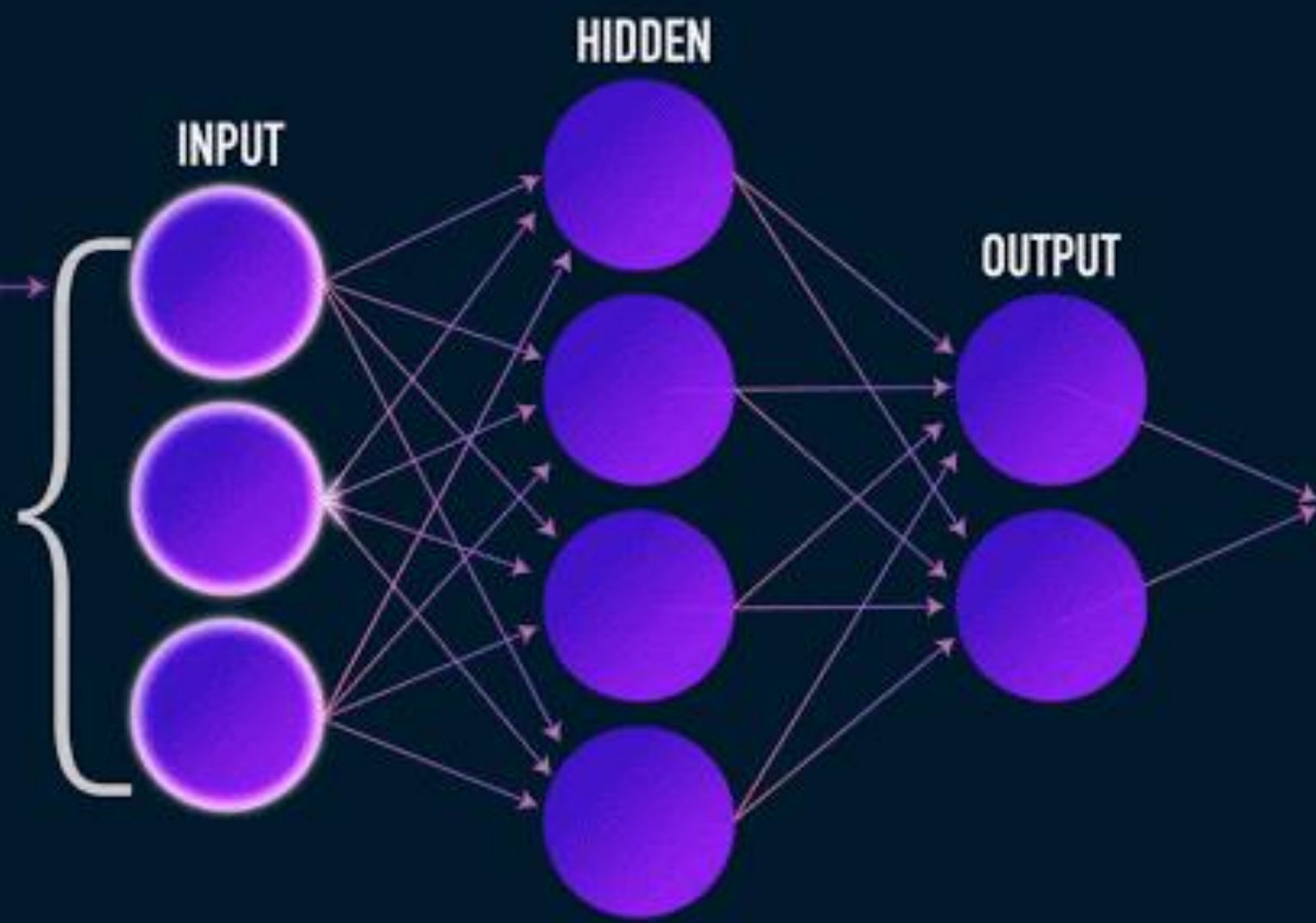
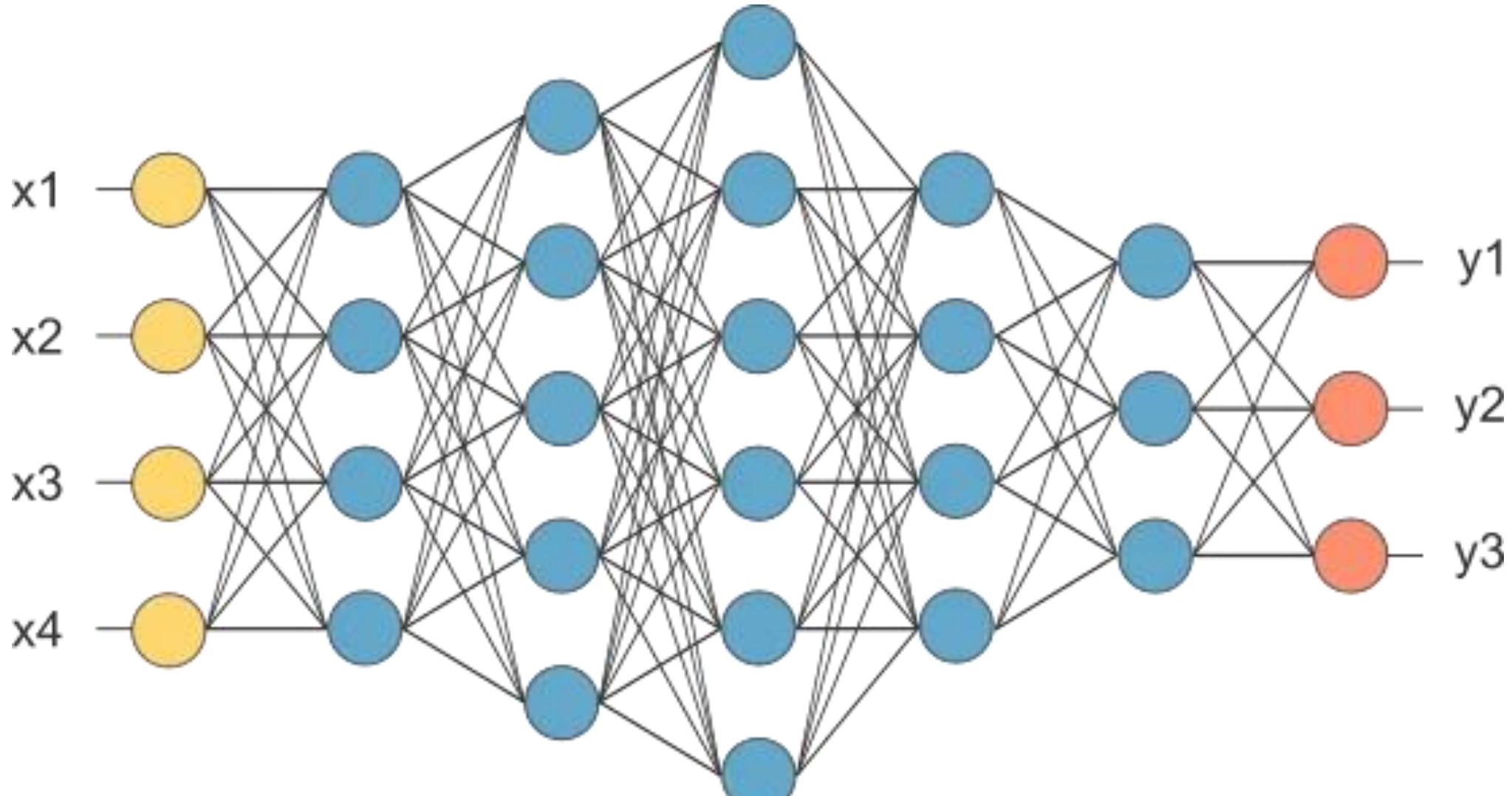


CNN



Why Conv layer? What is the problem with Dense layers?



Convolution



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

Convolution

- *Kernel* = grid of weights
- Kernel is “applied” to the image
- Traditionally used in image processing

1	2	-1
0	1	2
-2	1	0

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 x 5 – Image Matrix



1	0	1
0	1	0
1	0	1

3 x 3 – Filter Matrix

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Learning to detect edges

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



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

→

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

Sobel filter



convolution

*

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

3x3

=

45°
70°
22°

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

Scharr filter



...

Kernels

- Feature detectors
- Kernels are learned

Oblique line detector

1	0	0
0	1	0
0	0	1

Vertical line detector

0	1	0
0	1	0
0	1	0

Architectural decisions for convolution

- Grid size
- Stride
- Depth
- Number of kernels

Grid size

- # of pixels for height/width
- Odd numbers

5 by 5

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

3 by 3

1	2	9
1	6	5
2	2	3

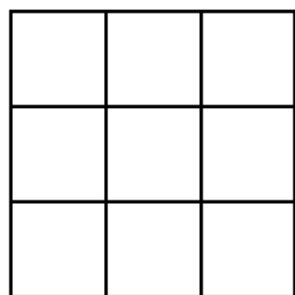
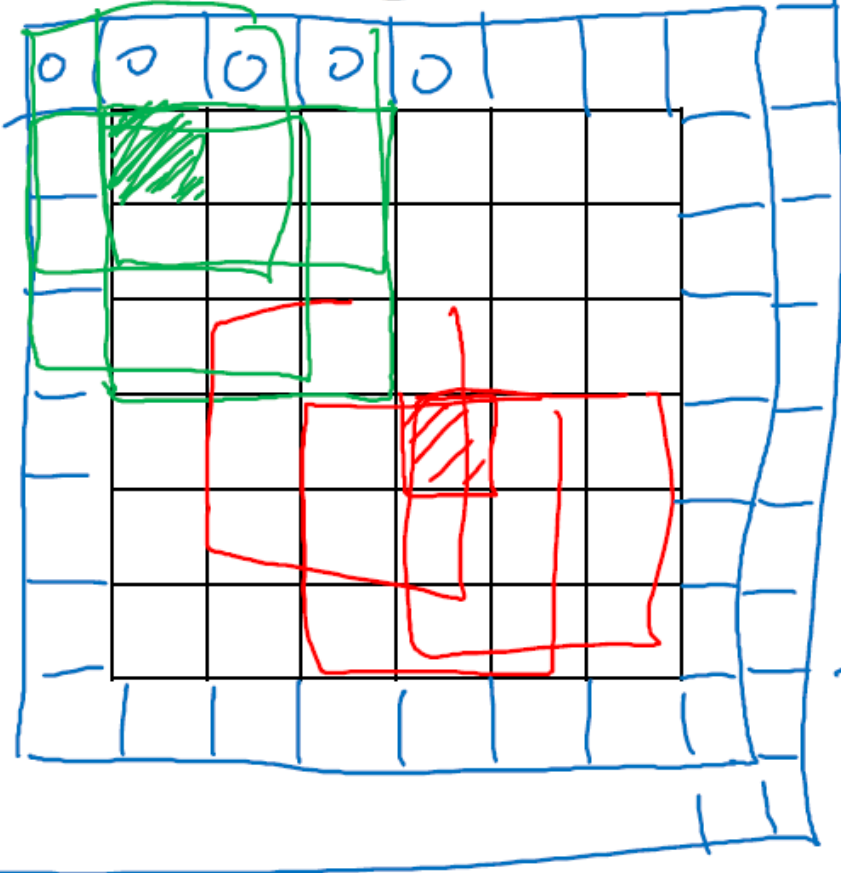
Padding

There are two problems arises with convolution:

1. Every time after convolution operation, original image size getting shrinks, as we have seen in above example six by six down to four by four and in image classification task there are multiple convolution layers so after multiple convolution operation, our original image will really get small but we don't want the image to shrink every time.
2. The second issue is that, when kernel moves over original images, it touches the edge of the image less number of times and touches the middle of the image more number of times and it overlaps also in the middle. So, the corner features of any image or on the edges aren't used much in the output.

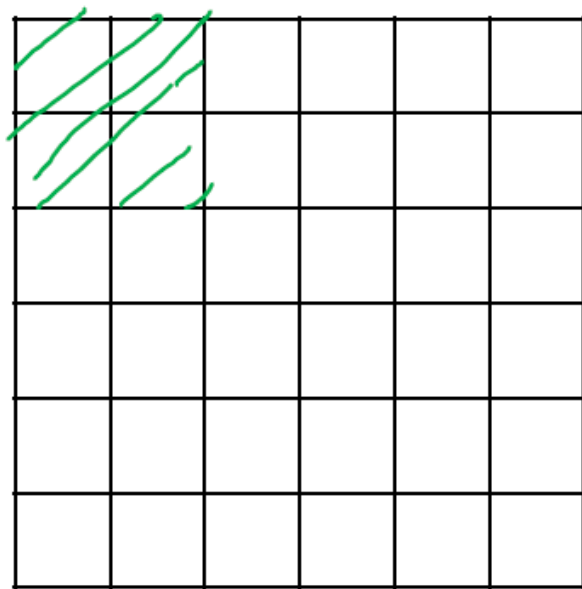
Padding

- shrinky output
- throw away info from edge



3x3
f x f

=



6x6

6x6 → 8x8
n x n

$n - f + 1 \times n - f + 1$

$6 - 3 + 1 = 4$

p = padding = 1

$n + 2p - f + 1 \times n + 2p - f + 1$

$6 + 2 - 3 + 1 \times \underline{\underline{4}} = 6 \times 6$

Valid and Same convolutions

→ no padding

“Valid”: $n \times n \quad * \quad f \times f \quad \rightarrow \quad \underline{n - f + 1} \times n - f + 1$

$6 \times 6 \quad * \quad 3 \times 3 \quad \rightarrow \quad 4 \times 4$

“Same”: Pad so that output size is the same as the input size.

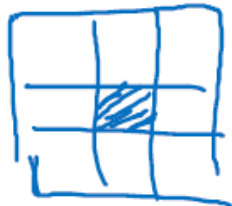
$$n + 2p - f + 1 \times n + 2p - f + 1$$
$$\cancel{n + 2p - f + 1} = \cancel{n} \Rightarrow \boxed{p = \frac{f-1}{2}}$$

$3 \times 3 \quad p = \frac{3-1}{2} = 1$

$5 \times 5 \quad f=5$

f is usually odd

1×1
 3×3
 5×5
 7×7

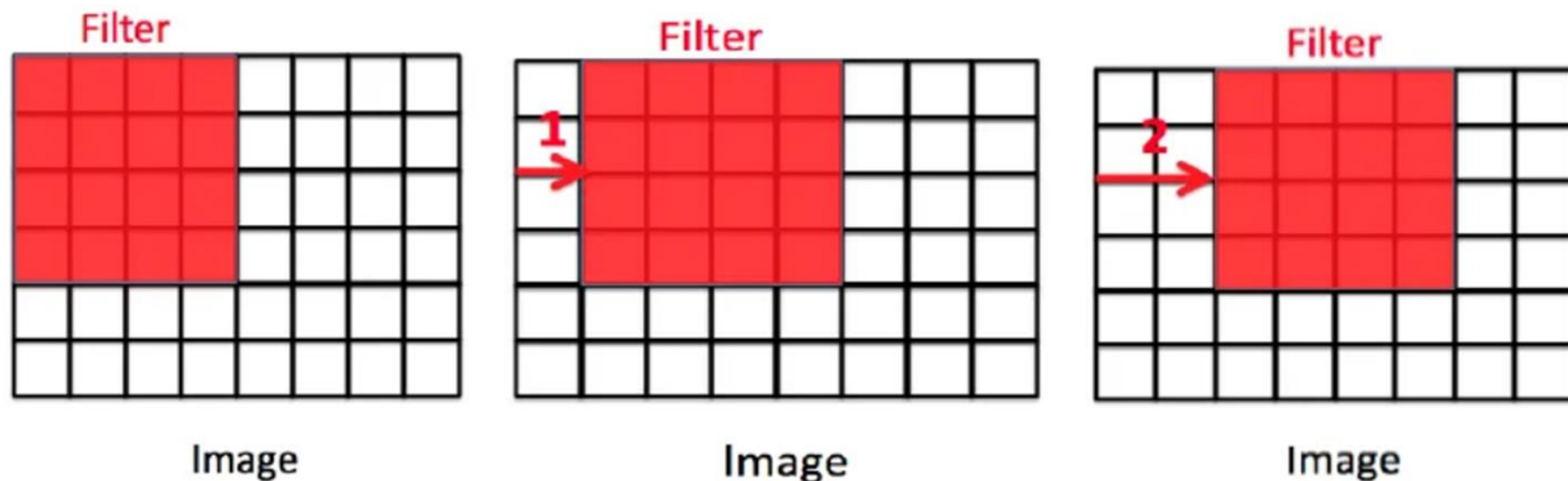


$p=2$

Stride

- Step size used for sliding kernel on image
- Indicated in pixels

Stride



left image: stride =0, middle image: stride = 1, right image: stride =2

Summary of convolutions

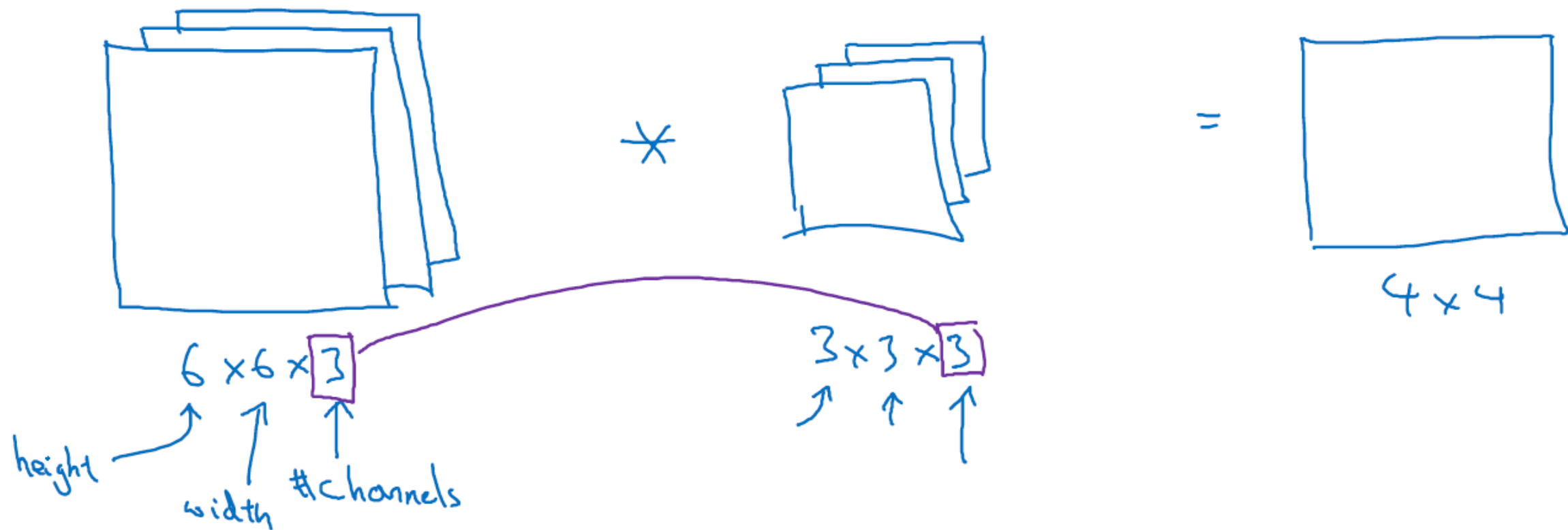
$n \times n$ image $f \times f$ filter

padding p stride s

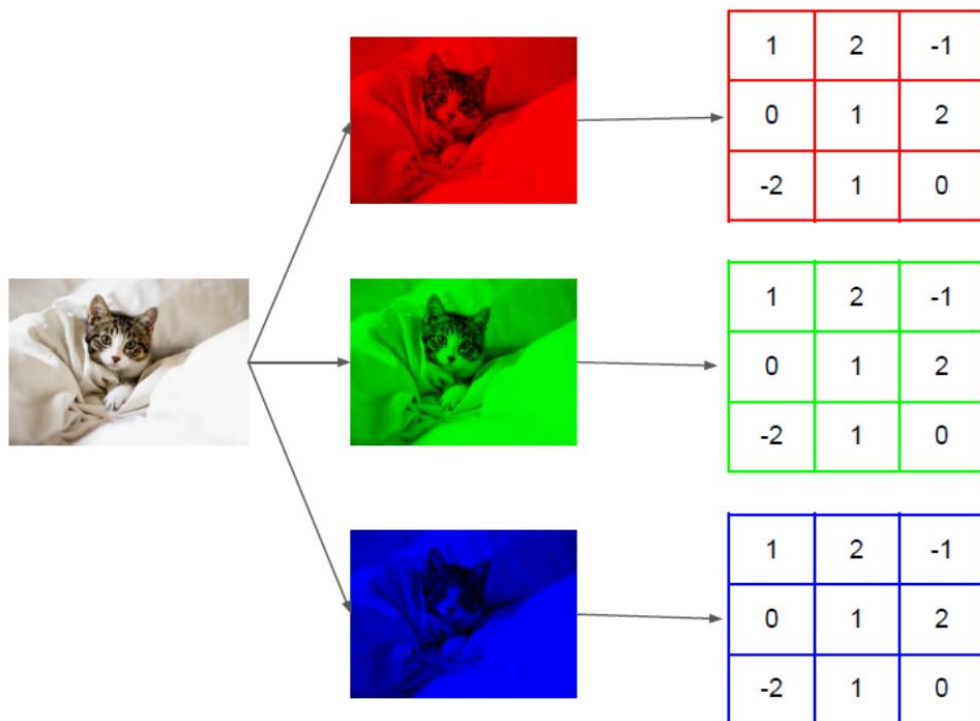
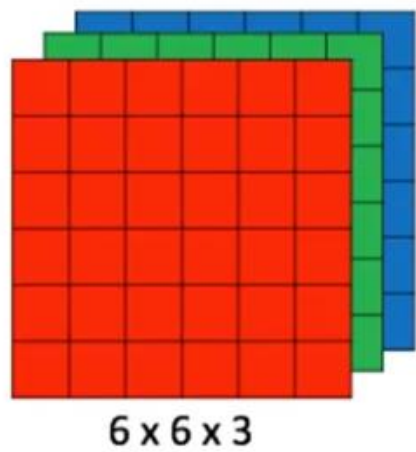
Output Size:

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \underbrace{\frac{n+2p-f}{s}} + 1 \right\rfloor$$

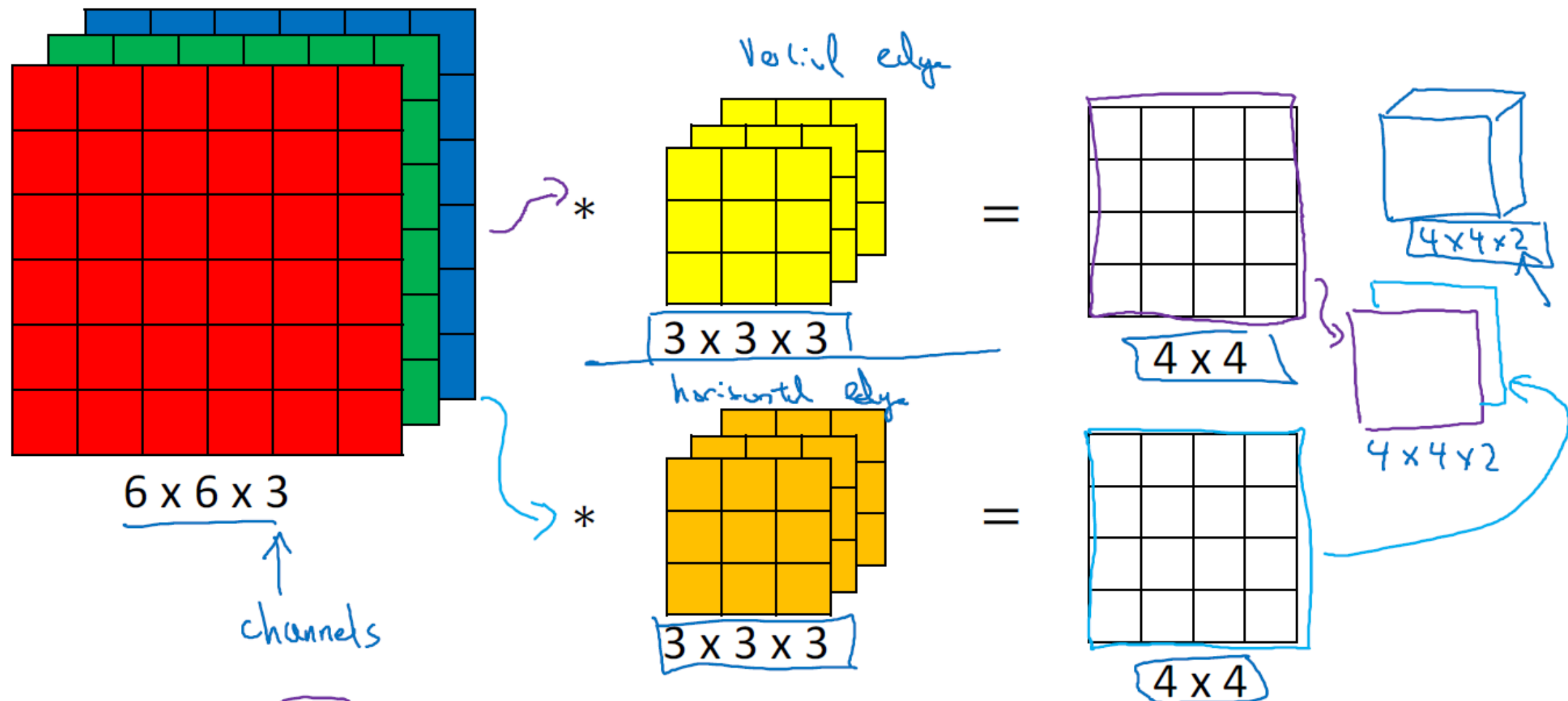
Convolutions on RGB images



Depth



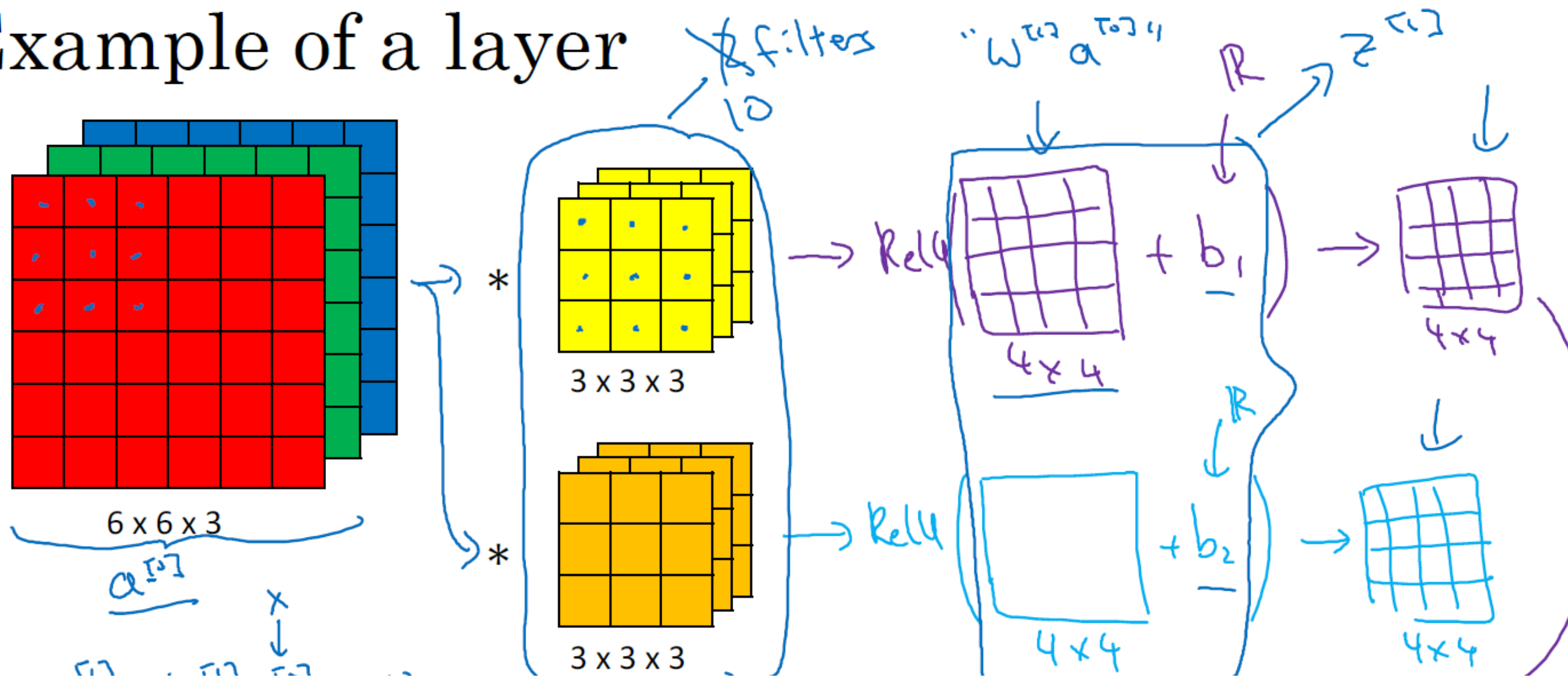
Multiple filters



Summary: $n \times n \times n_c$ \times $f \times f \times n_c$ \rightarrow $\frac{n-f+1}{4} \times \frac{n-f+1}{4} \times \frac{n_c}{2}$ \uparrow #filters

$6 \times 6 \times 3$ $3 \times 3 \times 3$ $4 \times 4 \times 2$

Example of a layer



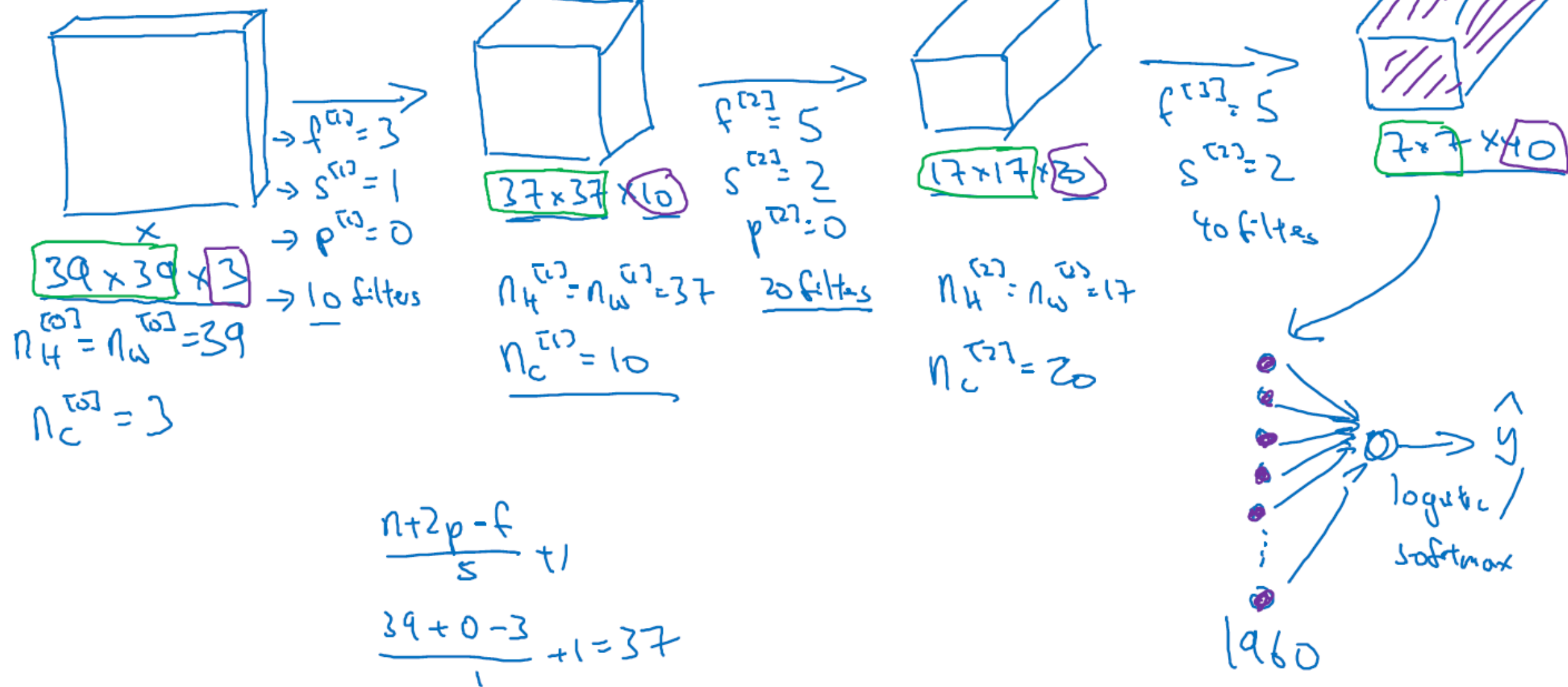
Number of parameters in one layer

If you have 10 filters that are $3 \times 3 \times 3$ in one layer of a neural network, how many parameters does that layer have?


General rule of parameter computation

- Filter size, stride, padding, and number of filters are given
- Input shape is provided
- #parameters = ???

Example ConvNet



Types of layer in a convolutional network:

- Convolution (conv) ←
 - Pooling (pool) ←
 - Fully connected (FC) ←
- 

Pooling

- Downsample the image
- Overlaying grid on image
- Max/average pooling
- No parameters

Why pooling?

- Reduce the size of the representation
- Speed up computation

Pooling settings

- Grid size
- Stride
- Type (e.g., max, average)

Max Pooling

4	3	1	5
1	3	4	8
4	5	4	3
6	5	9	4

$$\text{Max}([4, 3, 1, 3]) = 4$$

4	3	1	5
1	3	4	8
4	5	4	3
6	5	9	4



4	8
6	9

Average Pooling

4	3	1	5
1	3	4	8
4	5	4	3
6	5	9	4

$$\text{Avg}([4, 3, 1, 3]) = 2.75$$

4	3	1	5
1	3	4	8
4	5	4	3
6	5	9	4



2.8	4.5
5.3	5.0

Summary of pooling

Hyperparameters:

f : filter size

s : stride

Max or average pooling

$$f=2, s=2$$

$$f=3, s=2$$



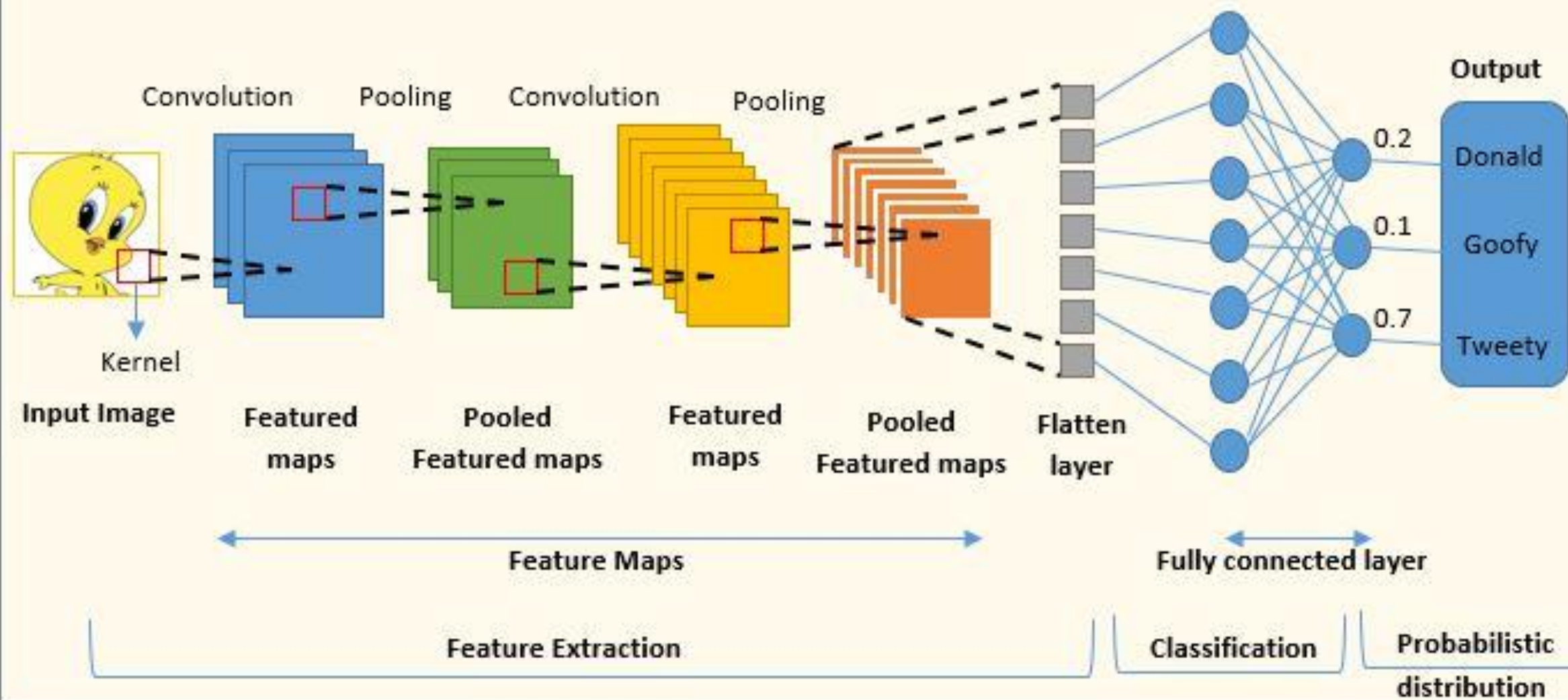
No parameters to learn!

$$n_H \times n_W \times \underline{n_C}$$

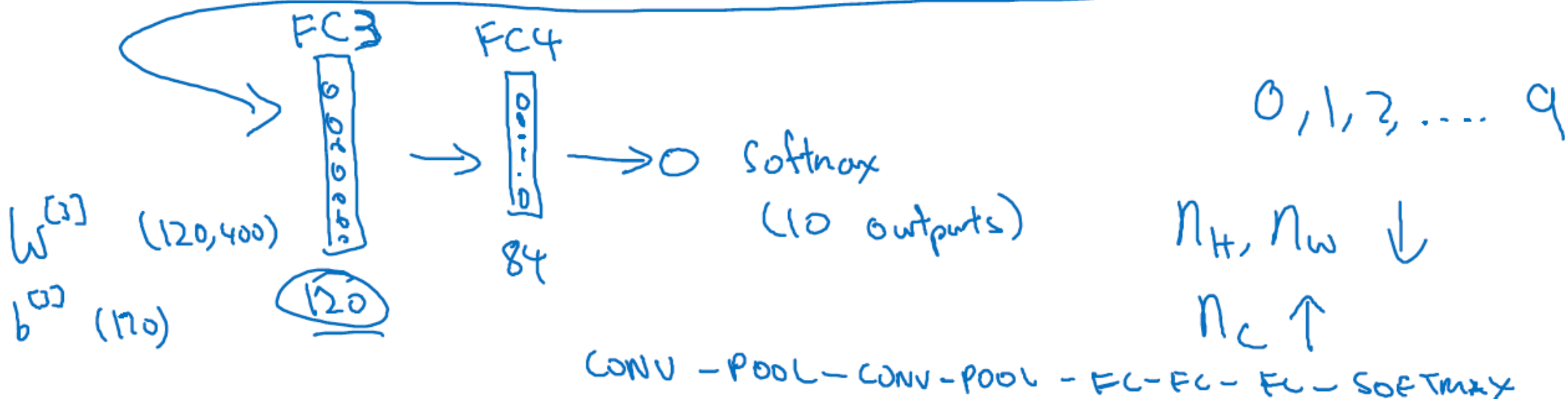
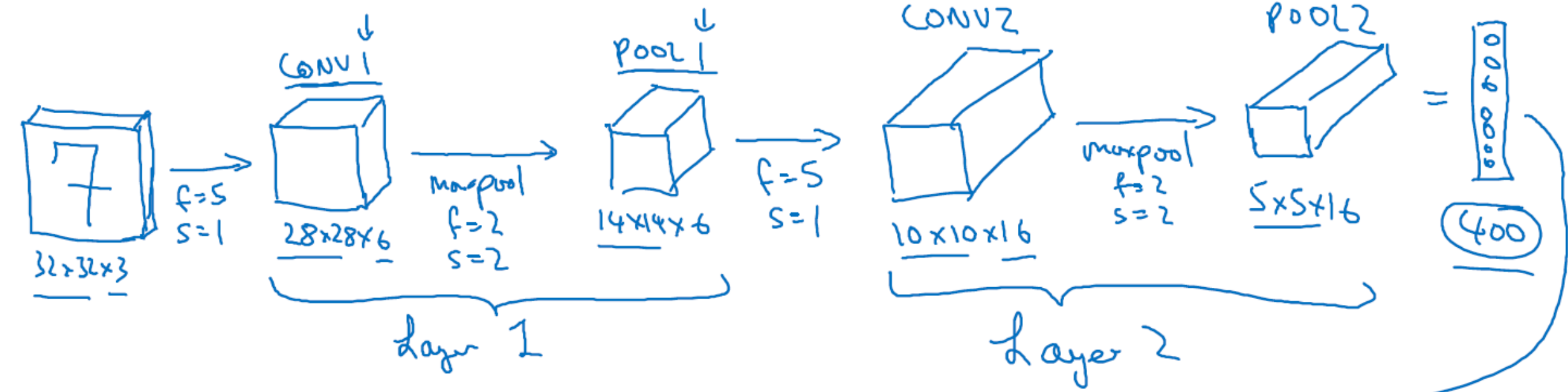


$$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times \underline{n_C}$$

A Typical Convolutional Neural Network (CNN)



Neural network example (LeNet-5)



Why convolutions

translation invariance

Parameter sharing: A feature detector (such as a vertical edge 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.

Advantage of convnet

- **Spatial Hierarchies and Local Receptive Fields:**

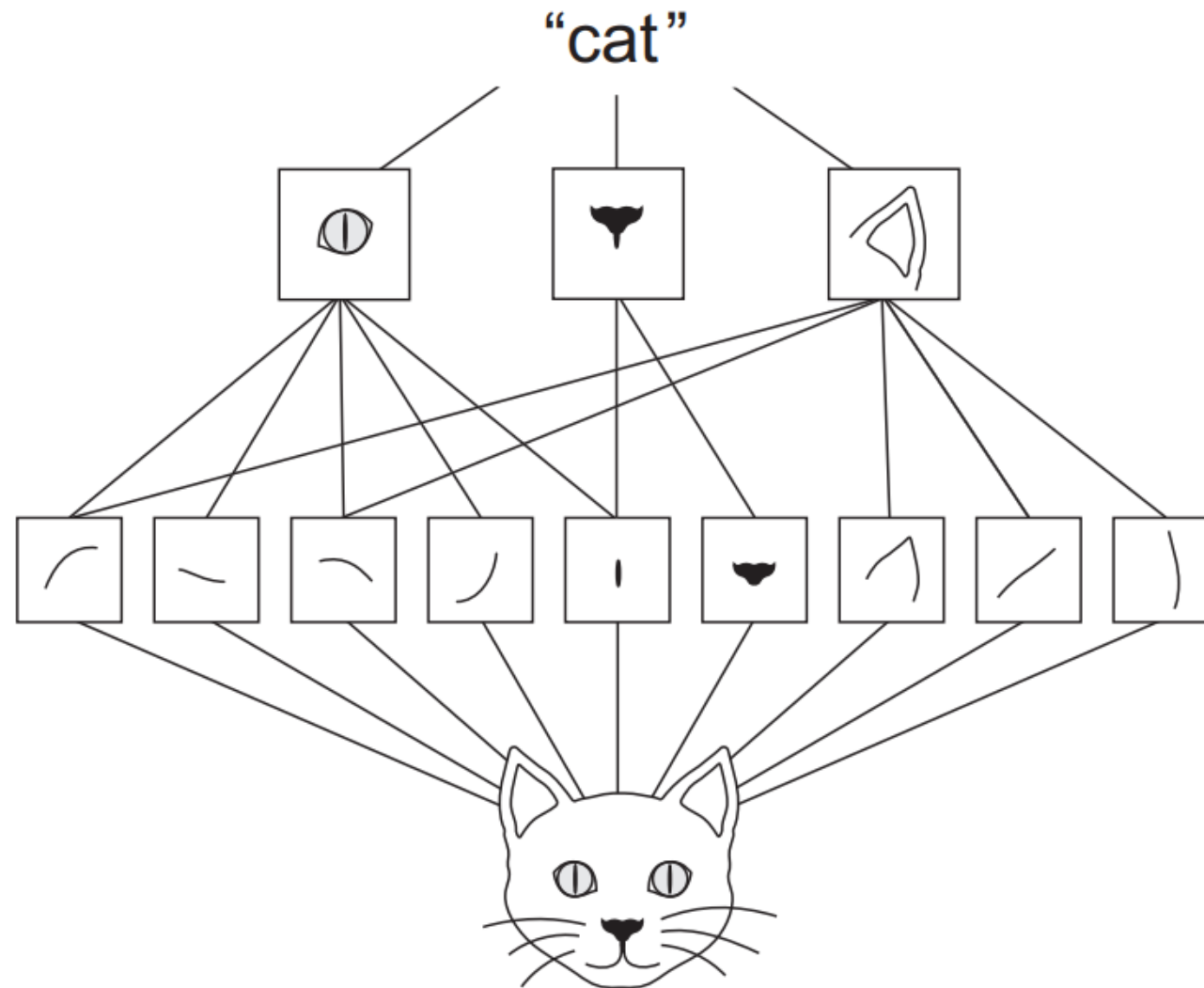
CNNs are specifically designed for tasks involving grid-like data, such as images. They use spatial hierarchies and local receptive fields, meaning they capture patterns in small local regions of the input data.

- **Parameter Sharing**

- **Translation Invariance:** CNNs are inherently translation-invariant, meaning they can recognize patterns regardless of their position in the input space.

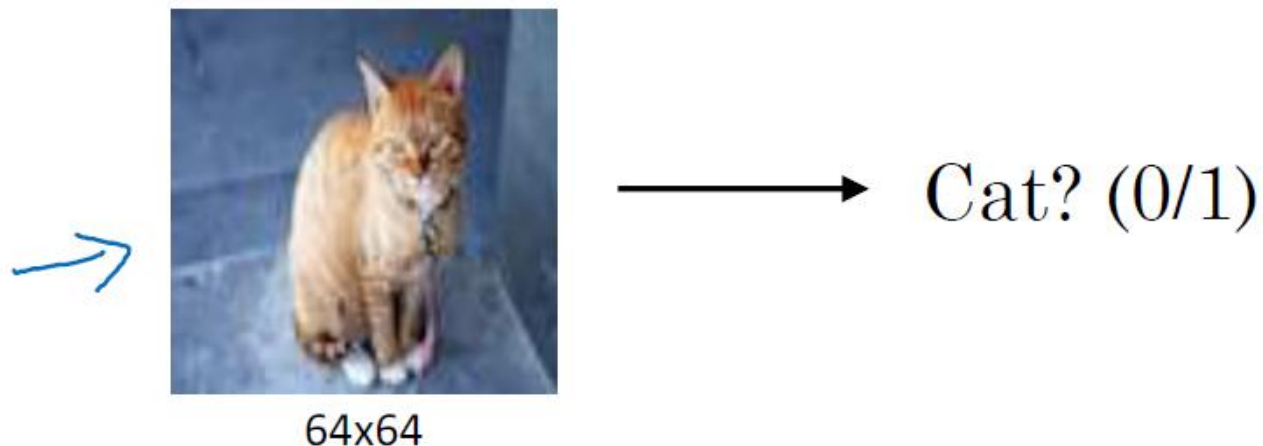
- **Feature Hierarchy:** CNNs automatically learn hierarchical representations of features. Lower layers capture simple features like edges and textures, while deeper layers capture more complex and abstract features.

A first convolution layer will learn small local patterns such as edges, a second convolution layer will learn larger patterns made of the features of the first layers, and so on. This allows convnets to efficiently learn increasingly complex and abstract visual concepts (because the visual world is fundamentally **spatially hierarchical**).



Computer Vision Problems

Image Classification



Neural Style Transfer



Object detection

