CSCE 636 Neural Networks (Deep Learning)

Lecture 7: Deep Learning for Computer Vision

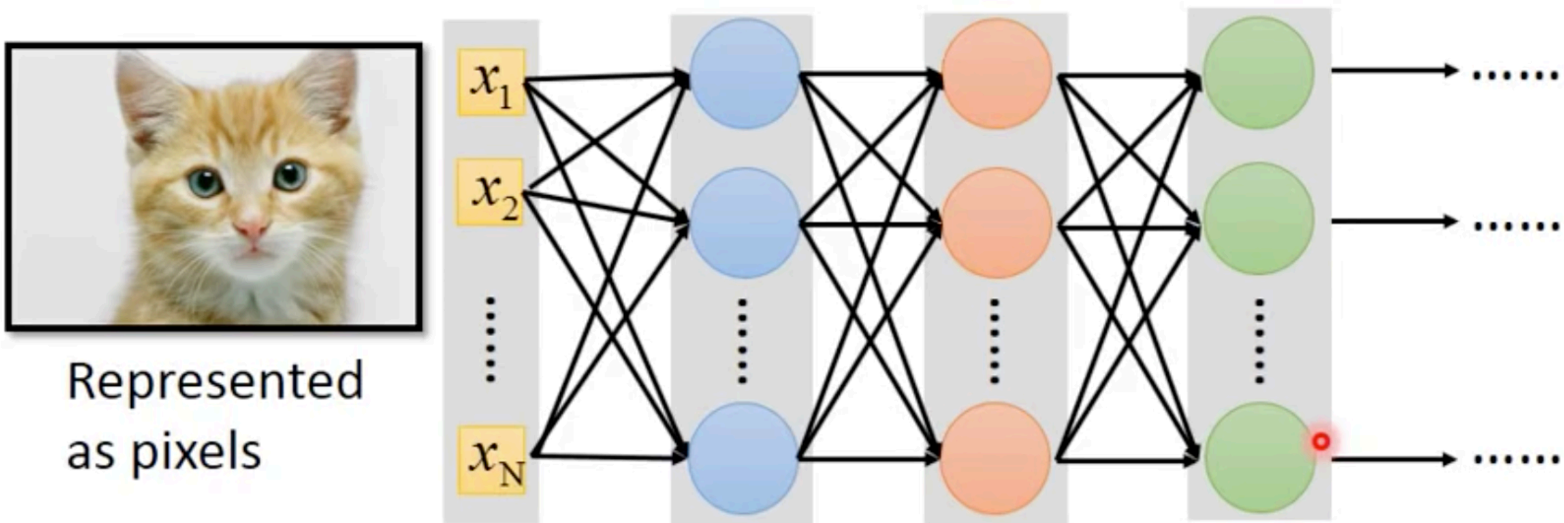
Anxiao (Andrew) Jiang

Based on the interesting lecture of Prof. Hung-yi Lee,
https://www.youtube.com/watch?v=FrKWiRv254g&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=19

Convolutional Neural Network (CNN)

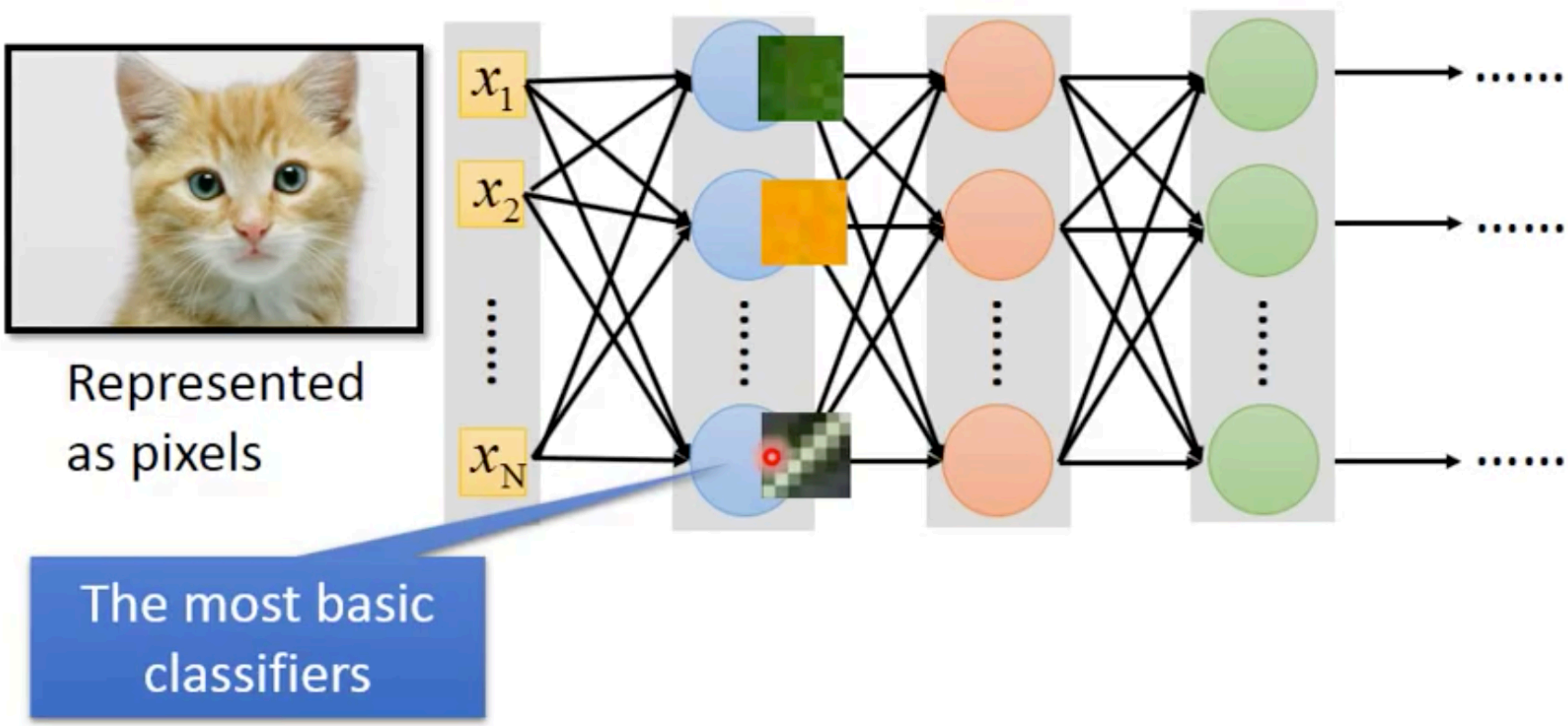
Why CNN for Image?

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



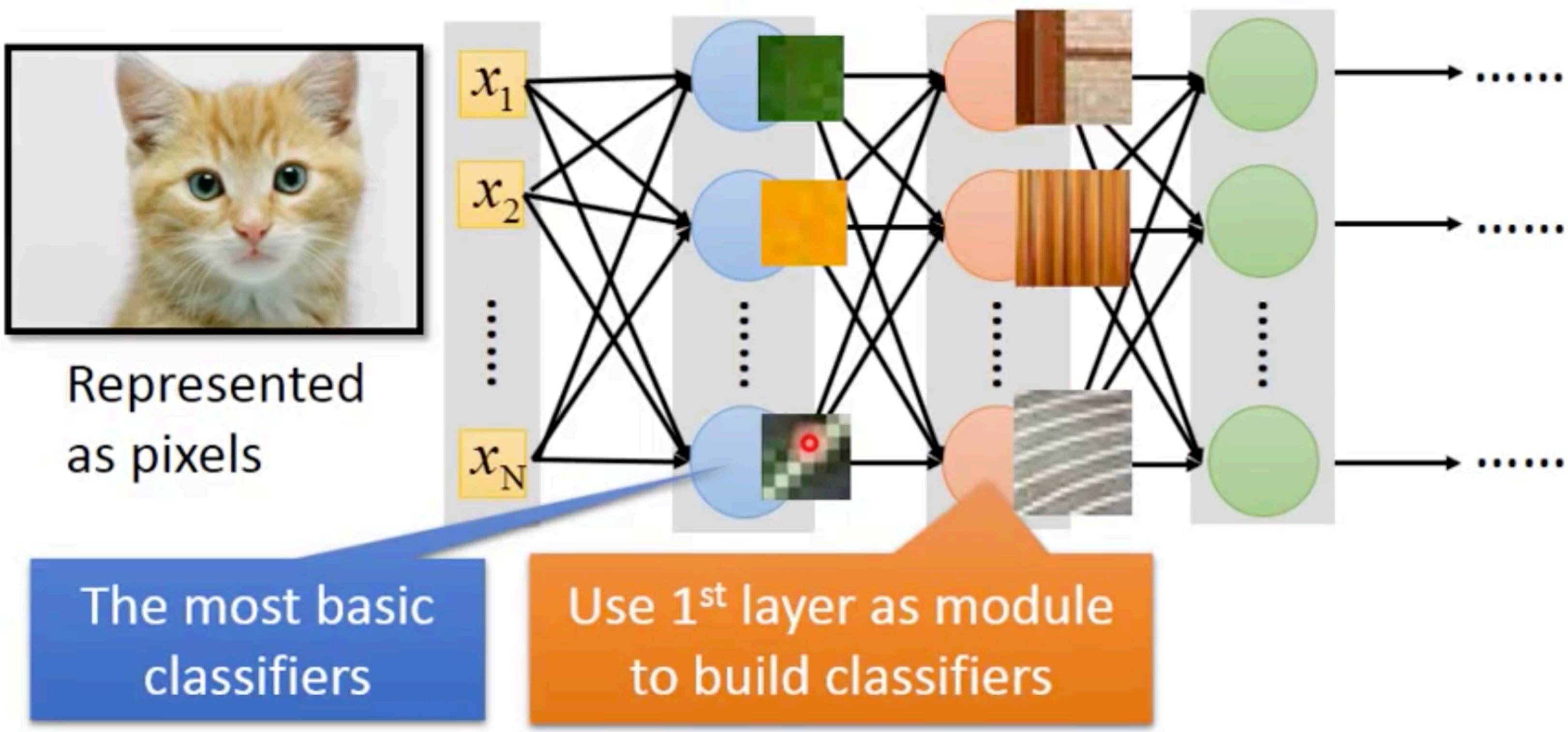
Why CNN for Image?

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

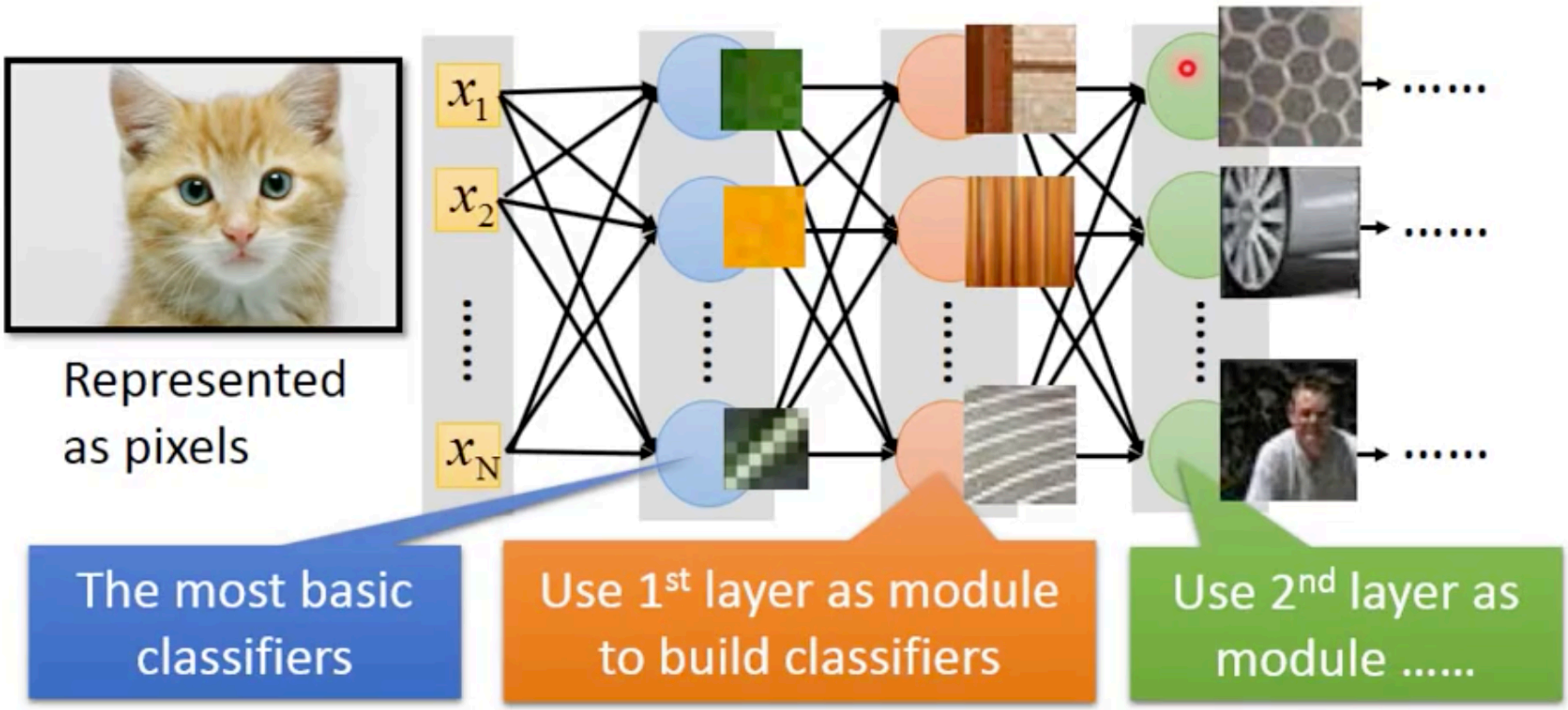


Why CNN for Image?

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



Why CNN for Image? [Zeiler, M. D., *ECCV 2014*]



Too many weights in a dense network!

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.

有沒有某一個 pattern 出現

Why CNN for Image

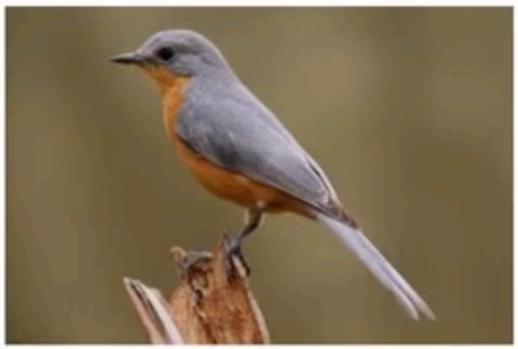
- Some patterns are much smaller than the whole image

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.



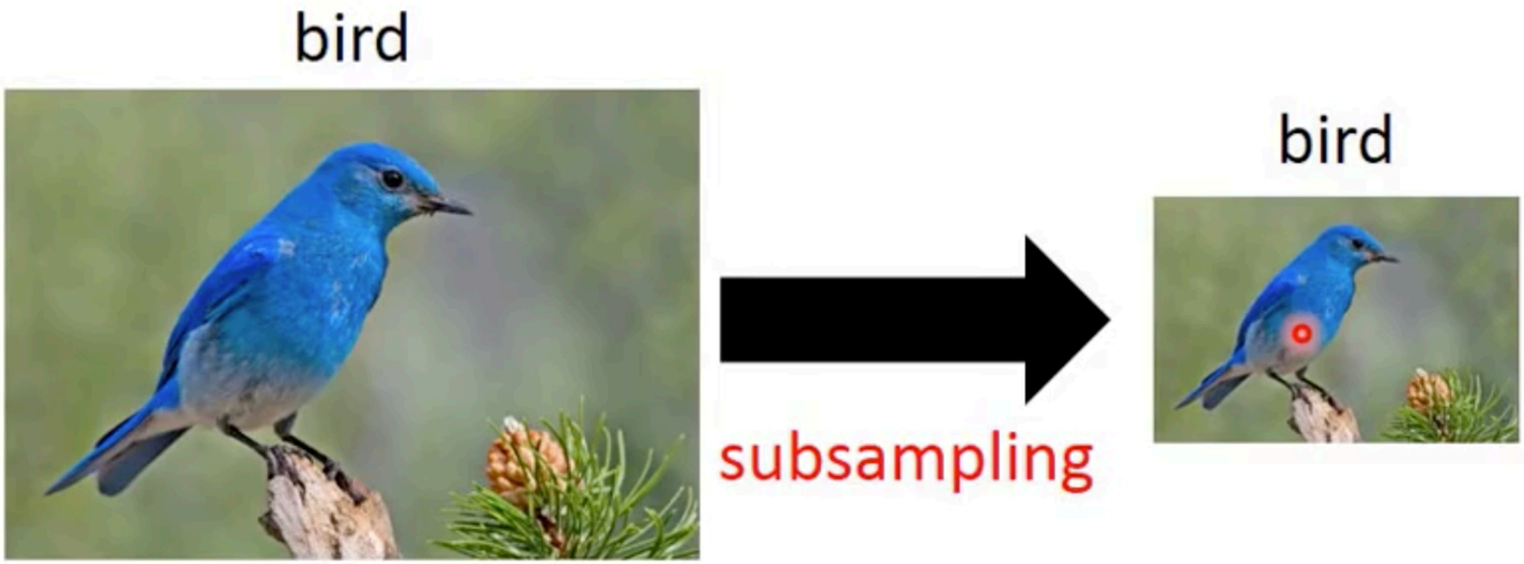
同樣的這個 pattern

Why CNN for Image

- The same patterns appear in different regions.

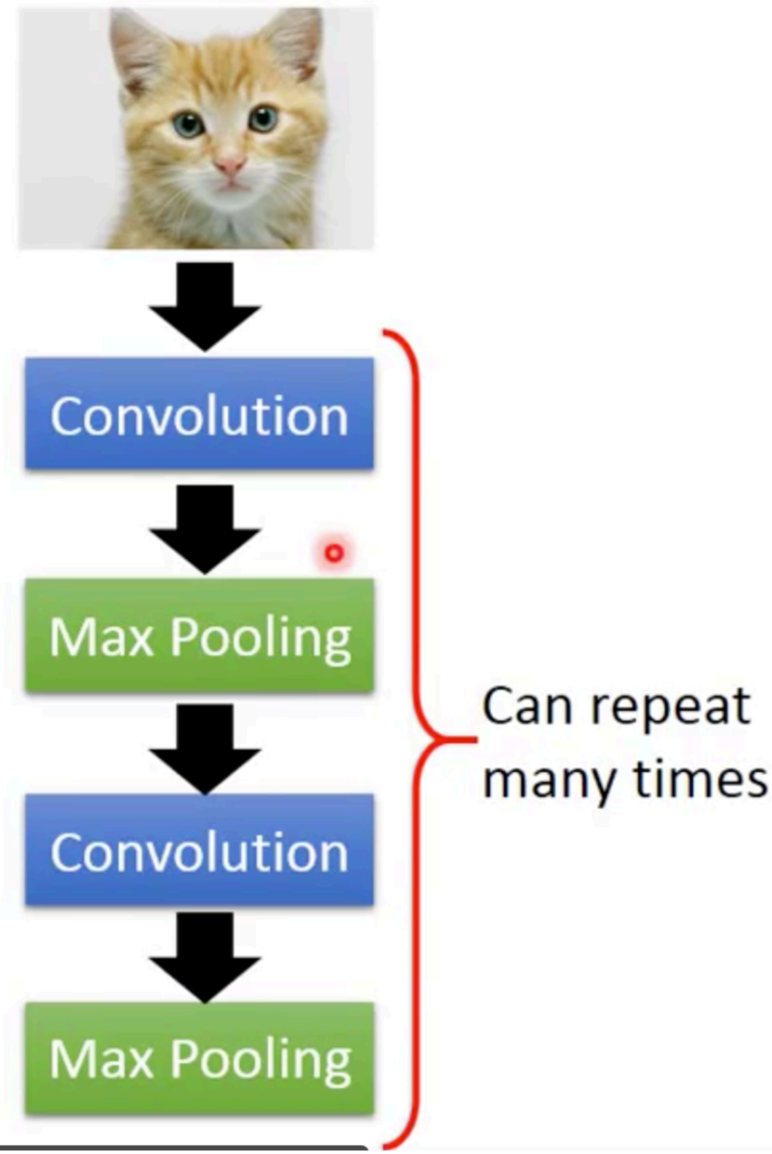
Why CNN for Image

- Subsampling the pixels will not change the object

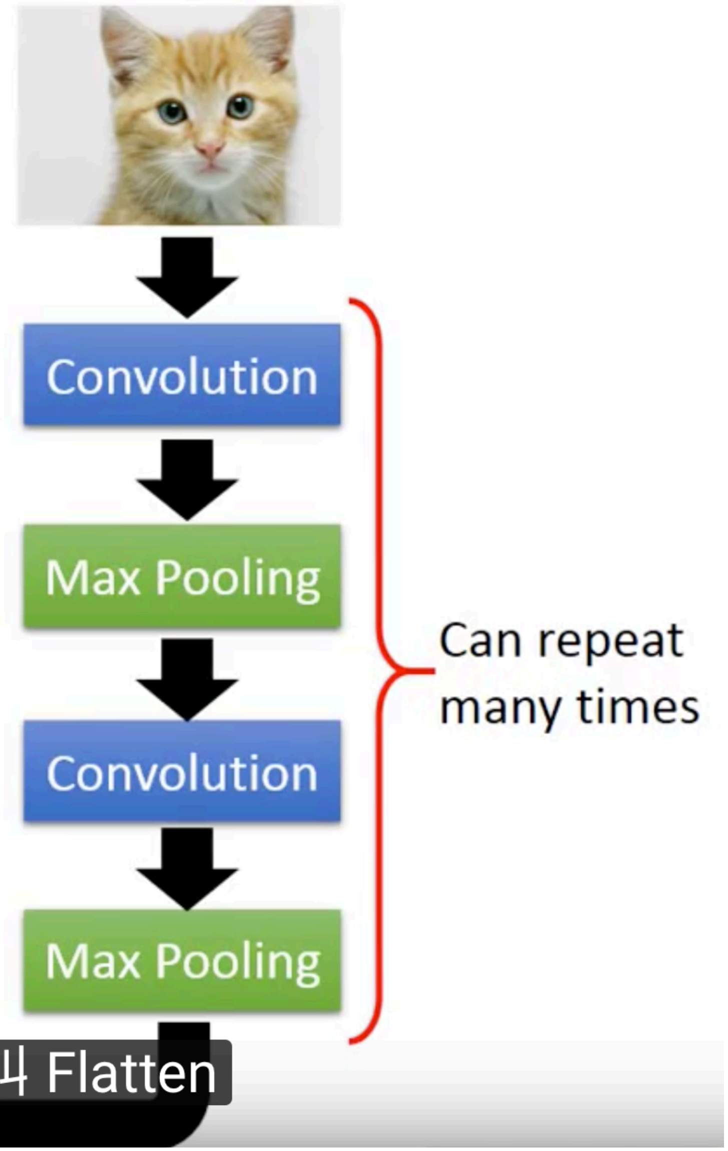


We can subsample the pixels to make image smaller

The whole CNN

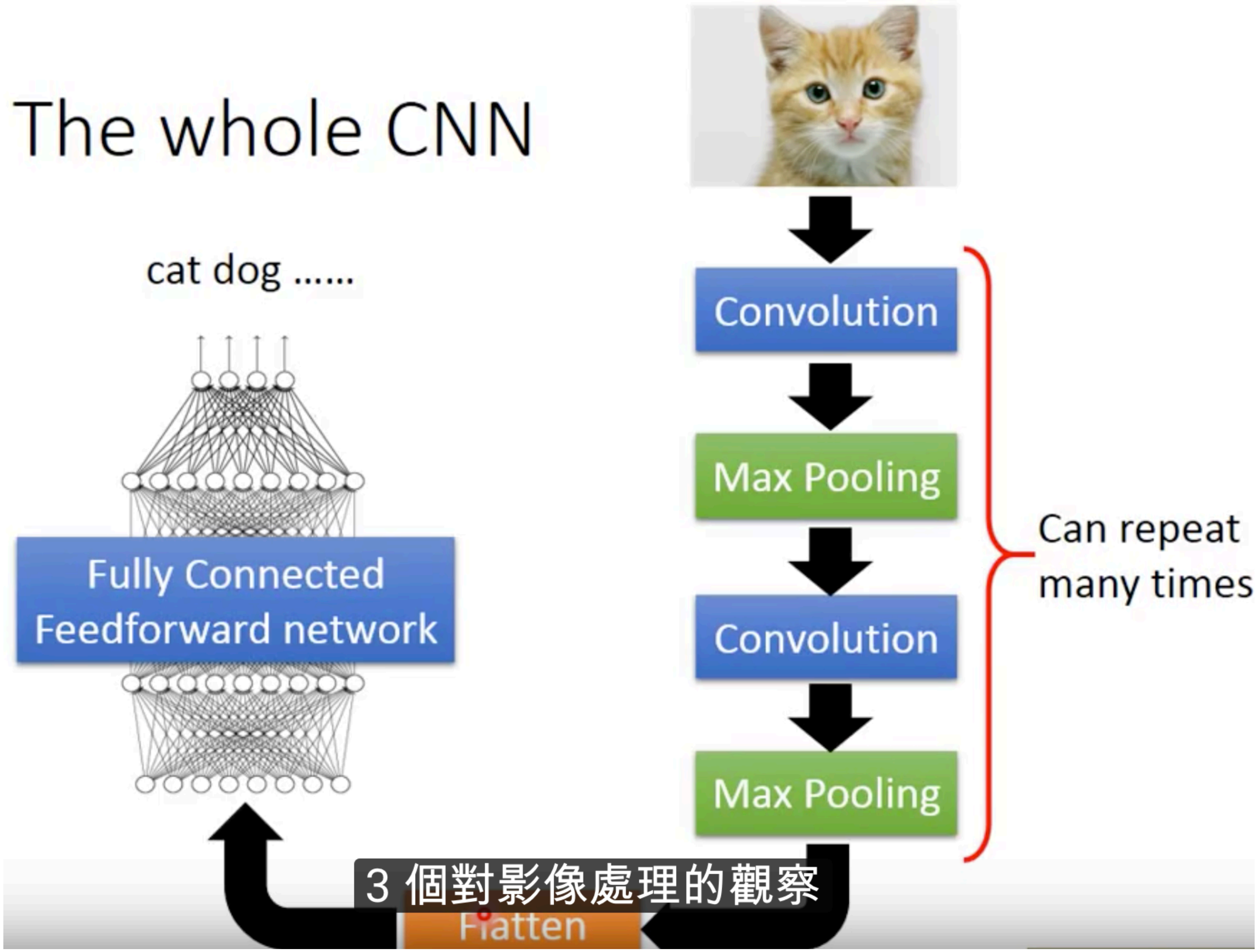


The whole CNN



你要做一件事情叫 Flatten

The whole CNN



The whole CNN

Property 1

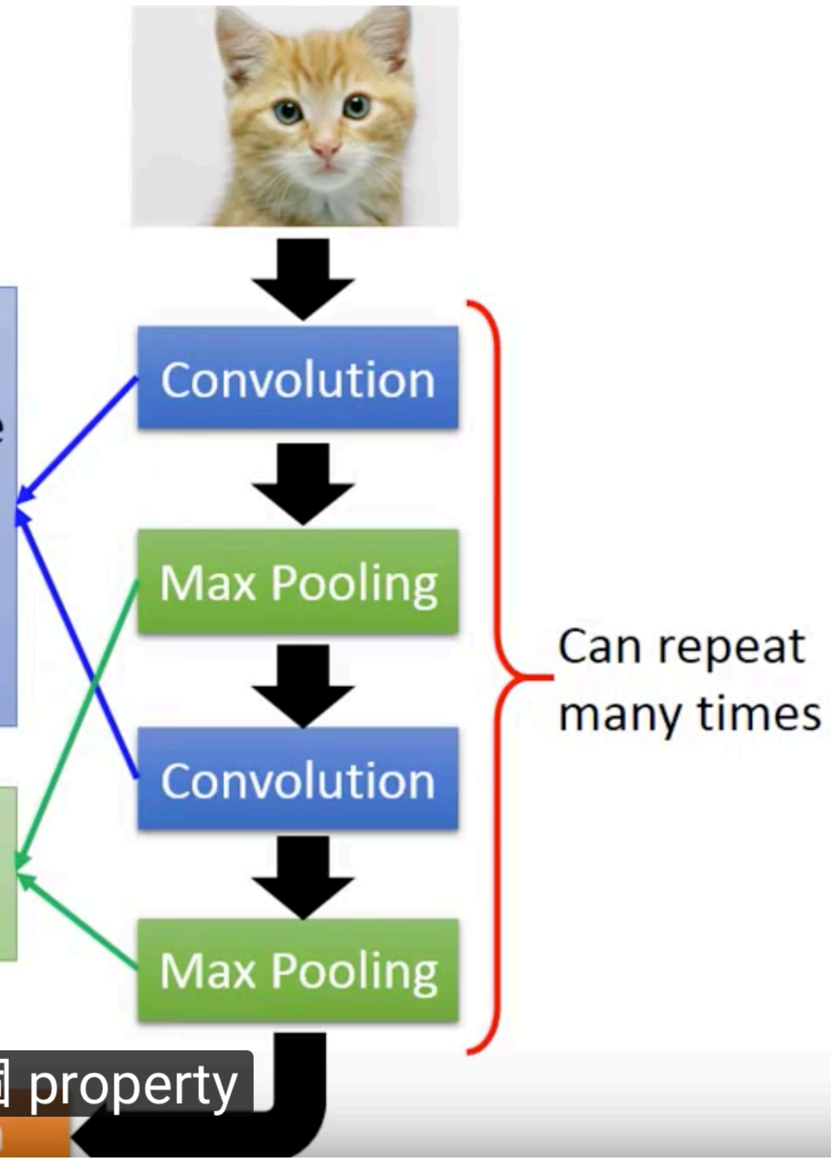
- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object



最後這個 property

CNN – Convolution

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 – Convolution

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

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

Filter 2

⋮

A filter corresponds to a set of neurons

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

⋮

CNN – Convolution

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

Filter 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

How does a filter operate?

CNN – Convolution

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

Filter 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 inner product (dot product)

CNN – Convolution

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

Filter 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

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

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 in the following slides.

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

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

CNN – Convolution

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

Filter 1

stride=1

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

6 x 6 image

3	-1	-3	-1

CNN – Convolution

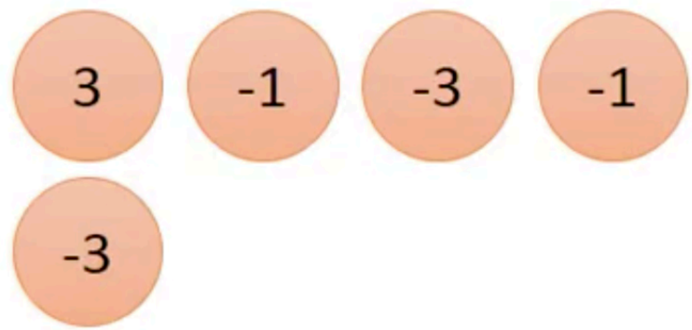
Filter 1

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



CNN – Convolution

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

Filter 1

stride=1

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

6 x 6 image

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

你就得到 -1

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

$\boxed{3}$	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
$\boxed{3}$	-2	-2	-1

出現最大的值

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

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

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

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

比如說，這裡會有一個 filter 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

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

再做 inner product 得到 -1

CNN – Convolution

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

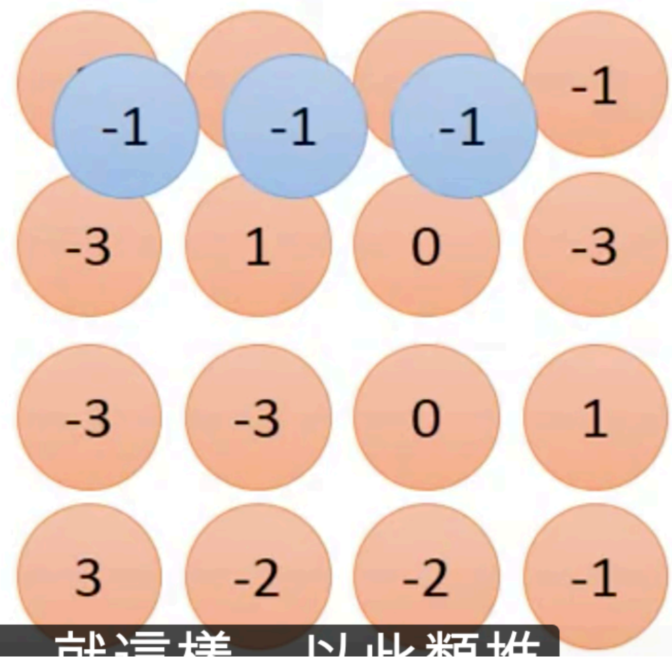
Filter 2

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



CNN – Convolution

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

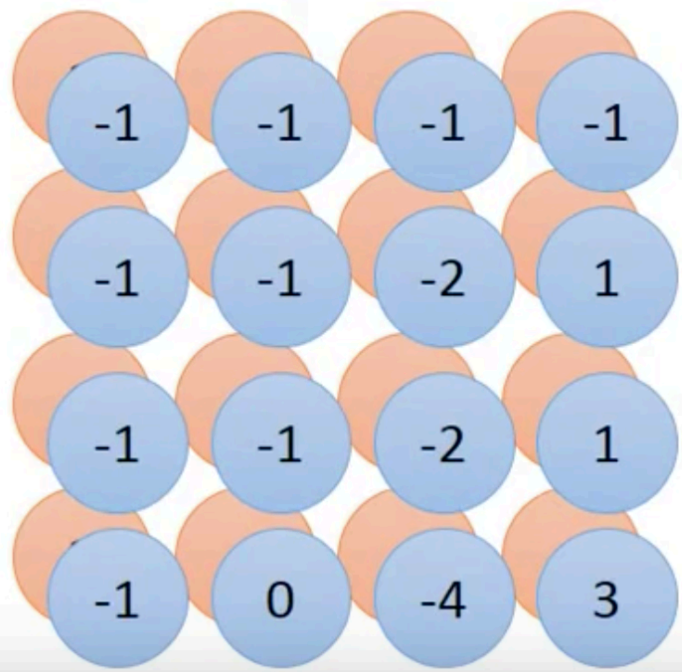
Filter 2

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



CNN – Convolution

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

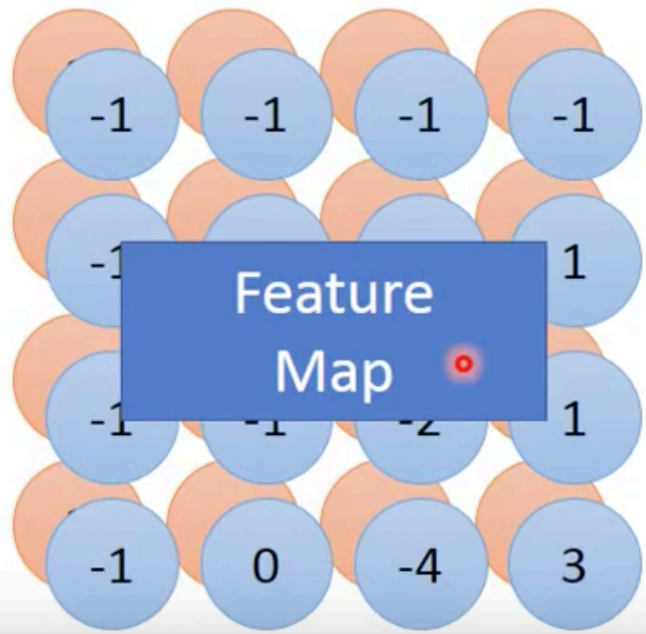
Filter 2

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

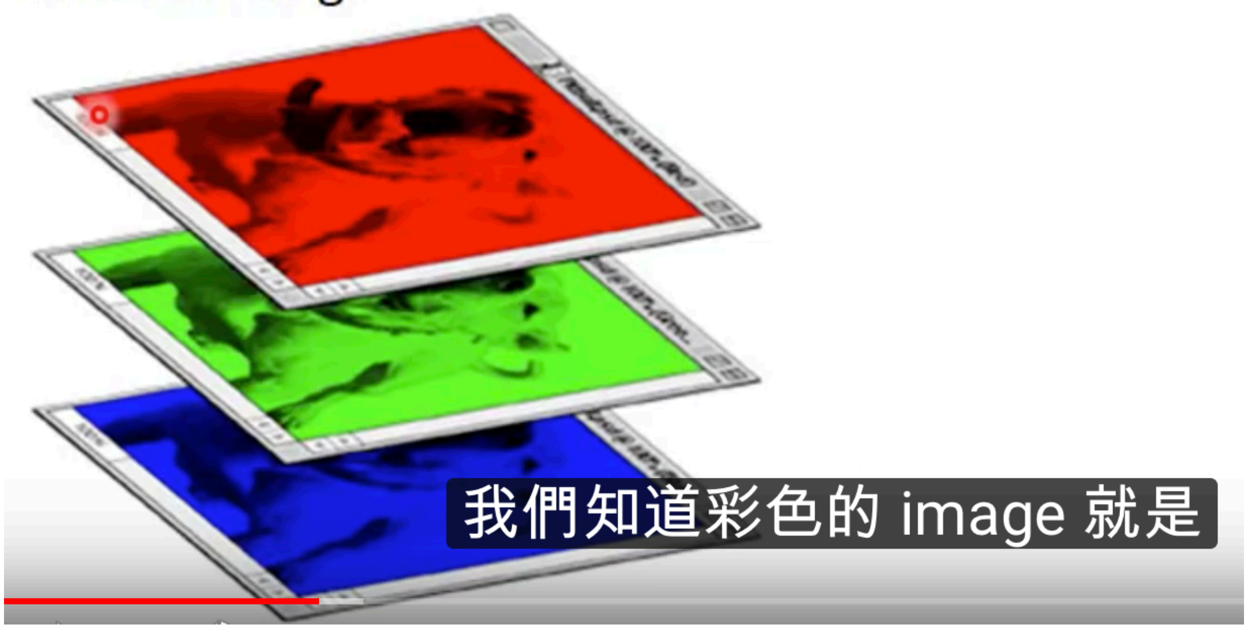
Do the same process for every filter



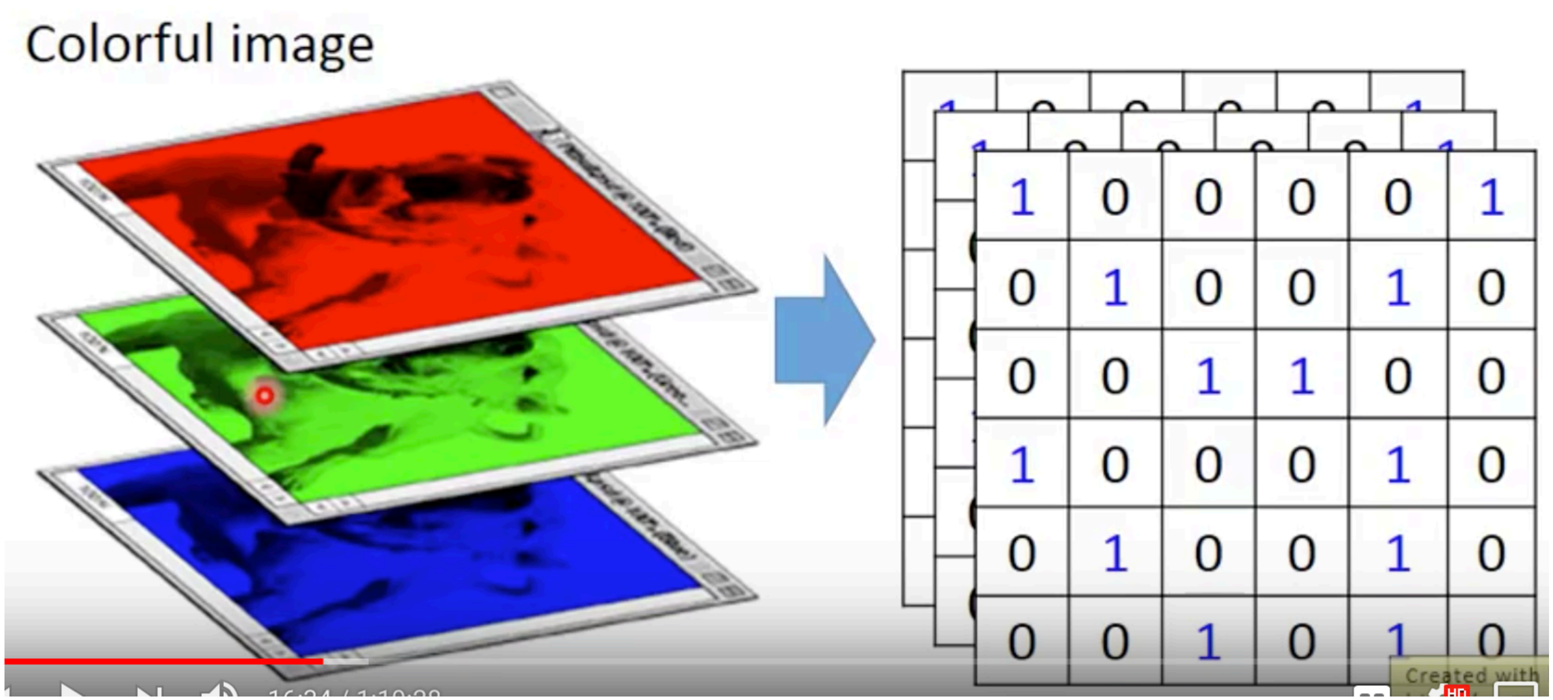
4 x 4 image

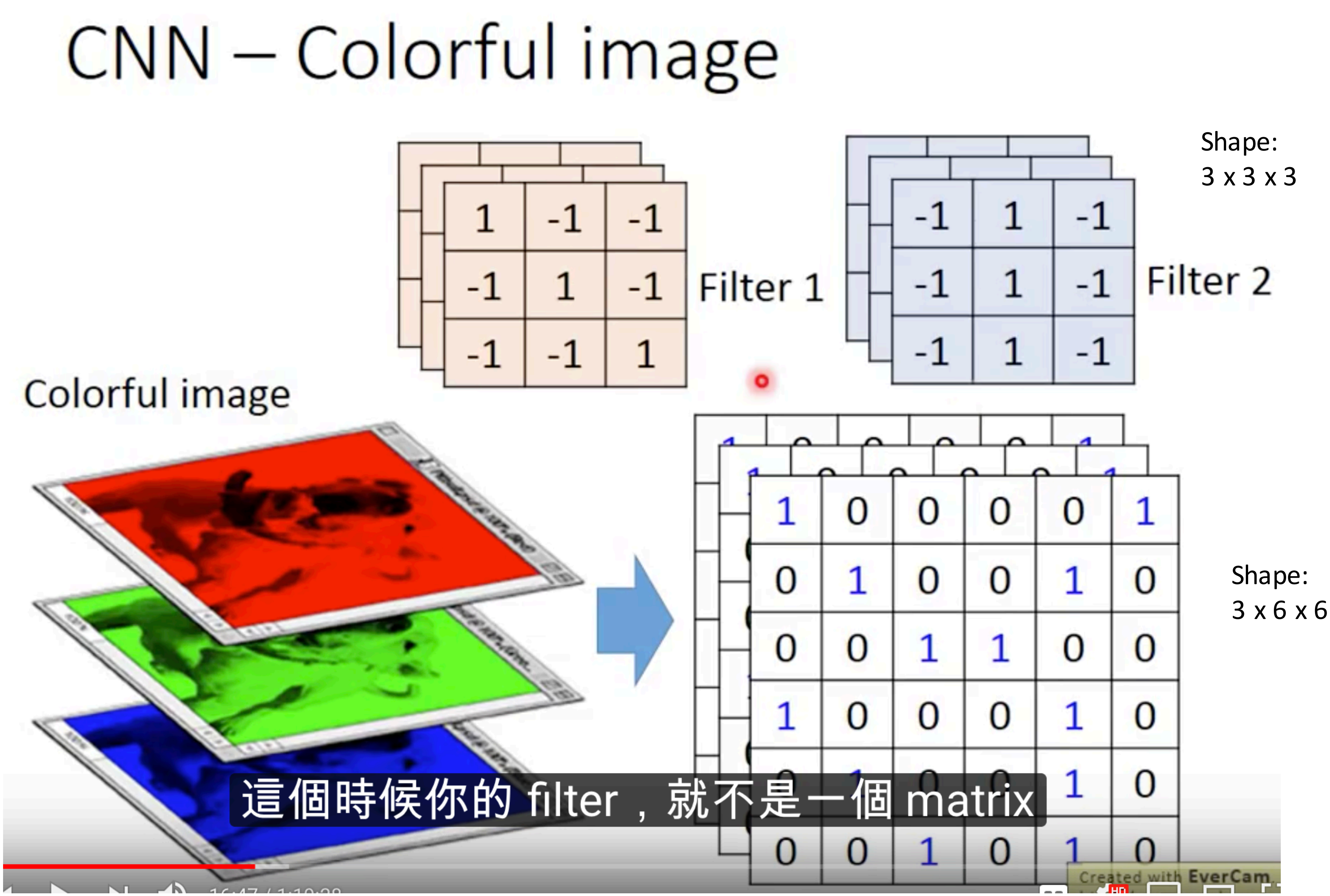
CNN – Colorful image

Colorful image

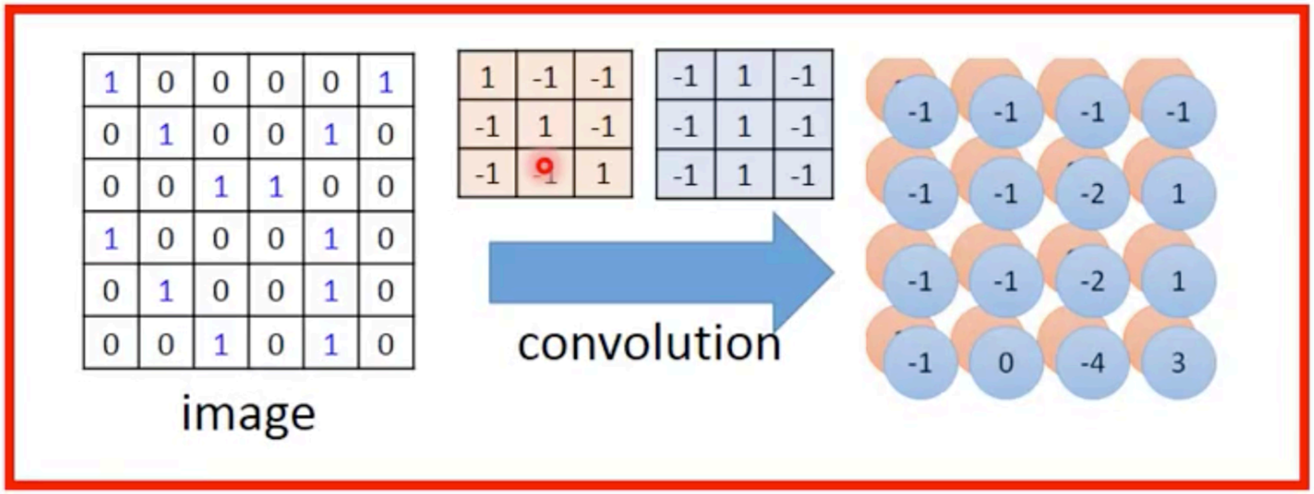


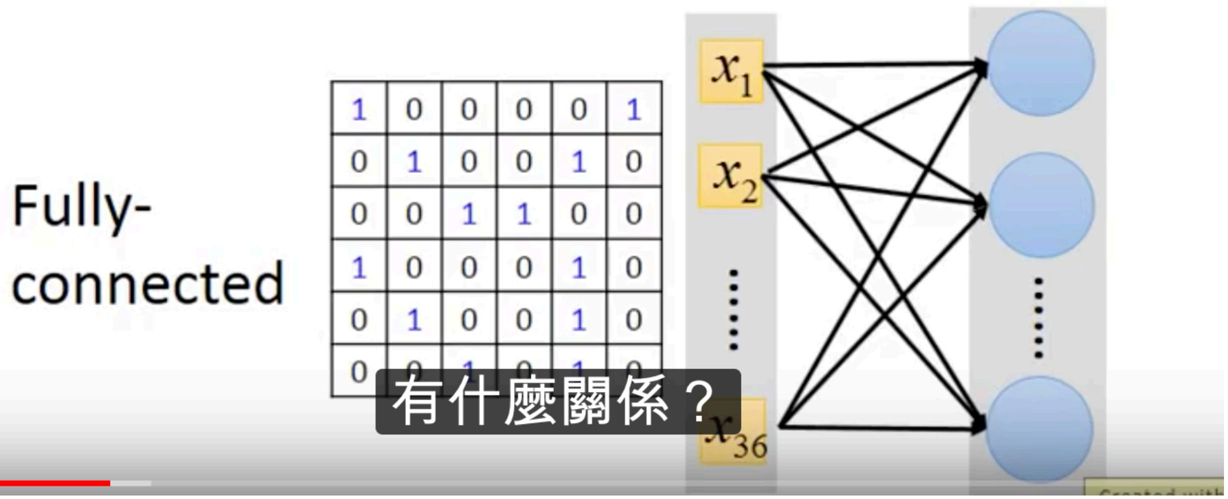
CNN – Colorful image




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
Shape:
3 x 3 x 3
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
Shape:
3 x 6 x 6
這個時候你的 filter，就不是一個 matrix
Created with EverCam

Convolution v.s. Fully Connected





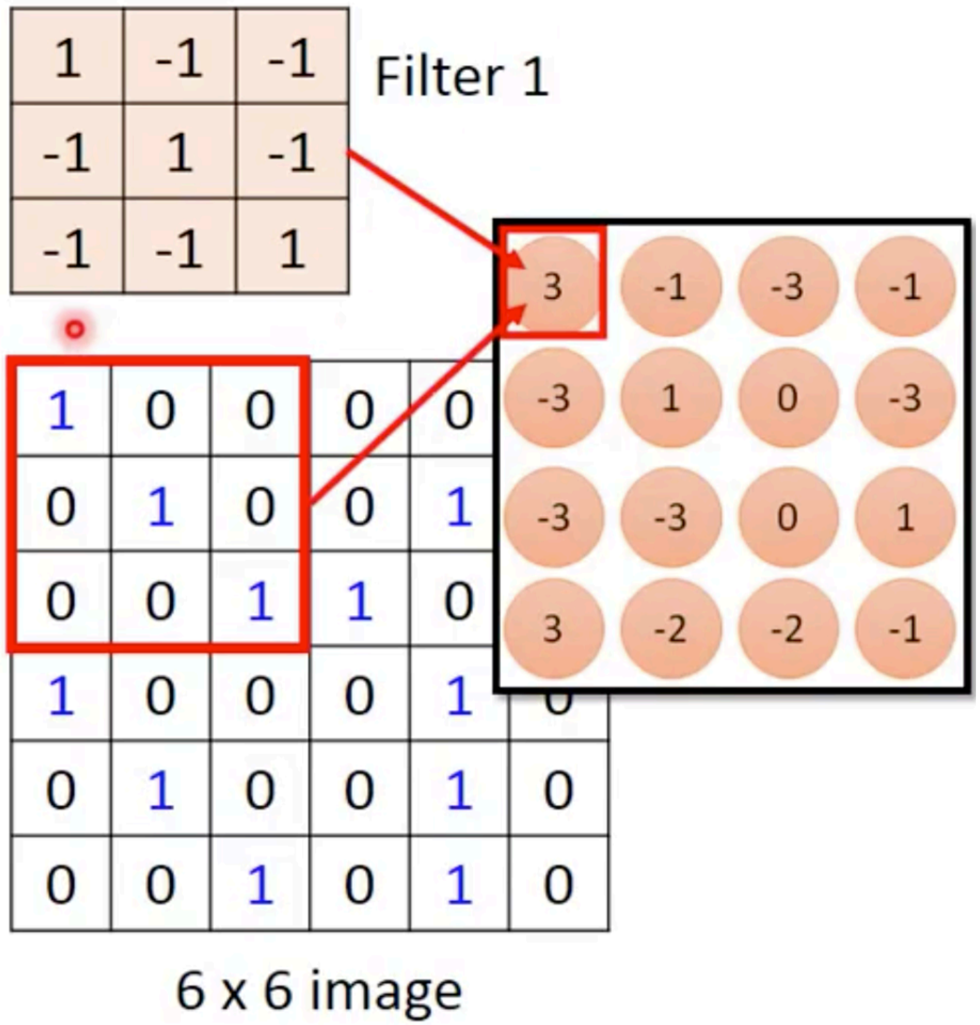
有什麼關係？

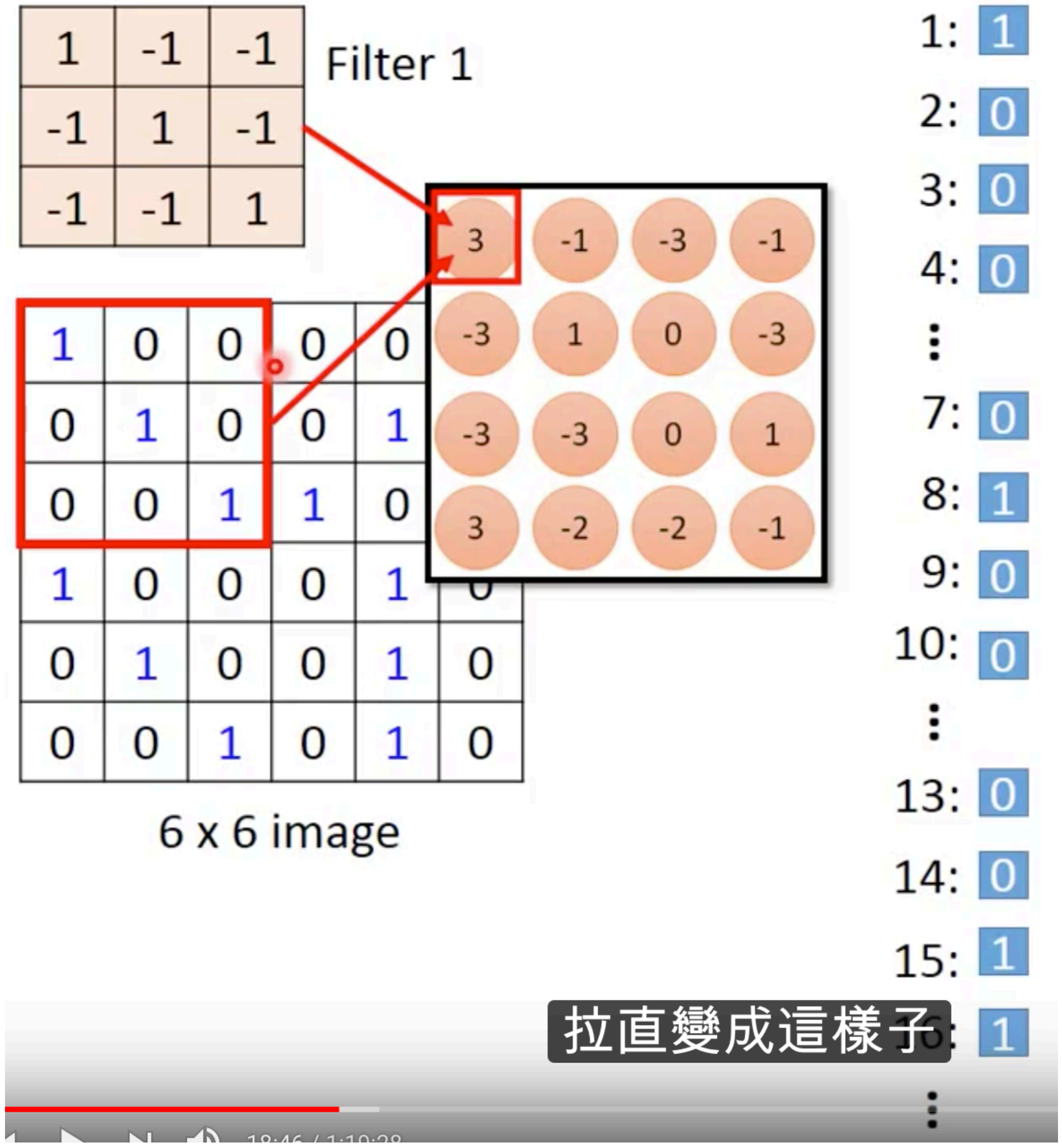
1	-1	-1
-1	1	-1
-1	-1	1

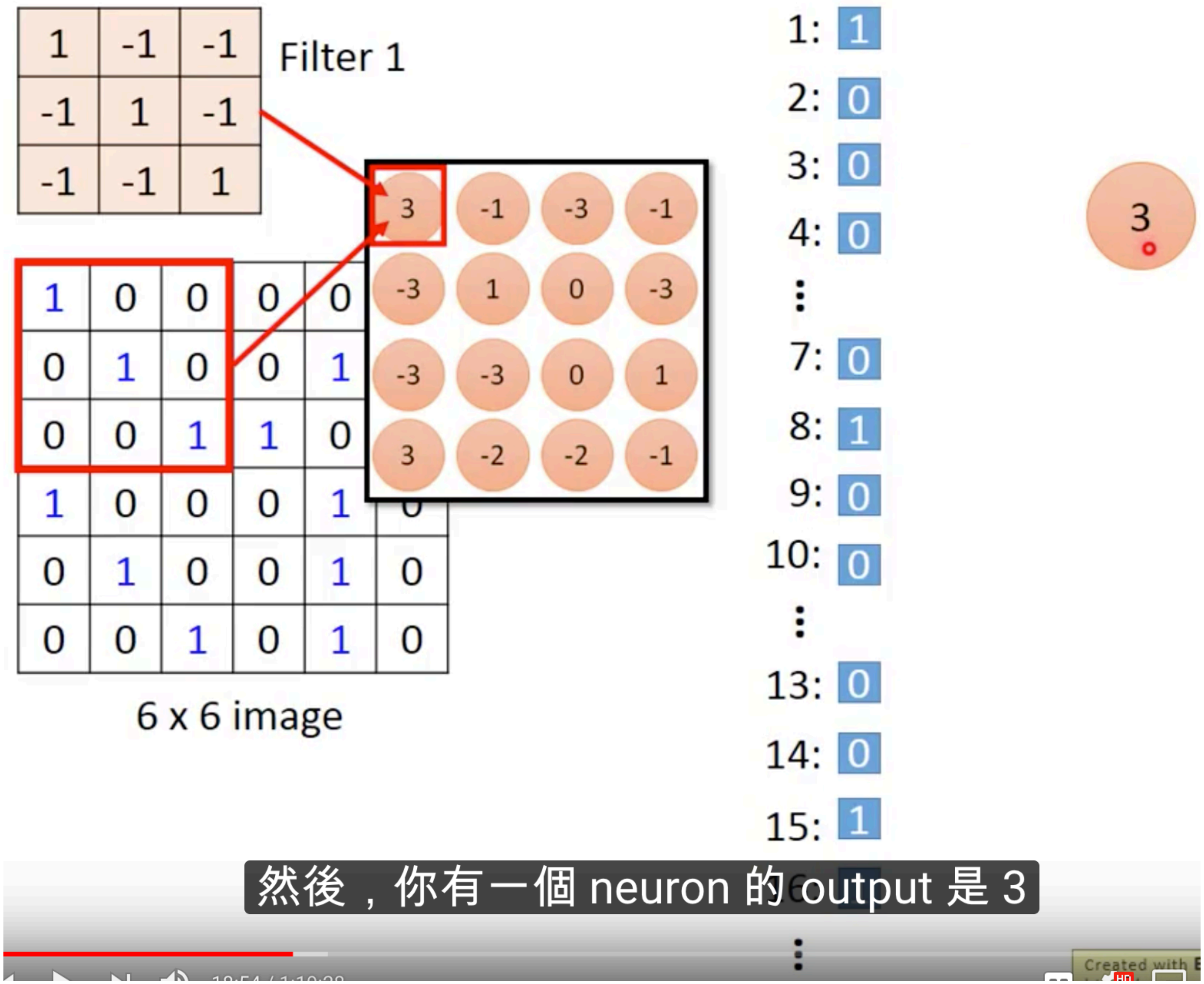
Filter 1

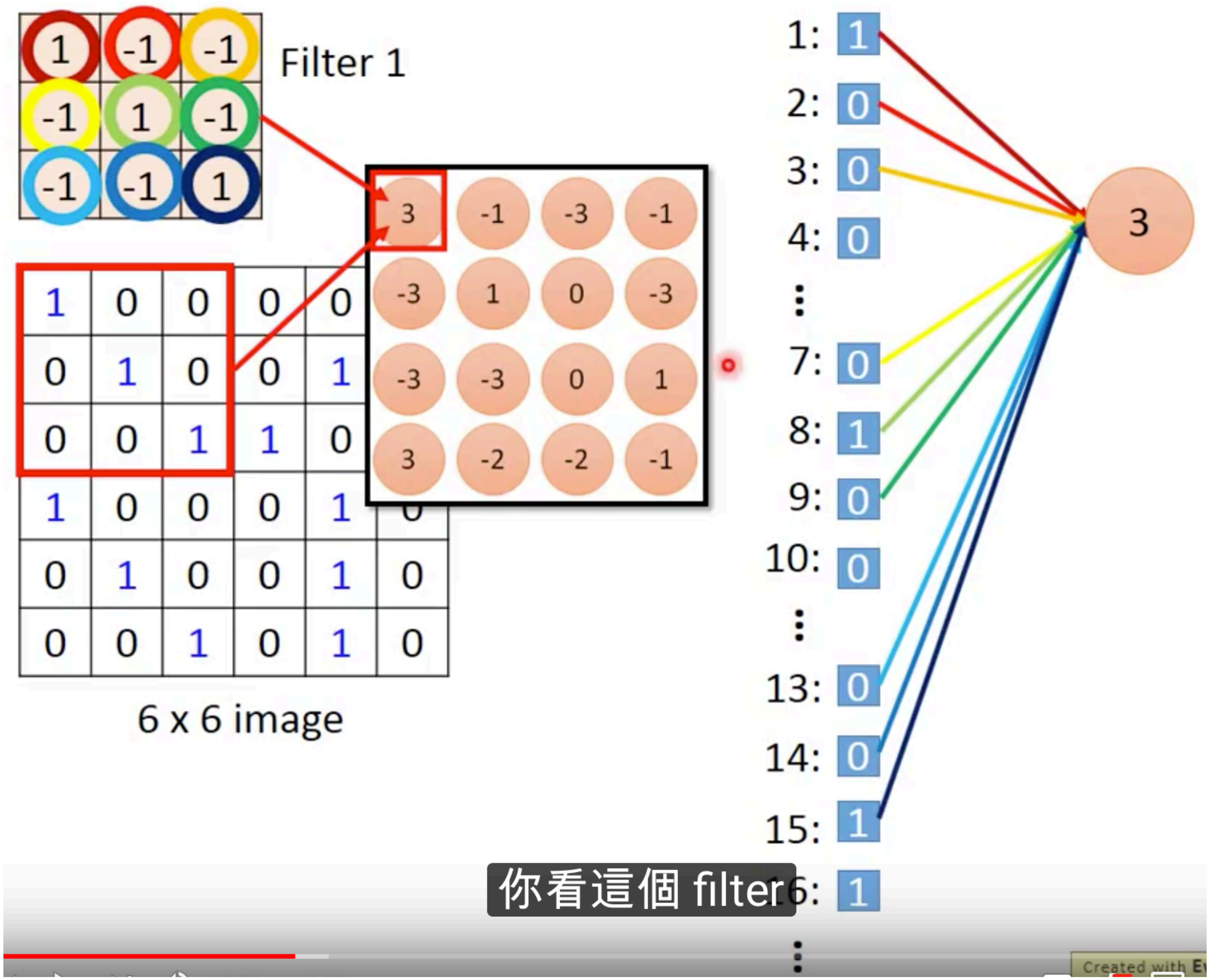
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

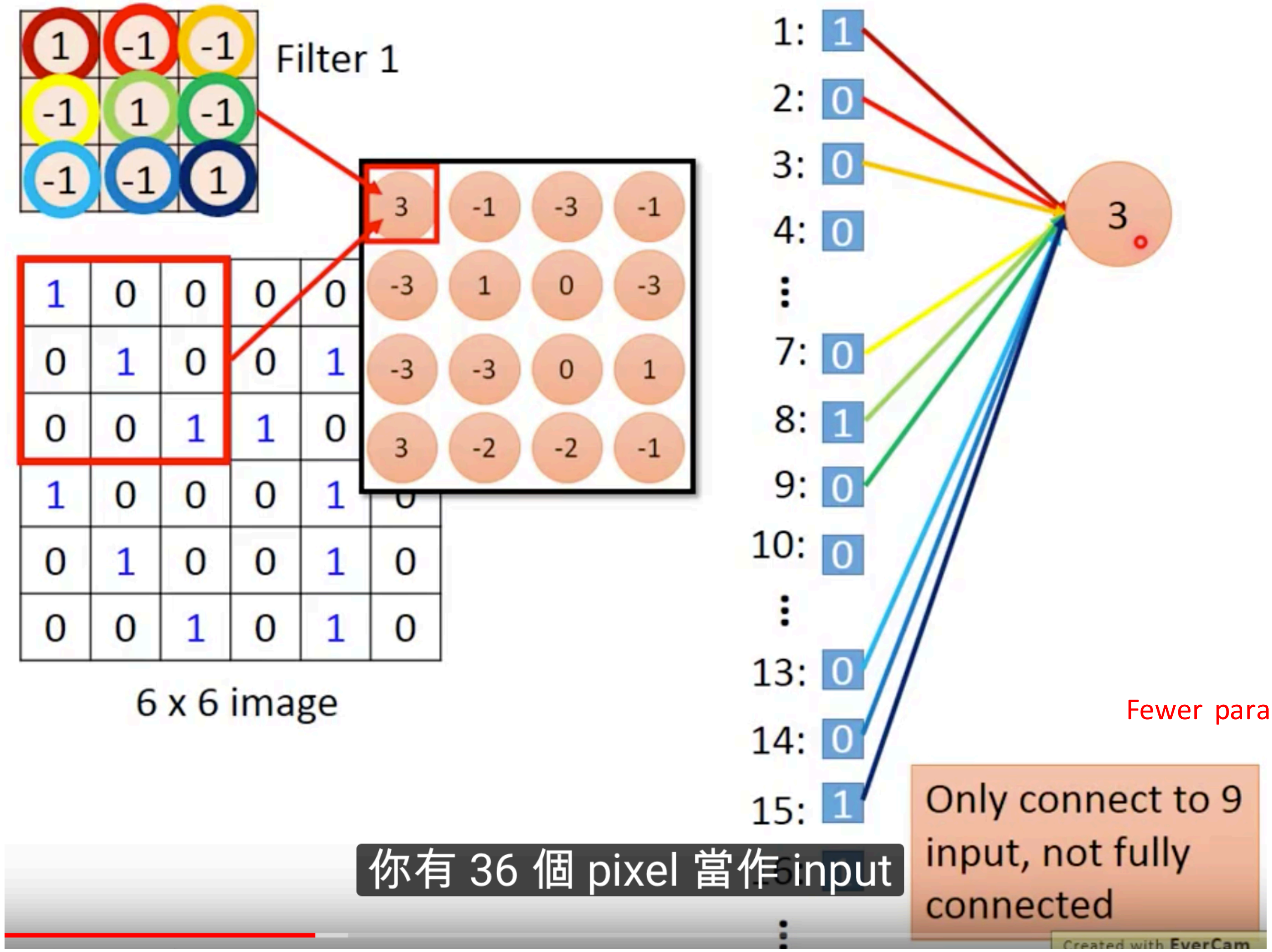
6 x 6 image


Filter 1
6 x 6 image

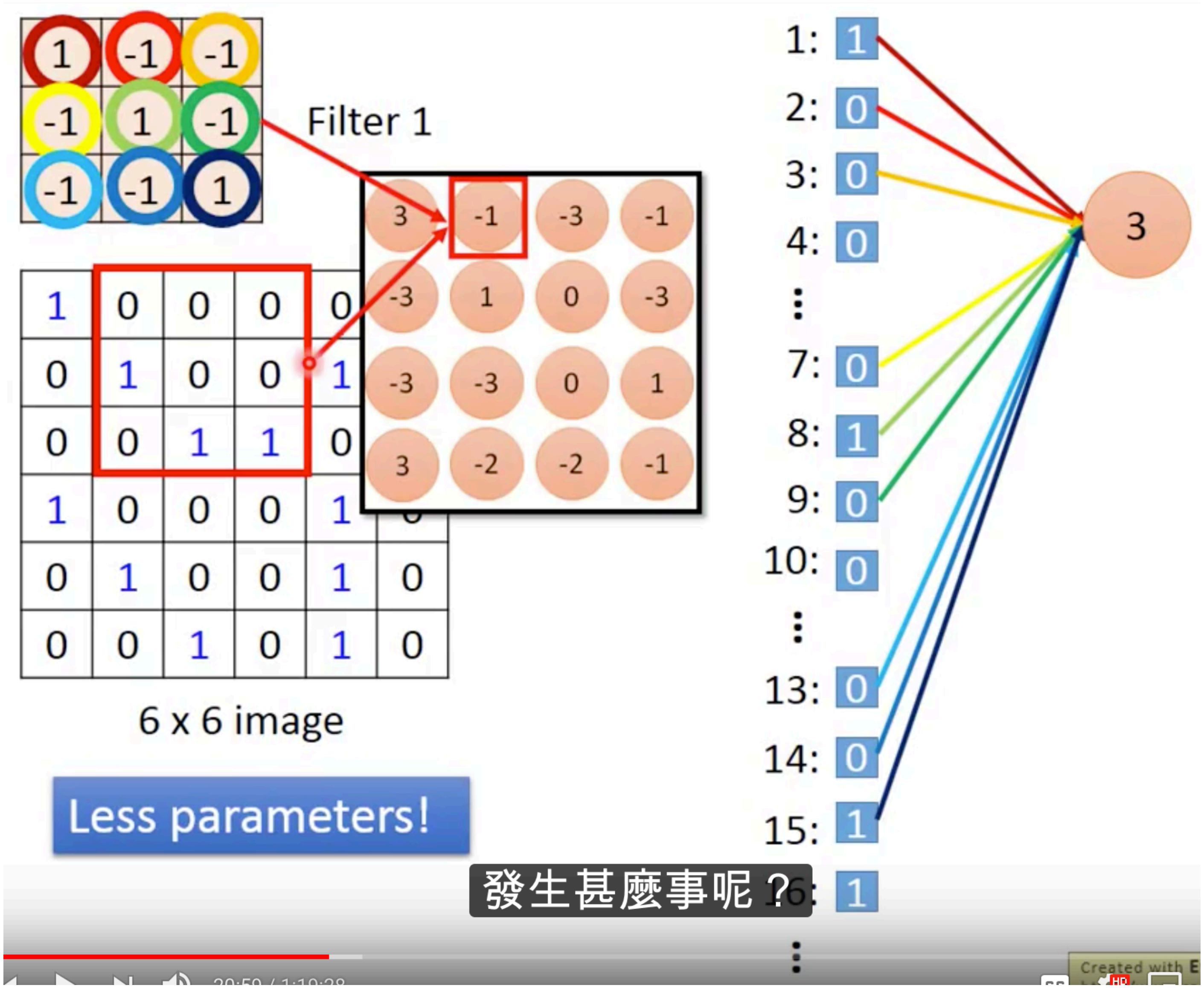

Filter 1
6 x 6 image
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
拉直變成這樣子


Filter 1
6 x 6 image
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
3
然後，你有一個 neuron 的 output 是 3

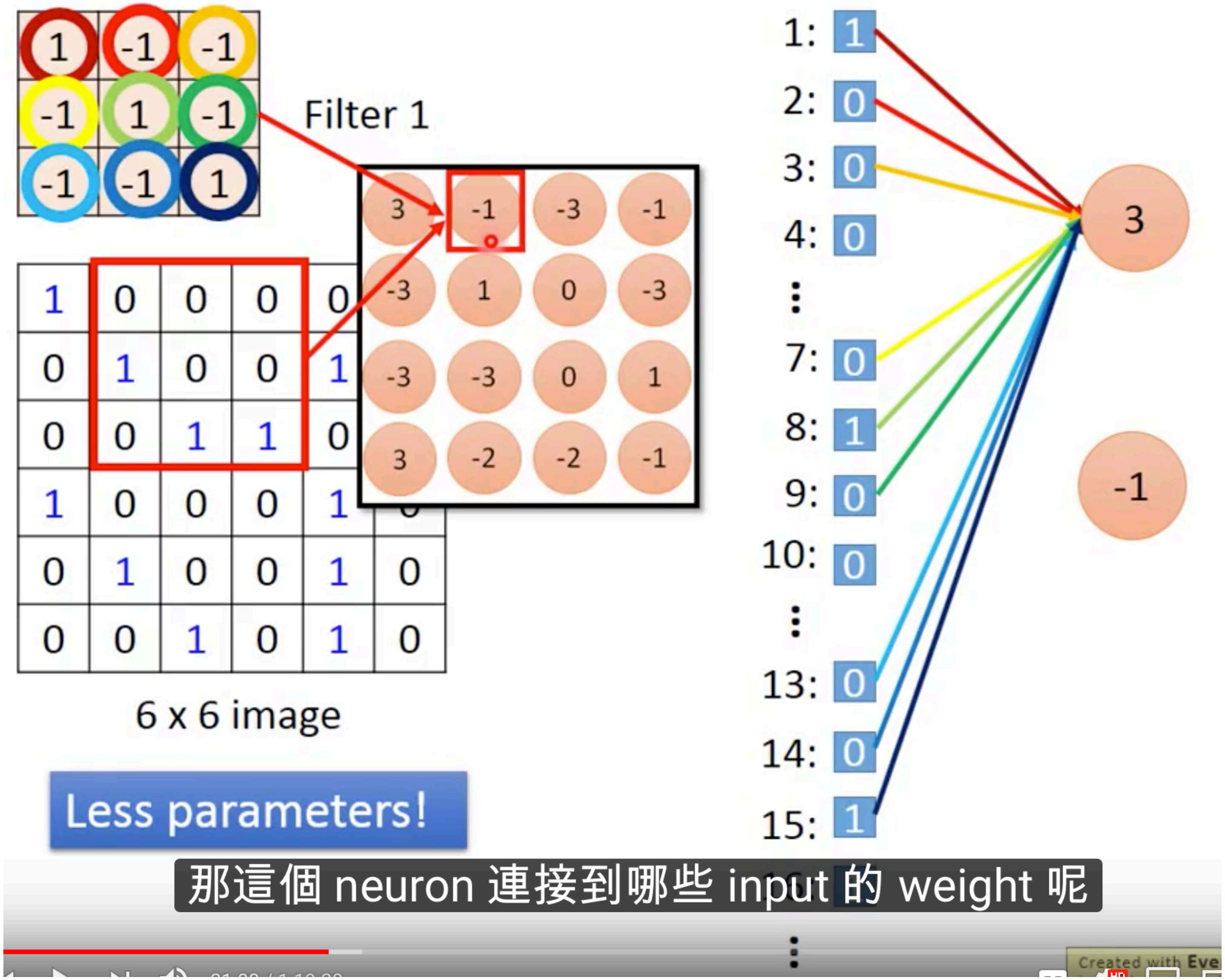

Filter 1
6 x 6 image
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
你看這個 filter



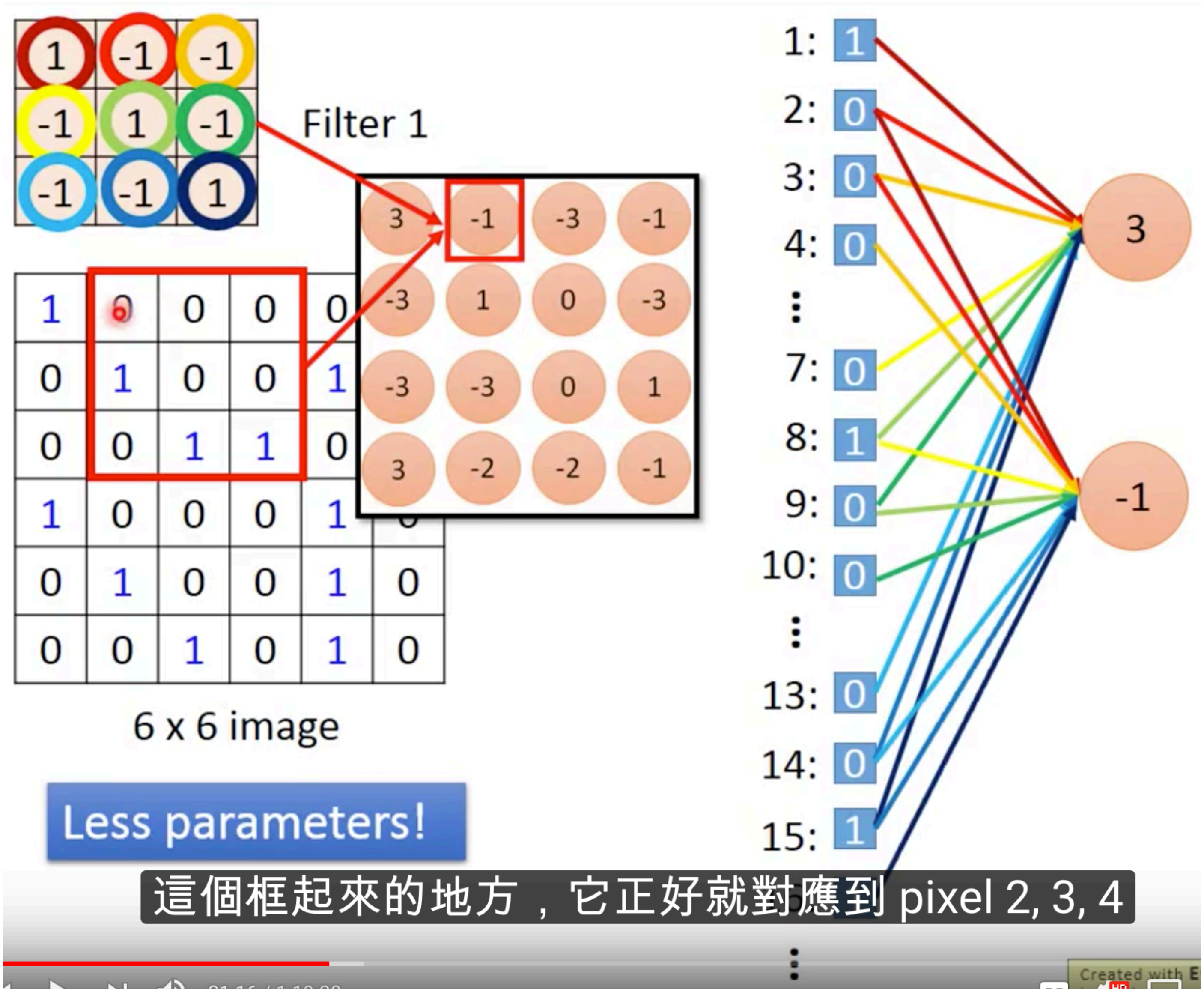
Fewer parameters!

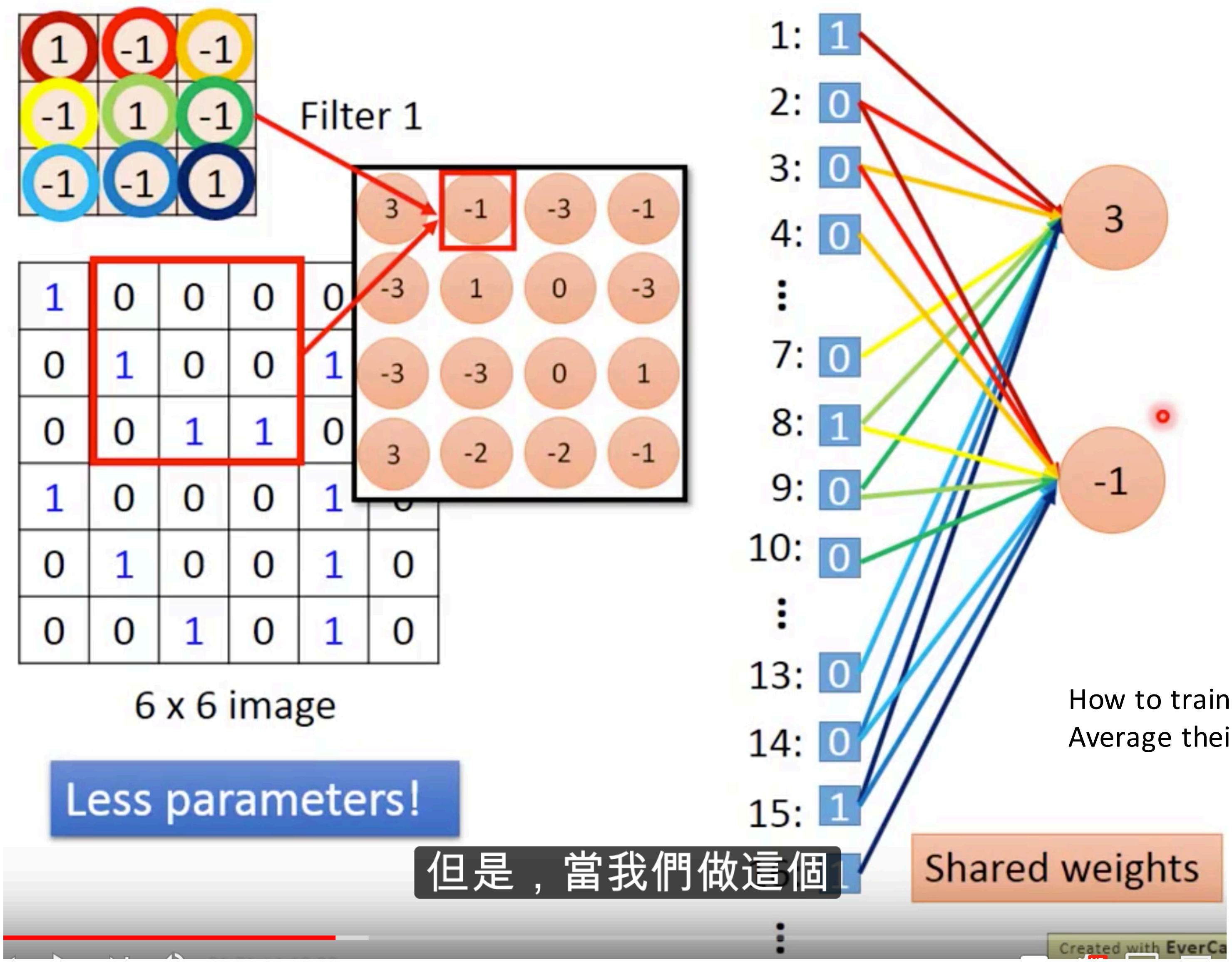


Filter 1
6 x 6 image
Less parameters!
1:
2:
3:
4:
7:
8:
9:
10:
13:
14:
15:
發生甚麼事呢？

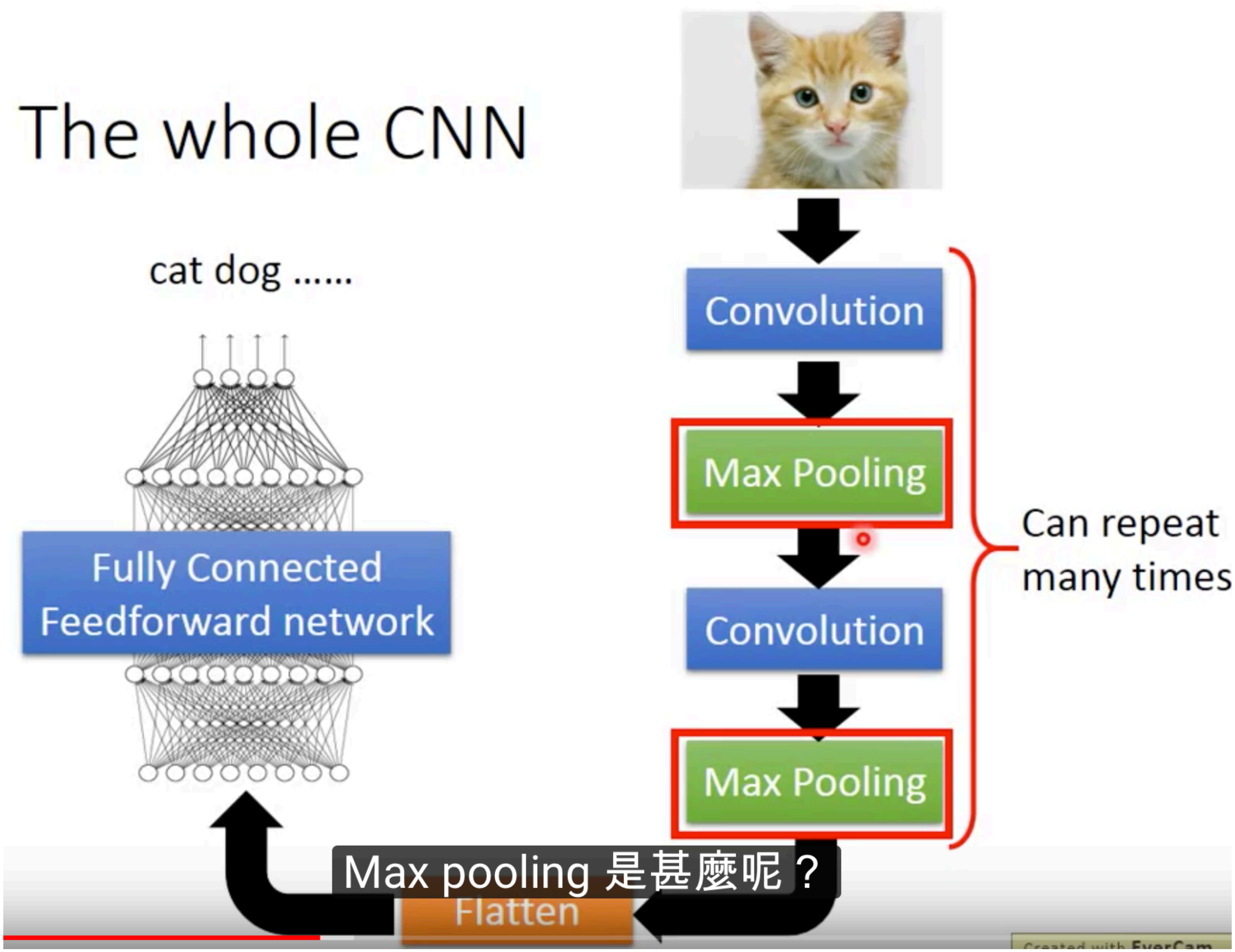


1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
Less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
3
-1
那這個 neuron 連接到哪些 input 的 weight 呢


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
3
-1
這個框起來的地方，它正好就對應到 pixel 2, 3, 4



How to train shared weight?
Average their gradients.


The whole CNN
cat dog ......
Fully Connected Feedforward network
Convolution
Max Pooling
Convolution
Max Pooling
Can repeat many times
Flatten
Max pooling 是甚麼呢？

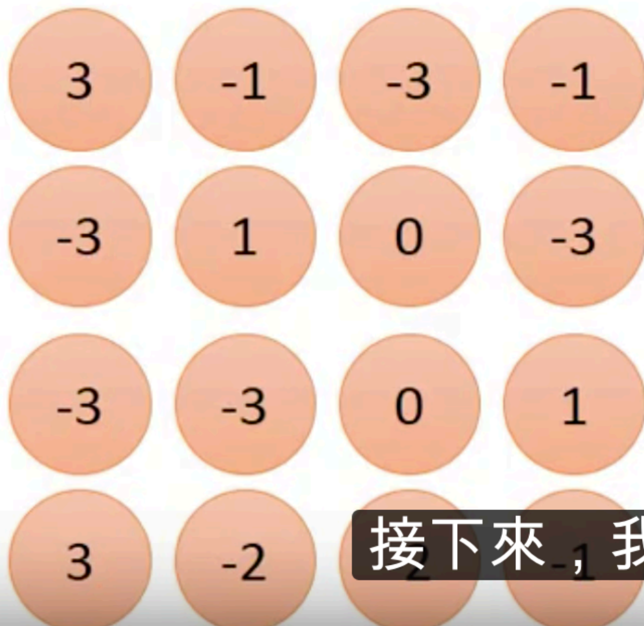
CNN – Max Pooling

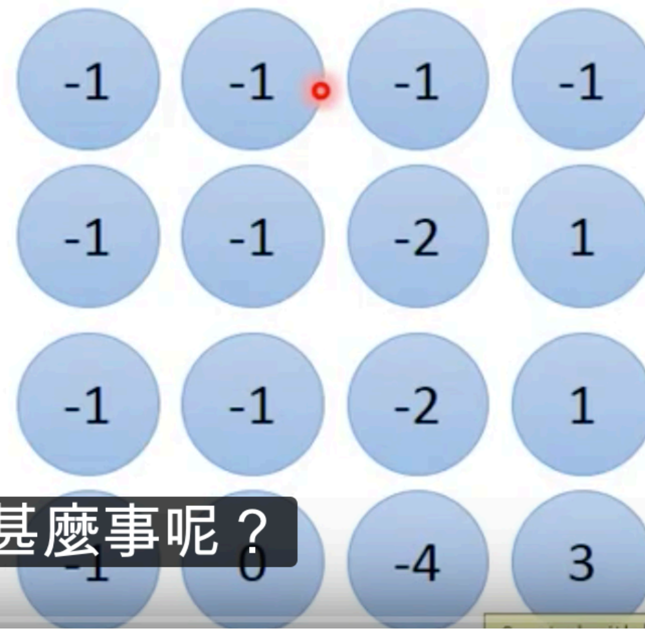
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Filter 2





接下來，我們做甚麼事呢？

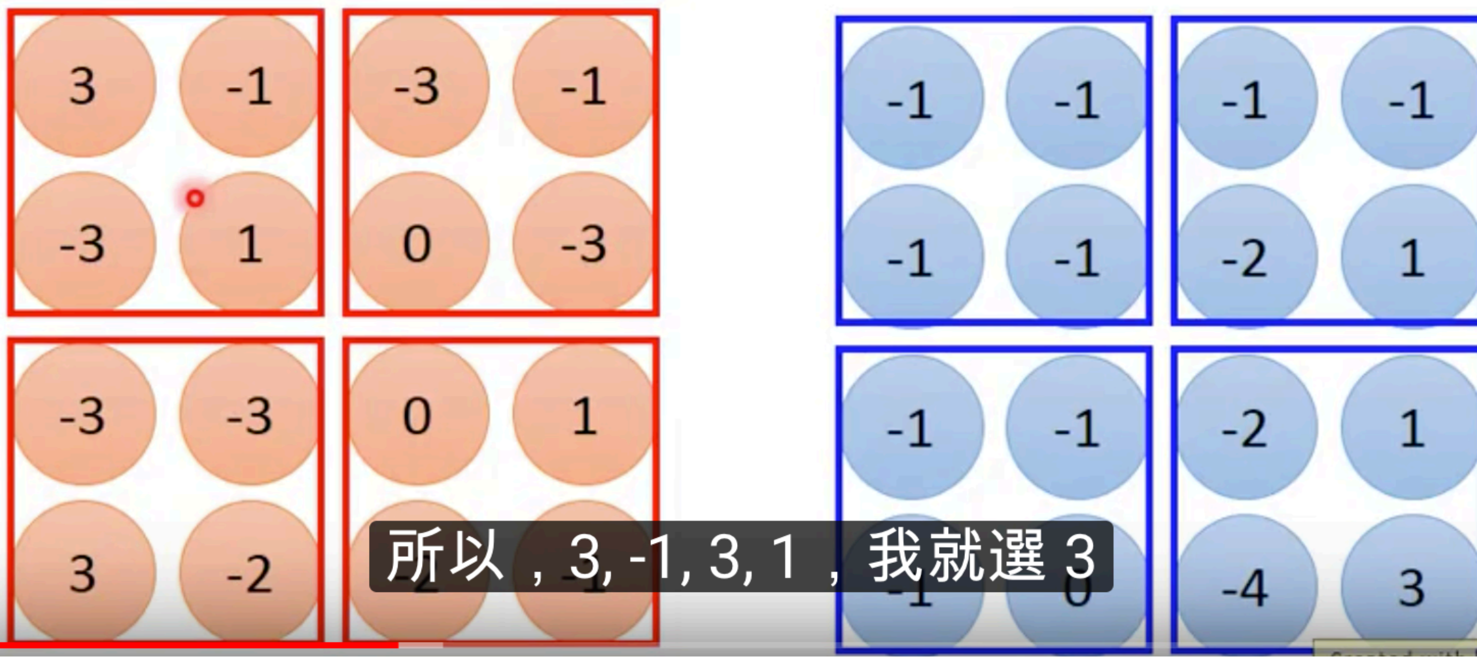
CNN – Max Pooling

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

Filter 1

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

Filter 2



所以，3, -1, 3, 1，我就選 3

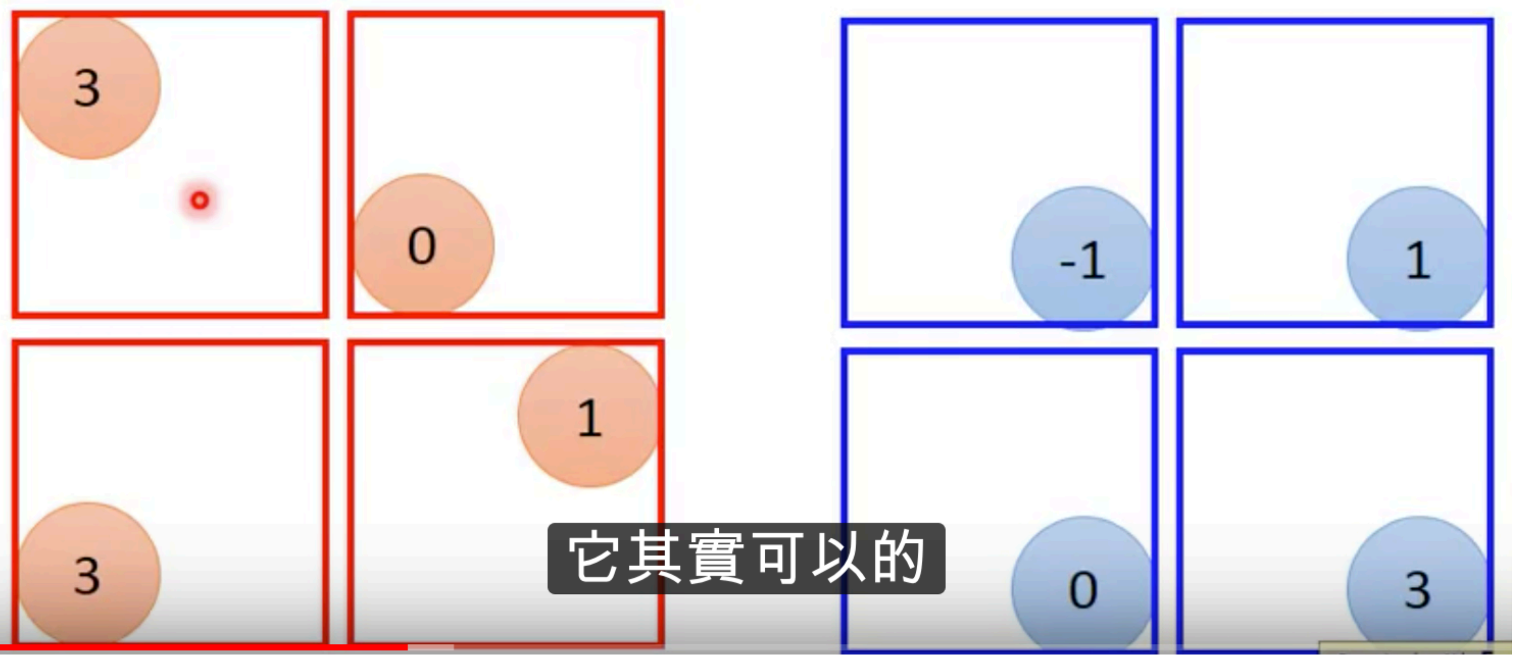
CNN – Max Pooling

Filter 1

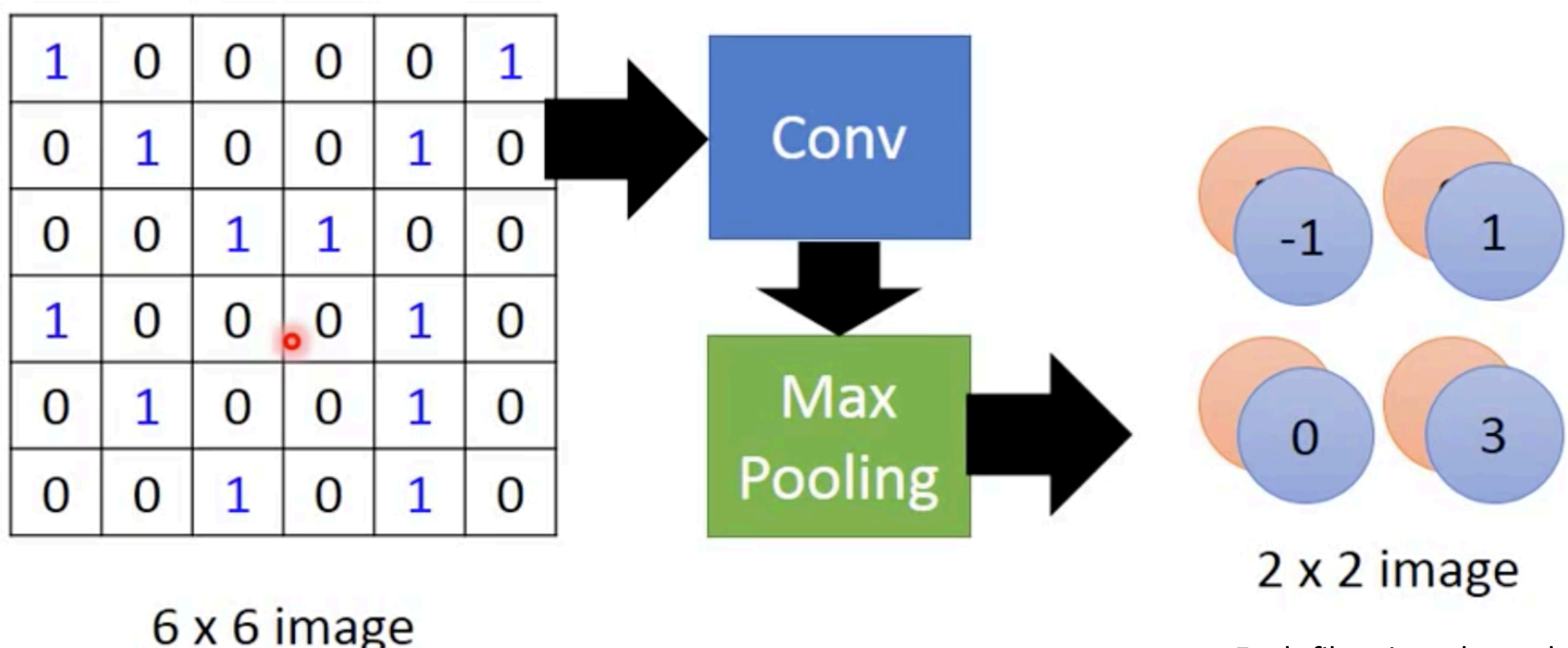
1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

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

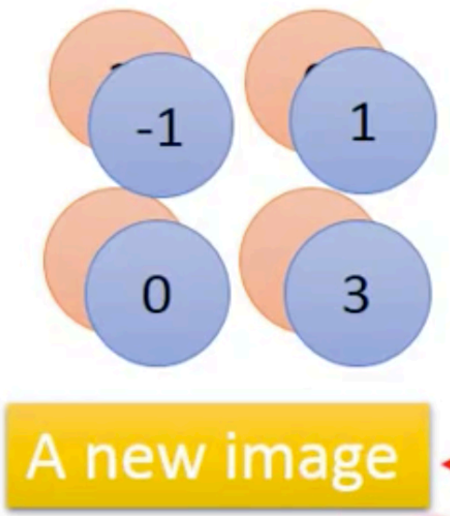


CNN – Max Pooling

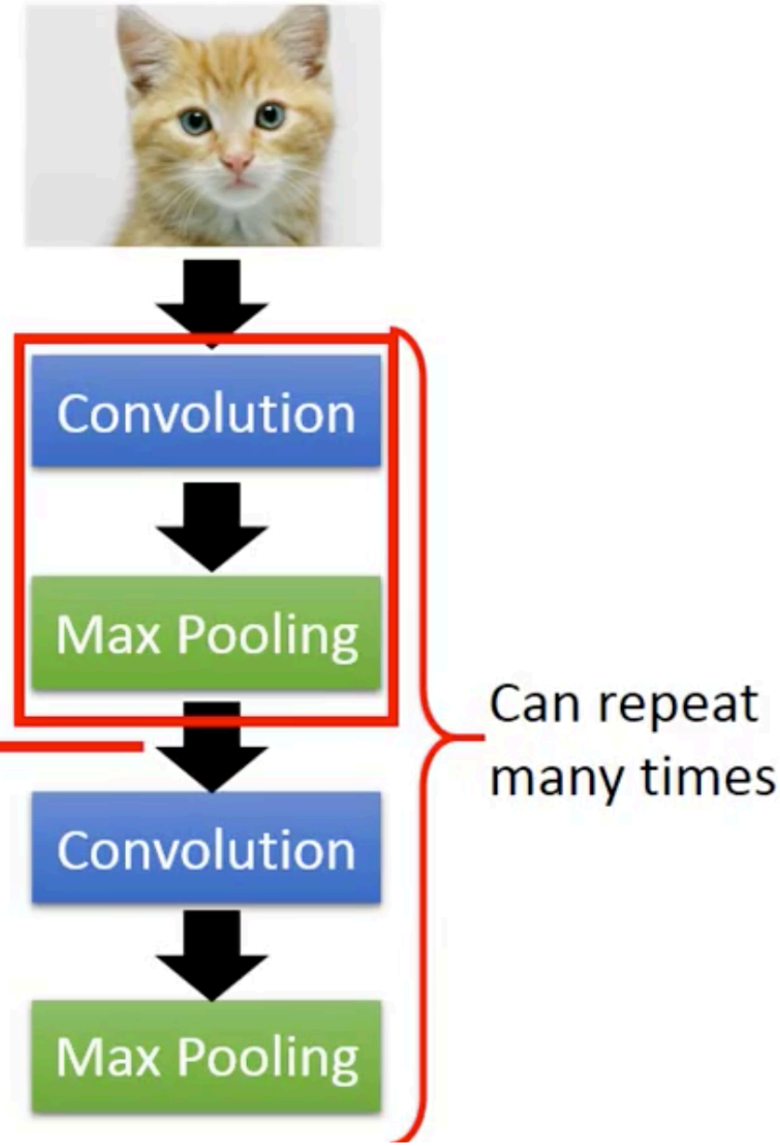


Each filter is a channel

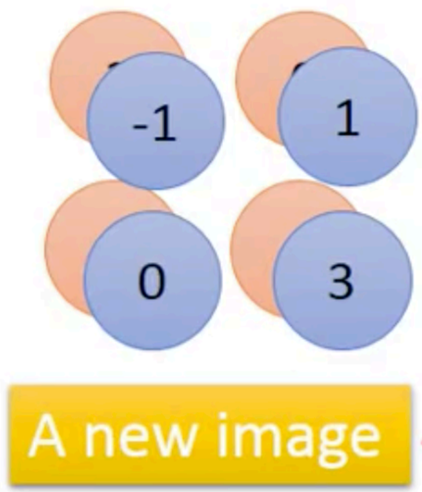
The whole CNN

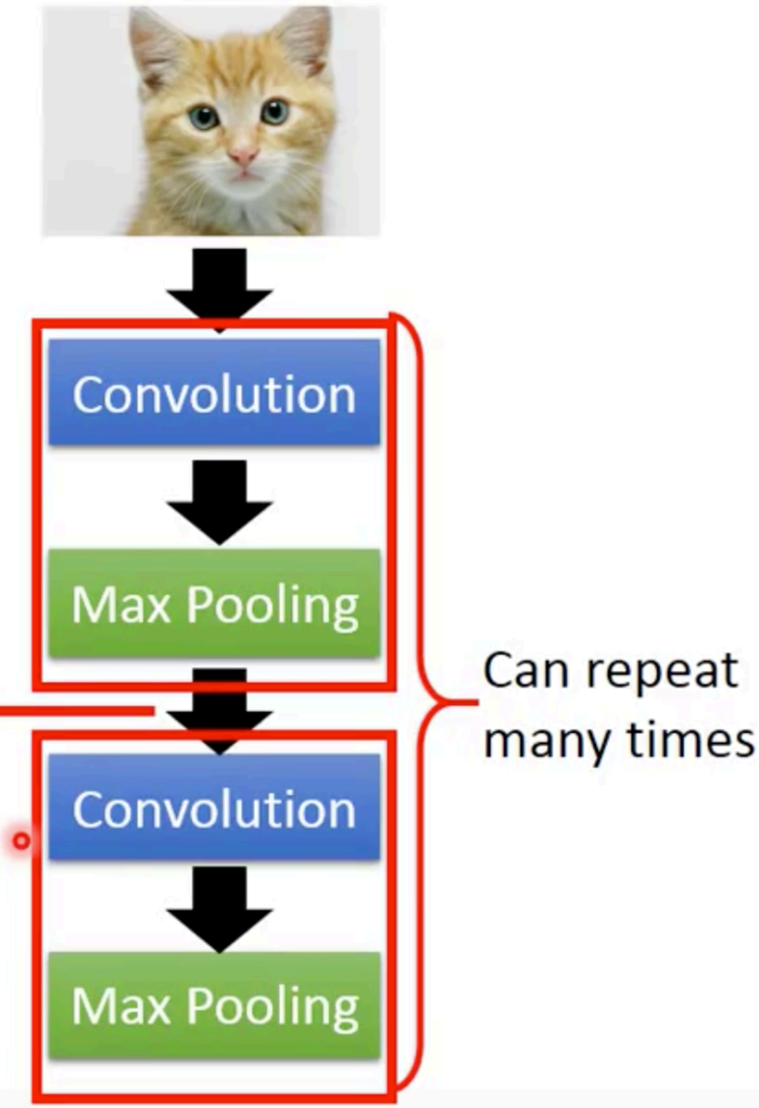


Smaller than the original image



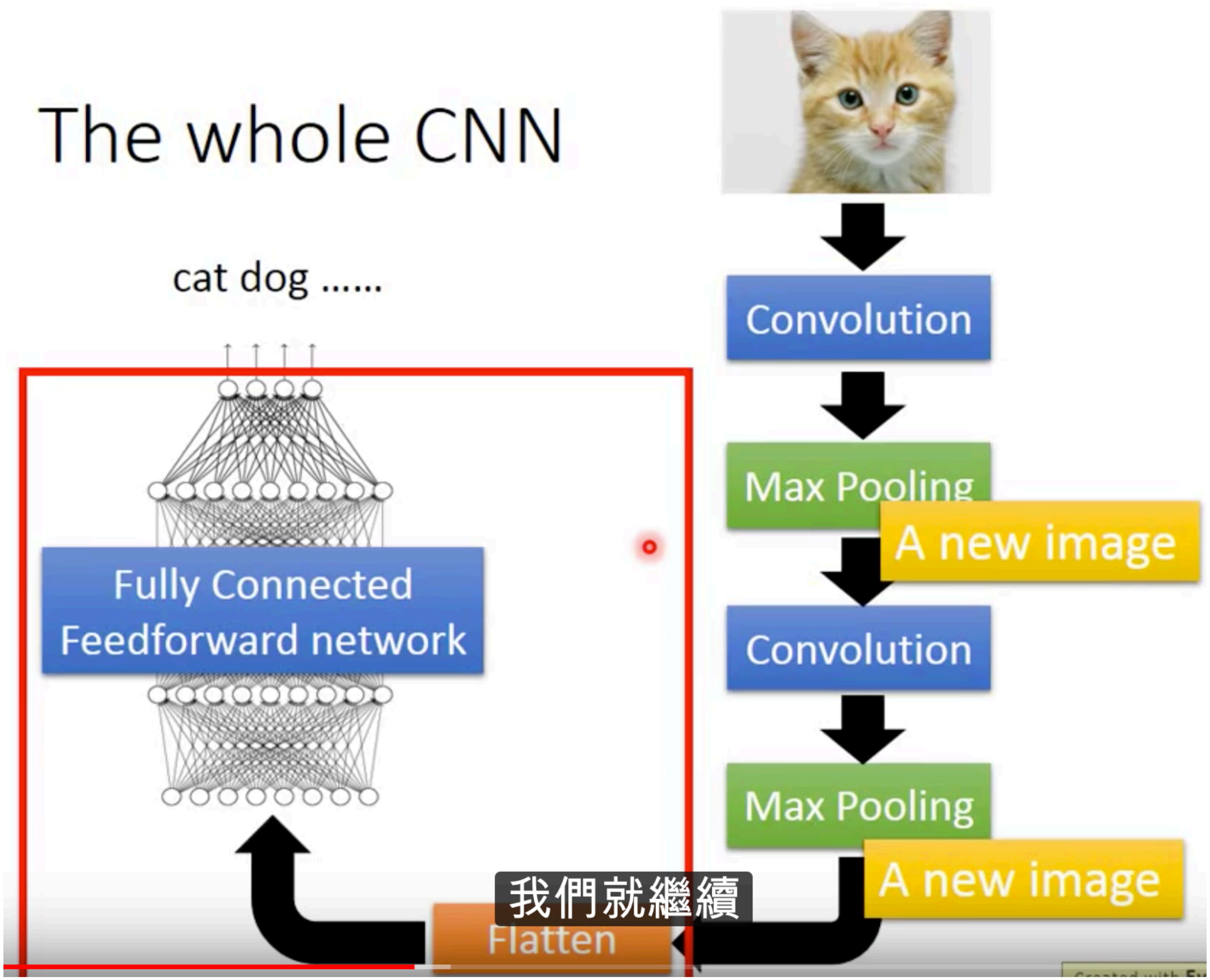
The whole CNN



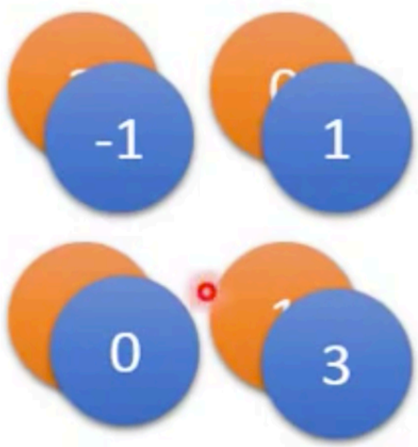


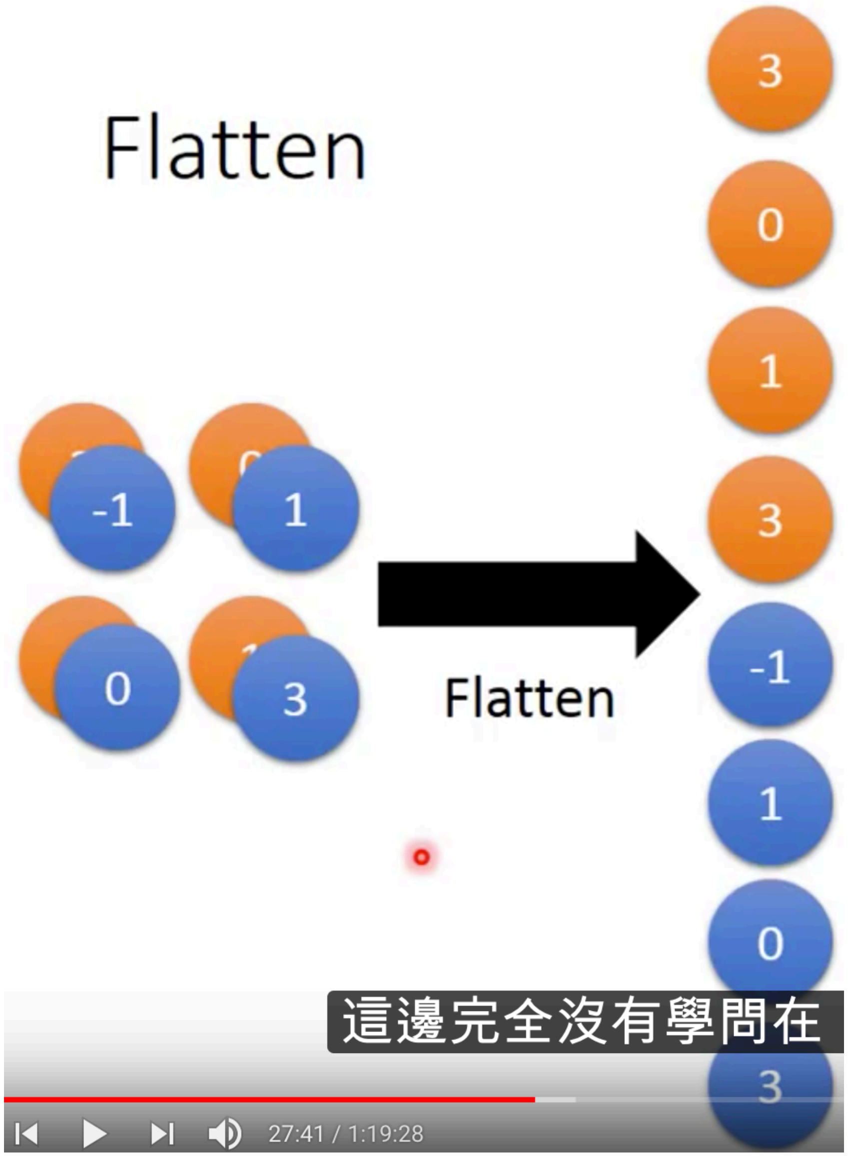
Smaller than the original image

The number of the channel is the number of filters

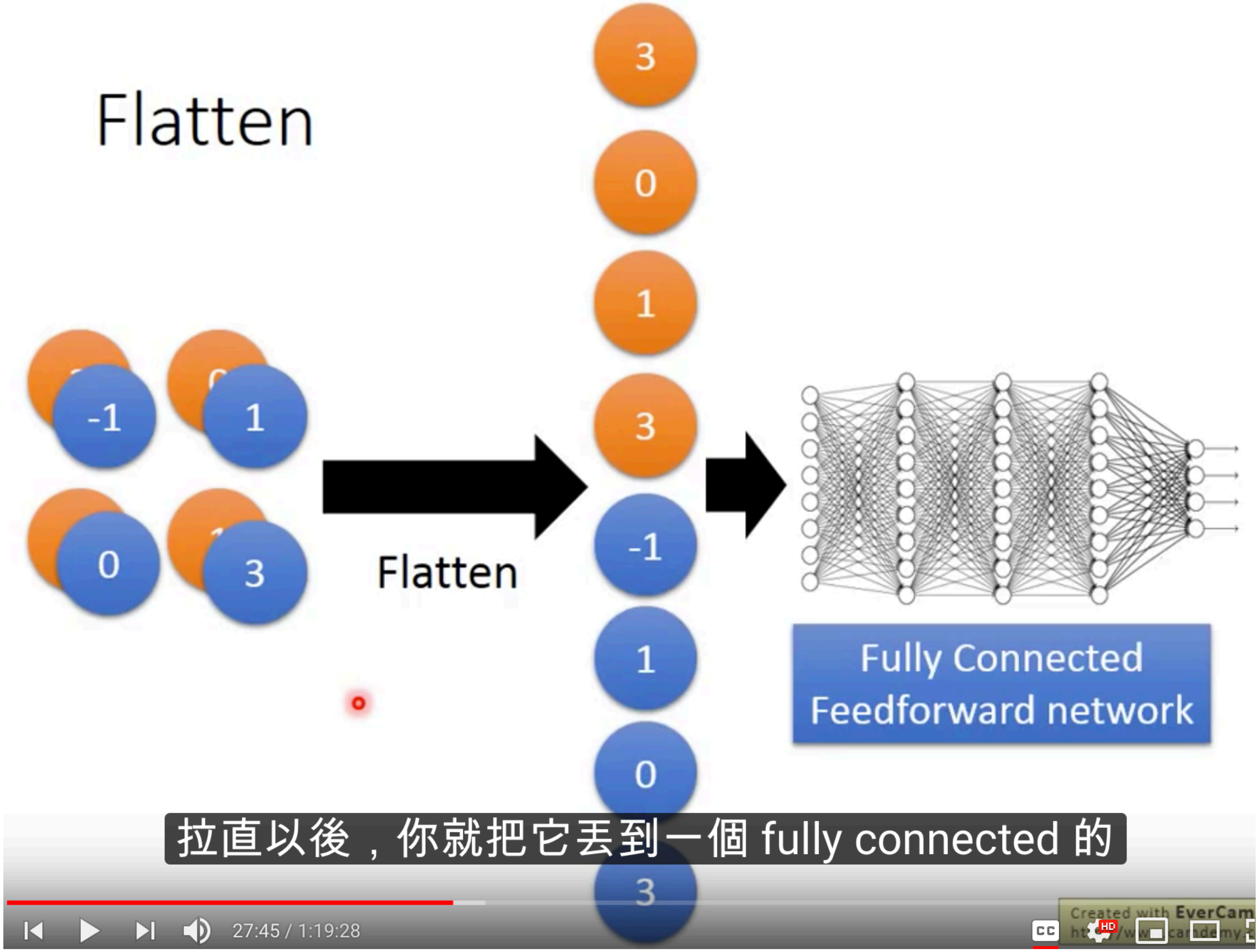

The whole CNN
cat dog ......
Fully Connected
Feedforward network
Convolution
Max Pooling
A new image
Convolution
Max Pooling
A new image
Flatten
我們就繼續

Flatten





Flatten
-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
這邊完全沒有學問在
27:41 / 1:19:28


Flatten
-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network
拉直以後，你就把它丟到一個 fully connected 的
27:45 / 1:19:28
Created with EverCam

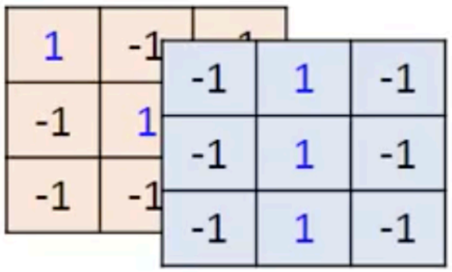
CNN in Keras

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)***

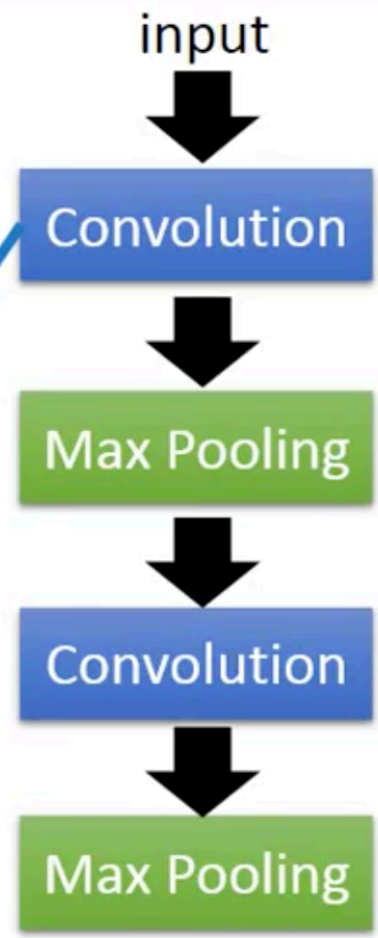
```
model2.add( Convolution2D( 25,3,3,
            input_shape=(1,28,28) ) )
```

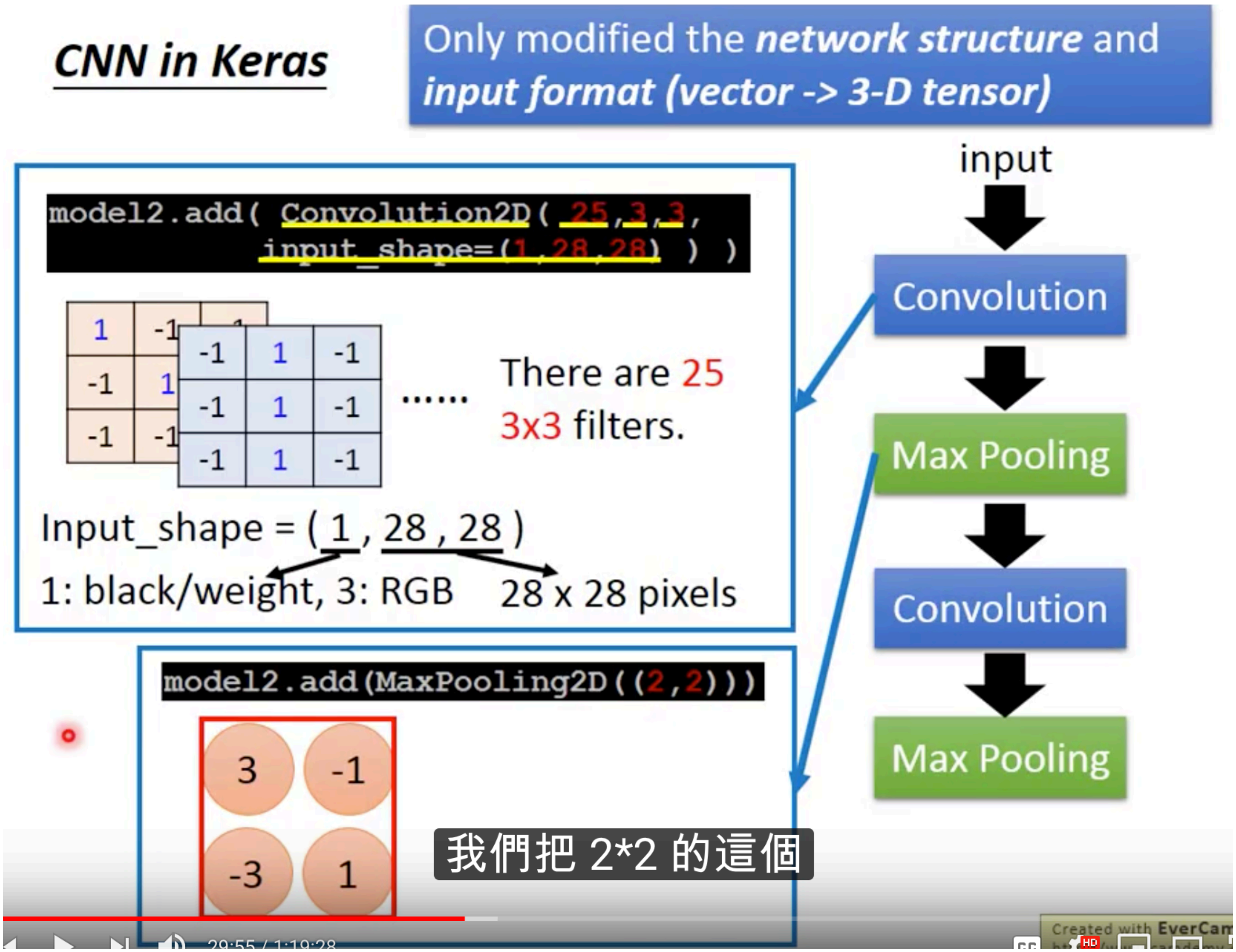


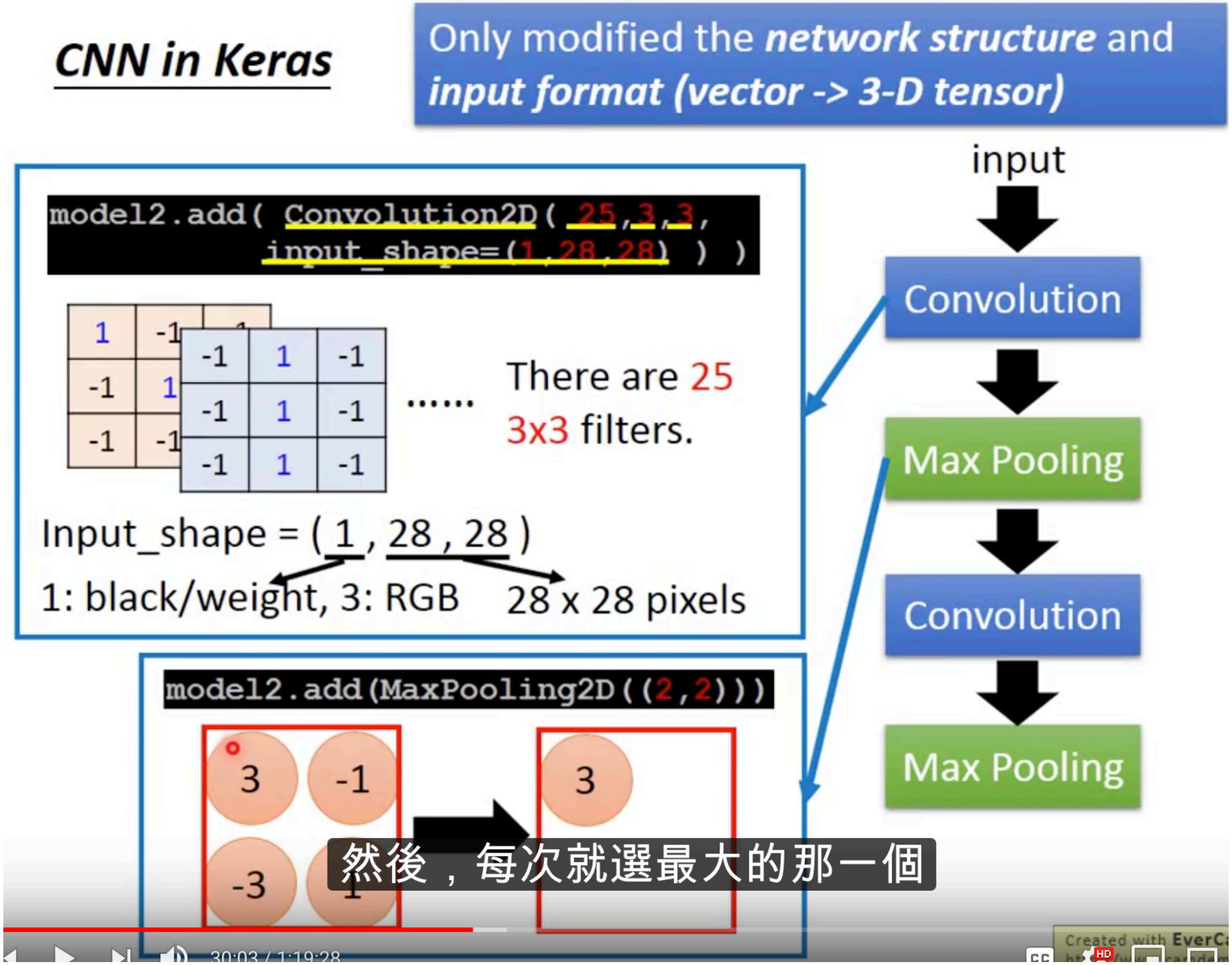
...... There are 25 3x3 filters.

Input_shape = (1 , 28 , 28)

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




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
我們把 2*2 的這個
29:55 / 1:19:28
Created with EverCam


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)))
1 -1
-1 1
-1 -1
-1 1 -1
-1 1 -1
-1 1 -1
......
There are 25 3x3 filters.
Input_shape = (1 , 28 , 28)
1: black/weight, 3: RGB
28 x 28 pixels
model2.add(MaxPooling2D((2,2)))
3 -1
-3 1
3
input
Convolution
Max Pooling
Convolution
Max Pooling
然後，每次就選最大的那一個

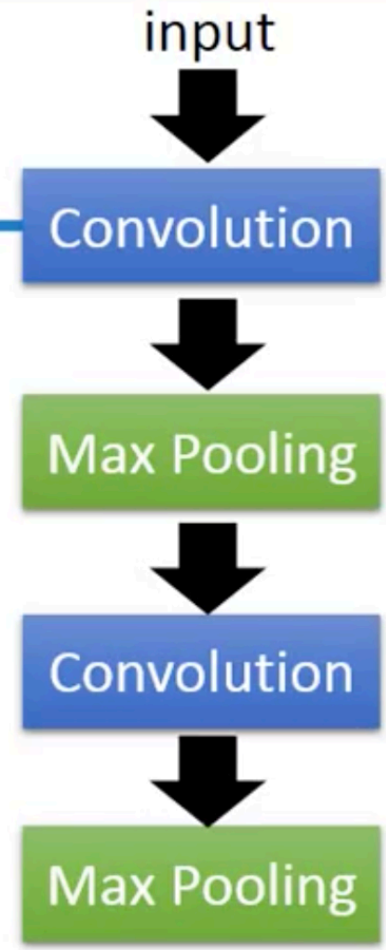
CNN in Keras

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=(1,28,28) ) )
```

25 x 26 x 26



CNN in Keras

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

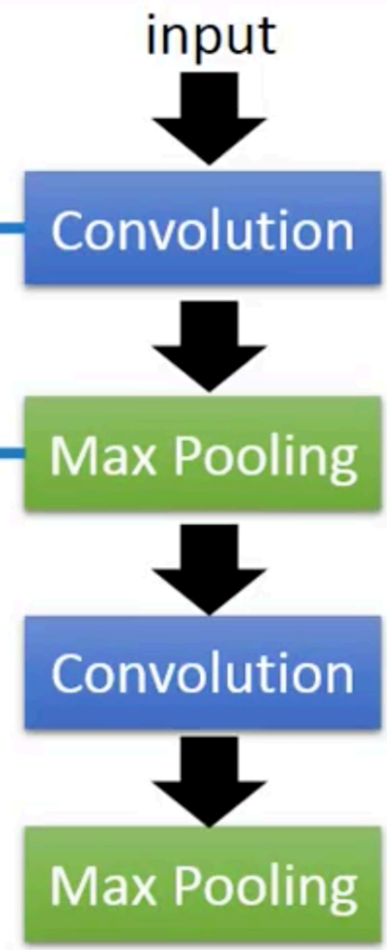
1 x 28 x 28

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

25 x 26 x 26

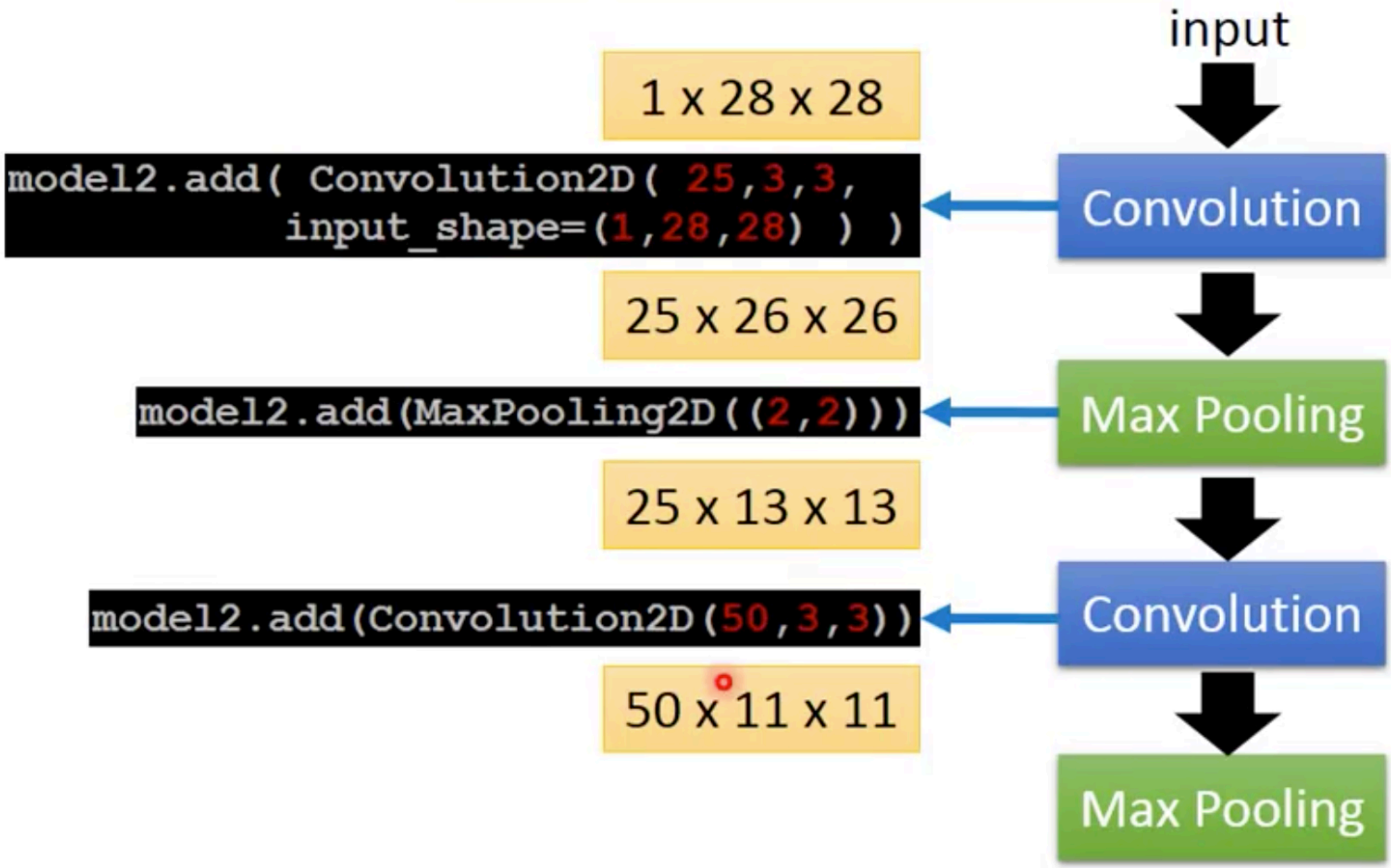
```
model2.add(MaxPooling2D((2,2)))
```

25 x 13 x 13



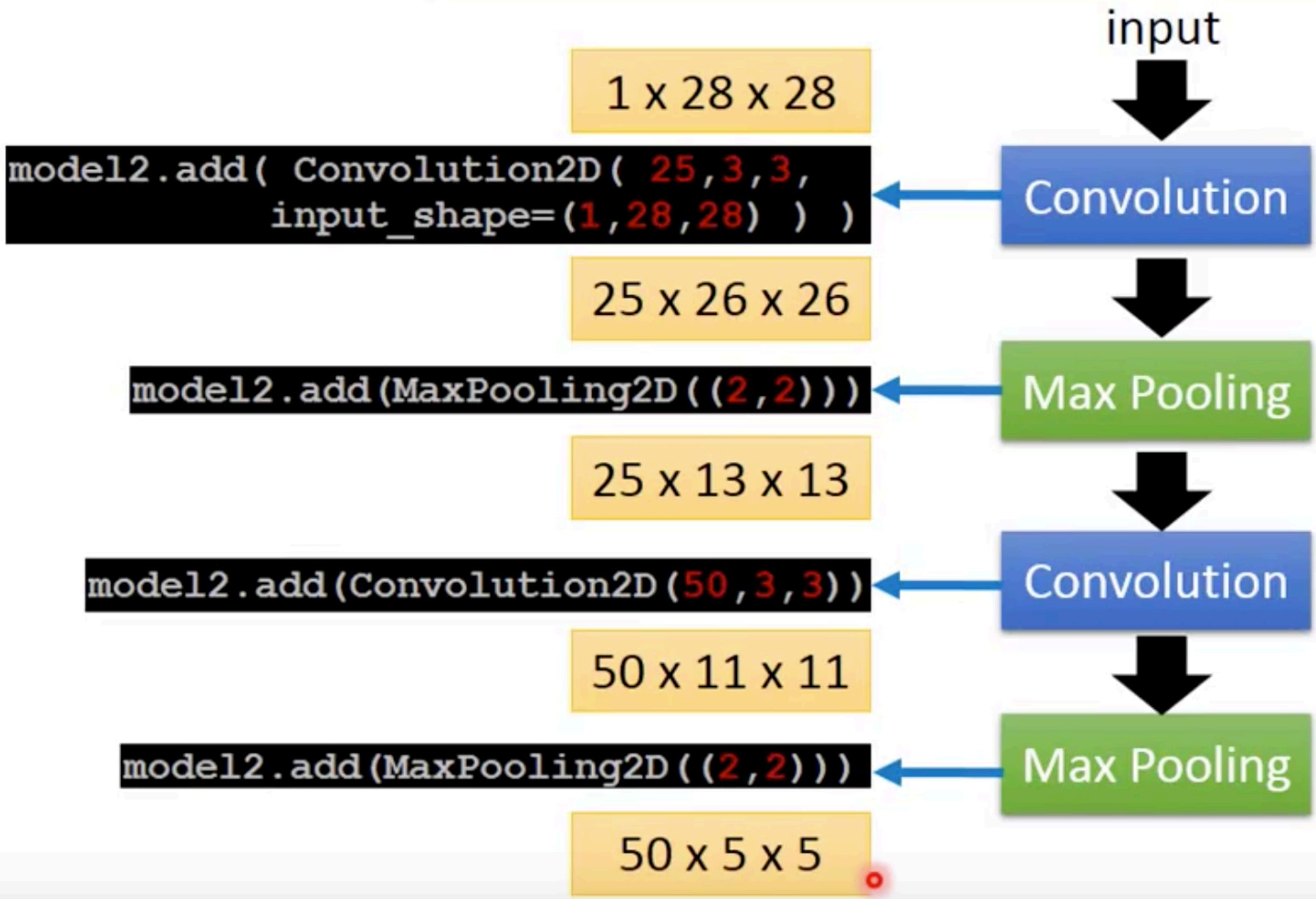
CNN in Keras

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



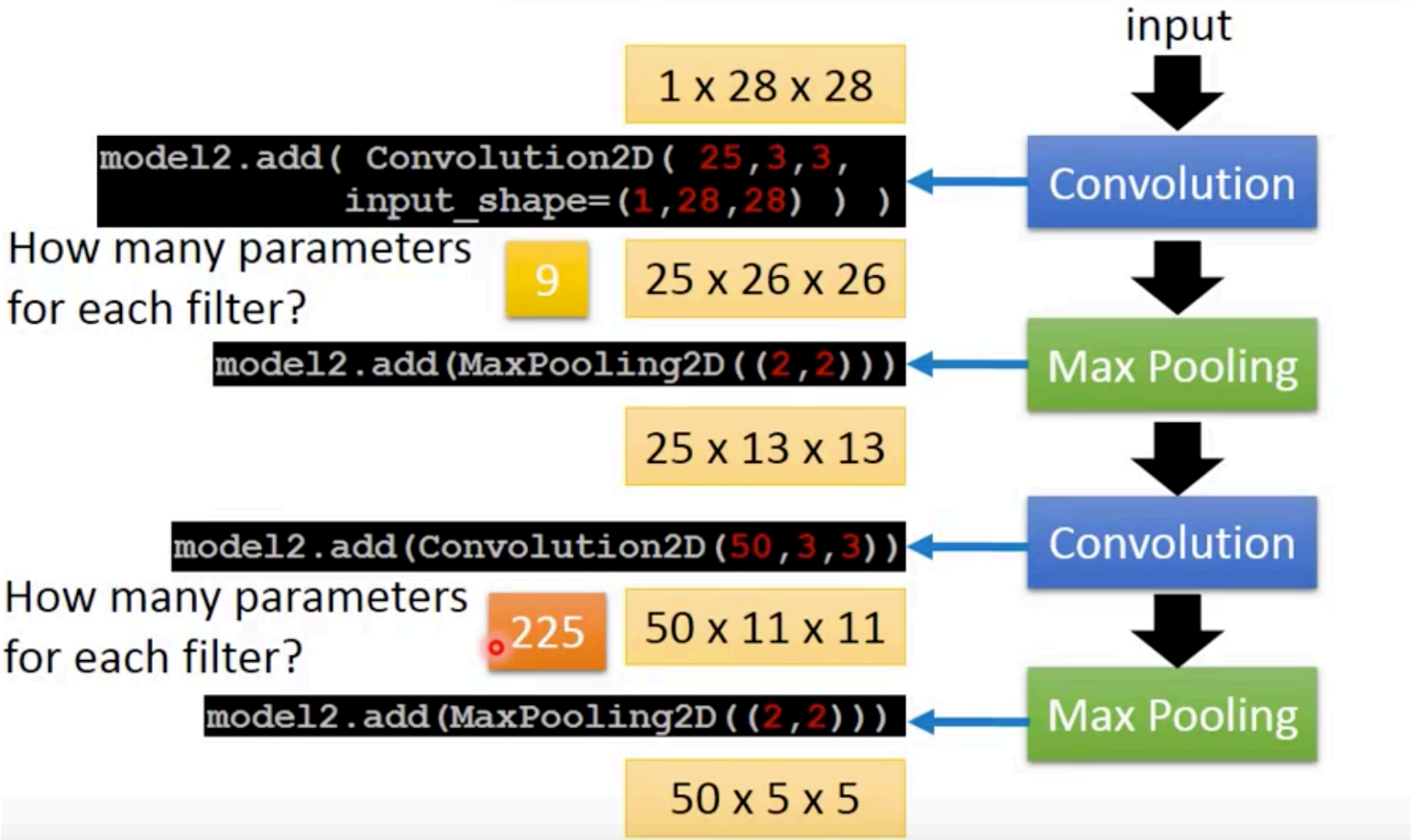
CNN in Keras

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



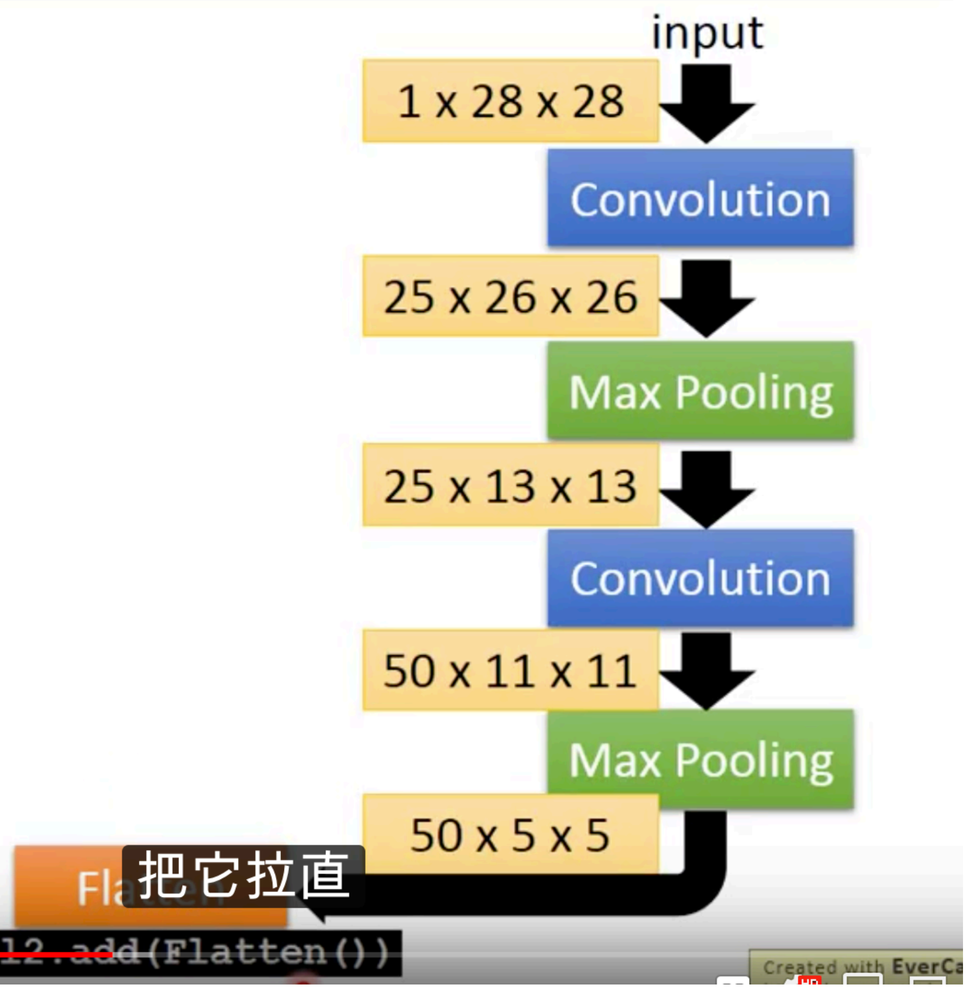
CNN in Keras

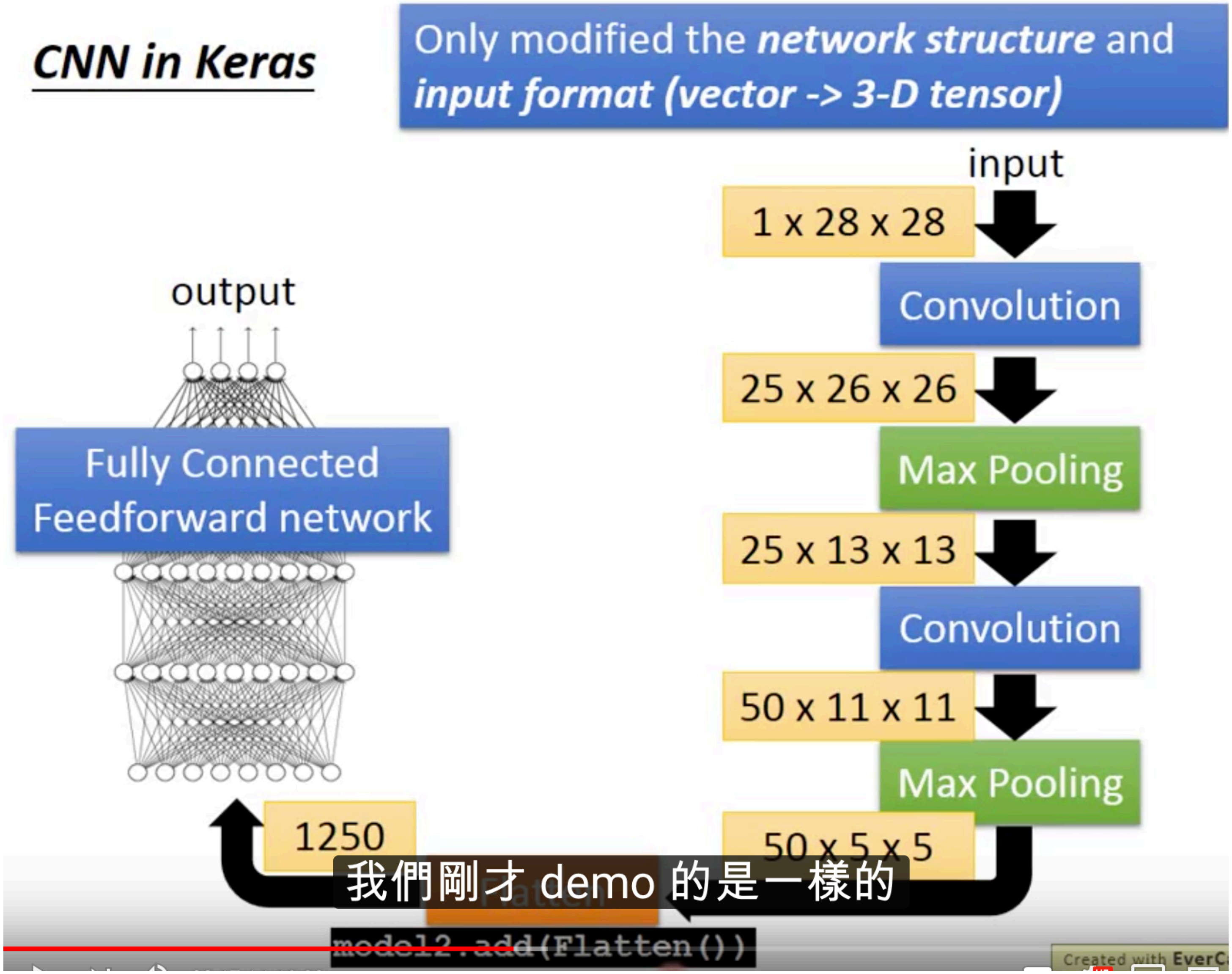
Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***



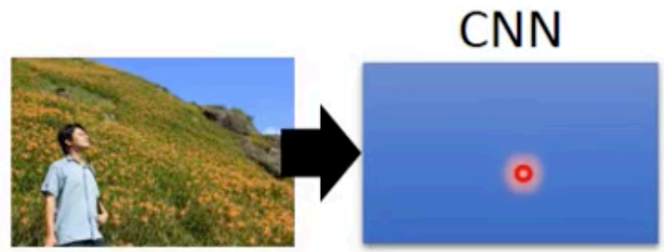
CNN in Keras

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




CNN in Keras
Only modified the network structure and input format (vector -> 3-D tensor)
input
1 x 28 x 28
Convolution
25 x 26 x 26
Max Pooling
25 x 13 x 13
Convolution
50 x 11 x 11
Max Pooling
50 x 5 x 5
output
Fully Connected Feedforward network
1250
我們剛才 demo 的是一樣的
model2.add(Flatten())

Deep Dream



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

Deep Dream

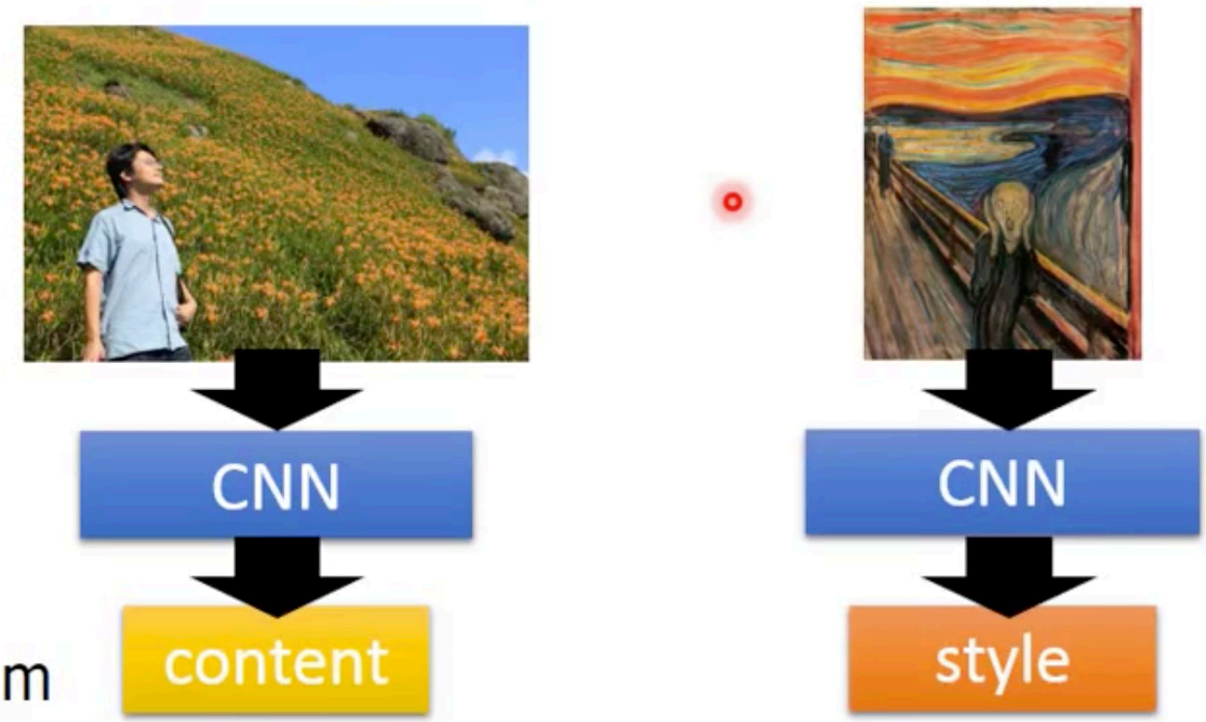
- Given a photo, machine adds what it sees ......

那 Deep Dream 還有一個進階的版本

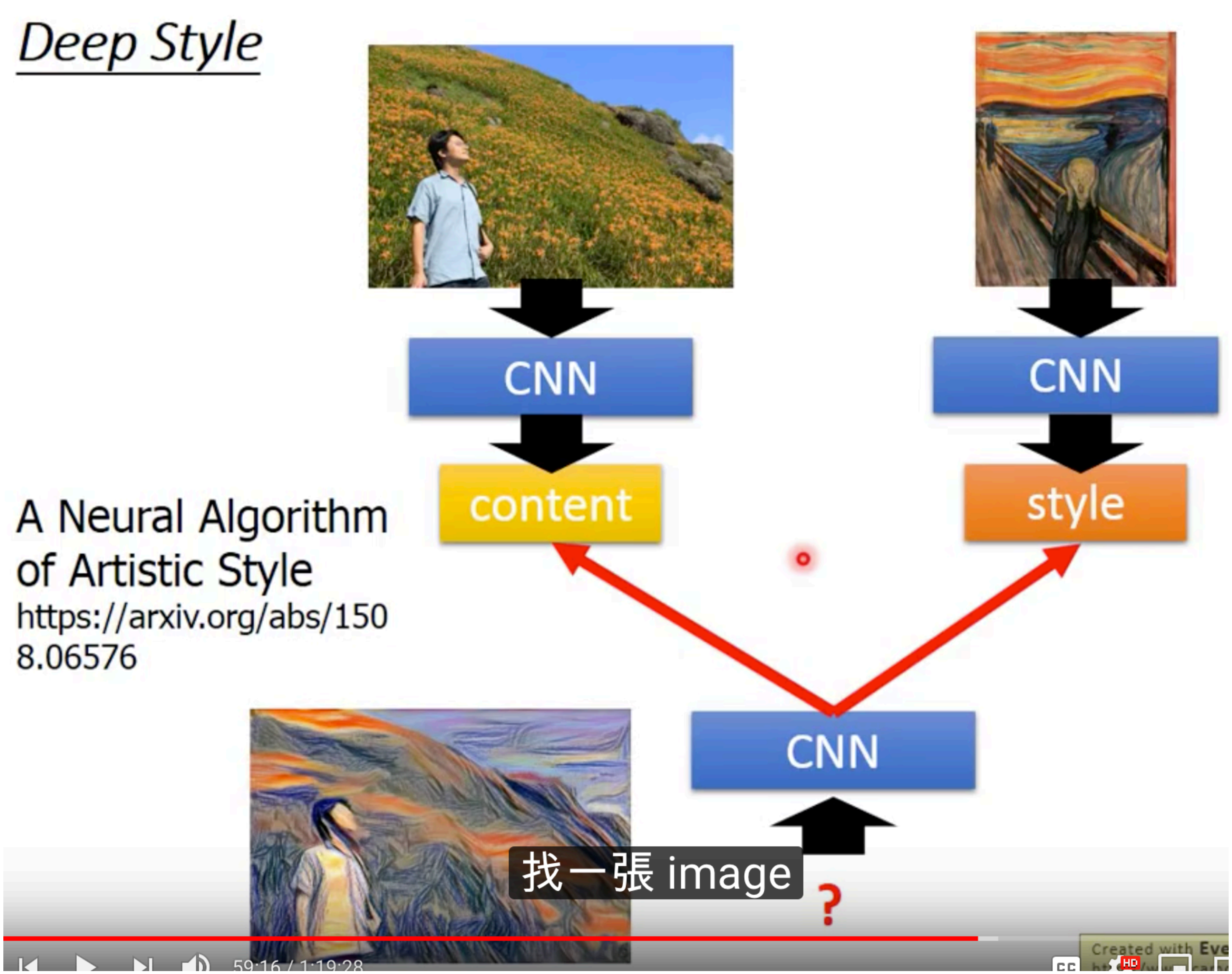
Deep Style

- Given a photo, make its style like famous paintings

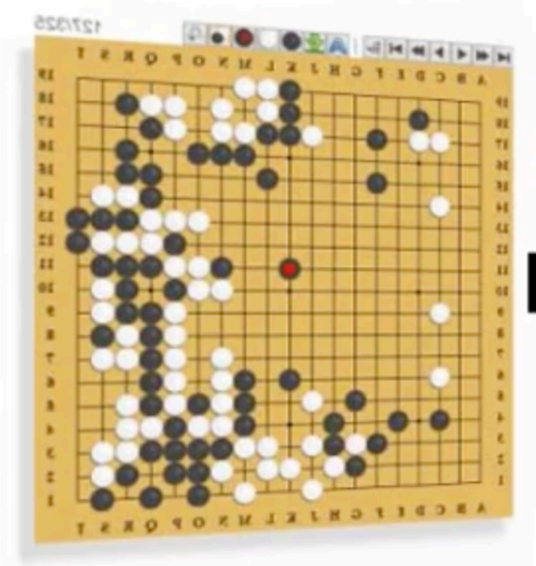
__Deep Style__



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


Deep Style
CNN
CNN
content
style
A Neural Algorithm of Artistic Style
https://arxiv.org/abs/1508.06576
CNN
?
找一張 image
59:16 / 1:19:28

More Application: Playing Go



19 x 19 vector

Network

Next move (19 x 19 positions)

19 x 19 vector

Black: 1

white: -1

Fully-connected feedforward network can be used

But CNN performs much better.

More Application: Playing Go

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

CNN

Target:
"天元" = 1
else = 0

CNN

Target:
"五之 5" = 1
else = 0

我們知道說，大家都是

Why CNN for playing Go?

- Some patterns are much smaller than the whole image

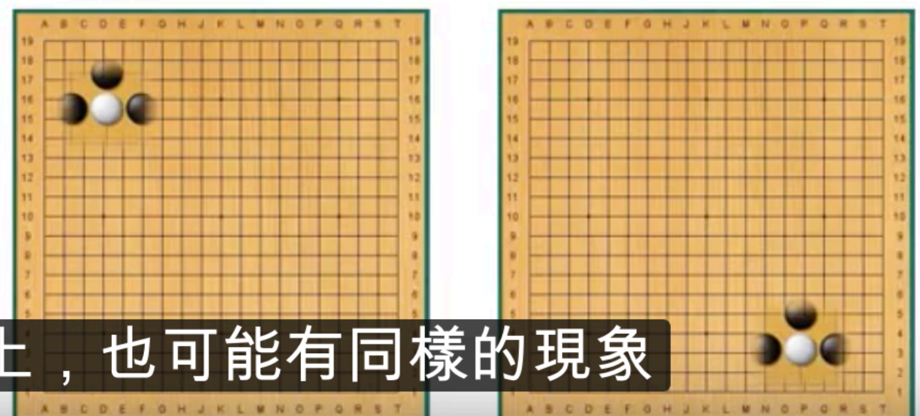
- The same patterns appear in different regions.

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.



在圍棋上，也可能有同樣的現象

Why CNN for playing Go?

- Subsampling the pixels will not change the object

➡ Max Pooling How to explain this???

Why CNN for playing Go?

- Subsampling the pixels will not change the object

→ Max Pooling How to explain this???

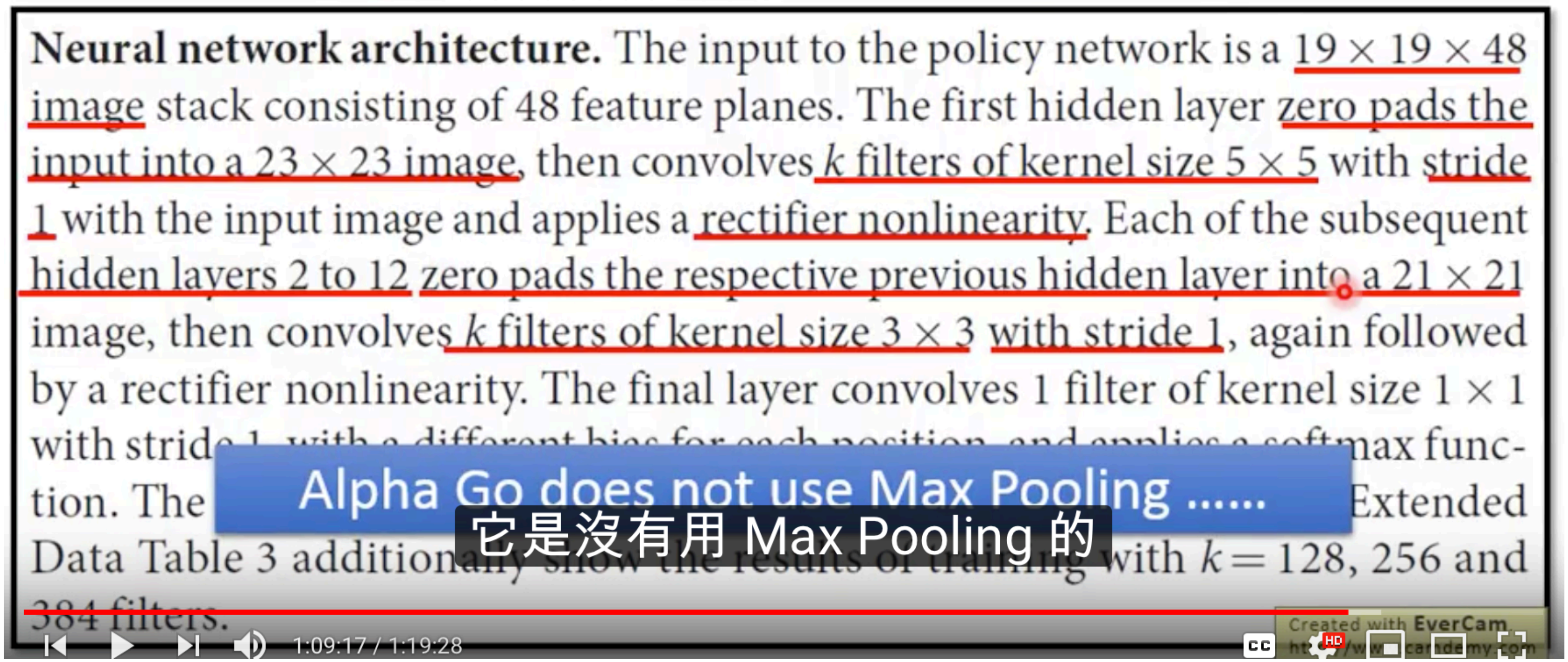
Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier 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 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.

叫吃的狀態呢，等等

Why CNN for playing Go?

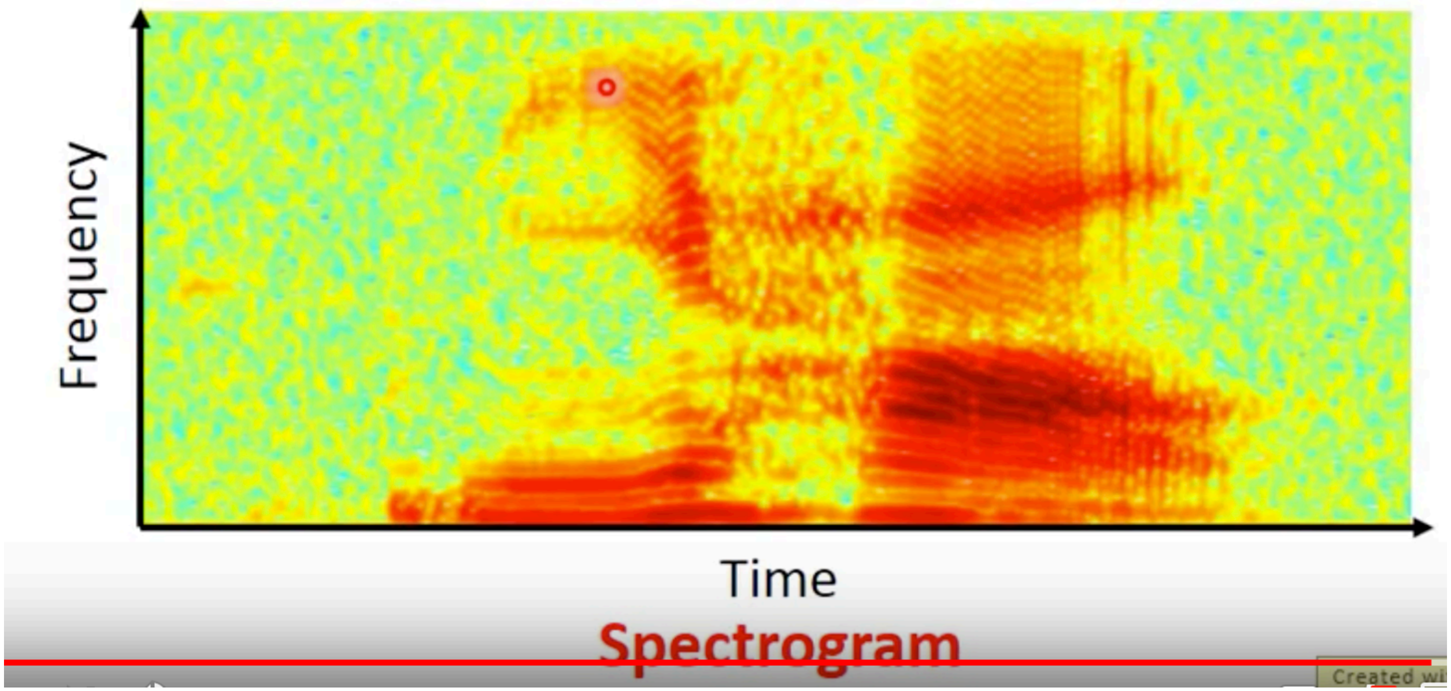
- Subsampling the pixels will not change the object

→ Max Pooling   How to explain this???



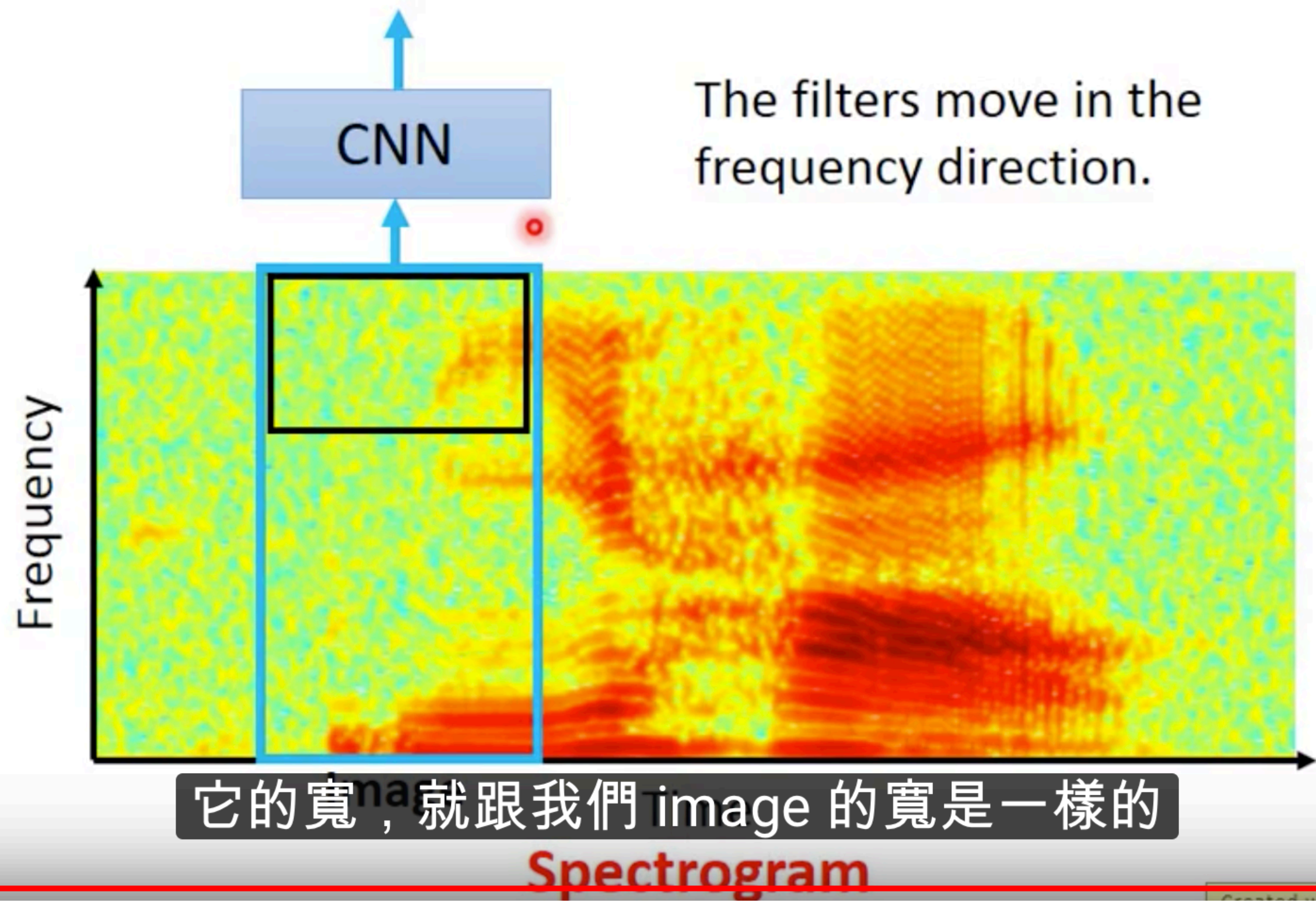
Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier 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.

More Application: Speech



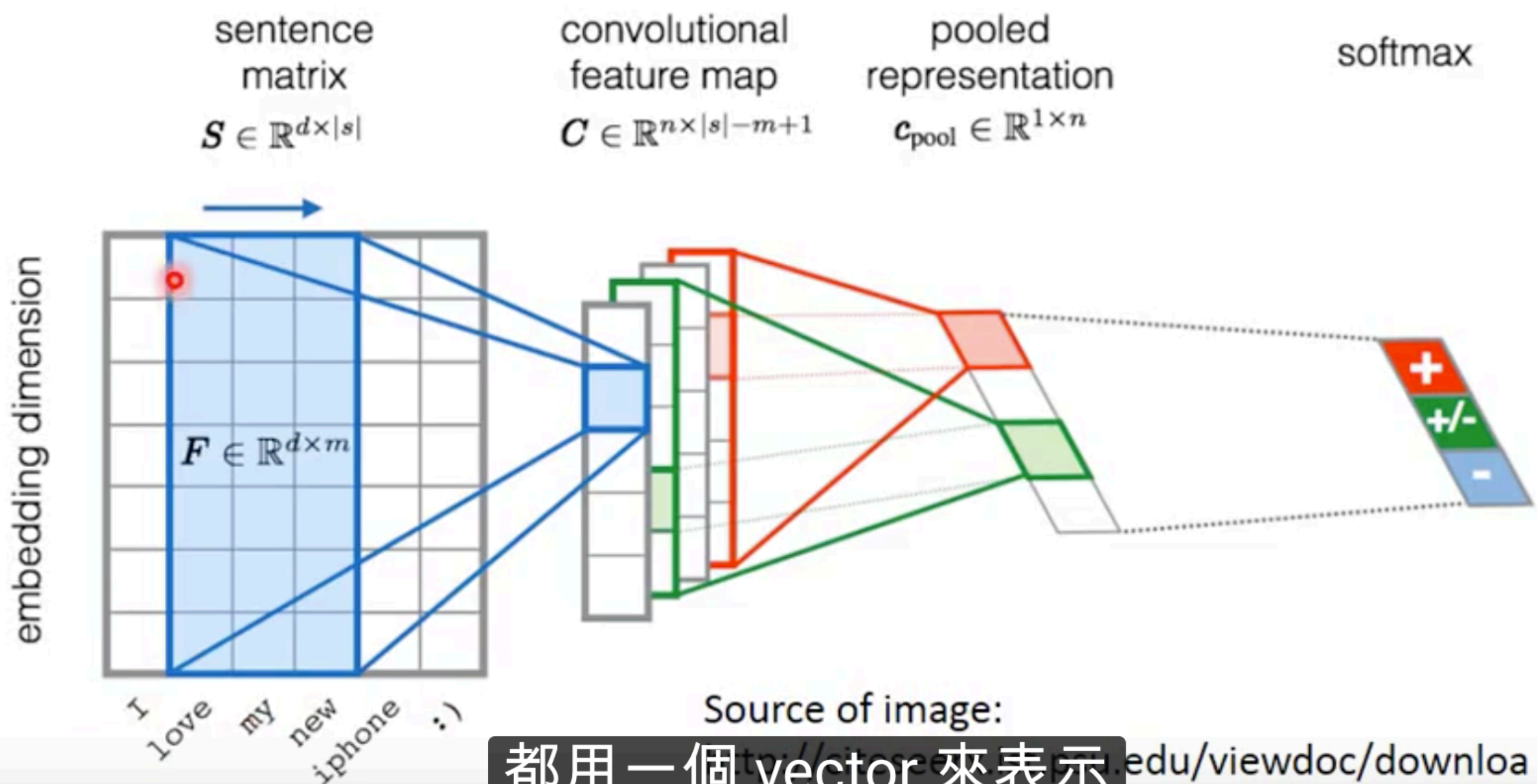
More Application: Speech

The filters move in the frequency direction.



它的寬，就跟我們 image 的寬是一樣的

More Application: Text



Source of image:
...edu/viewdoc/downloa
d?doi=10.1.1.703.6858&rep=rep1&type=pdf

都用一個 vector 來表示

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/