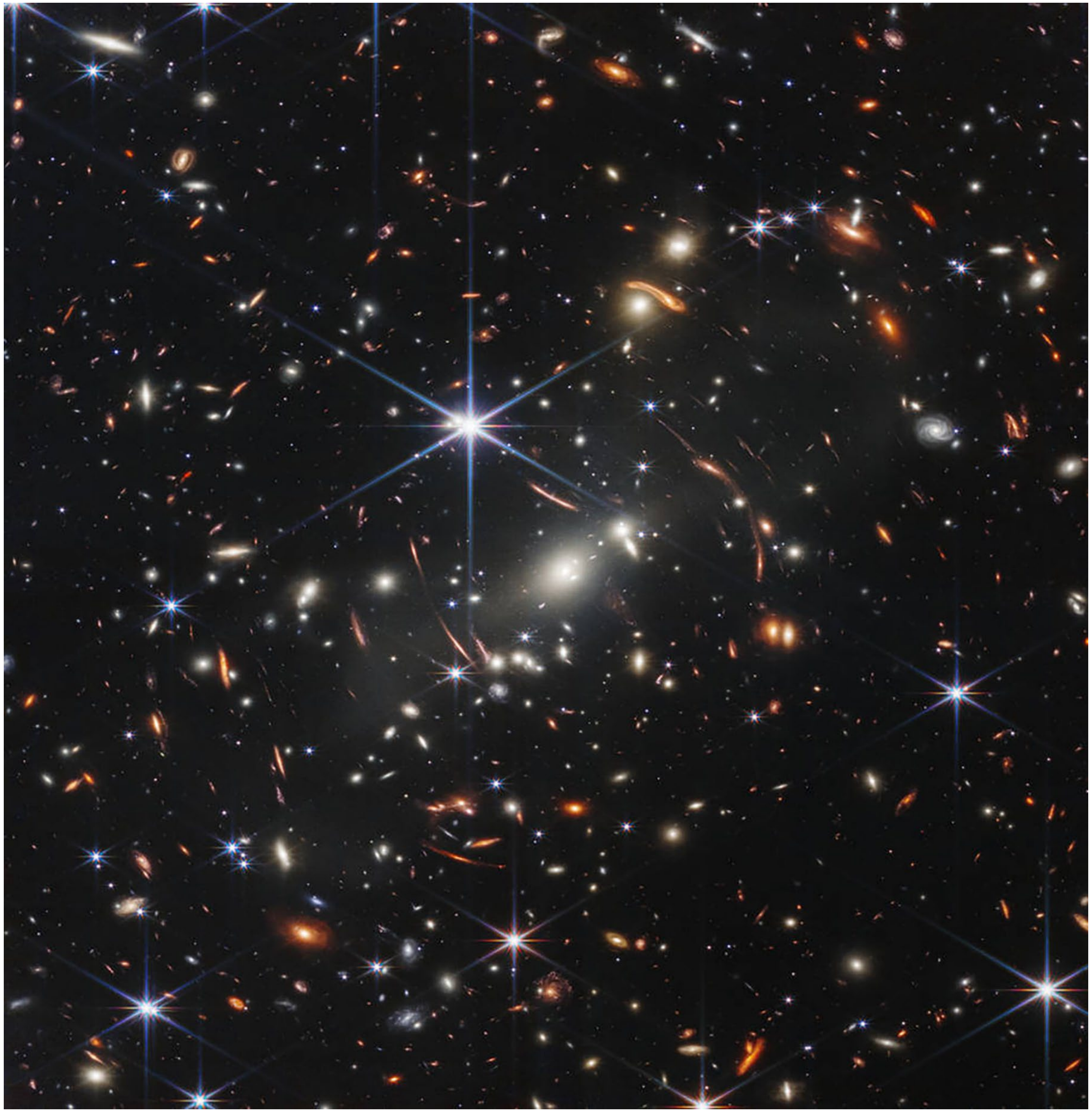
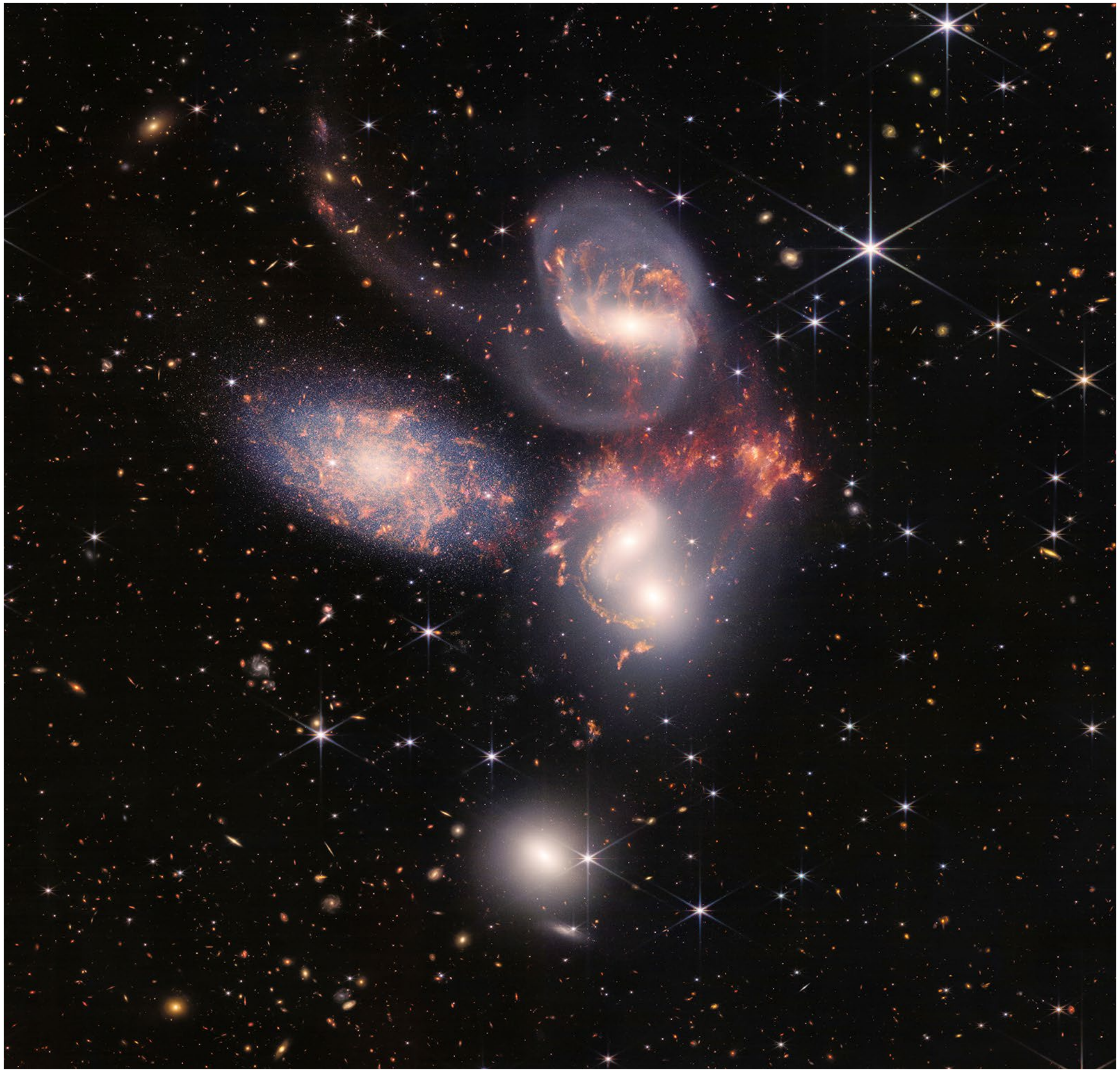
James Webb Space Telescope



The landscape of “mountains” and “valleys”

A visual grouping of five galaxies



CONVOLUTIONAL NEURAL NETWORK

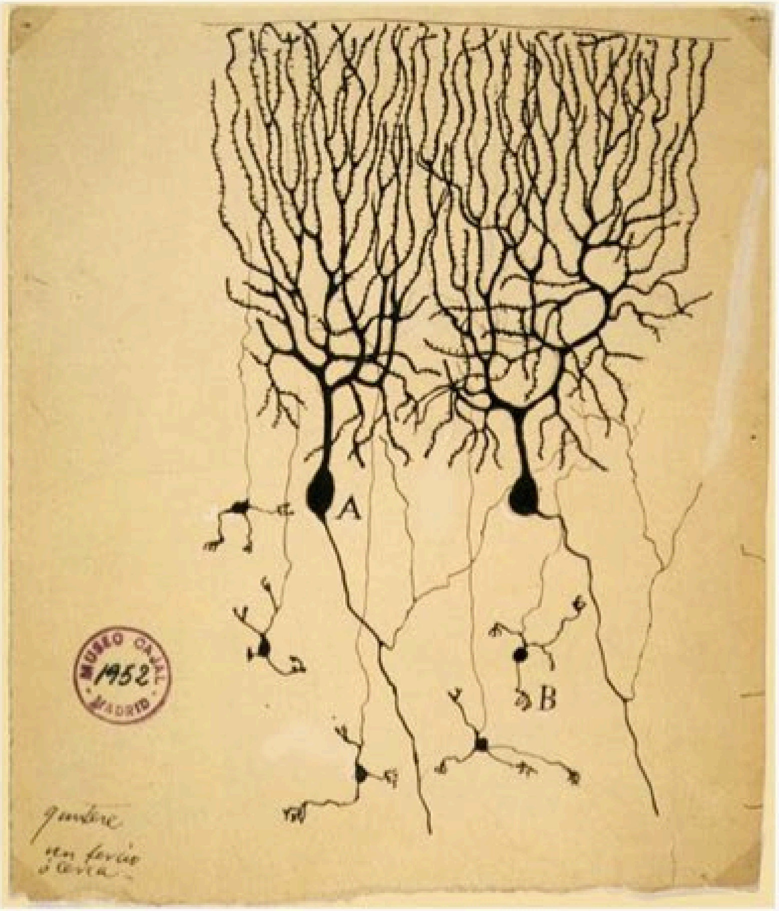
Mahdi Roozbahani

Georgia Tech

Great visualization tool:
https://poloclub.github.io/cnn-explainer/

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

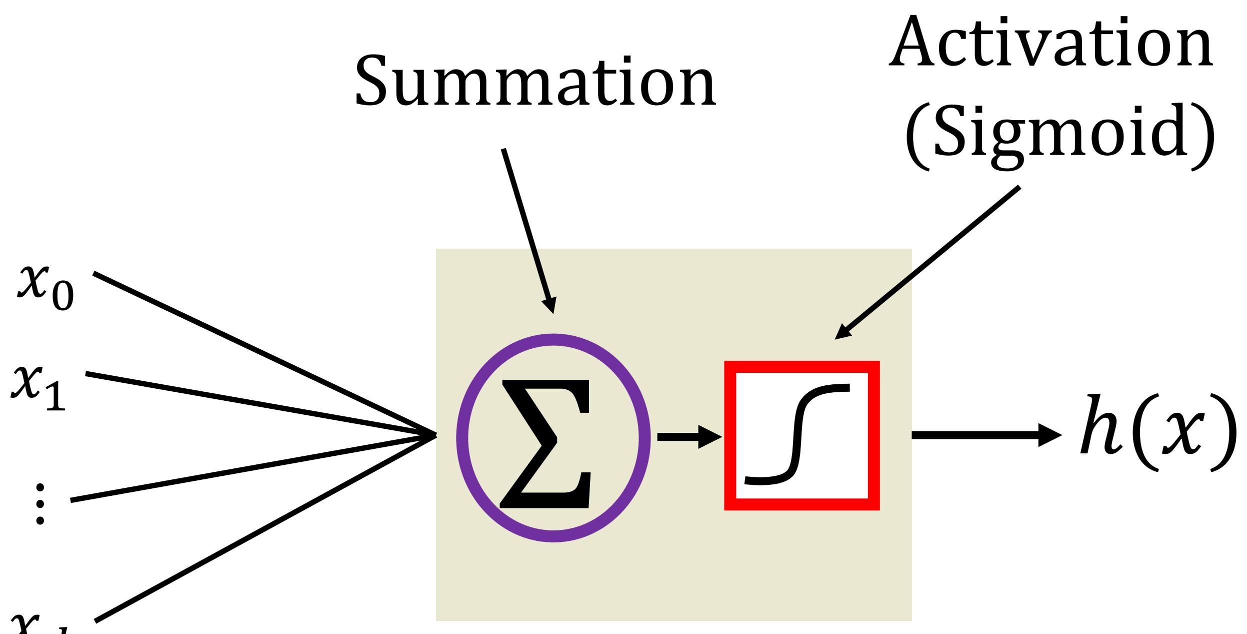
Inspiration from Biological Neurons



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

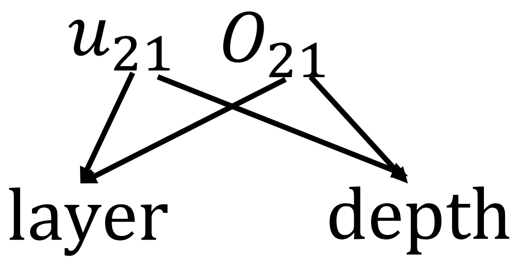


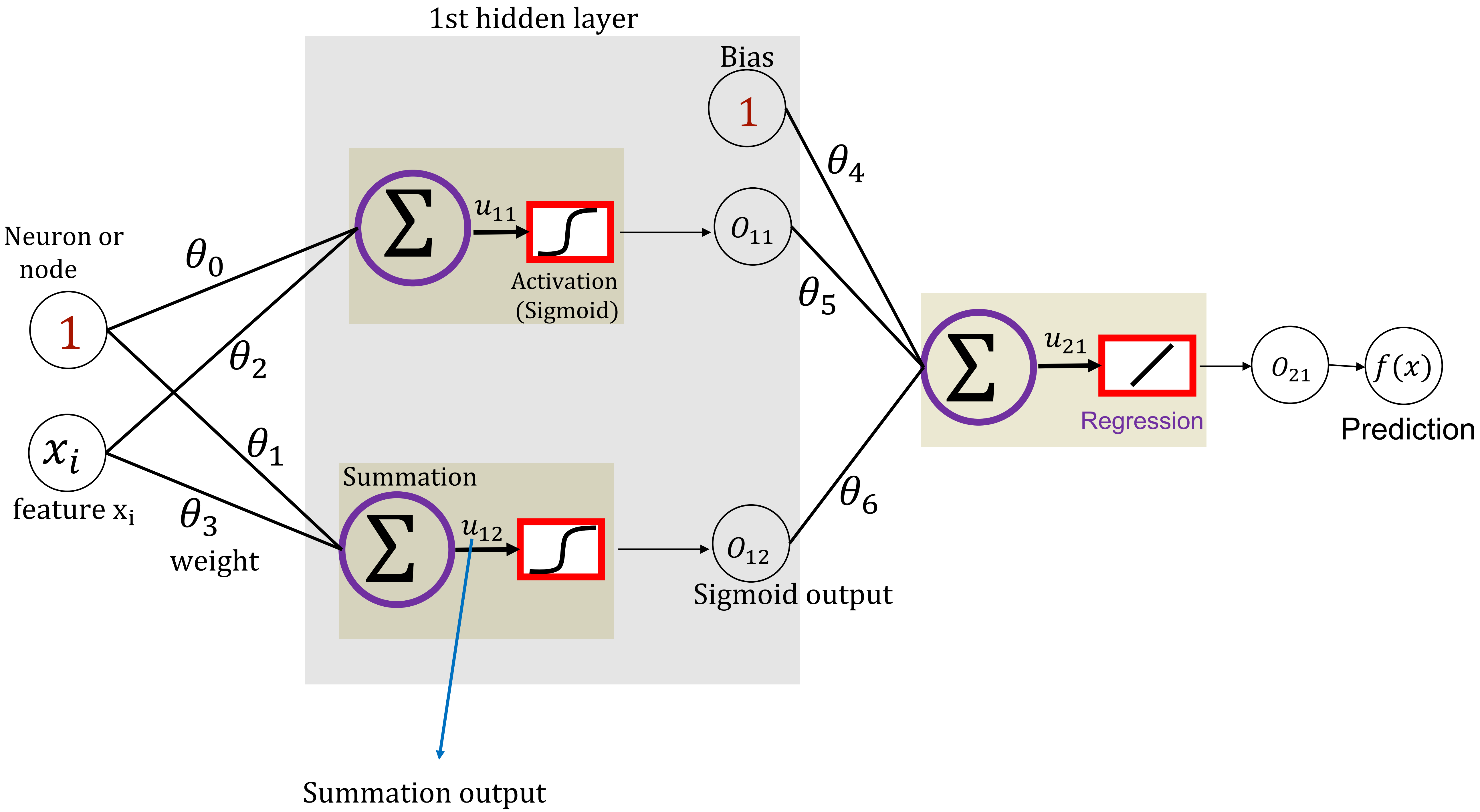
$h(x) = u(x\theta) \rightarrow$ logistic regression

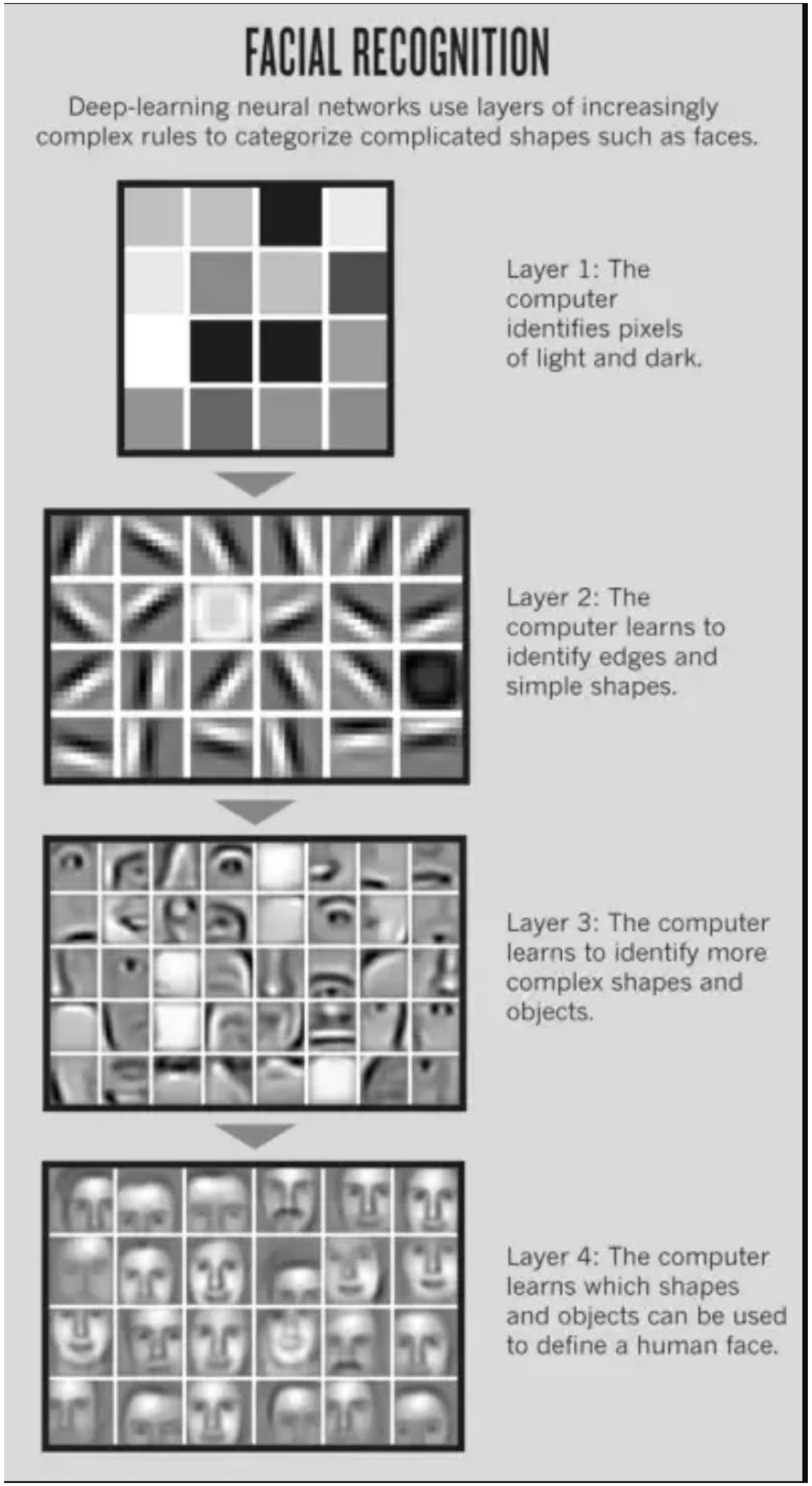
$$output = activation(x\theta \;+ b)$$

Name of the neuron	Activation function: $activation(z)$
Linear unit	$x\theta$
Threshold/sign unit	$sign(x\theta)$
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.

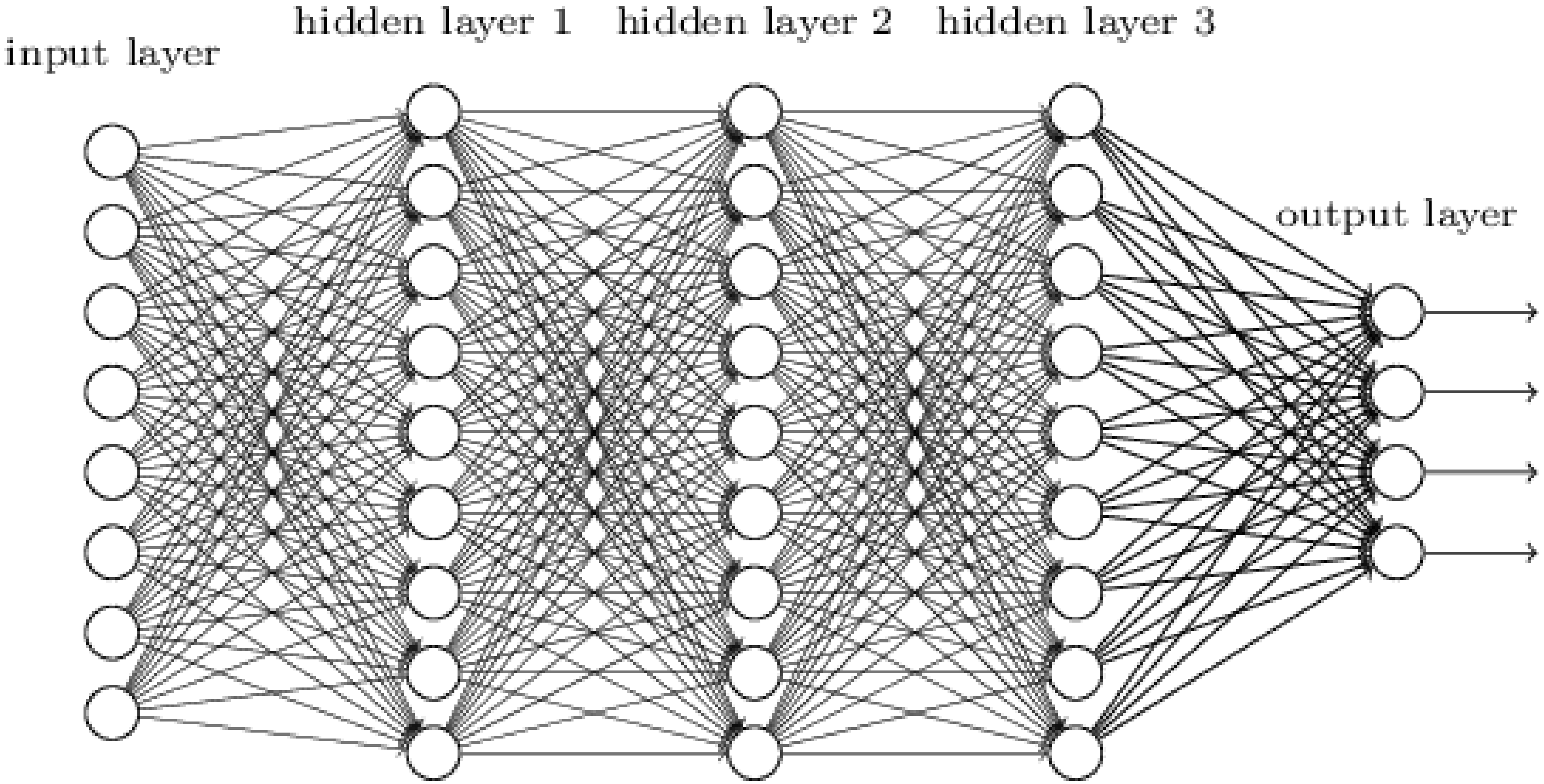
Layer 2: The computer learns to identify edges and simple shapes.

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

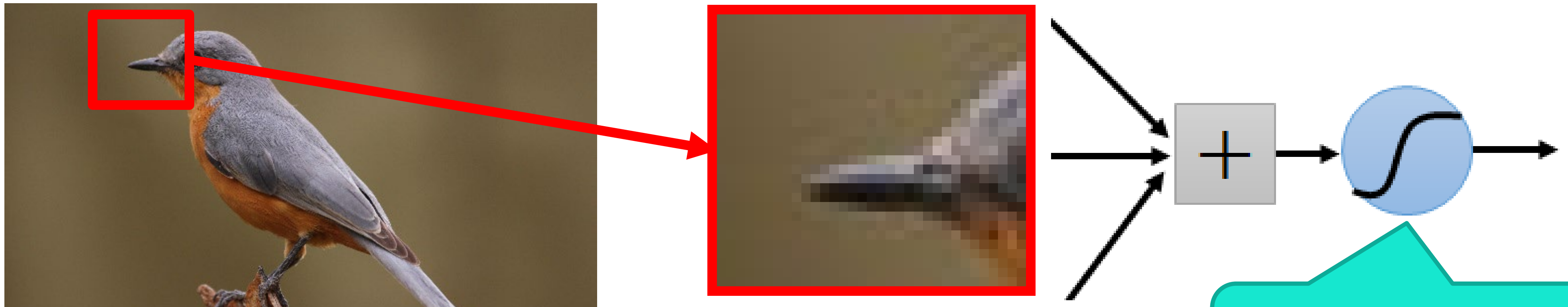
- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?



Consider learning an image:

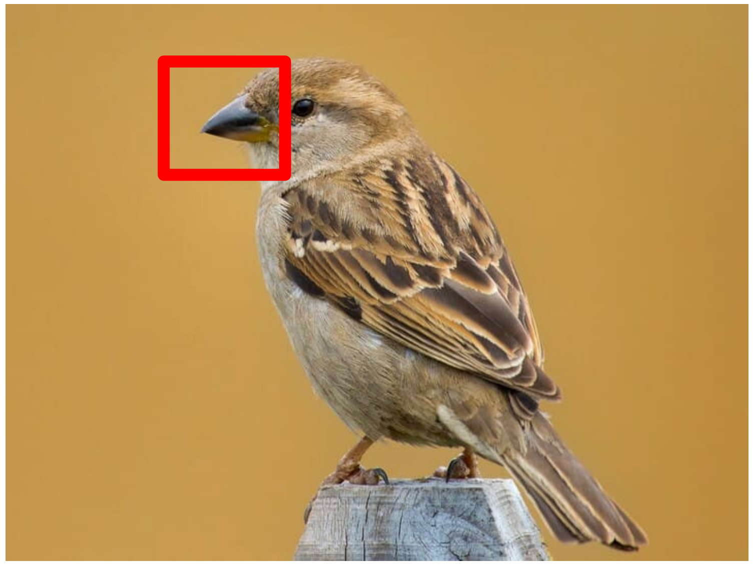
- Some patterns are much smaller than the whole image

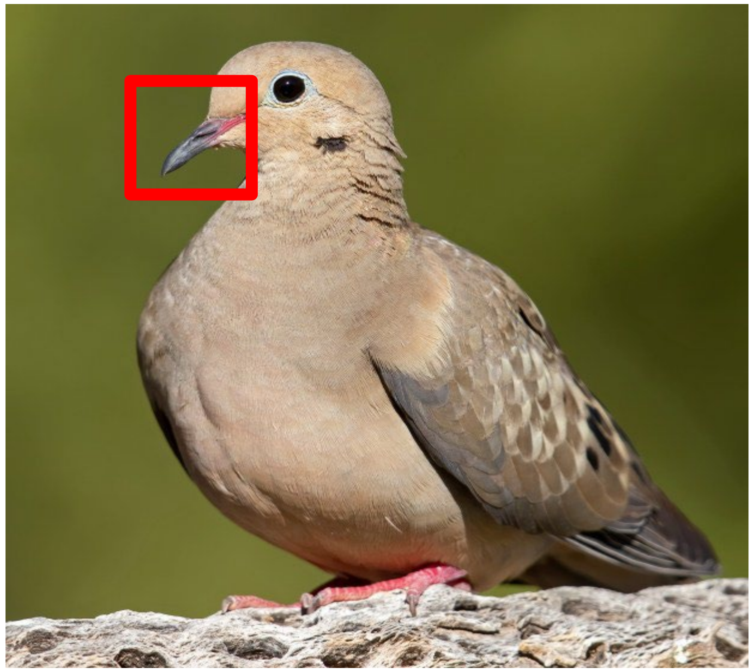
Can represent a small region with fewer parameters

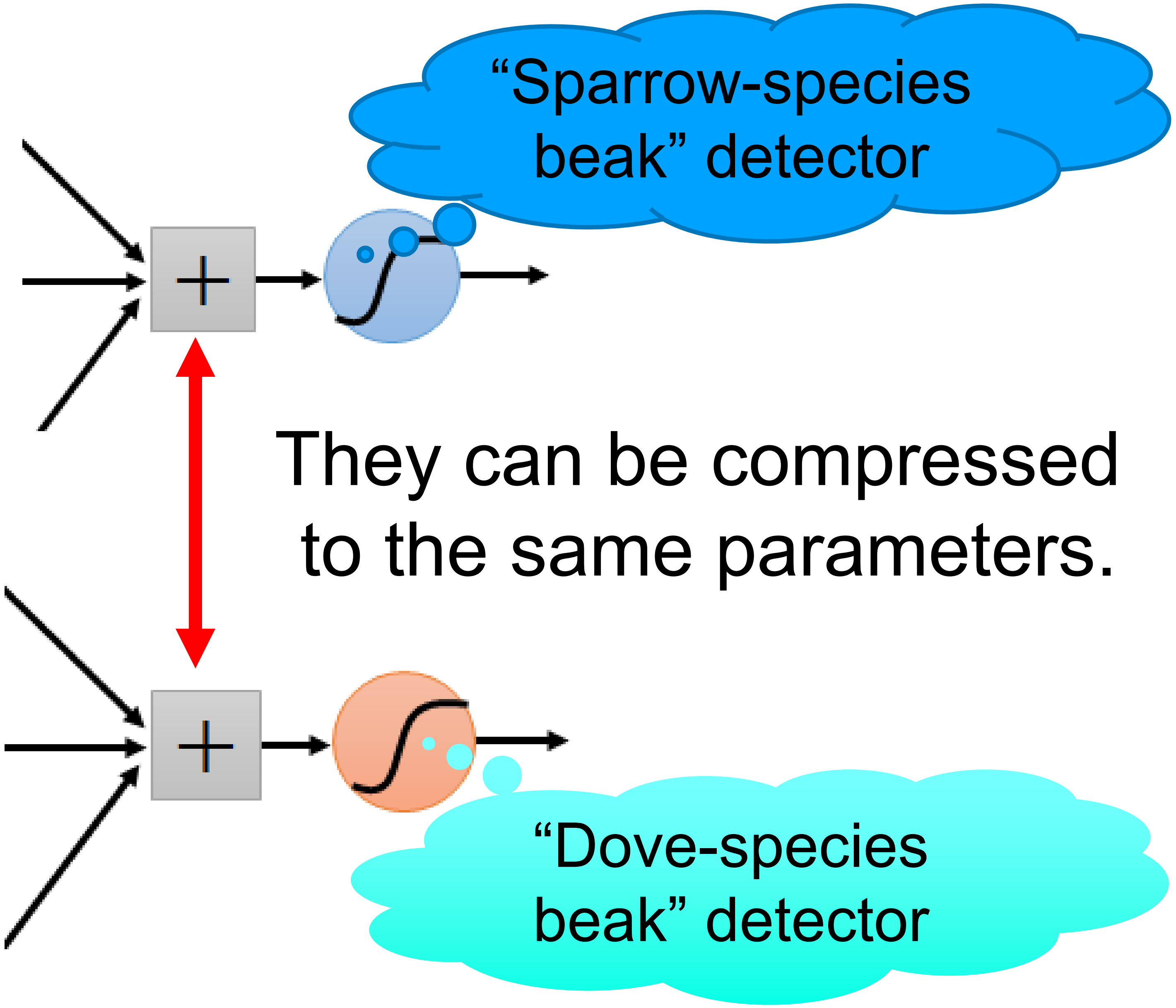


“beak” detector

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".







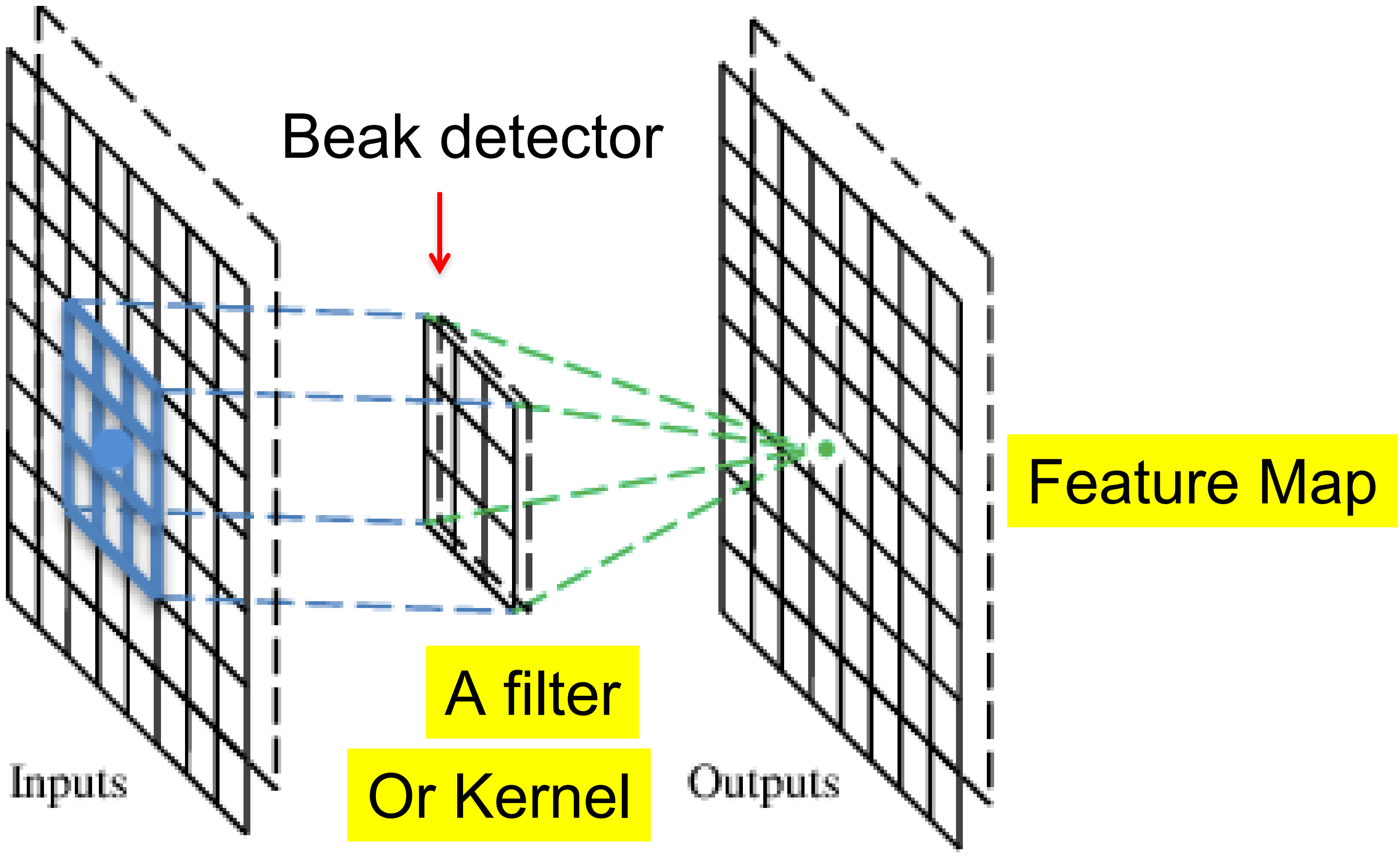
"Sparrow-species beak" detector

They can be compressed to the same parameters.

"Dove-species beak" detector

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

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

These are the network parameters to be learned.

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

Filter 1

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

Filter 2

⋮ ⋮

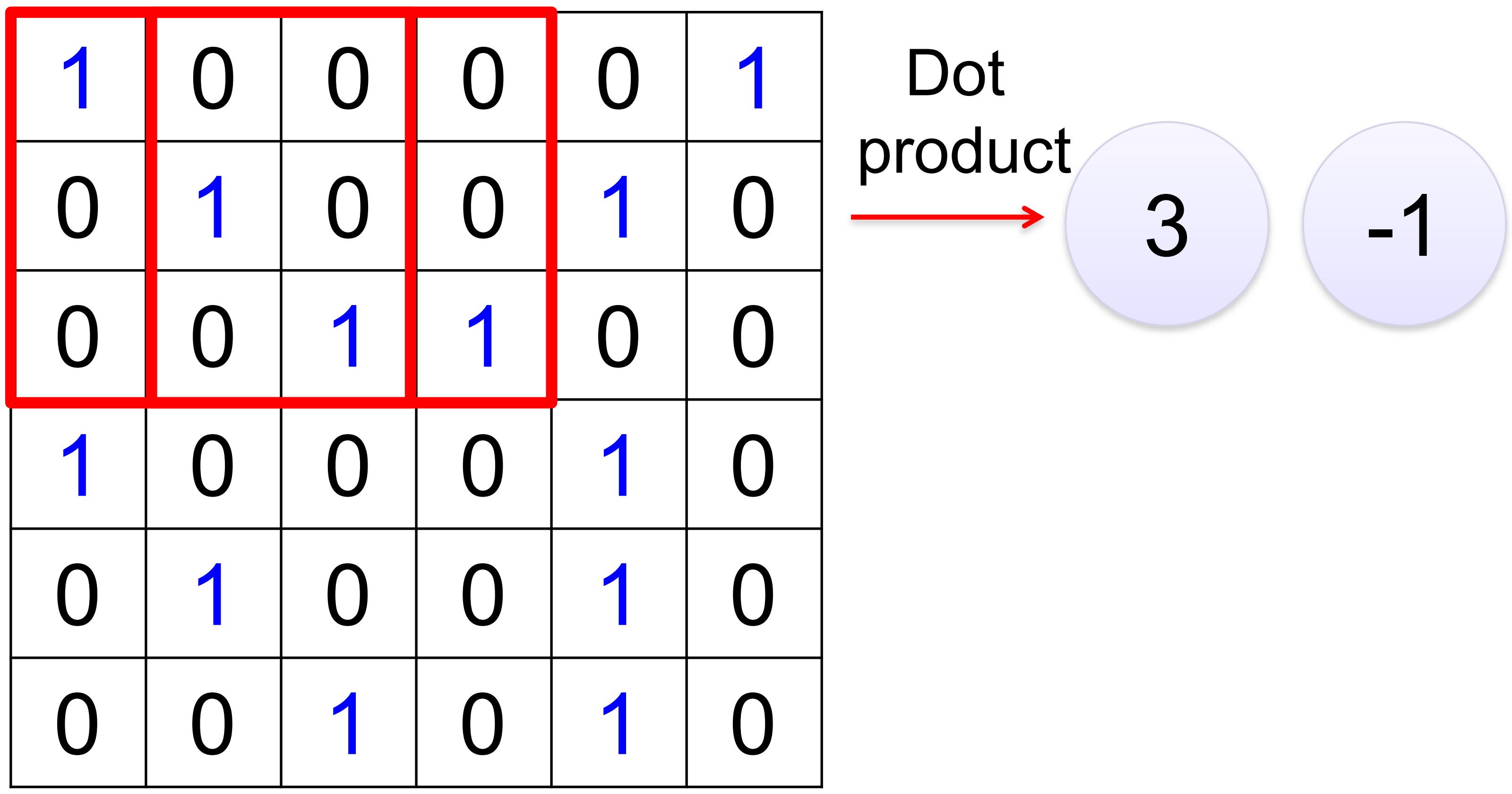
Each filter detects a small pattern (3 x 3).

Convolution

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

Filter 1

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

Convolution

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

Filter 1

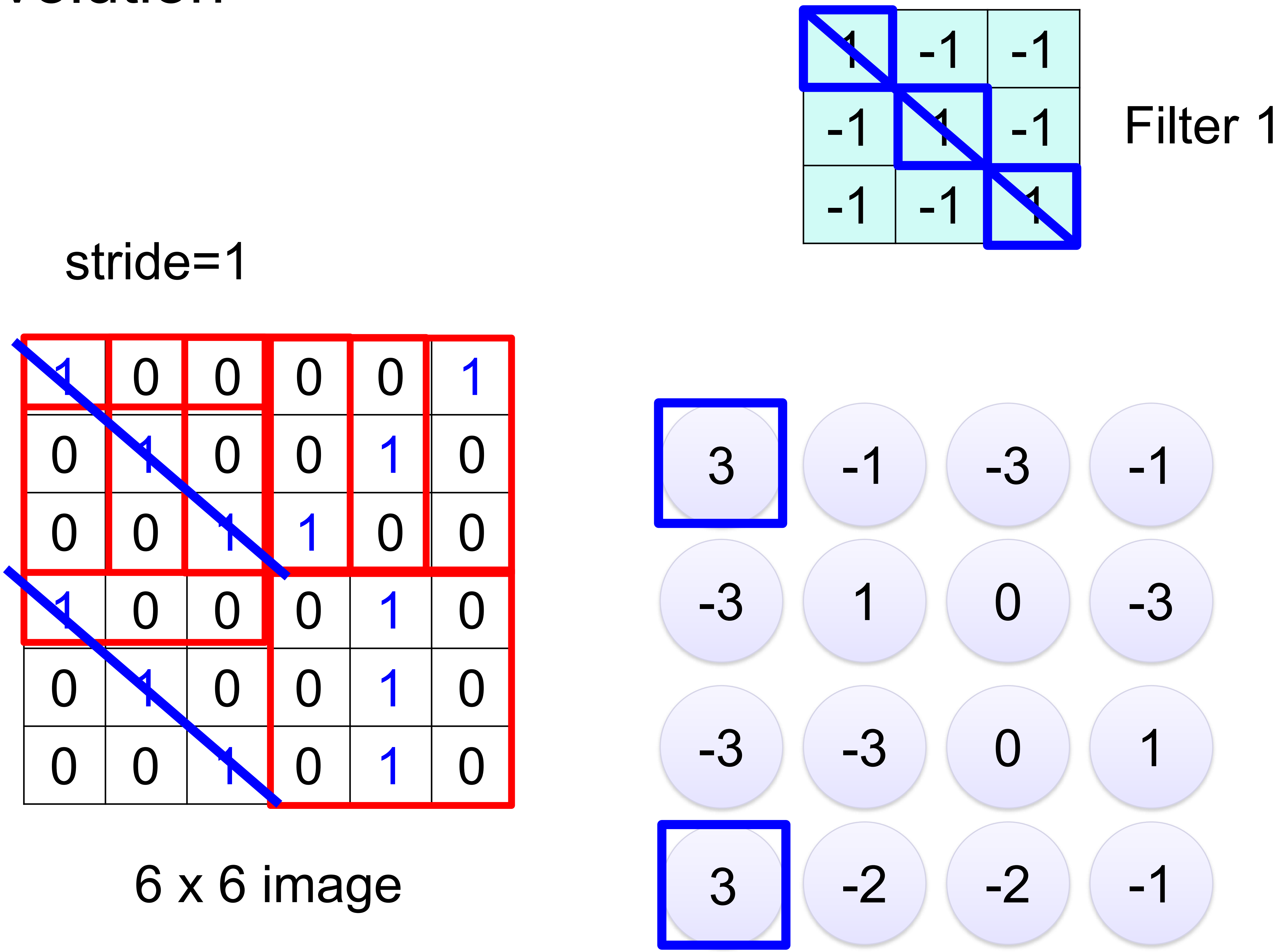
If stride=2

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

3 -3

Convolution



Filter 1

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

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

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

Convolution

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

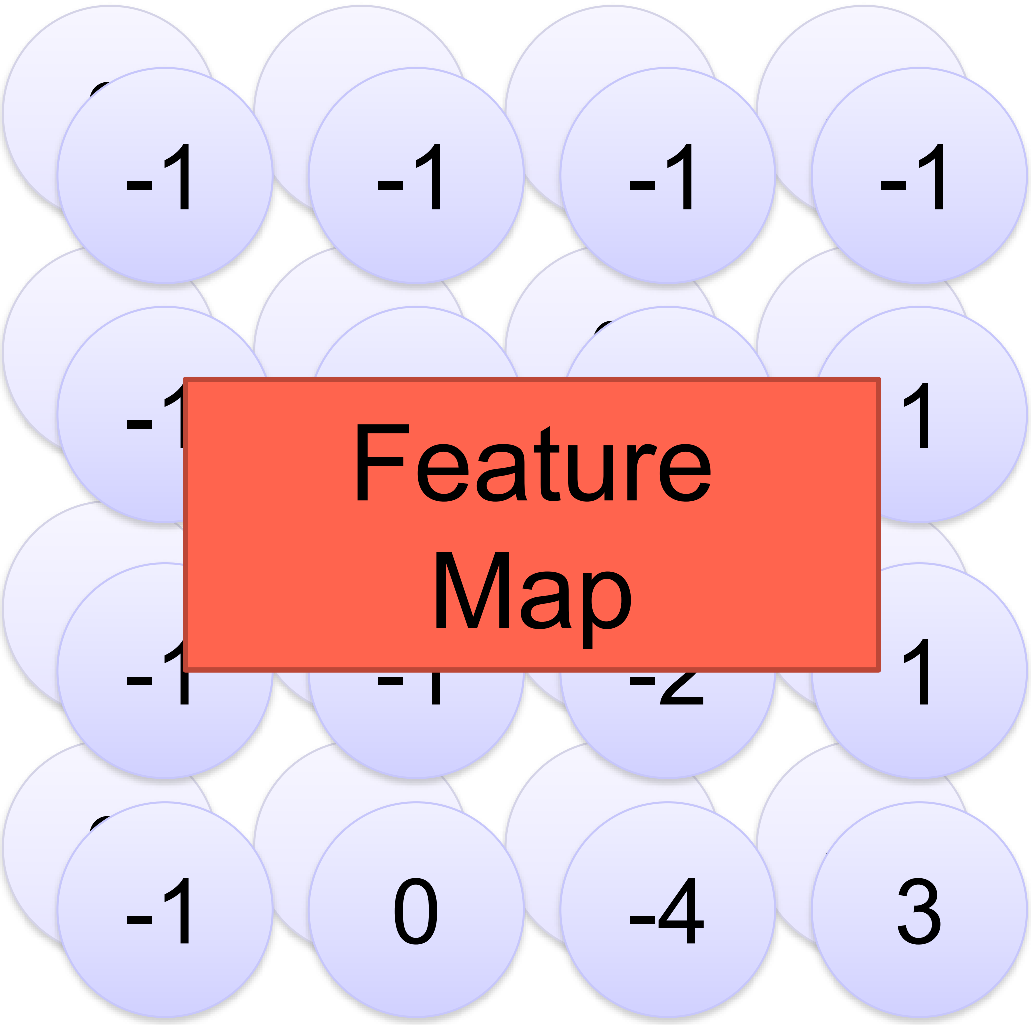
Filter 2

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

Repeat this for each filter



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

Color image: RGB 3 channels

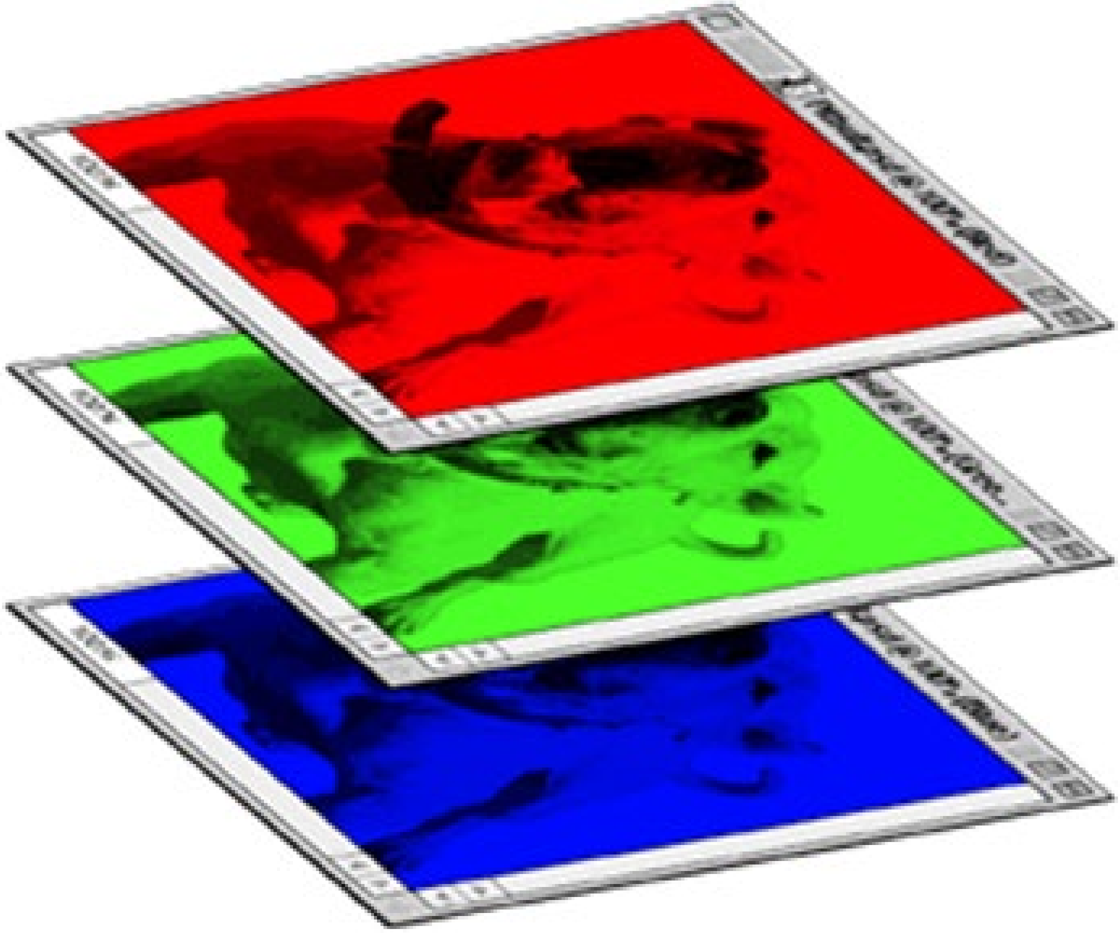
Filter 1

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

Filter 2

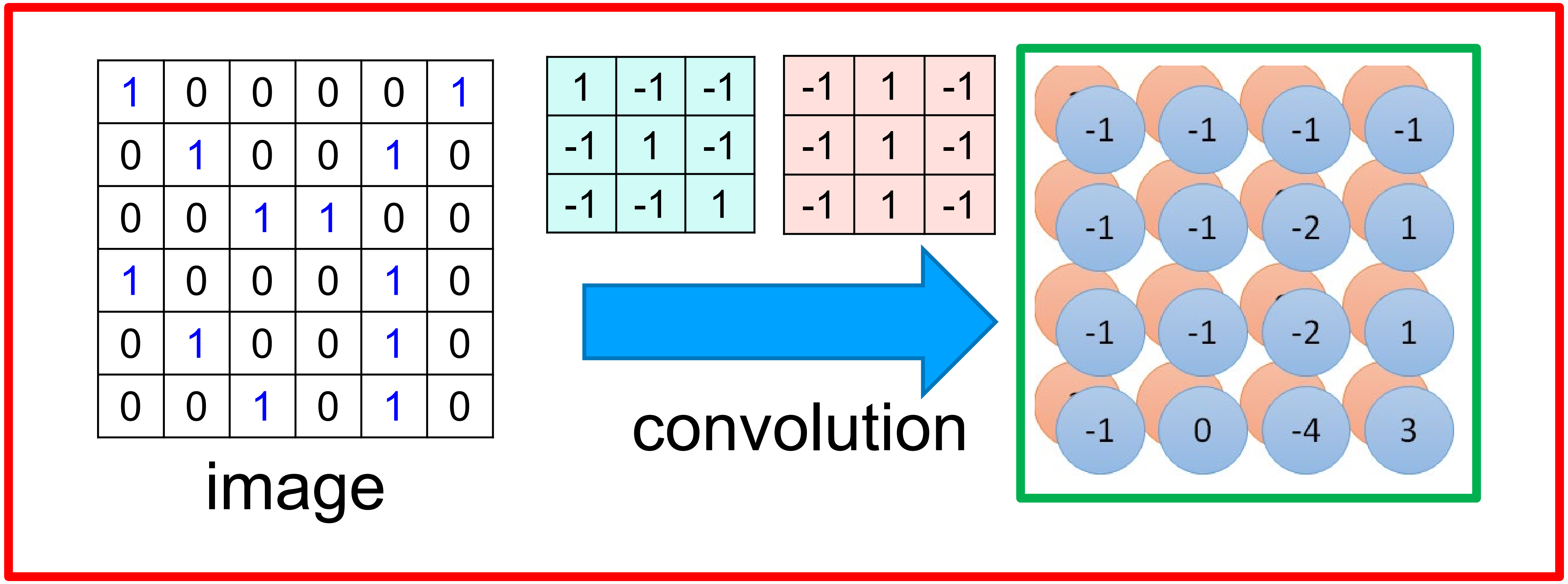
-1	1	-1
-1	1	-1
-1	1	-1

Color image



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

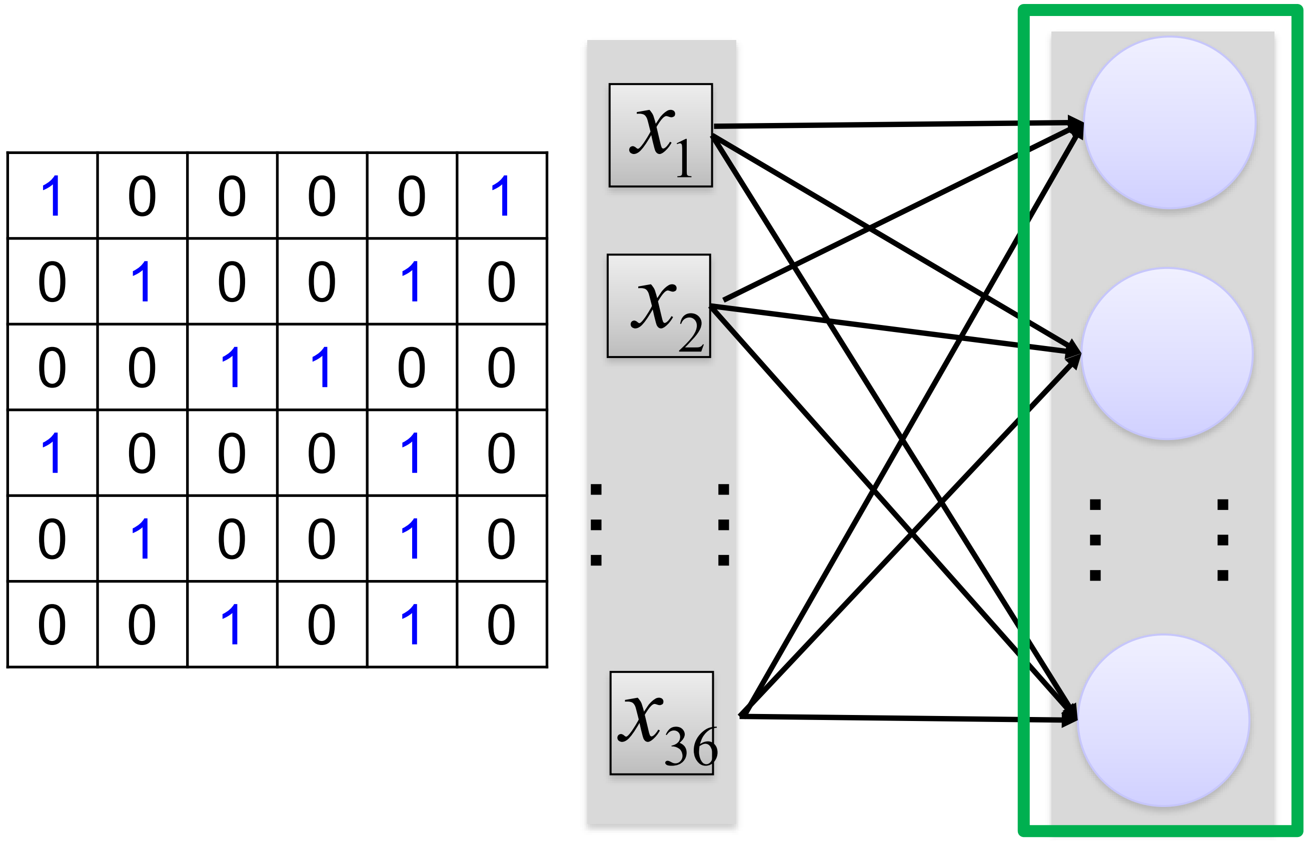
1	-1	-1
-1	1	-1
-1	-1	1

-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

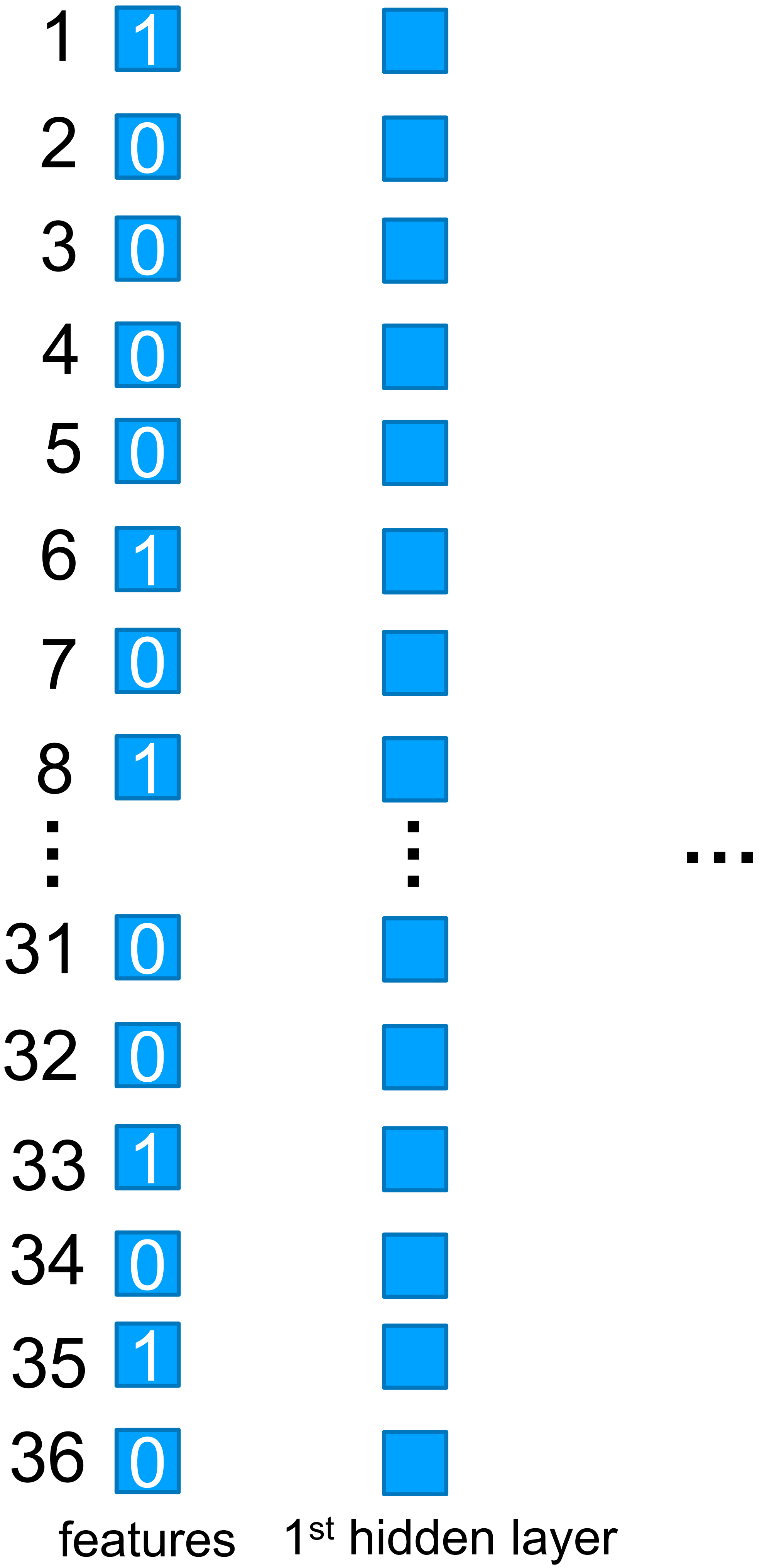


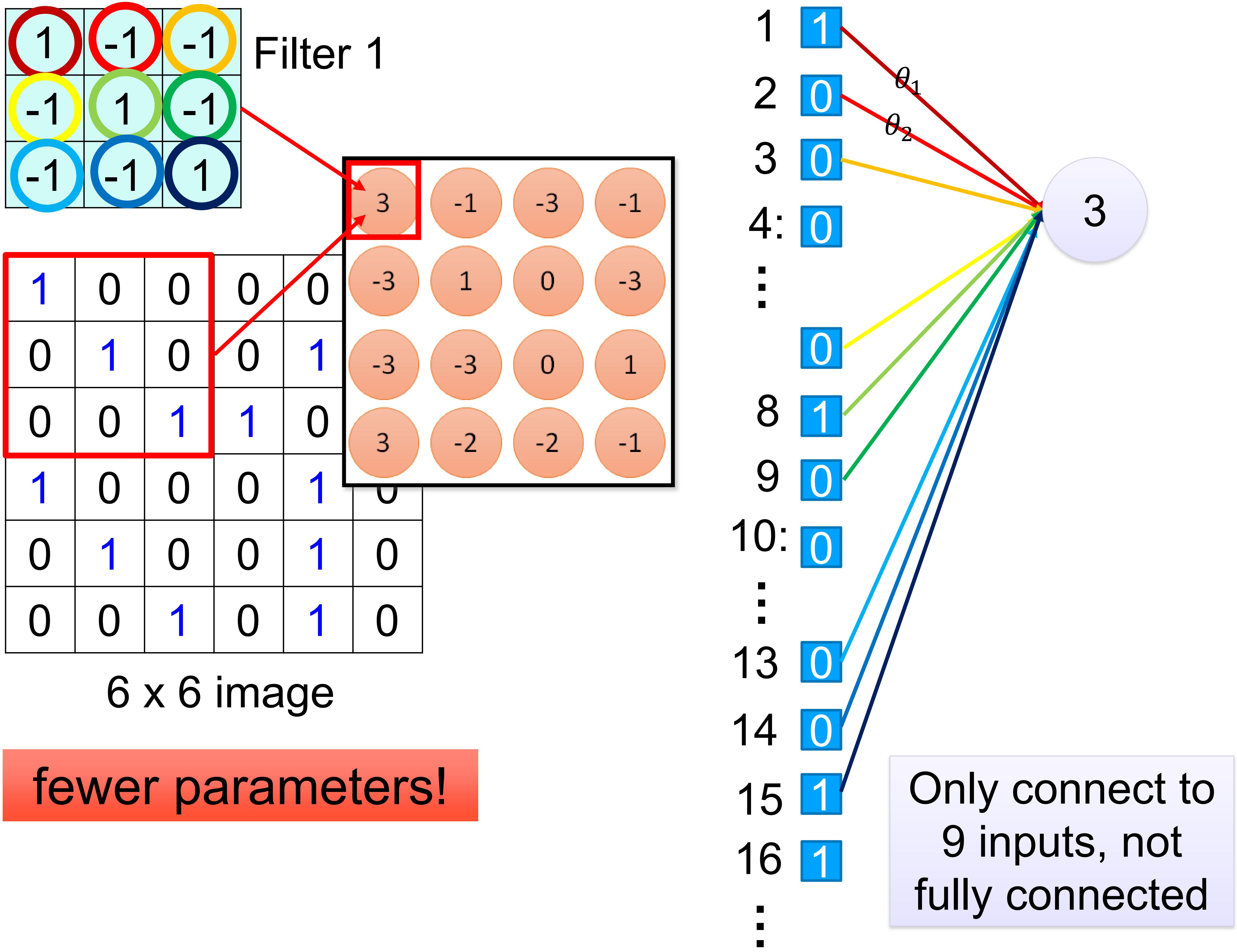
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

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





Filter 1

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

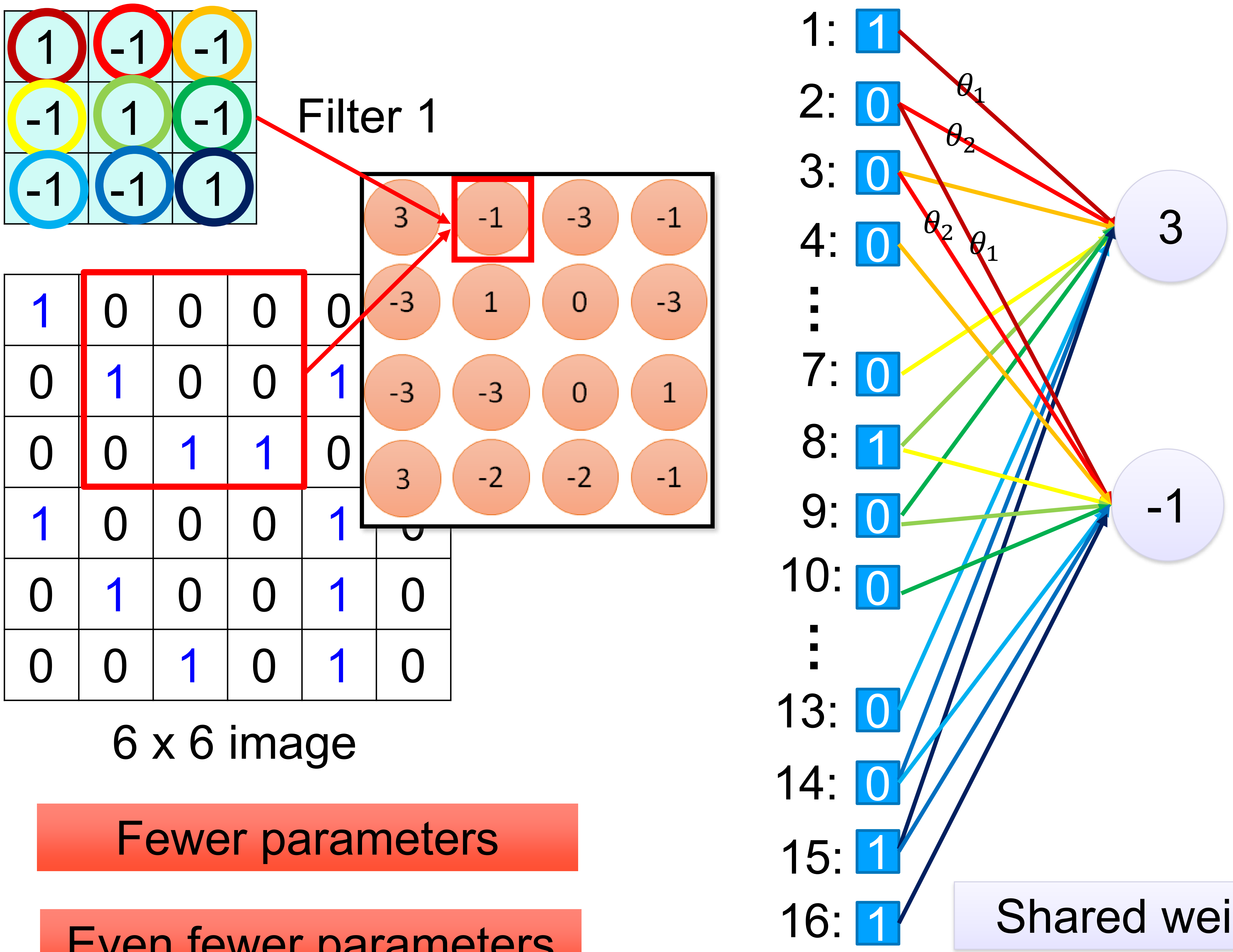
3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

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

6 x 6 image

fewer parameters!

Only connect to 9 inputs, not fully connected



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

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

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

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

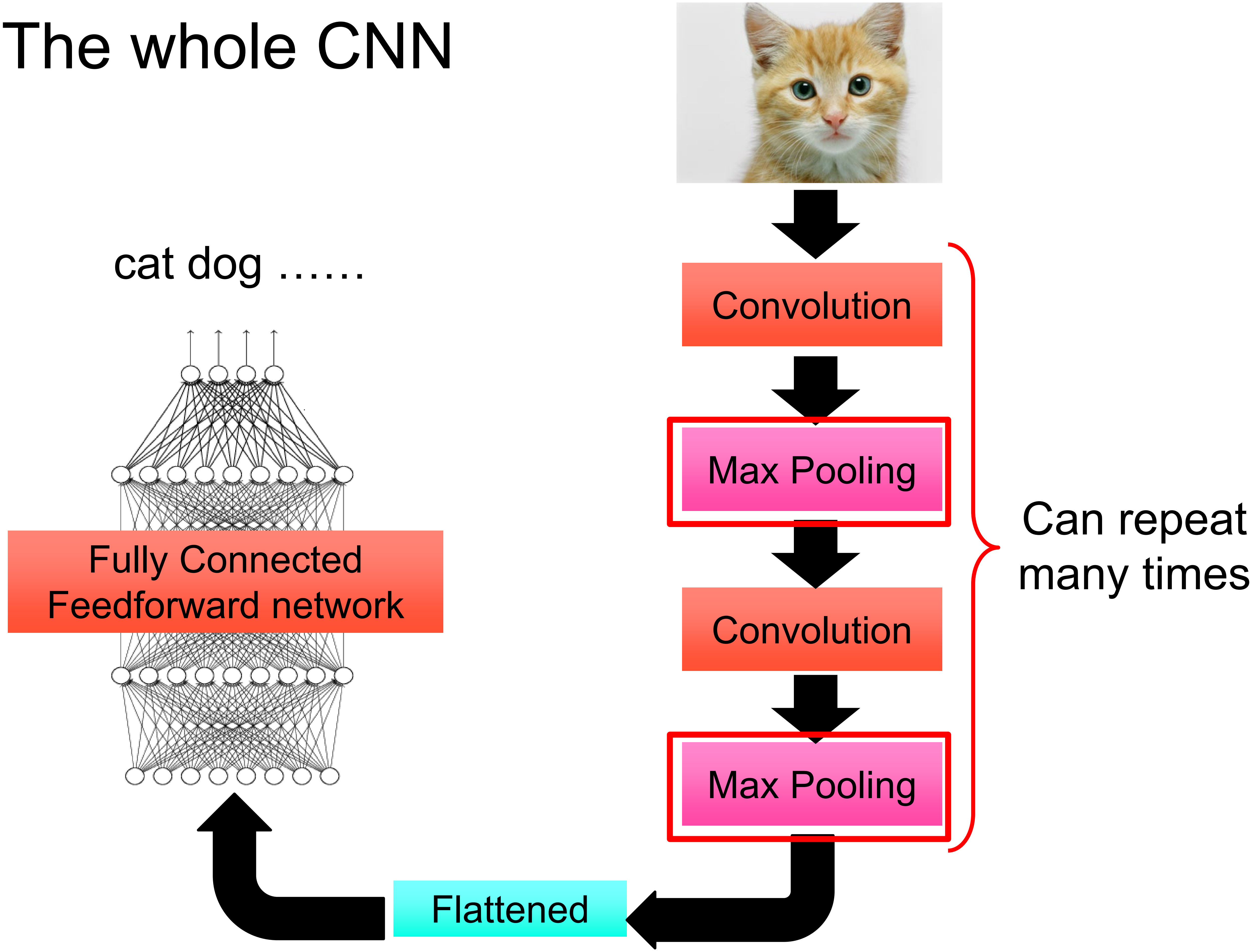
Fewer parameters

Even fewer parameters

Shared weights

Ex. ○ constrained to be identical

The whole CNN



cat dog ……

Fully Connected Feedforward network

Convolution

Max Pooling

Convolution

Max Pooling

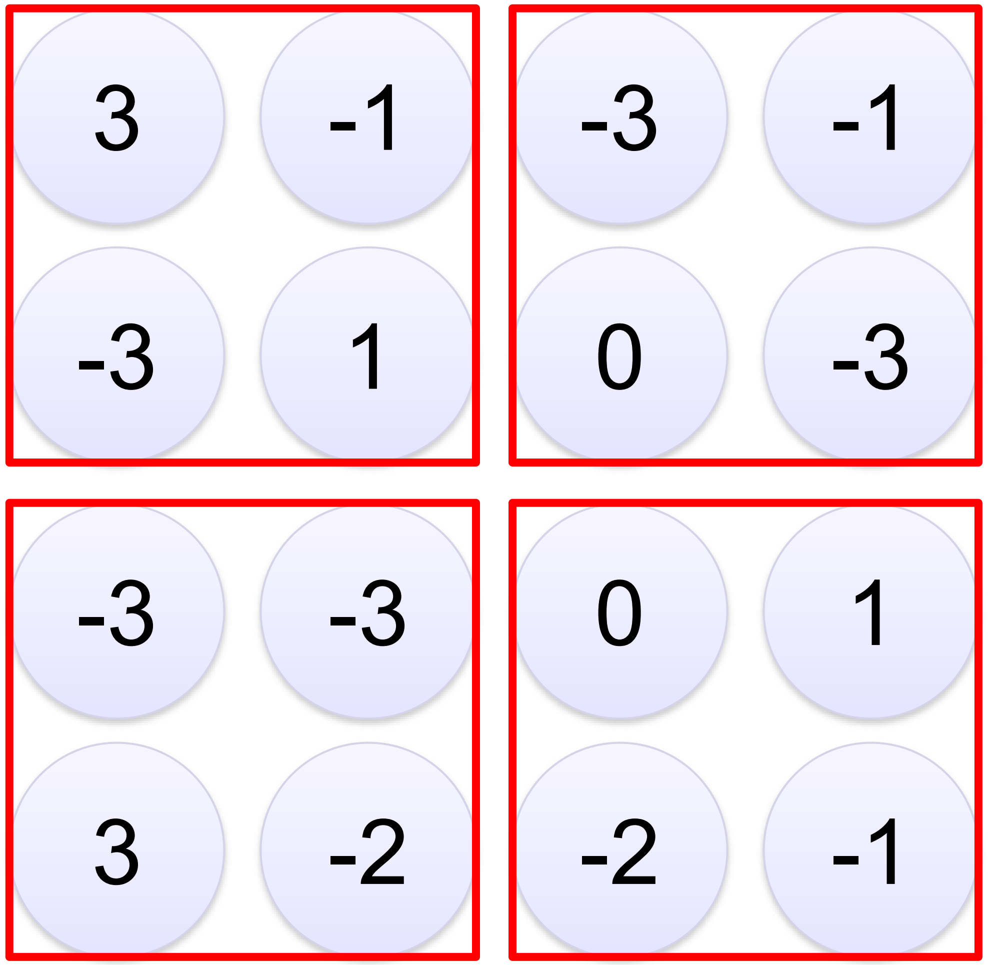
Can repeat many times

Flattened

Max Pooling

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

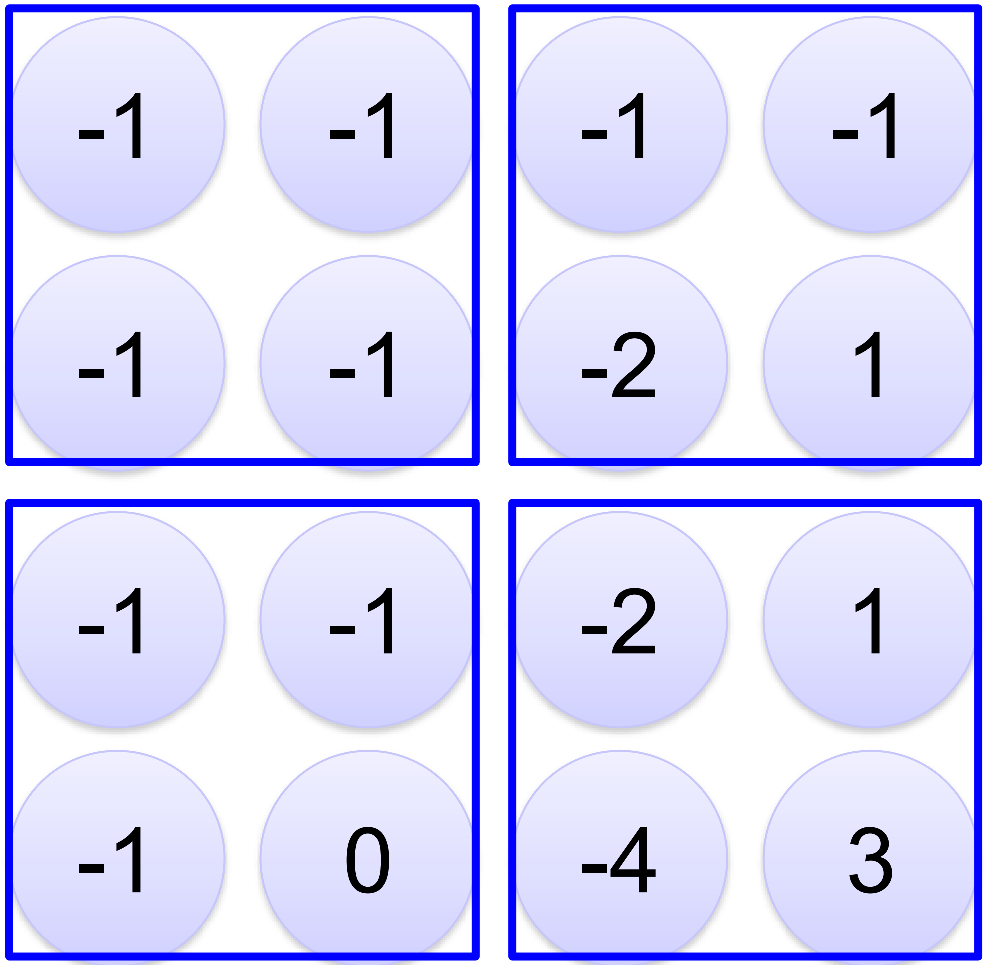
Filter 1



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

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

Filter 2



-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:

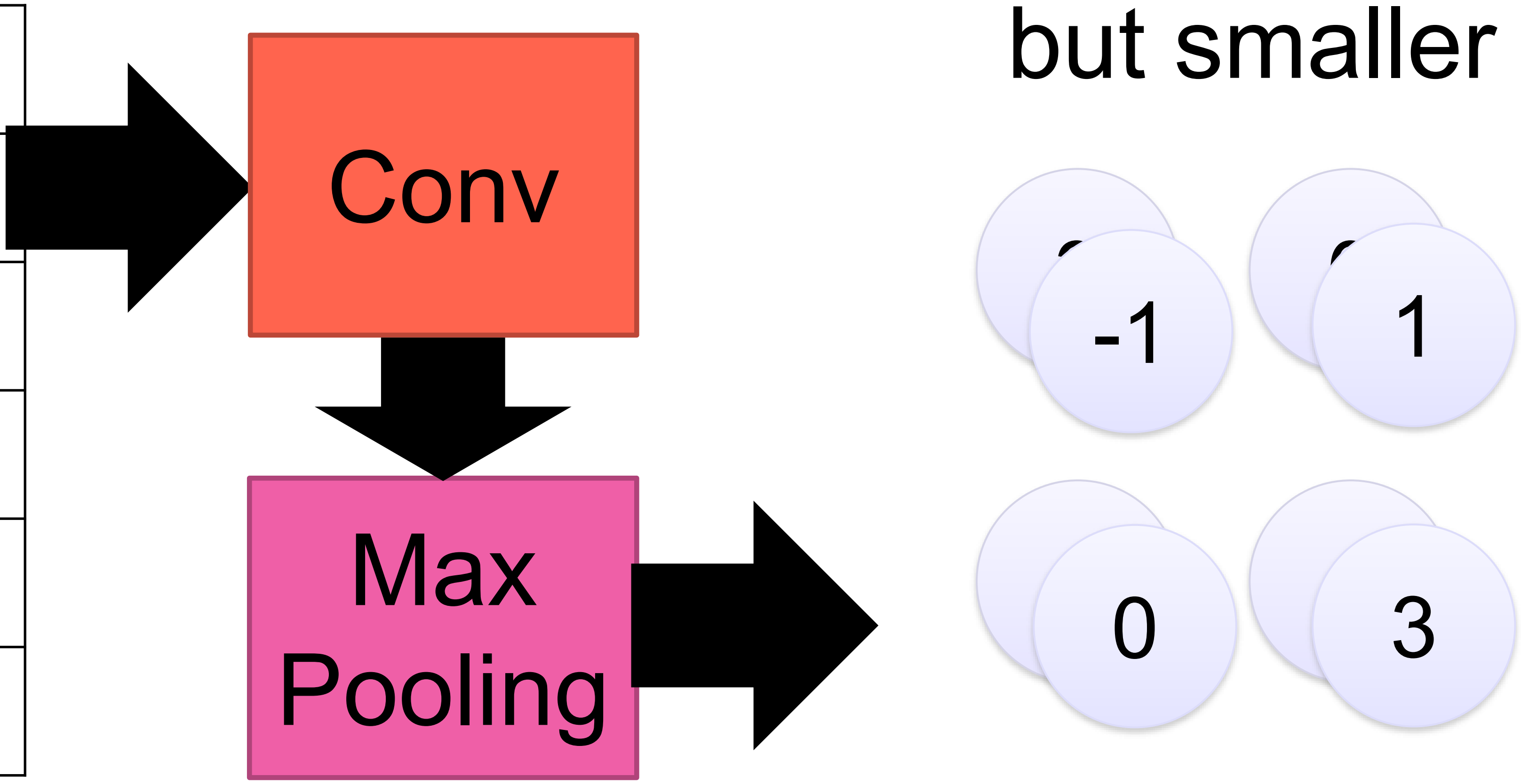
- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

Max Pooling

New image
but smaller

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

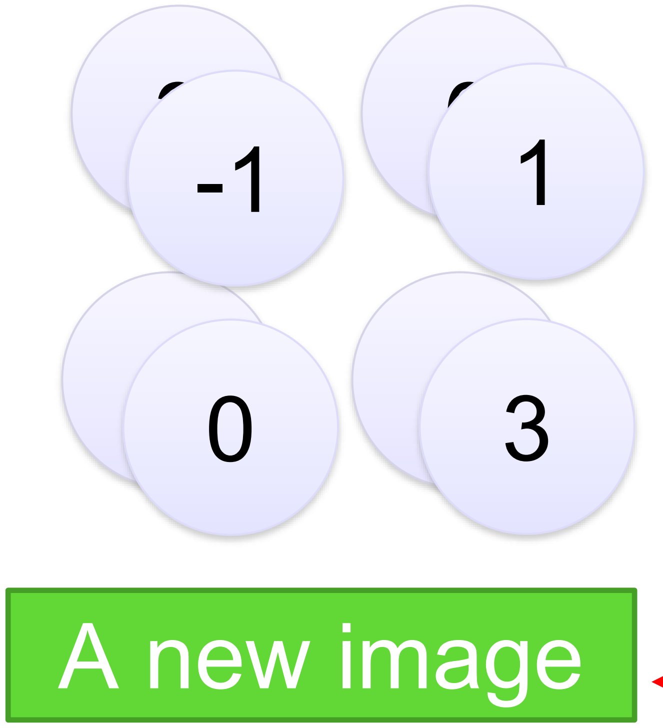
6 x 6 image



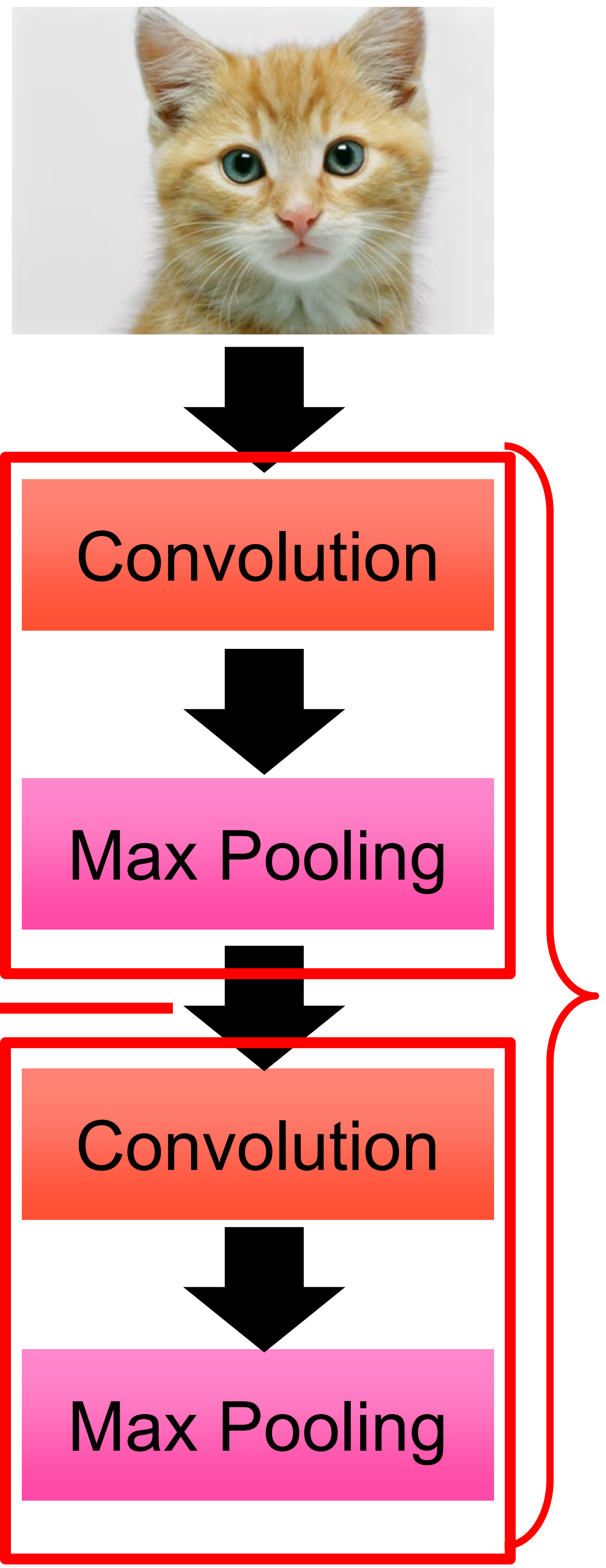
2 x 2 image

Each filter
is a channel

The whole CNN



A new image


Convolution

Max Pooling

Convolution

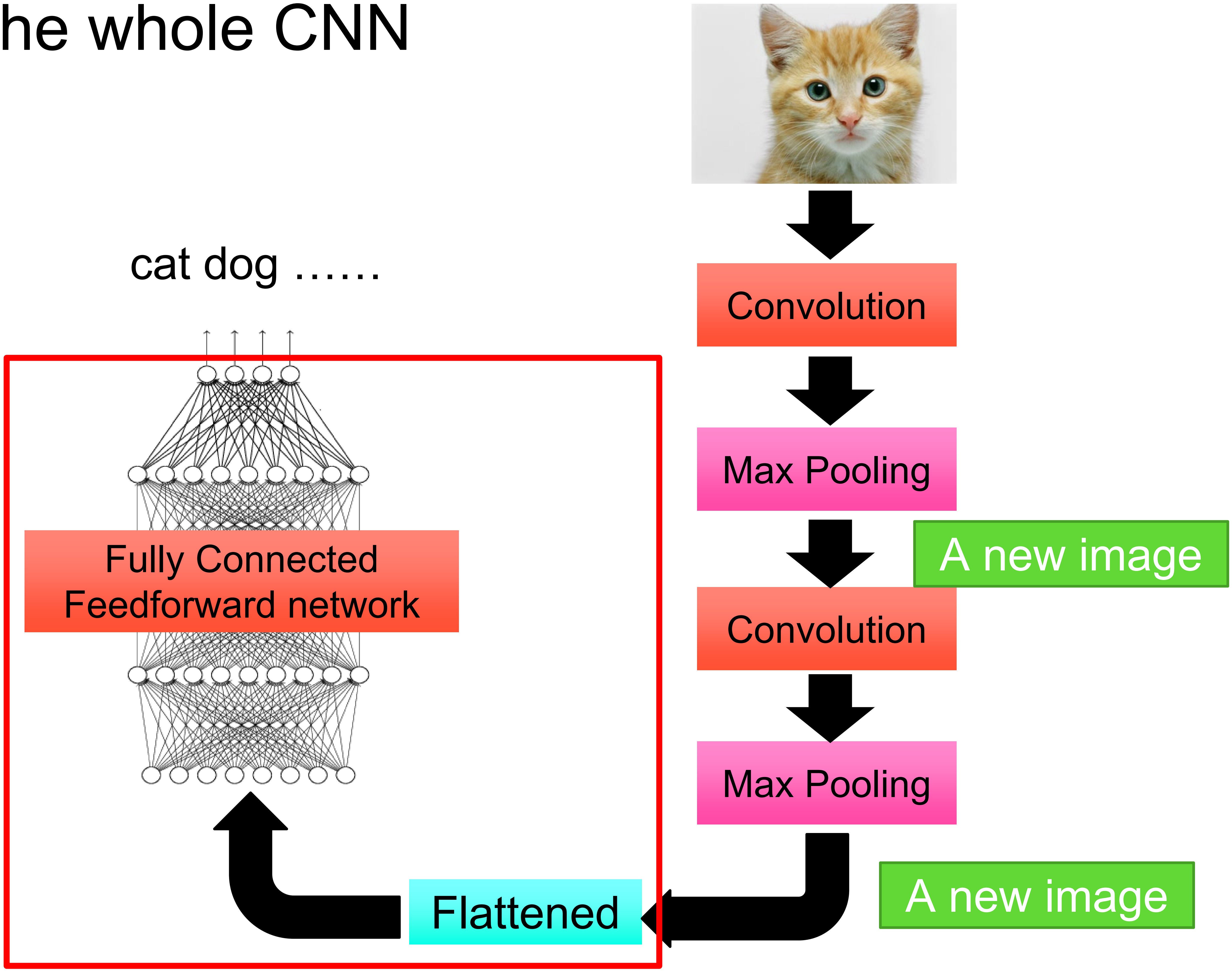
Max Pooling

Can repeat many times

Smaller than the original image

The number of channels is the number of filters

The whole CNN



Convolution

Max Pooling

A new image

Convolution

Max Pooling

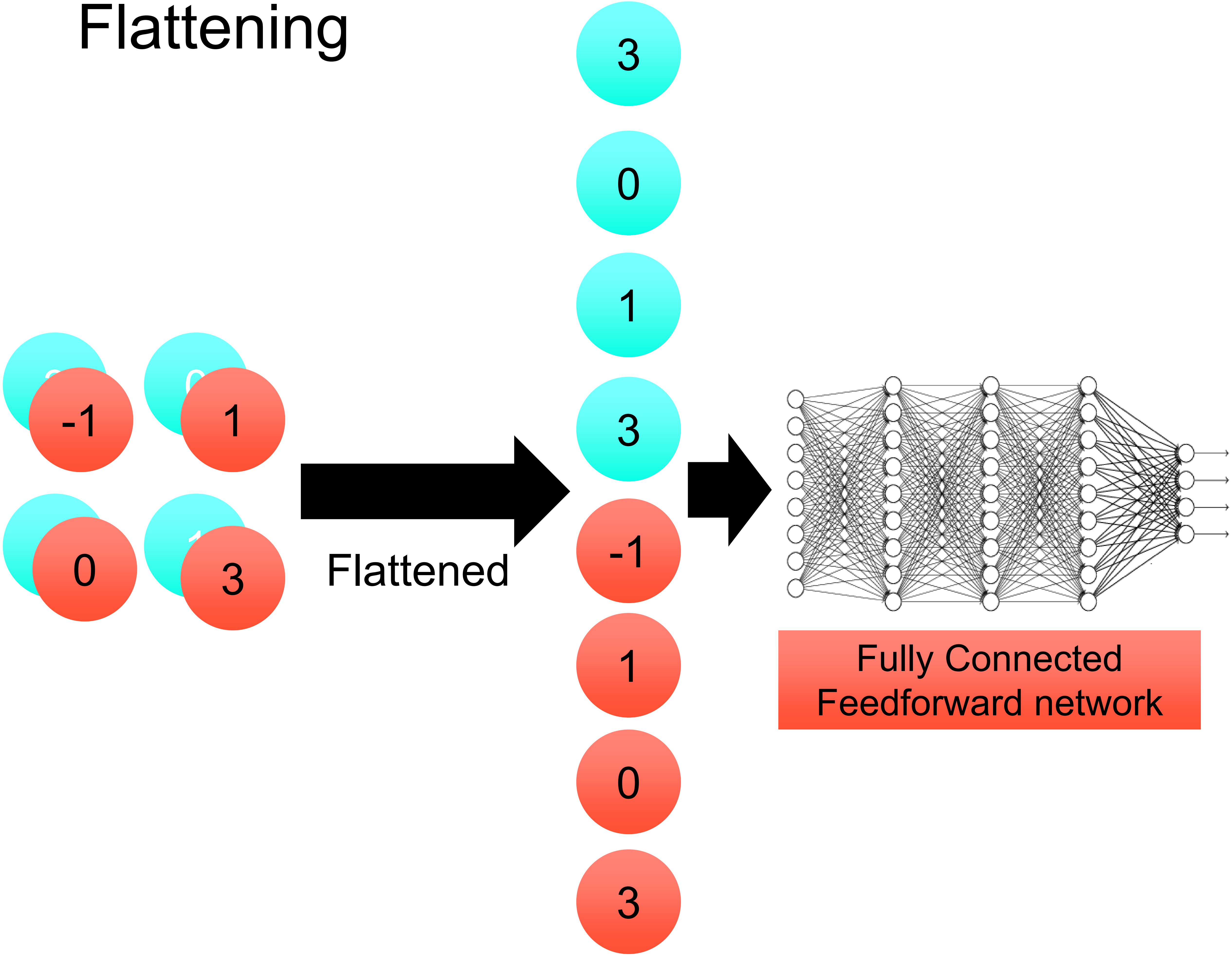
A new image

Flattened

Fully Connected Feedforward network

cat dog ……

Flattening



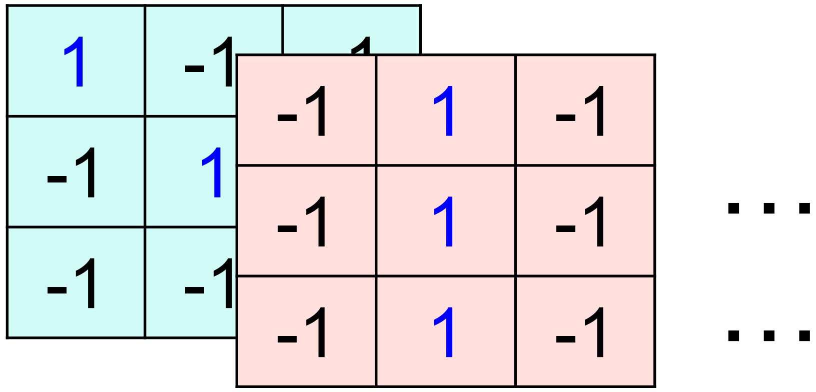
Flattened

Fully Connected Feedforward network

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)) )
```



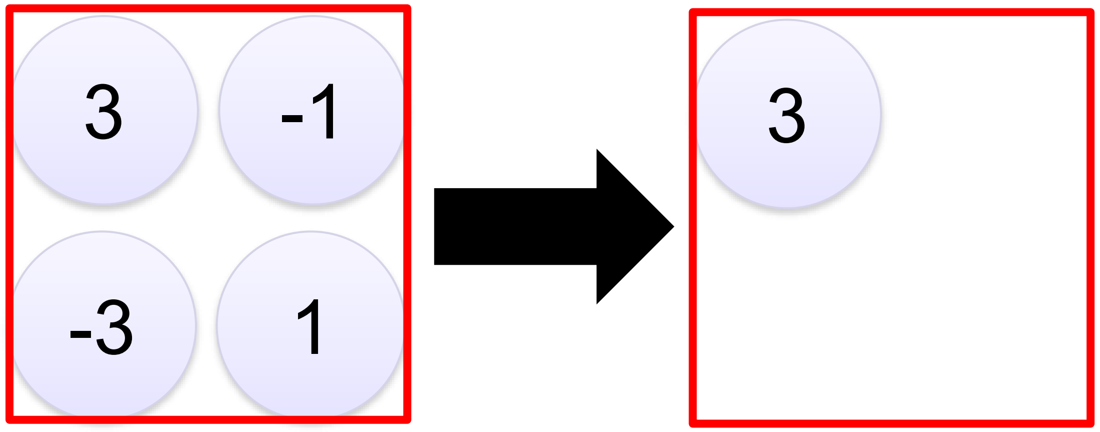
There are 25 3x3 filters.

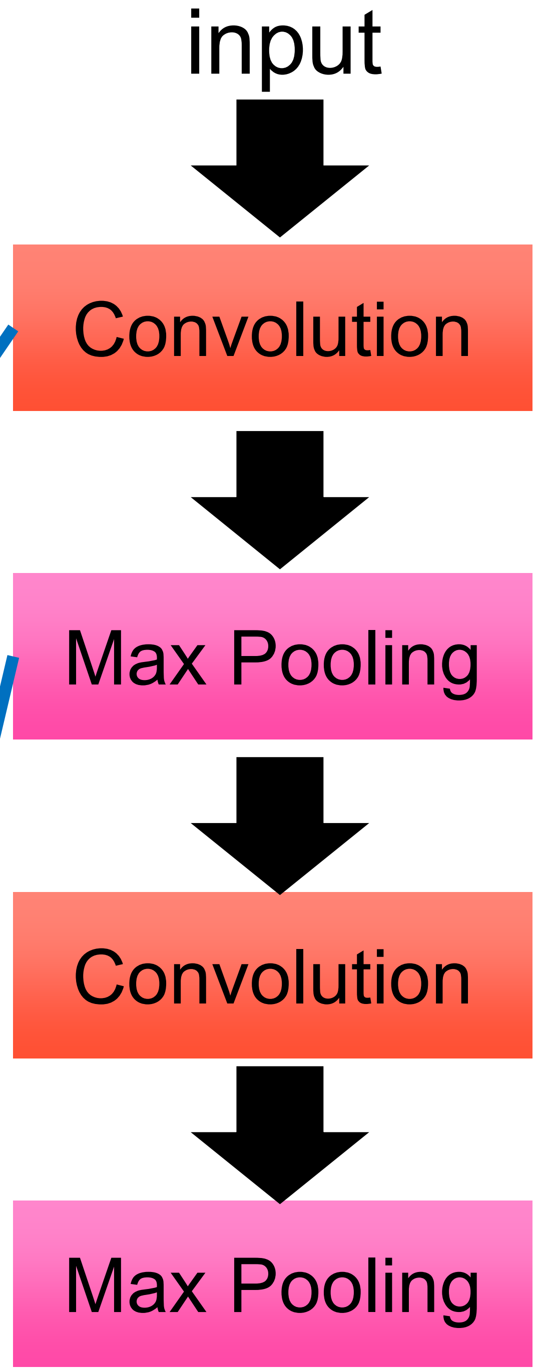
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

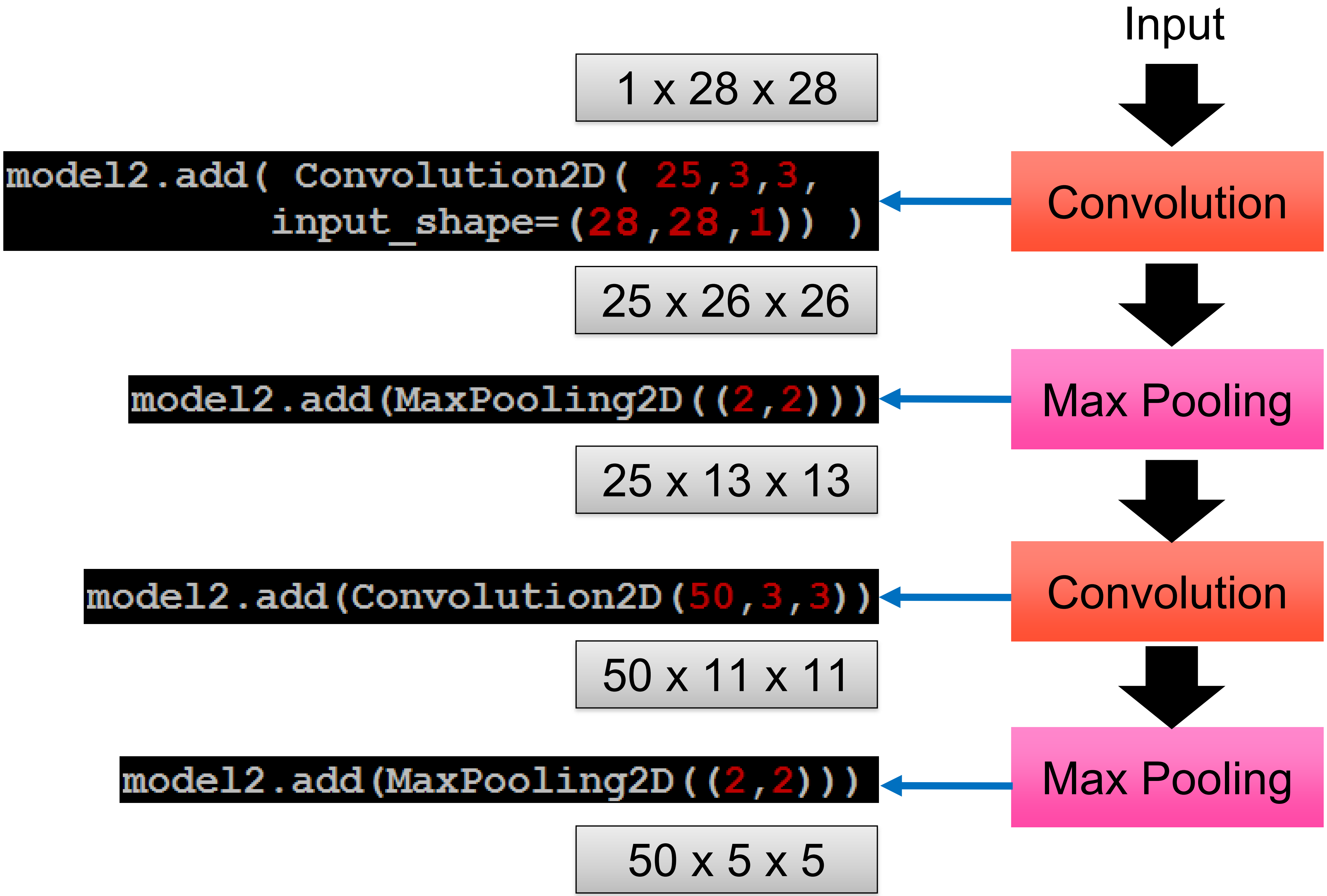
```
model2.add(MaxPooling2D((2,2)))
```





CNN in Keras

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



Input

1 x 28 x 28

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

Convolution

25 x 26 x 26

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

Max Pooling

25 x 13 x 13

```
model2.add(Convolution2D(50,3,3))
```

Convolution

50 x 11 x 11

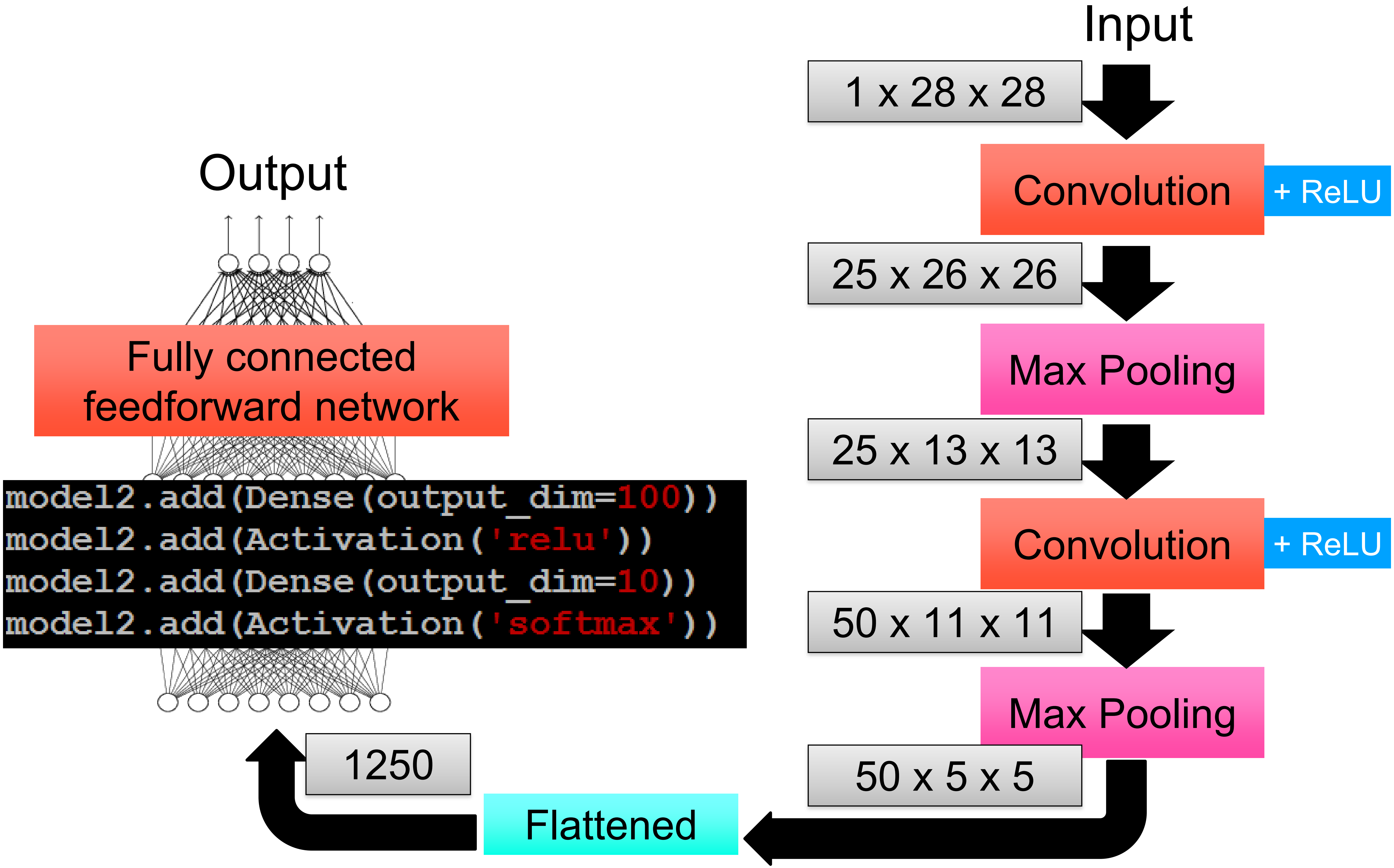
```
model2.add(MaxPooling2D((2,2)))
```

Max Pooling

50 x 5 x 5

CNN in Keras

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



Input

1 x 28 x 28

Convolution + ReLU

25 x 26 x 26

Max Pooling

25 x 13 x 13

Convolution + ReLU

50 x 11 x 11

Max Pooling

50 x 5 x 5

Flattened

```
model2.add(Flatten())
```

1250

Fully connected feedforward network

```
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
```

Output

Number of Parameters

25X3X3+25 parameters

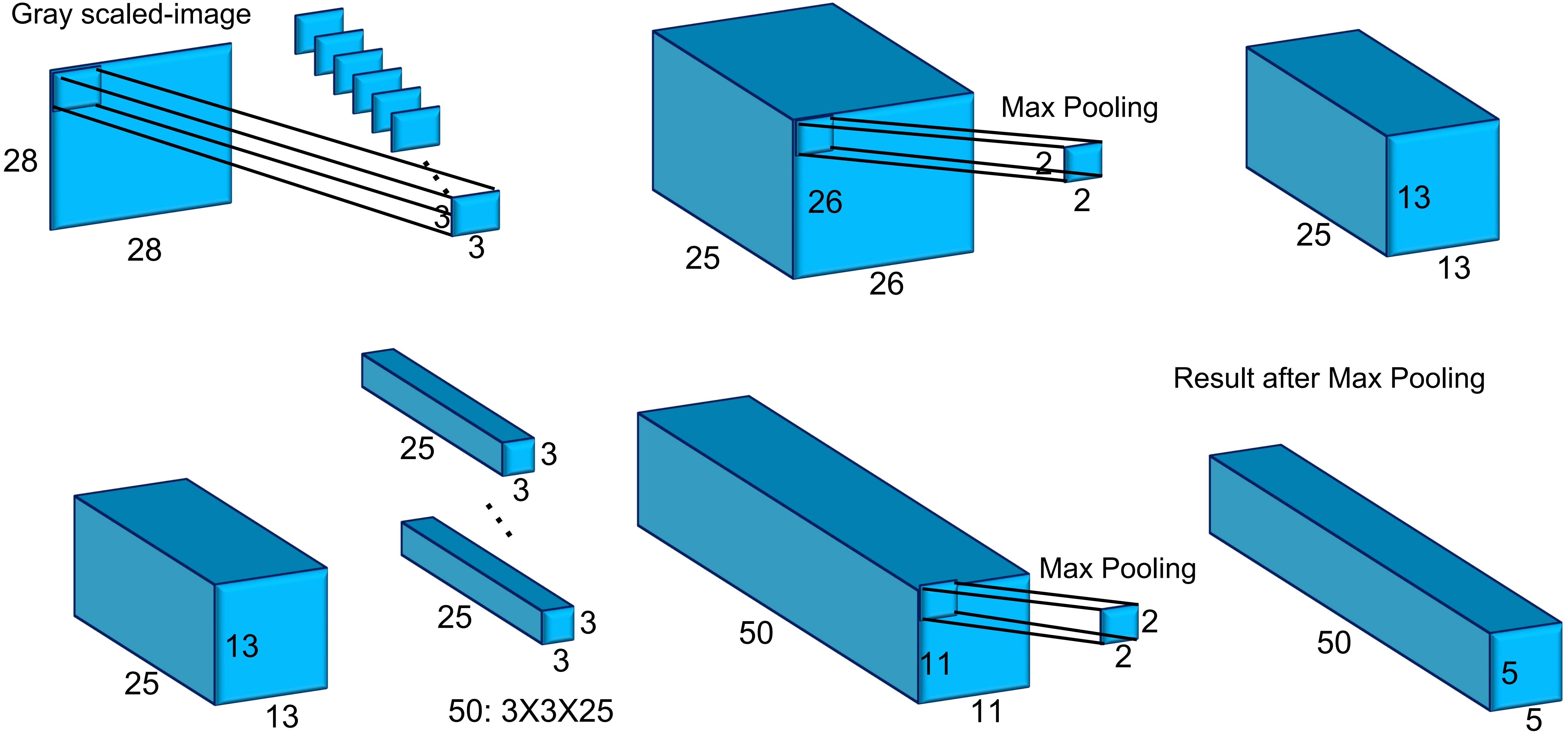
25 filters - Conv1

25: 3X3

Gray scaled-image

Convoluted result for 25 filters

Result after Max Pooling

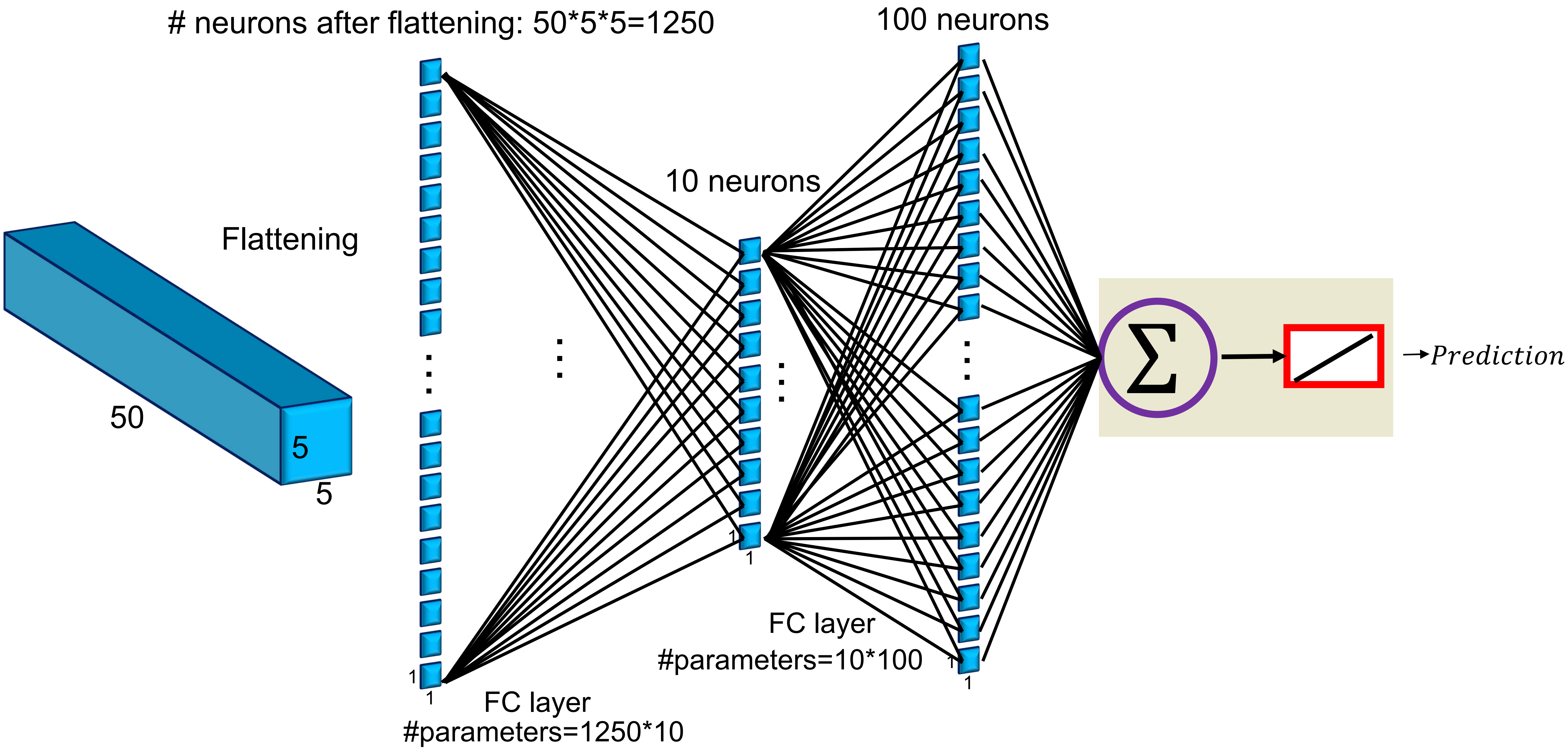


50: 3X3X25

50 filters – Conv2

Convoluted result for 50 filters

50X3X3X25+50 parameters



# neurons after flattening: 50*5*5=1250

100 neurons

10 neurons

Flattening

FC layer
#parameters=1250*10

FC layer
#parameters=10*100

10 CNN Architecture