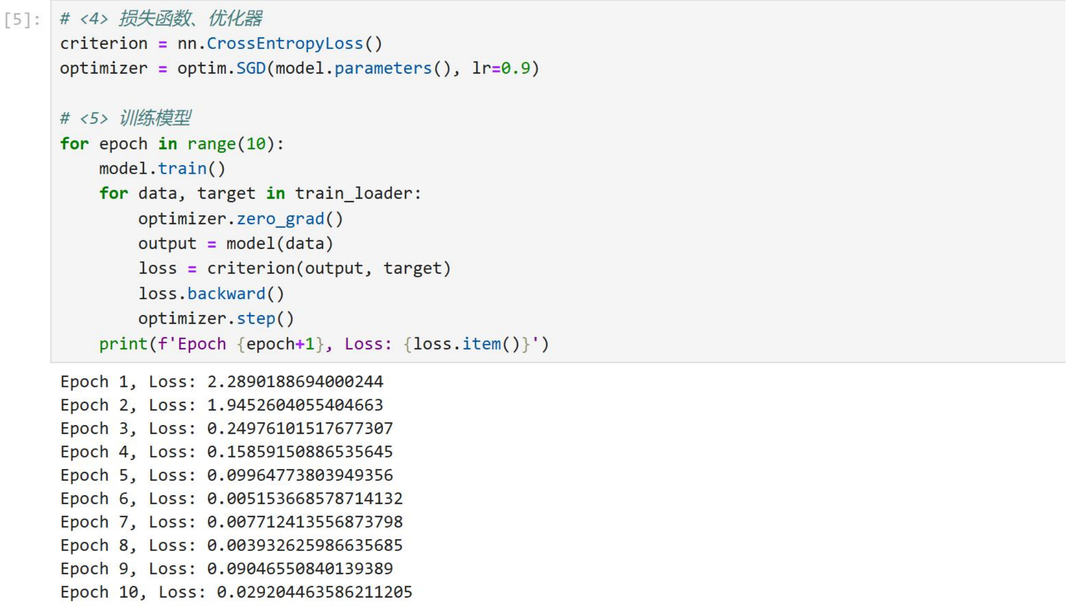
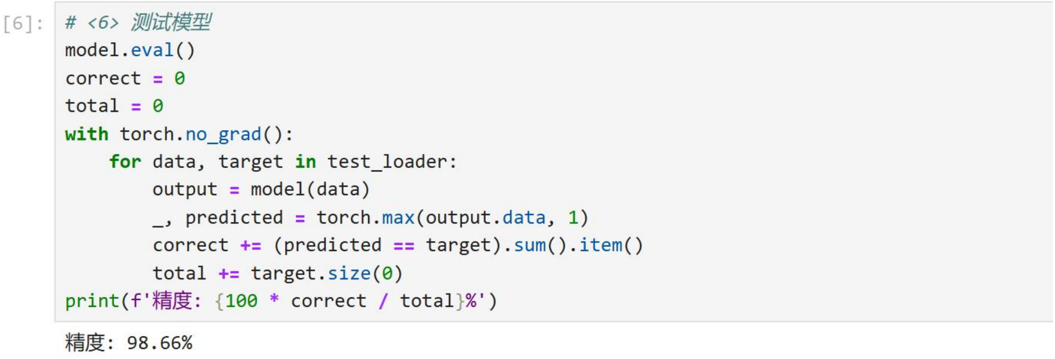
原始结果：



```
[5]: # <4> 损失函数、优化器
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.9)

# <5> 训练模型
for epoch in range(10):
    model.train()
    for data, target in train_loader:
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()
    print(f'Epoch {epoch+1}, Loss: {loss.item()}')
```

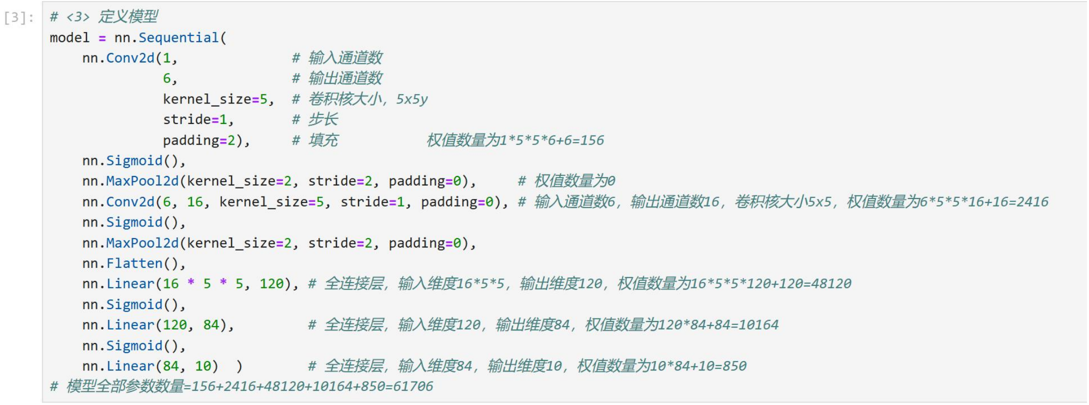
```
Epoch 1, Loss: 2.2890188694000244
Epoch 2, Loss: 1.9452604055404663
Epoch 3, Loss: 0.24976101517677307
Epoch 4, Loss: 0.15859150886535645
Epoch 5, Loss: 0.09964773803949356
Epoch 6, Loss: 0.005153668578714132
Epoch 7, Loss: 0.007712413556873798
Epoch 8, Loss: 0.003932625986635685
Epoch 9, Loss: 0.09046550840139389
Epoch 10, Loss: 0.029204463586211205
```



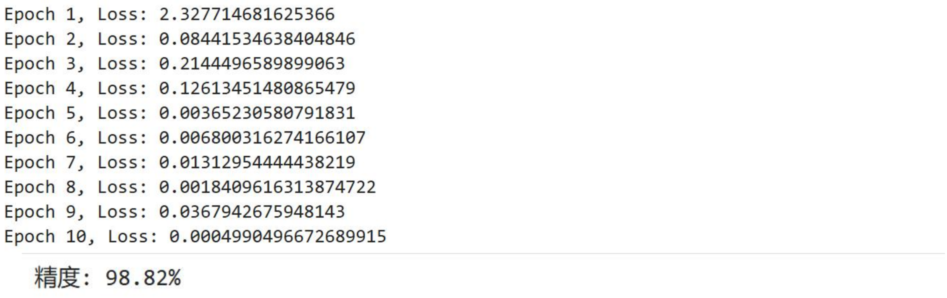
```
[6]: # <6> 测试模型
model.eval()
correct = 0
total = 0
with torch.no_grad():
    for data, target in test_loader:
        output = model(data)
        _, predicted = torch.max(output.data, 1)
        correct += (predicted == target).sum().item()
        total += target.size(0)
print(f'精度: {100 * correct / total}%')
```

```
精度: 98.66%
```

A）改了最大池化（精度和原始结果比出现了提升）：



```
[3]: # <3> 定义模型
model = nn.Sequential(
    nn.Conv2d(1,              # 输入通道数
              6,              # 输出通道数
              kernel_size=5,  # 卷积核大小, 5x5y
              stride=1,       # 步长
              padding=2),     # 填充           权值数量为1*5*5*6+6=156
    nn.Sigmoid(),
    nn.MaxPool2d(kernel_size=2, stride=2, padding=0),     # 权值数量为0
    nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=0), # 输入通道数6, 输出通道数16, 卷积核大小5x5, 权值数量为6*5*5*16+16=2416
    nn.Sigmoid(),
    nn.MaxPool2d(kernel_size=2, stride=2, padding=0),
    nn.Flatten(),
    nn.Linear(16 * 5 * 5, 120), # 全连接层, 输入维度16*5*5, 输出维度120, 权值数量为16*5*5*120+120=48120
    nn.Sigmoid(),
    nn.Linear(120, 84),         # 全连接层, 输入维度120, 输出维度84, 权值数量为120*84+84=10164
    nn.Sigmoid(),
    nn.Linear(84, 10)  )        # 全连接层, 输入维度84, 输出维度10, 权值数量为10*84+10=850
# 模型全部参数数量=156+2416+48120+10164+850=61706
```

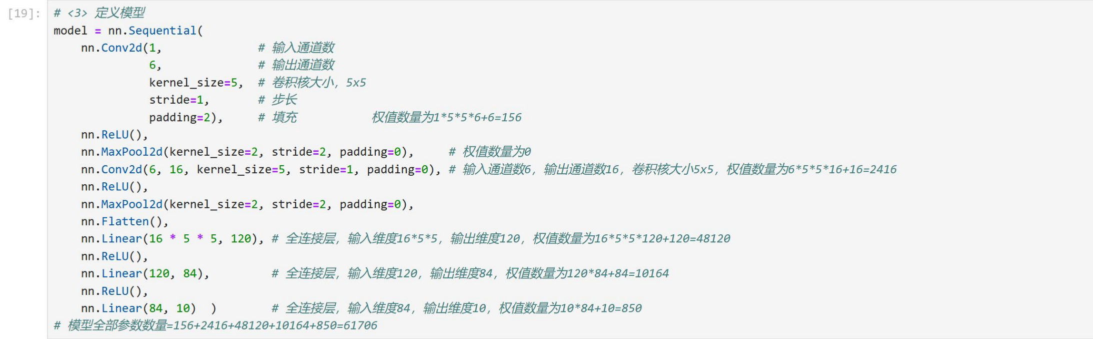


```
Epoch 1, Loss: 2.327714681625366
Epoch 2, Loss: 0.08441534638404846
Epoch 3, Loss: 0.2144496589899063
Epoch 4, Loss: 0.12613451480865479
Epoch 5, Loss: 0.00365230580791831
Epoch 6, Loss: 0.006800316274166107
Epoch 7, Loss: 0.01312954444438219
Epoch 8, Loss: 0.0018409616313874722
Epoch 9, Loss: 0.0367942675948143
Epoch 10, Loss: 0.0004990496672689915
```

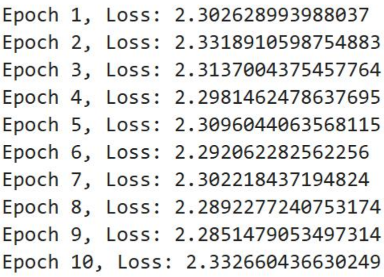
```
精度: 98.82%
```

由于更改此部分后准确率出现了提升，因此保留最大池化的更动，继续完成后续部分。

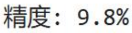
B）将 Sigmoid 改成了 ReLU：



```
[19]: # <3> 定义模型
      model = nn.Sequential(
          nn.Conv2d(1,              # 输入通道数
                    6,              # 输出通道数
                    kernel_size=5,  # 卷积核大小, 5x5
                    stride=1,       # 步长
                    padding=2),     # 填充           权值数量为1*5*5*6+6=156
          nn.ReLU(),
          nn.MaxPool2d(kernel_size=2, stride=2, padding=0),     # 权值数量为0
          nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=0), # 输入通道数6, 输出通道数16, 卷积核大小5x5, 权值数量为6*5*5*16+16=2416
          nn.ReLU(),
          nn.MaxPool2d(kernel_size=2, stride=2, padding=0),
          nn.Flatten(),
          nn.Linear(16 * 5 * 5, 120), # 全连接层, 输入维度16*5*5, 输出维度120, 权值数量为16*5*5*120+120=48120
          nn.ReLU(),
          nn.Linear(120, 84),         # 全连接层, 输入维度120, 输出维度84, 权值数量为120*84+84=10164
          nn.ReLU(),
          nn.Linear(84, 10)  )        # 全连接层, 输入维度84, 输出维度10, 权值数量为10*84+10=850
      # 模型全部参数数量=156+2416+48120+10164+850=61706
```



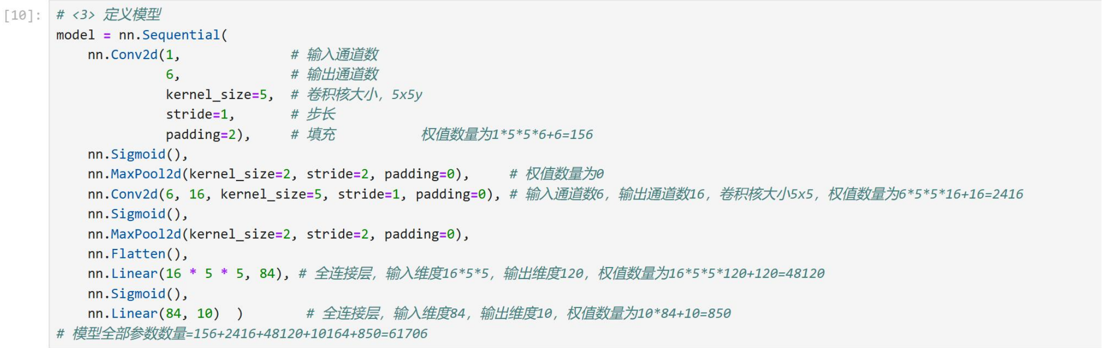
```
Epoch 1, Loss: 2.302628993988037
Epoch 2, Loss: 2.3318910598754883
Epoch 3, Loss: 2.3137004375457764
Epoch 4, Loss: 2.2981462478637695
Epoch 5, Loss: 2.3096044063568115
Epoch 6, Loss: 2.292062282562256
Epoch 7, Loss: 2.302218437194824
Epoch 8, Loss: 2.2892277240753174
Epoch 9, Loss: 2.2851479053497314
Epoch 10, Loss: 2.332660436630249
```



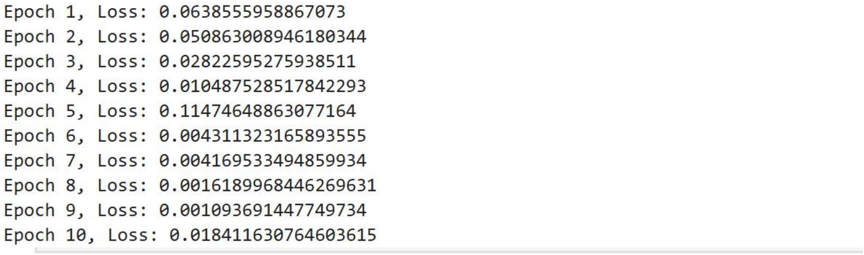
```
精度： 9.8%
```

可能是 dead ReLU/梯度消失问题，所以损失不降低，精度也很低。

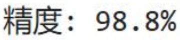
C）改了全连接层数：



```
[10]: # <3> 定义模型
      model = nn.Sequential(
          nn.Conv2d(1,              # 输入通道数
                    6,              # 输出通道数
                    kernel_size=5,  # 卷积核大小, 5x5y
                    stride=1,       # 步长
                    padding=2),     # 填充           权值数量为1*5*5*6+6=156
          nn.Sigmoid(),
          nn.MaxPool2d(kernel_size=2, stride=2, padding=0),     # 权值数量为0
          nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=0), # 输入通道数6, 输出通道数16, 卷积核大小5x5, 权值数量为6*5*5*16+16=2416
          nn.Sigmoid(),
          nn.MaxPool2d(kernel_size=2, stride=2, padding=0),
          nn.Flatten(),
          nn.Linear(16 * 5 * 5, 84), # 全连接层, 输入维度16*5*5, 输出维度120, 权值数量为16*5*5*120+120=48120
          nn.Sigmoid(),
          nn.Linear(84, 10)  )        # 全连接层, 输入维度84, 输出维度10, 权值数量为10*84+10=850
      # 模型全部参数数量=156+2416+48120+10164+850=61706
```



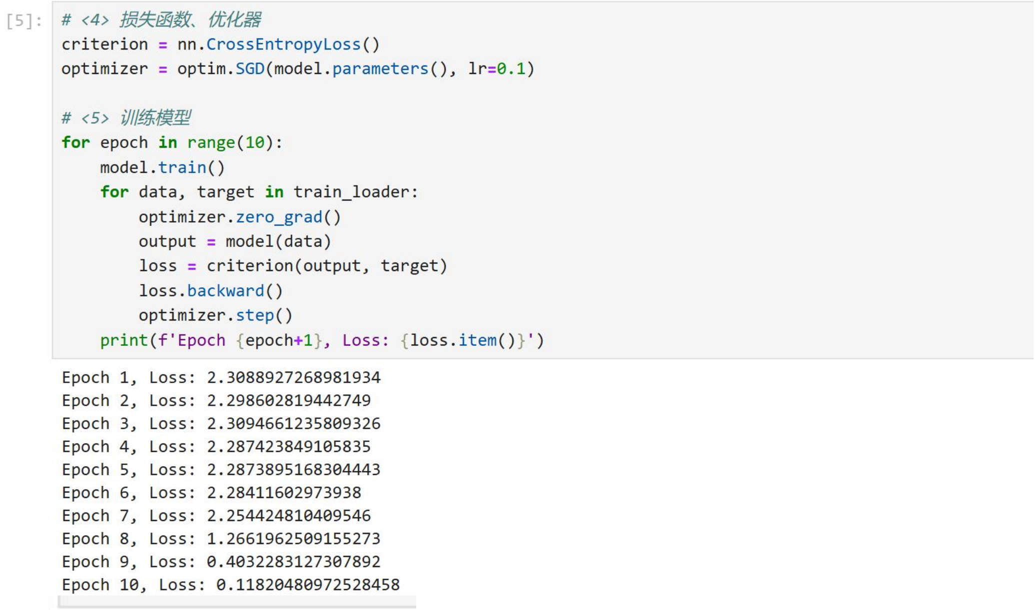
```
Epoch 1, Loss: 0.0638555958867073
Epoch 2, Loss: 0.050863008946180344
Epoch 3, Loss: 0.02822595275938511
Epoch 4, Loss: 0.010487528517842293
Epoch 5, Loss: 0.11474648863077164
Epoch 6, Loss: 0.004311323165893555
Epoch 7, Loss: 0.004169533494859934
Epoch 8, Loss: 0.0016189968446269631
Epoch 9, Loss: 0.001093691447749734
Epoch 10, Loss: 0.018411630764603615
```



```
精度： 98.8%
```

虽然比原始结果得精度高，但改了最大池化和减少全连接层数 vs 只改最大池化相比低了 0.2%，因此后续不在减少全链接层的基础上训练。

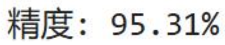
D）改了学习率（因为学习率从 0.9 降低至 0.1，因此损失下降得比较慢）：



```
[5]: # <4> 损失函数、优化器
     criterion = nn.CrossEntropyLoss()
     optimizer = optim.SGD(model.parameters(), lr=0.1)

     # <5> 训练模型
     for epoch in range(10):
         model.train()
         for data, target in train_loader:
             optimizer.zero_grad()
             output = model(data)
             loss = criterion(output, target)
             loss.backward()
             optimizer.step()
         print(f'Epoch {epoch+1}, Loss: {loss.item()}')
```

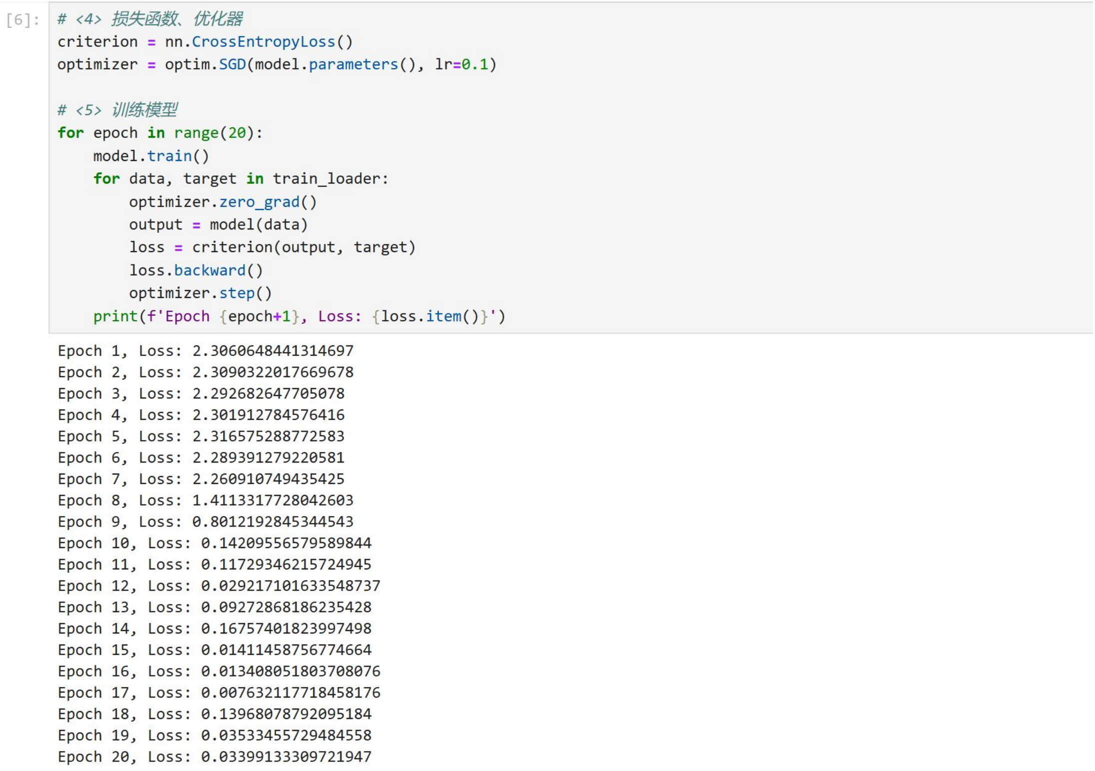
```
Epoch 1, Loss: 2.3088927268981934
Epoch 2, Loss: 2.298602819442749
Epoch 3, Loss: 2.3094661235809326
Epoch 4, Loss: 2.287423849105835
Epoch 5, Loss: 2.2873895168304443
Epoch 6, Loss: 2.28411602973938
Epoch 7, Loss: 2.254424810409546
Epoch 8, Loss: 1.2661962509155273
Epoch 9, Loss: 0.4032283127307892
Epoch 10, Loss: 0.11820480972528458
```



精度：95.31%

虽然结果不如只更改最大池化的情况，但后续尝试在保留学习率=0.1 的情况下增加 epoch 进行观察。

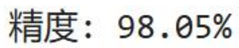
E）将 Epoch 增加到 20（损失率变得比较正常，准确率也和原始数据 差不多。）：



```
[6]: # <4> 损失函数、优化器
     criterion = nn.CrossEntropyLoss()
     optimizer = optim.SGD(model.parameters(), lr=0.1)

     # <5> 训练模型
     for epoch in range(20):
         model.train()
         for data, target in train_loader:
             optimizer.zero_grad()
             output = model(data)
             loss = criterion(output, target)
             loss.backward()
             optimizer.step()
         print(f'Epoch {epoch+1}, Loss: {loss.item()}')
```

```
Epoch 1, Loss: 2.3060648441314697
Epoch 2, Loss: 2.3090322017669678
Epoch 3, Loss: 2.292682647705078
Epoch 4, Loss: 2.301912784576416
Epoch 5, Loss: 2.316575288772583
Epoch 6, Loss: 2.289391279220581
Epoch 7, Loss: 2.260910749435425
Epoch 8, Loss: 1.4113317728042603
Epoch 9, Loss: 0.8012192845344543
Epoch 10, Loss: 0.14209556579589844
Epoch 11, Loss: 0.11729346215724945
Epoch 12, Loss: 0.029217101633548737
Epoch 13, Loss: 0.09272868186235428
Epoch 14, Loss: 0.16757401823997498
Epoch 15, Loss: 0.01411458756774664
Epoch 16, Loss: 0.013408051803708076
Epoch 17, Loss: 0.007632117718458176
Epoch 18, Loss: 0.13968078792095184
Epoch 19, Loss: 0.03533455729484558
Epoch 20, Loss: 0.03399133309721947
```



精度：98.05%

感想：由于开始没搞清楚题目要求，我将题目要求当成了一个个的步骤去顺序做，再改了 ReLU 的“步骤”后后面所有的“步骤”都是损失率无法降低，精度 10%左右。后续咨询了助教才发现原来不是顺序的步骤，而是分开的要求，所以其实本次作业不难但我却意外花了不少时间。