南開大學
Nankai University

网络空间安全学院

深度学习实验报告

实验一：前馈神经网络

姓名：武桐西

学号：2112515

专业：信息安全

指导教师：侯淇彬

2024 年 6 月 25 日

目录

1 实验目的

本次实验的主要内容是前馈神经网络（Feed-Forward Network, FFN），其主要代表网络为多层感知机（Multi-Layer Perceptron, MLP）。通过本次实验，需要掌握以下内容：

1. 掌握前馈神经网络（FFN）的基本原理。
2. 学会使用 PyTorch 搭建简单的 FFN 实现 MNIST 数据集分类。
3. 掌握如何改进网络结构、调试参数以提升网络识别性能。

本次实验的总揽如下：

1. 实现基础的 MLP 网络结构，并在 MNIST 数据集上训练和测试。
2. 实现笔者自己调整参数后改进的 MLP 网络结构，并在 MNIST 数据集上训练和测试。
3. 实现 MLP-Mixer 网络 [2] 结构，并在 MNIST 数据集上训练和测试。

2 实验环境

本实验在 macOS 系统下测试，搭配 M3 Max 芯片，使用 Python`3.11.5`解释器，PyTorch 版本为`2.2.0`（使用 MPS 进行硬件加速），如表1所示。

表 1: 实验环境

操作系统	macOS	硬件	M3 Max (支持 MPS)
Python	3.11.5	PyTorch	2.2.0 (支持 MPS)

本次实验的代码已上传GitHub。

3 文件目录结构

本次实验的文件目录结构如下：

```
MLP
├── data ..... Data Path
├── models ..... Models
│   ├── mlp_base.py ..... 基础的 MLP 模型
│   ├── mlp.py ..... 改进的 MLP 模型
│   └── mlp_mixer.py ..... MLP-Mixer 模型
├── trainer.py ..... Trainer
├── visualization.py ..... Visualization
└── main.py ..... Main Script
```

4 实验原理与实验过程

在本部分，首先介绍本次实验的目标任务和数据集，然后介绍三种前馈神经网络架构的实现，最后介绍整个实验的流程。

4.1 目标任务

本次实验需要实现**图像分类**任务，使用 MNIST 数据集进行**手写数字识别**，该数据集中的图像为灰度图像（只有一个通道），其形状为 28×28。

4.2 MLP Base

该基础网络由课堂教程中给出，由三层全连接层组成，使用 Dropout 进行正则化来抑制过拟合。网络结构如代码1所示。其他不再赘述。

代码 1: MLP Base Model Architecture

```
MLPBase(
  (fc1): Linear(in_features=784, out_features=100, bias=True)
  (fc1_drop): Dropout(p=0.2, inplace=False)
  (fc2): Linear(in_features=100, out_features=80, bias=True)
  (fc2_drop): Dropout(p=0.2, inplace=False)
  (fc3): Linear(in_features=80, out_features=10, bias=True)
)
```

4.3 MLP

笔者基于大量的实验调参，最终设计出如代码2所示的 MLP 网络。该网络由四层全连接层组成，使用 ReLU 激活函数，并采用 Dropout 正则化方法抑制过拟合。

代码 2: MLP Model Architecture

```
MLP(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_stack): Sequential(
    (0): Linear(in_features=784, out_features=512, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=512, out_features=512, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=512, out_features=128, bias=True)
    (7): ReLU()
    (8): Dropout(p=0.2, inplace=False)
    (9): Linear(in_features=128, out_features=10, bias=True)
  )
)
```

4.4 MLP-Mixer

MLP-Mixer[2] 是一种纯基于多层感知机（MLP）的视觉模型，由谷歌在 2021 年提出。该模型摒弃了传统卷积神经网络（CNN）和自注意力机制（如 Vision Transformer）中的卷积和自注意力操作，完全依赖 MLP 来实现图像分类任务，在图像分类任务上取得了与 CNN 和 Vision Transformer 相当的性能。其架构如下：

- **Token Mixing MLP**：该层在每个图像补丁之间混合信息，通过转置和 reshaping 操作来跨补丁混合特征。
- **Channel Mixing MLP**：该层在每个补丁的通道内混合信息，通过逐通道的 MLP 操作来混合特征。
- **两层 MLP**：每个 MLP 包含两层，通常带有 GELU 激活函数和 Dropout 层来增强模型的泛化能力。

其网络架构如代码3所示和图4.1所示。

代码 3: MLP-Mixer Model Architecture

```
MLPMixer(
  (patch_proj): Linear(in_features=49, out_features=256, bias=True)
  (token_mixing): Sequential(
    (0): Linear(in_features=16, out_features=16, bias=True)
    (1): GELU(approximate='none')
    (2): Linear(in_features=16, out_features=16, bias=True)
    (3): GELU(approximate='none')
  )
  (channel_mixing): Sequential(
    (0): Linear(in_features=256, out_features=256, bias=True)
    (1): GELU(approximate='none')
    (2): Linear(in_features=256, out_features=256, bias=True)
    (3): GELU(approximate='none')
  )
  (classifier): Linear(in_features=256, out_features=10, bias=True)
)
```

需要注意的是，笔者最初实现时并未添加**残差连接**，因此导致网络非常难以训练，在增加**残差连接**后，该情况得到解决。

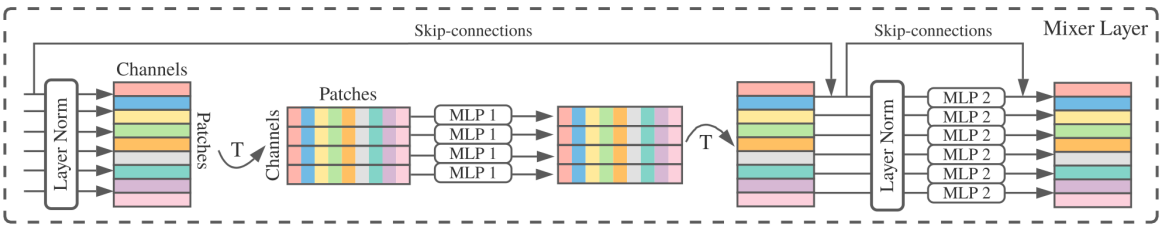


图 4.1: MLP-Mixer 网络架构 [2]

4.5 实验流程

本次实验的实验流程如下：

1. **数据预处理**：加载 MNIST 数据集，并进行数据预处理（转化为张量形式等）。
2. **模型选择**：根据命令行参数的解析选择要使用的模型。
3. **模型训练**：使用 Trainer 类进行模型训练。
4. **模型评估**：在验证集上评估模型，并保存模型。
5. **结果可视化**：绘制训练过程中的损失以及准确率曲线（训练集和验证集），并绘制混淆矩阵。

5 实验结果与分析

5.1 实验设置

本实验中，所有的超参数设置均相同，如表2所示。

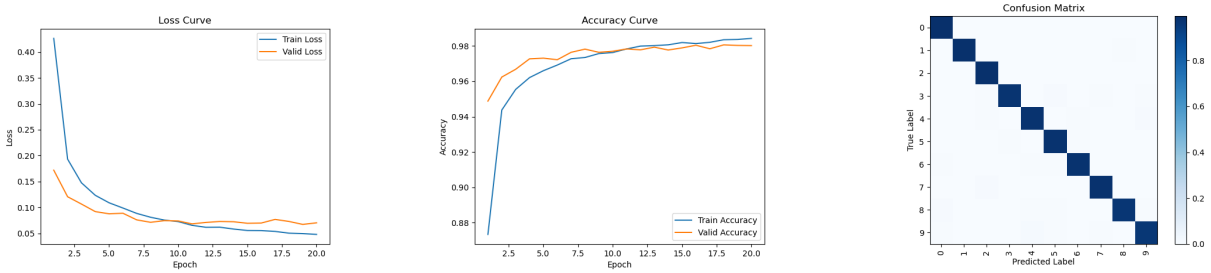
表 2: 实验设置

batch size	64	# of epoch	20
loss function	CrossEntropyLoss	learning rate	0.001
优化器	Adam	硬件加速	MPS

对于 MLP-Mixer，其 patch 的大小设置为 7×7，由于输入图像的形状为 $1 \times 28 \times 28$，因此一张图像被划分为 $4 \times 4 = 16$ 个 patch。

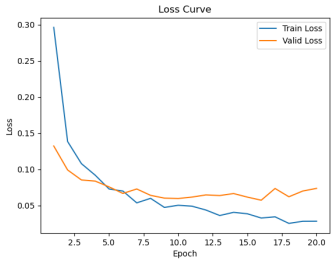
5.2 结果与分析

课堂教程中给出的基础 MLP 模型（MLP Base）、笔者自己设计的 MLP 模型（MLP）、MLP-Mixer 模型的实验结果分别如图5.2、图5.3和图5.4所示。实验结果的汇总对比如表3所示。

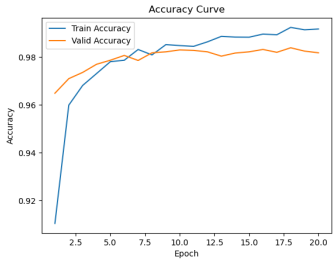


(a) 损失曲线 (b) 准确率曲线 (c) 混淆矩阵

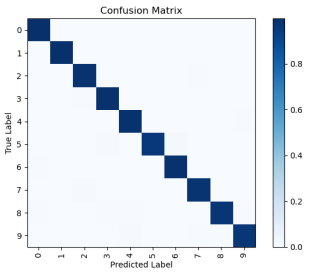
图 5.2: MLP Base Model 结果



(a) 损失曲线

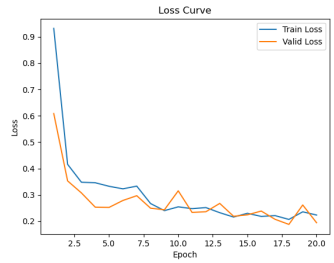


(b) 准确率曲线

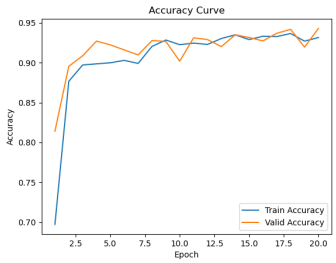


(c) 混淆矩阵

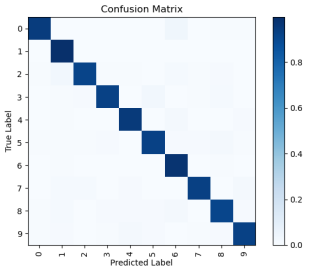
图 5.3: MLP Model 结果



(a) 损失曲线



(b) 准确率曲线



(c) 混淆矩阵

图 5.4: MLP-Mixer Model 结果

从图中可知，三种模型均达到了 94%+ 的验证集准确率，其中 MLP Base 与 MLP 模型更是达到了 98%+ 的验证集准确率。从损失曲线和准确率曲线上可以看出，训练集与验证集上的损失（或准确率）具有同步变化的趋势，先快速下降（上升），随后趋于平稳，逐渐收敛，三种模型均**不存在过拟合**现象。从混淆矩阵中可以看出，三种模型在各个类别上表现都非常出色，原因是该数据集比较简单、容易被简单的 MLP 模型学习到。

5.3 对比总结

表 3: 实验结果

Model	Accuracy
MLP Base	98.04%
MLP	98.39%
MLP-Mixer	94.30%

由表3可知，MLP-Mixer 的表现比经典的 MLP 模型略差，推测可能的原因是 MNIST 数据集过于简单，因此仅使用全连接即可较好地学习到数据特征，而 Token Mixing 与 Channel Mixing 操作比较复杂，反而制约了模型对数据的学习。

6 总结与体会

在本次实验中，笔者深入了解前馈神经网络（或多层感知机）的原理，并编程实现基础 MLP 和最近的 MLP-Mixer[2]，进一步熟悉了 PyTorch 框架的梯度优化等基本操作。

经过本次实验，笔者也有了一些**网络结构设计**和**调参**的经验与体悟：

- 在本实验的 MNIST 手写数字识别中，经典的 MLP 网络即可表现的非常好，比如三层或四层的多层感知机。
- 使用 Adam 优化器的效果一般要由于 SGD 优化器。
- 学习率与 batch size 的设置对本次实验（MNIST 手写数字识别）的模型表现影响不大。
- 网络架构并非越深越好、越复杂越好，比如 MLP-Mixer 的性能在简单数据集（MNIST）上的表现不如传统的 MLP。越复杂的网络在数据简单时越容易过拟合，而过于简单的网络在处理复杂数据时会欠拟合，因此需要针对不同的任务（数据集）选取不同复杂度的模型。
- 可以使用批归一化 BatchNorm、Dropout 等正则化方法抑制模型的过拟合现象。
- 从直观和经验上理解，模型的宽度与提取的特征数量有关，模型的深度与特征提取的抽象程度有关。

参考文献

[1] Qibin Hou, Zihang Jiang, Li Yuan, Ming-Ming Cheng, Shuicheng Yan, and Jiashi Feng. Vision permutator: A permutable mlp-like architecture for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):1328–1334, 2022.

[2] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[3] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):5314–5321, 2022.

[4] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. https://D2L.ai.