4.Install TensorFlow GPU

Before we install TensorFlow GPU, we need to complete configuration of CUDA. About configuration of CUDA, please refer to 【1.Preparation tutorial】

1. Install pip

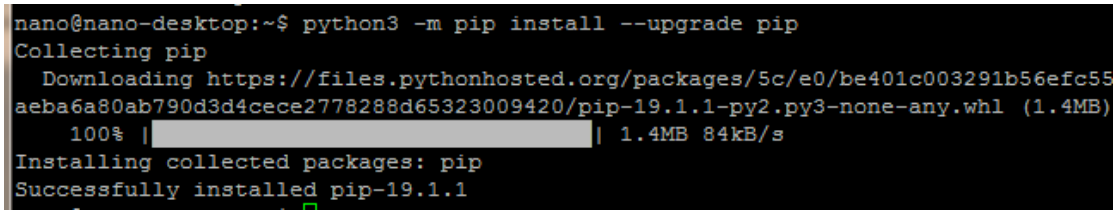
We need to input command:

sudo apt-get install python3-pip python3-dev

After installation, pip is version 9.01, you need to upgrade it to the latest version. After upgrading, the pip version is 19.1.1. There will be a small bug after the upgrade, you need to manually change it.

We need to input command:

python3 -m pip install --upgrade pip #upgrade pip



sudo vim /usr/bin/pip3 #Open pip3 file

Replace

```
from pip import main
if __name__ == '__main__':
    sys.exit(main())
```

to

```
from pip import __main__
if __name__ == '__main__':
    sys.exit(__main__._main())
```

After modification is complete, we need to input command:

pip3 -V

2.We need to input command to install some software package:

sudo apt-get install python3-numpy

(It is an extension library of Python language, which supports a large number of dimensional arrays and matrix operations, and also provides a large number of mathematical function libraries for array operations.)

sudo apt-get install python3-scipy

(Scipy is a common software package used in the fields of mathematics, science, and engineering, which can handle interpolation, integration,

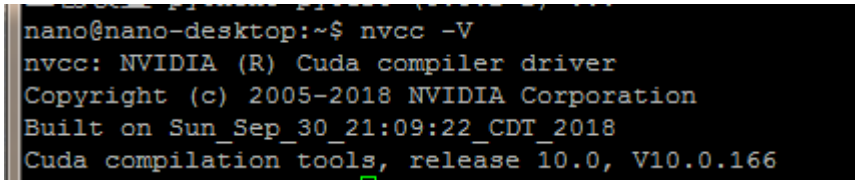
optimization, image processing, numerical solution of ordinary differential equations, signal processing, etc.)

sudo apt-get install python3-pandas
(Pandas is a tool based on NumPy, which is created to solve data analysis tasks. Pandas includes a large number of libraries and some standard data models, and provides the tools needed to efficiently operate large data sets. Pandas provides A large number of functions and methods that enable us to process data quickly and easily.)

sudo apt-get install python3-matplotlib
(Matplotlib is a 2D plotting library for Python that generates publishing-quality graphics in a variety of hardcopy formats and a cross-platform interactive environment)

sudo apt-get install python3-sklearn
(Simple and efficient data mining and data analysis tools)

3.Install TensorFlow GPU version
1)Check if CUDA is installed properly
We need to input command:
nvcc -V
If you can see the CUDA version number, as shown below, it is installed correctly.



2)Install the required package
We need to input command:
sudo apt-get install python3-pip libhdf5-serial-dev hdf5-tools

3)Install TensorFlow GPU version
We need to input the following command:
pip3 install --extra-index-url
https://developer.download.nvidia.com/compute/redist/jp/v42
tensorflow-gpu==1.13.1+nv19.3 --user

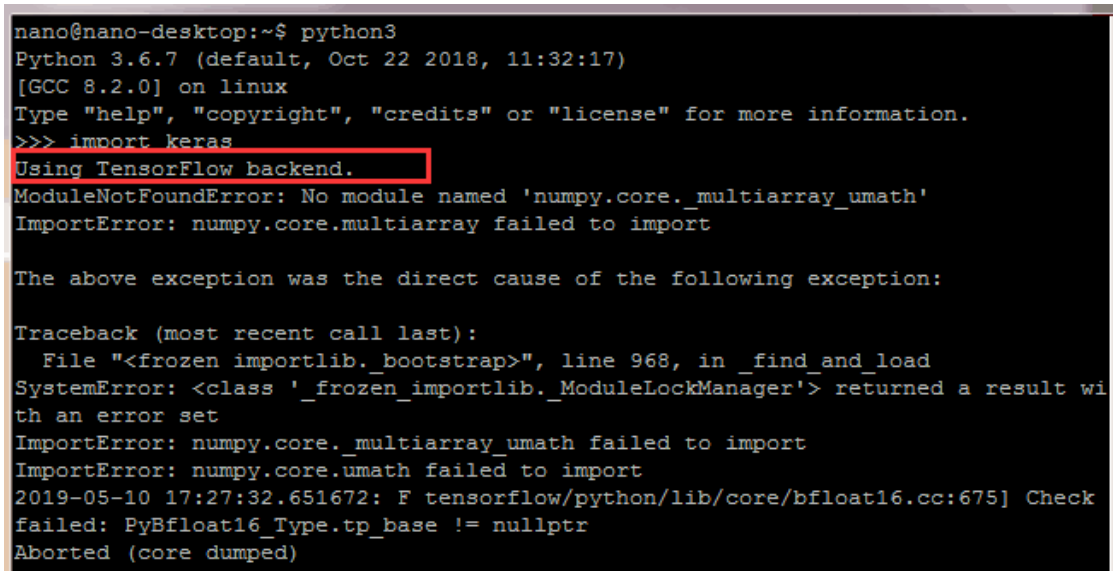
4. Install Keras
Keras is a high-level neural network API written in Python. It can run with TensorFlow, CNTK, or Theano as the backend.

We can input the following command to install Keras:
sudo pip3 install keras

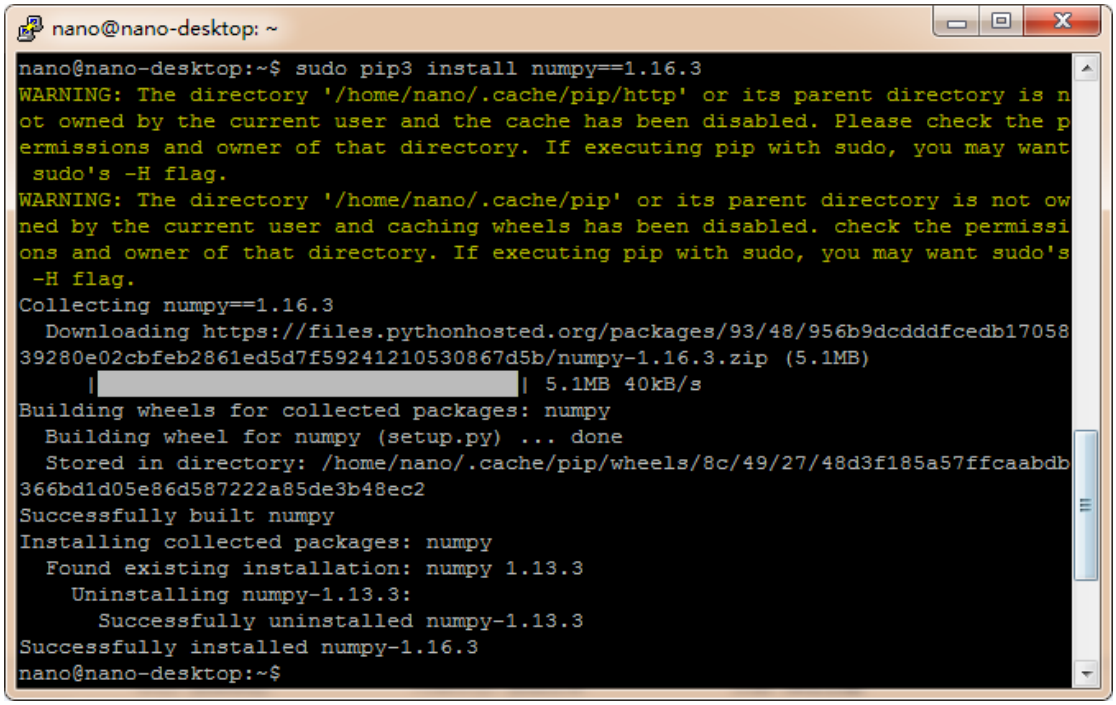
After the installation is complete, we can input **python3** and check the installation results. When we input **import keras**, the following prompts “using TensorFlow backend”, it proves that Keras is installed successfully and uses TensorFlow as backend.

```
beckhans@Jetson:~$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
Using TensorFlow backend.
>>>
```

If there is a problem like the one shown below, you need to update the version of numpy.



We can input the following command:
sudo pip3 install numpy==1.16.3



5. Test TensorFlow

We can use vi to create a new python file name: tensorflowDemo.py and then copy the following code into it. After saving, run it with **python3 tensorflowDemo.py**. This section must be run in a graphical interface, because a chart will appear.

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

x_data = np.linspace(-0.5, 0.5, 200)[:, np.newaxis]
noise = np.random.normal(0, 0.02, x_data.shape)
y_data = np.square(x_data) + noise

x = tf.placeholder(tf.float32, [None, 1])
y = tf.placeholder(tf.float32, [None, 1])

# Input layer one neuron, output layer one neuron, middle 10 neurons
# First layer
Weights_L1 = tf.Variable(tf.random.normal([1, 10]))
Biases_L1 = tf.Variable(tf.zeros([1, 10]))
Wx_plus_b_L1 = tf.matmul(x, Weights_L1) + Biases_L1
L1 = tf.nn.tanh(Wx_plus_b_L1)

# Second layer
```

```
Weights_L2 = tf.Variable(tf.random.normal([10, 1]))
Biases_L2 = tf.Variable(tf.zeros([1, 1]))
Wx_plus_b_L2 = tf.matmul(L1, Weights_L2) + Biases_L2
pred = tf.nn.tanh(Wx_plus_b_L2)

# Loss function
loss = tf.reduce_mean(tf.square(y - pred))

# Train
train = tf.train.GradientDescentOptimizer(0.1).minimize(loss)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(2000):
        sess.run(train, feed_dict={x: x_data, y: y_data})
        print("第{0}次，loss = {1}".format(i, sess.run(loss,feed_dict={x: x_data,
y: y_data})))
    pred_vaule = sess.run(pred, feed_dict={x: x_data})
    plt.figure()
    plt.scatter(x_data, y_data)
    plt.plot(x_data, pred_vaule, 'r-', lw=5)
    plt.show()
```