**实验七 行人检测 作业报告**

1. 实验目的
2. 掌握基于卷积神经网络的目标检测的基本原理。
3. 了解迁移学习思想，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. 实验要求
7. 该实验基于TensorFlow Object Detection API实现对行人的检测，环境配置为Ubuntu 16.04， TensorFlow 1.9.0。
8. 对于TensorFlow Object Detection API的详细使用教程，请自行阅读官方提供的[文档](https://github.com/tensorflow/models/tree/master/research/object_detection)（点击查看），该实验指导书仅对实现行人检测相关的操作进行介绍。
9. 该指导书未涉及到Faster R-CNN原理性及代码实现的解释，详情请参考课件并自行查阅Faster R-CNN原始论文及开源代码。
10. 实验所用工具及数据集
11. 该实验需要下载TensorFlow Object Detection API的[源码](https://github.com/tensorflow/models)（该实验使用的API源码已给出，因为涉及环境的版本问题没有使用最新版的API源码，如果使用最新版本的API源码的话，应该需要使用TensorFlow 1.12.0，并按API安装文档进行相关配置）。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。
12. 该实验提供一个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)，该数据集包含一个视频TownCentreXVID.avi和标签文件TownCentre-groundtruth.top。其中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，我们主要需要上述标红的数据。

1. 下载Faster R-CNN预训练模型，[下载地址](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，也可以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

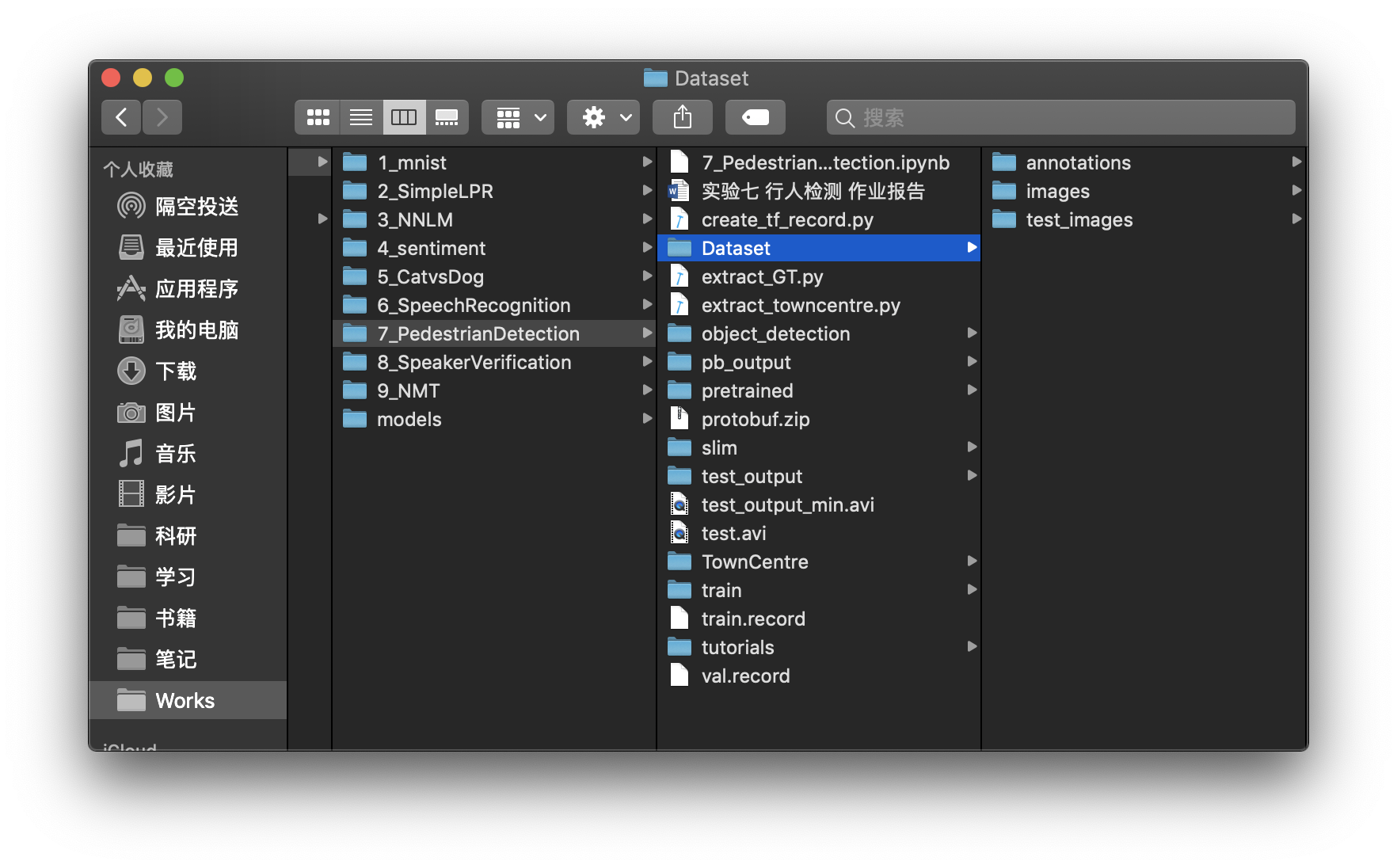
| - json

| - label\_map.pbtxt

| - trainval.txt

1. 代码实现

**1. 准备工作**



首先需要安装protobuf，MacOS下使用如下命令：

**brew install protobuf**

然后进行protobuf编译：

protoc object\_detection/protos/\*.proto --python\_out=.

将object\_detection和slim加入环境变量：

**export PYTHONPATH="$PYTHONPATH:/Users/cuitianyu/Works/JupyterProject/Deep learning/7\_PedestrianDetection:/Users/cuitianyu/Works/JupyterProject/Deep learning/7\_PedestrianDetection/slim"**

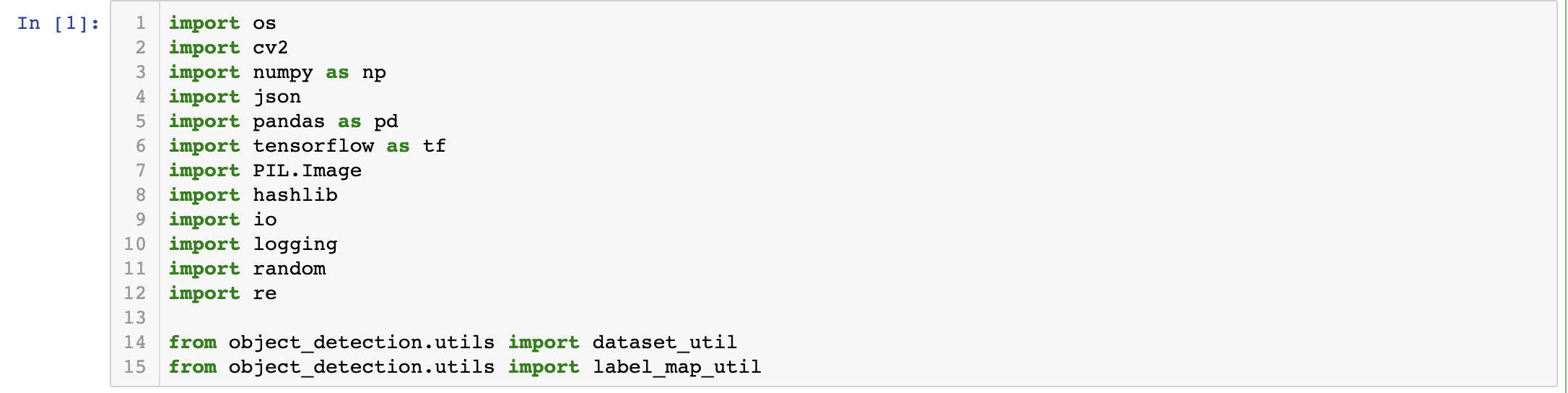
其次进行预训练模型参数配置:

打开pretrained文件夹下pipeline.config文件，将其中train\_config修改如下，这其中包括了几个路径的修改，还有需要删除其中一个schedule，其下的step为0会导致tensorflow PythonAPI报错。具体修改后的内容如下：



**2. 代码实现**

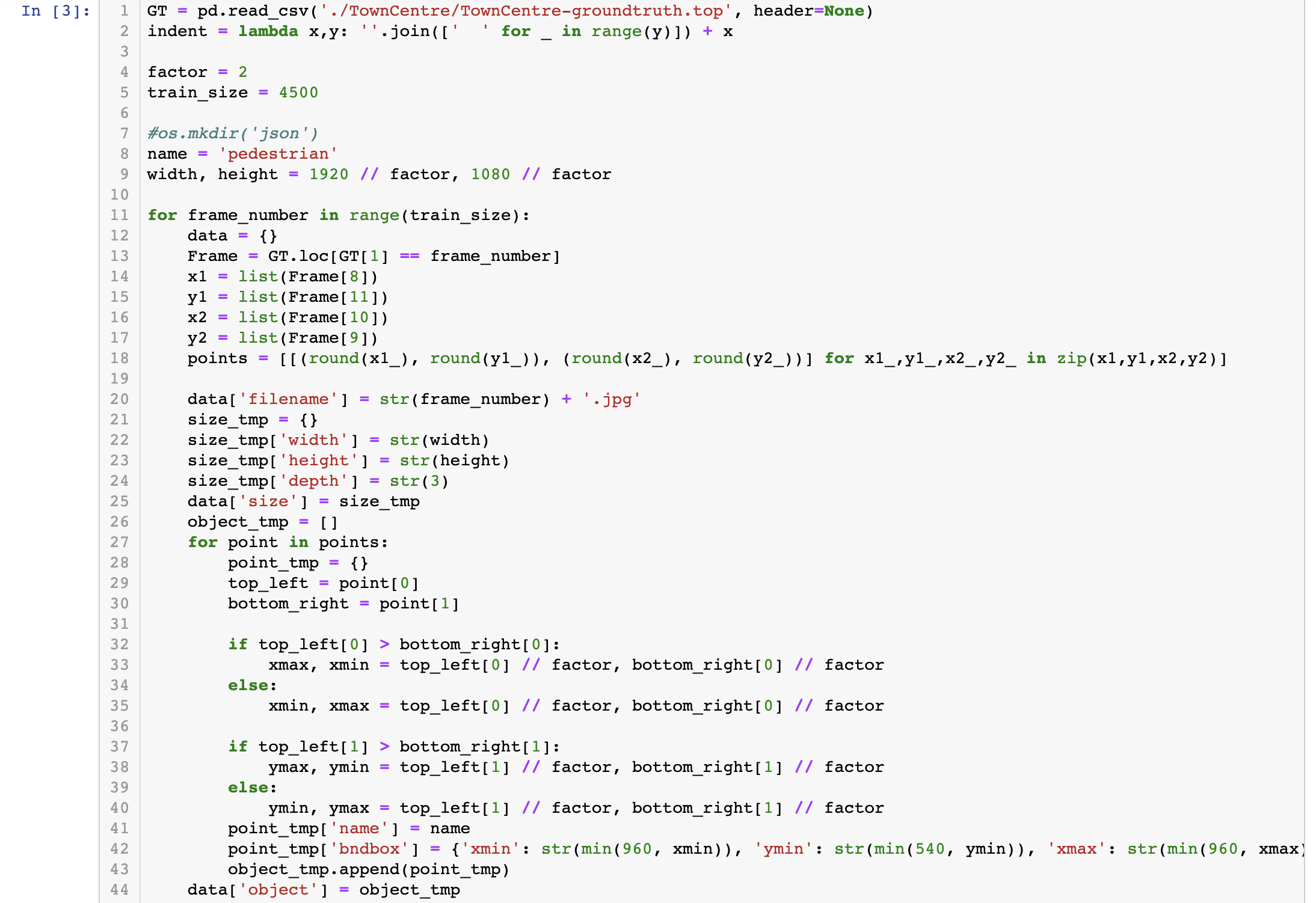
代码依赖

****

读取数据集视频，将其按帧读取并存为图片，图片前4500张为训练集，其余为测试集：

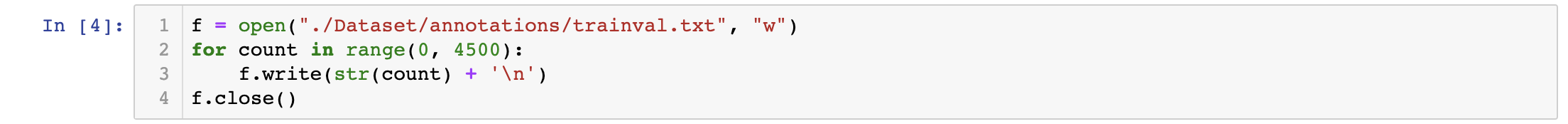
****

读取数据集.top文件，将其中数据保存为json格式以备使用：

****

****

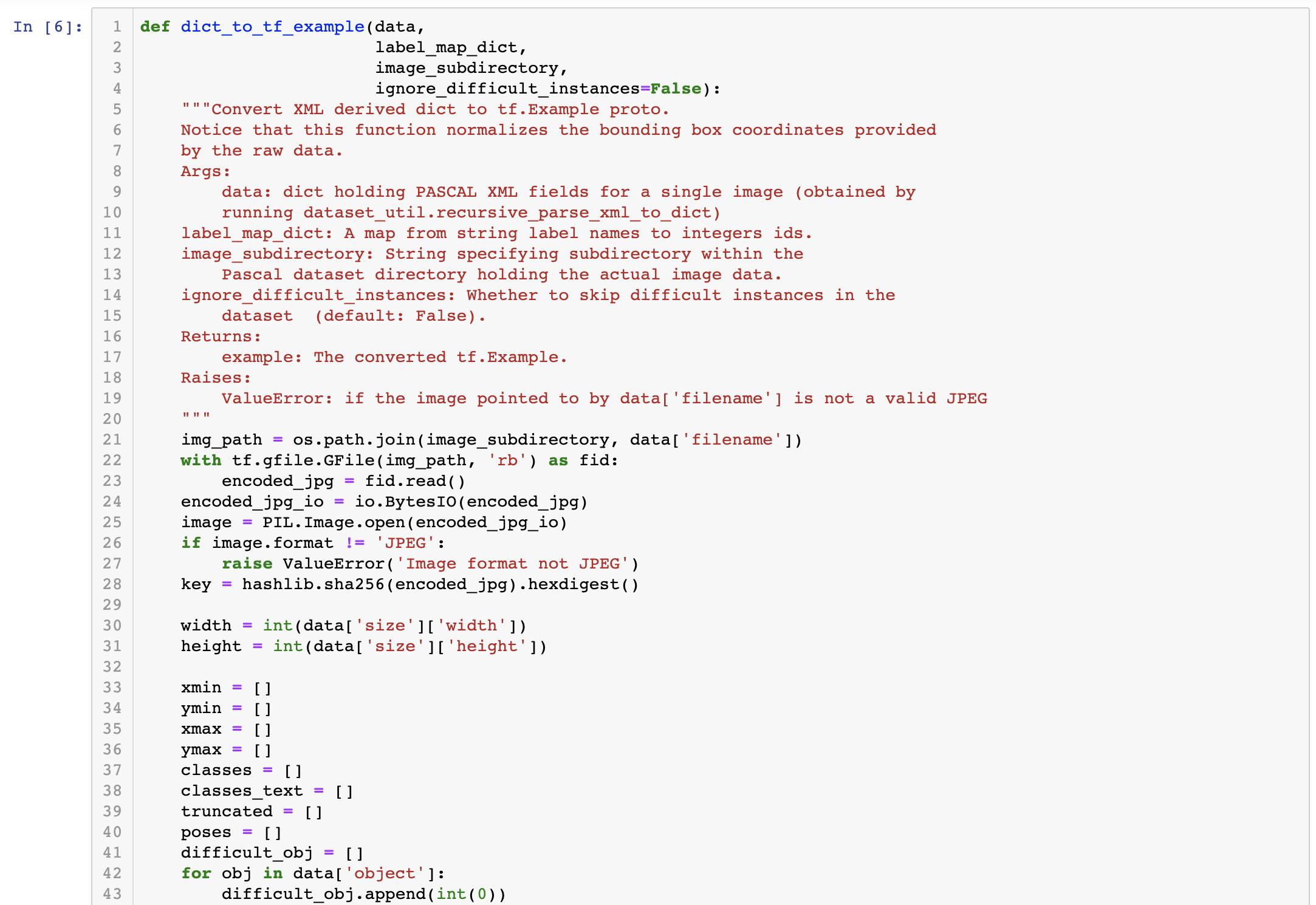
新建API所需的trainval.txt文件,存入训练集编号：

****

新建API所需的label\_map.pbtxt文件，标明所有类别label，本实验只有行人一类：

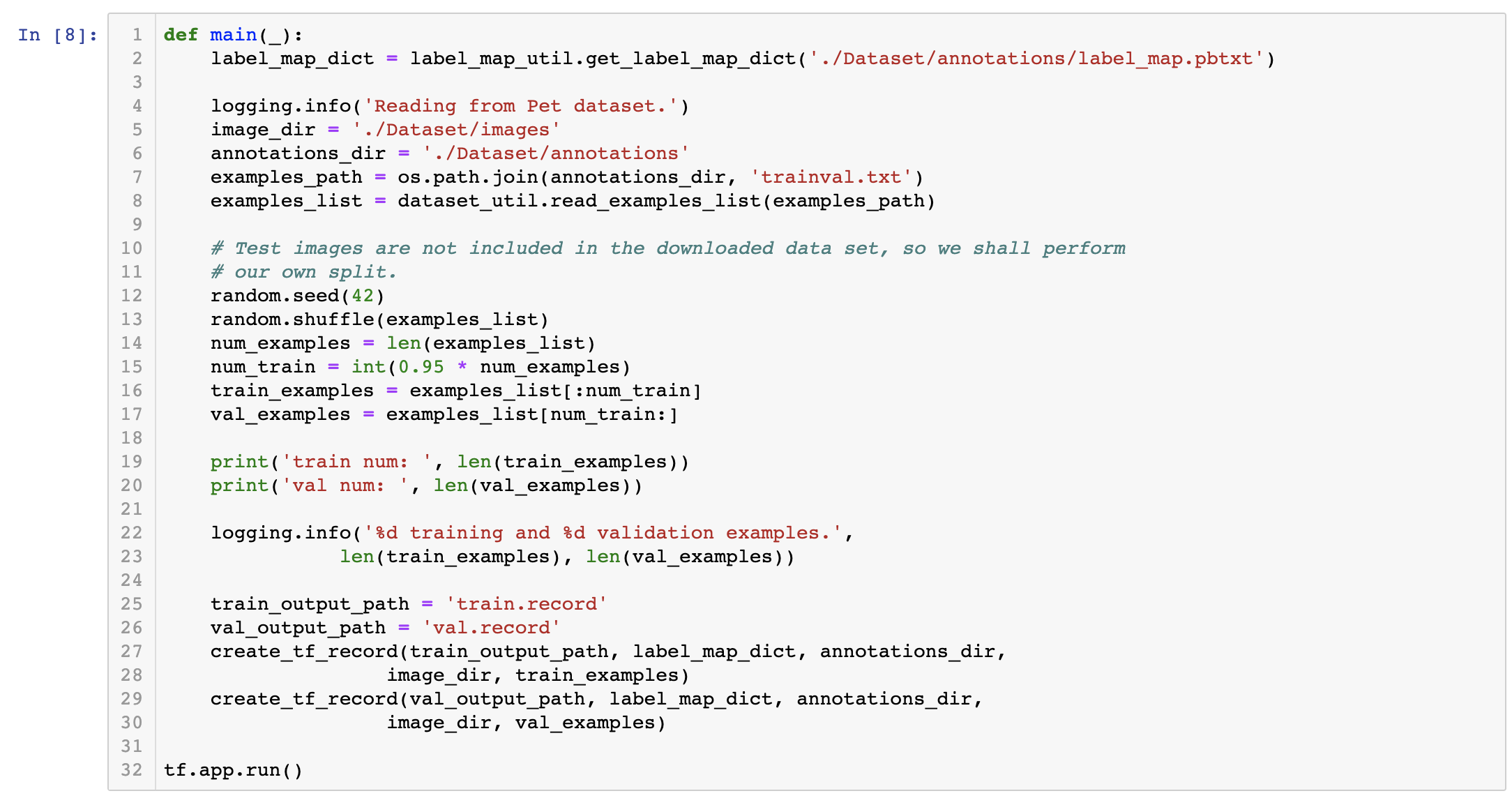


读取json文件，将所需的内容制作成tf\_record，其中将数据区分为训练集和验证集：









接下来进行模型训练，保存模型在train\目录下，模型训练的命令如下：

pythonw object\_detection/model\_main.py \

--pipeline\_config\_path="/Users/cuitianyu/Works/JupyterProject/Deep

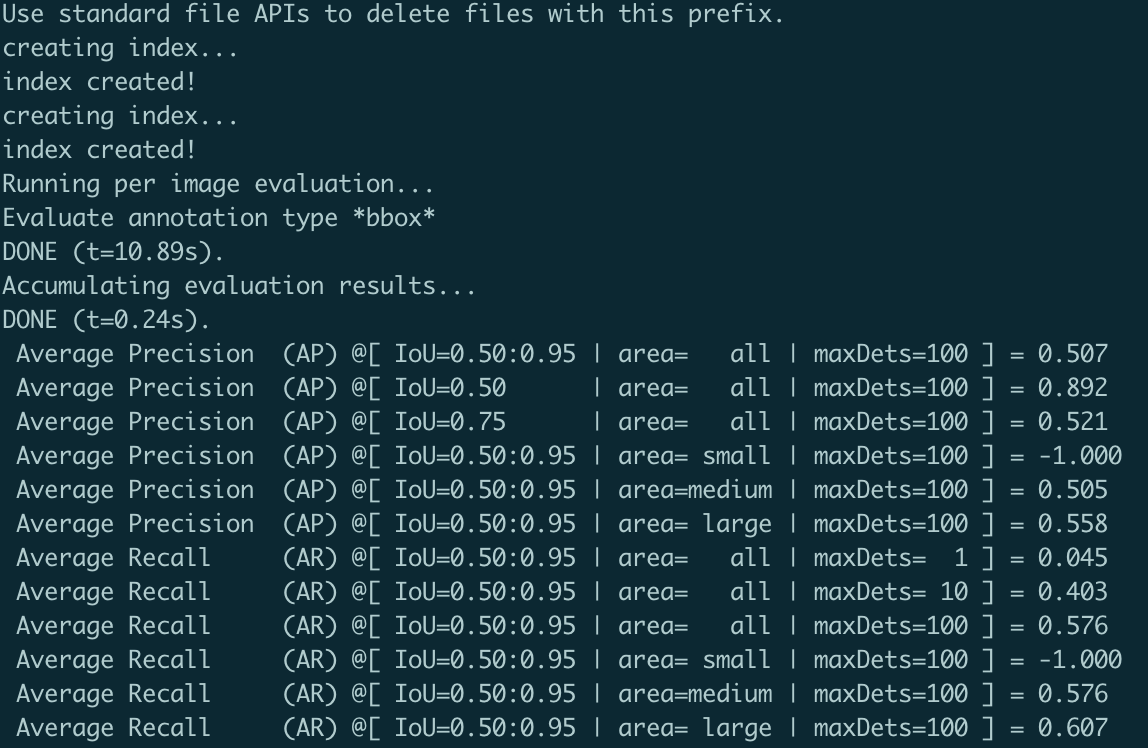
learning/7\_PedestrianDetection/pretrained/pipeline.config" \

--model\_dir=train \

--num\_train\_steps=1000 \

--sample\_1\_of\_n\_eval\_examples=1 \

--alsologtostderr



最后将生成训练的graph保存在pb\_output目录下：

python object\_detection/export\_inference\_graph.py \

--input\_type=image\_tensor \

--pipeline\_config\_path=pretrained/pipeline.config \

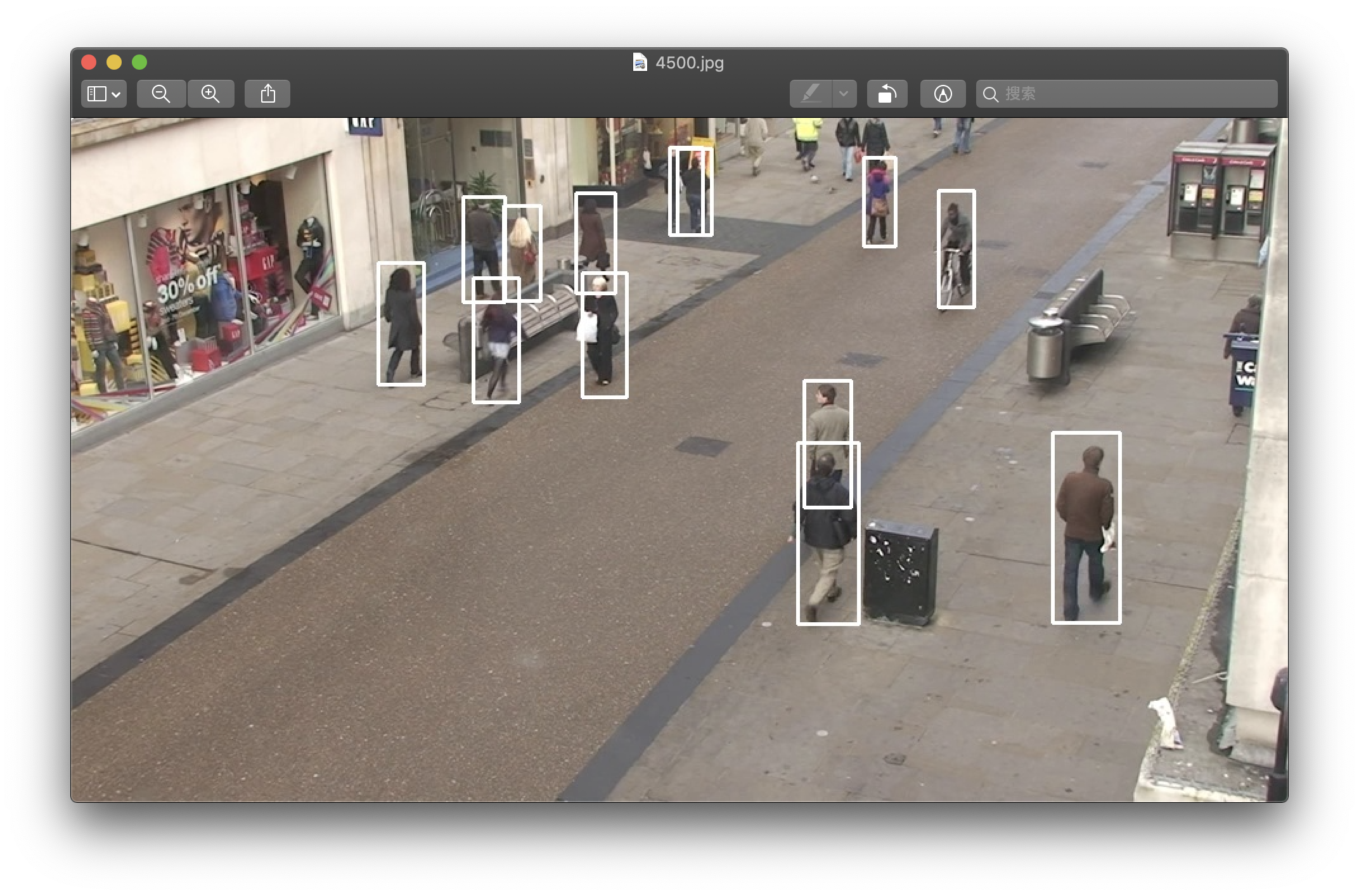
--trained\_checkpoint\_prefix=train/model.ckpt-1000 \

--output\_directory=pb\_output

读取保存的.pb文件，读取test\_images进行测试，画出目标边框并保存图片



测试结果：



将所有输出图片制作成为输出视频：



最终test\_output\_min.avi即是得到的输出视频，由于只训练了1000轮，所以模型会误识别橱窗里的假人，并且在两人或多人重合时识别困难，增加训练轮数长时间训练可能解决这一问题。