Installing Tensorflow on Linux

June 28, 2017

This tutorial introduces how to install TensorFlow on Linux. Here, we installed it on Ubuntu 16.04 for an example. Also, you can refer to the TensorFlow official installation tutorial.

After installing, we take linear regression to explain how to run a TensorFlow program by Python.

1 Installation

1.1 Determine which TensorFlow to install

There are two types of TensorFlow you can install: CPU support version and GPU support version.

If your system has a NVIDIA® GPU, you can install the GPU support version. TensorFlow programs typically run significantly faster on a GPU than on a CPU. However, the installation progress of GPU version is more complicated than CPU version, cause you need to install CUDA and cuDNN firstly.

On the other hand, if your system doesn't have a NVIDIA® GPU or you want to simplify the installation progress even if has a NVIDIA® GPU, you can install CPU support version.

1.2 Installing CUDA and cuDNN

If you want to install the GPU version TensorFlow, you need to install the CUDA and cuDNN firstly. If you just want to install CPU version, please skip this section to Installing TensorFlow.

1.2.1 Installing CUDA

At first, let we install the CUDA first. Go to the CUDA download page, and download the CUDA file you need, as the Fig.1 shown. Here, we download the CUDA® Toolkit 8.0 for Ubuntu 16.04.

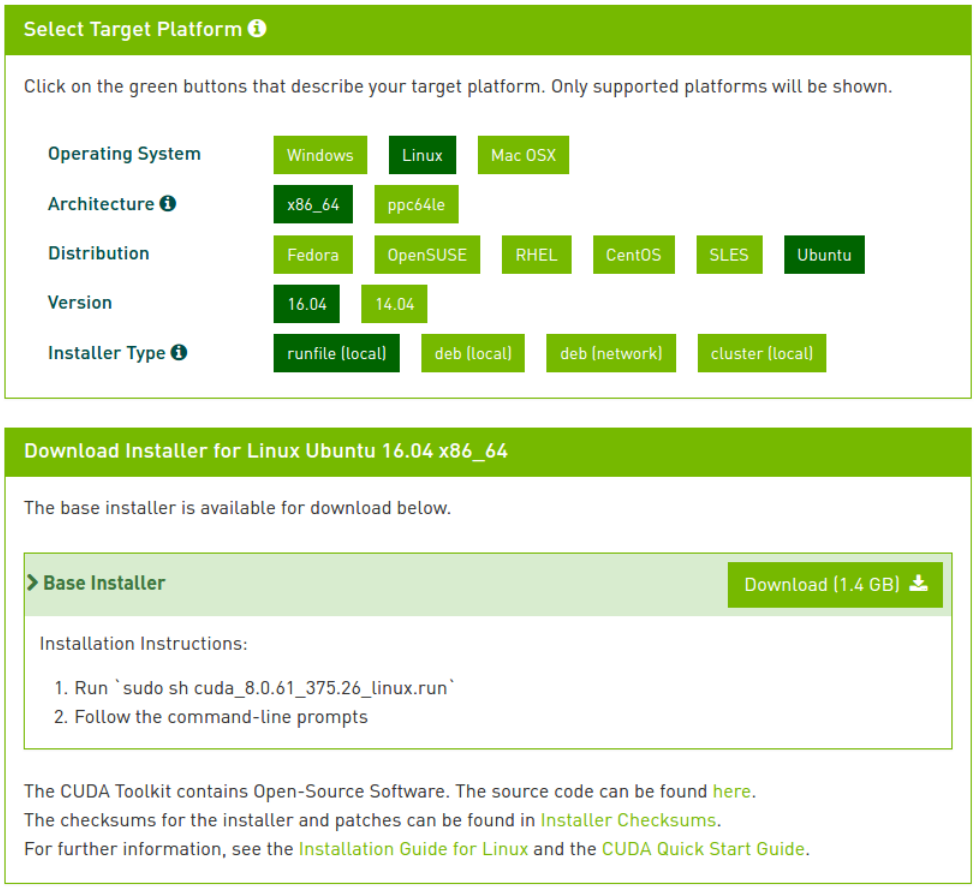

Figure 1: CUDA download page

When we finish it, now let's install it. Change to the directories you saved the CUDA file using terminal, and start the installation. The command is shown as below and you can see the terminal as Fig.2.

```
$ cd Desktop/
$ sudo sh cuda_8.0.61_375.26_linux.run
```

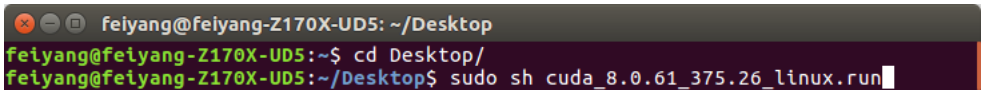

Figure 2: Start CUDA installation

Then, it will go to "End User License Agreement", and you can press "CTRL + C" skip this part. And then, you need to choose some options about installation, which is shown as Fig.3.

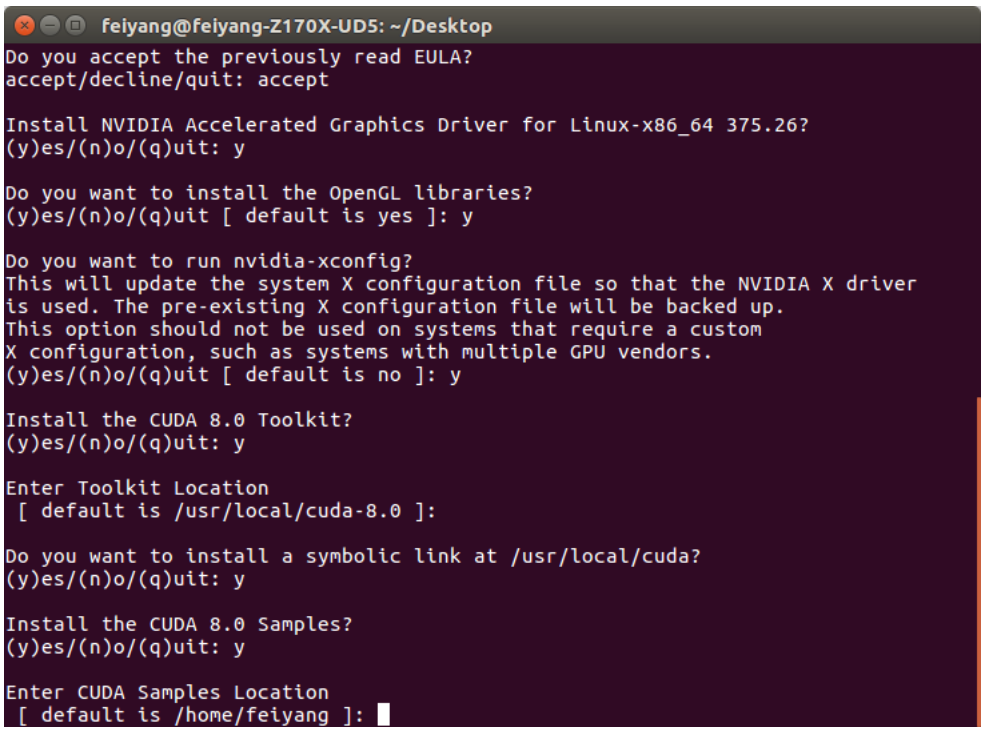

Figure 3: CUDA installation options

Waiting some time till it finishes, and rebooting your system, the CUDA has already been installed.

1.2.2 Installing cuDNN

Then, we can install cuDNN. Go to the cuDNN download page,and download the cuDNN file you need, as the Fig.4 shown. Here, we download the cuDNN v6.0 for CUDA 8.0 and Linux.

When it finishes, uncompress cuDNN and copy the files to the directory of CUDA you installed before by command below,

```
$ cd Desktop/
$ tar xvzf cudnn-8.0-linux-x64-v6.0.tgz
$ sudo cp cuda/include/cudnn.h /usr/local/cuda/include
$ sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64
$ sudo chmod a+r /usr/local/cuda/include/cudnn.h \
        /usr/local/cuda/lib64/libcudnn*
```

Till now, we finish installing the cuDNN. Now we can start to install TensorFlow.

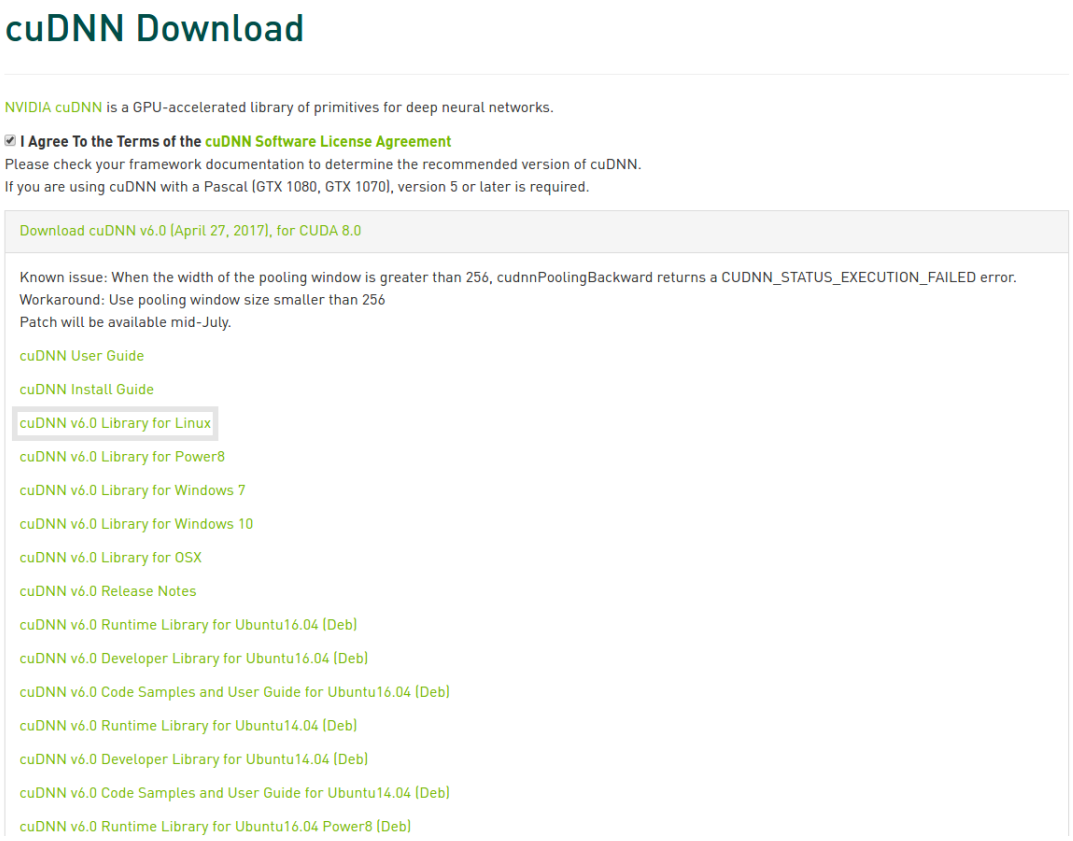

Figure 4: cuDNN download page

1.3 Installing TensorFlow

On the TensorFlow official installation tutorial, it introduces 4 ways to install TensorFlow. Here, we suggest you use the Anaconda, which will make the programing easier. For other ways, please refer TensorFlow official installation tutorial.

1.3.1 Installing Anaconda

At first, we need to install the Anaconda. Go to the Anaconda download page, and you need to choose which version of Python you will use. Here, we choose 64-BIT(X86) INSTALLER.

Then, install Anaconda with the command below.

For Python 3.6 version,

```
$ cd Desktop/
$ bash Anaconda3-4.4.0-Linux-x86_64.sh
```

For Python 2.7 version,

```
$ cd Desktop/
$ bash Anaconda2-4.4.0-Linux-x86_64.sh
```

It is easily to install it, we don't want to talk it any more.

1.3.2 Installing TensorFlow with Anaconda

Now we can start install TensorFlow eventually with Anaconda.

First, open the terminal and create a conda environment named **tensorflow** by invoking the following command:

```
$ conda create -n tensorflow
```

Then activate the conda environment created above by:

```
$ source activate tensorflow
```

At last, install the TensorFlow inside your conda environment:

```
$ pip install --ignore-installed --upgrade TF_PYTHON_URL
```

where **TF_PYTHON_URL** is the URL of the TensorFlow Python package. You should take care the Python version and the CPU or GPU version. For example, the following command installs the GPU version of TensorFlow for Python 3.6:

```
$ pip install --ignore-installed --upgrade \
        https://storage.googleapis.com/tensorflow/linux/gpu\
        /tensorflow_gpu-1.1.0-cp36-cp36m-linux_x86_64.whl
```

1.4 Validating your installation

At last, we need to run a short TensorFlow program to validate your installation.

Invoke Python from your shell as follows:

```
$ python
```

Enter the following short program inside the Python interactive shell:

```
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
```

If the systems outputs the following, then you are ready to begin writting TensorFlow programs:

```
Hello, TensorFlow!
```

2 Linear Regression

Now, we can start our first TensorFlow program: Linear Regression. I believe this is a good start for your TensorFlow study.

There are two parts in this section. First, we need to install an IDE for programming. Here, the PyCharm is introduced as a powerful IDE for Python, and we will explain how to install and deploy it. Of course, you can choose other IDEs, and you can skip to the Linear Regression Program.

2.1 Installing PyCharm

Pycharm is a powerful IDE for Python programming, but it is a little heavy. You can download the community version or apply student-free professional version.

When you finishing it, you can install the Pycharm by invoking the following command:

```
$ cd Desktop/
$ tar -xzf pycharm-professional-2017.1.4.tar.gz
$ mv pycharm-2017.1.4/ ~/
$ cd ~/pycharm-2017.1.4/bin/
$ sh pycharm.sh
```

Now, open the Pycharm and configure the setting. Click "File" and "Default Settings" . Then click "Project Interpreter", choose the Python interpreter in "tensorflow" environment, as the Fig.5 shown.

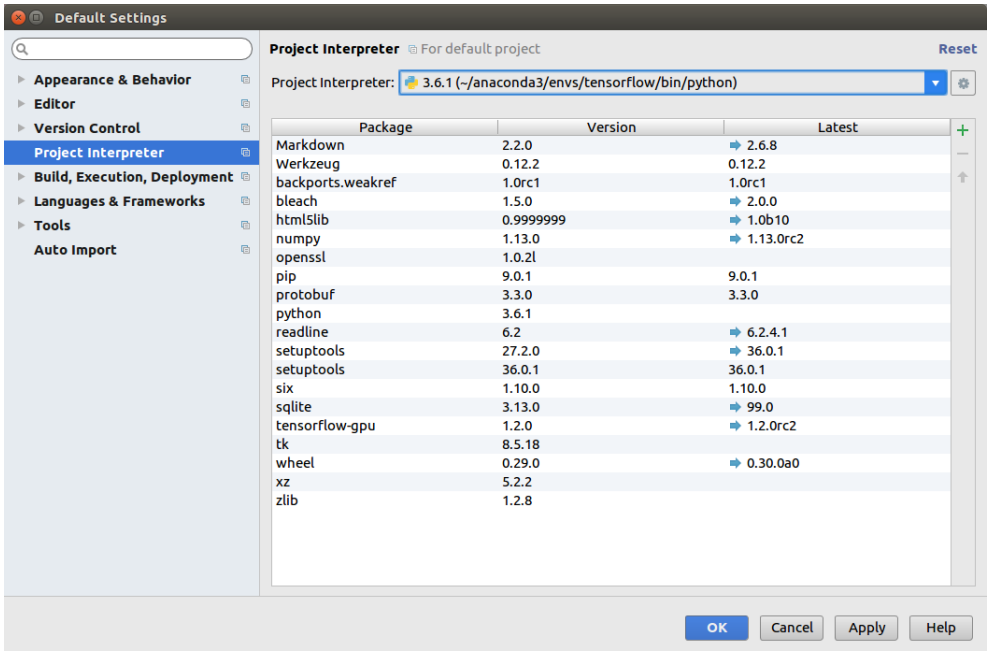

Figure 5: Pycharm interpreter settings

Now, the settings for Pycharm are all done.

2.2 Linear Regression Program

Now we can start our first TensorFlow program, Linear Regression. Our task is find a best line to fit all the points in the X-Y plane.

At first, we need to import the "tensorflow" module to the Python, and also "numpy" for scientific computing, "matplotlib" for 2D figure plotting. The code is shown below:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

Then generating a sets of data as the training set by "numpy", which is 2 dimensions. "x_data" is 100 evenly spaced numbers between 0 to 1. "y_data" has the linear relationship with "x_data". In order to make it a little hard to learn, we add some noise to the "y_data". The code is shown below:

```
x_data = np.float32(np.linspace(0, 1, 100))
x_data = x_data.reshape(1, 100)
y_data = 0.5 * x_data + 0.2 + np.random.normal(0, 0.01, 100)
```

Then we need to create a linear model using the TensorFlow, "W" and "b" stand for the wight and the bias of the linear equation. The initial value of bias is 0 and the weight is a random variable from uniform distribution. "y" is the out in our model. The code is shown below:

```
b = tf.Variable(tf.zeros([1]))
W = tf.Variable(tf.random_uniform([1,1], -1.0, 1.0))
y = tf.matmul(W, x_data) + b
```

In our task, we need to train the model we created here to fit the training set. In other words, we need to make the difference between "y_data" — the training target and "y" — the model output small. Here, the "loss" is the variance to measure the difference.

The gradient descent algorithm is used as the optimizer, and the learning rate is 0.5. The code is shown as below.

```
loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)
```

Now, we can start train our training. At first, we need to initial our variables in model, and iterate the model 201 times. Every 20 iterations the program will print the model parameters. The code is shown below:

```
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for step in range(0, 201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))
```

Actually, we have already get the training result from the last iteration. In order to see the it more intuitively, let's draw the result in the figure, which is shown in Fig.6. Also, the code is attached below:

```
y_estimated = sess.run(W) * x_data + sess.run(b)
x_data = x_data.reshape(100)
y_data = y_data.reshape(100)
y_estimated = y_estimated.reshape(100)
plt.axis([0,1,0,0.8])
plt.plot(x_data,y_data,'b.')
plt.plot(x_data, y_estimated,'r-')
plt.show()
```

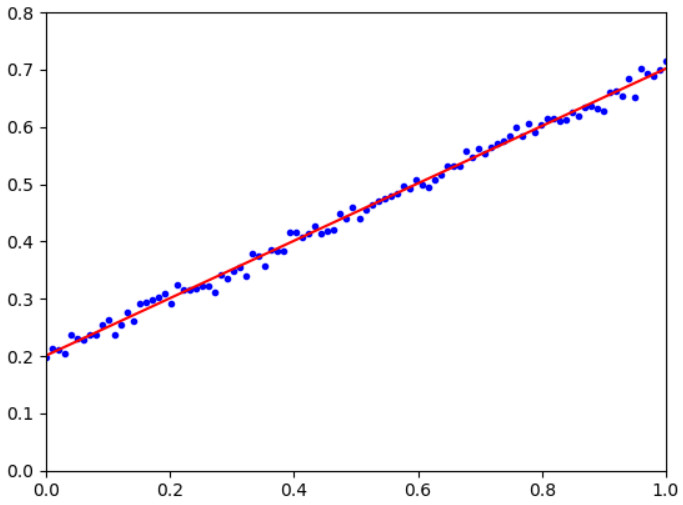

Figure 6: Linear Regression Result

Till now, we finish the our first TensorFlow program — Linear Regression.