MIS 583 Assignment 5: YOLO Object Detection on PASCAL VOC

Before we start, please put your name and SID in following format:
: LASTNAME Firstname, ?00000000 // e.g.) 李晨愷 M114020035

Your Answer:
Hi I'm XXX, XXXXXXXXXX.

Google Colab Setup

Next we need to run a few commands to set up our environment on Google Colab. If you are running this notebook on a local machine you can skip this section.

Run the following cell to mount your Google Drive. Follow the link, sign in to your Google account (the same account you used to store this notebook!) and copy the authorization code into the text box that appears below.

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

How to Get Data

請先到共用雲端硬碟將檔案 `VOCdevkit_2007.zip` ，建立捷徑到自己的雲端硬碟中。

> 操作步驟
>
> 1. 點開雲端連結
> 2. 點選右上角「新增雲端硬碟捷徑」
> 3. 點選「我的雲端硬碟」
> 4. 點選「新增捷徑」

完成以上流程會在你的雲端硬碟中建立一個檔案的捷徑，接著我們在colab中取得權限即可使用。

Unzip Data

解壓縮 `VOCdevkit_2007.zip`

- `VOC2007` ：包含了train/val的所有圖片
- `VOC2007test` ：包含了test的所有圖片

其中 `train` 的圖片 3756 張， `val` 的圖片 1255 張， `test` 的圖片 4950 張。

注意: 若有另外設定存放在雲端硬碟中的路徑，請記得本處路徑也須做更動。

Notice: Please put "VOCdevkit_2007" folder under data folder.

In [ ]:
```
!unzip -qq ./drive/MyDrive/VOCdevkit_2007.zip
```

Import package

In [1]:
```
import os
import random

import cv2
import numpy as np

import csv

import torch
from torch.utils.data import DataLoader
from torchvision import models

from src.resnet_yolo import resnet50
from yolo_loss import YoloLoss
from src.dataset import VocDetectorDataset
from src.eval_voc import evaluate, test_evaluate
from src.predict import predict_image
from src.config import VOC_CLASSES, COLORS
from kaggle_submission import write_csv

import matplotlib.pyplot as plt
import collections

%matplotlib inline
%load_ext autoreload
%autoreload 2
```

Initialization

In [2]:
```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

In [3]:
```
# YOLO network hyperparameters
B = 2  # number of bounding box predictions per cell
S = 14  # width/height of network output grid (larger than 7x7 from paper
```

To implement Yolo we will rely on a pretrained classifier as the backbone for our detection network. PyTorch offers a variety of models which are pretrained on ImageNet in the `torchvision.models` package. In particular, we will use the ResNet50 architecture as a base for our detector. This is different from the base architecture in the Yolo paper and also results in a different output grid size (14x14 instead of 7x7).

Models are typically pretrained on ImageNet since the dataset is very large (> 1 million images) and widely used. The pretrained model provides a very useful weight initialization for our detector, so that the network is able to learn quickly and effectively.

In [4]:
```
load_network_path = None #'checkpoints/best_detector.pth'
pretrained = True

# use to load a previously trained network
if load_network_path is not None:
    print('Loading saved network from {}'.format(load_network_path))
    net = resnet50().to(device)
    net.load_state_dict(torch.load(load_network_path))
else:
    print('Load pre-trained model')
    net = resnet50(pretrained=pretrained).to(device)
```

```
Load pre-trained model
```

```
/home/vllab/anaconda3/lib/python3.11/site-packages/torchvision/models/_ut
ils.py:208: UserWarning: The parameter 'pretrained' is deprecated since
0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
/home/vllab/anaconda3/lib/python3.11/site-packages/torchvision/models/_ut
ils.py:223: UserWarning: Arguments other than a weight enum or `None` for
'weights' are deprecated since 0.13 and may be removed in the future. The
current behavior is equivalent to passing `weights=ResNet50_Weights.IMAGE
NET1K_V1`. You can also use `weights=ResNet50_Weights.DEFAULT` to get the
most up-to-date weights.
  warnings.warn(msg)
```

In [5]:
```
learning_rate = 0.001
num_epochs = 50 #50
batch_size = 10 #24

# Yolo loss component coefficients (as given in Yolo v1 paper)
lambda_coord = 5
lambda_noobj = 0.5
```

Reading Pascal Data

The train dataset loader also using a variety of data augmentation techniques including random shift, scaling, crop, and flips. Data augmentation is slightly more complicated for detection datasets since the bounding box annotations must be kept consistent throughout the transformations.

Since the output of the detector network we train is an SxSx(B*5+C), we use an encoder to convert the original bounding box coordinates into relative grid bounding box coordinates corresponding to the expected output. We also use a decoder which allows us to convert the opposite direction into image coordinate bounding boxes.

Notice: Please put "VOCdevkit_2007" folder under data folder.

In [6]:
```
file_root_train = 'data/VOCdevkit_2007/VOC2007/JPEGImages/'
annotation_file_train = 'data/voc2007train.txt'

train_dataset = VocDetectorDataset(root_img_dir=file_root_train,dataset_f
train_loader = DataLoader(train_dataset,batch_size=batch_size,shuffle=Tru
print('Loaded %d train images' % len(train_dataset))
```

```
Initializing dataset
Loaded 3756 train images
```

In [7]:
```
file_root_val = 'data/VOCdevkit_2007/VOC2007/JPEGImages/'
annotation_file_val = 'data/voc2007val.txt'

val_dataset = VocDetectorDataset(root_img_dir=file_root_val,dataset_file=
val_loader = DataLoader(val_dataset,batch_size=batch_size,shuffle=False,n
print('Loaded %d val images' % len(val_dataset))
```

```
Initializing dataset
Loaded 1255 val images
```

In [8]:
```
data = train_dataset[0]
```

Set up training tools

In [9]:
```
criterion = YoloLoss(S, B, lambda_coord, lambda_noobj)
optimizer = torch.optim.SGD(net.parameters(), lr=learning_rate, momentum=
```

Train detector

In [10]:
```
best_val_loss = np.inf
learning_rate = 1e-3
for epoch in range(num_epochs):
    torch.cuda.empty_cache()
    net.train()

    # Update learning rate late in training
    if epoch == 30 or epoch == 40:
        learning_rate /= 10.0

    for param_group in optimizer.param_groups:
        param_group['lr'] = learning_rate

    print('\n\nStarting epoch %d / %d' % (epoch + 1, num_epochs))
    print('Learning Rate for this epoch: {}'.format(learning_rate))

    total_loss = collections.defaultdict(int)

    for i, data in enumerate(train_loader):
        data = (item.to(device) for item in data)
        images, target_boxes, target_cls, has_object_map = data
        pred = net(images)
        loss_dict = criterion(pred, target_boxes, target_cls, has_object_
        for key in loss_dict:
            total_loss[key] += loss_dict[key].item()

        optimizer.zero_grad()
        loss_dict['total_loss'].backward()
        optimizer.step()

        if (i+1) % 50 == 0:
            outstring = 'Epoch [%d/%d], Iter [%d/%d], Loss: ' % ((epoch+1
            outstring += ', '.join( "%s=%.3f" % (key[:-5], val / (i+1)) f
            print(outstring)

    # evaluate the network on the val data
    if (epoch + 1) % 5 == 0:
        val_aps = evaluate(net, val_dataset_file=annotation_file_val, img
        print(epoch, val_aps)
    with torch.no_grad():
        val_loss = 0.0
        net.eval()
        for i, data in enumerate(val_loader):
            data = (item.to(device) for item in data)
            images, target_boxes, target_cls, has_object_map = data

            pred = net(images)
            loss_dict = criterion(pred, target_boxes, target_cls, has_obj
            val_loss += loss_dict['total_loss'].item()
        val_loss /= len(val_loader)

    if best_val_loss > val_loss:
        best_val_loss = val_loss
        print('Updating best val loss: %.5f' % best_val_loss)
        torch.save(net.state_dict(),'checkpoints/best_detector.pth')

    if (epoch+1) in [5, 10, 20, 30, 40]:
        torch.save(net.state_dict(),'checkpoints/detector_epoch_%d.pth' %

    torch.save(net.state_dict(),'checkpoints/detector.pth')
```

```
Starting epoch 1 / 50
Learning Rate for this epoch: 0.001
Epoch [1/50], Iter [50/376], Loss: total=25.513, reg=4.523, containing_ob
j=0.325, no_obj=12.562, cls=8.102
Epoch [1/50], Iter [100/376], Loss: total=17.515, reg=4.121, containing_o
bj=0.410, no_obj=6.510, cls=6.474
Epoch [1/50], Iter [150/376], Loss: total=14.232, reg=3.711, containing_o
bj=0.471, no_obj=4.446, cls=5.605
Epoch [1/50], Iter [200/376], Loss: total=12.407, reg=3.465, containing_o
bj=0.498, no_obj=3.397, cls=5.048
Epoch [1/50], Iter [250/376], Loss: total=11.233, reg=3.296, containing_o
bj=0.532, no_obj=2.759, cls=4.646
Epoch [1/50], Iter [300/376], Loss: total=10.404, reg=3.166, containing_o
bj=0.558, no_obj=2.329, cls=4.351
Epoch [1/50], Iter [350/376], Loss: total=9.768, reg=3.063, containing_ob
j=0.581, no_obj=2.019, cls=4.105
Updating best val loss: 5.75227


Starting epoch 2 / 50
Learning Rate for this epoch: 0.001
Epoch [2/50], Iter [50/376], Loss: total=5.409, reg=2.228, containing_obj
=0.739, no_obj=0.134, cls=2.308
Epoch [2/50], Iter [100/376], Loss: total=5.431, reg=2.226, containing_ob
j=0.751, no_obj=0.128, cls=2.325
Epoch [2/50], Iter [150/376], Loss: total=5.440, reg=2.225, containing_ob
j=0.762, no_obj=0.123, cls=2.329
Epoch [2/50], Iter [200/376], Loss: total=5.354, reg=2.207, containing_ob
j=0.770, no_obj=0.119, cls=2.258
Epoch [2/50], Iter [250/376], Loss: total=5.301, reg=2.206, containing_ob
j=0.776, no_obj=0.115, cls=2.204
Epoch [2/50], Iter [300/376], Loss: total=5.243, reg=2.196, containing_ob
j=0.772, no_obj=0.111, cls=2.164
Epoch [2/50], Iter [350/376], Loss: total=5.240, reg=2.203, containing_ob
j=0.777, no_obj=0.108, cls=2.153
Updating best val loss: 5.20640


Starting epoch 3 / 50
Learning Rate for this epoch: 0.001
Epoch [3/50], Iter [50/376], Loss: total=4.688, reg=2.073, containing_obj
=0.792, no_obj=0.082, cls=1.741
Epoch [3/50], Iter [100/376], Loss: total=4.797, reg=2.102, containing_ob
j=0.807, no_obj=0.080, cls=1.808
Epoch [3/50], Iter [150/376], Loss: total=4.774, reg=2.064, containing_ob
j=0.811, no_obj=0.078, cls=1.820
Epoch [3/50], Iter [200/376], Loss: total=4.808, reg=2.090, containing_ob
j=0.816, no_obj=0.077, cls=1.825
Epoch [3/50], Iter [250/376], Loss: total=4.780, reg=2.077, containing_ob
j=0.822, no_obj=0.076, cls=1.805
Epoch [3/50], Iter [300/376], Loss: total=4.778, reg=2.085, containing_ob
j=0.829, no_obj=0.074, cls=1.790
Epoch [3/50], Iter [350/376], Loss: total=4.768, reg=2.080, containing_ob
j=0.830, no_obj=0.073, cls=1.784
Updating best val loss: 4.87400


Starting epoch 4 / 50
Learning Rate for this epoch: 0.001
Epoch [4/50], Iter [50/376], Loss: total=4.495, reg=1.937, containing_obj
=0.821, no_obj=0.065, cls=1.673
Epoch [4/50], Iter [100/376], Loss: total=4.437, reg=1.950, containing_ob
```

```
j=0.819, no_obj=0.064, cls=1.603
Epoch [4/50], Iter [150/376], Loss: total=4.418, reg=1.944, containing_ob
j=0.830, no_obj=0.064, cls=1.580
Epoch [4/50], Iter [200/376], Loss: total=4.497, reg=1.991, containing_ob
j=0.842, no_obj=0.064, cls=1.601
Epoch [4/50], Iter [250/376], Loss: total=4.436, reg=1.964, containing_ob
j=0.842, no_obj=0.064, cls=1.566
Epoch [4/50], Iter [300/376], Loss: total=4.426, reg=1.960, containing_ob
j=0.836, no_obj=0.064, cls=1.567
Epoch [4/50], Iter [350/376], Loss: total=4.435, reg=1.969, containing_ob
j=0.846, no_obj=0.064, cls=1.557
Updating best val loss: 4.67710


Starting epoch 5 / 50
Learning Rate for this epoch: 0.001
Epoch [5/50], Iter [50/376], Loss: total=4.580, reg=2.039, containing_obj
=0.905, no_obj=0.066, cls=1.570
Epoch [5/50], Iter [100/376], Loss: total=4.488, reg=1.972, containing_ob
j=0.884, no_obj=0.067, cls=1.564
Epoch [5/50], Iter [150/376], Loss: total=4.411, reg=1.940, containing_ob
j=0.877, no_obj=0.068, cls=1.526
Epoch [5/50], Iter [200/376], Loss: total=4.375, reg=1.932, containing_ob
j=0.869, no_obj=0.069, cls=1.505
Epoch [5/50], Iter [250/376], Loss: total=4.342, reg=1.923, containing_ob
j=0.864, no_obj=0.070, cls=1.485
Epoch [5/50], Iter [300/376], Loss: total=4.328, reg=1.917, containing_ob
j=0.867, no_obj=0.071, cls=1.473
Epoch [5/50], Iter [350/376], Loss: total=4.300, reg=1.903, containing_ob
j=0.860, no_obj=0.072, cls=1.465
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:27<00:00, 46.27it/s]
```

```
---class aeroplane ap 0.025---
---class bicycle ap 0.0---
---class bird ap 0.0--- (no predictions for this class)
---class boat ap 0.0--- (no predictions for this class)
---class bottle ap 0.0--- (no predictions for this class)
---class bus ap 0.0--- (no predictions for this class)
---class car ap 0.1964678090677125---
---class cat ap 0.0--- (no predictions for this class)
---class chair ap 0.005---
---class cow ap 0.0--- (no predictions for this class)
---class diningtable ap 0.0--- (no predictions for this class)
---class dog ap 0.0--- (no predictions for this class)
---class horse ap 0.0681063122923588---
---class motorbike ap 0.0--- (no predictions for this class)
---class person ap 0.0835911979597899---
---class pottedplant ap 0.0--- (no predictions for this class)
---class sheep ap 0.0---
---class sofa ap 0.0--- (no predictions for this class)
---class train ap 0.0--- (no predictions for this class)
---class tvmonitor ap 0.0---
---map 0.018908265965993064---
4 [0.025, 0.0, 0.0, 0.0, 0.0, 0.0, 0.1964678090677125, 0.0, 0.005, 0.0,
0.0, 0.0, 0.0681063122923588, 0.0, 0.0835911979597899, 0.0, 0.0, 0.0, 0.
0, 0.0]
Updating best val loss: 4.48595


Starting epoch 6 / 50
Learning Rate for this epoch: 0.001
Epoch [6/50], Iter [50/376], Loss: total=3.969, reg=1.729, containing_obj
=0.831, no_obj=0.083, cls=1.325
Epoch [6/50], Iter [100/376], Loss: total=4.160, reg=1.836, containing_ob
j=0.851, no_obj=0.085, cls=1.388
Epoch [6/50], Iter [150/376], Loss: total=4.129, reg=1.828, containing_ob
j=0.846, no_obj=0.086, cls=1.369
Epoch [6/50], Iter [200/376], Loss: total=4.145, reg=1.849, containing_ob
j=0.848, no_obj=0.088, cls=1.360
Epoch [6/50], Iter [250/376], Loss: total=4.131, reg=1.830, containing_ob
j=0.846, no_obj=0.089, cls=1.366
Epoch [6/50], Iter [300/376], Loss: total=4.110, reg=1.817, containing_ob
j=0.844, no_obj=0.090, cls=1.359
Epoch [6/50], Iter [350/376], Loss: total=4.098, reg=1.818, containing_ob
j=0.841, no_obj=0.090, cls=1.348
Updating best val loss: 4.29943


Starting epoch 7 / 50
Learning Rate for this epoch: 0.001
Epoch [7/50], Iter [50/376], Loss: total=4.010, reg=1.779, containing_obj
=0.841, no_obj=0.092, cls=1.298
Epoch [7/50], Iter [100/376], Loss: total=3.916, reg=1.727, containing_ob
j=0.825, no_obj=0.093, cls=1.271
Epoch [7/50], Iter [150/376], Loss: total=3.935, reg=1.743, containing_ob
j=0.821, no_obj=0.094, cls=1.277
Epoch [7/50], Iter [200/376], Loss: total=3.884, reg=1.726, containing_ob
j=0.809, no_obj=0.096, cls=1.254
Epoch [7/50], Iter [250/376], Loss: total=3.947, reg=1.762, containing_ob
j=0.831, no_obj=0.096, cls=1.258
Epoch [7/50], Iter [300/376], Loss: total=3.910, reg=1.742, containing_ob
j=0.826, no_obj=0.096, cls=1.246
Epoch [7/50], Iter [350/376], Loss: total=3.927, reg=1.755, containing_ob
j=0.832, no_obj=0.097, cls=1.242
Updating best val loss: 4.18828
```

```
Starting epoch 8 / 50
Learning Rate for this epoch: 0.001
Epoch [8/50], Iter [50/376], Loss: total=3.873, reg=1.712, containing_obj
=0.868, no_obj=0.098, cls=1.195
Epoch [8/50], Iter [100/376], Loss: total=3.920, reg=1.756, containing_ob
j=0.869, no_obj=0.099, cls=1.195
Epoch [8/50], Iter [150/376], Loss: total=3.947, reg=1.782, containing_ob
j=0.877, no_obj=0.099, cls=1.189
Epoch [8/50], Iter [200/376], Loss: total=3.907, reg=1.755, containing_ob
j=0.862, no_obj=0.102, cls=1.189
Epoch [8/50], Iter [250/376], Loss: total=3.865, reg=1.731, containing_ob
j=0.852, no_obj=0.102, cls=1.180
Epoch [8/50], Iter [300/376], Loss: total=3.812, reg=1.711, containing_ob
j=0.837, no_obj=0.103, cls=1.161
Epoch [8/50], Iter [350/376], Loss: total=3.778, reg=1.697, containing_ob
j=0.828, no_obj=0.103, cls=1.149
Updating best val loss: 3.97744


Starting epoch 9 / 50
Learning Rate for this epoch: 0.001
Epoch [9/50], Iter [50/376], Loss: total=3.627, reg=1.623, containing_obj
=0.848, no_obj=0.112, cls=1.043
Epoch [9/50], Iter [100/376], Loss: total=3.679, reg=1.634, containing_ob
j=0.831, no_obj=0.111, cls=1.103
Epoch [9/50], Iter [150/376], Loss: total=3.670, reg=1.641, containing_ob
j=0.831, no_obj=0.108, cls=1.090
Epoch [9/50], Iter [200/376], Loss: total=3.622, reg=1.628, containing_ob
j=0.817, no_obj=0.110, cls=1.066
Epoch [9/50], Iter [250/376], Loss: total=3.552, reg=1.592, containing_ob
j=0.803, no_obj=0.111, cls=1.046
Epoch [9/50], Iter [300/376], Loss: total=3.544, reg=1.591, containing_ob
j=0.801, no_obj=0.112, cls=1.041
Epoch [9/50], Iter [350/376], Loss: total=3.568, reg=1.609, containing_ob
j=0.807, no_obj=0.111, cls=1.041
Updating best val loss: 3.89062


Starting epoch 10 / 50
Learning Rate for this epoch: 0.001
Epoch [10/50], Iter [50/376], Loss: total=3.527, reg=1.592, containing_ob
j=0.808, no_obj=0.112, cls=1.015
Epoch [10/50], Iter [100/376], Loss: total=3.509, reg=1.592, containing_o
bj=0.826, no_obj=0.113, cls=0.978
Epoch [10/50], Iter [150/376], Loss: total=3.489, reg=1.586, containing_o
bj=0.813, no_obj=0.113, cls=0.977
Epoch [10/50], Iter [200/376], Loss: total=3.509, reg=1.590, containing_o
bj=0.815, no_obj=0.111, cls=0.993
Epoch [10/50], Iter [250/376], Loss: total=3.465, reg=1.571, containing_o
bj=0.812, no_obj=0.111, cls=0.971
Epoch [10/50], Iter [300/376], Loss: total=3.482, reg=1.579, containing_o
bj=0.811, no_obj=0.112, cls=0.980
Epoch [10/50], Iter [350/376], Loss: total=3.475, reg=1.579, containing_o
bj=0.810, no_obj=0.113, cls=0.974
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:31<00:00, 39.58it/s]
```

```
---class aeroplane ap 0.3643968402920609---
---class bicycle ap 0.1922373583939121---
---class bird ap 0.1549462249066992---
---class boat ap 0.06230938028927922---
---class bottle ap 0.0--- (no predictions for this class)
---class bus ap 0.045454545454545456---
---class car ap 0.20934365069154598---
---class cat ap 0.16304916193681487---
---class chair ap 0.14782833580839763---
---class cow ap 0.002898550724637681---
---class diningtable ap 0.015873015873015872---
---class dog ap 0.1223697140146381---
---class horse ap 0.3135874682544261---
---class motorbike ap 0.1006728778467909---
---class person ap 0.22540235175931725---
---class pottedplant ap 0.018017344497607654---
---class sheep ap 0.04662058371735791---
---class sofa ap 0.0--- (no predictions for this class)
---class train ap 0.3447655374268278---
---class tvmonitor ap 0.21836505100352627---
---map 0.13740689964457004---
9 [0.3643968402920609, 0.1922373583939121, 0.1549462249066992, 0.06230938
028927922, 0.0, 0.045454545454545456, 0.20934365069154598, 0.163049161936
81487, 0.14782833580839763, 0.002898550724637681, 0.015873015873015872,
0.1223697140146381, 0.3135874682544261, 0.1006728778467909, 0.22540235175
931725, 0.018017344497607654, 0.04662058371735791, 0.0, 0.344765537426827
8, 0.21836505100352627]
Updating best val loss: 3.73310


Starting epoch 11 / 50
Learning Rate for this epoch: 0.001
Epoch [11/50], Iter [50/376], Loss: total=3.599, reg=1.707, containing_ob
j=0.805, no_obj=0.121, cls=0.966
Epoch [11/50], Iter [100/376], Loss: total=3.495, reg=1.637, containing_o
bj=0.788, no_obj=0.127, cls=0.943
Epoch [11/50], Iter [150/376], Loss: total=3.510, reg=1.636, containing_o
bj=0.798, no_obj=0.127, cls=0.950
Epoch [11/50], Iter [200/376], Loss: total=3.485, reg=1.634, containing_o
bj=0.799, no_obj=0.125, cls=0.927
Epoch [11/50], Iter [250/376], Loss: total=3.441, reg=1.607, containing_o
bj=0.798, no_obj=0.125, cls=0.910
Epoch [11/50], Iter [300/376], Loss: total=3.406, reg=1.582, containing_o
bj=0.796, no_obj=0.125, cls=0.902
Epoch [11/50], Iter [350/376], Loss: total=3.377, reg=1.570, containing_o
bj=0.788, no_obj=0.127, cls=0.892
Updating best val loss: 3.68438


Starting epoch 12 / 50
Learning Rate for this epoch: 0.001
Epoch [12/50], Iter [50/376], Loss: total=3.250, reg=1.551, containing_ob
j=0.788, no_obj=0.122, cls=0.788
Epoch [12/50], Iter [100/376], Loss: total=3.211, reg=1.493, containing_o
bj=0.790, no_obj=0.128, cls=0.800
Epoch [12/50], Iter [150/376], Loss: total=3.230, reg=1.505, containing_o
bj=0.791, no_obj=0.129, cls=0.805
Epoch [12/50], Iter [200/376], Loss: total=3.224, reg=1.489, containing_o
bj=0.797, no_obj=0.130, cls=0.809
Epoch [12/50], Iter [250/376], Loss: total=3.246, reg=1.492, containing_o
bj=0.805, no_obj=0.131, cls=0.818
Epoch [12/50], Iter [300/376], Loss: total=3.238, reg=1.492, containing_o
bj=0.796, no_obj=0.133, cls=0.817
```

```
Epoch [12/50], Iter [350/376], Loss: total=3.229, reg=1.494, containing_o
bj=0.794, no_obj=0.132, cls=0.810
Updating best val loss: 3.62757


Starting epoch 13 / 50
Learning Rate for this epoch: 0.001
Epoch [13/50], Iter [50/376], Loss: total=3.182, reg=1.532, containing_ob
j=0.808, no_obj=0.125, cls=0.717
Epoch [13/50], Iter [100/376], Loss: total=3.209, reg=1.543, containing_o
bj=0.794, no_obj=0.130, cls=0.742
Epoch [13/50], Iter [150/376], Loss: total=3.207, reg=1.536, containing_o
bj=0.790, no_obj=0.133, cls=0.747
Epoch [13/50], Iter [200/376], Loss: total=3.182, reg=1.527, containing_o
bj=0.790, no_obj=0.134, cls=0.731
Epoch [13/50], Iter [250/376], Loss: total=3.239, reg=1.541, containing_o
bj=0.800, no_obj=0.136, cls=0.762
Epoch [13/50], Iter [300/376], Loss: total=3.227, reg=1.534, containing_o
bj=0.794, no_obj=0.137, cls=0.762
Epoch [13/50], Iter [350/376], Loss: total=3.206, reg=1.525, containing_o
bj=0.791, no_obj=0.138, cls=0.752
Updating best val loss: 3.44784


Starting epoch 14 / 50
Learning Rate for this epoch: 0.001
Epoch [14/50], Iter [50/376], Loss: total=3.159, reg=1.526, containing_ob
j=0.787, no_obj=0.142, cls=0.703
Epoch [14/50], Iter [100/376], Loss: total=3.127, reg=1.486, containing_o
bj=0.793, no_obj=0.143, cls=0.704
Epoch [14/50], Iter [150/376], Loss: total=3.075, reg=1.446, containing_o
bj=0.777, no_obj=0.147, cls=0.705
Epoch [14/50], Iter [200/376], Loss: total=3.049, reg=1.437, containing_o
bj=0.778, no_obj=0.145, cls=0.689
Epoch [14/50], Iter [250/376], Loss: total=3.059, reg=1.447, containing_o
bj=0.773, no_obj=0.145, cls=0.694
Epoch [14/50], Iter [300/376], Loss: total=3.033, reg=1.428, containing_o
bj=0.772, no_obj=0.145, cls=0.688
Epoch [14/50], Iter [350/376], Loss: total=3.029, reg=1.430, containing_o
bj=0.771, no_obj=0.145, cls=0.683


Starting epoch 15 / 50
Learning Rate for this epoch: 0.001
Epoch [15/50], Iter [50/376], Loss: total=3.091, reg=1.536, containing_ob
j=0.751, no_obj=0.142, cls=0.663
Epoch [15/50], Iter [100/376], Loss: total=3.027, reg=1.480, containing_o
bj=0.759, no_obj=0.142, cls=0.646
Epoch [15/50], Iter [150/376], Loss: total=2.975, reg=1.436, containing_o
bj=0.752, no_obj=0.146, cls=0.641
Epoch [15/50], Iter [200/376], Loss: total=2.972, reg=1.431, containing_o
bj=0.752, no_obj=0.146, cls=0.642
Epoch [15/50], Iter [250/376], Loss: total=2.990, reg=1.441, containing_o
bj=0.759, no_obj=0.147, cls=0.643
Epoch [15/50], Iter [300/376], Loss: total=3.006, reg=1.449, containing_o
bj=0.762, no_obj=0.147, cls=0.648
Epoch [15/50], Iter [350/376], Loss: total=2.976, reg=1.431, containing_o
bj=0.756, no_obj=0.147, cls=0.642
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:29<00:00, 42.57it/s]
```

```
---class aeroplane ap 0.4045504648081668---
---class bicycle ap 0.4463032691884042---
---class bird ap 0.2830405492191656---
---class boat ap 0.11454245410421258---
---class bottle ap 0.04894188859979564---
---class bus ap 0.30500097125097136---
---class car ap 0.4243559011497385---
---class cat ap 0.44621142185097856---
---class chair ap 0.10666651101138883---
---class cow ap 0.16640451957801572---
---class diningtable ap 0.0985134794658604---
---class dog ap 0.31808471724911386---
---class horse ap 0.3777354208684758---
---class motorbike ap 0.358902738910503---
---class person ap 0.3067608130342685---
---class pottedplant ap 0.04072315621518484---
---class sheep ap 0.07188683835163946---
---class sofa ap 0.20867516023574773---
---class train ap 0.5509435840315452---
---class tvmonitor ap 0.25565016639035504---
---map 0.26669470127567657---
14 [0.4045504648081668, 0.4463032691884042, 0.2830405492191656, 0.1145424
5410421258, 0.04894188859979564, 0.30500097125097136, 0.4243559011497385,
0.44621142185097856, 0.10666651101138883, 0.16640451957801572, 0.09851347
94658604, 0.31808471724911386, 0.3777354208684758, 0.358902738910503, 0.3
067608130342685, 0.04072315621518484, 0.07188683835163946, 0.208675160235
74773, 0.5509435840315452, 0.25565016639035504]
Updating best val loss: 3.35581


Starting epoch 16 / 50
Learning Rate for this epoch: 0.001
Epoch [16/50], Iter [50/376], Loss: total=2.722, reg=1.274, containing_ob
j=0.713, no_obj=0.154, cls=0.581
Epoch [16/50], Iter [100/376], Loss: total=2.781, reg=1.311, containing_o
bj=0.722, no_obj=0.151, cls=0.597
Epoch [16/50], Iter [150/376], Loss: total=2.795, reg=1.344, containing_o
bj=0.720, no_obj=0.151, cls=0.581
Epoch [16/50], Iter [200/376], Loss: total=2.807, reg=1.346, containing_o
bj=0.730, no_obj=0.151, cls=0.582
Epoch [16/50], Iter [250/376], Loss: total=2.822, reg=1.354, containing_o
bj=0.731, no_obj=0.152, cls=0.585
Epoch [16/50], Iter [300/376], Loss: total=2.834, reg=1.362, containing_o
bj=0.736, no_obj=0.152, cls=0.584
Epoch [16/50], Iter [350/376], Loss: total=2.841, reg=1.362, containing_o
bj=0.737, no_obj=0.153, cls=0.590
Updating best val loss: 3.28643


Starting epoch 17 / 50
Learning Rate for this epoch: 0.001
Epoch [17/50], Iter [50/376], Loss: total=2.765, reg=1.267, containing_ob
j=0.760, no_obj=0.163, cls=0.575
Epoch [17/50], Iter [100/376], Loss: total=2.696, reg=1.266, containing_o
bj=0.724, no_obj=0.161, cls=0.544
Epoch [17/50], Iter [150/376], Loss: total=2.768, reg=1.306, containing_o
bj=0.752, no_obj=0.160, cls=0.550
Epoch [17/50], Iter [200/376], Loss: total=2.757, reg=1.302, containing_o
bj=0.744, no_obj=0.158, cls=0.553
Epoch [17/50], Iter [250/376], Loss: total=2.748, reg=1.303, containing_o
bj=0.740, no_obj=0.158, cls=0.548
Epoch [17/50], Iter [300/376], Loss: total=2.770, reg=1.315, containing_o
bj=0.743, no_obj=0.157, cls=0.555
```

```
Epoch [17/50], Iter [350/376], Loss: total=2.783, reg=1.328, containing_o
bj=0.746, no_obj=0.157, cls=0.552


Starting epoch 18 / 50
Learning Rate for this epoch: 0.001
Epoch [18/50], Iter [50/376], Loss: total=2.674, reg=1.293, containing_ob
j=0.717, no_obj=0.156, cls=0.508
Epoch [18/50], Iter [100/376], Loss: total=2.732, reg=1.320, containing_o
bj=0.741, no_obj=0.157, cls=0.514
Epoch [18/50], Iter [150/376], Loss: total=2.682, reg=1.288, containing_o
bj=0.735, no_obj=0.158, cls=0.500
Epoch [18/50], Iter [200/376], Loss: total=2.645, reg=1.274, containing_o
bj=0.721, no_obj=0.160, cls=0.490
Epoch [18/50], Iter [250/376], Loss: total=2.670, reg=1.287, containing_o
bj=0.724, no_obj=0.162, cls=0.498
Epoch [18/50], Iter [300/376], Loss: total=2.716, reg=1.312, containing_o
bj=0.738, no_obj=0.161, cls=0.505
Epoch [18/50], Iter [350/376], Loss: total=2.714, reg=1.302, containing_o
bj=0.739, no_obj=0.161, cls=0.511
Updating best val loss: 3.23497


Starting epoch 19 / 50
Learning Rate for this epoch: 0.001
Epoch [19/50], Iter [50/376], Loss: total=2.824, reg=1.351, containing_ob
j=0.792, no_obj=0.154, cls=0.526
Epoch [19/50], Iter [100/376], Loss: total=2.783, reg=1.352, containing_o
bj=0.757, no_obj=0.159, cls=0.515
Epoch [19/50], Iter [150/376], Loss: total=2.750, reg=1.344, containing_o
bj=0.749, no_obj=0.163, cls=0.495
Epoch [19/50], Iter [200/376], Loss: total=2.736, reg=1.324, containing_o
bj=0.745, no_obj=0.165, cls=0.502
Epoch [19/50], Iter [250/376], Loss: total=2.711, reg=1.317, containing_o
bj=0.728, no_obj=0.166, cls=0.500
Epoch [19/50], Iter [300/376], Loss: total=2.716, reg=1.318, containing_o
bj=0.731, no_obj=0.166, cls=0.501
Epoch [19/50], Iter [350/376], Loss: total=2.691, reg=1.305, containing_o
bj=0.723, no_obj=0.166, cls=0.496
Updating best val loss: 3.18714


Starting epoch 20 / 50
Learning Rate for this epoch: 0.001
Epoch [20/50], Iter [50/376], Loss: total=2.646, reg=1.266, containing_ob
j=0.739, no_obj=0.164, cls=0.477
Epoch [20/50], Iter [100/376], Loss: total=2.598, reg=1.248, containing_o
bj=0.733, no_obj=0.168, cls=0.448
Epoch [20/50], Iter [150/376], Loss: total=2.627, reg=1.249, containing_o
bj=0.738, no_obj=0.171, cls=0.468
Epoch [20/50], Iter [200/376], Loss: total=2.622, reg=1.246, containing_o
bj=0.733, no_obj=0.170, cls=0.474
Epoch [20/50], Iter [250/376], Loss: total=2.624, reg=1.245, containing_o
bj=0.726, no_obj=0.168, cls=0.484
Epoch [20/50], Iter [300/376], Loss: total=2.606, reg=1.234, containing_o
bj=0.727, no_obj=0.168, cls=0.476
Epoch [20/50], Iter [350/376], Loss: total=2.626, reg=1.243, containing_o
bj=0.732, no_obj=0.169, cls=0.482
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:30<00:00, 41.55it/s]
```

```
---class aeroplane ap 0.463734176361879---
---class bicycle ap 0.41961030567924384---
---class bird ap 0.3448736285758268---
---class boat ap 0.19848204434841468---
---class bottle ap 0.052481380012674---
---class bus ap 0.4726812076812077---
---class car ap 0.5336402697097424---
---class cat ap 0.521963302512223---
---class chair ap 0.1815518612753533---
---class cow ap 0.2715012236149891---
---class diningtable ap 0.21904761904761905---
---class dog ap 0.36781198029490775---
---class horse ap 0.49332605928915324---
---class motorbike ap 0.47378755714414655---
---class person ap 0.39169918672039705---
---class pottedplant ap 0.12707049419900188---
---class sheep ap 0.10316951972513716---
---class sofa ap 0.2533215701562475---
---class train ap 0.6177718157696752---
---class tvmonitor ap 0.316290172522788---
---map 0.3411907687320313---
19 [0.463734176361879, 0.41961030567924384, 0.3448736285758268, 0.1984820
4434841468, 0.052481380012674, 0.4726812076812077, 0.5336402697097424, 0.
521963302512223, 0.1815518612753533, 0.2715012236149891, 0.21904761904761
905, 0.36781198029490775, 0.49332605928915324, 0.47378755714414655, 0.391
69918672039705, 0.12707049419900188, 0.10316951972513716, 0.2533215701562
475, 0.6177718157696752, 0.316290172522788]
Updating best val loss: 3.15271


Starting epoch 21 / 50
Learning Rate for this epoch: 0.001
Epoch [21/50], Iter [50/376], Loss: total=2.365, reg=1.132, containing_ob
j=0.692, no_obj=0.159, cls=0.381
Epoch [21/50], Iter [100/376], Loss: total=2.516, reg=1.206, containing_o
bj=0.720, no_obj=0.167, cls=0.424
Epoch [21/50], Iter [150/376], Loss: total=2.531, reg=1.205, containing_o
bj=0.722, no_obj=0.169, cls=0.436
Epoch [21/50], Iter [200/376], Loss: total=2.538, reg=1.205, containing_o
bj=0.717, no_obj=0.171, cls=0.445
Epoch [21/50], Iter [250/376], Loss: total=2.550, reg=1.211, containing_o
bj=0.723, no_obj=0.171, cls=0.444
Epoch [21/50], Iter [300/376], Loss: total=2.567, reg=1.221, containing_o
bj=0.722, no_obj=0.170, cls=0.454
Epoch [21/50], Iter [350/376], Loss: total=2.574, reg=1.226, containing_o
bj=0.717, no_obj=0.171, cls=0.460
Updating best val loss: 3.11067


Starting epoch 22 / 50
Learning Rate for this epoch: 0.001
Epoch [22/50], Iter [50/376], Loss: total=2.577, reg=1.257, containing_ob
j=0.687, no_obj=0.174, cls=0.458
Epoch [22/50], Iter [100/376], Loss: total=2.533, reg=1.234, containing_o
bj=0.682, no_obj=0.177, cls=0.439
Epoch [22/50], Iter [150/376], Loss: total=2.459, reg=1.187, containing_o
bj=0.677, no_obj=0.179, cls=0.417
Epoch [22/50], Iter [200/376], Loss: total=2.479, reg=1.193, containing_o
bj=0.682, no_obj=0.177, cls=0.427
Epoch [22/50], Iter [250/376], Loss: total=2.493, reg=1.200, containing_o
bj=0.687, no_obj=0.175, cls=0.431
Epoch [22/50], Iter [300/376], Loss: total=2.498, reg=1.201, containing_o
bj=0.693, no_obj=0.175, cls=0.429
```

```
Epoch [22/50], Iter [350/376], Loss: total=2.521, reg=1.213, containing_o
bj=0.701, no_obj=0.174, cls=0.433


Starting epoch 23 / 50
Learning Rate for this epoch: 0.001
Epoch [23/50], Iter [50/376], Loss: total=2.365, reg=1.091, containing_ob
j=0.700, no_obj=0.184, cls=0.389
Epoch [23/50], Iter [100/376], Loss: total=2.459, reg=1.179, containing_o
bj=0.699, no_obj=0.182, cls=0.399
Epoch [23/50], Iter [150/376], Loss: total=2.424, reg=1.153, containing_o
bj=0.692, no_obj=0.179, cls=0.400
Epoch [23/50], Iter [200/376], Loss: total=2.452, reg=1.178, containing_o
bj=0.700, no_obj=0.178, cls=0.396
Epoch [23/50], Iter [250/376], Loss: total=2.456, reg=1.176, containing_o
bj=0.699, no_obj=0.177, cls=0.403
Epoch [23/50], Iter [300/376], Loss: total=2.488, reg=1.201, containing_o
bj=0.707, no_obj=0.176, cls=0.404
Epoch [23/50], Iter [350/376], Loss: total=2.486, reg=1.199, containing_o
bj=0.708, no_obj=0.175, cls=0.404


Starting epoch 24 / 50
Learning Rate for this epoch: 0.001
Epoch [24/50], Iter [50/376], Loss: total=2.499, reg=1.209, containing_ob
j=0.698, no_obj=0.173, cls=0.421
Epoch [24/50], Iter [100/376], Loss: total=2.428, reg=1.173, containing_o
bj=0.690, no_obj=0.169, cls=0.396
Epoch [24/50], Iter [150/376], Loss: total=2.432, reg=1.171, containing_o
bj=0.688, no_obj=0.170, cls=0.403
Epoch [24/50], Iter [200/376], Loss: total=2.414, reg=1.167, containing_o
bj=0.685, no_obj=0.171, cls=0.391
Epoch [24/50], Iter [250/376], Loss: total=2.418, reg=1.170, containing_o
bj=0.690, no_obj=0.171, cls=0.387
Epoch [24/50], Iter [300/376], Loss: total=2.449, reg=1.191, containing_o
bj=0.694, no_obj=0.172, cls=0.393
Epoch [24/50], Iter [350/376], Loss: total=2.446, reg=1.187, containing_o
bj=0.697, no_obj=0.173, cls=0.389
Updating best val loss: 3.10698


Starting epoch 25 / 50
Learning Rate for this epoch: 0.001
Epoch [25/50], Iter [50/376], Loss: total=2.414, reg=1.173, containing_ob
j=0.681, no_obj=0.188, cls=0.373
Epoch [25/50], Iter [100/376], Loss: total=2.400, reg=1.175, containing_o
bj=0.685, no_obj=0.183, cls=0.357
Epoch [25/50], Iter [150/376], Loss: total=2.437, reg=1.191, containing_o
bj=0.693, no_obj=0.182, cls=0.370
Epoch [25/50], Iter [200/376], Loss: total=2.418, reg=1.175, containing_o
bj=0.692, no_obj=0.182, cls=0.369
Epoch [25/50], Iter [250/376], Loss: total=2.441, reg=1.179, containing_o
bj=0.702, no_obj=0.181, cls=0.378
Epoch [25/50], Iter [300/376], Loss: total=2.417, reg=1.168, containing_o
bj=0.692, no_obj=0.181, cls=0.376
Epoch [25/50], Iter [350/376], Loss: total=2.410, reg=1.163, containing_o
bj=0.688, no_obj=0.181, cls=0.379
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:29<00:00, 42.31it/s]
```

```
---class aeroplane ap 0.5218103515577985---
---class bicycle ap 0.4708644592327018---
---class bird ap 0.3294436582210608---
---class boat ap 0.2194489953298088---
---class bottle ap 0.07634967993151967---
---class bus ap 0.429233474248137---
---class car ap 0.5398943111691781---
---class cat ap 0.5520262059729792---
---class chair ap 0.22115497736966336---
---class cow ap 0.3013534906053184---
---class diningtable ap 0.24788971144013158---
---class dog ap 0.4587325670314989---
---class horse ap 0.5623657607662635---
---class motorbike ap 0.38571302992260625---
---class person ap 0.3992836230898037---
---class pottedplant ap 0.09453575983336408---
---class sheep ap 0.2514674463689348---
---class sofa ap 0.3294820069245356---
---class train ap 0.6489954178320607---
---class tvmonitor ap 0.35931971256692563---
---map 0.3699682319707145---
24 [0.5218103515577985, 0.4708644592327018, 0.3294436582210608, 0.2194489
953298088, 0.07634967993151967, 0.429233474248137, 0.5398943111691781, 0.
5520262059729792, 0.22115497736966336, 0.3013534906053184, 0.247889711440
13158, 0.4587325670314989, 0.5623657607662635, 0.38571302992260625, 0.399
2836230898037, 0.09453575983336408, 0.2514674463689348, 0.329482006924535
6, 0.6489954178320607, 0.35931971256692563]
Updating best val loss: 3.09639


Starting epoch 26 / 50
Learning Rate for this epoch: 0.001
Epoch [26/50], Iter [50/376], Loss: total=2.409, reg=1.146, containing_ob
j=0.716, no_obj=0.174, cls=0.373
Epoch [26/50], Iter [100/376], Loss: total=2.373, reg=1.120, containing_o
bj=0.707, no_obj=0.179, cls=0.367
Epoch [26/50], Iter [150/376], Loss: total=2.362, reg=1.131, containing_o
bj=0.696, no_obj=0.183, cls=0.351
Epoch [26/50], Iter [200/376], Loss: total=2.331, reg=1.113, containing_o
bj=0.689, no_obj=0.183, cls=0.345
Epoch [26/50], Iter [250/376], Loss: total=2.349, reg=1.120, containing_o
bj=0.696, no_obj=0.183, cls=0.350
Epoch [26/50], Iter [300/376], Loss: total=2.354, reg=1.120, containing_o
bj=0.695, no_obj=0.183, cls=0.355
Epoch [26/50], Iter [350/376], Loss: total=2.366, reg=1.130, containing_o
bj=0.701, no_obj=0.181, cls=0.355
Updating best val loss: 3.07667


Starting epoch 27 / 50
Learning Rate for this epoch: 0.001
Epoch [27/50], Iter [50/376], Loss: total=2.433, reg=1.193, containing_ob
j=0.705, no_obj=0.171, cls=0.363
Epoch [27/50], Iter [100/376], Loss: total=2.381, reg=1.163, containing_o
bj=0.684, no_obj=0.180, cls=0.354
Epoch [27/50], Iter [150/376], Loss: total=2.334, reg=1.130, containing_o
bj=0.686, no_obj=0.182, cls=0.337
Epoch [27/50], Iter [200/376], Loss: total=2.353, reg=1.134, containing_o
bj=0.695, no_obj=0.182, cls=0.342
Epoch [27/50], Iter [250/376], Loss: total=2.349, reg=1.136, containing_o
bj=0.688, no_obj=0.183, cls=0.340
Epoch [27/50], Iter [300/376], Loss: total=2.363, reg=1.146, containing_o
bj=0.690, no_obj=0.183, cls=0.344
```

```
Epoch [27/50], Iter [350/376], Loss: total=2.360, reg=1.144, containing_o
bj=0.687, no_obj=0.184, cls=0.345
Updating best val loss: 3.07555


Starting epoch 28 / 50
Learning Rate for this epoch: 0.001
Epoch [28/50], Iter [50/376], Loss: total=2.030, reg=0.946, containing_ob
j=0.603, no_obj=0.179, cls=0.303
Epoch [28/50], Iter [100/376], Loss: total=2.123, reg=0.980, containing_o
bj=0.631, no_obj=0.182, cls=0.330
Epoch [28/50], Iter [150/376], Loss: total=2.264, reg=1.059, containing_o
bj=0.675, no_obj=0.188, cls=0.341
Epoch [28/50], Iter [200/376], Loss: total=2.247, reg=1.056, containing_o
bj=0.660, no_obj=0.191, cls=0.341
Epoch [28/50], Iter [250/376], Loss: total=2.257, reg=1.066, containing_o
bj=0.665, no_obj=0.189, cls=0.337
Epoch [28/50], Iter [300/376], Loss: total=2.259, reg=1.067, containing_o
bj=0.676, no_obj=0.188, cls=0.327
Epoch [28/50], Iter [350/376], Loss: total=2.272, reg=1.072, containing_o
bj=0.679, no_obj=0.188, cls=0.333
Updating best val loss: 3.05180


Starting epoch 29 / 50
Learning Rate for this epoch: 0.001
Epoch [29/50], Iter [50/376], Loss: total=2.099, reg=0.987, containing_ob
j=0.622, no_obj=0.192, cls=0.299
Epoch [29/50], Iter [100/376], Loss: total=2.243, reg=1.074, containing_o
bj=0.661, no_obj=0.186, cls=0.322
Epoch [29/50], Iter [150/376], Loss: total=2.263, reg=1.083, containing_o
bj=0.675, no_obj=0.188, cls=0.317
Epoch [29/50], Iter [200/376], Loss: total=2.238, reg=1.074, containing_o
bj=0.668, no_obj=0.185, cls=0.311
Epoch [29/50], Iter [250/376], Loss: total=2.240, reg=1.075, containing_o
bj=0.668, no_obj=0.187, cls=0.311
Epoch [29/50], Iter [300/376], Loss: total=2.260, reg=1.080, containing_o
bj=0.672, no_obj=0.187, cls=0.321
Epoch [29/50], Iter [350/376], Loss: total=2.264, reg=1.077, containing_o
bj=0.672, no_obj=0.188, cls=0.327


Starting epoch 30 / 50
Learning Rate for this epoch: 0.001
Epoch [30/50], Iter [50/376], Loss: total=2.315, reg=1.103, containing_ob
j=0.737, no_obj=0.187, cls=0.288
Epoch [30/50], Iter [100/376], Loss: total=2.241, reg=1.073, containing_o
bj=0.680, no_obj=0.187, cls=0.301
Epoch [30/50], Iter [150/376], Loss: total=2.277, reg=1.074, containing_o
bj=0.694, no_obj=0.185, cls=0.324
Epoch [30/50], Iter [200/376], Loss: total=2.258, reg=1.065, containing_o
bj=0.683, no_obj=0.188, cls=0.321
Epoch [30/50], Iter [250/376], Loss: total=2.246, reg=1.059, containing_o
bj=0.675, no_obj=0.188, cls=0.324
Epoch [30/50], Iter [300/376], Loss: total=2.229, reg=1.052, containing_o
bj=0.673, no_obj=0.188, cls=0.315
Epoch [30/50], Iter [350/376], Loss: total=2.231, reg=1.055, containing_o
bj=0.675, no_obj=0.187, cls=0.314
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:30<00:00, 41.64it/s]
```

```
---class aeroplane ap 0.5467547067562715---
---class bicycle ap 0.2863469863469863---
---class bird ap 0.4170628429507444---
---class boat ap 0.24993316829057033---
---class bottle ap 0.09088491633972368---
---class bus ap 0.5725843951466496---
---class car ap 0.5278500722928762---
---class cat ap 0.5877564306385779---
---class chair ap 0.1460357809678931---
---class cow ap 0.27852150965755457---
---class diningtable ap 0.285407496331866---
---class dog ap 0.5024480837649319---
---class horse ap 0.5257528702147107---
---class motorbike ap 0.5020982218475488---
---class person ap 0.42966190804762466---
---class pottedplant ap 0.13467542955281012---
---class sheep ap 0.2370838302764299---
---class sofa ap 0.34880121568094735---
---class train ap 0.6826296942612603---
---class tvmonitor ap 0.4977820749551855---
---map 0.39250358171605815---
29 [0.5467547067562715, 0.2863469863469863, 0.4170628429507444, 0.2499331
6829057033, 0.09088491633972368, 0.5725843951466496, 0.5278500722928762,
0.5877564306385779, 0.1460357809678931, 0.27852150965755457, 0.2854074963
31866, 0.5024480837649319, 0.5257528702147107, 0.5020982218475488, 0.4296
6190804762466, 0.13467542955281012, 0.2370838302764299, 0.348801215680947
35, 0.6826296942612603, 0.4977820749551855]


Starting epoch 31 / 50
Learning Rate for this epoch: 0.0001
Epoch [31/50], Iter [50/376], Loss: total=2.122, reg=0.992, containing_ob
j=0.632, no_obj=0.205, cls=0.293
Epoch [31/50], Iter [100/376], Loss: total=2.153, reg=1.027, containing_o
bj=0.643, no_obj=0.202, cls=0.282
Epoch [31/50], Iter [150/376], Loss: total=2.093, reg=0.992, containing_o
bj=0.636, no_obj=0.200, cls=0.265
Epoch [31/50], Iter [200/376], Loss: total=2.081, reg=0.987, containing_o
bj=0.634, no_obj=0.199, cls=0.262
Epoch [31/50], Iter [250/376], Loss: total=2.096, reg=0.991, containing_o
bj=0.632, no_obj=0.199, cls=0.274
Epoch [31/50], Iter [300/376], Loss: total=2.098, reg=0.988, containing_o
bj=0.636, no_obj=0.199, cls=0.275
Epoch [31/50], Iter [350/376], Loss: total=2.083, reg=0.981, containing_o
bj=0.634, no_obj=0.199, cls=0.268
Updating best val loss: 2.91346


Starting epoch 32 / 50
Learning Rate for this epoch: 0.0001
Epoch [32/50], Iter [50/376], Loss: total=2.146, reg=1.013, containing_ob
j=0.676, no_obj=0.194, cls=0.263
Epoch [32/50], Iter [100/376], Loss: total=2.078, reg=0.969, containing_o
bj=0.655, no_obj=0.197, cls=0.257
Epoch [32/50], Iter [150/376], Loss: total=2.008, reg=0.938, containing_o
bj=0.621, no_obj=0.198, cls=0.251
Epoch [32/50], Iter [200/376], Loss: total=2.024, reg=0.956, containing_o
bj=0.629, no_obj=0.197, cls=0.243
Epoch [32/50], Iter [250/376], Loss: total=2.049, reg=0.967, containing_o
bj=0.639, no_obj=0.195, cls=0.248
Epoch [32/50], Iter [300/376], Loss: total=2.023, reg=0.955, containing_o
bj=0.629, no_obj=0.196, cls=0.243
Epoch [32/50], Iter [350/376], Loss: total=2.021, reg=0.955, containing_o
```

```
bj=0.630, no_obj=0.196, cls=0.240
Updating best val loss: 2.89792


Starting epoch 33 / 50
Learning Rate for this epoch: 0.0001
Epoch [33/50], Iter [50/376], Loss: total=1.910, reg=0.911, containing_ob
j=0.598, no_obj=0.194, cls=0.208
Epoch [33/50], Iter [100/376], Loss: total=1.934, reg=0.910, containing_o
bj=0.603, no_obj=0.197, cls=0.224
Epoch [33/50], Iter [150/376], Loss: total=1.952, reg=0.928, containing_o
bj=0.602, no_obj=0.194, cls=0.227
Epoch [33/50], Iter [200/376], Loss: total=1.958, reg=0.924, containing_o
bj=0.609, no_obj=0.193, cls=0.232
Epoch [33/50], Iter [250/376], Loss: total=1.960, reg=0.931, containing_o
bj=0.607, no_obj=0.193, cls=0.228
Epoch [33/50], Iter [300/376], Loss: total=1.975, reg=0.937, containing_o
bj=0.616, no_obj=0.193, cls=0.229
Epoch [33/50], Iter [350/376], Loss: total=1.978, reg=0.937, containing_o
bj=0.618, no_obj=0.197, cls=0.226
Updating best val loss: 2.89615


Starting epoch 34 / 50
Learning Rate for this epoch: 0.0001
Epoch [34/50], Iter [50/376], Loss: total=1.874, reg=0.876, containing_ob
j=0.580, no_obj=0.201, cls=0.217
Epoch [34/50], Iter [100/376], Loss: total=1.961, reg=0.910, containing_o
bj=0.633, no_obj=0.199, cls=0.219
Epoch [34/50], Iter [150/376], Loss: total=1.978, reg=0.929, containing_o
bj=0.630, no_obj=0.200, cls=0.219
Epoch [34/50], Iter [200/376], Loss: total=1.988, reg=0.936, containing_o
bj=0.628, no_obj=0.199, cls=0.223
Epoch [34/50], Iter [250/376], Loss: total=1.972, reg=0.926, containing_o
bj=0.625, no_obj=0.199, cls=0.222
Epoch [34/50], Iter [300/376], Loss: total=1.965, reg=0.922, containing_o
bj=0.624, no_obj=0.198, cls=0.222
Epoch [34/50], Iter [350/376], Loss: total=1.963, reg=0.918, containing_o
bj=0.624, no_obj=0.196, cls=0.224


Starting epoch 35 / 50
Learning Rate for this epoch: 0.0001
Epoch [35/50], Iter [50/376], Loss: total=1.883, reg=0.898, containing_ob
j=0.602, no_obj=0.206, cls=0.176
Epoch [35/50], Iter [100/376], Loss: total=1.934, reg=0.922, containing_o
bj=0.615, no_obj=0.198, cls=0.199
Epoch [35/50], Iter [150/376], Loss: total=1.930, reg=0.924, containing_o
bj=0.602, no_obj=0.196, cls=0.208
Epoch [35/50], Iter [200/376], Loss: total=1.953, reg=0.933, containing_o
bj=0.617, no_obj=0.194, cls=0.209
Epoch [35/50], Iter [250/376], Loss: total=1.974, reg=0.946, containing_o
bj=0.616, no_obj=0.196, cls=0.217
Epoch [35/50], Iter [300/376], Loss: total=1.984, reg=0.952, containing_o
bj=0.623, no_obj=0.195, cls=0.213
Epoch [35/50], Iter [350/376], Loss: total=1.974, reg=0.948, containing_o
bj=0.619, no_obj=0.195, cls=0.212
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:30<00:00, 41.57it/s]
```

```
---class aeroplane ap 0.6377968686238712---
---class bicycle ap 0.5667887280495053---
---class bird ap 0.46997222536825306---
---class boat ap 0.37182674467463855---
---class bottle ap 0.09680341688317132---
---class bus ap 0.5790651220639974---
---class car ap 0.6075175676983549---
---class cat ap 0.6120606141089372---
---class chair ap 0.24257145866370386---
---class cow ap 0.44134634392027794---
---class diningtable ap 0.4123018004078838---
---class dog ap 0.5482859765218625---
---class horse ap 0.6060001096126354---
---class motorbike ap 0.4835638087882555---
---class person ap 0.4823366068407742---
---class pottedplant ap 0.16793959773763412---
---class sheep ap 0.21527094867099883---
---class sofa ap 0.44712695289110366---
---class train ap 0.7427171182784261---
---class tvmonitor ap 0.4859033888937304---
---map 0.46085976993490074---
34 [0.6377968686238712, 0.5667887280495053, 0.46997222536825306, 0.371826
74467463855, 0.09680341688317132, 0.5790651220639974, 0.6075175676983549,
0.6120606141089372, 0.24257145866370386, 0.44134634392027794, 0.412301800
4078838, 0.5482859765218625, 0.6060001096126354, 0.4835638087882555, 0.48
23366068407742, 0.16793959773763412, 0.21527094867099883, 0.4471269528911
0366, 0.7427171182784261, 0.4859033888937304]
Updating best val loss: 2.88914


Starting epoch 36 / 50
Learning Rate for this epoch: 0.0001
Epoch [36/50], Iter [50/376], Loss: total=1.925, reg=0.911, containing_ob
j=0.615, no_obj=0.189, cls=0.209
Epoch [36/50], Iter [100/376], Loss: total=1.920, reg=0.908, containing_o
bj=0.616, no_obj=0.191, cls=0.204
Epoch [36/50], Iter [150/376], Loss: total=1.945, reg=0.919, containing_o
bj=0.622, no_obj=0.194, cls=0.210
Epoch [36/50], Iter [200/376], Loss: total=1.932, reg=0.908, containing_o
bj=0.615, no_obj=0.196, cls=0.213
Epoch [36/50], Iter [250/376], Loss: total=1.925, reg=0.906, containing_o
bj=0.610, no_obj=0.196, cls=0.212
Epoch [36/50], Iter [300/376], Loss: total=1.933, reg=0.912, containing_o
bj=0.610, no_obj=0.198, cls=0.213
Epoch [36/50], Iter [350/376], Loss: total=1.932, reg=0.909, containing_o
bj=0.613, no_obj=0.199, cls=0.211
Updating best val loss: 2.88862


Starting epoch 37 / 50
Learning Rate for this epoch: 0.0001
Epoch [37/50], Iter [50/376], Loss: total=1.929, reg=0.904, containing_ob
j=0.611, no_obj=0.188, cls=0.226
Epoch [37/50], Iter [100/376], Loss: total=1.914, reg=0.899, containing_o
bj=0.613, no_obj=0.189, cls=0.212
Epoch [37/50], Iter [150/376], Loss: total=1.915, reg=0.907, containing_o
bj=0.602, no_obj=0.192, cls=0.214
Epoch [37/50], Iter [200/376], Loss: total=1.897, reg=0.891, containing_o
bj=0.601, no_obj=0.193, cls=0.212
Epoch [37/50], Iter [250/376], Loss: total=1.893, reg=0.882, containing_o
bj=0.608, no_obj=0.195, cls=0.208
Epoch [37/50], Iter [300/376], Loss: total=1.883, reg=0.882, containing_o
bj=0.602, no_obj=0.196, cls=0.203
```

```
Epoch [37/50], Iter [350/376], Loss: total=1.890, reg=0.892, containing_o
bj=0.600, no_obj=0.196, cls=0.201


Starting epoch 38 / 50
Learning Rate for this epoch: 0.0001
Epoch [38/50], Iter [50/376], Loss: total=2.028, reg=0.960, containing_ob
j=0.655, no_obj=0.197, cls=0.216
Epoch [38/50], Iter [100/376], Loss: total=1.961, reg=0.925, containing_o
bj=0.614, no_obj=0.203, cls=0.219
Epoch [38/50], Iter [150/376], Loss: total=1.926, reg=0.905, containing_o
bj=0.602, no_obj=0.201, cls=0.219
Epoch [38/50], Iter [200/376], Loss: total=1.889, reg=0.881, containing_o
bj=0.599, no_obj=0.197, cls=0.212
Epoch [38/50], Iter [250/376], Loss: total=1.888, reg=0.887, containing_o
bj=0.602, no_obj=0.195, cls=0.204
Epoch [38/50], Iter [300/376], Loss: total=1.893, reg=0.888, containing_o
bj=0.600, no_obj=0.197, cls=0.207
Epoch [38/50], Iter [350/376], Loss: total=1.904, reg=0.894, containing_o
bj=0.606, no_obj=0.196, cls=0.207
Updating best val loss: 2.87321


Starting epoch 39 / 50
Learning Rate for this epoch: 0.0001
Epoch [39/50], Iter [50/376], Loss: total=1.810, reg=0.840, containing_ob
j=0.574, no_obj=0.206, cls=0.190
Epoch [39/50], Iter [100/376], Loss: total=1.797, reg=0.850, containing_o
bj=0.569, no_obj=0.197, cls=0.181
Epoch [39/50], Iter [150/376], Loss: total=1.807, reg=0.854, containing_o
bj=0.565, no_obj=0.200, cls=0.188
Epoch [39/50], Iter [200/376], Loss: total=1.860, reg=0.876, containing_o
bj=0.590, no_obj=0.197, cls=0.197
Epoch [39/50], Iter [250/376], Loss: total=1.893, reg=0.889, containing_o
bj=0.603, no_obj=0.200, cls=0.201
Epoch [39/50], Iter [300/376], Loss: total=1.883, reg=0.884, containing_o
bj=0.599, no_obj=0.200, cls=0.200
Epoch [39/50], Iter [350/376], Loss: total=1.878, reg=0.878, containing_o
bj=0.600, no_obj=0.202, cls=0.199
Updating best val loss: 2.86539


Starting epoch 40 / 50
Learning Rate for this epoch: 0.0001
Epoch [40/50], Iter [50/376], Loss: total=1.770, reg=0.829, containing_ob
j=0.563, no_obj=0.199, cls=0.178
Epoch [40/50], Iter [100/376], Loss: total=1.897, reg=0.905, containing_o
bj=0.605, no_obj=0.197, cls=0.190
Epoch [40/50], Iter [150/376], Loss: total=1.908, reg=0.912, containing_o
bj=0.608, no_obj=0.198, cls=0.191
Epoch [40/50], Iter [200/376], Loss: total=1.887, reg=0.904, containing_o
bj=0.597, no_obj=0.198, cls=0.188
Epoch [40/50], Iter [250/376], Loss: total=1.918, reg=0.920, containing_o
bj=0.601, no_obj=0.199, cls=0.199
Epoch [40/50], Iter [300/376], Loss: total=1.917, reg=0.919, containing_o
bj=0.601, no_obj=0.198, cls=0.198
Epoch [40/50], Iter [350/376], Loss: total=1.900, reg=0.907, containing_o
bj=0.594, no_obj=0.199, cls=0.200
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:31<00:00, 40.46it/s]
```

```
---class aeroplane ap 0.6066087445192966---
---class bicycle ap 0.6281836594807055---
---class bird ap 0.48134857753185806---
---class boat ap 0.33467068117301446---
---class bottle ap 0.11195236760820834---
---class bus ap 0.6042953252921691---
---class car ap 0.5999580808788909---
---class cat ap 0.6784794915498209---
---class chair ap 0.27070023683480604---
---class cow ap 0.41359863185121226---
---class diningtable ap 0.3970850438792401---
---class dog ap 0.5000666930655675---
---class horse ap 0.6227635489522279---
---class motorbike ap 0.5057718706736282---
---class person ap 0.4811362308326159---
---class pottedplant ap 0.2084698489485198---
---class sheep ap 0.23708982197908138---
---class sofa ap 0.4653092890264147---
---class train ap 0.6976349222635745---
---class tvmonitor ap 0.5007717640738744---
---map 0.46729474152073636---
39 [0.6066087445192966, 0.6281836594807055, 0.48134857753185806, 0.334670
68117301446, 0.11195236760820834, 0.6042953252921691, 0.5999580808788909,
0.6784794915498209, 0.27070023683480604, 0.41359863185121226, 0.397085043
8792401, 0.5000666930655675, 0.6227635489522279, 0.5057718706736282, 0.48
11362308326159, 0.2084698489485198, 0.23708982197908138, 0.46530928902641
47, 0.6976349222635745, 0.5007717640738744]
Updating best val loss: 2.86508


Starting epoch 41 / 50
Learning Rate for this epoch: 1e-05
Epoch [41/50], Iter [50/376], Loss: total=1.921, reg=0.917, containing_ob
j=0.605, no_obj=0.201, cls=0.197
Epoch [41/50], Iter [100/376], Loss: total=1.913, reg=0.923, containing_o
bj=0.598, no_obj=0.199, cls=0.194
Epoch [41/50], Iter [150/376], Loss: total=1.889, reg=0.901, containing_o
bj=0.599, no_obj=0.197, cls=0.192
Epoch [41/50], Iter [200/376], Loss: total=1.885, reg=0.893, containing_o
bj=0.596, no_obj=0.196, cls=0.199
Epoch [41/50], Iter [250/376], Loss: total=1.910, reg=0.909, containing_o
bj=0.602, no_obj=0.197, cls=0.203
Epoch [41/50], Iter [300/376], Loss: total=1.893, reg=0.901, containing_o
bj=0.598, no_obj=0.197, cls=0.197
Epoch [41/50], Iter [350/376], Loss: total=1.876, reg=0.886, containing_o
bj=0.598, no_obj=0.197, cls=0.194
Updating best val loss: 2.85467


Starting epoch 42 / 50
Learning Rate for this epoch: 1e-05
Epoch [42/50], Iter [50/376], Loss: total=1.901, reg=0.853, containing_ob
j=0.660, no_obj=0.198, cls=0.190
Epoch [42/50], Iter [100/376], Loss: total=1.934, reg=0.885, containing_o
bj=0.645, no_obj=0.205, cls=0.199
Epoch [42/50], Iter [150/376], Loss: total=1.928, reg=0.887, containing_o
bj=0.640, no_obj=0.201, cls=0.200
Epoch [42/50], Iter [200/376], Loss: total=1.891, reg=0.874, containing_o
bj=0.621, no_obj=0.199, cls=0.197
Epoch [42/50], Iter [250/376], Loss: total=1.894, reg=0.873, containing_o
bj=0.625, no_obj=0.198, cls=0.198
Epoch [42/50], Iter [300/376], Loss: total=1.868, reg=0.861, containing_o
bj=0.610, no_obj=0.198, cls=0.199
```

```
Epoch [42/50], Iter [350/376], Loss: total=1.853, reg=0.857, containing_o
bj=0.604, no_obj=0.199, cls=0.193


Starting epoch 43 / 50
Learning Rate for this epoch: 1e-05
Epoch [43/50], Iter [50/376], Loss: total=1.805, reg=0.835, containing_ob
j=0.582, no_obj=0.198, cls=0.189
Epoch [43/50], Iter [100/376], Loss: total=1.842, reg=0.862, containing_o
bj=0.592, no_obj=0.203, cls=0.185
Epoch [43/50], Iter [150/376], Loss: total=1.882, reg=0.875, containing_o
bj=0.616, no_obj=0.200, cls=0.192
Epoch [43/50], Iter [200/376], Loss: total=1.867, reg=0.868, containing_o
bj=0.607, no_obj=0.200, cls=0.193
Epoch [43/50], Iter [250/376], Loss: total=1.866, reg=0.865, containing_o
bj=0.606, no_obj=0.200, cls=0.194
Epoch [43/50], Iter [300/376], Loss: total=1.855, reg=0.865, containing_o
bj=0.597, no_obj=0.201, cls=0.191
Epoch [43/50], Iter [350/376], Loss: total=1.862, reg=0.866, containing_o
bj=0.601, no_obj=0.202, cls=0.192


Starting epoch 44 / 50
Learning Rate for this epoch: 1e-05
Epoch [44/50], Iter [50/376], Loss: total=2.012, reg=0.918, containing_ob
j=0.669, no_obj=0.195, cls=0.230
Epoch [44/50], Iter [100/376], Loss: total=1.926, reg=0.881, containing_o
bj=0.635, no_obj=0.199, cls=0.210
Epoch [44/50], Iter [150/376], Loss: total=1.882, reg=0.864, containing_o
bj=0.618, no_obj=0.196, cls=0.203
Epoch [44/50], Iter [200/376], Loss: total=1.877, reg=0.873, containing_o
bj=0.612, no_obj=0.195, cls=0.197
Epoch [44/50], Iter [250/376], Loss: total=1.877, reg=0.874, containing_o
bj=0.611, no_obj=0.197, cls=0.195
Epoch [44/50], Iter [300/376], Loss: total=1.867, reg=0.872, containing_o
bj=0.609, no_obj=0.197, cls=0.190
Epoch [44/50], Iter [350/376], Loss: total=1.855, reg=0.867, containing_o
bj=0.605, no_obj=0.196, cls=0.187


Starting epoch 45 / 50
Learning Rate for this epoch: 1e-05
Epoch [45/50], Iter [50/376], Loss: total=1.874, reg=0.894, containing_ob
j=0.598, no_obj=0.200, cls=0.182
Epoch [45/50], Iter [100/376], Loss: total=1.856, reg=0.890, containing_o
bj=0.590, no_obj=0.199, cls=0.177
Epoch [45/50], Iter [150/376], Loss: total=1.943, reg=0.932, containing_o
bj=0.613, no_obj=0.200, cls=0.198
Epoch [45/50], Iter [200/376], Loss: total=1.884, reg=0.896, containing_o
bj=0.595, no_obj=0.199, cls=0.194
Epoch [45/50], Iter [250/376], Loss: total=1.869, reg=0.885, containing_o
bj=0.594, no_obj=0.198, cls=0.192
Epoch [45/50], Iter [300/376], Loss: total=1.872, reg=0.884, containing_o
bj=0.595, no_obj=0.198, cls=0.195
Epoch [45/50], Iter [350/376], Loss: total=1.876, reg=0.886, containing_o
bj=0.598, no_obj=0.198, cls=0.193
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:31<00:00, 39.68it/s]
```

```
---class aeroplane ap 0.6315298082012735---
---class bicycle ap 0.6063405225886289---
---class bird ap 0.4748701324428052---
---class boat ap 0.3379636587710013---
---class bottle ap 0.12907940871832974---
---class bus ap 0.58675829597474---
---class car ap 0.5952232554253093---
---class cat ap 0.6942820852814685---
---class chair ap 0.2875845859646111---
---class cow ap 0.4112014098300425---
---class diningtable ap 0.39412874338872167---
---class dog ap 0.5408598337314102---
---class horse ap 0.6077761749065336---
---class motorbike ap 0.4971991777340232---
---class person ap 0.4781225518822114---
---class pottedplant ap 0.16282772016168517---
---class sheep ap 0.23349715527705323---
---class sofa ap 0.4777071774878793---
---class train ap 0.7342102167256553---
---class tvmonitor ap 0.5013883516527632---
---map 0.4691275133073073---
44 [0.6315298082012735, 0.6063405225886289, 0.4748701324428052, 0.3379636
587710013, 0.12907940871832974, 0.58675829597474, 0.5952232554253093, 0.6
942820852814685, 0.2875845859646111, 0.4112014098300425, 0.39412874338872
167, 0.5408598337314102, 0.6077761749065336, 0.4971991777340232, 0.478122
5518822114, 0.16282772016168517, 0.23349715527705323, 0.4777071774878793,
0.7342102167256553, 0.5013883516527632]
```

```
Starting epoch 46 / 50
Learning Rate for this epoch: 1e-05
Epoch [46/50], Iter [50/376], Loss: total=1.770, reg=0.793, containing_ob
j=0.581, no_obj=0.200, cls=0.196
Epoch [46/50], Iter [100/376], Loss: total=1.835, reg=0.838, containing_o
bj=0.594, no_obj=0.202, cls=0.202
Epoch [46/50], Iter [150/376], Loss: total=1.898, reg=0.880, containing_o
bj=0.621, no_obj=0.198, cls=0.199
Epoch [46/50], Iter [200/376], Loss: total=1.880, reg=0.880, containing_o
bj=0.608, no_obj=0.197, cls=0.195
Epoch [46/50], Iter [250/376], Loss: total=1.837, reg=0.852, containing_o
bj=0.597, no_obj=0.200, cls=0.188
Epoch [46/50], Iter [300/376], Loss: total=1.823, reg=0.842, containing_o
bj=0.592, no_obj=0.199, cls=0.190
Epoch [46/50], Iter [350/376], Loss: total=1.825, reg=0.844, containing_o
bj=0.592, no_obj=0.198, cls=0.191


Starting epoch 47 / 50
Learning Rate for this epoch: 1e-05
Epoch [47/50], Iter [50/376], Loss: total=1.801, reg=0.813, containing_ob
j=0.587, no_obj=0.198, cls=0.203
Epoch [47/50], Iter [100/376], Loss: total=1.876, reg=0.868, containing_o
bj=0.618, no_obj=0.197, cls=0.193
Epoch [47/50], Iter [150/376], Loss: total=1.856, reg=0.858, containing_o
bj=0.606, no_obj=0.198, cls=0.193
Epoch [47/50], Iter [200/376], Loss: total=1.822, reg=0.847, containing_o
bj=0.594, no_obj=0.196, cls=0.184
Epoch [47/50], Iter [250/376], Loss: total=1.812, reg=0.842, containing_o
bj=0.591, no_obj=0.197, cls=0.182
Epoch [47/50], Iter [300/376], Loss: total=1.812, reg=0.839, containing_o
bj=0.596, no_obj=0.197, cls=0.181
Epoch [47/50], Iter [350/376], Loss: total=1.826, reg=0.848, containing_o
bj=0.596, no_obj=0.198, cls=0.184


Starting epoch 48 / 50
Learning Rate for this epoch: 1e-05
Epoch [48/50], Iter [50/376], Loss: total=1.882, reg=0.886, containing_ob
j=0.598, no_obj=0.192, cls=0.206
Epoch [48/50], Iter [100/376], Loss: total=1.858, reg=0.874, containing_o
bj=0.592, no_obj=0.192, cls=0.200
Epoch [48/50], Iter [150/376], Loss: total=1.857, reg=0.871, containing_o
bj=0.598, no_obj=0.195, cls=0.193
Epoch [48/50], Iter [200/376], Loss: total=1.871, reg=0.883, containing_o
bj=0.598, no_obj=0.196, cls=0.193
Epoch [48/50], Iter [250/376], Loss: total=1.850, reg=0.868, containing_o
bj=0.595, no_obj=0.198, cls=0.189
Epoch [48/50], Iter [300/376], Loss: total=1.849, reg=0.868, containing_o
bj=0.596, no_obj=0.198, cls=0.188
Epoch [48/50], Iter [350/376], Loss: total=1.852, reg=0.868, containing_o
bj=0.600, no_obj=0.198, cls=0.185


Starting epoch 49 / 50
Learning Rate for this epoch: 1e-05
Epoch [49/50], Iter [50/376], Loss: total=1.855, reg=0.875, containing_ob
j=0.600, no_obj=0.201, cls=0.180
Epoch [49/50], Iter [100/376], Loss: total=1.854, reg=0.873, containing_o
bj=0.605, no_obj=0.201, cls=0.175
Epoch [49/50], Iter [150/376], Loss: total=1.828, reg=0.855, containing_o
bj=0.599, no_obj=0.200, cls=0.174
```

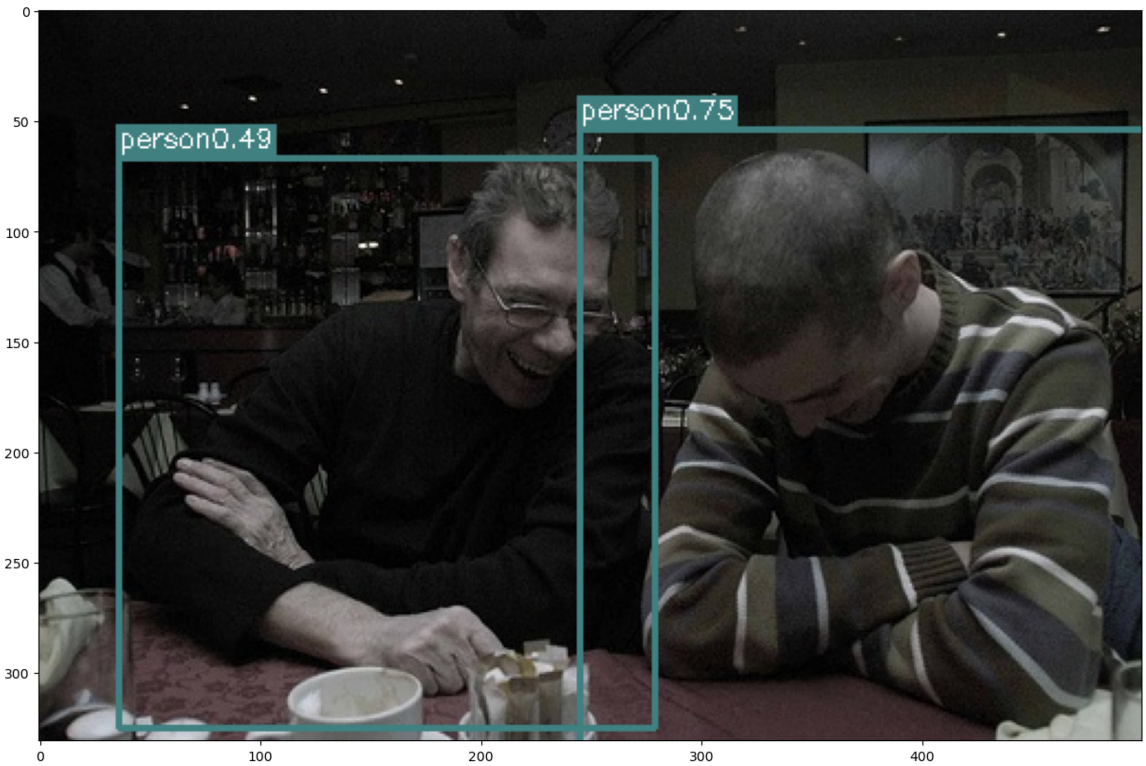
```
Epoch [49/50], Iter [200/376], Loss: total=1.796, reg=0.841, containing_obj=0.585, no_obj=0.199, cls=0.171
Epoch [49/50], Iter [250/376], Loss: total=1.796, reg=0.845, containing_obj=0.583, no_obj=0.199, cls=0.169
Epoch [49/50], Iter [300/376], Loss: total=1.801, reg=0.845, containing_obj=0.587, no_obj=0.199, cls=0.170
Epoch [49/50], Iter [350/376], Loss: total=1.814, reg=0.845, containing_obj=0.594, no_obj=0.199, cls=0.176
Updating best val loss: 2.85190


Starting epoch 50 / 50
Learning Rate for this epoch: 1e-05
Epoch [50/50], Iter [50/376], Loss: total=1.835, reg=0.874, containing_obj=0.590, no_obj=0.200, cls=0.171
Epoch [50/50], Iter [100/376], Loss: total=1.887, reg=0.887, containing_obj=0.616, no_obj=0.201, cls=0.183
Epoch [50/50], Iter [150/376], Loss: total=1.880, reg=0.880, containing_obj=0.613, no_obj=0.201, cls=0.186
Epoch [50/50], Iter [200/376], Loss: total=1.880, reg=0.880, containing_obj=0.609, no_obj=0.203, cls=0.188
Epoch [50/50], Iter [250/376], Loss: total=1.850, reg=0.860, containing_obj=0.598, no_obj=0.202, cls=0.191
Epoch [50/50], Iter [300/376], Loss: total=1.858, reg=0.871, containing_obj=0.599, no_obj=0.200, cls=0.188
Epoch [50/50], Iter [350/376], Loss: total=1.859, reg=0.872, containing_obj=0.598, no_obj=0.199, cls=0.190
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:31<00:00, 39.32it/s]
---class aeroplane ap 0.615949735601417---
---class bicycle ap 0.6157775321704524---
---class bird ap 0.4810726534511093---
---class boat ap 0.3178122370921667---
---class bottle ap 0.1078000151010163---
---class bus ap 0.5471419164722934---
---class car ap 0.5769939332811684---
---class cat ap 0.697558629482799---
---class chair ap 0.28496726010042256---
---class cow ap 0.43116441550961815---
---class diningtable ap 0.39881556890649394---
---class dog ap 0.5418315921532685---
---class horse ap 0.6021238639196907---
---class motorbike ap 0.5365404275951765---
---class person ap 0.45401377308644064---
---class pottedplant ap 0.15835678113767676---
---class sheep ap 0.22214626489230463---
---class sofa ap 0.4785335245351405---
---class train ap 0.7137208695164142---
---class tvmonitor ap 0.5034555538904202---
---map 0.46428882739477456---
49 [0.615949735601417, 0.6157775321704524, 0.4810726534511093, 0.3178122370921667, 0.1078000151010163, 0.5471419164722934, 0.5769939332811684, 0.697558629482799, 0.28496726010042256, 0.43116441550961815, 0.39881556890649394, 0.5418315921532685, 0.6021238639196907, 0.5365404275951765, 0.45401377308644064, 0.15835678113767676, 0.22214626489230463, 0.4785335245351405, 0.7137208695164142, 0.5034555538904202]
```

View example predictions

In [ ]:
```
ckpt = torch.load('checkpoints/best_detector.pth')
net = resnet50()
net.load_state_dict(ckpt)
net.to(device)
net.eval()

# select random image from val set
image_name = random.choice(val_dataset.fnames)
image = cv2.imread(os.path.join(file_root_val, image_name))
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

print('predicting...')
result = predict_image(net, image_name, root_img_directory=file_root_val)
for left_up, right_bottom, class_name, _, prob in result:
    color = COLORS[VOC_CLASSES.index(class_name)]
    cv2.rectangle(image, left_up, right_bottom, color, 2)
    label = class_name + str(round(prob, 2))
    text_size, baseline = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX
    p1 = (left_up[0], left_up[1] - text_size[1])
    cv2.rectangle(image, (p1[0] - 2 // 2, p1[1] - 2 - baseline), (p1[0] +
                  color, -1)
    cv2.putText(image, label, (p1[0], p1[1] + baseline), cv2.FONT_HERSHEY

plt.figure(figsize = (15,15))
plt.imshow(image)
```

```
predicting...
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7f7b7019b450>
```



Kaggle submission (85%)

Predict Result

Predict the results based on testing set. Upload to Kaggle.

How to upload

1. Click the folder icon in the left hand side of Colab.
2. Right click "result.csv". Select "Download"
3. To kaggle. Click "Submit Predictions"
4. Upload the result.csv
5. System will automaticlaly calculate the accuracy of 50% dataset and publish this result to leaderboard.

預測 `test` 並將結果上傳至Kaggle。**連結**

執行完畢此區的程式碼後，會將 `test` 預測完的結果存下來。

上傳流程

1. 點選左側選單最下方的資料夾圖示
2. 右鍵「result.csv」
3. 點選「Download」
4. 至連結網頁點選「Submit Predictions」
5. 將剛剛下載的檔案上傳
6. 系統會計算並公布其中50%資料的正確率

In [16]:
```
root_test = 'data/VOCdevkit_2007/VOC2007test/JPEGImages/'
file_test = 'data/voc2007test.txt'

ckpt = torch.load('checkpoints/best_detector.pth')
net = resnet50()
net.load_state_dict(ckpt)
net = net.to(device)
```

By using the test_evaluate function, you will obtain predictions for each image.

In [17]:
```
preds_submission = test_evaluate(net, test_dataset_file=file_test, img_ro
```

```
---Evaluate model on test samples---
  0%|          | 0/4950 [00:00<?, ?it/s]100%|██████████| 4950/4950 [02:03
<00:00, 40.12it/s]
```

The write_csv function will use preds_submission to write into a CSV file called 'result.csv'.

In [18]:
```
write_csv(preds_submission)
```

Report (15%)

In your report, please include:

a. A brief discussion on your implementation.

b. Report the best train and validation accuracy in all of your experiments and discuss any strategies or tricks you've employed.

c. Report the results for extra credits and also provide a discussion, if any.

Extra Credit (15%)

• Pick a fun video like **this one**, run your detector on it (a subset of frames would be OK), and produce a video showing your results.

• Try to replace the provided pre-trained network with a different one and train with the YOLO loss on top to attempt to get better accuracy.

• Or any other methods that you try to improve the performance.

In [ ]:
```
import cv2
import torch
import torchvision.transforms as transforms
from src.config import YOLO_IMG_DIM, VOC_IMG_MEAN, VOC_CLASSES, COLORS
from torch.autograd import Variable
from src.predict import decoder

net = resnet50(pretrained=False)
net.load_state_dict(torch.load('checkpoints/detector.pth'))
net.eval()
net.to(device)

# 视频读取
cap = cv2.VideoCapture('snl-digital-short-yolo-snl.mp4')
fourcc = cv2.VideoWriter_fourcc(*'mp4v')

# 获取视频的宽度和高度
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

out = cv2.VideoWriter('snl-digital-short-yolo-snl_output.mp4', fourcc, 20


while cap.isOpened():
    ret, image = cap.read() # ret,frame
    if not ret:
        break

    result = []
    frame = cv2.resize(image, (YOLO_IMG_DIM, YOLO_IMG_DIM)) # YOLO_IMG_DI
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    #img = cv2.resize(image, (YOLO_IMG_DIM, YOLO_IMG_DIM))
    h, w, _ = frame.shape
    img = cv2.resize(image, (YOLO_IMG_DIM, YOLO_IMG_DIM)) # YOLO_IMG_DIM,
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    mean = VOC_IMG_MEAN
    img = img - np.array(mean, dtype=np.float32)


    transform = transforms.Compose(
        [
            transforms.ToTensor(),
        ]
    )
    img = transform(img)
    with torch.no_grad():
        img = Variable(img[None, :, :, :])
        img = img.cuda()

        pred = net(img)  # 1xSxSx(B*5+C)
        pred = pred.cpu()
        boxes, cls_indexs, probs = decoder(pred)

        for i, box in enumerate(boxes):
            x1 = int(box[0] * w)
            x2 = int(box[2] * w)
            y1 = int(box[1] * h)
            y2 = int(box[3] * h)
            cls_index = cls_indexs[i]
            cls_index = int(cls_index)  # convert LongTensor to int
            prob = probs[i]
            prob = float(prob)
```

```
            prob = float(prob)
            result.append(
                [(x1, y1), (x2, y2), VOC_CLASSES[cls_index], prob]
            )
        for left_up, right_bottom, class_name, prob in result:
            color = COLORS[VOC_CLASSES.index(class_name)]
            cv2.rectangle(frame, left_up, right_bottom, color, 2)
            label = class_name + str(round(prob, 2))
            text_size, baseline = cv2.getTextSize(label, cv2.FONT_HERSHEY
            p1 = (left_up[0], left_up[1] - text_size[1])
            cv2.rectangle(frame, (p1[0] - 2 // 2, p1[1] - 2 - baseline),
                        color, -1)
            cv2.putText(frame, label, (p1[0], p1[1] + baseline), cv2.FONT

        out.write(frame)

cap.release()
out.release()
```

The video can be seen through the link below:

https://drive.google.com/file/d/1lc5o3AT2AwelwwkCqVZHt2gqJPFCELQT
/view?usp=sharing

try other model

In [11]:
```
# YOLO network hyperparameters
B = 2  # number of bounding box predictions per cell
S = 14  # width/height of network output grid (larger than 7x7 from paper

learning_rate = 0.001
num_epochs = 20
batch_size = 10 # 24

# Yolo loss component coefficients (as given in Yolo v1 paper)
lambda_coord = 5
lambda_noobj = 0.5

criterion = YoloLoss(S, B, lambda_coord, lambda_noobj)
optimizer = torch.optim.SGD(net.parameters(), lr=learning_rate, momentum=
```

In [12]:
```
from src.resnet_yolo import resnet101, InceptionV3
from src.mymodel import FPN,resnet50

# load_network_path = None #'checkpoints/best_detector.pth'
# pretrained = True

# # use to load a previously trained network
# if load_network_path is not None:
#     print('Loading saved network from {}'.format(load_network_path))
#     net = resnet101().to(device)
#     net.load_state_dict(torch.load(load_network_path))
# else:
#     print('Load pre-trained model')
#     net = resnet101(pretrained=pretrained).to(device)

# load_network_path = None #'checkpoints/best_detector.pth'
# pretrained = True

# use to load a previously trained network
# if load_network_path is not None:
#     print('Loading saved network from {}'.format(load_network_path))
#     net = resnet50().to(device)
#     net.load_state_dict(torch.load(load_network_path))
# else:
#     print('Load pre-trained model')
#     net = resnet50(pretrained=pretrained).to(device)


ckpt = torch.load('checkpoints/best_detector.pth')
net = resnet50()
net.load_state_dict(ckpt)
net = net.to(device)

net = FPN(net)
net = net.to(device)
```

In [13]:
```
best_val_loss = np.inf
learning_rate = 1e-3
for epoch in range(num_epochs):
    torch.cuda.empty_cache()
    net.train()

    # Update learning rate late in training
    if epoch == 30 or epoch == 40:
        learning_rate /= 10.0

    for param_group in optimizer.param_groups:
        param_group['lr'] = learning_rate

    print('\n\nStarting epoch %d / %d' % (epoch + 1, num_epochs))
    print('Learning Rate for this epoch: {}'.format(learning_rate))

    total_loss = collections.defaultdict(int)

    for i, data in enumerate(train_loader):
        data = (item.to(device) for item in data)
        images, target_boxes, target_cls, has_object_map = data
        pred = net(images)
        loss_dict = criterion(pred, target_boxes, target_cls, has_object_
        for key in loss_dict:
            total_loss[key] += loss_dict[key].item()

        optimizer.zero_grad()
        loss_dict['total_loss'].backward()
        optimizer.step()

        if (i+1) % 50 == 0:
            outstring = 'Epoch [%d/%d], Iter [%d/%d], Loss: ' % ((epoch+1
            outstring += ', '.join( "%s=%.3f" % (key[:-5], val / (i+1)) f
            print(outstring)

    # evaluate the network on the val data
    if (epoch + 1) % 5 == 0:
        val_aps = evaluate(net, val_dataset_file=annotation_file_val, img
        print(epoch, val_aps)
    with torch.no_grad():
        val_loss = 0.0
        net.eval()
        for i, data in enumerate(val_loader):
            data = (item.to(device) for item in data)
            images, target_boxes, target_cls, has_object_map = data

            pred = net(images)
            loss_dict = criterion(pred, target_boxes, target_cls, has_obj
            val_loss += loss_dict['total_loss'].item()
        val_loss /= len(val_loader)

    if best_val_loss > val_loss:
        best_val_loss = val_loss
        print('Updating best val loss: %.5f' % best_val_loss)
        torch.save(net.state_dict(),'checkpoints/FPN_best_detector.pth')

    if (epoch+1) in [5, 10, 20, 30, 40]:
        torch.save(net.state_dict(),'checkpoints/FPN_detector_epoch_%d.pt

    torch.save(net.state_dict(),'checkpoints/FPN_detector.pth')
```

```
Starting epoch 1 / 20
Learning Rate for this epoch: 0.001
Epoch [1/20], Iter [50/376], Loss: total=7.847, reg=3.150, containing_obj
=0.578, no_obj=0.060, cls=4.059
Epoch [1/20], Iter [100/376], Loss: total=7.756, reg=3.061, containing_ob
j=0.584, no_obj=0.059, cls=4.052
Epoch [1/20], Iter [150/376], Loss: total=7.745, reg=3.052, containing_ob
j=0.594, no_obj=0.058, cls=4.042
Epoch [1/20], Iter [200/376], Loss: total=7.822, reg=3.079, containing_ob
j=0.601, no_obj=0.058, cls=4.084
Epoch [1/20], Iter [250/376], Loss: total=7.934, reg=3.136, containing_ob
j=0.607, no_obj=0.058, cls=4.133
Epoch [1/20], Iter [300/376], Loss: total=7.930, reg=3.151, containing_ob
j=0.605, no_obj=0.057, cls=4.117
Epoch [1/20], Iter [350/376], Loss: total=7.895, reg=3.133, containing_ob
j=0.599, no_obj=0.057, cls=4.106
Updating best val loss: 8.60175


Starting epoch 2 / 20
Learning Rate for this epoch: 0.001
Epoch [2/20], Iter [50/376], Loss: total=7.813, reg=3.137, containing_obj
=0.588, no_obj=0.056, cls=4.032
Epoch [2/20], Iter [100/376], Loss: total=8.094, reg=3.237, containing_ob
j=0.606, no_obj=0.055, cls=4.197
Epoch [2/20], Iter [150/376], Loss: total=7.924, reg=3.174, containing_ob
j=0.602, no_obj=0.056, cls=4.092
Epoch [2/20], Iter [200/376], Loss: total=7.990, reg=3.194, containing_ob
j=0.603, no_obj=0.056, cls=4.136
Epoch [2/20], Iter [250/376], Loss: total=7.919, reg=3.158, containing_ob
j=0.603, no_obj=0.056, cls=4.102
Epoch [2/20], Iter [300/376], Loss: total=8.018, reg=3.193, containing_ob
j=0.606, no_obj=0.056, cls=4.163
Epoch [2/20], Iter [350/376], Loss: total=8.025, reg=3.204, containing_ob
j=0.604, no_obj=0.056, cls=4.161


Starting epoch 3 / 20
Learning Rate for this epoch: 0.001
Epoch [3/20], Iter [50/376], Loss: total=7.754, reg=3.037, containing_obj
=0.603, no_obj=0.056, cls=4.057
Epoch [3/20], Iter [100/376], Loss: total=8.046, reg=3.142, containing_ob
j=0.623, no_obj=0.056, cls=4.225
Epoch [3/20], Iter [150/376], Loss: total=8.239, reg=3.293, containing_ob
j=0.616, no_obj=0.056, cls=4.274
Epoch [3/20], Iter [200/376], Loss: total=8.253, reg=3.301, containing_ob
j=0.618, no_obj=0.056, cls=4.278
Epoch [3/20], Iter [250/376], Loss: total=8.143, reg=3.263, containing_ob
j=0.612, no_obj=0.056, cls=4.211
Epoch [3/20], Iter [300/376], Loss: total=8.128, reg=3.277, containing_ob
j=0.608, no_obj=0.056, cls=4.187
Epoch [3/20], Iter [350/376], Loss: total=8.027, reg=3.229, containing_ob
j=0.605, no_obj=0.056, cls=4.137
Updating best val loss: 8.59608


Starting epoch 4 / 20
Learning Rate for this epoch: 0.001
Epoch [4/20], Iter [50/376], Loss: total=8.350, reg=3.364, containing_obj
=0.602, no_obj=0.056, cls=4.327
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call las
/home/vllab/Desktop/A5/A5.ipynb Cell 44 line 2
     <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=18'>19</a> for i, data in enumerate(train_loader):
     <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=19'>20</a>     data = (item.to(device) for item in data)
---> <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=20'>21</a>     images, target_boxes, target_cls, has_object_
ap = data
     <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=21'>22</a>     pred = net(images)
     <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=22'>23</a>     loss_dict = criterion(pred, target_boxes, ta
et_cls, has_object_map)

/home/vllab/Desktop/A5/A5.ipynb Cell 44 line 2
     <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=16'>17</a> total_loss = collections.defaultdict(int)
     <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=18'>19</a> for i, data in enumerate(train_loader):
---> <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=19'>20</a>     data = (item.to(device) for item in data)
     <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=20'>21</a>     images, target_boxes, target_cls, has_object_
ap = data
     <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X61sZm
ZQ%3D%3D?line=21'>22</a>     pred = net(images)

KeyboardInterrupt:
```

In [14]:

```
net.eval()

# select random image from val set
image_name = random.choice(val_dataset.fnames)
image = cv2.imread(os.path.join(file_root_val, image_name))
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

print('predicting...')
result = predict_image(net, image_name, root_img_directory=file_root_val)
for left_up, right_bottom, class_name, _, prob in result:
    color = COLORS[VOC_CLASSES.index(class_name)]
    cv2.rectangle(image, left_up, right_bottom, color, 2)
    label = class_name + str(round(prob, 2))
    text_size, baseline = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX
    p1 = (left_up[0], left_up[1] - text_size[1])
    cv2.rectangle(image, (p1[0] - 2 // 2, p1[1] - 2 - baseline), (p1[0] +
                  color, -1)
    cv2.putText(image, label, (p1[0], p1[1] + baseline), cv2.FONT_HERSHEY

plt.figure(figsize = (15,15))
plt.imshow(image)
```

```
predicting...
```

Out[14]:

```
<matplotlib.image.AxesImage at 0x7f19022ad650>
```

In [15]:
```
root_test = 'data/VOCdevkit_2007/VOC2007test/JPEGImages/'
file_test = 'data/voc2007test.txt'

ckpt = torch.load('checkpoints/FPN_best_detector.pth')
net = resnet50(pretrained=True).to(device)

net = FPN(net)
net.load_state_dict(ckpt)
net = net.to(device)

preds_submission = test_evaluate(net, test_dataset_file=file_test, img_ro
write_csv(preds_submission)
```

```
---Evaluate model on test samples---
 28%|███       | 1381/4950 [03:06<08:00,  7.42it/s]
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call las
/home/vllab/Desktop/A5/A5.ipynb Cell 46 line 1
      <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X63sZ
sZQ%3D%3D?line=7'>8</a> net.load_state_dict(ckpt)
      <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X63sZ
sZQ%3D%3D?line=8'>9</a> net = net.to(device)
---> <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X63sZm
ZQ%3D%3D?line=10'>11</a> preds_submission = test_evaluate(net, test_datas
_file=file_test, img_root=root_test)
     <a href='vscode-notebook-cell:/home/vllab/Desktop/A5/A5.ipynb#X63sZm
ZQ%3D%3D?line=11'>12</a> write_csv(preds_submission)

File ~/Desktop/A5/src/eval_voc.py:207, in test_evaluate(model, test_datas
_file, img_root, test_loader)
    205 model.eval()
    206 for image_path in tqdm(image_list):
--> 207     result = predict_image(model, image_path, root_img_directory=
g_root)
    208     for (
    209         (x1, y1),
    210         (x2, y2),
   (...)
    213         prob,
    214     ) in result:  # image_id is actually image_path
    215         preds[class_name].append([image_id, prob, x1, y1, x2, y2]

File ~/Desktop/A5/src/predict.py:144, in predict_image(model, image_name,
oot_img_directory)
    141 img = img.cuda()
    143 pred = model(img)  # 1xSxSx(B*5+C)
--> 144 pred = pred.cpu()
    145 boxes, cls_indexs, probs = decoder(pred)
    147 for i, box in enumerate(boxes):

KeyboardInterrupt:
```