Name: 李俊賢　Dep.:資工碩一　Student ID:R07922103

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): Dropout(p=0.5)
    (2): Linear(in_features=4096, out_features=1274, bias=True)
  )
)
```

Model：Model 主要是在 VGG16bn 之後接上一層 fully connected layer(7*7*512 to 4096)、一層 dropout(p=0.5)、及最後再接上一層 fully connected layer(4096 to 7*7*26)，之後再 rezise 成 7*7*26

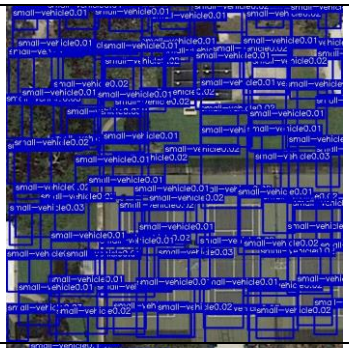
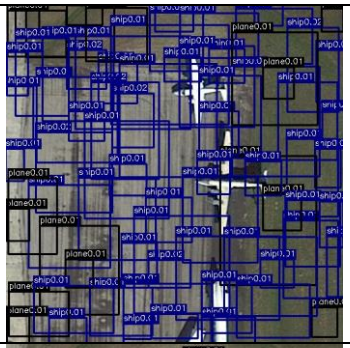
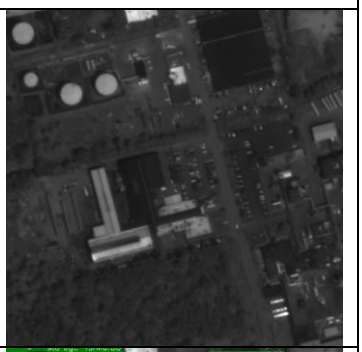
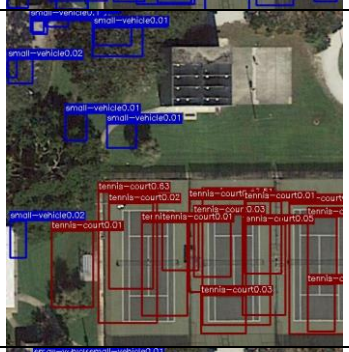
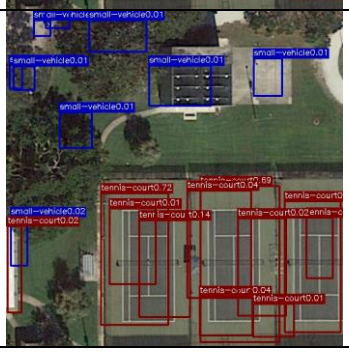
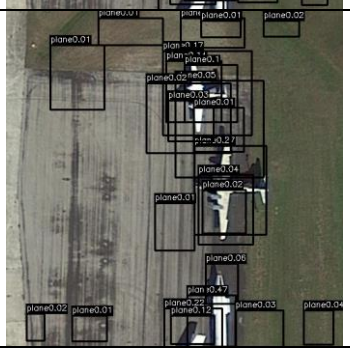
Optimizer：Optimizer 是使用 SGD，其中 lr=0.001、momentum=0.9、weight_decay=0.0001

Batch Size：32

NMS_confidence_threshold：0.01

NMS_IOU_threshold：0.5

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. (For example, results of 1st, 10th, 20th epoch)

image / epoch	val1500/0076.jpg	val1500/0086.jpg	val1500/0907.jpg
epoch1 (mAP=0.0011)			
epoch81 (mAP=0.0761)			
epoch176 (mAP=0.1134)			

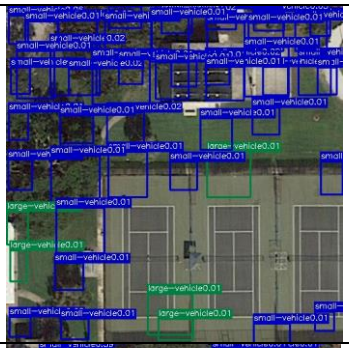
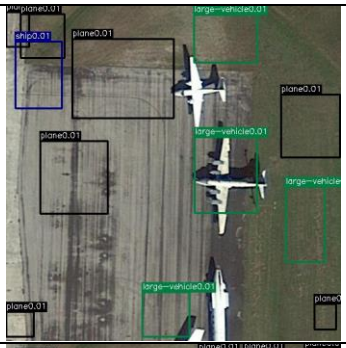
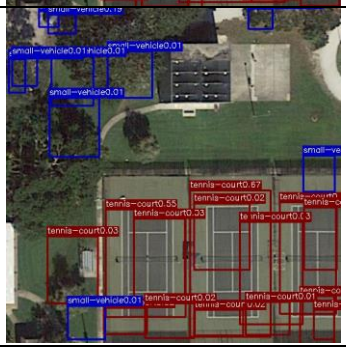
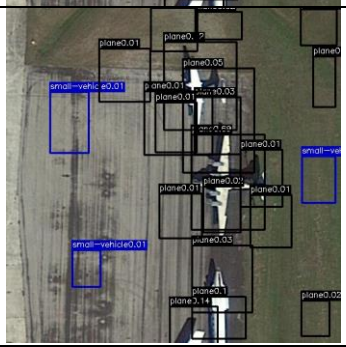
3. (10%) Implement an improved model which performs better than your baseline model. Print the network architecture of this model and describe it.

Improved Model：Improved Model 基本架構跟 Base Model 一樣，VGG16bn 沒有做更動，yolo 部分也沒更動，只有在 VGG16bn 與 yolo 層之間，加了一個 1*1 的 convolution layer，其中該 layer 包含 Conv2d(512, 128, kernel_size=(1,1), stride=(1,1))、BatchNorm2d(128)、ReLU(True)。如下圖有我們的完整架構。
Optimizer：使用 Adam，lr=0.0001、weight_decay=0.00001
Batch：32
NMS_confidence_threshold：0.01
NMS_IOU_threshold：0.5

```
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)
  )
  (conv): Sequential(
    (0): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace)
  )
  (yolo): Sequential(
    (0): Linear(in_features=6272, out_features=4096, bias=True)
    (1): Dropout(p=0.5)
    (2): Linear(in_features=4096, out_features=1274, bias=True)
  )
)
```

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.

同樣利用上述參數，產生出如以下表格。

image / epoch	val1500/0076.jpg	val1500/0086.jpg	val1500/0907.jpg
epoch1 (mAP=0.0001)			
epoch51 (mAP=0.1278)			
epoch98 (mAP=0.1838)			

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.

Model / Class	Base-Model (mAP=0.1134)	Improved-Model (mAP=0.1838)
plane (8723 筆)	0.222952	**0.431699**
baseball-diamond (515 筆)	0.147727	**0.156064**
bridge (2114 筆)	0.046946	**0.161032**
ground-track-field (621 筆)	**0.111888**	0.100313
small-vehicle (116228 筆)	0.090909	**0.095783**

large-vehicle (23746 筆)	0.117607	**0.157662**
ship (34585 筆)	**0.090909**	0.064064
tennis-court (3279 筆)	0.491936	**0.639486**
basketball-court (661 筆)	0.068181	**0.218181**
storage-tank (5199 筆)	**0.098144**	0.046161
soccer-ball-field (590 筆)	0.089610	**0.407404**
roundabout (537 筆)	0.045454	**0.103448**
harbor (7457 筆)	0.078532	**0.139161**
swimming-pool (1977 筆)	0.114164	**0.221705**
helicopter (434 筆)	0.0	0.0
container-crane (136 筆)	0.0	0.0

從以上表格看得出來，Improved Model 在大多數 class 的情況下表現都比 Base Model 好很多，而如此多數的增長，我們可以推測 Improved Model 比較不 overfit 原本的 testing data。而其證據是根據[1]論文中所提及，convolution 1*1 的卷積不僅可以用於降低資料維度，同時也能學習維度之間的關係性。換句話作解釋就是，今天我原本的維度是 512*7*7，而我利用 128 個 1*1 的 kernel 去降維成 128*7*7，而在這降維之中，每一個 output 7*7 都是我一個 kernel 跟原本 512 個維度去做不同權重所累加起來的結果，也就是說假設今天我的 512 維度中有某個維度特別 overfit 原本的 data，我可以利用這 128 個 1*1kernel 去做不同程度的權重並疊加，如此一來的成果就是能降低該 overfit 維度的影響力，並間接提升其他維度的權重，讓他們之間達到一個平衡有關連的狀態，所以才說 1*1 的 convolution kernel 可以學到不同維度之間的關係性，並同時藉由不同權重減少 overfitting 的可能性。而在 Improved Model 中，剛好就是加入了 1*1 的 convolution layer，並在其後接上一個 BatchNorm 及 ReLU 讓整體資料的影響力更鮮明，並不那麼 overfit，所以 Improved Model 才能在 mAP 上有較好的表現結果。

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

延續第五小題的表格，我們可以看到 helicopter 跟 container-crane 的 AP 都比其他類別來得低上許多，AP 根本就只有 0。而其中 container-crane 的 ground truth data 只有 136 筆，相較於每個 class 筆數平均下來的 12925 筆低上了許多，這是我認為 container-crane 訓練不好的原因，因為筆數太少。而 helicopter 的 ground truth data 筆數有 434 筆，雖然也是相對較少，但其他與之相近的 baseball-diamond 有 515 筆、basketball-court 有

661 筆、roundabout 有 537 筆、soccer-ball-field 有 590 筆、ground-track-field 有 621 筆，這些 class 都訓練得比 helicopter 還要好上許多，我認為其原因是因為 helicopter 在 ground truth data 中與地板的顏色太過相近，helicopter 幾乎就是灰色的小蚊子，而地板也幾乎都是灰色的水泥地或是滑行跑道，他們之間的顏色太過相近以至於 Model 難以分析出哪邊才是 helicopter。

而其他 AP 不到 0.1 的 class 我認為都是筆數不夠多或是物件太小，例如 small-vehicle 明明有大量的 data，但卻因為物件小、密集度高，無法用 7*7 如此大的 grid 去判斷細小的物件，也就是說可能一個 grid 裡面就包含了好幾個 small-vehicle 以至於該 grid 無法正確預測出 small-vehicle 的正確位子。而其他能高於 0.1AP 的 class 大多都是有足夠訓練 data，或是物體本身夠大夠鮮明，足以使用 7*7 大小的 grid 去代表，例如 large-vehicle 以及 tennis-court。

7. Reference

[1] M. Lin, Q. Chen, and S. Yan. Network in network. arXiv:1312.4400, 2013.

8. Collaborators

我整份報告有疑問的地方，都是與我 ImLab 的實驗室同學們一起討論，我們僅限於題目定義與演算法的討論，實作的 code 都是各自寫各自的。以下是同學們的 ID，b04901190、R07922002、R07922024、R07922043、R97922120