浙江大学
ZHEJIANG UNIVERSITY

人工智能基础实验报告

作业名称 实验报告五

姓　　名 叶畅飞

学　　号 3240103132

电子邮箱 yeelysia@qq.com

联系电话 19557031080

指导老师 钟先平

2025 年 4 月 29 日

实验背景和目的

随着深度学习技术的飞速发展，神经网络在图像分类、语音识别、自然语言处理等领域取得了显著的成果。多层感知机（MLP）和卷积神经网络（CNN）是两种经典的神经网络架构，广泛应用于各种任务中。

MLP 是一种前馈神经网络，由输入层、一个或多个隐藏层和输出层组成。它通过全连接的方式将输入数据映射到输出结果，适用于处理结构化数据。然而，在处理图像数据时，MLP 面临着一些挑战。图像数据具有高维度和复杂的空间结构，MLP 需要大量的参数来学习这些结构，导致计算成本高、训练时间长，且容易过拟合。

CNN 是为处理具有网格状拓扑结构的数据而设计的神经网络，主要包含卷积层、池化层和全连接层。它通过局部连接和权重共享的方式减少了参数数量，同时能够提取图像的空间特征。CNN 在图像分类任务中表现出色，能够自动学习图像的层次化特征表示，具有很强的泛化能力。

本实验旨在通过对多层感知机（MLP）和卷积神经网络（CNN）的实现、训练和评估，深入理解两种模型的结构特点、性能差异以及适用场景。从基础模型开始，逐步探索更复杂的网络架构，最终通过对比分析，掌握深度学习模型设计与评估的关键技能。具体目标包括：

1. 掌握 MLP 和 CNN 的基本原理和实现方法。
2. 了解不同网络结构对模型性能的影响。
3. 学习深度学习模型训练、评估和可视化的方法。
4. 通过对比实验，理解不同模型在图像分类任务中的优缺点。
5. 培养深度学习模型调优和问题解决的能力。

目　　录

实验背景和目的 I
1　实验原理简述 1
1.1　多层感知机（MLP） 1
1.2　卷积神经网络（CNN） 1
2　实验过程描述 2
2.1　实验环境搭建 2
2.2　实验步骤 2
3　实现代码 3
4　实验结果与分析 4
4.1　思考问题 1 4
4.2　思考问题 2 4
4.3　任务 1： 5
5　创新探索任务的设计、实现与结果 6
6　结论与思考 7
参考文献 8

1 实验原理简述

1.1 多层感知机（MLP）

多层感知机（MLP）是一种经典的前馈神经网络，其主要特点如下：

- 全连接结构：MLP 的每一层神经元都与下一层的所有神经元相连。输入层接收输入数据，经过一个或多个隐藏层的处理，最终输出层产生分类结果。
- 非线性激活函数：为了引入非线性特性，MLP 在每一层的输出上应用非线性激活函数，如 ReLU（Rectified Linear Unit）、Sigmoid 或 Tanh 等。这些激活函数使得网络能够学习复杂的非线- 性关系。
- 反向传播训练[1]：MLP 通过反向传播算法进行训练。在训练过程中，网络的输出与真实标签之间的误差通过损失函数计算，然后通过反向传播将误差逐层传递回网络的输入层，并根据梯度下降法更新网络的权重，以最小化损失函数。

在本实验中，MLP 模型的实现包括：

- SimpleMLP：单隐层 MLP，结构简单，适合初步理解 MLP 的工作原理。
- DeepMLP：多隐层 MLP，通过增加隐藏层的数量来提高模型的表达能力，同时引入 BatchNorm 和 Dropout 技术来改善训练性能和防止过拟合。
- ResidualMLP：带有残差连接的 MLP，通过残差学习解决深层网络训练中的梯度消失问题。

1.2 卷积神经网络（CNN）

卷积神经网络（CNN）是专门为处理具有网格状拓扑结构的数据（如图像）而设计的神经网络[2]，其主要特点如下：

- 局部连接：CNN 的每个神经元只与输入数据的一个局部区域连接，这种局部连接方式使得网络能够捕捉到图像的局部特征。
- 权重共享：同一特征图的所有神经元共享相同的权重，这大大减少了模型的参数量，同时提高了模型的泛化能力。
- 多层次特征提取：CNN 通过堆叠多个卷积层和池化层，逐步提取图像的层次化特征。低层卷积层通常检测简单的边缘、纹理等特征，而高层卷积层则组合这些简单特征形成更复杂的表示。
- 池化层：池化层用于降低特征图的空间维度，同时保留重要的特征信息。常用的池化操作包括最大池化（Max Pooling）和平均池化（Average Pooling）。
- 全连接层：在 CNN 的最后阶段，通常会将卷积层和池化层提取的特征图展平为一维向量，然后通过全连接层进行分类。

在本实验中，CNN 模型的实现包括：

- SimpleCNN：简单的 CNN，包含两个卷积层和一个池化层，适合初步理解 CNN 的基本结构和工作原理。
- MediumCNN：中等复杂度的 CNN，增加了更多的卷积层和池化层，并引入 BatchNorm 和 Dropout 技术来改善模型的性能。
- VGGStyleNet：基于 VGG 架构的 CNN，使用小尺寸（3×3）卷积核和 2×2 最大池化层，通过堆叠多个卷积层来增加网络深度，具有规整的结构。
- SimpleResNet：简化的 ResNet，包含残差连接，通过残差学习解决了深层网络训练中的梯度消失问题，使得训练更深的网络成为可能。

2 实验过程描述

2.1 实验环境搭建

本实验使用了 Python 编程语言和 PyTorch 深度学习框架进行模型的实现和训练。本地实验环境配置如下：

- 操作系统: Arch Linux
- Python 版本: 3.10.16
- PyTorch 版本: 2.7.0+cu128
- CUDA 版本: 12.8
- 数据集: CIFAR-10
- 其他依赖库: NumPy, Matplotlib, scikit-learn 等

2.2 实验步骤

2.2.1 第一部分：基础 MLP 模型

2.2.1.1 了解 MLP 模型结构

查看 models/mlp.py 文件，理解三种 MLP 模型的结构：

- SimpleMLP：单隐层 MLP。
- DeepMLP：多隐层 MLP，带有 BatchNorm 和 Dropout。
- ResidualMLP：带有残差连接的 MLP。

3 实现代码

4 实验结果与分析

4.1 思考问题 1

1. 数据结构
   - 空间信息丢失: MLP 需要将输入图像展平为一维向量 (例如, 3x28x28 的 RGB 图像变为 2352 维向量), 破坏了像素间的空间局部性 (如相邻像素的关联性)。这种结构忽略了图像的 2D/3D 拓扑关系，导致模型难以捕捉边缘、纹理等局部模式。
   - 平移不变性缺失: 图像中的目标无论位于哪个位置都应被识别，但 MLP 对输入顺序敏感（例如，平移后的图像会被视为完全不同的输入），需依赖大量数据隐式学习平移不变性，效率低下。
2. 参数量
   - 全连接的高计算成本：MLP 的层间全连接导致参数量爆炸。例如，处理 224x224 的 RGB 图像时，输入层到第一隐藏层（假设隐藏层大小 1024）的参数量为 224x224x3x1024 ≈ 154M，远超 CNN 的局部连接方式。
   - 过拟合风险：高参数量需要极大训练数据量来避免过拟合，而真实图像数据集通常标注成本高昂，难以满足 MLP 的需求。
3. 特征提取能力
   - 手工设计特征的局限性: 传统 MLP 依赖人工设计特征（如 SIFT+HOG+MLP 组合），但图像的低级（边缘、角点）到高级特征（物体部件、整体）需分层自动提取。MLP 缺乏层级归纳偏置，难以像 CNN 那样通过卷积核分层捕获特征。
   - 全局依赖与局部感知的矛盾: MLP 理论上可通过深层网络拟合任意函数，但实际训练中难以同时兼顾局部细节（如纹理）和全局语义（如物体类别）。而 CNN 通过卷积-池化-深层的结构天然支持这一过程。

4.2 思考问题 2

1. 局部感知
   - 操作方式：卷积核（如 3x3）在图像上滑动，每次仅计算局部区域（如 3x3 像素块）的加权和，而非 MLP 的全连接。
   - 保留空间结构：像素的邻域关系（如边缘、纹理）被显式保留，避免展平操作导致的空间信息破坏。
2. 权重共享
   - 操作方式：同一卷积核在整个图像上共享参数，无论检测目标位于图像中心还是角落。

- 平移不变性：相同模式（如猫耳）在不同位置会被同一核识别，无需 MLP 那样为每个位置学习独立参数。

3. 层次化特征提取
   - 低级→高级特征：
     - 浅层卷积：捕捉边缘、颜色等局部特征。
     - 深层卷积：组合局部特征为高级语义（如物体部件、整体类别）。
   - 空间信息传递：通过池化（如 2x2 Max Pooling）逐步降低分辨率，但保留主要空间结构。

4.3 任务 1：

```python
from torch.nn import nn

class TwoLayerMLP(nn.model):

  def __init__(self,input_dim=3*32*32):
    super(TwoLayerMLP,self).__init__()
    self.flatten = nn.Flatten()

    # 第一层
    self.fc1 = nn.Linear(input_dim, 1024)
    self.bn1 = nn.BatchNorm1d(1024)

    # 第二层
    self.fc2 = nn.Linear(1024, 512)

    # 激活和Dropout
    self.relu = nn.ReLU()
    self.dropout = nn.Dropout(dropout_rate)

  def forward(self, x):
    x = self.flatten(x)

    # 第一层
    x = self.fc1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.dropout(x)

    # 第二层
    x = self.fc2(x)

    return x
```

5 创新探索任务的设计、实现与结果

6 结论与思考

参考文献

[1] Ronald J. Williams David E. Rumelhart Geoffrey E. Hinton. Learning representations by back-propagating errors. Nature, 1986, 323: 533-536

[2] Geoffrey E. Hinton Alex Krizhevsky Ilya Sutskever. ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2012, 25(2): 1097-1105