Convolutional Neural Networks

(Many slides from Stanford CS230, Stanford CS231n, and MIT 6.S191)

Outline

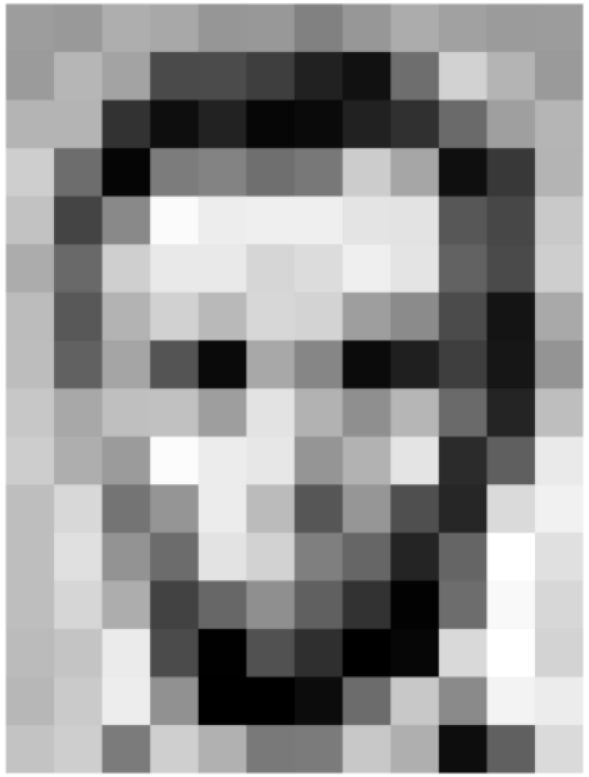
1. Basic Idea of Convolutional Neural Networks
2. CNN: Three Types of Layers
3. Convolution Layer (跟數學上的不太一樣，但概念是一樣的)
4. Pooling Layer
5. Fully connected layer
6. Summary

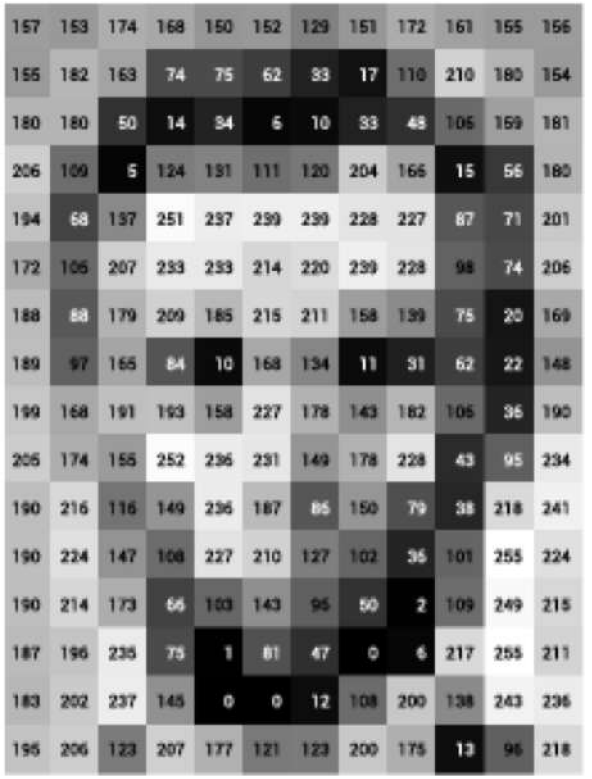
Convolutional Neural Networks

- A convolutional neural network (CNN) is a specialized type of **deep learning** models with one or more **convolutional** layers (and some other layers).
- It is most commonly applied to **computer vision** tasks to understand digital images, such as processing, analyzing, and feature extraction.(最常應用在處理電腦影像的問題)
- 應用：可以訓練電腦去看醫學的片子，節省醫生的時間

最有名的：辨識images
應用：處理手寫辨識(支票、簽名)

Images are Numbers





157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

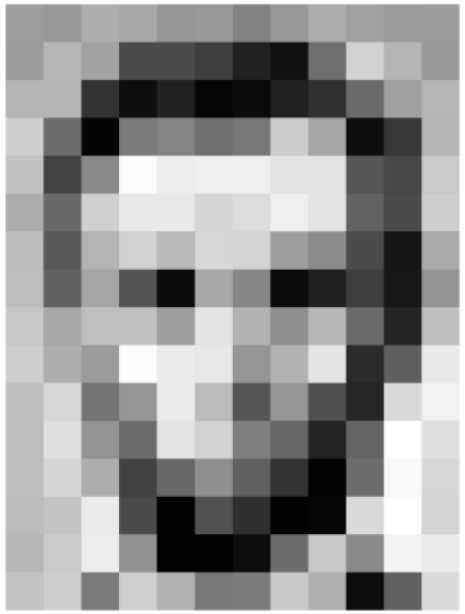
What the computer sees

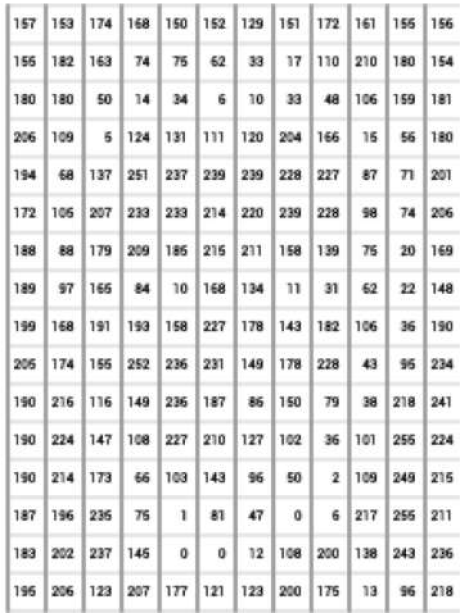
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

Tasks in Computer Vision

此例有使用soft max，
下方矩陣的sum是1





157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

classification

Lincoln	0.8
Washington	0.1
Jefferson	0.05
Obama	0.05

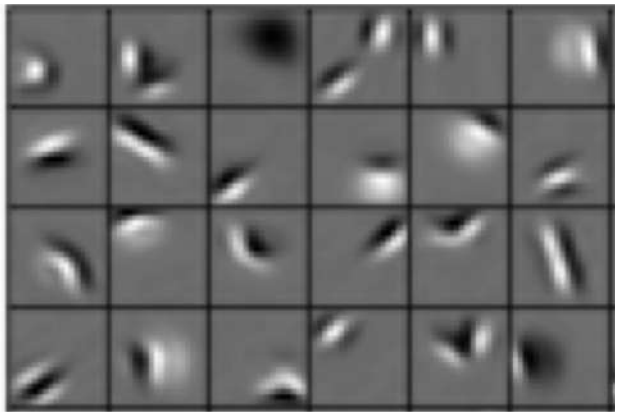
Input Image　　Pixel Representation

- **Classification**: output variable takes class label. Can produce probability of belonging to a particular class

Learning Feature Representations

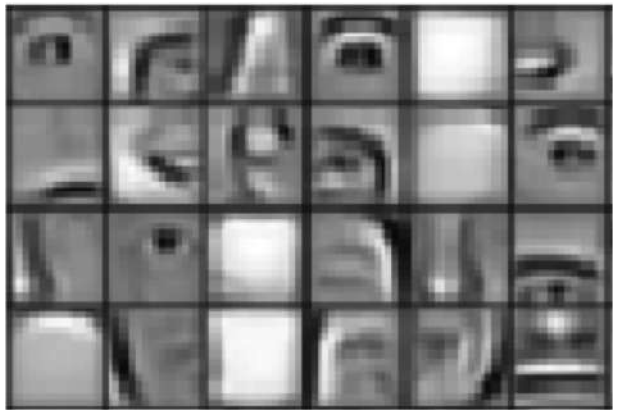
Can we learn a **hierarchy of features** directly from the data instead of hand engineering(資料的前置處理)?

Low level features



Edges, dark spots

Mid level features



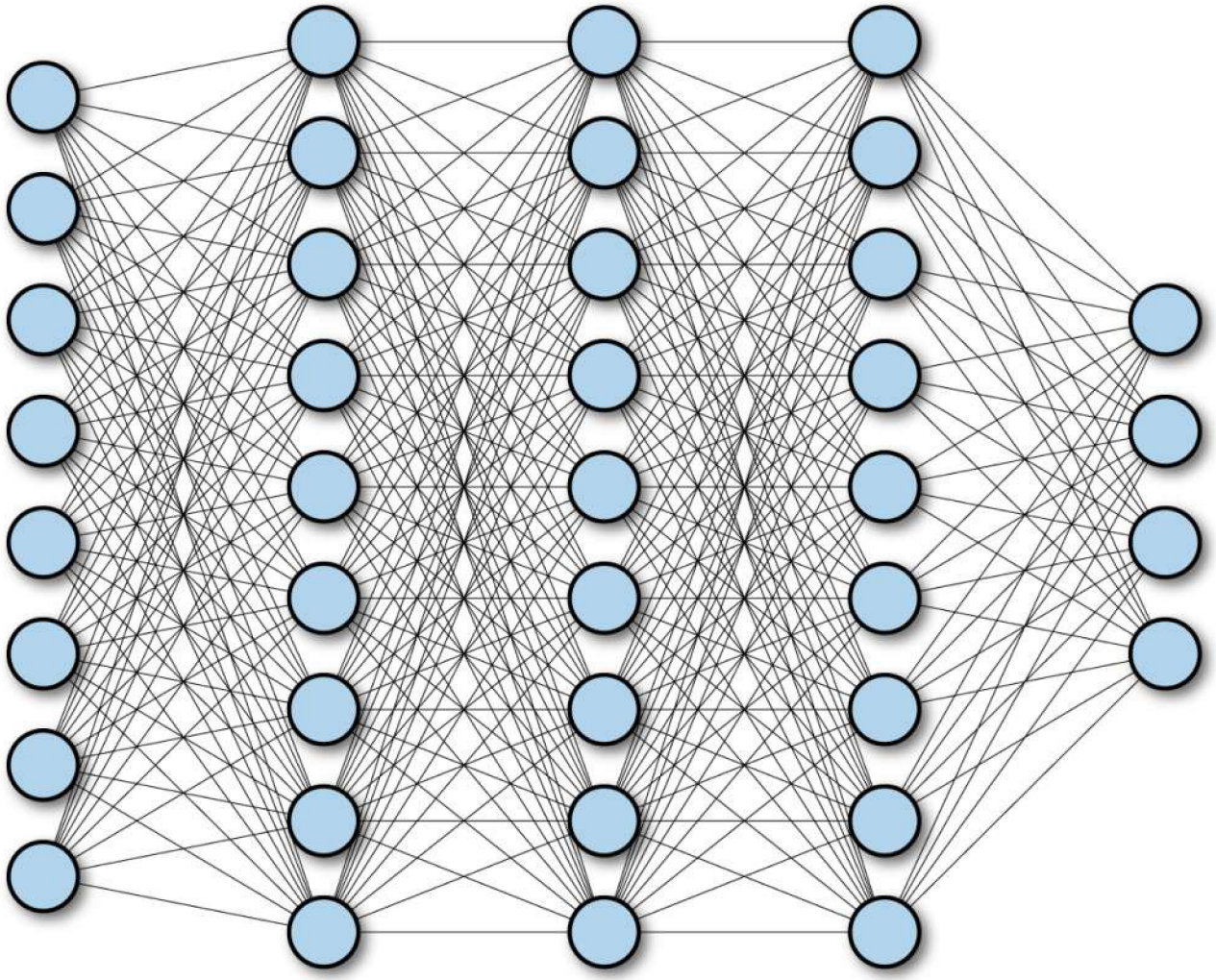
Eyes, ears,nose

High level features

Facial structure

先告訴電腦眼睛在哪裡?嘴巴在哪裡?
偵測好後再告訴電腦眼睛是怎樣大小是怎樣?(早期還沒有convolution neural network時的做法)

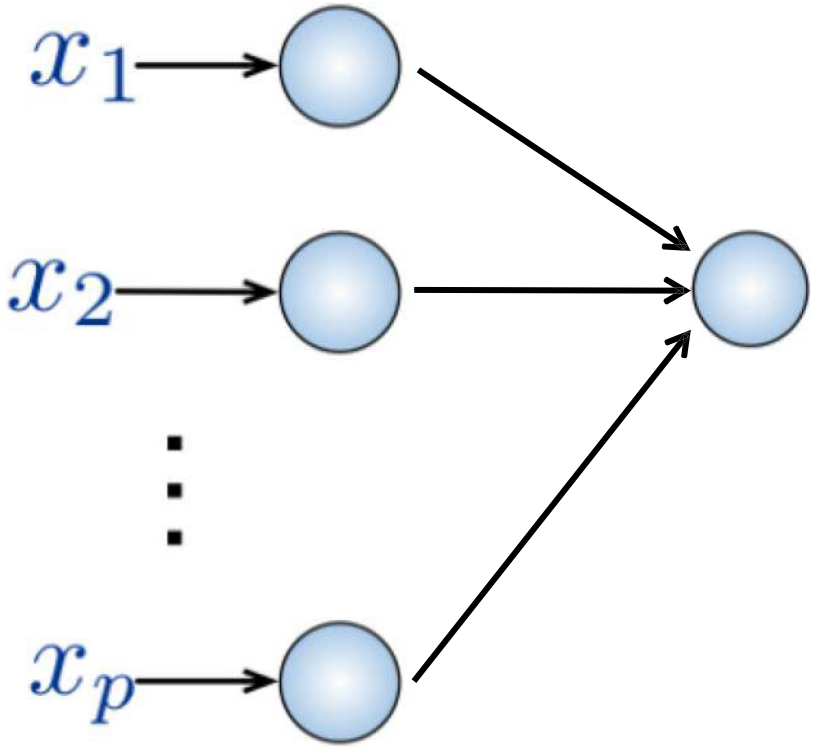
Fully Connected Neural Network



Fully Connected Neural Network

Input:

- 2D image
- Vector of pixel values



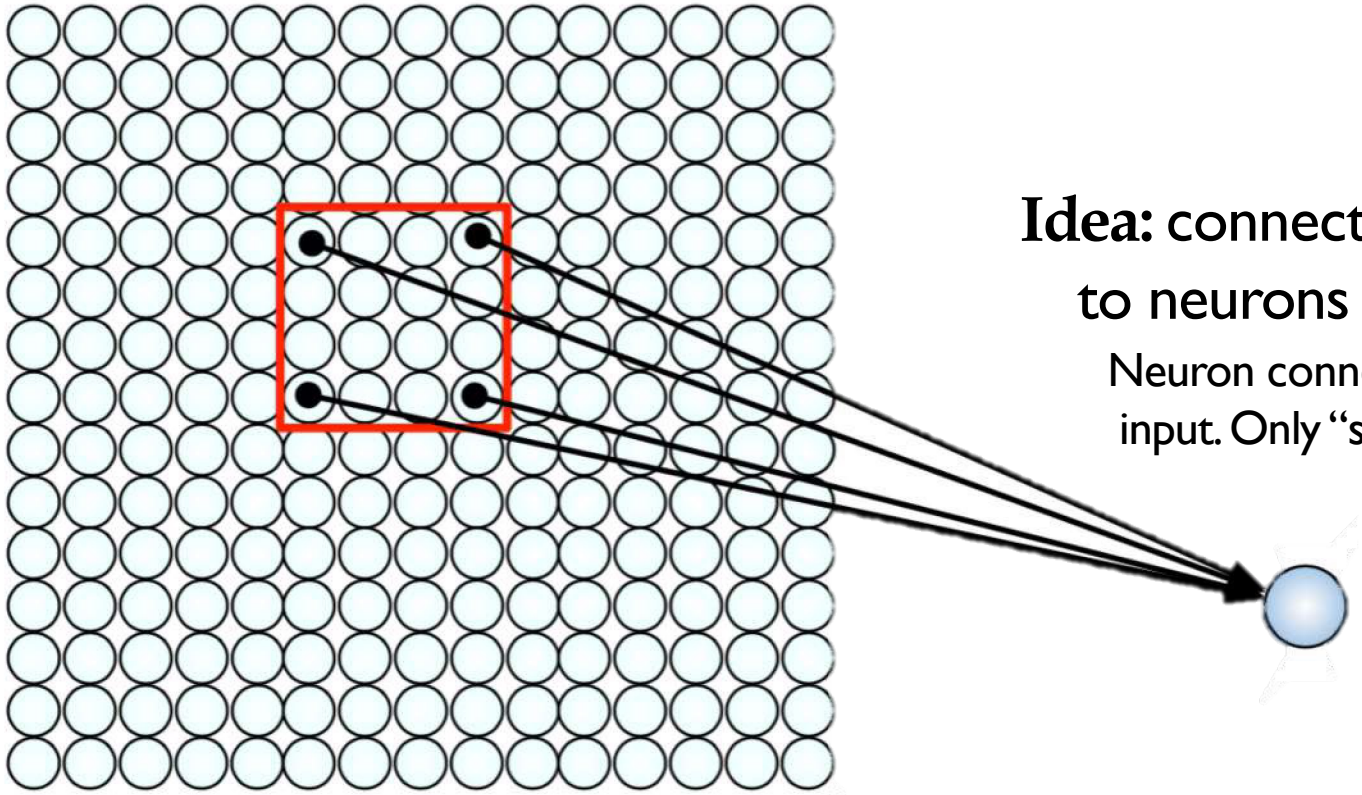
Fully Connected:

- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

How can we use **spatial structure** in the input to inform the architecture of the network?

Using Spatial Structure

Input: 2D image.
Array of pixel values

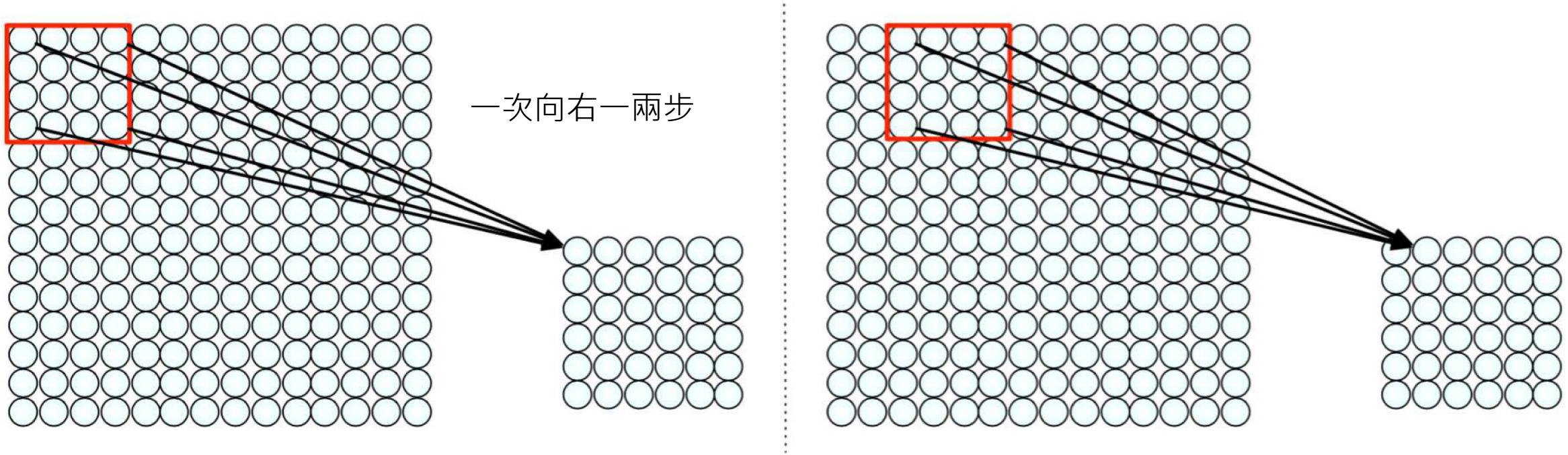


Idea: connect patches of input to neurons in hidden layer.

Neuron connected to region of input. Only "sees" these values.

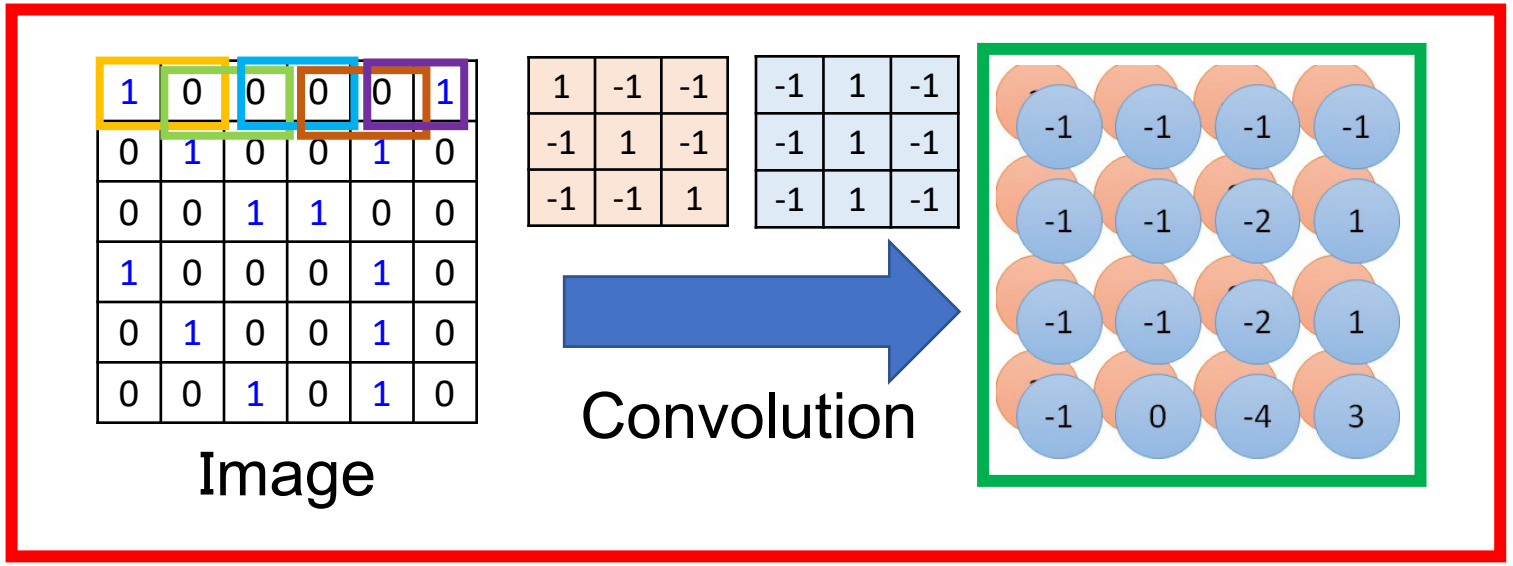
Using Spatial Structure

用16個weight來掃描整個二維影像，所以它的parameter銳減

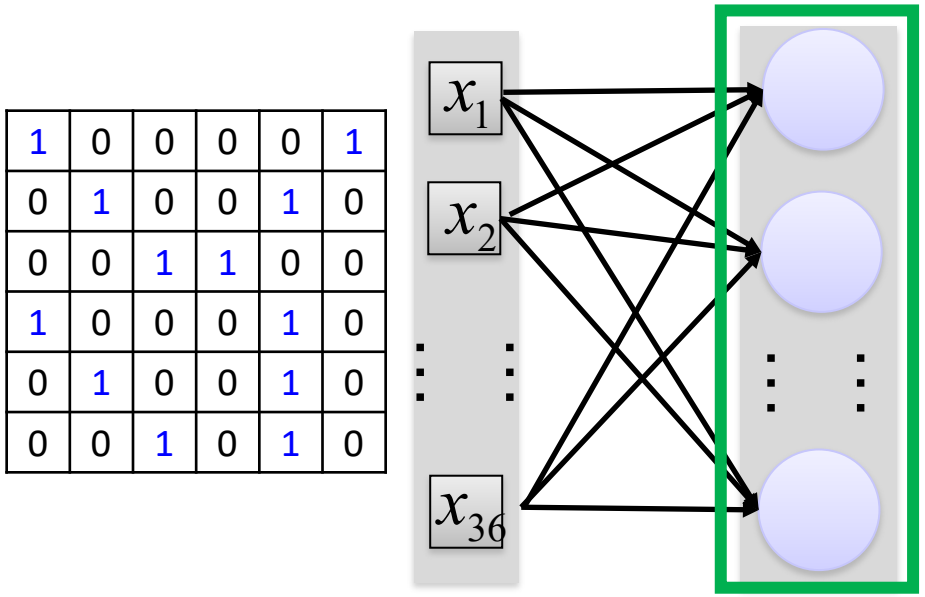


Connect patch in input layer to a single neuron in subsequent layer.
Use a sliding window to define connections.
How can we **weight** *the patch to detect particular features?*

Convolution vs. Fully Connected



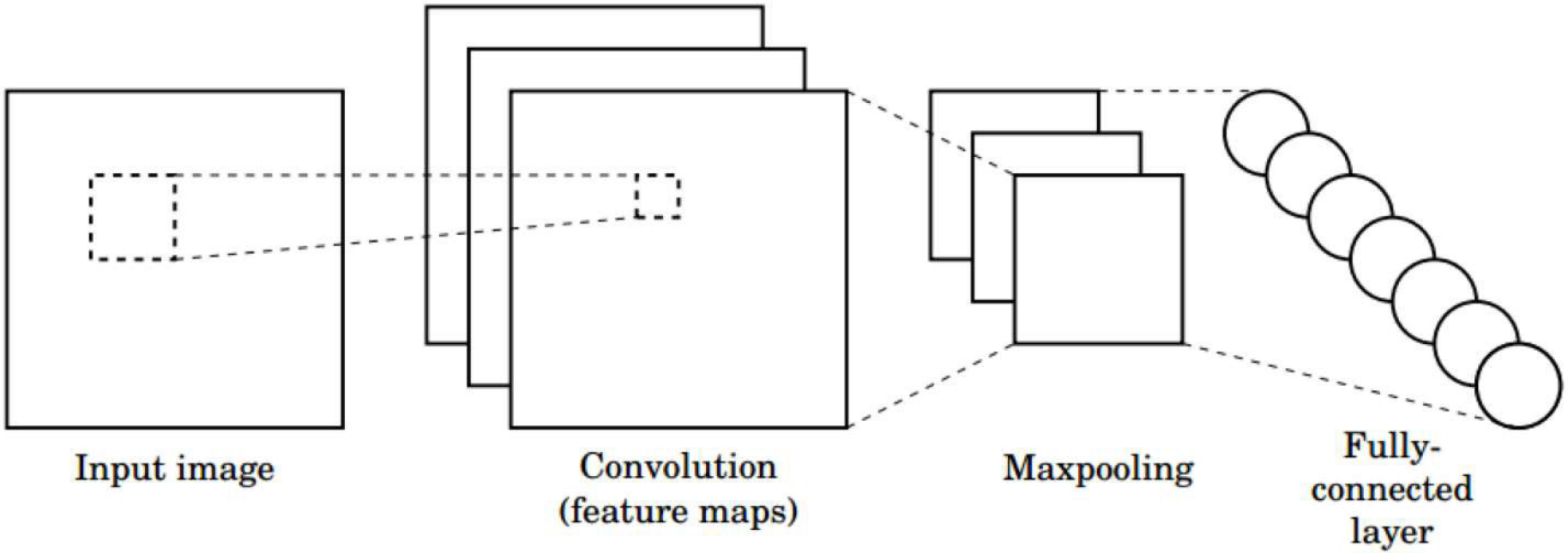
Fully-connected



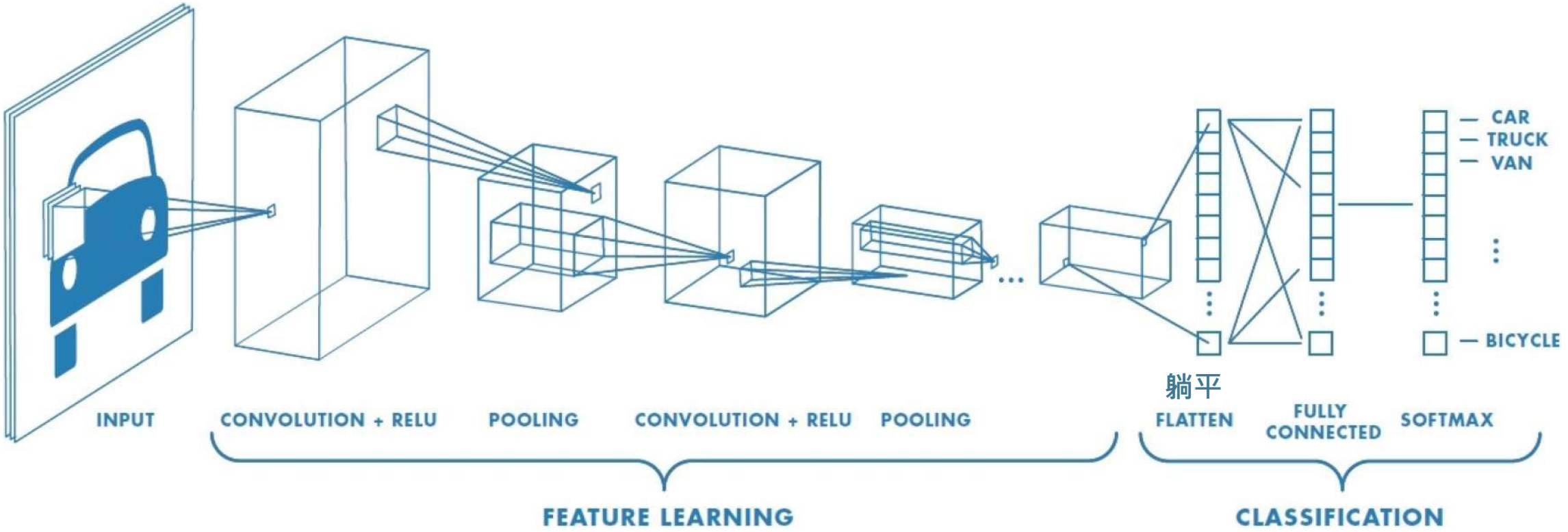
CNN: Three Types of Layers

- In a convolutional neural network (CNN), there are basically three types of layers:
  - **Convolution layer卷積層**
    - 卷積運算就是將原始圖片的與特定的Feature Detector(filter)做卷積運算(符號⊗)
  - **Pooling layer池化層**
    - 在Pooling Layer這邊主要是採用Max Pooling，Max Pooling的概念很簡單只要挑出矩陣當中的最大值就好，Max Pooling主要的好處是當圖片整個平移幾個Pixel的話對判斷上完全不會造成影響，以及有很好的抗雜訊功能。
  - **Fully connected layer全連接層**
    - 將之前的結果平坦化之後接到最基本的神經網絡

CNN: Three Types of Layers



CNN: Three Types of Layers

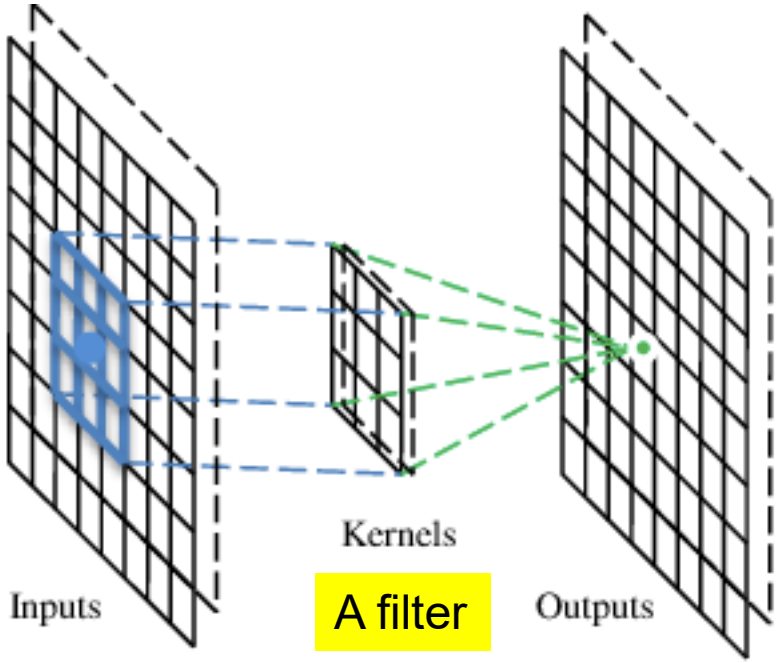


Convolution Layer

1. Filter (Kernel)
2. Padding
3. Stride
4. Convolutions Over Volumes
5. One Layer of a Convolutional Network
6. Why Convolutions?

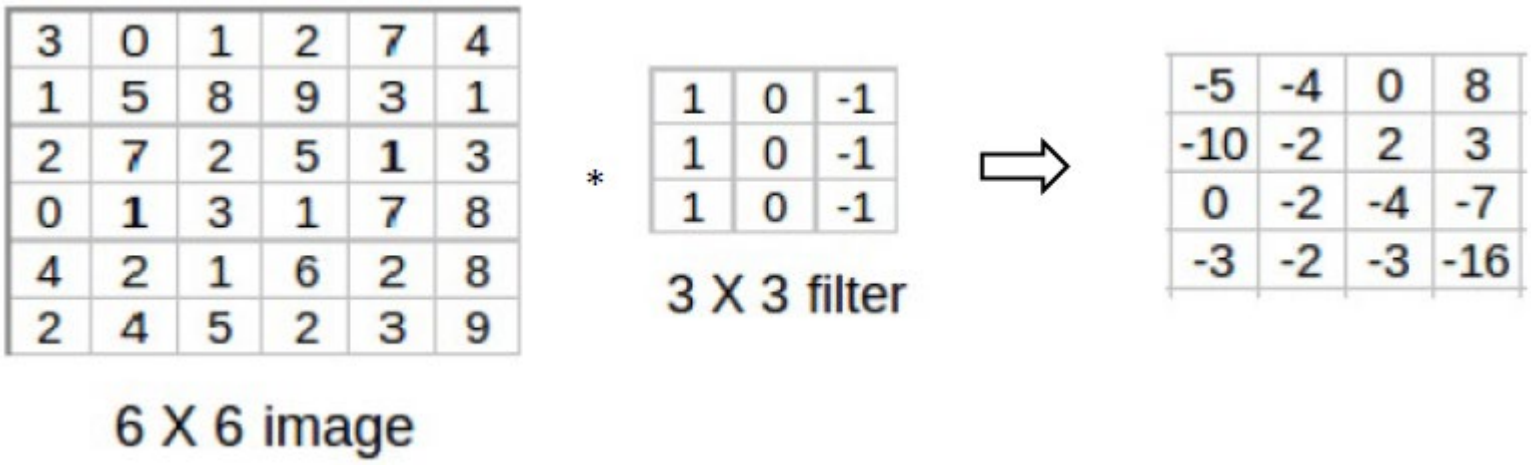
A convolutional layer

A convolutional layer has a number of filters that does convolutional operation.



Filter (Kernel)

- Example: Convolve a 6 × 6 image with a 3 × 3 filter:



純量越大，兩者越像，
跟cosθ有關

Filter

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

These are the network parameters to be learned.

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

Filter 1

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

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Filter

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

Filter 1

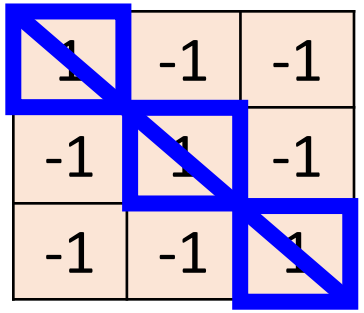
Stride = 1

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

Dot product → 3 -1

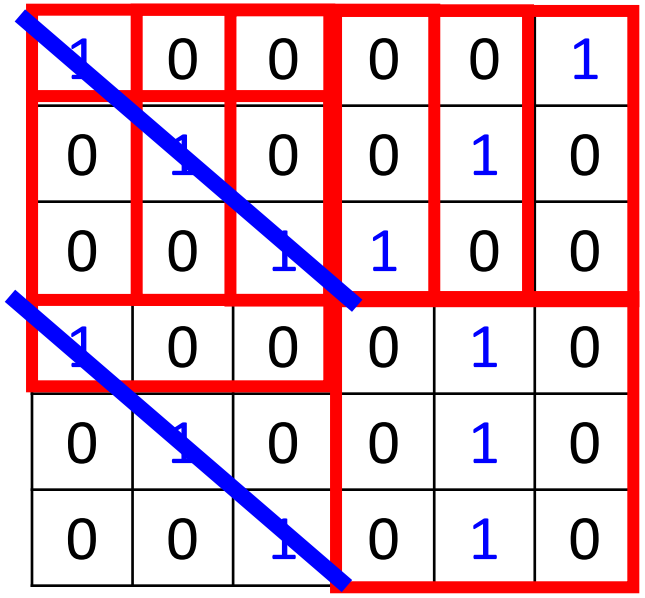
6 x 6 image

Filter

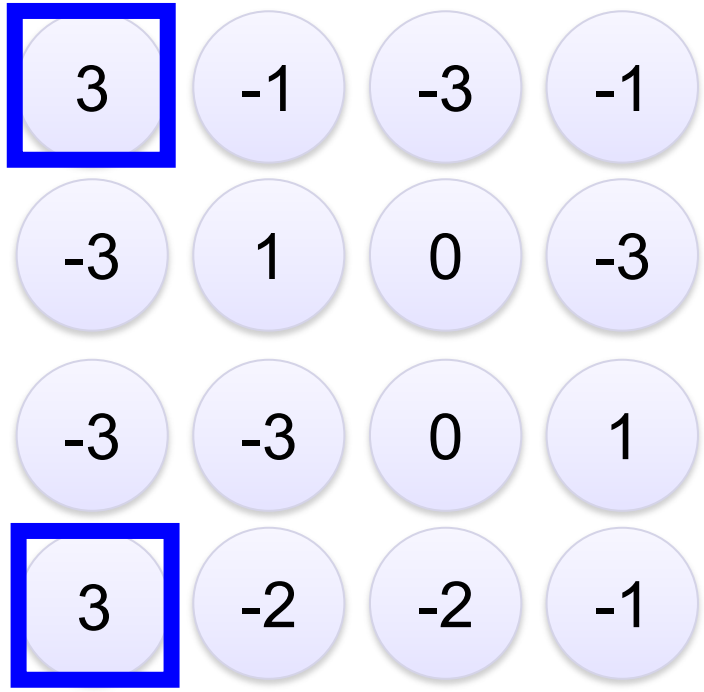


Filter 1

Stride = 1



6 x 6 image



Convolution

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

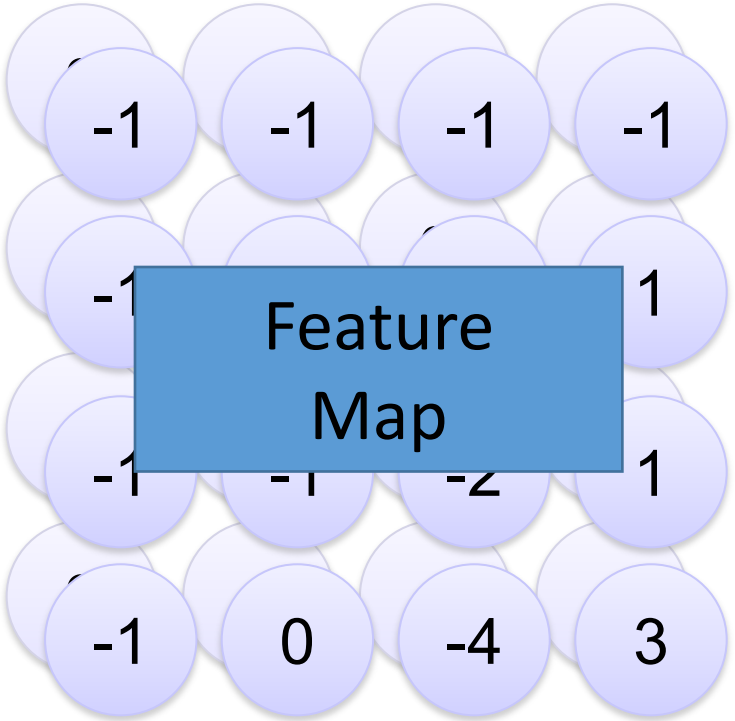
Filter 2

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

Repeat this for each filter



Two 4 x 4 images
Forming two 4 x 4 matrix

Convolution

- Convolving an input of 6×6 dimensions with a 3×3 filter results in 4×4 output.
- We can generalize it:
  - Input size: $n \times n$
  - Filter size: $f \times f$

  $\Rightarrow$ Output size: $(n - f + 1) \times (n - f + 1)$

Padding 補丁

奇數比較適合
偶數不適合

- Two disadvantages after we apply a convolutional operation (卷積的缺點) :
  - Throwing away information from the edges
  - Shrinking the size of the image

⇒ (解決辦法)Pad the image with an additional border of zeros to overcome these issues.

Padding (需要幾層padding?)

- If we add '*p*' padding layers such that the output size is the same as the input size, then:

$$n + 2p - f + 1 = n \implies p = (f - 1) / 2$$

- For example:
  - If we use a 3×3 filter then one layer of zeros must be added to the borders. (p = (3 – 1)/2)
  - Similarly, if a 5×5 filter is used then two layers of zeros must be appended to the border of the image.

Padding

- There are two common choices for padding:

  Same convolution:
  padding so that the output size is the same as the input size
  (補丁讓輸出的size和輸入的size一樣)

  Valid convolution: no padding(沒有補丁)

Stride 滑步

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

Filter 1

If stride = 2

會除不盡

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

Stride controls how the filter convolves around the input image.

Stride

- The dimensions for stride s with padding p will be:

  Input size: $n \times n$

  Filter size: $f \times f$

  $\Rightarrow$ Output size: $\left[\frac{n+2p-f}{s}+1\right]\left[\frac{n+2p-f}{s}+1\right]$

- Example : We use a 3×3 filter to perform a valid convolution on a 7×7 input image with a stride of 2. Then the output size will be:

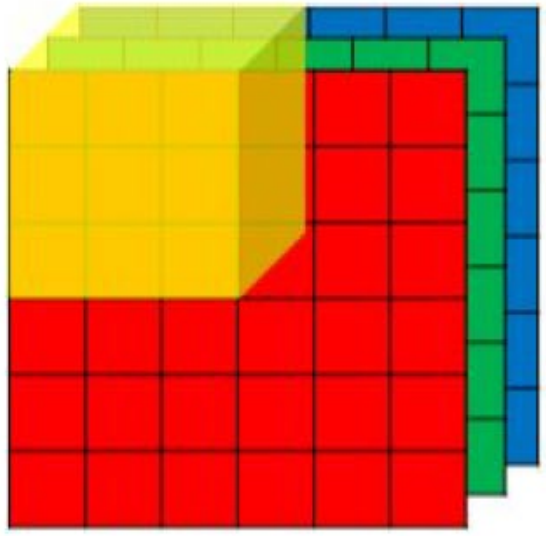
$$\left[\frac{n+2p-f}{s}+1\right]\left[\frac{n+2p-f}{s}+1\right] = \left[\frac{7+2*0-3}{2}+1\right]\left[\frac{7+2*0-3}{2}+1\right]$$

$$= \left[\frac{4}{2}+1\right]\left[\frac{4}{2}+1\right] = 3\times 3$$

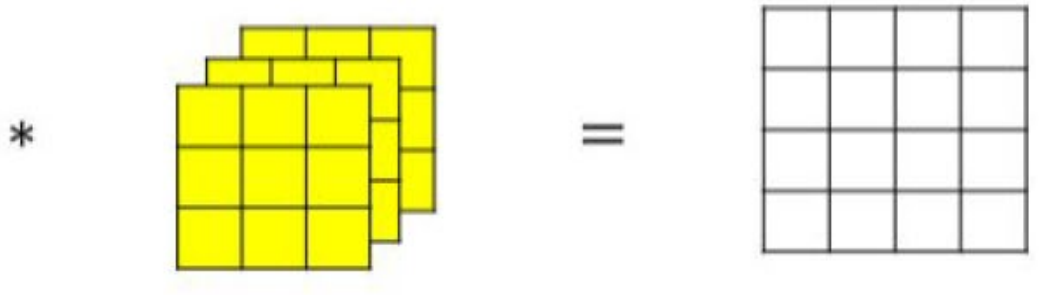
Convolutions Over Volumes

- Suppose that we have an RGB image of shape $6 \times 6 \times 3$ instead of a 2-D image. Then we will use a $3 \times 3 \times 3$ filter instead of a 3×3 filter.

  For example:



The number of channels in the input and a filter should be the same.
(輸入和過濾的channels必須相同)

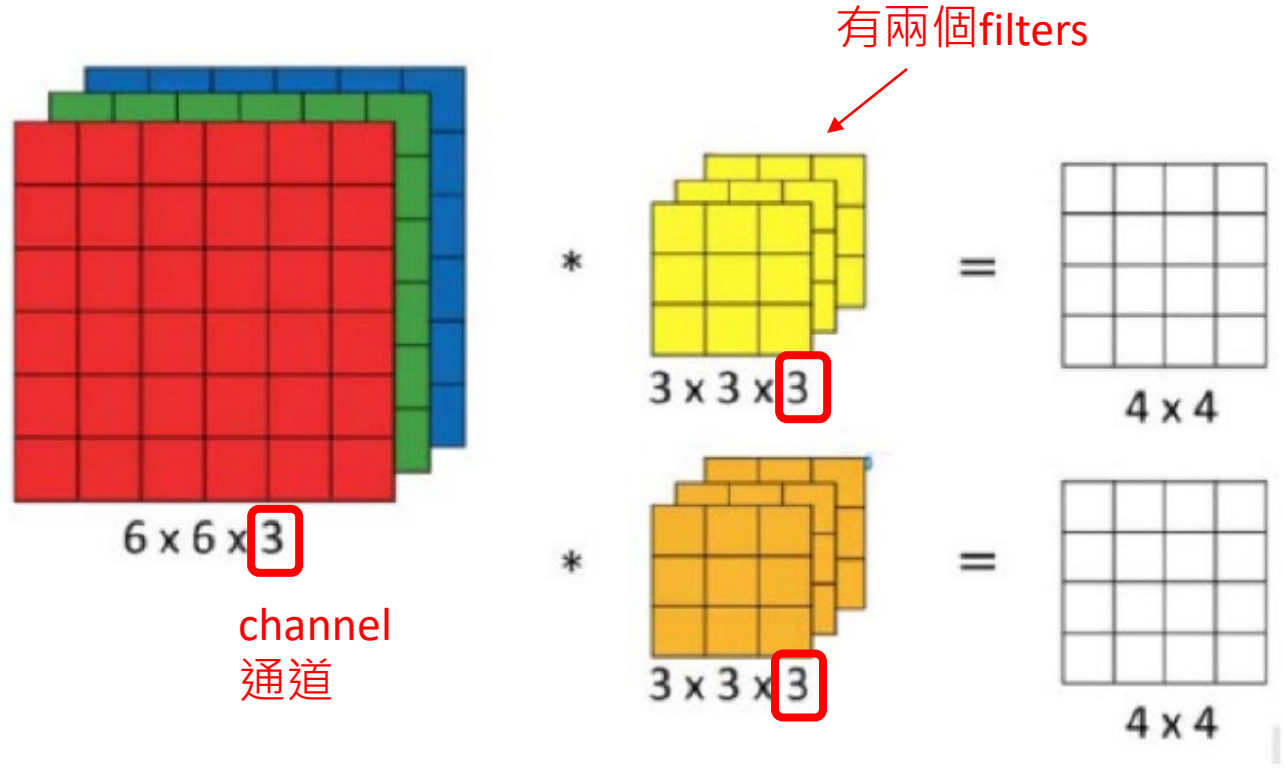


Each element of the output is the sum of the element-wise product of 27 values from an input window and the filter.

Convolutions Over Volumes

- Multiple filters:

  If we use two filters, the output dimensions will be $4 \times 4 \times 2$.



Convolutions Over Volumes

- Input size: $n \times n$

  Filter size: $f \times f$

  Stride: s

  Padding: p

  n_c: the number of channels in the input and filters

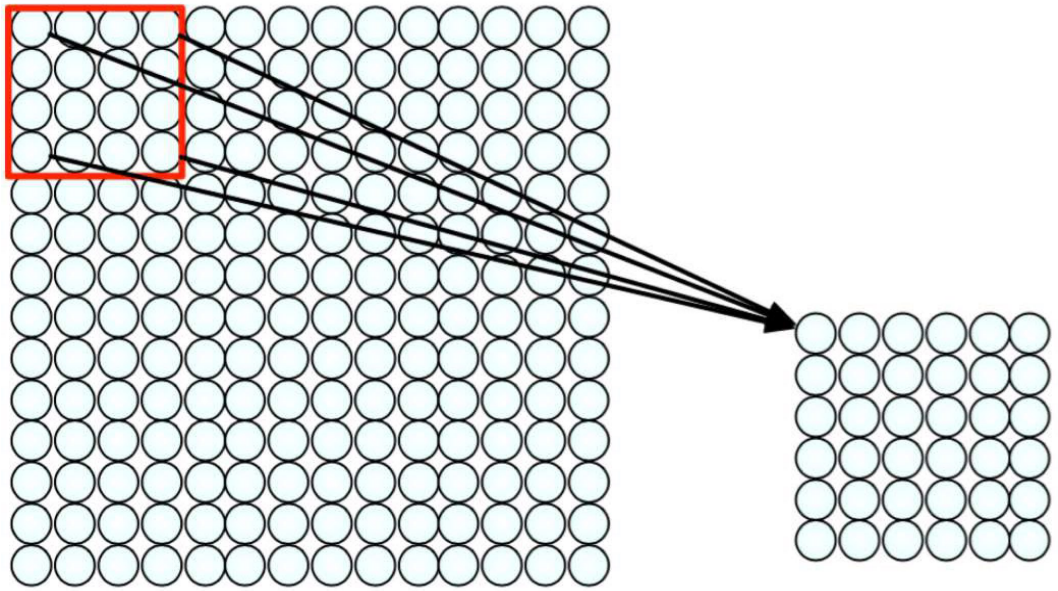
  n_c': the number of filters (有多少filters)

$\Rightarrow$ $(n \times n \times n_c) * (f \times f \times n_c) \Rightarrow \left[\frac{n+2p-f}{s}+1\right] * \left[\frac{n+2p-f}{s}+1\right] \times n_c'$

- For example:

  $(6 \times 6 \times 3) * (3 \times 3 \times 3)$ with $p = 0$, $s = 1$, and $n_c' = 2 \Rightarrow 4 \times 4 \times 2$

One Layer of a Convolutional Network



For a neuron in hidden layer:

- Take inputs from patch
- Compute weighted sum + bias
- Apply an activation function

4x4 filter: matrix of weights w_{ij}

$$\sum_{i=0}^{3}\sum_{j=0}^{3} w_{ij}\, x_{i+p,j+q} + b$$

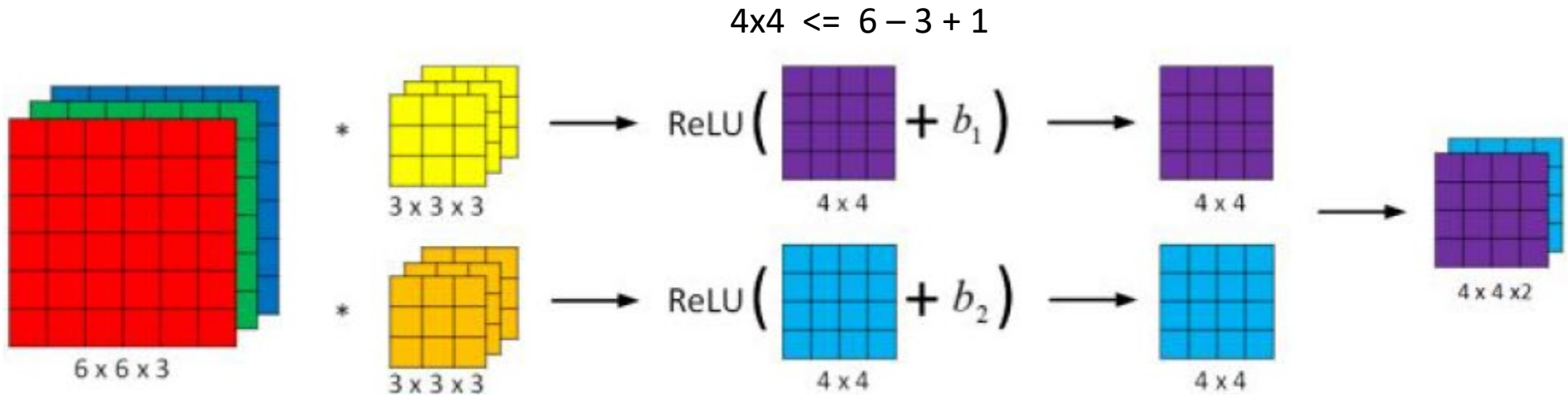
for neuron (p,q) in hidden layer

1) applying a window of weights
2) computing linear combinations
3) activating with non-linear function

One Layer of a Convolutional Network

- Once we get an output after taking a dot product between the filter and a small chunk of the image, we add a bias term to the output and finally apply an activation function.
- Note that the number of parameters in the convolutional layer is independent of the size of the image. It essentially **depends on the filter size**.

One Layer of a Convolutional Network



One Layer of a Convolutional Network

- **Example**:
- Given an RGB image of shape $32 \times 32 \times 3$, suppose we have 6 filters, each of shape $5 \times 5 \times 3$ with stride 1 and pad 0.

What will be the output volume size?

Solution:

Output volume size:

$$\left[\frac{n + 2p - f}{s} + 1\right]\left[\frac{n + 2p - f}{s} + 1\right] \times n_c'$$

$$= \left[\frac{32 + 2*0 - 5}{1} + 1\right]\left[\frac{32 + 2*0 - 5}{1} + 1\right] \times 6 = 28 \times 28 \times 6$$

One Layer of a Convolutional Network

- Example:
- Given an RGB image of shape $32 \times 32 \times 3$, suppose we have 6 filters, each of shape $5 \times 5 \times 3$ with stride 1 and pad 0.

  What will be the number of parameters in the layer?

  Solution:

  Number of parameters for each filter (一個filter)= 5*5*3 = 75

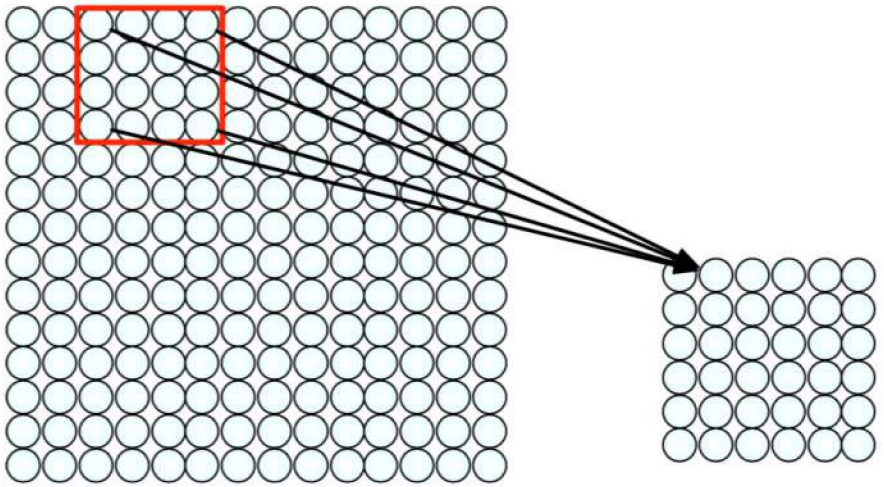
  There will be a bias term for each filter, so total parameters per filter = 76 (一個filter會有一個bias，所以+1)

  As there are 6 filters, the total parameters for that layer = 76*6 = 456 (共有6個filters，所以會有76*6(個)

Why Convolutions?

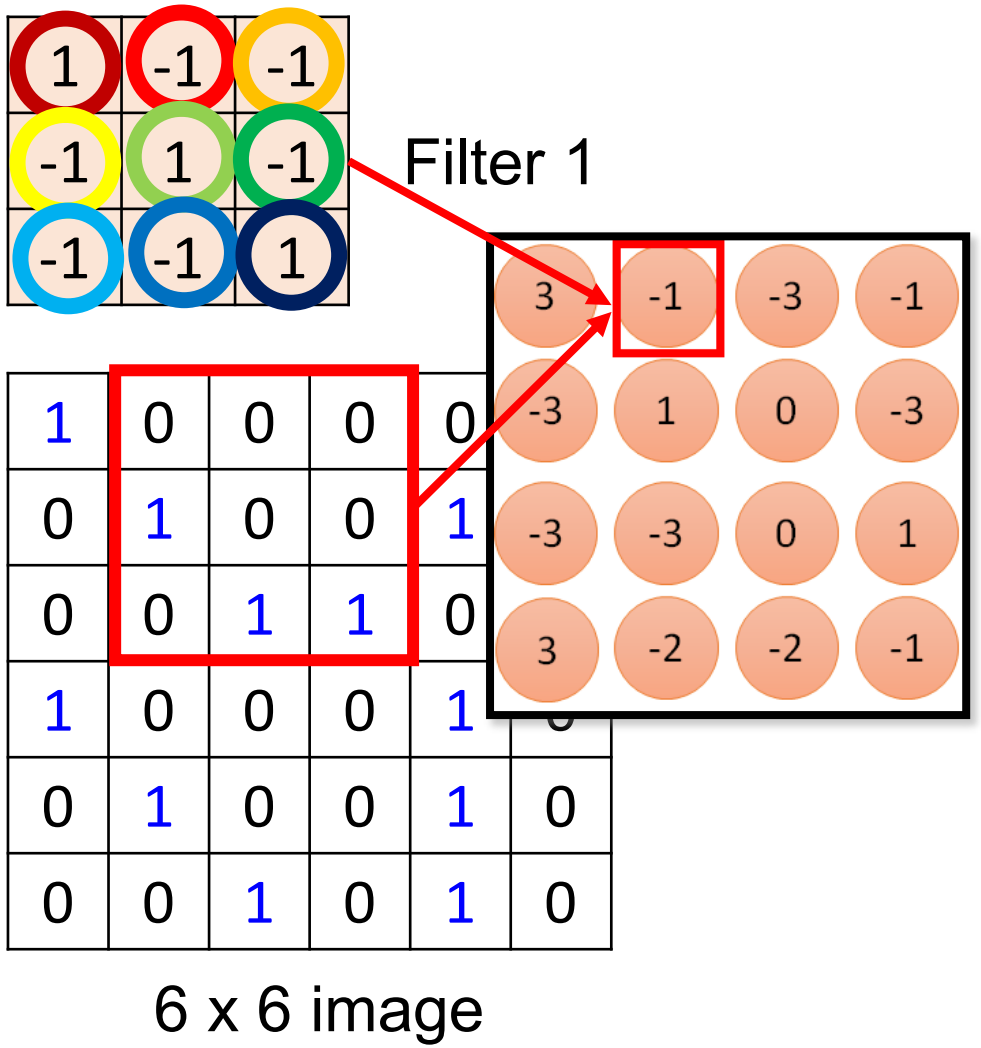
- Two main advantages(優點) of convolutions are:
  - Parameter sharing
    - To learn the same feature in different parts of the image, a single feature detector (filter) is convolved over the entire input and hence the parameters are shared.
  - Sparsity of connections
    - In each layer, each output value depends only on a small number of inputs.

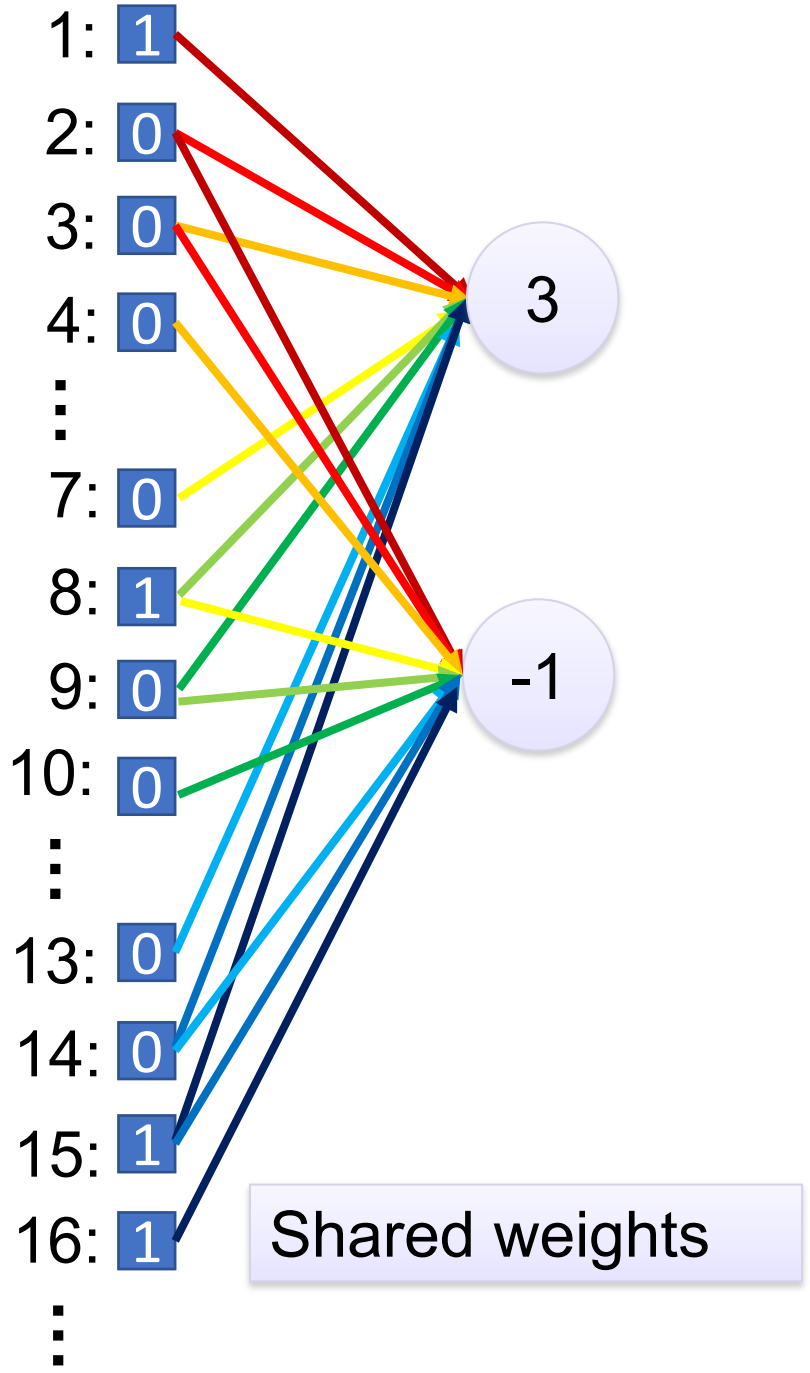
Why Convolutions?



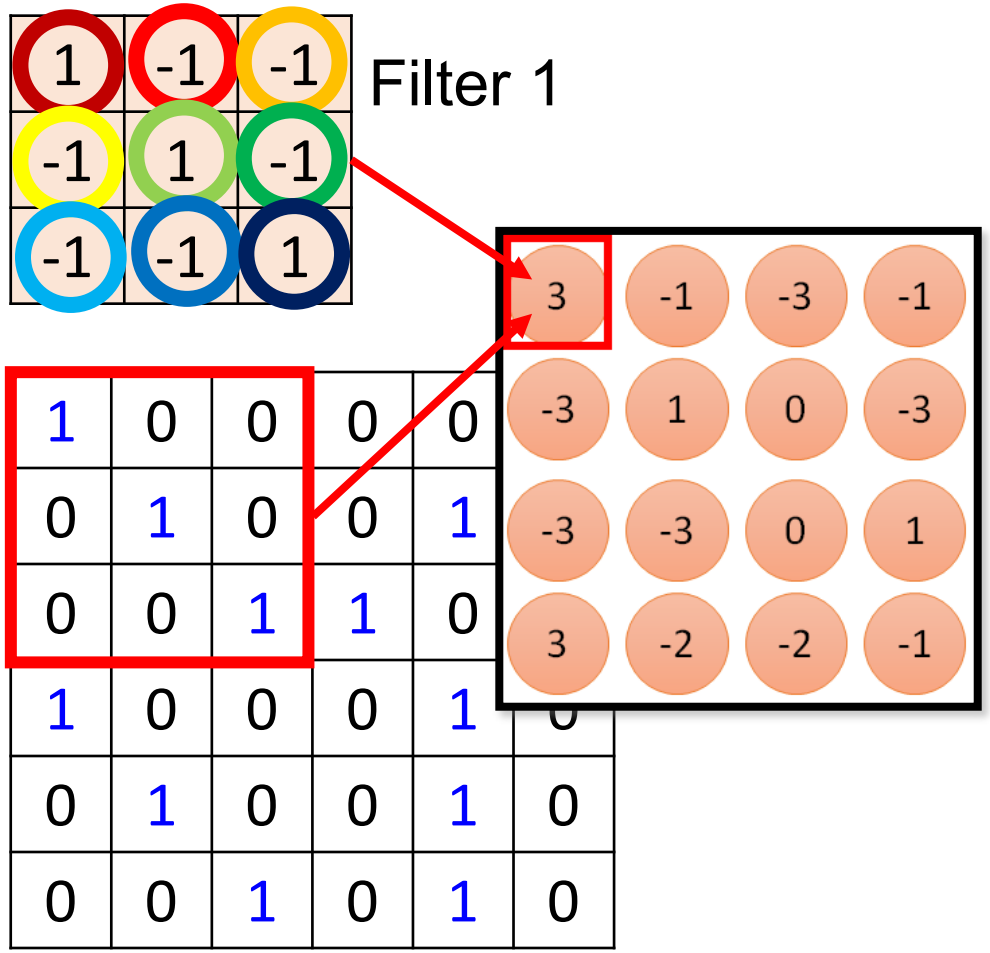
Filter of size 4x4 : 16 different weights
Apply this same filter to 4x4 patches in input

Parameter sharing

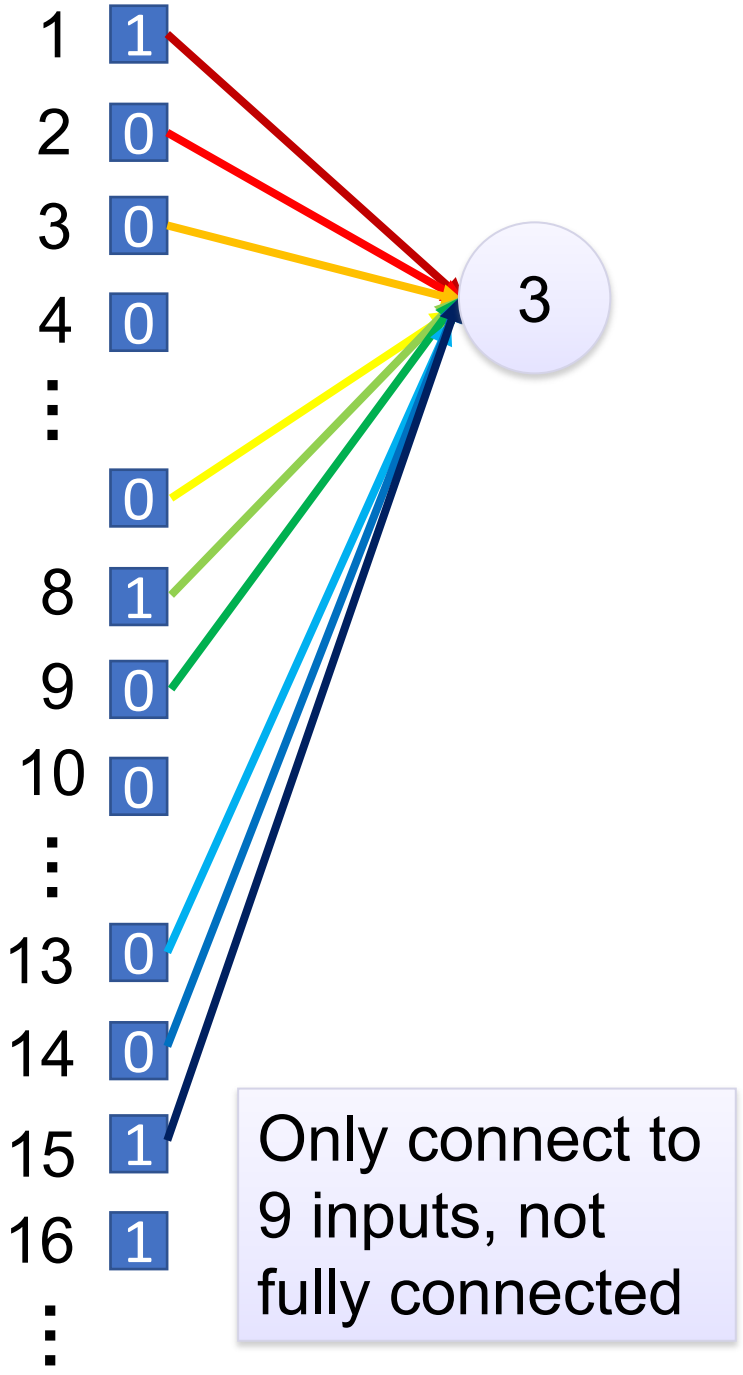




Sparsity of Connections

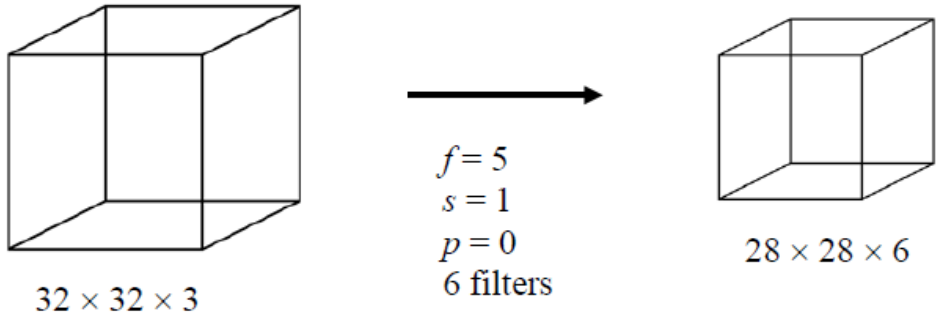


fewer parameters!



Only connect to 9 inputs, not fully connected

Sparsity of Connections



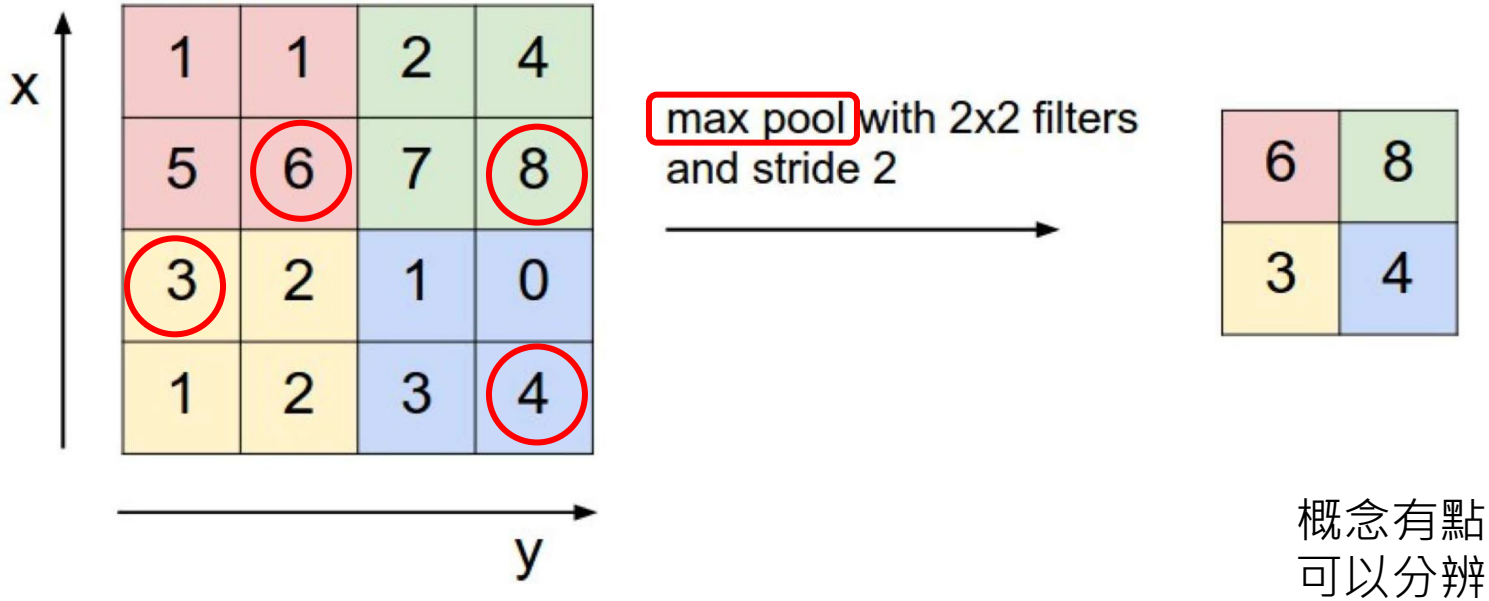
- Example:
- Fully connected layer:
  - Number of neurons in the input layer = 32*32*3 = 3072
  - Number of neurons in the output layer = 28*28*6 = 4704
  - Number of parameters = 4704 * 3072 + 4704 (biases) $\cong 14*10^6$
- Convolution layer:
  - Number of parameters = (5*5*3 + 1) * 6 = 456

$\Rightarrow$ Convolutional layers reduce the number of parameters and speed up the training of the model significantly.

Pooling Layer 池化層

Goal：保留原來的資訊量，
降低運算的時間

Pooling layers are generally used to reduce the size of the inputs and hence speed up the computation(減少輸入的規模來加速運算的時間)



1) Reduced dimensionality (減少維度)
2) Spatial invariance (空間不變性)

概念有點像：照片不需要解析度那麼高，就可以分辨出那個是一棵樹

Pooling Layer

如果是辨識痣，如果是用平均pooling，資訊會被平均，
痣的資訊會不明顯，所以用max pooling

- We may apply average pooling(平均的) instead of max pooling(最大的). However, the max pooling is more often used than the average pooling.
- In practice, we usually apply pooling without padding(通常padding為0).
- In an RGB image of shape $n_h \times n_w \times n_c$, we perform max pooling independently for each channel.
- If the input of the pooling layer is $n_h \times n_w \times n_c$, then the output will be:

$$\left[\frac{n_h - f}{s} + 1\right] \times \left[\frac{n_w - f}{s} + 1\right] \times n_c$$

做完pooling，channel的數目必須保持一樣

Why Pooling?

- Down sample and preserve spatial invariance
- 一個壓縮圖片並保留重要資訊的方法

bird

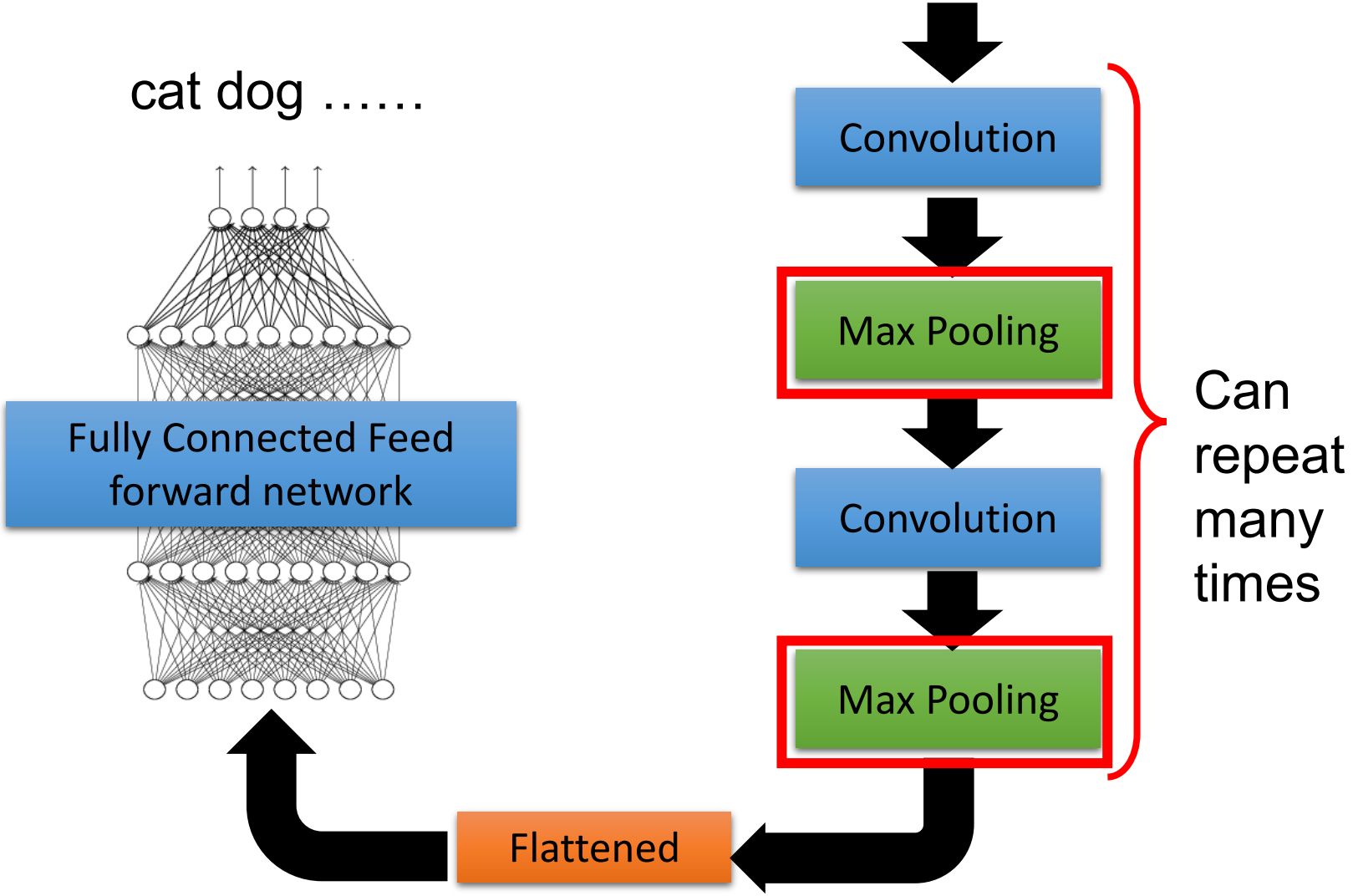
bird

We can subsample the pixels to make image smaller

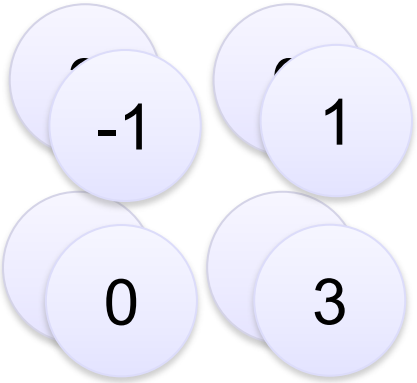
⇒Fewer parameters to characterize the image

⇒Speed up the computation

The whole CNN



The whole CNN



A new image

Convolution

Max Pooling

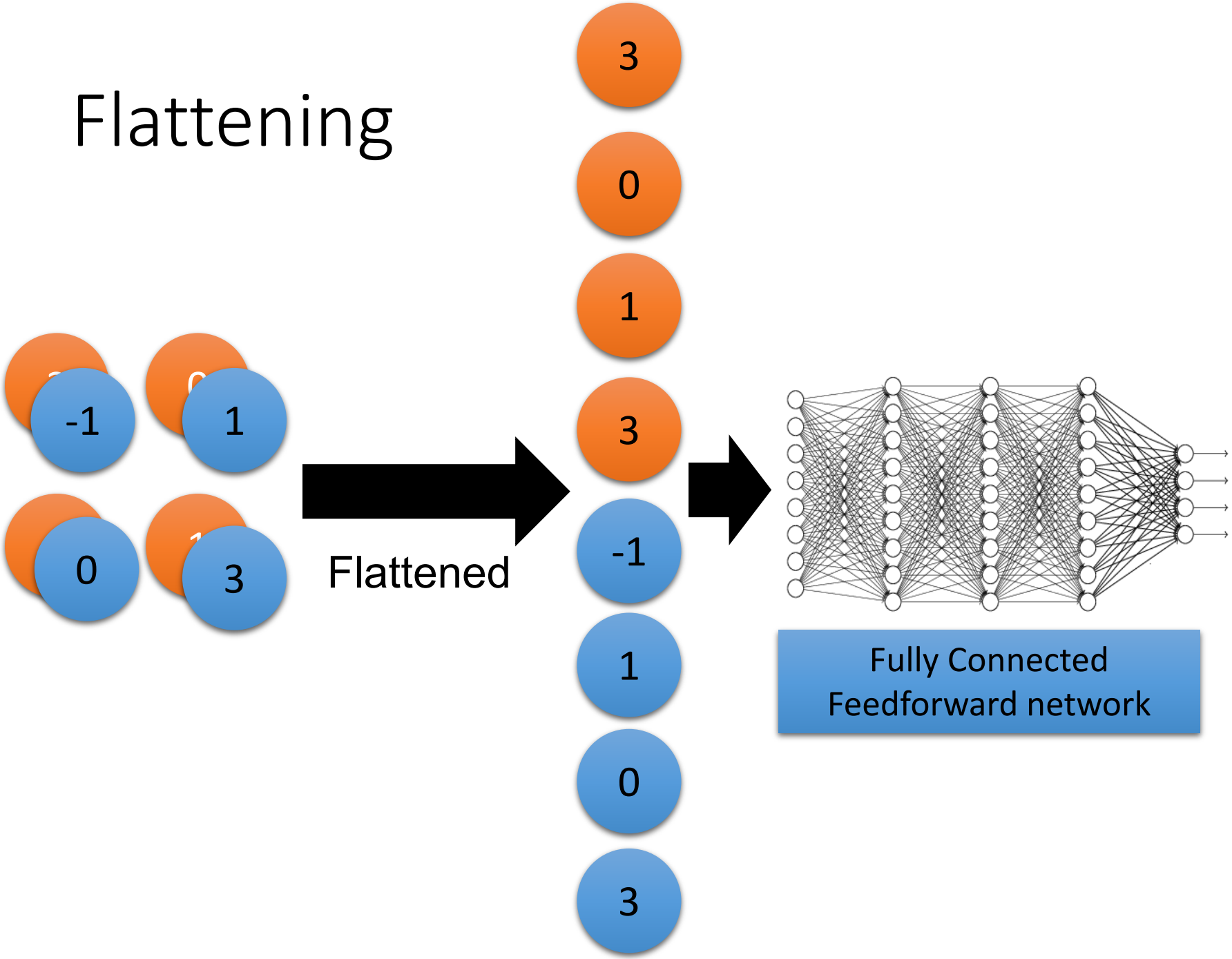
Convolution

Max Pooling

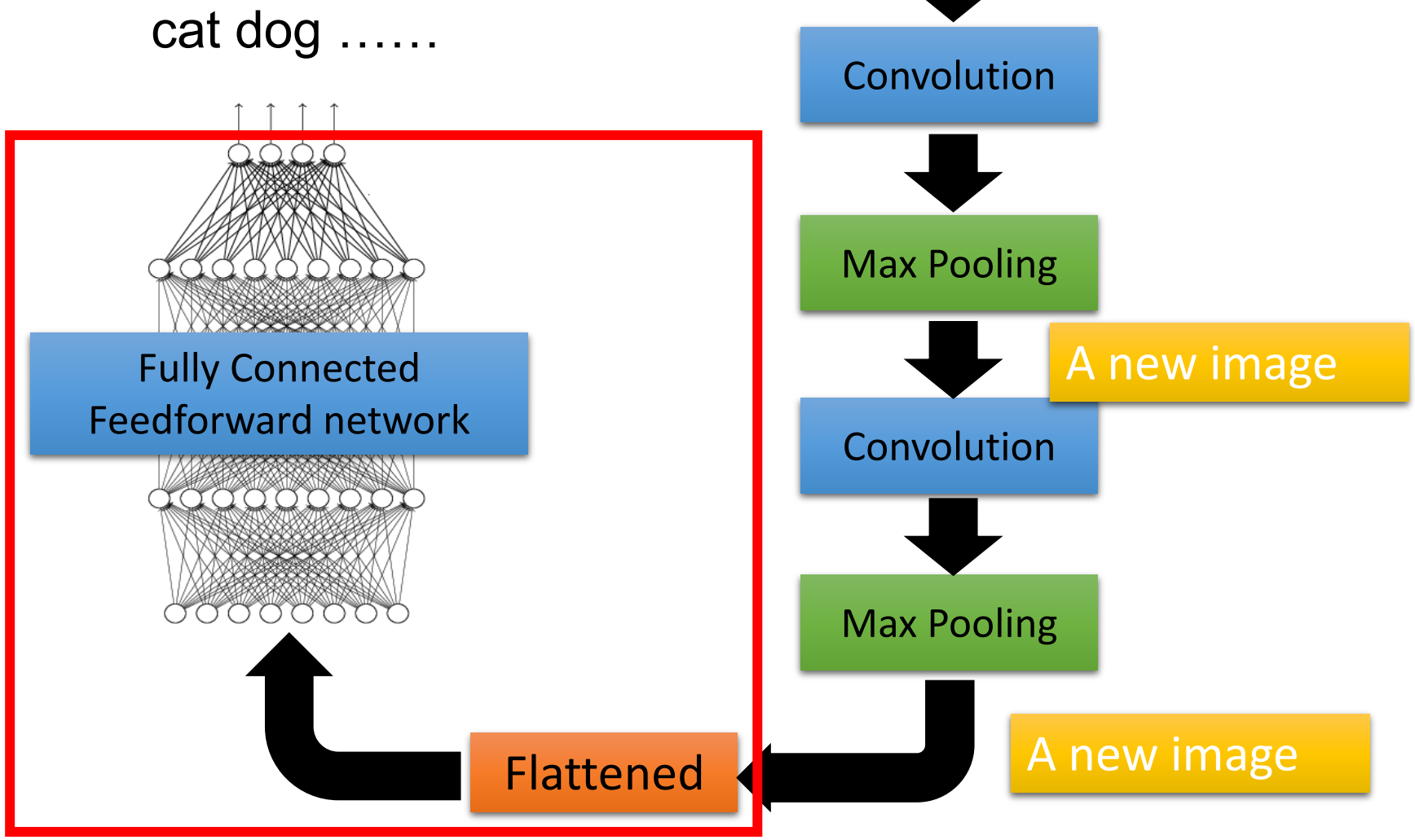
Can repeat many times

Smaller than the original image

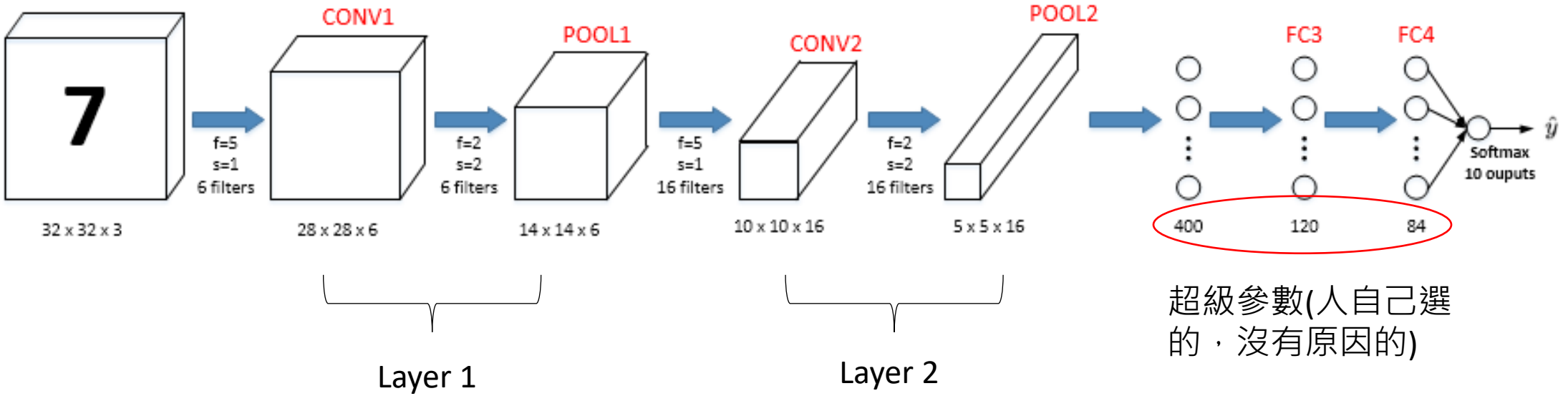
Flattening


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

The whole CNN



Convolutional Neural Network Example



超級參數(人自己選的，沒有原因的)

Convolutional Neural Network Example

	Activation Shape	Activation Size	# Parameters
Input	(32, 32, 3)	3,072	0
CONV1 (f =5, s = 1)	(28, 28, 6)	4,704	456
POOL1 (f =2, s = 2)	(14, 14, 6)	1,176	0
CONV2 (f =5, s = 1)	(10, 10, 16)	1,600	2,416
POOL2 (f =2, s = 2)	(5, 5, 16)	400	0
FC3	(120, 1)	120	48,120 (=120*400+120)
FC4	(84, 1)	84	10,164
Softmax	(10, 1)	10	850

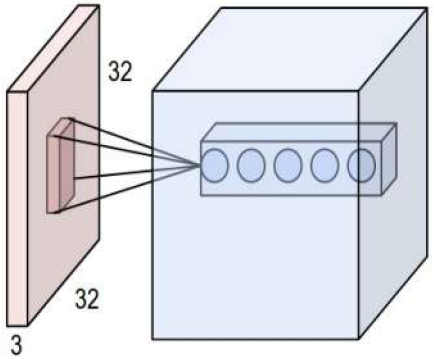
Deep Learning for Computer Vision: Summary

Foundations

- Why computer vision?
- Representing images
- Convolutions for feature extraction

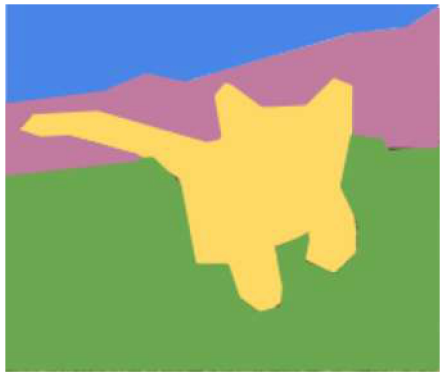
CNNs

- CNN architecture
- Application to classification
- ImageNet



Applications

- Segmentation, object detection, image captioning
- Visualization



Resources

Stanford CS230: Deep Learning

Stanford CS231n: Convolutional Neural Networks for Visual Recognition

MIT 6.S191: Introduction to Deep Learning