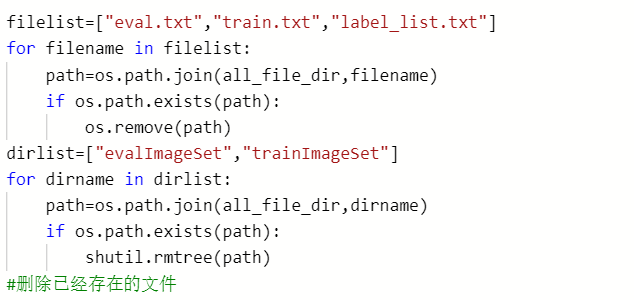
**实验5：VGG网络**

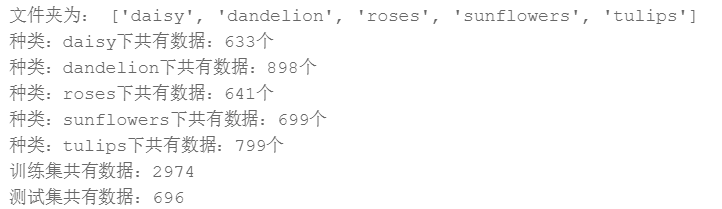
张靖祥 计算机172 2017304010413

1. 代码分析
   1. notebook代码分析

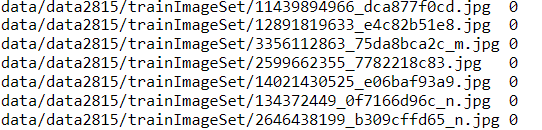
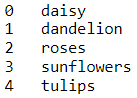
在预处理数据的过程中，我添加了如下代码，用于删除已经存在的文件/目录



数据的种类数量为：



将label标签与其对应的数值的映射，写入到label\_list.txt中

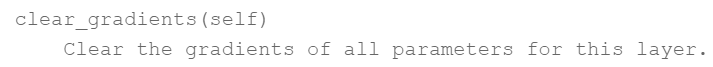
 

* 1. train文件代码分析

该文件调用了config、Vgg、reader三个python文件

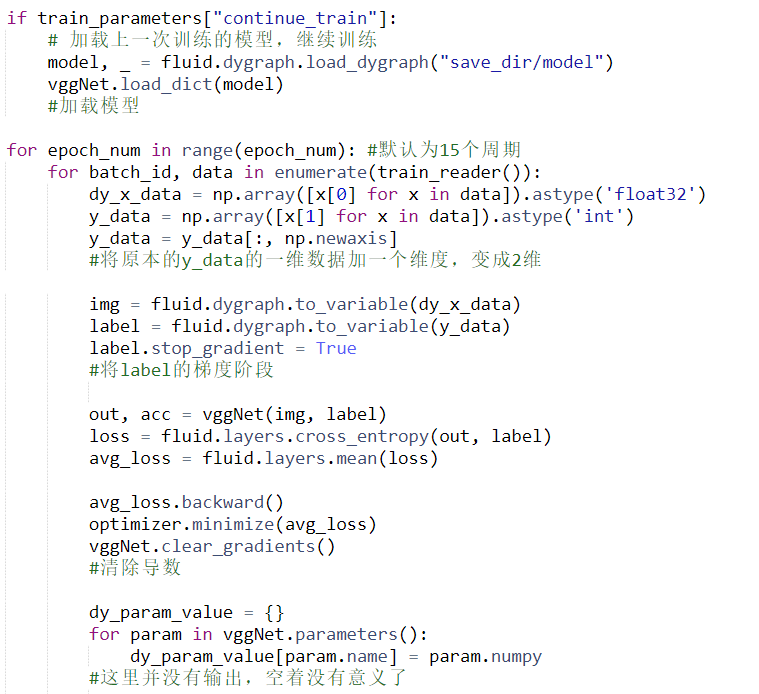
对于stop gradient的理解：<https://zhuanlan.zhihu.com/p/35003434> 相当于手动梯度消失

对于clear gradient的解释：清除所有的导数



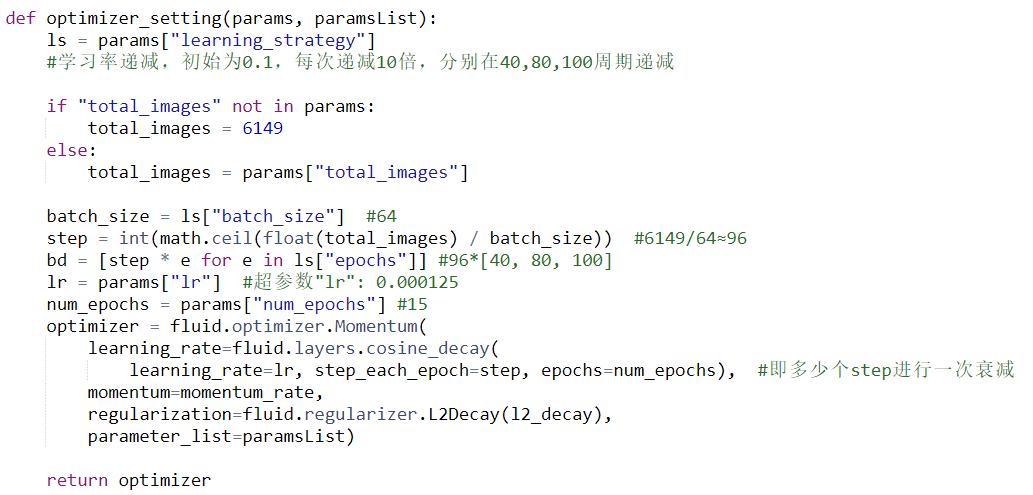
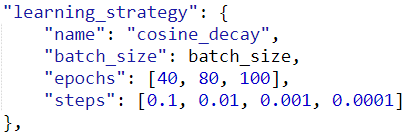
代码的注释：





Cosine decay的理解：<https://www.sohu.com/a/217389557_717210> 网页的后面，是诸多学习率衰减的方法之一（其实我认为不同的学习率衰减的方法，对最终结果的影响差别不大）

对于优化器的设定，我认为存在以下问题



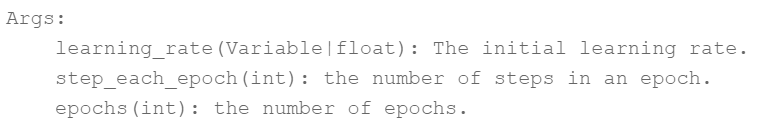
Bd并没有用到，应当删除

按照上面的代码设定，total\_image=6149，而实际的总数量为3000左右，所以会出现计算的问题

Cosine decay的调用如下fluid.layers.cosine\_decay(learning\_rate=lr, step\_each\_epoch=step, epochs=num\_epochs)

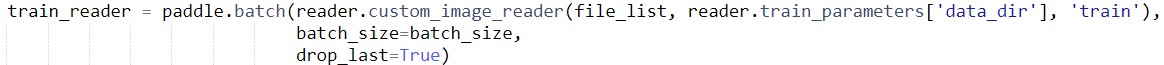
Ir为超参数，step为6149/64≈96，与实际不符，周期为15个，下面会说具体怎么改

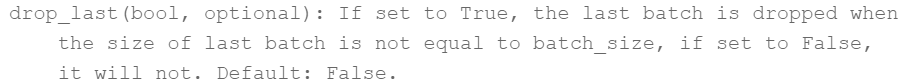
下面传入的参数，依次为：初始学习率、每个epoch的step、总的epoch



可以看出，本代码中初始学习率Ir=0.000125，实在是太低了，根本不用衰减。

最后一个问题是，paddle的batch中的drop\_last参数，解释如下，用于丢弃最后一个大小不合格的batch



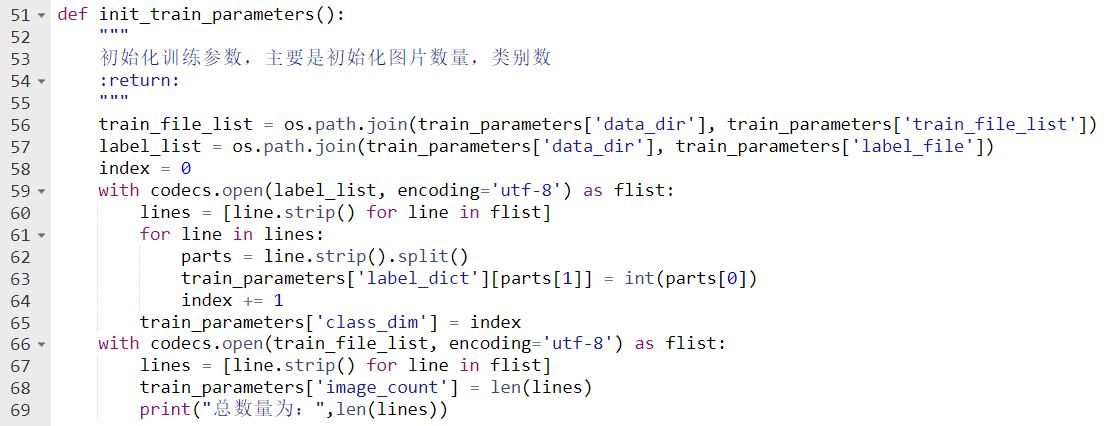


* 1. config文件说明

config的最后调用了init函数

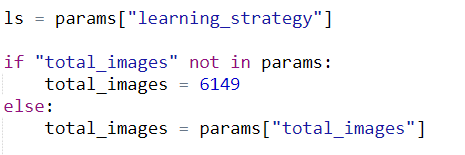


初始化训练函数，有两个作用，1、计算数据的维度（class\_dim），2、计算数据的数量（image\_count）



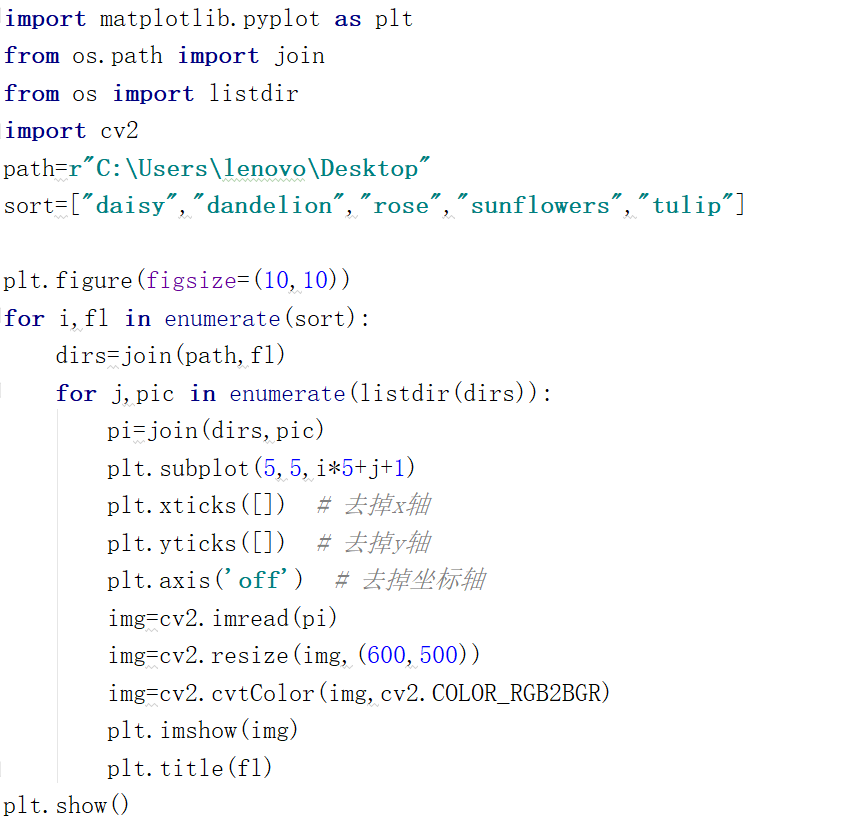
于是image\_count被记录到parameter中。

因此，在train中的optimizer setting中应当改为：

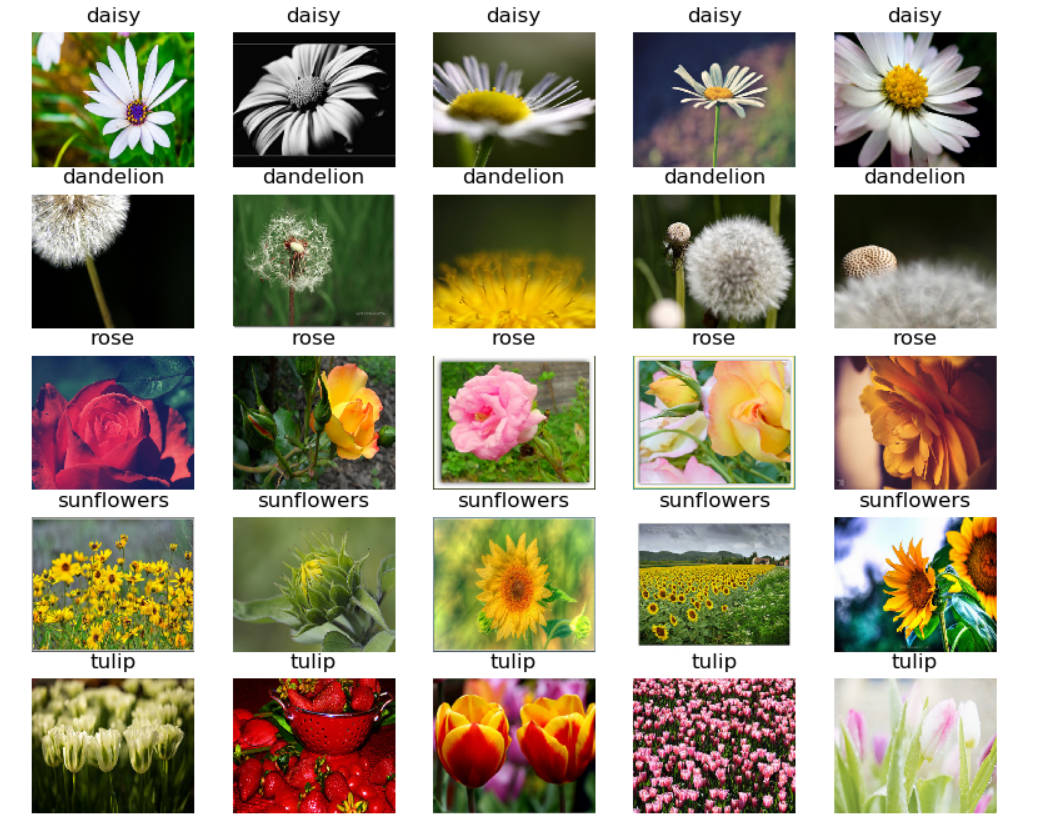


* 1. 数据分析

我将每个组的花分别下载5张，并使用了如下的代码进行显示



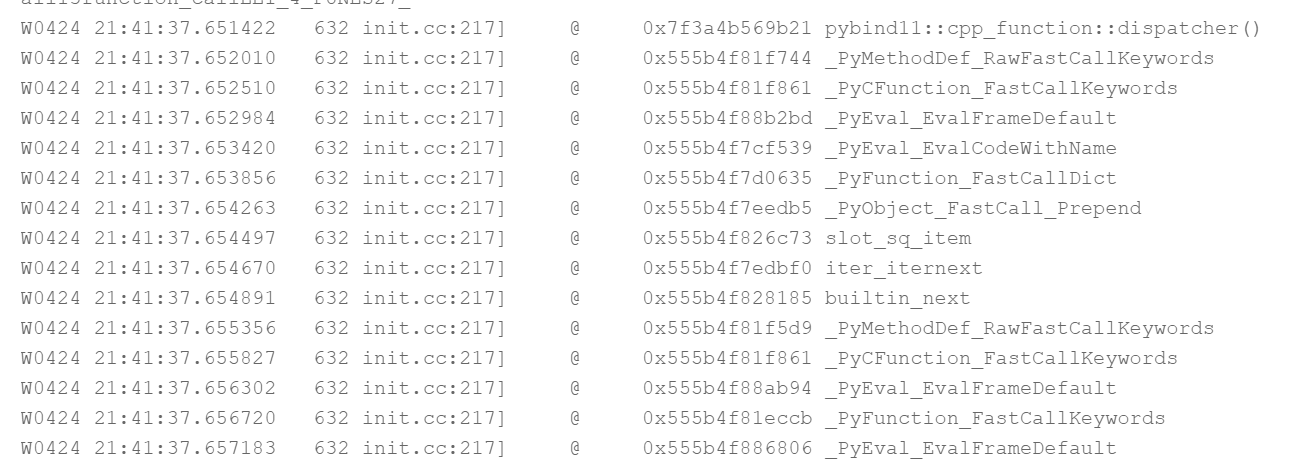
美丽的鲜花



直观上来看，数据的区分度确实很低，确实难以分辨。并且一共只有3000张图片，通过数据的随机变化（数据扩增），每个epoch只能学习3000张不用的图片，一共只运行15个epoch，总的学习数量还不足45000个。可以预知其准确率不会很高。

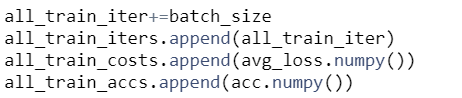
1. 实际的训练
   1. 训练过程显示

首先，我尝试在py文件里画图，结果失败，原因：图像输出无法显示在控制台



于是我觉得将训练的数据保存下来，再回到jupyter notebook 中读取并绘图显示

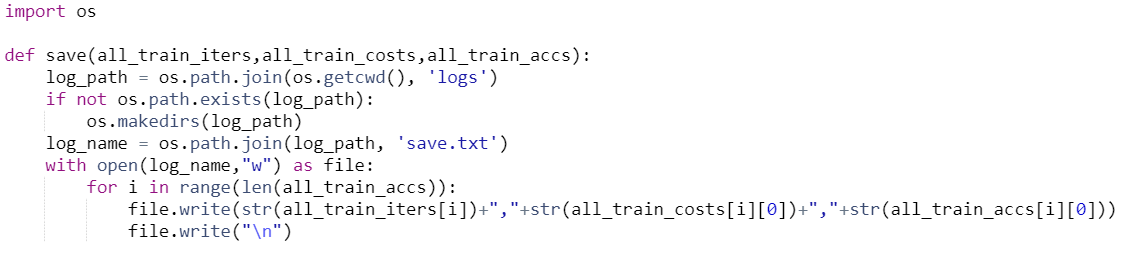
训练记录：



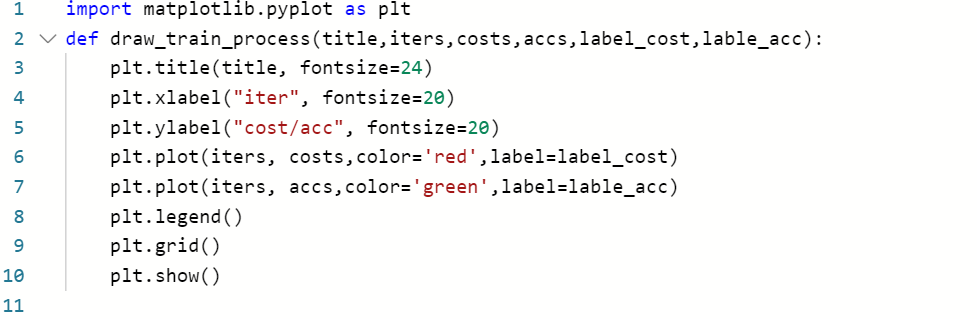
最后调用保存函数



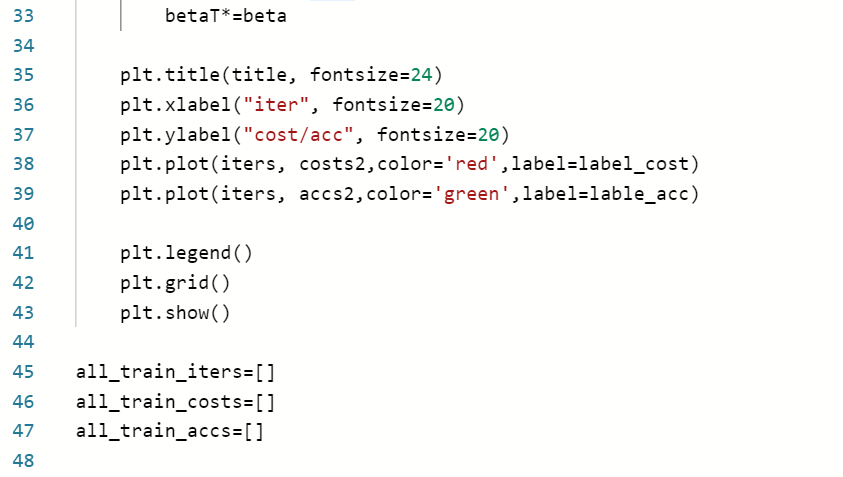
保存函数：



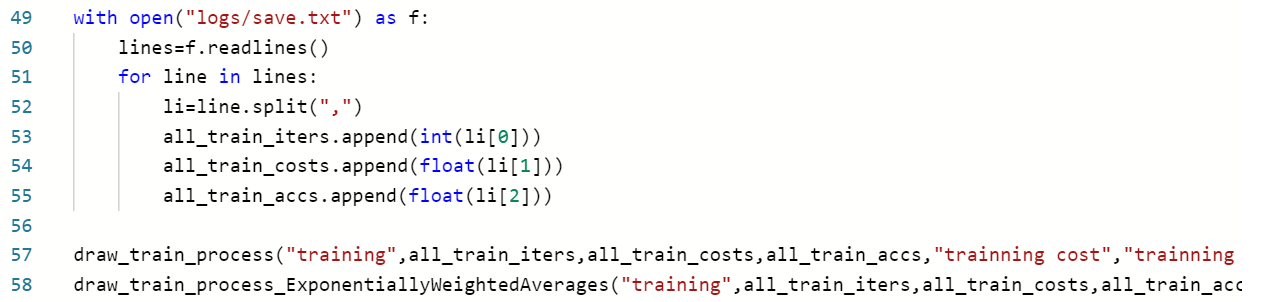
在jupyter界面的绘图函数：







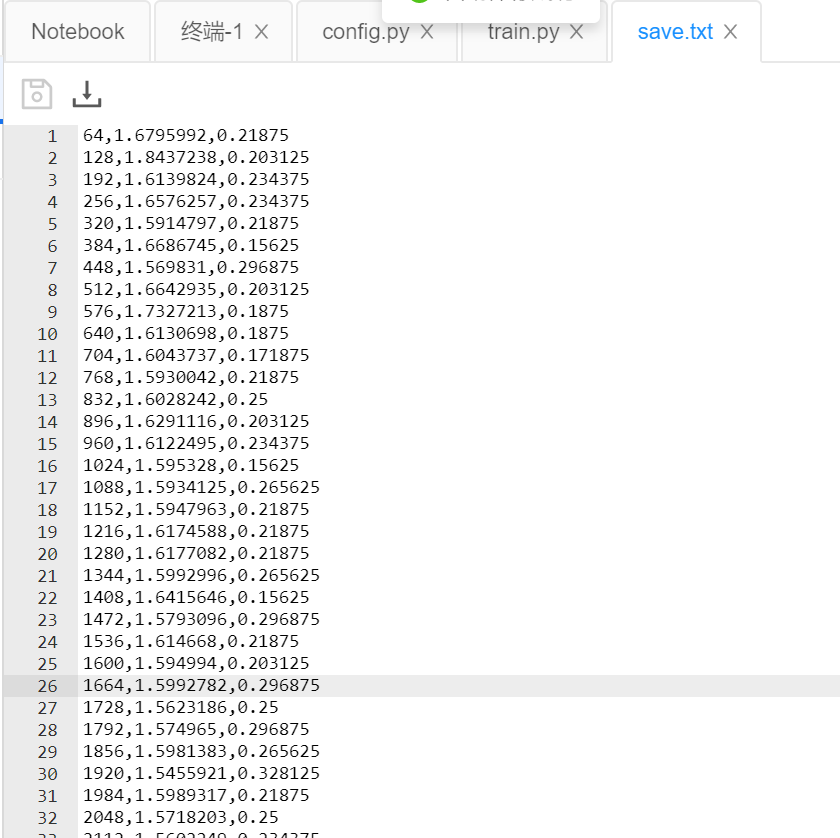
在jupyter界面的读取数据函数：

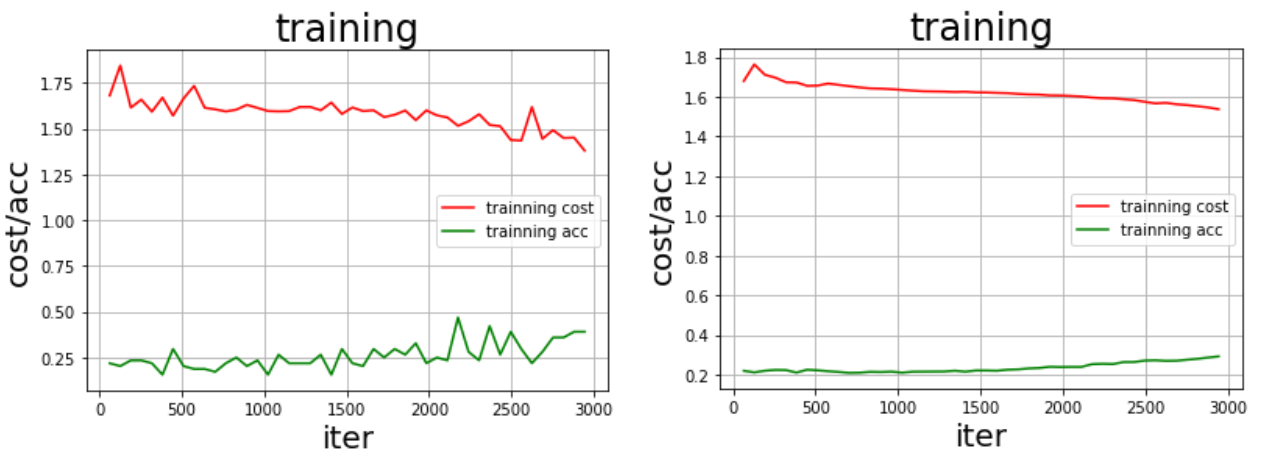


层次关系如下：



进行一个周期的测试，测试记录以及画图是否正常

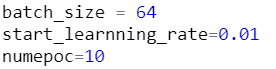


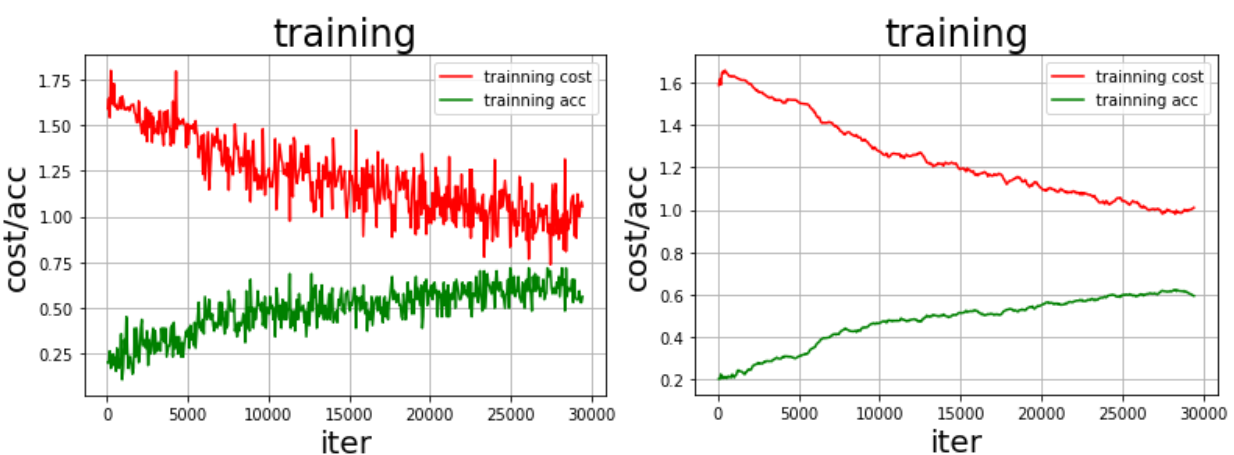


正常显示，其中右图为指数加权平均后的图像。

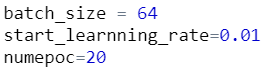
* 1. 训练及测试

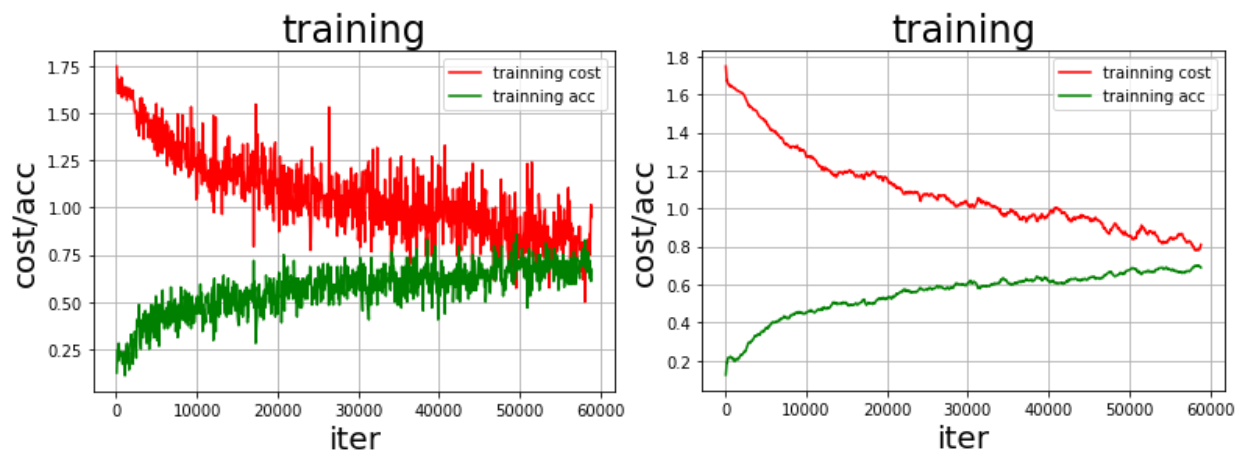
参数：



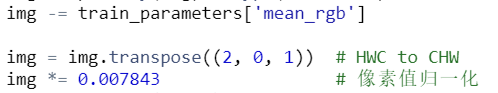


参数：

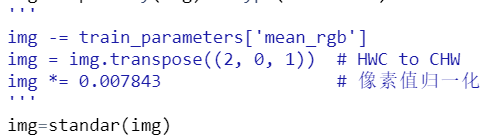


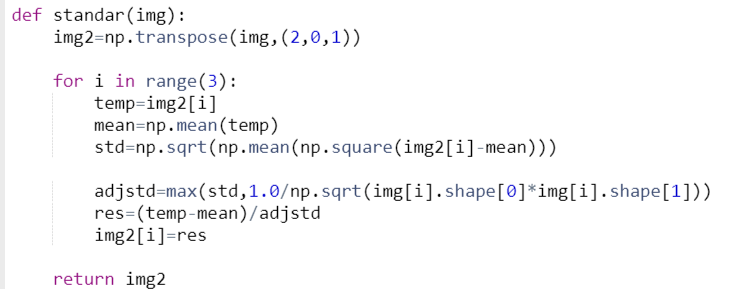


我改变了源程序的图像预处理方式，原方法为减去127后除以255

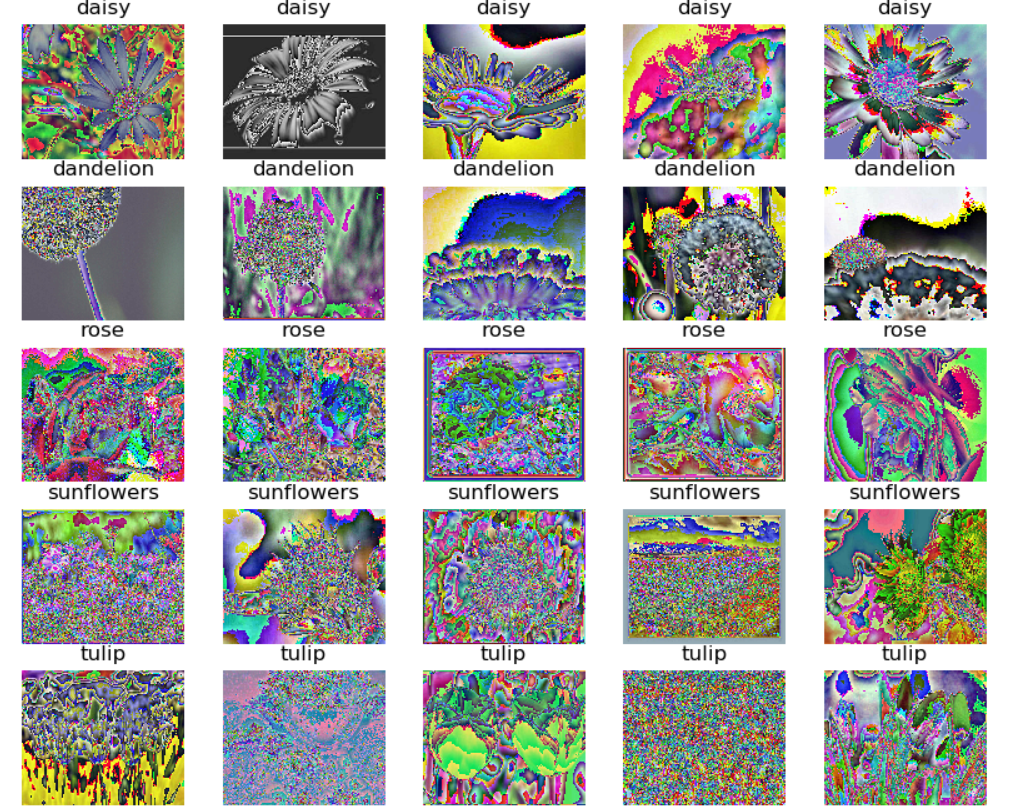


我采用了标准化的方法

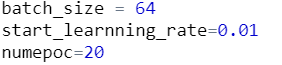


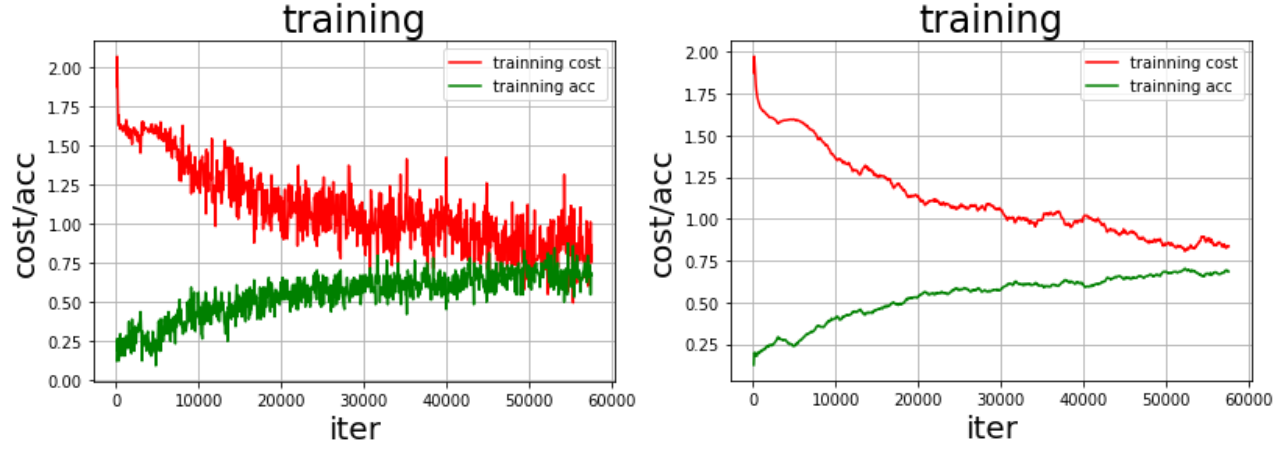


显示效果如下（注：此处图像为我下载后，在本地的python用相同的处理方式处理后在本地显示的，为了进行显示，在标准化后我成了255以进行显示，图片尽显扭曲之美）



参数：（非对比实验）



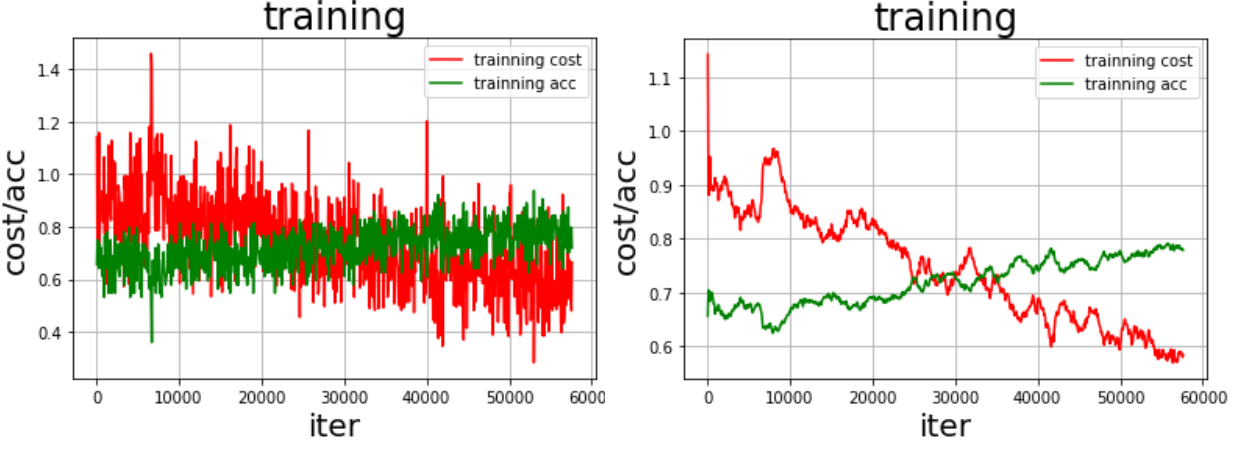


准确率接近0.75

测试继续训练，20~40个epoch



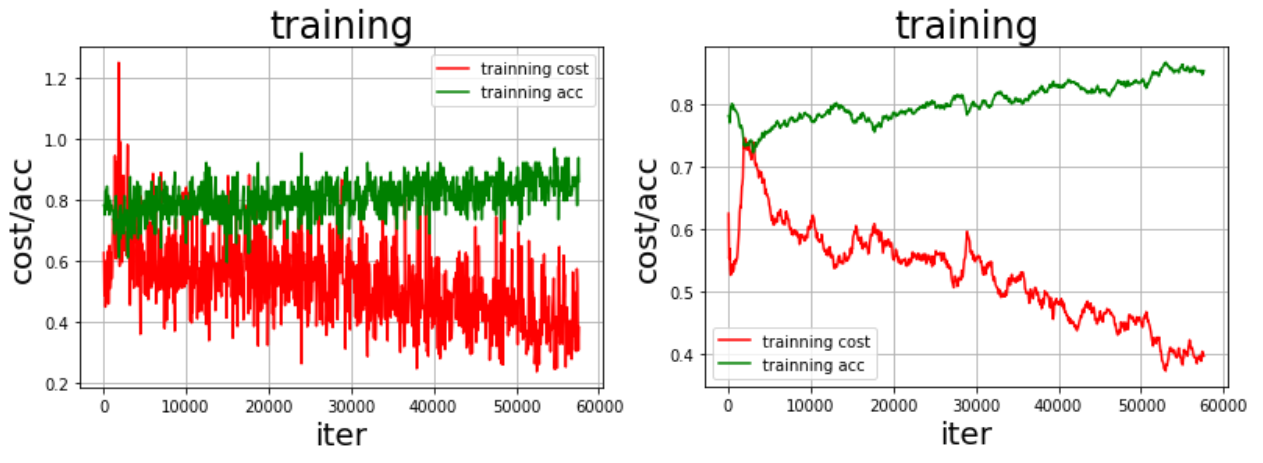
参数不变，和上面一样



验证集上的准确率是



周期40~60



验证集上的准确率是

