

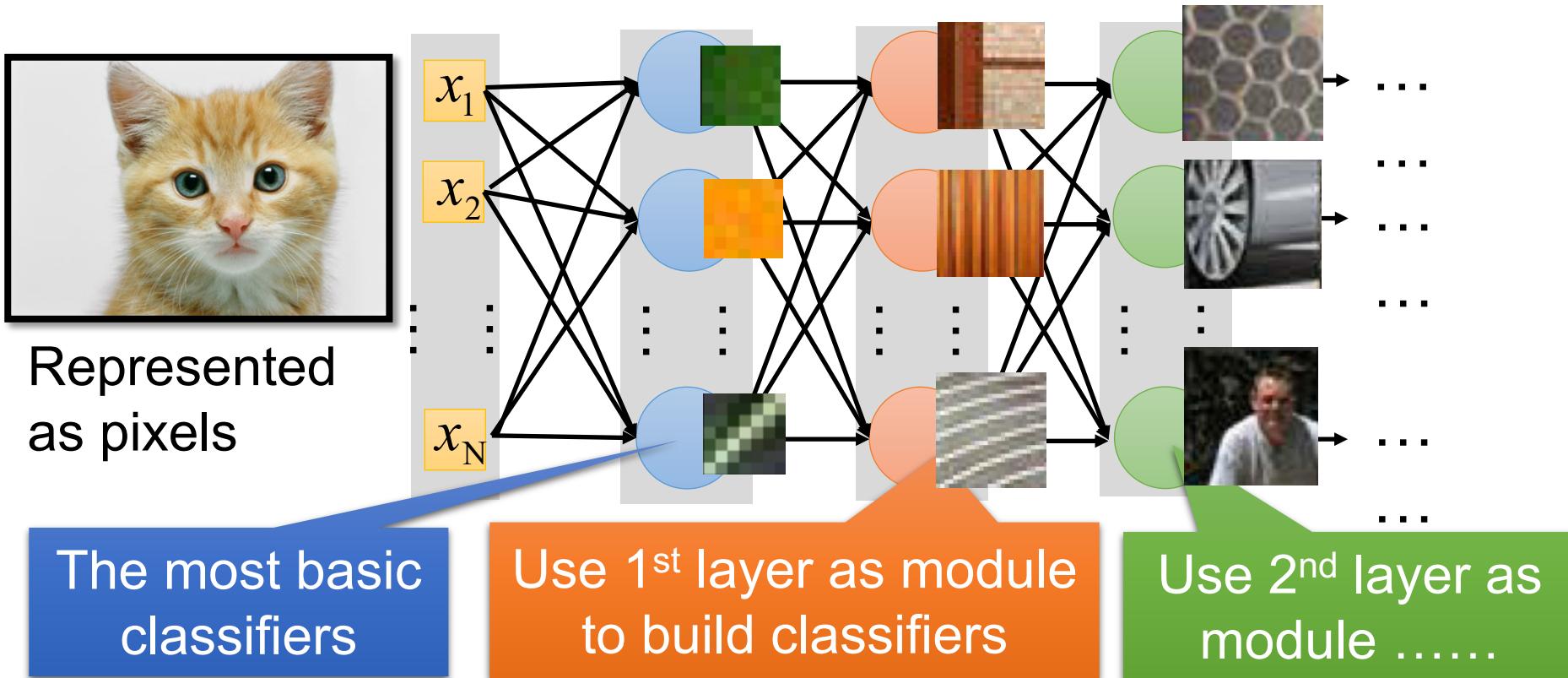
# Deep learning

## -- Convolutional Neural Network (CNN)

Junjie Cao @ DLUT  
Spring 2019

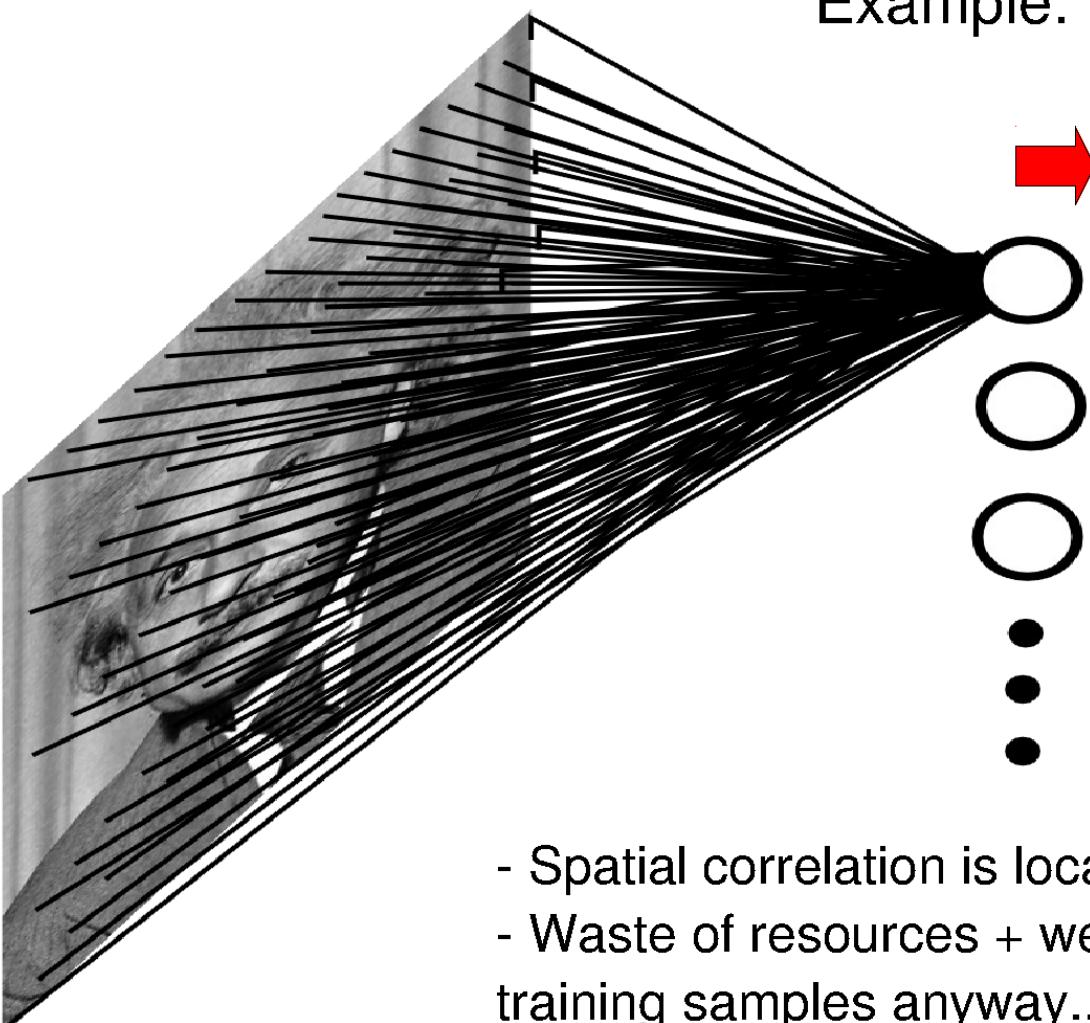
# Why CNN for Image?

[Zeiler, M. D., ECCV 2014]



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

# Images as input to neural networks



Example: 200x200 image

40K hidden units

**~2B parameters!!!**

What can we do?

- Subsampling?
- Pruning: an in-processing or postprocessing operation?
- Design a better network from beginning?

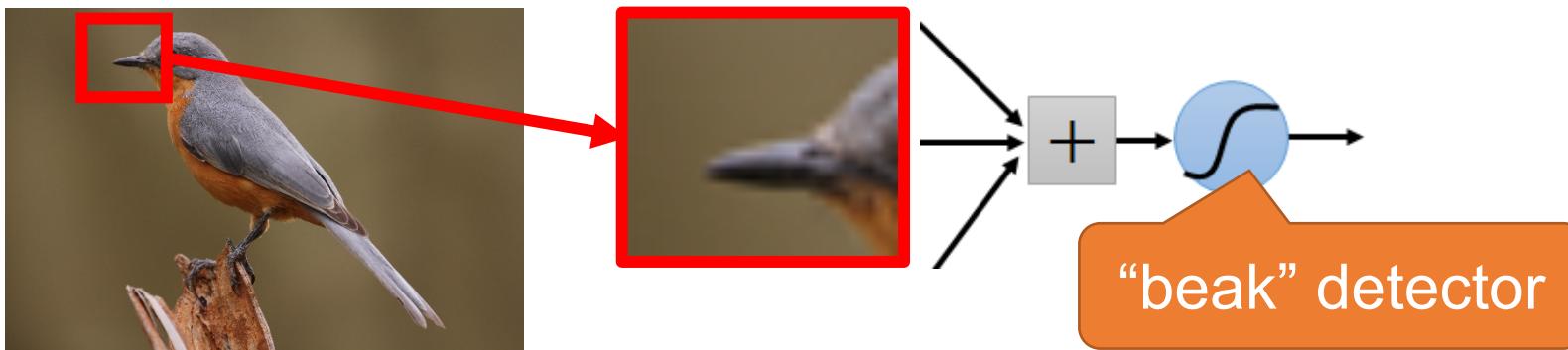
- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

# Full connection may be not suitable for Images

- **Sparse interactions:** some patterns are much smaller than the whole image

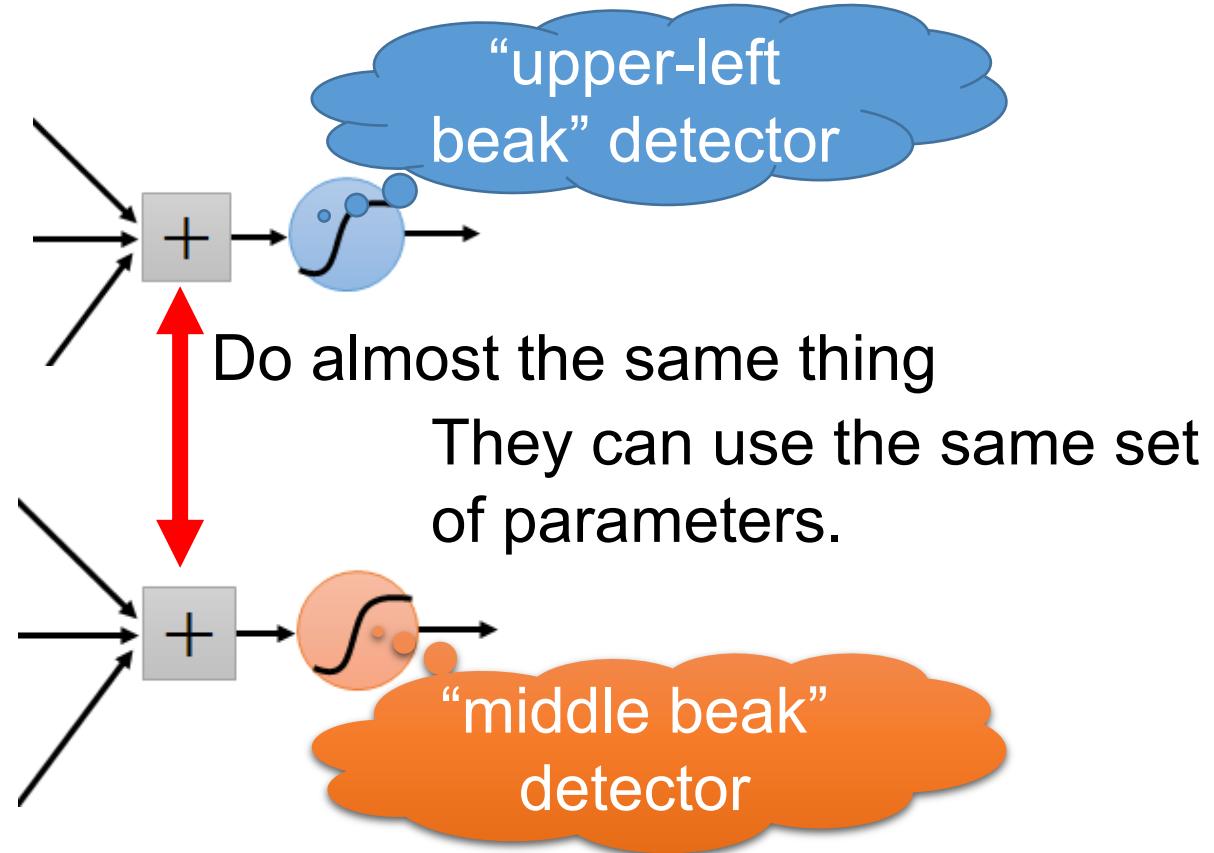
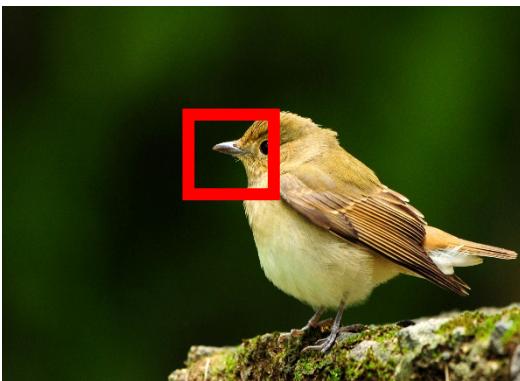
A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters



# Full connection may be not suitable for Images

- The **same patterns** appear in **different locations**.

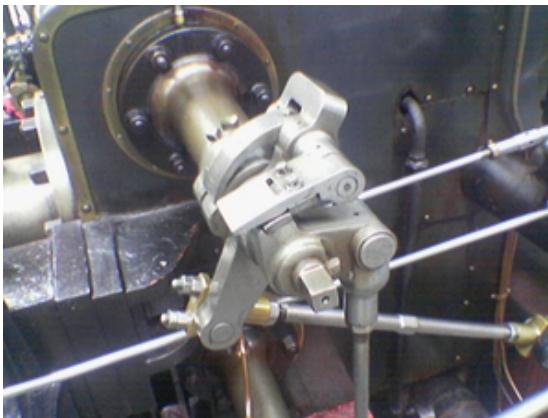


# Why convolution are useful

Allow us to find **interesting insights/features** from images!

Example:

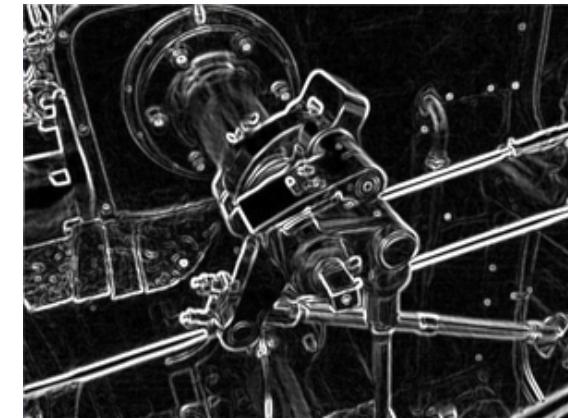
- Abstraction: remove lighting, material, etc. in some degree
- with same structure: edges with horizontal component



\*

0	$-\frac{1}{2}$	0
0	0	0
0	$\frac{1}{2}$	0

=



# Other operation for reducing parameters

- Subsampling/pooling the pixels will not change the object

bird



bird



subsampling

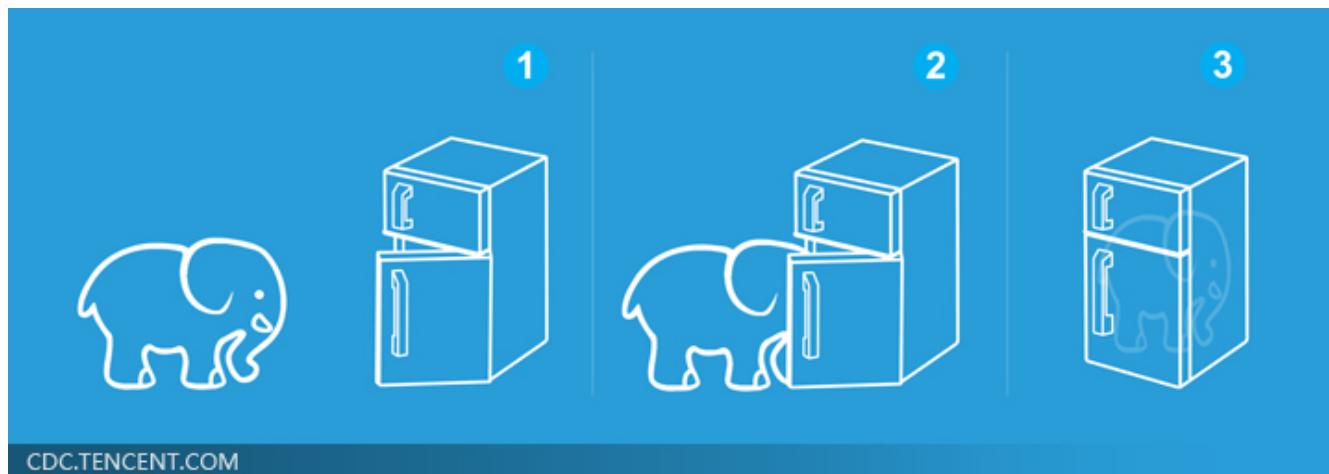
We can subsample the pixels to make image smaller

→ **Less parameters** for the network to process the image

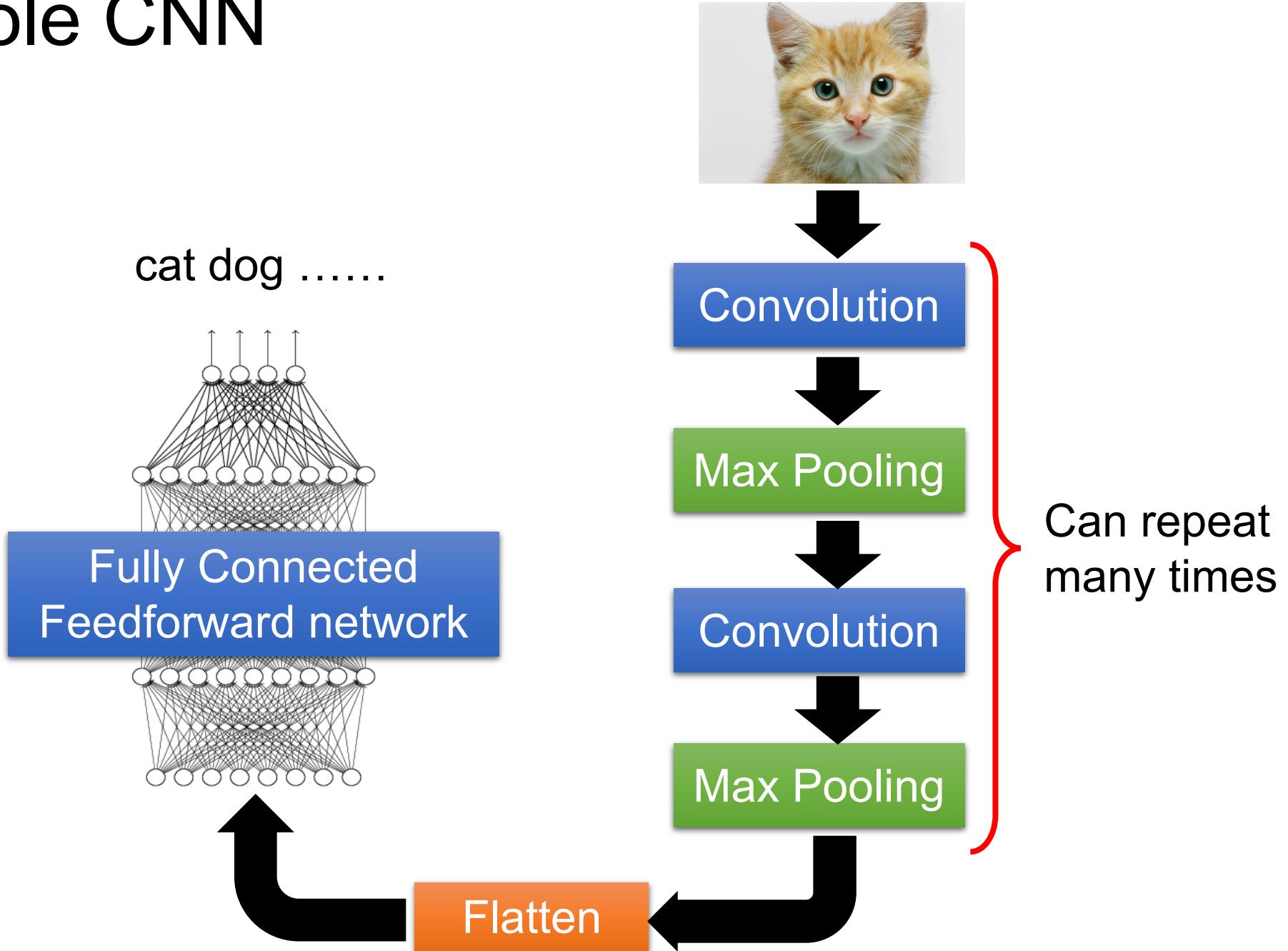
# Three Steps for Deep Learning



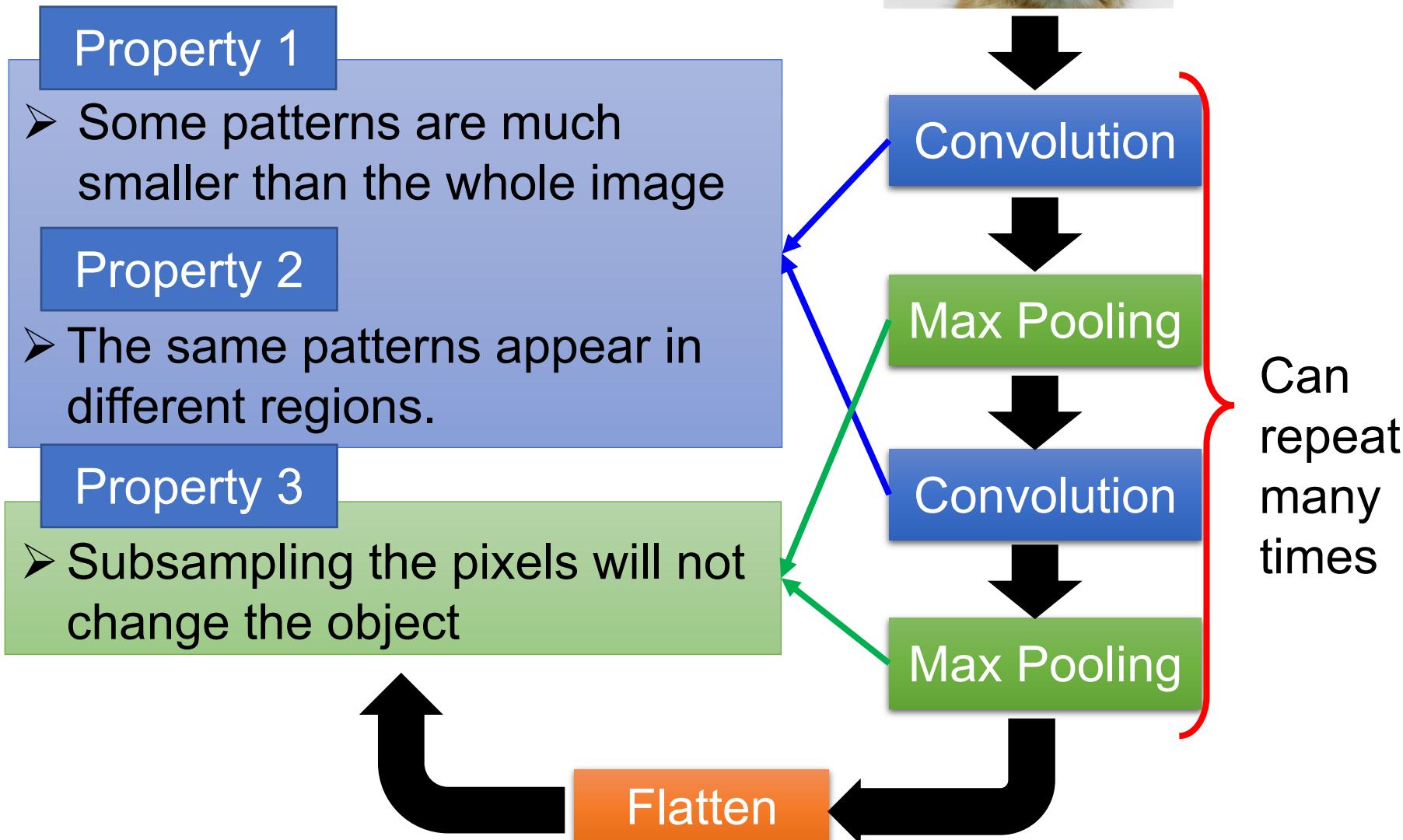
Deep Learning is so simple .....



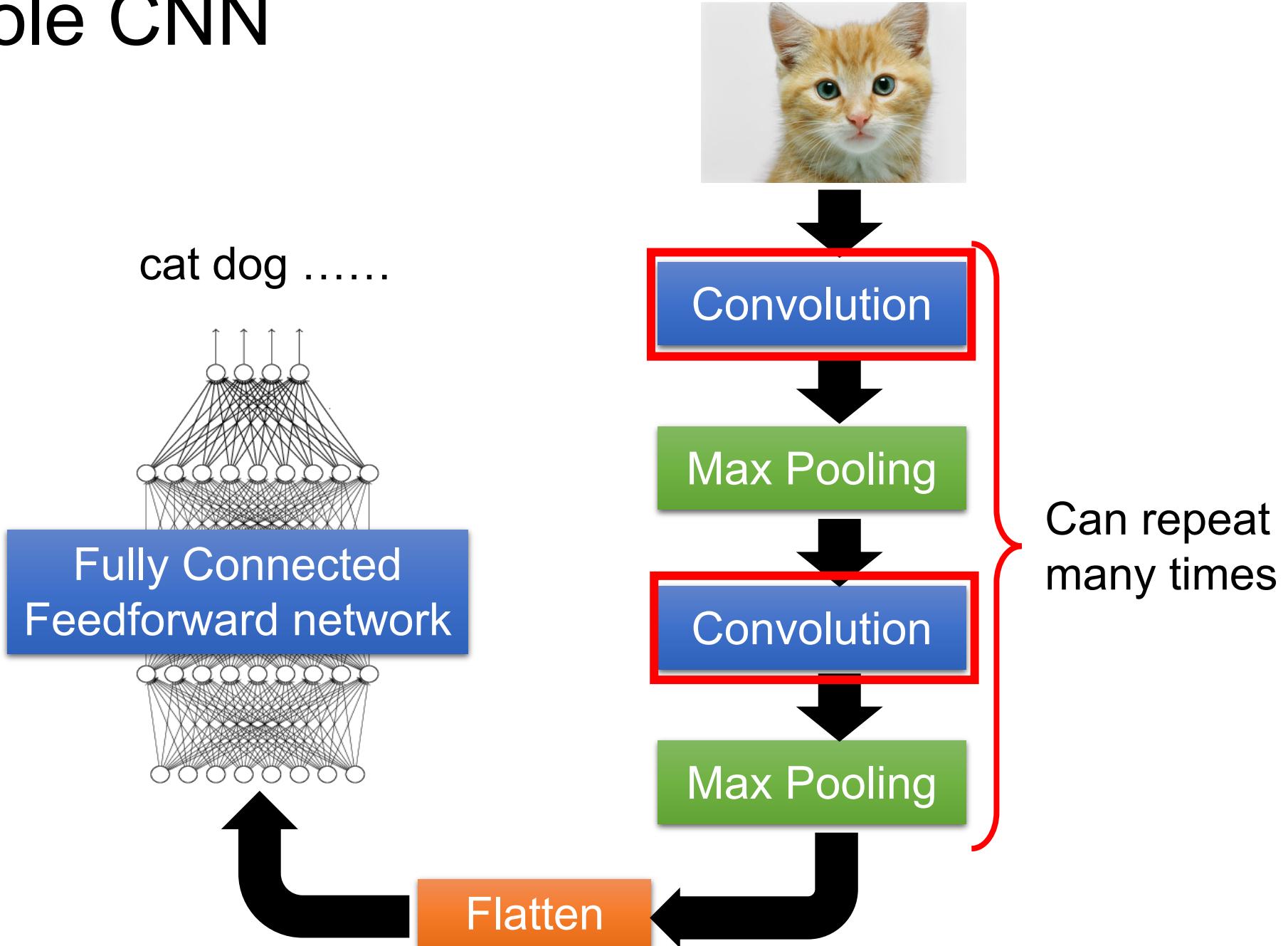
# The whole CNN



# The whole CNN



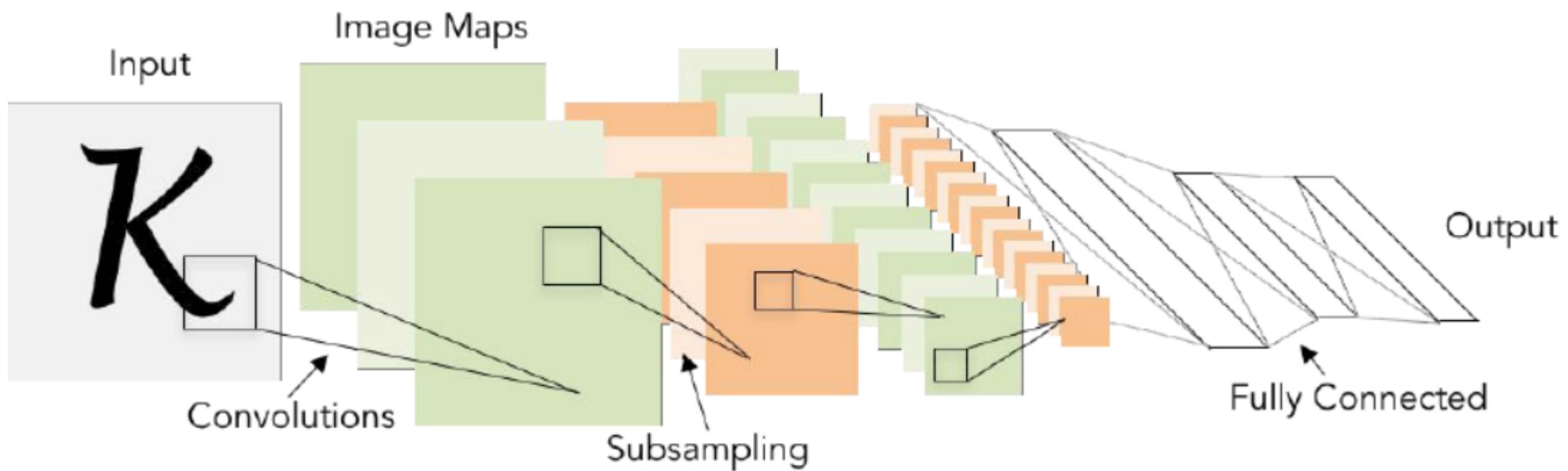
# The whole CNN



# A bit of history: LeNet-5

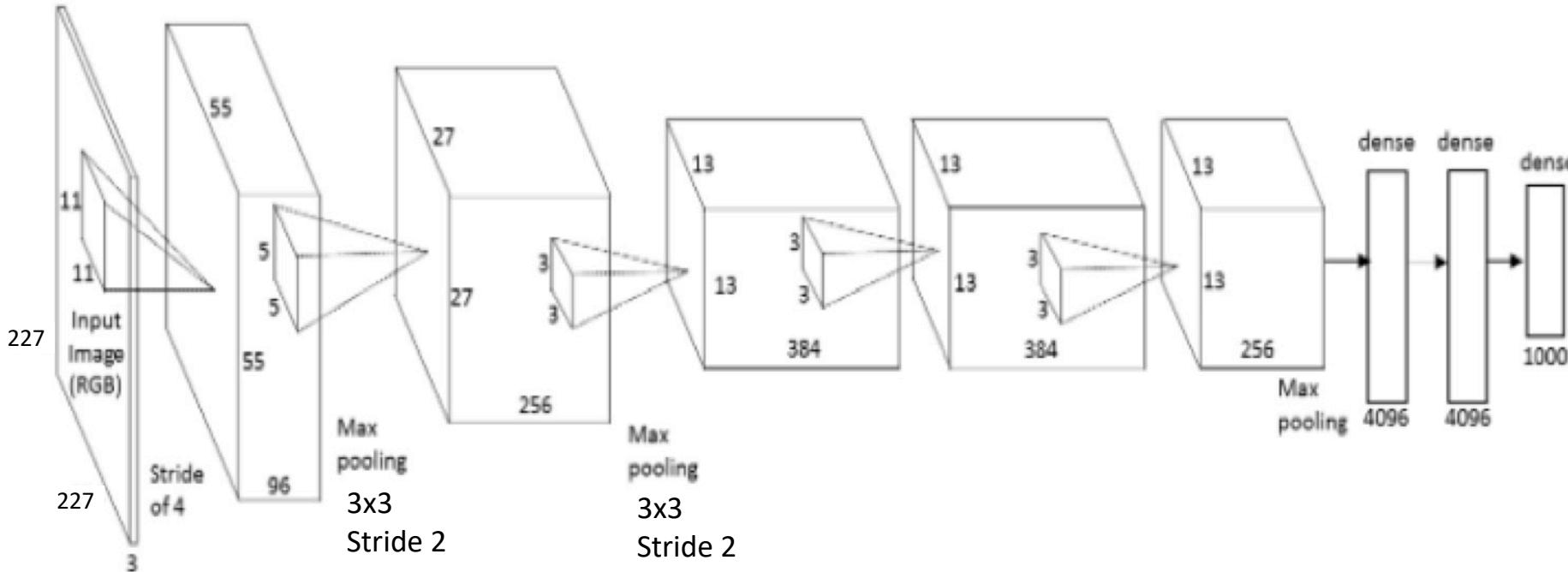
- Gradient-based learning applied to document recognition

*[LeCun, Bottou, Bengio, Haffner 1998]*



# A bit of history: AlexNet diagram (simplified)

Input size  
227 x 227 x 3



**Conv 1**

$11 \times 11 \times 3$

Stride 4

96 filters

**Conv 2**

$5 \times 5 \times 96$

Stride 1

256 filters

**Conv 3**

$3 \times 3 \times 256$

Stride 1

384 filters

**Conv 4**

$3 \times 3 \times 192$

Stride 1

384 filters

**Conv 4**

$3 \times 3 \times 192$

Stride 1

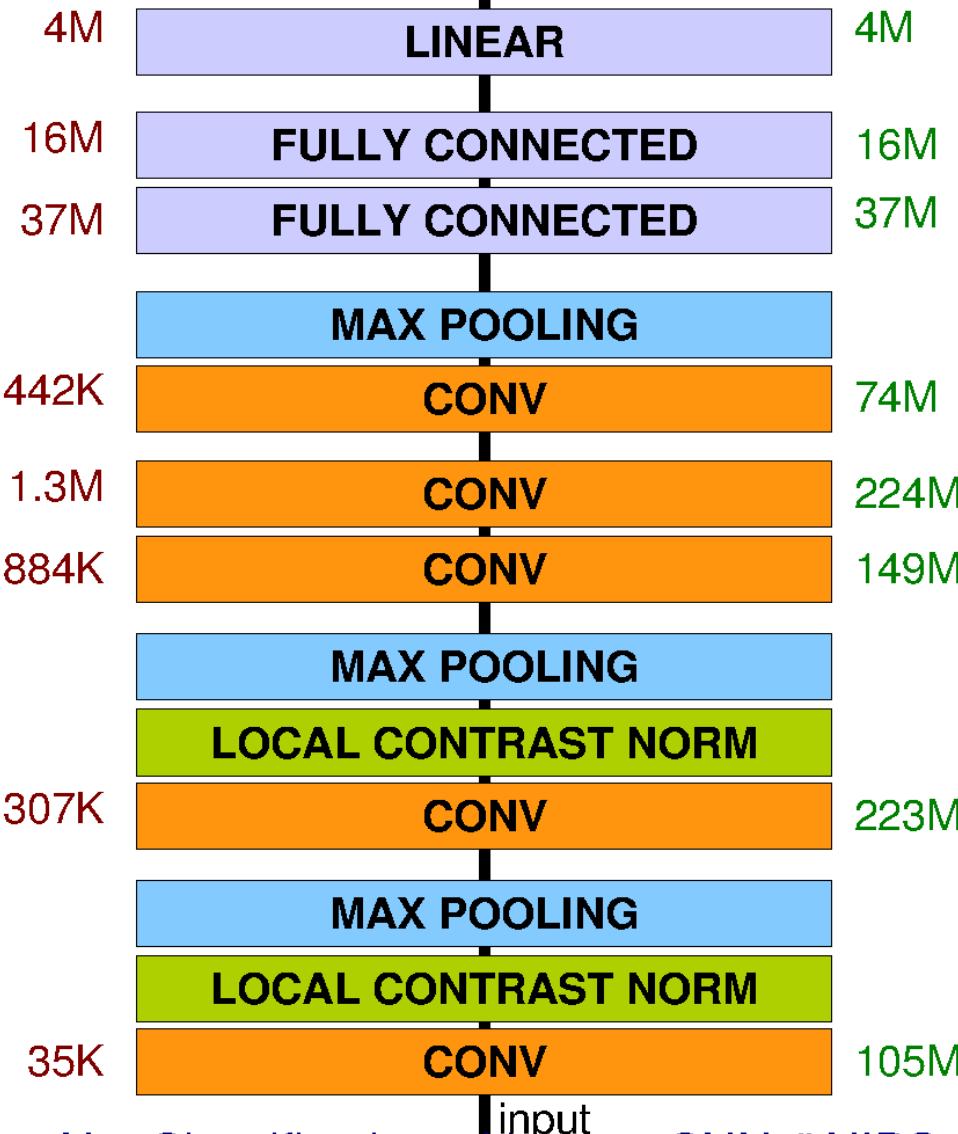
256 filters

# Architecture for Classification

Total nr. params: 60M

category  
prediction

Total nr. flops: 832M

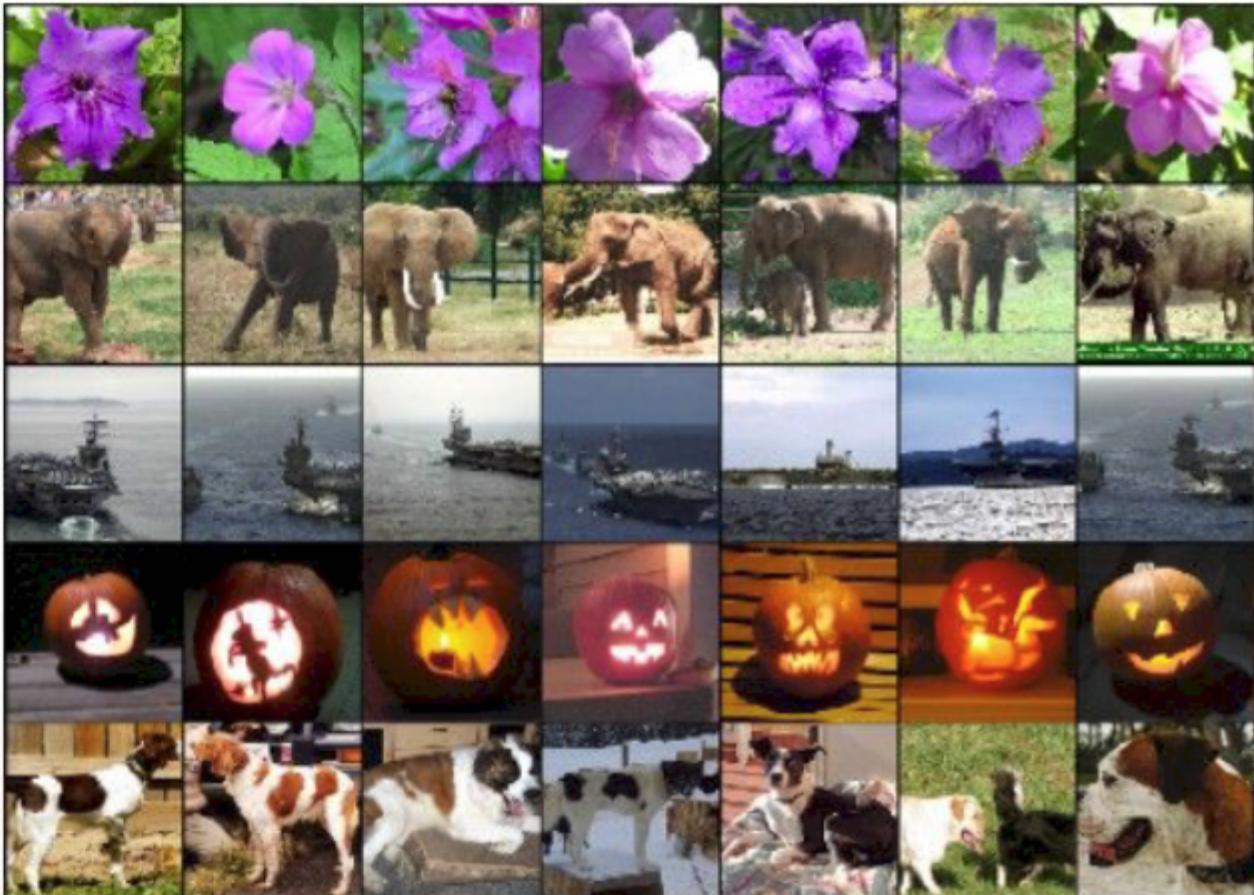


# Fast-forward to today: ConvNets are everywhere

Classification



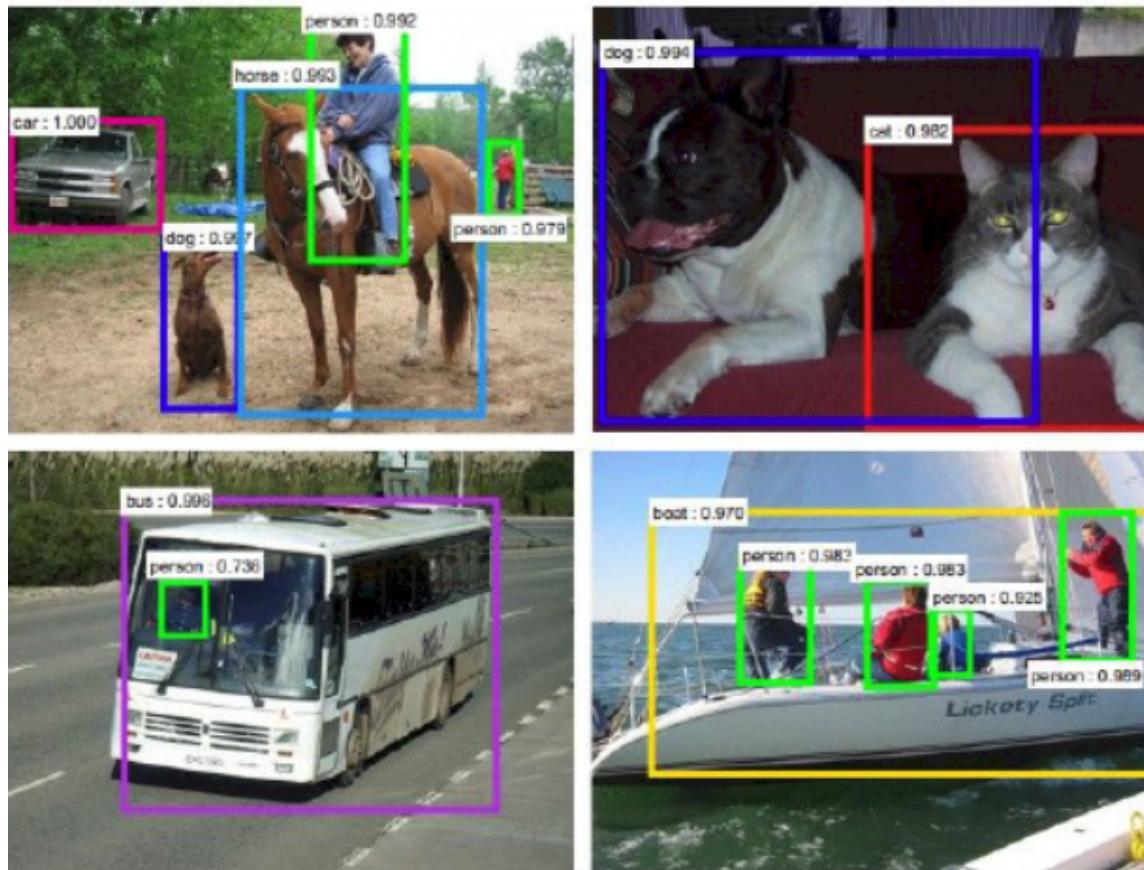
Retrieval



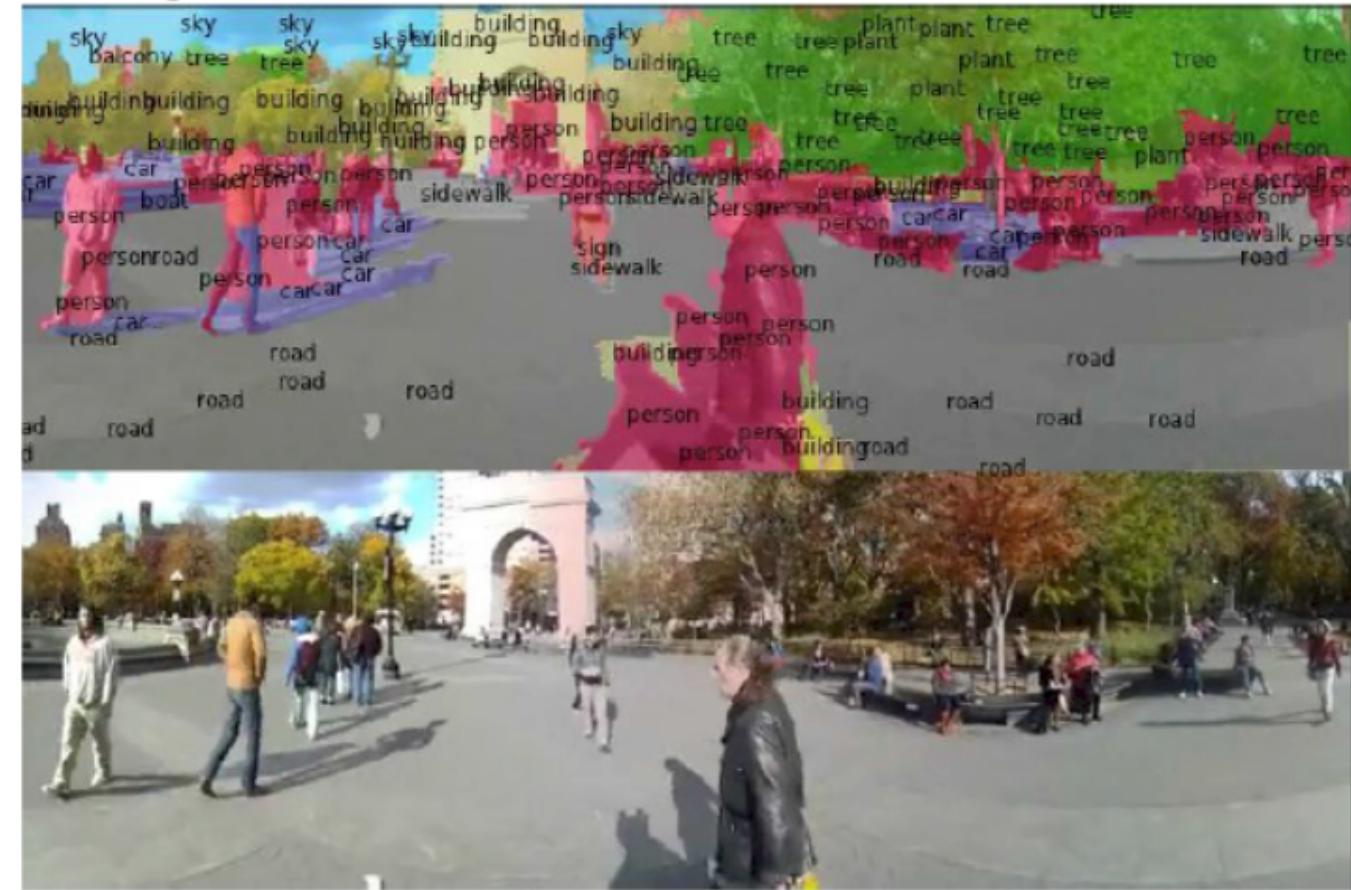
Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

# Fast-forward to today: ConvNets are everywhere

## Detection



## Segmentation



Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015. Reproduced with permission.

[*Faster R-CNN: Ren, He, Girshick, Sun 2015*]

Figures copyright Clement Farabet, 2012.  
Reproduced with permission.

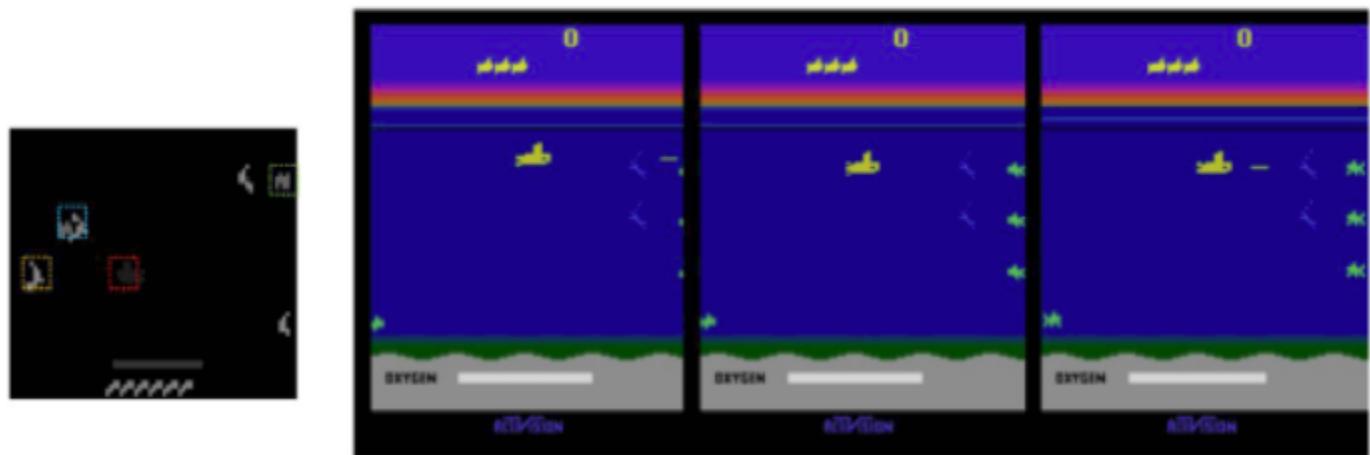
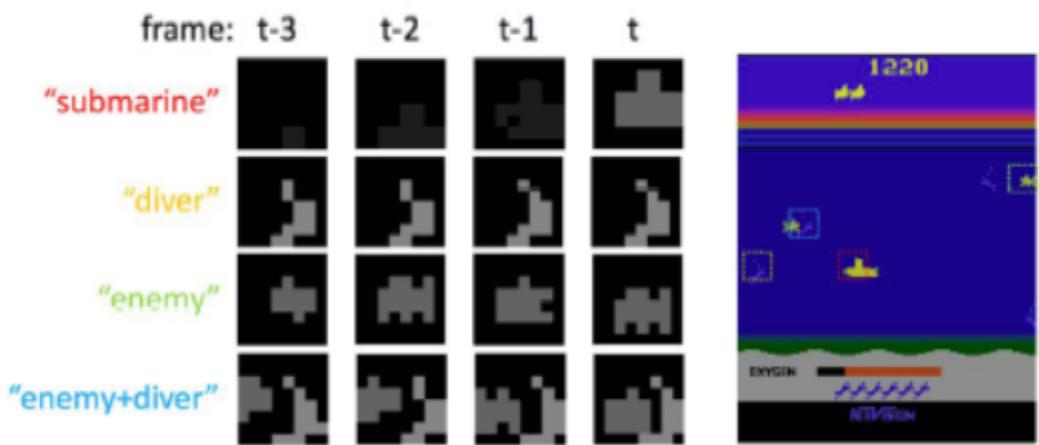
[*Farabet et al., 2012*]

# Fast-forward to today: ConvNets are everywhere



Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]



[Guo et al. 2014]

Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

# Fast-forward to today: ConvNets are everywhere

No errors



*A white teddy bear sitting in the grass*



*A man riding a wave on top of a surfboard*

Minor errors



*A man in a baseball uniform throwing a ball*



*A cat sitting on a suitcase on the floor*

Somewhat related



*A woman is holding a cat in her hand*



*A woman standing on a beach holding a surfboard*

Image  
Captioning

[Vinyals et al., 2015]  
[Karpathy and Fei-Fei, 2015]

All images are CC0 Public domain:  
<https://pixabay.com/en/luggage-antique-cat-1643010/>  
<https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1823436/>  
<https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/>  
<https://pixabay.com/en/woman-female-model-portrait-adult-983987/>  
<https://pixabay.com/en/handstand-lake-meditation-496008/>  
<https://pixabay.com/en/baseball-player-shortstop-infield-1045263/>

Captions generated by Justin Johnson using [Neuraltalk2](#)

# 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

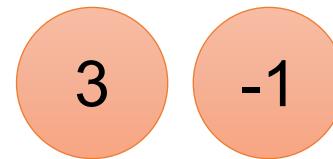
stride=1

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

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

Filter 1



# CNN – Convolution

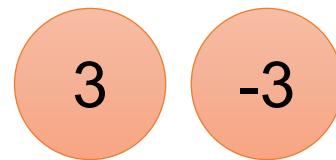
If stride=2

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

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

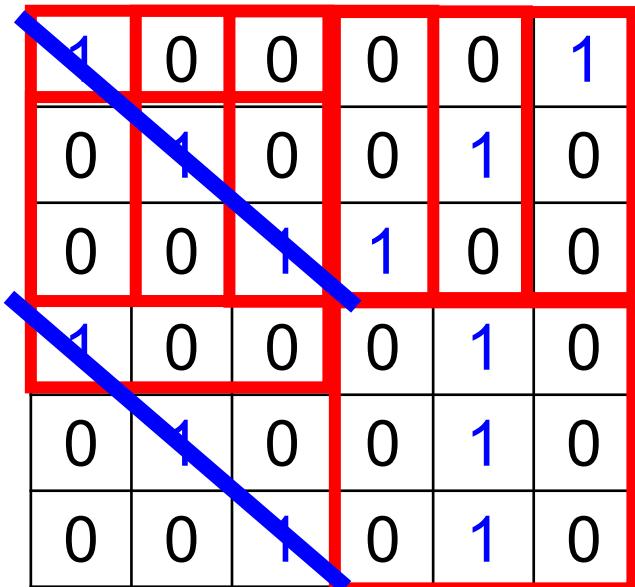
Filter 1



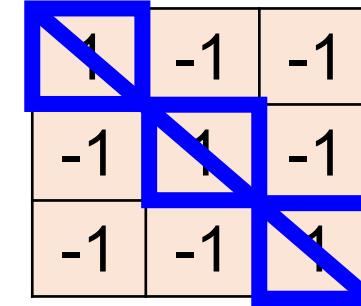
We set stride=1 below

# CNN – Convolution

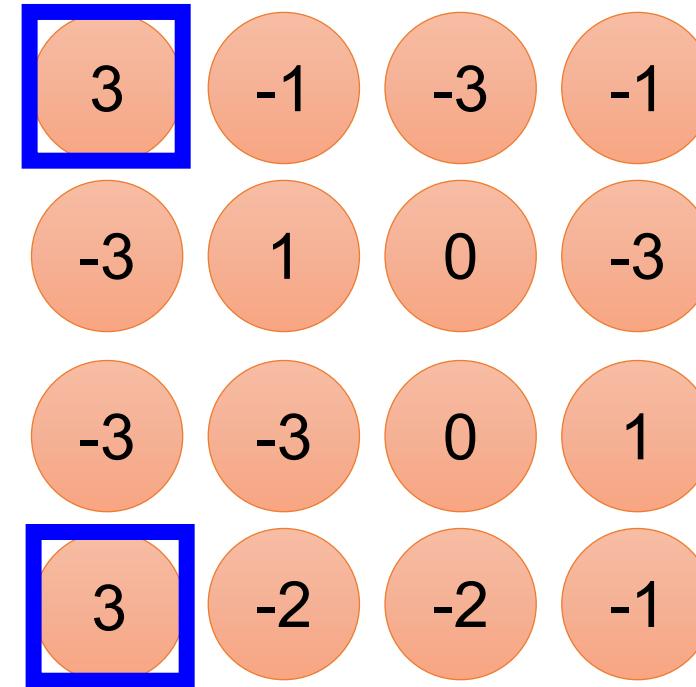
stride=1



6 x 6 image



Filter 1



Property 2

# CNN – Convolution

stride=1

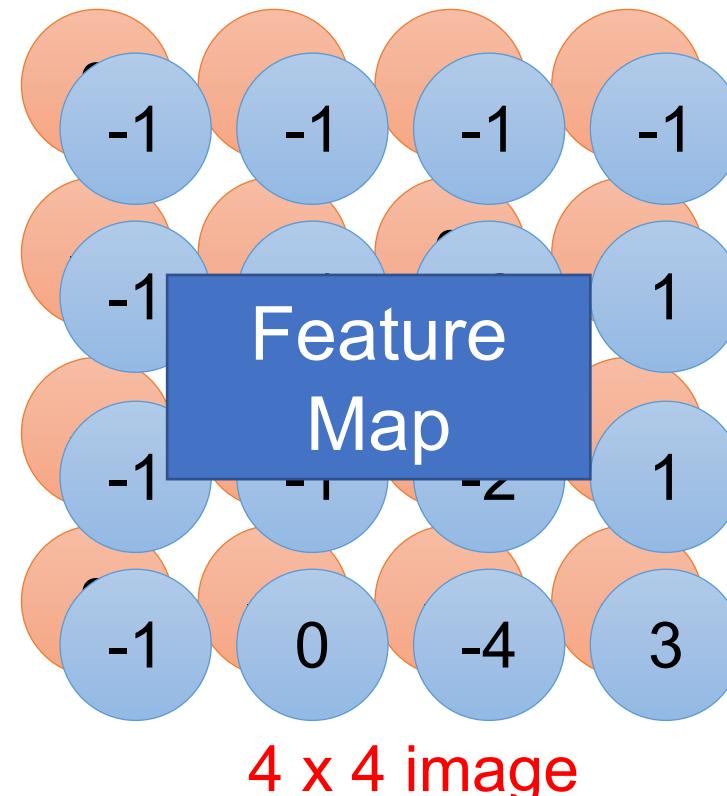
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

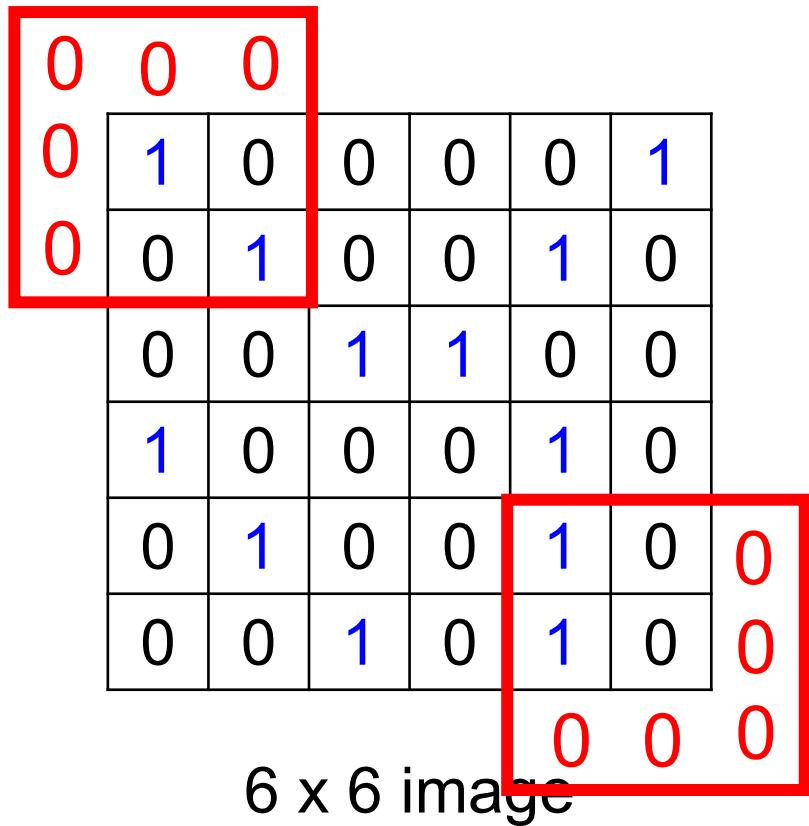
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Do the same process  
for every filter



# CNN – Zero Padding



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

Filter 1

You will get another 6 x 6 images in this way

Zero padding

# CNN – Colorful image

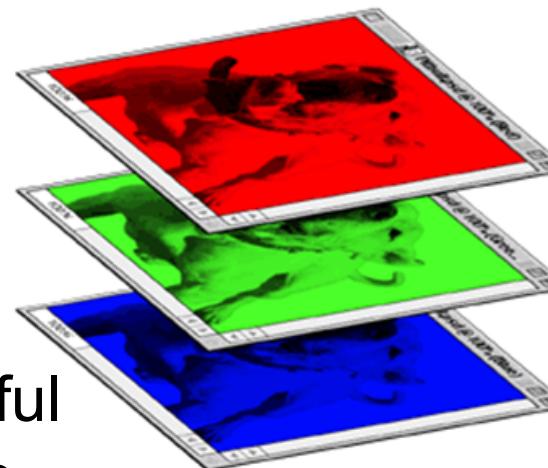
- 2d convolution => 3d convolution

0	0	-1
-1	1	-1
-1	-1	1

1	-1	-1
0	0	-1
-1	-1	1

1	-1	-1
3	1	-1
2	0	0

Filter 1



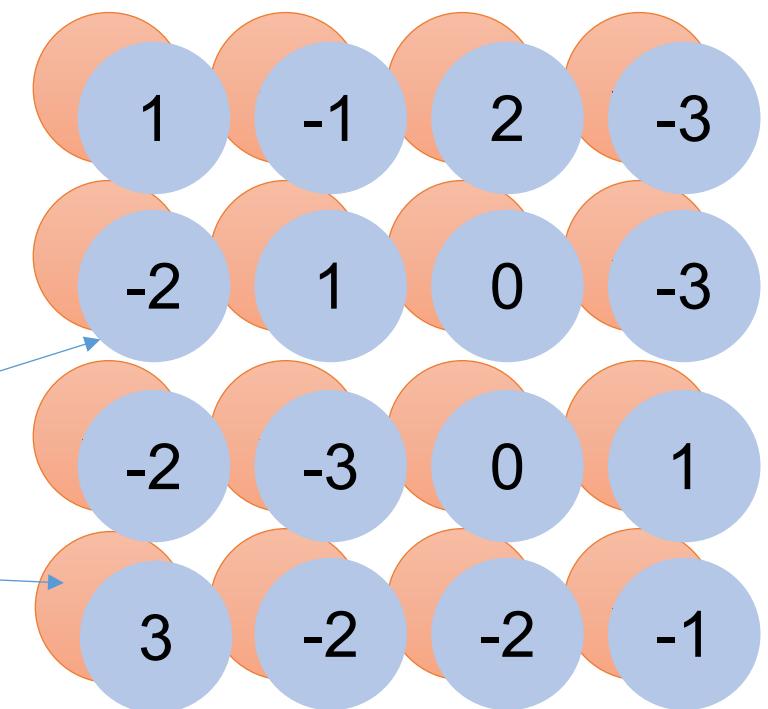
Colorful  
image



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

Filter 2

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



# Examples time:

- Input volume: **32x32x3**
- **10 5x5 filters with stride 1, pad 2**
- Output volume size: ?

**(32+2\*2-5)/1+1 = 32 spatially, so**

**32x32x10**

# Examples time:

- Input volume: **32x32x3**
- **10 5x5 filters with stride 1, pad 2**
- Number of parameters in this layer?

each filter has  **$5 \times 5 \times 3 + 1 = 76$  params (+1 for bias)**

$$\Rightarrow 76 \times 10 = 760$$

## Common settings:

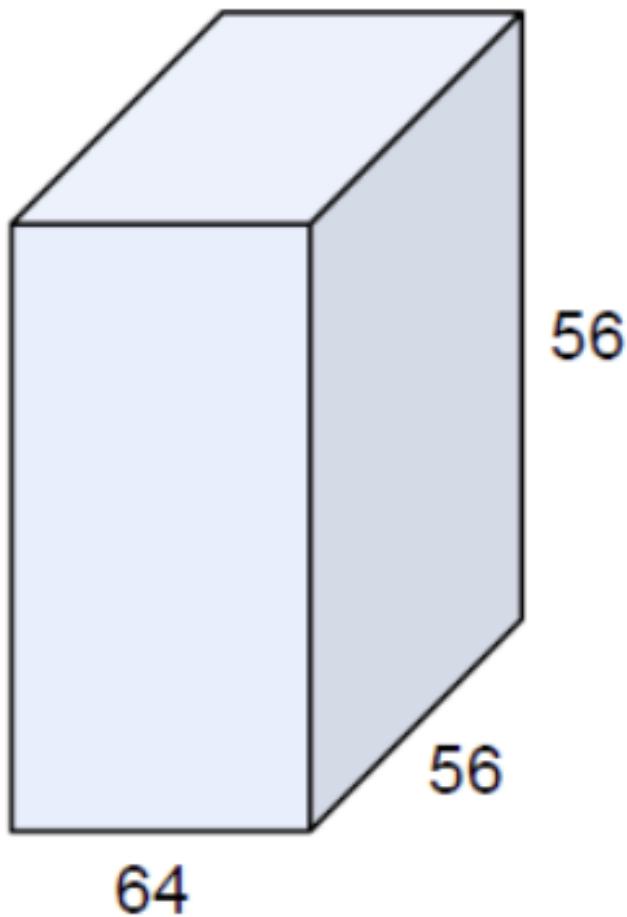
**Summary.** To summarize, the Conv Layer:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters  $K$ ,
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
  - the amount of zero padding  $P$ .
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$  (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces  $F \cdot F \cdot D_1$  weights per filter, for a total of  $(F \cdot F \cdot D_1) \cdot K$  weights and  $K$  biases.
- In the output volume, the  $d$ -th depth slice (of size  $W_2 \times H_2$ ) is the result of performing a valid convolution of the  $d$ -th filter over the input volume with a stride of  $S$ , and then offset by  $d$ -th bias.

$K = (\text{powers of 2, e.g. } 32, 64, 128, 512)$

- $F = 3, S = 1, P = 1$
- $F = 5, S = 1, P = 2$
- $F = 5, S = 2, P = ?$  (whatever fits)
- $F = 1, S = 1, P = 0$

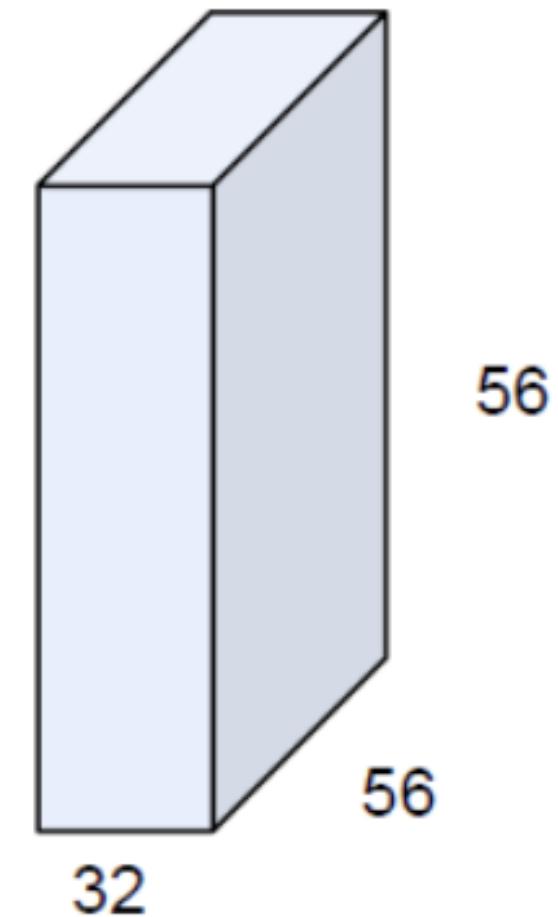
(btw, 1x1 convolution layers make perfect sense)



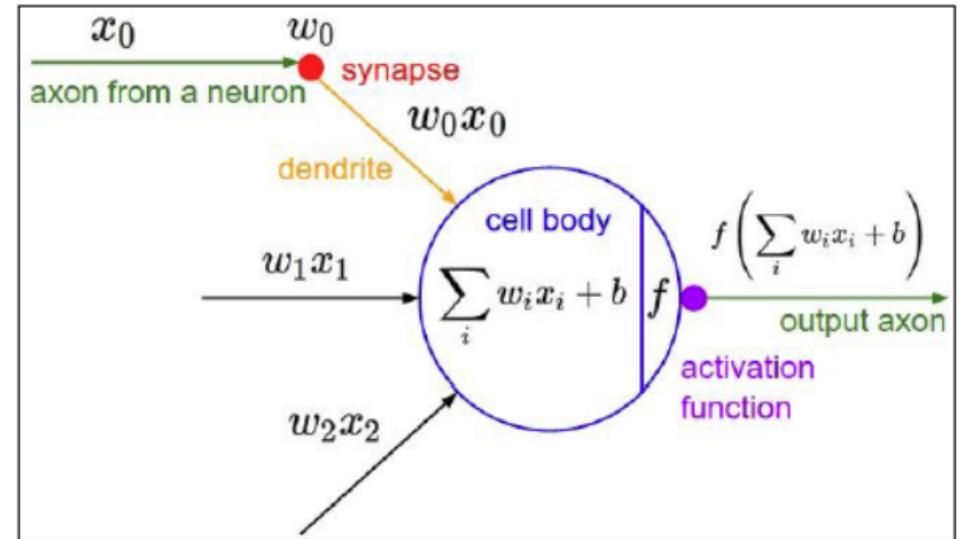
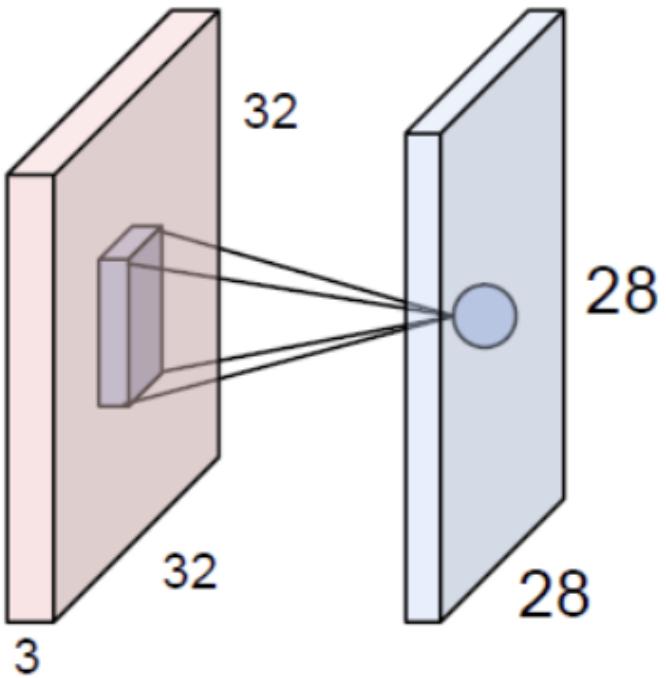
1x1 CONV  
with 32 filters

---

(each filter has size  
1x1x64, and performs a  
64-dimensional dot  
product)



# The brain/neuron view of CONV Layer

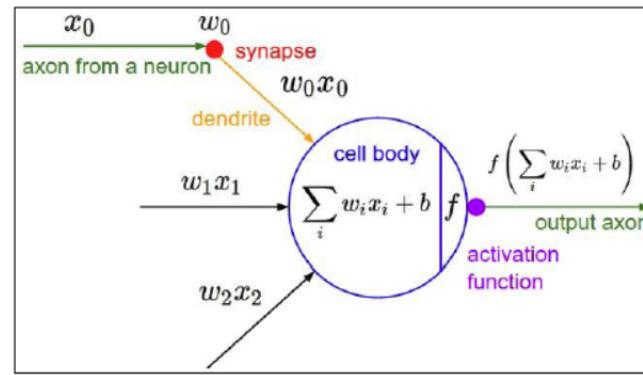
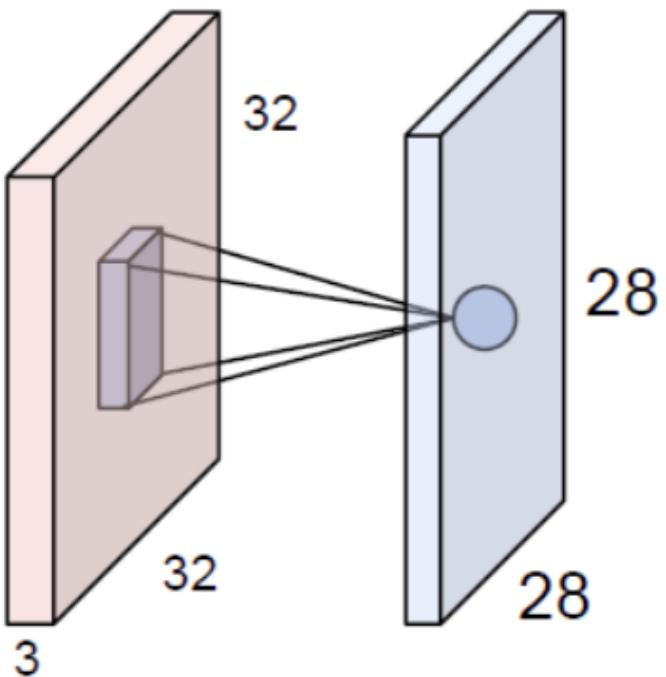


It's just a neuron with local connectivity...

**1 number:**

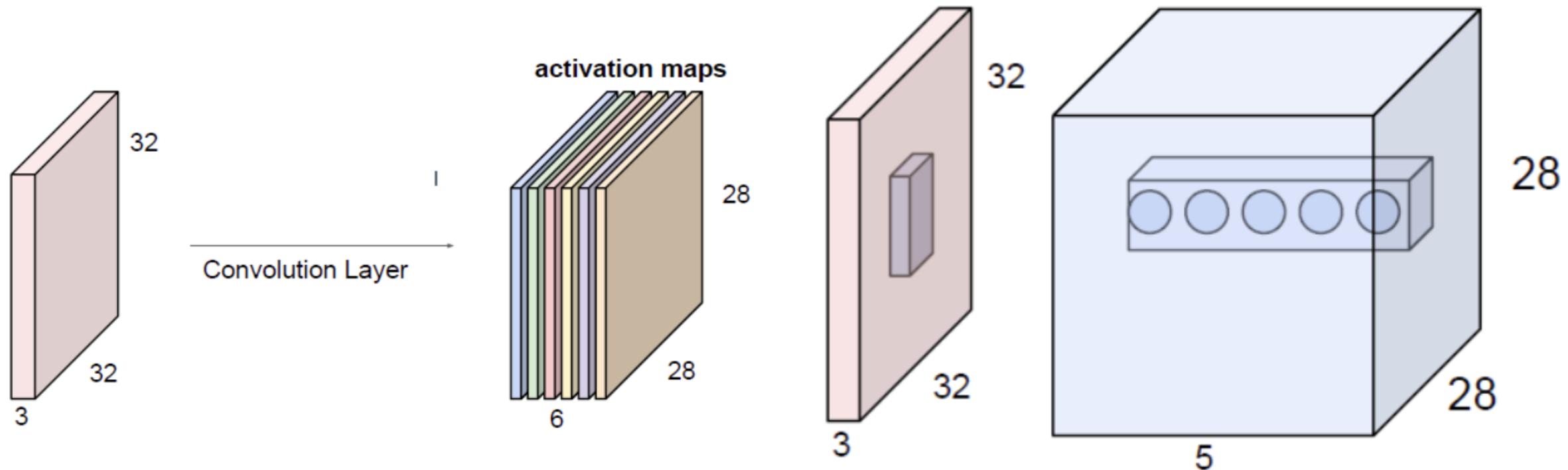
a dot product between the filter and this part of the image  
(i.e.  $5 \times 5 \times 3 = 75$ -dimensional dot product)

# The brain/neuron view of CONV Layer



- An activation map is a 28x28 sheet of neuron outputs:
  1. Each is connected to a small region in the input
  2. All of them share parameters
- “5x5 filter” -> “5x5 receptive field for each neuron”

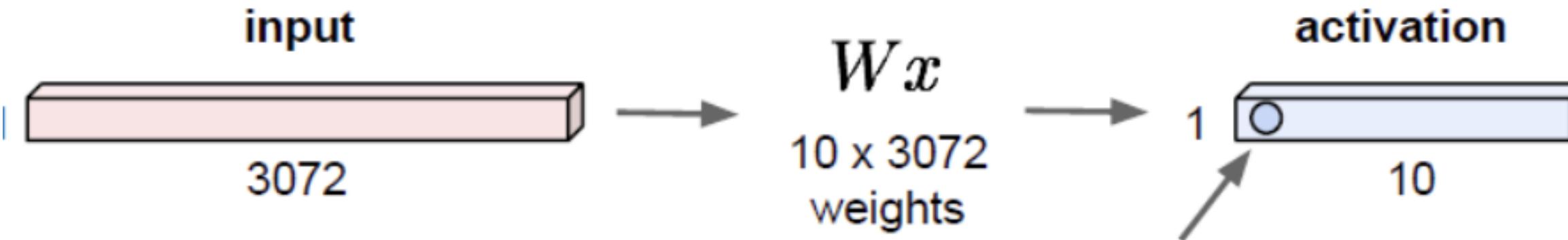
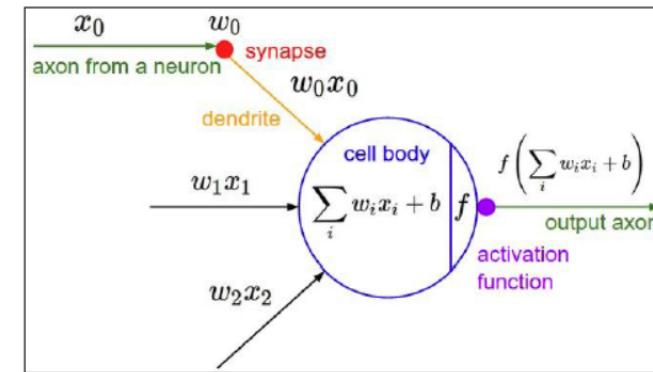
# The brain/neuron view of CONV Layer



- E.g. with 6 (left) or 5 (right) filters,
- CONV layer consists of neurons arranged in a 3D grid (28x28x5)
- There will be 5 different neurons all looking at the **same** region in the input volume

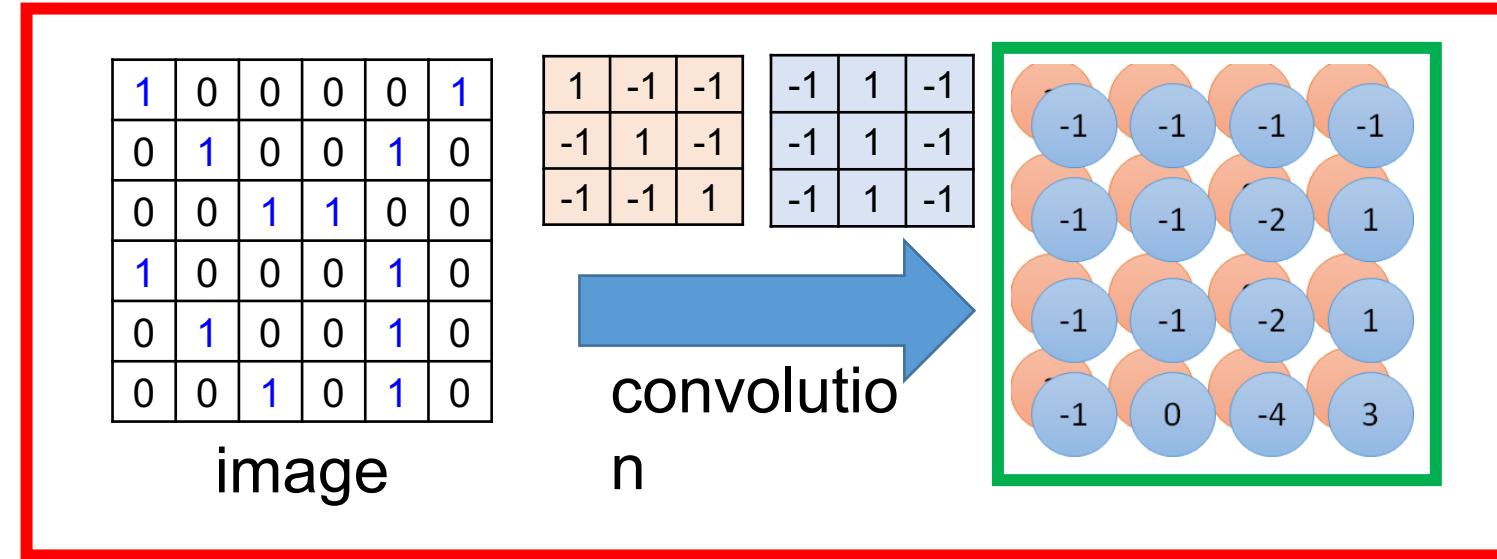
# Reminder: Fully Connected Layer

- 32x32x3 image -> stretch to 3072 x 1
- **Each neuron looks at the full input volume**



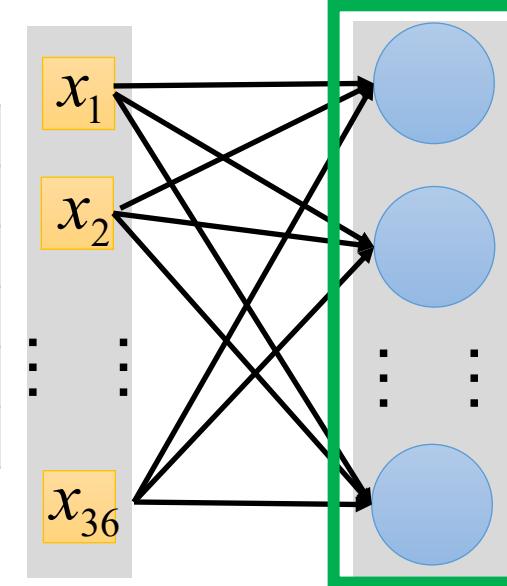
**1 number:**  
the result of taking a dot product  
between a row of  $W$  and the input  
(a 3072-dimensional dot product)

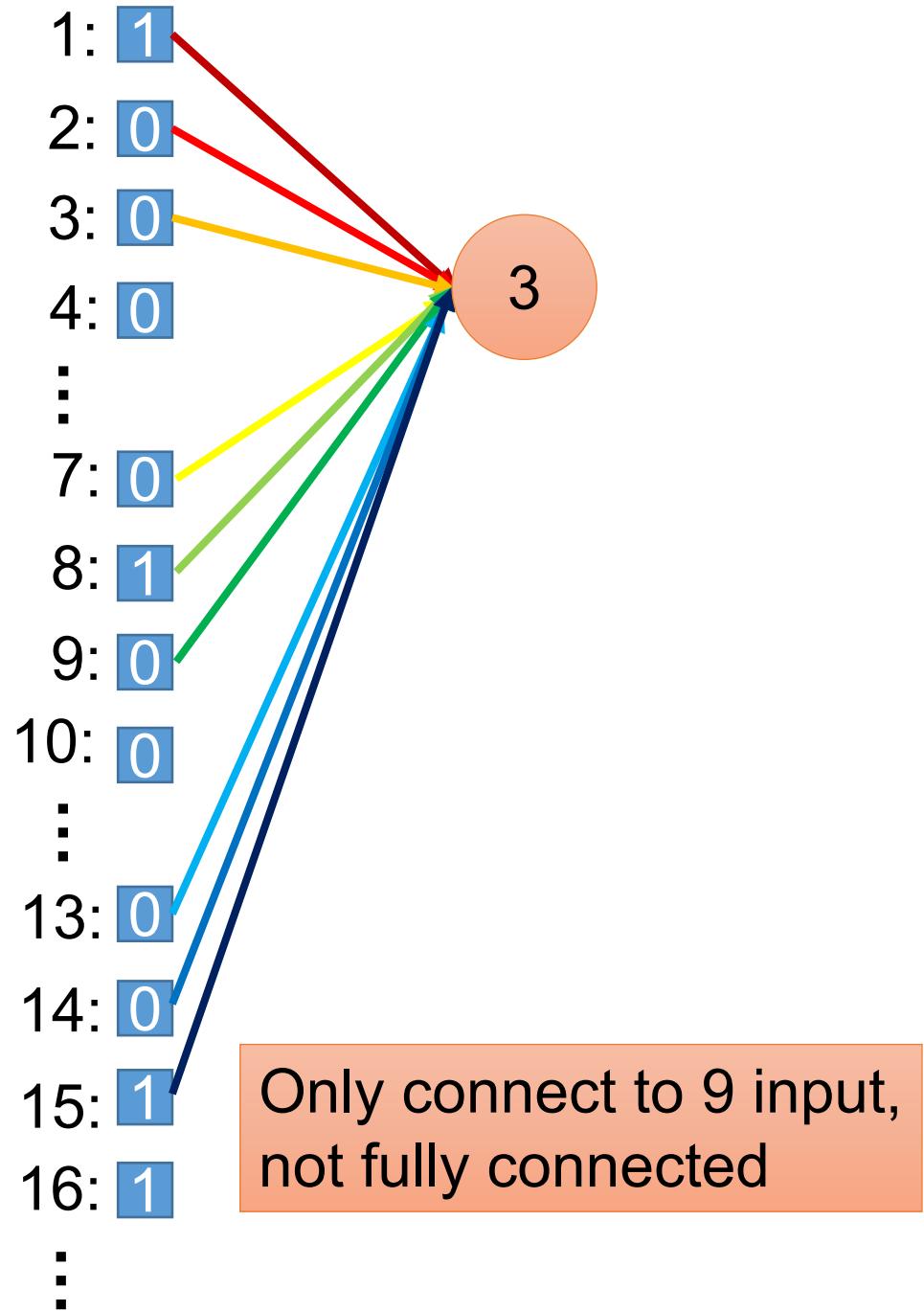
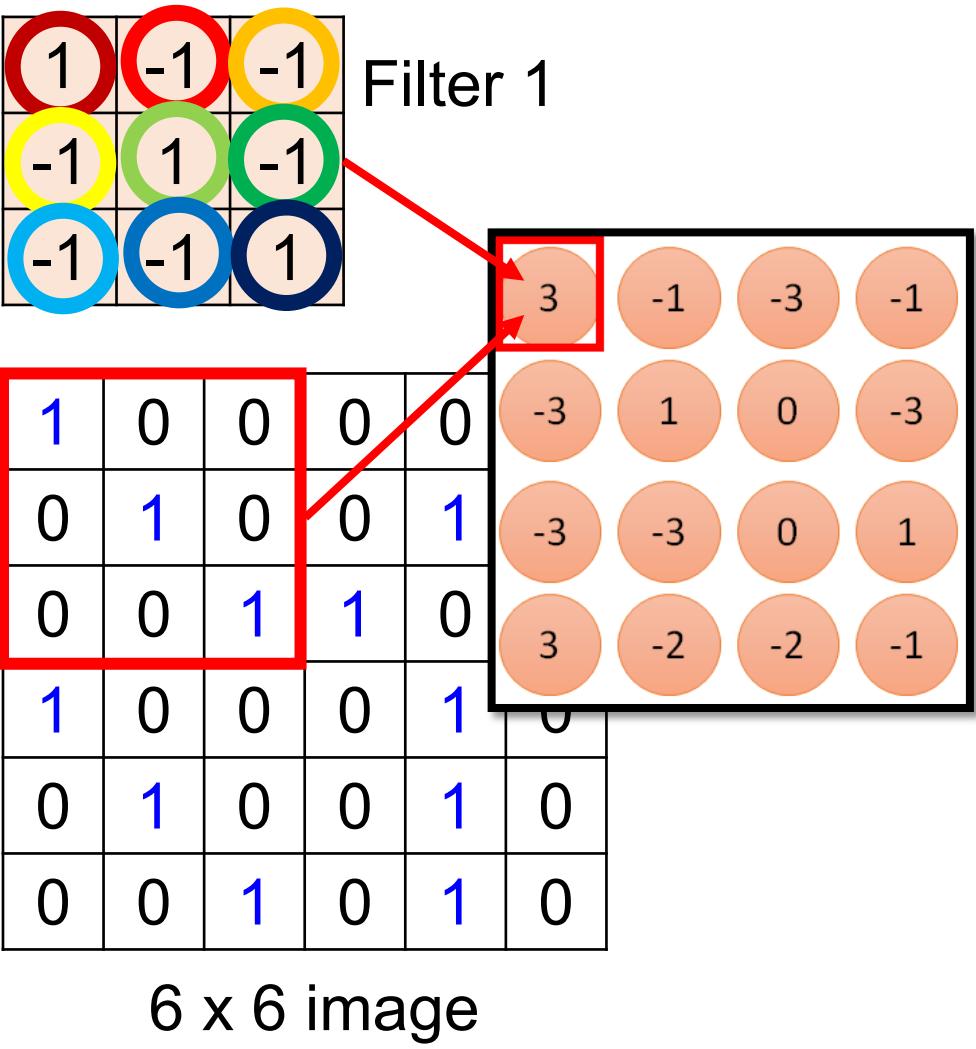
# Convolution v.s. Fully Connected

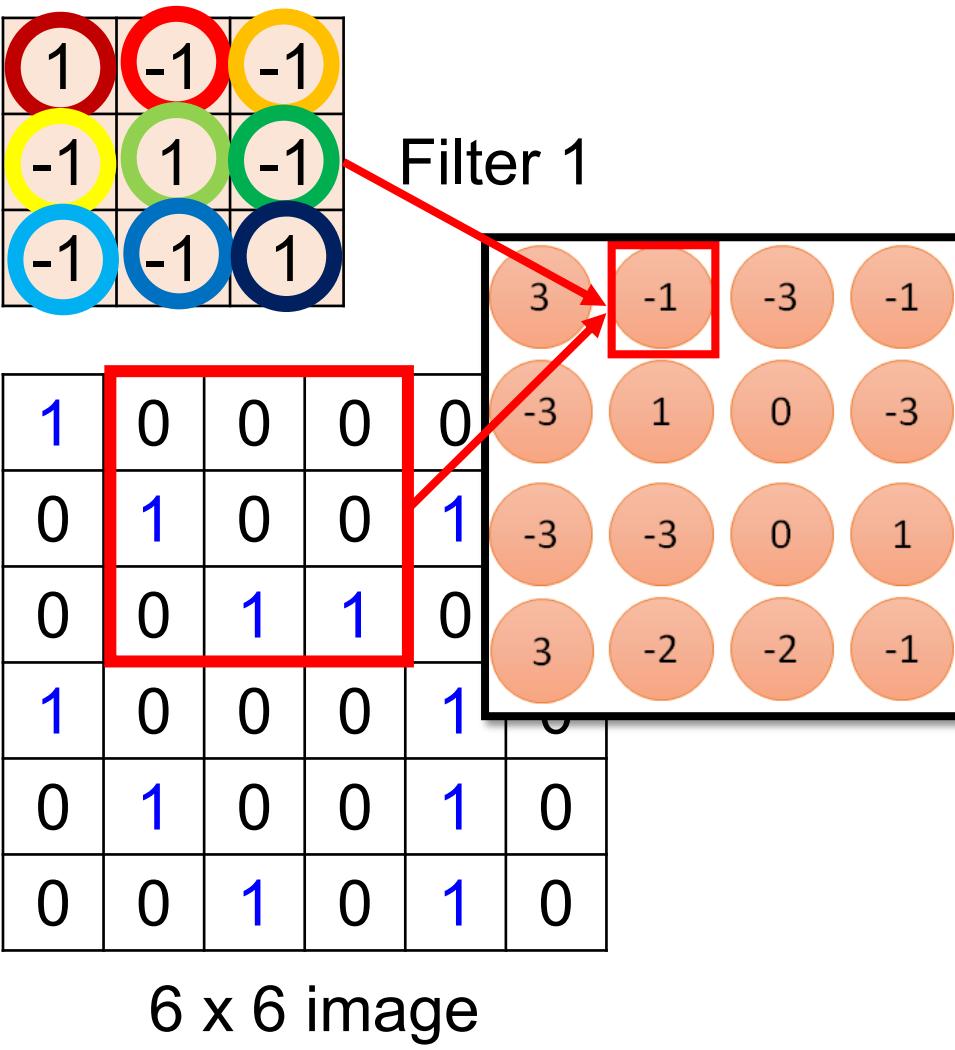


Fully-connected

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

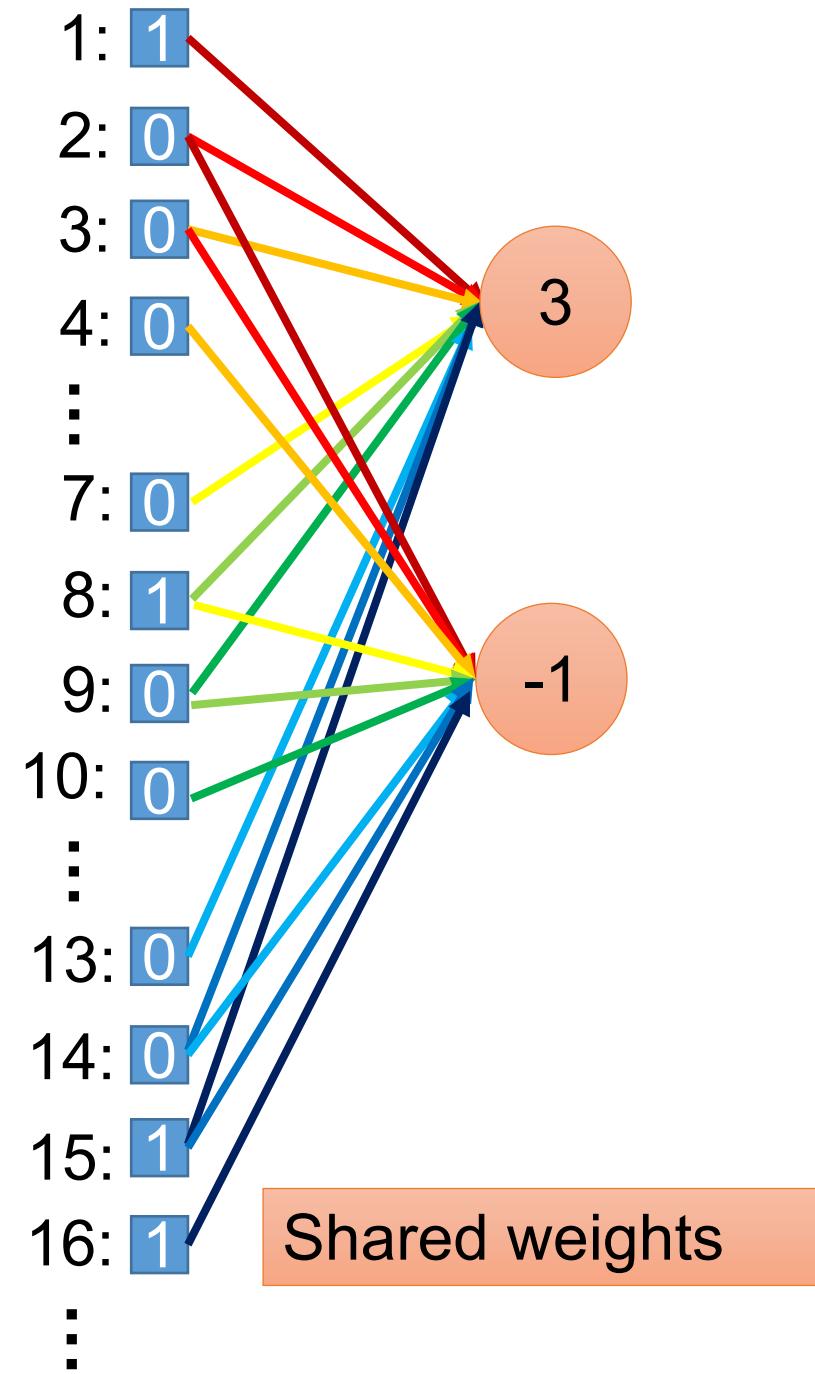




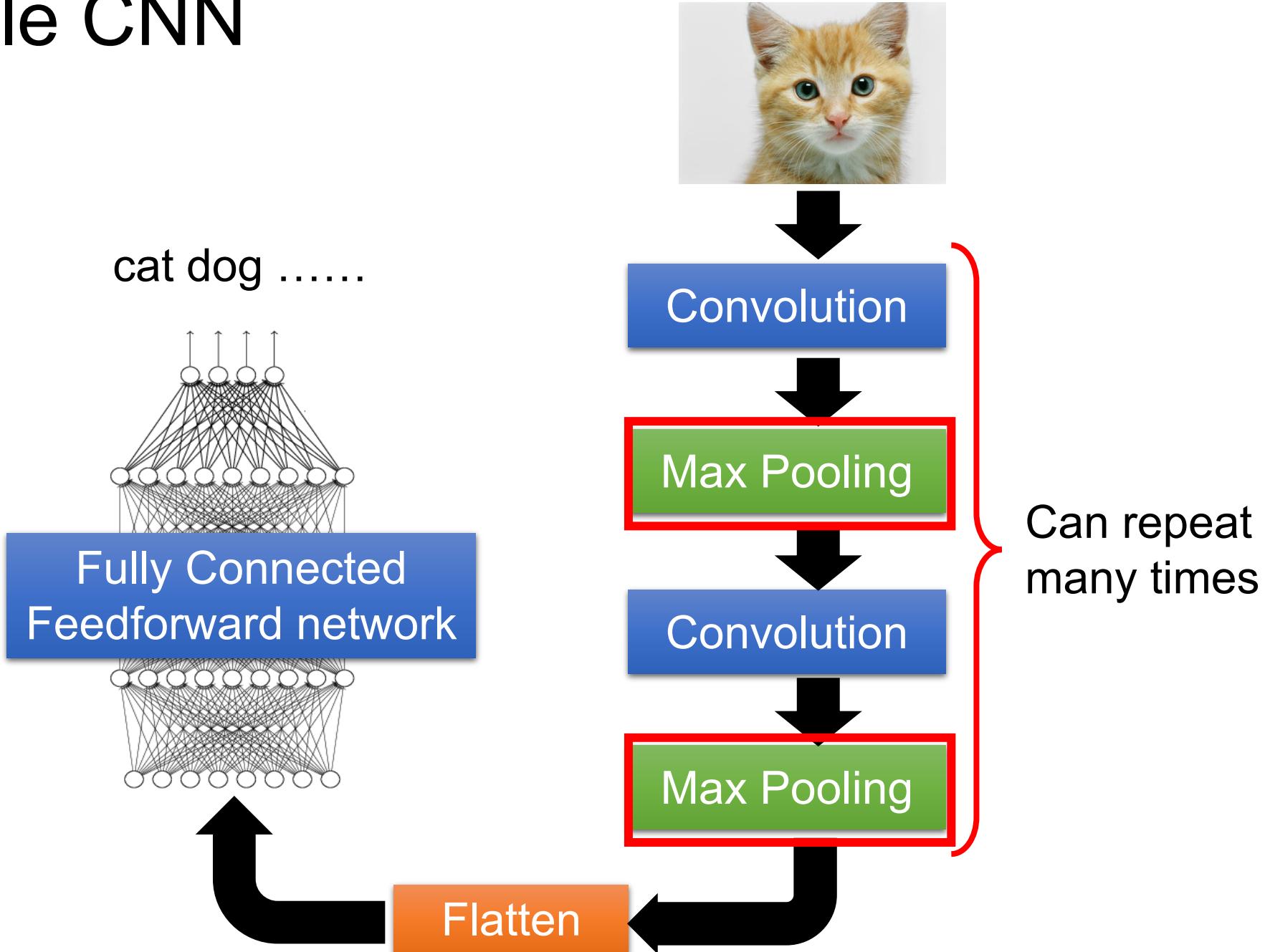


Less parameters!

Even less parameters!



# The whole CNN



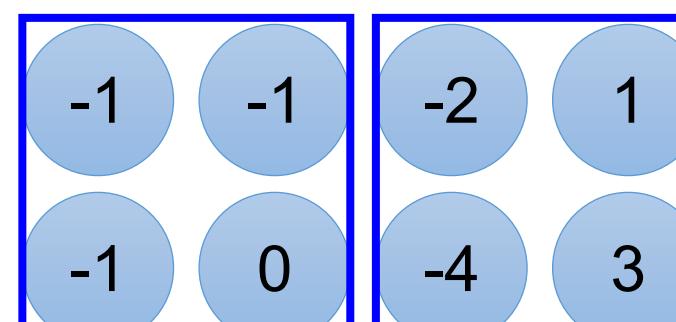
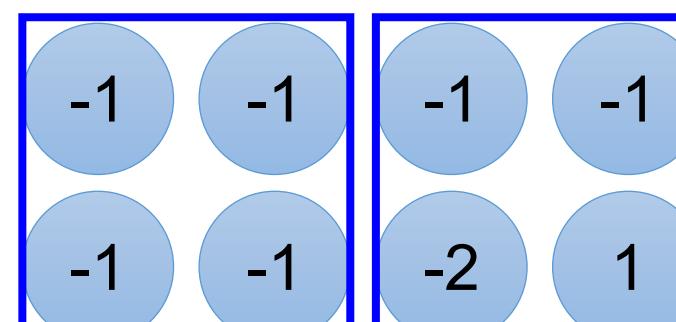
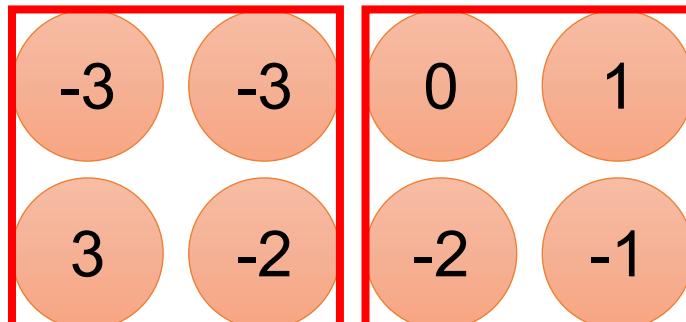
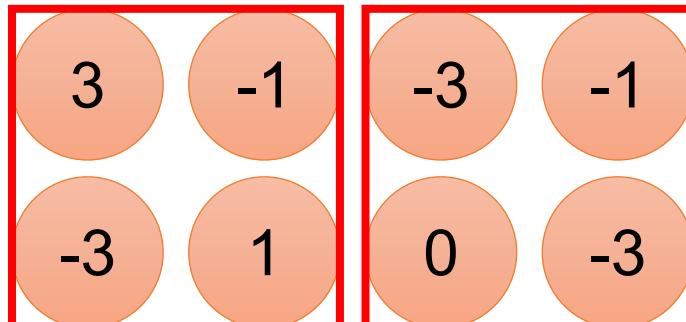
# CNN – Max Pooling

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

Filter 1

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

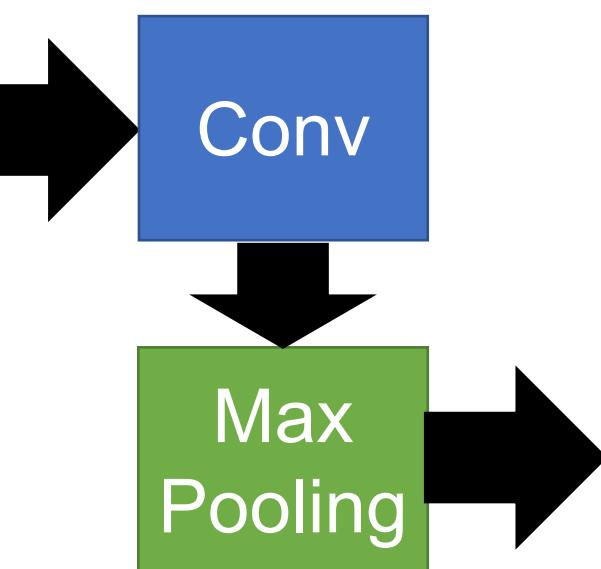
Filter 2



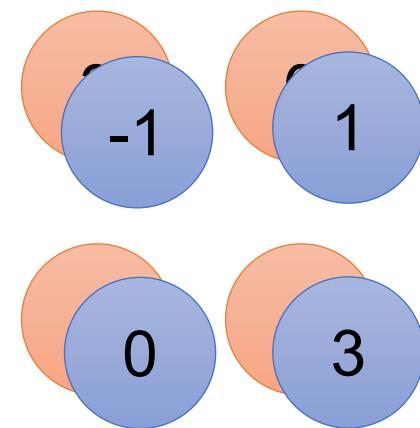
# CNN – Max Pooling

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



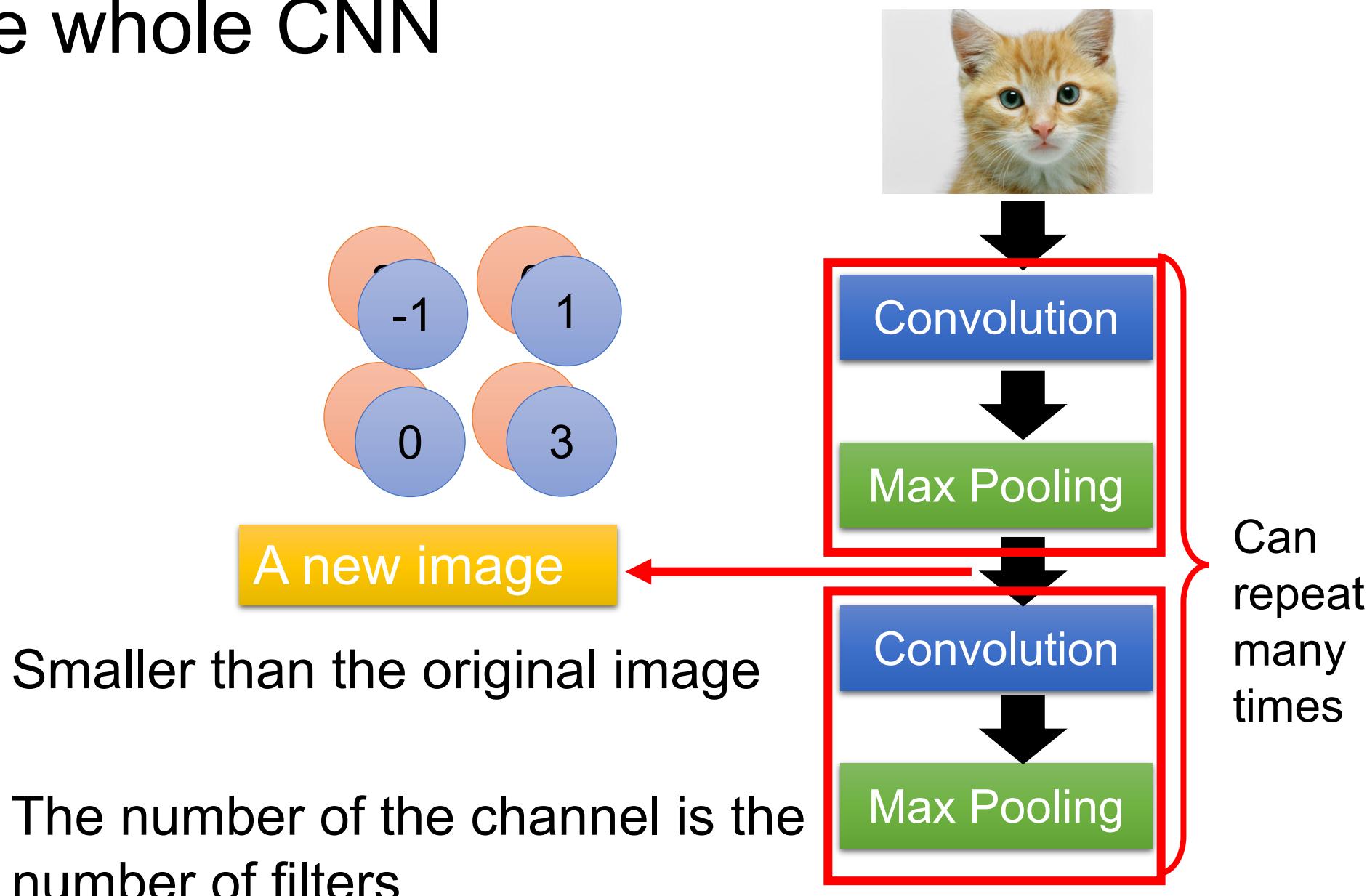
New image  
but smaller



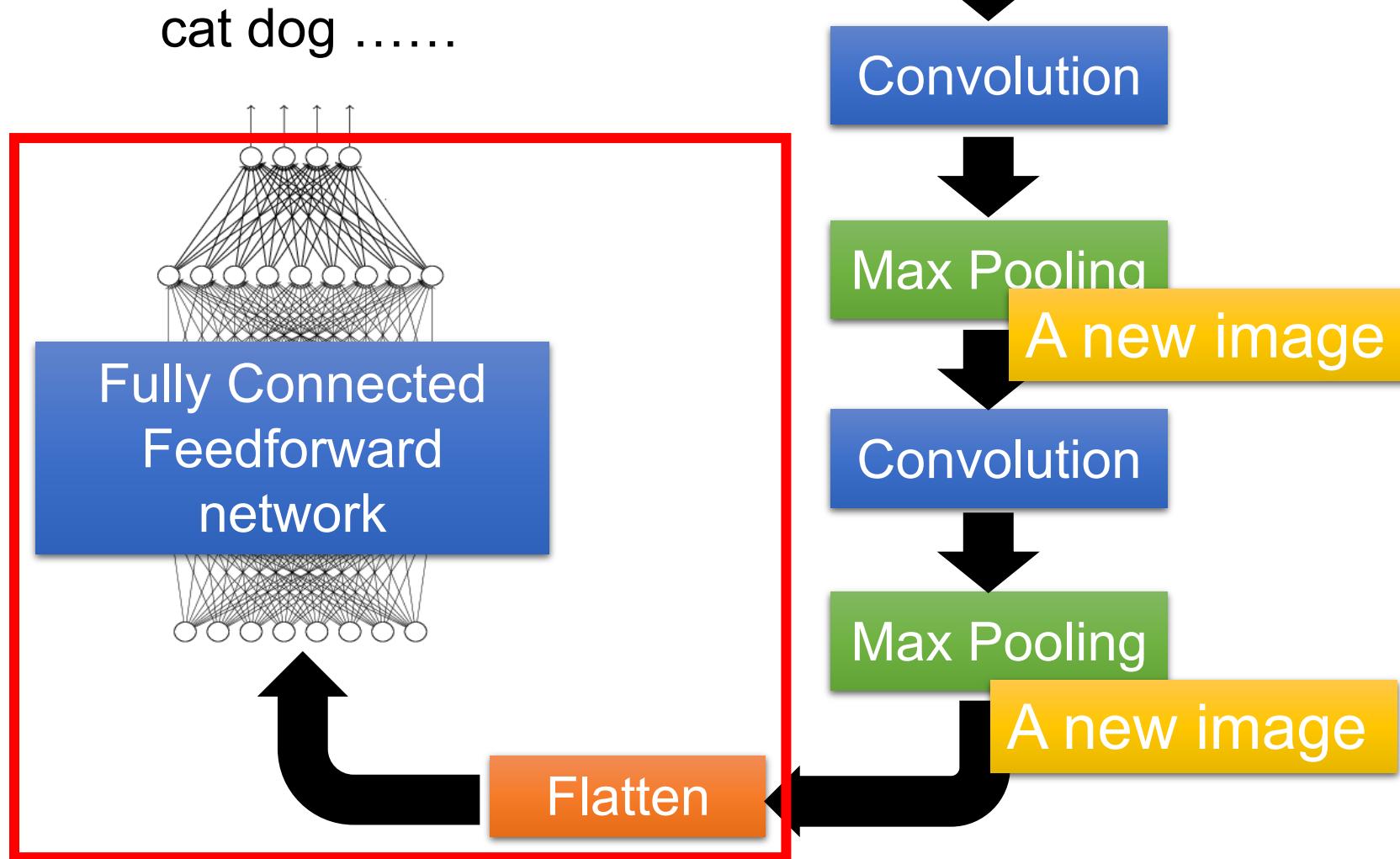
2 x 2 image

Each filter  
is a channel

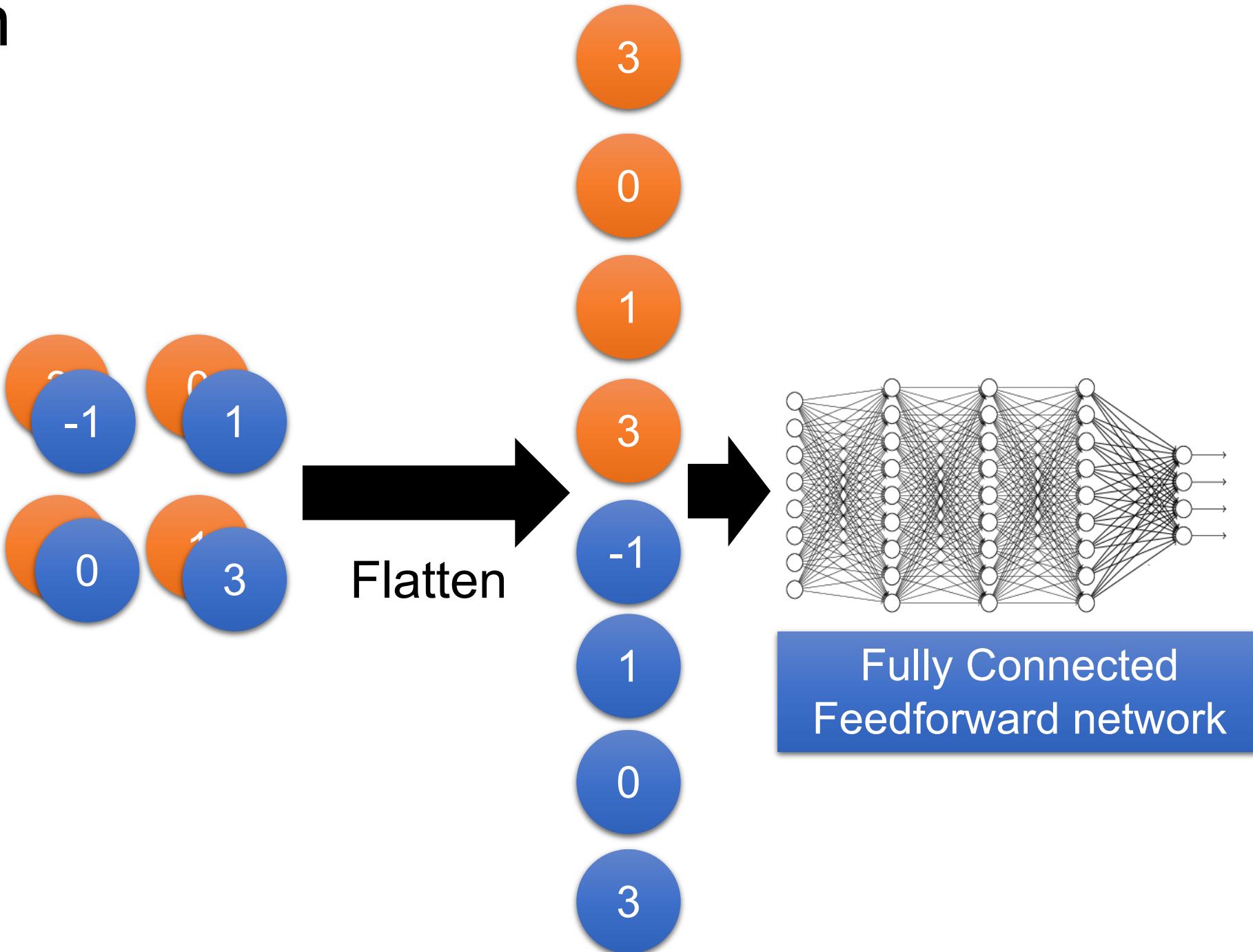
# The whole CNN



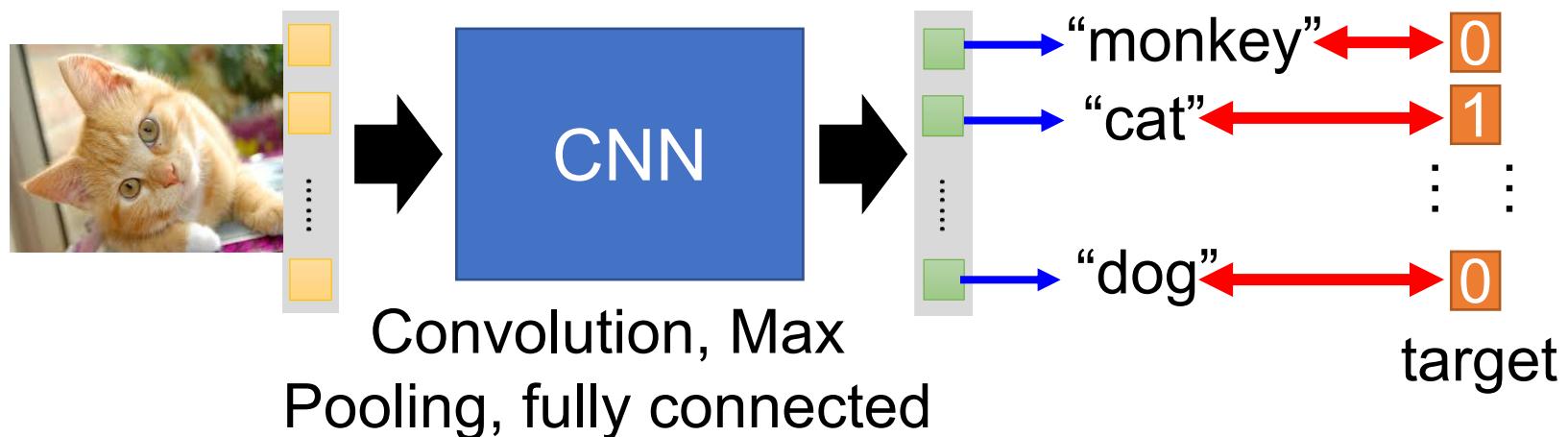
# The whole CNN



# Flatten



# Convolutional Neural Network

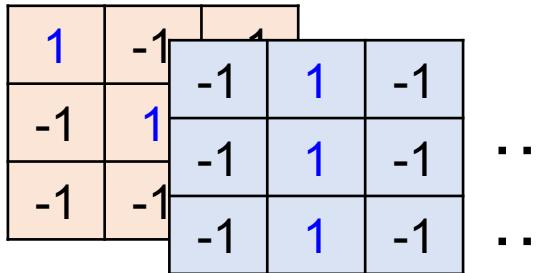


Learning: Nothing special, just gradient descent .....

## 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) ) )
```

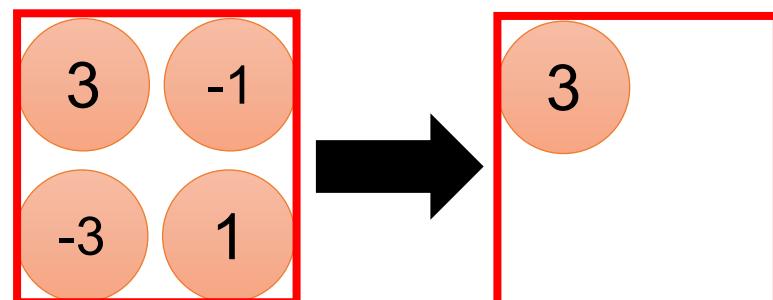


There are 25  
3x3 filters.

Input\_shape = ( 1 , 28 , 28 )

1: black/weight, 3: RGB      28 x 28 pixels

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



input

Convolution

Max Pooling

Convolution

Max Pooling

## CNN in Keras

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

How many parameters  
for each filter?

9

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

25 x 26 x 26

How many parameters  
for each filter?

225

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

50 x 11 x 11

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

50 x 5 x 5

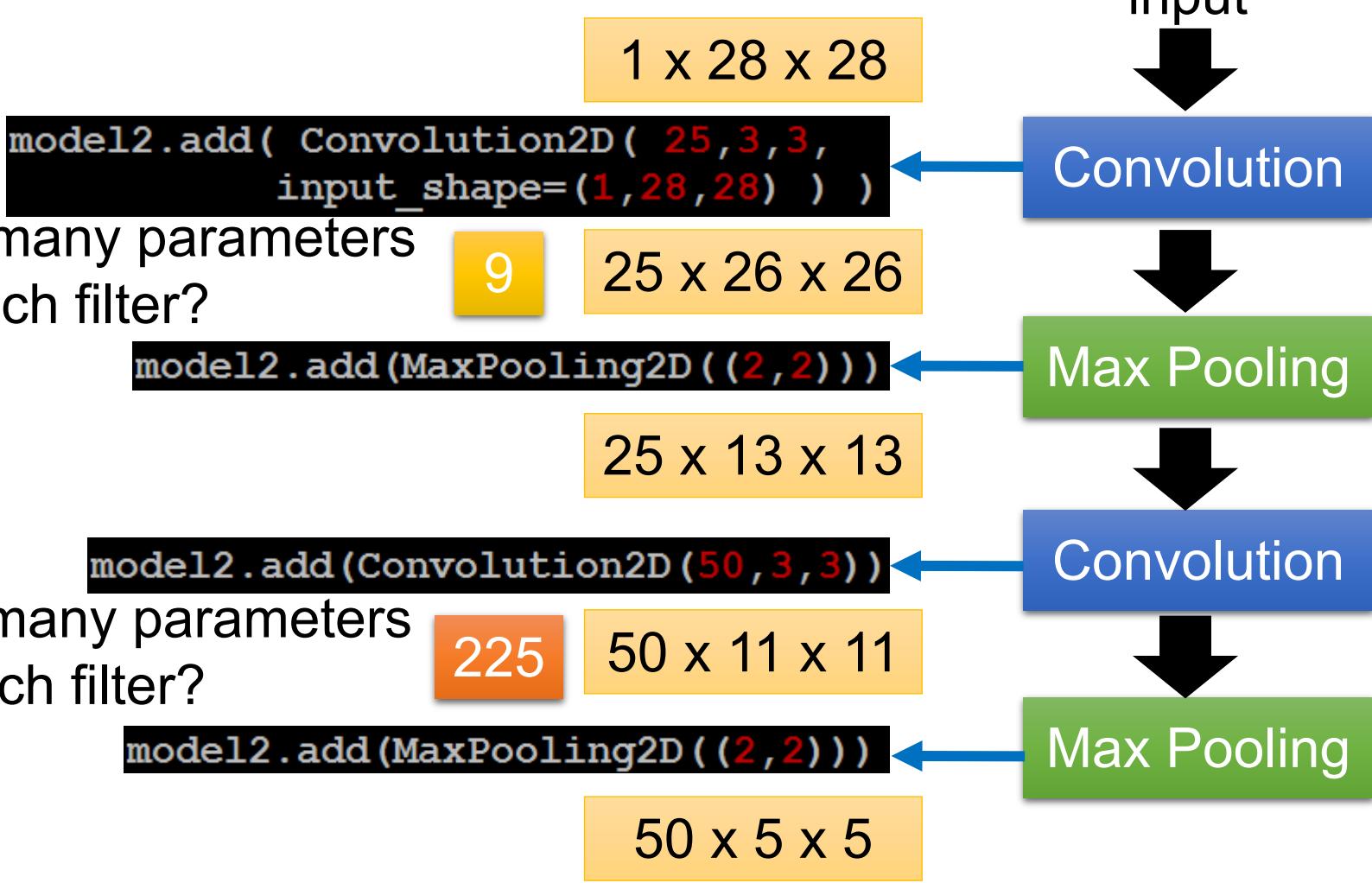
input

Convolution

Max Pooling

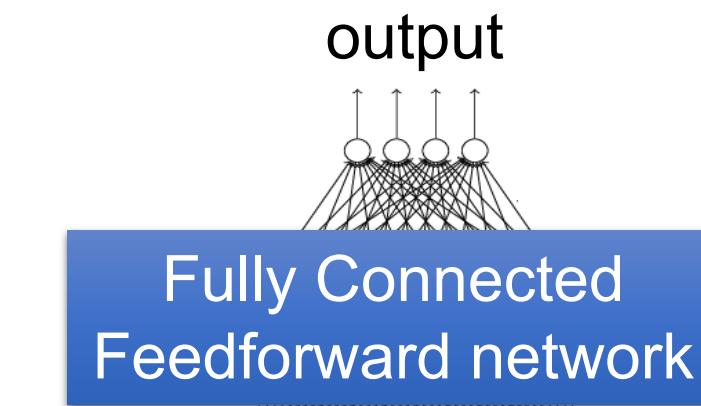
Convolution

Max Pooling

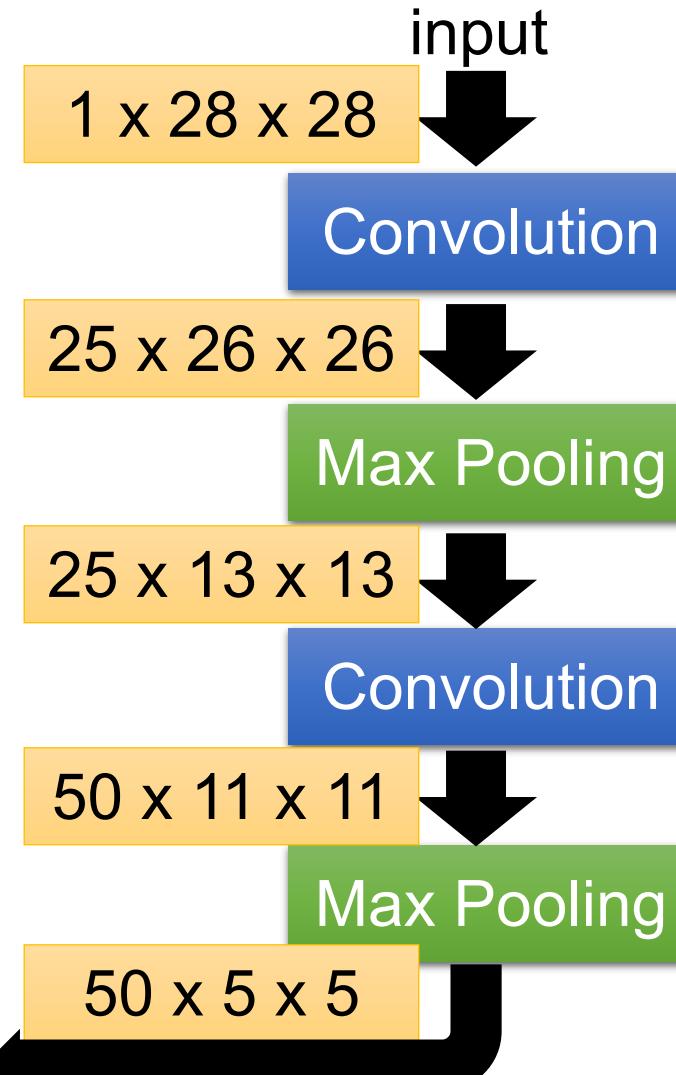
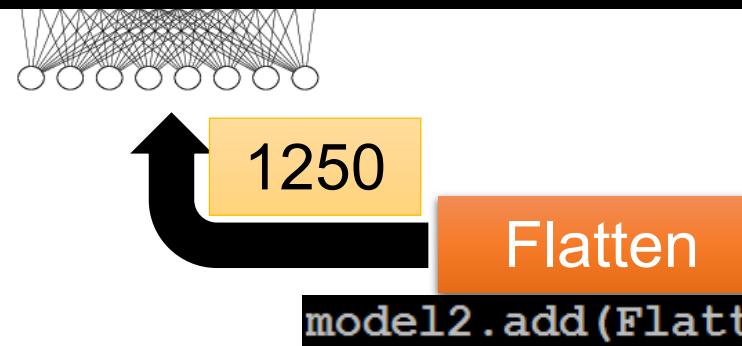


# CNN in Keras

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

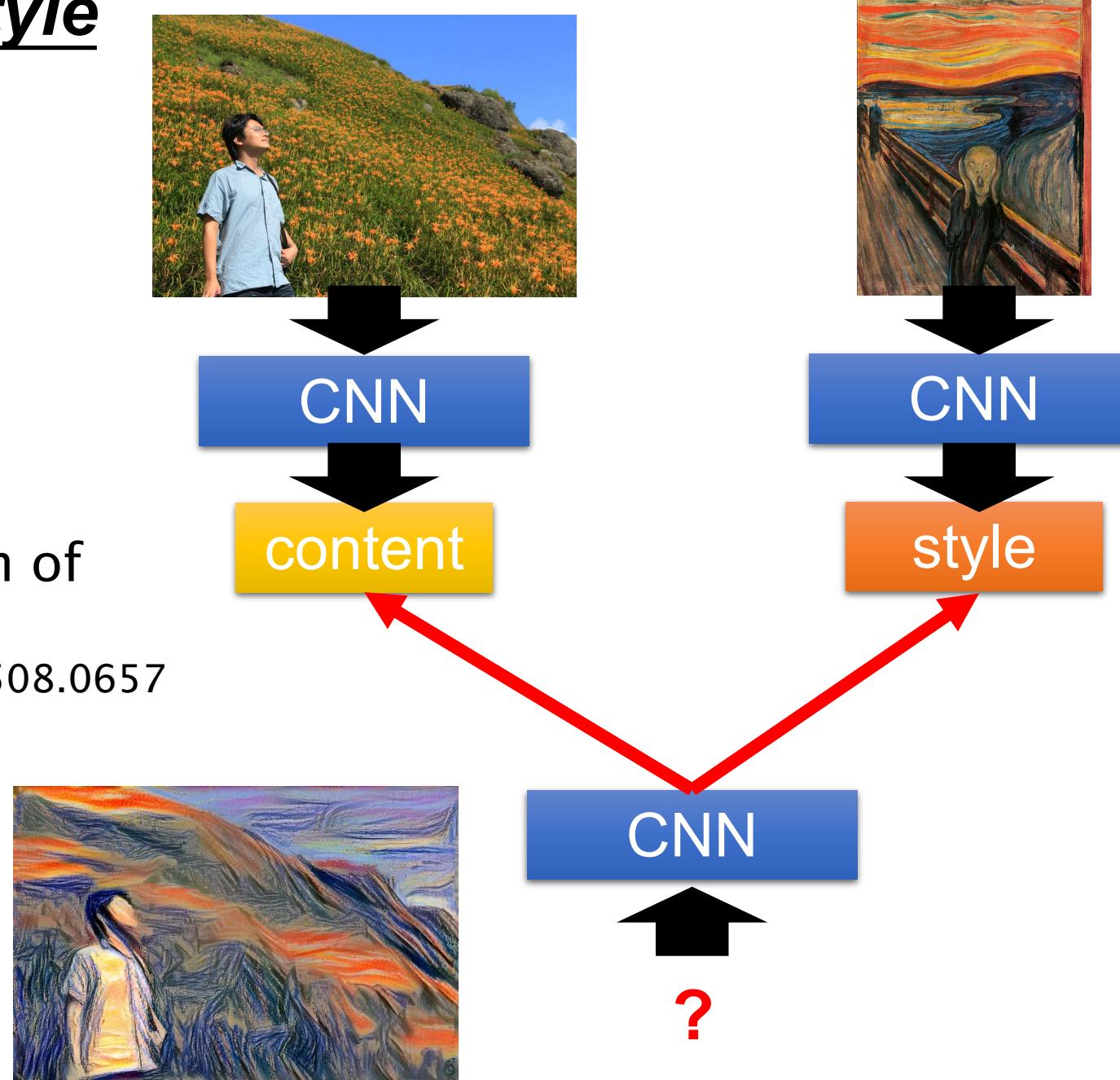


```
model2.add(Dense(output_dim=100))  
model2.add(Activation('relu'))  
model2.add(Dense(output_dim=10))  
model2.add(Activation('softmax'))
```

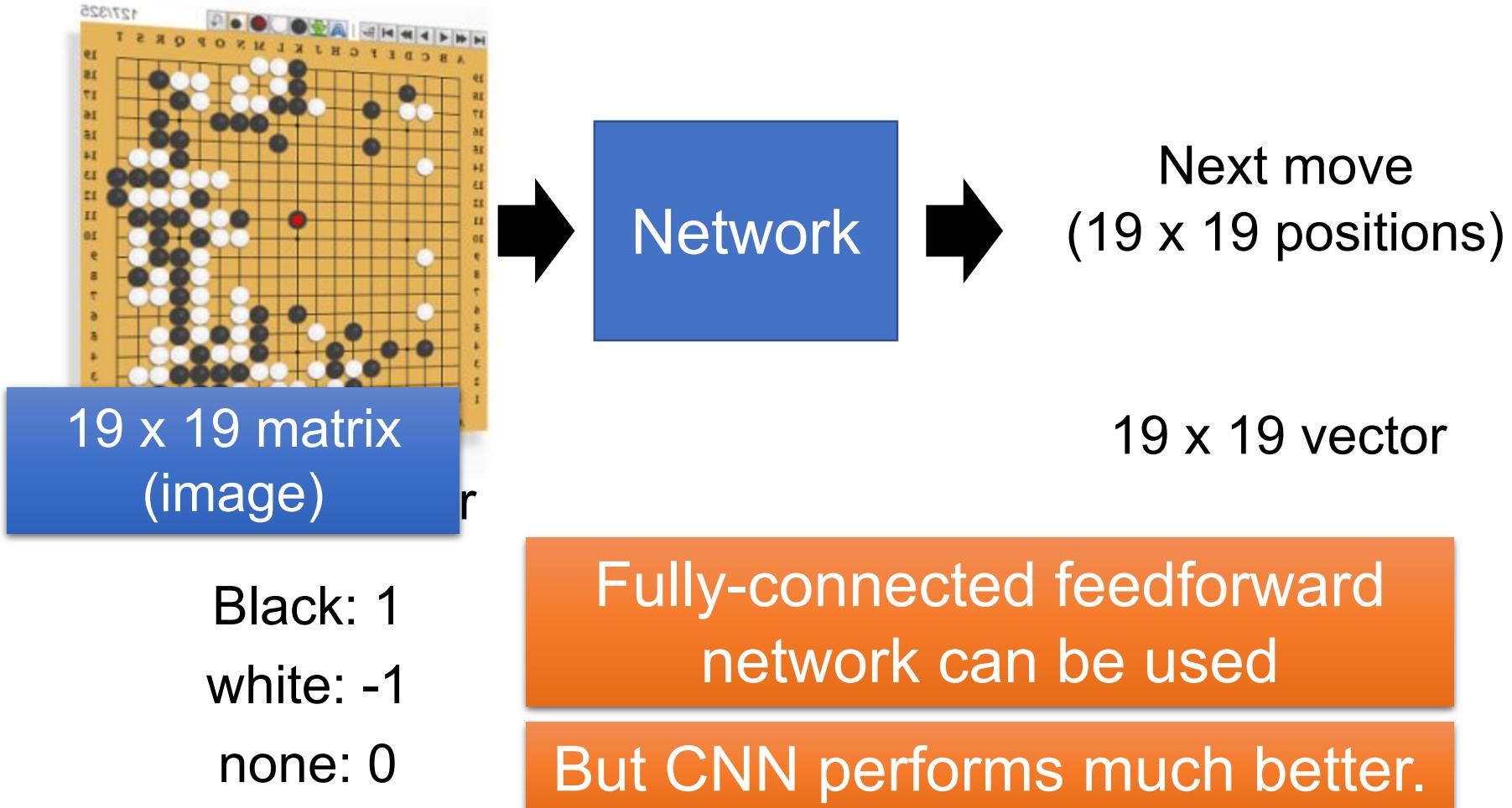


# Deep Style

A Neural Algorithm of  
Artistic Style  
<https://arxiv.org/abs/1508.06576>

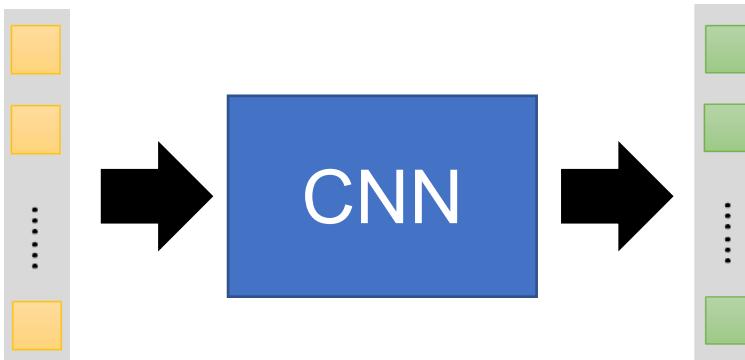


# More Application: Playing Go



# More Application: Playing Go

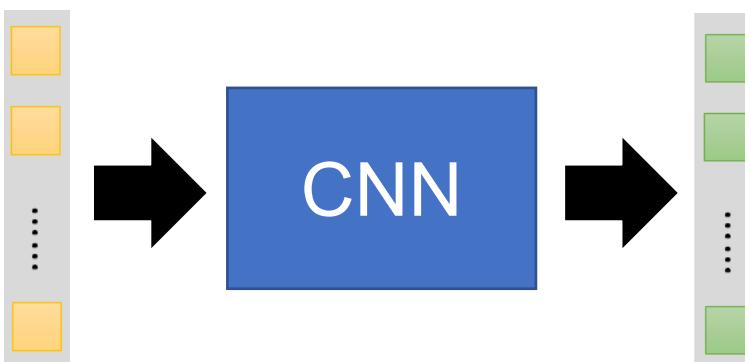
Training: record of previous plays  
黑: 5之五 → 白: 天元 → 黑: 五之五 ...



CNN



Target:  
“天元” = 1  
else = 0



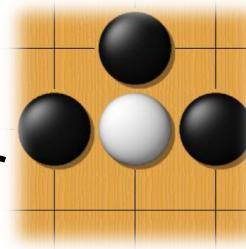
CNN



Target:  
“五之5” = 1  
else = 0

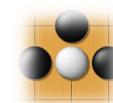
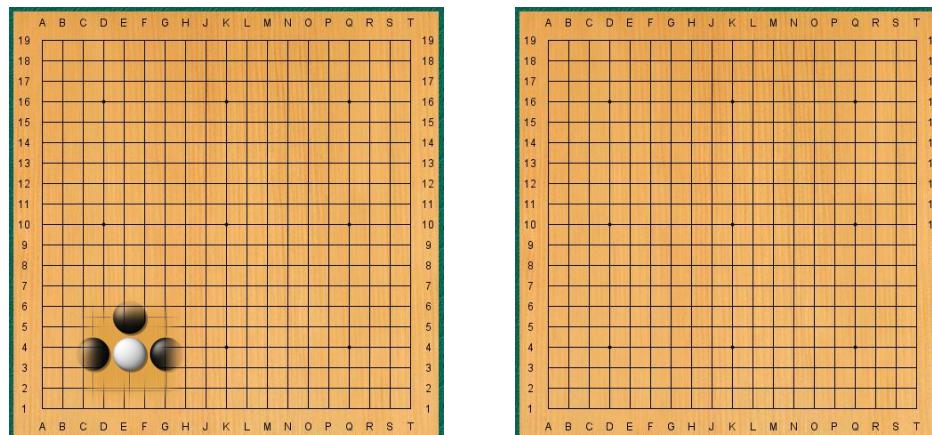
# Why CNN for playing Go?

- Some patterns are much smaller than the whole image



Alpha Go uses  $5 \times 5$  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  $23 \times 23$  image, then convolves  $k$  filters of kernel size  $5 \times 5$  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  $21 \times 21$  image, then convolves  $k$  filters of kernel size  $3 \times 3$  with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size  $1 \times 1$  with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used  $k = 192$  filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with  $k = 128, 256$  and  $384$  filters.

Alpha Go does not use Max Pooling .....

# Summary

- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like

**[(CONV-RELU)\*N-POOL?] \* M - (FC-RELU)\*K, SOFTMAX**

where N is usually up to ~5, M is large,  $0 \leq K \leq 2$ .

- but recent advances such as ResNet/GoogLeNet challenge this paradigm

# Thanks