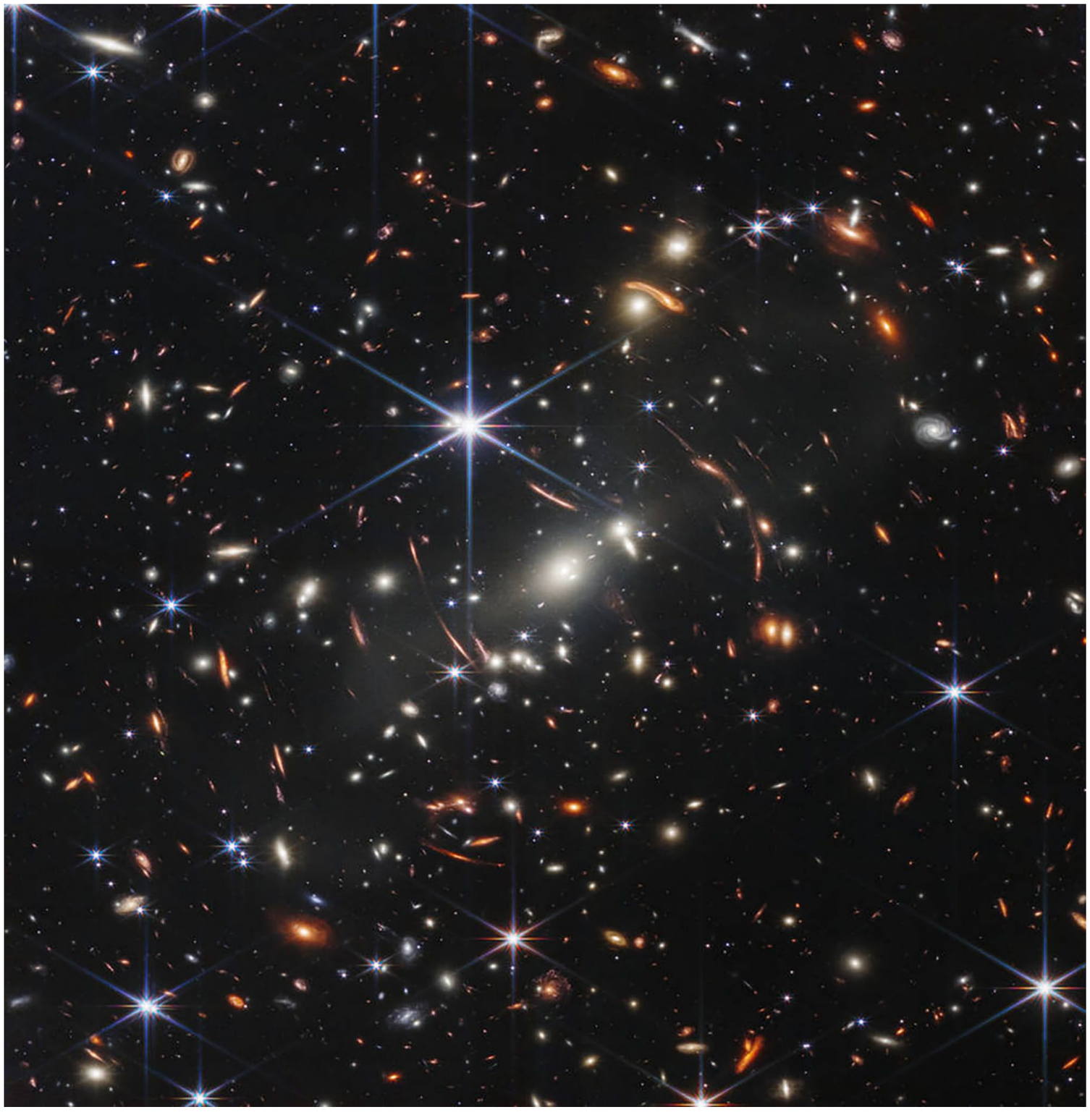
James Webb Space Telescope



The landscape of “mountains” and “valleys”

A visual grouping of five galaxies

CONVOLUTIONAL NEURAL NETWORK

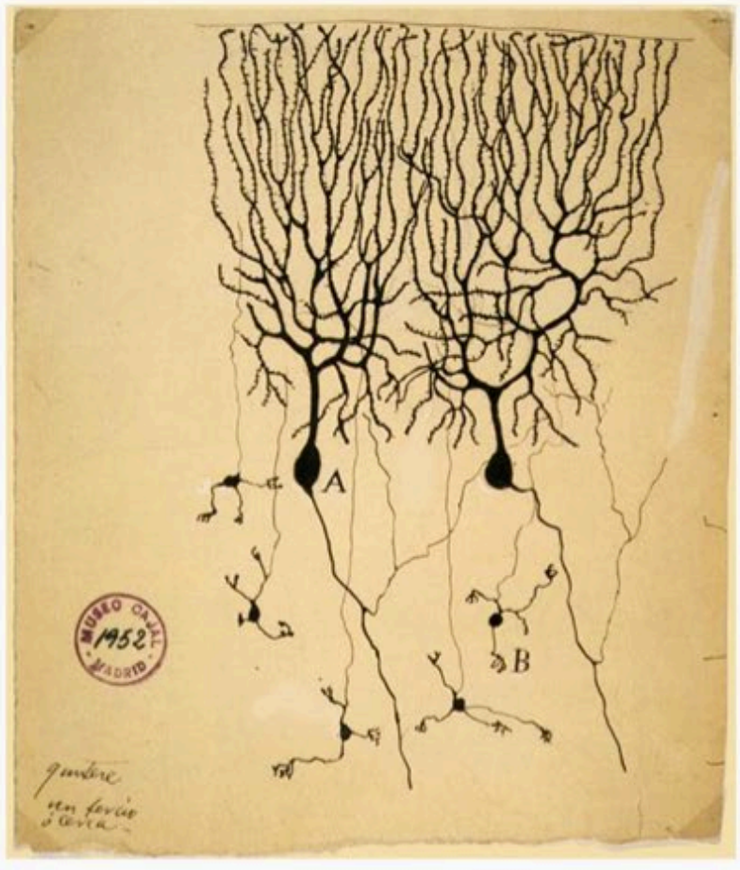
Mahdi Roozbahani

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

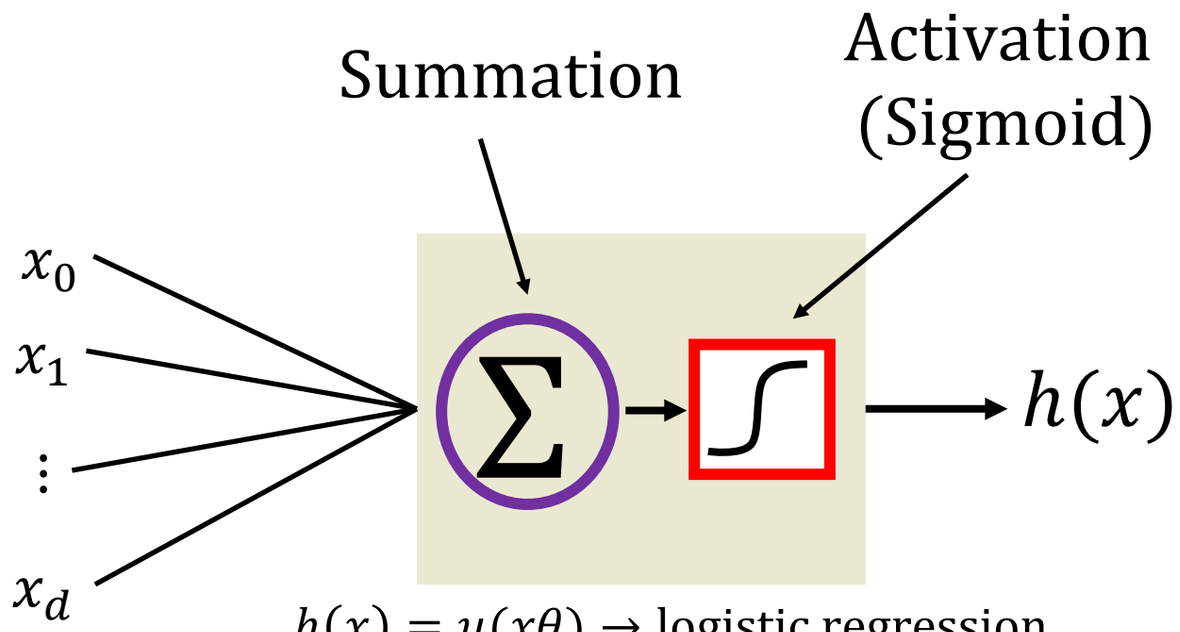
Inspiration from Biological Neurons



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

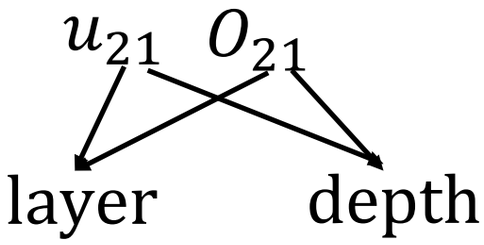


$h(x) = u(x\theta) \rightarrow$ logistic regression

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

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



1st hidden layer

Bias

Neuron or node

θ_0

θ_2

θ_1

θ_3

θ_4

θ_5

θ_6

u_{11}

u_{12}

u_{21}

O_{11}

O_{12}

O_{21}

$f(x)$

x_i

feature x_i

weight

Activation (Sigmoid)

Summation

Sigmoid output

Regression

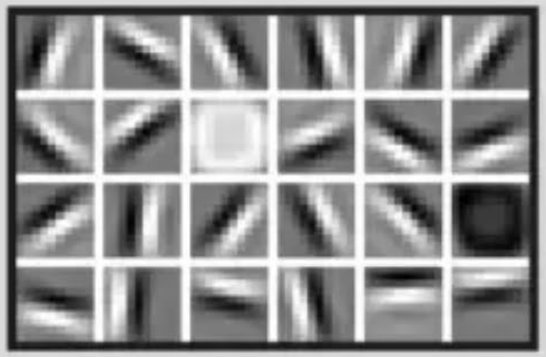
Prediction

Summation output

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.



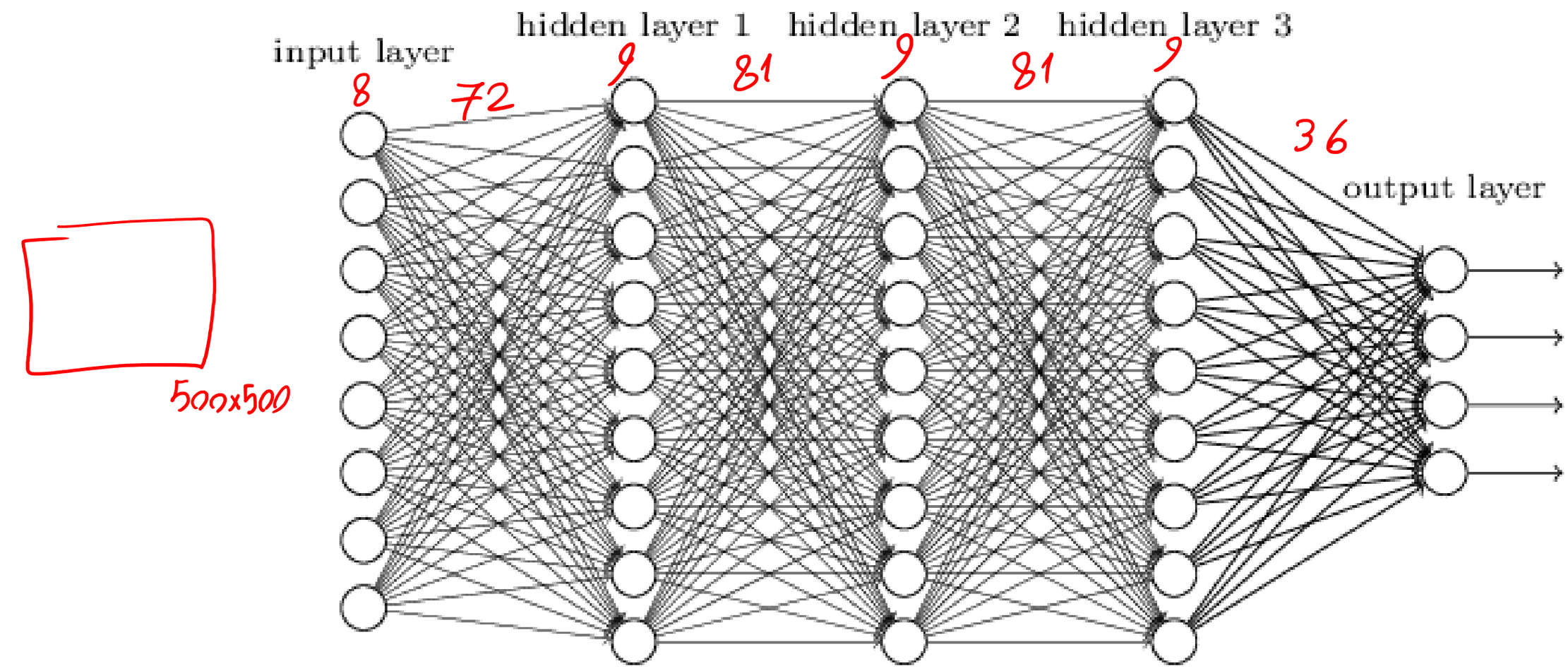
Layer 2: The computer learns to identify edges and simple shapes.

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

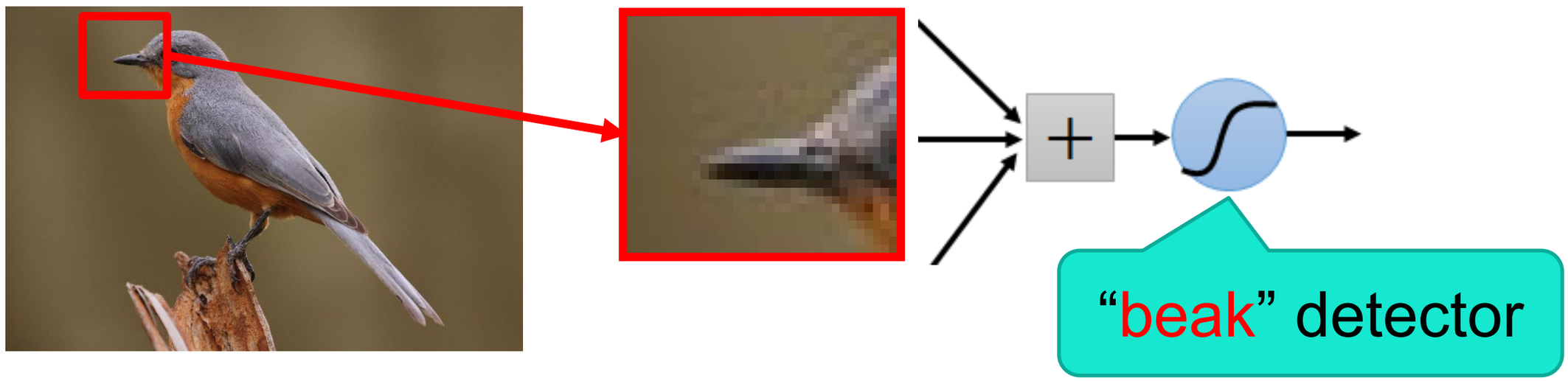
- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?



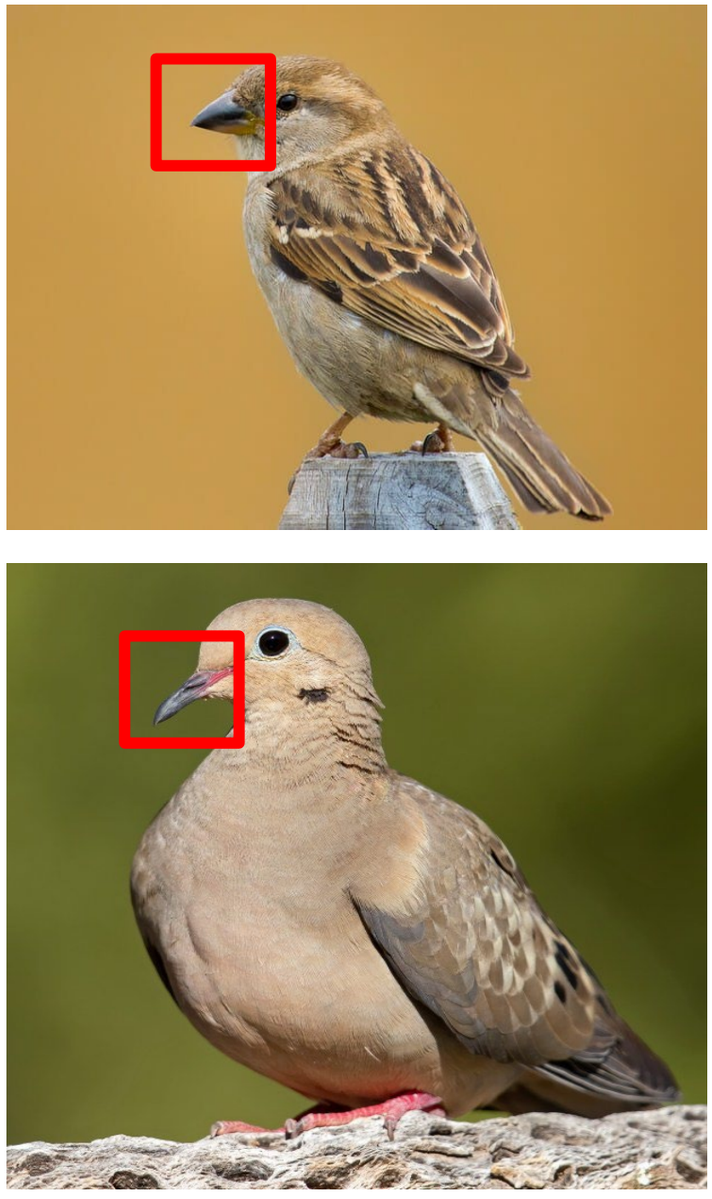
Consider learning an image:

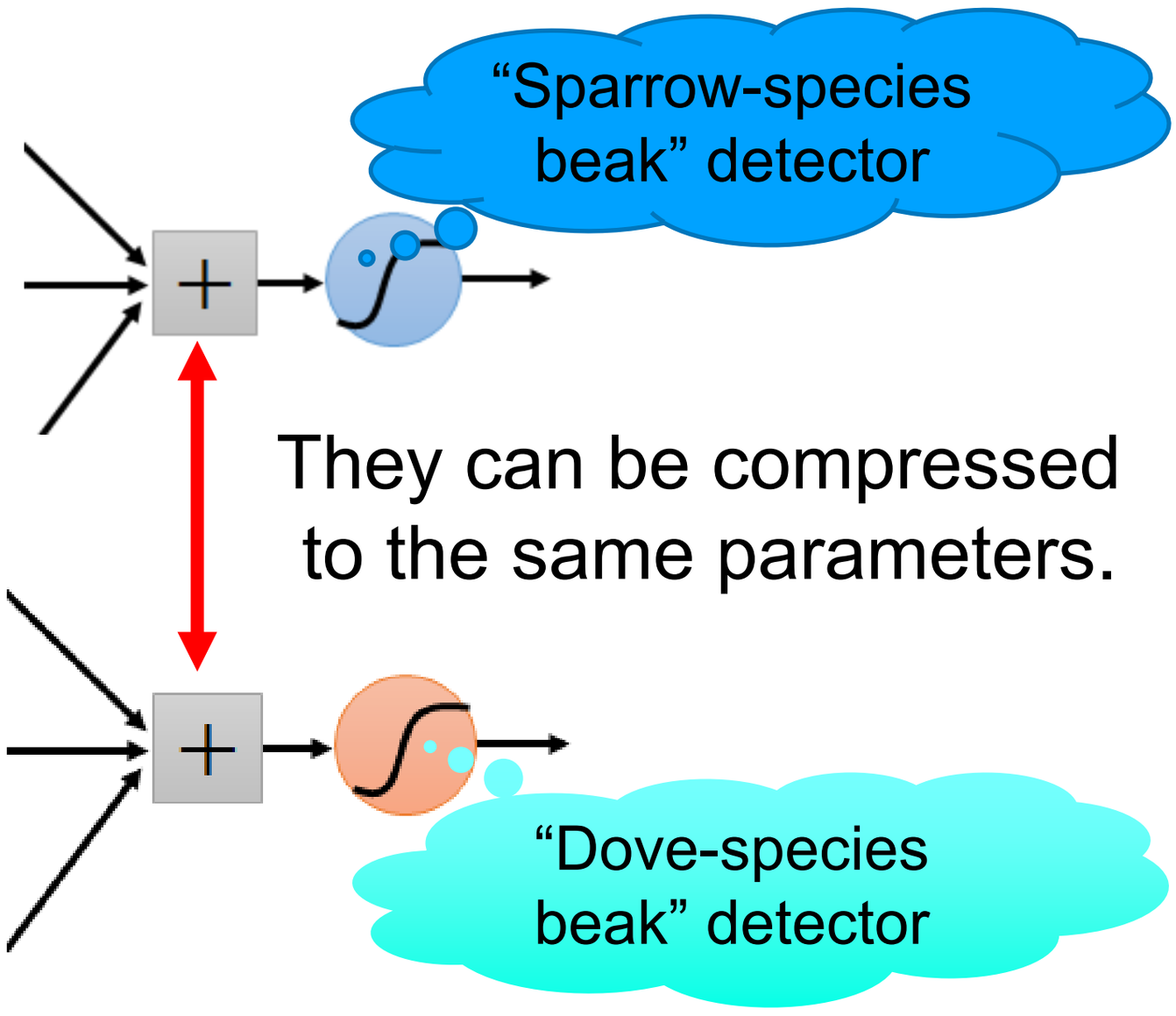
- Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters



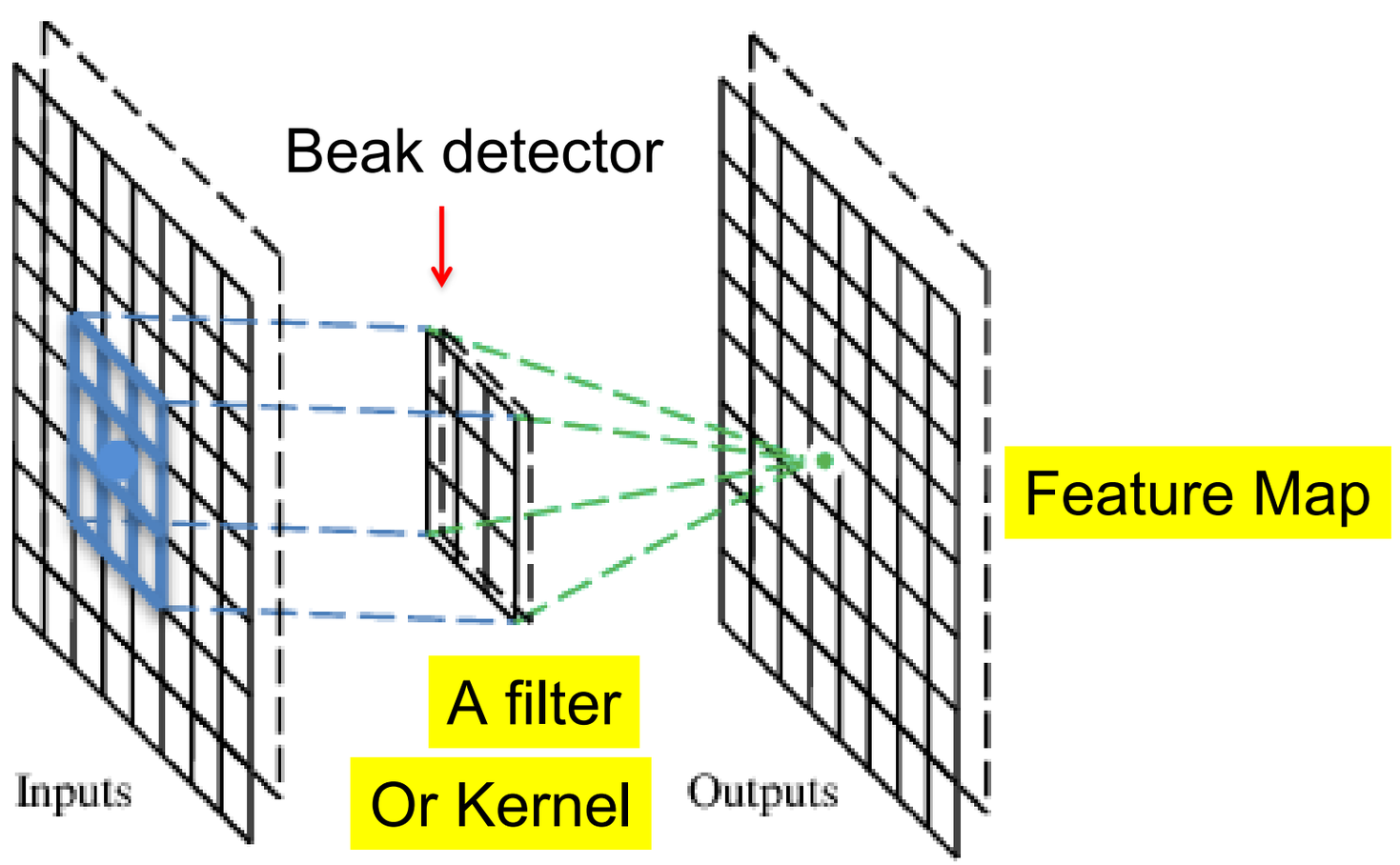
Same pattern appears in different places:
They can be compressed!
What about training a lot of such “small” detectors and each detector must “move around”.





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

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

Dot product →

3 -1

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

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

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

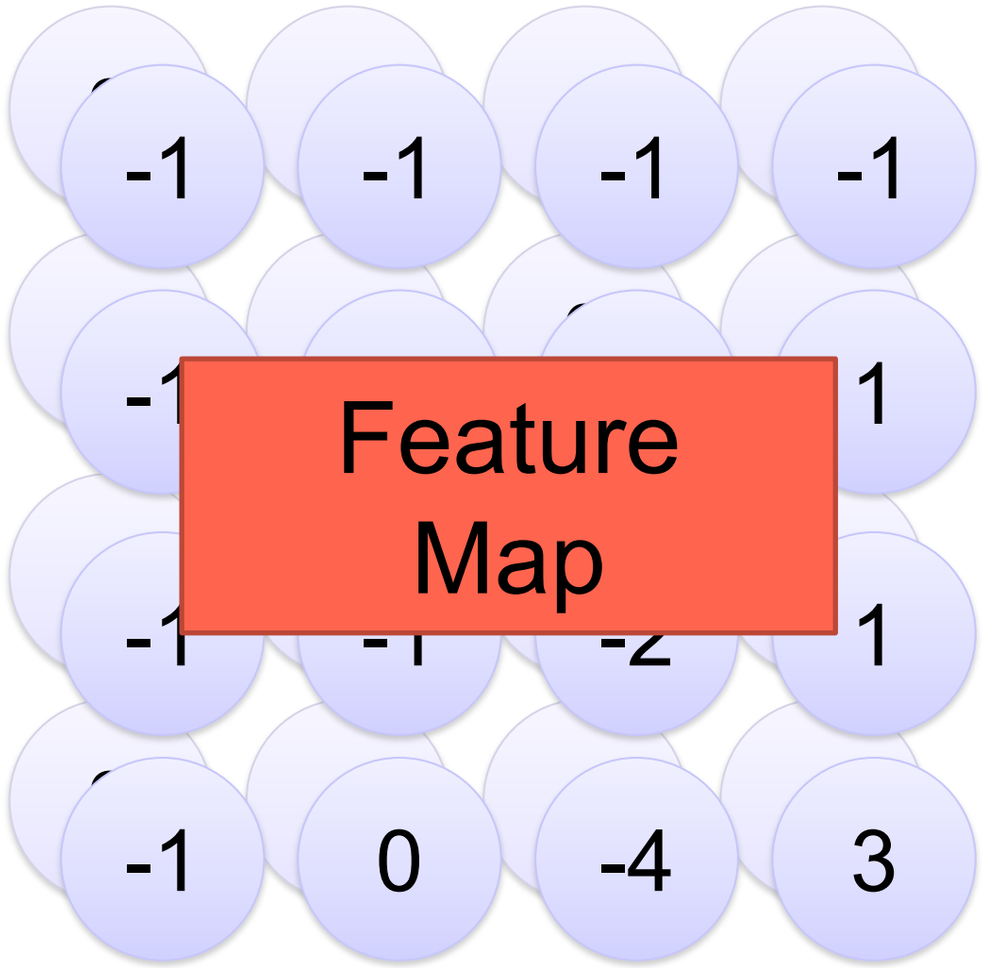
Filter 2

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



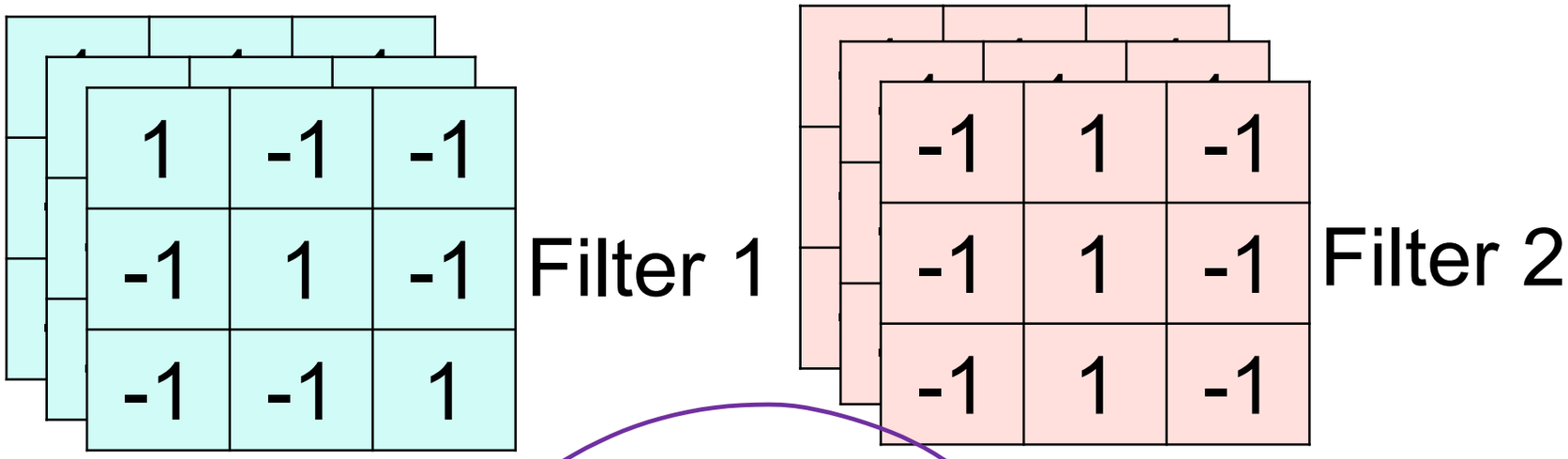
Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB 3 channels

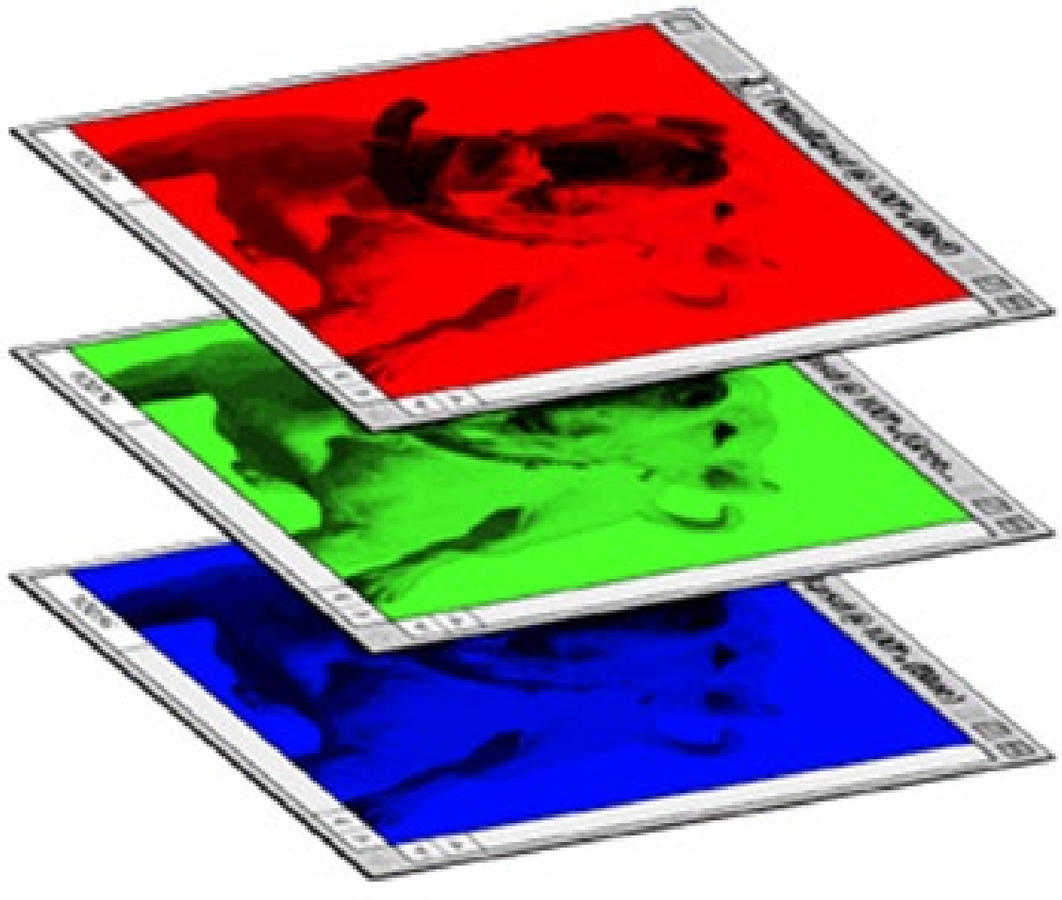
sub-matrix [1 0 0 . . . . .]$_{1x27}$ = sub

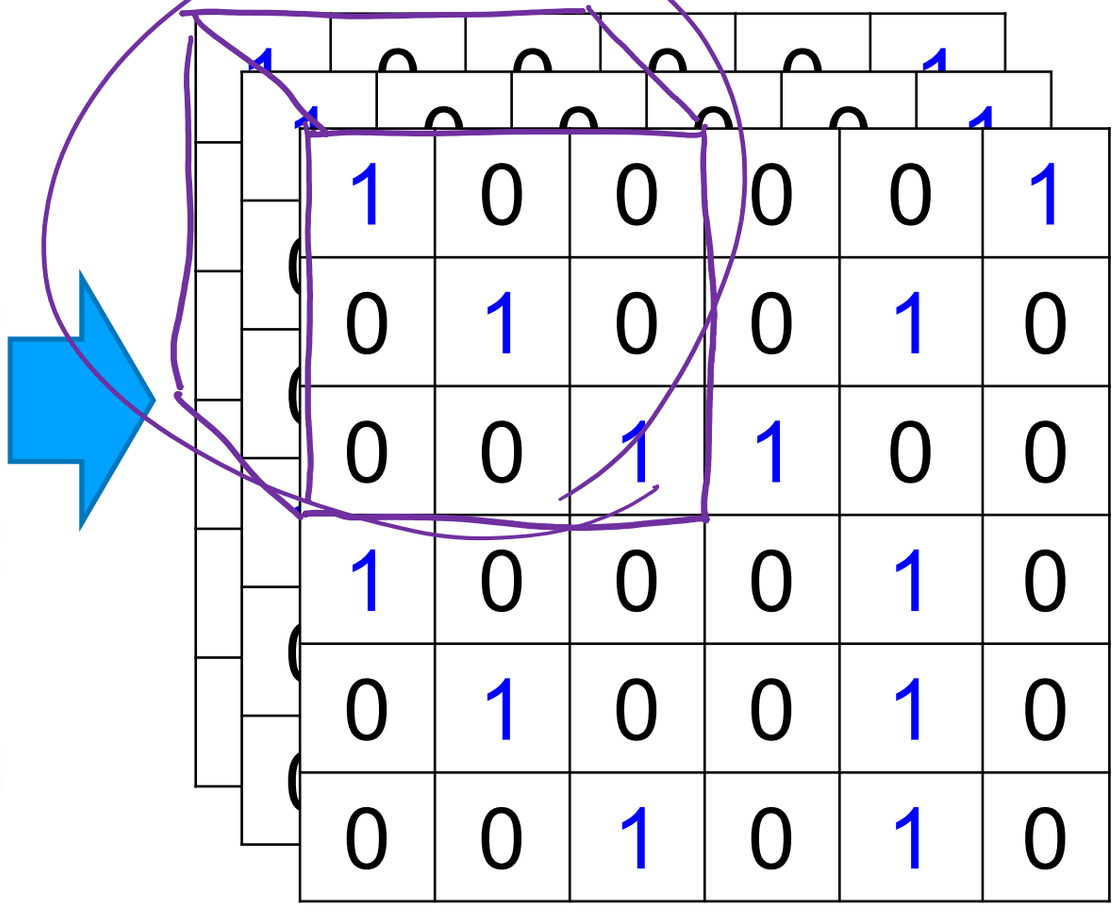
filter [1 -1 -1 . . . . .]$_{x27}$ = f

sub.f = IR

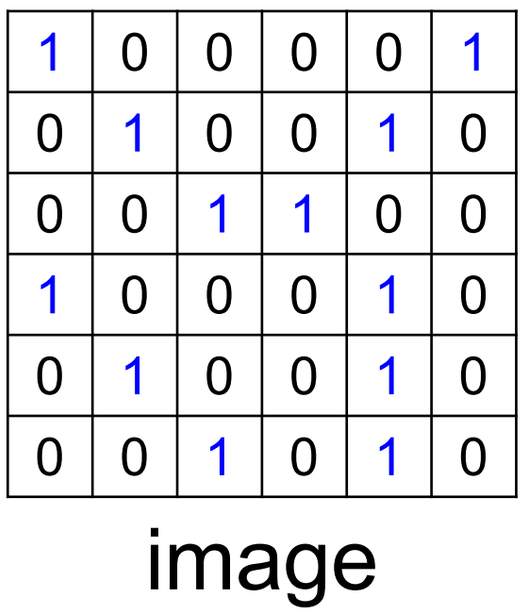


Color image

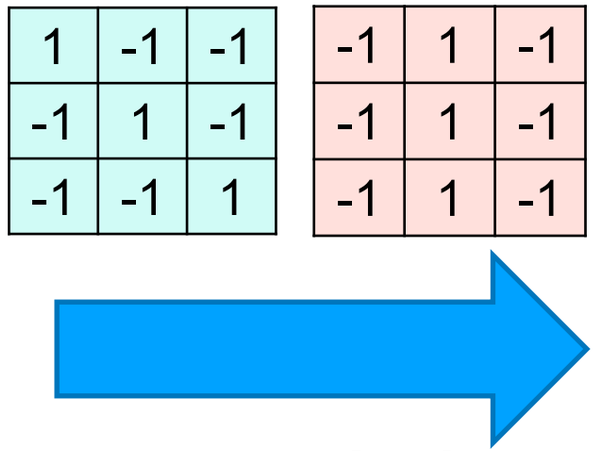




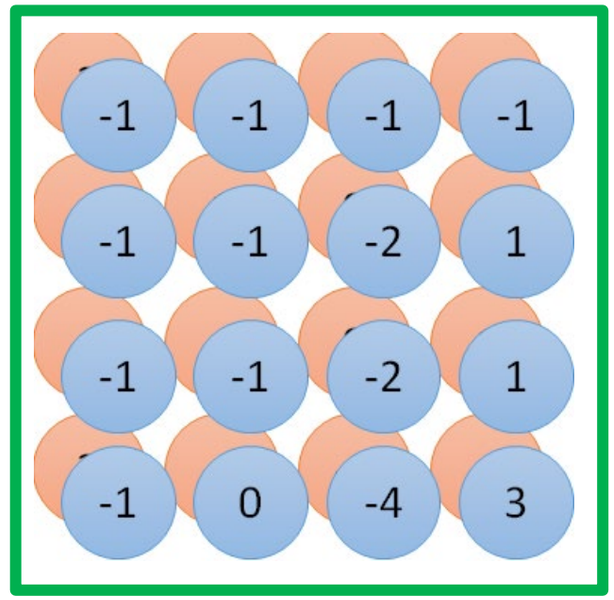
Convolution v.s. Fully Connected



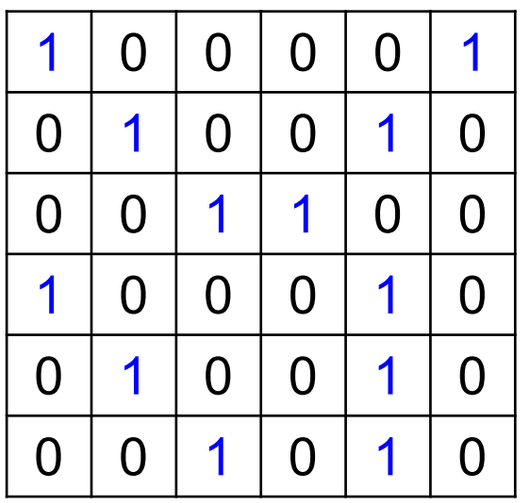
image

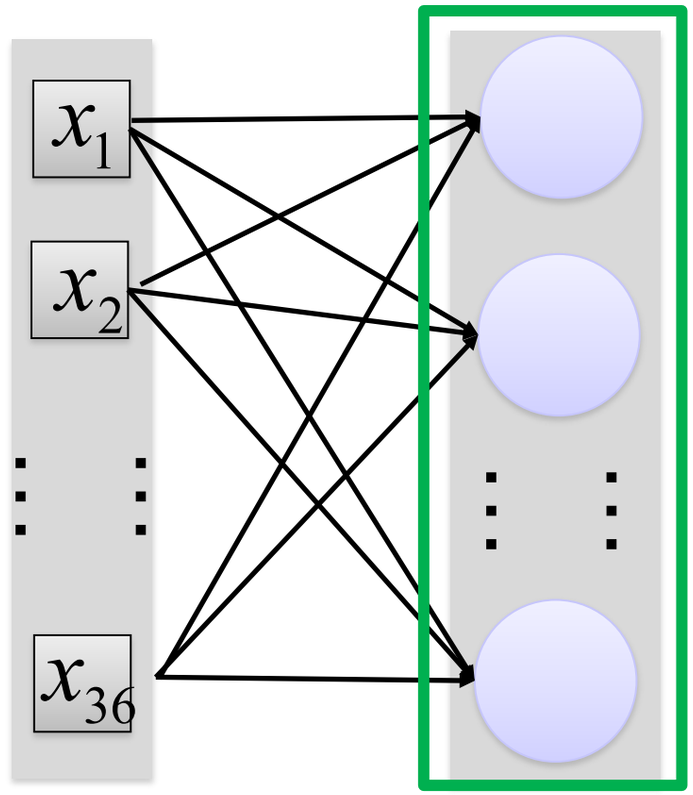


convolution



Fully-connected

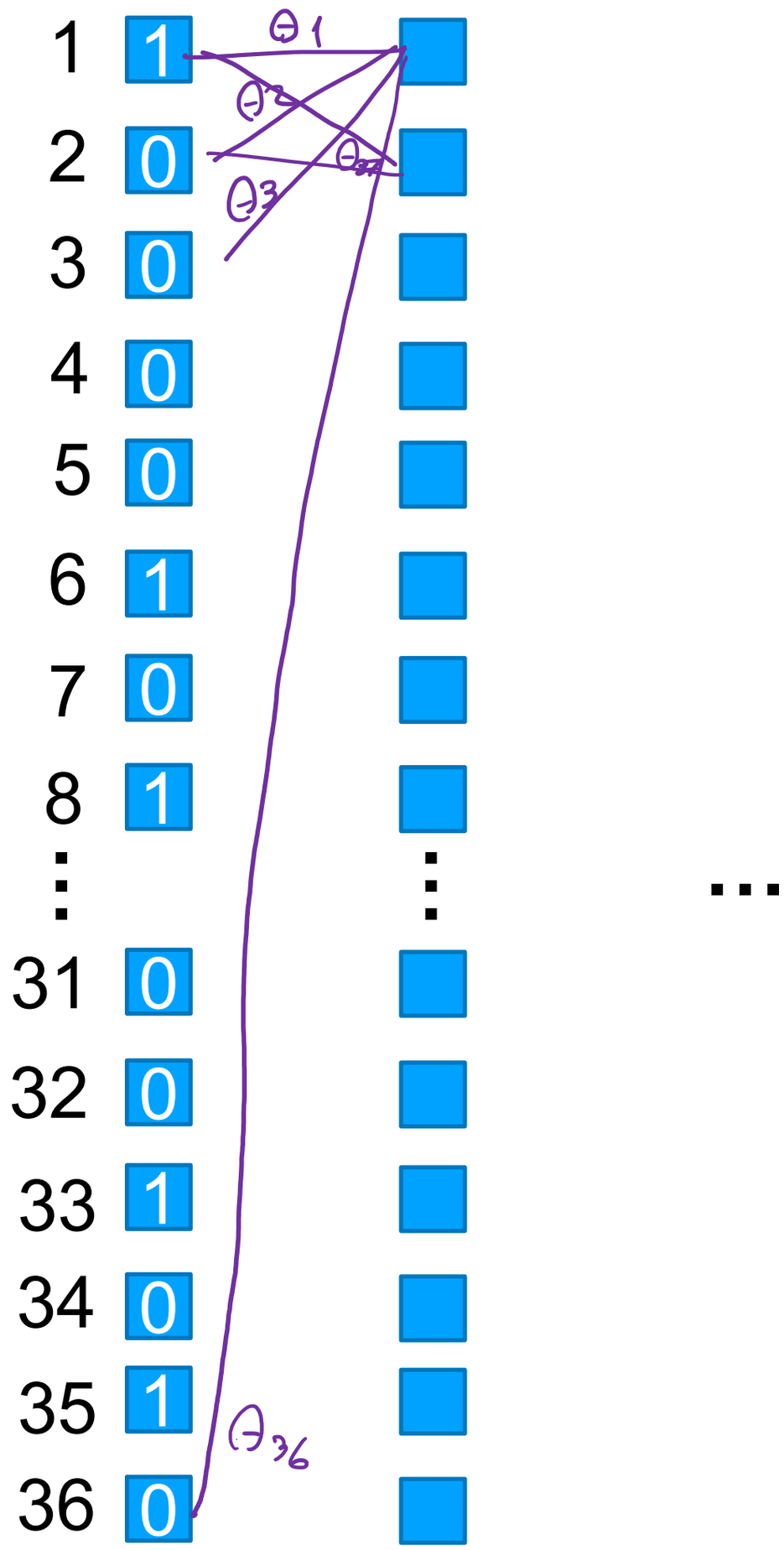


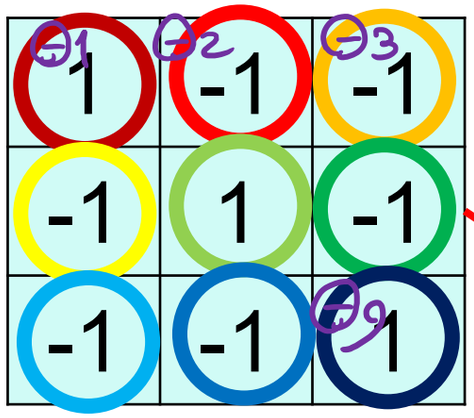


Conventional Fully Connected layers (FC layers)

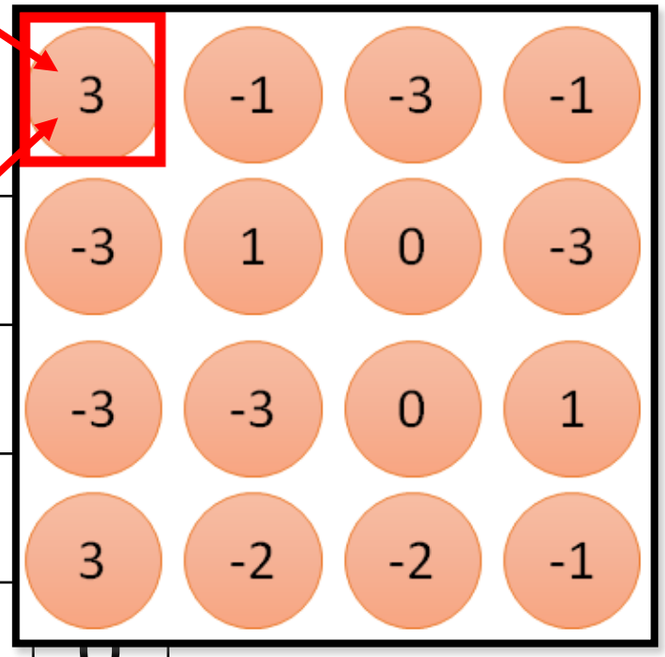
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image





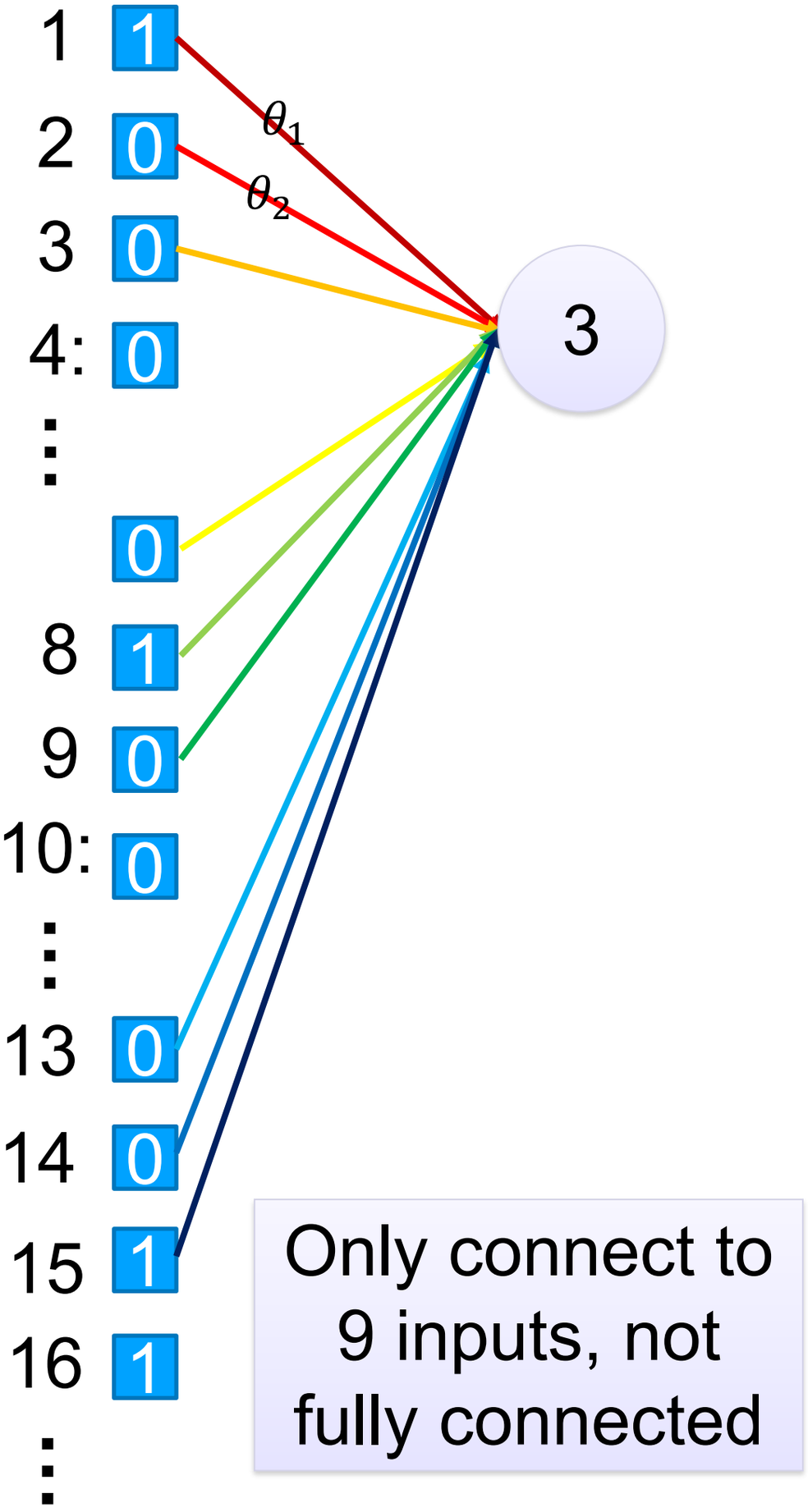
Filter 1



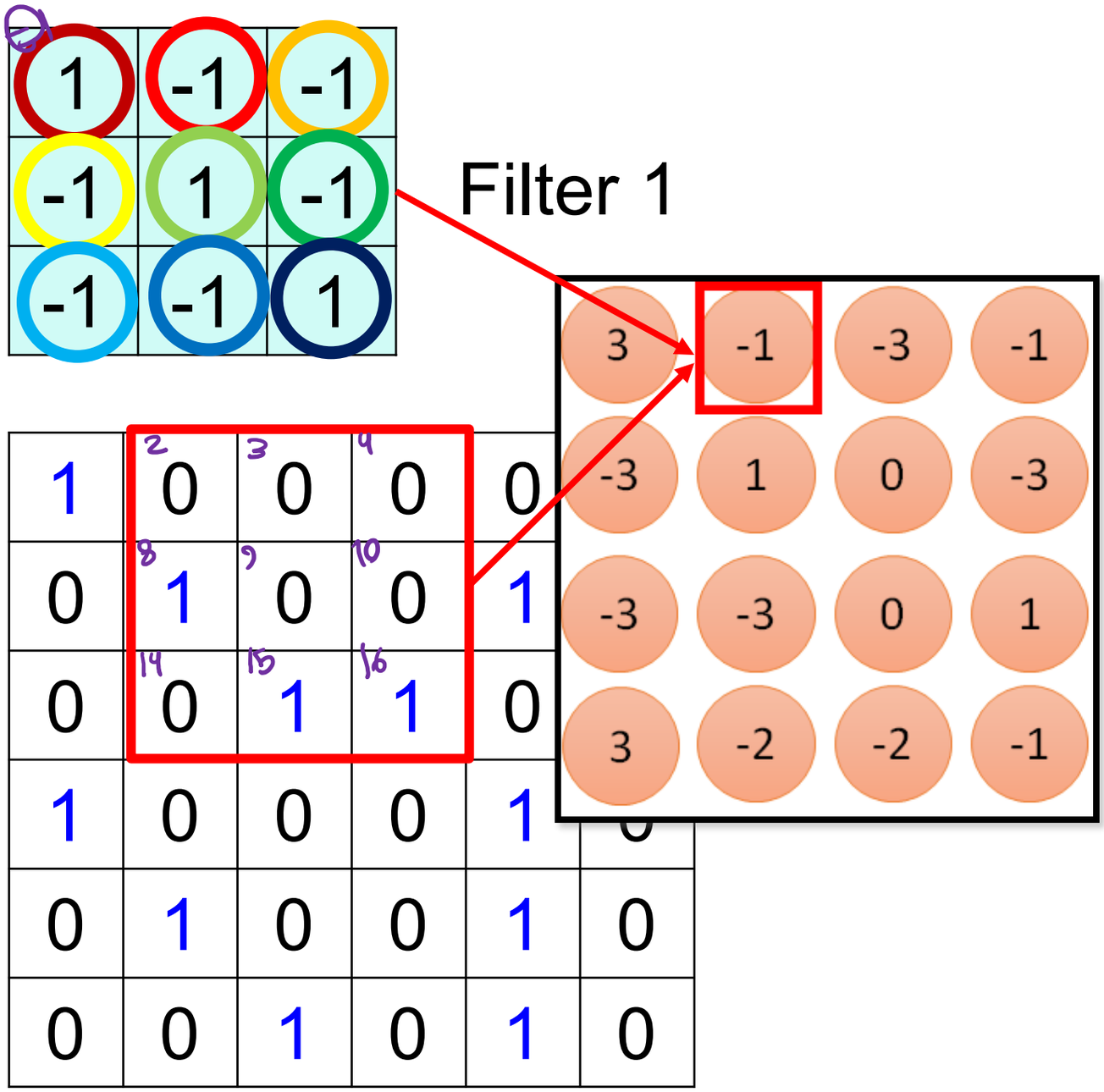
1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

fewer parameters!

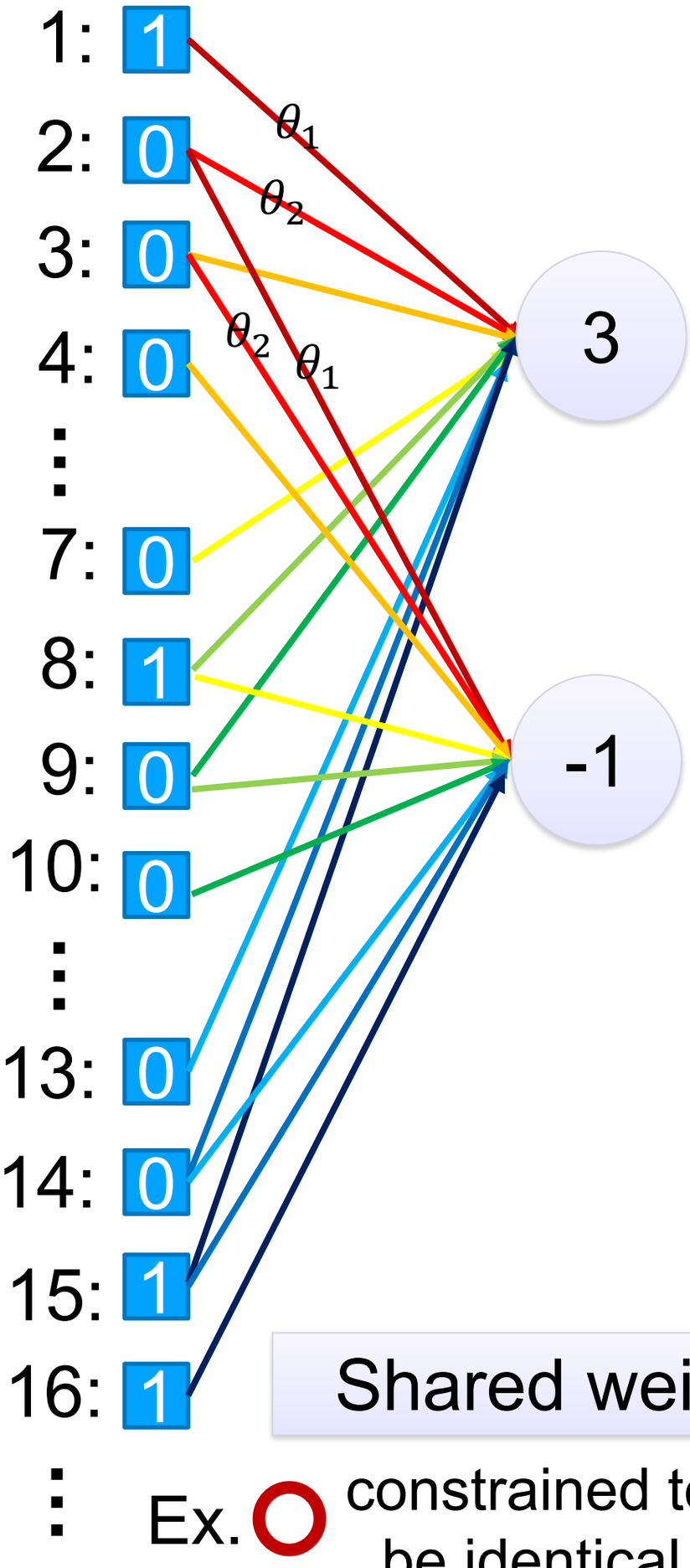


Only connect to 9 inputs, not fully connected



Fewer parameters

Even fewer parameters



The whole CNN

Convolution + ReLU

Max Pooling

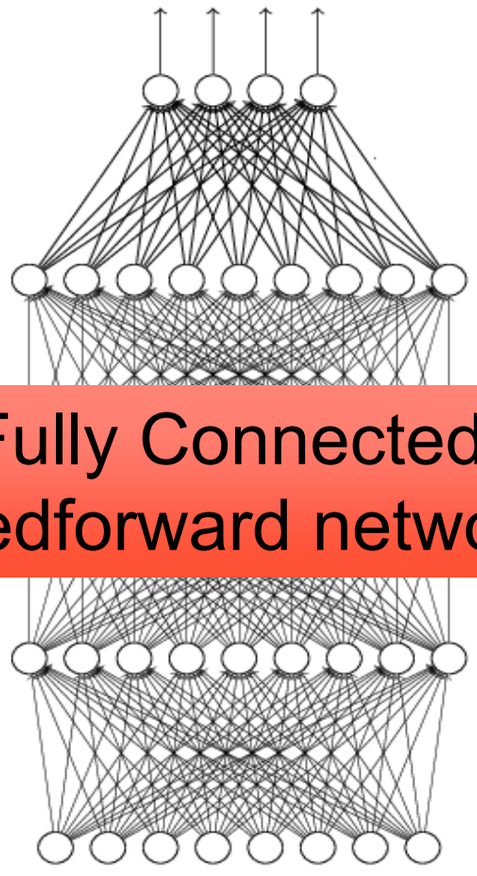
Can repeat many times

Convolution

Max Pooling

Flattened

cat dog ……



Fully Connected Feedforward network

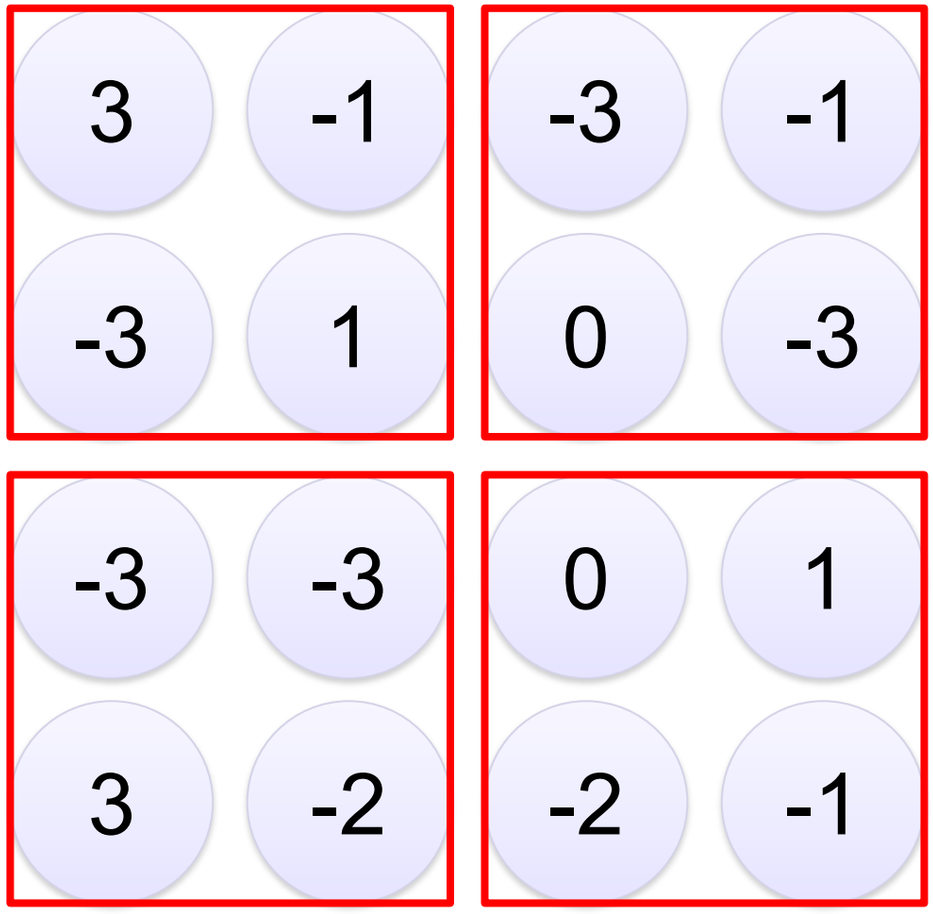
Max Pooling

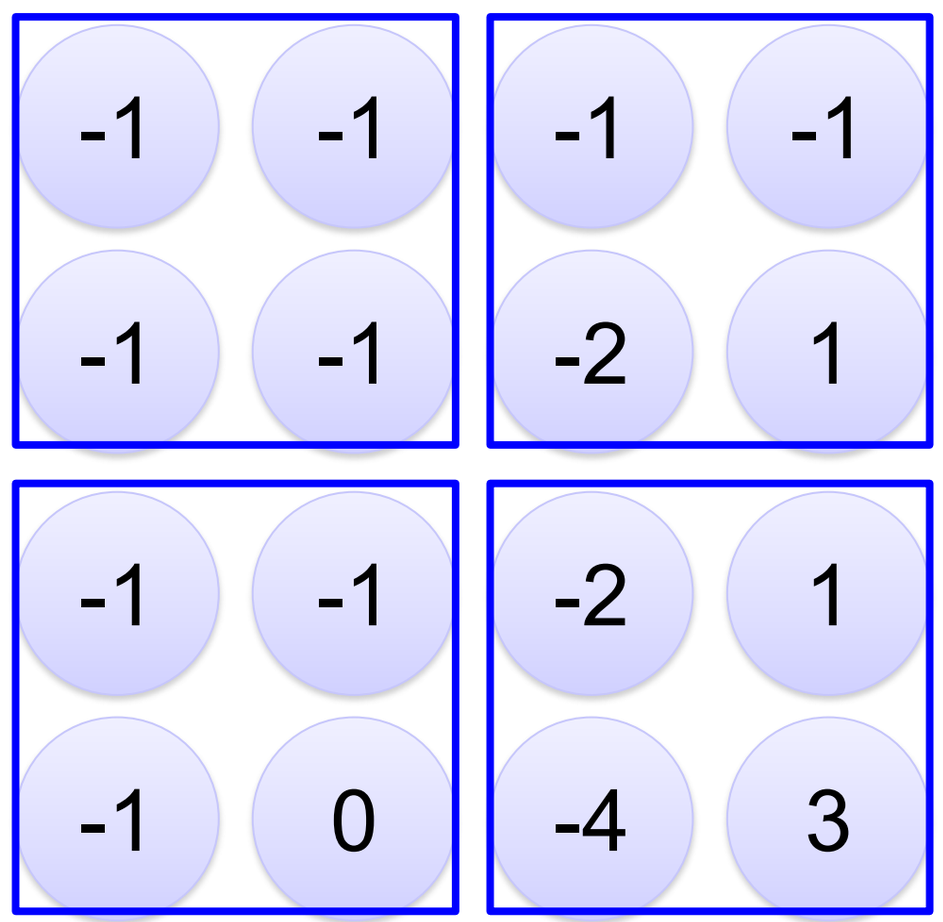
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Filter 2





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 + ReLU

2 x 4x4

Max Pooling

New image but smaller

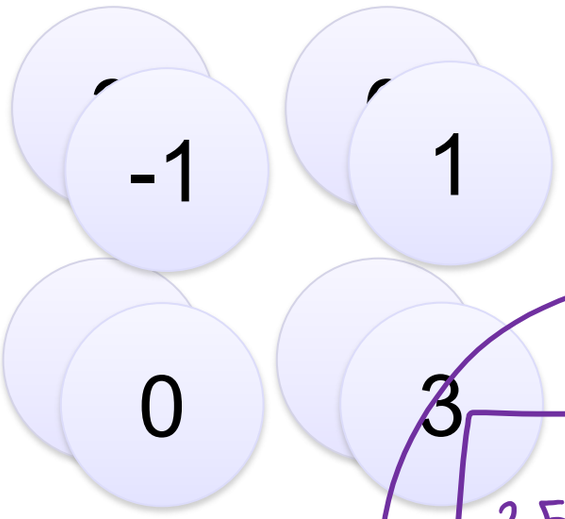
-1 1

0 3

2x 2 x 2 image

Each filter is a channel

The whole CNN



A new image

25 x 16 x 16

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

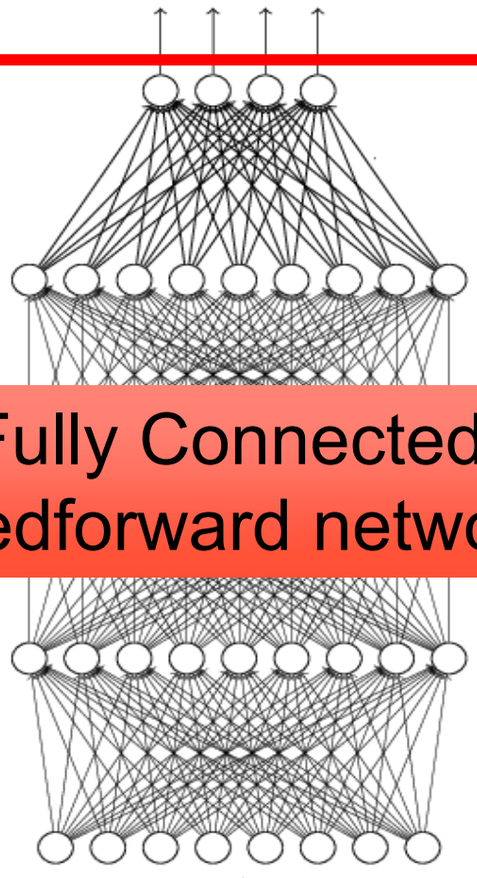
Max Pooling

5 x 10 x10

A new image

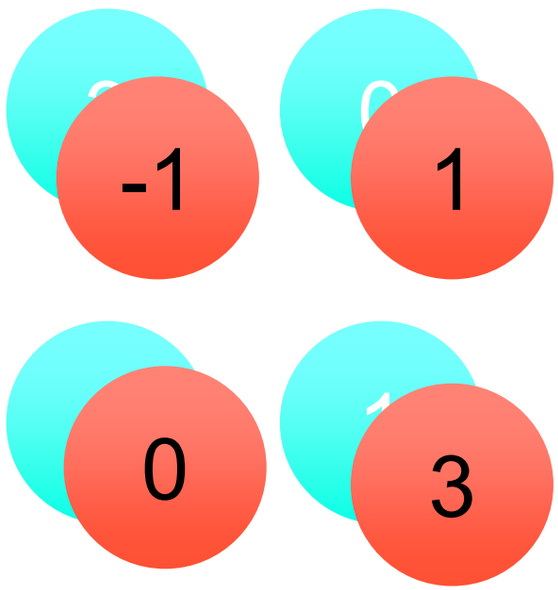
Flattened

cat dog ……

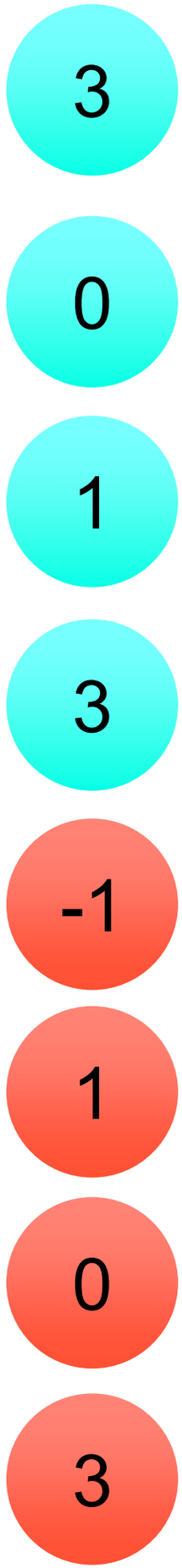


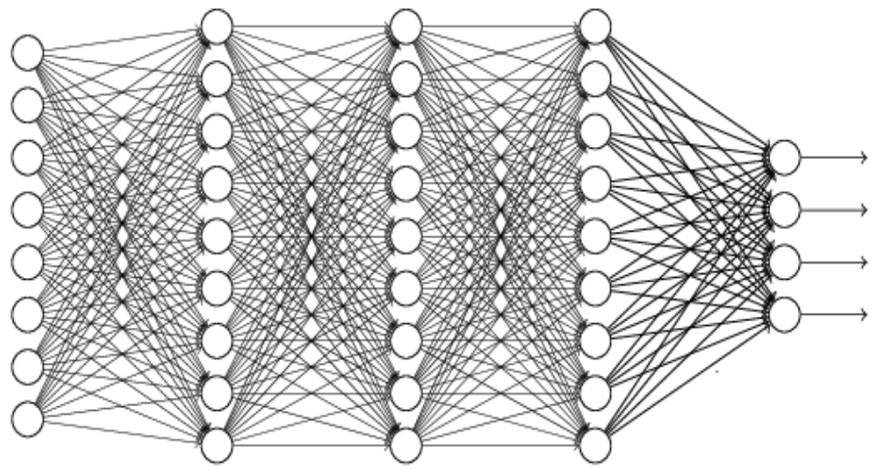
Fully Connected Feedforward network

Flattening



Flattened



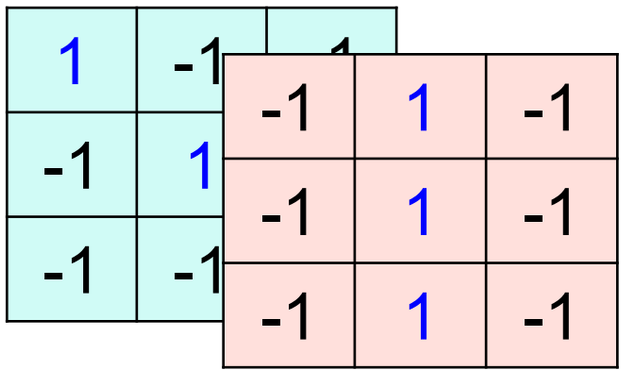


Fully Connected
Feedforward network

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)) )
```



…

…

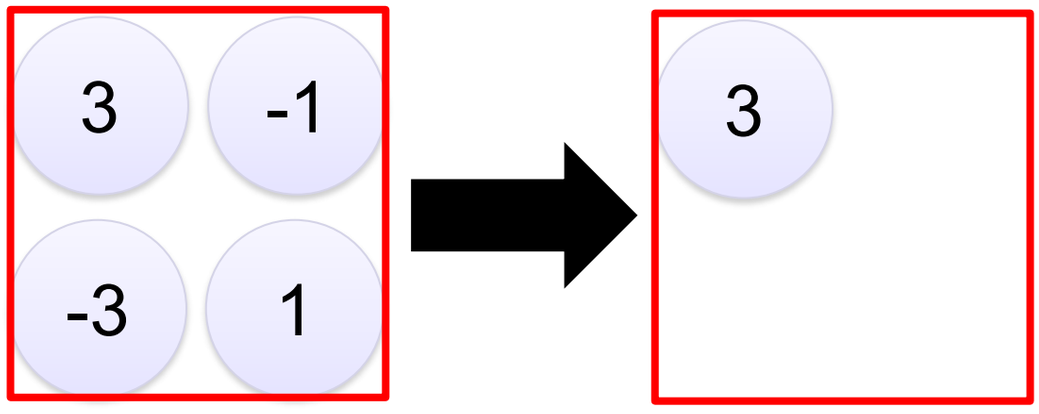
There are 25 3x3 filters.

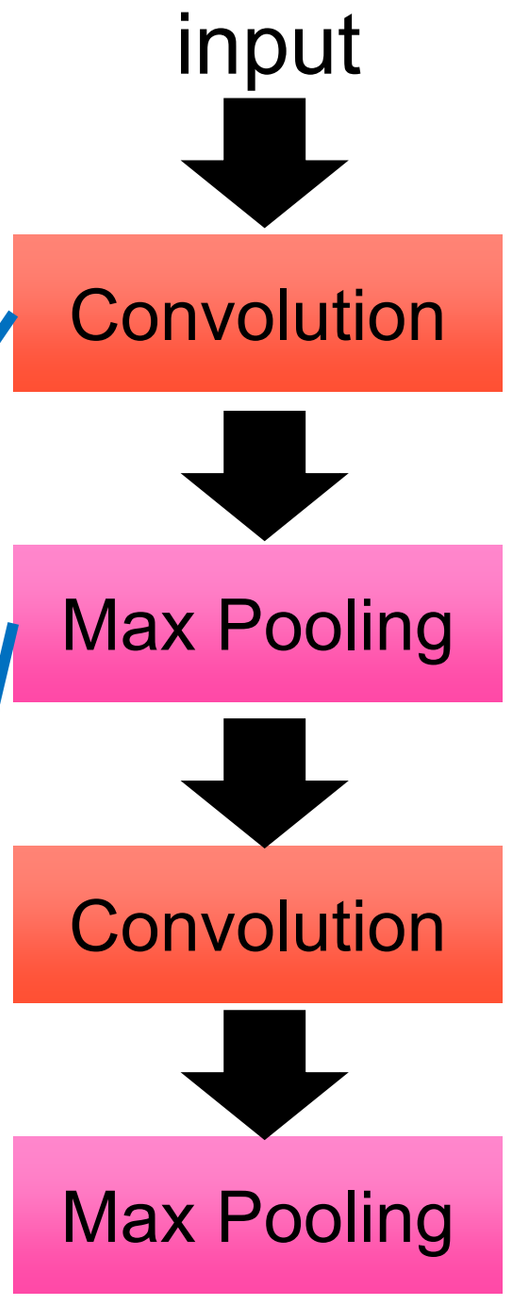
Input_shape = (28 , 28 , 1)

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

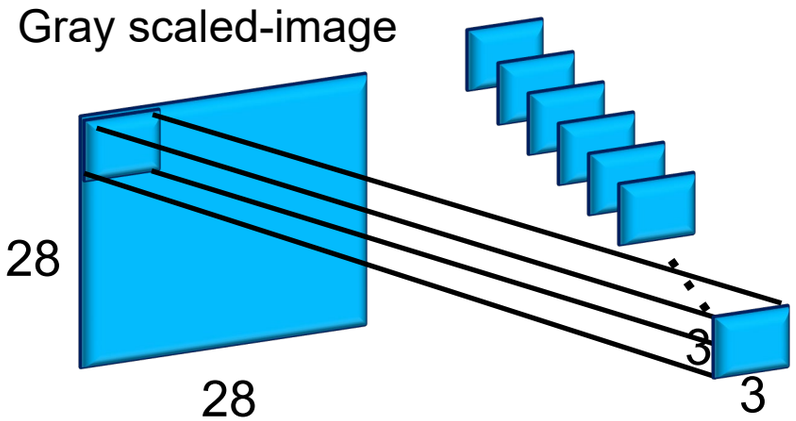
Number of Parameters

25X3X3+25 parameters

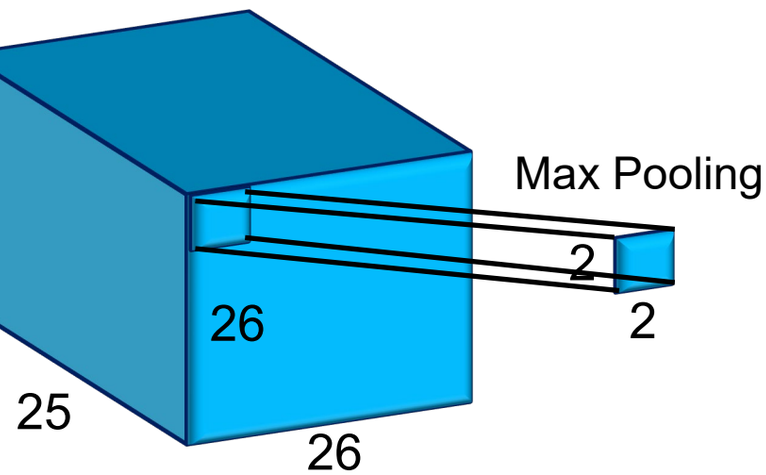
25 filters - Conv1

25: 3X3

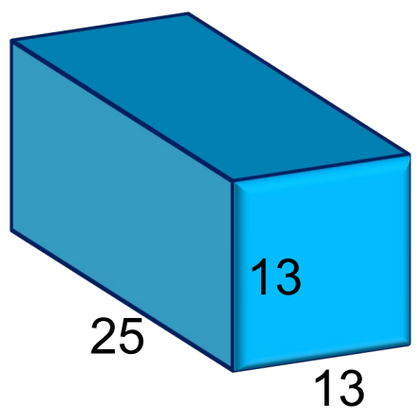
Gray scaled-image

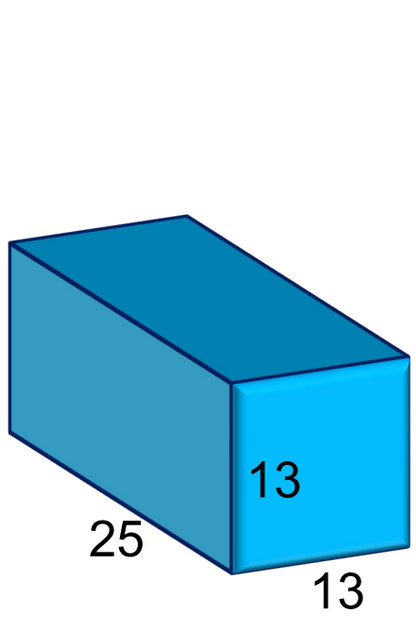


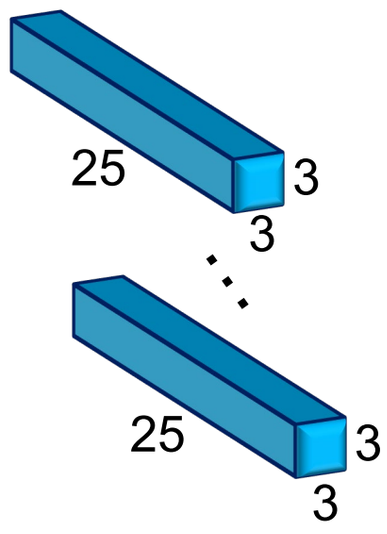
Convoluted result for 25 filters



Result after Max Pooling

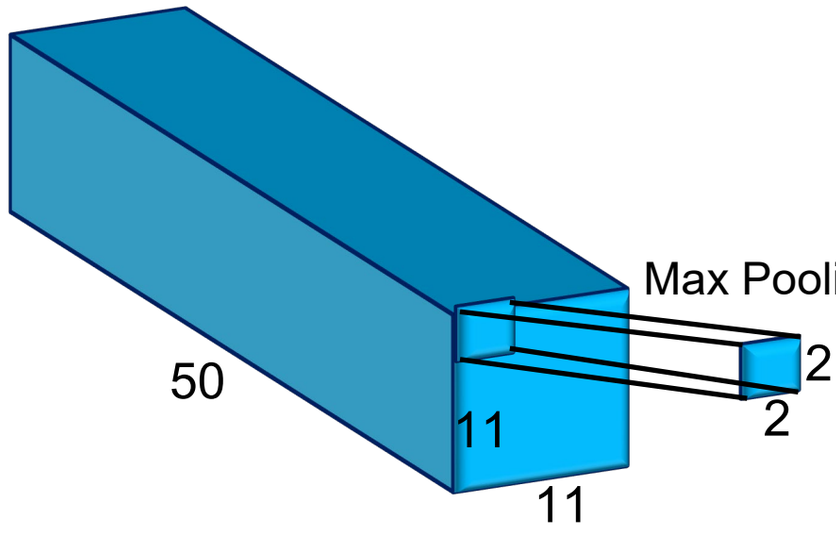






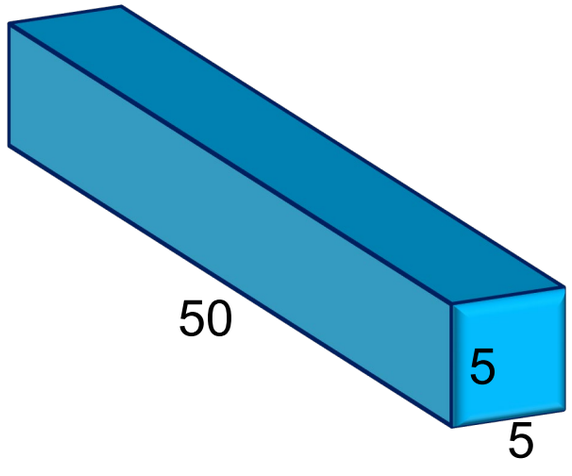
50: 3X3X25

50 filters – Conv2

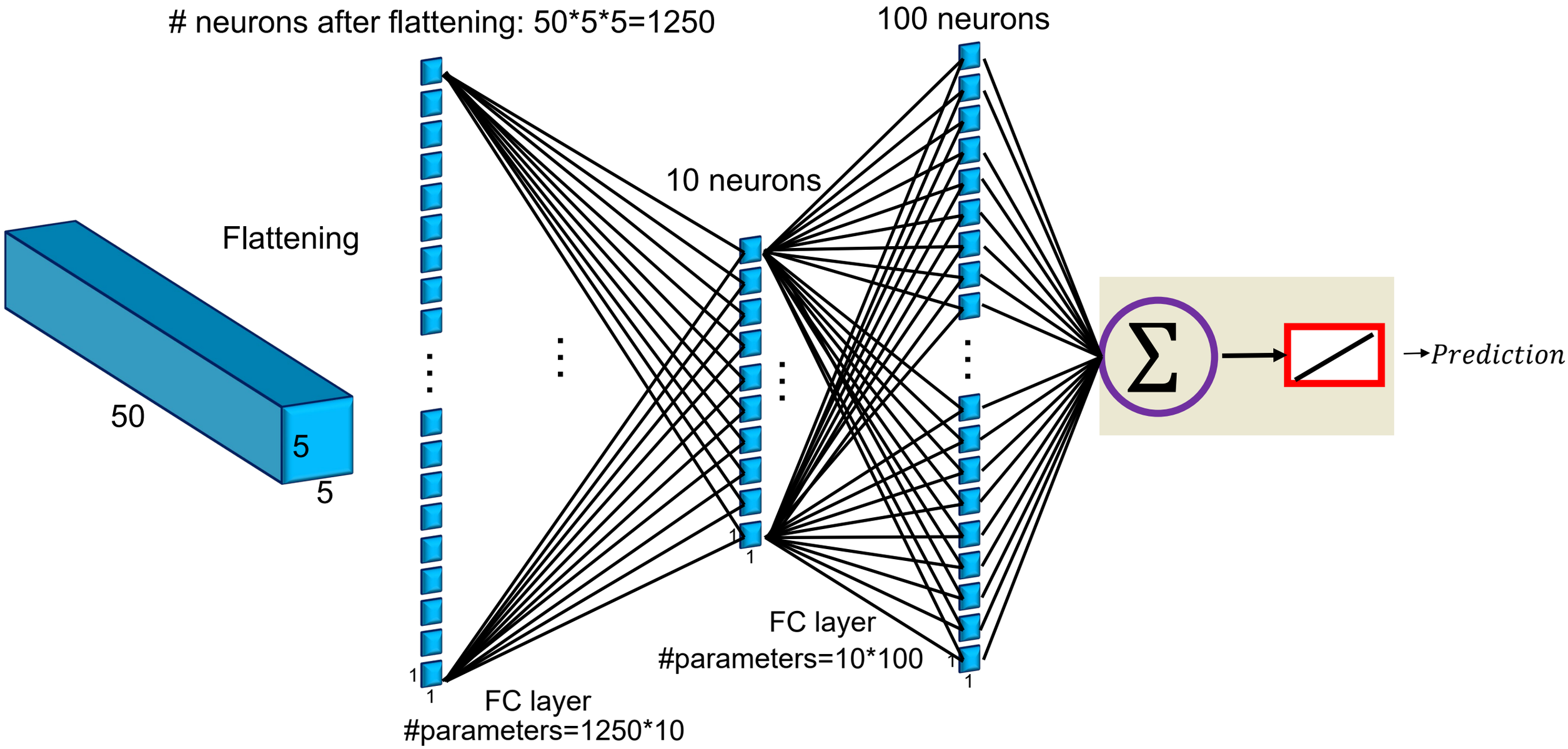


Convoluted result for 50 filters

Result after Max Pooling



50X3X3X25+50 parameters



[10 CNN Architecture]