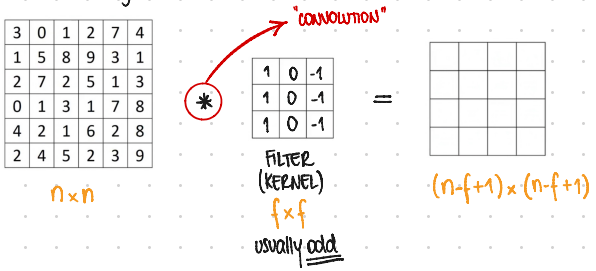
CONV NETS

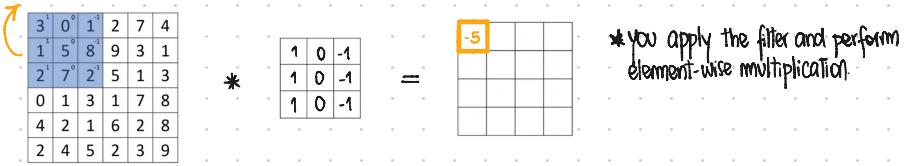
COURSE IV

EDGE DETECTION EXAMPLE

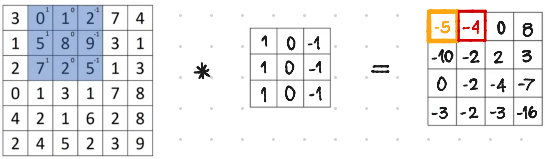
* vertical edge detection:

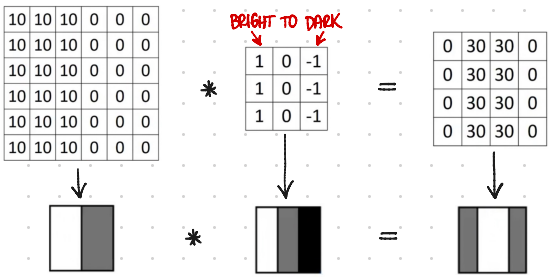


$3\times1 + 1\times1 + 2\times1 + 0\times0 + 0\times5 + 0\times7 + (-1)1 + (-1)8 + (-1)2 = -5$



* you apply the filter and perform element-wise multiplication





* horizontal edge detection:

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

$*$

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

$=$

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

* what filter is the best filter?

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

SOBEL FILTER: it gives a heigher weight to the middle row and makes it more robust.

3	0	-3
10	0	-10
3	0	-3

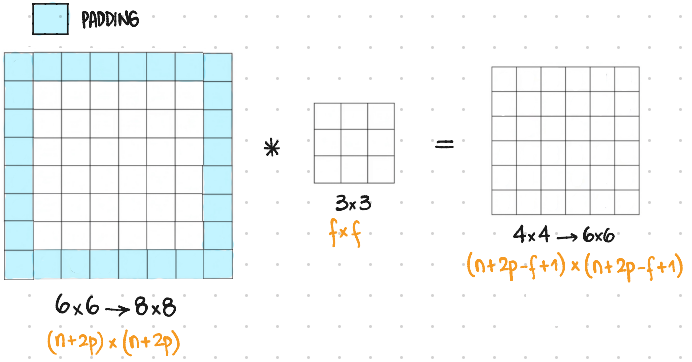
SCHARR FILTER

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

* you can treat each element of the filter as a parameter and use backprop to calculate them.

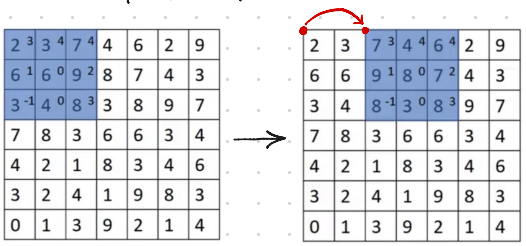
PADDING

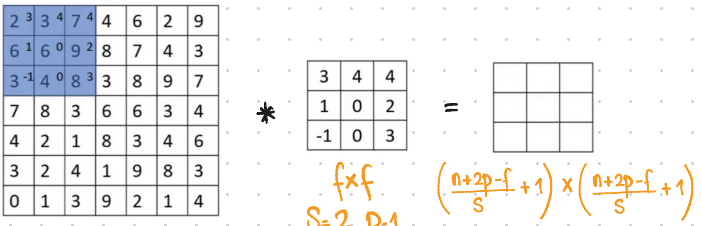
* it helps with → shrinking output when applying filters
  → throwing away image info from the edge
* how much to pad? → VALID: no padding
  → SAME: pad so the output size = input size



STRIDED CONVOLUTIONS

* how many steps the filter moves.





$n \times n$

$f \times f$

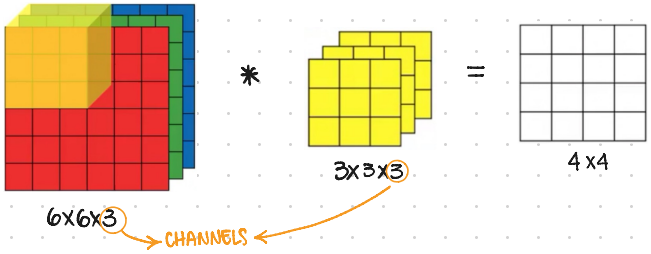
$S=2, P=1$

$\left(\frac{n+2p-f}{s}+1\right) \times \left(\frac{n+2p-f}{s}+1\right)$

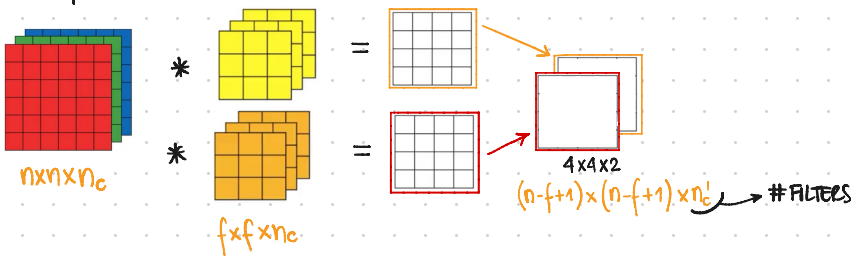
if it's not an integer, we round down.

CONVOLUTIONS OVER VOLUME

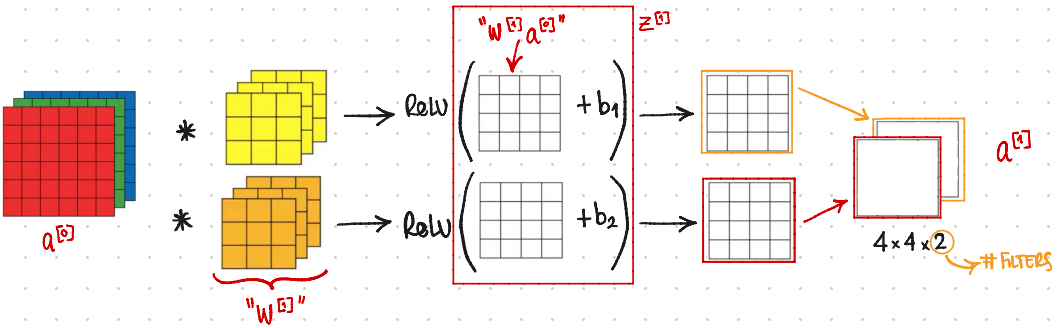
* convolutions on RGB images:



* can you use multiple filters at the same time?



ONE LAYER OF A CONV NET



* if you have 10 filters of size 3×3×3, how many parameters you have?

3×3×3 = 27 (size filter) + bias = 28 × 10 (# filters)

NOTATION:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding size

$s^{[l]}$ = stride

$n_c^{[l]}$ = # filters

input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$

output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$$n_{H,W}^{[l]} = \left[\frac{n^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right]$$

$f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$ = filter size

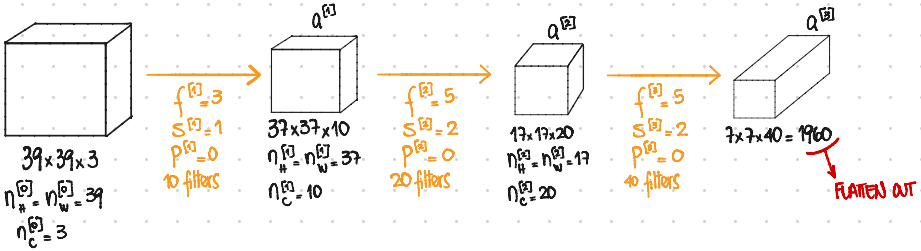
$a^{[l]} \longrightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$A^{[l]} \longrightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

weights $\longrightarrow f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias $\longrightarrow n_c^{[l]}$

EXAMPLE

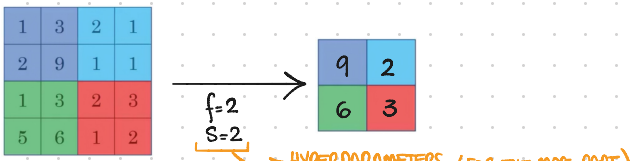


* Types of layers in a CNN → CONV (convolution), POOL (pooling), FC (fully connected) — Simpler to define

POOLING LAYERS

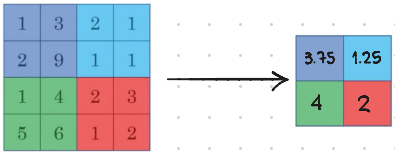
* Useful for → reduce size of the representation, speed up the computation, make some of the features that detects more robust

MAX POOLING:



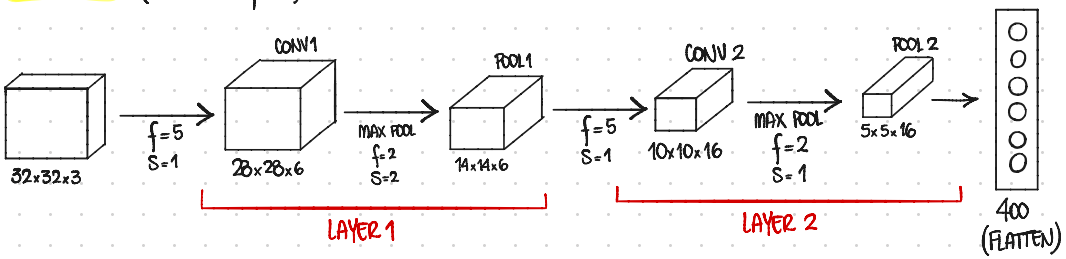
* you take the heighest value in the region
* the important features detected in the quadrants (regions) remains preserved in the output
* it has hyperparameters but no parameters to learn, unlike convolution.

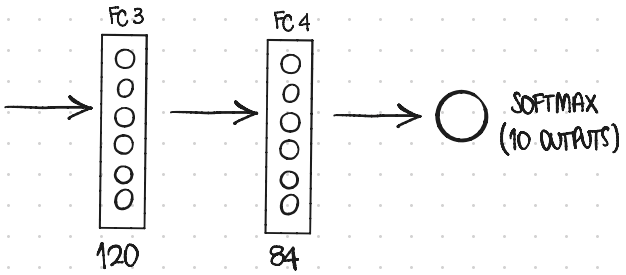
AVERAGE POOLING:



* not used as often as MAX POOL. Sometimes used in very deep (NN) to collapse your representation, e.g. from 7x7x1000 to 1x1x1000.

EXAMPLE (LeNet-5 inspired)





* as you move deep into the network... $n_H, n_W \downarrow$ $n_C \uparrow$

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	608
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	3216
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48120
FC4	(84,1)	84	10164
Softmax	(10,1)	10	850

$(5\times5\times3+1)\times8$

$(5\times5\times8+1)\times16$

MOST PARAMETERS TEND TO BE IN THE FC LAYERS

$84\times10+10$

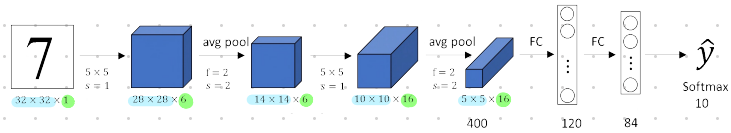
IT DROPS GRADUALLY AS THE NN GETS DEEPER

WHY CONV NETS?

* advantages → parameter sharing: a feature detector that's useful in one part of the image is probably useful in another part of the image.
→ sparsity of connections: in each layer, each output value depends only on a small number of inputs.

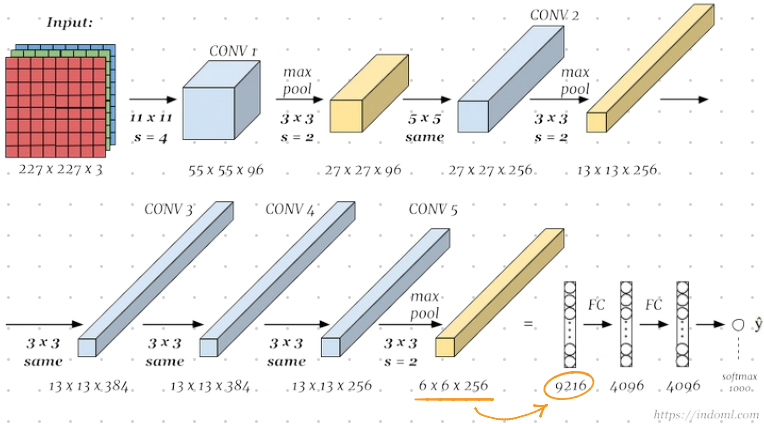
CLASSIC NETWORKS

* LeNet 5:

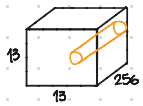


* it has 60k parameters → relatively small compared to other networks.
* as you move deeper into the network, n_H and n_W tend to go ↓
  n_C tends to go ↑
* general structure: conv pool + conv pool + FC + FC + output
* section II and III of the paper.

* AlexNet:



* Similar to LeNet, but much bigger → ~60M parameters
  60k parameters
* it has more hidden layers
  been trained on much more data (ImageNet)
* it uses ReLU activations and Local Response Normalization
  → normalize inputs based on one location and across all input channels. } You don't want too many active neurons with a very high activation.



DOESN'T ACTUALLY HELP THAT MUCH.