Faster-rcnn程序详解

第0章 软件版本及硬件型号

0.1 软件版本

Ubuntu14.04、CUDA8、CUDNN5.1 for CUDA8、opencv3.1、python版本faster-rcnn、GPU版本、faster训练方式为alt-opt

0.2 硬件型号

I7四代CPU、内存4G、固态硬盘250G、影驰GTX1070

第1章 py-faster-rcnn配置运行faster\_rcnn\_end2end—VGG\_CNN\_M\_1024

1.1 下载数据test data and VOCdevkit

下载原网址：<http://www.cs.berkeley.edu/~rbg/fast-rcnn-data>

或者终端输入：

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

1.2 解压生成文件夹 VOCdevkit

1. cd到data目录下

cd /home/jc/py-faster-rcnn/data

2. 就在data目录下输入下列命令解压，解压文件会自动跑到devkit这个文件夹，其中VOC2007会自动包含trainval和test这两个文件，即终端输入：

tar xvf VOCtrainval\_06-Nov-2007.tar

tar xvf VOCtest\_06-Nov-2007.tar

tar xvf VOCdevkit\_08-Jun-2007.tar

1.3 文件夹包含目录

$VOCdevkit/ #development kit

$VOCdevkit/VOCcode/ #VOC utility code

$VOCdevkit/VOC2007 #image sets, annotations, etc.

1.4 建立链接

cd /home/jc/py-faster-rcnn/data

sudo ln -s VOCdevkit VOCdevkit2007

1.5 下载pre-trained ImageNet models

下载地址：<https://people.eecs.berkeley.edu/~rbg/faster-rcnn-data/>

ubuntu下终端下载程序：

./data/scripts/fetch\_imagenet\_models.sh

1.6 训练

1. cd /home/jc/py-faster-rcnn

2. 执行：

./experiments/scripts/faster\_rcnn\_end2end.sh 0 VGG\_CNN\_M\_1024 pascal\_voc --set EXP\_DIR foobar RNG\_SEED 42 TRAIN.SCALES "[400,500,600,700]"

执行命令解释：

# ./experiments/scripts/faster\_rcnn\_end2end.sh GPU NET DATASET [options args to {train,test}\_net.py]

# DATASET is either pascal\_voc or coco.

#

# Example:

# ./experiments/scripts/faster\_rcnn\_end2end.sh 0 VGG\_CNN\_M\_1024 pascal\_voc \

# --set EXP\_DIR foobar RNG\_SEED 42 TRAIN.SCALES "[400, 500, 600, 700]"

3. 结果显示：

在py-faster-rcnn/output目录下生成foobar文件夹，其中包含voc\_2007\_test和voc\_2007\_trainval两个文件夹。

第2章 运用labelimg制作训练数据(win10版本)

我这里安装的是windows版本，linux版本安装更加容易，参考官网安装即可：

<https://github.com/tzutalin/labelImg>

2.1 安装Anaconda

1. 这里选择安装Anaconda，它是一款十分好用的python集成安装环境

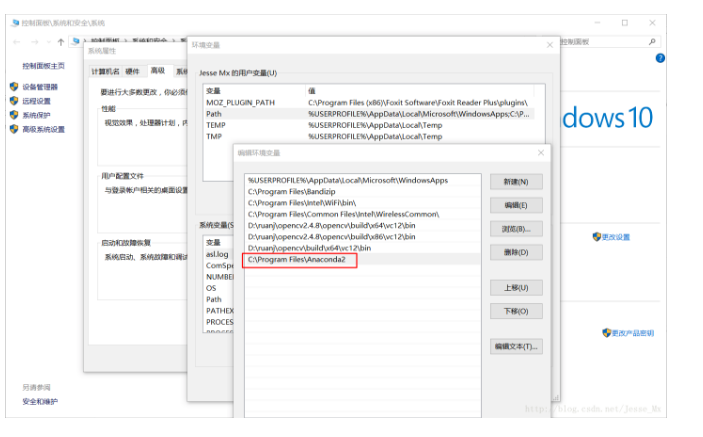
2. 首先从官网下载最新的版本，这里我下载的是64位的python2.7版本

下载地址：<https://www.continuum.io/downloads>

3. 然后安装到系统默认位置，我的位置是：C:\ProgramData\Anaconda2

4. 安装成功后建议在环境变量path里面增加一条“C:\Program Files\Anaconda2

点击“path”，“编辑”，在原来的基础上增加“;C:\Program Files\Anaconda2”，注意有“;”（应为分号）



2.2 安装PyQt包

1. 打开命令行窗口，输入conda list ，会列出所有预装的python扩展包，可以看到里面已经有了lxml，但是缺少PyQt

2. 以管理员身份打开cmd（否则会出错），在“开始”点击右键，选择“命令提示符(管理员)”

3. 命令行输入：conda install pyqt=4

2.3 安装labelimg

1. 在原作者的github下载源码压缩包，点击“clone or download”选择“download .zip”，解压可得到名为labelImg-master的文件夹

Github下载地址：<https://github.com/tzutalin/labelImg>

2. 把该文件夹放到c盘目录下（文件在d盘下cd需要两步参见备注），在cmd中输入：

cd C:\labelImg-master（进入该文件夹目录）

3. 依次输入下面语句即可打开软件：

pyrcc4 -o resources.py resources.qrc

python labelImg.py

备注：

2.4 使用方法

1. 修改默认的XML文件保存位置：

使用快捷键“Ctrl+R”，改为自定义位置，这里的路径一定不能包含中文，否则无法保存

2. 源码文件夹中使用notepad++打开data/predefined\_classes.txt，修改默认类别，比如改成person、car、motorcycle三个类别

3. “Open Dir”打开图片文件夹（注意：图片文件夹路径不能有中文），选择第一张图片开始进行标注，使用“Create RectBox”或者“w”开始画框，选择类别及命名

完成一张图片后点击“Save”保存，此时XML文件已经保存到本地了，点击“Next Image”转到下一张图片

2.5 labelimg产生的xml与要求的有区别

labelimg产生的xml与要求的有区别，但是经过测试，直接把该xml文件输入进去可以直接运行程序，不需要更改xml格式

2.6 参考网址

1. <https://github.com/tzutalin/labelImg>

2. <http://blog.csdn.net/jesse_mx/article/details/53606897>

第3章 win10下更改labelimg代码

3.1 运行Jupyter NoteBook

1. 点击Jupyter NoteBook打开，在cmd中点击回车，跳转到浏览器中

2. New->python2

3. 在Ln中输入：

%cd C:\labelImg-master #转到目录，“Shift+Enter”运行

%load labelImg.py #载入程序，“Shift+Enter”运行

3.2 参考网址

1. <http://blog.csdn.net/tina_ttl/article/details/51031113>

第4章 Faster-rcnn代码更改过程

4.1 VOC2007数据集介绍及制作自己的数据集

参考VOC2007的数据集格式，主要包括三个部分:

JPEGImages：存放你用来训练的原始图像

Annotations：存放原始图像中的Object的坐标信息，XML格式

ImageSets/Main：指定用来train，trainval，val和test的图片的编号

1. JPEGImages

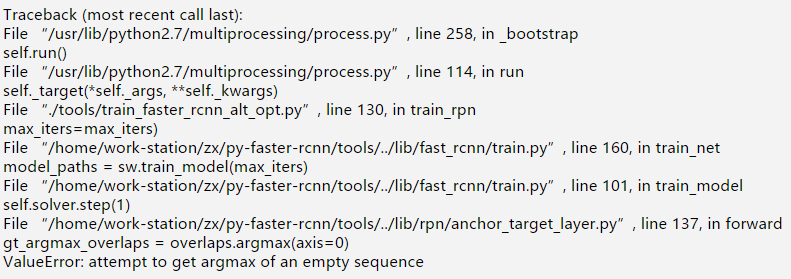
替换自己的图片时需要注意以下几点：

a. 编号要以6为数字命名，例如000034.jpg

b. 图片要是JPEG/JPG格式的，PNG之类的需要自己转换下

c. 图片的长宽比(width/height)要在0.462-6.828之间，就是太过瘦长的图片不要

如果长宽比过大或过小会出现如下问题：



2. Annotations

faster rcnn训练需要图像的bounding box信息作为监督(ground truth)，所以你需要将你的所有可能的object使用框标注，并写上坐标，最终是一个XML格式的文件，一个训练图片对应Annotations下的一个同名的XML文件

使用第二章的labelimg工具制作的xml文件格式和voc2007的格式稍有不同，但可以直接输入训练，不需要更改代码

3. Imagesets/Main

a. 因为VOC的数据集可以做很多的CV任务，比如Object detection, Semantic segementation, Edge detection等，所以Imageset下有几个子文件夹(Layout, Main, Segementation)，我们只要修改下Main下的文件就可以了(train.txt, trainval.txt, val.txt, test.txt)，里面写上你想要进行任务的图片的编号

b. 将上述你的数据集放在py-faster-rcnn/data/VOCdevkit2007/VOC2007下面，替换原始VOC2007的JPEGIMages，Imagesets，Annotations

c. train.txt, trainval.txt, val.txt, test.txt中的数字标号可以不按照顺序随机排列

4.2 faster-rcnn工作目录及训练方式介绍

1. 工作目录介绍：

caffe-fast-rcnn —> caffe框架

data —> 存放数据，以及读取文件的cache

experiments —>存放配置文件以及运行的log文件,配置文件

lib —> python接口

models —> 三种模型, ZF(S)/VGG1024(M)/VGG16(L)

output —> 输出的model存放的位置，不训练此文件夹没有

tools —> 训练和测试的python文件

2. 训练方式介绍：

Alternative training(alt-opt)

Approximate joint training(end-to-end)

a. 推荐使用第二种，因为第二种使用的显存更小，而且训练会更快，同时准确率差不多，两种方式需要修改的代码是不一样的

b. 同时faster rcnn提供了三种训练模型，小型的ZFmodel，中型的VGG\_CNN\_M\_1024和大型的VGG16，论文中说VGG16效果比其他两个好，但是同时占用更大的GPU显存(~11GB)

c. 我使用的是VGG\_CNN\_M\_1024 model + alternative training，需要检测的类别只有一类，加上背景所以总共是两类(background+captcha)

4.3 代码修改

1. py-faster-rcnn/models/pascal\_voc/VGG\_CNN\_M\_1024/faster\_rcnn\_alt\_opt/stage1\_fast\_rcnn\_train.pt

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" #按训练集类别改，该值为类别数+1

}

}

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 #按训练集类别改，该值为类别数+1

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 #按训练集类别改，该值为（类别数+1）\*4，四个顶点坐标

weight\_filler {

type: "gaussian"

std: 0.001

}

bias\_filler {

type: "constant"

value: 0

}

}

}

2. py-faster-rcnn/models/pascal\_voc/ VGG\_CNN\_M\_1024/faster\_rcnn\_alt\_opt/stage1\_rpn\_train.pt

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" #按训练集类别改，该值为类别数+1

}

}

3. py-faster-rcnn/models/pascal\_voc/ VGG\_CNN\_M\_1024/faster\_rcnn\_alt\_opt/stage2\_fast\_rcnn\_train.pt

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" #按训练集类别改，该值为类别数+1

}

}

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 #按训练集类别改，该值为类别数+1

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 #按训练集类别改，该值为（类别数+1）\*4,四个顶点坐标

weight\_filler {

type: "gaussian"

std: 0.001

}

bias\_filler {

type: "constant"

value: 0

}

}

}

4. py-faster-rcnn/models/pascal\_voc/ VGG\_CNN\_M\_1024/faster\_rcnn\_alt\_opt/stage2\_rpn\_train.pt

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" #按训练集类别改，该值为类别数+1

}

}

5. py-faster-rcnn/models/pascal\_voc/ VGG\_CNN\_M\_1024/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 #按训练集类别改，该值为类别数+1

}

}

layer {

name: "bbox\_pred"

type: "InnerProduct"

bottom: "fc7"

top: " bbox\_pred "

inner\_product\_param {

num\_output: 8 #按训练集类别改，该值为（类别数+1）\*4,四个顶点坐标

}

}

6. py-faster-rcnn/lib/datasets/pascal\_voc.py

class pascal\_voc(imdb):

def \_\_init\_\_(self, image\_set, year, devkit\_path=None):

imdb.\_\_init\_\_(self, 'voc\_' + year + '\_' + image\_set)

self.\_year = year

self.\_image\_set = image\_set

self.\_devkit\_path = self.\_get\_default\_path() if devkit\_path is None \

else devkit\_path

self.\_data\_path = os.path.join(self.\_devkit\_path, 'VOC' + self.\_year)

self.\_classes = ('\_\_background\_\_', # always index 0

captcha' # 有几个类别此处就写几个，我是两个

)

如果你的标签含有大写字母，可能会出现KeyError的错误，所以建议全部使用小写字母

7. py-faster-rcnn/lib/datasets/imdb.py（我测试过这步不改仍可以运行）

将append\_flipped\_images函数改为如下形式:

def append\_flipped\_images(self):

num\_images = self.num\_images

widths = [PIL.Image.open(self.image\_path\_at(i)).size[0]

for i in xrange(num\_images)]

for i in xrange(num\_images):

boxes = self.roidb[i]['boxes'].copy()

oldx1 = boxes[:, 0].copy()

oldx2 = boxes[:, 2].copy()

boxes[:, 0] = widths[i] - oldx2 - 1

print boxes[:, 0]

boxes[:, 2] = widths[i] - oldx1 - 1

print boxes[:, 0]

assert (boxes[:, 2] >= boxes[:, 0]).all()

entry = {'boxes' : boxes,

'gt\_overlaps' : self.roidb[i]['gt\_overlaps'],

'gt\_classes' : self.roidb[i]['gt\_classes'],

'flipped' : True}

self.roidb.append(entry)

self.\_image\_index = self.\_image\_index \* 2

4.4 训练

训练前需要注意几个问题：

1. cache问题

假如你之前训练了官方的VOC2007的数据集或其他的数据集，是会产生cache的问题的，需要在重新训练新的数据之前将其删除（否则会出错），删除一下两个文件：

(1) py-faster-rcnn/output

(2) py-faster-rcnn/data/cache

2 训练参数

py-faster-rcnn/models/pascal\_voc/ VGG\_CNN\_M\_1024/faster\_rcnn\_alt\_opt/stage\_fast\_rcnn\_solver\*.pt

base\_lr: 0.001

lr\_policy: 'step'

step\_size: 30000

display: 20

....

迭代次数在文件py-faster-rcnn/tools/train\_faster\_rcnn\_alt\_opt.py中进行修改

max\_iters = [80000, 40000, 80000, 40000]

分别对应rpn第1阶段，fast rcnn第1阶段，rpn第2阶段，fast rcnn第2阶段的迭代次数，自己修改即可，不过注意这里的值不要小于上面的solver里面的step\_size的大小

3. 在eclipse中输入参数

进入Run Configurations->Arguments，在Program arguments中输入：

--gpu 0

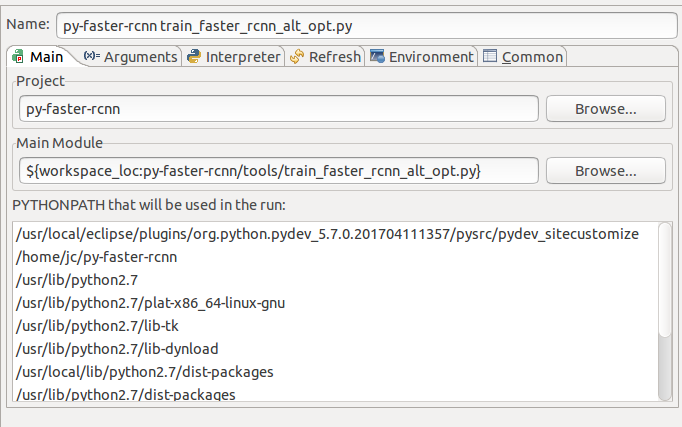
--net\_name VGG\_CNN\_M\_1024

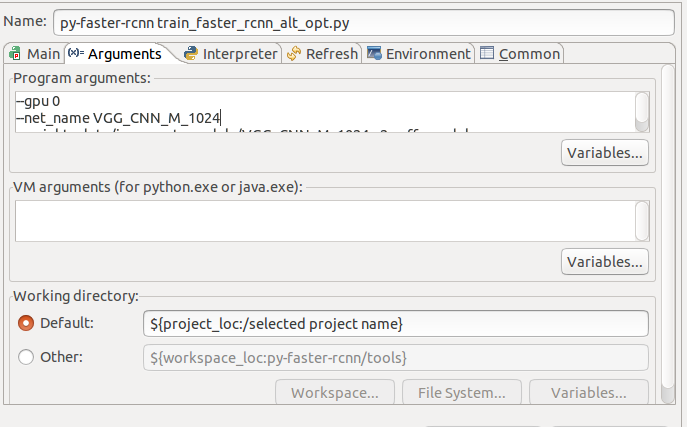
--weights data/imagenet\_models/VGG\_CNN\_M\_1024.v2.caffemodel

--imdb voc\_2007\_trainval

--cfg experiments/cfgs/faster\_rcnn\_alt\_opt.yml

输入参数截图如下：





4. 备注：也可以不在eclipse添加参数，直接把参数写到代码中

4.5 预测阶段

训练完毕，得到训练模型，接下来进行预测

1. 将py-faster-rcnn/output/faster\_rcnn\_alt\_opt/voc\_2007\_trainval/VGG16\_faster\_rcnn\_final.caffemodel，拷贝到py-faster-rcnn/data/faster\_rcnn\_models下

2. 将你需要进行test的images放在py-faster-rcnn/data/demo下

3. 修改py-faster-rcnn/tools/demo.py文件

a.

NETS = {

'vgg\_cnnc\_m\_1024': ('VGG\_CNN\_M\_1024',

'VGG\_CNN\_M\_1024\_faster\_rcnn\_final.caffemodel' ),

'vgg16': ('VGG16',

'VGG16\_faster\_rcnn\_final.caffemodel'),

'zf': ('ZF',

'ZF\_faster\_rcnn\_final.caffemodel')}

b.

CLASSES = ('\_\_background\_\_', 'dog') #参考你自己的类别写

c.

parser.add\_argument('--net', dest='demo\_net', help='Network to use [vgg\_cnnc\_m\_1024]', choices=NETS.keys(), default='vgg\_cnnc\_m\_1024')

d.

im\_names = ['1559.jpg','1564.jpg'] # 改成自己的test image的name

经过以上的更改即可训练自己的数据，若出现问题去仔细调试代码、google解决，问题肯定会解决的，只是时间的问题！

4.6 常见错误解析

1. 如果出现“KeyError: ‘bomb’”说明代码中的key未更改

定位到：py-faster-rcnn/lib/datasets/pascal\_voc.py

2. 如果出现“asseteError[0:2]”，试试把cache文件删除

3. 如果出现“no such file or ditectory:’exripents/cfgs/faster\_rcnn\_alt\_opt.yml”，

更改run config->arguments->default

4.7 参考网址

1. <http://blog.csdn.net/Gavin__Zhou/article/details/52052915>