背锅侠

Faster R-CNN教程

Faster R-CNN教程

最后更新日期：2016年4月29日

本教程主要基于python版本的 faster R-CNN ，因为python layer的使用，这个版本会比 matlab 的版本速度慢10%，但是准确率应该是差不多的。

目前已经实现的有两种方式：

1. Alternative training
2. **Approximate joint training**

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，同时准确率差不多甚至略高一点。

Contents

1. 配置环境
2. 安装步骤
3. Demo
4. 建立自己的数据集
5. 训练和检测

配置环境

1配置python layers

```
#In your Makefile.config, make sure to have this line uncommented
WITH_PYTHON_LAYER := 1
# Unrelatedly, it's also recommended that you use CUDNN
USE_CUDNN := 1
```

2安装几个依赖 `cython, python-opencv, easydict`

```
sudo apt-get install python-opencv
sudo pip install cython easydict
```

安装步骤

1克隆工程

```
git clone --recursive https://github.com/rbgirshick/py-faster-rcnn.git
```

2编译Cython模块

```
cd $FRCN_ROOT/lib
make
```

3编译caffe和pycaffe

```
cd $FRCN_ROOT/caffe-fast-rcnn
# Now follow the Caffe installation instructions here:
#   http://caffe.berkeleyvision.org/installation.html

# If you're experienced with Caffe and have all of the requirements installed
# and your Makefile.config in place, then simply do:
make -j8 && make pycaffe
```

Demo

安装步骤完成后，就可以运行一下demo了。

```
cd $FRCN_ROOT
./tools/demo.py
```


常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

C++(2)
caffe(1)
leetcode(4)
笔记
深度学习(3)
算法(4)

随笔档案

2016年4月 (7)
2016年3月 (20)
2016年2月 (3)
2016年1月 (1)
2015年11月 (1)
2015年10月 (2)

最新评论

1. Re:Ubuntu安装opencv wit…
我是2.4.13按你的方法重新编译一遍，还是报错说没有gpu.hppter/denseImage.cpp:4:31: fatal error: opencv2/gpu/gpu.hpp: No such.....
--北海盗

2. Re:Faster R-CNN教程
@妳叶咚我你好，我也遇到跟你一样的问题了，返回的boxes有nan

训练自己的训练集

工程目录简介

首先工程的根目录简单的称为 FRCN_ROOT，可以看到根目录下有以下几个文件夹

- caffe-fast-rcnn

这里是caffe框架目录

- data

用来存放pretrained模型，比如imagenet上的，以及读取文件的cache缓存

- experiments

存放配置文件以及运行的log文件，另外这个目录下有scripts可以用end2end或者alt_opt两种方式训练。

- lib

用来存放一些python接口文件，如其下的datasets主要负责数据库读取，config负责cnn一些训练的配置选项。

- models

里面存放了三个模型文件，小型网络的ZF，大型网络VGG16，中型网络VGG_CNN_M_1024。推荐使用VGG16，如果使用端到端的approximate joint training方法，开启CuDNN，只需要3G的显存即可。

- output

这里存放的是训练完成后的输出目录，默认会在faster_rcnn_end2end文件夹下

- tools

里面存放的是训练和测试的Python文件。

创建数据集

接下来我们就要创建自己的数据集了，这部分主要在lib目录里操作。这里下面存在3个目录：

- datasets

在这里修改读写数据的接口主要是datasets目录下

- fast_rcnn

值，请问您解决了吗？...
--小小而言

3. Re:Faster R-CNN教程
博主你好！我训练自己的数据集的时候出现scores, boxes = im_detect(net, im)，返回的boxes有nan值，这个怎么解决啊？然后就会提示这种错误 draw raise.....
--妳叻咚我

4. Re:Faster R-CNN教程
@背锅侠学习率设定为0.0001了，但是还是不收敛，loss甚至是变大的趋势，我在想是不是跟数据集有关系，...
--bokeyuanck

5. Re:Faster R-CNN教程
@majisong你用的自己的数据集？test.prototxt是否进行了相应的修改？scores, boxes = im_detect(net, im)这句话返回的就是检测框和得分，你可以依次写入t.....
--背锅侠

阅读排行榜

1. Faster R-CNN教程(18410)
2. Caffe的Solver参数设置(38…
3. 321. Create Maximum Nu…
4. Ubuntu安装opencv with cu…
5. 论文笔记：Ten years of pe…

评论排行榜

1. Faster R-CNN教程(14)
2. 321. Create Maximum Nu…
3. Ubuntu安装opencv with cu…

推荐排行榜

主要存放的是python的训练和测试脚本，以及训练的配置文件config.py

- nms

做非极大抑制的部分，有gpu和cpu两种实现方式

- roi_data_layer

主要是一些ROI处理操作

- rpn

这就是RPN的核心代码部分，有生成proposals和anchor的方法

- transform
- utils

1构建自己的IMDB子类

1.1文件概述
可有看到datasets目录下主要有三个文件，分别是

- factory.py
- imdb.py
- pascal_voc.py

factory.py 是个工厂类，用类生成imdb类并且返回数据库共网络训练和测试使用；imdb.py 这里是数据库读写类的基类，分装了许多db的操作，但是具体的一些文件读写需要继承继续读写；pascal_voc.py Ross在这里用pascal_voc.py这个类来操作。

1.2读取文件函数分析
接下来我来介绍一下pasca_voc.py这个文件，我们主要是基于这个文件进行修改，里面有几个重要的函数需要修改

- def **init**(self, image_set, year, devkit_path=None)
  这个是初始化函数，它对应着的是pascal_voc的数据集访问格式，其实我们将其接口修改的更简单一点。
- def image_path_at(self, i)
  根据第i个图像样本返回其对应的path，其调用了image_path_from_index(self, index)作为其具体实现
- def image_path_from_index(self, index)
  实现了 image_path的具体功能

1. Faster R-CNN教程(5)
2. 321. Create Maximum Nu…
3. Ubuntu安装opencv with cu…

- def _load_image_set_index(self)
  加载了样本的list文件
- def _get_default_path(self)
  获得数据集地址
- def gt_roidb(self)
  读取并返回ground_truth的db
- def selective_search_roidb
  读取并返回ROI的db，这个是fast rcnn用的，faster版本的不用管这个函数。
- def _load_selective_search_roidb(self, gt_roidb)
  加载预选框的文件
- def selective_search_IJCV_roidb(self)
  在这里调用读取Ground_truth和ROI db并将db合并
- def _load_selective_search_IJCV_roidb(self, gt_roidb)
  这里是专门读取作者在IJCV上用的dataset
- def **_load_pascal_annotation**(self, index)
  这个函数是读取gt的具体实现
- def _write_voc_results_file(self, all_boxes)
  voc的检测结果写入到文件
- def _do_matlab_eval(self, comp_id, output_dir='output')
  根据matlab的evluation接口来做结果的分析
- def evaluate_detections
  其调用了_do_matlab_eval
- def competition_mode
  设置competitoin_mode，加了一些噪点

1.3训练数据格式

在我的检测任务里，我主要是在SED数据集上做行人检测，因此我这里只有background 和person 两类物体，为了操作方便，我像pascal_voc数据集里面一样每个图像用一个xml来标注。如果大家不知道怎么生成xml文件，可以用这个工具 labelImg？

这里我要特别提醒一下大家，一定要注意坐标格式，一定要注意坐标格式，一定要注意坐标格式，重要的事情说三遍！！！要不然你会犯很多错误都会是因为坐标不一致引起的报错。

1.4修改读取接口

这里是原始的pascal_voc的init函数，在这里，由于我们自己的数据集往往比voc的数据集要更简单的一些，在作

者代码里面用了很多的路径拼接，我们不用去迎合他的格式，将这些操作简单化即可，在这里我会一一列举每个我修改过的函数。这里按照文件中的顺序排列。

修改后的初始化函数：

```
class hs(imdb):
    def __init__(self, image_set, devkit_path=None):  # modified
        imdb.__init__(self, image_set)
        self._image_set = image_set
        self._devkit_path =  devkit_path   #datasets路径
        self._data_path = os.path.join(self._devkit_path,image_set)   #图片文件夹路径
        self._classes = ('__background__', # always index 0
                         'person')   #two classes
        self._class_to_ind = dict(zip(self.classes, xrange(self.num_classes))) # form the
dict{'__background__':'0','person':'1'}
        self._image_ext = '.jpg'
        self._image_index = self._load_image_set_index('ImageList.txt')
        # Default to roidb handler
        self._roidb_handler = self.selective_search_roidb
        self._salt = str(uuid.uuid4())
        self._comp_id = 'comp4'

        # PASCAL specific config options
        self.config = {'cleanup'     : True,
                       'use_salt'    : True,
                       'use_diff'    : False,
                       'matlab_eval' : False,
                       'rpn_file'    : None,
                       'min_size'    : 16}  #小于16个像素的框扔掉

        assert os.path.exists(self._devkit_path), \
                'VOCdevkit path does not exist: {}'.format(self._devkit_path)
        assert os.path.exists(self._data_path), \
                'Path does not exist: {}'.format(self._data_path)
```

修改后的image_path_from_index：

```
def image_path_from_index(self, index): #modified
    """
    Construct an image path from the image's "index" identifier.
```

```
    """
    image_path = os.path.join(self._data_path,index +'.jpg')
    assert os.path.exists(image_path), \
            'Path does not exist: {}'.format(image_path)
    return image_path
```

修改后的_load_image_set_index：

```
def _load_image_set_index(self,imagelist): # modified
    """
    Load the indexes listed in this dataset's image set file.
    """
    # Example path to image set file:
    # self._devkit_path + /VOCdevkit2007/VOC2007/ImageSets/Main/val.txt
    image_set_file = os.path.join(self._devkit_path, imagelist)
    assert os.path.exists(image_set_file), \
            'Path does not exist: {}'.format(image_set_file)
    with open(image_set_file) as f:
        image_index = [x.strip() for x in f.readlines()]
    return image_index
```

gt_roidb(self):

这个函数里有个生成ground truth的文件，我需要特别说明一下，如果你再次训练的时候修改了数据库，比如添加或者删除了一些样本，但是你的数据库名字函数原来那个，必须要在data/cache/目录下把数据库的缓存文件.pkl给删除掉，否则其不会重新读取相应的数据库，而是直接从之前读入然后缓存的pkl文件中读取进来，这样修改的数据库并没有进入网络，而是加载了老版本的数据。

修改的_load_pascal_annotation(self, index):

```
def _load_pascal_annotation(self, index):    #modified
    """
    Load image and bounding boxes info from XML file in the PASCAL VOC
    format.
    """
    filename = os.path.join(self._devkit_path, 'Annotations', index + '.xml')
    tree = ET.parse(filename)
    objs = tree.findall('object')
    if not self.config['use_diff']:
        # Exclude the samples labeled as difficult
```

```
        non_diff_objs = [
            obj for obj in objs if int(obj.find('difficult').text) == 0]
        # if len(non_diff_objs) != len(objs):
        #     print 'Removed {} difficult objects'.format(
        #         len(objs) - len(non_diff_objs))
        objs = non_diff_objs
    num_objs = len(objs)

    boxes = np.zeros((num_objs, 4), dtype=np.uint16)
    gt_classes = np.zeros((num_objs), dtype=np.int32)
    overlaps = np.zeros((num_objs, self.num_classes), dtype=np.float32)
    # "Seg" area for pascal is just the box area
    seg_areas = np.zeros((num_objs), dtype=np.float32)

    # Load object bounding boxes into a data frame.
    for ix, obj in enumerate(objs):
        bbox = obj.find('bndbox')
        # Make pixel indexes 0-based
        x1 = float(bbox.find('xmin').text)
        y1 = float(bbox.find('ymin').text)
        x2 = float(bbox.find('xmax').text)
        y2 = float(bbox.find('ymax').text)
        cls = self._class_to_ind[obj.find('name').text.lower().strip()]
        boxes[ix, :] = [x1, y1, x2, y2]
        gt_classes[ix] = cls
        overlaps[ix, cls] = 1.0
        seg_areas[ix] = (x2 - x1 + 1) * (y2 - y1 + 1)

    overlaps = scipy.sparse.csr_matrix(overlaps)

    return {'boxes' : boxes,
            'gt_classes': gt_classes,
            'gt_overlaps' : overlaps,
            'flipped' : False,
            'seg_areas' : seg_areas}
```

因为我和Pascal用了一样的xml格式，所以这个函数我的改动不多。如果你想用txt文件保存ground truth，做出相应的修改即可。

坐标的顺序强调一下，要左上右下，并且x1必须要小于x2，这个是基本，反了会在坐标水平变换的时候会出错，坐标从0开始，如果已经是0，则不需要再-1。如果怕出错，可以直接把出界的的直接置0.

记得在最后的main下面也修改相应的路径

```
from datasets.hs import hs
d = hs('hs', '/home/zyy/workspace/wangml/py-faster-rcnn/lib/datasets/')
res = d.roidb
from IPython import embed; embed()
```

OK，在这里我们已经完成了整个的读取接口的改写。

2修改factory.py

当网络训练时会调用factory里面的get方法获得相应的imdb，
首先在文件头import 把pascal_voc改成hs

```
# --------------------------------------------------------
# Fast R-CNN
# Copyright (c) 2015 Microsoft
# Licensed under The MIT License [see LICENSE for details]
# Written by Ross Girshick
# --------------------------------------------------------

"""Factory method for easily getting imdbs by name."""

__sets = {}

from datasets.hs import hs
import numpy as np

# # Set up voc_<year>_<split> using selective search "fast" mode
# for year in ['2007', '2012']:
#     for split in ['train', 'val', 'trainval', 'test']:
#         name = 'voc_{}_{}'.format(year, split)
#         __sets[name] = (lambda split=split, year=year: pascal_voc(split, year))
#
# # Set up coco_2014_<split>
# for year in ['2014']:
#     for split in ['train', 'val', 'minival', 'valminusminival']:
#         name = 'coco_{}_{}'.format(year, split)
```

```
#         __sets[name] = (lambda split=split, year=year: coco(split, year))
#
# # Set up coco_2015_<split>
# for year in ['2015']:
#     for split in ['test', 'test-dev']:
#         name = 'coco_{}_{}'.format(year, split)
#         __sets[name] = (lambda split=split, year=year: coco(split, year))

name = 'hs'
devkit = '/home/zyy/workspace/wangml/py-faster-rcnn/lib/datasets/'
__sets['hs'] = (lambda name = name,devkit = devkit: hs(name,devkit))

def get_imdb(name):
    """Get an imdb (image database) by name."""
    if not __sets.has_key(name):
        raise KeyError('Unknown dataset: {}'.format(name))
    return __sets[name]()

def list_imdbs():
    """List all registered imdbs."""
    return __sets.keys()
```

训练和检测

1.预训练模型介绍

首先在data目录下，有两个目录

- faster_rcnn_models/
- imagenet_models/

faster_rcnn_model文件夹下面是作者用faster rcnn训练好的三个网络,分别对应着小、中、大型网络，大家可以试用一下这几个网络，看一些检测效果，他们训练都迭代了80000次，数据集都是pascal_voc的数据集。

imagenet_model文件夹下面是在Imagenet上训练好的通用模型，在这里用来初始化网络的参数.

在这里我比较推荐先用中型网络训练，中型网络训练和检测的速度都比较快，效果也都比较理想，大型网络的话训练速度比较慢，中型网络训练大概半天，大型网络的话用25个小时。

2.修改模型文件配置

模型文件在models下面对应的网络文件夹下，在这里我用中型网络的配置文件修改为例子

比如：我的检测目标物是person ，那么我的类别就有两个类别即 background 和 person
因此，首先打开网络的模型文件夹，打开train.prototxt
修改的地方重要有三个
分别是个地方

1. 首先在data层把num_classes 从原来的21类 20类+背景 ，改成 2类 人+背景
2. 接在在cls_score层把num_output 从原来的21 改成 2
3. 在bbox_pred层把num_output 从原来的84 改成8， 为检测类别个数乘以4，比如这里是2类那就是2*4=8

OK，如果你要进一步修改网络训练中的学习速率，步长，gamma值，以及输出模型的名字，需要在同目录下的solver.prototxt中修改。

3.启动Fast RCNN网络训练

```
python ./tools/train_net.py --gpu 1 --solver models/hs/faster_rcnn_end2end/solver.prototxt --weights data/imagenet_models/VGG_CNN_M_1024.v2.caffemodel --imdb hs --iters 80000 --cfg experiments/cfgs/faster_rcnn_end2end.yml
```

参数讲解：

- 这里的–是两个-，不要输错
- train_net.py是网络的训练文件，之后的参数都是附带的输入参数
- --gpu 代表机器上的GPU编号，如果是nvidia系列的tesla显卡，可以在终端中输入nvidia-smi来查看当前的显卡负荷，选择合适的显卡
- --solver 代表模型的配置文件，train.prototxt的文件路径已经包含在这个文件之中
- --weights 代表初始化的权重文件，这里用的是Imagenet上预训练好的模型，中型的网络我们选择用VGG_CNN_M_1024.v2.caffemodel
- --imdb 这里给出的训练的数据库名字需要在factory.py的_sets中，我在文件里面有_sets[‘hs’]，train_net.py这个文件会调用factory.py再生成hs这个类，来读取数据

4.启动Fast RCNN网络检测
可以参考tools下面的demo.py 文件，来做检测，并且将检测的坐标结果输出到相应的txt文件中。

最后

鉴于之前我用的版本是15年11月的版本，有些小伙伴在使用此教程时会有一些错误，所以我重新做了部分修订，目前能够在2016年4月29日版本的版本上成功运行，如果有问题，随时联系我。

参考博客： http://www.cnblogs.com/louyihang-loves-baiyan/p/4885659.html

分类: caffe , 深度学习

背锅侠
关注 - 0
粉丝 - 11
+加关注

5 0

« 上一篇： 最长递增子序列
» 下一篇： Introduction to debugging neural networks

posted @ 2016-04-14 11:17 背锅侠 阅读(18410) 评论(14) 编辑 收藏

评论列表

#1楼 2016-07-06 16:10 Yan消云散_

Mark！ 博主好热心

支持(0) 反对(0)

#2楼 2016-09-18 16:07 aspirin_kb

Hi，你的这篇文章真的很有用，谢谢你的贡献。我有一个问题请教，pascal_voc.py文件中有个_load_pascal_annotation(self, index)方法，你在说明时写到“坐标的顺序强调一下，要左上右下，并且x1必须要小于x2，这个是基本，反了会在坐标水平变换的时候会出错，坐标从0开始，如果已经是0，则不需要再-1。如果怕出错，可以直接把出界的的直接置0.”。请问这句话是什么意思？我用Imagelabel标注的图像数据，需要减一吗？

支持(0) 反对(0)

#3楼 [楼主] 2016-09-19 17:10 背锅侠

@ aspirin_kb
xml中的ground truth矩形框给出的是左上角和右下角的坐标(x1,y1),(x2,y2)。坐标应该是从0开始，如果你的标注坐标是从1开始的，就需要-1。labelImg这个工具在做大量ground truth的标注时，不太好用，所以我为自己的任务专门写了一个小工具。labelImg这个工具坐标应该是从0还是1开始的，我也记不太清了。从1开始的坐标，有可能在做镜像翻转的时候造成坐标超出图像边界的情况，所以需要注意坐标的格式。

支持(0) 反对(0)

#4楼 2016-09-19 18:37 aspirin_kb

@ 背锅侠
原来如此，labelImg的坐标是1-based，所以需要减1.请问你自己的小工具有通用性吗？可不可以分享一下？谢谢

支持(0) 反对(0)

#5楼 [楼主] 2016-09-19 18:52 背锅侠

@ aspirin_kb
并没有通用性，所以我才在博客放了labelImg这个工具。

支持(0) 反对(0)

#6楼 2016-09-19 18:54 aspirin_kb

@ 背锅侠
好吧。现在最大的问题是标注自己的图像训练集的时候需要手动标注，如果有什么办法减轻这种工作量就好了。

支持(0) 反对(0)

#7楼 2016-10-17 09:27 jiongnima

博主您好，看了您的文章，我觉得很受益，我想请问下我们在标定图片制作自己的数据集的时候，图片的尺寸是多少呢？期待您的回答。

支持(0) 反对(0)

#8楼 2016-11-08 13:47 majisong

@背锅侠 ，博主，我用这个方法训练了，还没完事，最后用zf_faster_rcnn_iter_70000.caffemodel这个模型输出来测试，修改了demo.py文件，结果：
Loaded network /Users/make/Desktop/py-faster-rcnn/data/faster_rcnn_models/zf_faster_rcnn_iter_70000.caffemodel
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Demo for data/demo/00010.jpg
Detection took 1.310s for 88 object proposals
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Demo for data/demo/00011.jpg
Detection took 1.361s for 71 object proposals

1、没有像demo那样，有图片跳出来，是不是训练失败了？
2、最后的“并且将检测的坐标结果输出到相应的txt文件中”，这个如何做呢？

支持(0) 反对(0)

#9楼 2016-11-09 16:41 bokeyuanck

博主你好，这篇文章很赞！
目前我在kitti上运行faster rcnn进行行人检测，但是在训练过程中loss一直不收敛，训练过程中总是有突变值或nan出现，请问您在训练过程中又遇到这种情况么？
另外kitti的标注文件有很多类别，我只关注pedestrian的话是不是可以直接在load annotation时直接将其他gt_bbox过滤掉？

支持(0) 反对(0)

#10楼 [楼主] 2016-11-09 18:30 背锅侠

@ bokeyuanck
如果经常出现nan或者突变值，是否是学习率设置过大？
把非行人过滤掉当然是可以的。

支持(0) 反对(0)

#11楼 [楼主] 2016-11-09 18:37 背锅侠

@ majisong
你用的自己的数据集？test.prototxt是否进行了相应的修改？

```
scores, boxes = im_detect(net, im)
```

这句话返回的就是检测框和得分，你可以依次写入txt即可。

支持(0) 反对(0)

#12楼 2016-11-09 18:56 bokeyuanck

@ 背锅侠
学习率设定为0.0001了，但是还是不收敛，loss甚至是变大的趋势，我在想是不是跟数据集有关系，

支持(0) 反对(0)

#13楼 2016-11-28 14:45 妳叭咚我

博主你好！我训练自己的数据集的时候出现scores, boxes = im_detect(net, im)，返回的boxes有nan值，这个怎么解决啊？
然后就会提示这种错误 draw
raise ValueError("posx and posy should be finite values")
ValueError: posx and posy should be finite values

支持(0) 反对(0)

#14楼 2017-03-10 13:35 小小而言

@ 妳叭咚我
你好，我也遇到跟你一样的问题了，返回的boxes有nan值，请问您解决了吗？

支持(0) 反对(0)

· 程序员，如何从平庸走向理想？
· 我为什么鼓励工程师写blog
» 更多知识库文章...