[一、安装 3](#_Toc1508806176)

[1、安装CUDA(8.0) 3](#_Toc1137594678)

[（1）我这里选择runfile安装 4](#_Toc285481584)

[（2）选择”.rpm”安装 5](#_Toc1561752884)

[（3）解决方法 5](#_Toc552695257)

[（4）添加环境变量 7](#_Toc791487411)

[（5）解决一直安装默认仓库CUDA的方法 8](#_Toc1523806632)

[（6）验证CUDA是否安装成功 8](#_Toc632460803)

[cd /usr/local/cuda-8.0/samples 8](#_Toc2020756018)

[2、安装CUDNN 9](#_Toc1531808123)

[3、安装OPENCV 10](#_Toc542546360)

[a）安装依赖 10](#_Toc807271553)

[b）编译opencv 11](#_Toc1524486331)

[C）参考网址 13](#_Toc1108000541)

[d）总结 13](#_Toc2146412950)

[4、安装tensorflow-gpu 13](#_Toc1618537094)

[5、安装Caffe 15](#_Toc1154226785)

[二、使用Yolo 18](#_Toc831765164)

[1、安装Yolo 18](#_Toc1178853290)

[（1）下载源码 18](#_Toc1413467539)

[（2）下载权重文件 18](#_Toc758968817)

[（3）测试 19](#_Toc1022211792)

[（4）测试视频问题 20](#_Toc504742283)

[（5）修改代码只显示一类目标 20](#_Toc1767108775)

[E）--->重新编译和测试 24](#_Toc2074356369)

[2、训练自己的数据集 24](#_Toc630826704)

[四YOLO原理理解 25](#_Toc595575317)

[1、源码结构 25](#_Toc576230639)

[三、杂记 35](#_Toc1333729061)

[1、测试yolov2-tensorflow 35](#_Toc1937903407)

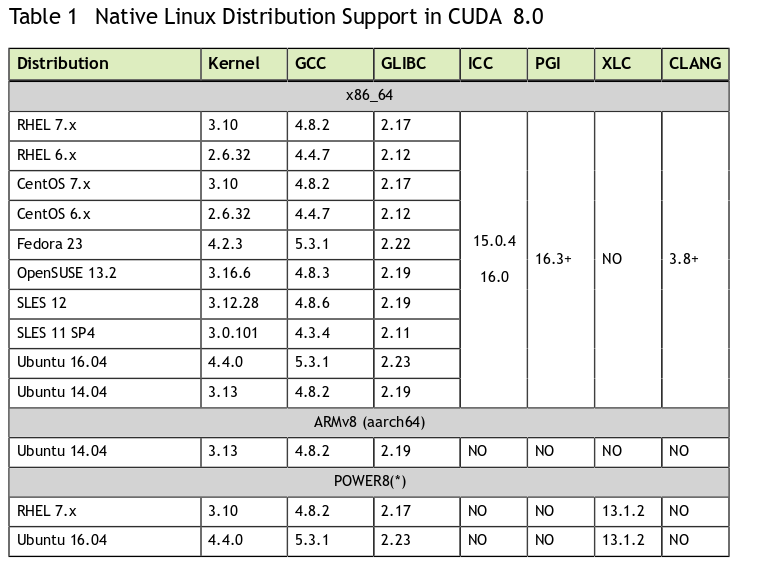
[2、QT配置 36](#_Toc88393560)

**Yolo有关**

# 一、安装

环境：centos7.3 + Quadro K620 + 32G内存

## 1、安装CUDA(8.0)



查看本机内核需要安装的一些依赖

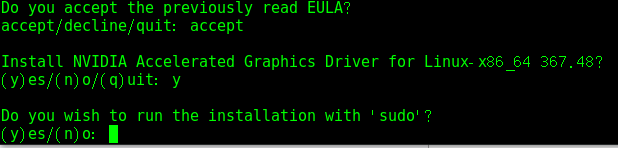
下载CUDA网址：

<https://developer.nvidia.com/cuda-toolkit-archive>

<https://developer.nvidia.com/cuda-downloads>

参考安装网址：  
<https://www.jianshu.com/p/59f80d77d705>

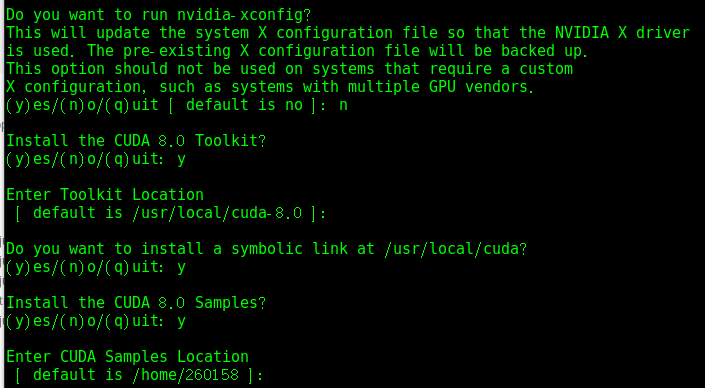
### （1）我这里选择runfile安装

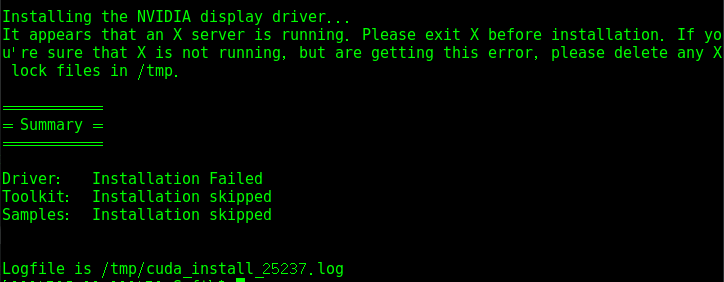


2017-12-30 09-48-47 的屏幕截图

2017-12-30 09-56-36 的屏幕截图

1. 接受协议？是
2. 同意安装驱动不？同意
3. 是否需要用sudo命令？用
4. 安装驱动需要用sudo命令？如果上一步选择是，则不会有这个问题。会提示你输密码。
5. 是否要安装OpenGL，一定不要安装。





按照以上的选择安装失败。

### （2）选择”.rpm”安装

这种方式也安装的不是我下载的。

解决以上的问题，删除NuxDextop仓库

vi /etc/yum.repos.d/nux-dextop.repo

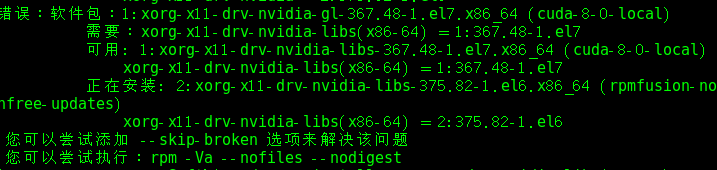
将文件中的enabled=1改成enabled=0，然后保存。

参考网址：

<http://blog.csdn.net/yaoqiwaimai/article/details/77163261>

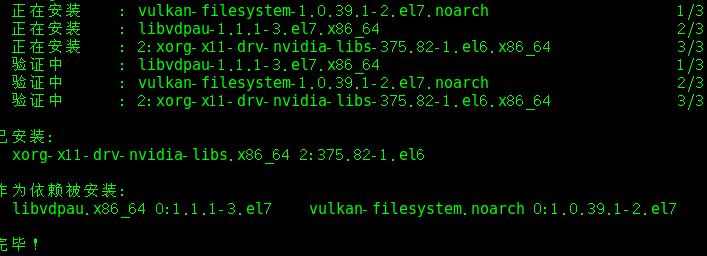
### （3）解决方法

会出现以下问题



解决办法1：

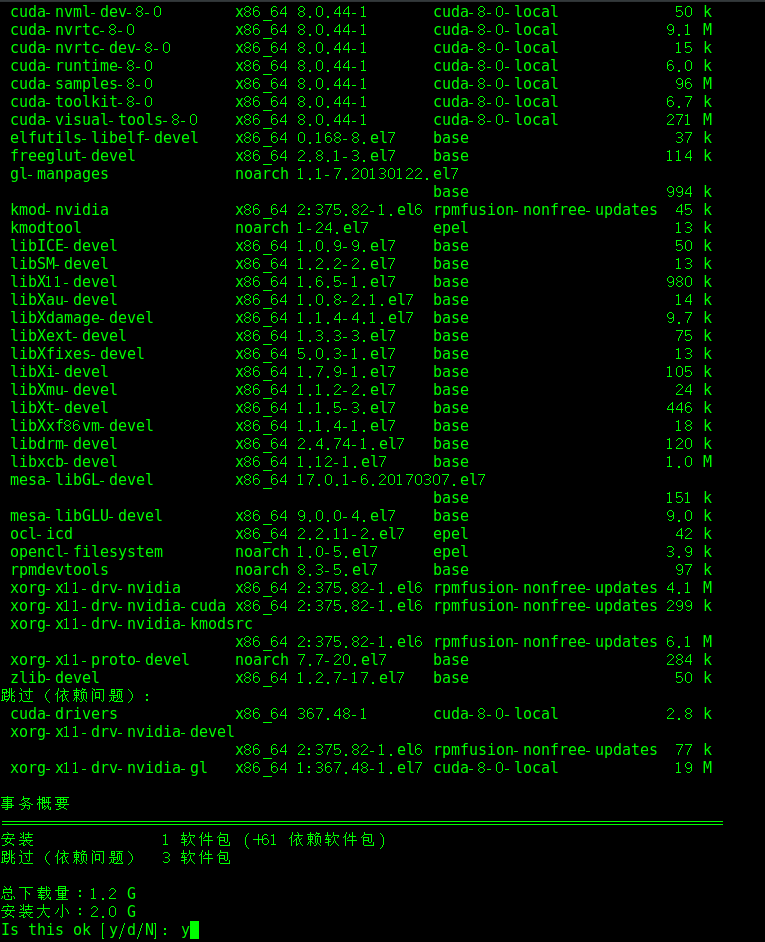
2017-12-30 16-59-20 的屏幕截图



事实证明如上办法不行。

解决办法2：

Sudo yum install --skip -broken

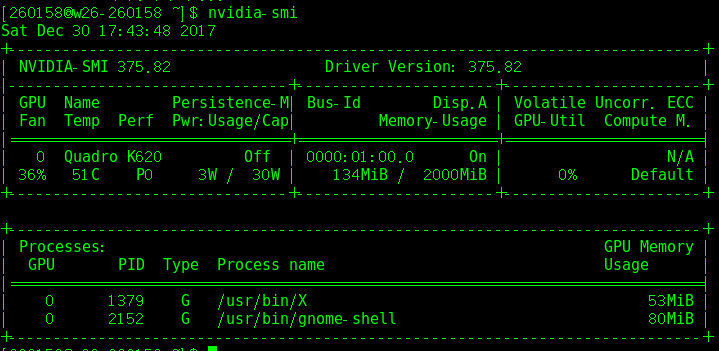


然后安装成功。

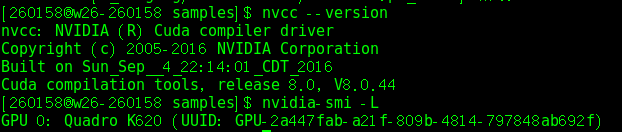
运行命令：nvidia-smi

结果：找不到命令

重启后，运行命令nvidia-smi



驱动加载成功。



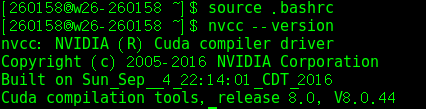
### （4）添加环境变量

添加环境变量：在～/ 目录下的.bashrc文件中添加

export CUDA\_HOME=/usr/local/cuda-8.0

export LD\_LIBRARY\_PATH=/usr/local/cuda-8.0/lib64:$LD\_LIBRARY\_PATH

export PATH=/usr/local/cuda-8.0/bin:$PATH



### （5）解决一直安装默认仓库CUDA的方法

参考网址：

<http://blog.csdn.net/yaoqiwaimai/article/details/77163261>

Vi /etc/yum.repos.d/nux-dextop.repo

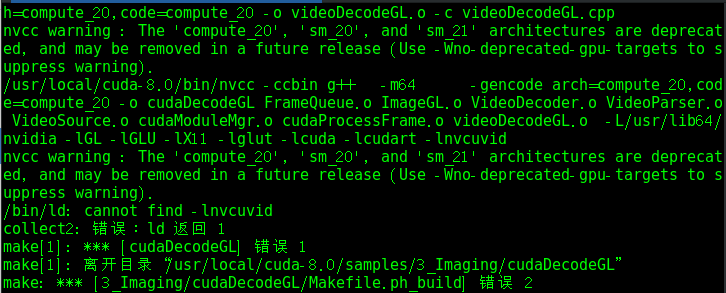
将文件中的enabled=1改成enabled=0，然后保存。

成功解决。

### （6）验证CUDA是否安装成功

## cd /usr/local/cuda-8.0/samples

Sudo make

出现以下的错误：  


目测是没有编译成功。待解决

测试cuda是否安装成功

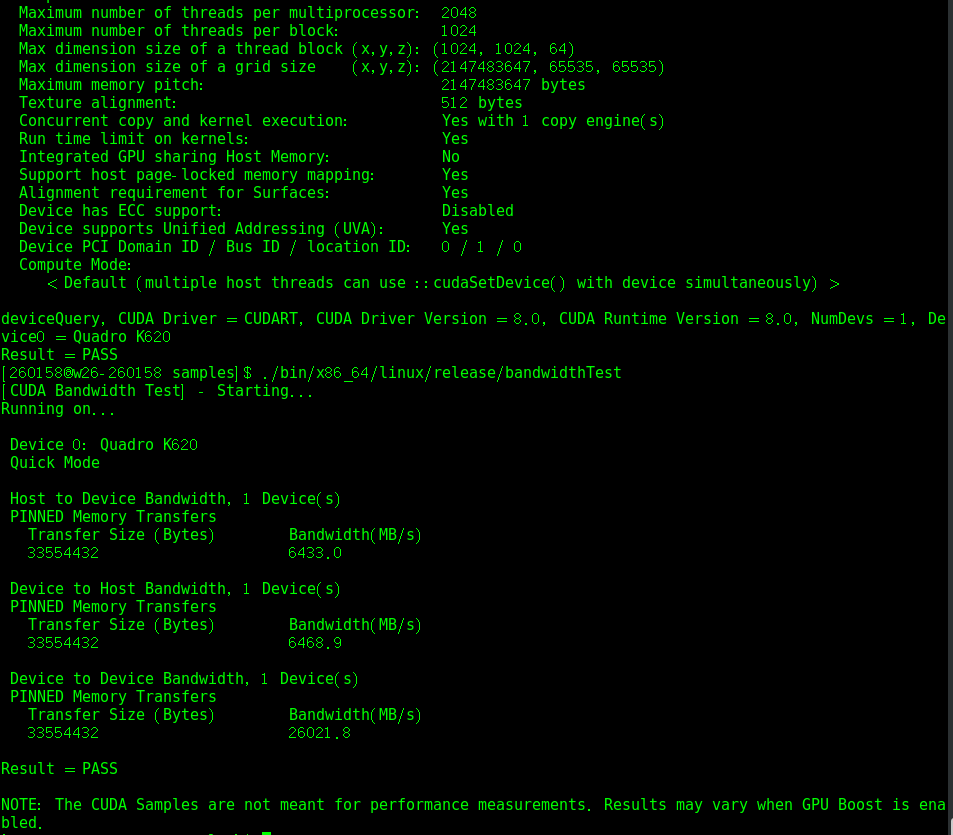
# cd /usr/local/cuda/samples/

# make

# ./bin/x86\_64/linux/release/deviceQuery

# ./bin/x86\_64/linux/release/bandwidthTest

显示 Result = PASS 则测试通过



虽然编译没有通过，但按照网上的资料，还是安装成功了。

### （7）禁用Nouveau kernel driver

Nvidia驱动下载

<http://www.nvidia.com/Download/index.aspx?lang=cn>

在安装cuda的时候，由于涉及到NVIDIA驱动的安装，使得nouveau驱动与NVIDIA驱动冲突，为了能够继续安装，必须禁用此驱动。禁用步骤如下：

把nouveau驱动加入黑名单，

即在/etc/modprobe.d/blacklist.conf的后面加入：

blacklist nouveau

options nouveau modset=0



## 2、安装CUDNN

下载网址：

<https://developer.nvidia.com/rdp/cudnn-download>

cuDNN是 CUDA的一个深度网络加速库，也是深度学习框架所需要的。安装cuDNN的过程实际上是下载cuDNN库并拷贝到CUDA的目录中。下载cuDNN库后，进入下载目录执行以下命令即完成安装。

tar xzvf cudnn-8.0-linux-x64-v6.tgz **#是6版本的**

sudo cp -P cuda/include/cudnn.h /usr/local/cuda/include

sudo cp -P cuda/lib64/lib\* /usr/local/cuda/lib64

sudo chmod a+r /usr/local/cuda/include/cudnn.h

sudo chmod a+r /usr/local/cuda/lib64/lib\*

因为没有软连接，所以下面的tensorflow报错。

设置软连接

cd /usr/local/cuda/lib64/

sudo rm -rf libcudnn.so libcudnn.so.7 #删除原有动态文件

sudo ln -s libcudnn.so.6.0.21 libcudnn.so.6 #生成软链接

sudo ln -s libcudnn.so.6 libcudnn.so #生成软链接

注意！！软链接中的数字6.0.21等参照文件夹lib64中原有的软链接名

sudo ldconfig

## 3、安装OPENCV

我安装的版本是2.4.10，安装教程去官网查找，需要安装依赖。

网址：<https://opencv.org/>

步骤

### a）安装依赖

--->  添加源

sudo yum -y install epel-release

--->gcc,g++等这些基本的工具,一般都有，没有的话再安装

sudo yum -y install gcc gcc-c++ cmake python-devel numpy

--->然后是opencv的依赖项

sudo yum -y install gtk2-devel gimp-devel gimp-devel-tools gimp-help-browser zlib-devel libtiff-devel libjpeg-devel libpng-devel libavc1394-devel libraw1394-devel libdc1394-devel jasper-devel jasper-utils swig python libtool nasm libv4l-devel gstreamer-plugins-base-devel

--->没有安装成功的包：

2017-12-30 18-56-51 的屏幕截图

Notes：安装的时候需要一大堆依赖（宁可信其有），且官网ubuntu的依赖和centos的依赖有所区别，要注意。

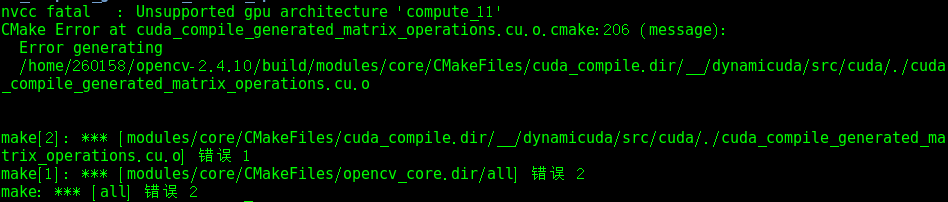
### b）编译opencv

我这里是已经安装成功CUDA后，开始编译OPENCV，所有中间会与没有装CUDA的有些出入。耗时也比较长。

按照官网步骤来

<https://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html>

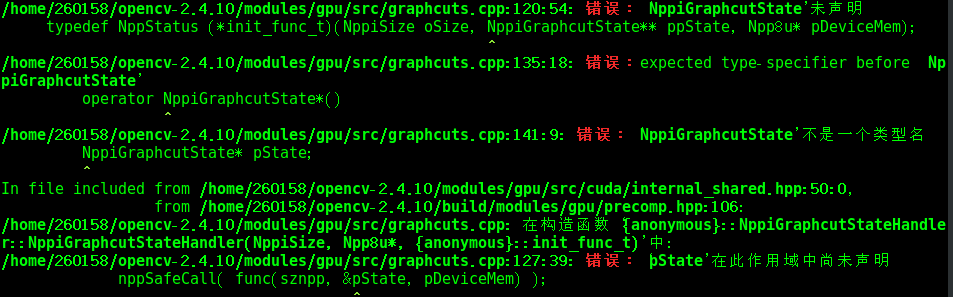
--->make错误1

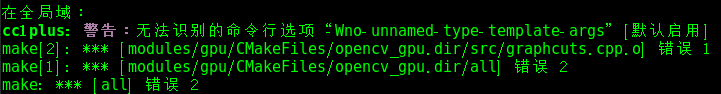


解决办法：

cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local -D CUDA\_GENERATION=Kepler ..

--->make错误2





解决办法：

cuda8.0较新，opencv-2.4.11较早，要编译通过需要修改源码：

修改/data/opencv-2.4.11/modules/gpu/src/graphcuts.cpp

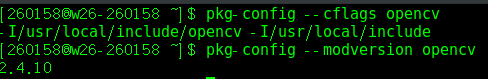
将 #if !defined (HAVE\_CUDA) || defined (CUDA\_DISABLER)

改为

#if !defined (HAVE\_CUDA) || defined (CUDA\_DISABLER) || (CUDART\_VERSION >= 8000)

重新编译即可。

安装成功



### C）参考网址

<http://blog.csdn.net/kakitgogogo/article/details/52490010>

<https://www.cnblogs.com/hxbbing/p/4894813.html>

### d）总结

注意: Anconda 自带是python-opencv，不属于opencv源码库， 加载其路径是不可以的，必须要编译opencv源码，生成库文 件和添加环境变量。参考以下网址：

<http://www.cnblogs.com/wyuzl/p/7889007.html>

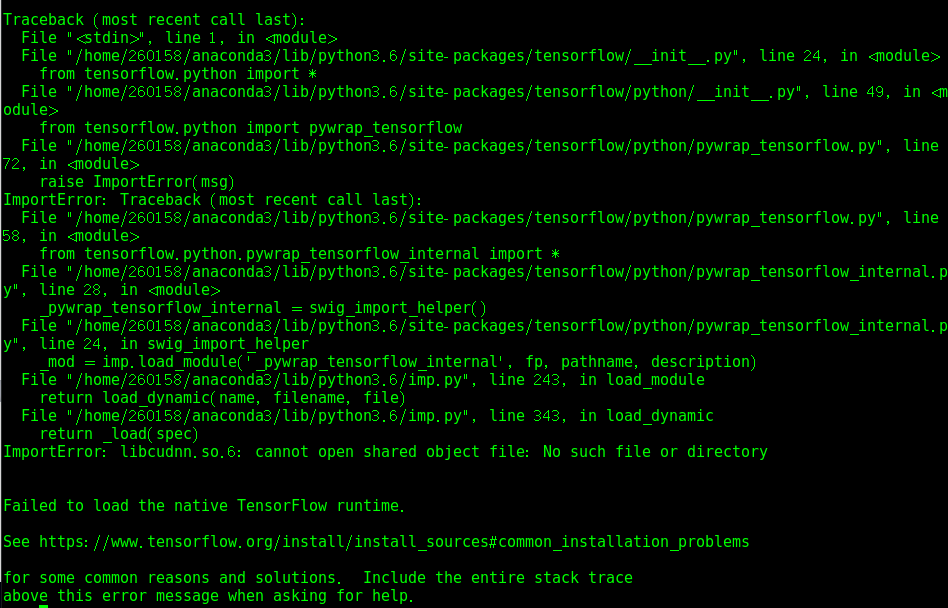
## 4、安装tensorflow-gpu

命令：pip install tensorflow

装的是最新版本的1.4.1

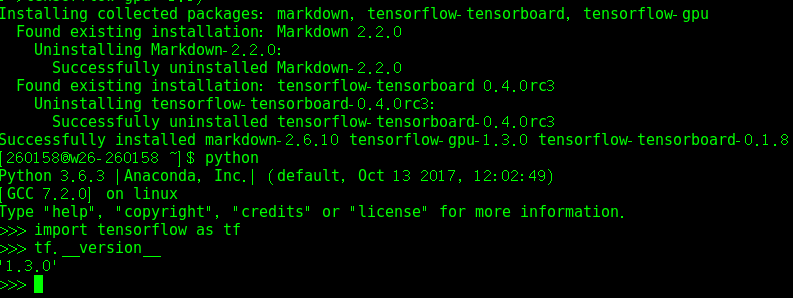
2018-01-02 13-51-02 的屏幕截图

加载时出现错误



截至2017.08.15 Tensorflow 尚不支持cuda9.0，所以建议用cuda 8.0 + cuDnn 6.0，这也是Tensorflow 1.3的缺省配置。

我现在改成tensorflow1.3，和上面对应的默认配置。

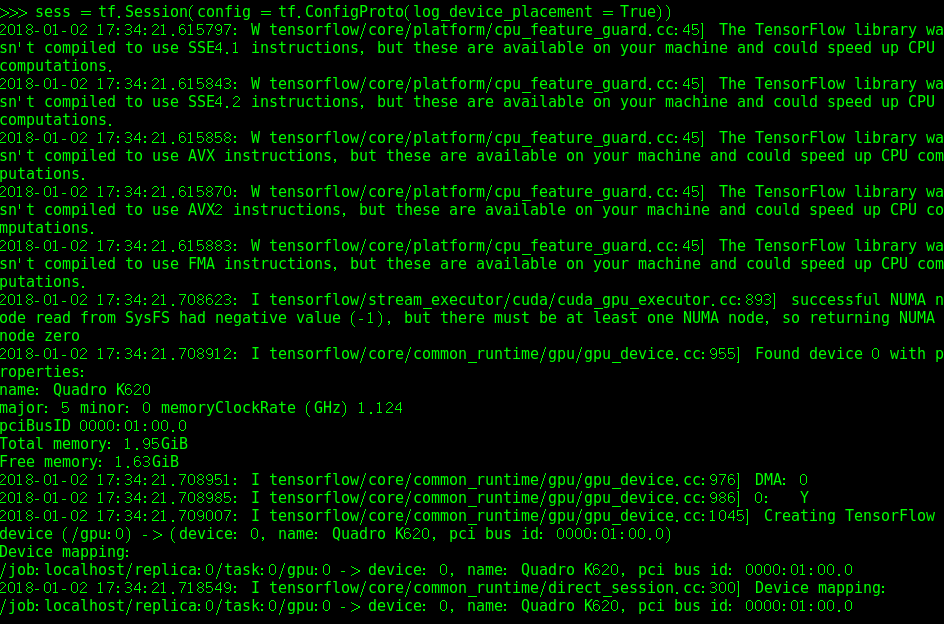


安装成功。

查看是否安装的是GPU版本

import tensorflow as tf

sess = tf.Session(config=tf.ConfigProto(log\_device\_placement=True))



安装成功。

## 5、安装Caffe

参考网址：

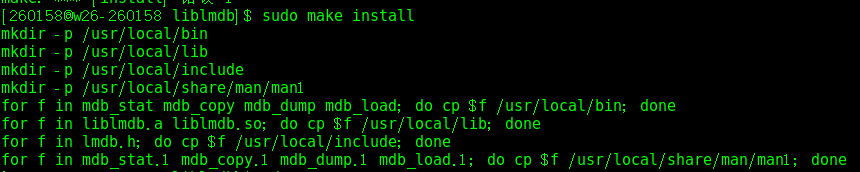
<http://2014.54chen.com/blog/2017/07/04/how-to-build-caffe-1-dot-0-with-anaconda-python3/>

没遇到问题，就是编译caffe的时候要注意根据自己电脑的安装配置去进行选择。

这个没有安装pycaffe。

（1）安装lmdb

git clone [https://github.com/LMDB/lmdb](https://github.com/LMDB/lmdb" \t "/home/260158/文档\\x/_blank)   
cd lmdb/libraries/liblmdb   
make && make install



（2）安装protobuf

参考网址：

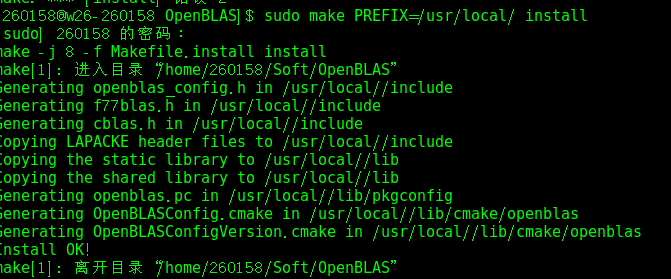
<http://www.linuxidc.com/Linux/2016-12/138716.htm>

我没有make check ，因为总是出错。

如果有错误，就用sudo命令。

（3）安装OpenBLAS

tar -xzvf OpenBLAS.tar.gz   
make   
make PREFIX=/usr/local/ install   
若在caffe编译的时候出现/usr/bin/ld: cannot find -lopenblas   
请在openblas编译文件夹下执行cp lib\* /usr/local/lib即可



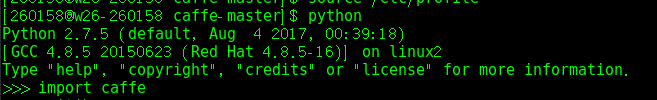
（4）以上都是之前的尝试

参考网址：

<http://blog.csdn.net/yimu_zhang/article/details/78578155>

这个是可以安装成功的

我用的是电脑自带的python2.7，没用anaconda。



## 6、安装labelImg

我用的是anaconda4.2，默认pyrhon3.6，没有安装成功。

然后切换python的环境变量，用本机自带的python2.7.5。然后去安装labelImg。

参考网址：

<http://blog.csdn.net/dgatiger/article/details/50331361>

<https://www.cnblogs.com/new-age/p/7071289.html>

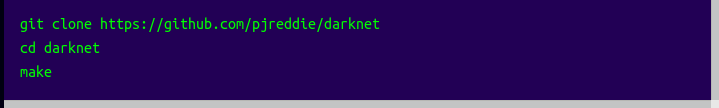
这个是安装pyqt4，然后安装源码安装labelImg。

# 二、使用Yolo

## 1、安装Yolo

官方网址：<https://pjreddie.com/darknet/yolo/>

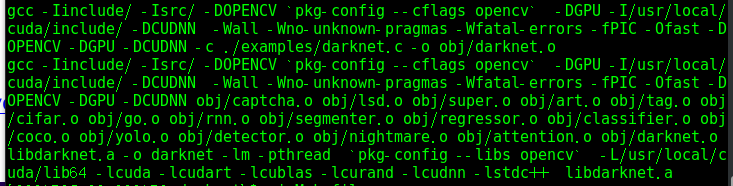
### （1）下载源码



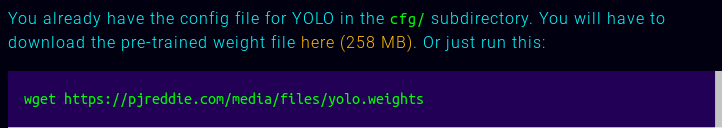
修改Makefile，根据自己的配置修改。将对应的0改为1。



然后进行编译,编译成功。

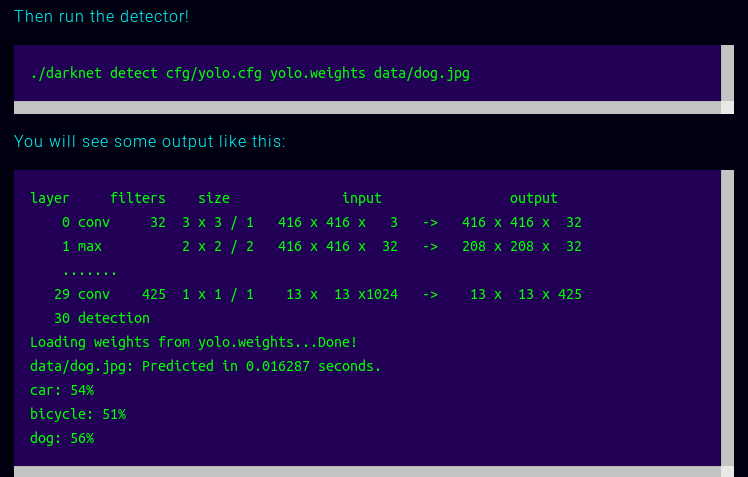


### （2）下载权重文件



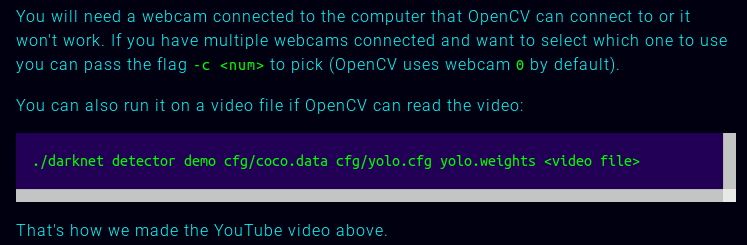
### （3）测试

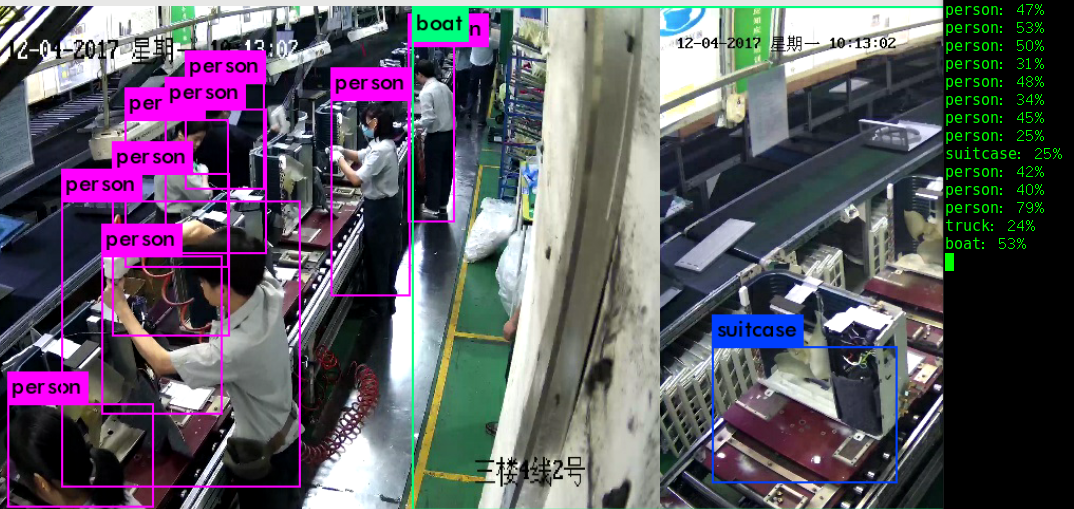
测试图片：



测试成功。

测试视频：





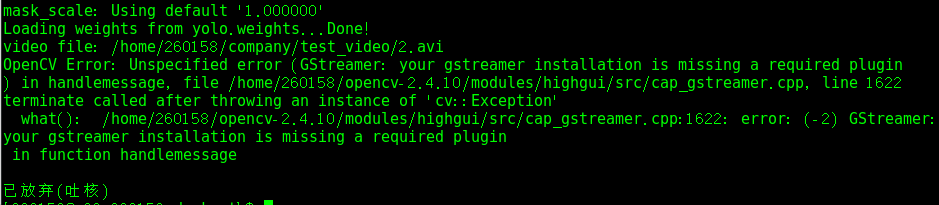
测试成功。

### （4）测试视频问题



后3个视频可以正常的读出来，但前3个视频是不可以的。可能是因为我后3个视频是自己用OPENCV生成的，而前3个是直接下载的。

错误如下：



在用opencv处理过后，就可以读取视频了。

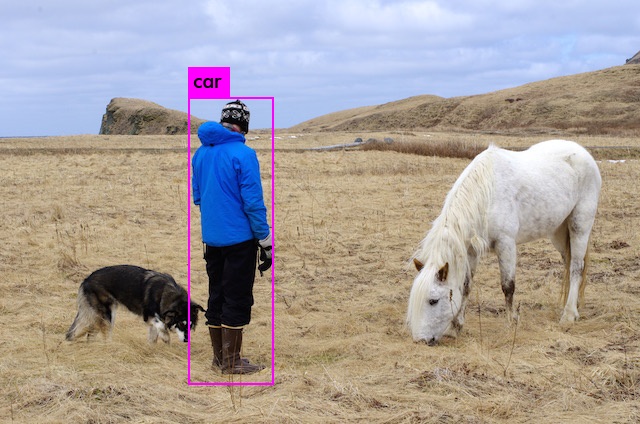
### （5）修改代码只显示一类目标

A）修改cfg/voc.data or coco.data

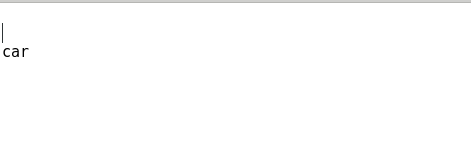
classess=20  or classess=80  改成 classes = 1

Data/\*.names 中的名称是目标框的名称。我把coco.names里改为car。结果如下：

2018-01-06 15-55-04 的屏幕截图



如果我把coco.names改成这样子：



就会变成如下这样：



说明这个文件中的内容和识别目标类别是没关系的。

B）修改examples/detector.c

void run\_detector(int argc, char \*\*argv)//该函数中，倒数第三行.line=5422

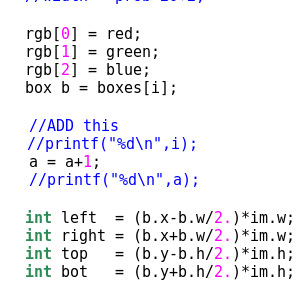
int classes = option\_find\_int(options, "classes", 1);//最后一个参数修改成1

void test\_detector(char \*datacfg, char \*cfgfile, char \*weightfile, char \*filename, float thresh)函数中5

draw\_detections(im, l.w\*l.h\*l.n, thresh, boxes, probs, names, alphabet, 1); //最后一个参数修改成1。line=481

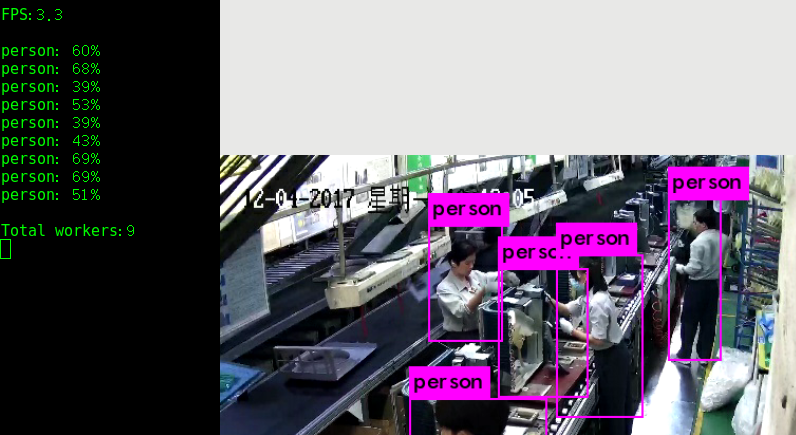
**NOTE：l.classes 是\*.names中的排序数**

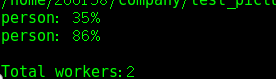
C）修改代码数出总人数，image.c中的draw\_detections（）函数



2018-01-08 14-44-12 的屏幕截图

目前测试视频成功。如下：



图片也测试成功，如下：  




D）这样的修改其实还是有点麻烦的，其实还可以这样：我假设你使用的是VOC.name, 在VOC中，car对应着6, person对应着14,然后修改代码。

在draw\_detections()函数中第一个for循环int class = max\_index(probs[i], classes);代码后面添加if(class!=6 && class!=14){continue;}

这样就可以检测人和车了。

还没有测试。。。。。。。

## E）--->重新编译和测试

**make clean**

**Make**

## 2、训练自己的数据集

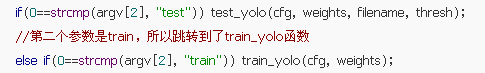
# 四YOLO原理理解

## 1、源码结构

（1）首先从main()函数开始看，该函数在darknet.c，在/darknet/examples/文件夹下。

该函数主要是命令行第一个输入参数的匹配，如果匹配到则直接跳到该函数。相当于算法框架的入口。

（2）在/darknet/examples/下的yolo.c文件，run\_yolo()函数

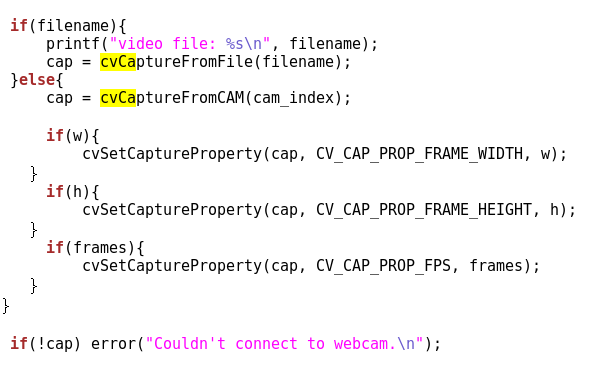


该文件主要是使用darknet 实现YOLO论文的思想。

我们现在是要训练自己的数据集，所以第二个参数是train.

（3）在/darknet/src/下的demo.c文件,demo()函数

主要实现yolo处理视频文件

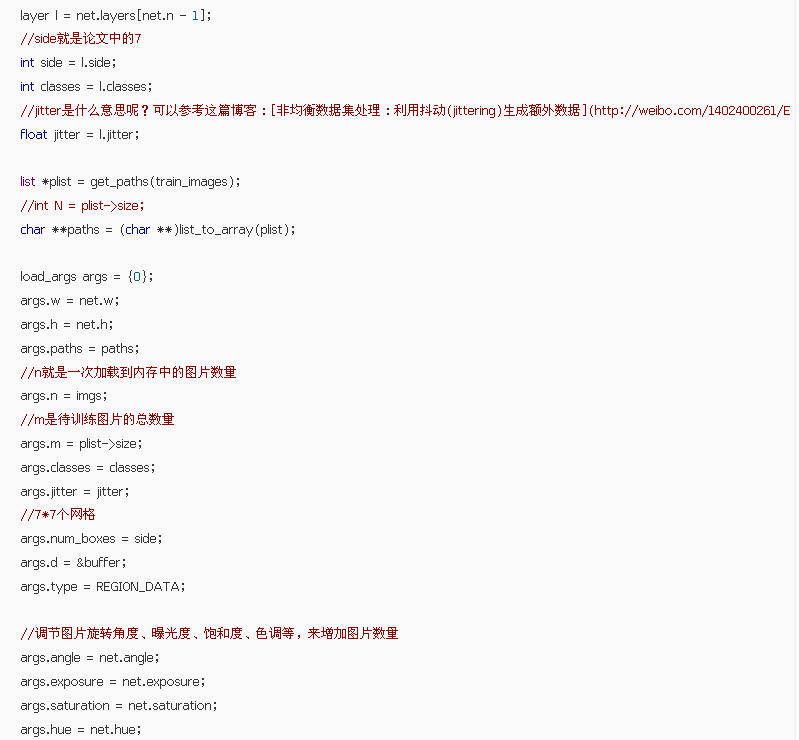


（4）在/darknet/examples/下的yolo.c文件，train\_yolo()函数



第三个参数是cfg/yolo.cfg。

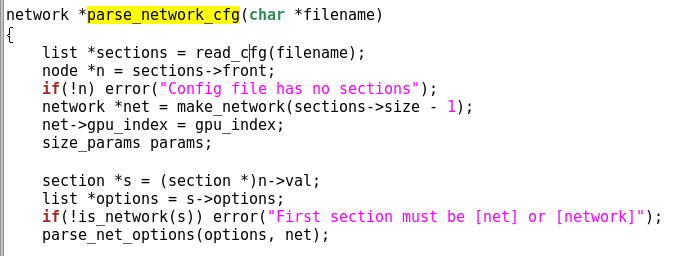
主要是加载神经网络架构和预训练参数，该函数的俩个参数指针指向这俩个文件。其余部分是对图像的一些预处理过程。



了解到这里就大致可以训练自己的数据了，下面是框架的大概实现过程。

（4）在/darknet/src/下的parser.c文件，网络参数解析函数

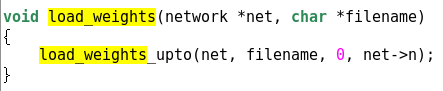
parse\_network\_cfg()



read\_cfg这个函数将所有的参数读到一个图中，这里的图可以理解为参数网，然后根据一定的规则去读这个参数网里的参数。即给参数定了一个读取的规则，这样就不会导致数据混乱。

（5）在/darknet/src/下的parser.c文件,加载权重函数

load\_weights()

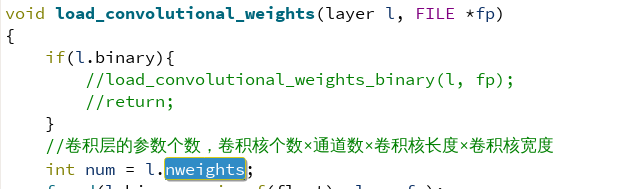


大概是这样，具体的函数没有去分析。

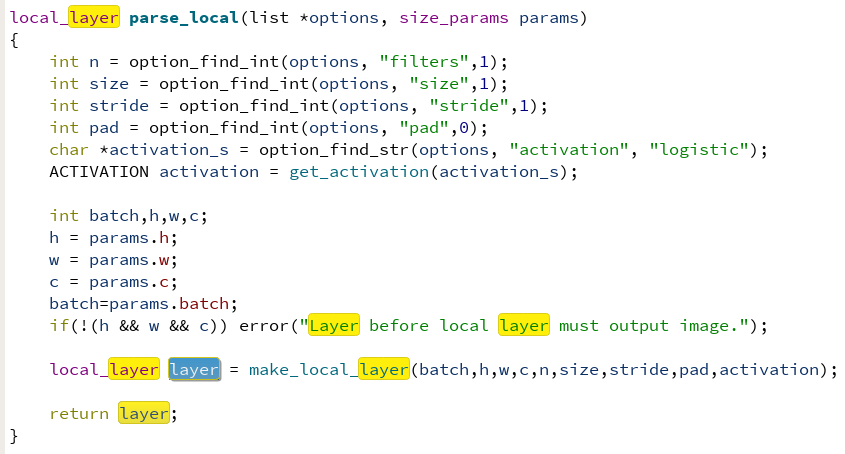
可见主要实现过程是调用load\_weights\_upto()函数



下面我们看卷积层的权重加载load\_convolutional\_weights()



下面看看如何得到结构体layer



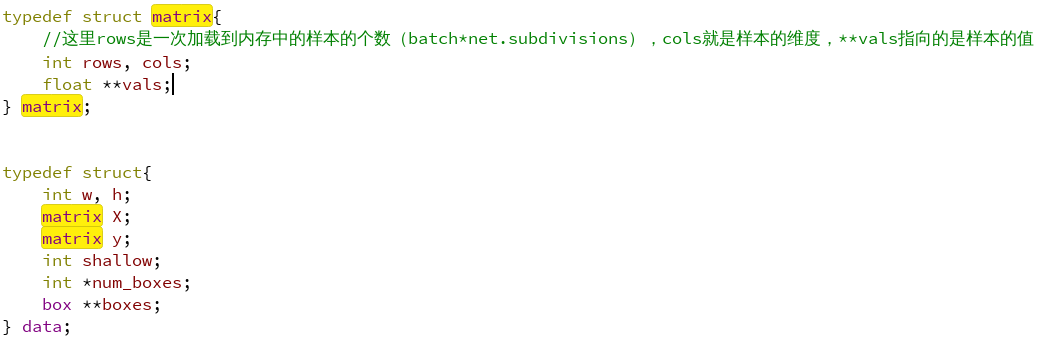
可见，这个函数返回一个结构，其包含batch等元素。

（6）在/darknet/src/下的network.c文件中，训练神经网络函数

train\_network(network \*net, data d)

注意：yolov2的大部分结构体定义都在 /include/目录下的darknet.h中。

先看下data和matrix结构体，我猜是打标好的数据结构

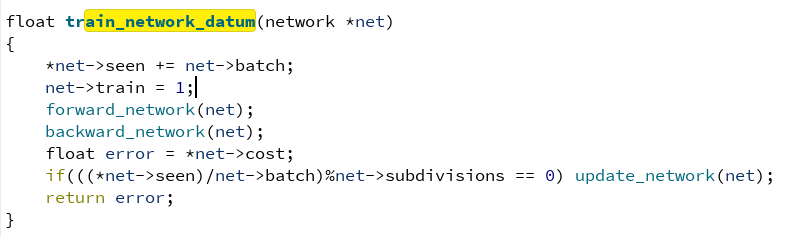


下面看train\_network()函数内容。



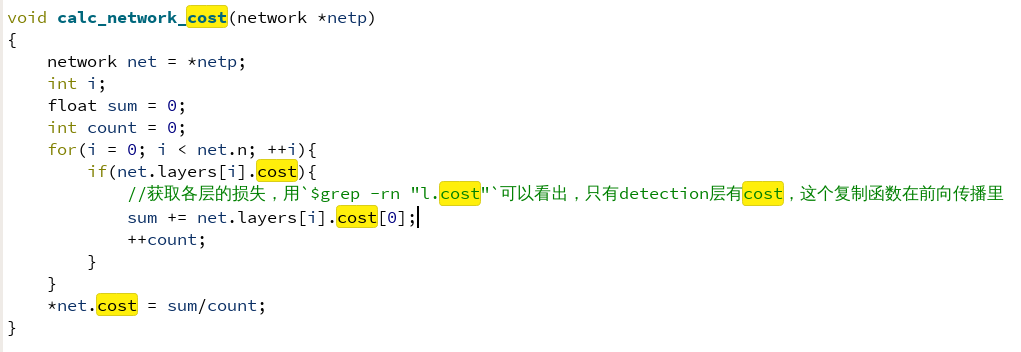
然后分析以上函数调用的其它函数，下面是data.c内容





训练主要是forward\_network和backward\_network,这俩个函数。

下面是计算代价函数



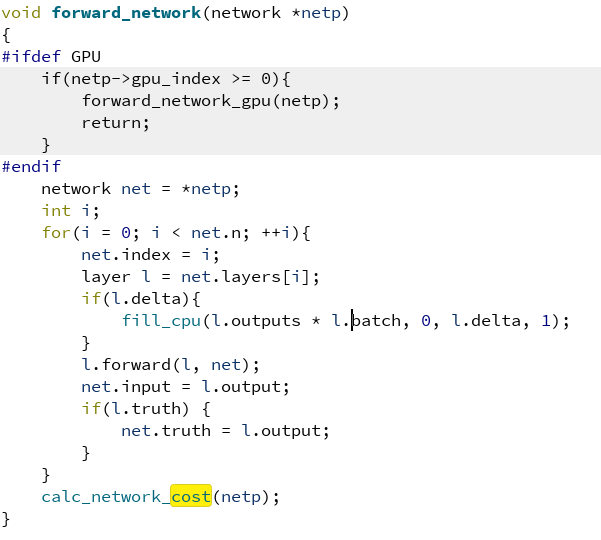
该函数最后返回平均损失。

到这里我们只剩下最重要的两个函数：

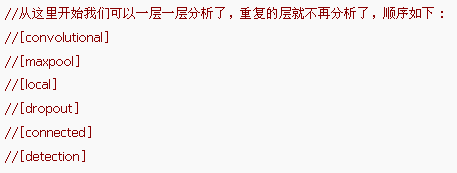
forward\_network(network \*netp); //前向传播

backward\_network(network \*netp); //反向传播

（7）在src/network.c文件，前向传播函数forward\_network(network \*netp)



分为GPU和CPU俩种情况，主要是forward\_network\_gpu()和forward()函数，下面分析每一层



由于还没看懂，所以先附上网络的解释。

<http://blog.csdn.net/u014540717/article/details/53232426>

（8）在src/network.c文件，反向传播函数backward\_network(network \*netp)

同样附上网址：

<http://blog.csdn.net/u014540717/article/details/53485680>

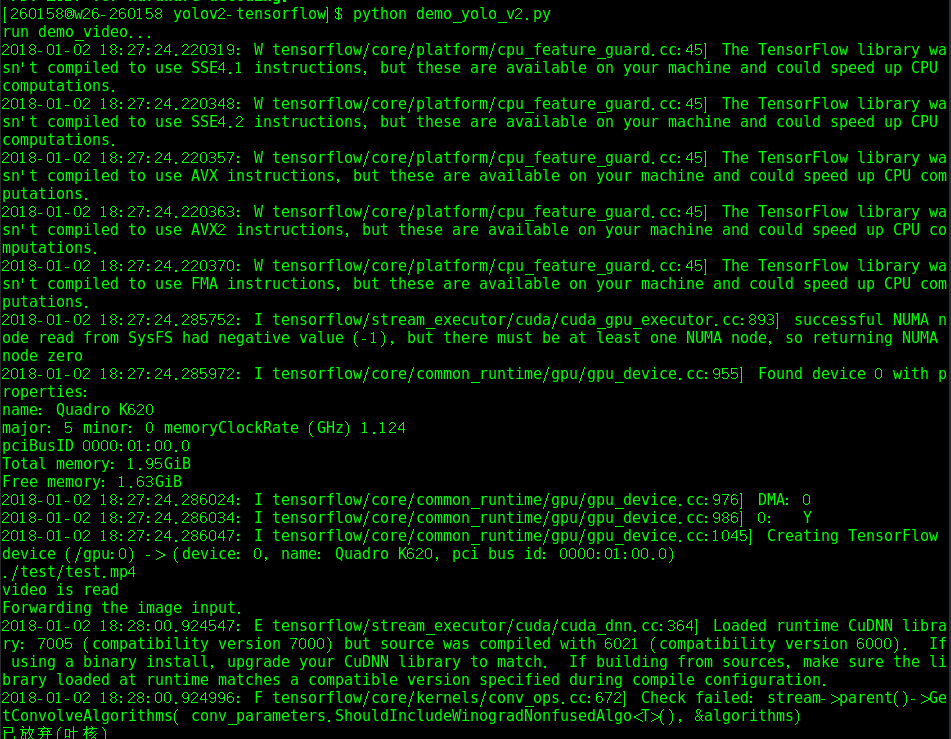
# 三、杂记

## 1、测试yolov2-tensorflow

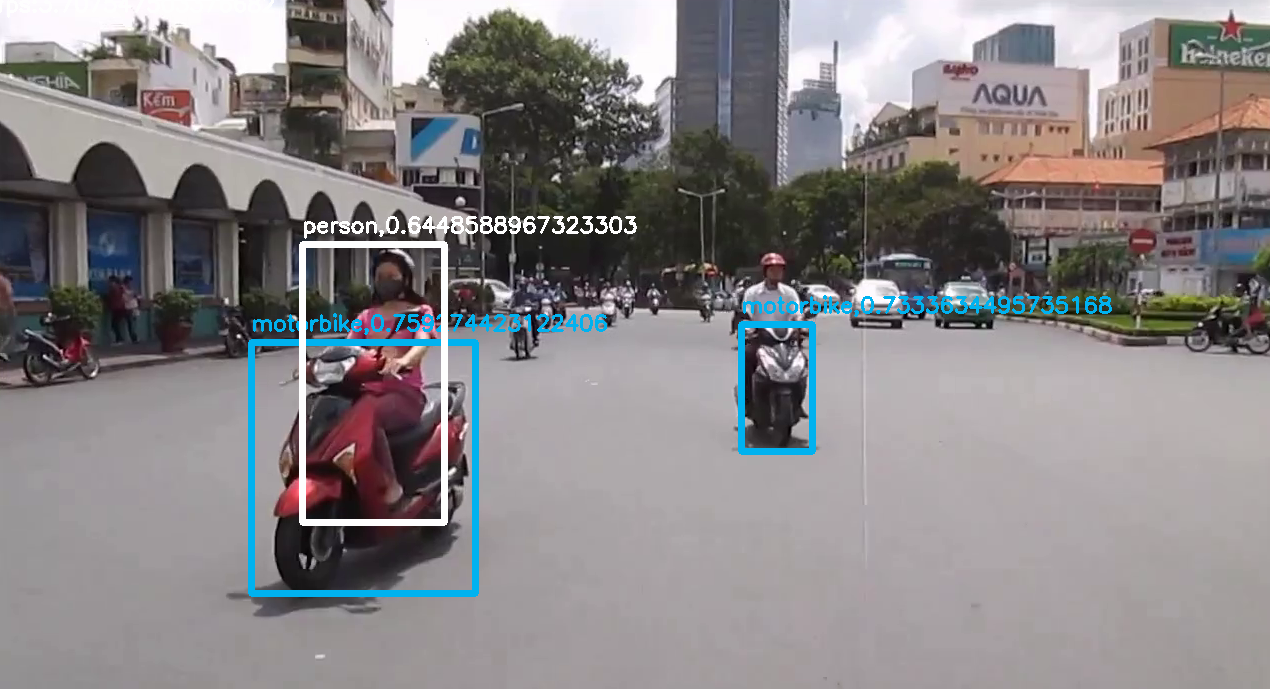
网址：<https://github.com/shishichang/yolov2-tensorflow>

出现错误：

大概意思就是源码是用CUDNN6.0.21版本编译的，不兼容，我的是v7版本的。



在改成v6后，测试自带视频，就成功了。如下：



## 2、QT配置

下载网址：

<https://www1.qt.io/download-open-source/?utm_referrer=https%3A%2F%2Fwww.qt.io%2Fdownload-qt-for-application-development#section-2>

用快捷方式打开的QT不能输入中文，

运行export QT\_IM\_MODULE=iBus

然后运行./qtcreator.sh，就可以输入中文了。

--->配置并运行OPENCV

在安装好Qt之后，只需要2步：

（1）新建项目

（2）修改pro文件

在pro文件里面，填写以下内容即可：

INCLUDEPATH += /usr/local/include \

/usr/local/include/opencv \

/usr/local/include/opencv2

LIBS += /usr/local/lib/libopencv\*