相較於 baseline 的 yolov3，本次作業使用 yolov4 opensource 做 inference，以下:

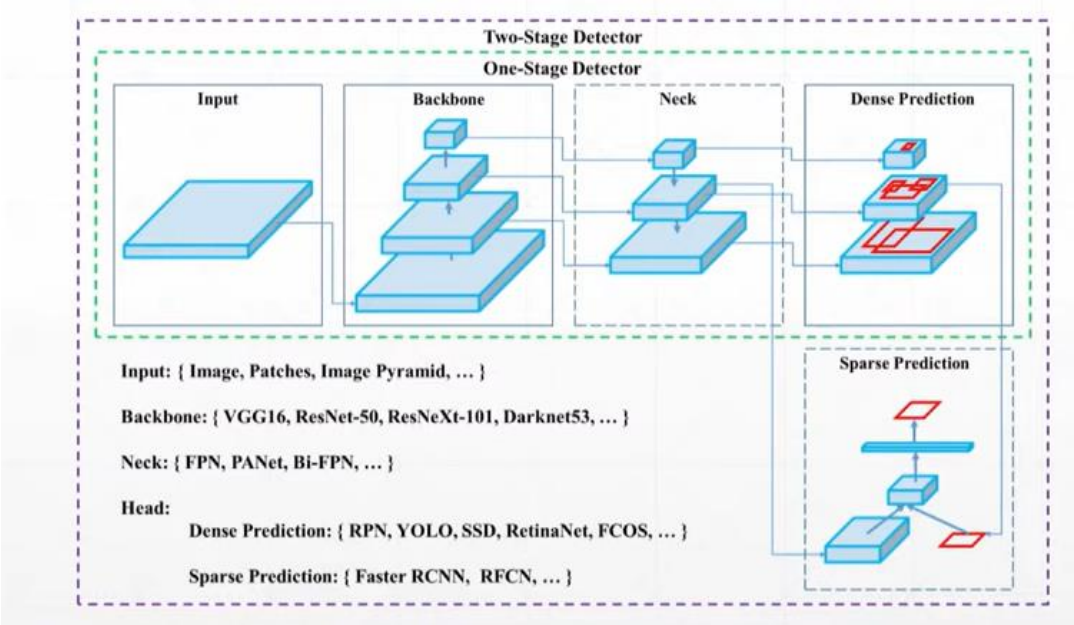
1.簡介物件偵測

2.簡介 yolov3

3.簡介 yolov4 的重要改動

4.用 yolov4 做出的結果

1.簡介物件偵測



目前 object detection 主要分成兩種，一種是 one-stage:如 yolo 家族，一種是 two-stage 如:RCNN 家族，模型架構則可分成三個架構:

Backbone:

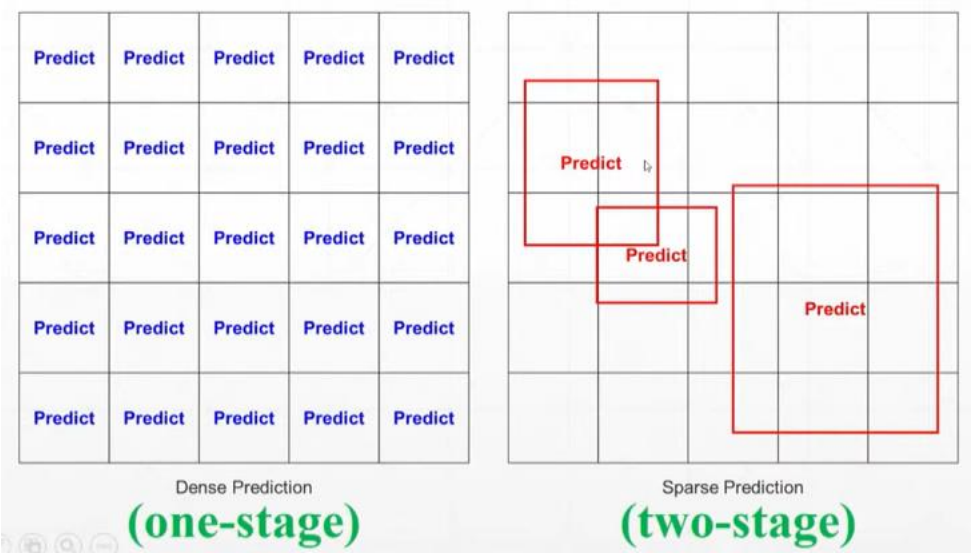
用來抽取 input 影像中的特徵。

Neck:

做特徵的整合，將特徵金字塔(feature pyramid)上層低解析度、豐富語意的 feature map 與下層高解析度低語意特徵的 feature map 結合(add or concat)，讓模型除能學到豐富的語意外，也不失物件的位置資訊。

Head:

dense prediction(one-stage)把影像分成一塊塊 grids，對每個 grid 都做預測， sparse prediction (two stage)先產生可能有 object 的 ROI 區域，再對這些區域預測物件所在位置與類別機率。



2. 簡介 yolov3

- **Backbone**: 使用 Darknet-53(預訓練於 ImageNet)

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

- **Neck**: FPN(feature pyramid network)，輸出三種不同尺寸的 feature map(13x13, 26x26, 52x52)
- **Head**: 三種不同的特徵圖，會各自輸出三種不同大小的 anchor boxes，即在一張特徵圖上的每個 grid 會預測三種不同大小 anchor boxes(以 anchor box 偏移值作輸出)、物件框信度分數(有無物件在框內)、各類別機率(如 COCO dataset 80 類)

最終輸出大小 ：$grid_h \times grid_w \times 3 \times (4+1+80)$ 如下圖

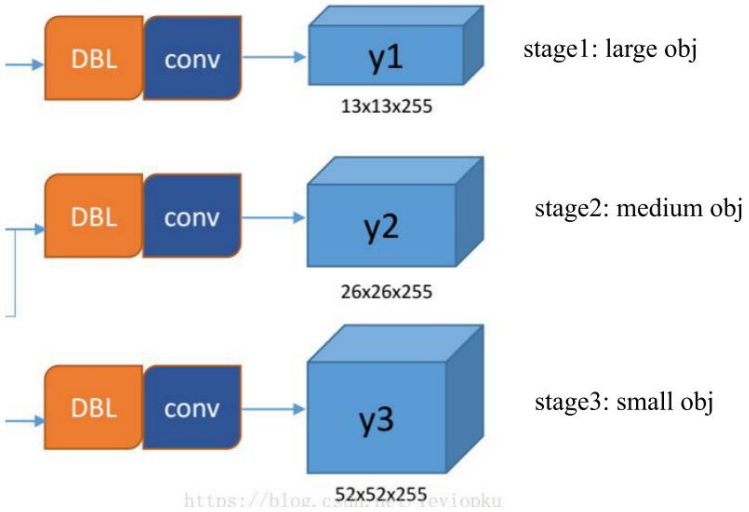
YOLO v3 output

3 * (5 + 80) = 255

3: box per grid

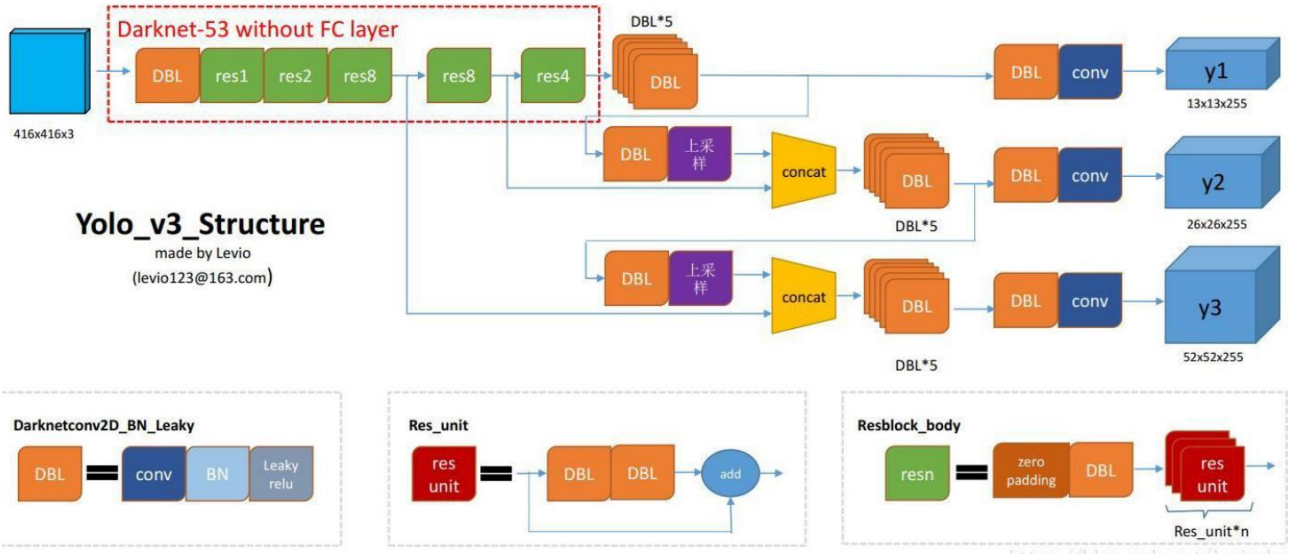
5: (x,y,w,h,confidence)

80: # of classes



Yolov3 整體架構

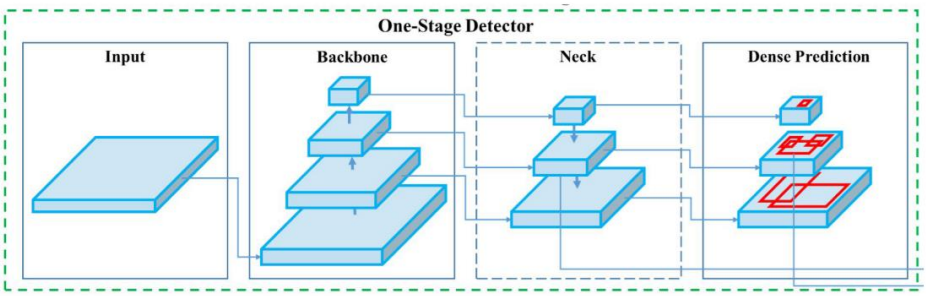
YOLO v3 (2018)



3. 簡介 yolov4 的重要改動

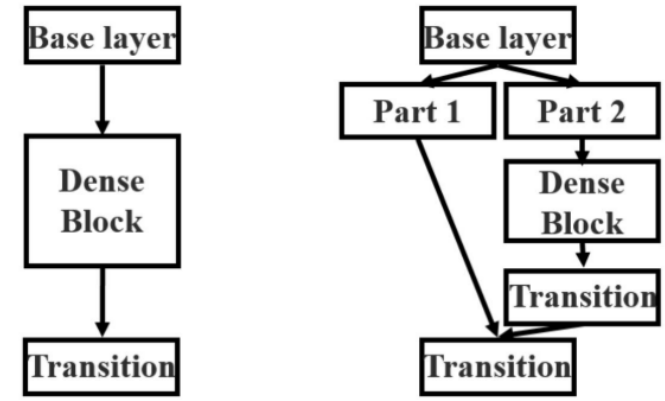
Yolo v4 Architecture

1. Backbone: **CSP**Darknet53
2. Neck: SPP + PANet
3. Head: YOLOv3 (anchor based)



yolov4 加入的新技術:

- **CSPNet: Cross Stage Partial Network**
  結合 PRNet（partial residual networks, PRNet）速度快的優點，同樣在 input 做分流，使計算量下降，另為了讓每層 layer 學到的資訊多樣性增加，多了一個 Transtion 層(如下圖一)，使 backpropogation 時，層與層得出的 gradient 有所不同，提高每一網路層參數的利用率，使最終的結果又快又準(如下圖二)。

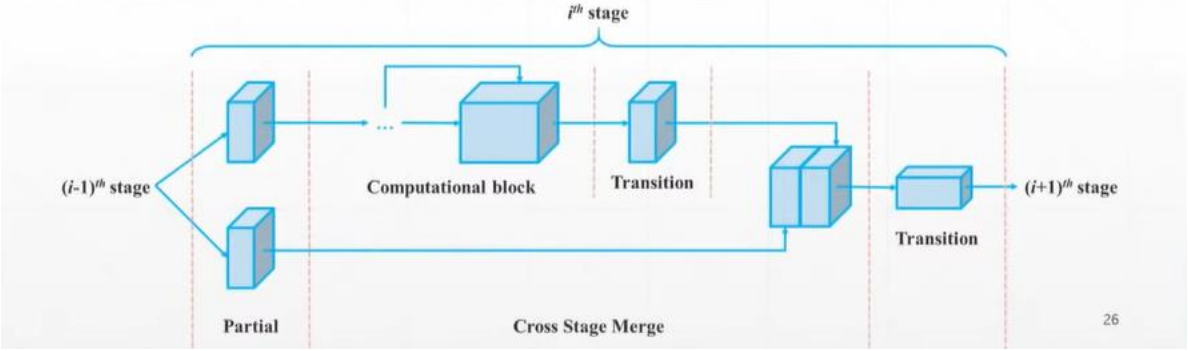


(a) DenseNet (b) CSPDenseNet

Backbone of YOLOv4

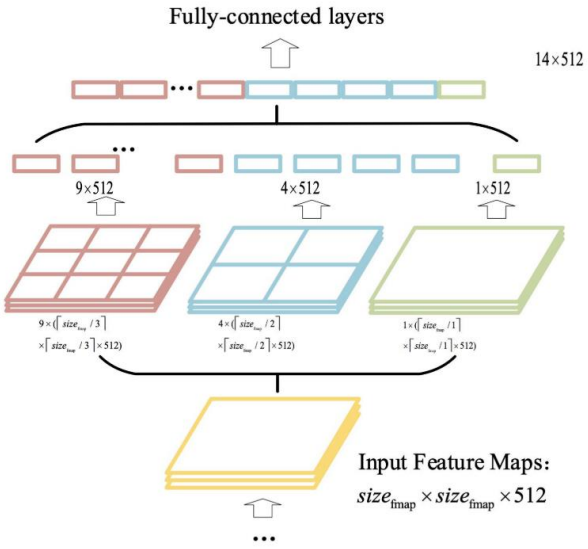
Table 1: Parameters of neural networks for image classification.

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26



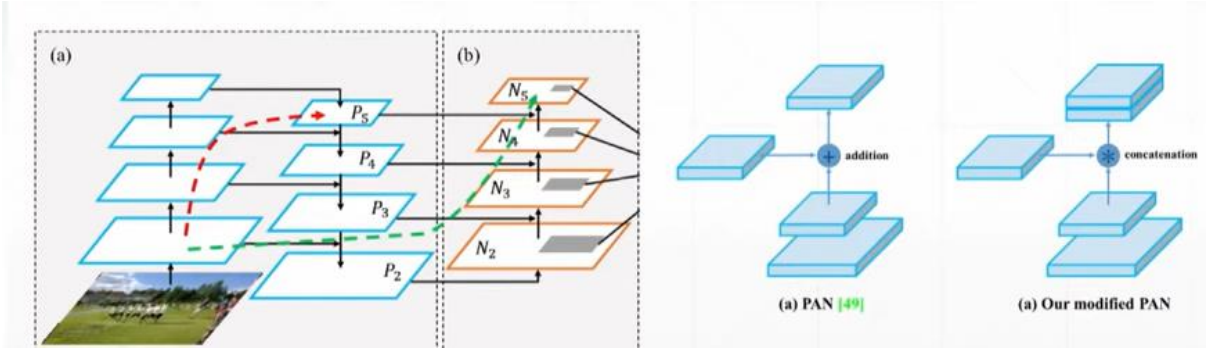
- **Spatial Pyramid Matching (SPP)**

  加入不同大小的 pooling size，增加卷積層的視解豐富度。



- **Path Aggregation Network (PANet)**

  在 neck 的部分感良原先的 PAN，將原先把 feature map 相加的方式改成 concat，以減少運算量，同時不丟失資訊。



- Activation function 的改良

  為了解決預測出來的機率若很高(~1)或很低(~0)，gradient 便趨緩的問題做的改變，提升模型準確度。

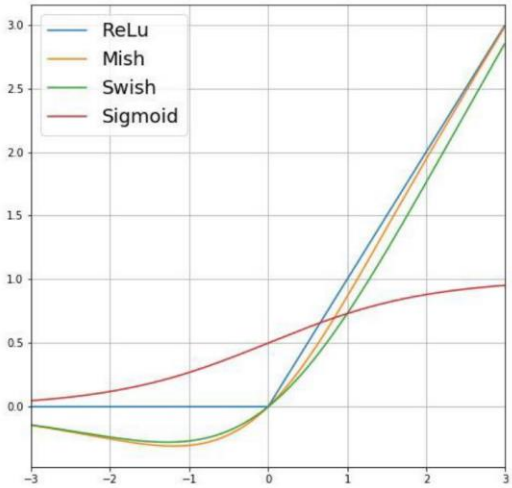
Activation functions

- Swish

  $f(x) = x * \text{sigmoid}(x)$

- Mish

  $f(x) = x \tanh(\ln(1 + e^x))$

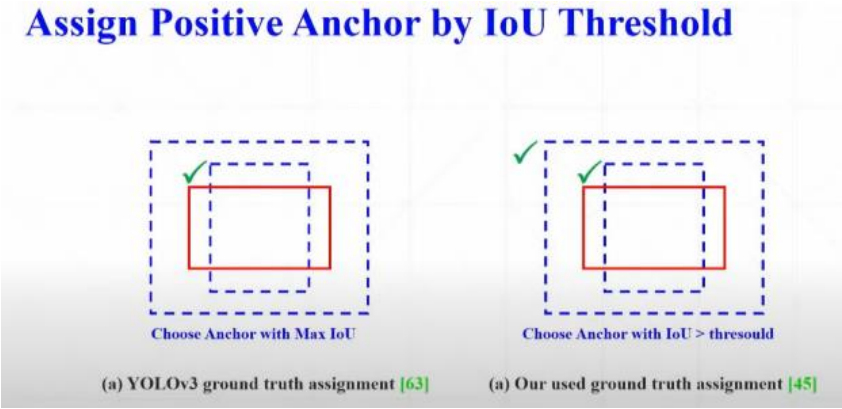


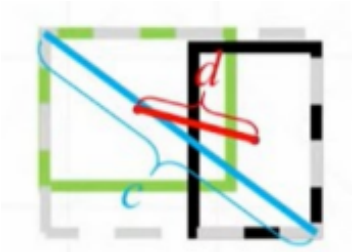
- **IoU 的改良**

  在 anchor box 預測上的改良:

  用 threshold，取代原本只取 IoU 最大才 assign 成 positive anchor。(下圖一)

  以 CIoU/DIoU 取代原先的 IoU/GIoU，比原先的 IoU 公式多考慮了預測出的 anchor box 中心與 ground truth 中心的距離&最長角對角的距離。(下圖二)





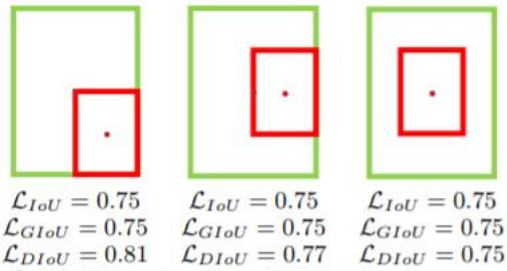
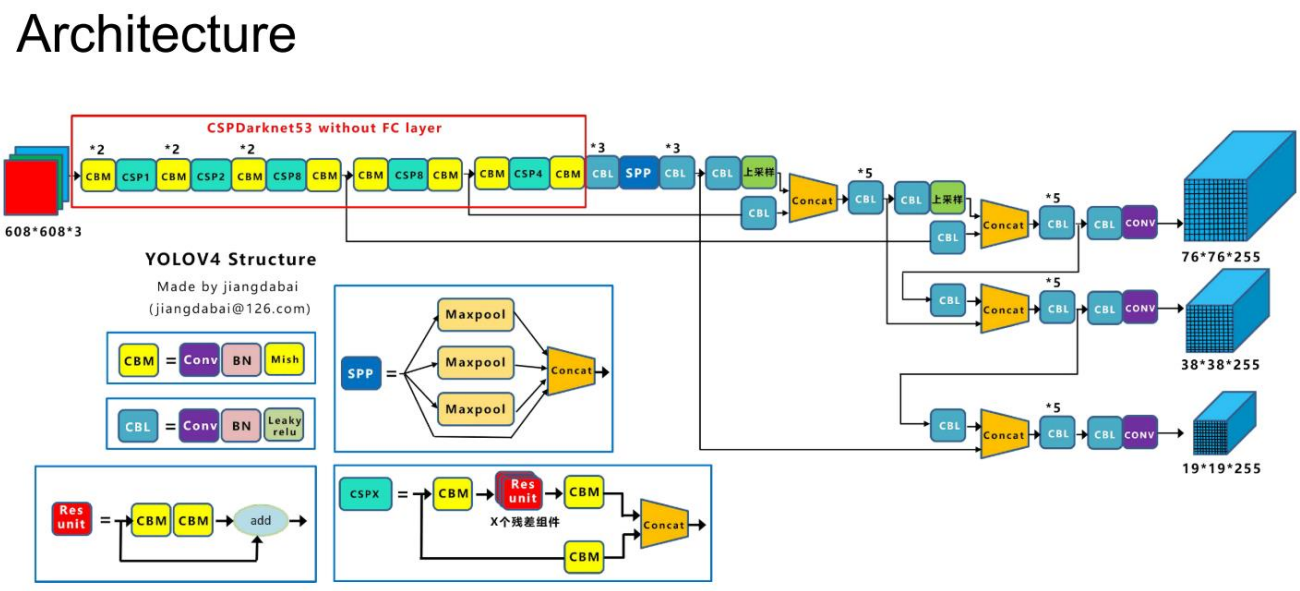


Figure 2: GIoU loss degrades to IoU loss for these cases, while our DIoU loss is still distinguishable. Green and red denote target box and predicted box respectively.

Yolov4 整體架構



3.用 yolov4 做出的結果

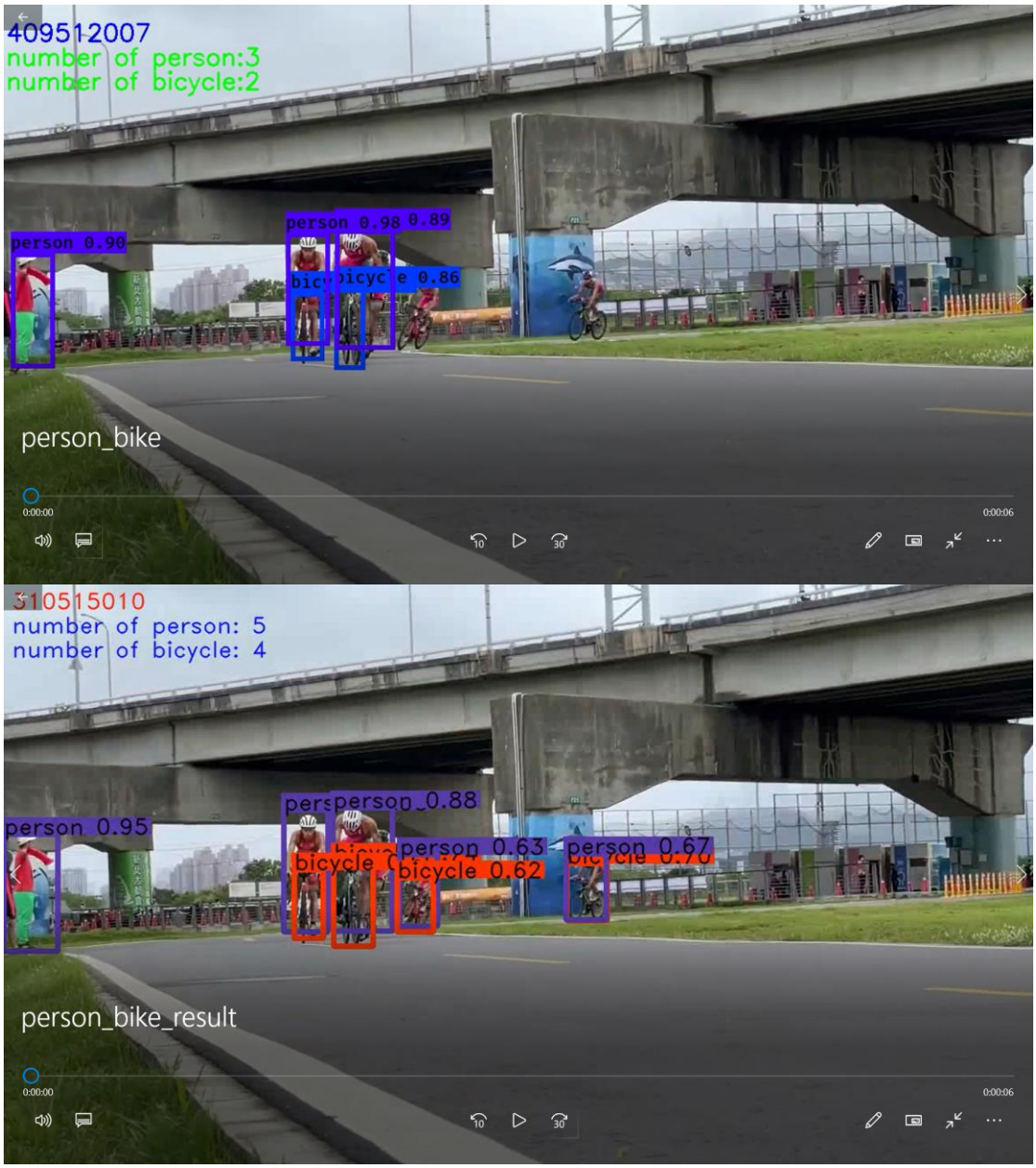
結果影片: output/person_bike_310515010.avi

主要使用的程式碼為 **main_yolov4.py**，其他做為 module 引入

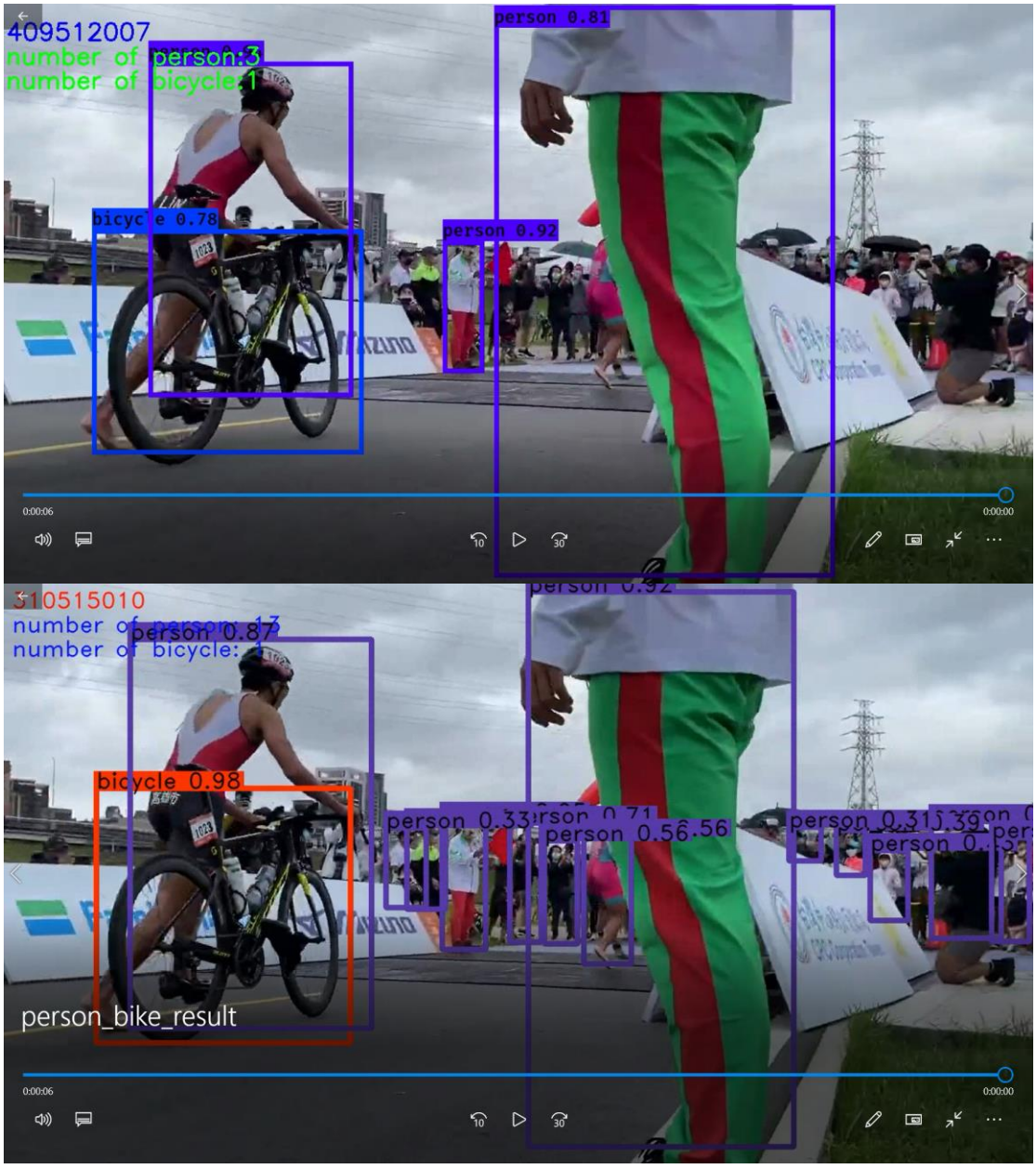
結果準確度明顯在很多處提升了(見影片)，以下舉幾個顯著例子:

(上:範例，下:我的結果)

一開頭



最後


409512007
number of person:3
number of bicycle:1
bicycle 0.78
person 0.81
person 0.92
0:00:06
0:00:00
310515010
person 0.87
bicycle 0.98
person 0.56
person_bike_result
0:00:06
0:00:00

中間

無重複的物件框預測

Reference :

- https://docs.google.com/presentation/d/124s8NIUNF9XXQmHMMH6xA0OwKballmgDzYQLmr7X4AQ/edit#slide=id.p
- https://www.youtube.com/watch?v=HdQgAF-rMKc
- https://news.cnyes.com/news/id/4547626
- https://github.com/taipingeric/yolo-v4-tf.keras