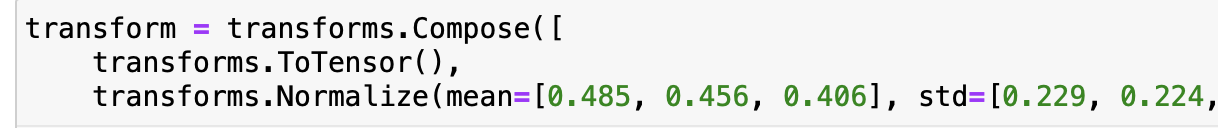
一、数据处理

1、Transform



torchvision中的transform是对图像进行预处理、数据增强等。  
Compose将多个处理步骤整合到一起。

ToTensor：将原始取值0-255像素值，归一化为0-1。

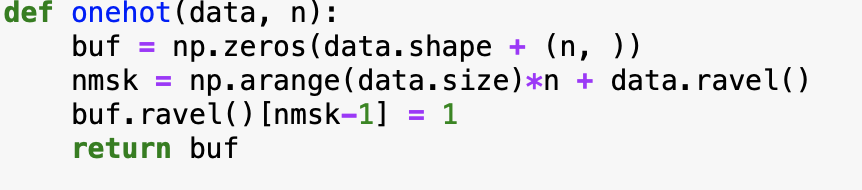
Normalize：用像素值的均值和标准偏差对像素值进行标准化。

2、One-Hot编码，又称一位有效编码。

主要采用N位寄存器对N个状态进行编码，每个状态都有它独立的寄存器位，并且在任意时候只有一位有效。此编码是分类变量作为二进制向量的表示。这首先要求将分类值映射到整数值，然后，每个整数值被表示为二进制向量。

np.arange()：此函数可有1、2、3个参数：如图我们只用了一个参数，即为终点，起点默认为0，步长默认为1，另外当有3个参数时，第一个参数为起点，第二个参数为终点，第三个参数为步长，其中步长可取小数。

np.ravel()：将多维数组降为一维



3、加载数据

os.listdir()此函数 以列表的形式返回指定的文件夹内的文件或文件夹名字

opencv处理图片，图中imread函数加一个0可以将图片转化为灰度图

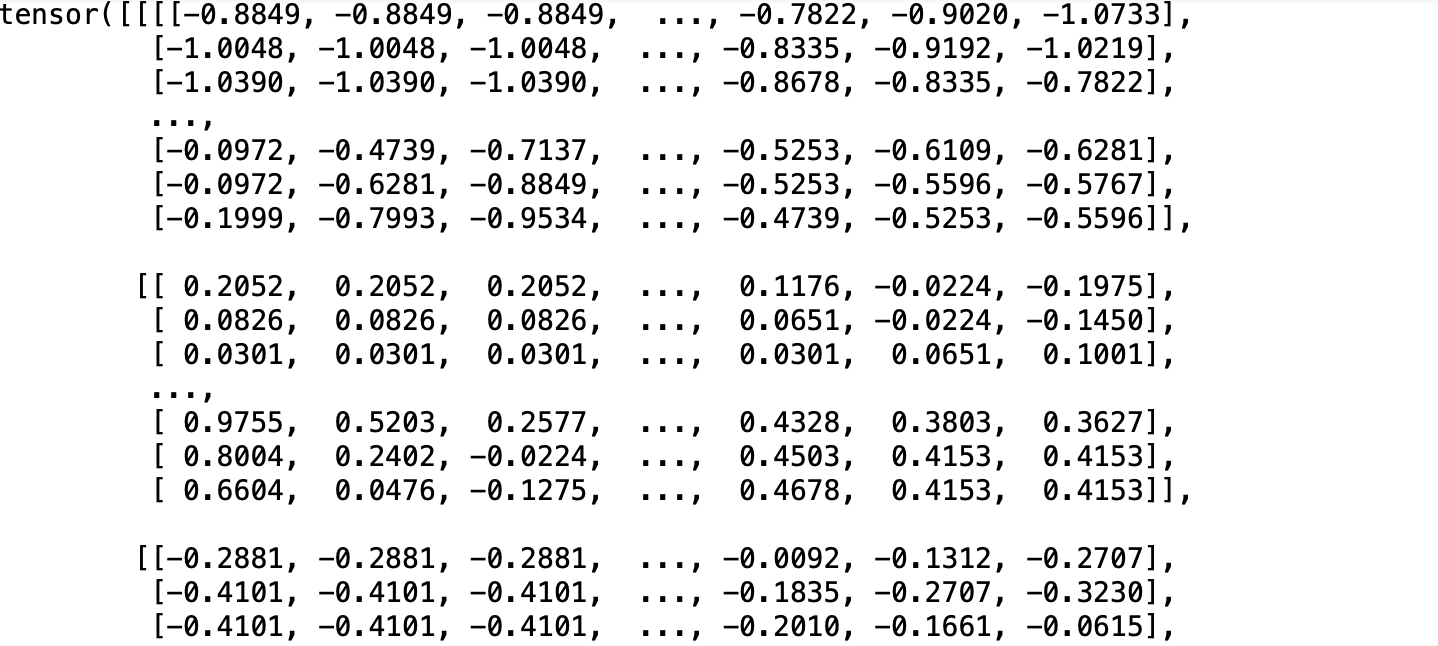
Unit8:无符号8位整数

处理图像过程中，RGB图像取值，范围为0-255，并且为了更好的处理图像，通常将像素值归一化到0-1范围内，故需要将像素值先取整数然后除以255，astype可以将像素值int类型转化为float



FloatTensor

torch.Tensor是默认的tensor类型（torch.FloatTensor）的简称。返回的值是float类型，如图查看我们的数据集中图片表示的像素值



DataLoader

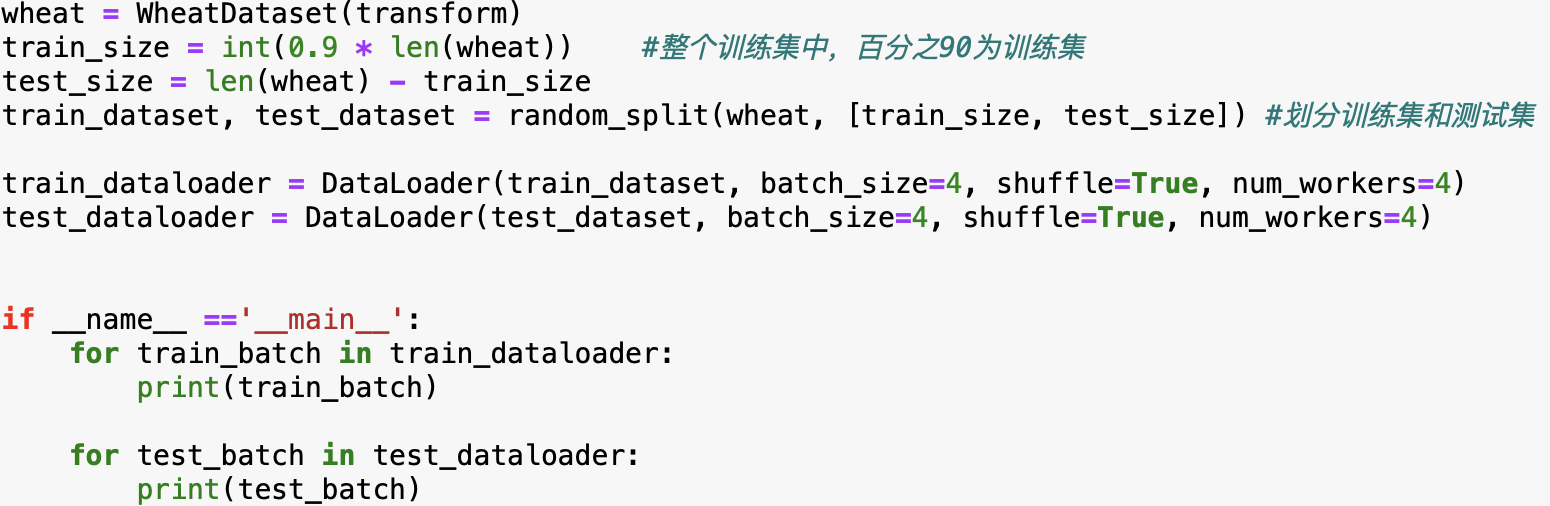
DataLoader负责将数据分批，下面是具体的参数解释：

batch\_size(int,optional):每个batch的样本数量，这里我们选择4。

shuffle(bool,optional):开始训练每个epoch前，选择true将数据打乱

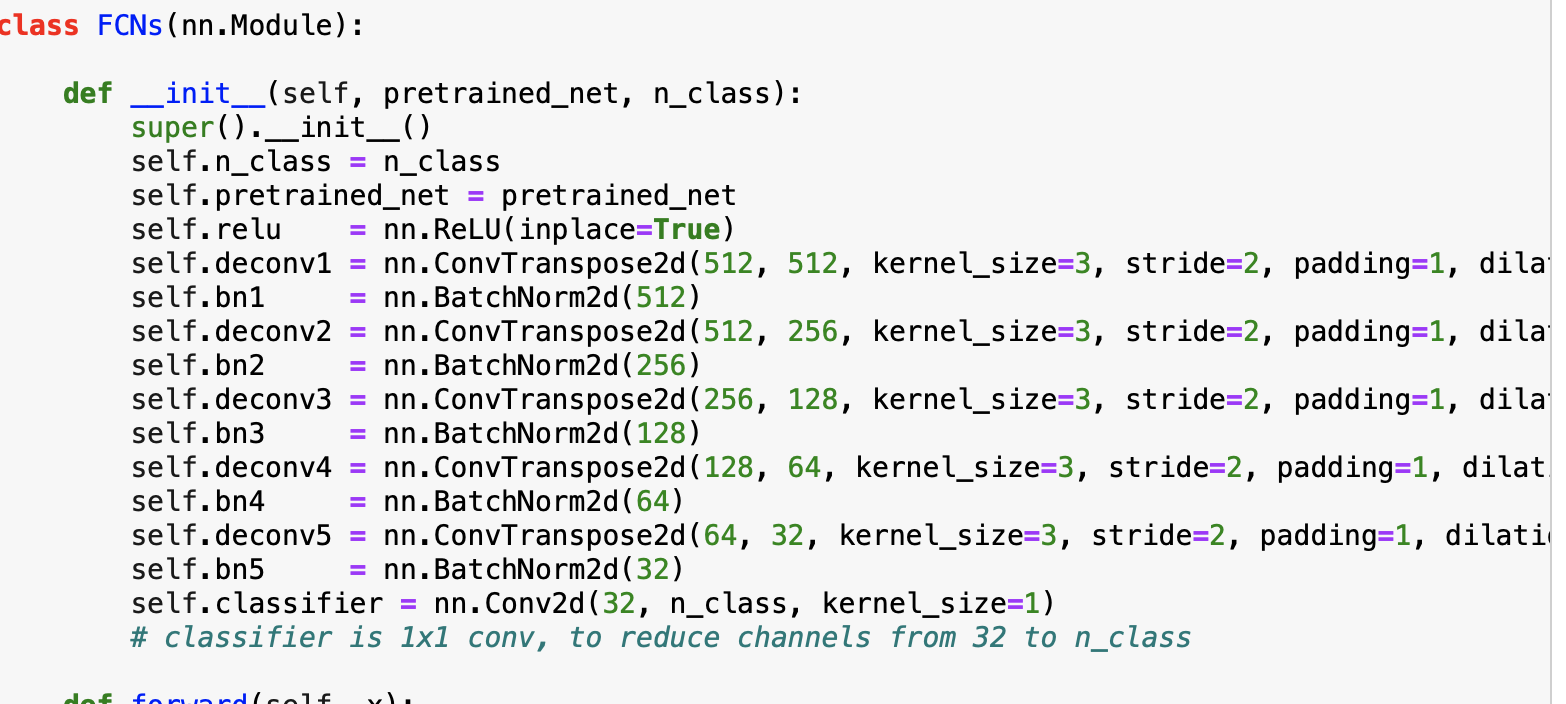
sampler(Sampler,optional):自定义从数据集中取样本的方式。如果设置这个参数，那么shuffle必需设为False。

num\_workers(int,optional):指定有几个进程处理data loading。我们这里选择4个进程



二、网络搭建

1、nn.ReLU(inplace = True)     inplace为True时，将会改变输入数据，否则不会改变原输入，只会产生新的输出。



2、torch.nn.Conv2d(in\_channels, out\_channels, kernel\_size, stride=1, padding=0, dilation=1, groups=1, bias=True)

nn.Conv2d:对由多个输入平面组成的输入信号进行二维卷积。 参数说明：

N：表示batch\_size大小，这里我们是32

C：表示channel个数

H,W：分别表示特征图的高和宽 stride：步长，控制cross-correlation步长默认为1。可设为一个整数，或一个(int,int)型的元组。

kenel\_size：卷积核尺寸，单个整数（在各个空间维度的相同长度）或由两个整数组成的list或tuple。

padding：zero-padding个数，注意，在卷积之前补零。

bias：添加偏置，为输出设定可学习的偏差。 图片经一次卷积后的尺寸：(original\_size-(kernel\_size-1))/stride

3、nn.ConvTranspose2d(in\_channels, out\_channels, kernel\_size, stride=1, padding=0,output\_padding=0, groups=1, bias=True, dilation=1)

ConvTranspose2d是进行反卷积操作。

参数说明：

in\_channels(int)：输入图像的通道个数。

out\_channels(int)：经卷积后，输出的通道个数。

kenel\_size(int or tuple)：卷积核大小。

stride(int,tuple or optional)：步长。

padding(int,tuple or optional)：输出的每一条边补充0的层数，高和宽都增加2\*padding。

output\_padding(int,tuple or optional):输出边补充0的层数，高和宽都增加padding。

groups(int,optional):从输入到输出通道的阻塞连接数。

bias(bool,optional):若为True，添加偏置。

dilation(int,tuple,optional):卷积核元素之间的间距。

4、nn.BatchNorm2d

作用是将数据归一化，不会因为数据太大而导致网络性能不稳定。其数学原理如下：

参数说明：

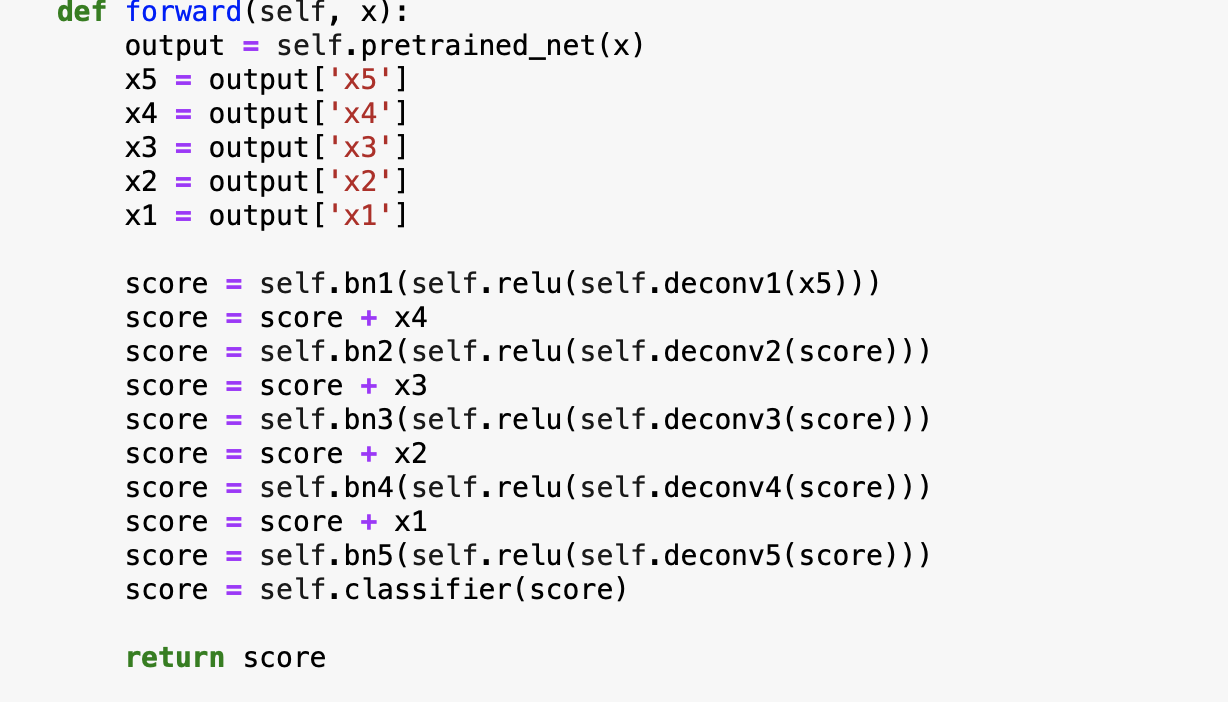
num\_features：特征数量。

eps：为了计算的稳定性，默认值为1e-5

momentum：运行过程中均值和方差的估计参数（相当于稳定系数，类似SGD中的momentum的系数）,默认值为0.1。

affine：当为True时，会给定可学习的系数矩阵gamma和beta，weight和bias会被使用。

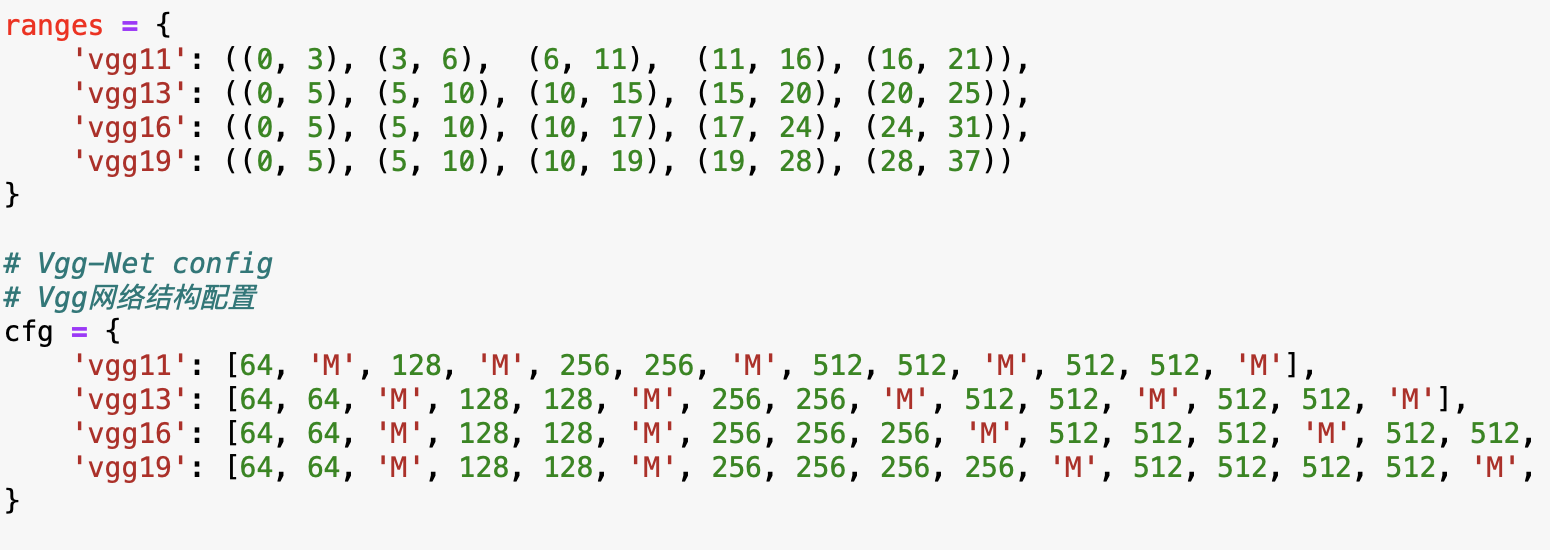
5、前向传播计算，带入到上面网络



6、FCN前半部分结构采用VGG网络模型



7、VGG网络结构配置

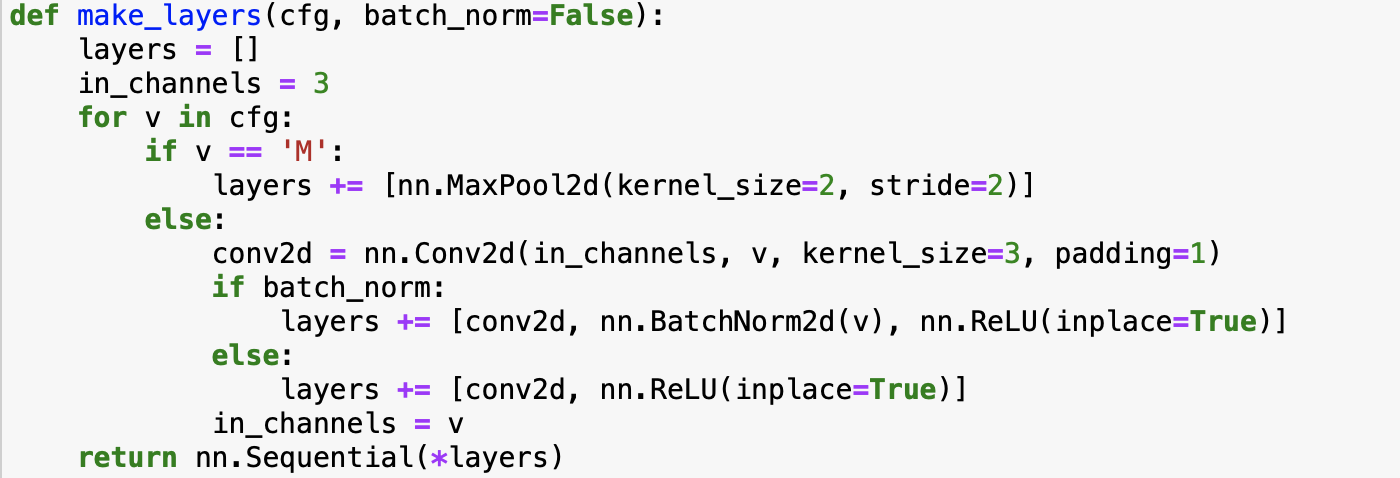


每层详情



1. Vgg-net搭建层数构造

利用之前网络结构配置cfg搭建网络



三、网络训练

1、nn.BCELoss()

pytorch中nn.CrossEntropyLoss为交叉熵损失函数，用于解决多分类、二分类问题。

nn.BCELoss()是二元交叉熵损失函数前面需要加上Sigmoid函数，加上nn.Sigmoid()语句即可。

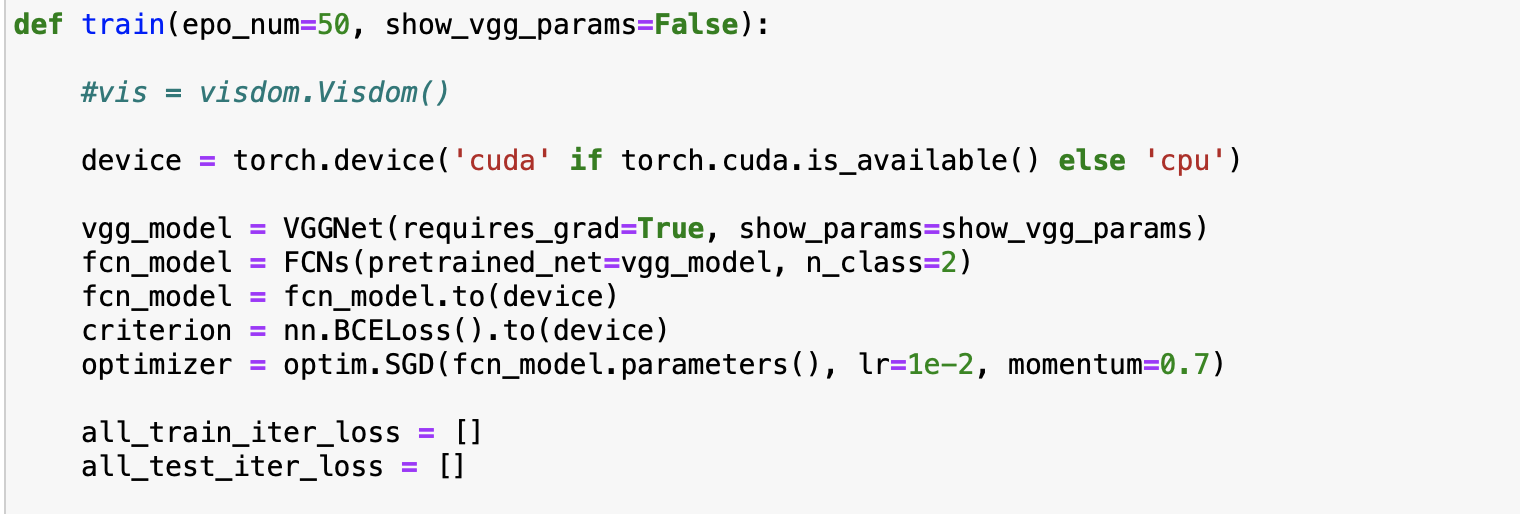
数学公式为Loss = -w\*[p\*log(q) + (1-p)\*log(1-q)]，其中p、q分别为理论标签、实际预测值，w为权重。

torch.nn.BCELoss(weight=None,size\_average=True,reduce=True,reduction='mean')

weight必须和target的shape一致，默认为None。

默认情况下reduce=True，size\_average=True，返回loss的均值，即loss.mean()

默认reduction=‘mean’，返回loss的平均值



2、optim.SGD(fcn\_model.parameters(), lr, momentum)

SGD的全称是Stochastic Gradient Descent（随机梯度下降）。

梯度下降法主要有三种：批量、随机、小批量

①批量梯度下降法：整个训练数据集计算梯度

②随机梯度下降法：随机取一个样本计算梯度

③小批量梯度下降法：选取少数（即batch\_size的大小）样本组成一个小批量样本，用这个小批量样本计算梯度。

fcn\_model.parameters()：获取fcn\_model网络中的参数。搭建好神经网络后，网络的参数都保存在parameters()函数中。

learning rate(lr)：学习率较小时，收敛到极值的速度较慢。学习率较大时，容易在收敛过程中发生震荡。

momentum:物理学中指力对时间的累积。每次x的更新量v=-dx\*lr，其中dx是目标函数func(x)对x的一阶导。

当本次梯度下降-dx\*lr的方向与上次更新量v的方向相同时，上次的更新量能够对本次的搜索起到一个正向加速的作用。

当本次梯度下降-dx\*lr的方向与上次更新量v的方向相反时，上次的更新量能够对本次的搜索起到一个减速的作用。

.detach()：返回一个新的Variable，从当前计算图中分离下来，但是仍指向原变量的存放位置，只是没有梯度，即使后期令requi\_grad=True，也不具有梯度。后面进行反向传播时，需要调用detach()的Variable就会停止，不会再继续向前传播。

