Lecture 3: CNN

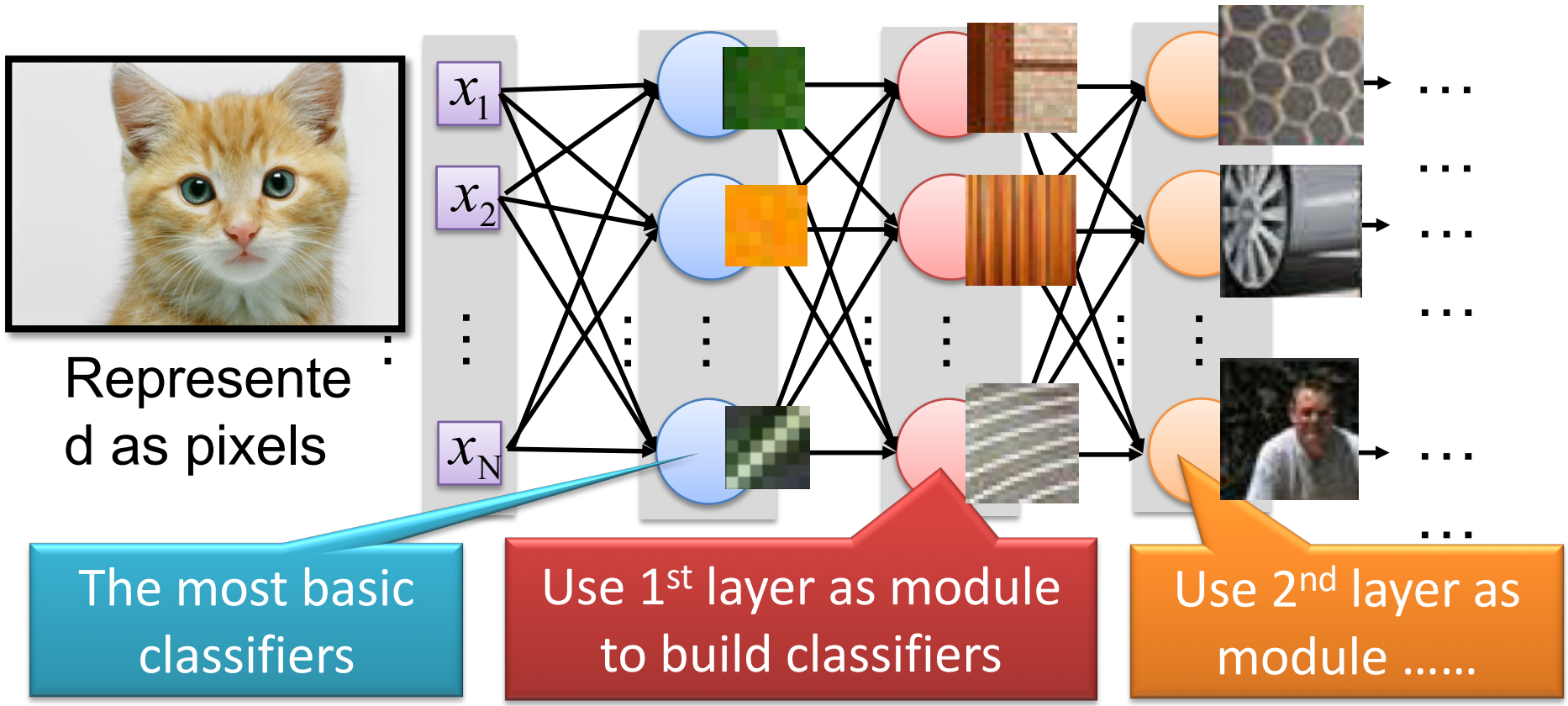
Hong-Han Shuai

ECE, NCTU

Thanks to the slides from Prof. Hung-Yi Lee in NTU.

Why CNN for Image?

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



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

Why CNN for Image

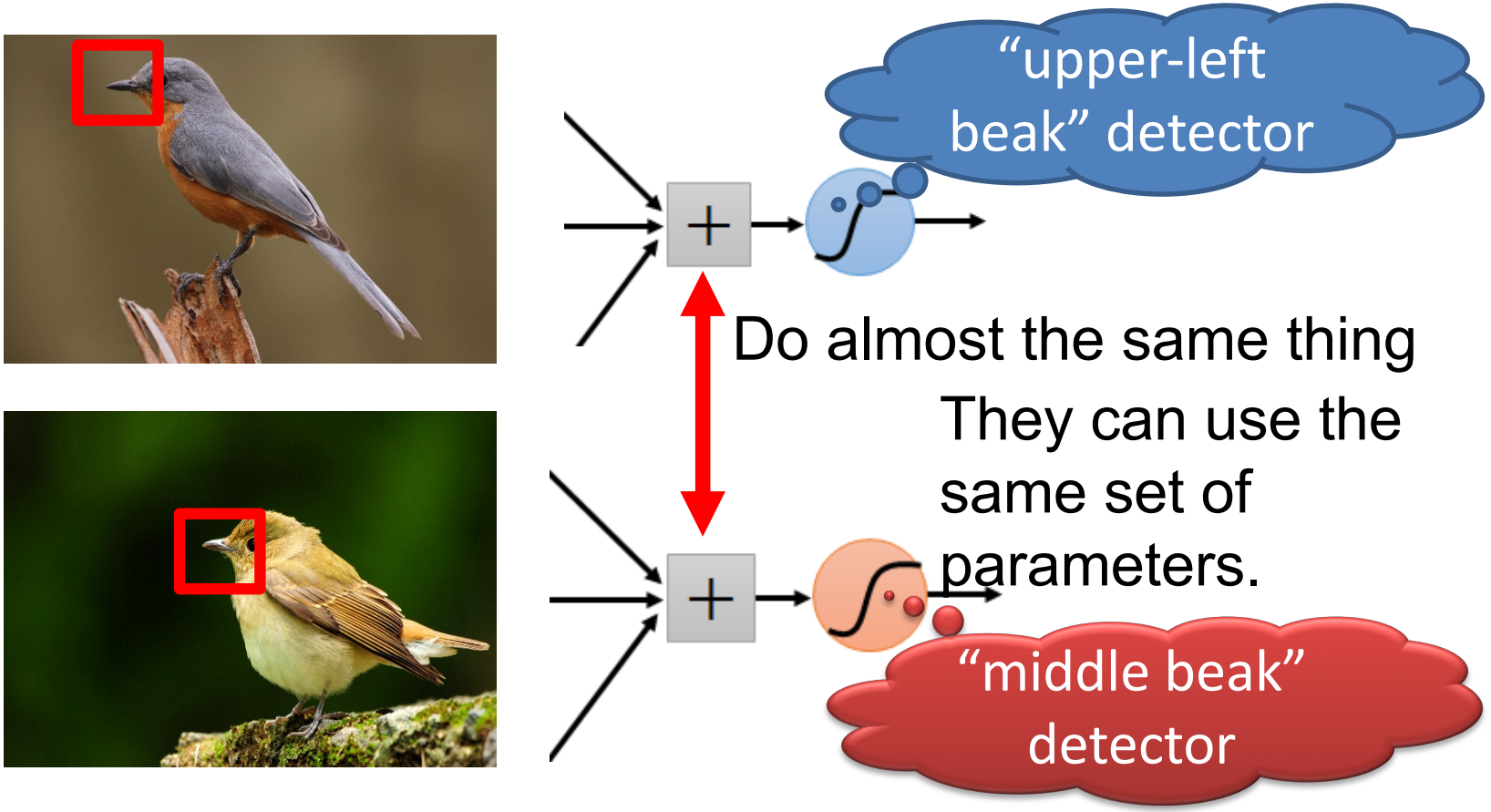
- Some patterns are much smaller than the whole image

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

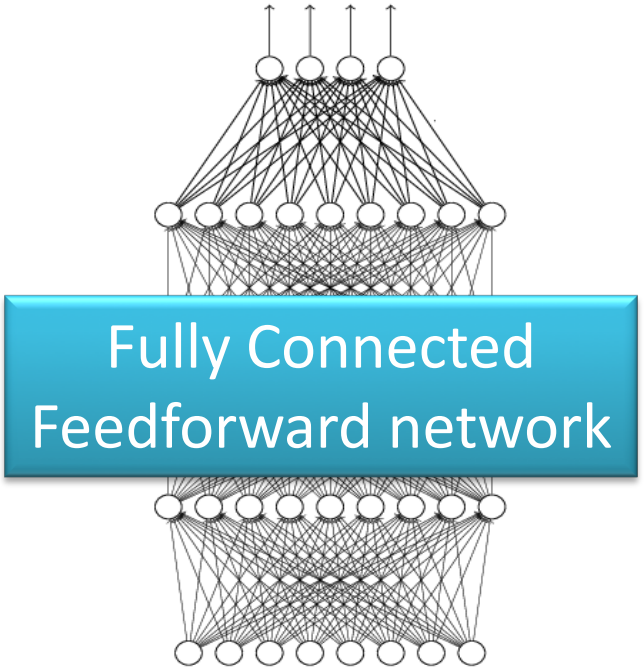
bird

subsampling

We can subsample the pixels to make image smaller

Less parameters for the network to process the image

The whole


Convolution
Max Pooling
Convolution
Max Pooling
Can repeat many times
Flatten
Fully Connected Feedforward network
cat dog ……

The whole

Property 1

- Some patterns are much smaller than the whole

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

Convolution

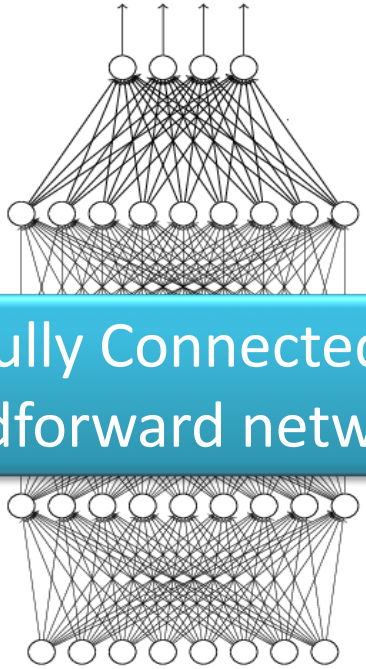
Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten



Fully Connected Feedforward network

cat dog ……

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 – Convol

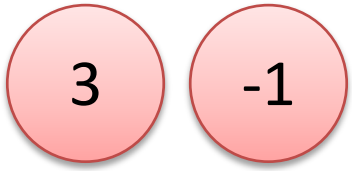
1	-1	-1
-1	1	-1
-1	-1	1

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



CNN – Convol

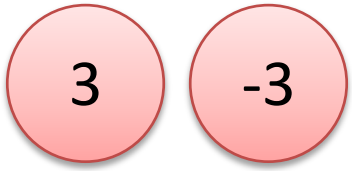
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

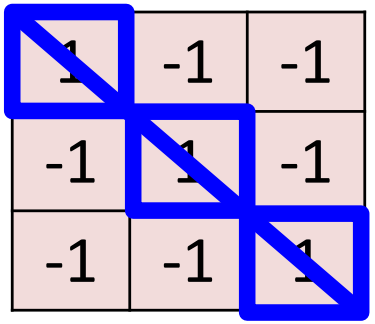
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



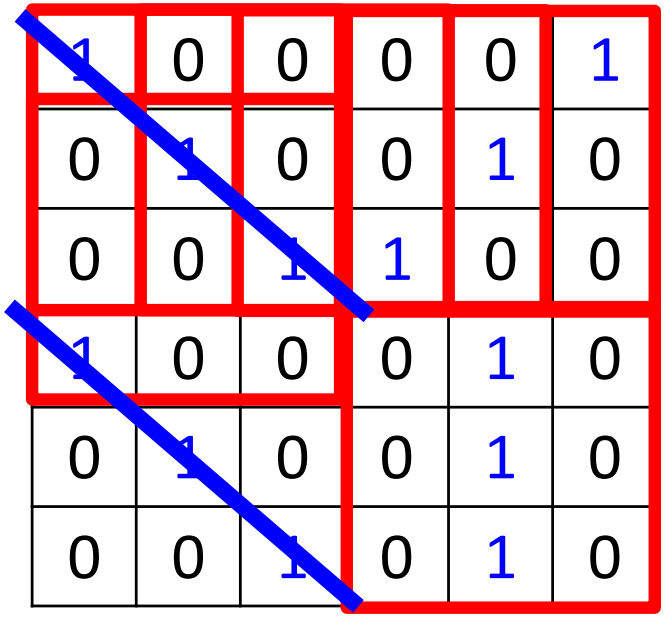
We set stride=1 below

CNN – Convol

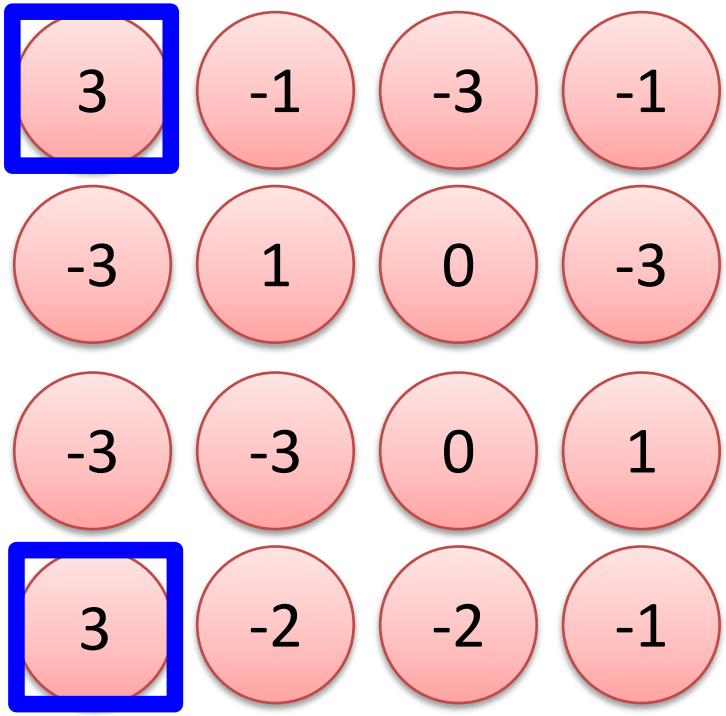


Filter 1

stride=1



6 x 6 image



Property 2

CNN – Convolu

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

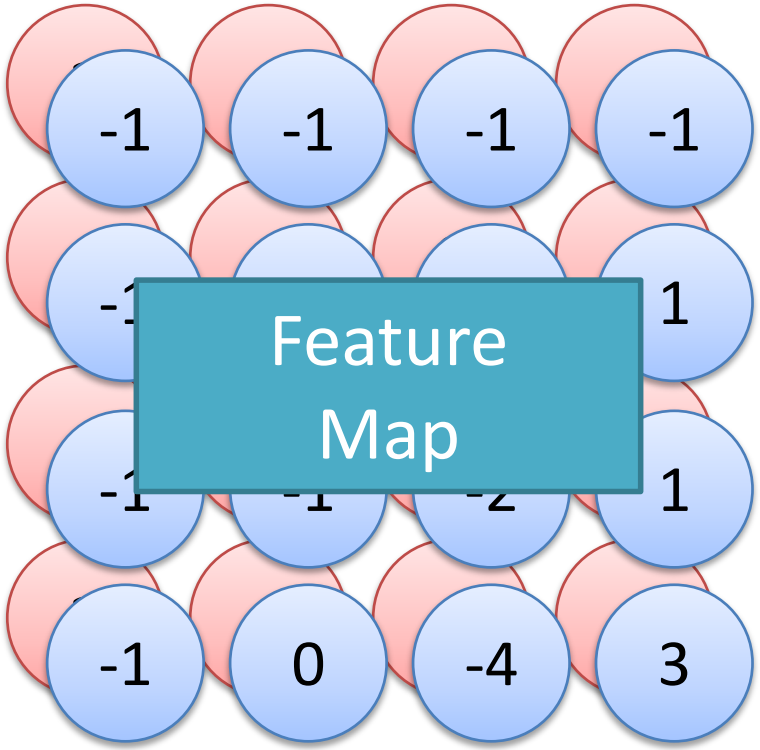
Filter 2

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

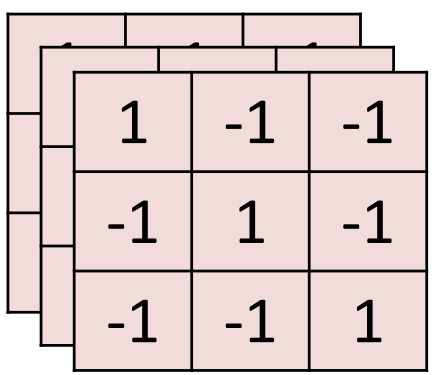
6 x 6 image

Do the same process for every filter

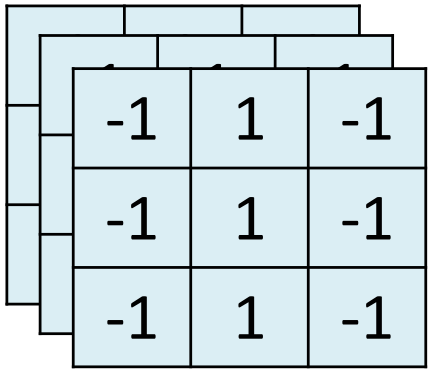


4 x 4 image

CNN – Colorful image

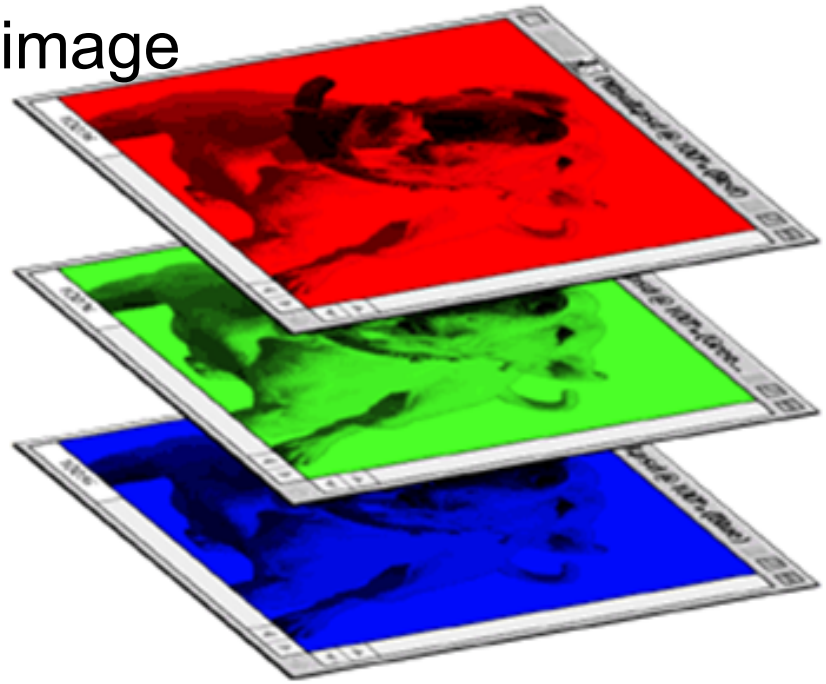


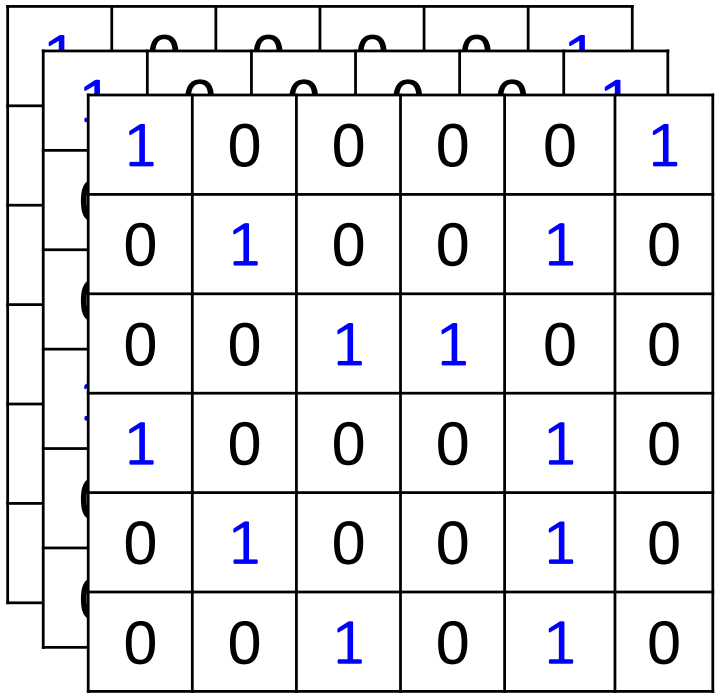
Filter 1



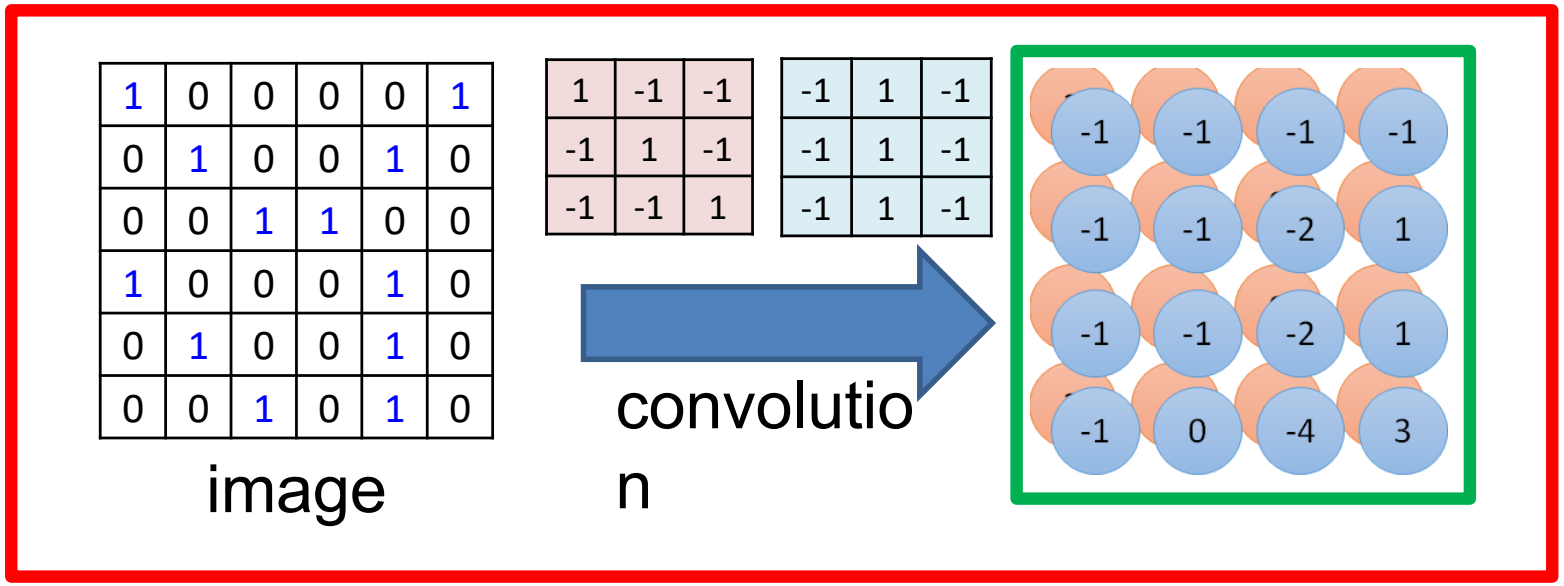
Filter 2

Colorful image

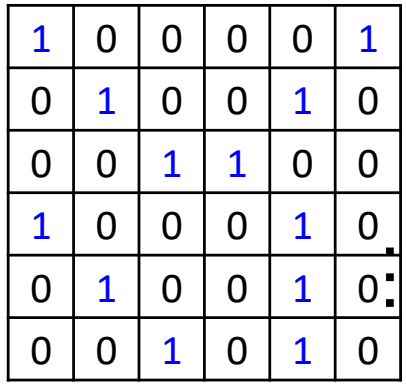


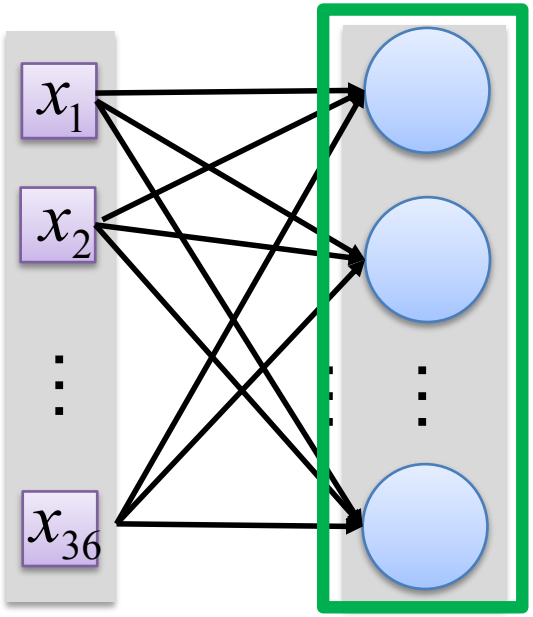


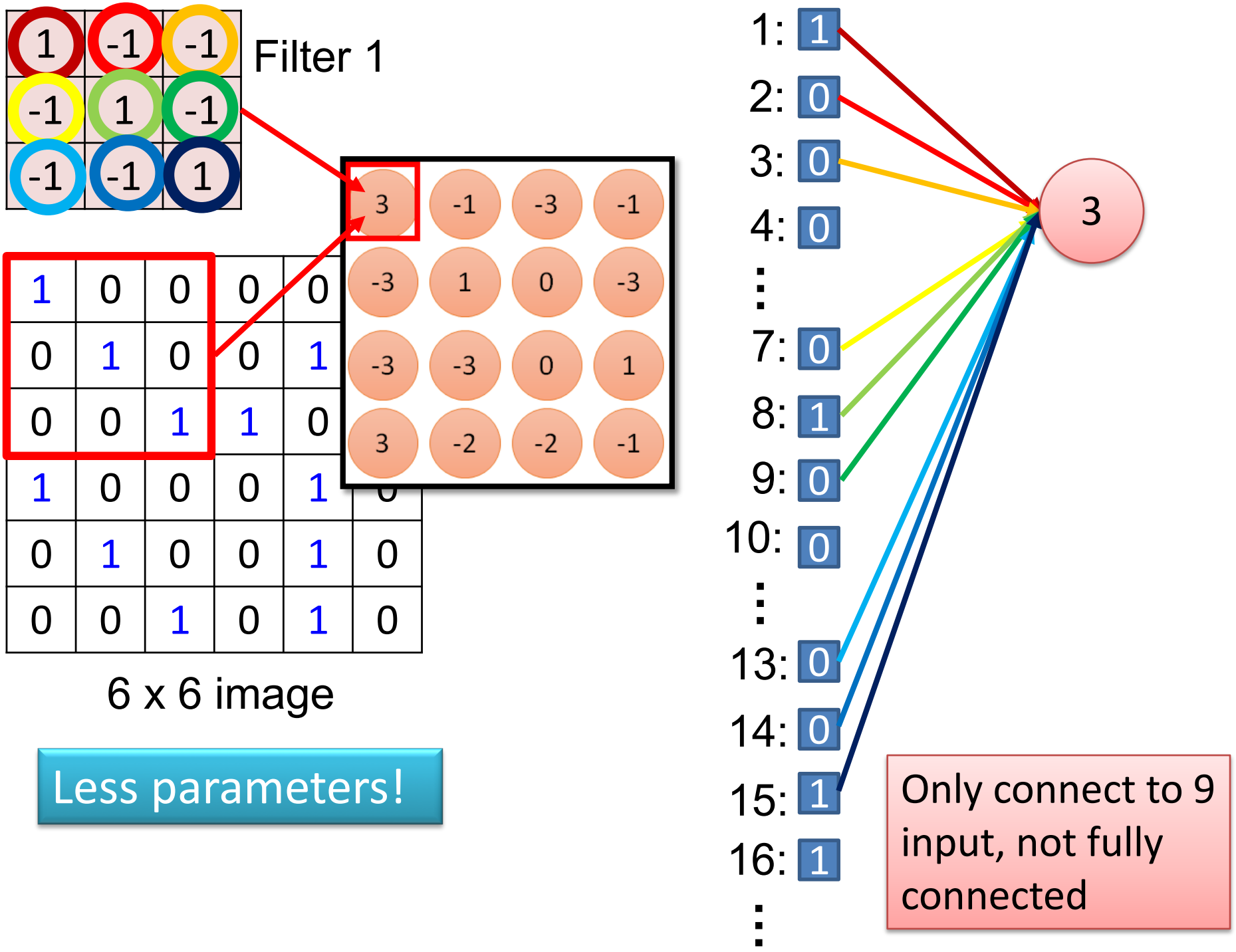
Convolution v.s. Fully Connected

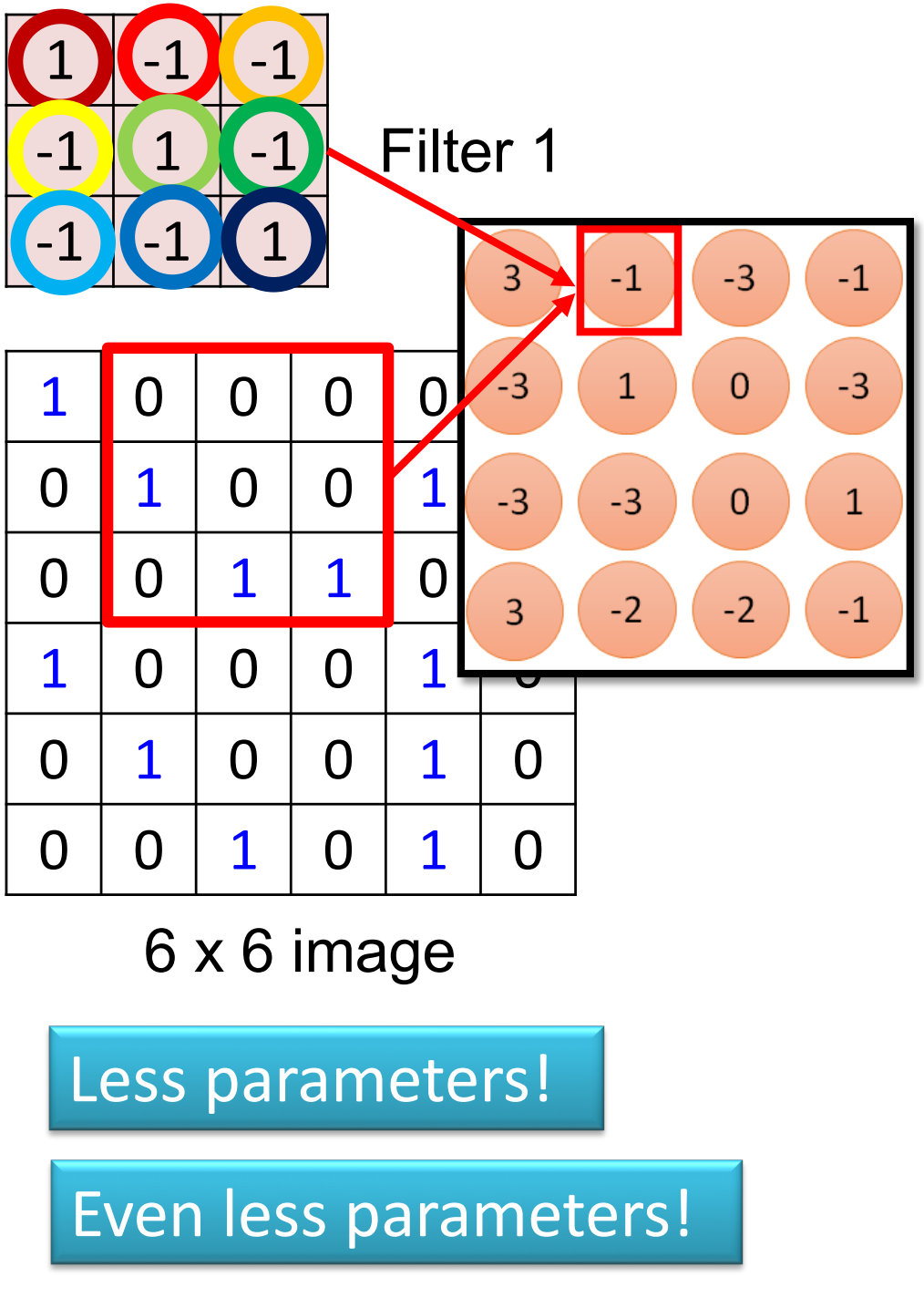


Fully-connected

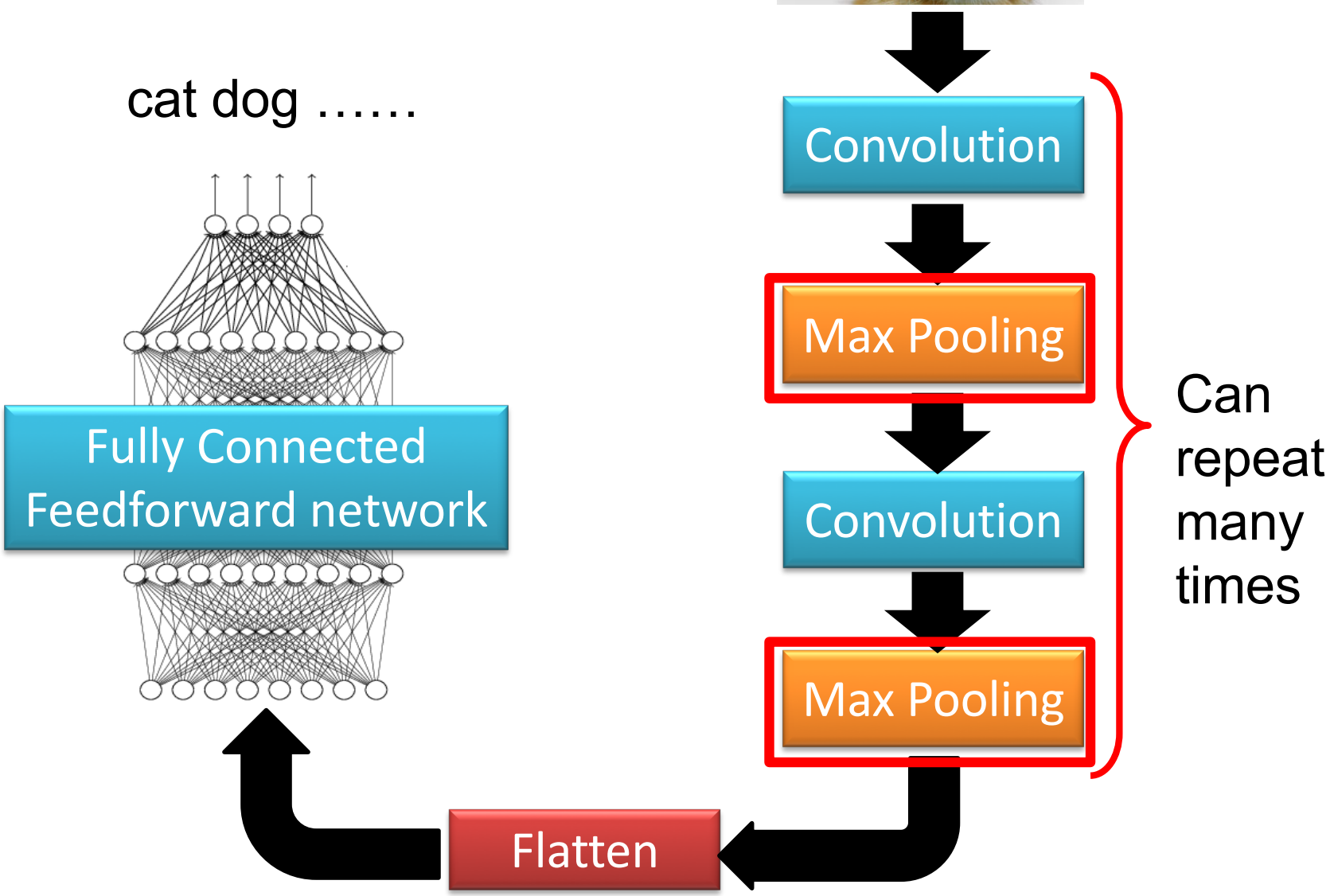





1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
Less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Only connect to 9 input, not fully connected


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


cat dog ……
Convolution
Max Pooling
Convolution
Max Pooling
Can repeat many times
Flatten
Fully Connected Feedforward network

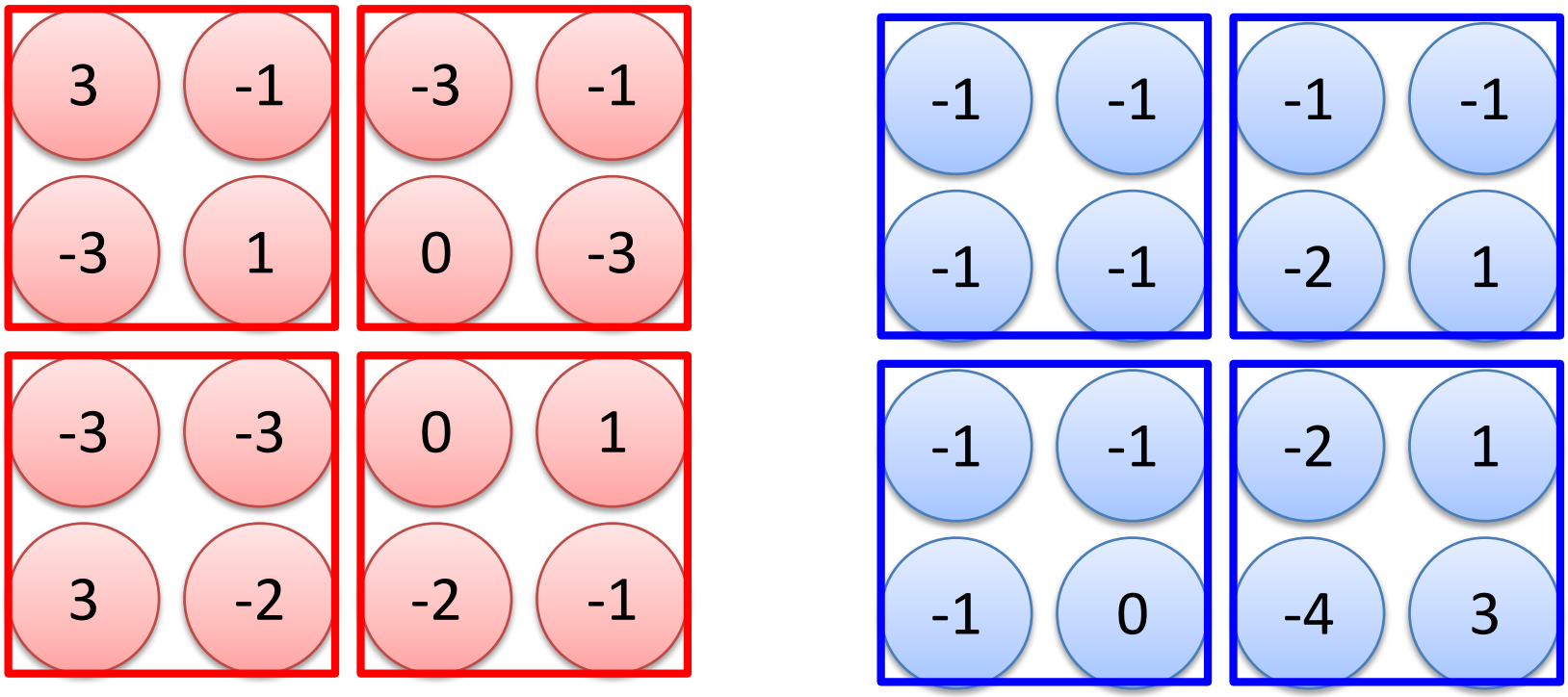
CNN – Max Pooling

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

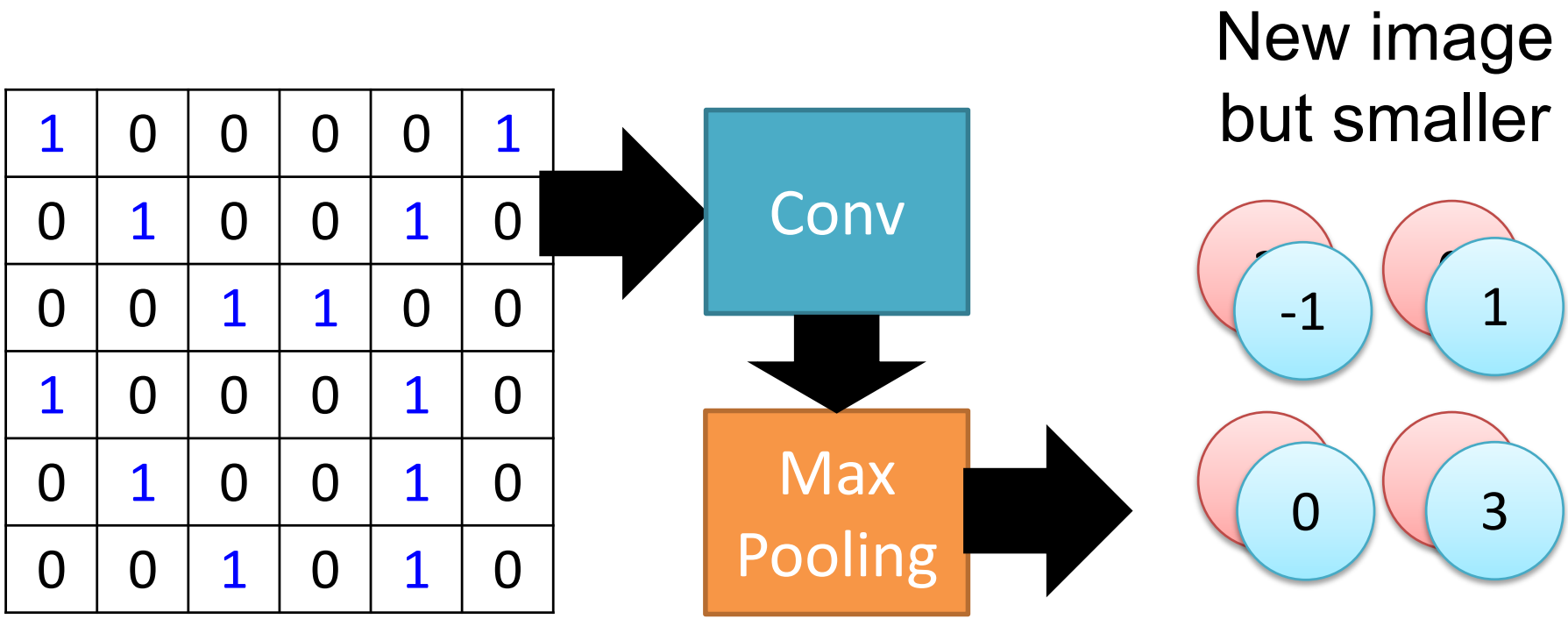
Filter 1

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

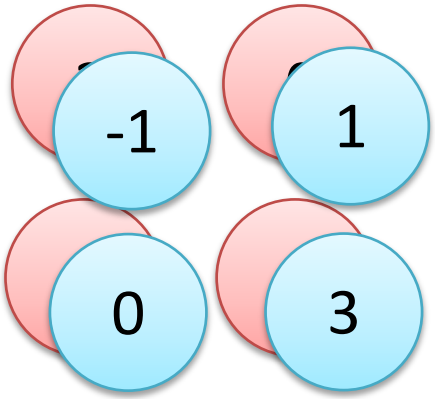
Filter 2



CNN – Max Pooling



The whole



A new image

Convolution

Max Pooling

Convolution

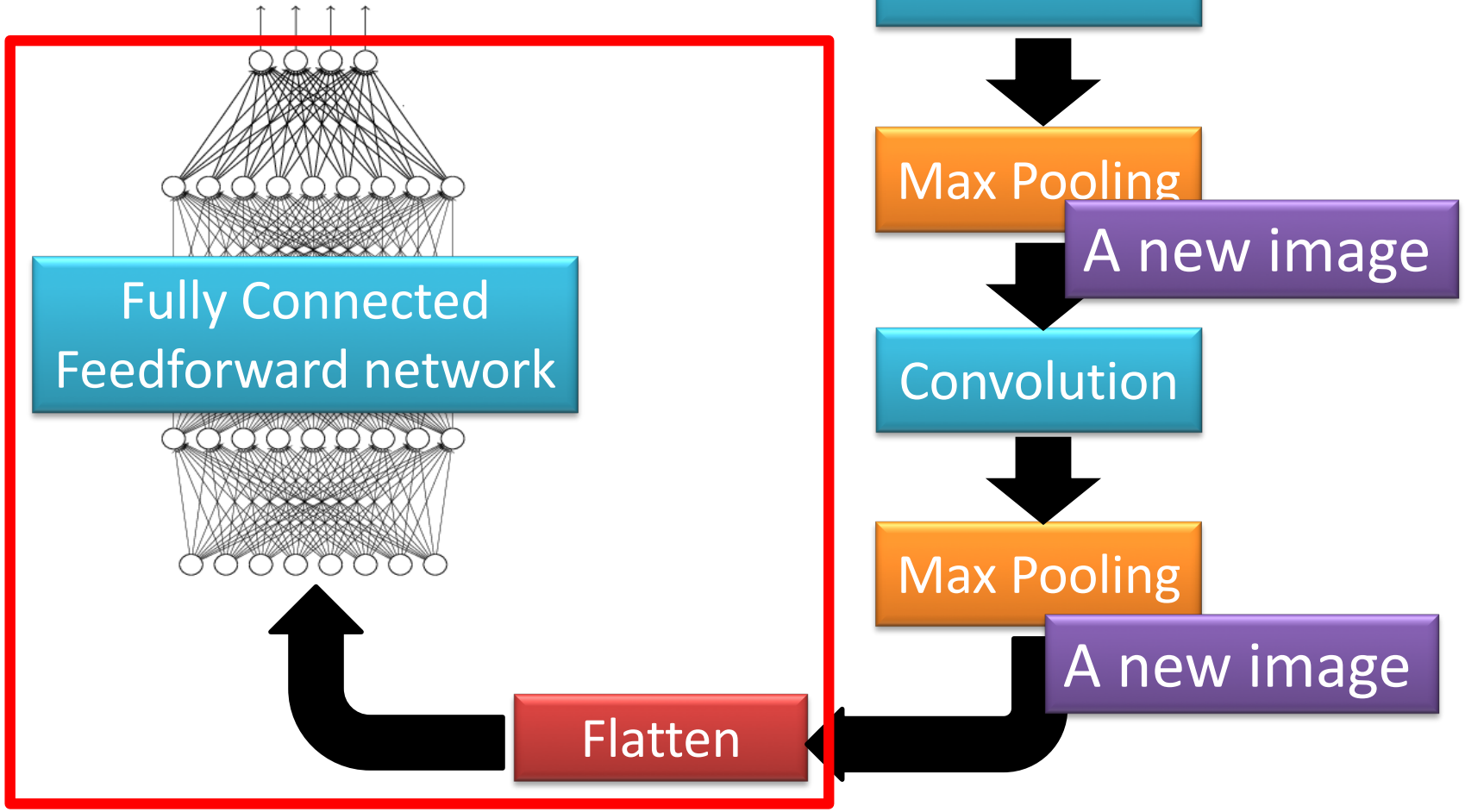
Max Pooling

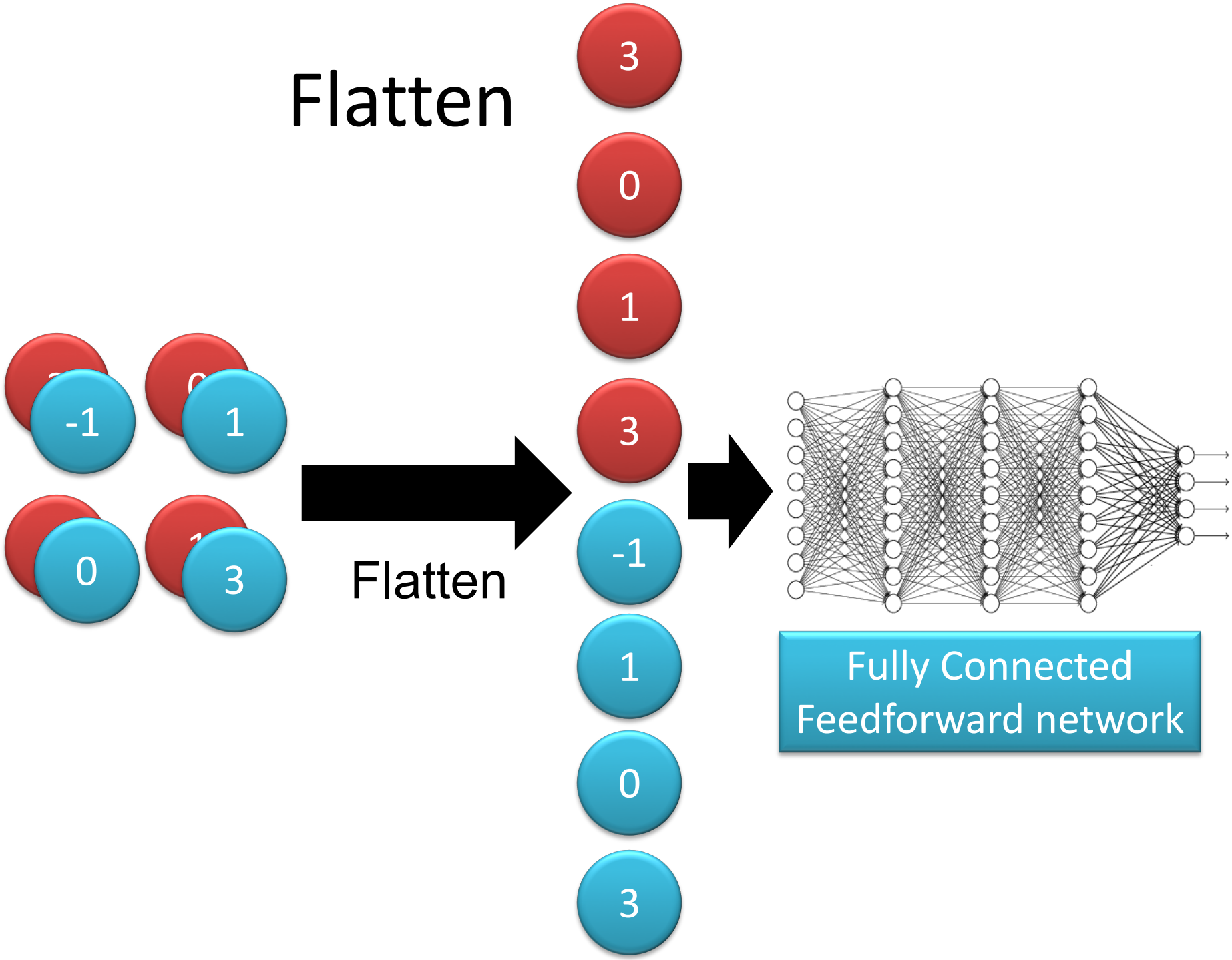
Can repeat many times

Smaller than the original image

The number of the channel is the number of filters

The whole

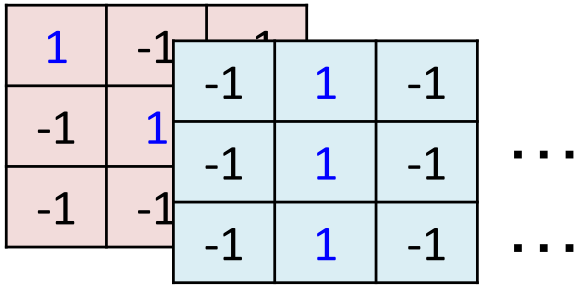

cat dog ……
Fully Connected Feedforward network
Flatten
Convolution
Max Pooling
A new image
Convolution
Max Pooling
A new image


Flatten
3
0
1
3
-1
1
1
0
3
-1
1
0
3
Flatten
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=(1,28,28) ) )
```



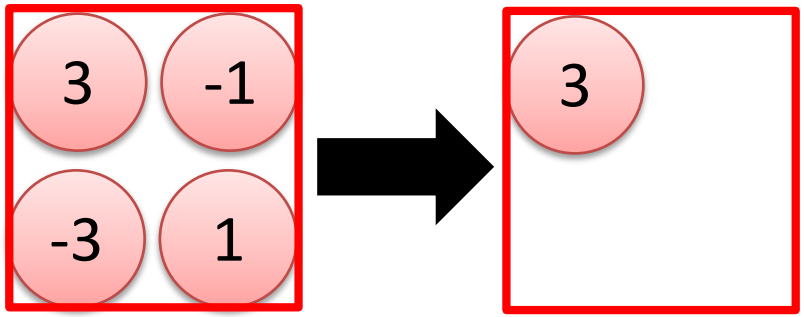
There are 25 3x3 filters.

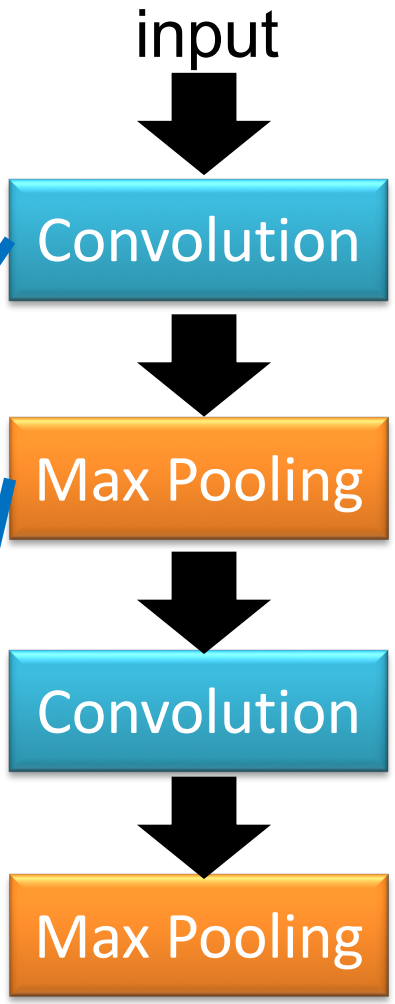
Input_shape = (1 , 28 , 28)

1: black/weight, 3: RGB

28 x 28 pixels

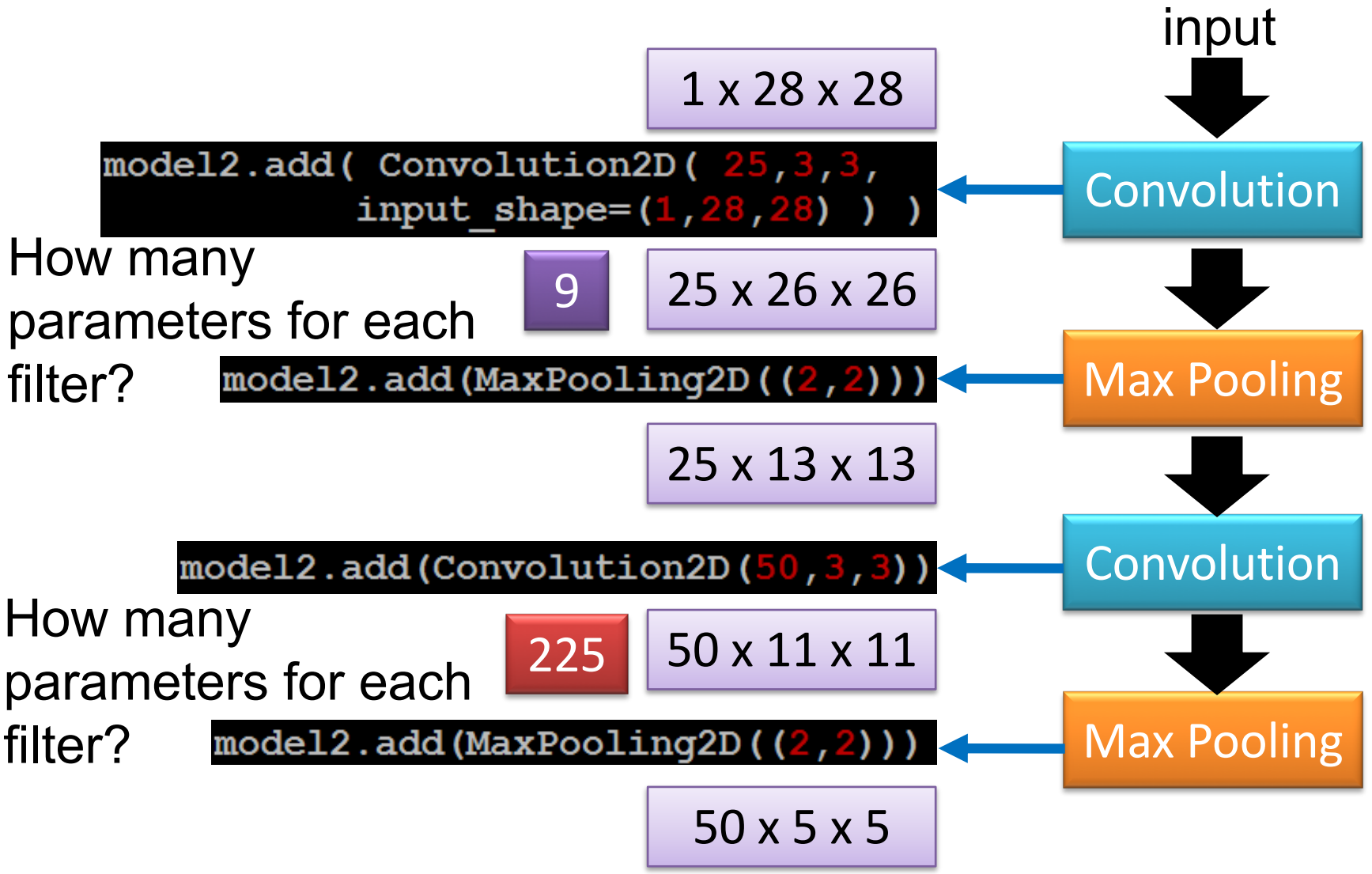
```
model2.add(MaxPooling2D((2,2)))
```





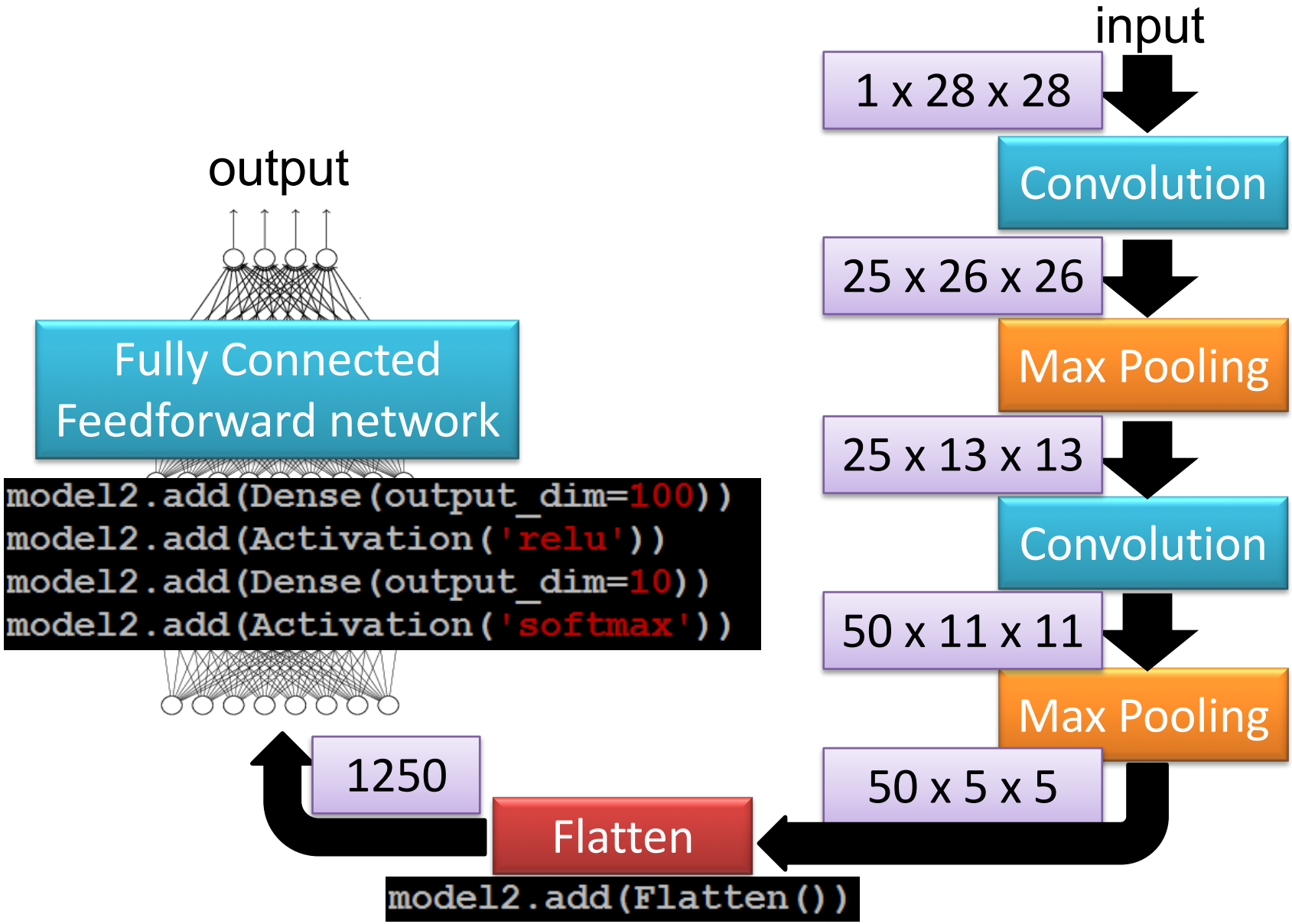
CNN in Keras

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



CNN in Keras

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



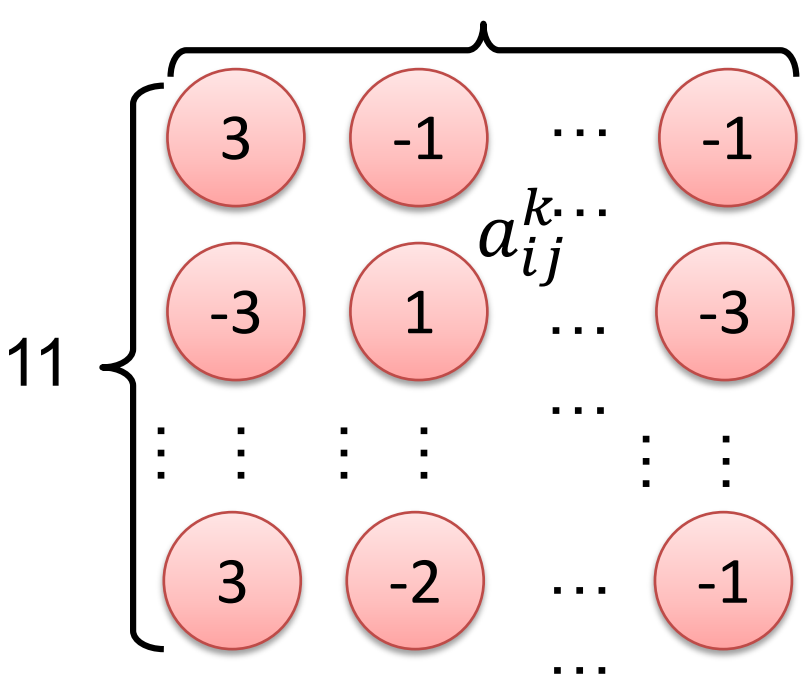
What does CNN learn?

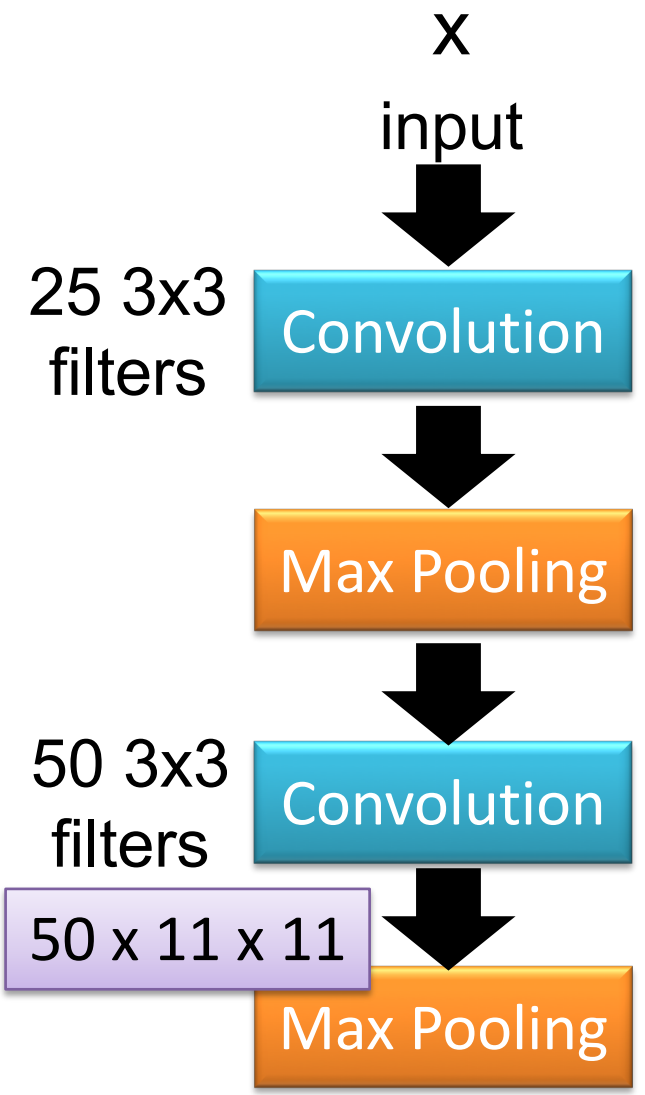
The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter:

$$a^k = \sum_{i=1}^{11}\sum_{j=1}^{11} a_{ij}^k$$

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



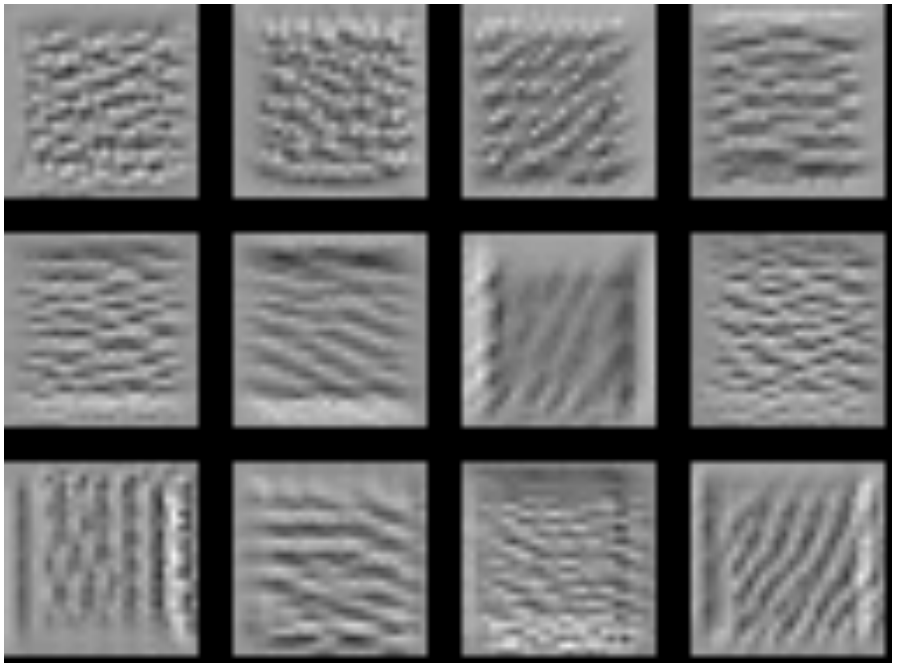


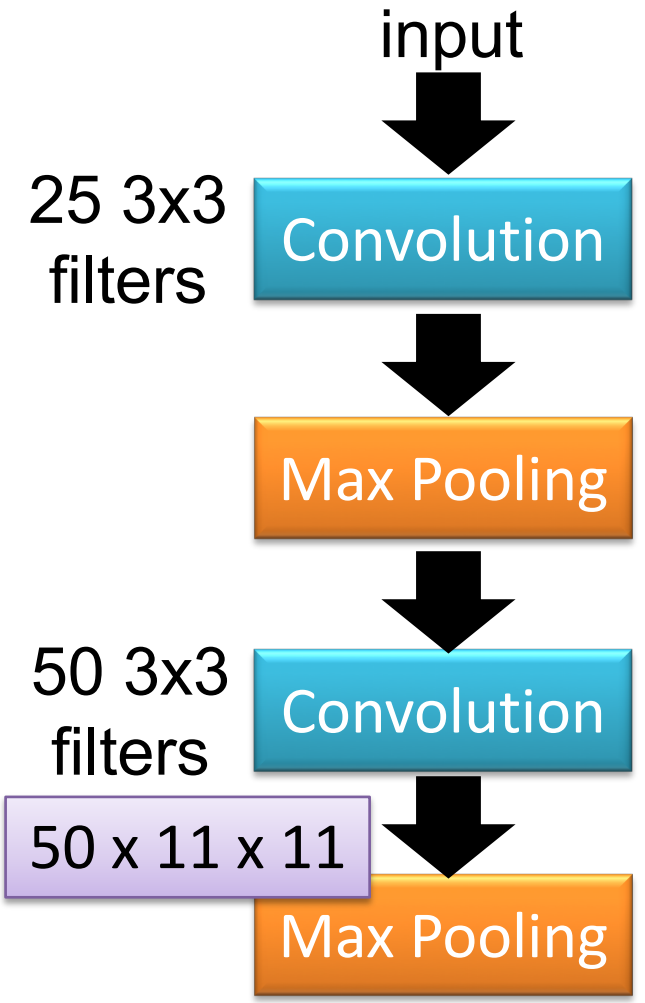
What does CNN learn?

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

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

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



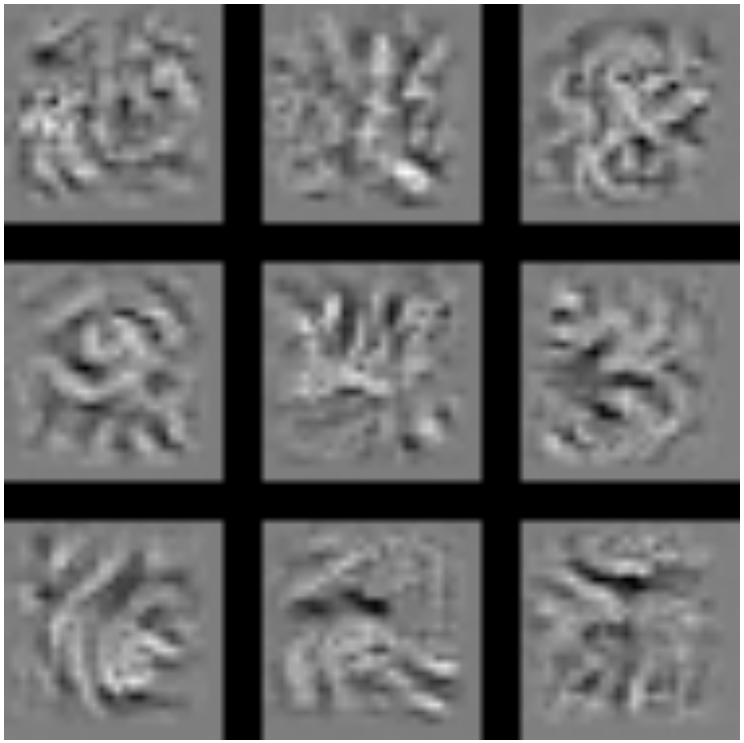


For each filter

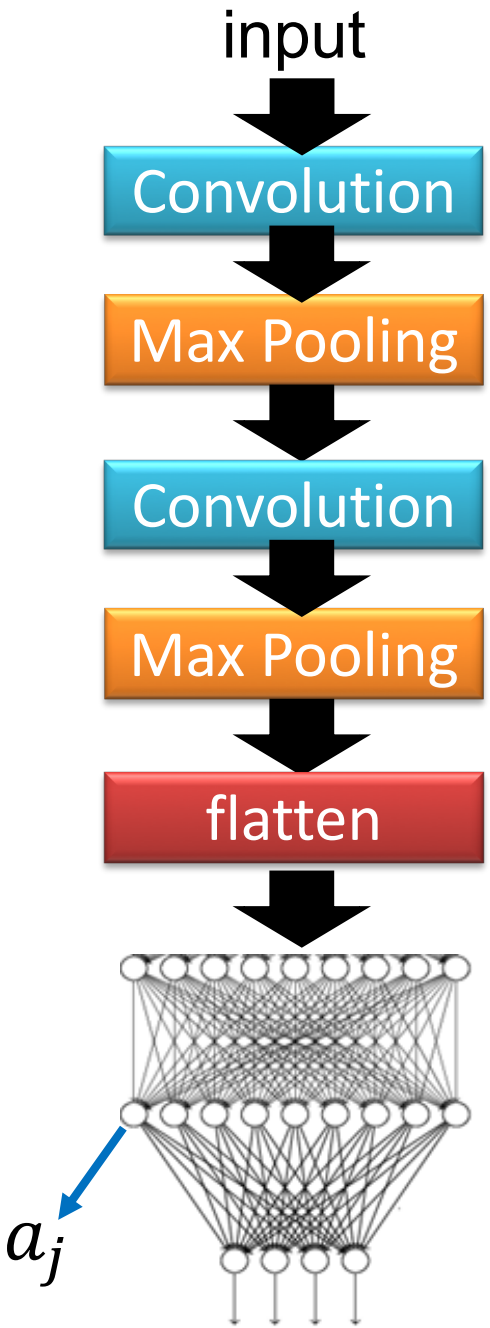
What does CNN learn?

Find an image maximizing the output of neuron:

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



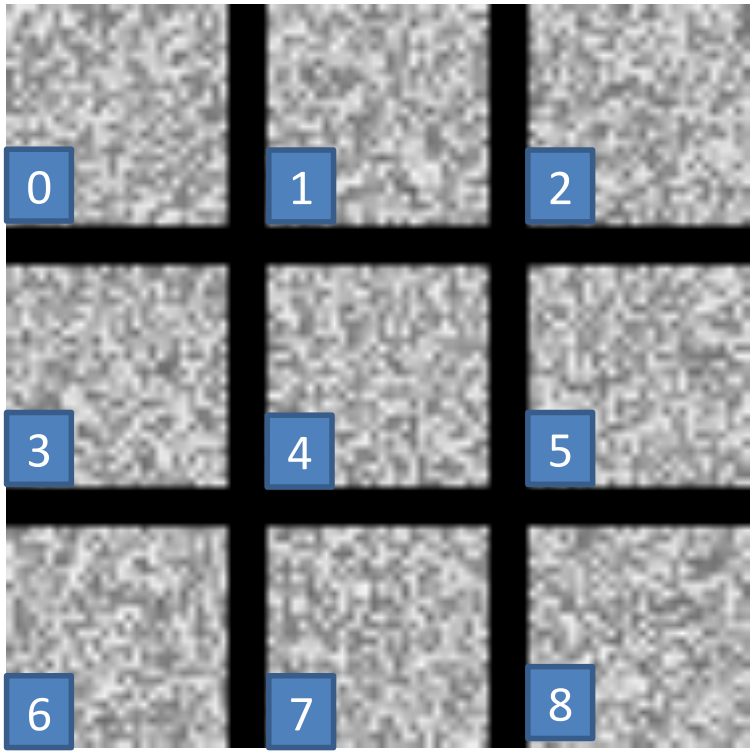
Each figure corresponds to a neuron

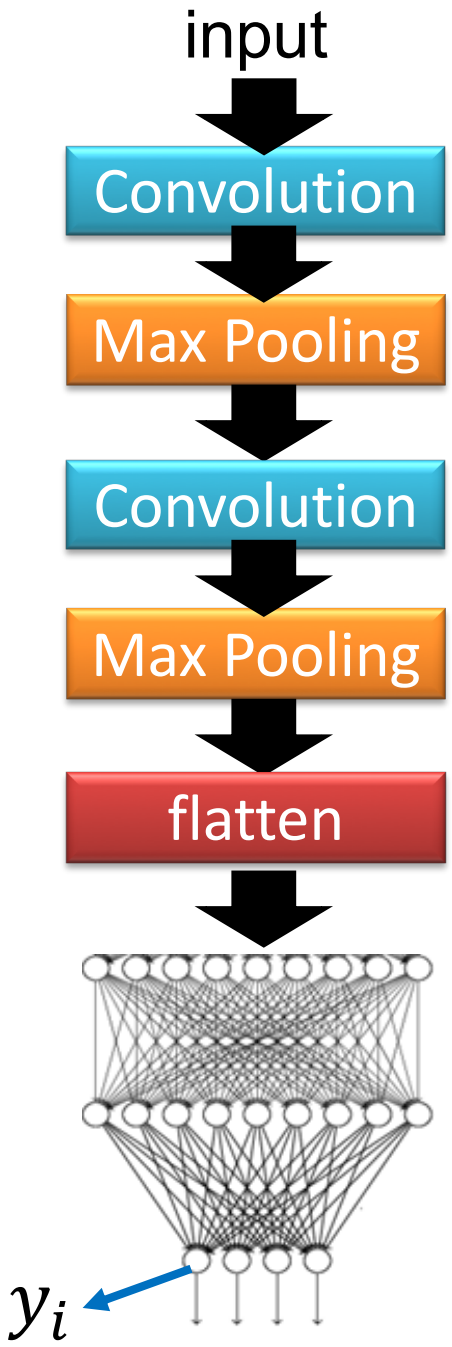


What does CNN learn?

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

Can we see digits?





Deep Neural Networks are Easily Fooled

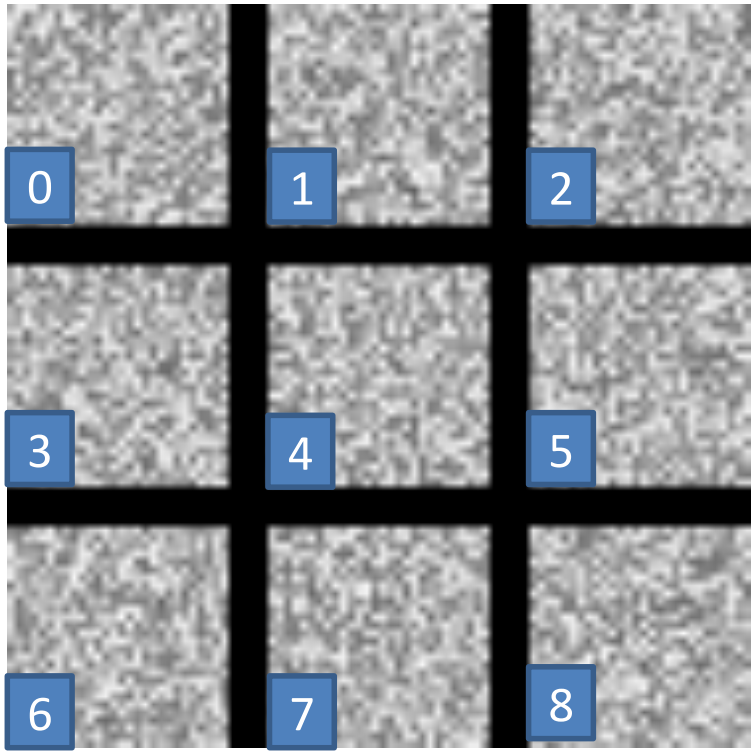
https://www.youtube.com/watch?v=M2IebCN9Ht4

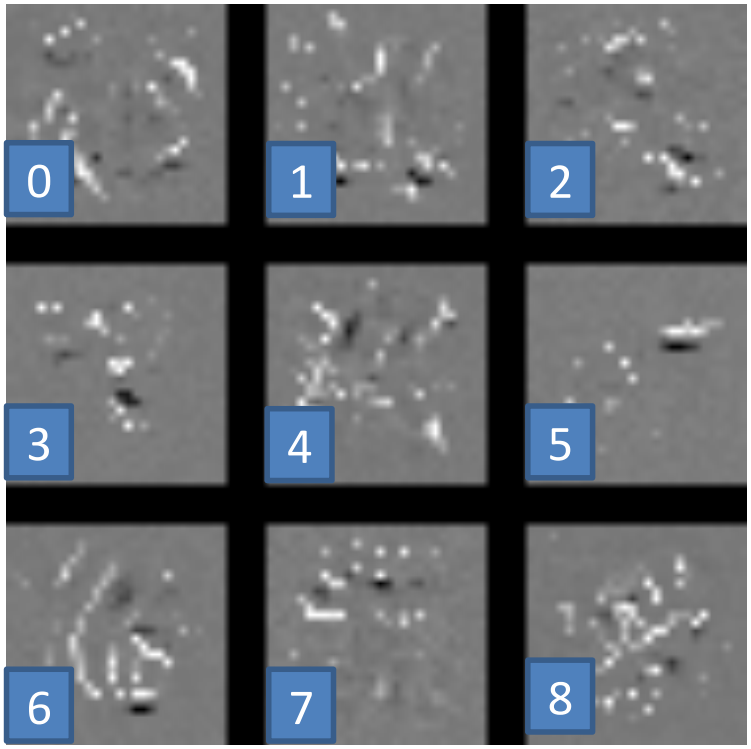
What does CNN learn?

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

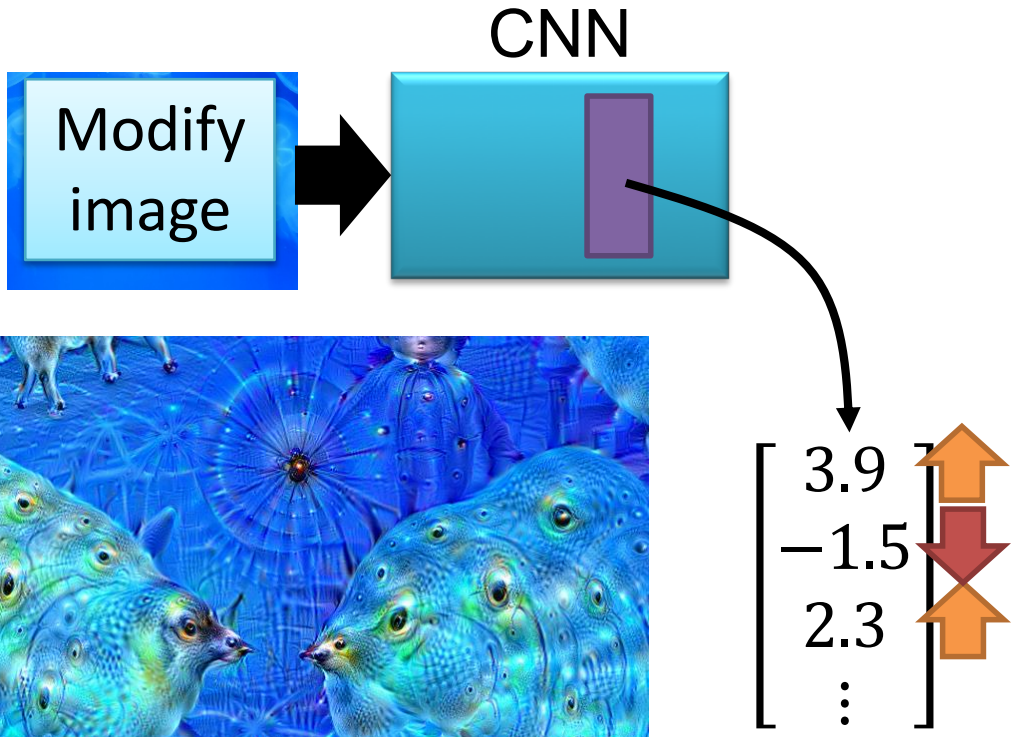
$$x^* = arg\max_x \left(y^i + \sum_{i,j} |x_{ij}| \right)$$

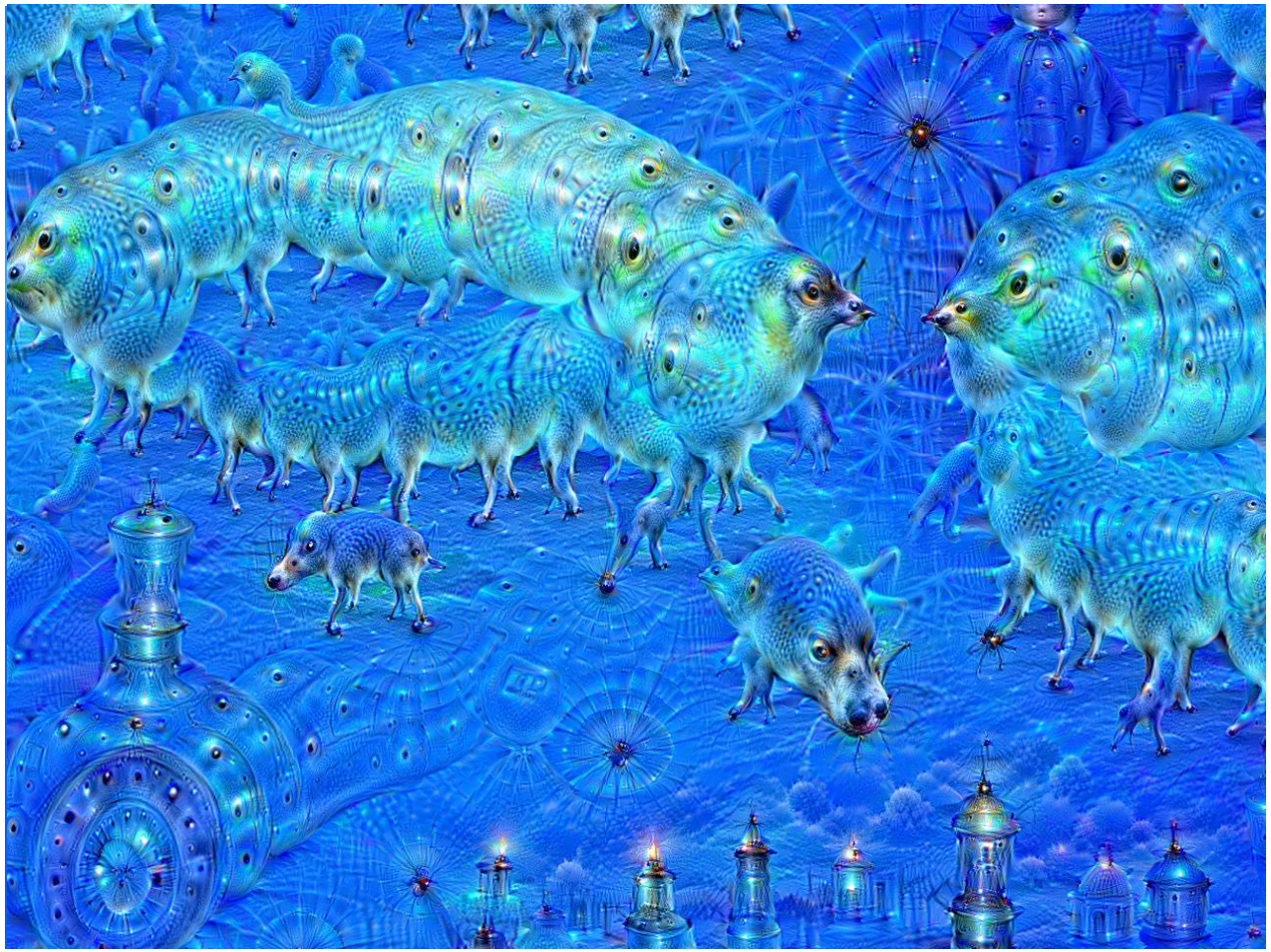
Over all pixel values





Deepdream





- https://deepdreamgenerator.com/

Demo

Deep Style

- Given a photo, make its style like famous paintings

Deep Style

CNN

CNN

content

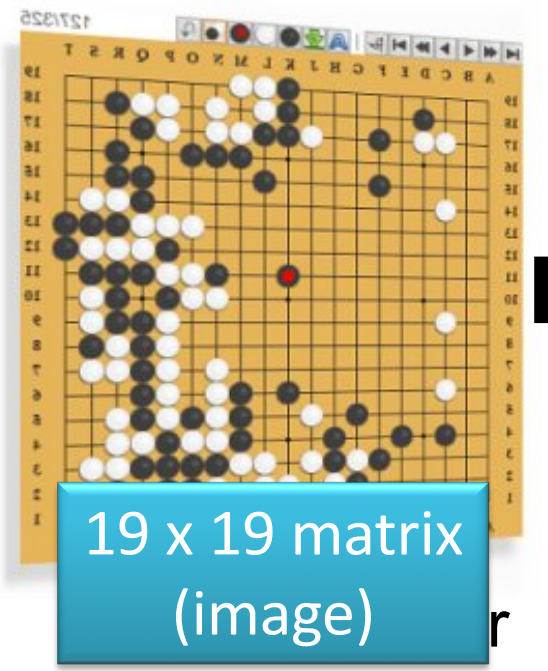
style

CNN

A Neural Algorithm of Artistic Style

https://arxiv.org/abs/1508.06576

More Application: Playing Go



19 x 19 matrix (image)

Network

Next move (19 x 19 positions)

19 x 19 vector

Black: 1

white: -1

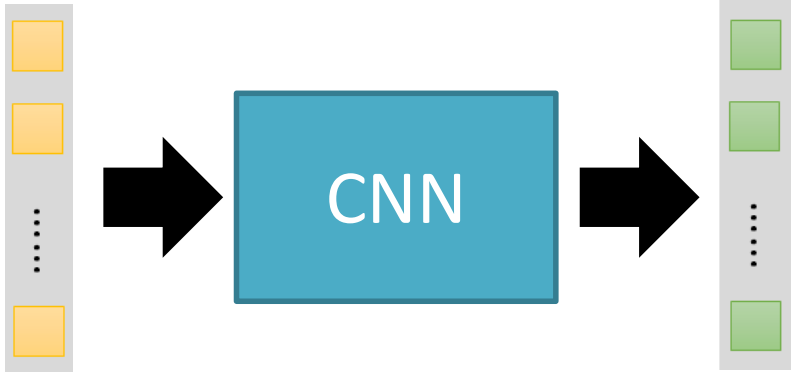
none: 0

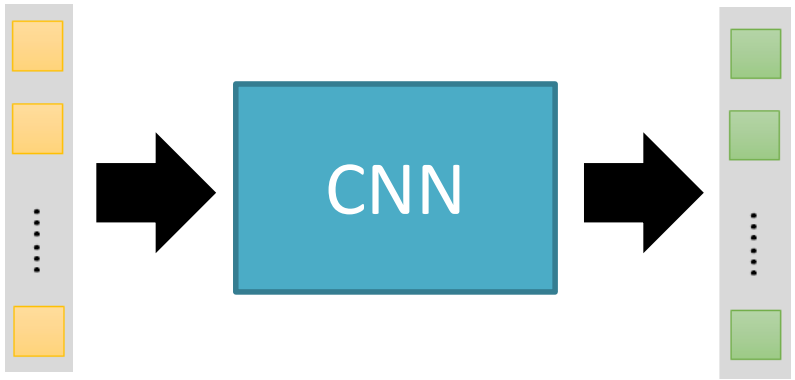
Fully-connected feedforward network can be used

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 …





Target:
“天元” = 1
else = 0

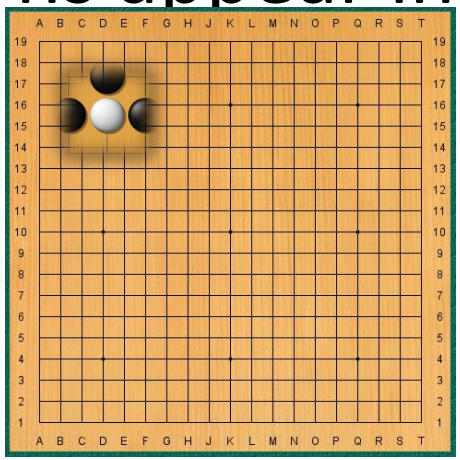
Target:
“五之 5” = 1
else = 0

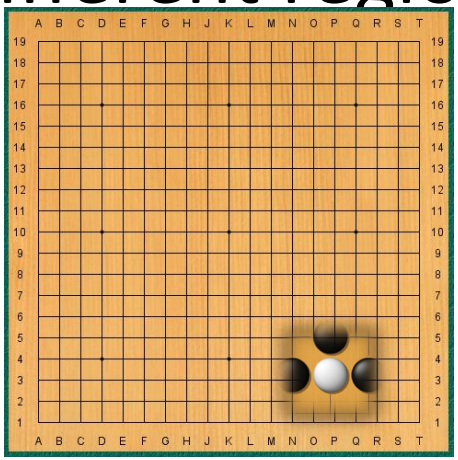
Why CNN for playing Go?

- 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.





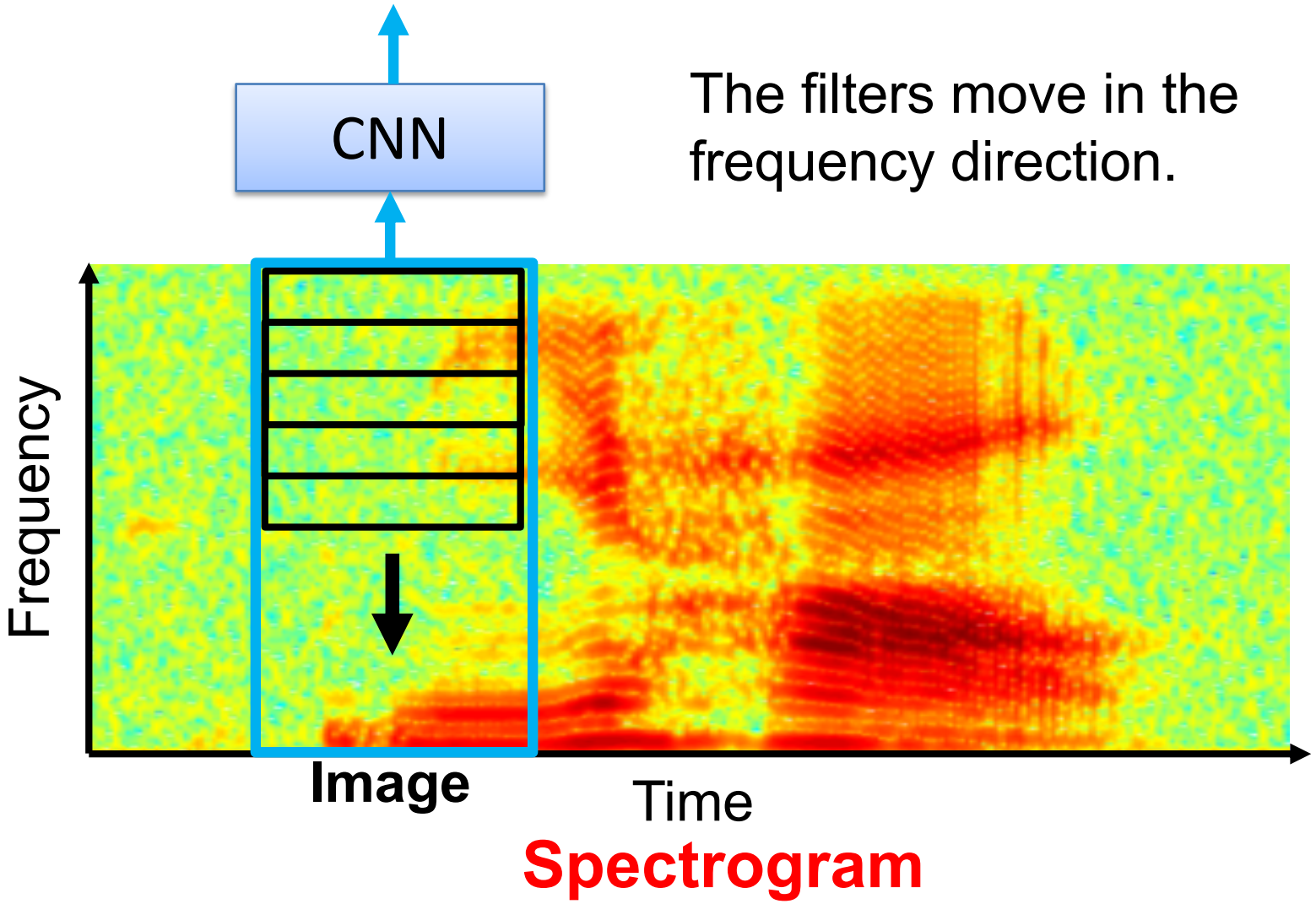
Why CNN for playing Go?

- 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 $19 \times 19 \times 48$ 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 stride 1, with a different bias for each position, and applies a softmax function. 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: Speech



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

Project Killer

Speaker: Hong-Han Shuai

Keras

- Keras Application
- https://keras.io/applications/
- Xception
- VGG16, VGG19
- ResNet50, ResNetV2
- InceptionV3, Inception
- MobileNet, MobileNetV2
- DenseNet
- NASNet

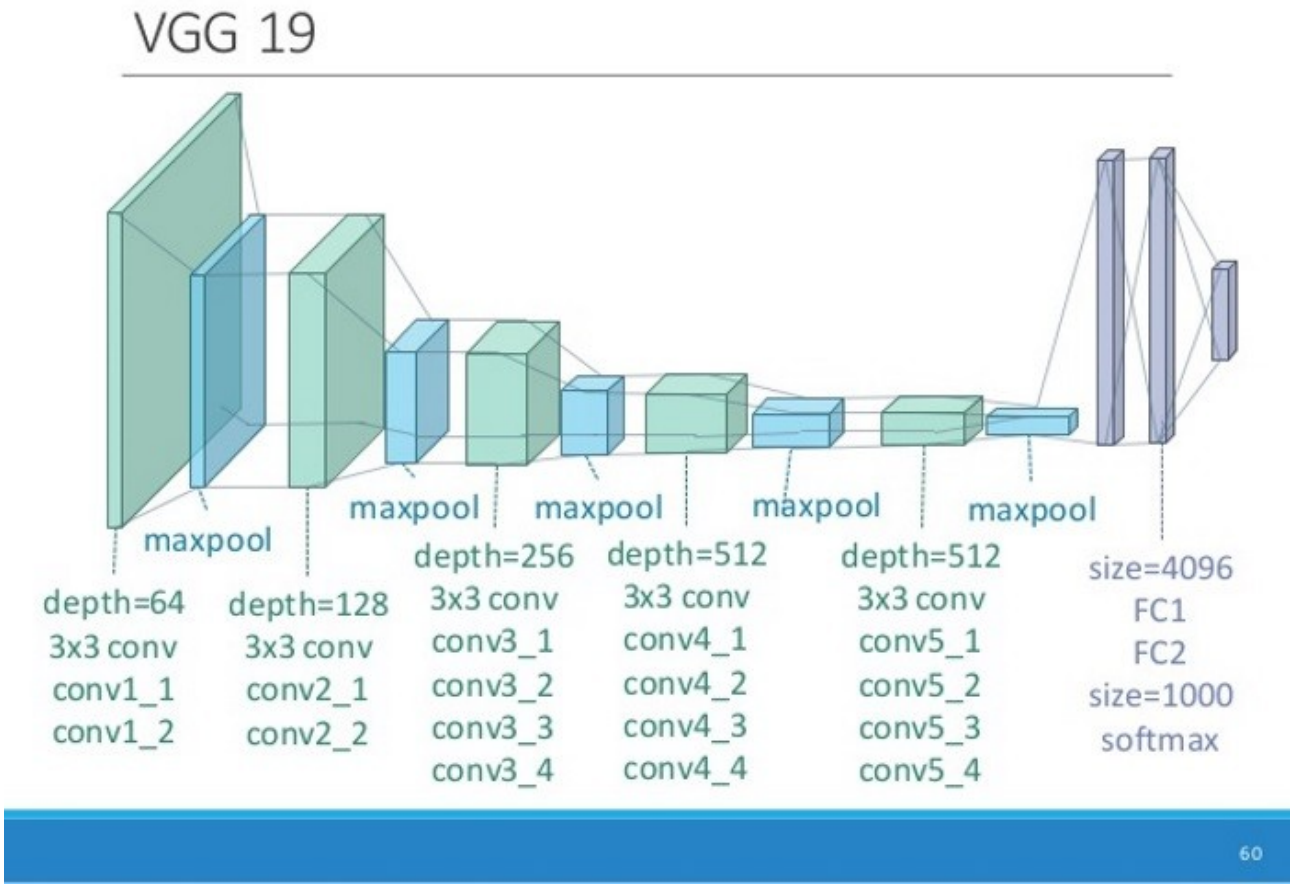
Performance

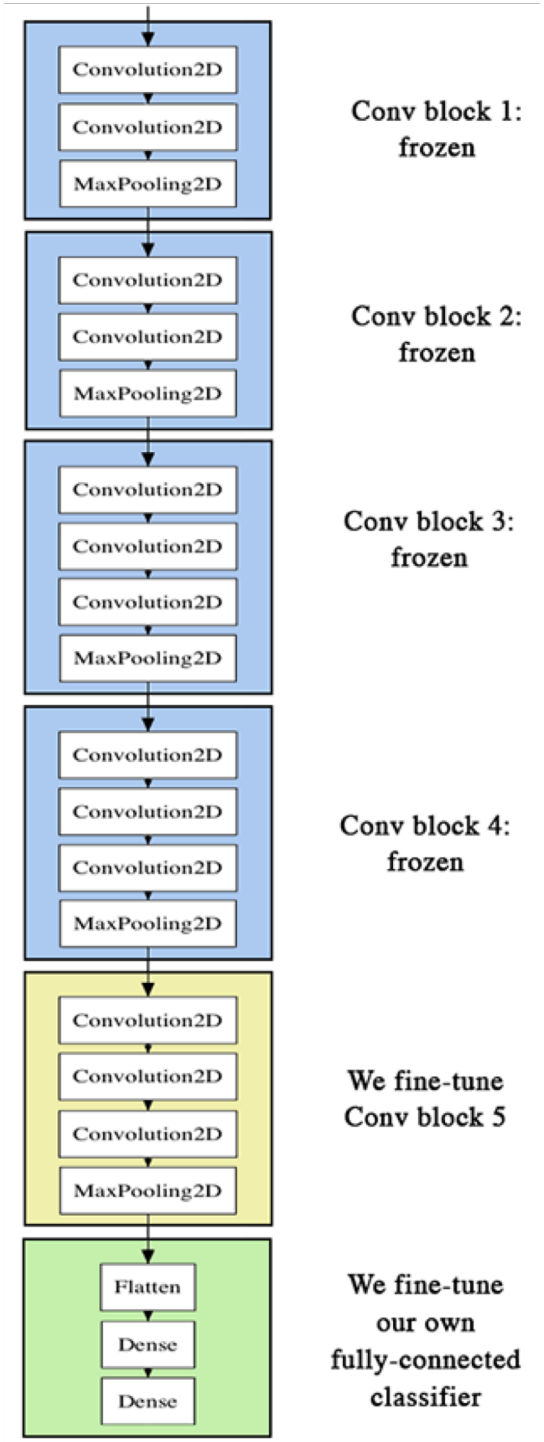
- All compatible to Theano and TensorFlow
  Except for Xception and MobileNet

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88

VGG

- Generic Solution for 1000 Classes





Transfer learning

- For example, you could transfer image recognition knowledge from a cat recognition app to a radiology diagnosis.
- Implementing transfer learning involves retraining the last few layers of the network used for a similar application domain with much more data.

Keras Application- VGG16

- model = VGG16(weights='imagenet', include_top=True)

model in “c:<使用者>.keras\models\”

- Install Graphviz, pydot, and graphviz
- ```
  from keras.utils import plot_model
  plot_model(model,
  to_file='model.png')
  ```
  ```

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input, decode_predictio
ns
import numpy as np

# include_top=True，表示會載入完整的 VGG16 模型，包括加在最後3層的卷積層
# include_top=False，表示會載入 VGG16 的模型，不包括加在最後3層的卷積層，通常是
取得 Features
# 若下載失敗，請先刪除 c:\<使用者>\.keras\models\vgg16_weights_tf_dim_orderi
ng_tf_kernels.h5
model = VGG16(weights='imagenet', include_top=False)

# Input：要辨識的影像
img_path = 'tiger.jpg'
#img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

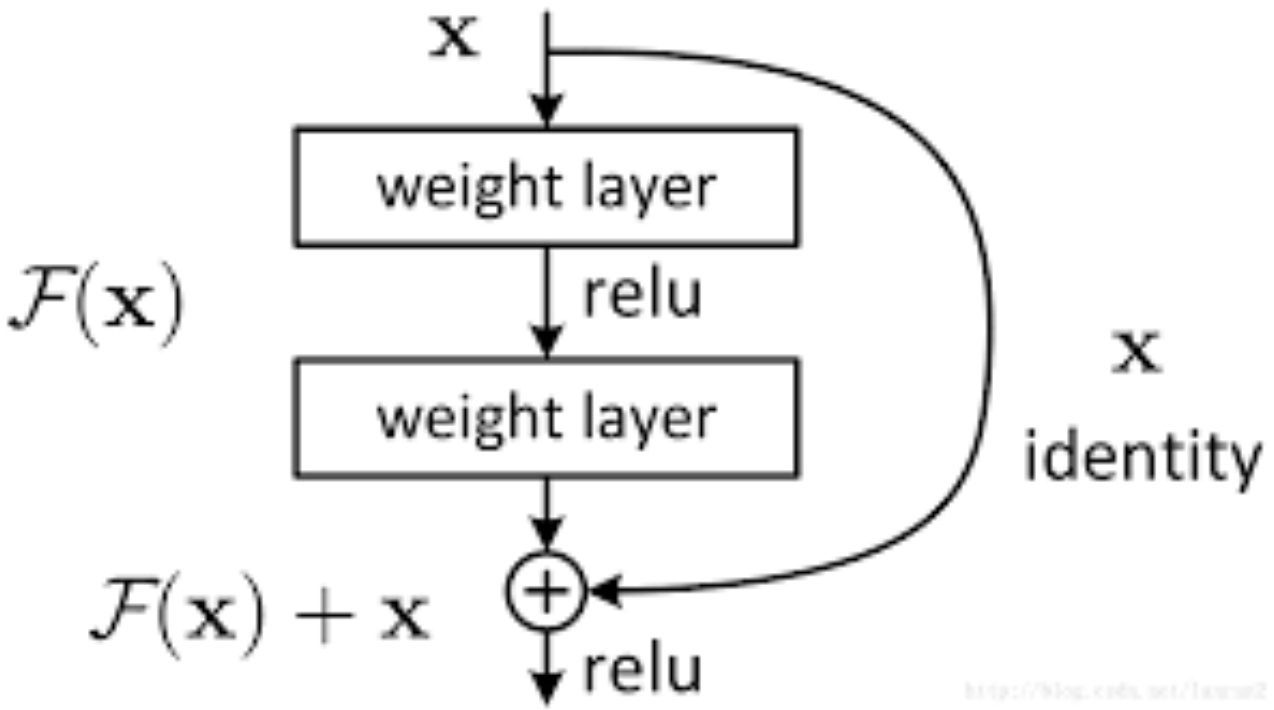
# 預測，取得features，維度為 (1,7,7,512)
features = model.predict(x)
# 取得前三個最可能的類別及機率
print('Predicted:', decode_predictions(features, top=3)[0])
```

Modification

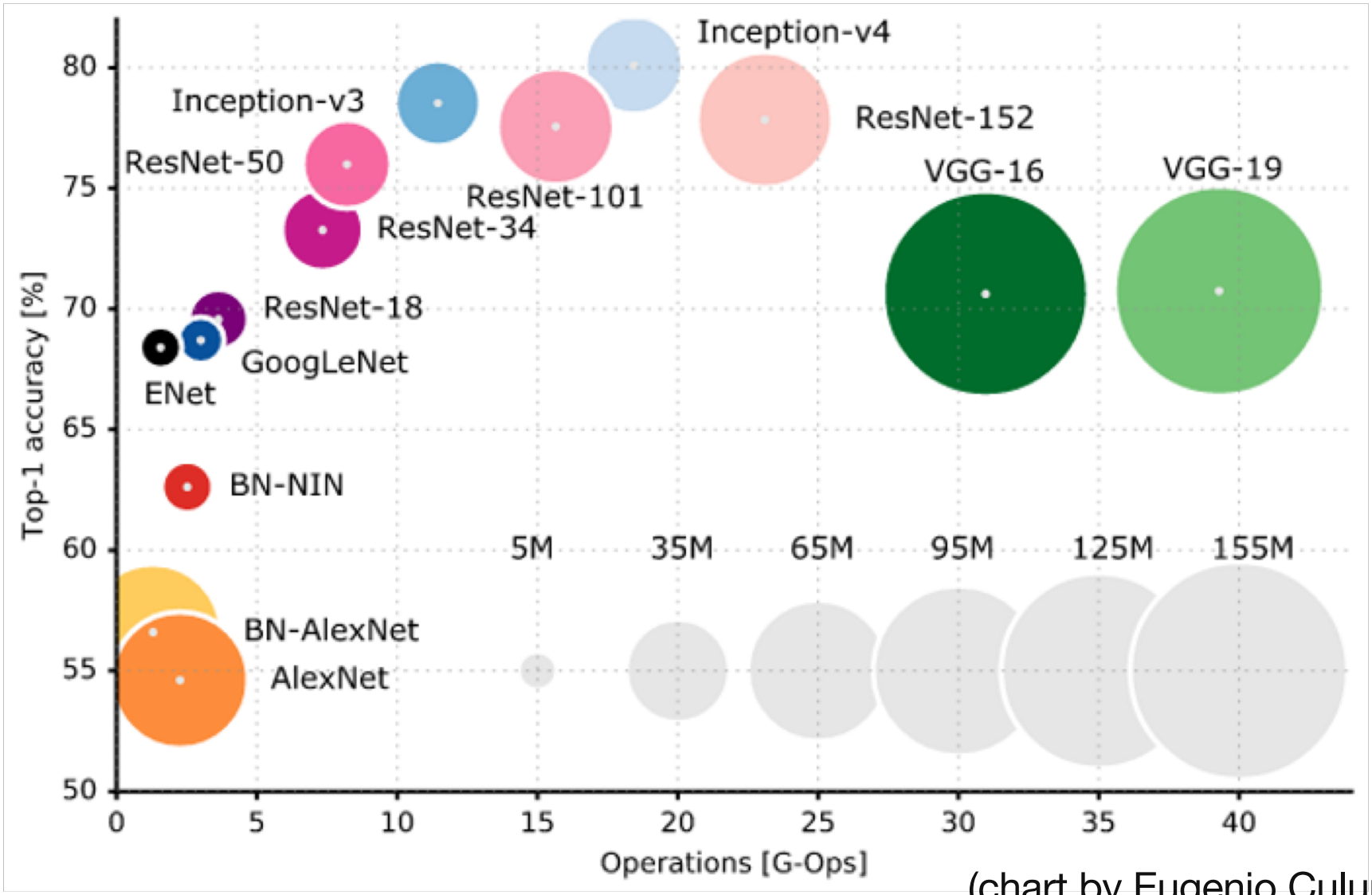
```
# 從頂部移出一層
model.layers.pop()
model.outputs = [model.layers[-1].output]
model.layers[-1].outbound_nodes = []
# 加一層，只辨識10類
from keras.layers import Dense
num_classes=10
x=Dense(num_classes, activation='softmax')(model.output)
# 重新建立模型結構
model=Model(model.input,x)
```

ResNet

- Deep Residual Learning for Image Recognition
- https://github.com/KaimingHe/deep-residual-networks



Standing on the shoulders of giants



(chart by Eugenio Culurciello

Object Recognition with Bounding Box

- Faster RCNN, YOLO (tiny, v2, v3,…), SSD