You can modify this report template, and upload your results in PDF format. Reports in other forms/formats will result in ZERO point. Reports written in either Chinese or English are both acceptable. The length of your report should NOT exceed 6 pages (**excluding bonus**).

Name: 謝子然　Dep.:資工碩一　Student ID:R07922089

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

```python
class VGG(nn.Module):

    def __init__(self, features, output_size=1274, image_size=448):
        # output: 1274 is 7 x 7 x (2 x 5 + 16)
        super(VGG, self).__init__()
        self.features = features
        self.image_size = image_size

        self.yolo = nn.Sequential(
            #TODO
            nn.Linear(512 * 7 * 7, 4096),
            nn.ReLU(True),
            nn.Dropout(),
            nn.Linear(4096, output_size)
        )
        self._initialize_weights()

    def forward(self, x):
        # [batch, 3, 488, 488]
        x = self.features(x)
        # [1, 512, 7, 7]
        x = x.view(x.size(0), -1)
        # [1, 25088]
        x = self.yolo(x)
        x = torch.sigmoid(x)
        x = x.view(-1,7,7,26) # note the posiotion of "depth(26)" is different
        return x

    def _initialize_weights(self):
        for m in self.modules():
            if isinstance(m, nn.Linear):
                m.weight.data.normal_(0, 0.01)
                m.bias.data.zero_()


def make_layers(cfg, batch_norm=False):
    layers = []
    in_channels = 3
    s = 1
    first_flag=True
    for v in cfg:
        s=1
        if (v==64 and first_flag):
            # for reducing feature map size
            s=2
            first_flag=False
        if v == 'M':
            layers += [nn.MaxPool2d(kernel_size=2, stride=2)]
        else:
            conv2d = nn.Conv2d(in_channels, v, kernel_size=3, stride=s, padding=1)
            if batch_norm:
                layers += [conv2d, nn.BatchNorm2d(v), nn.ReLU(inplace=True)]
            else:
```

models.py 32,28 21%

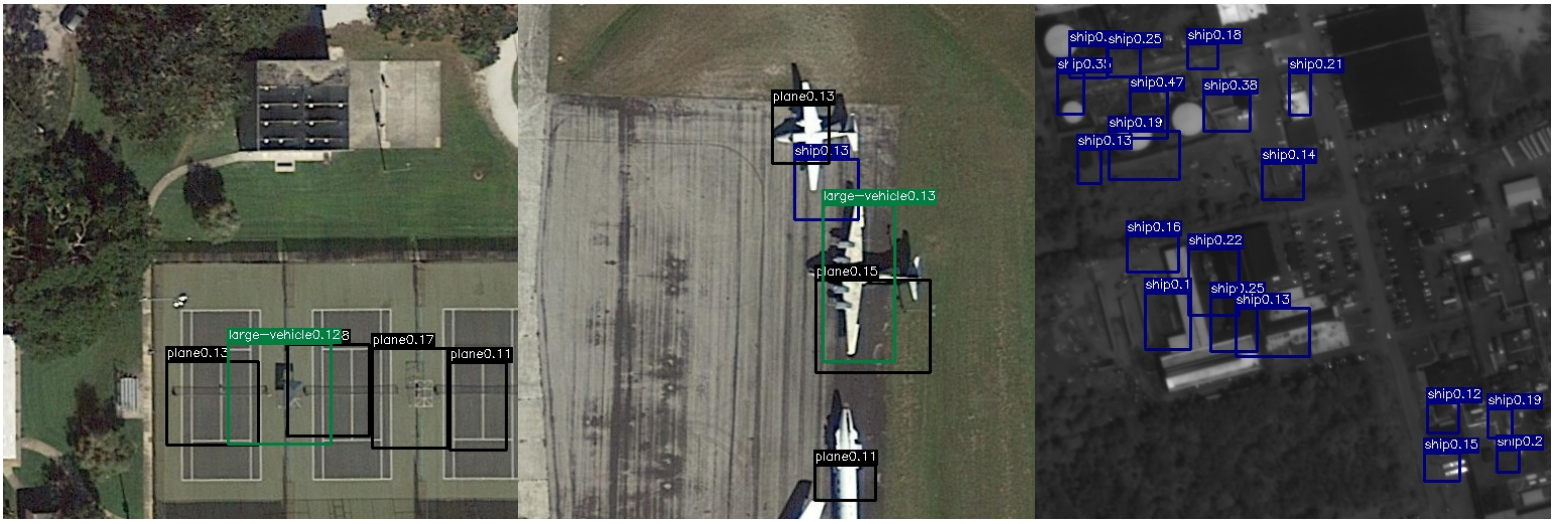
```python


def make_layers(cfg, batch_norm=False):
    layers = []
    in_channels = 3
    s = 1
    first_flag=True
    for v in cfg:
        s=1
        if (v==64 and first_flag):
            # for reducing feature map size
            s=2
            first_flag=False
        if v == 'M':
            layers += [nn.MaxPool2d(kernel_size=2, stride=2)]
        else:
            conv2d = nn.Conv2d(in_channels, v, kernel_size=3, stride=s, padding=1)
            if batch_norm:
                layers += [conv2d, nn.BatchNorm2d(v), nn.ReLU(inplace=True)]
            else:
                layers += [conv2d, nn.ReLU(inplace=True)]
            in_channels = v
    return nn.Sequential(*layers)

def conv_bn_relu(in_channels,out_channels,kernel_size=3,stride=2,padding=1):
    return nn.Sequential(
        nn.Conv2d(in_channels,out_channels,kernel_size=kernel_size,padding=padding,stride=stride),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(True)
    )


cfg = {
    'D': [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 'M', 512, 512, 512, 'M', 512, 512, 512, 'M'],
}




def Yolov1_vgg16bn(pretrained=False, **kwargs):
    """VGG 16-layer model (configuration "D") with batch normalization
    Args:
        pretrained (bool): If True, returns a model pre-trained on ImageNet
    """
    yolo = VGG(make_layers(cfg['D'], batch_norm=True), **kwargs)
    if pretrained:
        vgg_state_dict = model_zoo.load_url(model_urls['vgg16_bn'])
        yolo_state_dict = yolo.state_dict()
        for k in vgg_state_dict.keys():
            if k in yolo_state_dict.keys() and k.startswith('features'):
                yolo_state_dict[k] = vgg_state_dict[k]
    yolo.load_state_dict(yolo_state_dict)
    return yolo
```

models.py 75,5 78%

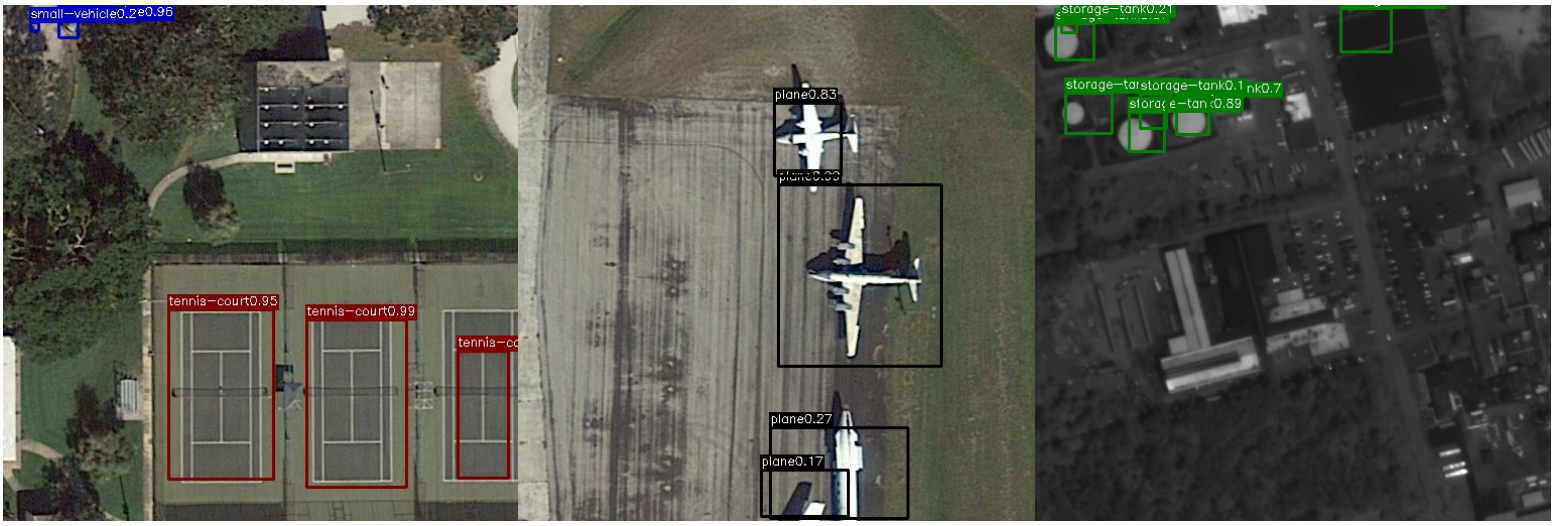
batch size = 15, learning rate fix at 10^-3, lambda_coord = 5, lambda_noobj=0.5, optimizer = SGD with momentum = 0.9, weight decay = 5e^-4, max epoch = 100

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)

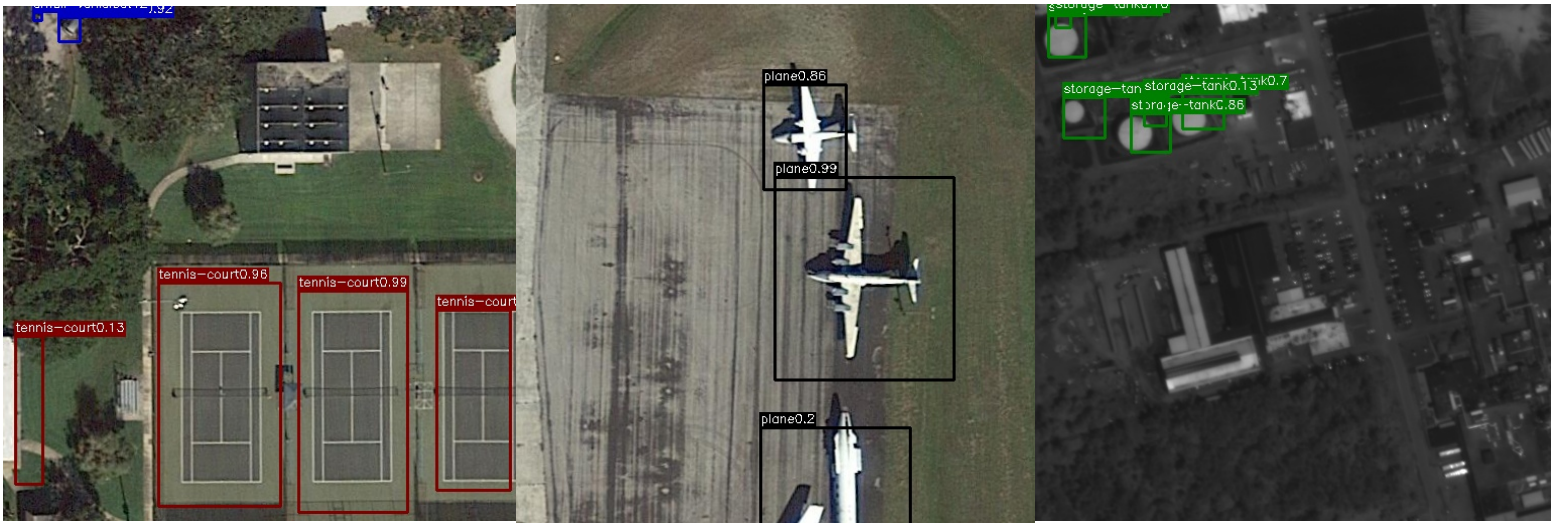
epoch 1



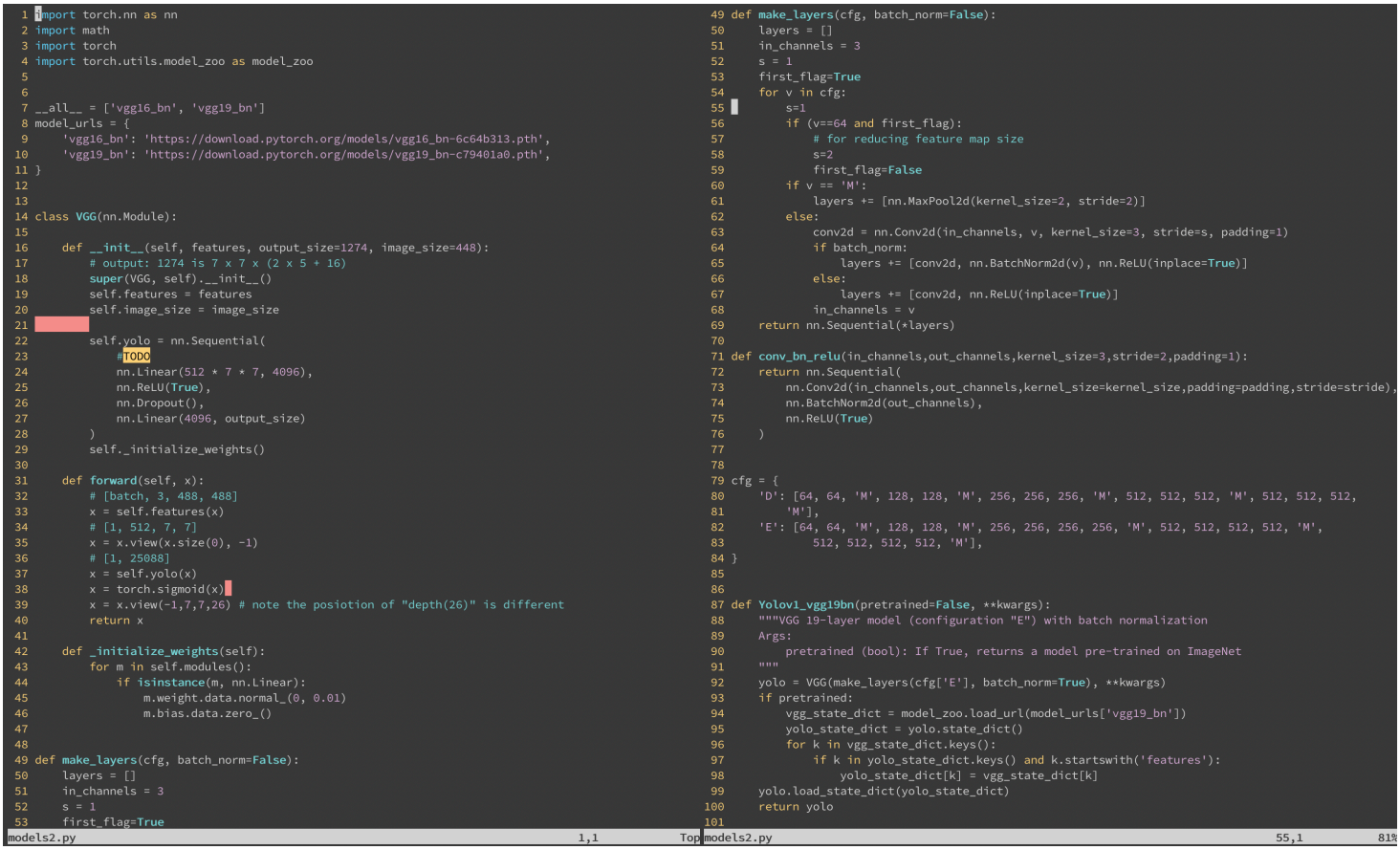
epoch 38



epoch 58



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



```
import torch.nn as nn
import math
import torch
import torch.utils.model_zoo as model_zoo


__all__ = ['vgg16_bn', 'vgg19_bn']
model_urls = {
    'vgg16_bn': 'https://download.pytorch.org/models/vgg16_bn-6c64b313.pth',
    'vgg19_bn': 'https://download.pytorch.org/models/vgg19_bn-c79401a0.pth',
}


class VGG(nn.Module):

    def __init__(self, features, output_size=1274, image_size=448):
        # output: 1274 is 7 x 7 x (2 x 5 + 16)
        super(VGG, self).__init__()
        self.features = features
        self.image_size = image_size

        self.yolo = nn.Sequential(
            #TODO
            nn.Linear(512 * 7 * 7, 4096),
            nn.ReLU(True),
            nn.Dropout(),
            nn.Linear(4096, output_size)
        )
        self._initialize_weights()

    def forward(self, x):
        # [batch, 3, 488, 488]
        x = self.features(x)
        # [1, 512, 7, 7]
        x = x.view(x.size(0), -1)
        # [1, 25088]
        x = self.yolo(x)
        x = torch.sigmoid(x)
        x = x.view(-1,7,7,26) # note the posiotion of "depth(26)" is different
        return x

    def _initialize_weights(self):
        for m in self.modules():
            if isinstance(m, nn.Linear):
                m.weight.data.normal_(0, 0.01)
                m.bias.data.zero_()


def make_layers(cfg, batch_norm=False):
    layers = []
    in_channels = 3
    s = 1
    first_flag=True
```

models2.py 1,1 Top

```
def make_layers(cfg, batch_norm=False):
    layers = []
    in_channels = 3
    s = 1
    first_flag=True
    for v in cfg:
        s=1
        if (v==64 and first_flag):
            # for reducing feature map size
            s=2
            first_flag=False
        if v == 'M':
            layers += [nn.MaxPool2d(kernel_size=2, stride=2)]
        else:
            conv2d = nn.Conv2d(in_channels, v, kernel_size=3, stride=s, padding=1)
            if batch_norm:
                layers += [conv2d, nn.BatchNorm2d(v), nn.ReLU(inplace=True)]
            else:
                layers += [conv2d, nn.ReLU(inplace=True)]
            in_channels = v
    return nn.Sequential(*layers)

def conv_bn_relu(in_channels,out_channels,kernel_size=3,stride=2,padding=1):
    return nn.Sequential(
        nn.Conv2d(in_channels,out_channels,kernel_size=kernel_size,padding=padding,stride=stride),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(True)
    )


cfg = {
    'D': [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 'M', 512, 512, 512, 'M', 512, 512, 512,
        'M'],
    'E': [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, 512, 512, 'M',
            512, 512, 512, 512, 'M'],
}


def Yolov1_vgg19bn(pretrained=False, **kwargs):
    """VGG 19-layer model (configuration "E") with batch normalization
    Args:
        pretrained (bool): If True, returns a model pre-trained on ImageNet
    """
    yolo = VGG(make_layers(cfg['E'], batch_norm=True), **kwargs)
    if pretrained:
        vgg_state_dict = model_zoo.load_url(model_urls['vgg19_bn'])
        yolo_state_dict = yolo.state_dict()
        for k in vgg_state_dict.keys():
            if k in yolo_state_dict.keys() and k.startswith('features'):
                yolo_state_dict[k] = vgg_state_dict[k]
    yolo.load_state_dict(yolo_state_dict)
    return yolo

```

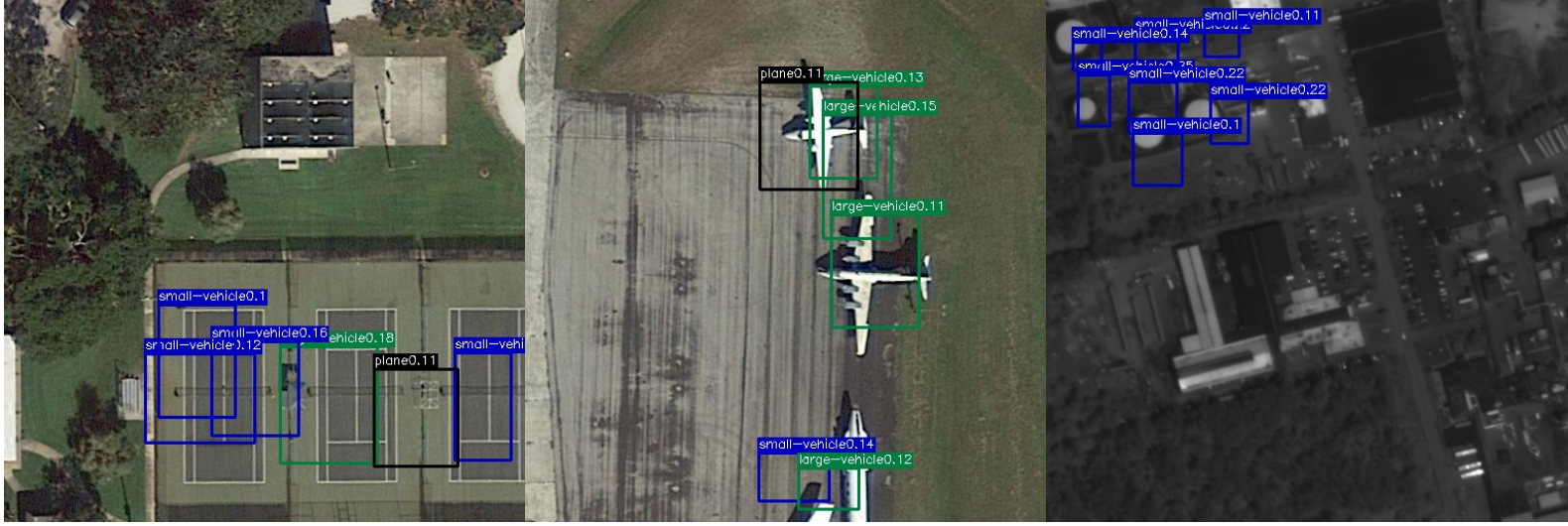
models2.py 55,1 81%

I changed the backbone network from VGG16 to VGG19.Besides, and I also change make the learning rate scale down by multiplying 0.8 every 30 epochs.

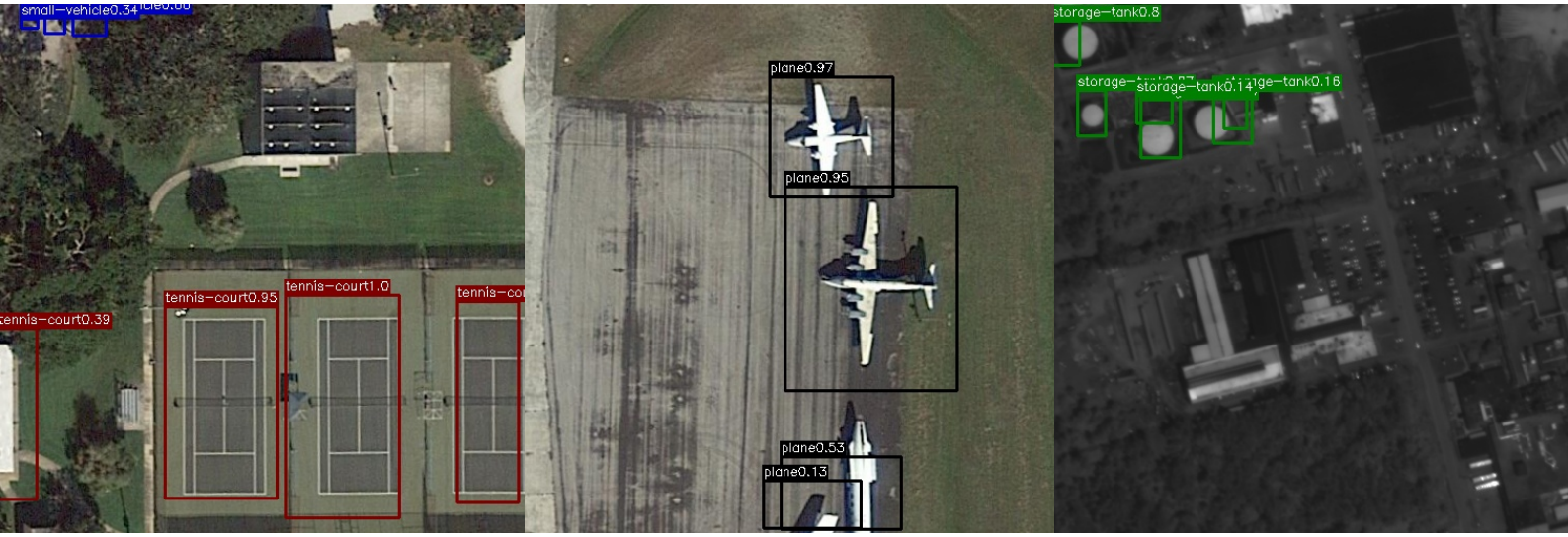
batch size = 15, lambda_coord = 5, lambda_noobj=0.5, optimizer = SGD with momentum = 0.9, weight decay = 5e^-4, max epoch = 100

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.

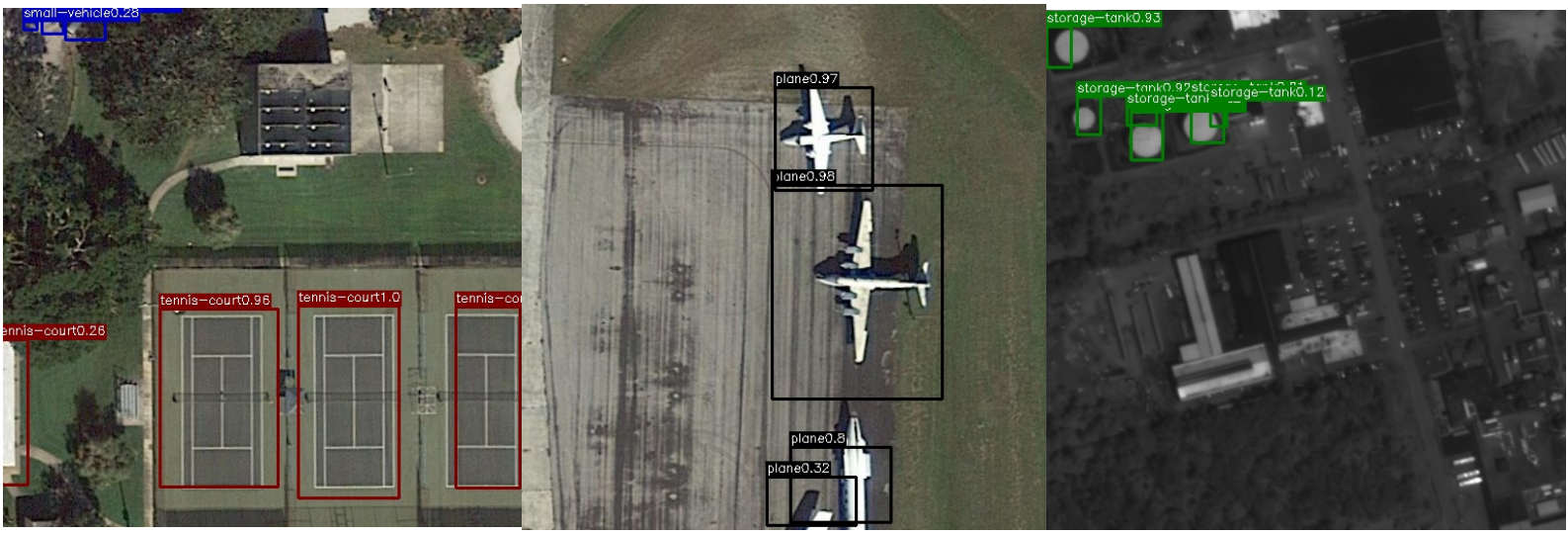
epoch 1



epoch 40



epoch 72



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.

mAP of baseline model: 0.13062

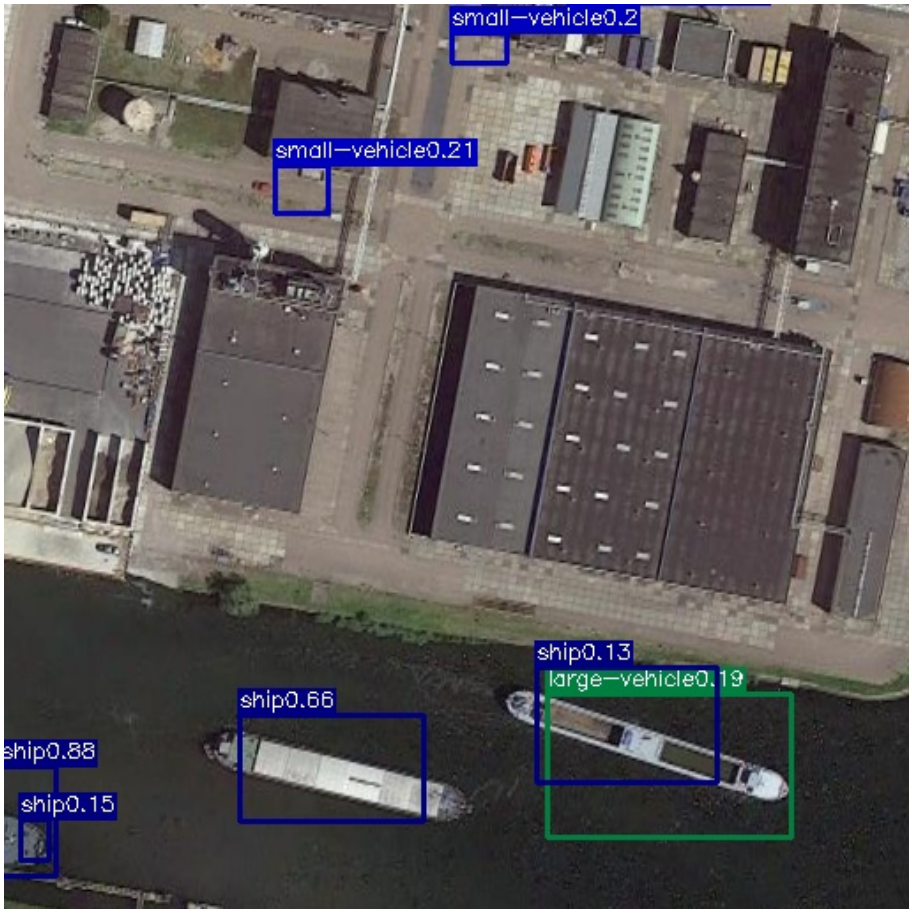
mAP of best model: 0.13217

The baseline model converged at epoch 58, and the best model converged at epoch 72, I think unfixed rate may help the network to find out the true local minimum. Besides, comparing to VGG16, VGG19 may have more power to extract features from input image, as we can see in the comparison of 0086.jpg, my best model do better when the plane is incomplete in image. I conclude that it learned more plane features therefore it has more confidence when it sees incomplete airplanes.

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

classname: plane	classname: small-vehicle	classname: basketball-court	classname: harbor
ap: 0.36414999560461647	ap: 0.005505289865479439	ap: 0.14545454545454545	ap: 0.10730313165095774
classname: baseball-diamond	classname: large-vehicle	classname: storage-tank	classname: swimming-pool
ap: 0.15844155844155847	ap: 0.12410147991543341	ap: 0.03827751196172249	ap: 0.07692307692307693
classname: bridge	classname: ship	classname: soccer-ball-field	classname: helicopter
ap: 0.15729099505695252	ap: 0.03745595344330556	ap: 0.11363636363636365	ap: 0.0
classname: ground-track-field	classname: tennis-court	classname: roundabout	classname: container-crane
ap: 0.11188811188811189	ap: 0.5651810671753822	ap: 0.1090909090909091	ap: 0.0

Ship sometime may be predicted as large-vehicle, as they actually looks pretty likely in aerial photos. Besides, Storage tank may easily infected by shadow. The following image shows a mis-predicted ship and a low-confidence storage tank.



7. Reference: Loss module and NMS are referenced and modified from: https://github.com/bobo0810/AnnotatedNetworkModelGit/tree/master/Yolov1_pytorch