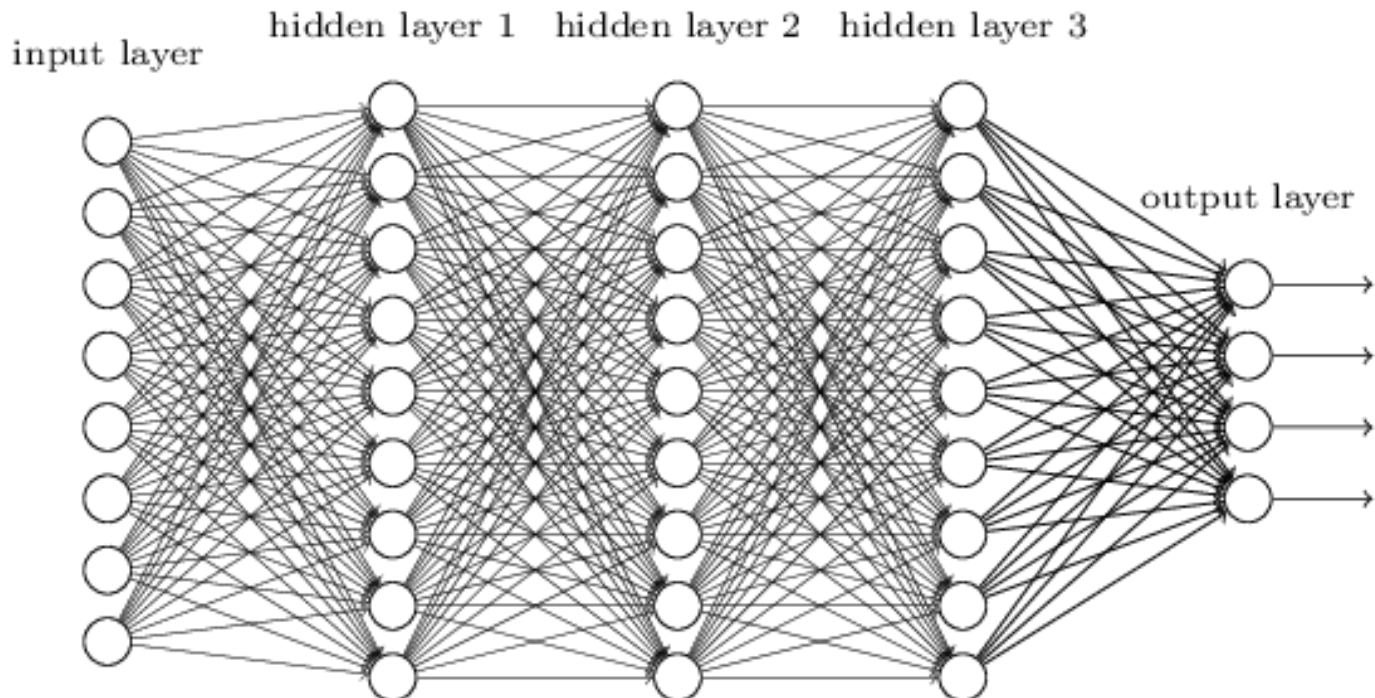


# Smaller Network: CNN

- 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:

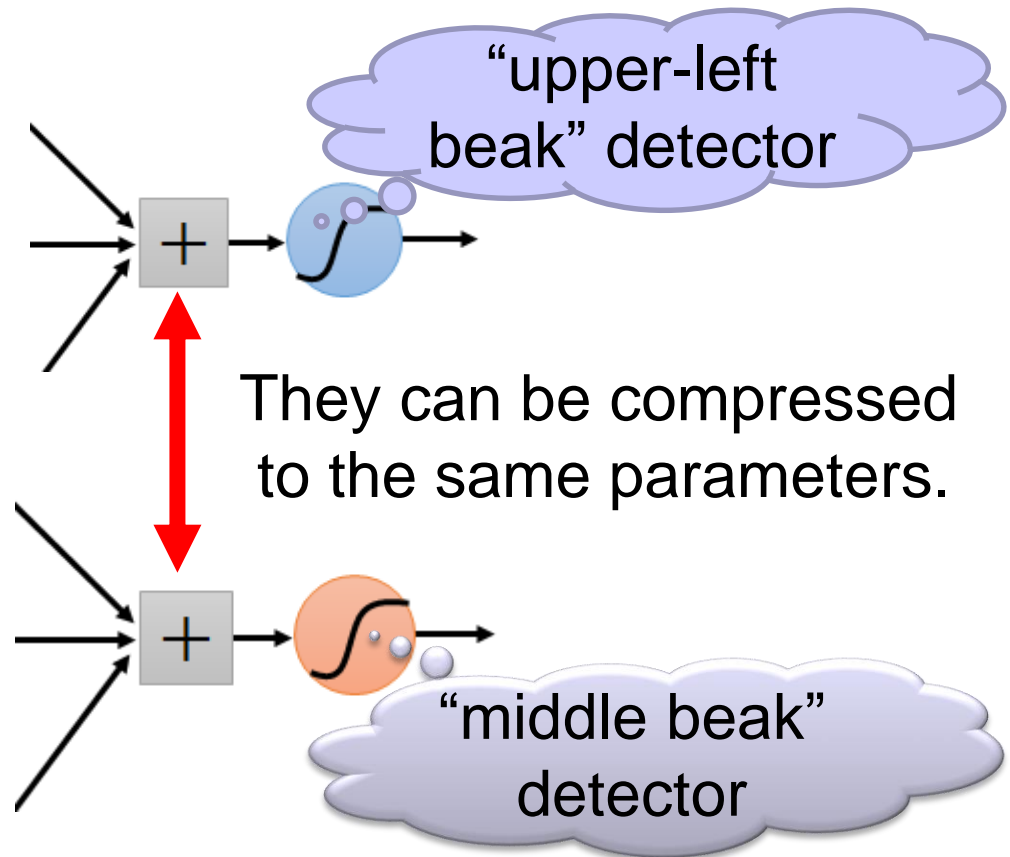
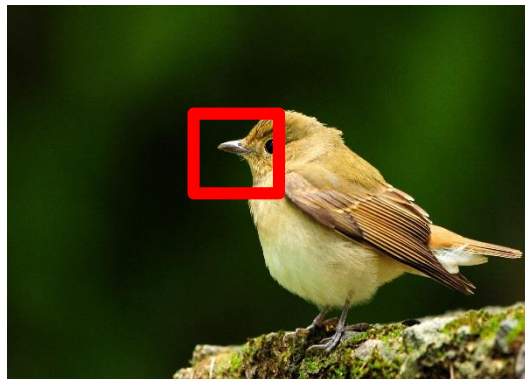
- Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters



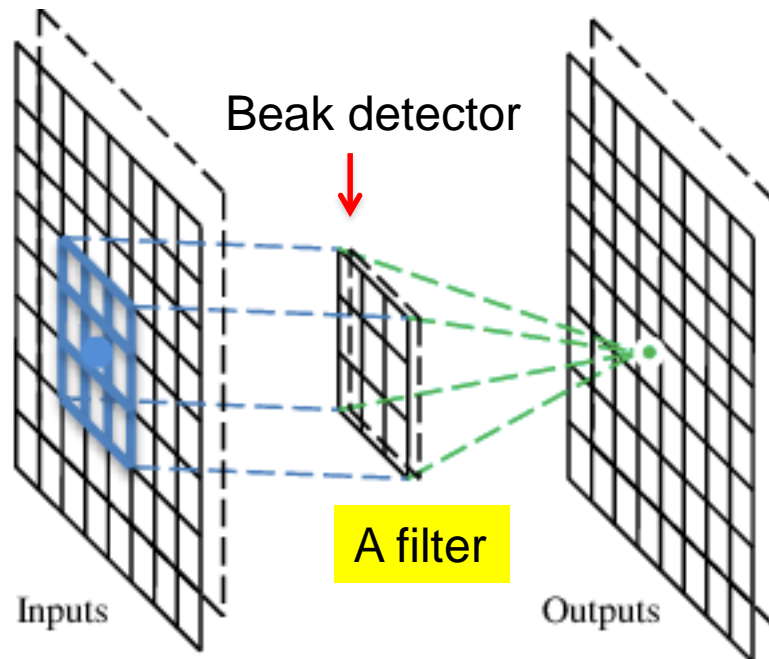
Same pattern appears in different places:  
They can be compressed!

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

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

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

Dot  
product



3

-1

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

3

-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

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



# Convolution

stride=1

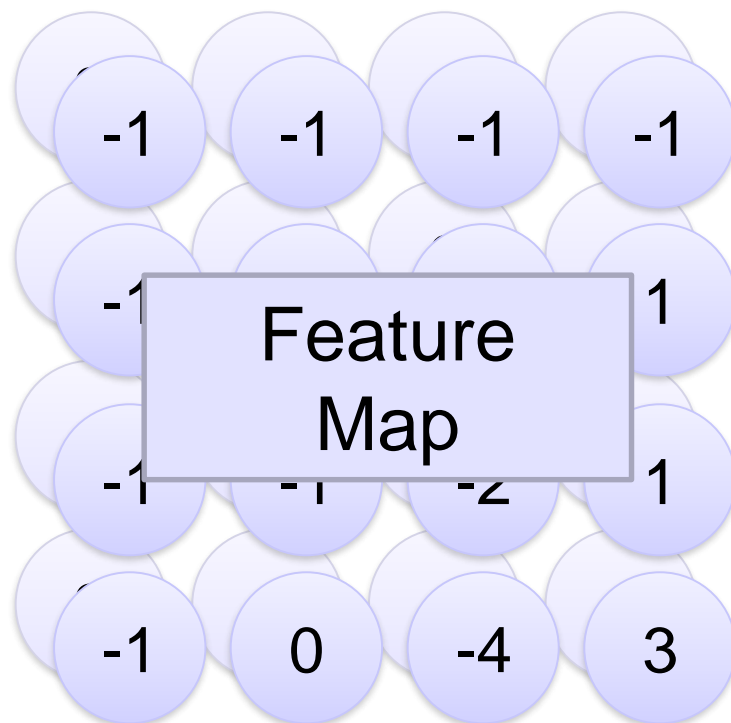
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

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

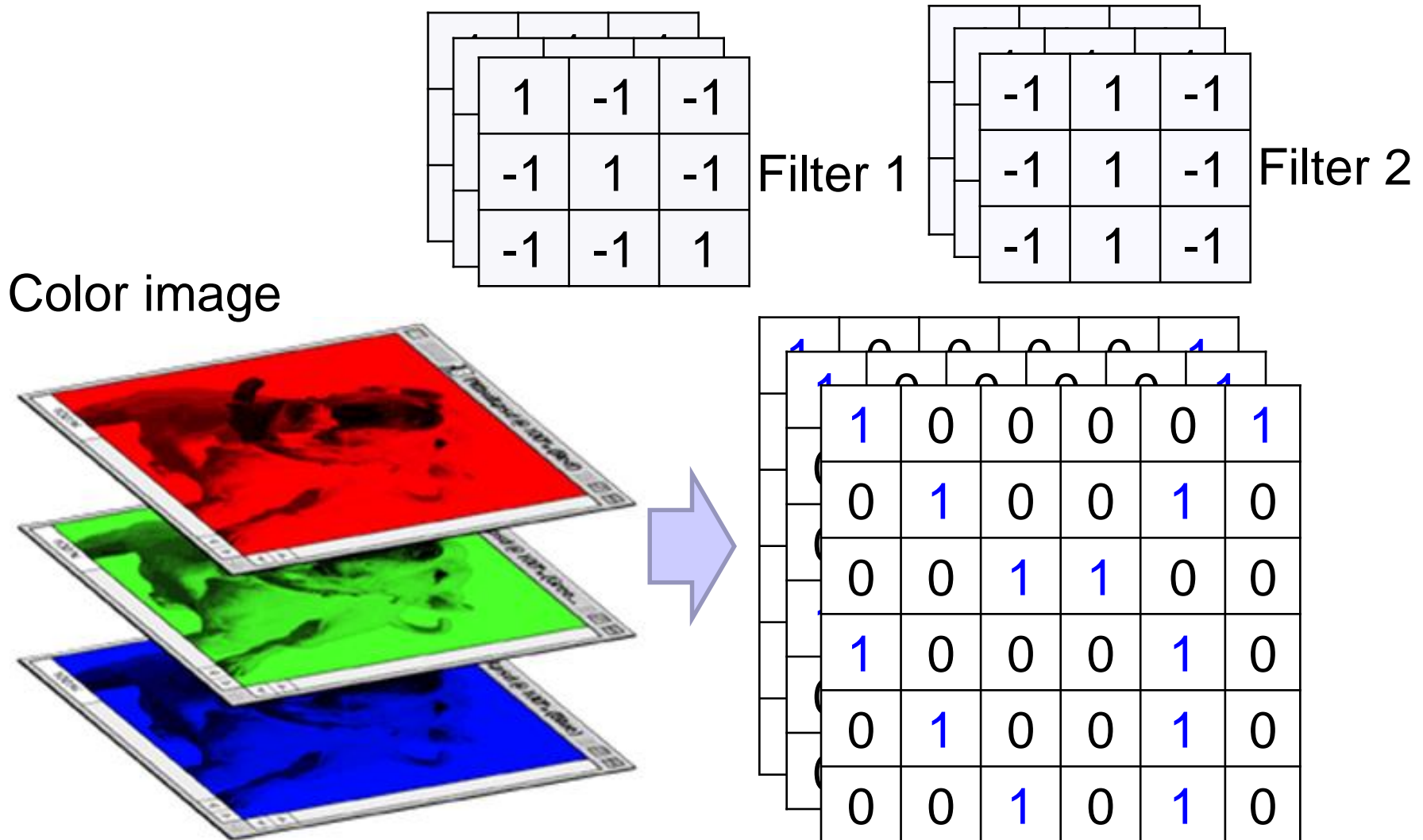
Filter 2

Repeat this for each filter

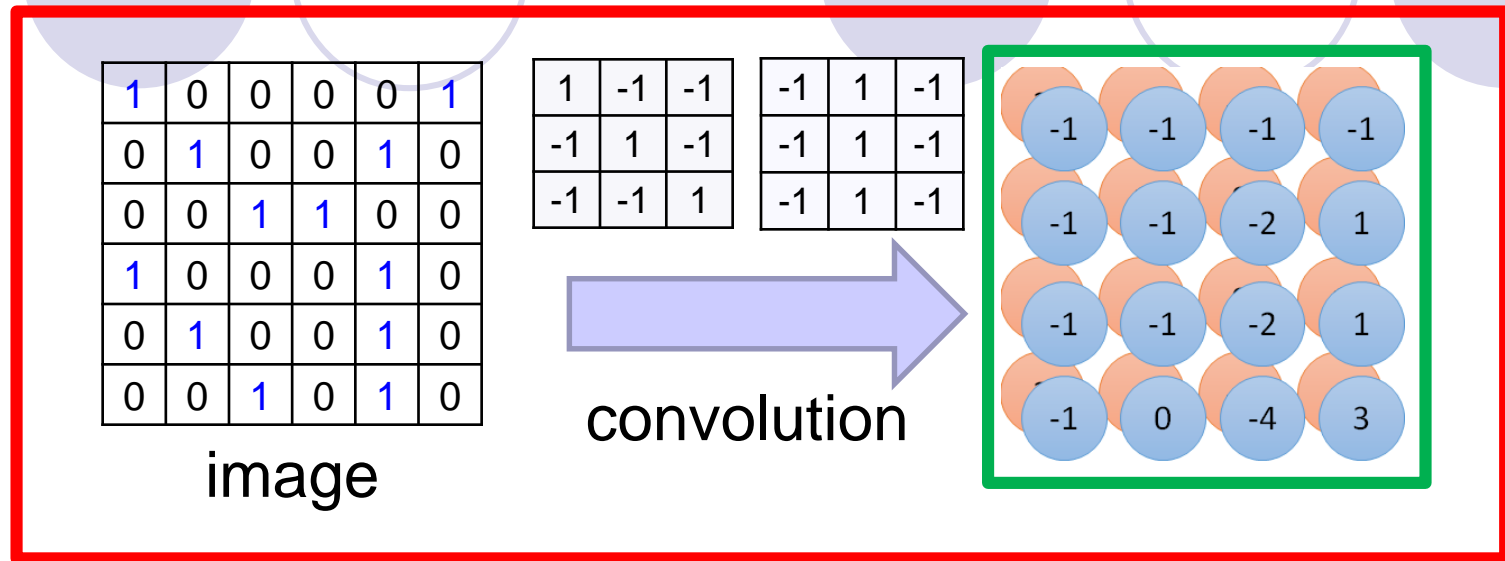


Two 4 x 4 images  
Forming 2 x 4 x 4 matrix

# Color image: RGB 3 channels

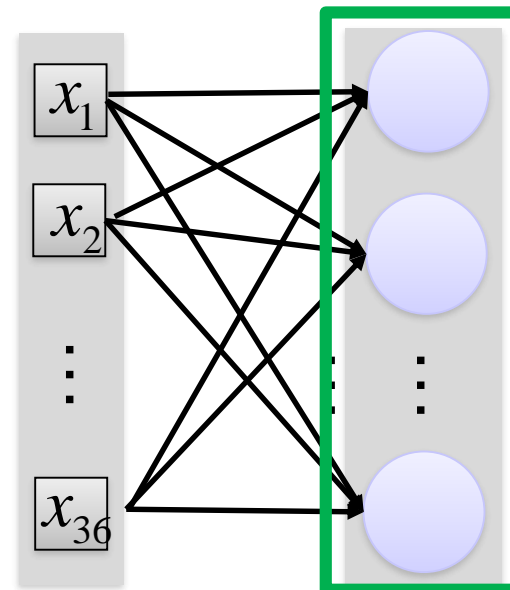


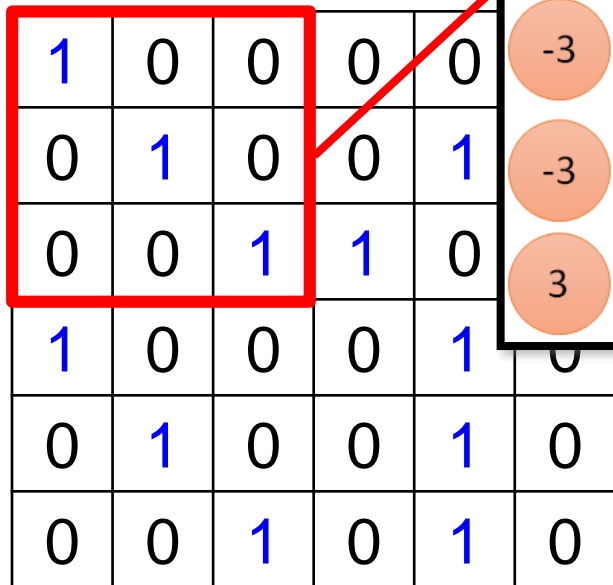
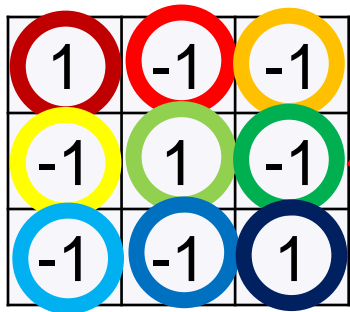
# Convolution v.s. Fully Connected



Fully-  
connected

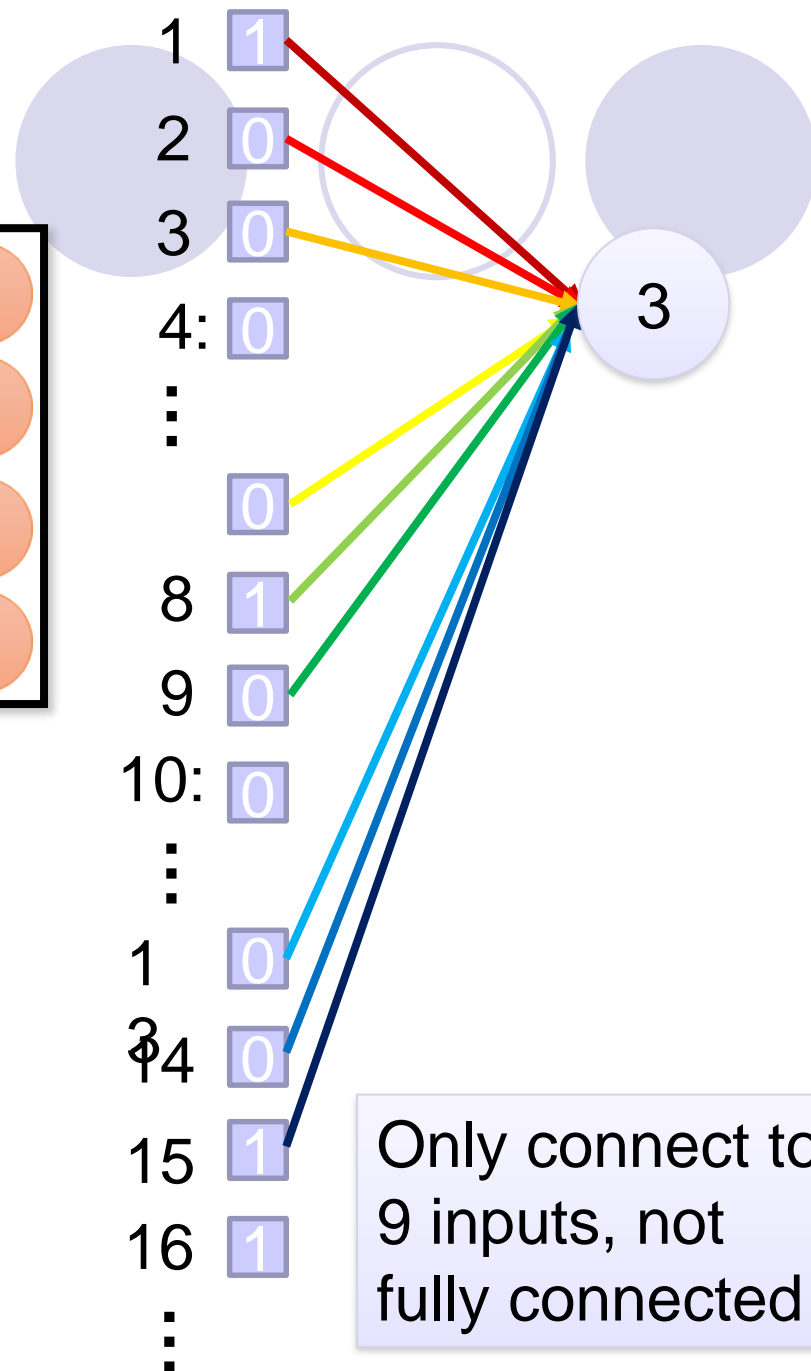
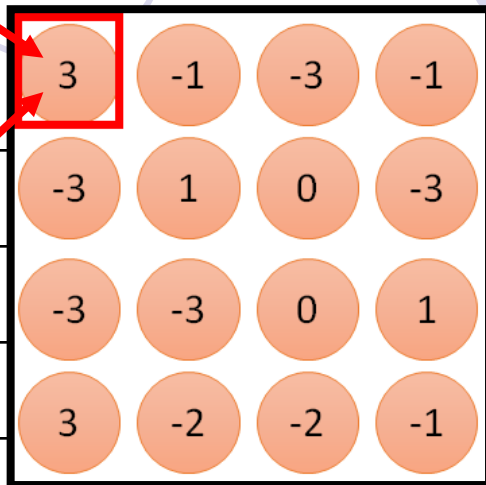
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

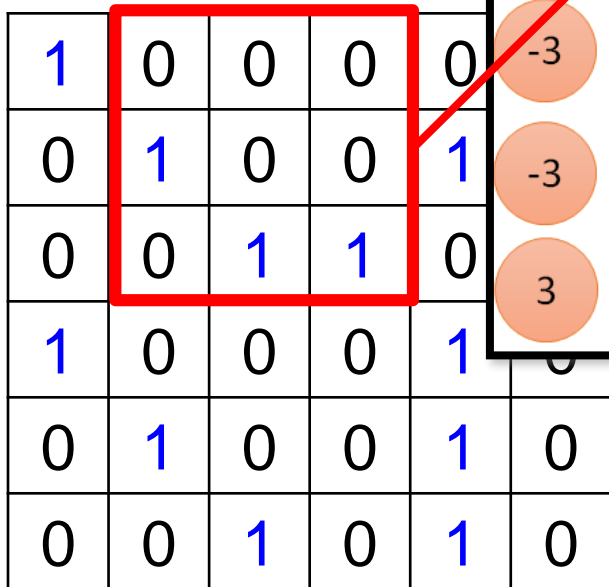
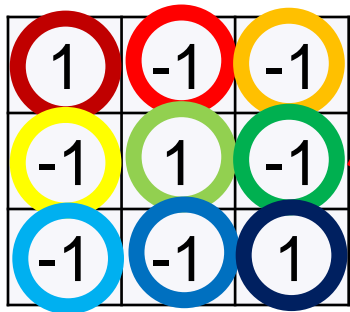




6 x 6 image

fewer parameters!

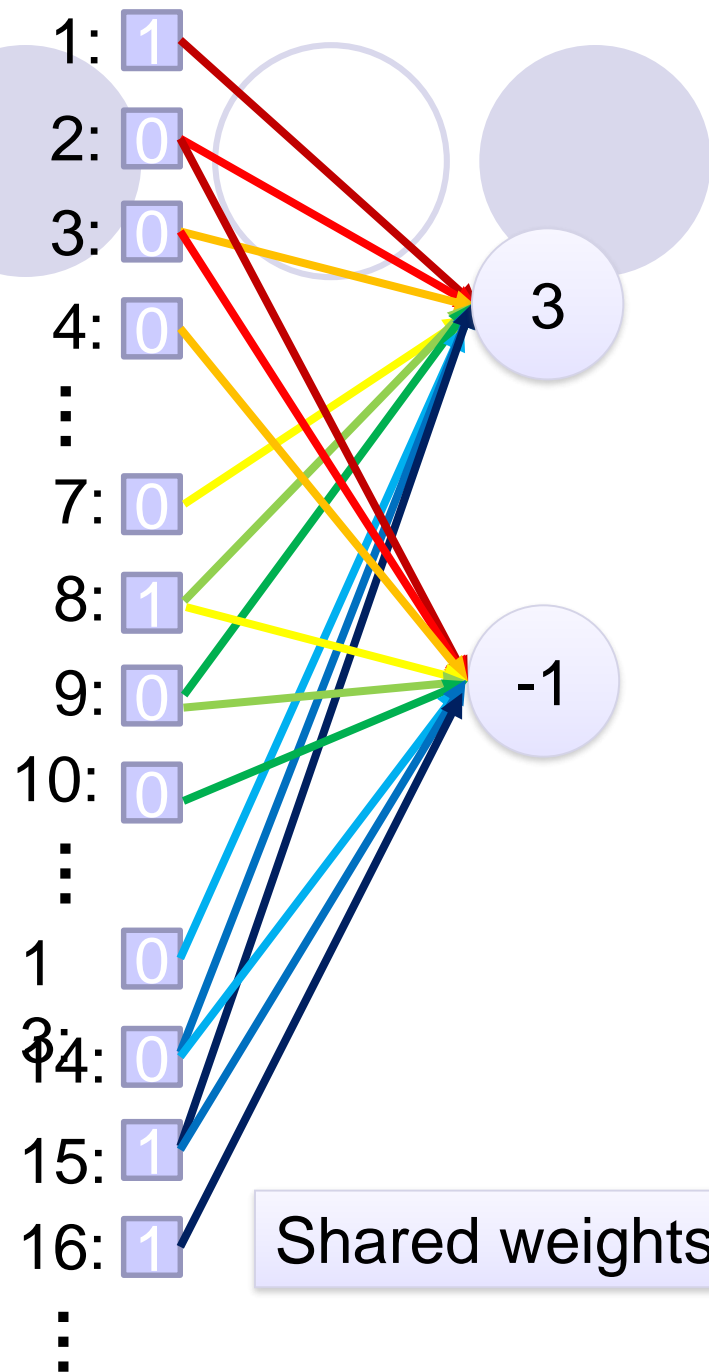
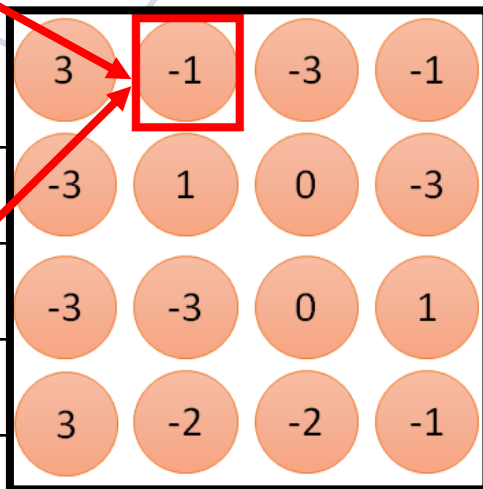




6 x 6 image

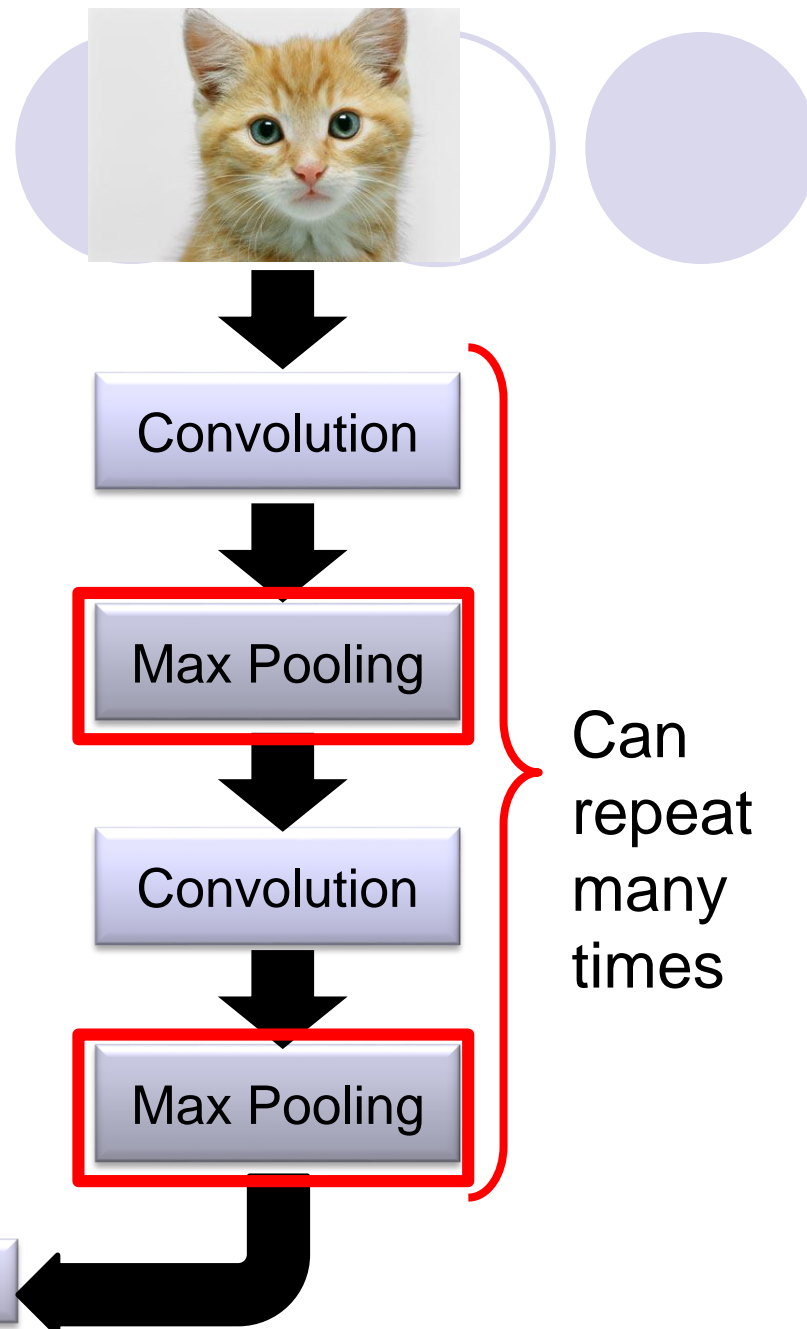
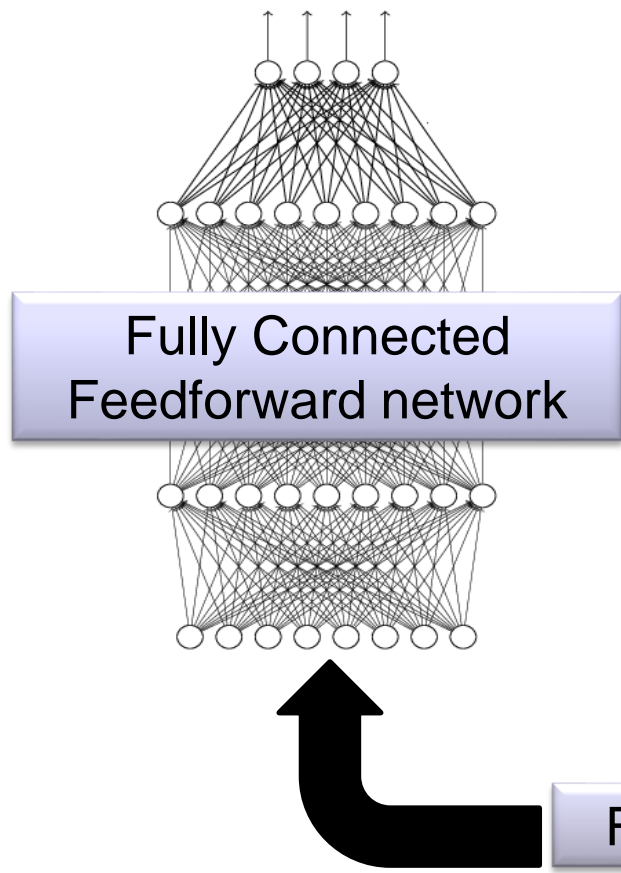
Fewer parameters

Even fewer parameters



# The whole CNN

cat dog .....



# 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



fewer parameters to characterize the image





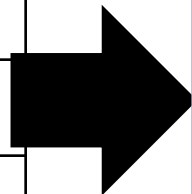
A CNN compresses a fully connected network in two ways:

- 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

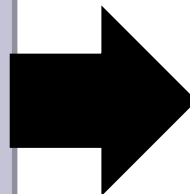
6 x 6 image



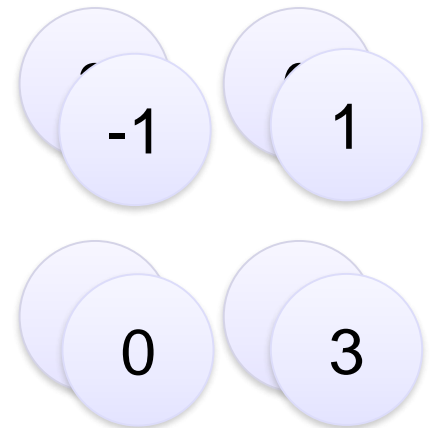
Conv



Max  
Pooling



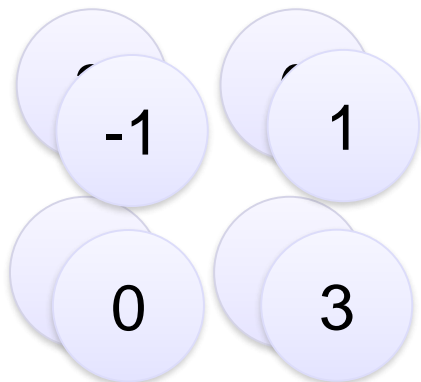
New image  
but smaller



2 x 2 image

Each filter  
is a channel

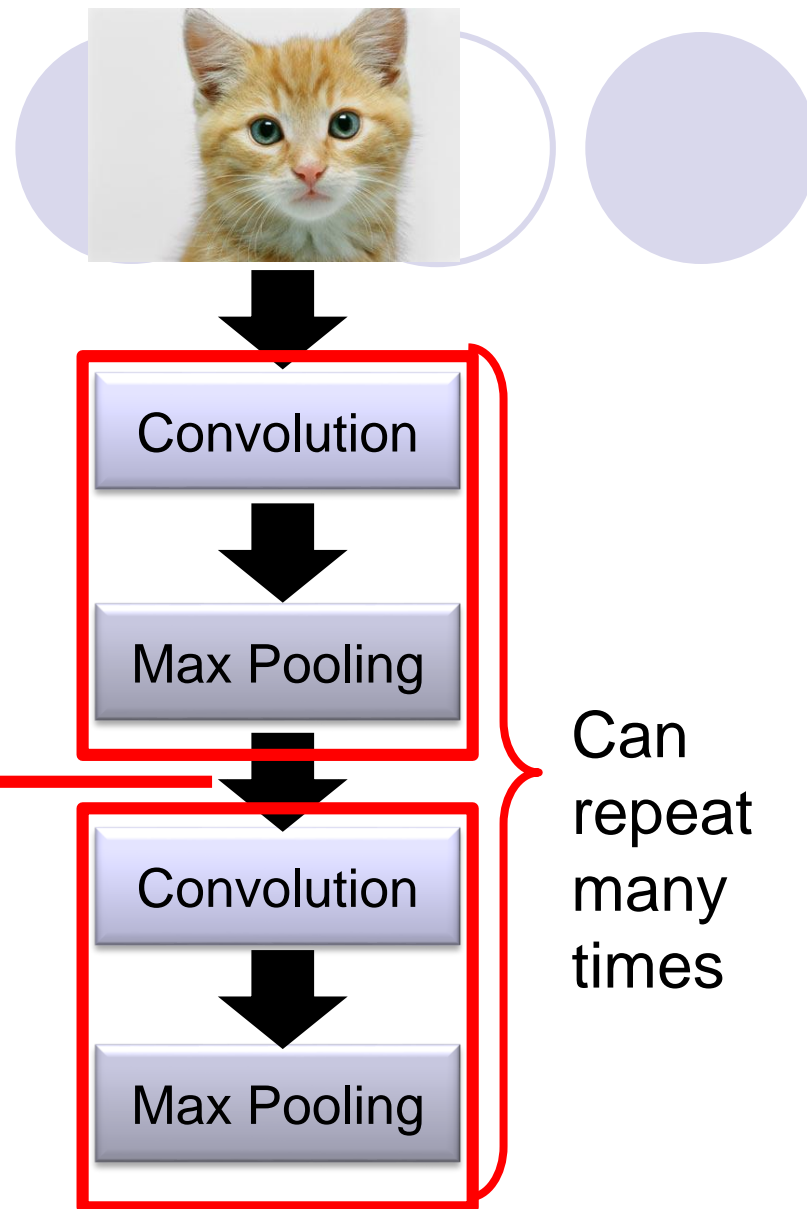
# The whole CNN



A new image

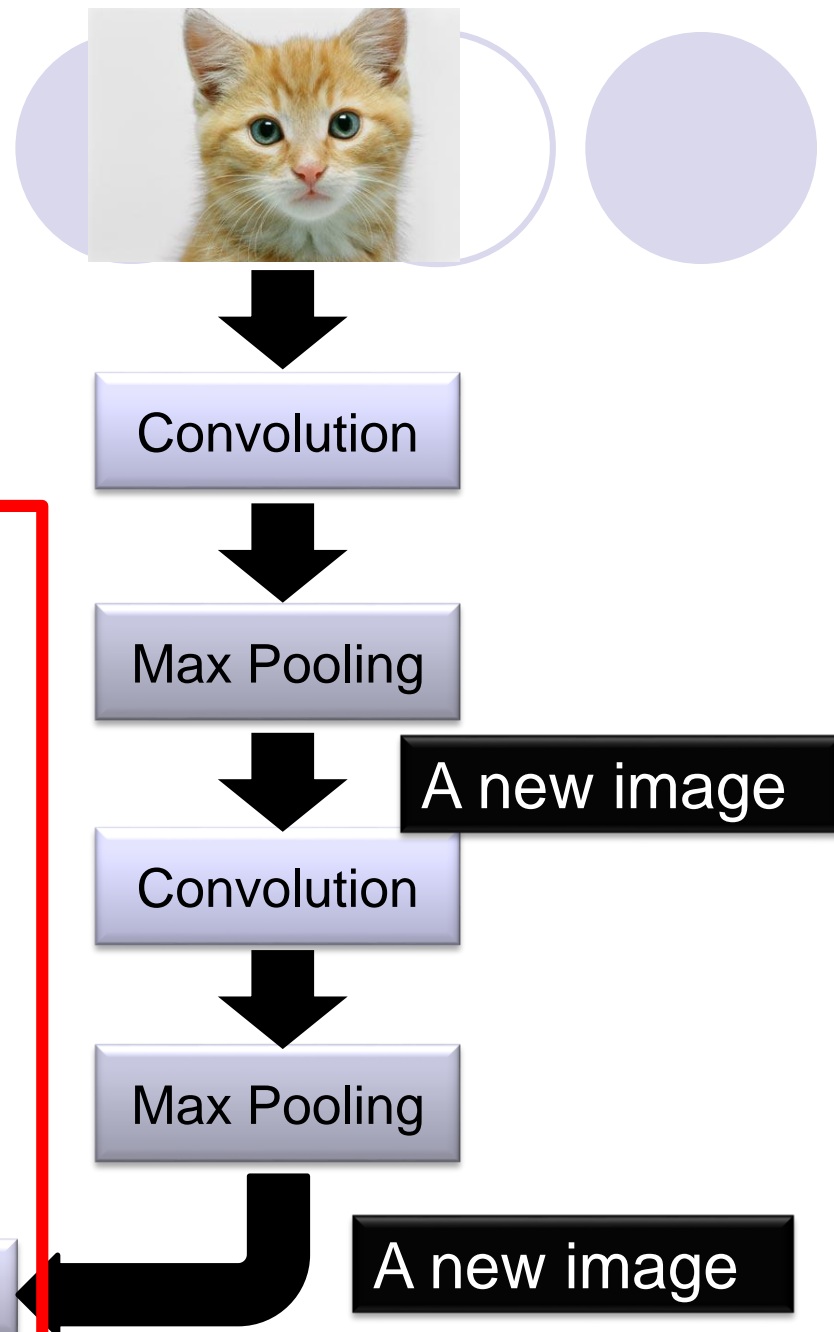
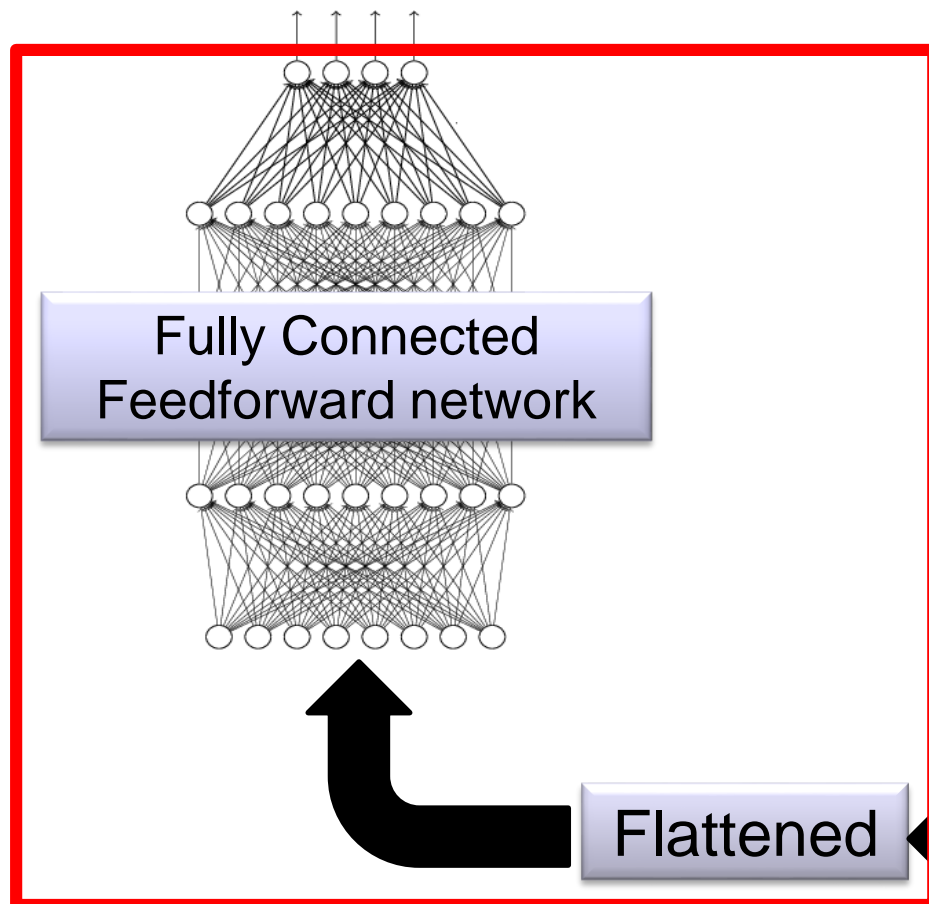
Smaller than the original image

The number of channels is the number of filters

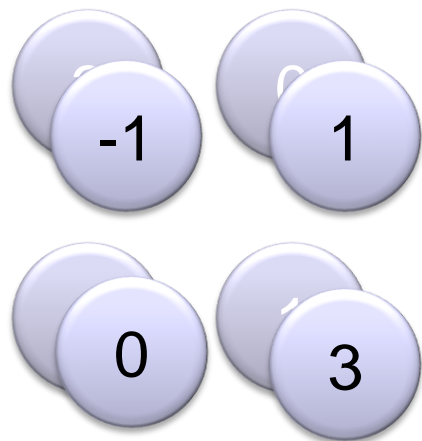


# The whole CNN

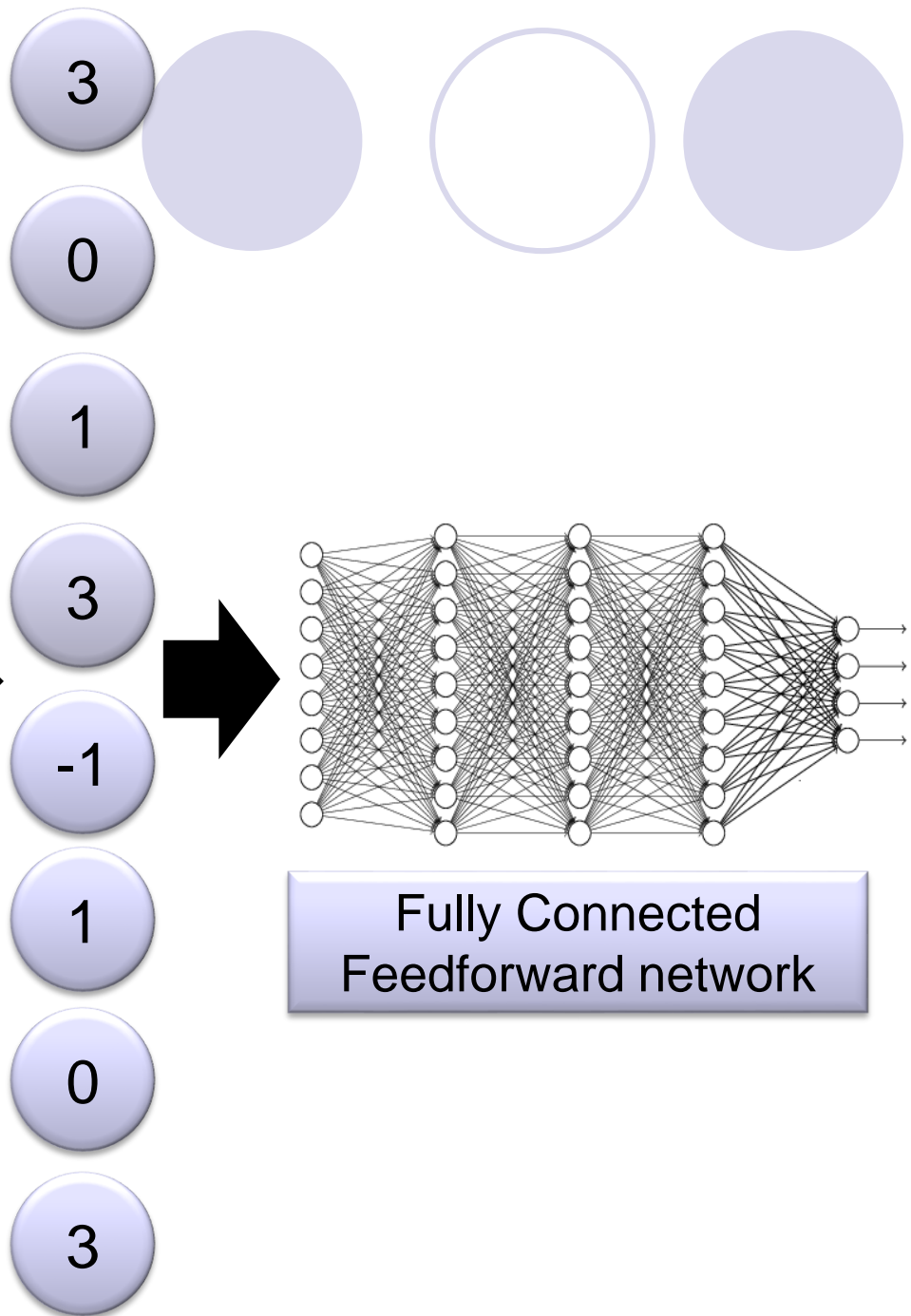
cat dog .....



# Flattening



Flattened



# 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
		-1	1	-1

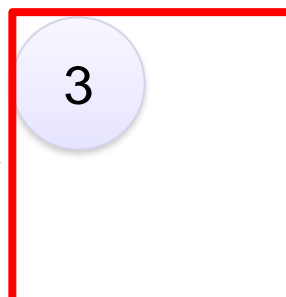
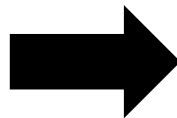
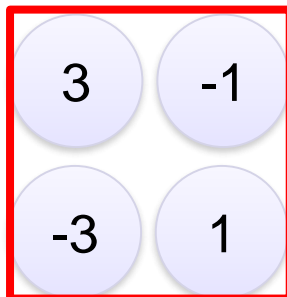
There are  
**25 3x3**  
filters.

Input\_shape = ( 28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

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



input

Convolution

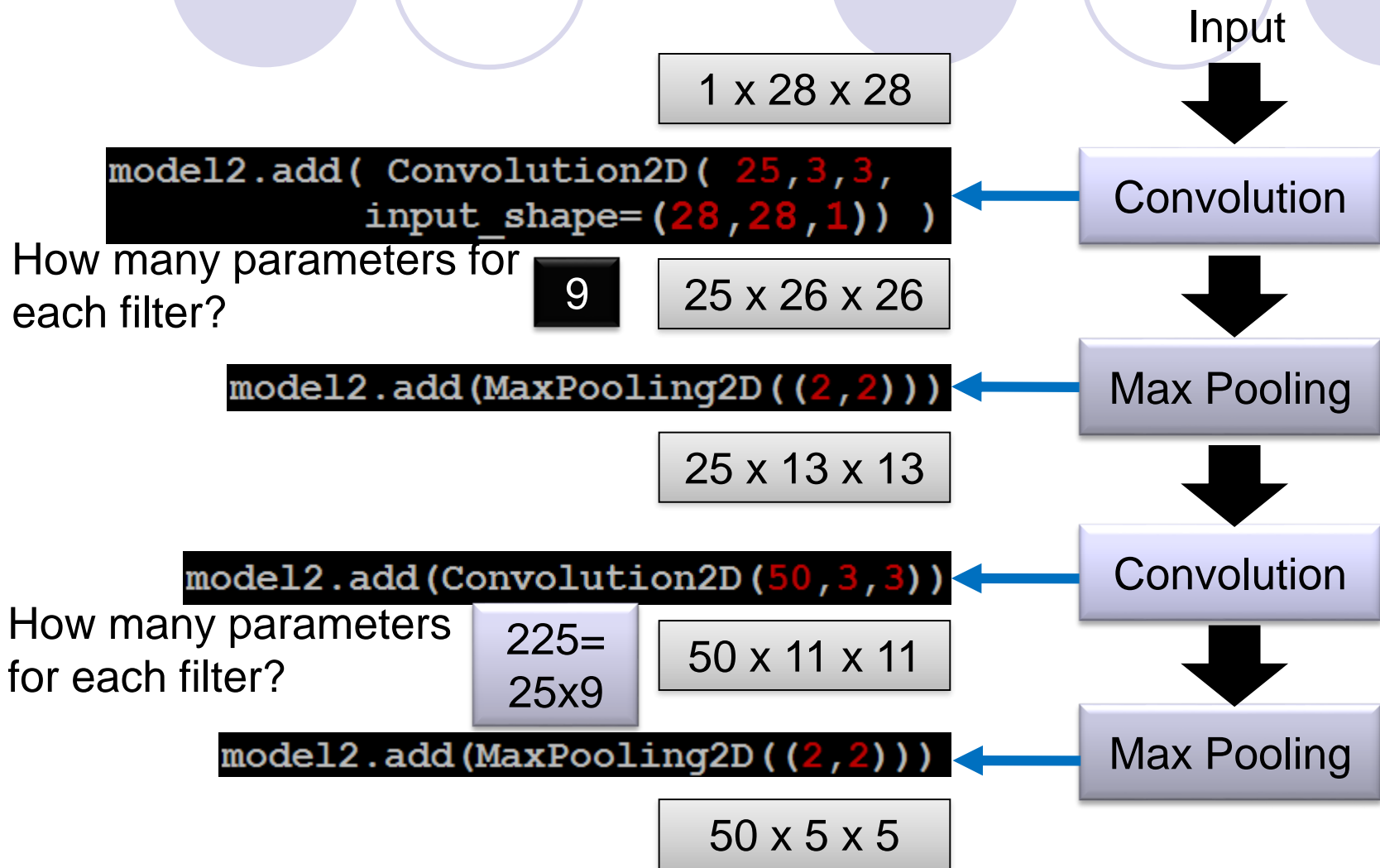
Max Pooling

Convolution

Max Pooling

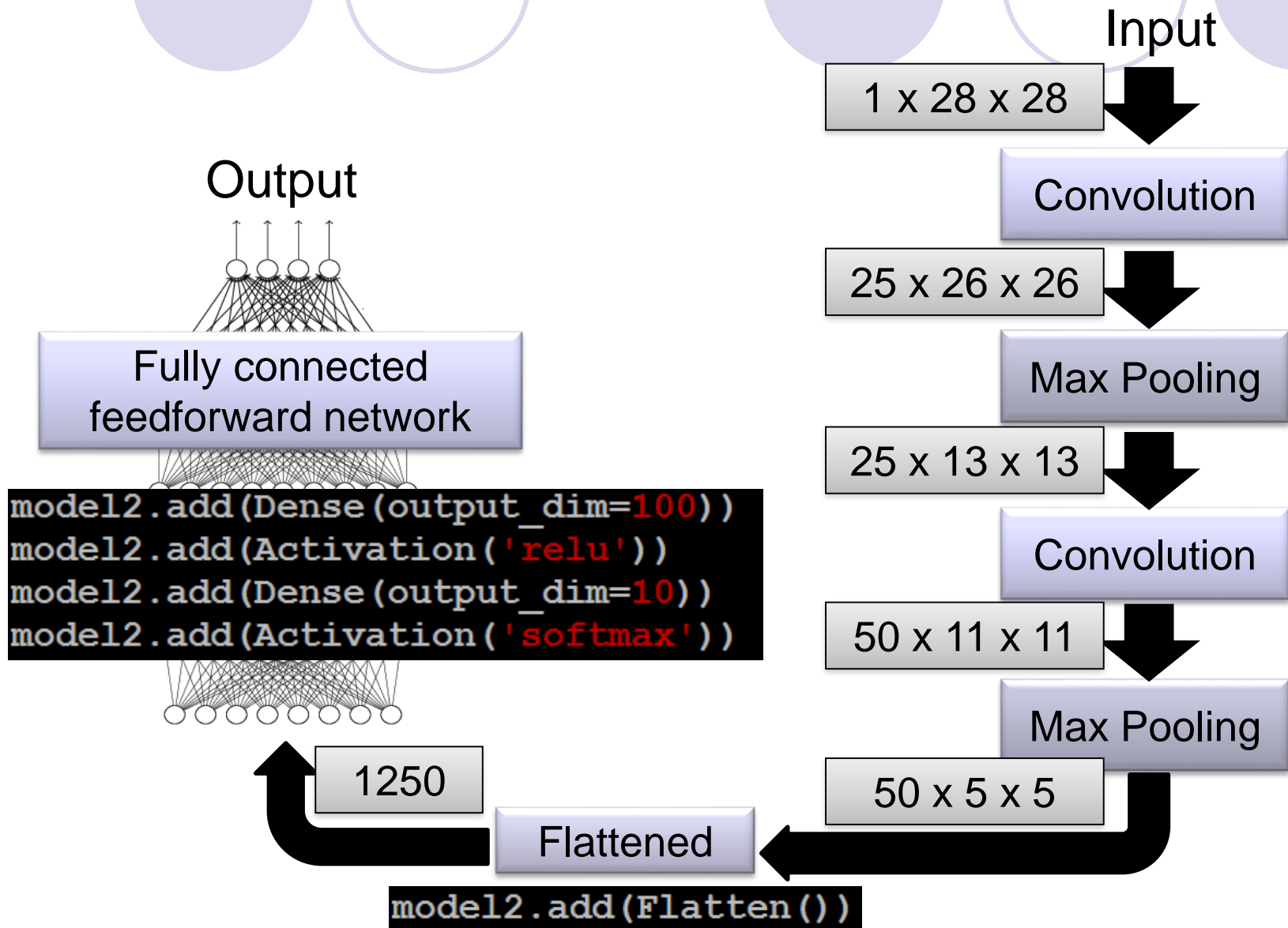
# CNN in Keras

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)**





# AlphaGo



19 x 19 matrix

Black: 1  
white: -1  
none: 0

Neural  
Network

Next move  
(19 x 19  
positions)

Fully-connected feedforward  
network can be used

But CNN performs much better

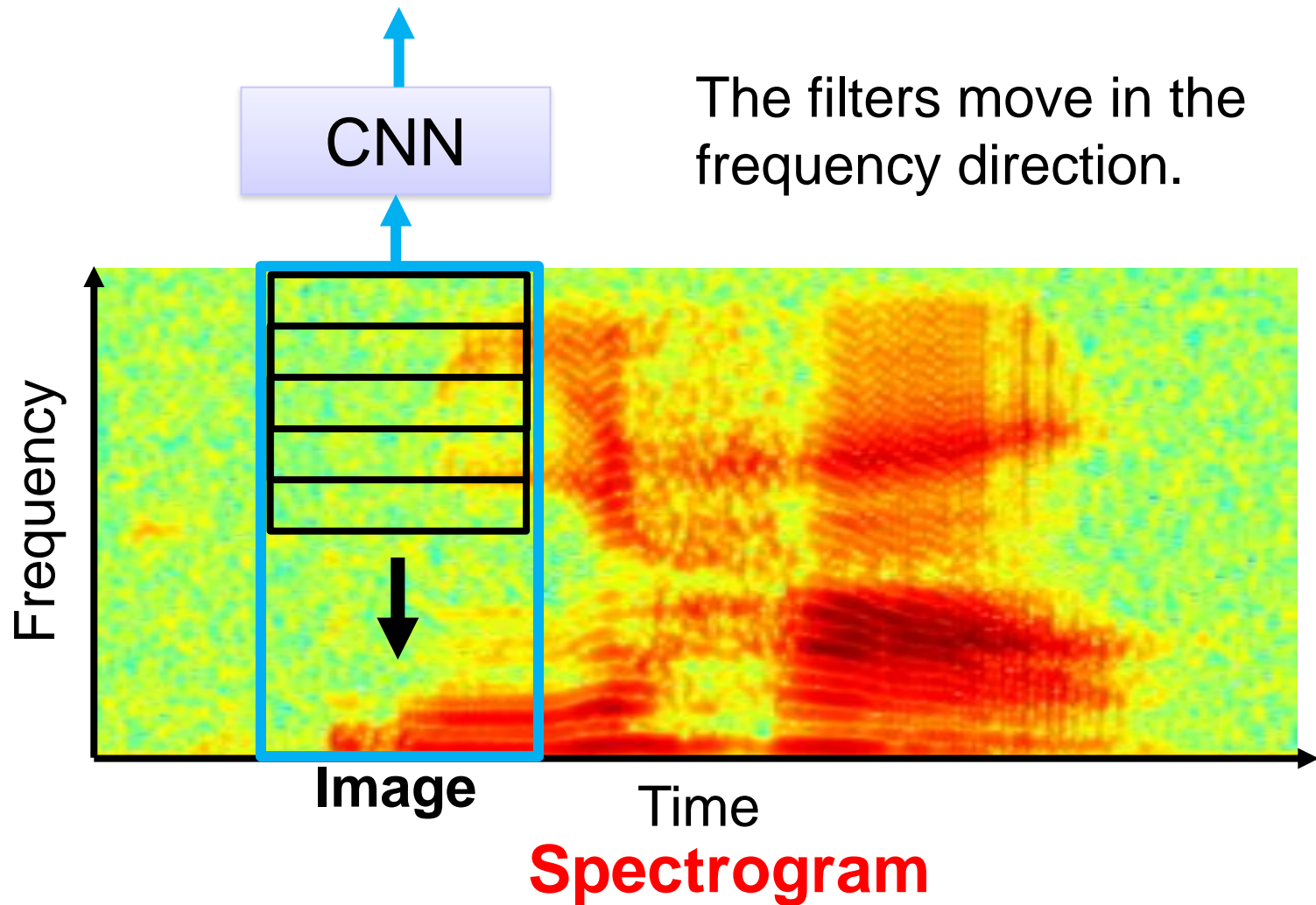
# AlphaGo's policy network

The following is quotation from their Nature article:

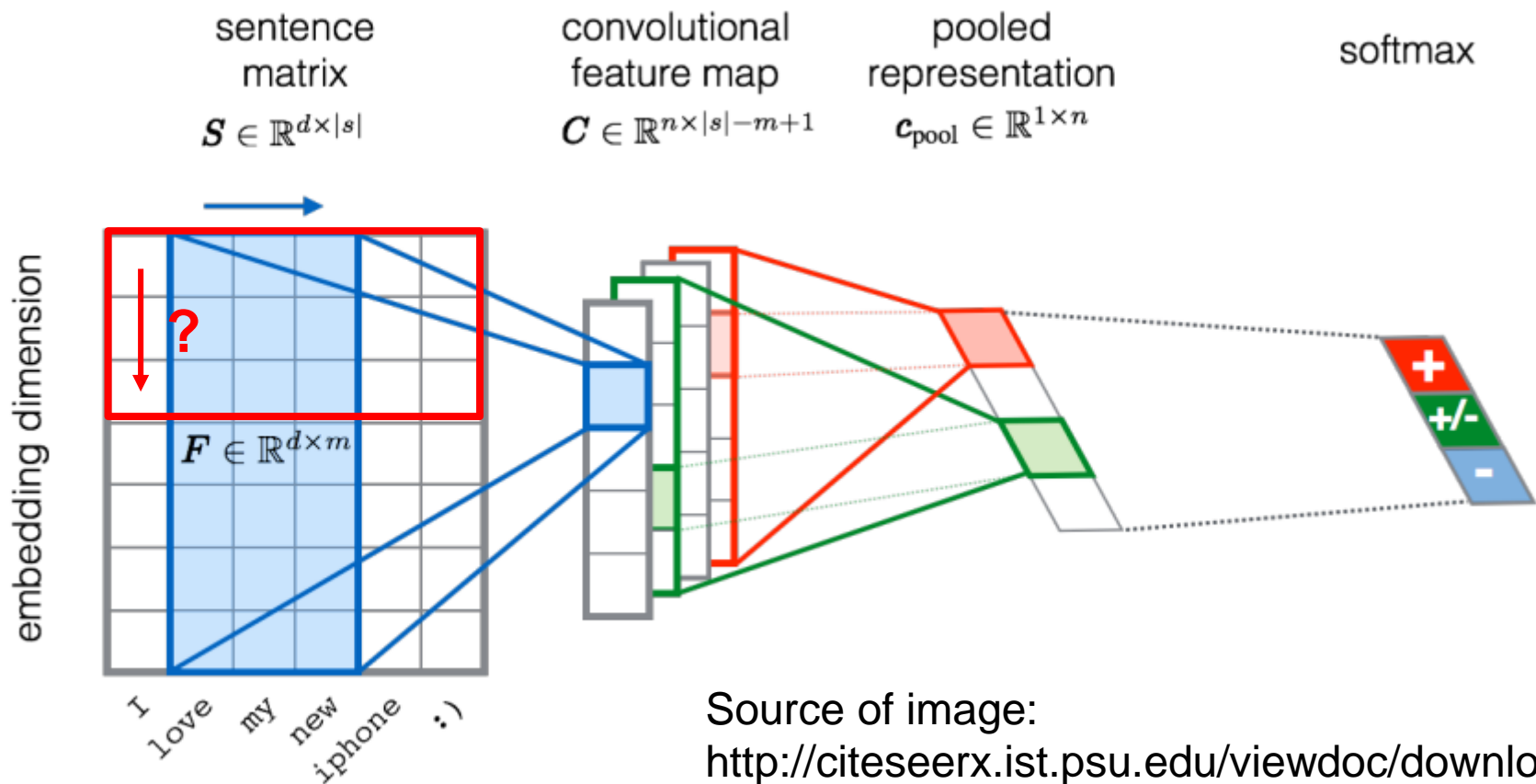
Note: AlphaGo does not use Max Pooling.

**Neural network architecture.** The input to the policy network is a  $19 \times 19 \times 48$  image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a  $23 \times 23$  image, then convolves  $k$  filters of kernel size  $5 \times 5$  with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a  $21 \times 21$  image, then convolves  $k$  filters of kernel size  $3 \times 3$  with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size  $1 \times 1$  with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used  $k = 192$  filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with  $k = 128, 256$  and 384 filters.

# CNN in speech recognition



# CNN in text classification



Source of image:  
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.6858&rep=rep1&type=pdf>