Name: 李尚倫　Dep.:電機碩一　Student ID:R07921001

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

My model structure:

```
# images in trainset: 15000
# images in testset: 1500
Image tensor in each batch: torch.Size([16, 3, 448, 448]) torch.float32
Label tensor in each batch: torch.Size([16, 7, 7, 26]) torch.float32
Device used: cuda:1
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)
  )
```

My validation loss after each epoch (total=100)

```
start training at:2019-04-18 22:56:26.199555
ep: 0 validation avg loss:10.356362890689931
ep: 1 validation avg loss:10.095357182178091
ep: 2 validation avg loss:8.101595389082076
ep: 3 validation avg loss:7.774121274339392
ep: 4 validation avg loss:7.458845290731876
ep: 5 validation avg loss:7.241731292389809
ep: 6 validation avg loss:7.058796752006449
ep: 7 validation avg loss:7.062682652727086
ep: 8 validation avg loss:7.006893988619459
ep: 9 validation avg loss:6.87901689460937
ep: 10 validation avg loss:6.6704280807616865
ep: 11 validation avg loss:6.639561348773063
ep: 12 validation avg loss:6.736094066437254
ep: 13 validation avg loss:6.613749264402593
ep: 14 validation avg loss:6.622309297957319
ep: 15 validation avg loss:6.611828272647046
ep: 16 validation avg loss:6.657504928872941
ep: 17 validation avg loss:6.538890232431128
ep: 18 validation avg loss:6.617375406813114
ep: 19 validation avg loss:6.596697377397659
ep: 20 validation avg loss:6.570734668285288
ep: 21 validation avg loss:6.57210753889794
ep: 22 validation avg loss:6.537826552035961
ep: 23 validation avg loss:6.511554888588317
ep: 24 validation avg loss:6.507252440807667
ep: 25 validation avg loss:6.547529488801956
ep: 26 validation avg loss:6.4599869
ep: 27 validation avg loss:6.4407971
ep: 28 validation avg loss:6.4699071
ep: 29 validation avg loss:6.4866299
ep: 30 validation avg loss:6.4601150
ep: 31 validation avg loss:6.4574133
ep: 32 validation avg loss:6.5216534
ep: 33 validation avg loss:6.4591988
ep: 34 validation avg loss:6.4108255
ep: 35 validation avg loss:6.4301283
ep: 36 validation avg loss:6.3874069
ep: 37 validation avg loss:6.4050627
ep: 38 validation avg loss:6.4467247
ep: 39 validation avg loss:6.4031966
ep: 40 validation avg loss:6.3803141
ep: 41 validation avg loss:6.3376551
ep: 42 validation avg loss:6.3775956
ep: 43 validation avg loss:6.3855392
ep: 44 validation avg loss:6.3295830
ep: 45 validation avg loss:6.3401954
ep: 46 validation avg loss:6.3276745
ep: 47 validation avg loss:6.2986236
ep: 48 validation avg loss:6.3476623
ep: 49 validation avg loss:6.3445273
ep: 50 validation avg loss:6.3240947
ep: 51 validation avg loss:6.252743386841835
ep: 52 validation avg loss:6.304935937232166
ep: 53 validation avg loss:6.307101659952326
ep: 54 validation avg loss:6.279629586329273
ep: 55 validation avg loss:6.285146495763292
ep: 56 validation avg loss:6.266564749022748
ep: 57 validation avg loss:6.253861858489666
ep: 58 validation avg loss:6.326019619373565
ep: 59 validation avg loss:6.291693913318017
ep: 60 validation avg loss:6.257302588604866
ep: 61 validation avg loss:6.290519491789189
ep: 62 validation avg loss:6.322128481157704
ep: 63 validation avg loss:6.282978194825192
ep: 64 validation avg loss:6.286127949014623
ep: 65 validation avg loss:6.255292488539473
ep: 66 validation avg loss:6.294270267511936
ep: 67 validation avg loss:6.3070366446008075
ep: 68 validation avg loss:6.245682438637348
ep: 69 validation avg loss:6.261626167500273
ep: 70 validation avg loss:6.2081703562685781
ep: 71 validation avg loss:6.179791452402764
ep: 72 validation avg loss:6.2119936809894885
ep: 73 validation avg loss:6.201217323541641
ep: 74 validation avg loss:6.197967041046061
ep: 75 validation avg loss:6.222665478574469
ep: 76 validation avg loss:6.1902280361094375
ep: 77 validation avg loss:6.177155426208009
ep: 78 validation avg loss:6.1833011723579245
ep: 79 validation avg loss:6.177338477144851
ep: 80 validation avg loss:6.168254052704953
ep: 81 validation avg loss:6.18616802007594
ep: 82 validation avg loss:6.180262013318691
ep: 83 validation avg loss:6.163080668829857
ep: 84 validation avg loss:6.190401676487415
ep: 85 validation avg loss:6.231179022408546
ep: 86 validation avg loss:6.182661835183489
ep: 87 validation avg loss:6.192373303022791
ep: 88 validation avg loss:6.188571841158765
ep: 89 validation avg loss:6.173107482651447
ep: 90 validation avg loss:6.165599903527727
ep: 91 validation avg loss:6.187912245380118
ep: 92 validation avg loss:6.170951772243418
ep: 93 validation avg loss:6.175140334570662
ep: 94 validation avg loss:6.177023880025174
ep: 95 validation avg loss:6.185619273718367
ep: 96 validation avg loss:6.196088489065779
ep: 97 validation avg loss:6.1840486582359
ep: 98 validation avg loss:6.190824881513068
ep: 99 validation avg loss:6.174271031897119
end training at:2019-04-19 10:34:10.818688
```

My final mAP: 0.09978

```
(robotpower) ➜ hw2-shannon112 git:(mas
al/val1500/labelpre ../hw2_train_val/va
classname: plane
ap:  0.23962525720150218
classname: baseball-diamond
ap:  0.09090909090909091
classname: bridge
ap:  0.09090909090909091
classname: ground-track-field
ap:  0.09090909090909091
classname: small-vehicle
ap:  0.017825311942959002
classname: large-vehicle
ap:  0.10816752775640322
classname: ship
ap:  0.09090909090909091
classname: tennis-court
ap:  0.39744212124531275
```

```
ap:  0.39744212124531275
classname: basketball-court
ap:  0.18181818181818182
classname: storage-tank
ap:  0.09090909090909091
classname: soccer-ball-field
ap:  0.057768318637883856
classname: roundabout
ap:  0.0
classname: harbor
ap:  0.0938838648262732
classname: swimming-pool
ap:  0.045454545454545456
classname: helicopter
ap:  0.0
classname: container-crane
ap:  0.0
map: 0.09978316146428226
```

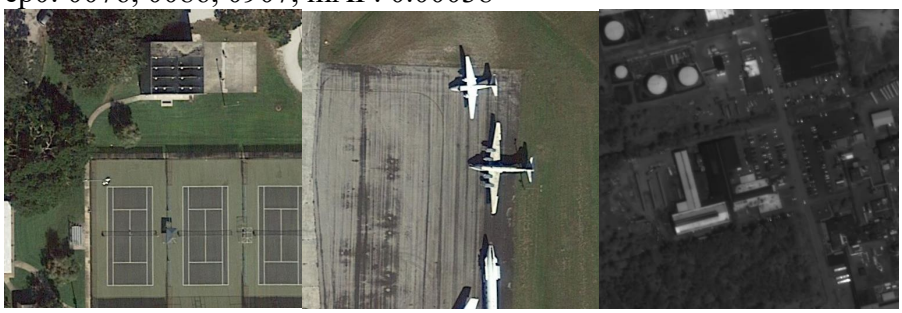
Detail of my training config:

model	baseline model
net	Vgg + nn + relu(T) + dropout + nn
optimizer	Torch.optim.SGD

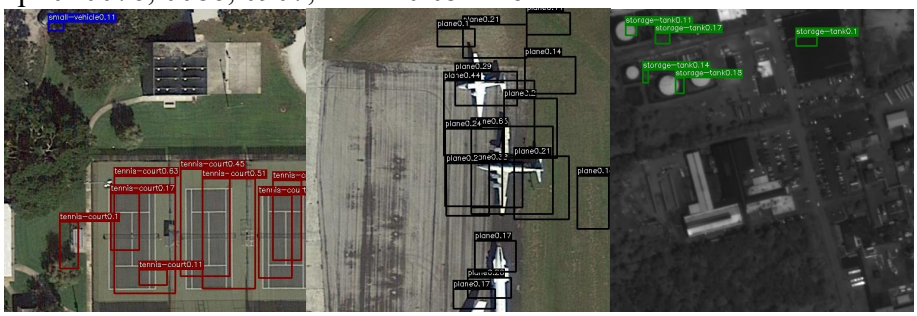
Total epoch	100
final min validation avg loss	ep83, 6.16308
max mAP value	0. 09978
file name	model.pth
yoloLoss	5, 0.05
batchsize	16
Learning rate	ep=0: 0.002; ep=10: 0.001; ep=70: 0.0005

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

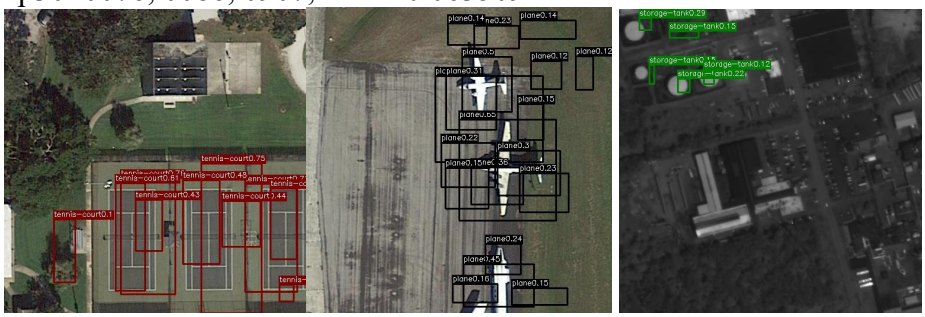
ep0: 0076, 0086, 0907, mAP: 0.00038



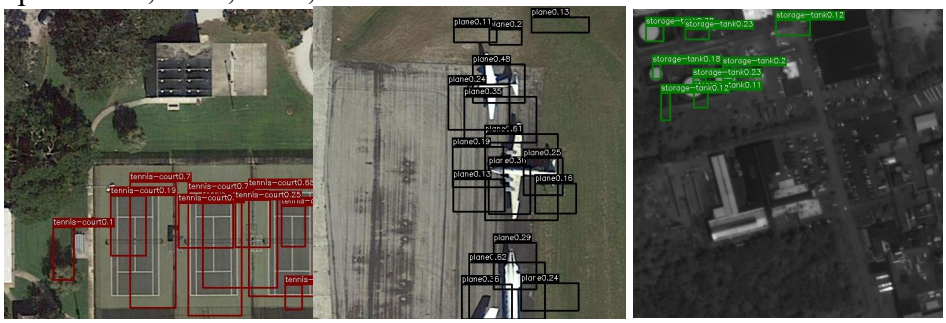
ep10: 0076, 0086, 0907, mAP: 0.054226



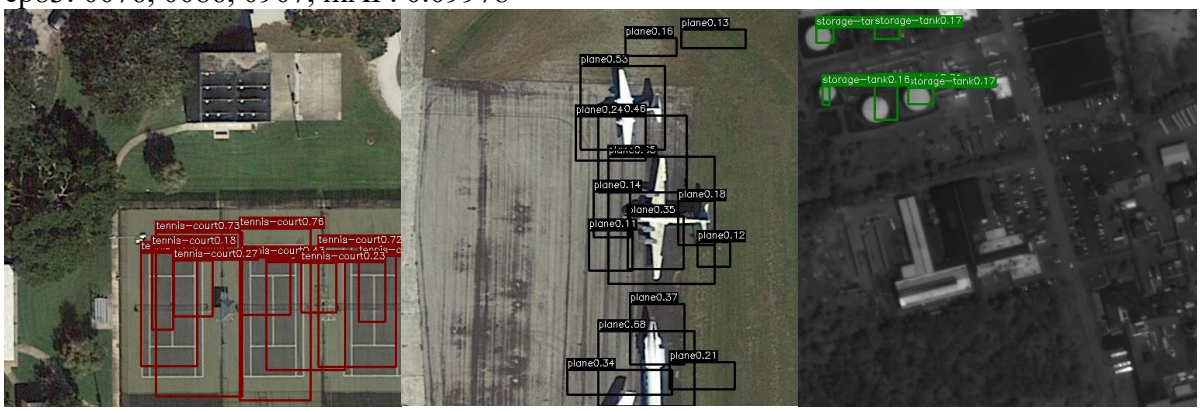
ep30: 0076, 0086, 0907, mAP: 0.063509



ep50: 0076, 0086, 0907, mAP: 0.080195



ep83: 0076, 0086, 0907, mAP: 0.09978



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

My model structure:

```
Image tensor in each batch: torch.Size([16, 3, 448, 448]) torch.float32
Label tensor in each batch: torch.Size([16, 7, 7, 26]) torch.float32
Device used: cuda
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): BatchNorm1d(4096, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace)
    (3): Dropout(p=0.5)
    (4): Linear(in_features=4096, out_features=1274, bias=True)
  )
)
```

My validation loss after each epoch (total=50+50)

```
(robotpower) ➜ map0799 git:(master) ✗ cat log.txt
start training at:2019-04-17 21:43:16.213668
ep: 0 validation avg loss:10.06872951223495
ep: 1 validation avg loss:8.729573873763389
ep: 2 validation avg loss:8.319310909890113
ep: 3 validation avg loss:7.914918709308543
ep: 4 validation avg loss:7.360897286141173
ep: 5 validation avg loss:7.329557435309633
ep: 6 validation avg loss:7.122440300089248
ep: 7 validation avg loss:6.967305245551657
ep: 8 validation avg loss:6.929756782342454
ep: 9 validation avg loss:6.817728574
ep: 10 validation avg loss:6.576482
ep: 11 validation avg loss:6.503626
ep: 12 validation avg loss:6.516508
ep: 13 validation avg loss:6.461811
ep: 14 validation avg loss:6.467506
ep: 15 validation avg loss:6.391404
ep: 16 validation avg loss:6.425808
ep: 17 validation avg loss:6.392189
ep: 18 validation avg loss:6.359246
ep: 19 validation avg loss:6.357668
ep: 20 validation avg loss:6.382333
ep: 21 validation avg loss:6.350242
ep: 22 validation avg loss:6.399354
ep: 23 validation avg loss:6.297270
ep: 24 validation avg loss:6.308721
ep: 25 validation avg loss:6.271134
ep: 26 validation avg loss:6.284024
ep: 27 validation avg loss:6.314559
ep: 28 validation avg loss:6.251921
ep: 29 validation avg loss:6.282540
ep: 30 validation avg loss:6.266604
ep: 31 validation avg loss:6.234743
ep: 32 validation avg loss:6.216395
ep: 33 validation avg loss:6.191748
ep: 34 validation avg loss:6.233467
ep: 35 validation avg loss:6.231111
ep: 36 validation avg loss:6.186664
ep: 37 validation avg loss:6.200248
ep: 38 validation avg loss:6.202762
ep: 39 validation avg loss:6.221648
ep: 40 validation avg loss:6.169998
ep: 41 validation avg loss:6.163518
ep: 42 validation avg loss:6.140497
ep: 43 validation avg loss:6.089315
ep: 44 validation avg loss:6.133689
ep: 45 validation avg loss:6.104324
ep: 46 validation avg loss:6.172813
ep: 47 validation avg loss:6.134975
ep: 48 validation avg loss:6.081174
ep: 49 validation avg loss:6.116184
end training at:2019-04-18 07:13:42.
```

```
(robotpower) ➜ hw2-shannon112 git:(master) ✗ cat improved
start training at:2019-04-18 17:34:00.188852
ep: 0 validation avg loss:6.116718309990903
ep: 1 validation avg loss:6.098700468210464
ep: 2 validation avg loss:6.106633674591146
ep: 3 validation avg loss:6.142048188980589
ep: 4 validation avg loss:6.0669383374934505
ep: 5 validation avg loss:6.077718415792952
ep: 6 validation avg loss:6.072012121373034
ep: 7 validation avg loss:6.053094916521235
ep: 8 validation avg loss:6.059084951243502
ep: 9 validation avg loss:6.139764069876772
ep: 10 validation avg loss:6.102295732878624
ep: 11 validation avg loss:6.0992554908310313
ep: 12 validation avg loss:6.061106471305198
ep: 13 validation avg loss:6.0645213685137116
ep: 14 validation avg loss:6.047614654327961
ep: 15 validation avg loss:6.046678025037684
ep: 16 validation avg loss:6.005262149775282
ep: 17 validation avg loss:6.056094892481545
ep: 18 validation avg loss:6.060883059786485
ep: 19 validation avg loss:6.033730769411046
ep: 20 validation avg loss:6.008313626685041
ep: 21 validation avg loss:6.031524663275861
ep: 22 validation avg loss:5.990406500532272
ep: 23 validation avg loss:6.015558838844299
ep: 24 validation avg loss:6.025749363163684
ep: 25 validation avg loss:6.003467188236561
ep: 26 validation avg loss:5.9935387721721165
ep: 27 validation avg loss:6.008212216356967
ep: 28 validation avg loss:6.024434804916382
ep: 29 validation avg loss:6.014916144786997
ep: 30 validation avg loss:5.999294792083984
ep: 31 validation avg loss:6.015501906263068
ep: 32 validation avg loss:5.995591247969485
ep: 33 validation avg loss:5.976795310669757
ep: 34 validation avg loss:6.033347381556288
ep: 35 validation avg loss:6.0504594616433405
ep: 36 validation avg loss:6.0285113859683905
ep: 37 validation avg loss:5.997956890375056
ep: 38 validation avg loss:6.071325248226207
ep: 39 validation avg loss:6.042464436368739
ep: 40 validation avg loss:6.0344803922988
ep: 41 validation avg loss:6.042002750203965
ep: 42 validation avg loss:6.039484994842651
ep: 43 validation avg loss:5.997727330060715
ep: 44 validation avg loss:5.997397342894939
ep: 45 validation avg loss:6.005715332766797
ep: 46 validation avg loss:6.031670395364153
ep: 47 validation avg loss:6.008545925642582
ep: 48 validation avg loss:6.004344787369383
ep: 49 validation avg loss:5.987756767171494
end training at:2019-04-19 03:00:23.877943
```

My final mAP: 0.102

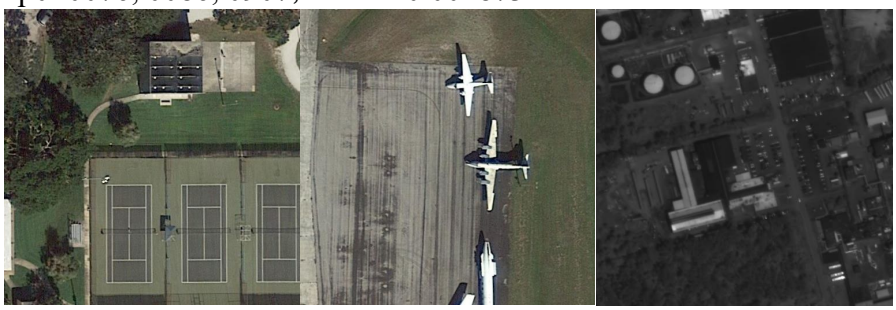
```
(robotpower) ➜ hw2-shannon112 git:(master) ✗ python h
al/val1500/labelpre ../hw2_train_val/val1500/labelTxt_
classname: plane
ap:  0.30825390909235156
classname: baseball-diamond
ap:  0.0
classname: bridge
ap:  0.09090909090909091
classname: ground-track-field
ap:  0.09090909090909091
classname: small-vehicle
ap:  0.0213903743315508
classname: large-vehicle
ap:  0.12009667224391152
classname: ship
ap:  0.09090909090909091
classname: tennis-court
ap:  0.4505861824304901
classname: basketball-court
ap:  0.09090909090909091
classname: storage-tank
ap:  0.08129096608720432
classname: soccer-ball-field
ap:  0.042148760330578516
classname: roundabout
ap:  0.0
classname: harbor
ap:  0.1530472770154374
classname: swimming-pool
ap:  0.10214504596527069
classname: helicopter
ap:  0.0
classname: container-crane
ap:  0.0
map: 0.1026622219458224
```

Detail of my training config:

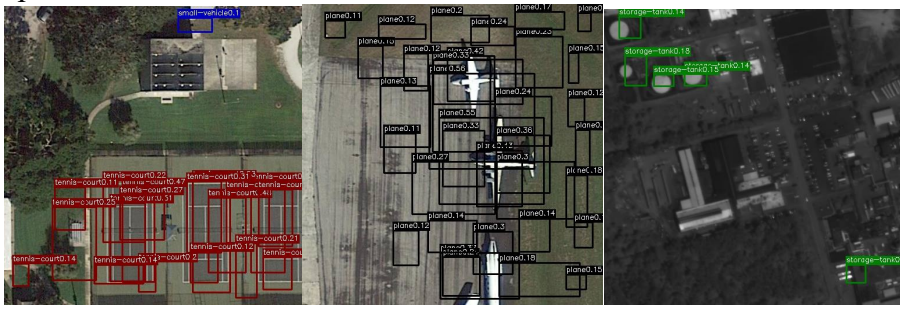
model	Improved model
net	Vgg + nn + bn + relu(T) + dropout + nn
optimizer	Torch.optim.SGD
Total epoch	50+50 (train 完 50 個後再 load 參數近來再 train50 個)
final min validation avg loss	第一次：ep48, 6.08117 第二次：ep83, 5.976795
max mAP value	0.1026622
file name	model_improved.pth
yoloLoss	5, 0.05
batchsize	16
Learning rate	第一次：ep=0: 0.002， ep=10: 0.001 第二次：ep=0: 0.001， ep=20: 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.

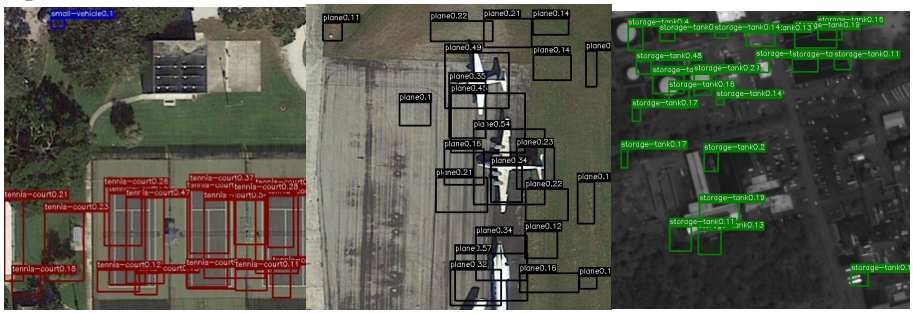
ep0: 0076, 0086, 0907， mAP: 0.002873



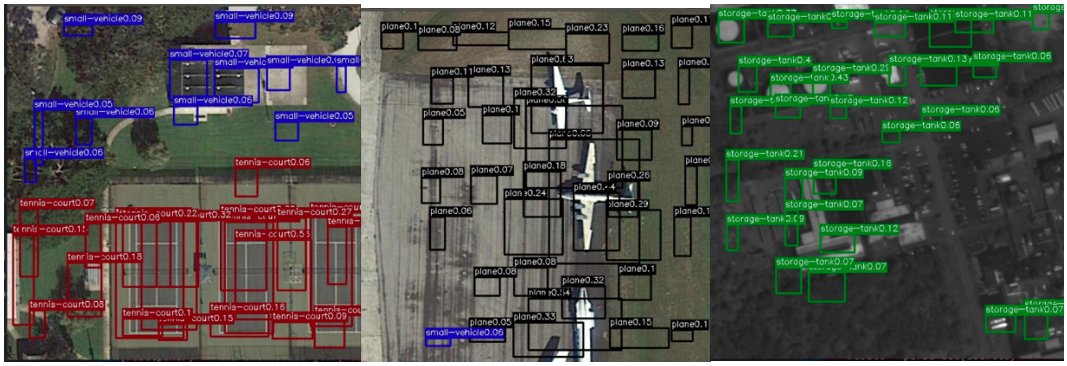
ep10: 0076, 0086, 0907， mAP: 0.0553



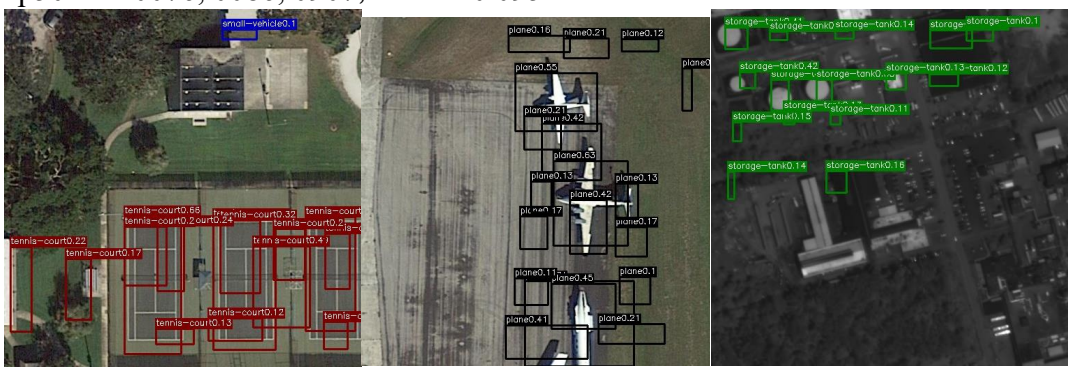
ep30: 0076, 0086, 0907， mAP:0.0729



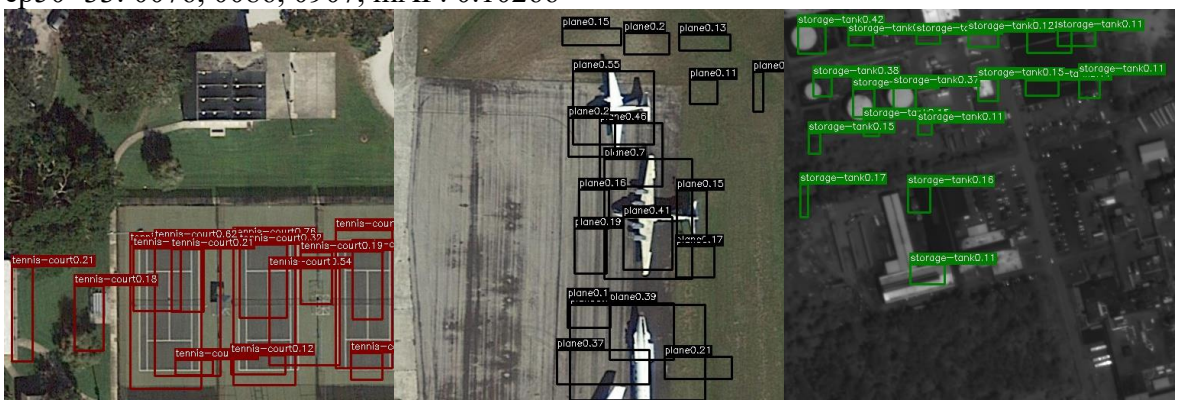
ep48: 0076, 0086, 0907， mAP: 0.0799



ep50+22: 0076, 0086, 0907, mAP: 0.0952



ep50+33: 0076, 0086, 0907, mAP: 0.10266



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.

1. All training experiment result

Model	model#1	model#2	model#3	Baseline model	Baseline model improved
Net	vgg+nn+relu(T)+drop+nn	vgg+nn+bn+relu(F)+drop+nn	vgg+nn+bn+relu(T)+drop+nn, warm up optimizer	vgg+nn+relu(T)+drop+nn	vgg+nn+bn+relu(T)+drop+nn
Total epoch	35	40	80	100	50+50
Min validation avg loss	ep35, 6.70188	ep37, 6.4778	ep56, 6.05058	ep83, 6.16308	ep48, 6.08117 ep33, 5.976795
Max	0.0569244	0.062547	0.5948679	0.099783	0.0799280,

mAP					0.1026622
file name	沒存到	best_fixclass	model3	best_baseline	best_improved
Loss	loss(5,0.5)	loss(5,0.05)	loss(5,0.5)	loss(5,0.05)	loss(5,0.05)
batchsize	16	16	32	16	16
Learning rate	0.001	0.001	ep=0: 0.002 ep=20: 0.001 ep=50: 0.0005	ep=0: 0.002 ep=10: 0.001 ep=70: 0.0005	ep=0: 0.002 ep=10: 0.001 下一批五十個 ep=0: 0.001 ep=20: 0.0005

2. Discussion

a. comparison between model#1 and model#2

model#1 是最基本款的配置，backbone 為 VGG16，然後接 Linear(512*7*7, 4096), ReLU(inplace), Dropout(0.5), 最後 Linear(4096, 7*7*26)，batchsize=16, learning rate 固定 0.001，loss 的參數也如同論文(5,0.5)。

model#2 則是在 model 裡加入一層 batchnorm1d(4096)變 nn, bn, relu, dropout, nn 的架構，loss 參數也將懲罰不含 bbox 的 confidence 的參數調低，讓 bbox 多出現一點，最後使 mAP 從 5.6%上升到了 6.2%

b. comparison between model#1 and model#3

model#1 是最基本款的配置，backbone 為 VGG16，然後接 Linear(512*7*7, 4096), ReLU(inplace), Dropout(0.5), 最後 Linear(4096, 7*7*26)，batchsize=16, learning rate 固定 0.001，loss 的參數也如同論文(5,0.5)。

model#3 則是在 model 裡加入一層 batchnorm1d(4096)變 nn, bn, relu, dropout, nn 的架構，batchsize 增大到 32，總 epoch 數也增加到 80，learning rate 一開始做 optimizer warm up，在第一個 epoch 線性從 0.0001 升到 0.002，然後再以 0.002 train 到 epoch20，下降至 0.001，epoch50 下降至 0.0005。透過這次的實作發現，optimizer warm up 和調大 batchsize 對我來說並不很有用。

c. baseline model

綜合前三種嘗試，和其他沒納入紀錄的小實驗，我使用基本的 vgg, nn, relu, dropout, nn 作為 model，loss function 調降至(5, 0.05)，batchsize 維持 16，調整總 epoch 數上升至 100，一次 training，learning rate 前 10 個 0.002，然後 0.001，第八十個後調為 0.0005

d. improved model

將 baseline 的 model 加入 batchnorm1d，以 vgg, nn, bn, relu, dropout, nn 作為 model，其他參數則和 baseline model 一致。由結果可知改善後的 model 收斂的 ep 數和原本的差不多都落在 ep83，但 validation avg loss 明顯有下降，在第 48ep 時就已經比原 model 83ep 時還，最終收斂在 5.97，loss 約比前一個 model 少了 0.1，mAP 也上升了 0.3%左右，且 ep0~83 的過程中 mAP 亦上升的比原 model 快。

3. Conclusion

加入 batchnorm1d、適時的調整 learning rate 和給予足夠的 ep 數，對 model 的 mAP 會有所改善。

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

Let's take a look at model #2 (vgg+nn+bn+relu(F)+drop+nn) , mAP:0.0625

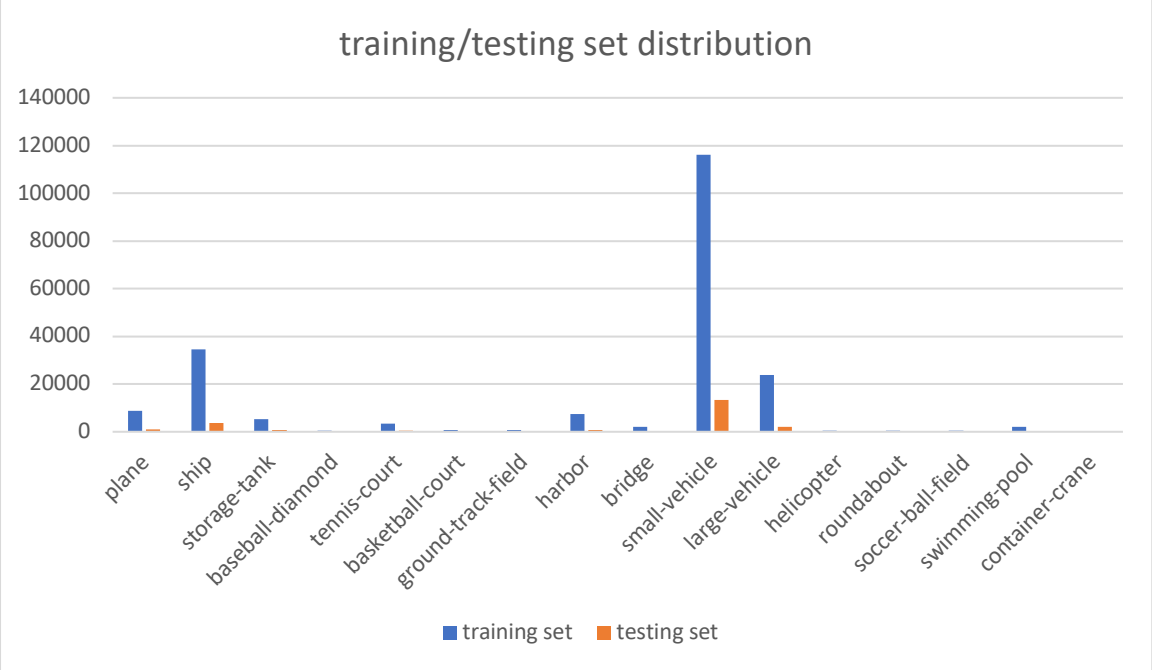
evaluation result:

```
(robotpower) ➜  hw2-shannon112 git:(ma
classname: plane
ap:  0.17959886537666025
classname: baseball-diamond
ap:  0.0
classname: bridge
ap:  4.673989249824725e-05
classname: ground-track-field
ap:  0.09090909090909091
classname: small-vehicle
ap:  0.03636363636363637
classname: large-vehicle
ap:  0.1076694600615254
classname: ship
ap:  0.07272727272727274
classname: tennis-court
ap:  0.27383112414323996
classname: basketball-court
ap:  0.0
classname: storage-tank
ap:  0.07272727272727274
classname: soccer-ball-field
ap:  0.0
classname: roundabout
ap:  0.0
classname: harbor
ap:  0.07597235238987816
classname: swimming-pool
ap:  0.09090909090909091
classname: helicopter
ap:  0.0
classname: container-crane
ap:  0.0
map: 0.06254718159376037
```

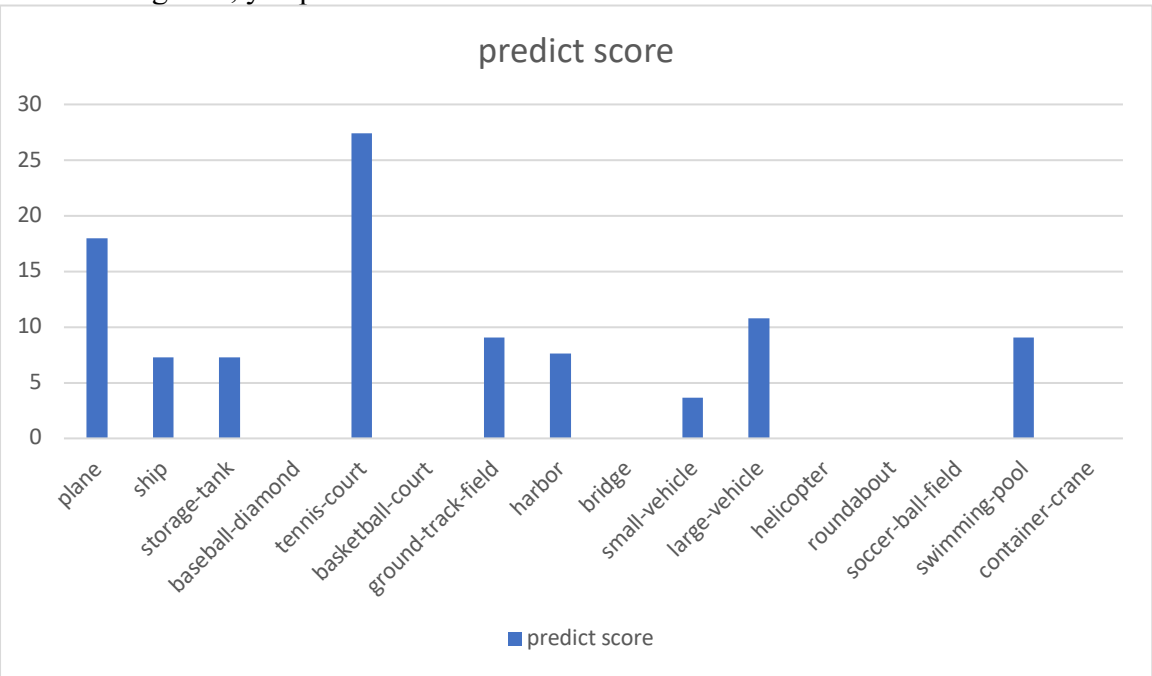
We could quickly find that tennis-court is exetremely higher than others. It causes me to check the lebals of images on training and testing dataset. Run the file counter.py and then get the result as below.

```
(robotpower) ➜  hw2-shannon112 git:(master) ✗ python counter.py
trainset [8723, 34585, 5199, 515, 3279, 661, 621, 7457, 2114, 116228, 23746, 434, 537, 590, 1977, 136]
testset [1074, 3798, 765, 75, 340, 52, 65, 825, 152, 13217, 2082, 26, 56, 68, 152, 1]
```

x is the categories, y is the number of category's bboxes



x is the categories, y is predict score



a. 由上述資料的分佈圖與實際預測分數的分佈圖可以發現，兩者分布圖並沒有一個很明顯的直接相關性，但也非完全無關，接下來幾點將解釋。

b. 資料分布圖顯示的 bbox 的種類大量分佈在 small-vehicle，然後則是 large-vehicle, harbor, ship, plane，再來是 storage-tank, tennis-court, swimming-pool，最後才是其他極少數。

c. 預測分數的分佈，在資料最多的 small-vehicle 並沒有呈現很好的成績(0<分數<5%)，但在其他資料也很多的類別 large-vehicle, harbor, ship, plane, storage-tank, tennis-court, swimming-pool 全都呈現了>5%的預測分數，因此由此可見 data 的 inbalance 是會影響預測結果的，會使 training 的結果 overfitting 那些數量很多的類別，導致準確率下降。

d. 而解法的話則可以考慮對多的資料做 sampling，或是針對較少資料的類別的照片做 Data augmentation 來生成更多的資料或混入雜訊，讓各類別的 bbox 分佈均勻。

e. 而雖然前一點發現了要數量比較多的類別才 train 的出>5%的結果，但預測分數的高低和數量則完全無關(e.g. small-vehicle 分佈數量對多，但分數非最高，甚至排不上前五)。

f. 預測分數較高者為 plane, tennis-court, large-vehicle, swimming-pool，和其他預測分數>5%的類別都屬於較大型的東西，之所以分數會比較高，是因為我 train 出來的類別狠準沒錯，但 bbox 常常匡不太好，而匡不好的情況下如果 bbox 本身

就很想，那麼一定很難和 ground truth 的 bbox 有所交集，算出來的分數自然就差，但若物件很大，bbox 即使沒匡的這麼好，但因為 ground truth 的匡也很大，所以容易有交集，在算 iou 的時候分數一定比小的物件來得高，所以最後 predict 的分數也會相對比較高。如下圖舉例：(0001, 0146, 0076)，匡 small-vehicle 這種較小的物件，與 ground truth bbox 的 iou 就會很小，即使偏移量差不多多的情況去匡網球場或港口時，另外兩者都能和原本的 ground truth 重合很多。

7.reference:

[1] yolov1 paper: https://arxiv.org/pdf/1506.02640.pdf
[2] pytorch framework(MINST demo): https://colab.research.google.com/drive/1wynOtNlGuSh7WdJ4pwcZ14lezMj9DVEy?fbclid=IwAR3qE-TuHfH4wijURr1UNmLZDBF9ox5GWFHkUfDnDq22g Q3bXDNWy_GVxes
[3] yolov1 pytorch implementation: https://github.com/xiongzihua/pytorch-YOLO-v1
[4] mns implementation: https://blog.csdn.net/hongxingabc/article/details/78996407
[5] pytorch document: https://pytorch.org/docs/stable/nn.html