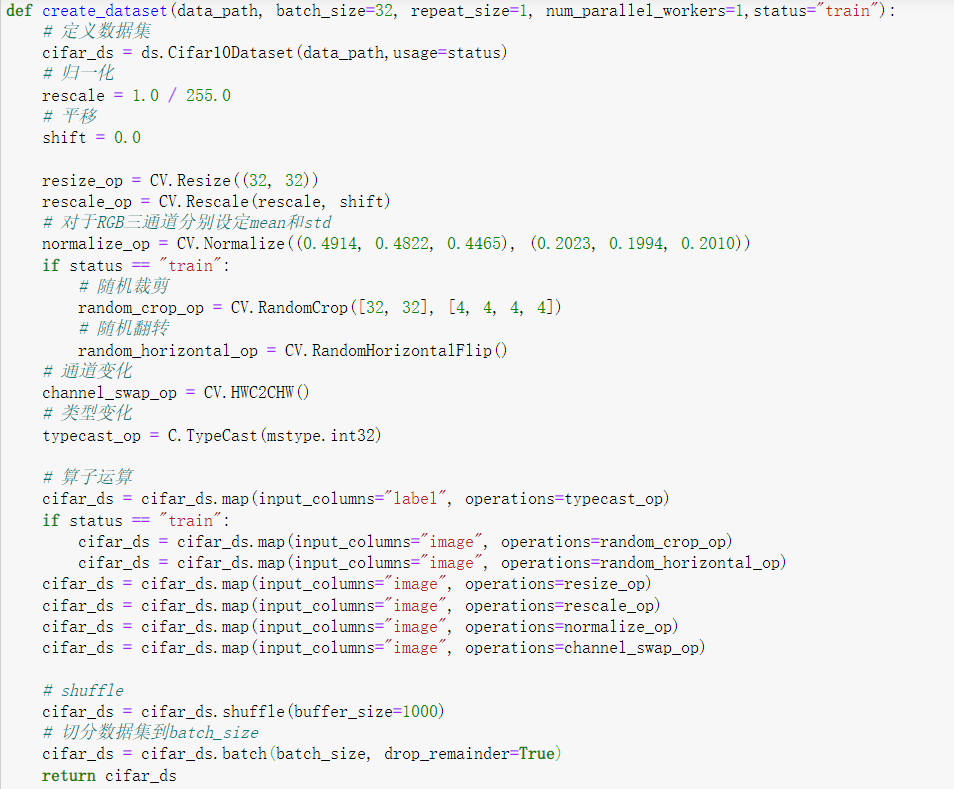
Mindspore架构训练使用的Cifar-10数据集为官方发布的二进制版本。

一、基于LeNet-5的分类

网络架构参考官方文档https://www.mindspore.cn/tutorial/training/zh-CN/r1.2/ quick\_start/quick\_start.html，数据集准备参考https://bbs.huaweicloud.com/blogs/ detail/212448。

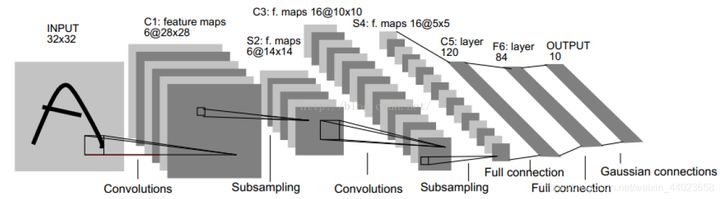
1.准备数据集：

Mindspore提供的Dataset.Cifar10Dataset()函数可以划分出训练集和测试集，这里对划分后的数据集再进行一定的处理。

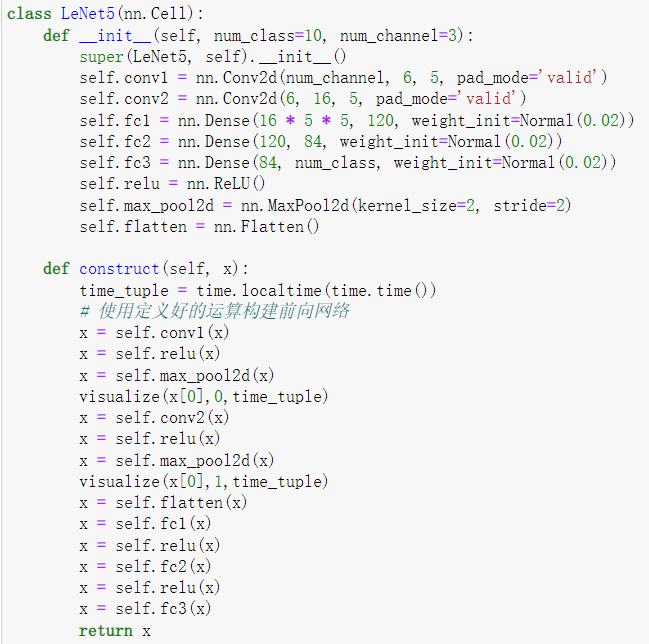


2.Lenet5网络架构及实现：

LeNet-5网络结构如下：



实现代码如下，调用了mindspore.nn库：



实例化：



定义损失函数：

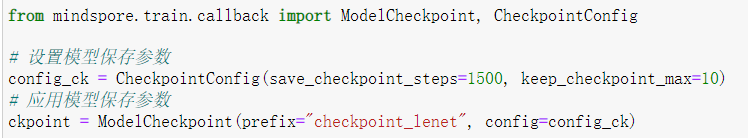


定义优化器:

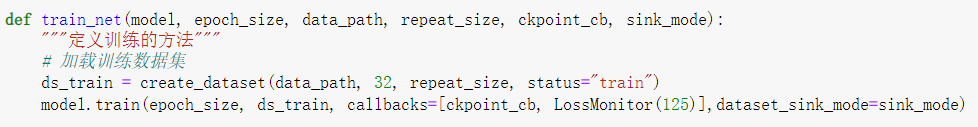


3.模型保存：

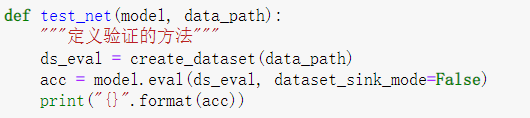
Mindspore可以调用mindspore.train.callback保存训练每一步的模型参数。



4.训练函数：

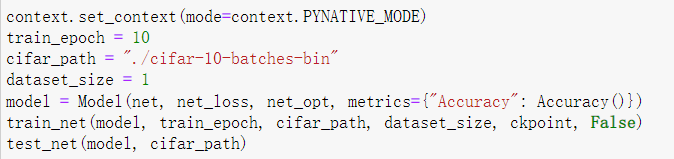


5.测试函数：



6.主函数：

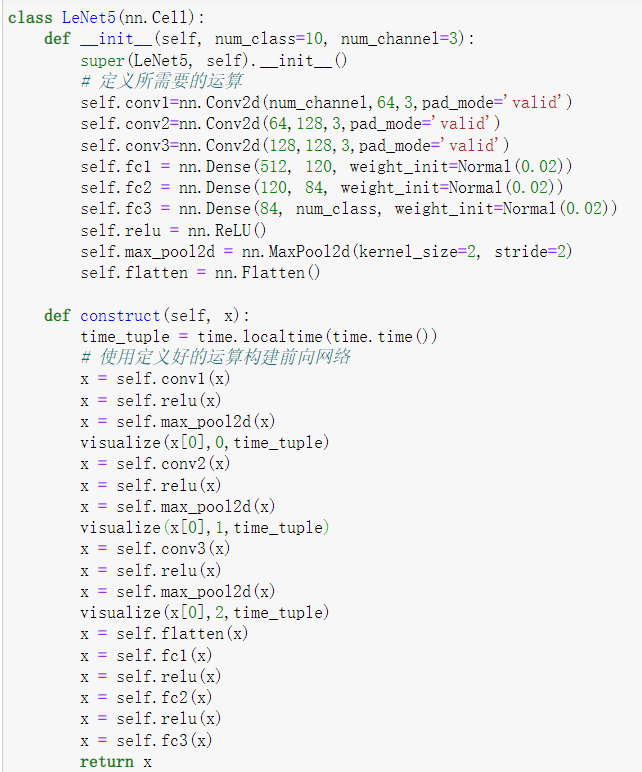
Mindspore中模型分为动态图模式和静态图模式，默认为静态图模式，但是加入可视化后会报错。为了实现可视化，调用context.set\_context将模型转换为动态图模式。



但是实验中发现动态图模式下训练时每个epoch的起始step计数有问题，暂时没有理清错误原因和解决方案。

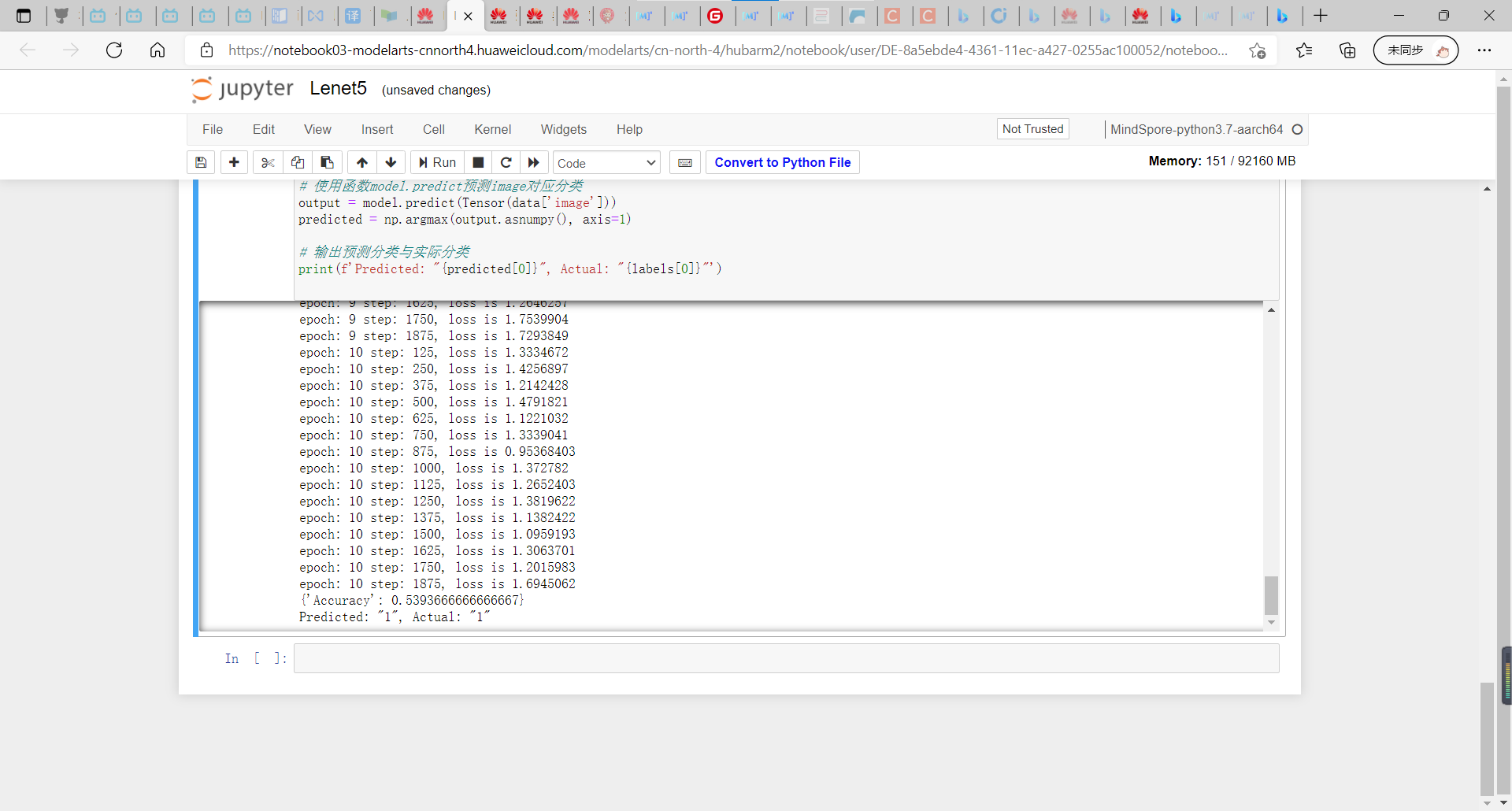
7. 改进网络：

由于LeNet-5训练效果不好，所以尝试了三层卷积并增加网络深度。

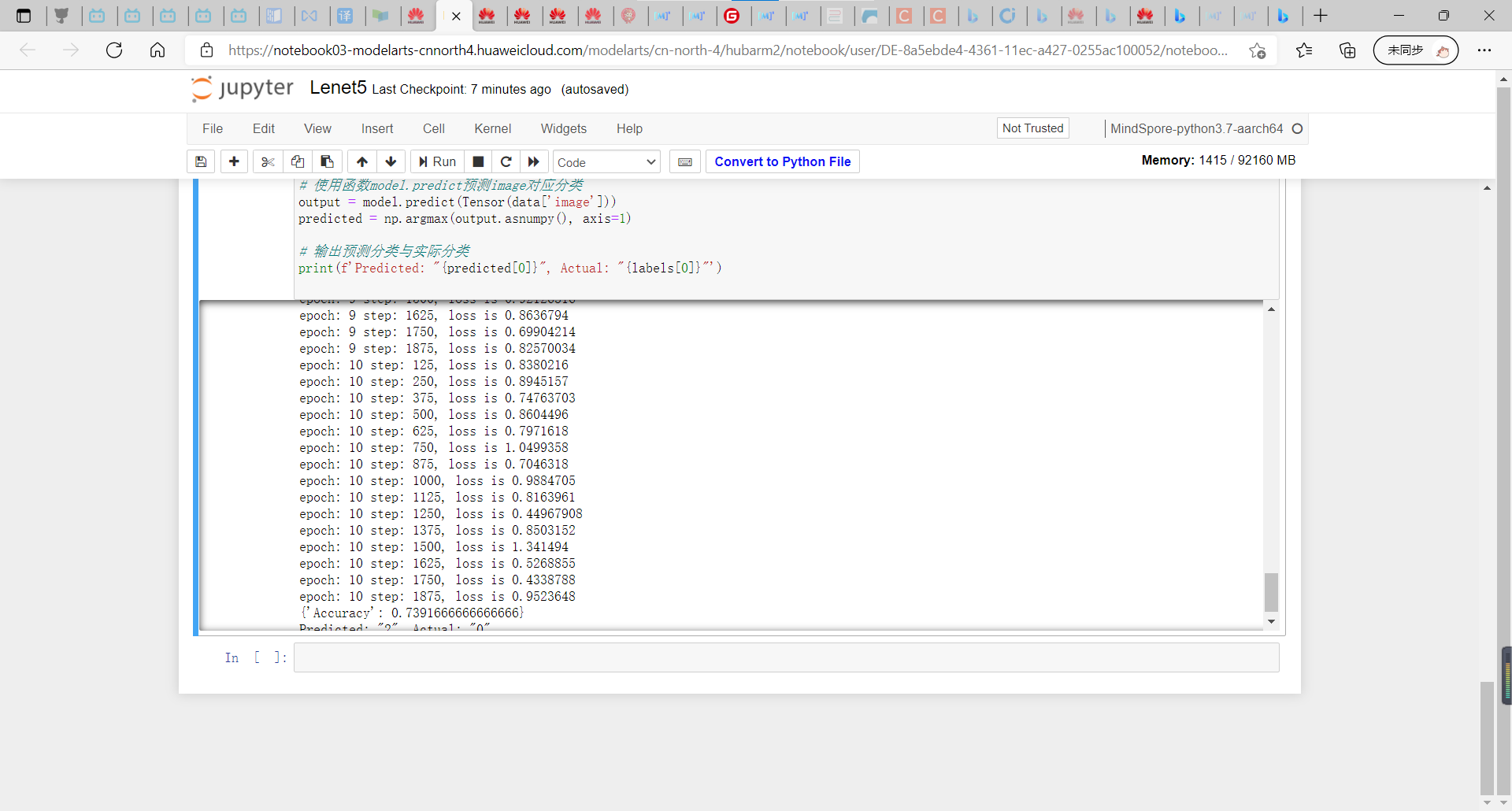


8.测试结果：

LeNet-5：

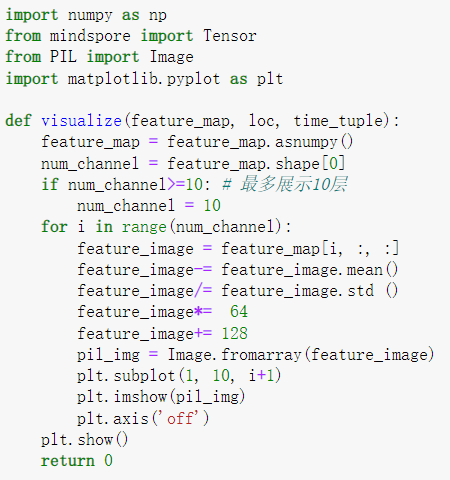


改进网络：



9. LeNet-5中间层可视化：

提取卷积后的特征图进行可视化，每次最多显示前10维的图片：



可视化结果举例：



可以看出LeNet-5在特征提取时第一层卷积后提取特征相对模糊但仍然可以大致分辨特征，第二层卷积后提取的特征几乎无法识别。因此推测LeNet-5训练表现不佳的原因包括数据集图片过于复杂且相对庞大因此网络无法基于提取特征做出有效的计算识别。

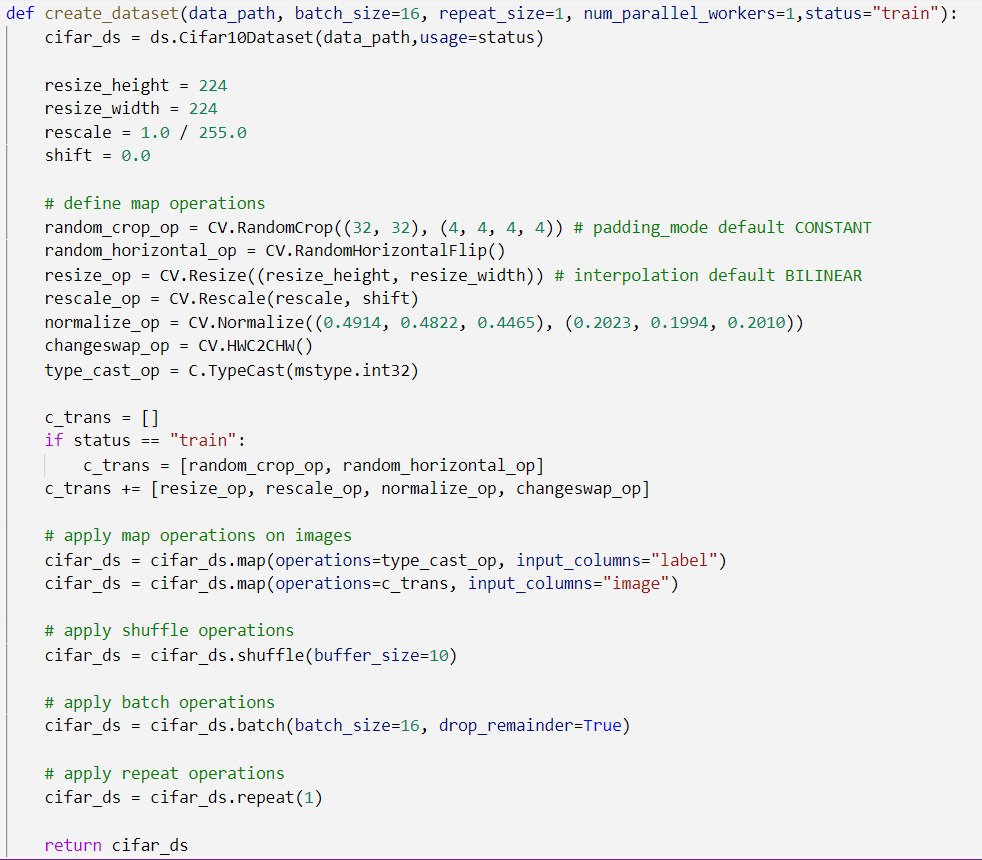
二、使用ResNet-50进行分类

由于LeNet-5及改进卷积网络识别准确率最大不超过80%，因此尝试了使用ResNet-50进行分类。

实现代码参考官方文档https://www.mindspore.cn/tutorial/training/zh-CN/r1.2/ advanced\_use/cv\_resnet50.html及其中提供的ResNet-50样例：https://gitee.com/ mindspore/docs/ tree/r1.2/tutorials/tutorial\_code/resnet。

1.准备数据集：

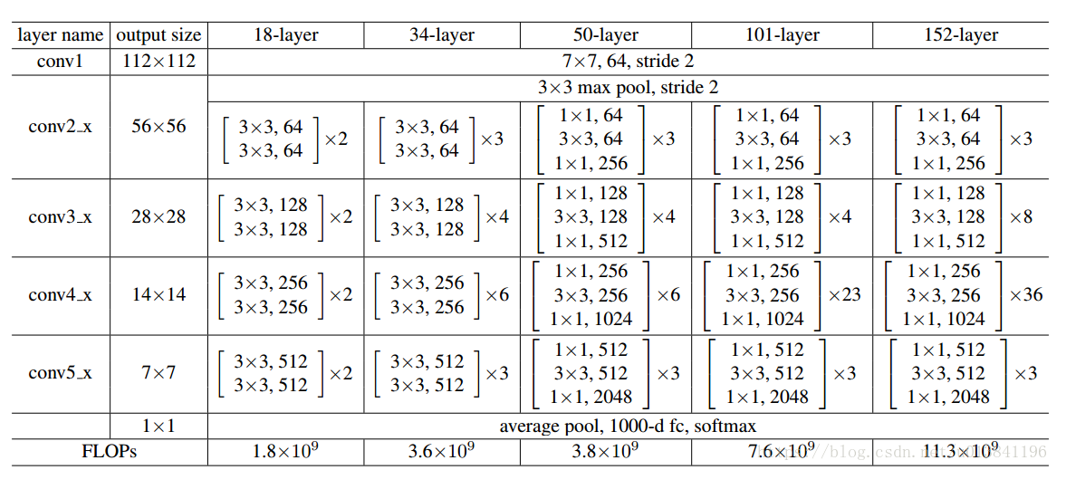
基于mindspore.dataset又进行了一定的数据处理。



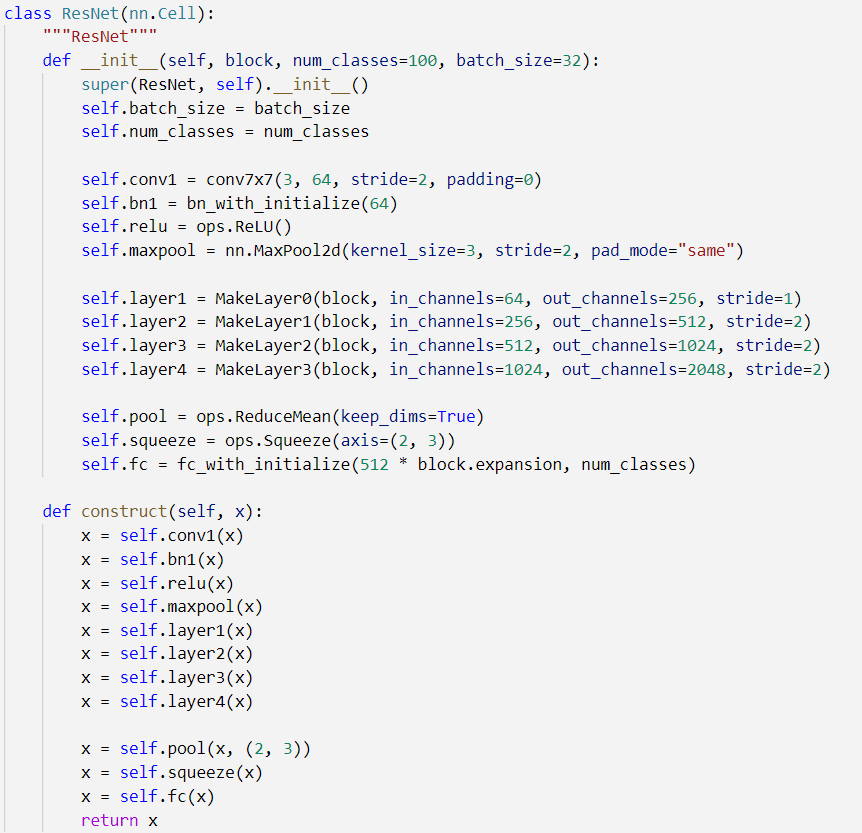
2.ResNet-50网络架构及实现：

ResNet网络引入了残差块的设计，使每层的神经网络学习上一次的输出残差，加快神经网络训练速度的同时提高训练的准确率。

ResNet-50网络结构如下：

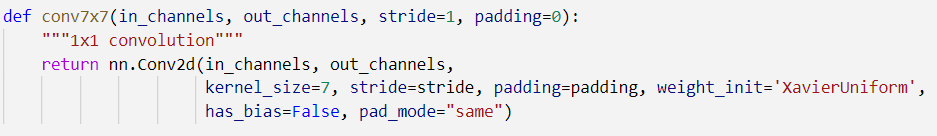


整体实现如下：

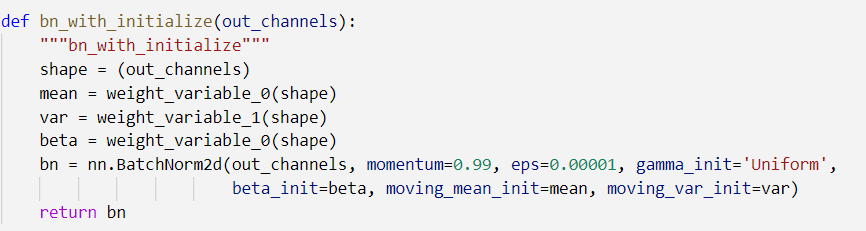


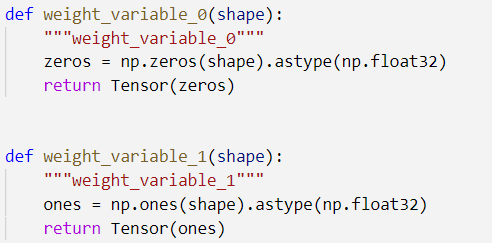
其中：

conv\_1层：

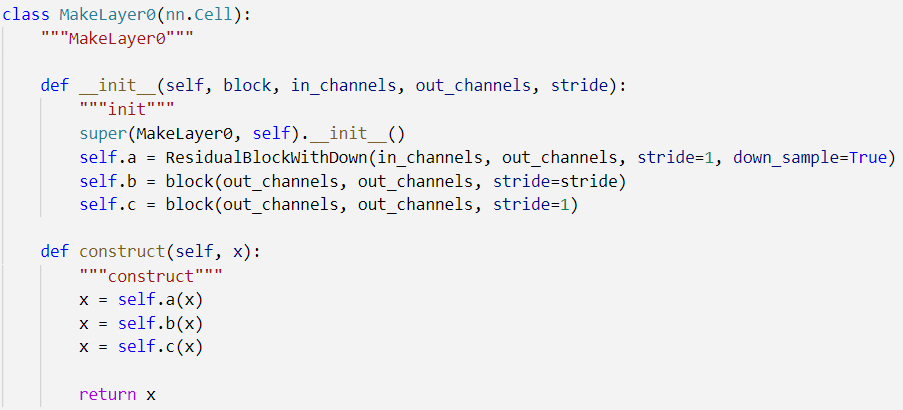


归一化：

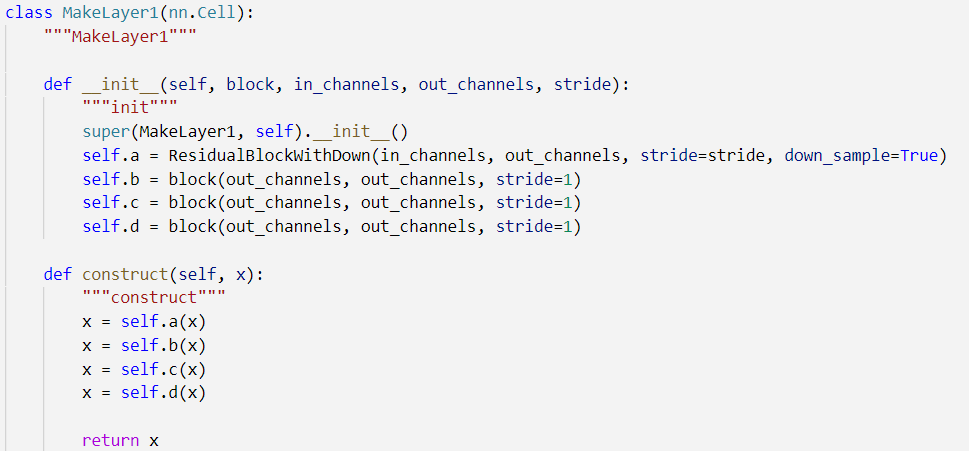




conv2\_x层：



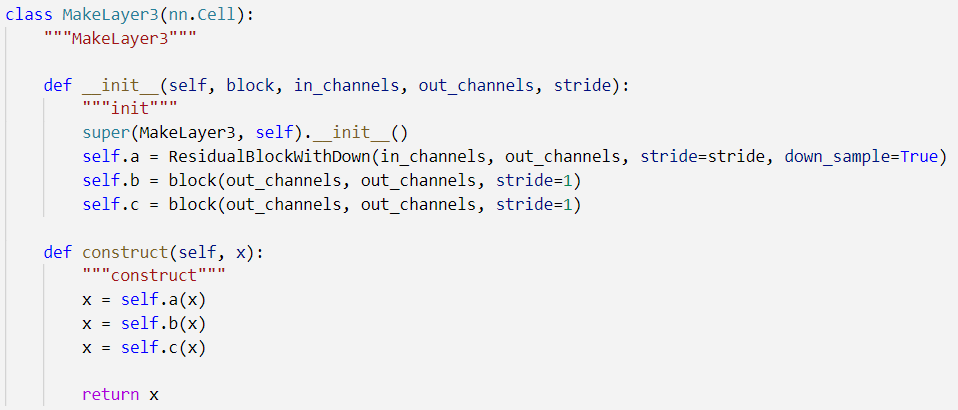
conv3\_x层：



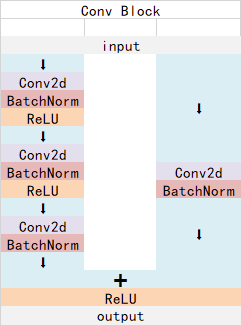
conv4\_x层：



conv5\_x层：

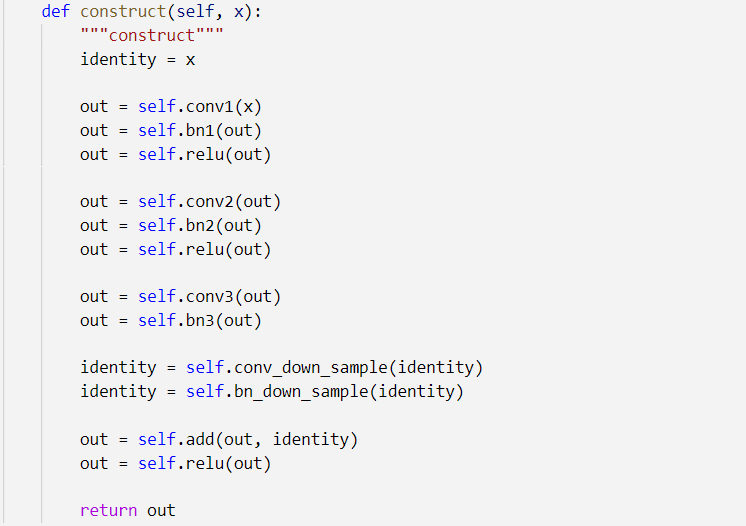


其中使用的残差块conv block（ResidualBlockWithDown类）的结构如下：



实现如下：





网络实例化：



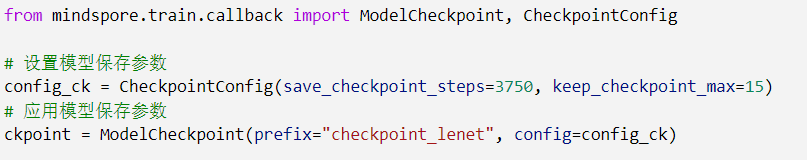
损失函数：



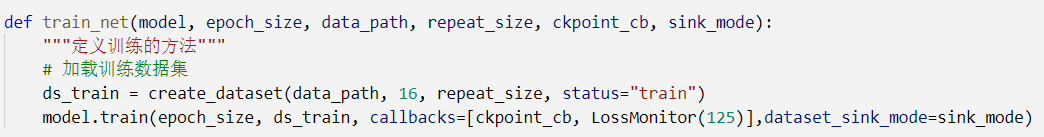
优化器：



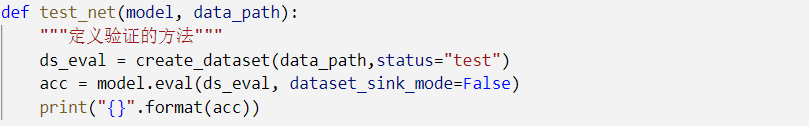
3.保存模型：



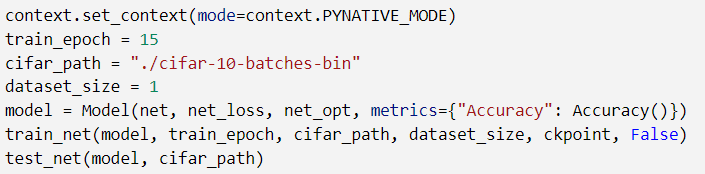
4.训练函数：



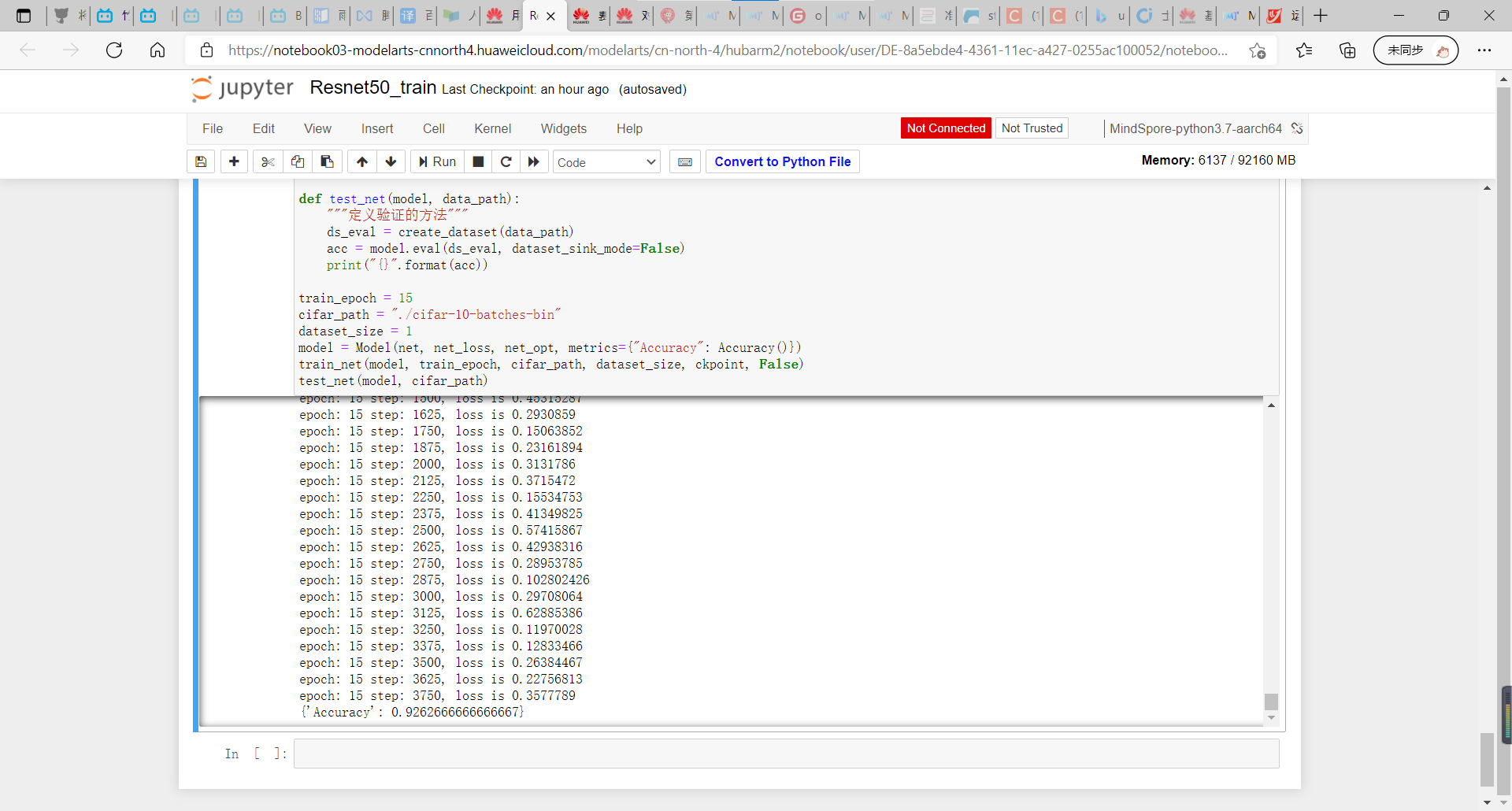
5.测试函数：



6.主函数：

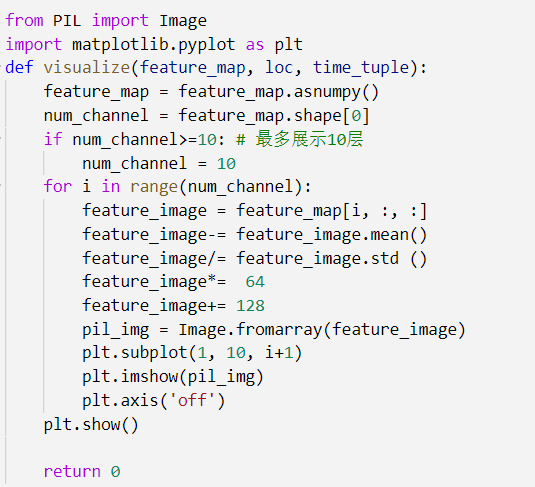


7.测试结果：

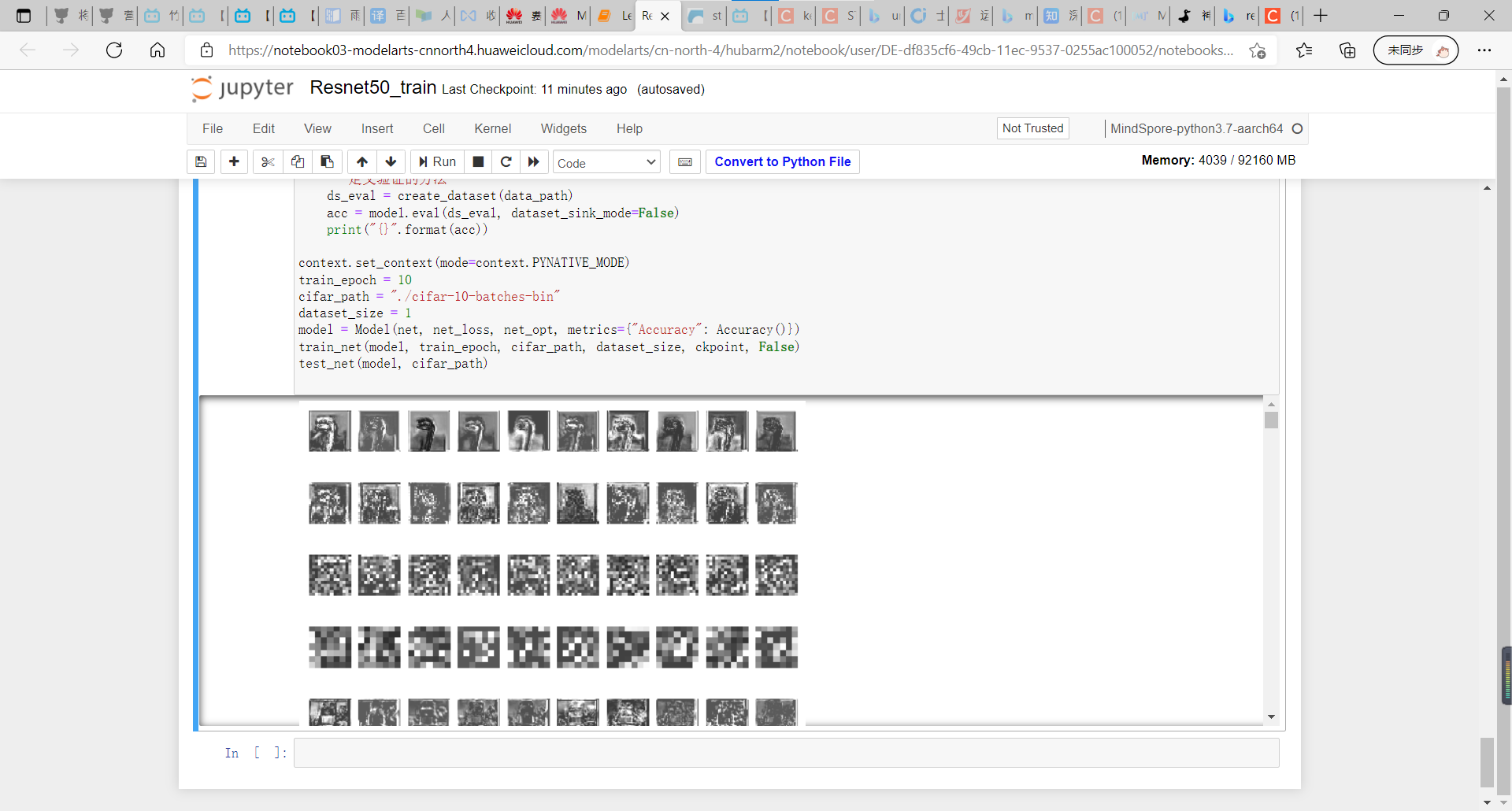


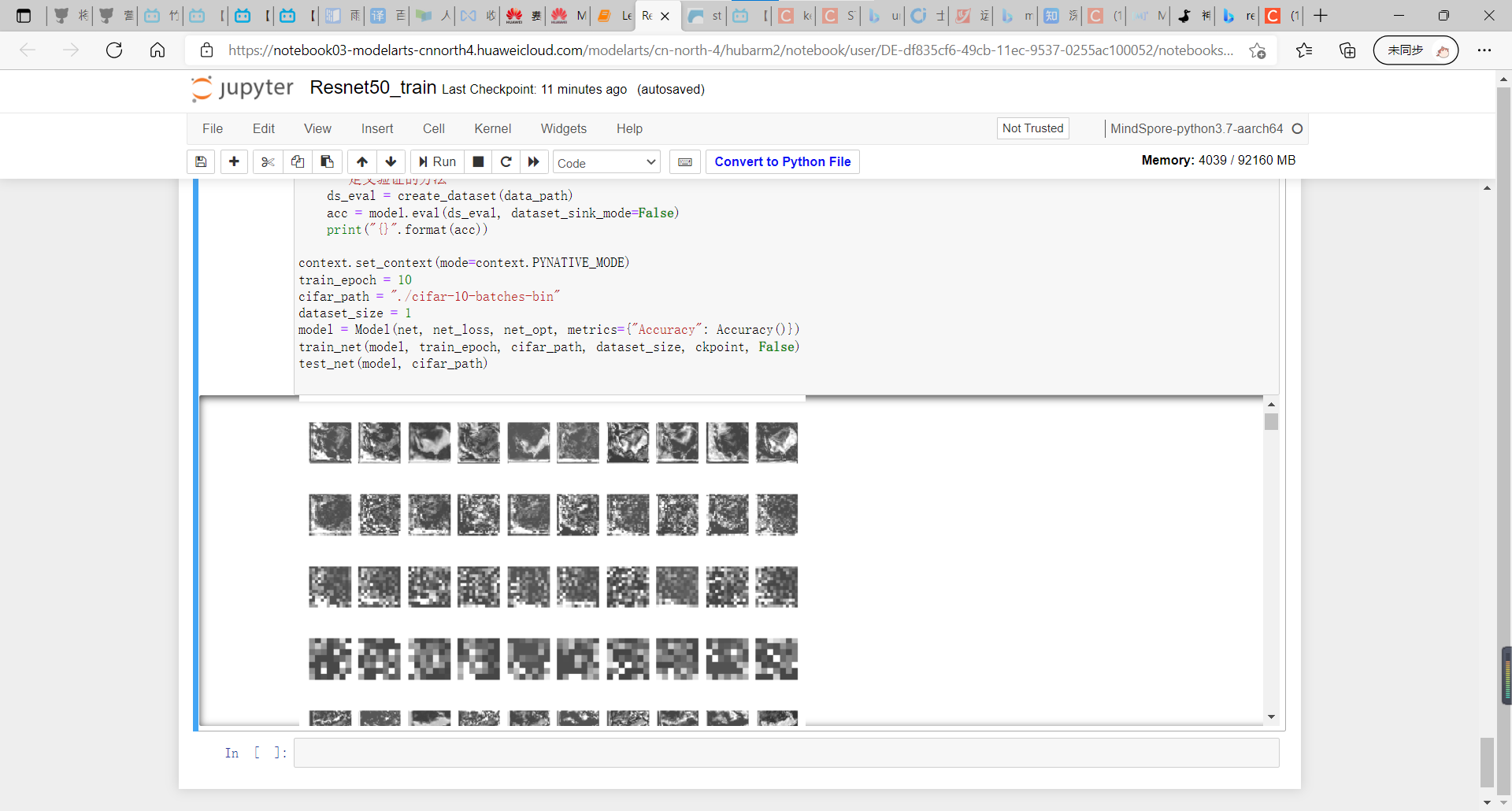
8.中间层可视化：

提取conv2\_x，conv3\_x，conv4\_x，conv5\_x的特征图，每层特征图只提取前10维。



可视化结果举例：



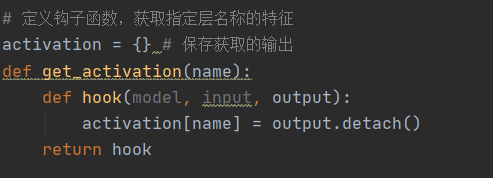


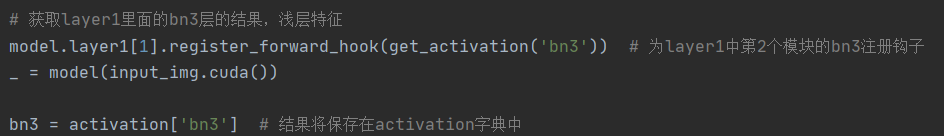
此时高层的可视化结果已经非常抽象，难以对应分辨特征细节。

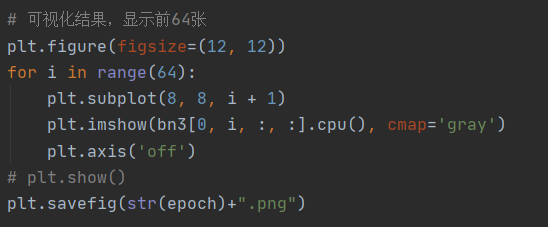
9.Pytorch可视化：

由于从训练网络提取出的特征图的打印过于细节难以有效辨识，在mindspore模块的不断报错后改用了pytorch进行可视化，便于对应原图像进行分辨。

代码实现：







结果举例：

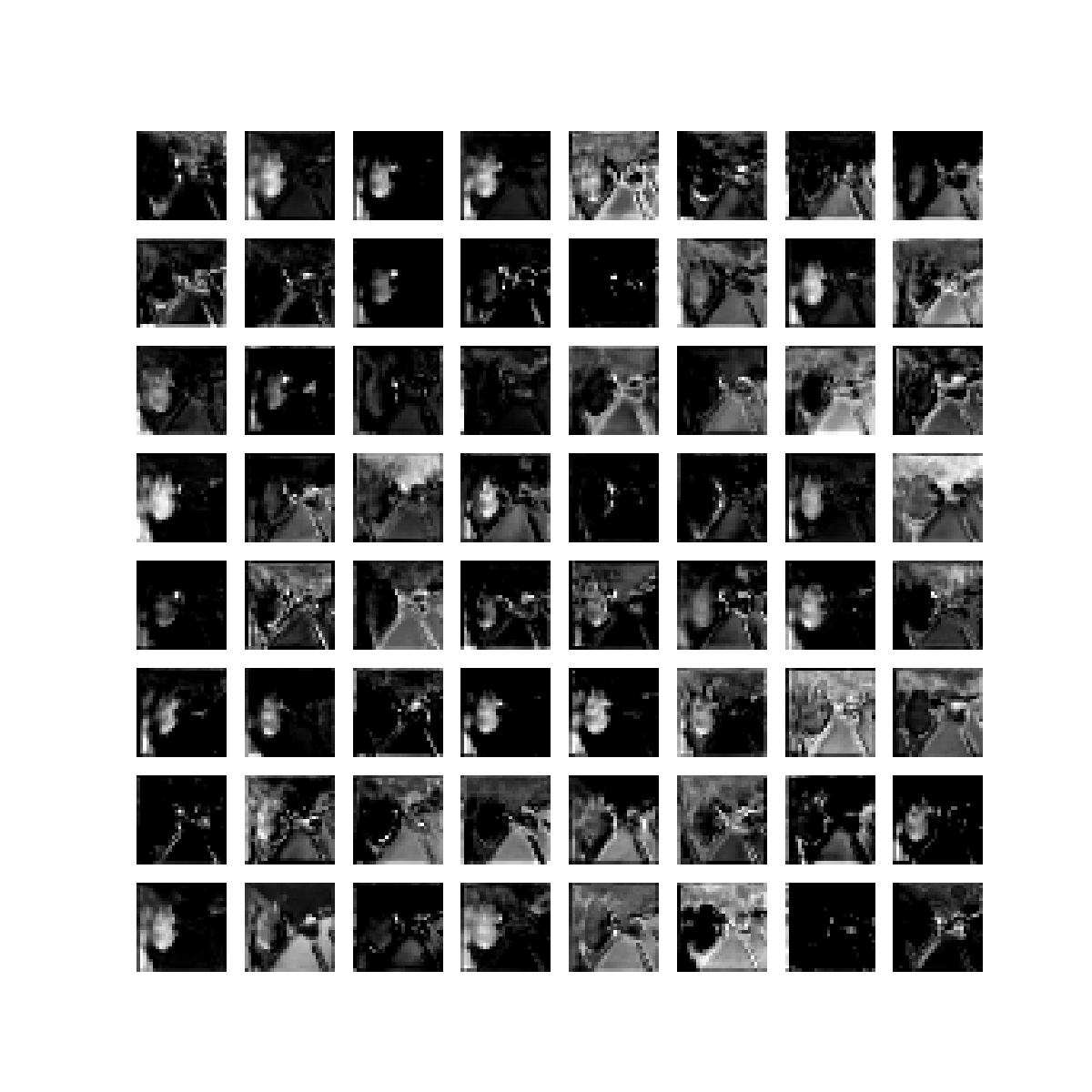
原图：



conv\_2x：



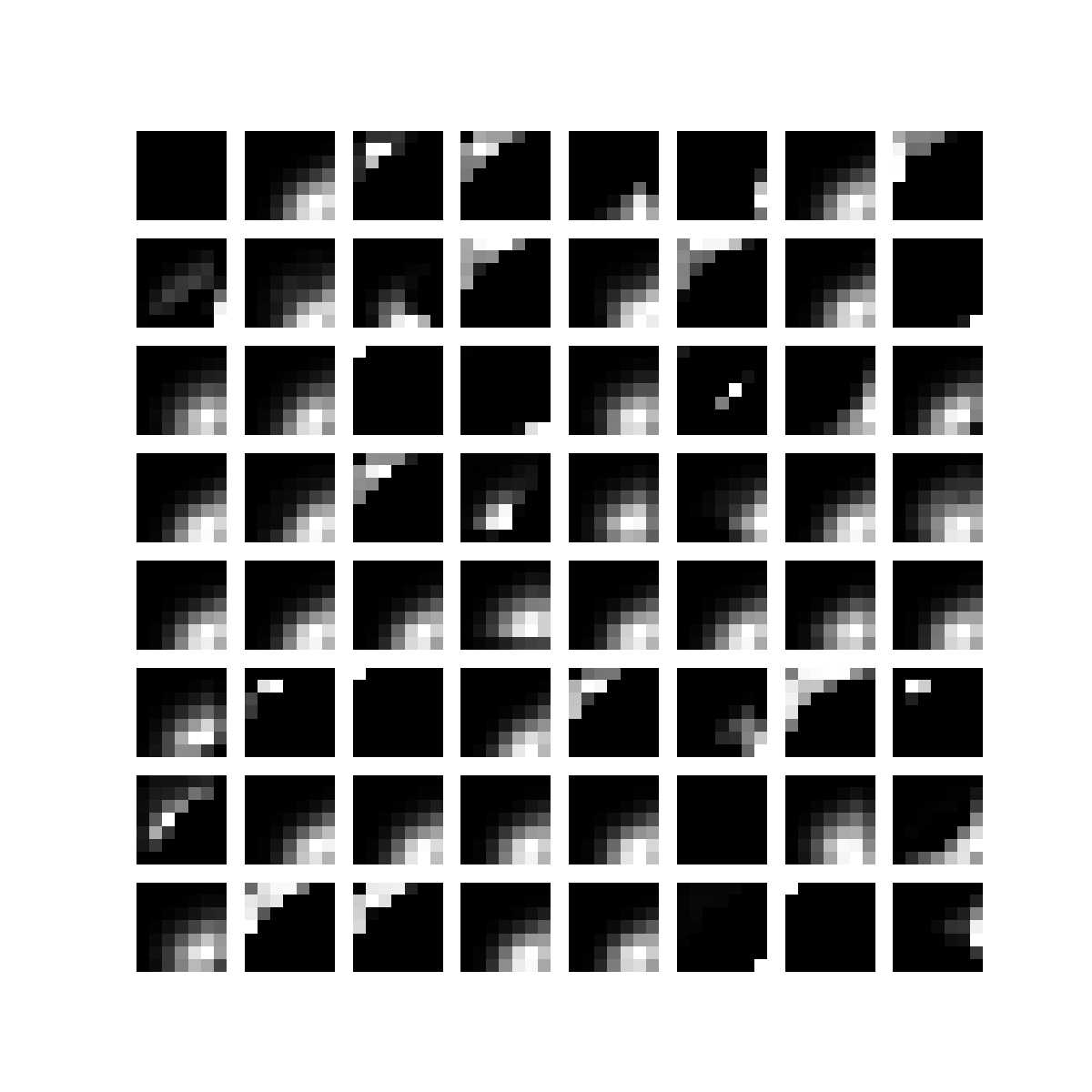
conv\_3x：



conv4\_x：



conv5\_x：



显然，越高的网络层提取到的特征不止会更加抽象，也更加关注输入图像的局部特征。