卷积神经网络

班级：自 73

姓名：陈昱宏

学号：2017011507

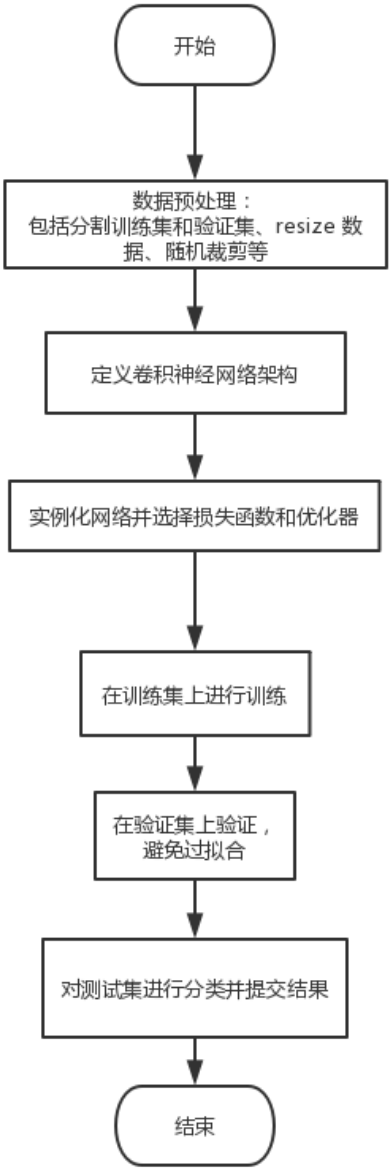
Kaggle 账号和排名：

账号：THUYuHong，Public 榜：124 名、0.87933，Private 榜：128 名、0.87571

一、任务描述：

选择适合的卷积神经网络架构和参数，利用助教提供的 30000 张已标记的服装图片进行训练，完成对 5000 张测试集的分类。

二、实验流程：



三、数据预处理:

（一）数据分割:

这部分我写了一个 python 脚本（cutdata.py），预先对训练数据进行分割，分别写入两个 npy 文件和两个 csv 文件，并确认过两个文件中没有重叠的数据。

此处采用的验证集选取方式为多倍交叉验证，倍率为5: 1。

（二）数据裁剪：

助教所提供的数据为$(N, 784)$的尺寸，但是神经网络的输入要是$(N, 1, H, W)$的形式，所以这里需要对数据进行 reshape，这个部分采用简单的 list 操作即可完成 reshape 操作。

此外，为了提高网络的效率，这里加入了随机裁剪的功能，将原先28×28的图片，裁剪成24×24的大小，裁剪方式如下：

1.选择两个随机数，范围是0~3之间的整数。

2.两个随机数记为(h, w)，将(h, w)作为左上的点，裁剪出24×24的图片。

（三）数据封装：

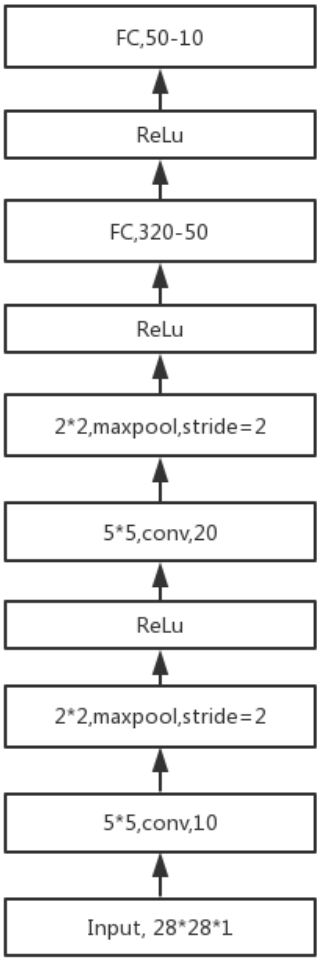
由于 DataLoader 需要接收的数据类型 torch.utils.data.Dataset，所以需要先对原本的 numpy 数组转换成 torch.Tensor，再封装成 torch.utils.data.Dataset，这个类的定义在 dataClass.py 的文件中，主要需要定义__init__()、__getitem__()、__len__()这三个函数。

四、网络结构选择：

由于尝试了很多神经网络，这里将介绍一些选择的网络和其性能，以下的神经网络中，损失函数选择的是交叉熵，优化器选择的是SGD。

（一）助教示例中的网络：

1.网络结构：

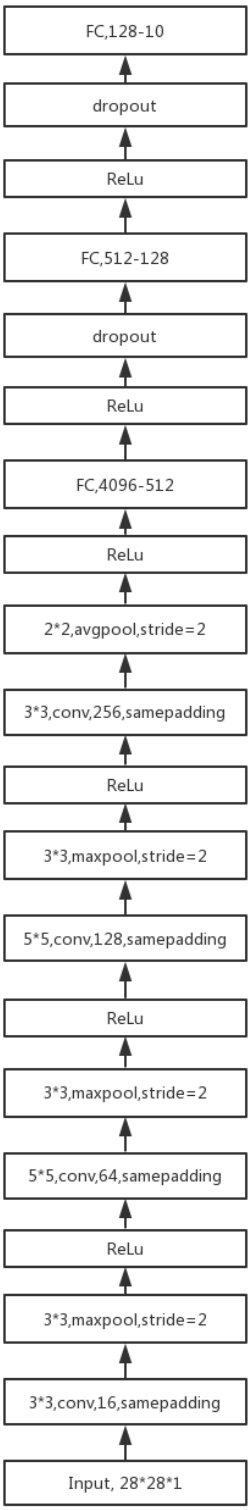


2.实验效果（学习率 0.001）：

将测试集数据提交上去后，最高仅到 0.80466，之后便无法再提升，因此后续在此网络的基础上，进行了网络的改进。

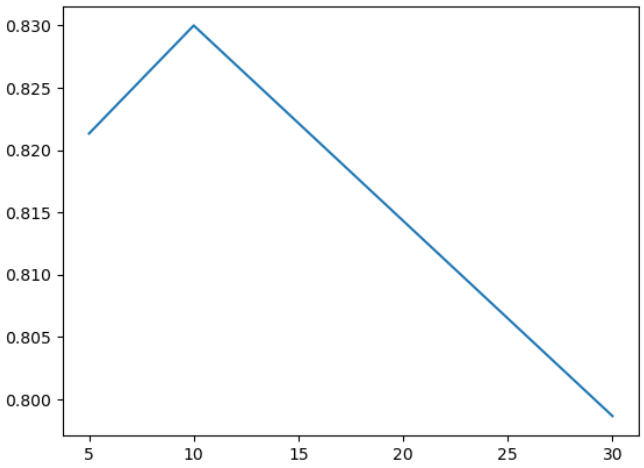
（二）自制 7 层神经网络：

1.网络结构：



2.实验效果（学习率 0.001）：

下图横轴代表训练次数，纵轴代表测试集的准确率：



由于之后找到了更好的神经网络，所以在提交三次文件后便舍弃此网络，因此上图可能并不能完全的代表此网络的效率，下面将会展开介绍之后的网络。

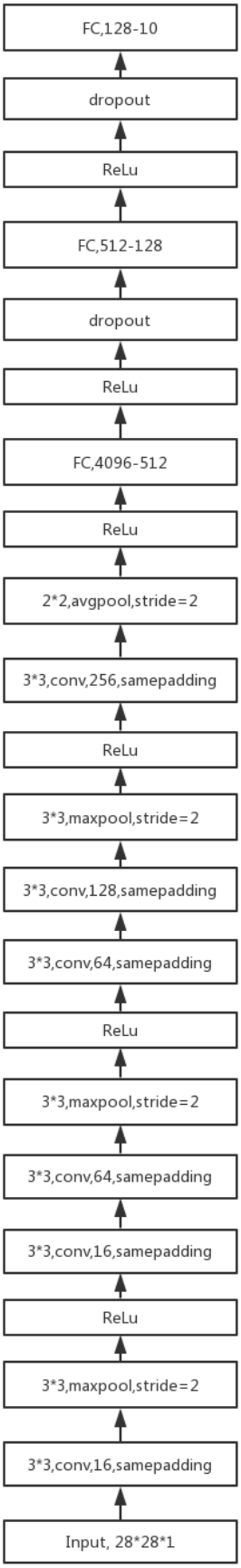
（三）自制 9 层神经网络：

1.网络结构：

这里参考 VGG 的改进手段，对自制 7 层神经网络进行改进。VGG 论文中提出感受野的概念，大卷积核的卷积层，可以通过多层小卷积核的卷积层叠加，得到相同的感受野。

这里将 5*5 大小的卷积层换成 2 层 3*3 大小的卷积层，如此可以得到相同的感受野，但可以用更少的参数去训练。

整体网络结构如下：


FC,128-10
dropout
ReLu
FC,512-128
dropout
ReLu
FC,4096-512
ReLu
2*2,avgpool,stride=2
3*3,conv,256,samepadding
ReLu
3*3,maxpool,stride=2
3*3,conv,128,samepadding
3*3,conv,64,samepadding
ReLu
3*3,maxpool,stride=2
3*3,conv,64,samepadding
3*3,conv,16,samepadding
ReLu
3*3,maxpool,stride=2
3*3,conv,16,samepadding
Input, 28*28*1

2.实验效果（学习率 0.001）：

虽然这个网络的表现在训练次数增加时，表现比自制 7 层神经网络好，但是它的测试集准确率无法有很显著的提升，因此在训练 30 次后，便舍弃这个网络。

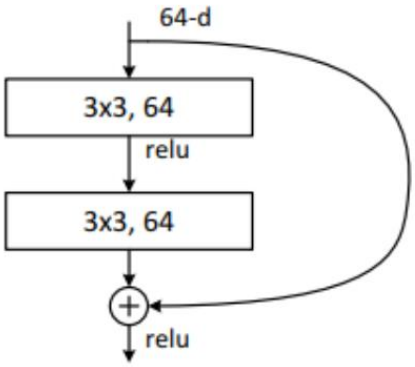
（四）ResNet18

ResNet18 的代码参考网上开源的代码 (https://blog.csdn.net/winycg/article/details/86709991)，并在此基础上，进行修改和改进。

1.网络结构：

这里采用的是经典的 ResNet18 架构，需要注意的是，因为我们的输入是 1 通道的，所以需要修改网络结构第一层的输入通道数。

残差块结构如下：

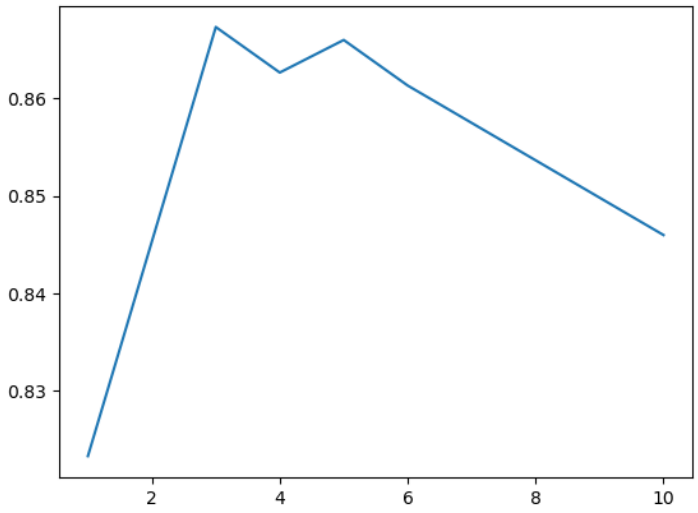


整体网络参考如下 18 层的结构，其中有根据实际输入进行微调：

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

2.实验效果与改进思路：

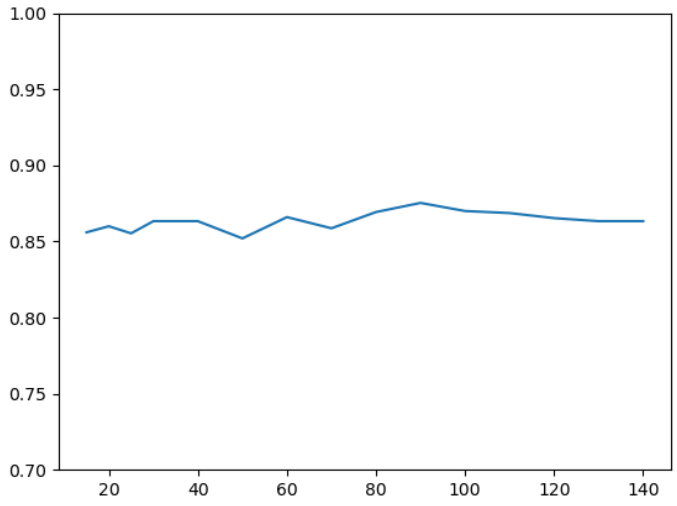
原始 ResNet18 的效果如下图（横轴为训练次数，纵轴为测试集正确率，学习率 0.001）

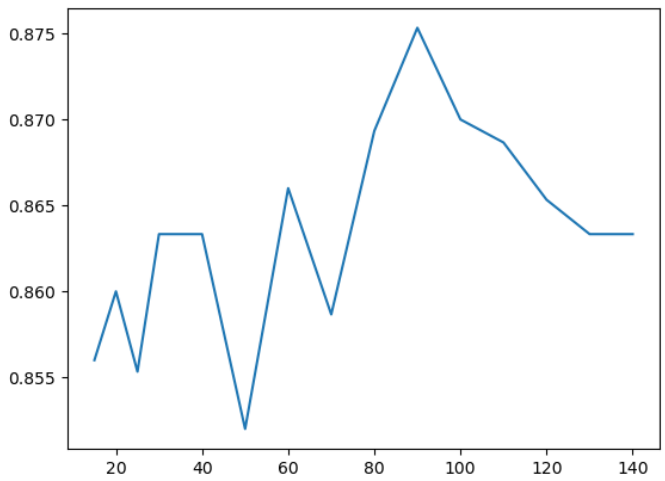


在看到训练 5 次之后，测试集的准确率开始下降，怀疑是过拟合，因此这里对卷积层的通道数进行调整，减少训练的参数，进行的主要改进为：将四个残差块的通道数改成 32、64、128、256，以此来减小训练参数，避免过拟合。

此外，减少过拟合情况的另一个改进方式，可以透过数据增广，增加训练的数据量，这里我选用随机裁剪，具体实现在数据预处理部分已经说明，此处不再赘述。

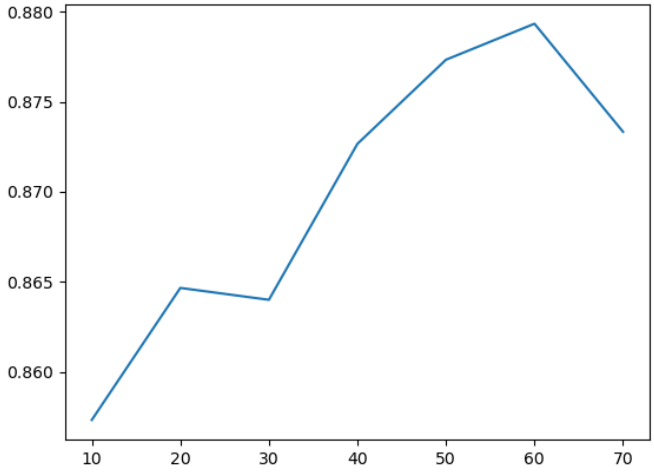
为了避免学习率过大导致在最小值附附近来回震荡，所以加上了学习率递减，我选用库函数的 cosine decay 进行学习率递减，得到的实现效果如下：



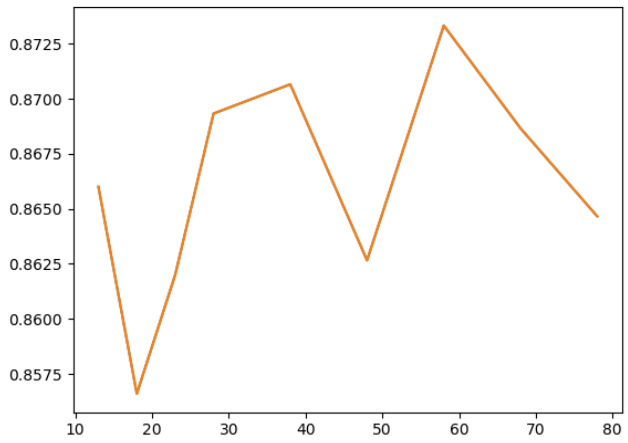


我尝试调整学习率，将学习率设为 0.01 后，再进行了尝试，得

到以下的效果图：



上面的这些效果虽然都有所提升，但在训练过程中，明明验证集上的数据并没有在训练时出现，所以验证集的结果应该跟测试集的结果相近，然而验证集的正确率却是和训练集相近，和测试集差异很大。因此我怀疑是有数据混入，所以另外写了 cutdata.py 脚本，将训练数据和验证数据直接拆分成两个文件，并检查无重复数据后，重新测试得到下图的效果：



五、实验总结：

本次实验主要是利用卷积神经网络，对于 10 类服装图片进行分类，在实验的过程中，我遇到了一个很奇怪的问题，在我事先分割出验证集后，且保证此验证集没有在训练过程中被见过，验证集和训练集的正确率都可以达到 0.99 以上，但是一提交上去后，正确率却连 0.88 都到不了，导致这个现象的原因也不应该是过拟合，因为验证集和训练集是完全分离的，直到交作业的时候，我依旧没有理解为何会出现这样的情况。

经过这一次的实验，我对于各个卷积过程的操作有了深入的了解，也自己尝试将已有的网络，改成符合自己数据格式的网络结构。