Install TensorFlow with GPU for Windows 10

by **Nitish S. Mutha**

January 22, 2017 in Tensorflow

On November 9, 2015 Google open sourced a software library called TensorFlow. TensorFlow is a software library used for Machine learning and Deep learning for numerical computation using data flow graphs. It can run on multiple CPUs and GPUs.
Since deep learning algorithms runs on huge data sets, it is extremely beneficial to run these algorithms on CUDA enabled Nvidia GPUs to achieve faster execution.

When I wanted to install TensorFlow GPU version on my machine, I browsed through internet and tensorflow.org for steps to download and setup. I could not find any good and clear source for setting up TensorFLow on local machine with GPU support for Windows. Most of the information available online was for Linux or Mac OS. Most search results online said there is no support for TensorFlow with GPU on Windows yet and few suggested to use virtual machines on Windows but again the would not utilize GPU.

Then I decided to explore myself and see if that is still the case or has Google recently released support for TensorFlow with GPU on Windows. After refering few pages on tensorflow.org I was able to setup TensorFlow GPU version on my Windows machine with ease. So now it is possible to have TensorFlow running on Windows with GPU support.

Requirements

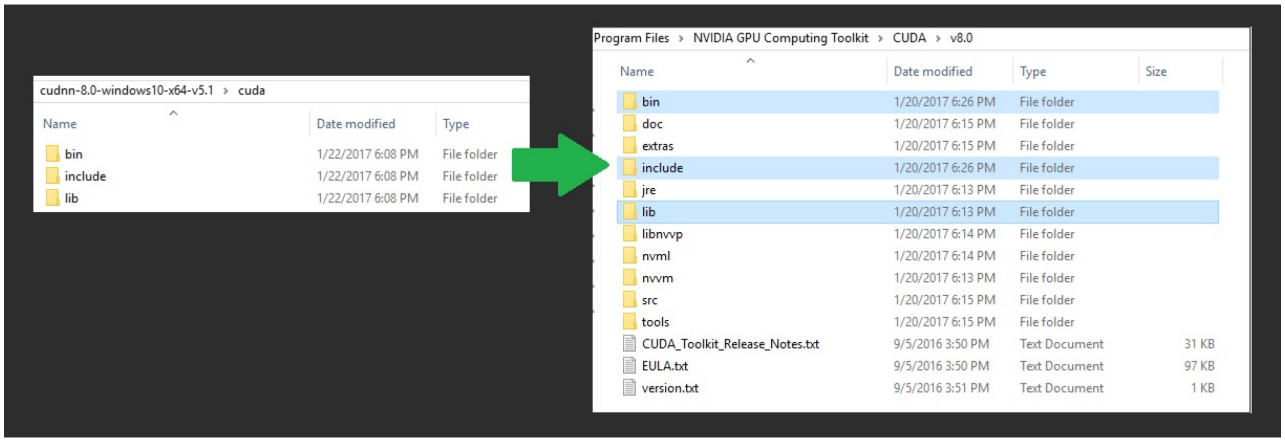
- Python 3.5
- Nvidia CUDA GPU. You can check here if your GPU is CUDA compatible.

Setting up your Nvidia GPU

You need to install Cuda Toolkit 8.0 and cuDNN v5.1 as the GPU version works best with these.

Download and install CUDA Toolkit
Toolkit version 8.0 or above: https://developer.nvidia.com/cuda-downloads
Example installation directory:
`C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0`

Download and install cuDNN
cuDNN version 5.1 library for Windows 10: https://developer.nvidia.com/cudnn
You would need to signup at Nvidia in order to download these files. Now extract the cuDNN files into your Toolkit directory.



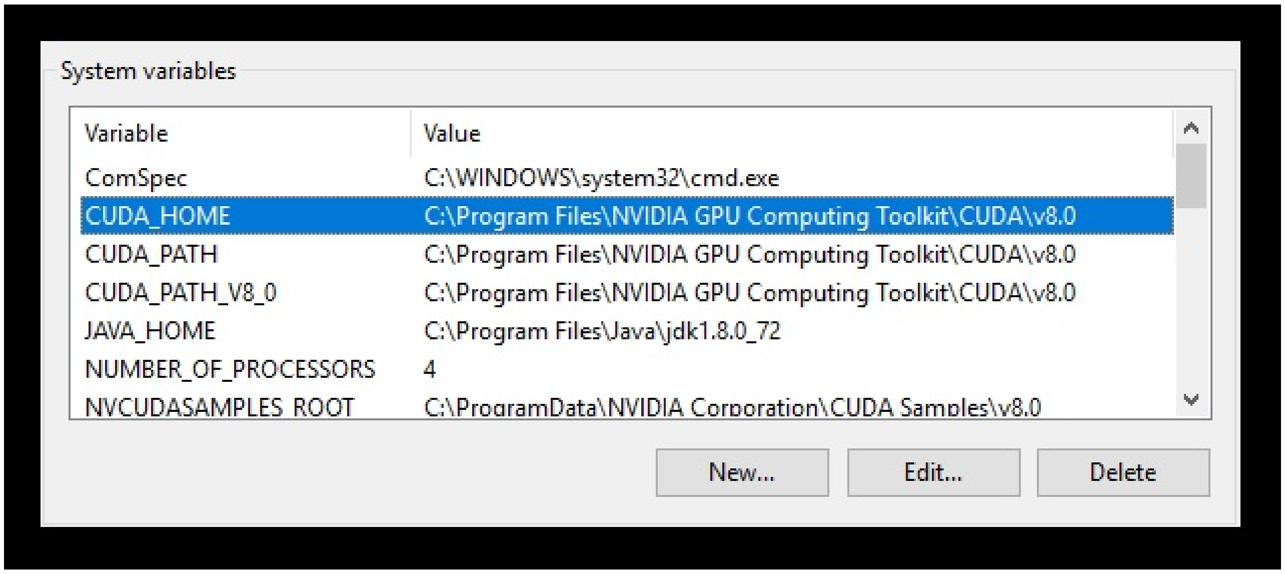
cudnn-8.0-windows10-x64-v5.1 › cuda

Name	Date modified	Type
bin	1/22/2017 6:08 PM	File folder
include	1/22/2017 6:08 PM	File folder
lib	1/22/2017 6:08 PM	File folder

Program Files › NVIDIA GPU Computing Toolkit › CUDA › v8.0

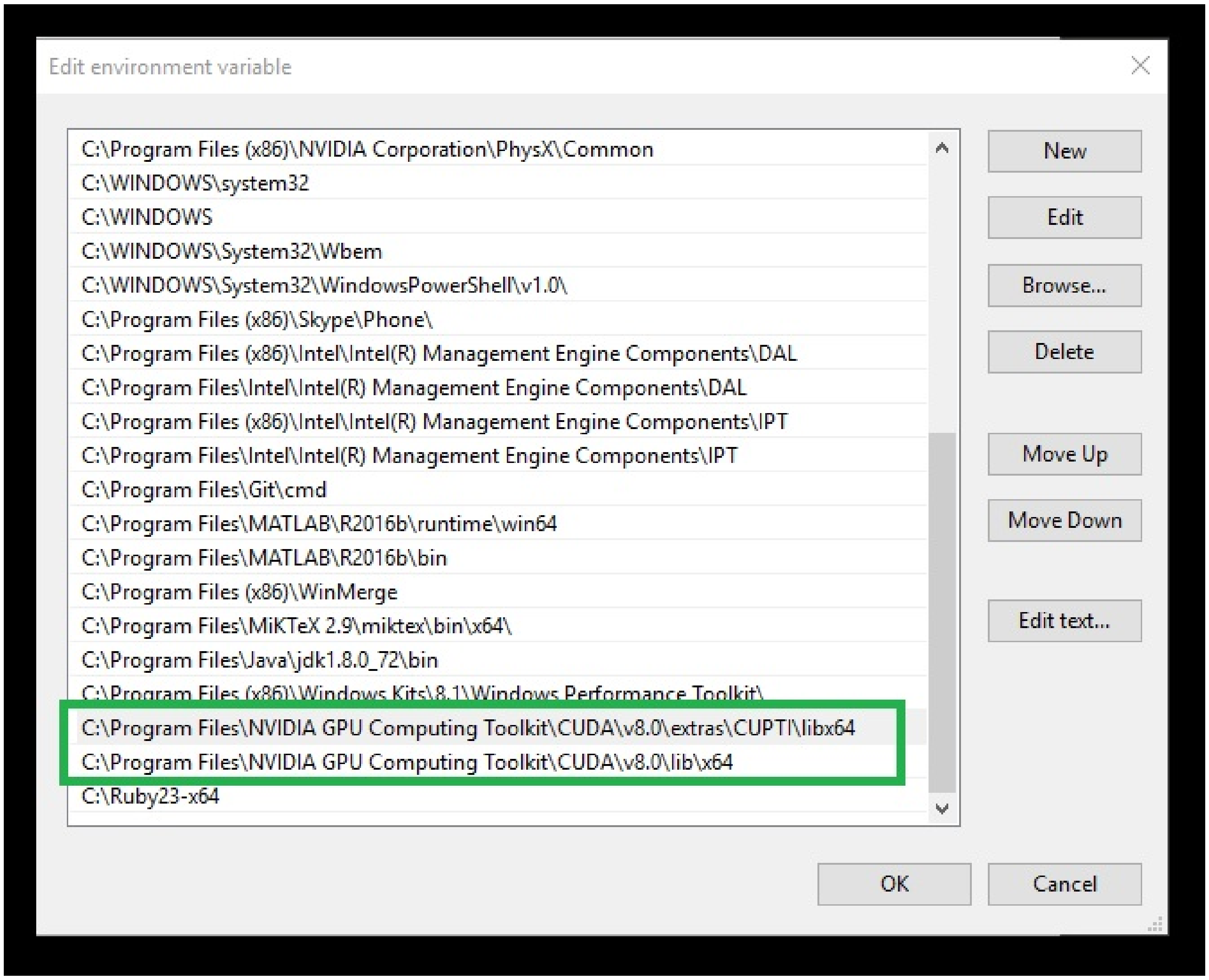
Name	Date modified	Type	Size
bin	1/20/2017 6:26 PM	File folder	
doc	1/20/2017 6:15 PM	File folder	
extras	1/20/2017 6:15 PM	File folder	
include	1/20/2017 6:26 PM	File folder	
jre	1/20/2017 6:13 PM	File folder	
lib	1/20/2017 6:13 PM	File folder	
libnvvp	1/20/2017 6:14 PM	File folder	
nvml	1/20/2017 6:14 PM	File folder	
nvvm	1/20/2017 6:13 PM	File folder	
src	1/20/2017 6:15 PM	File folder	
tools	1/20/2017 6:15 PM	File folder	
CUDA_Toolkit_Release_Notes.txt	9/5/2016 3:50 PM	Text Document	31 KB
EULA.txt	9/5/2016 3:50 PM	Text Document	97 KB
version.txt	9/5/2016 3:51 PM	Text Document	1 KB

Environmental variables
Ensure after installing CUDA toolkit, the CUDA_HOME is set in the environmental variables. If not then you need to add it manually..


System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
CUDA_HOME	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0
CUDA_PATH	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0
CUDA_PATH_V8_0	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_72
NUMBER_OF_PROCESSORS	4
NVCUDASAMPLES_ROOT	C:\ProgramData\NVIDIA Corporation\CUDA Samples\v8.0

And path variables as..


Edit environment variable

```
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
C:\WINDOWS\system32
C:\WINDOWS
C:\WINDOWS\System32\Wbem
C:\WINDOWS\System32\WindowsPowerShell\v1.0\
C:\Program Files (x86)\Skype\Phone\
C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL
C:\Program Files\Intel\Intel(R) Management Engine Components\DAL
C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT
C:\Program Files\Intel\Intel(R) Management Engine Components\IPT
C:\Program Files\Git\cmd
C:\Program Files\MATLAB\R2016b\runtime\win64
C:\Program Files\MATLAB\R2016b\bin
C:\Program Files (x86)\WinMerge
C:\Program Files\MiKTeX 2.9\miktex\bin\x64\
C:\Program Files\Java\jdk1.8.0_72\bin
C:\Program Files (x86)\Windows Kits\8.1\Windows Performance Toolkit\
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0\extras\CUPTI\libx64
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0\lib\x64
C:\Ruby23-x64
```

Install Anaconda

We will install Anaconda as it helps us to easily manage separate environments for specific distributions of Python, without disturbing the version of python installed on your system.

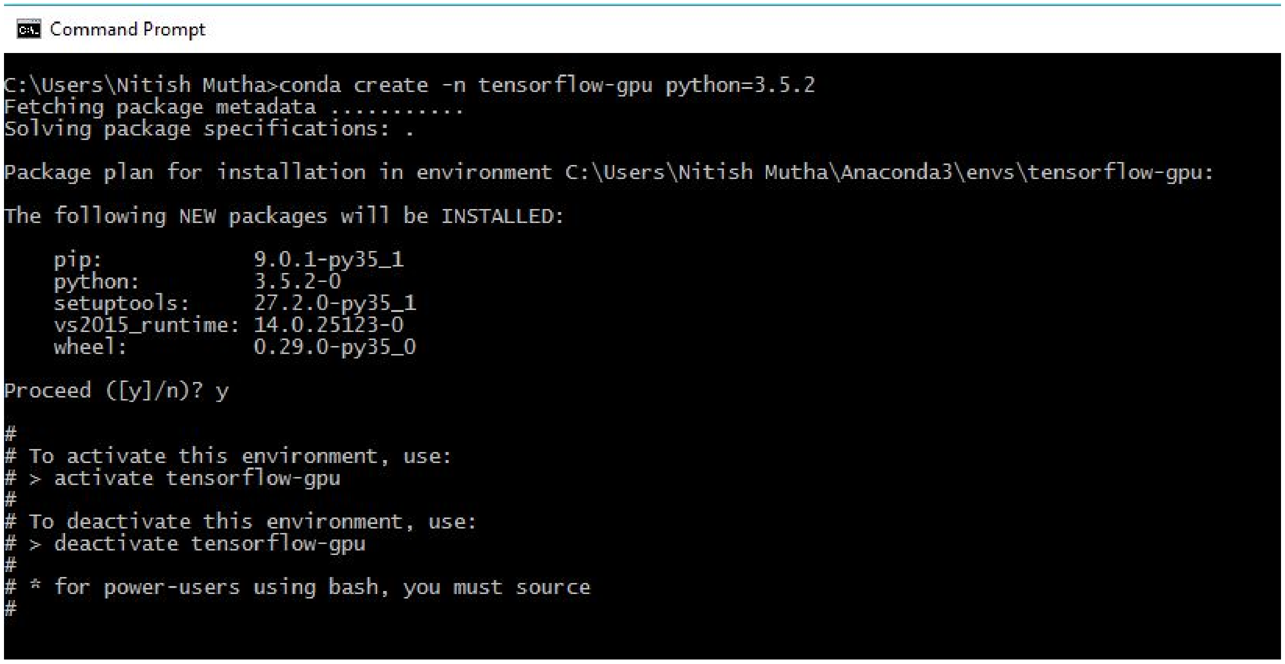
Download and install

Anaconda

Create conda environment

Create new environment, with the name tensorflow-gpu and python version 3.5.2

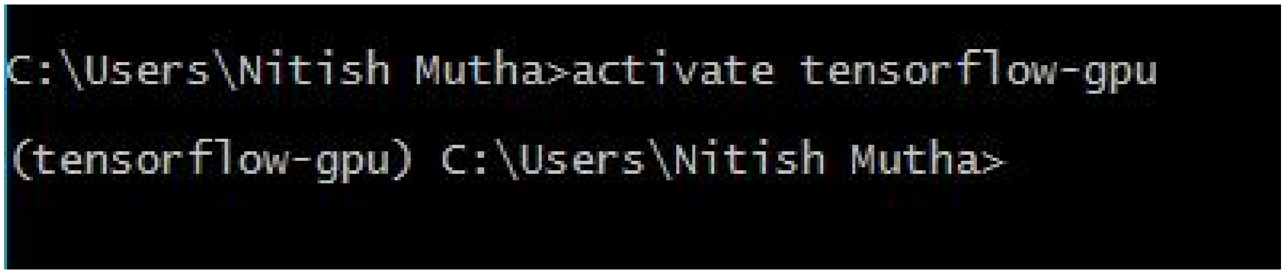
`conda create -n tensorflow-gpu python=3.5.2`



Command Prompt

```
C:\Users\Nitish Mutha>conda create -n tensorflow-gpu python=3.5.2
Fetching package metadata ...........
Solving package specifications: .

Package plan for installation in environment C:\Users\Nitish Mutha\Anaconda3\envs\tensorflow-gpu:

The following NEW packages will be INSTALLED:

    pip:            9.0.1-py35_1
    python:         3.5.2-0
    setuptools:     27.2.0-py35_1
    vs2015_runtime: 14.0.25123-0
    wheel:          0.29.0-py35_0

Proceed ([y]/n)? y

#
# To activate this environment, use:
# > activate tensorflow-gpu
#
# To deactivate this environment, use:
# > deactivate tensorflow-gpu
#
# * for power-users using bash, you must source
#
```

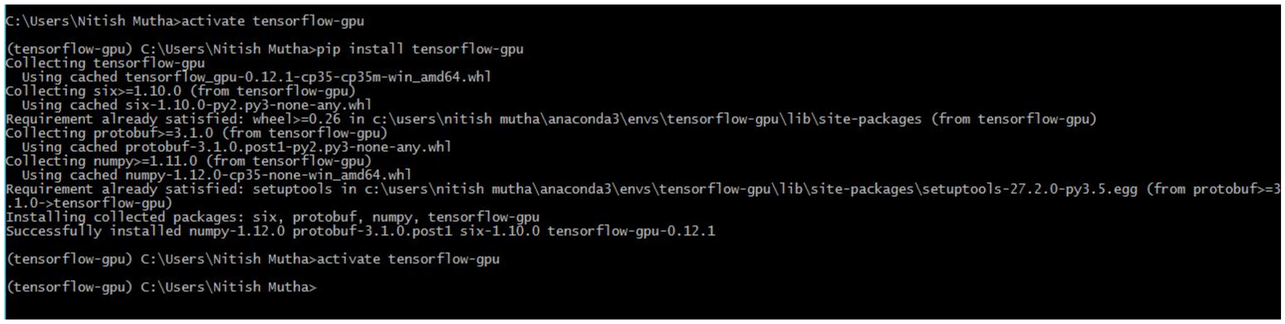
Activate the environment `activate tensorflow-gpu`



```
C:\Users\Nitish Mutha>activate tensorflow-gpu

(tensorflow-gpu) C:\Users\Nitish Mutha>
```

Install tensorFlow

`pip install tensorflow-gpu`



```
C:\Users\Nitish Mutha>activate tensorflow-gpu

(tensorflow-gpu) C:\Users\Nitish Mutha>pip install tensorflow-gpu
Collecting tensorflow-gpu
  Using cached tensorflow_gpu-0.12.1-cp35-cp35m-win_amd64.whl
Collecting six>=1.10.0 (from tensorflow-gpu)
  Using cached six-1.10.0-py2.py3-none-any.whl
Requirement already satisfied: wheel>=0.26 in c:\users\nitish mutha\anaconda3\envs\tensorflow-gpu\lib\site-packages (from tensorflow-gpu)
Collecting protobuf>=3.1.0 (from tensorflow-gpu)
  Using cached protobuf-3.1.0.post1-py2.py3-none-any.whl
Collecting numpy>=1.11.0 (from tensorflow-gpu)
  Using cached numpy-1.12.0-cp35-none-win_amd64.whl
Requirement already satisfied: setuptools in c:\users\nitish mutha\anaconda3\envs\tensorflow-gpu\lib\site-packages\setuptools-27.2.0-py3.5.egg (from protobuf>=3
.1.0->tensorflow-gpu)
Installing collected packages: six, protobuf, numpy, tensorflow-gpu
Successfully installed numpy-1.12.0 protobuf-3.1.0.post1 six-1.10.0 tensorflow-gpu-0.12.1

(tensorflow-gpu) C:\Users\Nitish Mutha>activate tensorflow-gpu

(tensorflow-gpu) C:\Users\Nitish Mutha>
```

Done. You have successfully installed TensorFlow with GPU on your Windows machine.

Now lets test if it is using GPU…

Activate environment we created

```
activate tensorflow-gpu
```

Test GPU

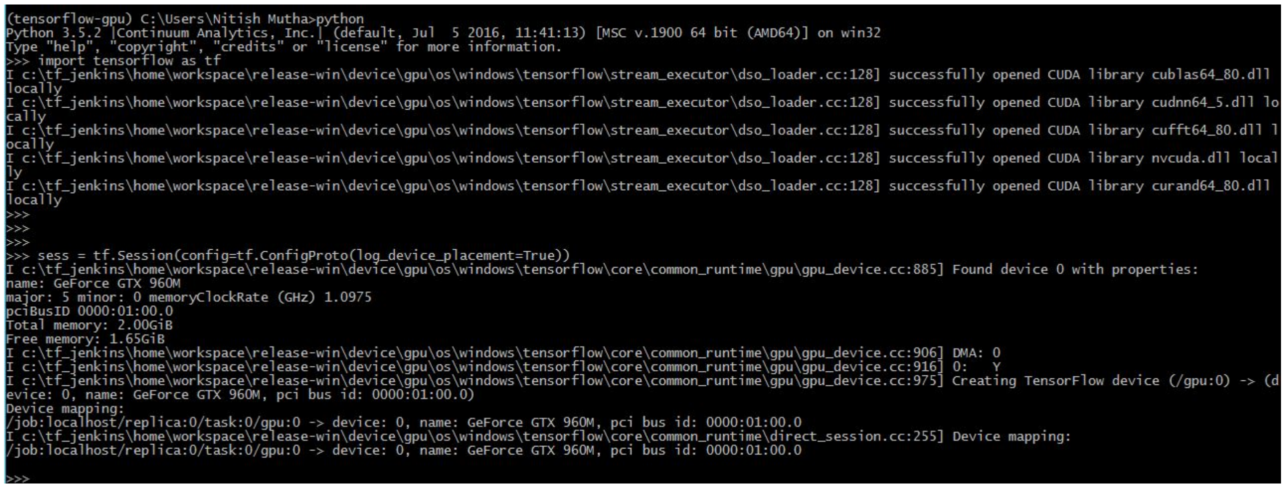
Enter into python shell

```
python
```

```
import tensorflow as tf
```

Now run this command and check if it identifies your GPU.

```
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
```



```
(tensorflow-gpu) C:\Users\Nitish Mutha>python
Python 3.5.2 |Continuum Analytics, Inc.| (default, Jul  5 2016, 11:41:13) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library cublas64_80.dll
locally
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library cudnn64_5.dll lo
cally
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library cufft64_80.dll l
ocally
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library nvcuda.dll local
ly
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\dso_loader.cc:128] successfully opened CUDA library curand64_80.dll
locally
>>>
>>>
>>>
>>> sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\core\common_runtime\gpu\gpu_device.cc:885] Found device 0 with properties:
name: GeForce GTX 960M
major: 5 minor: 0 memoryClockRate (GHz) 1.0975
pciBusID 0000:01:00.0
Total memory: 2.00GiB
Free memory: 1.65GiB
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\core\common_runtime\gpu\gpu_device.cc:906] DMA: 0
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\core\common_runtime\gpu\gpu_device.cc:916] 0:   Y
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\core\common_runtime\gpu\gpu_device.cc:975] Creating TensorFlow device (/gpu:0) -> (d
evice: 0, name: GeForce GTX 960M, pci bus id: 0000:01:00.0)
Device mapping:
/job:localhost/replica:0/task:0/gpu:0 -> device: 0, name: GeForce GTX 960M, pci bus id: 0000:01:00.0
I c:\tf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\core\common_runtime\direct_session.cc:255] Device mapping:
/job:localhost/replica:0/task:0/gpu:0 -> device: 0, name: GeForce GTX 960M, pci bus id: 0000:01:00.0

>>>
```

That's all. Time to play with it.