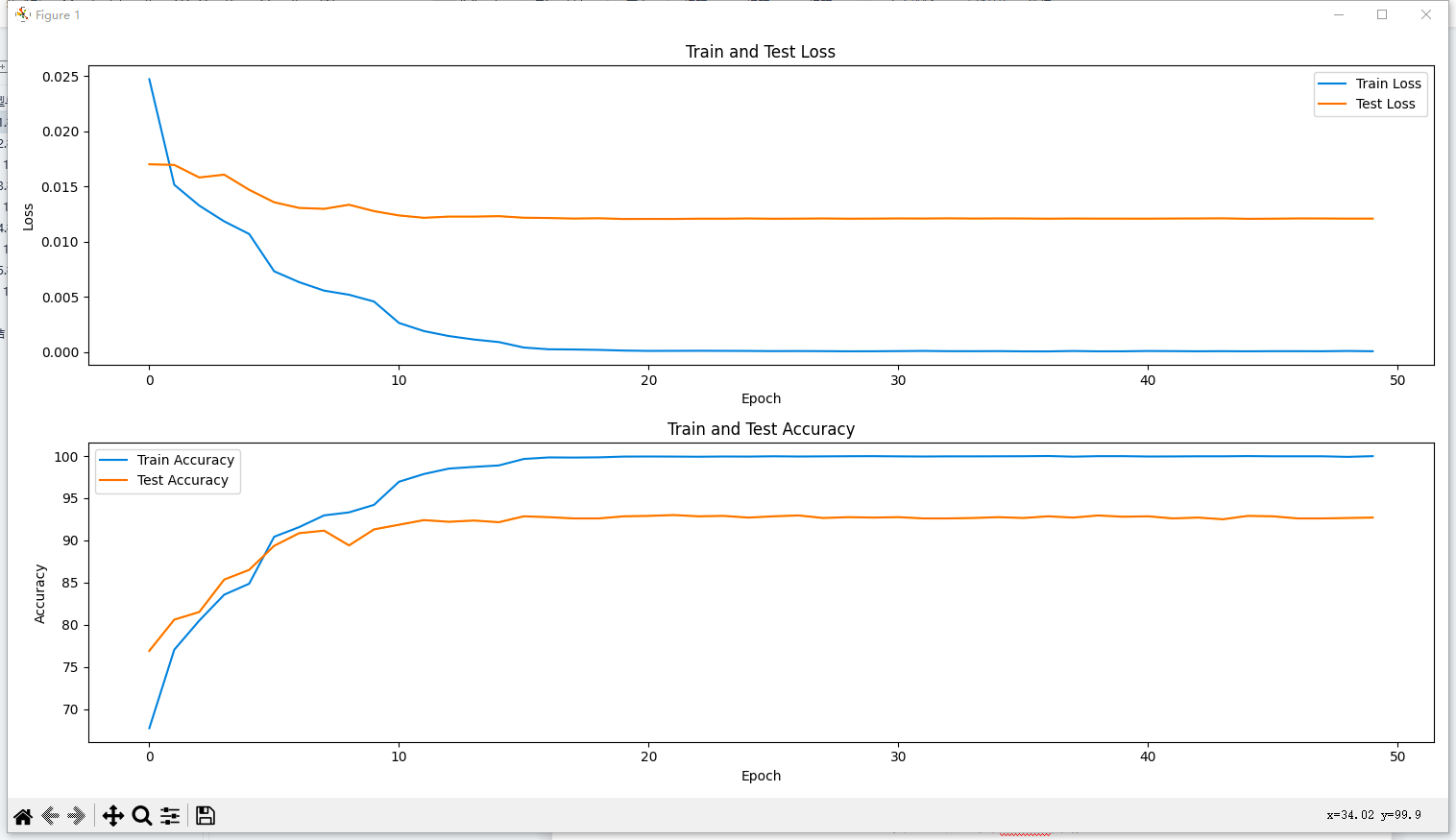
# 模型与训练

## 模型说明

本次训练中，超参数：学习率：初始0.05，采用动态学习率，每5步x0.2；BATCH\_SIZE:64，

通过继承利用torch库中的ResNet模型并加以改造，构建了一个ResNet基本块（BasicBlock）的ResNet。ResNet网络如图 1.1所示，该ResNet网络有4层，创建的神经网络如图 1.2所示，本次实验一共尝试了4中不同的网络构型，通过改变残差块的结构和残差块的数量实现。用sklearn中自带的mnist数据库进行训练与测试，如图 1.3所示

[2,2,2,2] 50轮最终测试集准确率92.70%



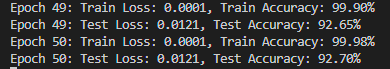
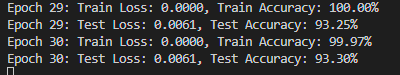
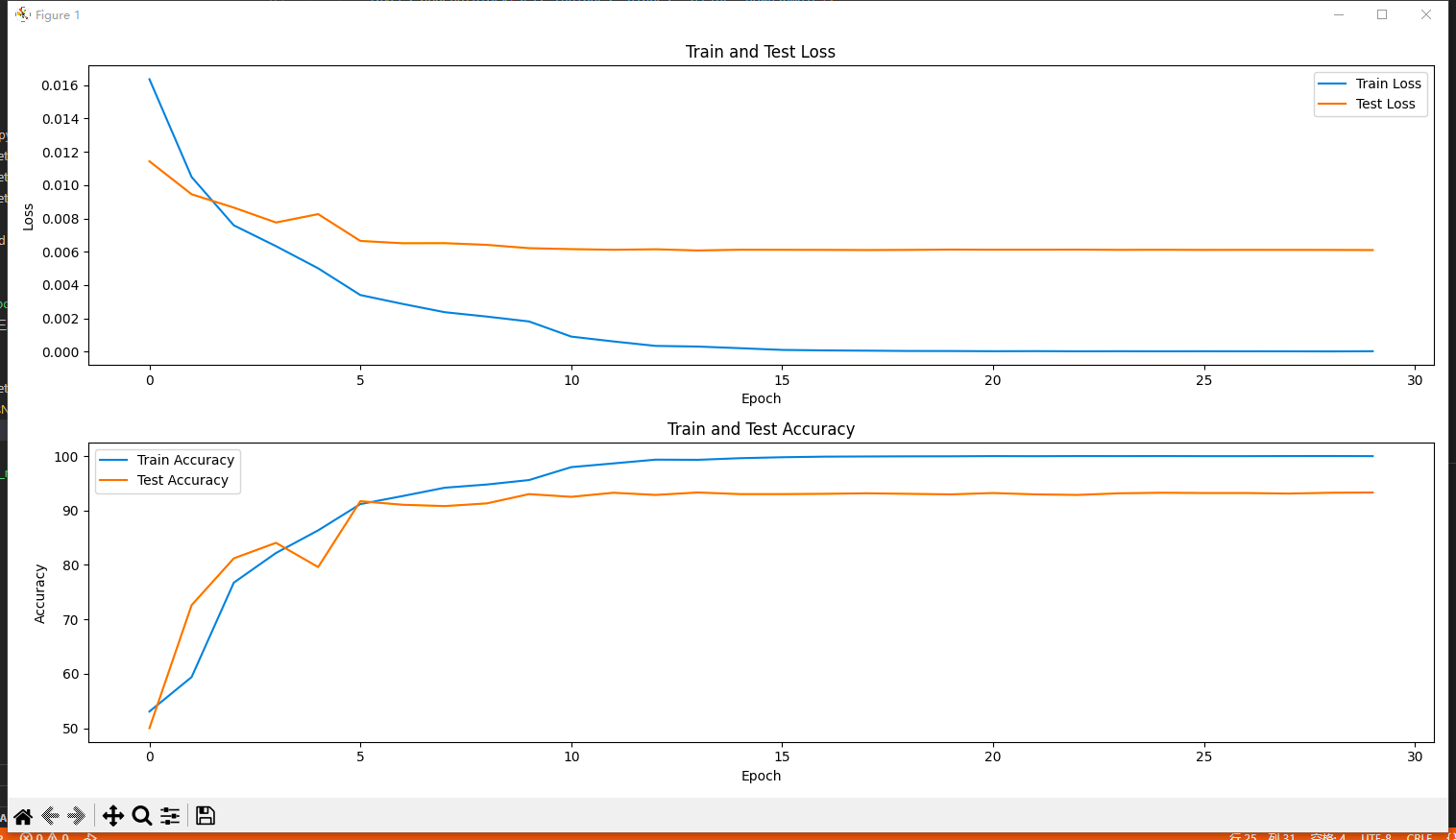


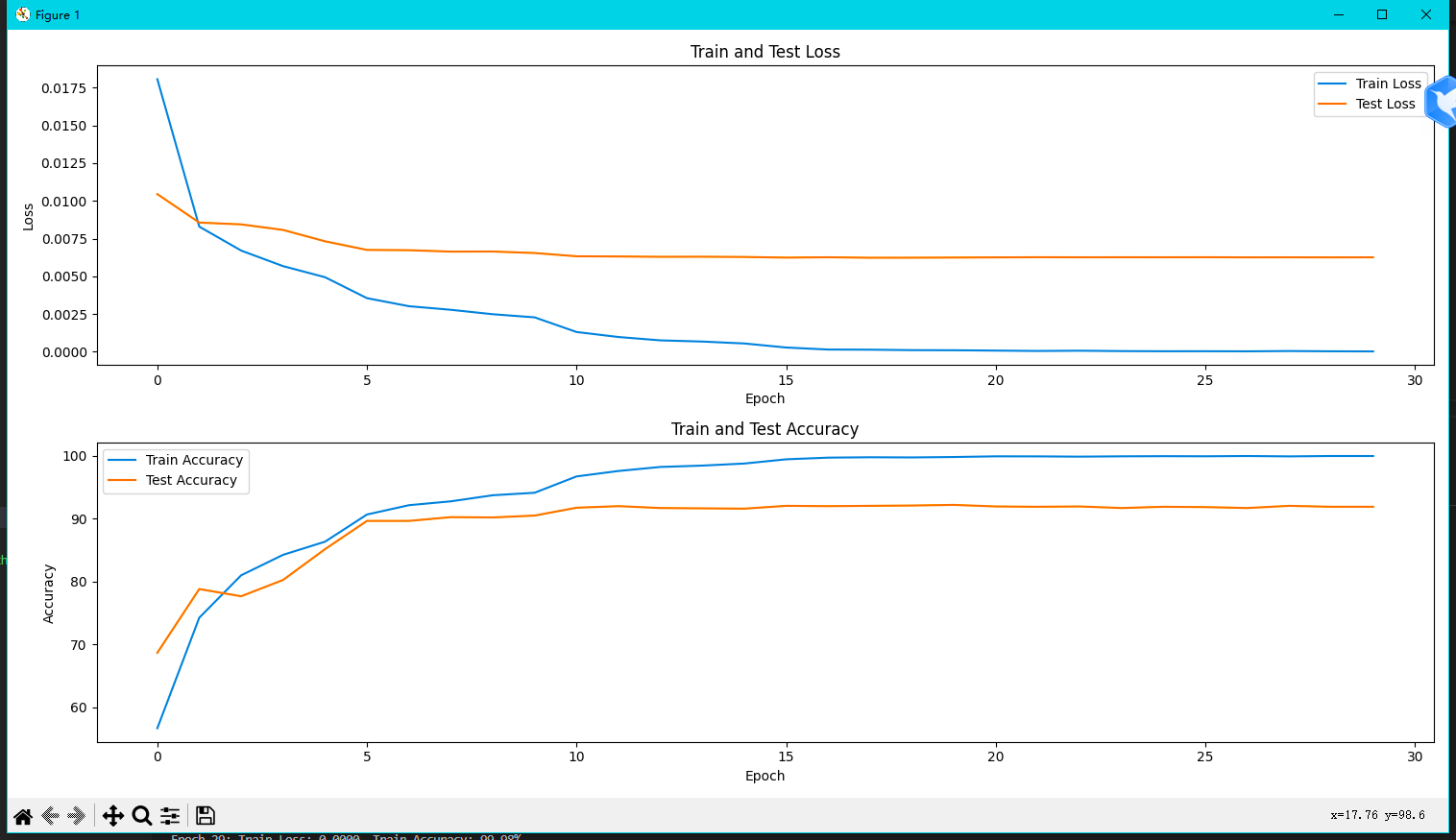
图 1.3

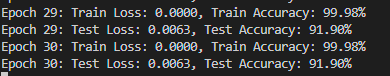
[2,4,4,2] 30轮



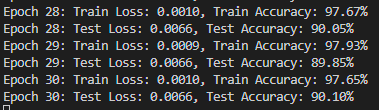


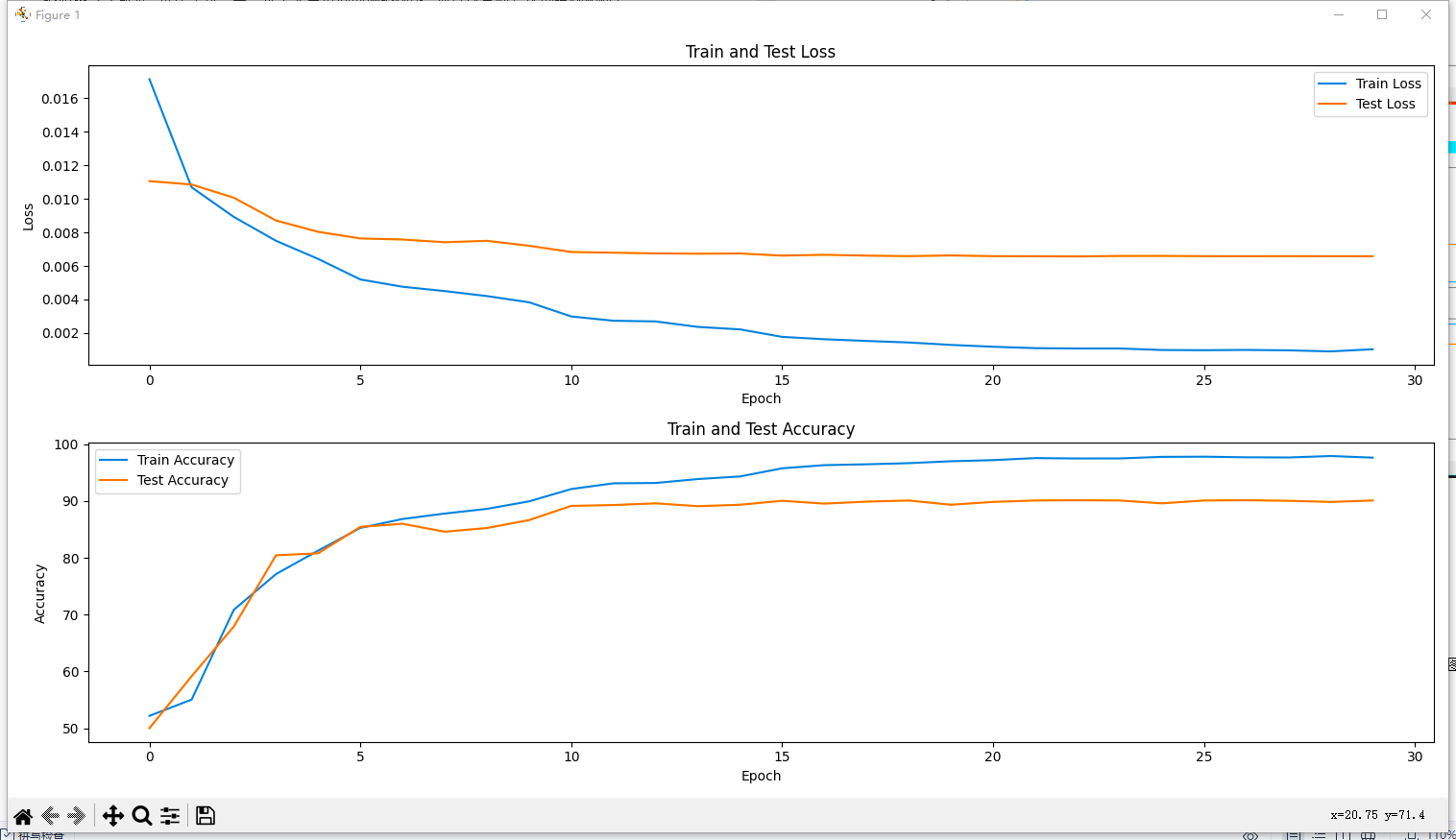
[4,4,4,4] 30 91.90%



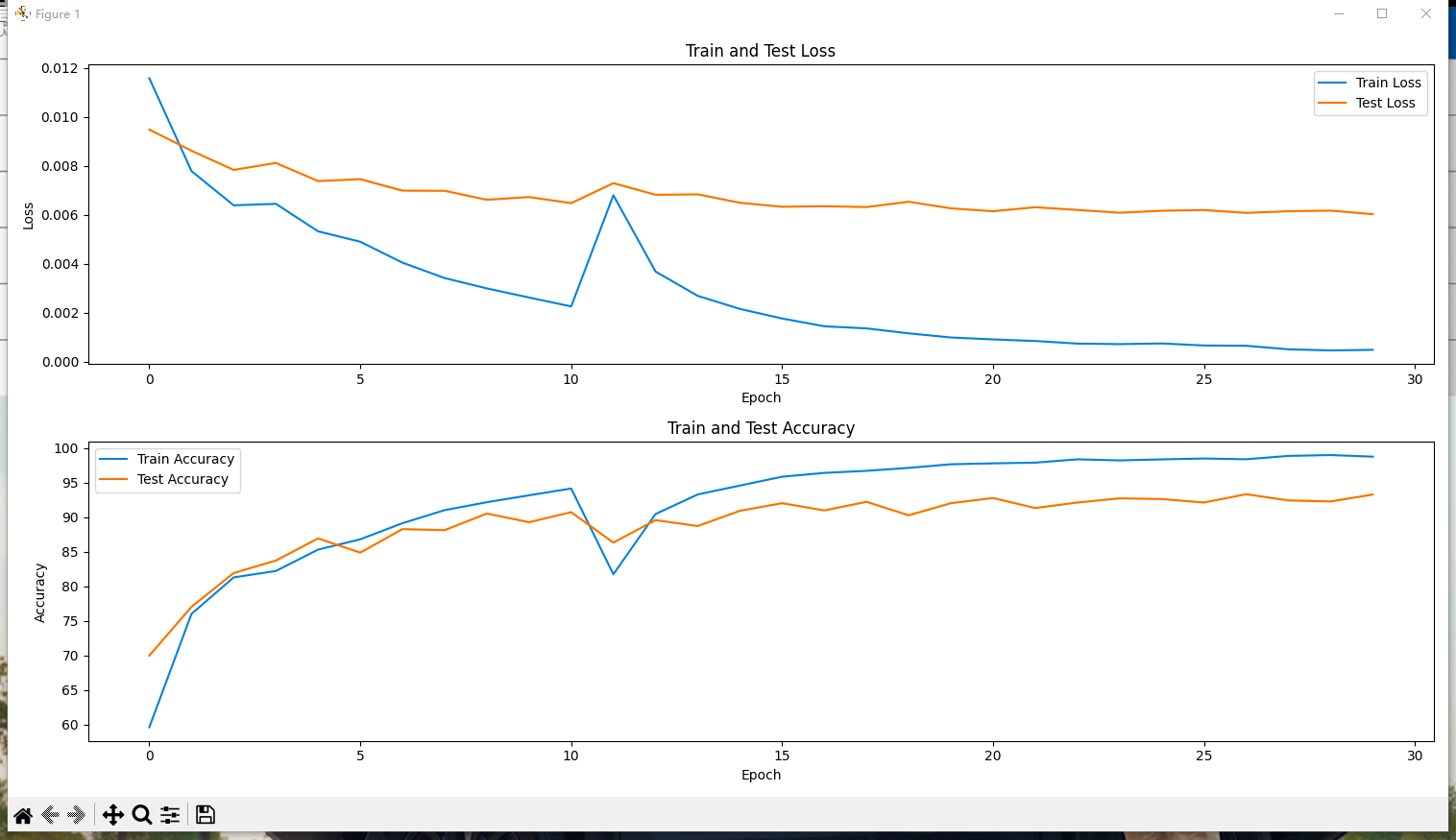


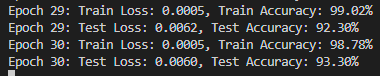
2，8，8，4，30轮，90.10%



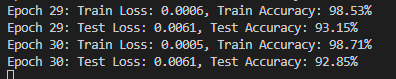


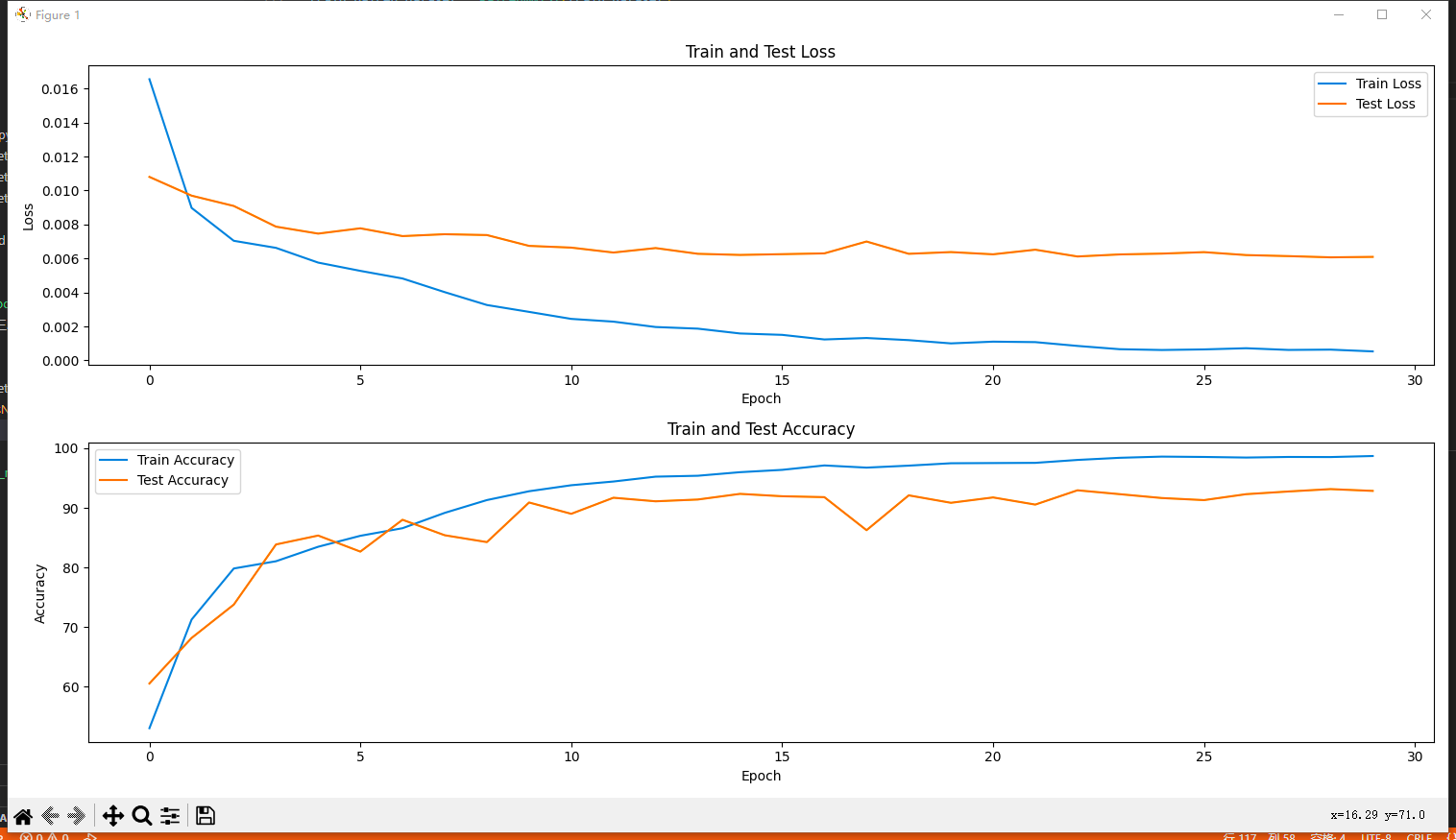
[2,4,4,2],学习率0.01





[2,4,4,2],学习率0.05





## **模型1**

模型1中使用了ResNet模块自带的标准残差块BasicBlock，每个标准残差块结构如图 1.4、图 1.5所示，残差快结构如下：

1. 3x3卷积层
2. 归一化层
3. ReLU函数激活层
4. 3x3卷积层
5. 归一化层
6. 下采样化层
7. 权重相加
8. ReLU激活函数层

该残差块先保存输入 x 到 identity，作为跳跃连接的输入，再将输入 x 通过卷积、归一化和ReLU激活函数的组合得到 out，将 out 再次通过卷积、归一化，如果存在下采样层 self.downsample，则对输入 x 进行下采样，然后将下采样后的输入 identity 与 out 相加，最后通过Relu激活函数得到最终输出。是一个标准的ResNet网络块，通过跳跃连接维持了梯度的信息流通，有助于更深层次的网络训练。

用此标准残差块构建了有10个分类，然后四层的残差块数量分别为[2,2,2,2]，一共有8个残差块的ResNet网络，如图 1.6所示，并在此网络上进行训练，训练轮数为20轮，观察每轮训练的训练损失与minibatch的关系。

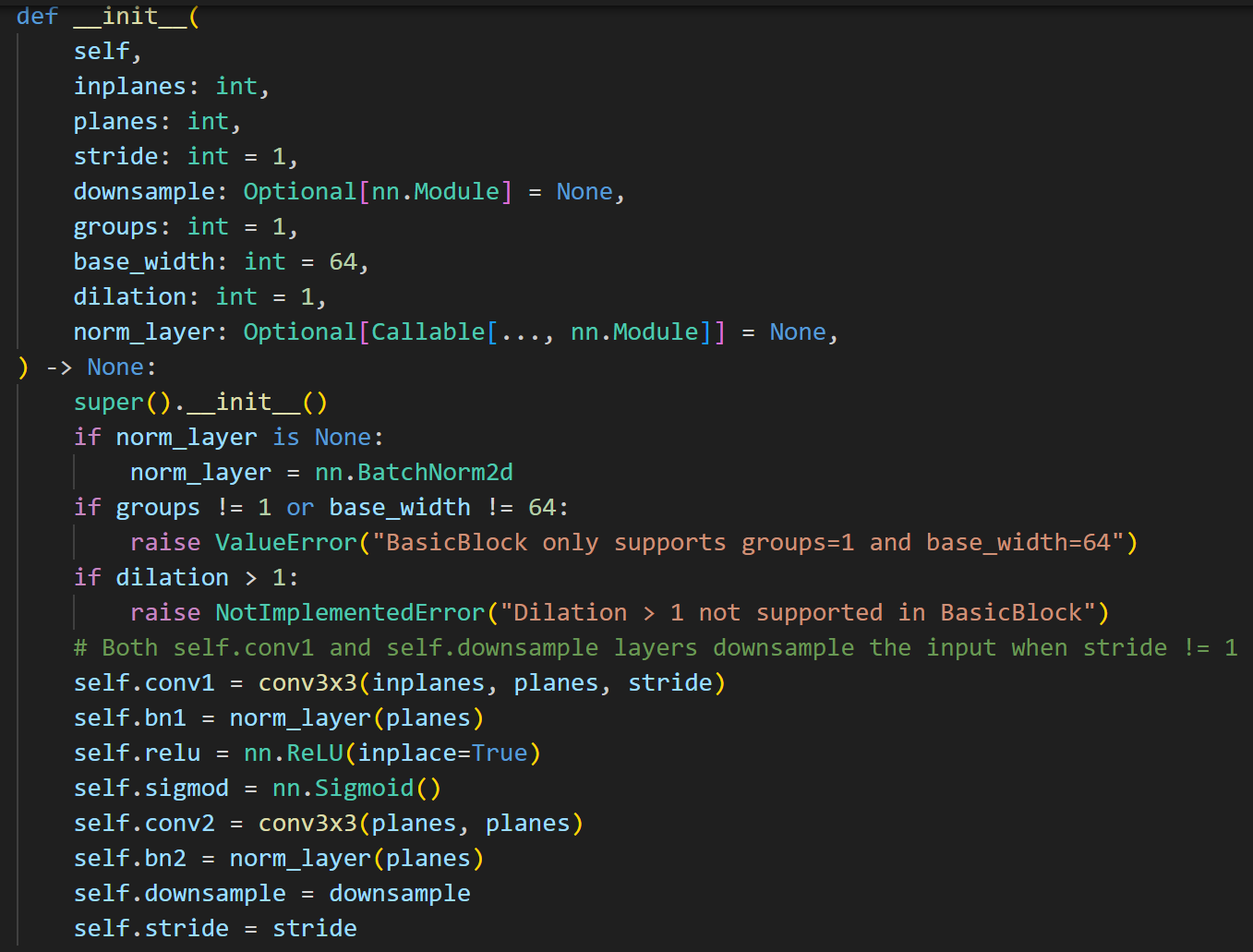


图 1.4 BasicBlock残差块结构

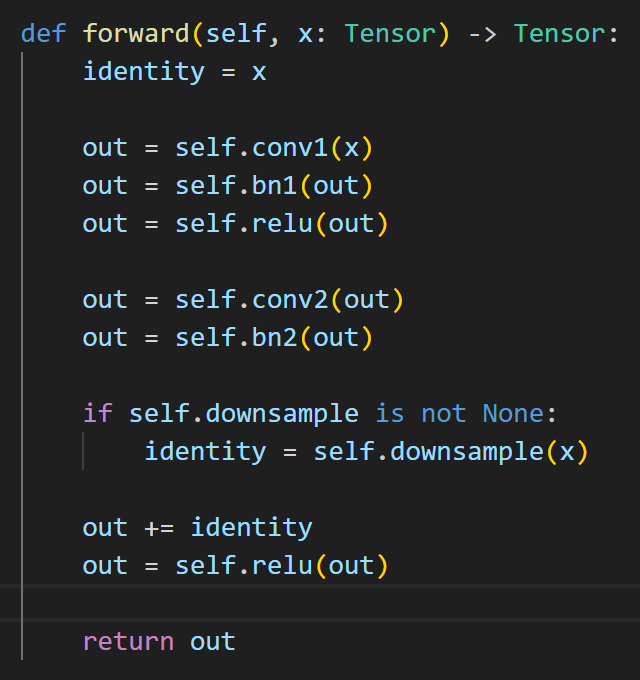


图 1.5 BasicBlock残差块结构



图 1.6 ResNet网络

### 训练结果

在此模型中，在单次训练中训练集损失随minibatch轮数的关系图 1.7、图 1.8和图 1.9所示，三张图片分别对应第1、第2、第3轮的训练集损失趋势，可以看到，在单轮训练内，训练集损失随着minibatch的增加而下降。在第一轮训练中尤为明显，除了有一个跳变外，训练集损失快速下降。而在第2和第3轮训练中，训练集损失随着minibatch在波动中下降，但趋势明显，因记录训练集损失会造成额外开销，在之后的模型测试中将不记录每一个Epoch内的训练集损失与minibatch关系。

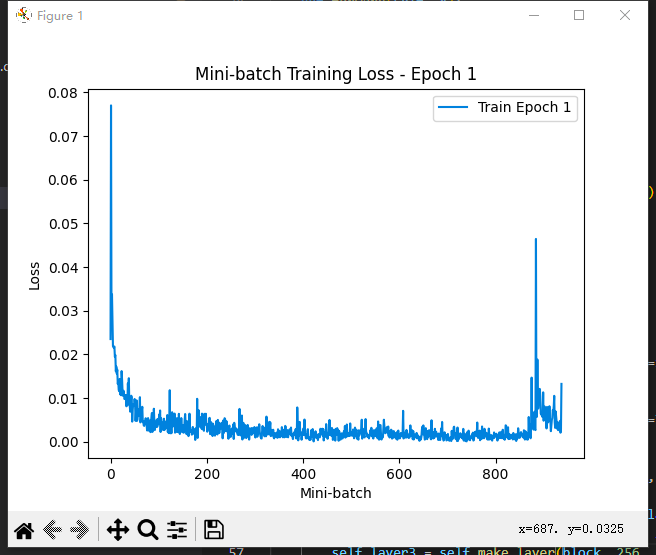


图 1.7 Epoch 1

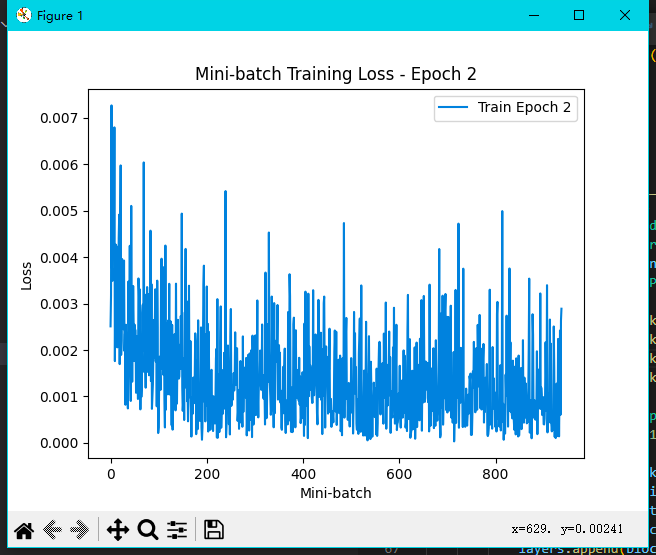


图 1.8 Epoch 2

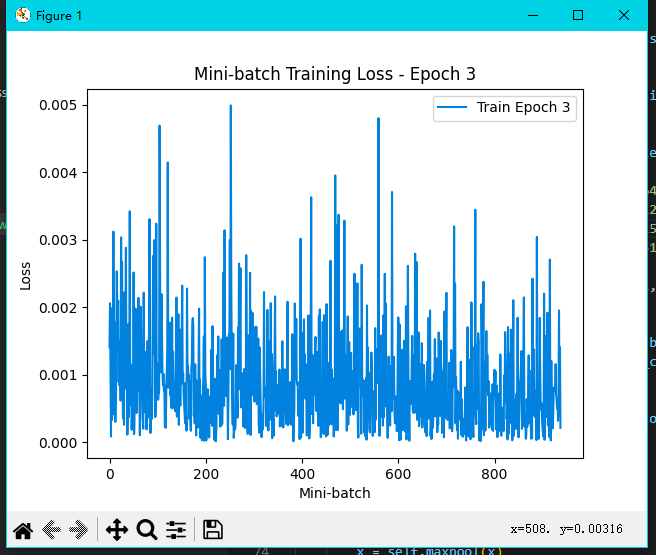


图 1.9 Epoch 3

训练20轮的训练集、测试集损失和准确率如图 1.10所示，最终测试集和训练集准确率如图 1.11所示。可以看到，随着训练轮数的增加，训练集和测试集损失下降，准确率吧上升，但到最后也许仍有下降的趋势，故增加训练轮数到30轮。

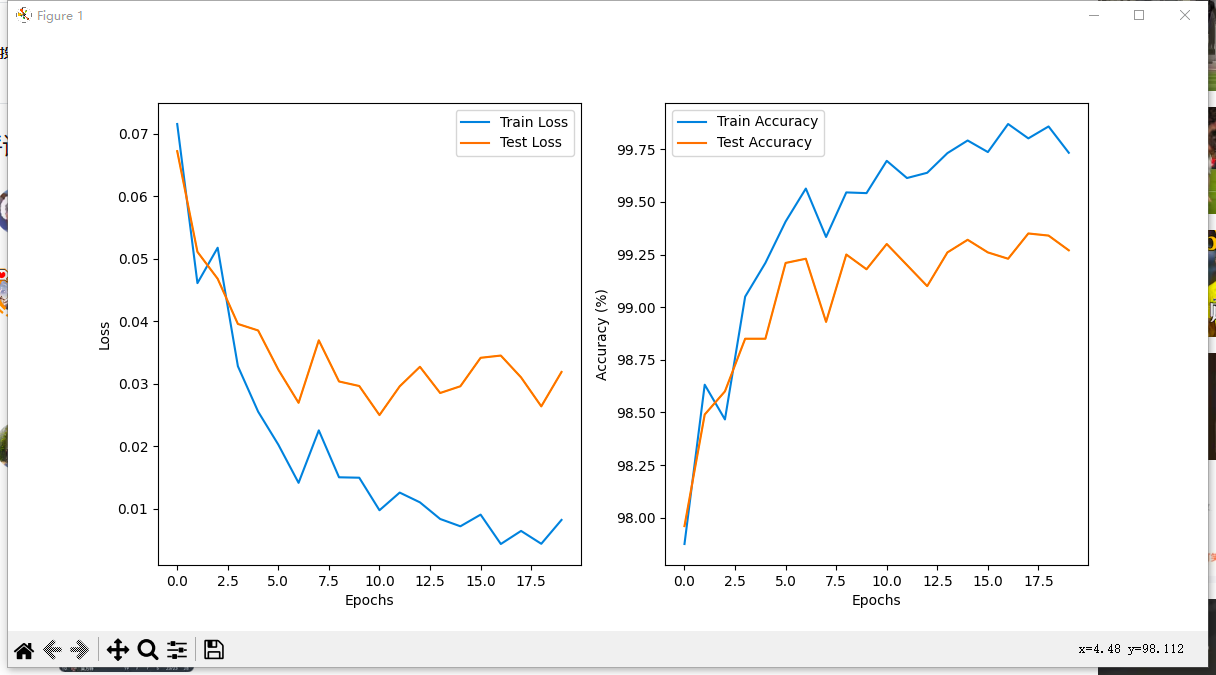


图 1.10 训练20轮



图 1.11 最终测试集和训练集准确率

训练30轮的结果如图 1.12所示，可以看到，在20轮后的训练集和测试集损失、

准确率趋于平缓，最后的准确率训练集和测试集的准确率分别为99.95%和99.30%，效果不错。

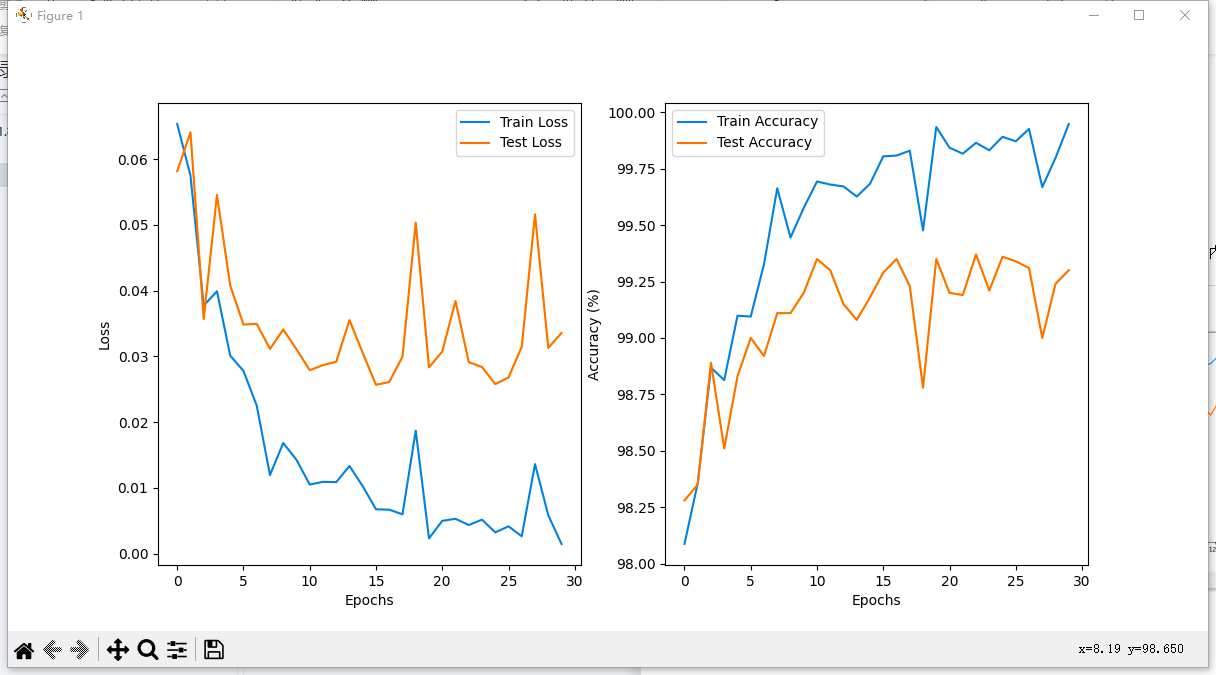


图 1.12 训练30轮

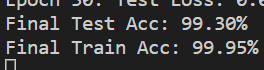


图 1.13 最终训练集和测试集准确率

## 模型2

在模型1的基础上，继续使用BasicBlock标准残差块，但扩大了残差块的规模，从[2,2,2,2]扩大到了[3,6,6,3]，第1层3个残差块，第2层6个残差块，第3层6个残差块，第4层3个残差块新的网络结构如图 1.14所示。



图 1.14 模型2网络结构

### 训练结果

用得到的网络进行30次训练，得到的训练集与测试集损失和准确率如图 1.15所示，最终的训练集和测试集准确率如图 1.16所示。因为残差块从模型1的8块增加到了18块，网络规模增加，所以训练时间比模型1长，，随着训练轮数增加，训练集和测试集损失下降，但略微出现了过拟合现象，测试集损失在10轮附近低点后略微上升，训练集损失一直下降。但效果不如模型1好，有可能是过大的网络模型对minst这类的小数据集适应性不太好。

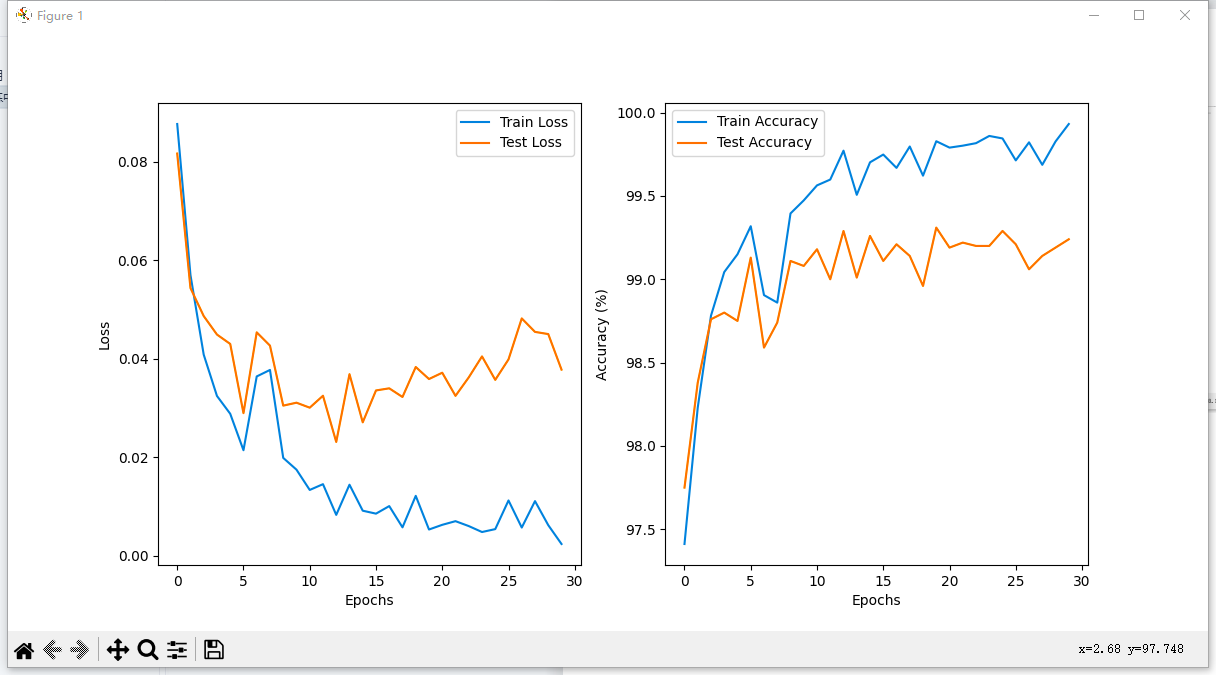


图 1.15 模型2训练30轮

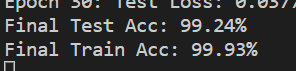


图 1.16 模型2最终训练集和测试集准确率

## 模型3

因考虑到过大的网络规模与较小的minst数据集兼容性并不好，故采用模型1的[2,2,2,2]的残差块规模，对标准残差块BasicBlock块进行改造，得到的新的残差块ModifyBasicBlock，该块的定义如图 1.17所示，将标准残差块的ReLU函数激活层换成了Sigmod函数激活层，以测试训练效果，新建的网络如图 1.18所示

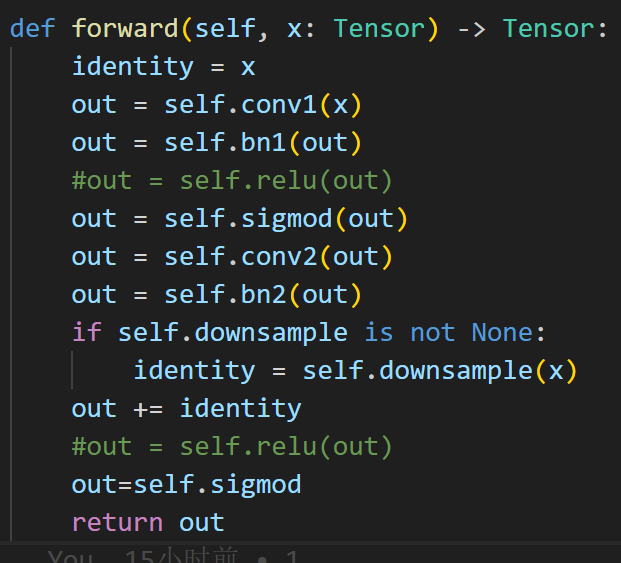


图 1.17



图 1.18

### 训练结果

对新的网络进行30次训练，得到的测试集与训练集损失与正确率如图 1.19所示，最终准确率如图 1.20所示，可以看到，用Sigmod激活函数后，初始的训练集和测试集损失较ReLU函数高，但随着训练轮数的增加快速下降，最后的测试集损失较使用ReLU函数的残差块高，效果不如使用ReLU激活函数的好。

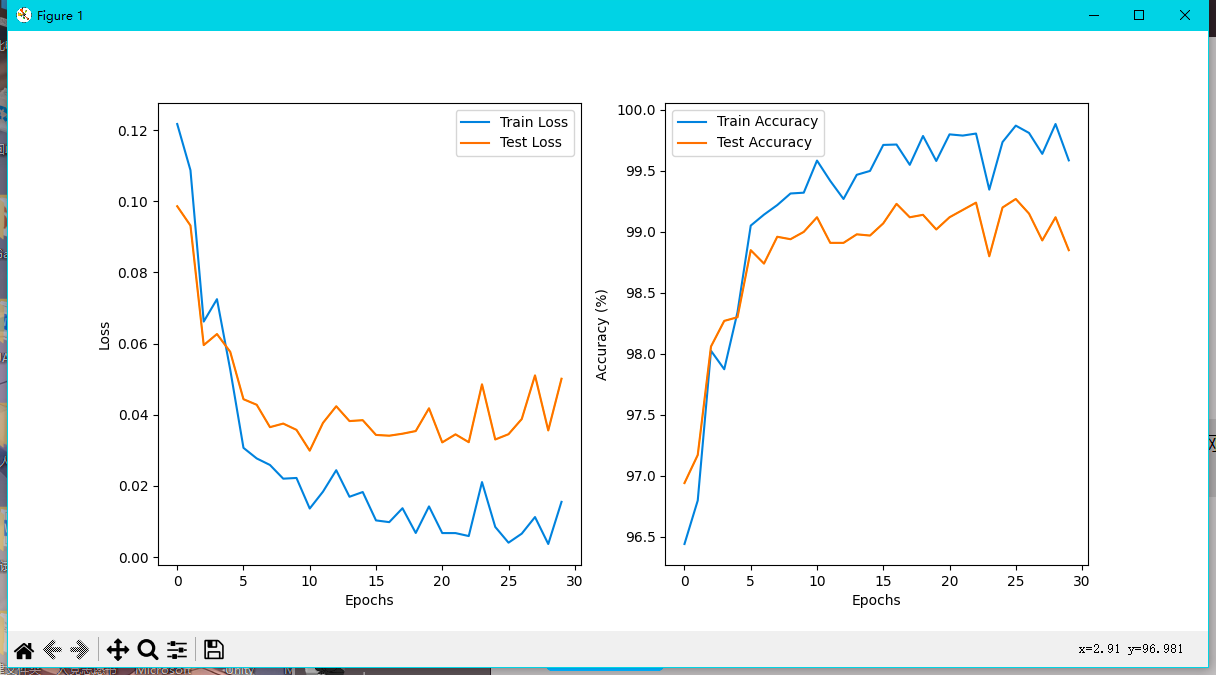


图 1.19



图 1.20

## 模型4

在模型1的基础上，对标准残差块BasicBlock进行修改，增加残差块的层数，得到ModifyBasicBlock\_NEW，结构如图 1.21所示，额外增加了一层ReLU函数激活层、卷积层和归一化层。

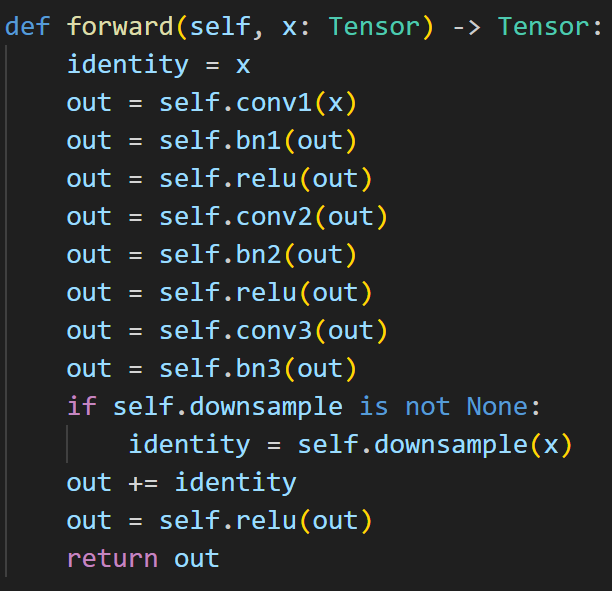


图 1.21

### 训练结果

进行30次训练，得到的训练集与测试集的损失和准确率如图 1.22所示，最终的训练集与测试集损失和准确率如图 1.23所示，可以看到，初始测试集和训练集损失较模型1较低，最终的训练集与测试集准确率也更高，但是效果不明显，且出现轻微过拟合现象，加深网络层次后训练开销大了一些。

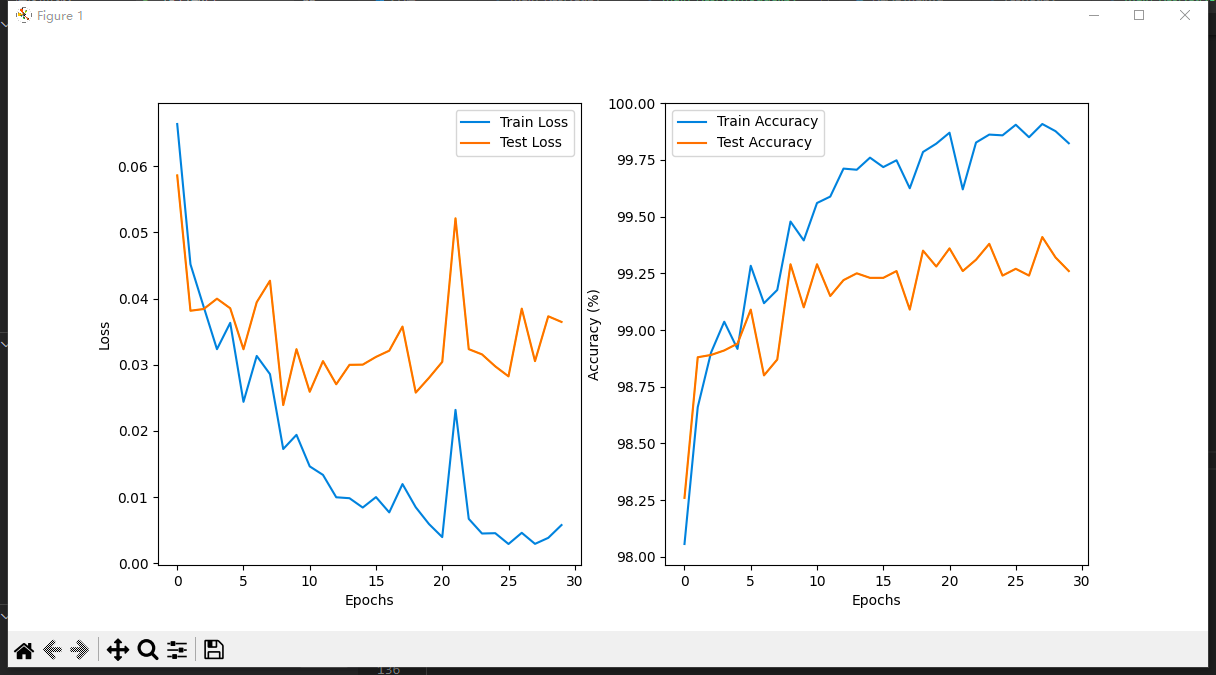


图 1.22



图 1.23

# 总结

使用了Pytorch的ResNet模块构建了残差网络，并完成对MNIST手写数字集的训练，准确率较高，相比于传统卷积，ResNet残差网络可以防止梯度爆炸等问题，防止过拟合，是优秀的网络。