摘要

本实验基于 MNIST 手写数字数据集，对比分析了机器学习和深度学习模型在数字分类任务中的性能和特点。

首先，选取了逻辑回归、随机森林和 XGBoost 等传统机器学习模型，对预处理后的图像数据进行分类。结果显示，随机森林模型在机器学习方法中表现最佳，测试准确率达到 96.82%，但总体上低于深度学习模型。

其次，采用卷积神经网络（CNN）和 ResNet18 等深度学习模型，直接对原始图像进行训练，充分利用了其自动特征提取能力。实验结果表明，这些深度学习模型的测试准确率均超过 99%，显著优于传统机器学习模型，体现了其在复杂模式识别任务中的优势。

此外，通过引入旋转数据增强技术，系统地探讨了数据增强对模型鲁棒性的影响。实验设计包括两种旋转方式：小幅度旋转（-12° 至 12°，2° 为步长）和大幅度旋转（-60° 至 60°，10° 为步长）。实验结果显示，小幅度旋转能够显著提高模型的分类性能，尤其是随机森林模型，测试准确率由 96.82% 提高至 98.02%。这一提升表明小幅度旋转增强能够丰富数据多样性，使模型更具泛化能力。而对于大幅度旋转，尽管随机森林模型表现出一定的鲁棒性（准确率微升至 96.98%），但其他模型，如逻辑回归和 XGBoost，准确率分别下降至 81.06% 和 85.41%，表明大范围扰动会显著破坏特征分布，对线性和非线性模型均构成挑战。

综上所述，本实验全面对比了机器学习与深度学习模型在手写数字分类任务中的性能表现，突出了深度学习模型在准确率和自动特征提取方面的优势。同时，实验还验证了数据增强技术的重要性：适度的数据增强（如小幅度旋转）能有效提高模型的鲁棒性，而过度扰动（如大幅度旋转）则可能破坏特征分布，降低模型性能。

第一部分: 探究机器学习与深度学习的差别

1 MNIST 数据集介绍

1.1 数据集概览

MNIST(Modified National Institute of Standards and Technology 数据集，作为机器学习和计算机视觉领域的瑰宝，是众多研究者和开发者入门及深入研究的基准数据集之一。该数据集以手写数字图像为核心，是一个经典的图像识别问题，为图像分类算法提供了一个简洁而有效的测试与训练基础。

MNIST 数据集包含 70,000 张灰度图像，这些图像均为 28×28 像素的手写数字，涵盖了 0 至 9 共十个数字类别。每一张图像都经过严格的规范化处理，确保数字位于图像中心，便于算法识别。

数据集的划分如下：

- 训练集：包含 60,00 张图像，是模型学习和掌握分类规律的主要来源。
- 测试集：包含 10,00 张图像，用于评估模型在实际应用中的泛化能力。

1.2 数据集构成

数据集由图像与标签构成：

- 图像：每张图像的尺寸是 28×28，每个像素点的值表示灰度值，范围是 0 到 255（0 代表黑色，255 代表白色，介于 0 和 255 之间的值代表不同的灰度级别）。
- 标签：每张图像对应一个标签，表示图像中的数字。

1.3 数据集应用

MNIST 数据集在机器学习和深度学习领域被广泛使用，尤其在图像分类任务中发挥着重要的作用，是经典的”Hello World” 数据集，以下是其主要用途：

1. 算法性能对比：由于其简洁性和适中规模，MNIST 数据集成为众多初学者和研究人员评估和比较各类机器学习算法性能的基石。
2. 神经网络训练：在深度学习领域，MNIST 数据集被广泛采用。它不仅有助于验证模型的基本能力，还能为新模型的研发提供有力的支持。

1.4 数据预处理

1. 归一化：这一步骤涉及将图像的像素值从原始的 $[0, 255]$ 范围调整到 $[0, 1]$ 区间。这种缩放有助于神经网络更快地收敛，因为它减少了数值的范围，使得梯度下降算法在训练过程中更加稳定。

2. 扁平化：对于传统的机器学习算法，我们需要将 28×28 的二维图像数据转换为一维向量。这一过程称为扁平化，它使得算法能够以向量形式处理图像数据，从而进行有效的特征提取和分类。

1.5 数据集的挑战

MNIST 数据集在机器学习和深度学习任务中有许多优势，但也存在一定的挑战：

1. 图像扭曲和旋转：手写数字的形态可能因为书写习惯而产生扭曲、倾斜或旋转，这对模型的识别能力提出了挑战。
2. 不同书写风格：由于数据集中手写数字来自不同个体，书写风格的多样性可能会对识别模型造成影响。
3. 噪声问题：尽管数据集经过预处理，但图像中可能仍然存在轻微的噪声，这可能会降低模型的准确性。

1.6 数据集的局限性

虽然 MNIST 数据集在许多研究中得到了广泛应用，但其局限性也需要注意：

1. 图像简单：MNIST 图像的尺寸较小（28×28 像素），内容相对简单，缺乏复杂性。这可能导致模型在面对更复杂图像时出现过拟合现象。
2. 缺乏多样性：MNIST 数据集中的手写数字图像主要来自有限的几种来源，限制了数据的多样性。在现实场景中，图像分类任务通常涉及更复杂的背景和多样化的样本，MNIST 数据集在这方面的代表性有限。

1.7 小组研究目标

基于 MNIST 数据集，我们小组提出以下两个研究问题：

1. 通过实验了解机器学习与深度学习之间的差别，包括模型结构差别和准确率的差异。
2. 鉴于 MNIST 数据集的局限性（如部分图片倾斜），通过旋转图片观察机器学习模型的准确度变化，以探索优化方法。

2 数据统一处理和模型评价标准

2.1 数据集加载

在本实验中，MNIST 数据集通过 PyTorch 的 `torchvision.datasets.MNIST` 方法进行加载（此方法训练集数据大小为 $60,000$，测试集大小为 $10,000$）。该方法具有以下优点：

- 内置数据预处理功能：通过该方法，可以方便地对数据进行标准化、调整尺寸等预处理操作，使数据集更容易直接用于模型训练和测试。
- 兼容性强：加载的数据集与 PyTorch 的模型和数据加载器（`DataLoader`）高度兼容，能够简化训练流程，特别适用于深度学习模型的构建和训练。

2.2 数据集划分

- 对于机器学习模型：将数据集划分为训练集（90%）和测试集（10%）。
- 对于深度学习模型：进一步将原本的测试集对半划分，将 5% 的数据作为验证集，剩余 5% 的数据作为测试集，即训练集（90%）、测试集（5%）和验证集（5%）。

这种划分方式统一保存在 JSON 文件中。通过这种方式，可以确保数据集划分的一致性，并避免每次实验时重新划分数据，从而降低因数据划分不一致引起的偏差。

2.3 机器学习与深度学习的处理方式

机器学习模型的处理方式 ：在机器学习模型中，处理 MNIST 数据的第一步是对图像进行扁平化，将每个 28×28 像素的图像展平成一个长度为 784 的向量。这是为了使得图像能够符合传统机器学习算法（如支持向量机 SVM、逻辑回归等）的输入要求。具体操作通过以下代码实现：

```
img.view(img.size(0), -1)
```

此外，为了提高模型的训练效果，图像还会经过尺寸调整和标准化等预处理步骤，以便优化模型性能。

深度学习模型的处理方式 ：在深度学习中，直接使用原始的 28×28 像素图像，而不需要进行扁平化。图像首先通过数据预处理进行调整尺寸（如将图像调整为 224×224 像素），接着进行灰度转换和标准化。卷积神经网络（CNN）能够自动从图像中提取特征，保留图像的空间结构信息，并能够有效识别图像中的复杂模式。具体实现包括：

- 使用 `transforms.Resize` 调整为指定的输入尺寸。
- 使用 `transforms.Grayscale` 转换为灰度图。
- 使用 `transforms.Normalize` 对图像进行标准化处理，以确保模型训练的稳定性和准确性。

2.4 模型评价标准

对于机器学习和深度学习，我们统一采用以下评价指标：

- 准确率：衡量模型预测正确样本的比例。
- 混淆矩阵：用于评估模型的分类效果，分析错误分类的具体类型。
- 可视化：随机挑选 25 张图片进行预测结果的可视化展示。

3 混淆矩阵与性能评估

混淆矩阵是一种用于评估分类模型性能的工具，特别适用于监督学习中的分类任务。通过将预测结果与实际标签进行对比，混淆矩阵能够直观地展示模型在不同类别上的表现。

3.1 混淆矩阵的基本结构

混淆矩阵通常表示如下：

$$\begin{bmatrix} \text{TP} & \text{FN} \\ \text{FP} & \text{TN} \end{bmatrix}$$

其中：

- TP（True Positive）：被正确分类为正类的样本数。
- FN（False Negative）：实际为正类但被错误分类为负类的样本数（漏检）。
- FP（False Positive）：实际为负类但被错误分类为正类的样本数（误报）。
- TN（True Negative）：被正确分类为负类的样本数。

3.2 性能指标

基于混淆矩阵，我们可以计算以下重要性能指标：

准确率 **(Accuracy)** 模型预测正确的样本占总样本的比例：

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

精确率 **(Precision)** 预测为正类的样本中，实际为正类的比例：

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

召回率 **(Recall)** 实际为正类的样本中，被正确预测为正类的比例：

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1 分数 **(F1 Score)** 精确率和召回率的调和平均值：

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

特异性 **(Specificity)** 实际为负类的样本中，被正确预测为负类的比例：

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

假阳性率 **(False Positive Rate, FPR)** 实际为负类的样本中，被错误预测为正类的比例：

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

3.3 数据不平衡的影响

在类别不平衡的数据集中，准确率可能不足以全面反映模型性能。例如，如果负类样本远多于正类样本，仅预测为负类即可获得较高的准确率。在这种情况下，精确率、召回率以及 F1 分数更具参考价值。

3.4 总结

混淆矩阵是分类模型性能评估的重要工具。通过结合多种性能指标，可以全面了解模型的分类能力，并针对具体需求优化模型。

4 机器学习部分

在机器学习部分，我们选择了逻辑回归、XGBoost 和随机森林来对 MNIST 数据集进行分类。

4.1 Logistic 回归

4.1.1 模型介绍

Logistic 回归是一种线性分类模型，适用于二分类问题，也可以扩展到多分类问题。尽管其表达能力相对较弱，但在较简单的问题中，特别是在特征向量已经被合理扁平化的情况下，Logistic 回归仍然是一个有效的模型。

假设输入特征 x 和输出 y 存在线性关系，这种关系可以表示为：

$$z = w^T x + b$$

其中：

- x 为输入特征向量；
- w 为权重向量；
- b 为偏置项；
- z 是模型的线性输出。

逻辑回归的预测函数为：

$$P(y = 1|x) = \frac{1}{1 + e^{-z}}$$

对于输入 x，类别 1 和类别 0 的概率分别为：

$$P(y = 1|x) = \frac{1}{1 + e^{-z}}, \quad P(y = 0|x) = 1 - P(y = 1|x)$$

4.1.2 数据处理

在本实验中，我们对 MNIST 数据集进行了数据预处理和模型参数设置，具体如下：

为了使数据适配逻辑回归模型，预处理步骤如下：

- 将图像统一调整为 32×32 的分辨率；
- 转为单通道灰度图像；
- 将像素值归一化到 $[-1, 1]$ 范围；
- 以张量形式加载数据。

此外，为了适配随机森林模型，我们进一步将二维图像数据扁平化为一维向量。数据集的划分通过外部的 JSON 文件完成，该文件包含训练集 (90%)、验证集 (5%) 和测试集 (5%) 的索引。在实验中，我们将验证集与测试集合并，以扩大测试数据规模，使模型能够在更广泛的数据分布上进行评估。。

4.1.3 模型参数设置

逻辑回归模型的超参数设置如下：

- 最大迭代次数 **(max_iter)**：设置为 1000，以确保模型能够充分收敛；
- 随机种子 **(random_state)**：设为 42，以保证结果的可重复性；
- 其他参数：逻辑回归模型默认使用 L2 正则化，并通过拟牛顿法（LBFGS）优化损失函数。

4.1.4 模型结果

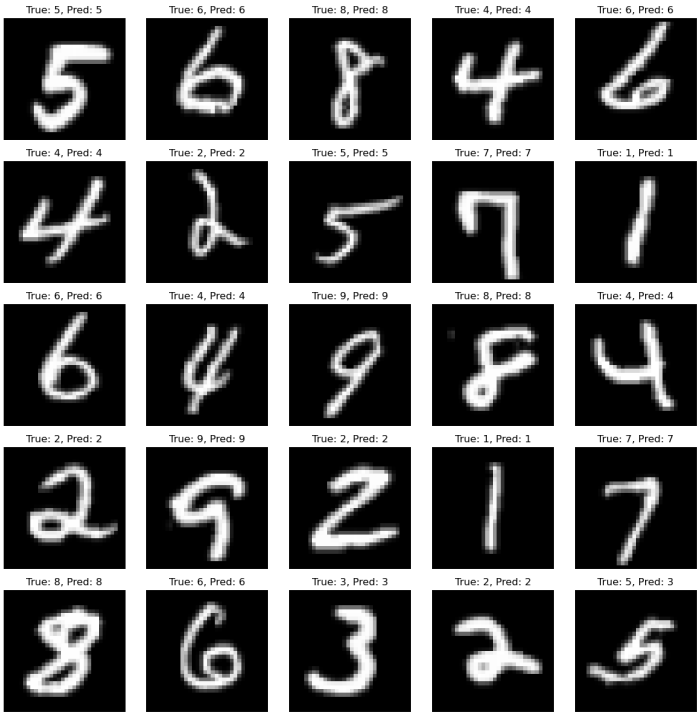

Figure 1: Logistic 回归部分结果

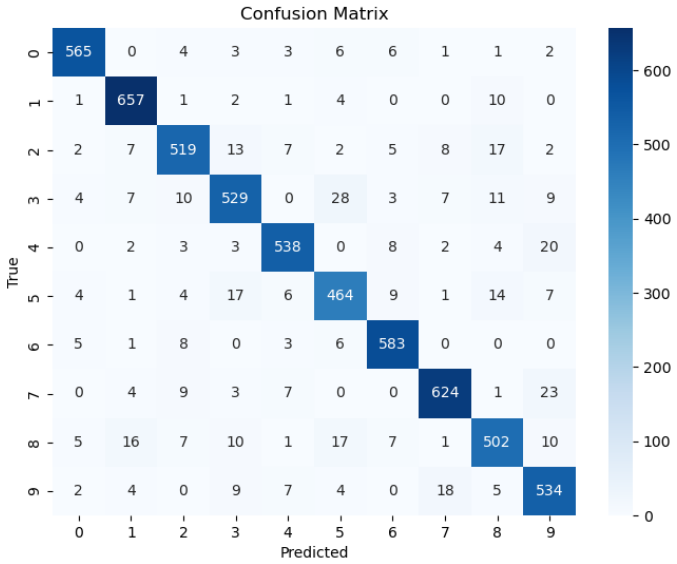

Figure 2: Logistic 回归混淆矩阵

经过 1000 次迭代计算后，Logistic 回归的测试准确率（Test Accuracy）为：0.9192

4.2 XGBoost 模型

XGBoost（eXtreme Gradient Boosting，极限梯度提升树）是一种集成学习算法，通过构建多个弱学习器来逐步提高预测性能。其目标函数结合了模型误差和正则项，通过泰勒二阶展开提高精度，并允许自定义损失函数。

4.2.1 模型介绍

XGBoost 模型的目标函数可以表示为：

$$\mathcal{L}(t) = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^{t} \Omega(f_k)$$

其中，$l(y_i, \hat{y}_i^{(t)})$ 表示模型预测值与真实值之间的误差，$\Omega(f_k)$ 是正则项。

正则项公式如下：

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

其中：

- γ：叶子节点的复杂度控制系数；
- λ：L2 正则化系数；
- T：树的叶子节点数；
- w_j：每个叶子节点的权重。

XGBoost 模型支持多种损失函数，以下是常用的两种：

1. 平方损失（用于回归问题）：

$$l(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

2. 逻辑损失（用于分类问题）：

$$l(y, \hat{y}) = -\left[y \log \hat{y} + (1 - y) \log(1 - \hat{y})\right]$$

XGBoost 模型通过优化目标函数 $\mathcal{L}(t)$，结合正则化项 $\Omega(f)$，构建多个弱学习器（如决策树），并利用加法模型逐步提升预测性能。

4.2.2 数据处理

在本实验中，我们使用 XGBoost 模型对 MNIST 数据集进行了分类任务，旨在评估该梯度提升决策树模型在手写数字识别中的表现。

数据预处理包括以下步骤：

- 将图像转换为张量形式；
- 将像素值归一化到 $[-1, 1]$ 的范围；
- 将二维图像数据扁平化为一维向量，以适配 XGBoost 模型的输入要求。

此外，为了适配随机森林模型，我们进一步将二维图像数据扁平化为一维向量。数据集的划分通过外部的 JSON 文件完成，该文件包含训练集、验证集和测试集的索引。在实验中，我们将验证集与测试集合并，以扩大测试数据规模，使模型能够在更广泛的数据分布上进行评估。

4.2.3 模型参数设置

XGBoost 模型的超参数设置如下：

- 弱学习器数量 **(n_estimators)**：设置为 100，以控制模型的复杂度；
- 学习率 **(learning_rate)**：设置为 0.1，用于控制每棵树对最终模型的贡献；
- 最大树深度 **(max_depth)**：设置为 3，以限制单棵树的复杂度，防止过拟合；
- 树构建方法 **(tree_method)**：根据硬件条件自动选择 `gpu_hist`（使用 GPU 加速）或 `hist`（使用直方图加速）；
- 随机种子 **(random_state)**：设为 42，以确保结果的可重复性。

4.2.4 模型结果

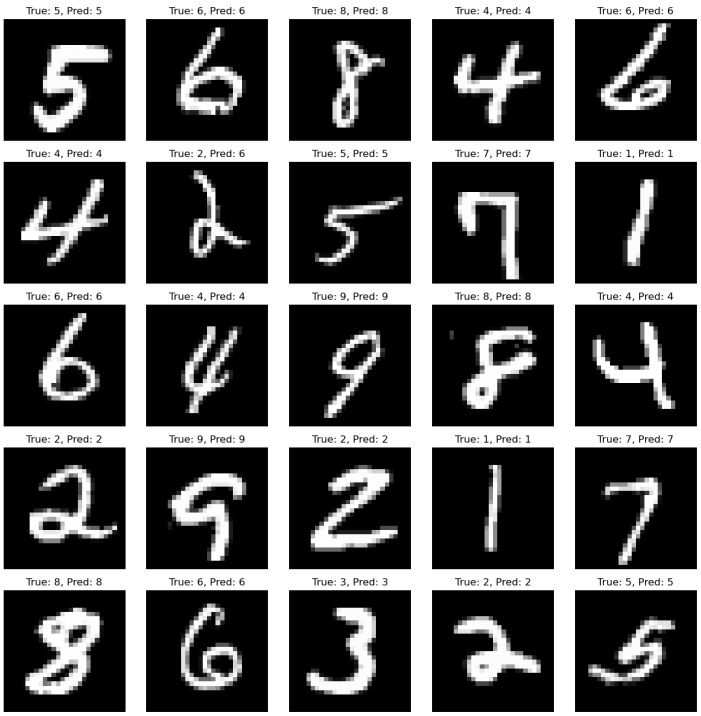

Figure 3: XGBoost 模型部分结果

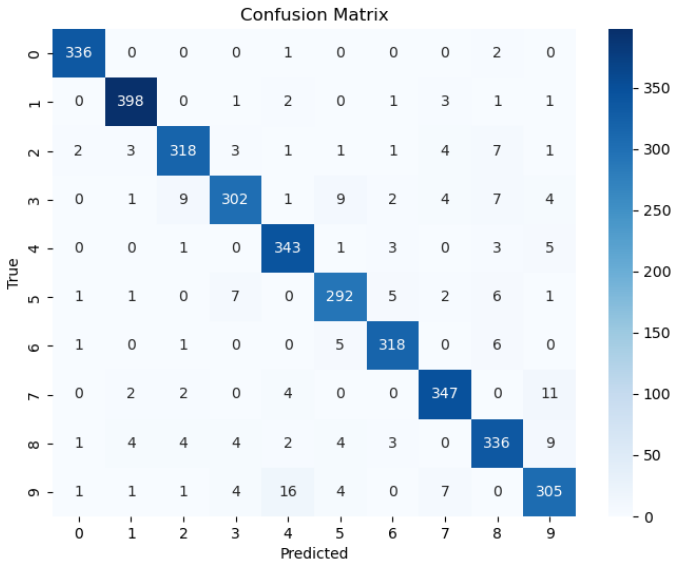

Figure 4: XGBoost 模型混淆矩阵

XGBoost 模型的测试准确率（Test Accuracy）为：0.9302

4.3 随机森林

4.3.1 模型介绍

随机森林是一种基于决策树的集成学习方法，通过构建多个决策树并将其结果进行汇总来提高分类精度。随机森林算法的主要步骤包括：

1. 数据抽样：从原始数据集中有放回地随机抽样，生成多个子数据集。
2. 树构建：对于每个子数据集，训练一个决策树模型。在每个节点上，随机选择一部分特征，从中找出最优特征进行分割。
3. 聚合结果：对于分类问题，随机森林通过投票选择多数分类结果；对于回归问题，则取所有树预测值的平均值。

随机森林的公式表示为：

$$F(x) = \frac{1}{B}\sum_{b=1}^{B} h_b(x)$$

其中，$h_b(x)$ 表示第 b 棵树的预测结果，B 为树的总数。

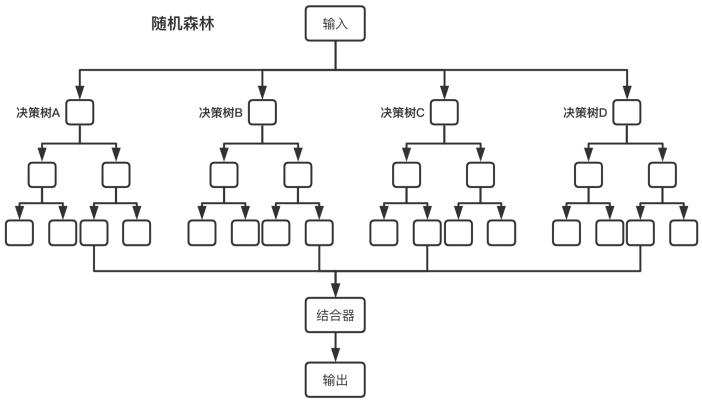

Figure 5: 随机森林具体流程

4.3.2 数据处理

数据预处理包括以下步骤：

- 将图像统一调整为 32×32 的分辨率；
- 转为单通道灰度图像；
- 将像素值归一化到 $[-1, 1]$ 的范围；
- 以张量形式加载数据。

此外，为了适配随机森林模型，我们进一步将二维图像数据扁平化为一维向量。数据集的划分通过外部的 JSON 文件完成，该文件包含训练集、验证集和测试集的索引。在实验中，我们将验证集与测试集合并，以扩大测试数据规模，使模型能够在更广泛的数据分布上进行评估。

4.3.3 模型参数设置

随机森林模型的主要超参数设置如下：

- 弱学习器数量 **(n_estimators)**：设置为 100，表示模型包含 100 棵决策树；
- 随机种子 **(random_state)**：设为 42，以确保实验结果的可重复性；
- 其他参数：随机森林使用默认参数配置，如基于不完全深度的决策树进行集成。

4.3.4 模型结果

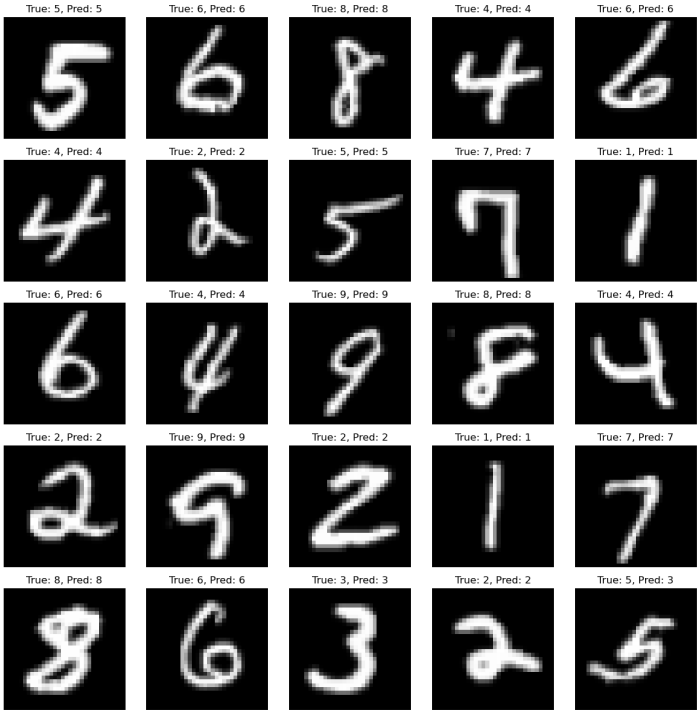

Figure 6: 随机森林部分结果

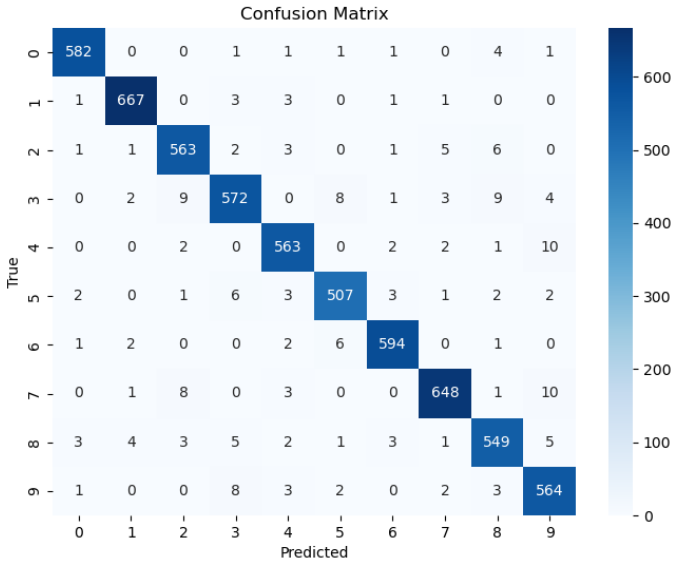

Figure 7: 随机森林混淆矩阵

随机森林的测试准确率（Test Accuracy）为：0.9682

4.4 结论

1. 机器学习的准确率已经达到了 90% 以上，表现良好。随着模型复杂度的增加，预测的准确率也随之上升。
2. 在随机的 25 张图片中，最后一张图片是预测错误的，但通过人眼可以轻易分辨数字的标签。因此，我们认为在处理手写数字标签分类的问题上，机器学习目前还不能超越人脑的水平。
3. 不同机器学习模型的架构差别较大。

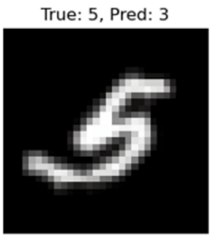

Figure 8: 三个模型都预测错误的图像

5 深度学习部分

在深度学习部分，我们选择了卷积神经网络（CNN）和 ResNet18 两种模型对 MNIST 数据集进行分类。对于两种模型，我们统一使用残差连接、Adam 优化器和 ReLU 激活函数。

5.1 卷积神经网络（CNN）

5.1.1 模型介绍

卷积神经网络（Convolutional Neural Network, CNN）是一种专门为处理图像数据而设计的神经网络结构，能够自动提取图像中的空间特征。通过使用卷积层、池化层和全连接层，CNN 能够在多个层次上学习图像的局部特征，并最终将其映射到类别标签。

5.1.2 数据处理

数据处理包括以下步骤：

- 将图像调整为 224×224 的分辨率，符合 ResNet 输入尺寸的要求；
- 将像素值归一化到 $[-1, 1]$ 的范围；
- 转换为 PyTorch 张量格式。

此外，数据集按照比例划分为训练集（90%）、验证集（5%）和测试集（5%）。数据划分索引通过外部 JSON 文件进行管理，以保证实验的可重复性和训练、测试样本的独立性。

5.1.3 模型结构

残差块

- 每个残差块包括两个卷积层，均采用 3×3 的卷积核，同时加入批量归一化（BatchNorm）和激活函数 ReLU。
- 在需要调整特征图大小或通道数时，使用 1×1 的卷积实现快捷连接。

网络架构　该 CNN 模型基于 ResNet 的残差学习思想，包括以下结构：

- 初始卷积层：7×7 卷积核，步幅为 2，配合批量归一化和最大池化；
- 四个残差模块：每个模块由两个残差块构成，逐步增加通道数（$64 \to 128 \to 256 \to 512$）；
- 全连接层：输入尺寸为 $512 \times 7 \times 7$，输出为 10 类。

5.1.4 训练设置

- 损失函数：交叉熵损失函数，用于多分类任务；
- 优化器：AdamW 优化器，学习率设置为 0.001，具有更好的权重衰减控制；
- 训练周期：共训练 10 个 epoch；
- 批量大小：每次迭代处理 64 张图像；
- 网络层数：模型总共有 34 层，包括初始卷积层、批量归一化层、最大池化层、4 个残差模块以及最终的全连接层；

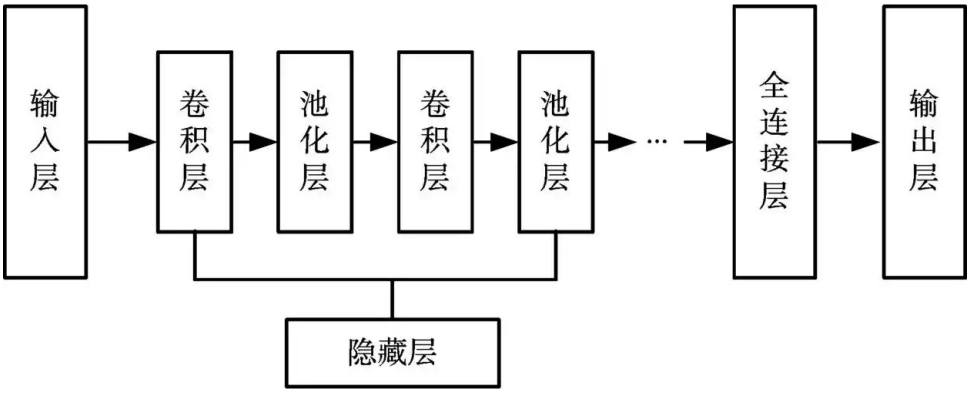

Figure 9: 卷积神经网络（CNN）模型构架

5.1.5 模型结果

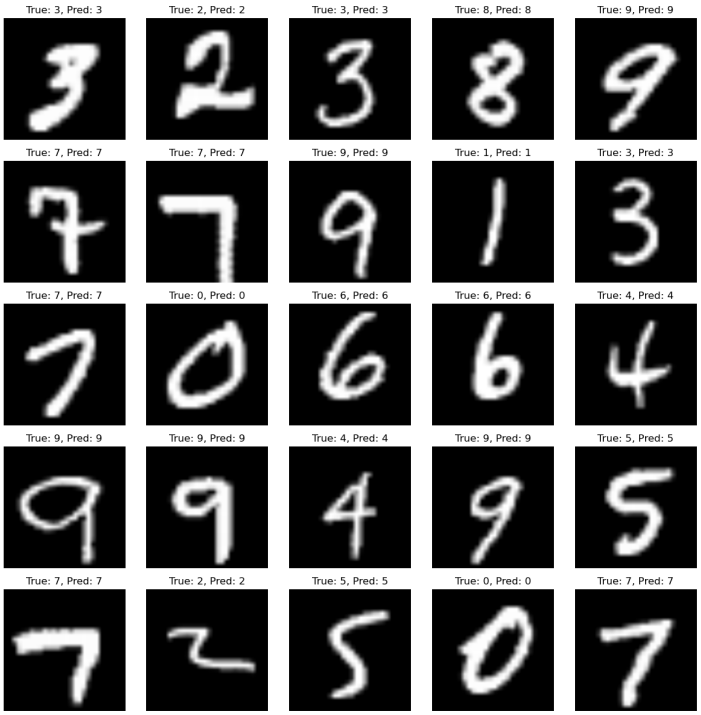

Figure 10: 卷积神经网络（CNN）部分结果

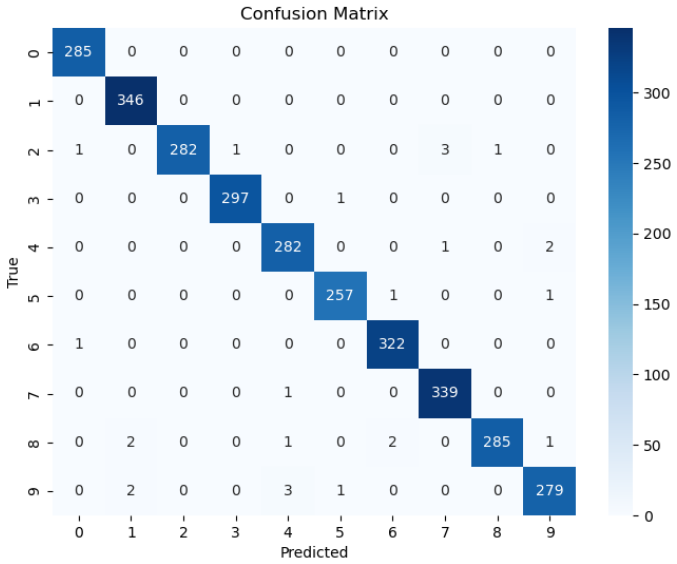

Figure 11: 卷积神经网络（CNN）混淆矩阵

经过训练后，卷积神经网络的测试准确率为：0.9913

此外，训练过程中产生的损失曲线（Loss Curve）和准确率曲线（Accuracy Curve）如下：

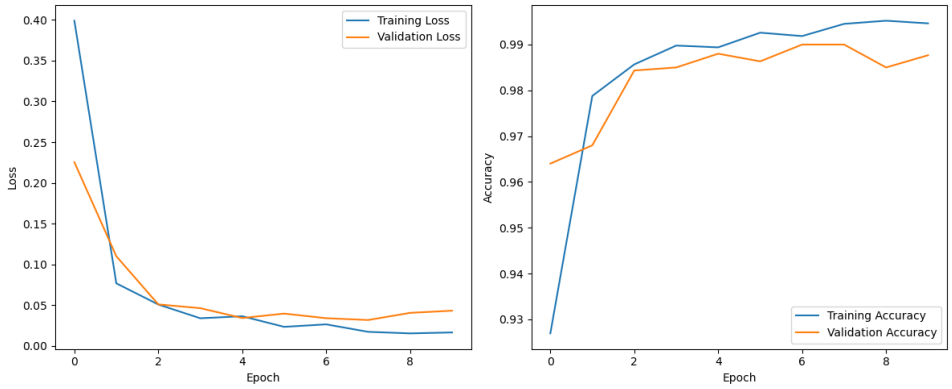

Figure 12: CNN 的损失曲线与准确率曲线

5.2 ResNet18

5.2.1 模型介绍

ResNet18 是一种深度残差网络（Residual Network），通过引入残差连接（Skip Connections）有效缓解了深层网络中的梯度消失问题和训练困难问题。ResNet 模型在许多计算机视觉任务中表现出了卓越的性能。

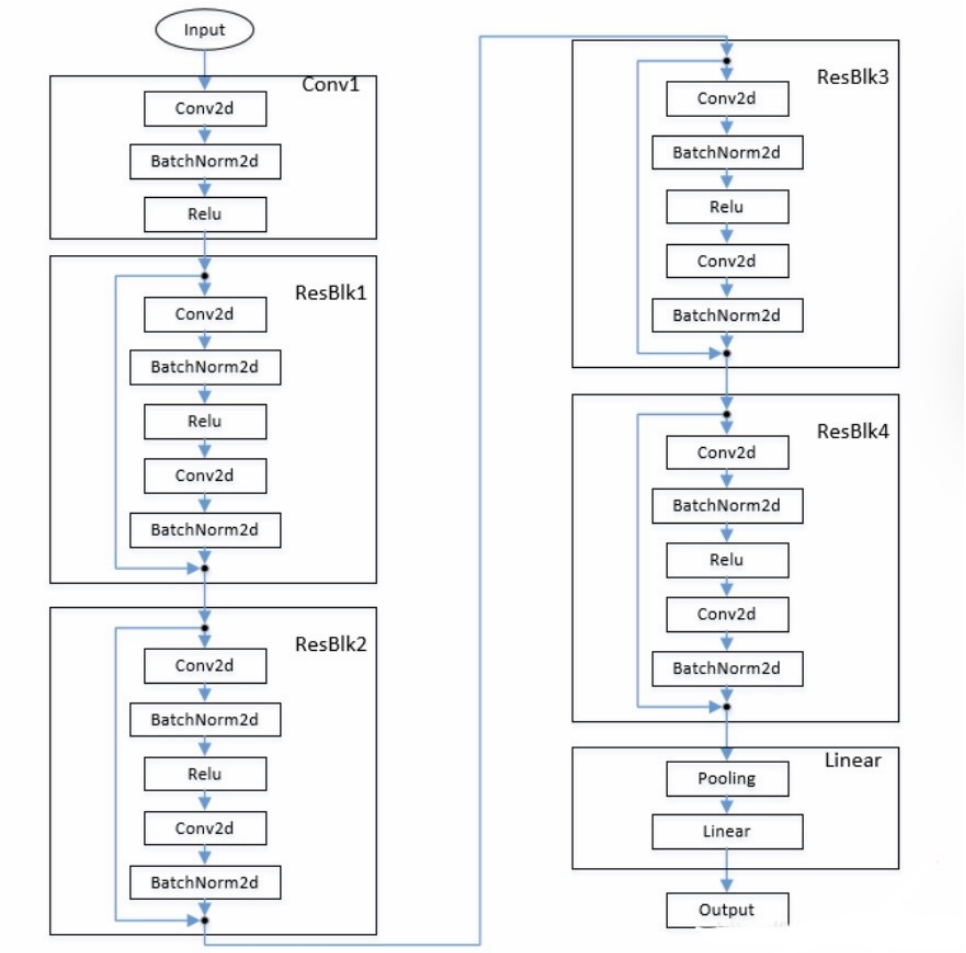

Figure 13: ResNet18 模型构架

5.2.2 数据处理

数据处理包括以下步骤：

- 将图像调整为 224×224 的分辨率，符合 ResNet 模型输入尺寸的要求；
- 将像素值归一化到 $[-1, 1]$ 的范围；
- 转换为 PyTorch 张量格式。

此外，数据集按照比例划分为训练集（90%）、验证集（5%）和测试集（5%）。数据划分的索引通过外部 JSON 文件进行管理，确保实验的可重复性和训练、测试样本的独立性。

5.2.3 模型结构

ResNet18 模型调整 我们使用了预训练的 ResNet18 模型，并针对 MNIST 数据集进行了如下调整：

- 修改了输入卷积层，将输入通道数从 3 修改为 1，以适配单通道灰度图像；
- 修改了全连接层，将输出类别数调整为 10，以适配 MNIST 数据集的分类任务；
- 保留了原模型的残差结构，充分利用预训练权重进行特征提取。

5.2.4 训练设置

- 损失函数：交叉熵损失函数，用于多分类任务；
- 优化器：AdamW 优化器，学习率设置为 0.001，具有更好的权重衰减控制；
- 训练周期：共训练 10 个 epoch；
- 批量大小：每次迭代处理 64 张图像；
- 网络层数：ResNet18 总共有 18 层，其中包括多个残差模块、卷积层和全连接层；

5.2.5 模型结果

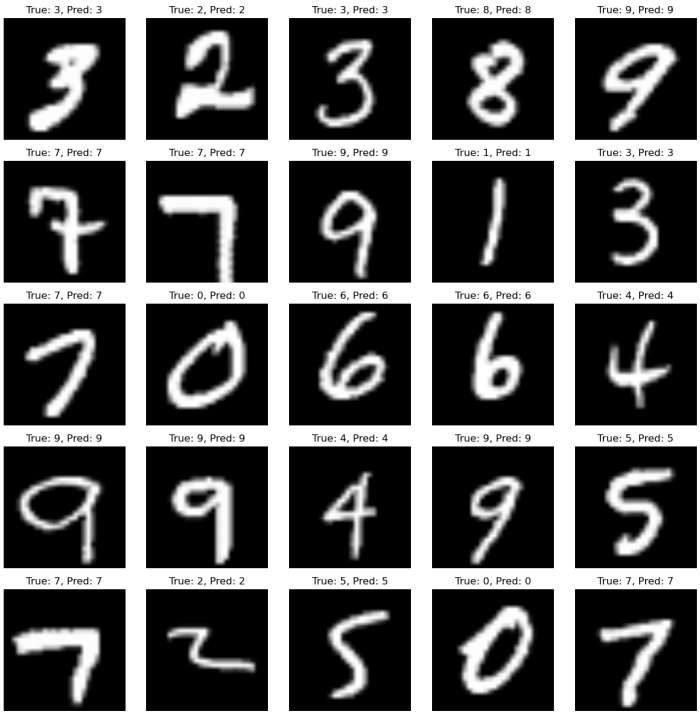

Figure 14: ResNet18 部分结果

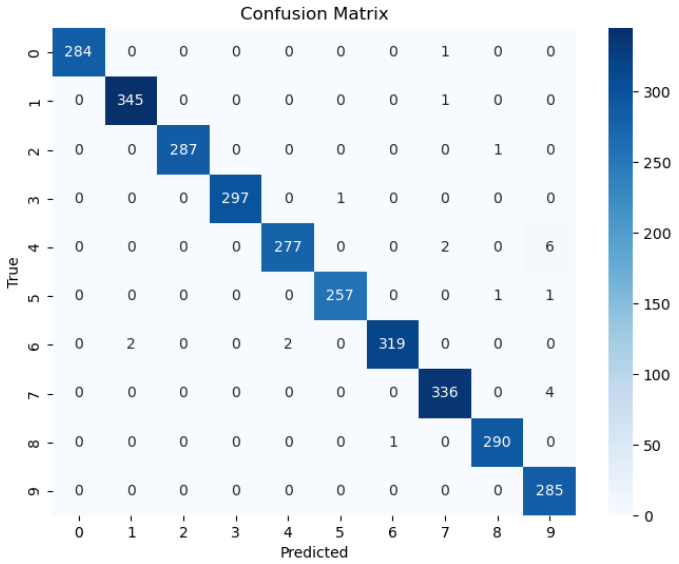

Figure 15: ResNet18 混淆矩阵

ResNet18 模型的测试准确率为：0.9923

同样地，训练过程中产生的损失曲线（Loss Curve）和准确率曲线（Accuracy Curve）如下：

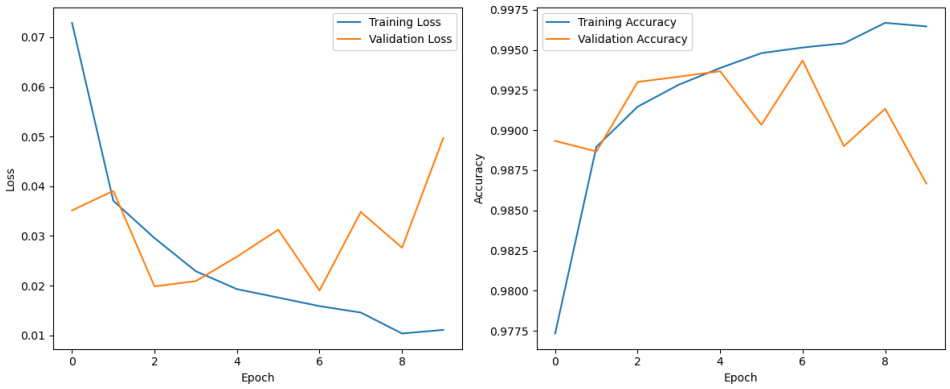

Figure 16: ResNet18 的损失曲线与准确率曲线

5.3 结论

1. 深度学习的准确率非常高，大概在 99% 以上。两个模型的准确率差别并不大。
2. 我们挑出了 20 张预测错误的图片，发现即使是人眼也不能轻易分辨出这些数字的真实标签。
3. 深度学习的模型架构比较统一。因此，我们认为在处理手写数字标签的问题上，深度学习在一定程度上可以替代人类。

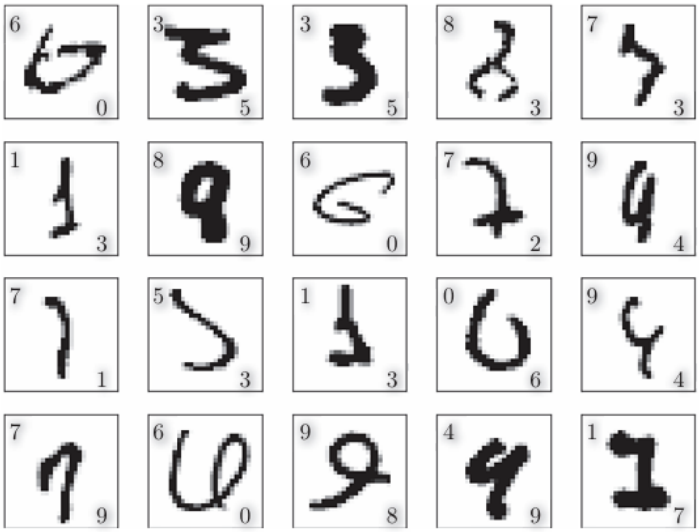

Figure 17: 深度学习预测错误图像

6 总结

通过本次实验，我们对机器学习与深度学习在 MNIST 数据集上的分类性能进行了全面的研究与对比，得出以下结论：

6.1 机器学习与深度学习的对比

1. 特征提取方式：
   - 在机器学习中，数据的特征量通常需要通过人工设计，而在深度学习中，神经网络能够直接从图像中学习特征。
   - 深度学习的卷积层可以自动提取图像的空间特征，而机器学习模型依赖于输入向量化后的数据。

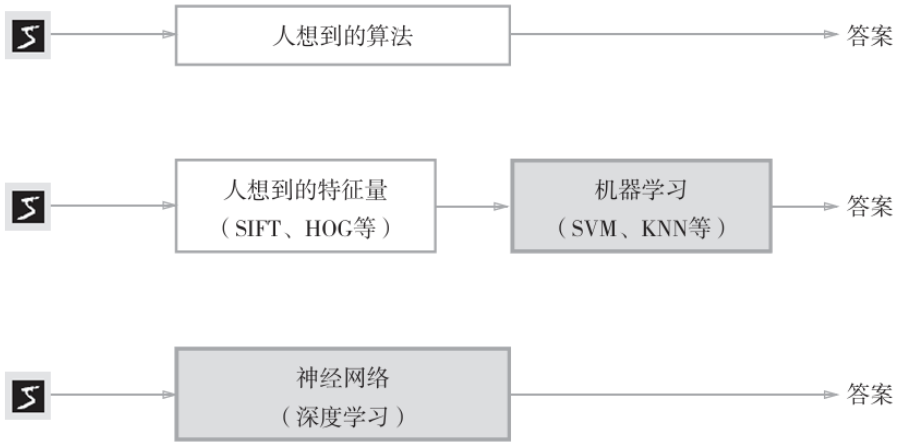

Figure 18: 人脑, 机器学习, 深度学习对比

2. 准确率对比：

- 深度学习模型（如 CNN 和 ResNet18）的准确率显著高于机器学习模型（如 Logistic 回归和随机森林）。
- 深度学习模型的测试准确率均在 99% 以上，而机器学习模型中，随机森林的表现最佳，测试准确率为 96.82%。

6.2 深度学习的优势与局限

1. 优势：
   - 深度学习在处理复杂的高维数据时具有显著的优势，尤其在 MNIST 这样的图像数据集上表现突出。
   - CNN 和 ResNet18 的统一架构和自动化特征提取能力使其在分类任务中更加高效。
2. 局限性：
   - 深度学习对数据的依赖程度较高，在低质量或噪声较大的数据上可能表现欠佳。
   - 深度学习模型的训练时间长，对计算资源要求较高。

6.3 实际问题中的启示

- 机器学习模型在一些简单或低维任务中仍然具有竞争力，尤其在计算资源有限的情况下。
- 深度学习模型在分类准确率和复杂任务处理能力上展现了更大的潜力，适合处理高维数据和大规模数据集。

7 不足与改进方向

1. 实验设计：
   - 未能统一机器学习与深度学习的模型复杂度（如学习深度和参数），可能导致对比不够公平。
   - 部分机器学习模型的参数未经过最优调节，可能对其性能产生影响。
2. 数据扩展：
   - MNIST 数据集的简单性可能限制了实验结果的广泛适用性。未来实验可以引入更复杂的图像数据集（如 CIFAR-10 或 ImageNet）。
   - 对旋转或扭曲数据的扩展分析未深入探讨，未来可以进一步研究不同预处理对模型表现的影响。
3. 技术优化：
   - 对深度学习模型，可尝试引入更先进的架构（如 ResNet50 或 Transformer）。
   - 对机器学习模型，可引入超参数优化方法（如网格搜索或贝叶斯优化）提升性能。

第二部分: 探讨旋转对手写数字识别模型准确率的影响

1 引言

根据上一实验中模型在测试集上的表现，我们发现虽然传统机器学习模型能够在标准 MNIST 数据集上取得较高的分类准确率，但在实际应用中，手写数字的形状往往因旋转、倾斜等变化而变得更加复杂。这种数据扰动可能导致模型性能的下降，特别是在未经过数据增强训练的情况下。

针对这一问题，我们小组决定开展第二个实验，主要探讨通过生成旋转数据增强训练集是否能够提高模型的鲁棒性和适应性。本次实验的重点是通过在训练数据中引入旋转样本，观察模型对标准测试集和旋转测试集的分类性能变化，同时分析不同机器学习模型（如 XGBoost、随机森林和逻辑回归）在这一任务上的适应能力。

这一实验的意义在于，不仅可以验证数据增强技术在手写数字识别中的实际效果，还能为设计更鲁棒的模型提供数据驱动的优化方向。

2 数据集与方法

2.1 数据集

本实验以 MNIST 数据集为基础，探讨旋转数据增强对手写数字识别的影响。为了实现这一目标，我们自定义了一个数据处理类 `RotatedMNIST`，该类在原始 MNIST 数据的基础上生成了包含不同旋转角度的扩展数据集。

2.2 旋转角度

在本实验中，我们将旋转角度分为两种不同的类型，以考察不同旋转强度对模型性能的影响：

- 大旋转：
  - 旋转角度范围较大，包括 $[-60°, -50°, \ldots, 60°]$，以 $10°$ 为步长。
  - 此类旋转增强方法可以显著增加数据的多样性，但可能引入更高的模型复杂性和训练难度。
- 小旋转：
  - 旋转角度范围较小，包括 $[-12°, -10°, \ldots, 12°]$，以 $2°$ 为步长。
  - 小幅度旋转增强更贴近真实世界中数字倾斜的情况，适合模拟实际手写场景的微扰动。

通过引入这两种旋转方式，可以分别评估模型对大幅度旋转和小幅度旋转数据的适应能力。

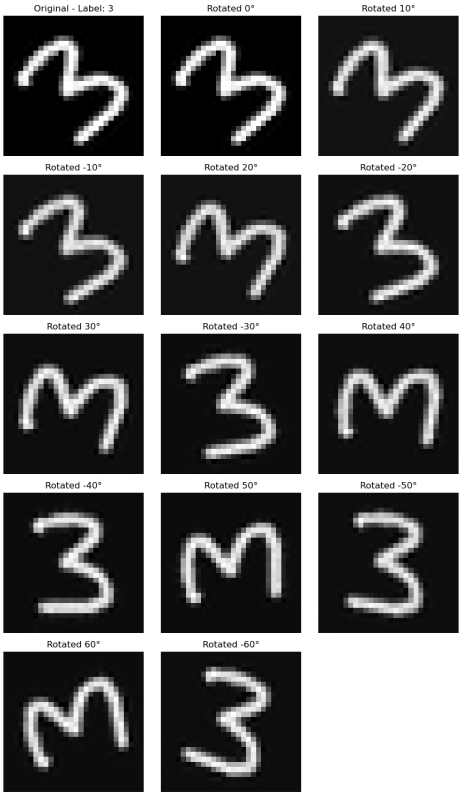

Figure 1: 大角度旋转

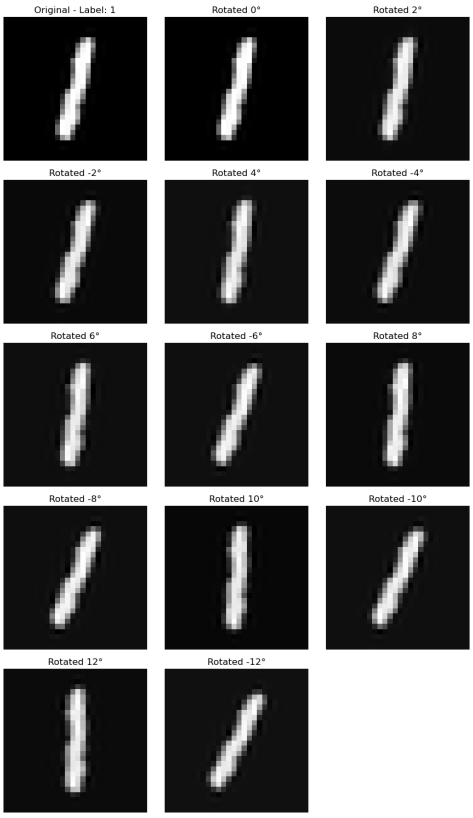

Figure 2: 小角度旋转

2.3 存储与内存优化

由于旋转数据增强会显著增加训练集的样本数量（大约为原始数据集的 k 倍，k 为旋转次数），扩展后的训练集体积变得非常庞大。为了解决大数据量带来的存储和内存问题，我们采用了以下优化措施：

- 内存映射（**MMAP**）格式：
  - 图像：以.dat 文件格式存储，支持按需加载，避免将整个数据集加载到内存中。
  - 标签：以.npy 文件格式存储，方便读取和索引。

  此方法能够高效利用磁盘空间，并减少内存压力，适合大规模数据集的处理。
- 数据量与内存问题：
  - 大旋转增强的数据集体积更大，例如 MNIST 的扩展数据集可以达到数十 GB。
  - 即使采用 MMAP 格式，训练时每次加载的批量数据也可能占用大量内存，特别是在多模型并行训练的情况下，可能导致内存资源不足。
- 解决方案：
  - 批量加载：使用 `DataLoader` 结合 MMAP 文件按需加载小批量数据，控制单次内存使用量。

2.4 数据可视化

为了验证旋转数据集生成的正确性，我们随机选取若干样本，展示其原始图像及旋转后的多个变种。通过可视化结果，直观验证旋转增强的效果，为实验分析提供基础。

3 实验结果

3.1 实验结果汇总

实验结果以分类准确率为衡量指标，同时计算了相对于无旋转数据集的提升率，具体如下：

模型	旋转数据集			提升率
	无旋转数据集	大范围旋转数据集	小范围旋转数据集	
Logistic Regression	0.9192	0.8106	0.9217	小范围：+0.27%；大范围：-11.81%
XGBoost	0.9302	0.8541	0.9314	小范围：+0.13%；大范围：-8.18%
Random Forest	0.9682	0.9698	0.9802	小范围：+1.24%；大范围：+0.17%

Table 1: 各模型在不同旋转数据集上的准确率及提升率

3.2 结果展示

3.2.1 Logistic Regression

- 无旋转数据集：准确率为 0.9192。
- 大范围旋转数据集：准确率下降至 0.8106，降幅为 -11.81%。
- 小范围旋转数据集：准确率提升至 0.9217，提升率为 +0.27%。

3.2.2 XGBoost

- 无旋转数据集：准确率为 0.9302。
- 大范围旋转数据集：准确率下降至 0.8541，降幅为 -8.18%。
- 小范围旋转数据集：准确率提升至 0.9314，提升率为 +0.13%。

3.2.3 Random Forest

- 无旋转数据集：准确率为 0.9682。
- 大范围旋转数据集：准确率微幅提升至 0.9698，提升率为 +0.17%。
- 小范围旋转数据集：准确率提升至 0.9802，提升率为 +1.24%。

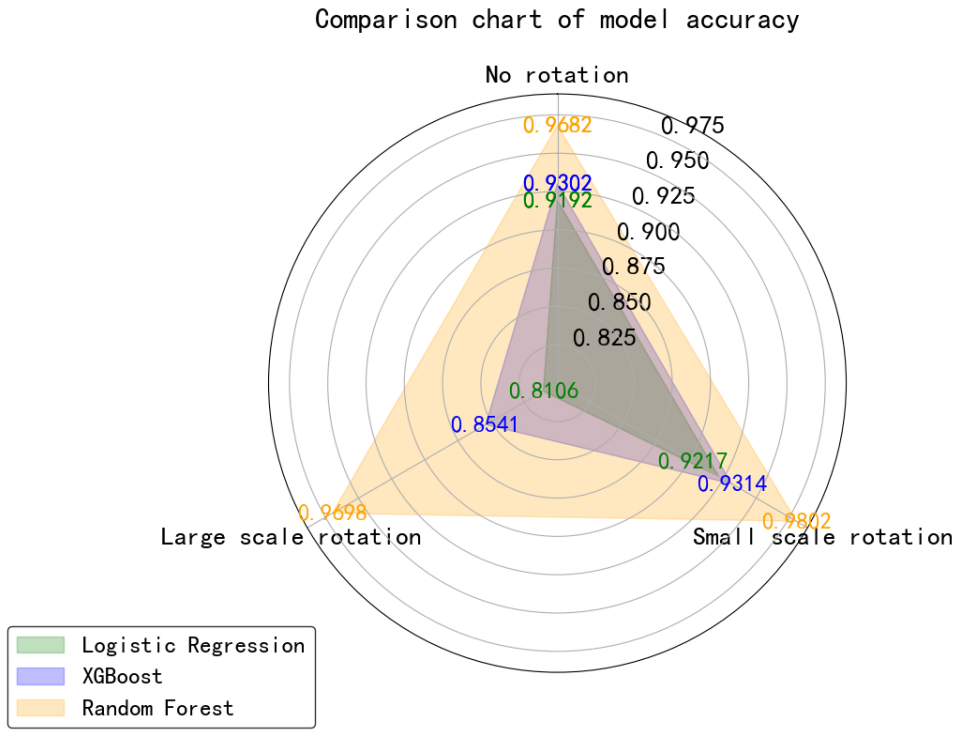

Figure 3: 准确率雷达图

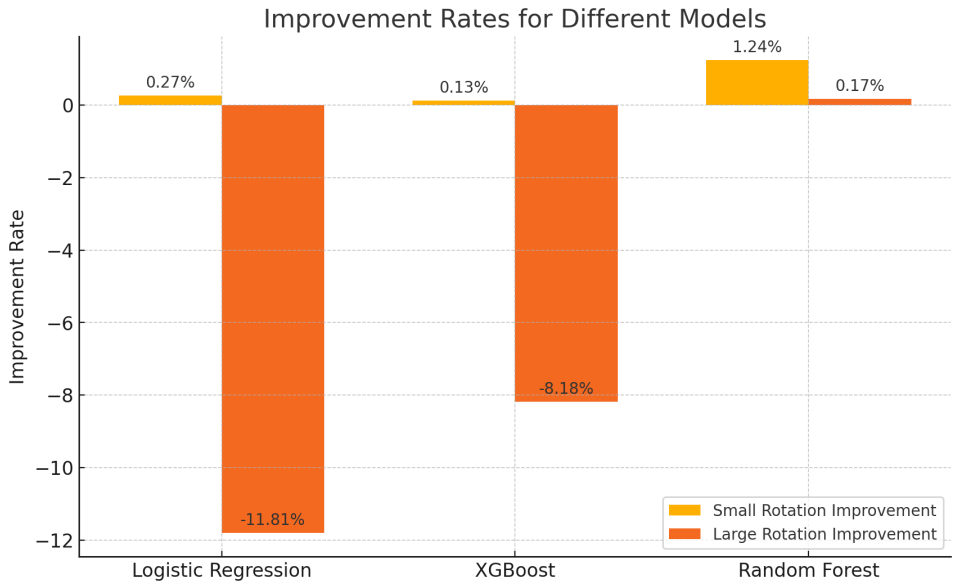

Figure 4: 准确率变化百分比直方图

4 结果分析

4.1 旋转增强对模型性能的影响

4.1.1 大范围旋转数据集

在大范围旋转数据集上，所有模型的性能均有所下降。具体来说，Logistic Regression 的准确率下降了 11.81%，XGBoost 的准确率下降了 8.18%，而 Random Forest 的准确率虽然也有所提升（+0.17%），但表现仍然较为稳定，反映了其对大范围旋转的适应能力。

4.1.2 小范围旋转数据集

小范围旋转数据集上，所有模型的准确率均有所提升。具体提升幅度依次为：Random Forest（+1.24%）、Logistic Regression（+0.27%）、XGBoost（+0.13%）。Random Forest 的提升最为显著，表明它对小范围旋转增强的数据更加敏感。

4.1.3 结论

从实验结果来看，大范围旋转数据集对所有模型的表现都有较大负面影响，尤其是对 Logistic Regression 和 XGBoost 的影响最为明显。而小范围旋转增强对模型性能的提升更为显著，尤其是对 Random Forest 的影响较大，表明小范围旋转增强能够有效提高模型对轻微变化的适应能力。

4.2 为什么大范围旋转会有这样的影响？

Logistic Regression 假设特征与标签之间存在线性关系。然而，大范围旋转会导致图像中像素值的空间分布发生显著变化，打破了这种线性假设。在这种情况下，模型难以捕捉到旋转后的数据特征，从而造成准确率的显著下降。

XGBoost 是基于决策树的梯度提升模型，能够处理非线性特征。尽管 XGBoost 在处理非线性数据时具有较好的鲁棒性，但大范围旋转引入的扰动依然对特征分布产生了较大影响。特别是大旋转角度会使得图像的特征在训练集中难以被有效区分，从而影响模型性能。

Random Forest 由多个决策树构成，其集成特性赋予了模型较强的鲁棒性。尽管大范围旋转会影响特征的分布，但由于 Random Forest 的随机性和多样性，多个树的集成有助于缓解这类问题。即使在大范围旋转的数据上，模型仍能保持较为稳定的性能，甚至略有提升。

4.2.1 总结

大范围旋转数据集显著扰乱了图像的结构和特征分布，特别是对线性模型（如 Logistic Regression）的影响较大。即使是非线性模型（如 XGBoost），也会受到较大旋转角度带来的数据扰动的影响。

4.3 为什么小范围旋转会有这样的影响？

小范围旋转通常不会破坏图像的结构，仍保持一定的规律性。对于 Logistic Regression 来说，这种微小的变化并不违反其线性假设，相反，适度的数据扰动反而有助于模型更好地学习到更普遍的特征。因此，小范围旋转增强可以轻微提升模型性能。

XGBoost 是基于决策树的模型，对于小范围旋转的数据，特征变化较小，因此能较好地捕捉到旋转后的特征，从而提升性能。此外，XGBoost 本身具有较强的泛化能力，能够较好地适应这些小扰动数据，进而提高分类准确性。

Random Forest 的集成特性使得它对小范围扰动特别敏感。对于小范围旋转数据，训练集的多样性得到增加，使得模型能够学习到更多的有用信息，从而进一步提高准确率。特别是在这种情况下，每棵树都能从不同的旋转角度中学习到不同的特征，最终提升整个模型的表现。

4.3.1 总结

小范围旋转增强对所有模型的表现均有正向影响，尤其是在增加数据多样性和训练集覆盖面的同时，能有效提升模型的泛化能力。相对于大范围旋转扰动，小范围旋转更适合数据增强，在保持原始数据分布的同时，增加了训练数据的多样性。

5 结论

通过对比和分析不同旋转范围的数据增强效果，我们发现：

- 大范围旋转对模型性能的影响较大，尤其是对于线性模型，扰动幅度越大，模型性能下降越明显。
- 小范围旋转则有助于提升模型性能，尤其是对于非线性模型（如 Random Forest 和 XGBoost），能够通过增强数据的多样性来提高模型的泛化能力。

6 不足与改进

6.1 不足之处

- 旋转角度的选择有限：在实验中，我们选择了固定的旋转角度（例如，顺时针和逆时针旋转角度为 [-30°, -20°, -10°, 0°, 10°, 20°, 30°]），这些角度虽然能覆盖一定的旋转范围，但可能并不能完全反

映手写数字的多样性。例如，某些特定的角度或更精细的旋转步长可能对模型的训练有更大的影响。

改进建议：可以考虑使用更丰富的旋转角度，甚至结合不同的图像预处理技术（如随机旋转），从而提高模型的鲁棒性和泛化能力。

- 数据增强范围较为单一：本实验主要采用了旋转数据增强，但实际应用中，手写数字的形态变化可能远不止旋转。其他如平移、缩放、模糊等图像变换可能对提升模型的性能也具有一定作用。
  改进建议：在未来的实验中，可以结合多种数据增强方法（如平移、缩放、裁剪、加噪声等），进一步扩大数据集的多样性，提升模型对手写数字的适应能力。

- 模型训练时间与资源消耗：虽然 Random Forest 和 XGBoost 在大范围旋转数据集上的表现优异，但由于它们本身是基于树的集成方法，训练过程通常较为耗时，并且随着数据集规模和模型复杂度的增加，训练时间和资源消耗可能进一步增加。
  改进建议：考虑使用更高效的算法，如深度神经网络（CNN），或采用分布式计算框架来加速训练过程。此外，可以尝试使用模型压缩技术来减小模型体积，提高推理速度。

- 没有对比不同的神经网络模型：本次实验中并未对比不同的深度学习模型（如卷积神经网络、全连接神经网络等）在旋转数据集上的表现。虽然树基模型表现较好，但卷积神经网络（CNN）在图像分类任务中通常能取得更好的性能，尤其是在处理包含旋转和其他变换的图像时。
  改进建议：未来可以在相同数据集上对比多种深度学习模型（如 CNN、ResNet 等），从而获得更加全面的评估。

- 未考虑模型超参数调优：在本实验中，我们使用了默认的超参数配置来训练模型。虽然默认配置可以提供一定的基准性能，但并没有进行详细的超参数调优。因此，模型的表现可能并未达到最优。
  改进建议：未来可以进行系统的超参数优化（如网格搜索、随机搜索等），以进一步提升模型的表现。

6.2 改进方向

- 结合更多的数据增强技术：目前实验只使用了旋转数据增强，但实际应用中，手写数字可能会有更多类型的变化。结合更多的增强技术，如平移、缩放、模糊、剪切等，可以更好地模拟实际情况，并提升模型的鲁棒性。
  使用混合增强技术（例如同时进行旋转和平移等操作），以增加数据的多样性，提升模型的性能。

- 引入深度学习模型进行对比：传统机器学习模型（如 Random Forest 和 XGBoost）在处理旋转数据集时表现出色，但深度学习模型（如卷积神经网络）通常能在图像识别任务中取得更好的表现。通过引入卷积神经网络（CNN）或更复杂的深度残差网络（ResNet），可以进一步提高模型在旋转数据集上的表现。
  可以通过预训练模型（例如使用 ImageNet 上预训练的 ResNet 或 VGG）进行迁移学习，进一步提升模型在手写数字识别任务中的准确性。

- 多样化旋转角度与扰动类型：在旋转角度的选择上，可以引入更多的角度，并结合其他形变方式（如模糊、缩放、平移等）进行数据增强，从而使训练集更加丰富。这有助于模型学习到更多样化的特征，提高其泛化能力。
  可以通过随机数据增强（例如随机旋转、缩放、翻转等）生成更多样的训练样本，以模拟手写数字在现实世界中的多种变形情况。

- 并行化与加速训练过程：随着数据集的增大和模型复杂度的提升，训练时间和计算资源消耗也随之增加。未来可以通过分布式训练或多 GPU 训练来加速模型训练过程。同时，可以考虑使用模型压缩或量化技术，以减少模型的计算量，提高推理速度。

- 细致的超参数优化：为了最大化模型的性能，未来可以对模型进行更细致的超参数调优。通过网格搜索或贝叶斯优化等方法，可以找到最优的超参数配置，以提升模型的准确性。