****问题汇总****

笔记整理人：天国之影（2019年6月24日）

# 说明

1. pytorch使用0.4版本的官方文档，安装的是1.0版本，其中某些函数的区别，同学在问的时候我会整理好写成Word文档。
2. 我会把每天学生经常问的问题整理成Word文档，先看Word文档中有没有解决方案，如果没有再在群里问。整理好的文档会在本篇文章中定期更新，方便大家查阅资料就能解决问题，提高学习效率。
3. 项目实战最后一周会进行学习，大家先学基础API，不然项目也看不懂，打牢基础是最重要的。
4. 官方文档有可能有误，具体报错问题可以在群里问，然后会整理到文档中。
5. 每天同一个问题大于2次，我就会整理到文档中，其他小概率问题忽略。

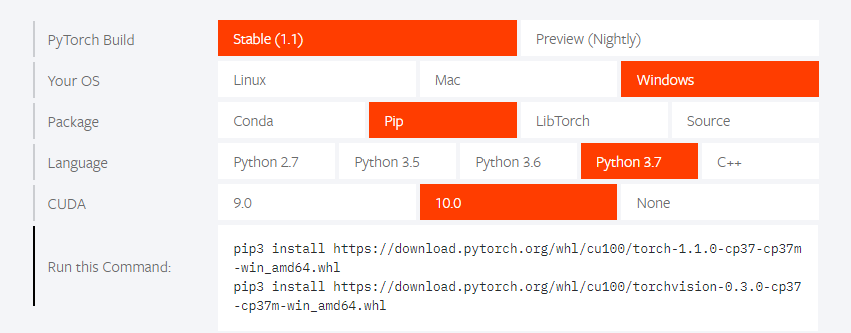
# 问题1：环境配置问题

请参考《1.window+pytorch配置》文档，只要按步骤来就不会报错，烦请仔细看。

在Pycharm中安装Pytorch

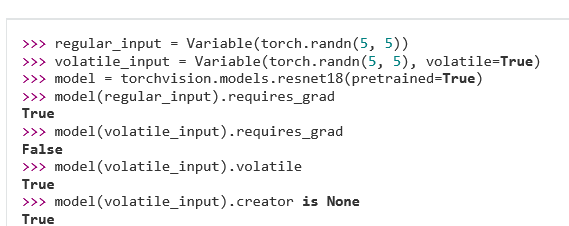
先访问网站：<https://pytorch.org/>

然后根据系统配置，选择下载方式



最后在Pycharm的Terminal运行命令即可。

# 问题2：自动求导机制问题



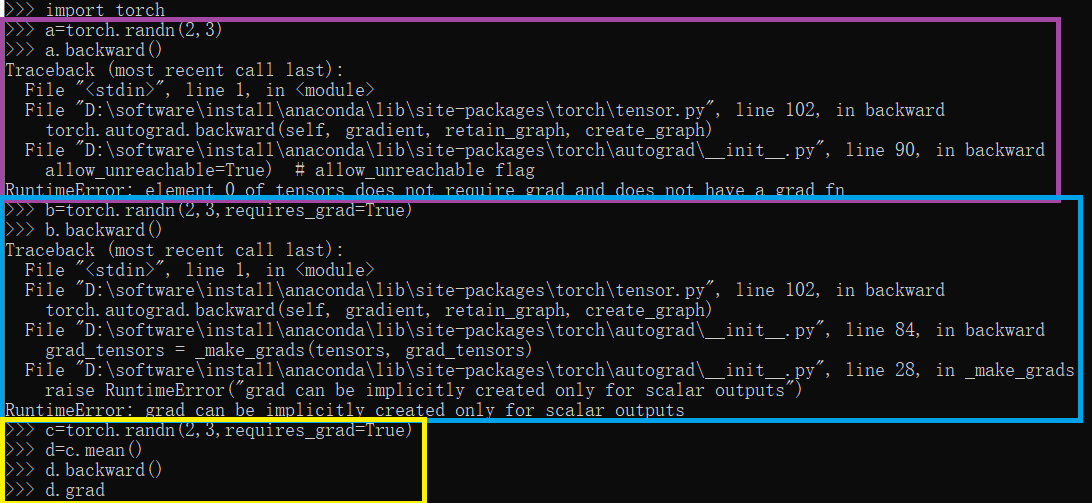
运行会报错，显示维度不一致：因为model需要四维张量（batch,channel,height,width）,而文档中是二维张量，所以报错。

注意：去掉variable函数，这是0.4版本的函数，我们安装的是1.0版本

|  |
| --- |
| regular\_input=torch.randn(1,3,224,224)#默认是True  volatile\_input=torch.randn(1,3,224,224)  Model=torchvision.models.resnet18(pretrained=True)  Model(regular\_input).requires\_grad  with torch.no\_grad():  Model(volatile\_input).requires\_grad |

总结：requires\_grad=Fasle时不需要更新梯度，适用于冻结某些层的梯度；volatile=True相当于requires\_grad=False，适用于推断阶段，不需要反向传播。这个现在已经取消了，使用with torch.no\_grad()来替代维度不一致问题是函数所需维度和输入维度不一致，这个不是难点，只需根据需求更改即可，比如这个例子model需要[batch,channel,height,width]=[batch,3,224,224]，后面讲网络结构会说到。

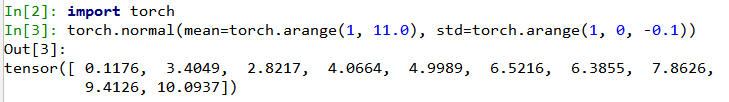
# 问题3：有关autograd和backward的问题

****

（1）紫框部分报错是因为默认创建tensor的requires\_grad=False,而backward需要requires\_grad=True;

（2）蓝框部分报错是因为backward需要是scalar的数据，也就是需要的是标量类型的张量，简单说dim=0的张量就标量，dim=1的张量就是向量，dim=2的张量就是二维矩阵。

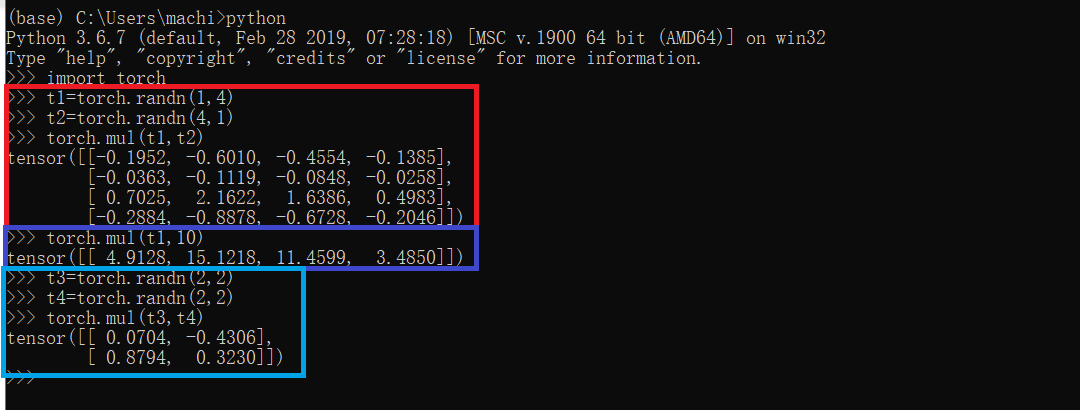
# 问题4：torch.normal参数问题



（1）参数问题，0.4版本是means,1.0版本是mean

（2）数据类型问题，mean和std需要float(浮点型)类型

# 问题5：torch.mul()函数



torch.mul(a, b)是矩阵a和b**对应位相乘**：

**有三种情况：**

（1）一个tensor乘一个常数

（2）两个shape完全一致的tensor

（3）如果形状维度不一致，可以触发boradcast机制的方法：

比如t1=torch.randn(1,4)，t2=torch.randn(4,1)可以触发boradcat机制

**什么是boradcat机制？**

参考资料：<https://blog.csdn.net/fendyu/article/details/79955154>

Boradcat总结起来三句话：

（1）如果两个数组在维度的数量上有差异，那么维度较少的数组的形状就会被用1填充在它的前导(左)边。

（2）如果两个数组的形状在任何维度上都不匹配，但等于1，那么在这个维度中，形状为1的数组将被拉伸以匹配另一个形状。

（3）如果在任何维度上，大小都不一致，且两者都不等于1，就会出现错误。

# 问题6：tensorboadx可视化安装问题

**注意事项：**在可视化部分需要安装tensorboadx，但是tensorboardx依赖于tensorflow的tensorboad，使用的时候先安装tensorflow和tensorboard，具体安装步骤如下：

（1）pip install tesnsorflow==1.2(tensorflow版本太高可能会有问题）

（2）pip install tensorboard

（3）pip install tensorboardx

安装好之后pytorch就可以使用tensorboardx可视化了。

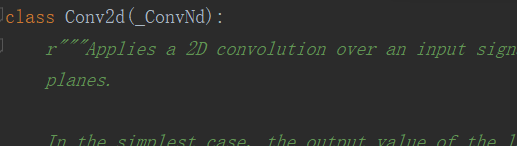
# 问题7：torch.all()函数

torch.all()是判断两个张量各个对应位置的值是否相等，有一个不相等就是0，全相等返回1。

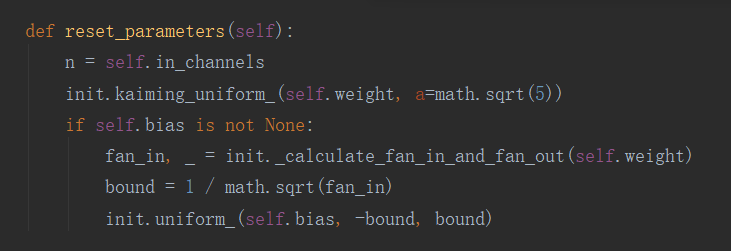
torch.all()可以参考numpy.all()这个函数，两个意义是一样的。

# 问题8：pytorch中默认的网络权重初始化方法是什么?其内置初始化方法中对relu激活函数的所有网络中一定是默认权重初始化方法最佳吗?

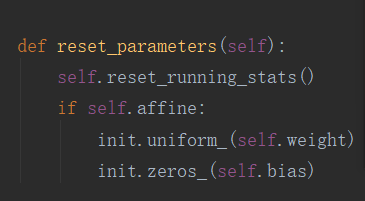
Conv2d继承自\_ConvNd，在\_ConvNd中，可以看到默认参数就是进行初始化



con2d的默认初始化方式：



BN继承（\_BatchNorm）,默认初始化方式：



# 问题9：torch.no\_grad()和eval()的区别？

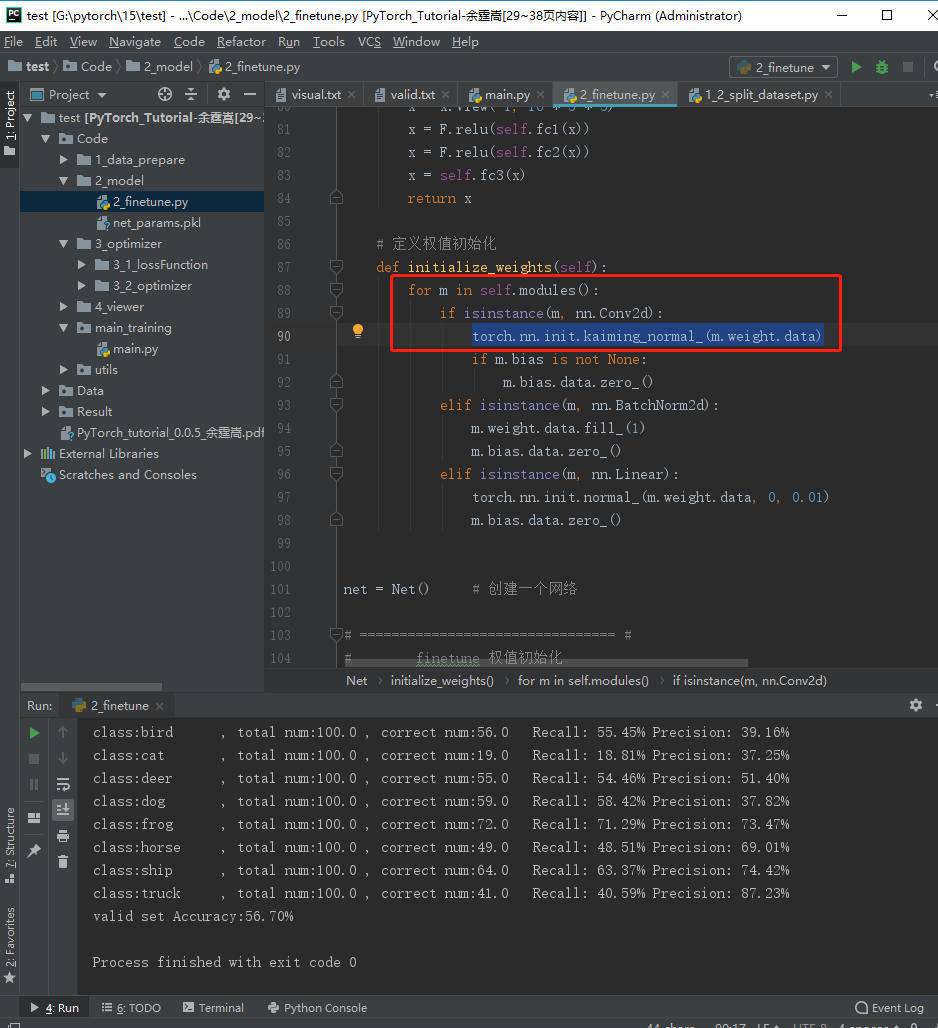
model.eval()：所有的图层处于评估模式，这样，batchnorm或dropout图层将在eval模型而不是训练模式下工作。

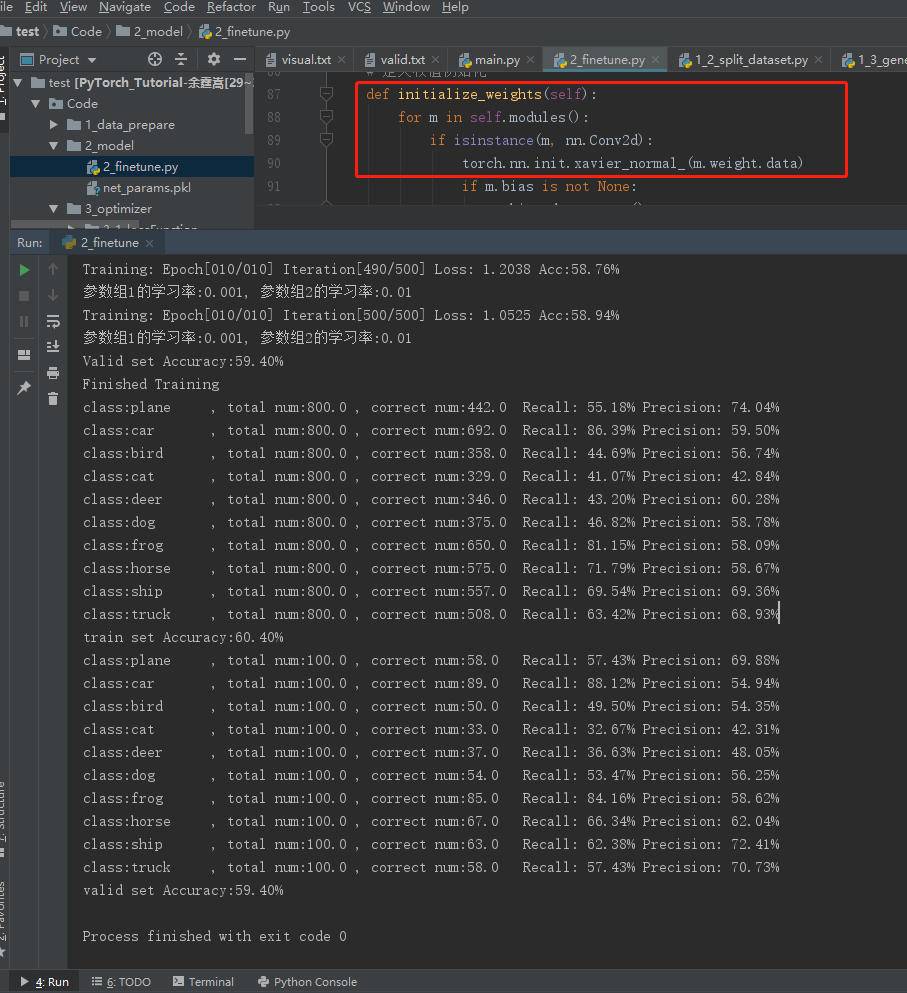
torch.no\_grad()：影响autograd引擎并停用它，将无法使用backprop。

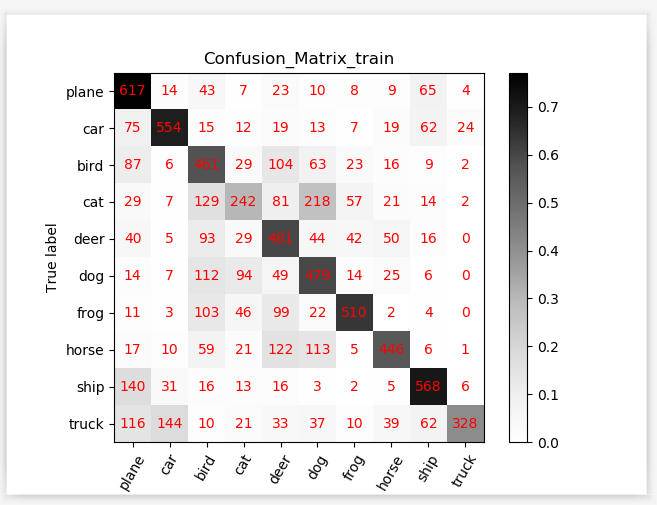
torch.no\_grad()主要在测试的时候减少内存和现存的占用，model.eval()主要是因为BN和dropout，如果没有这两个，可以不设置成model.eval()

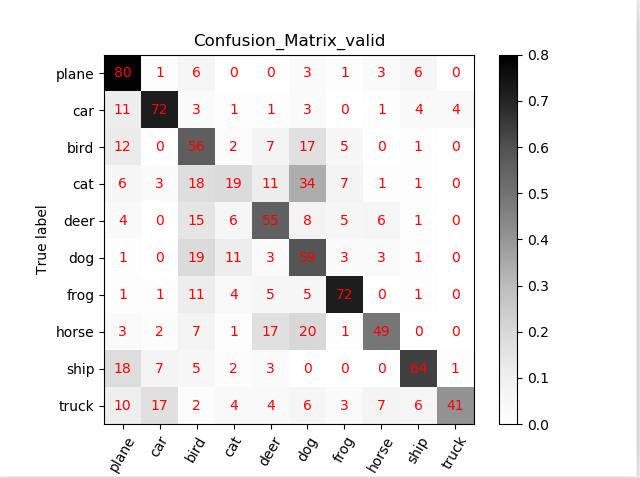
# 问题10：想问下3.1作业2在finetune的时候，为什么用xavier和kaiming的正态分布初始化之后训练，最后准确率都在50%左右呢？可是出来的混淆矩阵又是还可以的。

作业2：比较初始化方法和finetune方法收敛速度和精度







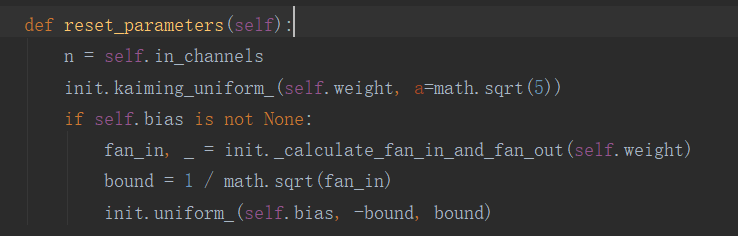


解答：

（1）因为本例子只是让大家了解方法流程，所以使用的最简单的Lenet网络，而cifar10是比较复杂的场景，简单网络模型不足以拟合复杂场景，所以可以看到训练数据集只有50%

（2）初始化方法，我们一般推荐默认初始化方法或者Finetune的方法，自己设计初始化方法往往达不到比较好的效果。

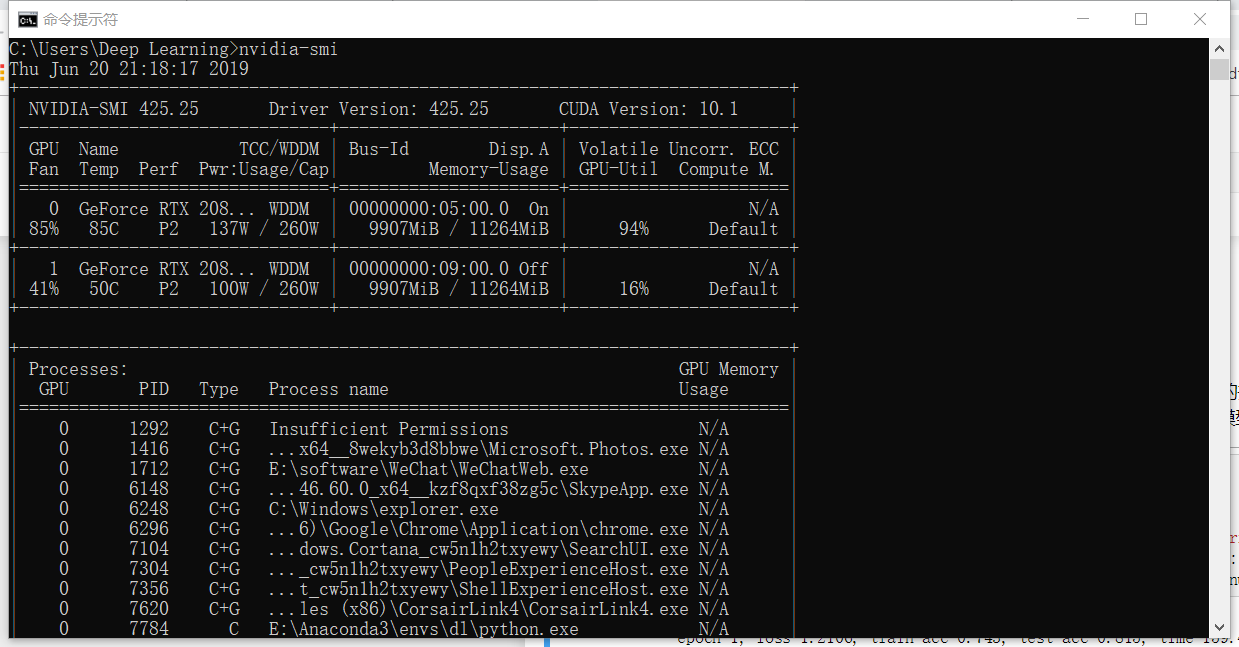
（3）默认conv初始化方法继承\_ConvNd：



其他默认初始化方法可以自行查看。

（4）测试效果50%~60%左右，所描述的混淆矩阵也是50%~60%左右。

# 问题11:请问两块gpu使用量不同怎么解决？



推荐这篇博客，讲的比较详细：

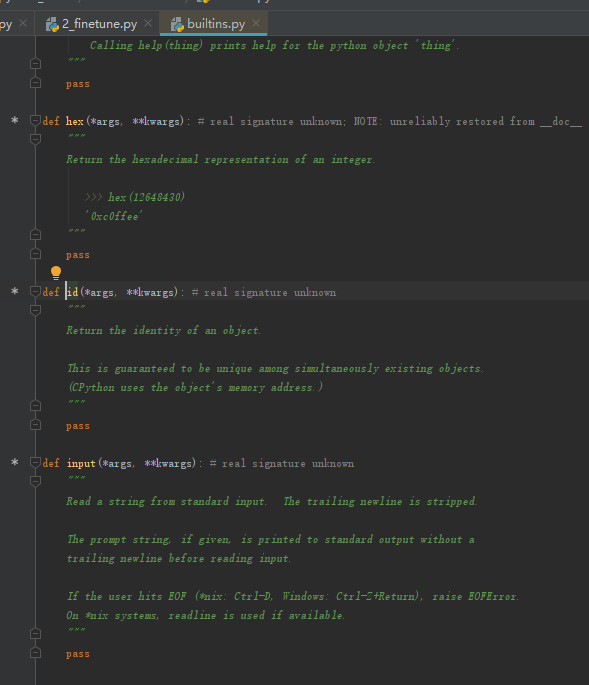
<https://www.cnblogs.com/ranjiewen/p/10113532.html>

# 问题12：为什么torch.nn.init的初始化函数都在后面加了下划线?有什么含义吗?

https://uploader.shimo.im/f/iTIvp7afRP0c3rne.png!thumbnail

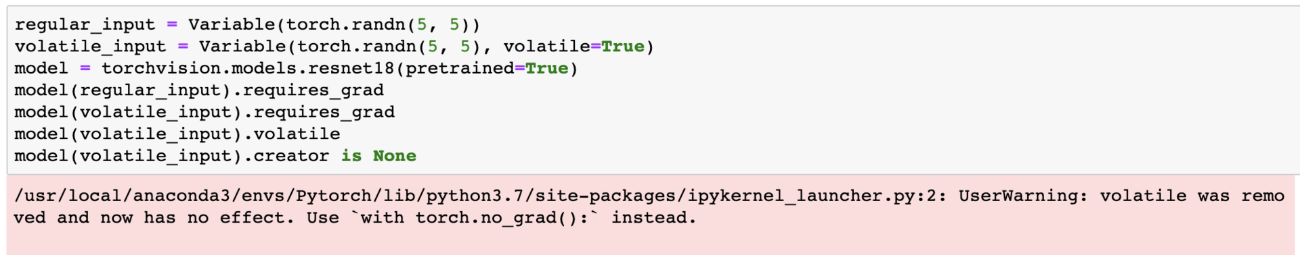
in-place operation在pytorch中是指改变一个tensor的值的时候，不经过复制操作，而是直接在原来的内存上改变它的值。可以把它成为原地操作符。在pytorch中经常加后缀“\_”来代表原地in-place operation。normal和normal\_功能一样。

# 问题13：有些py文件里的函数并没有实际定义，全都是注释，这是什么意思？



这些只是提供一些接口，python封装了c、c++,具体实现是c/c++中。

# 问题14：请问课程中用的是什么时候的官方文档？volatile在新版本中应该怎样被替换掉呢？



（1）目前使用的0.4版本，因为1.0版本没有中文文档，考虑大家阅读英文文档可能不太方便，所以使用的0.4版本（2）使用with torch.no\_grad替代了volatile

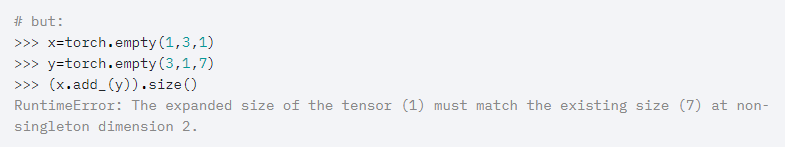
# 问题15：3.2作业在同一个模型上比较不同loss函数精度的问题

在余大神给的那个分类模型上，把loss从交叉熵改成MSE会报错，这两个loss函数可以比较吗？是不是MSE就不能用在余大神那个分类模型中啊？

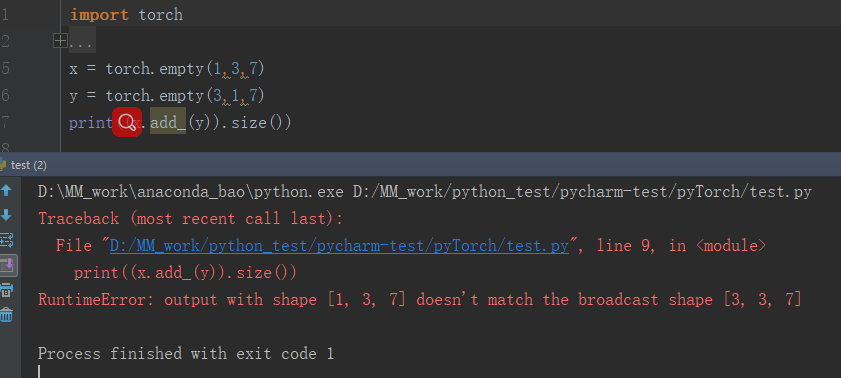
解答：MSE均方误差函数一般评价相似度或者距离的损失函数，而在分类模型中我们常用交叉熵损失函数。

# 问题16：add函数和add\_()函数的调用问题

pytorch官网中示例代码：



改成下图依然报错求问原因？



解答：需要了解add函数和add\_的区别，也就是in\_place操作（见[问题12](#_问题12：为什么torch.nn.init的初始化函数都在后面加了下划)）：in-place operation在pytorch中是指改变一个tensor的值的时候，不经过复制操作，而是直接在原来的内存上改变它的值。可以把它成为原地操作符。在pytorch中经常加后缀“\_”来代表原地in-place operation。

# 问题17：关于nn.Linear的参数设置问题

self.fc1 = nn.Linear(16 \* 5 \* 5, 120)

self.fc2 = nn.Linear(120, 84)

self.fc3 = nn.Linear(84, 10)

这里为啥是120、84，不能改成其他的？

解答：这些都是超参数，可以自己更改，只是Lenet作者当时在做mninst数据分类时，在参数为120,84时效果比较好（2）另外一个原因不能太大，比如（1200,840）时导致参数太多，过拟合，而参数太少，比如（12,8）又会导致欠拟合，所以这些参数的设置一般都是根据经验设定。

# 问题18：pytorch要怎么实现模型融合？

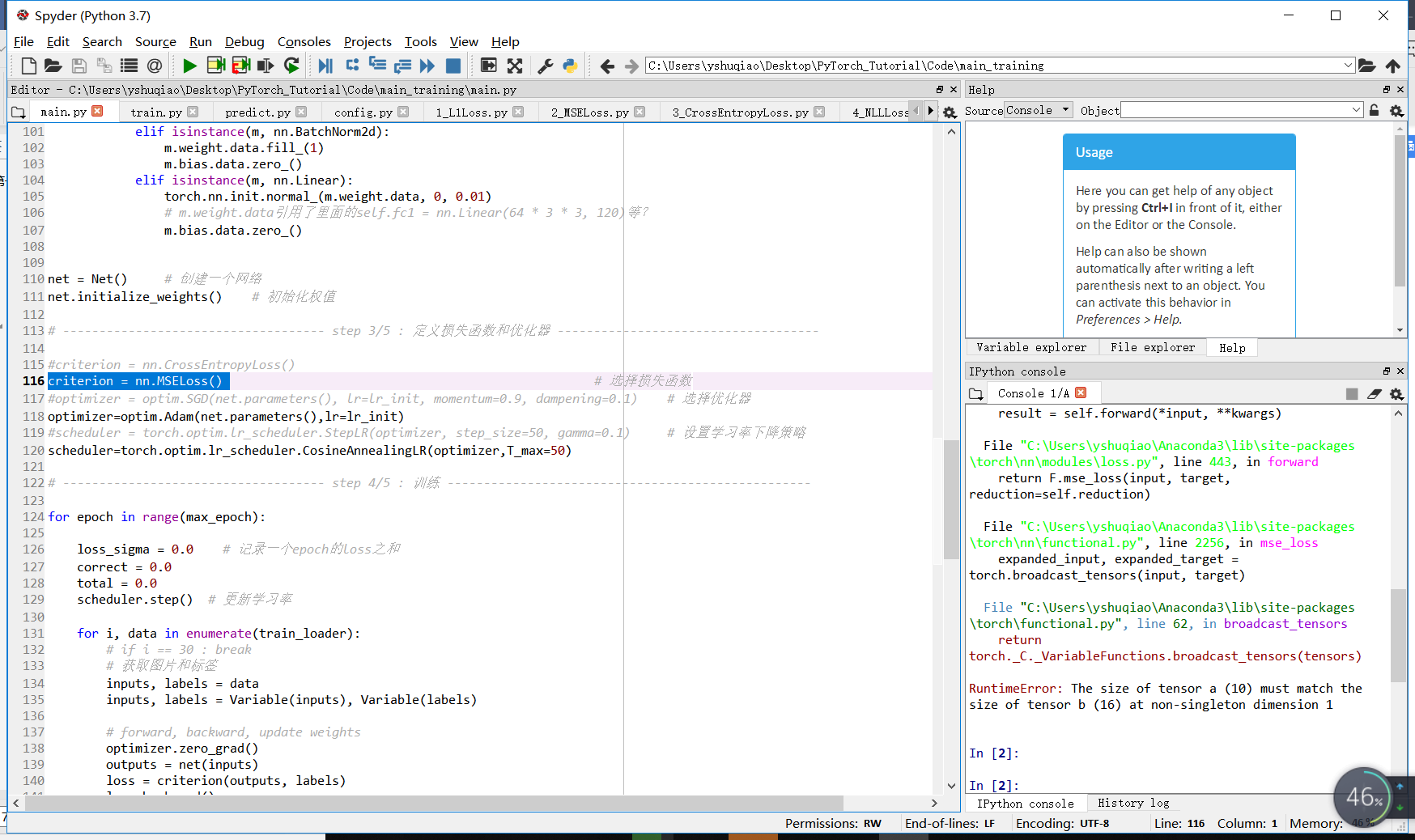
解答：问题中的“模型融合”是指多卷积核网络模型融合还是类似FPN前后的网络融合？如果是第一个情况参考inception网络，如果第二种参考darknet53网络模型

# 问题19：pytorch 预训练模型怎么得到的？如何从0开始训练网络？

解答：（1）预训练模型网络下载得到，也可以自己训练一个大数据集作为预训练模型来训练自己的小数据。（2）不使用预训练模型就是从0训练。

# 问题20：关于nn.MSELoss的参数调整问题

试着把nn.CrossEntropyLoss改为nn.MSELoss，运行出错，看提示，大概是这个损失函数的输入和标签的tensor不相符，可能需要调整数据导入时的batch\_size和标准化的参数(transform)?但还不会调。



解答：前两天有同学问这个问题，请参考前两天的。

# 问题21：SGD优化器调参经验

比较SGD优化器与原来的Adam优化器，差好多，可能要调整参数才能得到比较合适的结果，但还没会调，比如lr(初始学习率)可以不设置为原来固定的0.001，让它默认缺省值会好点?（调参方法大家总是有疑惑，请老师传授经验）

解答：学习率属于超参数的范畴，本身设置就是依靠经验，我一般常用adam+cos的方法调整学习率。

# 问题22：fine tuning代码的一些问题

余霆嵩老师code的第二个，fine tuning代码，他把模型的全连接层3给移除了，并且单独对该层设置学效率及优化方法，请问为什么要这么做？以及对单独一层设置学效率和优化方法（SGD）有什么用？学习率以及优化方法的设置，是只需要对单独一层设置即可了吗？

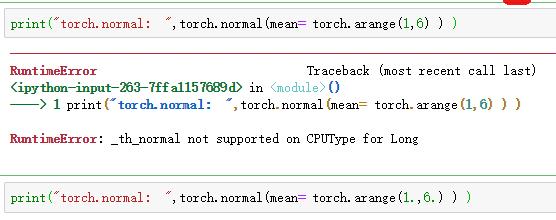
解答：（1）单独设置学习率以及优化算法的目的是给大家举例说明可以这样设置每层的学习率和优化算法,在以后自己应有中清楚模型初始化方法除了默认和finetune,每层还可以单独设置。

（2）单独对一层设置学习率和优化算法的作用是只针对这一层使用特定的学习率和优化算法，其他层的学习率和优化算法和指定层不同，比如我们在finetune时最后一层全连接层需要单独设置，因为只学习最后一层的模型参数，其他层的参数不变。

# 问题23：为什么trainset做augmentation之后，原始的trainset不会一起扔到网络里训练？

解答：因为源码里transform对应的是所有对象在dataset的\_\_get\_\_里，就像你写Totensor一样，难道还希望一部分不变？想要保留的话，写两个不同的transorm结果在一加就行了。

# 问题24：张量的split问题



如何把这个分成tensor([1]),tensor([2),tensor([3])？

解答：首先明白torch.split是拆分张量的API：torch.split(tensor, split\_size, dim=0)，需要确保将输入张量分割成相等形状的chunks（如果可分）。如果沿指定维的张量形状大小不能被split\_size整分，则最后一个分块会小于其它分块。

使用举例：

|  |
| --- |
| x = torch.randn(3, 10, 6)  a, b, c = x.split(1, 0) # 在0维进行间隔维1的拆分  a.size(), b.size(), c.size()  (torch.Size([1, 10, 6]), torch.Size([1, 10, 6]), torch.Size([1, 10, 6])) |

推荐阅读链接：

<https://blog.csdn.net/weixin_44613063/article/details/89576810>