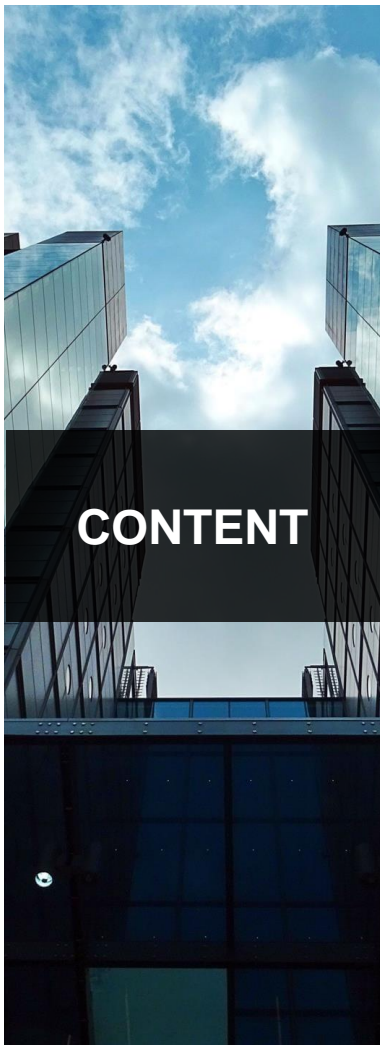


January 5 2019

深度学习环境搭建

孟超

m542946369@gmail.com



1

- 一：安装显卡驱动，配置依赖环境
-

2

- 二、下载SSD网络源码，修改参数
-

3

- 三、目标检测SSD网络demo展示
-

4

- 四、智能物流车行人检测演示
-

January 5 2019

/01

安装显卡驱动，配置依赖环境

1.1 GPU简介

GPU(Graphics Processing Unit)即图形处理器，又称显示核心、视觉处理器、显示芯片，是一种专门在个人电脑、工作站、游戏机和一些移动设备(如平板电脑、智能手机等)上作图像运算工作的微处理器。

目前深度学习实验室主流的显卡为GTX1080Ti，如图1所示。专业图像处理显卡TITANV如图2。



图1



图2

1.2 安装英伟达显卡驱动

根据机器的GPU型号，在NVIDIA官网搜索GPU对应的驱动的版本号。如图3所示。



图3

电脑重启之后，打开终端，输入：nvidia-smi，出现GPU信息表示驱动安装成功。

```
buu123@buu123: /usr/local/cuda-9.0/samples/1_Uutilities/deviceQuery
Device has ECC support: Disabled
Device supports Unified Addressing (UVA): Yes
Supports Cooperative Kernel Launch: Yes
Supports MultiDevice Co-op Kernel Launch: Yes
Device PCI Domain ID / Bus ID / location ID: 0 / 101 / 0
Compute Mode:
  < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 9.1, CUDA Runtime Version = 9.0, NumDevs = 1
Result = PASS
buu123@buu123: /usr/local/cuda-9.0/samples/1_Uutilities/deviceQuery$ nvidia-smi
Thu Jan  3 17:09:40 2019

+-----+
| NVIDIA-SMI 387.34                Driver Version: 387.34                |
+-----+-----+-----+-----+-----+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
|  0    Graphics Device      Off   | 00000000:65:00:0 | On           N/A     |
| 31%   45C    P8      27W / 250W | 298MiB / 12055MiB |      0%      Default |
+-----+-----+-----+-----+-----+-----+

+-----+
| Processes:                                                                  GPU Memory |
|  GPU       PID    Type   Process name                      Usage      |
+-----+-----+-----+-----+-----+-----+
|      0      1133     G   /usr/lib/xorg/Xorg                  166MiB     |
|      0      2007     G    compiz                          130MiB     |
+-----+-----+-----+-----+-----+-----+
buu123@buu123: /usr/local/cuda-9.0/samples/1_Uutilities/deviceQuery$
```

1.3 CUDA简介

CUDA(Compute Unified Device Architecture, 统一计算设备架构), 是显卡厂商NVIDIA在2007年推出的并行计算平台和编程模型。它利用图形处理器(GPU)能力, 实现计算性能的显著提高。**CUDA是一种由NVIDIA (独有) 推出的通用并行计算架构**, 该架构使GPU能够解决复杂的计算问题, 从而能通过程序控制底层的硬件进行计算。**它包含了CUDA指令集架构(ISA)以及GPU内部的并行计算引擎**。开发人员可以使用C/C++/11语言来为CUDA架构编写程序。CUDA提供host-device的编程模式以及非常多的接口函数和科学计算库, 通过同时执行大量的线程而达到并行的目的。

1.4 安装CUDA

下载好cuda 8.0的runfile安装包，安装包放在下载目录下。

- 1、cd到下载目录
- 2、终端输入下列代码sudo sh cuda_8.0.61_375.26_linux.run
- 3、终端开始安装，一直按空格，直到下面显示100%
- 4、输入accept，接受协议
- 5、输入n，不安装驱动，【这个地方一定输入n】 接下来的输入y或者default直接回车就行，具体如下图4：

```
Do you accept the previously read EULA?
accept/decline/quit:accept

Install NVIDIA Accelerated Graphics Driver for Linux-x86_64 375.26?
(y)es/(n)o/(q)uit: n

Install the CUDA 8.0 Toolkit?
(y)es/(n)o/(q)uit: y

Enter Toolkit Location
[ default is /usr/local/cuda-8.0 ]:

Do you want to install a symbolic link at /usr/local/cuda?
(y)es/(n)o/(q)uit: y

Install the CUDA 8.0 Samples?
(y)es/(n)o/(q)uit: y

Enter CUDA Samples Location
[ default is /home/heimu ]:

Installing the CUDA Toolkit in /usr/local/cuda-8.0
```

图4

1.5、cuDNN介绍

NVIDIA cuDNN是用于深度神经网络的GPU加速库。它强调性能、易用性和低内存开销。NVIDIA cuDNN可以集成到更高级别的机器学习框架中，如加州大学伯克利分校的流行CAFFE软件。简单的，插入式设计可以让开发人员专注于设计和实现神经网络模型，而不是调整性能，同时还可以在GPU上实现高性能现代并行计算。

1.6、安装cuDNN

1、下载相对应的cudnn版本

2、建立软连接

3 cuda Samples 测试

(1) cd 到切换到CUDA 8.0 Samples默认安装路径(即在NVIDIA_CUDA-8.0_Samples目录下)。

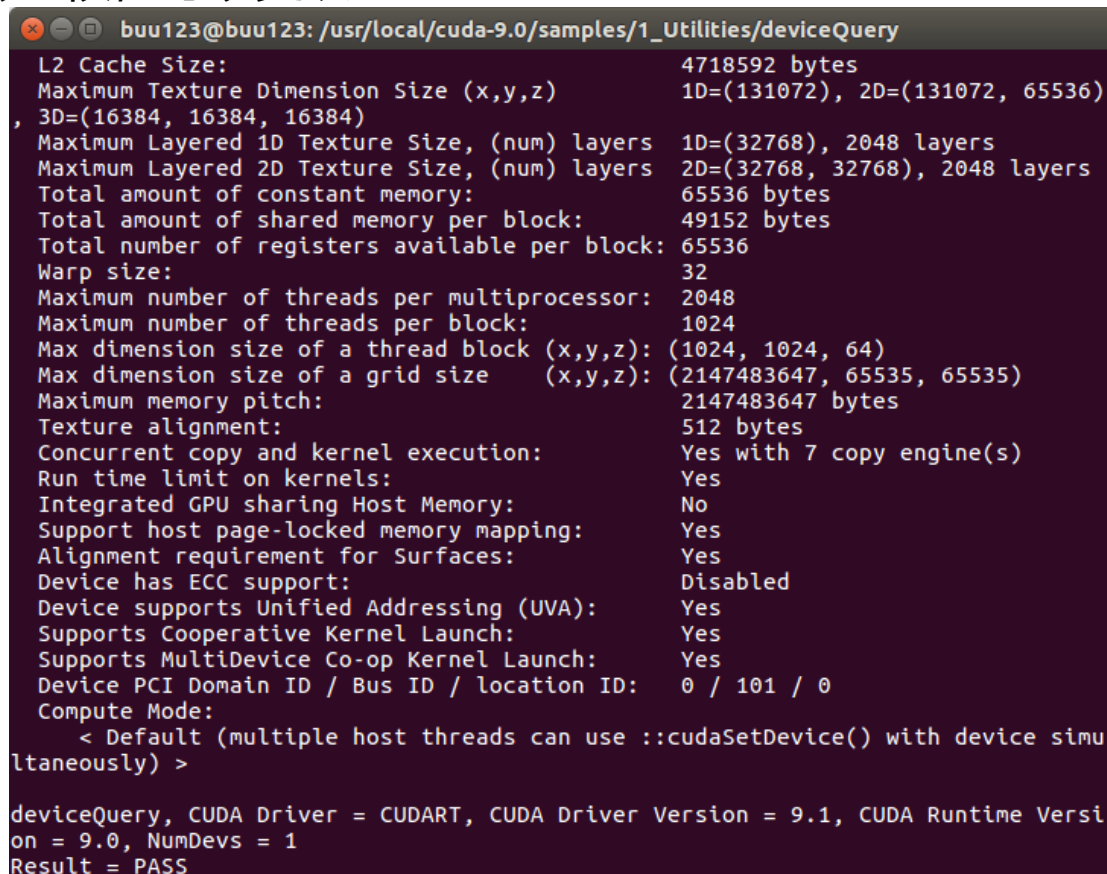
(2) 终端输入: sudo make all -j8 #8代表8核, 可以更改

(3) 完成后终端依次输入

cd bin/x86_64/linux/release

./deviceQuery

结果如下图, 表明cuda和cudnn配置成功



```
buu123@buu123: /usr/local/cuda-9.0/samples/1_Uutilities/deviceQuery
L2 Cache Size: 4718592 bytes
Maximum Texture Dimension Size (x,y,z) 1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 49152 bytes
Total number of registers available per block: 65536
Warp size: 32
Maximum number of threads per multiprocessor: 2048
Maximum number of threads per block: 1024
Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
Maximum memory pitch: 2147483647 bytes
Texture alignment: 512 bytes
Concurrent copy and kernel execution: Yes with 7 copy engine(s)
Run time limit on kernels: Yes
Integrated GPU sharing Host Memory: No
Support host page-locked memory mapping: Yes
Alignment requirement for Surfaces: Yes
Device has ECC support: Disabled
Device supports Unified Addressing (UVA): Yes
Supports Cooperative Kernel Launch: Yes
Supports MultiDevice Co-op Kernel Launch: Yes
Device PCI Domain ID / Bus ID / location ID: 0 / 101 / 0
Compute Mode:
< Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 9.1, CUDA Runtime Version = 9.0, NumDevs = 1
Result = PASS
```

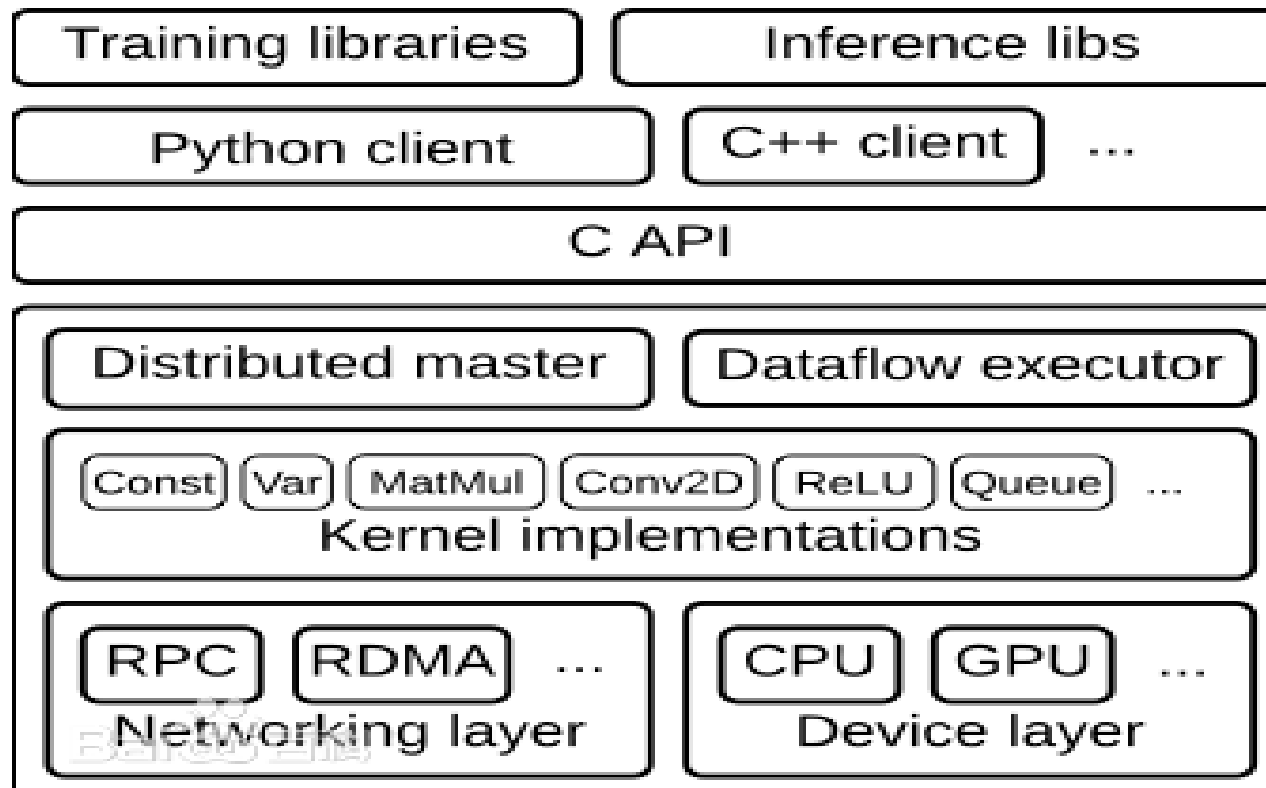
1.7 TensorFlow介绍

TensorFlow是一个基于数据流编程（dataflow programming）的符号数学系统，被广泛应用于各类机器学习（machine learning）算法的编程实现，其前身是谷歌的神经网络算法库DistBelief。

Tensorflow拥有多层次结构，可部署于各类服务器、PC终端和网页并支持GPU和TPU高性能数值计算，被广泛应用于谷歌内部的产品开发和各领域的科学研究。

1.7 TensorFlow介绍

分布式TensorFlow的核心组件 (core runtime) 包括: 分发中心 (distributed master)、执行器 (dataflow executor/worker service)、内核应用 (kernel implementation) 和最底端的设备层 (device layer) /网络层 (networking layer) 。



1.4安装TensorFlow

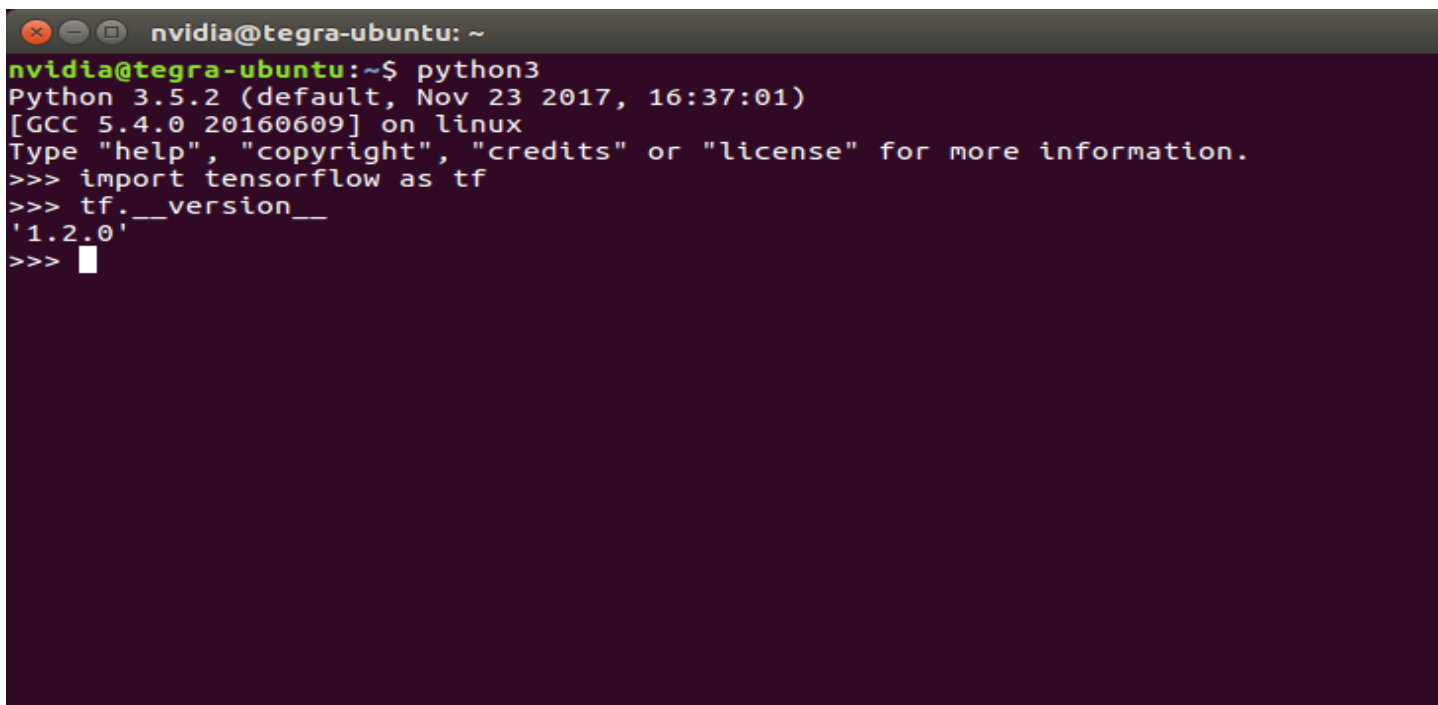
1、TensorFlow可以安装CPU和GPU两种版本，根据机器选择合适版本

CPU版本安装命令如下：sudo pip install tensorflow

GPU版本安装命令如下：sudo pip install tensorflow-gpu

2、测试结果

进入python编译环境，导入TensorFlow，输出版本号，如下图5所示。

A terminal window with a dark purple background and white text. The window title is 'nvidia@tegra-ubuntu: ~'. The user has entered 'python3' at the prompt. The terminal shows the Python 3.5.2 shell environment, including the GCC version and a prompt to type 'help', 'copyright', 'credits', or 'license'. The user then enters 'import tensorflow as tf' and 'tf.__version__', which outputs '1.2.0'.

```
nvidia@tegra-ubuntu: ~  
nvidia@tegra-ubuntu:~$ python3  
Python 3.5.2 (default, Nov 23 2017, 16:37:01)  
[GCC 5.4.0 20160609] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import tensorflow as tf  
>>> tf.__version__  
'1.2.0'  
>>> 
```

图5

1.4安装其他依赖

Numpy-- Python的一种开源的数值计算扩展

Scipy--构建于NumPy之上，提供了一个用于在Python中进行科学计算的工具集

Pillow-- Python图片处理模块

Cython--Python支持C语言扩展的编译器

Matplotlib-- Python 的 2D绘图库，它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形

Keras--纯Python编写而成并基Tensorflow、Theano以及CNTK后端

Opencv--一个基于BSD许可（开源）发行的跨平台计算机视觉库



















January 5 2019

/02

下载SSD网络源码，修改参数

2.1 下载SSD源码

搭建SSD框架，<https://github.com/balancap/SSD-Tensorflow>
下载解压。

 checkpoints	2019-01-03 下午...	文件夹	
 datasets	2019-01-03 下午...	文件夹	
 demo	2019-01-03 下午...	文件夹	
 deployment	2019-01-03 下午...	文件夹	
 nets	2019-01-03 下午...	文件夹	
 notebooks	2019-01-03 下午...	文件夹	
 pictures	2019-01-03 下午...	文件夹	
 preprocessing	2019-01-03 下午...	文件夹	
 tf_extended	2019-01-03 下午...	文件夹	
 .gitignore	2017-04-10 下午...	GITIGNORE 文件	1 KB
 caffe_to_tensorflow.py	2017-04-10 下午...	JetBrains PyChar...	3 KB
 COMMANDS.md	2017-04-10 下午...	MD 文件	12 KB
 eval_ssd_network.py	2017-04-10 下午...	JetBrains PyChar...	16 KB
 inspect_checkpoint.py	2017-04-10 下午...	JetBrains PyChar...	5 KB
 README.md	2017-04-10 下午...	MD 文件	10 KB
 tf_convert_data.py	2017-04-10 下午...	JetBrains PyChar...	2 KB
 tf_utils.py	2017-04-10 下午...	JetBrains PyChar...	10 KB
 train_ssd_network.py	2017-04-10 下午...	JetBrains PyChar...	18 KB

2.2 采集与标注数据

- (1) 用摄像机采集数据
- (2) 使用Labellmg标注数据，标注格式如图6。

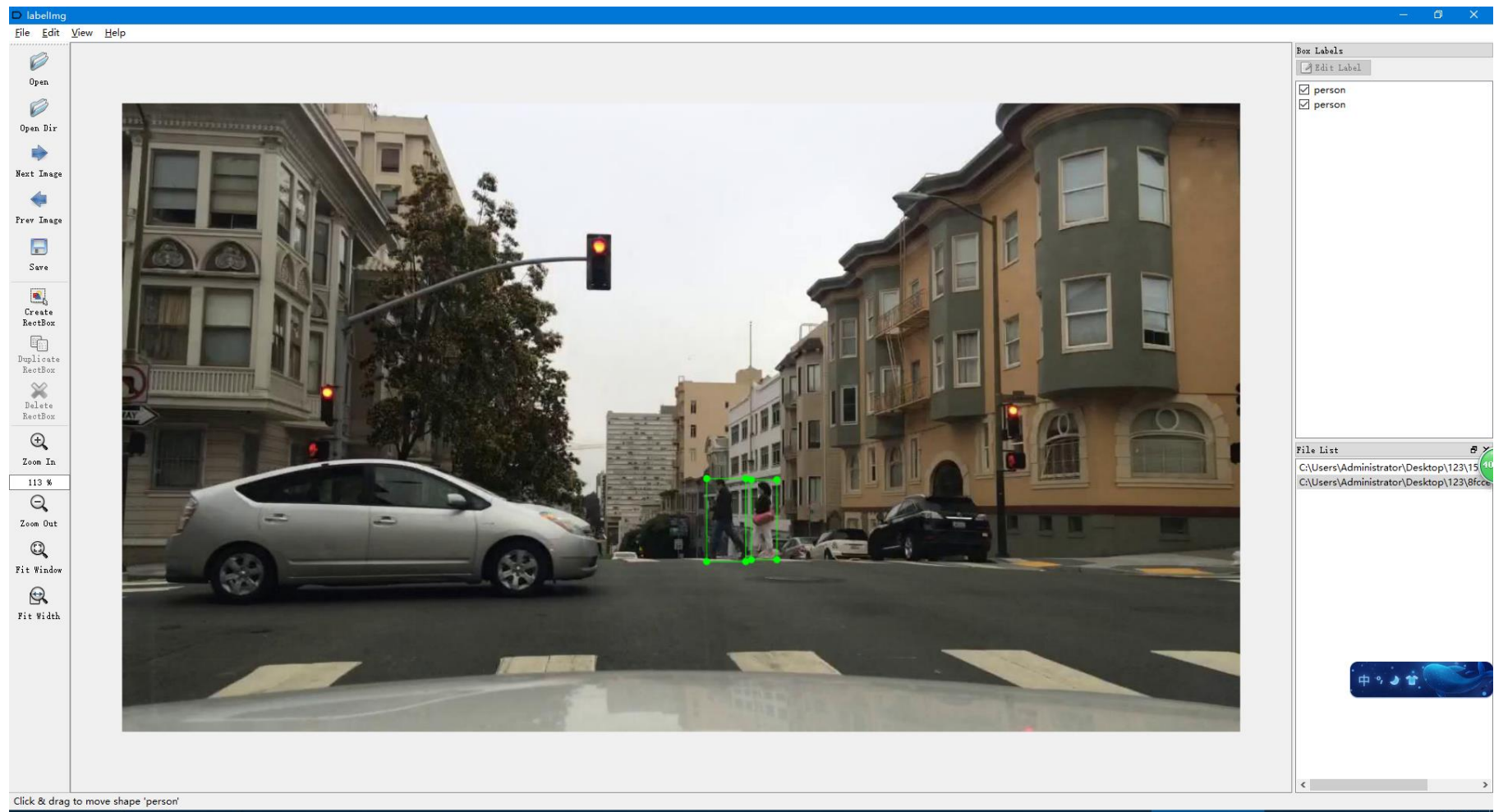


图6

2.3修改网络参数

1、下载pascalvoc数据，自己的数据根据voc格式改写，如图7所示。

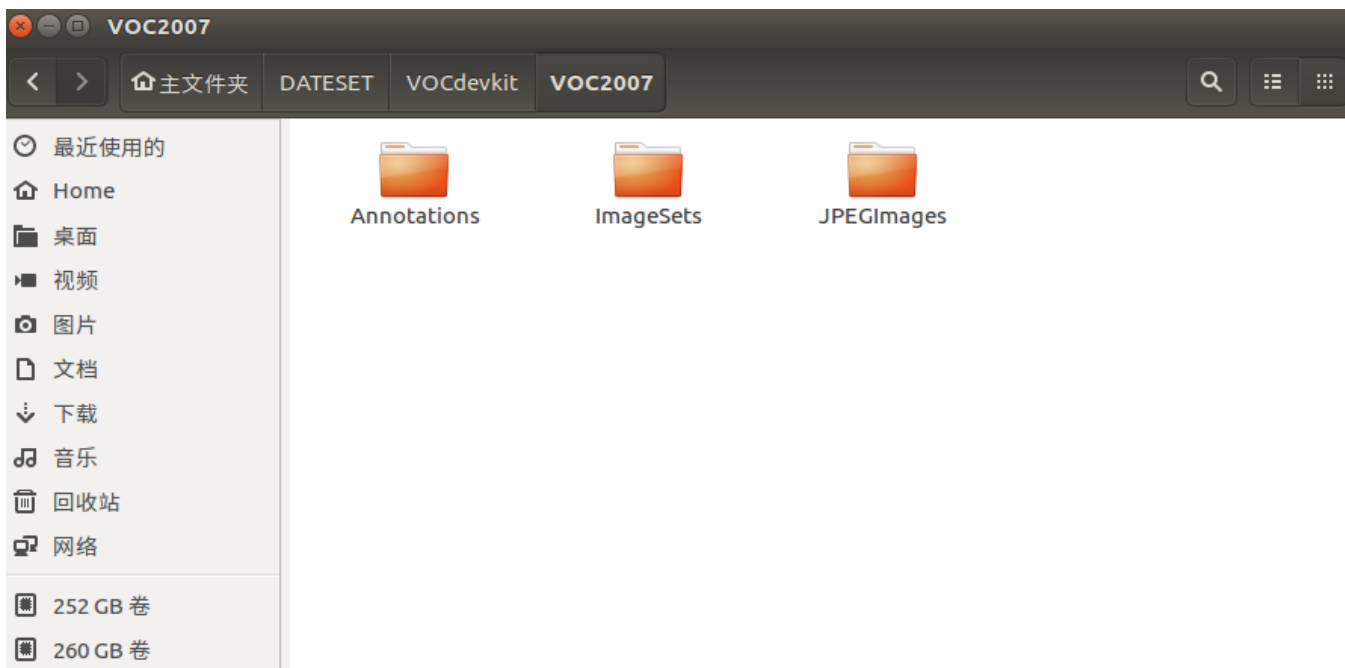


图7

2.3修改网络参数

2、标记数据，生成txt文件，train.txt, trainval.txt, test.txt, val.txt

3、修改datasets文件夹中pascalvoc_common.py文件，将训练类修改别成自己的

#原始的

```
# VOC_LABELS = {  
#   'none': (0, 'Background'),  
#   'aeroplane': (1, 'Vehicle'),  
#   'bicycle': (2, 'Vehicle'),  
#   'bird': (3, 'Animal'),  
#   'boat': (4, 'Vehicle'),  
#   'bottle': (5, 'Indoor'),  
#   'bus': (6, 'Vehicle'),  
#   'car': (7, 'Vehicle'),  
#   'cat': (8, 'Animal'),  
#   'chair': (9, 'Indoor'),  
#   'cow': (10, 'Animal'),  
#   'diningtable': (11, 'Indoor'),  
#   'dog': (12, 'Animal'),  
#   'horse': (13, 'Animal'),  
#   'motorbike': (14, 'Vehicle'),  
#   'person': (15, 'Person'),  
#   'pottedplant': (16, 'Indoor'),  
#   'sheep': (17, 'Animal'),  
#   'sofa': (18, 'Indoor'),  
#   'train': (19, 'Vehicle'),  
#   'tvmonitor': (20, 'Indoor'),  
# }
```

#修改后的

```
VOC_LABELS = {  
    'none': (0, 'Background'),  
    'pantograph': (1, 'Vehicle'),  
}
```

2.3修改网络参数

4、将图像数据转换为tfrecords格式

SSD-Tensorflow-master→datasets→pascalvoc_to_tfrecords.py 。 。 。 然后更改文件的**83**行读取方式为'**rb**') 如果你的文件不是.png格式，也可以修改图片的类型。

```
# Read the image file.  
filename = directory + DIRECTORY_IMAGES + name + '.png'  
image_data = tf.gfile.FastGFile(filename, 'rb').read()
```

2.3修改网络参数

在SSD-Tensorflow-master文件夹下创建tf_convert_data.sh

```
#!/bin/bash
```

```
#this is a shell script to convert pascal VOC datasets into tf-records only
```

```
#directory where the original dataset is stored
```

```
DATASET_DIR=/home/buu123/Model/SSD/SSD-Tensorflow-
```

```
master/VOCdevkit/VOC2007/    #VOC数据保存的文件夹（VOC的目录格式未改变）
```

```
#output directory where to store TFRecords files
```

```
OUTPUT_DIR=/home/buu123/Model/SSD/SSD-Tensorflow-
```

```
master/VOCdevkit/VOC2007/SegmentationClass  #自己建立的保存tfrecords数据的文件夹
```

```
python ./tf_convert_data.py \
```

```
--dataset_name=pascalvoc \
```

```
--dataset_dir=${DATASET_DIR} \
```

```
--output_name=voc_2007_train \
```

```
--output_dir=${OUTPUT_DIR}
```

在终端运行tf_convert_data.sh文件

2.3修改网络参数

train_ssd_network.py 修改第 154 行的最大训练步数，将 None 改为比如 50000。

(tf.contrib.slim.learning.training函数中max-step为None时训练会无限进行。)

- A. nets/ssd_vgg_300.py (因为使用此网络结构)，修改87 和88行的类别
- B. train_ssd_network.py,修改类别120行，GPU占用量，学习率，batch_size等
- C. eval_ssd_network.py 修改类别，66行
- D. datasets/pascalvoc_2007.py 根据自己的训练数据修改整个文件

2.4训练模型

1、新建train.sh文件

输入：

```
#!/bin/bash DATASET_DIR=./tfrecords ###训练集转化成tfrecords存储的路径
TRAIN_DIR=./logs/ ###存储训练结果的路径，包括checkpoint和event，自行指定 CHECKPOINT_PATH=./checkpoints/vgg_16.ckpt ###下载vgg_16模型
python train_ssd_network.py \
--train_dir=${TRAIN_DIR} \
--dataset_dir=${DATASET_DIR} \
--dataset_name=xxxxxx \ ###具体指定，改为your_data_name，如果你在前面搞清楚了如何转换自己的数据的话
--dataset_split_name=train \
--model_name=ssd_300_vgg \
--num_classes=4
--checkpoint_path=${CHECKPOINT_PATH} \ --checkpoint_model_scope=vgg_16 \ ##### 改 为 vgg_16 --
checkpoint_exclude_scopes=ssd_300_vgg/conv6,ssd_300_vgg/conv7,ssd_300_vgg/block8,ssd_300_vgg/block9,ssd_300_vgg/block10,ssd_300_vgg/block11,
ssd_300_vgg/block4_box,ssd_300_vgg/block7_box,ssd_300_vgg/block8_box,ssd_300_vgg/block9_box,ssd_300_vgg/block10_box,ssd_300_vgg/block11_box \
trainable_scopes=ssd_300_vgg/conv6,ssd_300_vgg/conv7,ssd_300_vgg/block8,ssd_300_vgg/block9,ssd_300_vgg/block10,ssd_300_vgg/block11,ssd_300_v
gg/block4_box,ssd_300_vgg/block7_box,ssd_300_vgg/block8_box,ssd_300_vgg/block9_box,ssd_300_vgg/block10_box,ssd_300_vgg/block11_box \ --
save_summaries_secs=60 \
--save_interval_secs=600 \
--weight_decay=0.0005 \
--optimizer=adam \
--learning_rate=0.001 \
--learning_rate_decay_factor=0.94 \
--batch_size=32
在SSD目录下打开终端，执行train.sh，开始训练。
```

January 5 2019

/03

目标检测SSD网络demo展示

目标检测SSD网络demo展示

训练完成之后模型文件会存储到指定文件夹中，如图8所示

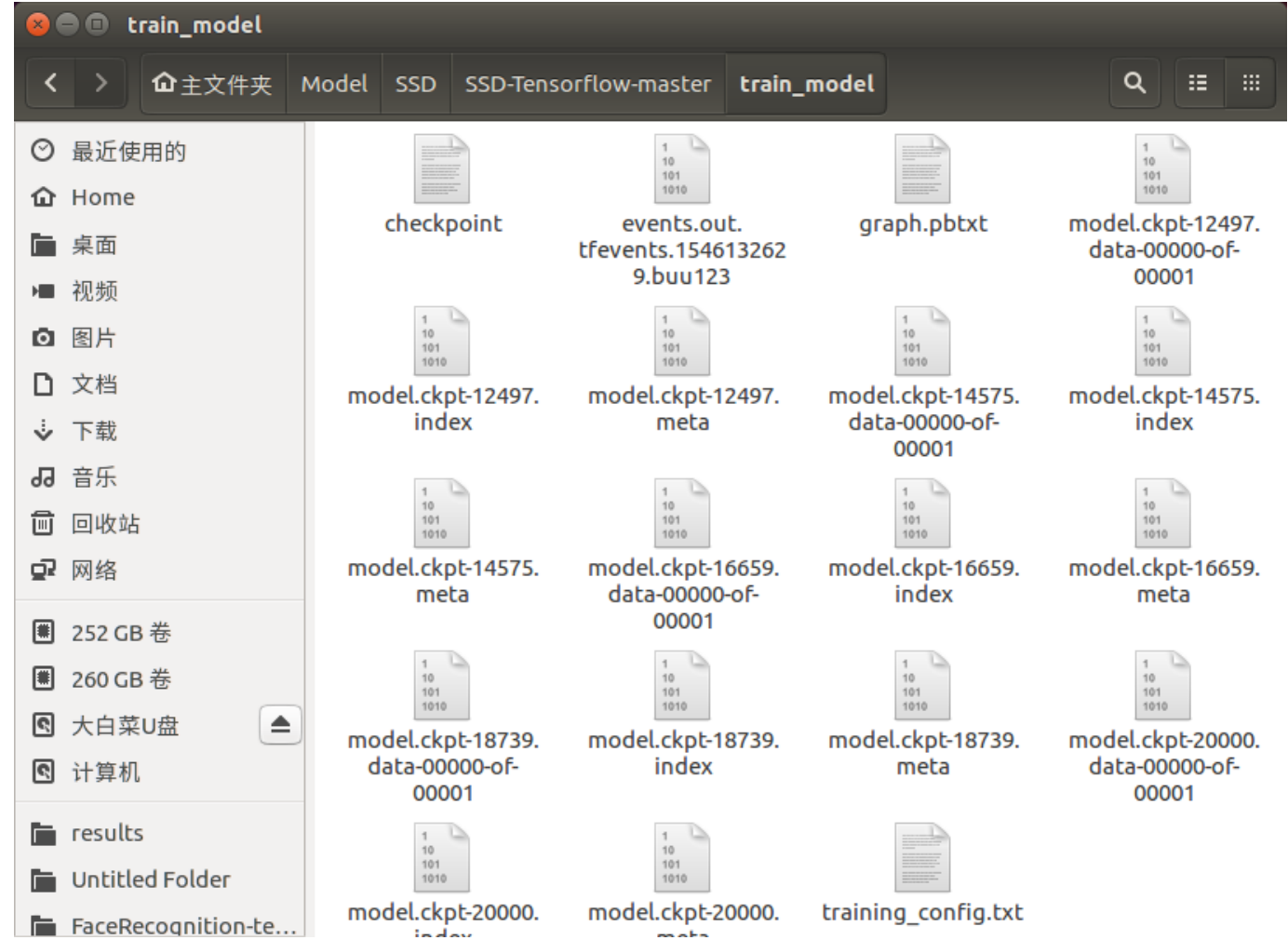


图8

目标检测SSD网络demo展示

预测图片结果如图9所示



图9

January 5 2019

/04

智能物流车行人检测演示

The background features a series of overlapping, wavy, translucent blue bands that create a sense of motion and depth. These bands originate from the left side and curve towards the right, eventually fading into a light blue gradient. The overall effect is clean, modern, and professional.

Thanks