2018 Big Data Ecosystems

Programming Assignment 1 - CV

Object detection has been an important problem in computer vision. In this assignment, we will study the Faster R-CNN algorithm which is a decent work done by Shaoqing Ren, Kaiming He, and Jian Sun. The research paper is titled “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal”, and is archived at https://arxiv.org/abs/1506.01497. Generally, faster R-CNN takes the fast R-CNN neural network but replace the selective search part by a region proposal network (RPN). Two neural network are used by fast R-CNN. One is ZF, the other is VGG-16. The RPN and fast R-CNN is unified by sharing the same convolutional layers of either ZF or VGG-16.

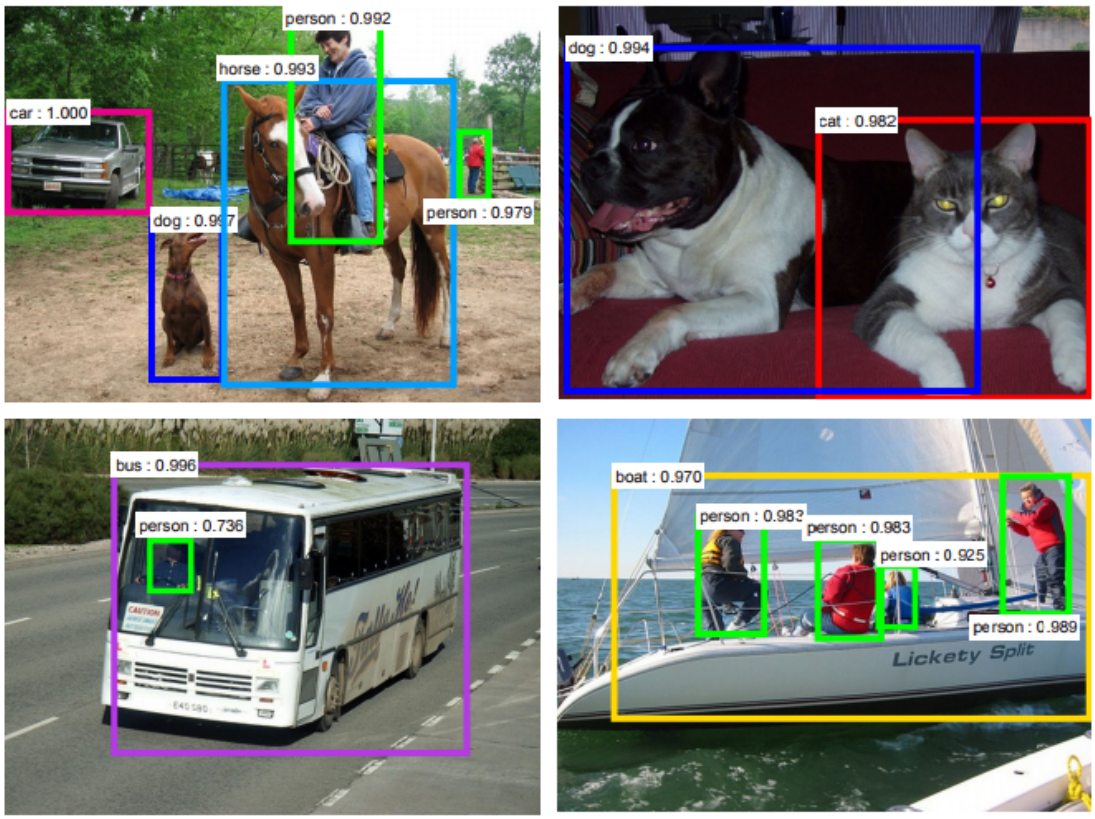

Figure 1 Detection results of faster R-CNN

Figure 1 shows the detection results of faster R-CNN. Faster R-CNN is able to find multiple different objects in same image. It draws a compact box around the object with label shown the class of the detected object and the score tells how much the object belongs to that class.

Q.1: do object detection using pretrained model on five of your own images

1.1 In the first part, you have to setup the py-faster-rcnn. The py-faster-rcnn code could be download from github: https://github.com/rbgirshick/py-faster-rcnn. Follow the readme carefully to install it. If you are working without an GPU, you need to install the code in CPU only mode. The following link could be used as references to install it in an Ubuntu system computer.

https://chunml.github.io/ChunML.github.io/project/Running-Faster-RCNN-Ubuntu/. During the installation, you could encounter some errors. For those unexpected error, google the error and you should be able to find a solution.

1.2 Next, download the pretrained Faster R-CNN models
Download the pretrained model using the fetch_faster_Rcnn_models.sh

```
cd $FRCN_ROOT
./data/scripts/fetch_faster_rcnn_models.sh
```

1.3 Get 5 of your own JPGE images, run the pretrained faster-RCNN model on your image, report the detection results with bounding box, label and score.
You are suggested to start from the file: “demo.py” in tools folder. The faster-RCNN is trained by PASCAL VOC 2007 dataset. It is able to detection 20 class objects including: aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train, tv monitor plus with background.
The detection results should return a label belongs to one of this 21 classes with the score.
The score is

$$p_{nk} = \exp(x_{nk}) / \sum_{k'} \exp(x_{nk'})$$

where p_{nk} is the score that the nth object is in the kth class.

Q.2: Fine Tuning the ZF network of Faster R-CNN by Alternating training do detection on your own image used in Q.1

You are suggested to start from the file: “tools/train_faster_rcnn_alt_opt.py”. In the “train_faster_rcnn_alt_opt.py”, several functions: “get_solvers”,”train_rpn”, “rpn_generate”, and “train_fast_rcnn” are called in main function to set the training parameters. The files related to set most of the training parameters are in the “/models” folder. Alternating training will be used to fine tune the Faster R-CNN. For both RPN, Fast R-CNN, during training, a multi-task loss objective function is minimized:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i [p_i^* \geq 1] L_{reg}(t_i, t_i^*)$$

The first task loss is the loss of classification which is the loss of ith anchor for true class k

$$L_{cls}(p_i, p_i^*) = -\log(p_{ik})$$

The second loss is the loss of bounding box regression

$$L_{reg}(t_i, t_i^*) = \sum_{\{x,y,w,h\}} smooth_{L_1}(t_i - v_i)$$

in which

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

Where t_i is predicted bounding box position for the ith anchor, v_i is the ground truth position of the ith anchor. ForL_{reg}, we only compute on anchor which are not background.
The training procedure is:
step 1: load mageNet-pretrained Fast R-CNN model to initialize the parameters

step 2: train the RPN network
step 3: generate proposals by the step-2 RPN.
step 4: use the proposals to train the Fast R-CNN network. The Fast R-CNN network is also initialize by the mageNet-pretrained Fast R-CNN model.
step 5: use the Fast R-CNN detector network to initialize RPN training, but fixed the shared convolutional layers of Fast R-CNN and RPN, only fine tune the RPN layers unique to Fast R-CNN
step 6: fine tune the unique layers of Fast R-CNN by fixing the shared convolutional layers of Fast R-CNN and RPN, only fine tune the RPN Fast R-CNN layers unique to Fast R-CNN.

2.1 Prepare dataset and pretrained model and create symlinks for the PASCAL VOC dataset
For Ubuntu system:

```
wget http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval_06-Nov-2007.tar
wget http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtest_06-Nov-2007.tar
wget http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCdevkit_08-Jun-2007.tar
```

```
tar xvf VOCtrainval_06-Nov-2007.tar
tar xvf VOCtest_06-Nov-2007.tar
tar xvf VOCdevkit_08-Jun-2007.tar
```

```
cd $FRCN_ROOT/data
ln -s $VOCdevkit VOCdevkit2007
```

2.2 Download pre-trained models

```
cd $FRCN_ROOT
./data/scripts/fetch_imagenet_models.sh
```

2.3 Train and test a Faster R-CNN detector suing the alternating optimization algorithm
For this step, definitely, you need a computer with GPU. The training of py-faster-rcnn code is implemented in GPU mode.

```
cd $FRCN_ROOT
./experiments/scripts/faster_rcnn_alt_opt.sh [GPU_ID] [NET] [--set ...]
# GPU_ID is the GPU you want to train on
# NET in {ZF, VGG_CNN_M_1024, VGG16} is the network arch to use
# --set ... allows you to specify fast_rcnn.config options, e.g.
#   --set EXP_DIR seed_rng1701 RNG_SEED 1701
```

py-faster-rcnn only provide GPU mode for “roi_pooling_layer” and “smooth_L1_loss_layer”.
In order to wok in CPU mode:
2.3.1 replace this two files in “py-faster-rcnn/caffe-fast-rcnn/src/caffe/layers/” by files https://github.com/neuleaf/faster-rcnn-cpu
2.3.2 modify the “/tools/train_faster_rcnn_alt_opt.py”
Comment the 34-36 that set GPU mode to train

```
34      #parser.add_argument('--gpu', dest='gpu_id',
35      #                    help='GPU device id to use [0]',
36      #                    default=0, type=int)
```

set caffe to CPU mode, change caffe.set_mode_gpu to caffe.set_mode_cpu()

```
101     # set up caffe
102     #caffe.set_mode_gpu()
103     #caffe.set_device(cfg.GPU_ID)
104     caffe.set_mode_cpu()
```

comment the code about “gpu_id” around 213

```
213     #cfg.GPU_ID = args.gpu_id
```

2.3.3 modify the “/experiments/scripts/faster_rcnn_alt_opt.sh”

Change the 46 line to

```
46  #time ./tools/train_faster_rcnn_alt_opt.py \
47  time cd /<PATH>/tools/
48  python train_faster_rcnn_alt_opt.py --net_name ${NET} \
```

where path is the “absolute path” of “tools”

change the 48,50 to the “absolute path of the file”, also comment the part about GPU_ID like ”—gpu &{GPU_ID}”

```
46  time ./tools/train_faster_rcnn_alt_opt.py --gpu ${GPU_ID} \
47    --net_name ${NET} \
48    --weights data/imagenet_models/${NET}.v2.caffemodel \
49    --imdb ${TRAIN_IMDB} \
50    --cfg experiments/cfgs/faster_rcnn_alt_opt.yml \
51    ${EXTRA_ARGS}
```

comment the 57 to 62 lines

```
57  time ./tools/test_net.py --gpu ${GPU_ID} \
58    --def models/${PT_DIR}/${NET}/faster_rcnn_alt_opt/faster_rcnn_test.pt \
59    --net ${NET_FINAL} \
60    --imdb ${TEST_IMDB} \
61    --cfg experiments/cfgs/faster_rcnn_alt_opt.yml \
62    ${EXTRA_ARGS}
```

2.3.4 Then you should be good to train the Faster R-CNN by doing

```
cd $FRCN_ROOT
./experiments/scripts/faster_rcnn_alt_opt.sh [GPU_ID] [NET] [--set ...]
# GPU_ID is the GPU you want to train on
# NET in {ZF, VGG_CNN_M_1024, VGG16} is the network arch to use
# --set ... allows you to specify fast_rcnn.config options, e.g.
#   --set EXP_DIR seed_rng1701 RNG_SEED 1701
```

2.3.5 If you encounter some errors related to solver, change the “train_net” path in the .pt files in folder “/models/pascal_voc/ZF/faster_rcnn_alt_opt” to “absolute path”. For other unexpected errors, google it, you should find a solution.

2.4 The fine tune model will be saved in “/output” folder

2.5 Use the fine tune model to do detection on your own image used in Q.1 and report the detection results: with bounding box, label and score. Copy your model in 2.4 to “py-faster-rcnn\data\faster_rcnn_models” where is the “demo.py” saved the pretrained model. Then modify the “demo.py” to test on your own image.

Q.3: Fine Tuning the ZF network of Faster R-CNN by Alternating training on Basketball

dataset. Do detection on five of your own basketball images.

In Q.2, you should have already install and run training with no errors. Now you are good to start training your own basketball dataset. You can download the dataset from:

https://www.dropbox.com/s/iywkgsrx2fx6t5q/basketball.tar.gz?dl=0

There are about 900 labelled basketball images from ImageNet (DISCLAIMER: This dataset should be only used for non-commercial research activities. Please follow the ImageNet rules about the use of the dataset).

Figure 1 basketball image

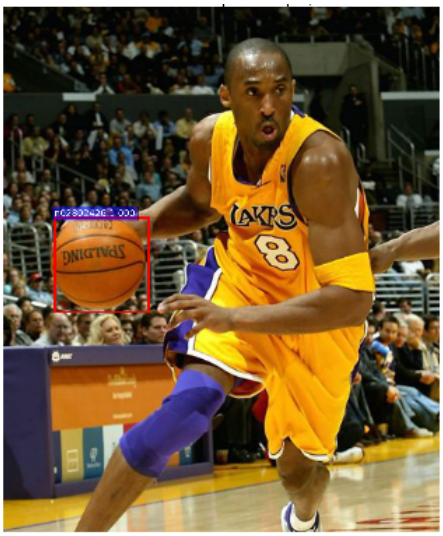

Figure 2 Detection results

3.1 Load dataset

In py-faster-rcnn, the files about load dataset are in "/lib/datasets/". "/datasets/pascal_voc.py", "/datasets/imdb.py", "/datasets/factory.py" is used to read data by py-faster-rcnn from dataset. You need to write you own code to read data from basketball dataset or modify these files

For students who want to modify files

3.1.1 please replace data in "/data/VOCdevkit2007/VOC2007" by your download basketball data.

3.1.2 modify "/datasets/pascal_voc.py"

Change number of classes in dataset, and "n02802426" is the class label of basketball. Also change the extension of image to ".JPEG"

```
30        self._classes = ('__background__', # always index 0
31                         'n02802426')
32        self._class_to_ind = dict(zip(self.classes, xrange(self.num_classes)))
33        self._image_ext = '.JPEG'
```

3.1.3 "/datasets/imdb.py"

Change line 104

```
103        num_images = self.num_images
104        widths = [PIL.Image.open(self.image_path_at(i)).size[0] for i in xrange(num_images)]
```

In case there is AssetionError, change the code

```
111             for b in range(len(boxes)):
112                 if boxes[b][2]<boxes[b][0]:
113                     boxes[b][0] = 0
114 
115             assert (boxes[:, 2] >= boxes[:, 0]).all()
```

3.1.5 “/datasets/factory.py”

Change line 18

```
18     for split in ['train', 'val', 'trainval']:
19         name = 'voc_{}_{}'.format(year, split)
20         __sets[name] = (lambda split=split, year=year: pascal_voc(split, year))
21 
```

3.2 Change network structure

The py-faster-rcnn is used for training 21 class objects. Now, we will change the network structure to train 2 class. One is background, the other is basketball.

3.2.1 change file “\models\pascal_voc\ZF\faster_rcnn_alt_opt\stage1_fast_rcnn_train.pt”

```
 2 layer {
 3   name: 'data'
 4   type: 'Python'
 5   top: 'data'
 6   top: 'rois'
 7   top: 'labels'
 8   top: 'bbox_targets'
 9   top: 'bbox_inside_weights'
10   top: 'bbox_outside_weights'
11   python_param {
12     module: 'roi_data_layer.layer'
13     layer: 'RoIDataLayer'
14     param_str: "'num_classes': 2"
15   }
16 }
```

```
layer {
  name: "cls_score"
  type: "InnerProduct"
  bottom: "fc7"
  top: "cls_score"
  param { lr_mult: 1.0 }
  param { lr_mult: 2.0 }
  inner_product_param {
    num_output: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

```
layer {
  name: "bbox_pred"
  type: "InnerProduct"
  bottom: "fc7"
  top: "bbox_pred"
  param { lr_mult: 1.0 }
  param { lr_mult: 2.0 }
  inner_product_param {
    num_output: 8
    weight_filler {
      type: "gaussian"
      std: 0.001
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

3.2.2 change file “\models\pascal_voc\ZF\faster_rcnn_alt_opt\stage1_rpn_train.pt”

```
  name: 'input-data'
  type: 'Python'
  top: 'data'
  top: 'im_info'
  top: 'gt_boxes'
  python_param {
    module: 'roi_data_layer.layer'
    layer: 'RoIDataLayer'
    param_str: "'num_classes': 2"
  }
}
```

3.2.3 change file “\models\pascal_voc\ZF\faster_rcnn_alt_opt\stage2_fast_rcnn_train.pt”

```
name: "ZF"
layer {
  name: 'data'
  type: 'Python'
  top: 'data'
  top: 'rois'
  top: 'labels'
  top: 'bbox_targets'
  top: 'bbox_inside_weights'
  top: 'bbox_outside_weights'
  python_param {
    module: 'roi_data_layer.layer'
    layer: 'RoIDataLayer'
    param_str: "'num_classes': 2"
  }
}

```

```
layer {
  name: "cls_score"
  type: "InnerProduct"
  bottom: "fc7"
  top: "cls_score"
  param { lr_mult: 1.0 }
  param { lr_mult: 2.0 }
  inner_product_param {
    num_output: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

```
layer {
  name: "bbox_pred"
  type: "InnerProduct"
  bottom: "fc7"
  top: "bbox_pred"
  param { lr_mult: 1.0 }
  param { lr_mult: 2.0 }
  inner_product_param {
    num_output: 8
    weight_filler {
      type: "gaussian"
      std: 0.001
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

3.2.4 change file “\models\pascal_voc\ZF\faster_rcnn_alt_opt\stage2_rpn_train.pt”

```
name: "ZF"
layer {
  name: 'input-data'
  type: 'Python'
  top: 'data'
  top: 'im_info'
  top: 'gt_boxes'
  python_param {
    module: 'roi_data_layer.layer'
    layer: 'RoIDataLayer'
    param_str: "'num_classes': 2"
  }
}
```

3.2.5 change file “\models\pascal_voc\ZF\faster_rcnn_alt_opt\faster_rcnn_test.pt”

```
layer {
  name: "cls_score"
  type: "InnerProduct"
  bottom: "fc7"
  top: "cls_score"
  inner_product_param {
    num_output: 2
  }
}
```

```
layer {
  name: "bbox_pred"
  type: "InnerProduct"
  bottom: "fc7"
  top: "bbox_pred"
  inner_product_param {
    num_output: 8
  }
}
```

3.3 Now you are good to start training

In command line: ./experiemtns/scripts/faster_Rcnn_alt_opt.sh 0 ZF pascal_voc

3.4 The trained model is saved in “/output/faster_rcnn_alt_opt/voc_2007_trainval/”, modify the “/tools/demo.py” file to do detection on your own five basketball image and report the detection results: with bounding box, label and score.