实验1：多层神经网络的numpy实现

实验1：多层神经网络的numpy实现

姓名：XXX

学号：XXX

日期：YYYY/MM/DD

一、实验目的

1. 掌握全连接神经网络的前向传播与反向传播原理
2. 理解激活函数、损失函数、正则化方法的作用
3. 通过手动实现神经网络，深入理解深度学习框架底层机制
4. 探索超参数对模型性能的影响规律

二、实验要求

1. **网络架构示例**

```
Input(784) → FC(256) → ReLU → FC(128) → ReLU → Output(10) → Softmax
```

   - 输入层：MNIST图像展平为784维向量
   - 隐藏层：至少包含2个全连接层（FC），激活函数需支持ReLU/Sigmoid/Tanh
   - 输出层：10维Softmax输出对应0-9分类概率

2. **核心实现要求**

- 仅使用 `numpy` 库，禁止调用 `PyTorch/TensorFlow/Keras` 等框架
- 需手动实现以下组件：
  - 全连接层（前向/反向传播）
  - 激活函数（ReLU/Sigmoid及其导数）
  - 交叉熵损失函数
  - L2正则化项
  - Mini-batch梯度下降优化器

3. **性能要求**
   - 测试集准确率需达到95%以上
   - 需对比分析不同超参数组合的性能差异

三、实验步骤

1. **数据预处理**
   - 加载MNIST数据集（建议使用 `sklearn.datasets.fetch_openml`）
   - 像素值归一化至[0,1]，对标签进行One-hot编码
   - 按8:2划分训练集/验证集

2. **基础实现**

```python
class FullyConnectedLayer:
    def __init__(self, input_dim, output_dim, activation='relu'):
        self.W = np.random.randn(input_dim, output_dim) * 0.01
        self.b = np.zeros((1, output_dim))
        self.activation = activation

    def forward(self, X):
        # 实现前向传播

    def backward(self, dZ, X):
        # 实现反向传播
```

3. **参数调优实验**

实验组	隐藏层结构	激活函数	学习率	Batch Size	L2系数	验证集准确率
Baseline	[256,128]	ReLU	0.01	64	1e-4	
组1	[512,256]	LeakyReLU	0.005	128	1e-3	
组2	[128]	Tanh	0.02	32	0	

4. **进阶优化（可选）**
   - 实现动量（Momentum）或Adam优化器
   - 添加Dropout层
   - 尝试Xavier/He参数初始化

四、实验报告要求

1. **代码规范**
   - 模块化设计（需包含`model.py`，`train.py`，`utils.py`）
   - 关键代码段需有注释（如梯度计算部分）

2. **结果分析**
   - 训练/验证损失曲线图（需体现过拟合控制）
   - 混淆矩阵与分类错误案例可视化
   - 超参数敏感性分析（如学习率对收敛速度的影响）

3. **讨论内容**
   - 不同激活函数对梯度消失的影响
   - 正则化方法如何平衡偏差-方差
   - 网络深度与模型性能的关系

五、验收标准

1. 完整代码（.zip或GitHub链接）
2. 实验报告（PDF格式，5-8页）
3. 测试集准确率截图（需≥95%）

六、参考实现提示

1. 反向传播推导建议使用计算图法
2. 梯度检查可使用数值梯度验证：

```
grad_numerical = (f(x+eps) - f(x-eps)) / (2*eps)
```

通过本实验，学生可深入理解神经网络底层原理，为后续学习卷积神经网络奠定基础。