公告

昵称：我若成风者
园龄：2年
粉丝：2
关注：3
+加关注

< 2018年3月 >

日	一	二	三	四	五	六
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

文字检测(7)
caffe(3)
SSD(2)
YOLO(1)
测试(1)
核函数(1)
计算机视觉(1)
目标检测(1)
全卷积网络(1)
训练(1)
更多

随笔分类

文字检测(9)

随笔档案

2018年3月 (1)
2018年1月 (2)
2017年8月 (1)
2017年6月 (1)
2017年5月 (1)
2017年4月 (6)

最新评论

1. Re:Caffe上用SSD训练和测试自己的数据
@Snowhitex都需要，预训练意思是指先在其他数据上训练一下，后面用新数据微调就好了。SSD前面用了VGG16作为前面几层，直接在人家训练好的基础上继续训练会效果好也省时间...
--我若成风者
2. Re:Caffe上用SSD训练和测试自己的数据
楼主您好，我想问一下预训练模型是干什么用的，训练其他数据的时候也需要下载这个吗
--Snowhitex
3. Re:Caffe上用SSD训练和测试自己的

caffe SSD目标检测lmdb数据格式制作

一、任务

现在用caffe做目标检测一般需要lmdb格式的数据，而目标检测的数据和目标分类的lmdb格式的制作难度不同。就目标检测来说，例如准备SSD需要的数据，一般需要以下几步：

1.准备图片并标注groundtruth

2.将图像和txt格式的gt转为VOC格式数据

3.将VOC格式数据转为lmdb格式数据

本文的重点在第**2、3步**，第一步标注任务用小代码实现即可。网络上大家制作数据格式一般是仿VOC0712的，建立各种目录，很麻烦还容易出错，现我整理了一下代码，只要两个代码，就可以从图片+txt格式gt的数据转化为lmdb格式，不需要额外的文件夹，换其他数据库也改动非常少，特别方便。

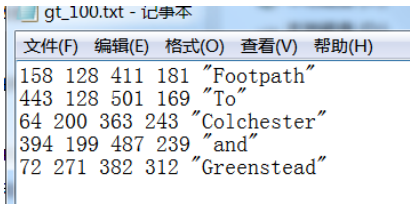
二、准备工作

本文基于已经标注好的数据，以ICDAR2013库为例，起始数据格式如下：

图片目录：ICDAR2013\img\test*.jpg和ICDAR2013\img\train*.jpg

gt目录:ICDAR2013\img\test\gt_*.txt和ICDAR2013\img\train\gt_*.txt

gt的格式为：



三、转VOC格式

1.建立如下目录：Annotations、ImageSets、JPEGImages、label

其中Annotations里面建空文件夹test和train，用来存放转换好的gt的xml形式。当然，可以只建单个，比如只要制作train的数据那就只要建立train文件夹就好了。

ImageSets里面建空文件夹Main，里面存放train.txt和test.txt，txt内容是图片的名字，不带.jpg的名字，初始是空的，是通过代码生成的。

JPEGImages里面建文件夹train和test，并把训练和测试集图片对应扔进去。

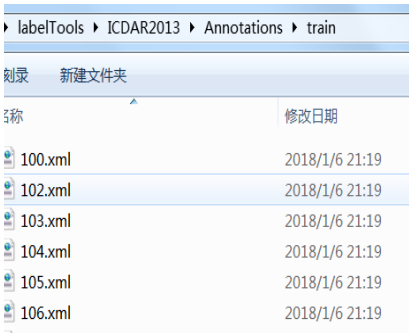
里面建文件夹train和test，并把训练和测试集的groundtruth的txt文件对应扔进去。

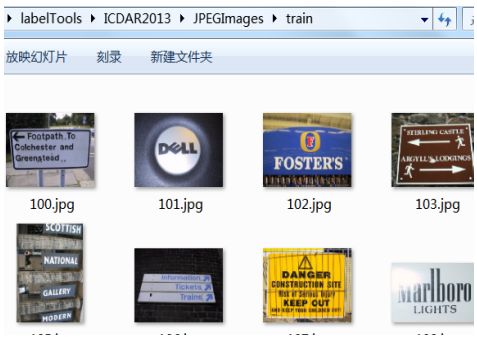
现在格式如下：

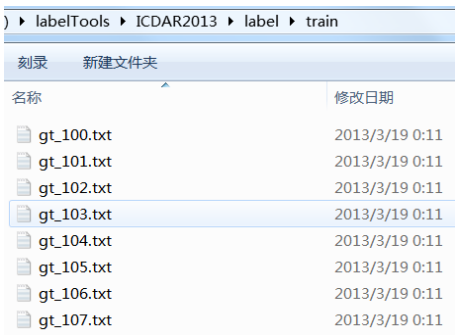
数据
好的，谢谢你
--独行的棕鹿
4. Re:Caffe上用SSD训练和测试自己的数据
@独行的棕鹿不好意思，最近转行了我忘记当时具体怎么弄的了o(╯□╰)o，好像是改的参数，可以参考下这个...
--我若成风者
5. Re:Caffe上用SSD训练和测试自己的数据
您好，我遇到了loss=nan的情况，请问您是怎么解决的？
--独行的棕鹿

阅读排行榜
1. Caffe上用SSD训练和测试自己的数据(17489)
2. caffe SSD目标检测lmdb数据格式制作(388)
3. YOLO(338)
4. Faster-R-CNN(214)
5. FCN-全卷积网络(180)

评论排行榜
1. Caffe上用SSD训练和测试自己的数据(5)







2.使用下面的create_voc_data.py生成xml文件和后续需要的txt文件

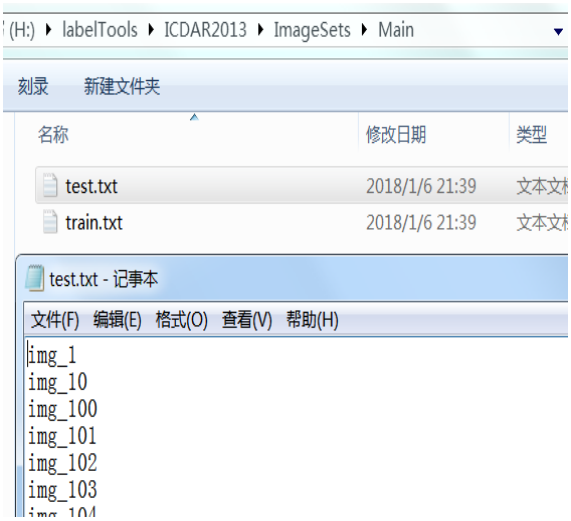
```
View Code
```

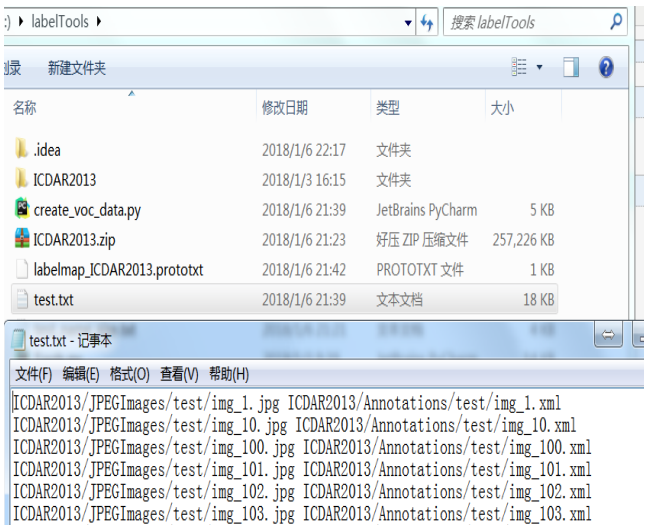
执行上面的代码就得到了

A.Annotations/test下的xml格式文件,只要修改type=train就可以得到训练集的xml格式的gt文件，下同。

B.ImageSets\Main下的test.txt文件

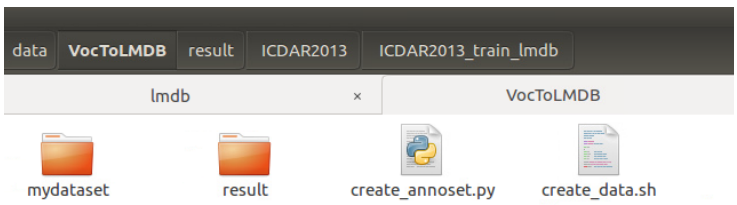
C.执行代码同级目录下的test.txt和test_name_size.txt。这两个文件本应该是用VOCDevit的create_data.sh实现的，此处用python脚本替代了，更方便。注意B和C中的txt文件内容不同，区别如下图：





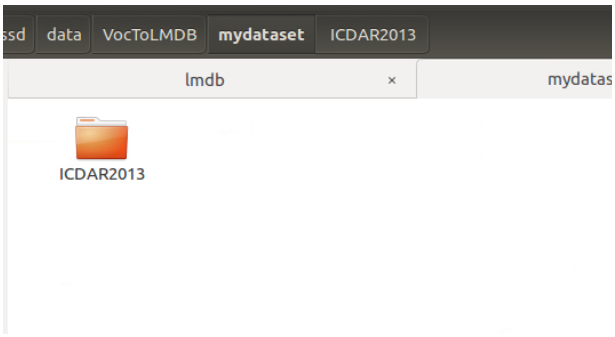
四、制作lmdb格式数据。

现在需要的目录格式是这样的：（mydataset里面存VOC数据，result里面存转好的Lmdb格式的数据和通过上述代码产生的中间结果文件）

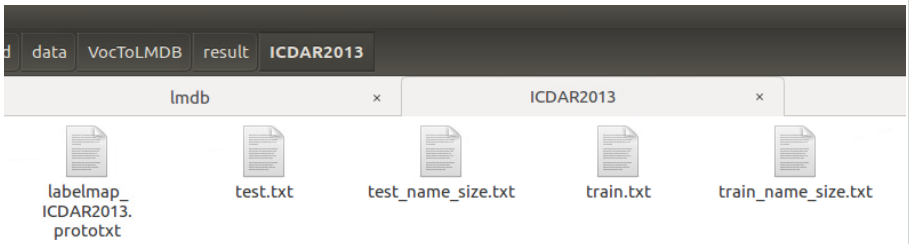


所以需要：

1、建立mydataset文件夹，把刚才制作好的VOC整个文件夹丢进去。以后换其他数据库同样整个丢进mydataset里面就可以。



2、建立result文件夹，下面建立$dataset_name文件夹，（比如ICDAR2013，跟VOC格式里面的名字一致就可以），并把刚才产生的几个文件丢进去。



其中的labelmap_ICDAR2013.prototxt是自己建的类别文件，可以仿照VOC0712里面的，如果做文字检测就只需要两类，那么内容就如下所示：

```
item {
  name: "none_of_the_above"
  label: 0
  display_name: "background"
}
item {
  name: "text"
  label: 1
  display_name: "text"
}
```

3.create_data.sh是VOC0712示例修改过来的，代码如下：

```
cur_dir=$(cd $( dirname ${BASH_SOURCE[0]} ) && pwd )
redo=1
#VOC格式数据存放的文件夹data_root_dir="$cur_dir/mydataset"
#训练集还是测试集，只是标识一下，就是放在一个文件夹里，放test或者train都是可以的，这样只是为了方便切换相同数据库的不同
#数据库名称，只是标记VOC数据在mydataset下面的哪个文件夹里面，结果又放在哪个文件夹里面。dataset_name="ICDAR2013"
mapfile="$cur_dir/result/$dataset_name/labelmap_$dataset_name.prototxt"
anno_type="detection"
db="lmdb"
min_dim=0
max_dim=0
width=0
height=0

extra_cmd="--encode-type=jpg --encoded"
if [ $redo ]
then
  extra_cmd="$extra_cmd --redo"
fi
for subset in $type
do
  #最后一个参数是快捷方式所在的位置，不用建这个文件夹，但是为了代码改的少参数还是要有，我们在下面的create_annoset.py
done
```

4、create_annoset.py是在SSD框架的build/tools里面的，为了方便我们直接把它复制过

来放在我们当前文件夹下，再稍微修改几个地方，修改后如下：

```
import argparse
import os
import shutil
import subprocess
import sys

from caffe.proto import caffe_pb2
from google.protobuf import text_format

if __name__ == "__main__":
  parser = argparse.ArgumentParser(description="Create AnnotatedDatum database")
  parser.add_argument("root",
      help="The root directory which contains the images and annotations.")
  parser.add_argument("listfile",
      help="The file which contains image paths and annotation info.")
  parser.add_argument("outdir",
      help="The output directory which stores the database file.")
  parser.add_argument("exampledir",
      help="The directory to store the link of the database files.")
  parser.add_argument("--redo", default = False, action = "store_true",
      help="Recreate the database.")
  parser.add_argument("--anno-type", default = "classification",
      help="The type of annotation {classification, detection}.")
  parser.add_argument("--label-type", default = "xml",
      help="The type of label file format for detection {xml, json, txt}.")
  parser.add_argument("--backend", default = "lmdb",
      help="The backend {lmdb, leveldb} for storing the result")
  parser.add_argument("--check-size", default = False, action = "store_true",
      help="Check that all the datum have the same size.")
  parser.add_argument("--encode-type", default = "",
      help="What type should we encode the image as ('png','jpg',...).")
  parser.add_argument("--encoded", default = False, action = "store_true",
      help="The encoded image will be save in datum.")
  parser.add_argument("--gray", default = False, action = "store_true",
      help="Treat images as grayscale ones.")
  parser.add_argument("--label-map-file", default = "",
      help="A file with LabelMap protobuf message.")
  parser.add_argument("--min-dim", default = 0, type = int,
      help="Minimum dimension images are resized to.")
  parser.add_argument("--max-dim", default = 0, type = int,
      help="Maximum dimension images are resized to.")
  parser.add_argument("--resize-height", default = 0, type = int,
      help="Height images are resized to.")
  parser.add_argument("--resize-width", default = 0, type = int,
      help="Width images are resized to.")
  parser.add_argument("--shuffle", default = False, action = "store_true",
      help="Randomly shuffle the order of images and their labels.")
  parser.add_argument("--check-label", default = False, action = "store_true",
      help="Check that there is no duplicated name/label.")

  args = parser.parse_args()
  root_dir = args.root
  list_file = args.listfile
  out_dir = args.outdir
  example_dir = args.exampledir

  redo = args.redo
  anno_type = args.anno_type
  label_type = args.label_type
  backend = args.backend
  check_size = args.check_size
  encode_type = args.encode_type
  encoded = args.encoded
  gray = args.gray
  label_map_file = args.label_map_file
  min_dim = args.min_dim
  max_dim = args.max_dim
  resize_height = args.resize_height
  resize_width = args.resize_width
  shuffle = args.shuffle
  check_label = args.check_label

  # check if root directory exists
  if not os.path.exists(root_dir):
```

```
    print "root directory: {} does not exist".format(root_dir)
    sys.exit()
# add "/" to root directory if needed
if root_dir[-1] != "/":
    root_dir += "/"
# check if list file exists
if not os.path.exists(list_file):
    print "list file: {} does not exist".format(list_file)
    sys.exit()
# check list file format is correct
with open(list_file, "r") as lf:
    for line in lf.readlines():
        img_file, anno = line.strip("\n").strip("\r").split(" ")

        if not os.path.exists(root_dir + img_file):
            print "image file: {} does not exist".format(root_dir + img_file)
        if anno_type == "classification":
            if not anno.isdigit():
                print "annotation: {} is not an integer".format(anno)
        elif anno_type == "detection":
            #print(root_dir + anno)
            #print(os.path.exists(root_dir + anno))
            if not os.path.exists(root_dir + anno):
                print "annofation file: {} does not exist".format(root_dir + anno)
                sys.exit()
        break
# check if label map file exist
if anno_type == "detection":
    if not os.path.exists(label_map_file):
        print "label map file: {} does not exist".format(label_map_file)
        sys.exit()
    label_map = caffe_pb2.LabelMap()
    lmf = open(label_map_file, "r")
    try:
        text_format.Merge(str(lmf.read()), label_map)
    except:
        print "Cannot parse label map file: {}".format(label_map_file)
        sys.exit()
out_parent_dir = os.path.dirname(out_dir)
if not os.path.exists(out_parent_dir):
    os.makedirs(out_parent_dir)
if os.path.exists(out_dir) and not redo:
    print "{} already exists and I do not hear redo".format(out_dir)
    sys.exit()
if os.path.exists(out_dir):
    shutil.rmtree(out_dir)

# get caffe root directory
#caffe_root = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
#print(caffe_root)
caffe_root='/dataL/ljy/caffe-ssd'
if anno_type == "detection":
    cmd = "{}/build/tools/convert_annoset" \
        " --anno_type={}" \
        " --label_type={}" \
        " --label_map_file={}" \
        " --check_label={}" \
        " --min_dim={}" \
        " --max_dim={}" \
        " --resize_height={}" \
        " --resize_width={}" \
        " --backend={}" \
        " --shuffle={}" \
        " --check_size={}" \
        " --encode_type={}" \
        " --encoded={}" \
        " --gray={}" \
        " {} {} {}" \
        .format(caffe_root, anno_type, label_type, label_map_file, check_label,
            min_dim, max_dim, resize_height, resize_width, backend, shuffle,
            check_size, encode_type, encoded, gray, root_dir, list_file, out_dir)
elif anno_type == "classification":
    cmd = "{}/build/tools/convert_annoset" \
        " --anno_type={}" \
        " --min_dim={}" \
        " --max_dim={}" \
```

```
        " --resize_height={}" \
        " --resize_width={}" \
        " --backend={}" \
        " --shuffle={}" \
        " --check_size={}" \
        " --encode_type={}" \
        " --encoded={}" \
        " --gray={}" \
        " {} {} {}" \
        .format(caffe_root, anno_type, min_dim, max_dim, resize_height,
            resize_width, backend, shuffle, check_size, encode_type, encoded,
            gray, root_dir, list_file, out_dir)
print cmd
process = subprocess.Popen(cmd.split(), stdout=subprocess.PIPE)
output = process.communicate()[0]

if not os.path.exists(example_dir):
  os.makedirs(example_dir)
link_dir = os.path.join(example_dir, os.path.basename(out_dir))
print(link_dir)
'''
if os.path.exists(link_dir):
  os.unlink(link_dir)

os.symlink(out_dir, link_dir)
'''
```

上面代码修改的地方是：

A.注释掉了最后三句。最后三句是创建快捷方式，可以注释掉。这里不注释掉会报错，原因不明，反正也不需要快捷方式，lmdb有了就万事俱备了。

B.img_file, anno = line.strip("\n").strip("\r").split(" ") ,这句加了("\r")。这句一般情况下改不改都行，但是如果create_voc_data.py是在windows上执行的，后面这个sh在Linux上执行报错就要改，因为windows和linux系统对换行的处理不同，完全按上述步骤会发现到Linux系统上把换号当回车处理了，导致明明路径是对的缺找不到相应文件。

C.caffe_root='/dataL/ljy/caffe-ssd'。这句是把caffe目录切过来。因为原来的代码是严格按照VOC0712数据做的，那么caffe_root就会跟我们不一样，就需要改。

执行create_data.sh就可以在result/ICDAR2013/下面看到我们得到的lmdb格式的数据了。对于相同数据集只要改type=test或者train就行，不用数据集只要改数据集名字就可以。

五、总结。

从无到有生成目标检测Lmdb的步骤为：

1. 获得待制作的图片
2. 用标记工具标记groundtruth,为txt类型的gt。
3. 按上面的步骤三建立VOC目录结构并用create_voc_data.py将2中的数据转为VOC格式。
4. 按上面的步骤四建立结果目录结构并用create_data.py将3中的数据转为lmdb格式，完成。

需要注意下面几点：

1.如何换数据集：只要在上面两个需要建目录的地方把ICDAR2013改成其他库，并把两个代码中的dataset_name改成相应数据集名称就行。

2.如何换相同数据集的的不同部分：比如把ICDAR2013的测试集换成训练集，只要在相应的目录下建立train文件夹，并改代码里面的type=train就可以。

标签: lmdb数据集, SSD, 文字检测, 目标检测, caffe

+加关注

« 上一篇：核函数
» 下一篇：caffe学习记录

posted @ 2018-01-06 22:38 我若成风者 阅读(388) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。