**深度学习实验指导书**

**实验一 前馈神经网络（FNN）**

1. **实验目的和要求**
2. 掌握前馈神经网络的结构；
3. 构建和训练一个简单的多层感知机来解决分类问题。
4. **实验工具**

Python编程环境；

神经网络库，如TensorFlow或PyTorch。

1. **实验内容**

基于公开的MNIST手写数字数据上的分类问题，构建一个包含至少一个隐藏层的前馈神经网络，其中包括选择适当的激活函数和损失函数实现训练模型，并记录训练过程中损失和准确率等实验结果。具体实验步骤如下：

1.下载公开的MNIST手写数字数据集。

（1）使用python脚本下载数据集



（2）

目录下包含两个文件夹，mnist 文件夹是 torchvision 用来存放下载的原始数据集文件的，而 mnist\_jpg 文件夹是脚本用来存放将原始数据转换并保存为 JPEG 格式图片的地方。



train-images-idx1-ubyte.gz：包含训练集的图像数据，格式为压缩的二进制文件。每个图像是 28x28 像素的灰度图像。

train-labels-idx1-ubyte.gz：包含训练集的标签数据，格式为压缩的二进制文件。每个标签对应一个图像，表示该图像所代表的数字（0-9）。

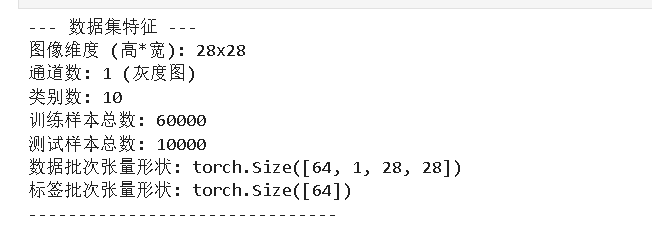
t10k-images-idx1-ubyte.gz：包含测试集的图像数据，格式与训练集相同。

t10k-labels-idx1-ubyte.gz：包含测试集的标签数据，格式与训练集相同。

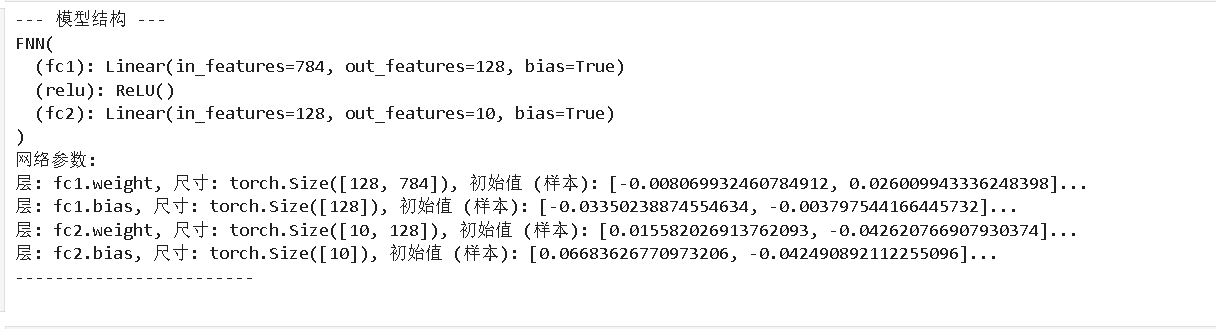
这些文件是 MNIST 数据集的标准格式，通常用于训练和测试机器学习模型。脚本会自动下载这些文件并解压，以便后续处理和使用。



2.学习和简述数据集的特征，包括数据的维度、类别数等。



3.构建一个包含至少一个隐藏层的前馈神经网络。



4.初始化网络参数，选择适当的激活函数，如ReLU函数。

# --- 初始化网络参数 (PyTorch 层隐式完成) ---

# PyTorch 层 (如 nn.Linear) 有默认的初始化方法 (例如权重的 Kaiming uniform 初始化)。

激活函数选用ReLU

5.编写代码实现神经网络的训练过程，包括前向传播和反向传播。

6.选择适当的损失函数，如交叉熵损失。

7.应用梯度下降算法进行参数优化。

# --- 6. 损失函数 ---

criterion = nn.CrossEntropyLoss()

print(f"--- 损失函数: {type(criterion).\_\_name\_\_} ---")

# --- 7. 优化器 (梯度下降) ---

optimizer = optim.SGD(model.parameters(), lr=LEARNING\_RATE)

# optimizer = optim.Adam(model.parameters(), lr=LEARNING\_RATE) # 可选的优化器

print(f"--- 优化器: {type(optimizer).\_\_name\_\_} (学习率: {LEARNING\_RATE}) ---\n")

损失函数 ：交叉熵损失

优化算法 ：随机梯度下降

学习率LR：0.01

8.训练模型，并记录训练过程中的损失和准确率变化。

for i, (images, labels) in enumerate(train\_loader):

images = images.to(device)

labels = labels.to(device)

# 前向传播

outputs = model(images)

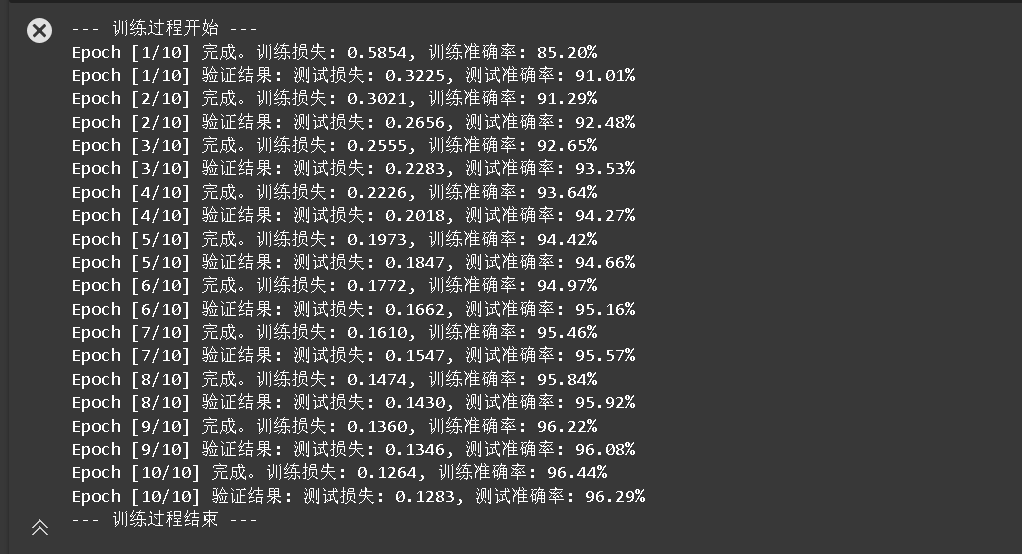
loss = criterion(outputs, labels)

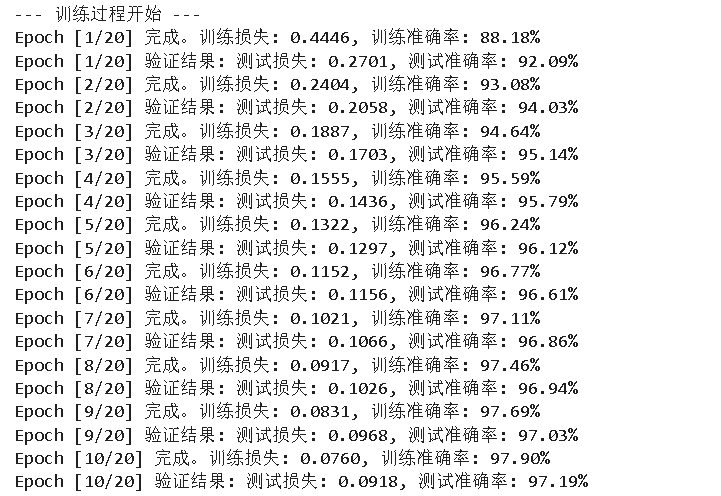
# 反向传播和优化

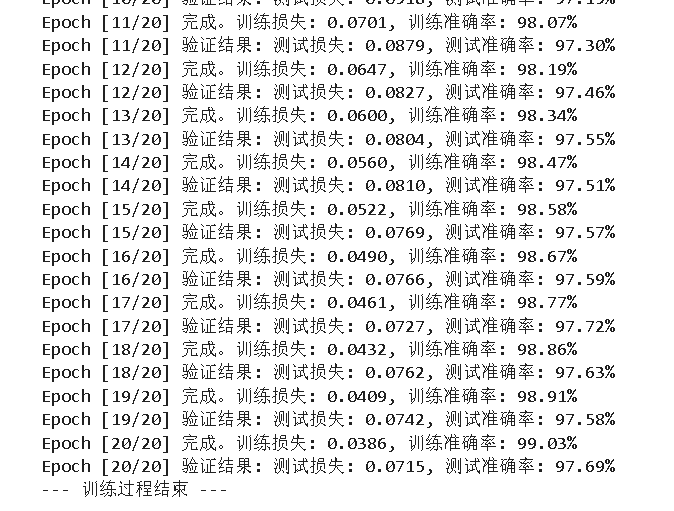
optimizer.zero\_grad() # 清除之前的梯度

loss.backward() # 计算梯度

optimizer.step() # 更新权重







9.评估模型在测试集上的性能。

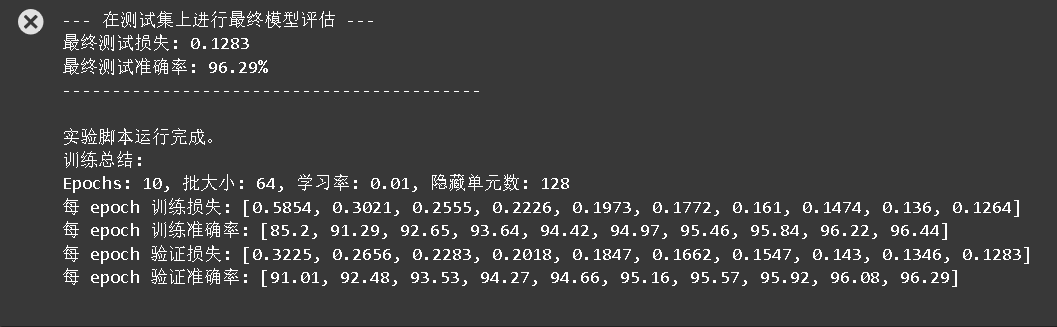
epoch\_val\_loss = running\_val\_loss / total\_val

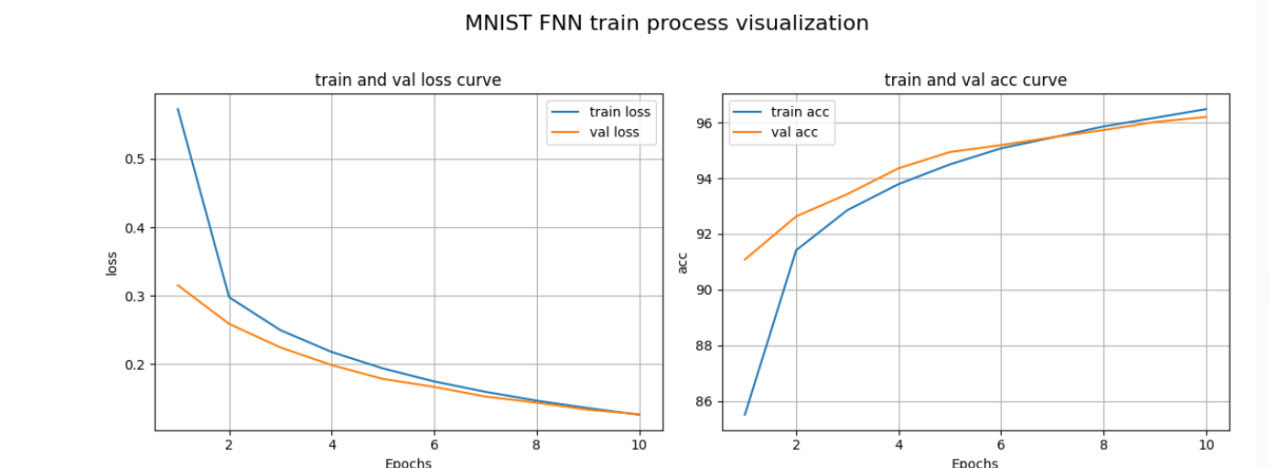
epoch\_acc\_val = 100 \* correct\_val / total\_val

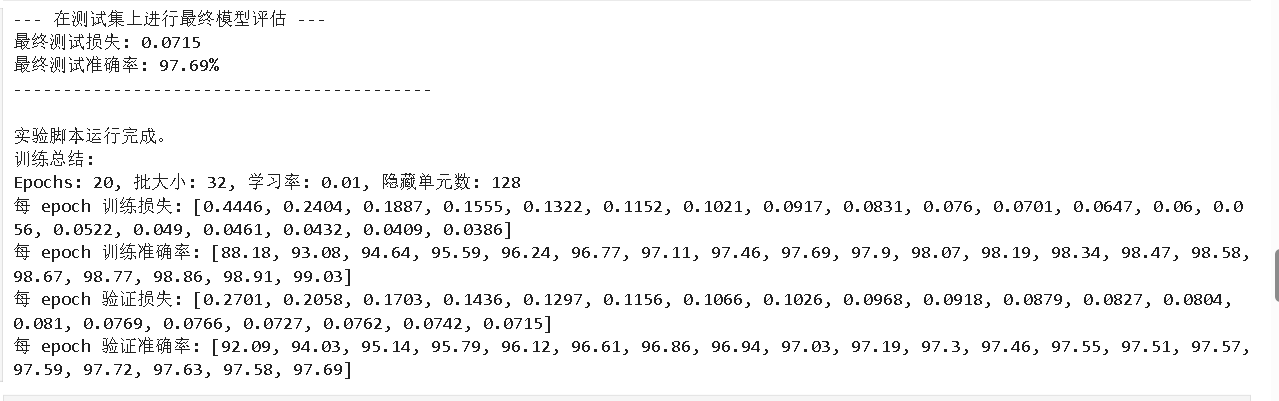
val\_losses.append(epoch\_val\_loss)

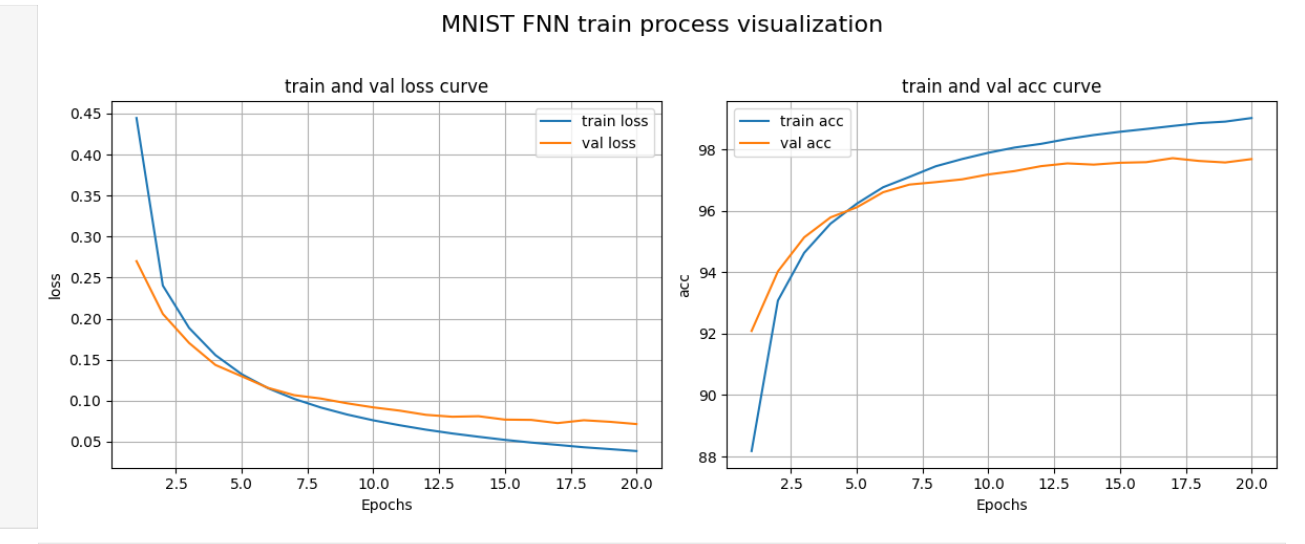
val\_accuracies.append(epoch\_acc\_val)

print(f'Epoch [{epoch+1}/{EPOCHS}] 验证结果: 测试损失: {epoch\_val\_loss:.4f}, 测试准确率: {epoch\_acc\_val:.2f}%')









1. **实验报告**

编写实验报告，总结实验结果以及可能的改进方法。

**实验二 卷积神经网络（CNN）**

1. **实验目的和要求**
2. 掌握卷积神经网络在图像处理中的应用；
3. 构建和训练一个基础的CNN来解决图像分类问题。
4. **实验工具**

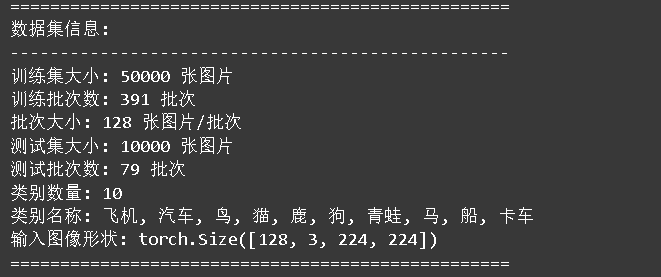
Python编程环境；

神经网络库，如TensorFlow或PyTorch。

1. **实验内容**

基于公开的CIFAR-10图像分类数据集上的图像分类问题，构建包含卷积层，激活函数，池化层和全连接层的卷积神经网络，通过尝试应用不同的优化器和调整超参数，记录模型在训练集和验证集上的性能对比等实验结果。具体实验步骤如下：

1. 下载公开的CIFAR-10图像分类数据集。
2. 分析和简述该数据集的特点。



CIFAR-10 数据集是一个广泛用于图像分类任务的公开数据集，具有以下特点：

（1）数据集规模：

训练集：包含 50,000 张彩色图像。

测试集：包含 10,000 张彩色图像。

总共有 60,000 张图像，分为 10 个类别。

图像类别：

10 个类别分别为：飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船、卡车。

每个类别在训练集和测试集中均匀分布，即每个类别有 5,000 张训练图像和 1,000 张测试图像。

（2）图像分辨率：

原始图像分辨率为 32x32 像素，每个图像为 RGB 彩色图像（3 通道）。

在脚本中，图像经过预处理（如随机裁剪、水平翻转）并调整大小为 224x224 像素，以适应 VGG16 的标准输入尺寸。

（3）预处理：

脚本中对图像进行了标准化处理，使用 CIFAR-10 的均值和标准差（(0.4914, 0.4822, 0.4465) 和 (0.2023, 0.1994, 0.2010)）进行归一化。

训练时还应用了数据增强技术（如随机裁剪和水平翻转），以提高模型的泛化能力。

（4）批次加载：

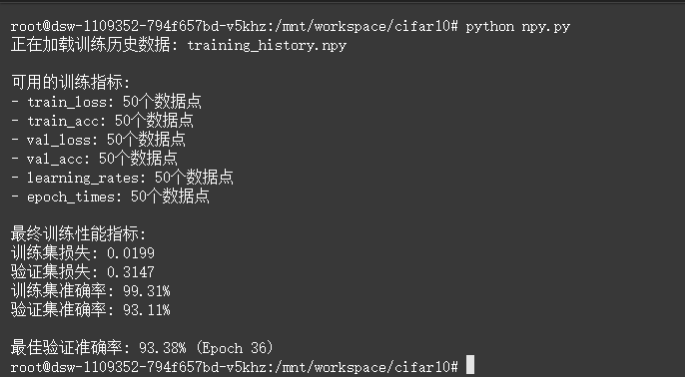
训练和测试数据通过 DataLoader 加载，支持分批处理。

训练数据在每个 epoch 中会被随机打乱（shuffle=True），而测试数据则保持固定顺序。

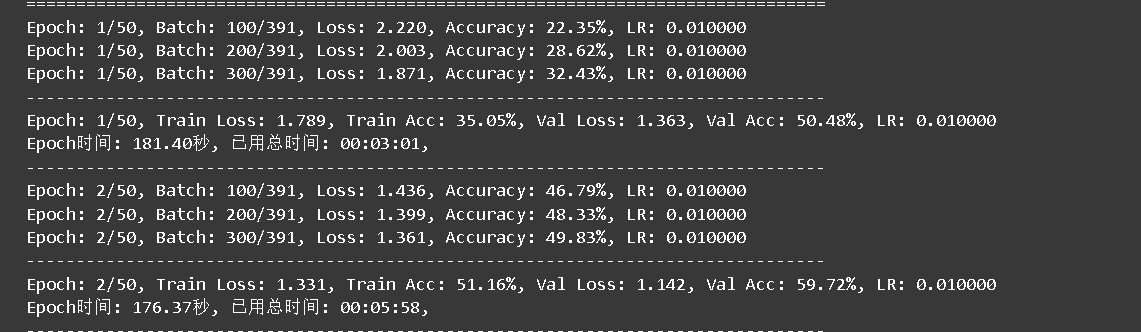
（5）输入形状：

经过预处理后，输入模型的图像形状为 [batch\_size, 3, 224, 224]，其中 3 表示 RGB 通道，224x224 是调整后的图像尺寸。

1. 构建一个基本的CNN结构，包含卷积层，激活函数，池化层和全连接层。
2. 初始化网络参数，训练模型。
3. 尝试应用不同的优化器和调整超参数。
4. 观察并记录模型在训练集和验证集上的性能。
5. 对模型进行评估和测试。

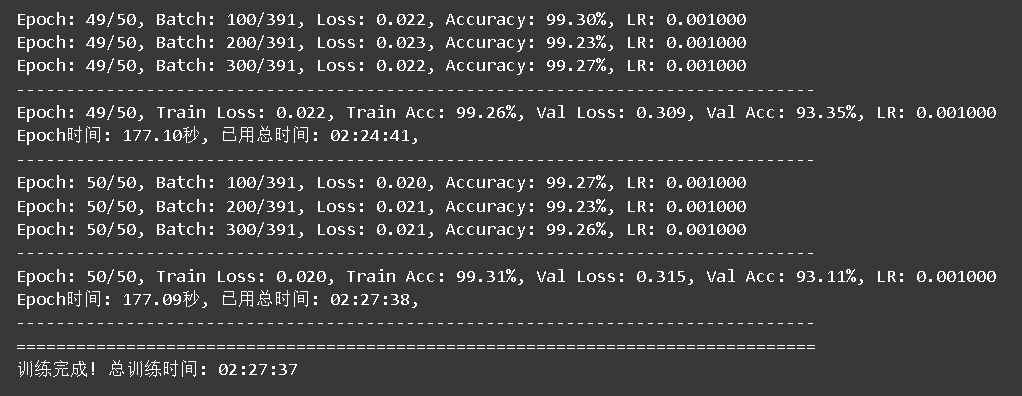






1. **实验报告**

编写实验报告，总结实验结果以及可能的改进方法。



**实验三 循环神经网络（RNN）**

1. **实验目的和要求**
2. 掌握循环神经网络在处理序列数据中的作用；
3. 构建和训练一个RNN来解决序列分类任务。
4. **实验工具**

Python编程环境；

神经网络库，如TensorFlow或PyTorch。

1. **实验内容**

基于公开的序列数据IMDb影评数据集上的情感分类问题，构建循环神经网络，可选用LSTM网络或者GRU。通过尝试应用不同的网络结构，观察对实验性能的影响，记录模型在训练集和验证集上的性能等实验结果。具体实验步骤如下：

1. 下载序列数据IMDb影评数据集。
2. 学习和简述数据集的特性。
3. 构建一个RNN模型，可选用LSTM或GRU单元。
4. 初始化网络参数，训练模型。
5. 探究不同的网络架构对结果的影响。
6. 观察模型在训练过程中的性能。
7. 评估模型在测试集上的性能。
8. **实验报告**

编写实验报告，分析结果和提出可能的改进方案。

**注意：每个实验分开提交实验报告，均参考后面的实验报告模板。**

**模板**

《深度学习实验》

实验报告

|  |  |
| --- | --- |
| 班级： |  |
| 姓名： |  |
| 学号： |  |

**实验一 \*\*\*\*\*\*\*\***

1. **实验目的与要求(标题1，微软雅黑，小四，加粗)**

**二、实验内容**

1. **实验方法**

**四、详细的模型设计及运行结果**

1. **模型设计及结果**
2. **模型的特色**

**五．实验总结**