TensorFlow Installation Instructions:

What I have learned is that to use the full capability of TensorFlow including TensorBoard on PC's you will probably have to install TensorFlow on the GPU. Unfortunately, support for running TensorFlow on Mac's has been discontinued. However, you can still run TensorFlow on the CPU's on Mac's. To install TensorFlow on the GPU on PC's cleanly you should uninstall the version of TensorFlow that you currently have on the CPU. But, there isn't any good way of doing that. So…

What I've found is a "work around." This is basically to install a new virtual environment then install TensorFlow on the GPU in the new virtual environment. I'll do the Windows and Mac installations separately with the Windows installation first followed by the MAC OS installation. The last few steps are for installing TensorFlow on GPU's. The key to much of this is to pay attention to the versions of different packages/codes. Just installing the latest versions of different packages/codes will NOT work. Different versions of different things are needed to be compatible. I have tried to document this as clearly as possible.

Windows installation on the GPU

1. First, set up a new virtual environment. You'll also need to modify the path in your environment to this new virtual environment. If you have questions don't hesitate to ask!
2. Next, check and make sure that your GPU card has a CUDA compute capability of 3.0 or higher. For more details about what this is go to: https://developer.nvidia.com/cuda-gpus and scroll down to the frequently asked questions[1]. If you don't have this graphics capability TensorFlow on the GPU will not work.
3. Next, you must have the **CUDA Toolkit 8.0** installed. The NVIDIA links will take you to CUDA 9.XX. Do not install 9.XX. It is not compatible with the version of TensorFlow we need to take full advantage of TensorBoard. You must install **CUDA Toolkit 8.0**. CUDA 8.0 is at: https://developer.nvidia.com/cuda-80-ga2-download-archive. You can find both the original download and subsequent patches here.
4. You will need to get the required drivers. Again, you will need an older version that the current version the links send you to. That is, the links look like they send you to cuDNN v7 but you really want cuDNN v6.0 for CUDA 8.0. You have to have an NVIDIA Developer account to download cuDNN files. This account is free you just have to sign up. Once you

[1] I've checked and for my Windows machine I had to use the Device Manager in the Control Panel. Apparently I have an NVIDIA GeForce GTX 745, whatever that is. I haven't looked it up online to see but I bought this machine specifically for its graphics capability. For my MacBook I followed the instructions online. It looks like I have an Intel HD Graphics 6000, again whatever that is. Given time I'll look these up to see what they are. In the meantime it looks like I have plenty of graphics capability.

have signed up and logged in go to: https://developer.nvidia.com/rdp/cudnn-download
You will have to check a box that you agree to the terms of the software license agreement to get to the downloads area. I have tried to make this easier and attached files for Windows 7 and 10 as well as MAC OSx. If these don't fit your system you'll need to download the correct version.

To do this download, download the cuDNN 5.1 file (which is for CUDA 8.0) for the operating system version of your machine to get the:

- cudnn64_5.dll in the CUDA\v8.0\bin directory
- cudnn.h the C/C++ header file in the CUDA\v8.0\include directory
- curand64_80.dll cudnn file (object file library) in the CUDA\v8.0\lib\x64 directory

The instructions say to "install" these. That isn't quite correct. You need to extract or unzip the cuDNN file and move those 3 files to the ~\bin, ~\include and ~\lib\x64 directories. For whatever reason when I first tried to put the zip file where it would extract directly to these directories it said there was a permissions problem and wouldn't extract the files. I'm not sure why. But, I wound up extracting the files to my Desktop and just moving them to the correct directories.

You will also need to add a path to the environment / parent directory of the directories where these files are located. I think we've done this enough that you understand what to do. If not don't hesitate to ask.

5. Once you have all this graphics related stuff installed, next you need to install TensorFlow on the GPU. I actually made a couple new virtual environments and tried using Anaconda to install. But, I found I didn't need to. The pip3 install worked just fine once I had all the graphics stuff done.

   You can find the installation instructions using pip3 and Anaconda at: https://www.tensorflow.org/install/install_windows. The one thing it doesn't talk about is you have to have activated your new virtual environment before you begin installing TensorFlow. And, that's true for any other package you'll want to use with TensorFlow. Now I have a bunch of such packages.

   So, you need to go to the directory that you have the path set-up for to activate your new virtual environment. Activate your new virtual environment and begin the installation process. If you are using pip3 you should be in a Command Prompt Window. Then,

```
C:\> pip3 install --upgrade tensorflow-gpu
```

   Don't forget the –gpu part!

6. The process for validating your installation is the same for Windows or MAC's. So, I'll cover that after the next section for installation on a MAC.

MAC OS installation

1. TensorFlow's webpage for MAC OS installation shows 4 ways to install: virtualenv, "native" pip, Docker or installing from sources. Both the virtualenv and the Docker installation will isolate this version of TensorFlow from other things, like other Python versions, already installed on your computer.

   I have not used Docker before but there is a lot about using TensorFlow with Docker on the Internet. Docker documentation claims that the containers created by Docker "enable true independence between applications and infrastructure and developers and IT ops…" It appears that Docker was originally intended to support organizations operating in the "cloud". If you are familiar with "images" as opposed to copies, it looks like Docker works off an image. So, when a container is made for TensorFlow then an image of TensorFlow is launched in that container when you want to use it. There are a couple extra steps if you want to use Jupyter notebook and if you want to use TensorBoard, which we do.

   Docker Community Edition is free. Docker's various Enterprise editions are fairly expensive.

2. So, I'm going with installing via a new virtual environment. I've installed a new virtual environment named "venv". Once you've installed your virtual environment you use the following to start it:

   $ source activate venv

   You'll know the virtual environment is active because the name of the virtual environment appears in parentheses before the command prompt. In my case, (venv) appears.

   And, you also need to make sure to install all the requisite packages within that virtual environment: numpy, scipy, scikit-learn, matplotlib, and pandas. You may think of others.

3. You will also need to have Homebrew installed. This makes it much easier to install what's coming next. If you don't have Homebrew installed enter:

   ```
   /usr/bin/ruby -e "$(curl –fsSL
   https://raw.githubusercontent.com/Homebrew/install/master/install)"
   ```

   at the prompt in a terminal. Next, make sure that homebrew is up-to-date enter the following at the prompt:

```
(venv) $ brew update
(venv) $ brew upgrade
```

4. The next thing to do is to install some additional tools. At the prompt enter:

   ```
   (venv) $ brew install coreutils
   (venv) $ brew install swig
   (venv) $ brew install bazel
   ```

   Check to make sure that you've installed bazel 0.1.4 or greater. Versions lower than 0.1.4 will not allow TensorFlow to be installed. The current version of bazel that gets installed is 0.7.0 – which is fine. To check your version enter the following at the prompt:

   ```
   (venv) $ bazel version
   ```

5. You need something called "clang" that you can also install with brew as follows:

   ```
   (venv) $ brew install -- with-clang llvm
   ```

   When I used this command I got a message "Warning: llvm: this formula has no -- with-clang option so it will be ignored!" However, when I checked to see if clang was installed it was. I opened a separate terminal to do this. So the virtual env was not used for this. Use:

   ```
   ls /usr/local/opt/llvm/bin  | grep clang
   ```

   and I got:

   ```
   clang
   clang++
   clang-3.9
   clang-apply-replacements
   clang-check
   clang-cl
   clang-format
   clang-include-fixer
   clang-query
   clang-rename
   clang-tblgen
   clang-tidy
   git-clang-format
   ```

6. Next… Apparently TensorFlow looks for a C library called OpenMP. That doesn't exist in clang so we need to also install gcc49. Use:

   ```
   (venv) $ brew install gcc49
   ```

The next steps are back to installing TensorFlow on GPU's…

7. At last you are ready to install CUDA. For Macs you need to install CUDA 7.5. CUDA 7.5 is a 1GB file. I cannot put this on Moodle for you. So, I'm uploading the file to a Google drive that you can get it from. You'll get the link in a separate message.

   In addition to installing CUDA 7.5 you must set-up the path in your environment. There are two parts to this. Use both:

   ```
   export PATH=/Developer/NVIDIA/CUDA-7.5/bin:$PATH
   export DYLD_LIBRARY_PATH=/Developer/NVIDIA/CUDA=7.5/lib:$DYLD_LIBRARY_PATH
   ```

   Then, to verify your CUDA installation use:

   ```
   nvcc –V
   ```

   You should get something like:

   ```
   nvcc: NVIDIA (R) Cuda compiler driver
   Copyright (c) 2005-2015 NVIDIA Corporation
   Built on Mon_Apr_11_13:23:40_CDT_2016
   Cuda compilation tools, release 7.5, V7.5.26
   ```

8. You need the NVIDIA cuDNN GPU-accelerated library of primitives for deep neural networks, specifically cuDNN v5.1 for CUDA 7.5. I'll include a copy of that on the Google drive too. Once you have it you need to extract the lib files:
   - libcudnn_static_a
   - libcudnn.5.dylib
   - libcudnn.dylib

   to /usr/local/cuda/lib. And, the file cudnn.h in the include directory to /usr/local/cuda/include. It doesn't make much difference how you extract these files from the tar zip-file as long as they wind up in the proper /usr/local/cuda directories. You may have to change the "owner" on the directories before you can copy or move these files to where they belong. You can use the chown command in your terminal to do this. For example

   ```
   sudo chown username:group filename
   ```

   For me this is:

   ```
   sudo chown marvinehamner:wheel lib
   ```

or for a directory:
sudo chown –R $(whoami) .

9. Almost done! Now you need to configure CUDA and TensorFlow on the GPU. You already have TensorFlow installed. It just isn't configured to run on your GPU yet.

   You need to do a couple things to **get the CUDA configuration process started**. First, in your terminal go to the directory where tensorflow is installed. Then enter:

   git clone –recurse-submodules https://github.com/tensorflow/tensorflow
   cd tensorflow
   git fetch origin pull/664/head:cuda_osx
   git checkout cuda_osx

   When you clone the tensorflow repository from github it will create another directory "tensorflow". So, you do need to change directories, i.e. "cd tensorflow" again or you'll get an error "Not a git repository…" The last command actually switches you to cuda_osx.

   Now you can do the configuration by entering:

   TF_UNOFFICIAL_SETTING=1 ./configure

   To set-up the configuration use:

   WARNING: You are configuring unofficial settings in TensorFlow. Because some external libraries are not backward compatible, these settings are largely untested and unsupported.

   Please specify the location of python. [Default is /usr/bin/python]: if you path to Python is different enter it here
   Do you wish to build TensorFlow with GPU support? [y/N] y
   GPU support will be enabled for TensorFlow
   Please specify the Cuda SDK version you want to use. [Default is 7.0]: 7.5
   Please specify the location where CUDA 7.5 toolkit is installed. Refer to README.md for more details. [Default is /usr/local/cuda]: if your path to the CUDA 7.5 toolkit is different enter it here
   Please specify the Cudnn version you want to use. [Default is 6.5]: 5
   Please specify the location where cuDNN 6.5 library is installed. Refer to README.md for more details. [Default is /usr/local/cuda]: this should be the default but again, if your path is different enter it here
   Please specify a list of comma-separated Cuda compute capabilities you want to build with. You can find the compute capability of your device at: https://developer.nvidia.com/cuda-gpus.

```
Please note that each additional compute capability significantly increases your build time and binary size.
[Default is: "3.5,5.2"]: 3.0
Setting up Cuda include
Setting up Cuda lib
Setting up Cuda bin
Setting up Cuda nvvm
Configuration finished
```

Now for the **TensorFlow configuration**, enter:

```
git checkout r1.0
./configure
```

And, this is where the fun really begins… I have had to copy the files to the ~/lib and ~/include directories in several places. I have had to change ownership for several directories where things were installed by “root”. It isn’t impossible it is just time consuming to get through the error messages and finish this configuration. But, in the end I got the message:

“Configuration finished”