1. 题目:

使用pytorch搭建深度卷积神经网络，并且进行图像分类识别。

1. 问题描述：

读取CIFAR数据集，编程实现卷积神经网络模型，利用数据集中给出训练集模型，并且在测试集中及进行分类以验证模型的功效。安装cuda可以更快速的跑这次的数据，在此过程中将设计的卷积网络模型给予展现（每层卷积核的大小，卷积核的个数，池化层类型，以及cost function，测试集上的精确率和它的图像变化曲线图。训练完成后，各层卷积核的可视化展示。

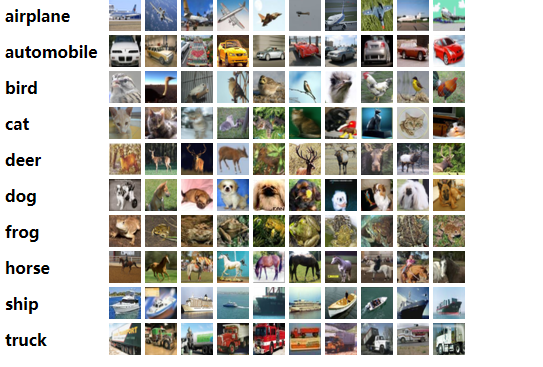
1. 数据集的描述：
2. 使用的数据集：CIFAR-10 图像数据集：<http://www.cs.toronto.edu/~kriz/cifar.html>.
3. 下载完成的数据集展现：



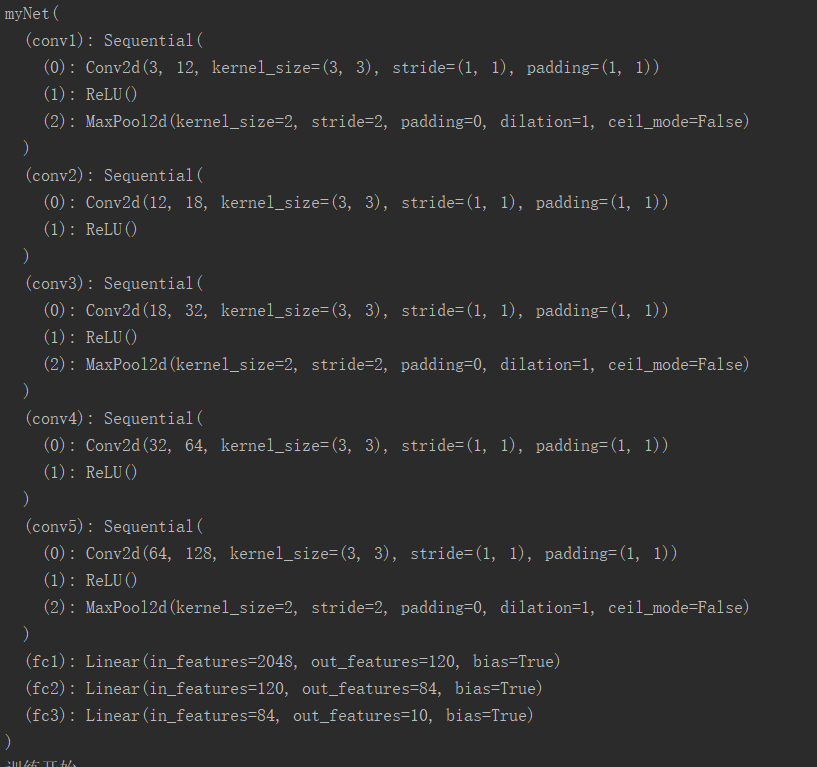
1. 开始读取数据集的格式：

Cifar数据集总共有10类（意为10个lables），分别为：airplane，automobile，bird，cat，deer，dog，frog，horse，ship，truck。全部数据集总共有60000张彩色图片，图像大小是3（R，G，B）通道的32\*32，即大小为32\*32\*3。数据分为测试集和训练集，训练集有50000张彩色图片，构成了5个训练批，每个训练批 10000张彩色图片，测试集10000张彩色图片，构成一个训练批。测试批的数据里，取自10类中的每一类，每一类随机取1000张，抽剩下的就随机排列组成了训练批。注意一个训练批中的各类图像并不一定数量相同，总的来看训练批，每一类都有5000张图。

1. 10类中的部分图片加以展示（该数据集来自于）<http://www.cs.toronto.edu/~kriz/cifar.html>.



1. 实验结果图：
2. 所设计的卷积神经网络的架构图：



每层卷积网络：

输入:32\*32\*3

①:conv1（卷积核大小3\*3，卷积核个数 12，池化层类型 maxpool,激活函数：relu，步长：1）

②:conv2（卷积核大小3\*3，卷积核个数 18，池化层类型为空,激活函数：relu，步长：1）

③:conv3（卷积核大小3\*3，卷积核个数 32，池化层类型 maxpool,激活函数：relu，步长：1）

④:conv4（卷积核大小3\*3，卷积核个数 64，池化层类型 为空,激活函数：relu，步长：1）

⑤:conv5（卷积核大小3\*3，卷积核个数 128，池化层类型 maxpool,激活函数：relu，步长：1）

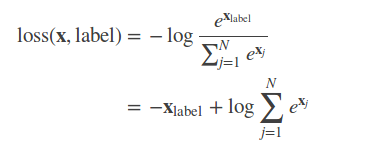
⑥：fc1：linear(in\_features=2048,out\_features=120)

⑦: fc1：linear(in\_features=120,out\_features=84)

⑧: fc1：linear(in\_features=84,out\_features=10)

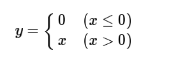
输出：4\*4\*128

（2）coss function：多类交叉熵函数。公式如下：

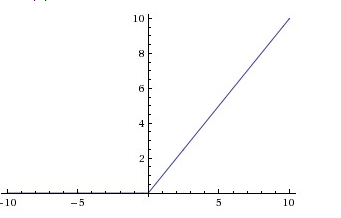


**x**∈R*N*，是没有经过 Softmax 的激活值，*N* 是 **x** 的维度大小（或者叫特征维度）；label∈[0,*C*−1] 是标量，是对应的标签，可以看到两者维度是不一样的。C 是要分类的个数。***w***∈R*C* 是维度为 *C* 的向量，表示标签的权重，样本少的类别，可以考虑把权重设置大一点。

（3）Relu函数：

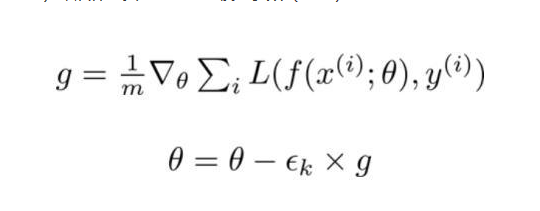


其图像为：



为什么采用relu（）函数：可以引入非线性因素，解决线性问题所不能解决的问题，relu会使其中的一部分神经元的输出为0，造成了网络的稀疏性，并且减少了网络的参数之间的相互依赖性，减少过拟合的发生。假如采用sigmoid函数，运算量太大，函数反向传播的时候，会发生梯度消失的情况。

1. 优化策略：

采用随机梯度优化算法（SGD）。通过一个小批量从梯度负方向上来更新模型参数。直接调用pytorch的包中的优化策略，其中学习率为0.001，动量为0.9。其公式为: 

1. 卷积核可视化：

第一层conv1的卷积核可视化：



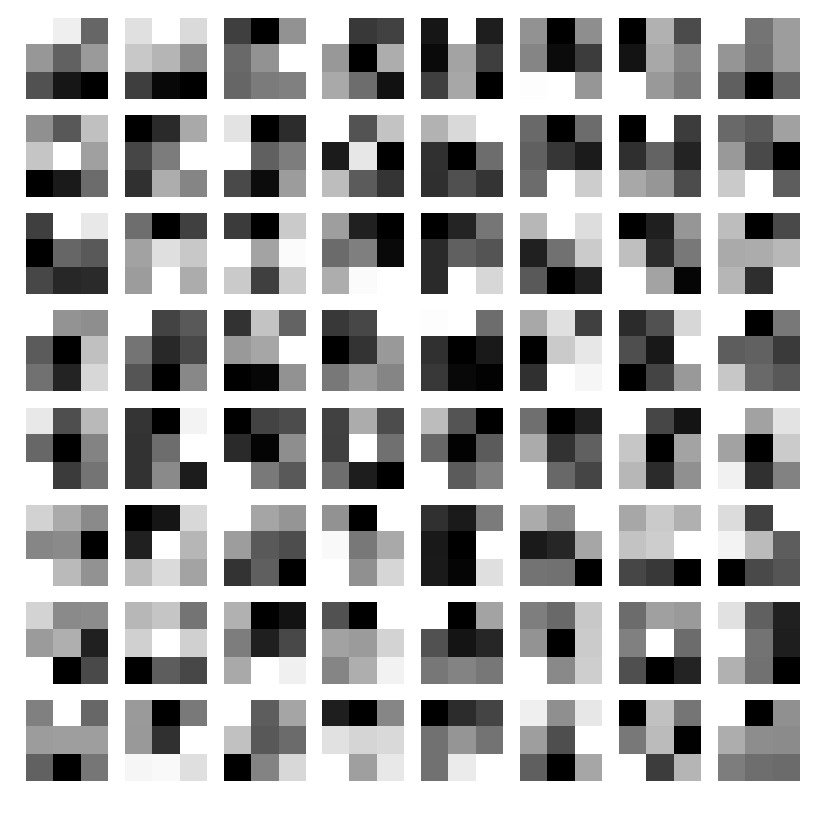
第二层conv2的卷积核可视化：



第三层conv3的卷积核可视化：



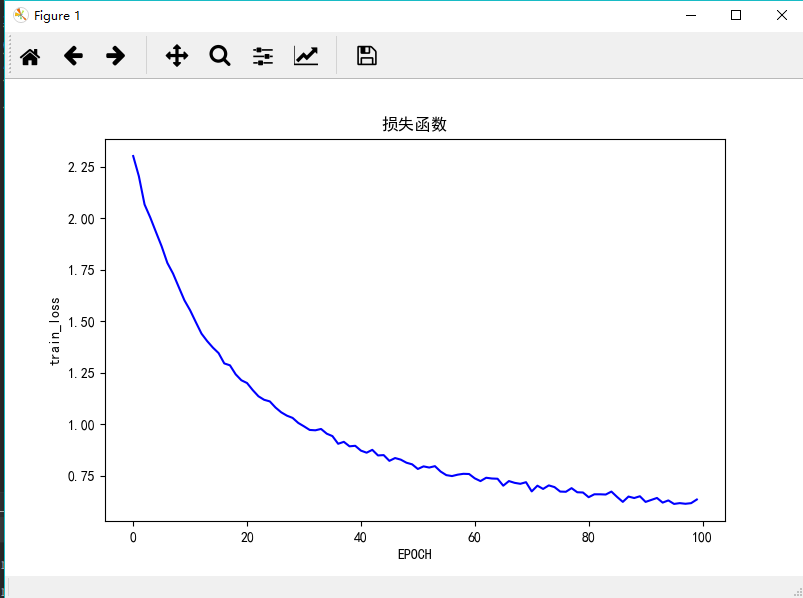
第四层conv4的卷积核可视化：

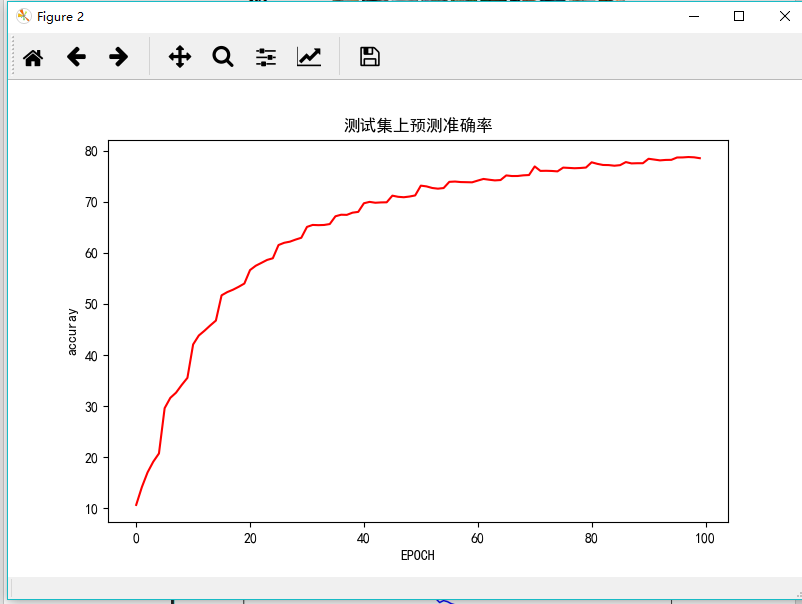


第五层conv5的卷积核可视化：



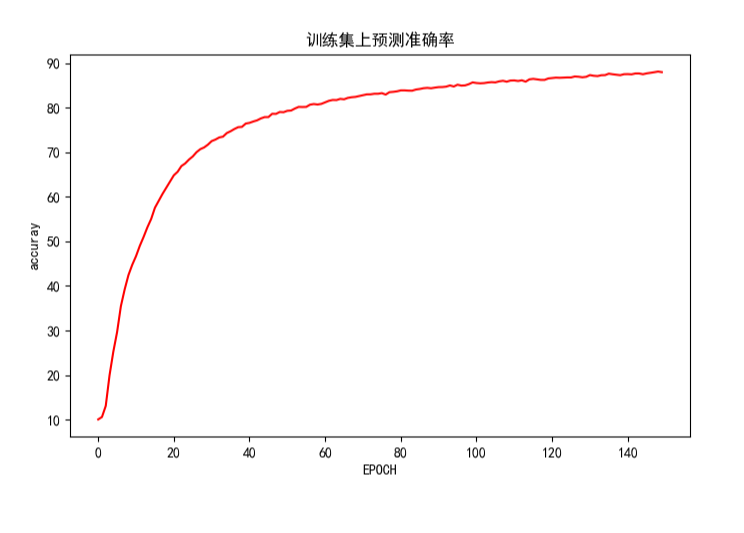
（2）图形：我将遍历次数设计为20，学习率为0.001，将前面的训练集进行了处理，把训练集进行了填充和一半概率翻转，一半概率不翻转，归一化处理，发现测试集上的精确率有所提高，但是发现EPOCH（遍历次数）的数字太小，从图中看训练集上的精确率一直在上升阶段，训练集并没有发生收敛（第二张图片上名称改为训练集上的预测精确率）当EPOCH=40时达到了一个较好的测试集准确率，但是仍然感觉并没有收敛：

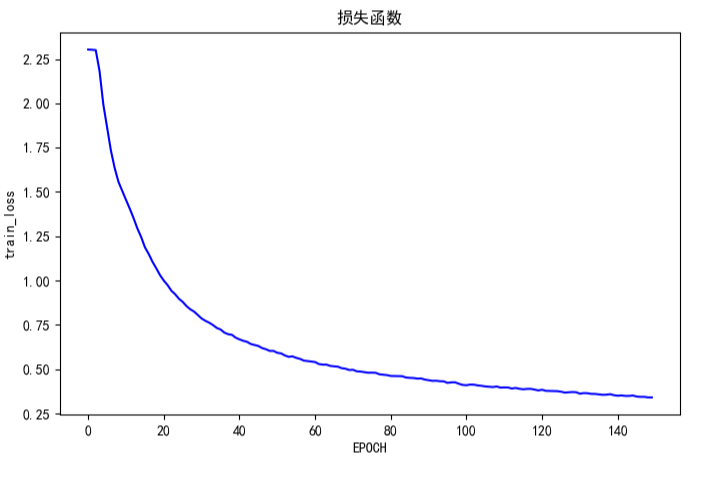


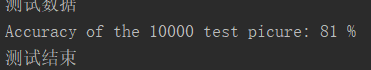




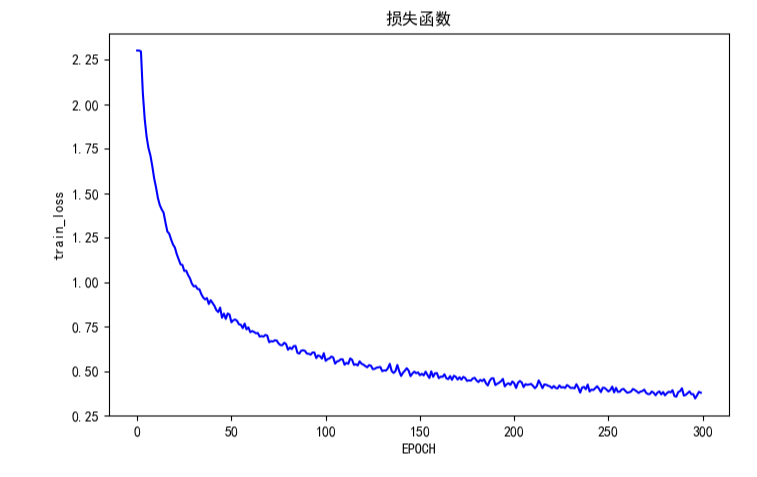
当EPOCH=150时，bactch-size=50得到一个较为准确的精确率：

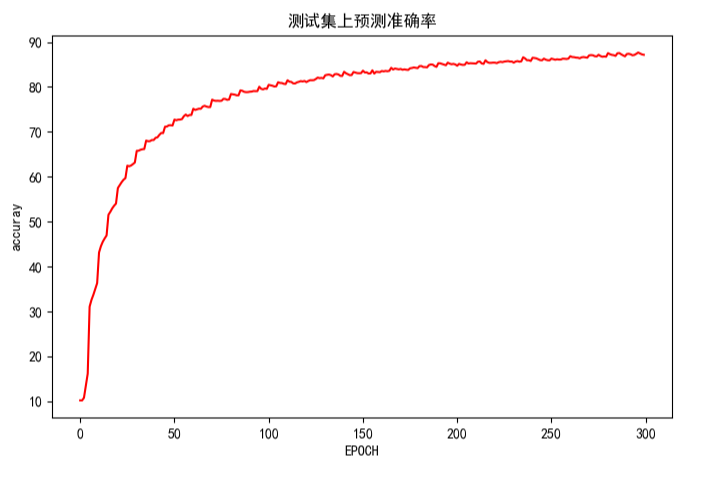






当卷积核的大小为5\*5进行卷积, EPOCH=60时，bactch-size=50：



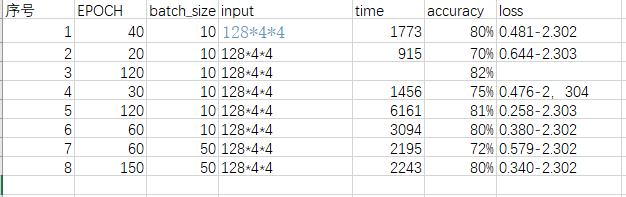


其测试集上的精确率达到了77%

（3）实验结果分析：

第一次拿只经过归一化的数据进行调试，发现精确率只有54%,EPOCH和batch\_size的值都太小了，导致数据还只是欠拟合，进而测试集的数据集精确率太低，模型测试太差。

首先根据这次试验，我做了一个关于改变这个模型的参数，来增加其中的一个参数，看其测试集上的精确率的变化情况。



从中可以发现batch\_size,EPOCH都会影响最后的结果，梯度下降自身就是一个迭代过程。随着EPOCH数量的增加，神经网络中权值跟新的迭代次数增多，运行过程中是不断拟合过程的，其中发生过拟合和欠拟合。选择一个好的batch\_size会优化网络模型的性能和速度，其中也有试过改变其输出，也会升高或者降低最后测试集上的精确率。在实验前就已经想好卷积几次，过程中也发生过不少错误，比如每一次卷积，激活函数下，我都会进行池化一次，发现一次次池化，就会使得最后输出减少为原来的一半，所以池化得没有东西。当然kneal—size的选择也会影响，从每一次跑完代码来看，5\*5的总是没有3\*3的扫描的效果好，我觉得3\*3扫描得到信息比较多一些。最后得到最好的模型，其测试集上精确率达到82%。

优化策略中的方式也会影响最终的结果，不同的方式使得梯度下降中loss下降得程度不同，Adam和Nesterov Momentum所得到得loss下降得最低，SGD中学习率的大小也会影响，学习率太大，导致梯度下降过程中会错过最佳的值，学习率太低，导致训练时间过长，在不同阶段的epoch，采用不同的学习率会使得梯度下降更加接近那个最小值，拟合得会更好，但是我在这次实验中采用得恒定不变的学习率。

1. 参考文献

Pytorch中文档：

<https://pytorch-cn.readthedocs.io/zh/latest/torchvision/torchvision-datasets/>

pytorch官网：

<https://pytorch.org/>

cifar数据集的卷积网络模型：

https://blog.csdn.net/sunqiande88/article/details/80100891