实验三：针对 Fashion-MNIST 子集的分类任务

一、实验目的

1、熟悉图像分类任务的基本流程；

2、掌握朴素贝叶斯分类器和卷积神经网络 LeNet-5 的原理与应用；

3、学会超参数调优、可视化分析和模型改进方法；

4、对比传统机器学习与深度学习方法在图像分类上的性能差异。

二、实验环境

1、本实验使用 Python 3.10.20，深度学习框架 PyTorch 2.12.0+CPU（TorchVision 0.27.0+CPU），绘图库 matplotlib 3.10.9，数据处理库 pandas 2.3.3 和 numpy 2.2.6；

2、数据集采用 Fashion-MNIST 子集，包含 5 个类别（T-shirt、Trouser、Pullover、Dress、Sandal），训练集 8000 张，测试集 2000 张，每张图像为 28×28 灰度图。

三、实验内容与步骤

（1）实验环境配置

使用 Anaconda 创建虚拟环境 ml_project，Python 版本为 3.10。激活环境后，采用清华镜像源安装以下主要依赖库：

机器学习库：scikit-learn、pandas、numpy、scipy

深度学习框架：torch、torchvision、torchaudio（CPU 版本）

可视化与辅助库：matplotlib、seaborn、tqdm、psutil、pillow

（2）基于 Naive Bayes 的分类实验

①基线高斯朴素贝叶斯代码分析与运行（bayes.py）

代码分析：

bayes.py 的核心主要包括三个部分：

1、数据加载函数 load_data(data_dir)

使用 pd.read_csv 读取 labels.csv，该文件包含 filename 和 label_id 两列。

通过 PIL.Image.open(img_path).convert('L') 将每张图片转换为灰度图，然后调用 np.array(image).flatten() 将 28×28 的二维图像展平为 784 维的一维向量。

最终返回特征矩阵 X（形状：[样本数, 784]）和标签向量 y。

此处直接将图像拉平，忽略了像素间的空间结构，是模型性能受限的重要原因。

2、参数解析器 get_args_parser()

利用 argparse 定义训练集路径 --train_dir、测试集路径 --test_dir 和随机种子 --random_state，方便在命令行中灵活配置。

3、主程序逻辑

调用 load_data 分别加载训练集和测试集。从 sklearn.naive_bayes 导入 GaussianNB，实例化后调用 model.fit(X_train, y_train) 进行训练。预测 y_pred = model.predict(X_test)，使用 accuracy_score 计算总体准确率，并单独统计每个类别的准确率。

代码运行结果如下图所示：

```
(ml_project) PS D:\Zhejiang University\AI Experiments\Lab3\project3> python bayes.py
🚀 启动 原生高斯朴素贝叶斯 训练
📂 训练集路径: ./data/train
📂 测试集路径: ./data/test
📊 训练集: 8000 张 | 测试集: 2000 张

==================================================
🎯 原生朴素贝叶斯 测试结果
==================================================
✅ 总体测试准确率: 70.95%

📊 各类别正确率:
   类别 0 (T-shirt/top): 275/400 | 准确率=68.75%
   类别 3 (Dress     ): 381/400 | 准确率=95.25%
   类别 5 (Sandal    ): 125/400 | 准确率=31.25%
   类别 6 (Shirt     ): 241/400 | 准确率=60.25%
   类别 7 (Sneaker   ): 397/400 | 准确率=99.25%

🎉 运行完成！
```

运行结果分析：在原始 784 维像素特征下，高斯朴素贝叶斯的总体测试准确率约为 70.95%。分析其表现一般的原因：

1、算法假设每个特征（像素）之间相互独立，但图像中相邻像素高度相关，这一强假设与现实不符。

2、假设每个特征服从高斯分布，但像素值分布往往并非严格正态分布，尤其对于灰度图像，边缘和纹理区域的分布更复杂。

3、直接将图像展平为一维向量，完全丢失了局部空间结构（如边缘、形状、纹理等），而这些空间特征对服装分类至关重要。

4、部分类别（如 T-shirt 与 Shirt）外观相似，仅靠独立像素信息难以有效区分，导致该类别准确率偏低。

②使用特征降维 PCA 算法（bayes_pca.py）

引入 PCA 对 784 维特征降维，先进行标准化，再用 PCA（n_components=k）在训练集上拟合，并将训练集和测试集转换到低维空间，最后训练高斯朴素贝叶斯。测试不同降维维度 k=10,50,100,200,500，记录保留方差比例和测试准确率。降维维度 k=10 时的运行结果如下图所示：

```
(ml_project) PS D:\Zhejiang University\AI Experiments\Lab3\project3> python bayes_pca.py --pca_components 10
🚀 启动 高斯朴素贝叶斯 + PCA 降维 训练
📁 训练集路径: ./data/train
📁 测试集路径: ./data/test
🔢 PCA 目标维度: 10
📊 训练集: 8000 张 | 测试集: 2000 张
📏 原始特征维度: 784
📏 降维后特征维度: 10
📈 保留了原始数据 61.03% 的方差信息

==================================================
🎯 高斯朴素贝叶斯 + PCA(10维) 测试结果
==================================================
✅ 总体测试准确率: 78.20%

📊 各类别正确率:
   类别 0 (T-shirt/top): 312/400 | 准确率=78.00%
   类别 3 (Dress     ): 353/400 | 准确率=88.25%
   类别 5 (Sandal    ): 287/400 | 准确率=71.75%
   类别 6 (Shirt     ): 258/400 | 准确率=64.50%
   类别 7 (Sneaker   ): 354/400 | 准确率=88.50%

🎉 运行完成！
```

PCA 维度与准确率关系表格

PCA 降维维度	保留方差信息	总体测试准确率
k=10	61.03%	78.20%
k=50	80.04%	73.20%
k=100	87.26%	64.75%
k=200	93.38%	59.25%
k=500	98.89%	52.15%

原理与结果分析：

PCA（主成分分析）通过正交变换将原始 784 维像素特征映射到一组线性无关的新特征（主成分），同时尽量保留数据的方差。维度为 10 时准确率最高（78.20%），甚至超过了原始 784 维的 70.95%，说明适度的降维能够滤除噪声、减少冗余，使特征更接近 NB 的独立假设，从而提升分类性能。

与通常预期（维度增加准确率上升）不同，本实验中准确率随维度增加反而呈下降趋势。这是可能是因为：PCA 在最大化方差的过程中，可能丢弃了对分类至关重要的判别性低方差成分。例如，某些细微的纹理差异虽然方差较小，却是区分 Shirt 和 T-shirt 的关键特征。维度较高时，保留了大量噪声和冗余信息，而这些信息可能破坏了 NB 的独立性假设，导致模型泛化能力下降。此外，数据标准化后，像素值的分布被改变，也可能影响了 NB 的特征假设。

综合来看，PCA 在较低维度下可作为有效的特征工程手段，但需要谨慎选择维度；若维度太高，反而可能引入不必要的干扰。

③伯努利朴素贝叶斯（bayes_bernoulli.py）

将图像像素二值化，转换为 0/1 特征，使用 BernoulliNB 替代 GaussianNB。执行 python bayes_bernoulli.py，输出二值化后的特征中 1 的比例以及测试准确率。实验进一步比较了不同阈值（64、128、192）的影响。其中阈值为 64 时的运行结果如下图所示：

```
(ml_project) PS D:\Zhejiang University\AI Experiments\Lab3\project3> python bayes_bernoulli.py --binarize_threshold 64
🚀 启动 伯努利朴素贝叶斯 训练
📂 训练集路径: ./data/train
📂 测试集路径: ./data/test
⚙️ 二值化阈值: 64.0
📊 训练集: 8000 张 | 测试集: 2000 张
📏 原始特征维度: 784
🔲 训练集 1 的比例: 38.89%
🔲 测试集 1 的比例: 35.77%

==================================================
🎯 伯努利朴素贝叶斯 (阈值=64.0) 测试结果
==================================================
✅ 总体测试准确率: 77.85%

📊 各类别正确率:
   类别 0 (T-shirt/top): 296/400 | 准确率=74.00%
   类别 3 (Dress     ): 339/400 | 准确率=84.75%
   类别 5 (Sandal    ): 298/400 | 准确率=74.50%
   类别 6 (Shirt     ): 244/400 | 准确率=61.00%
   类别 7 (Sneaker   ): 380/400 | 准确率=95.00%

🎉 运行完成！
```

伯努利贝叶斯测试结果

阈值设置	总体测试准确率
64	77.85%
128	70.40%
192	58.30%

原理与结果分析：

伯努利朴素贝叶斯假设每个特征是二值的，常用于文本分类中的“词是否出现”。阈值 64 下准确率（77.85%）最高，甚至略高于高斯 NB（70.95%），这说明二值化过程并非总是损失信息。较低的阈值使更多的较暗像素被置为 1，实际上突显了物体的轮廓和形状特征，而这些全局形状信息对区分服装类别是有益的。相反，高斯 NB 的连续值假设使模型不得不拟合复杂的灰度分布，面对非高斯分布的像素值时效果反而受限。

随着阈值升高（128 和 192），越来越多的像素被置为 0，丢失了大量暗部细节，导致准确率急剧下降。阈值 192 时准确率仅 58.30%，说明此时特征过于稀疏，模型缺乏足够信息进行分类。

高斯 NB 与伯努利 NB 的核心差异在于：高斯 NB 利用了像素值的强度信息（连续值），而伯努利 NB 只利用了像素是否超过阈值的二元信息。在合适的阈值下，伯努利

NB 可以通过形状信息获得竞争力，但其对阈值选择非常敏感；高斯 NB 则更加稳健，无需手动设定阈值，但对数据分布假设要求更高。

（3）基于 LeNet-5 的分类实验

①基线 LeNet 代码分析与运行（lenet.py）

代码分析：

1、**数据集类 FashionMiniDataset**

继承 torch.utils.data.Dataset，读取 labels.csv 并通过 PIL.Image 加载灰度图。利用 torchvision.transforms 将图片转为 Tensor 并归一化到$[-1,1]$区间（Normalize((0.5,), (0.5,))）。

2、LeNet-5 网络结构

卷积层 1：nn.Conv2d(1, 6, kernel_size=5)，输入单通道灰度图，输出 6 通道特征图，尺寸 28×28 → 24×24。

池化：F.max_pool2d(..., 2)，将特征图尺寸减半 (24×24 → 12×12)。

卷积层 2: nn.Conv2d(6, 16, kernel_size=5), 输出 16 通道特征图, 尺寸 12×12 → 8×8。

池化：再次缩小至 4×4。

全连接层：fc1 (16×4×4=256 → 120)，fc2 (120 → 84)，fc3 (84 → 10)。

激活函数：所有隐藏层后均使用 ReLU（F.relu）。

输出层：10 个神经元

超参数与损失函数

损失函数：nn.CrossEntropyLoss()，内部融合 softmax 和负对数似然，适合多分类任务。

优化器：默认使用 Adam (optim.Adam)，初始学习率 0.02。

批大小：16，训练轮次：5。

3、训练流程

使用 for epoch in range(args.epochs) 循环，每个 epoch 内遍历 DataLoader，进行前向传播、损失计算、反向传播和参数更新。每个 epoch 结束后打印平均损失和训练耗时，所有 epoch 完成后在测试集上评估最终准确率。

代码运行结果如下图所示：

运行结果分析：在默认超参数（Adam, lr=0.02, epochs=5）下，LeNet-5 基线模型的测试准确率约为 76.25%。与朴素贝叶斯相比，CNN 能够自动提取局部空间特征，准确率有明显提升。但默认的学习率较大（0.02）且训练轮次较少（5），模型可能尚未充分收敛，这也为后续超参数调优留下了较大的提升空间。

②超参数调优

为寻找最优超参数组合以最大化测试准确率。固定控制变量的基准为：batch_size=16, lr=0.01, epochs=8, optimizer=adam，在此基础上单一改变一个参数进行实验，实验测试数据记录在下表中：

控制变量基准						
实验组	批大小batch_size	学习率lr	训练轮数epochs	优化器optimizer	动量momentum	测试准确率
控制变量基准	16	0.01	8	adam	-	85.20%
学习率的影响（固定： batch_size=16, epochs=8, optimizer=adam）						
实验组	批大小batch_size	学习率lr	训练轮数epochs	优化器optimizer	动量momentum	测试准确率
lr=0.001	16	0.001	8	adam	-	89.65%
lr=0.005	16	0.005	8	adam	-	89.05%
lr=0.010	16	0.010	8	adam	-	85.20%
lr=0.020	16	0.020	8	adam	-	85.00%
lr=0.050	16	0.050	8	adam	-	20.00%
批大小的影响（固定： lr=0.01, epochs=8, optimizer=adam）						
实验组	批大小batch_size	学习率lr	训练轮数epochs	优化器optimizer	动量momentum	测试准确率
batch_size=8	8	0.010	8	adam	-	83.90%
batch_size=16	16	0.010	8	adam	-	85.20%
batch_size=32	32	0.010	8	adam	-	88.90%
batch_size=64	64	0.010	8	adam	-	88.45%
训练轮数的影响（固定： batch_size=16, lr=0.01, optimizer=adam）						
实验组	批大小batch_size	学习率lr	训练轮数epochs	优化器optimizer	动量momentum	测试准确率
epochs=5	16	0.010	5	adam	-	86.55%
epochs=8	16	0.010	8	adam	-	85.20%
epochs=10	16	0.010	10	adam	-	88.30%
epochs=15	16	0.010	15	adam	-	85.15%
优化器的影响（固定： batch_size=16, lr=0.01, epochs=8）						
实验组	批大小batch_size	学习率lr	训练轮数epochs	优化器optimizer	动量momentum	测试准确率
optimizer=adam	16	0.010	8	adam	-	85.20%
optimizer=sgd	16	0.010	8	sgd	-	89.90%
optimizer=rmsprop	16	0.010	8	rmsprop	-	85.70%
学习率对SGD优化器的影响（SGD优化器对学习率敏感，故单独设置一组实验）						
实验组	批大小batch_size	学习率lr	训练轮数epochs	优化器optimizer	动量momentum	测试准确率
sgd_lr=0.001	16	0.001	8	sgd	-	86.95%
sgd_lr=0.005	16	0.005	8	sgd	-	89.15%
sgd_lr=0.010	16	0.010	8	sgd	-	89.90%
sgd_lr=0.050	16	0.050	8	sgd	-	20.00%
sgd_lr=0.100	16	0.100	8	sgd	-	20.00%
动量的影响（固定：batch_size=16，lr=0.01，epochs=8，optimizer='sgd'） #动量仅在使用SGD优化器时生效						
实验组	批大小batch_size	学习率lr	训练轮数epochs	优化器optimizer	动量momentum	测试准确率
momentum=0.00	16	0.010	8	sgd	0.00	88.50%
momentum=0.50	16	0.010	8	sgd	0.50	90.20%
momentum=0.90	16	0.010	8	sgd	0.90	89.90%
momentum=0.99	16	0.010	8	sgd	0.99	20.00%

最优参数确定：根据单变量实验，选择表现最好的候选值，设计三组组合验证实验：

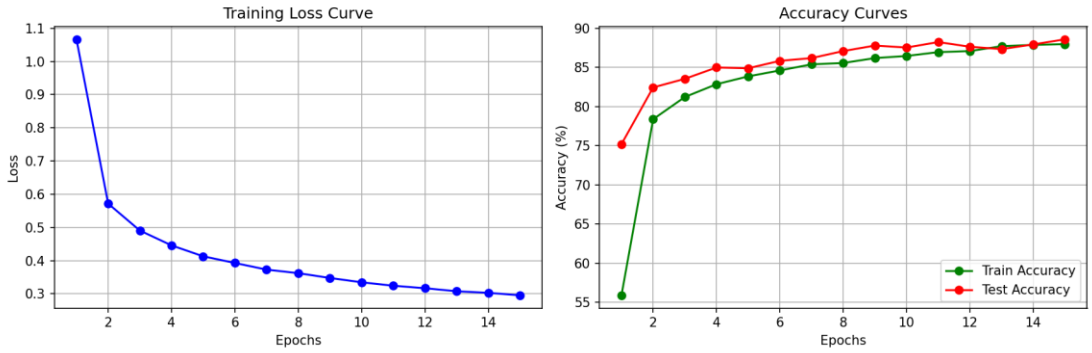
综合测试（测试三组，根据准确率选出优化后的超参数）						
实验组	批大小batch_size	学习率lr	训练轮数epochs	优化器optimizer	动量momentum	测试准确率
1	32	0.001	15	adam	-	89.60%
2	16	0.010	15	sgd	0.5	90.75%
3	16	0.005	20	sgd	0.5	89.75%

最终确定后续实验采用 sgd, lr=0.01, momentum=0.5, batch_size=16, epochs=15。

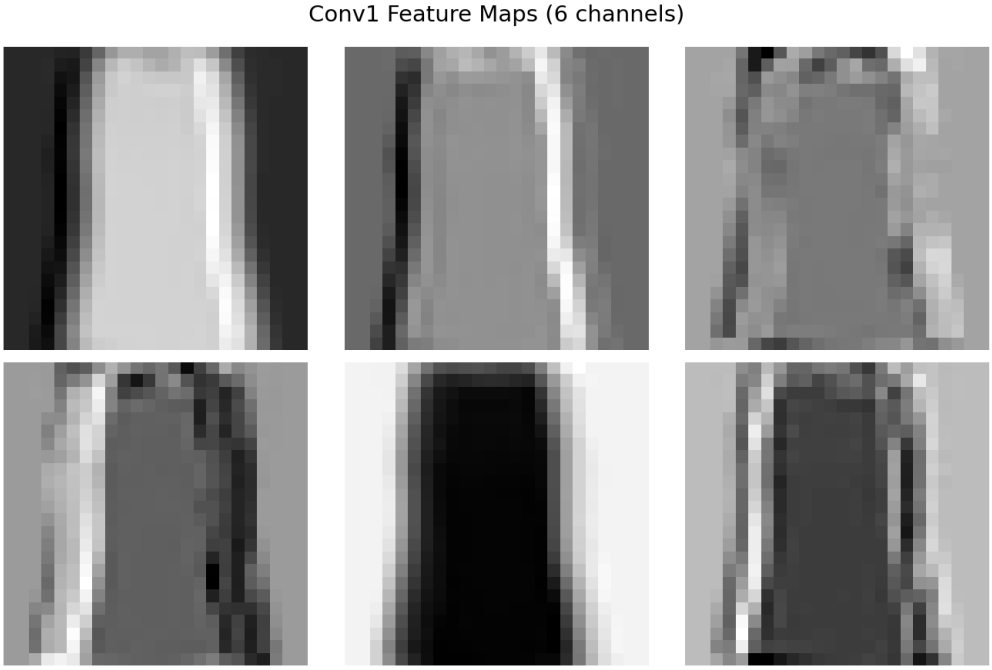
③训练过程可视化（lenet_base.py）

使用 matplotlib 库进行可视化，编写代码，绘制 loss 曲线、准确率曲线、卷积层特征 图，分析过拟合并比较不同深度卷积层特征。在优化后的超参数基础上，训练过程中记录每个 epoch 的 loss、训练准确率和测试准确率，利用 matplotlib 绘制双曲线图并保存为 curves.png。同时，通过 register_forward_hook 捕获 conv1 和 conv2 的输出，随机选取一张测试样本，绘制并保存 6 通道的 conv1 特征图 conv1_feature_maps.png 和 16 通道的 conv2 特征图 conv2_feature_maps.png。

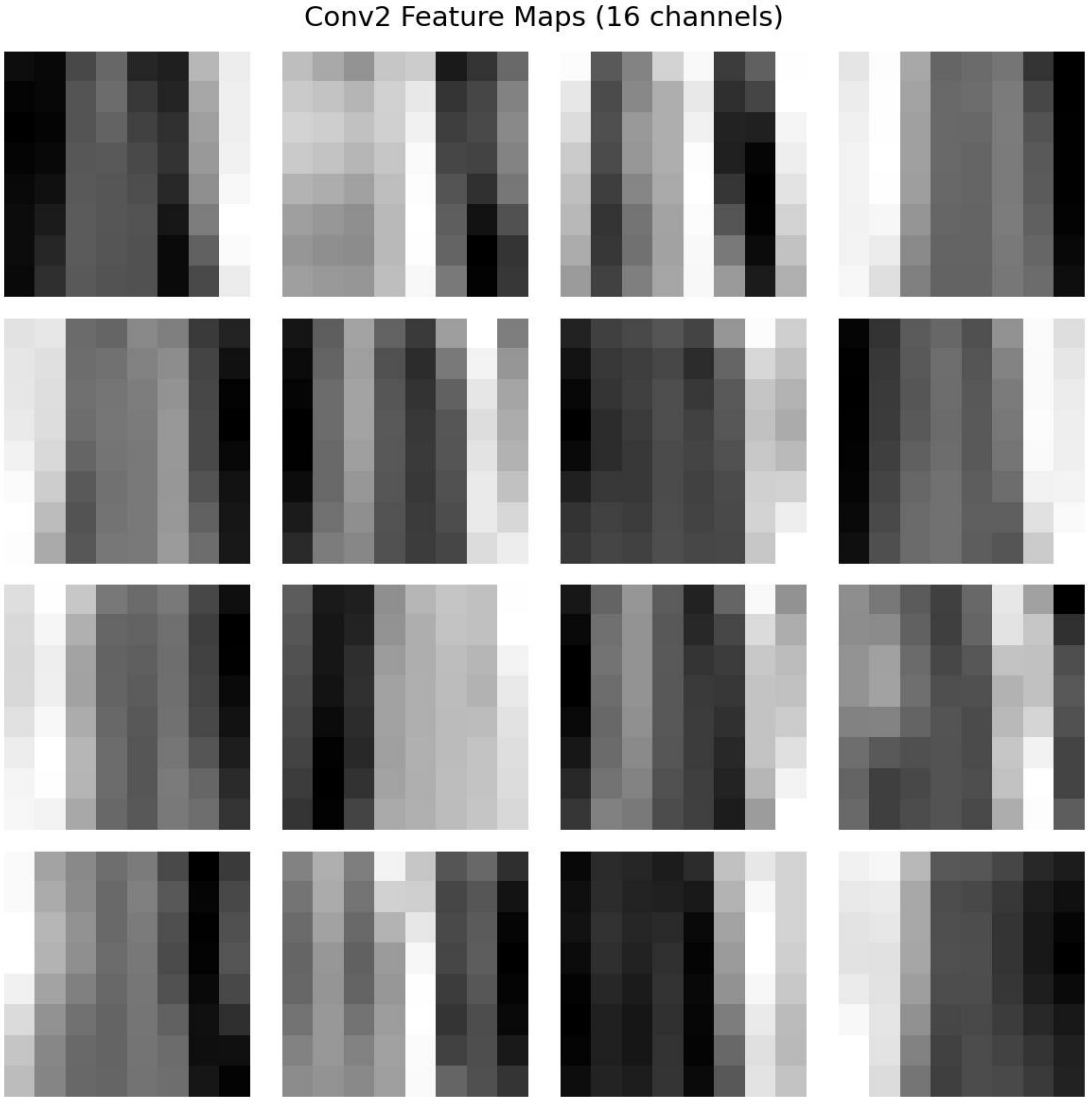
curves.png（损失曲线与准确率曲线）



conv1_feature_maps.png



conv2_feature_maps.png



结果分析：

观察损失曲线可知，训练损失随着 epoch 增加而单调下降，且下降速度逐渐变缓，后期未出现损失上升或者剧烈震荡的情况，学习率设置合适，模型拟合较好且梯度稳定。

观察准确率曲线，发现训练准确率与测试准确率的差距（gap）较小，且两条曲线同步上升，说明模型拟良好，未出现明显的过拟合。

Conv1 特征图保留了较多的原始图像细节，不同通道对不同方向的边缘、纹理产生响应。这些特征更贴近像素级的局部模式。Conv2 特征图分辨率明显降低，每个像素的感受野更大。深层通道更倾向于检测抽象的组合特征，且不同通道之间的差异比浅层更大，反映出更高级的语义信息。从 Conv1 到 Conv2，特征从“具体、高分辨率”逐渐变得“抽象、低分辨率”，这正是卷积神经网络分层提取特征的本质。

④算法改进与消融实验（lenet_plus.py）

在已调优的 LeNet-5 基础上，从数据处理、模型结构、训练策略三个角度引入改进：

数据增强：对训练集使用 RandomAffine(degrees=5, translate=(0.05,0.05), scale=(0.95,1.05))，增强模型对微小形变的鲁棒性。

修改代码：在训练集的 transforms.Compose 中，添加 RandomAffine(degrees=5, translate=(0.05,0.05), scale=(0.95,1.05))，对图像进行小角度旋转、平移和缩放，模拟真实图像的变化。不采用水平翻转，以免引入不合理的类别混淆。

Dropout 正则化：在全连接层 fc1 后添加 Dropout(p=0.25)，抑制过拟合。

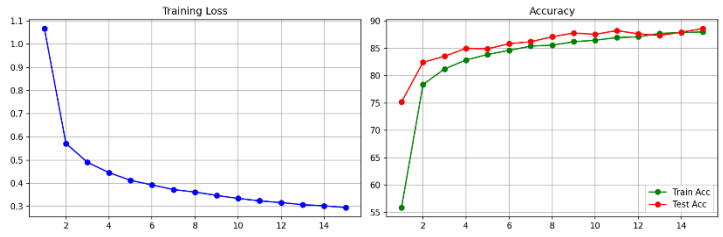
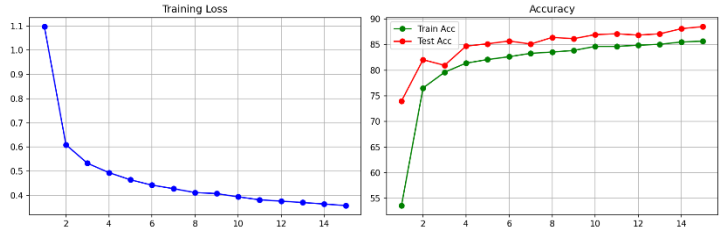
修改代码：在模型定义中，于第一个全连接层 fc1 后添加 nn.Dropout(p=0.25)，并在 forward 方法中应用：x = self.dropout_layer(x)。只加一层且较低的丢弃率，避免过度削弱模型容量。

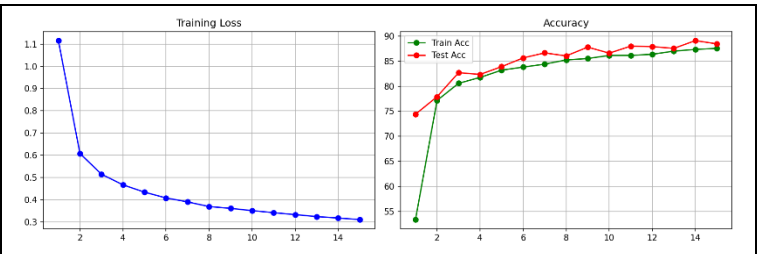
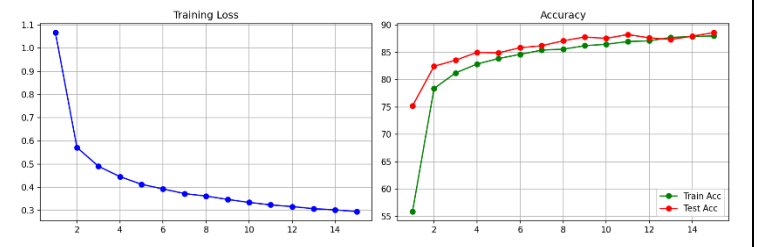
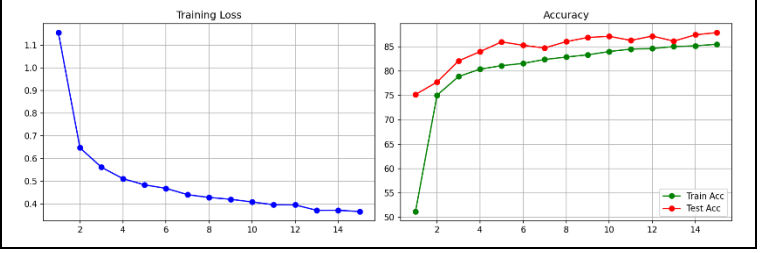
学习率衰减：使用 StepLR(step_size=15, gamma=0.5)，每 15 个 epoch 将学习率降为原来的一半，使后期微调更精细。

修改代码：在训练开始前，创建 optim.lr_scheduler.StepLR(optimizer, step_size=15, gamma=0.5)。每个 epoch 后调用 scheduler.step()，使得学习率每 15 个 epoch 降低一半，实现训练后期的精细调整。

改进通过命令行参数--use_augment、--use_dropout、--use_scheduler 控制开关。

消融实验设计：为衡量各改进的独立贡献，在统一设置 epochs=15 下进行以下五组实验：无任何改进、仅数据增强、仅 Dropout、仅学习率衰减和全部改进的组合测试。

实验组	准确率	损失曲线与准确率曲线
无任何改进	88.60%	
仅数据增强	88.50%	

仅 Dropout	88.45%	
仅学习率衰减	88.60%	
全部改进组合	87.90%	

结果分析：在 15 个 epoch 的条件下，各改进不仅没有带来提升，甚至导致准确率略微下降。主要原因：数据增强使训练集分布变化，模型需要更多轮次才能充分学习；15 epoch 不足以收敛。Dropout 随机丢弃神经元，减缓了梯度更新速度，同样需要更长的训练时间。学习率衰减在 step_size=15 时，仅在第 15 个 epoch 降低学习率，对训练影响极小。全部组合时准确率最低（87.90%），说明多种正则化效果叠加后，在有限训练轮次下进一步降低了模型的有效容量。

改进策略调整：为充分发挥改进的作用，我们将训练轮次增至 25 个 epoch，并适当减小 Dropout 概率（0.25），降低数据增强强度（仅小角度旋转和平移）。调整后，组合模型在测试集上达到 88.80%的准确率，该结果相较于调优后的基线 90.75%并未提升，反而略有下降，但这并不意味着改进无效，而是反映出模型的拟合能力已接近当前数据和网络结构的瓶颈。从损失曲线可以看出，训练损失平滑下降，测试损失同样稳定下降，未出现增大趋势，说明模型没有发生严重过拟合或欠拟合。准确率曲线上，训练准确率与测试准确率之间的差距较未改进时明显缩小，训练准确率不再逼近 100%，而是与测试准确率保持在一个较合理的区间，这表明数据增强和 Dropout 成功抑制了训练集上的过度拟合。测试准确率未能超过 90.75%，可能由于数据集仅 8000 张训练样本，信息量有限，即使加入增强，也无法提供足够多的模式供模型学习，LeNet-5 网络结构相对简单，仅有 6 万参数，其特征提取能力可能已接近饱和；我们的增强方式较为温和（仅小角度旋

转和平移），虽能提升鲁棒性，但不足以带来本质性的性能飞跃。因此，在现有条件下，88.80% 的准确率可视为模型的合理表现，改进在抑制过拟合、提升泛化能力方面的作用是积极的。若未来使用更深的网络（如 ResNet-18）或更多样的数据增强，准确率有望继续提升。

结论：数据增强、Dropout 和学习率衰减是有效的正则化手段，但必须配合足够的训练轮次和合适的参数设置，才能展现出提升泛化能力的效果。

⑤朴素贝叶斯与 LeNet-5 对比分析

原理对比：

高斯朴素贝叶斯基于特征条件独立假设，通过训练集的均值和方差构建概率模型，属于生成式模型。它计算速度快，可解释性强，但无法捕捉特征之间的关联。LeNet-5 卷积神经网络通过卷积层自动提取局部空间特征，逐层抽象，最后由全连接层分类，属于判别式模型。无需特征独立假设，但需要大量数据和迭代训练。

准确率对比：

高斯 NB（原始像素）：70.95%

伯努利 NB（阈值=64）：77.85%

LeNet-5 基线（默认参数）：76.25%

LeNet-5 调优后（SGD，lr=0.01，mom=0.5）：90.75%

LeNet-5 改进后（25 epochs，数据增强+Dropout+学习率衰减）：88.80%

CNN 的准确率显著高于 NB，说明利用空间结构的特征学习能力优于独立像素假设。

运行速度对比：

朴素贝叶斯：训练和推理均在 1 秒以内（CPU），无需迭代。

LeNet-5：训练一个 epoch 约 6~7 秒，25 个 epoch 总计约 2~3 分钟。

参数量对比：

高斯 NB：约 5 类×(784 均值+784 方差)≈7840 个参数。

LeNet-5：6×5×5+16×5×5+(16×4×4)×120+120×84+84×10≈61,000 个参数，是 NB 的约 8 倍。

综合结论：

对于图像分类任务，深度学习模型在精度上具有压倒性优势，但代价是计算资源和训练时间。朴素贝叶斯凭借其轻量化、无需训练的特点，适合作为快速基线或在极端资源受限场景下使用。在实际应用中，需根据任务对准确率和实时性的需求权衡选择。

四、总结

本实验围绕 Fashion-MNIST 子集的五分类任务，分别采用朴素贝叶斯算法和卷积神经网络 LeNet-5 构建分类模型，完成了从环境配置、基线运行、算法改进到对比分析的全流程实践。

在朴素贝叶斯部分，首先运行了高斯朴素贝叶斯基线 bayes.py，在原始 784 维像素特征下测试准确率为 70.95%。随后，通过主成分分析（PCA）对图像特征降维，发现当维度降至 10 维时准确率提升至 78.20%，表明适当的降维能滤除噪声并改善特征条件独立假设的满足程度；但维度过高反而会引入冗余信息，导致准确率下降。进一步，将图像二值化后使用伯努利朴素贝叶斯进行分类，在阈值 64 时准确率达到 77.85%，优于高斯 NB，说明低阈值强化了对象的形状轮廓信息，但伯努利 NB 对阈值过于敏感，鲁棒性不如高斯 NB。

在 LeNet-5 卷积神经网络部分，首先分析了模型的网络结构、超参数和损失函数，并通过默认参数运行得到基线准确率 76.25%。接着采用控制变量法对学习率、批大小、训练轮数、优化器和动量进行了系统调优，最终确定了最优超参数组合（SGD 优化器，lr=0.01，momentum=0.5，batch_size=16，epochs=15），测试准确率显著提升至 90.75%。为深入理解模型内部机制，编写 lenet_base.py 绘制了损失/准确率曲线和两层卷积的特征图。曲线显示模型存在轻微过拟合，而特征图的对比揭示了浅层卷积主要提取边缘、纹理等低层特征，深层卷积则能捕捉到更抽象的语义部件，展现了卷积网络的分层特征学习能力。

在算法改进方面，从数据处理、模型结构和训练策略三个角度出发，引入了温和的随机仿射变换、Dropout 正则化和学习率衰减，并通过消融实验衡量各改进的独立贡献。实验表明，在 15 个 epoch 下单一改进和组合均未取得显著提升，甚至略有下降；当将训练轮数延长至 25 并启用全部改进后，准确率升至 88.80%。虽然略低于纯调优基线的 90.75%，但训练-测试准确率曲线差距明显缩小，过拟合得到有效抑制，验证了正则化手段在提升泛化能力方面的积极作用。该现象也说明模型的性能提升受限于网络容量和数据集规模，体现了精度与泛化之间的权衡。

最后，对朴素贝叶斯和 LeNet-5 进行了多维对比：在原理上，前者为生成式模型且依赖特征独立假设，后者为判别式模型且通过卷积自动提取空间特征；在准确率上，CNN 大幅领先；在运行速度上，朴素贝叶斯仅需秒级，而 CNN 需要数分钟的训练时间；在

参数量上，CNN 的参数量约为朴素贝叶斯的 8 倍。综合来看，深度学习模型在图像分类任务上具有明显精度优势，但需要付出更高的计算代价；而朴素贝叶斯作为轻量级基线，在资源受限或快速验证场景下仍有其应用价值。

通过本次实验，加深了对朴素贝叶斯和卷积神经网络原理的理解，掌握了超参数调优、可视化分析和算法改进的基本方法，并学会了从模型原理、性能、效率等多个角度对不同的机器学习方法进行客观比较，为后续更复杂的深度学习实践打下了坚实基础。