Deep learning Setup

2017-11-23 07:24

Contents

Description : *Installing tools for NVIDIA GPU & creating a Deeplearing Setup (caffe)*

Table of Contents

[TOC]

Specifications

- OS - Ubuntu 16.04
- Graphics - Nvidia Optimus (GTX1070 + IntelHD)

After fresh installation of Ubuntu 16.04 “Disabling nouveau driver”

```
sudo apt-get update
sudo apt-get upgrade
```

Install one editor which you like the most

```
sudo apt-get install vim
```

Before installing Drivers into your Hybrid system first we need to disable the nouveau (default display driver comes with linux) because it comes above all. Press 'CTRL+Alt+F1' you will enter shell now enter your username and password credentials and then continue

```
sudo vim /etc/modprobe.d/blacklist.conf
```

Now add the following lines in the end of the file (Save & Exit)

```
blacklist nouveau
blacklist lbm-nouveau
options nouveau modeset=0
alias nouveau off
alias lbm-nouveau off
```

Now get back to terminal and enter the following and later update the kernel and reboot

```
echo options nouveau modeset=0 | sudo tee -a /etc/modprobe.d/nouveau-kms.conf
sudo update-initramfs -u
sudo reboot
```

Intel+Nvidia GPU working setup using bumblebee and primus

Basically using primus we can switch between the graphics and we take the help of bumblebee to make it smooth and we also take the help of a GUI indicator to make the transistions more simple.

Add the following repositories

```
sudo apt-add-repository ppa:graphics-drivers
sudo apt-add-repository ppa:bumblebee/testing
sudo apt-add-repository ppa:nilarimogard/webupd8
sudo apt-get update
```

Go to Settings » Software & Updates » Additional Drivers Select Nvidia-378 driver (because it is stable and it worked for me) and click on Apply and then Restart the system.

After Restarting you can see the Nvidia-driver being selected as the display driver which previously was Xorg's nouveau.For further conforamtion you can check with the following command and the output will be comething like this.

```
nvidia-smi

[root@localhost release]# nvidia-smi
Wed Sep 26 23:16:16 2012
+------------------------------------------------------+
| NVIDIA-SMI 3.295.41   Driver Version: 295.41         |
```

```
|-------------------------------+----------------------+----------------------+
| Nb.  Name                     | Bus Id        Disp.  | Volatile ECC SB / DB |
| Fan   Temp   Power Usage /Cap | Memory Usage         | GPU Util. Compute M. |
|===============================+======================+======================|
| 0.  Tesla C2050               | 0000:05:00.0  On     |         0          0 |
|  30%   62 C  P0    N/A /  N/A |   3%   70MB / 2687MB |   44%     Default    |
|-------------------------------+----------------------+----------------------|
| Compute processes:                                                GPU Memory |
|  GPU  PID     Process name                                        Usage      |
|=============================================================================|
|  0.  7336     ./align                                                61MB  |
+-----------------------------------------------------------------------------+
```

Now either use command line or Synaptics Manager to install the requirements inorder to keep it simple I shall use Synaptic Manager to demonstrate 1. Enter bumblebee in the search dialog then you will be able to see bumblebee, bumblebee-nvidia, primus select all the three and Mark up for Installation and then click Apply 2. After installing above three we check for bbswitch-dkms in search dialog box.It can be seen as already installed (if not then install it) We get back to our terminal and take the help of prime to select Intel Graphics as primary

```
sudo prime-select intel
sudo reboot
```

Now enter prime-indicator(/plus) in search and mark up for installation and restart your system. Inorder to make the bumblebee and bbswitch to take care of your system and use latest nvidia driver which has been installed go to the following file and edit

```
sudo vim /etc/bumblebee/bumblebee.conf
```

Now update the following contents

```
Driver= should be changed to
Driver=nvidia
```

In [driver-nvidia] section replace all nvidia-current terms to nvidia-378(If you have installed 378 or else replace it with the driver number which has been installed) and also in the same section replace

```
PMMethod=auto
PMMethod=bbswitch
```

Now restart

```
sudo reboot
```

We are done with our Nvidia driver installation and we also can switch between Intel and Nvidia Graphics which will help with saving the battery

Installation of CUDA-8.0 and verifying if it works or not

Now switch to Nvidia Graphics and download the run file. In my case I have downloaded `cuda_8.0.61_375.26_linux.run` file because previous ones need a below 4.9 gcc compiler but when it comes to 16.04 by defualt it installs gcc-5.0 and the installation of Caffe requires a gcc-5 compiler to work (portbuf). After downloading go to the specific folder and then

```
chmod 755 cuda_8.0.61_375.26_linux.run
sudo ./cuda_8.0.61_375.26_linux.run
```

Enter no when asked to install Nvidia driver and rest all can be entered as Yes. Don't worry if it shows something like this

```
===========

= Summary =

===========

Driver: Not Selected

Toolkit: Installed in /usr/local/cuda-8.0

Samples: Installed in /home/username, but missing recommended libraries

Please make sure that

- PATH includes /usr/local/cuda-8.0/bin

- LD_LIBRARY_PATH includes /usr/local/cuda-8.0/lib64, or, add /usr/local/cuda-8.0/lib64 to /

To uninstall the CUDA Toolkit, run the uninstall script in /usr/local/cuda-8.0/bin

Please see CUDA_Installation_Guide_Linux.pdf in /usr/local/cuda-8.0/doc/pdf for detailed inf

***WARNING: Incomplete installation! This installation did not install the CUDA Driver. A dr
```

Now (Optional not required)

```
sudo modprobe nvidia
```

```
sudo vim /etc/profile
```

```
and enter in the end

export PATH=/usr/local/cuda-8.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-8.0/lib64:$LD_LIBRARY_PATH
```

Now save & exit

```
sudo ldconfig
```

The setup is complete for CUDA now it's time to verify this

```
sudo apt-get install freeglut3-dev build-essential libx11-dev libxmu-dev libxi-dev  libglu1-
```

Now go to the location where samples folder is installed by default it is installed at ~/

```
cd into the samples directory
cd 1_Utilities/deviceQuery
sudo make
sudo ./deviceQuery
```

it should show something like this

```
Device 0: Quadro M1000M
  CUDA Driver Version / Runtime Version          8.0 / 7.5
  CUDA Capability Major/Minor version number:    5.0
  Total amount of global memory:                 2002 MBytes (2099642368 bytes)
  ( 4) Multiprocessors, (128) CUDA Cores/MP:     512 CUDA Cores
  GPU Max Clock rate:                            1072 MHz (1.07 GHz)
  Memory Clock rate:                             2505 Mhz
  Memory Bus Width:                              128-bit
  L2 Cache Size:                                 2097152 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4096, 4(
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 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 1 copy engine(s)
  Run time limit on kernels:                     No
  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
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
```

```
    < Default (multiple host threads can use with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 8.0, CUDA Runtime Version = 7.5, Nu
Result = PASS
```

Similarly we conduct the bandwidth test which will also show PASS something similar to above and then we confirm its installation. If it shows fail then there is some error in CUDA installation.

```
cd ..
cd bandwidthTest
sudo make
sudo ./bandwidthTest
```

With this we are ready with our system to use CUDA and NVIDIA GPU.

Installation of CUDNN (Easiest of All I should say)

Go to Nvidia's site and download cuDNN (I myself used cuDNN 5.1) you will get almost 98MB file now extract the contents and go to the extracted folder

```
cd /cuda
sudo cp -P include/cudnn.h /usr/include
sudo cp -P lib64/libcudnn* /usr/lib/x86_64-linux-gnu/
sudo chmod a+r /usr/lib/x86_64-linux-gnu/libcudnn*
```

Installing OpenCV3.2 (may not be complete but is enough for working with caffe)

In Ubuntu 16.04, install the dependencies first and then build the OpenCV 3.2 from source.

```
sudo apt-get install --assume-yes build-essential cmake git
sudo apt-get install --assume-yes pkg-config unzip ffmpeg qtbase5-dev python-dev python3-dev
sudo apt-get install --assume-yes libopencv-dev libgtk-3-dev libdc1394-22 libdc1394-22-dev l
sudo apt-get install --assume-yes libavcodec-dev libavformat-dev libswscale-dev libxine2-dev
sudo apt-get install --assume-yes libv4l-dev libtbb-dev libfaac-dev libmp3lame-dev libopenco
sudo apt-get install --assume-yes libvorbis-dev libxvidcore-dev v4l-utils python-vtk
sudo apt-get install --assume-yes liblapacke-dev libopenblas-dev checkinstall
sudo apt-get install --assume-yes libgdal-dev
```

Download the latest source archive for OpenCV 3.2 from https://github.com/opencv/opencv/archive/3.2.0.zip

Enter the unpacked directory. Execute

```
mkdir build
cd build/
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D FORCE_VTK=ON -D WITH
make -j $(($(nproc) + 1))
```

To complete the installation execute the following

```
sudo make install
sudo /bin/bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
sudo ldconfig
sudo apt-get update
```

Verify installation with

```
python
>>> import cv2
```

If it doesn't work then there is some error with OpenCV3.2 installation. With this we are done with our OpenCV3 installation next we jump into Caffe installation.

Installing Caffe in 16.04 along with support of OpenCV3 & GPU (CUDA+cuDNN)

For pre-requisites we execute the following lines

```
sudo apt-get update
sudo apt-get upgrade

sudo apt-get install -y build-essential cmake git pkg-config
sudo apt-get install -y libprotobuf-dev libleveldb-dev libsnappy-dev libhdf5-serial-dev prot
sudo apt-get install -y libatlas-base-dev
sudo apt-get install -y --no-install-recommends libboost-all-dev
sudo apt-get install -y libgflags-dev libgoogle-glog-dev liblmdb-dev

# (Python general)
sudo apt-get install -y python-pip

# (Python 2.7 development files)
sudo apt-get install -y python-dev
sudo apt-get install -y python-numpy python-scipy
```

We next clone the Caffe repo.

```
cd ~
git clone https://github.com/BVLC/caffe.git
```

We make changes in Makefile.config and Makefile and configure to proceed the Caffe installation smoothly.

```
cd ~/caffe
cp Makefile.config.example Makefile.config
sudo vim Makefile.config
```

We now make the following changes and configure the copied Makefile.config (by uncommenting and editing the following lines in the file)

```
USE_CUDNN := 1
OPENCV_VERSION := 3

Change
CUDA_DIR := /usr/local/cuda
to
CUDA_DIR := /usr/local/cuda-8.0

PYTHON_INCLUDE := /usr/include/python2.7 /usr/lib/python2.7/dist-packages/numpy/core/include
WITH_PYTHON_LAYER := 1
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
```

Somtimes the PYTHON_INCLUDE may differ in some systems check for the presence of numpy core files

```
PYTHON_INCLUDE := /usr/include/python2.7 /usr/local/lib/python2.7/dist-packages/numpy/core/i
WITH_PYTHON_LAYER := 1
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
```

Now edit Makefile (above we edited Makefile.config)

```
cd ~/caffe
sudo vim Makefile

Change
NVCCFLAGS += -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON_FLAGS)
to
NVCCFLAGS += -D_FORCE_INLINES -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON_FLAGS)
```

Now we install some python requirements by taking pip's help

```
cd ~/caffe/python
for req in $(cat requirements.txt); do sudo -H pip install $req --upgrade; done
```

Now it's time to check make and check caffe's installation

```
cd ~/caffe
make all -j $(($(nproc) + 1))
make test
make runtest

make pycaffe
make distribute

sudo vim ~/.bashrc
add the follwing line to the file
export PYTHONPATH=~/caffe/python:$PYTHONPATH
source ~/.bashrc
```

Now you can verify your installation with (for python2.7)

```
python
>>> import caffe
```

Now, we are ready with are our Deep Learning setup, Get Going!