****Windows下tensorflow的安装及使用****

整理：袁沅祥 2018年4月

# **1、安装Anaconda**

Anaconda是由Python提供支持的领先的开放数据科学平台。Anaconda的开源版本是Python和R的高性能分发版本，包括超过100个用于数据科学的最流行的Python，R和Scala软件包。【来自[Anaconda官方下载页面](https://www.continuum.io/downloads#windows)（www.anaconda.com）】

具体使用见[Anaconda官方教程](https://conda.io/docs/test-drive.html)，简单易懂！

本文安装路径：D:\Anaconda3

安装完成后将下述路径添加到环境变量Path。

D:\Anaconda3

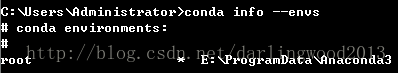
D:\Anaconda3\Scripts

## 1.1 **Anaconda初步学习**

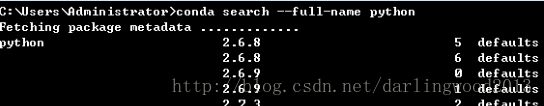
1.检查Anaconda是否成功安装：conda --version

版本

2.检测目前安装了哪些环境：conda info --envs

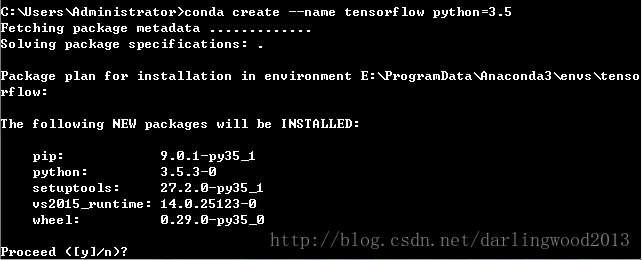


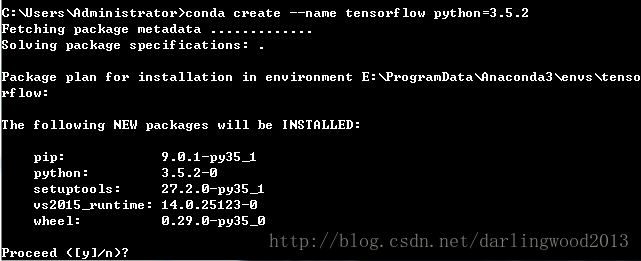
3.检查目前有哪些版本的python可以安装：conda search --full-name python

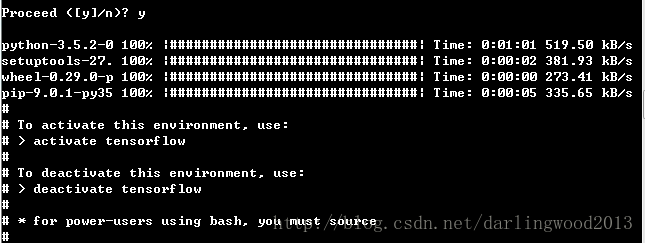


4.安装不同版本的python：conda create --name tf35 python=3.5

（注意：表示创建一个名字为tf35的python3.5环境，3.6或其他版本可能不适应于tensorflow）



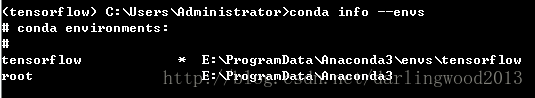




5.按照提示，激活之：activate tensorflow

http://img.blog.csdn.net/20170305110812582?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvZGFybGluZ3dvb2QyMDEz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

6.确保名叫tensorflow的环境已经被成功添加：conda info --envs



7.检查新环境中的python版本：python --version

http://img.blog.csdn.net/20170305111337328?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvZGFybGluZ3dvb2QyMDEz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

8.退出当前环境：deactivate

http://img.blog.csdn.net/20170305111547368?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvZGFybGluZ3dvb2QyMDEz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

# **2、安装Tensorflow**

## 2.1 安装前准备

TensorFlow有两个版本：**CPU**版本和**GPU**版本。GPU版本需要**CUDA**和**cuDNN**的支持，CPU版本不需要。如果你要安装GPU版本，**请先确认你的显卡支持CUDA**。我安装的是GPU版本，采用**pip安装方式**，所以就以GPU安装为例，CPU版本只不过不需要安装CUDA和cuDNN。

1. 在[这里](https://developer.nvidia.com/cuda-gpus)确认你的显卡支持CUDA。
2. 确保你的Python版本是**3.5** 64位及以上。（TensorFlow从1.2开始支持Python3.6，之前的**官方**是不支持的）
3. 确保你有稳定的网络连接。
4. 确保你的pip版本>=8.1。用pip -V查看当前pip版本，用python -m pip install -U pip升级pip。

以上条件符合，那么恭喜你可以开始下载 CUDA 和 cuDNN 的安装包了，注意版本号会由于 TensorFlow 不同版本有变化，此处请结合下面的[安装 CUDA](http://blog.csdn.net/u010099080/article/details/53418159#%E5%AE%89%E8%A3%85-cuda)和[安装 cuDNN](http://blog.csdn.net/u010099080/article/details/53418159#%E5%AE%89%E8%A3%85-cudnn)说明）。

## 2.2 安装 TensorFlow

由于 Google 那帮人已经把 TensorFlow 打成了一个 pip 安装包，所以现在可以用正常安装包的方式安装 TensorFlow 了，就是进入命令行执行下面这一条简单的语句：

# GPU版本

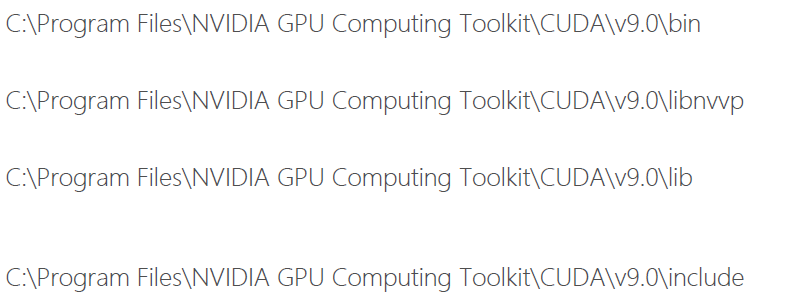
pip install --upgrade tensorflow-gpu

# CPU版本

pip install --upgrade tensorflow

我在安装gpu版本遇到“OSError raw write()”错误，转而用cpu版本。

（**注意：**使用命令pip install --ignore-installed –upgrade tensorflow-gpu却成功了）安装CUDA之后，将cuDNN解压至CUDA\V9.0目录。



CUDA环境变量

(tf35) C:\Users\96291>pip install --upgrade tensorflow

Collecting tensorflow

Downloading tensorflow-1.6.0-cp35-cp35m-win\_amd64.whl (32.3MB)

# **3、TensorFlow例程上手**

装了个新东西，我们先把它用起来吧！

概念什么的跑完第一个小程序再看！

找点成就感才好继续下去！

示例来源：[MINIST For ML Beginners](https://www.tensorflow.org/get_started/mnist/beginners)

MINST数据集：

1. 55000训练集，10000测试集，5000验证集
2. 每张图片都是28pixels\*28pixels

代码：

|  |
| --- |
| #获得数据集  from tensorflow.examples.tutorials.mnist import input\_data  mnist = input\_data.read\_data\_sets("MNIST\_data/", one\_hot=True)  import tensorflow as tf  #输入图像数据占位符  x = tf.placeholder(tf.float32, [None, 784])  #权值和偏差  W = tf.Variable(tf.zeros([784, 10]))  b = tf.Variable(tf.zeros([10]))  #使用softmax模型  y = tf.nn.softmax(tf.matmul(x, W) + b)  #代价函数占位符  y\_ = tf.placeholder(tf.float32, [None, 10])  #交叉熵评估代价  cross\_entropy = tf.reduce\_mean(-tf.reduce\_sum(y\_ \* tf.log(y), reduction\_indices=[1]))  #使用梯度下降算法优化：学习速率为0.5  train\_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross\_entropy)  #Session  sess = tf.InteractiveSession()  #初始化变量  tf.global\_variables\_initializer().run()  #训练模型，训练1000次  for \_ in range(1000):  batch\_xs, batch\_ys = mnist.train.next\_batch(100)  sess.run(train\_step, feed\_dict={x: batch\_xs, y\_: batch\_ys})  #计算正确率  correct\_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y\_,1))  accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))  print(sess.run(accuracy, feed\_dict={x: mnist.test.images, y\_: mnist.test.labels})) |

将上述代码保存为MINST.py，在控制台执行python MINST.py，运行完毕后输出模型的准确率。

# **4、安装protobuf**

<https://github.com/google/protobuf/releases>

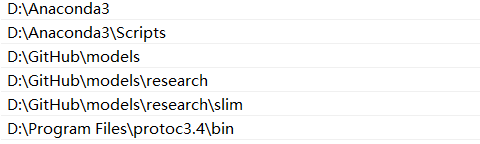
下载[**protoc-3.4.0-win32.zip**](https://github.com/google/protobuf/releases/download/v3.4.0/protoc-3.4.0-win32.zip)

# ****5、安装object\_detection API****

将<https://github.com/tensorflow/models>克隆到

D:\GitHub\models\

添加环境变量



cd /d D:\GitHub\models\research

首先执行：

protoc D:\GitHub\models\research\object\_detection\protos\\*.proto --python\_out=. --proto\_path=D:\GitHub\models\research\

会在object\_detection\protos生成一系列.py文件。（路径中”\”和”/”均可以）

运行测试文件：

(tf35) D:\GitHub\models\research>python object\_detection/builders/model\_builder\_test.py

报错：ImportError: No module named 'object\_detection'

# **6、添加tensorflow\_model.pth**

解决上述错误方法是需要添加tensorflow\_model.pth文件，填写内容如下：

D:\GitHub\models\research

D:\GitHub\models\research\slim

将此文件复制到D:\Anaconda3\envs\tf35\Lib\site-packages目录下。

需要pip安装的包：

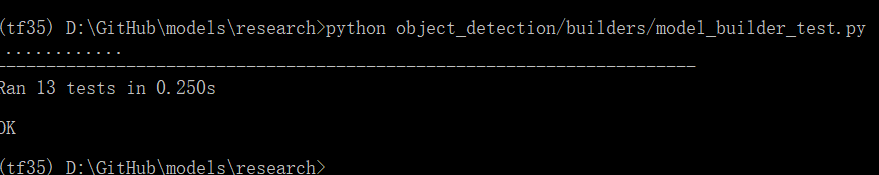
numpy

lxml

Pillow

matplotlib

最终例子运行成功：



# **7、目标检测例子**

预训练模型下载：

<https://github.com/tensorflow/models/tree/master/research/slim>

在D:\GitHub目录下新建my\_path目录，用于存放本机模型。

## 7.1 生成record

python D:/GitHub/models/research/object\_detection/dataset\_tools/create\_pet\_tf\_record.py --label\_map\_path=D:/GitHub/models/research/object\_detection/data/pet\_label\_map.pbtxt --data\_dir=D:/GitHub/my\_path/pets --output\_dir=D:/GitHub/my\_path/pets

参数说明：

--label\_map\_path 训练数据类别的描述文件

--data\_dir 存放训练数据images和annotations的文件

--output\_dir tfrecords保存路径（文件夹）

## 7.2 编辑配置文件（config）

（1）

fine\_tune\_checkpoint: "dir/model.ckpt" #上面第二步下载的预训练模型

（2）

train\_input\_reader: {

tf\_record\_input\_reader {

input\_path: "dir/pet\_train\_with\_masks.record" #预处理数据生成的tfrecords文件

}

label\_map\_path: "dir/pet\_label\_map.pbtxt" #这个文件在预处理数据时也用到过，用来说明训练样本的类别信息

}

（3）

eval\_input\_reader: {

tf\_record\_input\_reader {

input\_path: "dir/pet\_val\_with\_masks.record" #预处理数据生成的tfrecords文件

}

label\_map\_path: "dir/pet\_label\_map.pbtxt" #这个文件和（2）中是同一个文件，用来说明训练样本的类别信息

shuffle: false

num\_readers: 1

}

## 7.3 训练模型

在训练过程中可以用tensorboard查看训练过程：

tensorboard --logdir=D:\GitHub\my\_path\pets

在控制台按Ctrl+C即可停止训练，然后输入下述命令保存模型。

python D:\GitHub\models\research\object\_detection\train.py --logtostderr --train\_dir=D:\GitHub\my\_path\pets --pipeline\_config\_path=D:\GitHub\my\_path\pets\pets\_pipeline.config

参数说明：

--train\_dir：模型的checkpoint和summary都会保存在这个路径下

--pipeline\_config\_path：上一步的配置文件

## 7.4 模型暂停与继续

保存58175（训练次数）相关的训练结果，并去掉58175标记，如图所示：



然后将checkpoint置空：

model\_checkpoint\_path: "model.ckpt"

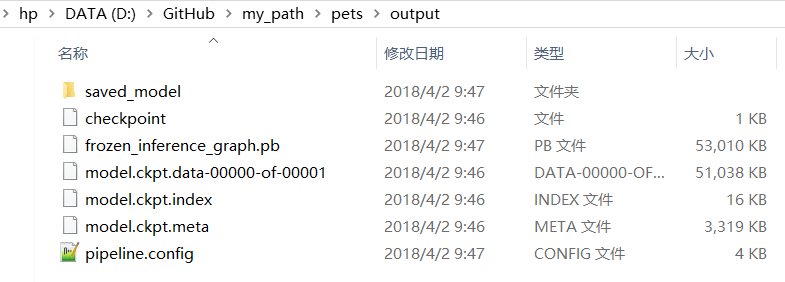
all\_model\_checkpoint\_paths: "model.ckpt"

按Ctrl+C可停止训练，下一次训练可以从58175开始。

## 7.5 模型导出

python D:\GitHub\models\research\object\_detection\export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path D:\GitHub\my\_path\pets\pets\_pipeline.config --trained\_checkpoint\_prefix D:\GitHub\my\_path\pets\model.ckpt-58175 --output\_directory D:\GitHub\my\_path\pets\output

其中58175代表训练次数。执行完上述语句后：



参考文献：

1 CSDN：TensorFlow在windows上安装与简单示例，<https://blog.csdn.net/darlingwood2013/article/details/60322258>

2 CSDN：【TensorFlow】Windows10 64 位下安装 TensorFlow - 官方原生支持，<https://blog.csdn.net/u010099080/article/details/53418159>

3 CSDN：TensorFlow在windows上安装与简单示例，<https://blog.csdn.net/darlingwood2013/article/details/60322258>

4 CSDN：[Win10使用TensorFlow调用object\_detection API](https://blog.csdn.net/Zlase/article/details/78734138)，<https://blog.csdn.net/zlase/article/details/78734138>

5 CSDN: tensorflow 轻松实现自己的目标检测, <https://blog.csdn.net/wangjian1204/article/details/79124018>