**4 行人检测**

1. 目的
2. 掌握基于卷积神经网络的目标检测的基本原理。
3. 了解迁移学习，在数据样本较小的情况下学会利用预训练模型提升自身模型精度。
4. 掌握如何组织TFRecord格式数据。
5. 掌握TensorFlow Object Detection API的使用。
6. 要求
7. 基于TensorFlow Object Detection API实现对行人的检测，环境配置为windows10， TensorFlow 1.12.0，python3.6。
8. TensorFlow Object Detection API的详细使用教程，官方提供的[文档](https://github.com/tensorflow/models/tree/master/research/object_detection)。
9. 所用工具及数据集
10. 该实验需要下载TensorFlow Object Detection API的[源码](https://github.com/tensorflow/models)（使用最新版本的API源码（20190621）），之后拷贝model/research/目录下的object\_detection和slim目录到为该实验创建的目录下。
11. 数据和训练结果下载地址。https://pan.baidu.com/s/1-y27Qh3LWviVALXNx8rSVQ
12. 小型行人检测数据集[TownCentre](http://www.robots.ox.ac.uk/ActiveVision/Research/Projects/2009bbenfold_headpose/project.html" \l "datasets)，包含一个视频TownCentreXVID.avi和标签文件TownCentre-groundtruth.top。TownCentreXVID.avi一共5 min，每1 sec包含25帧图像（1920\*1080），因此一共包含7500帧图像；TownCentre-groundtruth.top包含前4500帧图像中行人的位置信息，每一行信息组织格式如下：

personNumber, frameNumber, headValid, bodyValid, headLeft, headTop, headRight, headBottom, bodyLeft, bodyTop, bodyRight, bodyBottom

* personNumber - A unique identifier for the individual person
* frameNumber - The frame number (counted from 0)
* headValid - 1 if the head region is valid, 0 otherwise
* bodyValid - 1 if the body region is valid, 0 otherwise
* headLeft,headTop,headRight,headBottom - The head bounding box in pixels
* bodyLeft,bodyTop,bodyRight,bodyBottom - The body bounding box in pixels

对于行人检测，主要需要上述标红的数据。

1. 下载Faster R-CNN预训练模型，[地址](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md)https://github.com/tensorflow/models/blob/master/research/object\_detection/g3doc/detection\_model\_zoo.md。参考下载faster\_rcnn\_inception\_v2\_coco，也可下载其他模型进行实验。主要需要如下文件：

| - model.ckpt.meta

| - model.ckpt.data-00000-of-00001

| - model.ckpt.index

| - pipeline.config

1. 实验的组织目录如下：

| - Pedestrian-Detection

| - object\_dection // API源码

| - slim

| - TownCentre //原始数据集

| - TownCentreXVID.avi

| - TownCentre-groundtruth.top

| - pretrained //存储预训练Faster R-CNN模型

| - Dataset

| - images //训练用图像

| - test\_images //可用于测试

| - annotations

| - xmls

| - label\_map.pbtxt

| - trainval.txt

四、实验步骤及注意事项

1 环境配置（将object\_detection和slim移到根目录下）

1）编译Protobuf，生产py文件

需要先安装Google的protobuf，下载protoc-3.4.0-win32.zip <https://github.com/protocolbuffers/protobuf/releases>

执行如下：protoc object\_detection/protos/\*.proto --python\_out=.

将生成一堆python文件

2）测试编译是否成功

python object\_detection/builders/model\_builder\_test.py

3）Windows下出现找不到object\_detection包的问题。解决方法

在..\Anaconda3\soft\Lib\site-packages新建一个pth文件，将PedestrianDetection文件夹和slim文件夹路径写入。

1. 最新版的API，需要安装coco包

Windows10下的安装方法见博客 https://www.cnblogs.com/dongzhiwu/p/9000804.html

2 制作image图像和xml标签数据

1. 提取每帧的img图像

利用opencv的VideoCapture从TownCentreXVID.avi中抽取用于训练的图像帧4500帧，将其尺寸减半，存储到Dataset/images中，剩余的帧作为测试图像，存储在Dataset/test\_images中，代码见extract\_towncentre.py。

1. 提取行人标注xml信息

从TownCentre-groundtruth.top提取前4500帧（0-4499）图像的行人信息，保存到Dataset/annotations/xmls。其中’size’保存了图像的尺寸信息，’object’包含了该帧中每一个行人的位置信息，’filename’保存了该帧的图像名，代码见extract\_GT.py。因为图像的尺寸缩小了2倍，所以标注中的信息也缩小2倍。

1. 构造trainval.txt文件

将用于训练的所有帧的文件名写入Dataset/annotation/trainval.txt文件中，即将0-4499写入到文件中，一行一个数字，代码见code/trainval.py。

1. 为行人指定id和标签，构造label\_map.pbtxt

为满足TensorFlow Object Detection的使用需求，为行人指定一个id和标签，将以下信息写入到Dataset/annotation/label\_map.pbtxt文件中：

item {

id: 1

name: ‘pedestrian’

}

3.将上述获取的images和xml文件转为TFRecord文件

images中一共包含4500张图像，其中95%用于训练，剩余图像用于验证（参考代码create\_tf\_record.py）。

其中除了保存在json文件的信息外，还包括图像像素信息及前面label\_map.pbtxt文件中保存的信息。

会得到train.tfrecord和val.tfrecord两个文件。

4 修改API训练所需的预训练模型中的pipeline.config文件,并放到根目录下

即前面下载的预训练模型中的pipeline.config文件，只需要修改以下几项，用于指定预训练模型所在路径，即TFRecord数据集所在路径（建议填写绝对路径，不然可能会报错无法找到文件）：

gradient\_clipping\_by\_norm**:** 10.0

fine\_tune\_checkpoint**:** "预训练模型model.ckpt的路径"

from\_detection\_checkpoint**:** true

num\_steps**:** 200000

**}**

train\_input\_reader **{**

label\_map\_path**:** "label\_map.pbtxt的路径"

tf\_record\_input\_reader **{**

input\_path**:** "train.record的路径"

**}**

**}**

eval\_config **{**

num\_examples**:** 8000

max\_evals**:** 10

use\_moving\_averages**:** false

**}**

eval\_input\_reader **{**

label\_map\_path**:** "label\_map.pbtxt的路径"

shuffle**:** false

num\_epochs**:** 1

num\_readers**:** 1

tf\_record\_input\_reader **{**

input\_path**:** "val.record的路径"

**}**

**}**

5 模型训练

总结前面的工作，一是数据的预处理，把数据转换为API所需的TFRecord文件格式；二是API的准备，包括预训练模型下载，API所需文件的准备（如label\_map.pbtxt文件），预训练模型**pipeline.config**文件的修改。

准备完成之后，可直接利用API进行模型的训练，训练命令如下（训练会自动保存模型结果到train目录下，按Ctrl+C随时结束训练）：

***注：***使用最新版API，模型的训练迁移到了model\_main.py：

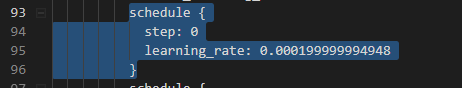
python object\_detection/model\_main.py \ --pipeline\_config\_path=pipeline.config \ --model\_dir=train \ --num\_train\_steps=10000 \ --sample\_1\_of\_n\_eval\_examples=1 \ --alsologtostderr

1）出现的问题

raise ValueError('First step cannot be zero.')

ValueError: First step cannot be zero.

解决方法：删除pipeline.Config中的下列代码



2）出现的问题：

Assertion failed: [maximum box coordinate value is larger than 1.100000: ] [1.15277779]

解决方法：修改extract\_GT的代码，将提取出的GT的长宽进行限制，不要超出图像的长宽。

6 模型测试

1. 将得到的checkpoint转换为pb文件

模型训练完成之后，会在train目录下得到一系列模型的checkpoint文件，选取最后的文件，将其转换为TensorFlow的pb文件（可理解为包含网络结构和参数的文件），API提供了转换代码，命令如下（10000修改为训练结束得到的数字）：

1. 利用pb\_output下的frozen\_inference\_graph.pb对训练后的模型进行测试

python object\_detection/export\_inference\_graph.py \ --input\_type=image\_tensor \ --pipeline\_config\_path=pipeline.config \ --trained\_checkpoint\_prefix=train/model.ckpt-**10000** \ --output\_directory=pb\_output

构造好模型之后，可使用OpenCV读取测试图像进行测试，将测试图像的结果进行保存(文件夹Dataset/result\_test\_images),参考代码predec.py，并保存为视频result.gif，代码见img2gif.py。

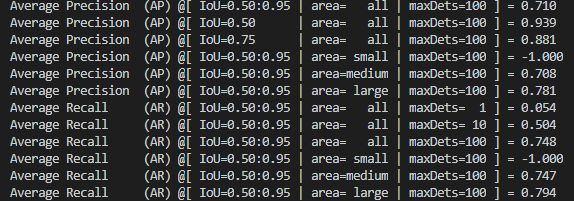
五 实验结果

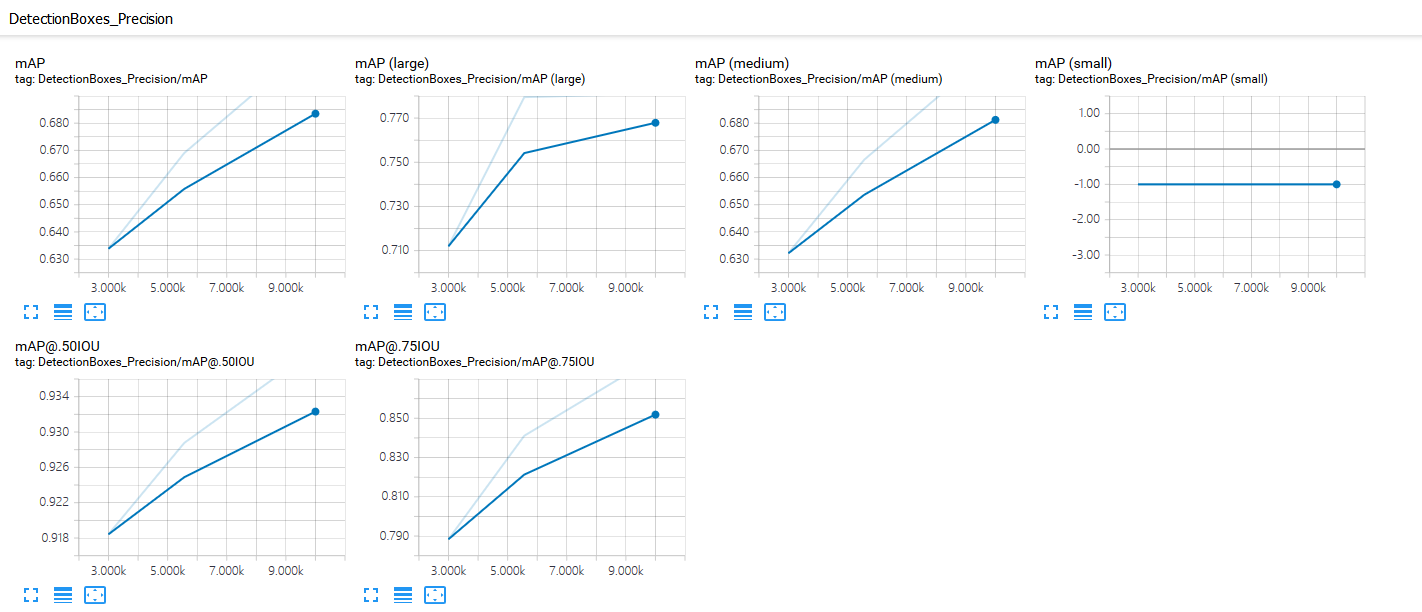
1. 训练模型的最终精度

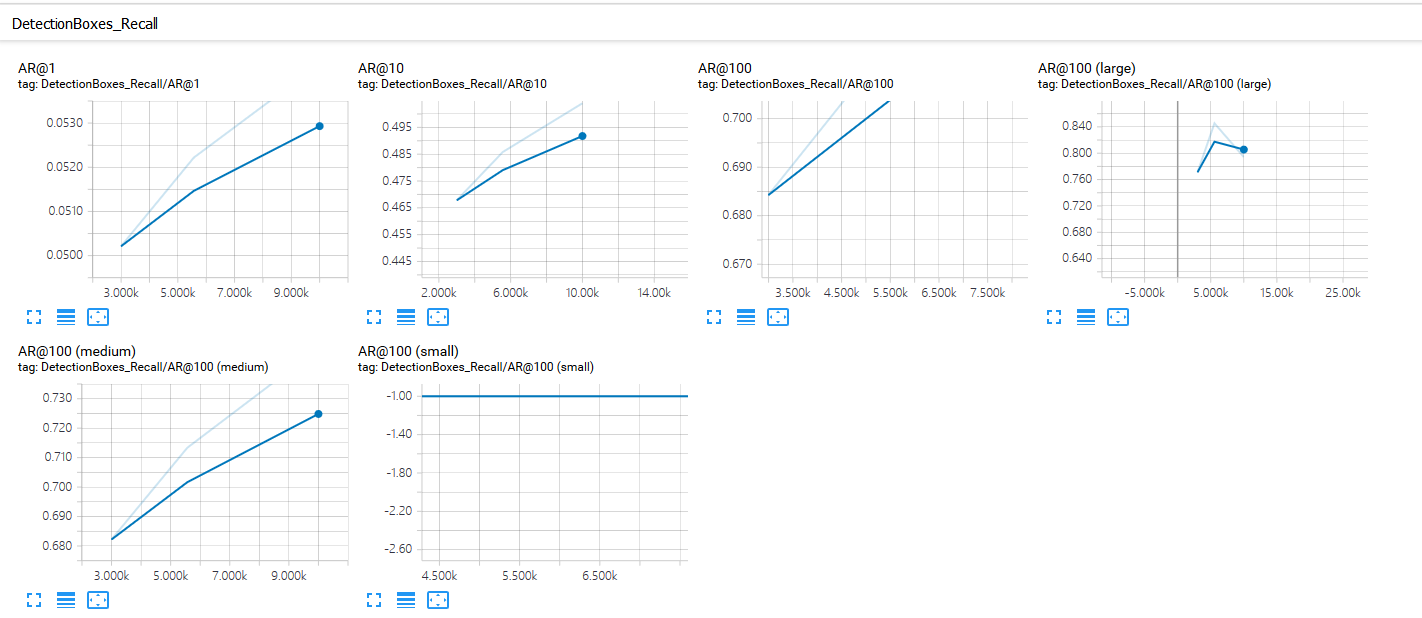
1）迭代次数：10k，下表为AP和AR值

|  |  |  |  |
| --- | --- | --- | --- |
| IoU | 0.5：0.95 | 0.5 | 0.75 |
| Average Precision（AP） | 0.710 | 0.939 | 0.881 |
| Average Recall | None | None | 0.748 |

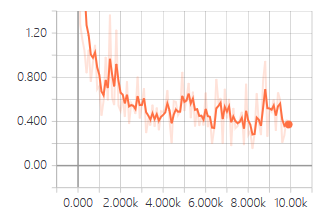
截图如下：







1. Loss=0.3574



1. 测试结果（视频见result\_4500\_5056.gif）

截图如下：

