Video Steaming and tracking HW2

機器人碩一 312605015 詹恆瑜

1. **資料前處理:**

(1)

整體資料夾結構及下述會提到的程式碼位置如下:

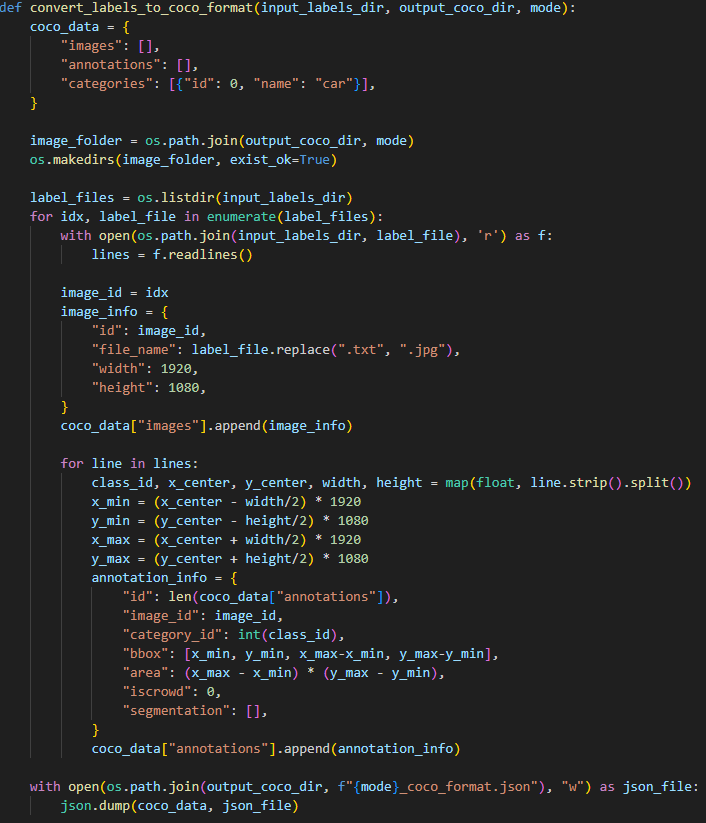
**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

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，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，我自己撰寫一個transform\_coco.py檔來完成。

$ cd Code/[Original or SE]

$ python transform\_coco.py



(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

image

(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。

IMG

(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: **一張含有 文字, 圖表, 螢幕擷取畫面, 方案 的圖片

自動產生的描述**

IMG

1. **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

1. **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 |

以上是我的嘗試結果，紅色為最後選取的權重檔。

* 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 |

以上是我的嘗試結果，紅色為最後選取的權重檔。

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，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。

1. **環境及系統資訊**

(1) 系統資訊

(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

1. **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。