

視訊串流與追蹤 LAB2

311605011 黃品振

1. Experiment setup:

- Data pre-process:

這份作業所提供的 dataset 是 yolo 格式的訓練資料，然而 YOLOX 並不支援 YOLO 格式的資料所以必須先透過轉換成 YOLOX 有支援的 COCO 格式或者是 VOC 格式，我是使用 COCO 格式來做訓練，因為這樣需要改的東西比較少。要從 YOLO 格式轉換成 COCO 格式需要將 txt 檔的資訊改寫入一個 JSON 檔，YOLO 格式的 bbox 是[center_x, center_y, width, height]，也要改寫成 COCO 格式的[left_top_x, left_top_y, width, height]。然後將處理好的 train data 以及 validate data 放到 YOLOX/datasets 裡面訓練。

2. Briefly explain my code:

I. Explain code and command line(without SE modules):

大部分的程式碼都來自 <https://github.com/Megvii-BaseDetection/YOLOX> 我只有改了以下幾點東西

- YOLOX/tools/demo.py:

我在裡面將 bounding box 以及信心值還有類別的資訊拿出來然後生成 txt 檔，方便拿去做後續 evaluate。

- YOLOX/yolox/data/datasets/coco_classes.py:

我將類別改掉因為我們這次只需要辨識一個類別，就是車子而已，所以我改成這樣。

```
YOLOX > yolox > data > datasets > coco_classes.py > ...
1  #!/usr/bin/env python3
2  # -*- coding:utf-8 -*-
3  # Copyright (c) Megvii, Inc. and its affiliates.
4
5  COCO_CLASSES = [
6      "car"
7  ]
8
```

- YOLOX/exps/example/custom/yolox_s.py:

這裡是需要變更最多的地方。為了要讓 model 知道 train 資料以及 Val 資料的 ground truth，也就是對應的 JSON 檔在哪裡，需要將路徑改掉。此外，因為我們只有一個類別，所以類別數量要改成 1，如果想要改變訓練的 epoch 數量也可以在這裡更改。更動如下所示。

```

class Exp(MyExp):
    def __init__(self):
        super(Exp, self).__init__()
        self.depth = 0.33
        self.width = 0.50
        self.exp_name = os.path.split(os.path.realpath(__file__))[1].split(".")[0]

        # Define yourself dataset path
        self.data_dir = "datasets/HW2_data"
        self.train_ann = "train.json"
        self.val_ann = "val.json"

        self.num_classes = 1

        self.max_epoch = 300
        self.data_num_workers = 4
        self.eval_interval = 1

```

- YOLOX/yolox/data/datasets/coco.py

我在這裡面將訓練資料夾的名稱改變成我的資料夾，如下圖所示：
(第 44 行，name="train"，train 是我的資料夾名稱)

```

class COCODataset(Dataset):
    """
    COCO dataset class.
    """

    def __init__(
        self,
        data_dir=None,
        json_file="instances_train2017.json",
        name="train",
        img_size=(416, 416),
        preproc=None,
        cache=False,
    ):

```

- YOLOX/yolox/exp/yolox_base.py

這裡要把 validate 資料的資料夾名稱改掉如下圖所示：
(第 278 行，把 val2017 改成我們的 validate 資料夾的名字)

```

def get_eval_loader(self, batch_size, is_distributed, testdev=False, legacy=False):
    from yolox.data import COCODataset, ValTransform

    valdataset = COCODataset(
        data_dir=self.data_dir,
        json_file=self.val_ann if not testdev else self.test_ann,
        name="val" if not testdev else "test2017",
        img_size=self.test_size,
        preproc=ValTransform(legacy=legacy),
    )

```

以上步驟完成後就在 YOLOX 的終端機下這個 command line 可以進行訓練了：

python tools/train.py -f exps/example/custom/yolox_s.py -b 10 --fp16 -c yolox_s.pth, yolox_s.pth 是 pretrained model weight 的路徑

II. Explain code and command line(with SE modules):

有加入 SE modules 與沒加入 SE modules 需要更動的地方很少，主要分為以下兩點：

- 加入 se modules 的 class:

我在 Code/SE/YOLOX/yolox/models 下建立一個 SEmodules.py，用來放 SE modules 的 Class，程式如下圖所示：

```

import torch
import torch.nn as nn

class SE(nn.Module):
    def __init__(self, channel, ratio=16):
        super(SE, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
            nn.Linear(channel, channel // ratio, bias=False),
            nn.ReLU(inplace=True),
            nn.Linear(channel // ratio, channel, bias=False),
            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

```

- 加入架構內：

我將我的 se modules 加在 backbone 的部份，分別加在要輸入到 neck 的三個通道（1024，512，256），所以我修改在 Code/SE/YOLOX/yolox/models/yolo_pafpn.py 這個檔案，改變的地方如下所示：

```

82         #se layers
83         self.se_1 = SE(int(in_channels[2] * width))
84         self.se_2 = SE(int(in_channels[1] * width))
85         self.se_3 = SE(int(in_channels[0] * width))
86
87     def forward(self, input):
88         """
89         Args:
90             inputs: input images.
91
92         Returns:
93             Tuple[Tensor]: FPN feature.
94         """
95
96         # backbone
97         out_features = self.backbone(input)
98         features = [out_features[f] for f in self.in_features]
99         [x2, x1, x0] = features
100
101         x0 = self.se_1(x0)
102         x1 = self.se_2(x1)
103         x2 = self.se_3(x2)
104
105         fpn_out0 = self.lateral_conv0(x0) # 1024->512/32
106         f_out0 = self.upsample(fpn_out0) # 512/16
107         f_out0 = torch.cat([f_out0, x1], 1) # 512->1024/16
108         f_out0 = self.C3_p4(f_out0) # 1024->512/16

```

3. How to test my model:

1. 生成 predict txt:

將放有圖片的資料夾放在 Code/Original/YOLOX 或是 Code/SE/YOLOX 這個大資料夾下面，接著進入 Code/Original 或者是 Code/SE 資料夾下的 YOLOX/tools/demo.py，修改名稱：

```

338 if __name__ == "__main__":
339     args = make_parser().parse_args()
340     exp = get_exp(args.exp_file, args.name)
341     name_list = os.listdir("test")
342     for img in name_list:
343         args.path = os.path.join("test", img)
344         main(exp, args)

```

將 341 行以及 343 行的“test”改成**放有圖片資料夾名稱**，並且下這個 Command line(在 YOLOX):

```
python tools/demo.py image -f exps/example/custom/yolox_s.py -c
best_ckpt.pth --device [gpu]
```

然後就會生成 txt 檔在**放有圖片資料夾**裡面，要將這些 txt 檔放到 Code/Original 或者是 Code/SE 下的 Object-Detection-Metrics/detections 下面，才能進一步去算 mAP。

2. 生成 ground truth txt:

將**含有原 test label**的資料夾放在 Code/Original 或者是 Code/SE 下面，然後將打開 classify.py，將第 4 行以及第 8 行的“val_labels”改成**含有原 test label**的資料夾的名稱，接著在終端打 python classify.py，就會產生一個名叫 gt_labels 的資料夾，裡面會含有格式轉換完畢的 txt 檔

3. Evaluate:

將 Code/Original/gt_labels 內的所有 txt 放到 Code/Original 或者是 Code/SE 下的 Object-Detection-Metrics/groundtruths 下，然後在 Object-Detection-Metrics 的終端下這個 command line:

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

就可以得到 mAP 以及 AP 的值了。

4. Notice:

如果在生成 predict txt(第一步驟) 有報一個找不到 yolox modules 的錯，就將 demo.py 的第 14 行註解刪掉，並在雙引號內加入 YOLOX 的絕對路徑，如下所示:

```
#sys.path.append(r"/home/james/NYCU/111fall/videostream/LAB2/HW2_311605011/Code/Original/YOLOX")
```

4. Validation results:

● Without SE:

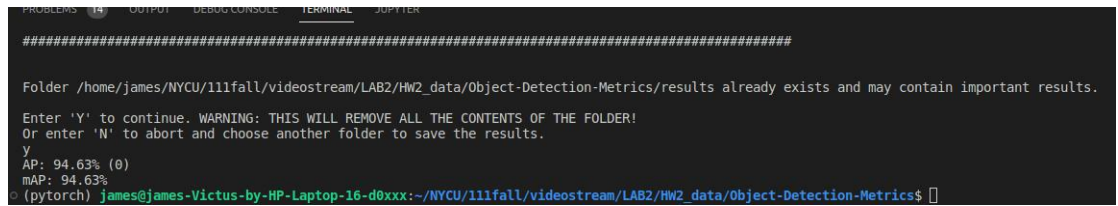
```

#####

Folder /home/james/NYCU/111fall/videostream/LAB2/HW2_data/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: 94.57% (0)
mAP: 94.57%
(pytorch) james@james-Victus-by-HP-Laptop-16-d0xxx:~/NYCU/111fall/videostream/LAB2/HW2_data/Object-Detection-Metrics$

```

- With SE



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
#####

Folder /home/james/NYCU/111fall/videostream/LAB2/HW2_data/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: 94.63% (0)
mAP: 94.63%
(pytorch) james@james-Victus-by-HP-Laptop-16-d0xxx:~/NYCU/111fall/videostream/LAB2/HW2_data/Object-Detection-Metrics$
```

5. Discussion

- Problems I encountered:

這次的作業我遇到最大的問題就是我一開始在將 **bounding box** 的資訊取出後，拿去與 **ground truth** 做比較，但是 **AP** 一直是 0，**debug** 了非常久，最後才發現拿出來的 **bounding box** 要是 **resize** 過的，不然他的 **size** 會是不對的。

- Which layer I add SE modules:

我將我的 **se modules** 加在 **backbone** 的部份，分別加在要輸入到 **neck** 的三個通道 (1024, 512, 256)，最後的結果比沒有加上 **SE** 好了 0.07%，其實沒有增加很多，我覺得可以也在 **neck** 或是 **head** 也加看看，也許成果會更明顯。

6. Environment setup:

1. create an environment(python=3.9)
2. conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 cudatoolkit=11.6 -c pytorch -c conda-forge
3. cd Code/Origin/YOLOX
4. pip3 install -v -e .

