機器學習期末報告

組員 ： S07240013 簡睿德

S07240044 高祺烜

S07240047 陳昱廷

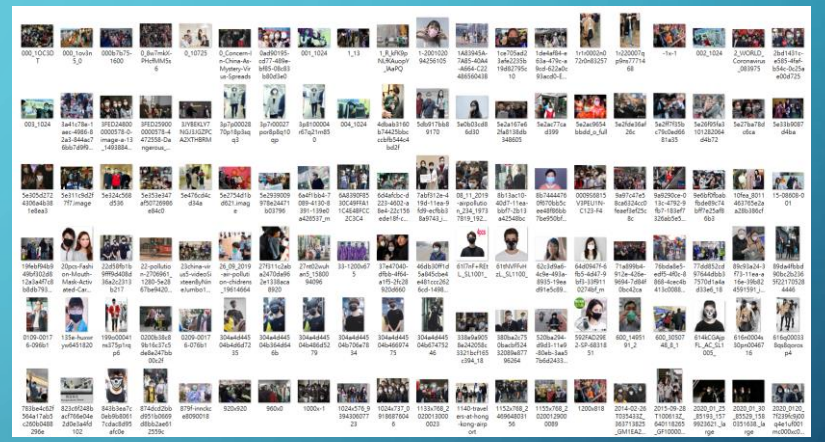
動機

由於前陣子疫情嚴重，在車站、校園、百貨公司等人流眾多處，大多需要人力監測有無戴口罩。透過相機偵測民眾是否戴口罩，可減少人力需求及人與人之間的接觸。

報告流程

製作流程

1. 利用yolov3模型來訓練678張圖片
2. 辨識圖片中的人物是否有戴口罩
3. 框住偵測物件區域並顯示辨識結果
   None, bad, good
4. 使用攝像頭即時偵測物件



(678張圖片訓練庫)

影像、數據檔:https://drive.google.com/file/d/1rf-ggLJ1igZaW7yMbApr2ysOenFOucJ3/view

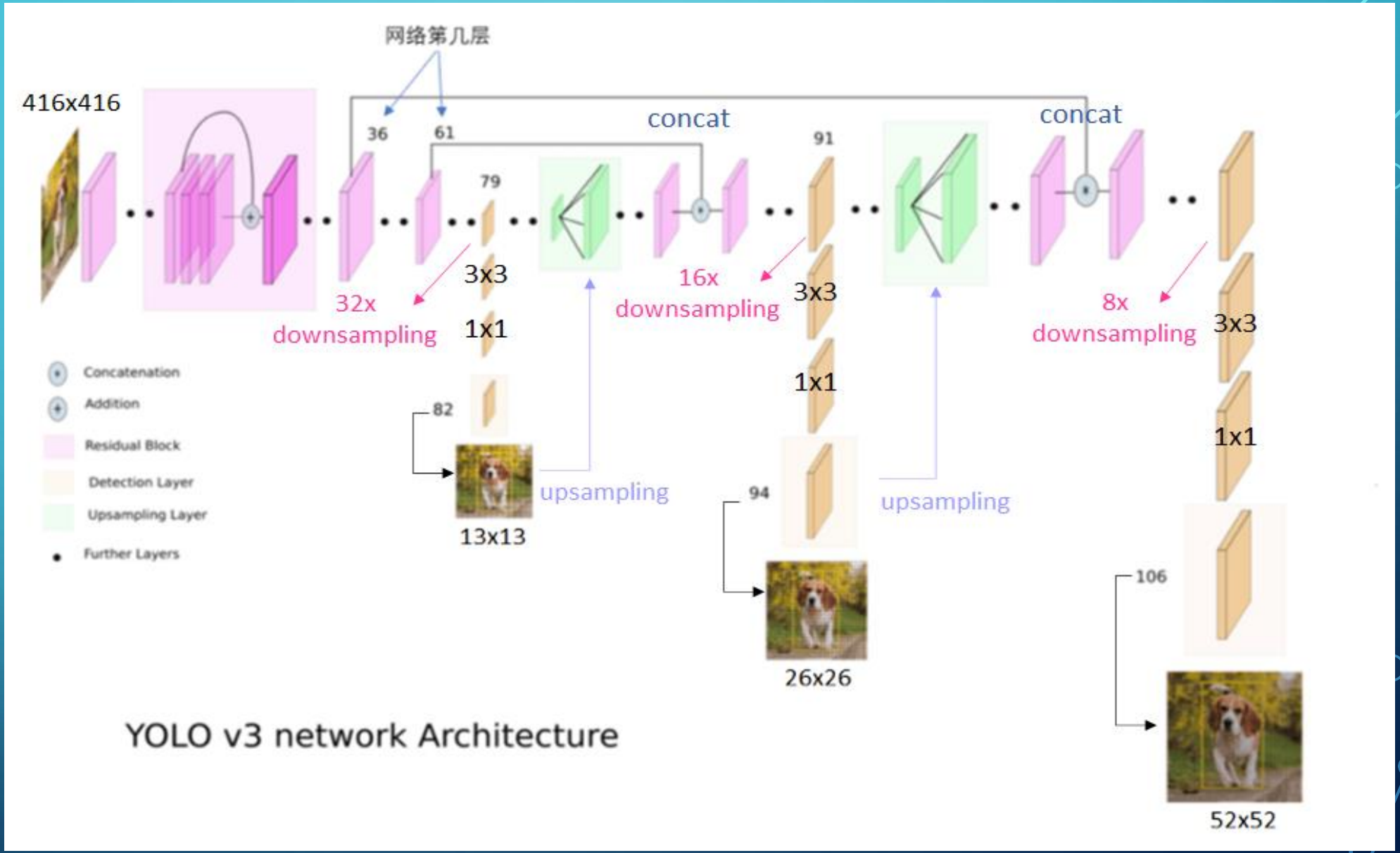
YOLO —YOU ONLY LOOK ONCE

YOLOv3是關於物件偵測(object detection)的類神經網路演算法。已發布的模型可識別圖像和視頻中不同的物件，但最重要的是它速度超快。

https://learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/

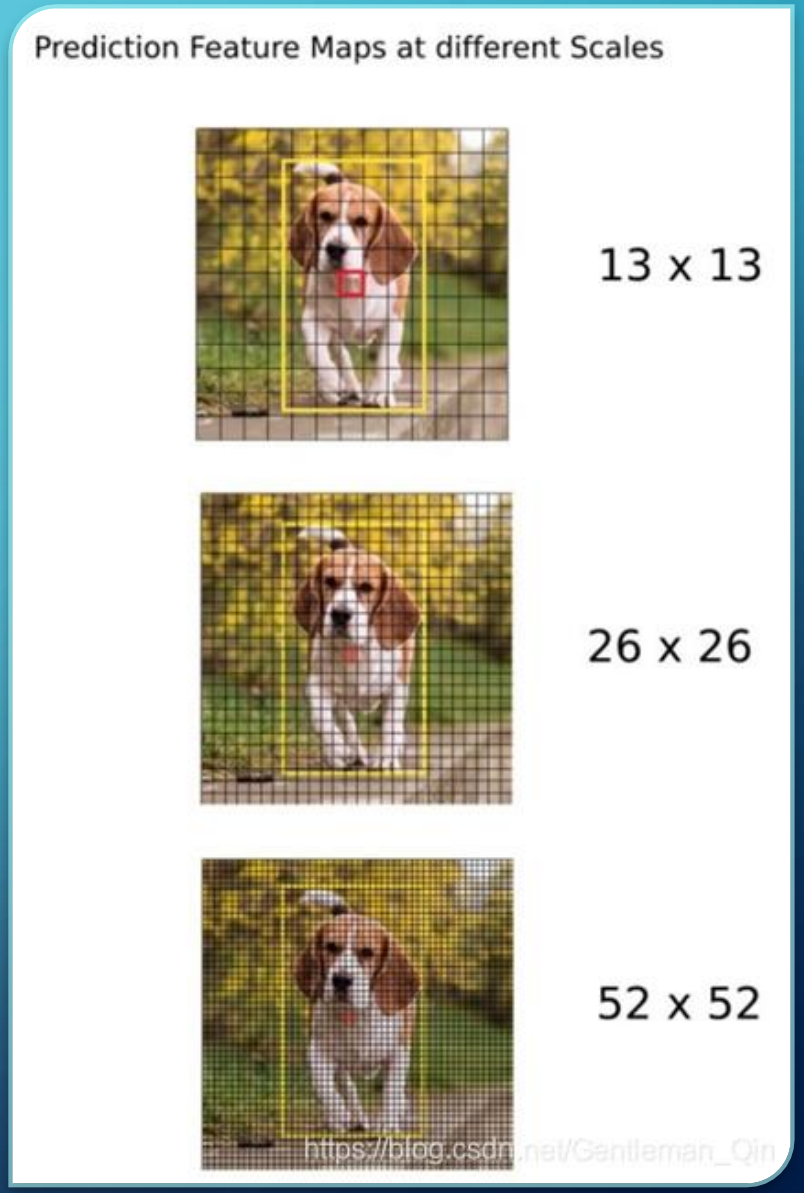
- DarkNet-53 是一個深度為 53 層的捲積神經網絡。您可以從 ImageNet 數據庫加載對超過一百萬張圖像進行訓練的網絡的預訓練版本。
- v2 的 darknet-19 變成了 v3 的 darknet-53是為了需要上取樣(up-sampling)，讓圖片改變尺寸，可以讓卷積層更多，雖然速度相對會比較慢，但是圖片分割得更細。

架構


网络第几层
416x416
36
61
79
91
concat
concat
3x3
1x1
32x
downsampling
16x
downsampling
8x
downsampling
3x3
1x1
3x3
1x1
82
94
106
upsampling
upsampling
13x13
26x26
52x52
Concatenation
Addition
Residual Block
Detection Layer
Upsampling Layer
Further Layers
YOLO v3 network Architecture

多尺度檢測

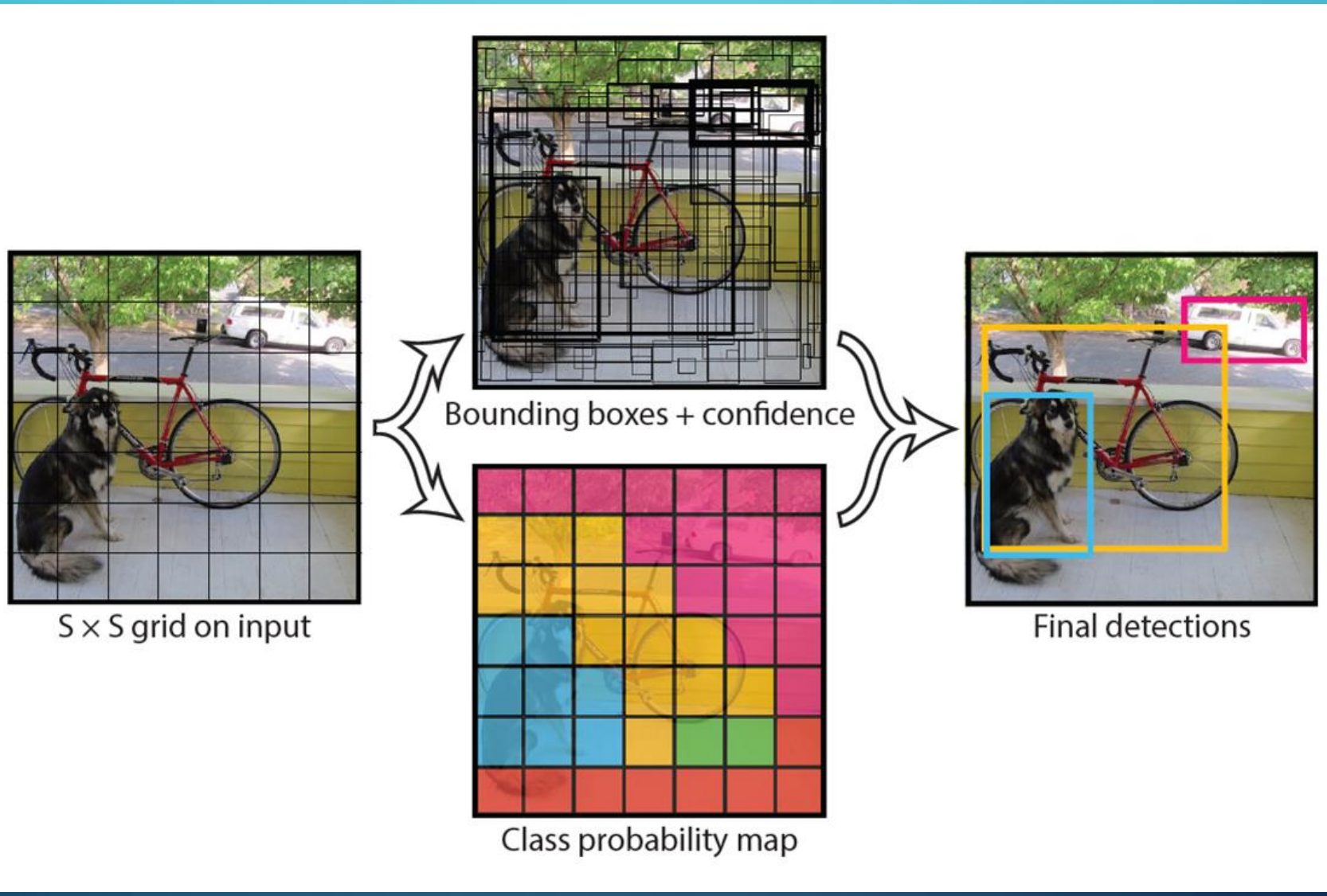
- v3最突出的特點是它可以在三種不同的尺度（13*13、26*26、52*52）上進行檢測，最終對於小目標的檢測效果提昇明顯。
- 利用捲積神經網路的滑動視窗演算法，將圖片分成SxS個網格(grid cell)來偵測物件所在，如果某個物件的中心點落在此網格內，則這網格負責預測這個物件。



https://even1018.pixnet.net/blog/post/355001102-yolo-v3-bounding-boxes%E5%9B%9E%E6%AD%B8%E5%8E%9F%E7%90%86%E5%AD%B8%E7%BF%92

YOLO是如何運作的

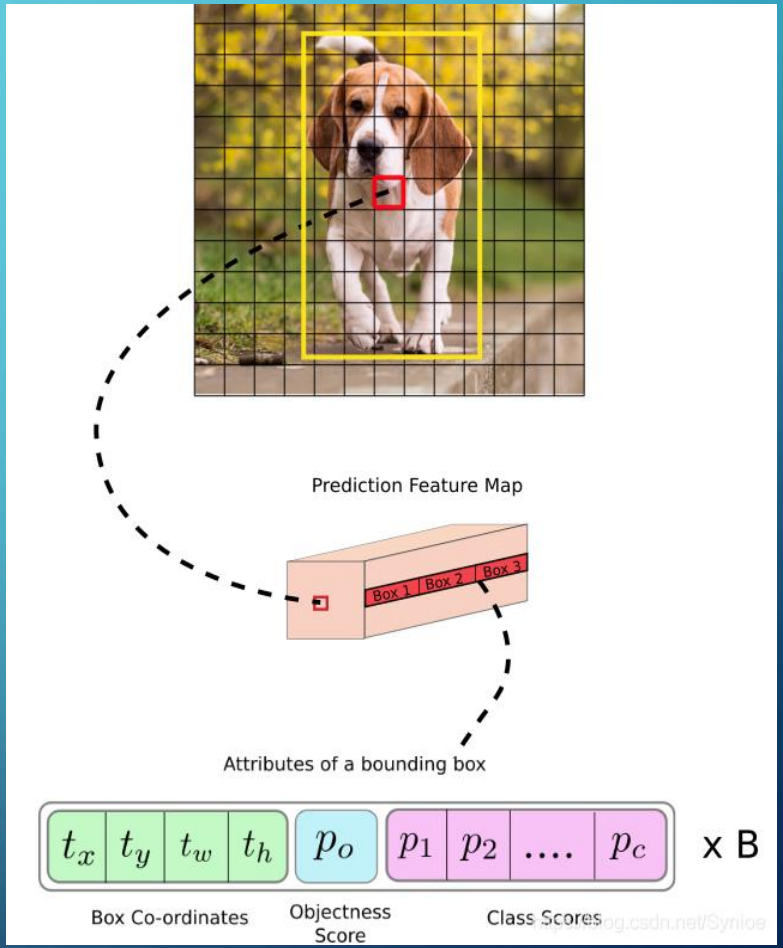
- 將單個神經網絡應用於完整圖像。該網絡將圖像劃分為多個區域並預測每個區域的邊界框和機率。這些邊界框由預測機率加權。
- 它與基於分類器的系統相比，此模型有以下幾個優點
  1. 它在測試時查看整個圖像，因此它的預測是根據圖像中的全局上下文提供的。
  2. 它還可以通過單個網絡評估進行預測，這與R-CNN等系統不同，R-CNN需要數千張圖像才能獲得。這使它非常快


Bounding boxes + confidence
S × S grid on input
Class probability map
Final detections

預測物件特徵

- 其每個網格內容存放的資訊是預測資訊（邊界框的坐標、類別標籤等）
- 每個尺度的 feature map 會預測出3個 Anchor prior，而 Anchor prior 的大小則採用K-means進行聚類分析(YOLOv3 延續了 YOLOv2 的作法)。

tx	ty	tw	th	P0	P1、p2、p3
預測中心點x	預測中心點y	預測框寬度	預測框高度	信心度	類別對應到的機率



程式碼–擷取物件位置

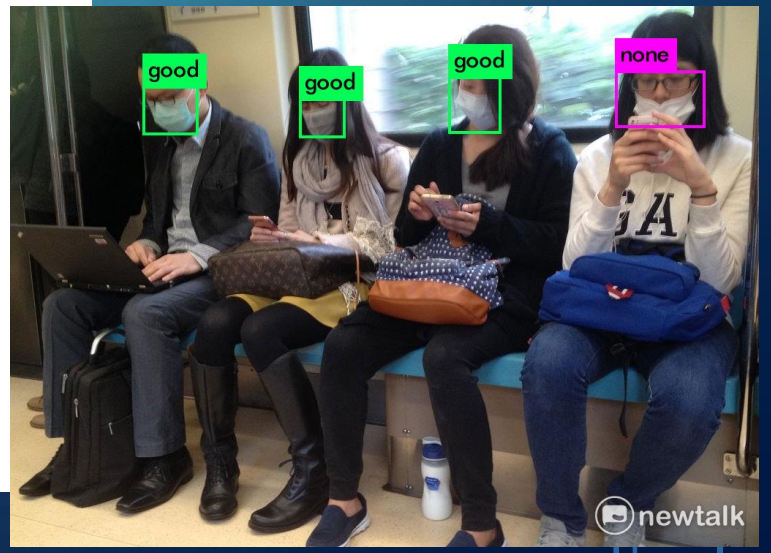
```
class_ids = []
confidences = []
boxes = []

for out in outs:
    for detection in out:
        tx, ty, tw, th, confidence = detection[0:5]
        scores = detection[5:]
        class_id = np.argmax(scores)
        if confidence > 0.3:
            center_x = int(tx * width)
            center_y = int(ty * height)
            w = int(tw * width)
            h = int(th * height)

            # 取得箱子方框座標
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)
print(len(boxes))
#non-maxima suppression
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.3, 0.4)
```

程式碼-偵測物件區域的框框

```
font = cv2.FONT_HERSHEY_PLAIN

for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        color = colors[class_ids[i]]
        cv2.rectangle(img2, (x, y), (x + w, y + h), color, 2)
        cv2.putText(img2, label, (x, y - 5), font, 2, color, 3)
#print(img2.shape)
#%pylab inline
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = [15, 10]
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
```

程式碼-DARKNET訓練模型

```
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  16  0.150 BFLOPs
    1 max          2 x 2 / 2   416 x 416 x  16   ->   208 x 208 x  16
    2 conv     32  3 x 3 / 1   208 x 208 x  16   ->   208 x 208 x  32  0.399 BFLOPs
    3 max          2 x 2 / 2   208 x 208 x  32   ->   104 x 104 x  32
    4 conv     64  3 x 3 / 1   104 x 104 x  32   ->   104 x 104 x  64  0.399 BFLOPs
    5 max          2 x 2 / 2   104 x 104 x  64   ->    52 x  52 x  64
    6 conv    128  3 x 3 / 1    52 x  52 x  64   ->    52 x  52 x 128  0.399 BFLOPs
    7 max          2 x 2 / 2    52 x  52 x 128   ->    26 x  26 x 128
    8 conv    256  3 x 3 / 1    26 x  26 x 128   ->    26 x  26 x 256  0.399 BFLOPs
    9 max          2 x 2 / 2    26 x  26 x 256   ->    13 x  13 x 256
   10 conv    512  3 x 3 / 1    13 x  13 x 256   ->    13 x  13 x 512  0.399 BFLOPs
   11 max          2 x 2 / 1    13 x  13 x 512   ->    13 x  13 x 512
   12 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   13 conv    256  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 256  0.089 BFLOPs
   14 conv    512  3 x 3 / 1    13 x  13 x 256   ->    13 x  13 x 512  0.399 BFLOPs
   15 conv     24  1 x 1 / 1    13 x  13 x 512   ->    13 x  13 x  24  0.004 BFLOPs
   16 yolo
   17 route  13
   18 conv    128  1 x 1 / 1    13 x  13 x 256   ->    13 x  13 x 128  0.011 BFLOPs
   19 upsample            2x    13 x  13 x 128   ->    26 x  26 x 128
   20 route  19 8
   21 conv    256  3 x 3 / 1    26 x  26 x 384   ->    26 x  26 x 256  1.196 BFLOPs
   22 conv     24  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x  24  0.008 BFLOPs
   23 yolo
Loading weights from /content/cfg_mask/yolov3-tiny_250000.weights...Done!
/content/yolo/703.jpg: Predicted in 0.003472 seconds.
good: 96%
good: 94%
good: 83%
none: 93%
```



YOLOV3 V2比較

V3優於V2、V1的地方

YOLOv3 在 YOLOv2 的基礎上，改良了網路 backbone、利用多尺度特徵圖（feature map）進行檢測、改用多個獨立的 Logistic regression 分類器取代softmax 來預測類別分類

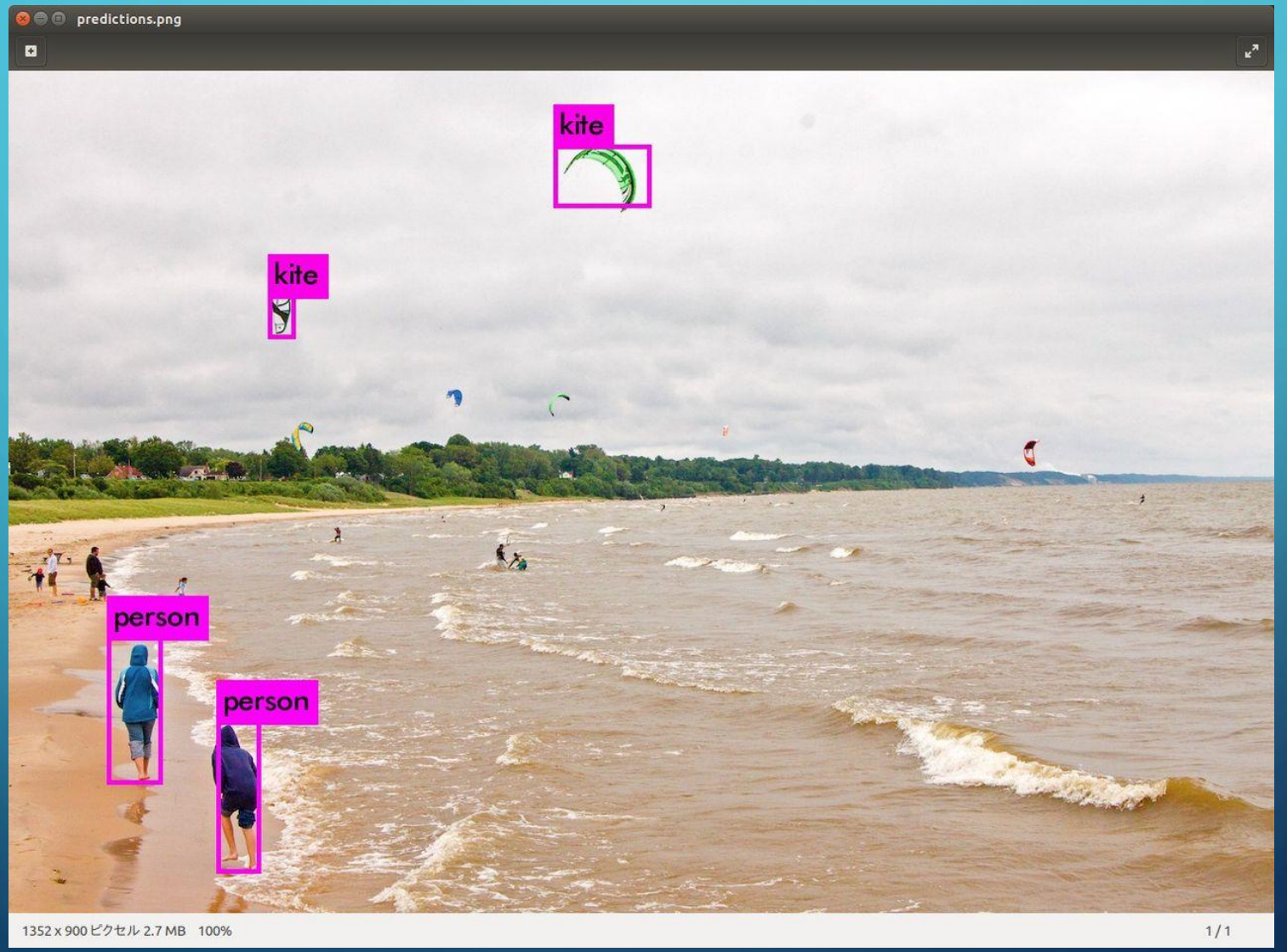
	YOLOv3	YOLOv2
grid size	13x13, 26x26, 52x52	13x13
Anchor Box 個數	3x3	5
loss function	confidence score 跟 class的loss function使用 binary cross-entropy	都使用 MSE (Mean Squared Error)

YOLOv1 vs YOLOv2 vs YOLOv3 輸出

YOLOv1	YOLOv2	YOLOv3
feature map (7x7) 的每一個 grid 中預測出 2個 bndBox 及分類機率值，每個 bndBox 預測出5個值	feature map (13x13) 的每一個 grid 中預測出 5個 bndBox (對應5個 Anchor Box)，每個 bndBox 預測出5個值及分類機率值	3個 feature map的每一個 grid 中預測出 3個 bndBox (對應3個 Anchor prior)，每個 bndBox 預測出5個值及分類機率值
總輸出: 7x7x(5x2+n) n為分類的數量	總輸出: 13x13x5x(5+n)	總輸出: 13x13x3x(5+n) + 26x26x3x(5+n) + 52x52x3x(5+n)

YOLOV2 VS YOLOV3

YoloV2


predictions.png
kite
kite
person
person
1352 x 900 ピクセル 2.7 MB 100%
1 / 1

YoloV3


kite
kite
kite
kite
kite
kite
kite
person
person
person
person
person
person
person
知乎 @Pascal

YoloV2


第八届中美互联网论坛
The 8th US-China Internet Industry Forum / Seattle, September 23, 2015
person
person
person
person
person
person
person
person
person
person

YoloV3


第八届中美互联网论坛
China Internet Industry Forum / Seattle, September 23, 2015
person
tie

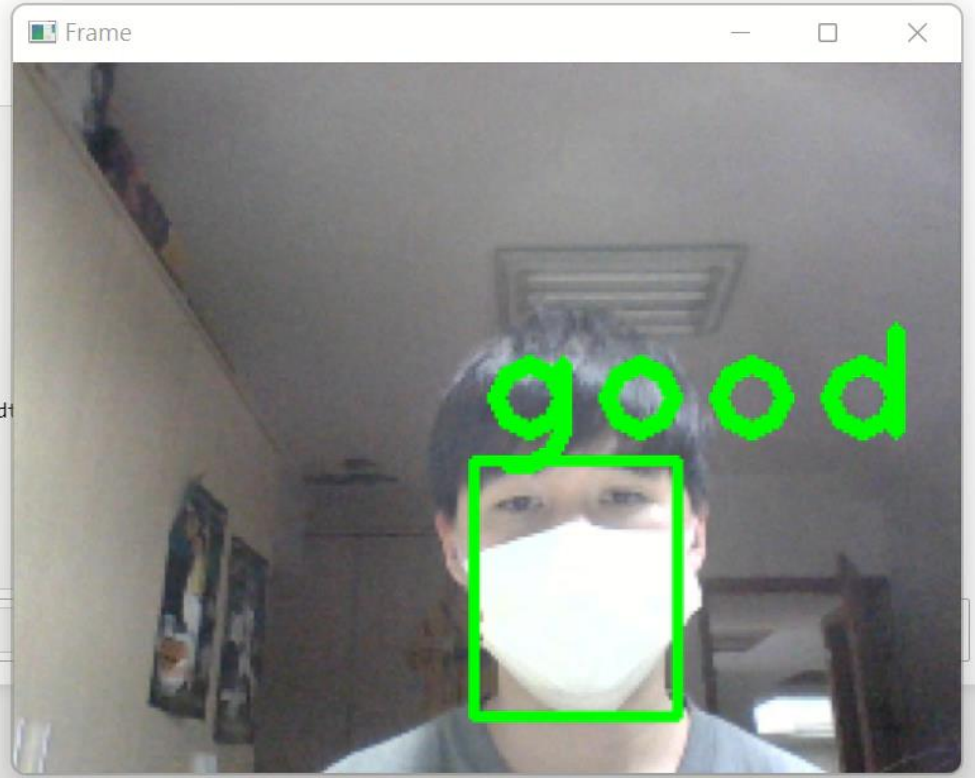
成果

即時偵測演示

使用攝像頭即時偵測物件

In [*]:
```
import cv2
import imutils
import time

VIDEO_IN = cv2.VideoCapture(0)


while True:
    hasFrame, frame = VIDEO_IN.read()

    img = yolo_detect(frame)
    cv2.imshow("Frame", imutils.resize(img, widt

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

VIDEO_IN.release()
cv2.destroyAllWindows()
```

In [ ]:



參考資料

https://github.com/ywchiu/largitdata/blob/master/code/Course_127.ipynb

https://github.com/ywchiu/largitdata/blob/master/code/Course_128.ipynb

https://www.youtube.com/watch?v=vGhIhitQHBE&t=37s

https://www.youtube.com/watch?v=C0Otd6_jGmE&t=1106s

https://www.youtube.com/watch?v=T_zFMRFCFfk