Convolutional Neural Network

Professor Hung-yi Lee
Professor Pei-Yuan Wu
National Taiwan University

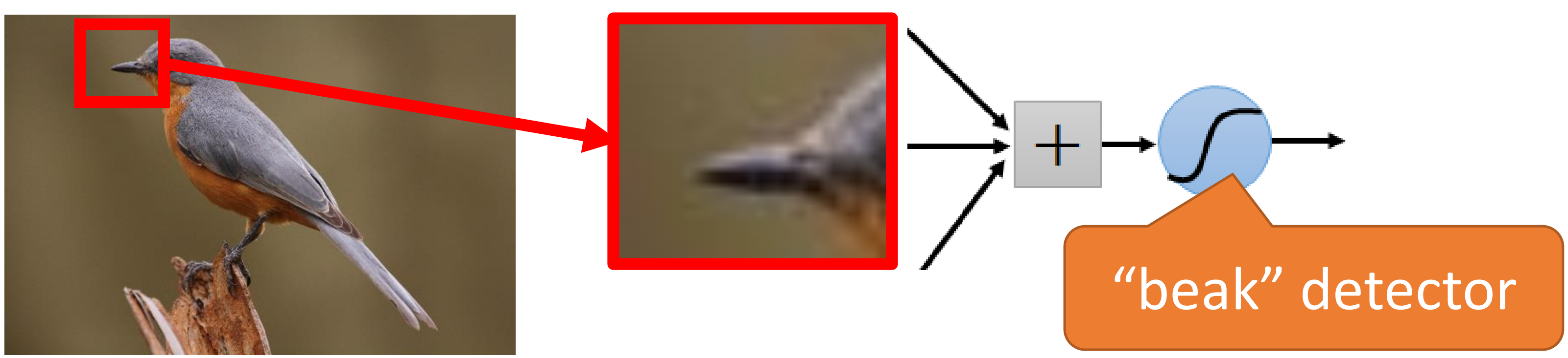
Can the network be simplified by considering the properties of images?

Why CNN for Image

- Some patterns are much smaller than the whole image

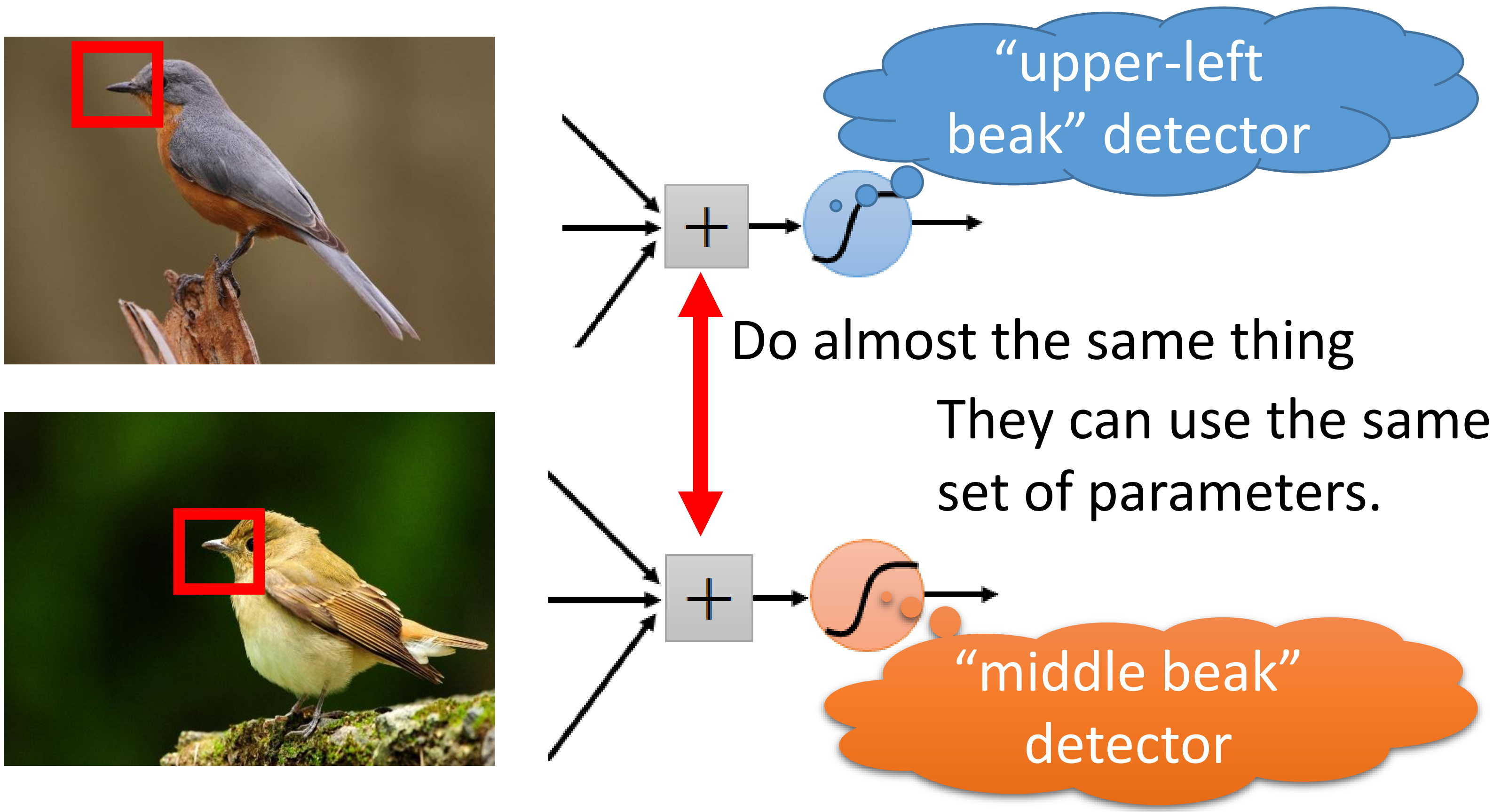
A neuron does not have to see the whole image to discover the pattern.

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

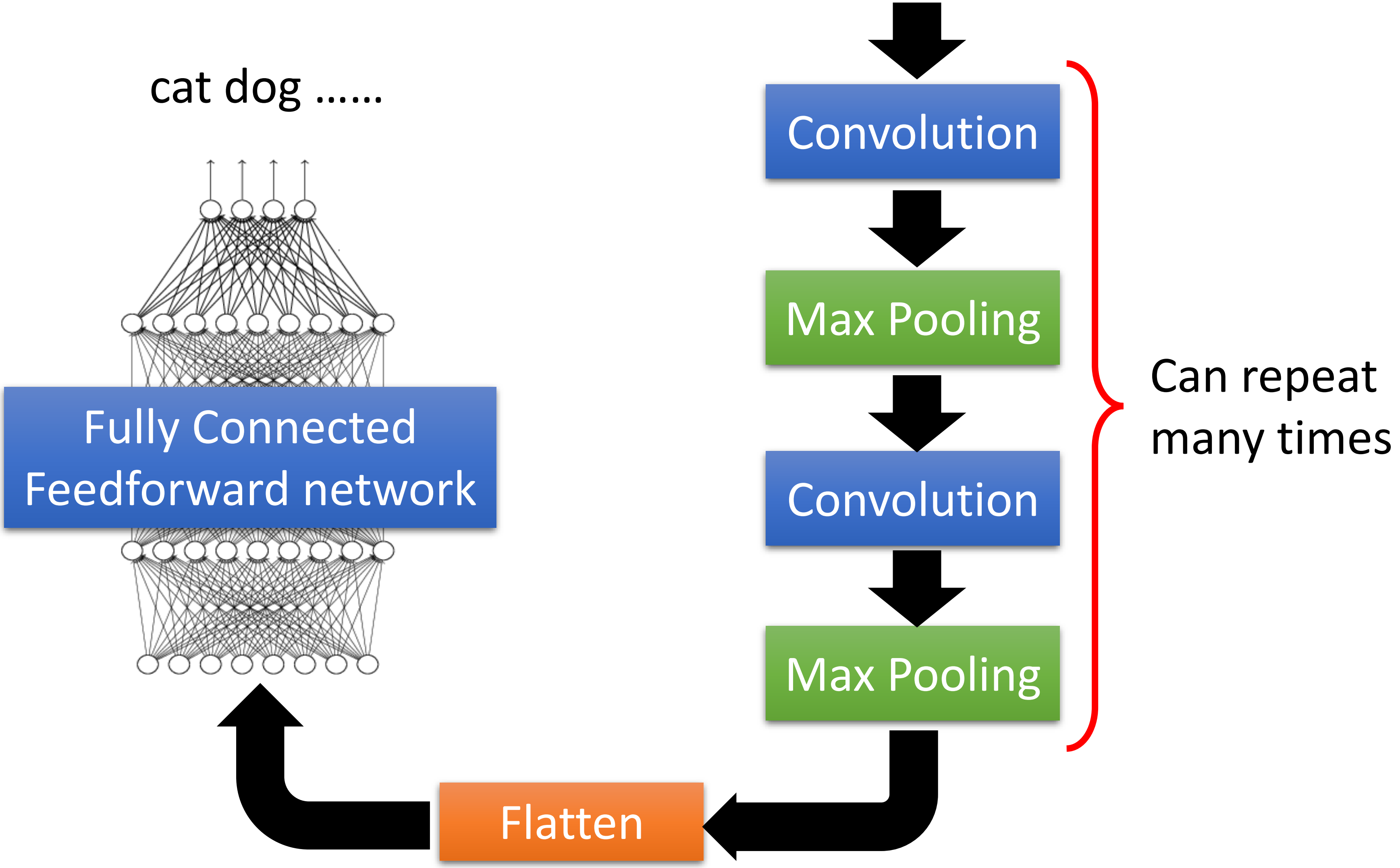
subsampling

bird

We can subsample the pixels to make image smaller

Less parameters for the network to process the image

The whole CNN



The whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

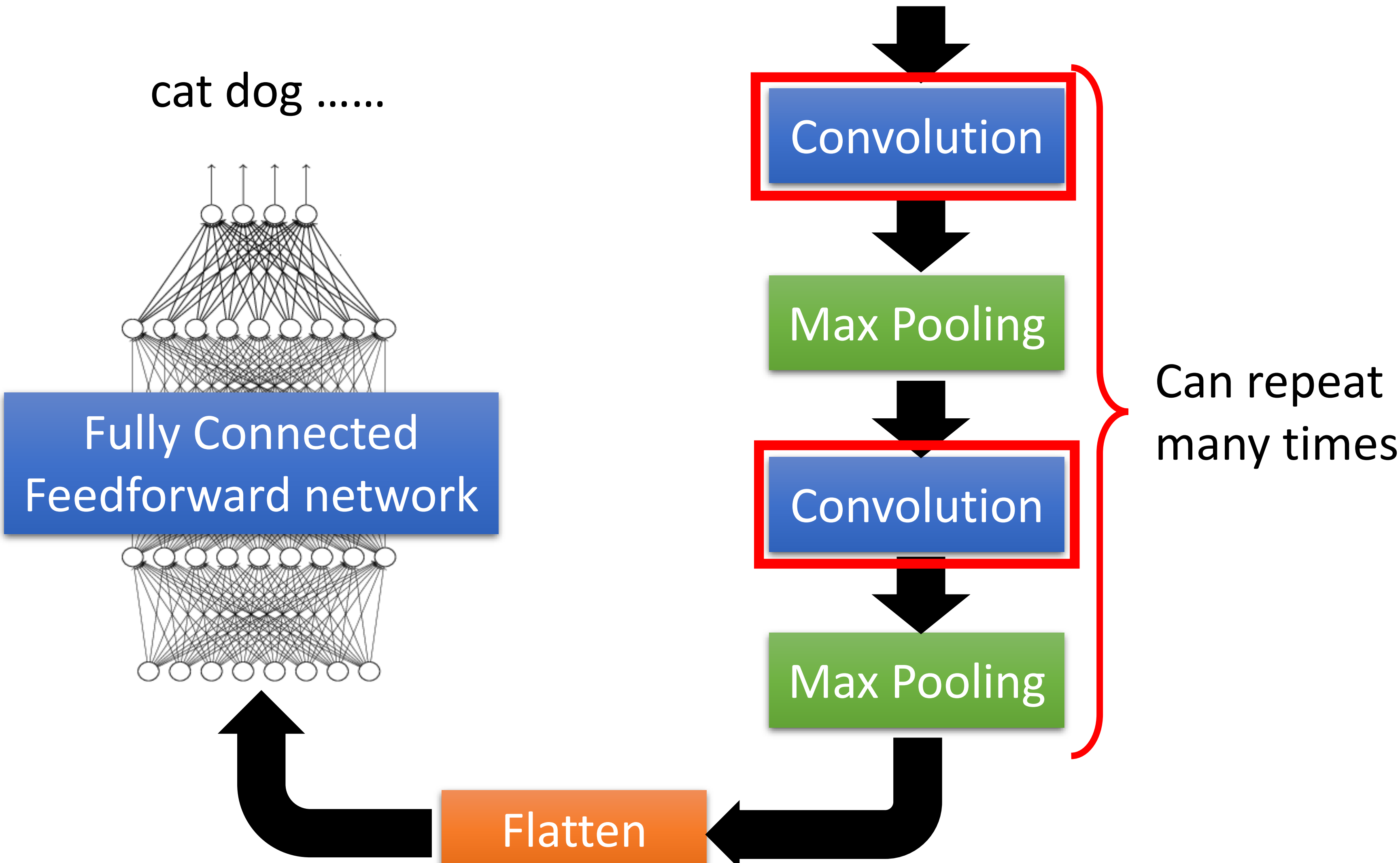
Convolution

Max Pooling

Can repeat many times

Flatten

The whole CNN



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

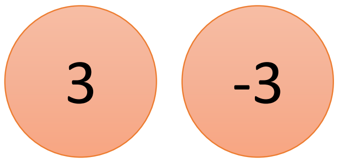
1	-1	-1
-1	1	-1
-1	-1	1

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



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

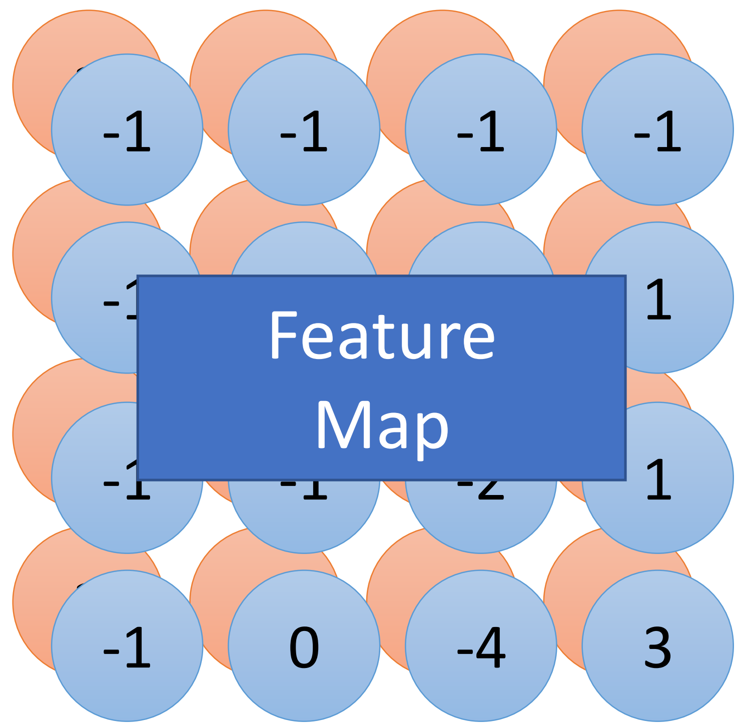
Filter 2

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

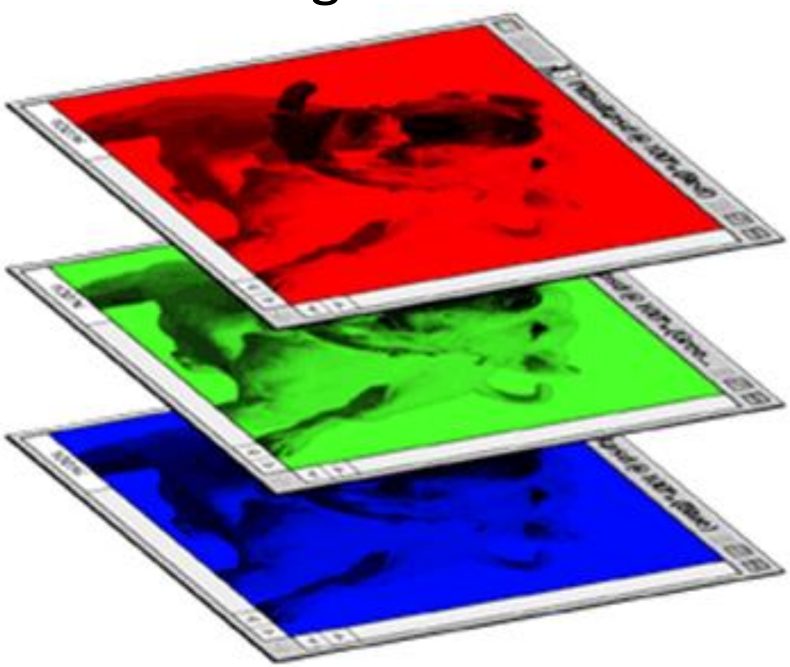
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

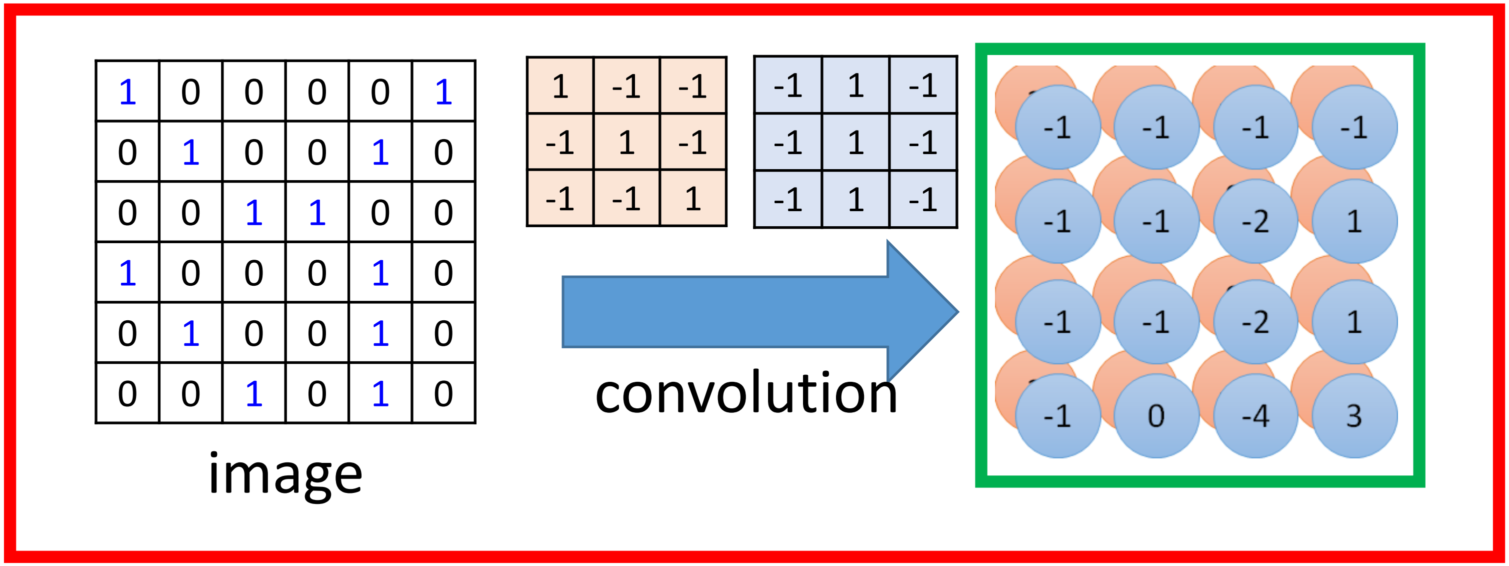
Filter 2

Colorful image

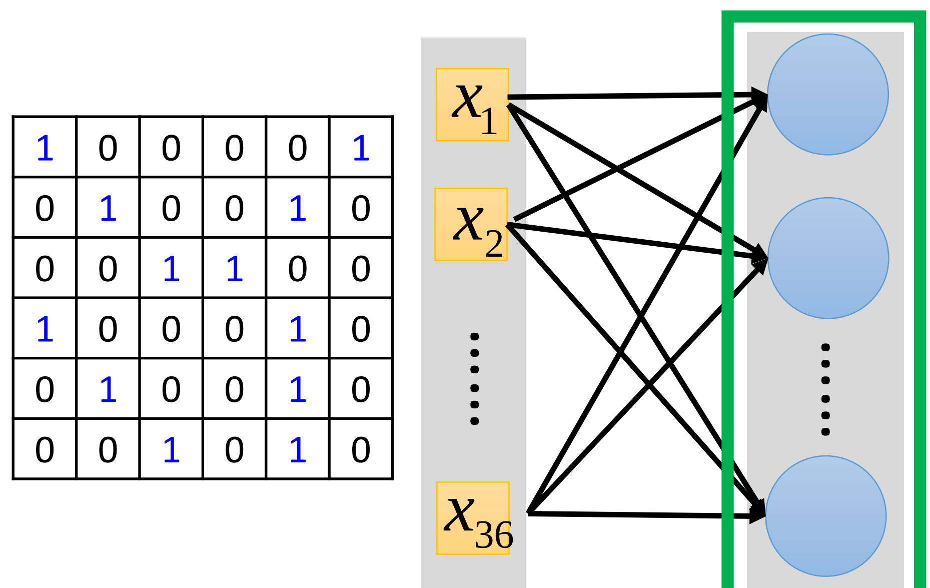


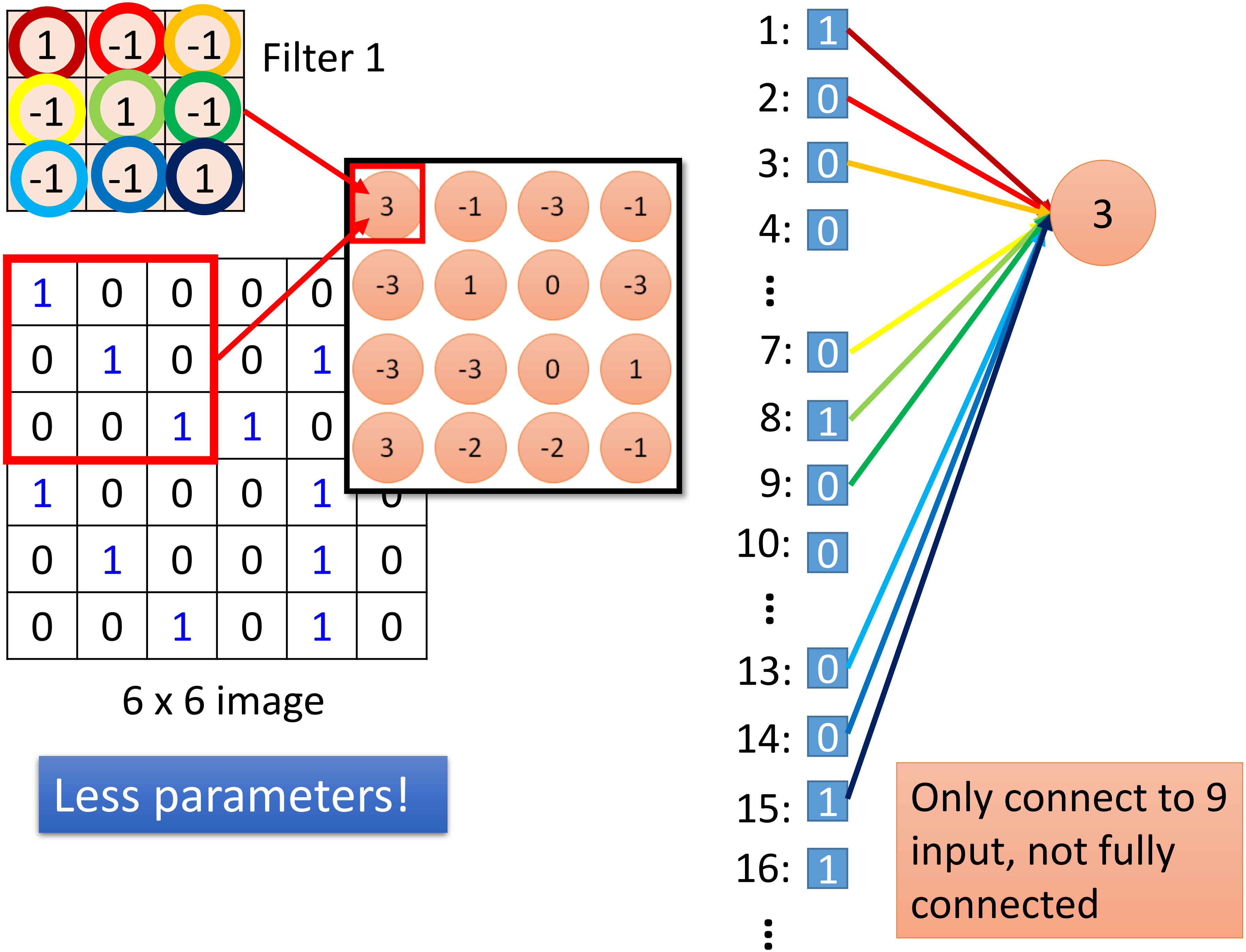
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Convolution v.s. Fully Connected

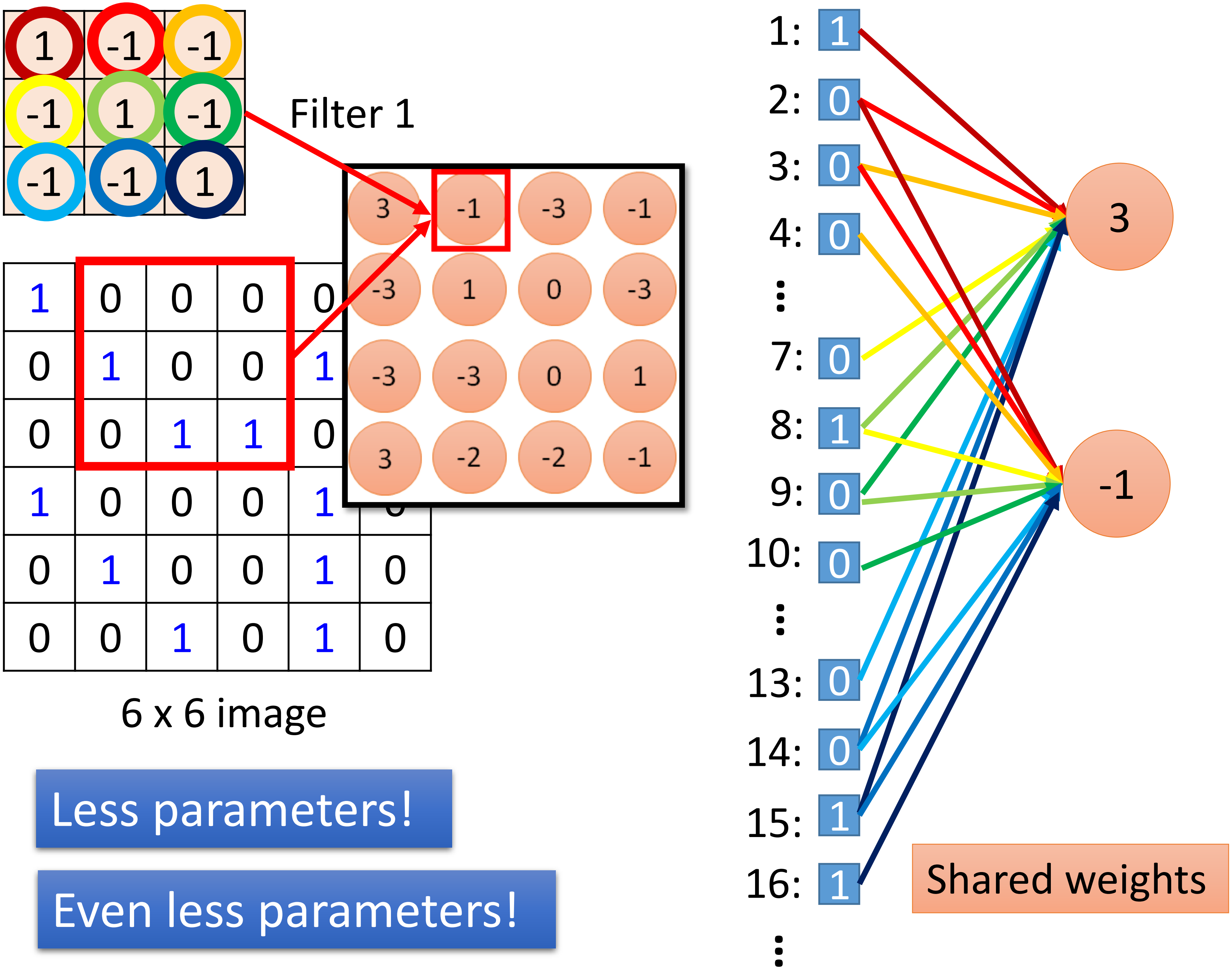


Fully-connected



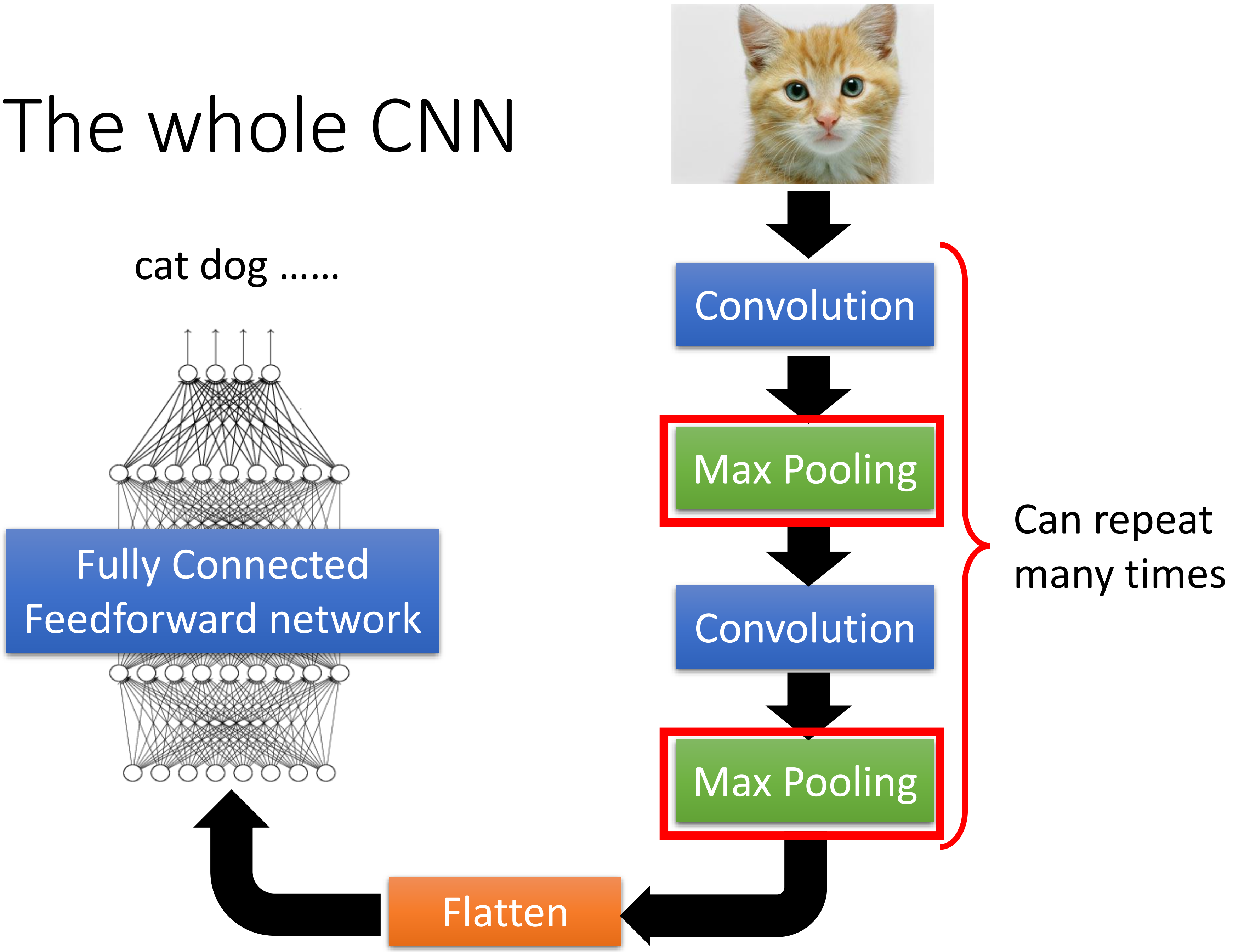


Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
6 x 6 image
Less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Only connect to 9 input, not fully connected



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 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
Less parameters!
Even less parameters!
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
-1
Shared weights

The whole CNN



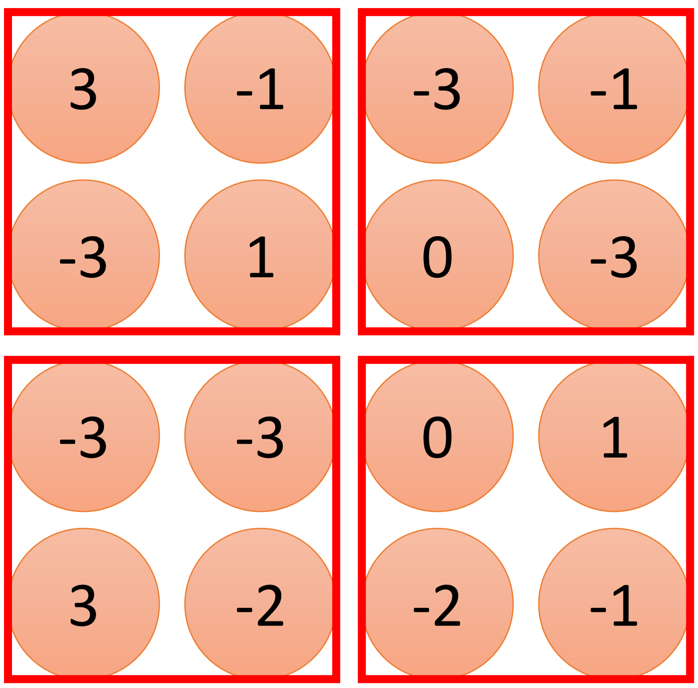
CNN – Max Pooling

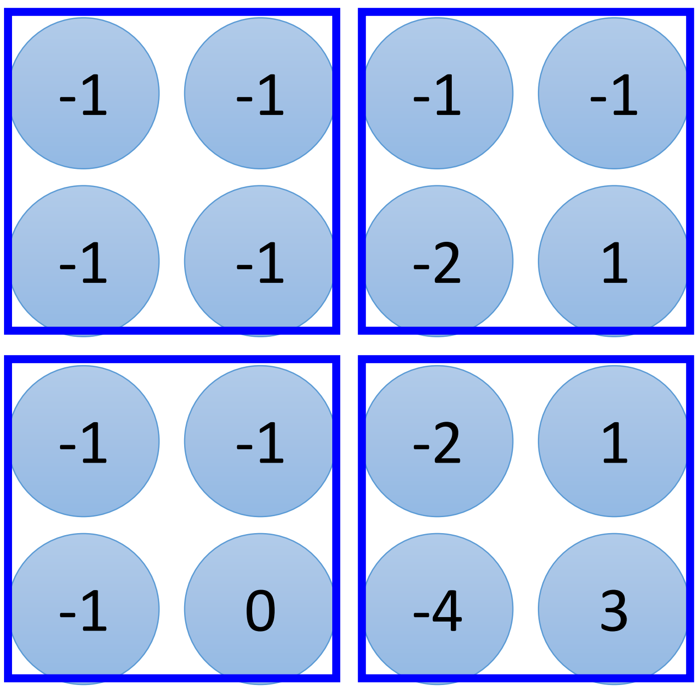
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

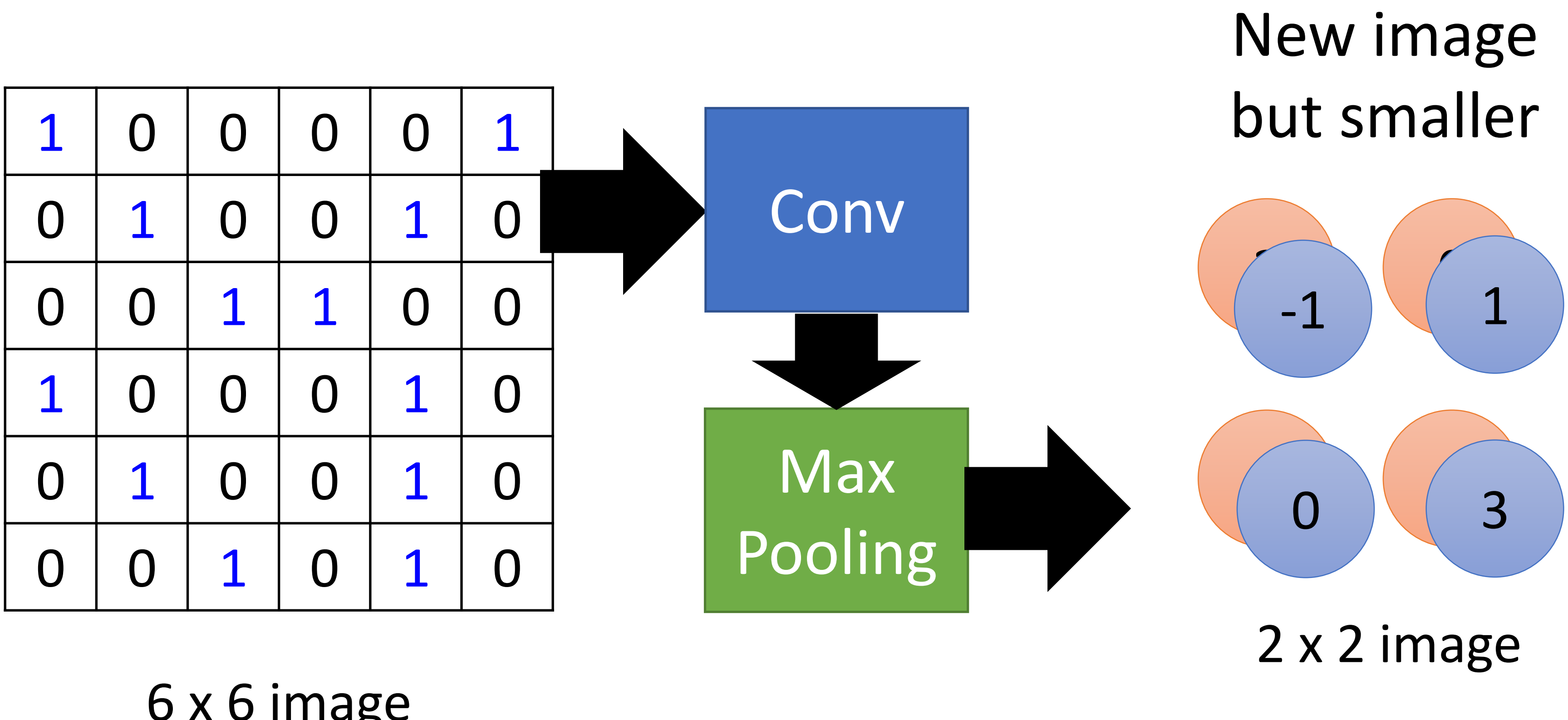
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2



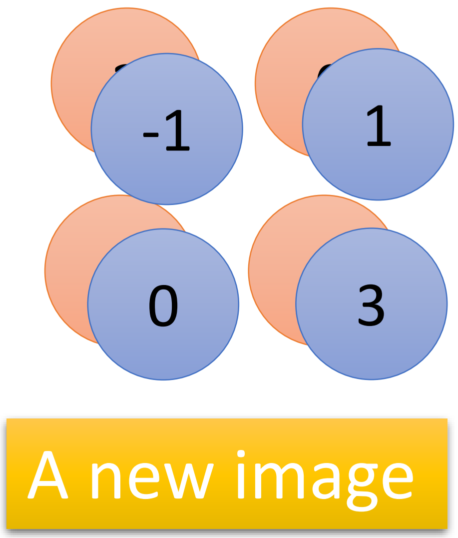


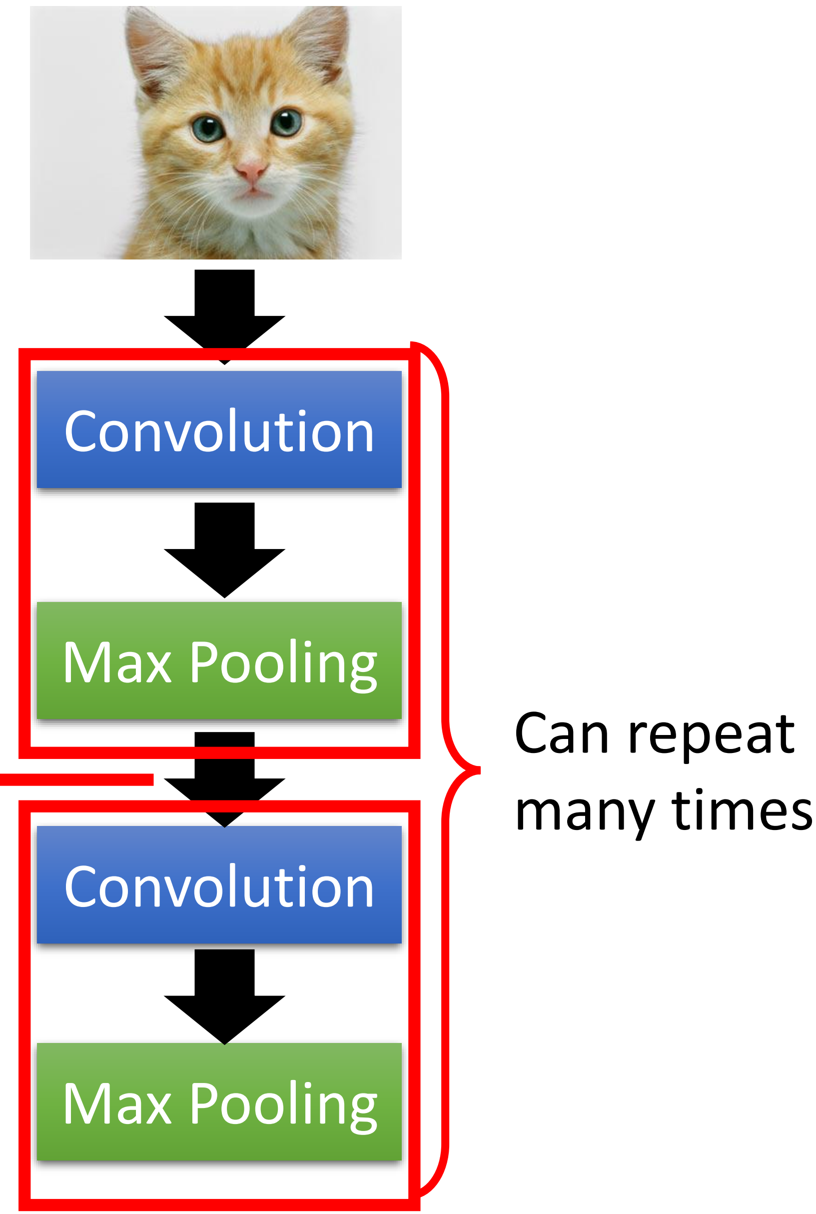
CNN – Max Pooling



Each filter
is a channel

The whole CNN

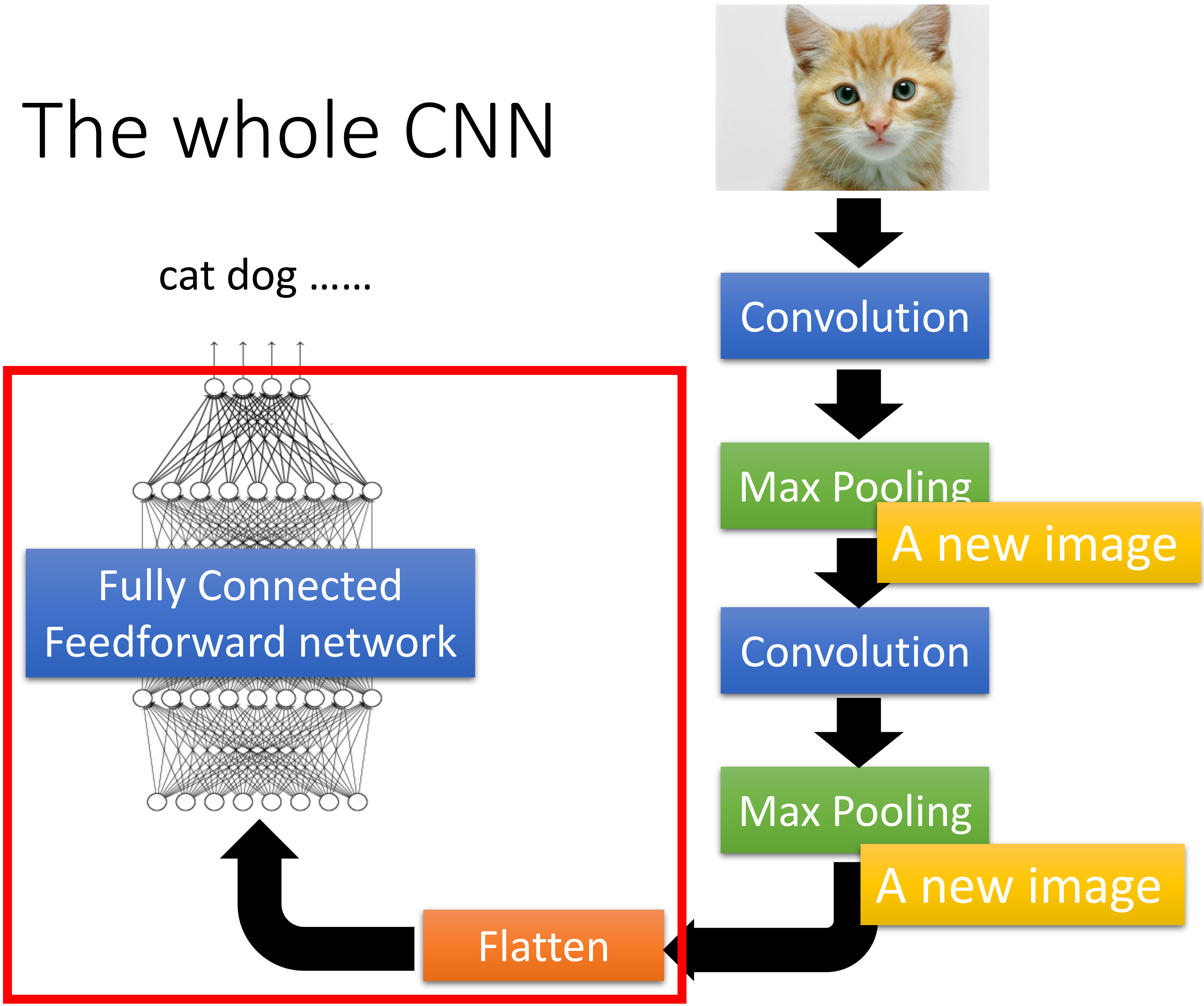




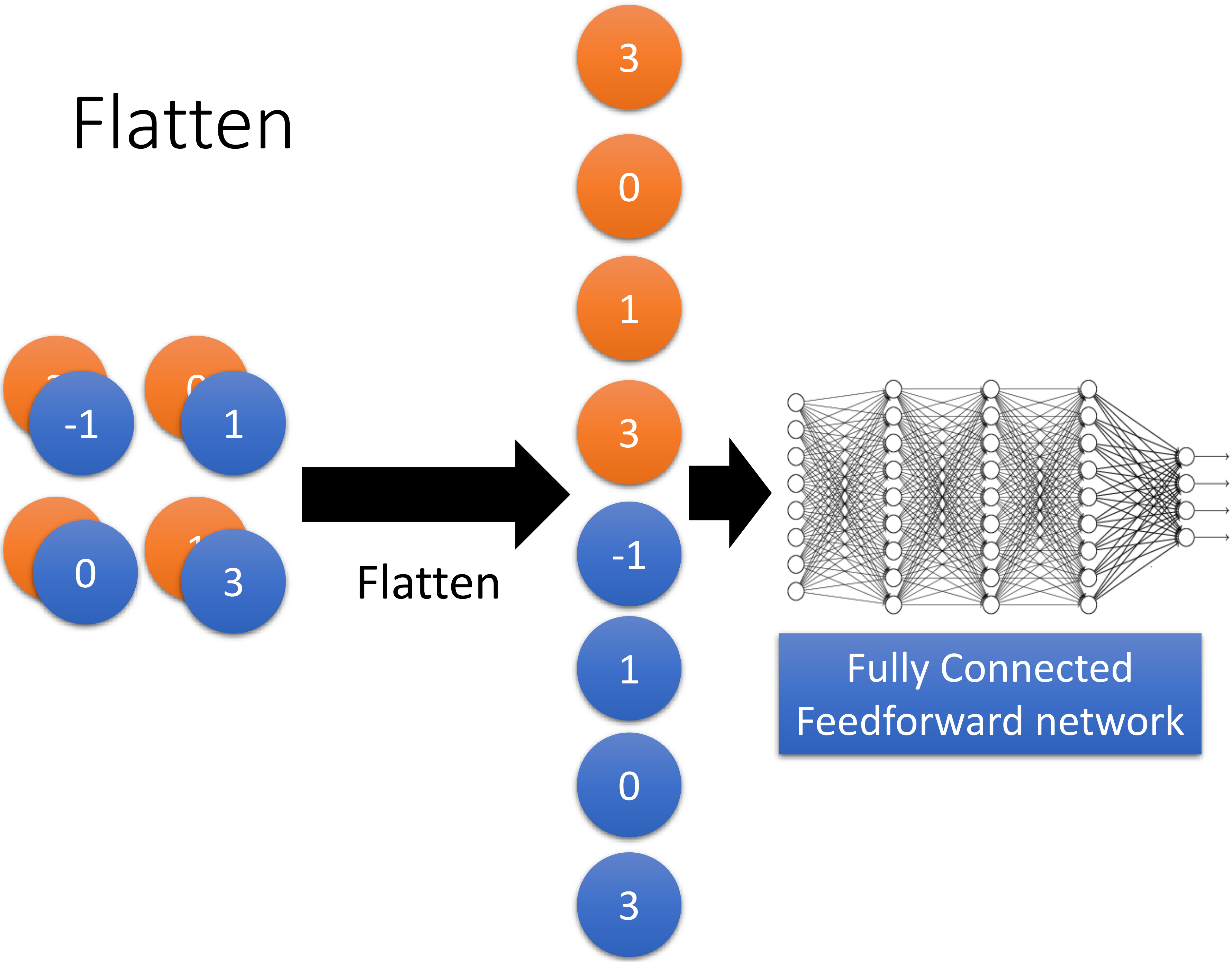
Smaller than the original image

The number of the channel is the number of filters

The whole CNN



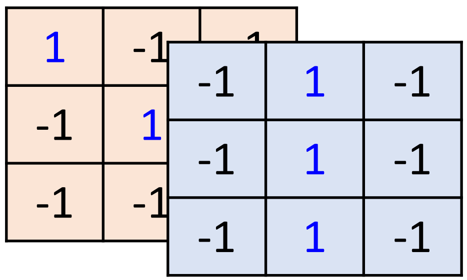
Flatten


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

CNN 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)) )
```



……

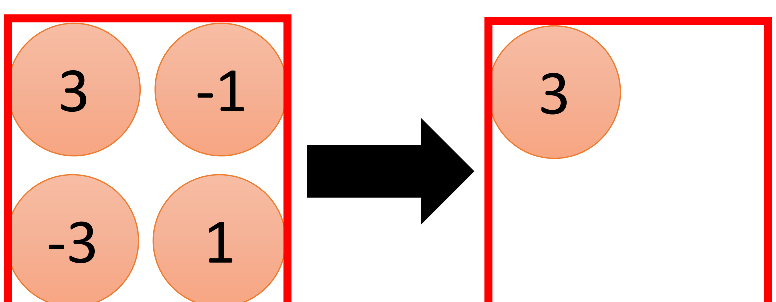
There are 25 3x3 filters.

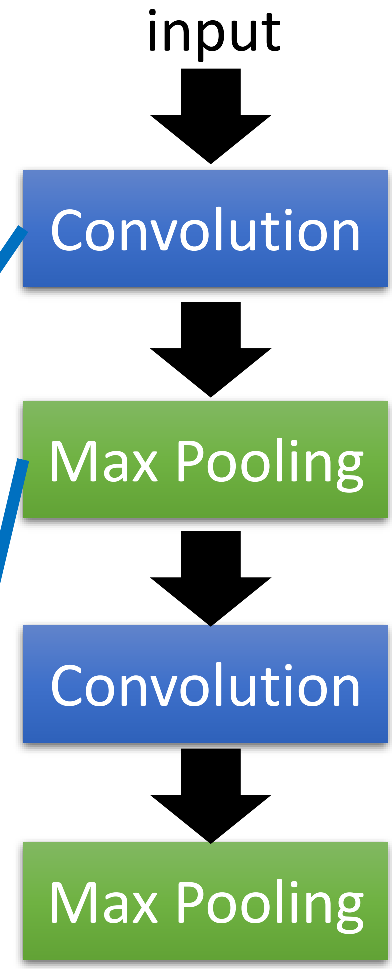
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

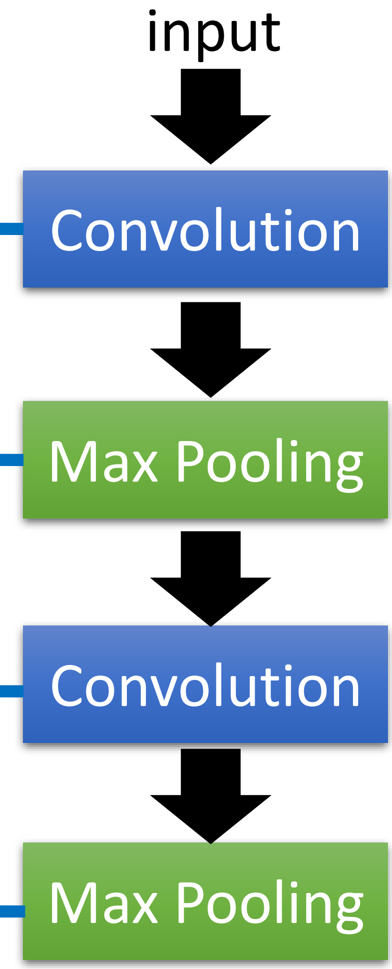
```
model2.add(MaxPooling2D((2,2)))
```





CNN in Keras

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



28 x 28 x 1

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

How many parameters for each filter? **9**

26 x 26 x 25

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

13 x 13 x 25

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

How many parameters for each filter? **225**

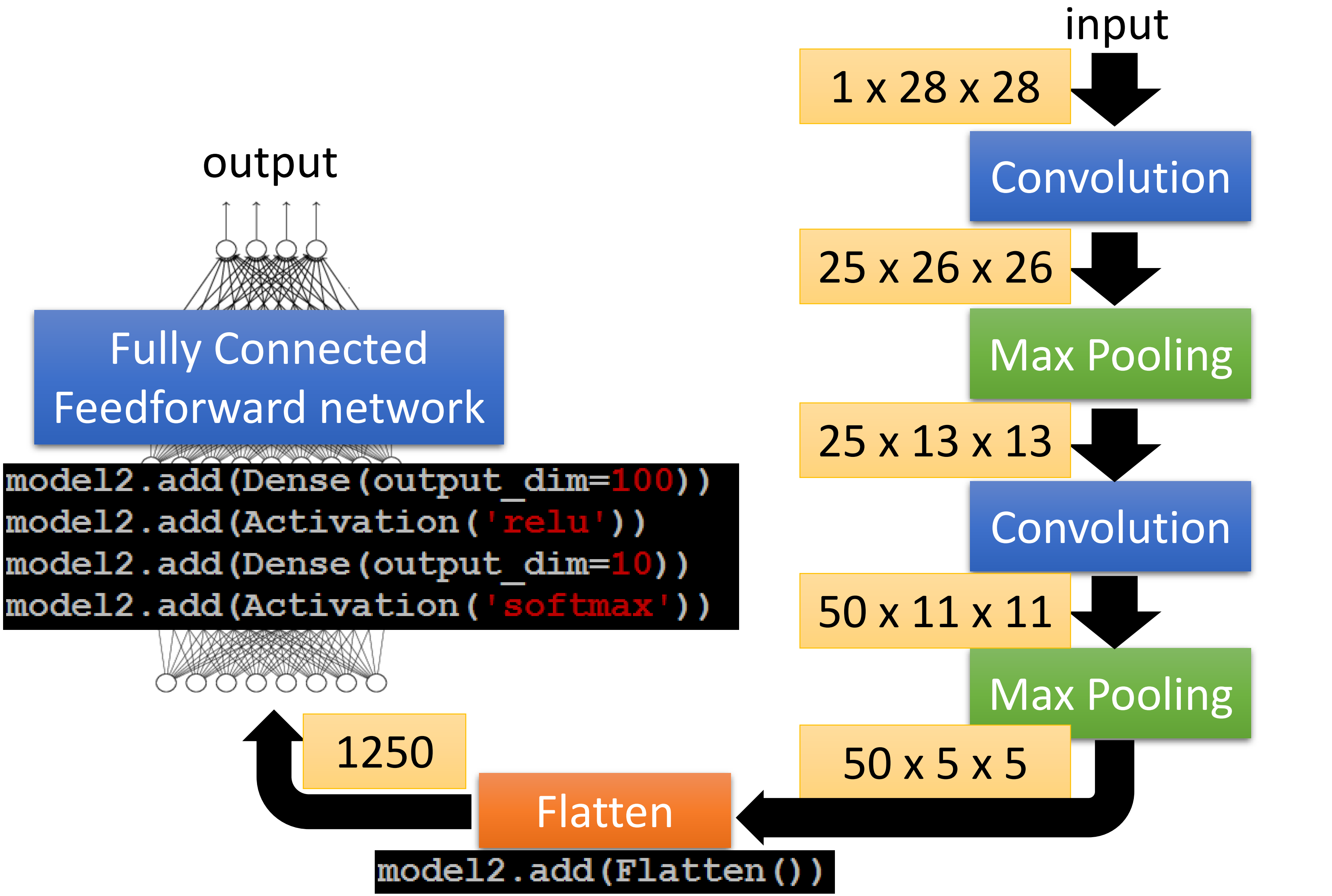
11 x 11 x 50

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

5 x 5 x 50

CNN in Keras

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




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

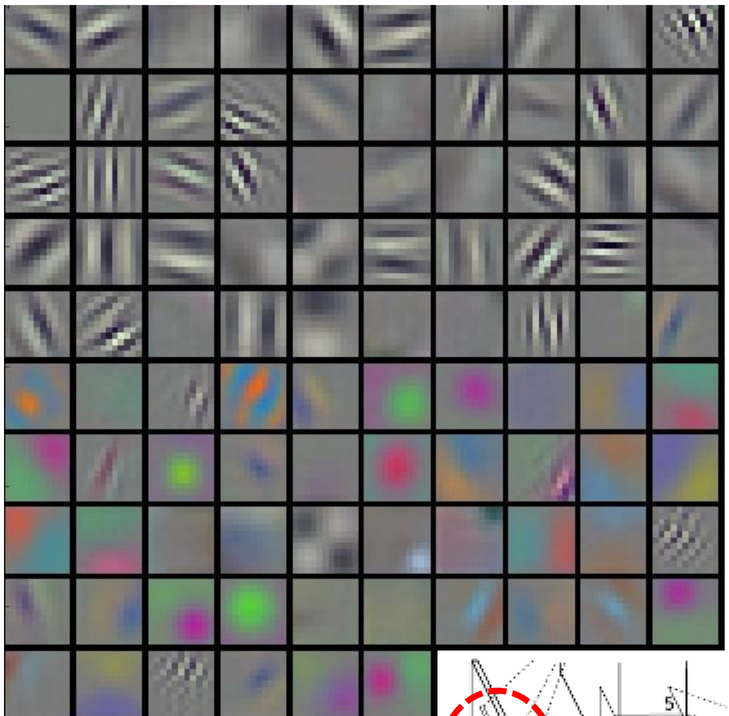
```
model2.add(Flatten())
```

What does machine learn?

http://newsneakernews.wpengine.netdna-cdn.com/wp-content/uploads/2016/11/rihanna-puma-creeper-velvet-release-date-02.jpg

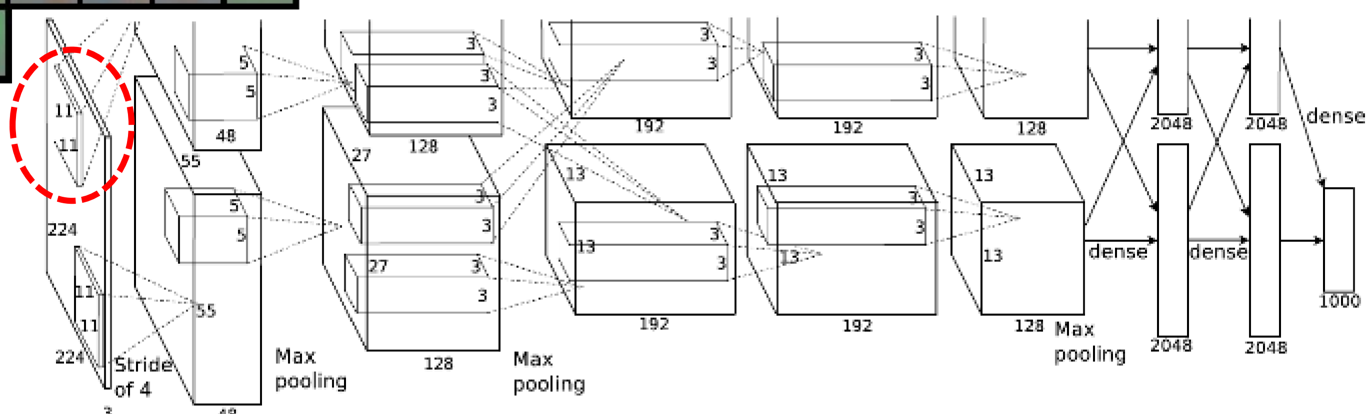
First Convolution Layer

- Typical-looking of filter weights on the trained first layer.



First layer:
11 x 11 x 3 (RGB)
48 filters x 2 streams

AlexNet

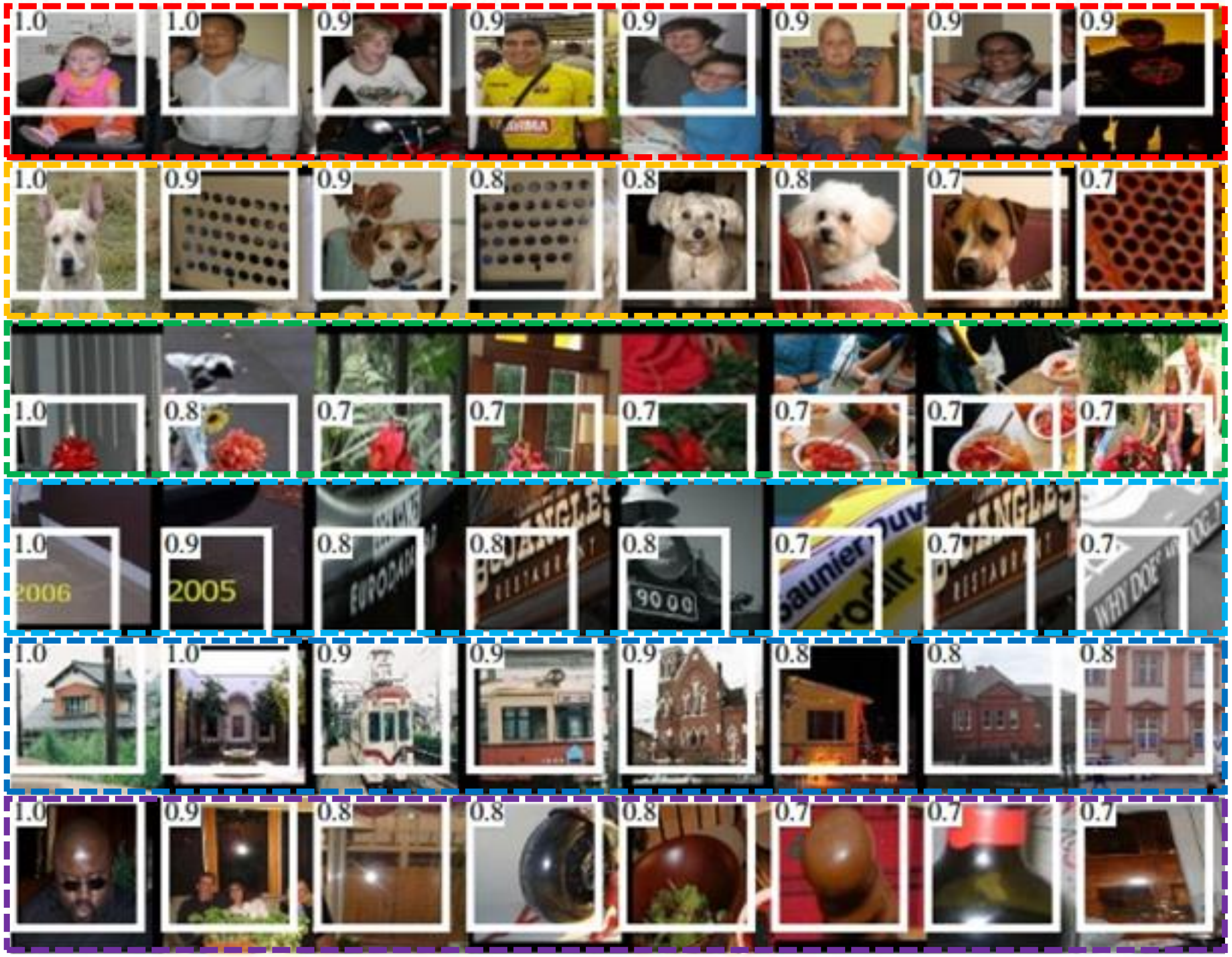


http://cs231n.github.io/understanding-cnn/

https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637

How about higher layers?

- Which images maximally activates a specific neuron.



Neuron A

Neuron B

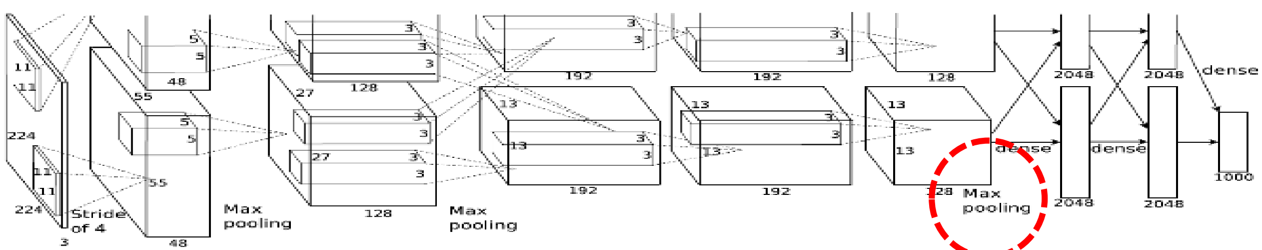
Neuron C

Neuron D

Neuron E

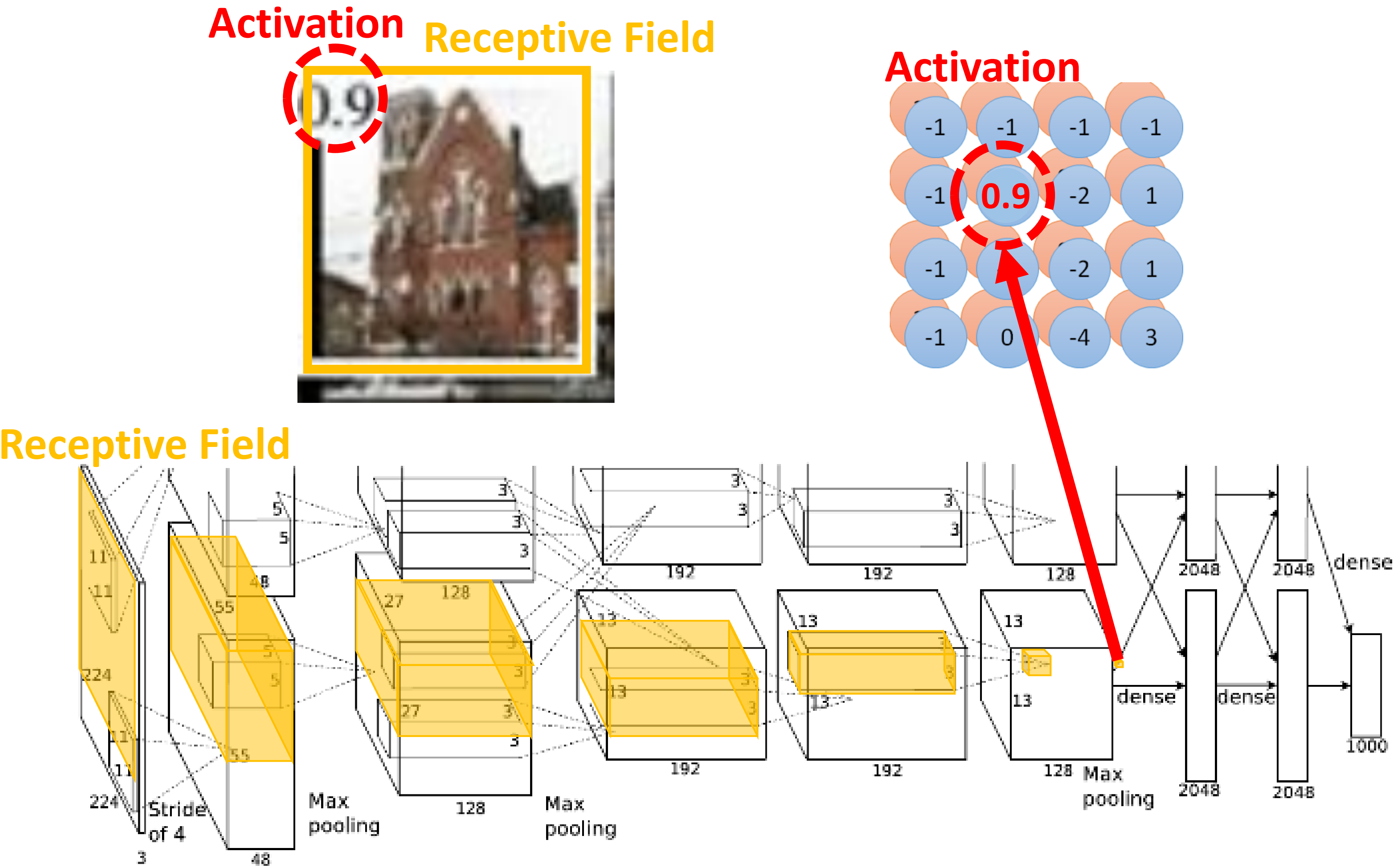
Neuron F

AlexNet



Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR, 2014

Activation and Receptive Field



What does CNN learn?

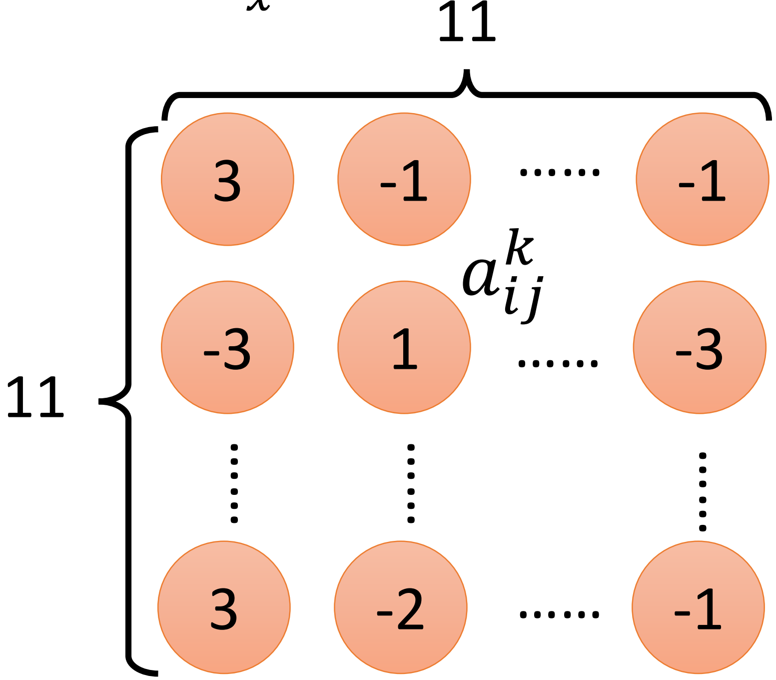
Idea: What is the image that maximally activates a specific filter?

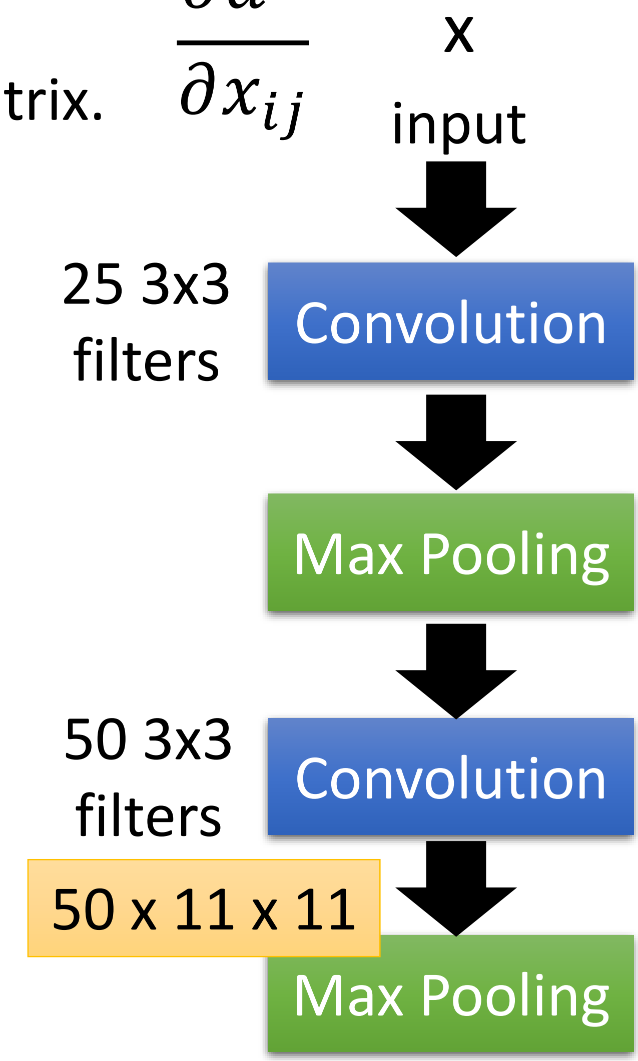
The output of the k-th filter is a 11 x 11 matrix.

$$\frac{\partial a^k}{\partial x_{ij}}$$

Degree of the activation of the k-th filter: $a^k = \sum_{i=1}^{11} \sum_{j=1}^{11} a_{ij}^k$

$$x^* = arg \max_x a^k$$ (gradient ascent)





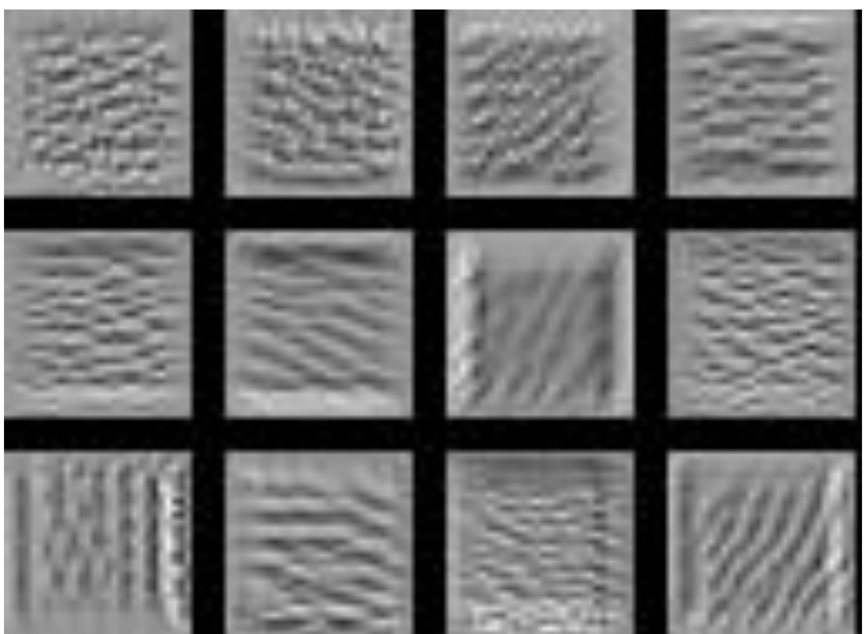
What does CNN learn?

Idea: What is the image that maximally activates a specific filter?

The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter: $a^k = \sum_{i=1}^{11} \sum_{j=1}^{11} a_{ij}^k$

$$x^* = arg \max_x a^k \quad \text{(gradient ascent)}$$



$$\frac{\partial a^k}{\partial x_{ij}}$$

x

input

25 3x3 filters — Convolution

Max Pooling

50 3x3 filters — Convolution

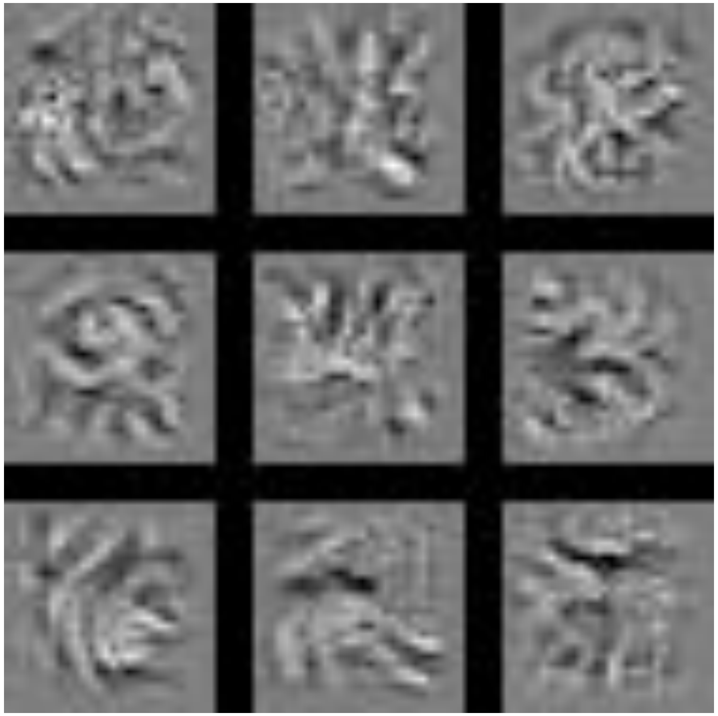
50 x 11 x 11

Max Pooling

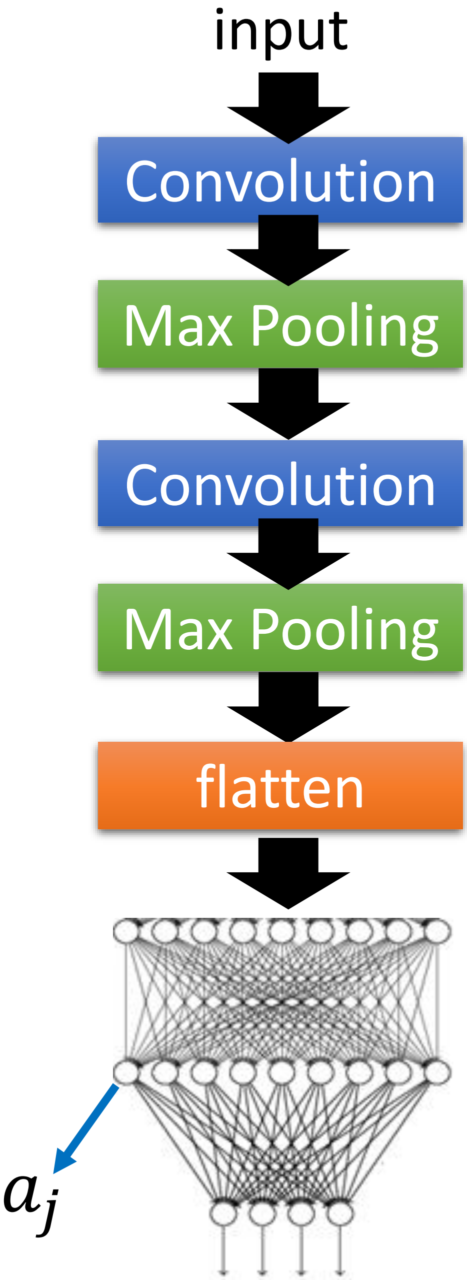
What does CNN learn?

Find an image maximizing the output of neuron:

$$x^* = arg \max_x a^j$$



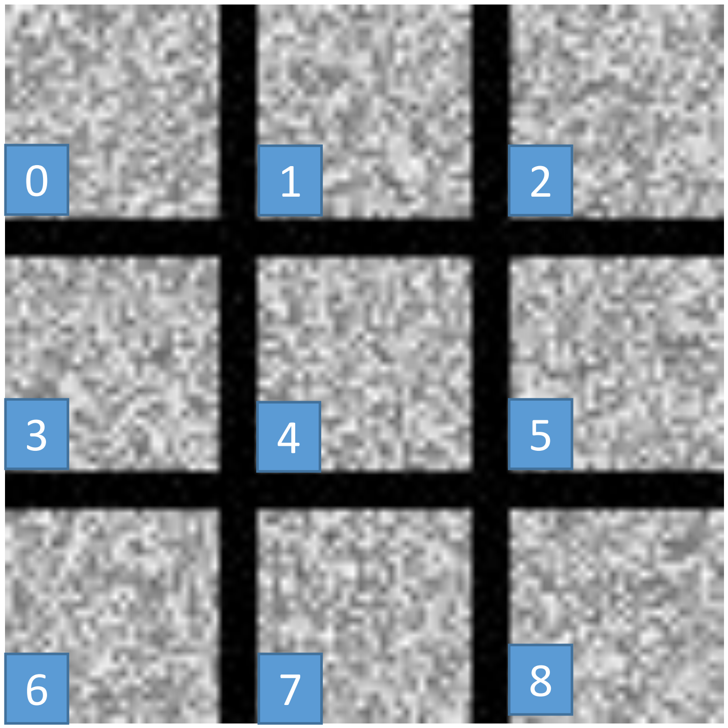
Each figure corresponds to a neuron



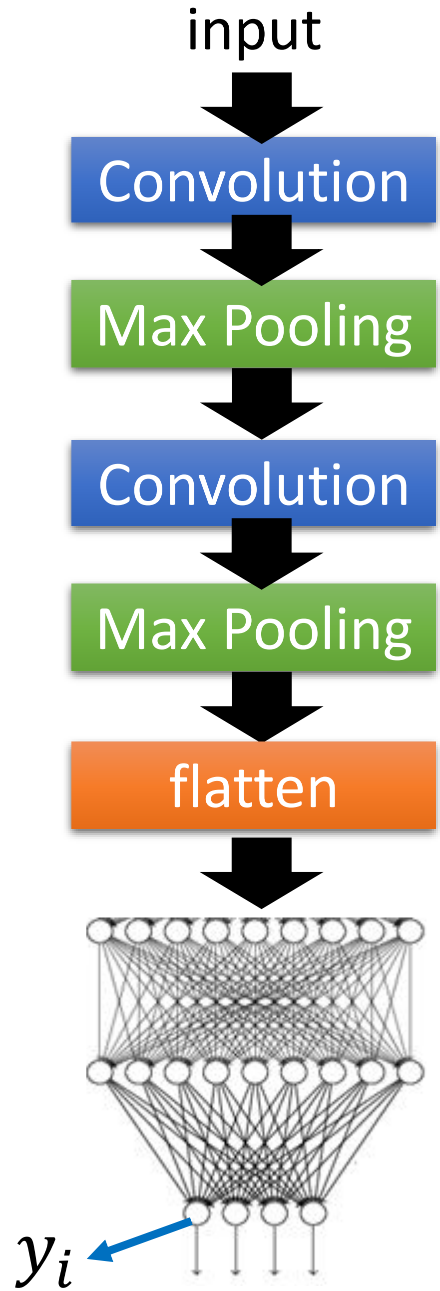
What does CNN learn?

$$x^* = arg \max_x y^i$$

Can we see digits?

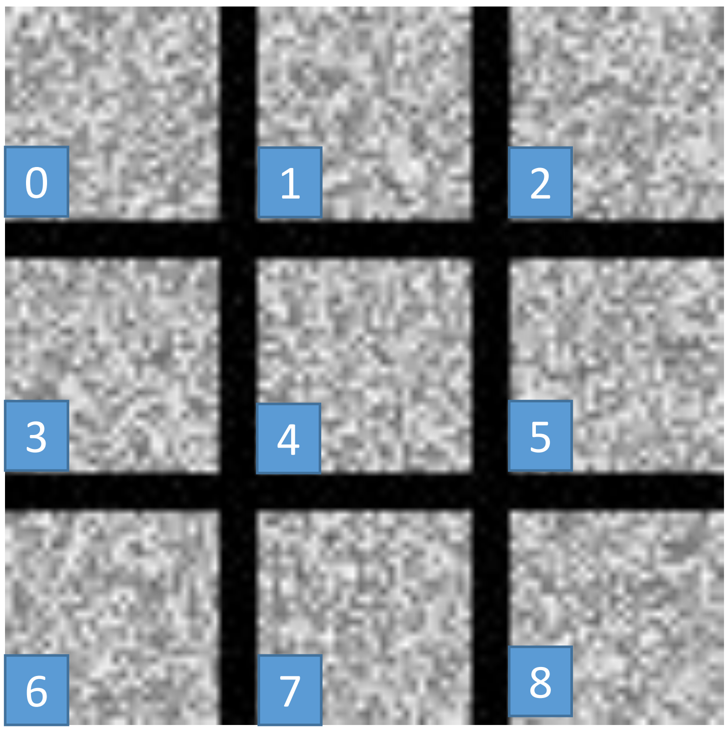


Deep Neural Networks are Easily Fooled
https://www.youtube.com/watch?v=M2IebCN9Ht4



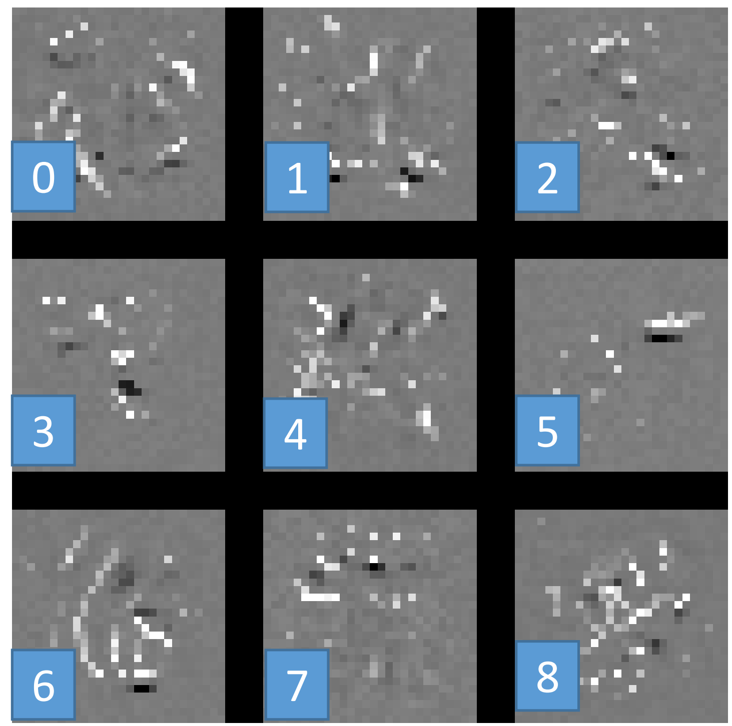
What does CNN learn?

$$x^* = arg\max_x y^i$$



Over all pixel values

$$x^* = arg\max_x \left(y^i - \boxed{\sum_{i,j} |x_{ij}|} \right)$$



$$x^* = arg \max_x \left(y^i - \sum_{i,j} \left| x_{ij} \right|^2 \right)$$

Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR, 2014

$$\left|\frac{\partial y_k}{\partial x_{ij}}\right|$$

y_k: the predicted class of the model

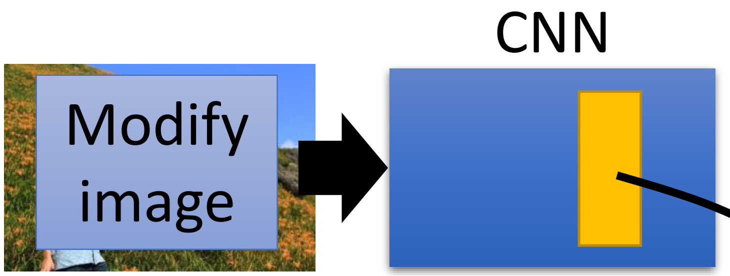
Pixel x_{ij}

Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR, 2014

Occlusion sensitivity

Reference: Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014* (pp. 818-833)

Deep Dream



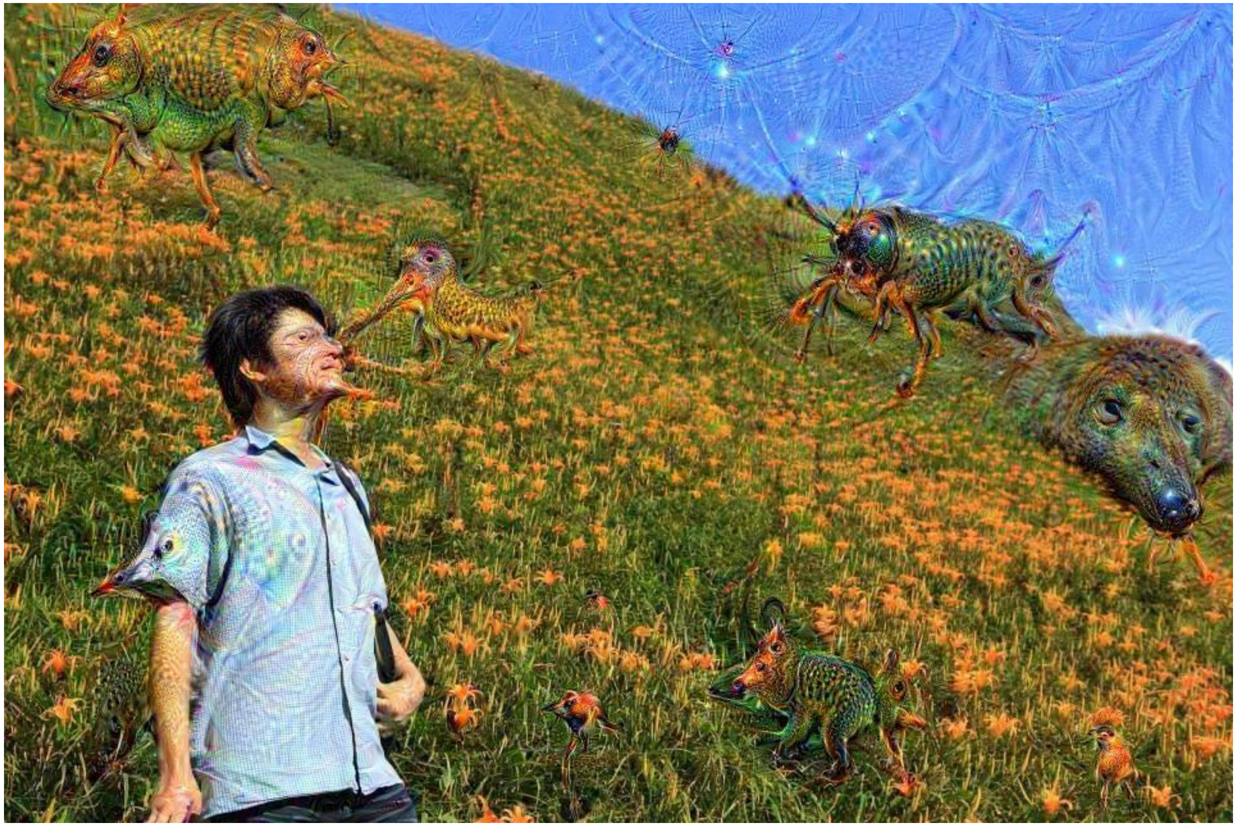
- Given a photo, machine adds what it sees ……

$$\begin{bmatrix} 3.9 \\ -1.5 \\ 2.3 \\ \vdots \end{bmatrix}$$

http://deepdreamgenerator.com/

Deep Dream

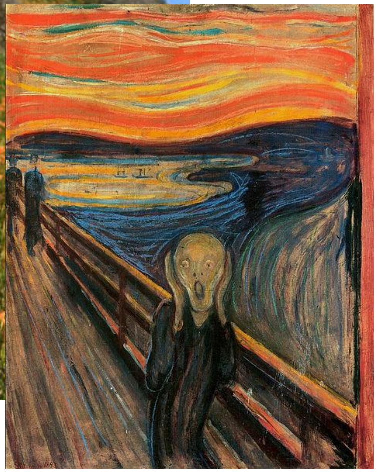
- Given a photo, machine adds what it sees ……



http://deepdreamgenerator.com/

Deep Style

- Given a photo, make its style like famous paintings



https://dreamscopeapp.com/

Deep Style

- Given a photo, make its style like famous paintings

https://dreamscopeapp.com/

Deep Style

I_c

$F_i^{\ell}(I)$: Feature map from filter i at layer ℓ computed from image I

CNN

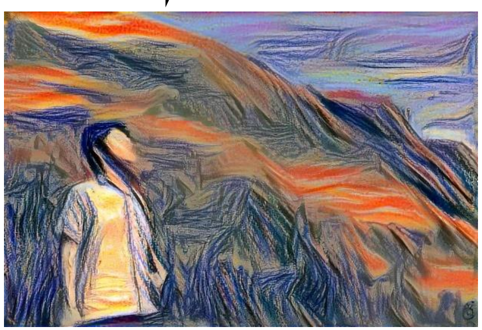
content

CNN

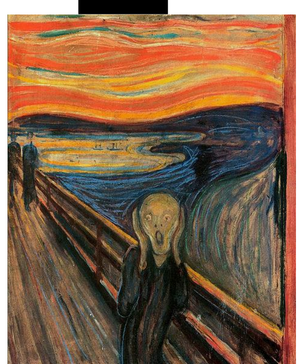
style

CNN

I_m



I_s



Content represented as feature maps

$F_i^{\ell}(I_c)$

$$L_{content}(I_m, I_c) = \sum_{\ell} \sum_{i,j} v_{ij}^{\ell} \left| F_i^{\ell}(I_m) - F_i^{\ell}(I_c) \right|^2$$

$$\boxed{\min_{I_m} \alpha L_{content}(I_m, I_c) + \beta L_{style}(I_m, I_s)}$$

$$L_{style}(I_m, I_s) = \sum_{\ell} \sum_{i,j} w_{ij}^{\ell} \left| G_{ij}^{\ell}(I_m) - G_{ij}^{\ell}(I_s) \right|^2$$

Style represented as correlation between feature maps

$$G_{ij}^{\ell}(I_s) = F_i^{\ell}(I_s) \cdot F_j^{\ell}(I_s)$$

A Neural Algorithm of Artistic Style

https://arxiv.org/abs/1508.06576

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.

More Application: Playing Go

Training: record of previous plays 黑: 5之五 → 白: 天元 → 黑: 五之5 …

CNN

Target:
"天元" = 1
else = 0

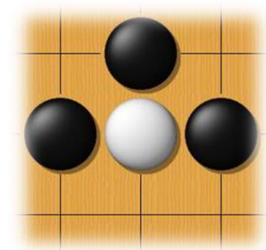
CNN

Target:
"五之 5" = 1
else = 0

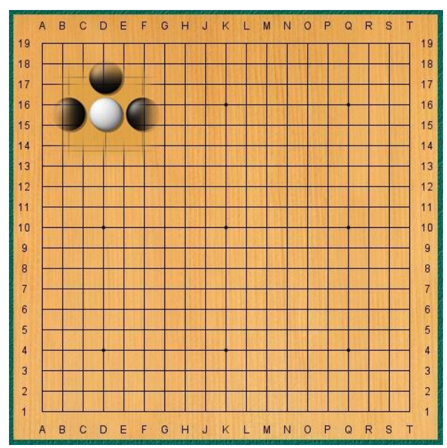
Why CNN for playing Go?

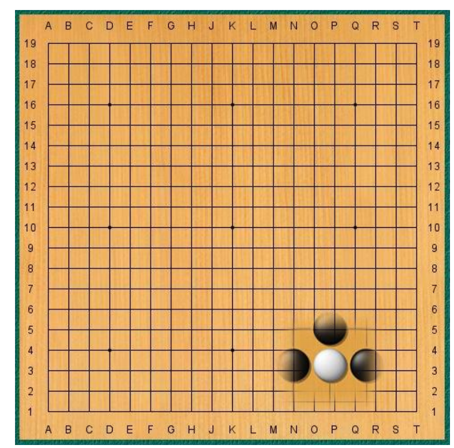
- Some patterns are much smaller than the whole image

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 × 19 × 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 ……

Image Dehazing

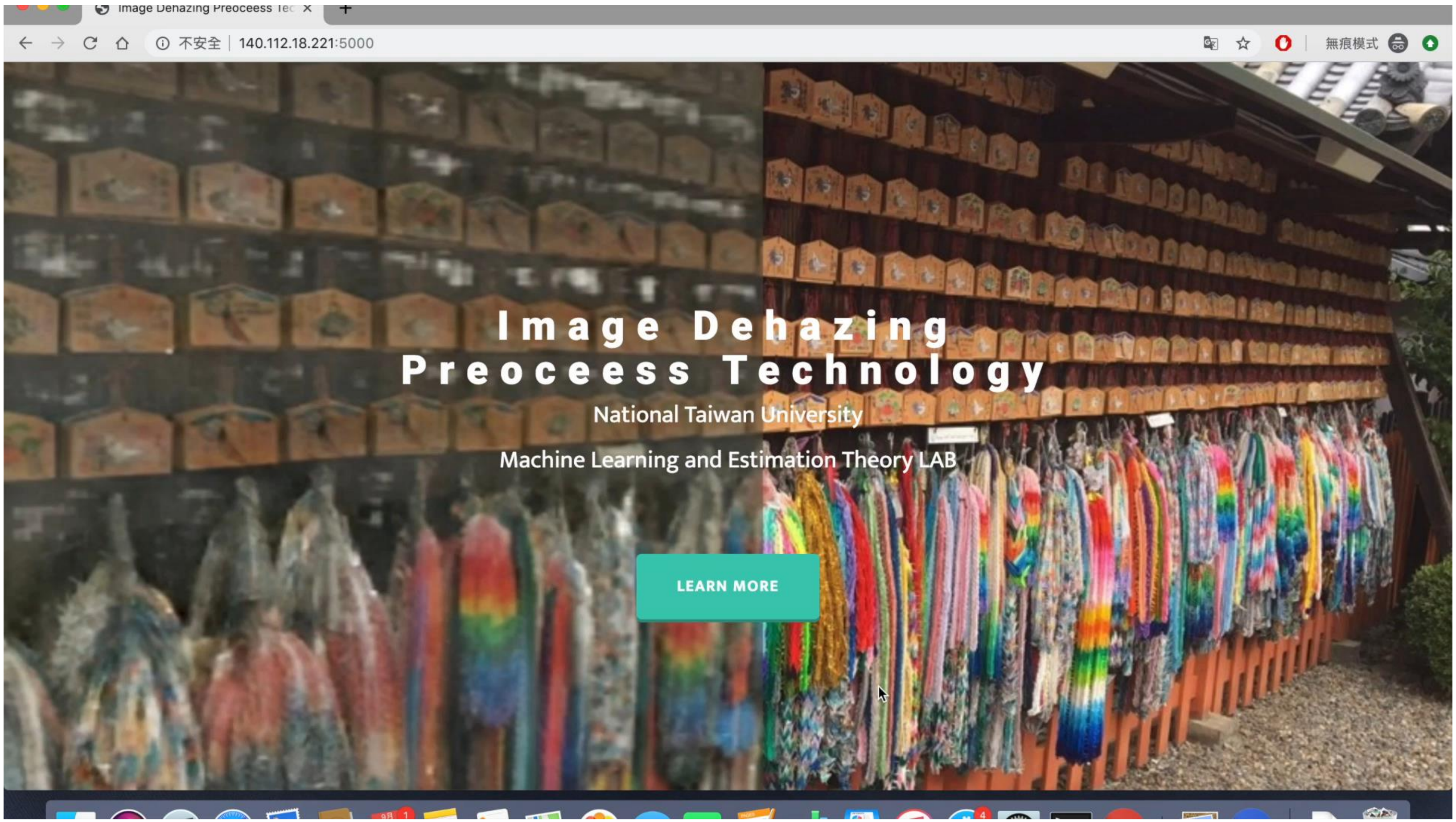
Input DCP [8] AOD-Net [11] DCPDN [28] GFN [19] EPDN [17] Ours GT

Indoor							
	DCP [8]	DehazeNet [5]	AOD-NET [11]	DCPDN [28]	GFN [19]	EPDN [17]	Ours
PSNR	16.62	21.14	19.06	15.85	22.30	25.06	31.24
SSIM	0.8179	0.8472	0.8504	0.8175	0.8800	0.9232	0.9719

Outdoor							
	DCP [8]	DehazeNet [5]	AOD-NET [11]	DCPDN [28]	GFN [19]	EPDN [17]	Ours
PSNR	19.13	22.46	20.29	19.93	21.55	22.57	23.69
SSIM	0.8148	0.8514	0.8765	0.8449	0.8444	0.8630	0.9275

Courtesy of 趙汝晉同學

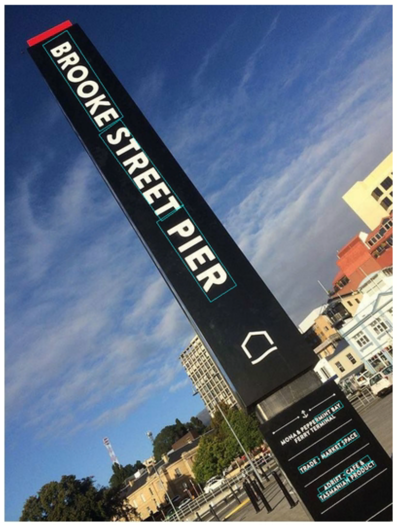
Image Dehazing Demo



Scene Text Detection/Recognition Demo

Scene Text Recognition Demo

瀏覽... 未選擇檔案。 Submit



- image size: 675x900
- cropped images

- 11 text lines:

1. **street, horizontal**
2. **brooke, horizontal**
3. **pier, horizontal**
4. **adrift, horizontal**
5. **carkets, horizontal**
6. **tasmania, horizontal**
7. **iproduce, horizontal**
8. **trade, horizontal**
9. **tspace, horizontal**
10. **cafers, horizontal**
11. **rmintban, horizontal**

JSON

- /static/results/4cdb2aa0-df56-11e9-bc49-8b63b82283f5/result.json

This is a demo for the arbitrarily oriented scene text recognition for both horizontal and vertical text.

Courtesy of 張嘉麟同學

Acknowledgment

- 感謝 Guobiao Mo 發現投影片上的打字錯誤