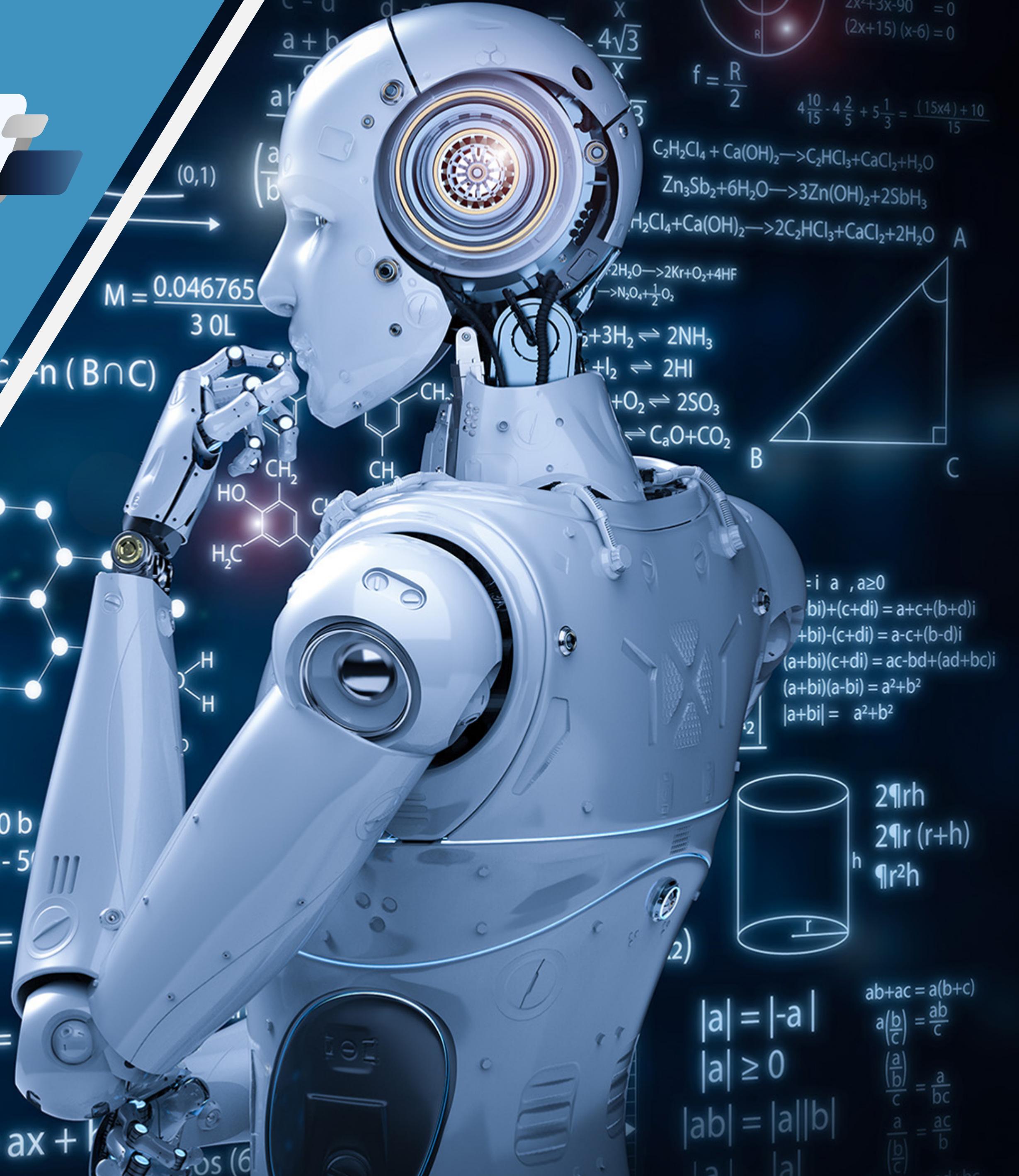


Day 32

深度學習與電腦視覺 學習馬拉松

cupay 陪跑專家：杜靖愷



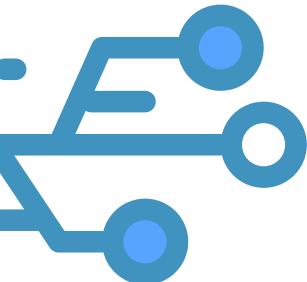


YOLO 簡介及算法理解

重要知識點



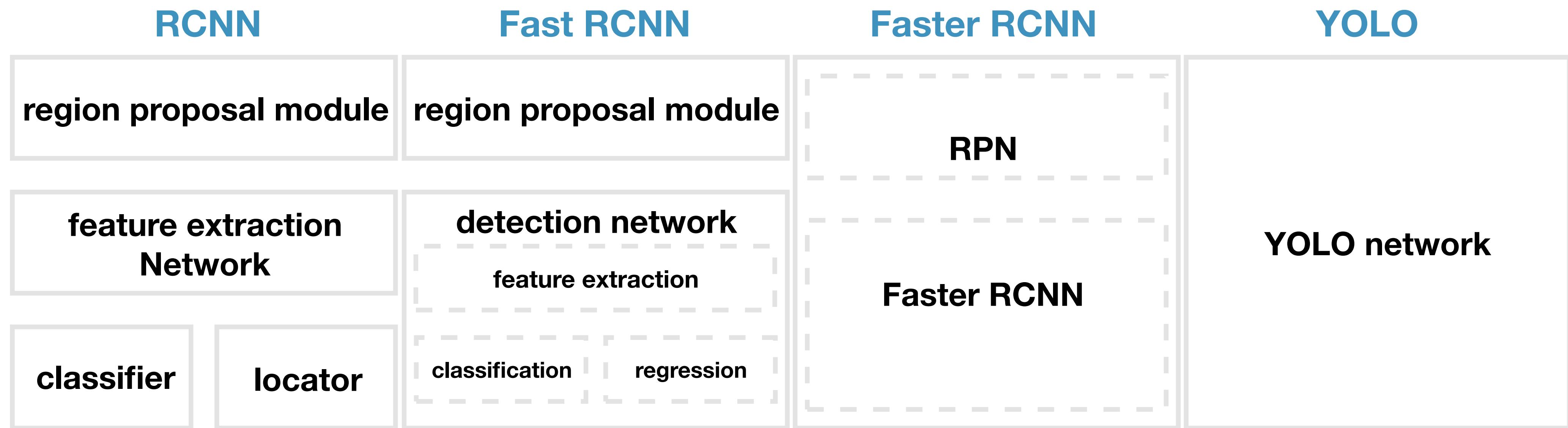
- 了解 YOLO 基本框架。
- YOLO 如何做到直接預測 bounding box 及其類別。



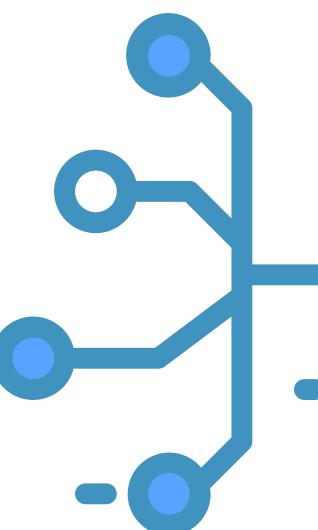
YOLO 簡介

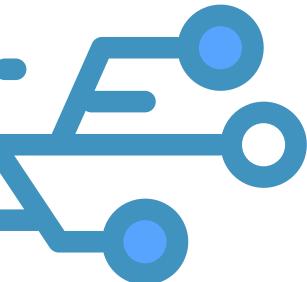


前面的課程有提過，YOLO 是 2016 年繼 RCNN、fast-RCNN 和 faster-RCNN 之後，Ross Girshick 以及 Joseph Redmon 針對當時偵測速度慢的問題提出的另一種 object detection 框架，目前已經從 YOLOv1 演進到 YOLOv3。



YOLO，是 You Only Look Once 的縮寫，意思是只需要看一眼，也是 YOLO 的核心思想。對比其他框架，YOLO 以整張圖作為網絡的輸入，直接在網路輸出層迴歸 (regression) 出 bounding box 的位置和該 bounding box 的物件類別，而這樣的設計使得 YOLO 在 Titan X 的 GPU 上能達到 45 FPS。





課程安排



CUPOY

YOLO 目前已經演進到第三個版本，但是單看 YOLOv3 的論文是沒有辦法很好地理解到個中精髓，如果要對 YOLO 有深度的掌握，建議還是要從 YOLOv1 開始了解。這部分的課程主要拆解成 3 個部分：

1. YOLO 算法理解

從概念上地理解 YOLO 整體架構的設計

2. YOLO 細節理解

詳細拆解 YOLO 的細節，搭配程式碼作業輔助理解

3. YOLOv3 keras 實現

網路上有許多 YOLOv3 的實現，本課程選了其中一個簡單易懂的 keras 版本介紹給大家

可依照個人需求來調整學習的順序，喜歡動手實作的朋友可以從 3 開始；喜歡理論的朋友可以先看 1 和 2

YOLO 算法理解

YOLOv3 keras 實現

圖片及影片物件偵測

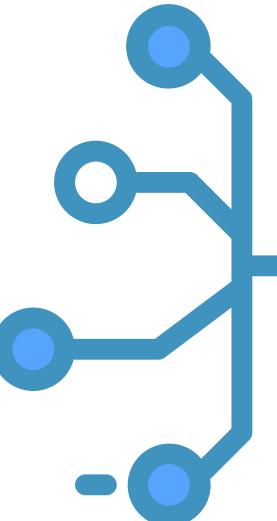
訓練模型

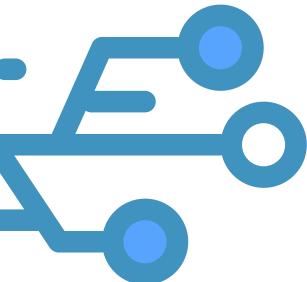
YOLO 細節理解

網路輸出及後處理

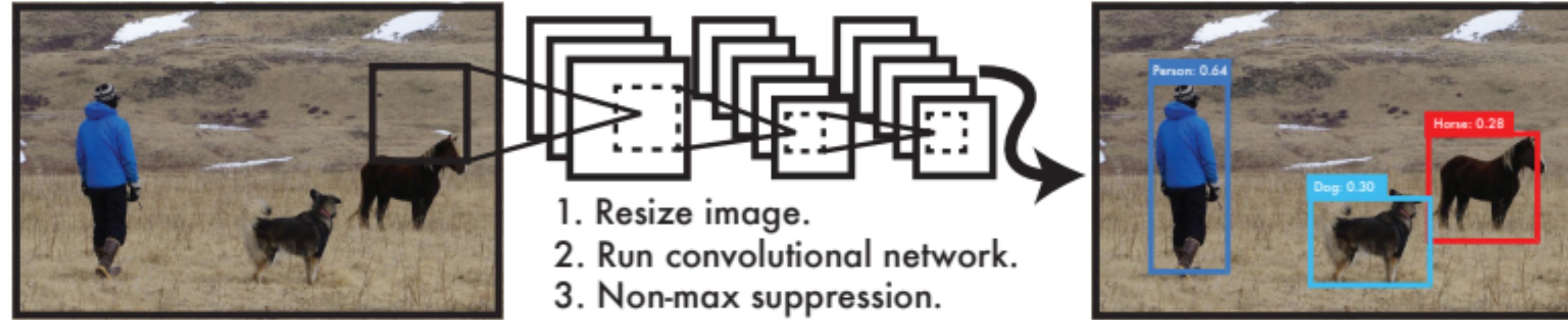
Loss function 理解

網路結構





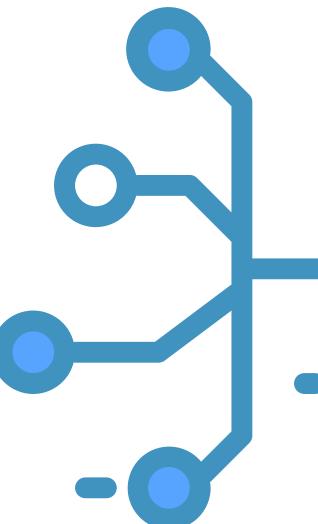
YOLO 算法理解



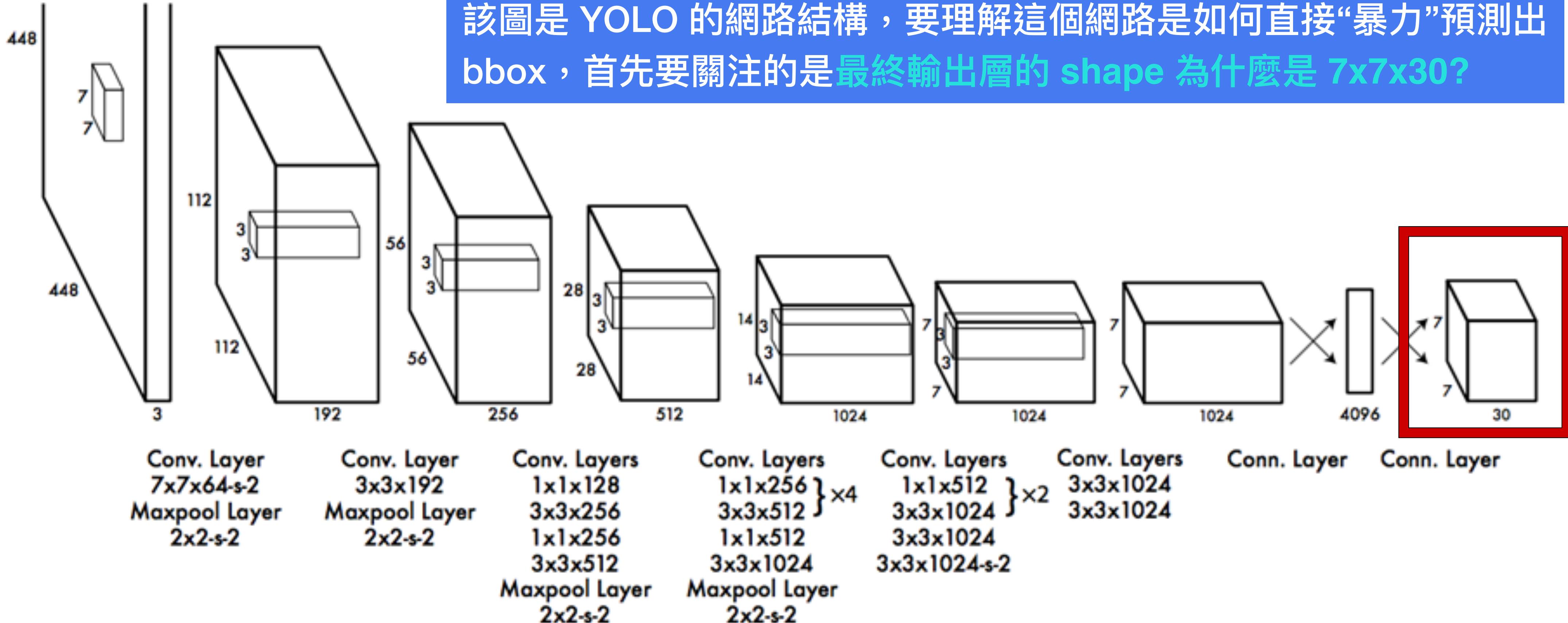
“We frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities.”

YOLO 把 object detection 定義為一個迴歸問題，直接在從圖像來預測出 bounding box (bbox) 的坐標和該 box 是否包含物件的信心度及物件的 class probabilities，偵測的流程大致上分為三步

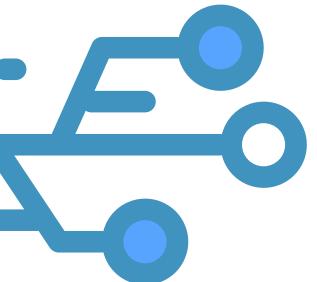
1. 將影像 resize 到 448x448，輸入網路
2. 執行 CNN，提取圖像的特徵，在輸出層“暴力”預測出 bbox 坐標和物件類別
3. 對輸出層的 bbox 信息做 NMS



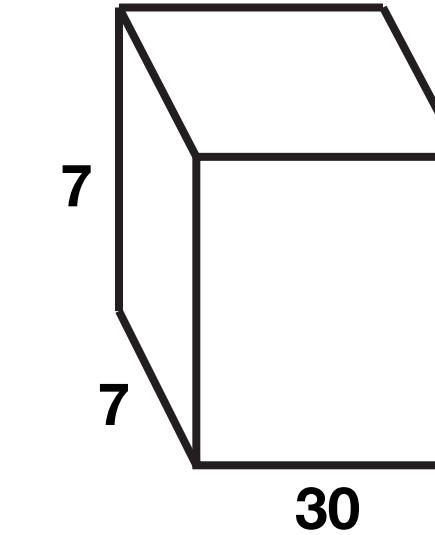
YOLO 算法理解 - 如何暴力 predict?



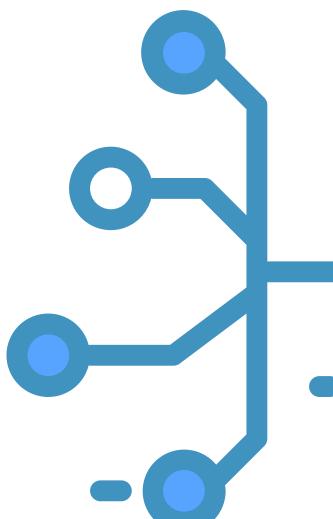
YOLO 的網路結構

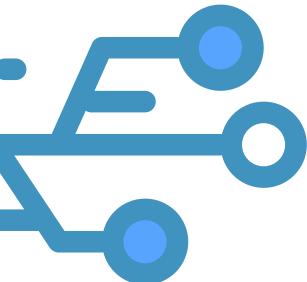


YOLO 算法理解 - 為什麼是 $7 \times 7 \times 30$?

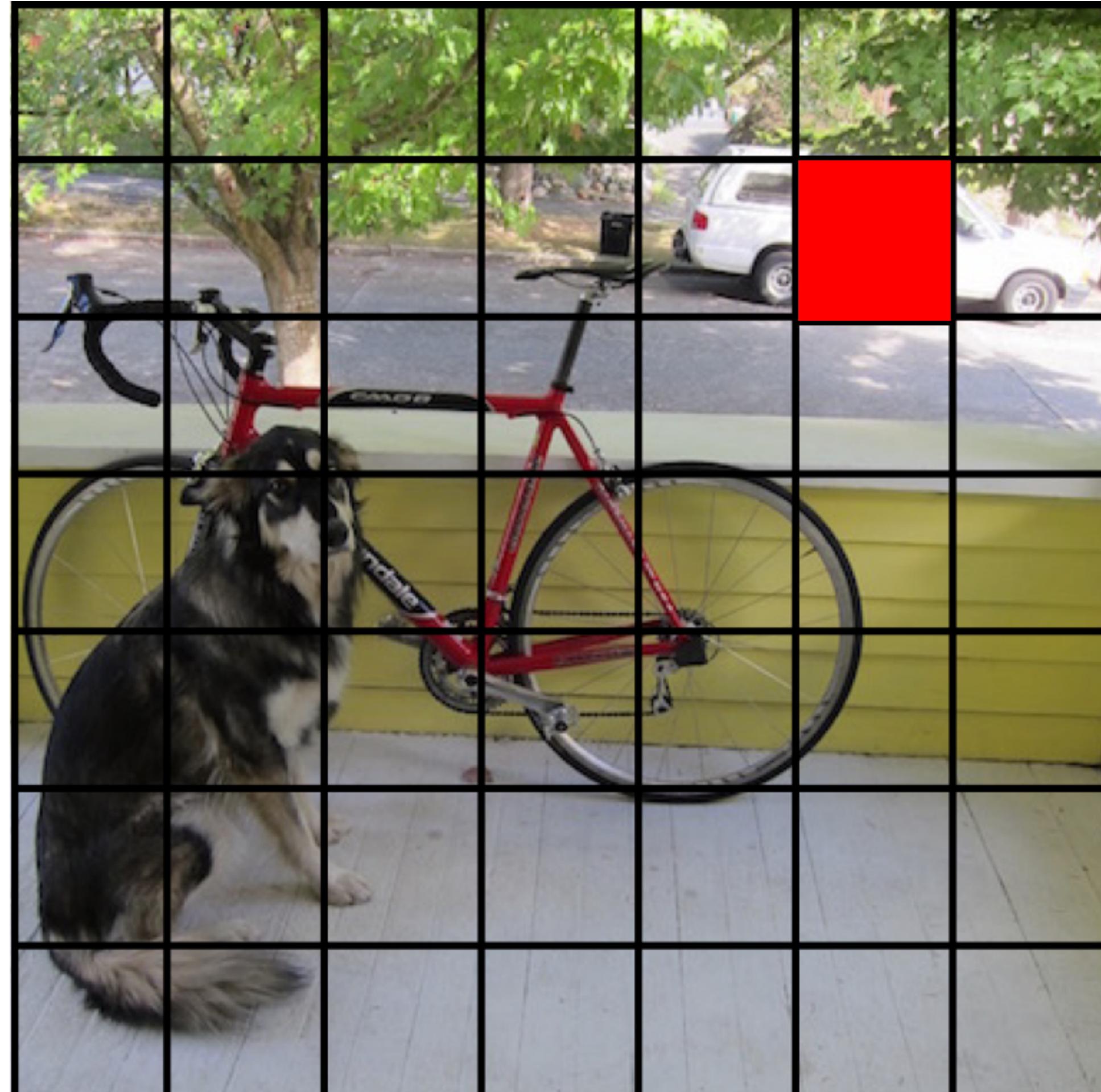


由之前的網路結構可以看出 v1 的輸出是一個 $7 \times 7 \times 30$ 的張量， 7×7 表示把輸入圖片劃分為 7×7 的網格， 7×7 的意義等下會詳細說明。而 $30 = (2^*5+20)$ ，代表 2 組表示 bbox 坐標信息的 5 個值 ($x, y, w, h, score$) 和而 20 表示論文採用的資料集 PASCAL VOC 中的物件類別數量。





YOLO 算法理解 - 7x7 的意義



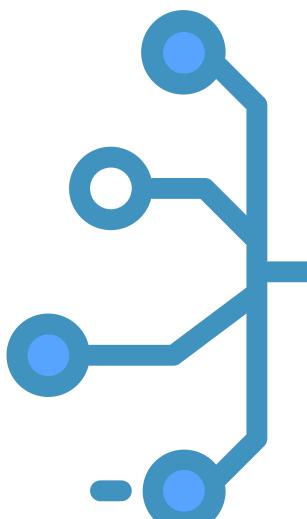
7

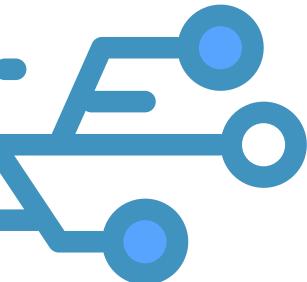
7

YOLO 將輸入圖像分成 7×7 個網格 (grid cell)，如果某個物體的中心落在這個網格中，則這個網格就負責預測這個物體。

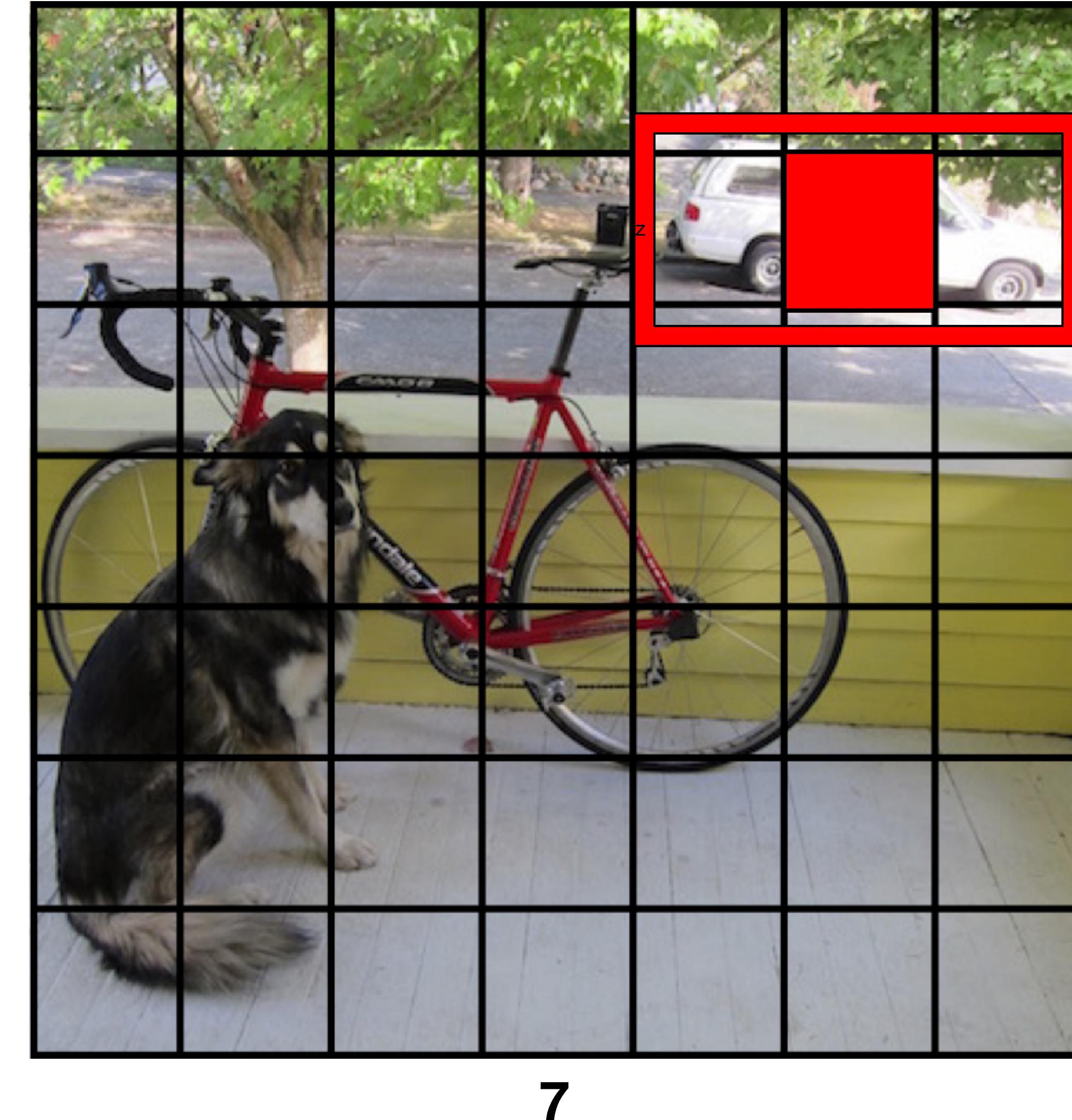
例子：左圖中的車落在紅色網格 (2, 6) 中，則該網格負責預測這輛車。

Note : (2, 6) 表示的是第 2 個 row，第 6 個 column 的網格。





YOLO 算法理解 - 7×7 的意義

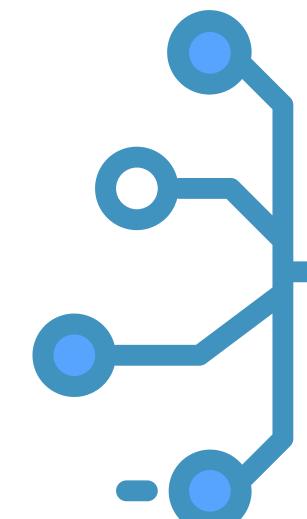


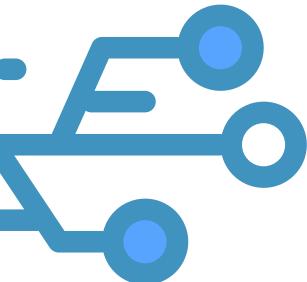
例子：左圖中的車落在紅色網格 (2, 6) 中，
則該網格負責預測這輛車。

YOLO 對每個 bbox 有 5 個預測的值：
 x, y, w, h 和 confidence

- x, y 代表該 bbox 的中心與網格邊界的相對值
- w, h 代表該 bbox 的寬高相對於輸入圖像寬高的比例
- confidence 代表該 bbox 有 object 的信心度，即該 bbox 與 groundtruth bbox 的 IOU 值

而每個網格負責預測 B 個 bbox 的信息，
YOLO 的設計中， $B = 2$

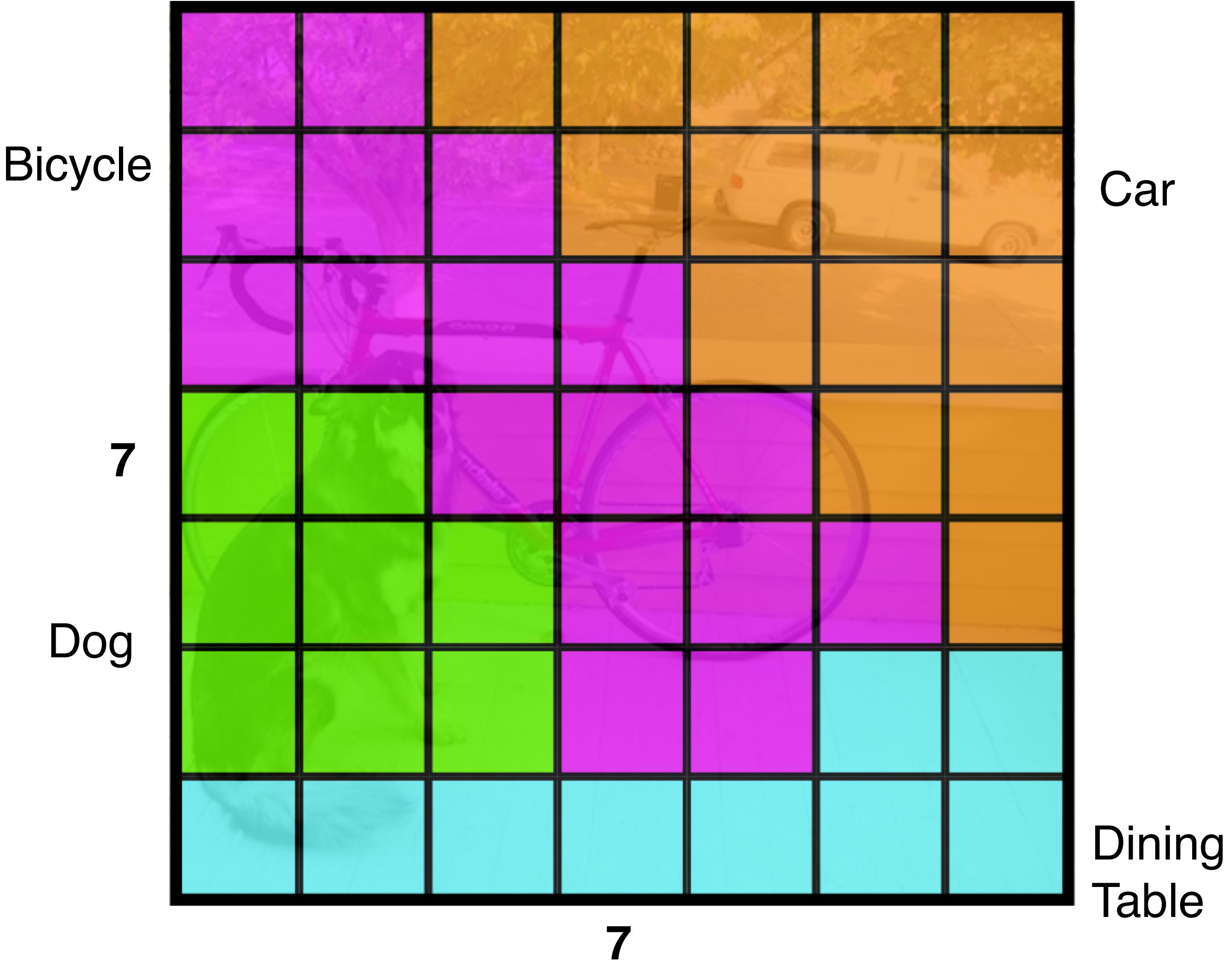




YOLO 算法理解 - 7x7 的意義



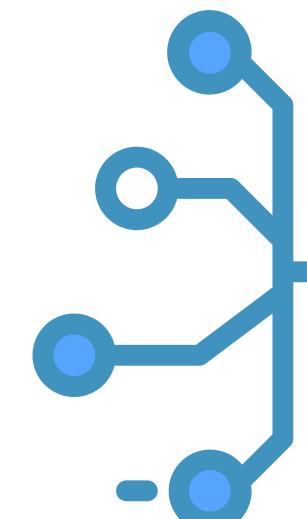
CUPOY

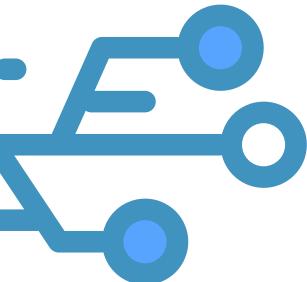


除此之外，每個網格還要預測有關類別的信息，如果有 C 個類別，就要預測 C 個類別的機率，取最高的作為該網格預測的物件類別，如左圖。

Note：類別機率是針對每個網格的，而 confidence 是針對每個 bbox 的。

到這裡，則 $S \times S$ 個網格，每個網格要預測 B 個 bbox 的信息以及 C 個類別機率。輸出就是 $S \times S \times (5 * B + C)$ 的一個 tensor。



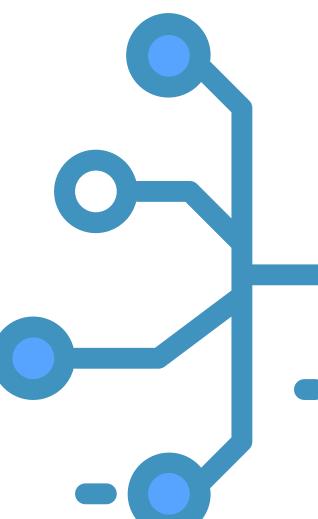
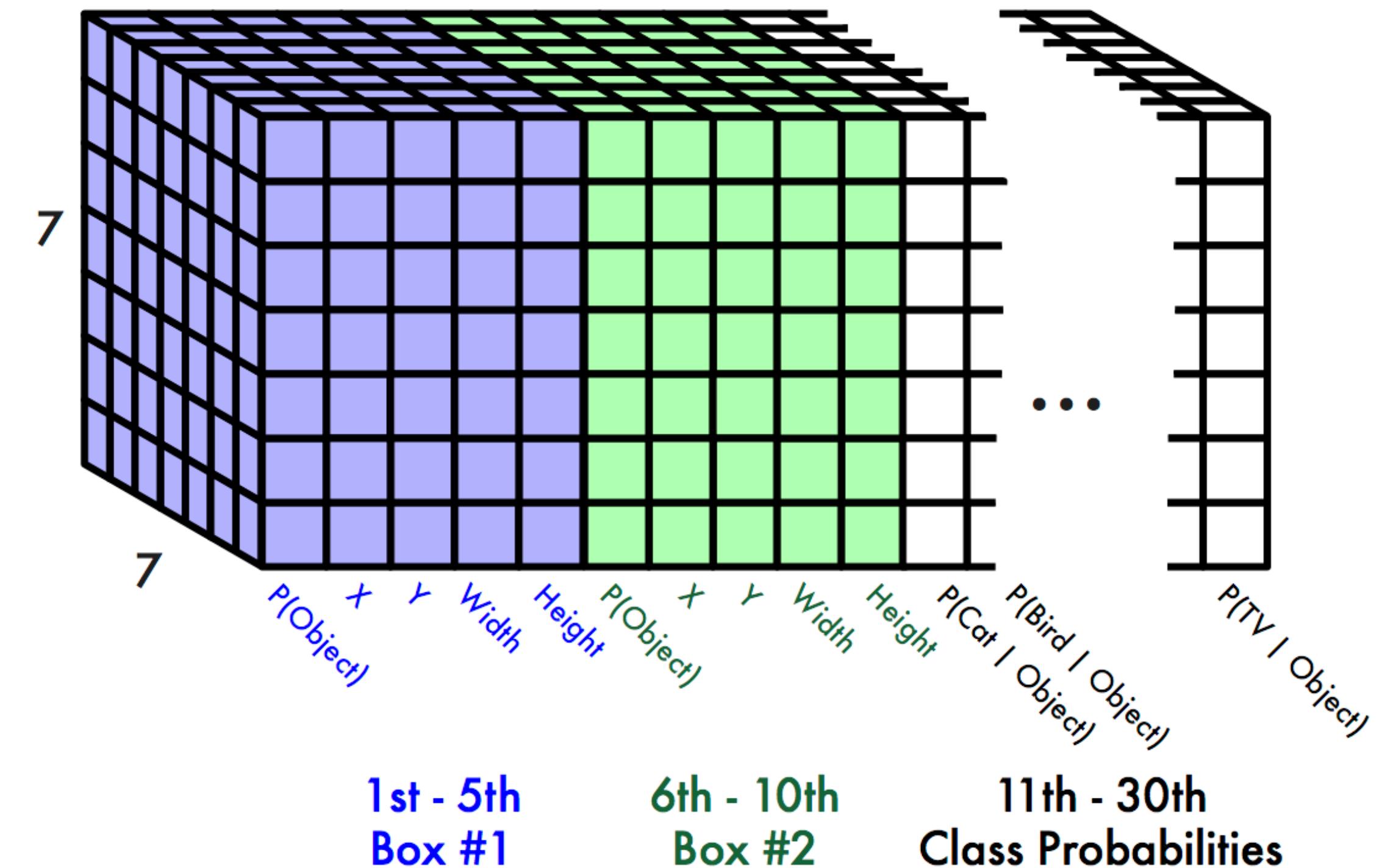


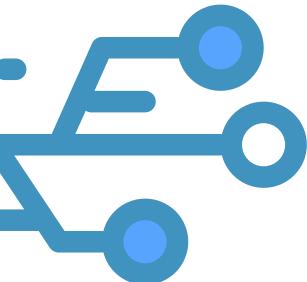
YOLO 算法理解 - output tensor 具體理解



右圖 $7 \times 7 \times 30$ tensor 為 YOLO 網路結構的輸出層，
每個 $1 \times 1 \times 30$ 的值對應原圖 7×7 網格中的一個，其中

- 第 1 到 5 的值(藍色) 預測
 - 第一個 bbox 的 ($x, y, w, h, confidence$)
- 第 6 到 10 的值 (綠色) 預測
 - 第二個 bbox 的 ($x, y, w, h, confidence$)
- 其餘 20 個值分別預測
 - Pascal VOC 中的 20 種 object

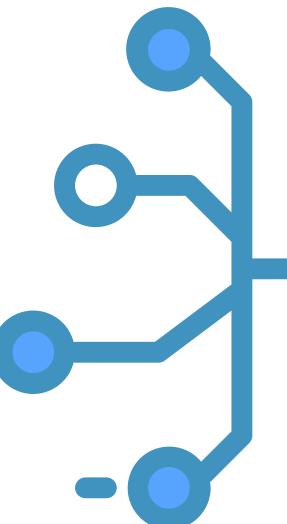
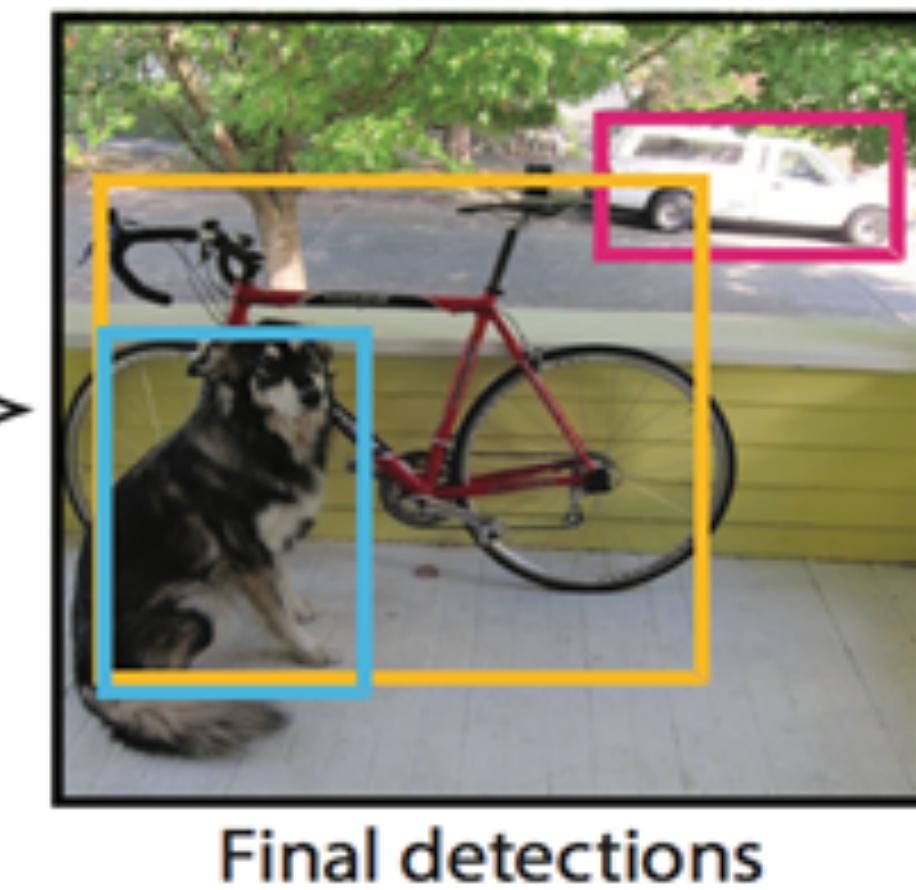
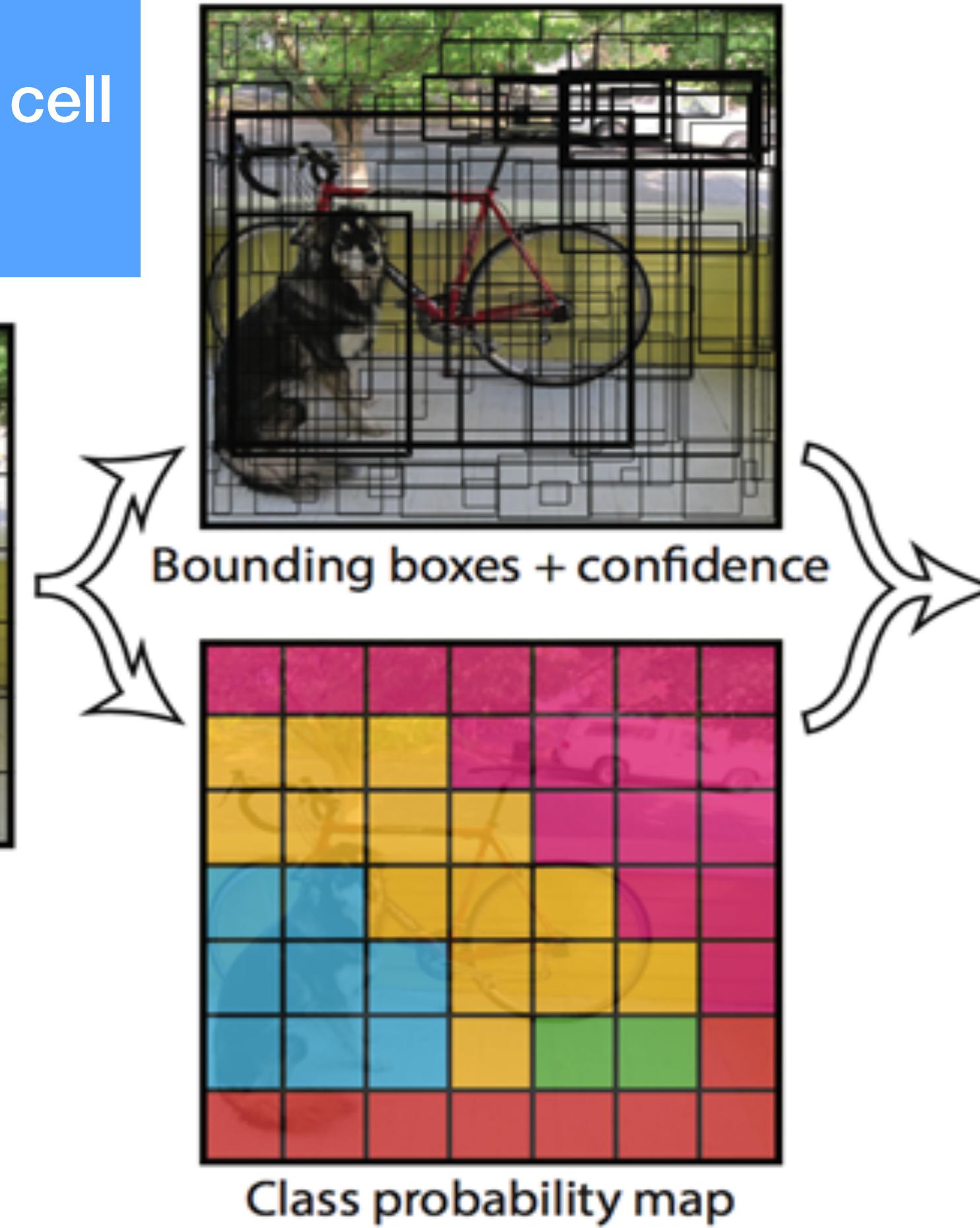
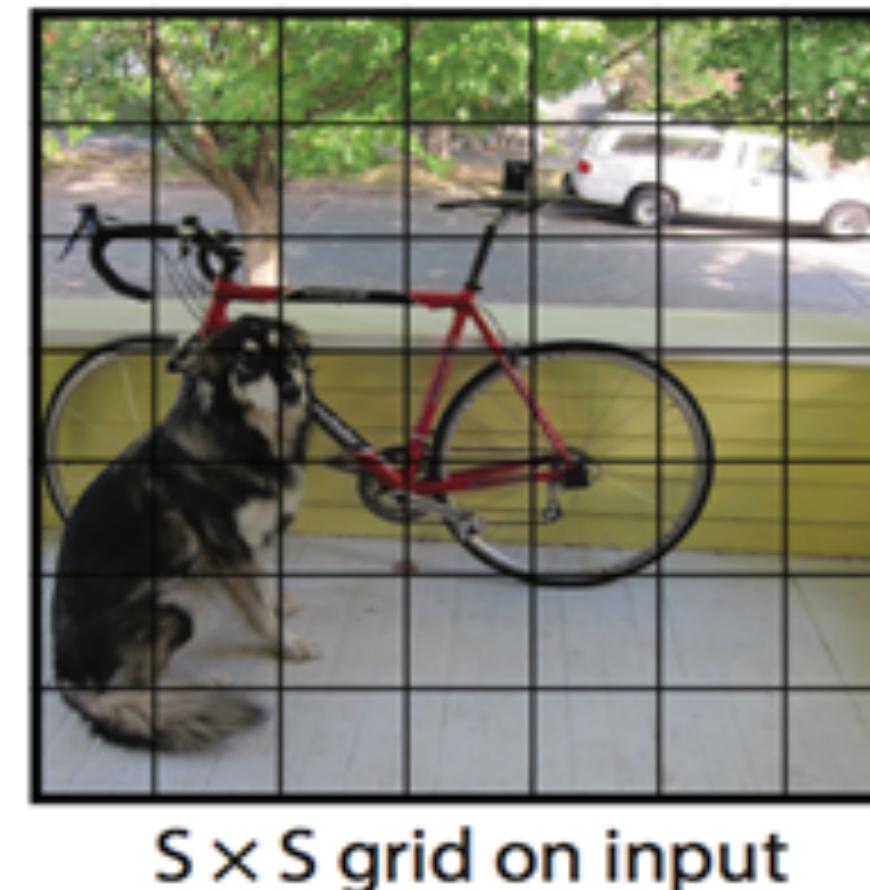


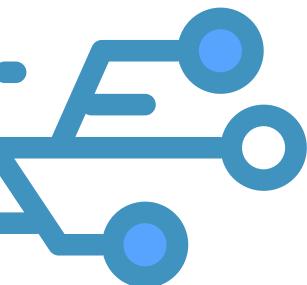


YOLO 算法理解 - 基本思路總結



把圖片劃分成 $S \times S$ 个grid cell
如果 object bbox 中心落在某個grid cell
內部，則該 cell 負責預測該 object

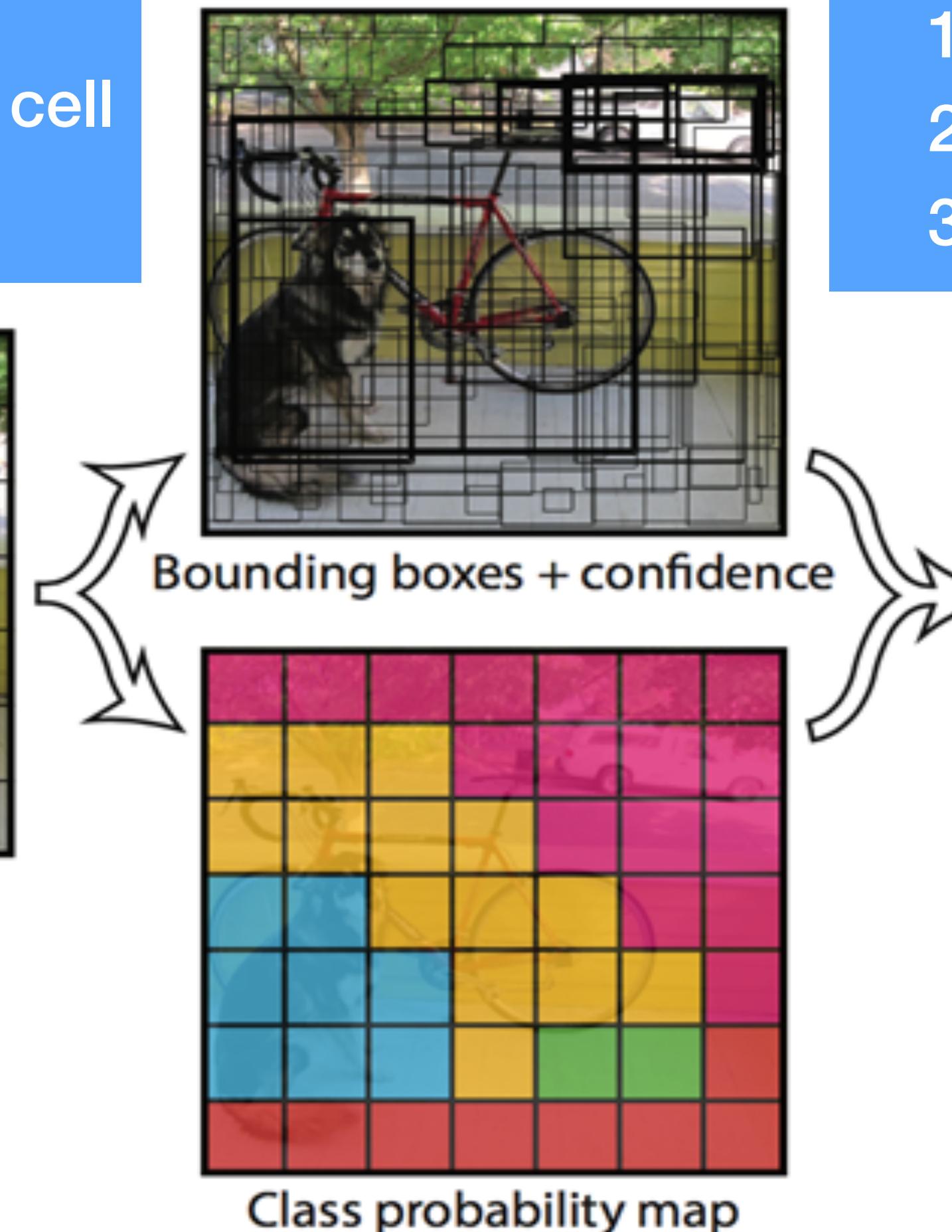
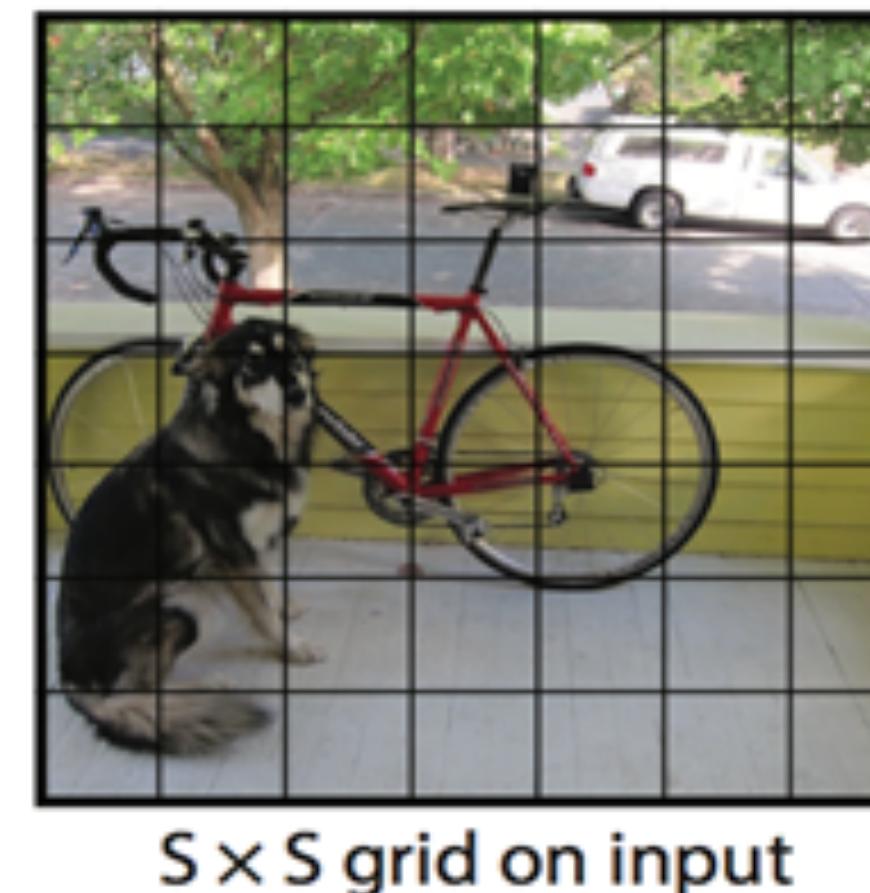




YOLO 算法理解 - 基本思路總結

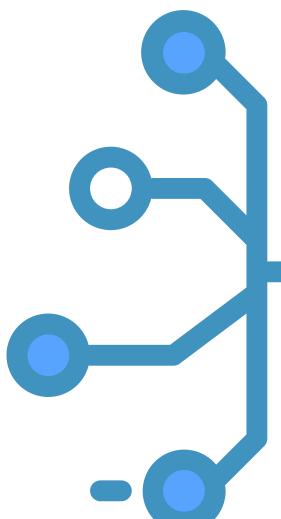


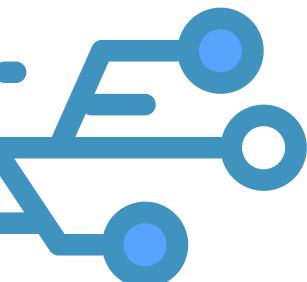
把圖片劃分成 $S \times S$ 個grid cell
如果 object bbox 中心落在某個grid cell
內部，則該 cell 負責預測該 object



每個 grid cell 預測

1. B 個 bbox 的位置 (x, y, w, h)
2. bbox 有沒有 object
3. C 個類別機率

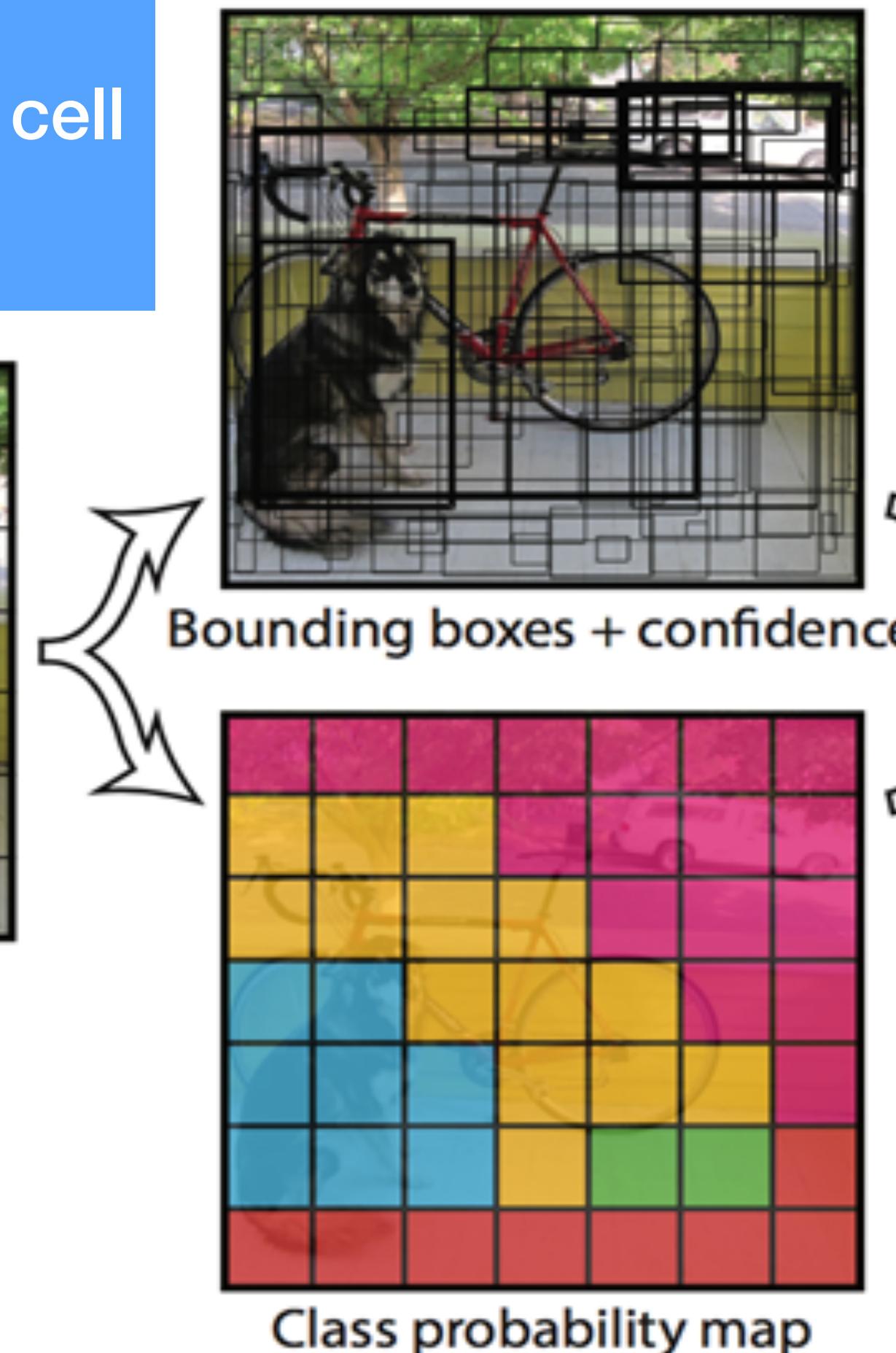
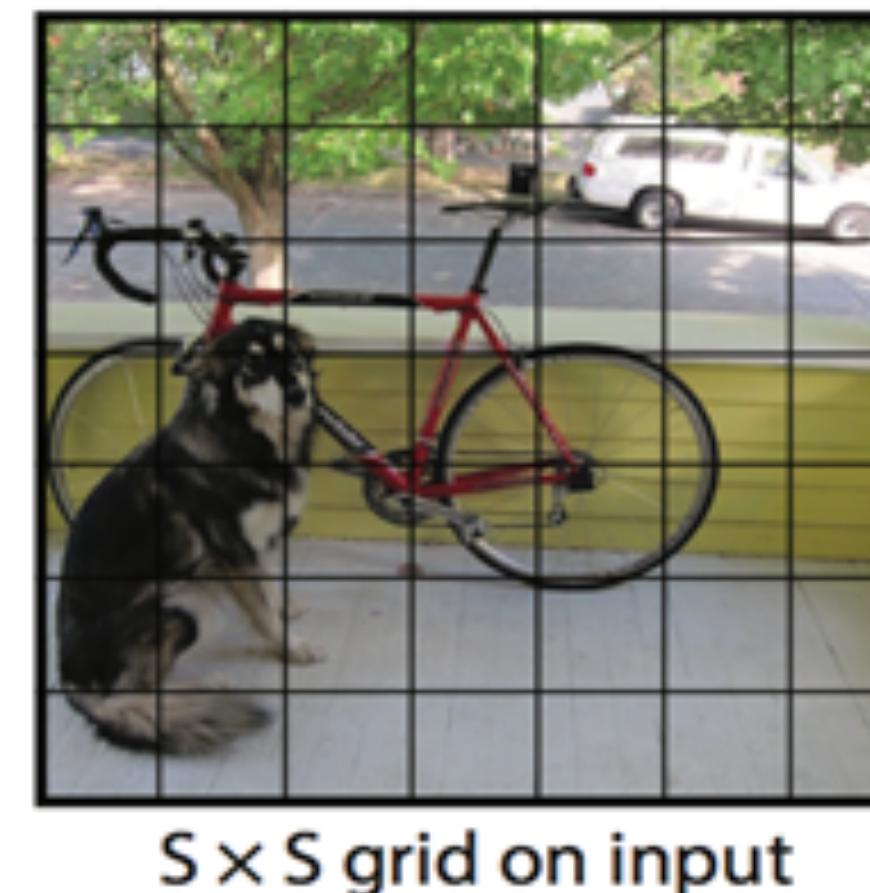




YOLO 算法理解 - 基本思路總結

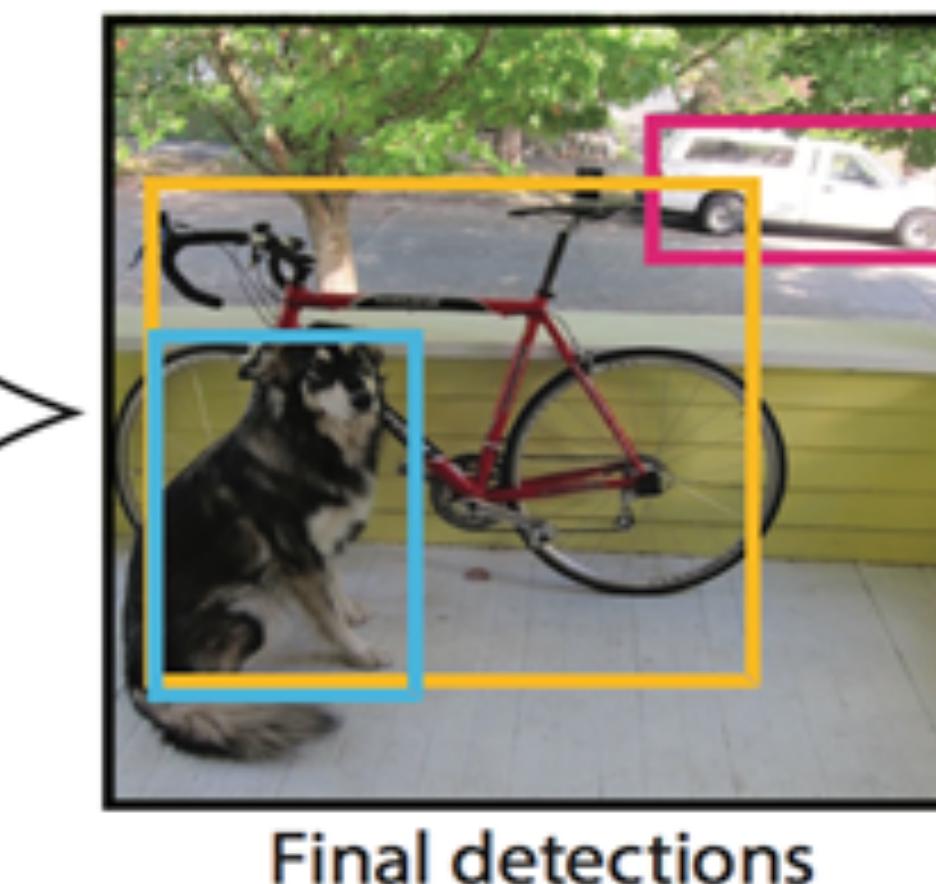


把圖片劃分成 $S \times S$ 個grid cell
如果 object bbox 中心落在某個grid cell
內部，則該 cell 負責預測該 object

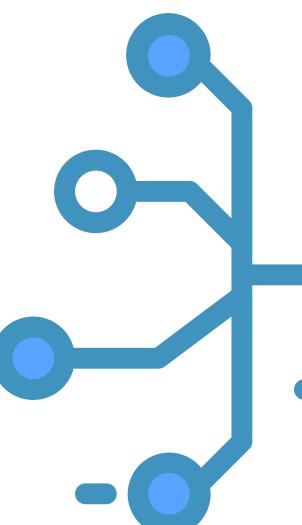


每個 grid cell 預測

1. B 個 bbox 的位置 (x, y, w, h)
2. bbox 有沒有 object
3. C 個類別機率

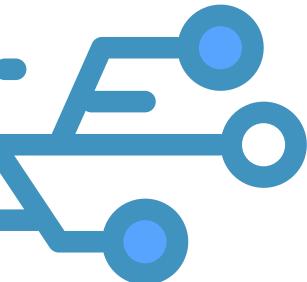


輸出 tensor: $S \times S \times (5B + C)$
 $S = 7, B = 2, C = 20$ (i.e 20 classes)
=> output 是 $7 \times 7 \times 30$ 的 tensor



知識點 回顧

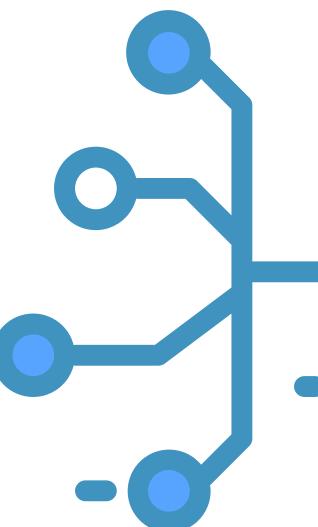
YOLO 的核心思想是直接從圖像來預測出 bbox 的坐標以及其類別，具體做法是把圖像切分為 $S \times S$ 個網格，每個網格預測 B 個 bbox 的信息以及 C 個類別機率，輸出就是 $S \times S \times (5 * B + C)$ 。因為這樣的設計使得 YOLO 檢測流程簡單，在保證一定檢測準確率的前提下，達到 45 fps (TitanX GPU) 的檢測速度。



參考資料



- YOLO 作者在 CVPR 2016 的簡報
- YOLOv1 : <https://pjreddie.com/darknet/yolov1/>
- YOLOv2 : <https://pjreddie.com/darknet/yolov2/>
- YOLOv3 : <https://pjreddie.com/darknet/yolo>
- 知乎 : 圖解 YOLO



解題時間 Let's Crack It



請跳出 PDF 至官網 Sample Code & 作業開始解題