1. ~~Please use Virtualbox to set up your virtual machine with Ubuntu 18.04. (Only 18.04 was tested)~~
   ~~https://www.toptechskills.com/linux-tutorials-courses/how-to-install-ubuntu-1804-bionic-virtualbox/~~

2. Install CUDAtoolkit 8.0

   2.1.Before install CUDA, you may need to check your gcc/g++ version, if it is newer than 5.3, you may need to downgrade it.
   Open a terminal and enter the command to check gcc version,
   gcc –version

   i.e., Set gcc/g++ to 4.8 (if you need to)
   (1) Download the gcc/g++ 4.8
   sudo apt-get update
   sudo apt-get upgrade
   sudo apt install gcc-4.8
   sudo apt install g++-4.8
   (2) Set the symbolic link
   sudo ln -s /usr/bin/gcc-5 /usr/local/bin/gcc
   sudo ln -s /usr/bin/g++-5 /usr/local/bin/g++

   2.2 Download and Install CUDA 8.0
   https://developer.nvidia.com/cuda-80-ga2-download-archive
   Please select Linux, x86_64, Ubuntu, 16.04, then runfile(local), respectively.
   Then you could click on the download button to download it, you may need a NVIDIA account to perform the downloading.
   After finishing the downloading, change to your downloaded file directory, and enter the command,
   sudo sh cuda_8.0.61_375.26_linux.run

   You might need to make it executable before running it.

   sudo chmod +x cuda_8.0.61_375.26_linux.run

   If you meet error messages regarding something like "Can't locate InstallUtils.pm in @INC", do the following:

   apt install libfile-find-rule-perl-perl
   mkdir cuda-8
   sudo sh cuda_8.0.61_375.26_linux.run --noexec --target cuda-8
   cp cuda-8/InstallUtils.pm /usr/share/perl5

   Then re-tun the install command.

You may use space key to quickly jump the user agreements, then, follow the command-line prompts.

Please note that, for the question about install the graphic driver, enter ‘no’ or ‘n’, because we are going to use gpgpu-sim as our ‘gpu’.

For the question about installation location, just press enter to install at the default location.

For the question about create a symbolic link for CUDA, please enter yes or y.

Please make sure there is no error message after installation, it is fine if there are some warning messages.

Then you need to add CUDA8.0 PATH to your .bashrc file.

Use vim or any edit tools in LINUX,

vim ~/.bashrc

And add those 3 lines to the end of the file, (press i to insert)

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

then press esc, and enter ‘:wq’ to save and quit the file

then enter the command

source ~/.bashrc

to exactly save the change.

Now you can use the following command to check the installation of CUDA

nvcc -V

And following are the expected outputs

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2016 NVIDIA Corporation
Built on Tue_Jan_10_13:22:03_CST_2017
Cuda compilation tools, release 8.0, V8.0.61
```

Now we are done for the CUDA installation, and you could use the same method to download the patch for CUDA 8.0.

We also need to set symbolic for gcc/g++ inside CUDA,

sudo ln -s /usr/bin/gcc-5 /usr/local/cuda-9.1/bin/gcc
sudo ln -s /usr/bin/g++-5 /usr/local/cuda-9.1/bin/g++

3. Install GPGPU-SIM

https://github.com/gpgpu-sim/gpgpu-sim_distribution

Please start with step 1 and install all the dependencies and use git to download the gpgpusim source codes to your machine.

git clone https://github.com/gpgpu-sim/gpgpu-sim_distribution.git

If you got complains about some dependencies or libraries are not supported or outdated, just skip those or google the new version's name.

For instance, as "AerialVision dependencies", you could do,

sudo apt-get install python-pmw python-ply python-numpy libpng-dev python-matplotlib

and for "CUDA SDK dependencies", you could do,

sudo apt-get install libxi-dev libxmu-dev freeglut3-dev

Now we have installed the gpgpu-sim, to build the GPU config file with gpgpu-sim,
Change your directory to gpgpu-sim, and use the following command,

source setup_enviroment

make

If you want to build again in future, just

make clean

source setup_enviroment

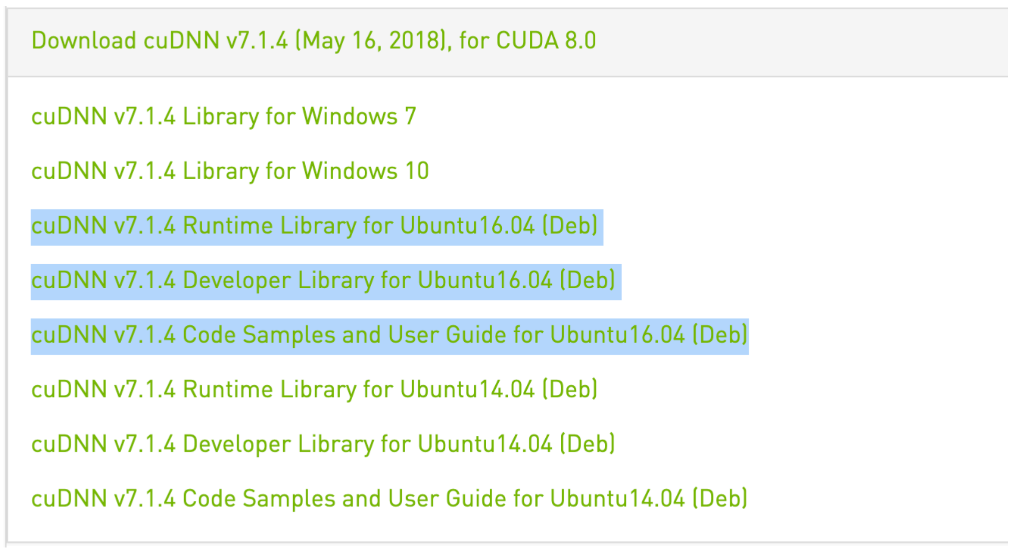
make

4. Install cuDNN 7.1.4

https://developer.nvidia.com/rdp/cudnn-archive

You will need NIVIDIA account to download cudnn, so please create an account with NIVIDA and log in.

Then, please find cuDNN v7.1.4 for CUDA 8.0 and select following 3 to download,



Download cuDNN v7.1.4 (May 16, 2018), for CUDA 8.0

cuDNN v7.1.4 Library for Windows 7

cuDNN v7.1.4 Library for Windows 10

cuDNN v7.1.4 Runtime Library for Ubuntu16.04 (Deb)

cuDNN v7.1.4 Developer Library for Ubuntu16.04 (Deb)

cuDNN v7.1.4 Code Samples and User Guide for Ubuntu16.04 (Deb)

cuDNN v7.1.4 Runtime Library for Ubuntu14.04 (Deb)

cuDNN v7.1.4 Developer Library for Ubuntu14.04 (Deb)

cuDNN v7.1.4 Code Samples and User Guide for Ubuntu14.04 (Deb)

After downloading, change to your directory contains cudnn Debian file.

First, install the run time library,

sudo dpkg -i libcudnn7_7.1.4.18-1+cuda8.0_amd64.deb

Then, install the dev library,

sudo dpkg -i libcudnn7-dev_7.1.4.18-1+cuda8.0_amd64.deb

Finally, install the code samples/docs,

sudo dpkg -i libcudnn7-doc_7.1.4.18-1+cuda8.0_amd64.deb

After installing cudnn, you could use following command to find cudnn.h

locate cudnn.h,

then add the following line into .bashrc file with the path of cudnn.h

```
export CUDNN_INCLUDE_DIR=/usr/include/cudnn.h
```

5. Run mnistCUDNN with GPGPU-SIM

- Copy ‘cudnn_samples_v7’ to home directory.

  ```
  cp -r -a /usr/src/cudnn_samples_v8/ $HOME
  ```

- Copy the SM7_TITANV configuration to the mnistCUDNN directory (check the tested-cfgs folder for other gpu’s configs. We will use ‘SM7_TITANV’).

  ```
  cd  ~/cudnn_samples_v7/mnistCUDNN

  cp -a ~/gpgpu-sim_distribution/configs/tested-cfgs/SM7_TITANV/* .
  ```

- Edit ~/cudnn_samples_v7/mnistCUDNN/Makefile, line 164(LIBRARIES ~) should be changed to follow lines. It changes dynamic linking to Static linking.

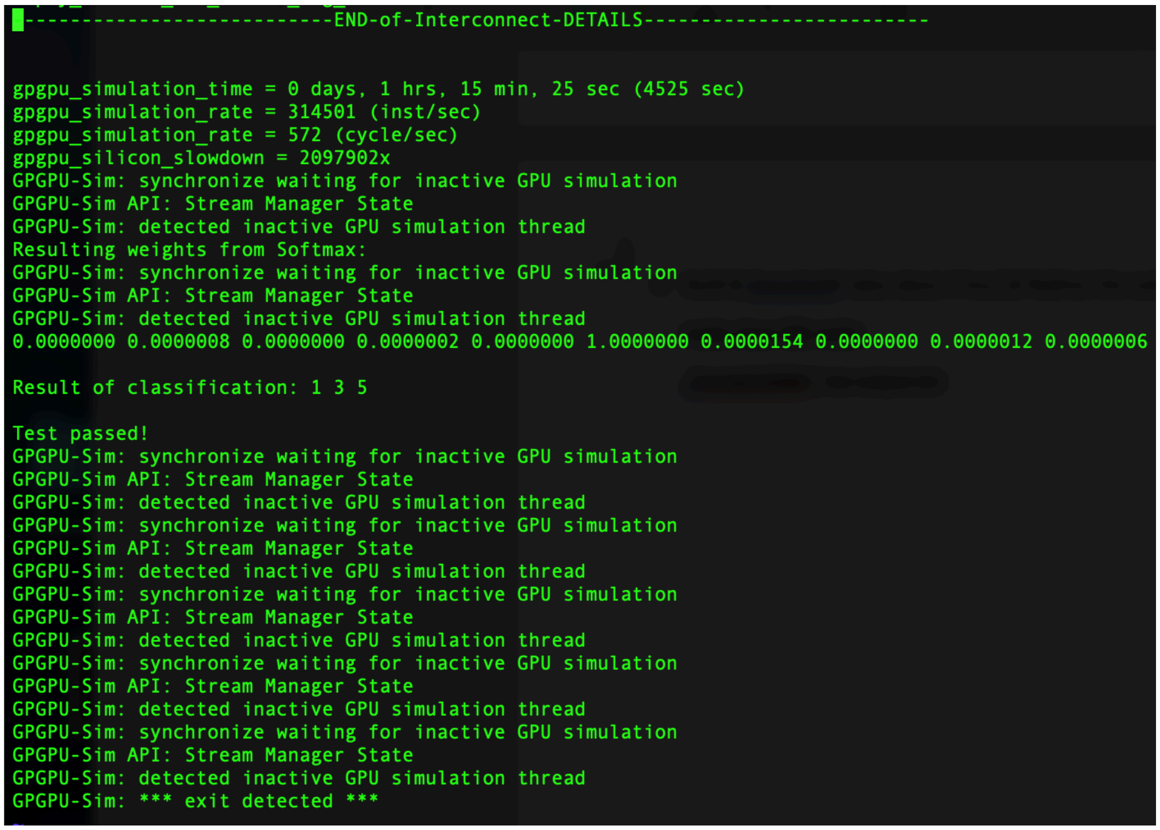
  ```
  LIBRARIES += -LFreeImage/lib/$(TARGET_OS)/$(TARGET_ARCH) -LFreeImage/lib/$
  (TARGET_OS) -lcudart -lcublas_static -lcudnn_static_v7 -lculibos -lfreeimage -lstdc++ -lm
  -ldl -lpthread
  ```

- Build & Execution(It takes about 1 hour) & Redirect the output to output.txt

  ```
  make clean && make

  ./mnistCUDNN | tee output.txt
  ```

  Looking for “Test passed!”



```
-------------------------END-of-Interconnect-DETAILS-------------------------


gpgpu_simulation_time = 0 days, 1 hrs, 15 min, 25 sec (4525 sec)
gpgpu_simulation_rate = 314501 (inst/sec)
gpgpu_simulation_rate = 572 (cycle/sec)
gpgpu_silicon_slowdown = 2097902x
GPGPU-Sim: synchronize waiting for inactive GPU simulation
GPGPU-Sim API: Stream Manager State
GPGPU-Sim: detected inactive GPU simulation thread
Resulting weights from Softmax:
GPGPU-Sim: synchronize waiting for inactive GPU simulation
GPGPU-Sim API: Stream Manager State
GPGPU-Sim: detected inactive GPU simulation thread
0.0000000 0.0000008 0.0000000 0.0000002 0.0000000 1.0000000 0.0000154 0.0000000 0.0000012 0.0000006

Result of classification: 1 3 5

Test passed!
GPGPU-Sim: synchronize waiting for inactive GPU simulation
GPGPU-Sim API: Stream Manager State
GPGPU-Sim: detected inactive GPU simulation thread
GPGPU-Sim: synchronize waiting for inactive GPU simulation
GPGPU-Sim API: Stream Manager State
GPGPU-Sim: detected inactive GPU simulation thread
GPGPU-Sim: synchronize waiting for inactive GPU simulation
GPGPU-Sim API: Stream Manager State
GPGPU-Sim: detected inactive GPU simulation thread
GPGPU-Sim: synchronize waiting for inactive GPU simulation
GPGPU-Sim API: Stream Manager State
GPGPU-Sim: detected inactive GPU simulation thread
GPGPU-Sim: synchronize waiting for inactive GPU simulation
GPGPU-Sim API: Stream Manager State
GPGPU-Sim: detected inactive GPU simulation thread
GPGPU-Sim: *** exit detected ***
```

Up to now, we have installed CUDA 8.0, gpgpu-sim and cudnn 7.1.4 successfully, and they could work together.

Reference: https://wheatbeer.medium.com/how-to-run-cudnn-with-gpgpu-sim-9e30447ee5a9