視訊串流與追蹤HW2

1. Experiment Setup:

由於各門課在這個時期的作業量有些多，因此我選擇使用別人已經在網路上開源的專案進行使用。

(1) 我第一個使用的參考為imageai，github網址如下
https://github.com/OlafenwaMoses/ImageAI#customdetectiontraining

在這項參考中但由於這已經是一個相當完整的專案，各個功能也已經整理成函式庫，在學習上反而造成阻礙，因此我使用其他教程中提供的yolov3的weight檔案，使用其他開源程式轉為yolov3的model檔案，並使用該model對這次作業提供的validation image進行detection。

(2) 基於上述理由，我改採用第二個參考專案 -- Pytorch-Yolov3 master，github網址如下:
https://github.com/eriklindernoren/PyTorch-YOLOv3
此專案並沒有像imageai一樣將所有的功能都打包成為函式，但也相對降低了他的閱讀難度，並且仍然提供了使用自己的training data以及validation data進行model訓練的方法，並且在訓練過程中，每一個epoch會使用validation data以mAP的方式進行驗證，確實滿足了這次作業的需求，故我選擇使用這個專案使用。

2. Briefly explain your code:

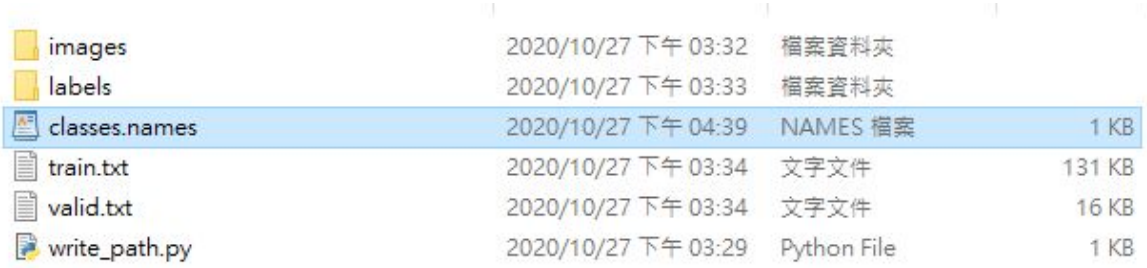
由於使用的是github的開源專案，在這個部分我將簡述我如何對這個專案進行操作。

(1) 專案建構:

根據Readme的指示，我們從專案的requirement.txt進行取得所需要的函示庫，如下圖:

其中，由於我的作業環境使用anaconda創建的conda環境，torch以及torchvision的部分我改用conda install進行處理。tensorflow與tensorboard的部分現行最新版本為2.3，但由於該專案當初編寫時是使用tensorflow 1代，故在這部分我有特別指定tensorflow版本為1.15。

(2) 訓練資料處理: 訓練用的資料夾如下



其中，images資料夾擺放training data以及validation data的圖片，labels則擺放training data以及validation data的標籤檔。classes.names檔案中則擺放欲分類的標籤名稱，對應標籤檔第一行的class name。train.txt擺放所有training data圖片的絕對路徑。valid.txt則擺放所有validation data圖片的絕對路徑。write_path.py則只是我用來快速列出所有圖片絕對路徑的程式檔。

(3) 訓練模型:

此專案提供train.py進行custom model training。training中提供的可調整參數如下圖:

```
$ train.py [-h] [--epochs EPOCHS] [--batch_size BATCH_SIZE]
               [--gradient_accumulations GRADIENT_ACCUMULATIONS]
               [--model_def MODEL_DEF] [--data_config DATA_CONFIG]
               [--pretrained_weights PRETRAINED_WEIGHTS] [--n_cpu N_CPU]
               [--img_size IMG_SIZE]
               [--checkpoint_interval CHECKPOINT_INTERVAL]
               [--evaluation_interval EVALUATION_INTERVAL]
               [--compute_map COMPUTE_MAP]
               [--multiscale_training MULTISCALE_TRAINING]
```

其中，除了batch size的部分由於default size會造成記憶體不足，故改使用batch size為4之外，其他都使用default的參數進行訓練。

每一個epoch訓練完成後會儲存在一個.pth檔案中(可視作weight檔)，最後我在epoch為21時停止訓練，因為mAP已超越要求的0.4。

(4) 測試模型

最後使用test.py配合相對應的參數，專案會自動將對應的test image輸出成畫好detection框框的格式。在這個部分我將偵測框的資訊以作業要求的方式進行輸出至detection-result的資料夾中。結果如下:

19_000000.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
0 1.000000 93 433 319 523
0 0.999997 1318 567 1570 674
```

3. Your validation result on your model:

由於我並沒有成功實作出se layer的版本(但仍然會在discussion中進行討論), 故此部分僅提供origin版本的實驗結果。

在epoch為10時，loss value大約為1.3~2.5之間，而mAP值則為0.53。

在epoch為21時，loss value大約為0.2~0.9之間，而mAP值則為0.71。

4. discussion

在我所參考的專案中，他使用.cfg檔(yolov3的config檔案)針對yolov3的網路進行定義，外觀如下:

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=16
subdivisions=1
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky
```

(僅節錄部分)

接著在parser中將config中的所有內容根據標籤以及標籤下的參數拆分並引入，根據這些參數定義相對應層的內容(eg. convolutional layer)。逐層加入整個model以建構完整的yolov3 model。

而SE layer的部分，根據我在網路上所查到的各項教學以及文章說明，得到結論是SE layer加在backbone之後的效果會達到最好，因為在這個位置最能凸顯出不同channel的重要程度，也因此而提高模型的學習表現。

實作上，根據網路上的教學，我需要在config檔案中加入resolution layer，並在parser中加入我所需要判別的項目後，根據這些項目重新定義SE layer。但由於專案龐大，加入SE Layer的部分頻頻出現錯誤，因此在SE layer的建構上失敗。

在這次作業中我遇到最大的困難，是對於Object detection model的不熟悉以及時間上的不足。由於不熟悉，在一開始進行資料收集的階段，我甚至連如何使用別人的專案都遇到了一些困難。而後在大致了解了上面所提到了建構整個訓練模型的流程後，也因為對使用config file建立model的方法不夠了解而找不到查詢的方向。

而後，在找到合適的教學後雖然對整體架構有了一點了解，並想要依照這樣的架構重新自己定義整個model，但一是找不到錯誤在哪，二是仍有其他科目的作業待完成，因此最後只能使用參考專案進行custom training並產出結果。