

目标检测笔记二：Object Detection API 小白实践指南

发表于2017-09-10

本文使用公开数据去运行Tensorflow 新推出的 Object Detection API 带大家实验 Faster RCNN 的 training。

Faster RCNN 是 object detection 中的经典方法, 而 object detection 主要是由 classification 与 localization 所组成, 可以参考 cs231n

网络上已经有一堆原理说明文了, 但是纯小白要实践来看看却总是很不知所措, 因此本文偏小白详细描述如何运作项目, 其次过度封装好的数据, 也让小白想应用的时候无从下手, 因此本文完成一个简单demo简述如何构建自己的数据集

TensorFlow model 官方开源网

址: https://github.com/tensorflow/models/tree/master/object_detection

自建的tensorflow有趣小项目开源网址: <https://github.com/luyishisi/tensorflow>, 持续更新小项目欢迎star

附各种依赖公开数据和模型的下载链接: <https://pan.baidu.com/s/1c23vV5A> 密码: 7877

想了解具体定位算法的原理和差异可以参考: 博客链接, 知乎链接

目录：

1. 环境安装{ubuntu与window 7}
2. 数据预处理
3. 修改配置
4. 开始训练
5. 测试模型

一.环境安装：

ubuntu：

1：TensorFlow环境二选一：

亲测用使用公开数据CPU需要在i5下跑一晚上，GPU只要30分钟，建议安装TensorFlow 1.00

		Python
1	<code>pip install tensorflow</code>	# For CPU
2	<code>pip install tensorflow-gpu</code>	# For GPU

2：依赖环境

	Python
1	<code>sudo apt-get install protobuf-compiler python-pil python-lxml</code>
2	<code>sudo pip install jupyter,matplotlib,pillow,lxml</code>

3：务必需要的操作

必须编译Protobuf库，在object_detection同级目录打开终端运行：

	Python
1	<code>protoc object_detection/protos/*.proto --python_out=.</code>

将object_detection加入到环境变量

打开.bashrc 修改下面PYTHONPATH为你的object_detection的路径

	Python
1	<code>export PYTHONPATH=\$PYTHONPATH:`pwd`:`pwd`/slim</code>

4：环境监测

在object_detection同级目录打开终端运行：

	Python
1	<code>python object_detection/builders/model_builder_test.py</code>

结果没有报错，并且返回若干秒数据，则说明环境搭建成功。

```
virtual-machine:~/models-master$ python object_detection/builders/model_builder_test.py
.....
-----
Ran 7 tests in 0.023s

OK
virtual-machine:~/models-master$
```

window 7

在window下回麻烦的多

1：打开下载好的tensorflow model文件夹

2：安装 protoc 在 github.com/google/protobuf

下载protoc-3.4.0-win32.zip。解压后在bin目录下有 protoc.exe。我将bin和include两个文件夹，移到C:\Windows目录下（在path的即可），然后再mdels（或者models-master）文件夹下运行如下命令：

```
protoc.exe object_detection/protos/*.proto --python_out=.
```

没有报错即正确

3：安装tensorflow model 以及slim

在models-master 目录下运行

	Python
1	<code>python setup.py install</code>

如果你这时候运行测试命令

	Python
1	<code>python object_detection/builders/model_builder_test.py</code>

会报错：ImportError: No module named nets

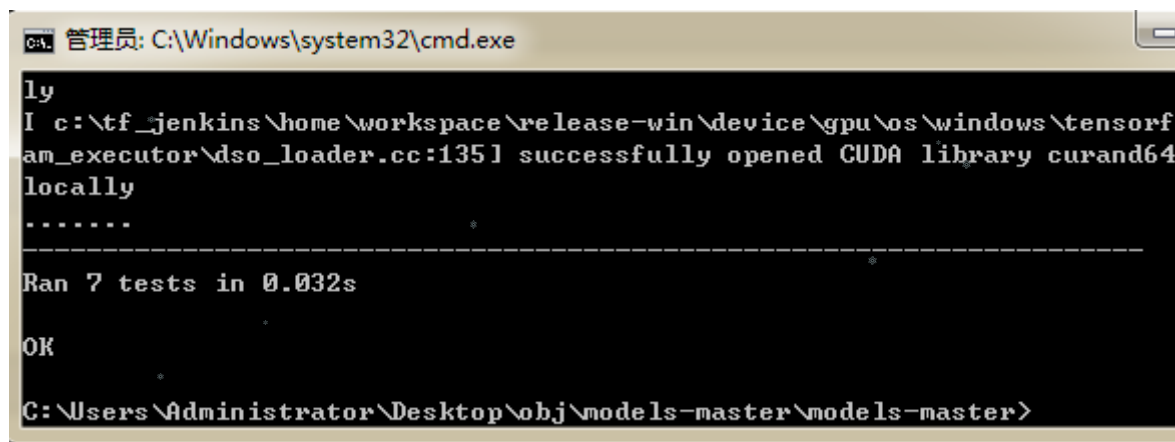
则需要进一步安装slim，在models-master\slim目录下也运行

```
1 python setup.py install
```

4：配置path环境变量，如下：

```
Python
1 C:\Users\Administrator\Desktop\obj\models-master\models-master:C:\Users\Adminis
```

5：测试成功



```
管理员: C:\Windows\system32\cmd.exe
ly
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\tensorflow\python\api\tf_record\am_executor\dso_loader.cc:1351 successfully opened CUDA library curand64_10.dll locally
-----
Ran 7 tests in 0.032s
OK
C:\Users\Administrator\Desktop\obj\models-master\models-master>
```

二.数据预处理

Tensorflow对象检测API必须使用TFRecord的档案格式，我用的是2007年的数据集，如果你手边有2012年的 - year要改成2012. 详细内容可参考标准TensorFlow格式，Pascal VOC数据集，我存放一份在百度云链接。。含预训练好的模型，和2007年的数据

数据预处理

解压缩VOCtrainval然后运行create_pascal_tf_record.py来处理成TFRecord。

```
Python
```

```
1 # From tensorflow/models/object_detection
2 tar -xvf VOCtrainval_11-May-2007.tar
3 python create_pascal_tf_record.py --data_dir=VOCdevkit \
4     --year=VOC2007 --set=train --output_path=pascal_train.record
5 python create_pascal_tf_record.py --data_dir=VOCdevkit \
6     --year=VOC2007 --set=val --output_path=pascal_val.record
```

这个create_pascal_tf_record.py做的事情分为三个部分

- 将每张图片注释参数（图片的宽度与高度，对象边界框，类名称，...等）跟标签映射（类ID跟类名称的对应关系）读出来并塞进tf.train.Example协议缓冲区
- 将tf.train.Example协议缓冲区序列化为字符串
- 最后tf.python_io.TFRecordWriter把字符串写入TFRecords

三.修改配置

直接从项目中复制一个样本出来改（object_detection/samples/configs/）我是使用的是faster_rcnn_resnet101_voc07.config

配置文件分成五个部分，

1. model模型的框架 meta-architecture, feature extractor...
2. train_config, 定义 optimizer (Momentum, Adam, Adagrad...), fine-tune model
3. eval_config, 定义valuation估值指标
4. train_input_config, 定义作为训练数据集与标签映射路径
5. eval_input_config, 定义作为估值数据集的路径与标签映射路径

主要修改这三部分

1：自定义路径指定模型位置

fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED/model.ckpt"

通常在进行训练时不会从头开始训练，大部份会利用别人已经训练好的参数来微调以减少训练的时间

fine_tune_checkpoint的数值为：你定义的faster_rcnn_resnet101_coco_11_06_2017位置（例如："object_detection/faster_rcnn_resnet101_coco_11_06_2017/model.ckpt"）

2：指定训练数据的label和record数据文件

label文件 官方已经有提供放在 object_detection/pascal_val.record

Python

```

1 train_input_reader: {
2   tf_record_input_reader { input_path: "PATH_TO_BE_CONFIGURED/pascal_train.rec
3   label_map_path: "PATH_TO_BE_CONFIGURED/pascal_label_map.pbtxt"}

```

3：指定测试数据的label和record数据文件

Python

```

1 eval_input_reader: {
2   tf_record_input_reader {   input_path: "PATH_TO_BE_CONFIGURED/pascal_val.rec
3   label_map_path: "PATH_TO_BE_CONFIGURED/pascal_label_map.pbtxt"
4 }

```

四.启动训练

构建标准项目结构，建立demo目录为主文件夹

- demo目录下包含（train和eval,config文件）
- train目录下包含（faster_rcnn_resnet101_coco_11_06_2017的解压后文件）
- eval是为空的，用于存放之后跑测试的文件

另外我比较喜欢在新建一个dete文件夹，存放上面处理后的record数据文件，和pascal_label_map.pbtxt类别映射表文件
然后开始运行吧！

Python

```

1 python object_detection/train.py \
2   --logtostderr \
3   --pipeline_config_path=${定义的Config} \
4   --train_dir=${训练结果要存放的目录}

```

如果你是按照上诉的标准结构的话则：

Python

```

1 python train.py \
2     --logtostderr \
3     --pipeline_config_path="./demo/**.config" \
4     --train_dir="./demo/train/"}
```

运行需要较大内存5-8G，训练时日志如下

当你的loss到0.5以下，基本就算训练的比较准了，可以在运行eval来看看你的测试结果。

```

INFO:tensorflow:global step 1687: loss = 0.7289 (0.541 sec/step)
INFO:tensorflow:global step 1688: loss = 0.8001 (0.526 sec/step)
INFO:tensorflow:global step 1689: loss = 0.4442 (0.590 sec/step)
INFO:tensorflow:global step 1690: loss = 0.7304 (0.629 sec/step)
INFO:tensorflow:global step 1691: loss = 0.6340 (0.603 sec/step)
INFO:tensorflow:global step 1692: loss = 1.2686 (0.516 sec/step)
INFO:tensorflow:global step 1693: loss = 0.5660 (0.567 sec/step)
```

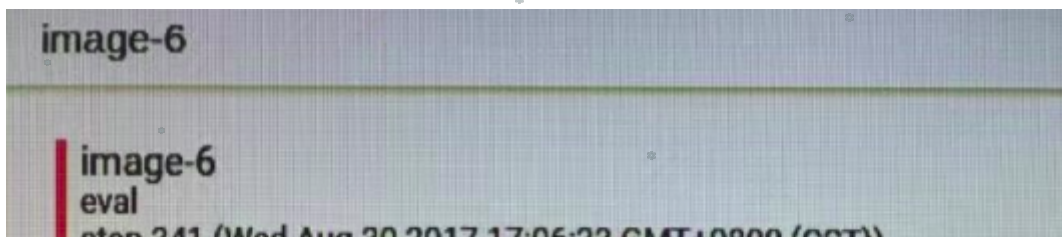
五.测试模型：

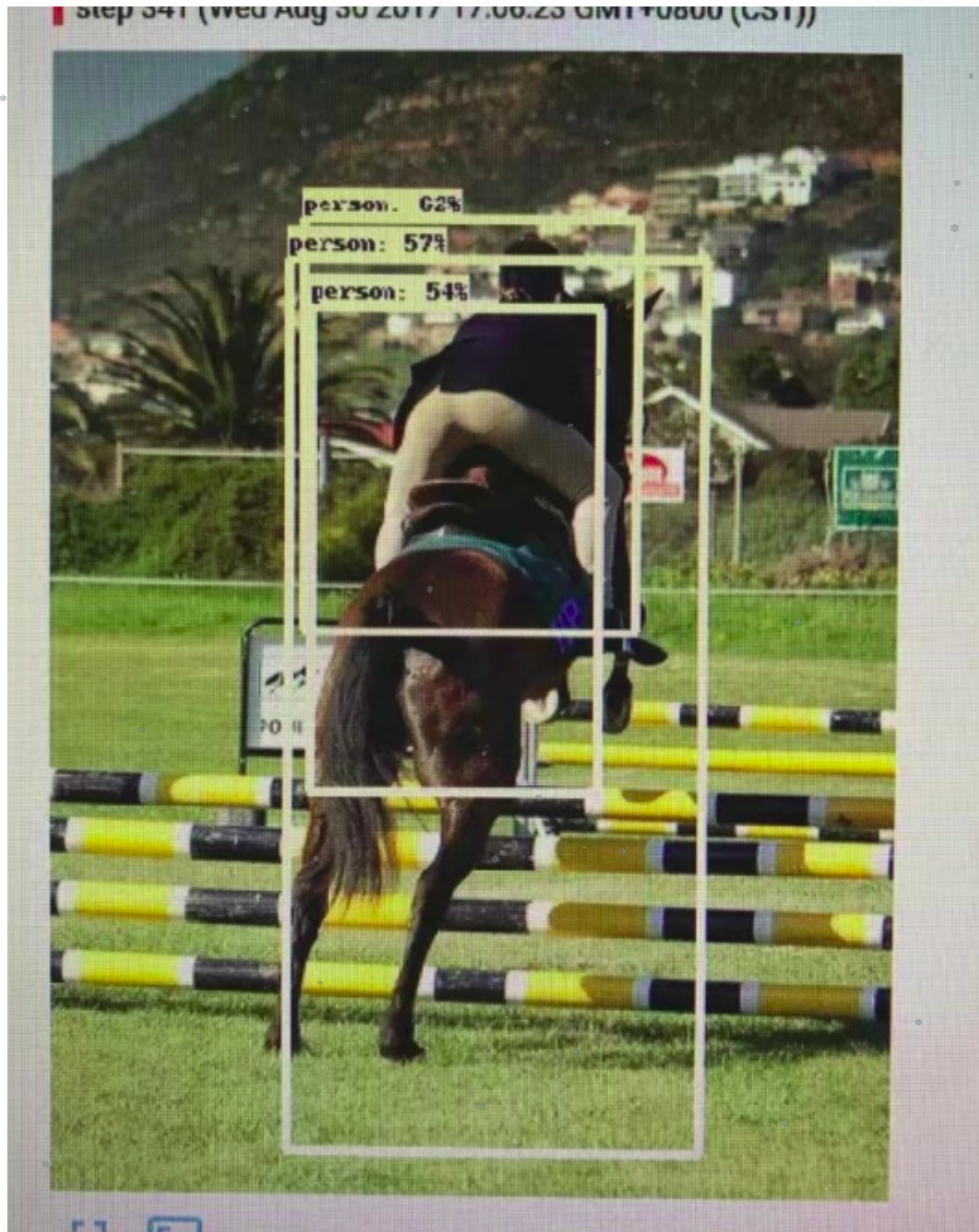
	Python
1	<code>python object_detection/eval.py \</code>
2	<code>--logtostderr \</code>
3	<code>--pipeline_config_path=\${定义的Config} \</code>
4	<code>--checkpoint_dir=\${训练模型存放的目录} \</code>
5	<code>--eval_dir=\${测试结果要存放的目录}</code>

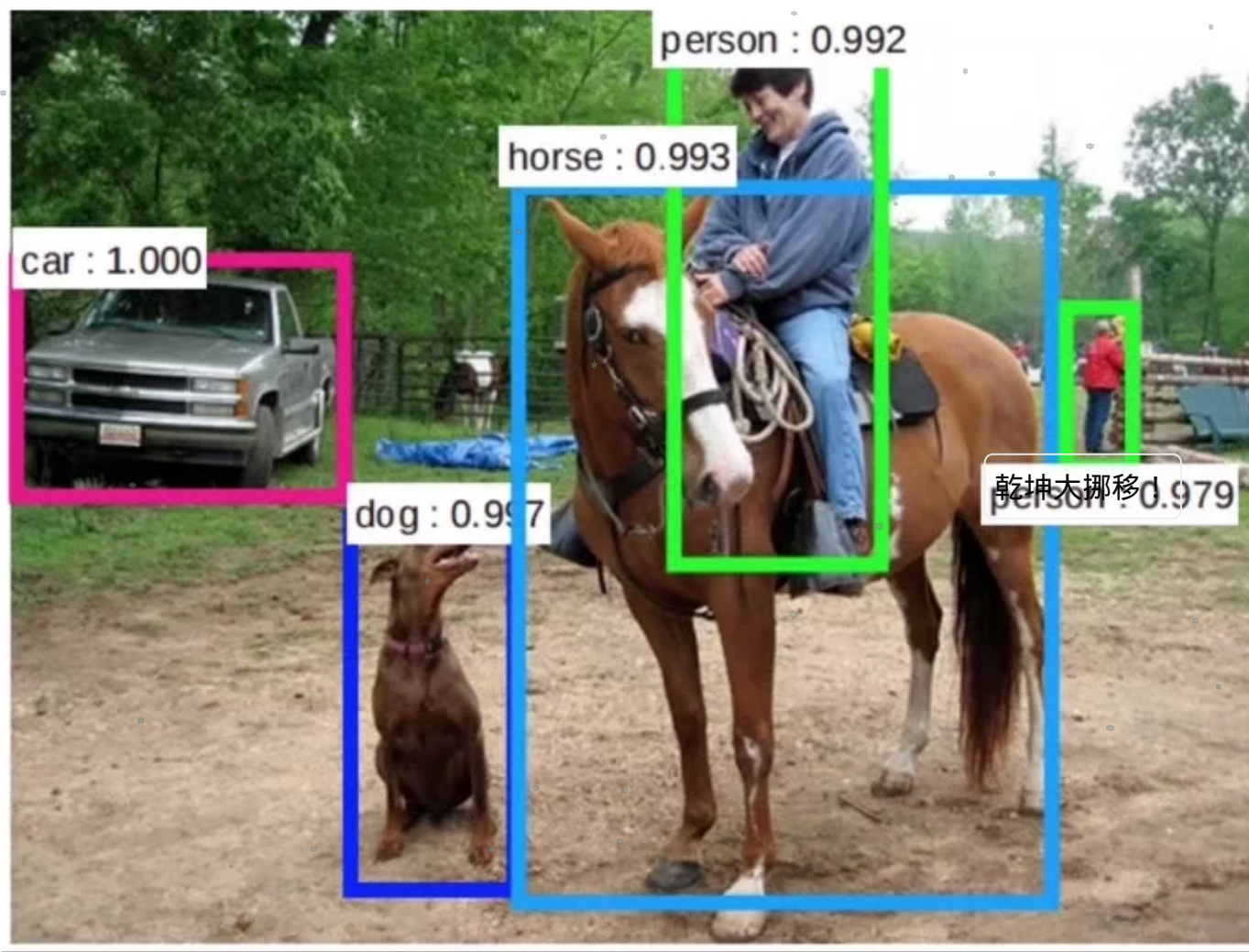
之后再针对这个demo启动tensorboard

	Python
1	<code>tensorboard --logdir demo</code>

之后浏览器查看127.0.1.1:6006,在image下即可看到具体的识别结果了。







六：训练自己定义的数据，
请看github：https://github.com/luyishisi/tensorflow/tree/master/4.Object_Detection
四种定位算法的原理对比：链接

原创文章，转载请注明：转载自URI-team

本文链接地址：目标检测笔记二：Object Detection API 小白实践指南

Related Posts:

1. CNN结构模型一句话概述：从LeNet到ShuffleNet
2. TensorFlow识别字母扭曲干扰型验证码-开放源码与98%模型
3. **TensorFlow资源大全-中文版**

此项目被张贴在 [机器学习](#) 和标记 [tensorflow](#)、[机器学习](#)、[深度学习](#)、[神经网络](#)。书签的 [permalink](#)

← 目标检测领域笔记一：四种算法入门与优缺点对比

[image net 2012数据集以及中文标签分享](#) →

分类目录

[acm \(24\)](#)

[face++ \(5\)](#)

[hackathon \(4\)](#)

[ios \(3\)](#)

[linux \(53\)](#)

[mac \(2\)](#)

[mysql \(6\)](#)

[python \(63\)](#)