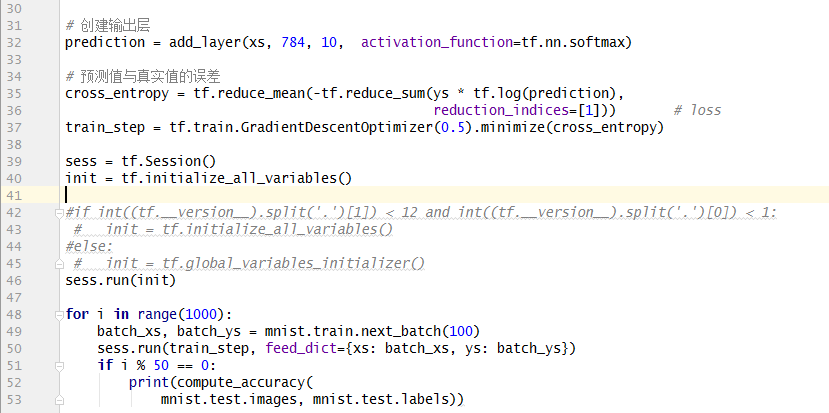
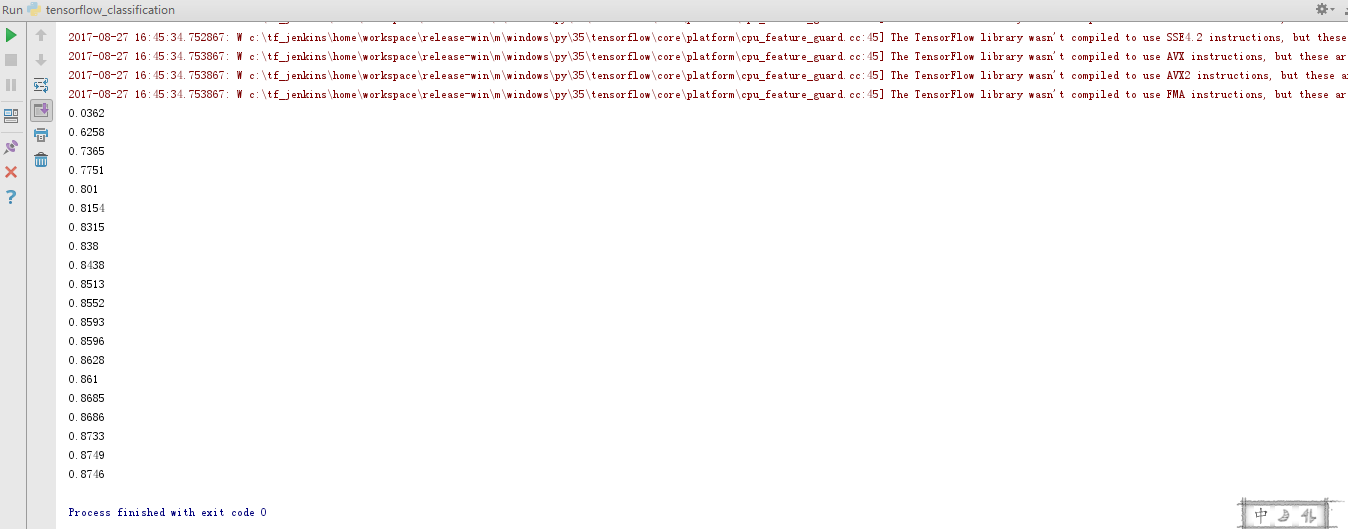
# Tensorflow 相关学习

## 1、使用tensorflow实现Classification

机器学习中的监督学习(supervised learning)问题大部分可以分成 Regression (回归)和Classification(分类)这两种.。使用Tensorflow都可以做到。回归是说要预测的值是一个连续的值，比如房价，汽车的速度，飞机的高度等等。而分类是指要把东西分成几类，比如猫狗猪牛等等。我之前的练习都是在实现 regression，这一次我尝试了用 Tensorflow 做 classification。

下面是具体的代码，数据集依然选用了mnist：

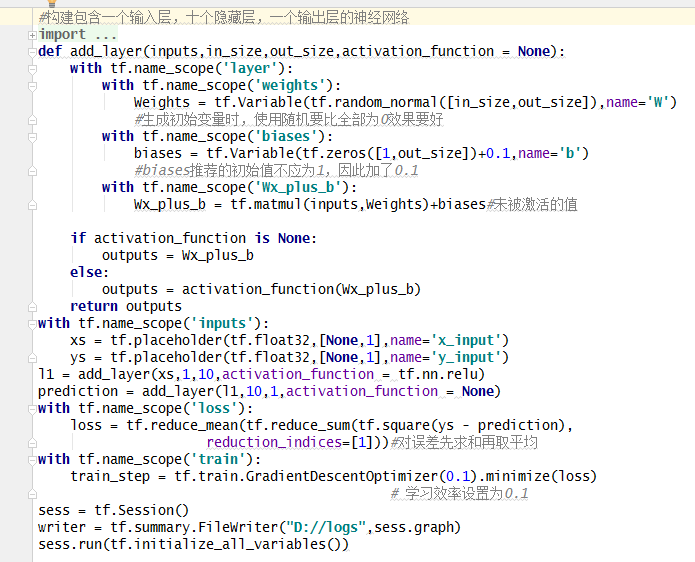
运行之后产生分类正确率的输出：



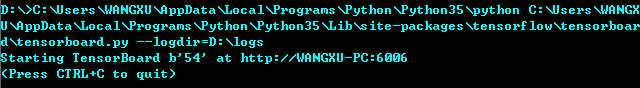
## 2、Tensorboard的初步使用

Tensorboard是Tensorflow自带的可视化模块，通过使用Tensorboard，我们能更直观的看见整个神经网络的结构，以及各个参数的变化情况并绘制变化图表。TensorBoard 是通过读取 TensorFlow 的event文件来运行的（TensorFlow 的event文件包含了在 TensorFlow 运行中涉及到的主要数据）。

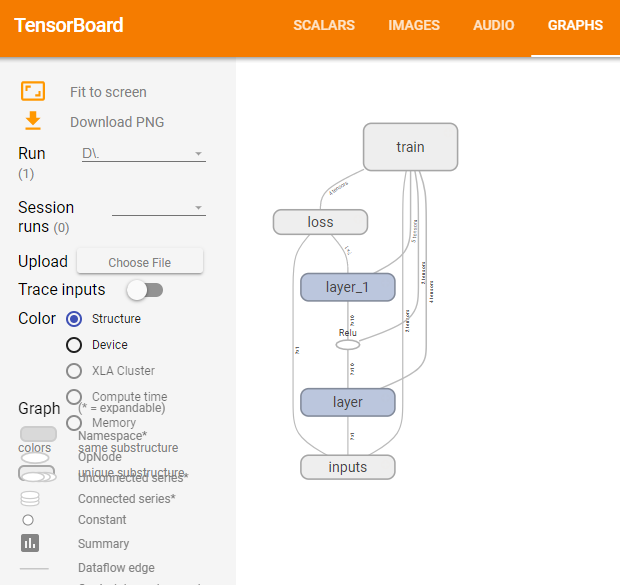
下面是在上次构建的简单神经网络的代码基础上，添加了各层和变量节点的命名之后整合而成的代码：



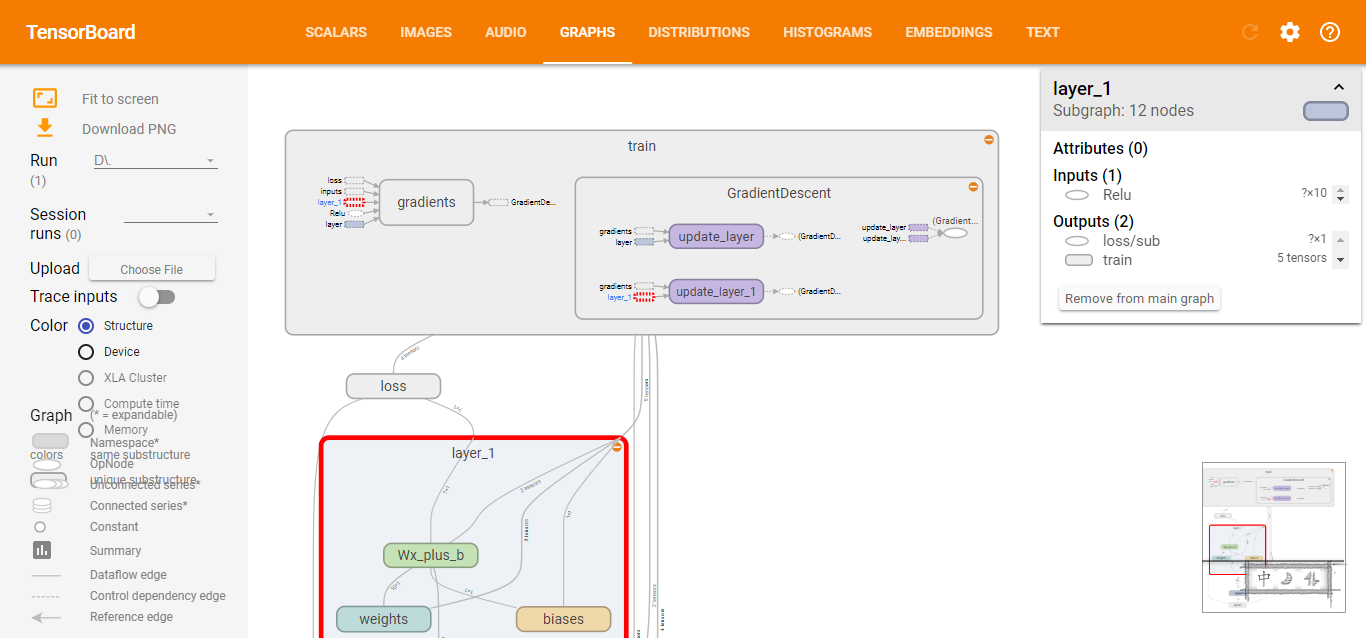
在成功生成了event文件之后，下面可以启动Tensorboard了。启动时第一个路径是python的路径（注意这个python必须是安装了TensorFlow的python），第二个路径是tensorboard.py所在的路径， logdir必须是运行程序时创建writer时使用的路径：



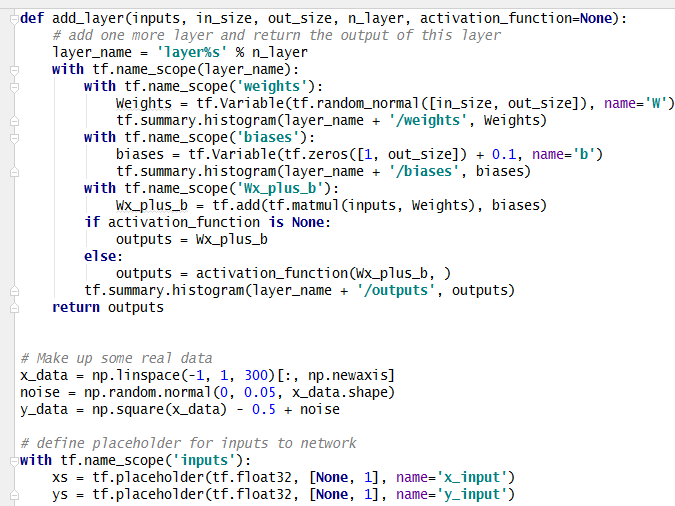
运行成功后，在浏览器输入主机名:6006 就可以访问到tensorboard的结果了。如下图展示的即为我所创建的神经网络的结构：

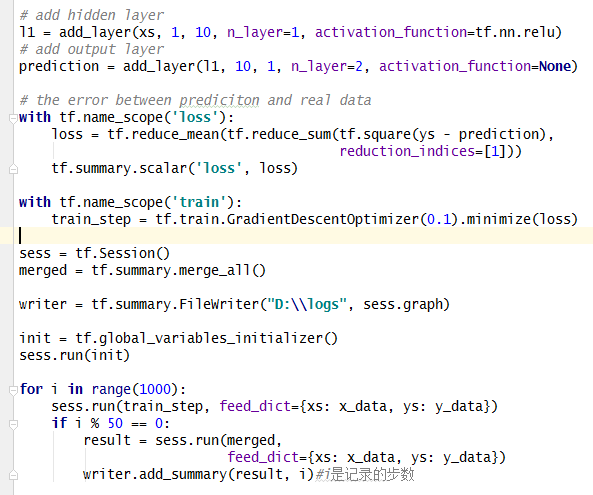


点击每个结构中的节点，还可以展开看到内部的结构：

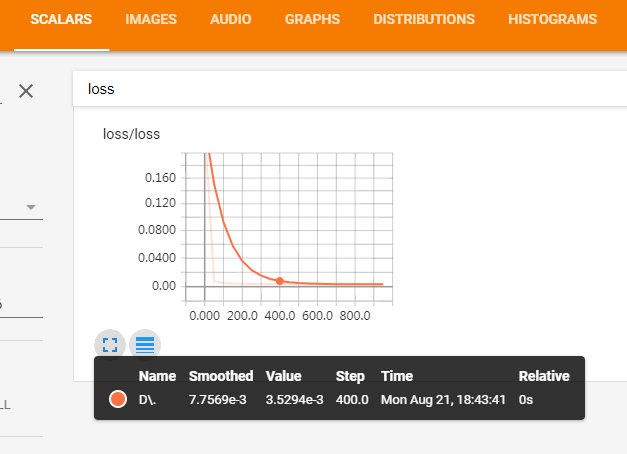


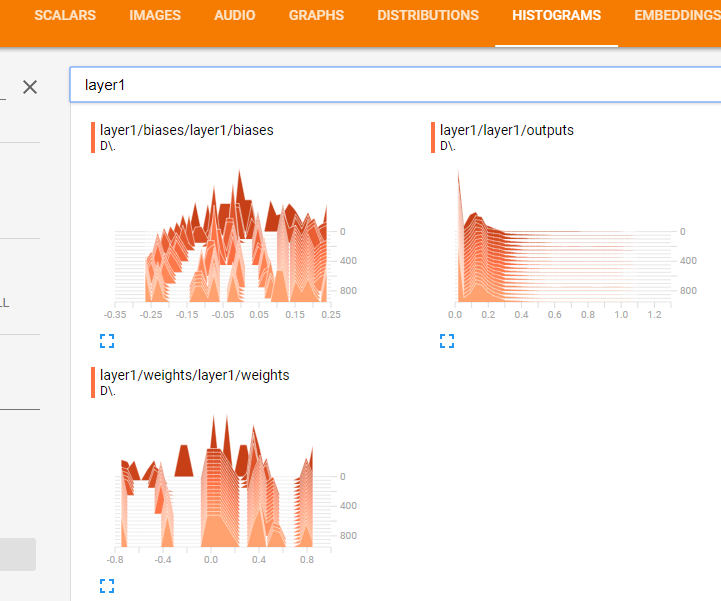
关于查看参数和输出的变化图，我找了另外一个简单的小例子试了一下，跟上一个程序相比，这个例子使用了 histogram\_summary 运算来收集权重变量和梯度输出：

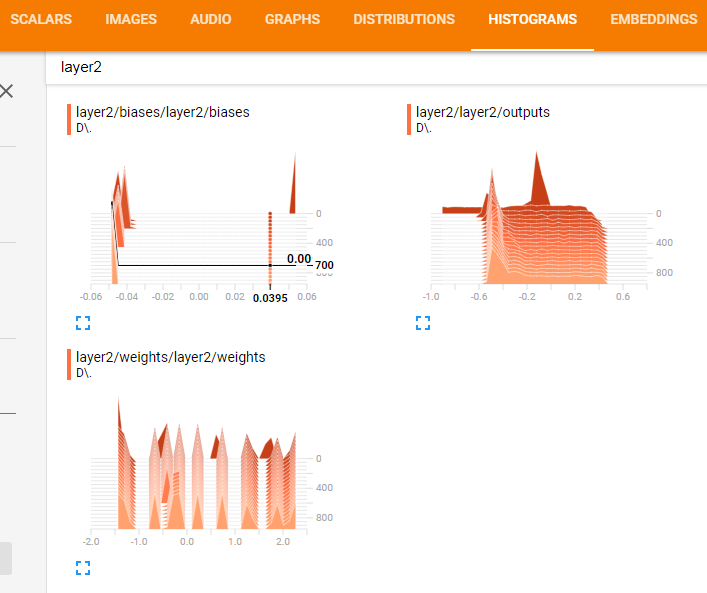




运行之后打开tensorboard，便可以看到loss、weights等参数的变化图了：







## 3、关于过拟合及解决方法

所谓过拟合，是指为了得到一致假设而使假设变得过度复杂的情况。“一个过配的模型试图连误差都去解释（而实际上误差是不需要解释的），导致泛化能力（经过查看资料，我对泛化的理解是，模型在它处于学习的过程中时，遇到没有遇见过的样本时，只根据已学习到的内容产生的表现）比较差，显然就过犹不及了。”这句话很好的诠释了过拟合产生的原因，不过这只是一部分原因，另一个原因是模型本身并不能很好地匹配数据，也就是说观测到的数据并不是由该模型产生的。究其具体原因，产生过拟合是因为：①由于对样本数据，可能存在隐单元的表示不唯一，即产生的分类的决策面不唯一。随着学习的进行,。BP算法使权值可能收敛过于复杂的决策面。②权值学习迭代次数足够多(Overtraining)，拟合到了训练数据中的噪声和训练样例中没有代表性的特征。

**过拟合解决方法**

为了防止过拟合，需要用到一些方法，此处提到的主要是：early stopping、数据集扩增（Data augmentation）以及Dropout。

①Early stopping

  对模型进行训练的过程即是对模型的参数进行学习更新的过程，这个参数学习的过程往往会用到一些迭代方法，如梯度下降（Gradient descent）学习算法。Early stopping便是一种用迭代次数截断的方法来防止过拟合的方法，即在模型对训练数据集迭代收敛之前停止迭代来防止过拟合。

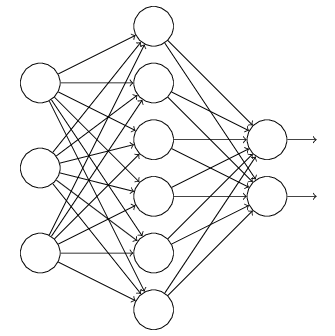
  Early stopping方法的具体做法是，在每一个Epoch结束时（一个Epoch集为对所有的训练数据的一轮遍历）计算validation data的accuracy，当accuracy不再提高时，就停止训练。这种做法很符合直观感受，因为accurary都不再提高了，在继续训练也是无益的，只会提高训练的时间。一般的做法是，在训练的过程中，记录到目前为止最好的validation accuracy，当连续10次Epoch（或者更多次）没达到最佳accuracy时，则可以认为accuracy不再提高了。此时便可以停止迭代了（Early Stopping）。这种策略也称为“No-improvement-in-n”，n即Epoch的次数，可以根据实际情况取，如10、20、30等等。

②数据集扩增

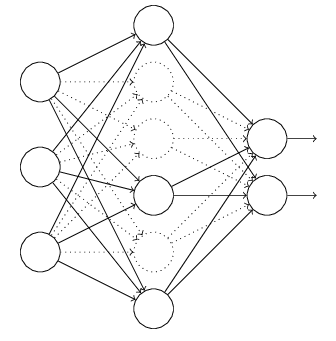
  在数据挖掘领域流行着这样的一句话，“有时候往往拥有更多的数据胜过一个好的模型”。因为在使用训练数据训练模型，通过这个模型对将来的数据进行拟合，而在这之间又一个假设便是，训练数据与将来的数据是独立同分布的。即使用当前的训练数据来对将来的数据进行估计与模拟，而更多的数据往往估计与模拟地更准确。因此，更多的数据有时候更优秀。但是往往条件有限，如人力物力财力的不足，而不能收集到更多的数据，如在进行分类的任务中，需要对数据进行打标，并且很多情况下都是人工得进行打标，因此一旦需要打标的数据量过多，就会导致效率低下以及可能出错的情况。所以，往往在这时候，需要采取一些计算的方式与策略在已有的数据集上进行手脚，以得到更多的数据。通俗地讲，数据机扩增即需要得到更多的符合要求的数据，即和已有的数据是独立同分布的，或者近似独立同分布的。一般有从数据源头采集更多数据、复制原有数据并加上随机噪声、重采样以及根据当前数据集估计数据分布参数，使用该分布产生更多数据等方法。

③Dropout

在神经网络中，有一种方法是通过修改神经网络本身结构来实现的，其名为Dropout。该方法是在对网络进行训练时用一种技巧（trick），对于如下所示的三层人工神经网络：



对于上图所示的网络，在训练开始时，随机得删除一些（可以设定为一半，也可以为1/3，1/4等）隐藏层神经元，即认为这些神经元不存在，同时保持输入层与输出层神经元的个数不变，这样便得到如下的ANN：



然后按照BP学习算法对ANN中的参数进行学习更新（虚线连接的单元不更新，因为认为这些神经元被临时删除了）。这样一次迭代更新便完成了。下一次迭代中，同样随机删除一些神经元，与上次不一样，做随机选择。这样一直进行瑕疵，直至训练结束。