基于卷积神经网络的手写数字识别

|  |  |
| --- | --- |
| 姓名： 张彦博 | 学号：2022116481 |

# 引言

## 1.1实验内容：

基于 PyTorch 框架搭建多层卷积神经网络（CNN）实现手写数字识别，对比逻辑回归、决策树、支持向量机（SVM）三种传统机器学习方法，分析不同模型在精确率、召回率、F1 值等指标上的表现，并探究学习率和优化器对 CNN 性能的影响。

## 1.2基础知识介绍（主要针对于卷积神经网络介绍）：

卷积神经网络主要有两个特性：

①对图像的平移不变性translation invariance）：不管检测对象出现在图像中的哪个位置，神经网络的前面几层应该对相同的图像区域具有相似的反应，即为“平移不变性”。

②局部性（locality）：神经网络的前面几层应该只探索输入图像中的局部区域，而不过度在意图像中相隔较远区域的关系，这就是“局部性”原则。最终，可以聚合这些局部特征，以在整个图像级别进行预测。

可以先通过数学公式进行理解

其中分别表示输入图像和隐藏表示中位置（i,j）处的像素。[V]被称为卷积核（convolution kernel）或者滤波器（filter），亦或简单地称之为该卷积层的权重，通常该权重是可学习的参数。 当图像处理的局部区域很小时，卷积神经网络与多层感知机的训练差异可能是巨大的

**总结：**

①图像的平移不变性使我们以相同的方式处理局部图像，而不在乎它的位置。

②局部性意味着计算相应的隐藏表示只需一小部分局部图像像素。

③在图像处理中，卷积层通常比全连接层需要更少的参数，但依旧获得高效用的模型。

# 2.卷积神经网络模型结构介绍

## 2.1模型结构：

|  |  |  |  |
| --- | --- | --- | --- |
| **层类型** | **参数配置** | **核心作用** | **设计特点** |
| 输入层 | 单通道灰度图像：MNIST(1\*28\*28) | 接收28×28像素的MNIST手写数字图像 | 原始数据输入，N为批次大小 |
| 卷积层1 | in=1,out=32,kernel=3, padding=1 | 提取初级特征（边缘、线条等） | 小卷积核精细捕捉局部特征 |
| 池化层1 | kerel=2,stride=2 | 空间降维，保留显著特征 | 最大池化增强平移不变性 |
| ReLU激活 | - | 引入非线性，加速收敛 | 解决梯度消失问题 |
| 卷积层2 | in=32,out=64,kernel=3, padding=1 | 提取高级特征（纹理、复杂形状） | 通道数翻倍提升特征复杂度 |
| 池化层2 | kernel=2,stride=2 | 二次降维，压缩信息量 | 减少后续计算量 |
| ReLU激活 | - | 增强特征表达能力 | 与卷积层协同工作 |
| 展平操作 | 64×7×7 = 3136 | 三维特征转一维向量 | 适配全连接层输入格式 |
| 全连接层 | in=3136, out=10 | 输出10个数字类别的概率分布 | 最终分类决策层 |

## 2.2损失函数与优化器

**损失函数介绍：**

交叉熵损失函数（CrossEntropyLoss），适用于多分类任务，计算公式为：Loss= 其中，yic为真实标签（one-hot 编码），为模型预测概率。

**优化器：**

Adam(自适应矩估计)：结合动量（一阶矩）和RMSprop（二阶矩）

SGD:每次使用一个batch的数据计算梯度，直接沿梯度反方向更新参数

RMSprop (均方根传播):自适应调整学习率，通过梯度平方的移动平均调整步长

# 实验设计

## 3.1数据集与预处理

**数据集说明：**

MNIST数据集(包含60,000张训练图像和10,000张测试图像，每张为28×28的灰度手写数字（0-9）)

**归一化说明：**

①输入图像原始像素值范围为[0,255]，ToTensor()将其缩放到[0,1]。

②通过均值0.5、标准差0.5归一化到[-1,1]

**数据加载：**

①batch\_size=64：平衡内存消耗与梯度更新效率

②shuffle=True：仅对训练集打乱顺序，防止批次间相关性影响(仅训练集)

实验测得数据加载耗时：5.43秒

## 3.2评价指标介绍

在分类任务中，我们采用以下核心指标进行模型性能评估：

| **指标名称** | **计算公式** | **解释说明** |
| --- | --- | --- |
| 准确率(Accuracy)​​ | (TP+TN)/(TP+TN+FP+FN) | 总体预测正确的比例 |
| 精确率(Precision)​​ | TP/(TP+FP) | 预测为正例中实际为正的比例 |
| 召回率(Recall)​​ | TP/(TP+FN) | 实际为正例中被正确预测的比例 |
| ​F1值​ | 2\*(Precision\*Recall)/(Precision+Recall) | 精确率和召回率的调和平均 |

​注：

​Macro平均：各类别指标的平均值（平等看待每个类别）

​Micro平均：按样本量加权的全局平均（更关注大类别）

本次实验我主要使用Macro平均

## 3.3 实验设置以及对比方法介绍

**I.CNN超参数实验：**

①学习率（0.001 vs 0.01 vs 0.1）

| **参数组** | **固定参数** | **变量参数** | **评估指标** |
| --- | --- | --- | --- |
| 学习率 | 优化器=SGD, epochs=20 | lr∈{0.001, 0.01, 0.1} | 训练损失曲线、准确率、精确率、召回率、F1值 |

②优化器（Adam vs SGD vs RMSPROP）

| **参数组** | **固定参数** | **变量参数** | **评估指标** |
| --- | --- | --- | --- |
| 优化器组 | lr=0.001, epochs=50 | optimizer∈{Adam,SGD,RMSprop} | 训练损失曲线、准确率、精确率、召回率、F1值、训练时间 |

**Ⅱ.CNN与其他模型比较：**

①逻辑回归模型

②决策树模型

③高斯朴素贝叶斯模型

④单层全连接网络模型（输入：784，输出：10）

| 参数组 | 评估指标 | 生成报告 |
| --- | --- | --- |
| CNN  逻辑回归  贝叶斯  决策树  全连接 | 准确率、精确率、召回率、F1值、训练时间 | 分类报告：生成对每一类标签的分类结果的指标（准确率、精确率、召回率、F1值） |

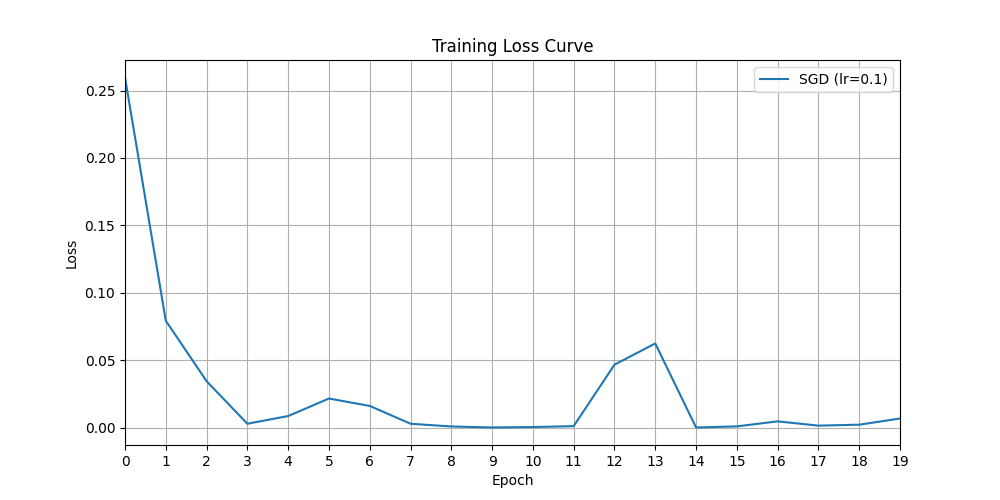
# 实验结果与分析

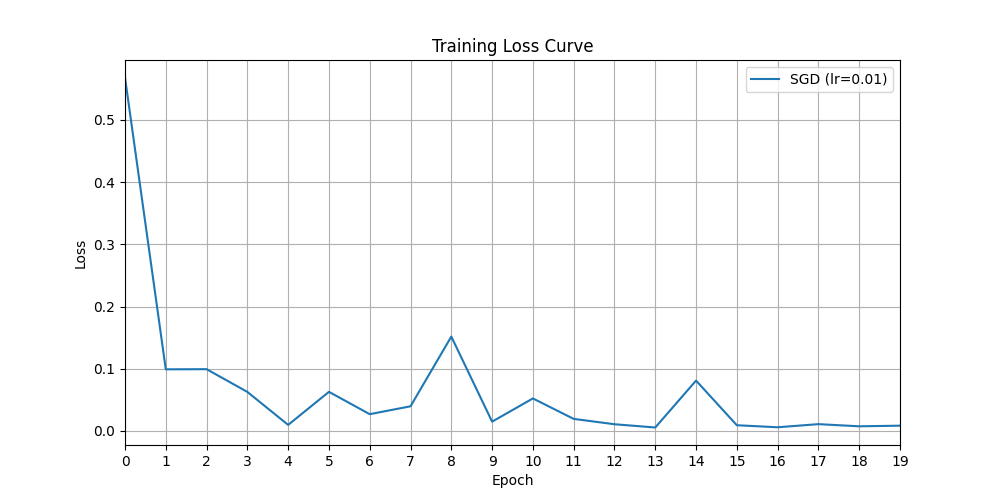
## CNN模型性能

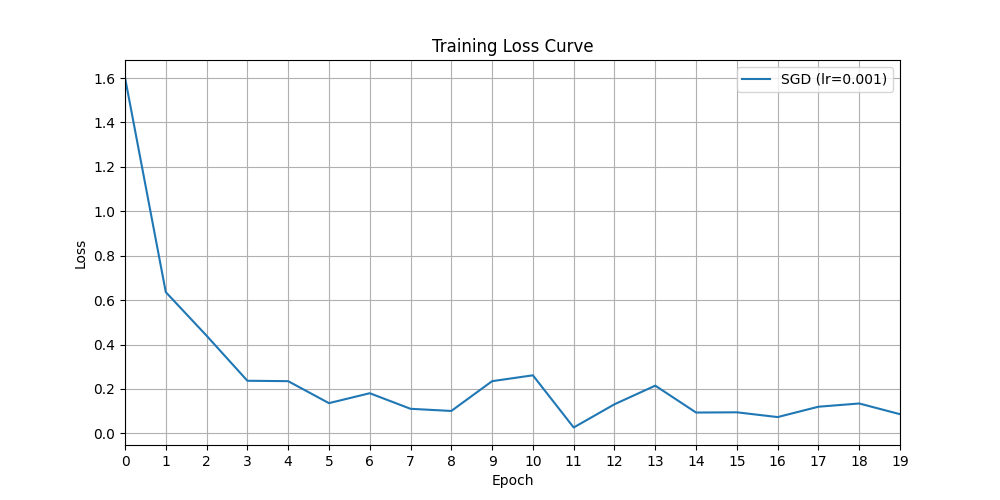
I与Ⅱ比较不同超参数组合下的指标对比：

### I.学习率对模型的影响：参数(SGD 优化器，epochs=20)

| **学习率** | **训练损失** | **测试准确率** | **测试精确率** | **测试召回率** | **测试 F1 值** |
| --- | --- | --- | --- | --- | --- |
| 0.001 | 0.086261 | 0.9673 | 0.9673 | 0.9672 | 0.9672 |
| 0.01 | 0.008448 | 0.9866 | 0.9866 | 0.9865 | 0.9865 |
| 0.1 | 0.006795 | 0.9897 | 0.9896 | 0.9896 | 0.9896 |







**分析：**

学习率越大，训练越快，但波动越大（0.1的曲线在第12 epoch出现明显反弹），这是由于lr过大，造成loss振荡

学习率越小，训练越稳，但可能欠拟合（0.001的最终损失和准确率均较差），尤其是在训练轮次较少的情况下

**结论：**

推荐在训练轮次较少(epoch<=30)的情况下使用：lr = 0.01

在训练轮次足够的情况下可以使用:lr = 0.001

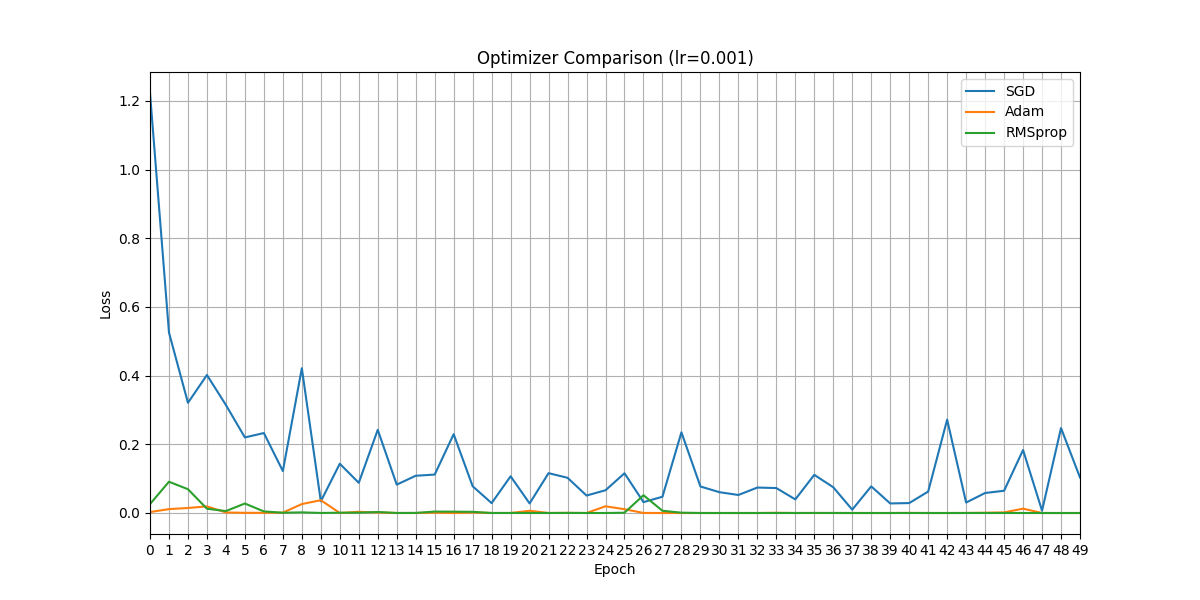
### Ⅱ.优化器对稳定性的影响:参数(学习率：0.001，epochs=50)

| **优化器** | **训练损失** | **测试准确率** | **测试精确率** | **测试召回率** | **测试 F1 值** |
| --- | --- | --- | --- | --- | --- |
| Adam | 0.000000 | 0.9915 | 0.9914 | 0.9914 | 0.9914 |
| SGD | 0.103293 | 0.9809 | 0.9810 | 0.9809 | 0.9809 |
| RMSPROP | 0.000000 | 0.9908 | 0.9907 | 0.9907 | 0.9907 |

优化器: SGD 训练时间: 889.13秒

优化器: ADAM 训练时间: 325.61秒

优化器: RMSPROP 训练时间: 495.04秒



分析：

**1. 三种优化器的表现差异：**

​Adam：下降最快且平稳，全程无震荡，最终损失为0；时间消耗最少

​RMSprop：整体平稳但初期有微小抖动，最终损失值为0

​SGD：全程剧烈震荡，持续锯齿状波动，收敛效果最差；时间消耗最多

1. **简单结论：**

不建议使用SGD，随机梯度下降由于通过少量数据（一个batch）的梯度来近似整个数据集的梯度，从而更新模型参数，这种方法容易造成loss振荡。综合而言，Adam优化器是最优选择

## 与其他四类模型方法比较

比较CNN与逻辑回归模型，FCN（单层全连接网络模型），决策树模型，以及高斯朴素贝叶斯模型性能。

**相关参数：**

CNN/FCN参数:批次大小(batch\_size=64),优化器(Adam), lr(0.01),epochs(20)

逻辑回归：多分类方法（multinomial->softmax回归） 最大决策次数(100)

决策树: 分裂标准(entropy) ， 最大树深度(不限制)

高斯朴素贝叶斯: 方差平滑系数（1e-9）

### 4.2.1实验结果

下面I以及Ⅱ分别是5个模型的实验指标结果以及分类报告对比

**I:机器学习方法结果**

| 模型 | 训练时间 | 测试准确率 | 测试精确率 | 测试召回率 | 测试 F1 值 |
| --- | --- | --- | --- | --- | --- |
| CNN | 127.27秒 | 0.9772 | 0.9772 | 0.9771 | 0.9771 |
| 逻辑回归 | 105.64秒 | 0.9262 | 0.9253 | 0.9251 | 0.9251 |
| ​FCN | 137.98秒 | 0.8971 | 0.9043 | 0.8965 | 0.8955 |
| 决策树 | 7.46秒 | 0.8730 | 0.8719 | 0.8713 | 0.8714 |
| 贝叶斯 | 0.30秒 | 0.5558 | 0.6865 | 0.5484 | 0.5090 |

**Ⅱ:分类报告对比**

|  |
| --- |
| >>>>>> 卷积神经网络 <<<<<<  precision recall f1-score support  0 0.9818 0.9908 0.9863 980  1 0.9816 0.9894 0.9855 1135  2 0.9794 0.9671 0.9732 1032  3 0.9869 0.9673 0.9770 1010  4 0.9719 0.9868 0.9793 982  5 0.9743 0.9765 0.9754 892  6 0.9926 0.9749 0.9837 958  7 0.9833 0.9718 0.9775 1028  8 0.9532 0.9836 0.9682 974  9 0.9672 0.9633 0.9652 1009  accuracy 0.9772 10000  macro avg 0.9772 0.9771 0.9771 10000  weighted avg 0.9773 0.9772 0.9772 10000 |
| >>>>>> 全连接网络 <<<<<<  precision recall f1-score support  0 0.9218 0.9857 0.9527 980  1 0.9792 0.9551 0.9670 1135  2 0.9664 0.8362 0.8966 1032  3 0.8739 0.9059 0.8896 1010  4 0.8694 0.9420 0.9042 982  5 0.9135 0.8408 0.8757 892  6 0.9371 0.9489 0.9429 958  7 0.8723 0.9368 0.9034 1028  8 0.7295 0.9302 0.8177 974  9 0.9801 0.6838 0.8056 1009  accuracy 0.8971 10000  macro avg 0.9043 0.8965 0.8955 10000  weighted avg 0.9057 0.8971 0.8965 10000 |
| >>>>>> 决策树 <<<<<<  precision recall f1-score support  0 0.9197 0.9347 0.9271 980  1 0.9436 0.9727 0.9579 1135  2 0.8529 0.8595 0.8562 1032  3 0.8268 0.8366 0.8317 1010  4 0.8674 0.8595 0.8634 982  5 0.8164 0.8274 0.8218 892  6 0.8947 0.8779 0.8862 958  7 0.9283 0.8570 0.8912 1028  8 0.8354 0.8337 0.8345 974  9 0.8337 0.8543 0.8439 1009  accuracy 0.8730 10000  macro avg 0.8719 0.8713 0.8714 10000  weighted avg 0.8734 0.8730 0.8730 10000 |
| >>>>>> 逻辑回归 <<<<<<  precision recall f1-score support  0 0.9522 0.9755 0.9637 980  1 0.9619 0.9789 0.9703 1135  2 0.9273 0.9021 0.9145 1032  3 0.9025 0.9168 0.9096 1010  4 0.9378 0.9369 0.9373 982  5 0.8983 0.8711 0.8845 892  6 0.9441 0.9520 0.9480 958  7 0.9330 0.9212 0.9271 1028  8 0.8835 0.8799 0.8817 974  9 0.9122 0.9167 0.9145 1009  accuracy 0.9262 10000  macro avg 0.9253 0.9251 0.9251 10000  weighted avg 0.9260 0.9262 0.9260 10000 |
| >>>>>> 贝叶斯分类器 <<<<<<  precision recall f1-score support  0 0.7902 0.8878 0.8361 980  1 0.8456 0.9507 0.8951 1135  2 0.9048 0.2578 0.4012 1032  3 0.7088 0.3495 0.4682 1010  4 0.8842 0.1711 0.2867 982  5 0.5500 0.0493 0.0905 892  6 0.6500 0.9342 0.7666 958  7 0.8777 0.2724 0.4157 1028  8 0.2842 0.6653 0.3983 974  9 0.3694 0.9465 0.5314 1009  accuracy 0.5558 10000  macro avg 0.6865 0.5484 0.5090 10000  weighted avg 0.6917 0.5558 0.5170 10000 |

### 4.2.2实验结果分析

**I:整体性能对比**

**CNN：**在测试准确率（97.72%）、精确率（97.72%）、召回率（97.71%）和F1值（97.71%）上均显著优于其他模型，展现了其在图像分类任务中的强大能力。其训练时间（127.27秒）虽然较长，但性能优势明显。

**逻辑回归：**表现次优，测试准确率为92.62%，F1值为92.51%，训练时间为105.64秒。其性能优于FCN和决策树，说明在MNIST数据集上，线性模型仍具有较强的分类能力。

**FCN**（全连接网络）： 的测试准确率为89.71%，F1值为89.55%，训练时间为137.98秒。其性能低于CNN和逻辑回归，但高于决策树和贝叶斯模型。值得注意的是，FCN的训练时间最长，但性能并未显著优于逻辑回归，表明其结构简单性可能限制了性能提升。

**决策树**：测试准确率为87.30%，F1值为87.14%，训练时间仅为7.46秒，是五种模型中训练速度最快的。尽管其性能略低于逻辑回归和FCN，但在时间效率上具有明显优势，适合快速原型开发或资源受限的场景。

**高斯朴素贝叶斯：**表现最差，测试准确率仅为55.58%，F1值为50.90%。虽然其训练时间极短（0.30秒），但性能远低于其他模型，表明其在处理图像数据时存在明显的局限性，可能由于其对特征独立性的强假设与图像数据的实际分布不符。

**Ⅱ：各类别表现分析**

**CNN：**在所有数字类别上的F1值均高于96%，表现最为均衡。例如，数字“0”和“1”的F1值分别达到98.63%和98.55%，而数字“8”和“9”的F1值稍低（96.82%和96.52%），但仍显著优于其他模型。

**逻辑回归：**在数字“1”和“0”上的表现最佳（F1值分别为97.03%和96.37%），但在数字“8”和“9”上的表现稍弱（88.17%和91.45%），表明其对某些类别的区分能力有限。

**FCN：**在数字“0”和“1”上的F1值较高（95.27%和96.70%），但在数字“8”和“9”上的表现较差（81.77%和80.56%），尤其是数字“9”的召回率仅为68.38%，说明其分类能力存在明显的不均衡性。

**决策树：**在数字“1”上的表现最佳（F1值为95.79%），但在其他类别上的表现较为平均，F1值集中在83%-89%之间。其分类能力相对稳定，但整体精度不高。

贝叶斯：表现极不均衡，数字“1”的F1值高达89.51%，而数字“5”的F1值仅为9.05%，召回率低至4.93%。这种极端的不均衡性表明贝叶斯模型在处理某些类别时完全失效。

**贝叶斯：**表现极不均衡，数字“1”的F1值高达89.51%，而数字“5”的F1值仅为9.05%，召回率低至4.93%。这种极端的不均衡性表明贝叶斯模型在处理某些类别时完全失效。

# 实验总体结论+原因推断

## 5.1实验现象原因推断：

①为何贝叶斯分类器效果很差？

高斯朴素贝叶斯在 MNIST手写数字分类任务上表现较差的原因主要源于其假设与数据特性的根本性冲突。MNIST每个像素的值通常是多模态分布。例如数字"1"的中间列像素可能集中高值（笔画区域）和低值（背景区域），而高斯分布只能拟合单峰。高斯朴素贝叶斯需要为每个特征估计均值和方差，但MNIST的784维空间中数据稀疏，导致参数估计不准。

②为何CNN相比于其他模型会效果好很多?

CNN通过卷积核局部感知像素区域，自动捕捉笔画、边缘等局部模式。同一卷积核在图像所有位置共享参数，大幅减少参数量，这样可以大大增加速度。且对图像数据本质特性（局部性、平移不变性、层次结构）的精准建模增加稳定性，比如无论数字"7"出现在图像左上角还是右下角，相同的卷积核均可识别。

## 5.2 实验结论

**I：参数选择上：**

实验表明，**学习率**lr=0.01（SGD优化器）在较少的训练轮次（epochs=20）下表现最佳，而lr=0.1虽然收敛更快，但可能导致训练不稳定（loss 振荡）。当epochs次数较大时，推荐使用学习率较小的数据。

**优化器对比**：Adam（lr=0.001）在训练稳定性、收敛速度和最终性能（99.15%准确率）上均优于SGD和RMSprop，是深度学习任务的首选优化器。

**Ⅱ：在模型选择上：**

实验结果表明，CNN是MNIST分类任务中的最优选择，其高精度和均衡的类别表现使其成为图像分类的首选模型。逻辑回归和决策树分别在性能和效率上具有独特优势，适合特定场景。贝叶斯的表现较差，需进一步优化或调整应用场景。

# 附录

## 工程结构

img-classify/

├── data/ # 数据集

├── bayesian.py # 贝叶斯分类器实现

├── config.py # 参数全局配置文件

├── decisiontree.py # 决策树分类器

├── FCNNet.py # 单层全连接网络

├── logisticregress.py # 逻辑回归模型

├── main.py # 主程序入口

├── network.py # CNN神经网络

├── checkpoints # CNN网络模型权重

└── utils.py # 工具函数（包含数据集读取函数等）

## 环境依赖

matplotlib==3.7.2  
numpy==1.23.5  
Pillow==11.2.1  
scikit\_learn==1.6.1  
torch==2.7.1+cu118  
torchvision==0.22.1+cu118

## 6.3参考文献

[1]周志华. 机器学习[M]. 北京: 清华大学出版社, 2016

[2]李沐, 等. 动手学深度学习[M]. 2版. 北京: 人民邮电出版社, 2023