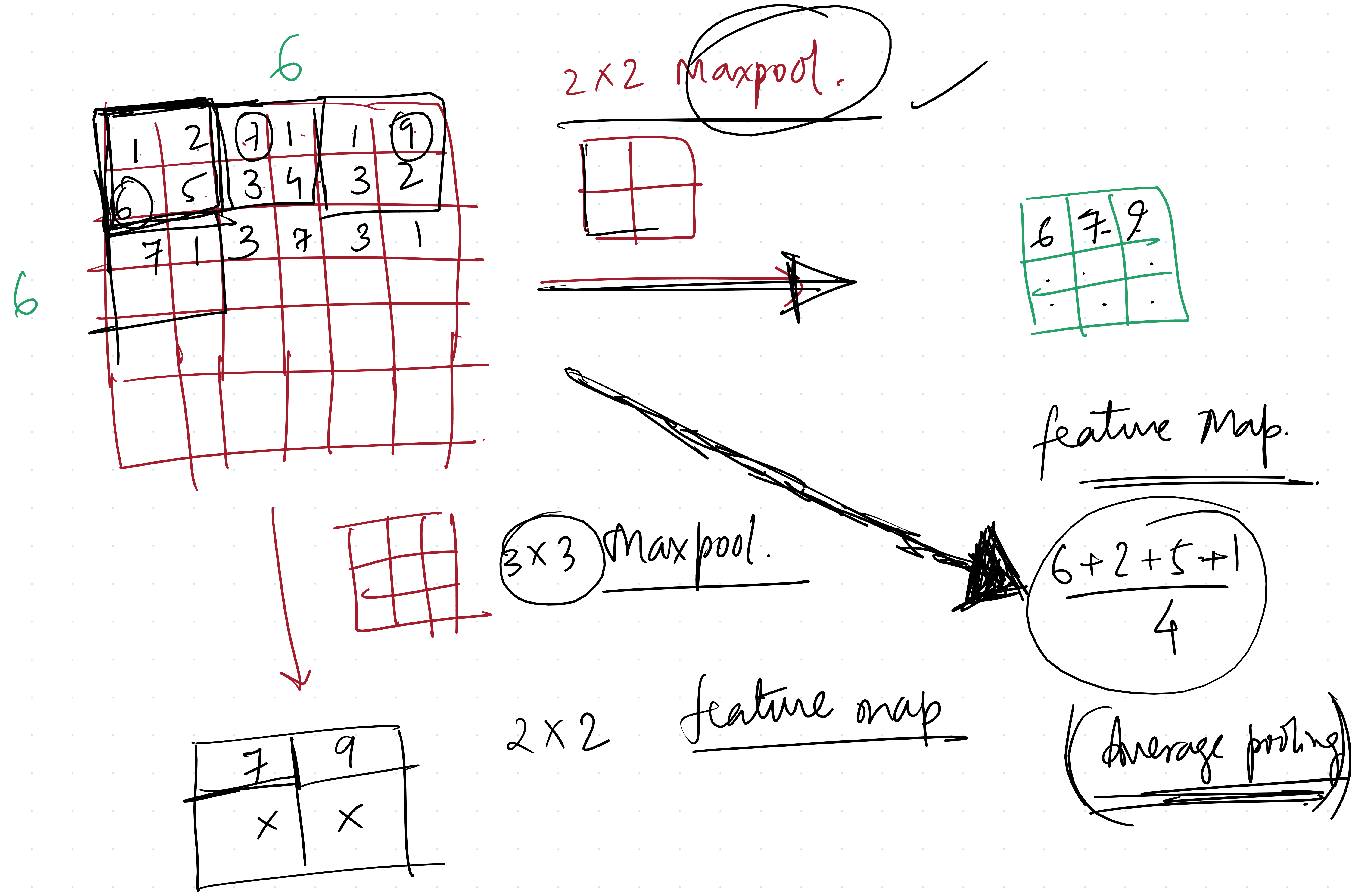


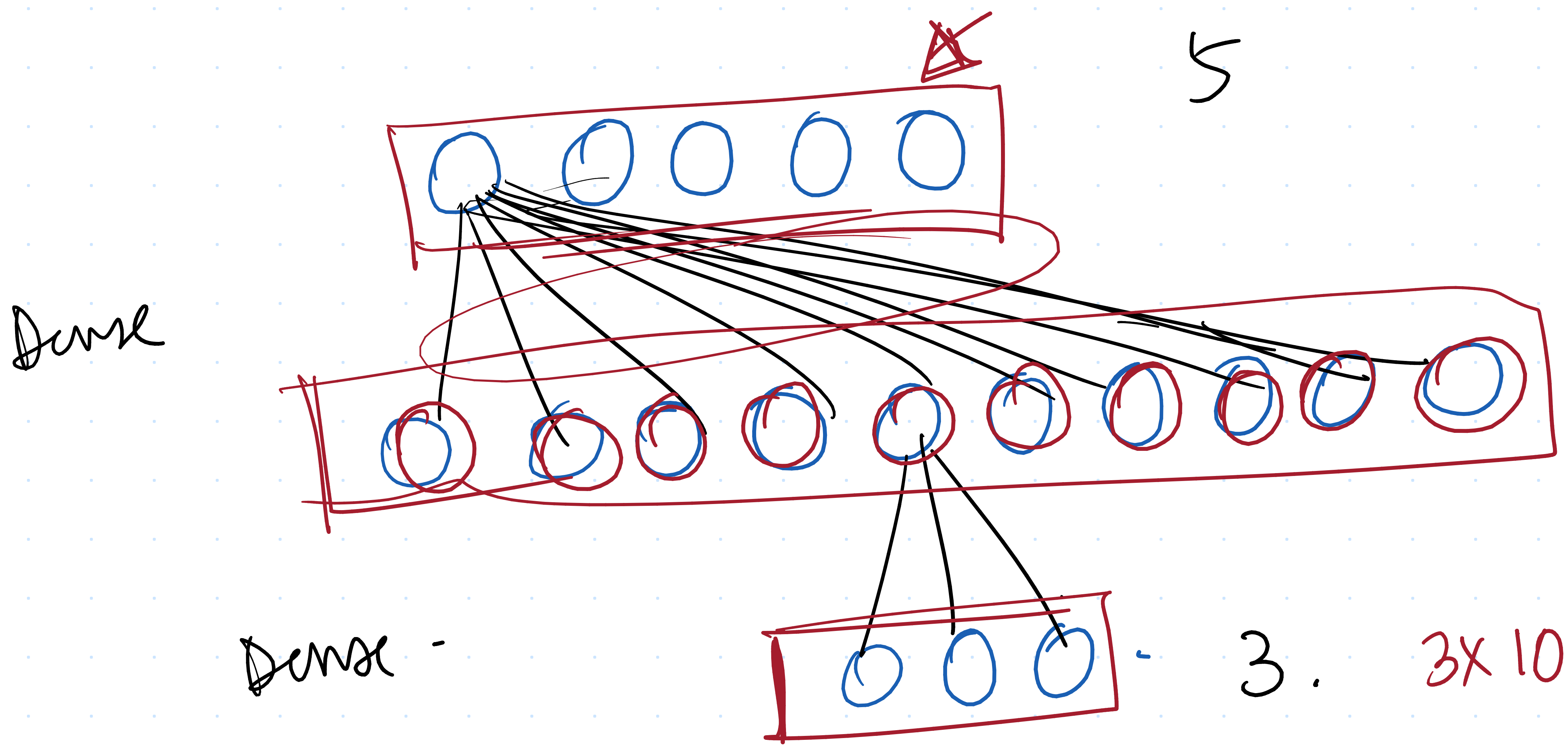
params

$5 \times 5 + 1 \Rightarrow 26.$

^.

26

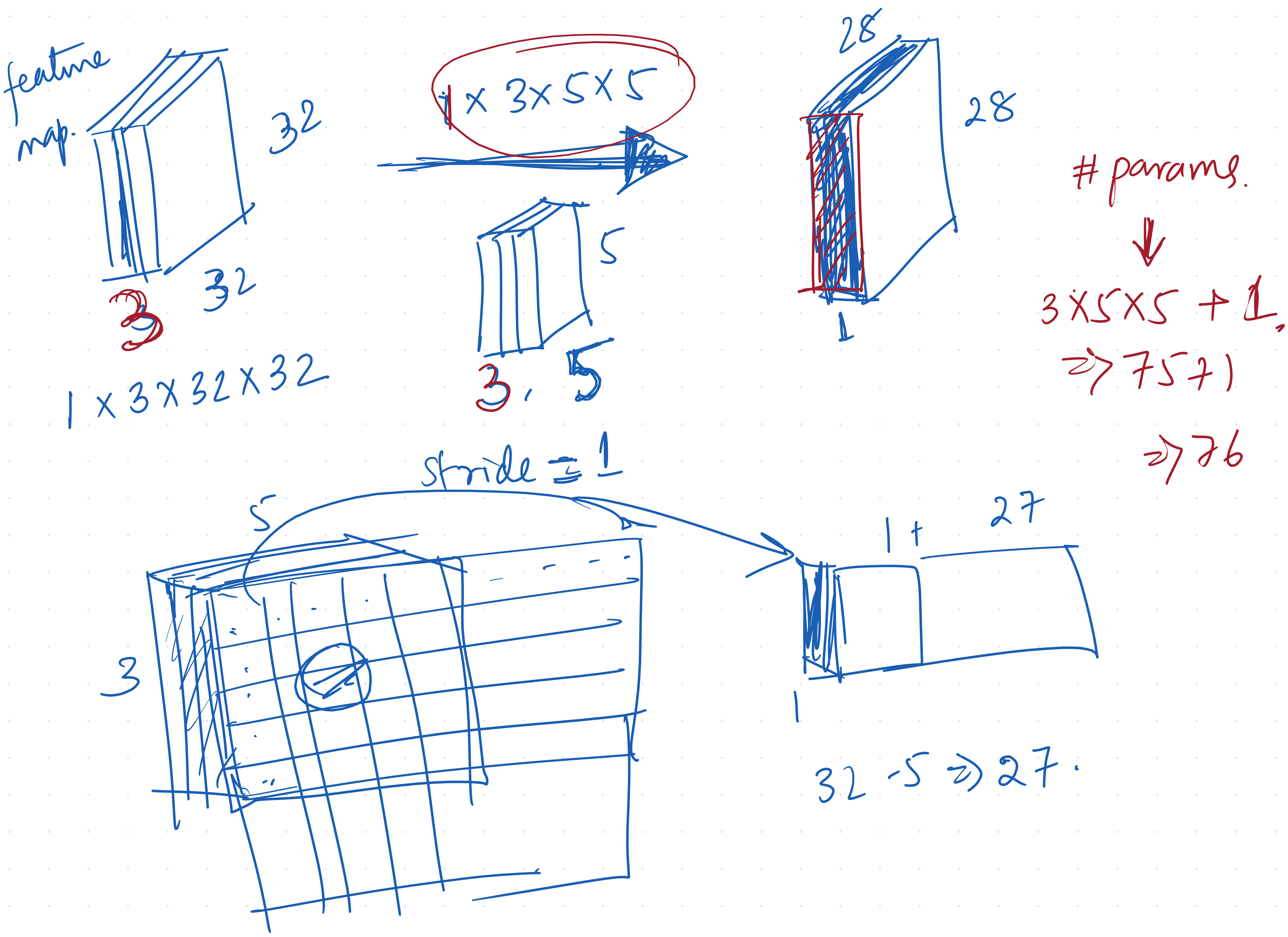


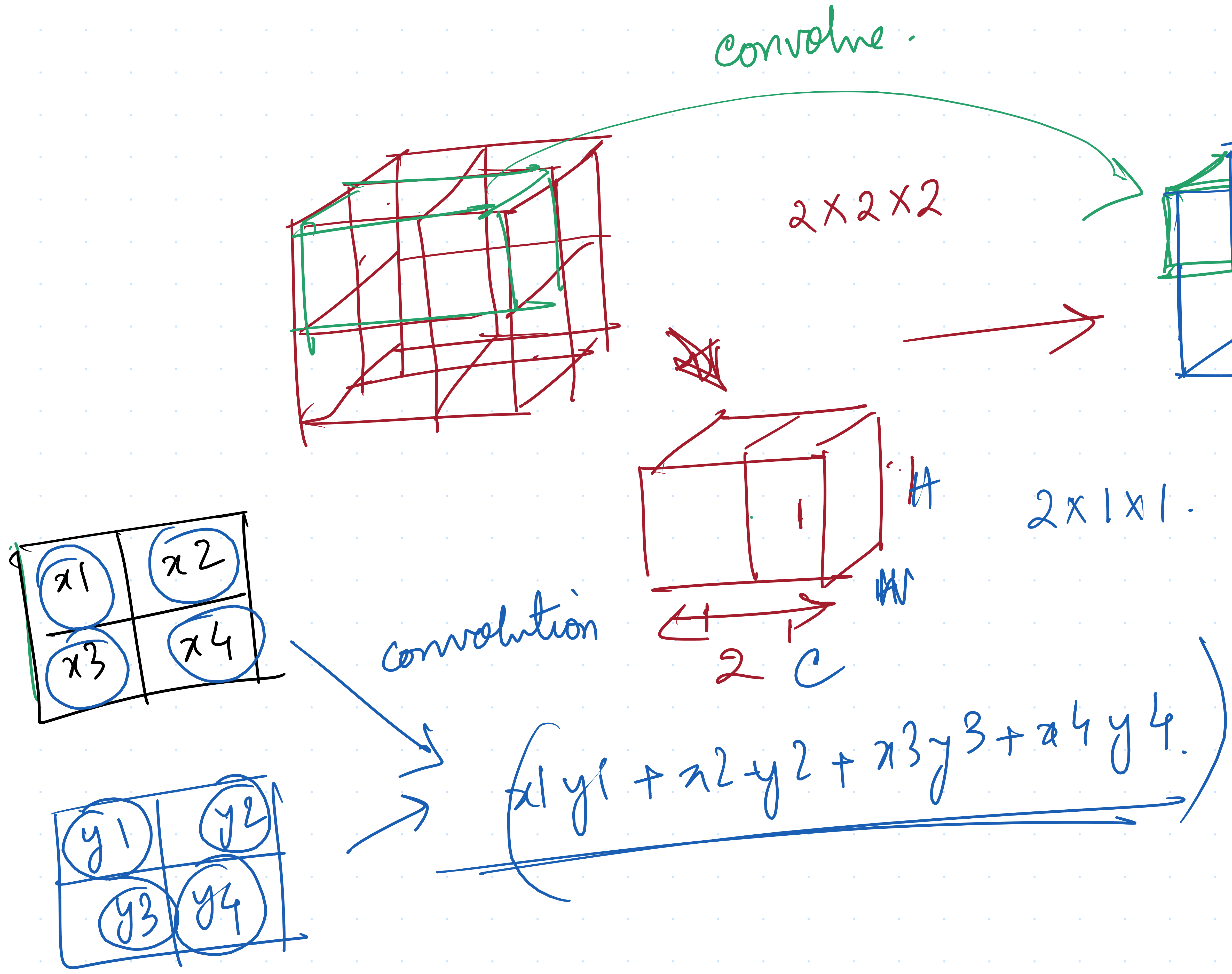


Sigmoid Activation

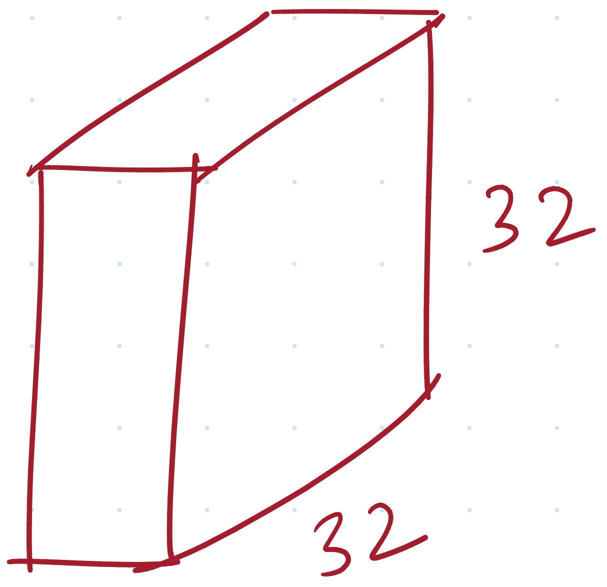
60
33
93

parameter = ?


feature map.
32
32
3
1 x 3 x 32 x 32
1 x 3 x 5 x 5
5
3, 5
28
28
1
params.
3 x 5 x 5 + 1.
=> 75 + 1
=> 76
stride = 1
5
3
1 +
27
1
32 - 5 => 27.


convolve.
2 x 2 x 2
1
H
W
2 C
2 x 1 x 1.
x1
x2
x3
x4
y1
y2
y3
y4
convolution
x1 y1 + x2 y2 + x3 y3 + x4 y4.

Convolution, correlation & filtering → Difference



bs × #C × H × W.

o.s. ic kh kw

6 × 3 × 5 × 5 →

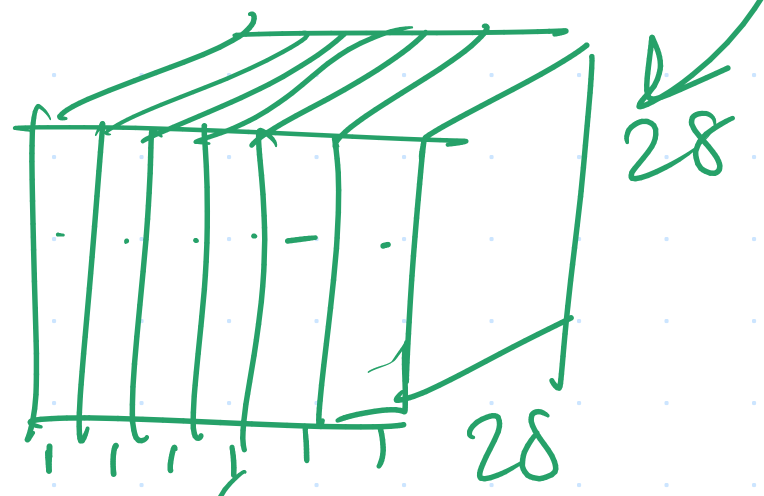
? (output ?)

(# parameters = ?)

(6×3×5×5 + 6)

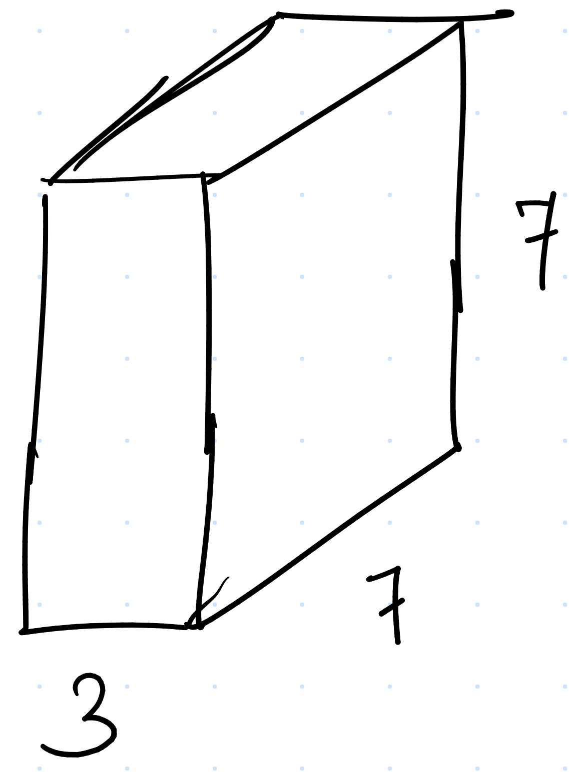
456

6 × 1×28×28.



6 × 28 × 28

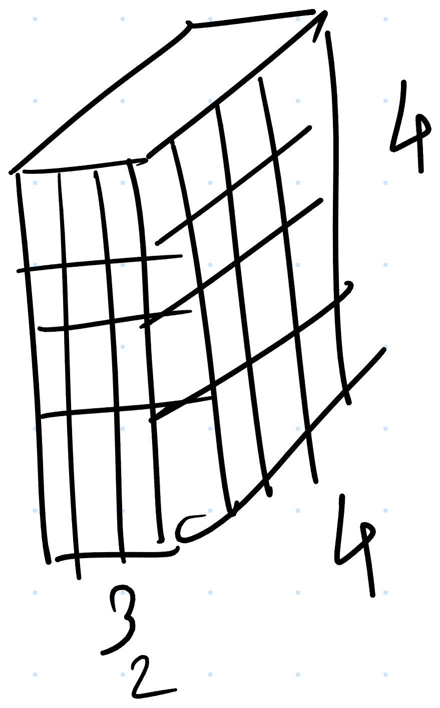
CNN Model Param Calculation.



3x3x3x3

stride = 2

padding = 1



feature Map.

#param

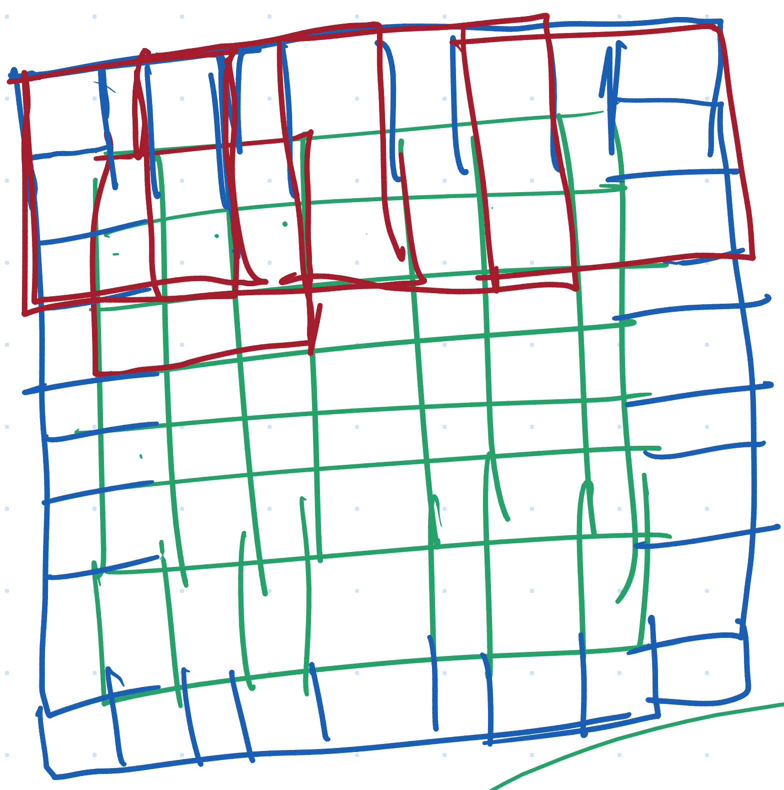
⇒ 3x3x3x3

+3

2
27
x3
81

⇒ 81 + 3 = 84

#



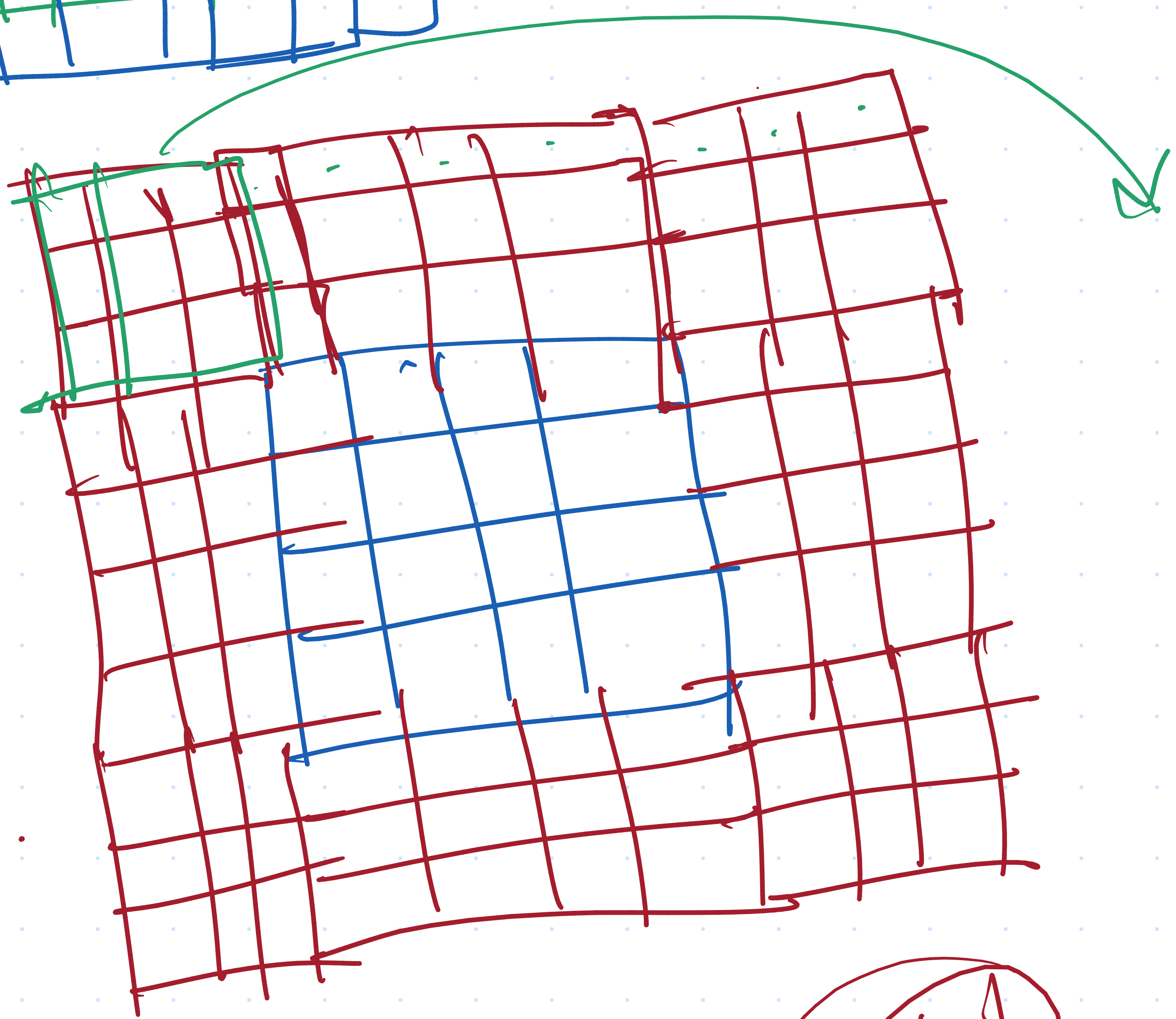
3x1x4x4.

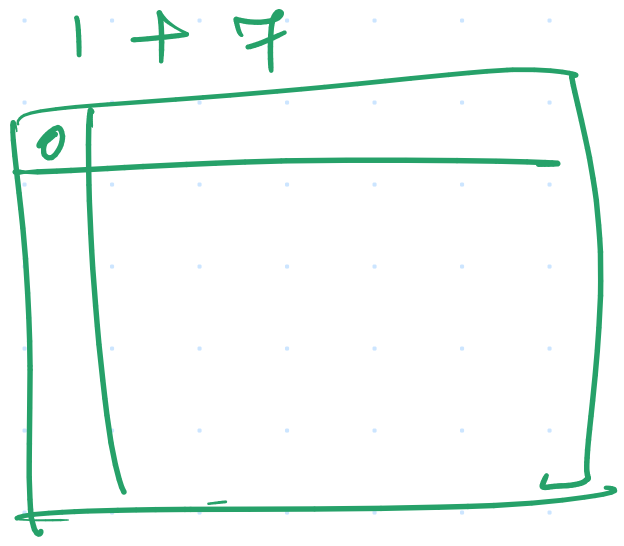
padding = 3.

stride = 1

3x3x3x3
o.s. i.c. h w

kernel.

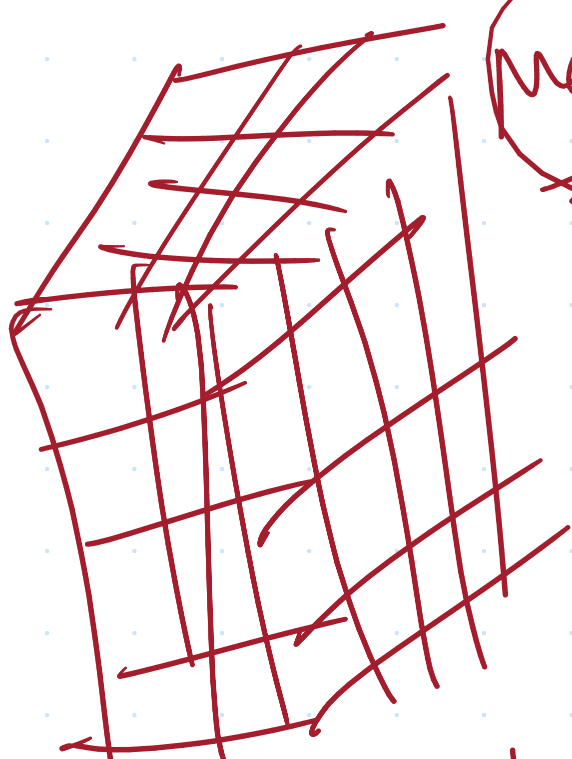




84

n-M
0 → 3
n 3
3.

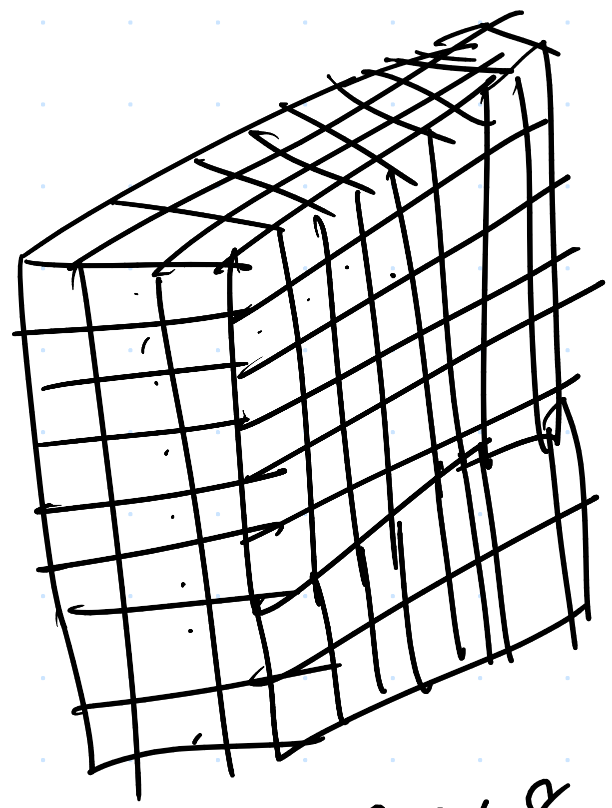
1x3x8x8



maxpool.

bw

(3+3)

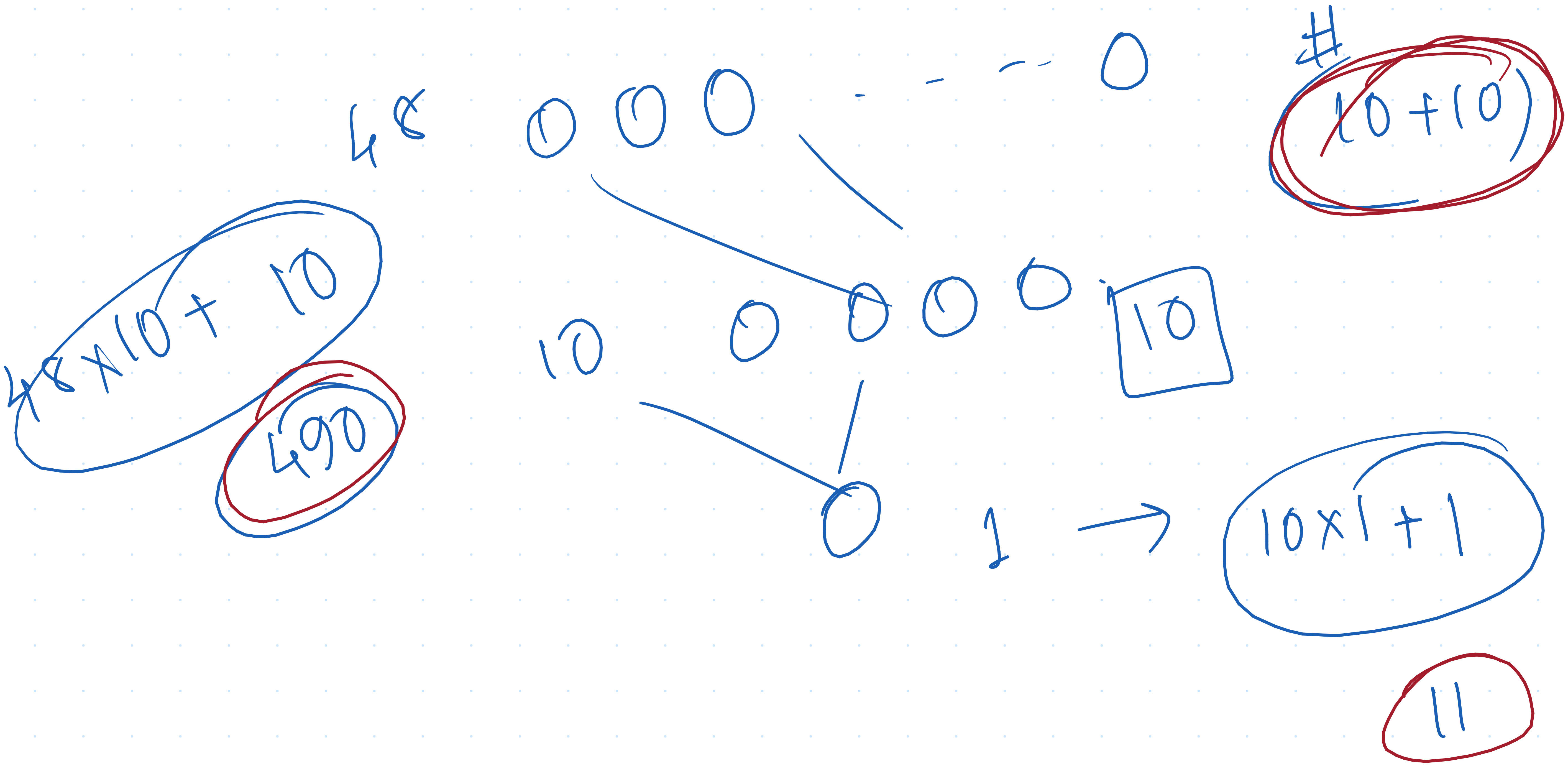


1x3x8x8

3x4x4.

48

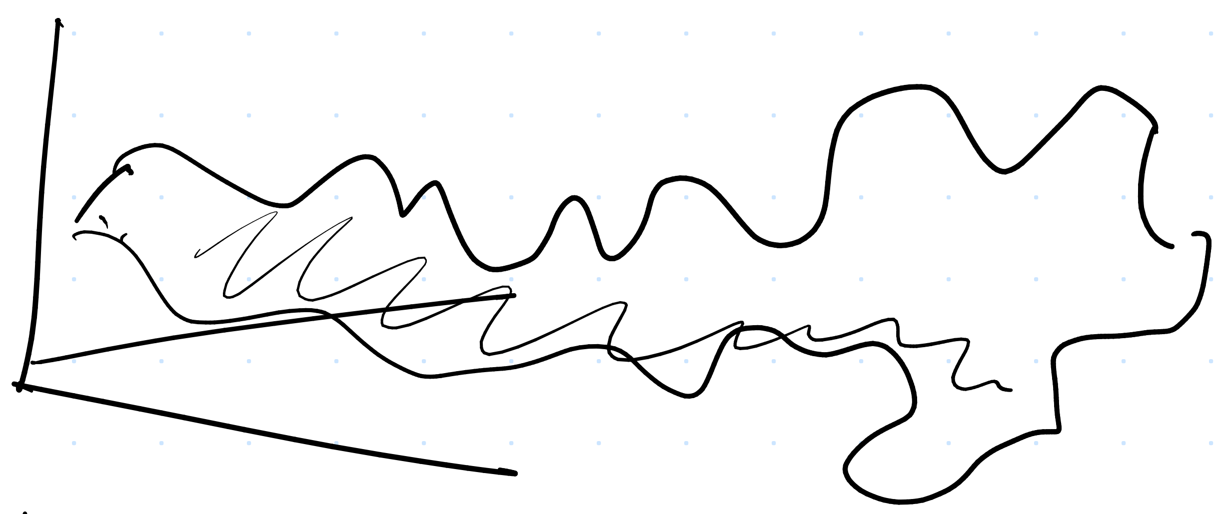
1x3x4x4

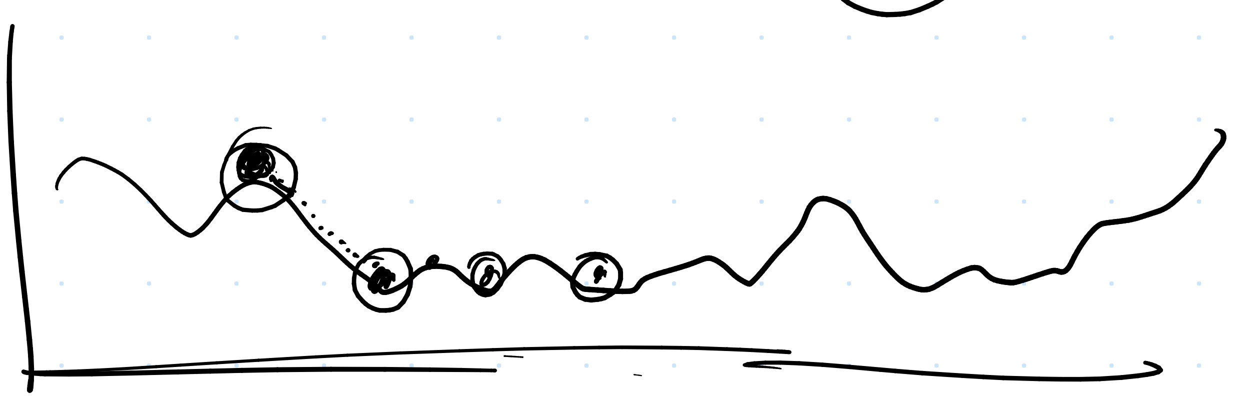


Backprop & Gradient Descent

Greedy Algorithm

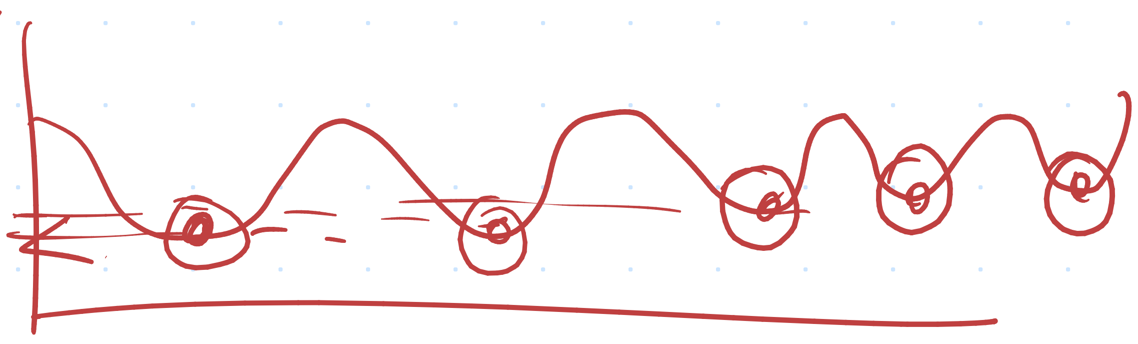
It always moves in the direction of reducing errors.





In N.N. one of the local optima is as good as the global optima

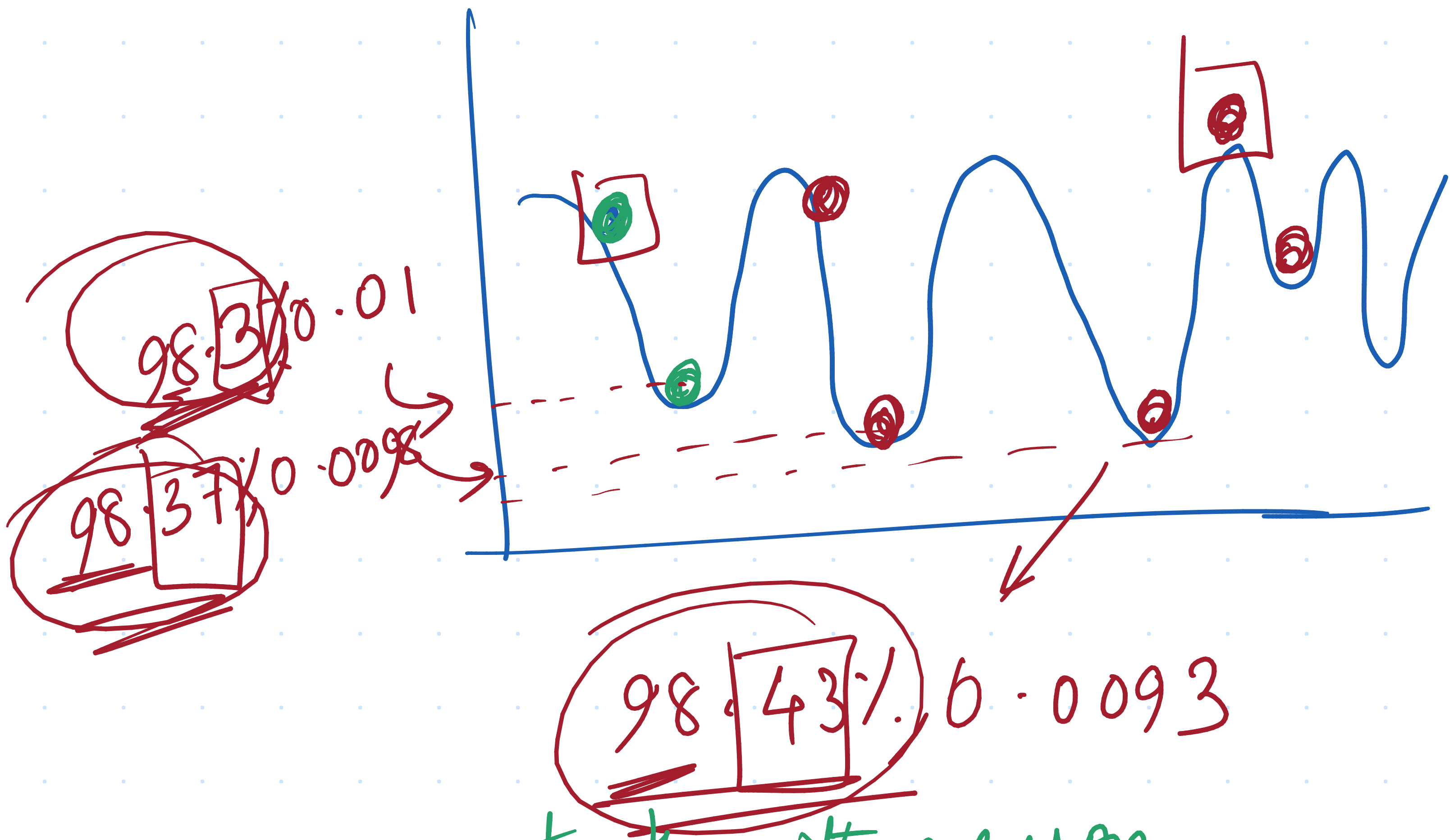
initialization will matter if you are using Gradient descent



Adam → ⋮ → One of the local optima → which is as good as global optima.

Weight update rule to minimize errors in loss functions.

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}$$



η → Learning rate

E → Loss function

w_{ji} → weight of connection from the ith neuron to the jth neuron.

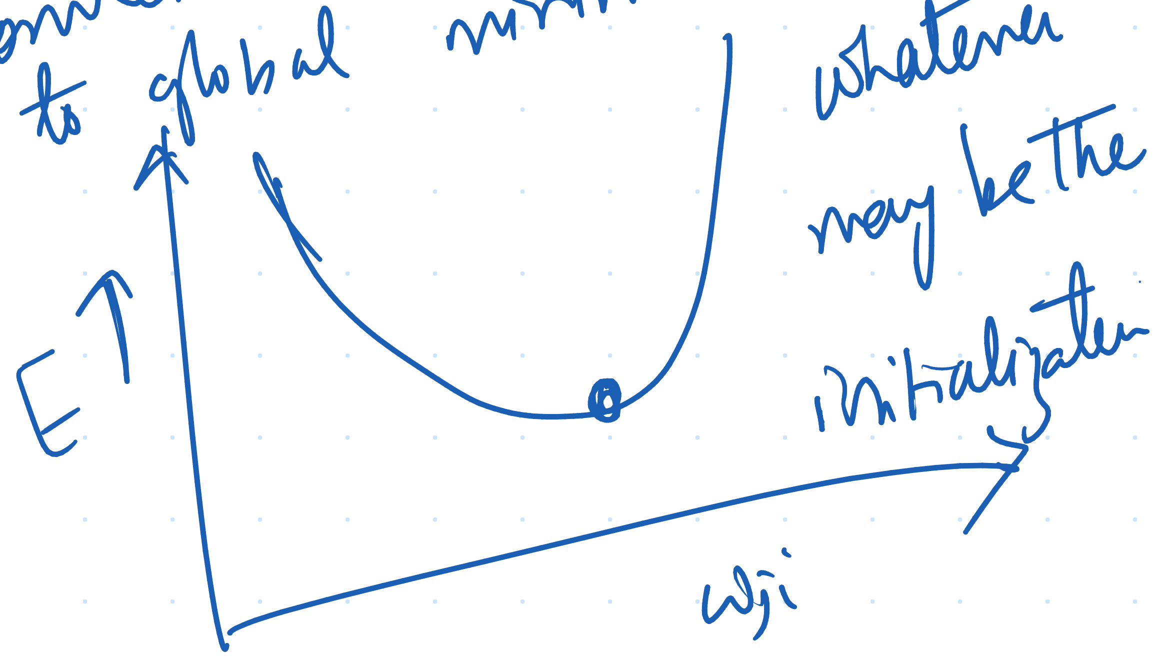
{98.3, 98.37, 98.43, ...} Always run for atleast 10-15 times & report the mean & standard deviation on the first dataset

{98.36 ± 0.07}

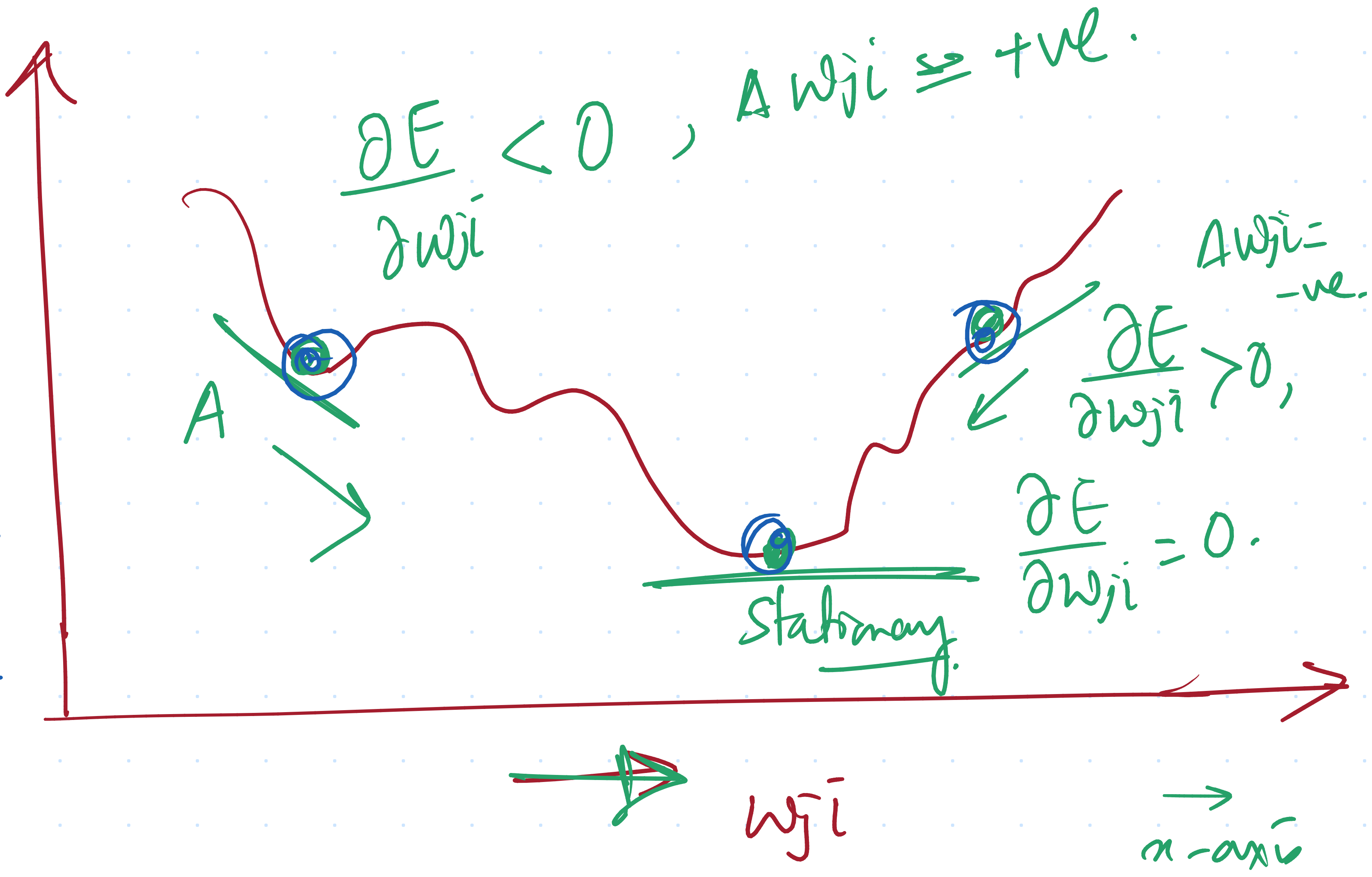
report

Linear Regression

Convex → Always converge to global minima



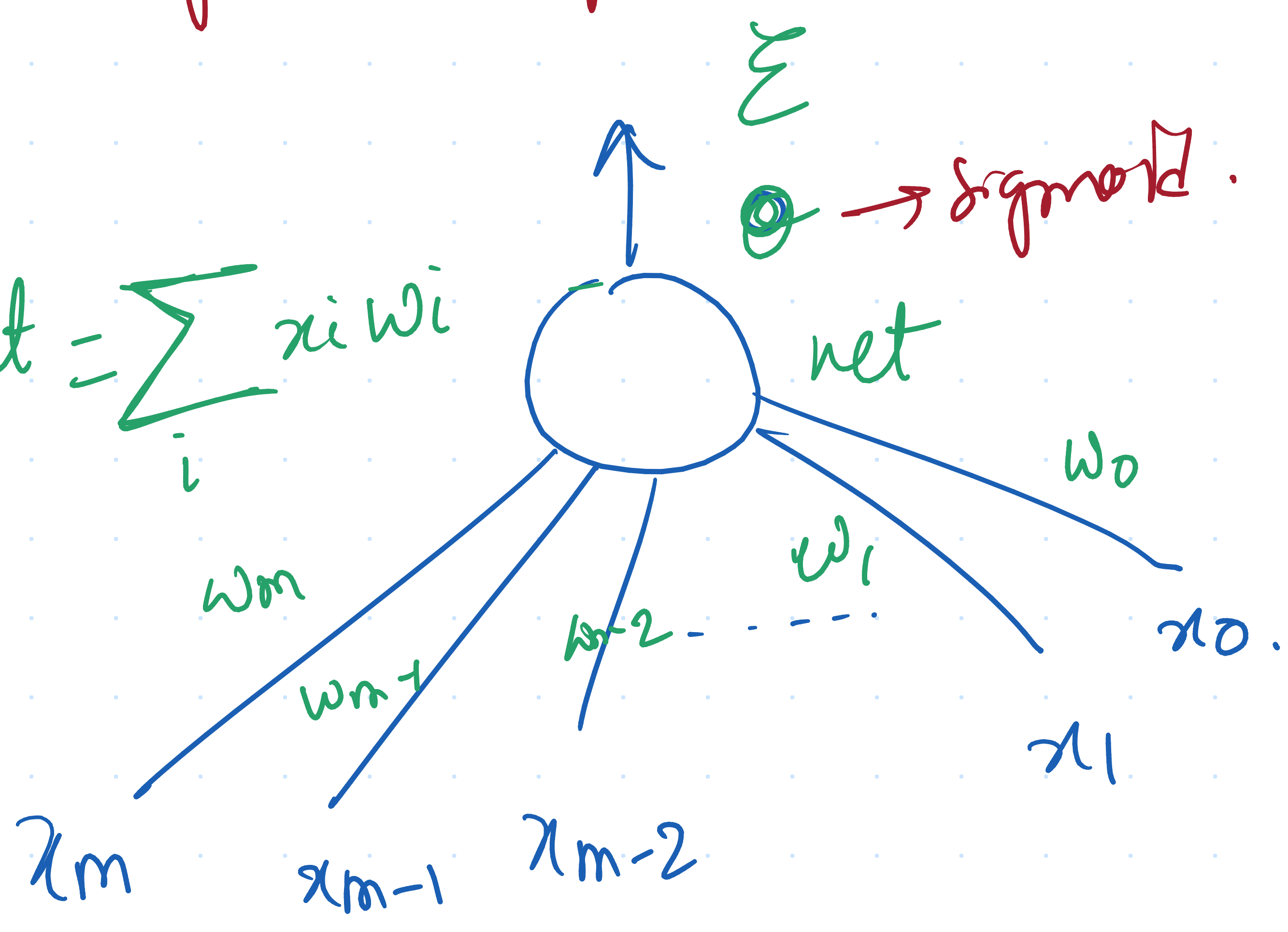
whatever may be the initialization



We will now derive the Single Sigmoid Neuron's cross entropy loss from a single data point

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \theta} \cdot \frac{\partial \theta}{\partial net} \cdot \frac{\partial net}{\partial w_1}$$

$$net = \sum_i x_i w_i$$



$$E = -t \log \theta - (1-t) \log(1-\theta)$$

cross entropy → binary

$$\frac{\partial E}{\partial \theta} = -\frac{t}{\theta} - \frac{(1-t)}{(1-\theta)}(-1)$$

('Small θ')

$$= \frac{-t(1-\theta) + (1-t)\theta}{\theta(1-\theta)} = \frac{-t + t\theta + \theta - t\theta}{\theta(1-\theta)}$$

$$= \frac{-t+\theta}{\theta(1-\theta)}$$

$$\boxed{\frac{\partial E}{\partial o} = -\frac{(t-o)}{o(1-o)}}$$

$$\boxed{\frac{\partial \theta}{\partial net} = \theta(1-\theta)}$$

$$\boxed{\frac{\partial net}{\partial w_1} = x_1}$$

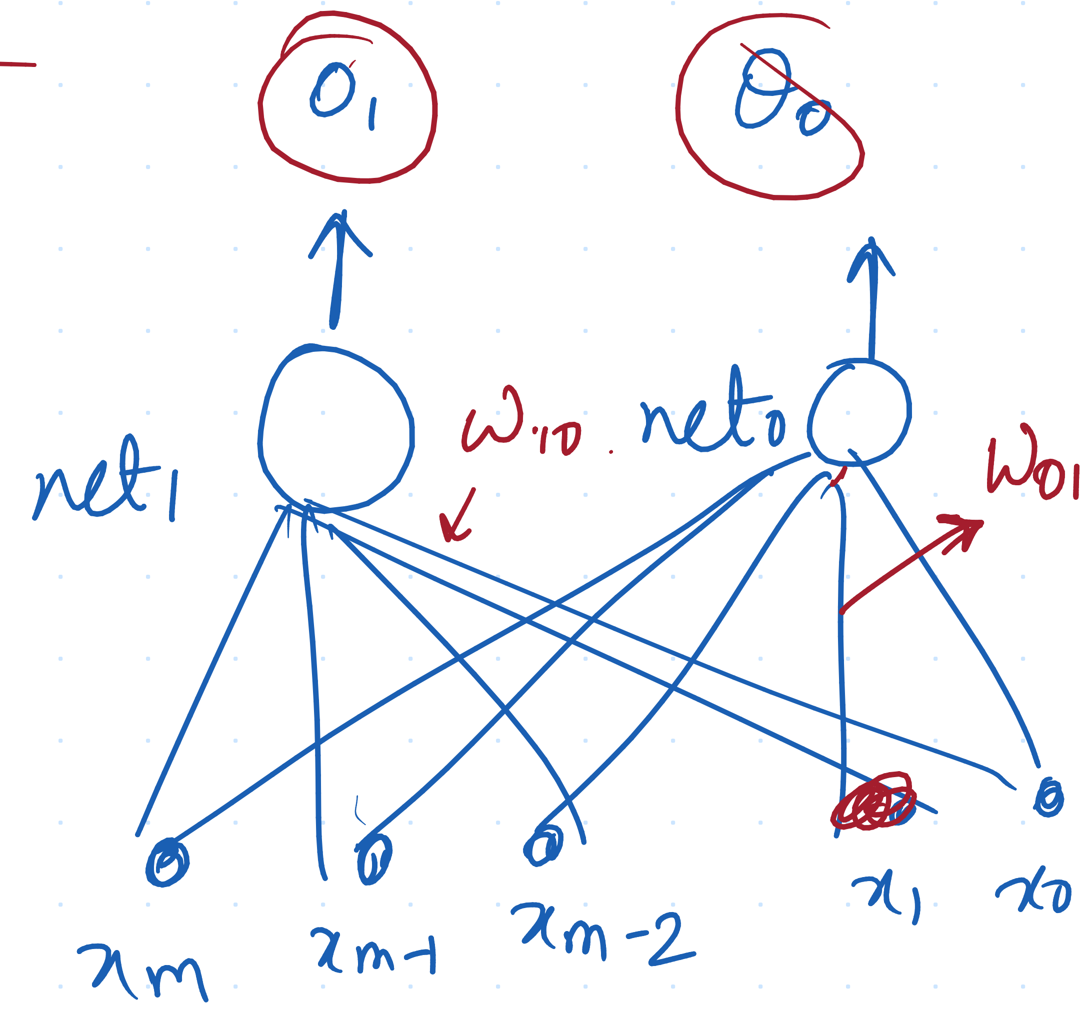
$$\frac{\partial E}{\partial w_1} = -\frac{(t-o)}{o(1-\theta)} \theta(1-\theta) \cdot x_1 = -(t-o)x_1$$

$$[\,-(t-o)x_m,\ -(t-o)x_{m-1},\ \ldots\ -(t-o)x_1,\ -(t-o)x_0\,]$$

$$\Delta w_1 = -\eta \frac{\partial E}{\partial w_1} = \eta (t-o) x_1$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta (t-o) x_i$$ ⟹ Derivative

Multiple Neurons → in output layer → Softmax & Cross Entropy loss → illustrated with 2 neurons.



$$O = \langle o_1, o_0 \rangle$$

$$NET = \langle net_1, net_0 \rangle$$

$$Softmax_i = \frac{e^{net_i}}{\sum_j e^{net_j}}$$

$$o_1 = \frac{e^{net_1}}{e^{net_0} + e^{net_1}} \qquad o_0 = \frac{e^{net_0}}{e^{net_0} + e^{net_1}}$$

jacobian.

$$\frac{\partial O}{\partial Net} = \begin{bmatrix} \frac{\partial o_0}{\partial net_0} & \frac{\partial o_1}{\partial net_0} \\ \frac{\partial o_0}{\partial net_1} & \frac{\partial o_1}{\partial net_1} \end{bmatrix} = \begin{bmatrix} o_0(1-o_0) & -o_0 o_1 \\ -o_1 o_0 & o_1(1-o_1) \end{bmatrix}$$

Refer to the derivative of softmax function.

for same → $o_i(1-o_i)$

for diff → $-o_i o_j$

Jacobian – first order partial derivative.

Hessian → 2nd order partial derivative.

m-tensor → Higher order partial derivative.

3-tensor, 4-tensor . . . .

→ Binary cross entropy

$$E = -t_1 \log o_1 - (1-t_1) \log(1-o_1)$$

$$\frac{\partial E}{\partial w_{11}} = -\frac{t_1}{o_1}\frac{\partial o_1}{\partial w_{11}} - \frac{t_0}{o_0}\frac{\partial o_0}{\partial w_{11}}$$

$$\frac{\partial o_1}{\partial w_{11}} = \frac{\partial o_1}{\partial net_1}\cdot\frac{\partial net_1}{\partial w_{11}} + \frac{\partial o_1}{\partial net_0}\cdot\frac{\partial net_0}{\partial w_{11}} \rightarrow 0.$$

$$= o_1(1-o_1)\cdot x_1 + -o_1 o_0 \, 0$$

$$= o_1(1-o_1)\cdot x_1.$$

$$\frac{\partial o_0}{\partial w_{11}} = \frac{\partial o_0}{\partial net_1} \cdot \frac{\partial net_1}{\partial w_{11}} + \frac{\partial o_0}{\partial net_0} \cdot \cancel{\frac{\partial net_0}{\partial w_{11}}} \rightarrow 0.$$

$$= -o_0 o_1 \; x_1 + \cancel{-o_0(1-o_0) \times 0} \rightarrow 0.$$

$$\frac{\partial E}{\partial w_{11}} = -\frac{t_1}{o_1} \cdot \frac{\partial o_1}{\partial w_{11}} - \frac{t_0}{o_0} \cdot \frac{\partial o_0}{\partial w_{11}}$$

$$= -\frac{t_1}{\cancel{o_1}} \cdot \cancel{o_1}(1-o_1) \cdot x_1 - \frac{t_0}{\cancel{o_0}} (-\cancel{o_0}\, o_1) \cdot x_1$$

$$= -t_1(1-o_1) \cdot x_1 + t_0\, o_1\, x_1$$

$$= x_1(-t_1 + o_1) = -(t_1 - o_1) \cdot x_1$$

$$\boxed{\Delta w_{11} = -\eta \frac{\partial E}{\partial w_{11}} = \eta (t_1 - o_1) x_1}$$