根据老师提供的文档要求，我仿真了以下不同的model对于Mnist测试集的表现，参数设置如下—SNN，MLP，CNN：

# 单层神经网络--SNN

class SimpleNN(nn.Module):

def \_\_init\_\_(self):

super(SimpleNN, self).\_\_init\_\_()

self.fc = nn.Linear(28 \* 28, 10)

def forward(self, x):

x = x.view(-1, 28 \* 28)

return self.fc(x)

# MLP--2个隐藏层，激活函数ReLU，Leaky ReLU

class MLP(nn.Module):

def \_\_init\_\_(self):

super(MLP, self).\_\_init\_\_()

self.fc1 = nn.Linear(28 \* 28, 256) # 第一层，256个神经元

self.fc2 = nn.Linear(256, 128) # 第二层，128个神经元

self.fc3 = nn.Linear(128, 10) # 输出层

self.relu = nn.ReLU()

self.leaky\_relu = nn.LeakyReLU(0.01) # 设置Leaky ReLU的负斜率

def forward(self, x):

x = x.view(-1, 28 \* 28) # 展平输入图像

x = self.relu(self.fc1(x)) # 第一层激活，ReLU

x = self.leaky\_relu(self.fc2(x)) # 第二层激活，Leaky ReLU

return self.fc3(x) # 输出层

以及两种不同结构的CNN模型：

# CNN model

class CNN(nn.Module):

def \_\_init\_\_(self):

super(CNN, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(1, 32, kernel\_size=3, padding=1) # 输入通道1（灰度图），输出通道32

self.conv2 = nn.Conv2d(32, 64, kernel\_size=3, padding=1) # 输入通道32，输出通道64

self.pool = nn.MaxPool2d(2, 2) # 2x2最大池化

self.fc1 = nn.Linear(64 \* 7 \* 7, 128) # 全连接层，输入64\*7\*7，输出128

self.fc2 = nn.Linear(128, 10) # 输出层，输出10个类别

def forward(self, x):

x = self.pool(torch.relu(self.conv1(x))) # 第一层卷积+激活+池化

x = self.pool(torch.relu(self.conv2(x))) # 第二层卷积+激活+池化

x = x.view(-1, 64 \* 7 \* 7) # 展平

x = torch.relu(self.fc1(x)) # 全连接层+激活

return self.fc2(x) # 输出层

# which means CNN2

class LightCNNWithMixedPooling(nn.Module):

def \_\_init\_\_(self):

super(LightCNNWithMixedPooling, self).\_\_init\_\_()

# 第一层卷积：16个输出通道

self.conv1 = nn.Conv2d(1, 16, kernel\_size=5, padding=1)

self.maxpool = nn.MaxPool2d(2, 2) # 最大池化

# 第二层卷积：32个输出通道

self.conv2 = nn.Conv2d(16, 32, kernel\_size=5, padding=1)

self.avgpool = nn.AvgPool2d(2, 2) # 平均池化

# 全连接层，设置输入为卷积和池化后展平的特征大小

self.fc1 = nn.Linear(32 \* 5 \* 5, 64) # 输入大小调整为32 \* 5 \* 5

self.fc2 = nn.Linear(64, 10) # 输出层

def forward(self, x):

x = self.maxpool(torch.relu(self.conv1(x))) # 第一层卷积 + ReLU激活 + 最大池化

x = self.avgpool(torch.relu(self.conv2(x))) # 第二层卷积 + ReLU激活 + 平均池化

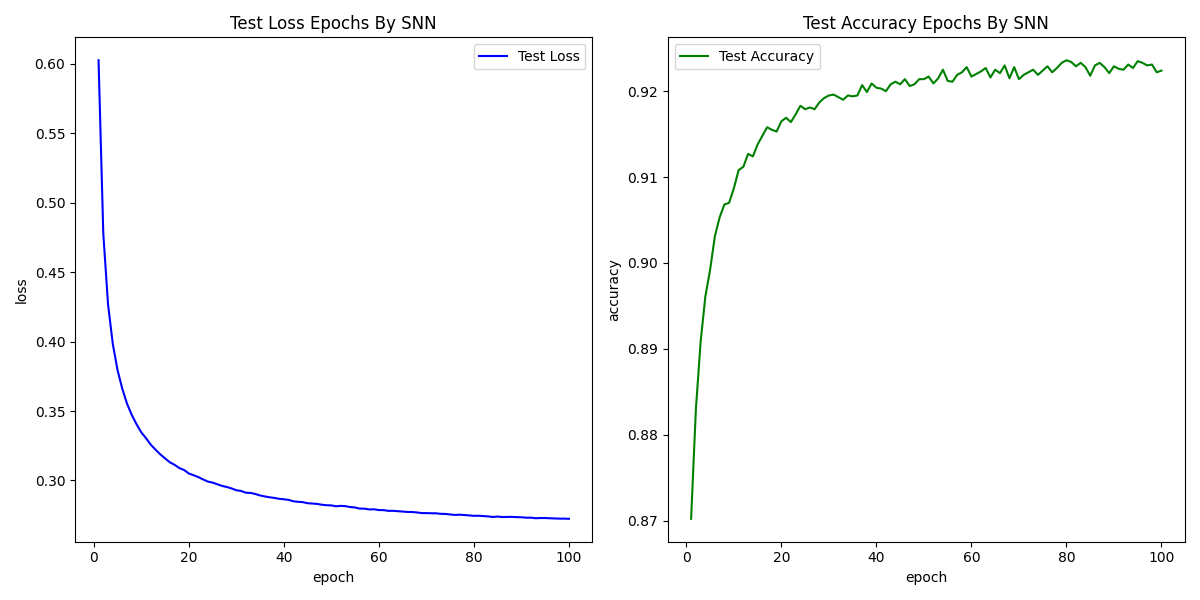
x = x.view(x.size(0), -1) # 展平特征图

x = torch.relu(self.fc1(x)) # 全连接层 + 激活函数

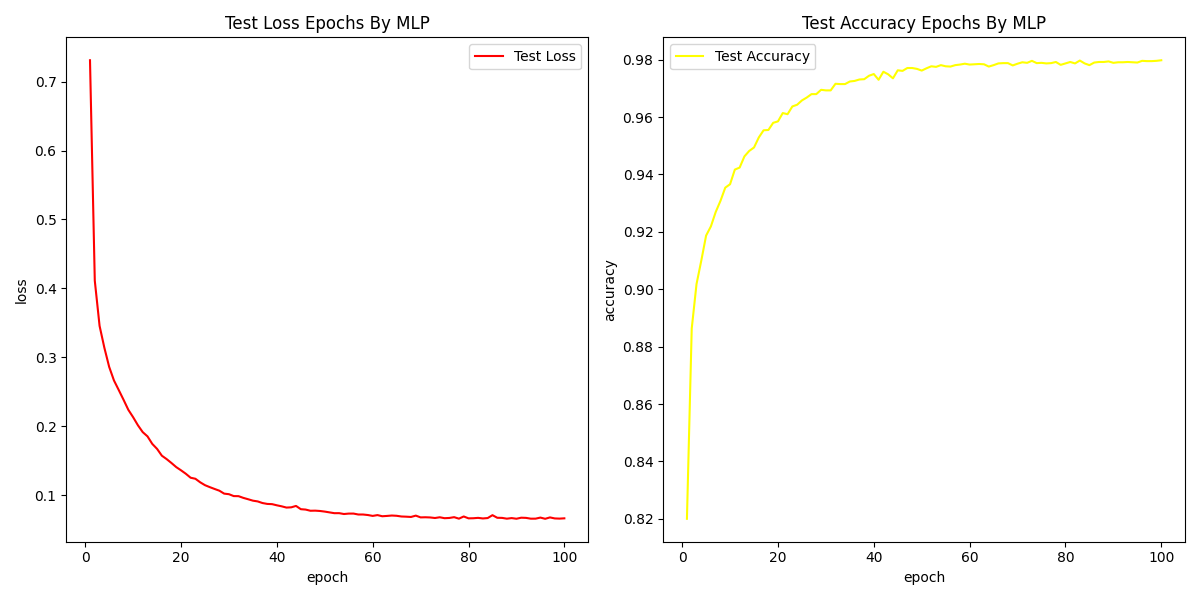
return self.fc2(x) # 输出层

针对于Mnist的测试集，仿真了100个epoch，学习率为0.01，四种不同的model对应的loss以及accuracy结果图如下所示：

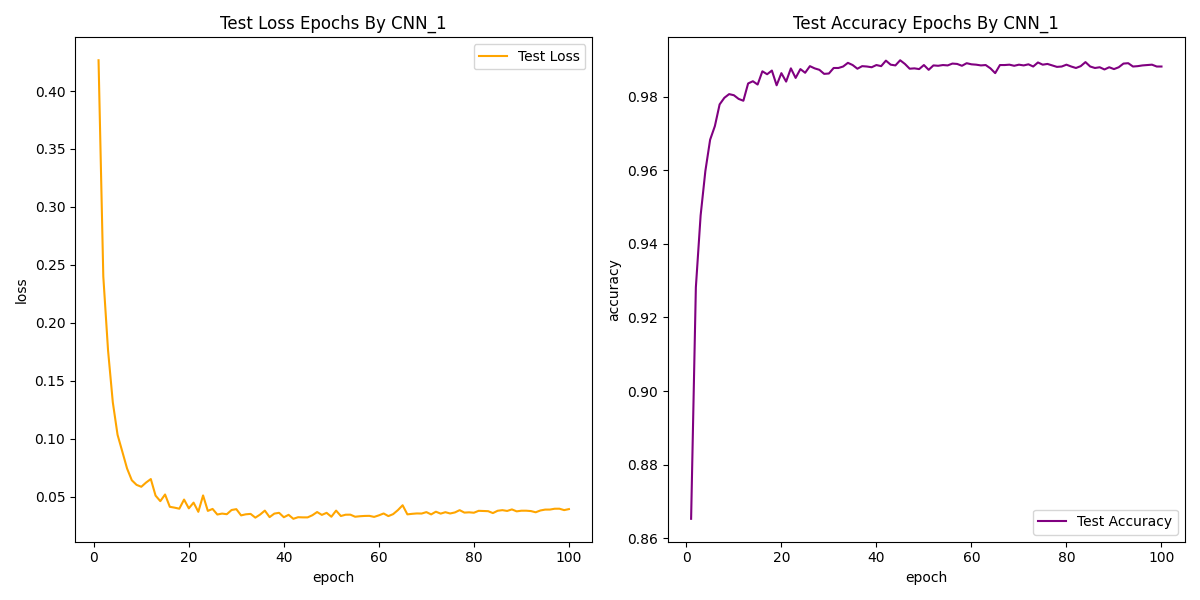
1、SNN



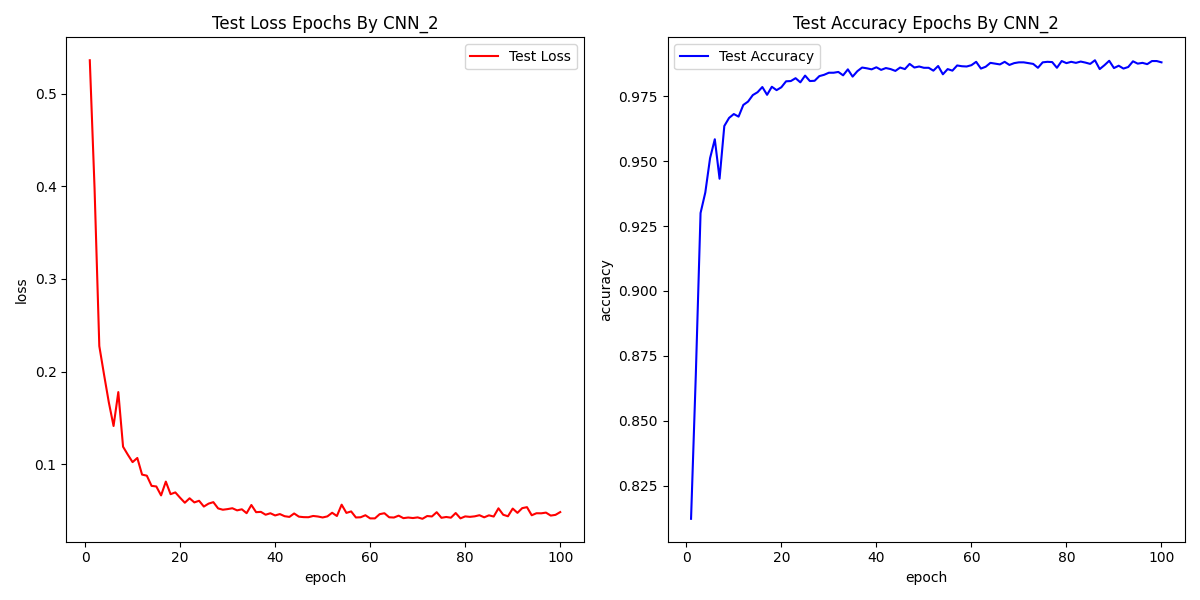
2、MLP



3、CNN1



4、CNN2



观察结果可以知道MLP的表现结果优于SNN，MLP包含多个隐藏层，而SNN只有一个隐藏层（或者没有隐藏层）。通过增加隐藏层，MLP能够学习更复杂、更抽象的特征，而SNN只能学习简单的线性特征，另一方面则是MLP每层通常会使用非线性激活函数（如ReLU、Sigmoid、Tanh等），使得网络能够学习到输入数据的非线性特征。而SNN在输出层之前仅使用一次非线性激活，因此其对非线性特征的建模能力较弱。而CNN的表现则是高于其余两者，这是因为其卷积核以及池化策略不仅有效的降低了特征数量，还有有效的提取了更深层次的特征，从CNN1和CNN2的表现来看，增大kernel\_size不一定能带来效能的提升，CNN2层结构也比CNN1少，这也可能是效果降低的原因之一。