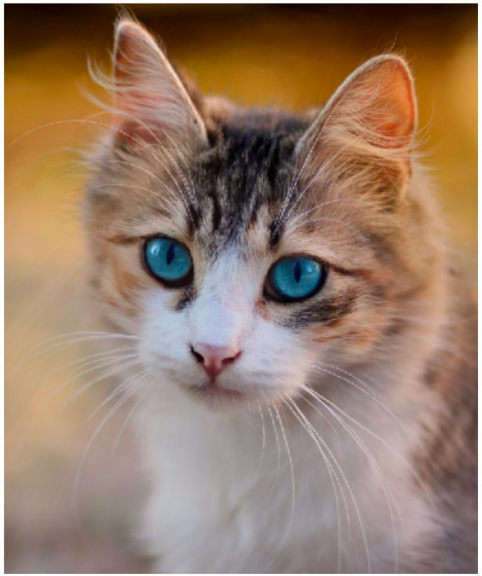
Convolutional Neural Network

CNN介紹

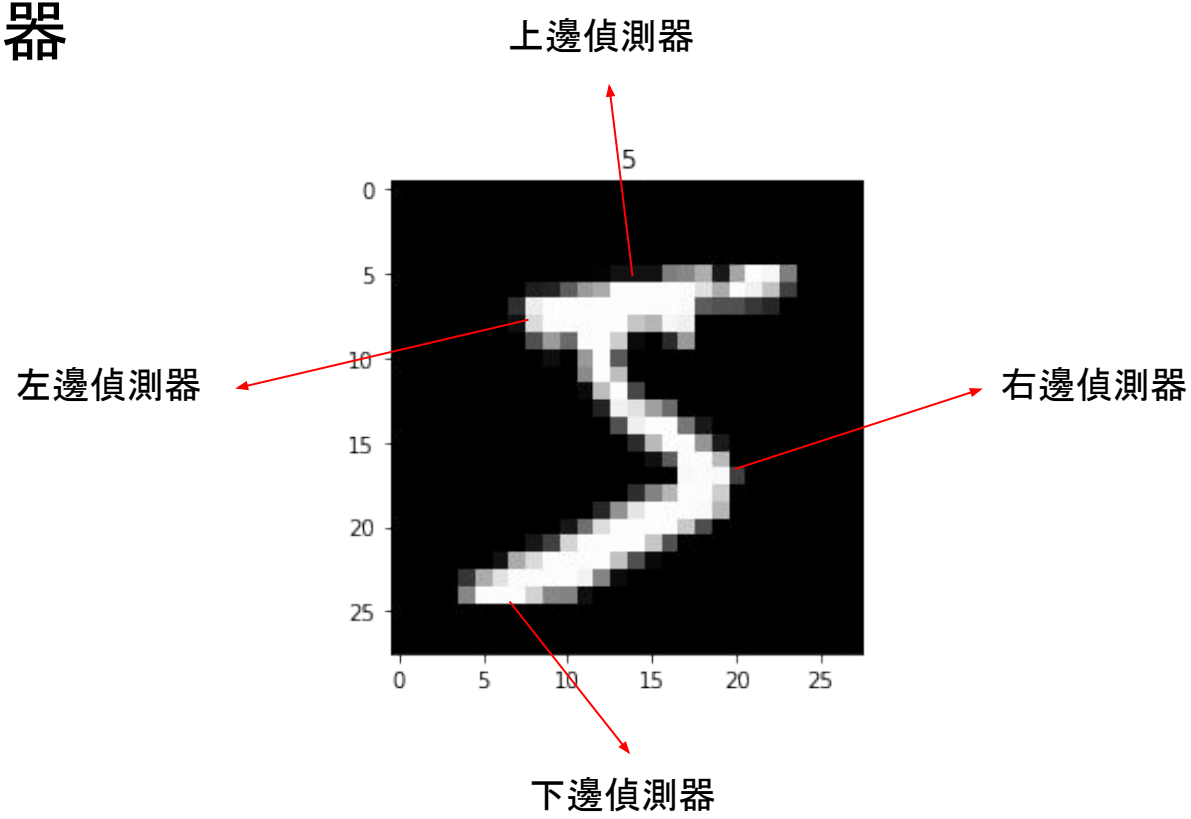


我們如何分辨

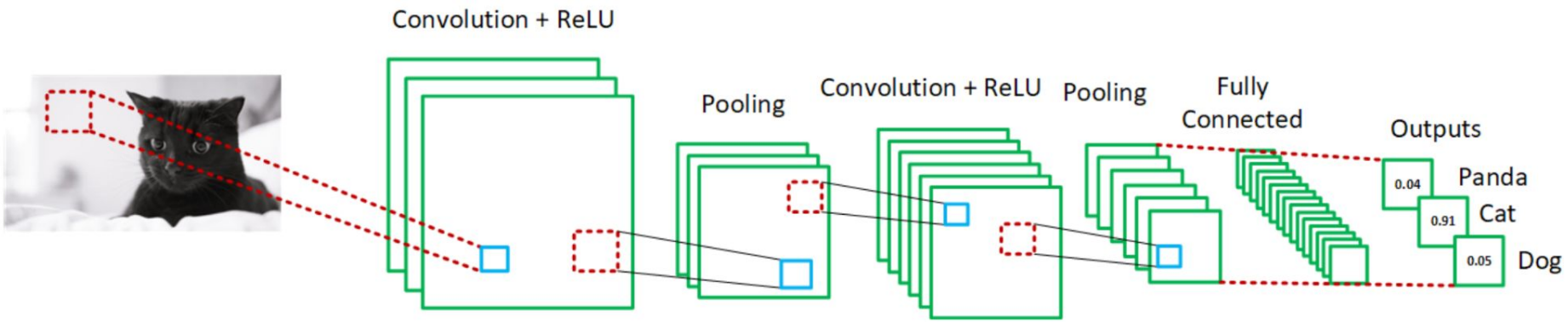
- 許多特徵能讓我們分辨狗和貓
- 鬍鬚、臉型、眼睛、毛、顏色等等
- 模型可以分辨這些特徵嗎？

CNN

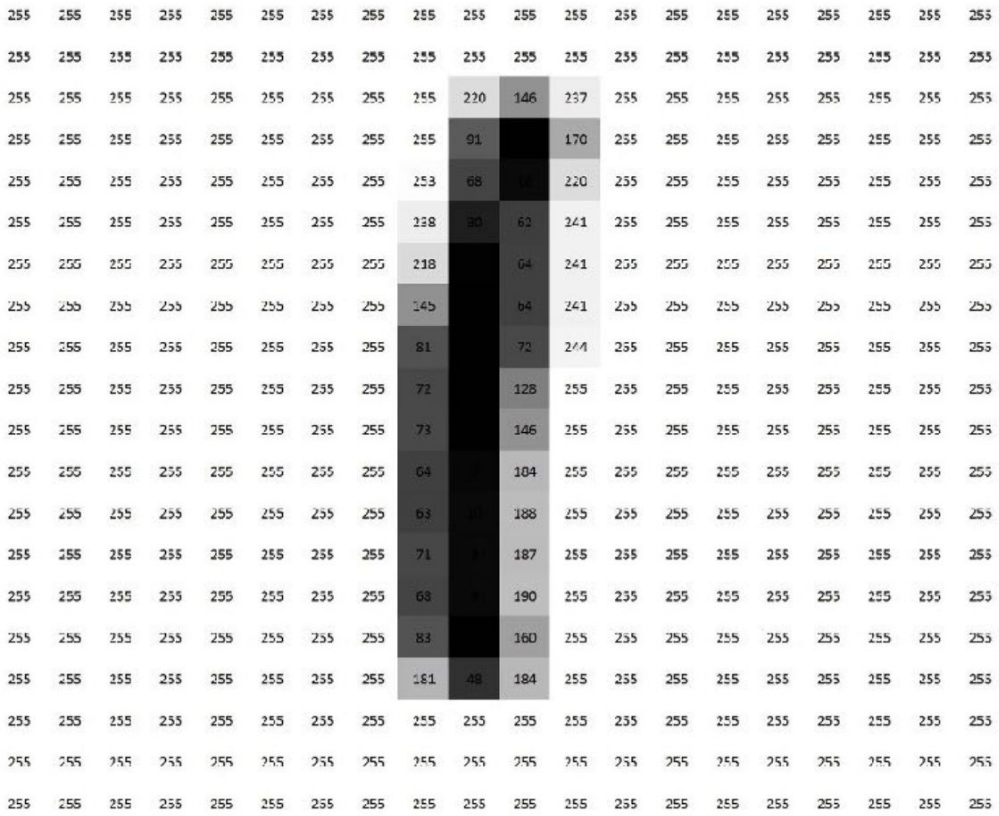
特徵偵測器



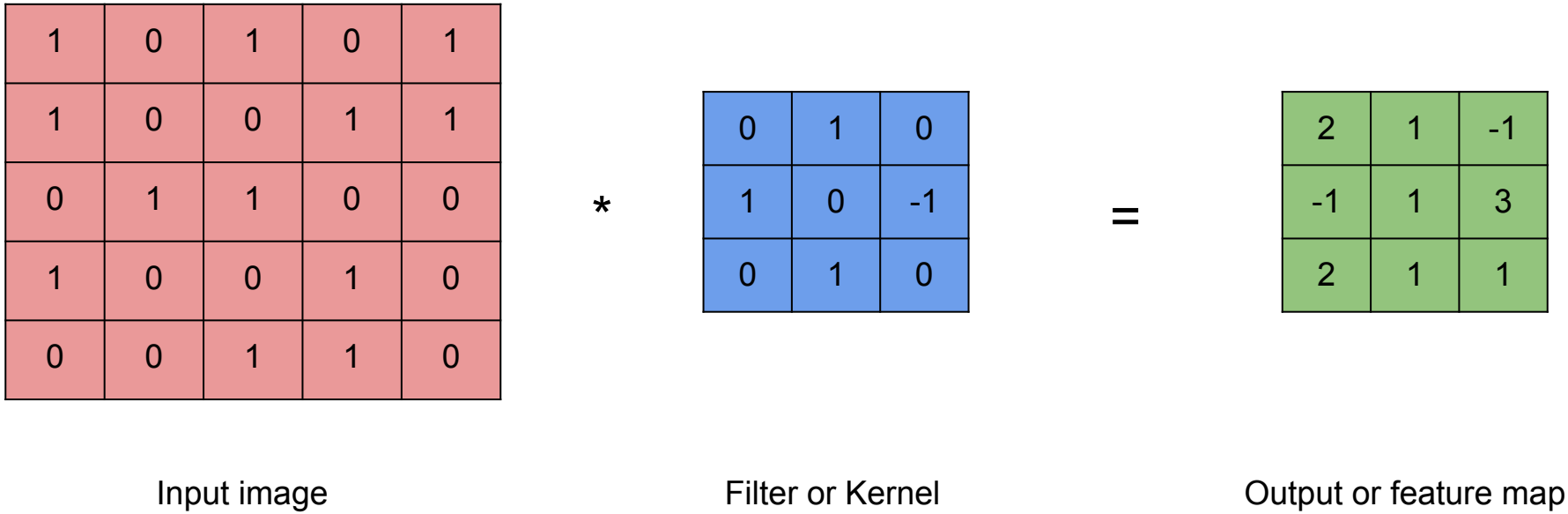
用CNN預測


Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
Fully Connected
Outputs
0.04
Panda
0.91
Cat
0.05
Dog

Convolution (卷積)


255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 220 146 237 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 91 170 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 253 68 220 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 238 80 62 241 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 218 64 241 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 145 64 241 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 81 72 244 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 72 128 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 73 146 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 64 104 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 63 188 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 71 187 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 68 190 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 83 160 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 181 48 184 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255

Convolution 運算



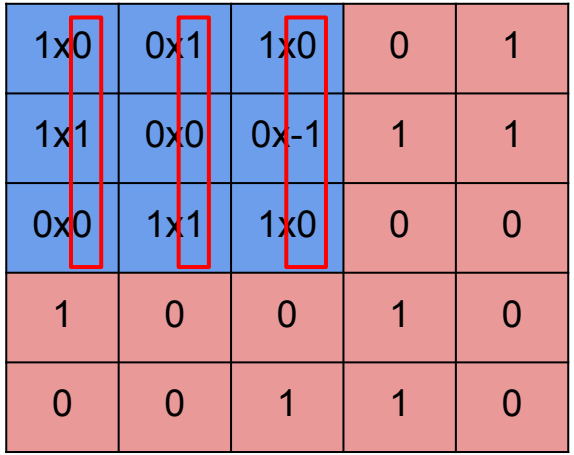
Input image

Filter or Kernel

Output or feature map

Convolution 運算

$$(1 \times 0) + (0 \times 1) + (1 \times 0) + (1 \times 1) + (0 \times 0) + (0 \times -1) + (0 \times 0) + (1 \times 1) + (1 \times 0) = 2$$



Input image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2		

Output or feature map

Convolution 運算

$$(0 \times 0) + (1 \times 1) + (0 \times 0) + (0 \times 1) + (0 \times 0) + (1 \times -1) + (1 \times 0) + (1 \times 1) + (0 \times 0) = 1$$

1	0x0	1x1	0x0	1
1	0x1	0x0	1x-1	1
0	1x0	1x1	0x0	0
1	0	0	1	0
0	0	1	1	0

Input image

*

0	1	0
1	0	-1
0	1	0

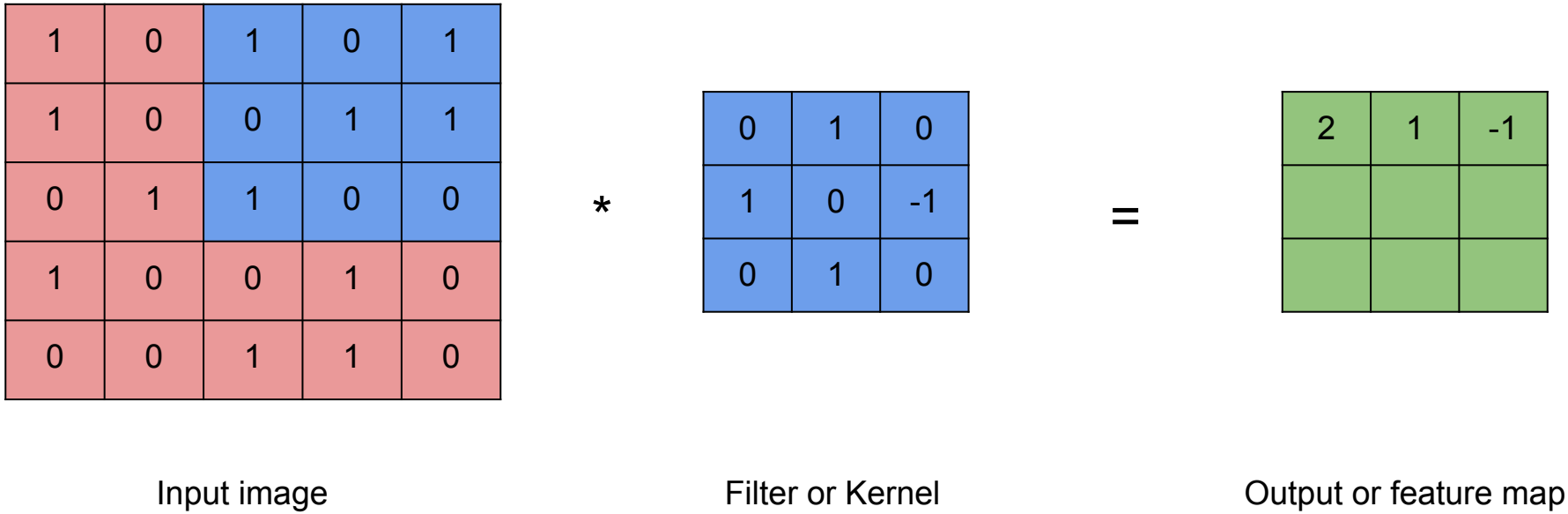
Filter or Kernel

=

2	1	

Output or feature map

Convolution 運算

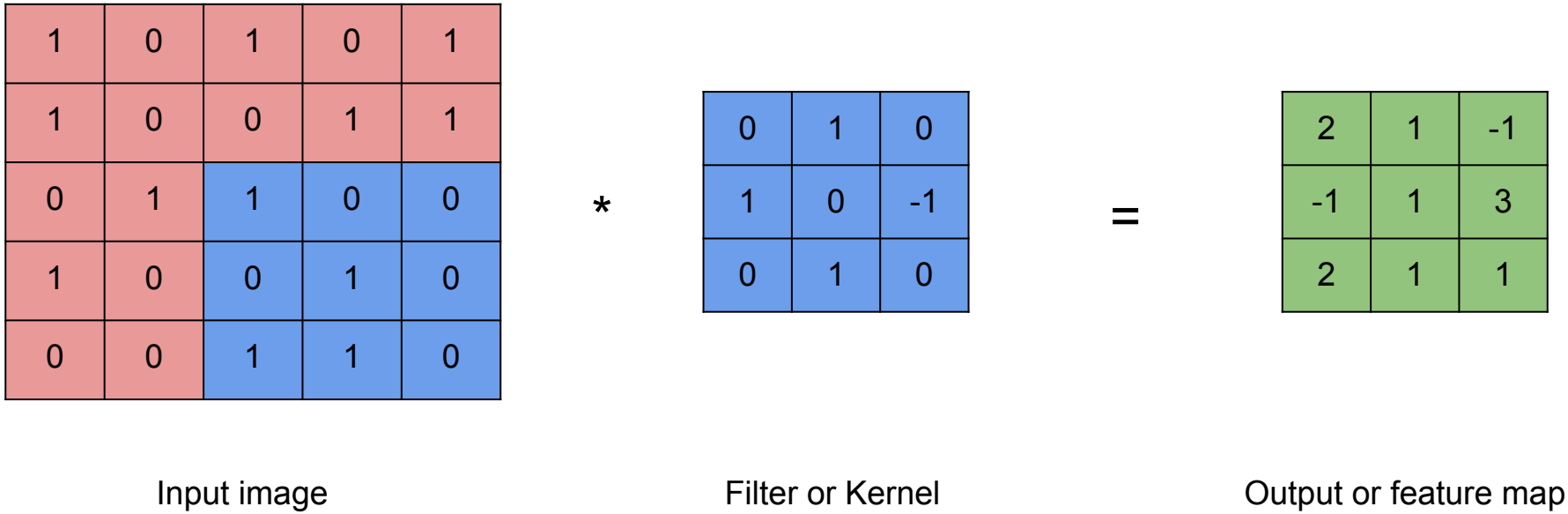


Input image

Filter or Kernel

Output or feature map

Convolution 運算



Input image

Filter or Kernel

Output or feature map

特徵偵測

- Filter or kernel 特徵偵測器
- Feature map = 偵測結果

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

= Vertical edge detector

Vertical edge detector

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

Input image

*

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

Filter or Kernel

=

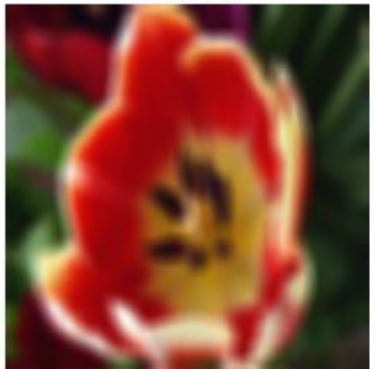
3	3	0
3	3	0
3	3	0

Output or feature map

各種 feature map

original

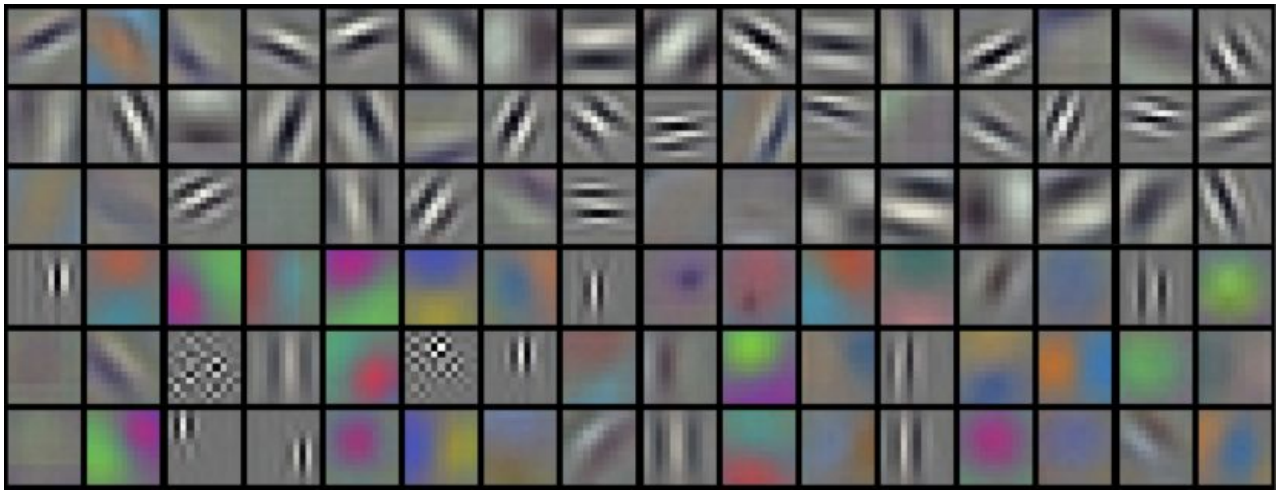
smooth



edge

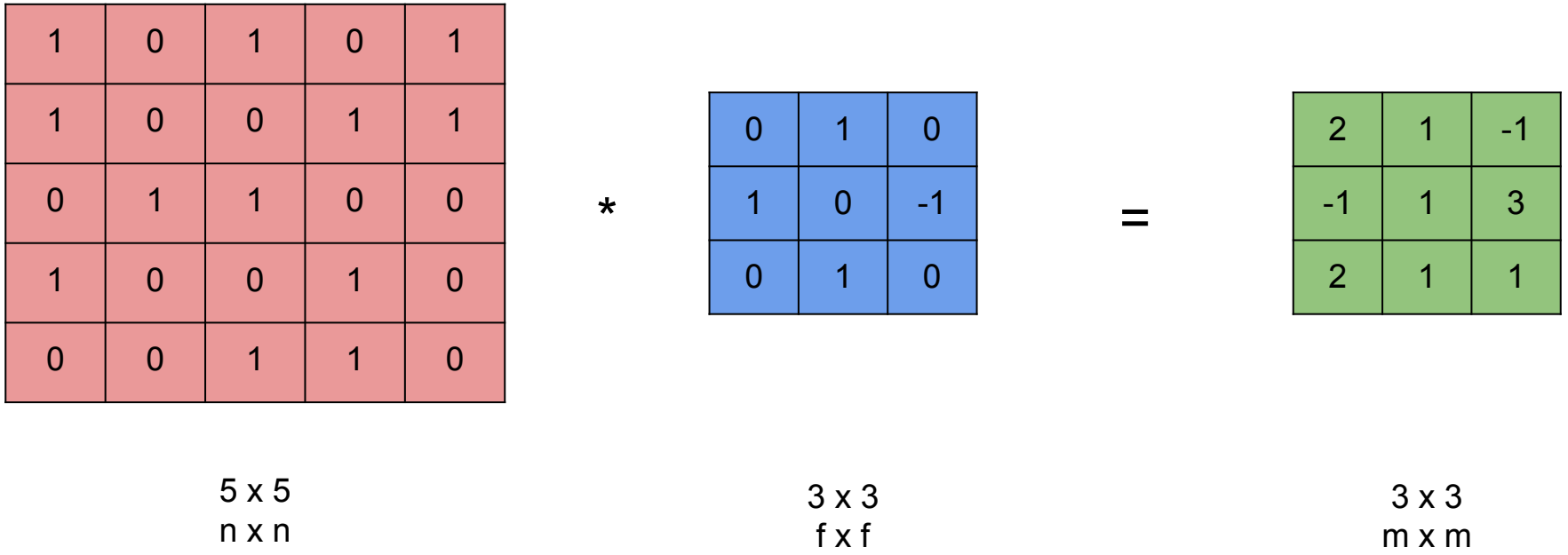
intensity

Filters 長怎樣？

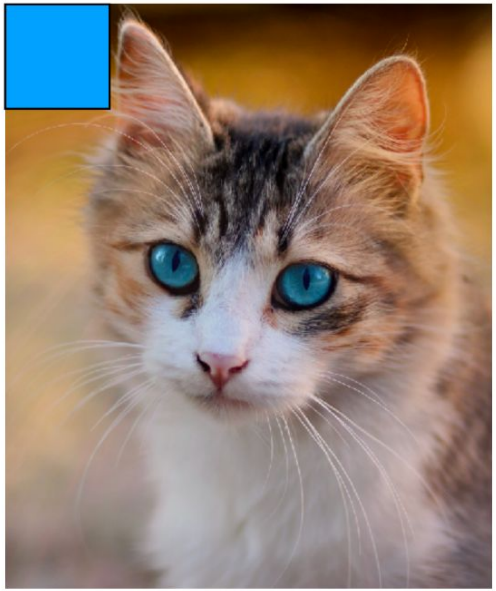


計算 feature map 的大小

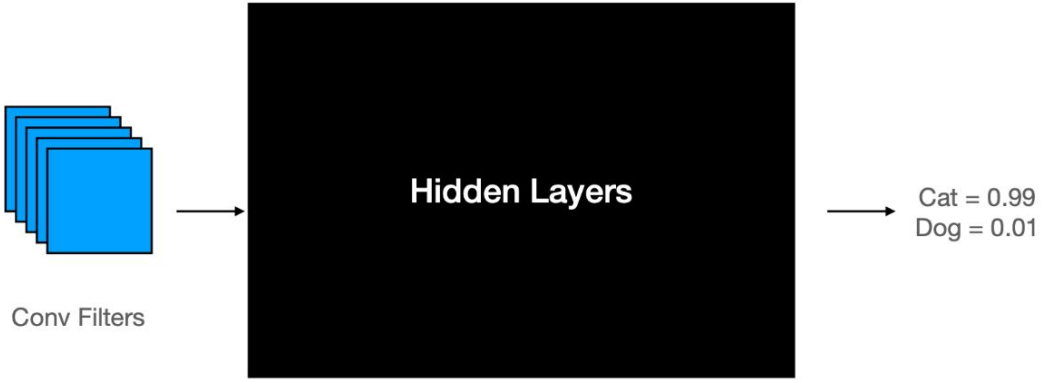
Feature Map Size = n − f + 1 = m
Feature Map Size = 5 − 3 + 1 = 3



如何分類 image

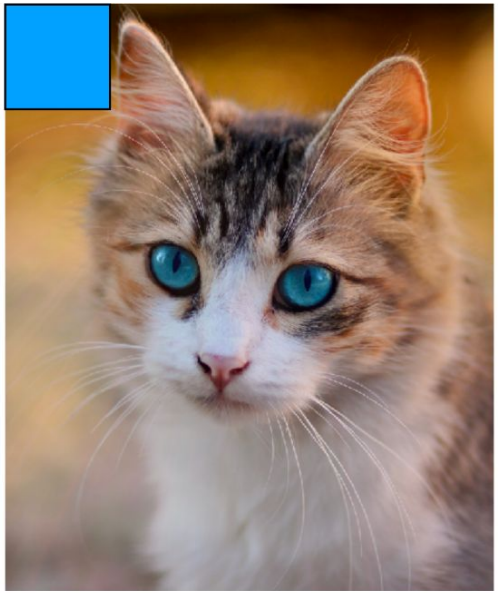


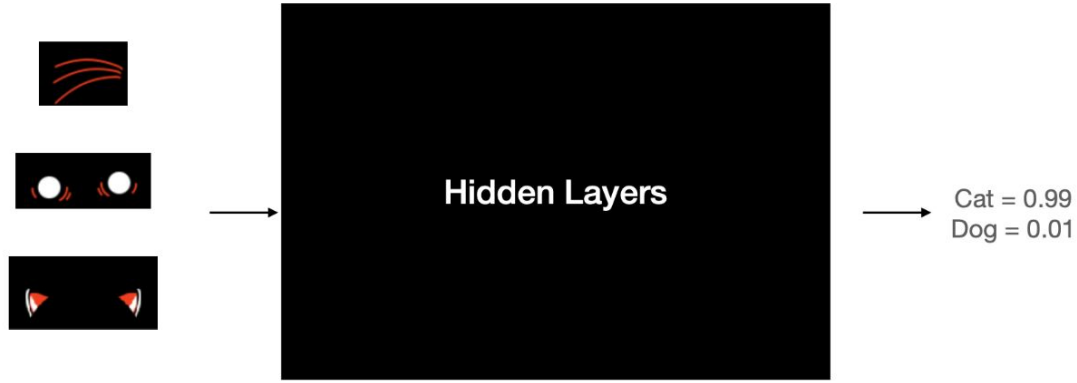
We perform Convolution operations with the input image and our filters



Uses the Feature Maps we output to determine the image class

如何分類 image





Conv Filters

Uses the Feature Maps we output to determine the image class

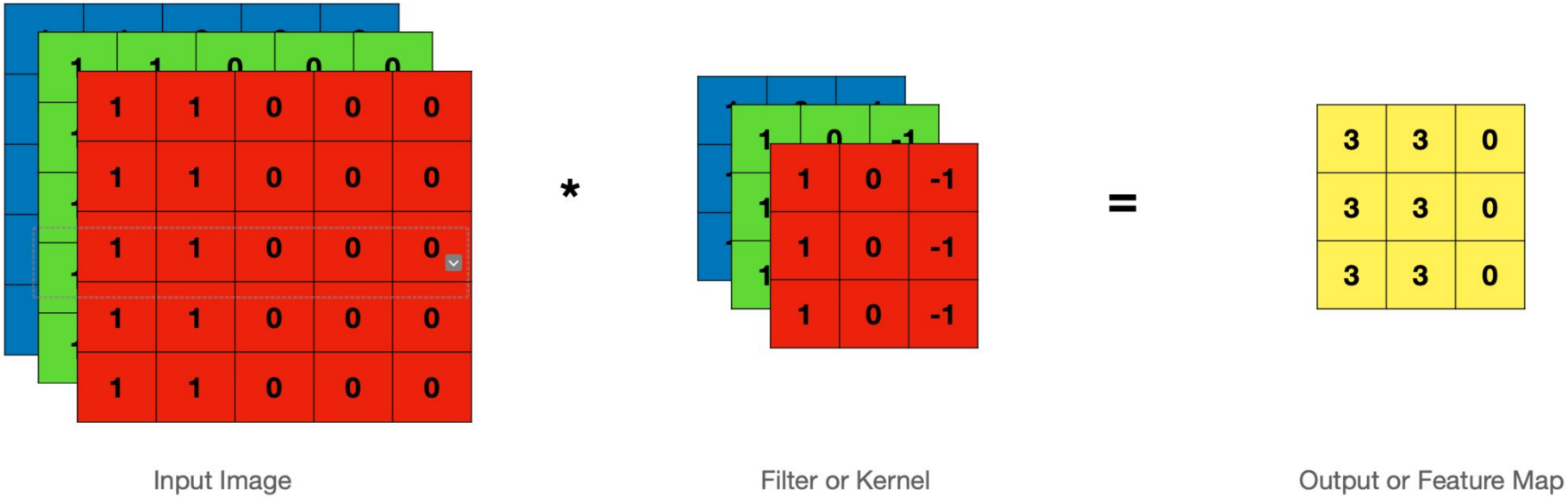
We perform Convolution operations with the input image and our filters

Convolution demo

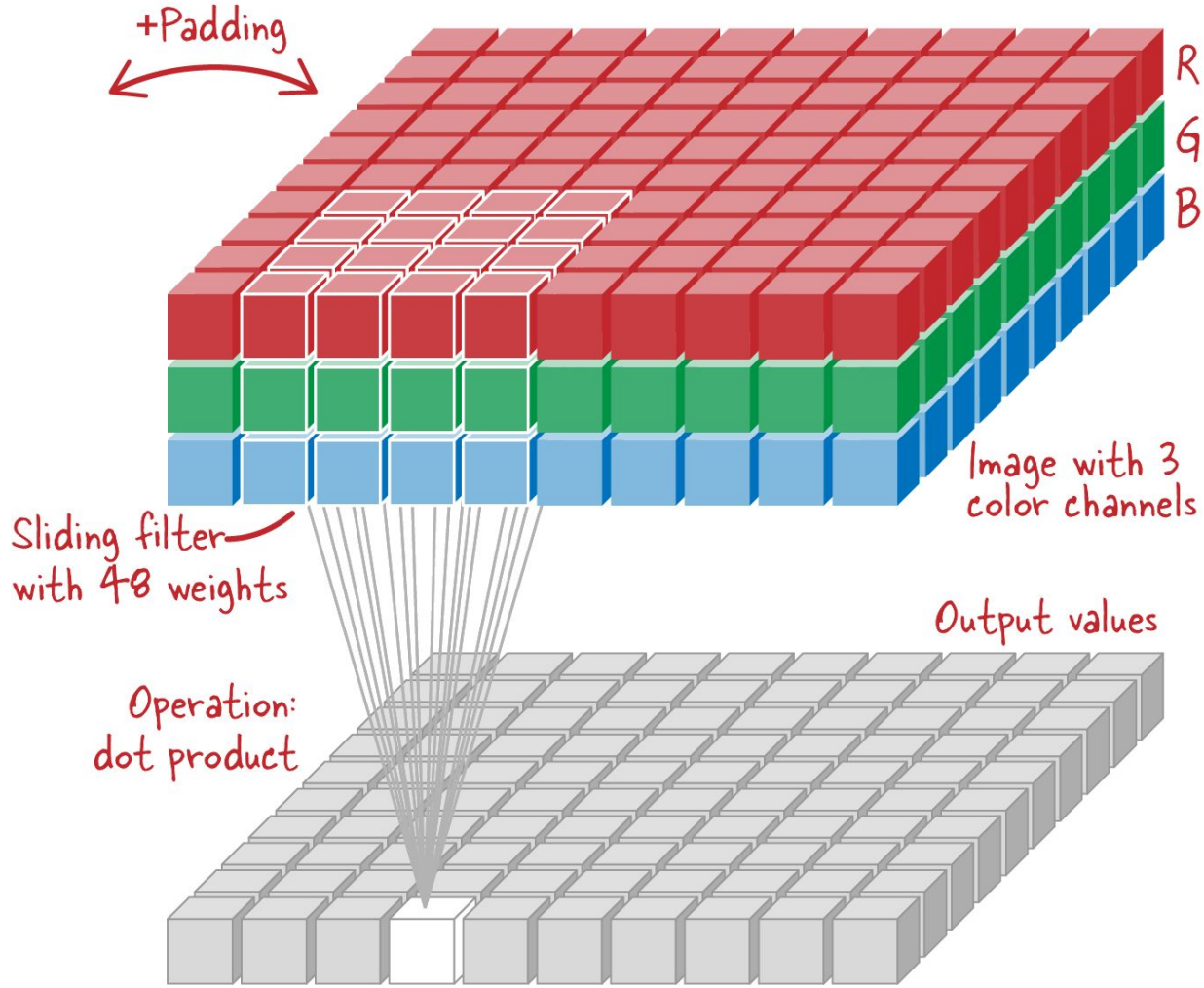
https://deeplizard.com/learn/video/YRhxdVk_sIs

Convolution on Color Images

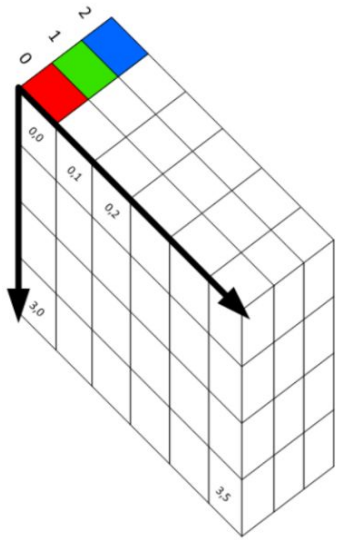
如何計算彩色 image 的 convolution



我們可以調整不同頻道的 filter 來關注某個顏色的特徵

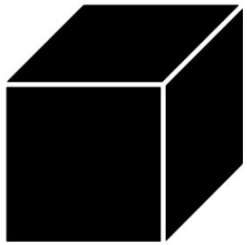

+Padding
R
G
B
Image with 3
color channels
Sliding filter
with 48 weights
Output values
Operation:
dot product

計算 feature map size



Input Image
5 x 5 x **3**

*



Filter or Kernel
3 x 3 x **3**

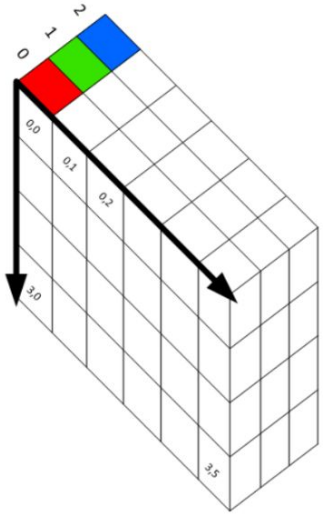
=

3	3	0
3	3	0
3	3	0

Output or Feature Map
3 x 3

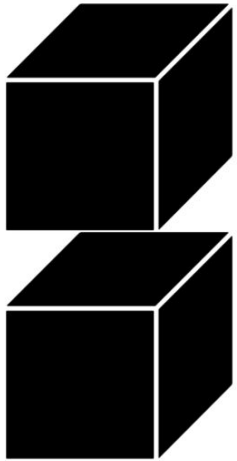
計算 feature map size

$$(n \times n \times nc) * (f \times f \times nc) = (n - f + 1) \times (n - f + 1) \times nf$$
$$(5 \times 5 \times 3) * (3 \times 3 \times 3) = 3 \times 3 \times 2$$



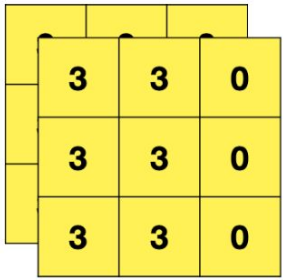
Input Image
5 x 5 x 3

*



2 Filters or Kernels
3 x 3 x 3 x 2

=



Output or Feature Map
3 x 3 x 2

Parameters for convolution filter

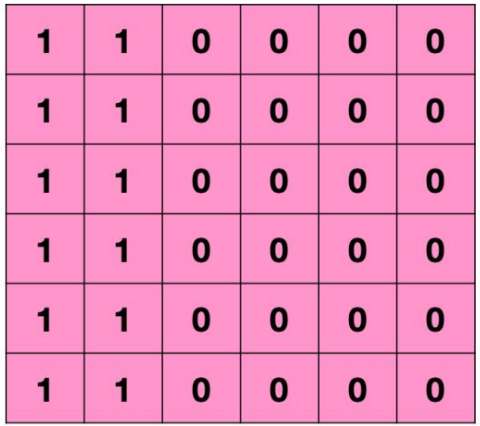
Convolution filter 有哪些超參數

- Kernel size (k x k)
- Depth
- Padding
- Stride

Kernel size

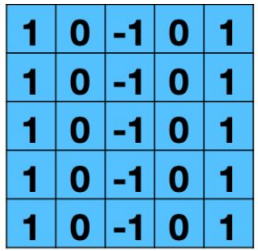
Feature Map Size = $n - f + 1 = m$

Feature Map Size = $6 - 5 + 1 = 2$



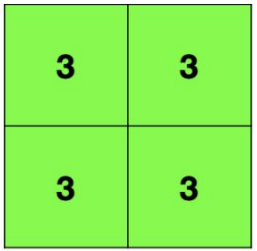
6 x 6

*



5 x 5

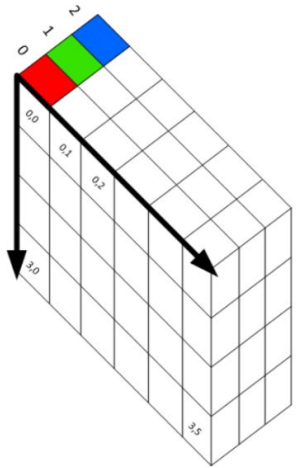
=



Output or Feature Map

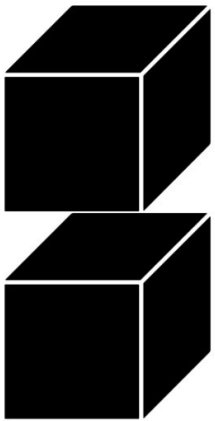
Depth

- Channels
- Filters / kernels



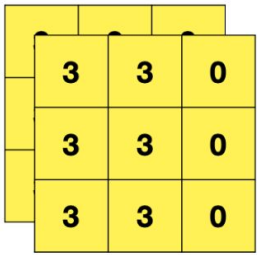
Input Image
5 x 5 x 3

*



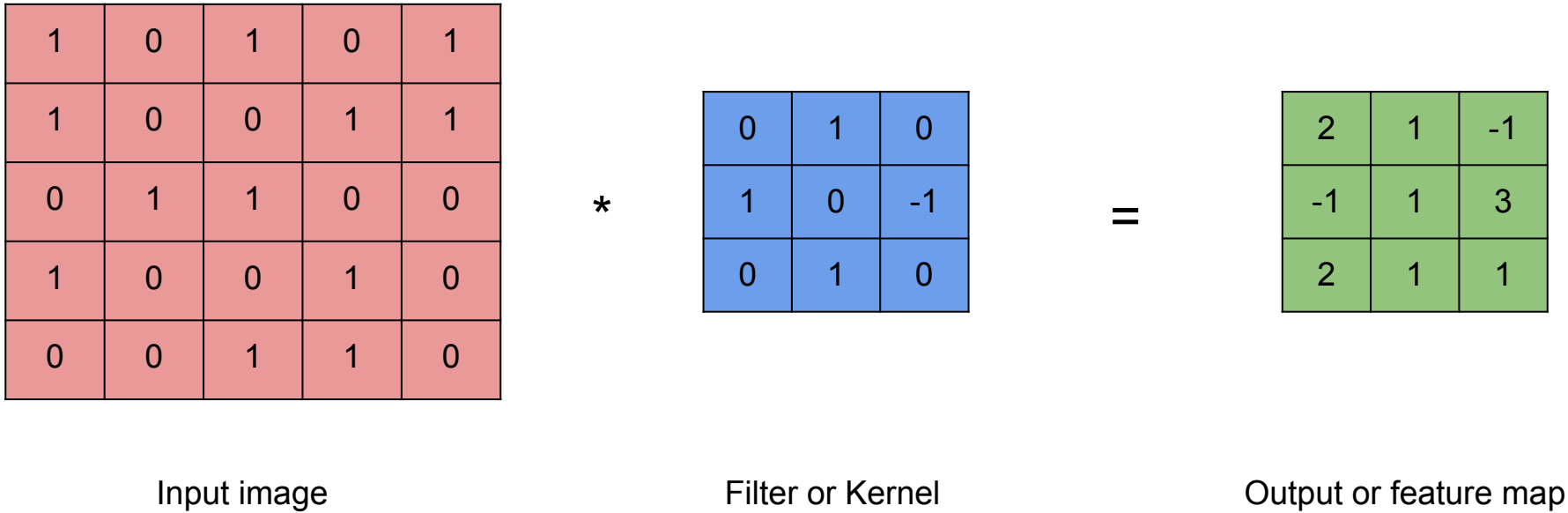
2 Filters or Kernels
3 x 3 x 3 x 2

=

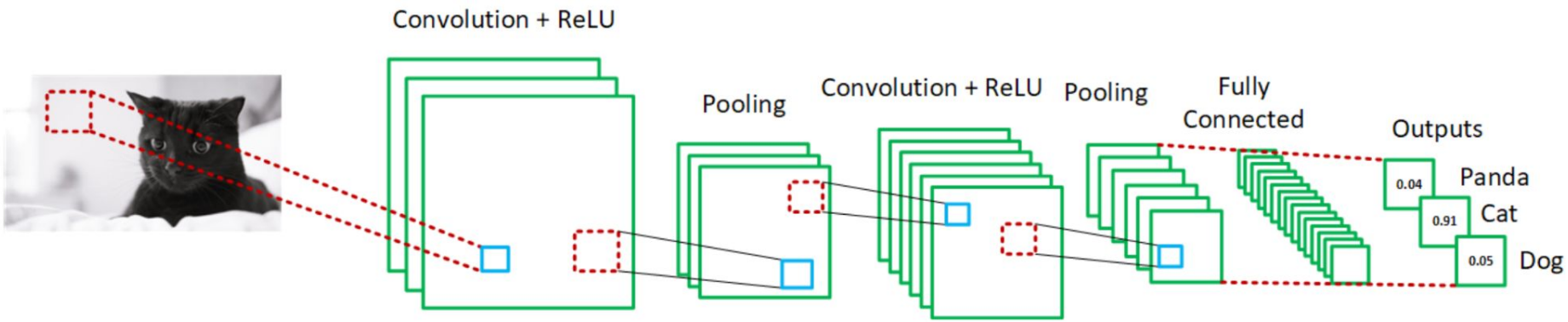


Output or Feature Map
3 x 3 x 2

Padding


1 0 1 0 1
1 0 0 1 1
0 1 1 0 0
1 0 0 1 0
0 0 1 1 0
*
0 1 0
1 0 -1
0 1 0
=
2 1 -1
-1 1 3
2 1 1
Input image
Filter or Kernel
Output or feature map

隨著convolution次數增加.....



如何避免image變太小

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

Input image

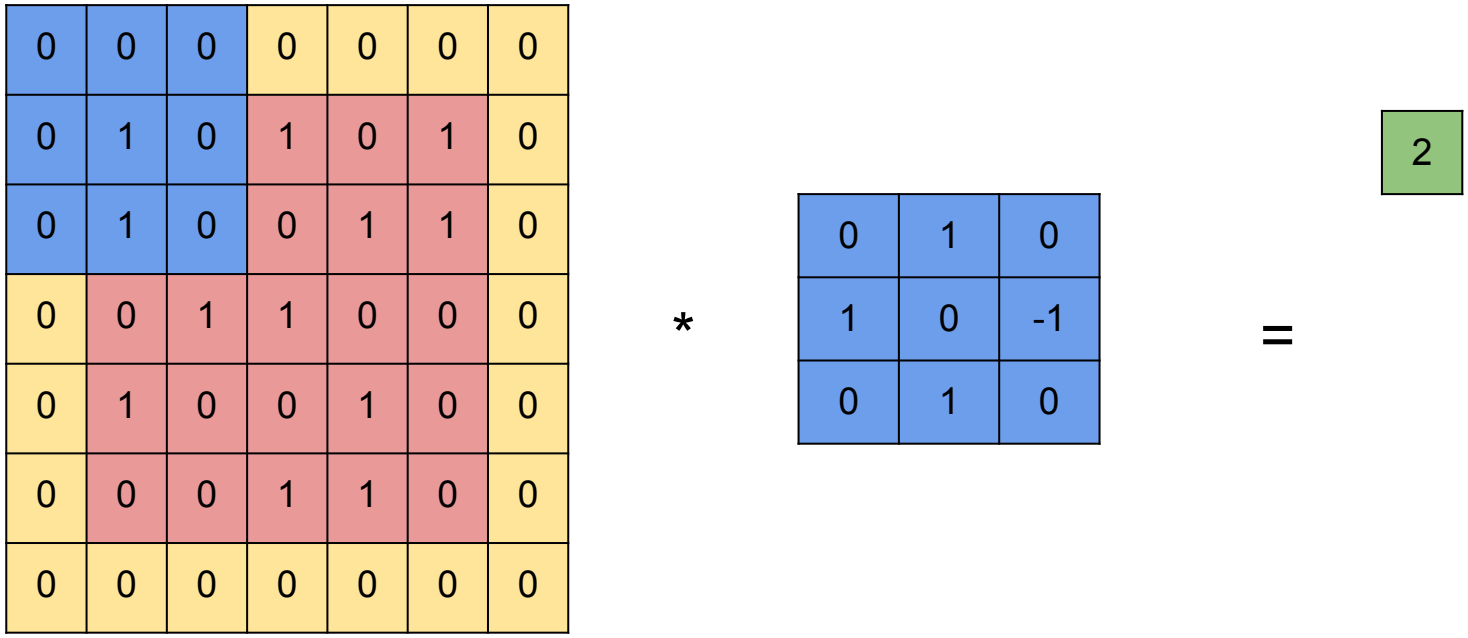
*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

如何避免image變太小

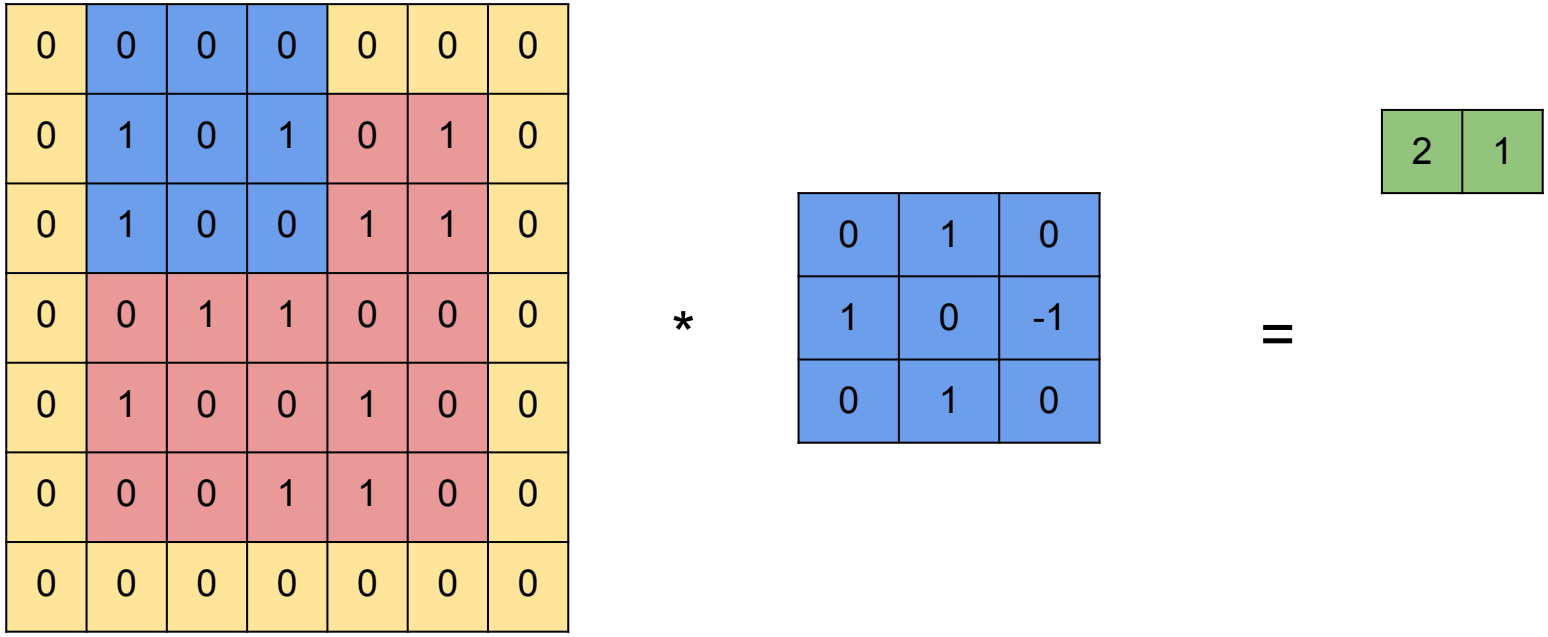


Input image

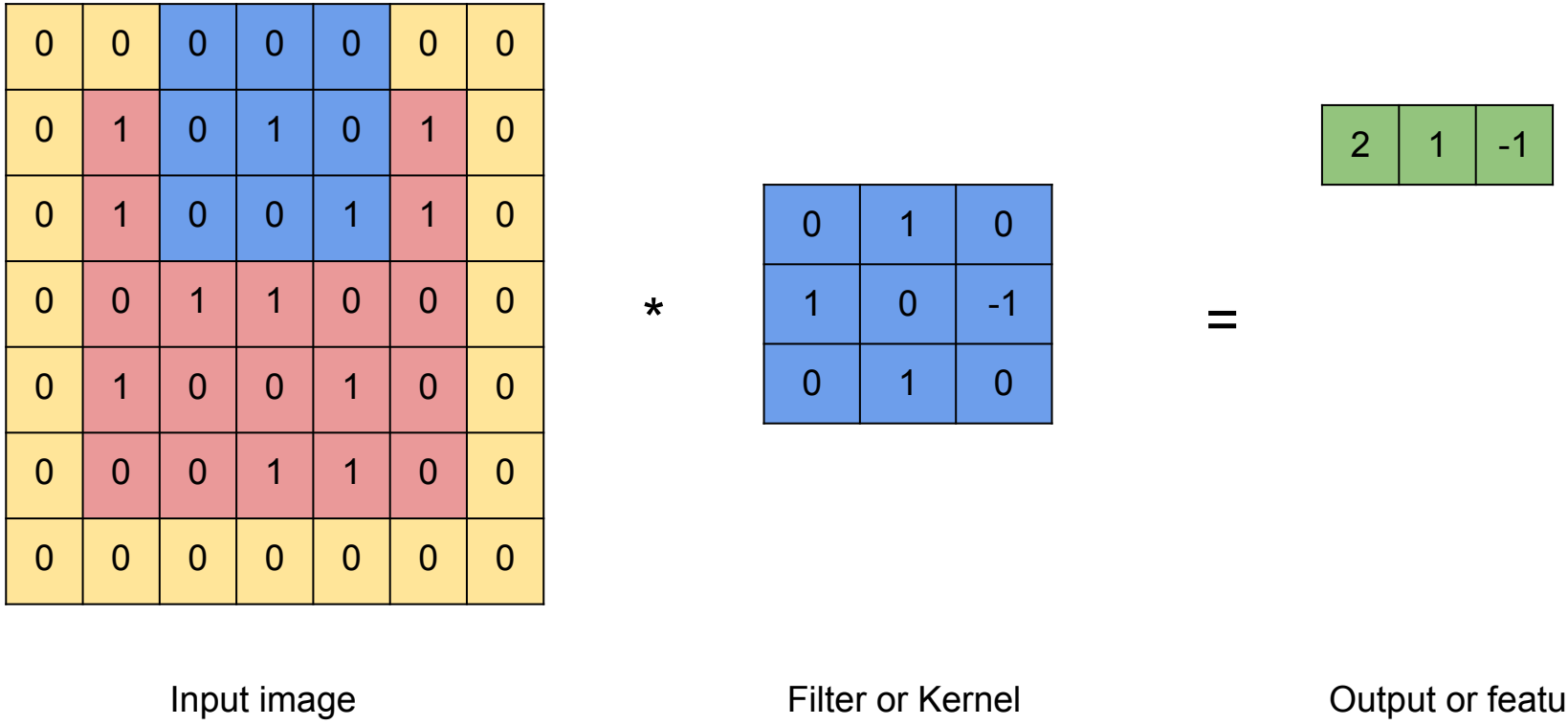
Filter or Kernel

Output or feature map

如何避免image變太小

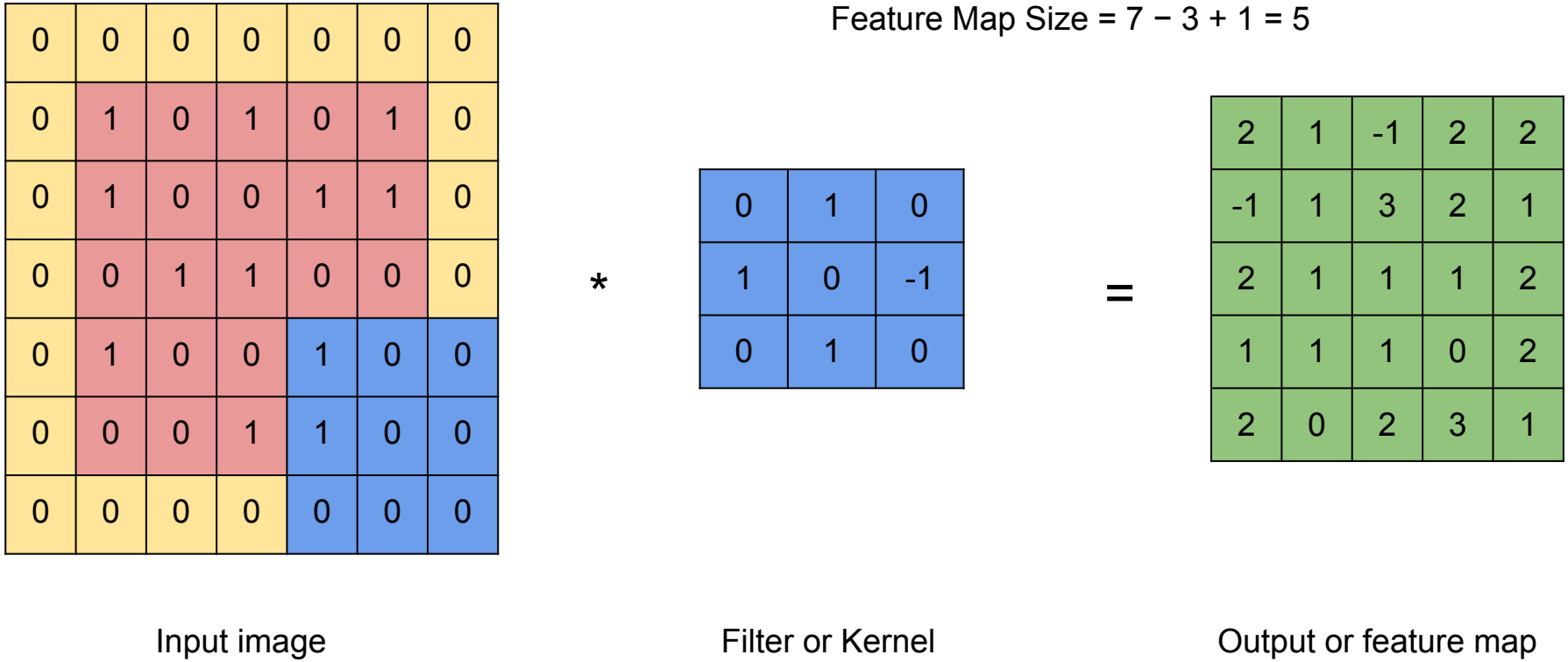


如何避免image變太小



如何避免image變太小

Feature Map Size = n − f + 1 = m
Feature Map Size = 7 − 3 + 1 = 5



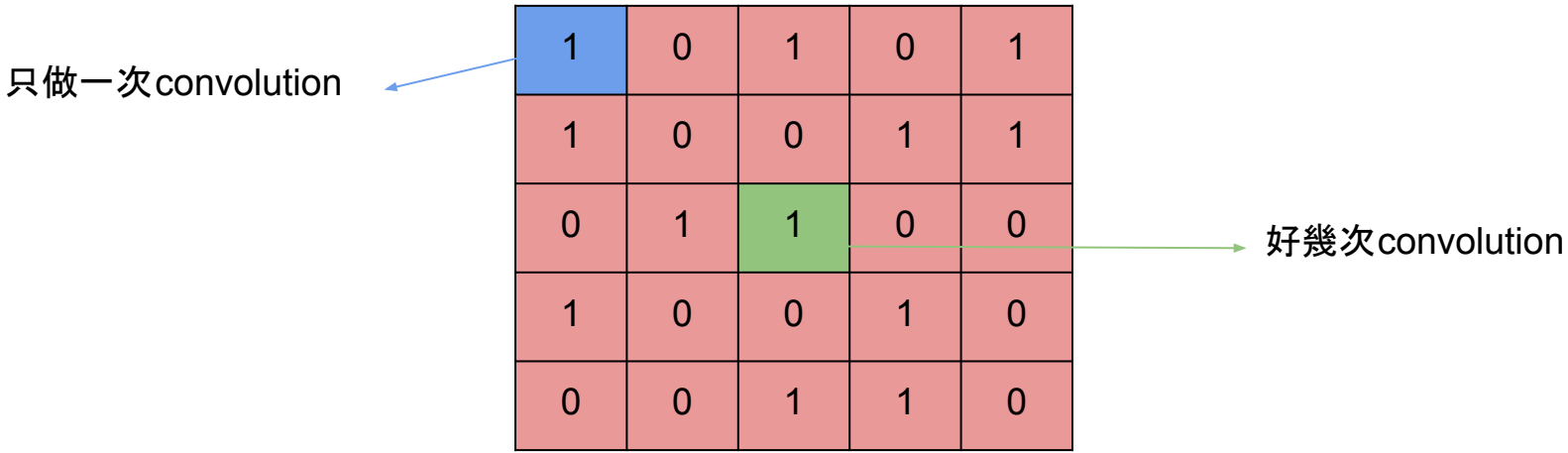
Input image

Filter or Kernel

Output or feature map

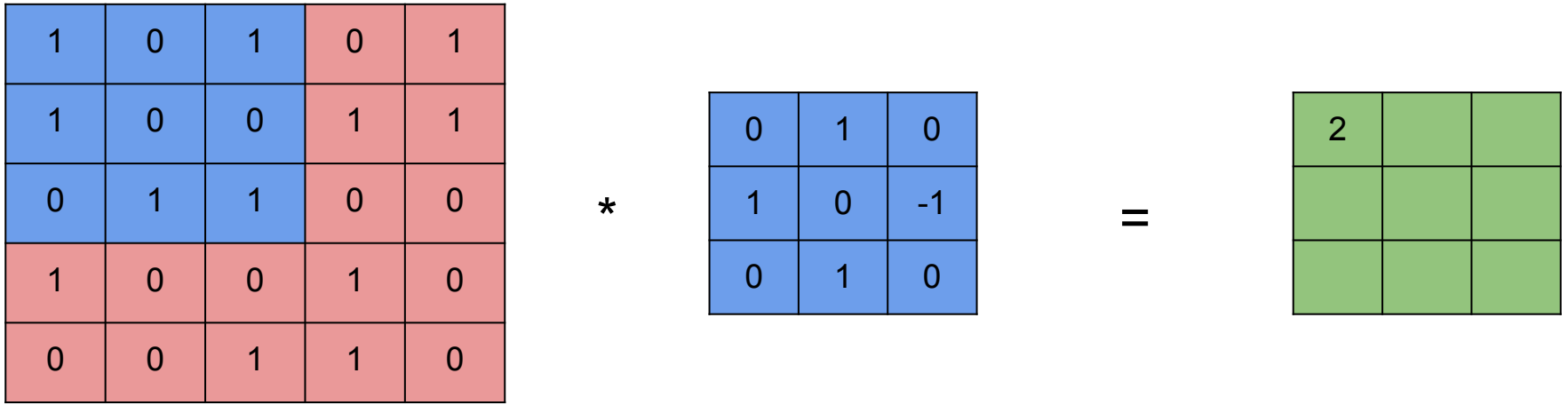
為什麼需要padding?

- 當使用較深的模型時，避免feature map持續變小
- 增加角落的pixel的貢獻



Stride

當stride = 1



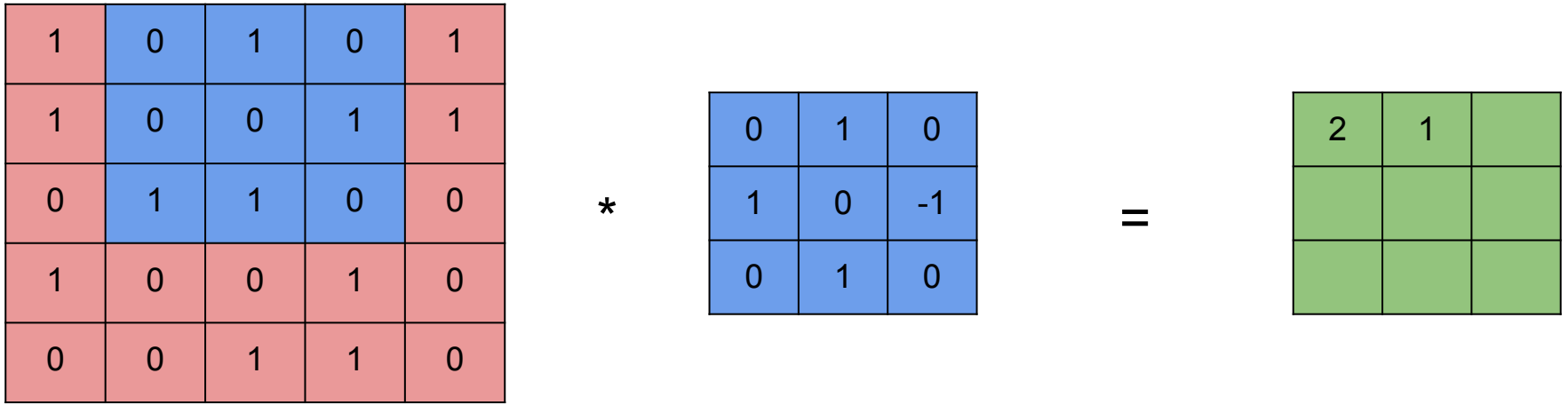
Input image

Filter or Kernel

Output or feature map

Stride

當stride = 1



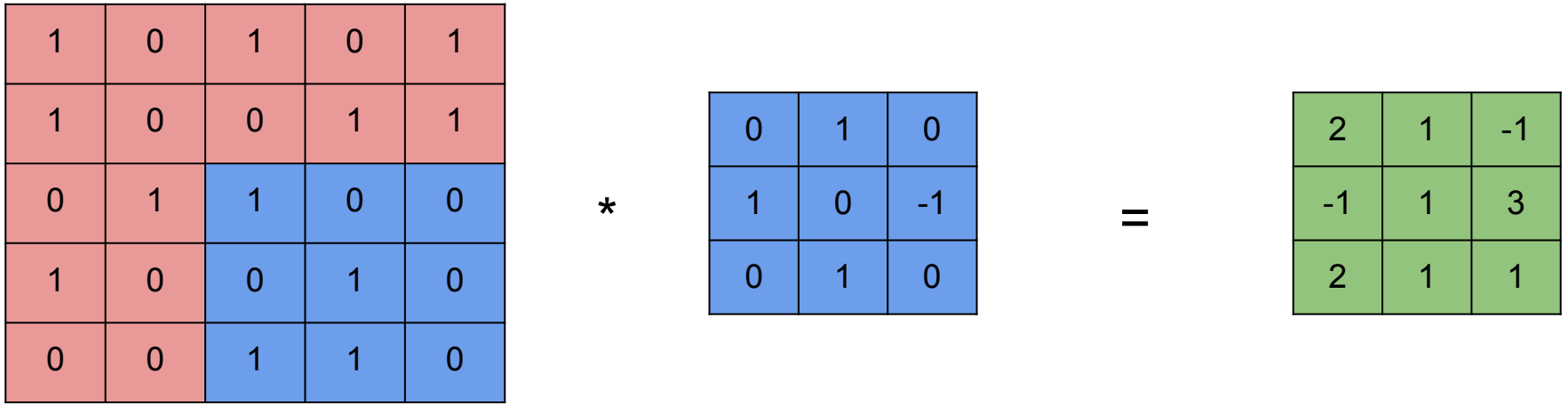
Input image

Filter or Kernel

Output or feature map

Stride

當stride = 1



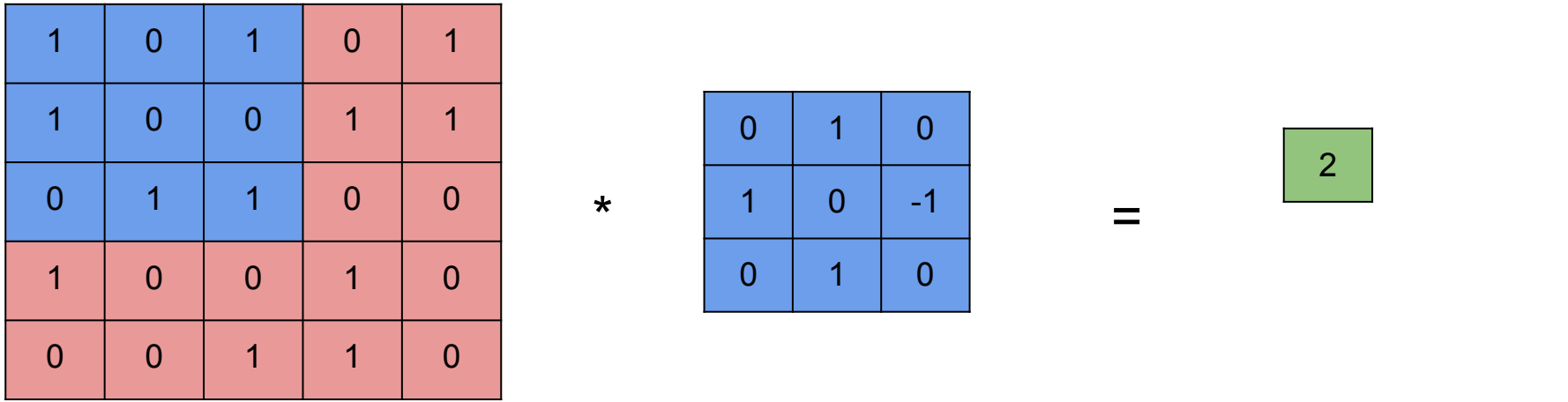
Input image

Filter or Kernel

Output or feature map

Stride

當stride = 2



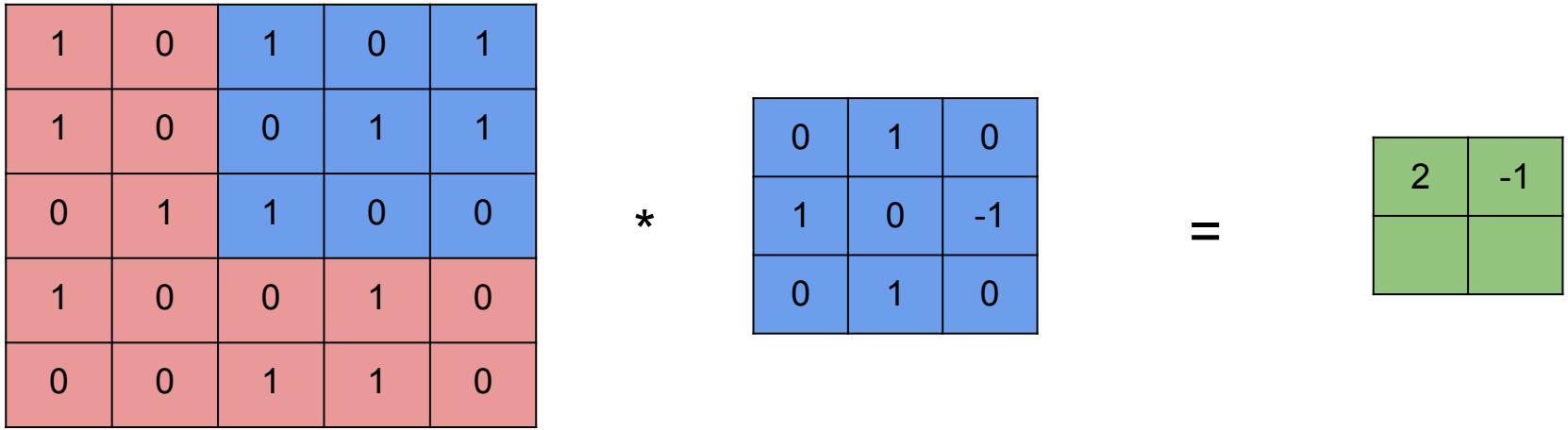
Input image

Filter or Kernel

Output or feature map

Stride

當stride = 2



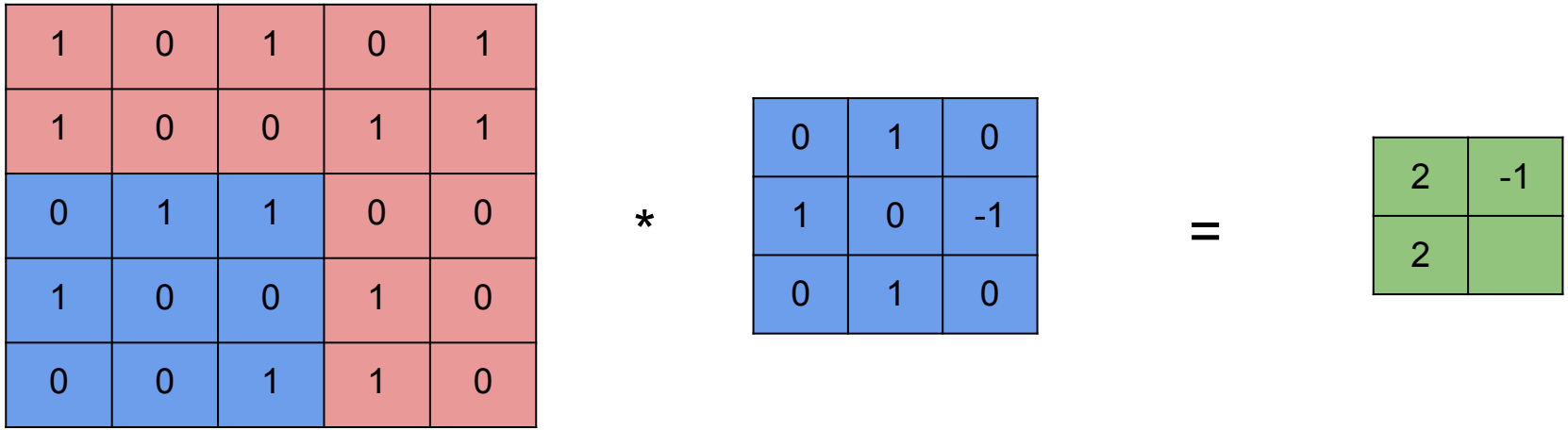
Input image

Filter or Kernel

Output or feature map

Stride

當stride = 2



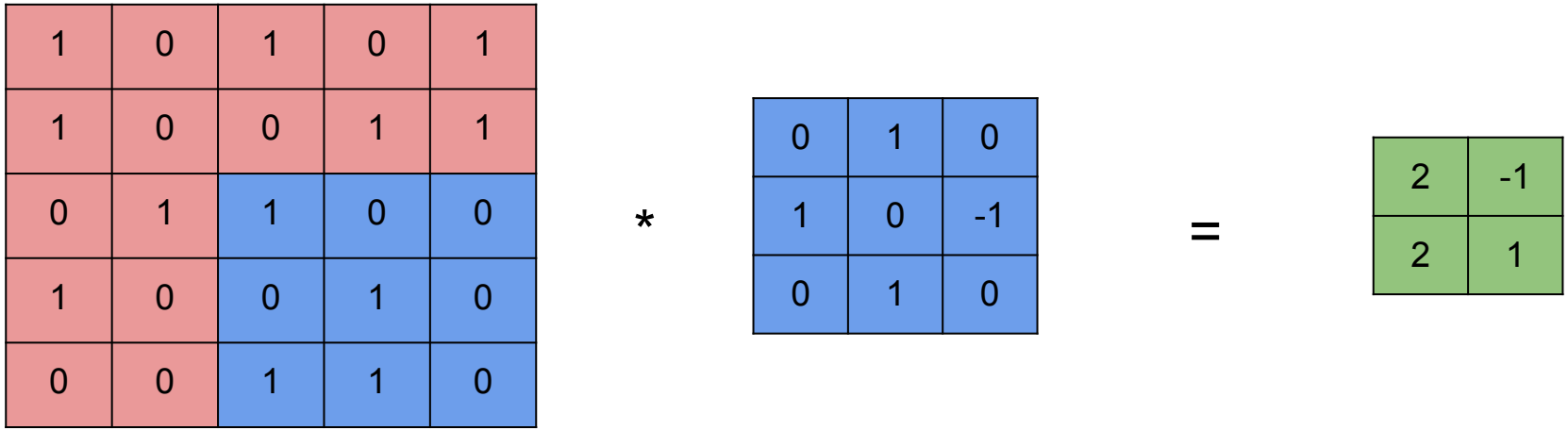
Input image

Filter or Kernel

Output or feature map

Stride

當stride = 2



Input image

Filter or Kernel

Output or feature map

使用不同stride

- 增加模型訓練效率
- 減少重複使用相同的pixel
- 控制feature map的大小

計算feature map大小

Stride = 2
Padding = 0

$$(\frac{n+2p-f}{s}+1)\times(\frac{n+2p-f}{s}+1)=(\frac{5+(2\times0)-3}{2}+1)\times(\frac{5+(2\times0)-3}{2}+1)=2\times2$$

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

Input image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	-1
2	1

Output or feature map

計算feature map大小

Stride = 1
Padding = 1

$$(\frac{n+2p-f}{s}+1)\times(\frac{n+2p-f}{s}+1)=(\frac{5+(2\times1)-3}{1}+1)\times(\frac{5+(2\times1)-3}{1}+1)=5\times5$$

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

Input image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

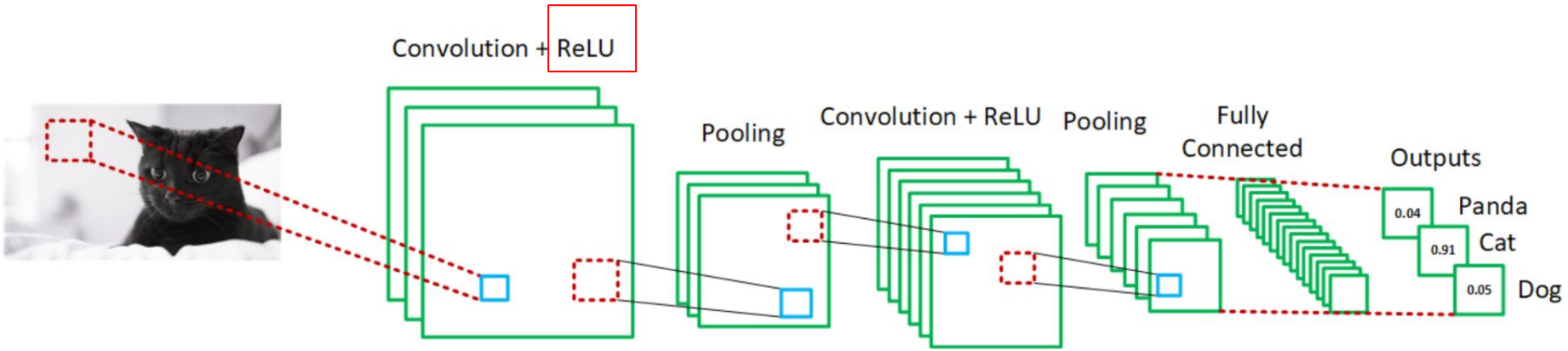
=

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

Output or feature map

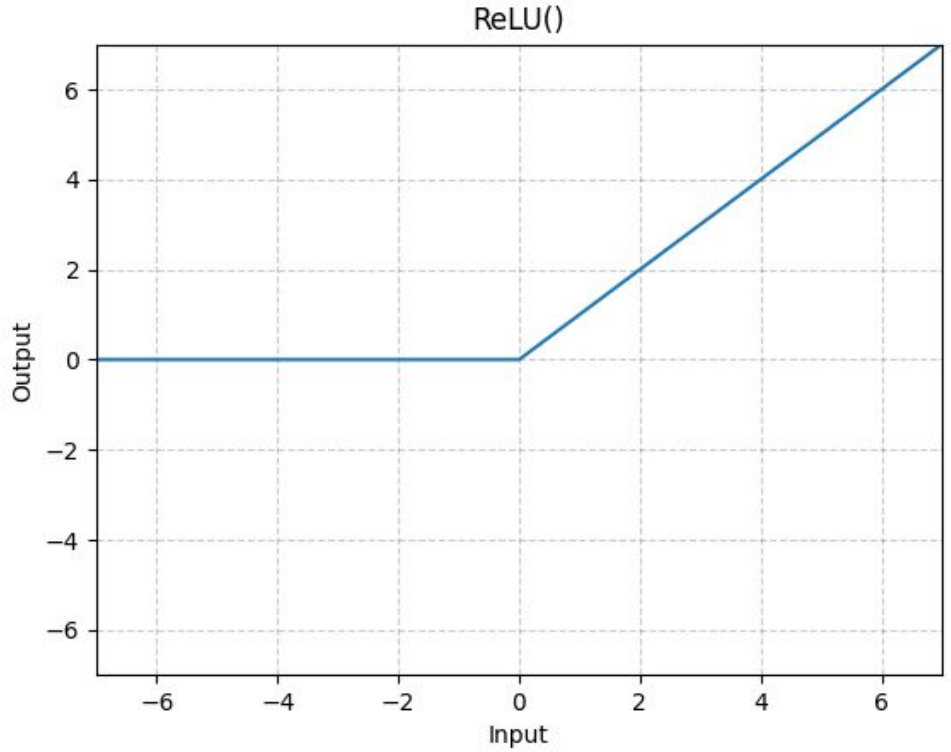
Activation layer ReLU

Activation layer ReLU



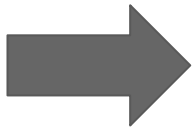
ReLU

- 將負數轉為0
- 保留正數



對feature map使用ReLU

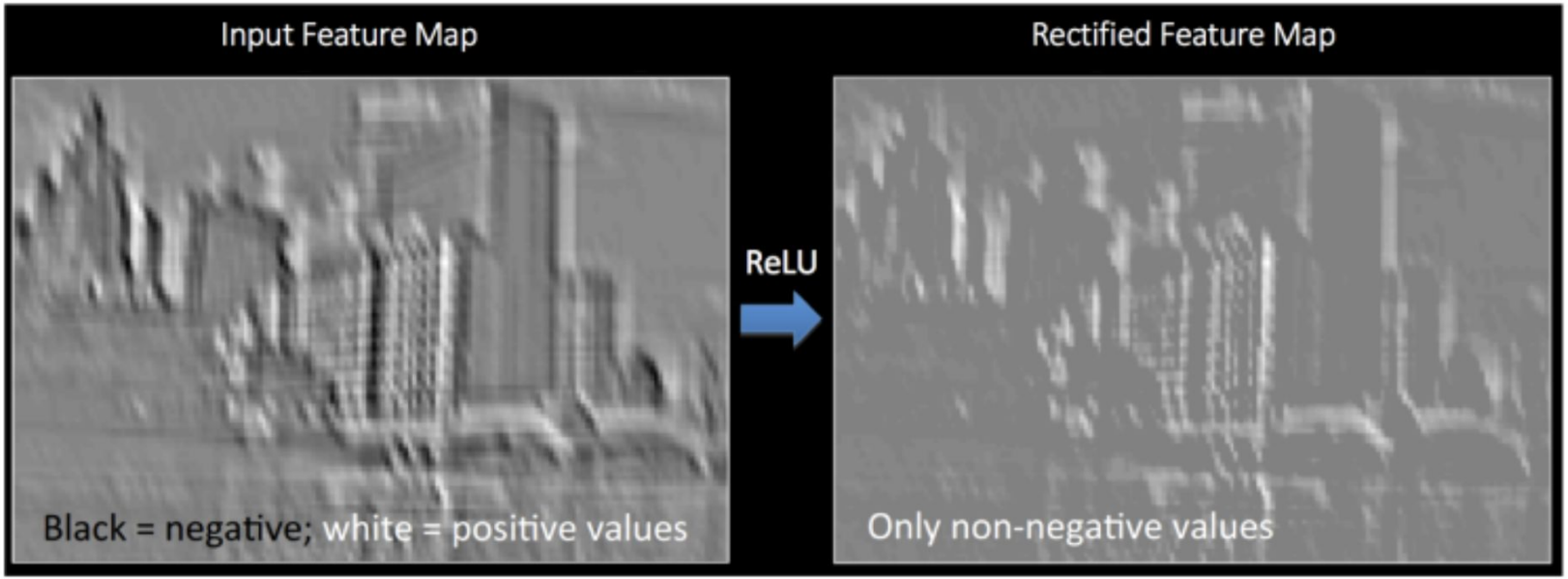
2	1	-1
-1	1	3
2	1	1



2	1	0
0	1	3
2	1	1

Output or feature map

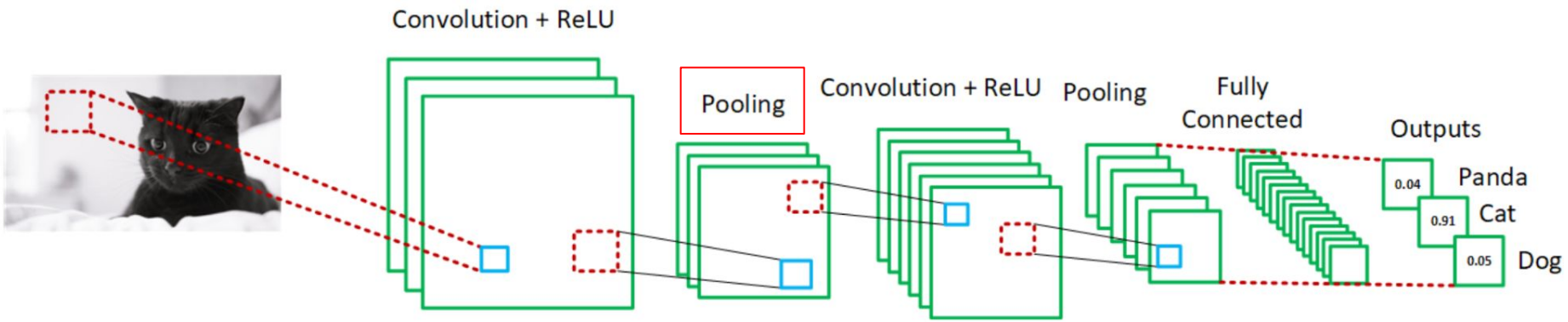
使用ReLU後的feature map



Source - http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Pooling layer

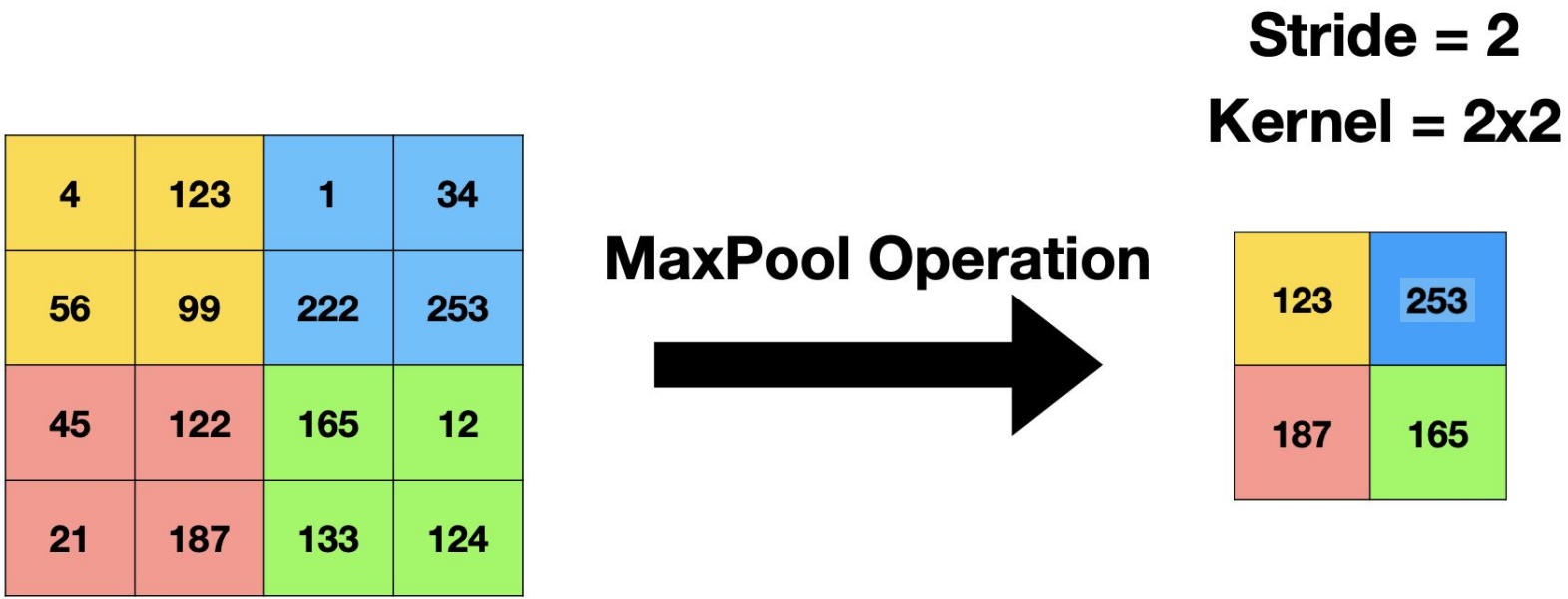
Pooling layer



Pooling

- 減少feature map的大小
- 在不丟失重要資訊的情況下減少訓練參數
- 也叫Subsampling or Downsampling

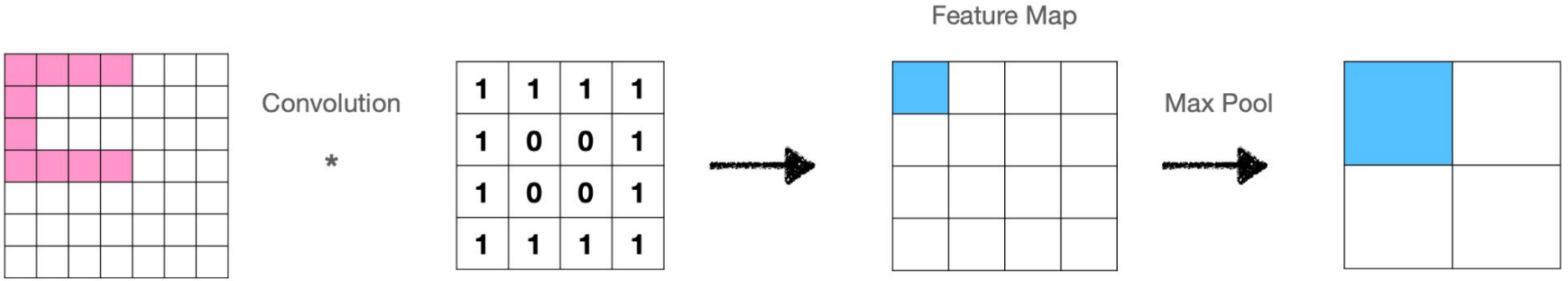
How max pooling works



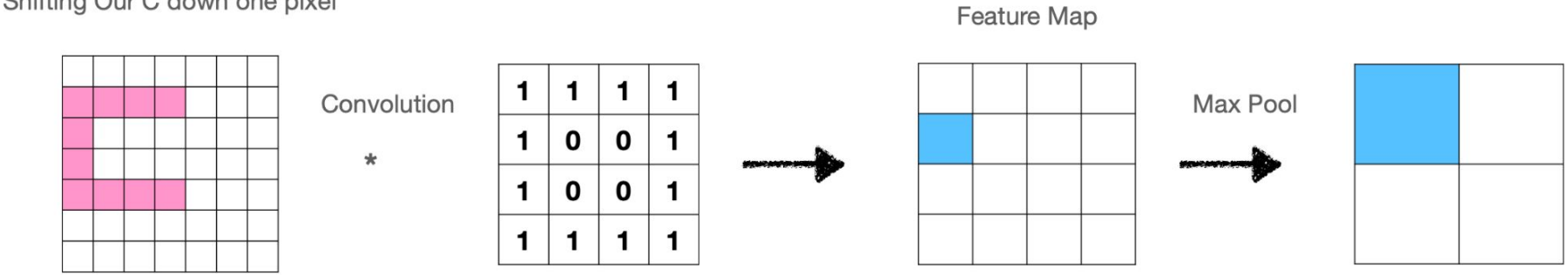
About pooling

- 常使用的pooling layer: 2x2 kernels, strides = 2, no padding
- Feature map size = ½ input size
- 其他pooling方法：average pooling, sum pooling

為什麼pooling有用



Shifting Our C down one pixel

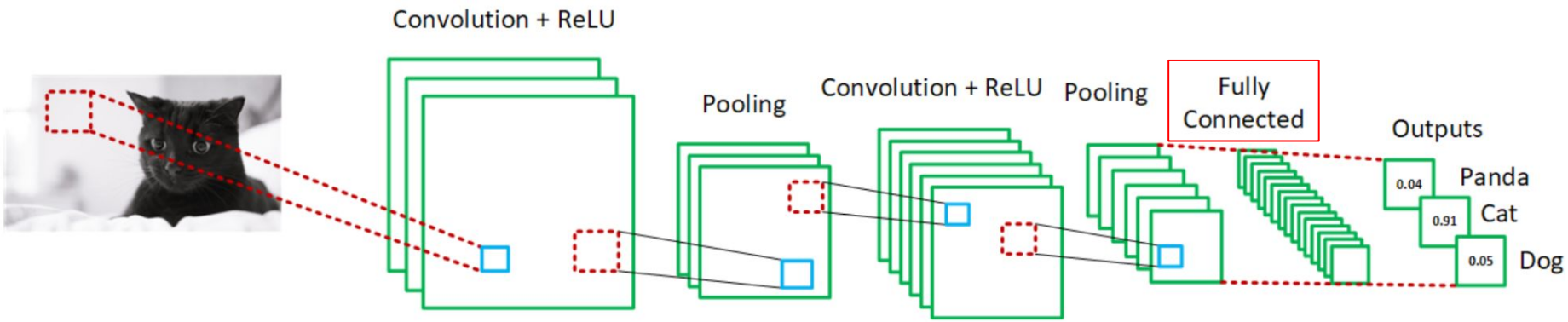


為什麼pooling有用

- 相鄰的pixel通常具有相似的特徵(尤其是初期的layer)
- 避免模型受到微小特徵差異的影響
- 在不丟失重要資訊的情況下減少訓練參數
- 太大的stride會丟失重要資訊

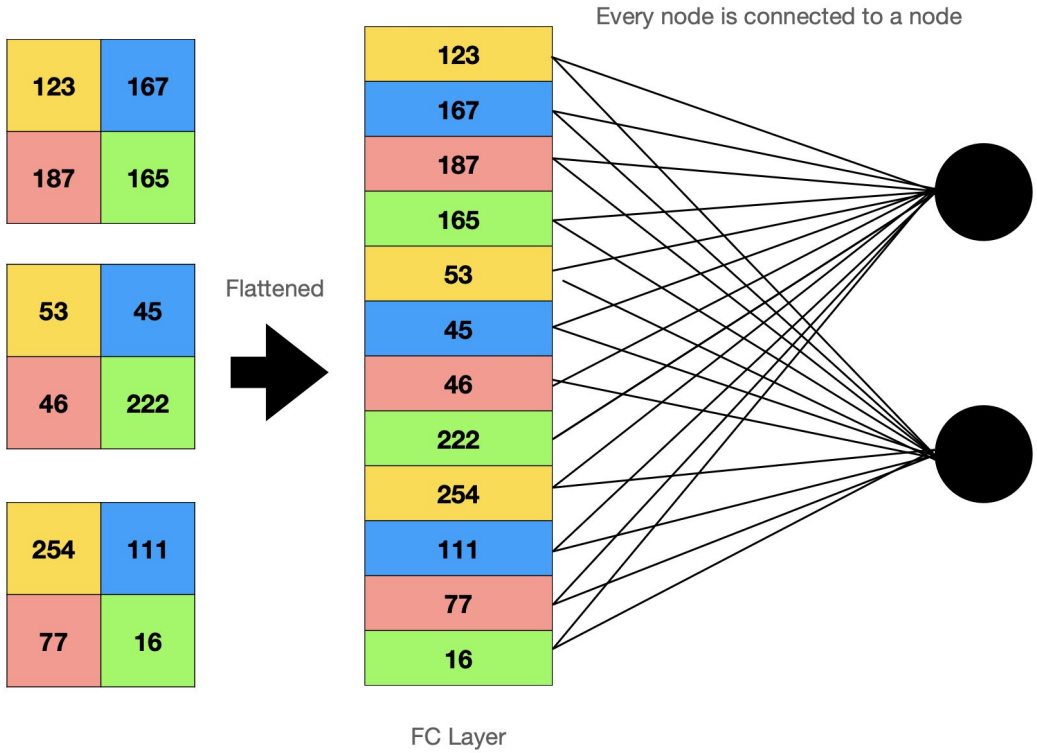
Fully connected layers

Fully connected layers



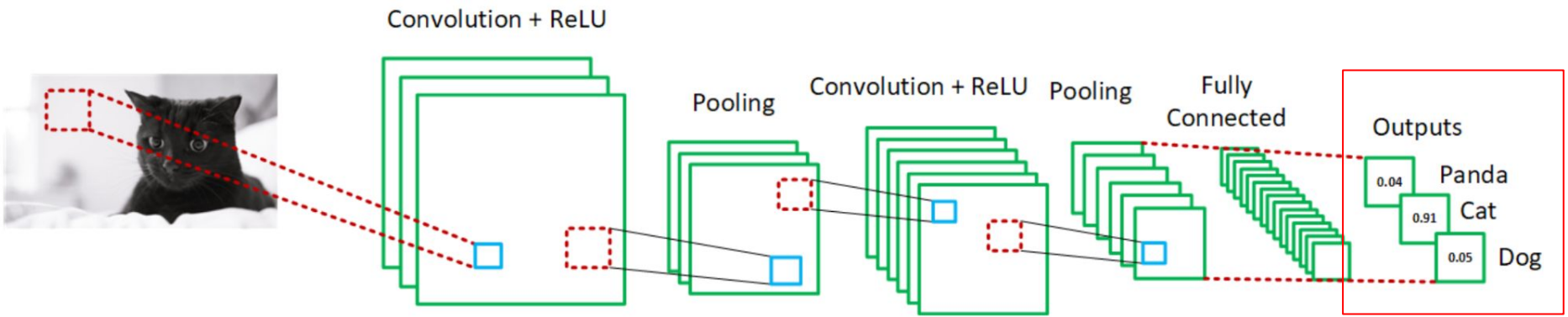
Fully connected layers

- FC layer上的所有點都和下一層的每個點連接
- 也叫Dense layer

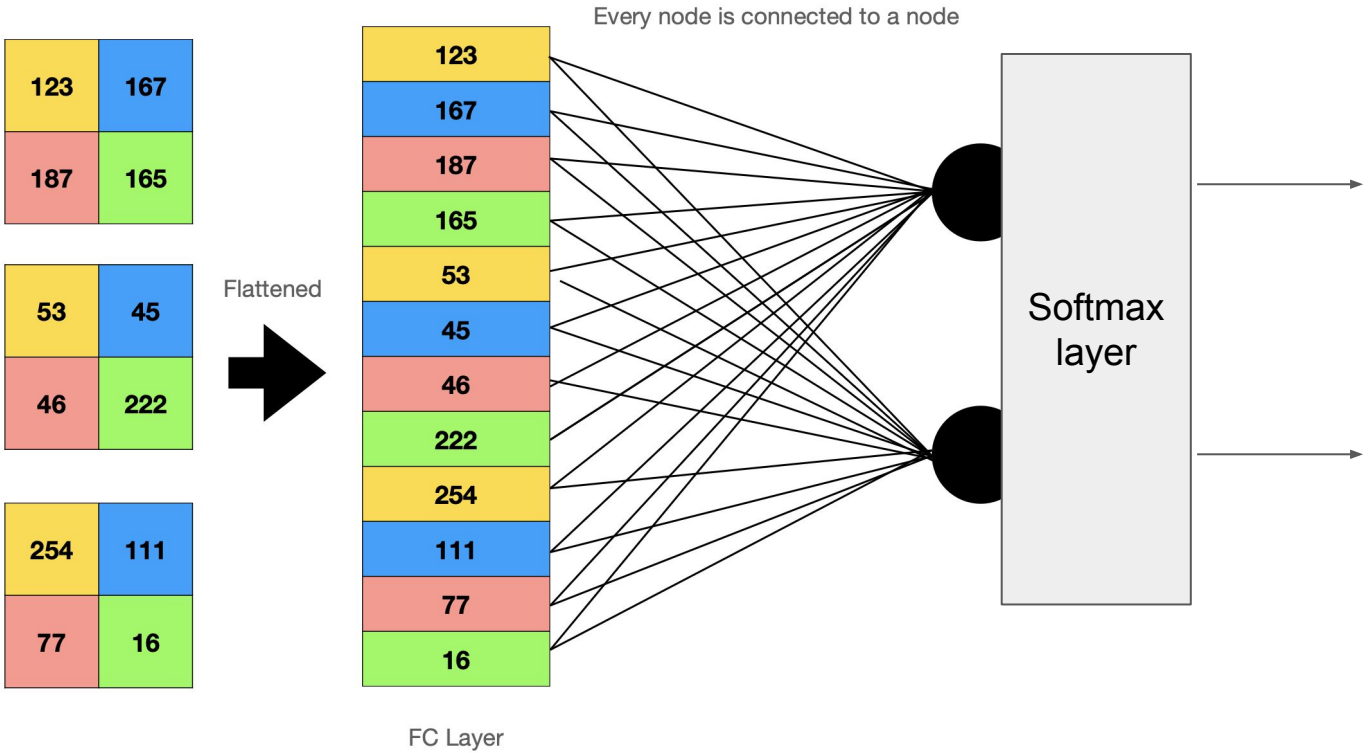


Softmax layer

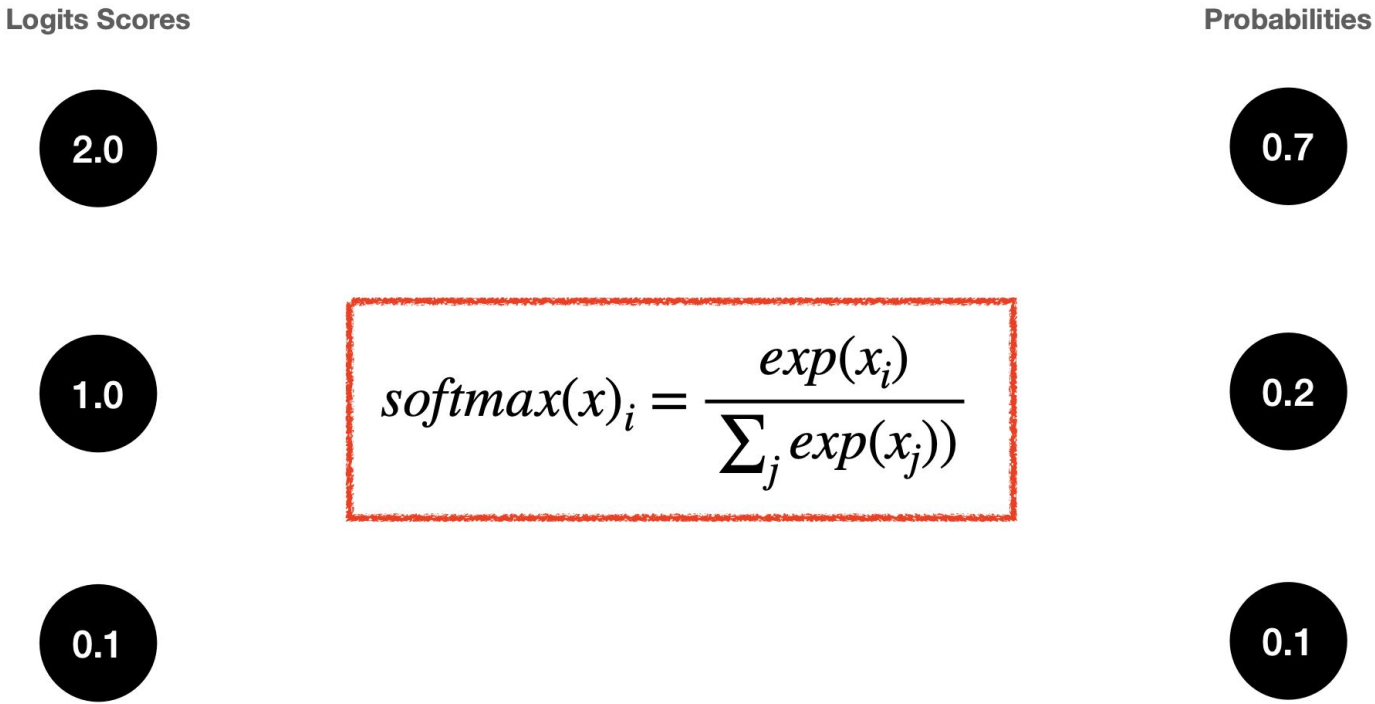
Softmax layers



Softmax layer

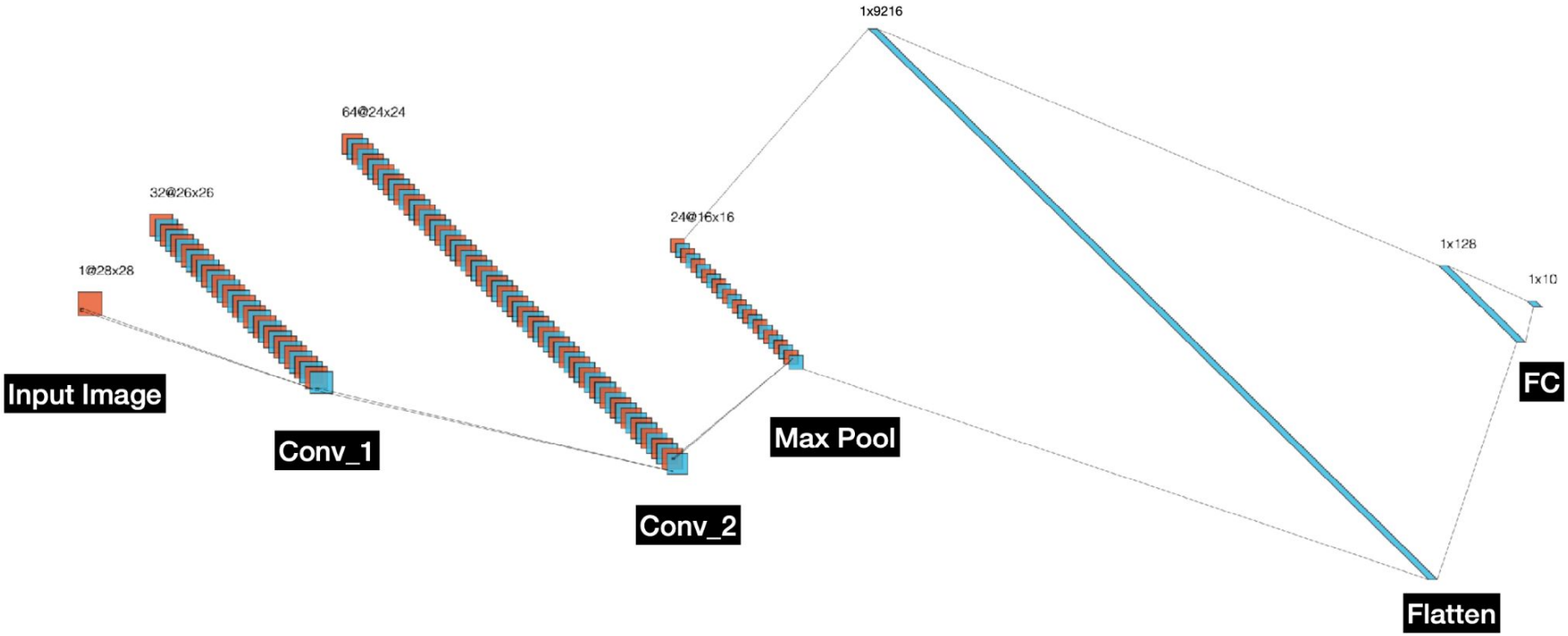


Softmax layer

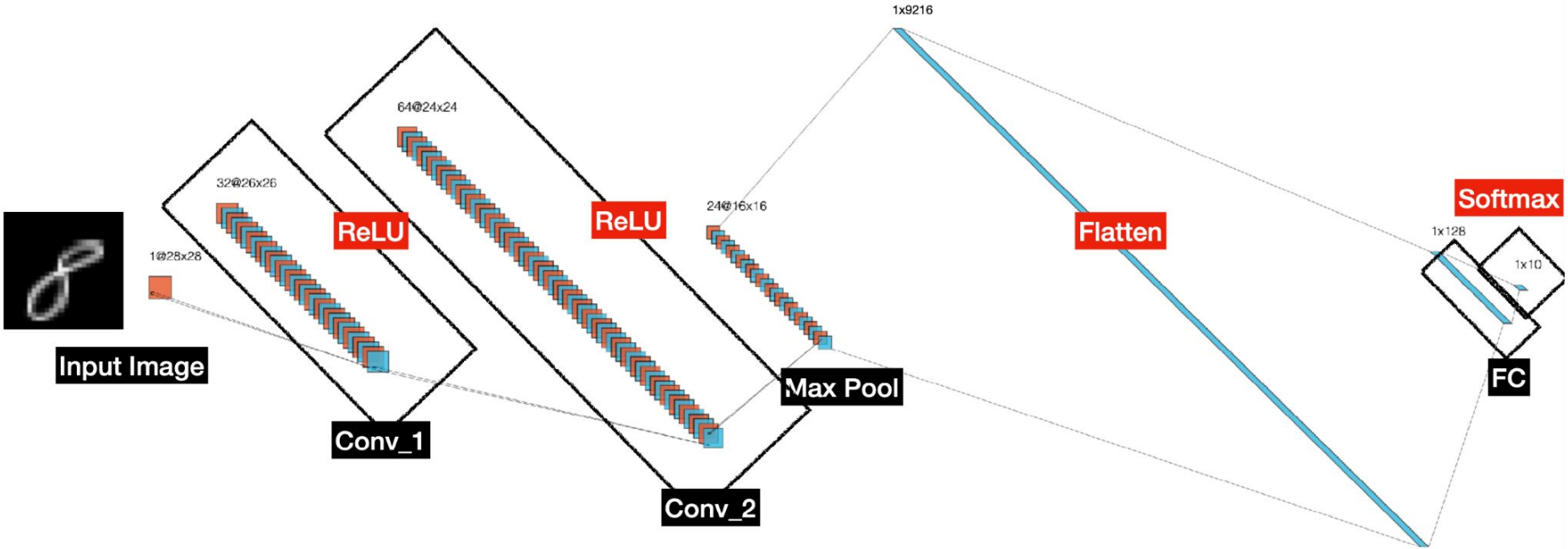


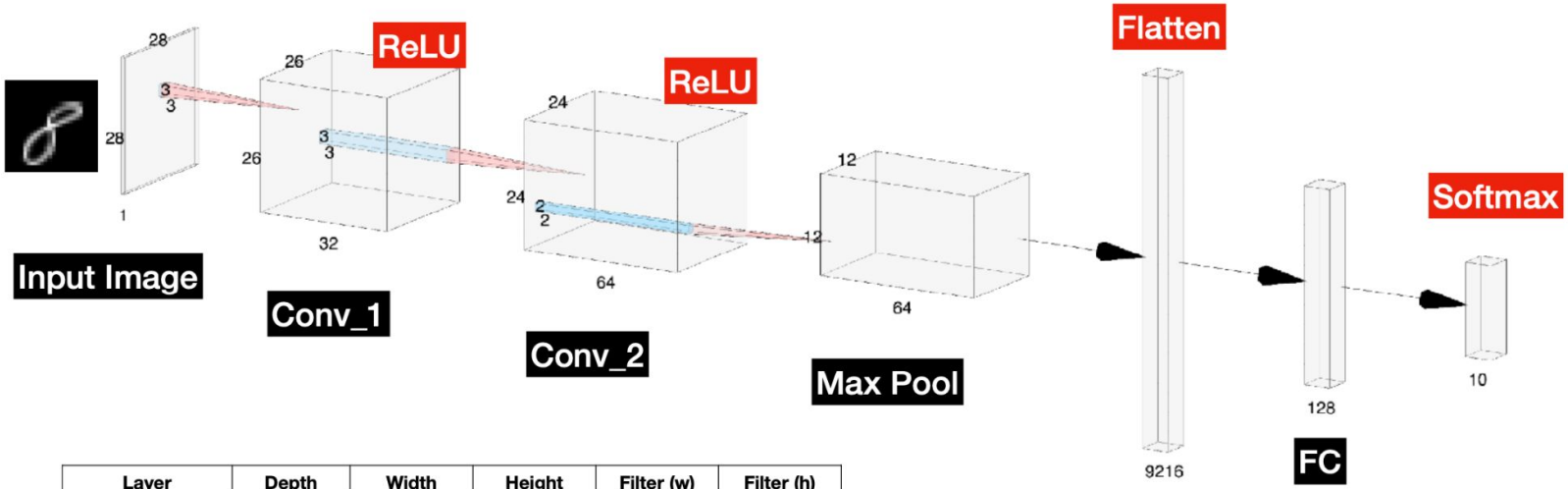
Putting together

簡易CNN


1x9216
64@24x24
32@26x26
24@16x16
1x128
1x10
1@28x28
Input Image
Conv_1
Conv_2
Max Pool
FC
Flatten

A 4 layer CNN for MNIST





Layer	Depth	Width	Height	Filter (w)	Filter (h)
Input	1	28	28		
Conv_1	32	26	26	3	3
Conv_2	64	24	24	3	3
Max Pool	64	12	12	2	2
Flatten	9216	1	1		
Fully Connected	128	1	1		
Output	10	1	1		

Notes:

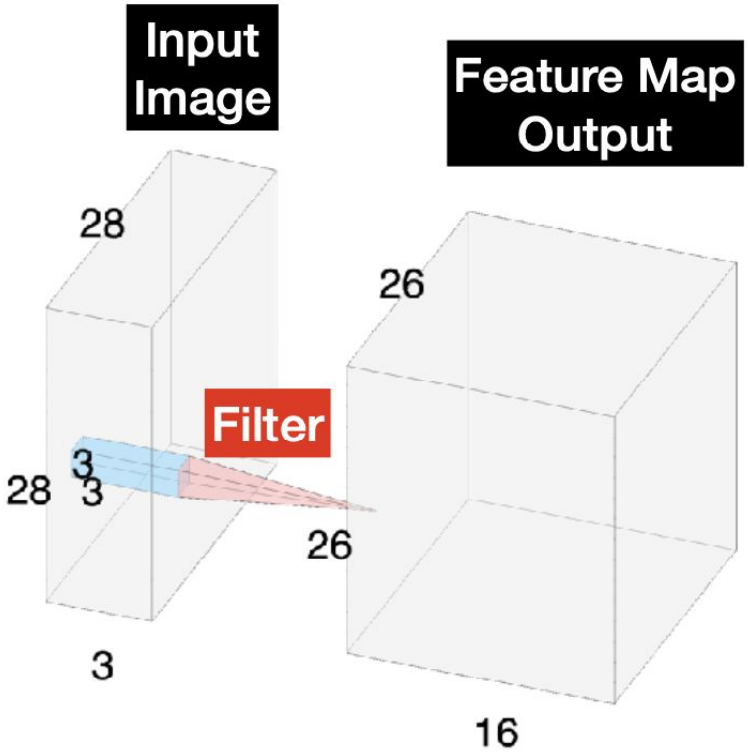
- We choose 32 & 64 Filters or Kernels for Conv_1 & Conv_2
- We choose to
- The Feature Maps are shown as Conv_1 and Conv_2
- Stride is 1
- Padding is 0 (not used)
- Max Pool Stride is 2

計算參數量

參數vs超參數

- 參數 (parameter)
  - 需要在訓練過程中讓模型去學習的變數
  - 又叫可學習的參數或權重(weights)
  - 隱藏層中的convolutional layer和fully connected layer都有權重
- 超參數 (hyperparameter)
  - 在訓練開始前由我們手動設定
  - 例如：節點數、activation function

Conv layer上的參數量

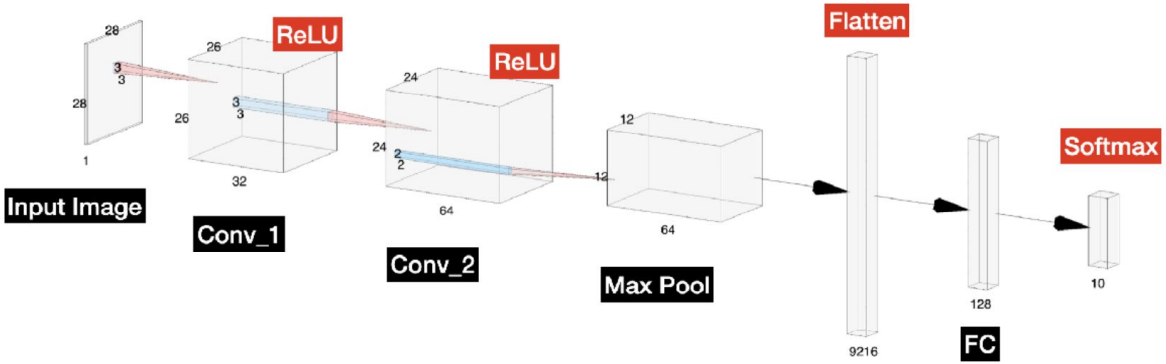


Decided by kernel size, depth, and number of filters

For one filter:
(weight * width * depth) + bias
= (3 * 3 * 3) + 1 = 10

For 16 filters:
((weight * width * depth) + bias) * N_f
= ((3 * 3 * 3) + 1) * 16 = 160

Simple CNN的參數量



Conv_1:
((Height × Width × Depth) + bias) × N_f
((3 × 3 × 1) + 1) × 32 = 320

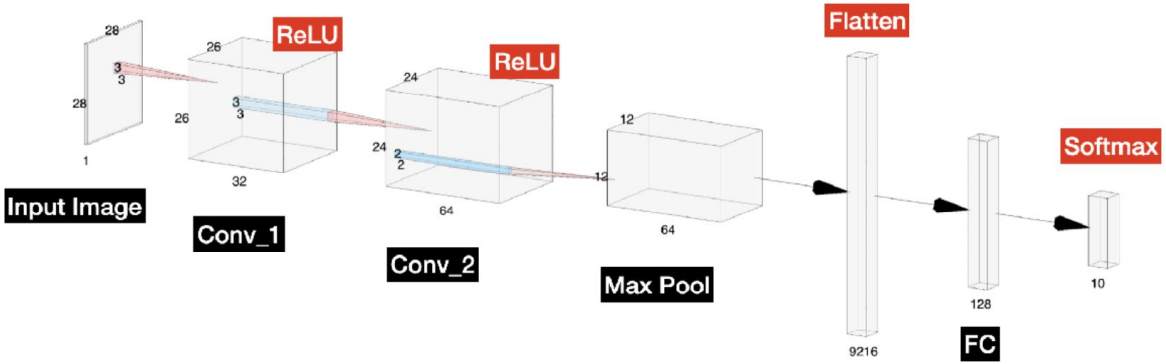
Conv_2:
((Height × Width × Depth) + bias) × N_f
((3 × 3 × 32) + 1) × 64 = 18,496

沒有可訓練的參數的層:
ReLU, Max Pool, Flatten

FC:
(Length + bias) × N_{nodes}
(9216 + 1) × 128 = 1,179,776

Final output (FC):
(Length + bias) × N_{nodes}
(128 + 1) × 10 = 1,290

Simple CNN的參數量



Total = 320 + 18496 + 1179776 + 1290 = **1,199,882**

Conv_1:
((Height × Width × Depth) + bias) × N_f
((3 × 3 × 1) + 1) × 32 = 320

Conv_2:
((Height × Width × Depth) + bias) × N_f
((3 × 3 × 32) + 1) × 64 = 18,494

沒有可訓練的參數的層:
ReLU, Max Pool, Flatten

FC:
(Length + bias) × N_{nodes}
(9216 + 1) × 128 = 1,179,776

Final output (FC):
(Length + bias) × N_{nodes}
(128 + 1) × 10 = 1,290

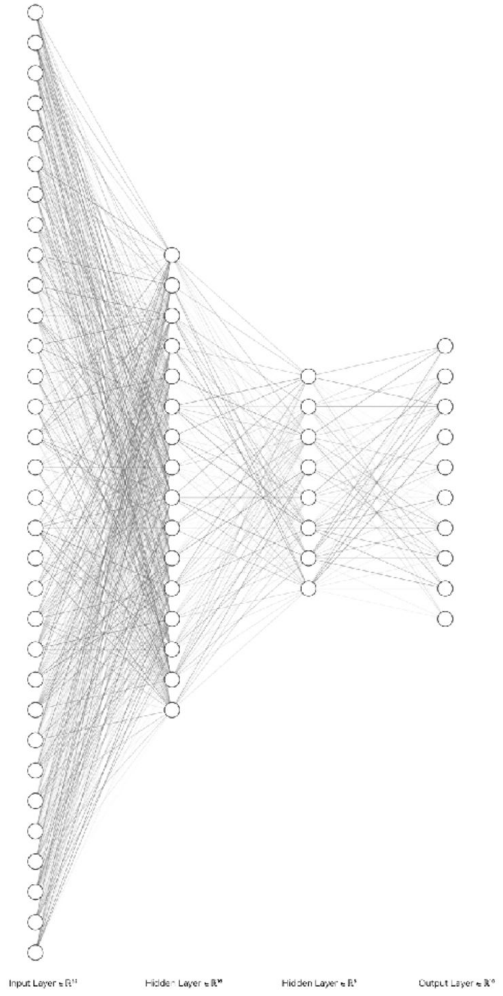
為什麼CNN有用？

如果使用一般的神經網絡

- 一張28 x 28的灰階圖片有784個pixels
- 參照CNN裡的Conv_1:
  32 filters for 26 x 26 feature maps
  = 21,632 nodes
- 參數量 = 784 * 21632 = 16,959,488 (一層)
- Conv_1參數量:320

如果使用一般的神經網絡

- 無法擴展到資料量大的輸入
  Ex: 320 x 320
- Overfitting

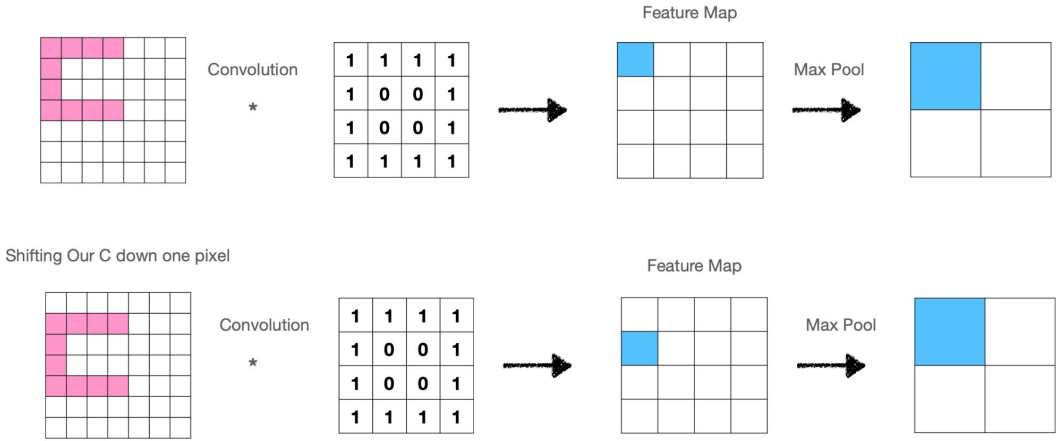


CNN的優點

Parameter sharing:
一個filter的參數可以用在image的不同地方

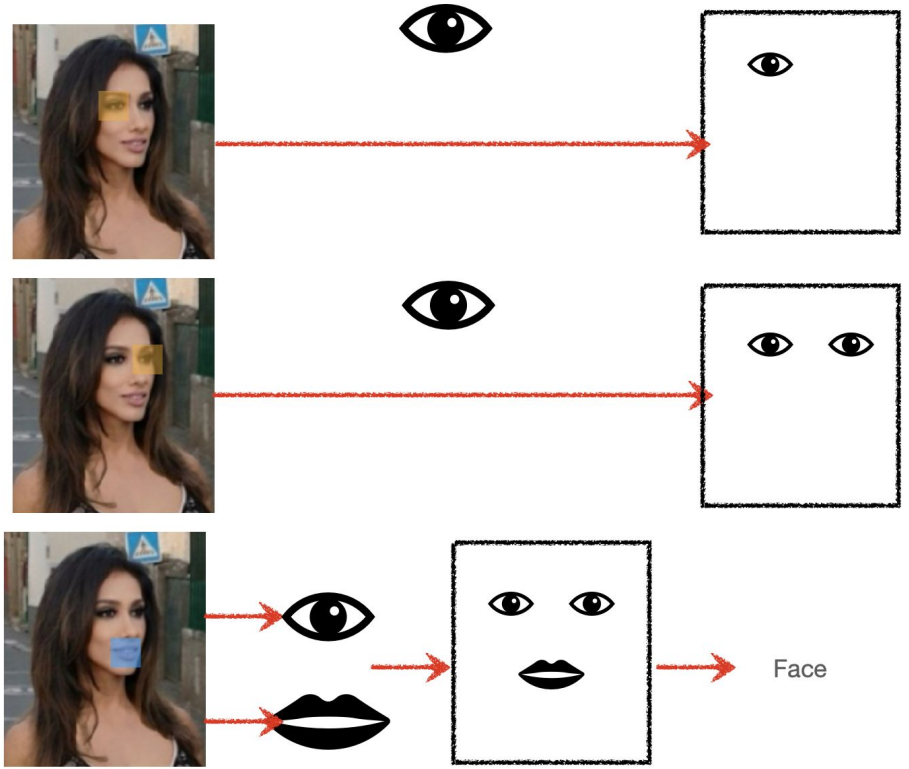
Sparsity of connection:
在不丟失重要資訊的情況下捨棄一些pixels

Invariance:
特徵可以出現在各個位置



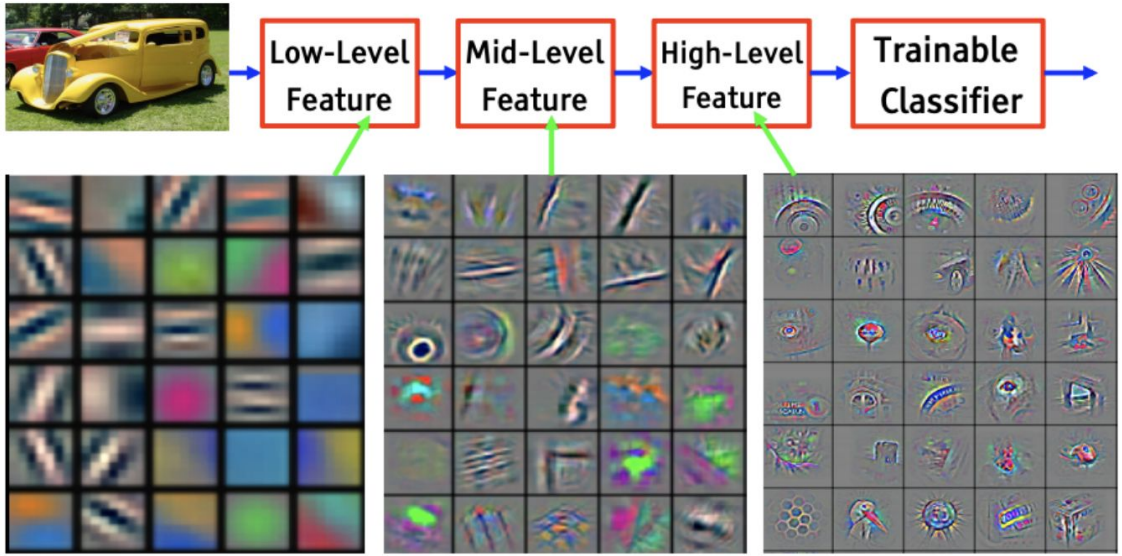
CNN背後的假設

- 低階的特徵會靠比較近
- 特徵可以出現在不同位置
- 高階的特徵是由低階特徵組成



訓練CNN

Conv filter學到了什麼



http://www.iro.umontreal.ca/~bengioy/talks/DL-Tutorial-NIPS2015.pdf

前面的層學低階的特徵
Ex: 邊緣、線條

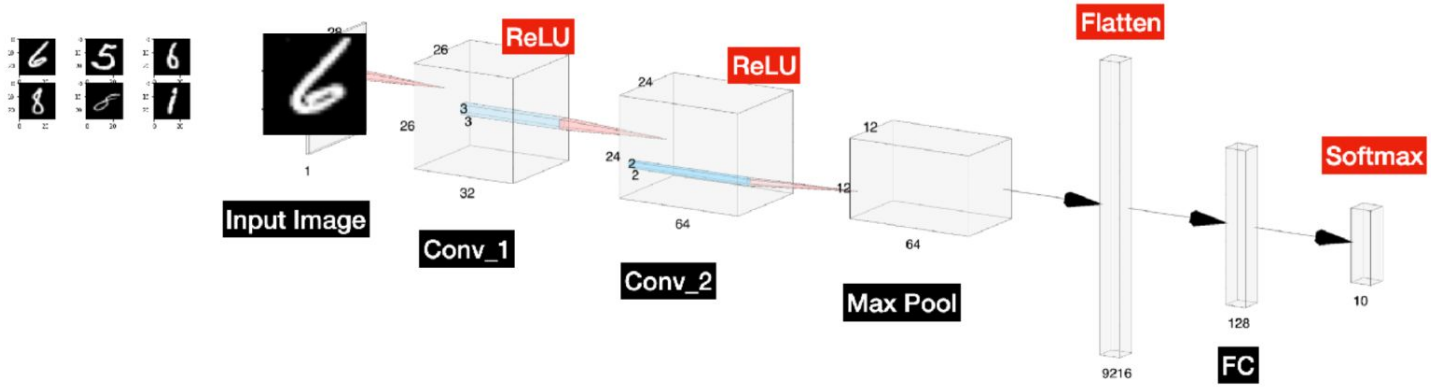
靠中間的層學中階的特徵
Ex: 眼睛

後面的層學高階的特徵
Ex: 鳥、花

Filter是如何學習的？

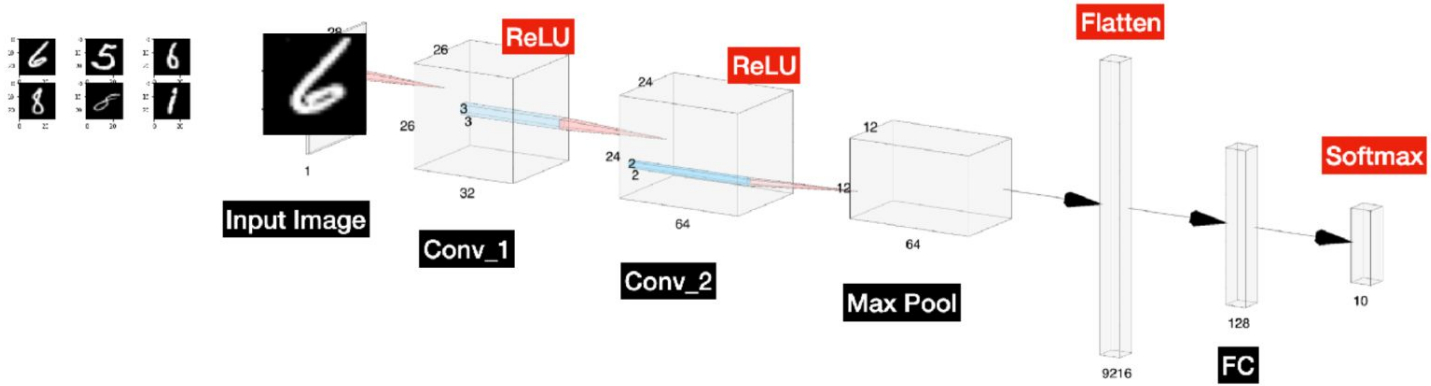
- 隨機初始權重 (weights)
- 將單張或多張圖片前向傳遞 (forward propagation)
- 計算誤差
- 使用反向傳遞 (backward propagation) 和梯度下降來更新權重
- 傳遞更多圖片 (one epoch)
- 反覆執行，直到訓練損失達到滿意程度

訓練過程



P	6	5	6	8	8	1
0	0	0.9	0.83	0.21	0.19	0.62
1	0.73	0.8	0.89	0.7	0.92	0.07
2	0.78	0.88	0.19	0.39	0.08	0.74
3	0.37	0.56	0.07	0.64	0.64	0.9
4	0.63	0.25	0.79	0.94	0.52	0.55
5	0.87	0.65	0.57	0.63	0.97	0.04
6	0.67	0.05	0.45	0.51	0.87	0.51
7	0.71	0.66	0.13	0.59	0.86	0.89
8	0.51	0.88	0.59	0.01	0.37	0.63
9	0.24	0.52	0.79	0.15	0.63	0.78

訓練過程



P	6	5	6	8	8	1
0	0	0.9	0.83	0.21	0.19	0.62
1	0.73	0.8	0.89	0.7	0.92	0.07
2	0.78	0.88	0.19	0.39	0.08	0.74
3	0.37	0.56	0.07	0.64	0.64	0.9
4	0.63	0.25	0.79	0.94	0.52	0.55
5	0.87	0.65	0.57	0.63	0.97	0.04
6	0.67	0.05	0.45	0.51	0.87	0.51
7	0.71	0.66	0.13	0.59	0.86	0.89
8	0.51	0.88	0.59	0.01	0.37	0.63
9	0.24	0.52	0.79	0.15	0.63	0.78

需要一個方法來告訴模型結果好壞

Loss function

Cross entropy

進一步細分

- Binary cross entropy -> binary class
- Categorical cross entropy -> multiclass

Class	Predicted Probabilities	Ground Truth
0	0.1	0
1	0.2	0
2	0.1	0
3	0.05	0
4	0.05	0
5	0.05	0
6	0.05	0
7	0.3	1
8	0.05	0
9	0.05	0

Cross entropy: 計算兩個機率分佈間的差異

$$L = -y \,.\, log(\hat{y})$$

正確值　　預測值

Class	Predicted Probabilities	Ground Truth
0	0.3	0
1	0.6	1
2	0.1	0

- $L = -\, y \,.\, log(\hat{y})$
- $L = -\,(0 \times log(0.3) + 1 \times log(0.6) + 0 \times log(0.1)$
- $L = -\,(0 + 1 \times -0.222 + 0) = 0.222$

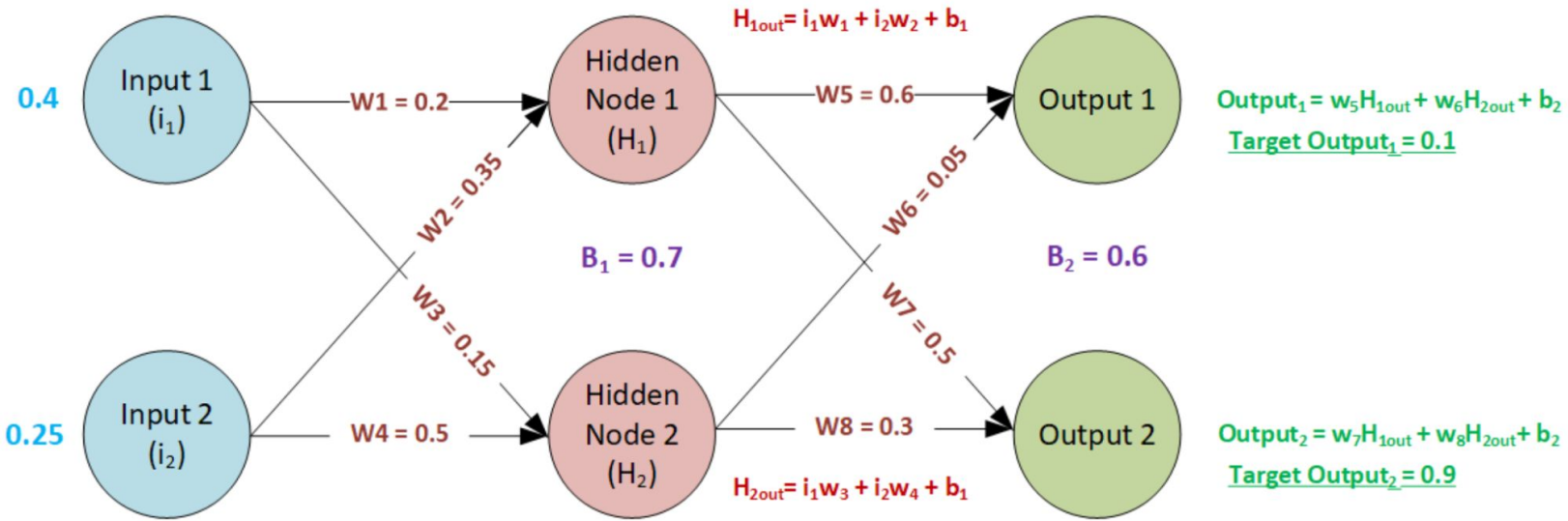
其他loss functions

- 迴歸任務 (regression): Mean square error (MSE)
  - Mean Square Error (MSE) = (目標 - 預測)^2
- 更多:
  - L1, L2
  - Hinge Loss
  - Mean Absolute Error (MAE)

$$MSE = \frac{1}{n}\sum (Y_i - \hat{Y}_i)^2$$

Backward propagation

Backward propagation (反向傳遞)



稍微增加W5到0.6001或稍微減少到0.5999，哪個才能減少訓練損失

Chain rule

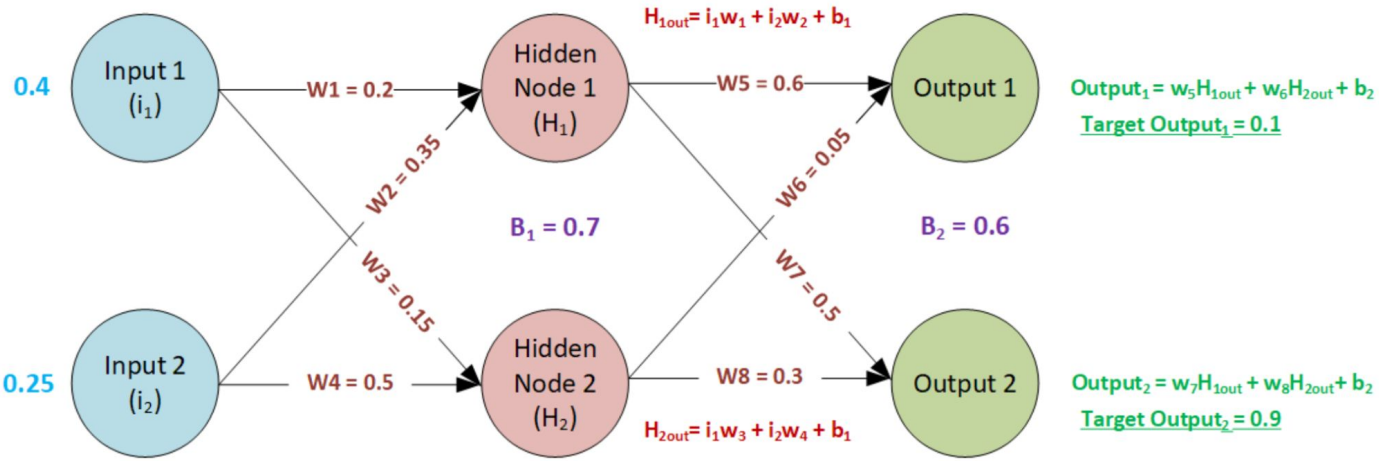
假設有兩個函式y = f(u), u = f(x), 則y的導數為

$$\frac{dy}{dx} = \frac{dy}{du} \times \frac{du}{dx}$$

If y is the total loss and x is a weight, then dy/dx = the gradient of x

Backward propagation details: https://www.youtube.com/watch?v=iyn2zdALii8

Example



稍微改變W5會對訓練損失影響多少？

$$\frac{dE_T}{dW_5}$$

E_T = 總訓練損失

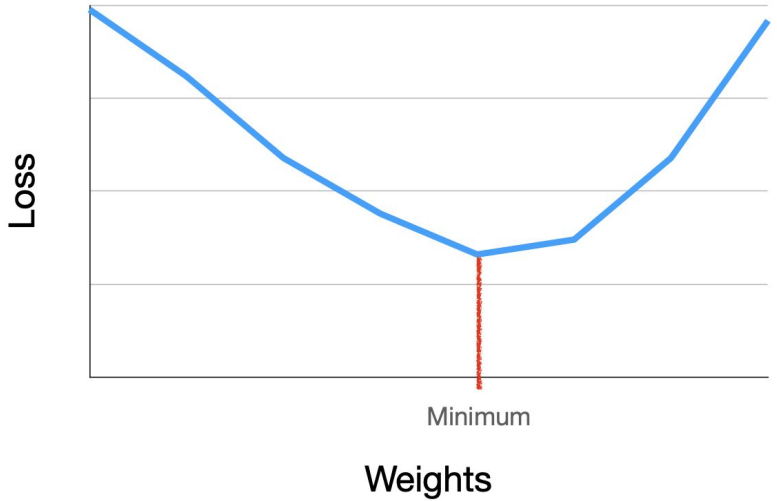
$$W_5 = W_{5old} - \lambda \times \frac{dE_T}{dW_5}$$

Gradient descent

Gradient descent details: https://www.youtube.com/watch?v=sDv4f4s2SB8

Loss function

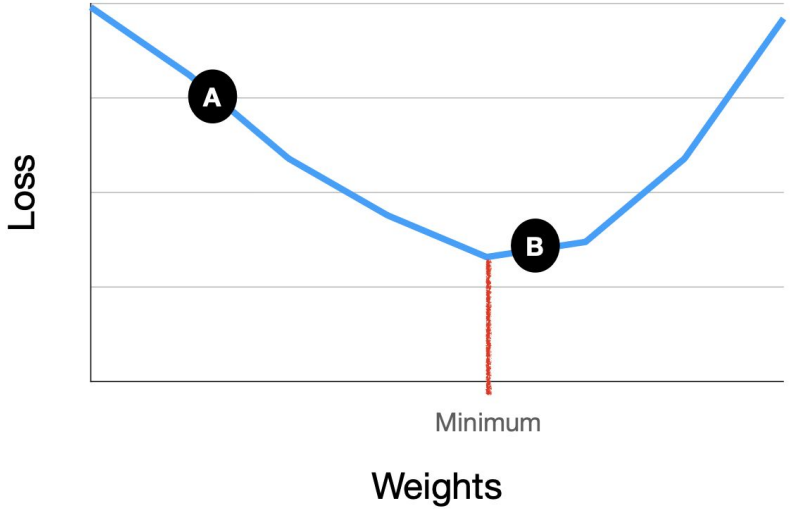
- 使用反向傳遞更新參數
  - wx + b
- 找到一組產生最低訓練損失的參數
- 更新的過程就是梯度下降 (gradient descent)



Gradient descent

- 稍微改變W會對損失E影響多少？

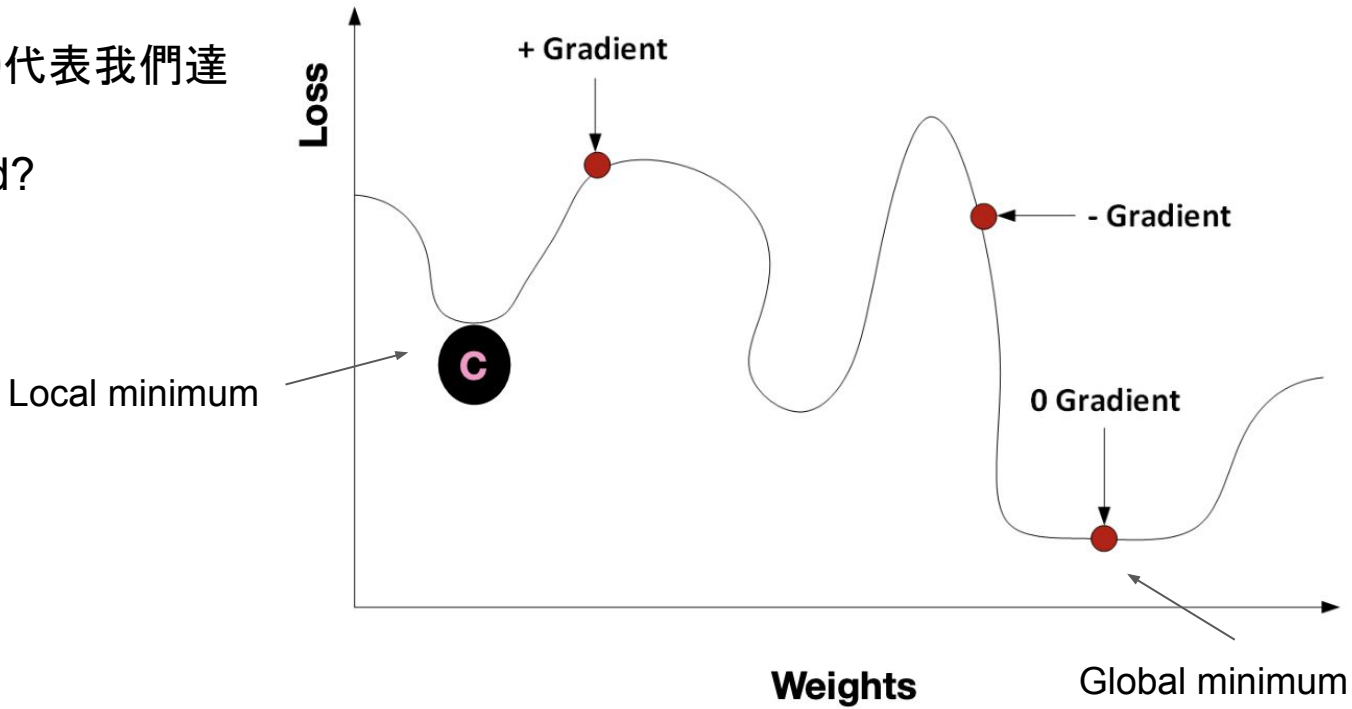
$$Gradient = \frac{dE}{dw}$$



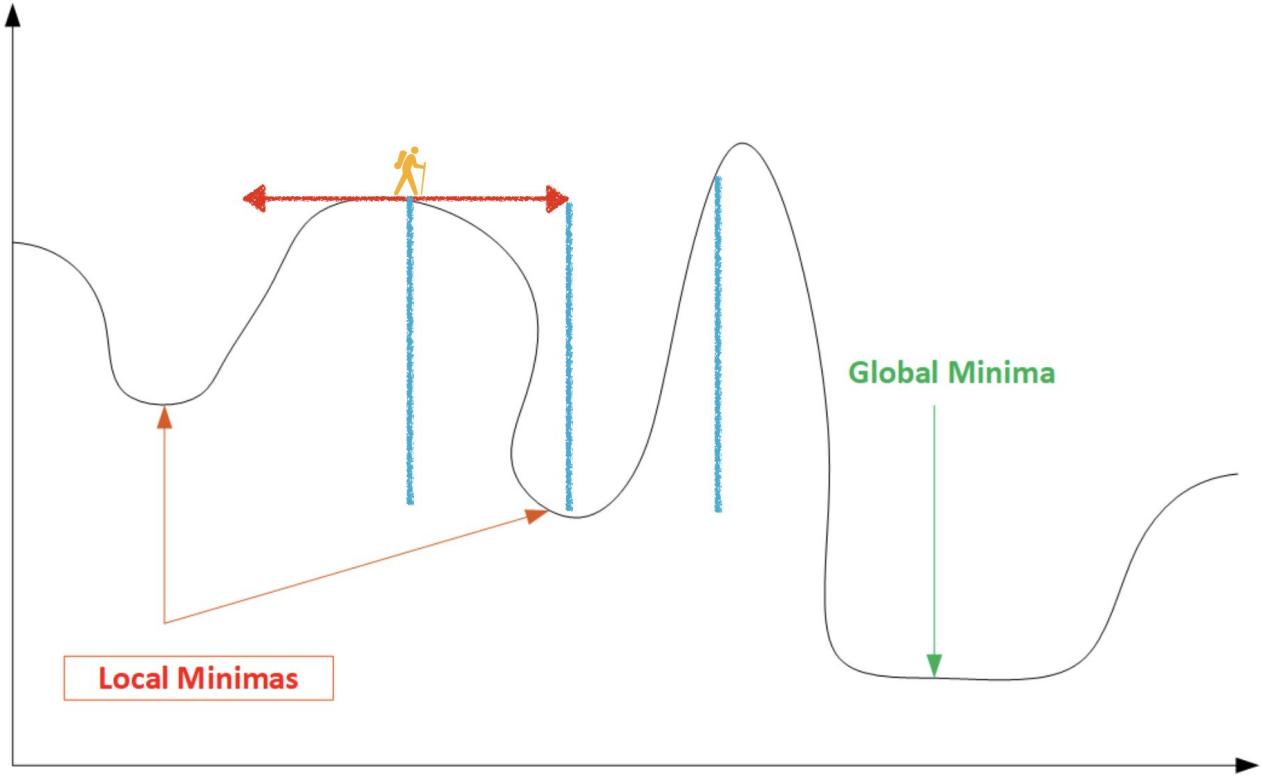
- 如果gradient為正數，代表損失隨著權重增加而增加
- 如果gradient為負數，代表損失隨著權重增加而減少
- Gradient的值會告訴模型該往哪邊移動

If gradient becomes 0

- Gradient為0代表我們達到低點了
- Good or bad?



Step size is important



Learning rate

$$W_5 = W_{5old} - \lambda \times \frac{dE_T}{dW_5}$$

- λ is learning rate
- Learning rate決定我們更新的步幅
- 找到最佳的learning rate可以避免模型卡在區域最佳解以及超過全域最佳解

Gradient descent的方法

- Batch gradient descent - 每次看完全部資料才更新權重
  - 增加計算資源以及時間
- Stochastic gradient descent - 每次看完一筆資料就更新權重
  - 計算很快
  - 不穩定且有可能訓練很久
- Mini-batch gradient descent - 每次看完一批資料就更新權重
  - 增加訓練速度並達到global minimum
  - Batch size → 8~256

Gradient descent的問題

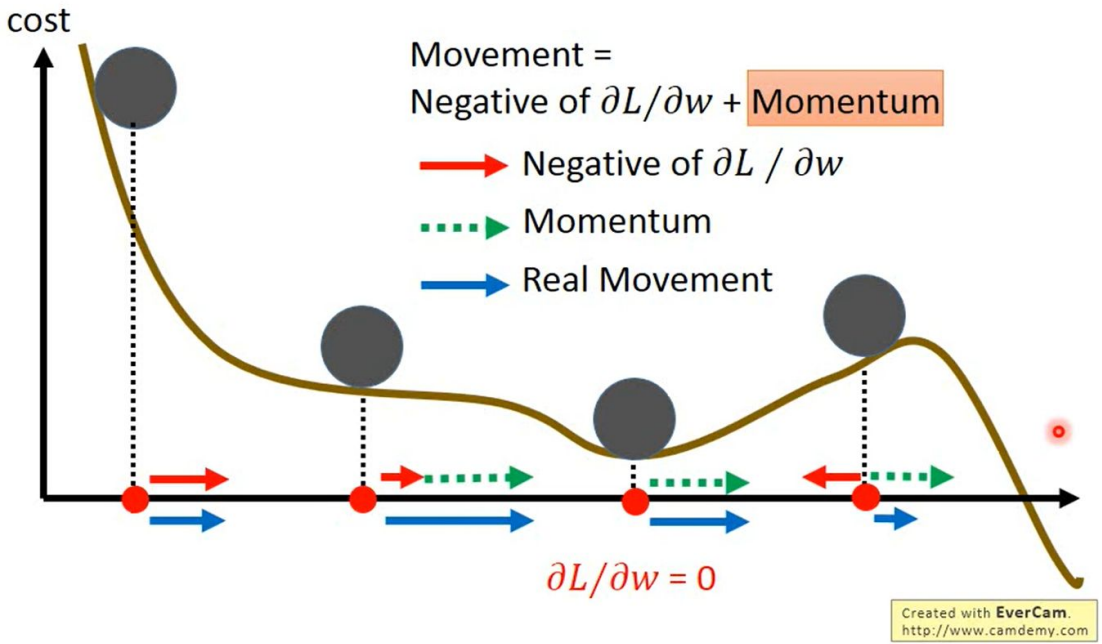
- 需要決定learning rate
- 使用固定的learning rate容易使模型卡在區域最佳解以及超過全域最佳解

Solution

- Momentum
- Learning rate schedule

Momentum

Still not guarantee reaching global minima, but give some hope ……



Source: https://ithelp.ithome.com.tw/articles/10276342

Quiz next week

- Scope: CNN
- Paper based (bring a pen!)

Implementation

Download notebook at:
https://github.com/albert831229/nchu-computer-vision/tree/main/113/day