Convolutional neural networks

Convolutional filters

Input Feature Map

3	5	2	8	1
9	7	5	4	3
2	0	6	1	6
6	3	7	9	2
1	4	9	5	1

Convolutional Filter

1	0	0
1	1	0
0	0	1

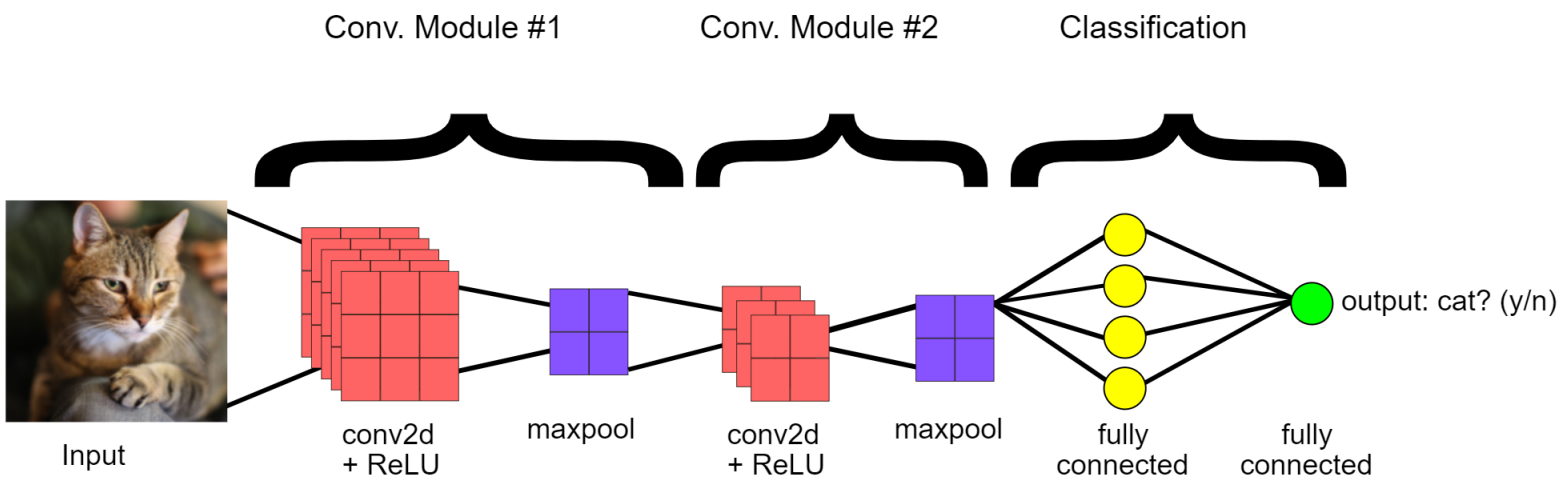
Input Feature Map

3×1	5×0	2×0	8	1
9×1	7×1	5×0	4	3
2×0	0×0	6×1	1	6
6	3	7	9	2
1	4	9	5	1

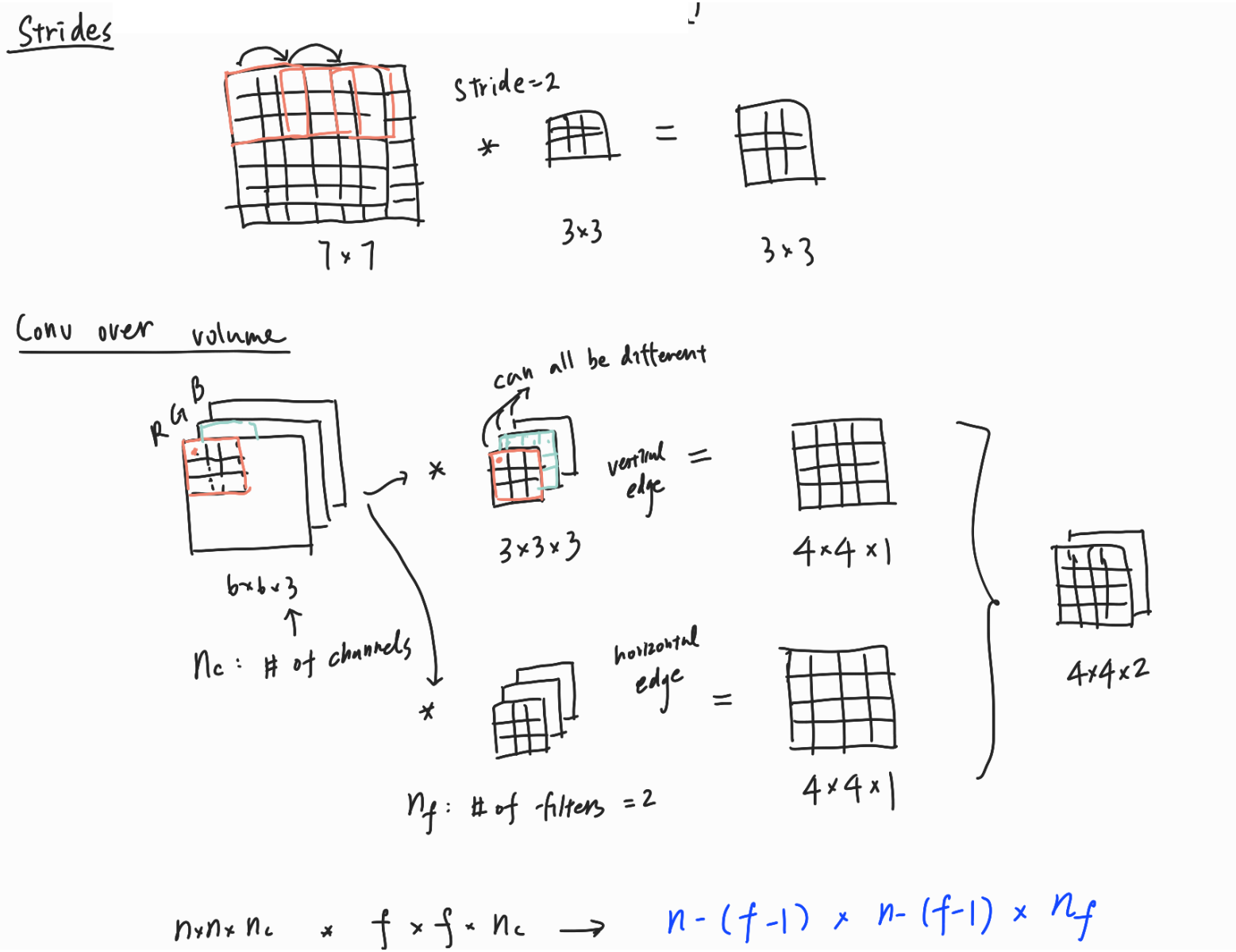
3+0+0+9+7+0+0+0+6

Output Feature Map

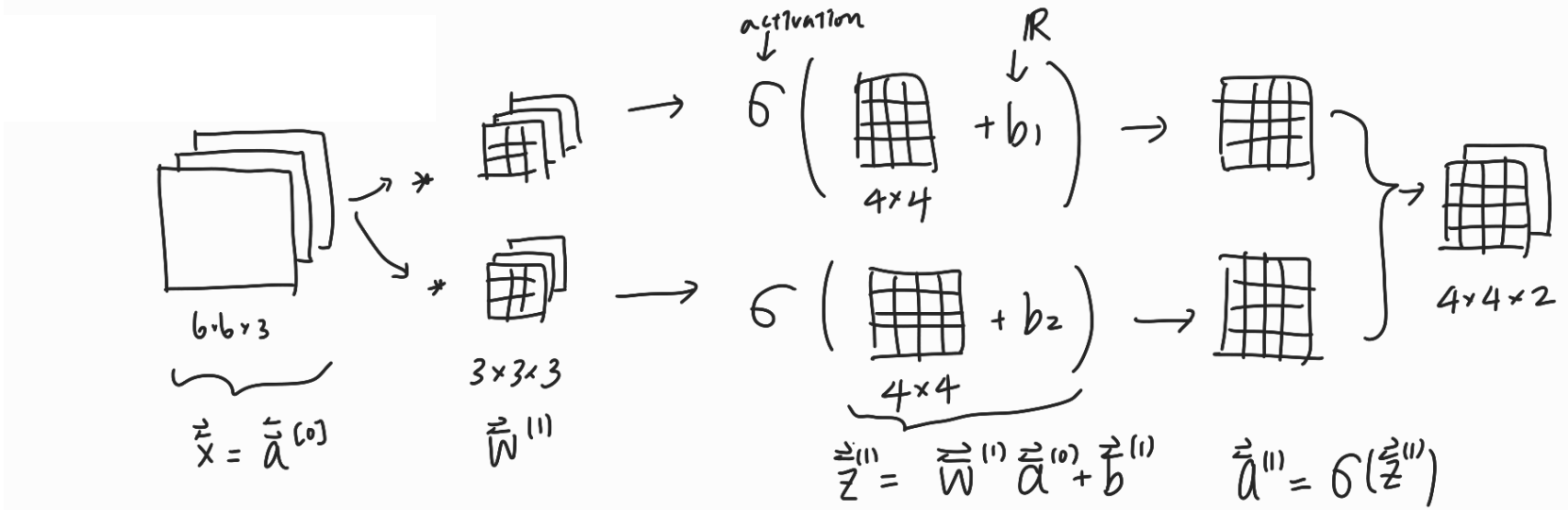
25	18	17
18	22	14
20	15	23



Strides


Strides
Stride=2
7×7
3×3
3×3
Conv over volume
can all be different
R G B
6×6×3
n_c : # of channels
3×3×3
vertical edge
4×4×1
horizontal edge
4×4×1
4×4×2
n_f : # of filters = 2
$n \times n \times n_c \quad * \quad f \times f \times n_c \longrightarrow n-(f-1) \times n-(f-1) \times n_f$

A convolutional layer



Q: If we have 10 3×3×3 filters, how many parameters do we have?

3×3×3 = 27

27 + 1 → 28

weights bias

28 × 10 = 280. #

Number of parameters in a conv layer

More generally. If layer l is a conv layer:

- filter size : $f^{[l]}$
- # of channels : $n_c^{[l-1]}$
- # of filters : $n_c^{[l]}$

input shape : $n^{[l-1]} \times n^{[l-1]} \times n_c^{[l-1]}$

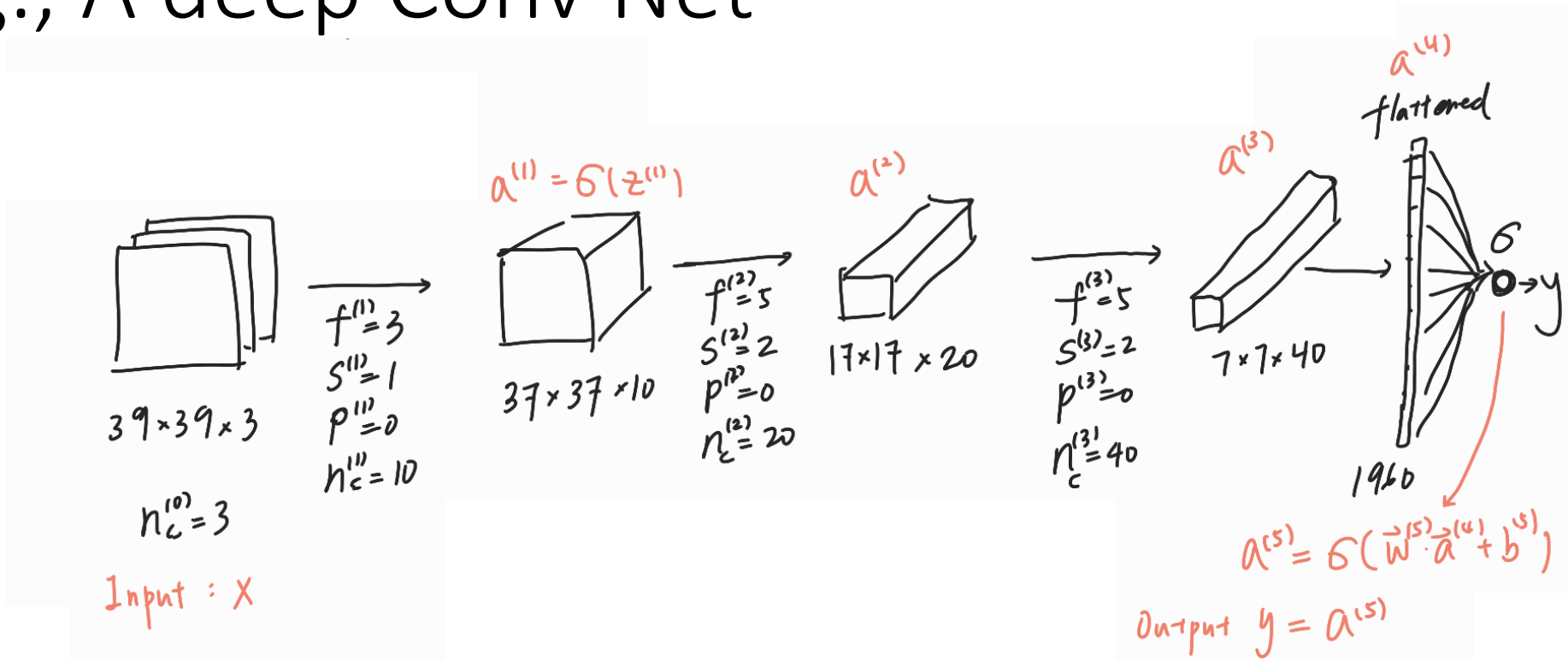
output shape : $n^{[l]} \times n^{[l]} \times n_c^{[l]}$

each filter : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

# of weights : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

# of bias : $n_c^{[l]}$

E.g., A deep Conv Net

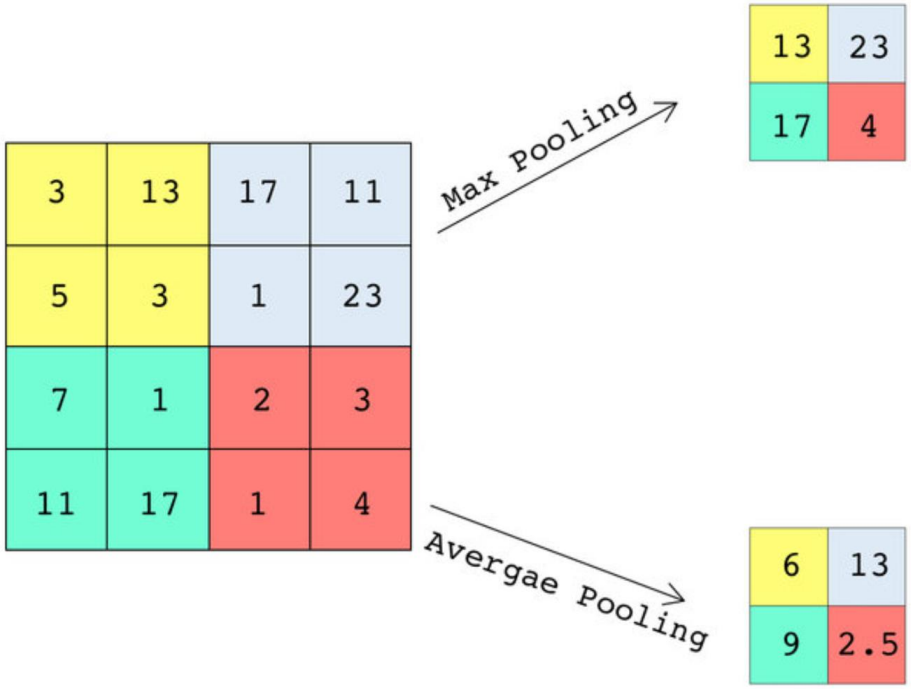


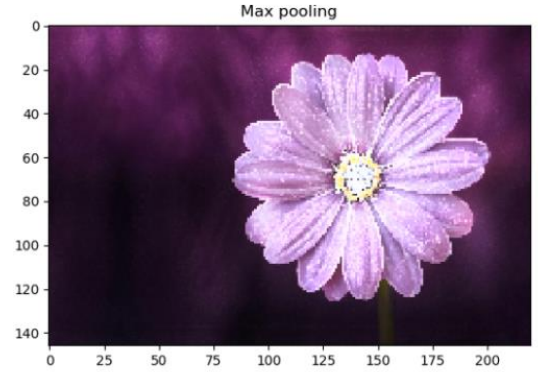
{ classification → final g = sigmoid.

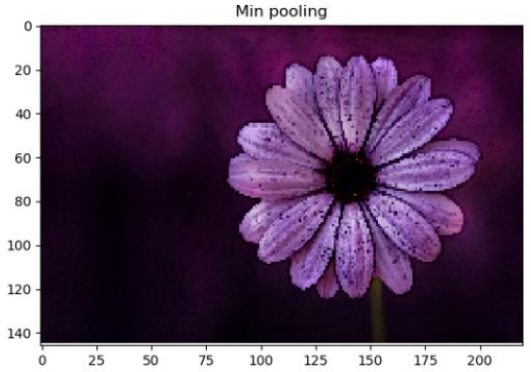
regression → no need for final nonlinear activation

Pooling

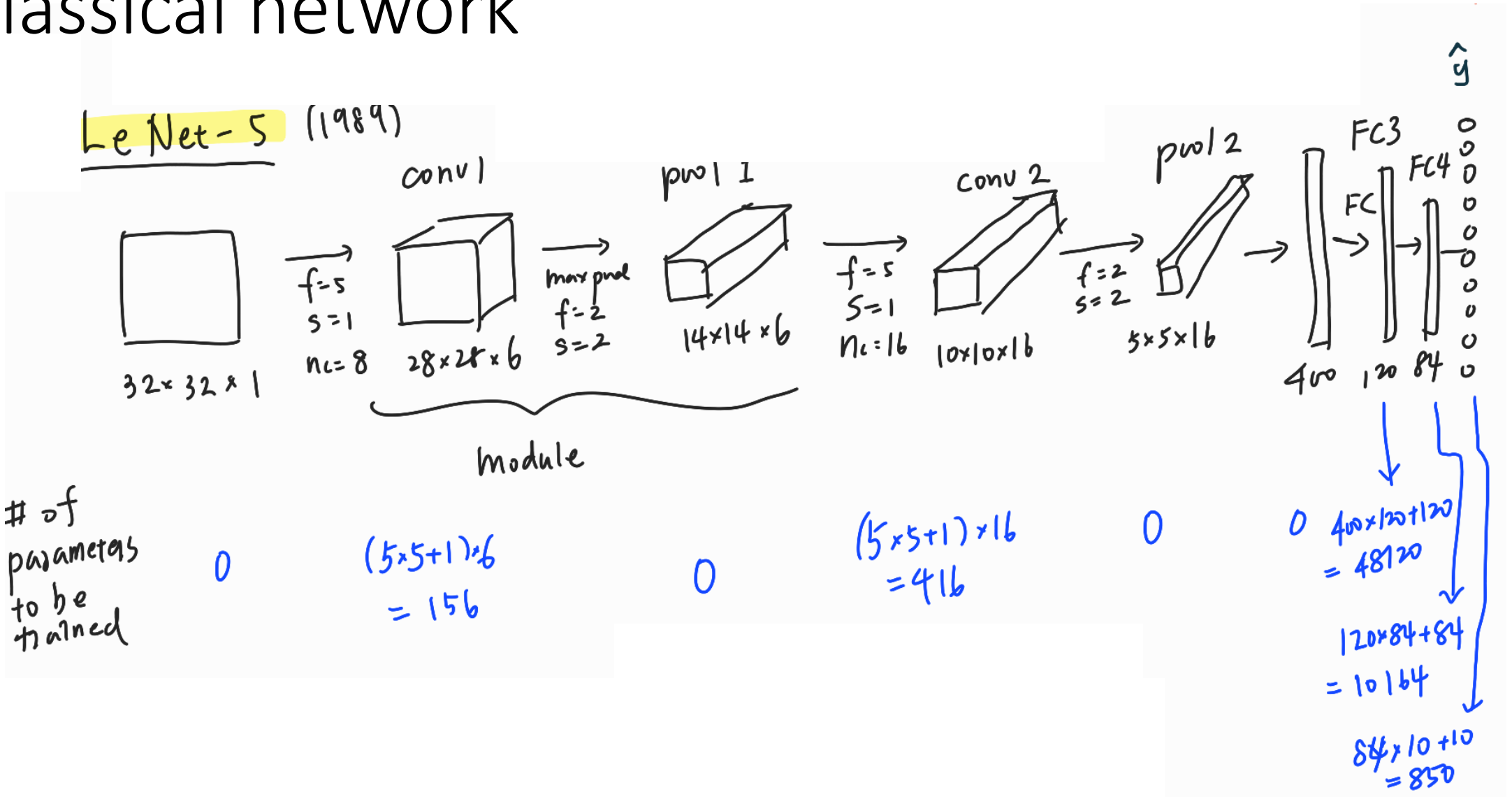
For a given layer, pooling reduces the size of the layer output



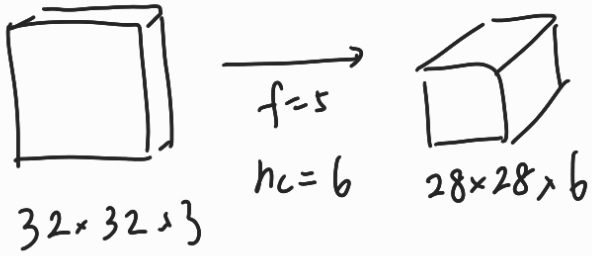




Classical network

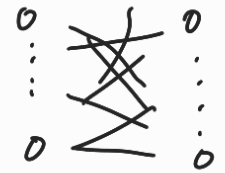


Why convolutions?



With conv → $(5 \times 5 + 1) \times 6 = 156$

With FC:



3072 4704

$3072 \times 4704 = 14M$!

- Parameter sharing: reusing the same weights for different parts of the image.

- Sparsity of connections: In FC, each output pixel is connected to all input pixels via weights.
  In Conv, each output pixel only depends on small # of input pixels.