Linux-Faster-rcnn安装全过程

第0章 软件版本及硬件型号

0.1 软件版本

Ubuntu14.04、CUDA8、CUDNN5.1 for CUDA8、opencv3.1、python版本faster-rcnn、GPU版本

0.2 硬件型号

I7四代CPU、内存4G、固态硬盘250G、影驰GTX1070

第1章 ubuntu系统安装过程

1.1 下载ubuntu14.04

下载地址：

<http://cdimage.ubuntu.com/netboot/14.04/?_ga=1.104910848.2099103184.1489250975>

1.2 制作安装系统U盘

1. 安装UitralSO

2. 制作安装系统U盘

参考网址：<http://jingyan.baidu.com/article/19020a0a396b6e529d2842cb.html>

1.3 直接U盘安装单系统

1. 到其他选项，分配分区（除了“/”，所有的都分配逻辑分区）

/ ：30G （分配住主分区）

/boot ：200M

swap ：4G（如果够大最好8G）

/home ：剩下的

2. 引导项不用更改，默认就行

参考网址：

<http://www.jianshu.com/p/2eebd6ad284d>

<http://bbs.51cto.com/thread-966627-1.html>

1.4 更新软件源

参考网址：<http://www.linuxidc.com/Linux/2014-04/100476.htm>

选择“阿里”或“163”的镜像源，如果出现软件无法update，很可能是软件源的问题，更换软件源可以解决问题

1.5 安装搜狗输入法（非必要）

参考网址：<http://jingyan.baidu.com/article/ad310e80ae6d971849f49ed3.html>

1.6 安装chromeium（非必要）

1. 在软件中心中安装chromium

2. 需要安装chromium flash插件

待续

第2章 安装相关依赖及NVIDIA显卡驱动

2.1 安装相关依赖项

1. sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler

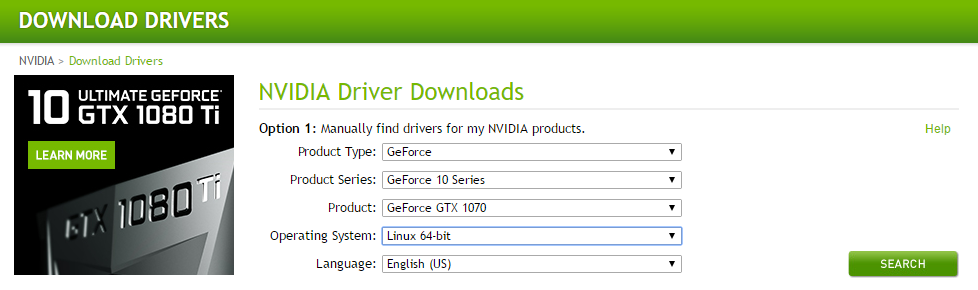
2. sudo apt-get install --no-install-recommends libboost-all-dev

3. sudo apt-get install libopenblas-dev liblapack-dev libatlas-base-dev

4. sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev

2.2 查询NVIDIA驱动

首先去官网(<http://www.nvidia.com/Download/index.aspx?lang=en-us>)查看适合自己显卡的驱动：



我下载的驱动版本如下：



2.3 安装NVIDIA驱动（上传版需更去掉用户密码）

1. 安装之前先卸载已经存在的驱动版本：

sudo apt-get remove --purge nvidia\*

2. 若电脑是集成显卡（NVIDIA独立显卡忽略此步骤），需要在安装之前禁止一项：

sudo service lightdm stop

3. 按ALT+CTRL+F1切换到字符界面（按ALT+CTRL+F7切换到图形界面）

切换到字符界面后需要登录：

用户名：

密码：

按回车进行登录

4. 安装时把驱动文件放在home下（进入字符界面前），不需要cd切换目录，执行以下指令安装驱动：

sudo add-apt-repository ppa:xorg-edgers/ppa

sudo apt-get update

chmod +x ./NVIDIA-Linux-x86\_64-375.39.run（chmod指令是给管理员权限）

sudo ./NVIDIA-Linux-x86\_64-375.39.run

5. 安装完成之后输入以下指令进行验证，若列出了GPU的信息列表则表示驱动安装成功：

sudo nvidia-smi

6. 安装完成后，开启图形界面：

sudo start lightdm

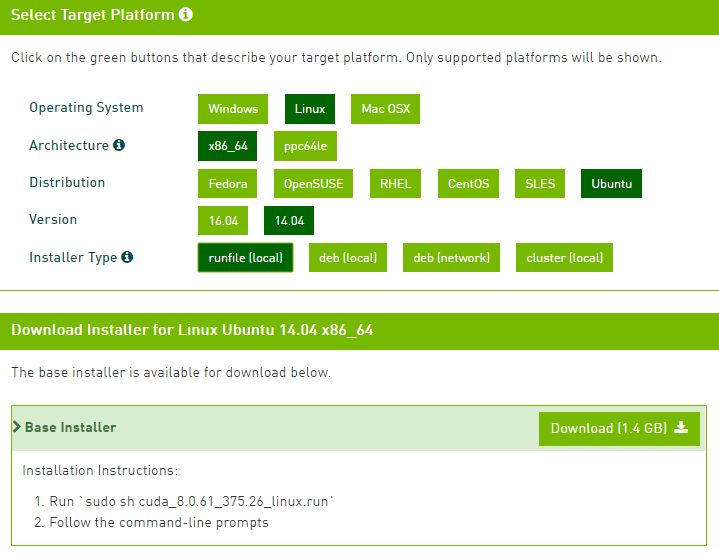
（关闭图形界面：sudo stop lightdm）

第3章 安装CUDA

3.1 下载CUDA

CUDA是NVIDIA的编程语言平台，想使用GPU就必须要使用cuda

首先在官网上(<https://developer.nvidia.com/cuda-downloads>)下载CUDA：

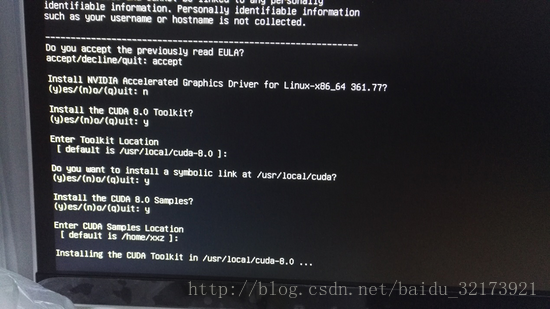


3.2 安装CUDA

1. 下载完成后执行以下命令：

sudo sh cuda\_8.0.61\_375.26\_linux.run

2. 之后界面会出现‘更多0%’，一直按回车；这里需要注意其中一个地方需要填写 N 才行，不能写 Y 。因为不需要再安装一次显卡的驱动，如图



3.3 配置环境变量

1. 打开~/.bashrc文件：

sudo gedit ~/.bashrc

2. 将以下内容写入到~/.bashrc尾部：

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

export LD\_LIBRARY\_PATH=/usr/local/cuda8.0/lib64${LD\_LIBRARY\_PATH:+:${LD\_LIBRARY\_PATH}}

3. 退出后

source ~/.bashrc

3.4 测试CUDA的sammples

cd /usr/local/cuda-8.0/samples/1\_Utilities/deviceQuery #由自己电脑目录决定

sudo make

sudo ./deviceQuery

3.5 参考网址

1. <http://blog.csdn.net/xuzhongxiong/article/details/52717285>

2. <http://blog.csdn.net/baidu_32173921/article/details/53510764>

第4章 安装cuDNN

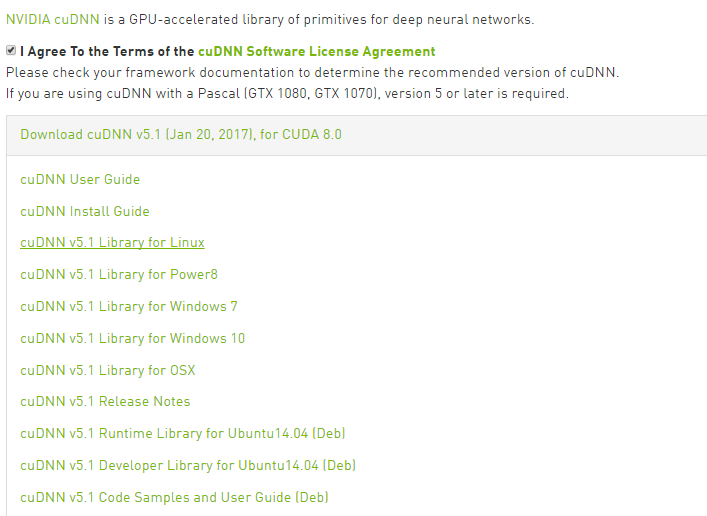
4.1 下载cuDNN（上传版需更去掉账号密码）

cuDNN是GPU加速计算深层神经网络的库

首先去官网(https://developer.nvidia.com/rdp/cudnn-download)下载cuDNN，需要注册一个账号才能下载：

账号：

密码：



4.2 安装cuDNN

1. 下载cuDNN5.1后直接在home下进行解压（解压后的文件名叫cuda）

2. cd到include目录下：cd /home/jc/cuda/include

sudo cp cudnn.h /usr/local/cuda/include/ #复制头文件

3. 再cd进入lib64目录下的动态文件进行复制和链接：

cd /home/jc/cuda/lib64

sudo cp lib\* /usr/local/cuda/lib64/ #复制动态链接库

cd /usr/local/cuda/lib64/

sudo rm -rf libcudnn.so libcudnn.so.5 #删除原有动态文件

sudo ln -s libcudnn.so.5.1.10 libcudnn.so.5 #生成软衔接（注意里面的5.1.10要根据自己下载的版本而定）

sudo ln -s libcudnn.so.5 libcudnn.so #生成软链接

4.3 参考网址

1. <http://blog.csdn.net/xuzhongxiong/article/details/52717285>

2. <http://blog.csdn.net/baidu_32173921/article/details/53510764>

第5章 安装opencv3.1

5.1 安装所需的库

1 [compiler] sudo apt-get install build-essential

2 [required] sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

3 [optional] sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev

5.2 下载opencv3.1

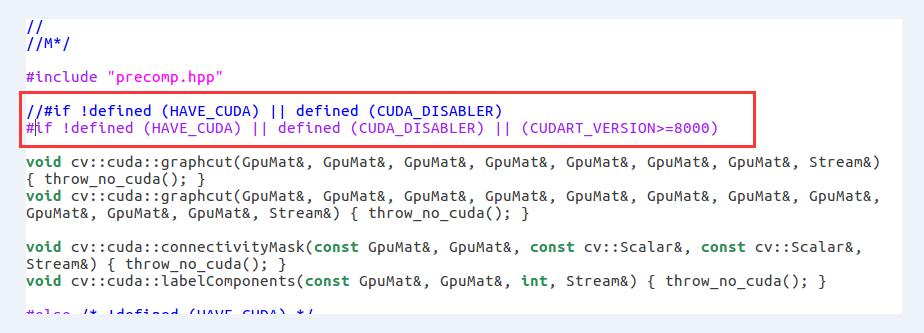
从官网下载<http://opencv.org/releases.html>，选3.10-Sources

5.3 编译opencv

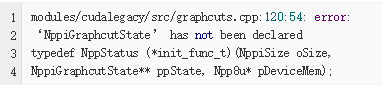
1. 将opencv3.1放到home下进行解压

2. 修改 ～/opencv/modules/cudalegacy/src/graphcuts.cpp文件内容，如图：

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



如果不操作会出现如下错误：



这是因为opencv3.1.0与cuda8.0不兼容导致的

3. 进行编译

cd ~/opencv-3.1.0

mkdir release

cd release

cmake –D CUDA\_NVCC\_FLAGS=--Wno-deprecated-gpu-targets -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local ..

（为了消除NVCC警告，-D CUDA\_NVCC\_FLAGS=--Wno-deprecated-gpu-targets用于指定CUDA编译器(nvcc)的编译选项，如果不指定--Wno-deprecated-gpu-targets选项则在编译Caffe时会产生如下编译警告：

nvcc warning : The ‘compute\_20’, ‘sm\_20’, and ‘sm\_21’ architectures are deprecated, and may be removed in a future release (Use -Wno-deprecated-gpu-targets to suppress warning).）

(cmake成功后会有提示)

make –j8 (GPU加速make)

sudo make install

5.4 参考网址

1. <http://www.tuicool.com/articles/nYJrYra>

2. <http://blog.csdn.net/xuzhongxiong/article/details/52717285>

第6章 安装caffe

6.1 从github上获取caffe：

git clone https://github.com/BVLC/caffe.git

6.2 makefile.config

因为make指令只能make Makefile.config文件，而Makefile.config.example是caffe给出的makefile例子，因此，首先将Makefile.config.example的内容复制到Makefile.config：

cd ./home/jc/caffe

sudo cp Makefile.config.example Makefile.config

6.3 打开并修改配置文件：

打开Makefile.config文件

cd ./home/jc/caffe

sudo gedit Makefile.config

根据个人情况修改文件：

a. 若使用cudnn，则

将

#USE\_CUDNN := 1

修改成：

USE\_CUDNN := 1

b. 若使用的opencv版本是3的，则

将

#OPENCV\_VERSION := 3

修改为：

OPENCV\_VERSION := 3

c. 若要使用Python来编写layer，则

将

#WITH\_PYTHON\_LAYER := 1

修改为

WITH\_PYTHON\_LAYER := 1

d. 重要的一项（我的是14.04不需要改） :

将# Whatever else you find you need goes here.下面的

INCLUDE\_DIRS := $(PYTHON\_INCLUDE) /usr/local/include

LIBRARY\_DIRS := $(PYTHON\_LIB) /usr/local/lib /usr/lib

修改为：

INCLUDE\_DIRS := $(PYTHON\_INCLUDE) /usr/local/include /usr/include/hdf5/serial

LIBRARY\_DIRS := $(PYTHON\_LIB) /usr/local/lib /usr/lib /usr/lib/x86\_64-linux-gnu /usr/lib/x86\_64-linux-gnu/hdf5/serial

（这是因为ubuntu16.04的文件包含位置发生了变化，尤其是需要用到的hdf5的位置，所以需要更改这一路径.）

6.4 修改makefile文件

打开makefile文件（caffe文件夹下，直接打开），做如下修改：

将：

NVCCFLAGS +=-ccbin=$(CXX) -Xcompiler-fPIC $(COMMON\_FLAGS)

替换为：

NVCCFLAGS += -D\_FORCE\_INLINES -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON\_FLAGS)

6.5 编辑/usr/local/cuda/include/host\_config.h将其中的第115行注释掉：

sudo gedit /usr/local/cuda/include/host\_config.h

将

#error -- unsupported GNU version! gcc versions later than 4.9 are not supported!

改为

//#error -- unsupported GNU version! gcc versions later than 4.9 are not supported!

6.6 将一些文件复制到/usr/local/lib文件夹下

#注意自己CUDA的版本号，到该文件夹下查处找对应标号

sudo cp /usr/local/cuda-8.0/lib64/libcudart.so.8.0 /usr/local/lib/libcudart.so.8.0 && sudo ldconfig

sudo cp /usr/local/cuda-8.0/lib64/libcublas.so.8.0 /usr/local/lib/libcublas.so.8.0 && sudo ldconfig

sudo cp /usr/local/cuda-8.0/lib64/libcurand.so.8.0 /usr/local/lib/libcurand.so.8.0 && sudo ldconfig

如果未操作该步骤可能会出现如下错误：

"libcudart.so.8.0 cannot open shared object file: No such file or directory"

6.7 将一些文件复制到/usr/local/lib文件夹下

#注意自己CUDA的版本号，到该文件夹下查处找对应标号

sudo cp /usr/local/cuda-8.0/lib64/libcudnn.so /usr/local/lib/libcudnn.so && sudo ldconfig

sudo cp /usr/local/cuda-8.0/lib64/libcudnn.so.5 /usr/local/lib/libcudnn.so.5 && sudo ldconfig

sudo cp /usr/local/cuda-8.0/lib64/libcudnn.so.5.1.10 /usr/local/lib/libcudnn.so.5.1.10 && sudo ldconfig

6.8 编译及测试

1. 编译

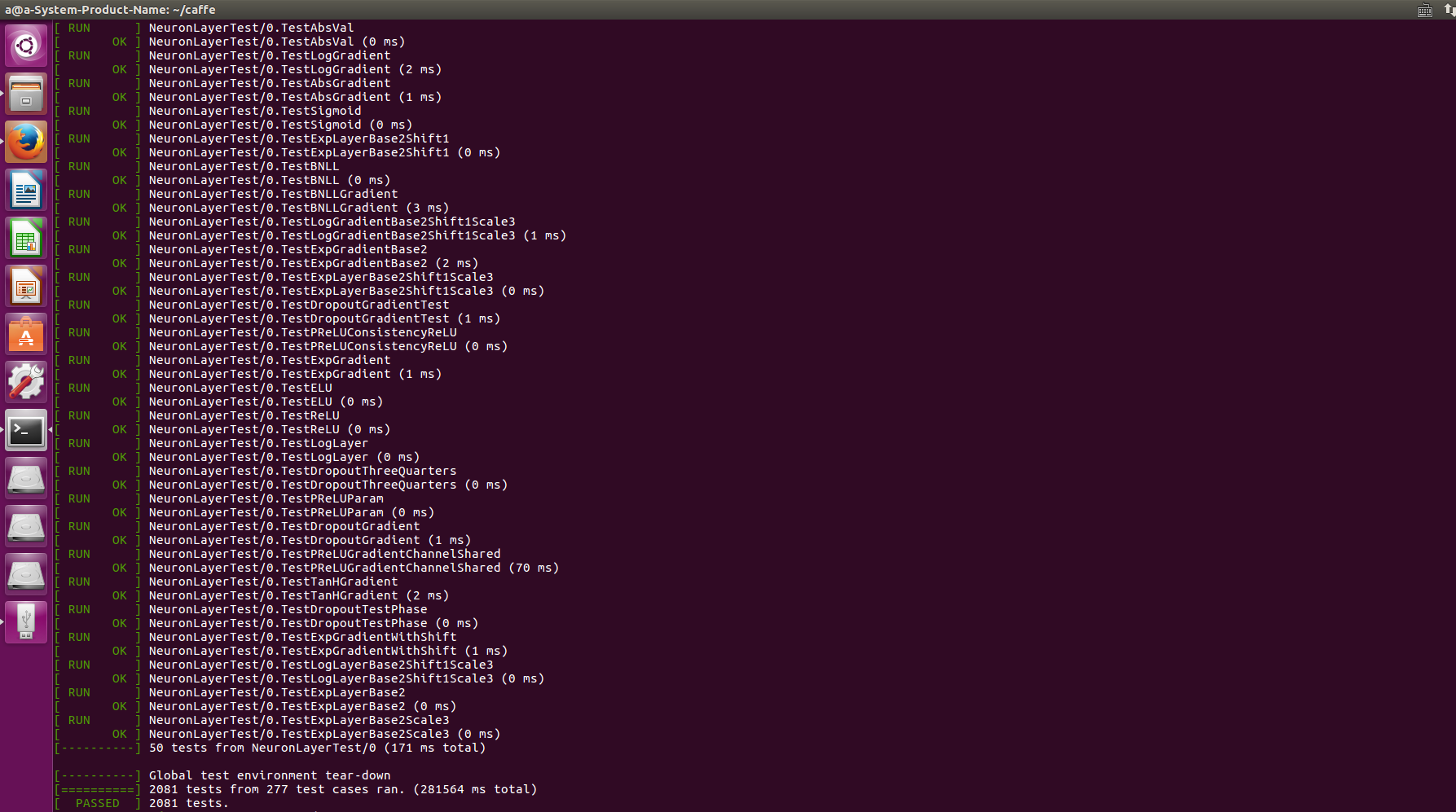
cd /home/jc/caffe

make all –j8

2. 测试

sudo make –j8 runtest

如果运行之后出现下图说明运行成功



6.9 MNIST数据集测试

配置caffe完成后，我们可以利用MNIST数据集对caffe进行测试，过程如下：

1. 将终端定位到Caffe根目录

cd ~/caffe

2. 下载MNIST数据库并解压缩

./data/mnist/get\_mnist.sh

3.将其转换成Lmdb数据库格式

./examples/mnist/create\_mnist.sh

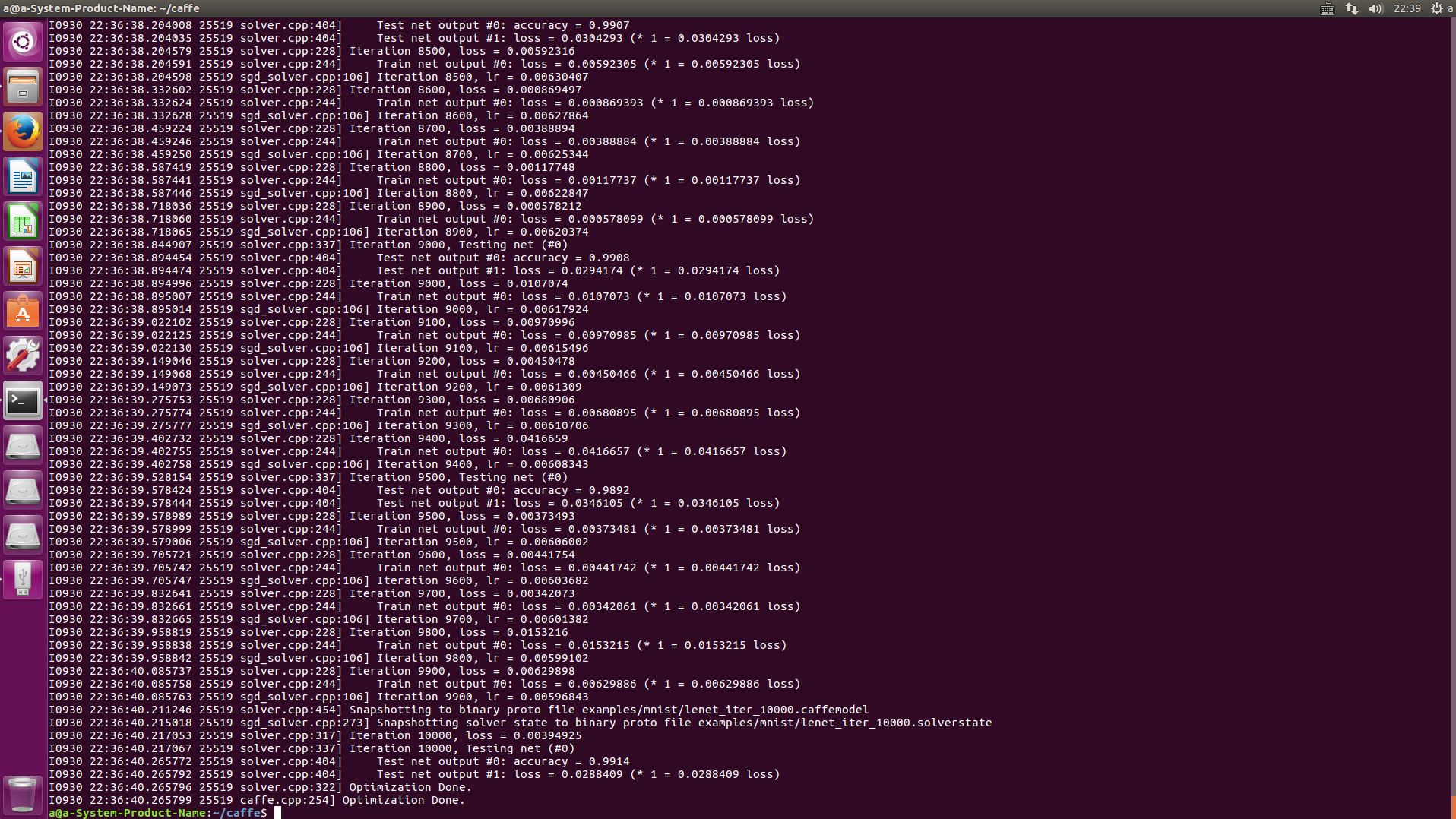
4.训练网络

./examples/mnist/train\_lenet.sh

5. 训练结果

训练的时候可以看到损失与精度数值，如下图：

可以看到最终训练精度是0.9914



6.10 若之前已经编译过caffe

1. 若之前已经编译过caffe，直接把该caffe目录拷贝过来，安装必备的库及可运行程序，若提示“.py文件没有权限”，则用“chmod +x”赋予权限。（如果没成功，重新编译~/caffe）

6.11 参考网址

1. <http://blog.csdn.net/u014696921/article/details/60140264>

2. <http://blog.csdn.net/xuzhongxiong/article/details/52717285>

第7章 faster rcnn安装

7.1 软件源无法更新

参考网址：<http://blog.csdn.net/loovejava/article/details/21837935>

7.2 安装pip

sudo apt-get install python-pip

可能出现问题：

下列软件包有未满足的依赖关系： python-pip : 依赖: python-setuptools (>= 0.6c1) 但是它将不会被安装 推荐: python-dev-all (>= 2.6) 但无法安装它 E: 无法修正错误，因为您要求某些软件包保持现状，就是它们破坏了软件包间的依赖关系。

解决方法：

1. 安装aptitude工具

sudo apt-get install aptitude

2. sudo aptitude install python-pip

下列动作将解决这些依赖关系： 保持 下列软件包于其当前版本：

1) python-pip [未安装的]

2) python-setuptools [未安装的]

3. 选项

**是否接受该解决方案？[Y/n/q/?] n**

下列动作将解决这些依赖关系：

降级 下列软件包：

1) python-pkg-resources [3.3-1ubuntu2 (now) -> 3.3-1ubuntu1 (trusty)]

**是否接受该解决方案？[Y/n/q/?] Y**

下列软件包将被“降级”：

python-pkg-resources

下列“新”软件包将被安装。

python-colorama{a} python-distlib{a} python-html5lib{a} python-pip

python-setuptools{a}

0 个软件包被升级，新安装 5 个， 1 个被降级， 0 个将被删除， 同时 0 个将不升级。

需要获取 604 kB 的存档。 解包后将要使用 2,513 kB。

**您要继续吗？[Y/n/?] Y**

7.3 安装python依赖包

sudo pip install Cython

sudo pip install easydict

sudo apt-get install python-opencv

7.4 下载源代码，修改并编译cython

1. 下载源代码

git clone --recursive https://github.com/rbgirshick/py-faster-rcnn.git

2. 修改并编译cython

cd py-faster-rcnn/lib

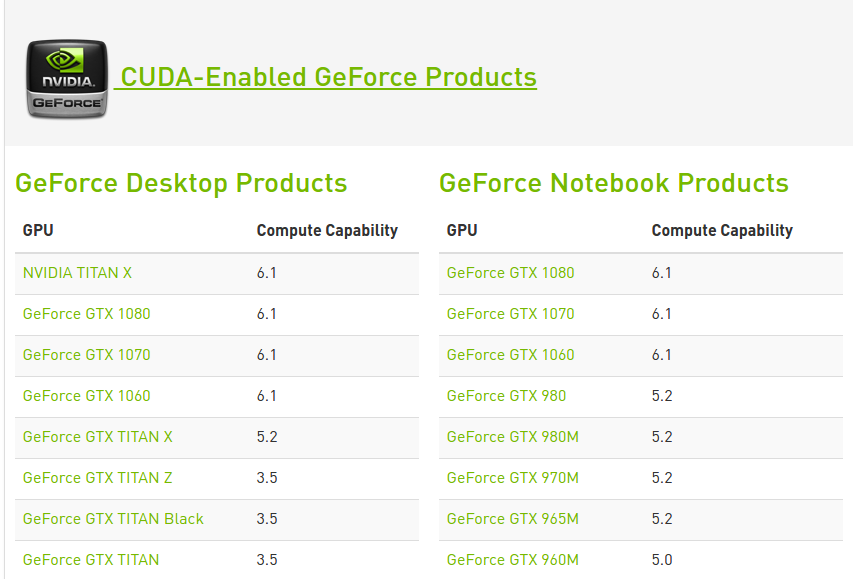
修改setup.py文件第135行

'nvcc': ['-arch=sm\_35',

根据自己的显卡计算能力进行修改,查询网址为:CUDA GPUs

因为是gtx1070，所以改为：

'nvcc': ['-arch=sm\_61',



3. 保存后执行

make –j8

7.5 编译faster前的准备工作

1. 修改Makefile.config文件

cd /home/jc/py-faster-rcnn/caffe-faster-rcnn

执行：

cp Makefile.config.example Makefile.config

对Makefile.config文件进行修改.

将

# USE\_CUDNN := 1

# WITH\_PYTHON\_LAYER := 1

改为

USE\_CUDNN := 1

WITH\_PYTHON\_LAYER := 1

注：# USE\_PKG\_CONFIG := 1一定不要改，否则会出现如下错误：

/usr/bin/ld: cannot find –lip…

2. 修改cudnn加速文件

这里需要注意一下，因为作者的caffe版本不支持新的cudnn加速，我们需要用caffe-master版的文件去替换faster中的文件，这样编译就不会出错啦。

a. caffe中的文件位置

/home/jc/caffe/ include/caffe/util/cudnn.hpp

/home/jc/caffe/ src/caffe/util/cudnn.cpp

/home/jc/caffe/ src/caffe/ layers/ #目录下16个cudnn\_开头的文件

/home/jc/caffe/include/caffe/layers/ #目录下8个cudnn\_开头的文件

b. faster rcnn中的文件位置

/home/jc/py-faster-rcnn/caffe-fast-rcnn/include/caffe/util/cudnn.hpp

/home/jc/py-faster-rcnn/caffe-fast-rcnn/src/caffe/util/cudnn.cpp

/home/jc/py-faster-rcnn/caffe-fast-rcnn/src/caffe/layers/ #目录下16个cudnn\_开头的文件

/home/jc/py-faster-rcnn/caffe-fast-rcnn/include/caffe/layers/ #目录下8个cudnn\_开头的文件

3. 修改一个include错误

打开py-faster-rcnn/caffe-fast-rcnn/src/caffe/test/test\_smooth\_L1\_loss\_layer.cpp

删除第十一行

#include "caffe/vision\_layers.hpp"

4. 修改无法引用opencv错误

打开/home/jc/py-faster-rcnn/caffe-fast-rcnn/Makefile文件，将

LIBRARIES += glog gflags protobuf boost\_system boost\_filesystem m hdf5\_hl hdf5

改为

LIBRARIES += glog gflags protobuf boost\_system boost\_filesystem m hdf5\_hl hdf5 m

opencv\_core opencv\_highgui opencv\_imgproc opencv\_imgcodecs

如果不进行改动，则会出现：

‘CV::imread(cv::String const&, int)’未定义引用等错误

7.6 编译faster

仍然在cd /home/jc/py-faster-rcnn/caffe-fast-rcnn下

# compile Caffe modules

make -j8 all

make -j8 test

# run tests to ensure all the tests are passed

make –j8 runtest

# build pycaffe

make –j8 pycaffe

7.7 下载已经训练好的模型

（注：cd $FRCN\_ROOT 这不是指令，告诉你转到FRCN的根目录下）

cd /home/jc/py-faster-rcnn

./data/scripts/fetch\_faster\_rcnn\_models.sh #下载数据

7.8 执行demo

0. 安装所有的包都需要在根目录下进行

1. 安装easydict

sudo pip install easydict

若不安装会出现如下错误：

ImportError: No module named easydict

2. 安装scikit-image

sudo pip install –U scikit-image

若不安装会出现如下错误：

ImportError: No module named skimage.io

3. 安装scipy

sudo apt-get install python-scipy

4.安装pyparsing

sudo apt-get install python-pyparsing

5. 安装cycler

sudo pip install cycler

6. 安装dateutil

sudo pip install python-dateutil

7. ImportError: No module named google.protobuf.internal

sudo apt-get install python-numpy python-scipy python-matplotlib python-sklearn python-skimage python-h5py python-protobuf python-leveldb python-networkx python-nose python-pandas python-gflags Cython ipython

sudo apt-get install protobuf-c-compiler protobuf-compiler

8. sudo pip install functools32

9. sudo apt-get install python-yaml

10. 参考网址：<http://blog.csdn.net/liyaohhh/article/details/50876777>

11. 执行demo：

cd py-faster-rcnn/tools

./demo.py

7.9 若之前已经编译过faster-rcnn

1. 若之前已经编译过faster-rcnn目录，直接把该faster-rcnn目录拷贝过来，安装必备的库及可运行程序，若提示“.py文件没有权限”，则用“chmod +x”赋予权限。（如果没成功，重新编译~/lib和~/caffe-fast-rcnn）

7.10 参考网址

1. <https://www.douban.com/note/568788483/>

2. <http://blog.csdn.net/xuzhongxiong/article/details/52717285>

第8章 安装faster的运行环境eclipse+pydev

8.1 手动安装JDK

1. JDK下载网址：http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

下载Linux x64 jdk-8u121-linux-x64.tar.gz

2. 解压

双击：jdk-8u121-linux-x64.tar.gz

解压出来是一个jdk1.8.0\_121文件夹，将其重命名为“java”

3. 将文件夹移动到安装目录（本文放在“/usr/lib/jvm/”目录下），建议更改文件夹名称以便缩短路径（本文改为“java”）

cd /usr/lib

sudo mkdir jvm

cd

sudo cp –r jdk1.8.0\_121 /usr/lib/jvm

4. 配置环境变量 ，更改/etc/profile文件

sudo gedit /etc/profile

在文件最后加上如下几行：

export JAVA\_HOME=/usr/lib/jvm/jdk1.8.0\_121

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib

export PATH=${JAVA\_HOME}/bin:$PATH

保存关闭，最后：

source /etc/profile

6. 可用java -version查看版本号验证是否安装成功。

8.2 安装eclipse

1. 从官网http://www.eclipse.org/downloads/下载Eclipse IDE for Java EE Developers的Linux版本

eclipse-jee-mars-1-linux-gtk-x86\_64.tar.gz

2. 解压

双击：eclipse-jee-mars-1-linux-gtk-x86\_64.tar.gz

解压出来是一个eclipse文件夹

3. 将文件夹移动到安装目录

sudo mv eclipse /usr/local/

4. 在eclipse /usr/local/下找到eclipse运行文件，双击

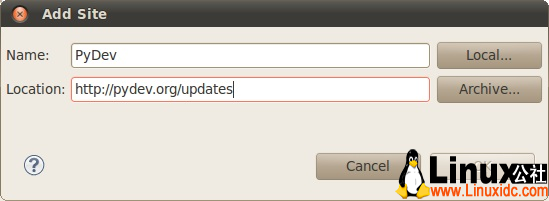
/usr/local/eclipse/eclipse

5. 右键锁定到任务栏

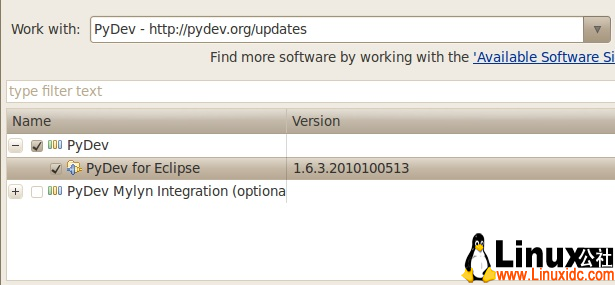
8.3 安装pydev插件

1. 在Help菜单中，选择Install New Software

2. 选择Add按钮，Name：PyDev（这个随便起），Location：http://pydev.org/updates（PyDev的更新地址），OK



3. PyDev下的PyDev for Eclipse，别的都不要选，否则依赖检查那关过不去。（注2）



4. 选择“相信”，最后重启Eclipse

5. 安装完pydev后发现没法新建工程：

安装好pydev后新建Python项目时提示”Project interpreter not specified“的错误,这是因为没有导入python开发环境所致

解决方法如下：

a.找到eclipse---->window----->preference.找到左侧栏边得pydev，展开，点击Interpreters---->Python InInterpreter

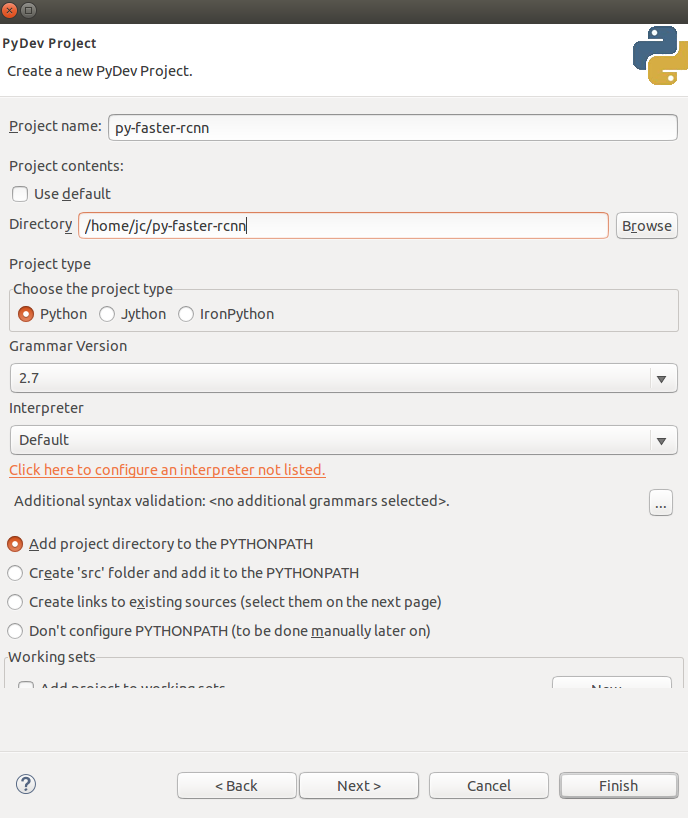
b. 在右侧找到New，名字可以不命名，找到python安装路径，python安装路径是：/usr/bin，选择python2.7即可

c. 最后Apply、OK，等待导入完成，即可新建python项目

6. 发现无法在py-faster-rcnn下新建工程

解决方法：因为py-faster-rcnn下已经存在了eclipse的工程文件，删除.project和.pydevproject两个文件即可

7. 建立工程如图



8.4 eclipse相关设置

IDE背景设置参见：<http://blog.csdn.net/alvine008/article/details/19970795>

字体大小设置参见：<http://blog.sina.com.cn/s/blog_5665582b01015ivc.html>

8.5 eclipse输入参数设置

IDE背景设置参考网址

1. <https://my.oschina.net/1462469/blog/374623>

8.6 参考网址

1. <http://www.linuxdiyf.com/linux/18510.html>