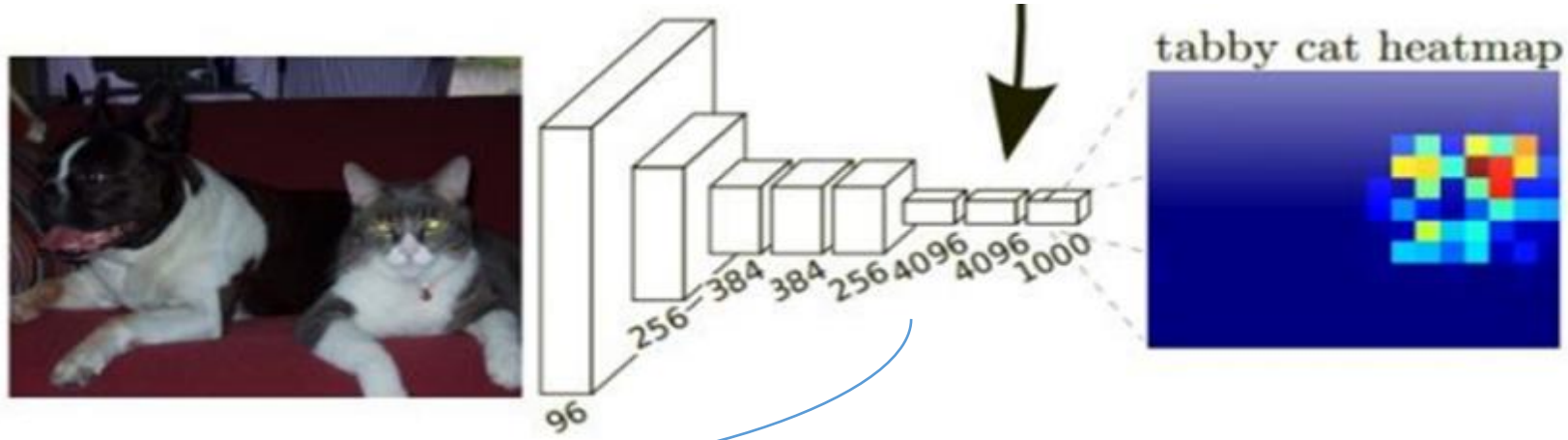
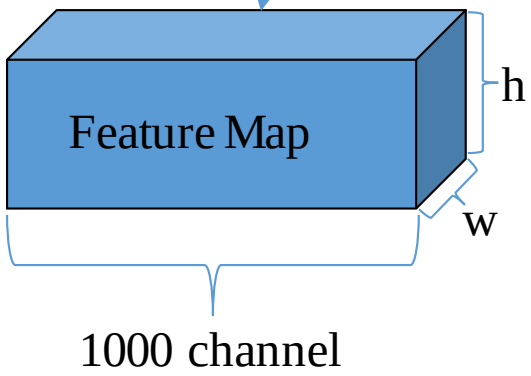
深度學習:
影像分割、物件偵測

黃志勝
義隆電子 人工智慧研發部
國立陽明交通大學 AI學院 合聘助理教授
台北科技大學 業師

Deep Learning: Feature Extractor

- Feature map可以做什麼?





- 這張圖是貓還是夠 (Classification)
- 框出圖片內有貓和狗的位置 (Object detection)
- 從框出的物件(貓和狗)中的描繪出實際物件的輪廓 (Instance Segmentation)
- 把圖片描繪出物件(貓和狗)的輪廓 (Semantic Segmentation)

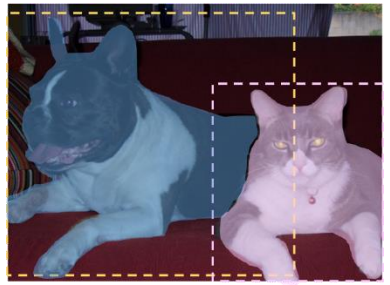
What is image segmentation?

- This is the process of dividing an image into multiple segments.
- Every pixel in the image is associated with an object type.

Semantic Segmentation

pixel-by-pixel
藍色Mask是狗
紅色Mask是貓
橘色Mask是沙發
綠色是Background

Instance Segmentation



Object detection後的框框內那些pixel是實際的物件。

What is image segmentation?

Image Segmentation

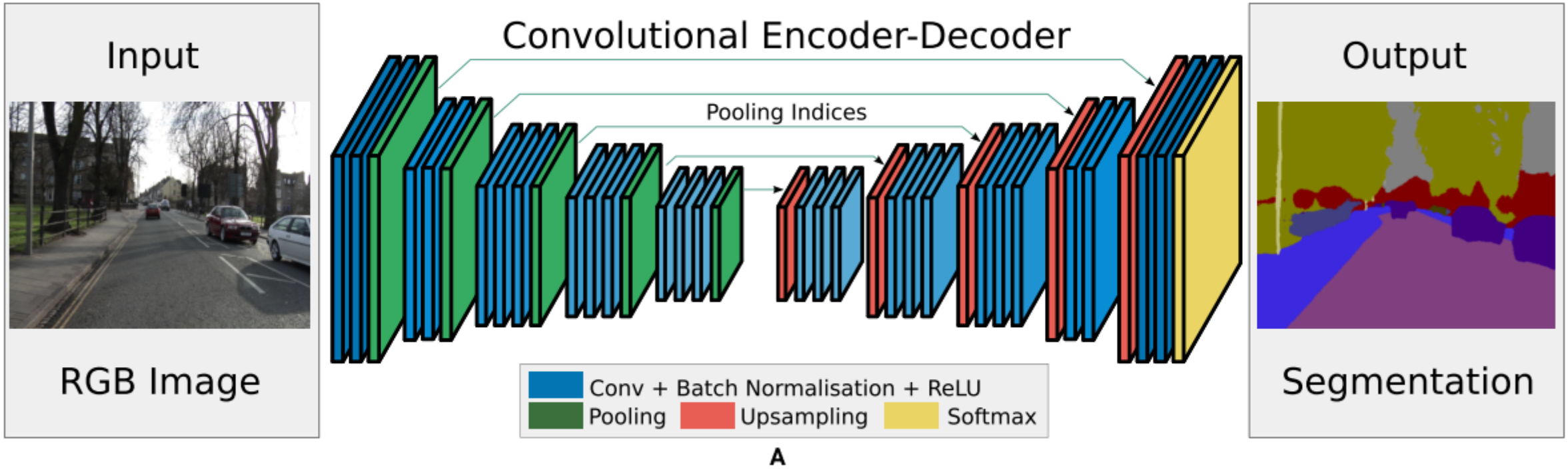
輸入圖片

人眼看到的

電腦看到的

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

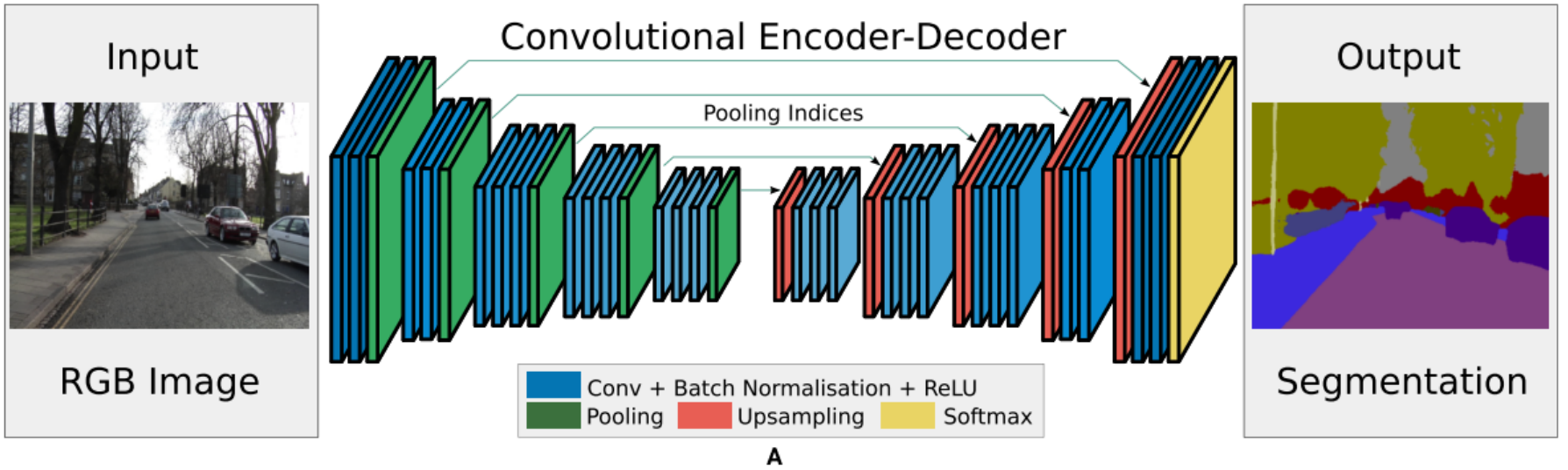
Image segmentation architectures



可原圖大小去訓練和inference:
缺點:計算資源需求大，內存要很夠。
優點: 結果較為準確。

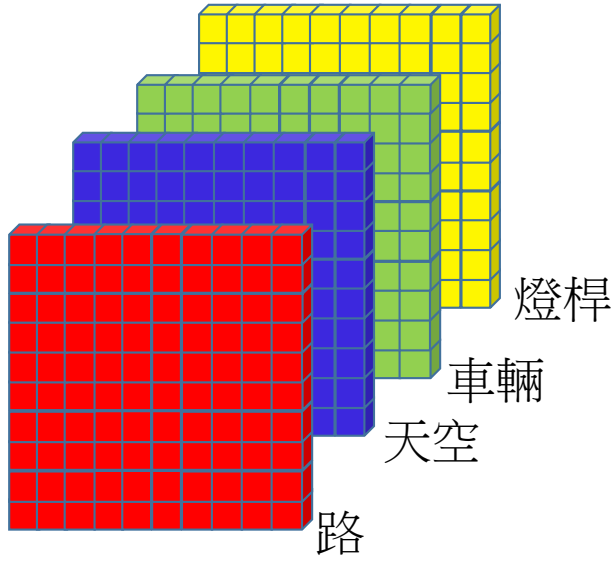
可原圖大小(或resize到固定大小)去訓練，但結果只從1/8大小的結果輸出:
缺點:結果較差。
優點: 快速。

Image segmentation architectures

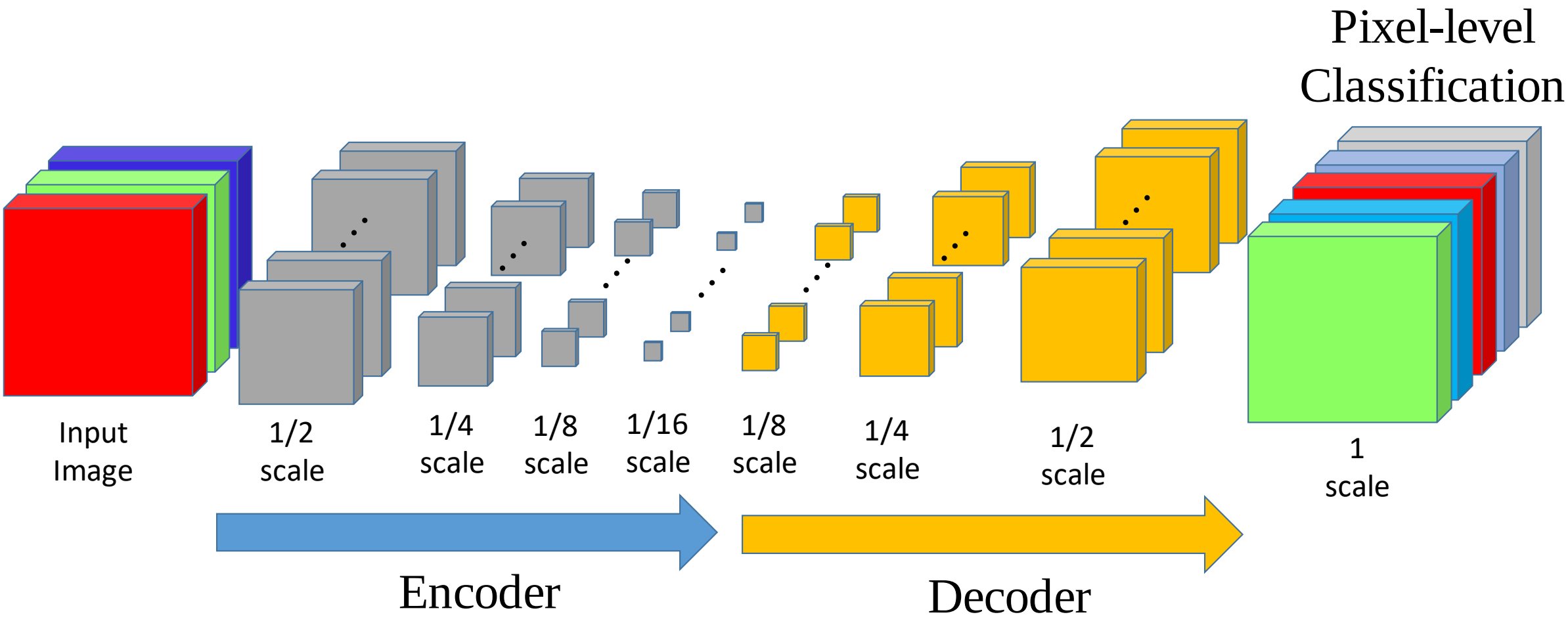


A

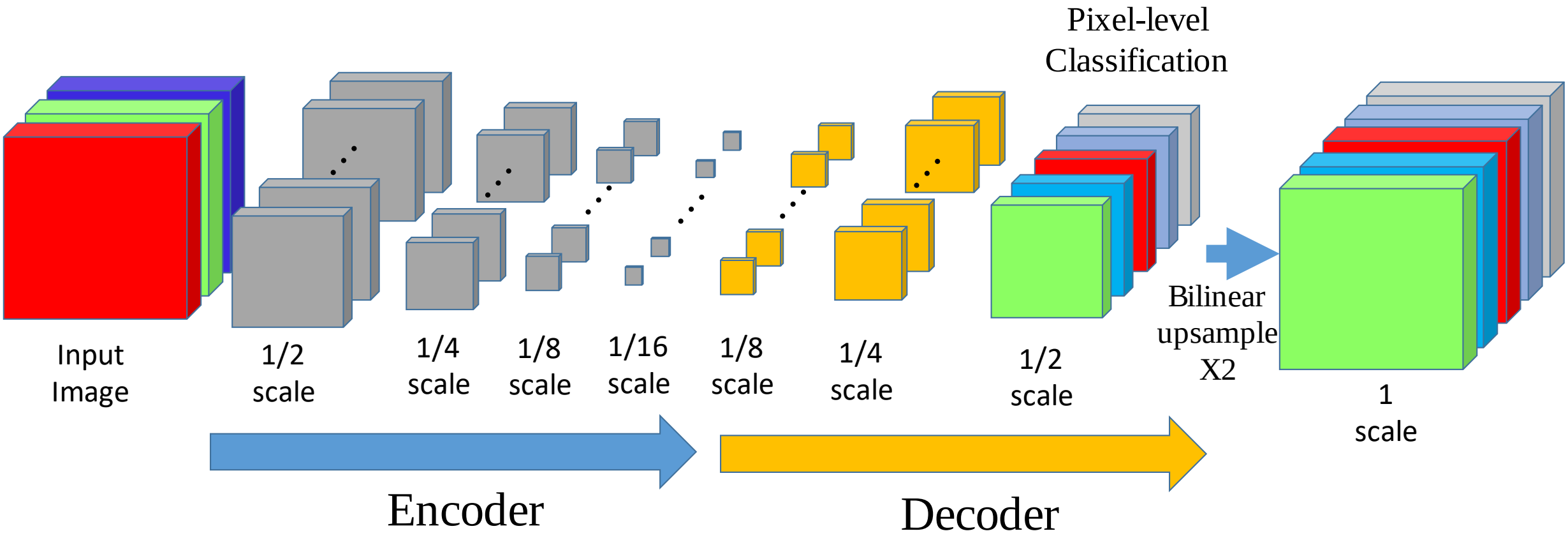
CNN



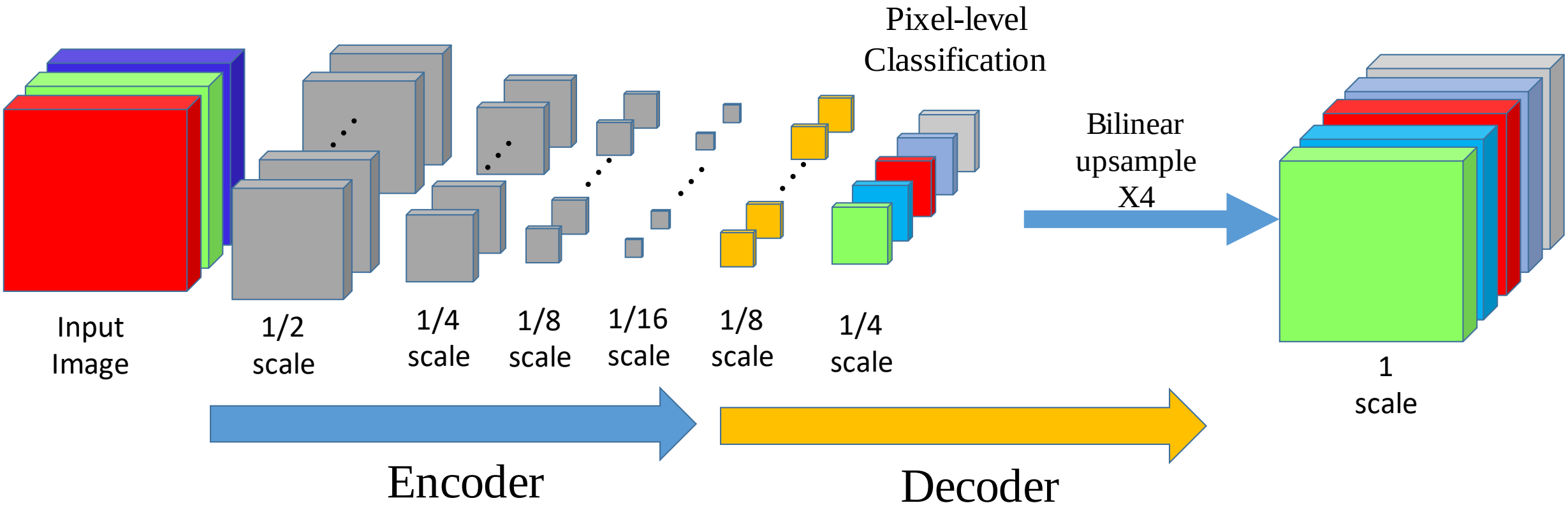
Semantic Segmentation



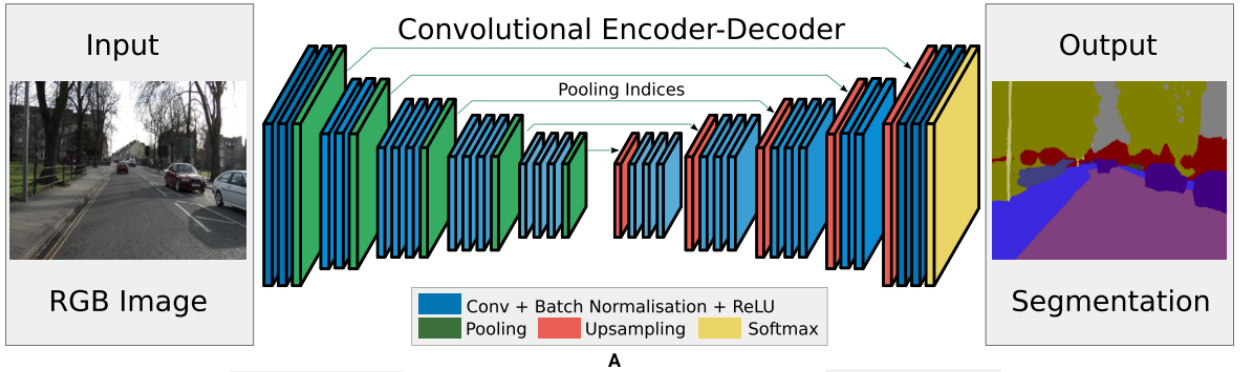
Semantic Segmentation



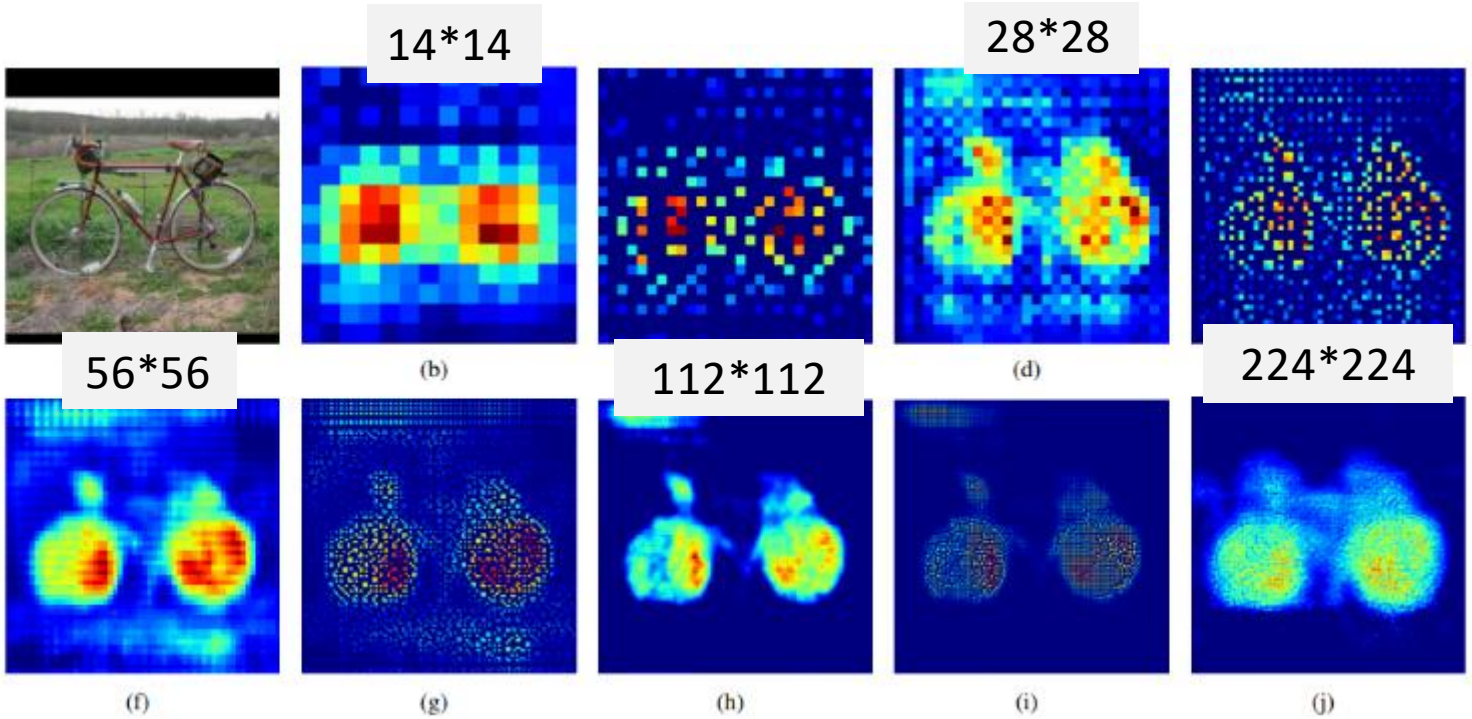
Semantic Segmentation



Semantic Segmentation

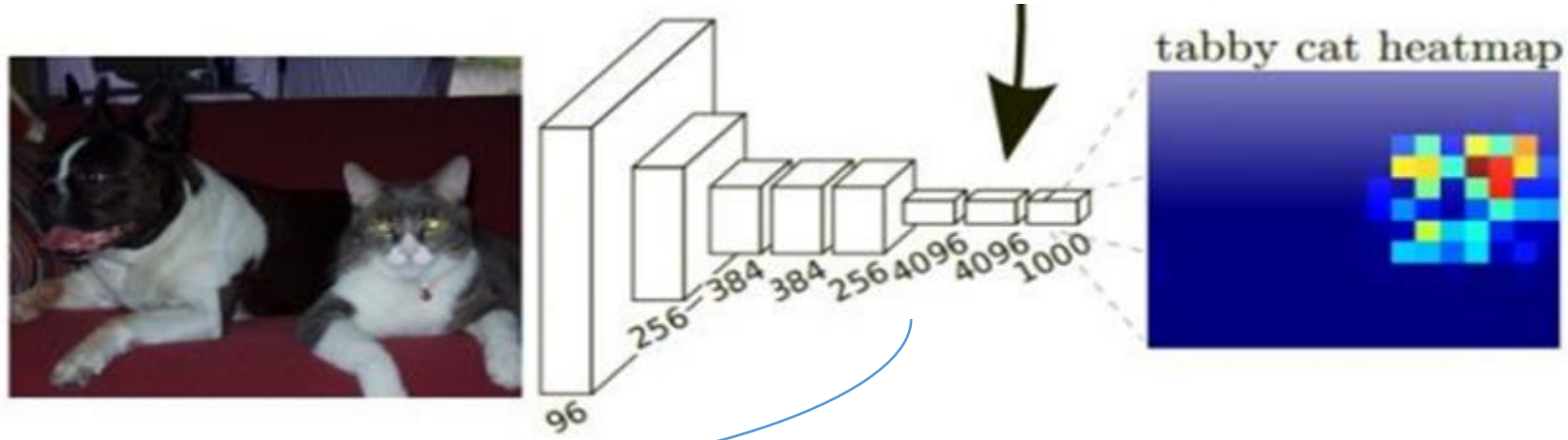


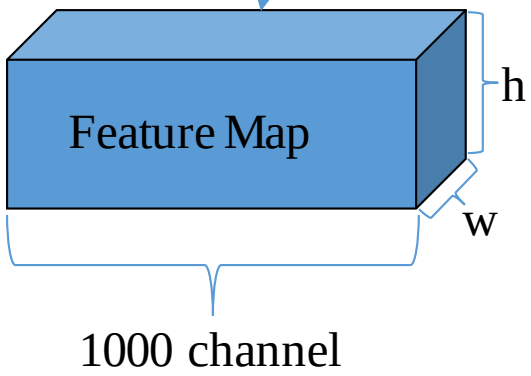
A



Deep Learning: Feature Extractor

- Feature map可以做什麼?





- 這張圖是貓還是夠 (Classification)
- 框出圖片內有貓和狗的位置 (Object detection)
- 從框出的物件(貓和狗)中的描繪出實際物件的輪廓 (Instance Segmentation)
- 把圖片描繪出物件(貓和狗)的輪廓 (Semantic Segmentation)

物件偵測

Backbone NetWork

Feature Map

h

w

channel

GAP

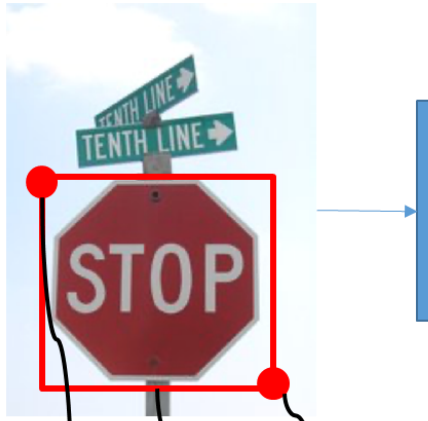
$\hat{x}_{min}$
$\hat{y}_{min}$
$\hat{x}_{max}$
$\hat{y}_{max}$
$\hat{p}_{background}$
$\hat{p}_{object}$

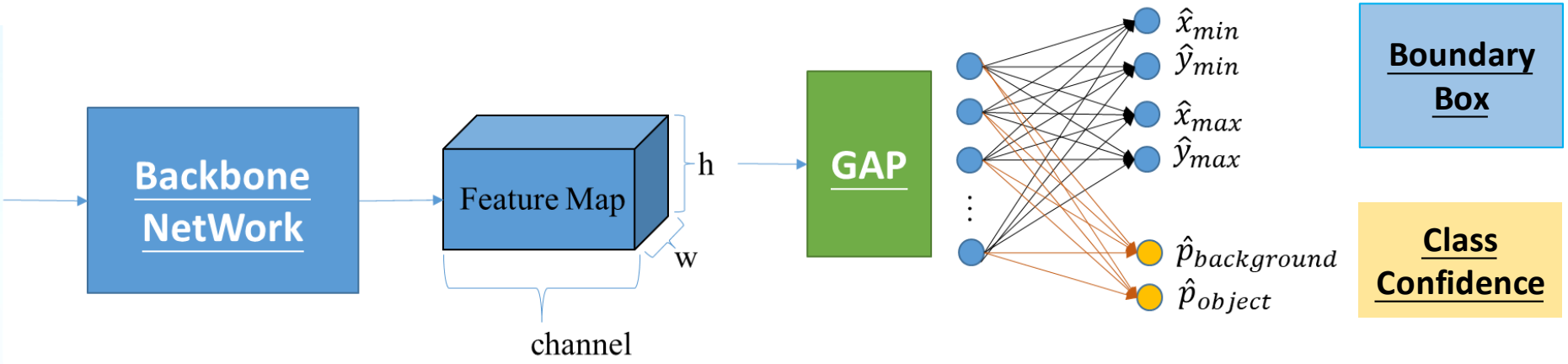
(x_{min}, y_{min})

(x_{max}, y_{max})

物件

物件偵測





目標:

1. $(\hat{x}_{min}, \hat{y}_{min}) \rightarrow (x_{min}, y_{min})$
2. $(\hat{x}_{max}, \hat{y}_{max}) \rightarrow (x_{max}, y_{max})$

→ Regression: MSE

3. $\hat{p}_{background} \rightarrow p_{background}$
4. $\hat{p}_{object} \rightarrow p_{object}$

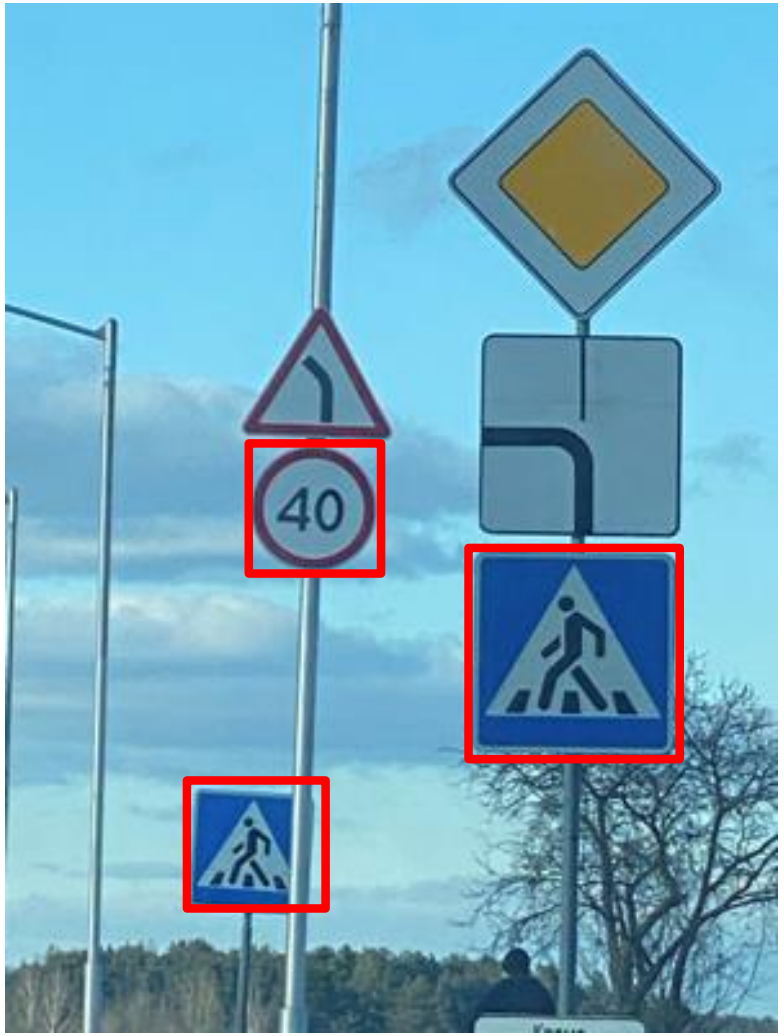
→ Classification: Cross Entropy

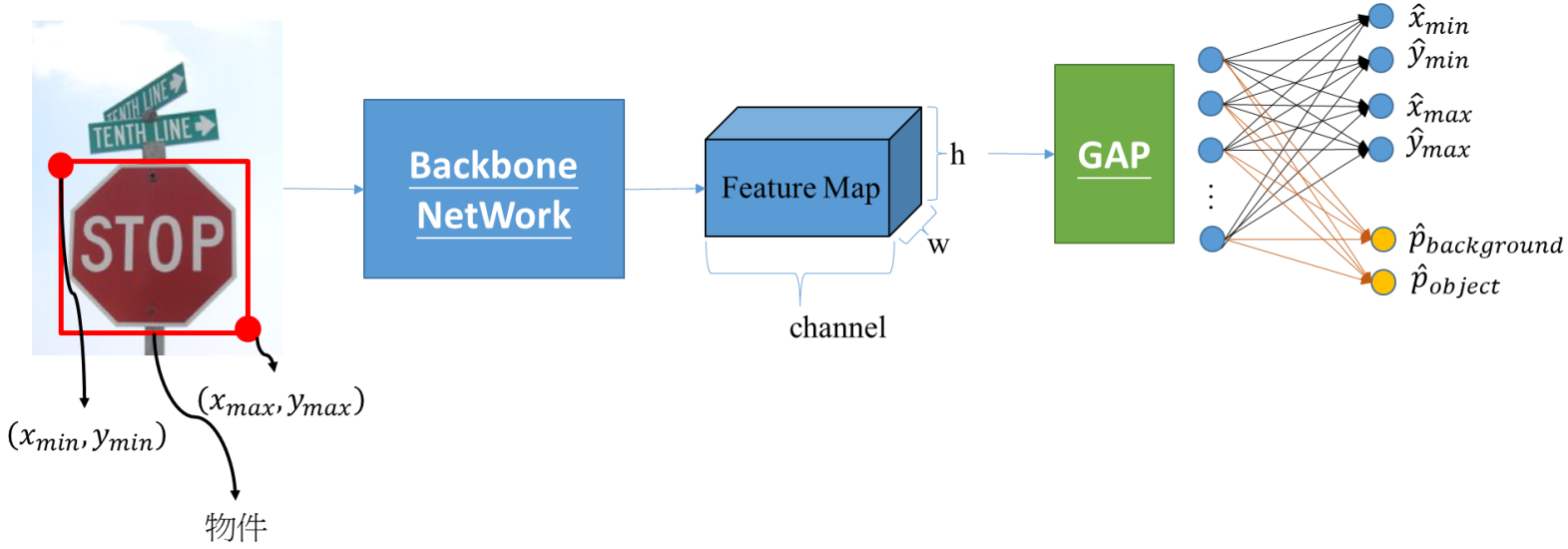
物件: One-hot Encoder

	Background	Object
x	$p_{background} = 0$	$p_{object} = 1$

最小化 {MSE (Regression)+Cross Entropy (Classification)}

上述的架構會有問題





- 物件數超過1 ????

預測框增加到B個

Backbone NetWork

GAP

$\hat{x}_{min}$ $\hat{y}_{min}$ $\hat{x}_{max}$ $\hat{y}_{max}$ Predicted box_1

$\hat{x}_{min}$ $\hat{y}_{min}$ $\hat{x}_{max}$ $\hat{y}_{max}$ Predicted box_B

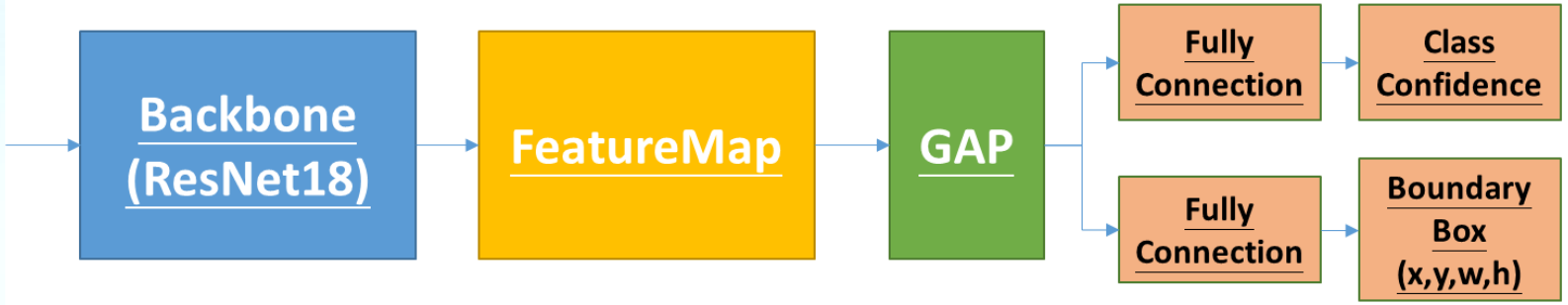
$box_1 = (x_{min}, y_{min}, x_{max}, y_{max})_1$

$box_2 = (x_{min}, y_{min}, x_{max}, y_{max})_2$

$\hat{p}_{background}$ $\hat{p}_{object}$ Predicted box_1

$\hat{p}_{background}$ $\hat{p}_{object}$ Predicted box_B

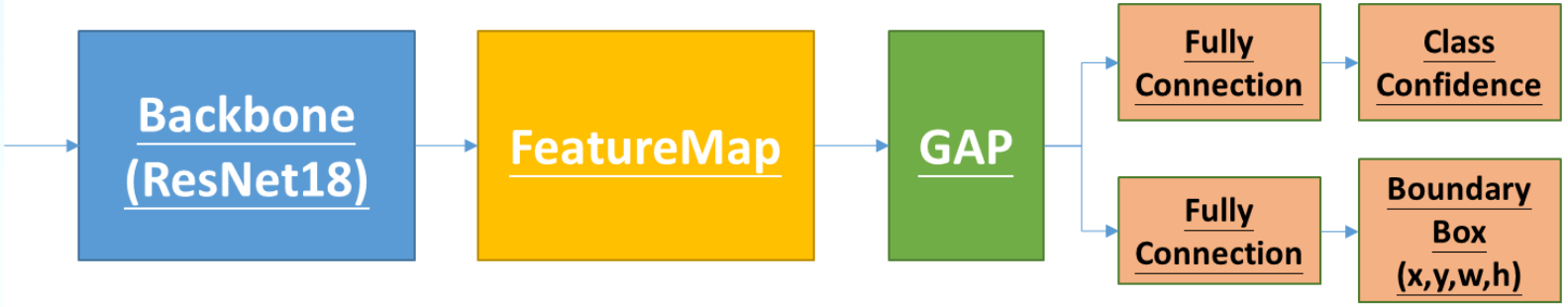
物件偵測

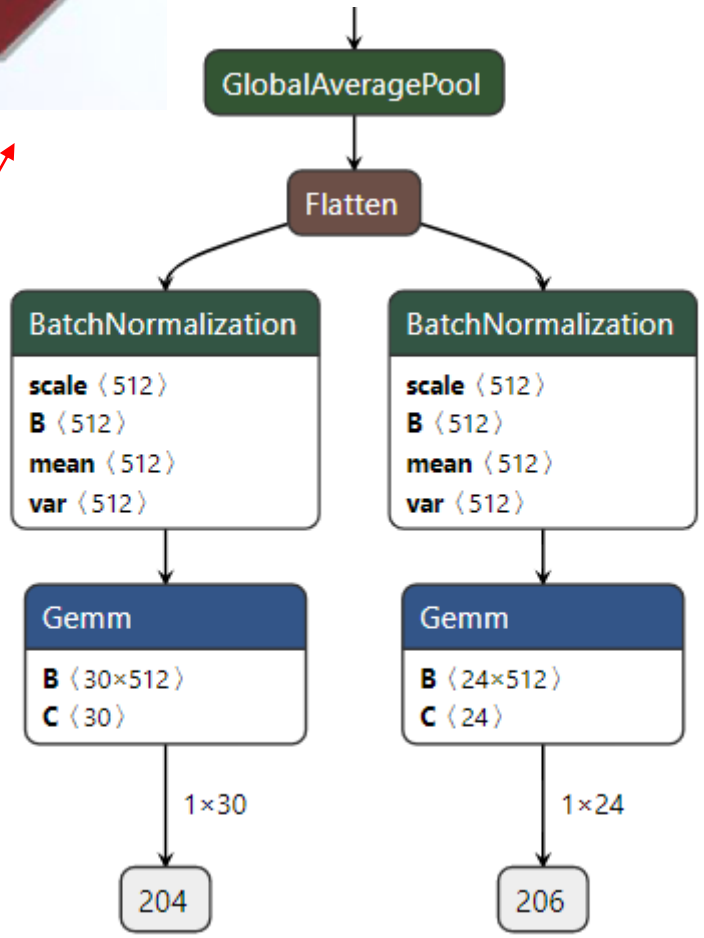


最後輸出預測6個物件框

Boundary Box輸出的特徵圖大小為
batch * (6*4)= batch * 24

Class Confidence輸出的特徵圖大小為
batch * (6*(n_class+1)) = batch * (6*5) = batch * 30



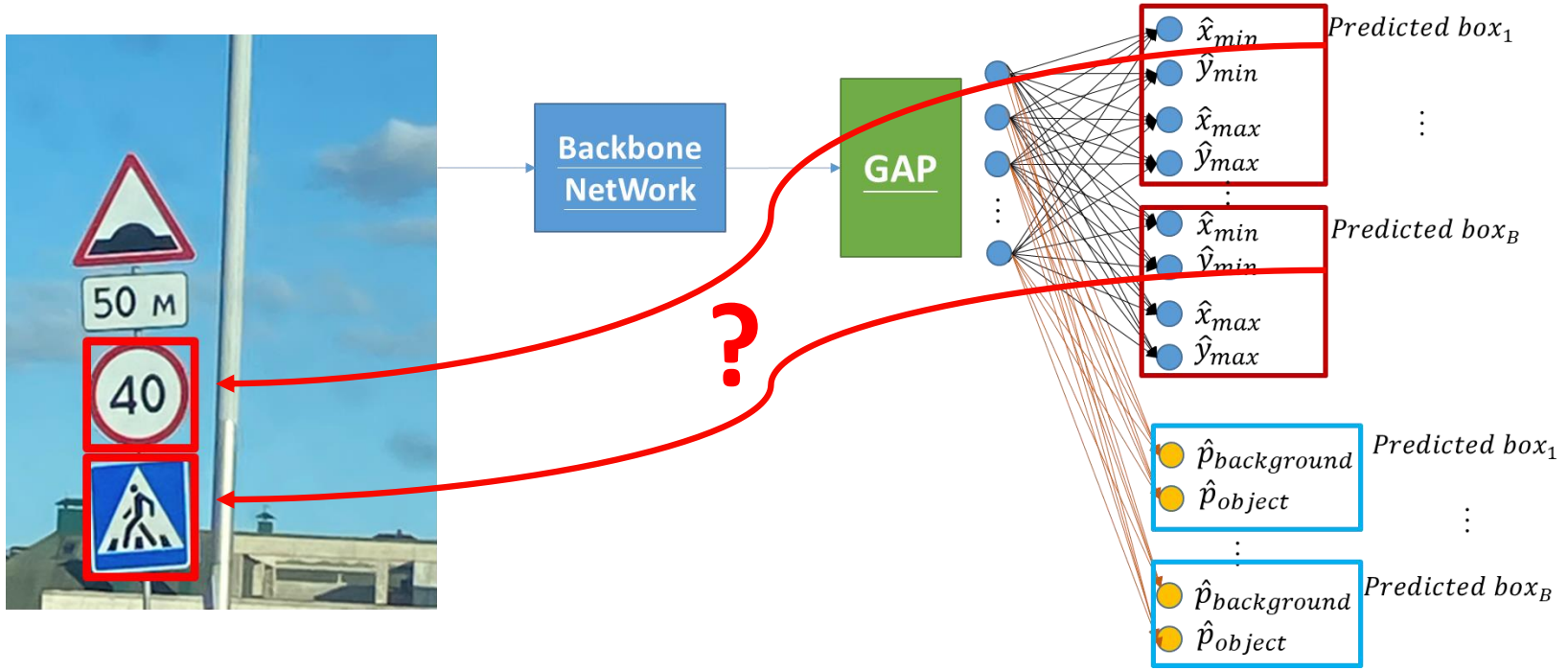


最後輸出預測6個物件框

Boundary Box輸出的特徵圖大小為
batch * (6*4)= batch * 24

Class Confidence輸出的特徵圖大小為
batch * (6*(n_class+1)) = batch * (6*5) = batch * 30

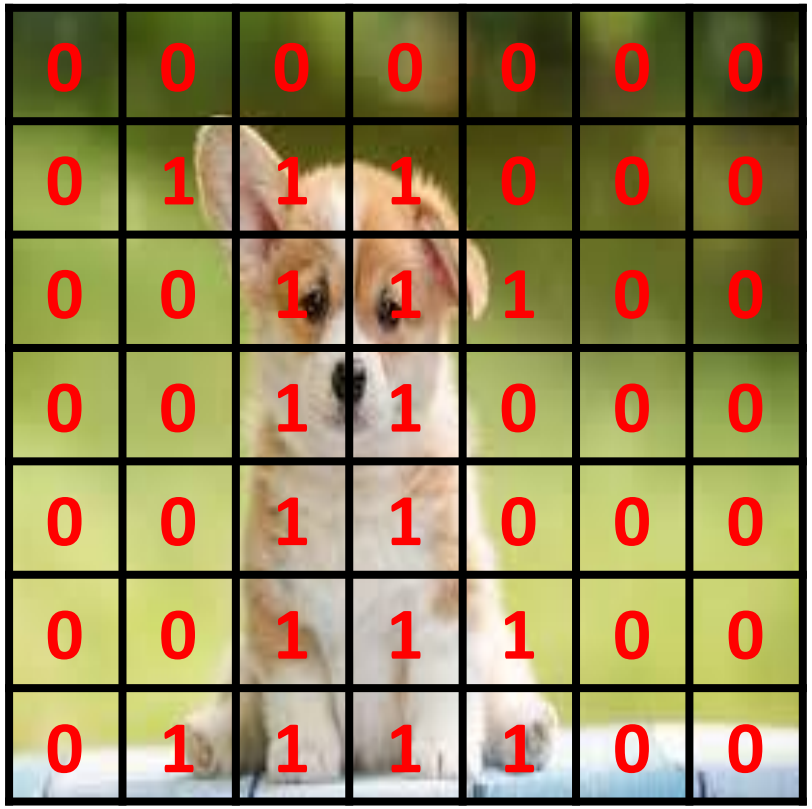
問題



學習的時候，$Predicted\ box_i, i = 1,2,\ldots,B$該預測哪一個物件框。
造成混亂學習，所以會學不起來。

One stage object detection

1^{obj}

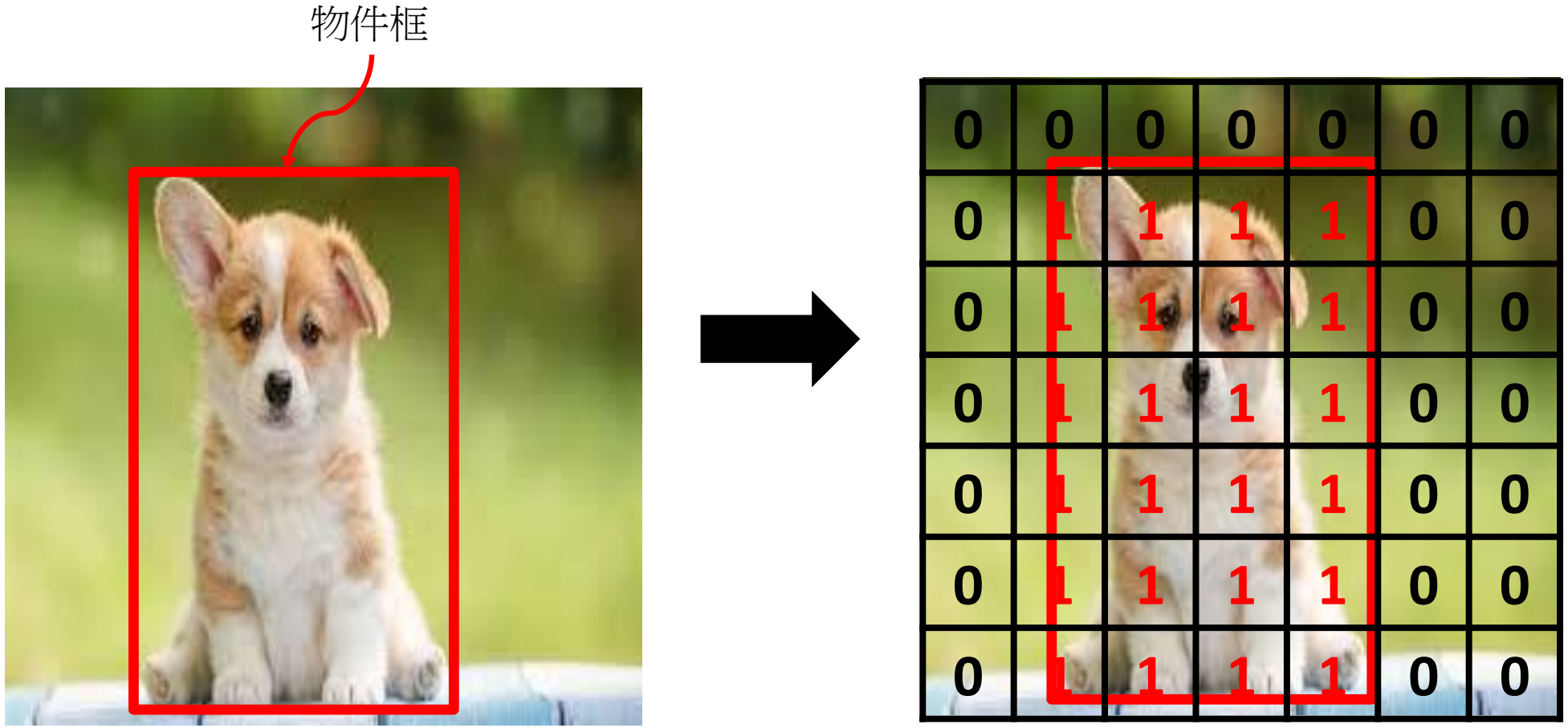


0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	1	1	0	0	0
0	0	1	1	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	0	0

每個grid都去預測物件

預測的時候需要估計物件的
中心位置: X,Y
長寬: H,W
信心度: 是否有物件
類別

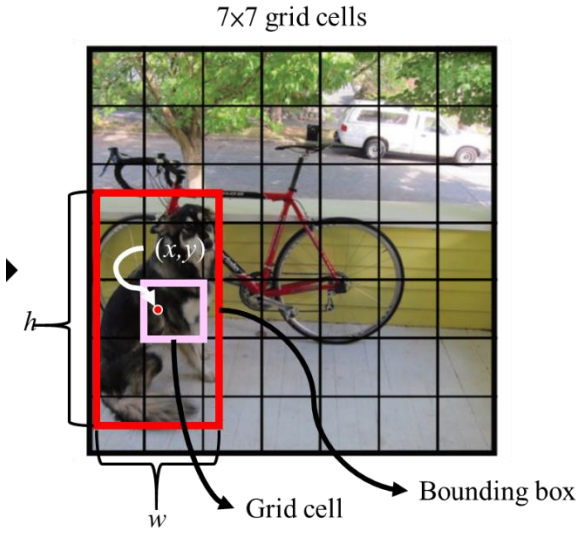
One stage object detection



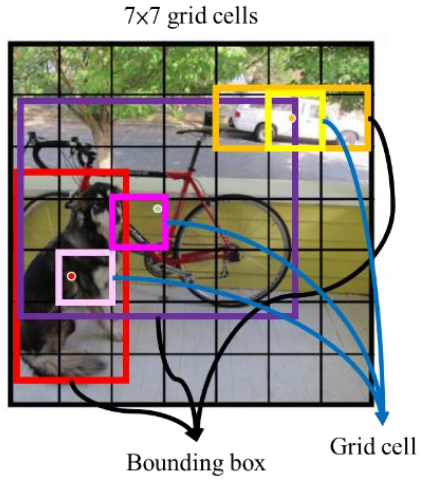
YOLOv1的detector怎麼處理

5: 每個Boundary box (x, y, w, h, confidence)

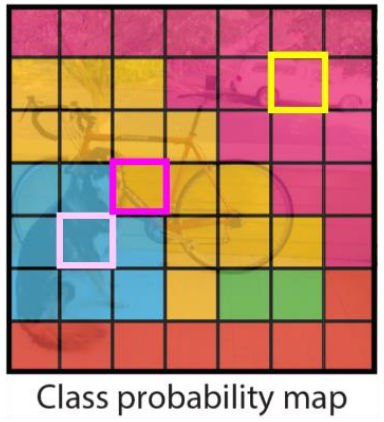
20:屬於20個類別的機率。



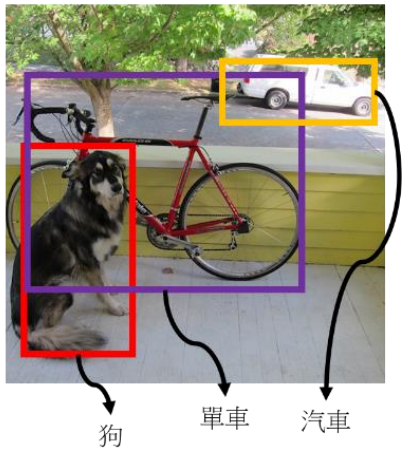
這個物件落在粉紅色框的grid cell
座標為這個grid cell內紅色框這個
Bounding box的中心(x,y)，高寬為h,w。



先利用閾值和NMS選出確定是物
件的**Bounding box**



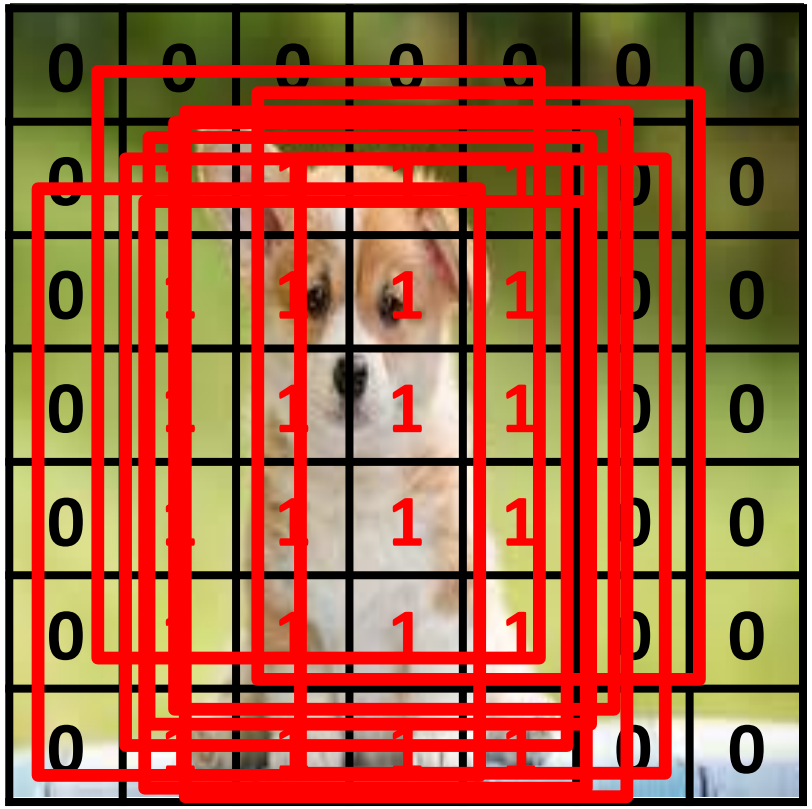
看選出物件所屬的grid cell屬於哪
一類的機率最大。
水藍色:「狗」的機率最大
黃色:「單車」的機率最大
粉紅:「汽車」的機率最大
橘色:「地板」的機率最大



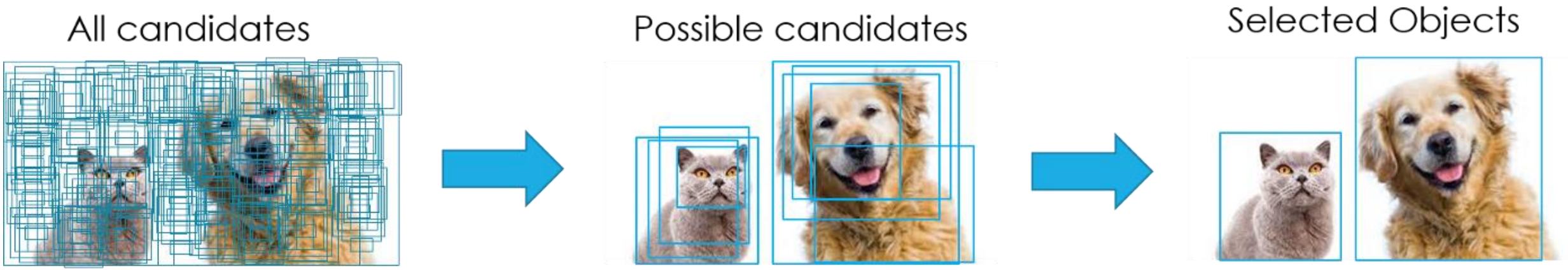
One stage object detection

One stage
object detection

會有一堆候選框



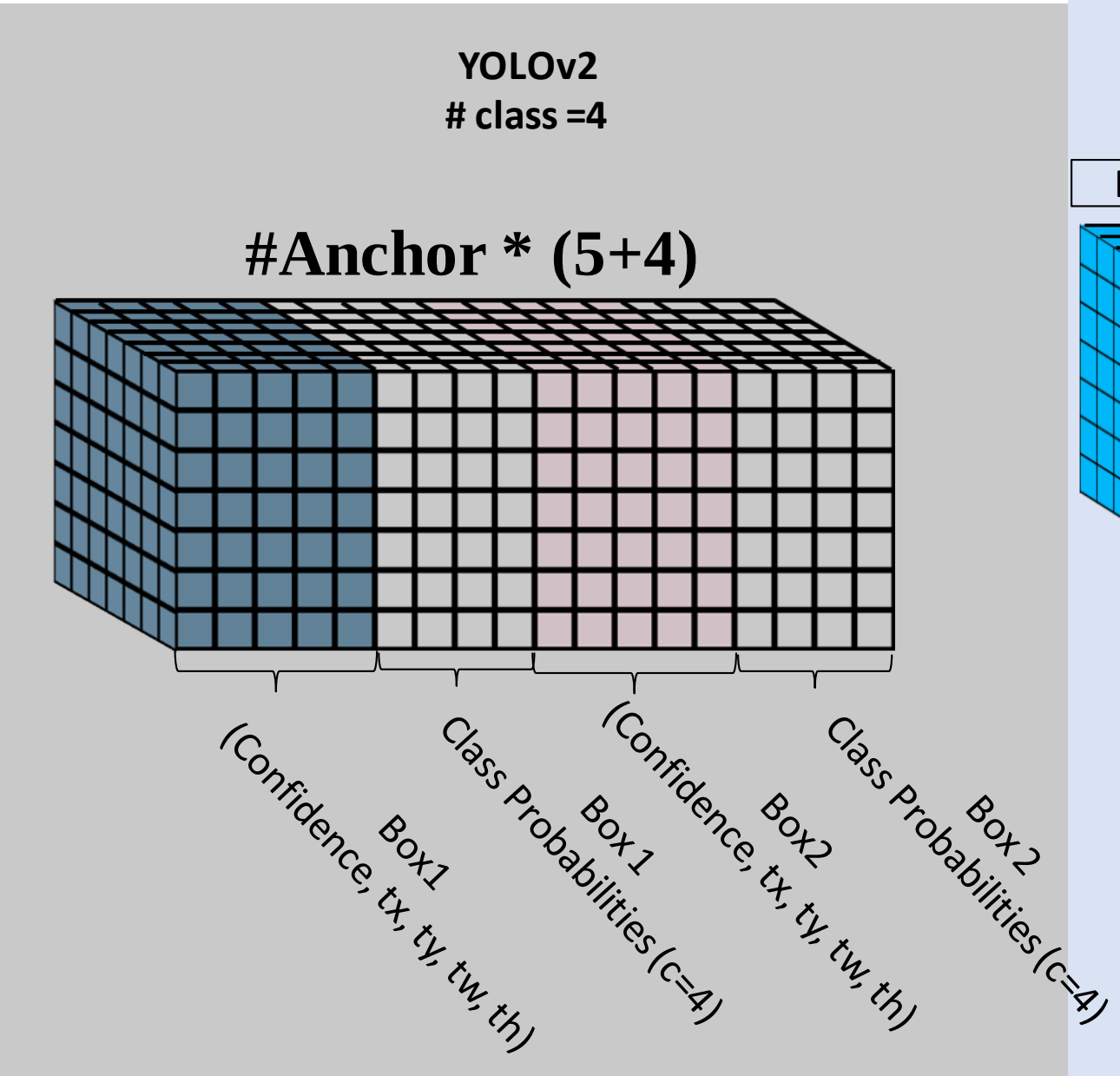
Non-Maximum Suppression (NMS)

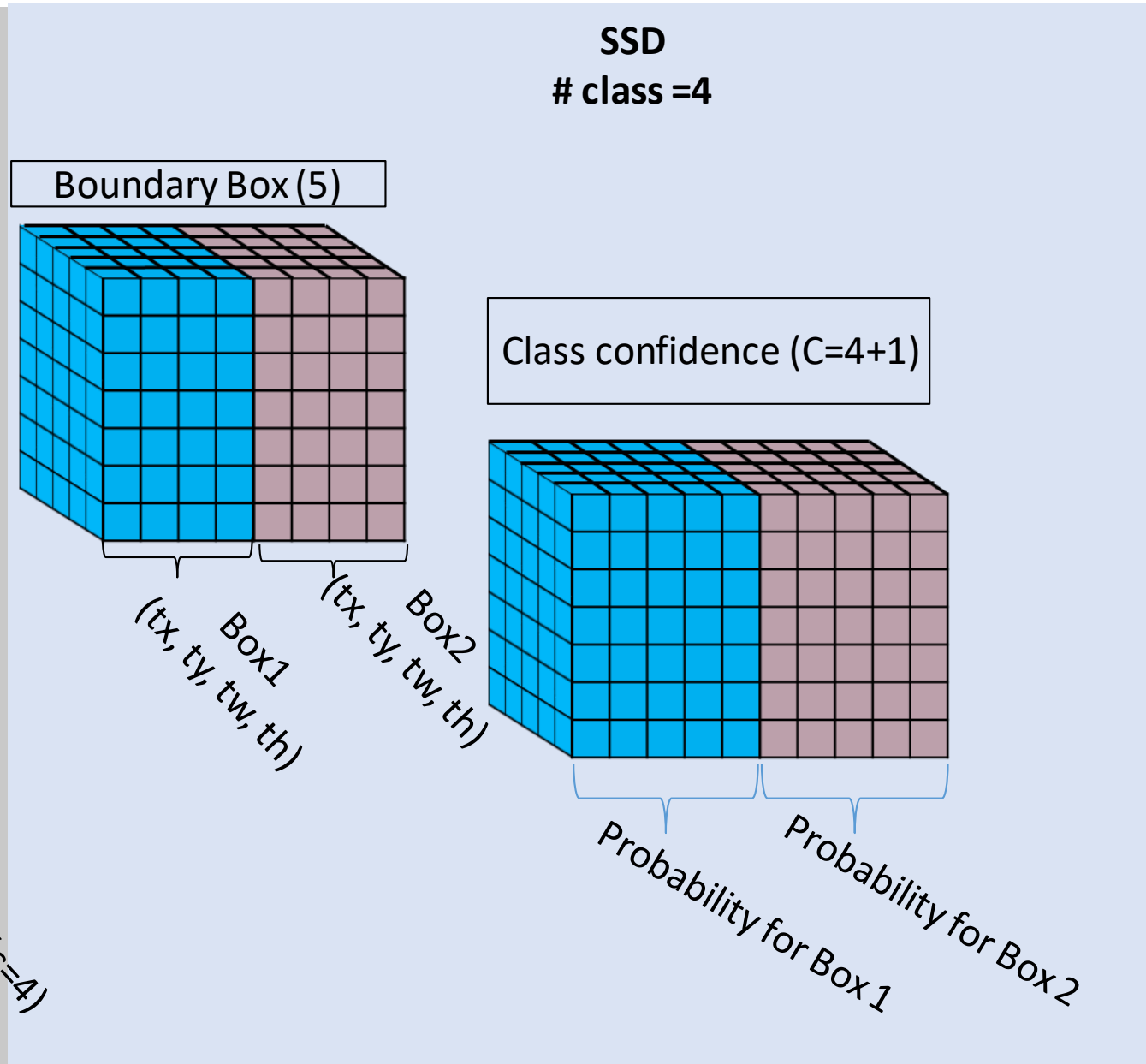


Confidence score > threshold

Non-Maximum Suppression (NMS)

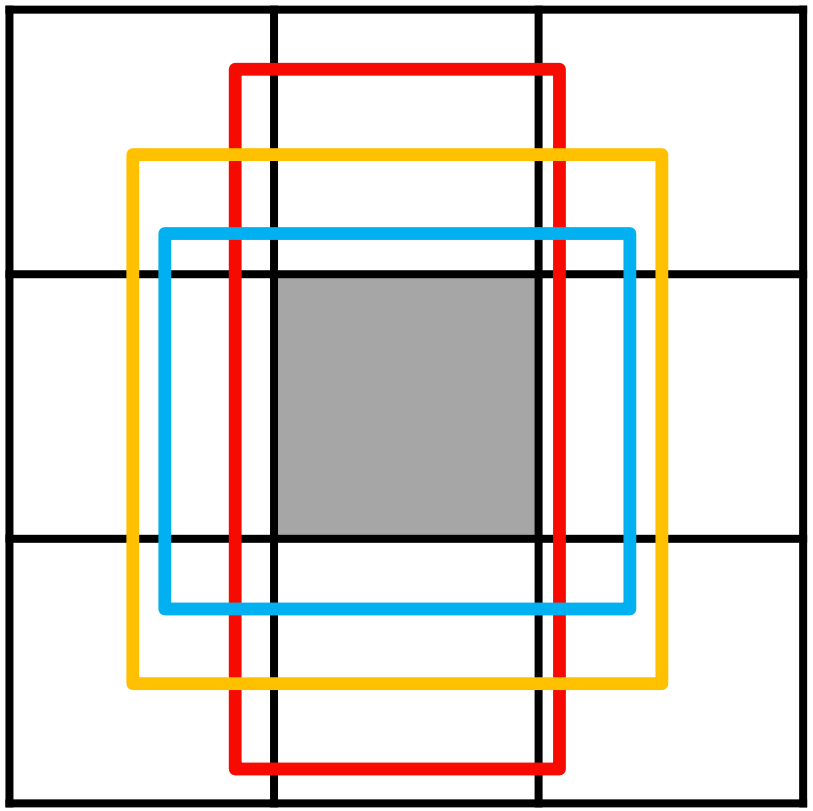
SSD vs YOLO


YOLOv2
class =4
#Anchor * (5+4)
Box1
(Confidence, tx, ty, tw, th)
Class Probabilities (c=4)
Box 1
(Confidence, tx, ty, tw, th)
Box2
Class Probabilities (c=4)
Box 2

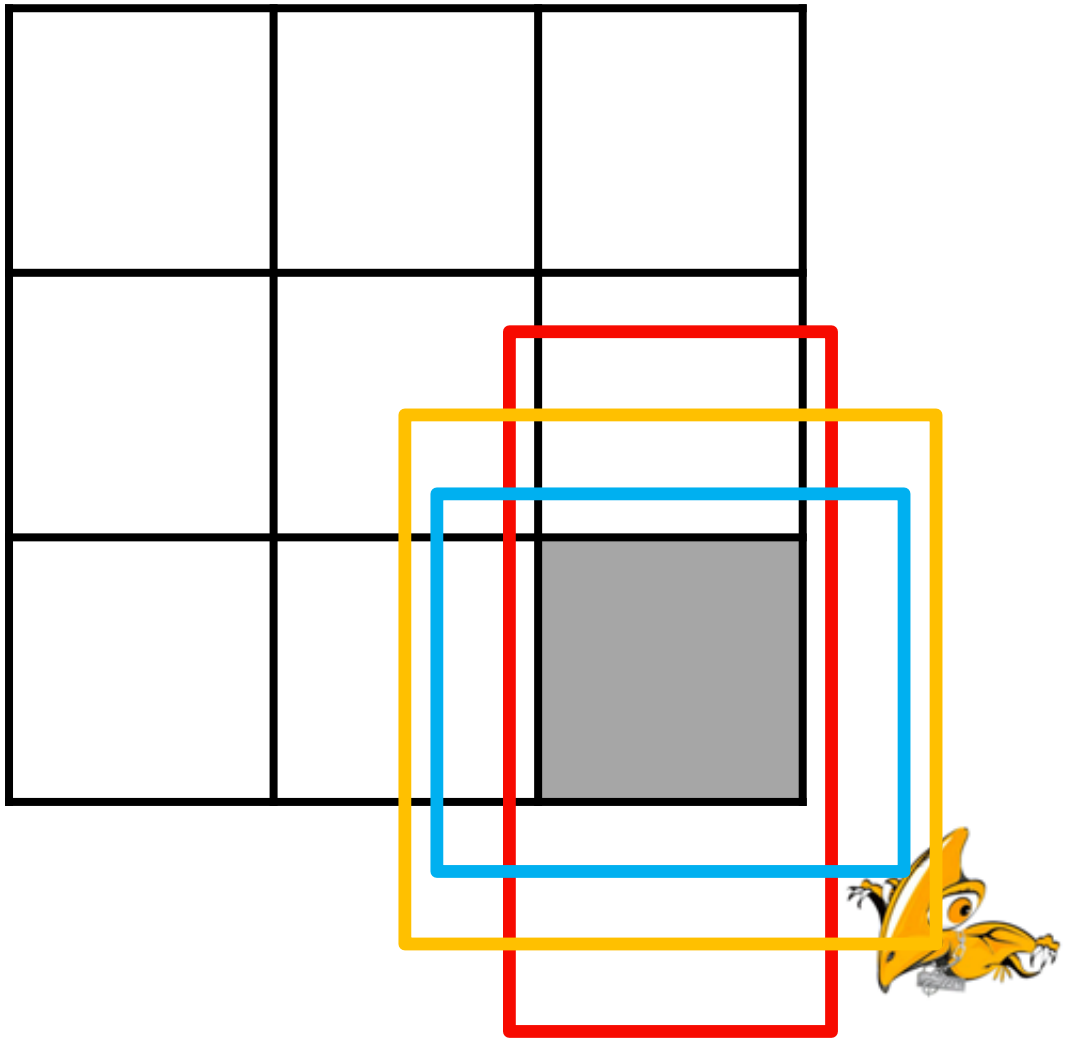

SSD
class =4
Boundary Box (5)
Box1
(tx, ty, tw, th)
Box2
(tx, ty, tw, th)
Class confidence (C=4+1)
Probability for Box 1
Probability for Box 2

YOLOv2後引入Anchor的想法

在最後預測的S*S 的每個grid cell都預設幾個anchors



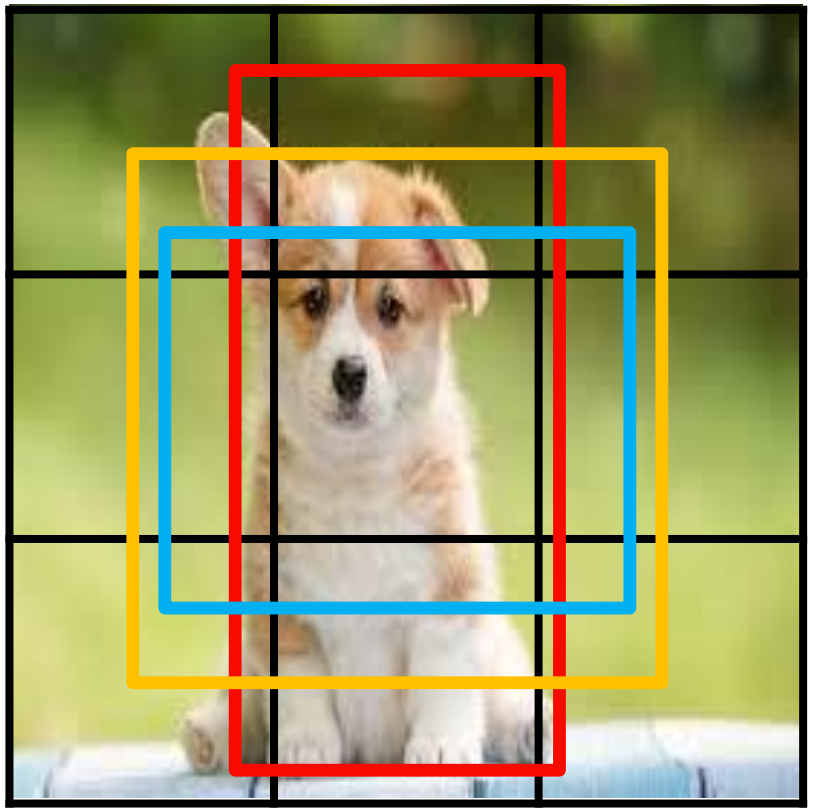
...



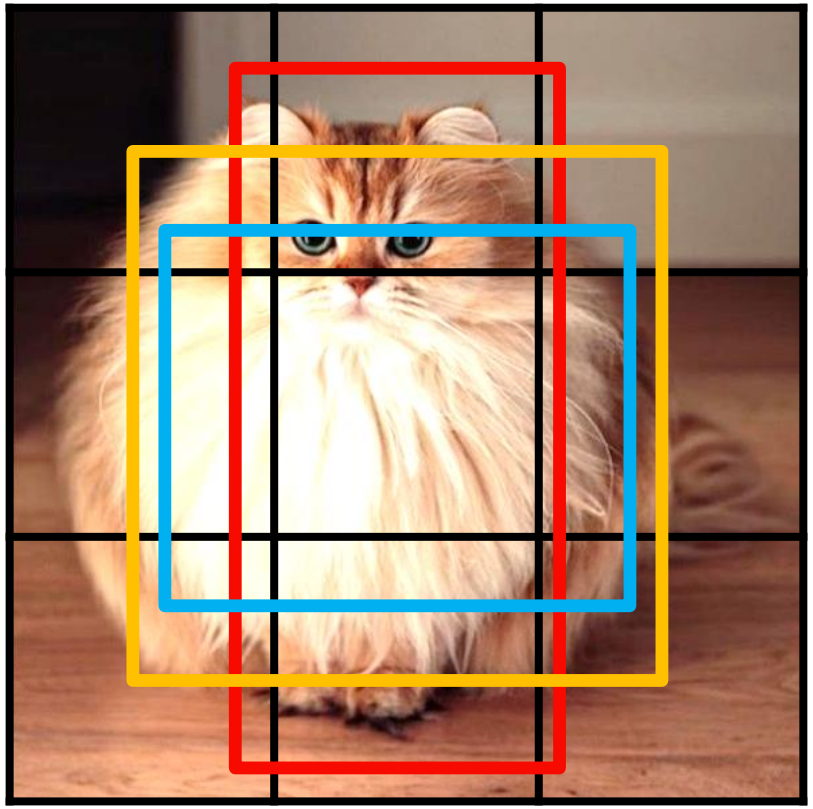
YOLOv2後引入Anchor的想法

不同的物件大小用不同anchors去Fit，所以我們在預測的時候只要去調整Anchor的大小來Fit物件就好。

紅色的Anchor比較適合狗



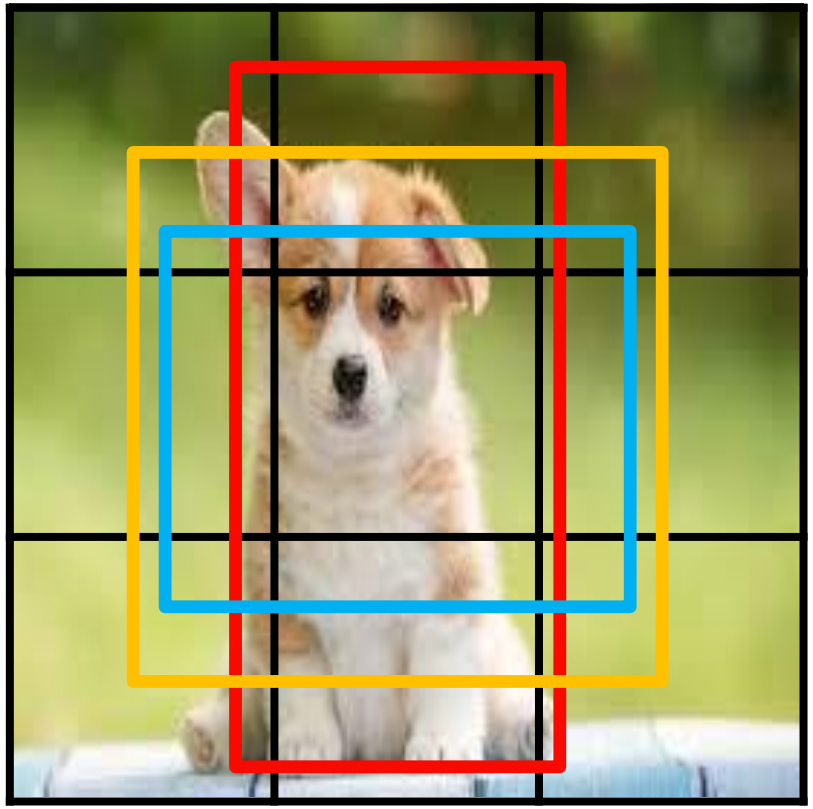
黃色的Anchor比較適合貓



YOLOv2後引入Anchor的想法

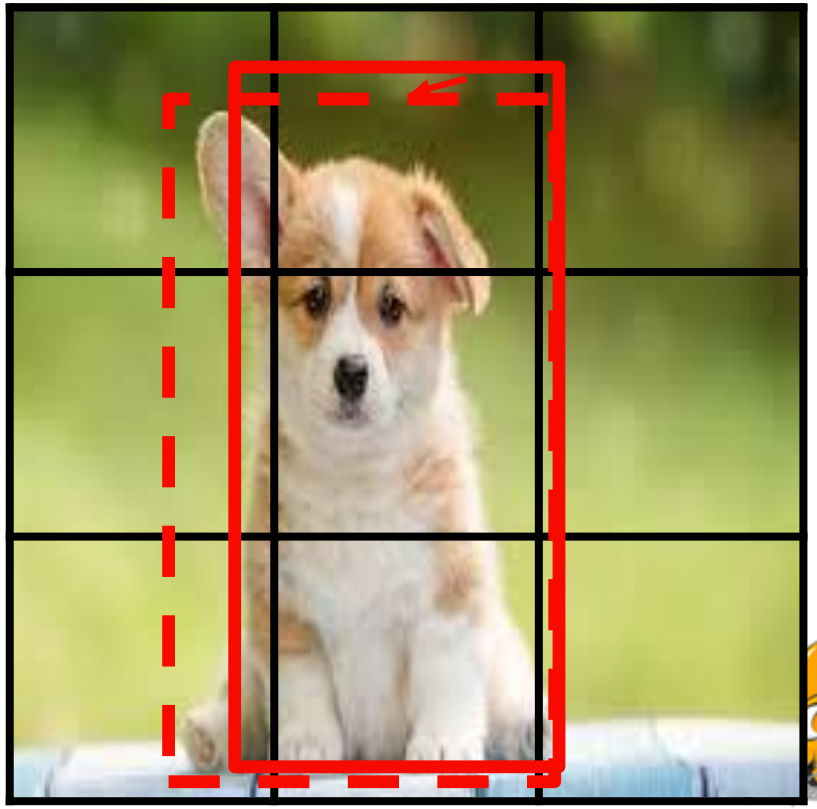
不同的物件大小用不同anchors去Fit，所以我們在預測的時候只要去調整Anchor的大小來Fit物件就好。

紅色的Anchor比較適合狗



神經網路學習
調整紅色的Anchor去fit狗

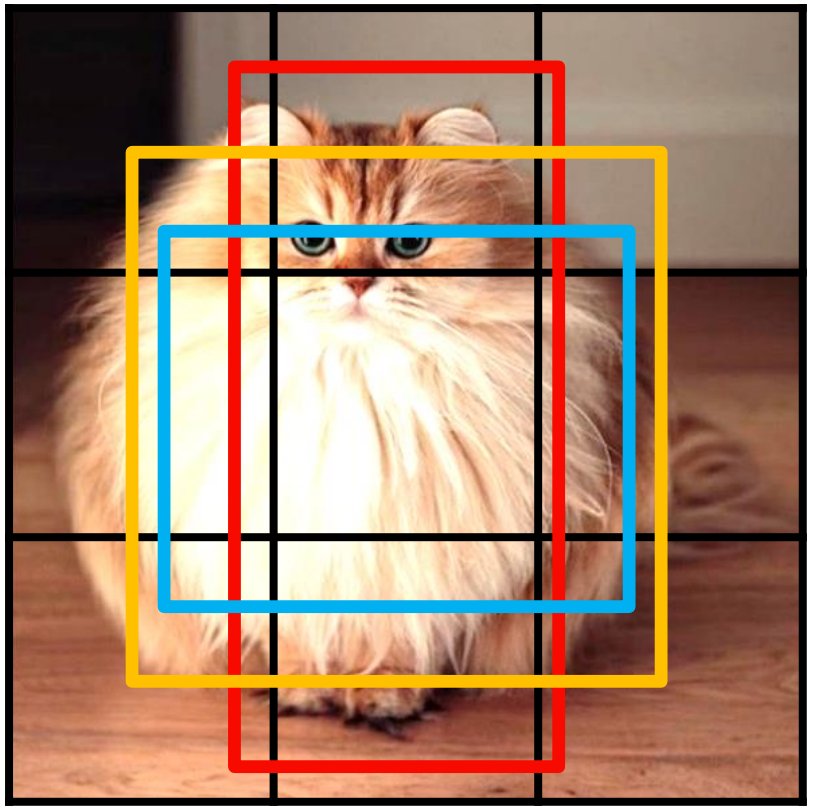
紅色anchor高縮小一點，往左下移動一點



YOLOv2後引入Anchor的想法

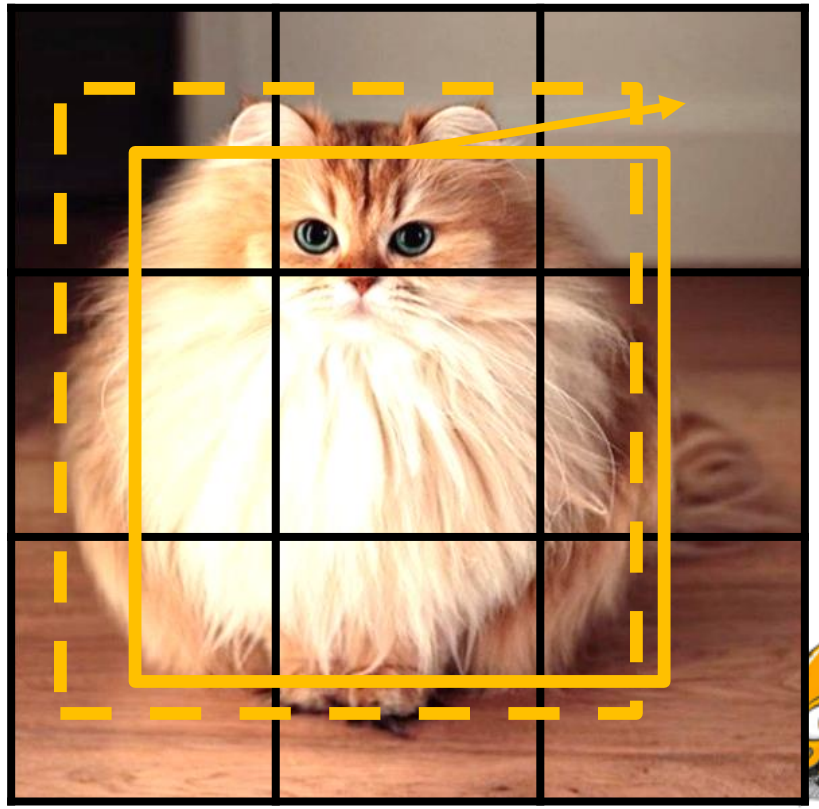
不同的物件大小用不同anchors去Fit，所以我們在預測的時候只要去調整Anchor的大小來Fit物件就好。

黃色的Anchor比較適合貓



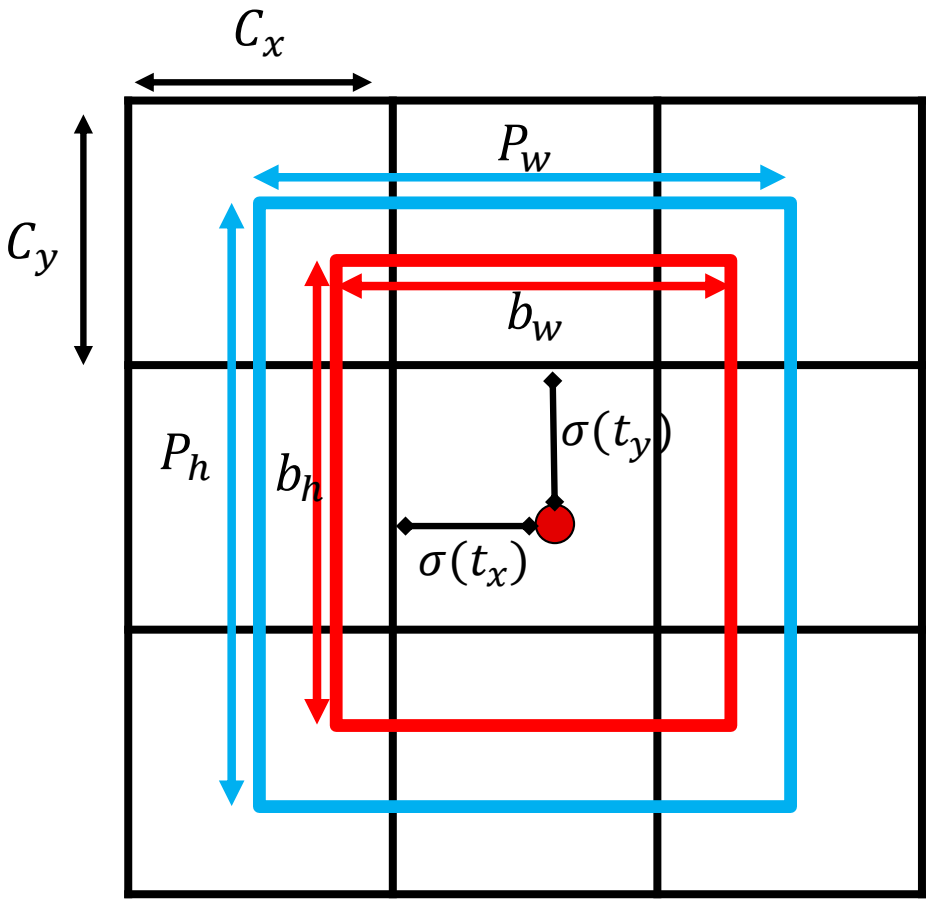
神經網路學習
調整黃色的Anchor去fit貓

黃色anchor長寬放大一點，中心稍移動



YOLO如何推算物件座標和大小

- 學習調整 Anchor (放大縮小和中心位移)
- **YOLOv2↑: (Confidence, tx, ty, tw, th)**



YOLO模型輸出(t_x, t_y, t_w, t_h)

模型輸出要轉換成為實際的物件中心座標(b_x, b_y)和寬高(b_w, b_h)

中心座標轉換:

$$b_x = \sigma(t_x) + C_x$$
$$b_y = \sigma(t_y) + C_y$$

寬高轉換由Anchor去進行縮放:

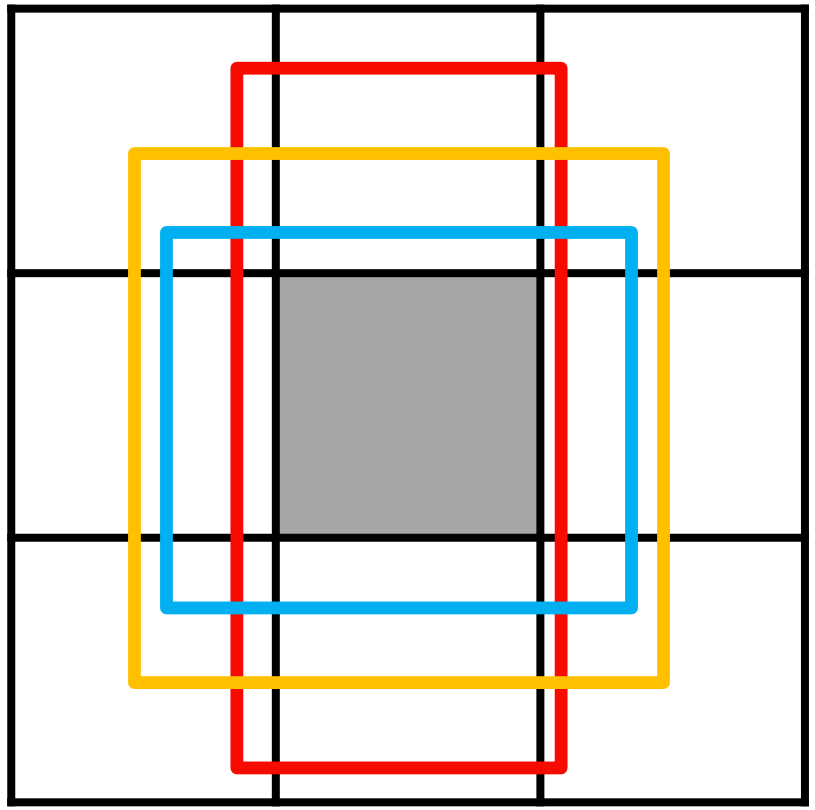
$$b_w = P_w e^{t_w}$$
$$b_h = P_h e^{t_h}$$

(P_w, P_h): 我們設定的Anchor寬高。

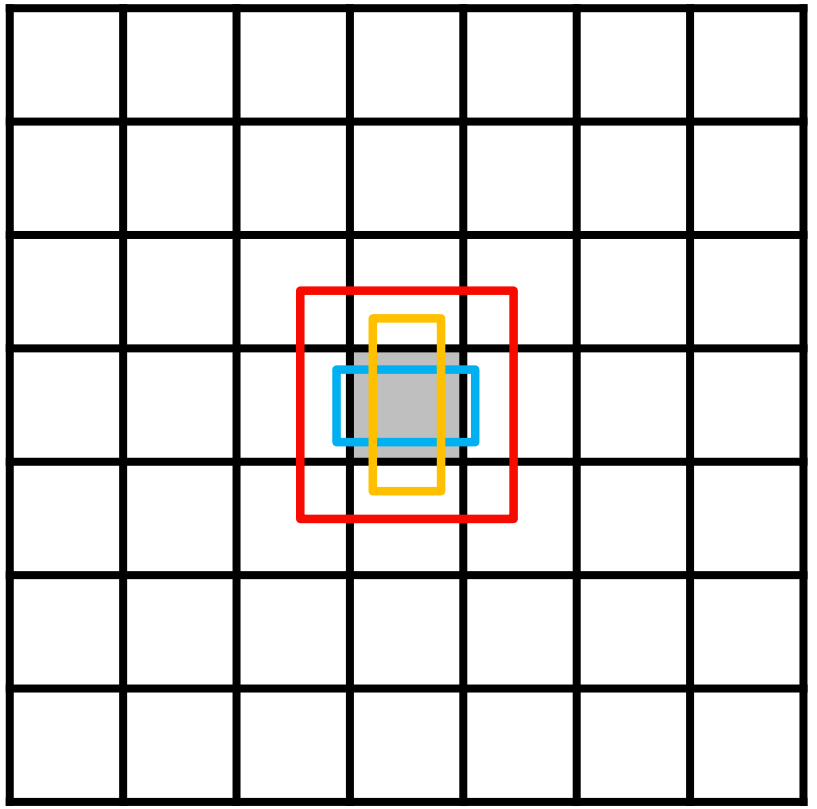
YOLOv3後引入Multi-scale Detection

在最後預測的S*S 的每個grid cell都預設幾個Anchors。

3*3的特徵圖



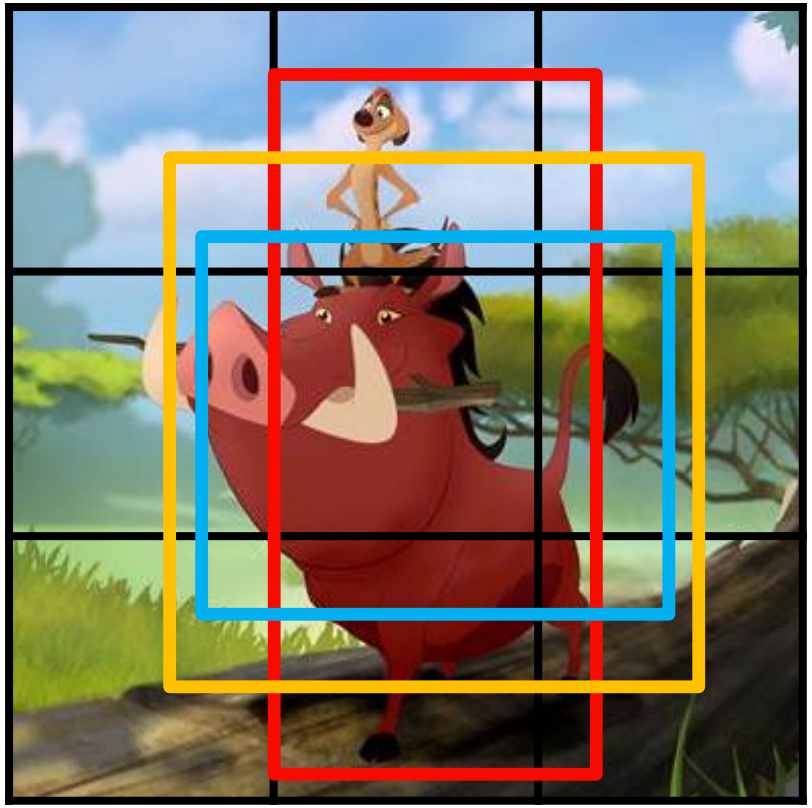
7*7的特徵圖



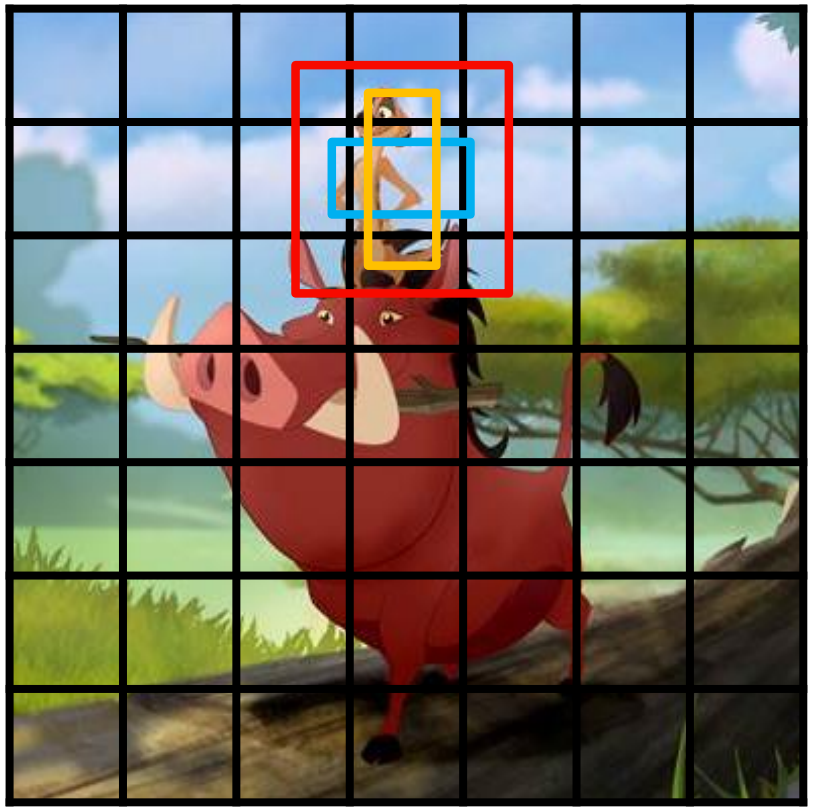
YOLOv3後引入Multi-scale Detection

在最後預測的S*S 的每個grid cell都預設幾個Anchors。

3*3的特徵圖



7*7的特徵圖



在小解析的特徵圖(3*3)適合最大物件偵測。

在大解析的特徵圖(7*7)適合最大物件偵測。

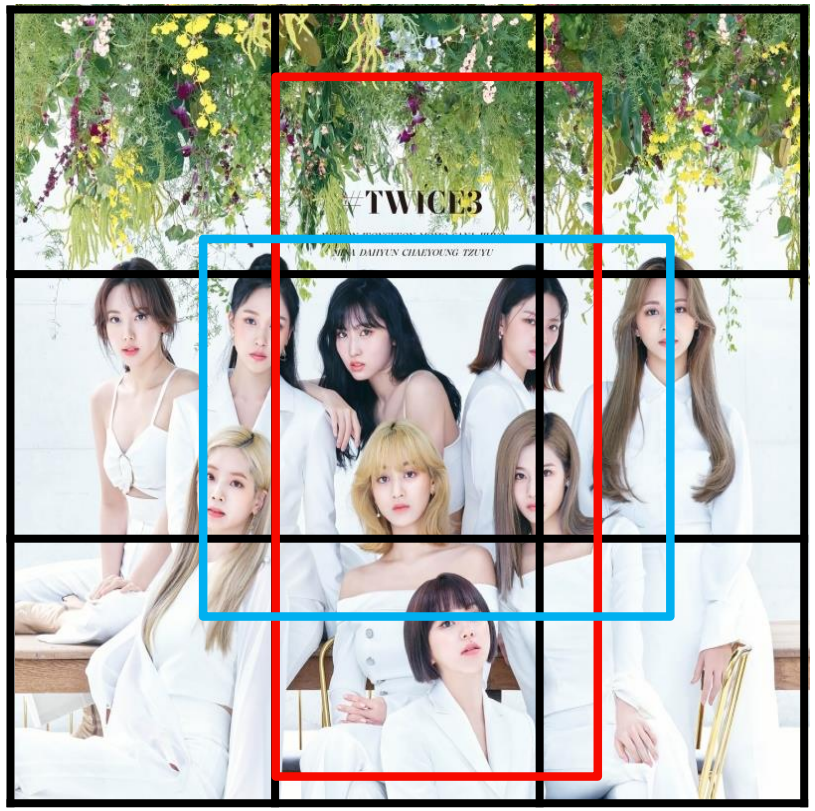
3*3特徵圖藍色的Anchor只需要微調就能fit彭彭

7*7特徵圖黃色的Anchor只需要微調就能fit丁滿

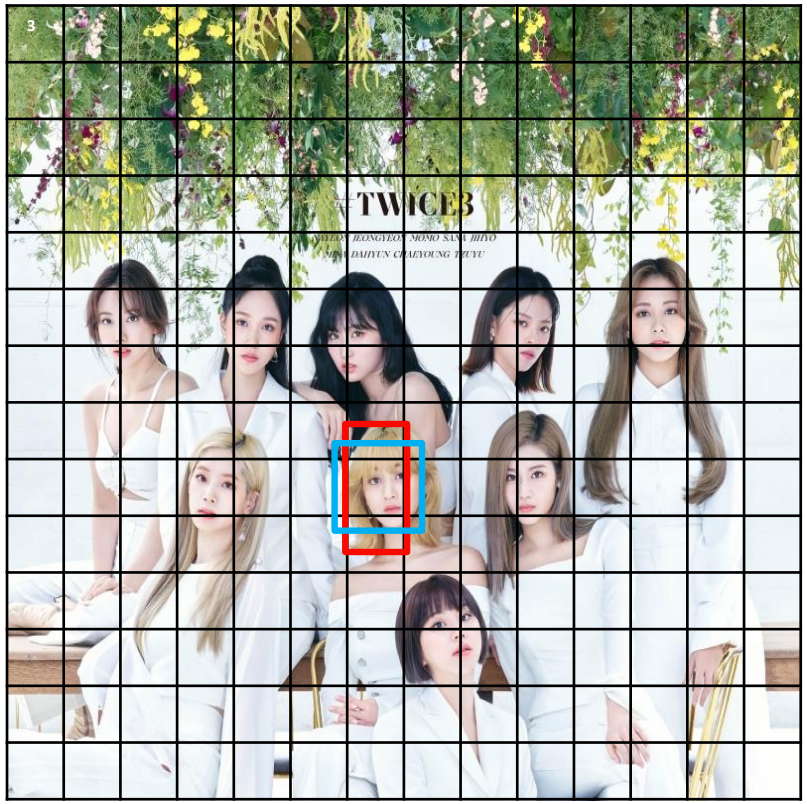
YOLOv3後引入Multi-scale Detection

人臉偵測

3*3的特徵圖



14*14的特徵圖



在3*3特徵圖設定的 Anchor 要縮小10倍，這樣的學習太難了

Model Inference Example

https://github.com/TommyHuang821/NTUT_EdgeAICourse/blob/main/main_onnx_segmentation_voc.ipynb

https://github.com/TommyHuang821/NTUT_EdgeAICourse/blob/main/main_pytorch_objectdetection_onnx.ipynb

https://github.com/TommyHuang821/NTUT_EdgeAICourse/blob/main/main_pytorch_imageclassification_onnx.ipynb