4、Jetson Xavier NX 安装 TensorFlow GPU 教程

今天的目标是安装 TensorFlow GPU 版本，安装 TensorFlow GPU 版本需要成功配置好 CUDA，没有配制好的请看教程 1。不过在安装 TensorFlow GPU 之前，有一些机器学习必须用到的安装包也需要来安装一下。

1. 安装 pip

因为 Jetson Xavier NX 中已经安装了 Python3.6 版本，所以安装 pip 还是比较简单的

sudo apt-get install python3-pip python3-dev

安装后 pip 是 9.01 版本，需要把它升级到最新版，升级后 pip 版本为 20.1。这里面升级后会有一个小 Bug，需要手动改一下

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

```
yahboom@yahboom-desktop:~$ python3 -m pip install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/00/b6/9cfa56b4081ad13874b0
c6f96af8ce16cfbc1cb06bedf8e9164ce5551ec1/pip-19.3.1-py2.py3-none-any.whl (1.4MB)
    100% |████████████████████████████████| 1.4MB 173kB/s
Installing collected packages: pip
Successfully installed pip-19.3.1
yahboom@yahboom-desktop:~$ pip -V
-bash: pip: command not found
yahboom@yahboom-desktop:~$
```

sudo vim /usr/bin/pip3　　#打开 pip3 文件
将原来的

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

改成

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

修改结束后保存。运行 pip3 -V 成功后显示

pip3 -V

```
jetbot@jetbot-desktop:~$ pip3 -V
pip 20.1 from /home/jetbot/.local/lib/python3.6/site-packages/pip (python 3.6)
jetbot@jetbot-desktop:~$
```

2. 安装那些机器学习领域非常重要的包

sudo apt-get install python3-numpy
(是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。)

sudo apt-get install python3-scipy
(Scipy 是一个用于数学、科学、工程领域的常用软件包，可以处理插值、积分、优化、图像处理、常微分方程数值解的求解、信号处理等问题。)

sudo apt-get install python3-pandas
(pandas 是基于 NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。pandas 提供了大量能使我们快速便捷地处理数据的函数和方法。你很快就会发现，它是使 Python 成为强大而高效的数据分析环境的重要因素之一。)

sudo apt-get install python3-matplotlib
(Matplotlib 是一个 Python 的 2D 绘图库，它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形)

sudo apt-get install python3-sklearn
(简单高效的数据挖掘和数据分析工具)

3. 安装 TensorFlow GPU 版

（1）确认 CUDA 已经被正常安装

nvcc -V
如果能看到 CUDA 版本号，即为正确安装

```
nx@nx-desktop:~$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Wed_Oct_23_21:14:42_PDT_2019
Cuda compilation tools, release 10.2, V10.2.89
```

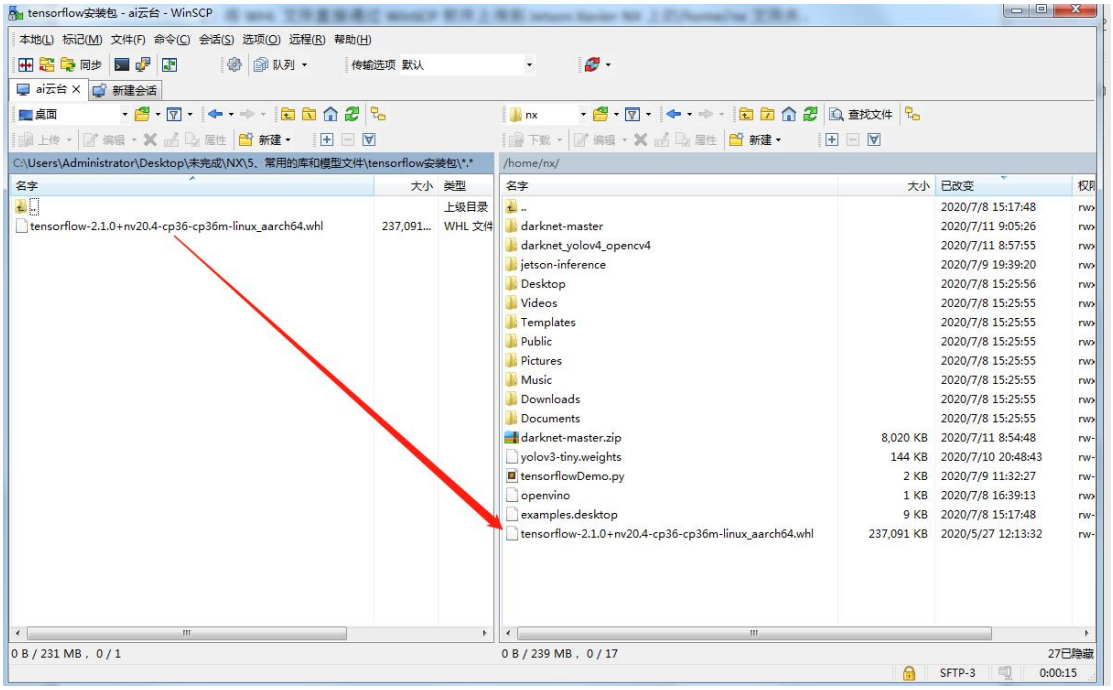
（2）安装所需要的包
sudo apt-get install libhdf5-serial-dev hdf5-tools libhdf5-dev zlib1g-dev zip libjpeg8-dev liblapack-dev libblas-dev gfortran

（3）安装 TensorFlow GPU 版本 (建议用离线方式)
（3.1）在线安装 TensorFlow GPU 版本
sudo pip3 install --pre --extra-index-url https://developer.download.nvidia.com/compute/redist/jp/v44 tensorflow

（3.2）离线安装 TensorFlow GPU 版本

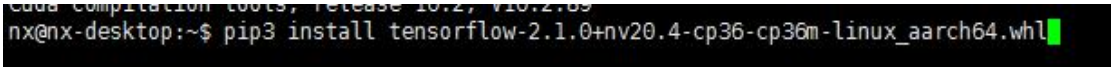
因为在线下载太慢了，所以我们可以选择离线包安装,tensorflow 安装包放在资料的常用的库和模型文件夹中。

1）将 WHL 文件直接通过 WinSCP 软件上传到 Jetson Xavier NX 上的/home/nx 文件夹。（WHL 文件在资料...\5、常用的库和模型文件\tensorflow 安装包 目录下。）



2）上传完成之后，输入指令

pip3 install tensorflow-2.1.0+nv20.4-cp36-cp36m-linux_aarch64.whl



下载途中可能也会需要在线安装一些软件包 直接 Y(YES)通过。

3）完成安装

4. 安装 Keras

Keras 是一个用 Python 编写的高级神经网络 API，它能够以 TensorFlow, CNTK, 或者 Theano 作为后端运行。Keras 的开发重点是支持快速的实验。能够以最小的时延把你的想法转换为实验结果，是做好研究的关键。

中文学习网站：https://keras.io/zh/

既然有了 TensorFlow，那就把 Keras 也安装上。我自己很喜欢 keras，让 TensorFlow 变得更加简单

sudo pip3 install keras

5. 测试 TensorFlow

跑一段自己写的非线性回归代码，速度还是挺快的，使用 vi 新建 python 文件命名：tensorflowDemo.py 然后复制以下代码进去。保存后使用 python3 tensorflowDemo.py 运行，这段必须在图形化界面下运行，因为会出现一张图表。

```
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])

# 输入层一个神经元，输出层一个神经元，中间 10 个
# 第一层
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)

# 第二层
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)

# 损失函数
loss = tf.reduce_mean(tf.square(y - pred))

# 训练
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()
```