Convolutional Nueral Networks:

History:

- Mark I Perceptron
  - ↳ Connected to a camera that produced 400 pixel image

$$f(x) = \begin{cases} 1, & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Update Rule:

$$w_i(t+1) = w_i(t) + \alpha \left(d_j - y_j(t)\right) x_{j,i}$$

- Adaline & Madaline:
  - ↳ Started to stack linear layer into multilayer perceptron
  - ↳ No backprop

- Rumelhart:
  - ↳ Backprop was introduced

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial O_{pj}} \times \frac{\partial O_{pj}}{\partial w_{ji}}$$

- Hinton & Sqlakhutdinov:
  - ↳ Showed how to train nn
  - ↳ Not really like today's nn
  - ↳ Required careful initialization

- Hubel & Wiesel:
  - ↳ Found that cells had hierarchical organization

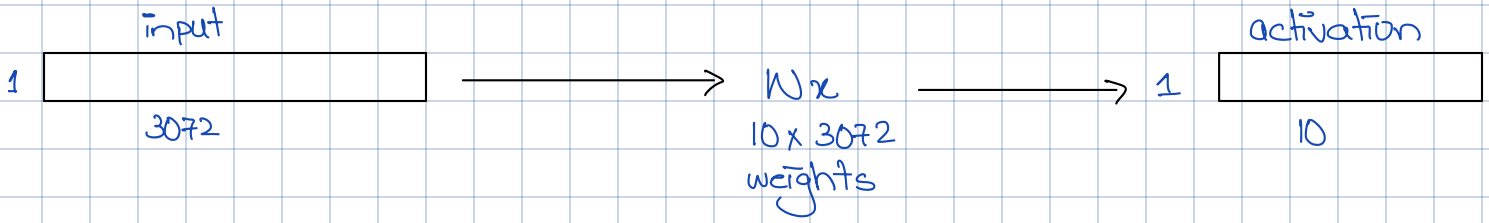
  - Retinal ganglion cell:
    - ↳ Respond to circular regions
  - Simple cells:
    - ↳ Responsive to oriented edges
  - Complex cells:
    - ↳ Reponsive to light orientation & movement
  - Hypercomplex cells:
    - ↳ Reponsive to movement with an end point (corners)

- Nueocognitron:
  - Simple cells: modifiable parametres
  - Complex cells: perform pooling

- Gradient based learning applied to document recognition:
  - ↳ Used backprop & Gradients to train CNNs
  - ↳ NOT used on complex data
  - ↳ AlexNet solved that

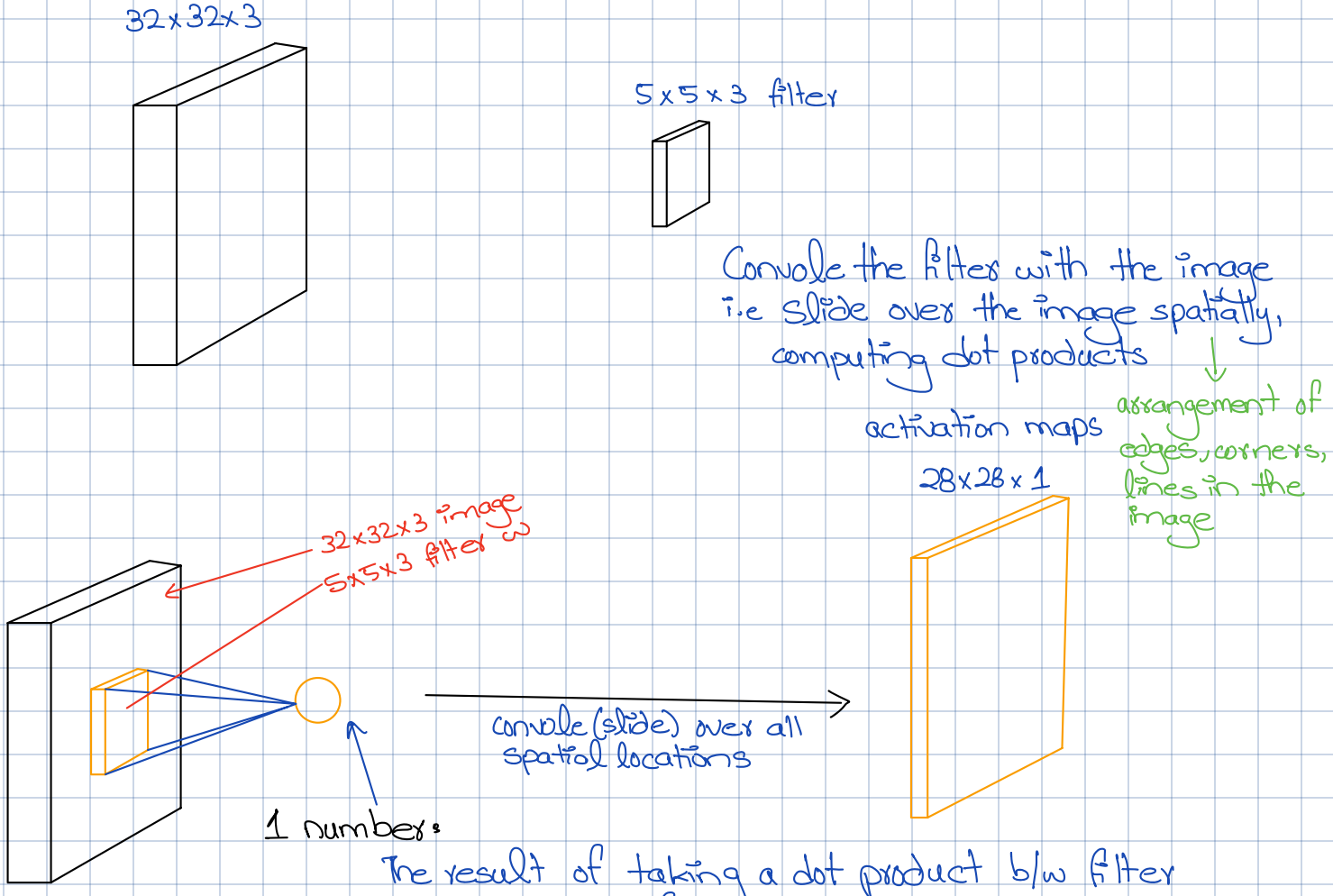
Fully connected layer:

↳ 32x32x3 image ⟶ Stretch to 3072 x 1

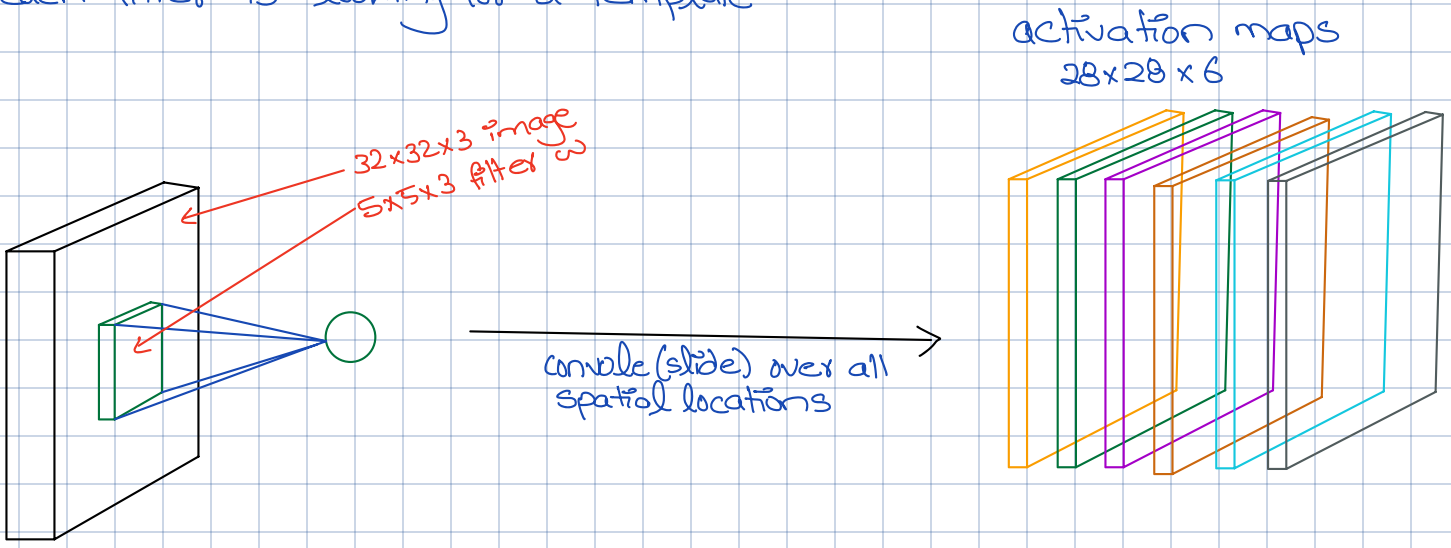


Convolution layer:

- 32x32x3 ⟶ preserve spatial structure

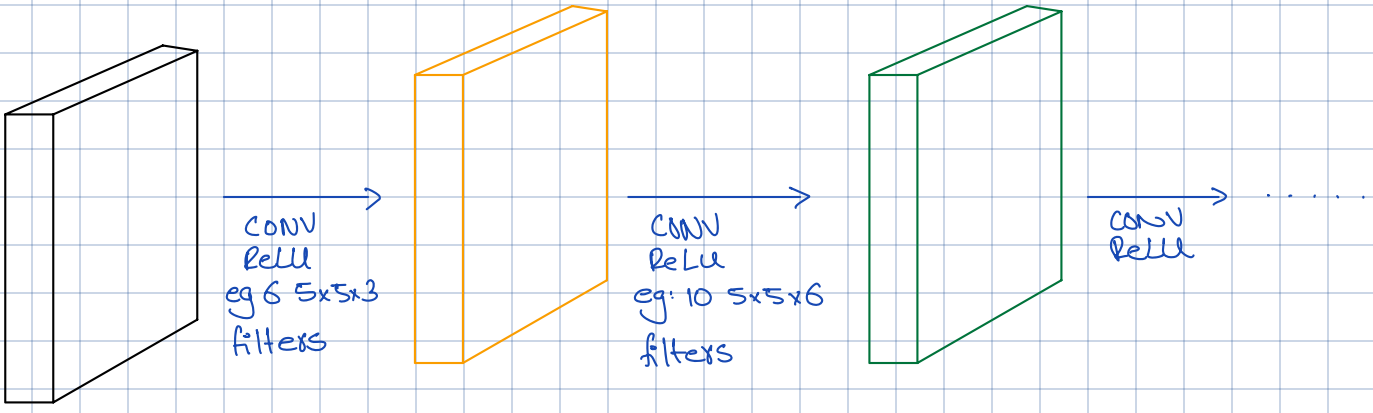


Consider a 6 filters 5x5x3, because we need to work with multiple filters, as each filter is looking for a template

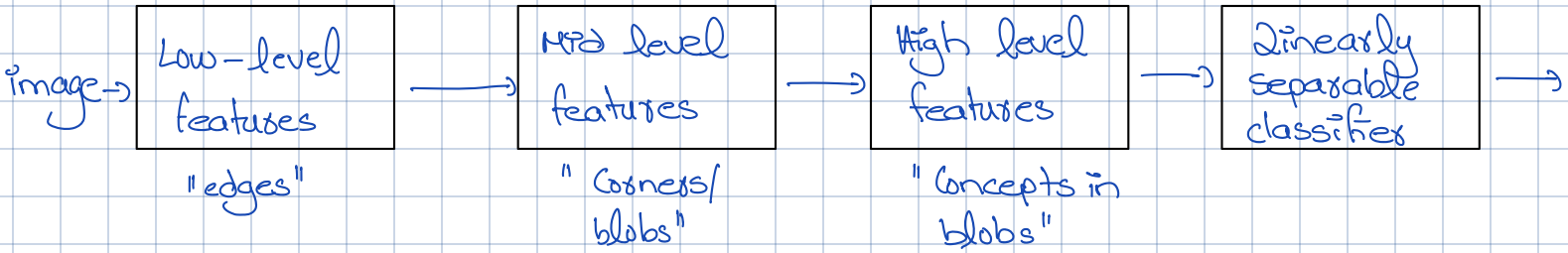


- We can stack these to get a new image of size 28x28x6!

ConvNet is a sequence of these convolution layers



In ConvNet, you end up learning this hierarchy of filters,



- We call layer convolution because it is related to convolution of two signals

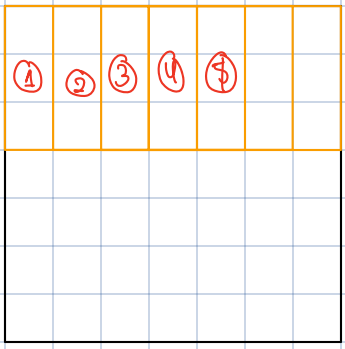
$$f[x_1, x_2] * g[x, y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x-n_1, y-n_2]$$

A closer look at spatial dimensions:

- How we get 28x28x1 activation map

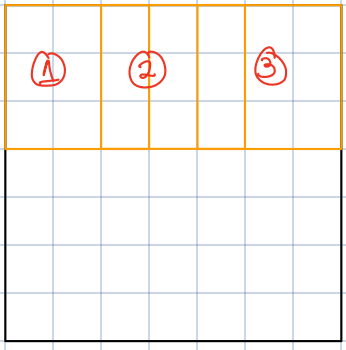
Consider 7x7 input, 3x3 filter & stride 1

↳ interval of slide



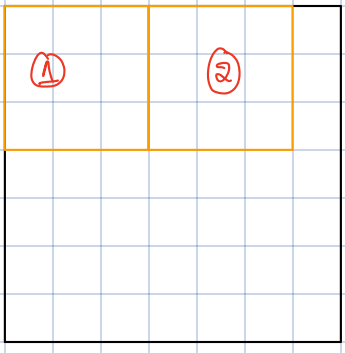
⇒ 5x5 output

Consider 7x7 input, 3x3 filter & stride 2



⇒ 3x3 output!

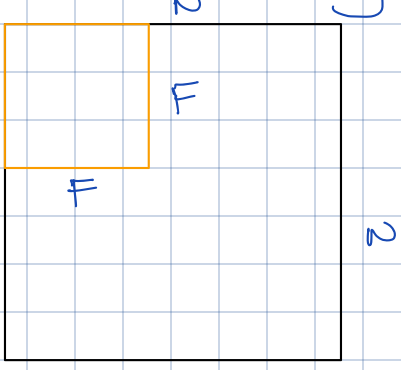
Consider 7x7 input, 3x3 filter & stride 3



⇒ Doesn't fit

So, we don't do this

Formula for checking how strides will work,



Output size:

$$\frac{(N-F)}{stride} + 1$$

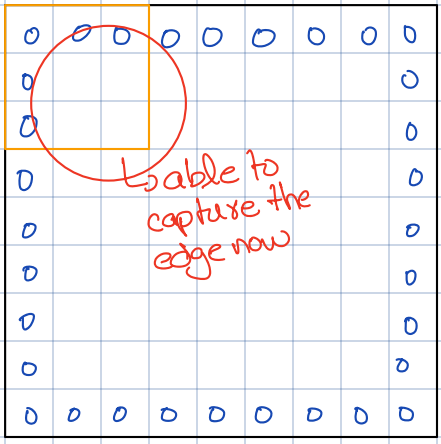
eg: $N=7$, $F=3$

stride 1: $(7-3)/1 + 1 = 5$

stride 2: $(7-3)/2 + 1 = 3$

stride 3: $(7-3)/3 + 1 = 2.33$ ↳ oops!

In practice, it is common to zero pad the corners so that filters can fully overlap the image



Now, $\frac{N-F+2\times P}{stride}+1$ (Padding ↑)

- If we have multiple convolution layers without zero padding with 5x5 filters, the volume will shrinks very quickly.

Example,

Input : 32
Filters: 10 of 5x5

Output size: $\frac{32+2\times2-5}{1}+1=32$

An activation map is a 28x28 sheet of nueron outputs:
1. Each is connected to a small region in the input
2. All of them share parametres

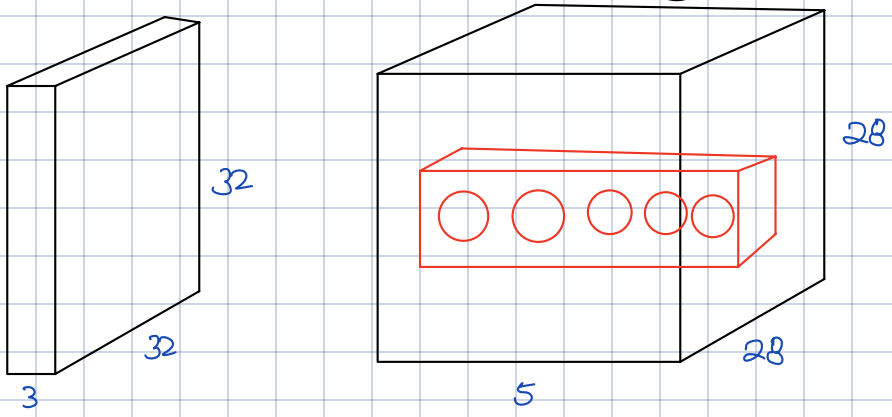
32x32x10

Number of params: $5\times5\times3+1=76$ (Filter ↑, b ↑)
$76\times10=760$

Hyperparametres:
1) Number of filters k (Powers of 2)
2) Their spatial extent F
3) The strides
4) The amount of zero padding P

Kernel = filters = receptive field

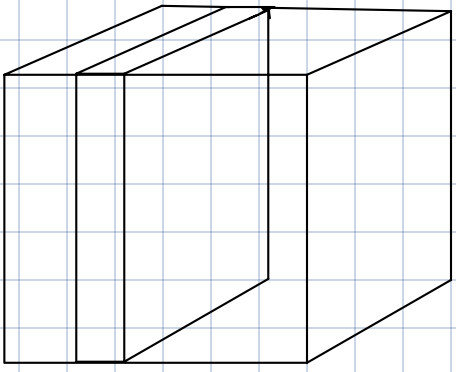
The brain/Nueron view of CONV layer:



The five nueorns are all looking at the same region but for different things

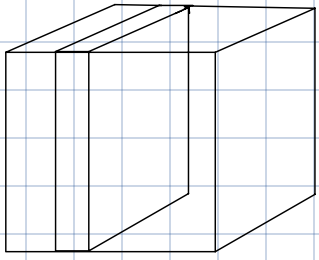
Pooling layer:

↳ Makes the representation smaller and more manageable
↳ Operates over each activation map idependently



224 x 224 x 64

Pool →



112 x 112 x 64

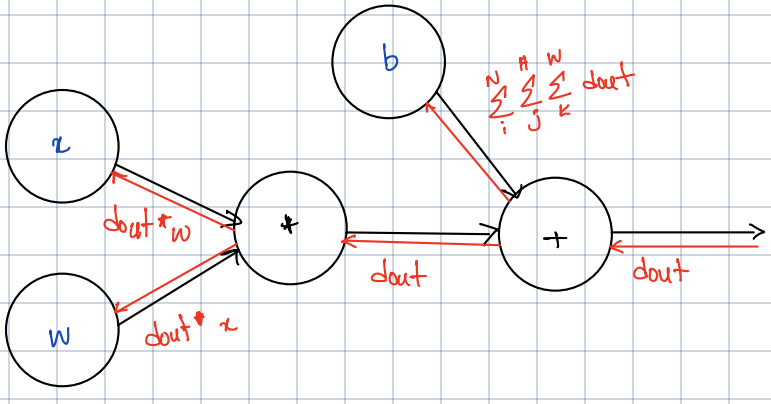
Max pooling:

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

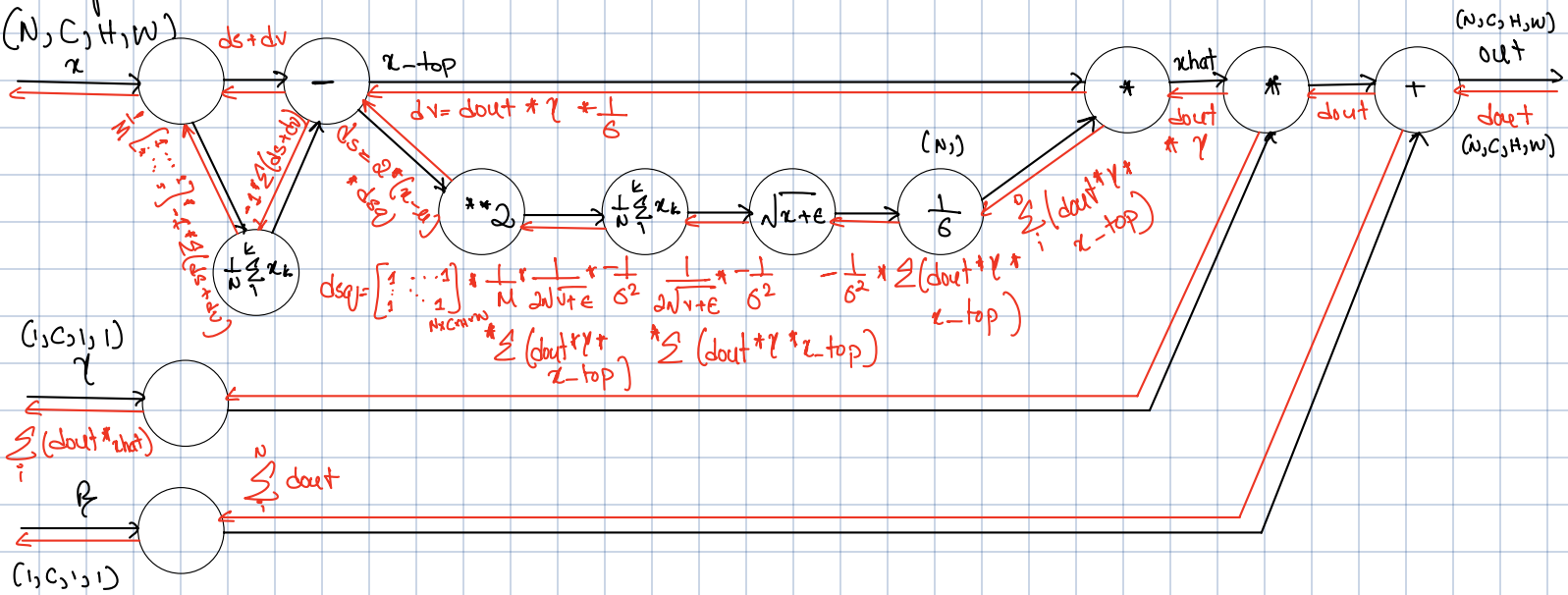
max pool with 2x2 filters
& stride 2 →

6	8
3	4

Assignment helper:



Group norm: $M = N * H * W$



$$dx = ds + dv - \frac{1}{M}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} * \sum (ds + dv)$$