

Video streaming and tracking - HW2

311551096 錢承

1. Experiment Setup

在訓練前，將 label 的格式轉換成 COCO 格式，也就是將 class、x_center、y_center、width、height 轉換成 class、left、top、right、bottom，如下圖。

```
def toCOCOformat(set):
    coco = Coco()
    coco.add_category(CocoCategory(id = 3, name = 'car'))

    training_dataset = glob.glob('gta/label/' + set + '_labels/*')

    for file in training_dataset:
        label = open(file, 'r')
        train = label.readlines()

        coco_image = CocoImage(file_name=file.split('/')[1][:3] + '.jpg', height=1080, width=1920)
        for i in range(len(train)):
            train[i] = train[i].strip()
            bbox = train[i].split(' ')

            x_min = float(bbox[1])*1920 - float(bbox[3])*1920 / 2
            y_min = float(bbox[2])*1080 - float(bbox[4])*1080 / 2
            width = float(bbox[3])*1920
            height = float(bbox[4])*1080
            coco_image.add_annotation(
                CocoAnnotation(
                    bbox=[x_min, y_min, width, height],
                    category_id = 3,
                    category_name='car'
                )
            )
        coco.add_image(coco_image)

    save_json(data=coco.json, save_path='instances_' + set + '.json')
```

訓練時候，我包含 SE 於與不包含 SE 的模型，分別訓練了 600 個 epochs。

在計算 mAP 時候，會將 label 由 class、x_center、y_center、width、height 格式，轉換成 class、left、top、right、bottom 格式，如下圖。

```
def evalformat(set):
    training_dataset = glob.glob('gta/label/'+ set + '_labels/*')

    for file in training_dataset:
        label = open(file, 'r')
        train = label.readlines()

        gt = open('val/groundtruths/'+file.split('/')[0], 'w')
        for i in range(len(train)):

            train[i] = train[i].strip()
            bbox = train[i].split(' ')

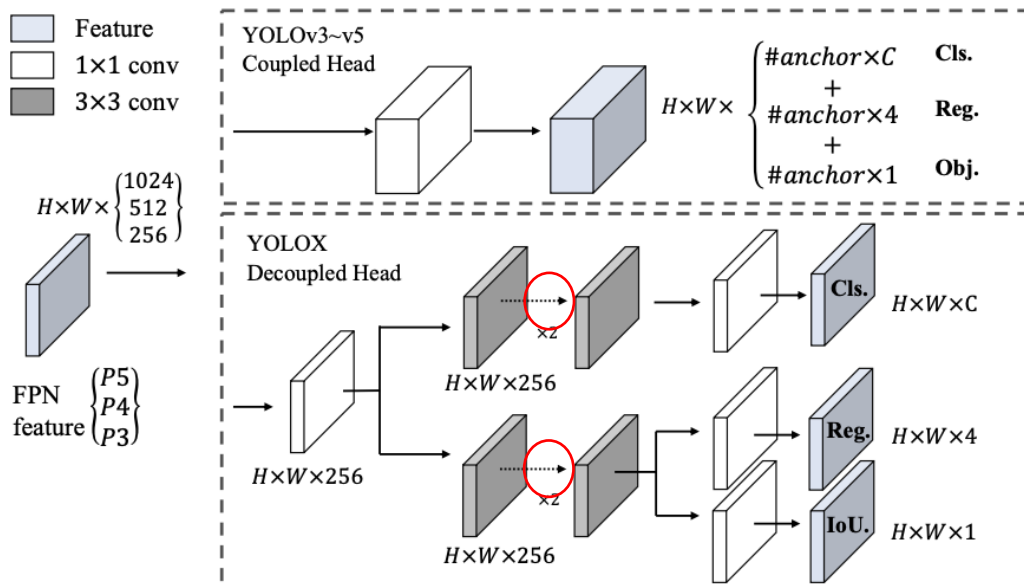
            x_min = float(bbox[1])*1920 - float(bbox[3])*1920 / 2
            y_min = float(bbox[2])*1080 - float(bbox[4])*1080 / 2
            width = float(bbox[3])*1920
            height = float(bbox[4])*1080

            gt.write(f'0 {x_min} {y_min} {x_min+width} {y_min+height}\n')
```

2. Brief explain your code

不包含 SE 的模型，於原本的 YOLOX 無異。

而包含 SE 的模型，我在 YOLOX 中 Decoupled Head 加入兩個 SE layer，於下圖紅圈處。



實作細節如下圖：

```
class SELayer(nn.Module):
    def __init__(self, channel, reduction = 16):
        super(SELayer, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
            nn.Linear(channel, channel//reduction),
            nn.ReLU(inplace = True),
            nn.Linear(channel//reduction, channel),
            nn.Sigmoid()
        )
```

```
self.cls_convs.append(
    nn.Sequential(
        *[
            Conv(
                in_channels=int(256 * width),
                out_channels=int(256 * width),
                ksize=3,
                stride=1,
                act=act,
            ),
            SELayer(channel = int(256 * width)),
            Conv(
                in_channels=int(256 * width),
                out_channels=int(256 * width),
                ksize=3,
                stride=1,
                act=act,
            ),
        ]
    )
)
```

```
self.reg_convs.append(
    nn.Sequential(
        *[
            Conv(
                in_channels=int(256 * width),
                out_channels=int(256 * width),
                ksize=3,
                stride=1,
                act=act,
            ),
            SELayer(channel = int(256 * width)),
            Conv(
                in_channels=int(256 * width),
                out_channels=int(256 * width),
                ksize=3,
                stride=1,
                act=act,
            ),
        ]
    )
)
```

利用 weight 產生 result 流程如下，

(1) 系統資訊：

System: Ubuntu 20.04.5 LTS
Kernel: 5.15.0-52-generic
Cuda: 11.6
GPU: NVIDIA GeForce RTX 3060 Ti

(2) 環境建置：

```
$ sudo apt install cmake  
$ conda env create -n lab2 python==3.9.13  
$ conda activate lab2  
$ pip3 install torch torchvision torchaudio --extra-index-url  
https://download.pytorch.org/whl/cu116  
*其中重要套件版本為 torch==1.13.0+cu116、torchaudio==0.13.0+cu116、  
torchvision==0.14.0+cu116
```

(3) Testing data 防止位置及命名：

將 testing data 放置於 Code 資料夾下，如下右圖。再來將 testing data 的資料解命名為 gta，且圖片部分的資料夾命名為 test，label 部分的資料夾命名為 test_labels。其中 label 格式依序為 class、x_center、y_center、width、height 如下左圖。

```
0 0.59453125 0.8930555555555556 0.08802083333333334 0.21388888888888888  
0 0.7171875 0.5055555555555555 0.1125 0.08148148148148149  
0 0.5263020833333333 0.4310185185185185 0.0515625 0.07685185185185185  
0 0.54375 0.3023148148148148 0.03333333333333333 0.06018518518518518  
0 0.55234375 0.27037037037037037 0.028645833333333332 0.03518518518518518  
0 0.09869791666666666 0.5041666666666667 0.12447916666666667 0.06130208333333333  
0 0.6130208333333333 0.300462962962963 0.03645833333333336 0.04  
0 0.6296875 0.16805555555555557 0.06875 0.09166666666666666  
class x_center y_center width height
```

```
-- gta  
|-- test  
|-- test_labels  
|-- Object-Detection-Metrics  
|-- Original  
|-- best_ckpt_original_e600.pth  
|-- Original_TransformLabelFormat_and_Inference.py  
|-- YOLOX  
-- SE  
|-- best_ckpt_SE_e600.pth  
|-- SE_TransformLabelFormat_and_Inference.py  
-- YOLOX
```

(4) 執行沒有加 SE (Original)，產生預測結果:

```
$ cd Code/Original/YOLOX
```

```
$ pip3 install -v -e .
```

```
$ cd Code/Original
```

```
$ python Original_TransformLabelFormat_and_Inference.py
```

*此結果將轉換 label(groundtruths)的格式，格式依序為 class、left、top、right、bottom。並且利用訓練完的 weight 產生預測結果，然後將結果輸出到 gta/Original/detections 中。

(5) 執行有加 SE，產生預測結果:

```
$ cd Code/SE/YOLOX
```

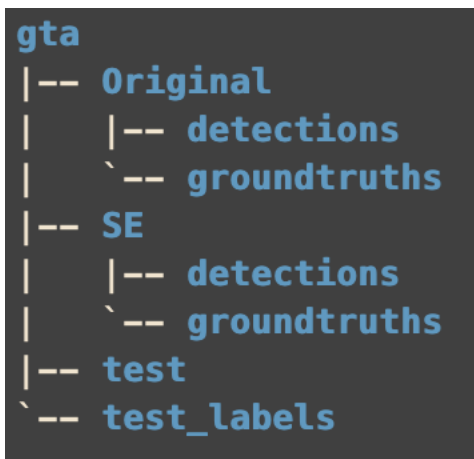
```
$ pip3 install -v -e .
```

```
$ cd Code/SE
```

```
$ python SE_TransformLabelFormat_and_Inference.py
```

*此結果將轉換 label(groundtruths)的格式，格式依序為 class、left、top、right、bottom。並且利用訓練完的 weight 產生預測結果，然後將結果輸出到 gta/SE/detections 中。

*執行完 (4) (5) 點後，資料夾結構如下圖，預測結果放置於 gta/Original/detections 和 gta/SE/detections。



```
gta
|-- Original
|   |-- detections
|   `-- groundtruths
|-- SE
|   |-- detections
|   `-- groundtruths
|-- test
`-- test_labels
```

(6) 計算 mAP(沒有加 SE):

```
$ cd Code/Object-Detection-Metrics
```

```
$ python pascalvoc.py -gt ../gta/Original/groundtruths -gtformat xyrb -  
det ../gta/Original/detections -detformat xyrb -t 0.85
```

(7) 計算 mAP(有 SE):

```
$ cd Code/Object-Detection-Metrics  
$ python pascalvoc.py -gt ../gta/SE/groundtruths -gtformat xyrb -  
det ../gta/SE/detections -detformat xyrb -t 0.85
```

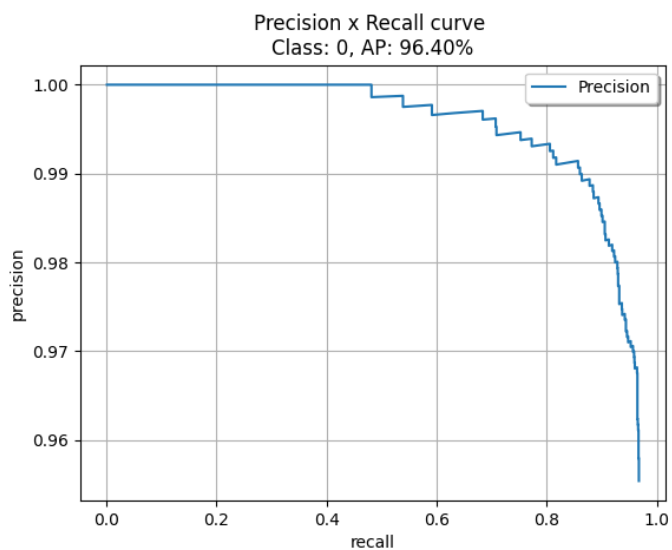
Reference: <https://github.com/Megvii-BaseDetection/YOLOX>

3. Screenshot your validation results on your two models (with/without SE module)

without SE :

mAP: 96.40%

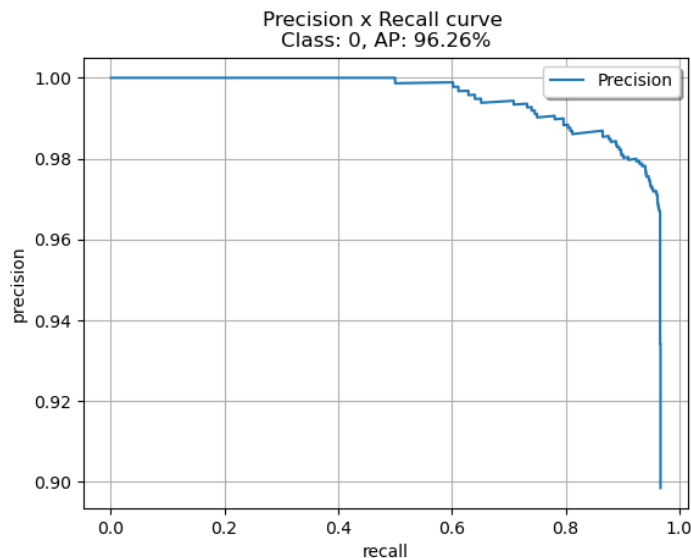
```
Folder /home/jamesqian/Documents/Video-streaming-and-tracking/Lab2/Object-Detection-Metrics/results already exists and may contain important results.  
Enter 'Y' to continue. WARNING: THIS WILL REMOVE ALL THE CONTENTS OF THE FOLDER!  
Or enter 'N' to abort and choose another folder to save the results.  
y  
AP: 96.40% (0)  
mAP: 96.40%
```



with SE :

mAP: 96.26%

```
Folder /home/jamesqian/Documents/Video-streaming-and-tracking/Lab2/Object-Detection-Metrics/results already exists and may contain important results.  
Enter 'Y' to continue. WARNING: THIS WILL REMOVE ALL THE CONTENTS OF THE FOLDER!  
Or enter 'N' to abort and choose another folder to save the results.  
y  
AP: 96.26% (8)  
mAP: 96.26%
```



4. Discussion

在實作過程中，主要遇到的困難是 label 的格式問題，我採用 YOLOX 官方的 code，而官方的 dataloader 主要是讀取 COCO 格式，我因此我撰寫程式轉換作業提供的格式到 COCO 格式。另外一個格式問題，主要是要去 Object-Detection-Metrics (<https://github.com/rafaelpadilla/Object-Detection-Metrics>) 計算 mAP 的時候，groundtruths 的格式是需要轉換成 class、left、top、right、bottom。

增加 SE layer 後，每一個 bbox 的準確率有提升，可是在 conf. 85 卻有出現錯誤的 bbox，下左圖為無 SE layer，下右圖為有 SE layer。

