

# Convolutional Neural Network

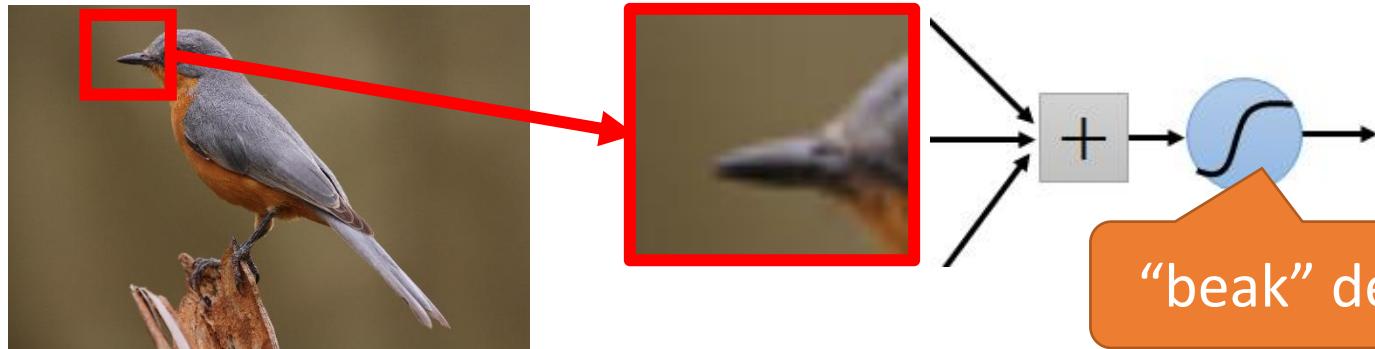
Can the network be simplified by  
considering the properties of images?

# 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.

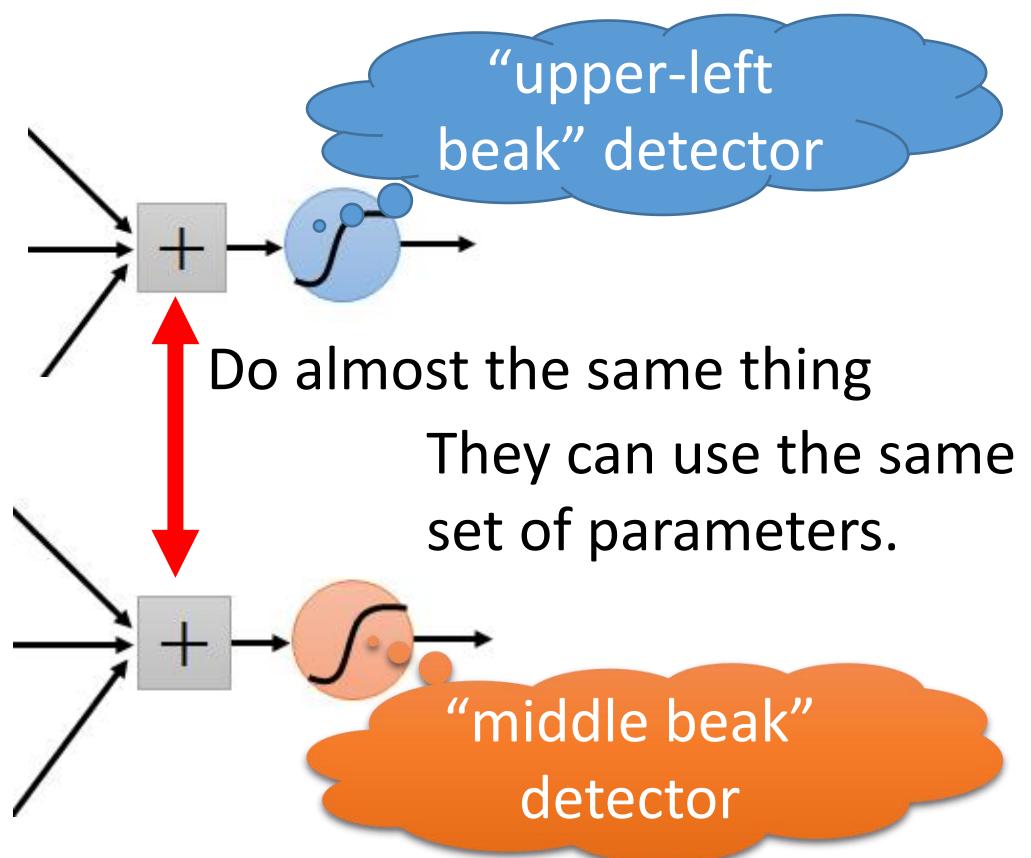
Connecting to small region with less parameters



“beak” detector

# 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

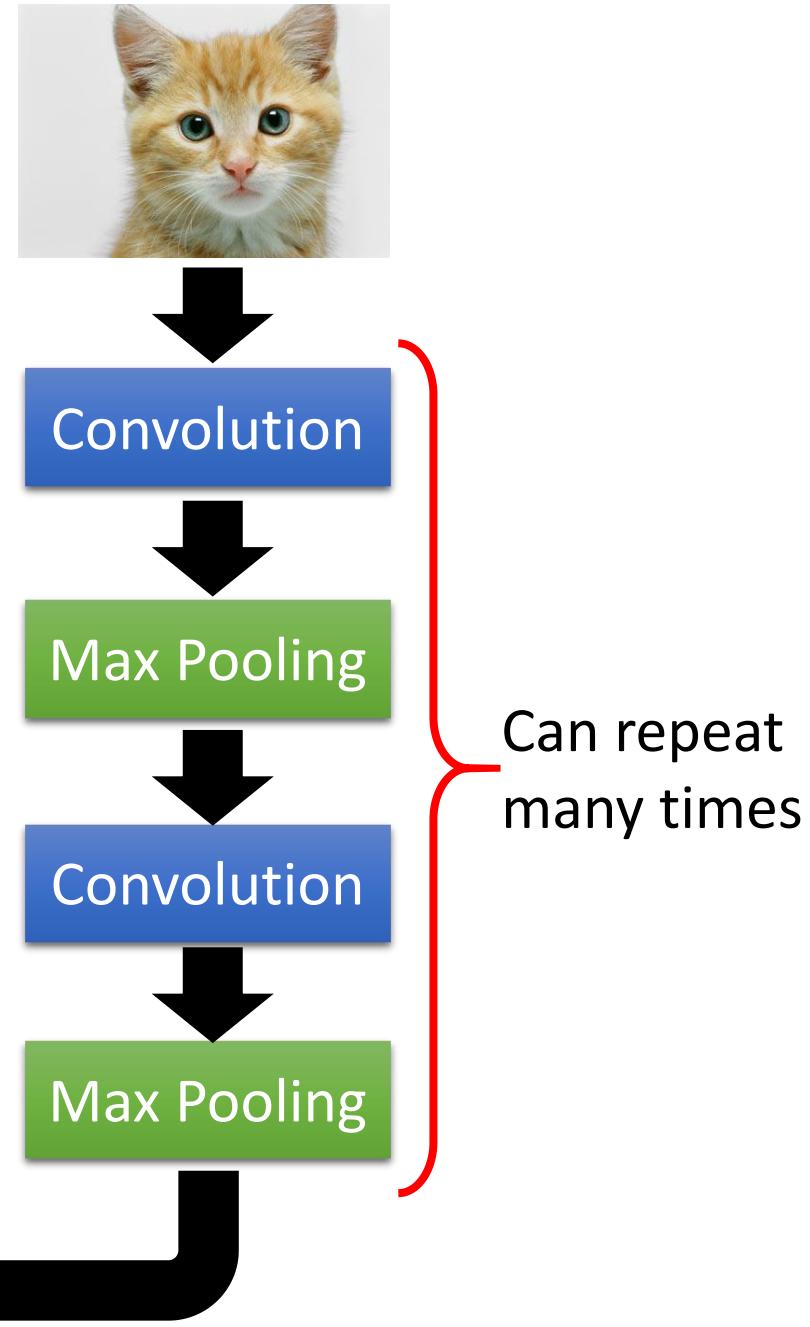
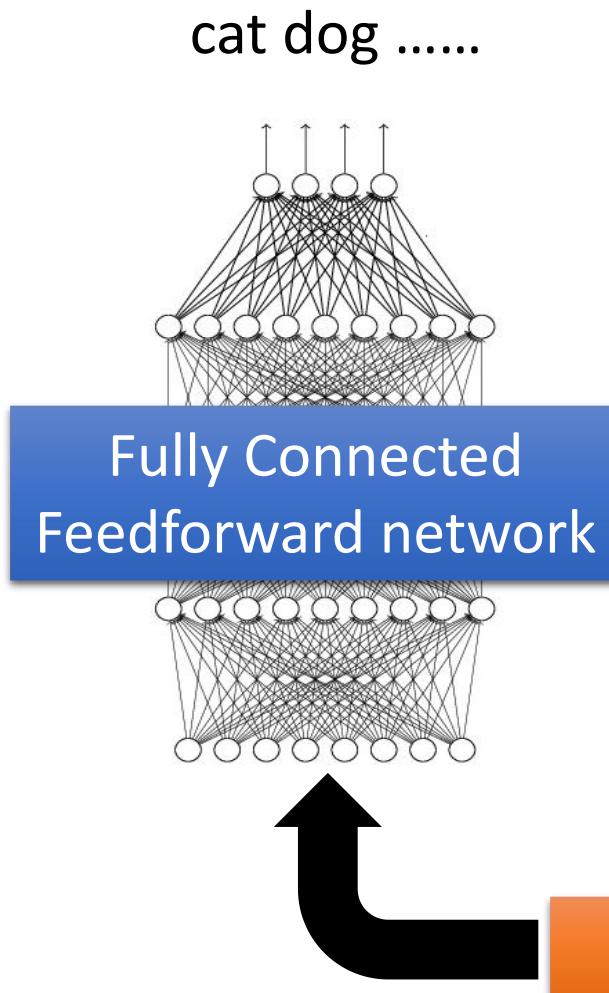


We can subsample the pixels to make image smaller



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

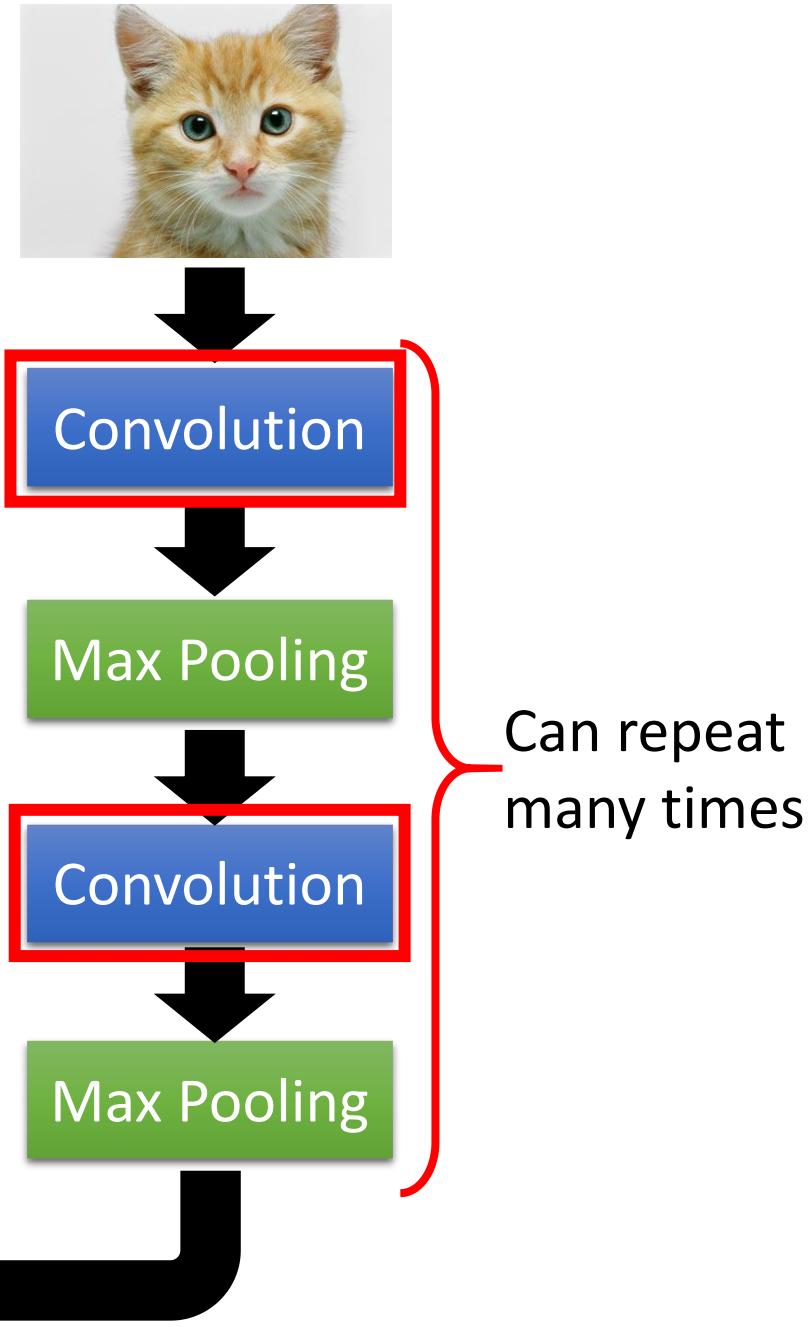
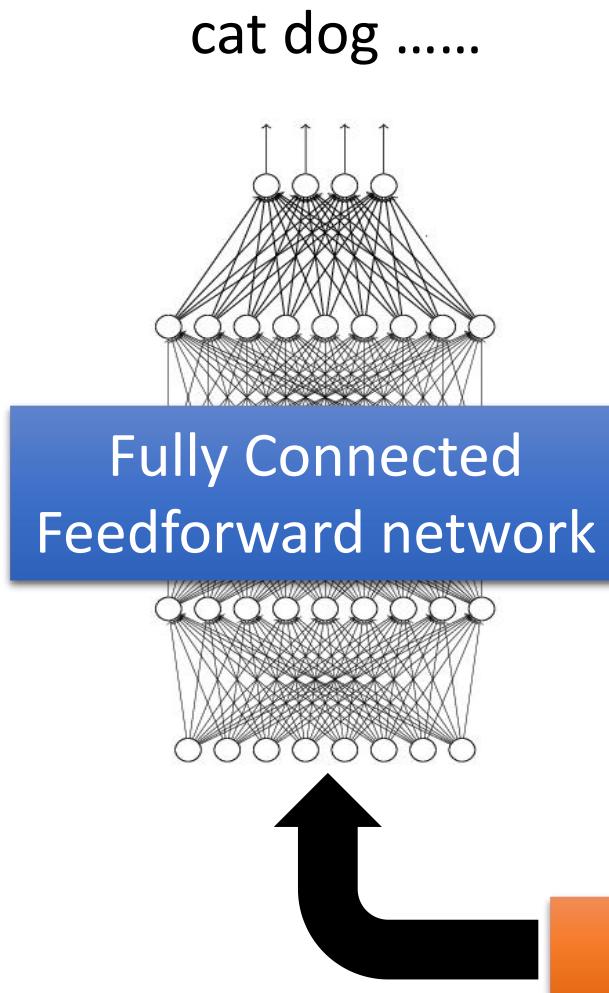
Max Pooling

Convolution

Max Pooling



# 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

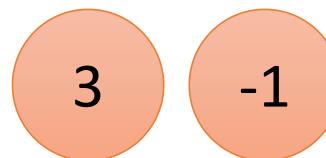
stride=1

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

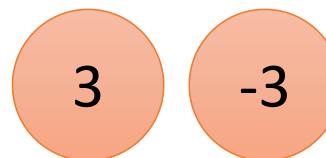
If stride=2

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

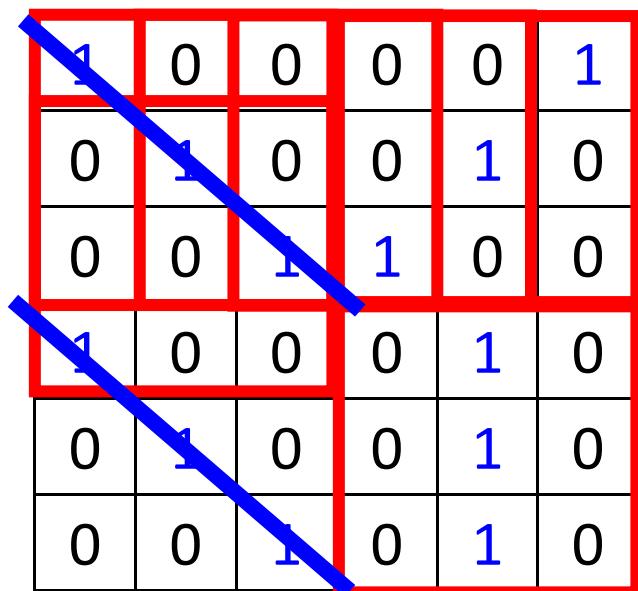
Filter 1



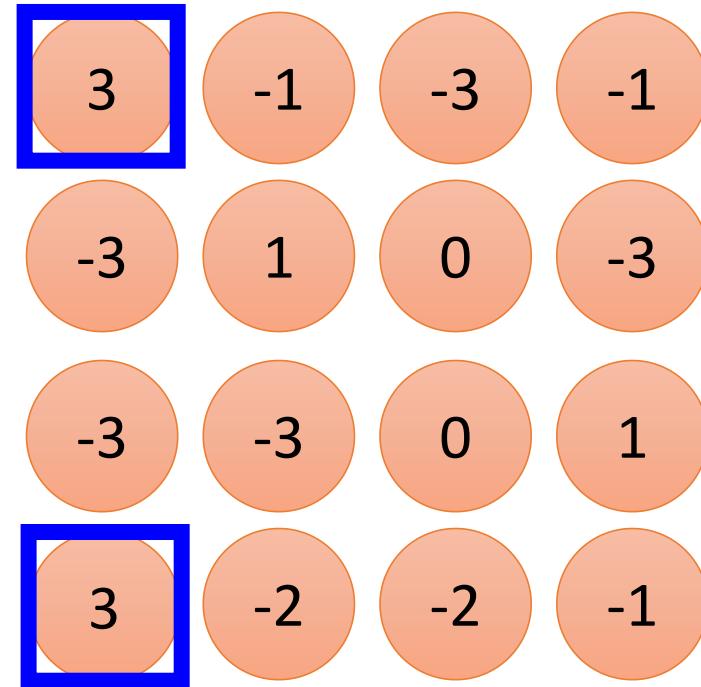
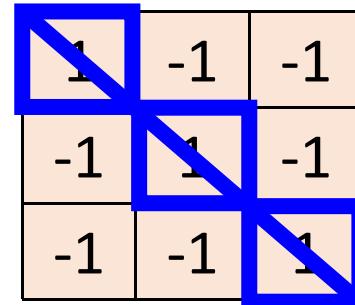
We set stride=1 below

# CNN – Convolution

stride=1



6 x 6 image



# CNN – Convolution

stride=1

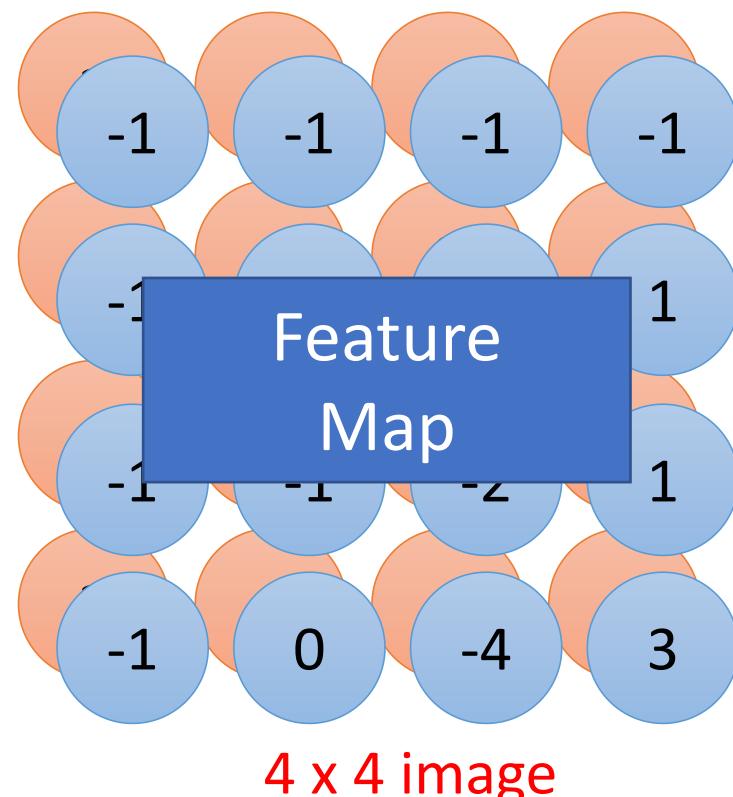
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

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

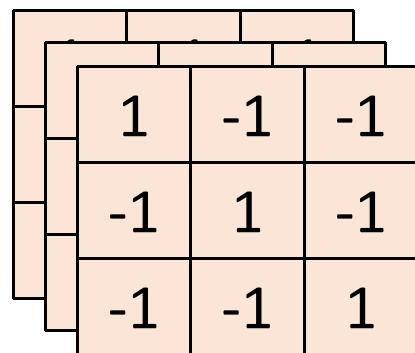
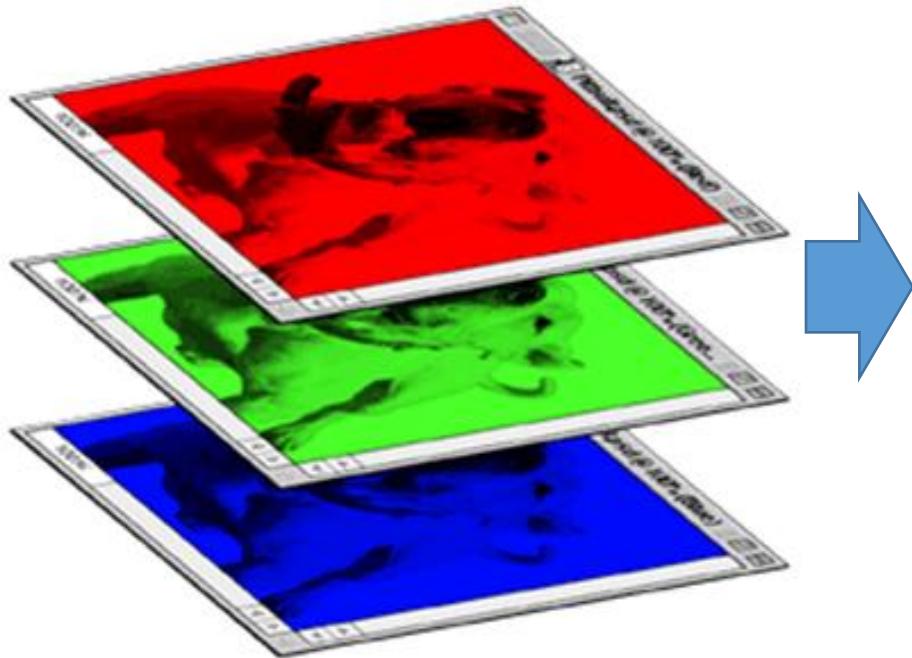
Filter 2

Do the same process for every filter

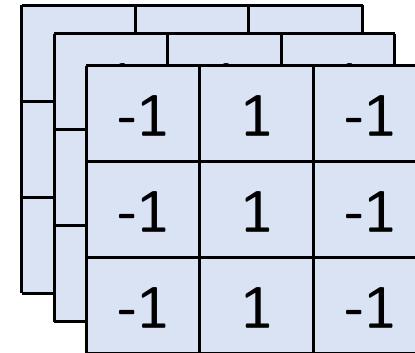


# CNN – Colorful image

# Colorful image



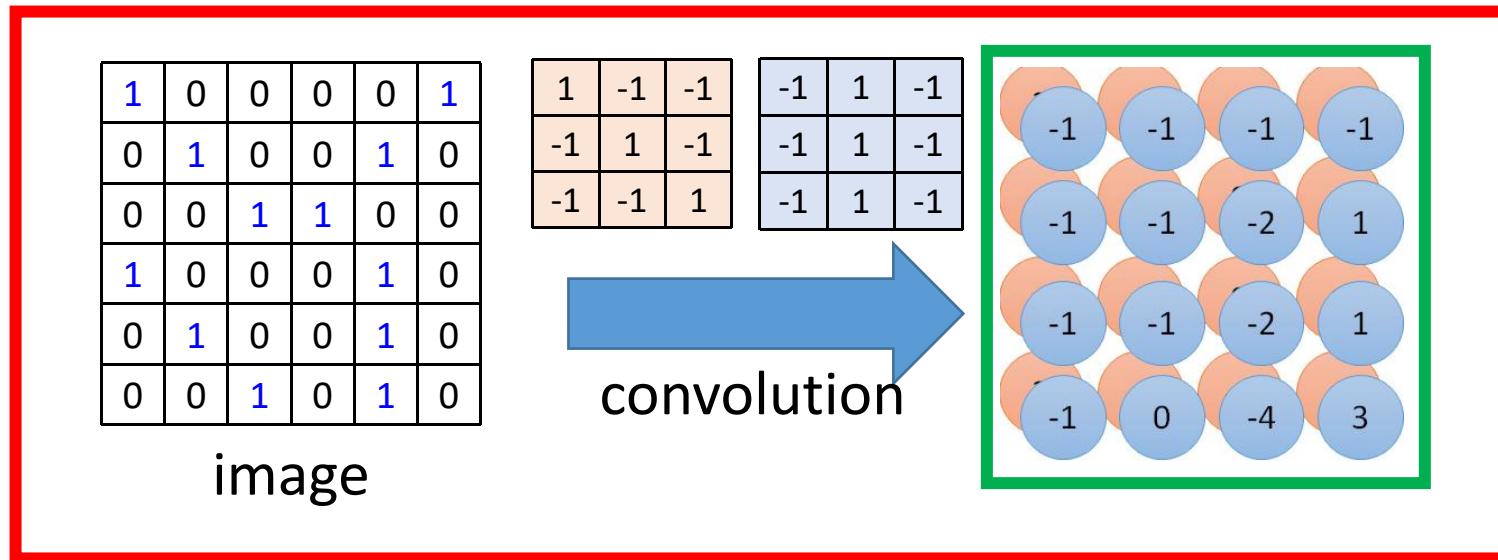
## Filter 1



## Filter 2

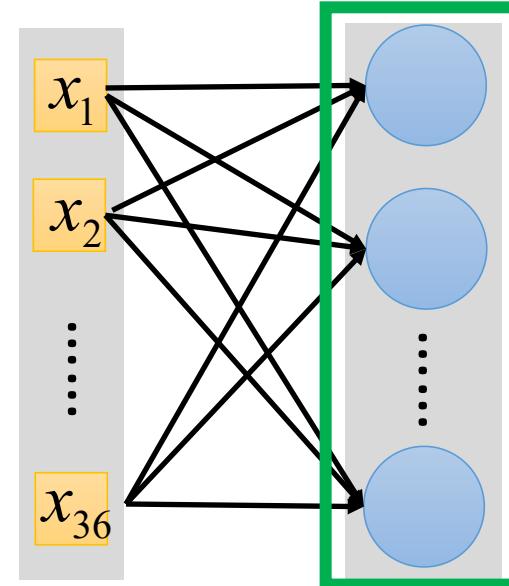
	1						1
1	1	0	0	0	0	1	
	0	1	0	0	1	0	
	0	0	1	1	0	0	
.	1	0	0	0	1	0	
	0	1	0	0	1	0	
	0	0	1	0	1	0	

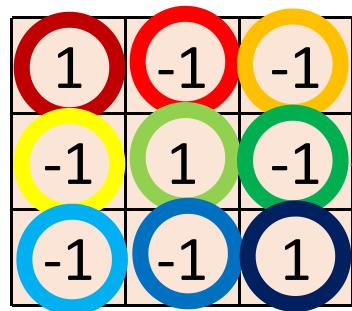
# Convolution v.s. Fully Connected



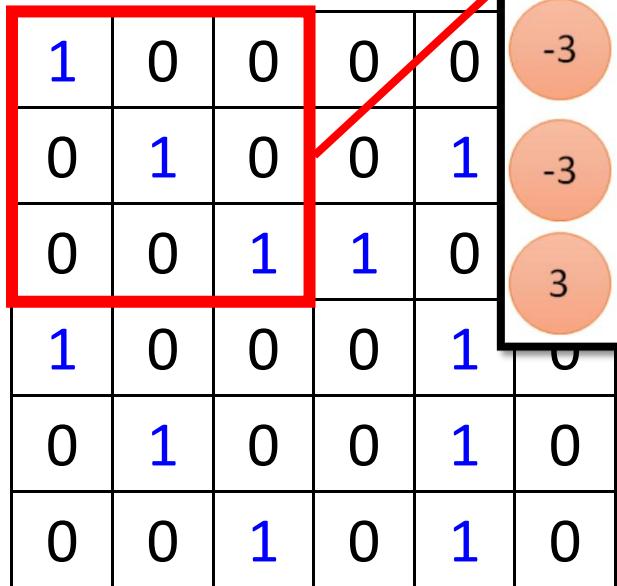
Fully-  
connected

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



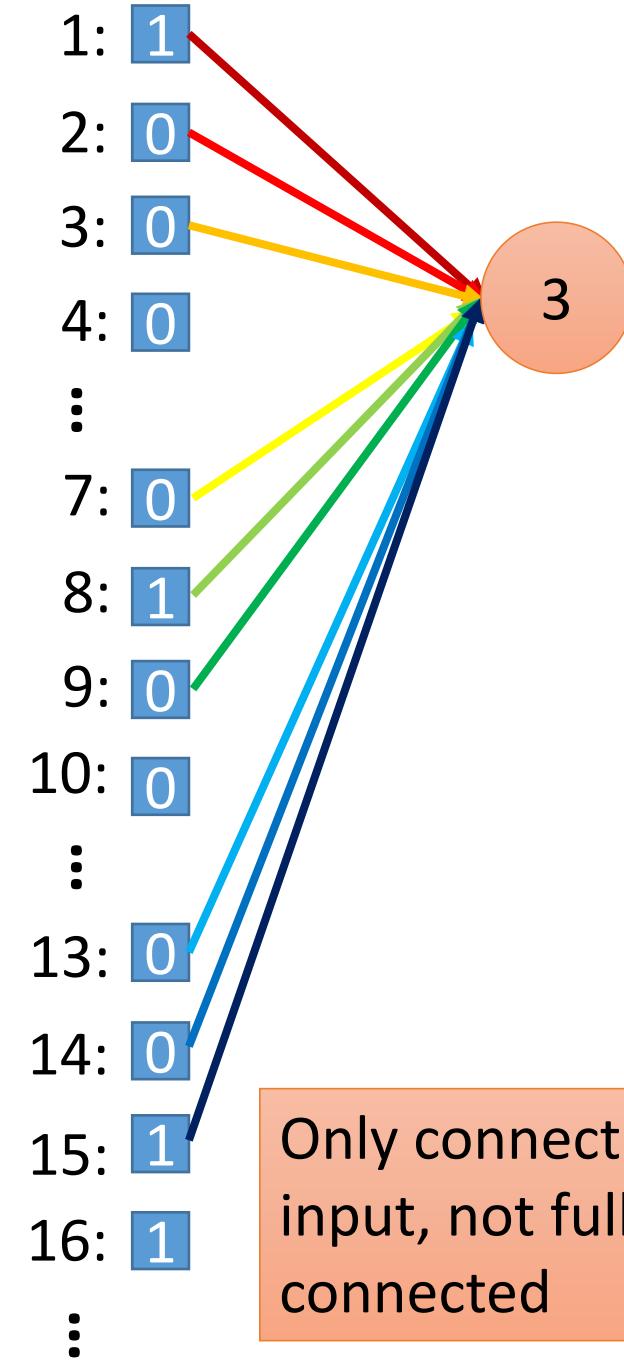
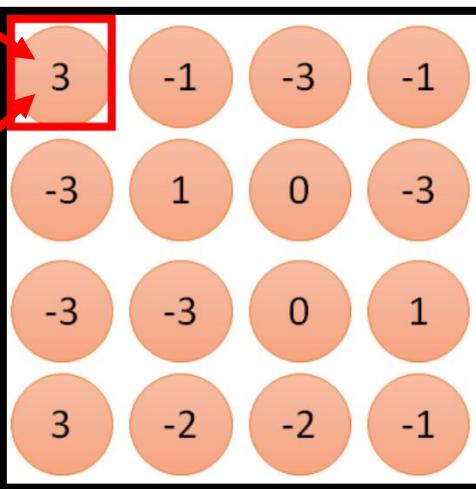


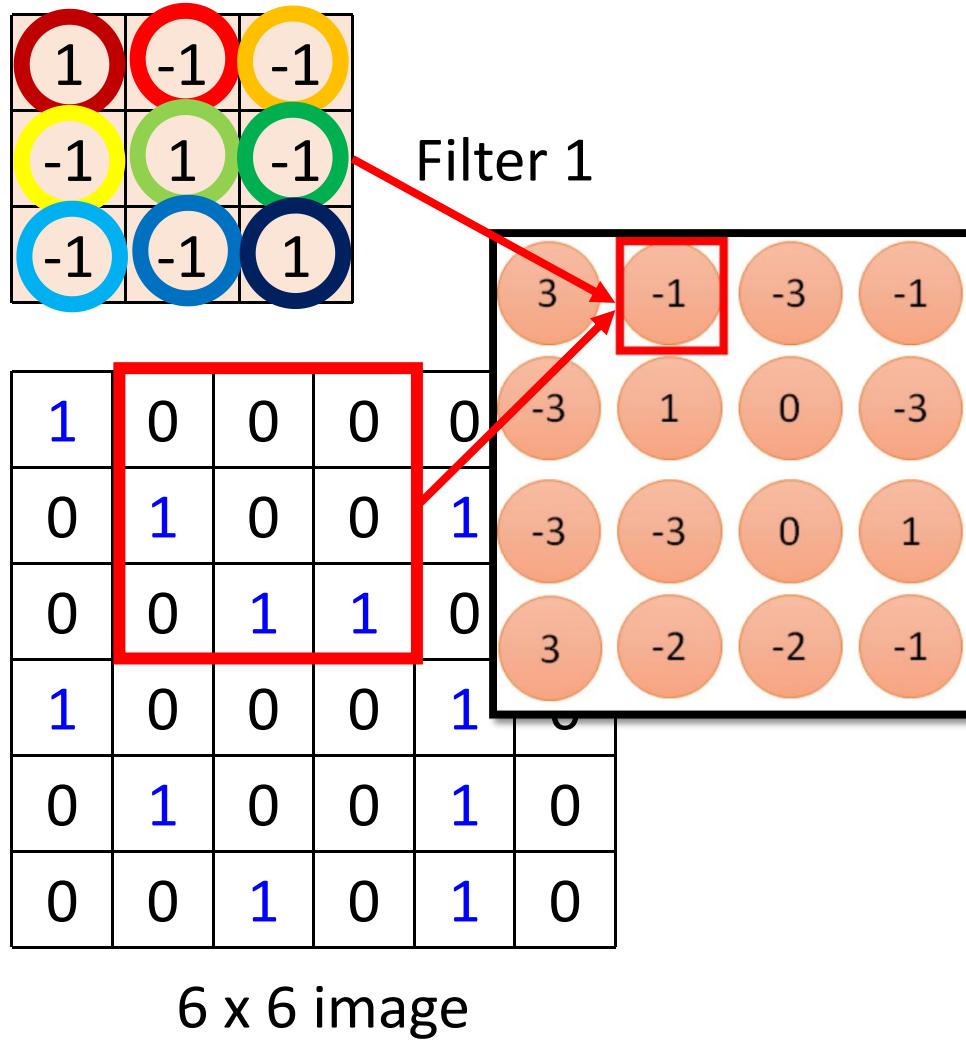
Filter 1



6 x 6 image

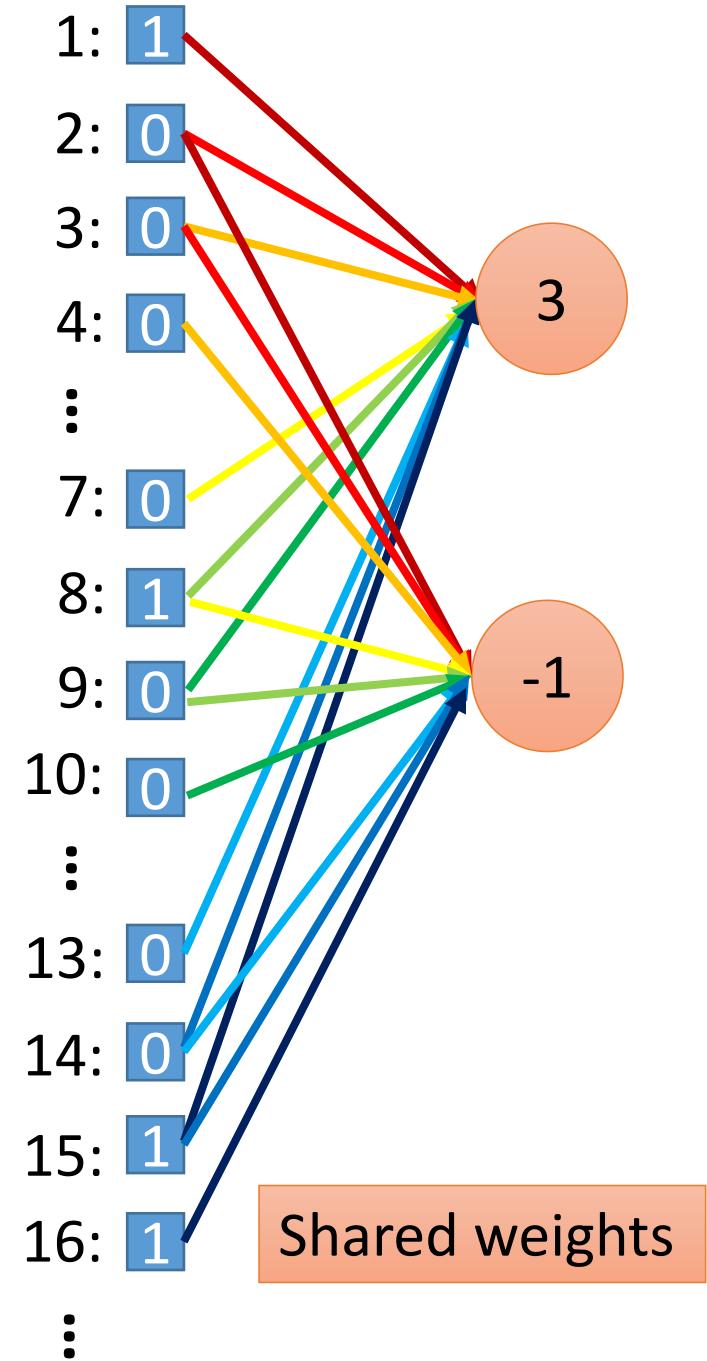
Less parameters!



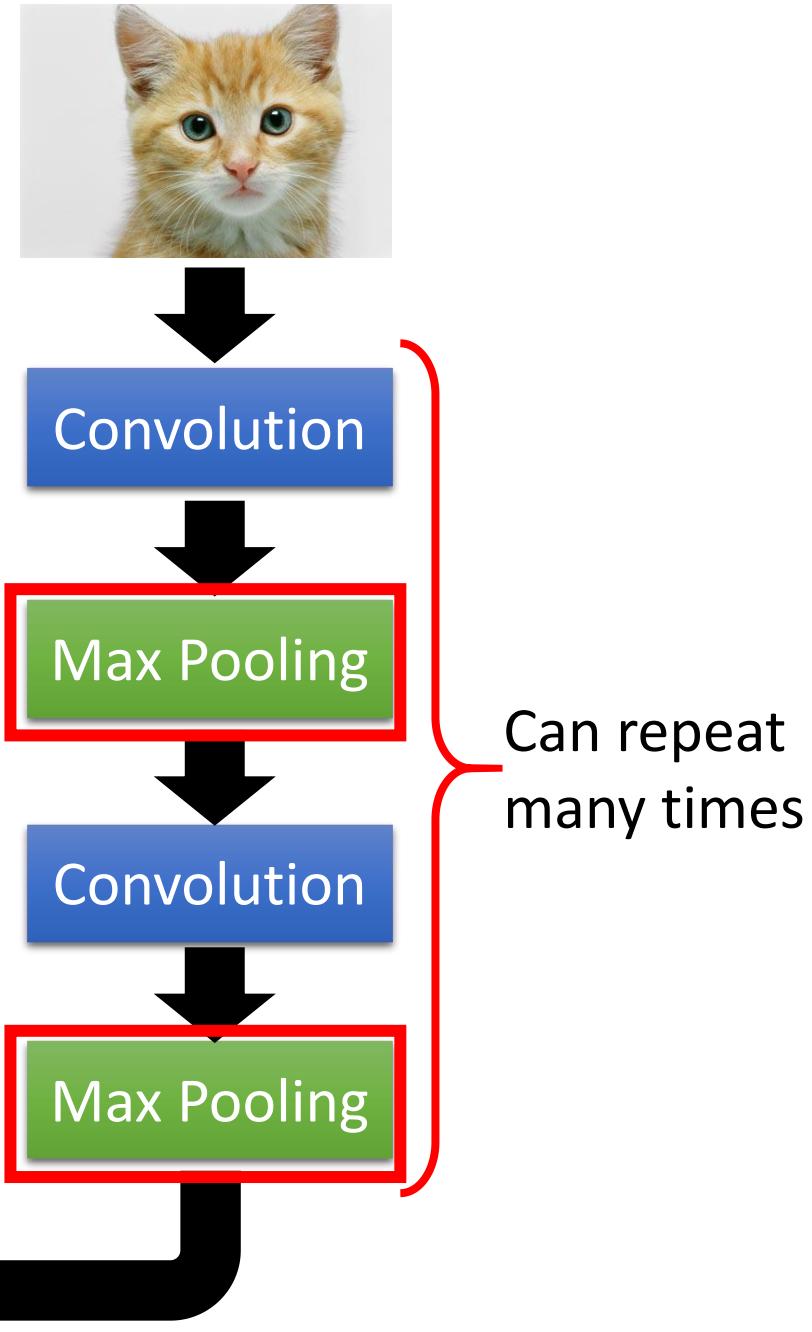
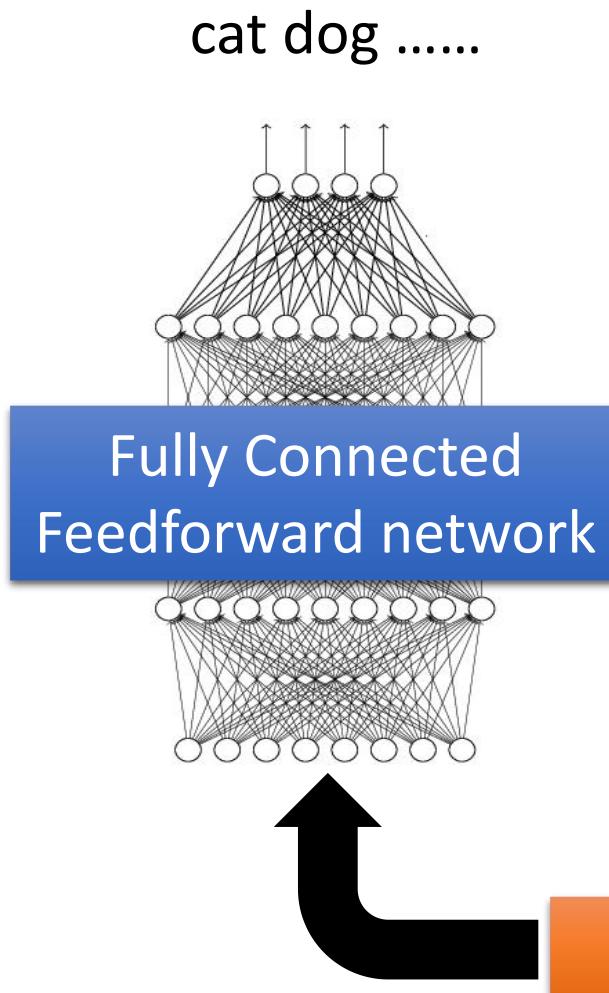


# Less parameters!

# Even less parameters!



# The whole CNN



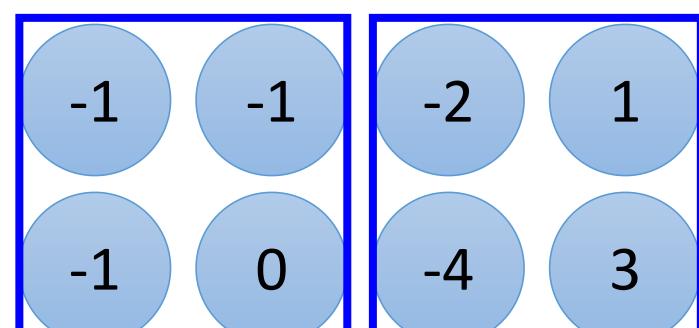
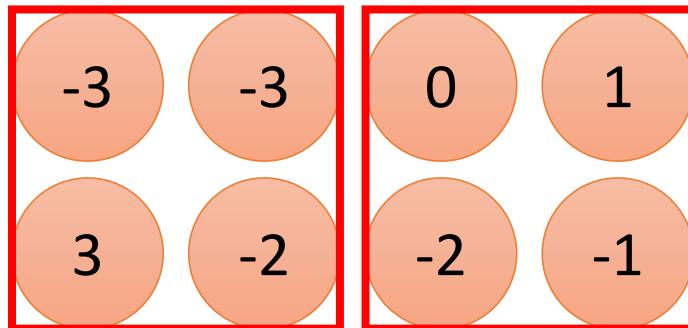
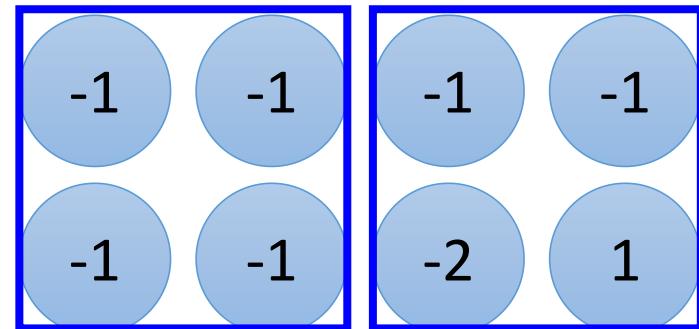
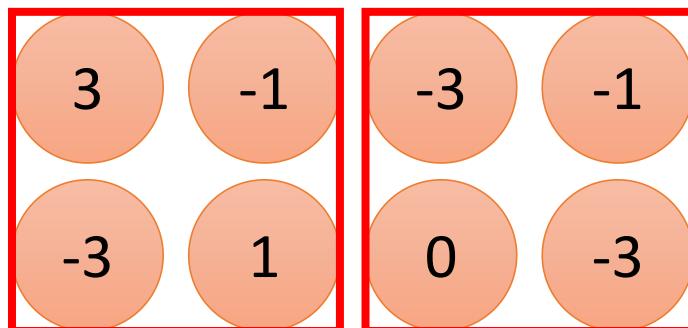
# CNN – Max Pooling

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

Filter 1

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

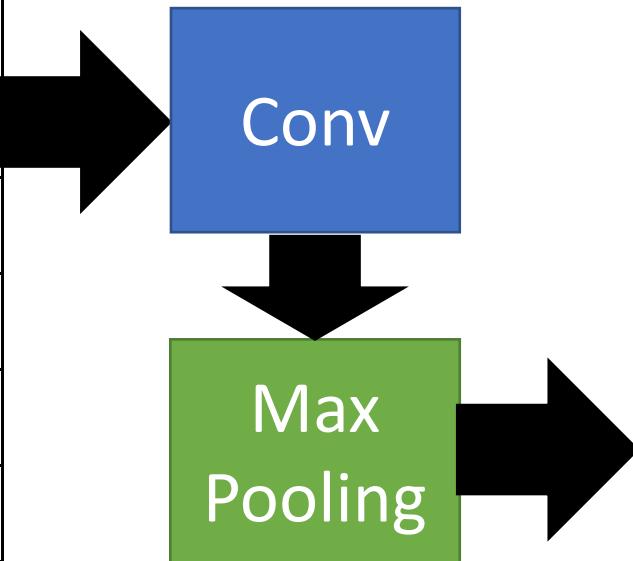
Filter 2



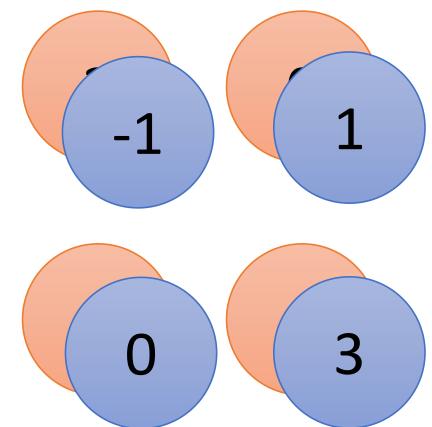
# CNN – Max Pooling

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



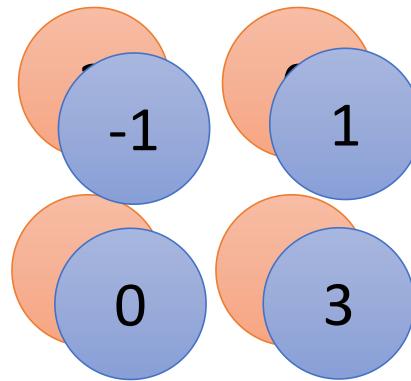
New image  
but smaller



2 x 2 image

Each filter  
is a channel

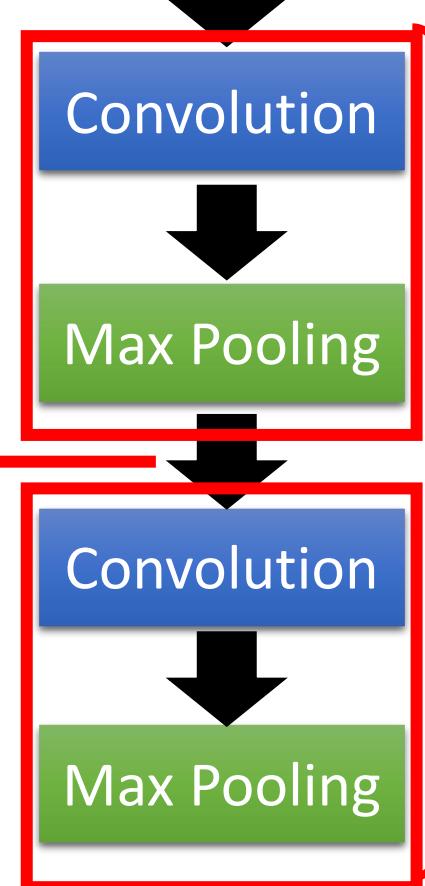
# The whole CNN



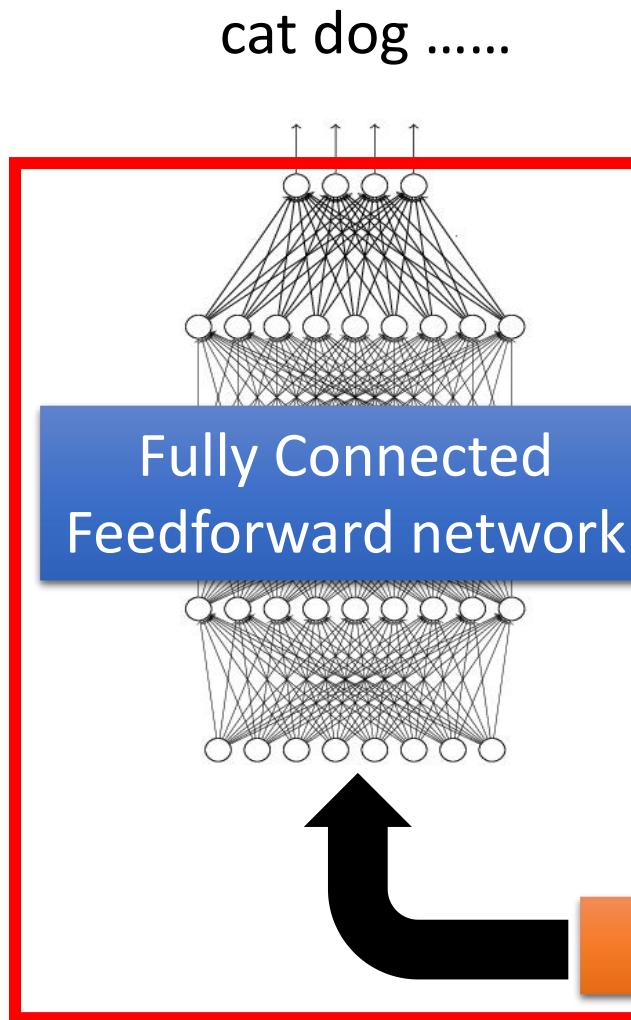
A new image

Smaller than the original image

The number of the channel is the number of filters



# The whole CNN



Convolution

Max Pooling

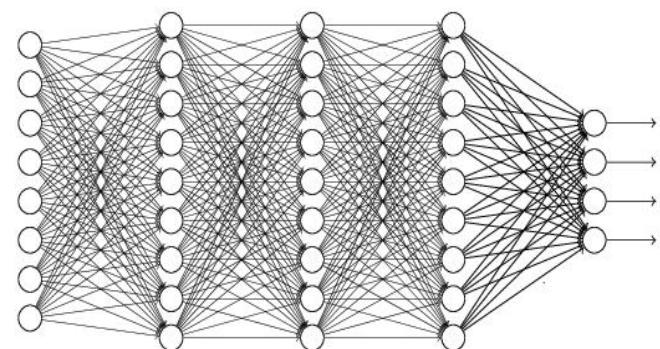
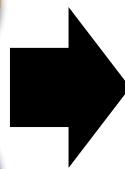
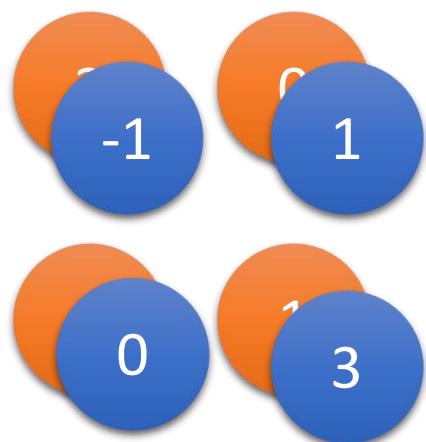
A new image

Convolution

Max Pooling

A new image

# Flatten

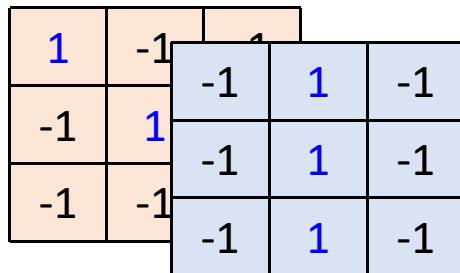


Fully Connected  
Feedforward network

## CNN in Keras

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

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

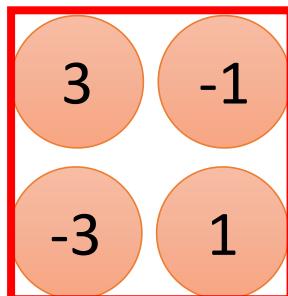


There are 25  
3x3 filters.

Input\_shape = ( 28, 28, 1)

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

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



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?

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

9

25 x 26 x 26

input

Convolution



Max Pooling



Convolution



Max Pooling

How many parameters  
for each filter?

225

50 x 11 x 11

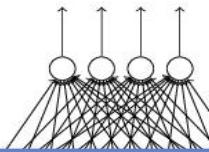
```
model2.add( MaxPooling2D( (2,2) ) )
```

50 x 5 x 5

## CNN in Keras

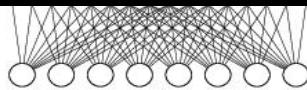
Only modified the *network structure* and *input format (vector -> 3-D tensor)*

output



Fully Connected  
Feedforward network

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



1250

Flatten

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

input

$1 \times 28 \times 28$

Convolution

$25 \times 26 \times 26$

Max Pooling

$25 \times 13 \times 13$

Convolution

$50 \times 11 \times 11$

Max Pooling

$50 \times 5 \times 5$

# What does machine learn?

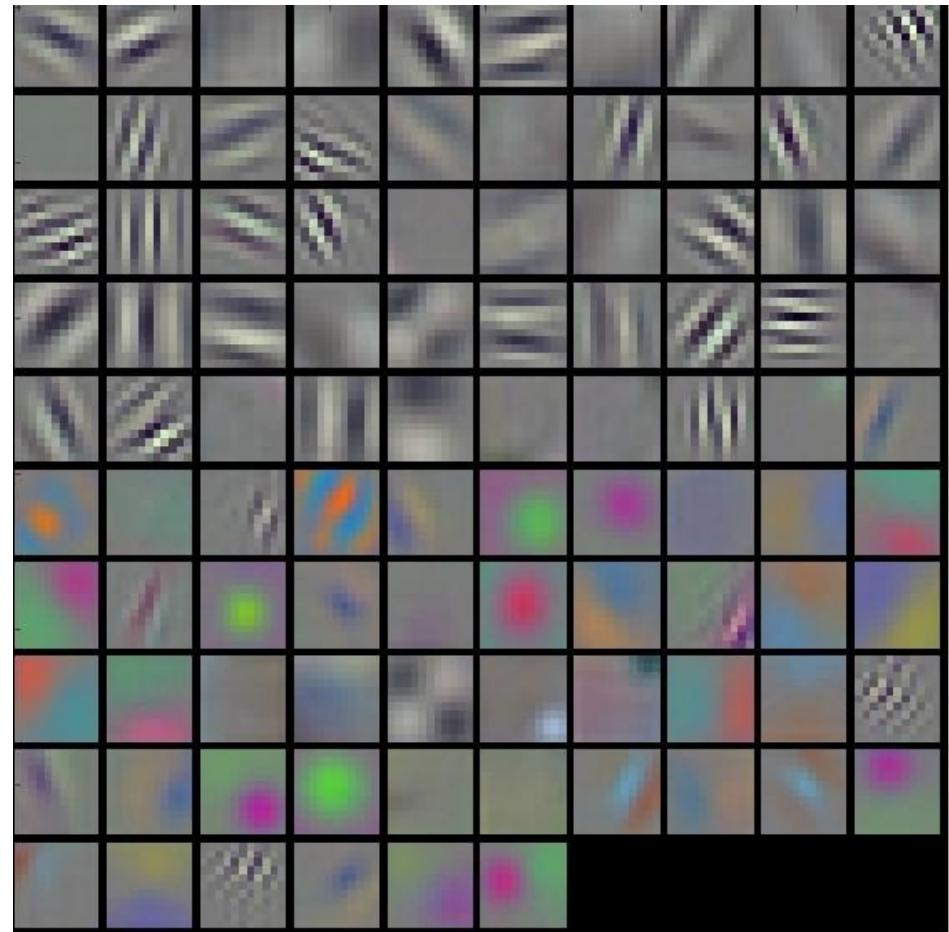


<http://newsneakernews.wpengine.netdna-cdn.com/wp-content/uploads/2016/11/rihanna-puma-creepervelvet-release-date-02.jpg>

# First Convolution Layer

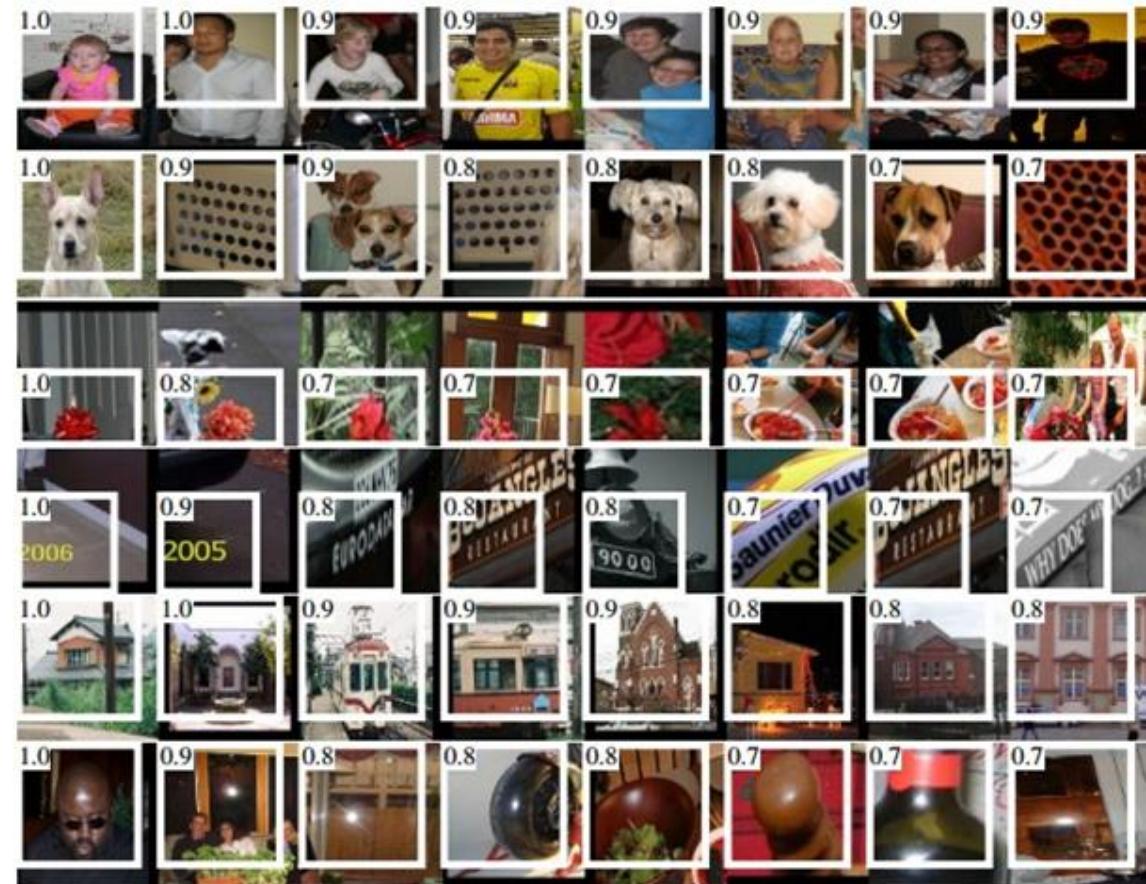
- Typical-looking filters on the trained first layer

11 x 11  
(AlexNet)



# How about higher layers?

- Which images make a specific neuron activate



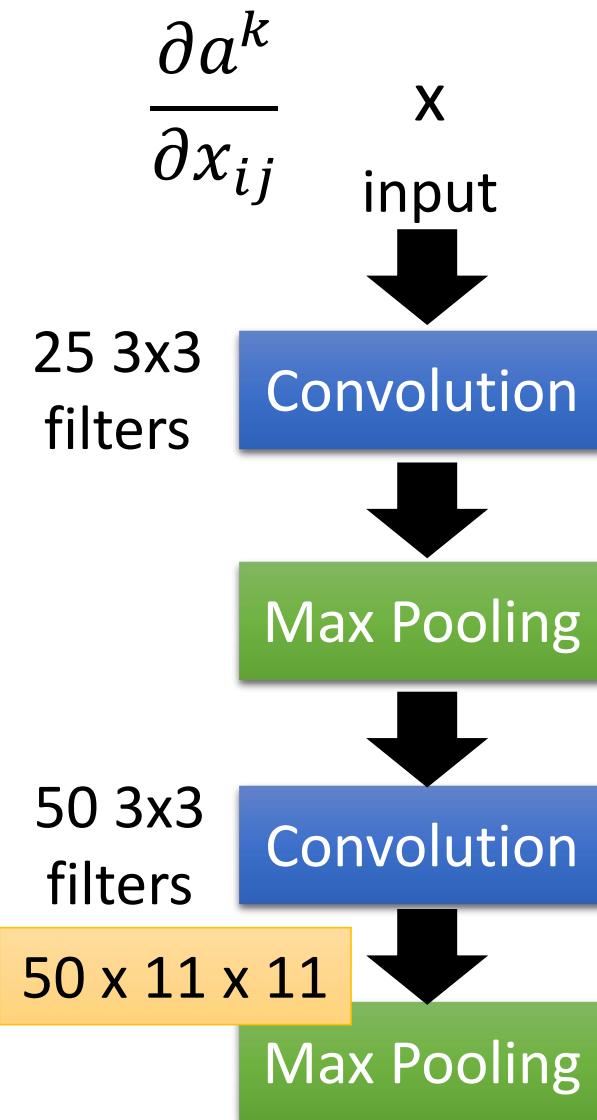
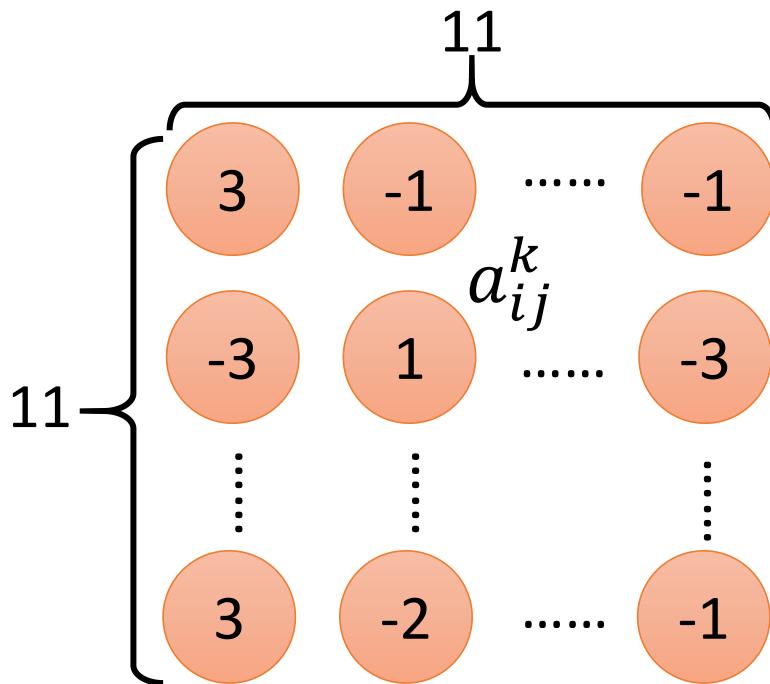
Ross Girshick, Jeff  
Donahue, Trevor  
Darrell, Jitendra Malik, "Rich  
feature hierarchies for accurate  
object detection and semantic  
segmentation", CVPR, 2014

## What does CNN learn?

The output of the k-th filter is a  $11 \times 11$  matrix.

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

$$x^* = \underset{x}{\operatorname{argmax}} a^k \text{ (gradient ascent)}$$

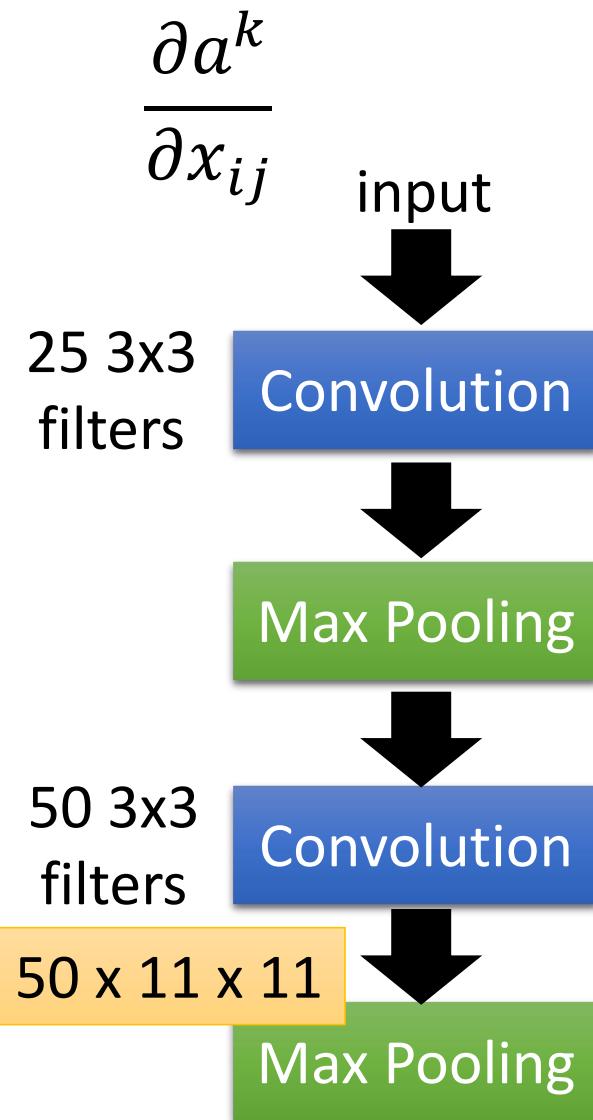
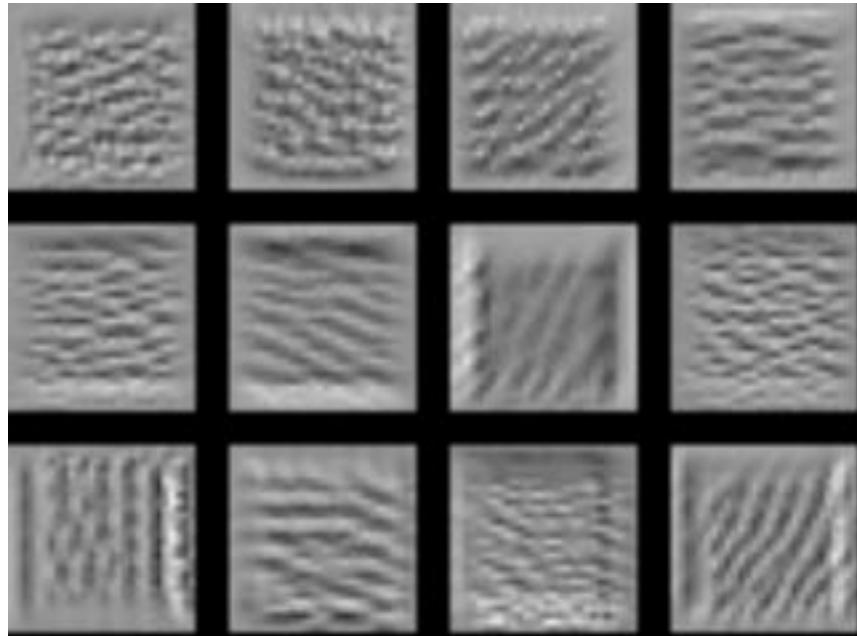


## What does CNN learn?

The output of the k-th filter is a  $11 \times 11$  matrix.

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

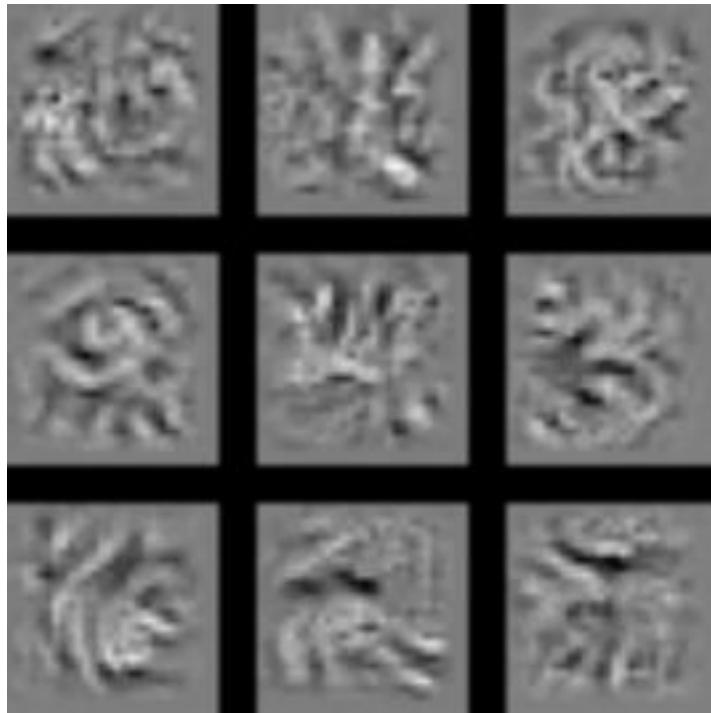
$$x^* = \underset{x}{\operatorname{argmax}} a^k \text{ (gradient ascent)}$$



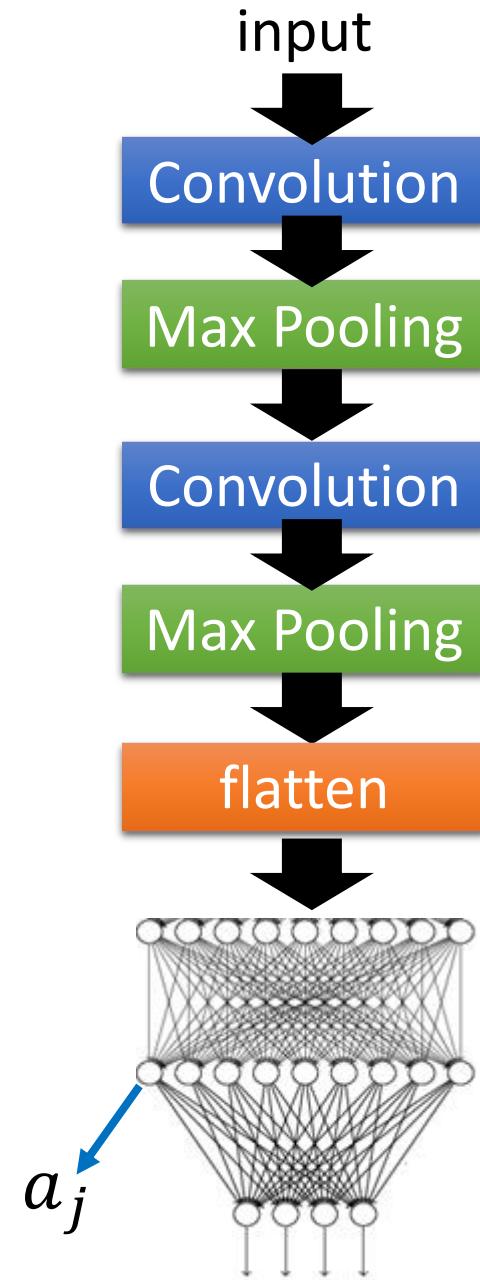
# What does CNN learn?

Find an image maximizing the output of neuron:

$$x^* = \operatorname{argmax}_x a_j$$



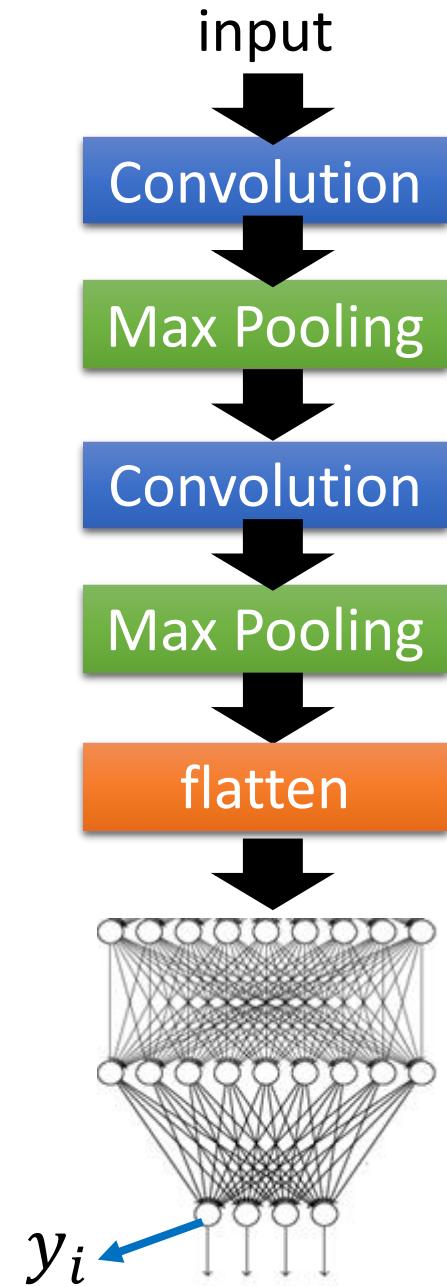
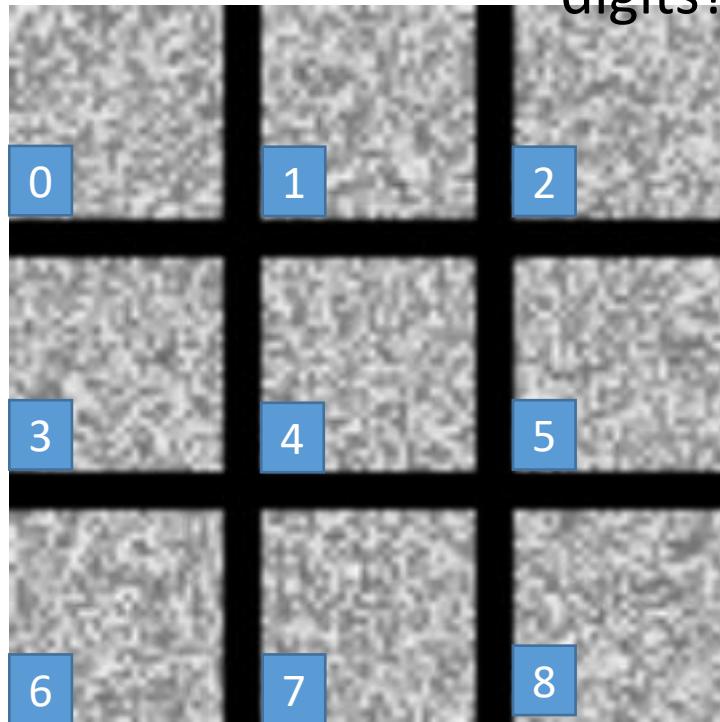
Each figure corresponds to a neuron



# What does CNN learn?

$$x^* = \operatorname{argmax}_x y^i$$

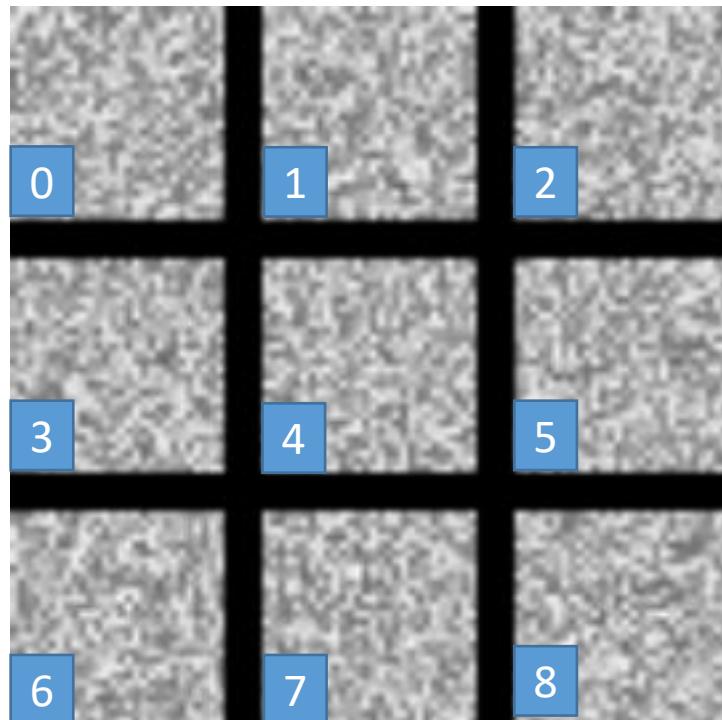
Can we see digits?



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

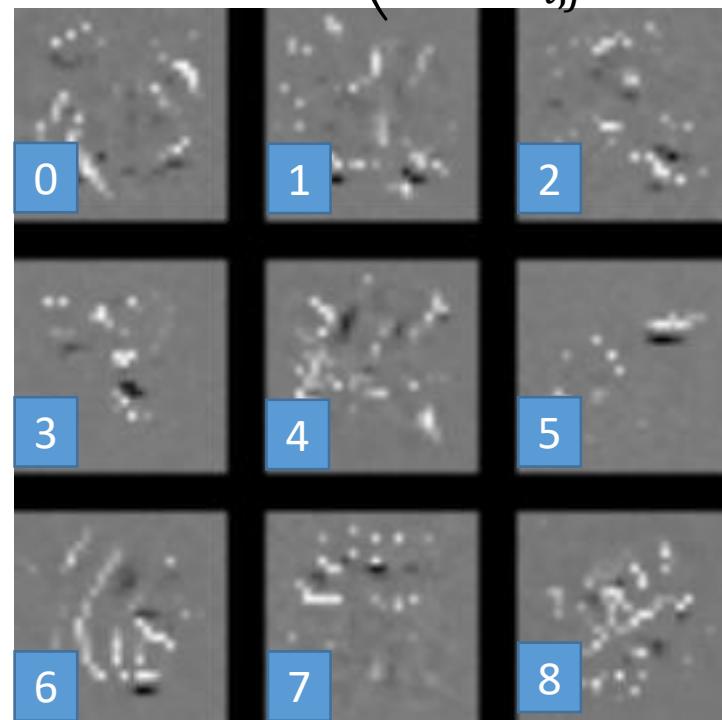
# What does CNN learn?

$$x^* = \operatorname{argmax}_x y^i$$



Over all  
pixel values

$$x^* = \operatorname{argmax}_x \left( y^i - \sum_{i,j} |x_{ij}| \right)$$



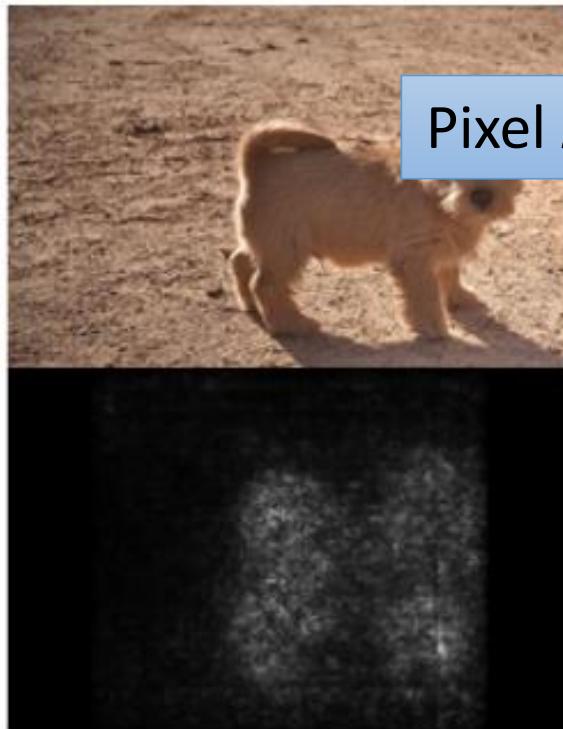


Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR, 2014

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

$y_k$ : the predicted class of the model

Pixel  $x_{ij}$



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



True Label: Pomeranian



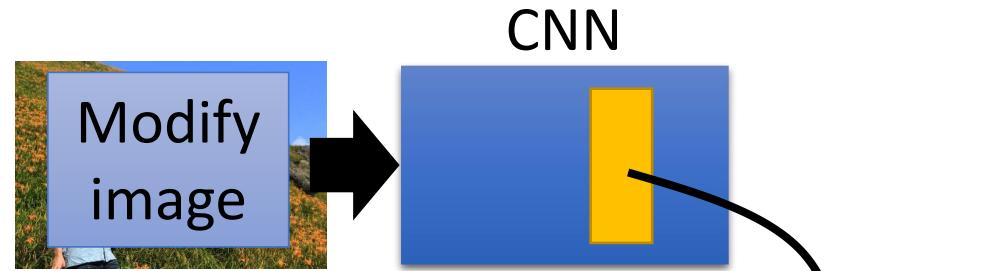
True Label: Car Wheel



True Label: Afghan Hound

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 .....



# Deep Dream

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



<http://deepdreamgenerator.com/>

# Deep Style

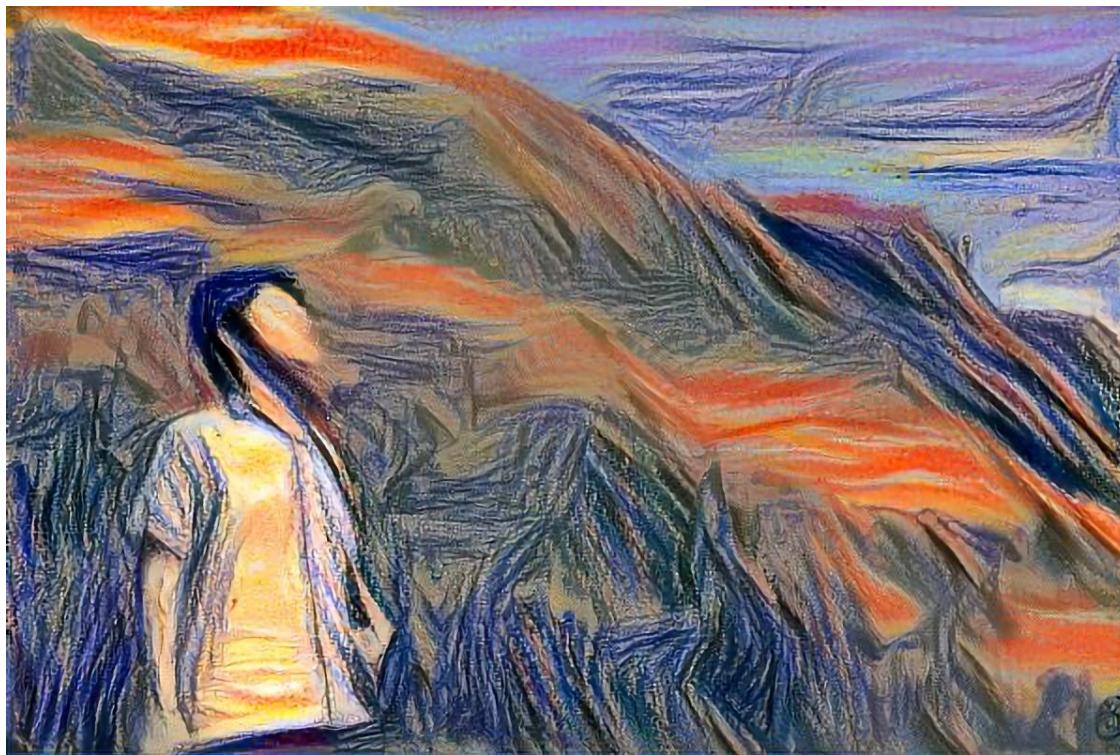
- Given a photo, make its style like famous paintings



<https://dreamscopeapp.com/>

# Deep Style

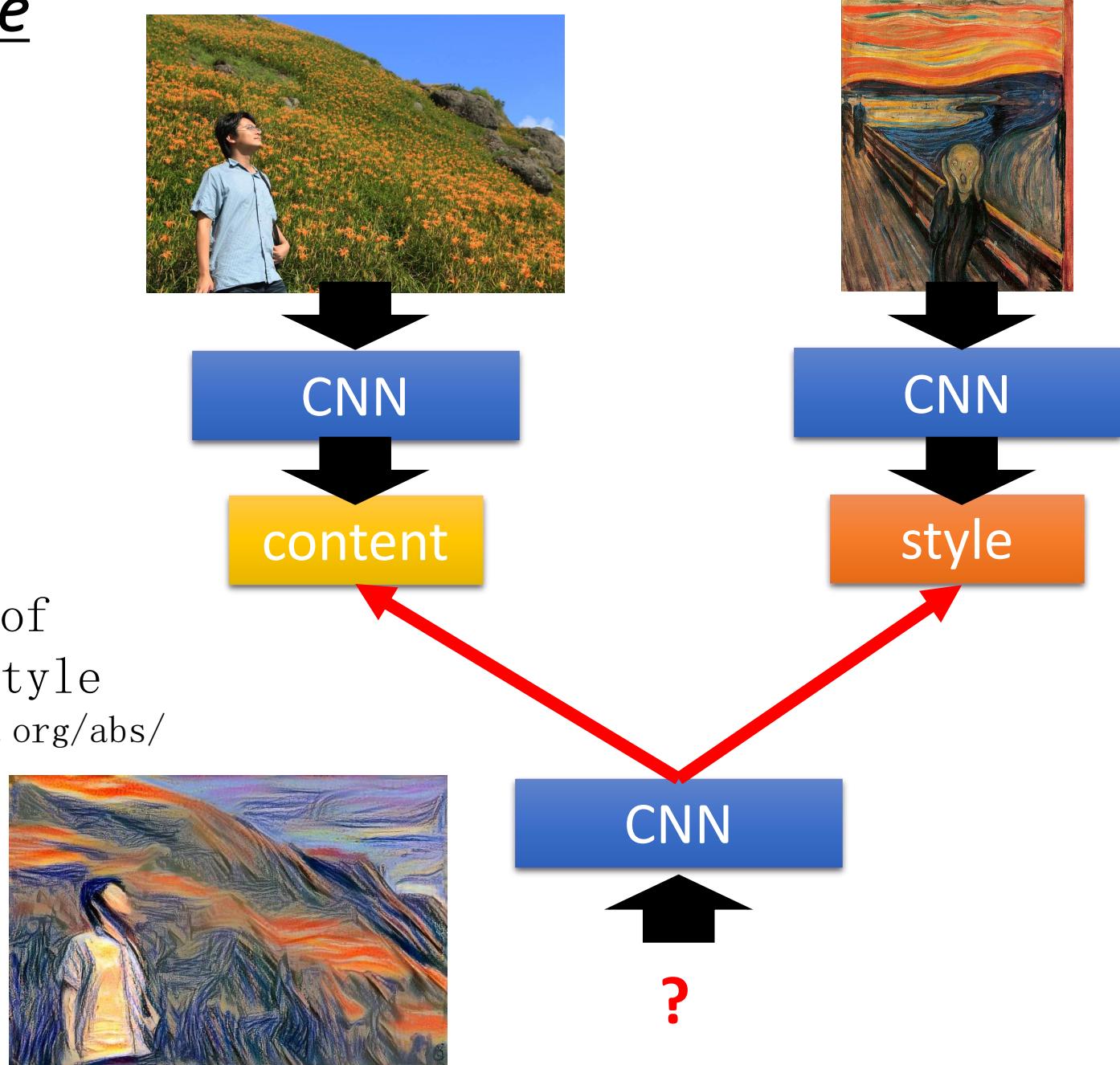
- Given a photo, make its style like famous paintings



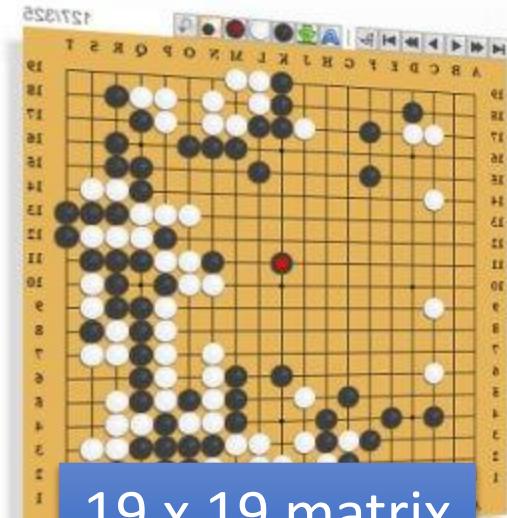
<https://dreamscopeapp.com/>

# Deep Style

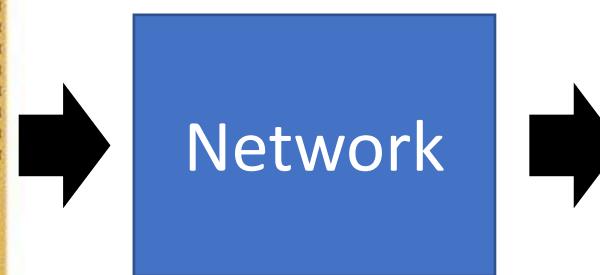
A Neural  
Algorithm of  
Artistic Style  
[https://arxiv.org/abs/  
1508.06576](https://arxiv.org/abs/1508.06576)



# More Application: Playing Go



19 x 19 matrix  
(image)



Next move  
(19 x 19  
positions)

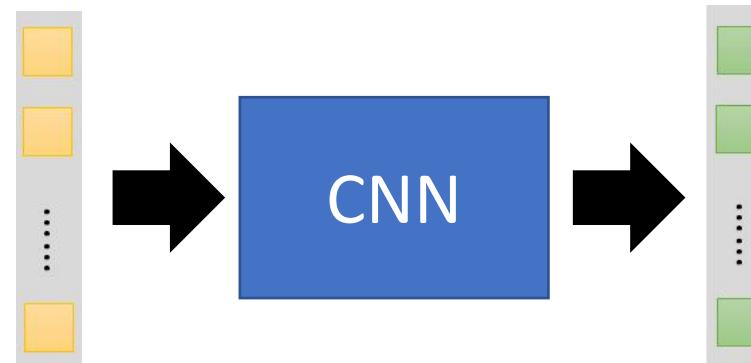
Black: 1  
white: -1  
none: 0

Fully-connected feedforward  
network can be used

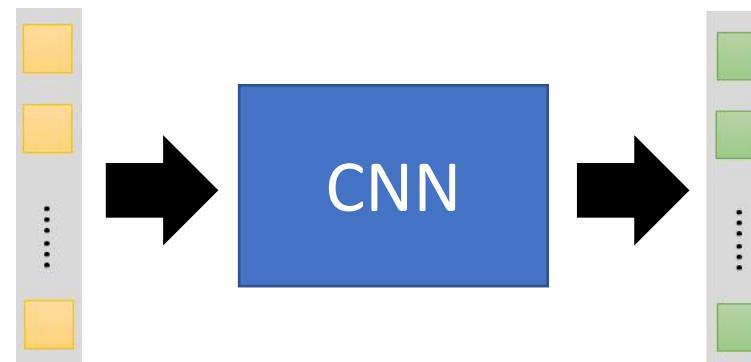
But CNN performs much better.

# More Application: Playing Go

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



Target:  
“天元” = 1  
else = 0

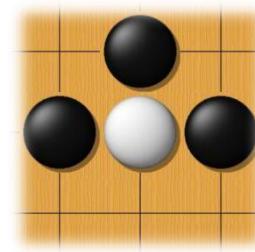


Target:  
“五之 5” = 1  
else = 0

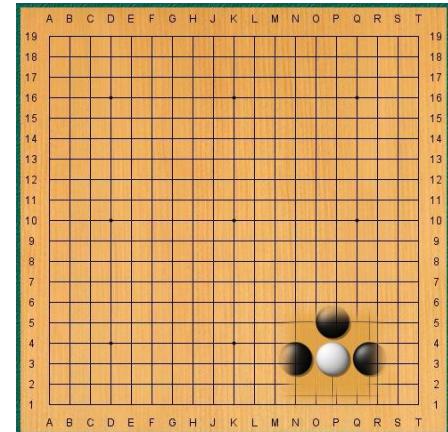
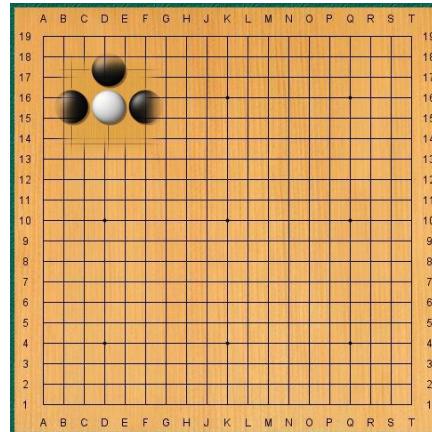
# Why CNN for playing Go?

- Some patterns are much smaller than the whole image

Alpha Go uses  $5 \times 5$  for first layer



- The same patterns appear in different regions.



# Why CNN for playing Go?

- Subsampling the pixels will not change the object

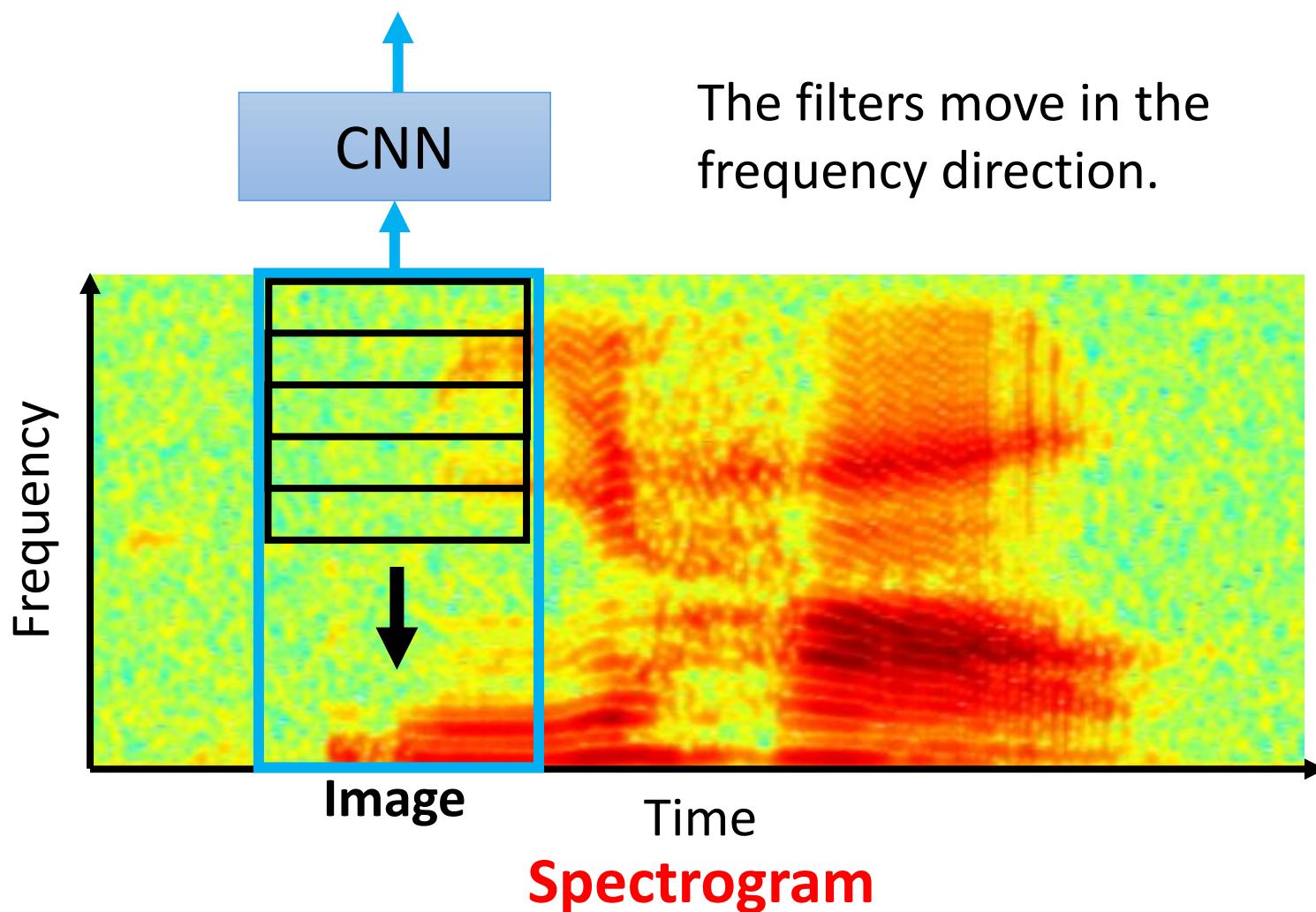


Max Pooling

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 Alpha Go does not use Max Pooling ..... Extended Data Table 3 additionally show the results of training with  $k = 128, 256$  and  $384$  filters.

# More Application: Speech



# More Application: Text

