Tensorflow serving

# 简介：

tensorflow serving 是tensorflow 生成环境中部署模型的工具。主要包括server 和client两部分。其中server 需要加载模型后提供服务。

# 一： 环境安装：

1. gRPC

pip install grpcio

1. 其他依赖：

sudo apt-get update && sudo apt-get install -y \  
        build-essential \  
        curl \  
        libcurl3-dev \  
        git \  
        libfreetype6-dev \  
        libpng12-dev \  
        libzmq3-dev \  
        pkg-config \  
        python-dev \  
        python-numpy \  
        python-pip \  
        software-properties-common \  
        swig \  
        zip \  
        zlib1g-dev

1. python api ( client )

pip install tensorflow-serving-api

1. ModelServer ( server )

echo "deb [arch=amd64] http://storage.googleapis.com/tensorflow-serving-apt stable tensorflow-model-server tensorflow-model-server-universal" | sudo tee /etc/apt/sources.list.d/tensorflow-serving.list

curl <https://storage.googleapis.com/tensorflow-serving-> apt/tensorflow-serving.release.pub.gpg | sudo apt-key add -

sudo apt-get update && sudo apt-get install tensorflow-model-server

# 二： 模型导出

模型导出是将tensorflow 中一个模型（graph）连同参数保存成modelserver需要的格式。在模型保存时需要指定该模型的输入和输入。Modelserver服务就是将接收的数据传入graph的输入、执行整个graph 的流程、返回graph输出的过程。

假设模型定义如下：

|  |
| --- |
| x = tf.identity(tf\_example['x'], name='x') # use tf.identity() to assign name  y\_ = tf.placeholder('float', shape=[None, 10])  w = tf.Variable(tf.zeros([784, 10]))  b = tf.Variable(tf.zeros([10]))  y = tf.nn.softmax(tf.matmul(x, w) + b, name='y') |

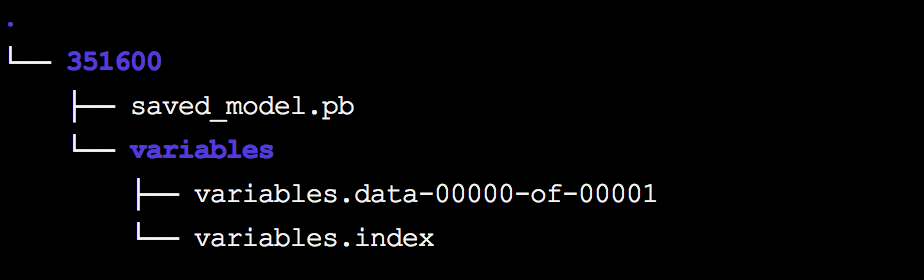
模型的预测（inferance）过程为：

|  |
| --- |
| Img = \*\*\*  Predict = Sess.run(fetches=y,feed\_dict={x:img}) |

模型导出：

|  |
| --- |
| builder = tf.saved\_model.builder.SavedModelBuilder(export\_path)  #获得 模型输入输出接口（tensor描述）信息  tensor\_info\_x = tf.saved\_model.utils.build\_tensor\_info(x)  tensor\_info\_y = tf.saved\_model.utils.build\_tensor\_info(y)  # 为保存的模型指定输入输出 ， 和类型（可以是预测、分类和回归）： 输入inputs={}， 输出：outputs={}  # 一个模型可以有多组接口（一个输入和一个输入为一组），这里定义一套接口  prediction\_signature = (  tf.saved\_model.signature\_def\_utils.build\_signature\_def(  inputs={'images': tensor\_info\_x},  outputs={'scores': tensor\_info\_y},  method\_name=tf.saved\_model.signature\_constants.PREDICT\_METHOD\_NAME))  legacy\_init\_op = tf.group(tf.tables\_initializer(), name='legacy\_init\_op')  #将要保存的信息添加到builder，sess 包含了模型的所有操作和参数，signature\_def\_map可以指定多组接口（这里只有一组）。  builder.add\_meta\_graph\_and\_variables(  sess, [tf.saved\_model.tag\_constants.SERVING],  signature\_def\_map={  'predict\_images':  prediction\_signature  },  legacy\_init\_op=legacy\_init\_op)  #写到文件  builder.save()  print 'Done exporting!' |

模型导出文件如下：



# 三： 模型加载：

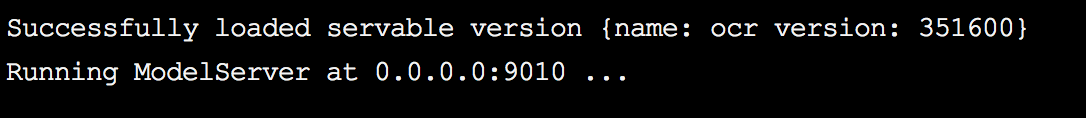
tensorflow\_model\_server

--port=9010 端口

--model\_name=ocr 服务名称

--model\_base\_path=/tmp/export 模型导出路径

成功加载后的响应：



# 四： 客户端

客户端需要指定主机、端口、服务名、接口名（一组输入和输出的名字）确定请求的服务。

|  |
| --- |
| class Serv\_client():  def \_\_init\_\_(self,host='localhost',port='9010'):  #定义主机和端口  host, port = host,port  #连接通道信息  channel = implementations.insecure\_channel(host, int(port))  self.stub = prediction\_service\_pb2.beta\_create\_PredictionService\_stub(channel)  #请求信息  self.request = predict\_pb2.PredictRequest()  #服务名称  self.request.model\_spec.name = 'ocr'  #使用哪一组接口  self.request.model\_spec.signature\_name = 'predict\_images'    def do\_inference(self,img):  #输入到接口的值，对应模型导出时的‘x’  self.request.inputs['images'].CopyFrom(  tf.contrib.util.make\_tensor\_proto(img, shape=(1, w,h)))    result\_future = self.stub.Predict.future(self.request, 3.0)  #使用回调函数的方式处理相应  result\_future.add\_done\_callback(  \_create\_rpc\_callback()) |

处理响应的回调函数

|  |
| --- |
| #处理相应用的回调函数  def \_create\_rpc\_callback():  def \_callback(result\_future):  exception = result\_future.exception()  if exception:  print(exception)  else:  response = np.array(  result\_future.result().outputs['result'].int\_val)  print（response）  return \_callback |