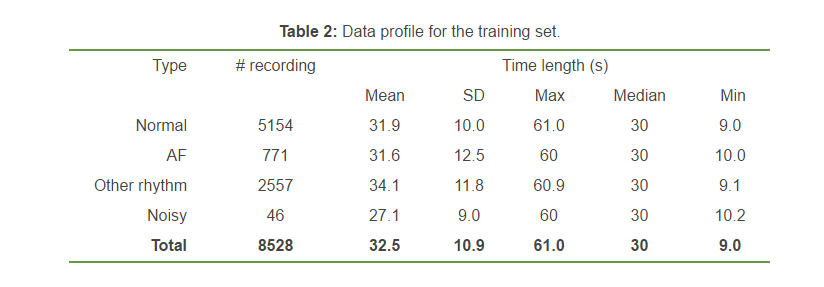
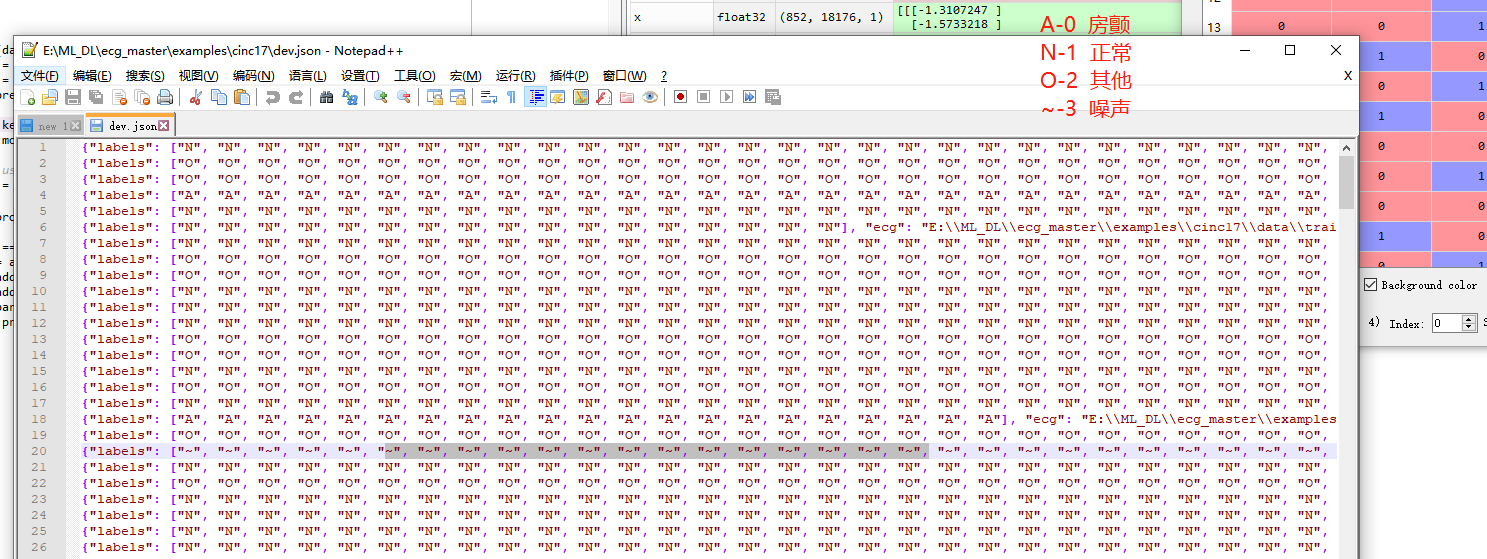
一．Demo数据和说明

These instructions go through the training and evaluation of a model on the

[Physionet 2017 challenge](<https://www.physionet.org/challenge/2017/>) dataset.

300Hz的单导联数据 9~60s长的数据



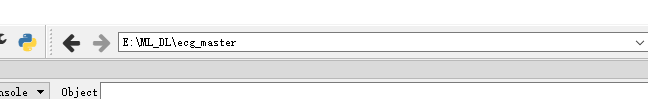


我们收集的亚心上面的房颤数据1887，每个动态截取上面4类1分钟的片段，如果没有的类型就不截取，每个动态数据最多截取上面4类各4段的数据，存为txt数据（N00001.txt......，AF00001.txt......，OR00001.txt......，NO00001.txt......），和钟兰实际截取时是一个时间段的数据（一个数据文件夹）先集中截完一个类型的，接着再从头截取其他类型的，这样中间不容易出现记录打混。由于目前心贴数据有限，只能在12导联的eHolter数据上找一个导联数据来模拟单导联数据进行数据的截取和标定，考虑到V1导联的房颤特征比较明显以及我们心贴单导联也是模拟的V1波形，我们就默认选择12导联的V1导联数据来截取，当V1数据不清楚明晰时再换其他的导联，如II导联数据来截取数据。

如果里面的数据多为全程房颤的话，就需要再额外的找其他的数据来截片段，以达到不同类型数据的平衡。

后期跑算法时用matlab转为mat格式的数据。

在spyder中的训练命令



run ecg/train.py examples/cinc17/config.json -e cinc17

测试命令

run ecg/predict.py examples/cinc17/dev.json saved/cinc17/AfPredict.hdf5

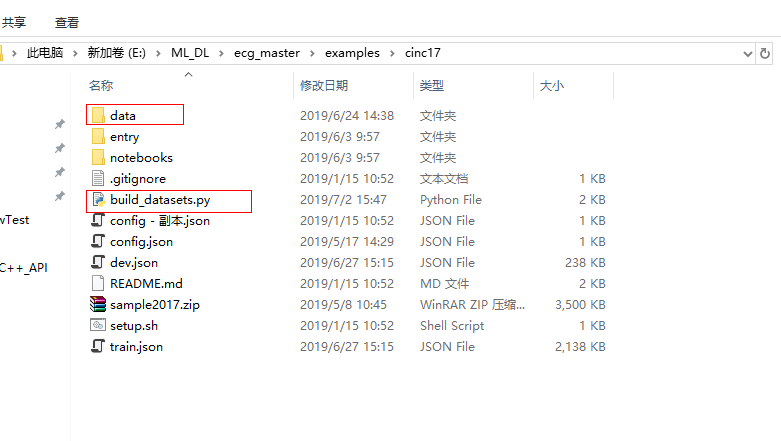
2019/11/19

数据截完，准备开始数据准备和模型训练

1. 新建eHolter房颤深度学习的数据准备文件夹：

E:\ML\_DL\ecg\_master\examples\cj\_eHolter\_AF

1. 仿照例子在cj\_eHolter\_AF文件夹里建立build\_datasets\_AF.py和相关数据文件

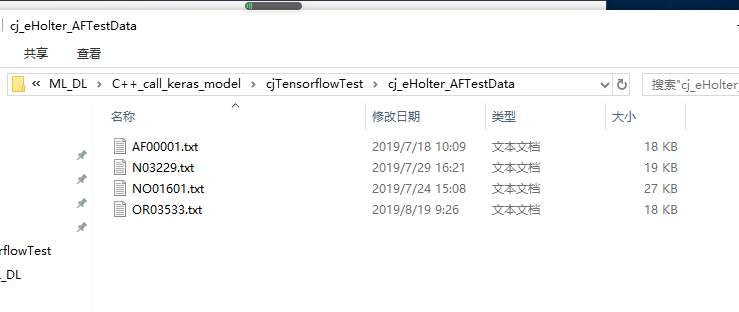


2019/11/28

1. A房颤 1-N正常 2-O其他 3-~噪声

2019/12/2

1.采用例子数据训练出来的模型运行我们eHolter上截出来的数据，结果也是意料中的不对，运行预测的结果大多时候是噪声，有时是其他类型，印证了深度学习的模型不能用于非同源的数据。



2019/12/10

1. 这2天陆续把数据都准备，整理好了，接着运行build\_datasets\_AF.py生成train.json和dev.json。

开始跑下面的程序训练模型

run ecg/train.py examples/cj\_eHolter\_AF/config\_AF.json -e cj\_eHolter\_AF

顺利跑完生成了模型

选择其中参数比较好的结果模型，并重命名为cjAfPredict.hdf5

跑验证集验证模型

run ecg/predict.py examples/cj\_eHolter\_AF/dev.json saved/cj\_eHolter\_AF/cjAfPredict.hdf5

比对了几个验证结果，都是对的

**在spyder中函数需要输入参数时用下面的方式运行，不需要输入参数时直接点击spyder中的运行按钮或者调试运行按钮**

**在E:\ML\_DL\ecg\_master目录下输入下面的命令：**

**训练：**

%run ecg/train.py examples/cj\_eHolter\_AF/config.json -e cj\_eHolter\_AF（直接运行）

%run -d ecg/train.py examples/cj\_eHolter\_AF/config.json -e cj\_eHolter\_AF（调试模型）

**预测：**

%run ecg/predict.py examples/cj\_eHolter\_AF/dev.json saved/cj\_eHolter\_AF/cjAfPredict.hdf5

%run -d ecg/predict.py examples/cj\_eHolter\_AF/dev.json saved/cj\_eHolter\_AF/cjAfPredict.hdf5

**前面加%是单步，不加%是一开始就运行到断点处。（有时又不灵，这个spyder软件还不成熟，不太稳定）**

2019/12/11

1. 将生成的模型转换成tensorflow的pb模型格式，供C++调用；

方法见E:\ML\_DL\C++\_call\_keras\_model\C++\_call\_keras\_model.docx

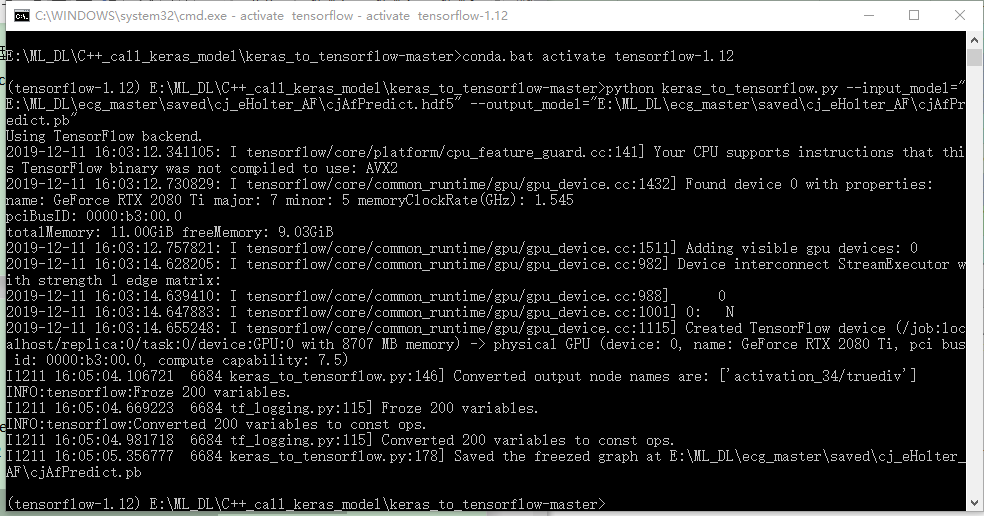
打开 cmd.exe

进入目录下cd E:\ML\_DL\C++\_call\_keras\_model\keras\_to\_tensorflow-master

激活tensorflow环境 activate tensorflow-1.12

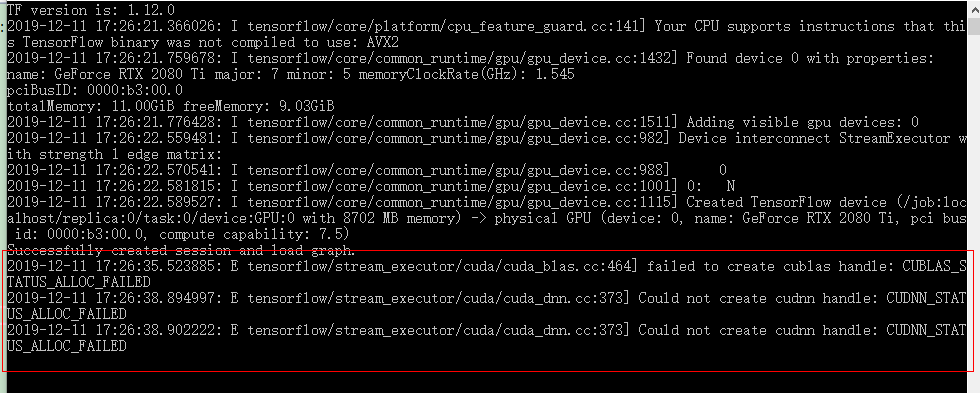
输入转换命令

python keras\_to\_tensorflow.py --input\_model="E:\ML\_DL\ecg\_master\saved\cj\_eHolter\_AF\cjAfPredict.hdf5" --output\_model="E:\ML\_DL\ecg\_master\saved\cj\_eHolter\_AF\cjAfPredict.pb"



转换成功

1. 打开vs测试工程调用生成的pb模型



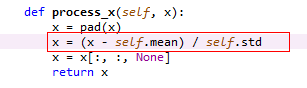
之前测试运行的好好的，这次突然报上面的错误，直接退出！

回想了下应该是跑模型的python程序没关，占用太大的内存空间导致vs程序退出。

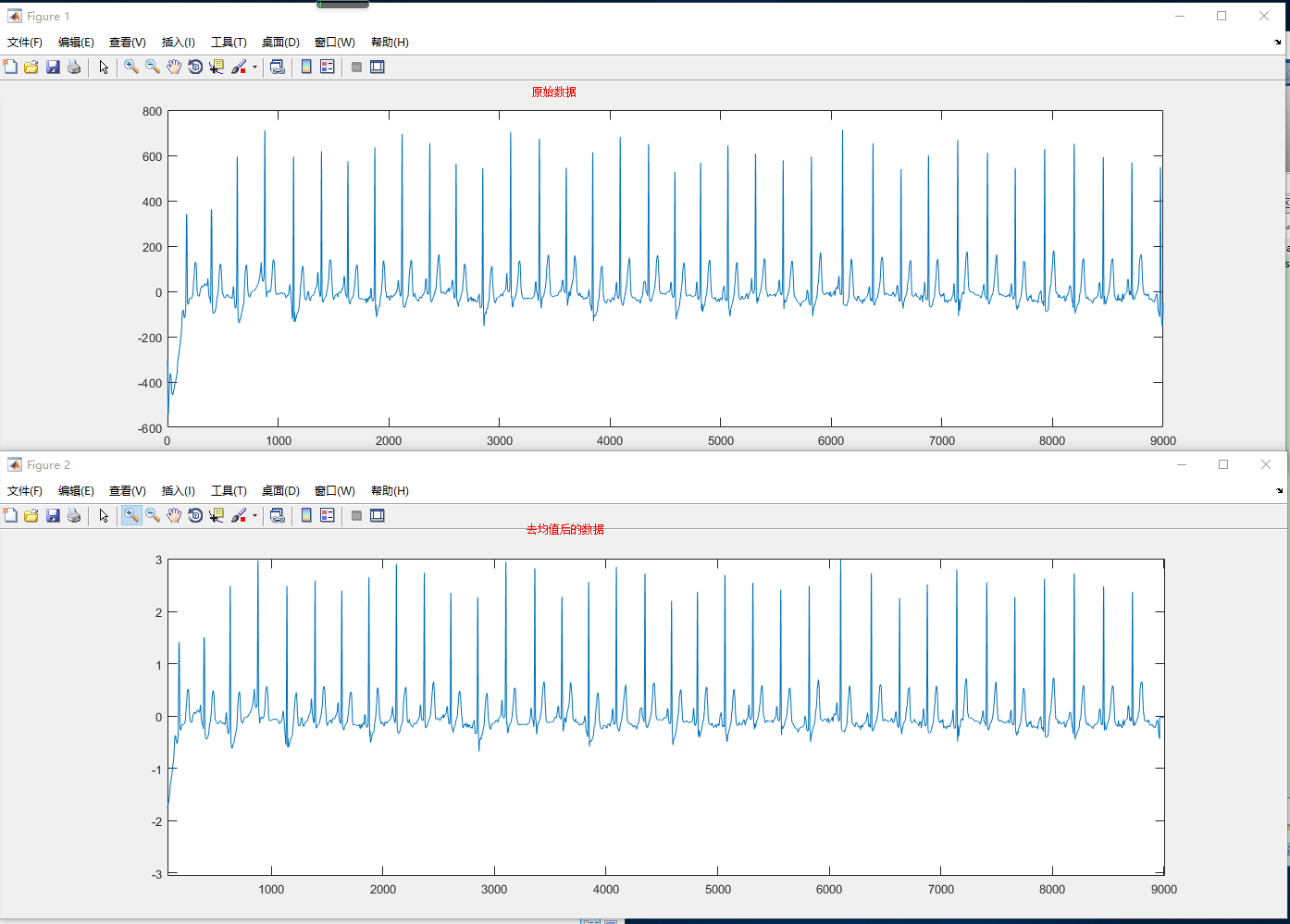
在python里用reset命令清空后再跑vs程序，程序运行OK，不退出！

但是运行的结果不对！

思索对比后感觉应该是原始数据要做个归一化的预处理后再调用模型预测，如下之前测试时调用模型预测对的数据就是从python里导出来的归一化后的数据

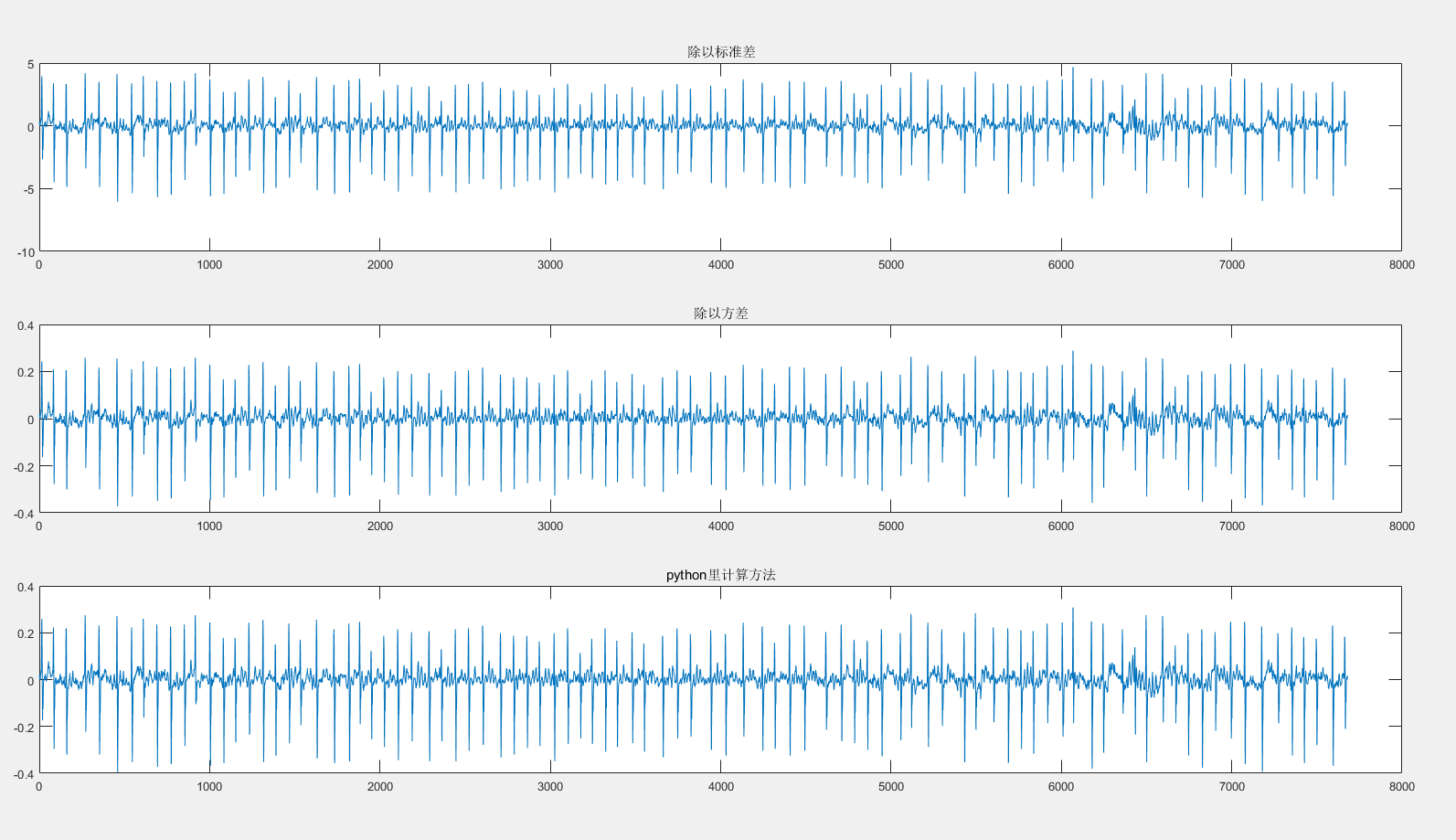


由于是批量测试，上面的self.mean和self.std是一个共同的值，是把所有数据串成一个数据计算的均值和标准差。通过调试这2个值应该是通过训练数据计算而得，且存在了模型里，在测试验证数据时，该值就是已经计算好的跟训练数据一样的值。

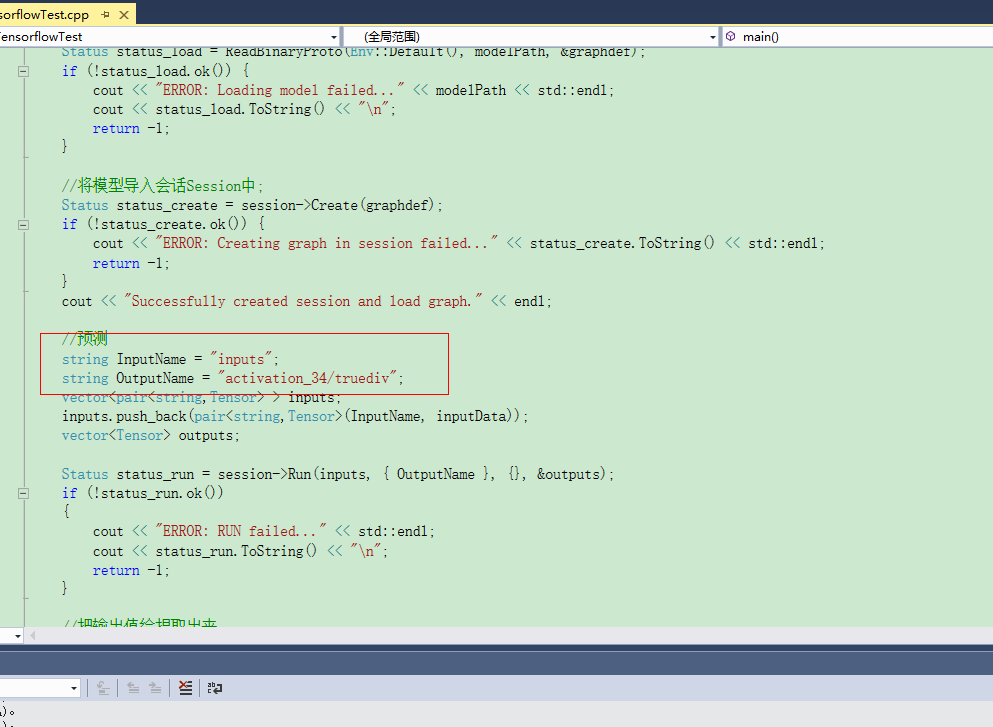


在验证数据里找了9个数据来测试验证，先是按照（x-均值）/标准差的方法归一化，但是9个数据中有3个数据结果不对（N33278-dev2\_1to2.txt，AF22675-dev4\_0to3.txt，N25355-dev5\_1to3or2.txt），后来将归一化方法改成（x-均值）/方差，结果就都对了！前后的差别就是数值的范围不一样，后者的值范围小，如下面所示：

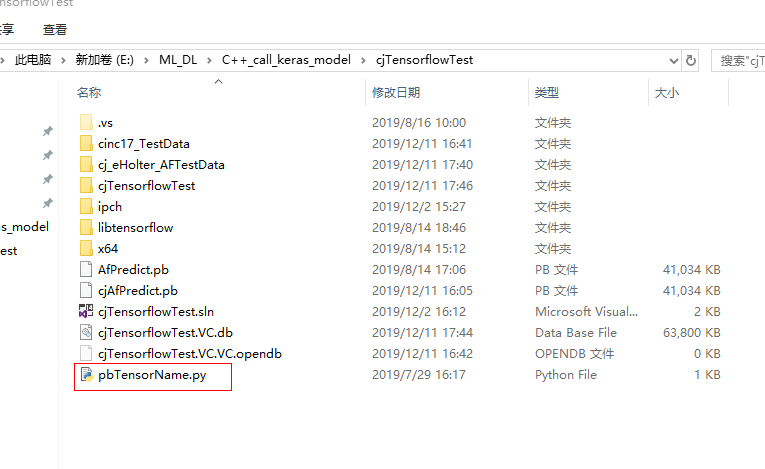
使用python测试验证集代码里的归一化方法（即上面的self.mean和self.std）处理后的数据在C++测试时结果也都是对的。

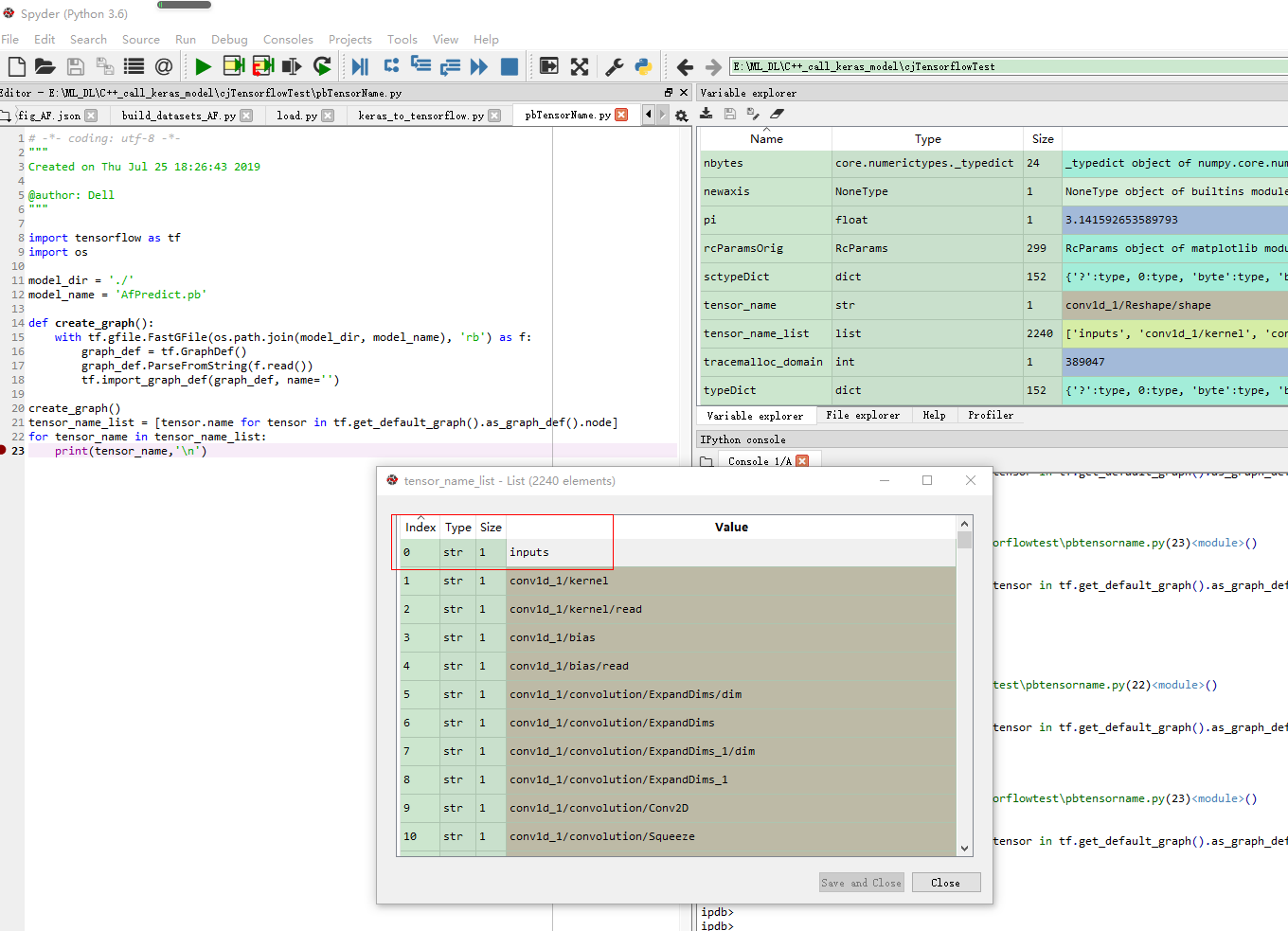


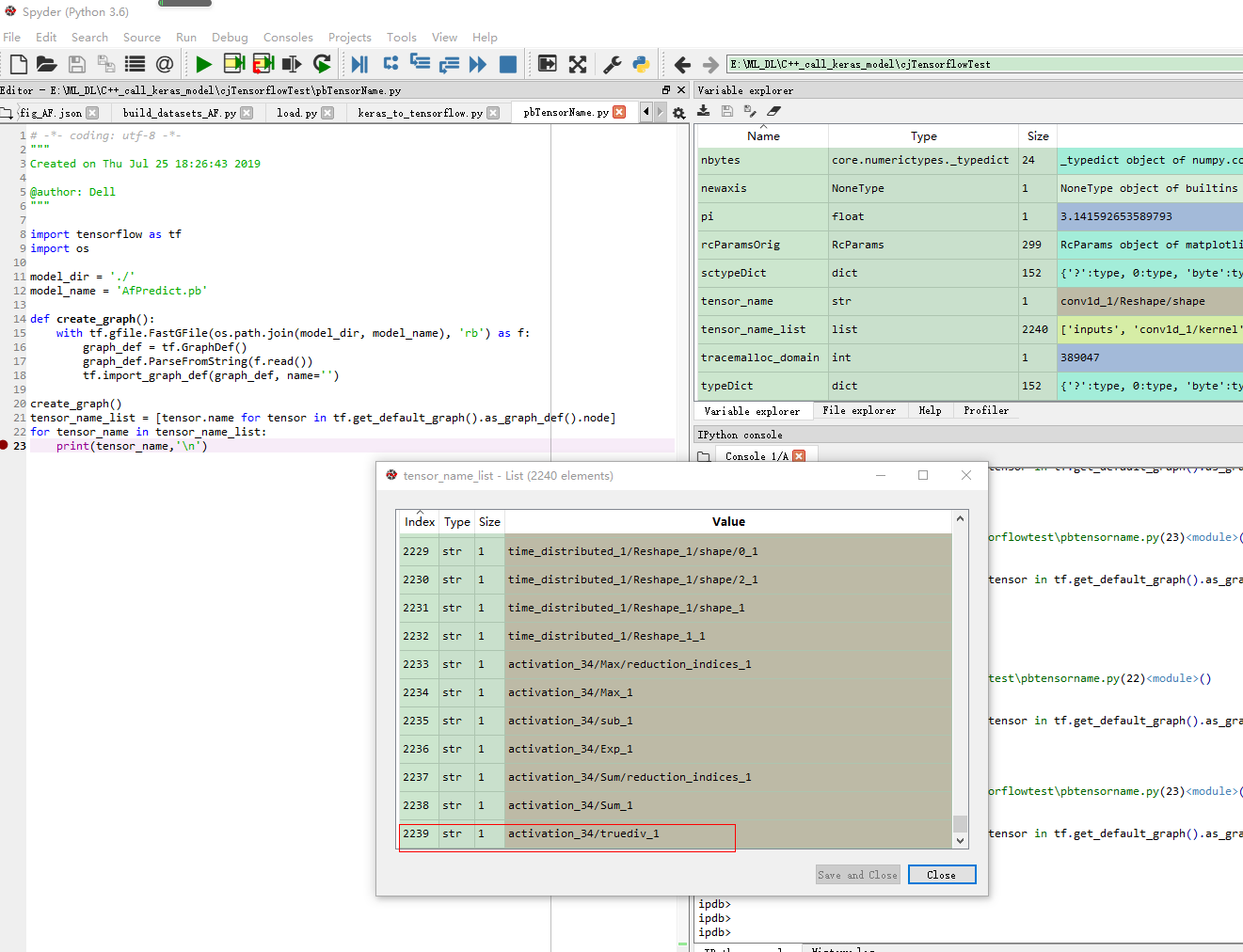
1. 其中回顾到的一个技术点是：vs在调用pb模型时需要知道模型的输入/输出名字，如下代码红框中的部分



这个名字是通过下面的python程序获取的







分别对应tensor\_name\_list里的第一个和最后一个值。

1. 注意测试在release下,因为tensorflow\_cc.dll和tensorflow\_cc.lib就是在release下编译的，有时需要调试也可以在debug下运行下，但有时会报错，报错如下：

