113-1 雲端運算與邊緣運算應用

End computing and edge computing applications.

Lab4

Effcientdet

Inference and application

授課老師： 王斯弘 老師

學　　生： B11123206 陳冠欣

中 華 民 國 １１３ 年 １２ 月 ２５ 日

目錄

壹、 efficientDet 簡介

EfficientDet 是 Google 團隊於 2020 年提出的一種高效物件偵測模型，基於 EfficientNet 的設計理念進一步優化，旨在達成準確率與運算效能的平衡。它在多個物件偵測基準（如 COCO）中表現出色，且具備高效率與靈活性，適合資源受限的環境

EfficientNet Backbone

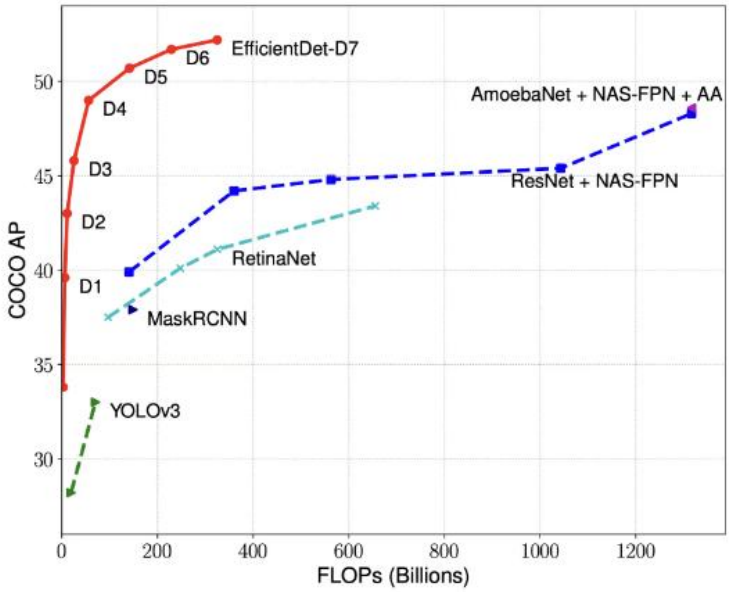
- EfficientDet 採用了 EfficientNet 作為主幹網路（backbone），用於提取影像的特徵。
- EfficientNet 的特點是基於 NAS（神經架構搜索）與 Compound Scaling 設計，能夠在計算成本和準確率之間找到最佳平衡。

BiFPN (Bidirectional Feature Pyramid Network)

- 傳統的 FPN（Feature Pyramid Network）在處理多尺度特徵時，可能會因信息流單向而限制表現。
- BiFPN 通過引入加權雙向特徵融合機制，能有效地在不同層級的特徵間傳遞信息，進一步提高偵測準確性。

Compound Scaling

- EfficientDet 使用一種統一的 Compound Scaling 方法，同時調整網路的深度、寬度與解析度，確保不同規模的模型（如 D0 到 D7）都能高效運行。
- 不同版本的模型適應不同的硬體資源，例如 D0 適合移動設備，而 D7 則針對高效能 GPU。



貳、 PC 端 模型訓練&推理

1. 安裝必要套件

安裝必要套件

```
[ ] %%shell
    eval "$(conda shell.bash hook)"
    conda activate myenv
    pip install tflite-model-maker
```

顯示隱藏的輸出內容

```
[ ] %%shell
    eval "$(conda shell.bash hook)"
    conda activate myenv
    pip install ipykernel
```

顯示隱藏的輸出內容

```
[ ]
    %%shell
    eval "$(conda shell.bash hook)"
    conda activate myenv
    pip install numpy==1.23.4
```

顯示隱藏的輸出內容

```
[ ]
    %%shell
    eval "$(conda shell.bash hook)"
    conda activate myenv
    pip install pycocotools
```

顯示隱藏的輸出內容

```
[ ] %%shell
    eval "$(conda shell.bash hook)"
    conda activate myenv
    pip uninstall tensorflow
```

顯示隱藏的輸出內容

```
[ ] %%shell
    eval "$(conda shell.bash hook)"
    conda activate myenv
    pip install tensorflow-gpu==2.8.4
```

顯示隱藏的輸出內容

2. 設定資料集路徑

```
23 images_in = 'efficientdet_dataset/oimage'
24 annotations_in = 'efficientdet_dataset/xml'
25 split_out_dir = "efficientdet_split_dataset"
26
```

3. 進行資料處理，資料夾需增加自己的資料集，資料集每個類別 150 張，並且需要 label，圖片大小約 640*480

```
8 train_dir, val_dir, test_dir = split_dataset(
9         images_in,
0         annotations_in,
1         val_split=0.2, test_split=0.2,
2         out_path=split_out_dir
3 )
```

4. 增加類別

```
label_map = ["M", "K", "S", "P", "A"]
```

5. 訓練模型

```
) print(f'train count: {len(train_data)}')
1 print(f'validation count: {len(validation_data)}')
2 print(f'test count: {len(test_data)}')
3 print('='*200)
```

```
train count: 540
validation count: 179
test count: 179
```

調整次數和批次

```
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
)
```

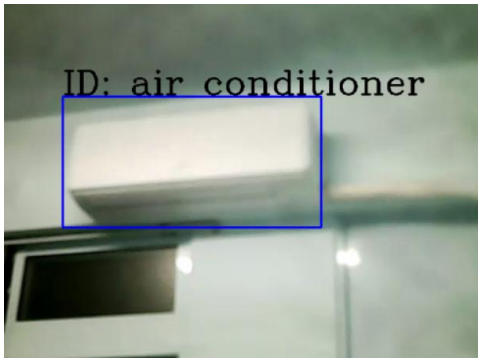
```
Epoch 2/200
1/7 [===>.....................] - ETA: 30s - det_loss: 1.6951
2/7 [=======>..................] - ETA: 25s - det_loss: 1.6962
3/7 [==========>.................] - ETA: 20s - det_loss: 1.6883
4/7 [==============>.............] - ETA: 15s - det_loss: 1.6727
5/7 [==================>.........] - ETA: 10s - det_loss: 1.6458
6/7 [======================>.....] - ETA: 5s - det_loss: 1.6221
7/7 [==============================] - ETA: 0s - det_loss: 1.6008
g_rate: 0.0400 - gradient_norm: 0.95042024-12-20 21:25:22.452894:
l not be optimized because the dataset does not implement the AsGr
7/7 [==============================] - 40s 6s/step - det_loss: 1.5
rning_rate: 0.0400 - gradient_norm: 0.9653 - val_det_loss: 1.2988
1.3621
Epoch 3/200
```

6. 驗證模型

```
w,h,c=input_details[0]['shape'][1:4]                    #  獲取輸入圖片大小
img_file='../datasets/efficientdet_dataset/oimage/20201110_222144_001.jpg'  #  一個圖片路徑
imgc  =  0
Pred  =  0
```

增加新的項目

```
if detection_scores[i]>0.5:
    outimg=cv2.rectangle(outimg, (i
    if detection_classes[i] ==1:
        ID = 'Mouse'
    elif detection_classes[i] ==2:
        ID = 'Keyboard'
    elif detection_classes[i] ==3:
        ID = 'Screen'
    elif detection_classes[i] ==4:
        ID = 'Phone'
    elif detection_classes[i] ==5:
        ID = 'air conditioner'
    print(detection_scores[i])
    outimg=cv2.putText(outimg,'ID:
```



7. 優化模型

```
!  edgetpu_compiler  '/content/drive/MyDrive/LAB4/efficientdet.tflite'

Edge TPU Compiler version 16.0.384591198
Started a compilation timeout timer of 180 seconds.

Model compiled successfully in 4475 ms.

Input model: /content/drive/MyDrive/LAB4/efficientdet.tflite
Input size: 4.24MiB
Output model: efficientdet_edgetpu.tflite
Output size: 5.61MiB
On-chip memory used for caching model parameters: 4.24MiB
On-chip memory remaining for caching model parameters: 3.27MiB
Off-chip memory used for streaming uncached model parameters: 0.00B
Number of Edge TPU subgraphs: 1
Total number of operations: 267
Operation log: efficientdet_edgetpu.log

Model successfully compiled but not all operations are supported by t
Number of operations that will run on Edge TPU: 264
Number of operations that will run on CPU: 3
See the operation log file for individual operation details.
Compilation child process completed within timeout period.
Compilation succeeded!
```

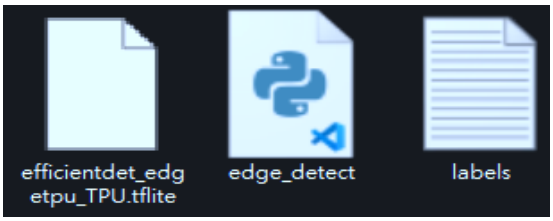
8. 優化完後下載模型

```
import  os
from  google.colab  import  files

files.download('efficientdet_edgetpu.tflite')
```

參、 Edge 端 操作流程-

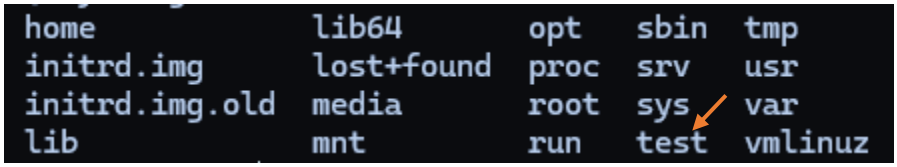
1. 準備檔案(優化後的 tflite 檔、要執行的 Py 檔、Labels 檔)



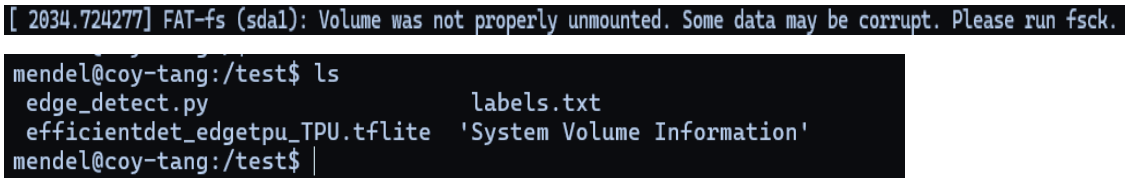
2. Labels 內容就是訓練時的類別

3. 進入到 dev board 的系統，進入 dev 中 尋找 sda1

4. 弄掛載 USB 的資料夾

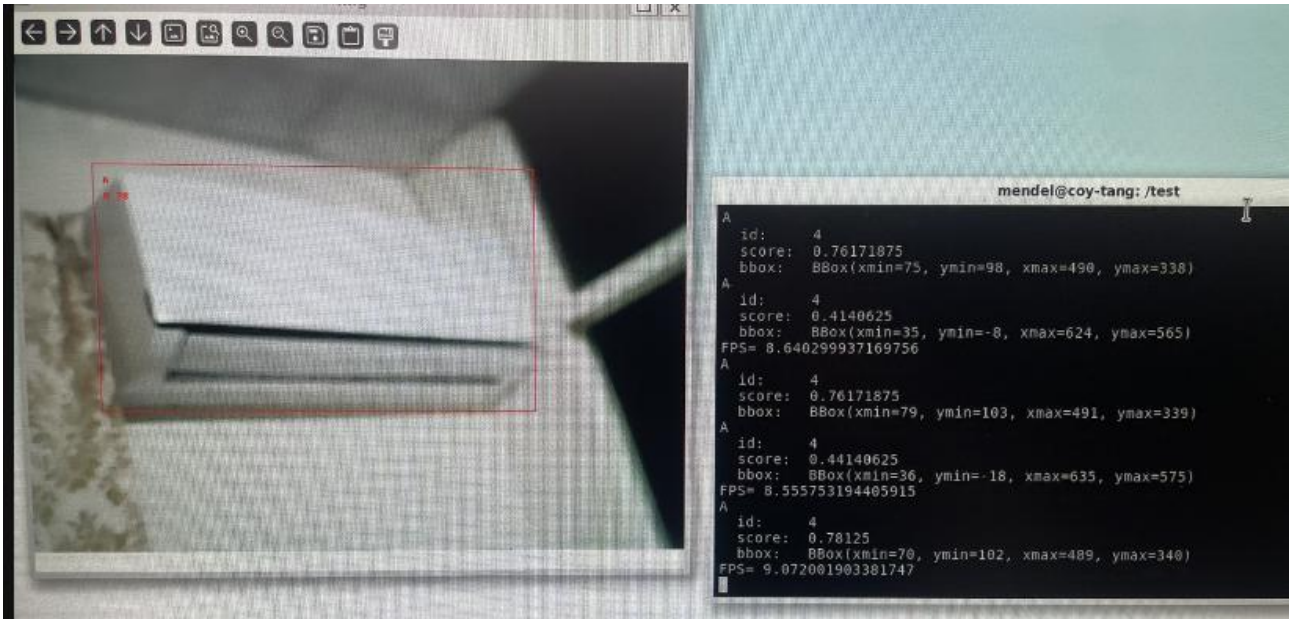


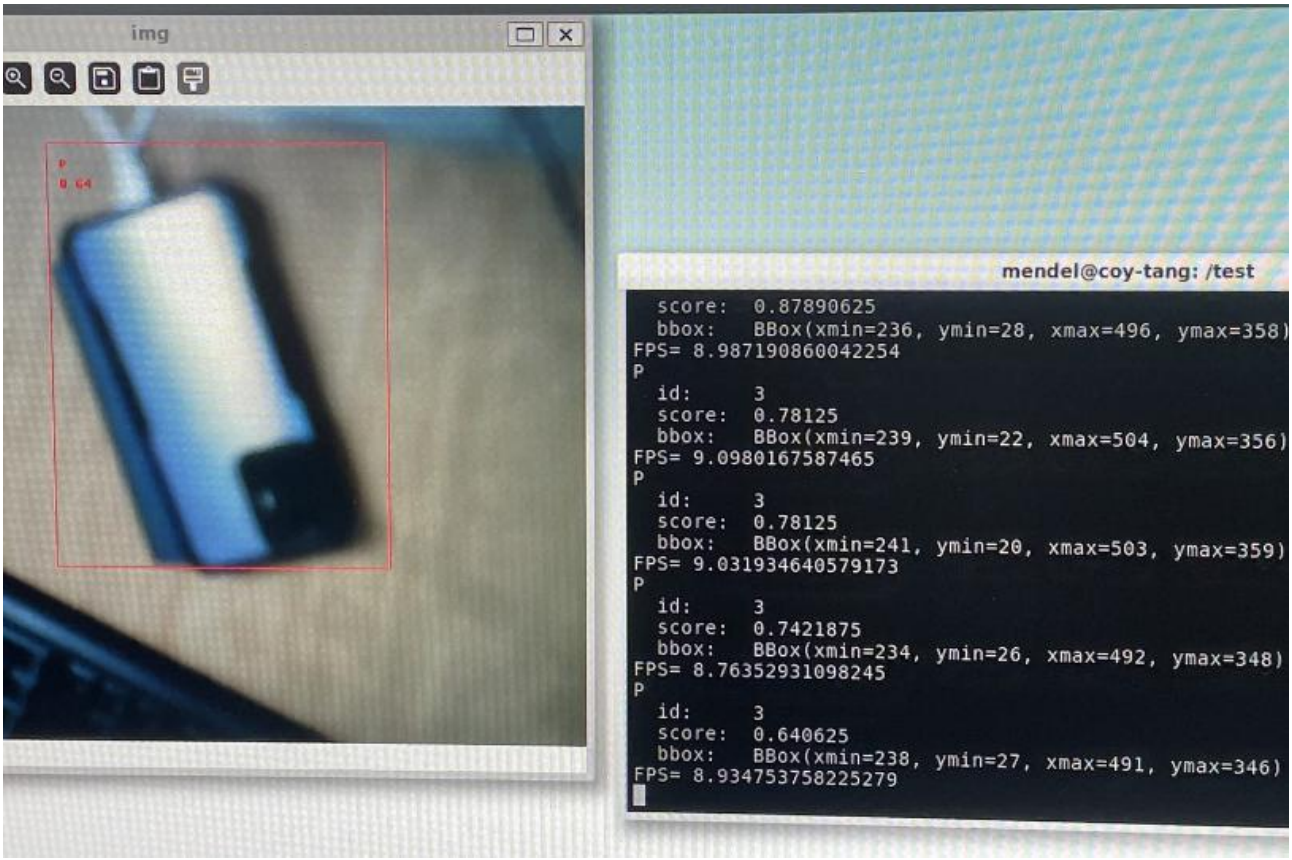
5. 輸入指令 將 USB 掛載到 test



6. 輸入執行指令 python3 edge_detect.py --model efficientdet_edgetpu_TPU.tflite --label labels.txt

肆、 Edge 端 驗證結果

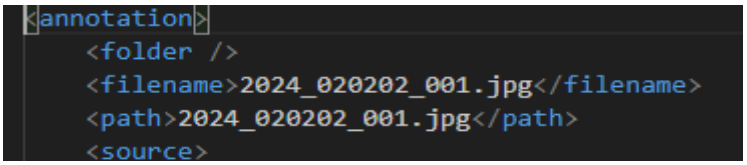




實作影片: https://youtu.be/bvBLoZIX0p4

伍、 遇到問題與解決

在新增的資料集中，即使已將 XML 和圖片的檔名修改完成，仍可能出現無法抓取資料的錯誤，這通常是因為 XML 檔內的標籤（如圖片檔名或路徑）沒有同步更新，因此在處理資料集時，不僅需要修改檔名，還必須確認 XML 檔內的標籤內容與檔案名稱完全一致，才能避免相關問題。



陸、 心得

在這次的 Lab4 實驗中，我學習了如何使用 EfficientDet 模型進行物件偵測，並完成了從 PC 端訓練到 Edge 端部署的整個流程。實驗中最大的收穫是了解了 BiFPN 和 Compound Scaling 等 EfficientDet 的設計特點，以及如何透過資料集的整理與模型優化來提高推論效能。

過程中，我遇到了資料集 XML 檔案標籤未更新的問題，導致模型無法正常運作。後來透過仔細檢查與修正，成功解決了這個問題，也讓我意識到資料準備的重要性。在 Edge 端操作時，掛載 USB 與執行模型的步驟讓我對邊緣運算的實務操作有了更多的理解。