前馈神经网络

姓名：崔江浩
学号：2011915
专业：计算机科学与技术

1 原始MLP

原始MLP的网络结构是一个简单的全连接网络（Fully Connected Network），具体描述如下：

1. 网络结构：

- 这个网络包括三个全连接层（ `fc1` ， `fc2` ， `fc3` ）以及对应的两个 Dropout 层（ `fc1_drop` ， `fc2_drop` ）。
- 全连接层在PyTorch中被称为 `nn.Linear` ，每个 `nn.Linear` 层都有输入和输出的大小。第一层 `fc1` 的输入大小为28*28，输出大小为100，代表这个网络接受的输入是一个大小为28*28的向量，输出是100个神经元。类似的，第二层 `fc2` 的输入大小为100，输出大小为80，第三层 `fc3` 的输入大小为80，输出大小为10。
- Dropout 层在训练期间以一定概率随机关闭部分神经元，这是为了防止过拟合。在这个网络中，这个概率设定为0.2。

2. 激活函数：

- 每个全连接层后面都使用了ReLU（Rectified Linear Unit）激活函数，这是一种常用的非线性激活函数。这通过 `F.relu` 实现。

3. 输出层：

- 最后的输出通过log softmax进行激活，因为这是一个多类别分类问题。log softmax不仅能将输出转换为概率，同时也保证了所有输出的和为1，方便进行概率解释。此外，使用log softmax激活的网络输出可以直接作为 `nn.CrossEntropyLoss` 的输入进行训练。

4. 优化器和损失函数：

- 使用随机梯度下降（SGD）优化器，学习率设定为0.01，动量设定为0.5。
- 使用交叉熵损失（Cross Entropy Loss）作为损失函数。这是多分类问题的常用损失函数。

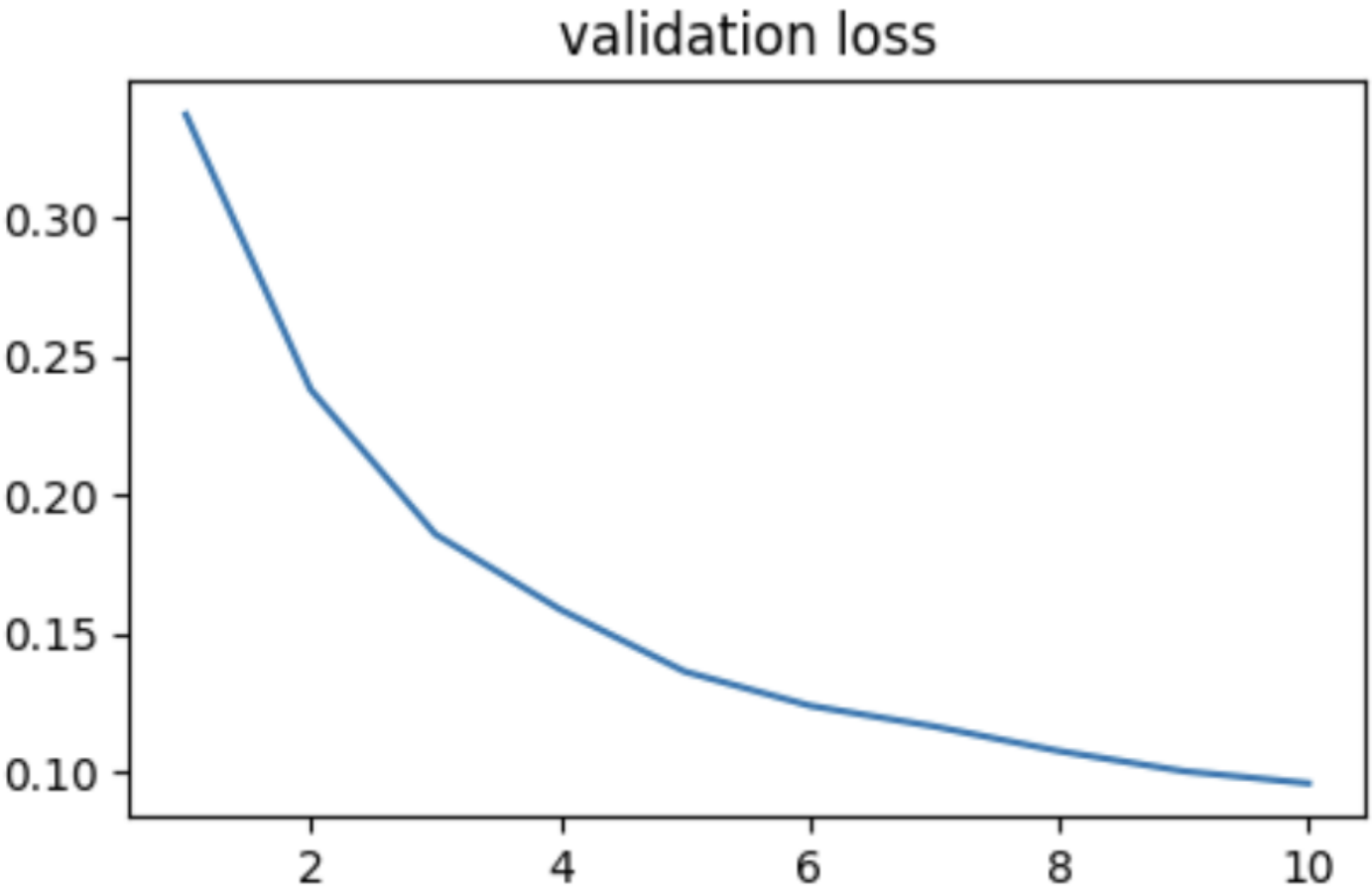
5. 训练和验证：

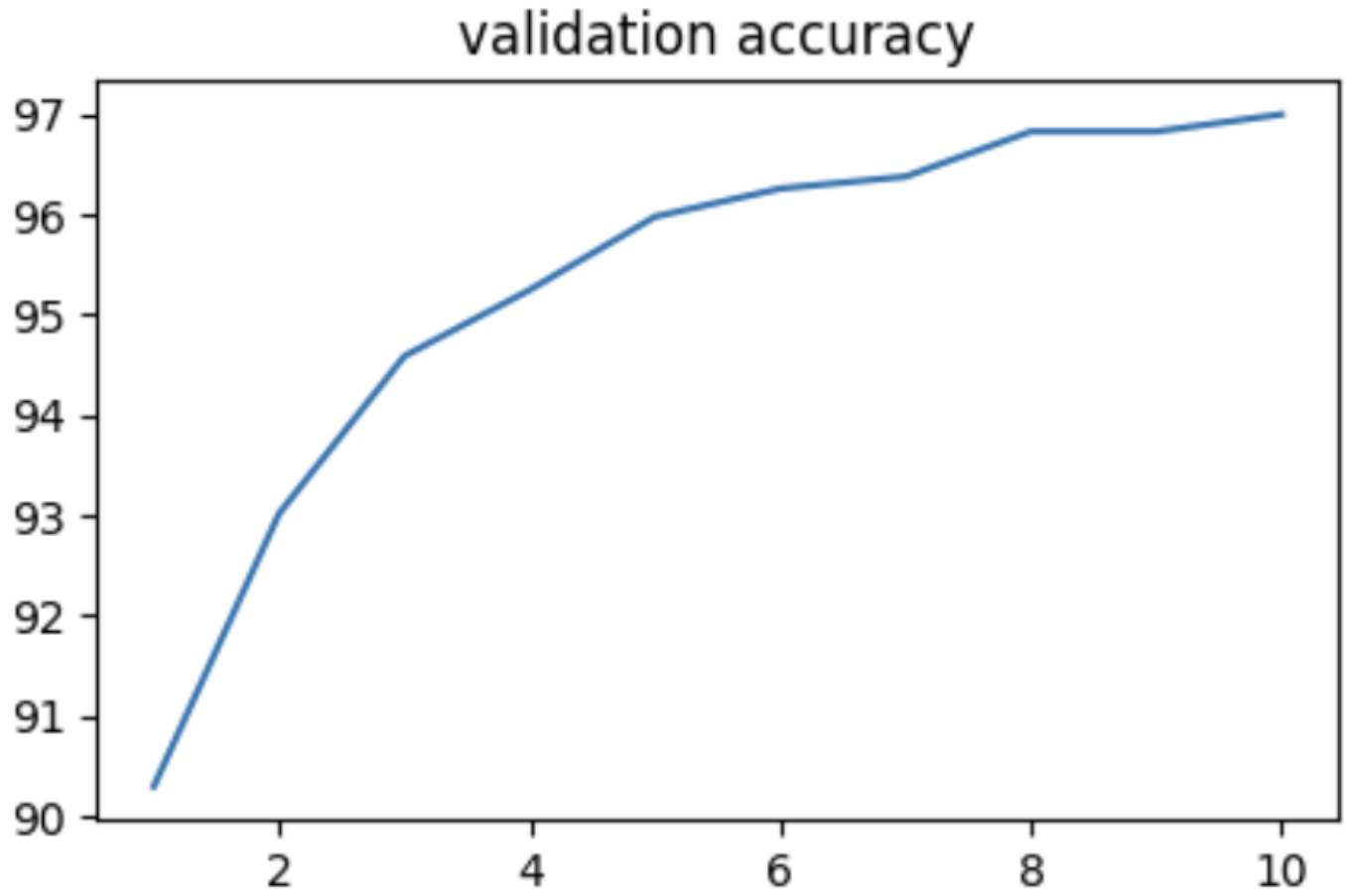
- 在训练阶段，每个batch的数据通过网络，计算输出和目标之间的损失，通过反向传播更新权重。
- 在验证阶段，网络切换到评估模式（关闭 Dropout 和 BatchNorm 等影响结果的层），并在所有验证数据上计算损失和准确率。

具体结构如下：

```
Net(
(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)
)
```

接下来是原始MLP的损失和准确率曲线：

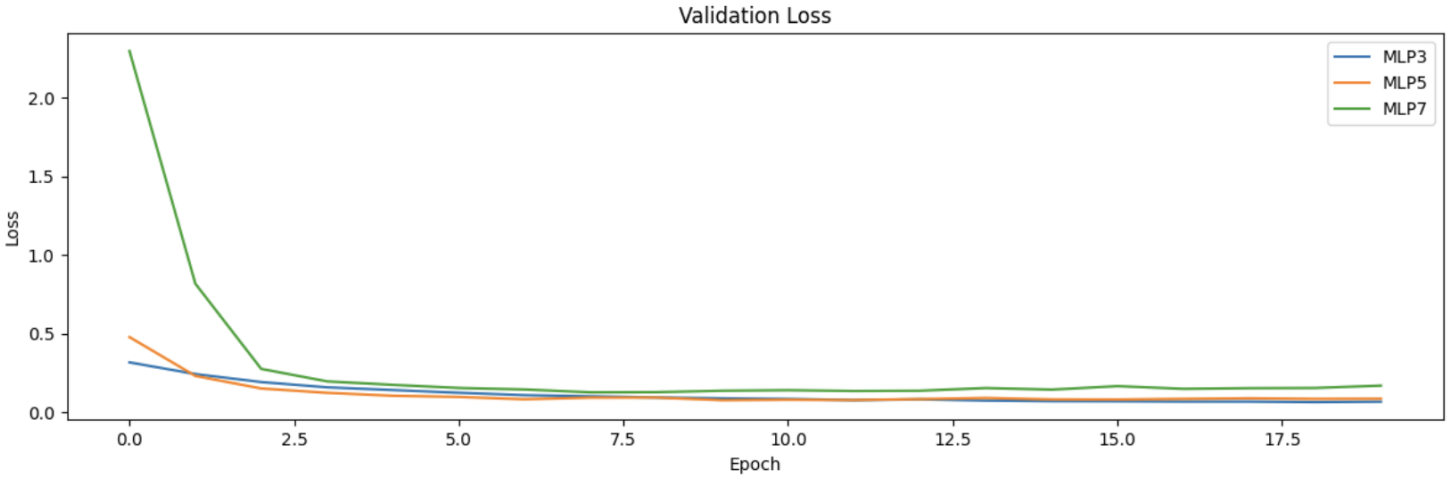


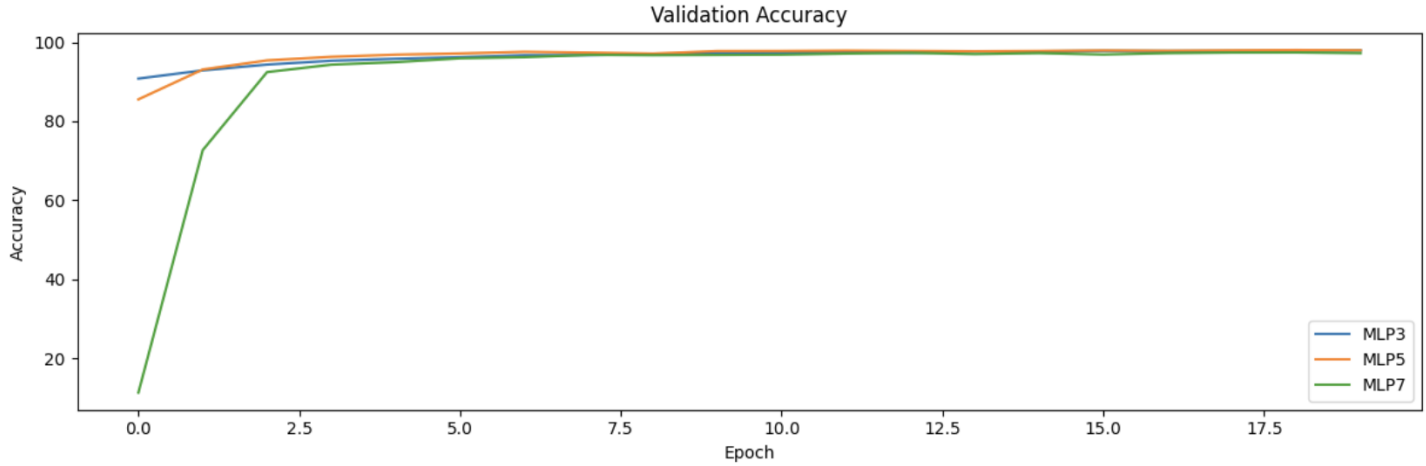


为了进一步探究全连接层参数（深度、宽度等）、优化器参数等对于最终实验结果的影响，在实验中分别尝试了MLP层数为3，5，7的情况，和隐藏层大小为[80,40,10]，[100,80,10]，[200,100,10]的配置，学习率设置为0.1，0.01和0.001。得到了结果如图所示。

1.1 MLP层数

首先对比了不同的MLP层数对最终实验结果的影响，对所有的网络都训练了20个epoch，得到实验结果如下：

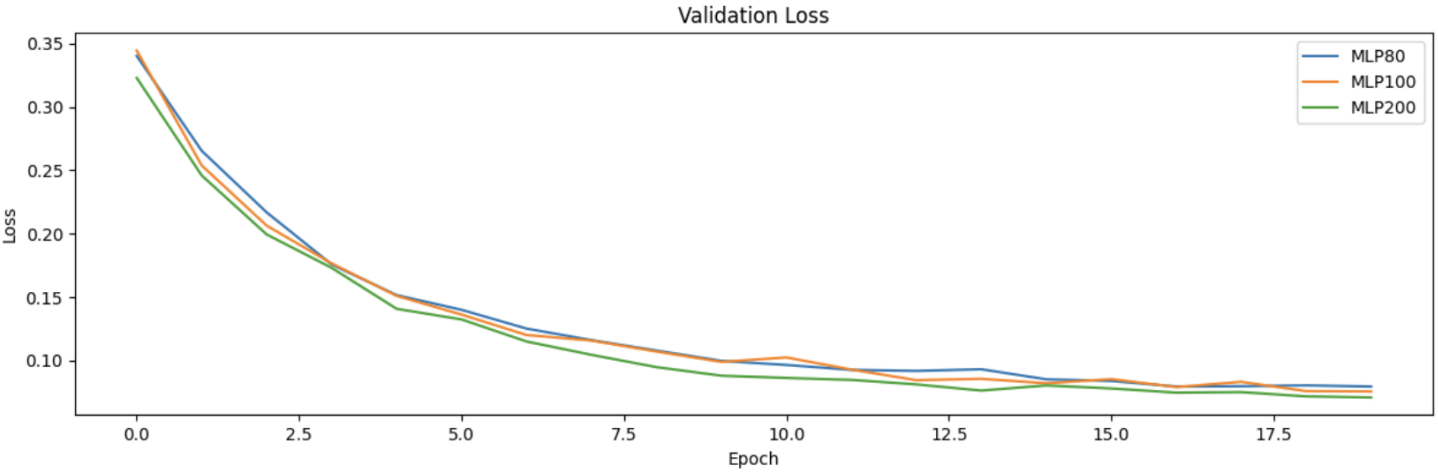


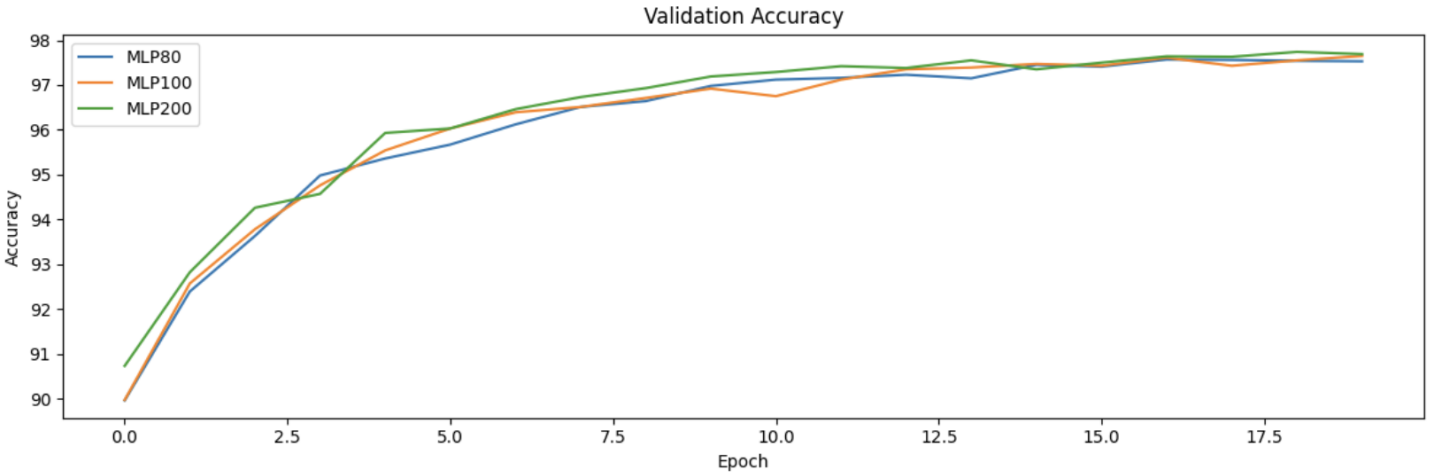


可以观察到MLP层数影响收敛速度，层数越深，网络结构越复杂，因此收敛速度越慢，同时，在MNIST数据集中的准确率表现相同，这可能是因为MNIST数据集较小，不需要过深的网络结构就可以得到较好的结果。

1.2 隐藏层宽度

首先对比了不同的隐藏层大小，对所有的网络都训练了20个epoch，得到实验结果如下：

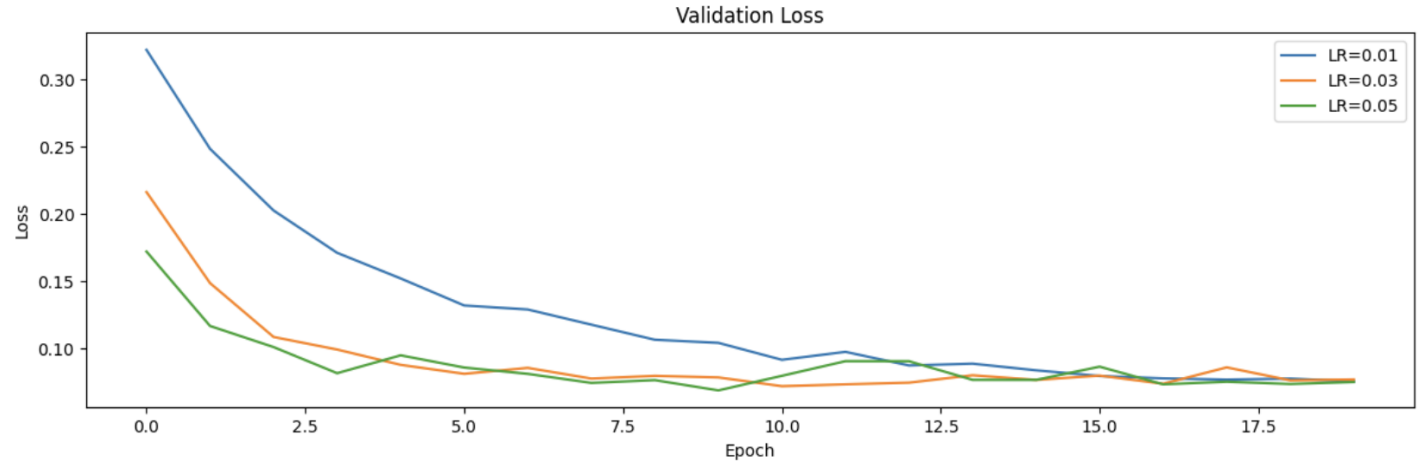


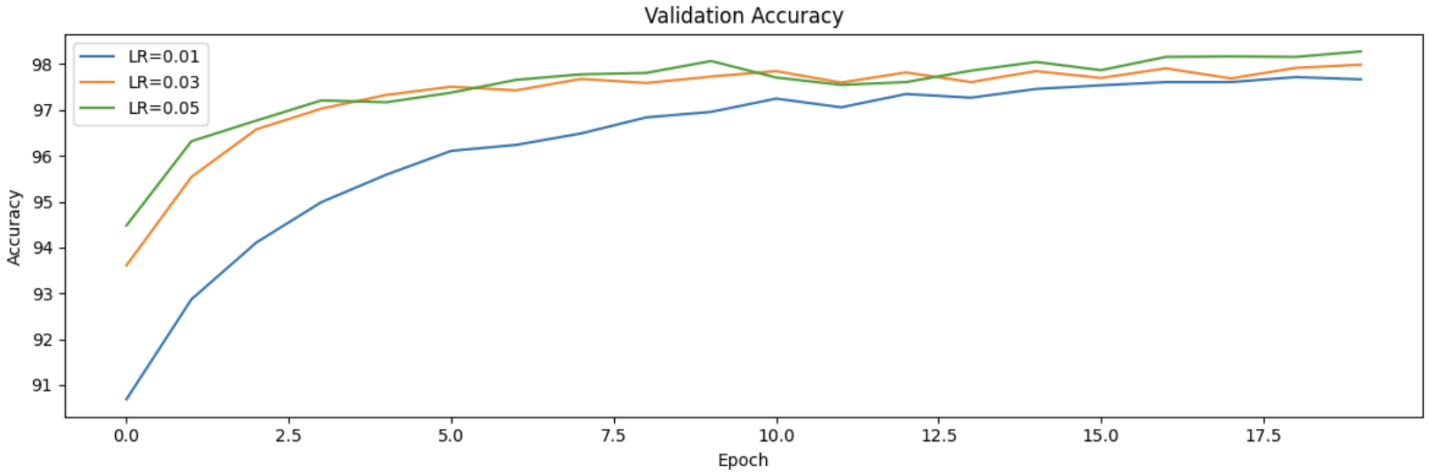


可以观察到，更宽的隐藏层可以获得更高的准确率，这是因为更宽的隐藏层可以提取更多的特征，是的不同类别之间的区别更容易辨别。

1.3 优化器参数

首先对比了不同的优化器参数，对所有的网络都训练了20个epoch，得到实验结果如下：





可以观察到更大的学习率不仅使模型获得更快的收敛速度，同时获得更高的准确率，这是由于本次分类的数据集较小，因此可以适当提升学习率以获得更快的收敛速度。

MLP_Mixer

MLP-Mixer由两种类型的层组成，它们分别在不同的维度上应用MLP：

1. Token-Mixing MLP层：在这一层，MLP应用于“空间”维度，即图像的每个像素位置（对于NLP任务，这将是每个token位置）。所有位置都使用相同的MLP（即权重共享）。这一层的目标是让模型有能力学习和使用不同位置之间的交互。

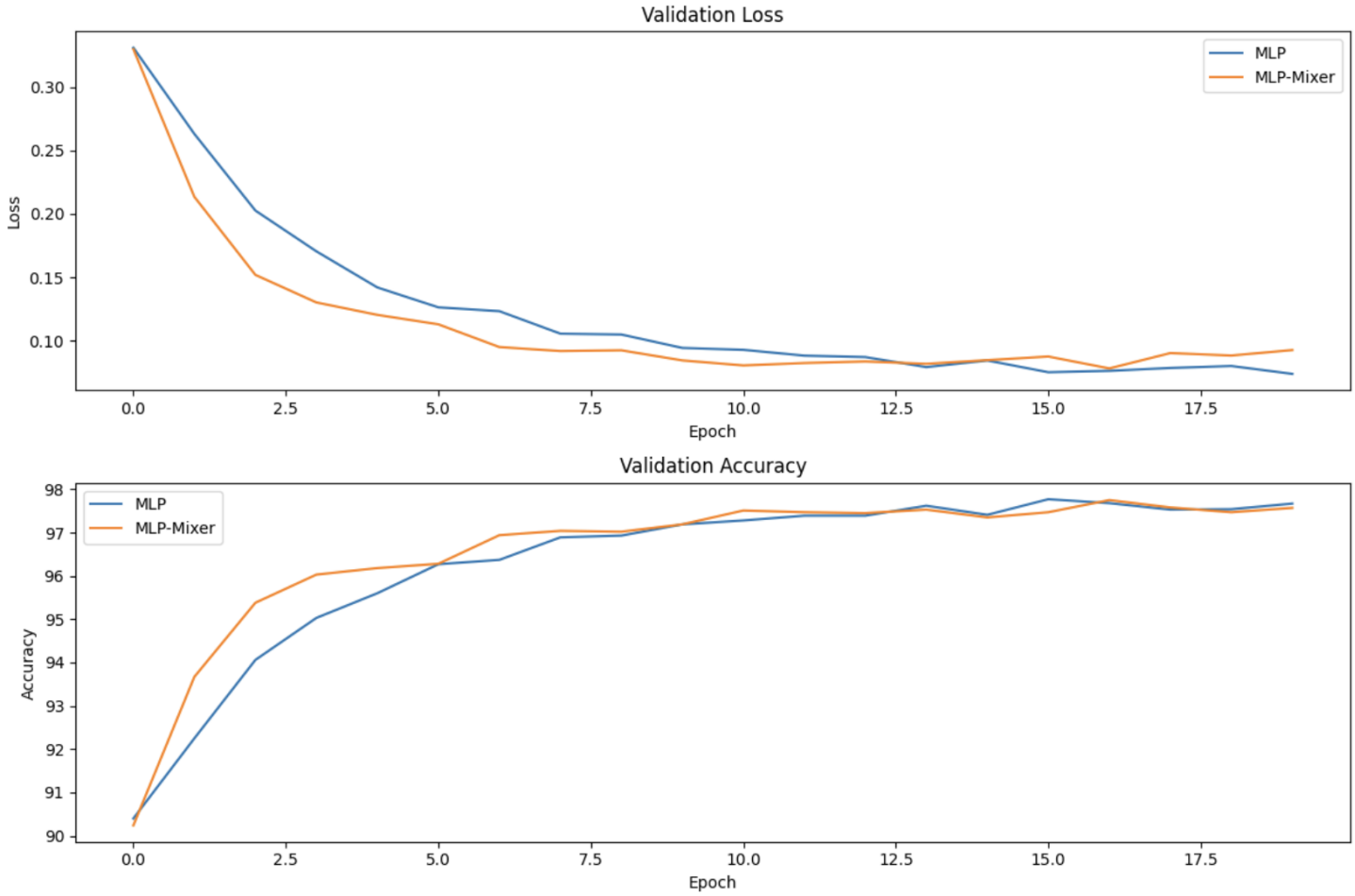
2. Channel-Mixing MLP层：在这一层，MLP应用于“通道”维度，即每个位置的特征向量。这一层的目标是让模型有能力学习和使用不同通道之间的交互。

原始的MLP-Mixer模型使用了块的结构，其中每个块包含一个Token-Mixing MLP层和一个Channel-Mixing MLP层。输入首先经过一个线性层进行降维，然后输入到一系列这样的块中，最后经过一个线性层进行分类。

然而，为了适应MNIST数据集，上述实现的MLP-Mixer进行了一些简化。我去掉了原始模型中的块结构，只保留了一个Token-Mixing MLP层和一个Channel-Mixing MLP层。另外，也没有对输入进行降维，因为MNIST的图像已经是很低的维度（28*28）。这使得实现更简单，但也限制了模型的表示能力。

值得注意的是，尽管MLP-Mixer的结构相对简单，但它在图像分类任务上的表现却与一些更复杂的模型（如ViT和CvT）相当。这表明，即使没有卷积和自注意力，仅使用MLP也能达到很好的效果。

其与原始MLP的对比如下所示：



可以观察到MLP-Mixer在收敛速度和准确率上都要略好于原始MLP，可能是因为MLP-Mixer通过在两个维度（tokens和channels）上进行混合，可以学习到输入数据中不同维度的交互和关联。这可能帮助模型更好地理解数据，从而提高了准确率和收敛速度。