Convolutional Neural Network

主讲人：崔磊

Why CNN for Image

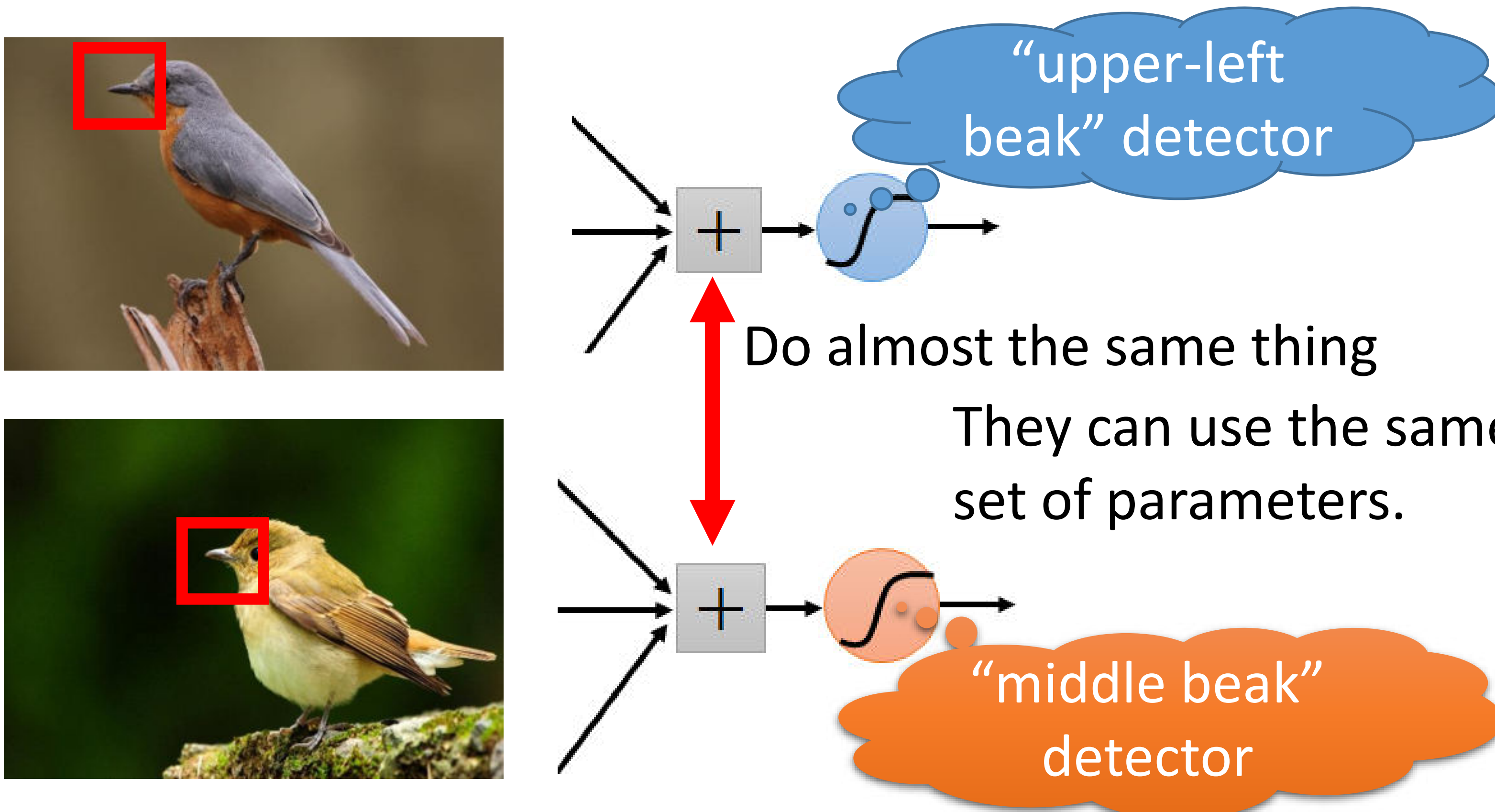
- Some patterns are much smaller than the whole image

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

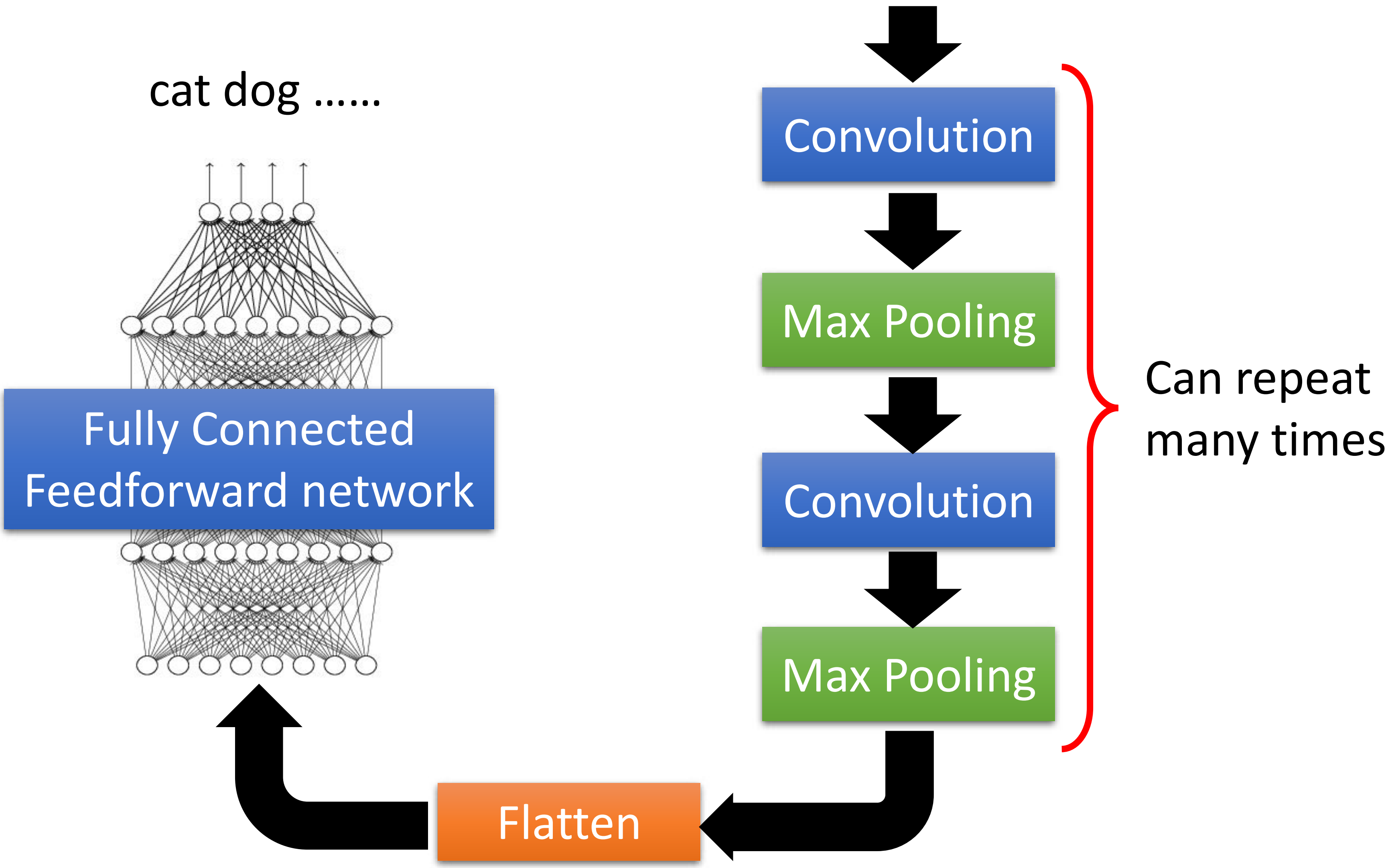
subsampling

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

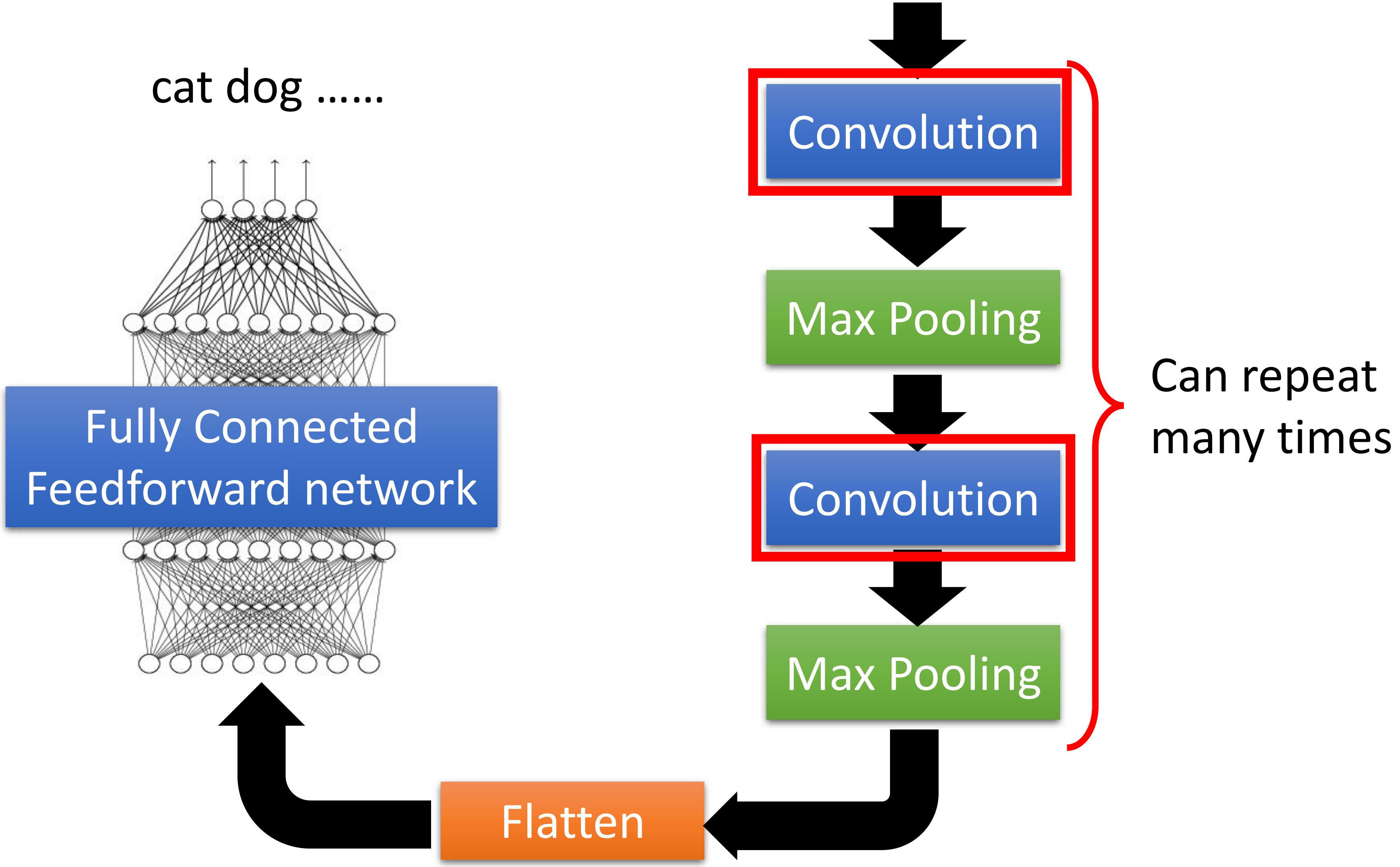
Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

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

3 -3

We set stride=1 below

CNN – Convolution

Filter 1

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

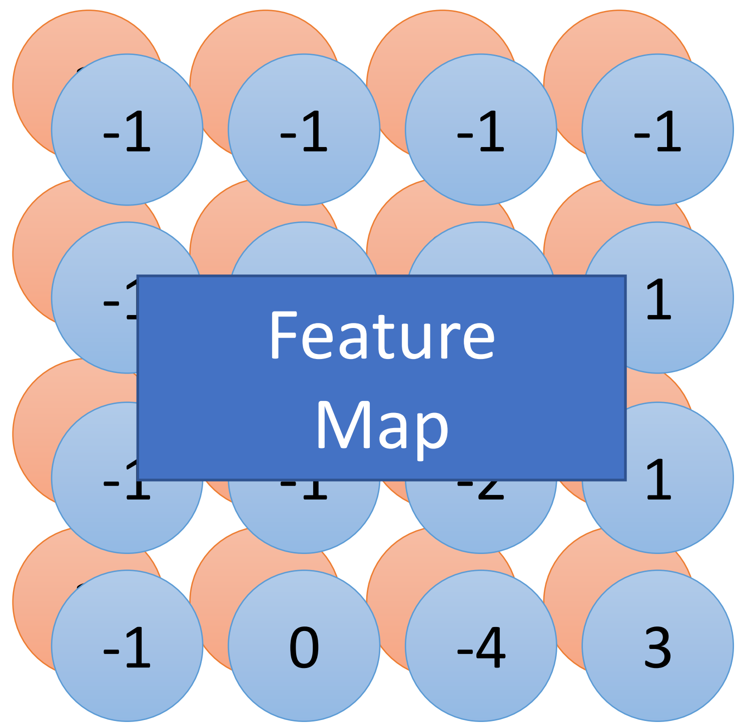
Filter 2

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

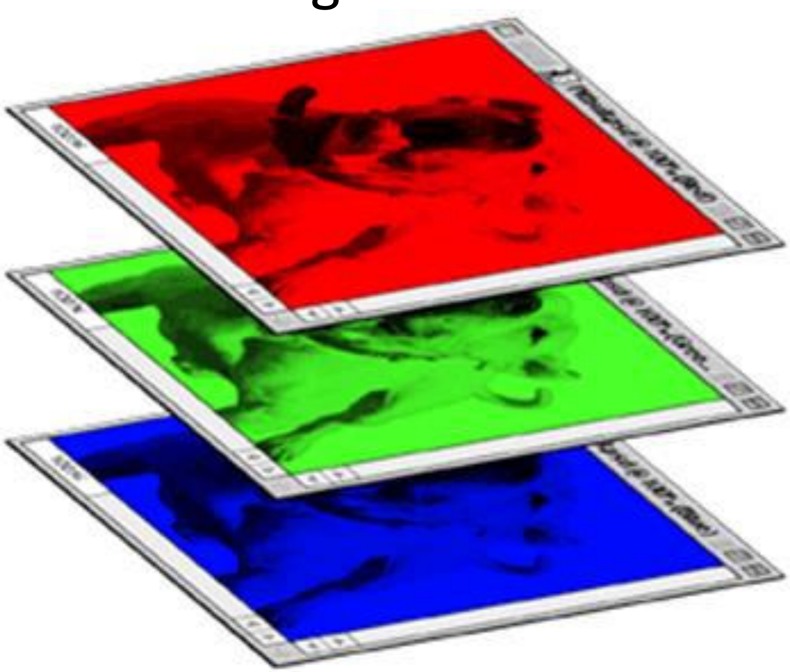
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

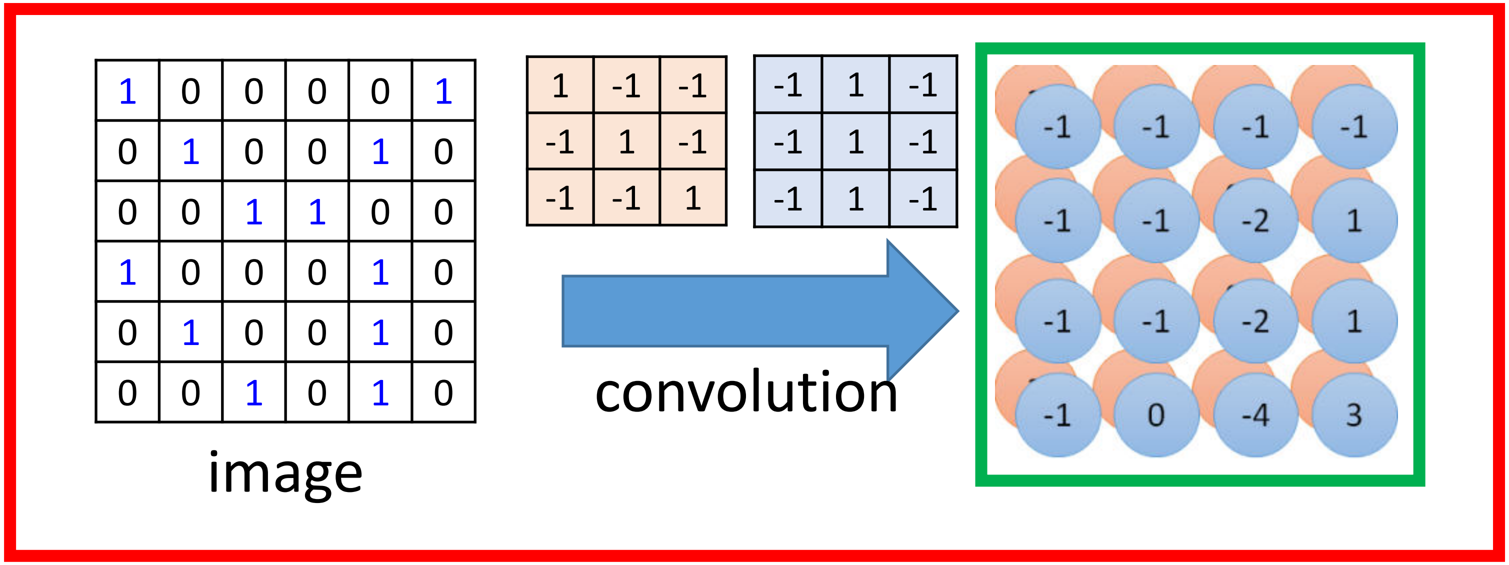
Filter 2

Colorful image

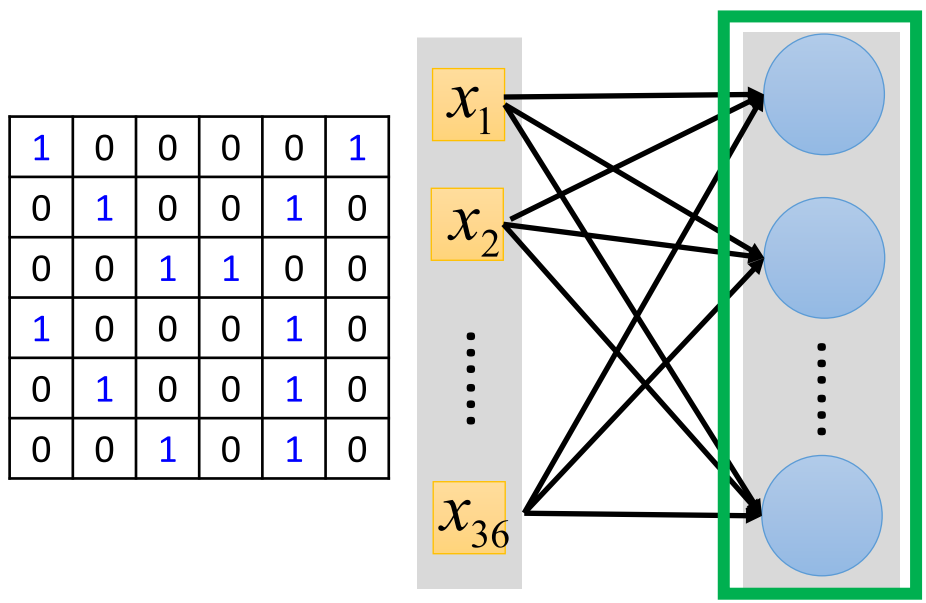


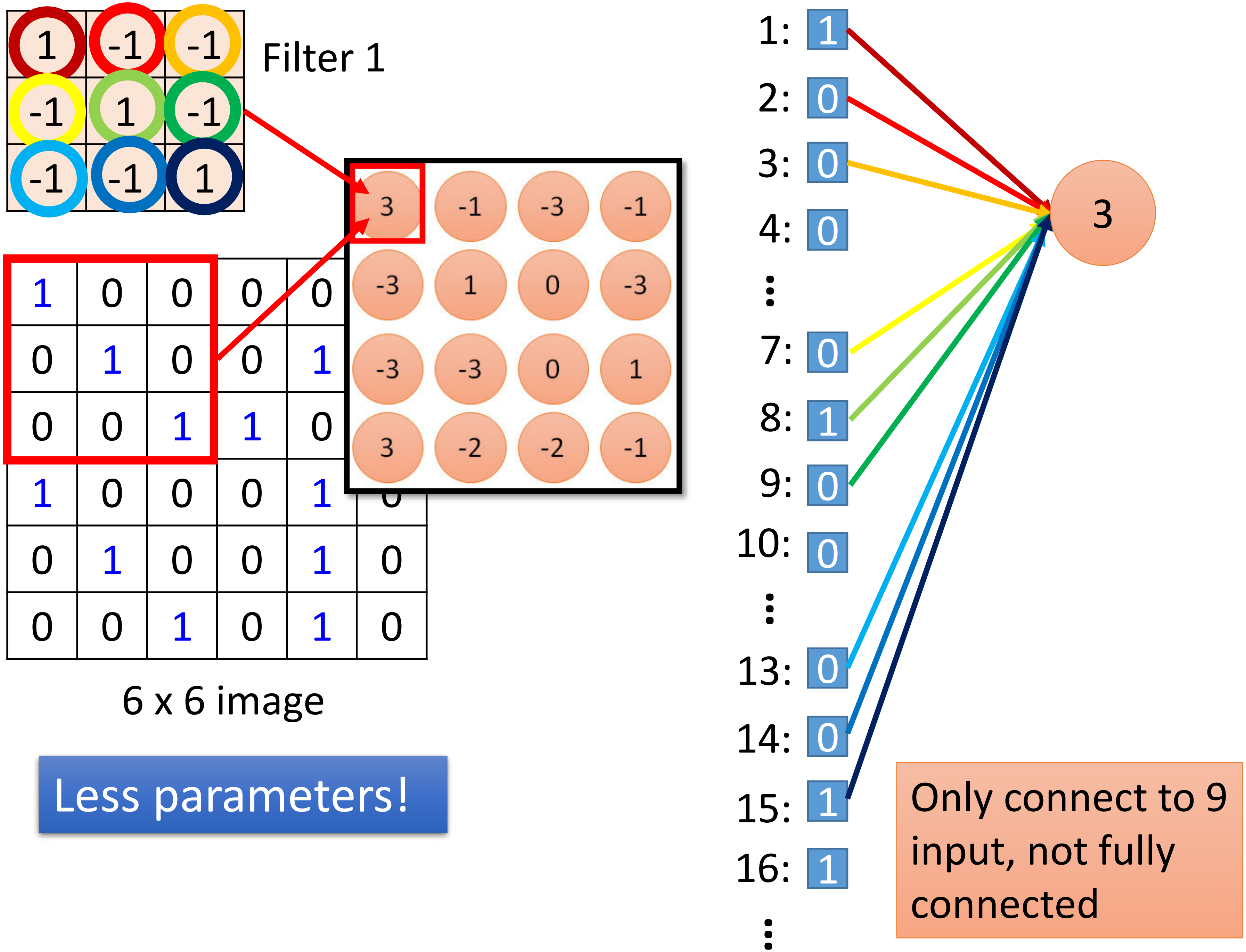
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

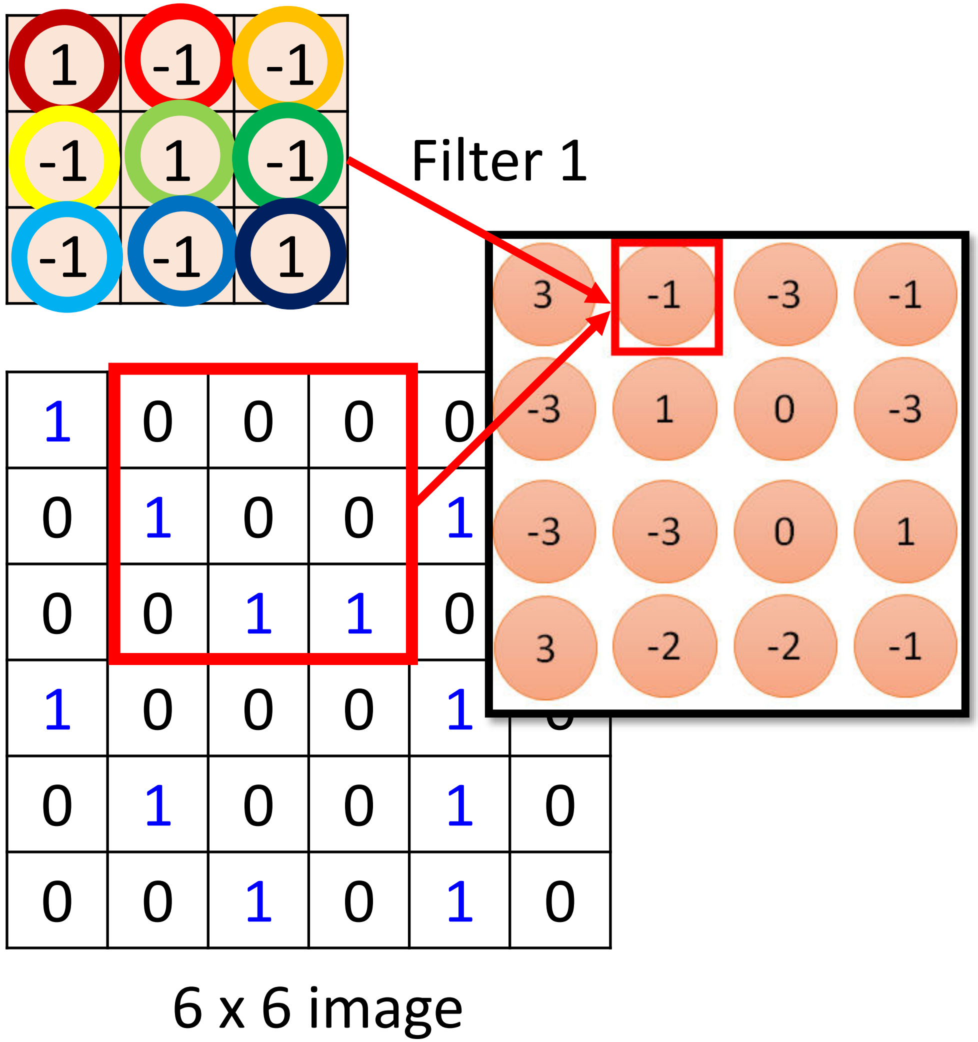
Convolution v.s. Fully Connected

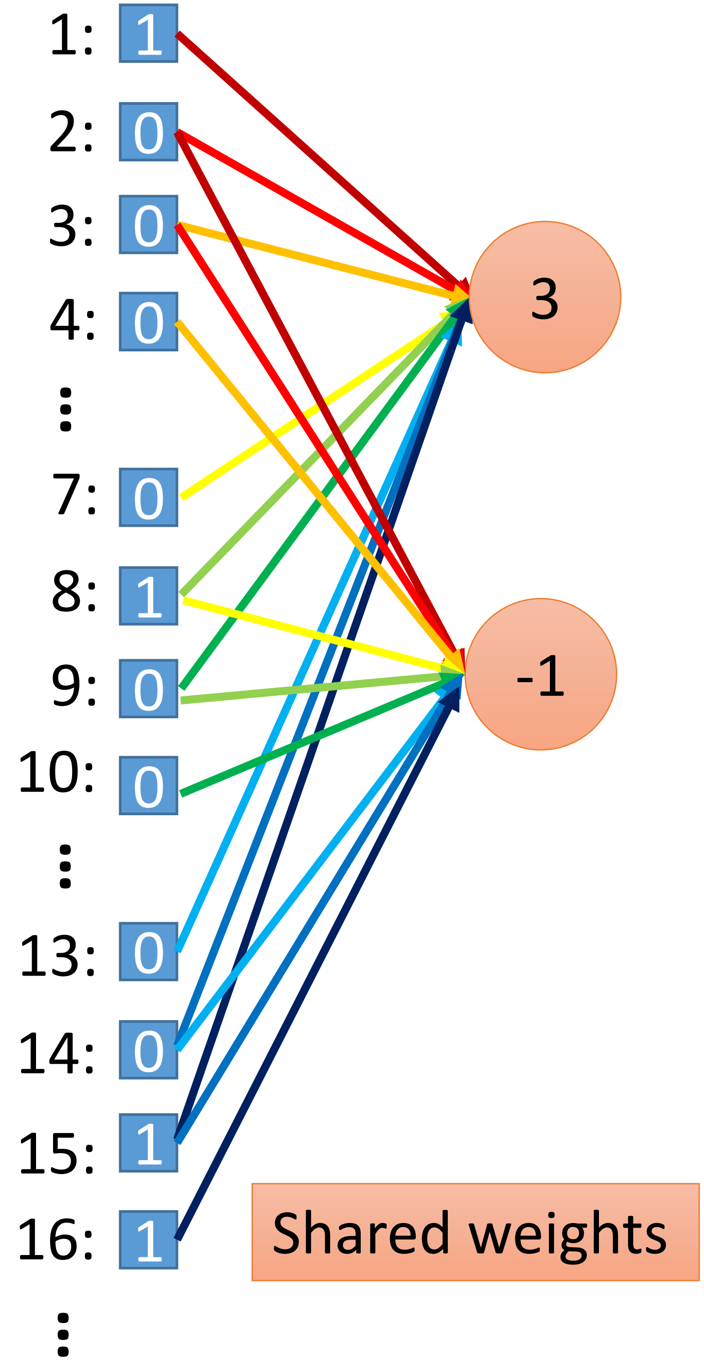


Fully-connected




Filter 1
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

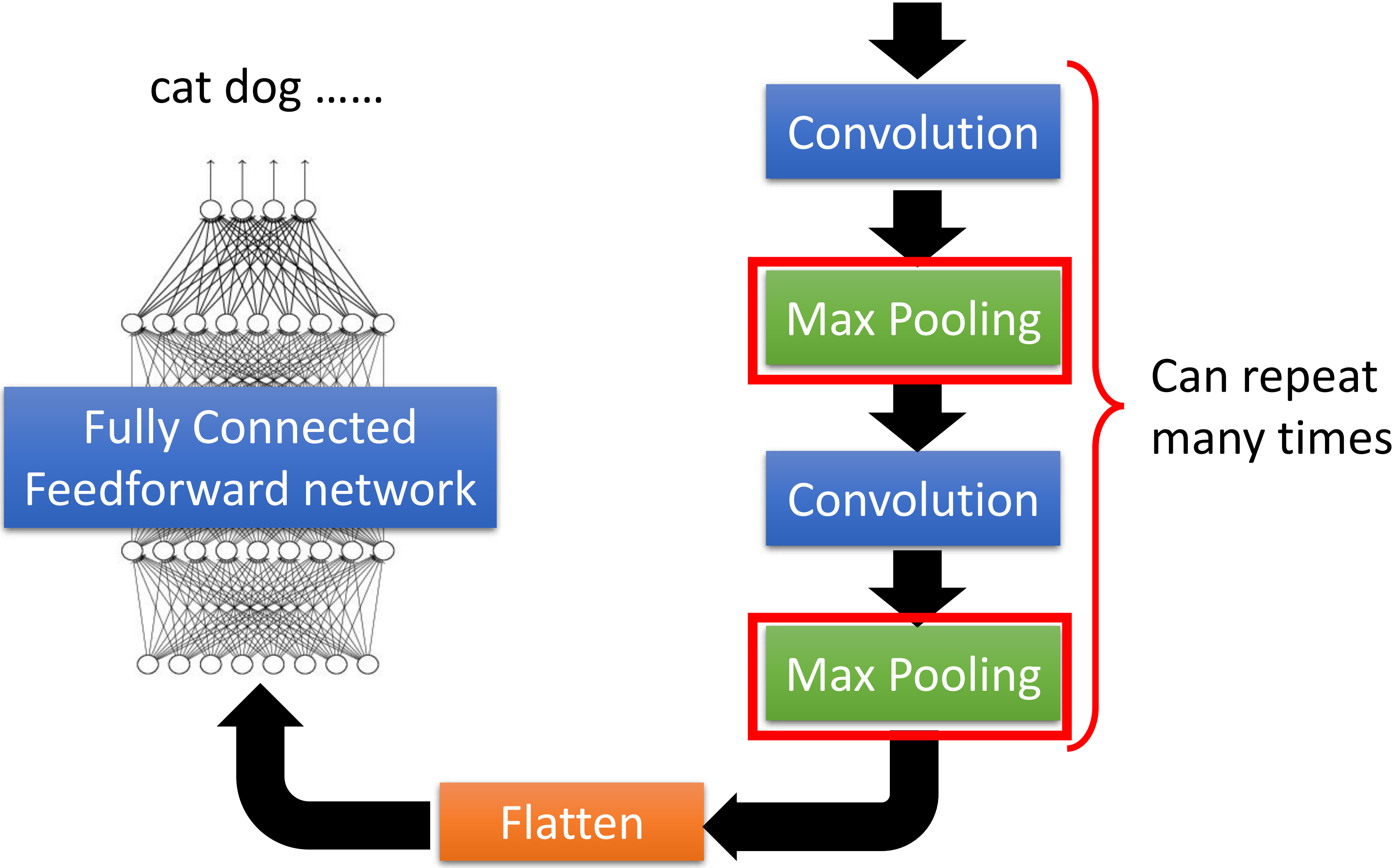




Less parameters!

Even less parameters!

The whole CNN



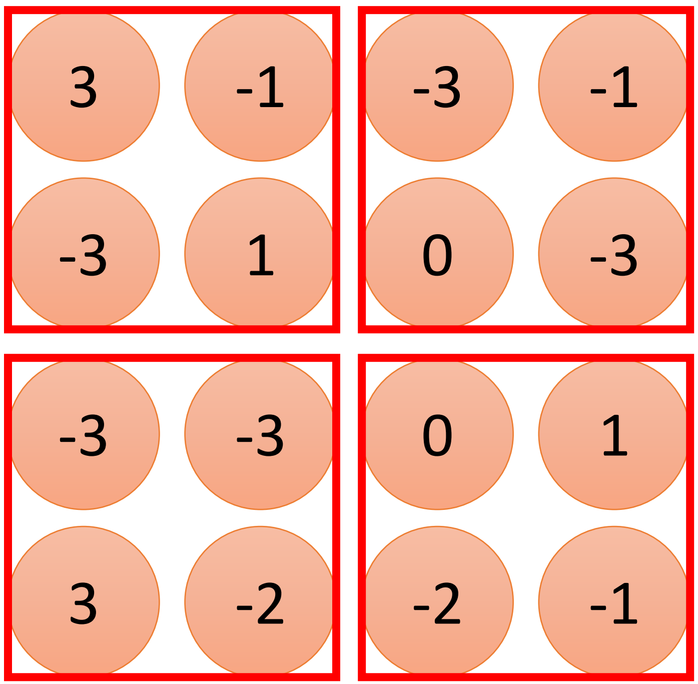
CNN – Max Pooling

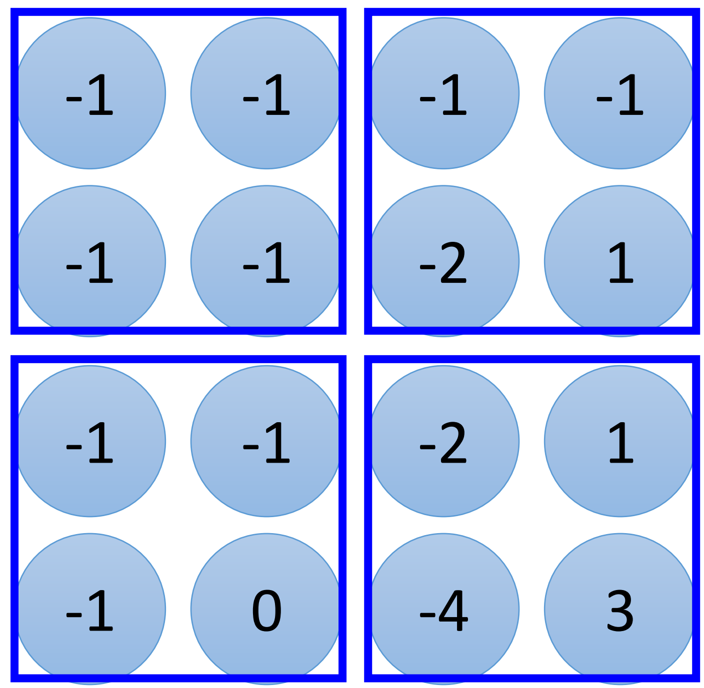
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

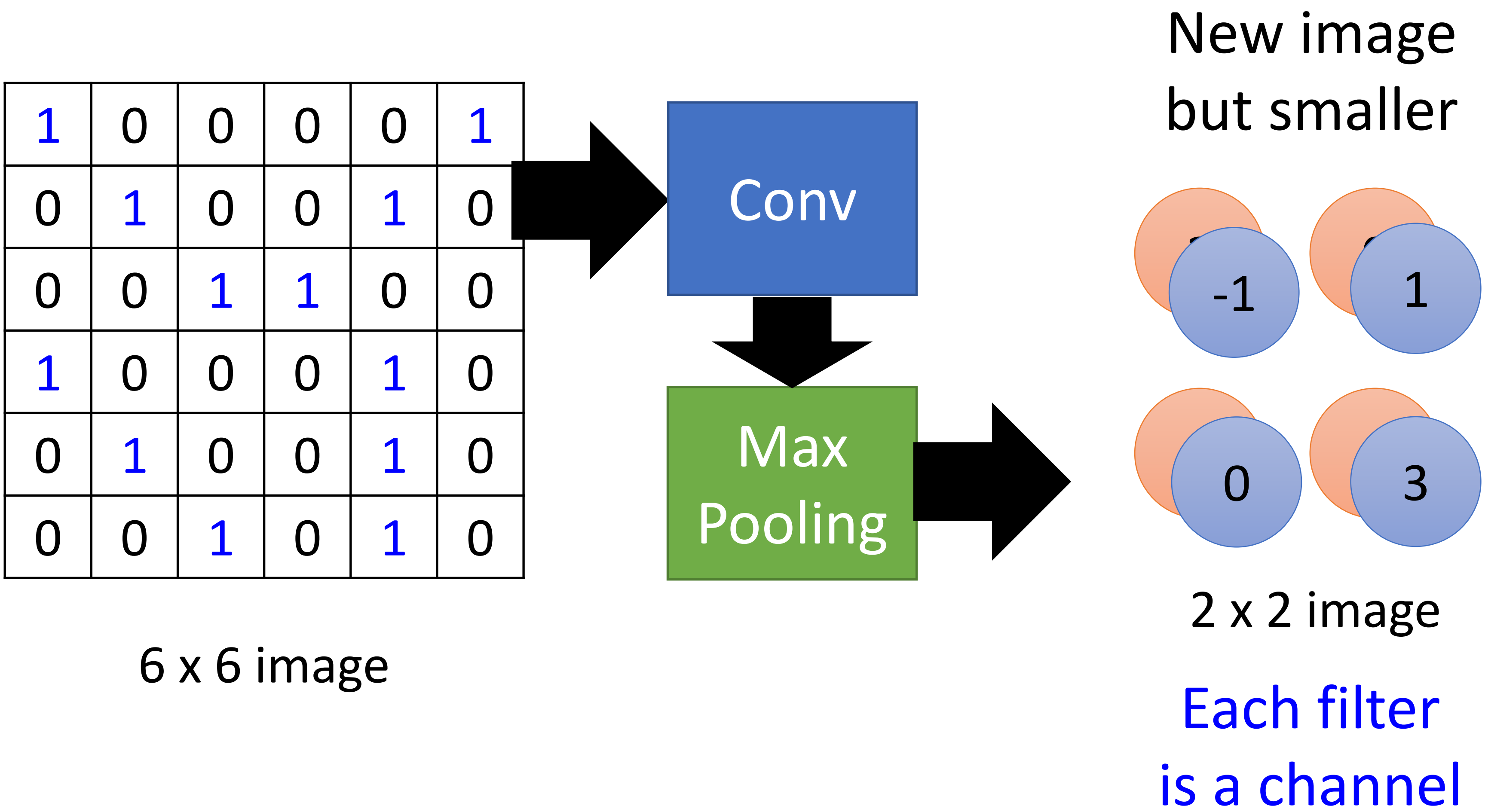
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

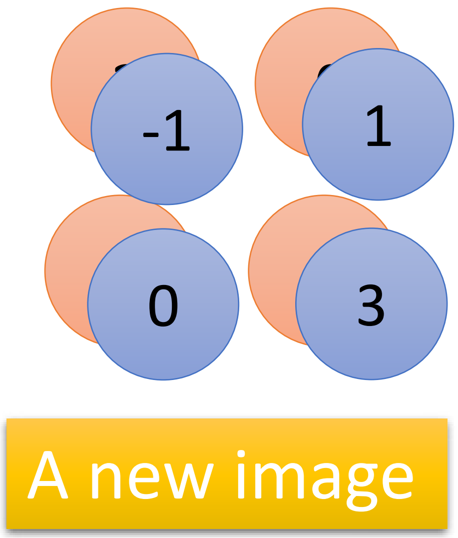


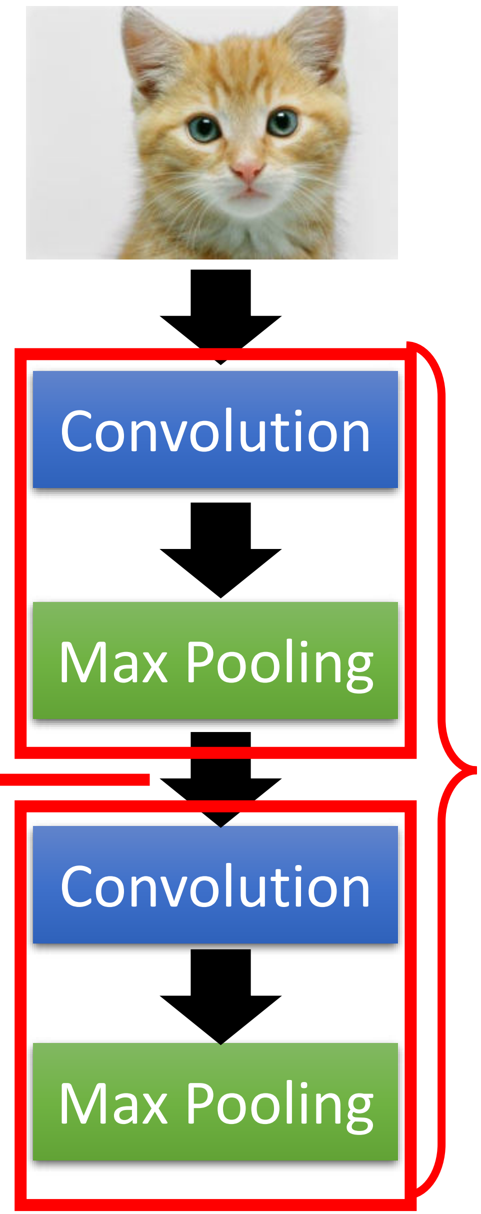


CNN – Max Pooling



The whole CNN

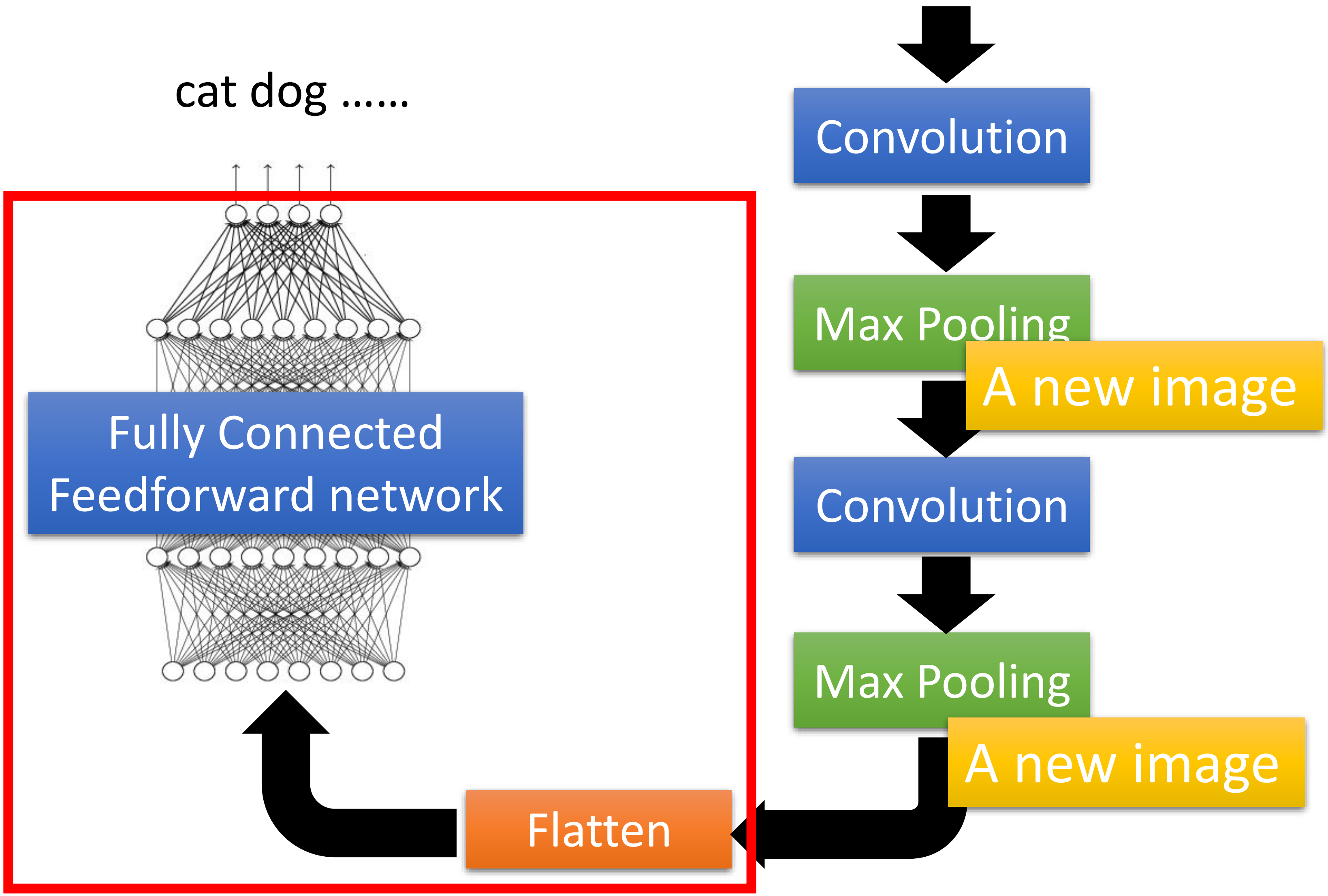




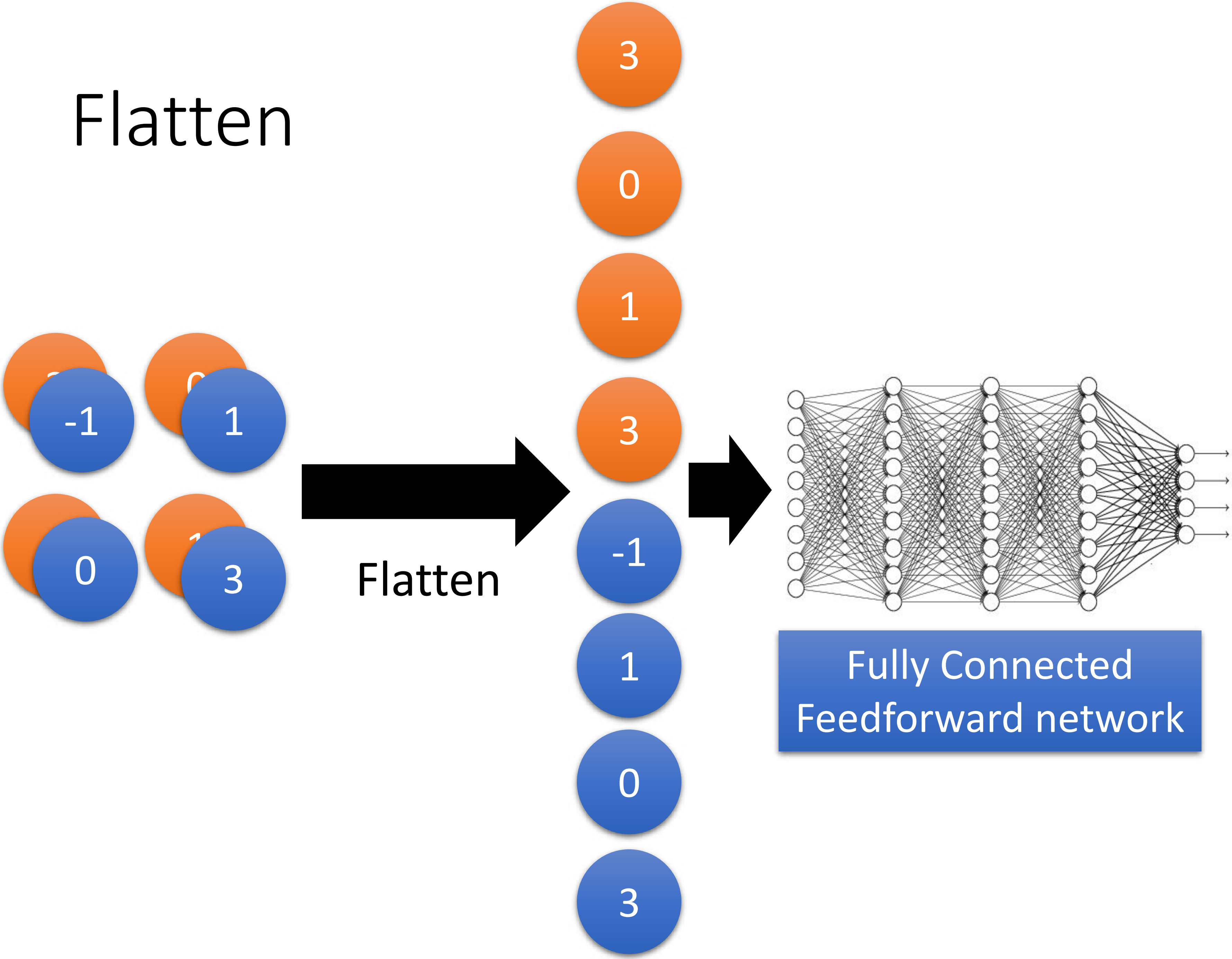
Smaller than the original image

The number of the channel is the number of filters

The whole CNN



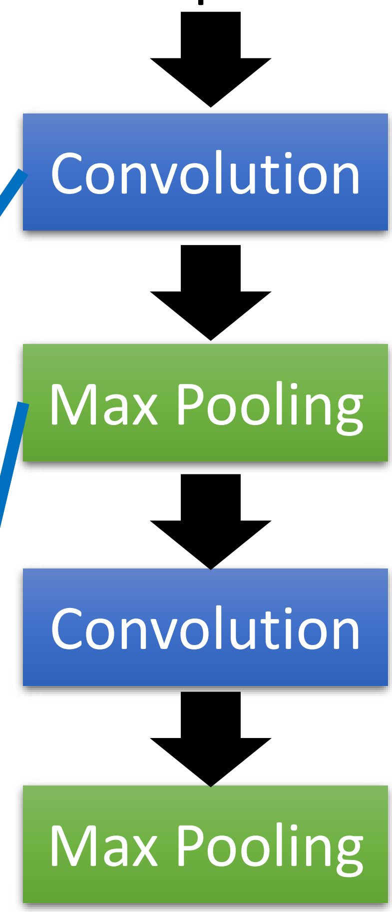
Flatten



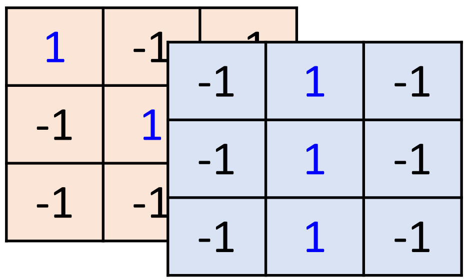
-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

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)) )
```

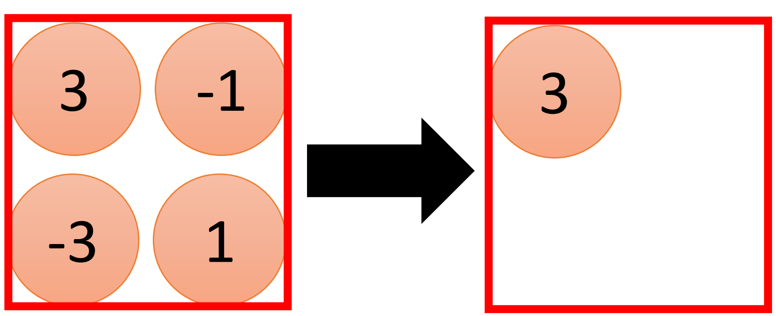


...... There are 25 3x3 filters.

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)***

1 x 28 x 28

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

How many parameters for each filter? 9

25 x 26 x 26

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

25 x 13 x 13

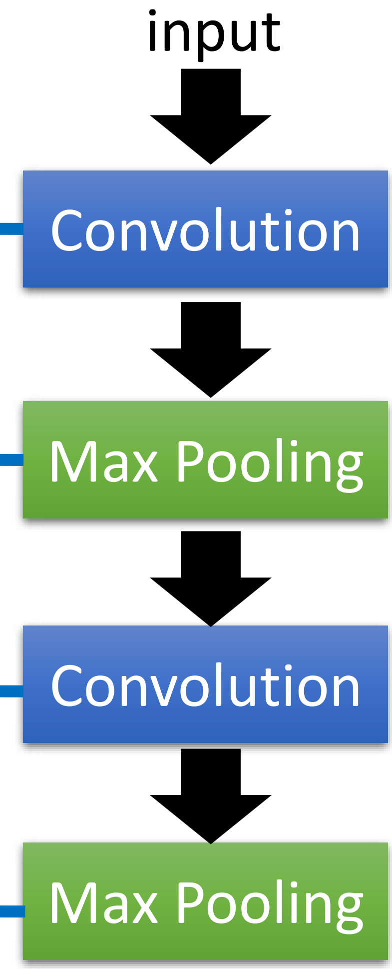
```
model2.add(Convolution2D(50,3,3))
```

How many parameters for each filter? 225

50 x 11 x 11

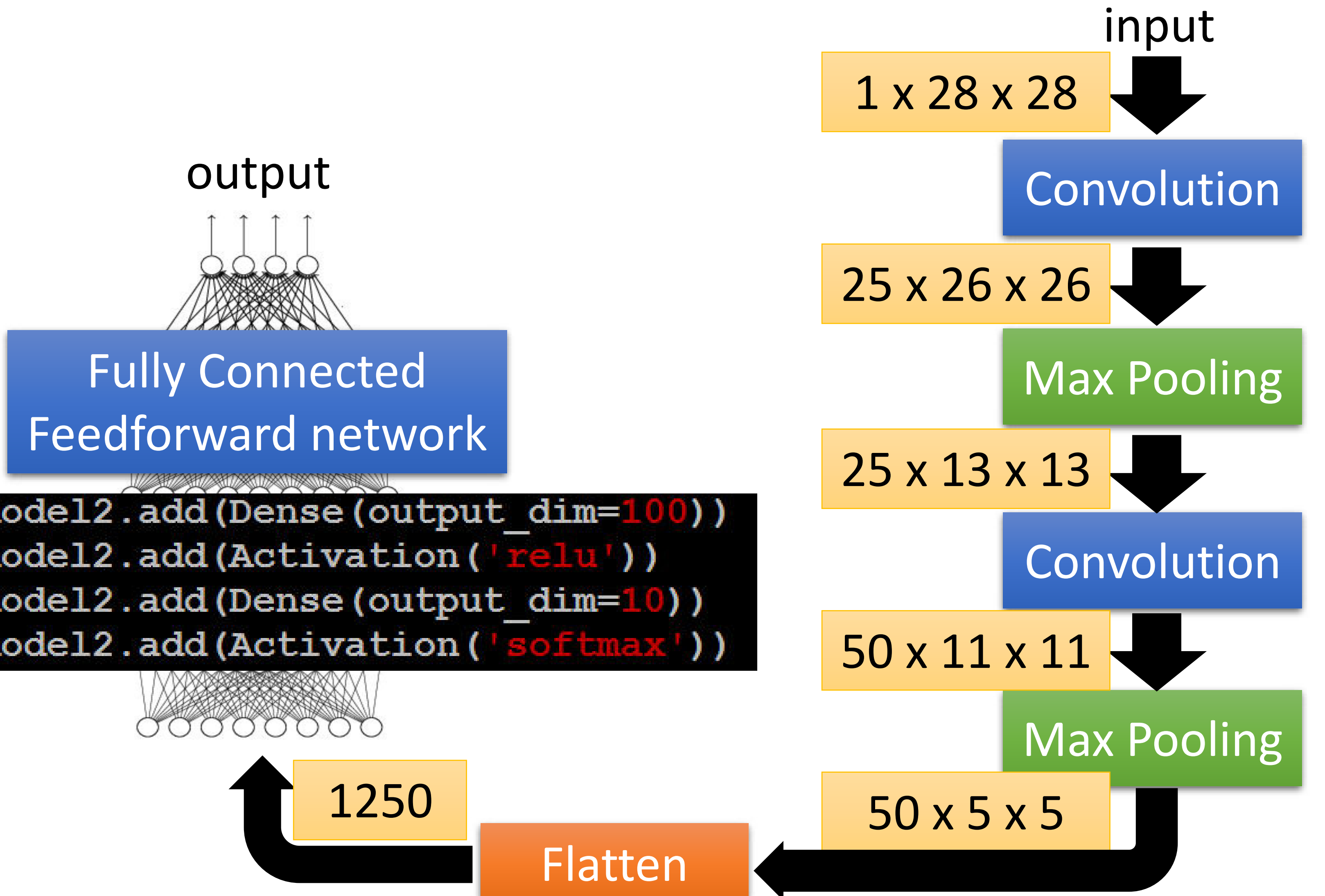
```
model2.add(MaxPooling2D((2,2)))
```

50 x 5 x 5



CNN 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'))
```

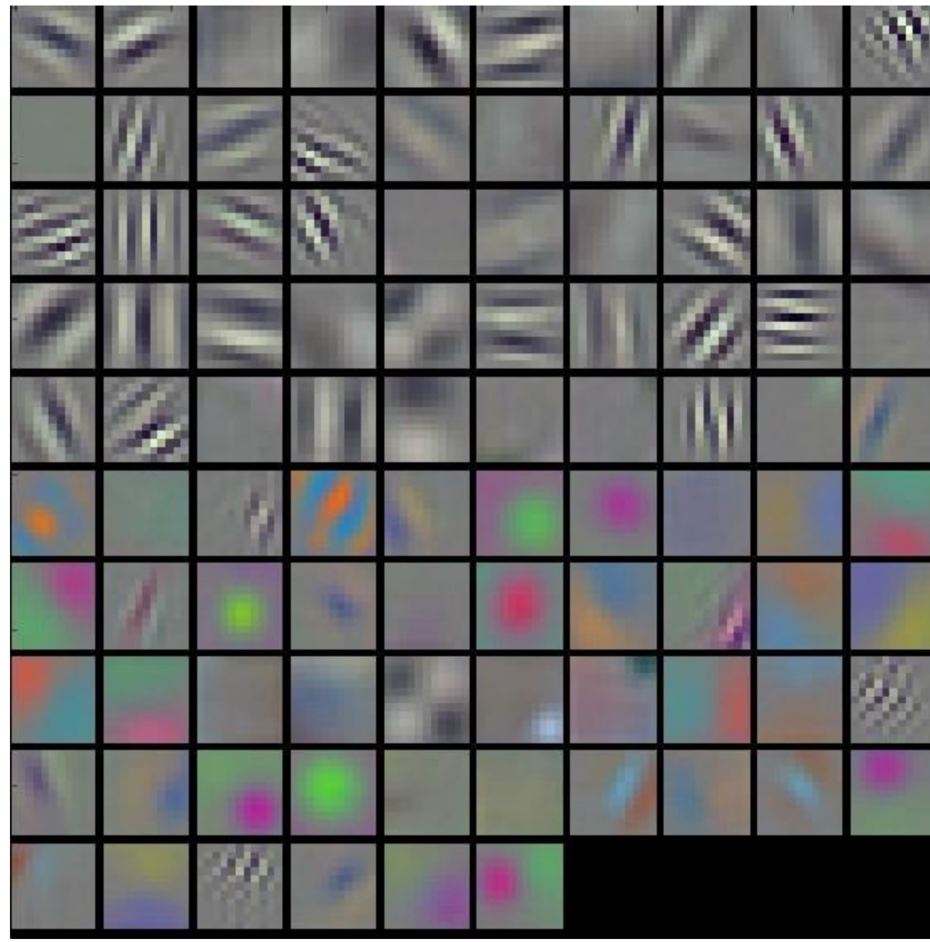
```
model2.add(Flatten())
```

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 filters on the trained first layer

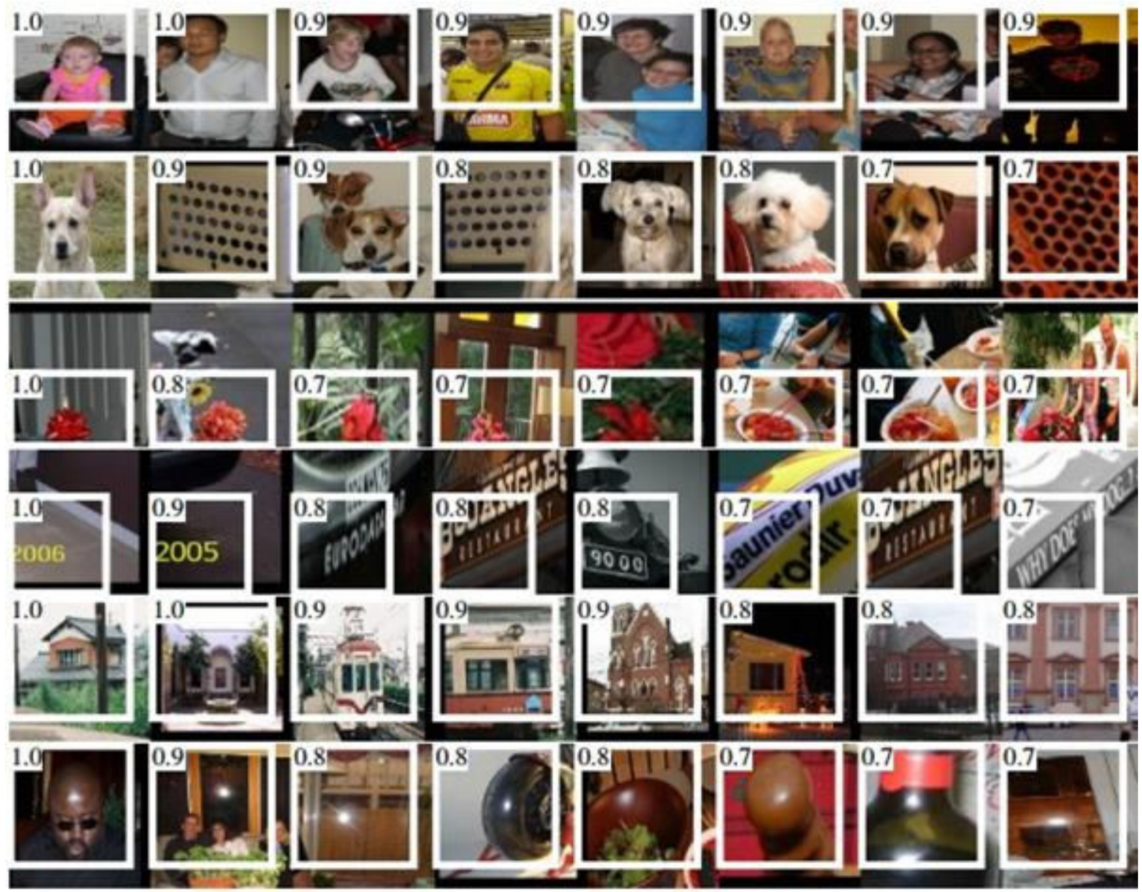


11 x 11
(AlexNet)

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

How about higher layers?

- Which images make a specific neuron activate



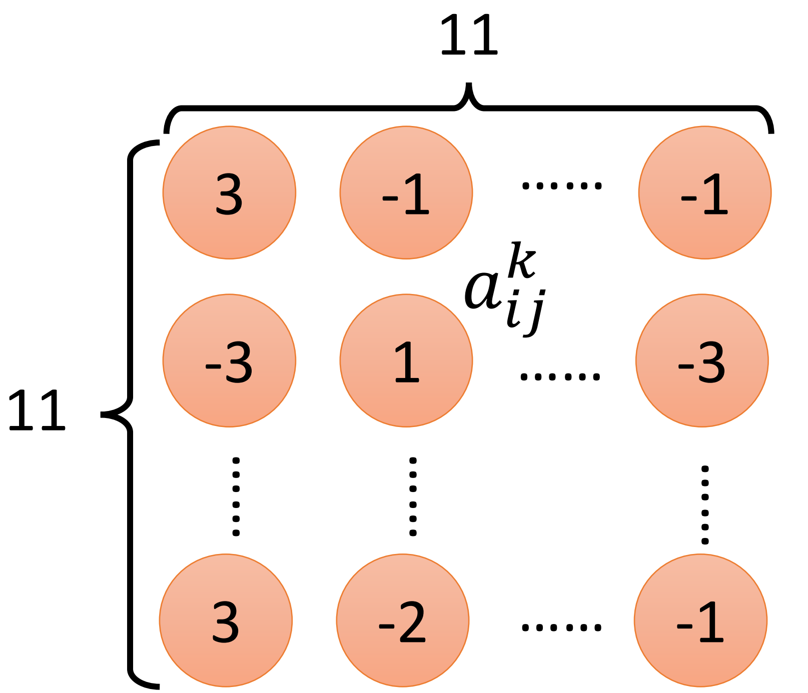
Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR, 2014

What does CNN learn?

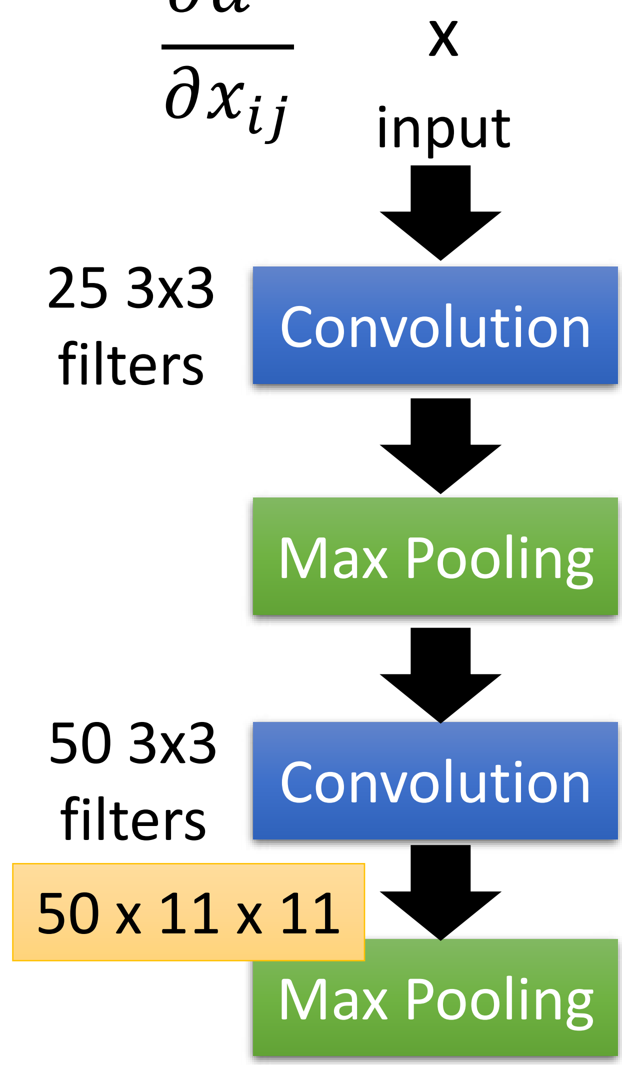
The output of the k-th filter is a 11 x 11 matrix.

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)



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



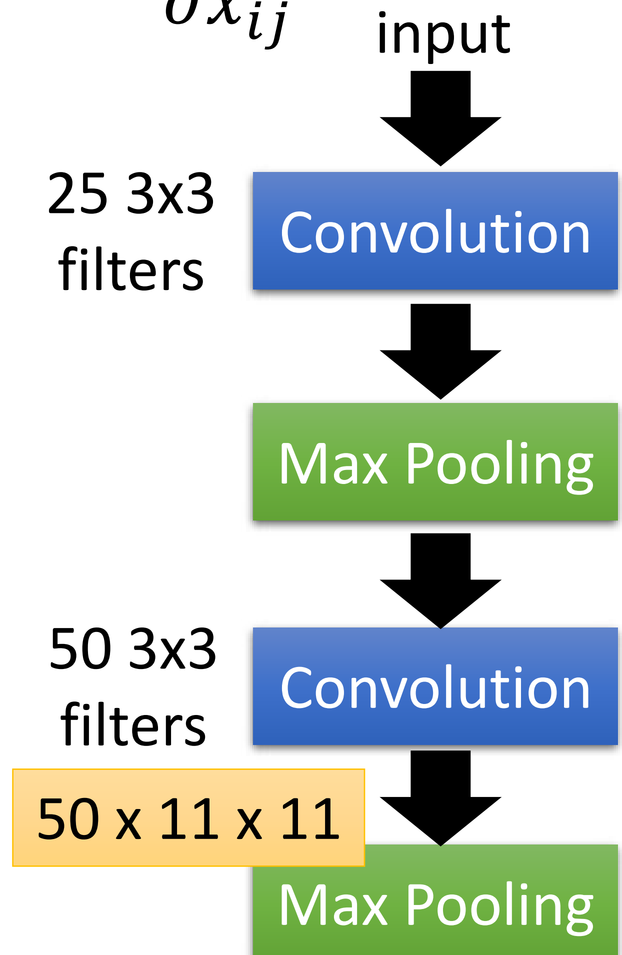
What does CNN learn?

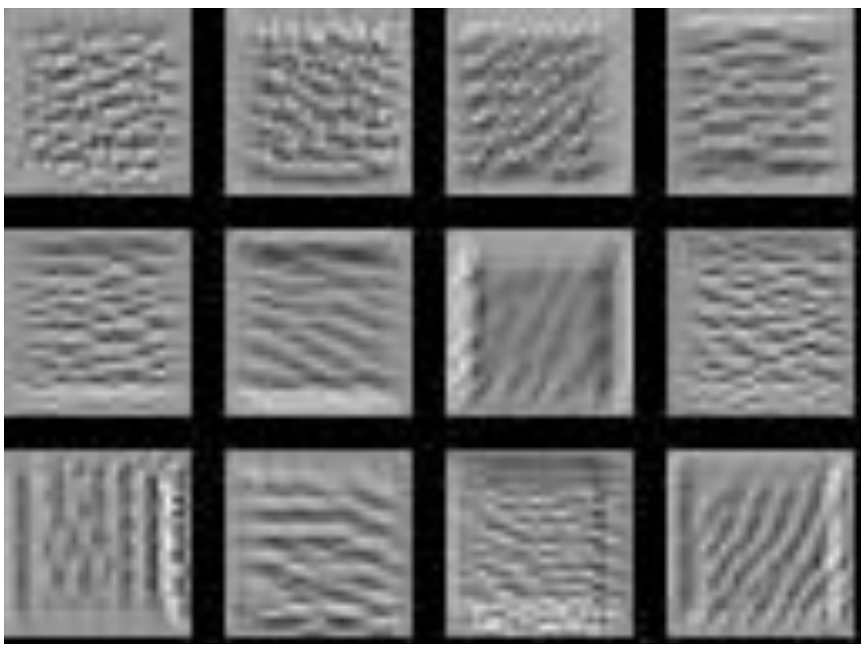
The output of the k-th filter is a 11 x 11 matrix.

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)

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



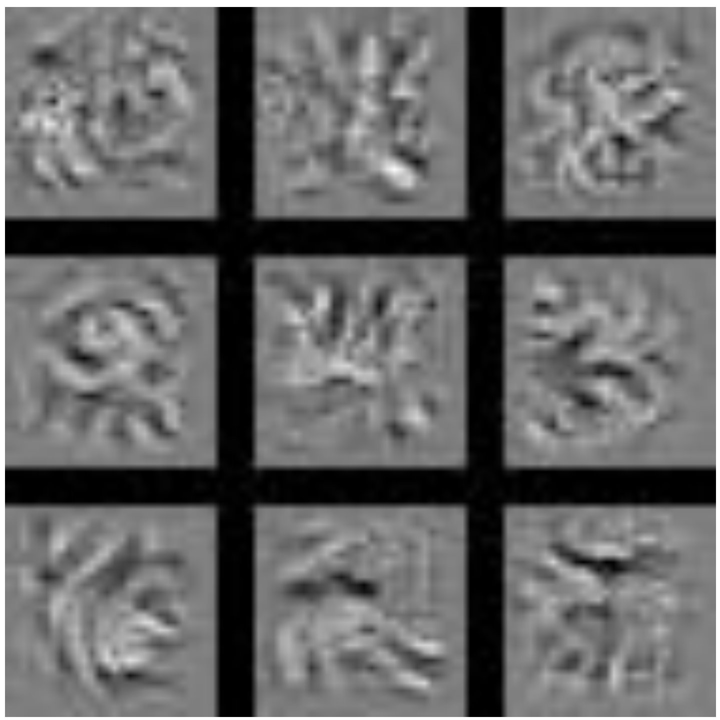


For each filter

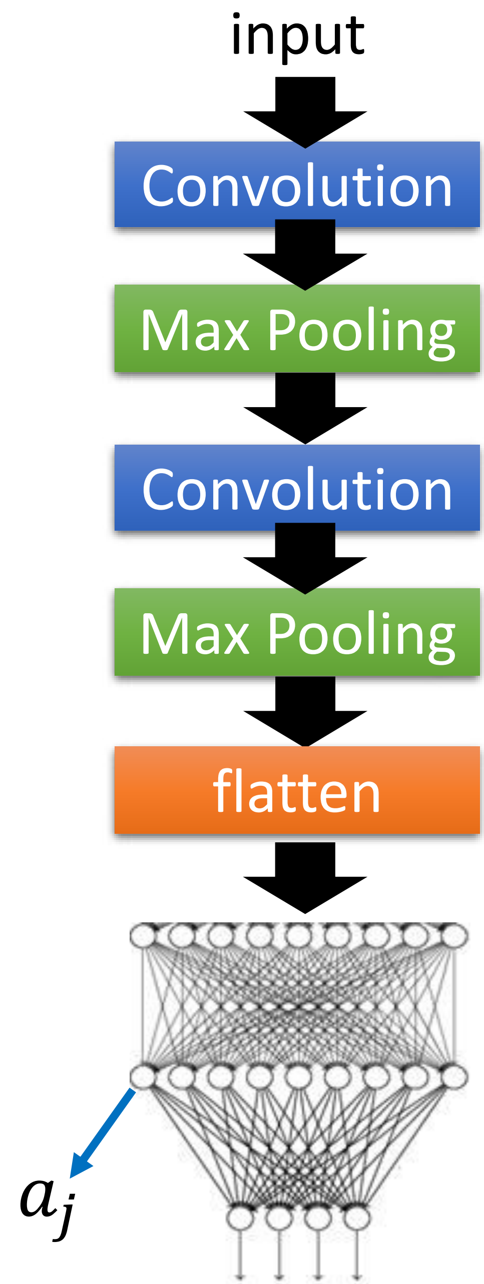
What does CNN learn?

Find an image maximizing the output of neuron:

$$x^* = arg \max_{x} a^j$$



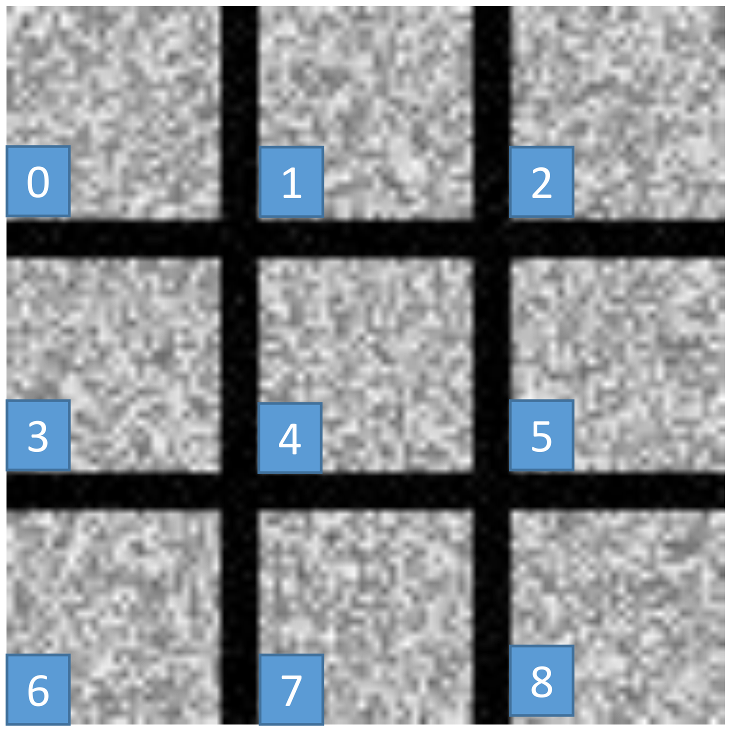
Each figure corresponds to a neuron

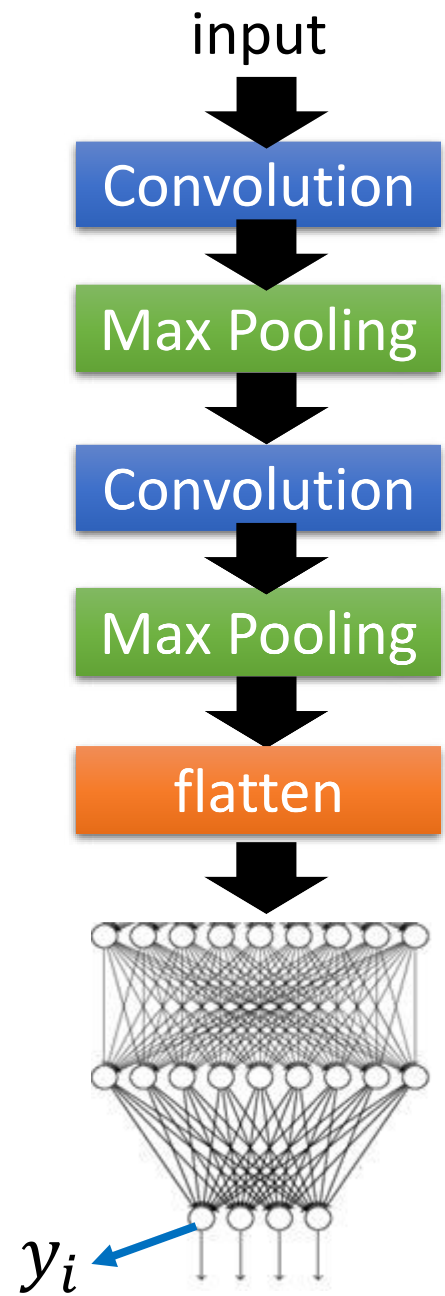


What does CNN learn?

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

Can we see digits?

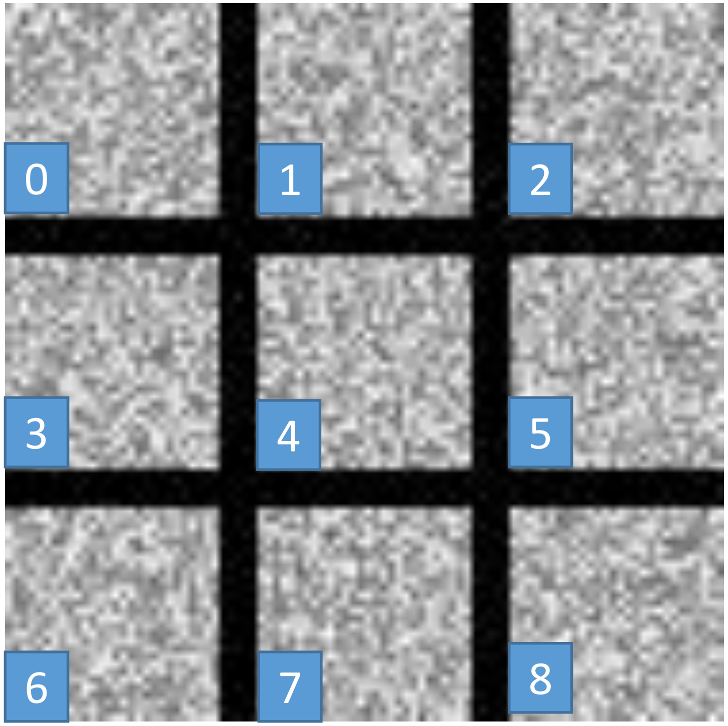




Deep Neural Networks are Easily Fooled
https://www.youtube.com/watch?v=M2IebCN9Ht4

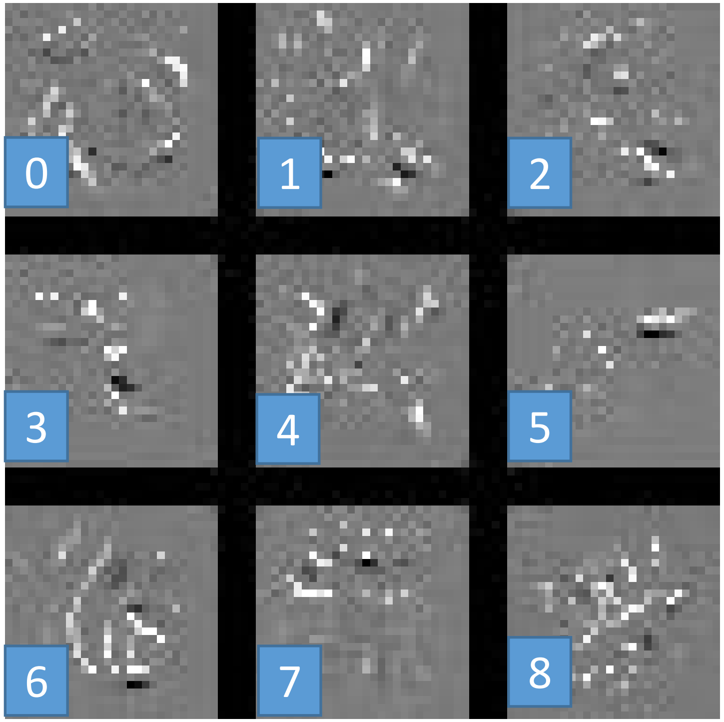
What does CNN learn?

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



Over all pixel values

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



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

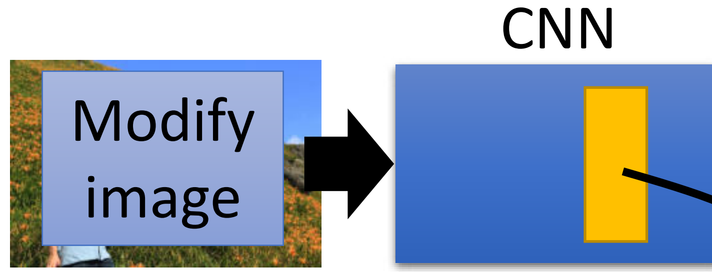
$$\left|\frac{\partial y_k}{\partial x_{ij}}\right|$$

y_k: the predicted class of the model

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

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

Deep Dream



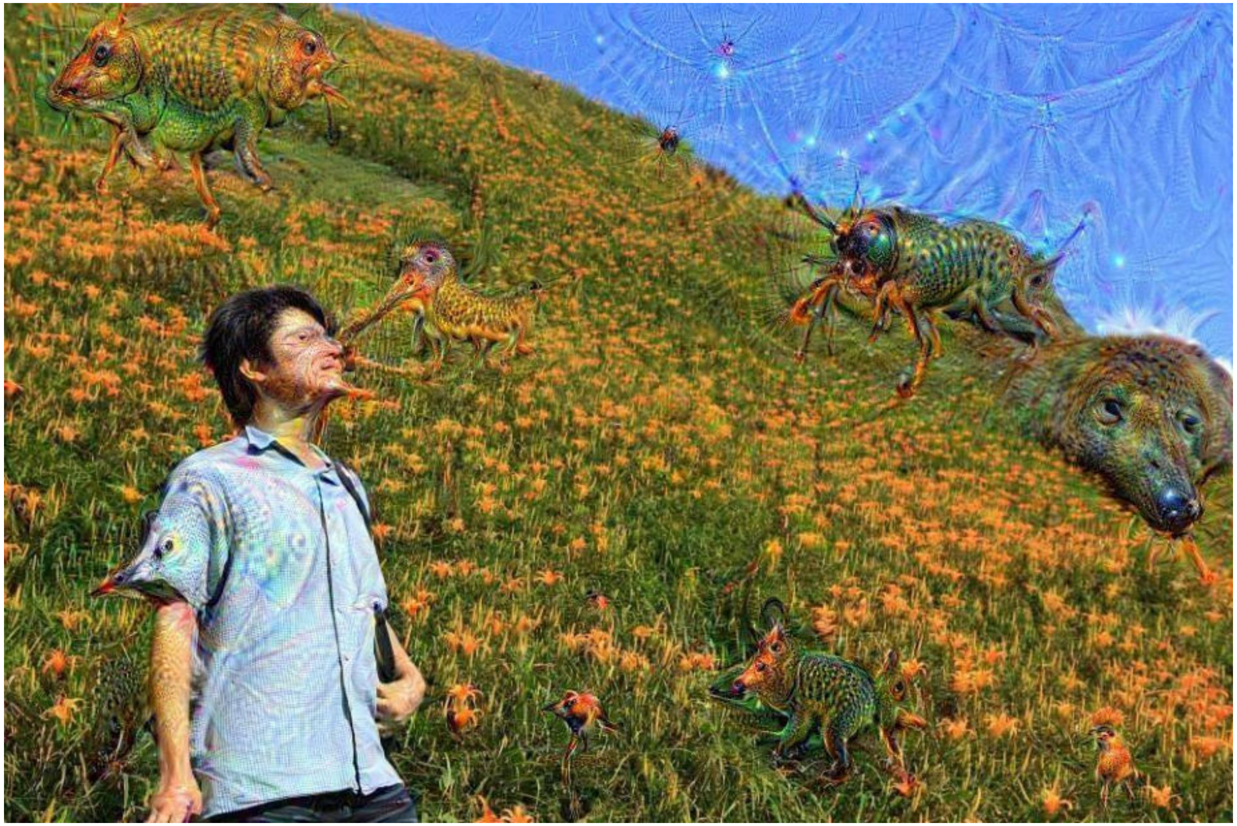
$$\begin{bmatrix} 3.9 \\ -1.5 \\ 2.3 \\ \vdots \end{bmatrix}$$

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

http://deepdreamgenerator.com/

Deep Dream

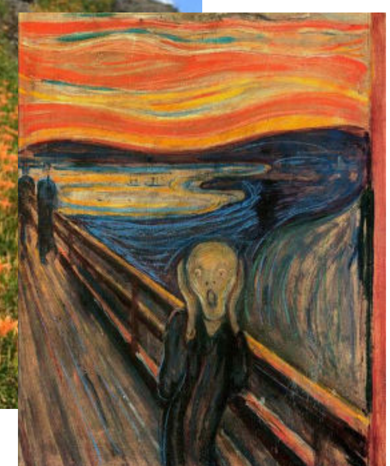
- Given a photo, machine adds what it sees ……



http://deepdreamgenerator.com/

Deep Style

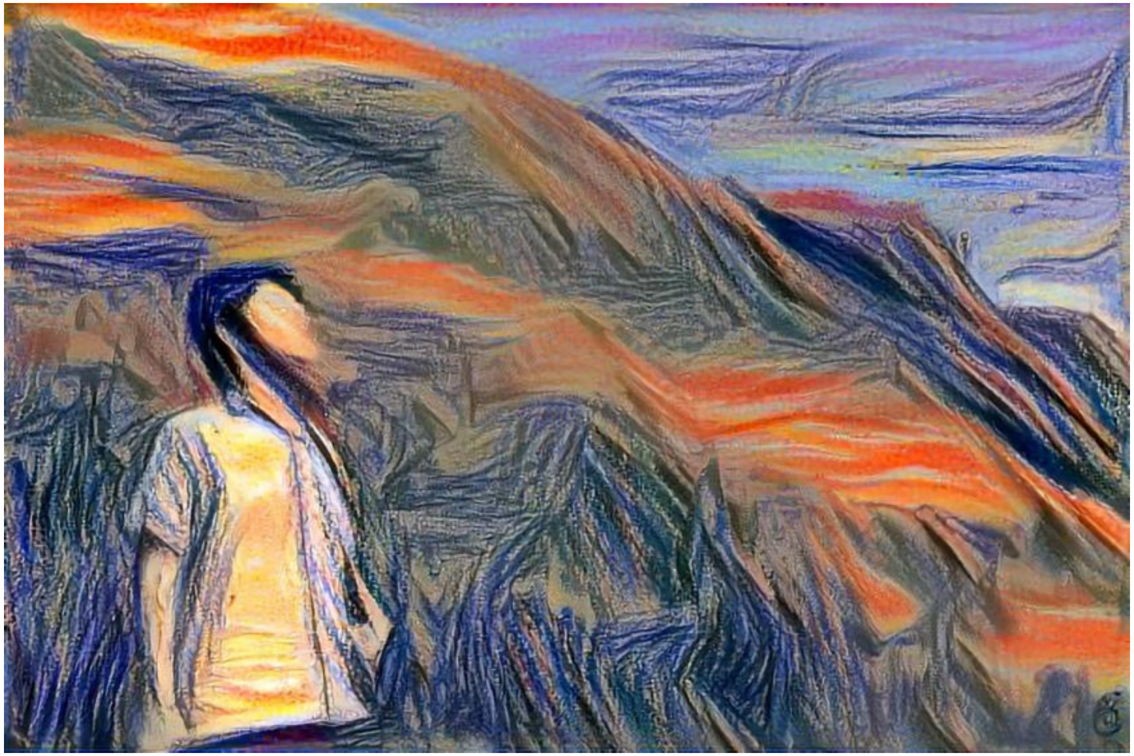
- Given a photo, make its style like famous paintings



https://dreamscopeapp.com/

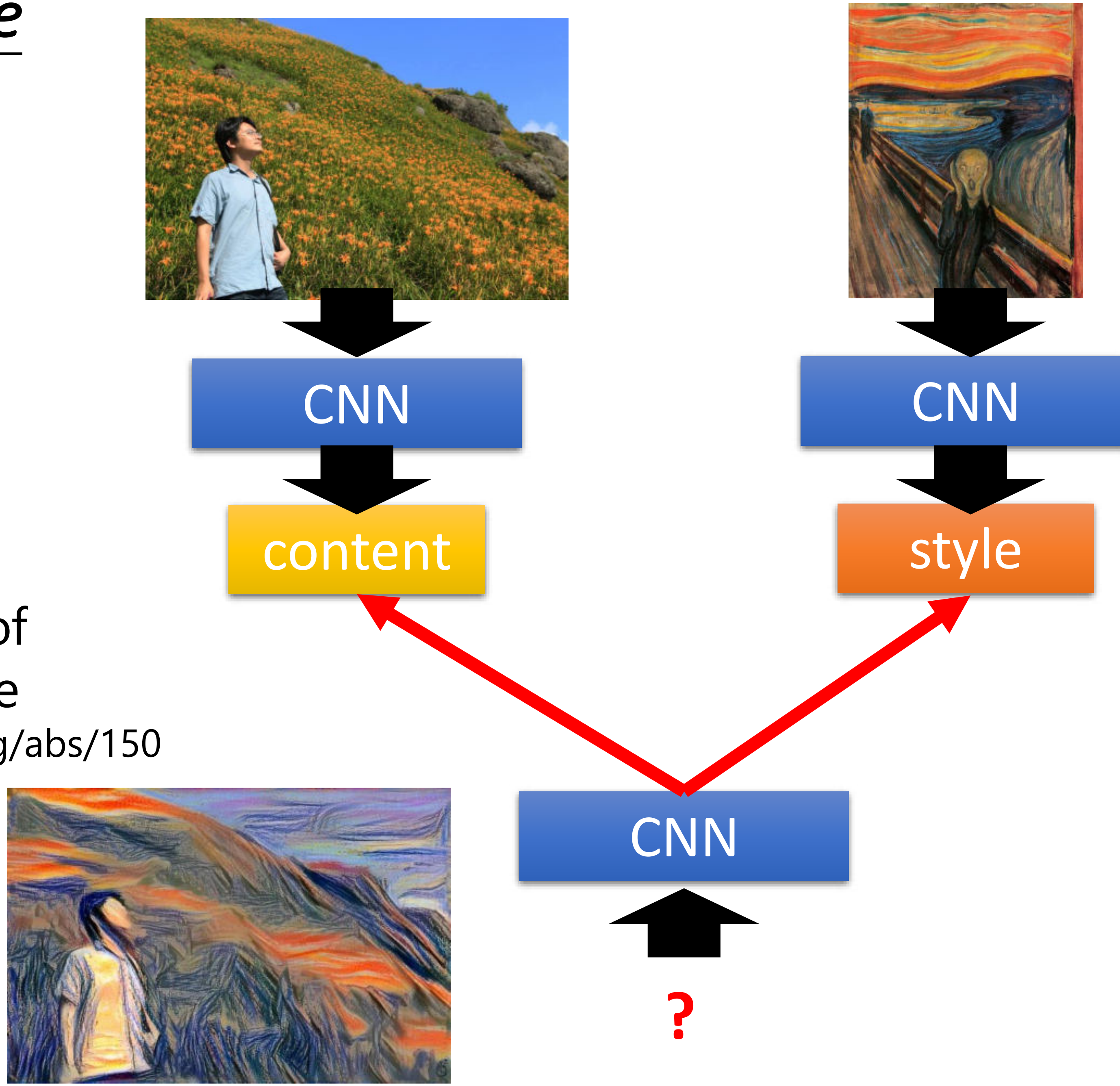
Deep Style

- Given a photo, make its style like famous paintings



https://dreamscopeapp.com/

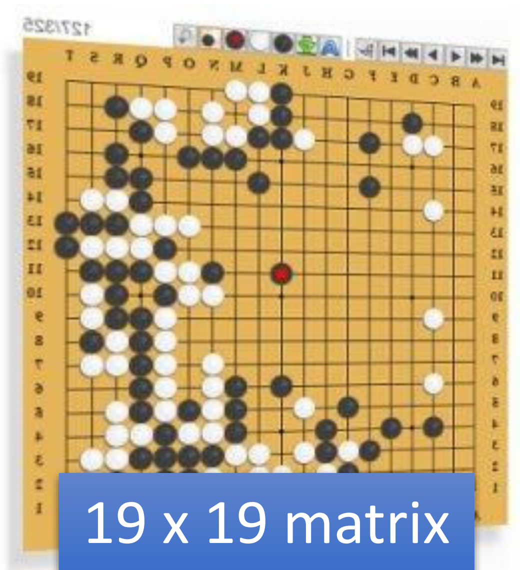
Deep Style



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)

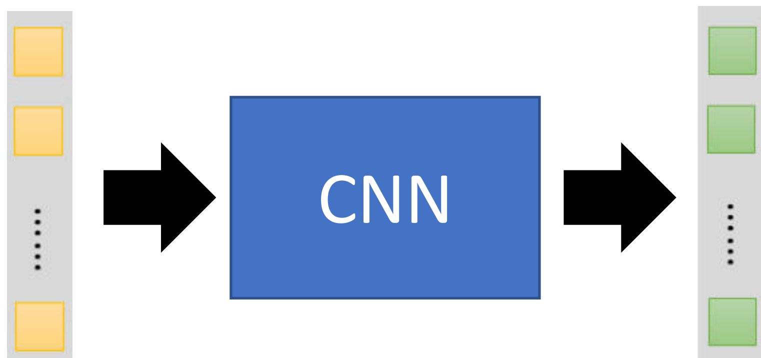
19 x 19 vector

Fully-connected feedforward network can be used

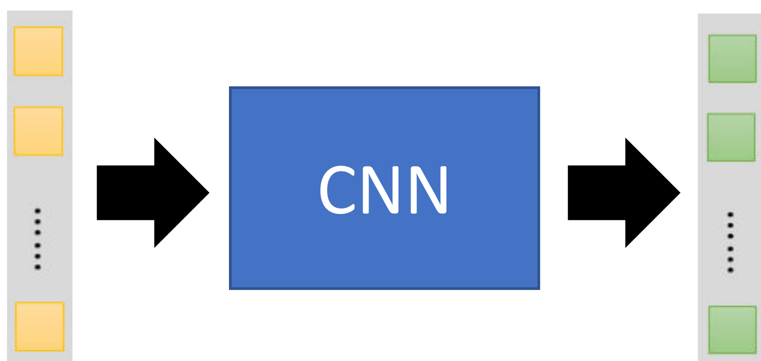
But CNN performs much better.

More Application: Playing Go

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



Target:
"天元" = 1
else = 0

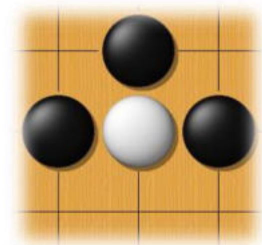


Target:
"五之 5" = 1
else = 0

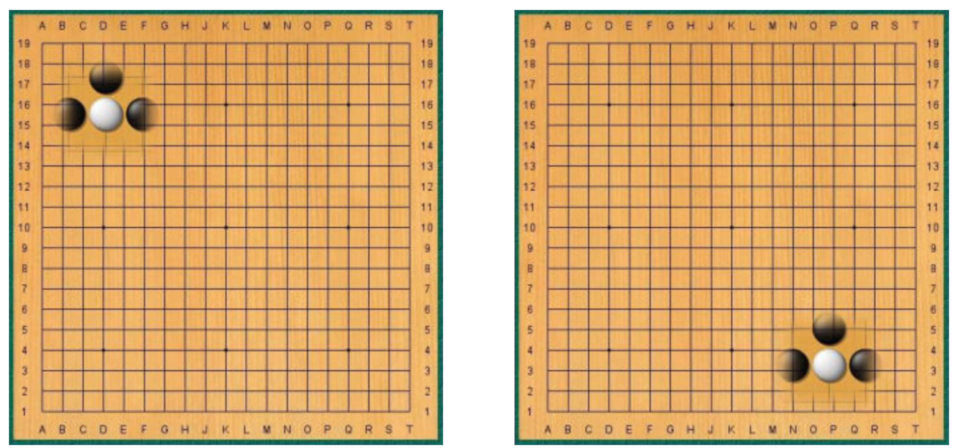
Why CNN for playing Go?

- 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

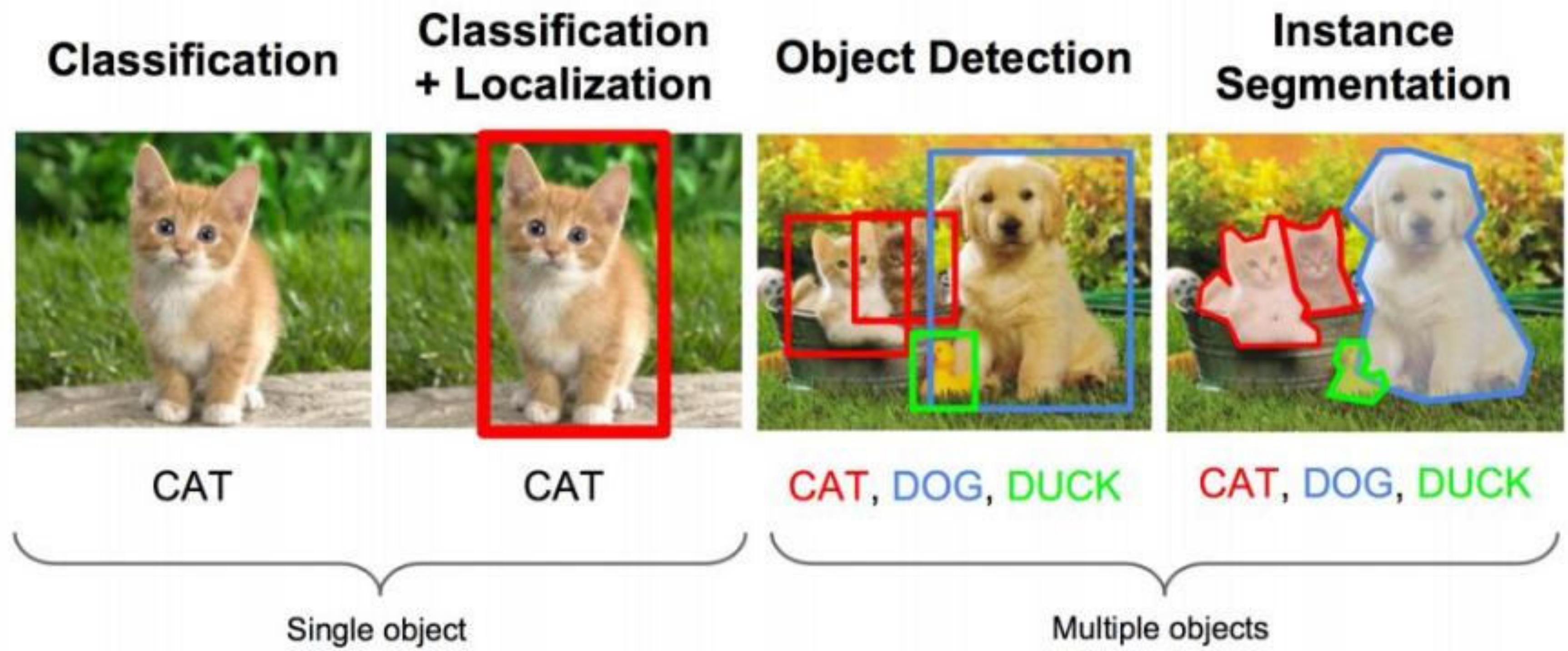
→ Max Pooling

How to explain this???

Neural network architecture. The input to the policy network is a 19 × 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 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 ……

Main Application: Object Detection



Main Application: Object Detection

Anchor based

1. One-stage

- **YOLO**
- **SSD**
- **Retina**

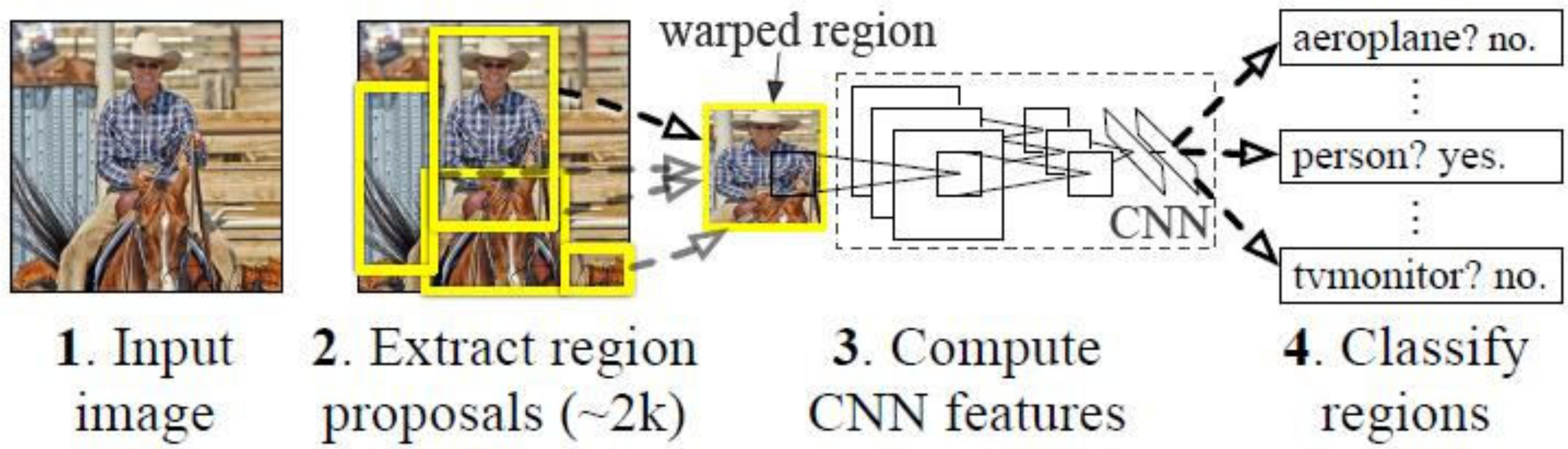
2. Two-stage

- R-CNN
- SPP-NET
- Fast R-CNN
- Faster R-CNN
- FPN

Anchor Free

- Corner Net （ECCV 2018）
- Feature Selective Anchor-Free Module （CVPR 2019）
- Center Net（arxiv 2020.04.16 ）
- FCOS (CVPR 2020）

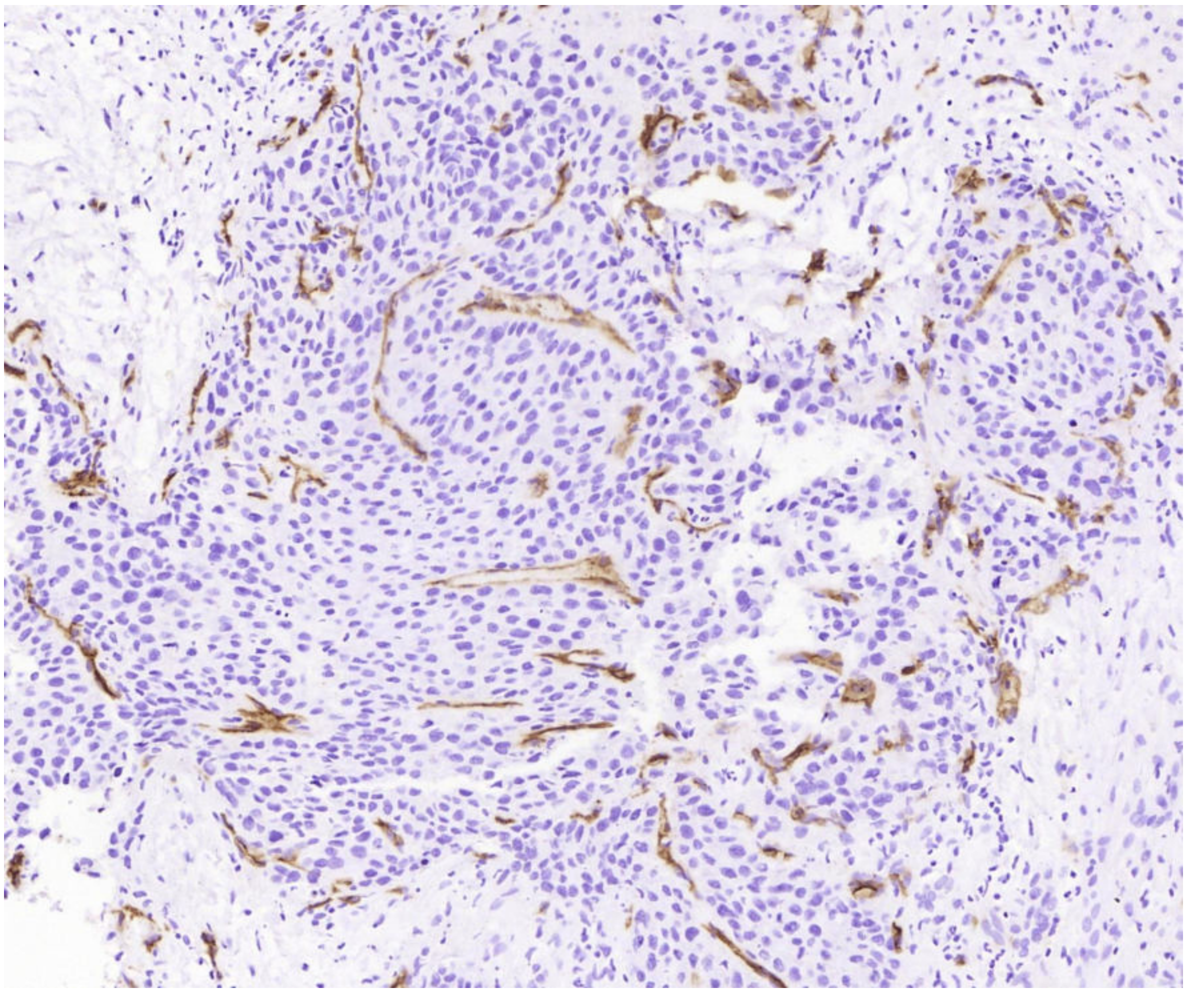
Main Application: Object Detection



RCNN算法流程分为4个步骤:

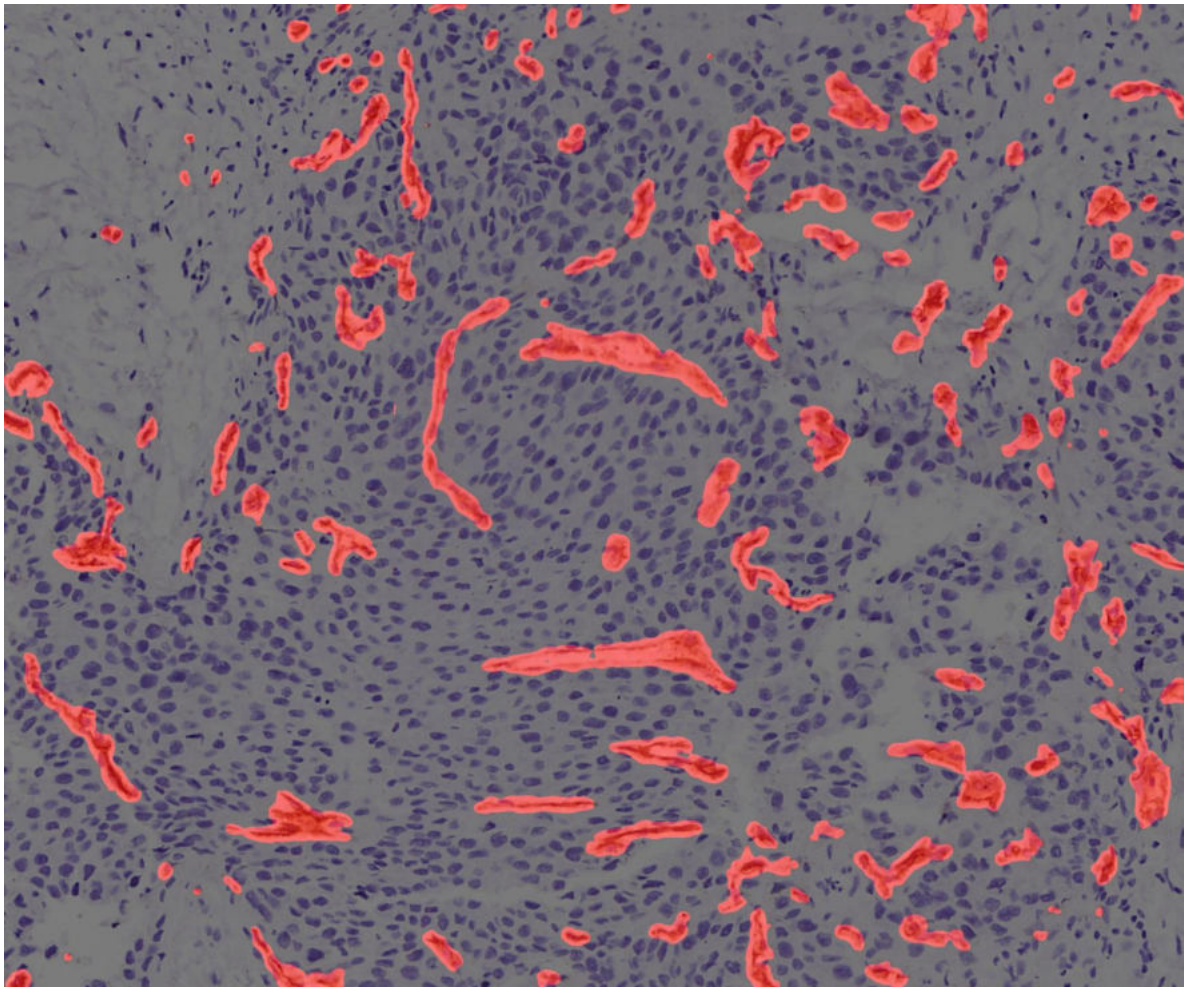
- 使用Selective Search方法对图像产生1K~2K个候选区域
- 对每个候选区域，使用深度网络提取特征
- 特征送入每一类的SVM分类器，判别是否属于该类
- 使用回归器精细修正候选框位置

Main Application: Object Segmentation



微血管个数：？
微血管周长：？
微血管面积：？
微血管密度：？

Main Application: Object Segmentation



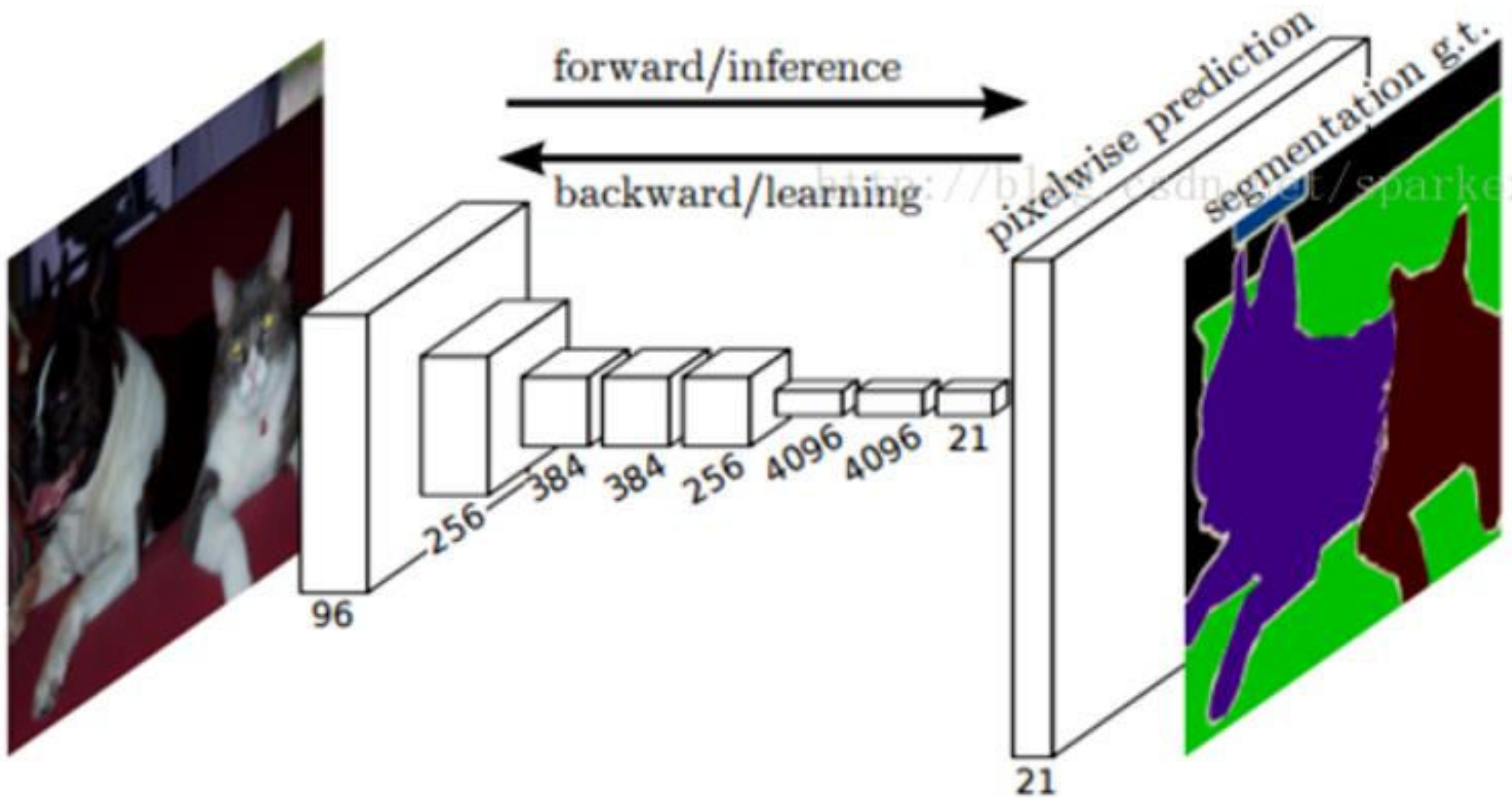
微血管个数: 103
微血管周长: 337.90μm
微血管面积: 53303.51μm^2
微血管面积比:8.593%

Main Application: Object Segmentation

- FCN 《Fully Convolutional Networks for Semantic Segmentation》 2014
- SegNet 《A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation》 2015
- U-Net 《Convolutional Networks for Biomedical Image Segmentation》 2015
- Deeplab V1 《Semantic image segmentation with deep convolutional nets and fully connected CRFs》 2015
  V2 《DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs》 2016
  V3 《Rethinking Atrous Convolution for Semantic Image Segmentation》 2017
  V3+ 《Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation》 2018

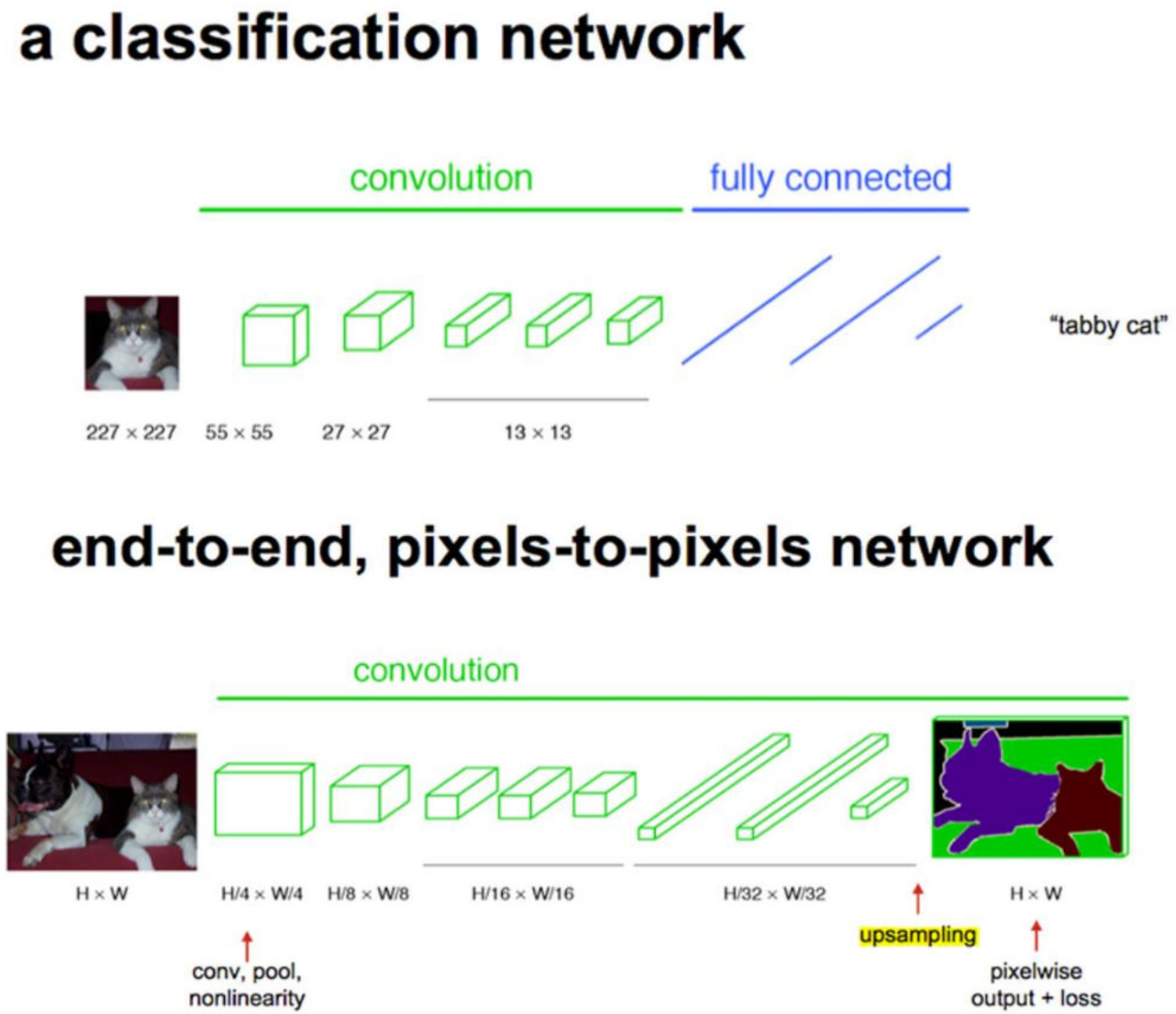
- Mask R-CNN 《Mask R-CNN》 2017

Main Application: Object Segmentation



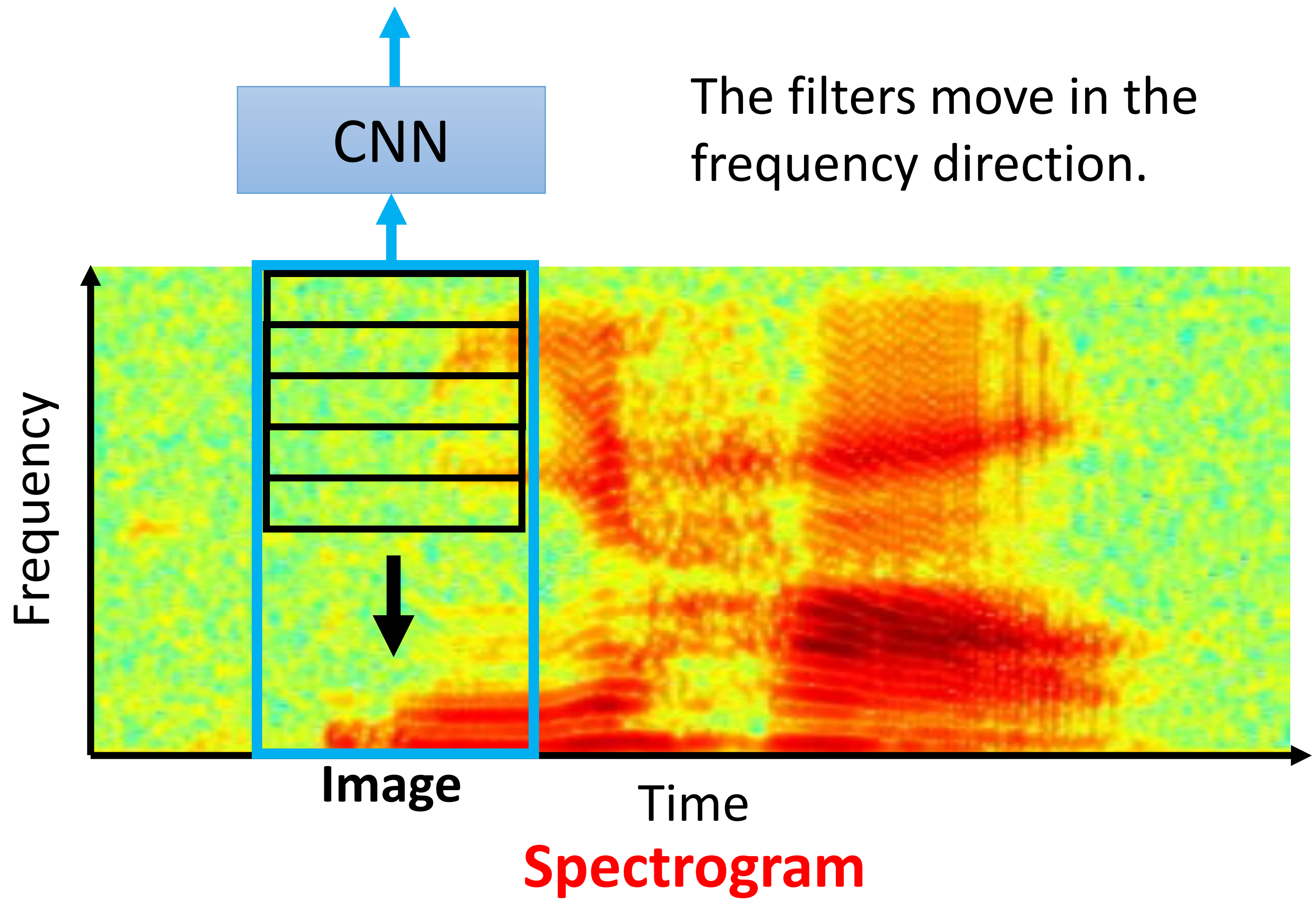
FCN从抽象的特征中恢复出每个像素所属的类别

Main Application: Object Segmentation

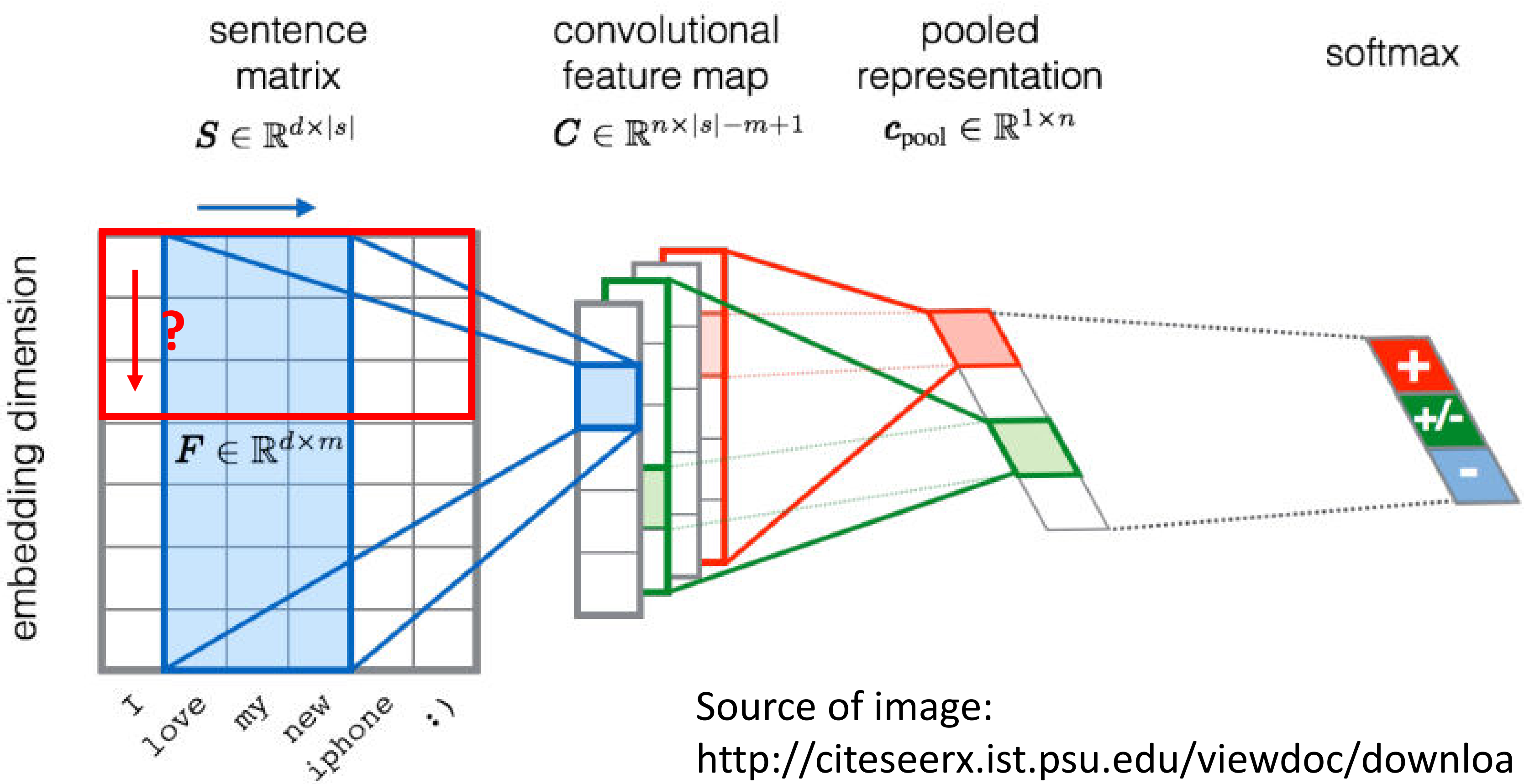


More Application: Speech

The filters move in the frequency direction.

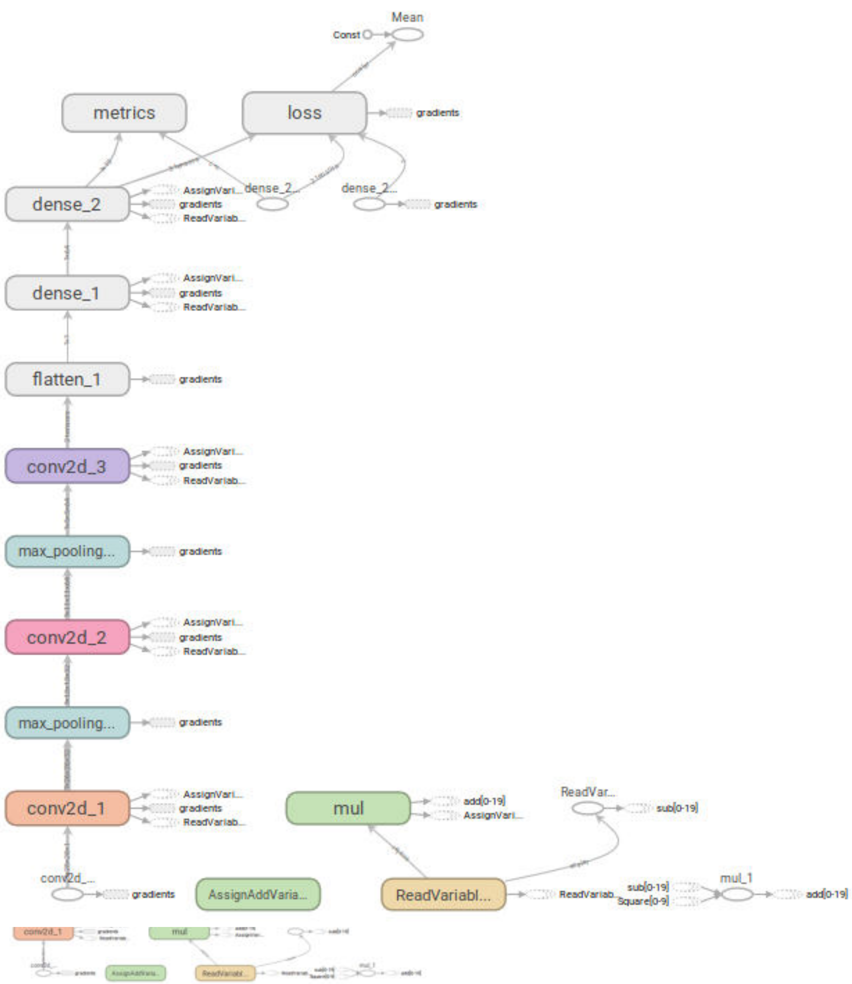


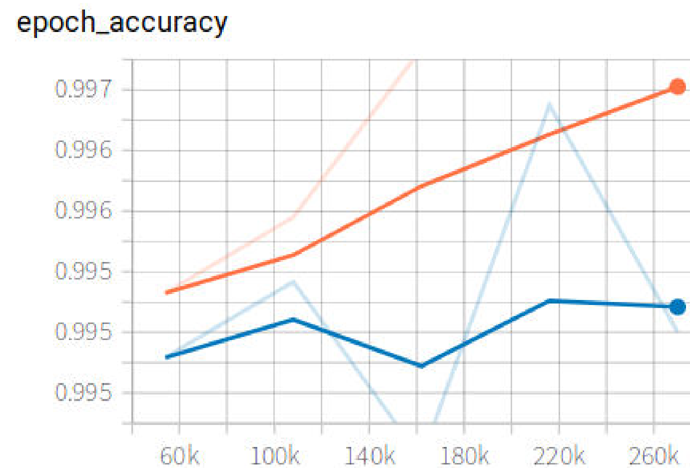
More Application: Text



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

tensorboard可视化



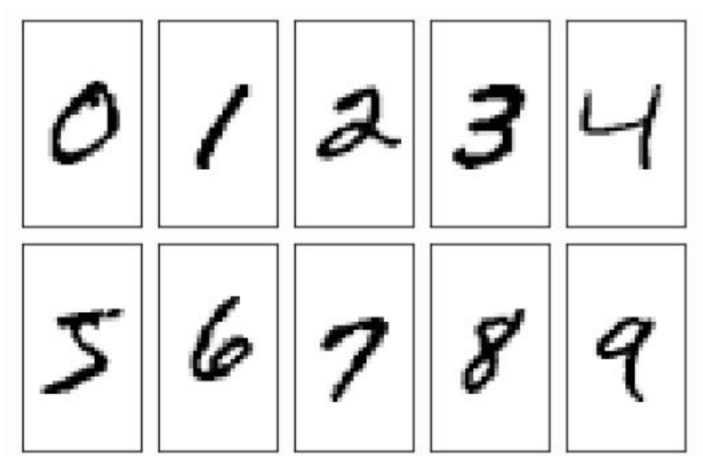


Convolutional Neural Network 回顾

1. CNN设计的动机是什么？
   - 局部特征图
   - 相同特征位置不同
   - 下采样不影响目标
2. CNN的架构
   - CNN操作
   - Max pooling
   - Flatten层
3. CNN的实现
   - Keras示例；pytorch、tf都可
4. CNN学到的是什么？
   - 局部特征；filter可视化
5. CNN的应用
   - 图像分析
   - 语音识别
   - 自然语言处理
6. 作业

作业

使用CNN卷积神经网络**实现**MNIST**手写字体识别**，并尽可能的实现CNN网络模型的可视化，此外，需结合第二章学习到的模型评估方法对实验结果进行评估，并给出评估结果。



THANKS！