**Fashion-MNIST时尚物品图像分类实验报告**

**姓名：方新宇 学号：2052324 学院：软件学院**

一、**实验名称**

Fashion-MNIST时尚物品图像分类实验

二、**实验目的**

基于Fashion-MNIST数据集，利用人工智能导论课上所学相关知识，实现对数据集图像的分类，同时熟悉pytorch这一简洁高效且模块组件化的深度学习框架，为后续进一步学习深度学习相关知识打下良好基础。

**三、实验环境**

操作系统：Ubuntu 18.04.5 LTS

深度学习框架：pytorch 1.10.0

所用编程语言：python 3.7.11

数据集: Fashion-MNIST

其它所用的包：matplotlib 3.4.3（用于绘图）; tqdm 4.62.3 (作为进度条);

**四、框架及相关名词&数据集介绍**

**1.pytorch深度学习框架：**

PyTorch是一个基于Torch的Python开源机器学习库，用于自然语言处理等应用程序。它主要由Facebookd的人工智能小组开发，不仅能够实现强大的GPU加速，同时还支持动态神经网络。

相比其它深度学习框架（如Tensorflow, keras），其主要优点与功能如下：

1.具有强大的GPU加速的张量计算（如Numpy）

2.包含自动求导系统的深度神经网络，使得零延迟地任意改变与优化神经网络结构

3.模块化封装良好，便于调用，同时可进行自定义扩展

**2.Fashion-MNIST数据集：**

Fashion-MNIST是一个替代原始的MNIST手写数字数据集的另一个图像数据集。 它是由Zalando（一家德国的时尚科技公司）旗下的研究部门提供。不同于MNIST手写数据集，Fashion-MNIST数据集包含了10个类别的图像的共7万个不同商品的正面图片，这10个类比分别是：t-shirt（T恤），trouser（牛仔裤），pullover（套衫），dress（裙子），coat（外套），sandal（凉鞋），shirt（衬衫），sneaker（运动鞋），bag（包），ankle boot（短靴）。Fashion-MNIST的大小、格式和训练集/测试集划分与原始的MNIST完全一致。60000/10000的训练/测试数据划分，图像是一个28\*28的像素数组，每个像素的值为0~255之间的8位无符号整数,最后一维表示通道个数。由于为灰度图像，故通道数为1。

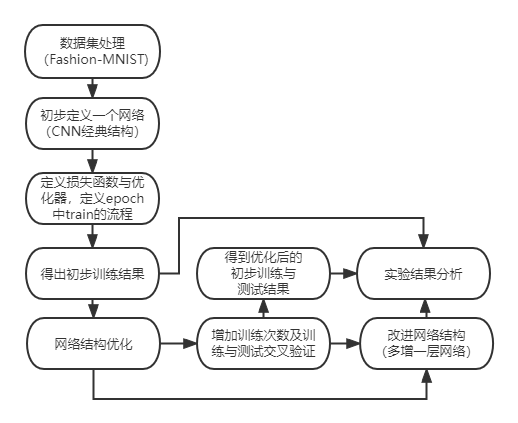
**3.Dropout处理层：**

在深度学习过程中，若模型的参数太多，而训练样本过少，训练出来的模型很容易产生过拟合（指模型在训练数据上损失函数较小，预测准确率较高；但是在测试数据上损失函数比较大，预测准确率较低的现象）。

Dropout可以作为训练深度神经网络的一种技巧。在每个训练批次中，通过让一半的隐层节点值为0，可以明显地减少过拟合现象。这种方式可以减少特征检测器（即隐层节点）间的相互作用，使模型泛化性更强，因为它不会太依赖某些局部的特征。

**五、实验内容**

**5.1实验流程图&实验流程说明**

首先，对数据集进行处理，将其下载至本地并利用torch中的模块进行划分与按batch打包。

其次，初步定义一个网络，并对损失函数，优化器与相关的超参数（记录过程中数据的普通参数）进行初始化。对epoch中train的过程进行设定，并得出初步的训练结果。

其后，对网络结构从训练次数增加&网络结构改进两方面入手进行优化，并使得训练集与测试集交叉验证优化。

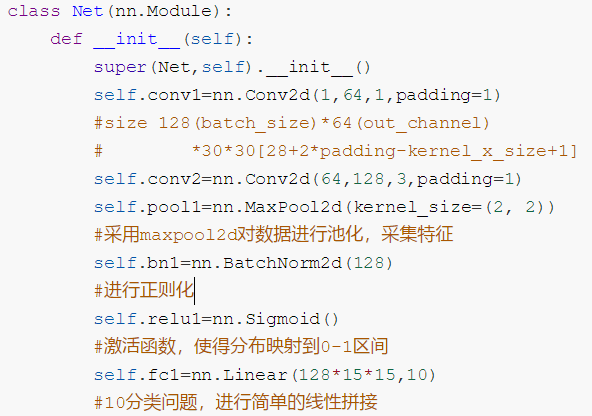
最后，对最终的实验结果进行分析。

**5.2实验步骤**

**（1）数据集处理**

我们通过torchvision的torchvision.datasets包来下载这个数据集，第一次调用时会自动从网上获取数据，放于data目录下。同时根据train参数的设定，可以将数据按6：1比例对训练数据与测试数据进行划分。在下载数据时，也可利用transform对数据进行进一步变换，此处的transform\_train即是利用pytorch中打包好的transform的模块，对训练的图像进行预处理，以给定的概率水平（随机）翻转图像，并将变换后的图像转化为张量，便于后续处理。

在加载完成数据后，我们利用pytorch中的dataloader类将数据装入dataloader中，将batch大小设定为128，便于数据的批处理与训练。同时，对训练的数据集进行shuffle（随机打乱），使得数据随机化,这样可以避免过拟合，让模型泛化能力更强，num\_worker设置为4（一次性创建4个进程，同步进行装载，使得速度更快）。

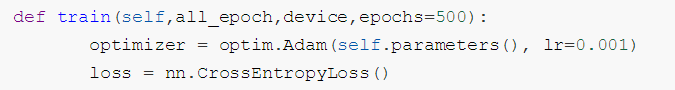
**（2）初步网络定义与搭建**

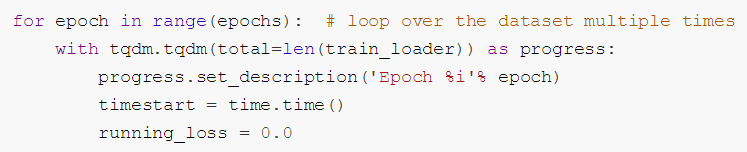
我们利用torch.nn的模块，在此基础上对网络结构进行组件化搭建，对各层网络进行定义。

采用传统的CNN网络结构，即两层卷积层+池化层+正则化+激活函数，对图像的特征进行提取，并转化为将图像特征转化为10个维度（10分类问题），供后续loss计算。

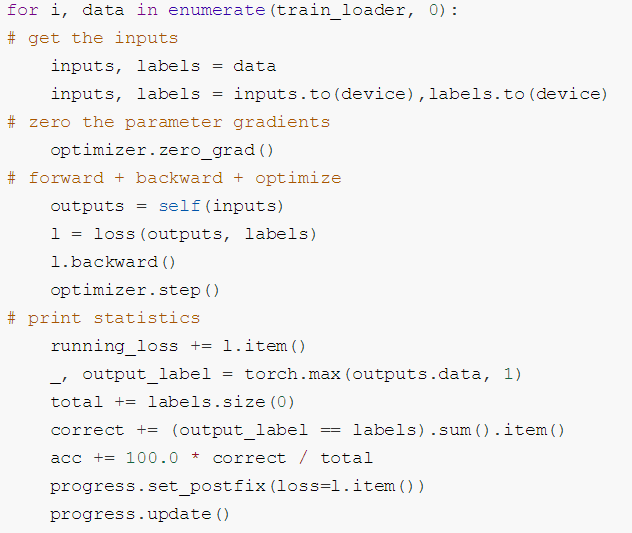
nn.Conv2d(a,b,kenerl\_size),a为输入的向量的通道数，b表示输出的向量的通道数，kenerl\_size若为单值n/数值对(p,q)，则表示卷积核的大小为n\*n/p\*q，padding表示外围填充的宽度。卷积层对输入图片中的信息（即图像特征）进行提取。 nn.maxpool2d(kenerl\_size=n\*m)指对数据进行池化操作，从n\*m的区域中提取最大值，放入结果中，从而对卷积层中提取的特征进行挑选。nn.batchNorm2d（x）指对数据进行正则化操作，x为数据的通道数，对数据做归一化处理,使其分布一致。nn.sigmoid()为激活函数，将数据分布映射到0-1之间，使得输出可以看作概率。nn.Linear(x,y)为全连接层，使得数据从x维度降低到y维度，将所有特征矩阵转化为特征向量，用作最后一层改变分类的个数。因为此处为10分类问题，所以将输出维度设定为10。

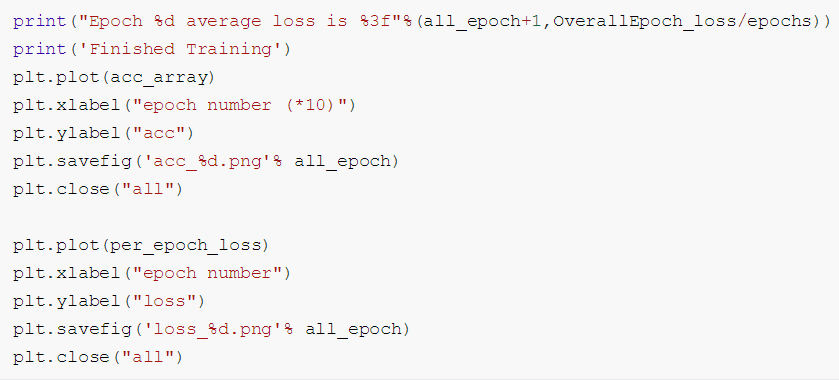
**（3）训练框架说明**

Optimizer选用adam，并将学习率设定为0.001，这是因为若学习率设置过高，易使损失函数直接越过全局最优点，容易发生梯度爆炸，loss 振动幅度较大，模型难以收敛。而设定太小又会导致学习过慢，容易陷入局部最优。而loss函数选用交叉熵损失函数。



对每一个train epoch，首先设置进度条，并对相关参数进行清零操作，防止数据累加引发计算混乱。

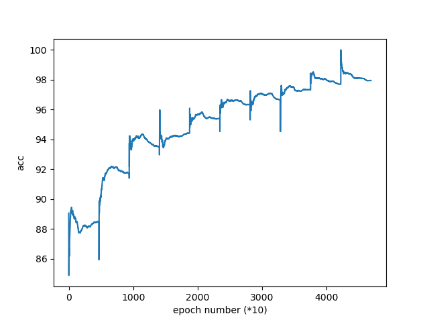
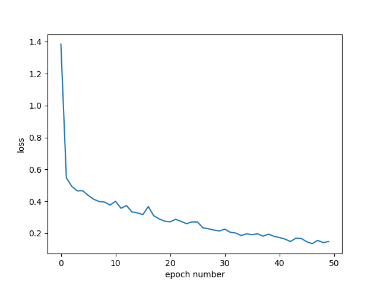
 其次，对epoch中的每一个mini-batch，实现一次迭代（iteration）。从train\_loader中获取input数据与数据的真实label(即ground-truth)，将数据移动到gpu上训练。在前向传播前对梯度清零，避免将多个batch的梯度混合起来累积，导致互相影响。将数据放入网络，进行前向传播后，将得出的结果与label比较，计算loss，并对loss进行反向传播，来实现可训练参数的更新。其后对optimizer执行一次优化步骤,通过梯度下降法来更新参数的值。最后对单个epoch中的accuaracy进行计算，并对进度条进行更新，为下一次iteration做准备。

 在每一个epoch结束后，计算epoch的平均loss,并绘制出这个train epoch中accuracy与loss的变化曲线。

在每一次train后直接进行test,初步设定train时只训练50epoch,以便于代码修改与结果观察。

test是在无梯度计算（with torch.no\_grad()）的情况下，将 model切换到测试模式，对loss与整体的acc进行计算，整体计算流程与train时相同，数据量大小为10000张图片。

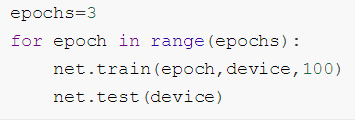
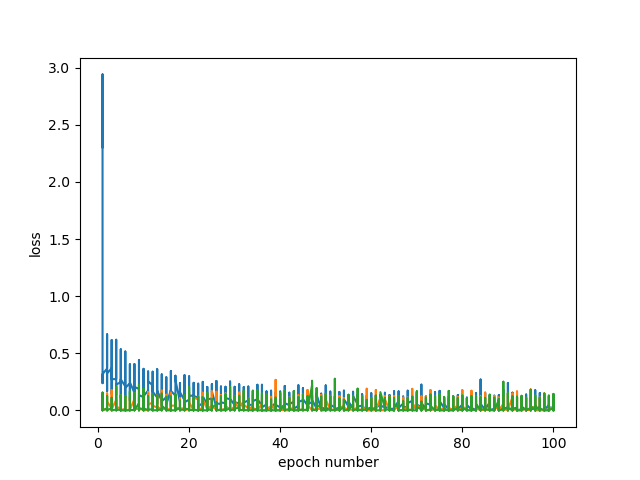
**（4）初步训练结果分析**

 Loss在train的第97个epoch收敛到0.278，测试集整体准确率为90.650 %。在训练过程中的loss与acc（即准确度）变化如下图所示。

可以看到训练过程中，Acc是波动增长的，但每个epoch中仍有预测不准确的iteration。

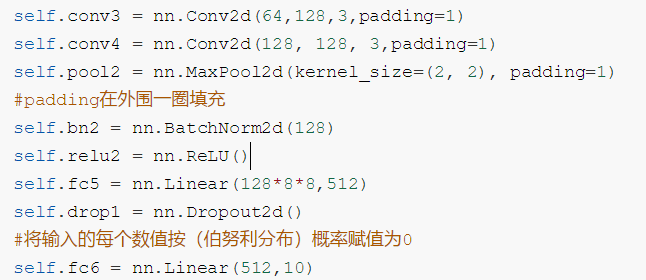
开始的loss仍较大，呈波动收敛的，整体效果好，但仍有不足。

**（5）初步改进--训练次数增加&交叉验证及改进效果分析**

 首先不更改网络结构，只增加训练epoch数为100，同时在每次train后即进行test，便于及时对模型进行评估与调整，总体的epoch（Overall Epoch）数量设定为3。

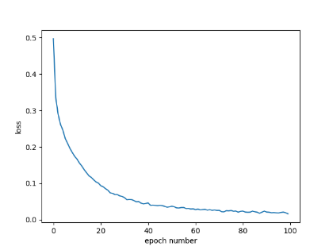
改进后结果相比之前并无过大变化，最大的测试集准确度为90.450%，最低loss为0.012385，均出现在整体的第三个epoch。但有明显改善的是epoch中出现的最小loss,为3.6e-6。右图为3个epoch的loss汇总图，可以看到除了蓝色（第一次训练）开始loss较大外，后面的loss均保持较小值，呈现一个波动，在部分epoch有一定回弹。

**（6）进一步改进—网络层数加深及最终实验效果分析**

 在原先网络基础上，我们新增一层网络，并在池化层后接入Dropout层，让某些神经元的激活值以一定的概率停止工作，这使模型不过度依赖某些局部的特征，泛化性更强，有效缓解过拟合的发生。

同时，我们将train的epoch数目增加为100，保持Overall Epoch数为3，以期待更好的优化效果。训练数据如下：

 可以发现加入多一层神经网络后，效果有明显改善，整体准确度均高于92%，并在第二个循环达到近93%，平均loss均小于0.02。Epoch Average loss呈减小趋势，并在第三个循环与test的average loss数值相近；Epoch Min loss同样也呈现减小趋势，说明模型在整体进行优化，但从出现的epoch不断提前，说明模型中存在优化不当或过度优化。

 整体loss在train的第一个Overall epoch类双曲线光滑下降（如左图），在第二个Overall epoch波动下降，整体数值约束在0.01-0.02。后续将根据模型优化程度，进一步增加网络层数与深度，提升训练-测试循环次数，辅助提升每万张照片预测准确度。代码及结果开源于：<https://github.com/FangXinyu-0913/DeepLearningCourse2022Spring>。

评分：20分

评分人：2054192 张子涵