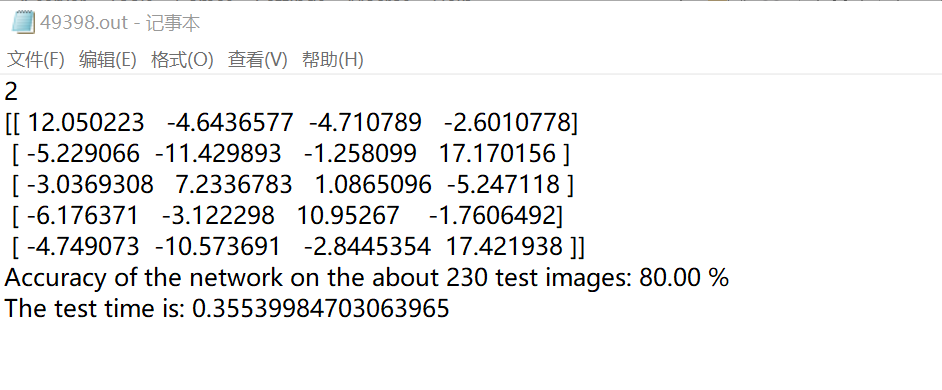
我们这里采用VGG16模型进行训练，对玉米生长状态进行分类，大致有'healthy', 'big\_leaf\_spot', 'small\_leaf\_spot', 'rust'四类，我们首先将数据集按照训练集占八成，测试集占两成分割，在训练集和测试集里分别新建四个文件夹，命名为0，1，2，3，里面放入分割好的对应的图片，每个文件夹里的图片都是同一种类。例如训练集和测试集0里放入的是是'healthy'的玉米叶图片，所以最终的输出结果0就默认对应'healthy'类。

我们采用的batch\_size = 32，初始学习率为0.01，然后我们一共训练50个epochs，每十个epochs更新一遍学习率，更新方式为lr\_new = 0.5\*lr\_old。这样做的一个原因是因为训练的epoch越大，我们离训练最优越近，这时我们需要把学习率改的更小以提升收敛效率。我们训练集一共是1152张各类玉米图片，也就是一个epoch需要训练36个batch，我们选择每9个batch将训练的loss打印出来，每个epoch结束打印学习率。加载数据集我们采用torchvision包带的ImageFolder函数。 VGG16的具体实现我们在nn.Module的继承类里实现，包括16层网络，13层卷积层和3层全连接层。由于原始VGG16处理的图片为224\*224尺寸的图片，而我们要处理的图片为256\*256尺寸格式为.jpg的图片，所以只需要修改一些参数，也就是我们需要按照我们自己的图片经过13层卷积之后得到的结果压成一维数组，再进行全连接层的处理。三层全连接层之后输出4个数据（原始VGG16模型是针对 ImageNet 图像库中的1000种类别分类，所以输出的是1000个数据），分别对应玉米叶每种种类的概率。我们训练的时候采用cuda调用显存加快训练速度。对于分类问题，我们常用交叉熵损失函数作为损失函数，这里也不例外。优化器我们选择SGD优化，momentum设置为0.8,权重衰减设置为0.001，一定程度上能防止模型过拟合。

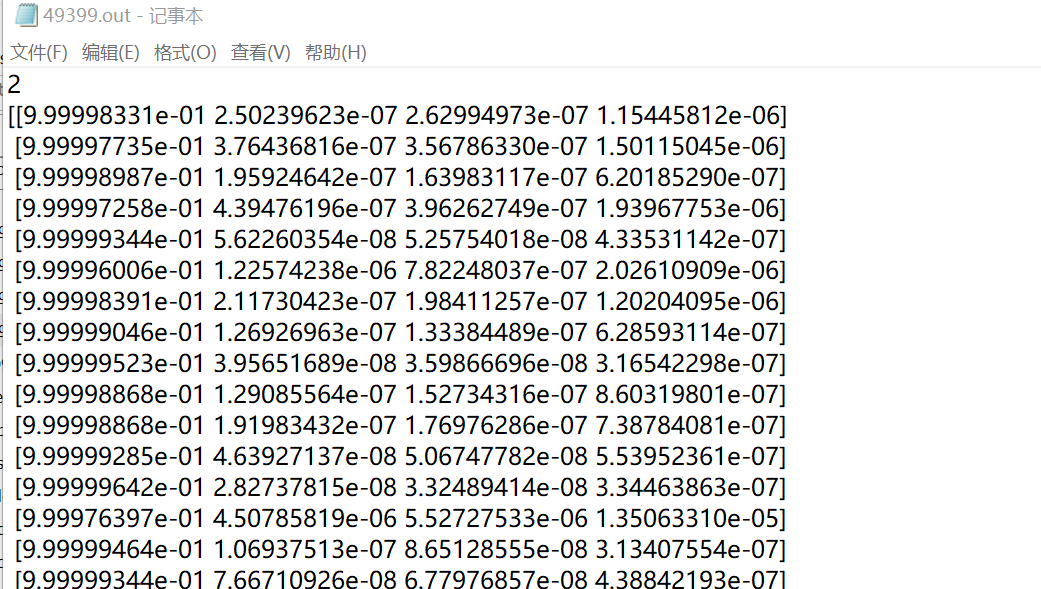
就这样我们开始训练，epoch从1循环到50，每个epoch里面一个batch一个batch地往模型里送，先是前向传播求loss，然后是逆向对各参数求导，再按照学习率，权重衰减更新各参数。我们的训练平台是NVIDIA Tesla V100 32GB显卡，训练时长约八分钟，经过五十个epoch的训练，我们绘出每个epoch的平均loss曲线可以看出，模型已经收敛。所以我们开始测试测试集，经过测试我们模型的精度能达到95%以上。

这里我们使用相对路径查找预训练模型和测试集，只要跟pretrained\_VGG16.py放在一个文件夹里就能被检索到。只要打开并运行pretrained\_VGG16.py就可以载入预训练模型也就是popcorn\_classification\_model.pkl。如果只是需要测试几张图片的话可以把测试图片放在对应的文件夹下，例如一张健康的玉米叶，放到test的0文件夹下，推理出来的结果会告诉你它是哪个种类的玉米叶，prediction跟真实的label做对照能得到准确率。如下图：

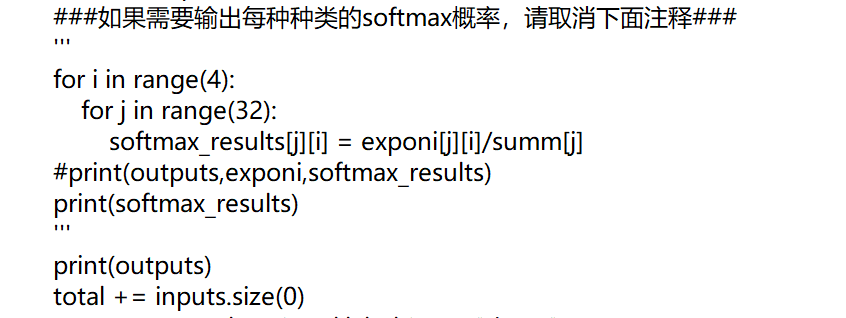


2代表我的节点上有两块gpu，下面的矩阵每行代表着每张图片推导的结果，我在0文件夹放了两张玉米叶图片，一张健康的（0），一张带锈病的（3），1文件夹里放了一张大叶病（1），2文件夹下面放了一张小叶病（2），3文件夹下放了一张锈病（3），推导出来的每行四个数中最大的数对应的索引就是预测的label。很明显我们发现虽然我把锈病的图片放在了健康的文件夹里，但是预训练模型还是识别出来了，由于这一张我放的位置不对，所以正确率为4/5，也就是80%。最后一行信息代表这几张图片推理需要的时间。

这样可能还是有点不直观，我们还可以用softmax把每张图片对应的每一行值转变成图片是某个label的概率（图片张数为batch整数倍的时候（32），否则会报错），如下图，由于测试集我没有shuffle，所以默认从0文件夹开始测试，我们发现最前面的都是预测图片label为0，也就是健康的玉米叶。



要实现上面的功能，需要在程序里取消下图中的注释，并注释掉print(outputs)。



**如果我们不知道一张图片的种类，我们可以直接找个test下面的文件夹放进去（0，1，2，3），不一定要是该图片的真实label，因为我们也不知道真实label是什么，推理出来的1\*4向量结果会告诉我们模型预测出的结果是什么。**