

CONVOLUTIONAL NEURAL NETWORK

Mahdi Roozbahani

Georgia Tech

Great visualization tool:

<https://poloclub.github.io/cnn-explainer/>

Slides are adopted on Ming Li (University of Waterloo – Deep learning part)
with some modifications

James Webb Space Telescope



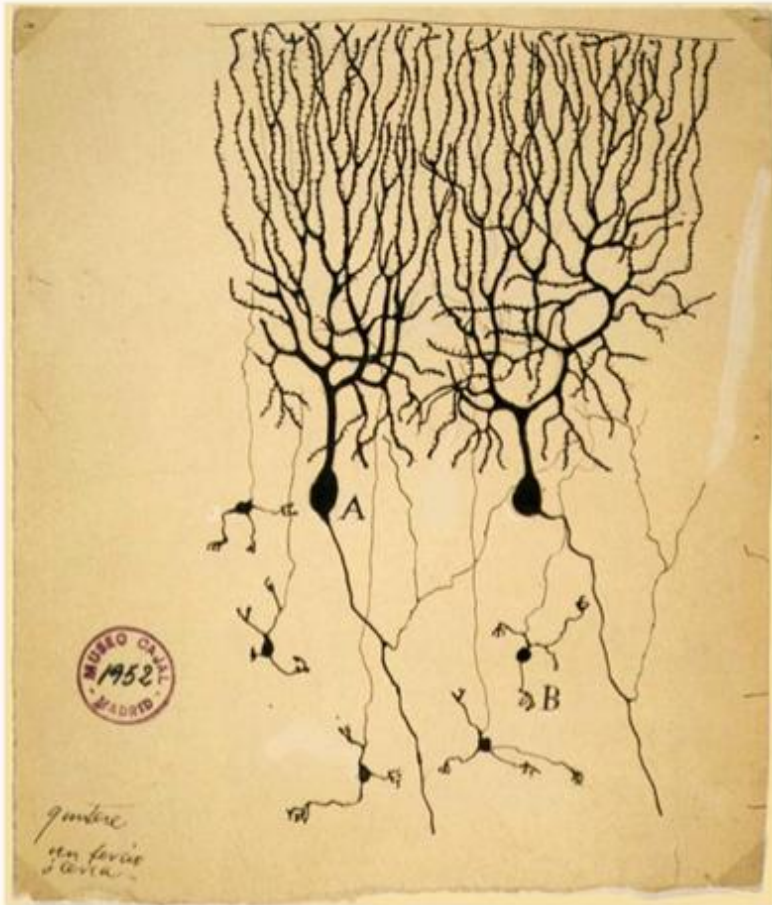
The landscape of “mountains” and “valleys”



A visual grouping of five galaxies



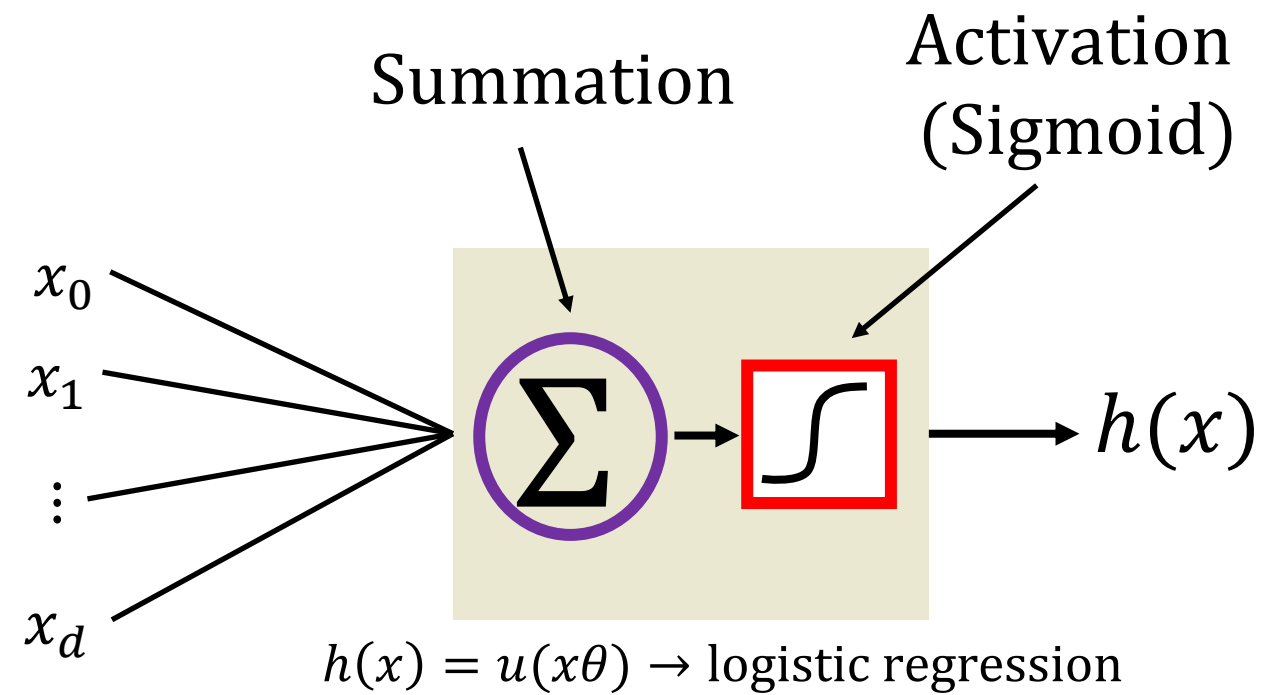
Inspiration from Biological Neurons



The first drawing of a brain cells by Santiago Ramón y Cajal in 1899

Neurons: core components of brain and the nervous system consisting of

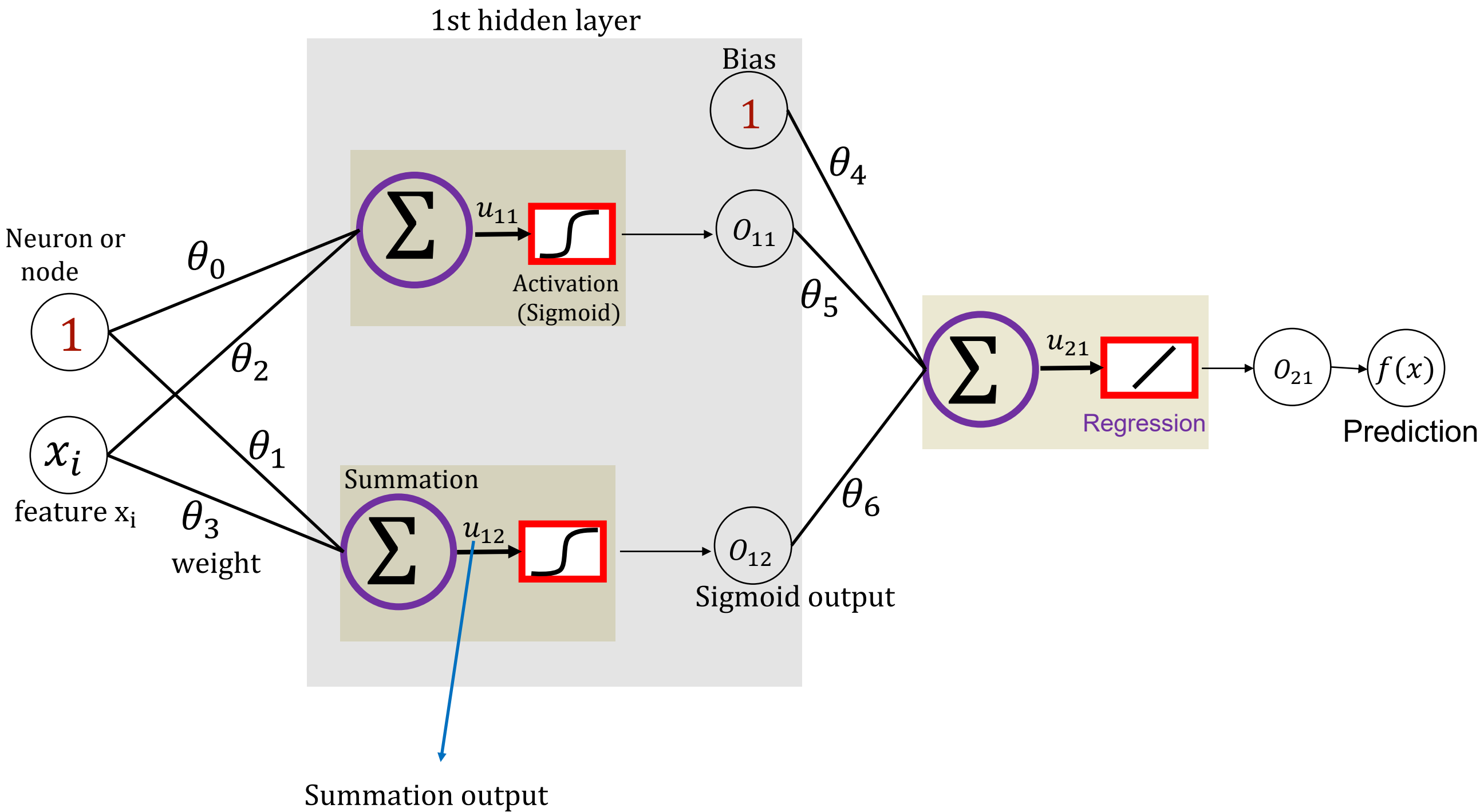
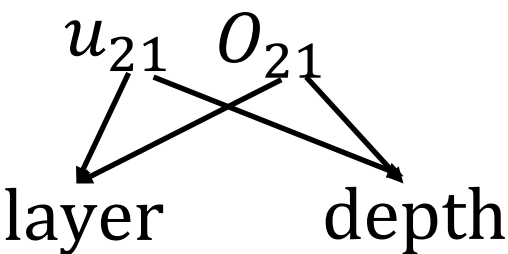
1. Dendrites that collect information from other neurons
2. An axon that generates outgoing spikes



$$\text{output} = \text{activation}(x\theta + b)$$

Name of the neuron	Activation function: $\text{activation}(z)$
Linear unit	$x\theta$
Threshold/sign unit	$\text{sign}(x\theta)$
Sigmoid unit	$\frac{1}{1 + \exp(x\theta)}$
Rectified linear unit (ReLU)	$\max(0, x\theta)$
Tanh unit	$\tanh(x\theta)$

NN Regression

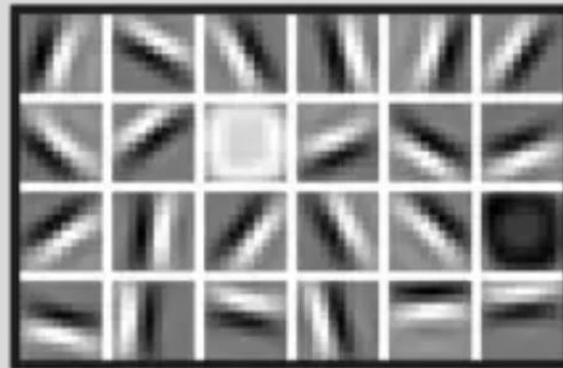


FACIAL RECOGNITION

Deep-learning neural networks use layers of increasingly complex rules to categorize complicated shapes such as faces.



Layer 1: The computer identifies pixels of light and dark.



Layer 2: The computer learns to identify edges and simple shapes.



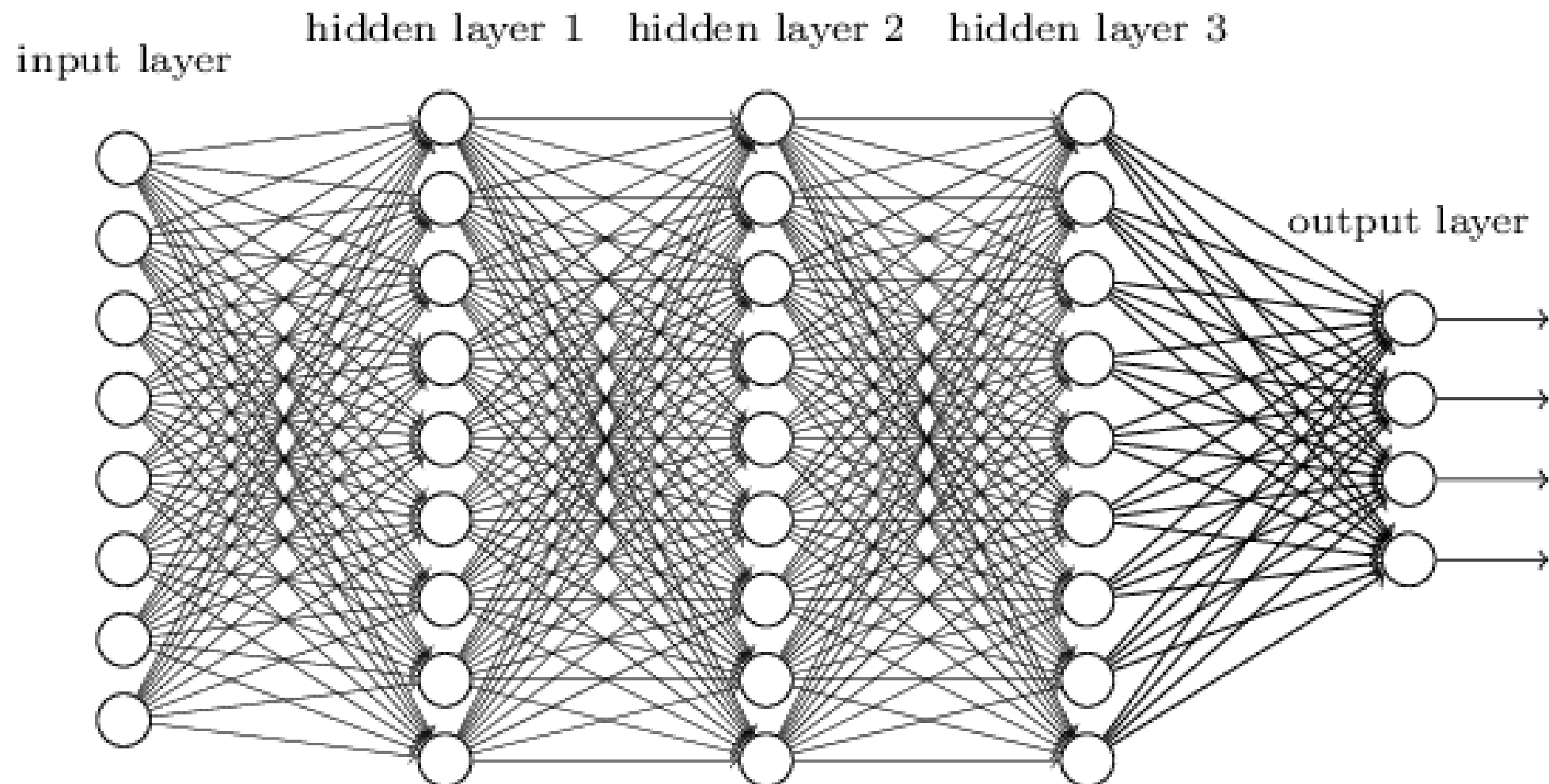
Layer 3: The computer learns to identify more complex shapes and objects.



Layer 4: The computer learns which shapes and objects can be used to define a human face.

Smaller Network: CNN

- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?



1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

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



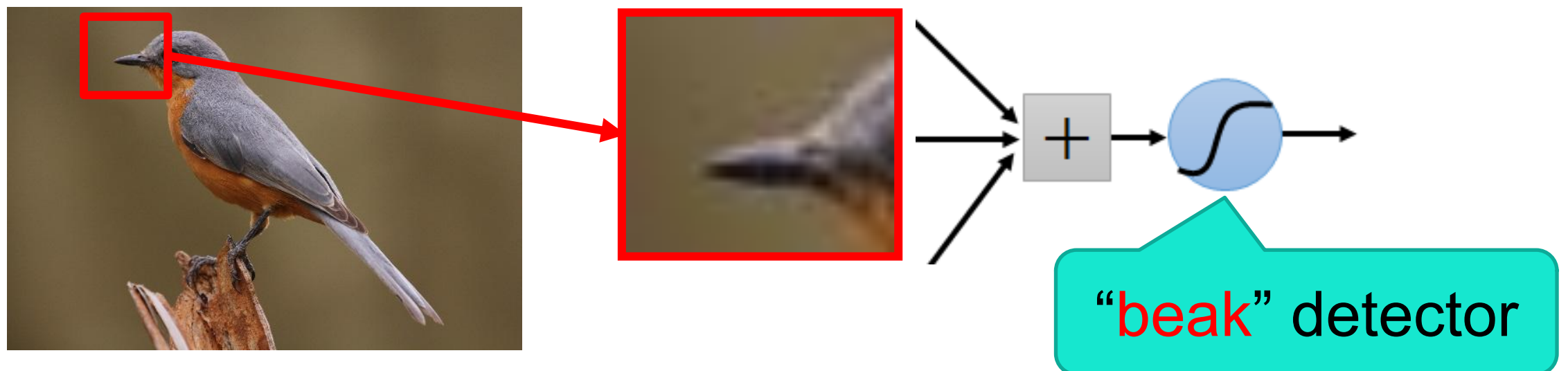
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0



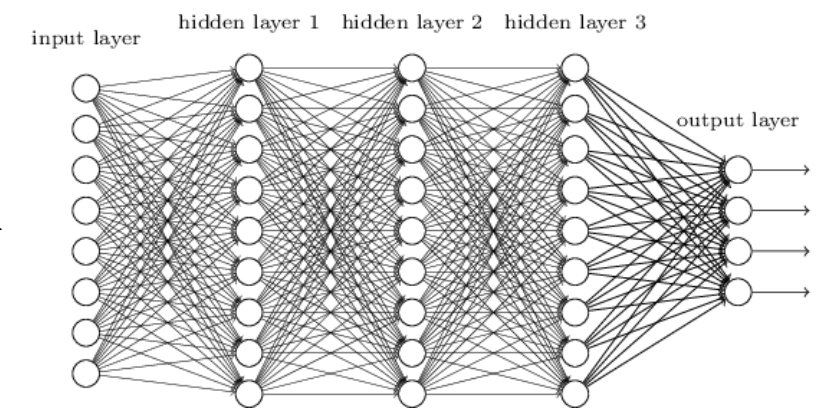
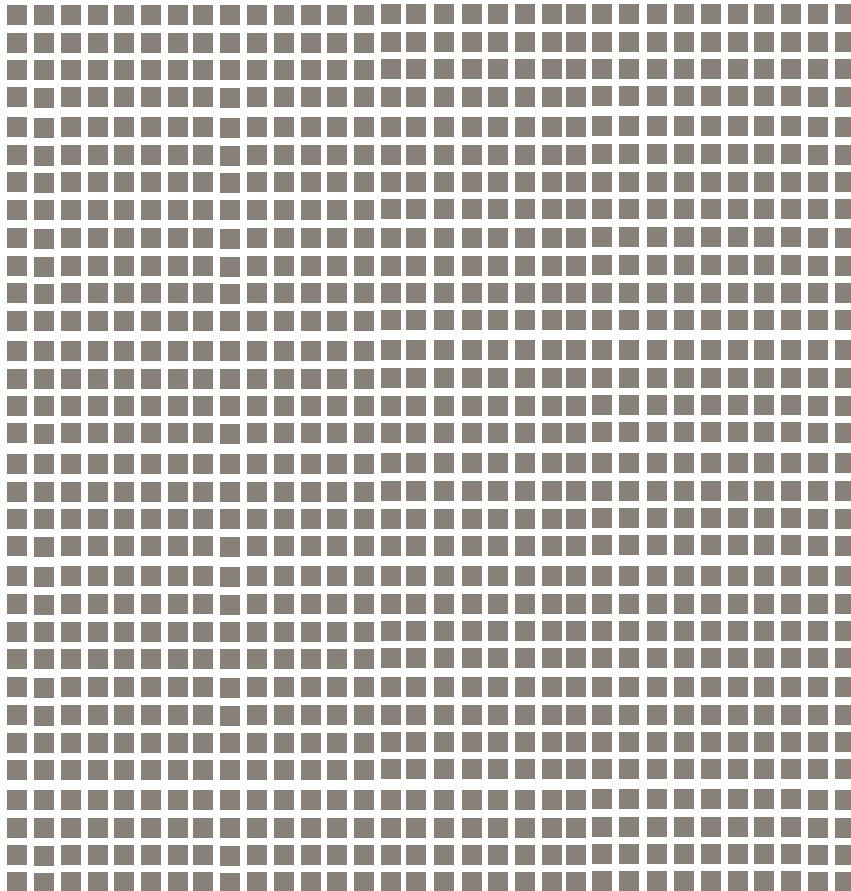
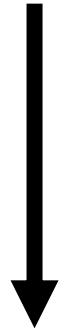
Consider learning an image:

- Some patterns are much smaller than the whole image

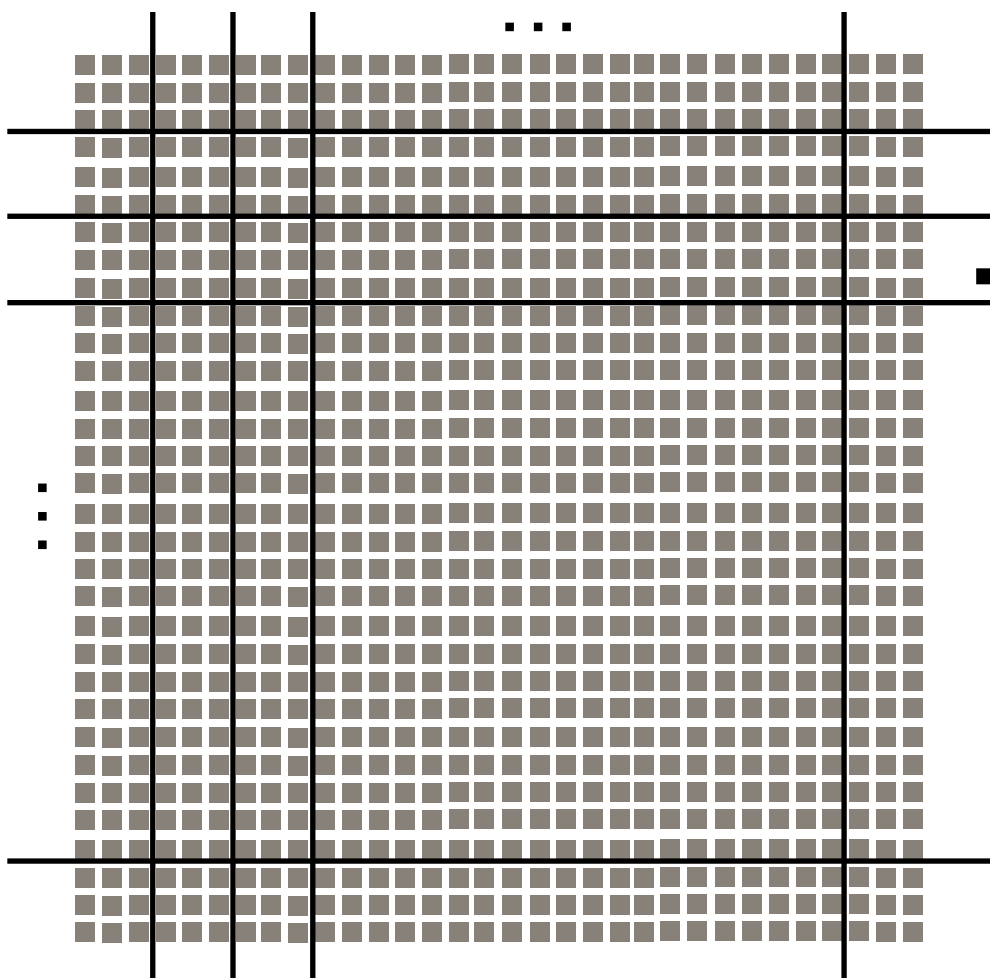
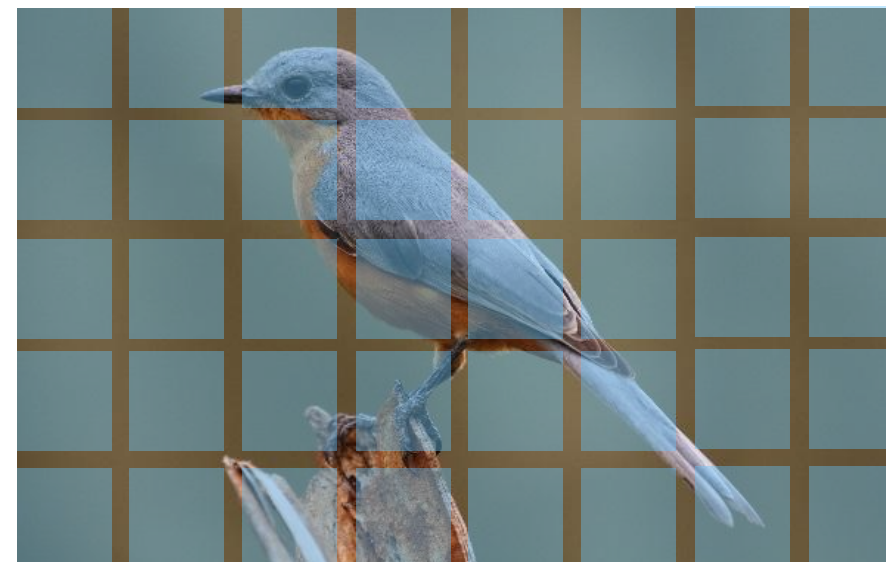
Can represent a small region with fewer parameters



600x600



600x600



3x3



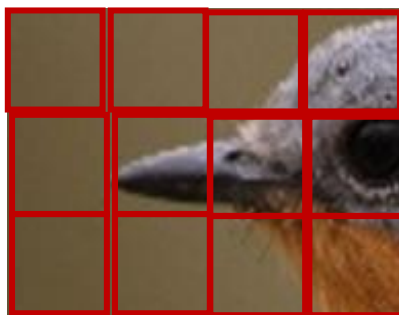
3x3



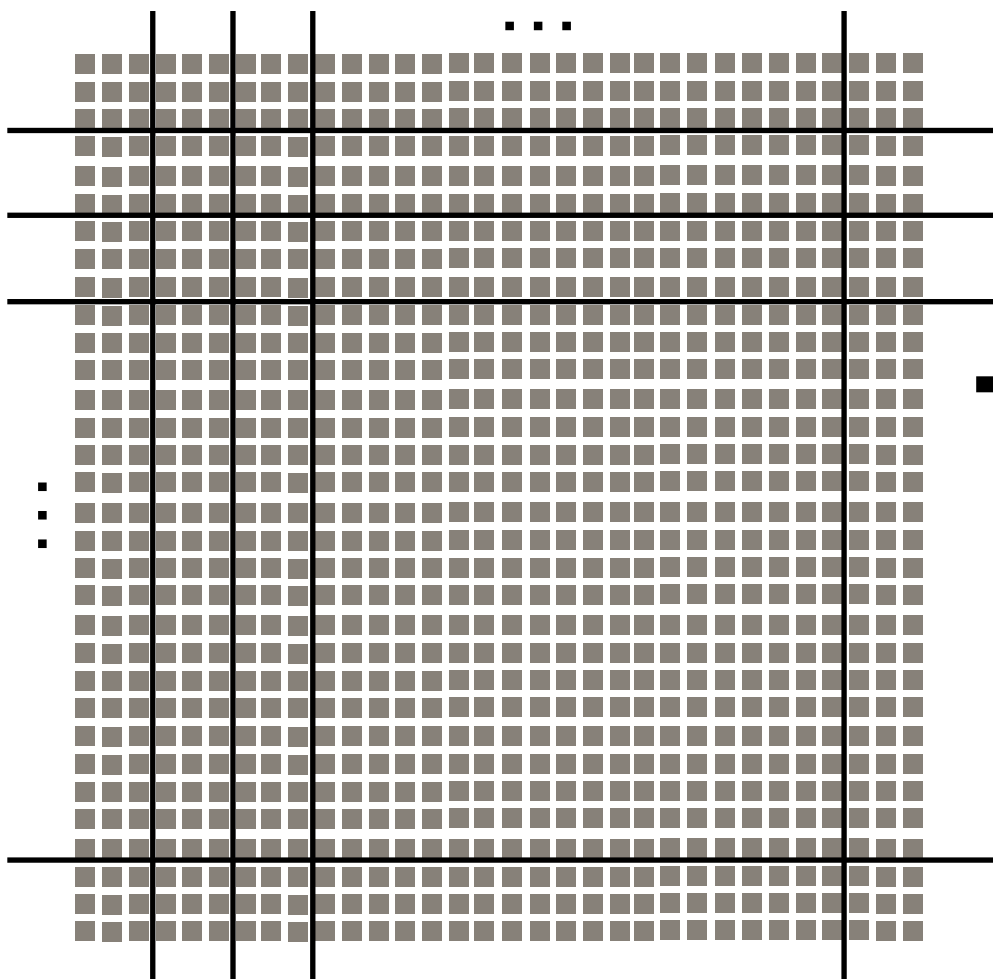
3x3

⋮

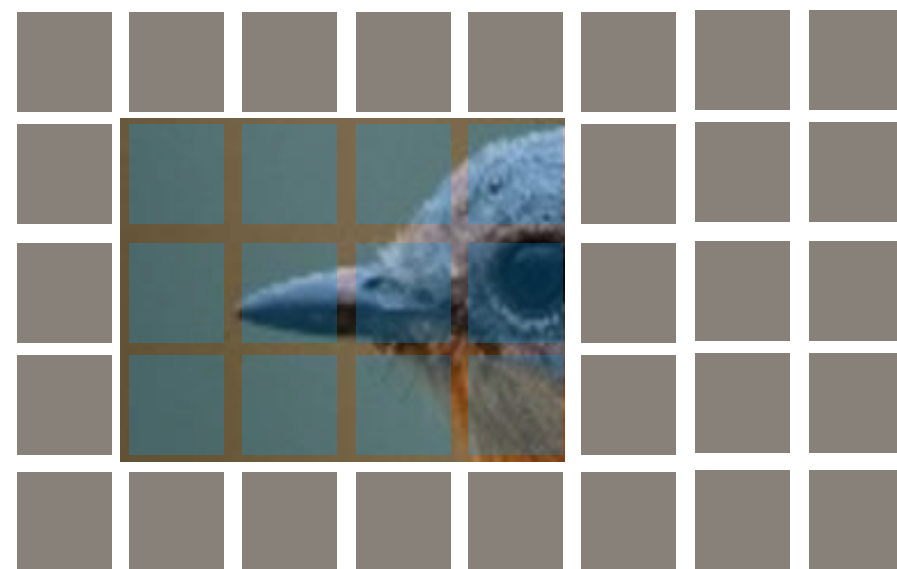




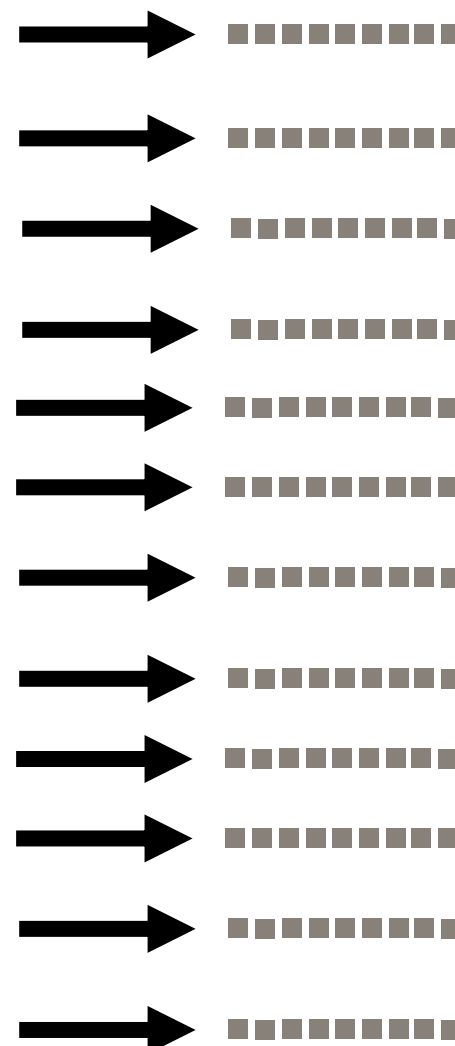
9x12



...

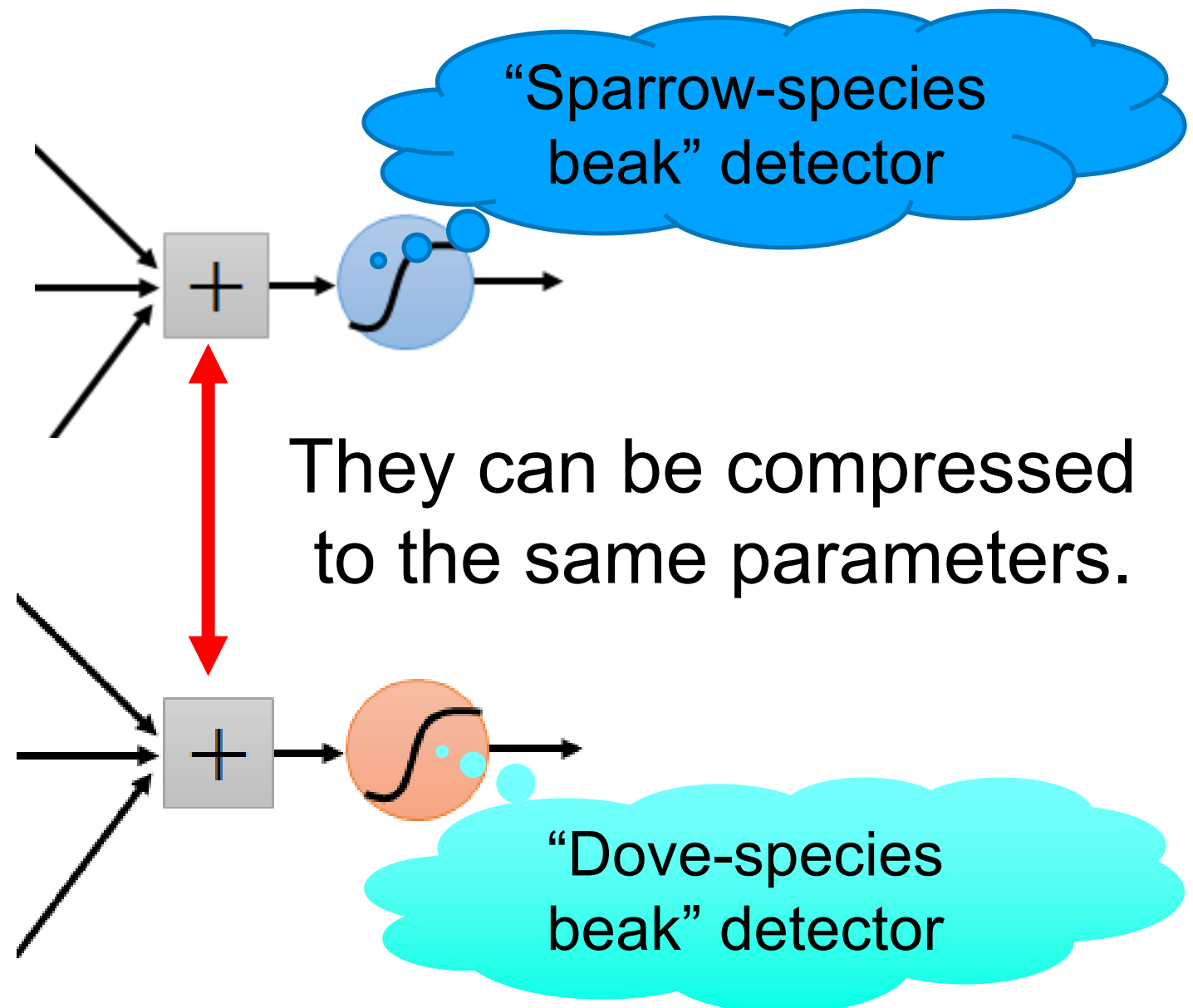
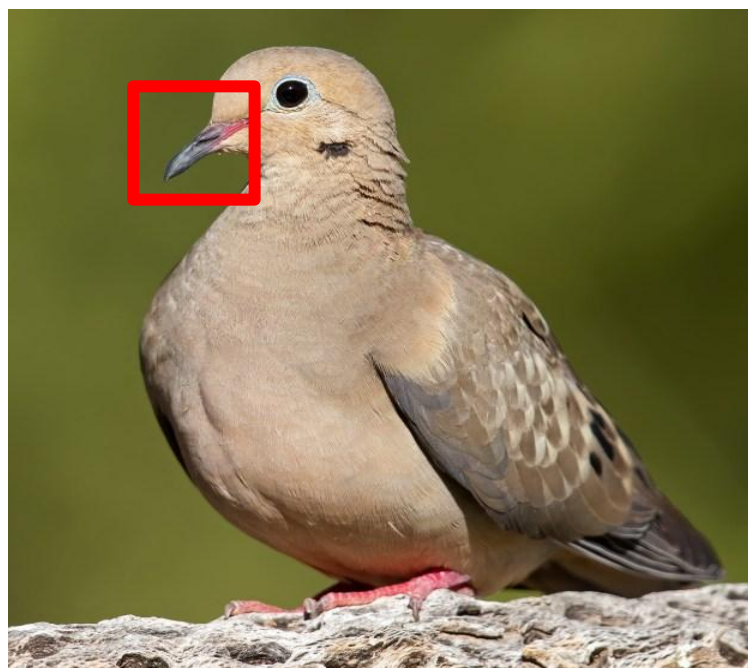
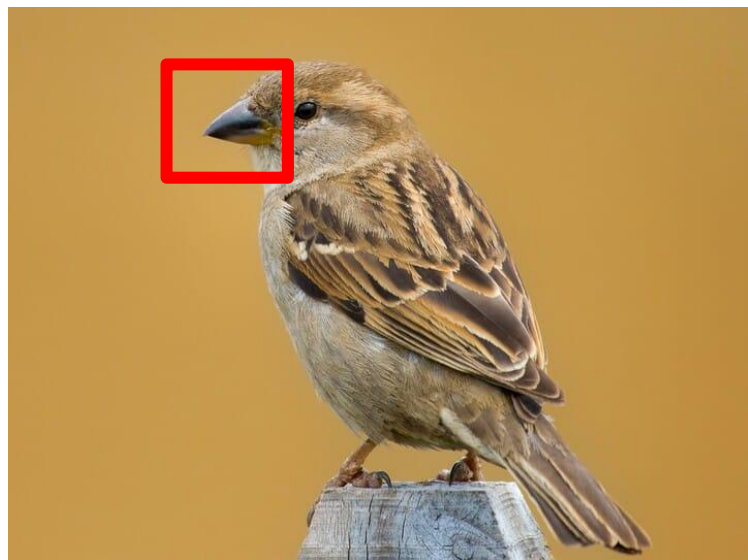


...



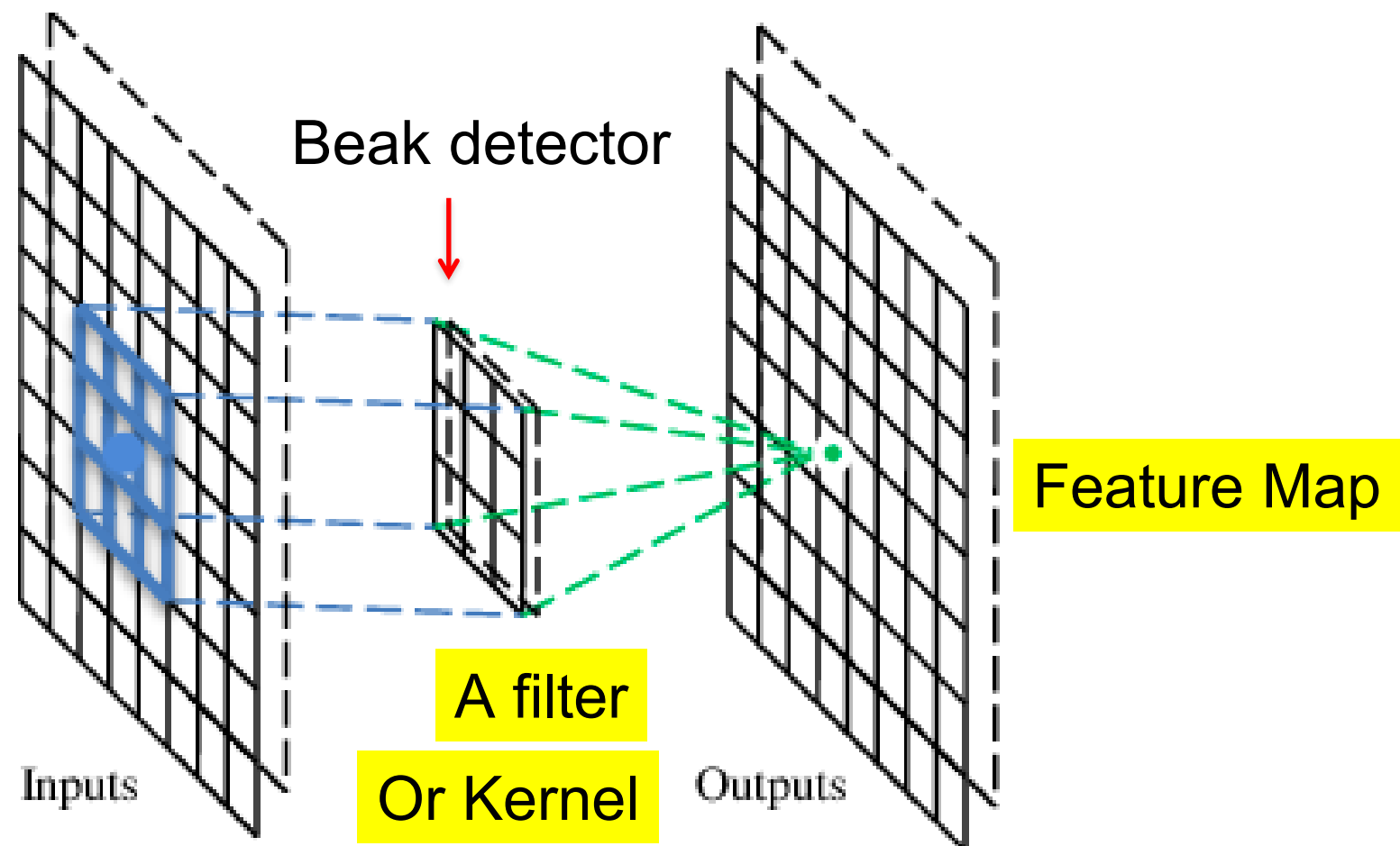
Same pattern appears in different places:
They can be compressed!

What about training a lot of such “small” detectors
and each detector must “move around”.



A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



Convolution

These are the network parameters to be learned.

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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

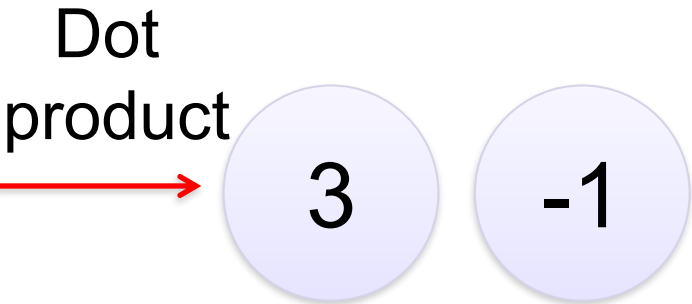
stride=1

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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1



Convolution

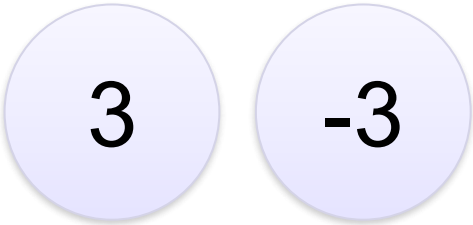
If stride=2

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

6 x 6 image

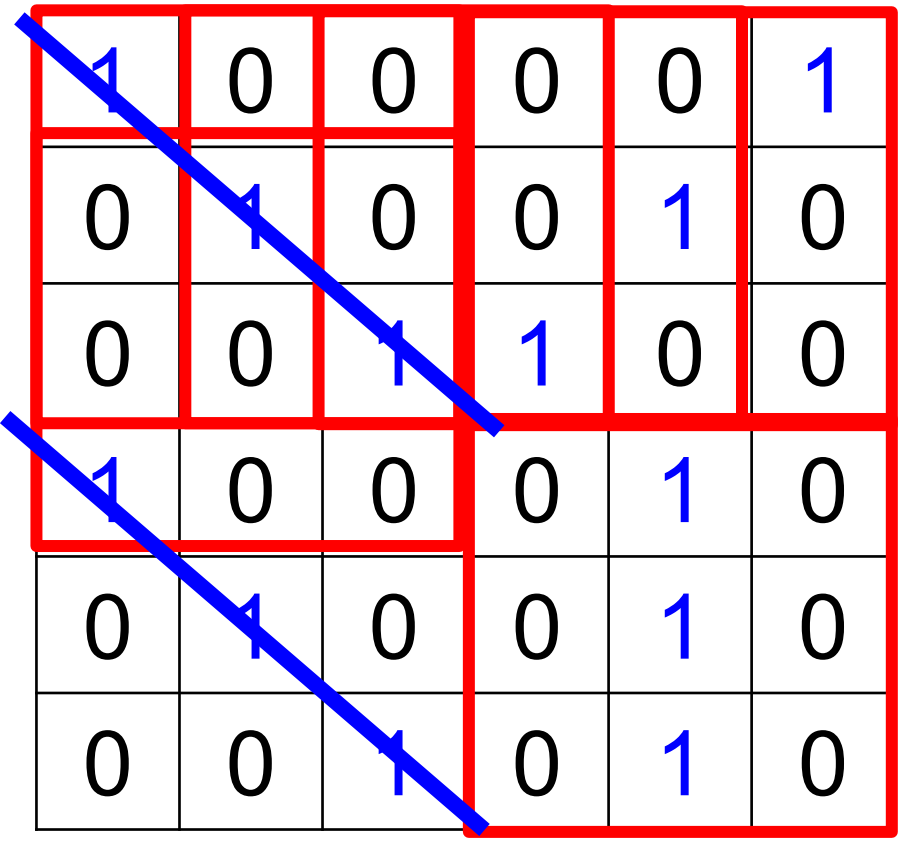
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

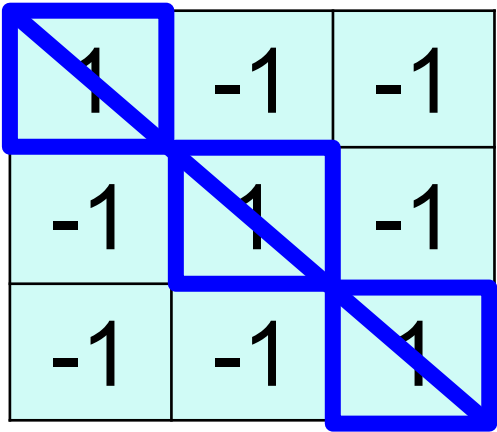


Convolution

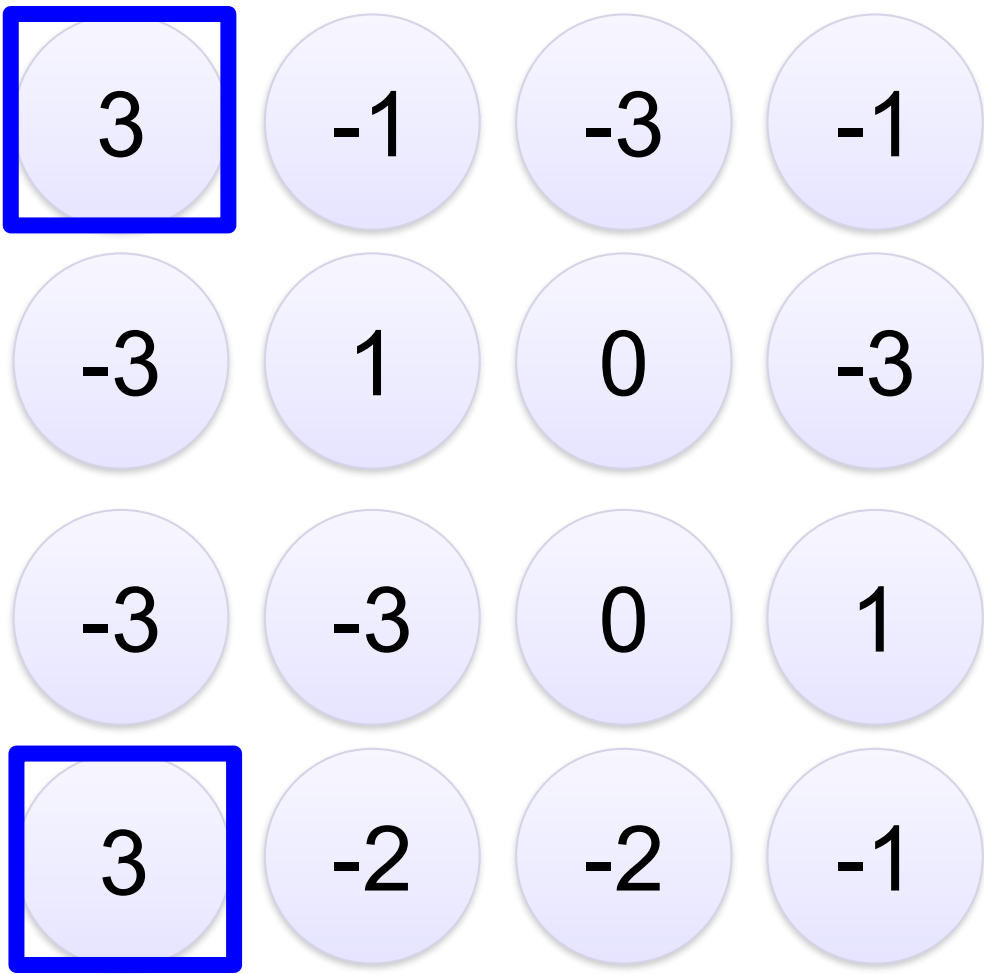
stride=1



6 x 6 image



Filter 1



Convolution

stride=1

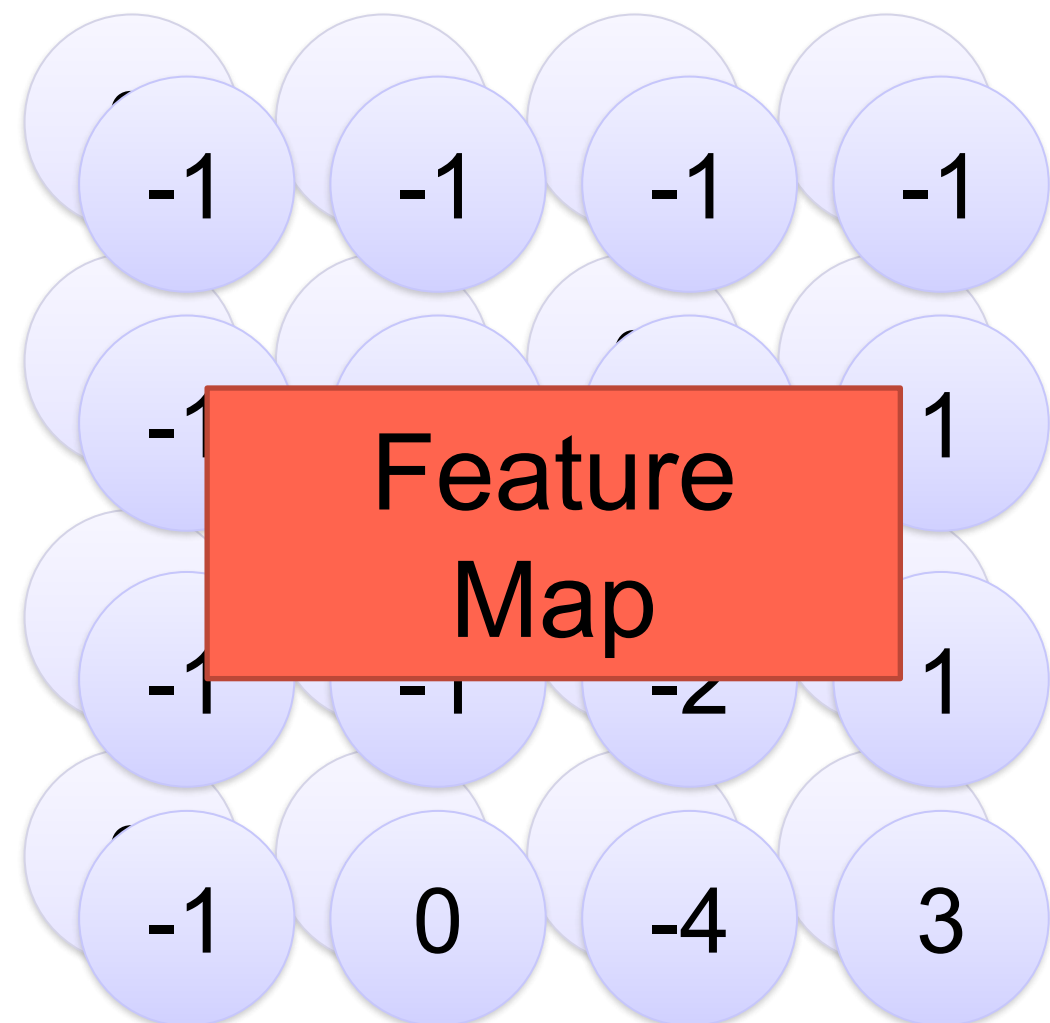
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

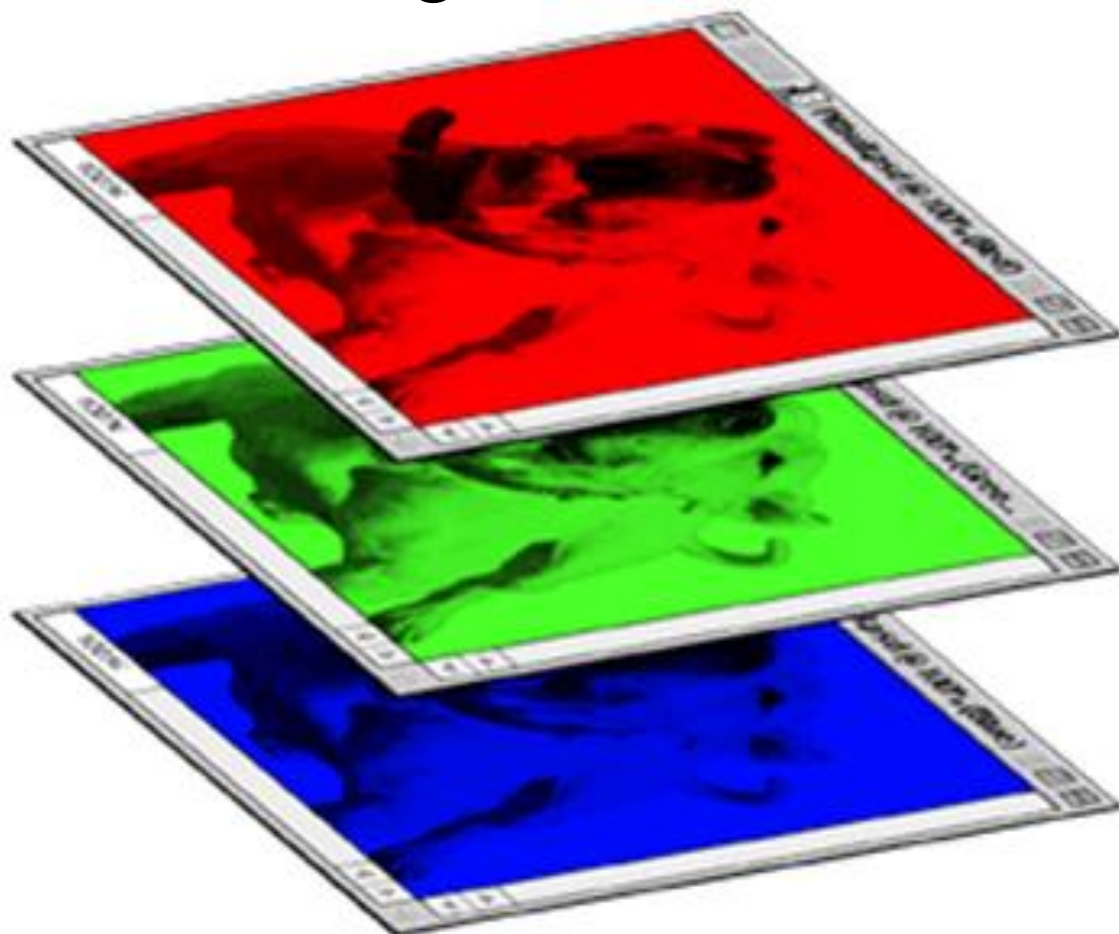
Repeat this for each filter



Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB 3 channels

Color image



1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2



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

Convolution v.s. Fully Connected

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

image

1	-1	-1
-1	1	-1
-1	-1	1

-1	1	-1
-1	1	-1
-1	1	-1

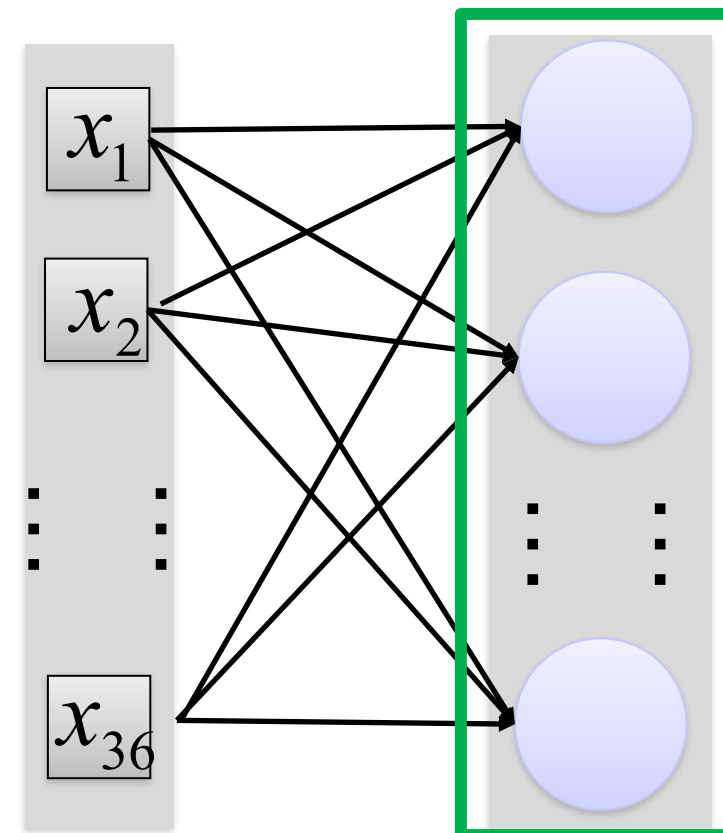


convolution

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

Fully-
connected

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



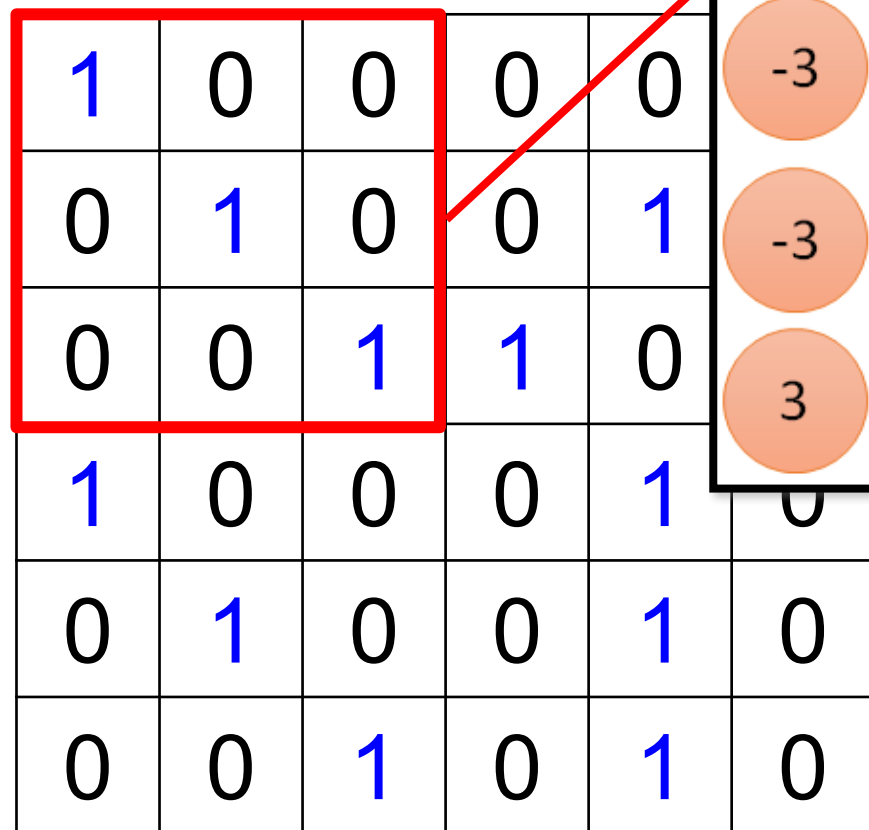
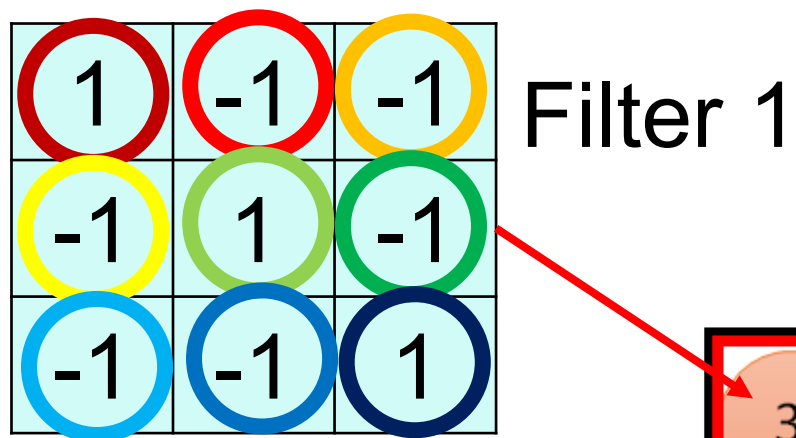
Conventional
Fully Connected
layers
(FC layers)

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

6 x 6 image

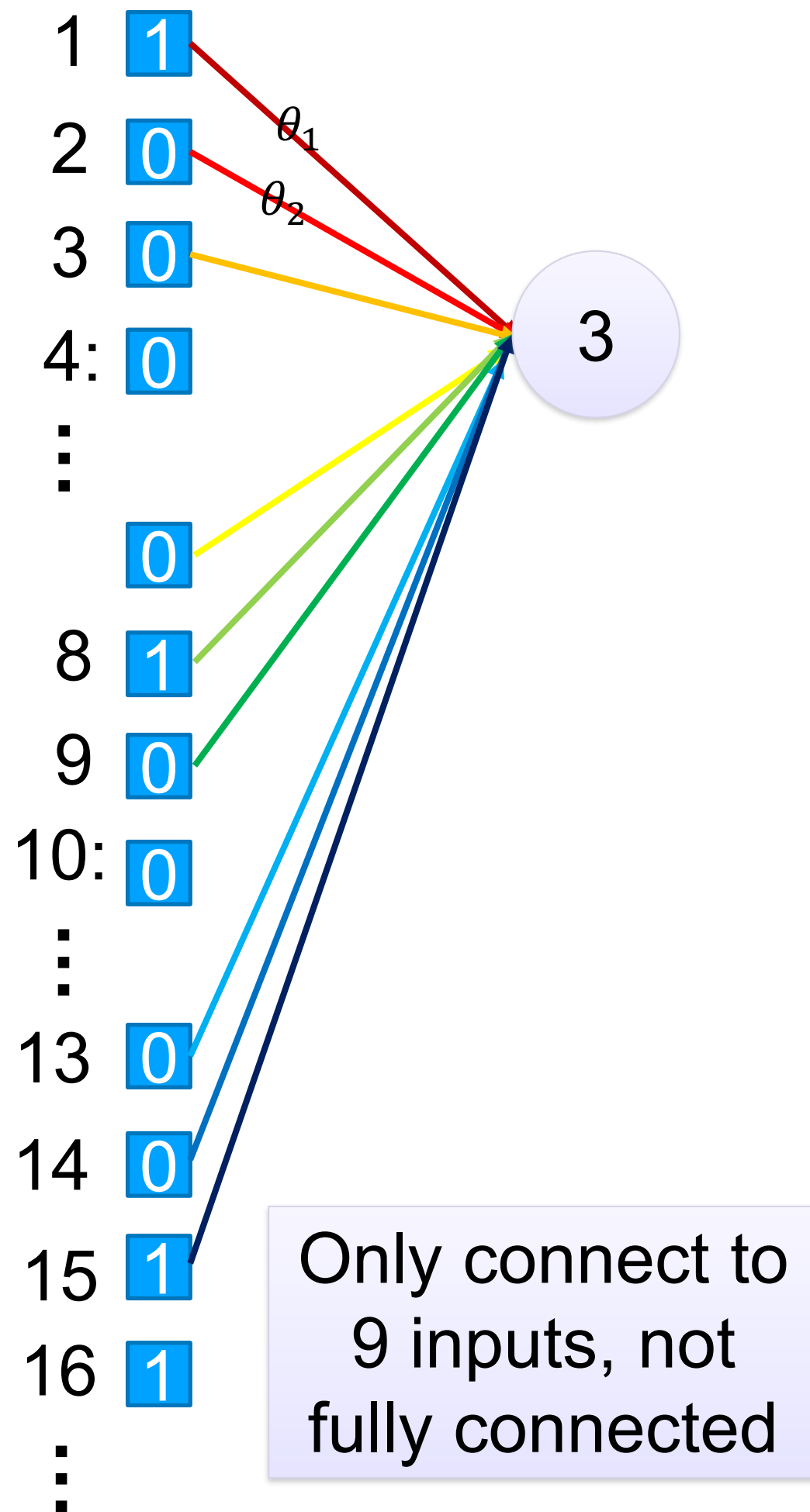
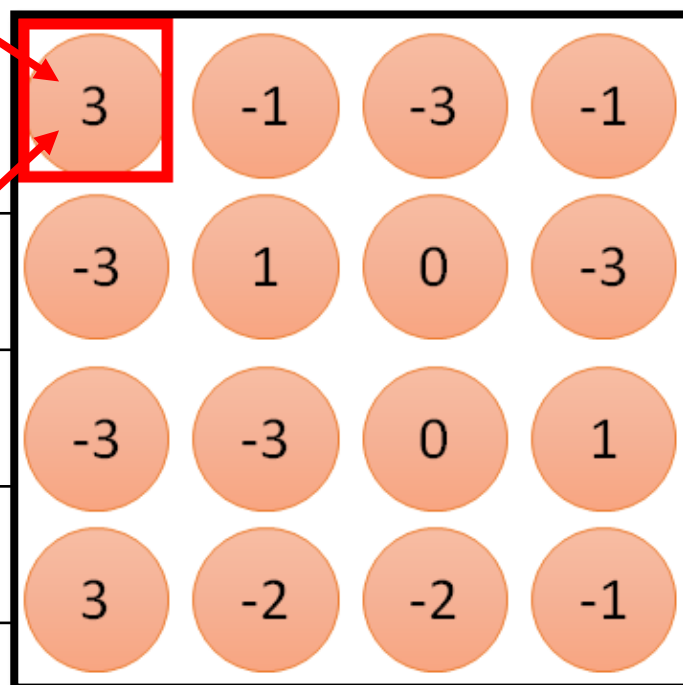
1	1	
2	0	
3	0	
4	0	
5	0	
6	1	
7	0	
8	1	
⋮		⋮
31	0	
32	0	
33	1	
34	0	
35	1	
36	0	

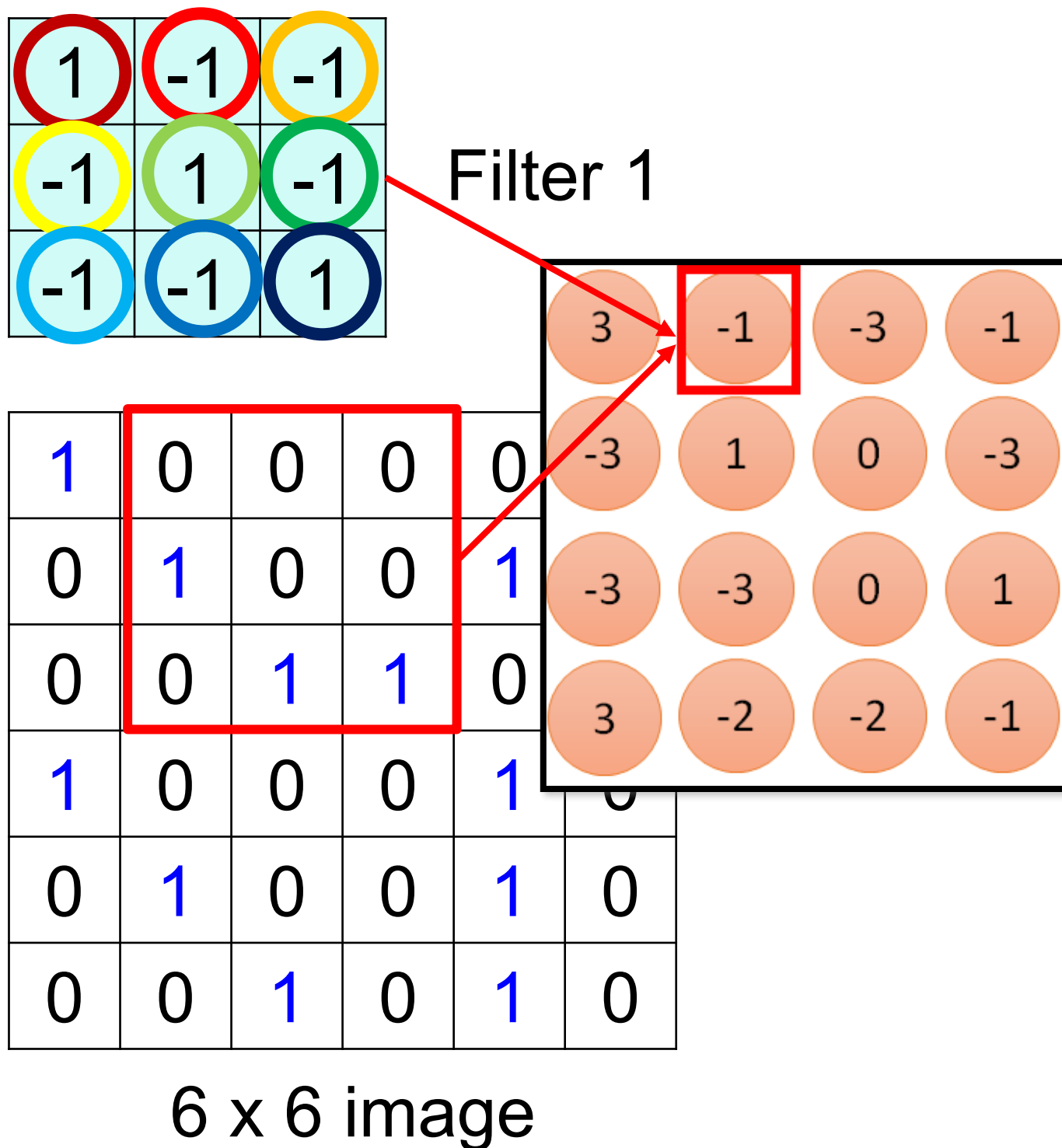
features 1st hidden layer



6 x 6 image

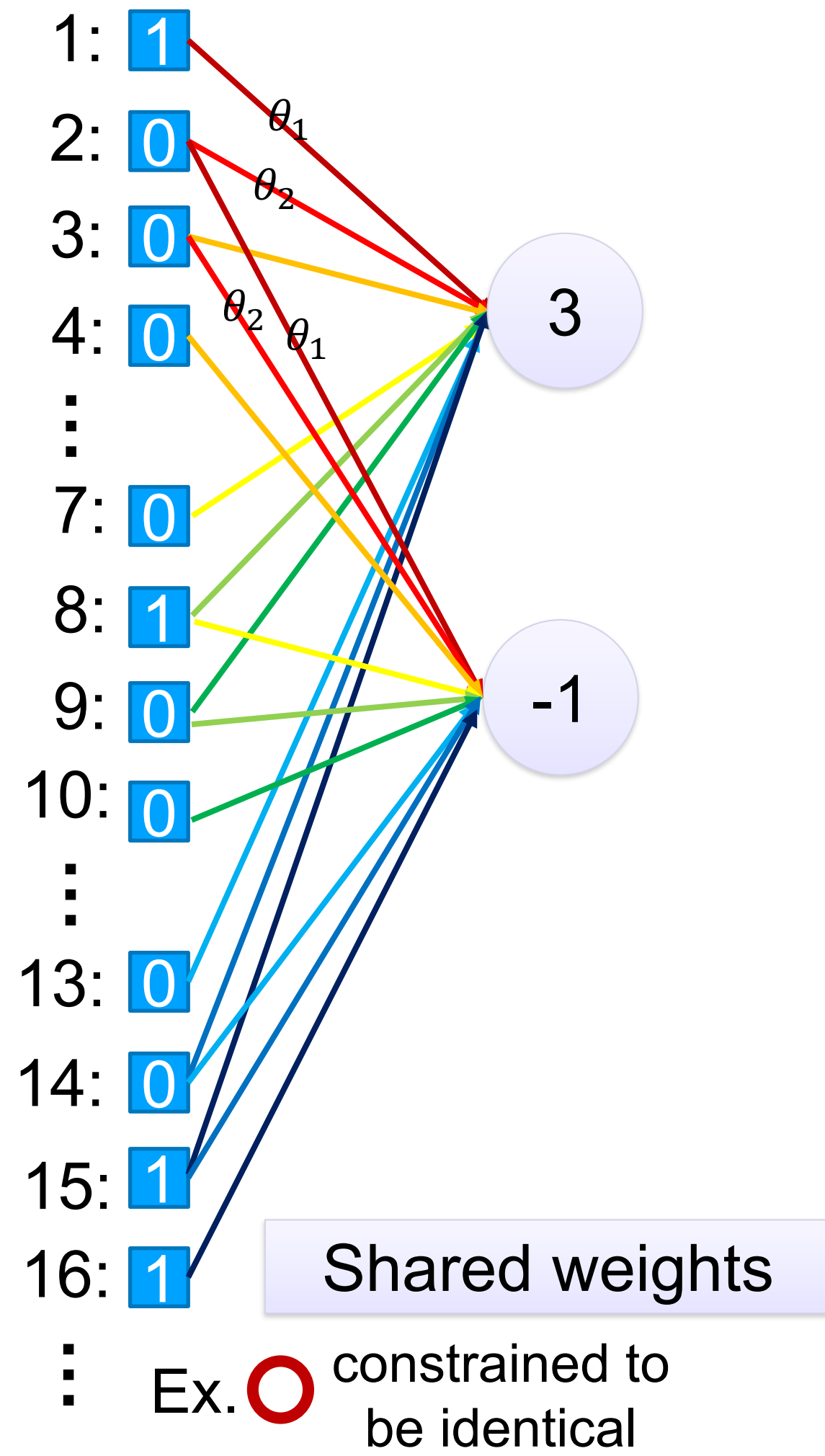
fewer parameters!





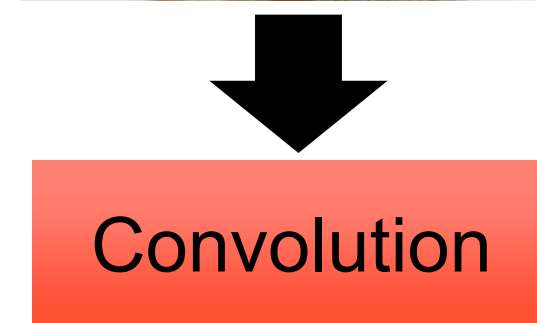
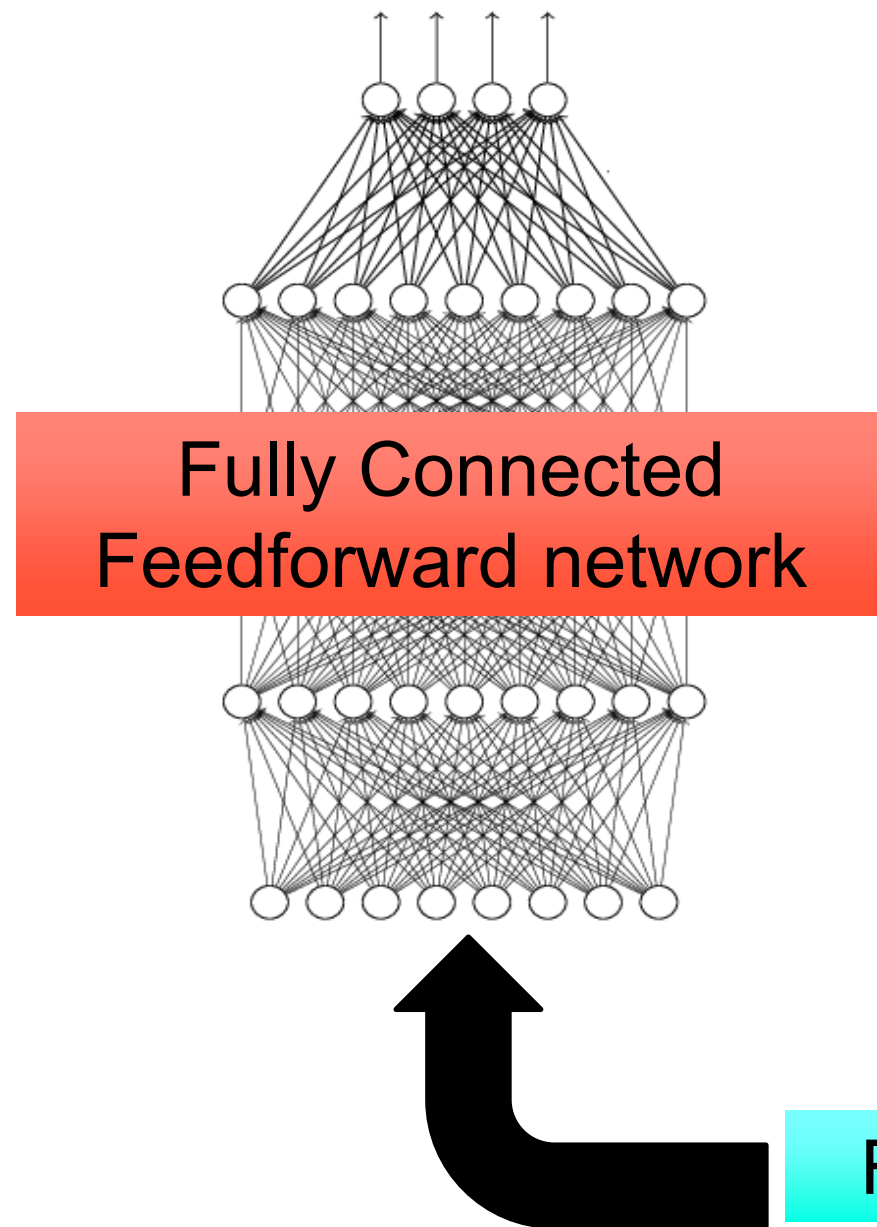
Fewer parameters

Even fewer parameters



The whole CNN

cat dog



Can repeat many times

A red bracket on the right side of the diagram groups the "Convolution" and "Max Pooling" boxes from the first stage, indicating that this sequence can be repeated multiple times.

Flattened

A thick black arrow points from the second "Max Pooling" box to a cyan rectangular box labeled "Flattened".

Max Pooling

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

Why Pooling

- Subsampling pixels will not change the object
bird

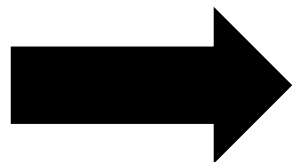


Subsampling



bird

We can subsample the pixels to make image smaller



fewer parameters to characterize the image

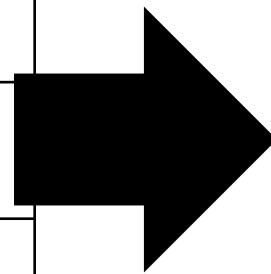
A CNN compresses a fully connected network in three ways:

- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

Max Pooling

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

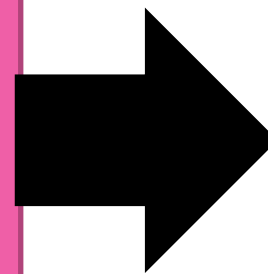
6 x 6 image



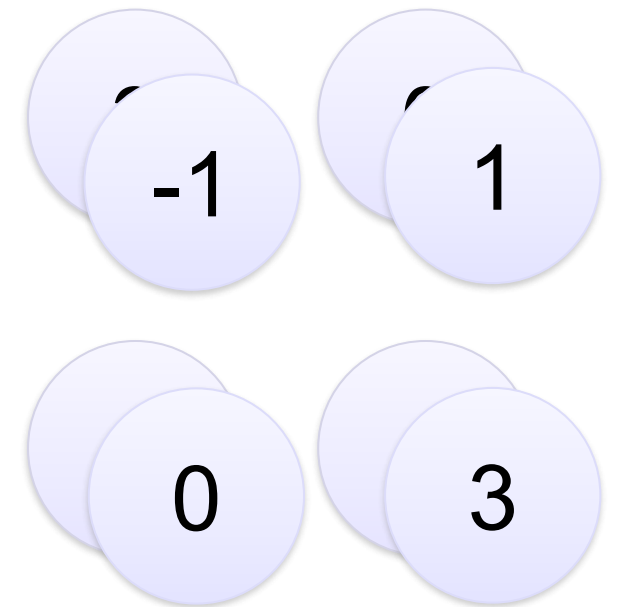
Conv



Max
Pooling



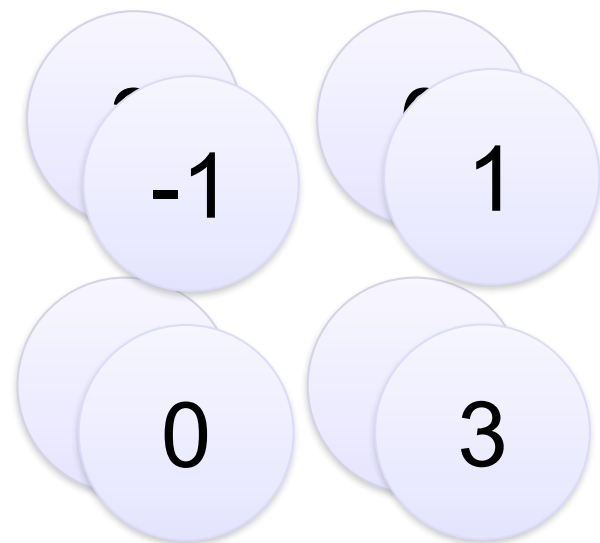
New image
but smaller



2 x 2 image

Each filter
is a channel

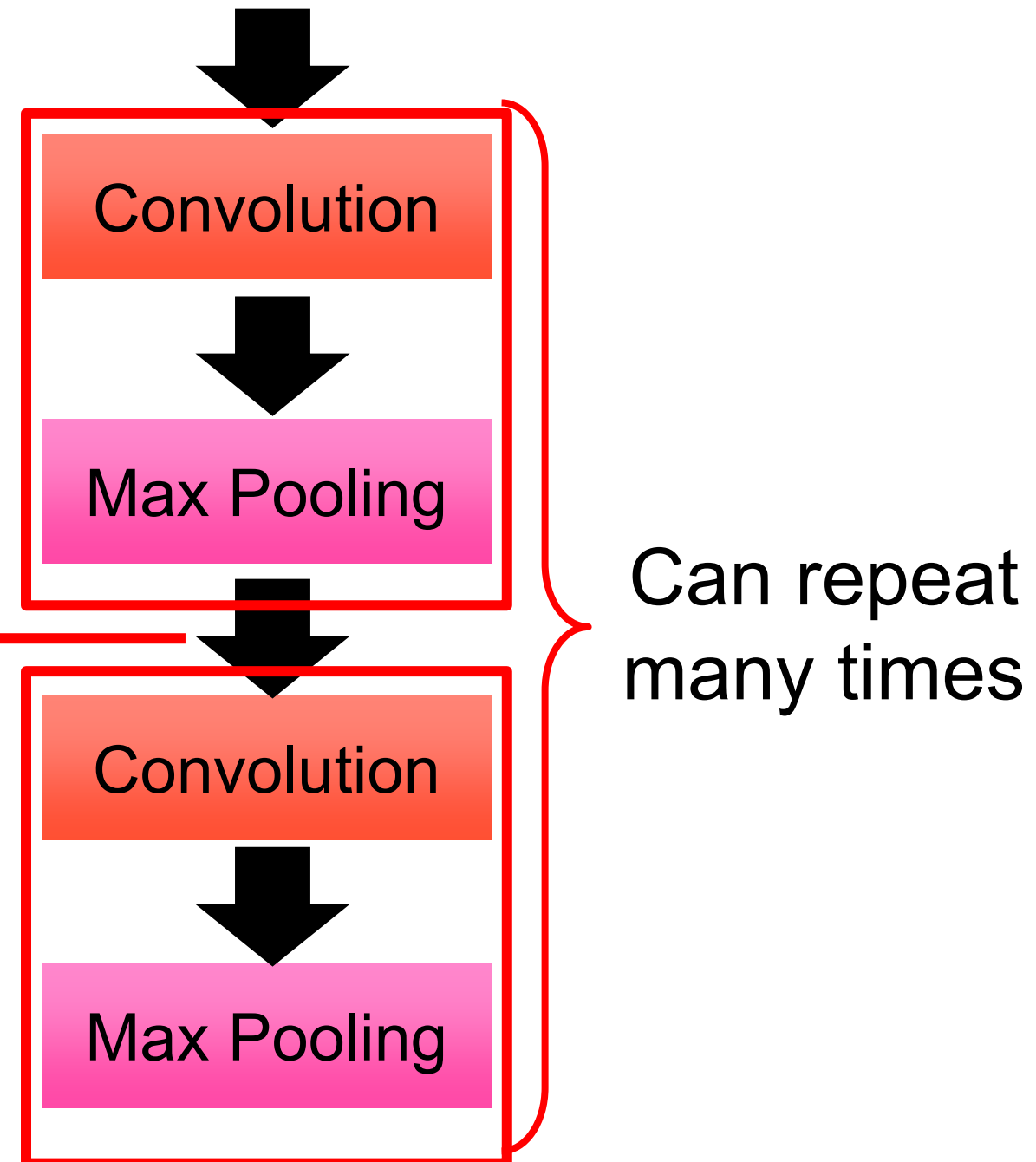
The whole CNN



A new image

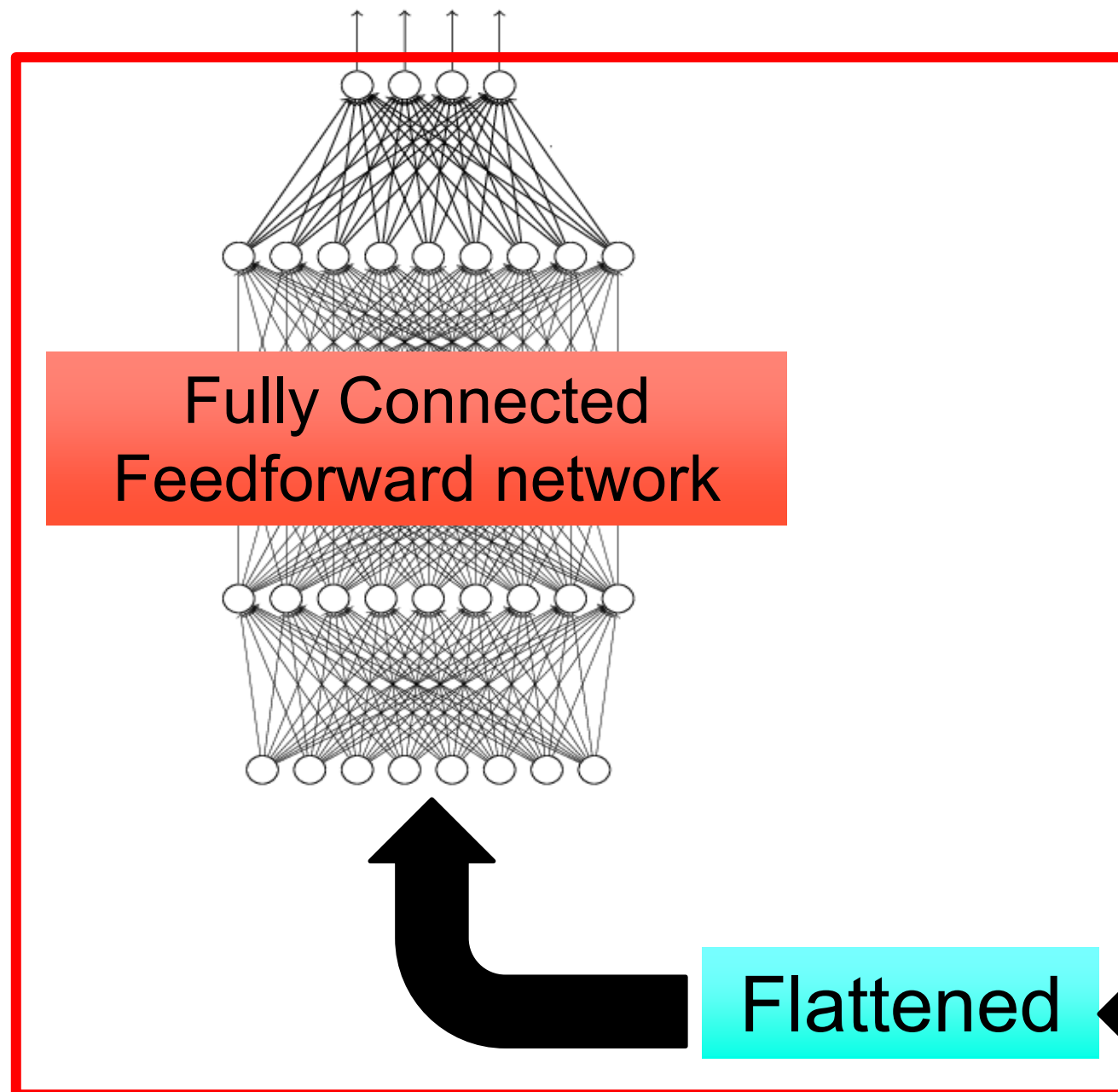
Smaller than the original image

The number of channels is the number of filters



The whole CNN

cat dog



Convolution

Max Pooling

A new image

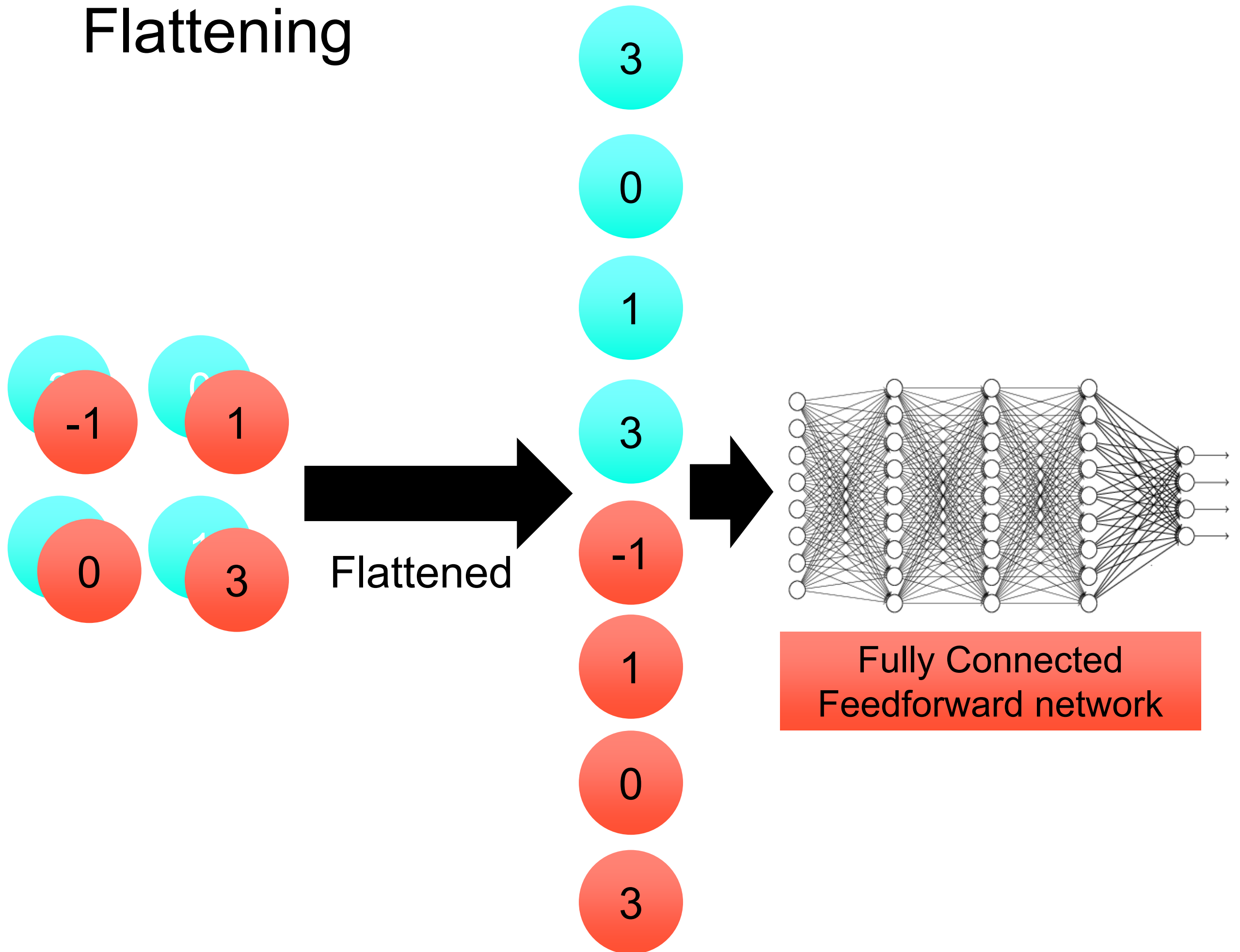
Convolution

Max Pooling

A new image

Flattened

Flattening



CNN in Keras

Only modified the *network structure* and *input format* (vector -> 3-D tensor)

```
model2.add( Convolution2D( 25,3,3,  
                           input_shape=(28,28,1)) )
```

1	-1	-1	1	-1
-1	1	-1	1	-1
-1	-1	-1	1	-1

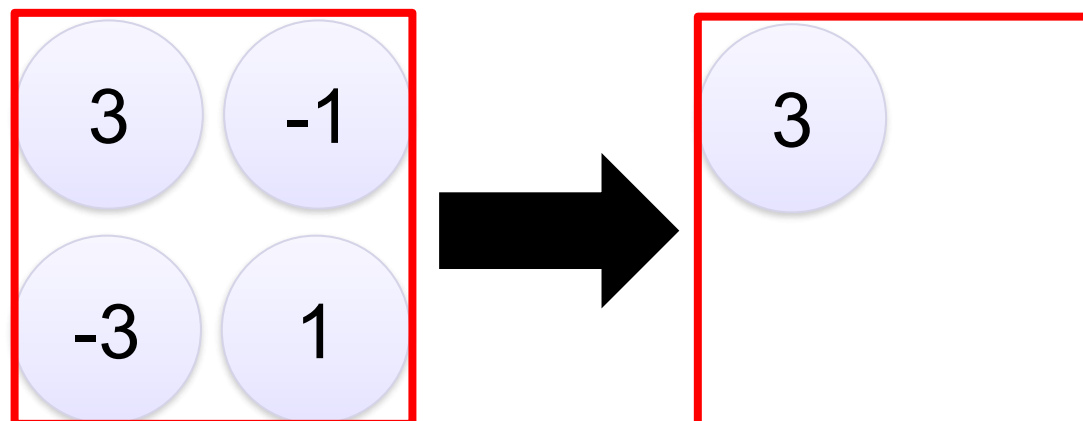
There are
25 3x3
filters.

Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

```
model2.add(MaxPooling2D((2,2)))
```



input

Convolution

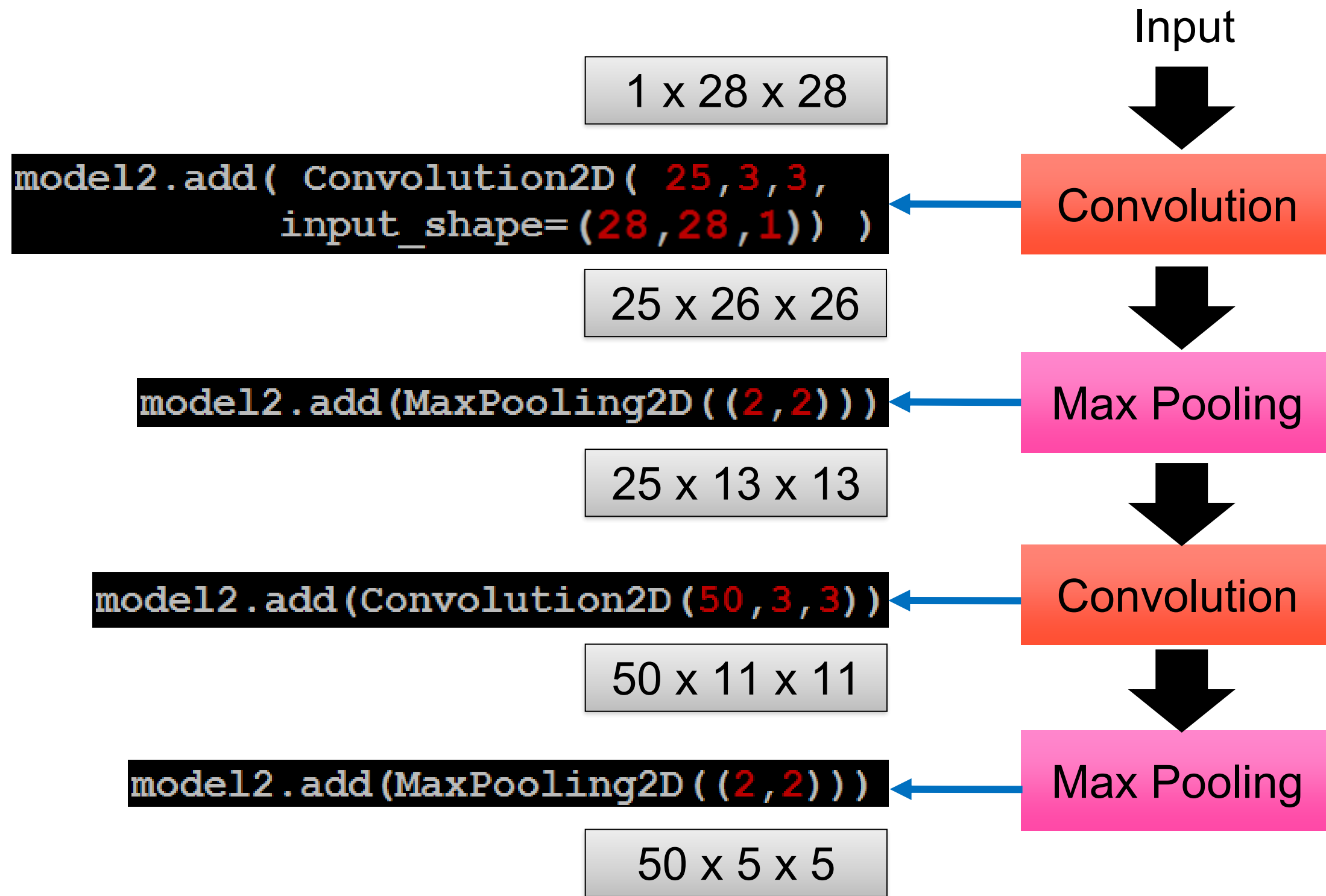
Max Pooling

Convolution

Max Pooling

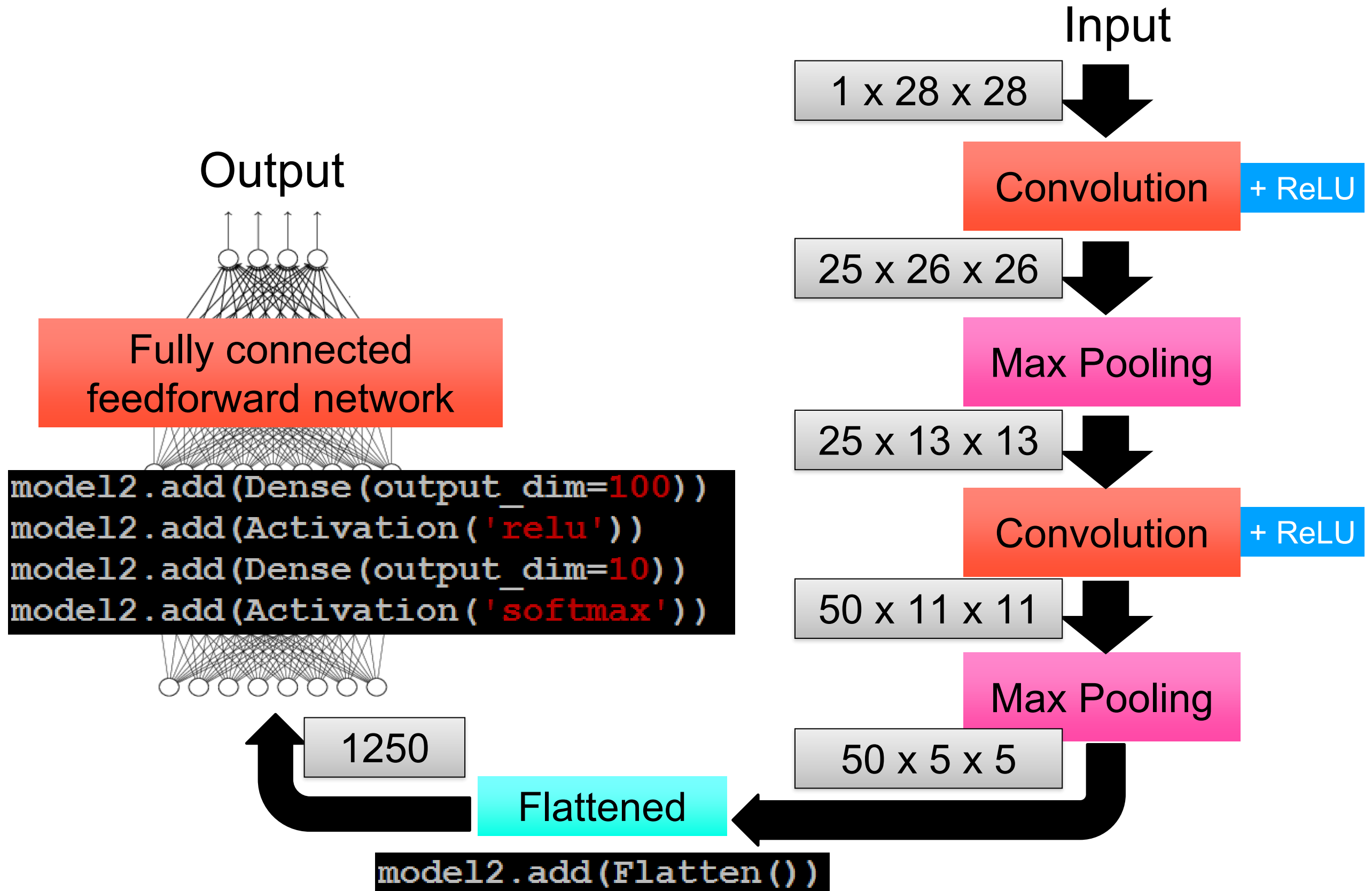
CNN in Keras

Only modified the *network structure* and *input format (vector -> 3-D array)*



CNN in Keras

Only modified the *network structure* and *input format* (vector -> 3-D array)



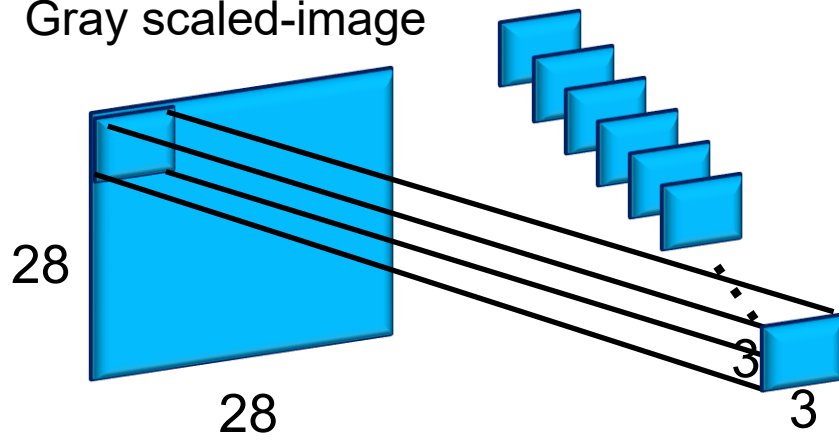
Number of Parameters

$25 \times 3 \times 3 + 25$ parameters

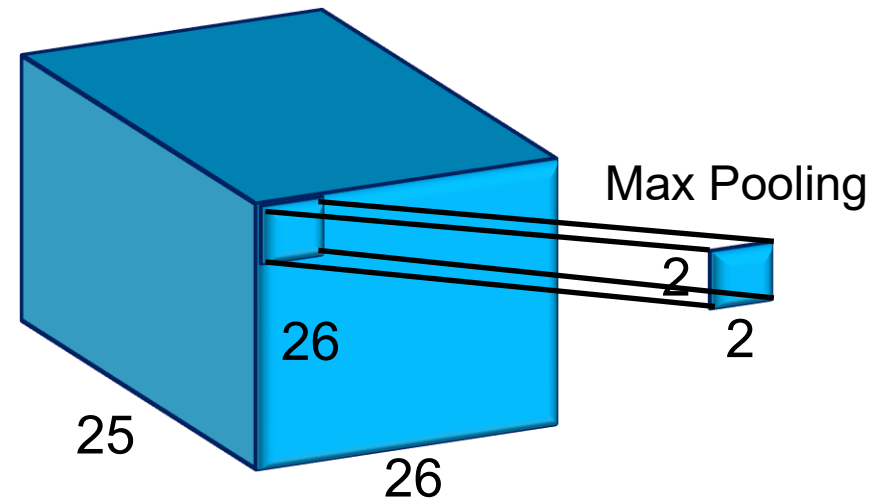
25 filters - Conv1

25: 3×3

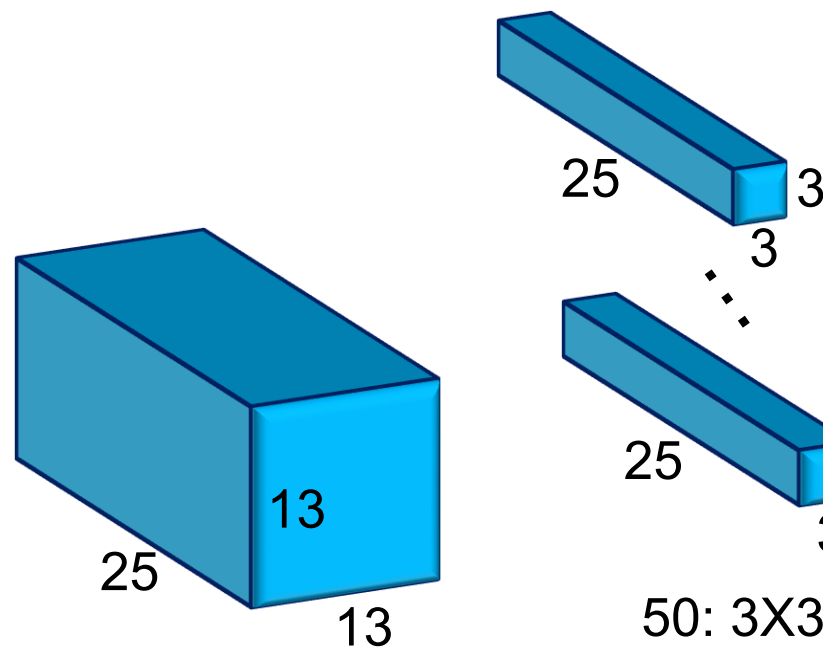
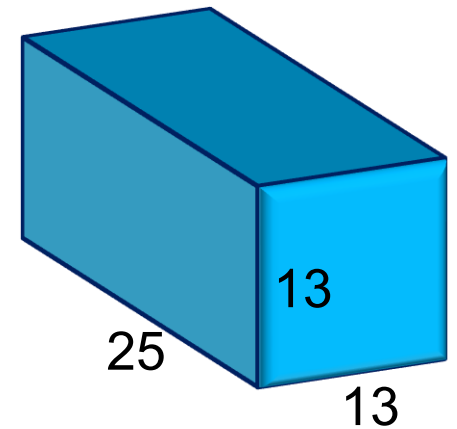
Gray scaled-image



Convolved result for 25 filters



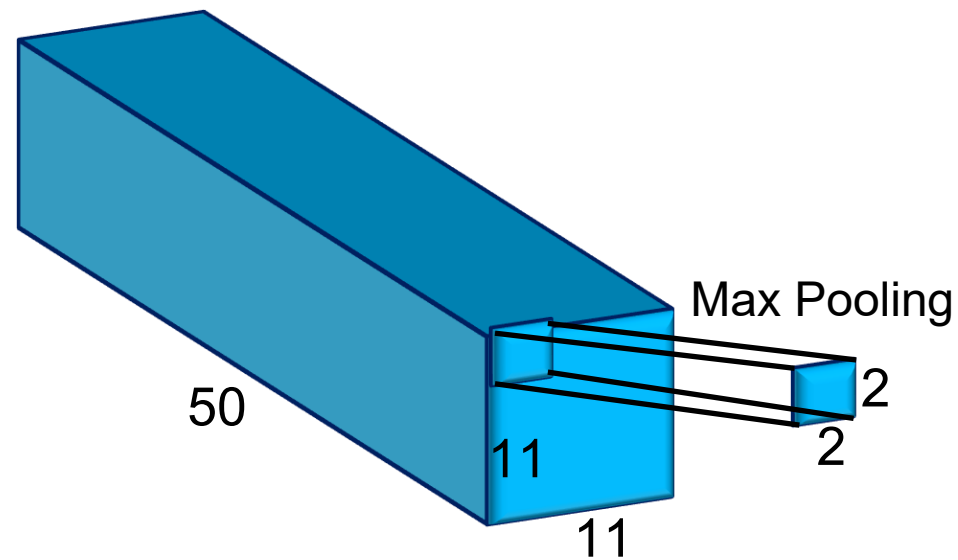
Result after Max Pooling



50: $3 \times 3 \times 25$

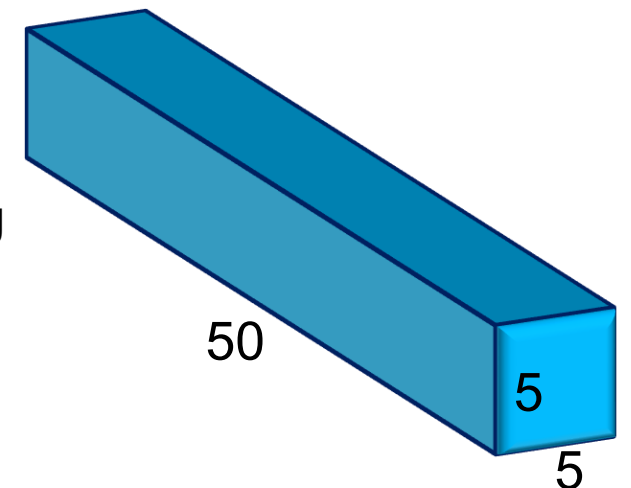
50 filters - Conv2

$50 \times 3 \times 3 \times 25 + 50$ parameters



Convolved result for 50 filters

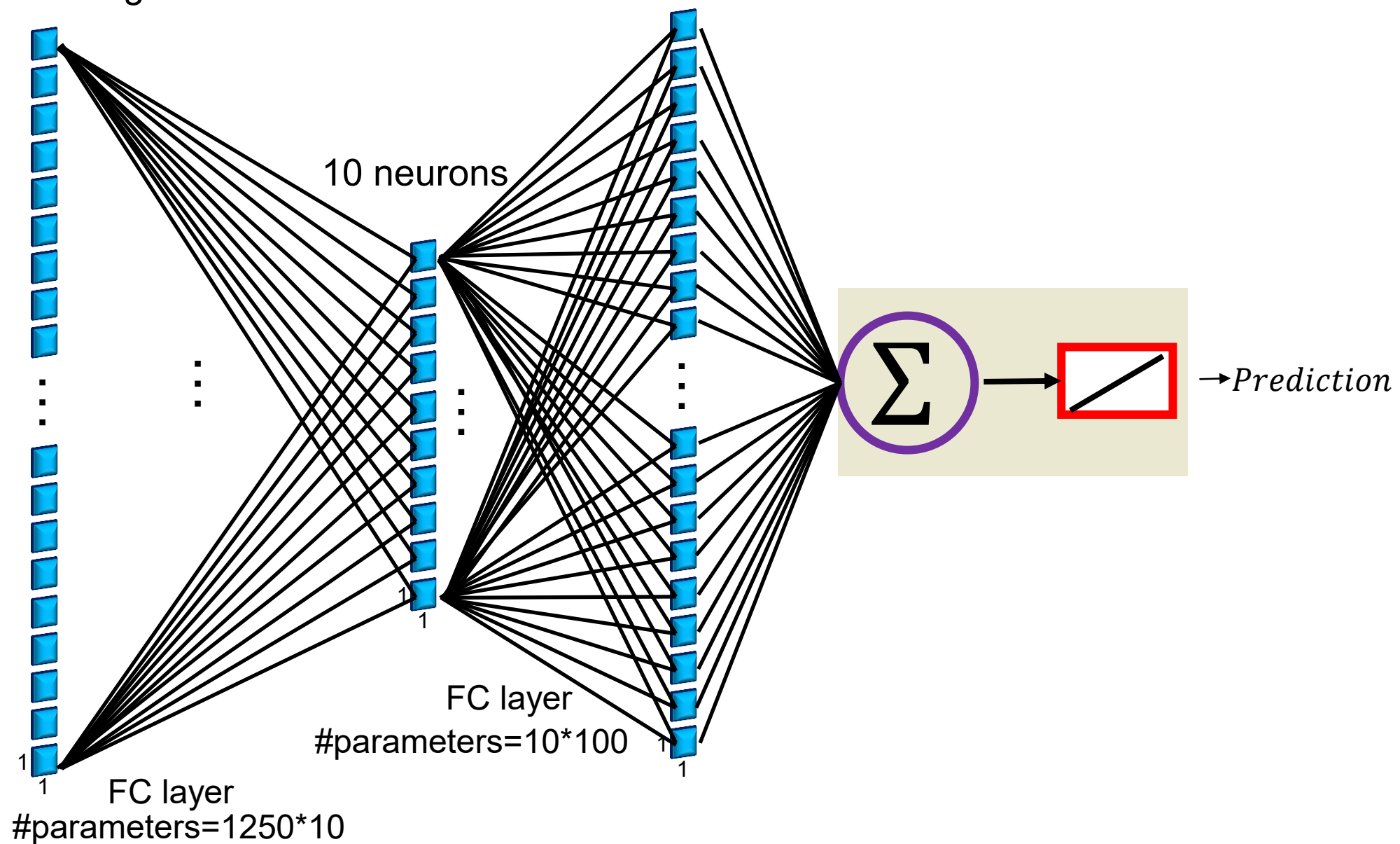
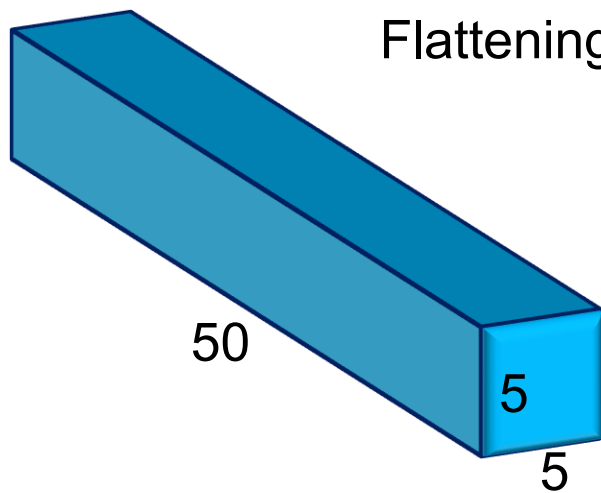
Result after Max Pooling



neurons after flattening: $50 \times 5 \times 5 = 1250$

100 neurons

Flattening



10 CNN Architecture