改装cuda7.5

电脑已有的配置经如下操作得到

《Ubuntu16.04+cuda8+caffe安装.docx》

《Ubuntu16.04安装matlab\_R2014b》

《Ubuntu16.04+cuda8+caffe+改为cudnn5.0安装.dot》

最终的结果为：

电脑配置：

系统：Ubuntu16.04

GPU: NVIDIA GTX 750 Ti

安装内容：

NVIDIA驱动：375.20

CUDA：8.0.

cuDNN：5.0

Opencv：3.1

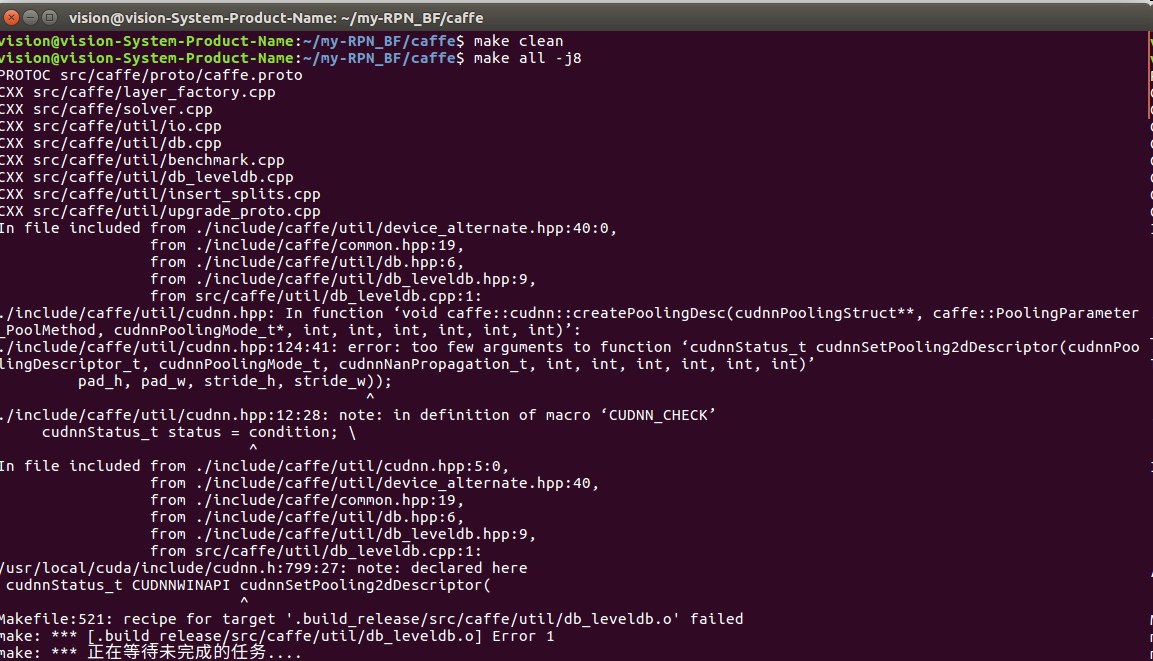
Python

Python notebook

Matlab2014b

改装cuda7.5原因：

要复现实验室行人检测论文RPN\_BF，由于原始环境为ubuntu14.04+cuda7.5+matlab2014b，在编译他的caffe时，即<http://github.com/zhangliliang/caffe/tree/RPN_BF>时，报



错误，怀疑是cuda版本的问题，原本以为是cudnn版本的问题，由5.1降到5.0还是报这个错。

主要参考链接：

http://blog.csdn.net/g0m3e/article/details/51420565#comments

1. **卸载cuda8.0**

参考链接：<http://www.th7.cn/system/lin/201601/149143.shtml>



本机操作：

cd /usr/local/cuda-8.0/bin

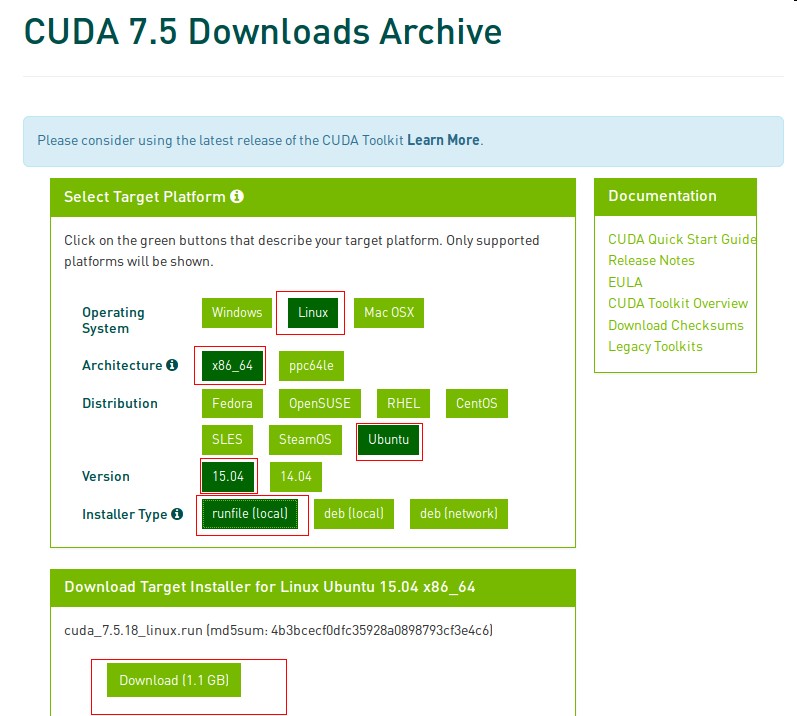
sudo ./uninstall\_cuda \_8.0.pl

（卸载后cuda-8.0的文件夹还在，不知道会不会有影响，暂时不管）

1. **安装cuda7.5**
2. 下载cuda7.5

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

（这个下载链接好难找的，要珍惜）



参考链接：<http://blog.csdn.net/g0m3e/article/details/51420565#comments>

1. 安装cuda7.5

以文件名为cuda.run为例，终端下输入

sh cuda.run –override

启动安装程序，此处有大量的条款，一路空格到最后 输入accept，依次输入y回车，然后n(不安装显卡驱动)，然后一路y回车，有一个地方需要输入密码，还有两个地方确认安装路径，直接回车即可，完成安装，默认安装路径是/usr/local

插图

1. 设置环境变量

打开~/.bashrc文件：

sudo vim ~/.bashrc

将以下内容写入到~/.bashrc尾部：

export PATH=/usr/local/cuda-7.5/bin${PATH:+:${PATH}}

export LD\_LIBRARY\_PATH=/usr/local/cuda-7.5/lib64${LD\_LIBRARY\_PATH:+:${LD\_LIBRARY\_PATH}}

保存退出（按Esc，输入命令:Wq,表示保存退出）

然后刷新

输入：



1. 测试cuda

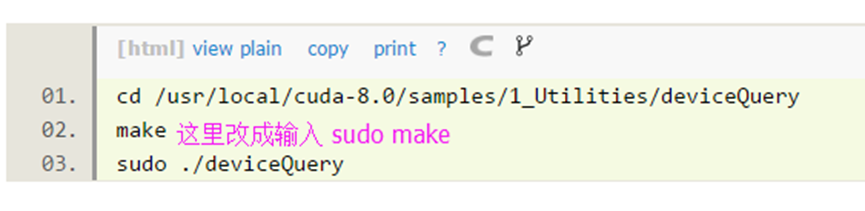
命令行回到最初文件夹，运行



结果如下：

插图：

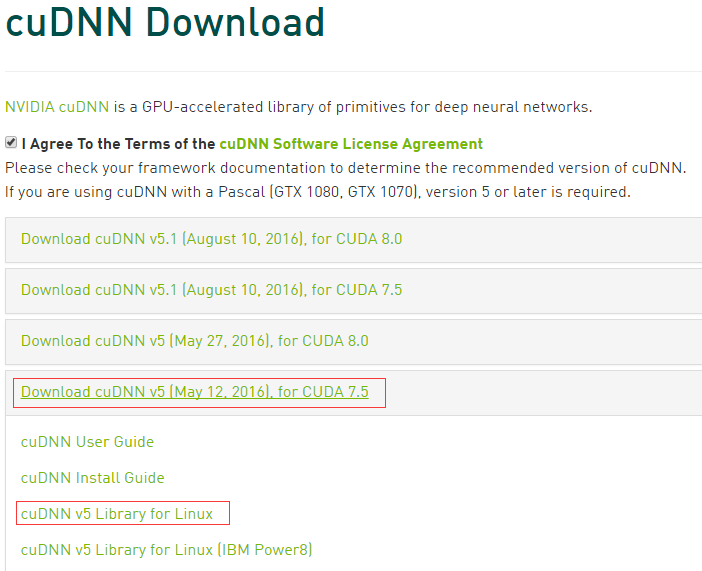
1. 测试cuda的sample



如果显示的是一些关于GPU的信息，则说明安装成功了。(最后显示是ＰＡＳＳ不是Faile!)

1. **安装cudnn5.0（对应cuda7.5版本）**
2. 下载cudnn5.0

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



1. 安装cuDNN

安装cuDNN比较简单，解压后把相应的文件拷贝到对应的CUDA目录下即可：

解压：

tar –zxvf cudnn-8.0-linux-x64-v5.1.tgz

cd进入cuDNN5.0解压之后的include目录：

cd cuda/include

在命令行进行如下操作

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

再将cd进入lib64目录下的动态文件进行复制和链接：



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

cd /usr/local/cuda/lib64/

（以下命令要根据不同版本的cudnn进行修改，不然后面caffe安装会有很多这边的问题）

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

sudo chmod u=rwx,g=rx,o=rx libcudnn.so.5.1.5 #修改文件权限

sudo ln -s libcudnn.so.5.1.5 libcudnn.so.5 #生成软衔接

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

更新链接库：

**sudo ldconfig**

测试cuda有很多方法，上面的测试是成功了，但是

cd /home/vision/NVIDIA\_CUDA-7.5\_Samples

sudo make all -j4

先是报cannot find lnvcuvid的错误，按照一下链接解决了这个问题

<http://blog.csdn.net/lanxueCC/article/details/53422255>

sudo make clean后

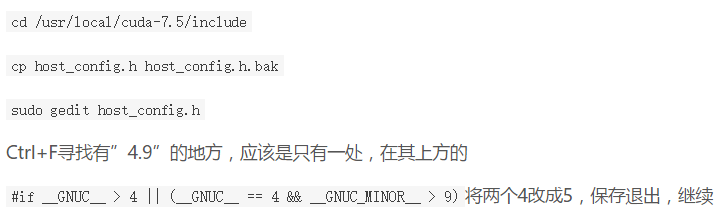
再sudo make all –j8

报了另外一个错：” memcpy” was not declared in this scope

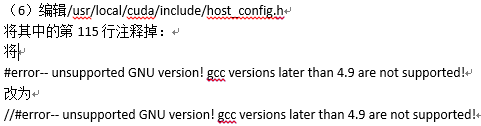
百度说是因为gcc版本太高了，要修改host\_config.h文件

有两种说法，

一、如下把4改为5



二、把这个error注释掉



然而这么做了还是报memcpy的错误

先放着了

参考链接：

linux下建立软链接

http://biyutong.iteye.com/blog/1445699

[编译cuda-7.5 samples文件报错](http://blog.csdn.net/lanxuecc/article/details/53422255)

<http://blog.csdn.net/lanxueCC/article/details/53422255>

**（四）安装opencv3.1**

（1）卸载原来安装的opencv3.1 （原来好像装出问题了，没管它，现在卸了重装）

参考链接：

[ubuntu14.04下OpenCV安装和卸载](http://blog.csdn.net/go_hyp/article/details/53422423)

<http://blog.csdn.net/go_hyp/article/details/53422423>

1. 进入opencv的源代码文件夹下的build（这是你在安装opencv时候自己命名的，cmake时候所在的目录）
2. 依次执行下面的代码，将已安装好的文件删除( 系统默认你的安装目录为： /usr/local/ )

make uninstall

cd ..

sudo rm -r build

sudo rm -r /usr/local/include/opencv2 /usr/local/include/opencv /usr/include/opencv /usr/include/opencv2 /usr/local/share/opencv /usr/local/share/OpenCV /usr/share/opencv /usr/share/OpenCV /usr/local/bin/opencv\* /usr/local/lib/libopencv

（这个命令的最后一个目录，我擅自改成了/usr/local/lib/libopencv\*，不知道是对还是错）

（ubuntu16.04好像自己装有默认的opencv的，我可能之前opencv3.1没装上，然后删了默认的opencv。。。）

（2）安装opencv3.1

*安装依赖*

参考链接：

<http://blog.csdn.net/jhszh418762259/article/details/52957495>

<http://blog.csdn.net/yonger_/article/details/52924516>

终端输入：

sudo apt-get install build-essential cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

*下载安装opencv3.1*

参考链接：

<http://blog.csdn.net/yonger_/article/details/52924516>

从官网(http://opencv.org/downloads.html)下载Opencv,并将其解压到你要安装的位置，假设解压到了/home/用户名/opencv-3.1.0（如我是 /home/vision/opencv-3.1.0）。

unzip opencv-3.1.0.zip -d ~

cd ~/opencv-3.1.0

mkdir build

cd build/

sudo cmake -D CMAKE\_BUILD\_TYPE=Release -D CMAKE\_INSTALL\_PREFIX=/usr/local/opencv3.1 ..

#( $(nproc)变量返回cpu核心，全速编译&&终端无输出，重定向到make.log文件&&make时间较长)

sudo make -j $(nproc) &> make.log

cat make.log #查看make结果

#(终端无输入，完毕后直接返回，结果可以cat make-install.log)

sudo make install &>make-install.log

cat make-install.log #查看install结果

（下面红色的两步可能/usr/local/lib的路径是错的，但在电脑上暂时还是这么配置的）

#加入动态链接库

sudo sh -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'

sudo ldconfig

sudo gedit /etc/profile #在最后加入两行

#若无gedit命令，sudo apt-get install gedit

PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/usr/local/lib/pkgconfig

export PKG\_CONFIG\_PATH

#pkg-config可以查看下面链接了解：

http://www.cppblog.com/colorful/archive/2012/05/05/173750.aspx

#最后

将ippicv中的64位lib文件复制到/usr/local/lib下

位于: ~ /opencv-3.1.0/3rdparty/ippicv/unpack/ippicv\_lnx/lib/intel64/libippicv.a

在~ /opencv-3.1.0/3rdparty/ippicv/unpack/ippicv\_lnx/lib/intel64目录下打开终端输入：

Sudo cp libippicv.a /usr/local/lib

注意：

参考链接：

<http://www.linuxidc.com/Linux/2016-07/132860.htm>

Cmake时可能出现一下错误

/usr/include/string.h: In function ‘void\* \_\_mempcpy\_inline(void\*, const void\*, size\_t)’: /usr/include/string.h:652:42: error: ‘memcpy’ was not declared in this scope return (char \*) memcpy (\_\_dest, \_\_src, \_\_n) + \_\_n;

这也是因为ubuntu16.04的个个g++版本太高的造成的，只需要在opencv-3.1.0目录下的CMakeList.txt 文件的开头加入

set(CMAKE\_CXX\_FLAGS “${CMAKE\_CXX\_FLAGS} -D\_FORCE\_INLINES”)

添加之后再次进行编译链接即可

*测试opencv3.1*

参考链接：

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

@ 创建工作目录

在 ~ 目录下（即/home/vision）新建文件夹 try

mkdir ~/try

cd ~/try

gedit DisplayImage.cpp

@ 编辑如下代码，保存退出

#include <stdio.h>

#include <opencv2/opencv.hpp>

using namespace cv;

int main(int argc, char\*\* argv )

{

if ( argc != 2 )

{

printf("usage: DisplayImage.out <Image\_Path>\n");

return -1;

}

Mat image;

image = imread( argv[1], 1 );

if ( !image.data )

{

printf("No image data \n");

return -1;

}

namedWindow("Display Image", WINDOW\_AUTOSIZE );

imshow("Display Image", image);

waitKey(0);

return 0;

}

@ 创建CMake编译文件

**gedit** CMakeLists.txt

写入如下内容：

cmake\_minimum\_required(**VERSION** 2.8)

project( **DisplayImage** )

find\_package( **OpenCV** REQUIRED )

add\_executable( **DisplayImage** DisplayImage.cpp )

target\_link\_libraries( **DisplayImage** ${OpenCV\_LIBS} )

由于opencv没有装在默认路径，所以要让CMake找到OpenCVConfig.cmake文件，这个文件指定了CMake要去哪里找OpenCV

参考链接：<http://blog.csdn.net/haizimin/article/details/51927386>

即在CMakeLists.txt中第一行后面加上

set(OpenCV\_DIR "/home/vision/opencv-3.1.0/build")

（将其添加在project(MyProjectName)之前）

我们期望使用哪个版本的Opencv，只要找到对应的OpenCVConfig.cmake文件，并且将其路径添加到工程的CMakeLists.txt中即可了

@编译：

在 try 目录下新建 build 文件夹

cd ~try

mkdir build

cmake ..

make

@执行

此时build 文件夹中已经产生了可执行文件DisplayImage，下载lena.jpg放在build文件夹下，运行./DisplayImage lena.jpg

@结果

显示一张图片



**（五）、在caffe中使用opencv3.1**

（1）将终端cd到要安装caffe的位置。

（2）从github上获取caffe：

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

注意：若没有安装Git，需要先安装Git：

sudo apt-get install git

（3）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：

sudo cp Makefile.config.example Makefile.config

（4）打开并修改配置文件：

sudo gedit Makefile.config #打开Makefile.config文件

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

d.重要的一项 :

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 /usr/local/opencv3.1/include

LIBRARY\_DIRS := $(PYTHON\_LIB) /usr/local/lib /usr/lib /usr/lib/x86\_64-linux-gnu /usr/lib/x86\_64-linux-gnu/hdf5/serial /usr/local/opencv3.1/lib

（这两句要比网上给的多红色字，是因为用的不是默认的opencv的安装路径，可以查看编译opencv-3.1.0的编译目录，本机为/home/vision/opencv-3.1.0/build/目录，找到unix-install/opencv.pc文件，opencv.pc文件及给出了安装的opencv的信息，可以得到这两个路径）

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.

（5）修改makefile文件

打开makefile文件，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）编辑/usr/local/cuda/include/host\_config.h

将其中的第115行注释掉：

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!

（7）编译

make all -j8 #-j根据自己电脑配置决定

编译过程中可能会出现如下错误：

错误1：

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

解决办法是将一些文件复制到/usr/local/lib文件夹下：

#注意自己CUDA的版本号！

sudo cp /usr/local/cuda-7.5/lib64/libcudart.so.7.5 /usr/local/lib/libcudart.so.7.5 && sudo ldconfig

sudo cp /usr/local/cuda-7.5/lib64/libcublas.so.7.5 /usr/local/lib/libcublas.so.7.5 && sudo ldconfig

sudo cp /usr/local/cuda-7.5/lib64/libcurand.so.7.5 /usr/local/lib/libcurand.so.7.5 && sudo ldconfig

错误2：

Usr/bin/Id:cannot find –lxxx错误

通常在软件编译时出现的usr/bin/ld:cannot find –lxxx的错误，主要的原因是库文件并没有导入到ld检索目录中

解决方式：

确认库文件是否存在，比如-l123, 在/usr/lib, /usr/local/lib,或者其他自定义的lib下有无lib123.so, 如果只是存在lib123.so.1, 那么可以通过ln -sv lib123.so.1 lib123.so，建立一个连接重建lib123.so.

（8）测试

make test -j4

make runtest -j4

如果运行之后出现下图，说明caffe配置成功。

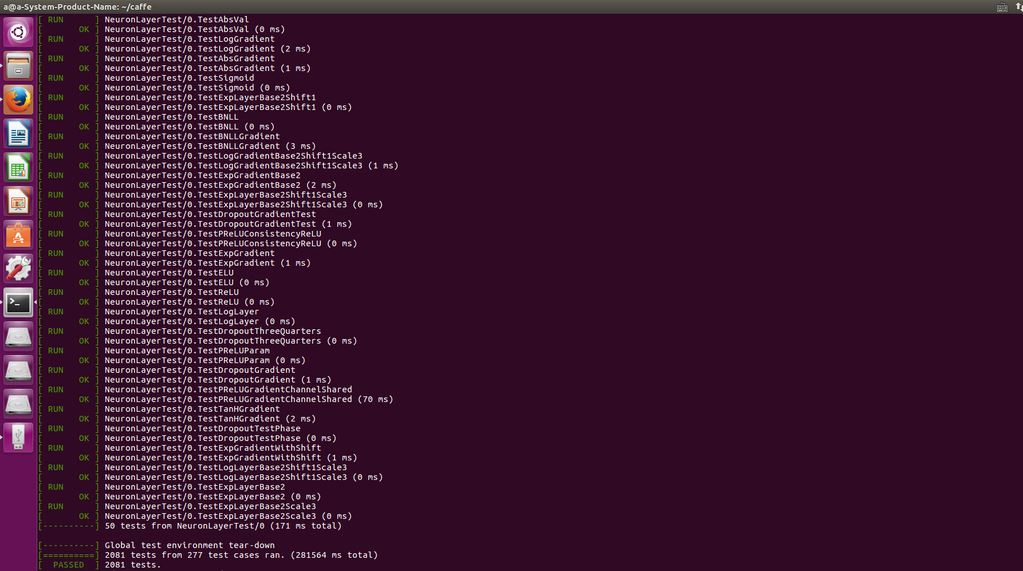


图6.caffe测试结果

到此caffe配置完毕！

注意：

运行make之后，运行可执行程序时，可能会提示找不到库的错误（not find libopencv-core.so.3.1之类），那时因为动态库目录不在程序搜索库目录的路径下，系统默认的动态库搜索路径有/usr/local/lib等，OpenCV4Tegra(opencv-2.4.12)生成的库都在/usr/local/lib目录，但opencv-3.1.0的库在/usr/local/opencv3.1/lib中，不在搜索路径中，因此，只需要添加其为搜索路径即可。添加方法为/etc/ld.so.conf.d/libc.conf中添加/usr/local/opencv3.1/lib。libc.conf文件内容如下：

#libc default configuration

/usr/local/lib

/usr/local/opencv3.1/lib

添加完保存退出，记得在终端输入

sudo ldconfig

使之生效！！！

参考链接：<http://blog.csdn.net/haizimin/article/details/51927386>

（要知道opencv-3.1.0的库在哪里，可以查看编译opencv-3.1.0的编译目录，本机为/home/vision/opencv-3.1.0/build/目录，找到unix-install/opencv.pc文件，opencv.pc文件及给出了安装的opencv的信息）

（六）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

caffe/data/mnist/目录下会多出训练集图片、训练集标签、测试集图片和测试集标签等4个文件

3.将其转换成Lmdb数据库格式

./examples/mnist/create\_mnist.sh

必须要在caffe根目录下运行create\_mnist.sh

此时在/caffe/examples/mnist/目录下生成mnist\_test\_lmdb和mnist\_train\_lmdb两个LMDB格式的训练集和测试集

4.训练网络

LeNet-5模型描述在/caffe/examples/mnist/lenet\_train\_test.prototxt

Solver配置文件在/caffe/examples/mnist/lenet\_solver.prototxt

执行文件在/caffe/examples/mnist/train的lenet.sh

当前位置在caffe根目录下

终端输入

./examples/mnist/train\_lenet.sh

即可开始训练，准确率一般都有0.99以上

训练的时候可以看到损失与精度数值，如下图：

图7.MNIST数据集训练

可以看到最终训练精度是0.9914。