Quick Tutorial to Install Necessary Deep Learning Packages

Created by Arka Bhowmik (2021)

Content

1. Setting-up tensorflow-gpu and necessary packages and libraries in secured server --- 1-2

2. Setting-up tensorflow and necessary packages and libraries in local computer --- 3-5

Setting-up a virtual environment tensorflow-gpu and jupyter lab for deep learning project

1. Go to directory with more space

```
[bhowmika@lilac-ln02 ~]$ cd /data/Arka/

[bhowmika@lilac-ln02 Arka]$ mkdir myenv

[bhowmika@lilac-ln02 Arka]$ cd myenv
```

2. Create your own **virtual environment** to install packages in it (path → /data/Arka/myenv)

```
[bhowmika@lilac-ln02 myenv]$ module load conda/conda3            # load a conda package

[bhowmika@lilac-ln02 myenv]$ conda create -p /data/Arka/myenv/tf_gpu tensorflow-gpu
```

3. After creating a virtual environment, activate the virtual environment (path → /data/Arka/myenv)

```
[bhowmika@lilac-ln02 Arka]$ module unload conda/conda3
```

```
[bhowmika@lilac-ln02 Arka]$ source
                    /opt/common/CentOS_7/anaconda/anaconda3/etc/profile.d/
                    conda.sh       # activates conda

[bhowmika@lilac-ln02 Arka]$ conda env list    # shows the list of environment available

[bhowmika@lilac-ln02 Arka]$ conda activate /data/Arka/myenv/tf_gpu

(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$    # you are now inside your virtual environment
```

4. You can install all necessary packages such as matplotlib, keras, pandas, plotly, scikit-learn, etc. It is important to note that many packages have some dependencies which may or may not work because of version used. For example, keras tuner is used for hyperparameter optimization but requires tensorflow 2.0 or above and have dependencies on particular versions of matplot lib, etc. Hence, skip updating packages that might create problem with keras tuner since it comes with bunch of packages.

```
(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$ pip install --upgrade pip

(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$ pip3 install keras-tuner --upgrade

(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$ pip3 install beautifulsoup4 pandas
                                                        plotly scikit-learn seaborn
                                                        statsmodels opencv-python
                                                        Jupyterlab

(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$ pip3 install imblearn tensorflow-
                                                        addons

(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$ pip3 install pandas
                                                        profiling[notebook]

(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$ jupyter nbextension enable --
                                                        py widgetsnbextension
```

5. You can create a default kernel for tf_gpu (only do first time)

```
(/data/Arka/myenv/tf-gpu) [bhowmika@lilac-ln02 Arka]$ pip3 install ipykernel

                    (You may skip installing ipykernel if already installed)

(/data/Arka/myenv/tf-gpu) [bhowmika@lilac-ln02 Arka]$ python -m ipykernel install --user --
name tf_gpu --display-name tf-gpu
```

6. Open jupyter notebook/jupyter-lab for coding

```
(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$ jupyter notebook --no-browser --
                                                      port=2828 --ip=0.0.0.0

                                  OR

(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$ jupyter-lab --no-browser --port=2828
                                                      --ip=0.0.0.0
```

- Use the link to access the folders and files in the lilac server

Note: *The port=2828 need to be the one which you have set in tunnel option in connection→SSH→Tunnels of putty. Also, always close or shut down the kernel to use the same port next time. Otherwise, you need to set new port in the Tunnel of putty*.

7. You can switch virtual environment by deactivating the current environment (for example tf_gpu)

```
(/data/Arka/myenv/tf_gpu) [bhowmika@lilac-ln02 Arka]$ conda deactivate
```

Any necessary packages can be separately installed later using pip install command. Please check the package information in the world wide web.

Setting-up tensorflow gpu, virtual environment, jupyter notebook and other packages for windows 10

1. Initially visit the tensor flow webpage to determine the package software requirements

https://www.tensorflow.org/install/source#gpu

➔ Let us focus on the tensorflow version 2.4.0 (steps tested). Please note the requirements for tensorflow 2.4.0, we need python version 3.6, cuDNN 8.0, CUDA 11.0

2. At the beginning, install visual studio by visiting visual studio webpage

https://visualstudio.microsoft.com/vs/community/ (*Download visual studio*)

➔ Install the visual studio installer (*proceed as instructed*). When prompted by installer which packages to install (*do not select any check boxes*) only install the visual studio core editor (*default*).

3. Next, install CUDA toolkit 11.0 (visit NIVIDIA CUDA webpage)

https://developer.nvidia.com/cuda-toolkit-archive (*Download CUDA Toolkit 11.0 Update1 (Aug 2020)*)

➔ Download and Install windows installer (*only use version 10 not server version and installer type: .exe(network)*).
➔ When prompted for installer installation option (*Select recommended express setting*).
➔ Next, finish the installation as instructed (*uncheck all in final step like desktop shortcut and all*).

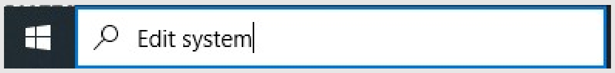
4. Next, download cuDNN version 8.0.5 (Deep Neural Network library)

https://developer.nvidia.com/cudnn (*To download cuDNN 8.0.5 one has to create a login follow the steps*)

➔ Login to the library download page and accept the terms.
➔ After accepting the terms download cuDNN 8.0.5 [November 9th 2020] compatibility with CUDA 11.0 (*Only download windows version of compatible library*).
➔ Next, unpack the downloaded zip folder to windows folder (*copy the unzipped contents (a) bin folder, (b) include folder, (c) lib folder and paste in C Drive → Program Files → Goto folder NVIDIA GPU Toolkit folder → CUDA folder → v11.0 folder → Paste copied files and replace the existing ones*).

5. Next, set system environment to use the library in your windows PC.

➔ Next, goto bin folder and libnvvp folder copy the path (*1:- C:\Program Files\....\v11.0\bin and 2: C:\Program Files\....\v11.0\libnvvp*).
➔ Next, search and open "Edit system environment variable". Search in windows.



Edit system

(next *page …*)

Contact: arkabhowmik@yahoo.co.uk

- ➔ Next, system property window will open → then open environment variables tab → Under user variables add a new path in variable path {by select and edit path and press new} to paste previously copied paths for bin and libnvpp separately. This will set the system environment paths.
- ➔ After completing all the above steps (*restart the computer*).

6. Create your own virtual environment

- ➔ Open *anaconda prompt* from search in search bar in windows and type the below commands
- ➔ (base) C:\Users\Arka> `conda create --name tf_gpu python==3.8` (*creates an environment, needs to be executed once*)
- ➔ (base) C:\Users\Arka> `conda env list` *# shows the list of environment available*
- ➔ (base) C:\Users\Arka> `conda activate tf_gpu` (*activates the environment, needs to be executed PC restarts*)
- ➔ (tf_gpu) C:\Users\Arka> `pip install tensorflow` (*installs both the CPU and GPU versions*)
- ➔ Notice the (base) is changed to (tf_gpu) after activate this will ensure all the changes will be made only in the environment tf_gpu. You can make multiple virtual environments for exploring different version of tensorflow.
- ➔ Now, you are all set.

Note: If the computer is not GPU compatible only CPU version will be installed. In case, failed to install GPU version though the computer has GPU capability than create a new environment as shown below.

- ➔ (base) C:\Users\Arka> `conda create --name tf_gpu2 python==3.8 tensorflow-gpu` (if earlier method fails *creates another environment gpu2, needs to be executed once*). Follow the same step to activate the tf_gpu2 as earlier.

7. You can install all necessary packages such as matplotlib, keras, pandas, plotly, scikit-learn, etc.

```
(tf_gpu) C:\Users\Arka>  pip install --upgrade pip

(tf_gpu) C:\Users\Arka>  pip3 install keras-tuner –upgrade

(tf_gpu) C:\Users\Arka>  pip3 install beautifulsoup4 pandas plotly scikit-learn seaborn
                         statsmodels opencv-python Jupyterlab imblearn tensorflow-addons

(tf_gpu) C:\Users\Arka>  pip3 install pandas-profiling [notebook]

(tf_gpu) C:\Users\Arka>  jupyter nbextension enable --py widgetsnbextension
```

8. You can create a default kernel for jupyter notebook tf_gpu (only do first time)

```
(tf_gpu) C:\Users\Arka>  pip3 install ipykernel

(tf_gpu) C:\Users\Arka> python -m ipykernel install --user --name tf_gpu --display-name tf_gpu
```

9. Open jupyter notebook/jupyter-lab for using prompt (instead of anaconda navigator)

```
(tf_gpu) C:\Users\Arka> jupyter notebook

                        OR

(tf_gpu) C:\Users\Arka> jupyter-lab
```

10. You can switch virtual environment by deactivating the current environment (for example tf_gpu)

```
(tf_gpu) C:\Users\Arka> conda deactivate

(base) C:\Users\Arka> conda activate tf_gpu2
```