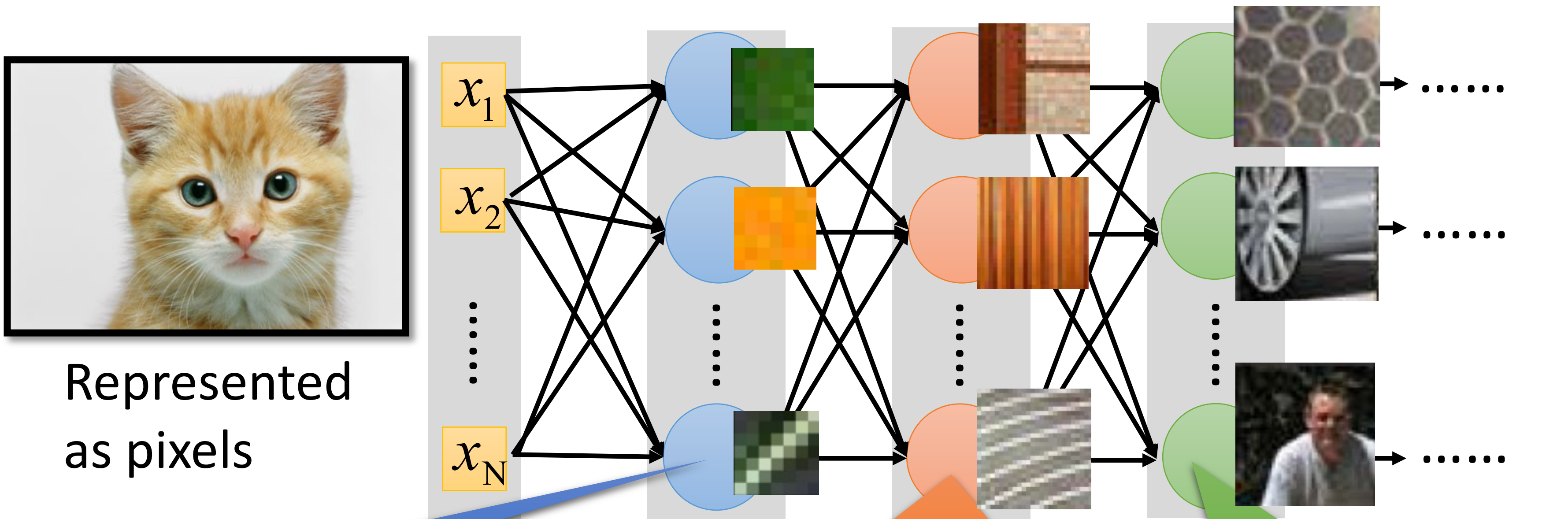
Convolutional Neural Network

So far

- Perceptron
- Fully connected layer (dense layer)
  - Every input neuron is connected to every output neuron.
  - It multiplies the input by a weight matrix and then adds a bias vector.
- Multi-layer Perceptron (MLP)
  - Forward pass
  - Activation functions
  - Backpropagation

Why CNN for Image?

[Zeiler, M. D., *ECCV 2014*]



Represented as pixels

The most basic classifiers

Use 1st layer as module to build classifiers

Use 2nd layer as module ……

Can the network be simplified by considering the properties of images?

Why CNN for Image

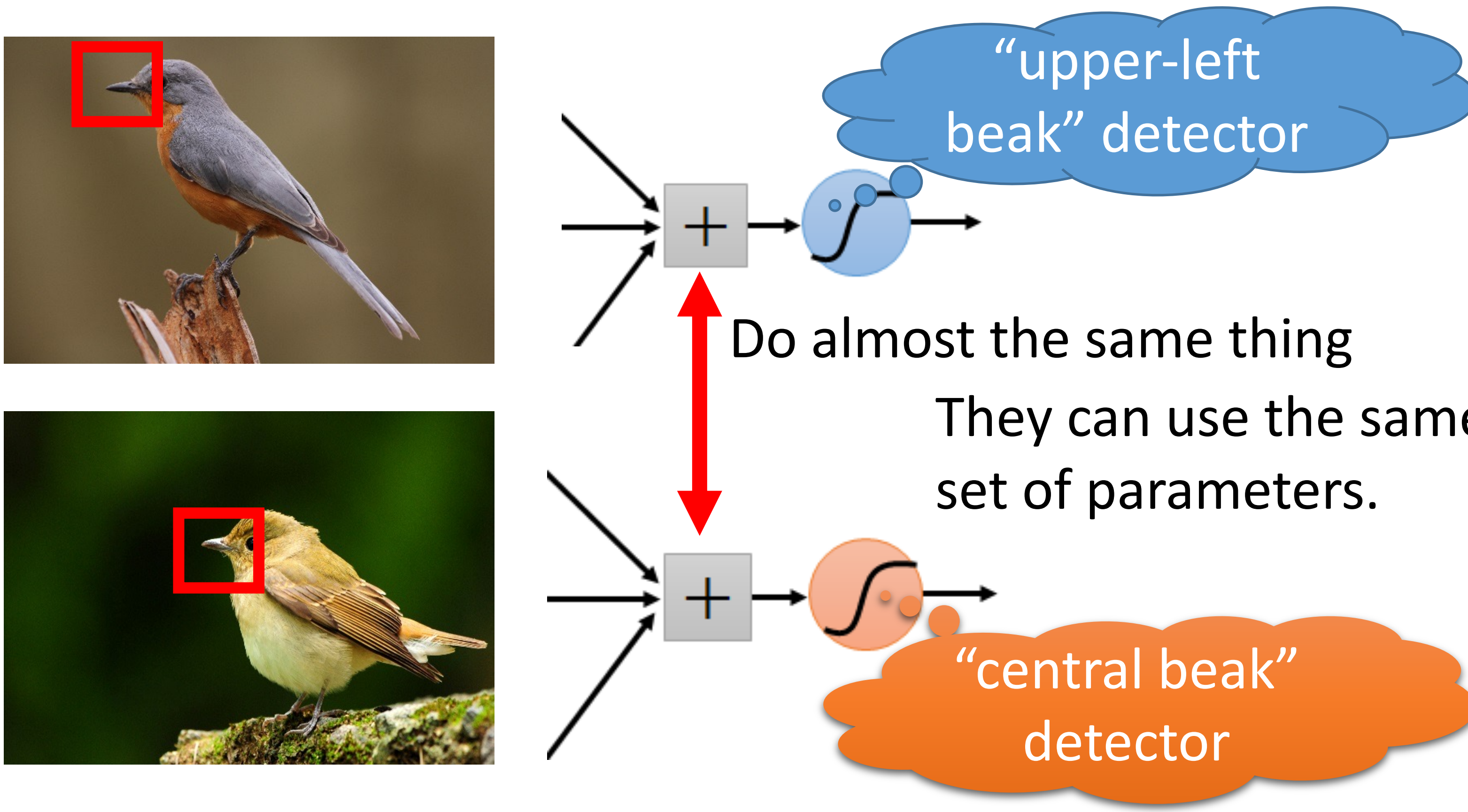
- Some patterns are much smaller than the whole image

A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

Why CNN for Image

- The same patterns appear in different regions.



Why CNN for Image

- Subsampling the pixels will not change the object

bird

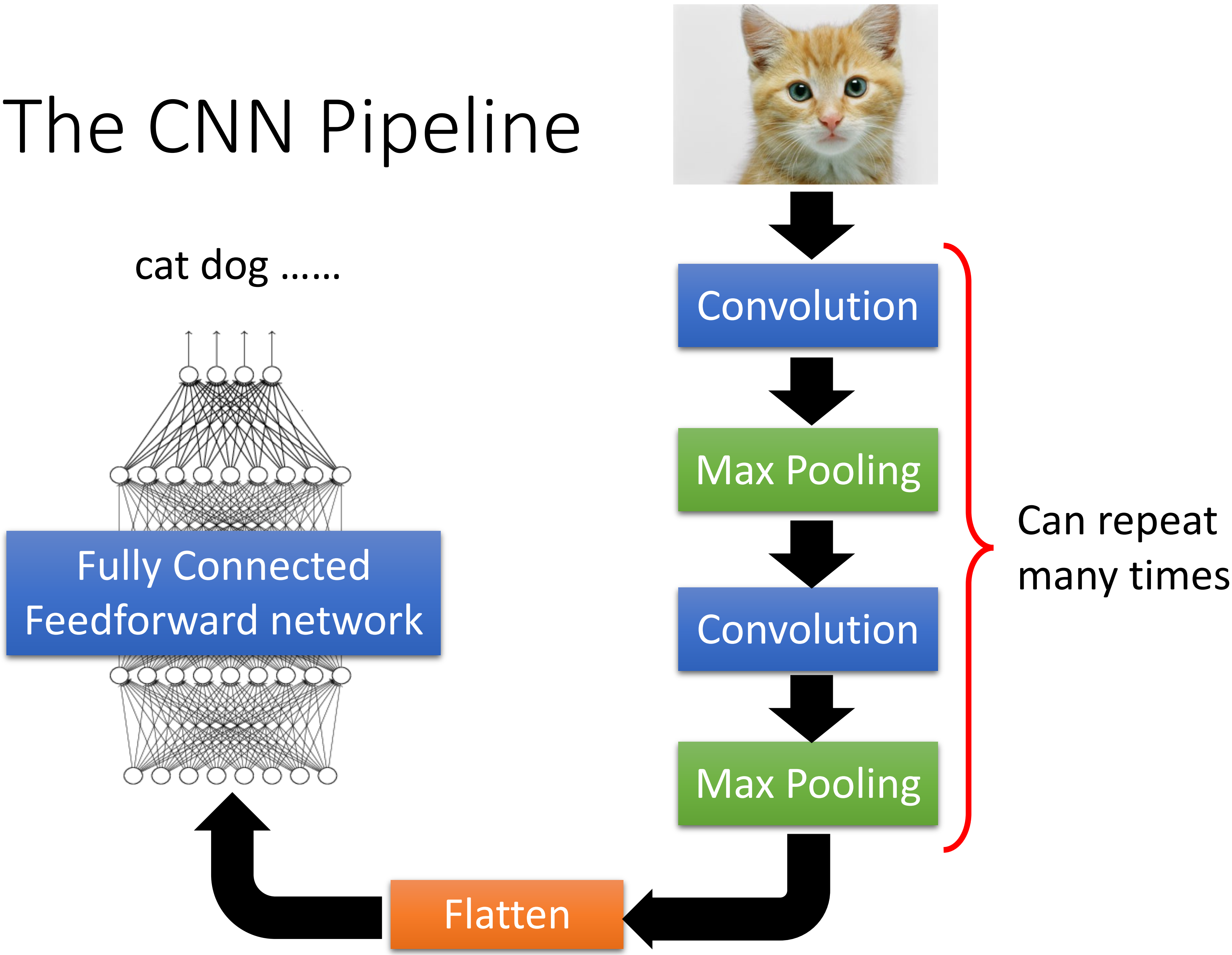
subsampling

bird

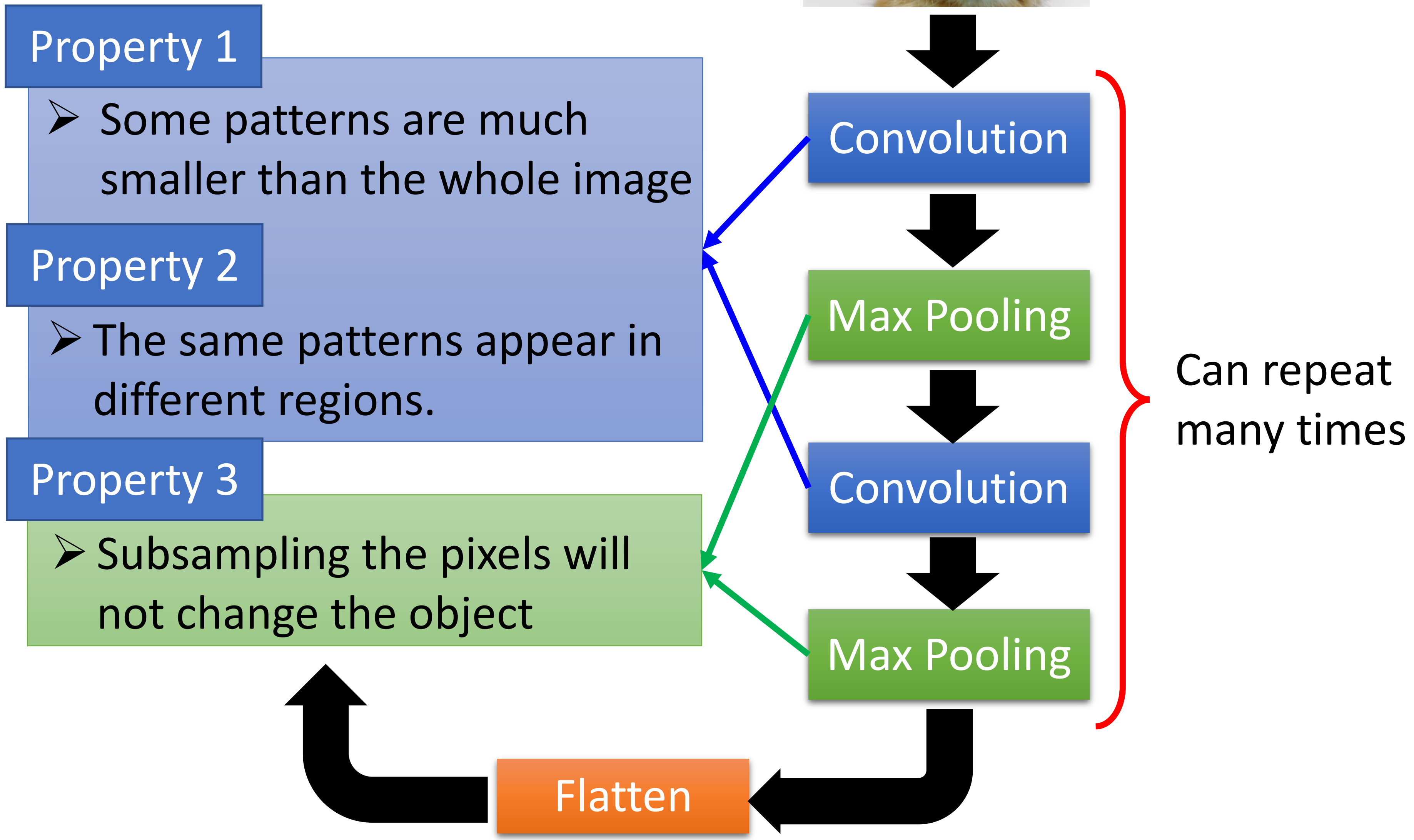
We can subsample the pixels to make image smaller

Less parameters for the network to process the image

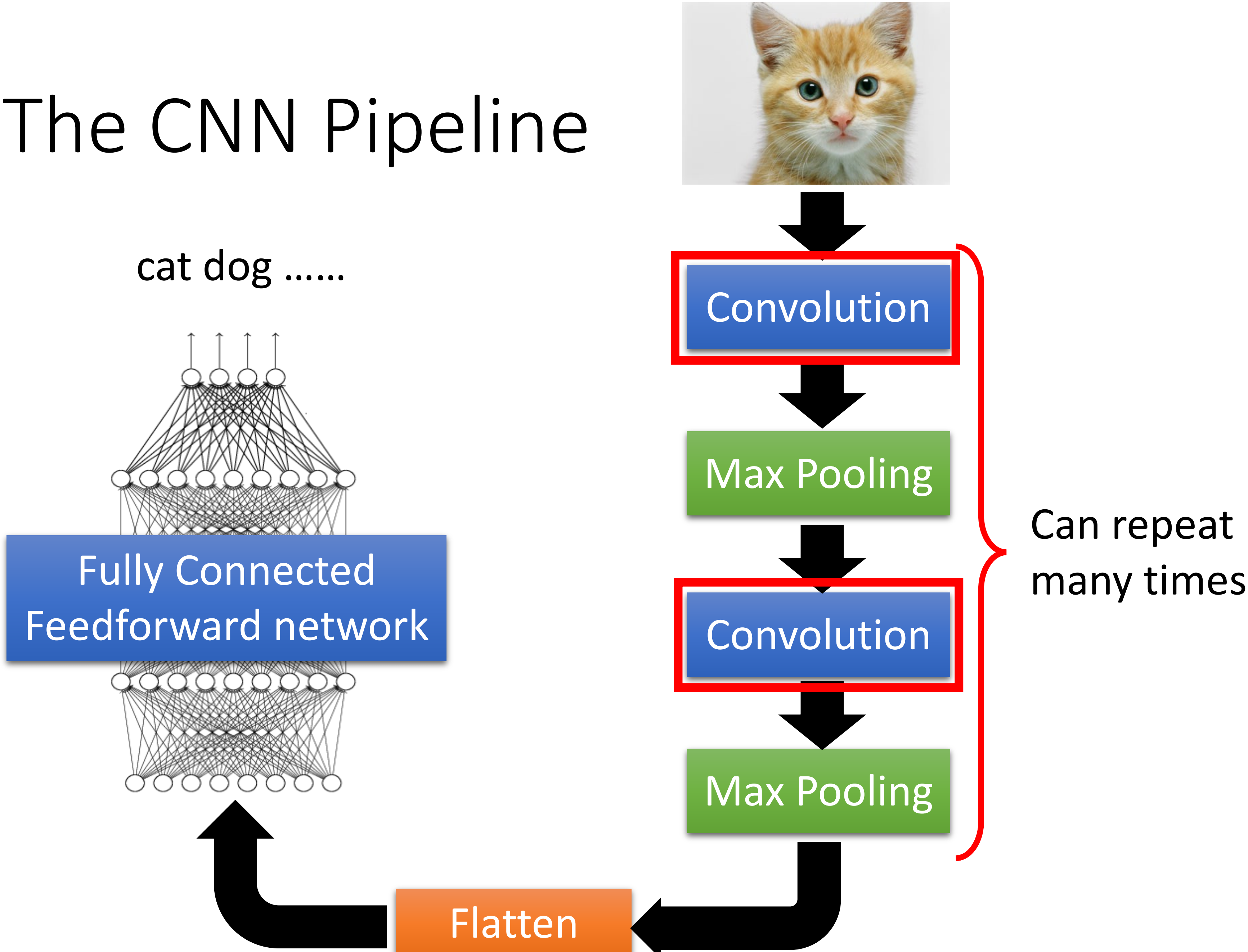
The CNN Pipeline



The CNN Pipeline



The CNN Pipeline



CNN – Convolution

Those 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

Matrix

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

Filter 2

Matrix

⋮

Property 1

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

CNN – 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

CNN – 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

We set stride=1 below

CNN – 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

Property 2

CNN – Convolution

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

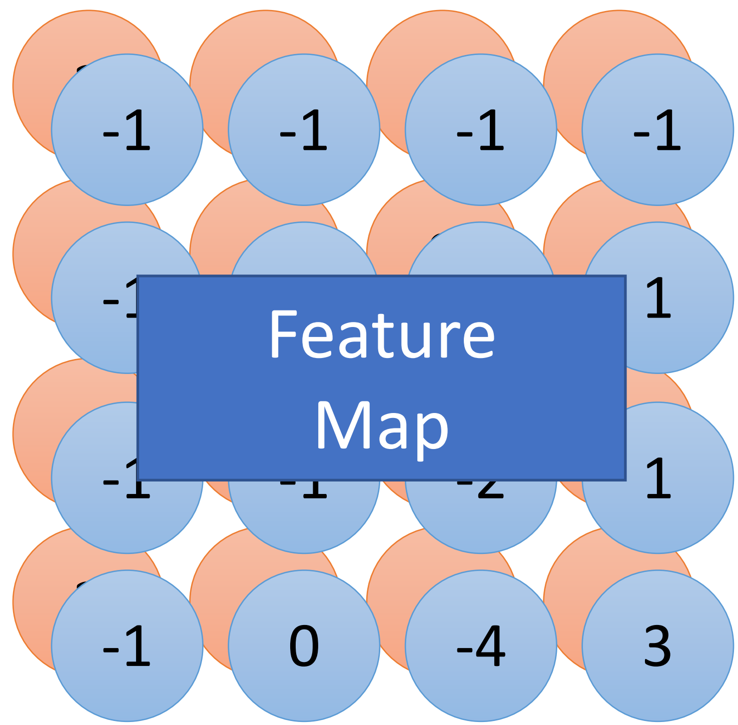
Filter 2

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

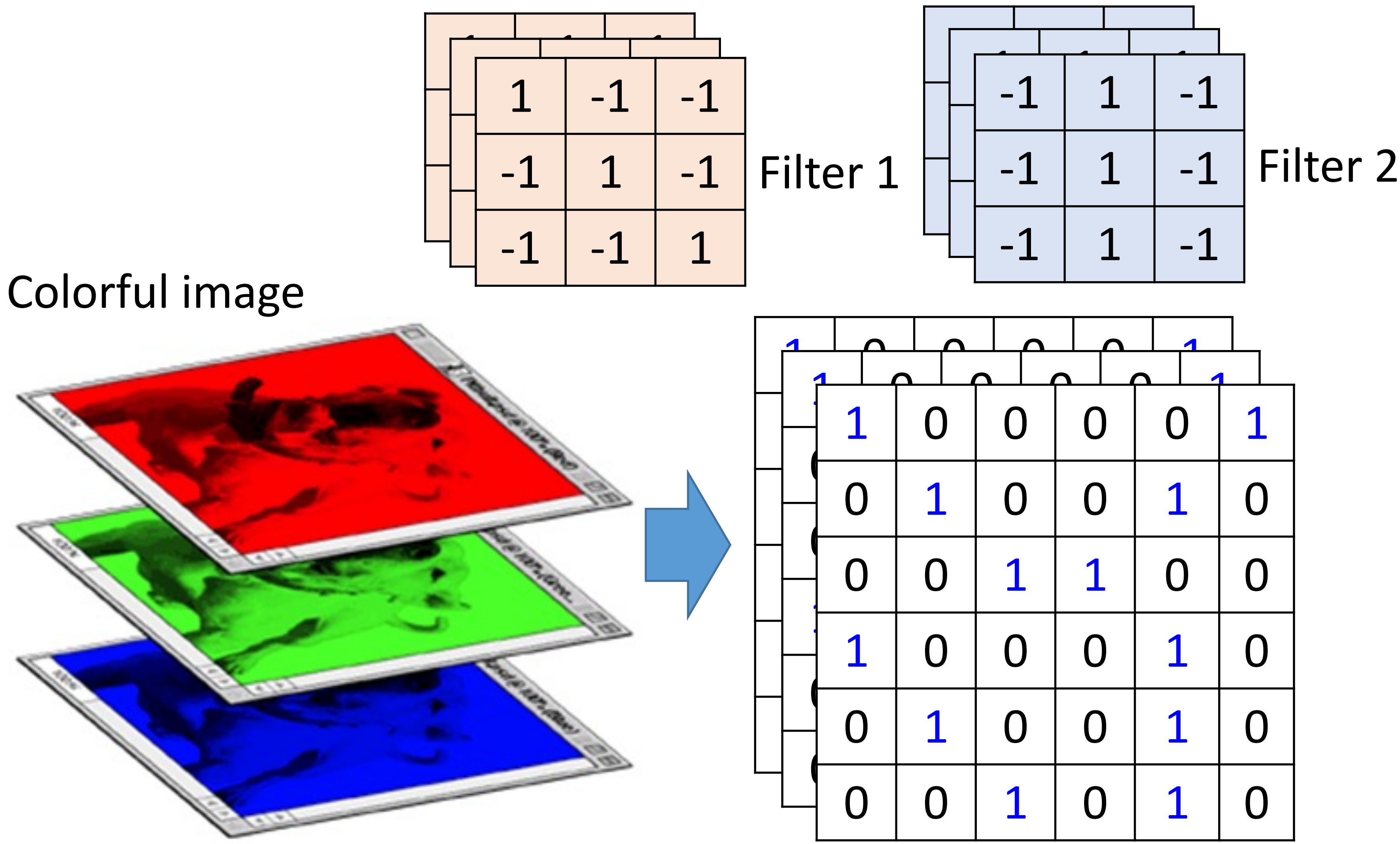
6 x 6 image

Do the same process for every filter

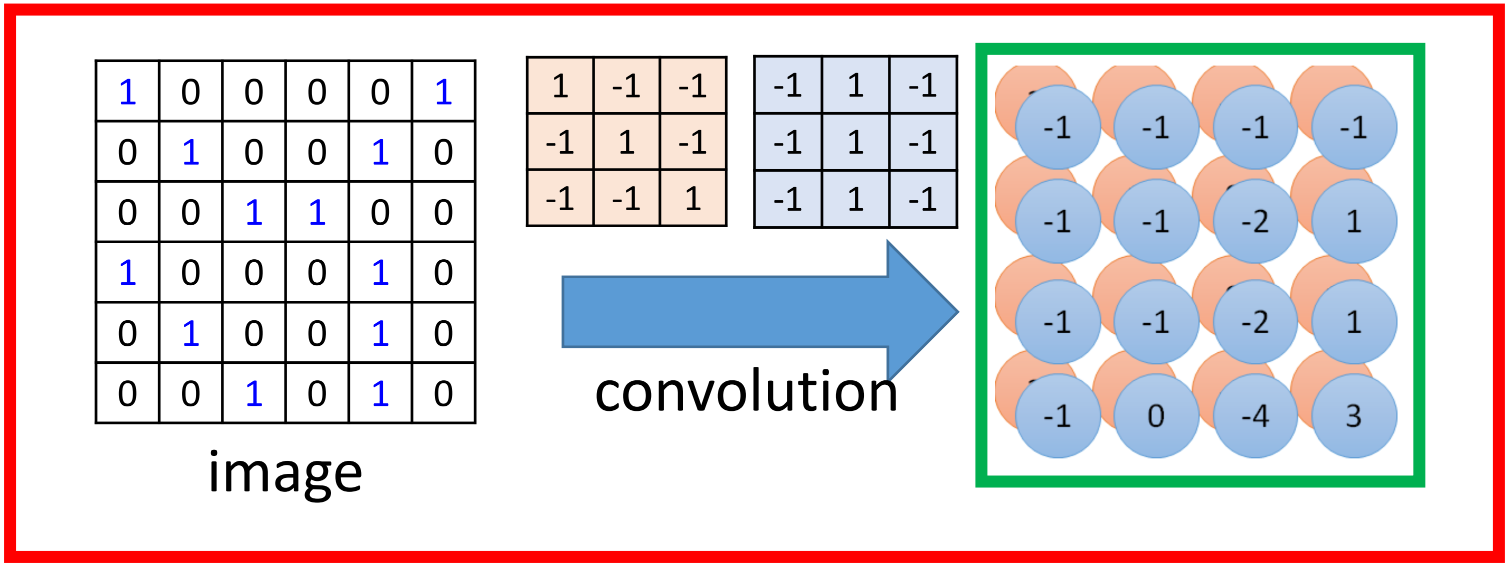


4 x 4 image

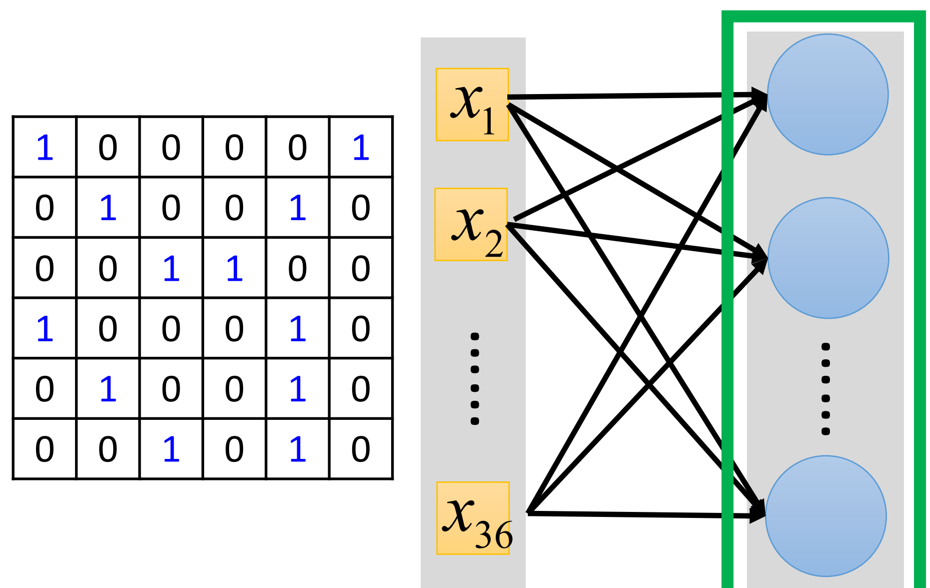
CNN – Colorful image

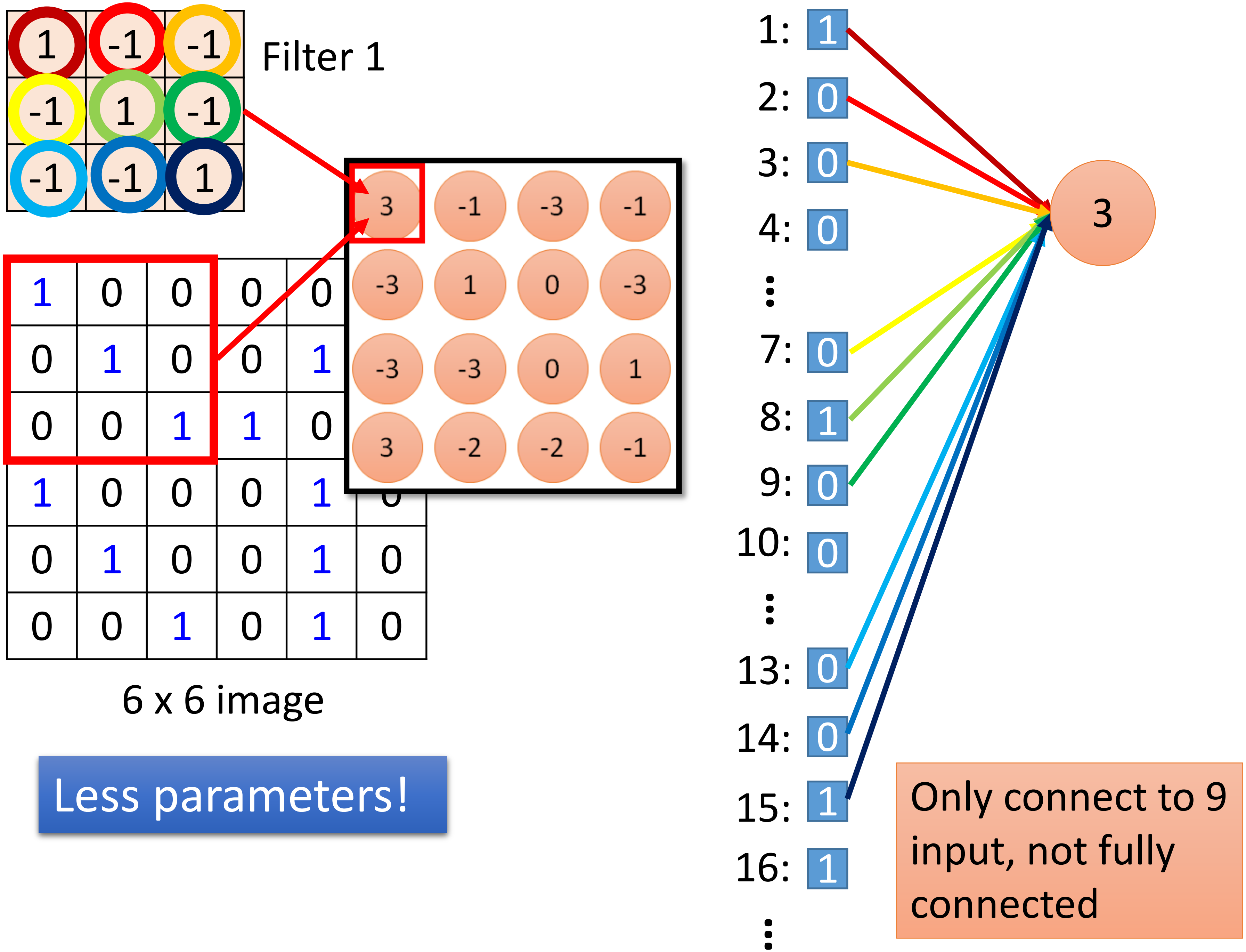


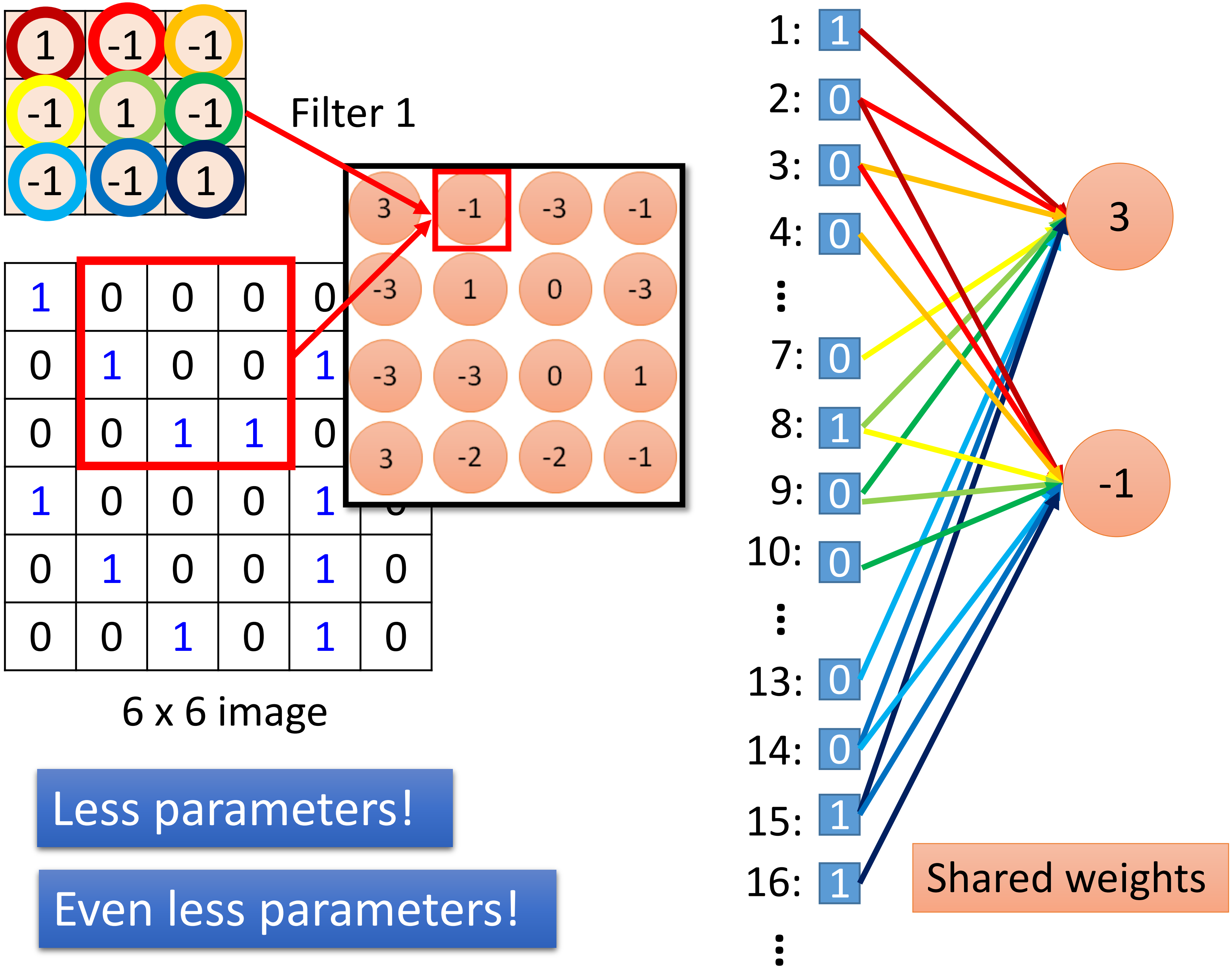
Convolution v.s. Fully Connected



Fully-connected

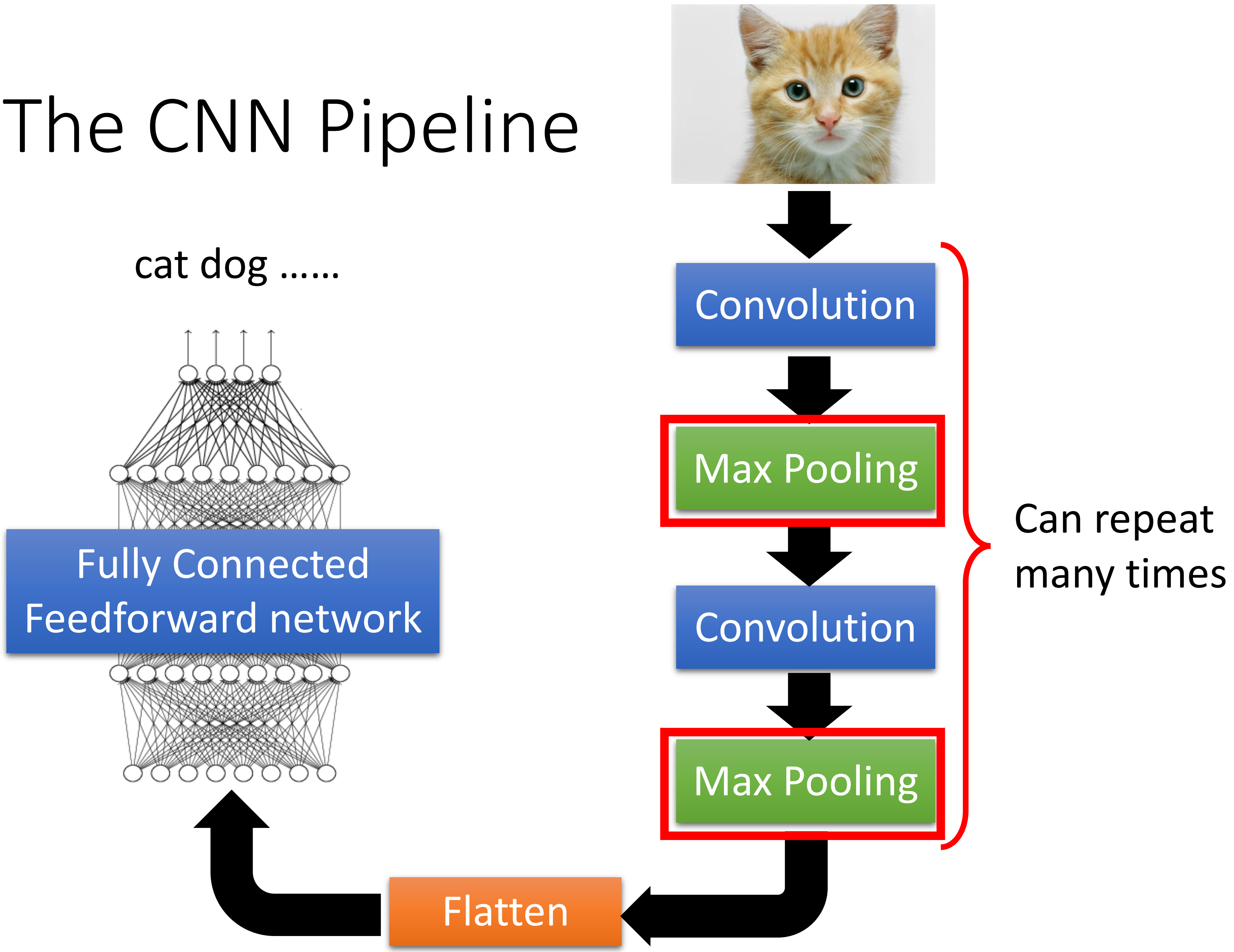



Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Less parameters!
Only connect to 9 input, not fully connected



Filter 1
6 x 6 image
Less parameters!
Even less parameters!
Shared weights

The CNN Pipeline



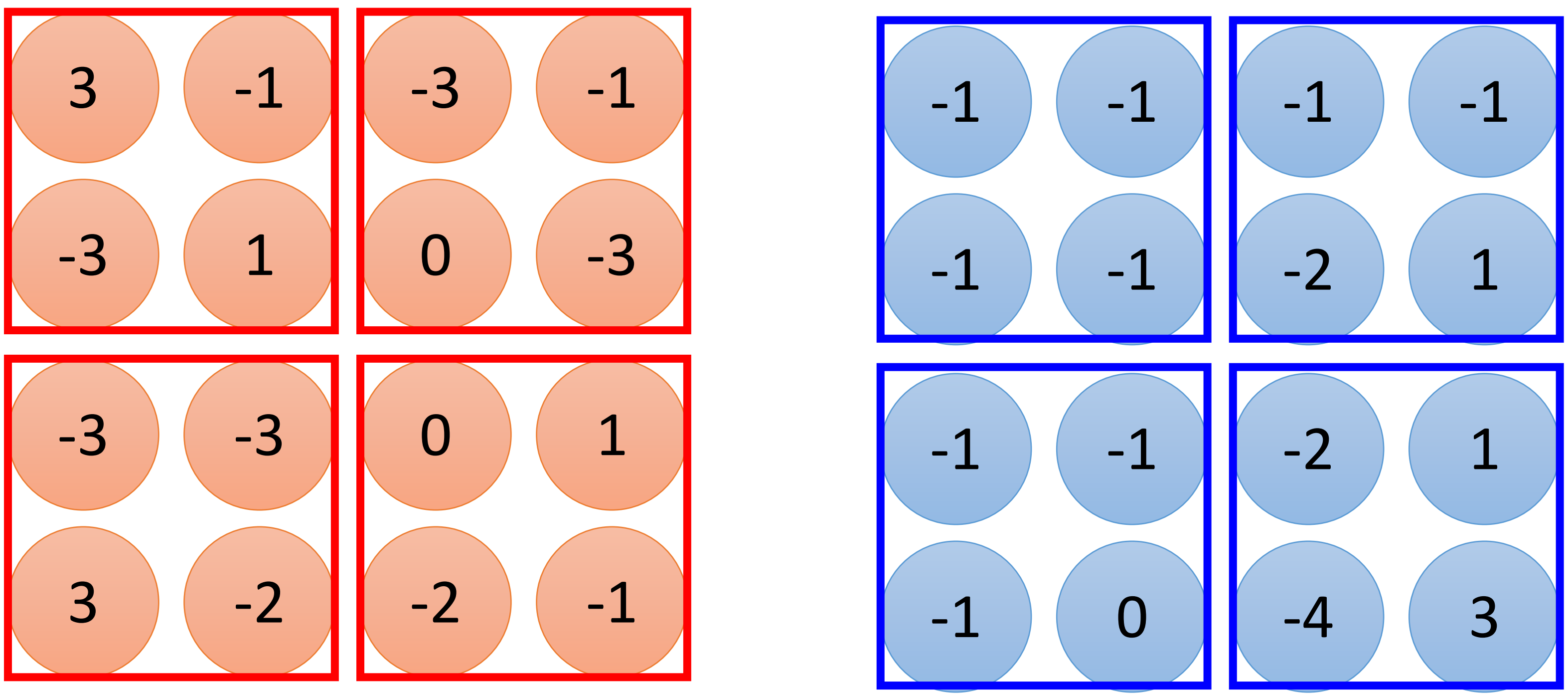
CNN – Max Pooling

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

Filter 1

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

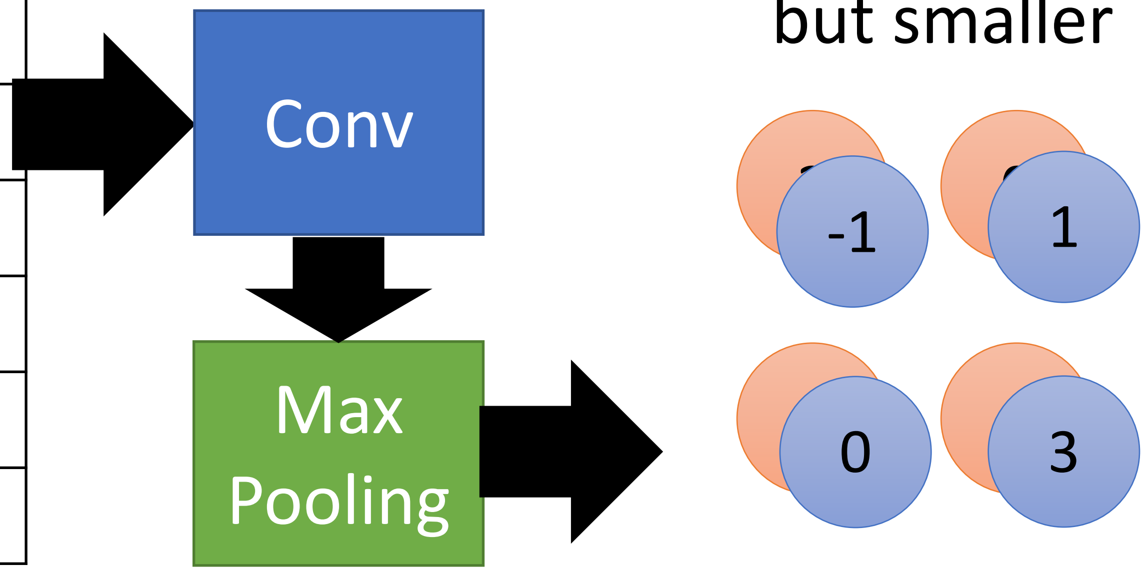
Filter 2



CNN – 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

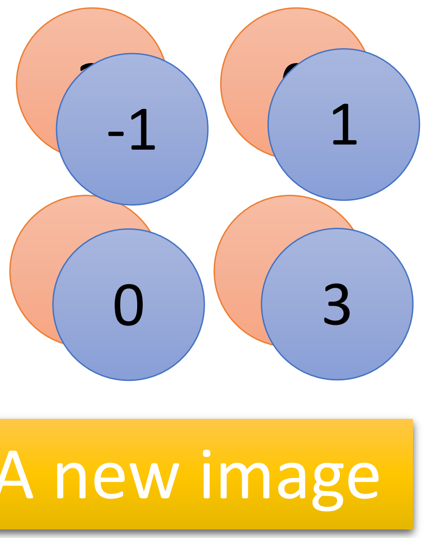


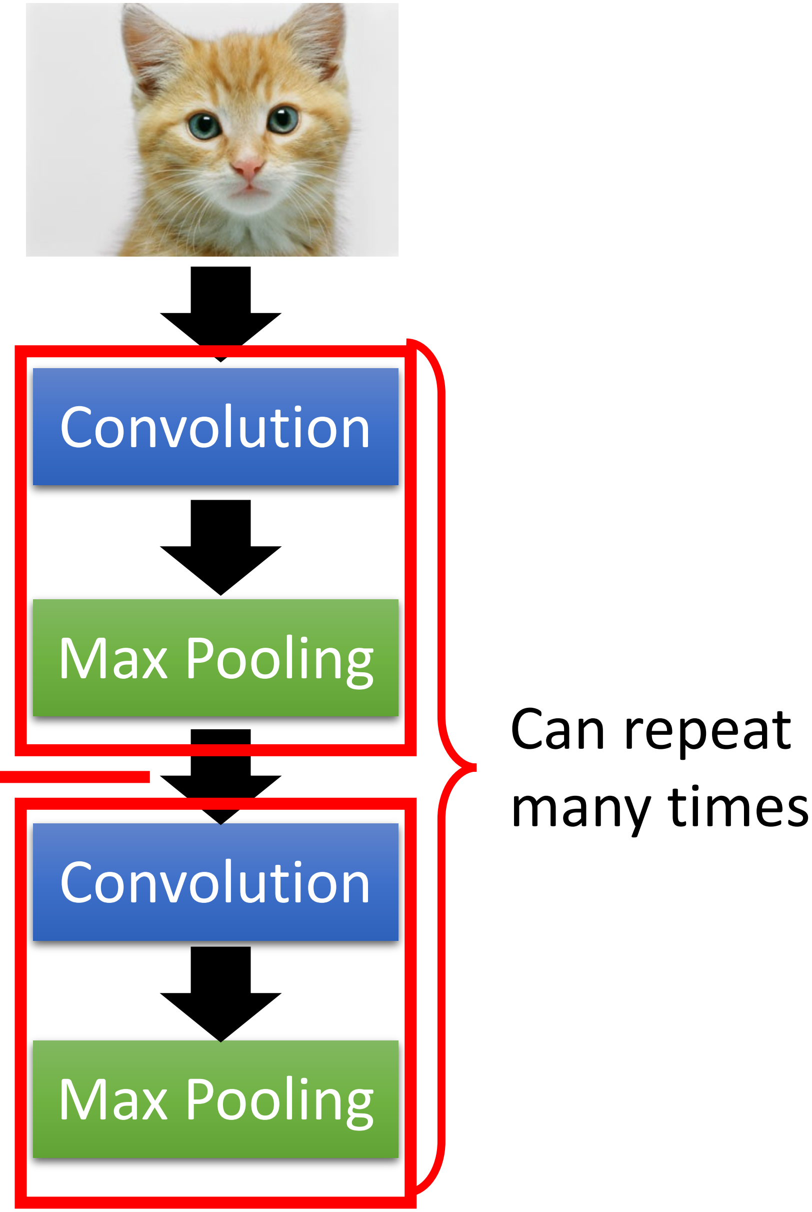
New image but smaller

2 x 2 image

Each filter is a channel

The CNN Pipeline

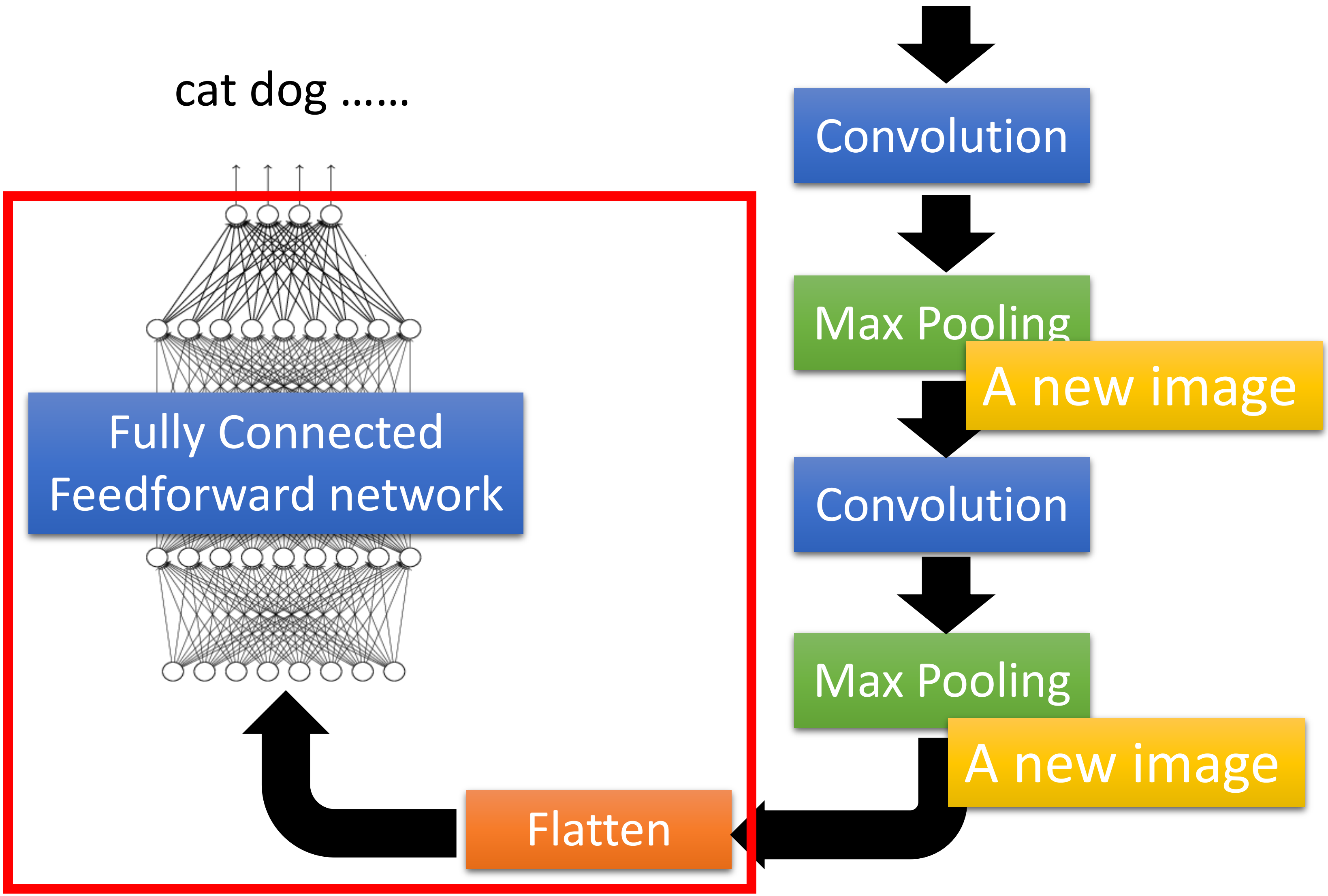




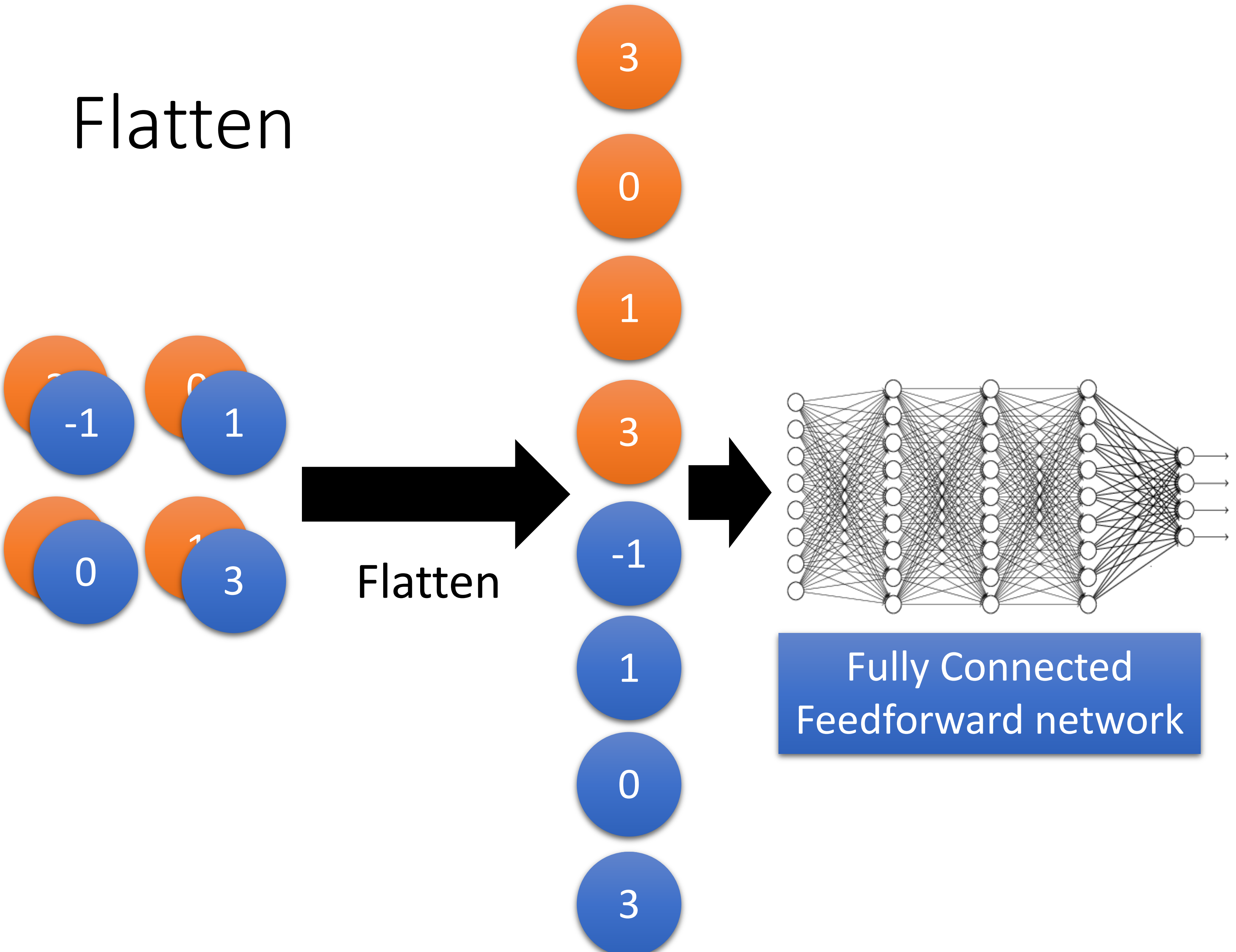
Smaller than the original image

The number of the channels
= the number of filters

The whole CNN



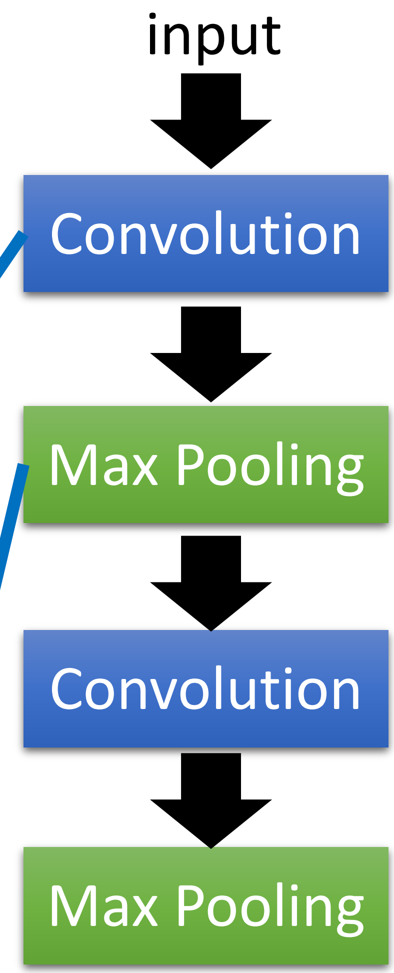
Flatten



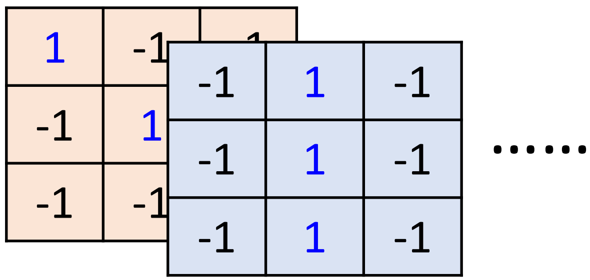
-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

CNN

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




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



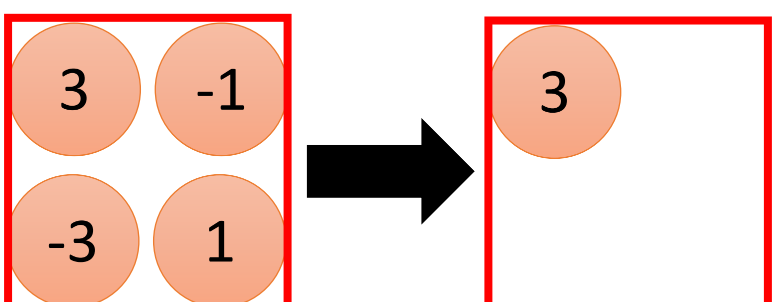
There are 25 3x3 filters.

Input_shape = (1 , 28 , 28)

1: black/weight, 3: RGB

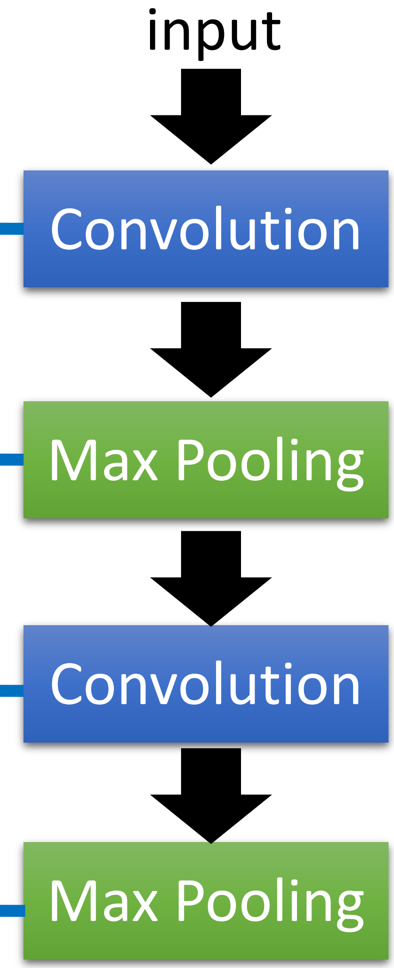
28 x 28 pixels

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



CNN

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



1 x 28 x 28

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

How many parameters for each filter? 9

25 x 26 x 26

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

25 x 13 x 13

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

How many parameters for each filter? 225

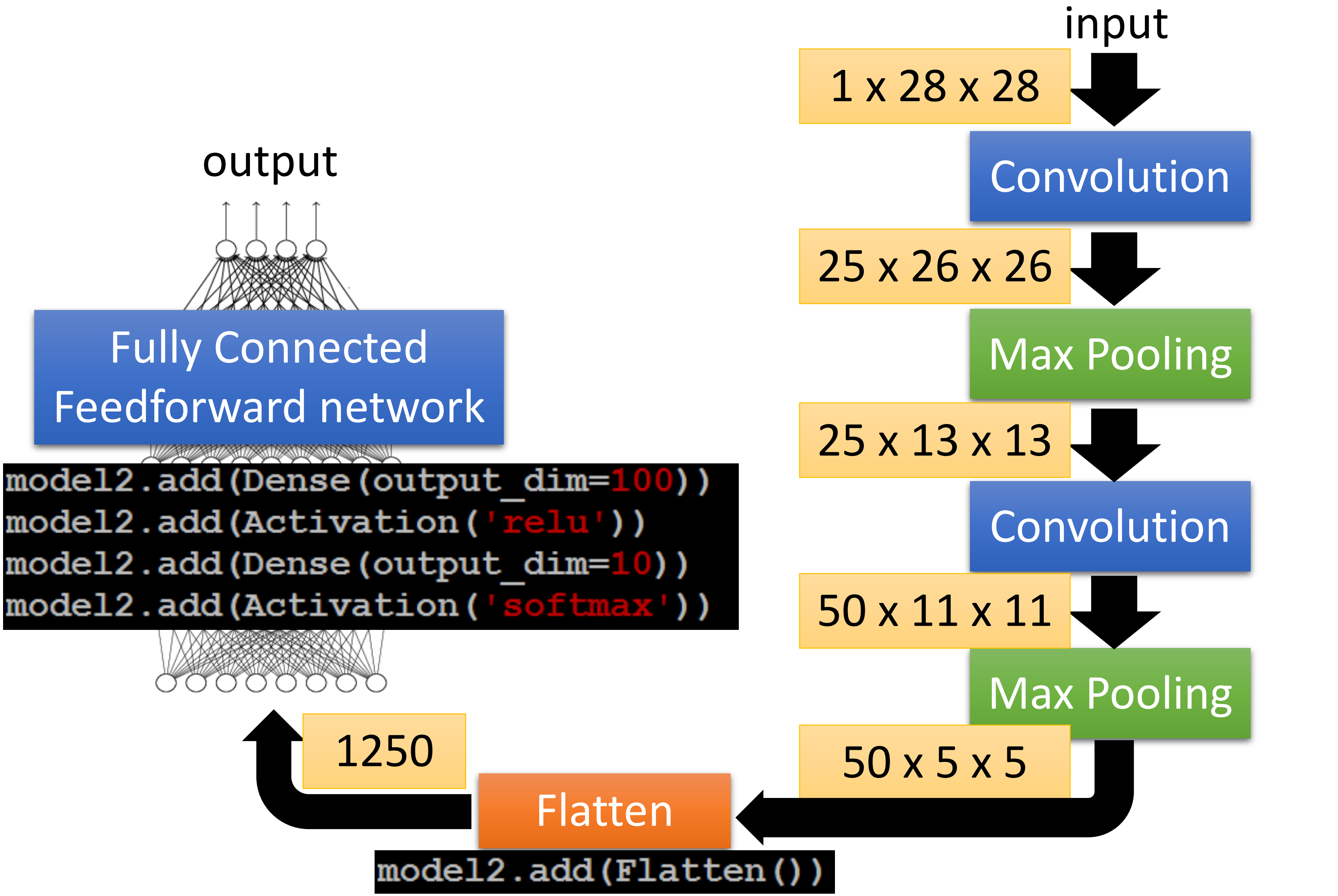
50 x 11 x 11

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

50 x 5 x 5

CNN

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



Formulas

- Number of total parameters in a Conv layer:
- $(C_{in} \times k \times k + 1) \times C_{out}$
    - C_{in}: number of input channels (e.g., RGB=3)
    - C_{out}: number of output channels (filters)
    - k: size of the filter
    - +1: accounts for one bias per output channel (filter)

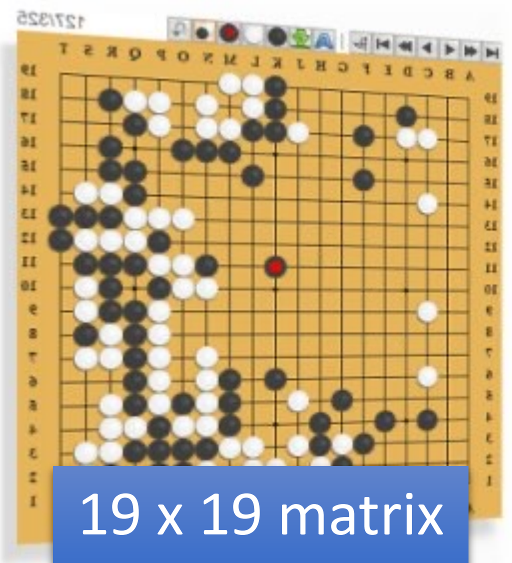
Output dimensions: Conv Layer

- $H_{out} = \lfloor \frac{H_{in} - k + 2P}{S} \rfloor + 1$
- $W_{out} = \lfloor \frac{W_{in} - k + 2P}{S} \rfloor + 1$
- C_{out}= number of filters used
    - k: kernel size
    - P: size of padding
    - S: Stride
    - $\lfloor \cdot \rfloor$ denotes floor (rounding down to nearest integer)

Output dimensions: Pooling Layer

- There is no padding in the pooling layer, hence
- $H_{out} = \lfloor \frac{H_{in} - k}{S} \rfloor + 1$
- $W_{out} = \lfloor \frac{W_{in} - k}{S} \rfloor + 1$
- $C_{in} = C_{out}$

More Application: Playing Go



19 x 19 matrix (image)

Black: 1

white: -1

none: 0

Network

Next move (19 x 19 positions)

19 x 19 vector

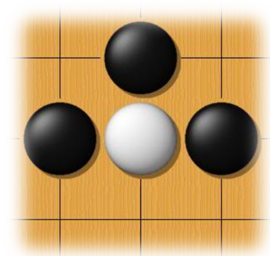
Fully-connected feedforward network can be used

But CNN performs much better.

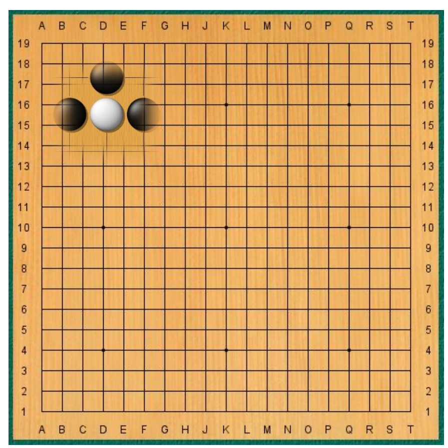
Why CNN for playing Go?

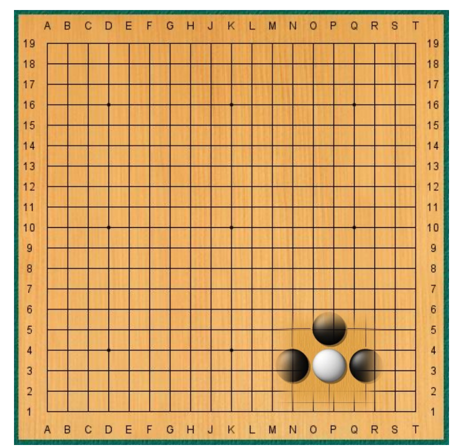
- Some patterns are much smaller than the whole image

Alpha Go uses 5 x 5 for first layer



- The same patterns appear in different regions.





Why CNN for playing Go?

- Subsampling the pixels will not change the object

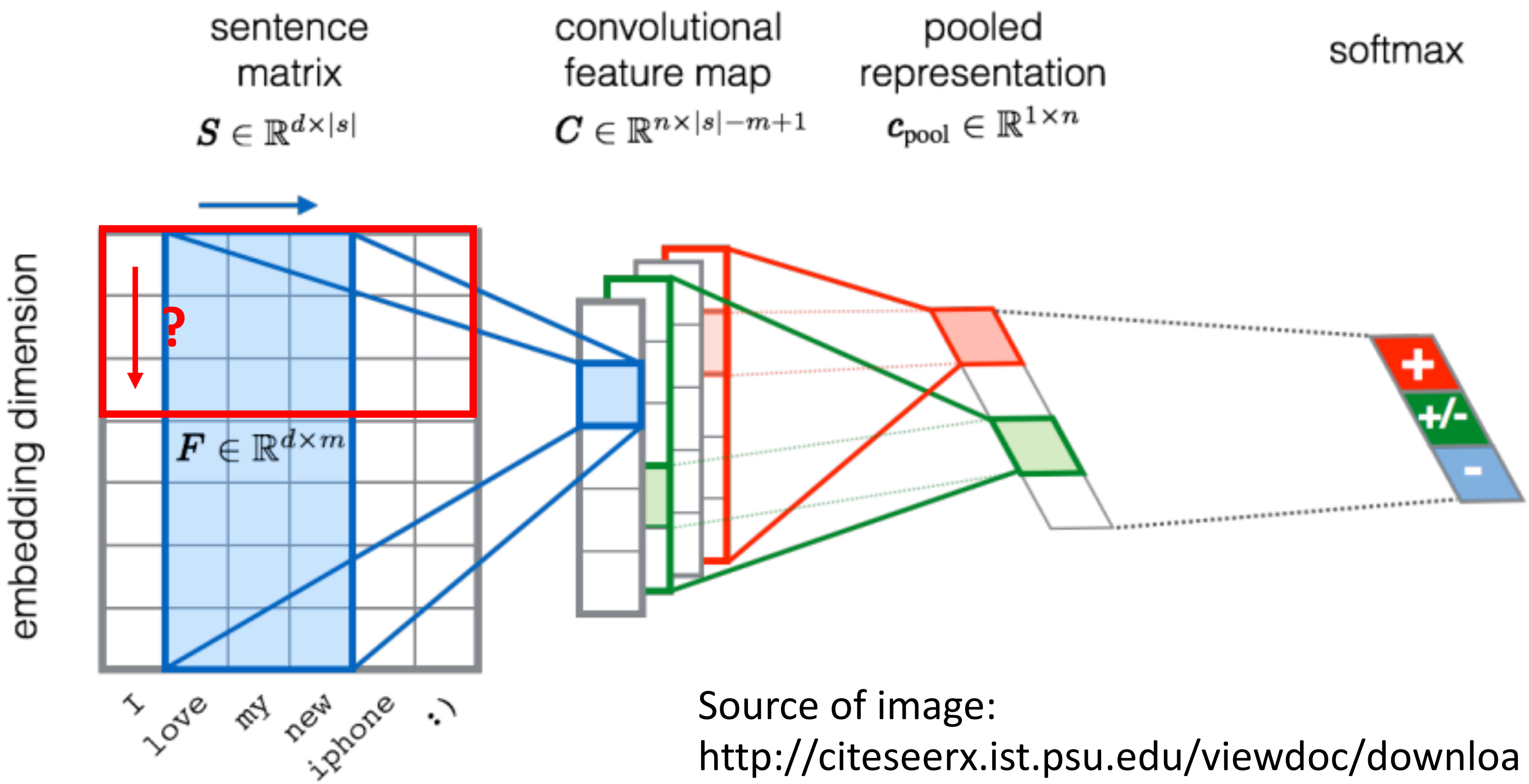
Max Pooling

How to explain this???

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 23×23 image, then convolves k filters of kernel size 5×5 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 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with strid … max func-
tion. The … Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

Alpha Go does not use Max Pooling ……

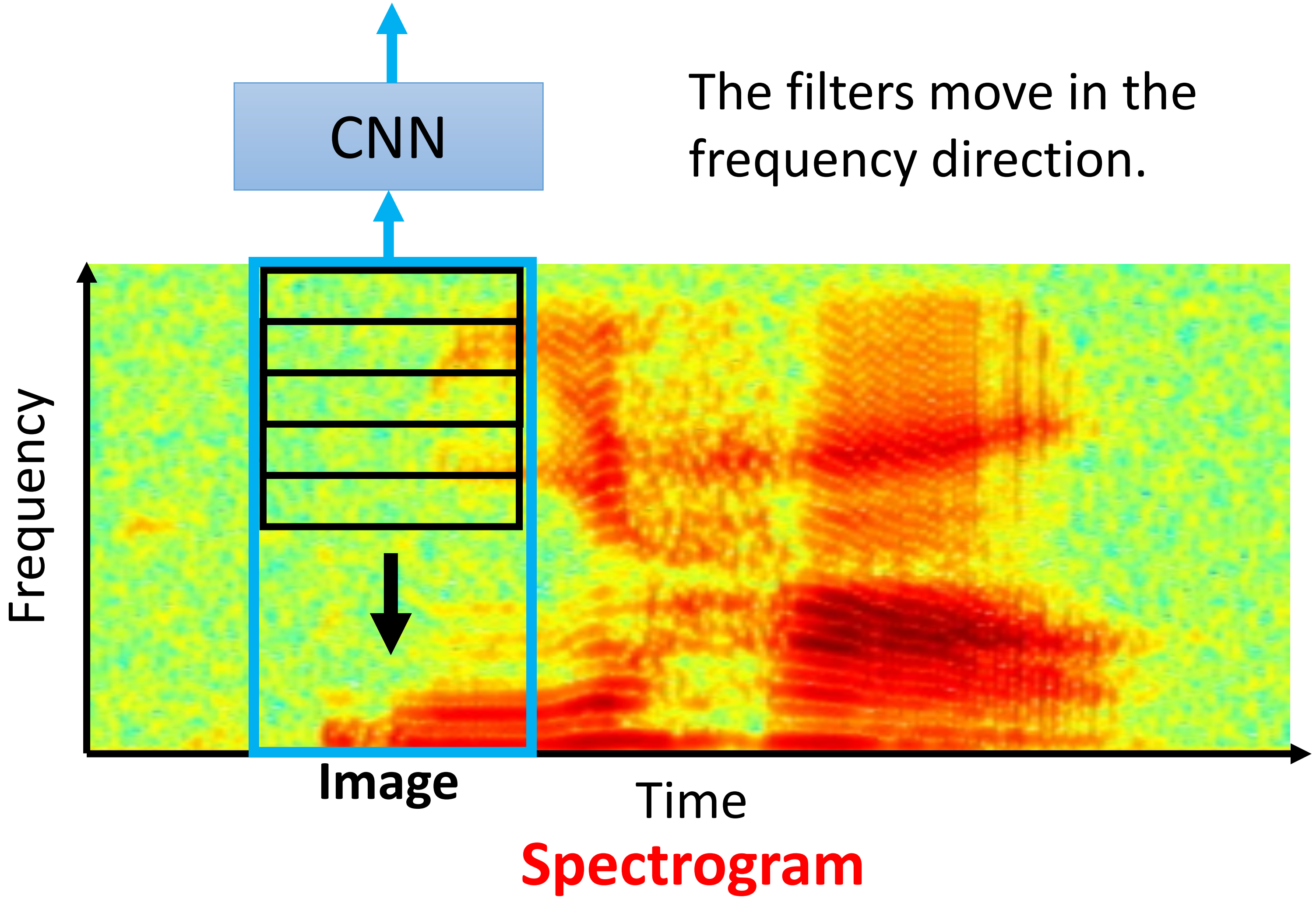
More Application: Text



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

More Application: Speech

The filters move in the frequency direction.



To learn more ……

- The methods of visualization in these slides
  - https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html
- More about visualization
  - http://cs231n.github.io/understanding-cnn/
- Very cool CNN visualization toolkit
  - http://yosinski.com/deepvis
  - http://scs.ryerson.ca/~aharley/vis/conv/
- The 9 Deep Learning Papers You Need To Know About
  - https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

To learn more ……

- How to let machine draw an image
    - PixelRNN
        - https://arxiv.org/abs/1601.06759
    - Variation Autoencoder (VAE)
        - https://arxiv.org/abs/1312.6114
    - Generative Adversarial Network (GAN)
        - http://arxiv.org/abs/1406.2661