## 卷积神经网络实验报告

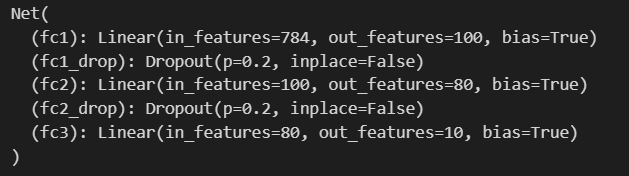
姓名：管昀玫 学号：2013750

##### 实验要求

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

##### 原始MLP

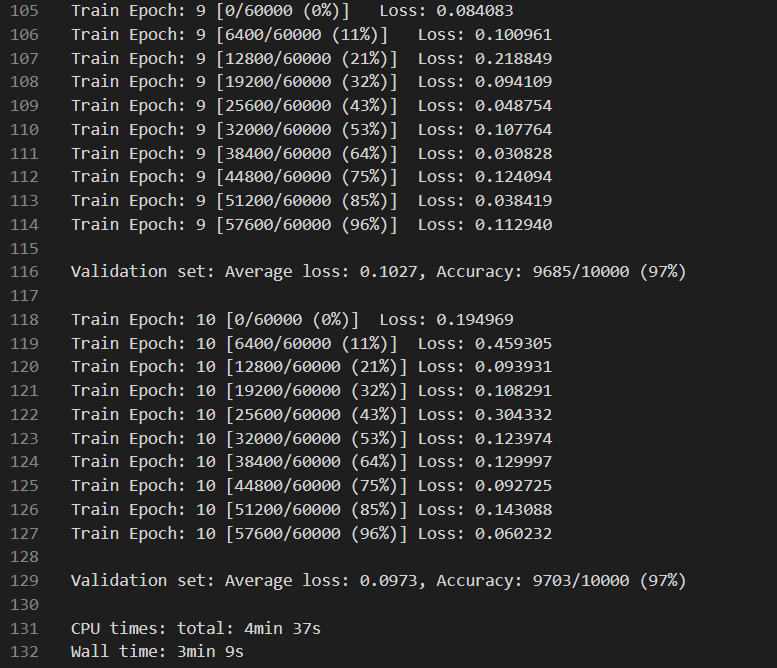
在给定的MLP中，网络结构如下图所示：



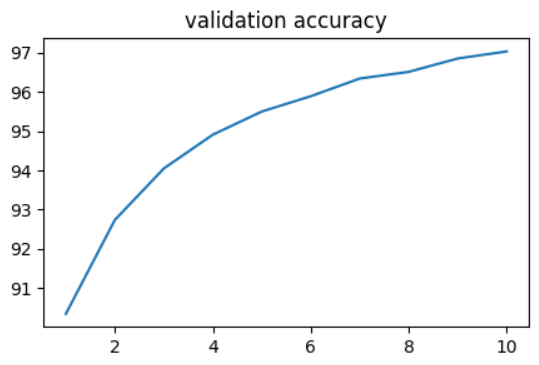
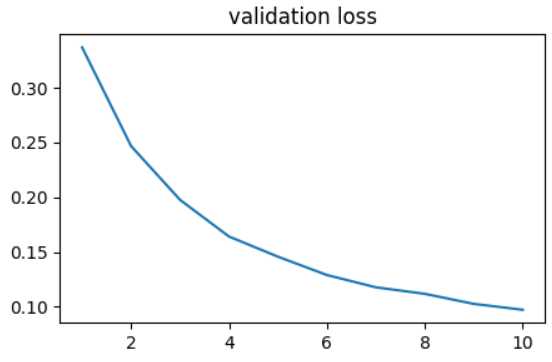
我们可以看到，网络由3个全连接层和2个丢弃层组成。全连接层，是每一个结点都与上一层的所有结点相连，用来把前边提取到的特征综合起来。由于其全相连的特性，一般全连接层的参数也是最多的。而dropout是为了防止过拟合，按照一定概率将神经网络训练单元从网络中移除。

1. 第一个全连接层（fc1）将输入映射到一个具有100个神经元的隐藏层。这一层的输出结果通过一个丢弃层（fc1\_drop）进行正则化。丢弃层可以随机丢弃一部分神经元的输出，有助于减少过拟合现象。
2. 第二个全连接层（fc2）将第一个隐藏层的输出映射到一个具有80个神经元的隐藏层。同样地，这一层的输出结果也会通过一个丢弃层（fc2\_drop）进行正则化。
3. 最后一个全连接层（fc3）将第二个隐藏层的输出映射到一个具有10个神经元的输出层。这个输出层的每个神经元对应于数字0到9的分类。输出层的激活函数是softmax函数，用于产生分类预测的概率分布。
4. 整个网络的训练过程涉及通过反向传播算法来调整网络的权重和偏置，以最小化训练样本上的损失函数。

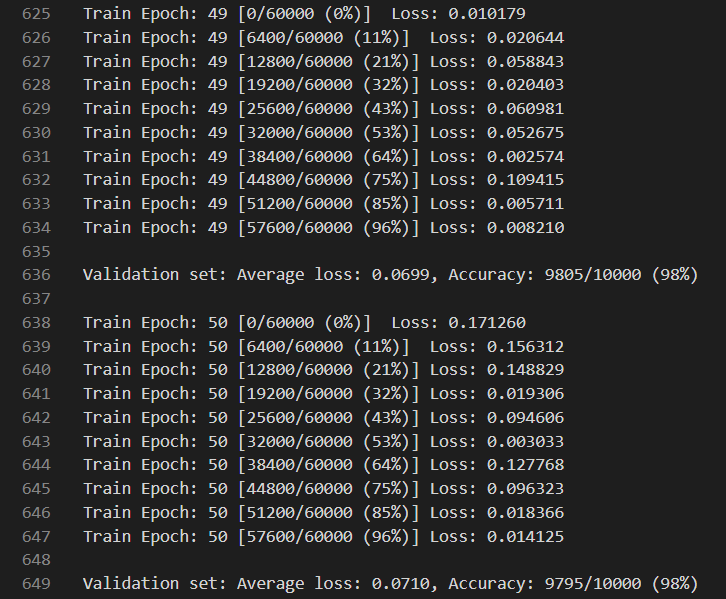
设置epoch=10，得到该网络的训练结果如下所示：



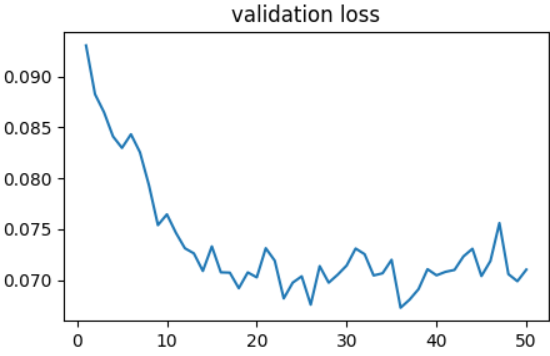
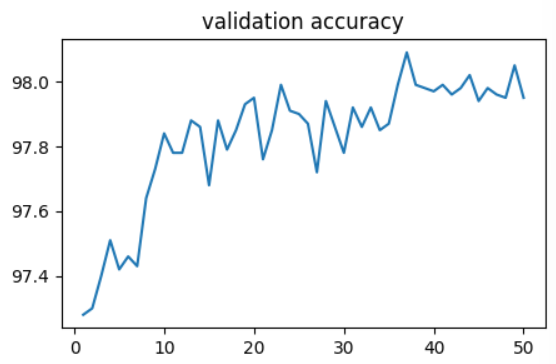
可以看到，在训练10轮之后，该网络的准确率达到了97%，Average Loss为0.0973. 但是从Loss的数据来看，该网络还没有达到收敛。



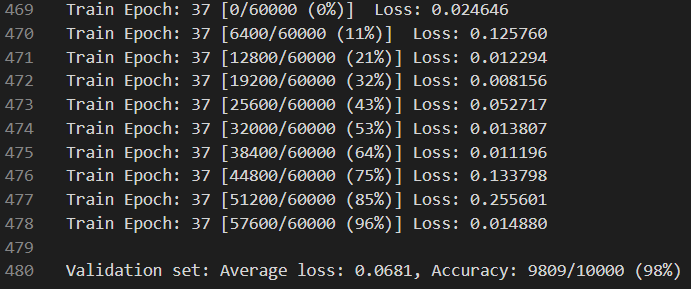
因此我尝试将epoch改为50，重新训练该网络：



可以看到，第49轮和第50轮的accuracy已经达到了98%，但是第50轮的loss不降反升。查看前几轮训练的loss，可以看出模型已经达到收敛了。

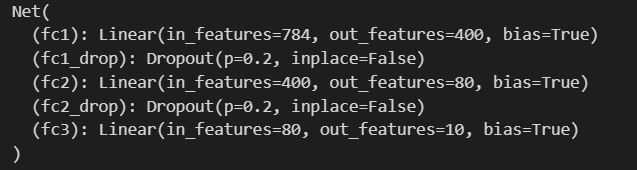
大约在第37轮左右时，模型已达到最后，之后有过拟合的迹象。查看log，在第37轮是模型最高正确分类准确率为9809/10000，如下图所示。



##### 优化MLP

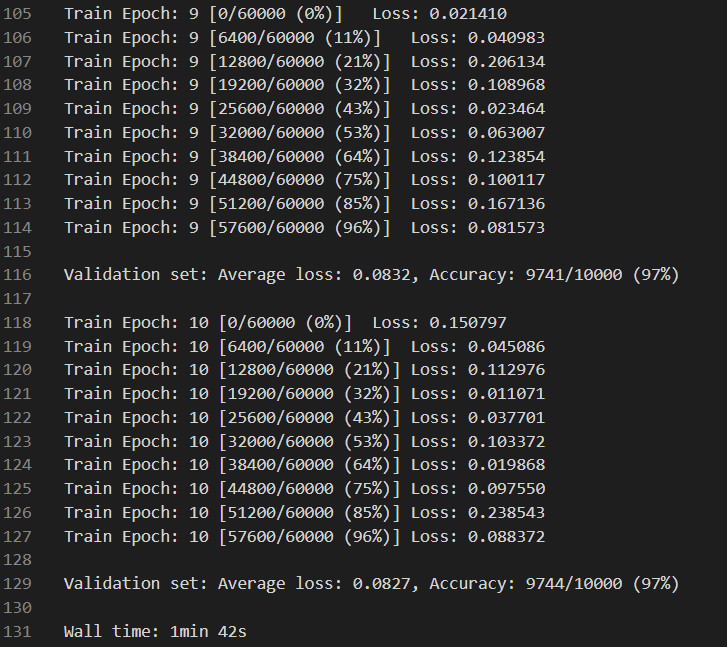
###### 优化1：调节MLP的全连接层参数

将MLP网络的参数调整至下图所示：



在这里，我将第一层的out\_features调整为400，这样做的原因是更大的维度可能可以更好地保持原始图像的特征，以带来更好的效果。其余的部分暂时保持不变。

训练该网络，将epoch设定为10，得到以下结果：



可以看到，虽然Accuracy还是97%，但是实际上正确分类的数量是9744（原始为9703），且loss变成了0.0827（原始为0.0973），说明实际上网络存在提升，这为我们进一步修改网络提供了方向。

###### 优化2：调整optimizer

原始的optimizer为：

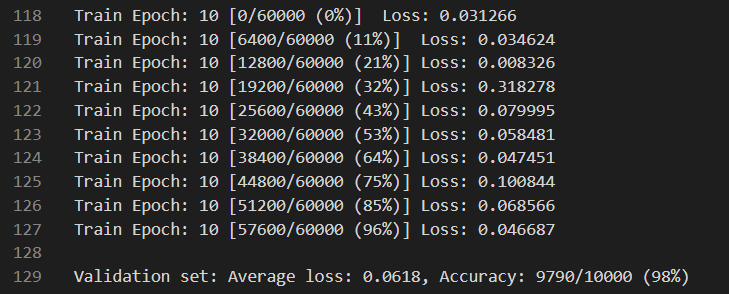
optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.5)

我尝试修改为：

optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.75, weight\_decay=3e-4)

* lr=0.01：学习率控制了每次参数更新的步长。较高的学习率会使模型更快地收敛，但可能会导致震荡或不稳定的训练过程，而较低的学习率可能会导致训练过程较慢。0.01是一个常见的初始学习率，我们暂时不修改它。
* momentum=0.75：动量参数控制了历史梯度对当前梯度更新的影响程度，取值范围通常在0到1之间。原始为0.5，我们将其设置为0.75，以求加快收敛速度并减小参数在局部极值点附近的波动。
* weight\_decay=3e-4：权重衰减是一种正则化技术，用于防止模型过拟合。它通过对模型的权重施加惩罚，促使模型学习到更简单的权重分布。在这里我们将其设置为3e-4。

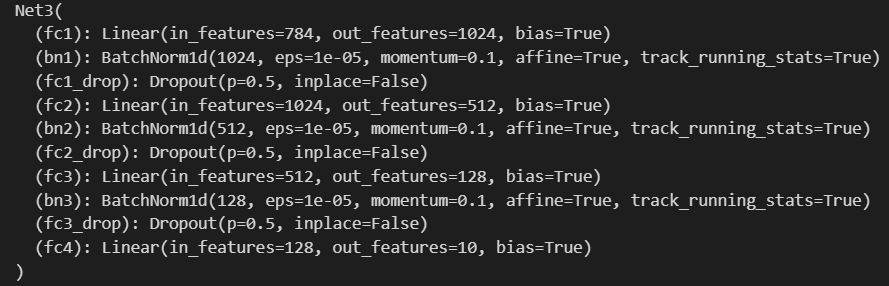
设置新的optimizer的同时，我们保持上一次优化的网络（即第一层的out\_features=400）。训练该模型，设置epoch为10，结果如下：



我们可以看到，仅仅10轮模型就迅速达到了98%的准确率，average loss为0.0618，提升明显，说明optimizer设置是有效的。

###### 优化3：更改网络结构，加入归一化层

重新编写网络结构，如下所示：

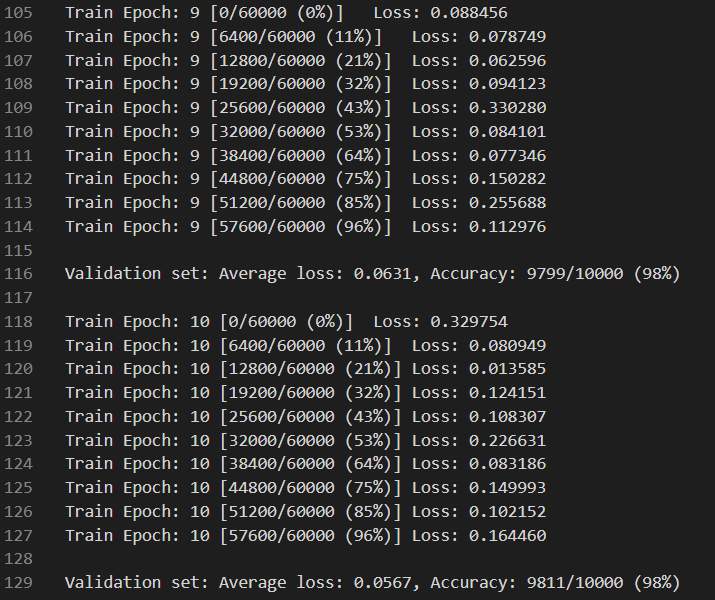


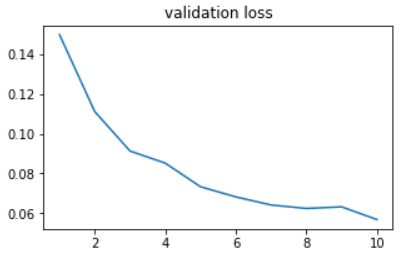
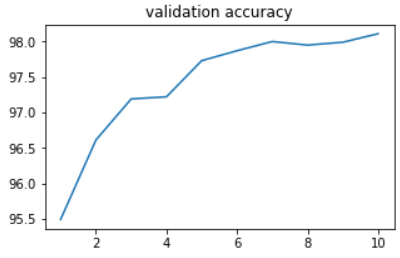
模型维度的变化为：784->1024->512->128->10。增加了全连接层的层数，更好地提取特征，细化了分类的过程，对于提升准确率有一定的帮助，但也有可能造成过拟合。

在每个全连接层后都增加了一个归一化层。归一化层的作用如下所示：

1. 加速训练：批归一化层可以减少网络的内部协变量偏移，有助于加速训练过程。它通过对每个小批量样本的特征进行归一化，使得网络对输入数据的小变化更加鲁棒。
2. 提高模型的稳定性：批归一化层对网络的激活值进行归一化，使得网络对初始权重的选择不敏感，减少了梯度消失和梯度爆炸的风险，提高了模型的稳定性。
3. 增强模型的泛化能力：批归一化层具有一定的正则化效果，可以减少过拟合的风险，提高模型在未见过的数据上的泛化能力。
4. 允许使用更高的学习率：批归一化层可以使得网络更容易适应较高的学习率，加快模型的收敛速度。

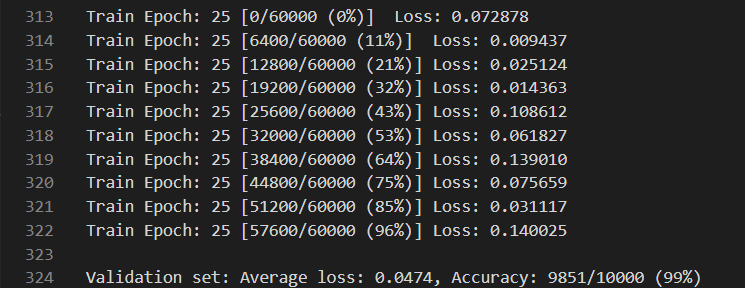
我们暂时不使用修改后的optimizer，以查看这个网络的原始性能。训练10轮，如下所示：

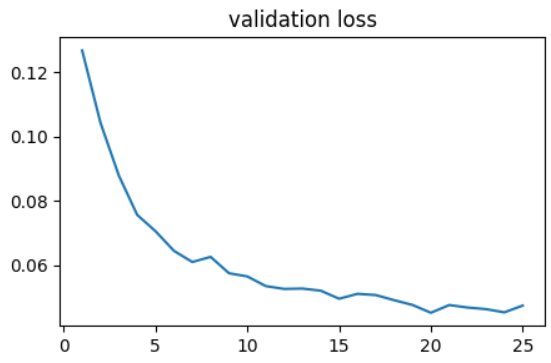
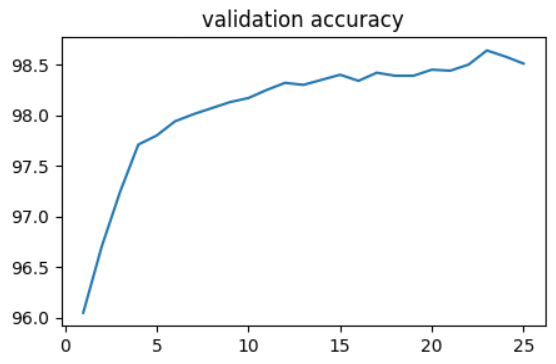


可以看到，模型轻轻松松达到98%的正确率，实际正确数量比baseline训练50轮还要高；average loss为0.0567，且还有收敛迹象。

现在我们添加进修改后的optimizer，训练25轮后，模型的数据如下所示：



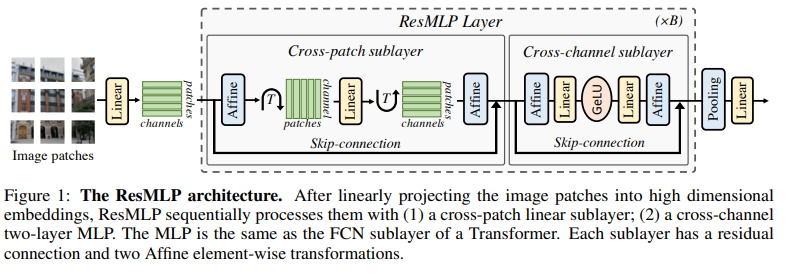
 

查看loss和accuracy可知，模型已大致达到收敛，而在第25轮是模型的准确率已经达到了99%，令人满意。

##### ResMLP

作为该实验的拓展部分，我选择实现ResMLP。ResMLP是一种简单的残差网络，它可以(i)替代一个线性层，其中图像patch在各个通道之间独立而相同地相互作用，以及(ii)替代一个两层前馈网络，其中每个通道在每个patch之间独立地相互作用。

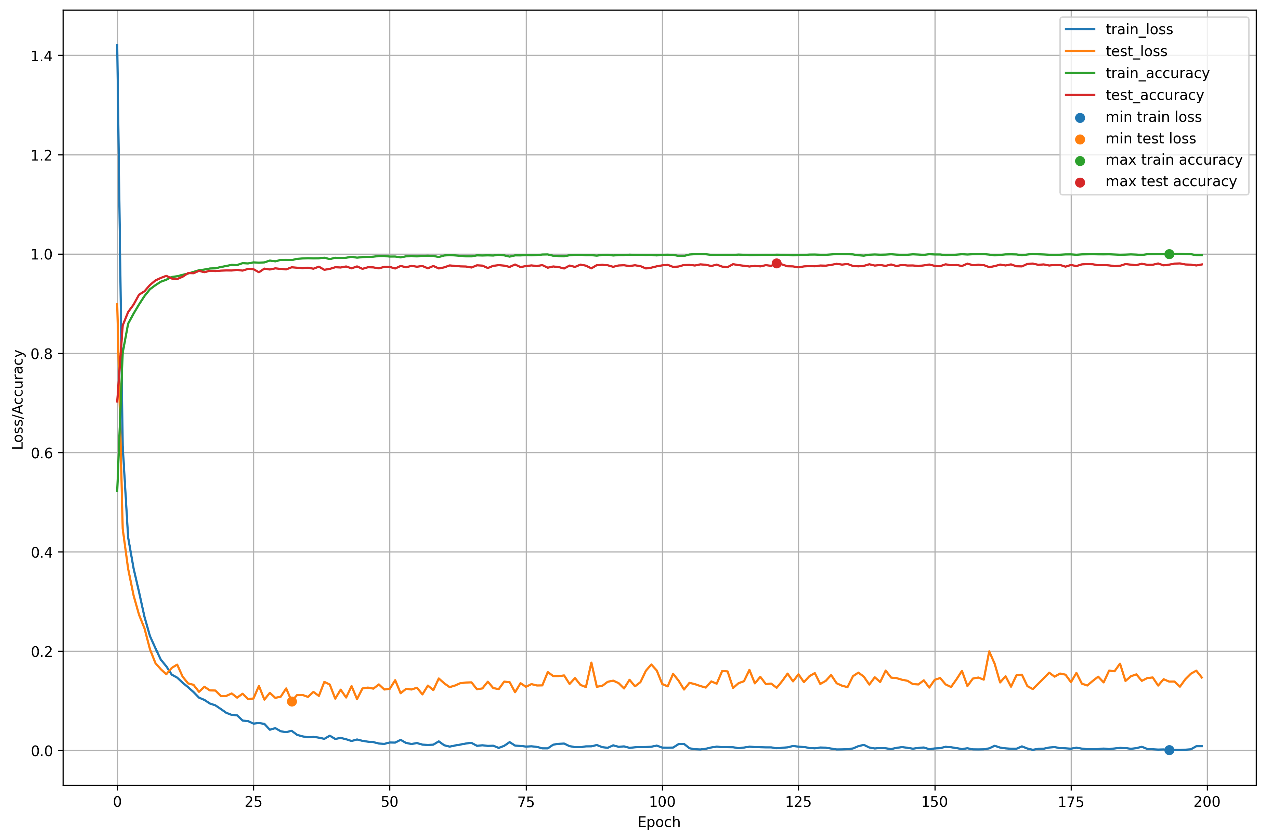
它的架构如下图所示：



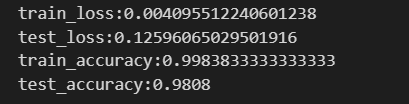
ResMLP体系结构受到ViT的强烈启发，但在几个方面要简单的多：用线性层替换了自注意力层，从而形成了只有线性层和GELU非线性的体系结构。当使用与DeiT和CaiT中相同的训练法方案时，ResMLP的训练比ViTs更稳定，可以消除特定批次或跨通道规范化的需要，如BN、GN、LN等，推测这种稳定性来源于用线性层替代自注意力层，最后使用线性层的另一个优点是可以可视化patch embedding之间的作用，揭示了类似于底层与深层卷积的滤波器。

ResMLP的源代码很容易获得，论文作者已在github上公开：<https://github.com/rishikksh20/ResMLP-pytorch/blob/master/resmlp.py>这里我直接使用作者的源码，但使用的数据集是作业要求的MNIST。

ResMLP的模型表现如下：



在epoch为121时，模型的test accuracy达到最大，为98.08%，如下图所示：



从曲线图中科院很直观地看到，在epoch很小的时候，ResMLP的性能还不如原始的MLP。猜测可能是由以下原因引起的：

1. **模型复杂度不匹配**：ResMLP是一种更复杂的模型，具有更多的参数和层级结构。在数据集较小且相对简单的MNIST上，复杂的模型可能过度拟合训练数据，导致泛化性能下降。相比之下，原始的MLP可能在MNIST数据集上能够更好地适应数据并取得更好的结果。
2. **训练数据不足**：如果训练数据集的大小较小，复杂的模型如ResMLP可能需要更多的数据来充分学习其参数。如果训练数据集不足以支撑ResMLP的复杂度，模型可能无法充分发挥其优势，导致性能低于原始的MLP。
3. **超参数调整**：ResMLP中可能存在一些需要调整的超参数，如层数、宽度、学习率等。这些超参数的选择对于模型的性能至关重要。如果没有经过充分的超参数搜索和调整，ResMLP可能无法发挥其最佳性能，而原始的MLP可能在默认超参数下表现更好。

##### 实验心得

深度学习调参虽然魔幻，但是经过这次实验我感觉摸索到了一些门道，心得如下所示：

1. 优化器：

* SGD 效果好于 Adam
* 设置 momentum 和 weight\_decay，后者对于过拟合问题效果其实不是很明显
* 如果N个epoch内，正确率不变则降低学习率

1. 激活函数：使用ReLU的效果明显好于Sigmoid，ReLU比sigmoid更能拟合非线性输出和输出特征的映射。
2. 批归一化：加入之后效果有明显的提升
3. MLP网络的最佳顺序：Linear -> Batch Norm -> Activation -> Droput -> Linear
4. 第一个全连接层的输出可以大一些，更大的维度可能可以更好地保持原始特征
5. 总体来说，先过拟合然后再慢慢尝试各种正则化方法，比较省时省力。
6. Lr虽然这次没涉及，但是可以尝试early stopping