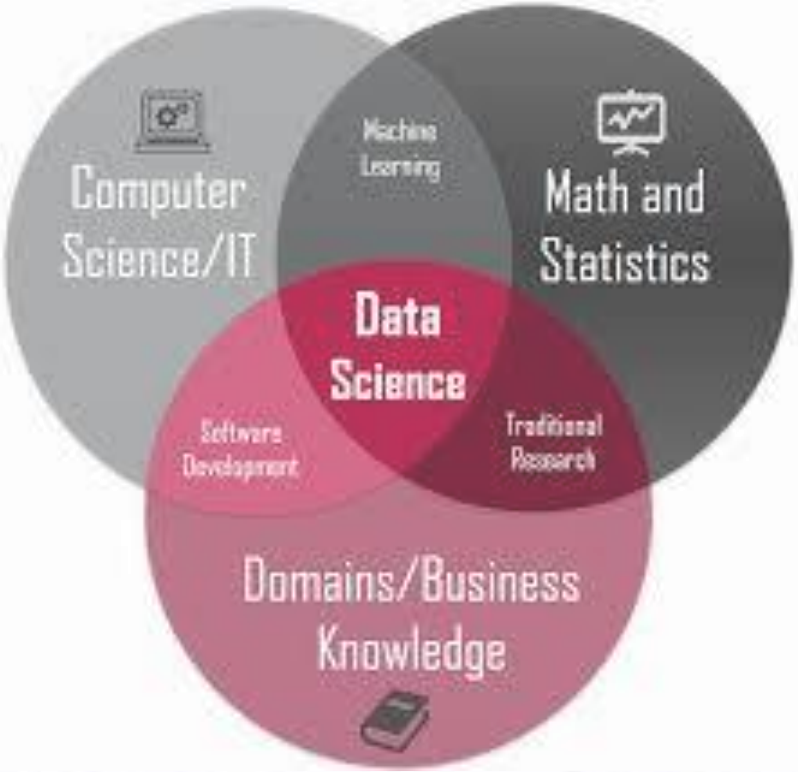
AI & DS



A01-影像辨識_YoLo4

2023.11_V1.1

單元大綱

- 認識YOLO演算法
- Google Colab雲端訓練客製化 YOLOv4物件辨識

Part 1 認識YOLO演算法

You Only Look Once，YOLO

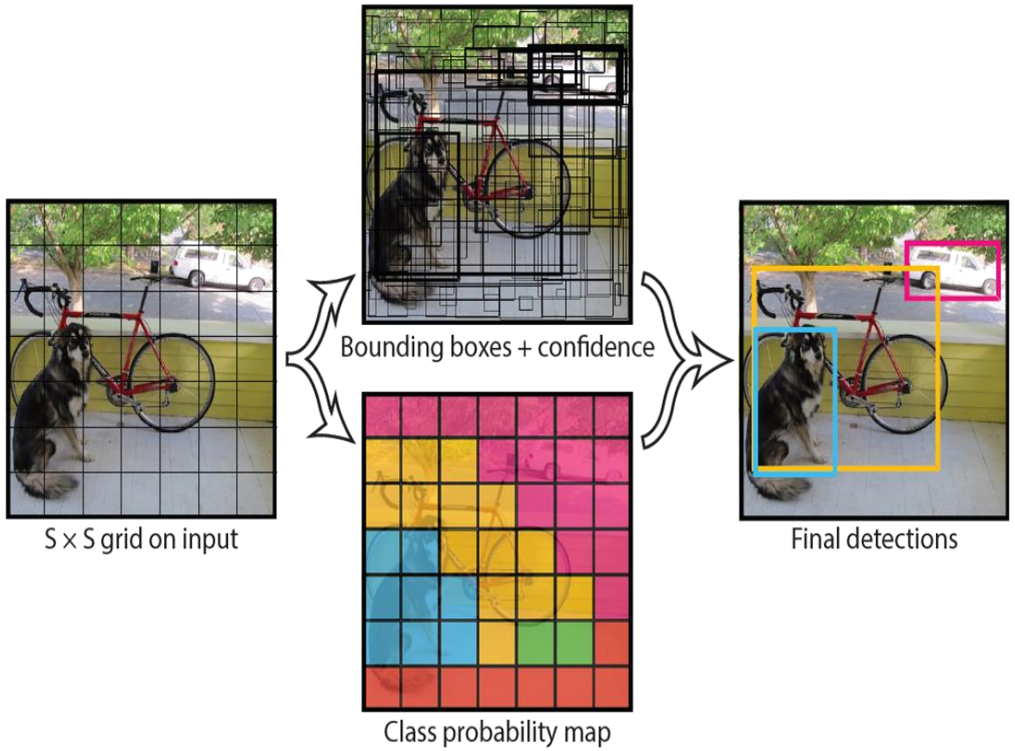
- 「只要讓我看一眼，我就知道這是什麼！（You Only Look Once，YOLO）」
- YOLO，是目前當紅的 AI 物件偵測演算法。中研院資訊科學研究所所長廖弘源及博士後研究員王建堯，與俄羅斯學者博科夫斯基（Alexey Bochkovskiy）共同研發最新的 YOLO 第四版（簡稱為 YOLO v4），一舉成為當前全世界最快、最高精準度的物件偵測系統。
- YOLO 是一個 one-stage 的 object detection 演算法，將整個影像輸入只需要一個 CNN 就可以一次性的預測多個目標物位置及類別，這種 end-to-end 的算法可以提升辨識速度，能夠實現 real-time 偵測並維持高準確度。
- **Darknet:**

  本來是由 YOLO 作者自己寫的 deep learning framework，不過原作者因為一些因素不再繼續維護， 改由俄羅斯的 AlexeyAB 接續。

[文章引用]
1. https://pansci.asia/archives/194503
2. https://pse.is/4z5jxl

YOLO 運作原理

- YOLO 的作法就是將輸入的影像切割成 SxS 的網格 (grid)，若被偵測物體的中心落入某個網格內，這個網格就要負責去偵測該物體。而每個網格要負責預測B個 bounding boxes (bndBox，在 YOLO 的設計中，YOLOv1: B=2, YOLOv2: B=5, YOLOv3: B=3) 和屬於各別類別的機率 (假設有C個類別)，其中對每個 bndBox 的預測會輸出5個預測值: x, y, w, h 以及 confidence。
  - x, y 代表該bndBox的中心座標與圖片寬高的比值，是bndBox歸一化後的中心座標
  - w, h代表該bndBox的寬高與輸入圖像寬高的比值，是bndBox歸一化後的寬高座標
  - confidence代表bndBox與Ground Truth的IOU值



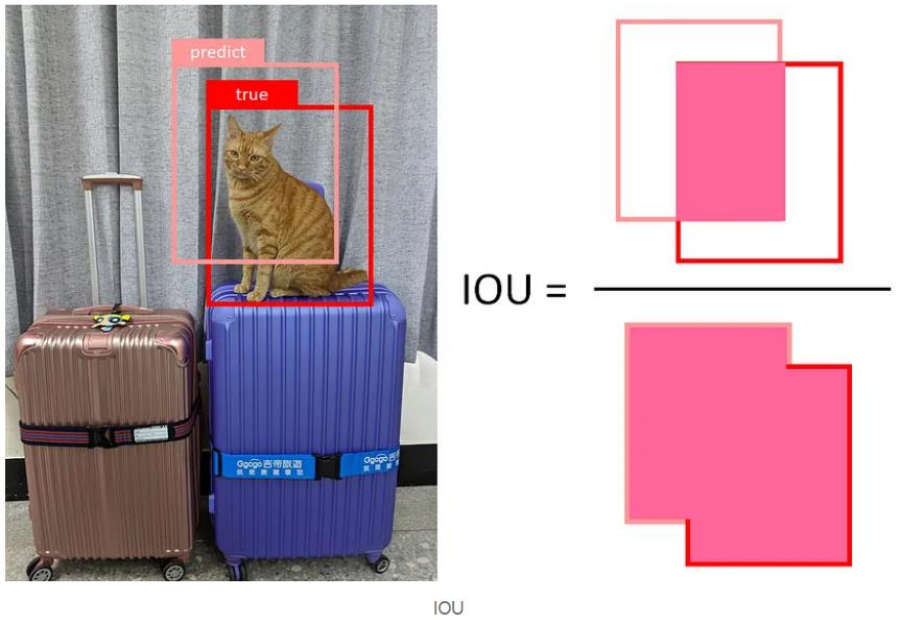
[文章引用] https://medium.com/ching-i/yolo-c49f70241aa7

YOLO3與4 可偵測物體

分類	物體
交通	人、自行車、汽車、摩托車、飛機、巴士、火車、卡車、船、紅綠燈、消防栓、停止標誌、停車收費表、板凳
動物	鳥、貓、狗、馬、羊、牛、象、熊、斑馬、長頸鹿
配件	背包、兩傘、手提包、領帶、手提箱
運動	飛盤、滑雪板、單板滑雪、運動用球、風箏、棒球棒、棒球手套、滑板、衝浪板、網球拍
廚房	瓶子、紅酒杯、杯子、叉子、刀、勺、碗
食物	香蕉、蘋果、三明治、橙子、西蘭花、胡蘿蔔、熱狗、比薩、甜甜圈、蛋糕
家具	椅子、沙發、盆栽植物、床、餐桌、馬桶、電視監視器、筆記本電腦、滑鼠、搖控器、鍵盤、手機、微波、烤箱、烤麵包機、水槽、冰箱、書、時鐘、花瓶、剪刀、泰迪熊、吹風機、牙刷

YOLO 的評估指標

- Evaluation metrics: YOLO 的評估指標主要採取 IOU 和 mAP:
- **IOU (Intersection over Union)**:即兩個 bndBox 的交集 / 兩個 bndBox 的聯集，也就是指 predict 的 bndBox與 Ground Truth 的 bndBox 的交集除以聯集，通常score > 0.5 就被認為是不錯的結果。



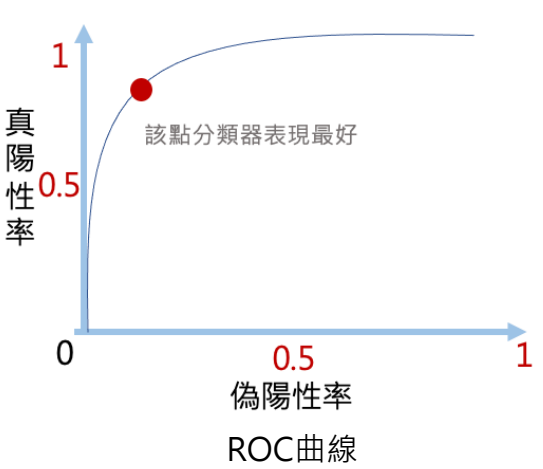
IOU

- **mAP (mean Average Precision)**

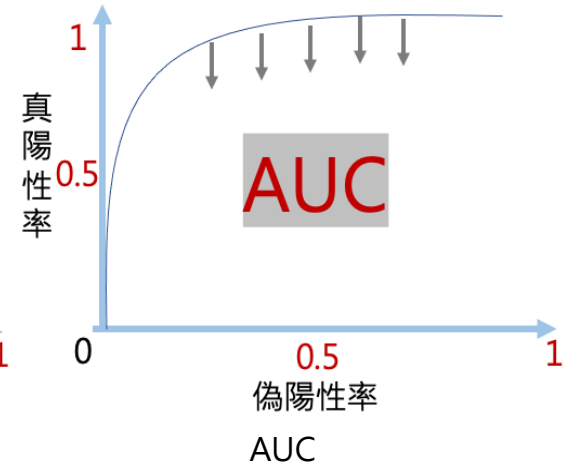
  即各類別ap的平均值，而ap就是指pr curve (precision-recall curve) 的面積 (area under curve, AUC),pr curve 是以 recall 為 x 軸、precision 為y 軸所繪製成的曲線，precision 及 recall 越高，代表模型效能越好。

[補充] AUC (Area Under Curve) 代表在ROC曲線底下的區域面積， ROC底下的面積越大越好，表示曲線更靠近左上方。

- AUC = 1 時，為最理想的情況，分類器做完美的預測。
- AUC > 0.5 時， 分類器效果比隨機猜測(盲猜)還理想，預測有效果。
- AUC = 0.5 時，分類器預測效果與隨機猜測相同，分類器沒有價值。
- AUC< 0.5 時，預測效果比盲猜還要差。



ROC曲線

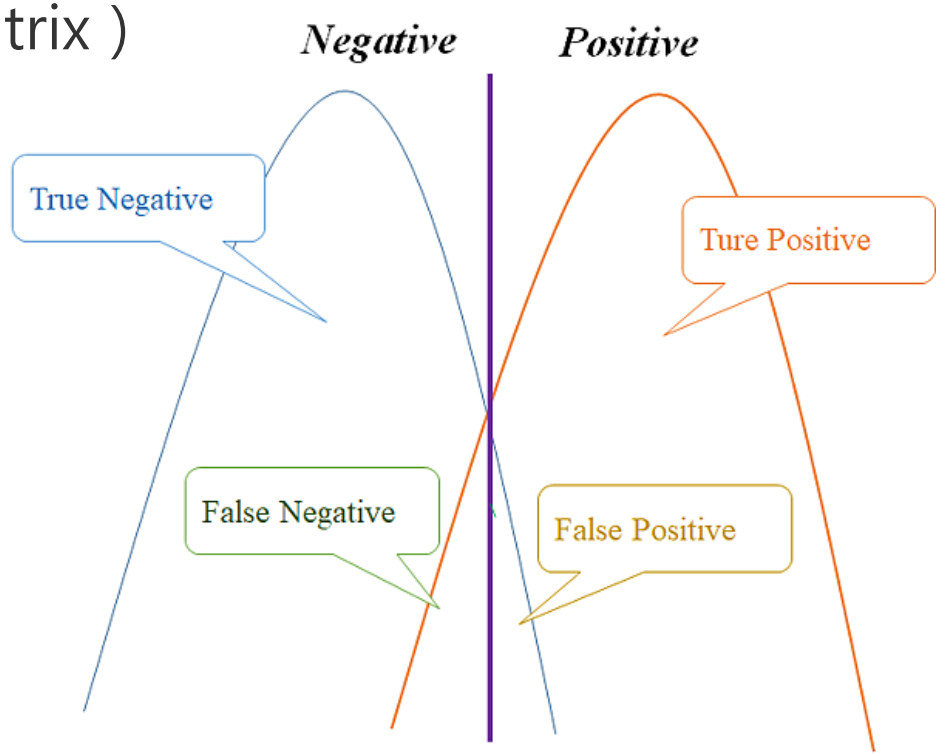


AUC

[文章引用] 1. https://medium.com/ching-i/yolo-c49f70241aa7
2. https://pse.is/4yezye

[補充]混淆矩陣(1/2)

- 通常分類模型的效能是以「**準確率**」(accuracy)來評估，但某些狀況會有特殊考量。
- 在攸關生命的病況篩選過程中，將罹患某種疾病的患者全部篩選出來，比準確率更重要。
- **精準率(precision)**與**召回率(recall)**是另外兩種評估模型優劣的指標, 尤其是**召回率**。
- **混淆矩陣**
- 面對二分類問題（陰性/陽性、正確/錯誤）時，常用的指標稱為混淆矩陣（Confusion Matrix）



預測 \ 真實	實際正向	實際負向
預測正向	True Positive (TP)	False Positive (FP)
預測負向	False Negative (FN)	True Negative (TN)

[Ref]: https://pse.is/4xspv7

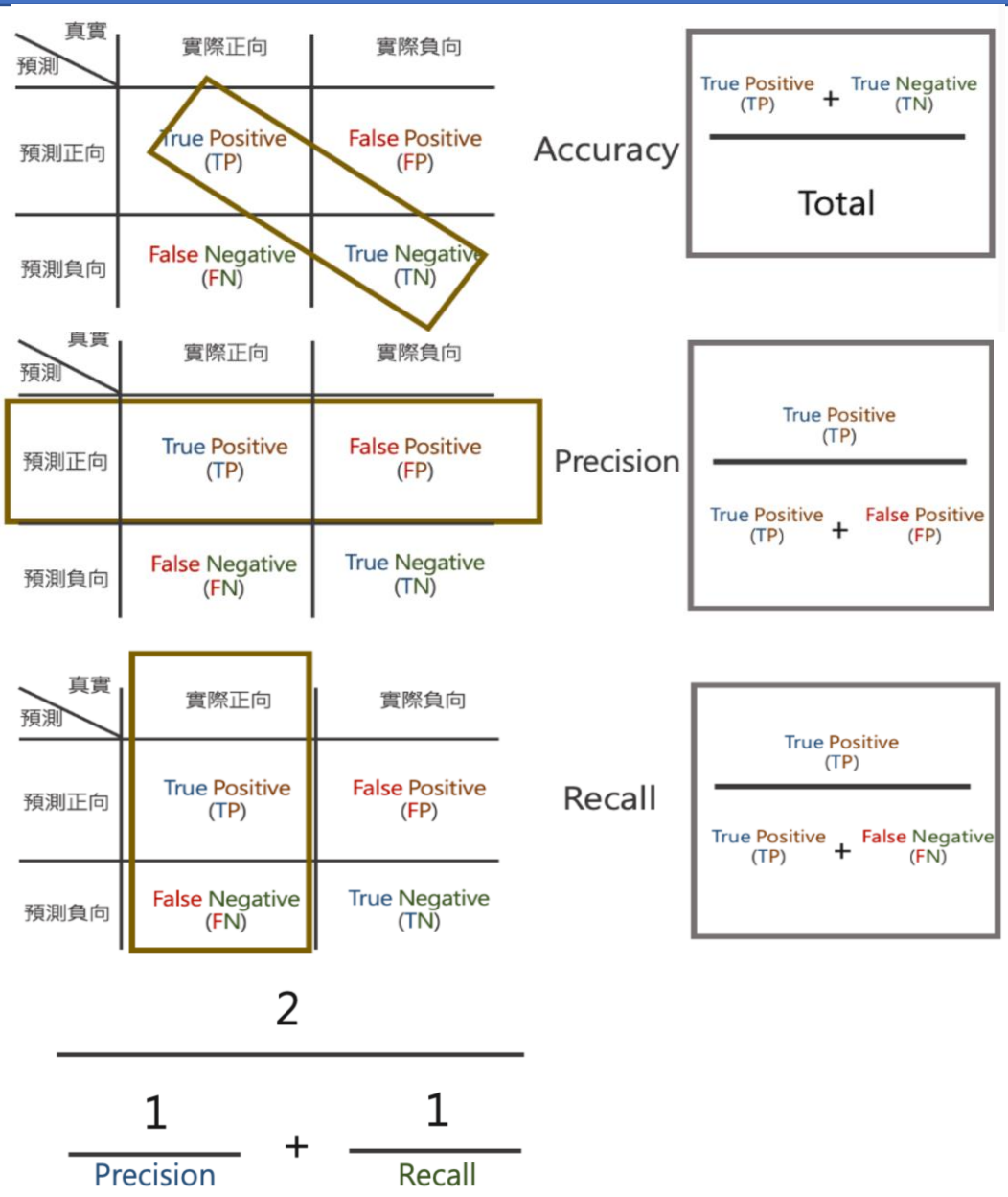
[補充]混淆矩陣(2/2)

- **準確率（Accuracy）**：
  最常用的指標，也就是將所有預測與實際相同的情況相加，並除以所有預測情形個數，也就是評估一模型，能成功預測到結果的準確度。

- **精準率（Precision）**：
  關注在True Positive身上，在預測正向(Positive)的情況下，成功預測到結果的比例。

- **召回率（Recall）**：
  關注在True Positive身上，但其看重的是實際情況在正向(Positive)的情況下，預測也是正向(Positive)的比率。

- **F1 Score**：
  當覺得Precision與Recall指標同等重要時，就用F1 Score來表示。



[Ref]: https://pse.is/4tzdb7

Part 2

Google Colab雲端訓練
客製化 YOLOv4物件辨識

建立YOLO開發環境步驟(1/3)

1.使用PyTorch來檢查GPU的狀況，第一個print是確認GPU能否運作，第二個print 是顯示顯示卡的名稱：

```
import torch
print(torch.cuda.is_available())
print(torch.cuda.get_device_name())
```

2.先將darknet的Github給Clone下來:

```
! git clone https://github.com/AlexeyAB/darknet.git
```

3.移動至darknet的資料夾

```
%cd ./darknet
```

4.修改Makefile，使用Linux的指令sed，-i代表會直接替換檔案內容，替換的模式選擇s(search)，第一個//包住的內容是要搜尋的內容，第二個//中的則是要替換掉的內容。

```
! sed -i 's/OPENCV=0/OPENCV=1/' Makefile
! sed -i 's/GPU=0/GPU=1/' Makefile
! sed -i 's/CUDNN=0/CUDNN=1/' Makefile
! sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
! sed -i 's/LIBSO=0/LIBSO=1/' Makefile
```

建立YOLO開發環境步驟(2/3)

5.下載權重：

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4-tiny.weights
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
```

6.透過下列指令進行測試，coco.data 存放資料集的資訊 像是圖片大小、類別等等；yolov4.cfg 則是存放yolov4神經網路模型的資訊；yolov4.weights 為剛剛下載的訓練好的權重；data/dog.jpg 為輸入的資料；-thresh 閥值 越大需要的信心指數越高。

```
!./darknet detector test ./cfg/coco.data ./cfg/yolov4.cfg ./yolov4.weights data/dog.jpg -i 0 -thresh 0.25
```

7.透過下列的程式來將結果顯示出來，因為matplotlib跟Jupyter有較高的相容性，而Colab使用的是Jupyter Notebook的環境，可以透過 %matplotlib inline 這段程式讓matplot的圖表顯示在Colab當中。

```
import cv2
import matplotlib.pyplot as plt
# 讓 matplot 圖表顯示在Jupyter Notebook裡面
%matplotlib inline
```

建立YOLO開發環境步驟(3/3)

8.第一段程式碼表示的是透過Python建構一個Inference的副函式叫做 darknet_helper，通過這個darknet_helper可以獲取到辨識結果與輸出結果的寬高比例。

```
def darknet_helper(img, width, height):
  darknet_image = make_image(width, height, 3)
  img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img_resized = cv2.resize(img_rgb, (width, height),
                           interpolation=cv2.INTER_LINEAR)...
```

9.第二段程式碼則是如何在Colab上運作Javascript的程式處理即時影像的部分

```
def js_to_image(js_reply):
  """
  Params:
        js_reply: JavaScript object containing image from webcam
  Returns:
        img: OpenCV BGR image
  """
```