Final Project

- Number of people: 1~2
- Form: written report (ppt)
- Scope: any topics related to CV
- Due: 12/24 (Tuesday)

Structure (just for reference)

- Introduction
  - Introduce your task and what you are trying to achieve
- Method
  - Data preparation and preprocessing
  - Model selection and training process
  - Etc.
- Results
- Demo (if you build a system)
  - Link, snapshot, or video

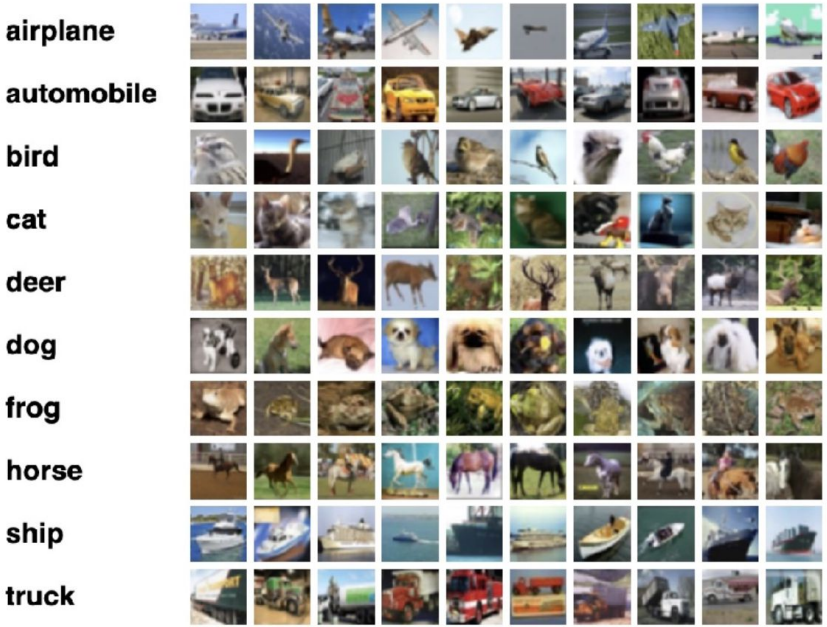
Grading Metrics

- Completeness (60%)
  - Details, correctness, etc.
- Novelty (20%)
  - Task, method, etc.
- Presentation (20%)
  - Demo, ppt slides, etc.

物件偵測

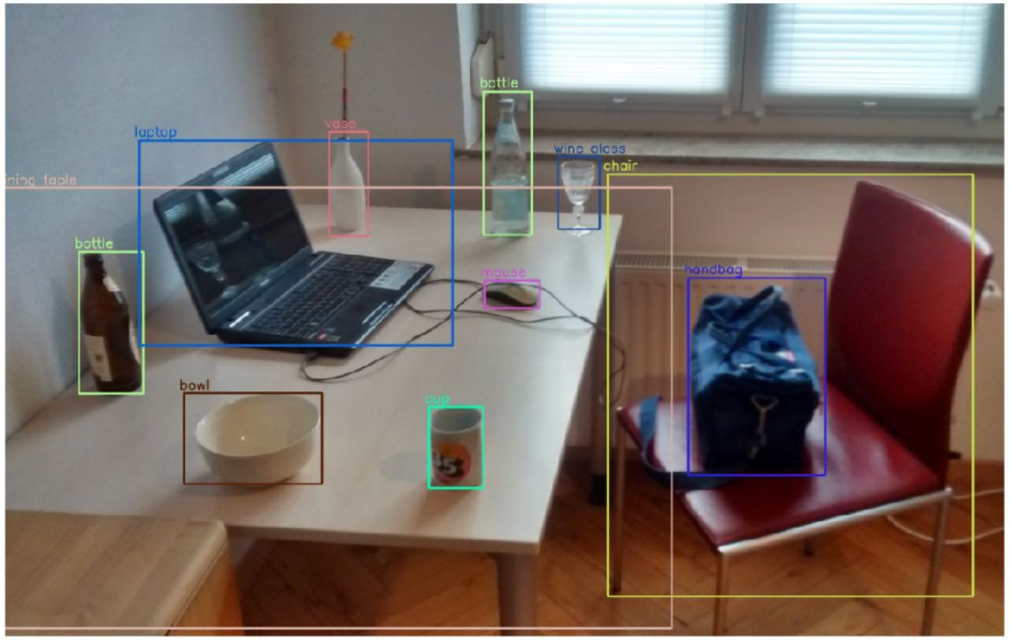
什麼是物件偵測

- 到目前為止，我們已經處理了許多影像分類的任務，也就是從整體上確定影像包含的內容



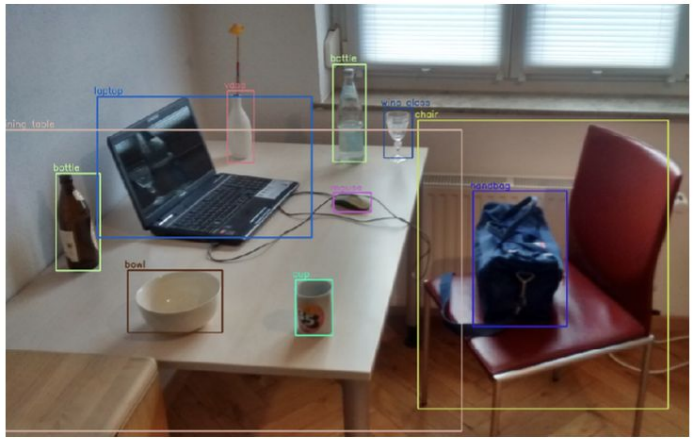
什麼是物件偵測

- 但是如果我們想對圖像中的物件進行單獨分類呢？



物件偵測容易嗎？

- 物件偵測是一個更複雜的問題
- 我們需要同時做兩件事:
  - 確定感興趣的物體所在的位置 - **Localization**
  - 確定該區域內的物體是什麼 - **物件分類**



物件偵測器

- 物件偵測器通常針對一個或多個類別進行訓練，並尋求產生一個已識別物件類別的邊界框
- 深度學習的方法：R-CNNs, SSDs, YOLO, Detectron, EfficientDet and RetinaNet
- 在深度學習方法中，有兩種主要類型：
- **Two-Shot** - 使用兩個階段，先進行區域提議，然後進行分類
- **Single Shot** - 同時進行定位和圖像分類

Object Segmentation

- 類似物件偵測，但使用不同的演算法
- 分割涉及不同物件類別的**像素級分類**

Classification vs Detection vs Segmentation

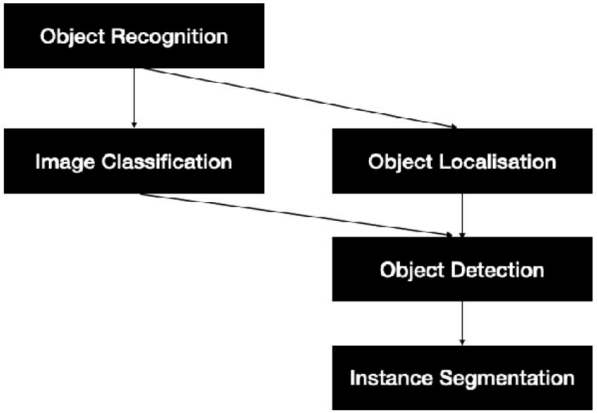
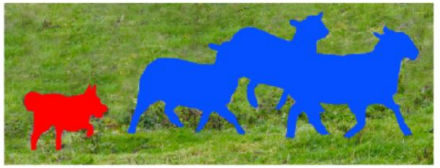
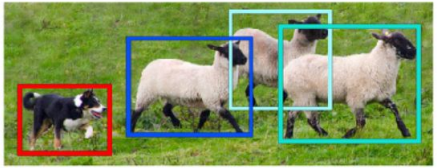
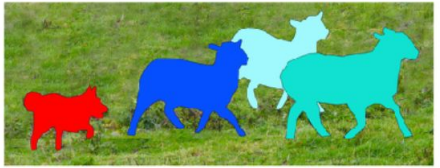
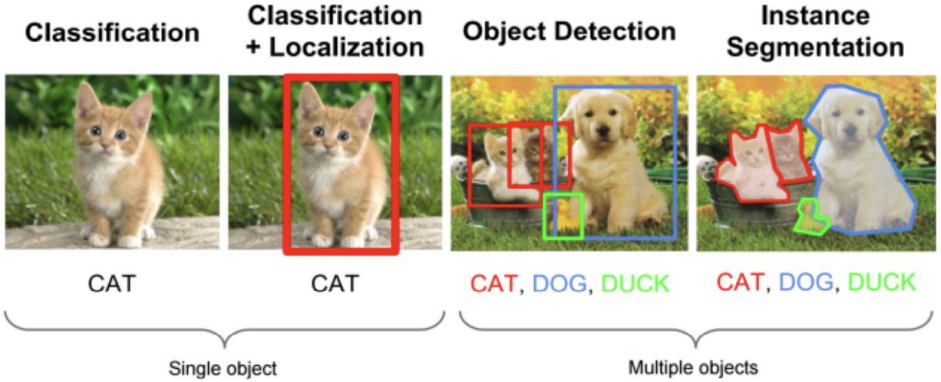


Image Recognition


Semantic Segmentation


Object Detection

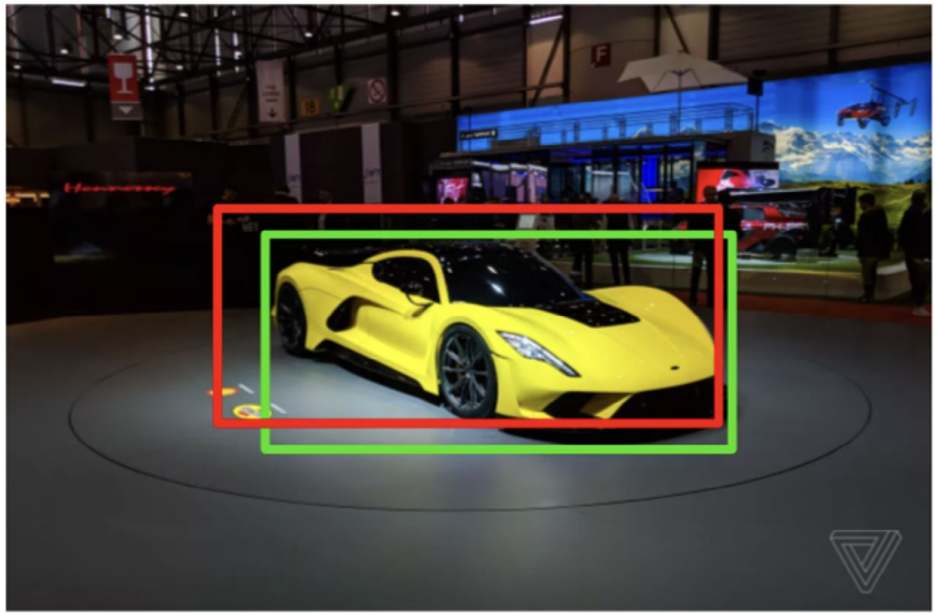

Instance Segmentation



Object Detection 應用

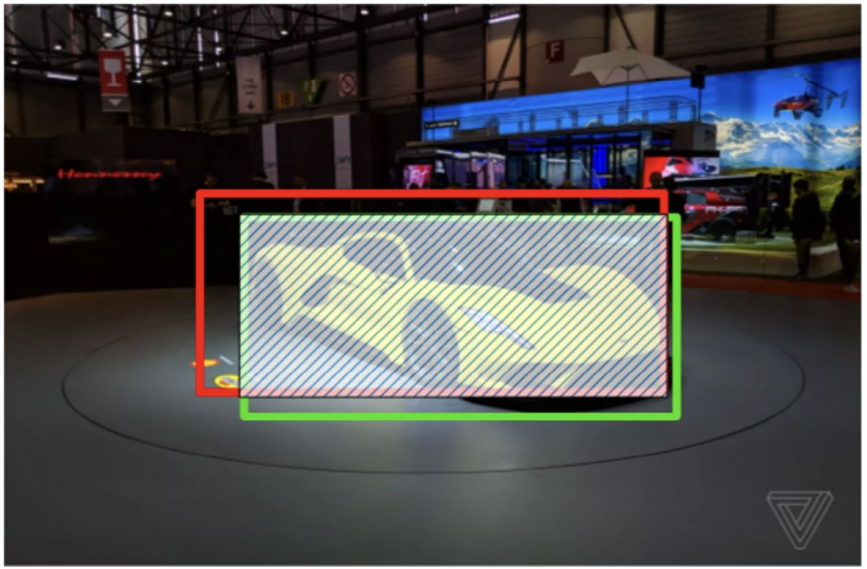
評估物件偵測器

- 假設綠色框是我們的真實答案
- 紅色框我們的物體偵測器提出的提案
- 如何評價這個提案的好壞？



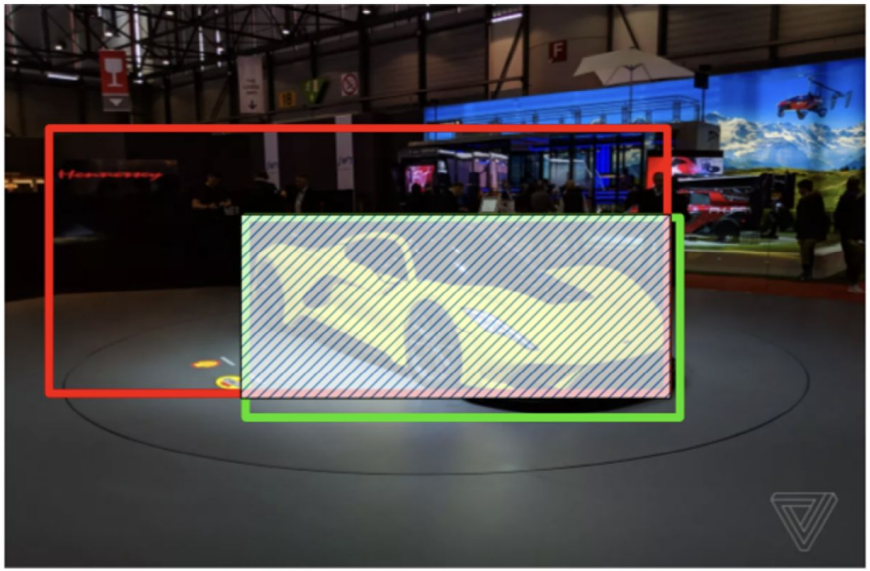
評估物件偵測器

- 看起來它幾乎覆蓋了 90% 的真實情況
- 然而，這是一個很好的衡量標準嗎？



評估物件偵測器

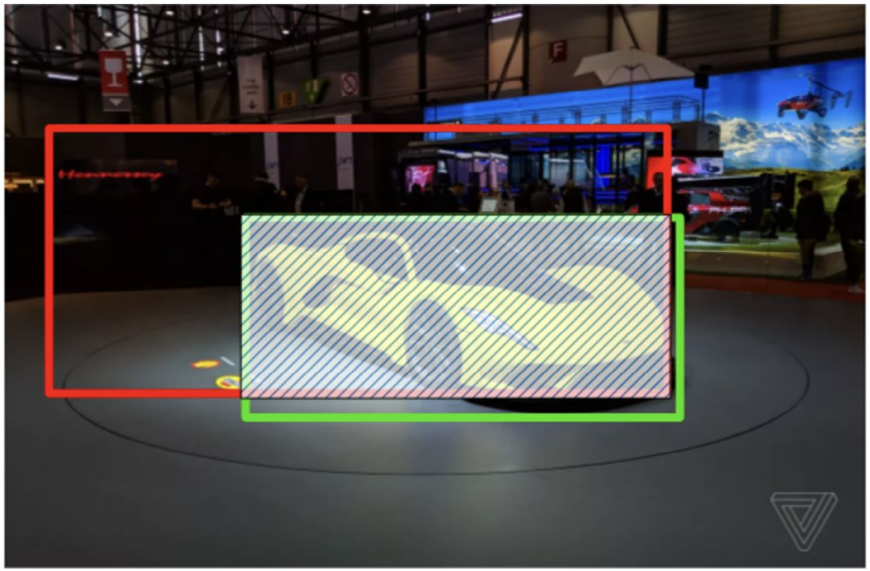
- 這個紅色框也覆蓋了綠色框的 90%
- 我們可以使用 **Intersection per Union** (IOU) 來解決這個問題



評估物件偵測器

$$IoU = \frac{\text{Size of Union}}{\text{Size of Prediction}}$$

- IoU 考慮重疊的部分與邊界框大小的比率
- IoU 超過 0.5 代表是可以接受的
- IoU 越高，邊界框越好



Mean Average Precision (mAP)

- mAP 是比較物件偵測模型的最佳方法
- mAP 結合了多個指標

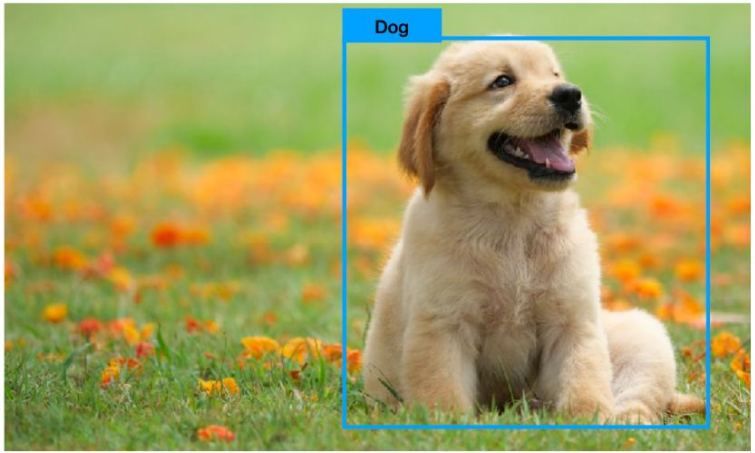
Precision, Recall 回顧

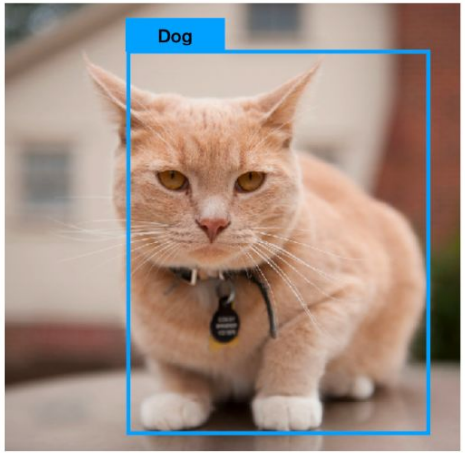
- Precision: 當你的模型預測為 positve 時，正確的機率是多少
- Recall: 你的模型在發現所有 positve 的表現如何

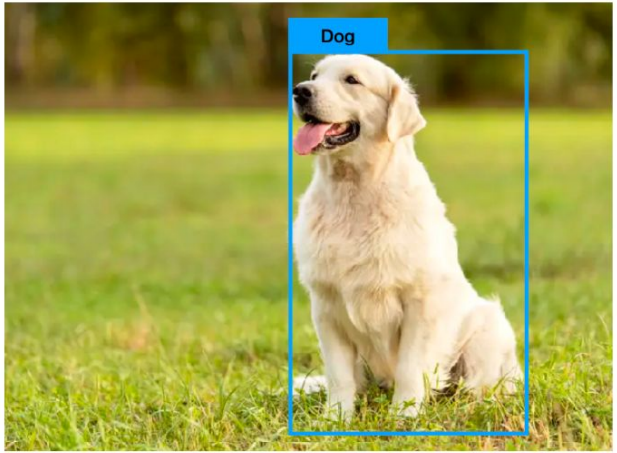
$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Precision

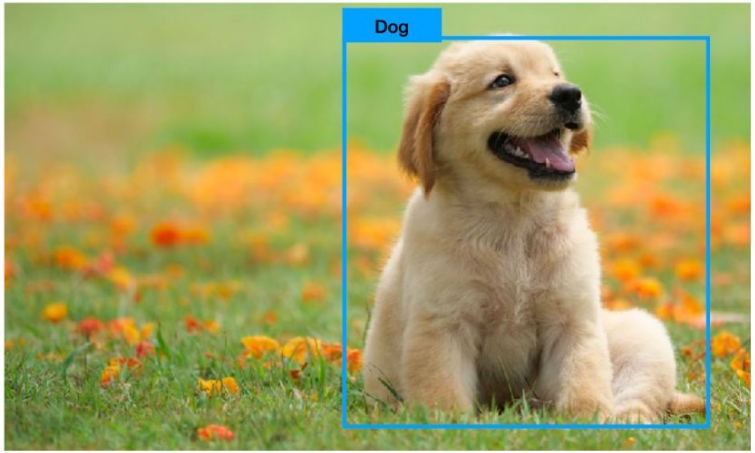


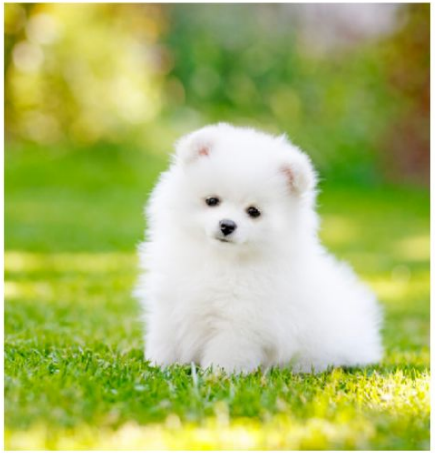


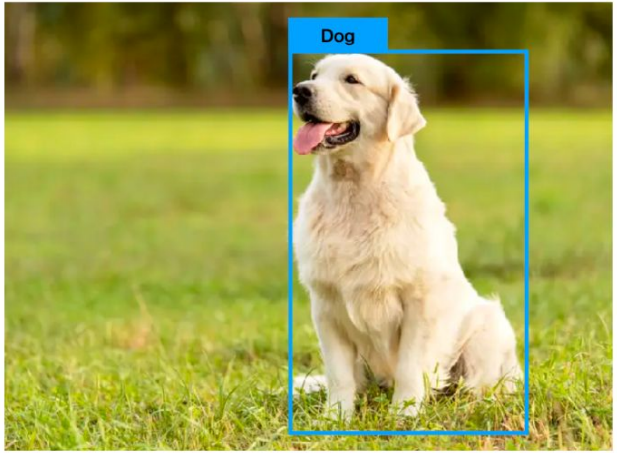


- 模型預測了狗 3 次，但只有 2 次是正確的
- Precision = 2/3

Recall

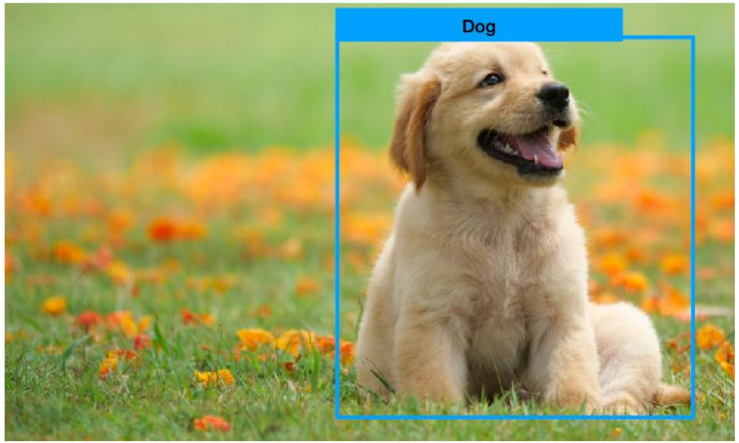


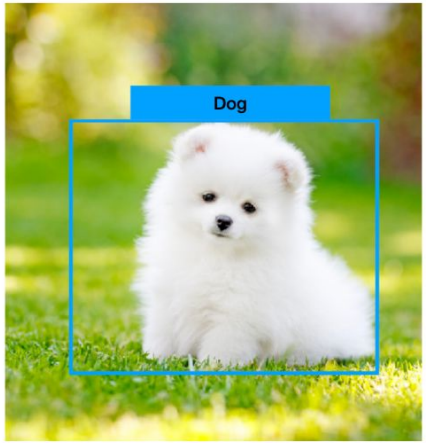


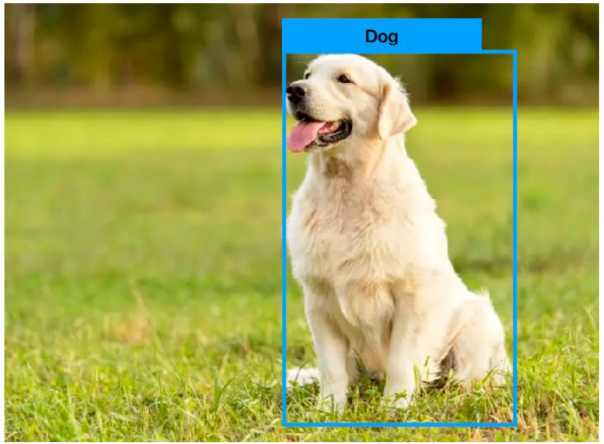


- 模型預測了 2 隻狗，但漏掉了 1 隻
- Recall = 2/3

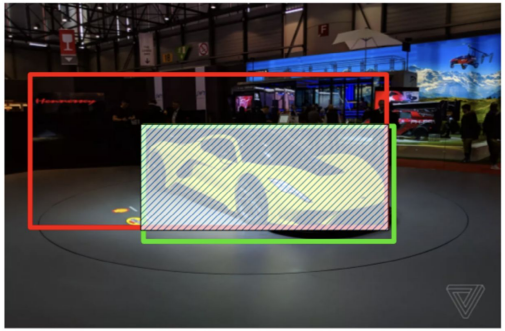
假如我們調整 IoU 的 Threshold







- IoU threshold 會決定邊界框是否出現
- 如果我們使用較嚴格的 IoU 標準，中間影像就可能不會產生邊界框
- 不同的 IoU threshold 會產生不同的 accuracy 和 recall



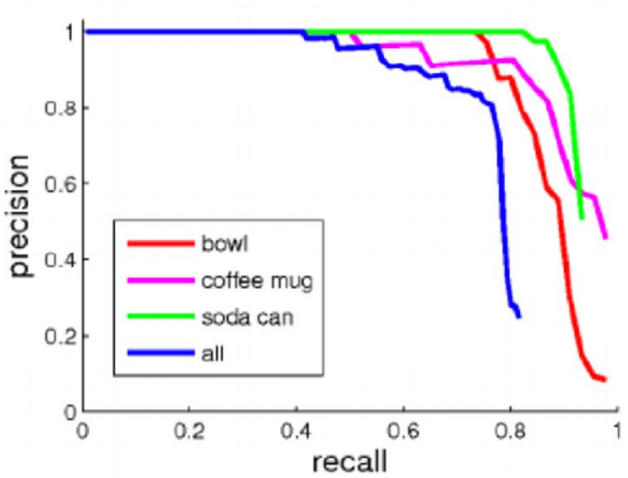
mAP

- **Average Precision (AP)** 指的是 precision-recall 曲線下的面積 (會隨著 IoU threshold 而改變)
- **Mean Average Precision or mAP** 是所有這些 AP 的平均值

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{the AP of class } k$$

$$n = \text{the number of classes}$$



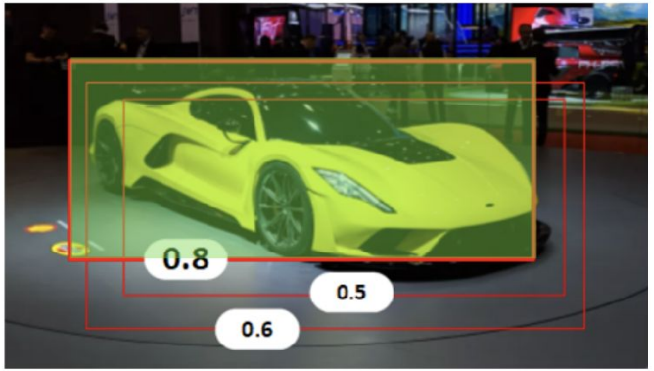
The COCO Dataset

- Object Detector Benchmark Dataset
- https://cocodataset.org/#home

Non-Maximum Suppression

A technique to clean up overlapping bounding boxes

Non-Maximum Suppression

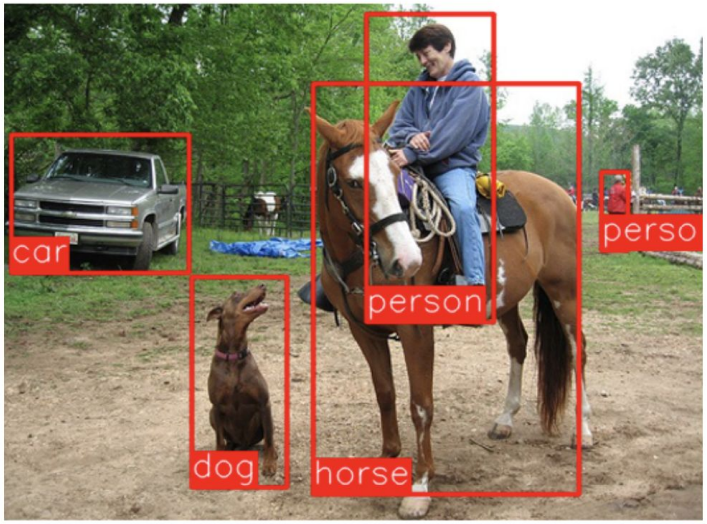


A technique to clean up overlapping bounding boxes

- 步驟 1 – 取得重疊的邊界框
- 步驟 2 – 取得每個框的 **confidence**（物件屬於某一類別的機率，例如汽車）
- 步驟 3 – 選擇 **confidence** 最高的邊界框：
  - 將其添加到 **Final Proposal List**（最初為空）
  - 將其從我們的 **Intitial Proposal List** 中刪除
- 步驟 4 – 計算 **Initial Proposal List** 中的框與 **Final Proposal List** 中的框的 IoU
- 步驟 5 – 如果 IoU 超過設定的閾值（通常為 0.5），這意味著框重疊很多，我們會捨棄它
- 步驟 6 – 我們對 **Initial Proposal List** 中的所有提案執行此操作

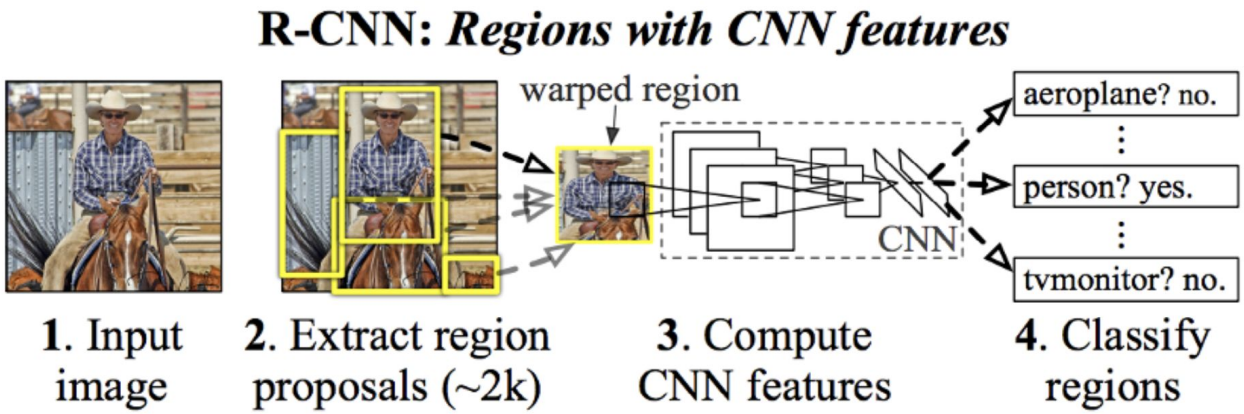
R-CNN

- Regions with CNNs
- 不是 Recurrent Neural Networks (RNNs)！
- R-CNN 是最早的基於深度學習的目標偵測器之一
- R-CNN 在 PASCAL VOC 比賽中獲得了很好的表現(類似 COCO 的 benchark 數據集)



R-CNNs 流程

- 候選區域生成：一張圖像生成約2k個候選區域(採用**Selective Search**方法)
- 特徵提取：對每個候選區域，使用深度捲積網路提取特徵(CNN)
- 類別判斷：特徵送入每一類的SVM分類器，判別是否屬於該類
- 位置精修：使用回歸器精細修正候選框位置

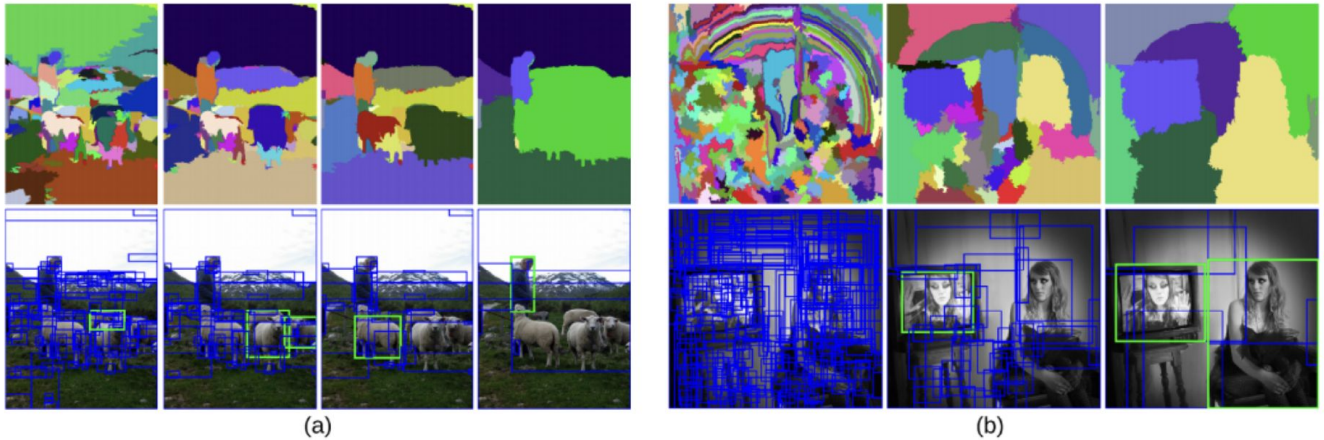


https://medium.com/lifes-a-struggle/%E5%8F%96%E5%BE%97-region-proposals-selective-search-%E5%90%AB%E7%A8%8B%E5%BC%8F%E7%A2%BC-be0aa5767901

Selective Search

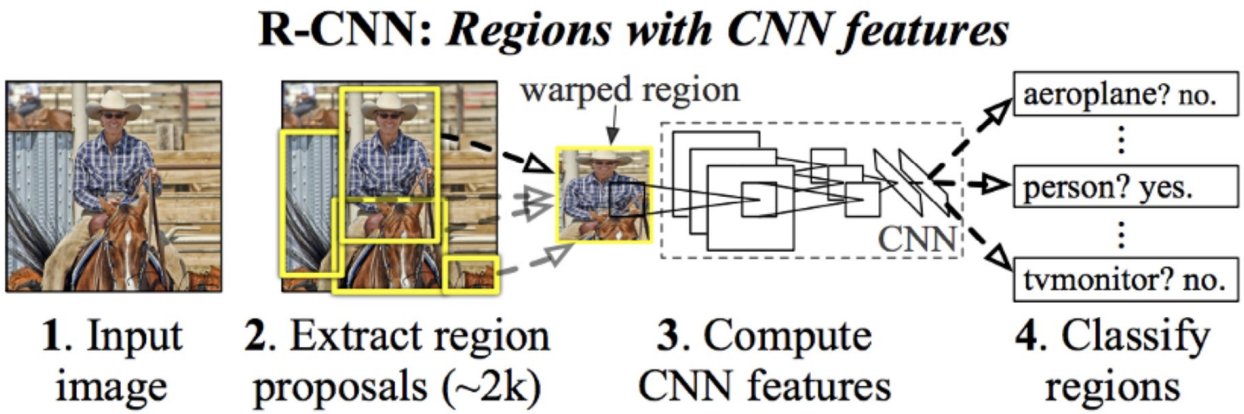
一種取得 **Region Proposals** 的方法

1. 產生初始區塊 (根據顏色、紋理、形狀等等)
2. 將各個區塊用 bounding box 包住，形成 Region proposals 的一員
3. 計算區塊之間的相似性 (**similarity measures**)，並且把相似性高的區塊合併
4. 反覆執行步驟2、3



特徵提取

- 當 Selective Search 找出這些 Region Proprosals 後，將這些提取的圖像傳遞到我們的 CNN（例如：在 ImageNet 上預訓練的模型）進行分類 (微調)。
- 我們不直接使用 CNN 進行分類（儘管我們可以），我們使用 SVM 對 CNN 提取的特徵進行分類。

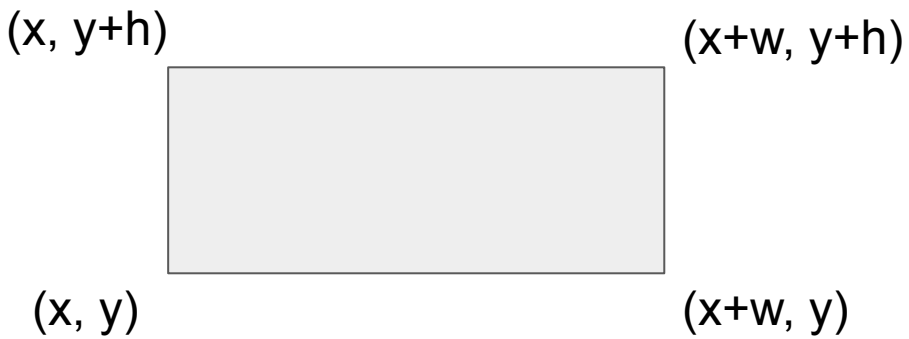


類別判斷

- 將 CNN 提取出來的特徵作為 SVM 的特徵，來進行預測
- 為什麼不直接使用CNN的分類結果？
  - 作者發現使用CNN直接分類結果並不注重於精確定位
  - 可能原因：因為CNN識別能力非常強大，所以不是那麼精確的定位也可以得到比較好的結果，導致不特別注重精確定位

位置精修 (Extra)

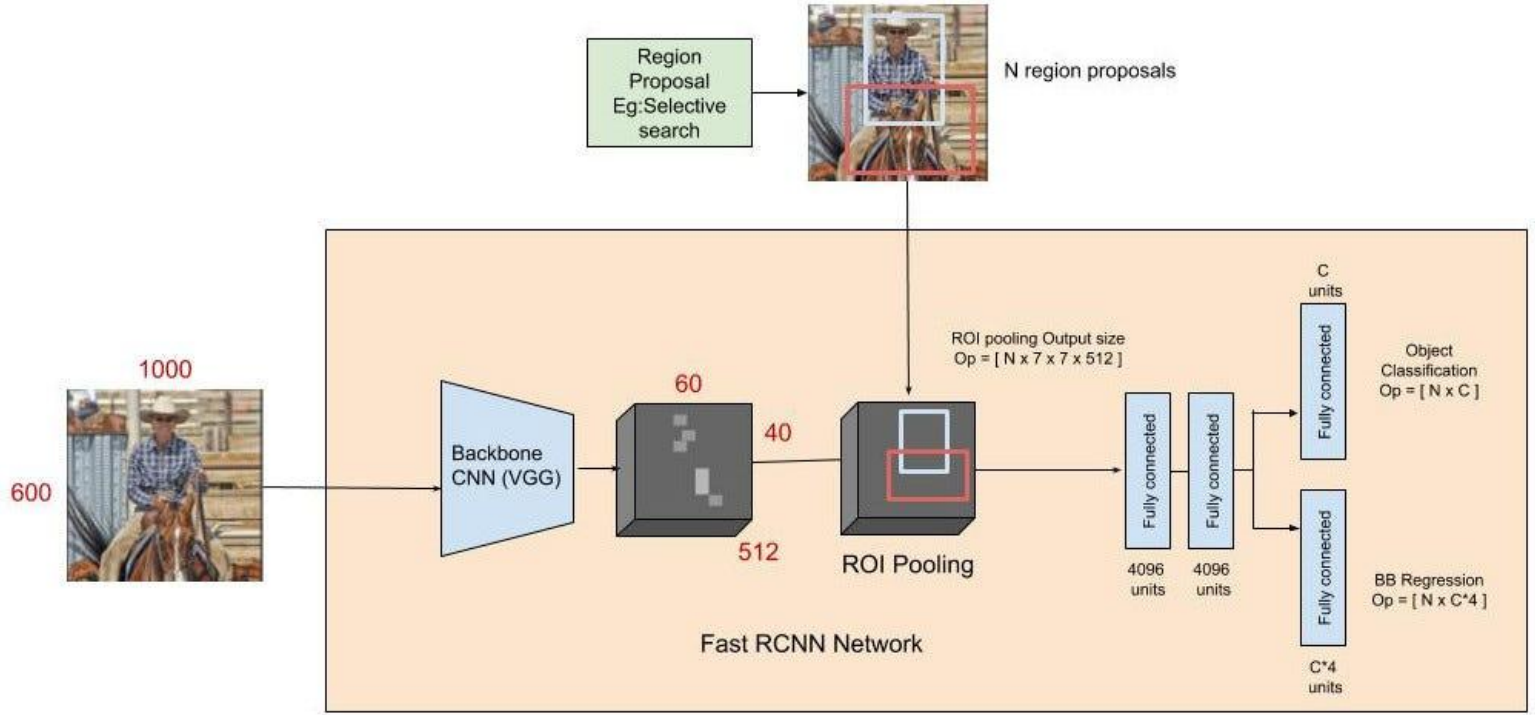
- 在對 Region Proposal 進行分類後，我們使用簡單的 Linear Regression 來產生更緊密的邊界框
- 預測 (x, y, w, h)



Fast R-CNNs

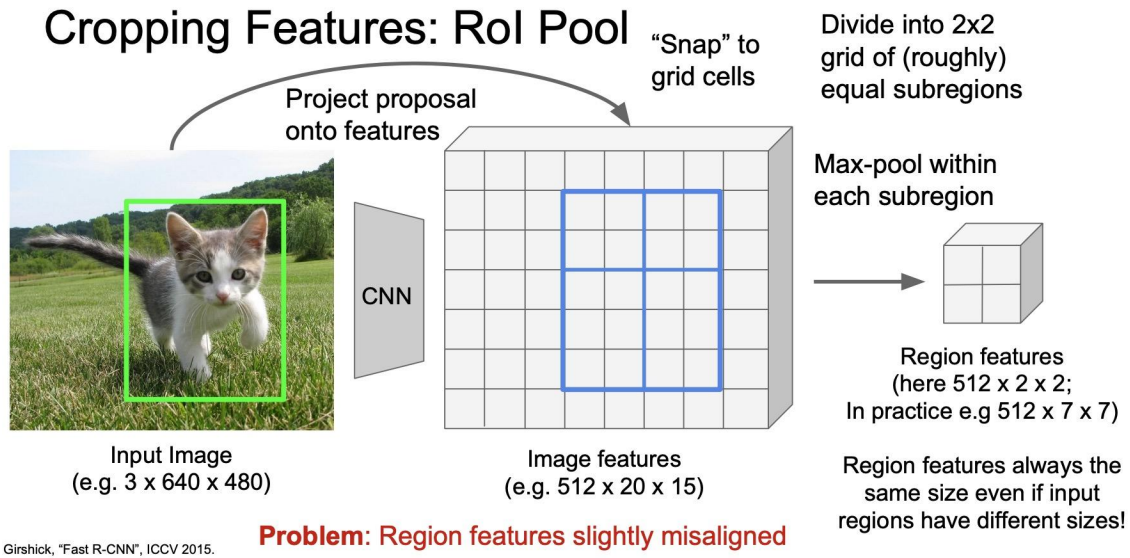
- R-CNN 的速度非常慢，因為每個邊界框都必須執行 CNN
- 訓練分太多階段
- Fast R-CNNs:
  - 拿掉SVM結構，改以softmax作為Classification的NN
  - 一樣是用selective search提出Region Proposals，或叫做ROI (Region of Interest)，但是改為先將整張影像通過conv.得到feature map後，將ROI映射在map上來取真正要用的ROI，因此一張影像只須過一次Convolution
  - 利用ROI pooling layer來統一ROI送進Fully-connected layer (FC)層前的維度

Fast R-CNNs 結構

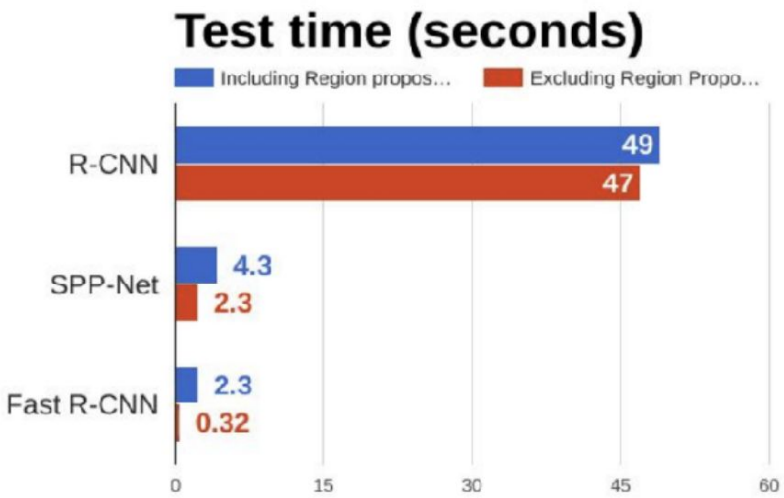


ROI Poolying Layer

- ROI可能是很多不同的長寬構成的框框，而VGGnet的FC層要以7x7的feature map為輸入
- 將所有 ROI 轉為 7x7 大小



Fast R-CNN Improvements



Faster R-CNNs

- 捨棄 Selective Search - 為了迎來更快的偵測速度，在Proposals上的處理也納入整個NN之中，一起用convolution來解決
- Region Proposals Network (RPN)
- Anchor Box

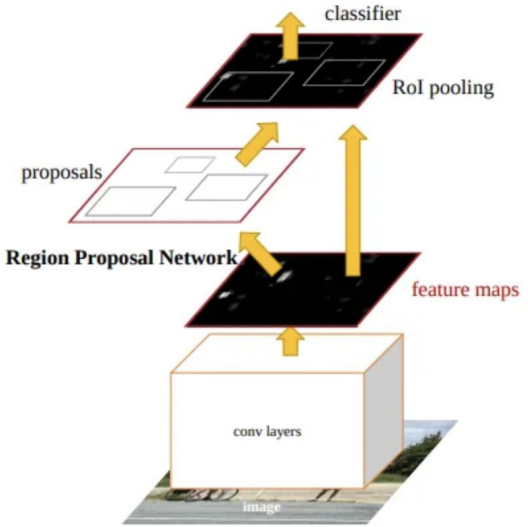
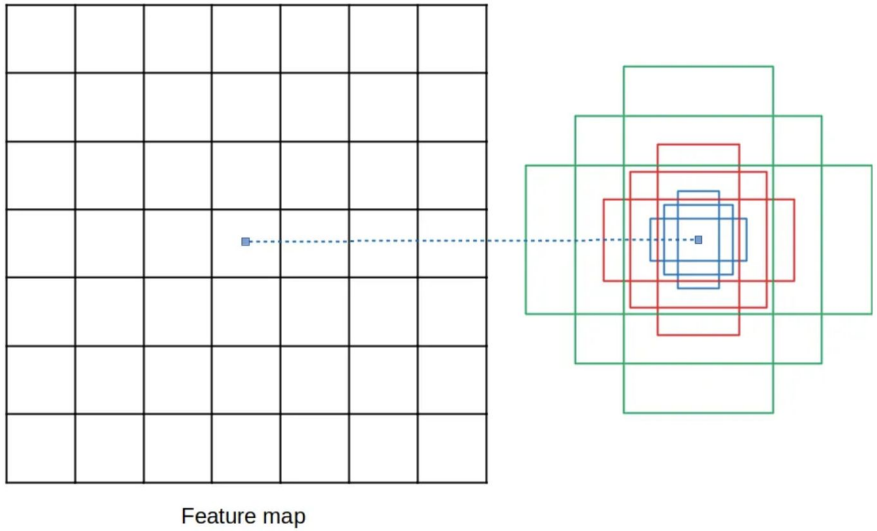


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Anchor

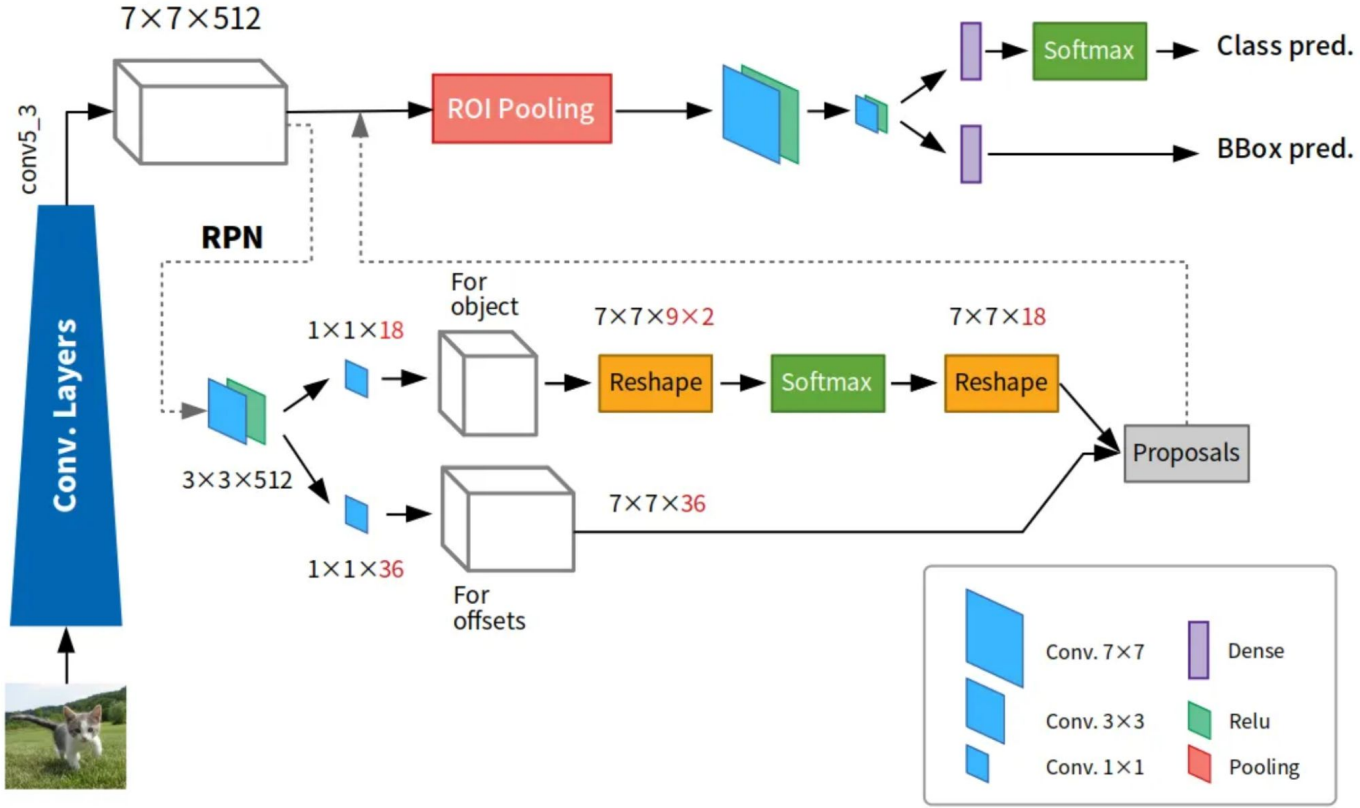
- Anchor 是指由我們訂出不同尺度、比例的 proposal 的邊界匡的雛形
- 原文採取3種尺度配上3種比例，得到k個anchor，也就是k=3×3=9
- 假設 backbone network 是VGG16 (feature map = 7×7×512)。會在7×7的格子上都佈上這9個anchor。
- 因此，在一個Feature map上初步產生的anchor就有7×7×9=441個



- 三種顏色代表3種尺度，各配上三種長寬比，而得到9種anchor

https://ivan-eng-murmur.medium.com/object-detection-s3-faster-rcnn-%E7%B0%A1%E4%BB%8B-5f37b13ccdd2

RPN



Single Shot Detectors (SSDs)

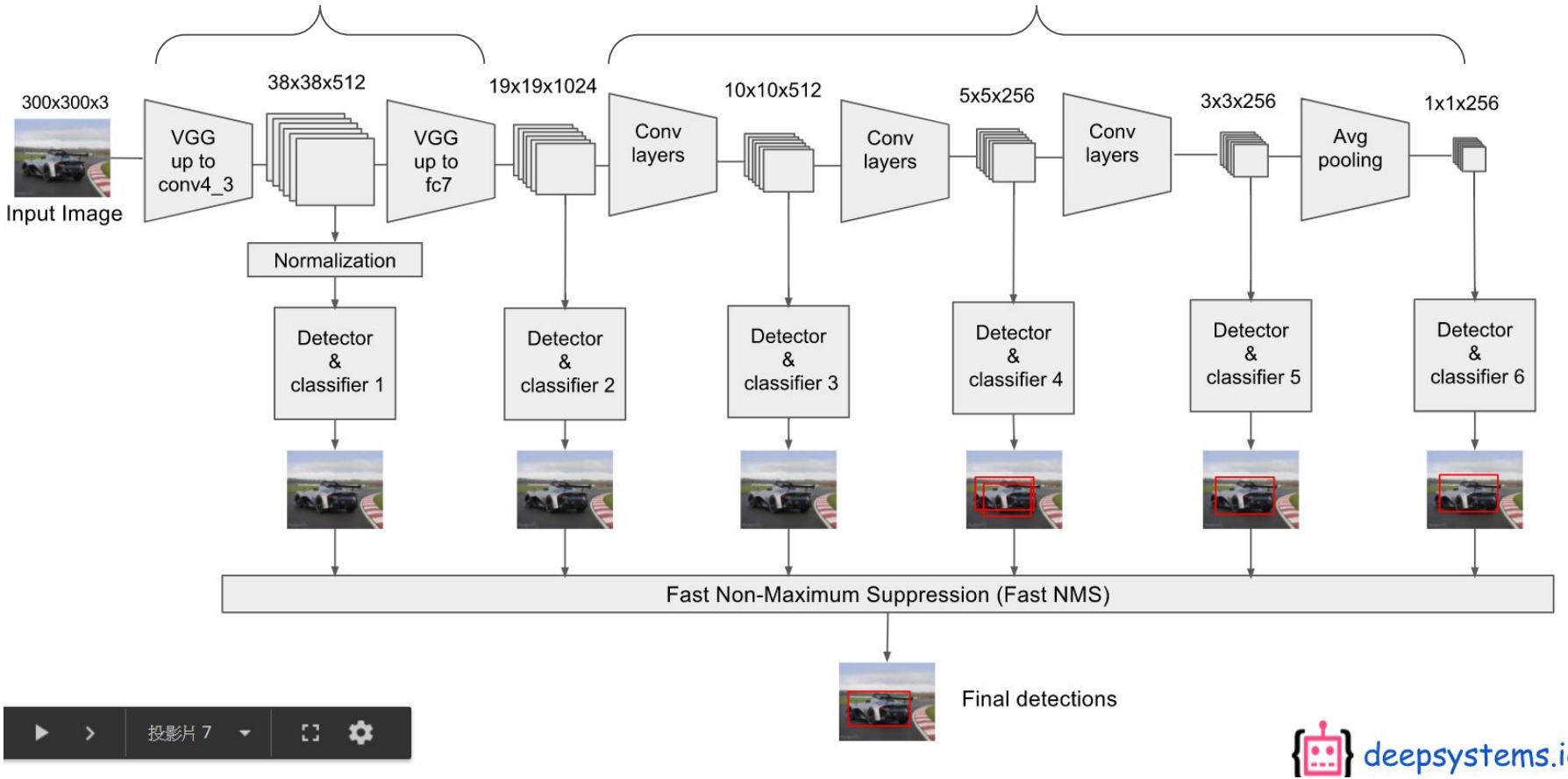
- R-CNN家族可以達到很好的表現
- 然而，它們的弱點在於，它們的推理時間性能仍然不是最佳的，在強大的硬體上通常最多只能以 7 fps 的速度運行。
- SSD 希望透過消除對 Region Proposal Network 的需求來提高速度。所以叫”Single Shot“

System	VOC2007 test *mAP*	FPS (Titan X)	Number of Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	~6000	~1000 x 600
YOLO (customized)	63.4	45	98	448 x 448
SSD300* (VGG16)	77.2	46	8732	300 x 300
SSD512* (VGG16)	**79.8**	19	24564	512 x 512

SSDs Structure

SSD 由兩個主要部分組成:

- Feature Map Extracter(已發表的論文中使用了 VGG16, 但 ResNet 或 DenseNet 可能提供更好的結果))
- Convolution Filter for Object Detection



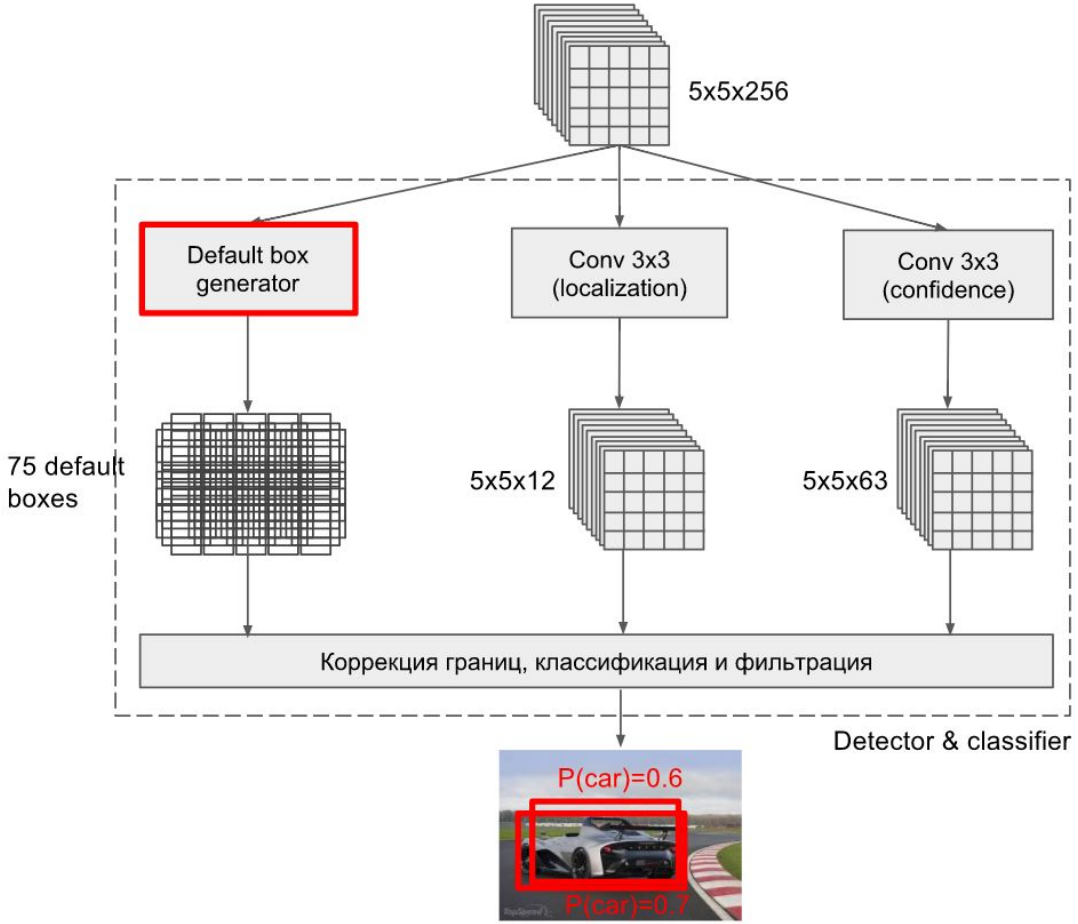
投影片 7

deepsystems.io

Details in the Detectors

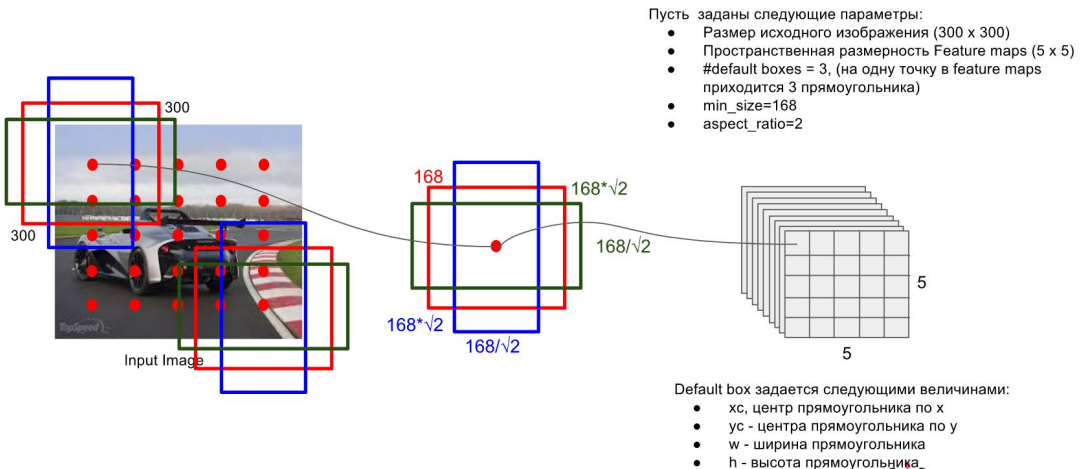
Let’s look at the 4th detector

- Input = 5x5x256 feature map
- Three outputs:
  - Box proposal (偵測框): 5x5x75
  - Localization (座標): 5x5x12
  - Confidence (類別): 5x5x63



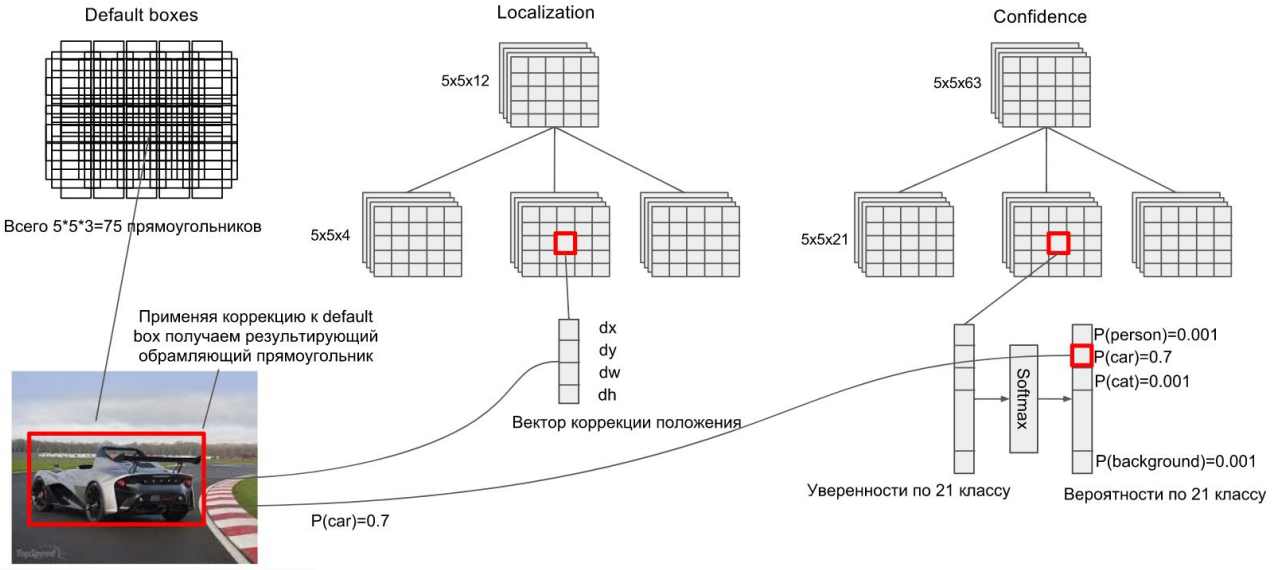
Box proposal

- 輸出為 5x5x75 的特徵圖
- 這個75是需特殊設置過的，必須符合(C+4)的倍數，C代表分類的總數，4代表預測box的四個值，在這裡C為21，因此 (21+4) * 3 = 75
- 這個範例中n=3，代表每一個小方塊的周圍產生三個defaut box(藍紅綠框框)來做預測

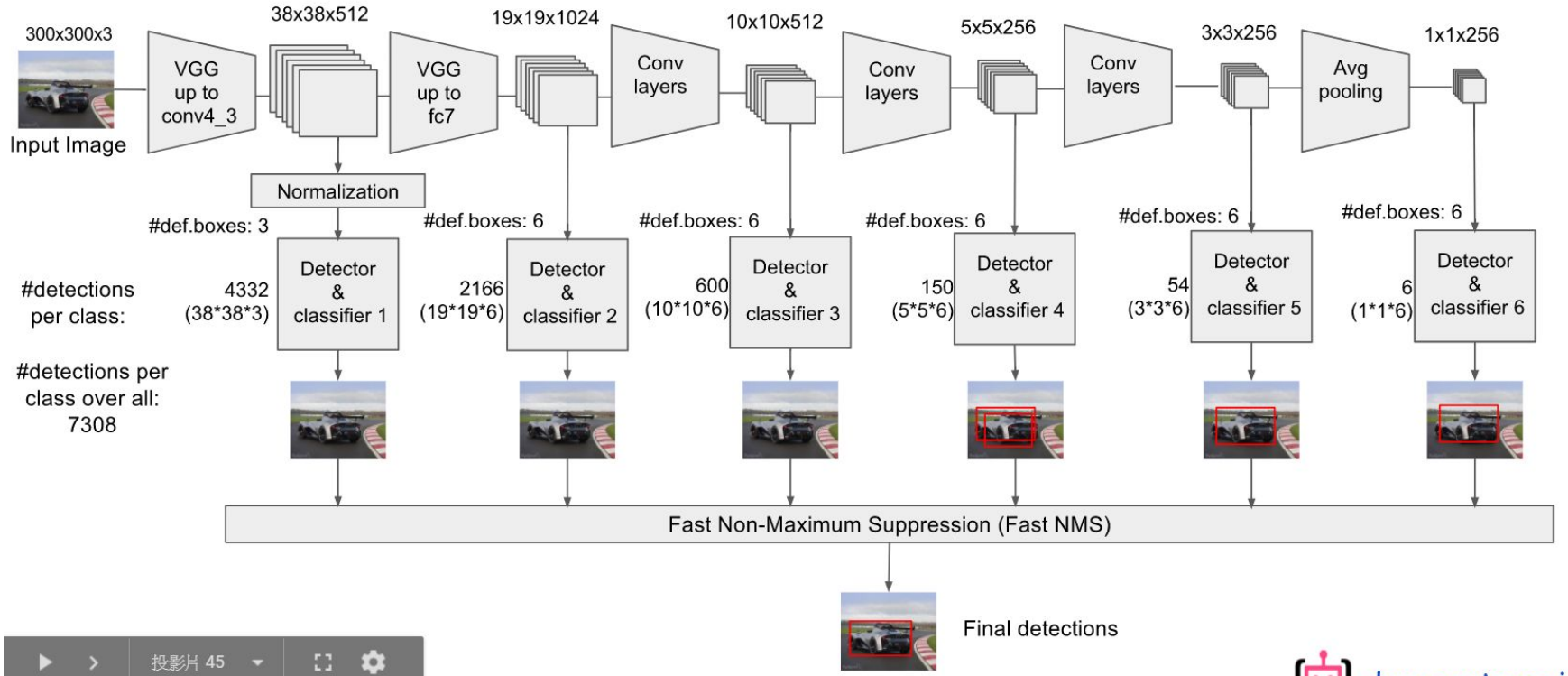


Localization and Confidence

- 因此，可以再把12、63分成三份，如中間的紅色小框框所示，深度為4，每一層各存不同的資訊，分別是中心點的座標x,y，以及其寬和高w,h，右邊的小紅框框也是同樣的道理，分別是20種類別再加上一個背景類別



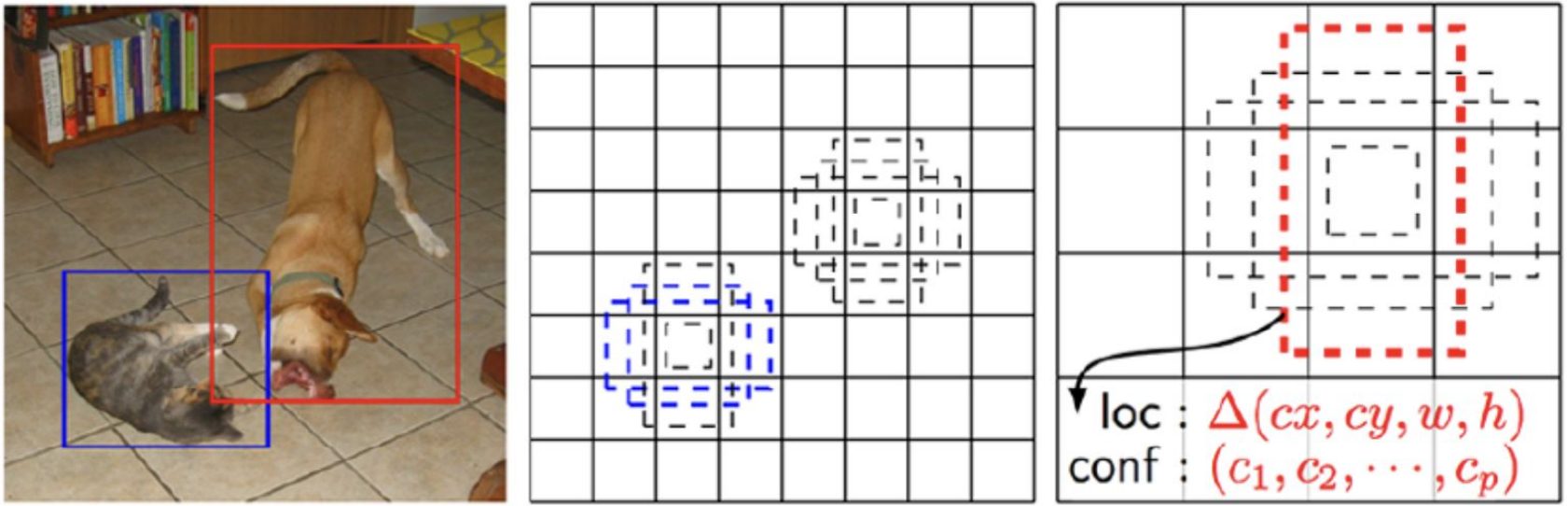
- 每一層的default box皆不同，將所有box集合起來就可以得到7308個default box



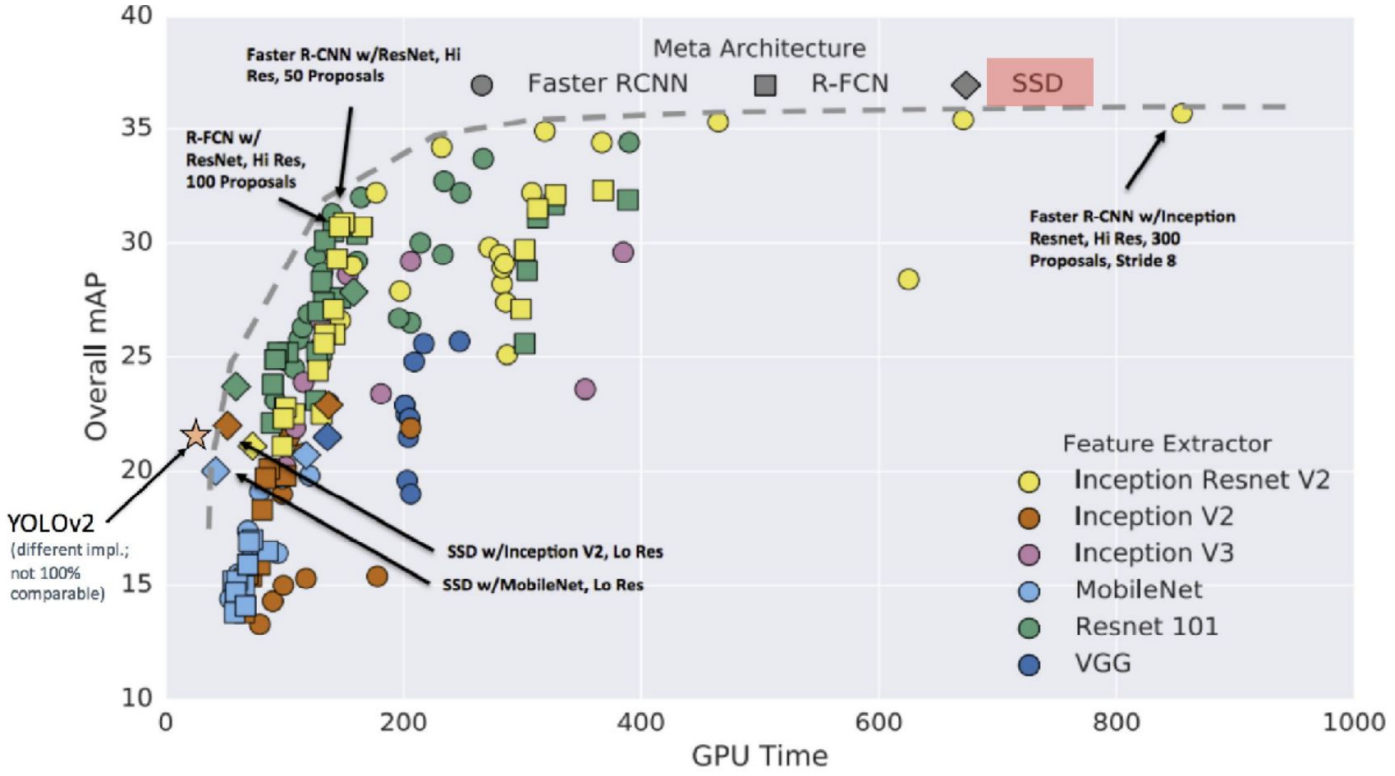
deepsystems.io

用不同大小的特徵圖做預測

- 較少的單元格可以檢測到較大的物體(例如狗和最右邊的圖)，而較多的單元格可以檢測較小的物體(例如貓)。
- Default box隨著取出的 feature map 大小有著不同的縮放比例 (Aspect Ratio)



SSD Performance

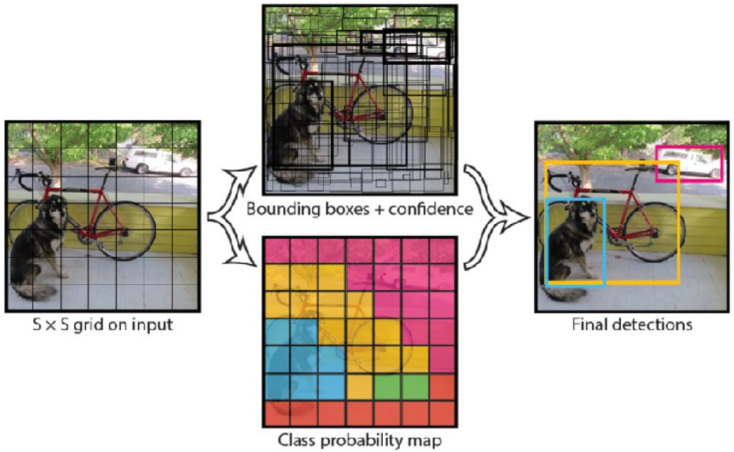


Take Aways

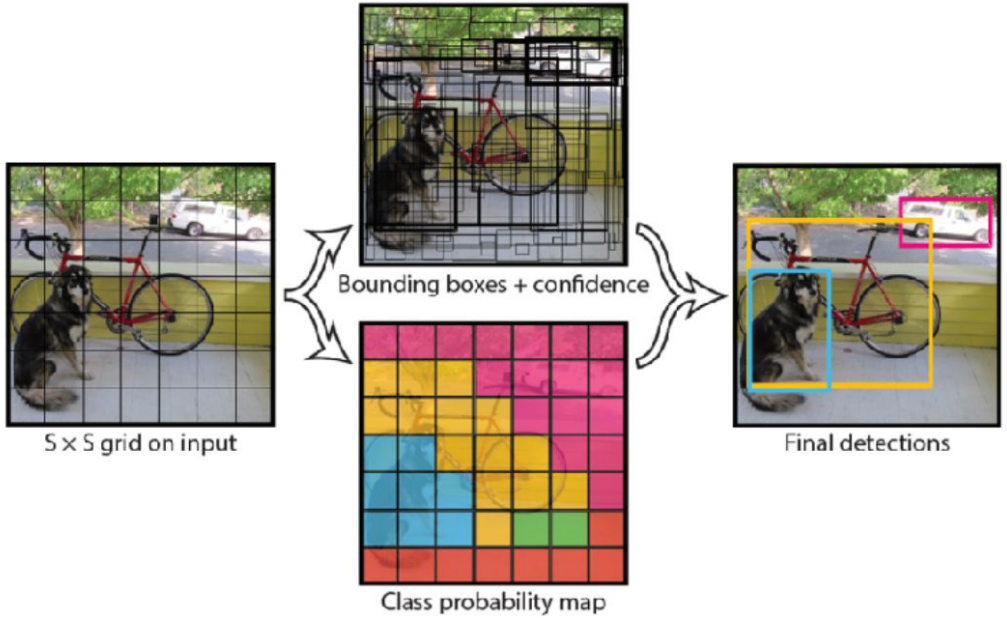
- SSDs are faster than Faster R-CNN but less accurate in detecting small objects.
- Accuracy increases if we increase the number of default boxes as well as better designed boxes
- Multi-scale feature maps improve detection at varying scales.
- For training, requires that ground truth data is assigned to specific outputs in the fixed set of detector outputs
- Slower but more accurate than YOLO
- Faster but less accurate than Faster R-CNN

YOLO - You Only Look Once

- YOLO v1 在 2015 年被提出 (和 SSD 一樣)
- YOLO is a single stage detector
- R-CNNs 家族表現很好，但執行速度是一個大問題
- SSDs 雖然在這方面改善與多，但它並沒有 R-CNNs 準確
- YOLO 希望找到速度和準確率之間的平衡

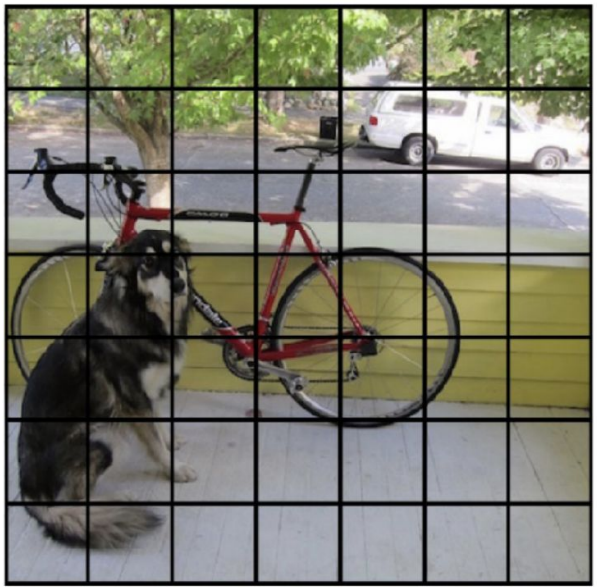


How Does YOLO Work?



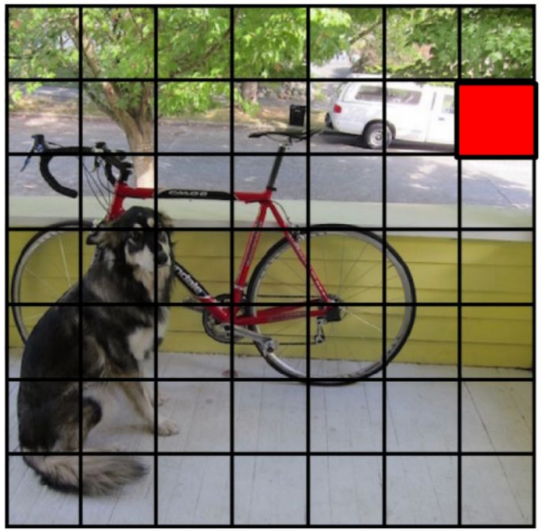
Split Image into Grid

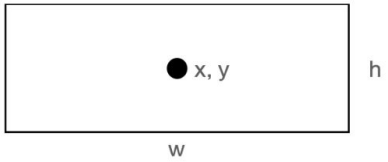
- Firstly, we split the image into an S x S grid, typically a 7 x 7.



Each Cell Gives B Boxes

- Next, each cell predicts B bounding boxes (x, y, w, h) and the confidences of each box having an object i.e the Probability that box has an object ~ P(Object)

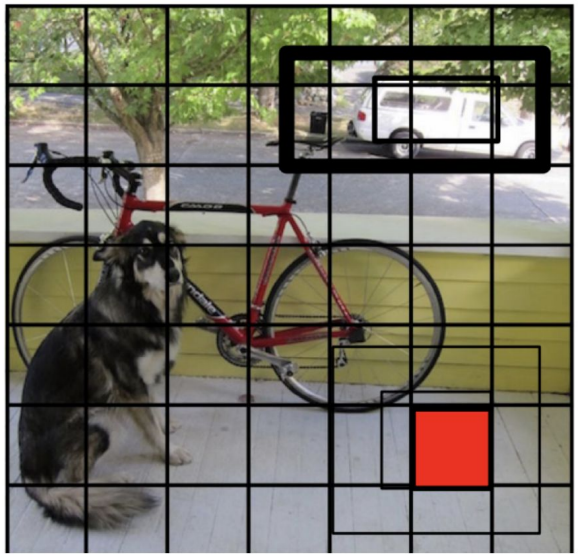




B = 2

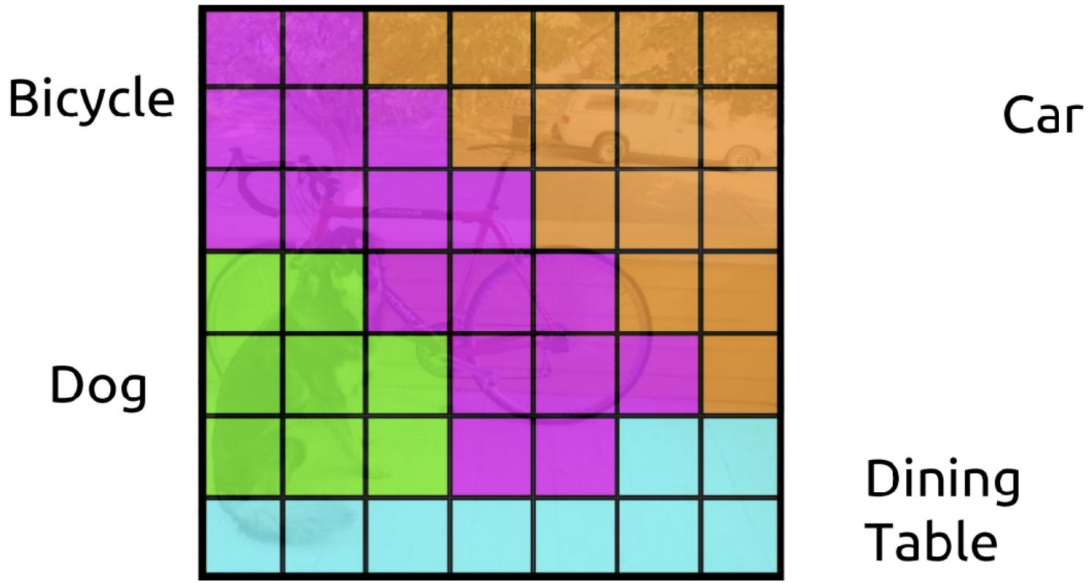
Generate All Bounding Boxes

- So now for each cell we have B boxes and the associated probabilities of each box having an object



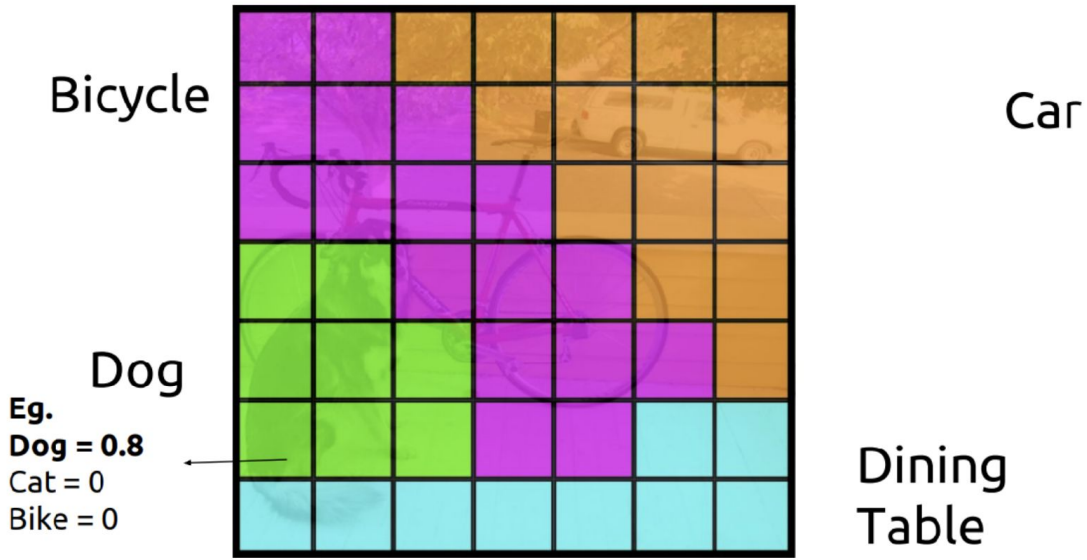
Class Probability per Cell

- Each cell predicts a class probability.



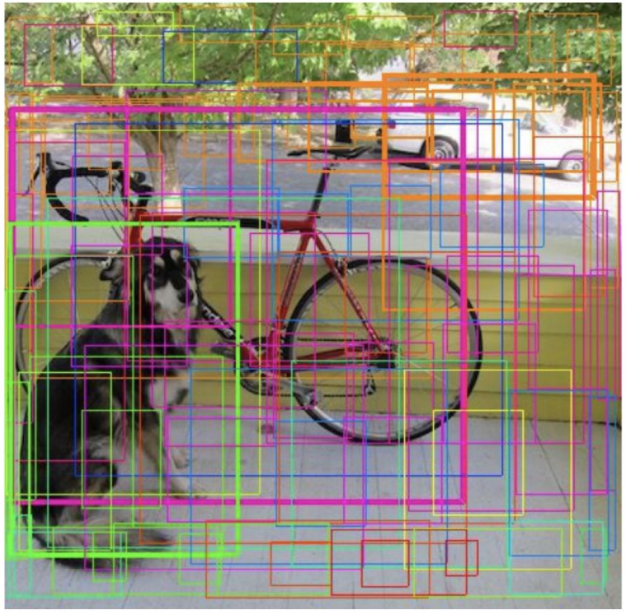
Class Probability per Cell

- Each cell provides the probability of the object class e.g. P(Dog | Object)



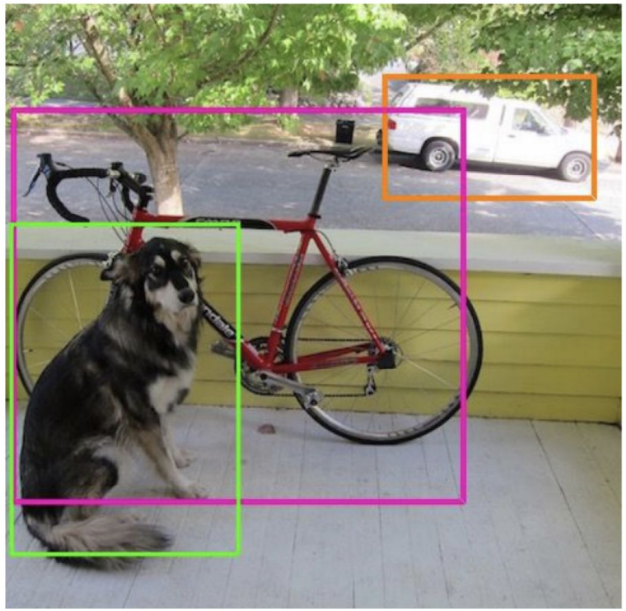
Combining Box and Class Predictions

- P(class) = P(class|object) * P(Object)
- Only predictions with P(Object) = 1 will be kept



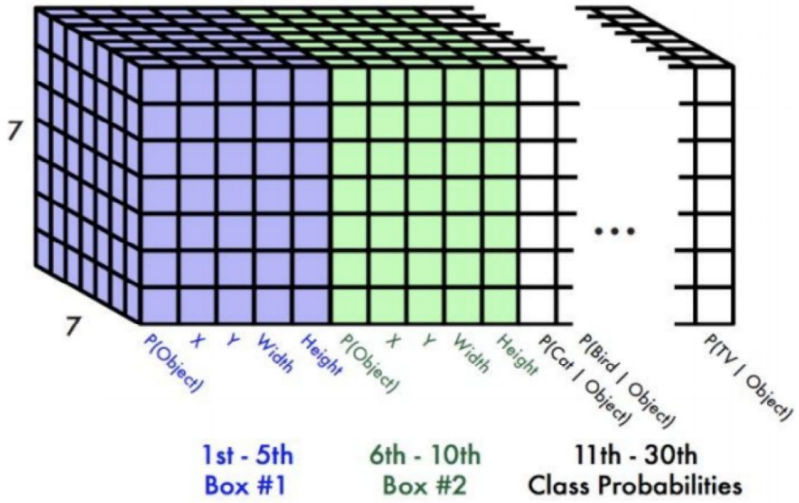
Apply Non Maximum Suppression (NMS)

- We threshold detections using NMS



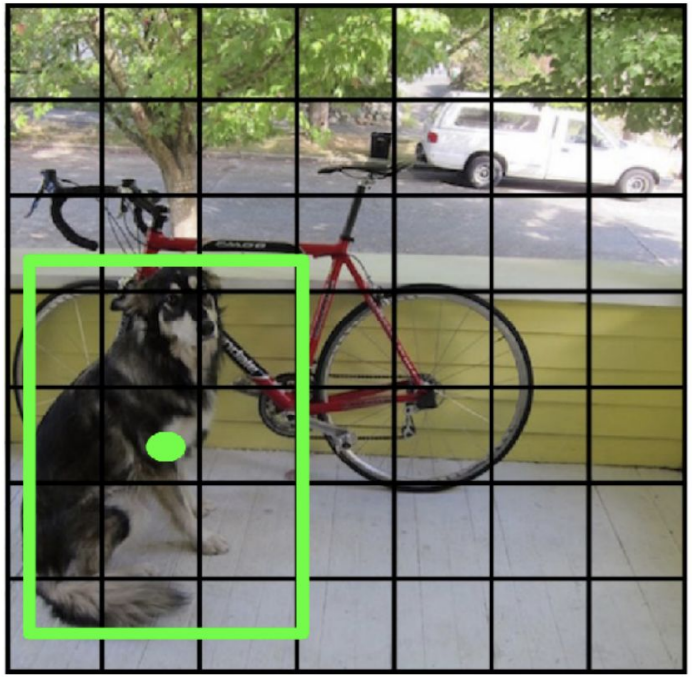
Output Predictions

- Each cell provides two bounding boxes
- For each bounding box we get:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- A number of class probabilities
- The output forms the dimensions
- S * S * (B * 5 + C)

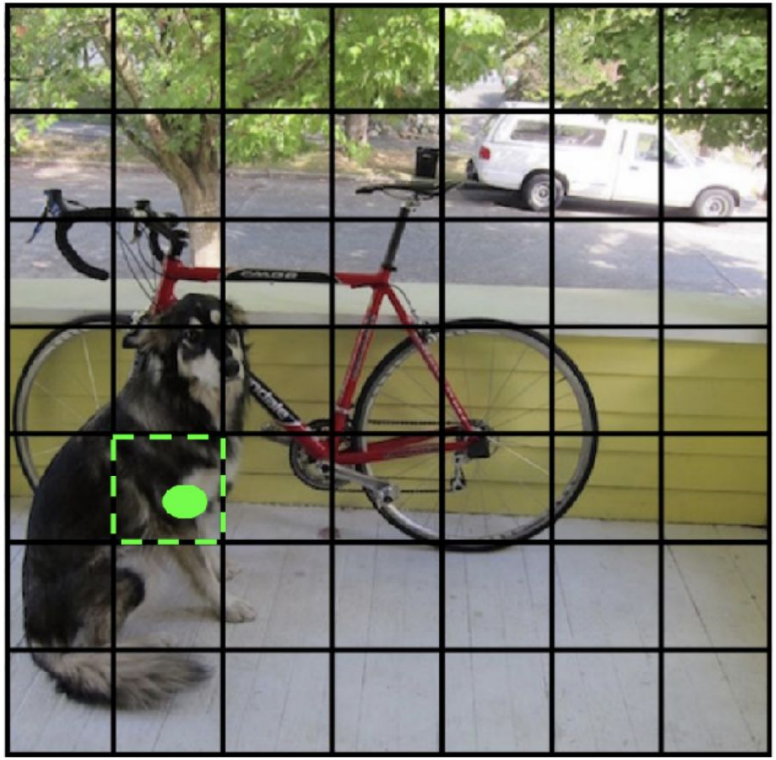


Training YOLO

- During training we have the ground truth labels that we use as our ‘targets’ for our model

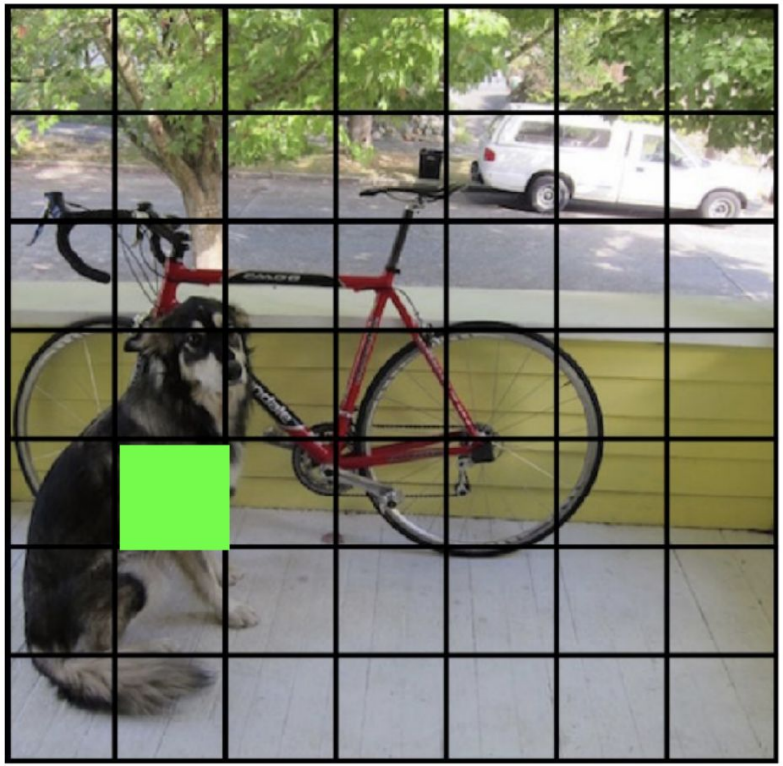


找到物件對應的網格細胞



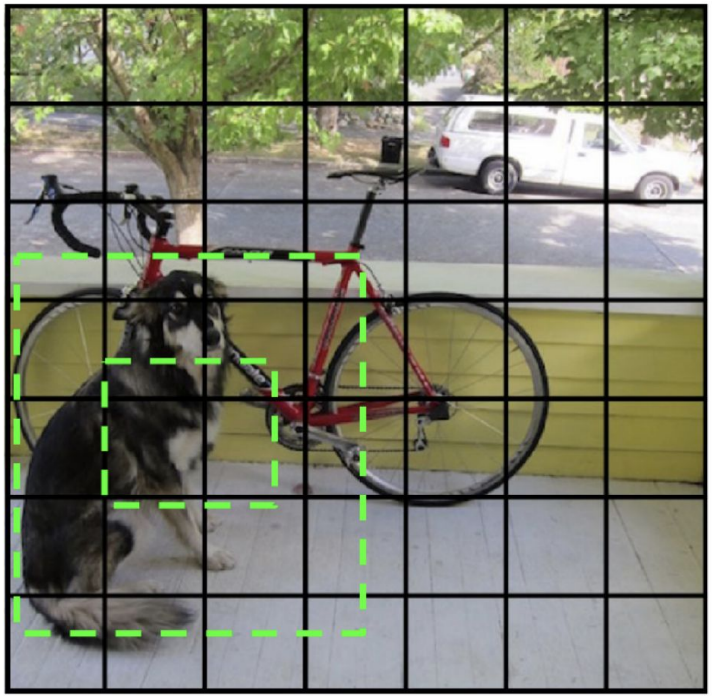
細胞的類別預測

Dog = 1
Cat = 0
Bike = 0
...



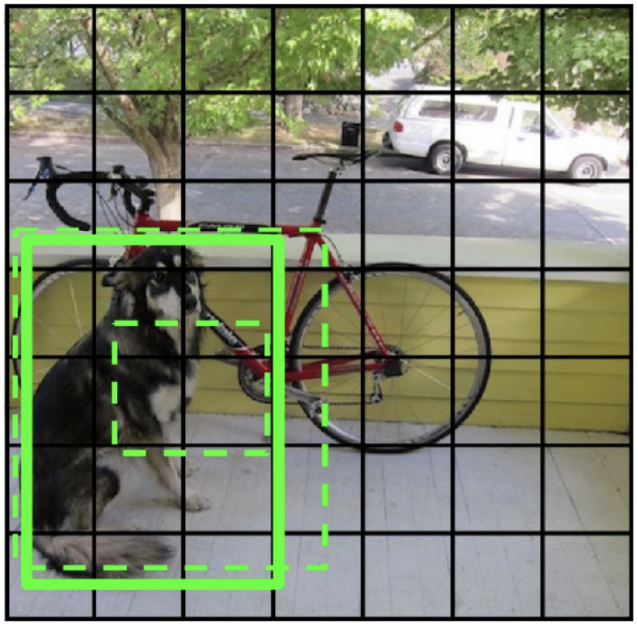
預測偵測框

- We then get the 2 bounding box predictions for that cell



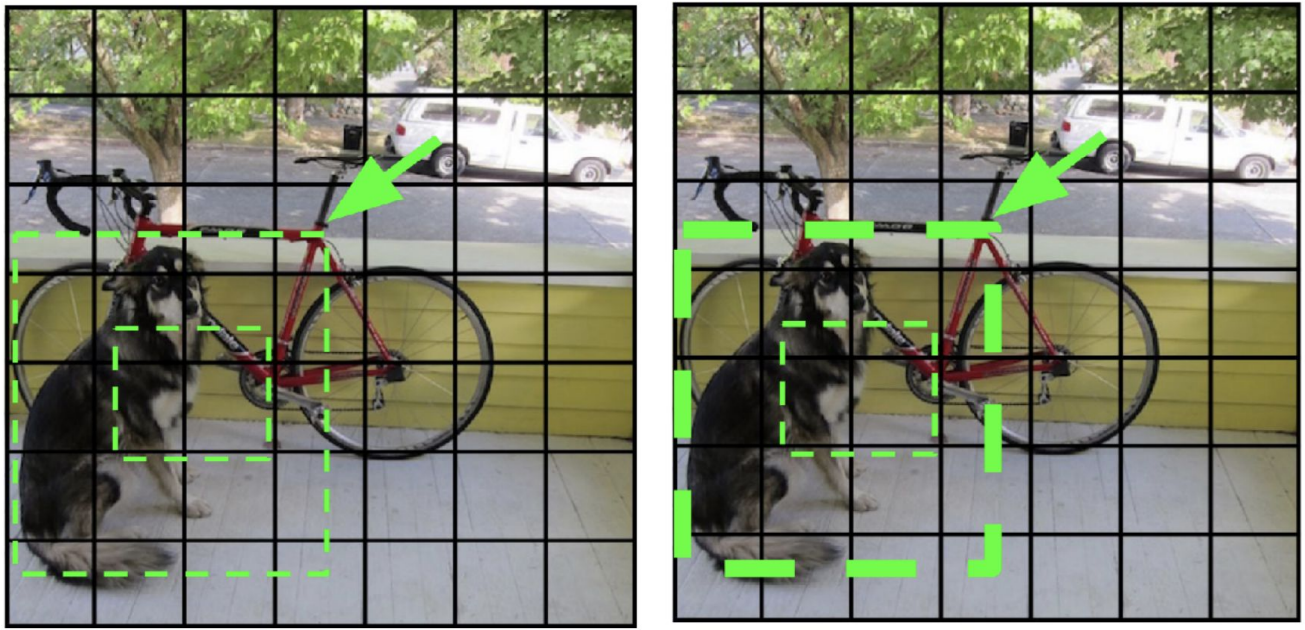
預測偵測框

- The best box is then adjusted (i.e. confidence increase) while the less accurate box is decreased in confidence.



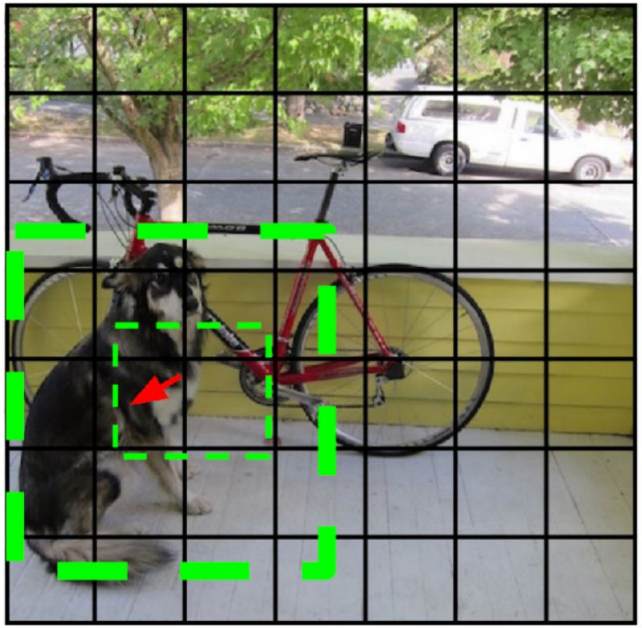
預測偵測框

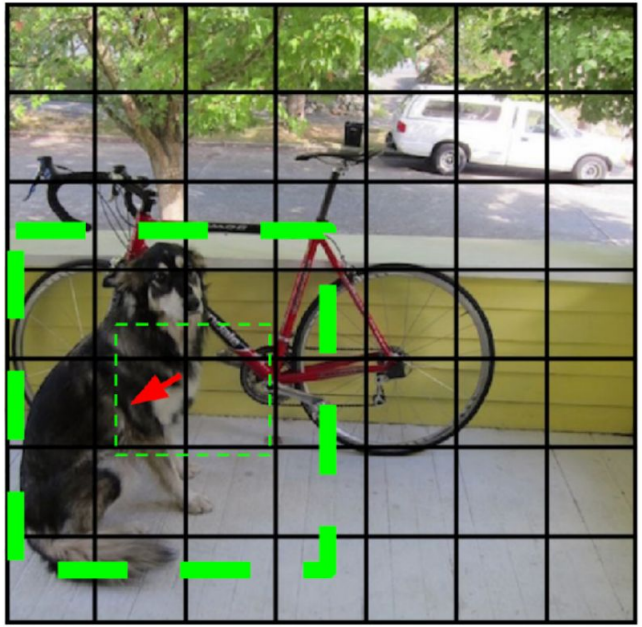
- The best box is then adjusted (i.e. confidence increase) while the less accurate box is decreased in confidence.



預測偵測框

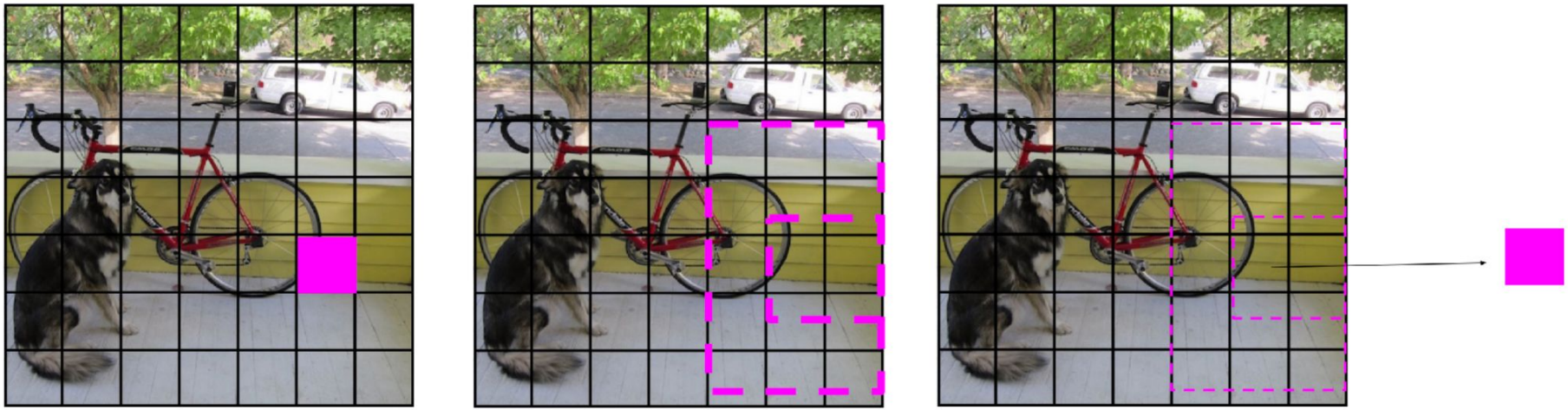
- The best box is then adjusted (i.e. confidence increase) while the less accurate box is decreased in confidence.





預測偵測框

- The boxes where no ground truth boxes lie are decreased in confidence
- We don’t adjust the class probabilities or coordinates of these boxes



YOLO’s Loss Functions

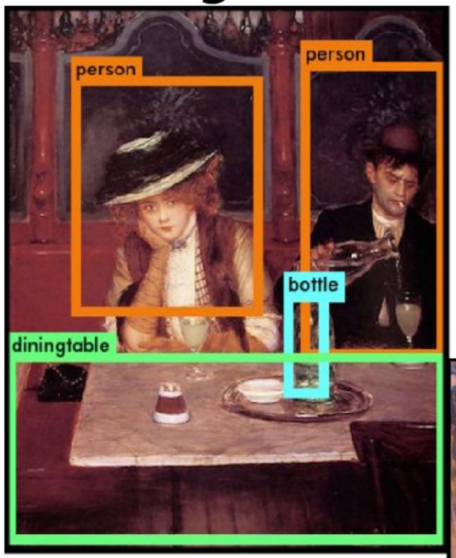
- Classification Loss - if an object is detected, it is the squared error loss of the class conditional probabilities for each class
- Localisation Loss - measures error from the predicted boundary box to the ground truth
- Confidence Loss - confidence that an object lies in the box
- 背景網格不考慮 classification loss 和 localisation loss

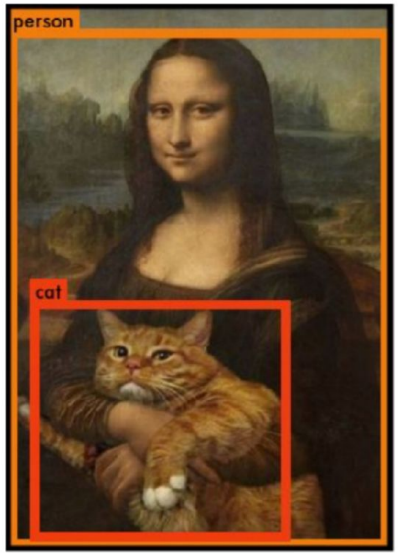
YOLO’s Performance

	Pascal 2007 mAP	**Speed**	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img

YOLO's Performance

YOLO generalizes well to new domains (like art)





Ref: https://pjreddie.com/publications/

YOLO Take Aways

- YOLO is fast!
- YOLOv4, v5 and X are perhaps the best in accuracy (mAP) in 2021
- Provides End2end training
- Low background error
- Performance (mAP) was not as good as slower Fast R-CNNs (Improved)
- More localisation errors (Improved)