

# 物件辨識與深度學習

David Chiu

# RCNN

# Sliding Window

用一固定大小的框框，逐一掃過整張圖片

將每次框出來的圖像丟到 CNN 中去判斷類別

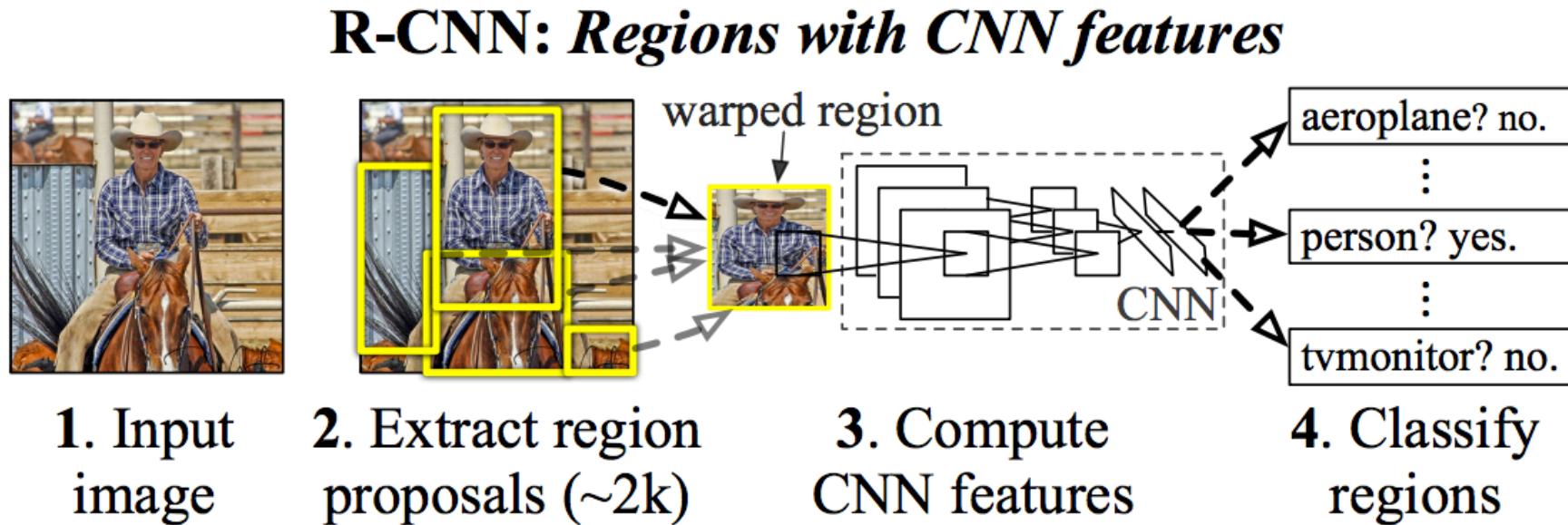
由於物體的大小是不可預知的，所以需用不同大小的框框去偵測



過於耗時

# R-CNN

- 產生一群約 2000 個可能的區域(region proposals)
- 由一個預先訓練好的 CNN 模型擷取特徵
- 以 SVM 來區分是否為物體或者背景
- 最後由線性回歸模型來校正物件位置(bounding box)



# Graph Base Image Segmentation

## ■ Graph Base Image Segmentation

## ■ Selective Search

1. 將圖片分割(Segment)的結果先各自畫出 bounding box
2. 找出合併相似度最高的兩個 box
3. 不斷合併成單一個 box 為止

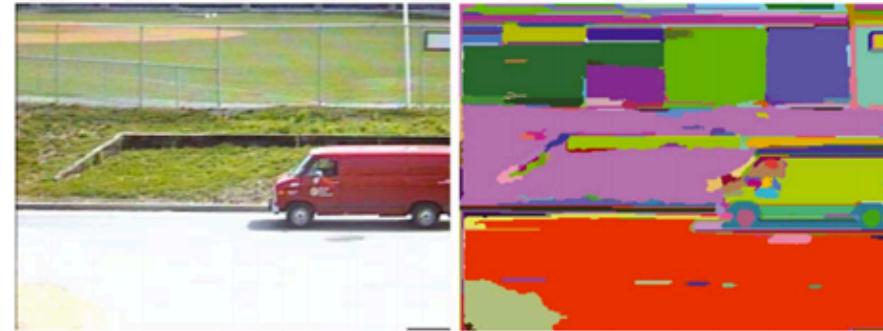


Figure 2. A street scene ( $320 \times 240$  color image), and the segmentation results produced by our algorithm ( $\sigma = 0.8, k = 300$ ).

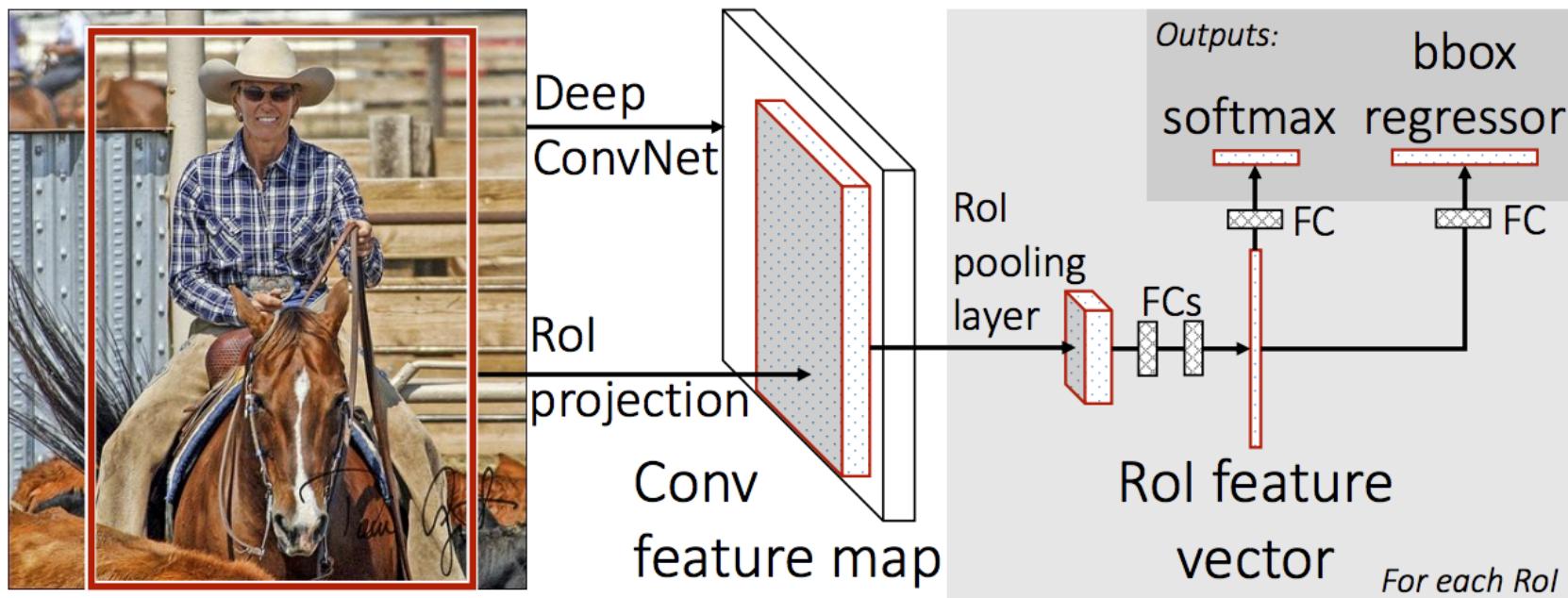


Figure 3. A baseball scene ( $432 \times 294$  grey image), and the segmentation results produced by our algorithm ( $\sigma = 0.8, k = 300$ ).

須針對所有的可能的區域(region proposals) 跑 CNN

# Fast R-CNN

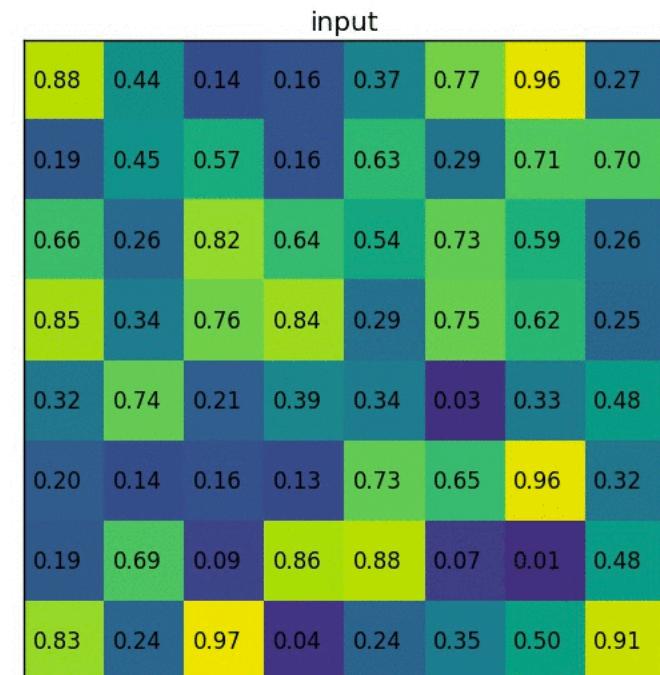
- 全部只算一次 CNN
- 將 CNN 撷取出來的特徵讓 2000 多個區域共用



<https://deepsense.ai/region-of-interest-pooling-explained/>

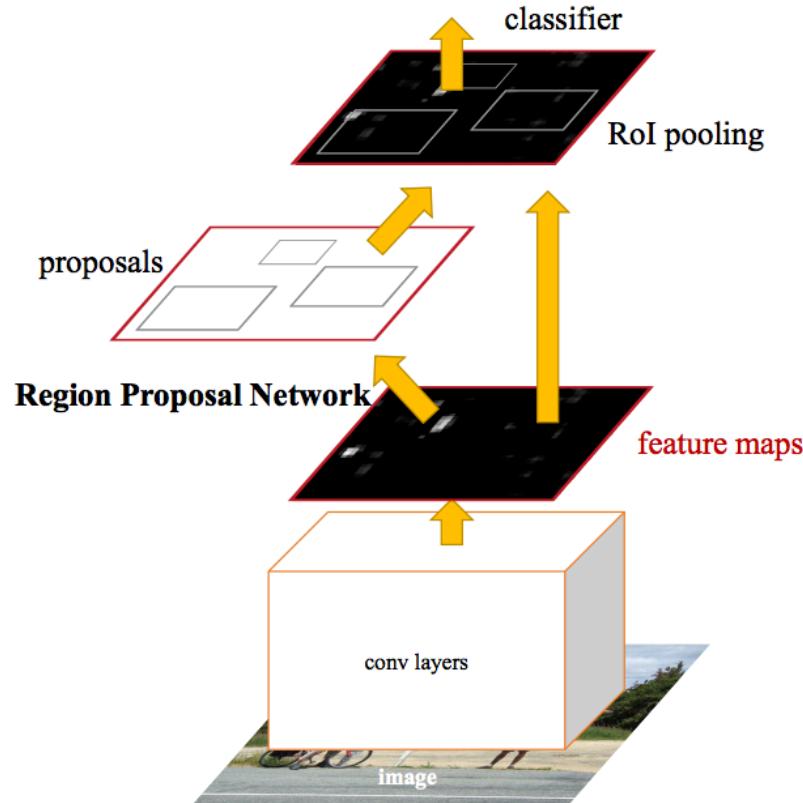
# Region of Interest Pooling

1. 預選可能區域 (Region proposals)
2. 做CNN
3. 得到一個 feature map ( $H * W$ )
4. 將預選可能區域(region proposals) 對應到 feature map 上
5. 取各自 region 的 MaxPooling
6. 各自連接上 FC 網路
7. 以 softmax 去作分類。
8. 作 bounding box 的線性回歸運算。



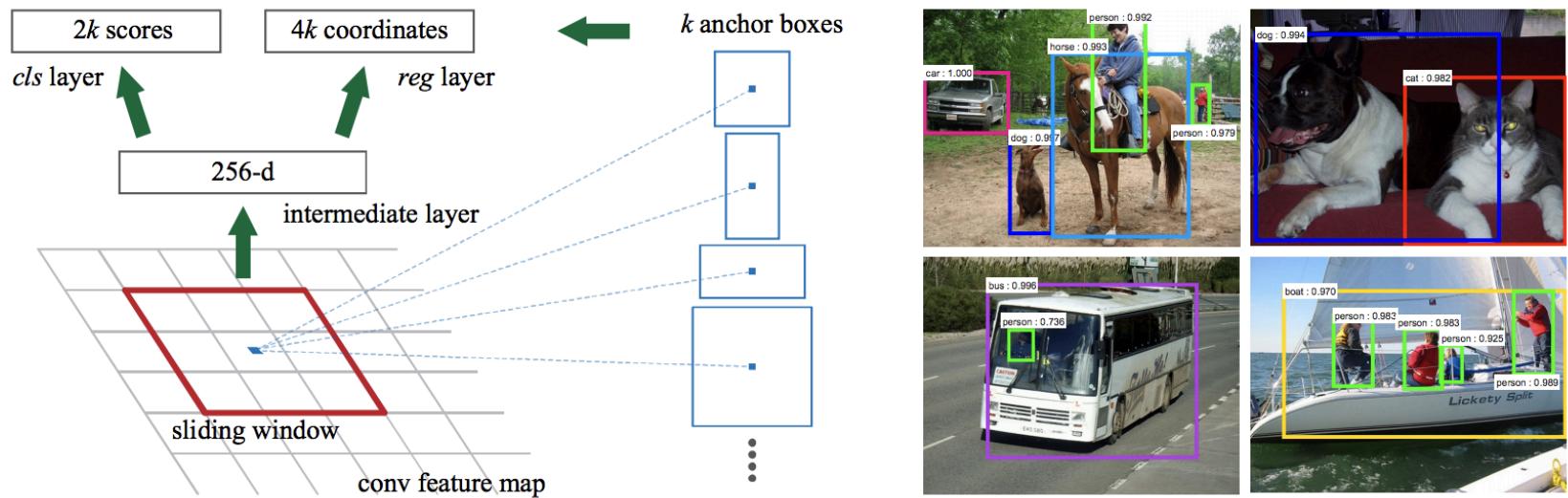
# Faster R-CNN

- 從 CNN 的 feature map 上選出 region proposals



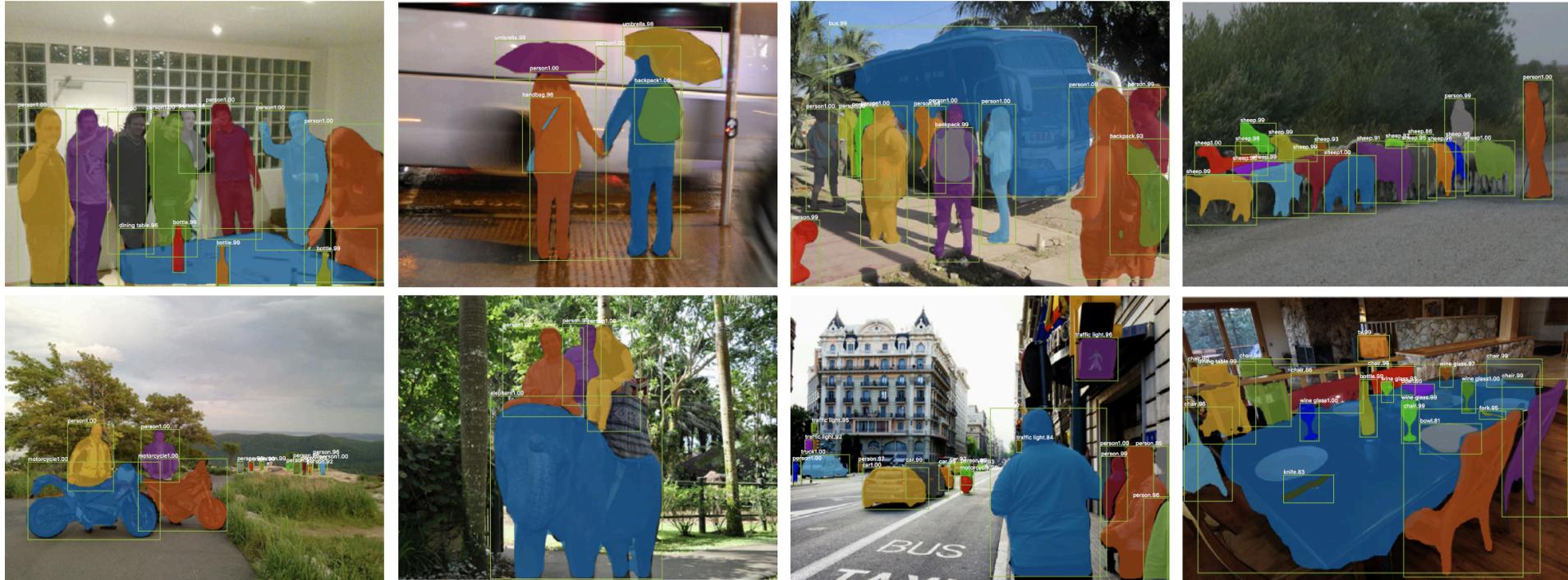
# Region Proposal Network

- RPN 在 feature map 上取 sliding window(anchor point)
  - 利用  $k$  個不同尺寸比例的 box 以同一個 anchor point 去計算可能包含物體的機率(score) · 取機率最高的 box 。(anchor box) 。每個 anchor point 會得到  $2k$  個 score · 以及  $4k$  個座標位置 (box 的X,Y,W,H) 。
- K 預設是取 3 種不同大小搭配 3 種不同長寬比的 anchor box · 所以為  $3 \times 3 = 9$



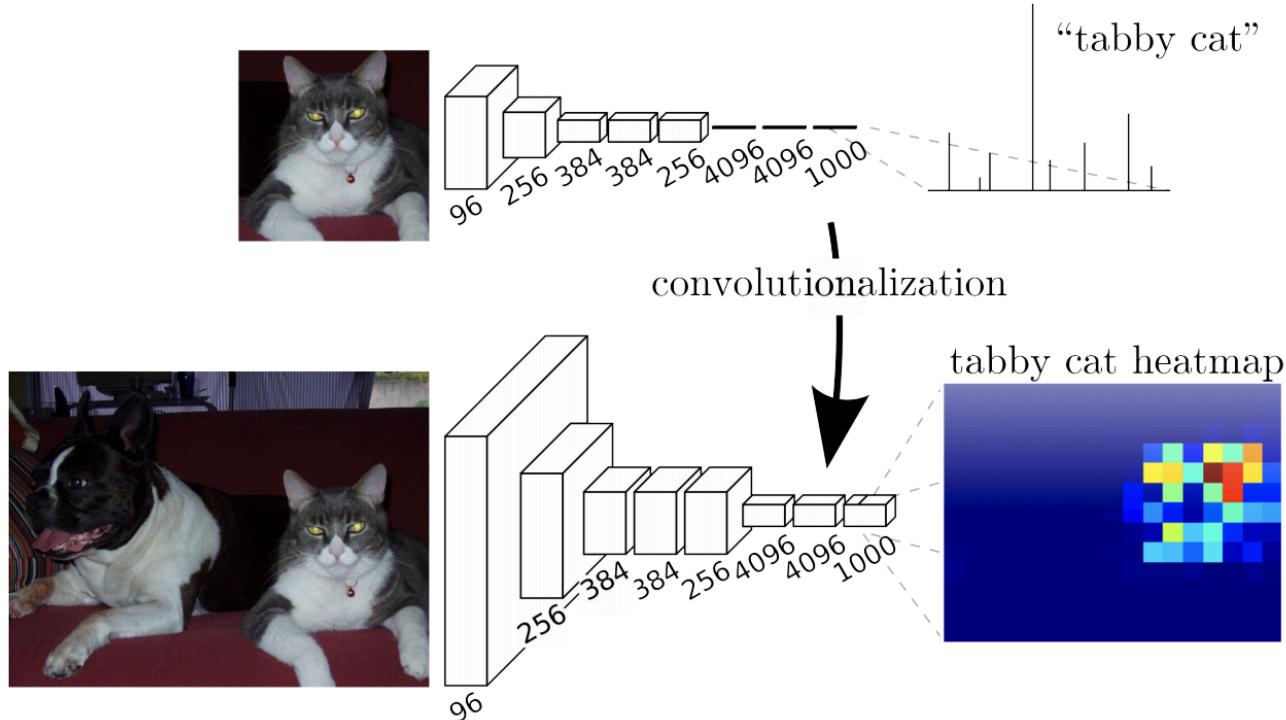
# Mask R-CNN

■ Mask R-CNN 可以找出物件輪廓，不只是 bounding box而已



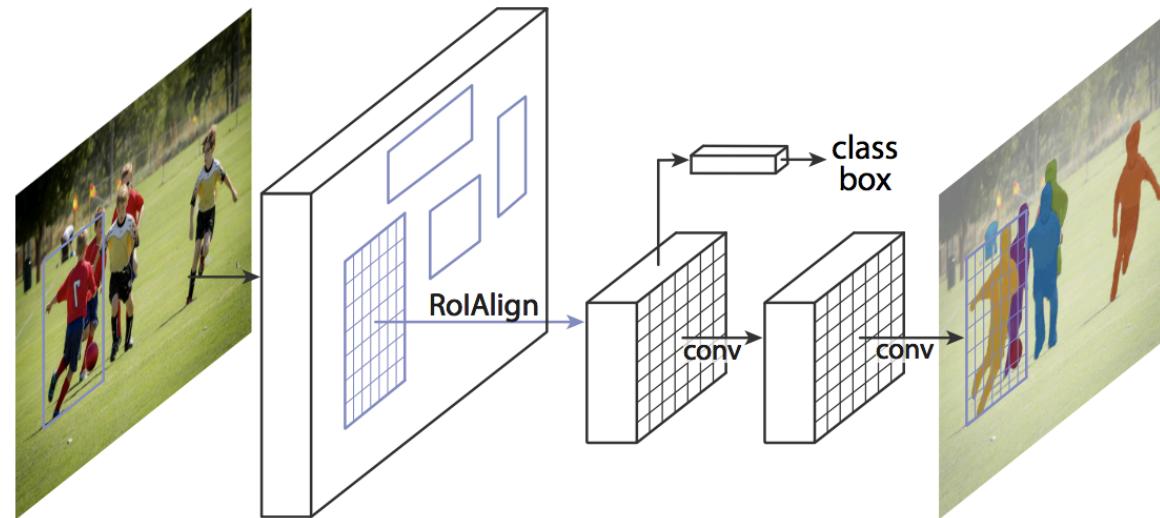
# FCN (Fully Convolutional Network)

- CNN 網絡，只能接受大小固定的輸入，得到每個類別的機率
- FCN 網路，最後兩層由卷積取代，輸出為  $h \times w \times 1000$ ，代表每個 pixel 種類的機率



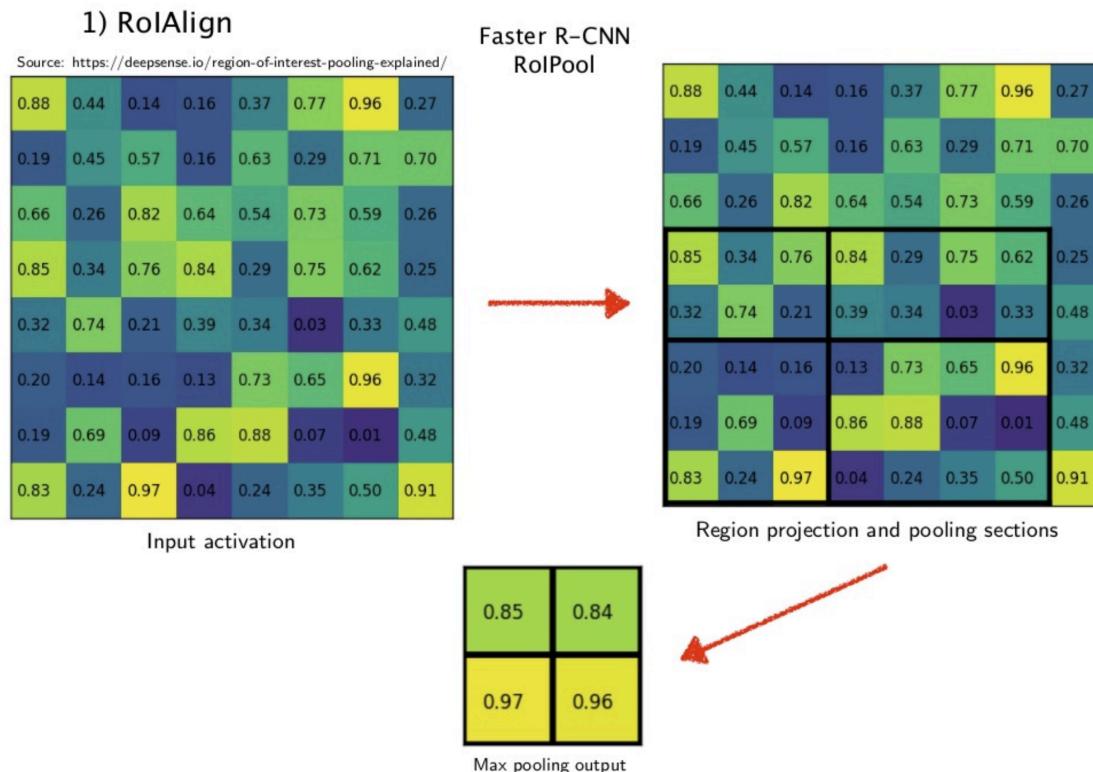
# Mask R-CNN

- 建構於 Faster R-CNN 之上，
- 透過 RoIPooling 取得 Region proposals 之後，針對每個 region 會再跑 FCN 取得遮罩分割
- Mask R-CNN 採用雙線性插值法([Bilinear Interpolation](#))來改善 RoIPooling，稱之為 RoIAlign，RoIAlign 會讓遮罩位置更準確。



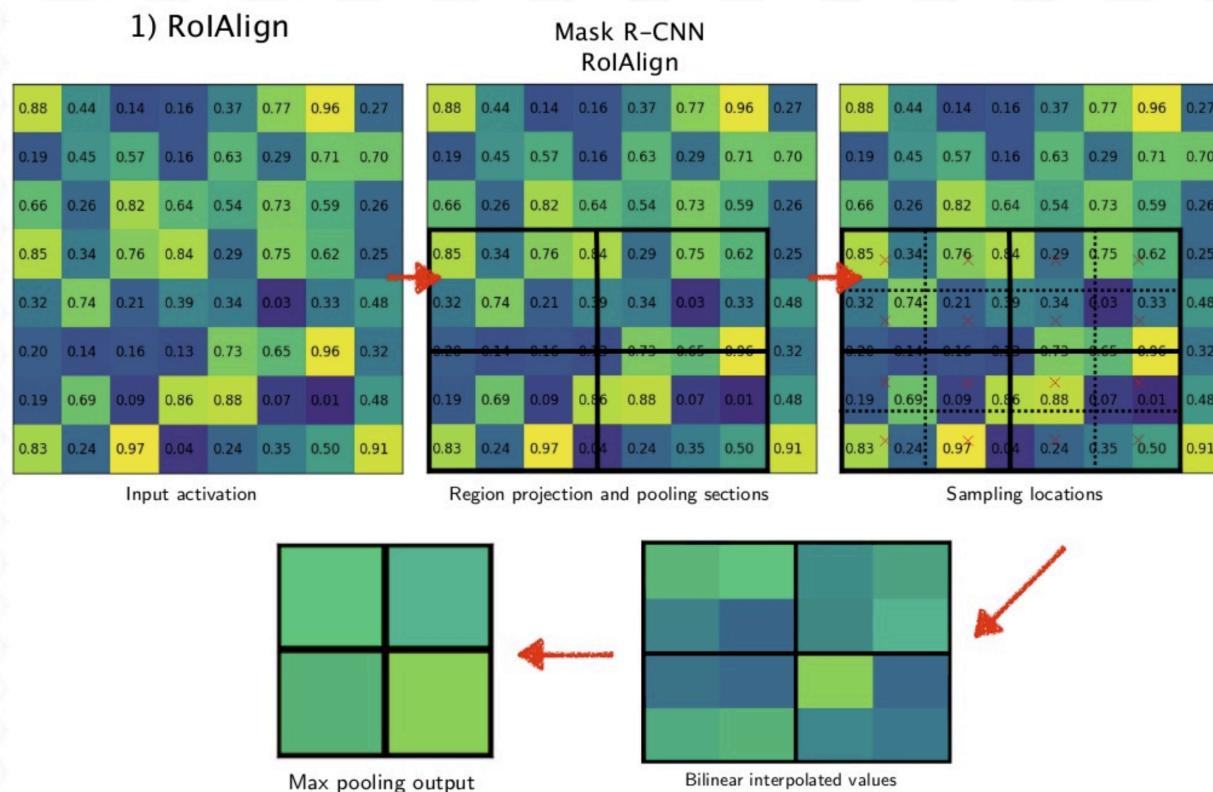
# ROI Pooling

■ 將一個 7x5 的 Anchor box 取 2x2 的 MaxPooling，由於使用最近插值法，會有偏差



# ROI Pooling

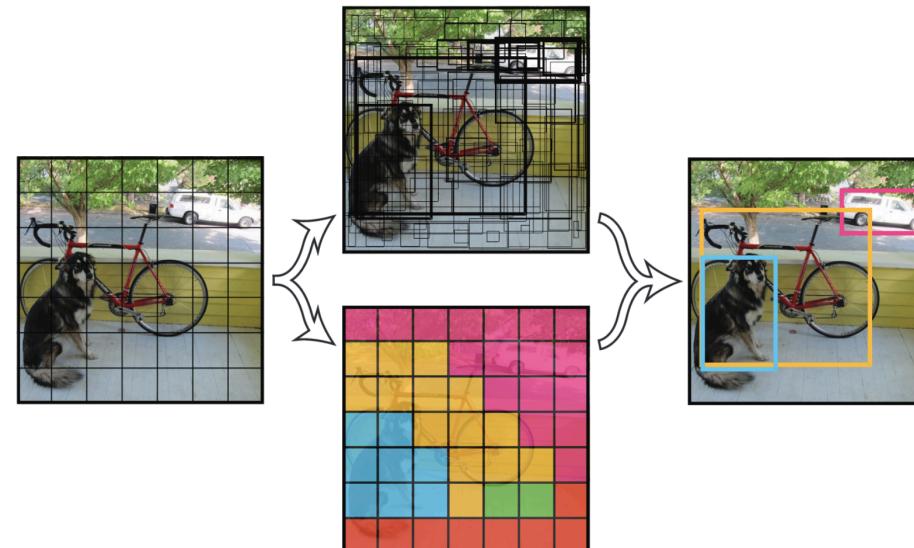
- 使用雙線性插值法(Bilinear Interpolation)，減少mis-alignment的問題



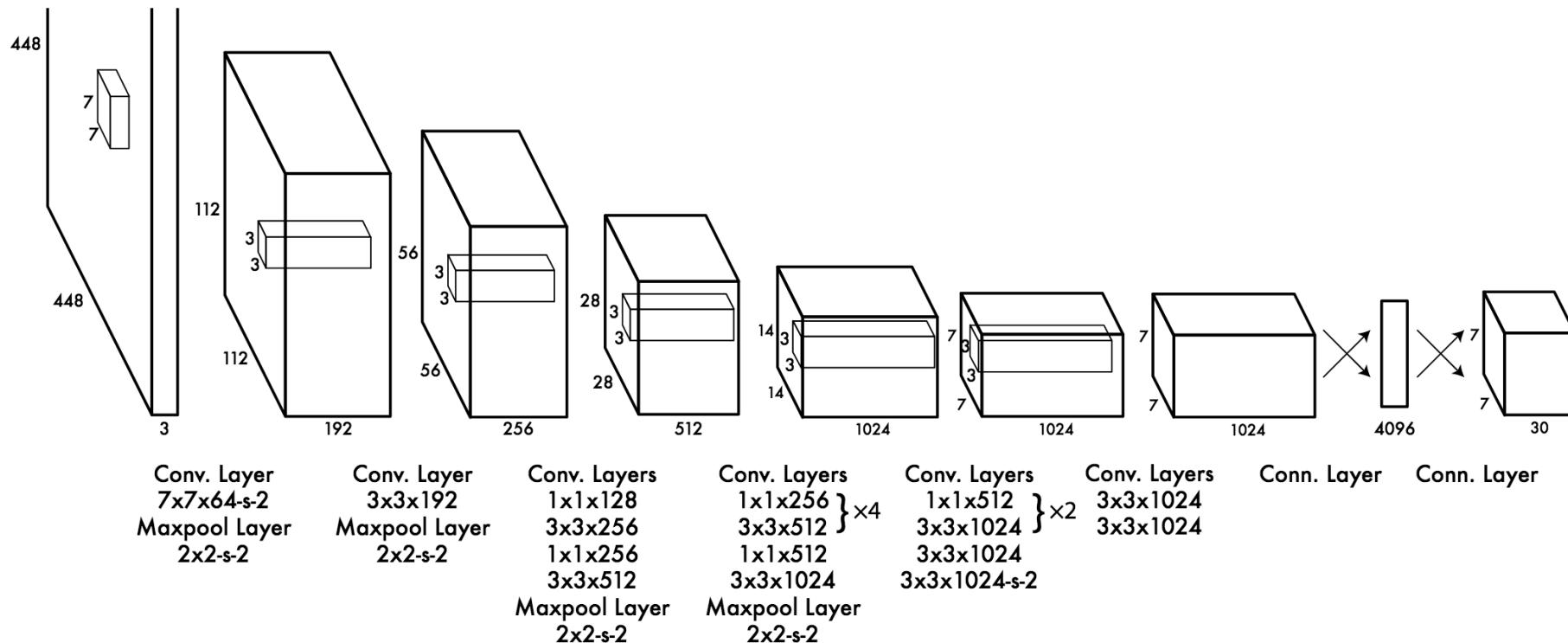
**Yolo**

# YOLO: You Only Look Once

- 將一張圖片切割成  $S \times S$  個方格，每個方格以自己為中心點各自去判斷  $B$  個 bounding boxes 中包含物體的 confidence score 跟種類
- $confidence\ score = Pr(Object) * IOU\ (ground\ truth)$ 
  - 如果該 bounding box 不包含任何 object，confidence score 便為零，
  - IOU 則為 bounding box 與 ground truth 的交集面積



# Yolo Network

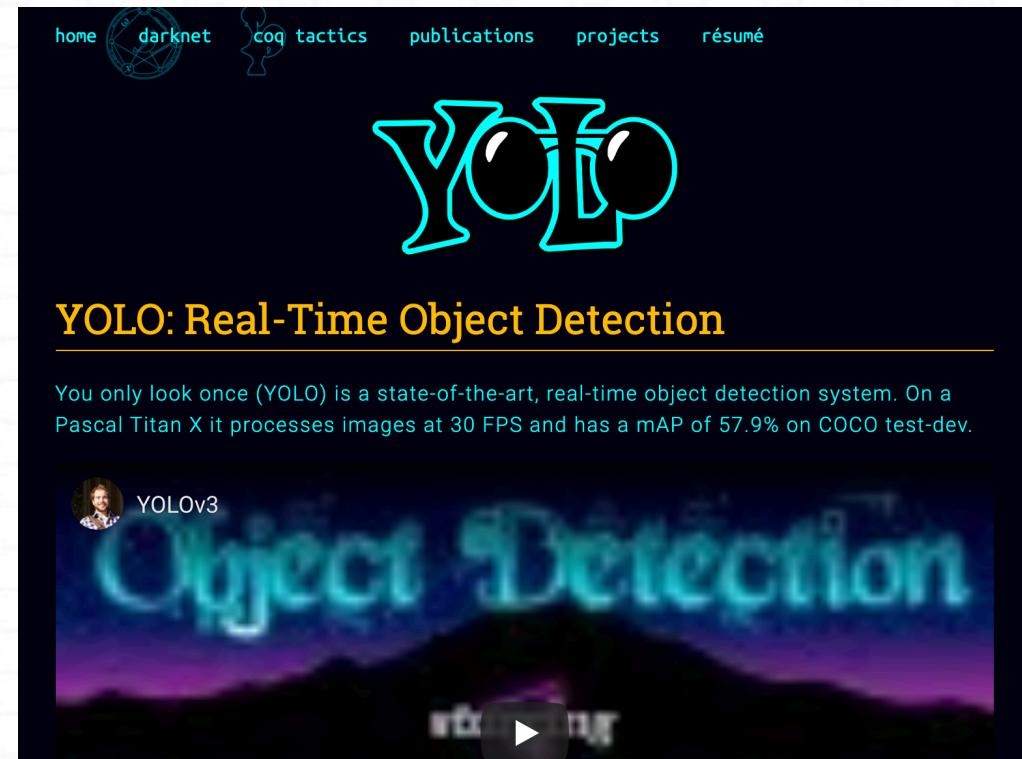


# Yolo v.s. RCNN

- YOLO 速度快
- YOLO 一次看整張圖片，背景錯誤偵測率 都只有 Fast R-CNN 的一半
- YOLO 的泛用性也比 R-CNN 或者 DPM 方式來得好很多

# 使用 Yolo

- 下載專案
  - git clone <https://github.com/pjreddie/darknet>
- 編譯專案
  - cd darknet
  - make
- 下載權重
  - wget <https://pjreddie.com/media/files/yolov3-tiny.weights>
- 執行辨識任務
  - ./darknet detect cfg/yolov3-tiny.cfg yolov3-tiny.weights data/dog.jpg



<https://pjreddie.com/darknet/yolo/>

# 參考內容

- 目标检测YOLO、SSD、RetinaNet、Faster RCNN、Mask RCNN(1)

<https://www.jianshu.com/p/3efeb56bd0ab>

- Region of interest pooling explained

<https://deepsense.ai/region-of-interest-pooling-explained/>

- 關於影像辨識，所有你應該知道的深度學習模型

<https://medium.com/cubo-ai/%E7%89%A9%E9%AB%94%E5%81%B5%E6%B8%AC-object-detection-740096ec4540>



# **THANK YOU**