Introduction to Deep Learning
Chapter 5: Convolutional Neural Networks (Part 1)

Pao-Ann Hsiung

National Chung Cheng University

Contents

Introduction

- Deep Learning for Computer Vision
  - Rapid advances in computer vision due to deep learning
  - Cross-fertilization between computer vision and other fields for deep learning
    - For example, speech recognition vs. computer vision

Computer Vision Problems

Image Classification

64x64

→ Cat? (0/1)

Neural Style Transfer

Object detection

Deep Learning on Large Images

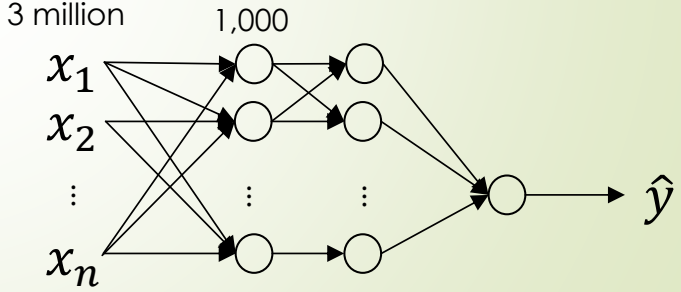
Cat? (0/1)

64x64x3 = 12288

1000x1000x3 = 3 million

$W^{[i]}$: $(1000, 3m)$

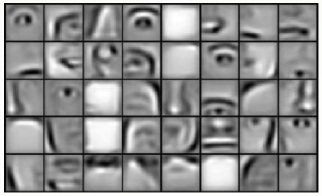
Size: 3 billion



Computer Vision Problem

找出邊界

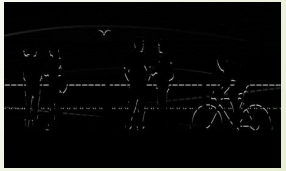
再從邊界找出形狀



再創造物件

垂直邊界

vertical edges



水平邊界

horizontal edges

Vertical edge detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

convolution

*

為0所以vertical

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

3 x 3 filter (kernel)

=

-5	**-4**	**0**	**9**
-10	**-2**	**2**	**3**
0	-2	-4	-7
-3	-2	-3	-16

Vertical edge detection examples

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0

Vertical edge detection

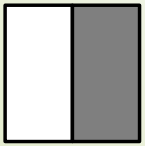
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

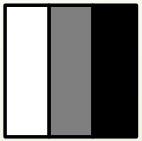
1	0	-1
1	0	-1
1	0	-1

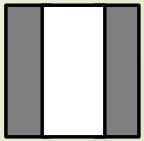
=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



*





Vertical and Horizontal Edge Detection

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

Vertical

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

Horizontal

較混亂

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=

越邊數字越大

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Learning to detect edges

透過filiter偵測不同東西

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

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

Sobel Filter

3	0	-3
10	0	-10
3	0	-3

Scharr Filter

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

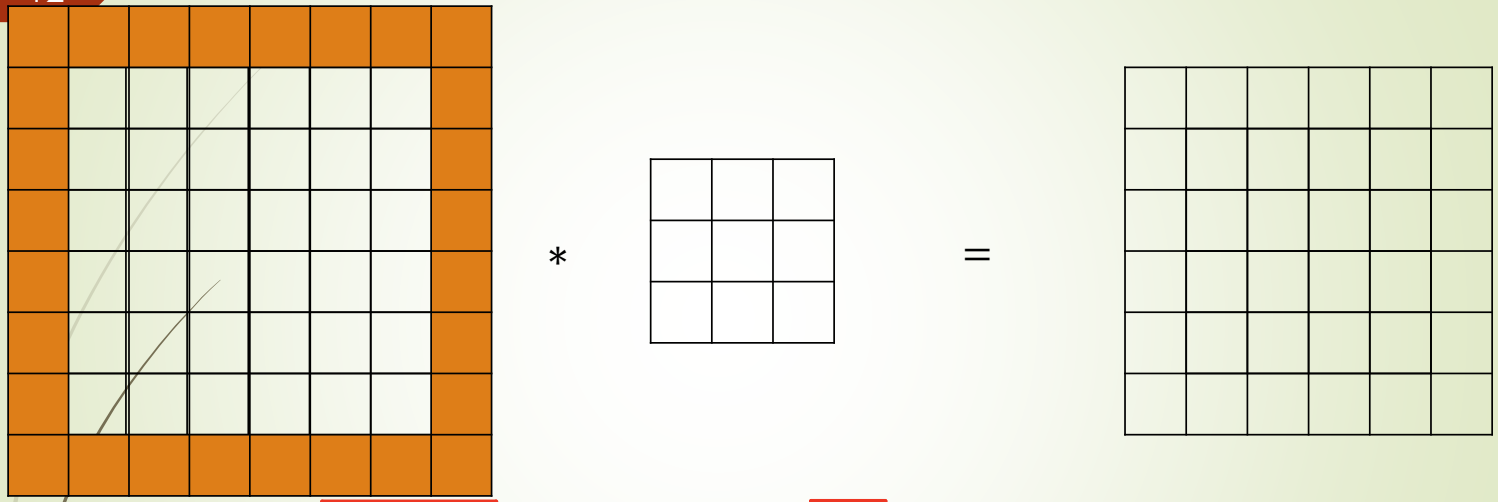
$*$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$=$

Padding

如果要保持原本的圖片大小



Size of input data: $n \times n$, Size of filter: $f \times f$

Size of output: $(n - f + 1) \times (n - f + 1)$

Example: $6 - 3 + 1 = 4$, hence 4×4, size is reduced.

中間比較常使用，加上padding讓原本size跟圖片一樣大小

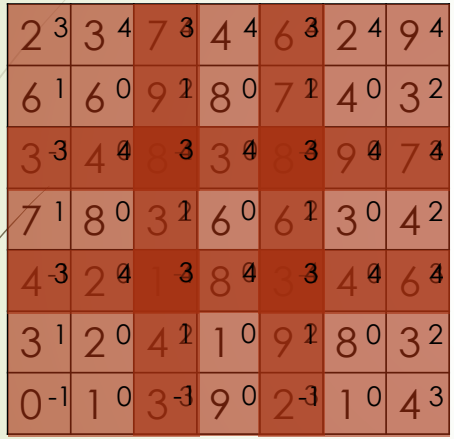
Use padding, extra border of 1 all around, ($p = 1$) gives output of

$(n + 2p - f + 1) \times (n + 2p - f + 1) = 6 \times 6$ (same as original data size)

Valid and Same Convolutions

- "**Valid**" (no padding): $n \times n * f \times f \rightarrow (n-f+1) \times (n-f+1)$ 有縮小
  - $6 \times 6 * 3 \times 3 \rightarrow 4 \times 4$
- "**Same**": Pad so that output size is the same as the input size Input與output一樣
  - $n + 2p - f + 1 = n \Rightarrow \boxed{p = \frac{f-1}{2}}$ 透過filiter決定padding
  - $f = 3, p = \frac{3-1}{2} = 1,\ or\ \ f = 5, p = 2$

Strided Convolutions



7 x 7

*

3	4	4
1	0	2
-1	0	3

3 x 3

=

圖片相較於走1次的小

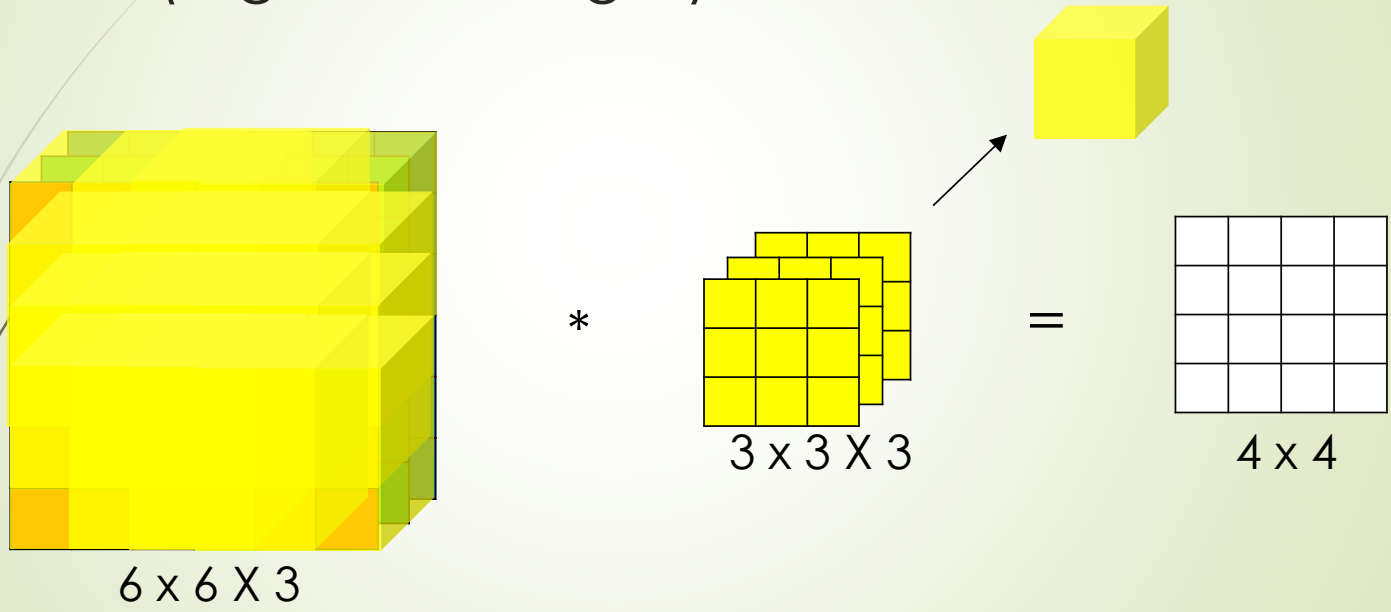
91	100	83
69	91	127
44	71	74

Stride = 2 每次走兩個pixel

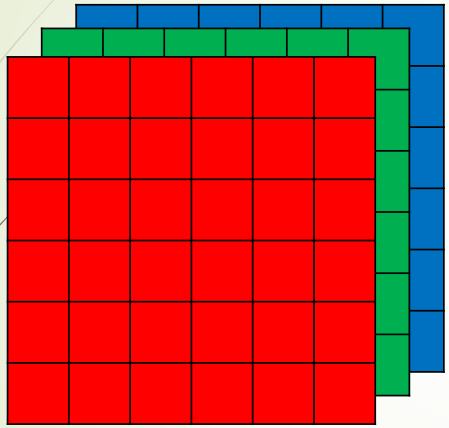
$$n \times n * f \times f \ (padding\ p, stride\ s) \Rightarrow \lfloor \frac{n+2p-f}{s} + 1 \rfloor \times \lfloor \frac{n+2p-f}{s} + 1 \rfloor$$

$$(7+0-3)/2+1 = 4/2+1 = 3$$

Convolutions over volumes (e.g., RGB images)



Multiple Filters



6 x 6 x 3

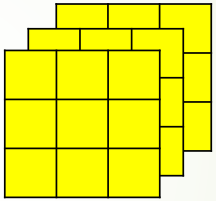
$n \times n \times n_c * f \times f \times n_c$

$\Rightarrow (n - f + 1) \times (n - f + 1) \times n_f$

$6 \times 6 \times 3 * 3 \times 3 \times 3$

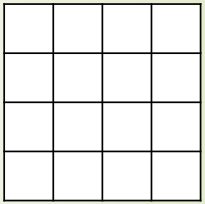
$\Rightarrow 4 \times 4 \times 2\ (\#filters)$

*



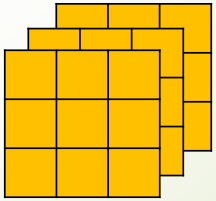
3 x 3 x 3

=



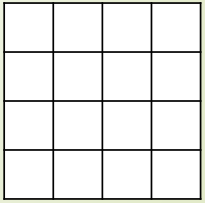
4 x 4

*



3 x 3 x 3

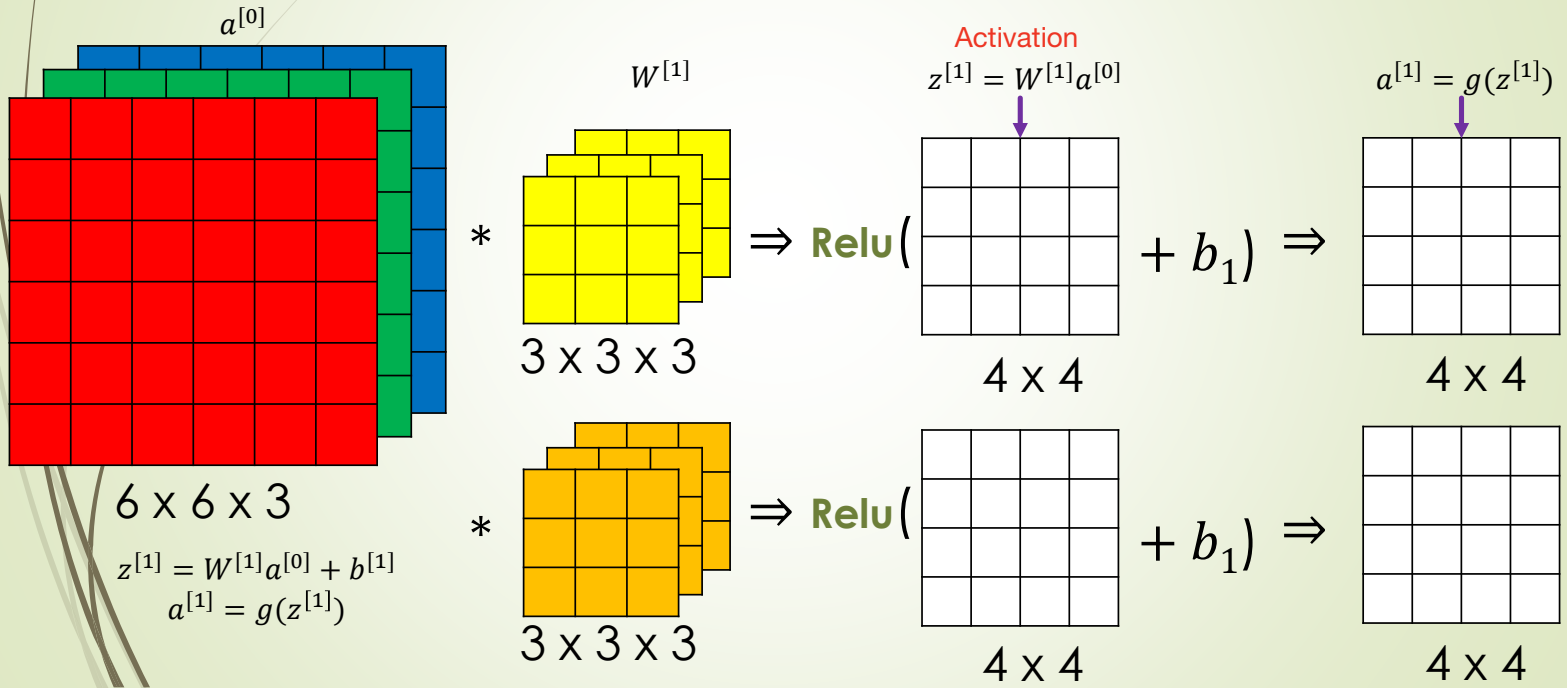
=



4 x 4

One Convolutional Layer

One Layer of Convolutional Neural Network

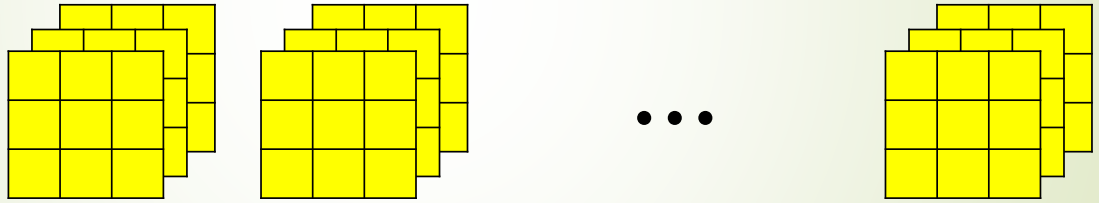


$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$

$a^{[1]} = g(z^{[1]})$

Number of Parameters in One Layer

- If you have 10 filters that are 3 x 3 x 3 in one layer of a neural network, how many parameters does that layer have?



- (3 x 3 x 3 + 1) x 10 = 280 parameters
- No matter how large the input image is, the number of parameters is fixed to 280 for 10 filters of size 3 x 3 x 3.

Summary of notation

If layer 1 is a convolution layer:

- $f^{[l]} = filter\ size$
- $p^{[l]} = padding$
- $s^{[l]} = stride$
- $n_C^{[l]} = number\ of\ filters$
- Each filter is: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]}$
- Input Size: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$
- Output Size: $n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$
- $n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$
- $n_W^{[l]} = \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$

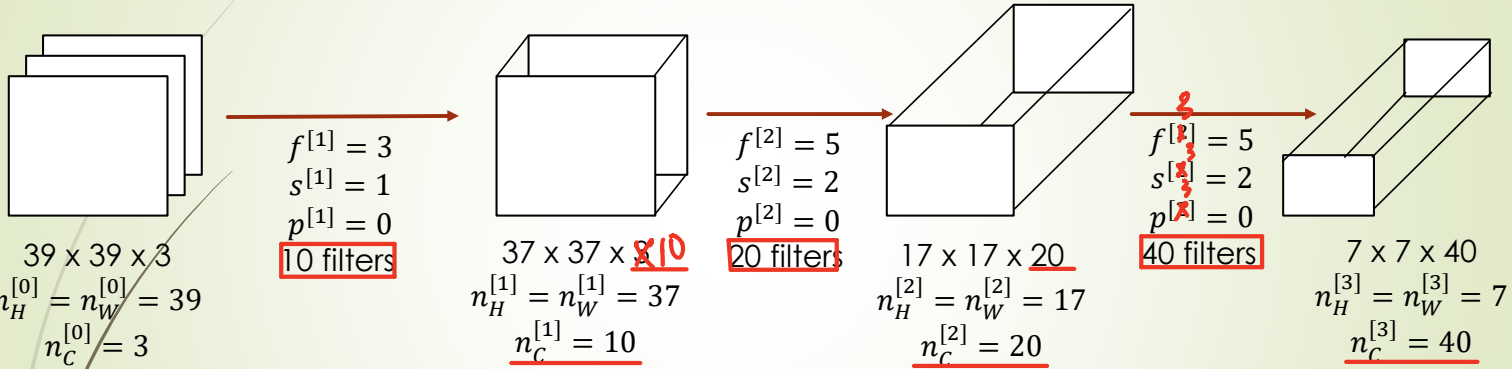
Activations: $a^{[l]} \Rightarrow n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}$

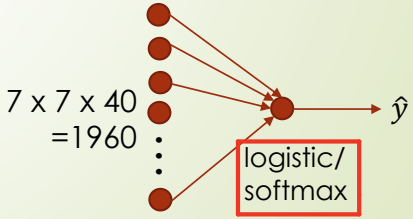
Bias: $n_C^{[l]}$

Example ConvNet

圖片變小，channel變多。
因為要辨識物件



$$\frac{n + 2p - f}{s} + 1 = \frac{39 + 0 - 3}{1} + 1 = 37$$



Types of layer in a convolutional network

- Convolutional (CONV)
- Pooling (POOL)
- Fully connected (FC)

Pooling Layer

Pooling Layer: Max Pooling

4個中最大

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

相素太大找最大的值縮小參數

Hyperparameters

***f* = 2**

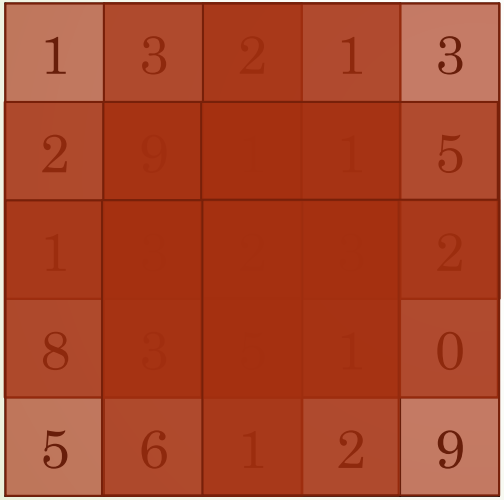
***s* = 2**

Max Pooling

9	2
6	3

No parameters!

Pooling Layer: Max Pooling



5 x 5

Hyperparameters

***f* = 3**

***s* = 1**

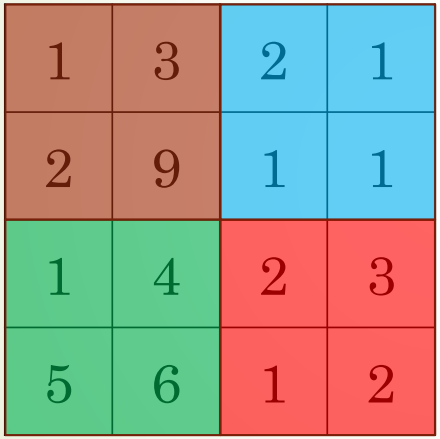
Max Pooling

9	9	5
9	9	5
8	6	9

3 x 3

Note: For multiple channels, the above max pooling is done for each channel

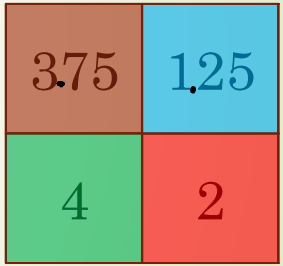
Pooling Layer: Average Pooling



Hyperparameters

$f = 2$

$s = 2$



Summary of Pooling

Pooling:減少size

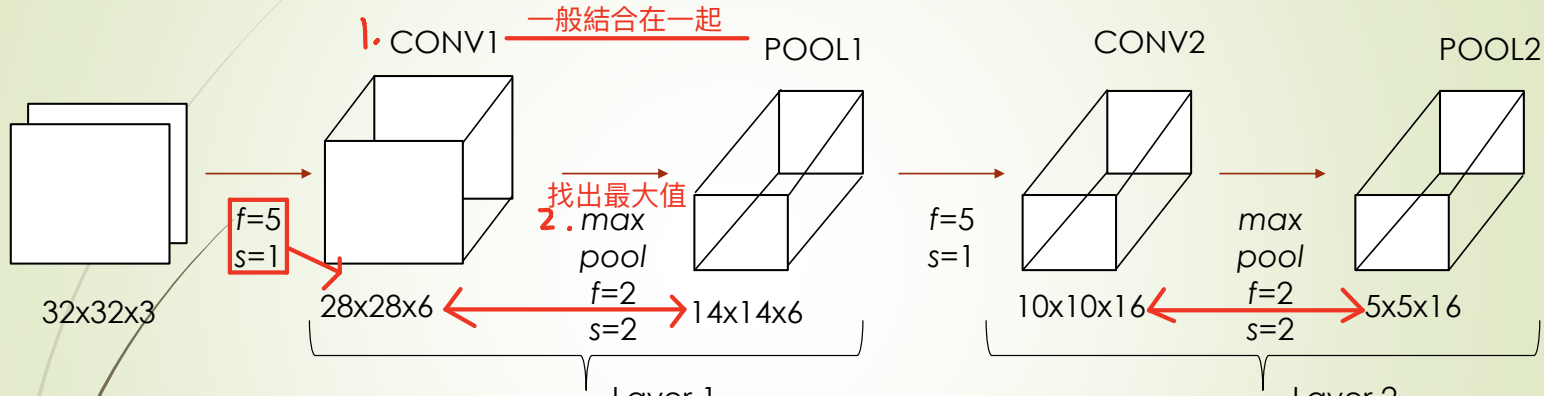
Padding:維持大小

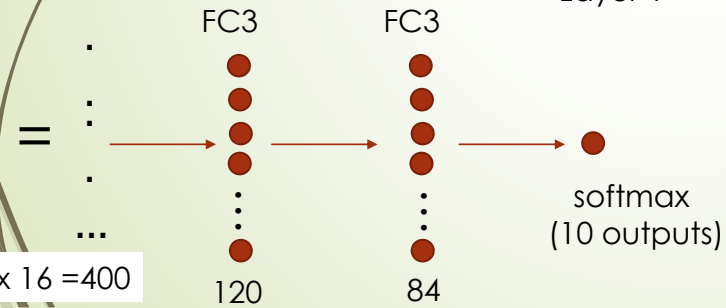
Hyperparameters

- f: filter size
- s: stride
- Max or average pooling
- Usually, p = 0, no padding

$$n_H \times n_W \times n_C \longrightarrow \left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times n_C$$

Neural Network Example (aka LeNet-5)





Notes:

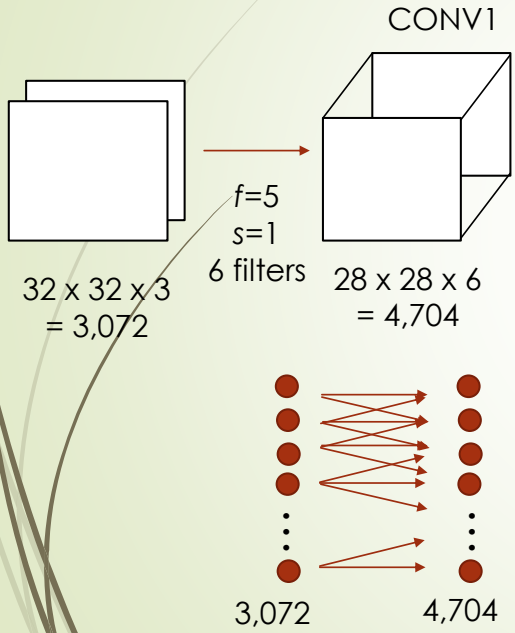
- Look up literatures and use the proven useful set of hyperparameters
- Image size ($n_H \times n_W$) decreases, while the number of channels n_C increases
- CONV-POOL-CONV-POOL-FC-FC-FC-SOFTMAX 隨機取點

Neural Network Example

	Activation Shape	Activation Size	# Parameters
Input	(32, 32, 3)	3,072	0
CONV1 (f=5, s=1)	(28, 28, 8)	6,272	208
POOL1	(14, 14, 8)	1,568	0
CONV2 (f=5, s=1)	(10, 10, 16)	1,600	416
POOL2	(5, 5, 16)	400	0
FC3	(120, 1)	120	48,001
FC4	(84, 1)	84	10,081
Softmax	(10, 1)	10	841

隨機取點

Why Convolutions work?



6 x (5 x 5 + 1) = 156 parameters

縮小參數

3,072 x 4,704 ≈ 14M

Large Difference!!

Why Convolutions work?

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

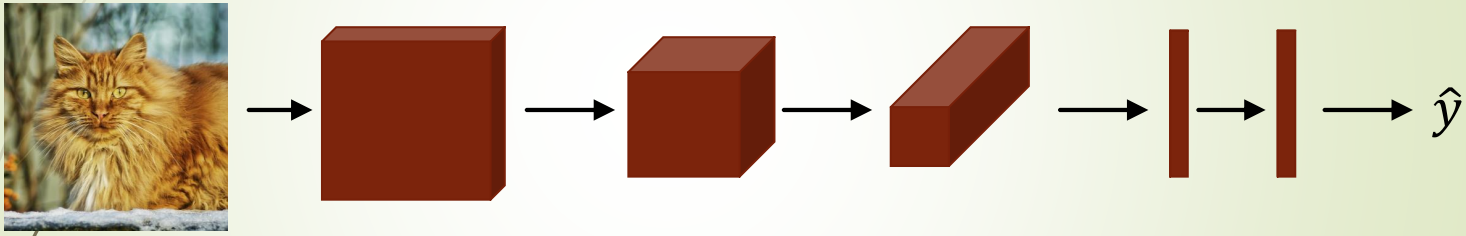
=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

- **Parameter Sharing**: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image 使用同一個filiter，可以移植到其他部分
- **Sparsity of connections**: In each layer, each output value depends only on a small number of inputs 一個像素點會與附近有相關

Putting it together

Training set $(x^{(1)}, y^{(1)})$ … $(x^{(m)}, y^{(m)})$.



Cost $J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Use gradient descent to optimize parameters to reduce J

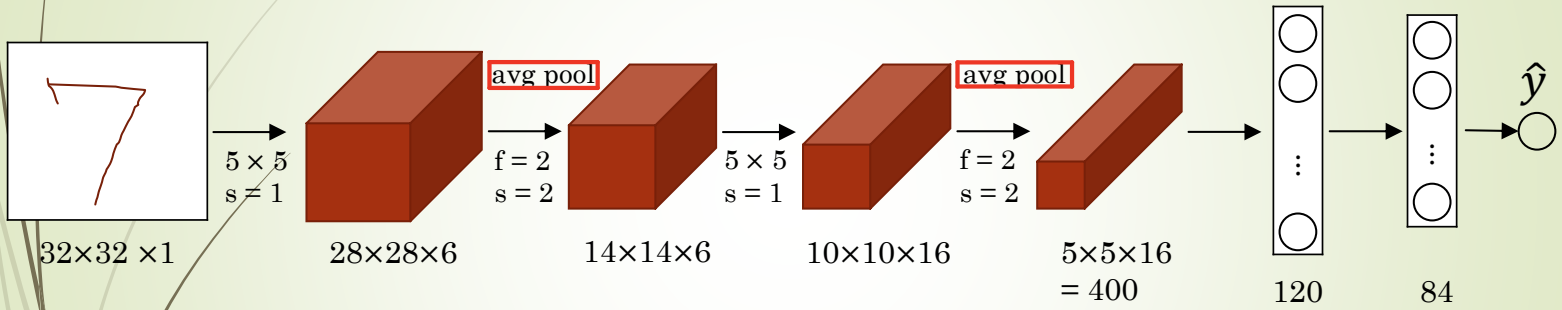
Case Studies

Le-Net5, AlexNet, VGG-16, ResNet

Why look at case studies?

- Often a trained NN can work also quite good in another application!
- Can use transfer learning
- Classic Networks
  - LeNet-5
  - AlexNet
  - VGG
- ResNet (Residual Network) with 152 layers (very deep)
- Inception

LeNet-5 像素辨識

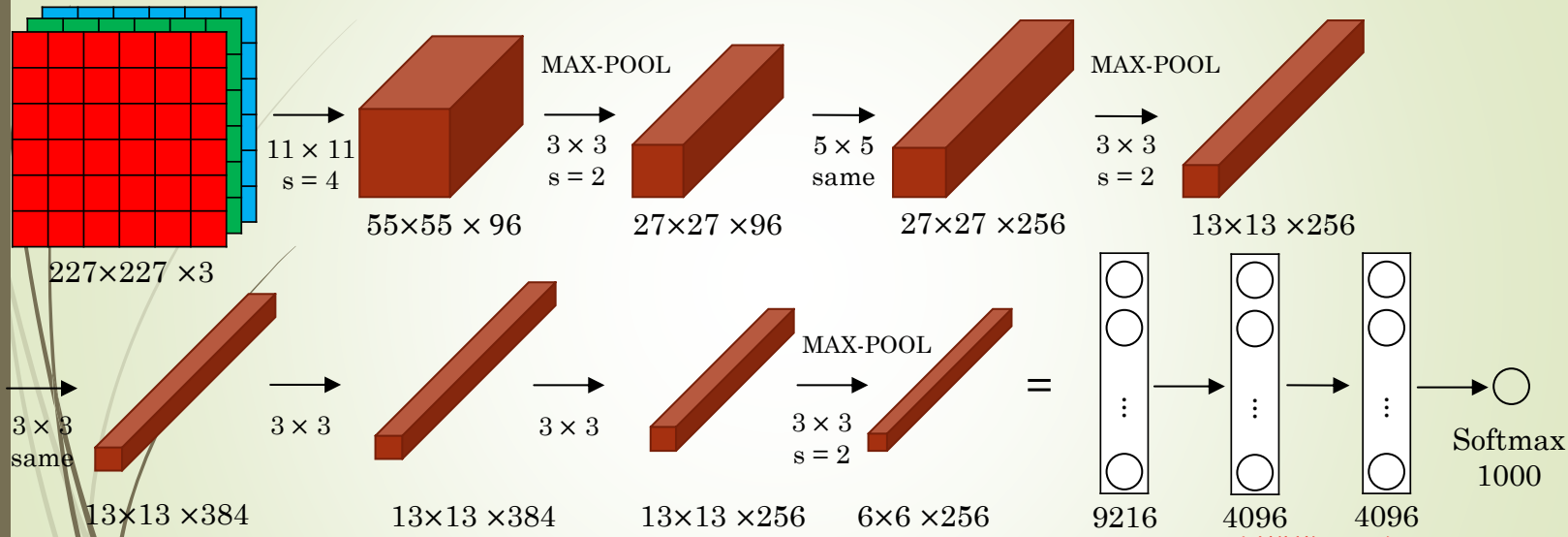


- 60K parameters (small by modern standards)
- n_H, n_W decreased with layers, n_C increased
- CONV – POOL – CONV – POOL – FC – FC - Output

Softmax

[LeCun et al., 1998. Gradient-based learning applied to document recognition]

AlexNet 辨識的東西越來越多



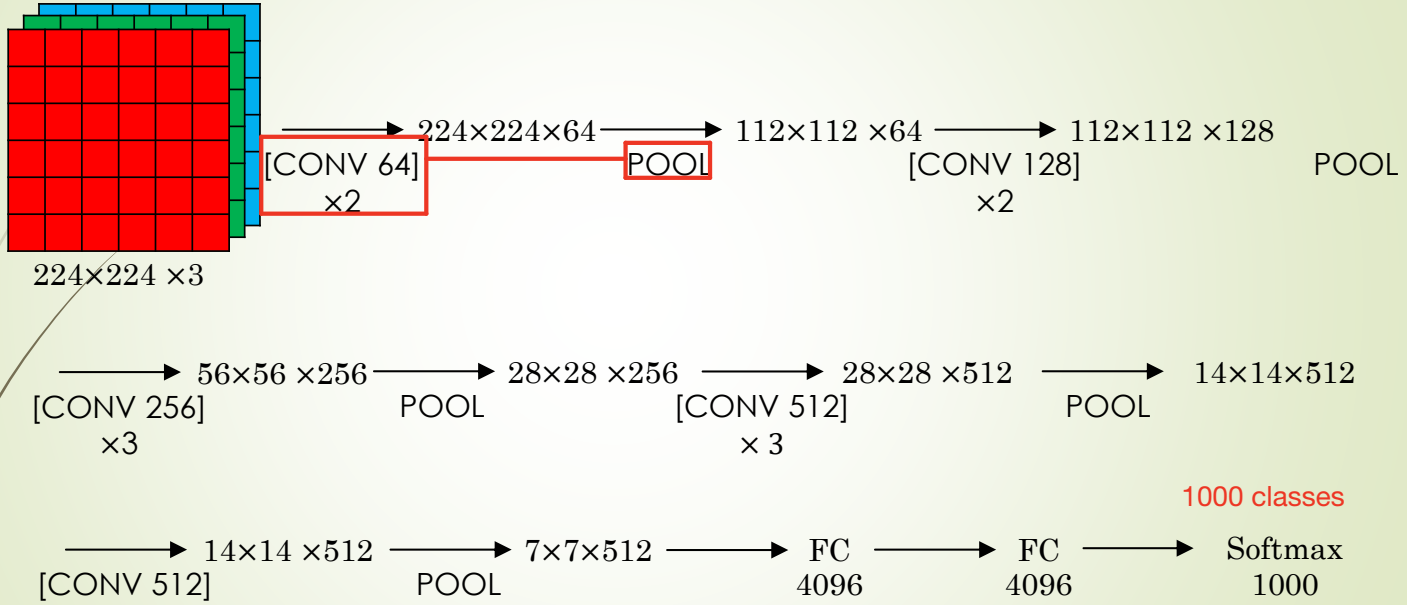
可以辨識1000個classes

- Similar to LeNet-5, but much bigger (60M parameters)
- Instead of Sigmoid/Tanh, AlexNet used ReLU
- Multiple GPUs: in the olden days, it was difficult to train on multiple GPUs, it is fine now!
- Local Response Normalization (LRN): less useful now!

[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

VGG-16

CONV = 3×3 filter, s = 1, same MAX-POOL = 2×2 , s = 2

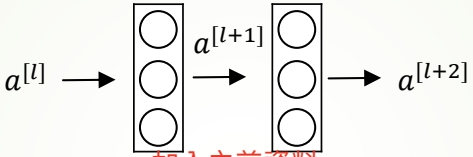


- ~138M parameters, very uniform, very large network
- VGG-19 a bigger network, almost similar to VGG-16 in performance

[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

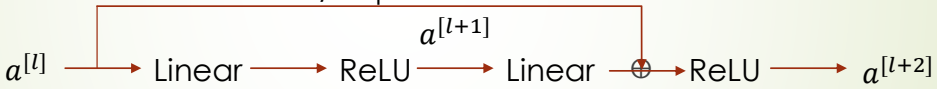
Residual Block (for Residual Networks or ResNets)

Layer太多不好，前面學到的可能忘掉



加入之前資料

short cut/ skip connection



$$z^{[l+1]} = W^{[l+1]}\, a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]})$$

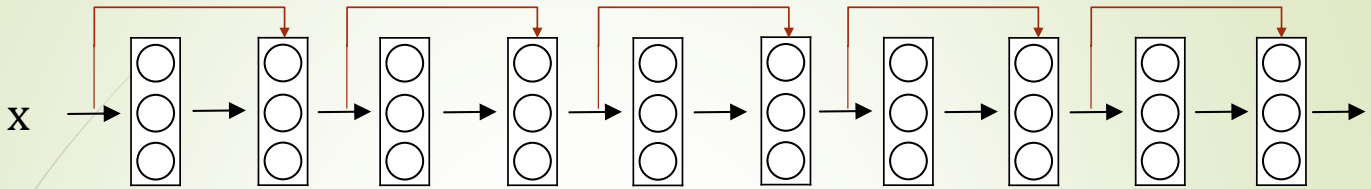
$$z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad \cancel{a^{[l+2]} = g(z^{[l+2]})}$$

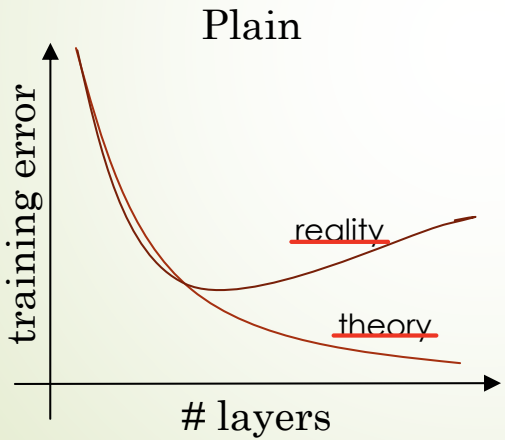
$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

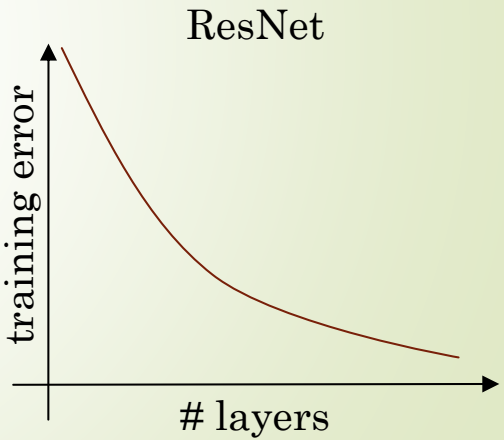
[He et al., 2015. Deep residual networks for image recognition]

Residual Network

第一個超過100layer





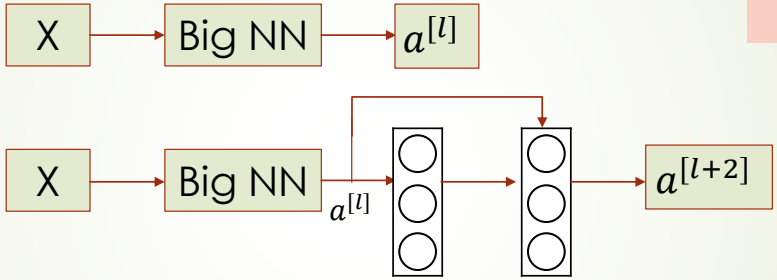


[He et al., 2015. Deep residual networks for image recognition]

Why do residual networks work?

Identity function is easy for Residual Block to learn!!!

之前學的有保留下來



- ReLU, where $a \geq 0$

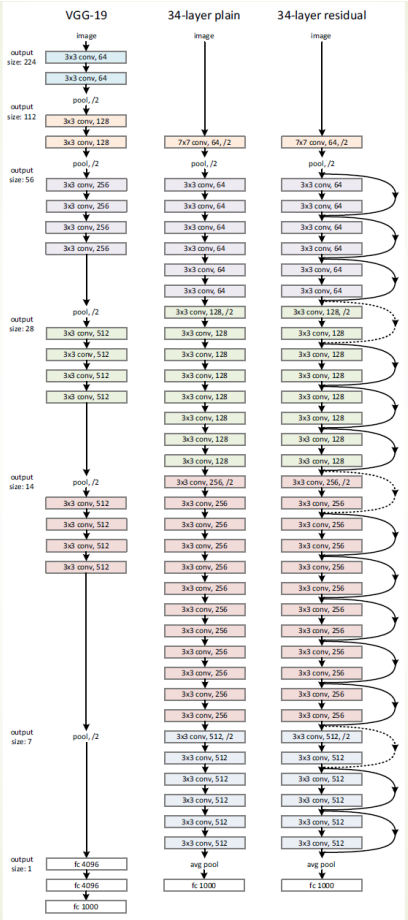
256

$\mathfrak{R}^{256\times128}$

- $a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right) = g\left(W^{[l+2]}a^{[l+1]} + W_s \;\; a^{[l]}\right) =$ $g\left(a^{[l]}\right) = a^{[l]}, \quad \text{if } W^{[l+2]} = b^{[l+2]} = 0$

128

ResNet



- Lots of 3 x 3 same CONV
  - Dimension preserved
- Thus, $z^{[l+2]} + a^{[l]}$ makes sense

[He et al., 2015. Deep residual networks for image recognition]

Network in network and 1x1 convolution

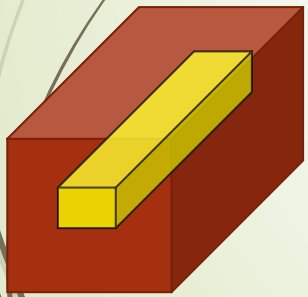
What does a 1x1 convolution do?

1	2	3	6	5	8
3	5	5	1	3	4
2	1	3	4	9	3
4	7	8	5	7	9
1	5	3	7	4	8
5	4	9	8	3	5

6 × 6

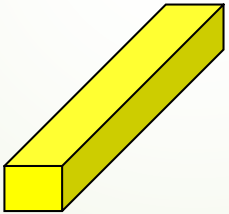
* [2] =

2	4	6			



6 × 6 × 32

*



1 × 1 × 32

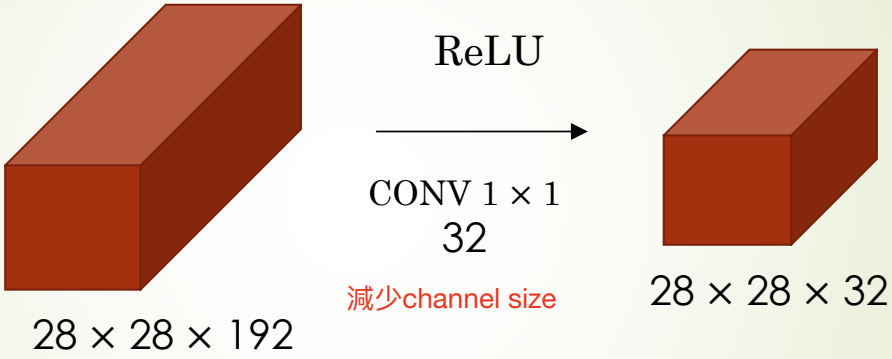
=

6 × 6 × # filters

[Lin et al., 2013. Network in network]

Using 1x1 convolutions

降低深度，但不降低原輸入二維的維度



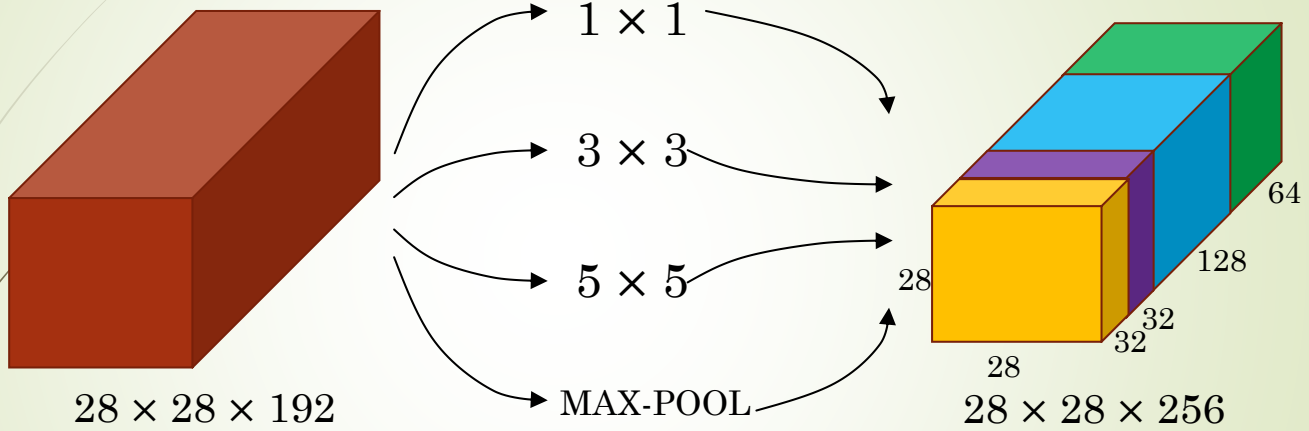
- Number of channels reduced from 192 to 32
- Adds non-linearity (when #channels not reduced)

非線性

[Lin et al., 2013. Network in network]

Inception Network

Motivation for inception network

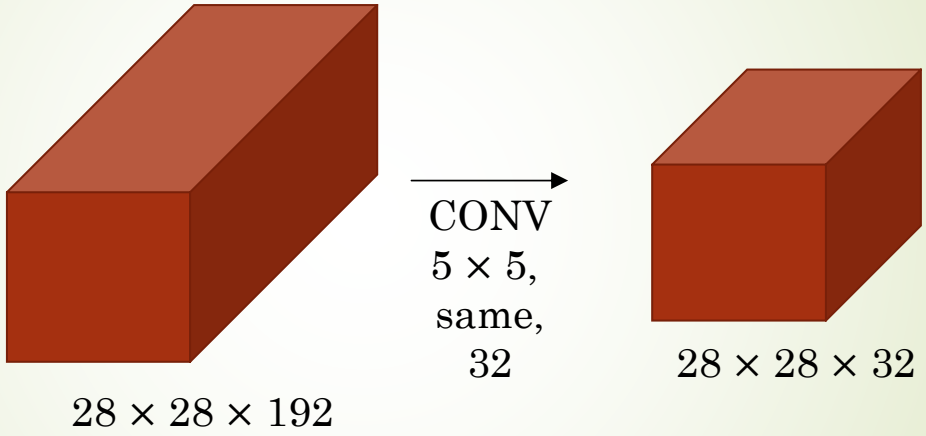


圖片中需要偵測物件大小可能不一

嘗試不同的大小

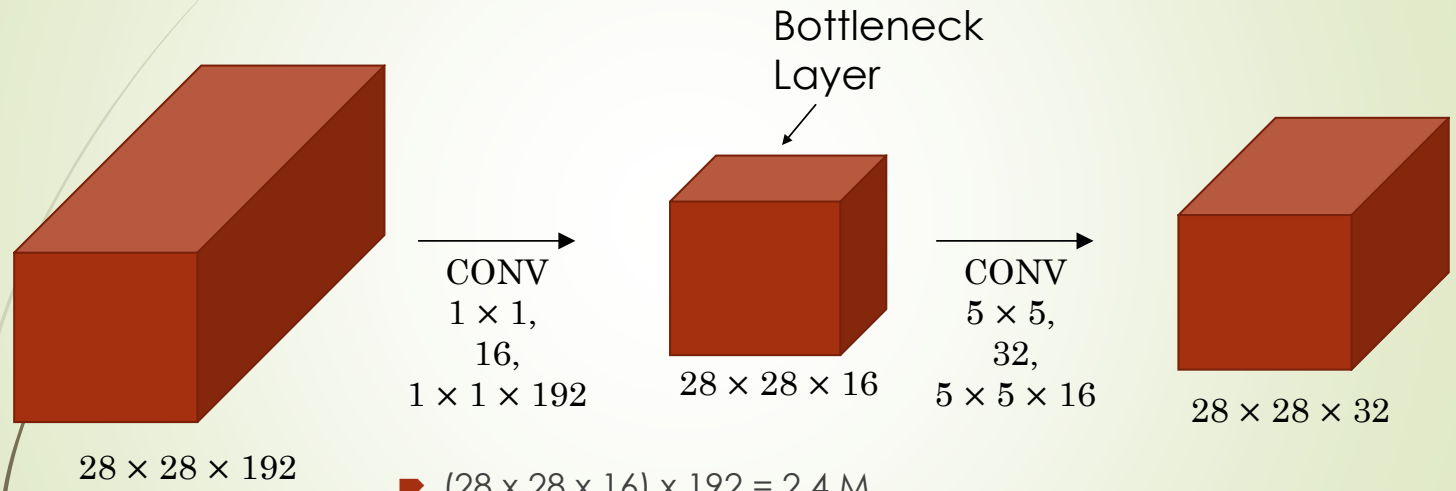
[Szegedy et al. 2014. Going deeper with convolutions]

The problem of computational cost



- 32 filters, each filter is of size 5 x 5 x 192
- #multiplies = (28 x 28 x 32) x (5 x 5 x 192) = 120 M

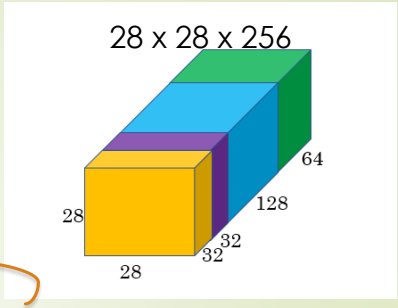
Using 1 x 1 convolution



- (28 x 28 x 16) x 192 = 2.4 M
- (28 x 28 x 32) x (5 x 5 x 16) = 10 M
- 2.4 + 10 = 12.4 M multiplications (1/10 that of original 5 x5
- #additions similar to #multiplications

Inception Network

Inception module



偵測不同size

降低深度

Previous Activation

28 x 28 x 192

1 × 1 CONV

1 × 1 CONV

3 × 3 CONV

1 × 1 CONV

5 × 5 CONV

MAXPOOL 3 × 3,s = 1 same

1 × 1 CONV

28 x 28 x 64

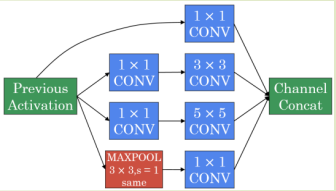
28 x 28 x 128

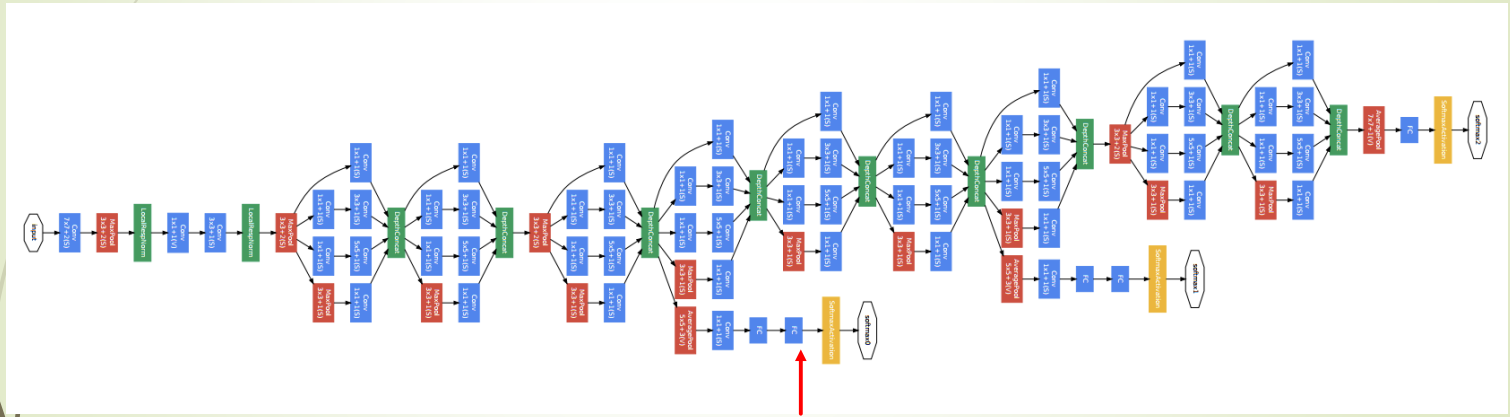
28 x 28 x 32

28 x 28 x 32

Channel Concat

28 x 28 x 256





Side branches uses softmax to produce predictions

Initially, Google developed GooLeNet!

[Szegedy et al., 2014, Going Deeper with Convolutions]

From the Inception Movie

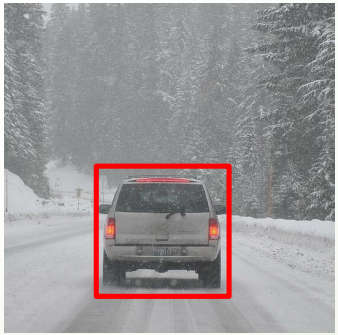
Object Detection

What are localization and detection?

Image classification

“Car”

Classification with localization

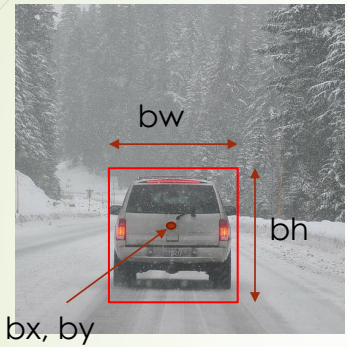


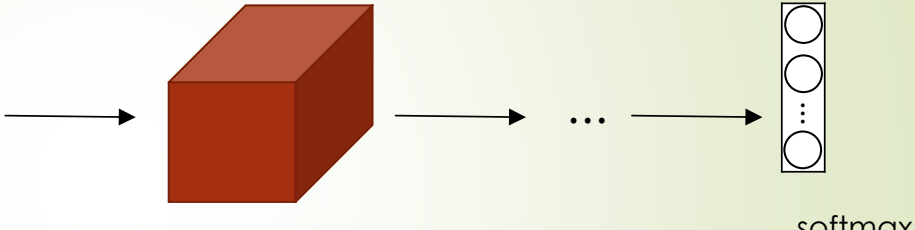
“Car”

Detection

“Multiple Objects”

Classification with Localization





softmax

bounding box
bx, by, bh, bw

1 - pedestrian
2 - car
3 - motorcycle
4 - background

先找出中點
再找出邊長

bx = 0.5
by = 0.7
bh = 0.3
bw = 0.4

Defining the target label y

- 1. pedestrian
- 2. car
- 3. motorcycle
- 4. background

- Need to output bx, by, bh, bw, class label (1-4)

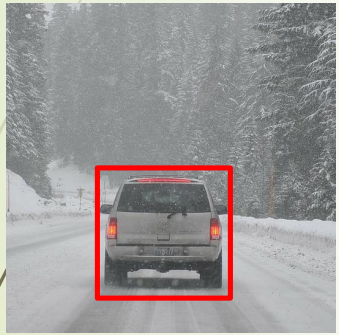
Is there an object?

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad or \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

Don't care.
沒有在object裡

$$\mathcal{L}(\hat{y}, y) = (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \cdots + (\hat{y}_8 - y_8)^2 \quad if\ y_1 = 1$$
$$(\hat{y}_1 - y_1)^2 \quad if\ y_0 = 1$$

Landmark Detection



b_x, b_y, b_h, b_w

偵測情緒

ConvNet

Facial emotion

Body posture

- $l_{1x}, l_{1y}, l_{2x}, l_{2y}, l_{3x}, l_{3y}, \ldots l_{64x}, l_{64y}$: *key landmarks*
- For facial emotions, body postures, etc.
- Landmarks should be consistent across different input images

Car detection example

一個圖片有很多物件

透過CNN偵測物件

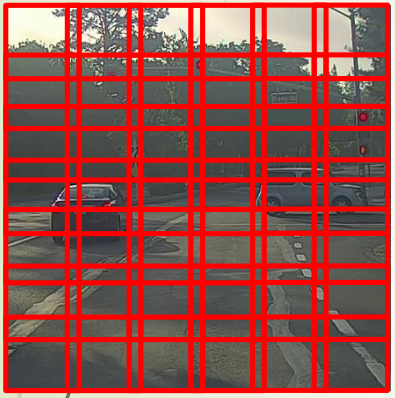
Training set: (closely cropped images)

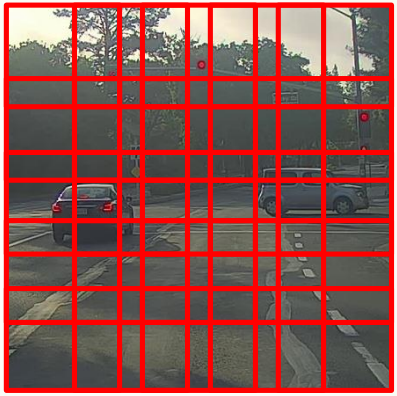
x	y
	1
	1
	1
	0
	0

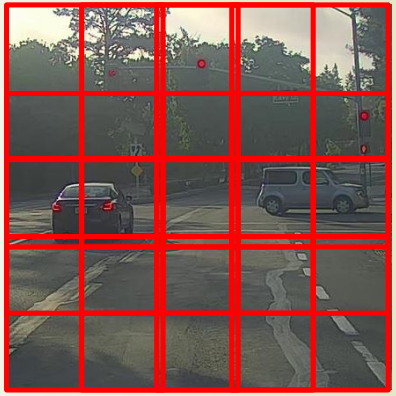
需要裁切圖片

Conv Net → y

Sliding windows detection



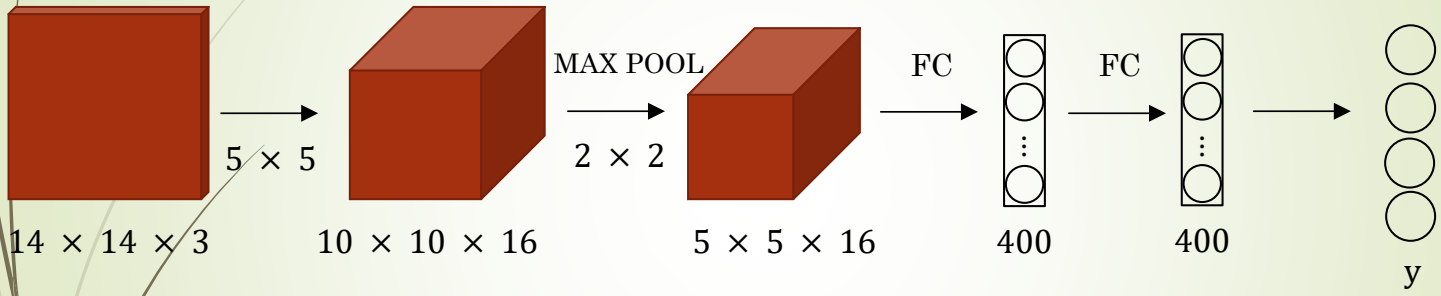


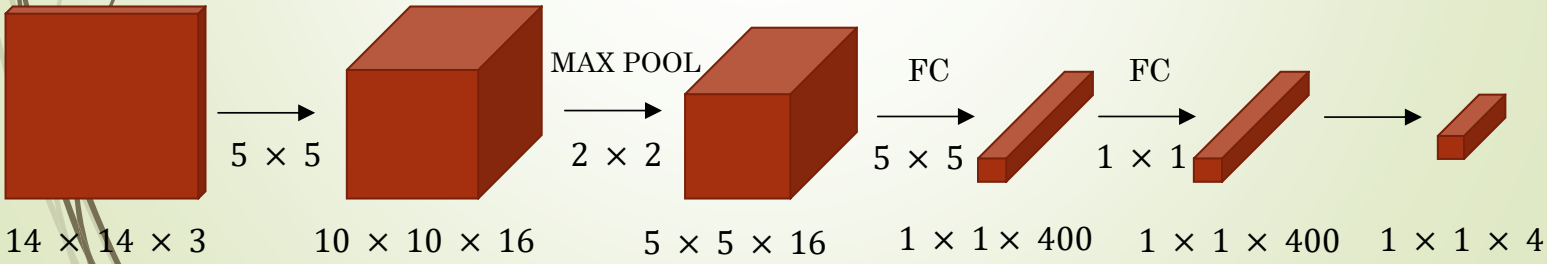


- Computational cost is high due to sliding windows (infeasibly slow)
- Solution: Implement convolutionally!

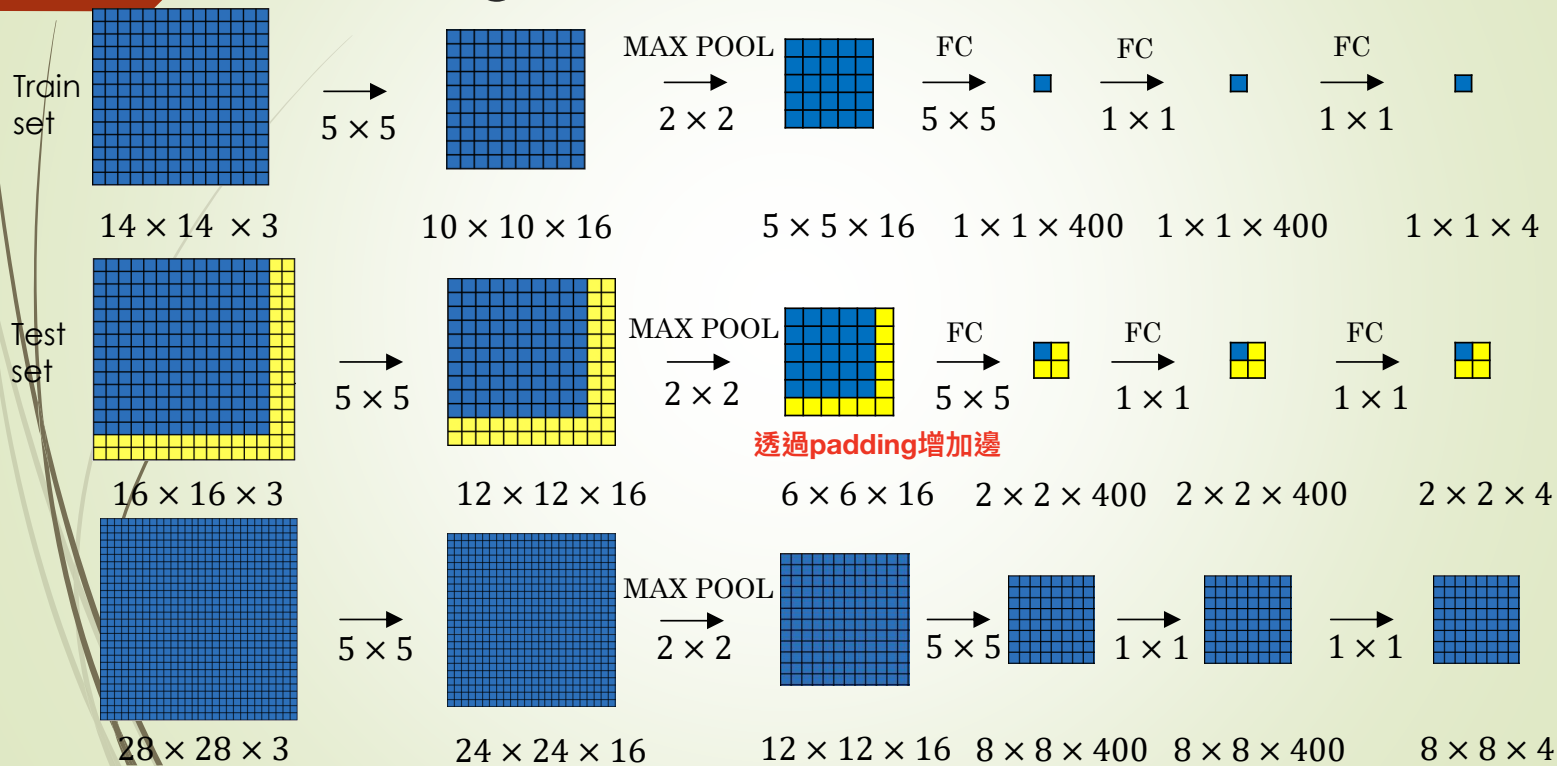
Object Detection: Convolutional Implementation of Sliding Windows

Turning FC layer into convolutional layers

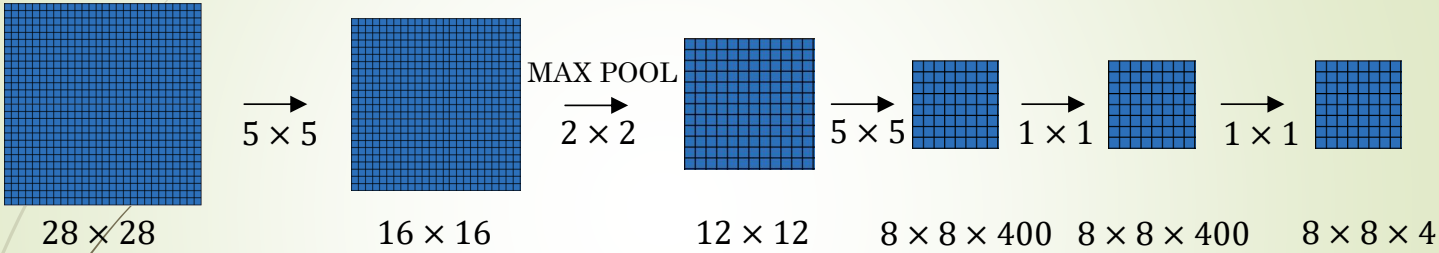




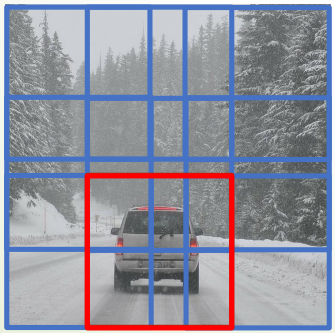
Convolutional Implementation of Sliding Windows

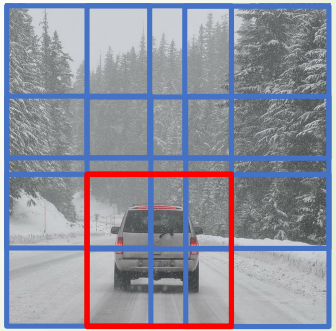


Convolutional Implementation of Sliding Windows



根據filiter偵測車子



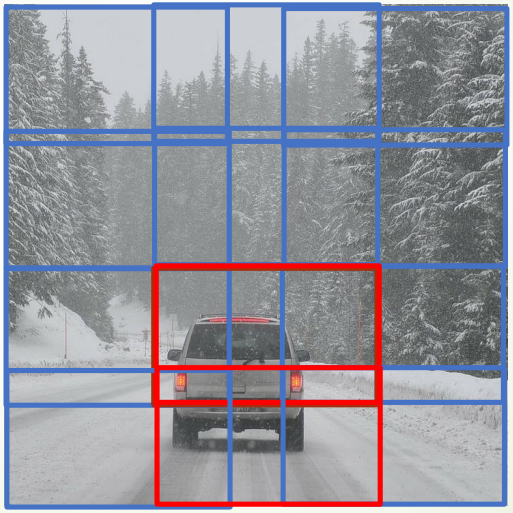


Bounding Box not very accurate!

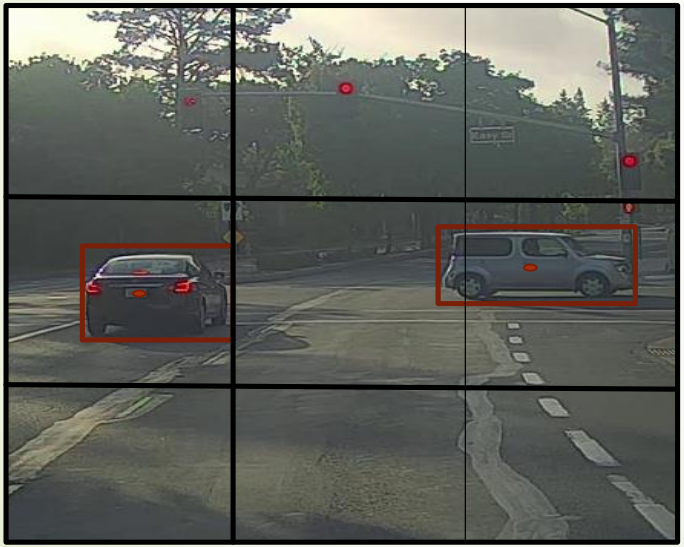
透過許多CNN偵測物件

Output accurate bounding boxes

- None of the boxes matches the car
- How to output more accurate bounding boxes?



YOLO Algorithm (You Only Look Once)



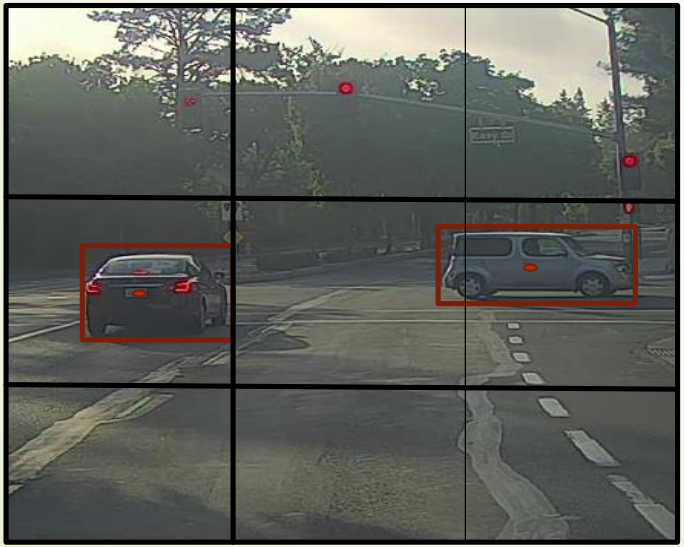
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- Output size: 3 x 3 x 8
  - Can be finer: 19 x 19 x 8
- Object is assigned to only that grid cell that contains the **center** of the object
- Very efficient and popular!

Redmon et al. 2015, You Only Look Once: Unified real-time object detection

Specify the bounding boxes

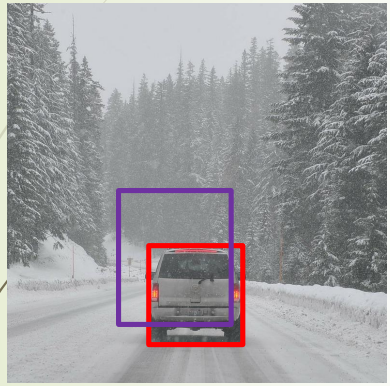
Bound box
在偵測
Car



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.4 \\ 0.3 \\ 0.9 \\ 0.5 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Object Detection: Intersection over union

Evaluating object localization



Intersection over Union (IoU)

= (size of $\square \cap \square$) / (size of $\square \cup \square$)

使用兩個box的交集

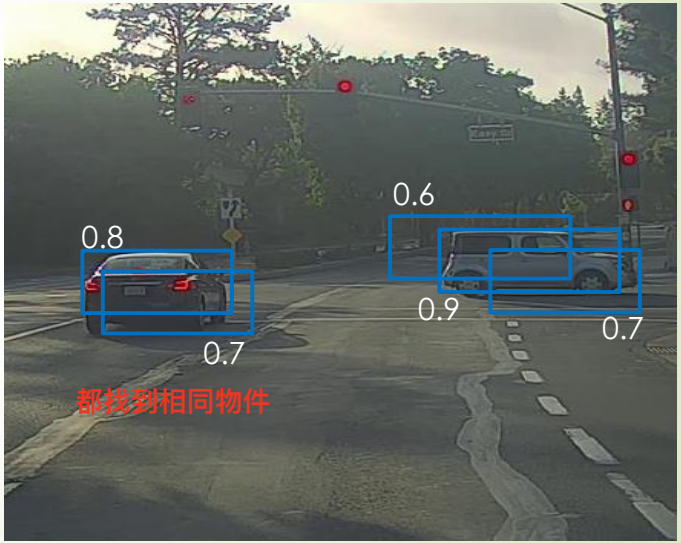
"Correct" if IoU $\geq$ 0.5 如果交集大於0.5

More generally, IoU is a measure of the overlap between two bounding boxes.

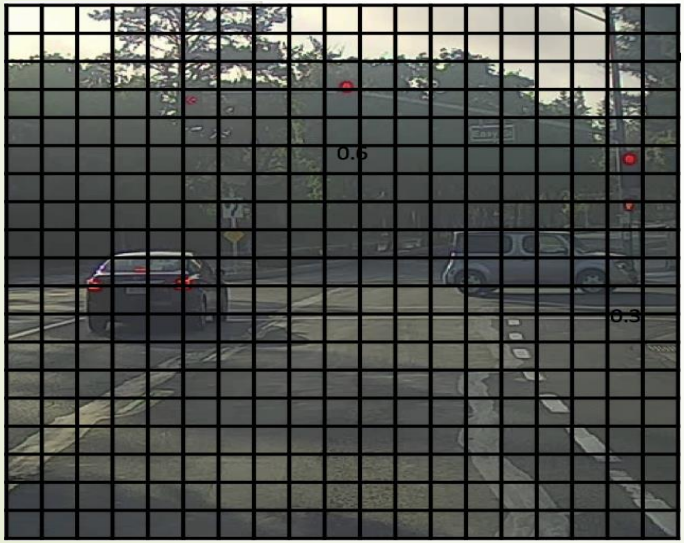
Object Detection
Non-Max Suppression

Non-max suppression example

- More than one grid cells claim that to have one object (multiple detections) 找出機率大的
- Leave only the grid with the max probability p_c, and suppress the grids with non-max claims



Non-max suppression algorithm



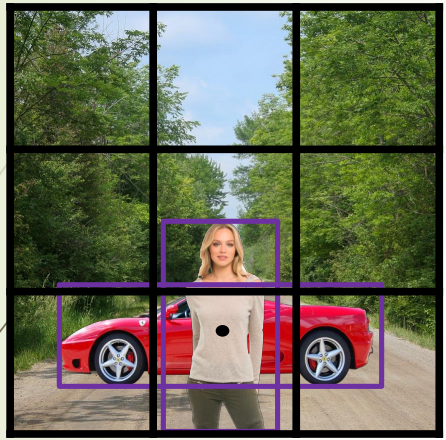
19 x 19

YOLO
找bounding box
使用IOU
最後使用Non-max 方法

- Each output prediction is $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$
- Discard all boxes with $p_c \leq 0.6$.
- While there are any remaining boxes:
  - Pick the box with the **largest p_c**. Output that as a prediction.
  - Discard any remaining box with **IoU ≥ 0.5 with the box output in the previous step**.

Object Detection
Anchor Boxes

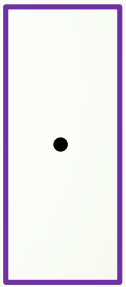
Overlapping objects 兩個物件重疊



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

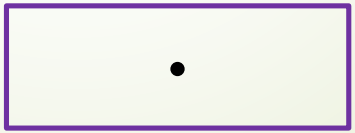
- Problem: Centers overlap and cannot detect two objects using previous method
- Method: Use 2 anchor boxes 使用兩個 anchor box

Anchor box 1:



兩個物件不同大小可以使用

Anchor box 2:



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

For anchor box 1

For anchor box 2

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Anchor box algorithm

Previously:

Each object in training image is assigned to grid cell that contains that object's midpoint.

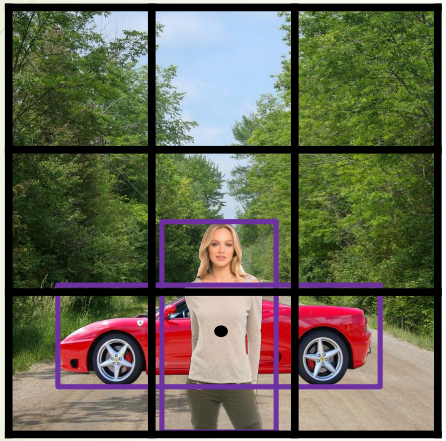
- Previously, size of output y: 3 x 3 x 8
- Now, size of output y: 3 x 3 x 16

With two anchor boxes:

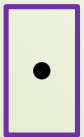
Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

i.e., assigned to (Grid cell, Anchor box)

Anchor box example



Anchor box 1:



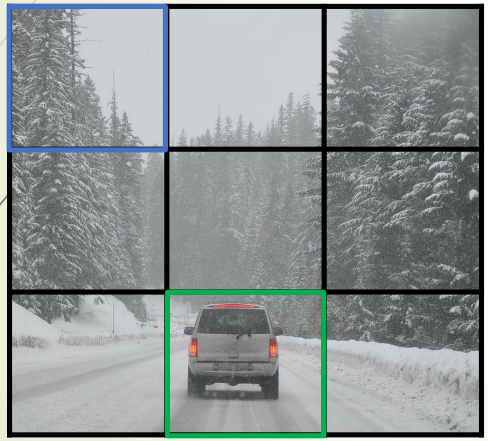
Anchor box 2:

$$y = \begin{matrix} \\ \\ \\ \text{路人} \\ \\ \\ \\ \\ \\ \\ \\ \text{車子} \\ \\ \\ \\ \end{matrix} \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

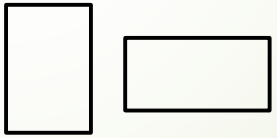
Object Detection
YOLO Algorithm

Training

1 - pedestrian
2 - car
3 - motorcycle



y =



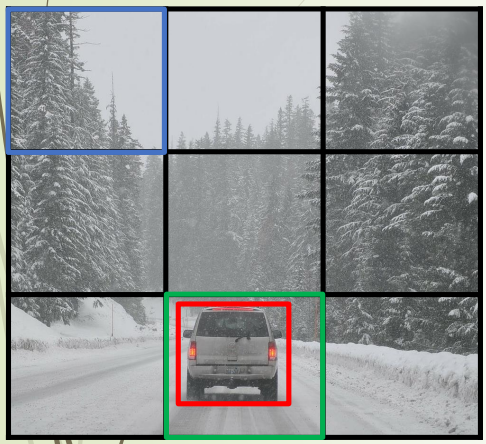
Anchor box 1 Anchor box 2

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

y is $3 \times 3 \times 2 \times 8$

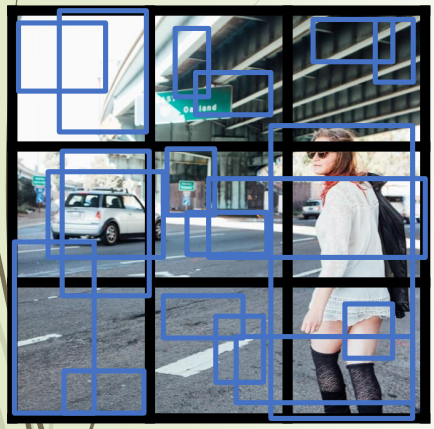
#anchors 5+#classes

Making Predictions



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Outputting the non-max suppressed outputs



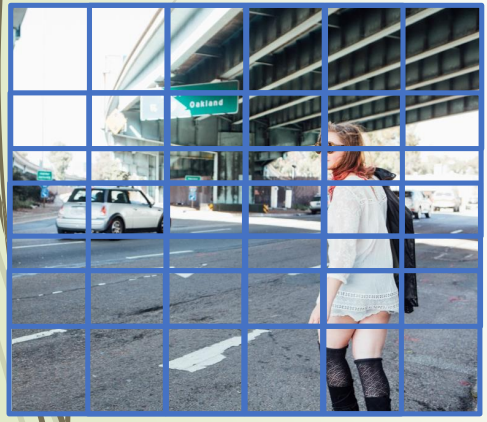
- For each grid call, get 2 predicted bounding boxes.
- Get rid of low probability predictions.
- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

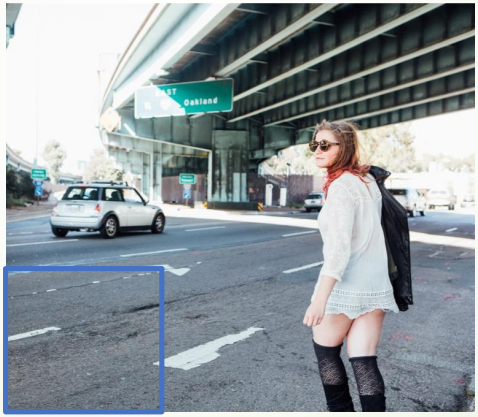
使用不同box
刪除機率低的IOU
產出

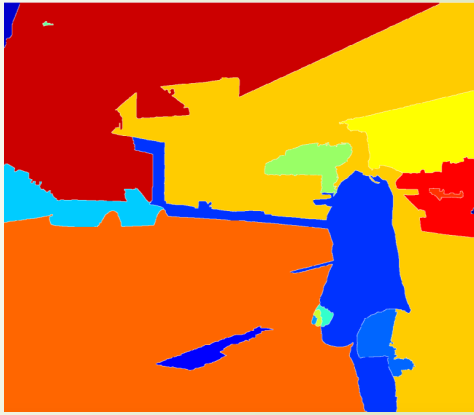
Region Proposal

Region proposal: R-CNN

只偵測藍色的部分







[Girshik et. al, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation]

Faster algorithms

R-CNN: Propose regions. Classify proposed regions one at a time. Output label + bounding box.

Fast R-CNN: Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions.

Faster R-CNN: Use convolutional network to propose regions.

[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]
[Girshik, 2015. Fast R-CNN]
[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks]