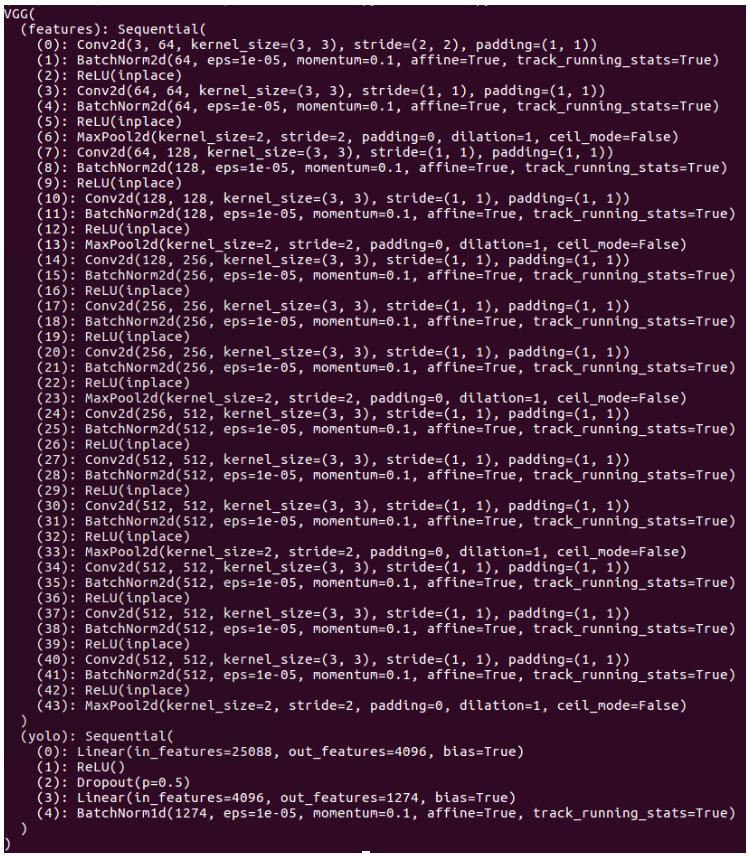
Deep learning for computer vision hw2

Name: 余奇安 Dep.:電機碩一 Student ID:R07921010

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

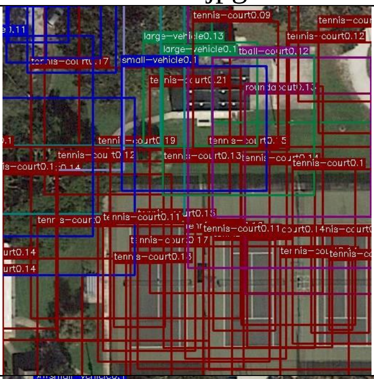
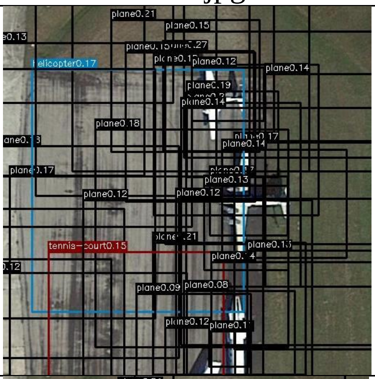
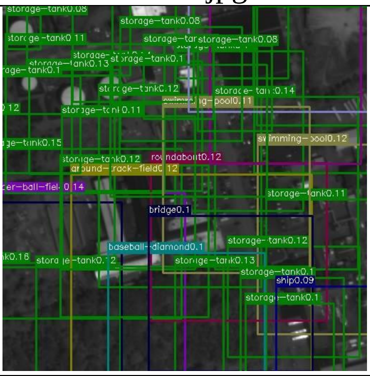
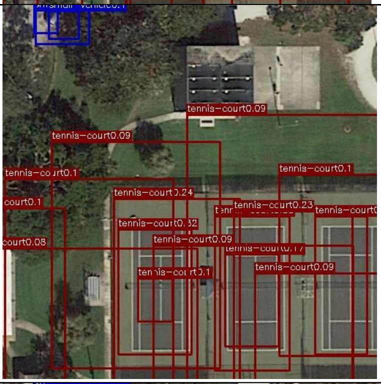
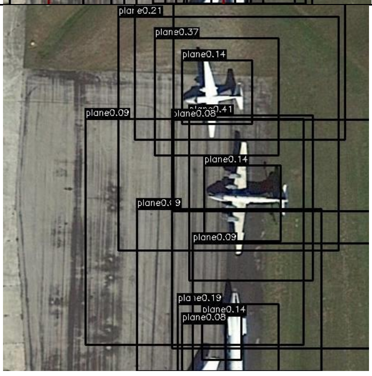
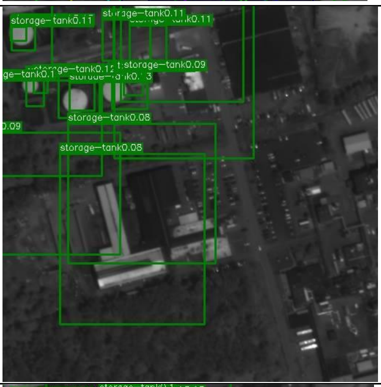
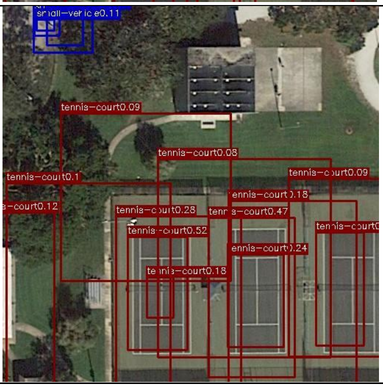
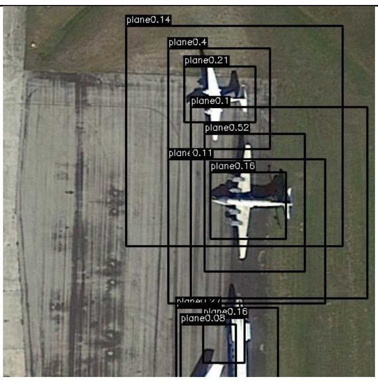
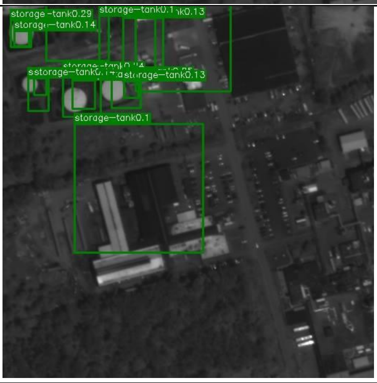


```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): 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()
    (2): Dropout(p=0.5)
    (3): Linear(in_features=4096, out_features=1274, bias=True)
    (4): BatchNorm1d(1274, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
```

I use SGD as optimizer with batch size 32, momentum 0.9, learning rate 0.001 for the first 30 epochs, 0.0001 for epoch 30~39, 0.00001 for epoch 30~60.

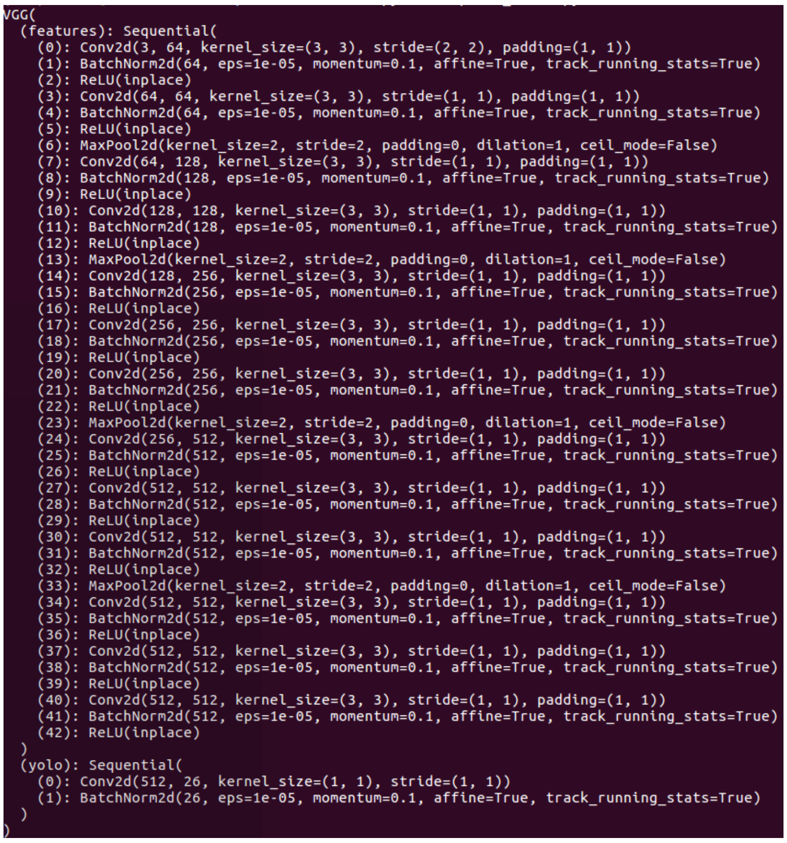
For the fully connect layer, I use ReLU as activation function and dropout(rate=0.5), linear with input size 4096, output size 1274, and eventually BatchNormal.

2. 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)

	0076.jpg	0086.jpg	0907.jpg	Map
Early @epoch5				0.0131
Middle @epoch20				0.0663
Final @epoch40				0.0927

We can see that the tennis court, airplane which have specific shape can be recognized correctly. However, lots of overlap bounding box still exist.

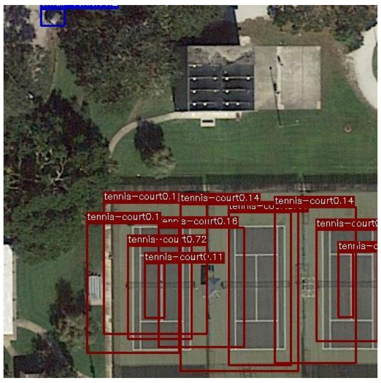
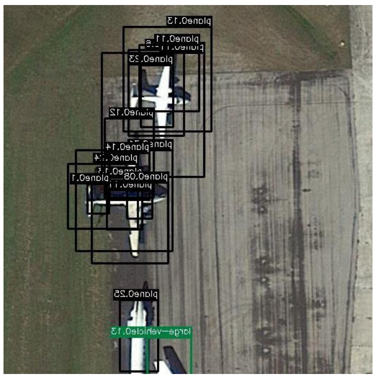
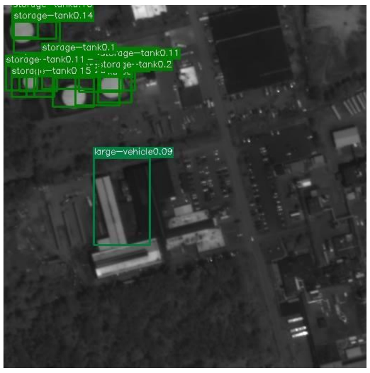
3. **Implement an improved model which performs better than your baseline model. Print the network architecture of this model and describe it.**

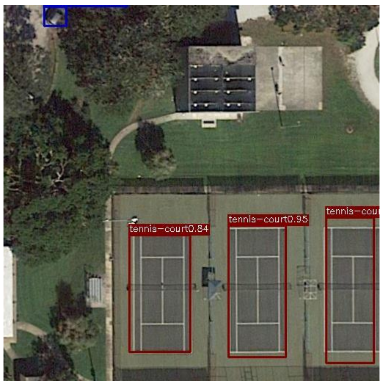
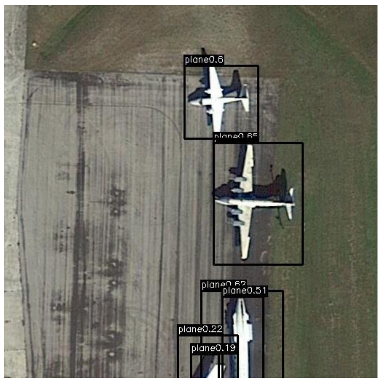
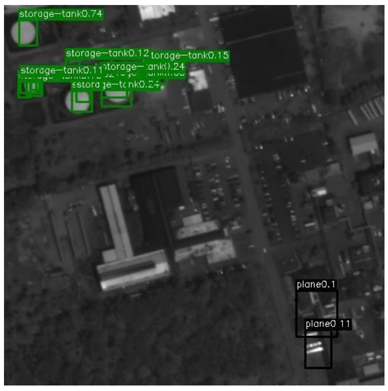
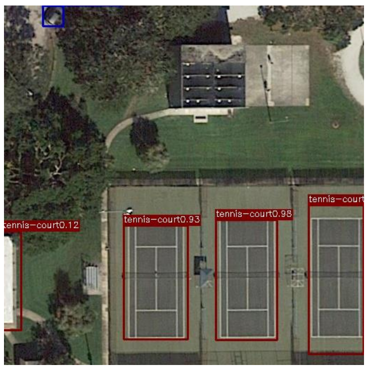
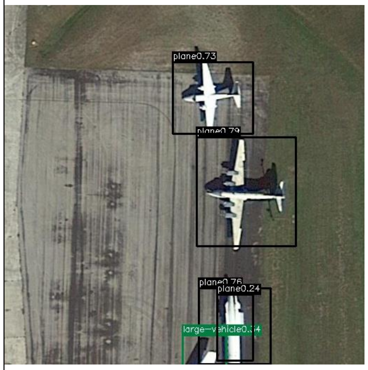
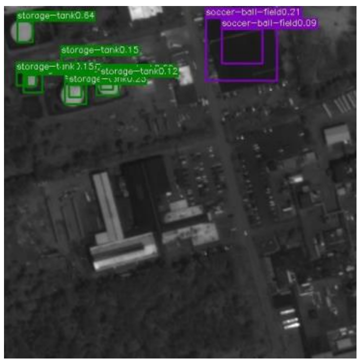


```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): 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)
  )
  (yolo): Sequential(
    (0): Conv2d(512, 26, kernel_size=(1, 1), stride=(1, 1))
    (1): BatchNorm2d(26, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
```

I replace the fully connected layer with Conv2d. Because I found that lots of object in images are very small, so I cut the grid to 14*14. Therefore, the input of Conv2d is 512 * 14 * 14, with kernel size (1, 1), stride (1,1) depth 26. The following is BatchNorm2d which normalizes the data.

4. **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.**

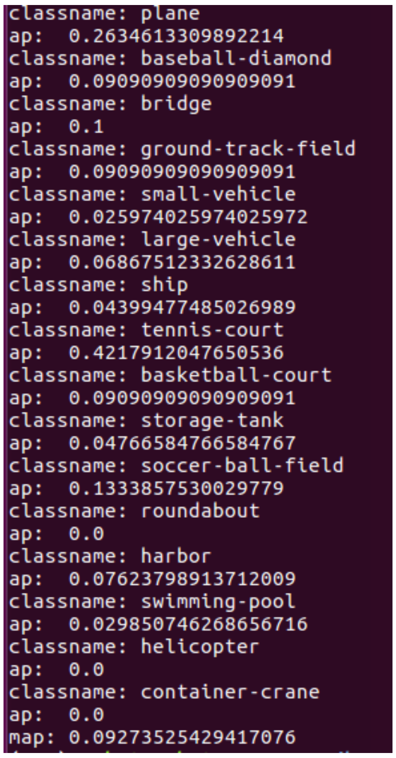
	0076.jpg	0086.jpg	0907.jpg	Map
Early @epoch1				0.12

Middle @epoch5				0.28
Final @epoch8				0.30

For the improve model, we can see that the airplane and tennis court have clear bounding box, and the confidence can reach 0.98.

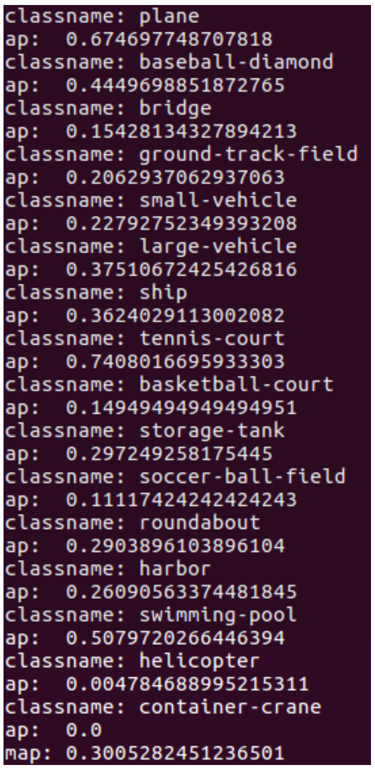
5. **Report mAP score of both models on the validation set. Discuss the reason why the improved model performs better than the baseline one. You may conduct some experiments and show some evidences to support your reasoning.**

Basic model @epoch 40:



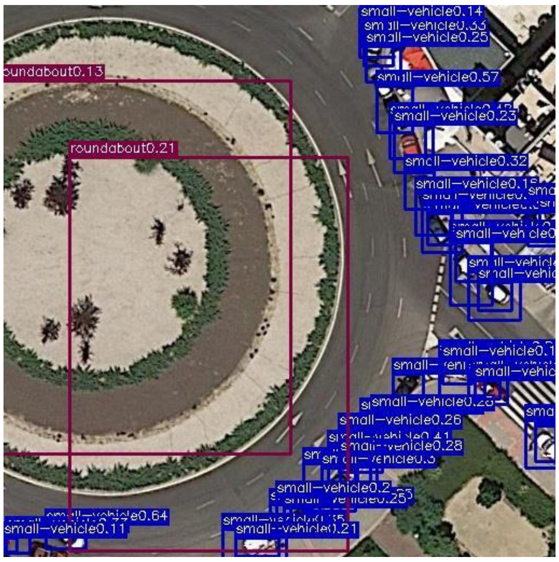
```
classname: plane
ap:  0.2634613309892214
classname: baseball-diamond
ap:  0.09090909090909091
classname: bridge
ap:  0.1
classname: ground-track-field
ap:  0.09090909090909091
classname: small-vehicle
ap:  0.025974025974025972
classname: large-vehicle
ap:  0.06867512332628611
classname: ship
ap:  0.04399477485026989
classname: tennis-court
ap:  0.4217912047650536
classname: basketball-court
ap:  0.09090909090909091
classname: storage-tank
ap:  0.04766584766584767
classname: soccer-ball-field
ap:  0.1333857530029779
classname: roundabout
ap:  0.0
classname: harbor
ap:  0.07623798913712009
classname: swimming-pool
ap:  0.029850746268656716
classname: helicopter
ap:  0.0
classname: container-crane
ap:  0.0
map: 0.09273525429417076
```

Improve model @epoch 8:



```
classname: plane
ap:  0.674697748707818
classname: baseball-diamond
ap:  0.4449698851872765
classname: bridge
ap:  0.15428134327894213
classname: ground-track-field
ap:  0.2062937062937063
classname: small-vehicle
ap:  0.22792752349393208
classname: large-vehicle
ap:  0.37510672425426816
classname: ship
ap:  0.3624029113002082
classname: tennis-court
ap:  0.7408016695933303
classname: basketball-court
ap:  0.14949494949494951
classname: storage-tank
ap:  0.297249258175445
classname: soccer-ball-field
ap:  0.11117424242424243
classname: roundabout
ap:  0.2903896103896104
classname: harbor
ap:  0.26090563374481845
classname: swimming-pool
ap:  0.5079720266446394
classname: helicopter
ap:  0.004784688995215311
classname: container-crane
ap:  0.0
map: 0.3005282451236501
```

After some observation of dataset, we can see that lots of object is very small, so I decide to make the grid finer to prevent there are two ground true object in the same grid.

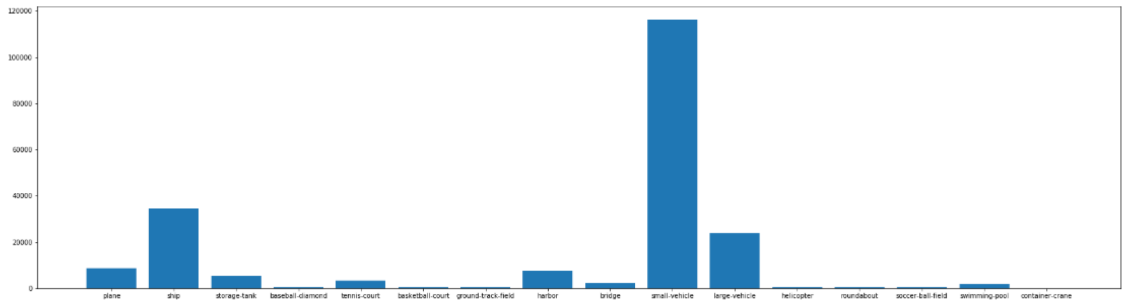


Cluster small car in image can be recognized in 14 * 14 grid.

Besides, I replace fully connected layer with convolution kernel to transform 512 * 14 * 14 vector to 26 * 14 * 14 dimension. The convolution layer can preserve not only image feature information, but also position information. With less weigh and more features extracted from image, the model performance has a great improvement. Even some class like roundabout which is 0 a.p can reach 0.29 in improve model.

6. **Which classes prediction perform worse than others? Why? You should describe and analyze it.**

Under the base line case, we can see that helicopter, container-crane, roundabout is relative low. After seeing the image, I view it reasonable because lots kind of these class is blur and ambiguous. Besides, the sample of these class is relative sparse and rare.



The distribution of each class in training data

```
'plane': 8723,
'ship': 34585,
'storage-tank': 5199,
'baseball-diamond': 515,
'tennis-court': 3279,
'basketball-court': 661,
'ground-track-field': 621,
'harbor': 7457,
'bridge': 2114,
'small-vehicle': 116228,
'large-vehicle': 23746,
'helicopter': 434,
'roundabout': 537,
'soccer-ball-field': 590,
'swimming-pool': 1977,
'container-crane': 136}
```

The object is at the edge of image

Because the helicopter, roundabout, container-crane is the less three sample in training data and bad image quality, the prediction result is the worst three.

7. Reference:

https://github.com/xiongzihua/pytorch-YOLO-v1
https://www.itread01.com/content/1541808910.html
深度學習之---yolov1,v2,v3 詳解

https://ceiba.ntu.edu.tw/course/449bf4/content/Pytorch_Tutorial_1.pdf
pytorch tutorial

https://colab.research.google.com/drive/1wynOtNlGuSh7WdJ4pwcZ14lezMj9DVEy?fbclid=IwAR3qE-TuHfH4wijURr1UNmLZDBF9ox5GWFHkUfDnDq22gQ3bXDNWy_GVxes

Pytorch Practice: Classifying MNIST images