電腦視覺應用
Applications of Computer Vision

Application for deep learning models: yolo object detection

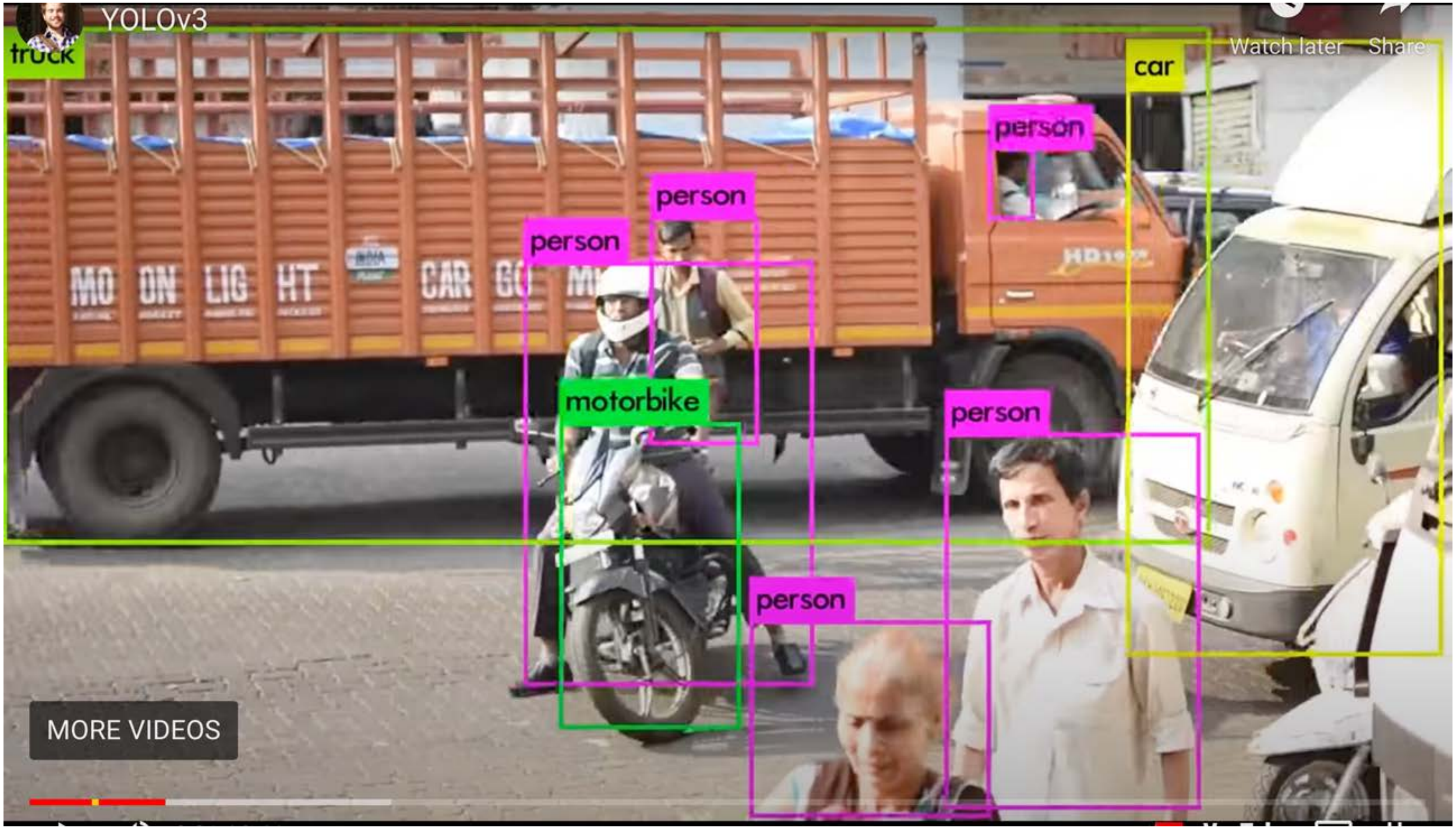
深度學習模型應用：
yolo 物件偵測

孫士韋

Shih-Wei Sun

swsun@newmedia.tnua.edu.tw

Yolo v3 (2018)

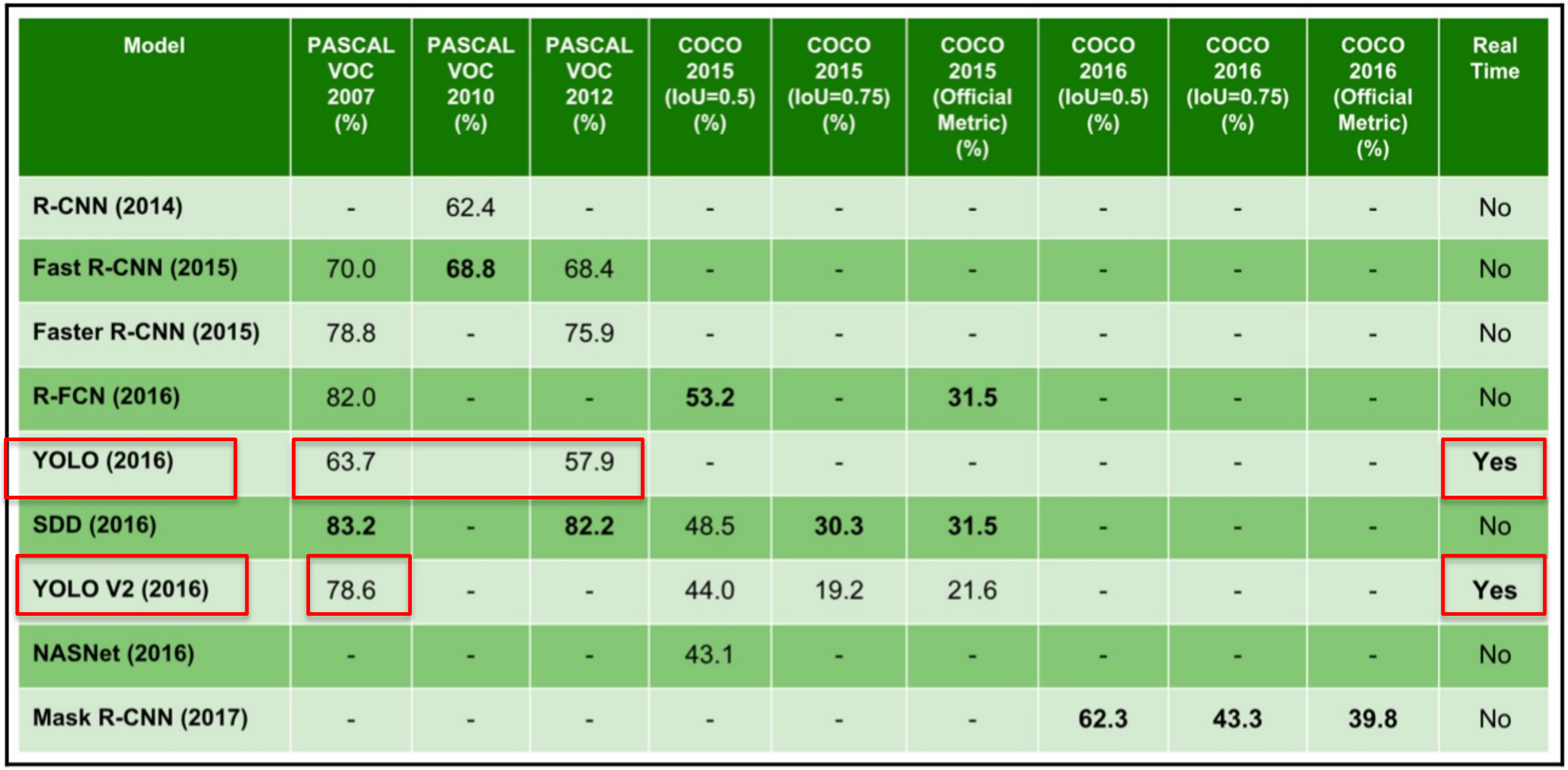


https://mropengate.blogspot.com/2018/06/yolo-yolov3.html

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

Deep learning for object detection

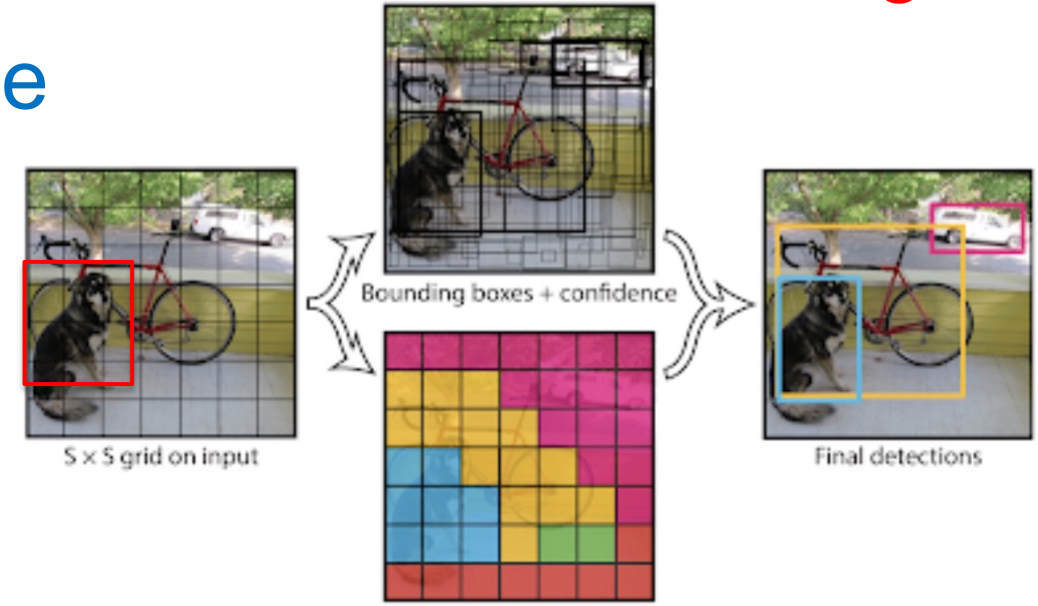
- mAP scores



Model	PASCAL VOC 2007 (%)	PASCAL VOC 2010 (%)	PASCAL VOC 2012 (%)	COCO 2015 (IoU=0.5) (%)	COCO 2015 (IoU=0.75) (%)	COCO 2015 (Official Metric) (%)	COCO 2016 (IoU=0.5) (%)	COCO 2016 (IoU=0.75) (%)	COCO 2016 (Official Metric) (%)	Real Time
R-CNN (2014)	-	62.4	-	-	-	-	-	-	-	No
Fast R-CNN (2015)	70.0	**68.8**	68.4	-	-	-	-	-	-	No
Faster R-CNN (2015)	78.8	-	75.9	-	-	-	-	-	-	No
R-FCN (2016)	82.0	-	-	**53.2**	-	**31.5**	-	-	-	No
YOLO (2016)	63.7		57.9	-	-	-	-	-	-	**Yes**
SDD (2016)	**83.2**	-	**82.2**	48.5	**30.3**	**31.5**	-	-	-	No
YOLO V2 (2016)	78.6	-	-	44.0	19.2	21.6	-	-	-	**Yes**
NASNet (2016)	-	-	-	43.1	-	-	-	-	-	No
Mask R-CNN (2017)	-	-	-	-	-	-	**62.3**	**43.3**	**39.8**	No

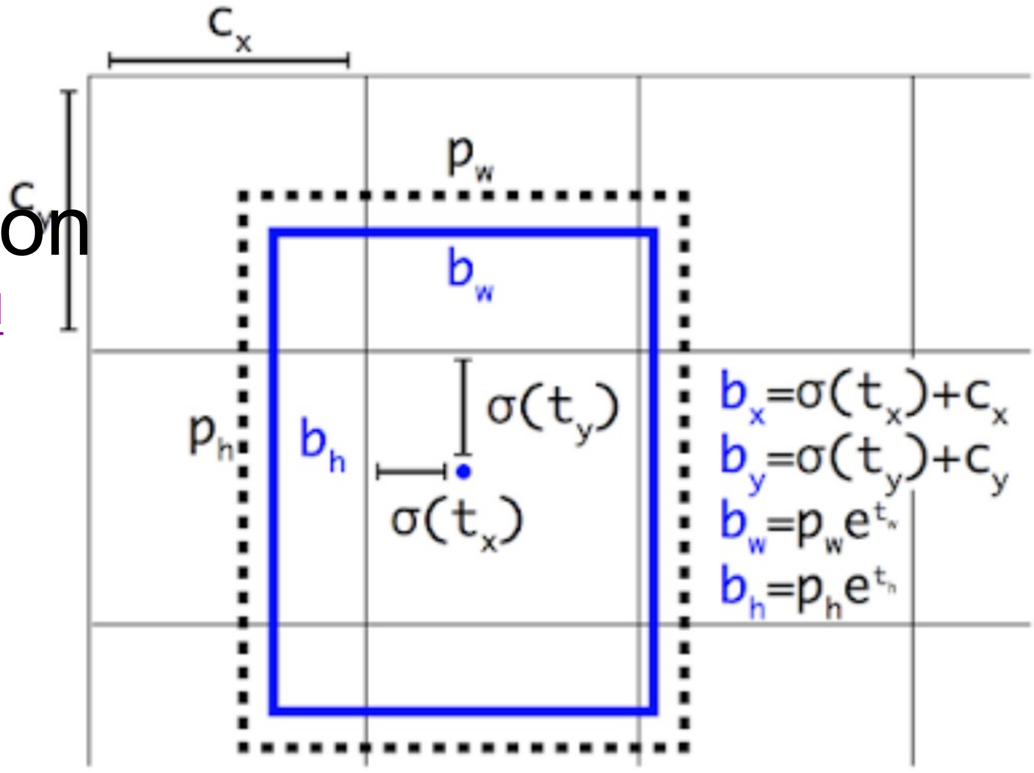
Basics for YOLO (1/3)

- Yolo: You only look once (2016, v1)
  - RCNN, fast RCNN, faster RCNN, Yolo
  - Whole image: input for the neural network (NN)
- Predict: location of the bounding box
- Real-time



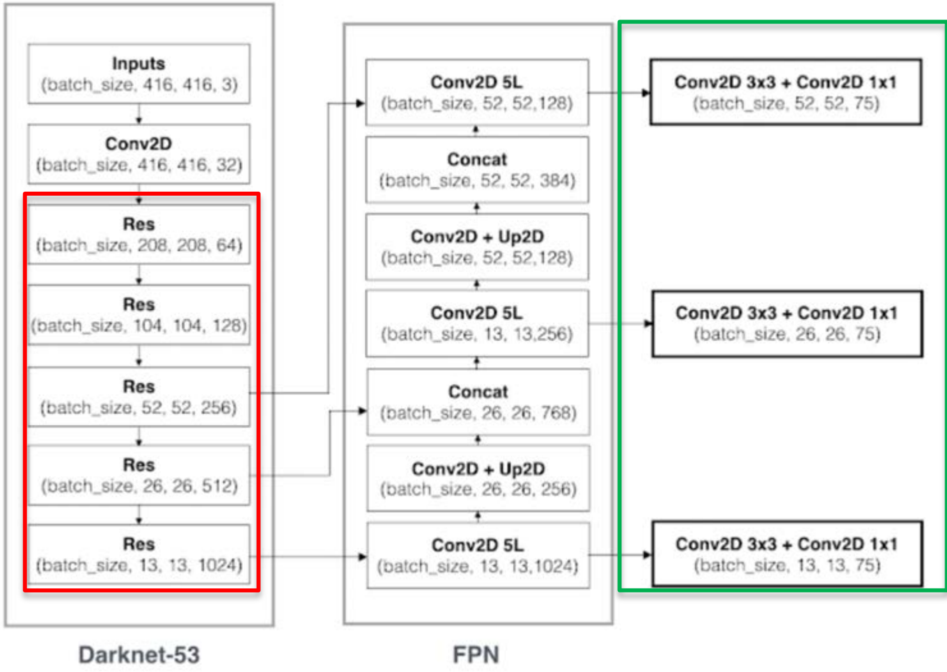
Basics for YOLO (2/3)

- Anchor box: from Faster RCNN (YOLO v2)
- Predict relative position in an Anchor box
  - K-means
- Conv layer
- Batch normalization
  - https://iter01.com/556333.html
- Remove dropout
- Resolution
  - improvement



Basics for YOLO (3/3)

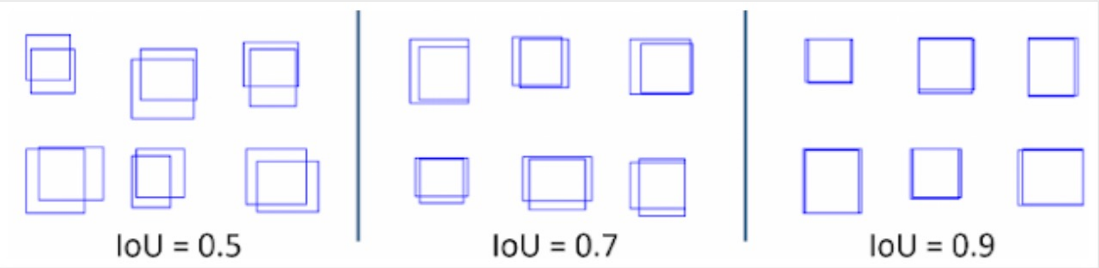
- Using Darknet-53: 53 layers deep NN
  - ResNet
- FPN network: Feature Pyramid Networks
  - Multi layers prediction
  - Each layer
    - 3 bounding boxes
- Smaller object detection

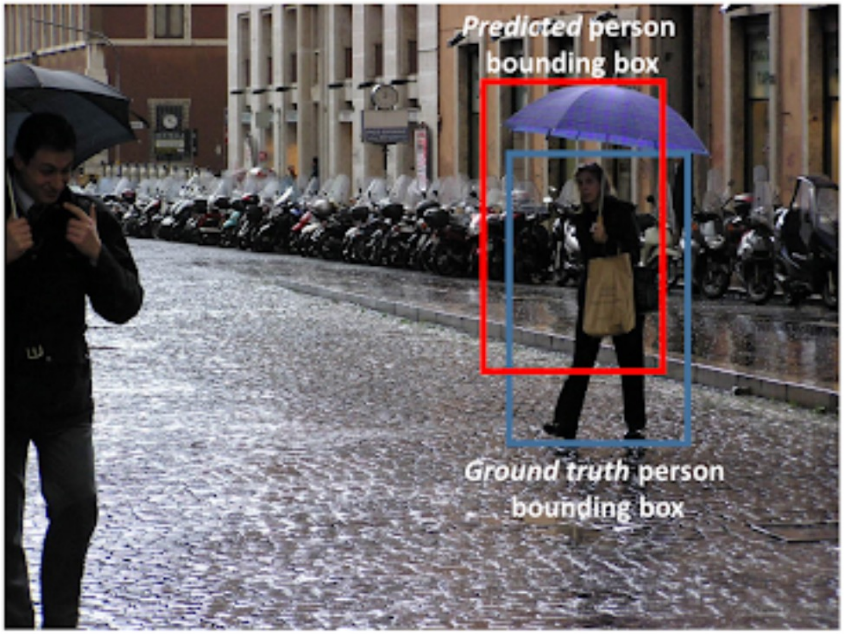


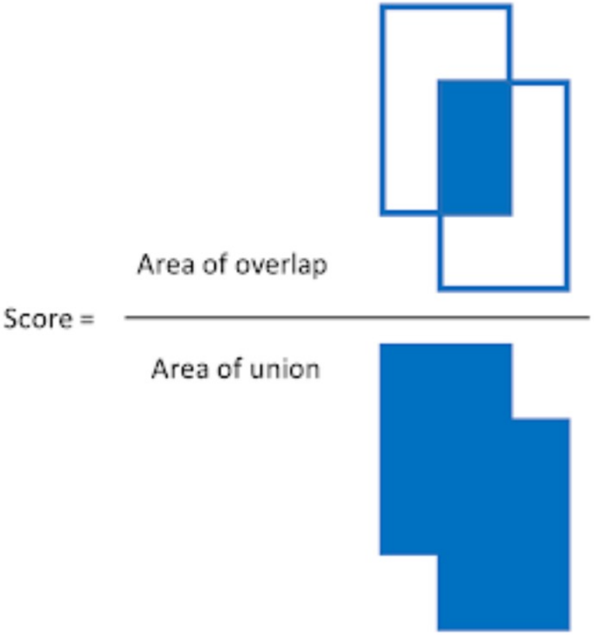
Evaluating a model

- IoU (Intersection over Union)

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$



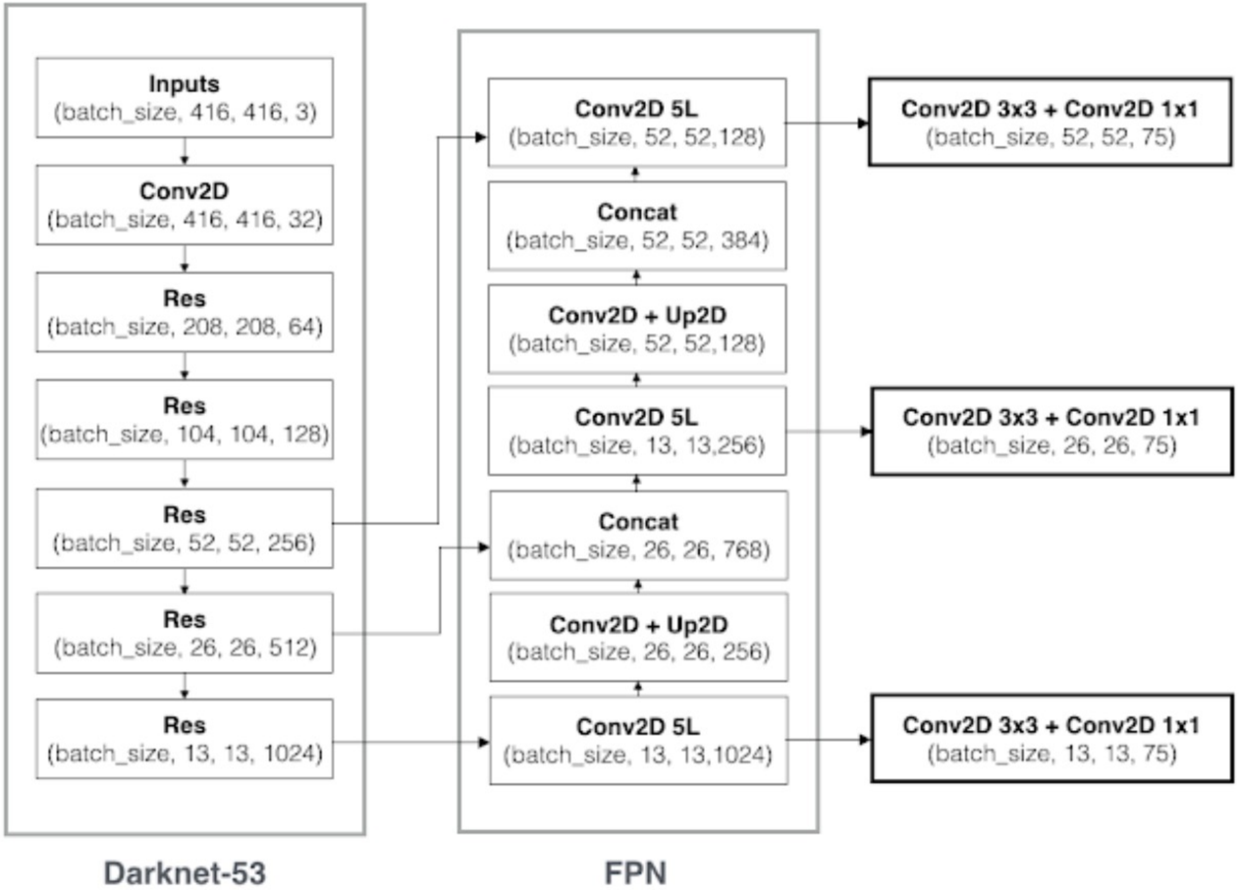

Predicted person bounding box
Ground truth person bounding box



Ref: https://people.cs.pitt.edu/~kovashka/cs1674_fa16/hw9p.html

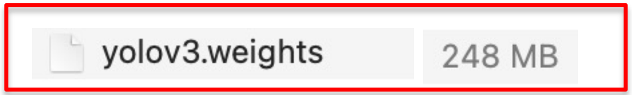
Neural Network Architecture

- Yolo v3 (2018) https://arxiv.org/abs/1804.02767
  - .cfg file https://mropengate.blogspot.com/2018/06/yolo-yolov3.html

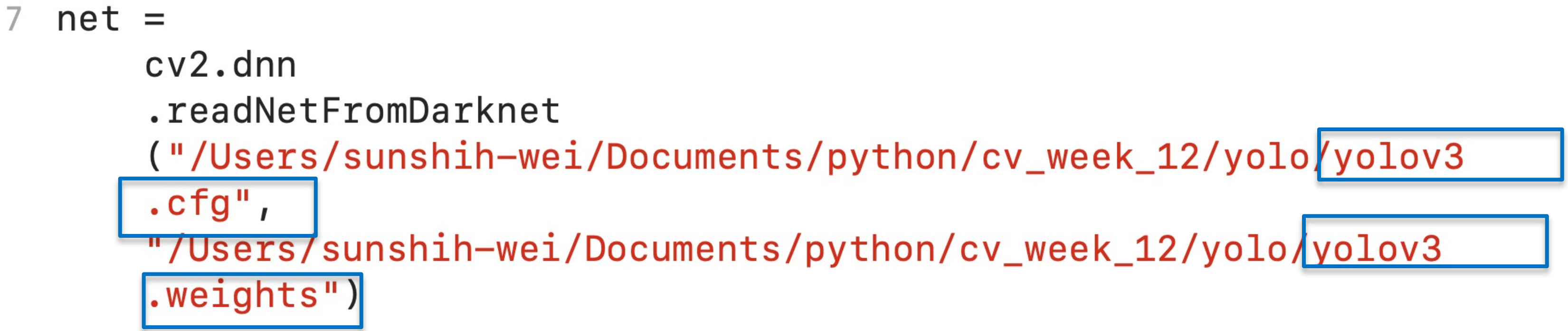


Step 1:
Loading the pre-trained model

- .cfg file
  - Defines the model architecture (neural network)
- .weights file (large)
  - Weights for the actual layers

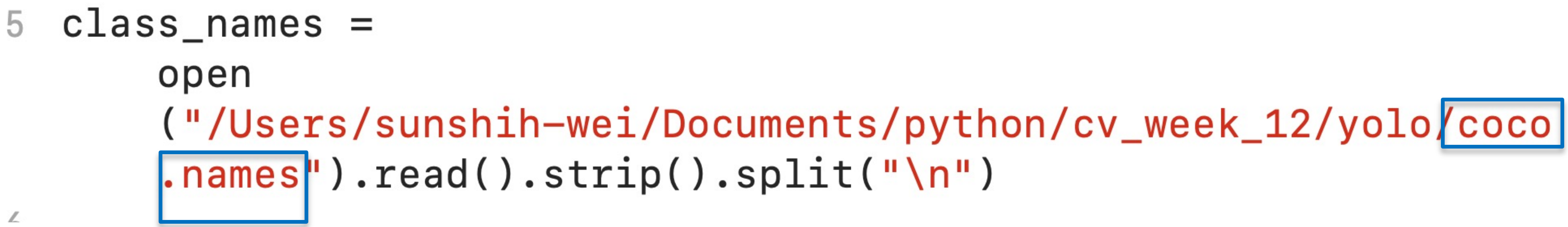


yolov3.weights 248 MB



```
net =
    cv2.dnn
    .readNetFromDarknet
    ("/Users/sunshih-wei/Documents/python/cv_week_12/yolo/yolov3
    .cfg",
    "/Users/sunshih-wei/Documents/python/cv_week_12/yolo/yolov3
    .weights")
```

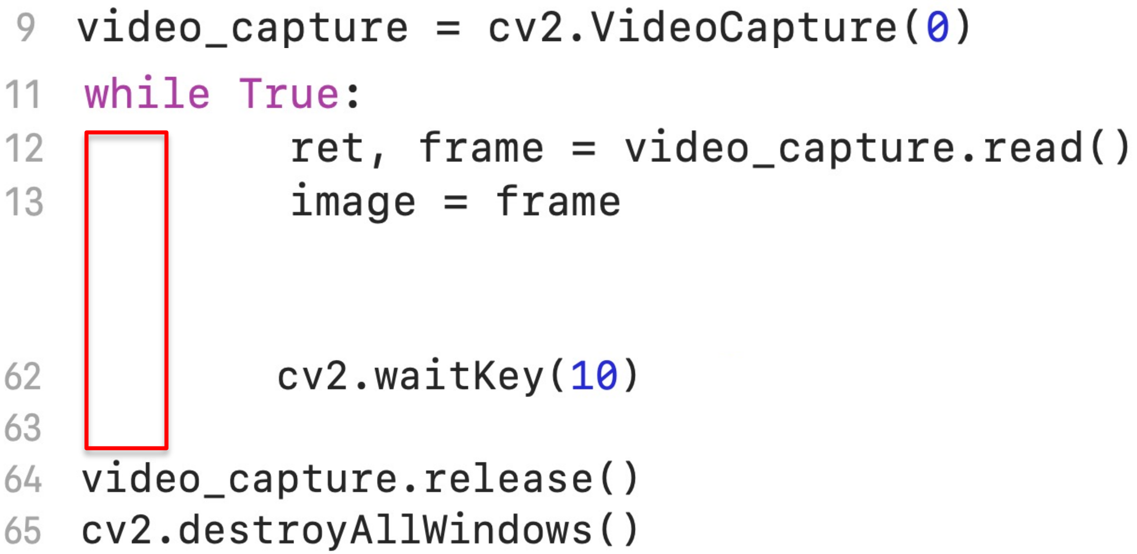
- .names file: class name file



```
class_names =
     open
     ("/Users/sunshih-wei/Documents/python/cv_week_12/yolo/coco
     .names").read().strip().split("\n")
```

Step 1-1:
Using the video camera

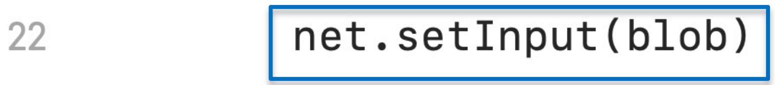
- cv2.videoCapture()



```
video_capture = cv2.VideoCapture(0)
while True:
        ret, frame = video_capture.read()
        image = frame


        cv2.waitKey(10)

video_capture.release()
cv2.destroyAllWindows()
```

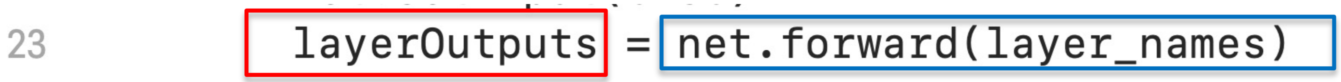
Step 2:
IO for the deep learning model

- Input to the network: blob (image)
  - net.setInput( )



```
net.setInput(blob)
```

- Output for computing the forward pass
  - net.forward( )



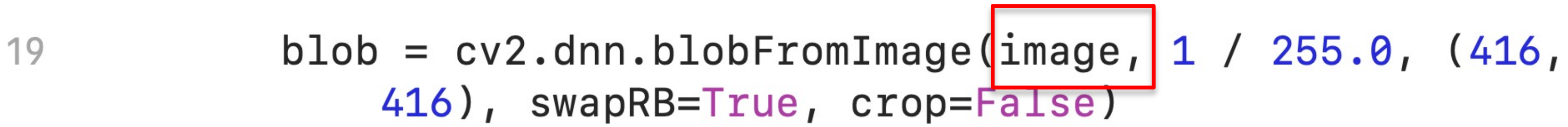
```
layerOutputs = net.forward(layer_names)
```

  - layerOutputs: an array of the detected items

Step 3, Foreground Detection: blobFromImage

- Mean subtraction (均值減法)

Original image



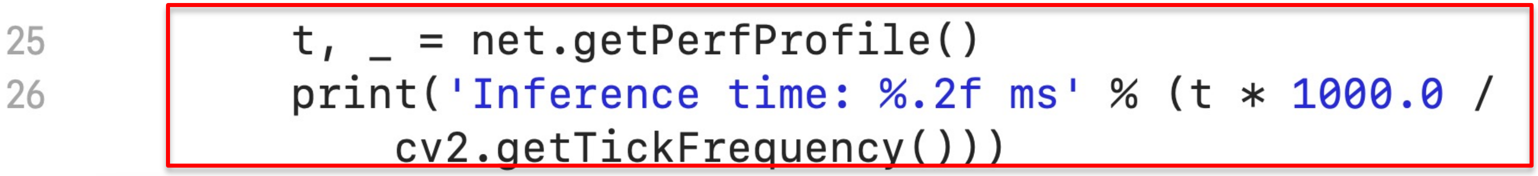
```
        blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416,
416), swapRB=True, crop=False)
```

Mean subtraction

https://www.twblogs.net/a/5e4e2905bd9eee101df43127

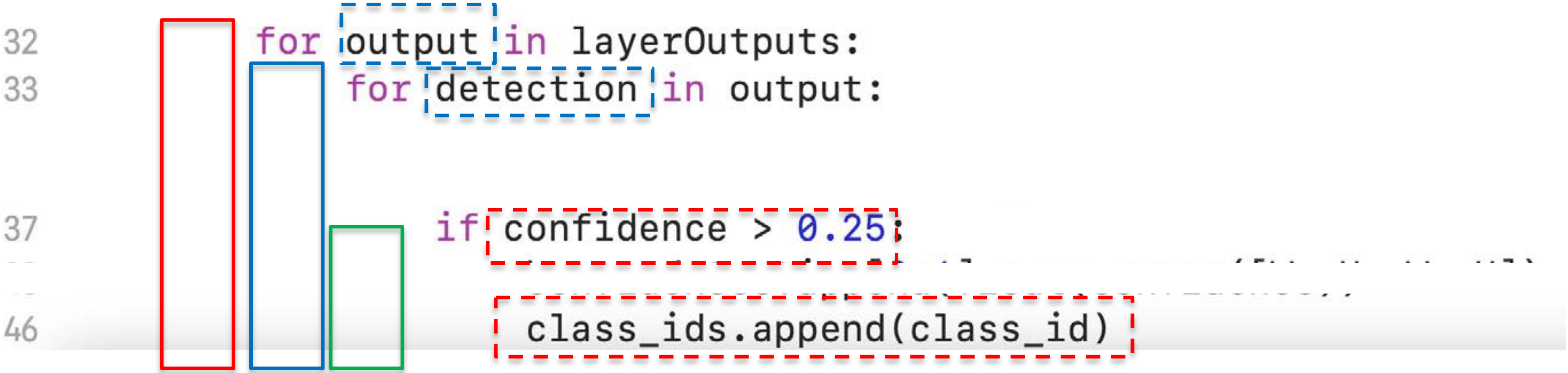
Step 4: Prediction results

- Show the inference time
  - Operation



```
    t, _ = net.getPerfProfile()
    print('Inference time: %.2f ms' % (t * 1000.0 /
        cv2.getTickFrequency()))
```

- For each detected item (output):

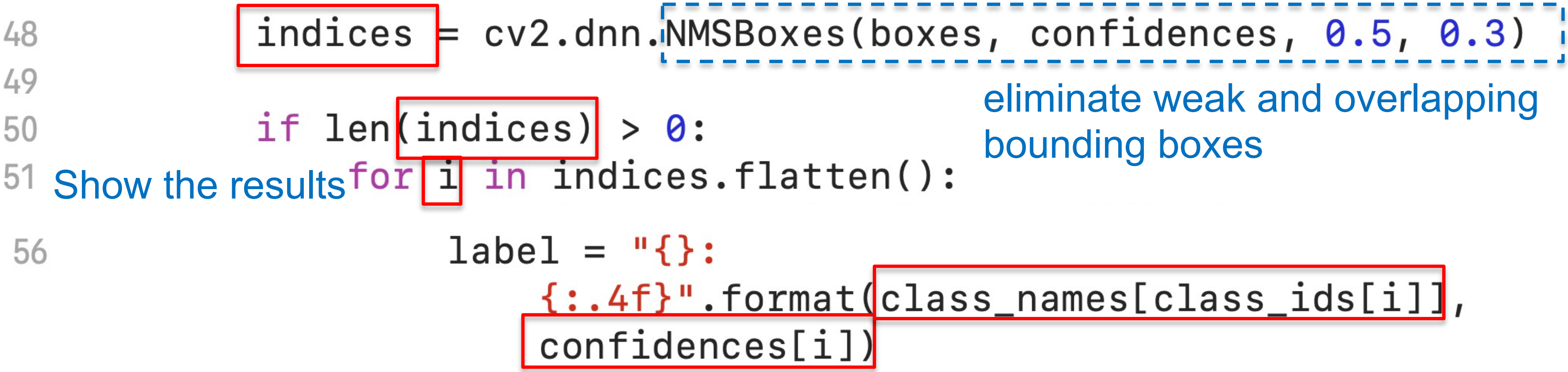


```
    for output in layerOutputs:
        for detection in output:

            if confidence > 0.25:

                class_ids.append(class_id)
```

- Get the class ID: class_id

Step 5: Each Detected Items (confidence>0.25)

- Draw the confidence and item ID

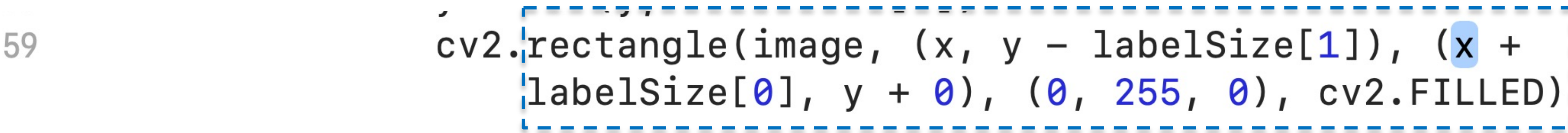


```
        indices = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.3)

        if len(indices) > 0:
            for i in indices.flatten():

                    label = "{}:
                        {:.4f}".format(class_names[class_ids[i]],
                        confidences[i])
```

eliminate weak and overlapping bounding boxes

Show the results

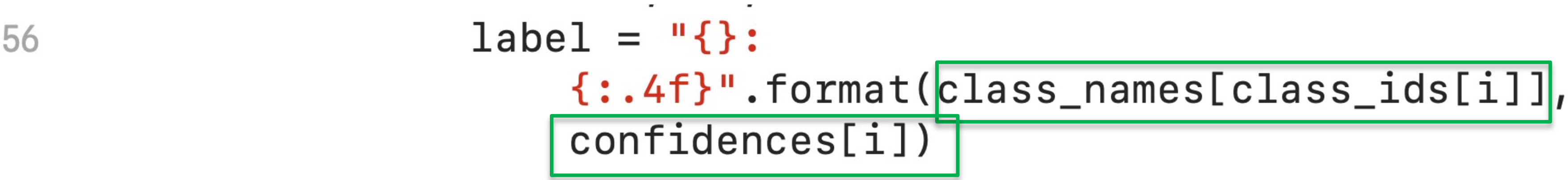
- Draw the rectangle



```
                    cv2.rectangle(image, (x, y - labelSize[1]), (x +
                        labelSize[0], y + 0), (0, 255, 0), cv2.FILLED)
```

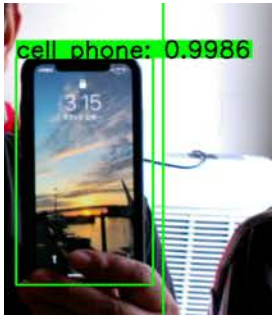
Step 6: Displaying results

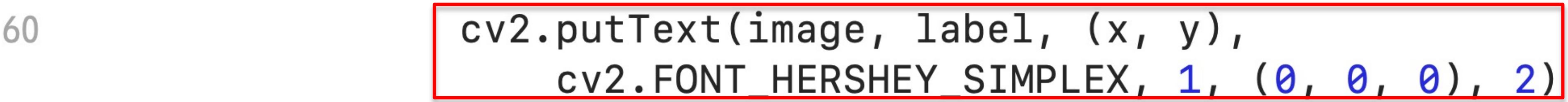
- Show the text: class name and confidence



```
label = "{}:
    {:.4f}".format(class_names[class_ids[i]],
    confidences[i])
```

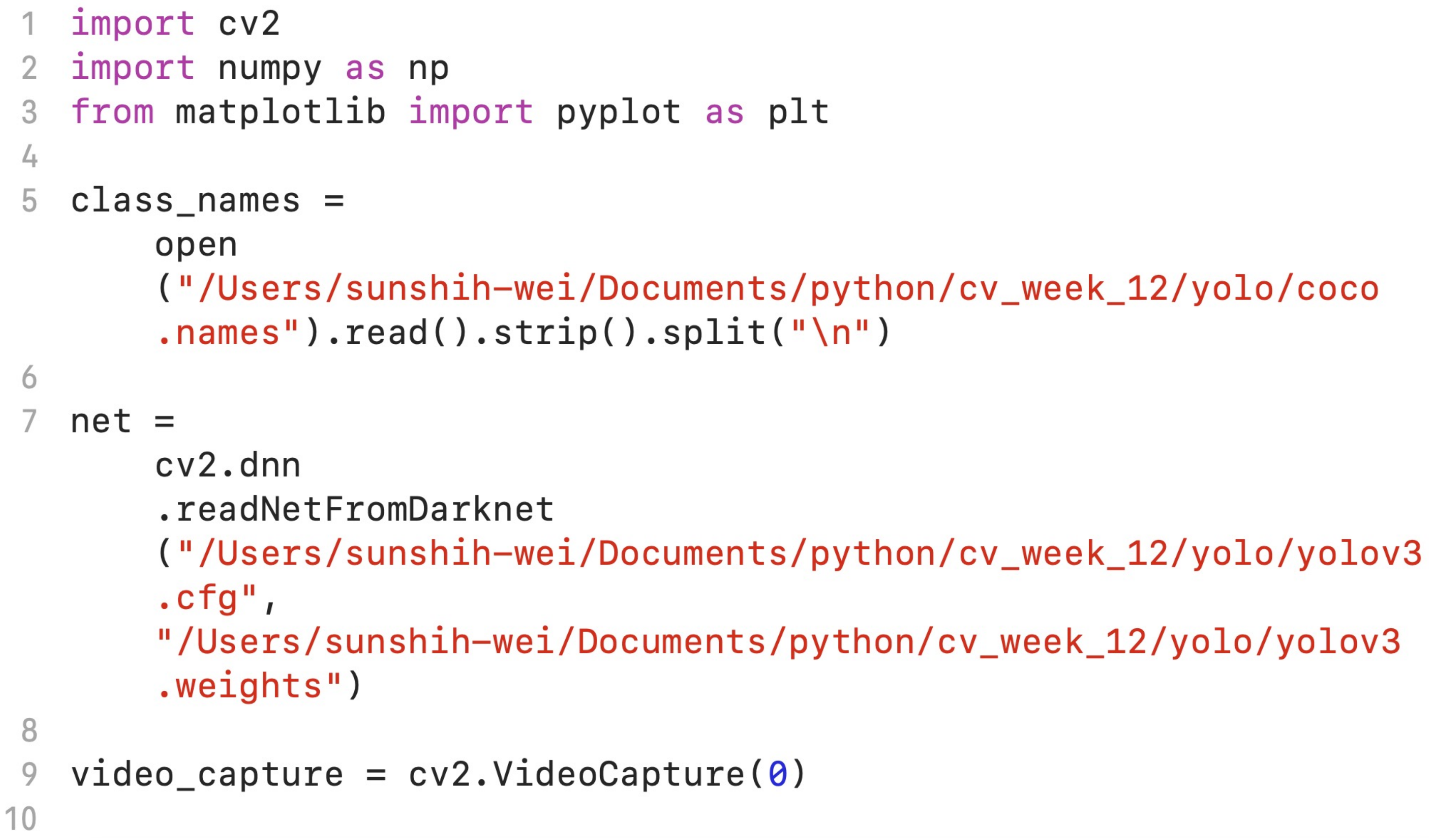
- Print the results





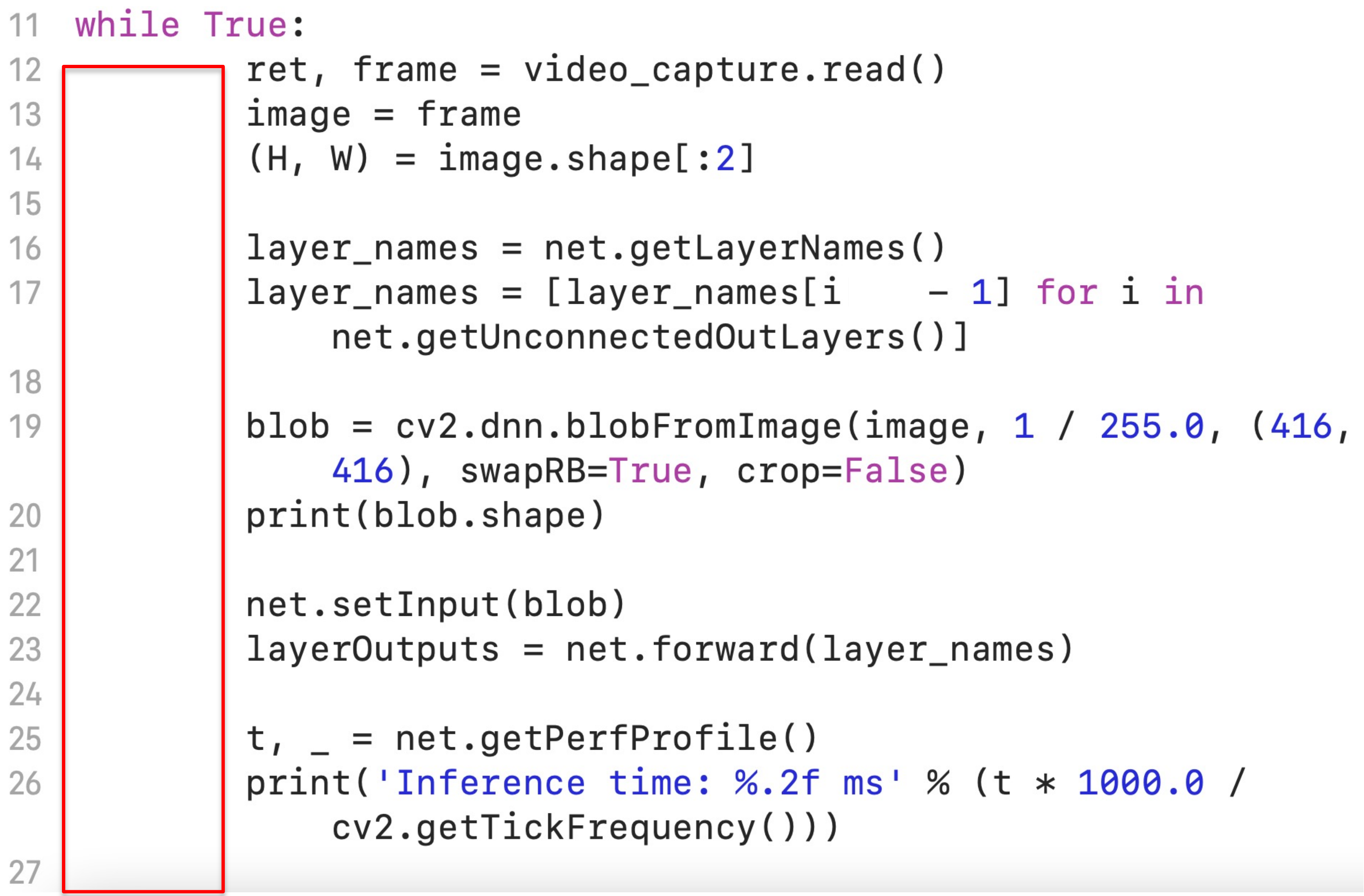
```
cv2.putText(image, label, (x, y),
    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)
```

Full code (1/5)



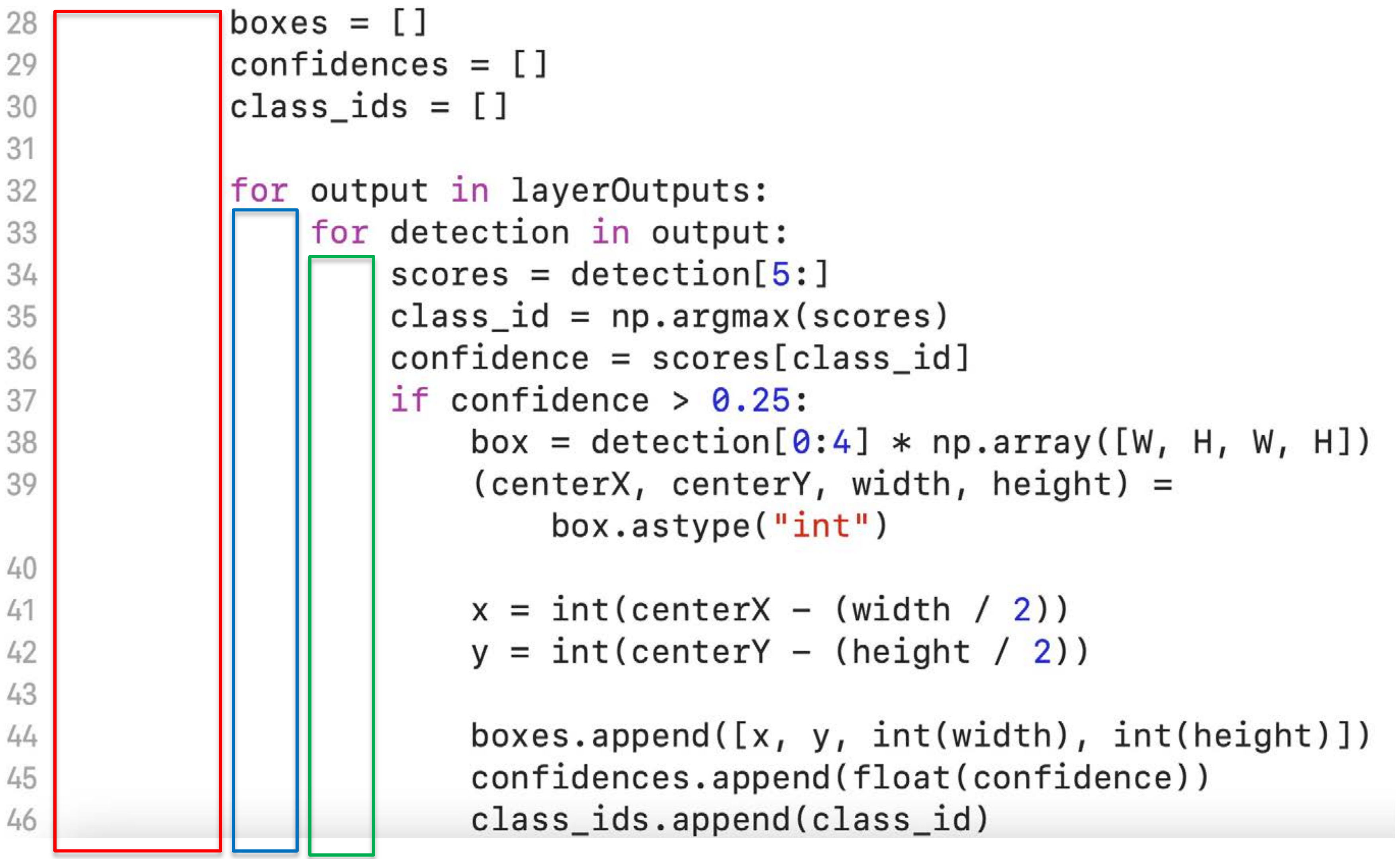
```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

class_names = open("/Users/sunshih-wei/Documents/python/cv_week_12/yolo/coco.names").read().strip().split("\n")

net = cv2.dnn.readNetFromDarknet("/Users/sunshih-wei/Documents/python/cv_week_12/yolo/yolov3.cfg", "/Users/sunshih-wei/Documents/python/cv_week_12/yolo/yolov3.weights")

video_capture = cv2.VideoCapture(0)

```

Full code (2/5)



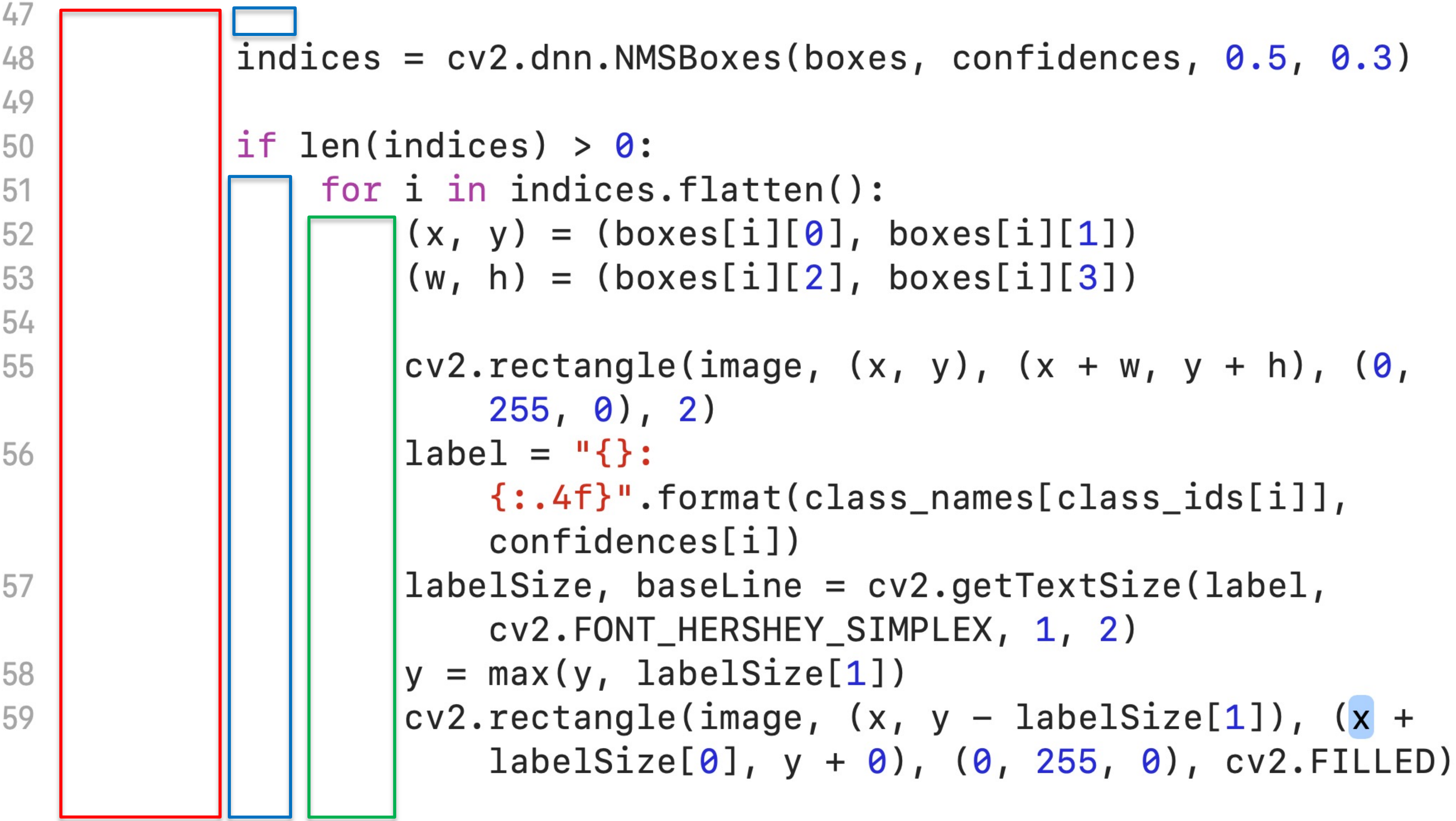
```
while True:
        ret, frame = video_capture.read()
        image = frame
        (H, W) = image.shape[:2]

        layer_names = net.getLayerNames()
        layer_names = [layer_names[i    - 1] for i in
            net.getUnconnectedOutLayers()]

        blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416,
            416), swapRB=True, crop=False)
        print(blob.shape)

        net.setInput(blob)
        layerOutputs = net.forward(layer_names)

        t, _ = net.getPerfProfile()
        print('Inference time: %.2f ms' % (t * 1000.0 /
            cv2.getTickFrequency()))
```

Full code (3/5)



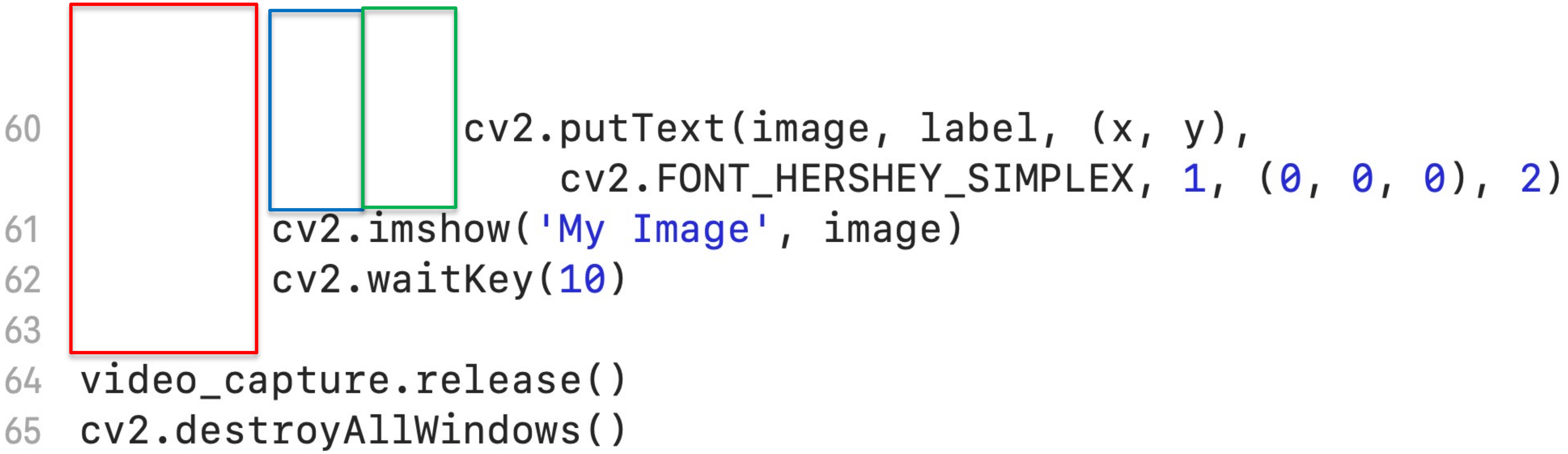
```
boxes = []
confidences = []
class_ids = []

for output in layerOutputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.25:
            box = detection[0:4] * np.array([W, H, W, H])
            (centerX, centerY, width, height) =
                box.astype("int")

            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))

            boxes.append([x, y, int(width), int(height)])
            confidences.append(float(confidence))
            class_ids.append(class_id)
```

Full code (4/5)



```
indices = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.3)

if len(indices) > 0:
    for i in indices.flatten():
        (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])

        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
        label = "{}: {:.4f}".format(class_names[class_ids[i]], confidences[i])
        labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
        y = max(y, labelSize[1])
        cv2.rectangle(image, (x, y - labelSize[1]), (x + labelSize[0], y + 0), (0, 255, 0), cv2.FILLED)
```

Full code (5/5)

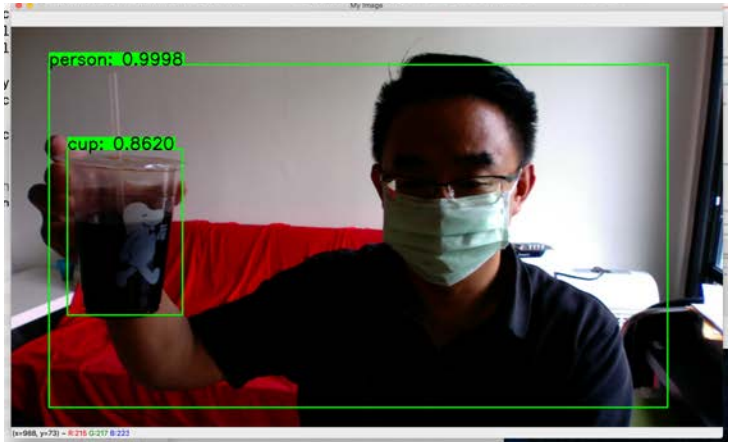


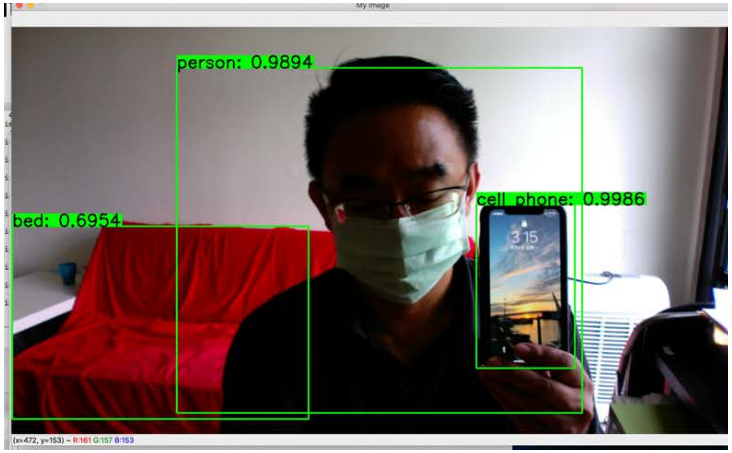
```
                cv2.putText(image, label, (x, y),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)
        cv2.imshow('My Image', image)
        cv2.waitKey(10)

video_capture.release()
cv2.destroyAllWindows()
```

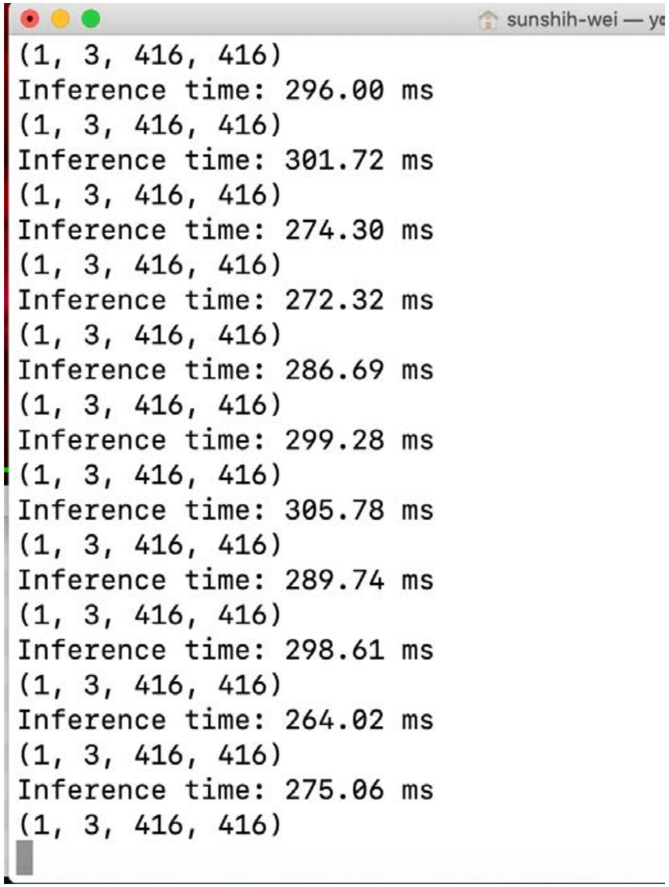
Practice 1

- Object detection results – from a webcam

Reults:





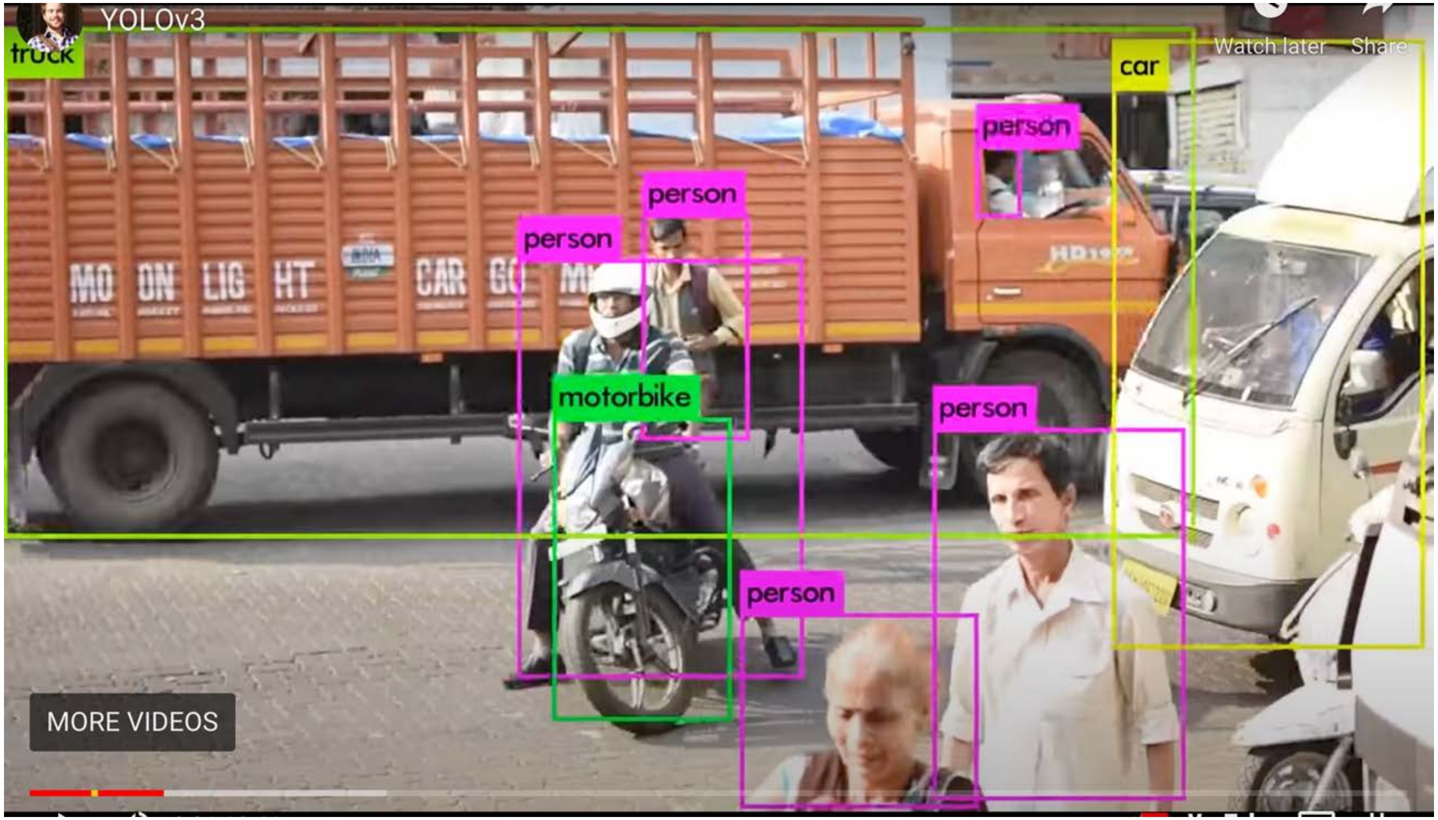


```
(1, 3, 416, 416)
Inference time: 296.00 ms
(1, 3, 416, 416)
Inference time: 301.72 ms
(1, 3, 416, 416)
Inference time: 274.30 ms
(1, 3, 416, 416)
Inference time: 272.32 ms
(1, 3, 416, 416)
Inference time: 286.69 ms
(1, 3, 416, 416)
Inference time: 299.28 ms
(1, 3, 416, 416)
Inference time: 305.78 ms
(1, 3, 416, 416)
Inference time: 289.74 ms
(1, 3, 416, 416)
Inference time: 298.61 ms
(1, 3, 416, 416)
Inference time: 264.02 ms
(1, 3, 416, 416)
Inference time: 275.06 ms
(1, 3, 416, 416)
```

Practice 2

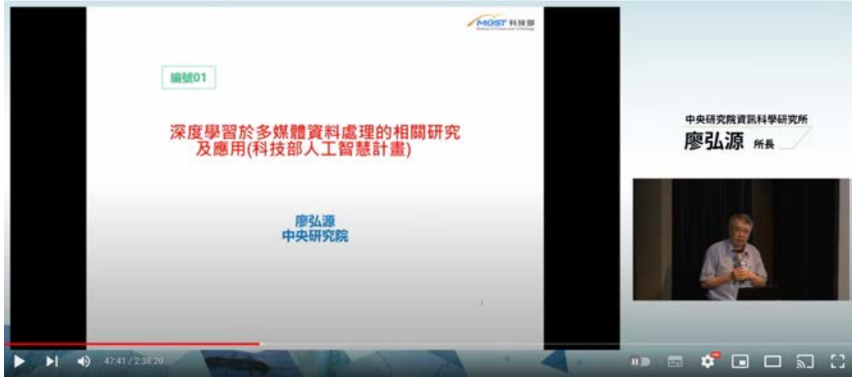
- Object detection from a still image

Reults:



Latest YOLO versions

- YOLO v4: 中研院資訊所廖弘源所長等人
  - https://www.youtube.com/watch?v=HdQqAF-rMKc



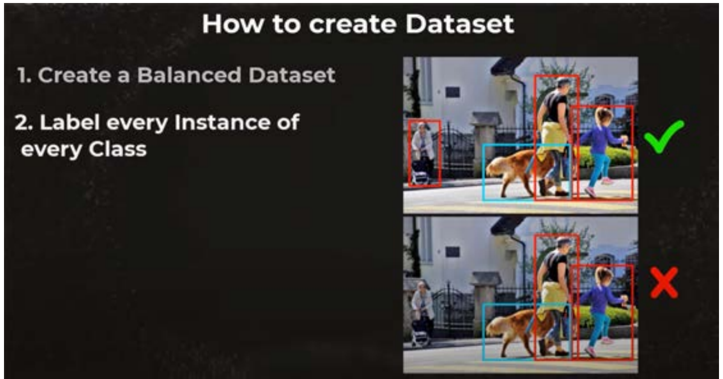
編號01

深度學習於多媒體資料處理的相關研究
及應用(科技部人工智慧計畫)

廖弘源
中央研究院

中央研究院資訊科學研究所
廖弘源 所長

- YOLO v5: [Glenn Jocher, 2020]
  - https://www.youtube.com/watch?v=wM1wn1bZ3S4



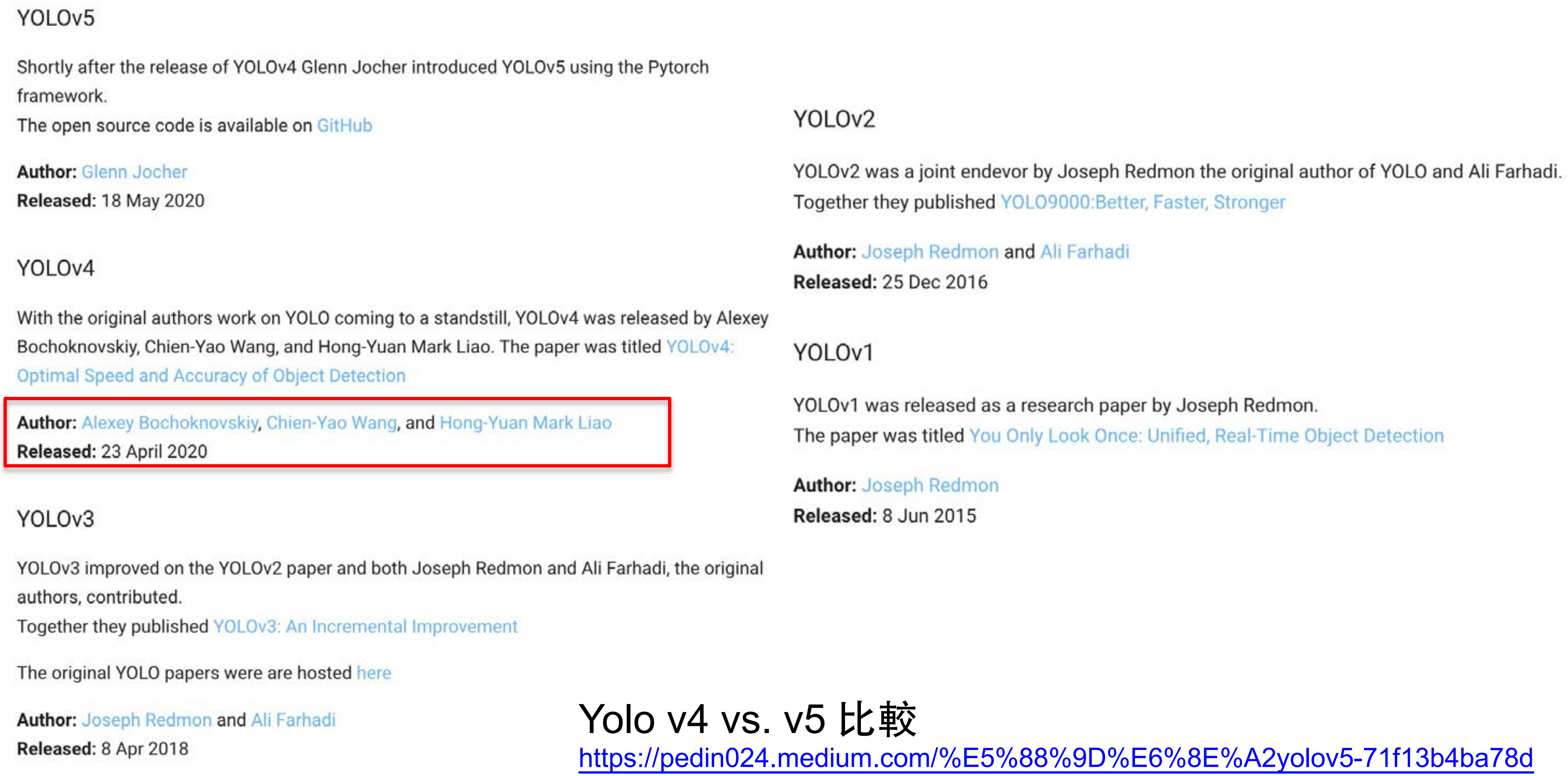
How to create Dataset

1. Create a Balanced Dataset

2. Label every Instance of every Class

History of YOLO

- https://docs.ultralytics.com/



YOLOv5

Shortly after the release of YOLOv4 Glenn Jocher introduced YOLOv5 using the Pytorch framework.
The open source code is available on GitHub

Author: Glenn Jocher
Released: 18 May 2020

YOLOv4

With the original authors work on YOLO coming to a standstill, YOLOv4 was released by Alexey Bochoknovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. The paper was titled YOLOv4: Optimal Speed and Accuracy of Object Detection

Author: Alexey Bochoknovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao
Released: 23 April 2020

YOLOv3

YOLOv3 improved on the YOLOv2 paper and both Joseph Redmon and Ali Farhadi, the original authors, contributed.
Together they published YOLOv3: An Incremental Improvement

The original YOLO papers were are hosted here

Author: Joseph Redmon and Ali Farhadi
Released: 8 Apr 2018

YOLOv2

YOLOv2 was a joint endevor by Joseph Redmon the original author of YOLO and Ali Farhadi.
Together they published YOLO9000:Better, Faster, Stronger

Author: Joseph Redmon and Ali Farhadi
Released: 25 Dec 2016

YOLOv1

YOLOv1 was released as a research paper by Joseph Redmon.
The paper was titled You Only Look Once: Unified, Real-Time Object Detection

Author: Joseph Redmon
Released: 8 Jun 2015

Yolo v4 vs. v5 比較
https://pedin024.medium.com/%E5%88%9D%E6%8E%A2yolov5-71f13b4ba78d

Textbook Reading

- Mastering OpenCV 4 with Python
  - Ch 12, Introduction to Deep Learning
    - OpenCV deep learning Classification
      - YOLO for object detection
        - p. 388 -p. 390