Pooling layer -> max pooling

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 9 & 2 \\ 6 & 3 \end{bmatrix}$$

4×4 → 2×2

Hyperparams:

filter = 2

stride = 2

No weights/params

$$\begin{bmatrix} 1 & 3 & 2 & 1 & 3 \\ 2 & 9 & 1 & 1 & 5 \\ 1 & 3 & 2 & 3 & 2 \\ 8 & 3 & 5 & 1 & 0 \\ 5 & 6 & 1 & 2 & 9 \end{bmatrix} \longrightarrow \begin{bmatrix} 9 & 9 & 5 \\ 9 & 9 & 5 \\ 8 & 6 & 9 \end{bmatrix}$$

$5 \times 5 \times n_c$ → $3 \times 3 \times n_c$

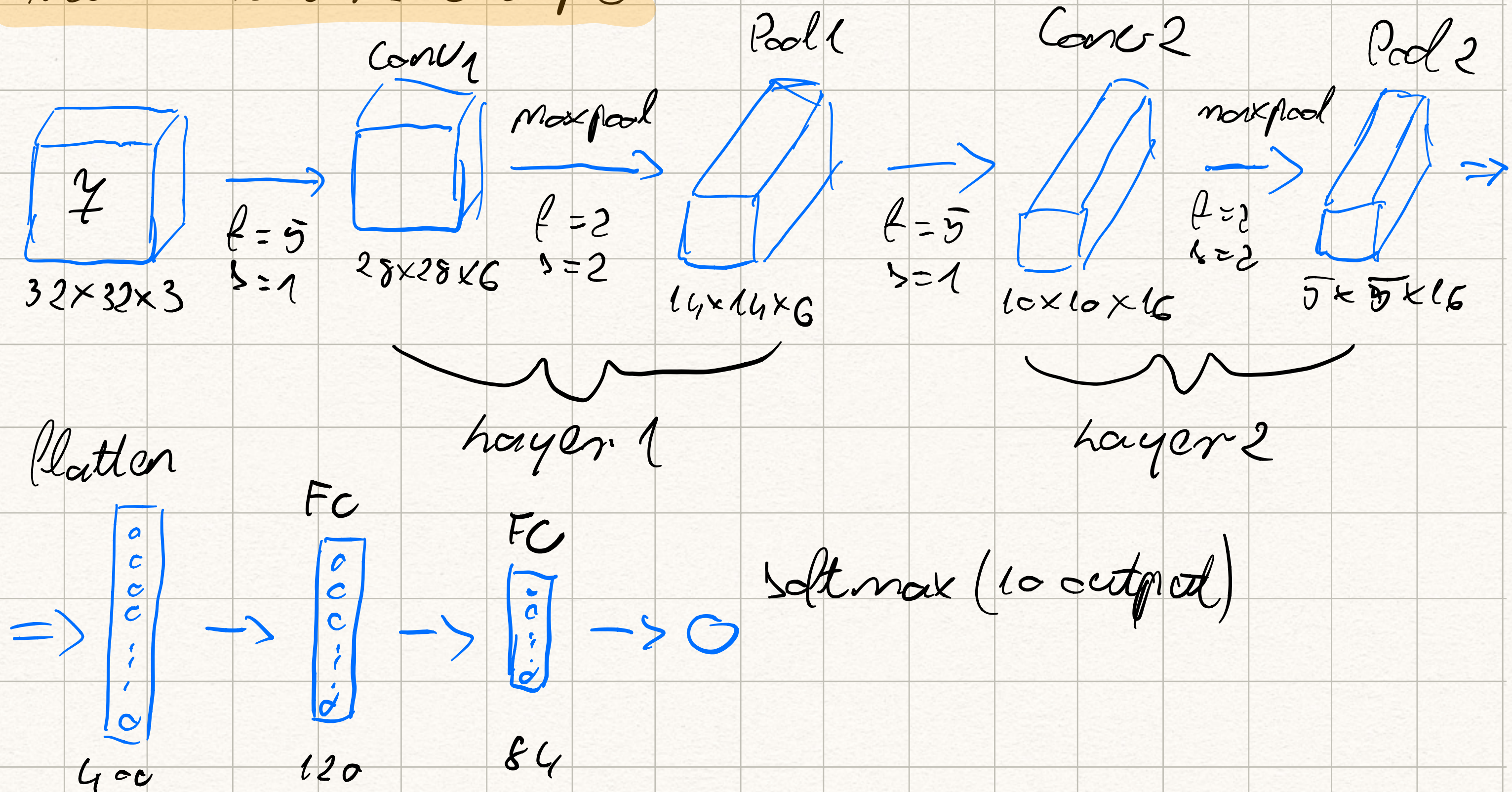
$f = 3$

$s = 1$

Pooling is applied independently across the channels.

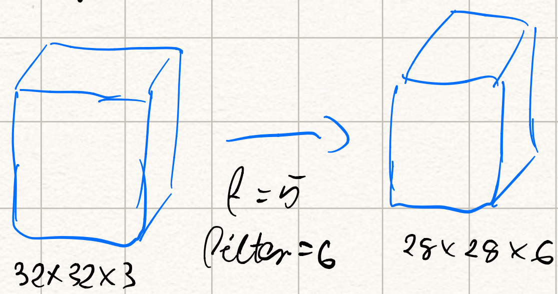
- reduce the size of height and with, but keep the #of channels
- detect features in different scales, max pool highlights strong features

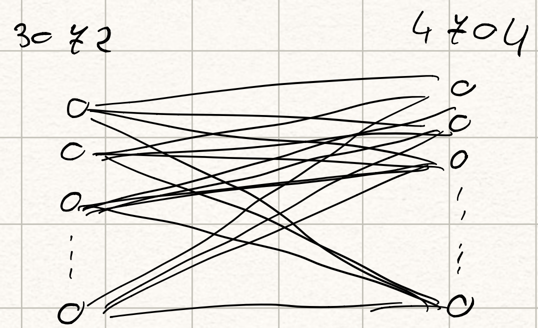
Neural network example



	Activation shape	Activation size	# parameters
Input	32×32×3	3072	0
Conv1 (f=5, s=1)	28×28×8	6272	208
Pool 1	14×14×8	1568	0
Conv2 (f=5, s=1)	10×10×16	1600	416
Pool 2	5×5×16	400	0
FC	120×1	120	48001
FC	84×1	84	10081
Softmax	10×1	10	841

Why convolution





conv params:

$5 \times 5 = 25 + 1 = 26$
bias

$26 \times 6 = 156$ params
filter

$3072 \times 4704 \approx 14$ million parameters

$\Rightarrow$ hard to train

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

- Parameter sharing: A feature detector that's useful 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.