



南開大學

Nankai University

计算机学院

深度学习与应用实验报告

MLP 实验报告

学号：2112529

姓名：赵廷枫

2024 年 6 月 19 日

目录

1 FNN 原理概述	2
2 初始版本 MLP 测试	2
3 MLP 优化	3
3.1 调整超参数	3
3.2 增加模型宽度	3
3.3 增加模型深度	4
3.4 调整训练方式	5
3.4.1 增加 epoch	5
3.4.2 修改优化器	5
4 ResMLP 实现与测试	6
5 KAN vs MLP	8
6 实验总结	9

1 FNN 原理概述

前馈神经网络（Feedforward Neural Network, FNN）是一种最基本且常见的人工神经网络结构。它由多个神经元层组成，包括输入层、一个或多个隐藏层和输出层。每一层中的神经元与下一层中的神经元全连接，信息在网络中单向传播，不存在反馈环路。

关于该模型的具体迭代流程如下：

1. **前向传播 (Forward Propagation)**: 数据从输入层传递到输出层，每一层的神经元通过计算权重和偏置的线性组合，并应用激活函数来生成输出。前向传播过程中，信息一次性通过网络，没有反馈或循环。
2. **损失计算**: 通过损失函数（如均方误差、交叉熵等）计算预测结果与实际标签之间的差距。损失函数的选择取决于具体任务和目标。
3. **反向传播 (Backpropagation)**: 反向传播算法通过链式法则计算损失函数关于每个权重和偏置的梯度，并使用这些梯度来更新网络参数。常用的优化算法包括梯度下降、随机梯度下降 (SGD) 以及 Adam 等。
4. **权重更新**: 根据反向传播计算的梯度，优化算法更新网络中的权重和偏置，以最小化损失函数，逐步提高模型的预测性能。

2 初始版本 MLP 测试

初始版本的网络结构如下：

```
1 Net(  
2     (fc1): Linear(in_features=784, out_features=100, bias=True)  
3     (fc1_drop): Dropout(p=0.2, inplace=False)  
4     (fc2): Linear(in_features=100, out_features=80, bias=True)  
5     (fc2_drop): Dropout(p=0.2, inplace=False)  
6     (fc3): Linear(in_features=80, out_features=10, bias=True)  
7 )
```

使用该网络结构进行训练，得到的损失曲线以及准确率曲线如下：

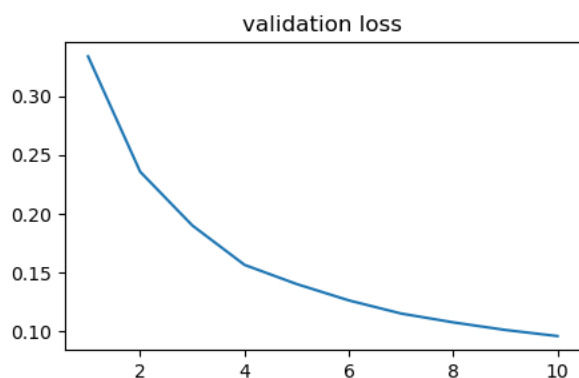


图 2.1: MLP 损失曲线

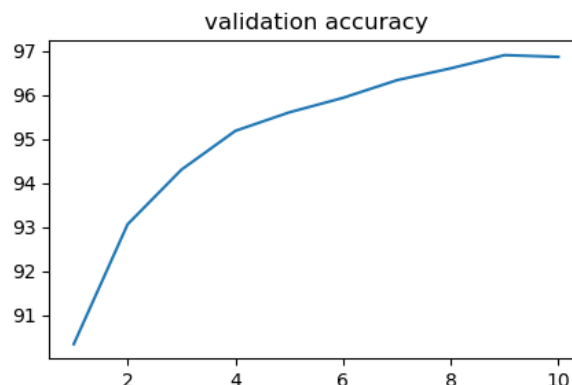


图 2.2: MLP 准确率曲线

分析：经过 10 个 epoch 的训练，最终的损失可以降低到 0.10 以下，准确率可以稳定在 97% 以上，取得了较为不错的效果。

3 MLP 优化

3.1 调整超参数

在这一部分，我主要是尝试修改随机梯度下降优化器的学习率，分别调整为：0.1、0.01、0.001 进行训练，然后进行实验，损失曲线以及准确率曲线如图所示：

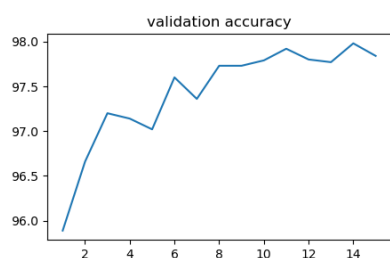


图 3.3: Lr=0.1

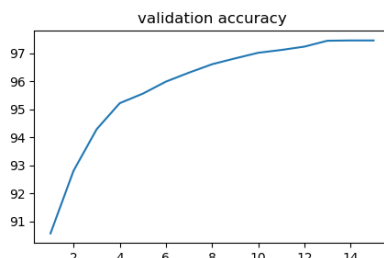


图 3.4: Lr=0.01

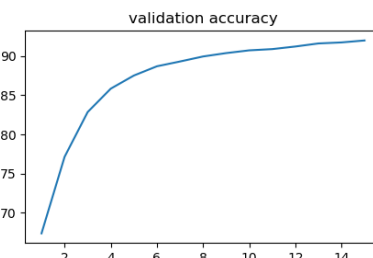


图 3.5: Lr=0.001

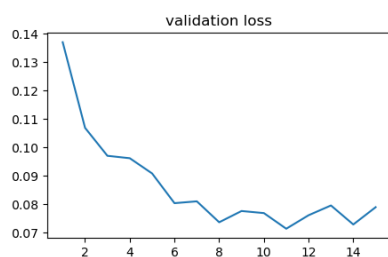


图 3.6: Lr=0.1

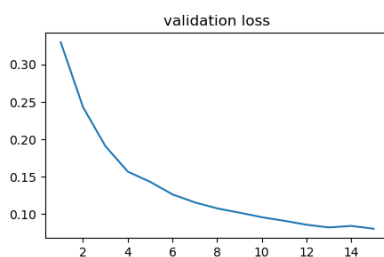


图 3.7: Lr=0.01

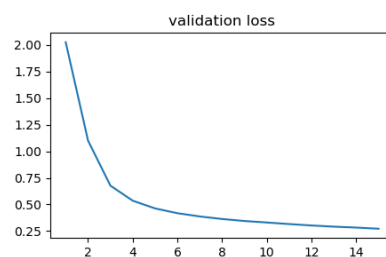


图 3.8: Lr=0.001

分析：从上述结果可以看出，学习率过大导致训练过程中出现局部过拟合，从而出现了震荡和不收敛、训练不稳定等现象，而学习率过小则导致收敛速度过慢、欠拟合问题，综上所述，学习率硬保持在一个合适的水平，0.01 针对随机梯度下降函数是一个较为合适的选择。

3.2 增加模型宽度

通过调整第一层线性层的输出特征数来增加模型的宽度，增加宽度之后的模型结构如下：

```

1 Net(
2   (fc1): Linear(in_features=784, out_features=200, bias=True)
3   (fc1_drop): Dropout(p=0.2, inplace=False)
4   (fc2): Linear(in_features=200, out_features=80, bias=True)
5   (fc2_drop): Dropout(p=0.2, inplace=False)
6   (fc3): Linear(in_features=80, out_features=10, bias=True)
7 )

```

使用该模型训练得到的损失曲线图和准确率曲线图如下：

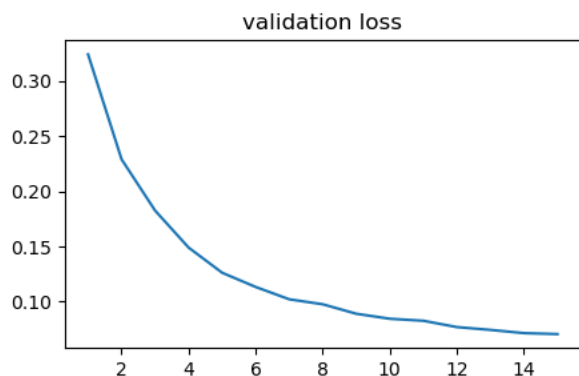


图 3.9: 增加模型宽度损失曲线

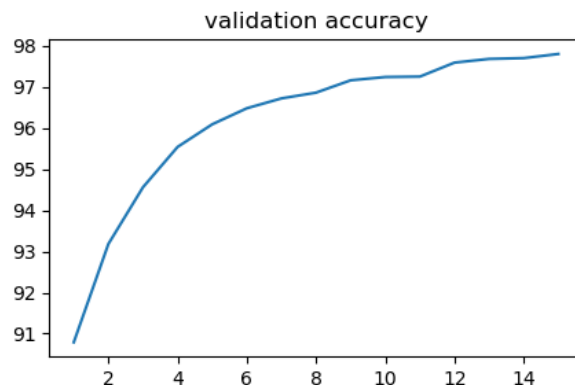


图 3.10: 增加模型宽度准确率曲线

分析：从实验结果中可以看出，增加宽度的模型训练之后的效果得到了提升，准确率提高且损失下降，有一定的改进效果。

3.3 增加模型深度

通过增加模型中隐藏层的数量来增加模型的深度，模型结果如下：

```

1 Net(
2   (fc1): Linear(in_features=784, out_features=100, bias=True)
3   (fc1_drop): Dropout(p=0.2, inplace=False)
4   (fc2): Linear(in_features=100, out_features=80, bias=True)
5   (fc2_drop): Dropout(p=0.2, inplace=False)
6   (fc3): Linear(in_features=80, out_features=40, bias=True)
7   (fc3_drop): Dropout(p=0.2, inplace=False)
8   (fc4): Linear(in_features=40, out_features=10, bias=True)
9 )

```

使用该模型训练得到的损失曲线图和准确率曲线图如下：

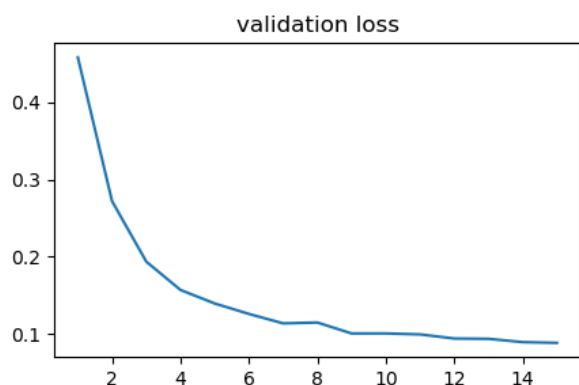


图 3.11: 增加模型深度损失曲线

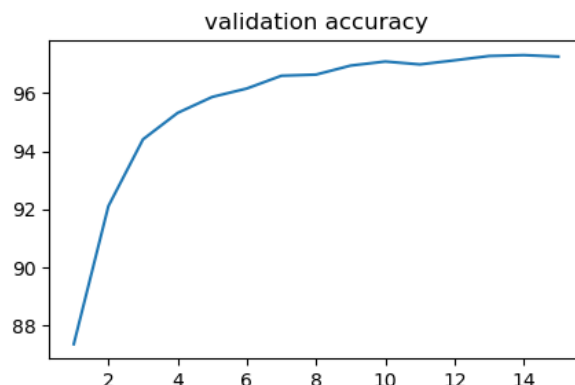


图 3.12: 增加模型深度准确率曲线

分析：从实验结果中可以看出，增加深度的模型训练之后的效果得到了提升，准确率提高且损失下降，有一定的改进效果。

3.4 调整训练方式

3.4.1 增加 epoch

我们调节 epoch 的大小，增加原先的 10 个 epoch 至 15 个 epoch，进行实验得到的损失和准确率变化曲线图如图所示：

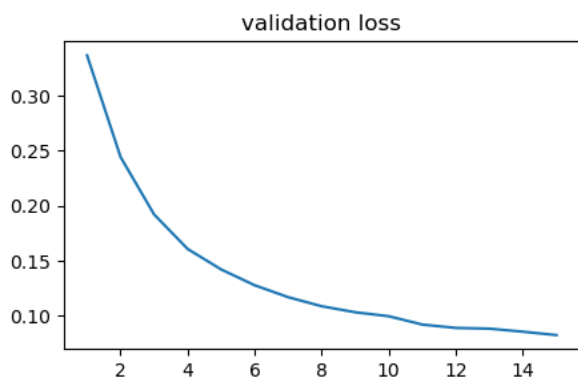


图 3.13: 增加 epoch 损失曲线

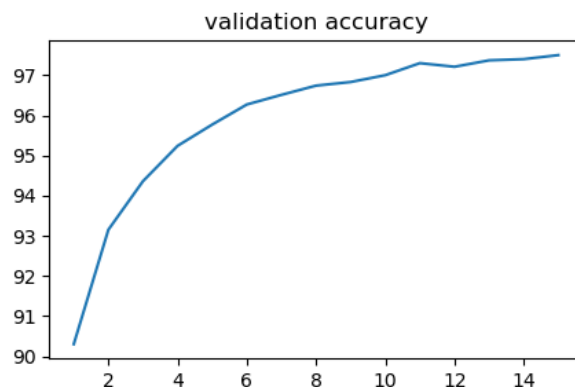


图 3.14: 增加 epoch 准确率曲线

从图中可以看出，虽然增加了 epoch，但是准确率提升幅度有限，损失也下降有限

3.4.2 修改优化器

我们使用 Adam 优化器来替换随机梯度下降，进行实验结果如图所示：

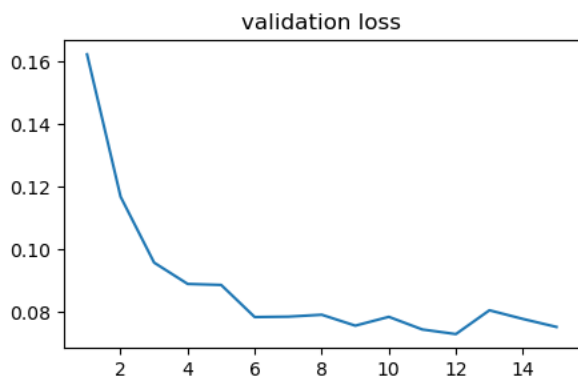


图 3.15: 使用 adam 优化器损失曲线

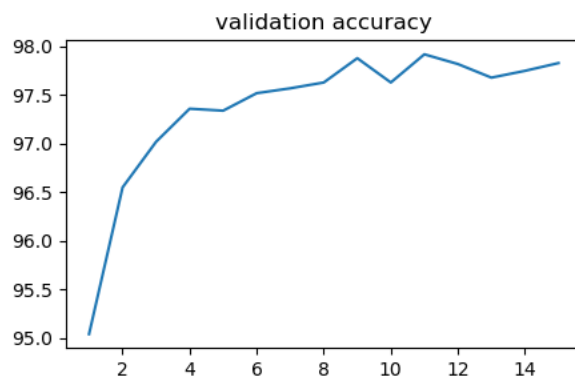


图 3.16: 使用 adam 优化器准确率曲线

可以看出 adam 优化器的总体效果与 SGD 基本持平，但是总体的训练过程存在更大的跌宕起伏，可能与 adam 优化器中的学习率设置有关，一般是使用较小的学习率可以充分发挥 adam 的优势。

4 ResMLP 实现与测试

基于 ResMLP 论文 [3], ResMLP 在传统的 MLP 模型的基础上引入了残差连接 (Residual Connections), 这种结构灵感来源于 ResNet [1]。

ResMLP 的目的是利用残差连接改进 MLP 的训练深度和效果, 使其在处理复杂的任务, 如图像分类时, 能够有更好的性能。

残差连接是一种通过将输入直接添加到输出的方法, 以帮助解决深度网络中的梯度消失和梯度爆炸问题。在 ResMLP 中, 这种结构允许模型在深层次中学习身份映射 (identity function), 从而能够训练更深的网络而不会导致性能下降。

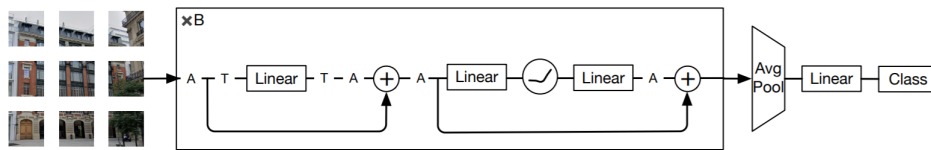


Figure 1: The ResMLP architecture: After flattening the patch into vectors, our network alternately processes them by (1) a communication layer between vectors implemented as a linear layer; (2) a two-layer residual perceptron. We denote by A the operator Aff , and by T the transposition.

图 4.17: ResMLP[3] 框架

理解上述论文中的原理之后, 我参考 [MNIST-ResMLP](#) 中的代码实现的 ResMLP 模型结构如下:

```

1 ResMLP(
2     (to_patch_embedding): Sequential(
3         (0): Conv2d(3, 384, kernel_size=(16, 16), stride=(16, 16))
4         (1): Rearrange('b c h w -> b (h w) c')
5     )
6     (mlp_blocks): ModuleList(
7         (0-11): MLPblock(
8             (pre_affine): Aff()
9             (token_mix): Sequential(
10                 (0): Rearrange('b n d -> b d n')
11                 (1): Linear(in_features=196, out_features=196, bias=True)
12                 (2): Rearrange('b d n -> b n d')
13             )
14             (ff): Sequential(
15                 (0): FeedForward(
16                     (net): Sequential(
17                         (0): Linear(in_features=384, out_features=1536, bias=True)
18                         (1): GELU(approximate='none')
19                         (2): Dropout(p=0.0, inplace=False)
20                         (3): Linear(in_features=1536, out_features=384, bias=True)
21                         (4): Dropout(p=0.0, inplace=False)
22                     )
23                 )
24             )
25             (post_affine): Aff()
26         )

```

```

27 )
28 (affine): Aff()
29 (mlp_head): Sequential(
30   (0): Linear(in_features=384, out_features=1000, bias=True)
31 )
32 )

```

使用 ResMLP 模型在 MNIST 训练和测试得到的损失曲线和准确率曲线如下：

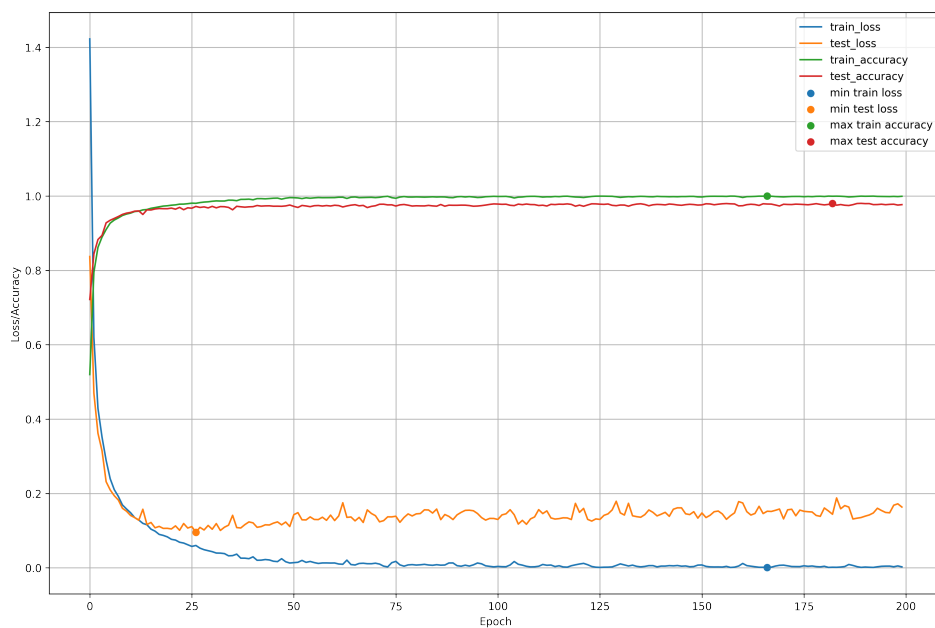


图 4.18: ResMLP 实验结果

通过实验测试得到的最高准确率为：98.09%，相较于原先的 MLP 在准确率上有了一定的提升，而且收敛速度较 MLP 也有了一定的提升。

5 KAN vs MLP

近期出现了一种新的理论范式：KAN: Kolmogorov-Arnold Networks[2]，有可能替代 MLP 从而进一步推进机器学习的发展。

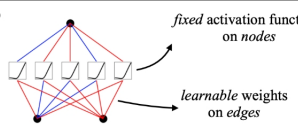
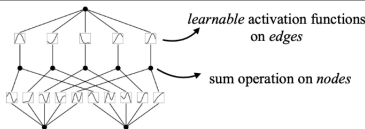
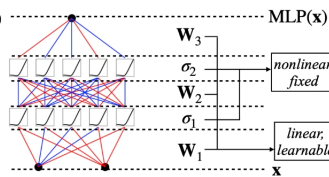
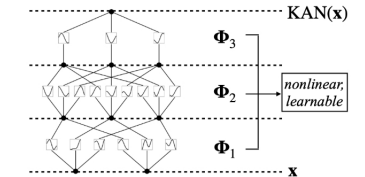
Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) 	(b) 
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	(c) 	(d) 

图 5.19: MLP 和 KAN 对比，图片来自 <https://github.com/KindXiaoming/pykan.git>

KAN 使用的是一种通过 B 样条 (B-splines) 来近似函数的方法。然而，B 样条在性能上较差，并且使用起来不是很直观。因此，SynodicMonth 在 **ChebyKAN** 尝试用切比雪夫多项式 (Chebyshev polynomials) 来代替 B 样条。切比雪夫多项式是一种定义在区间 $[-1, 1]$ 上的正交多项式，非常适合用于函数逼近，并且可以递归计算。

我这里参考的是上述仓库中的代码来解决 MNIST 数据集分类问题，训练过程的损失和准确率曲线如图所示：

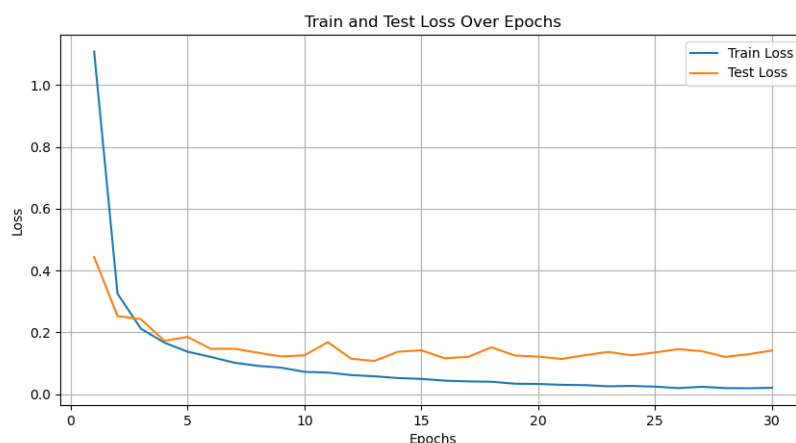


图 5.20: ChebyKAN 模型效果

分析：从上述结果中可以看出，在 MNIST 数据集上 KAN 的收敛速度比 MLP 更快，且最终的准确率基本上和 MLP 持平，由此可见，KAN 存在一定的潜力来代替 MLP，但是该模型的很多理论在实际中的应用还需要验证。

6 实验总结

本次实验对前馈神经网络 (FNN)、多层感知器 (MLP)、以及 Kolmogorov-Arnold Networks (KAN) 的不同实现进行了测试和比较。在对 MLP 的优化过程中, 发现学习率对训练效果有显著影响。学习率过大会导致训练过程中的震荡和不收敛, 而学习率过小则导致收敛速度过慢, 最终确定 0.01 是一个较为合适的选择。

通过引入残差连接的 ResMLP 在改进 MLP 的训练深度和效果方面表现出色。残差连接帮助模型在深层次中学习身份映射, 解决了深度网络中的梯度消失和梯度爆炸问题, 使得 ResMLP 能够处理更复杂的任务。

此外, 实验还尝试了使用切比雪夫多项式替代 KAN 中的 B 样条。切比雪夫多项式在函数逼近方面表现良好, 并且训练过程中的损失和准确率曲线更加稳定。使用 ChebyKAN 模型对 MINIT 数据集进行分类, 证明了这一改进的有效性。

总体而言, 选择合适的模型结构和优化方法对于提升模型性能至关重要。优化后的 ResMLP 和使用切比雪夫多项式的 ChebyKAN 在处理复杂任务时表现出更好的性能, 展示了在实际应用中通过调整模型和优化策略可以显著提高神经网络的效果。

总而言之, 收获颇丰。

参考文献

- [1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [2] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2024.
- [3] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Izacard Gautier, Joulin Armand, Synnaeve Gabriel, Jakob Verbeek, et al. Feedforward networks for image classification with data-efficient training. *Feedforward networks for image classification with data-efficient training* in, 2021.