Video Steaming and tracking HW2

機器人碩一 312605015 詹恆瑜

一、 資料前處理與程式撰寫:

(1)

整體資料夾結構及下述會提到的程式碼位置如下:

```
[Original or SE]
| transform_coco.py
|--- YOLOX
|    |--- tools
|         |-- demo.py
|         `-- my_txt_to_1980txt.py
|
|---datasets
|    |--- vstcar
|         |--- annotations(放train、val的json.)
|         |--- train2017  (放train照片)
|         `--- val2017    (放val照片)
```

根據原先 YOLOX 的程式碼在 YOLOX/dataset 中創立名為 vstcar 的資料夾，並在裡面創立檔名相同標註檔、訓練集、驗證集的資料夾，分別為 annotations、train2017、val2017 三個資料夾，再將我們的 train_labels、val_labels 的訓練標註 txt.檔轉為 coco 的格式，也就是將助教原先的[class x_center y_center width height]形式改為依照 coco 形式，train_labels、 val_labels 會分別生成各一個.json 檔放在 annotations 資料夾中，我自己撰寫 一個 transform_coco.py 檔來完成，另外再將訓練的 train 照片放在 train2017 資料夾中且將驗證的 val 照片放在 val2017 中。

```
$ cd Code/[Original or SE]
$ python transform_coco.py
```

```python
def convert_labels_to_coco_format(input_labels_dir, output_coco_dir, mode):
    coco_data = {
        "images": [],
        "annotations": [],
        "categories": [{"id": 0, "name": "car"}],
    }

    image_folder = os.path.join(output_coco_dir, mode)
    os.makedirs(image_folder, exist_ok=True)

    label_files = os.listdir(input_labels_dir)
    for idx, label_file in enumerate(label_files):
        with open(os.path.join(input_labels_dir, label_file), 'r') as f:
            lines = f.readlines()

        image_id = idx
        image_info = {
            "id": image_id,
            "file_name": label_file.replace(".txt", ".jpg"),
            "width": 1920,
            "height": 1080,
        }
        coco_data["images"].append(image_info)

        for line in lines:
            class_id, x_center, y_center, width, height = map(float, line.strip().split())
            x_min = (x_center - width/2) * 1920
            y_min = (y_center - height/2) * 1080
            x_max = (x_center + width/2) * 1920
            y_max = (y_center + height/2) * 1080
            annotation_info = {
                "id": len(coco_data["annotations"]),
                "image_id": image_id,
                "category_id": int(class_id),
                "bbox": [x_min, y_min, x_max-x_min, y_max-y_min],
                "area": (x_max - x_min) * (y_max - y_min),
                "iscrowd": 0,
                "segmentation": [],
            }
            coco_data["annotations"].append(annotation_info)

    with open(os.path.join(output_coco_dir, f"{mode}_coco_format.json"), "w") as json_file:
        json.dump(coco_data, json_file)
```

(2)

接著將寫一個程式碼檔名為 my_txt_to_1980txt.py，路徑在 YOLOX/tools 中，將原先的 val_labels 的 txt. 檔改為[class left top right bottom]的格式，並放在助教給的 Object-Detection-Metrics/groudtruths 中，之後把自己預測出的 txt. 檔放在 Object-Detection-Metrics/detections 中，就可以利用助教給的程式碼測試自己 val 的預測精度，並做紀錄。

$ cd Code/[Original or SE]/YOLOX/tools

$ python my_txt_to_1980txt.py

$ cd Code/SE/Object-Detection-Metrics

$ python pascalvoc.py -t 0.85 -gtformat xyrb -detformat xyrb -np

```python
import os

def convert_val_labels(input_file, output_folder):
    with open(input_file, 'r') as f:
        lines = f.readlines()

    output_file = os.path.join(output_folder, os.path.basename(input_file))

    for line in lines:
        class_idx, x_center, y_center, width, height = map(float, line.strip().split())

        left = int(x_center * 1920 - (width * 1920) / 2)
        top = int(y_center * 1080 - (height * 1080) / 2)
        right = int(left + width * 1920)
        bottom = int(top + height * 1080)


        output_line = "{} {} {} {} {}\n".format(int(class_idx), left, top, right, bottom)

        with open(output_file, 'a') as f_out:
            f_out.write(output_line)


input_folder = "C:/software/python/HW2_ObjectDetection_2023/val_labels"
output_folder = "C:/software/python/HW2_ObjectDetection_2023/Object-Detection-Metrics/groundtruths"


os.makedirs(output_folder, exist_ok=True)


input_files = [os.path.join(input_folder, f) for f in os.listdir(input_folder) if f.endswith('.txt')]


for input_file in input_files:

    output_file = os.path.join(output_folder, os.path.basename(input_file))
    open(output_file, 'w').close()

    convert_val_labels(input_file, output_folder)
```

(3)

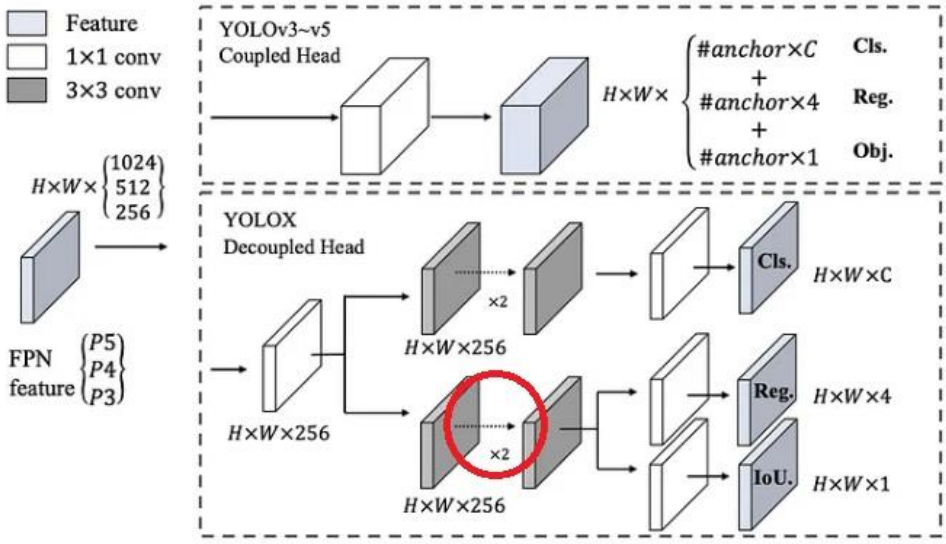
在路徑 YOLOX 中輸入指令”python tools/python.py -f exp/example/custom/yolox_s.py -d 1 -b 8 --fp16 -o -c weights/yolox_s.pth”訓練完之後，且確定使用 tools/demo.py 可以偵測出所要的車子後，我將原先偵測出框框的 demo.py 中加改入一段程式碼把框框的位置依照[class confidence left top right bottom]紀錄，之後當我執行 python tools/demo.py image -f exp/example/custom/yolox_s.py -c YOLOX_outputs/yolox_s/best_ckpt.pth –path C:/software/python/HW2_312605015/Code/Original/val –conf 0.35 –nms 0.45 –save_result –device gpu 後會把照片和偵測完的各照片框框資訊 txt.檔的資料夾放在一起，如此就可以得到結果，先利用測試 val 和 groudtrush 精度後再輸出 test 放在對應的資料夾中。

```python
def save_detection_result(self, output, img_info, save_folder):
    if output is None:
        return
    output = output.cpu()

    bboxes = output[:, 0:4]


    bboxes /= img_info["ratio"]

    cls = output[:, 6]
    scores = output[:, 4] * output[:, 5]

    file_name = img_info["file_name"].split('.')[0] + ".txt"
    save_path = os.path.join(save_folder, file_name)
    with open(save_path, 'w') as f:
        for i in range(len(bboxes)):
            line = "{} {} {} {} {} {}\n".format(
                int(cls[i]),
                scores[i],
                int(bboxes[i][0]),
                int(bboxes[i][1]),
                int(bboxes[i][2]),
                int(bboxes[i][3])
            )
            f.write(line)
```

```python
def image_demo(predictor, vis_folder, path, current_time, save_result):
    if os.path.isdir(path):
        files = get_image_list(path)
    else:
        files = [path]
    files.sort()
    for image_name in files:
        outputs, img_info = predictor.inference(image_name)
        result_image = predictor.visual(outputs[0], img_info, predictor.confthre)
        if save_result:

            save_folder = os.path.join(
                vis_folder, time.strftime("%Y_%m_%d_%H_%M_%S", current_time)
            )
            os.makedirs(save_folder, exist_ok=True)


            results_folder = os.path.join(save_folder, "results")
            os.makedirs(results_folder, exist_ok=True)

            save_file_name = os.path.join(save_folder, os.path.basename(image_name))
            logger.info("Saving detection result in {}".format(save_file_name))
            cv2.imwrite(save_file_name, result_image)


            predictor.save_detection_result(outputs[0], img_info, results_folder)

            ch = cv2.waitKey(0)
            if ch == 27 or ch == ord("q") or ch == ord("Q"):
                break
```

(4)

在實驗 SE Layer 中我在 YOLOX 的 Regression conv.加入 SE Layer+CBAM 來做改善，細節位置在如圖:




```python
class SELayer(nn.Module):
    def __int__(self, channel,reduction=16):
        super(SELayer,self)._int_()
        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()
        )

    def forward(self, x):
        b, c, _, _ = x.size()
        y = self.avg_pool(x).view(b, c)
        y= self.fc(y).view(b, c, 1 ,1)
        return x * y
```

```python
class CBAM(nn.Module):
    def __init__(self, channel, reduction=16):
        super(CBAM, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.max_pool = nn.AdaptiveMaxPool2d(1)

        self.fc1 = nn.Sequential(
            nn.Conv2d(channel, channel // reduction, kernel_size=1, padding=0),
            nn.ReLU(inplace=True),
            nn.Conv2d(channel // reduction, channel, kernel_size=1, padding=0)
        )

        self.fc2 = nn.Sequential(
            nn.Conv2d(channel, channel // reduction, kernel_size=1, padding=0),
            nn.ReLU(inplace=True),
            nn.Conv2d(channel // reduction, channel, kernel_size=1, padding=0)
        )

        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        b, c, _, _ = x.size()


        y_avg = self.avg_pool(x)
        y_max = self.max_pool(x)
        y_avg = self.fc1(y_avg)
        y_max = self.fc2(y_max)
        y = self.sigmoid(y_avg + y_max)


        z_avg = torch.mean(y, dim=1, keepdim=True)
        z_max, _ = torch.max(y, dim=1, keepdim=True)
        z = self.sigmoid(z_avg + z_max)

        return x * y * z
```

```python
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)),
            CBAM(channel=int(256 * width)),
            Conv(
                in_channels=int(256 * width),
                out_channels=int(256 * width),
                ksize=3,
                stride=1,
                act=act,
            ),
        ]
    )
)
```

二、 執行階段

(1)執行 Original 訓練及 demo 其結果。

```
$ cd Code/Original/YOLOX
$ pip3 install -v -e
$ python python tools/python.py -f exp/example/custom/yolox_s.py -d 1 -
b 8 --fp16 -o -c weights/yolox_s.pth
$ python tools/demo.py image -f exp/example/custom/yolox_s.py -c
YOLOX_outputs/yolox_s/best_ckpt.pth –path
C:/software/python/HW2_312605015/Code/Original/val –conf 0.35 –nms
0.45 –save_result –device gpu
```

* C:/software/python/HW2_312605015/Code/Original/val 改為自己的 val 或 test 的位置。
* 因為有改過原先的 demo.py 檔(上面有提到)，故最後預測的結果 txt.檔會在 Code/Original/YOLOX/YOLOX/YOLOX_outputs/yolox_s/vis_res/該時間資料夾中的 results 資料夾中。

- 若要預測結果:

先將 results 結果的 txt 檔放在 Code/SE/Object-Detection-Metrics/detections 中。

```
$ cd Code/SE/Object-Detection-Metrics
$ python pascalvoc.py -t 0.85 -gtformat xyrb -detformat xyrb -np
```

(2)執行 SE Layer 訓練及 demo 其結果。

```
$ cd Code/SE/YOLOX
$ pip3 install -v -e
$ python python tools/python.py -f exp/example/custom/yolox_s.py -d 1 -
b 8 --fp16 -o -c weights/yolox_s.pth
$ python tools/demo.py image -f exp/example/custom/yolox_s.py -c
YOLOX_outputs/yolox_s/best_ckpt.pth –path
C:/software/python/HW2_312605015/Code/SE/val –conf 0.35 –nms 0.45
–save_result –device gpu
```

* C:/software/python/HW2_312605015/Code/SE/val 改為自己的 val 或 test 的位置

*因為有改過原先的 demo.py 檔(上面有提到)，故最後預測的結果 txt.檔會在 Code/SE/YOLOX/YOLOX/YOLOX_outputs/yolox_s/vis_res/該時間資料夾中的 results 資料夾中。

➢ 若要預測結果:

先將 results 結果的 txt 檔放在 Code/SE/Object-Detection-Metrics/detections 中。

```
$ cd Code/SE/Object-Detection-Metrics
$ python pascalvoc.py -t 0.85 -gtformat xyrb -detformat xyrb -np
```

三、 作業實驗結果報告:

- Original part:

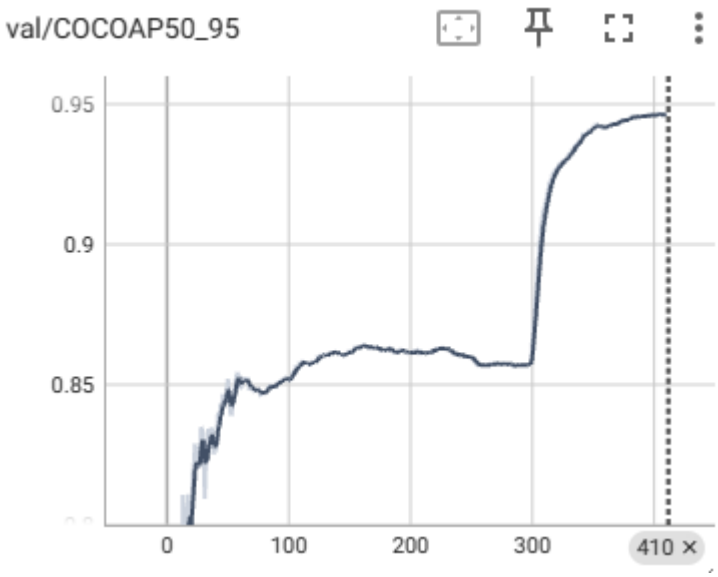
我首先使用預設 epoch300 去跑訓練並利用我訓練完最好的權重檔去做做預測並測試我的 val 精度，在訓練完後得到我的精度為 MAP=95.58%。

接著我為了要提高精度，我先去觀察我的訓練過程，我發現在 200 多開始就會起伏非常少，直到最後 15 個 epoch 停止 augmentation 後才開始有顯著的提升效果，所以我把我 epoch 提高到 410 後也將 augmentation 改為倒數 45 個 epoch 停止並再做訓練，訓練完不斷的利用 tensorboard 的圖知道 MAP 還有可以再上升的趨勢，所以我把 augmentation 再調到倒數 100、125、110 停止，去做嘗試發現 100~125 就不太會有明顯變化，甚至有時候 100 的值會高於 125。

之後我再提整 Batch size 的大小，我發現我把原先的 32 改為 16 後時間減半了，訓練效果也大於等於 Batch 32 的效果，所以我後來我又多加嘗試了 batch 8，綜合以上的嘗試，我最好選取訓練最好的當作我最後的權重。

	Epochs	MAP	No Aug	Time	Batch Size
Original	300	95.85	15	12hr	32
Original	410	96.59	100	14hr	32
Original	410	96.62	125	14hr	32
Original	410	96.49	125	5.7hr	16
Original	410	96.61	120	5.7hr	16
Original	410	97.06	110	5.7-6.hr	8

```
Folder C:\software\python\HW2_312605015\Code\Original\Object-Detection-Metrics\results already exists and may contain im
portant 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: 97.06% (0)
mAP: 97.06%
```



以上是我的嘗試結果，紅色為最後選取的權重檔和利用 tensorboard 出來的訓練過程圖，可以知道到後面收斂時起伏變得很小。 。

- SE Layer part:

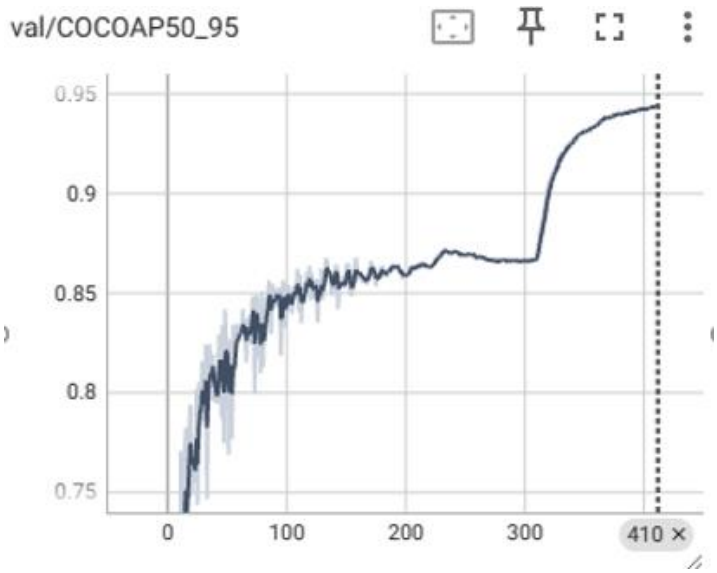
我首先分別在 YOLOX 的 ecoupled head 結構中的 classification 和 regression 的 covolution 中加入 SE Layer，其他參數沒改，同樣為 epoch300 去訓練，最後把訓練完的模型一樣做 val 的預測和測試，最後的 MAP 為 95.2%，這讓我發現兩邊都加入這樣使效果變差了。

於是我因為覺得辨別類別只有一個，所以不在 classification conv 中加入 SE Layer，只在 Regression conv 中加入 SE Layer 並同時加入 CBAM 來試看看效果，同樣的驗證步驟去做 val 精度測試，沒想到竟然跟上一個實驗 classification 和 regression 的 covolution 中加入 SE Layer 的精度一樣為 95.2%。

於是我測試把跟 Original 一樣，把 Epoch 300 改為 410 且將倒數 15 個 epoch 停止資料 augmentation 改為倒數 45、100、125、110 個 epoch 就停止來做測試，結果同樣發現到 100 以上後效果不再穩定變好，batch size 方面我也試過 32、16、8 效果都差不多，甚至有時候 Batch size 調小會好一些，我就在這幾次實驗測試中找一次最好 MAP 的權重當作我最後的權重。

	Epochs	MAP	No Aug	Time	Batch Size
SE(Reg.)+SE(Clas.)	300	95.4	15	12hr	32
SE(Reg.)+CBAM	410	97.1	100	11.5hr	32
SE(Reg.)+CBAM	410	96.49	125	11.5hr	32
SE(Reg.)+CBAM	410	96.91	125	5.7hr	16
SE(Reg.)+CBAM	410	96.8	110	5.7-6.hr	8

```
Folder C:\software\python\HW2_312605015\Code\SE\Object-Detection-Metrics\results already exists and
t 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: 97.10% (0)
mAP: 97.10%
```



以上是我的嘗試結果，紅色為最後選取的權重檔，下圖為使用 tensorboard 畫出來的訓練過程圖，可以知道到後面收斂時起伏變得很小。

由 Original 和 SE 的 Epoch410、NO AUG100、Batch size32 和 Epoch410、NO AUG125、Batch size16 看，加入 SE 的效果有變好一些。

但由其他訓練結果來看，我發現有時候反爾會變差一些些，我想可能和隨機種子帶來的實驗誤差有些關係，因為我從 tensorboard 畫出來的途中有看到到後面逐漸收斂時提昇都是好一點點而已，這時候做調整對那效果都是蠻微小的，另外一點，可以確定的是當 Batch size 條小之後速度快了很多，結果大部分都可能會在好一些。

四、 環境及系統資訊

(1) 系統資訊

裝置名稱:DESKTOP-NDAHSQ6
處理器:AMD Ryzen 5 7500F 6-Core Processor 3.70 GHz
已安裝記憶體(RAM) 32.0 GB
系統類型 64 位元作業系統，x64 型處理器
顯卡:NVIDIA GeForce RTX 3080
版本:Windows 11 專業版
版本:22H2
OS 組建 22621.2428

(2) 環境建置

```
pip install cuda11.8
$ cd ./HW_312605015/Code
$ conda env create -f environment.yml
$ conda activate yolox
$ conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c
pytorch -c nvidia
```

(Original 和 SE 的環境是同一個，我把它分別放
進./Code/Original、./Code/SE 的資料夾中，檔名為 environment.yml)

五、 心得

做這一次作業讓我第一次接觸 YOLOX 也第一次認識什麼是 coco 格式，透過不斷的研究學習到如何將所有的 txt.檔轉變為可以讀取的 json 檔和得到框框的資訊等等，在這之中我就花了不少時間，也學到了很多，直到最後可以執行之後才完成一部分而已，我在 original 用預設設定跑 300 個 epoch 後就在思考如何訓練得更好，於是做了以上一連串的測試，我跑一次 300 個 epoch 就要花 9 個小時，只能利用訓練的時間來做資料查詢並思考如何修正的更好，也很高興在最後不斷的測試之後有得到更好效果，也蠻有成就感的。

在一次次測試中，雖然試了不少不一樣的嘗試，但也不是每一次效果都有變好，要在這其中都要不斷思考哪裡出了問題並再去修改，這也讓我更加熟悉 yolox 的內部設計和結構，做完這次作業我覺得非常充實。