**不同复杂程度的神经网络对比实验报告**

本次实验对比了LeNet, AlexNet, VGG-11以及ResNet在Fashion-MNist数据集上的表现。本报告分为神经网络原理、主要代码和算法流程介绍以及实验结果分析三个部分。

**神经网络原理**

这部分分为全连接层、卷积层、池化层和残差块进行展开。

全连接层先经过一个线性变换，再经过一个激活函数。用向量化表达线性变换，设l层的输入向量为，输出向量为，偏移为，并且定义系数矩阵，其第行第个元素为从层的神经元连接到层的神经元的系数，与的关系为：

常用的激活函数有

和

后者具有良好的数学性质——连续、处处可导，但是具有计算复杂度高、收敛速度慢、且容易导致梯度消失的缺点（，当x偏离0时导数值接近于0，而ReLU只有当x为负值时导数值为0），所以在实践中更常使用ReLU作为激活函数。

卷积层有卷积核大小、填充和步长这些超参数。卷积是将卷积核与图像的对应位置相乘后相加，再加上一个偏移量，填充是扩充原图像的边界（比如用0进行扩充），步长是卷积核每次在图像上移动的距离。设前一层的通道数为m，该层的通道数为，则需要个卷积核，设输入图像大小为，填充为，卷积核大小为，则经过卷积操作后图像大小为

其中为或，代表输出图像的高或宽，方括号表示下取整函数。经过卷积层后，还需要再经过一次激活函数。卷积层的作用主要是两个，第一个是参数共享，由于图片在不同地方可能有类似的模式，在图片的某个地方提取出有效特征的检测子很有可能在其他地方也能提取出有效特征；第二个是连接的稀疏性，输出矩阵的每一个元素只取决于输入矩阵中该点附近的元素。

池化层是为了浓缩特征信息，减少特征点数量，最后能将特征点放进全连接层中。这一层没有需要机器学习的参数，只有人为设定的参数。常用的池化层包括最大值池化和平均值池化，最大值池化提取图像的最主要特征，而平均值池化提取某个区域所有特征的平均值。池化层也有填充，步长等超参数，与卷积层类似，此处不赘述。

残差块主要运用于很深的网络，此时某些网络层可能不仅不会使得结果变得更好，反而会使得结果变得更差，残差块的作用是提供 “捷径”绕过这些层。形式化表达其步骤，沿用全连接层的定义，并且设激活函数为，则

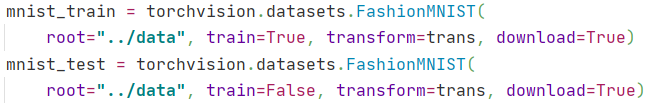
当为零矩阵、为零向量时，（如果均采用ReLU损失函数，）, 相当于绕过了第和层。

搭建神经网络时，一般先经过卷积层和池化层，不断减小图像大小，增加图像通道数，之后经过全连接层。最后给出预测值时常常使用softmax函数：

其中，，是输出层向量，函数返回值表示预测值向量，其中第个数表示预测为第类的概率。

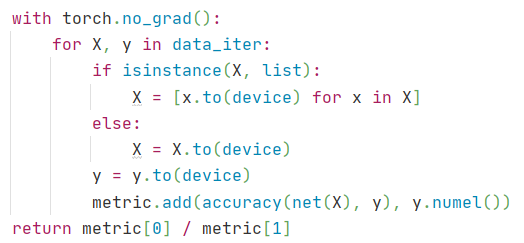
**主要代码和算法流程**

首先介绍四个神经网络共有的部分——读取数据、精确度计算、训练，再分别介绍网络结构。代码采用pytorch框架，在租用的A5000云GPU上进行训练。

读取数据可以直接从torchvision.datasets.FashionMNIST读取，核心代码如下：

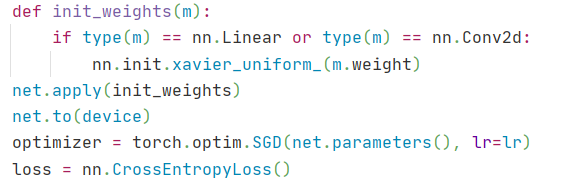
训练集有60,000张图片，测试集有10,000张图片，每张图片是28\*28的灰度图像，总共有十个类别。值得注意的是，除了LeNet的其他网络都是224\*224的输入，所以要先用插值函数将图像扩充成224\*224的大小。将数据读入后，为了做随机梯度下降，会将训练集和测试集都分成若干批次(batch)，其中训练集需要先打乱顺序。所以读入函数返回的是对于训练集和测试集的迭代器，用迭代器对数据集进行遍历，返回X和y，其中X是一个四维Tensor，表示batch\_size\*通道数\*图像大小，y是一个一维Tensor，表示该批次中图像的标签值。

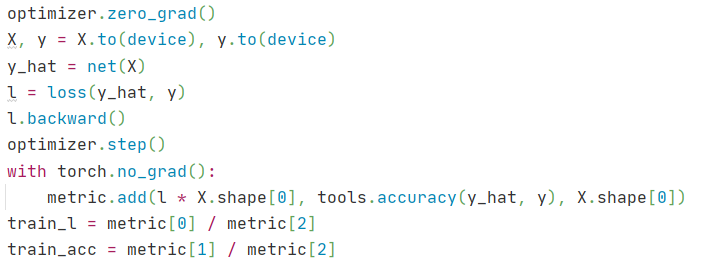
精确度计算采用矩阵运算的方式，加快计算速度。代码通过累积每个批次内net(X)和y相同的元素（代表预测对的样本），最后除以样本总数，得到精确度。核心代码如下：



其中，metric的作用是累计器。

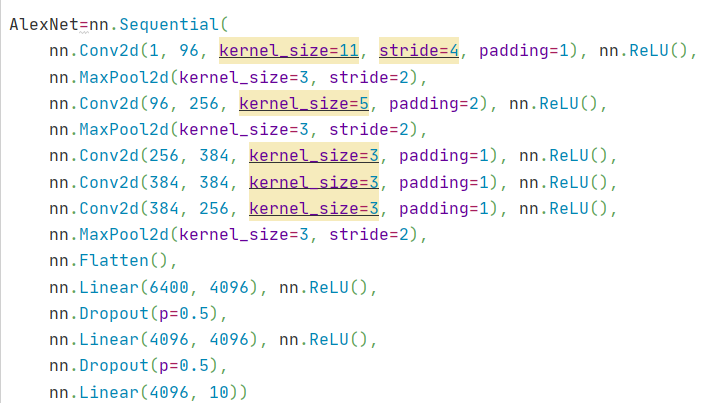
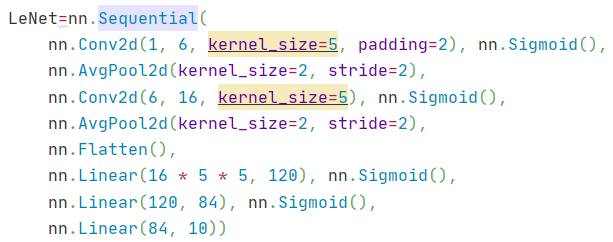
训练时先随机初始化，定义损失函数，然后做n个epoch的随机梯度下降，而做随机梯度下降的过程是先打乱训练集的顺序（一个epoch只打乱一次），按照顺序每次从打乱的训练集中取一个批次，对这个批次计算损失函数的梯度，然后进行梯度更新，将训练集中所有的数据全部取完后，完成这个epoch的训练。核心代码如下：





接下来介绍网络结构。

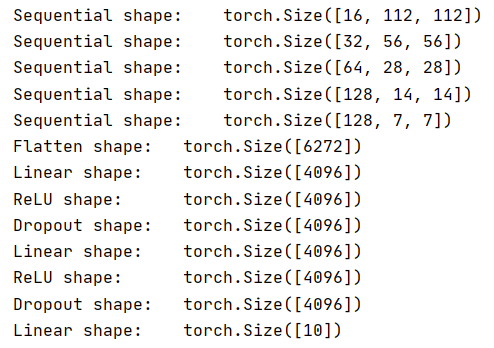
对于LeNet, 网络结构如下左：



可以看见，先经过两次卷积和池化层组合，再拉成向量，进入全连接层。

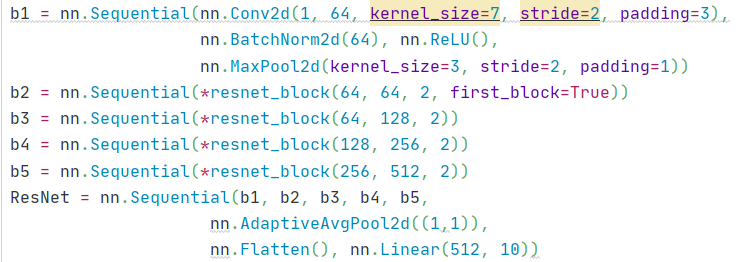
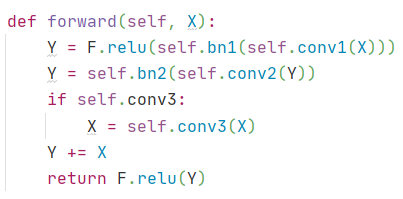
对于AlexNet，网络结构如上右。可以看见，先经过若干层卷积和池化组合，再进入全连接层，并且在全连接层后增加了一个dropout层。dropout是一种防止过拟合的正则化技巧，表示每个节点有的概率在进行前向传递时不被计算，而其他节点的值要除以（防止梯度消失），这样，可以防止只依赖于某个特征进行梯度计算，有利于将权值分散开。

对于VGG-11，主要是将若干卷积层和一个池化层这个网络结构作为一个VGG块来进行使用，网络结果打印出来如下：



其中每一个Sequential都是一个卷积层、激活层和池化层组成的VGG块，其维数分别表示卷积层数量、输入的通道数和输出的通道数。

对于ResNet，类似VGG-11,用原理部分介绍的残差块来搭建网络，搭建残差块的核心代码如下左：



整体结构如上右，resnet\_block的参数分别表示输入的通道数、输出的通道数和残差块的数量，并且，增加了批归一化层(batch normalization)，这一层的作用是保证输出的平均值和方差在输入发生变化时保持不变。

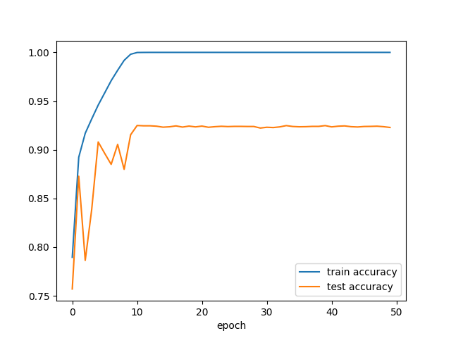
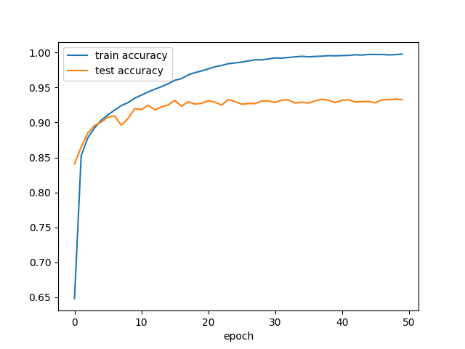
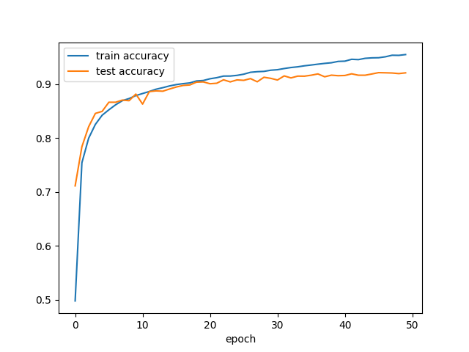
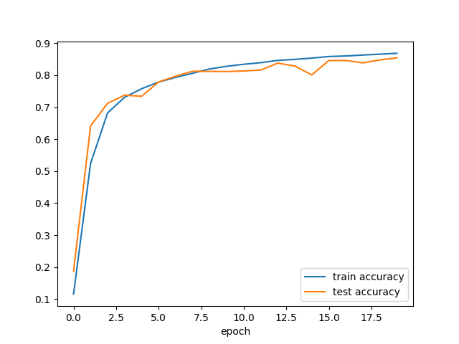
**实验结果分析**

基本超参数设置如下：

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | learning rate | number of epochs | batch size |
| LeNet | 0.9 | 50 | 256 |
| AlexNet | 0.01 | 50 | 128 |
| VGG-11 | 0.05 | 50 | 128 |
| ResNet | 0.05 | 50 | 256 |

实验结果如下：

各算法训练与测试误差随epoch变化如下（从左到右，上到下分别是LeNet, AlexNet, VGG-11, ResNet）：



从这个结果中可以看出，随着模型变得复杂，在训练集上的精度越来越高，而在测试集上的精度几乎没有提高，说明过于复杂的模型只是学到了一些训练集特有的特征，而没有学到真正能反映数据分布的参数，这是导致过拟合的常见原因，然而在该实验中，并没有出现明显的过拟合现象（随着训练集精度的提高测试集精度反而下降），原因是训练的样本足够多(60,000张图片)，且训练集的数据分布与测试集的数据分布较为接近。