DLCV HW2

鄭承昀 機械碩一 R07522823

1. (5%) Print the network architecture of your YoloV1-vgg16bn model and describe your training config. (optimizer,batch size….and so on)

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU(inplace)
    (10): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (12): ReLU(inplace)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (14): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (16): ReLU(inplace)
    (17): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (19): ReLU(inplace)
    (20): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (22): ReLU(inplace)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (26): ReLU(inplace)
    (27): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (28): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (29): ReLU(inplace)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (32): ReLU(inplace)
    (33): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (35): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (36): ReLU(inplace)
    (37): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (38): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (39): ReLU(inplace)
    (40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (42): ReLU(inplace)
    (43): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (yolo): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace)
    (2): Dropout(p=0.5)
    (3): Linear(in_features=4096, out_features=1274, bias=True)
  )
)
```

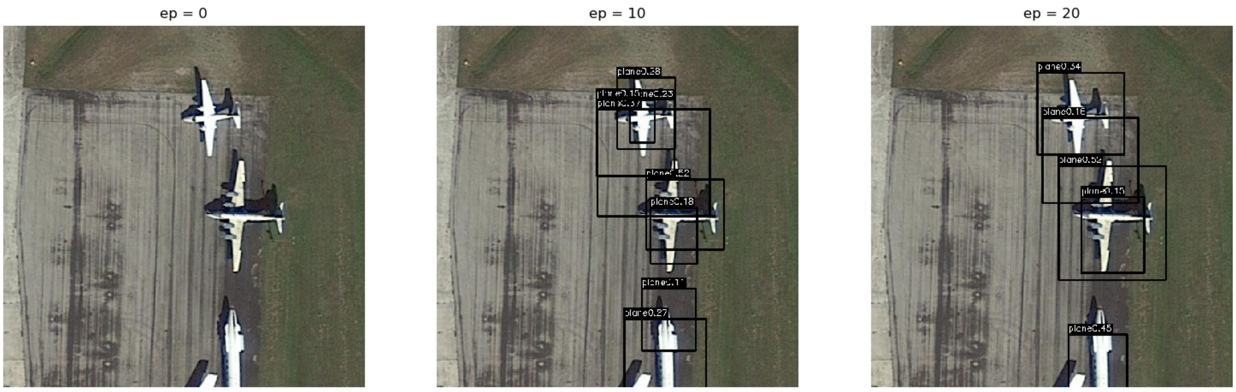
- batch size: 8
- epoch:100
- learning rate: lr = 0.001 → after 30 epochs = 0.0001 → after 40 epochs = 0.00001

- optimizer:stochastic gradient descent(SGD)
- momentum:0.9
- weight decay:0.0005

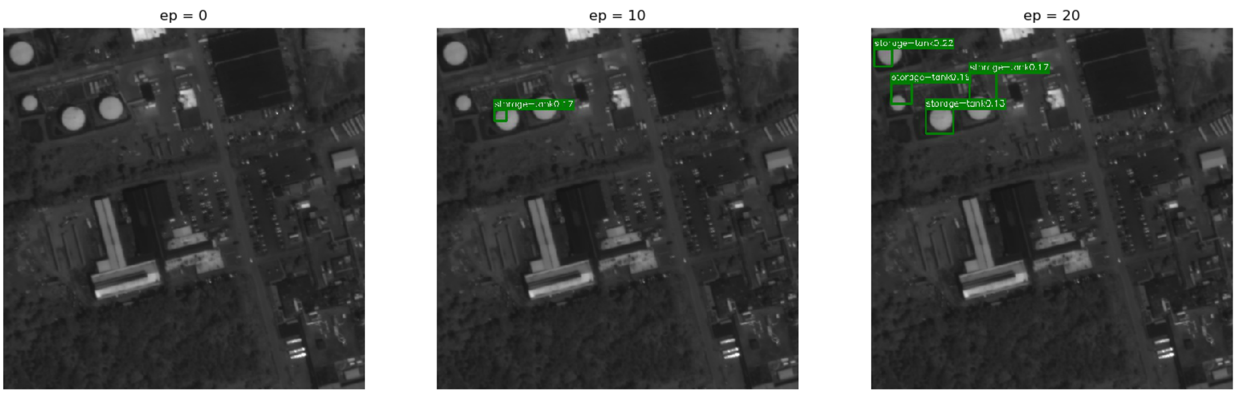
2. (10%) Show the predicted bbox image of “val1500/0076.jpg”, “val1500/0086.jpg”, “val1500/0907.jpg” during the early, middle, and the final stage during the training stage. (1st, 10th, 20th epoch)

    - val1500/0076.jpg

    - val1500/0086.jpg



    - val1500/0907.jpg



3. (10%) Implement an improved model which performs better than your baseline model. Print the network architecture of this model and describe it.

    這邊我使用ResNet50來取代掉原本的VGG16來進行特徵的抽取。因為ResNet的model太大了，我就簡單描述一下我的架構。一張3* 448* 448的圖片丟入ResNet，在ResNet最後將average pooling改成再一

個Conv layer，將其轉成7* 7 *26。

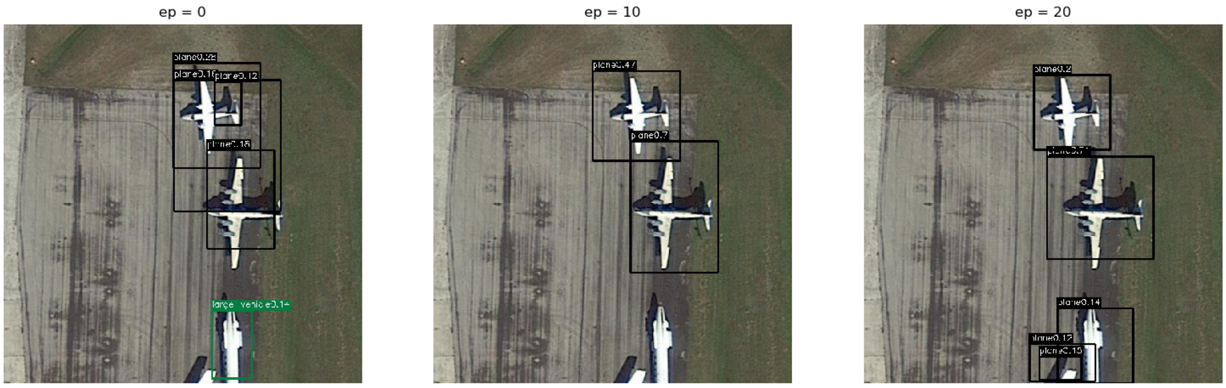
(1,3,448,448)→ResNet50→(1,2048,7,7)→Conv→(1,26,7,7)→sigmoid+resize→(1,7,7,26)

- batch size: 8
- epoch:50
- learning rate: lr = 0.001 → after 30 epochs = 0.0001 → after 40 epochs = 0.00001
- optimizer:stochastic gradient descent(SGD)
- momentum:0.9
- weight decay:0.0005

4. (10%) Show the predicted bbox image of "val1500/0076.jpg", "val1500/0086.jpg", "val1500/0907.jpg" during the early, middle, and the final stage during the training process of this improved model.

- val1500/0076.jpg

- val1500/0086.jpg



- val1500/0907.jpg

5. (15%) Report mAP score of both models on the validation set. Discuss the reason why the improved model performs better than the baseline one. You may conduct some experiments and show some evidences to support your reasoning.

VGG16

```
classname: plane
ap:  0.31367720108120895
classname: baseball-diamond
ap:  0.0
classname: bridge
ap:  0.12609970674486803
classname: ground-track-field
ap:  0.045454545454545456
classname: small-vehicle
ap:  0.0303030303030303
classname: large-vehicle
ap:  0.12370925288519943
classname: ship
ap:  0.09090909090909091
classname: tennis-court
ap:  0.5200592946142817
classname: basketball-court
ap:  0.09090909090909091
classname: storage-tank
ap:  0.025974025974025972
classname: soccer-ball-field
ap:  0.15151515151515152
classname: roundabout
ap:  0.0
classname: harbor
ap:  0.08901575122907084
classname: swimming-pool
ap:  0.05994729907773386
classname: helicopter
ap:  0.0
classname: container-crane
ap:  0.0
map: 0.10422334004358111
```

ResNet

```
classname: plane
ap:  0.6179549241579183
classname: baseball-diamond
ap:  0.40269963896446115
classname: bridge
ap:  0.22946760841497682
classname: ground-track-field
ap:  0.006993006993006993
classname: small-vehicle
ap:  0.14661221255052767
classname: large-vehicle
ap:  0.2565234830223309
classname: ship
ap:  0.09610964256981956
classname: tennis-court
ap:  0.7694122495972866
classname: basketball-court
ap:  0.2703349282296651
classname: storage-tank
ap:  0.24757275727572758
classname: soccer-ball-field
ap:  0.41626794258373206
classname: roundabout
ap:  0.08333333333333333
classname: harbor
ap:  0.28465857294315544
classname: swimming-pool
ap:  0.31788433340852695
classname: helicopter
ap:  0.0
classname: container-crane
ap:  0.0
map: 0.2591140396277793
```

我們可以發現只是將抽取特徵的架構由VGG16改成ResNet50其mAP就有顯著的提昇。其原因可能為：

- ResNet比VGG16來的深，所以model就可以從圖片中找到更high-level的feature來進行辨識。這個部份可以從VGG16有很多種類的辨識正確率為零，像是baseball-diamond和roundabout，但是在ResNet的架構下這兩個都有成功辨識出來，其原因應該就是ResNet成功找到能辨識這兩種類別的feature。
- ResNet最後我用Conv來取代掉的fully connected layer。這樣可以保留空間中和圖像的資訊。
- ResNet可以避免當weight小於0的時候node沒用的問題。

6. bonus (5%) Which classes prediction perform worse than others? Why? You should describe and analyze it.

從上面的mAP可以發現有好幾類的ap是零。我去把有這幾類的照片挑出來，並將預測的bounding box畫出來。從中我推測出幾個可能的原因：

- 其中baseball-diamond、roundabout這兩類我發現大部分都沒有成功預測出出來。其原因可能為這這兩個種類都沒辦法找到決定性的特徵。在ResNet這比較多層的架構中就能成功找到feature來做預測。
- container-crane這一類也是基本上都沒預測出來，可能因為training data太少的問題。
- helicopter的部份我發現它很容易將其辨識成飛機。其原因可能為有直昇機的data相對比較少一點，然後直昇機跟飛機又都有相似的特徵。

7. 參考資料

https://github.com/xiongzihua/pytorch-YOLO-v1

主要都是參考這份github。原本自己寫的loss function mAP只能到7%左右，所以後來使用這份的loss function去做修改。而NMS的部份，我發現自己寫的跑得速度太慢了，後來也改由使用這份的NMS去做修改。