目标检测-足球和梅西

1. 安装`darknet`

同以上项目。安装前，可将足球检测项目重命名为darknet_ball，将messi检测项目重命名为darknet_messi。

2. 给自己的数据集打标签

1) 添加自定义类别

修改文件labelImg/data/predefined_classes.txt

```
ball
messi
trophy
```

2）使用labelImg进行图像标注

用labelImg标注生成PASCAL VOC格式的xml标记文件

3. 整理自己的数据集

1) 下载项目文件：

从百度网盘下载到darknet目录下并解压

```
链接：https://pan.baidu.com/s/1R-azYMOEwOZ5dQpfi2OMVQ
提取码：guk3
```

- VOCdevkit_bm.tar.gz
- visualization.tar.gz
- testfiles.tar.gz
- gen_files.py
- gen_anchors.py
- reval_voc.py
- voc_eval.py
- draw_pr.py

2）生成训练和测试文件

执行

```
python gen_files.py
```

在VOCdevkit / VOC2007目录下可以看到生成了文件夹labels ,同时在darknet下生成了两个文件2007_train.txt和2007_test.txt。2007_train.txt和2007_test.txt分别给出了训练图片文件和测试图片文件的列表，含有每个图片的路径和文件名。另外，在VOCdevkit / VOC2007/ImageSets/Main目录下生产了两个文件test.txt和train.txt，分别给出了训练图片文件和测试图片文件的列表，但只含有每个图片的文件名（不含路径和扩展名）。

labels下的文件是images文件夹下每一个图像的yolo格式的标注文件，这是由Annotations的xml标注文件转换来的。

4. 修改配置文件

1）新建data/voc.names文件

可以复制data/voc.names再根据自己情况的修改；可以重新命名如：data/voc-bm.names

2）新建 cfg/voc.data文件

可以复制cfg/voc.data再根据自己情况的修改；可以重新命名如：cfg/voc-bm.data

3）新建cfg/yolov3-voc.cfg

可以复制cfg/yolov3-voc.cfg再根据自己情况的修改；可以重新命名cfg/yolov3-voc-bm.cfg：

在cfg/yolov3-voc.cfg文件中，三个yolo层和各自前面的conv层的参数需要修改：

三个yolo层都要改：yolo层 classes=2; conv层 filters=21

5. 训练自己的数据集

1）在`darknet`目录下载权重文件：

wget https://pjreddie.com/media/files/darknet53.conv.74

这里的训练使用迁移学习，所以下载的yolo预训练的权重文件（不含全连接层）

2）训练

```
./darknet  detector train cfg/voc-bm.data cfg/yolov3-voc-bm.cfg darknet53.conv.74
```

如需要存储训练日志，执行

```
./darknet detector train cfg/voc-bm.data cfg/yolov3-voc-bm.cfg darknet53.conv.74 2>1 |
tee visualization/train_yolov3_bm.log
```

执行前应建立visualization目录。可通过将visualization.zip解压到darknet项目目录下。

3) 训练log文件分析

```
cd visualization
```

修改 extract_log.py文件，改动

extract_log('train_yolov3_bm.log','train_log_loss.txt','images')
extract_log('train_yolov3_bm.log','train_log_iou.txt','IOU')

```
python extract_log.py
```

得到两个文件: train_log_loss.txt, train_log_iou.txt

改变其中的lines的值。

然后，执行:

```
python train_loss_visualization.py
```

```
python train_iou_visualization.py
```

得到avg_loss.png和Region Avg IOU.png

4) 训练设置

- batch=64
- subdivisions=16
- width=608
- height=608
- max_batches = 4000
- policy=steps
- steps=3200,3600

6. 测试训练出的网络模型

训练好后可以在backup看到权重文件

尝试test前要修改cfg文件，切换到test模式。

可以重新建立一个测试cfg文件, 如yolov3-voc-bm-test.cfg

测试图片:

```
./darknet detector test cfg/voc-bm.data cfg/yolov3-voc-bm-test.cfg backup/yolov3-voc-
bm_final.weights testfiles/test_img5.jpg
```

测试视频:

```
./darknet detector demo cfg/voc-bm.data cfg/yolov3-voc-bm-test.cfg backup/yolov3-voc-
bm_final.weights testfiles/messi.mp4
```

7. 性能统计

计算mAP

首先执行

```
./darknet detector valid cfg/voc-bm.data cfg/yolov3-voc-bm-test.cfg backup/yolov3-voc-bm_final.weights
```

生成results/comp4_det_test_ball.txt和results/comp4_det_test_messi.txt文件

然后执行

```
python reval_voc.py  --voc_dir /home/bai/darknet/VOCdevkit --year 2007 --image_set test --classes  /home/bai/darknet/data/voc-bm.names testBM
```

生成testBM/ball_pr.pkl文件和testBM/messi_pr.pkl文件

画出PR曲线

修改文件draw_pr.py

fr = open('testBM/ball_pr.pkl','rb')

fr = open('testBM/messi_pr.pkl','rb')

然后可画出PR曲线，通过执行

```
python draw_pr.py
```

8. 修改默认的先验框大小

1) 使用k-means聚类获得自己数据集的先验框大小

修改gen_anchors.py文件

```
width_in_cfg_file = 608.
height_in_cfg_file = 608.
```

执行

```
python  gen_anchors.py
```

得到的anchor大小

anchorbox.w*32

anchorbox.h*32

2) 修改cfg文件中的先验框大小

3) 重新训练和测试

8. 修改默认的先验框大小

1) 使用k-means聚类获得自己数据集的先验框大小

修改gen_anchors.py文件

```
width_in_cfg_file = 608.
height_in_cfg_file = 608.
```

执行

```
python  gen_anchors.py
```

得到的anchor大小

anchorbox.w*32

anchorbox.h*32

2) 修改cfg文件中的先验框大小

3) 重新训练和测试