國立雲林科技大學電子工程系

人工智慧深度學習

Lab6
EfficientDet

第十二組

指導教授: 夏世昌 教授

王斯弘 助理教授

組員:四電子三A B10913014 林廷緯

四電子三B B10913158 陳維翔

四電子三B B10913154 曹宸維

中華民國111年12月8日

一、 題目

二、 基本介紹

1. EfficientDet 介紹

2. EfficientDet 實作方式

3. EdgeAI 平台介紹

4. EdgeAI 實作方式

5. 輕量化 tflite 轉換

三、 程式說明

1. EfficientDet 模型及訓練結果

2. EfficientDet 模型推理結果

四、 EdgeAI 平台驗證結果

五、 心得與討論

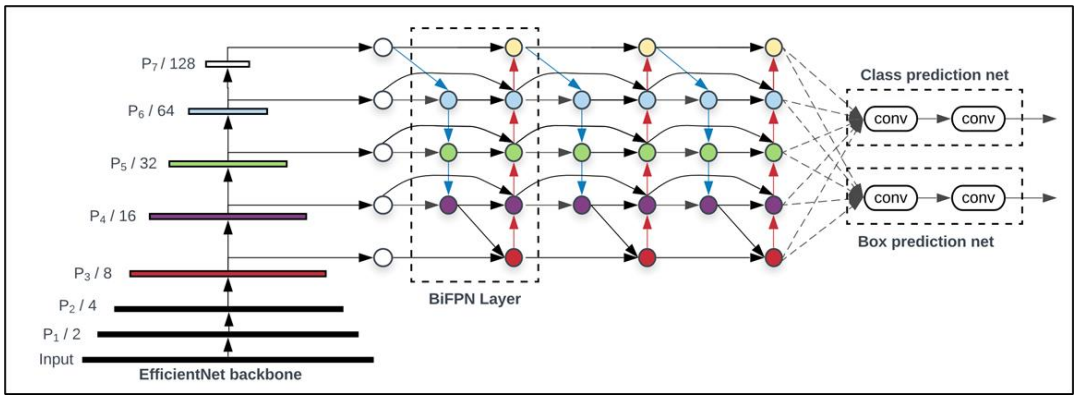
一、 題目

EfficientDet 架構模型訓練及 EdgeAI 平台驗證

二、 基本介紹

1. EfficientDet 介紹

EfficientDet 以 EfficientNet 為預訓練之骨幹網路為基礎下，用複雜的雙向融合 BiFPN 改進了 PANet，BiFPN 在架構上進行了改善，刪除了只有一個輸入或輸出邊對於整體融合特徵網路作用不大的節點以簡化雙向網路，如果輸入與輸出節點在同一層，架構會在節點之間新增額外的邊以在不增加太多成本的情況融合更多特徵，並且處理每個雙向路徑作為一個特徵網路層多次使用，最後加上帶權重的特徵融合機制，能以較低 FLOPs 的情況下維持相同或較好的準確率。



2. EfficientDet 實作方式

將 EfficientNet 拿來做 backbone，這樣從 EfficientNet B0~B6，就可以控制 Backbone 的規模;neck 部分,BiFPN 的 channel 數量、重複的 layer 數量也可以控制；此外還有 head 部分的層數，以及輸入圖片的分辨率,這些組成了 EfficientDet 的 scaling config。

Model	mAP	#Params	Ratio	#FLOPS	Ratio	GPU LAT(ms)	Speedup	CPU LAT(s)	Speedup
EfficientDet-D0	**32.4**	**3.9M**	**1x**	**2.5B**	**1x**	**16** ±1.6	**1x**	**0.32** ±0.002	**1x**
YOLOv3 [26]	33.0	-	-	71B	28x	51†	-	-	-
EfficientDet-D1	**38.3**	**6.6M**	**1x**	**6B**	**1x**	**20** ±1.1	**1x**	**0.74** ±0.003	**1x**
MaskRCNN [8]	37.9	44.4M	6.7x	149B	25x	92†	-	-	-
RetinaNet-R50 (640) [17]	37.0	34.0M	6.7x	97B	16x	27 ±1.1	1.4x	2.8 ±0.017	3.8x
RetinaNet-R101 (640) [17]	37.9	53.0M	8x	127B	21x	34 ±0.5	1.7x	3.6 ±0.012	4.9x
EfficientDet-D2	**41.1**	**8.1M**	**1x**	**11B**	**1x**	**24** ±0.5	**1x**	**1.2** ±0.003	**1x**
RetinaNet-R50 (1024) [17]	40.1	34.0M	4.3x	248B	23x	51 ±0.9	2.0x	7.5 ±0.006	6.3x
RetinaNet-R101 (1024) [17]	41.1	53.0M	6.6x	326B	30x	65 ±0.4	2.7x	9.7 ±0.038	8.1x
NAS-FPN R-50 (640) [5]	39.9	60.3M	7.5x	141B	13x	41 ±0.6	1.7x	4.1 ±0.027	3.4x
EfficientDet-D3	**44.3**	**12.0M**	**1x**	**25B**	**1x**	**42** ±0.8	**1x**	**2.5** ±0.002	**1x**
NAS-FPN R-50 (1024) [5]	44.2	60.3M	5.1x	360B	15x	79 ±0.3	1.9x	11 ±0.063	4.4x
NAS-FPN R-50 (1280) [5]	44.8	60.3M	5.1x	563B	23x	119 ±0.9	2.8x	17 ±0.150	6.8x
EfficientDet-D4	**46.6**	**20.7M**	**1x**	**55B**	**1x**	**74** ±0.5	**1x**	**4.8** ±0.003	**1x**
NAS-FPN R50 (1280@384)	45.4	104 M	5.1x	1043B	19x	173 ±0.7	2.3x	27 ±0.056	5.6x
EfficientDet-D5 + AA	**49.8**	**33.7M**	**1x**	**136B**	**1x**	**141** ±2.1	**1x**	**11** ±0.002	**1x**
AmoebaNet+ NAS-FPN + AA(1280) [37]	48.6	185M	5.5x	1317B	9.7x	259 ±1.2	1.8x	38 ±0.084	3.5x
EfficientDet-D6 + AA	**50.6**	**51.9M**	**1x**	**227B**	**1x**	**190** ±1.1	**1x**	**16**±0.003	**1x**
AmoebaNet+ NAS-FPN + AA(1536) [37]	50.7	209M	4.0x	3045B	13x	608 ±1.4	3.2x	83 ±0.092	5.2x
EfficientDet-D7 + AA	**51.0**	**51.9M**	**1x**	**326B**	**1x**	**262** ±2.2	**1x**	**24** ±0.003	**1x**

We omit ensemble and test-time multi-scale results [23, 7].
†Latency numbers marked with † are from papers, and all others are measured on the same machine.

3. EdgeAI 平台介紹

EdgeAI 平台採用樹莓派 3B+，搭配 Google TPU，而將 AI 模型移動至樹莓派時，需使用 TensorFlowLite 檔，壓縮模型大小，以減少樹莓派空間的浪費並加快讀取速度。

4. EdgeAI 實作方式

下載 win32 燒入器，至樹梅派官網下載作業系統，在 win32 燒入器上選擇載下來的作業系統壓縮包，插上磁碟卡並燒入作業系統。

將記憶卡插入樹莓派並開機，經過簡單的初始設定後開啟 Terminal，並依序輸入以下指令，安裝相關套件。

```
1 echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee /etc/apt/sources.list.d/coral-edgetpu.list
2 curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
3 sudo apt-get update
4 sudo apt-get install libedgetpu1-std
5 sudo apt-get install python3-edgetpu
6 sudo apt-get install edgetpu-examples
7 sudo pip3 install jupyterlab opencv-python matplotlib
8 sudo apt-get install python3-pycoral
```

建好環境後，將「tflite 模型」、「label」、「測試圖片」、「py 執行檔」複製到樹莓派後，並下達以下指令，等待一段時間後即可看到影像辨識的結果。

```
python3 py執行檔 --model tflite模型 --labels label --input 測試圖片
```

5. 輕量化 tflite 轉換

利用虛擬機安裝 ubuntu 系統，下達以下指令來建置環境:

```
1  echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee /etc/apt/sources.list.d/coral-edgetpu.list
2  curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
3  sudo apt-get update
4  sudo apt-get install edgetpu-compiler
```

將 h5 檔案轉換成 tflite 檔案並傳輸至虛擬機中，執行以下指令來進行壓縮優化:

```
edgetpu_compiler tflite 模型
```

最後再將優化後的模型複製到樹梅派即可。

三、 程式說明

1. EfficientDet 模型及訓練結果

匯入套件

```
import numpy as np
import os
import pycocotools
import tqdm as notebook_tqdm
import tensorflow as tf
assert tf.__version__.startswith('2')

import tflite_model_maker

from tflite_model_maker.config import ExportFormat
from tflite_model_maker import model_spec
from tflite_model_maker import object_detector

tf.get_logger().setLevel('ERROR')
from absl import logging
logging.set_verbosity(logging.ERROR)

print(tf.__version__)
print(tflite_model_maker.__version__)
```

設定資料夾路徑

```
images_in = './oimage/'
annotations_in = './xml/'
```

建立/設定資料集

```
label_map= ["M", "K", "S", "bottom"]
train_images_dir = './new_split-dataset_2/train/images/'
train_annotations_dir = './new_split-dataset_2/train/annotations'
val_images_dir = './new_split-dataset_2/validation/images/'
val_annotations_dir = './new_split-dataset_2/validation/annotations'
test_images_dir = './new_split-dataset_2/test/images/'
test_annotations_dir = './new_split-dataset_2/test/annotations'
```

```
train_data = object_detector.DataLoader.from_pascal_voc(
      train_images_dir, train_annotations_dir, label_map=label_map, cache_dir="./cache_data_ta/train", num_shards=1, max_num_images=3000)

validation_data = object_detector.DataLoader.from_pascal_voc(
      val_images_dir, val_annotations_dir, label_map=label_map, cache_dir="./cache_data_ta/validation",num_shards=1, max_num_images=1000)

test_data = object_detector.DataLoader.from_pascal_voc(
     test_images_dir, test_annotations_dir, label_map=label_map, cache_dir="./cache_data_ta/test",num_shards=1, max_num_images=1000)
```

```
train_data = object_detector.DataLoader.from_cache('./cache_data_ta/train/4bcdd345de559c097bc3f19ecc95ac13')
validation_data = object_detector.DataLoader.from_cache('./cache_data_ta/validation/4bcdd345de559c097bc3f19ecc95ac13')
test_data = object_detector.DataLoader.from_cache('./cache_data_ta/test/4bcdd345de559c097bc3f19ecc95ac13')
```

定義 Model

```
[ ]    spec = object_detector.EfficientDetLite0Spec()

model = object_detector.create(train_data=train_data,
                               model_spec=spec,
                               validation_data=validation_data,
                               epochs=200,
                               batch_size=16,
                               train_whole_model=True,
                               do_train=True)
```

儲存結果

```
TFLITE_FILENAME = 'efficientdet.tflite'
LABELS_FILENAME = 'labels.txt'
SAVED_FILENAME = 'model'
model.export(export_dir='./export', tflite_filename=TFLITE_FILENAME, label_filename=LABELS_FILENAME, saved_model_filename=SAVED_FILENAME,
             export_format=[ExportFormat.TFLITE, ExportFormat.LABEL, ExportFormat.SAVED_MODEL])
```

2. EfficientDet 模型推理結果

匯入套件

```
import tensorflow as tf
import cv2 as cv
import numpy as np
import glob
```

載入模型檔案

```
tf.compat.v1.reset_default_graph()
interpreter = tf.lite.Interpreter('/content/drive/MyDrive/Colab Notebooks/efficientdet/export/efficientdet.tflite')
interpreter.allocate_tensors()                                                    # 初始化模型
```

讀取測試圖片

```
w,h,c=input_details[0]['shape'][1:4]                    # 獲取輸入圖片大小
img_file='/content/drive/MyDrive/Colab Notebooks/efficientdet/20201110_222144_002.jpg' # 一個圖片路徑
imgc = 0
Pred = 0


img=cv.imread(img_file)                                  # 讀取圖片
img=img.astype(np.uint8)                                 # 轉整數
img=cv.resize(img,(w,h))                                 # 調整圖片大小
img=cv.cvtColor(img,cv.COLOR_BGR2RGB)                    # BGR to RGB
img=img.reshape((w,h,c))                                 # 調整陣列大小
img=np.expand_dims(img,0)                                # 轉 4D 陣列
```

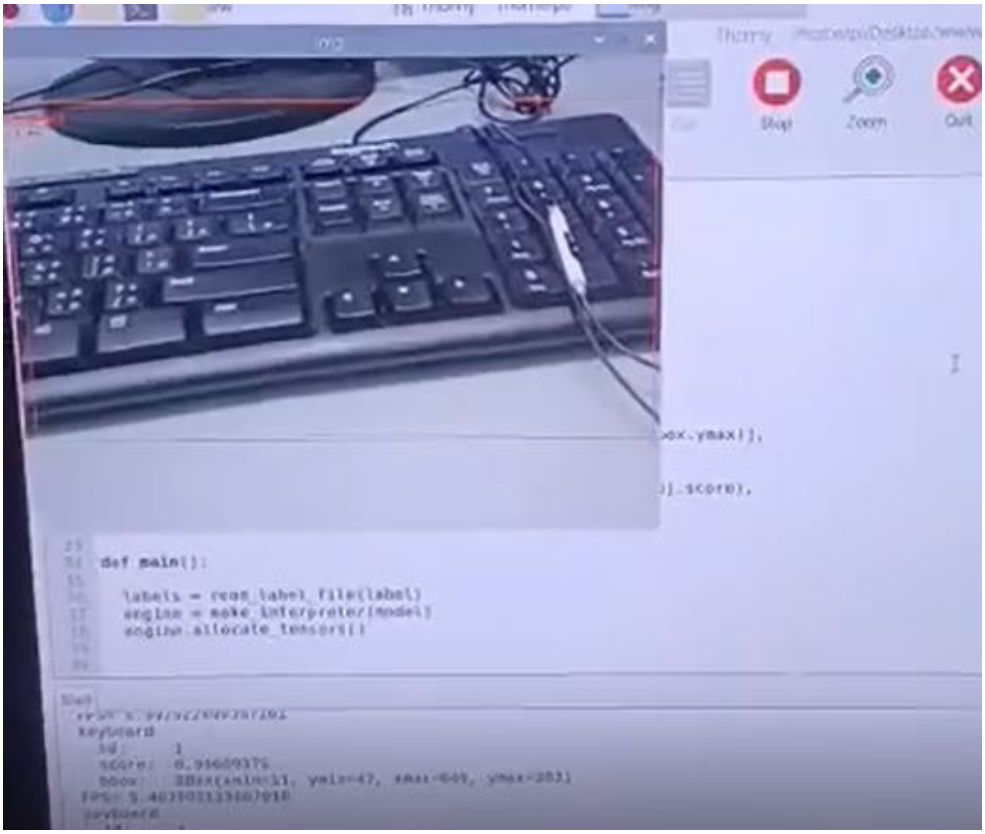
```
interpreter.set_tensor(input_details[0]['index'],  img)  #  輸入資料到模型
interpreter.invoke()#  模型計算輸出

detection_scores  =  interpreter.get_tensor(output_details[0]['index'])  #  獲取輸出各框的得分
detection_boxes  =  interpreter.get_tensor(output_details[1]['index'])  #  獲取輸出框座標  (百分比)
num_boxes  =  interpreter.get_tensor(output_details[2]['index'])    #  獲取輸出總框數量
detection_classes  =  interpreter.get_tensor(output_details[3]['index'])  #  輸出框類別
```

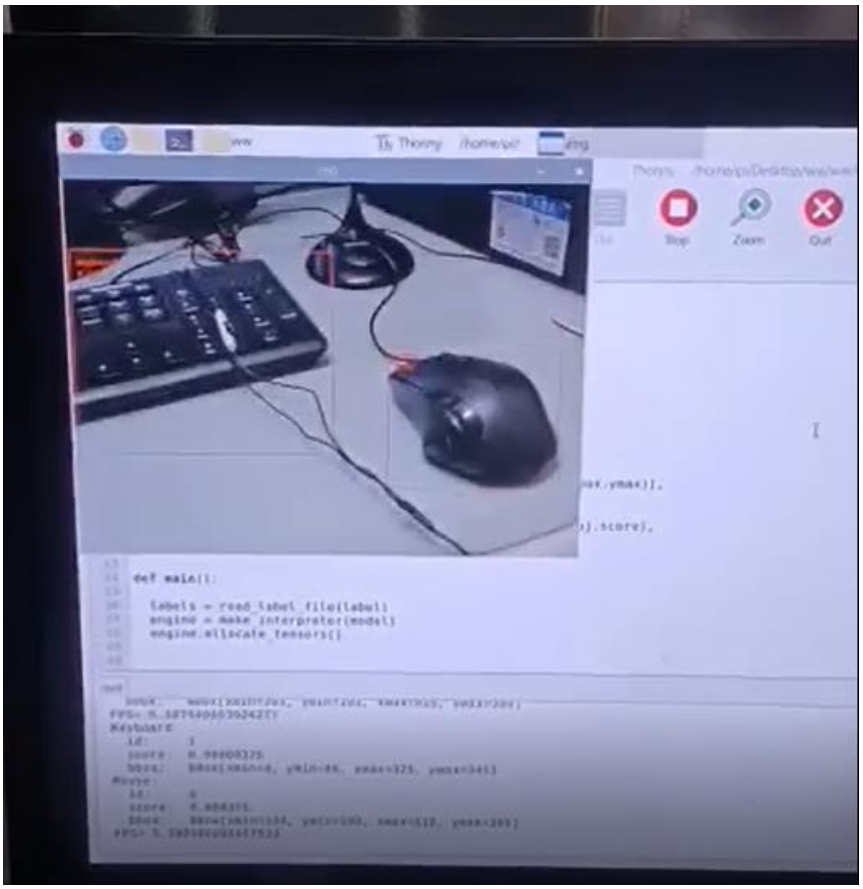
```
num_boxes=int(num_boxes[0])    #  取得整數值
detection_boxes=detection_boxes.reshape(num_boxes,4)    #  調整陣列為10x4大小
detection_classes=detection_classes.astype(dtype=np.int).reshape(num_boxes)+1  #  類別為整數值  (+1跳過背景類)
detection_scores=detection_scores.reshape(num_boxes)    #  調整陣列為10大小
#print(detection_boxes)                    #框座標
#print(detection_classes)                  #各個框類別
#print(detection_scores)                    #框分數
#print(num_boxes)                          #總共的框
#  讀取圖片並畫框
```

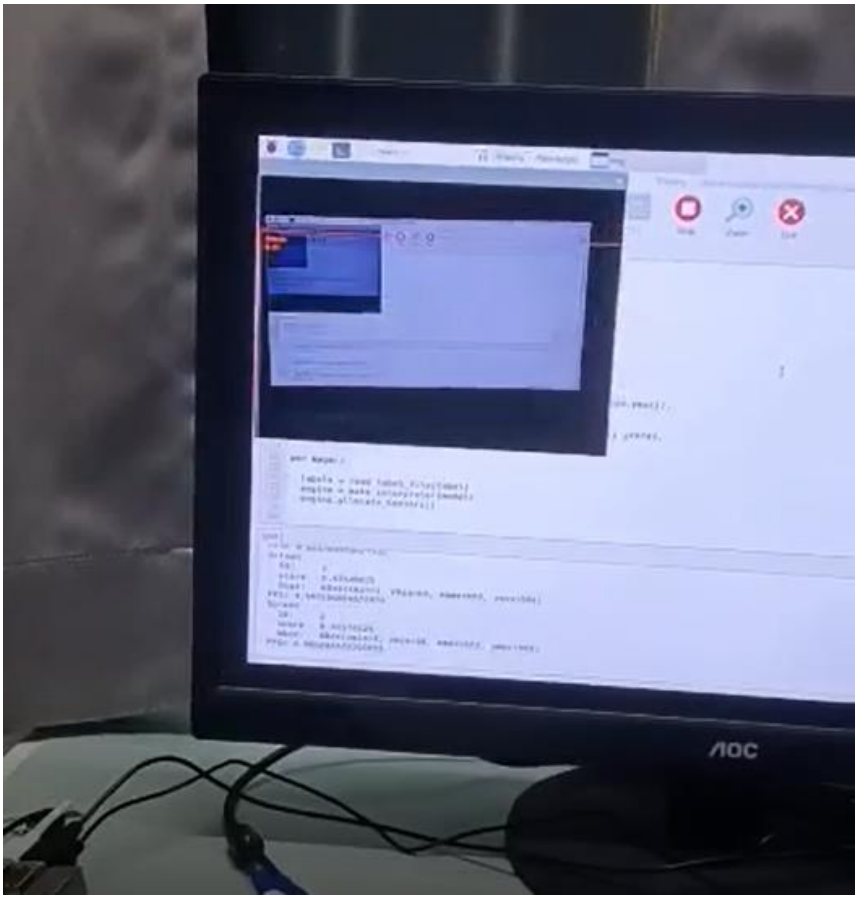
判斷類別

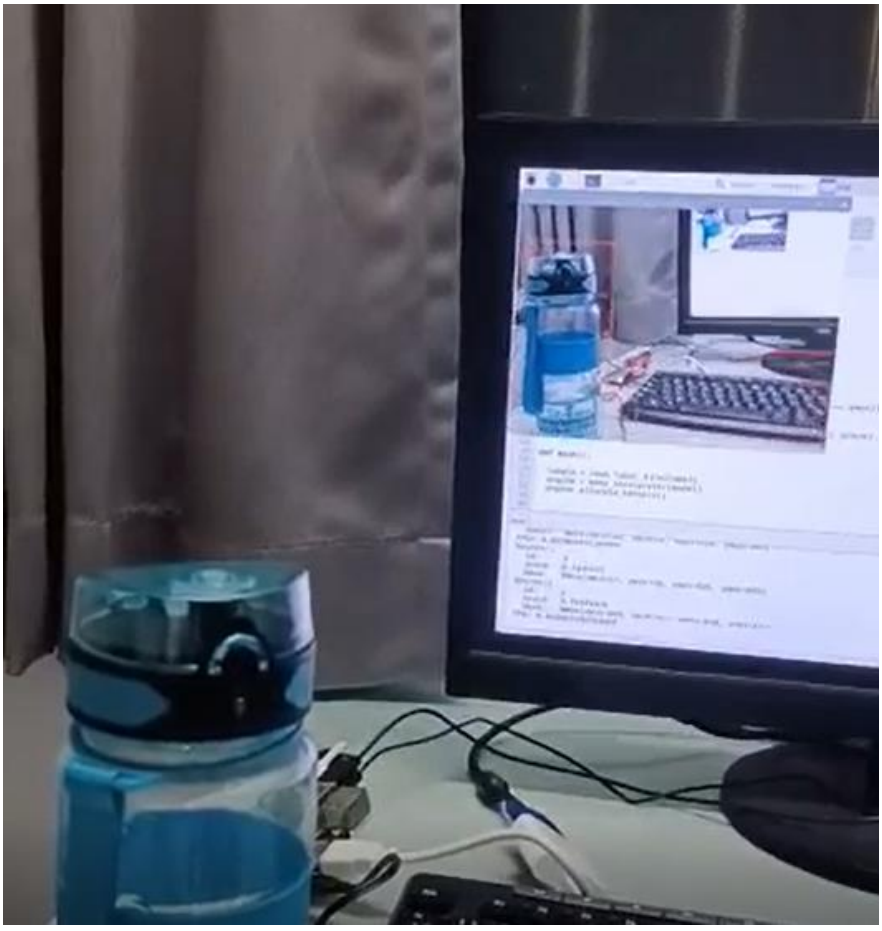
```
if  detection_scores[i]>0.5:
      outimg=cv.rectangle(outimg,  (int(detection_boxes[i,1]),  int(detection_boxes[i,0])),  (int(detection_boxes[i,3]),  int(detection_boxes[i,2])),  (0,0,255),2)
      if  detection_classes[i]  ==1:
            ID  =  'Mouse'
      elif  detection_classes[i]  ==2:
            ID  =  'Keyboard'
      elif  detection_classes[i]  ==3:
            ID  =  'Screen'
      outimg=cv.putText(outimg,'ID:  %s'%(ID),  (int(detection_boxes[i,1]),int(detection_boxes[i,0]-3)),cv.FONT_HERSHEY_COMPLEX    ,1.5,(0,0,0),2)
```

四、 EdgeAI 平台驗證結果






AOC



五、 心得與討論

陳維翔:

這次的 lab 遇到了很多問題，中間因為發現圖片沒有 resize 過，結果要在 resize 過後再做一遍，結果直接拖到下課，還要回家把它做完。

曹宸維:

這次學的是 EfficientDet，之前沒有接觸過，對我來說是一個很新鮮的東西，他能夠將拍到的東西分類、追蹤物品並用紅框圈起來，在以後的專題也許會用到。

林廷緯:

這次的 lab 花的時間很久，把拍到的圖片上傳到雲端就花了半小時，在圈圖片時手指差點抽筋，在做的時候發現沒有 resize 過要重用，又要再圈一遍，但最後還是順利完成了。