实验报告二

1 总述

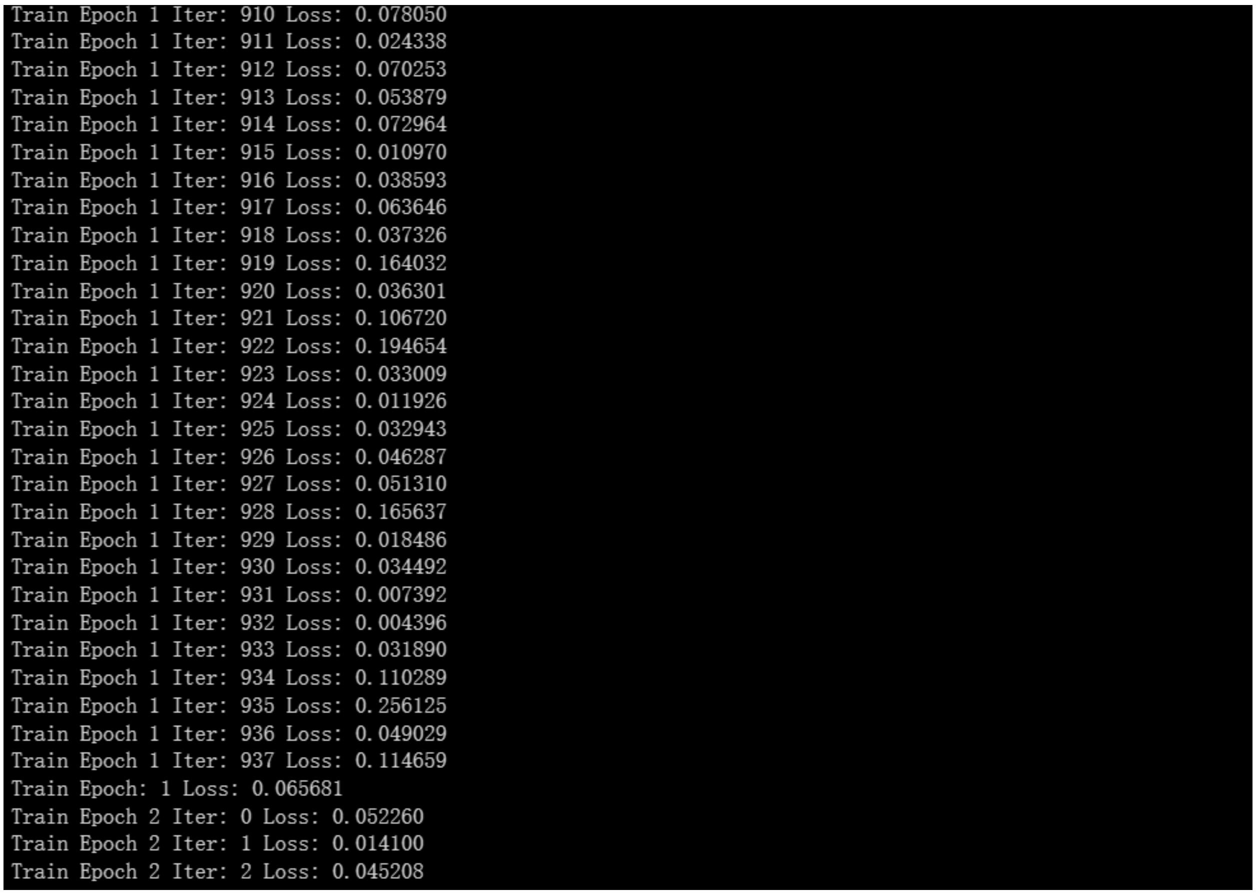
1.1 神经网络架构

- 神经网络架构模仿的是ResNet-18的架构;
- 激活函数选用ReLU函数;
- 网络中包括五个卷积组，将后四个卷积组命名为`layer1`-`layer4`，每个`layer`包括两个残差块，每个残差块包括两个卷积层，最后通过一个全连接层输出，所以总共为18层:
  - 第1个卷积组只包含1次卷积计算操作，卷积核为7×7， 步长为2，padding为3;
  - `layer1` 包含两个残差块，卷积核大小都为3×3, `layer1`步长为1，padding为1，所以`layer1`不会进行降采样;
  - `layer2-layer4`，每个 `layer` 包含 2 个 残差块，卷积核大小都为3×3,但是第 1 个 残差块 的第 1 个卷积层的步长为2，会进行降采样。在进行`shortcut`连接时，会经过`downsample`层，进行降采样和降维。而第二个卷积层步长为1，不会进行下采样， `shortcut`连接也不会经过`downsample`层

1.2 损失

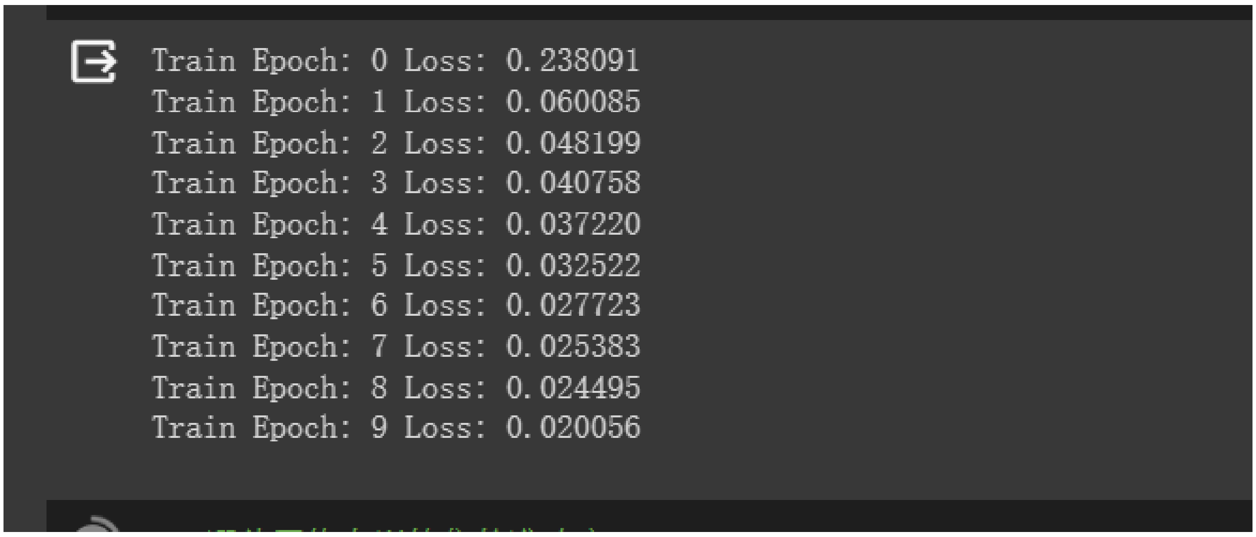
- 共10个epoch，训练集batch大小为64，学习率为0.01

- 截取训练时的部分输出如图所示：



```
Train Epoch 1 Iter: 910 Loss: 0.078050
Train Epoch 1 Iter: 911 Loss: 0.024338
Train Epoch 1 Iter: 912 Loss: 0.070253
Train Epoch 1 Iter: 913 Loss: 0.053879
Train Epoch 1 Iter: 914 Loss: 0.072964
Train Epoch 1 Iter: 915 Loss: 0.010970
Train Epoch 1 Iter: 916 Loss: 0.038593
Train Epoch 1 Iter: 917 Loss: 0.063646
Train Epoch 1 Iter: 918 Loss: 0.037326
Train Epoch 1 Iter: 919 Loss: 0.164032
Train Epoch 1 Iter: 920 Loss: 0.036301
Train Epoch 1 Iter: 921 Loss: 0.106720
Train Epoch 1 Iter: 922 Loss: 0.194654
Train Epoch 1 Iter: 923 Loss: 0.033009
Train Epoch 1 Iter: 924 Loss: 0.011926
Train Epoch 1 Iter: 925 Loss: 0.032943
Train Epoch 1 Iter: 926 Loss: 0.046287
Train Epoch 1 Iter: 927 Loss: 0.051310
Train Epoch 1 Iter: 928 Loss: 0.165637
Train Epoch 1 Iter: 929 Loss: 0.018486
Train Epoch 1 Iter: 930 Loss: 0.034492
Train Epoch 1 Iter: 931 Loss: 0.007392
Train Epoch 1 Iter: 932 Loss: 0.004396
Train Epoch 1 Iter: 933 Loss: 0.031890
Train Epoch 1 Iter: 934 Loss: 0.110289
Train Epoch 1 Iter: 935 Loss: 0.256125
Train Epoch 1 Iter: 936 Loss: 0.049029
Train Epoch 1 Iter: 937 Loss: 0.114659
Train Epoch: 1 Loss: 0.065681
Train Epoch 2 Iter: 0 Loss: 0.052260
Train Epoch 2 Iter: 1 Loss: 0.014100
Train Epoch 2 Iter: 2 Loss: 0.045208
```

- 每个epoch结束后的loss如图所示：

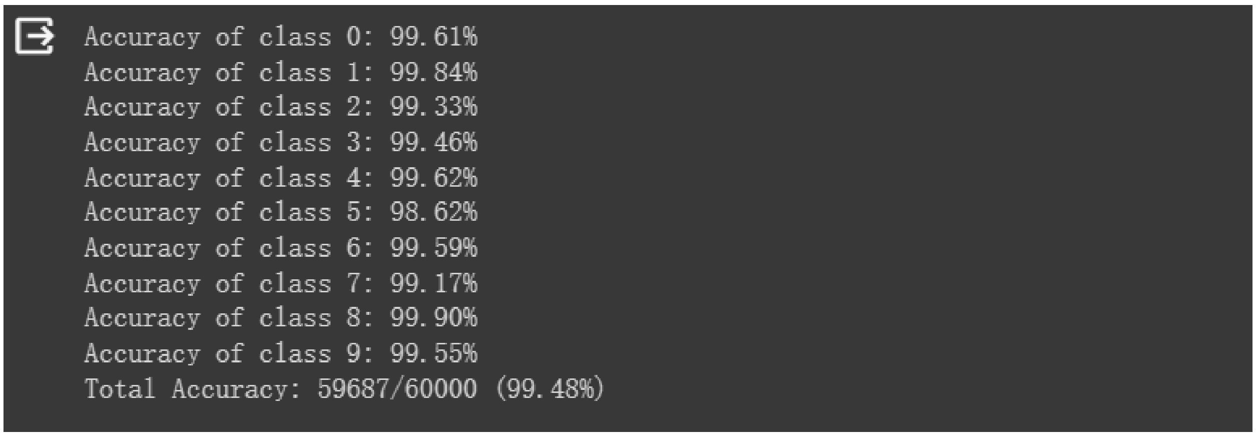


```
Train Epoch: 0 Loss: 0.238091
Train Epoch: 1 Loss: 0.060085
Train Epoch: 2 Loss: 0.048199
Train Epoch: 3 Loss: 0.040758
Train Epoch: 4 Loss: 0.037220
Train Epoch: 5 Loss: 0.032522
Train Epoch: 6 Loss: 0.027723
Train Epoch: 7 Loss: 0.025383
Train Epoch: 8 Loss: 0.024495
Train Epoch: 9 Loss: 0.020056
```

- 模型在10个epoch之后，Loss缩小至了0.20056

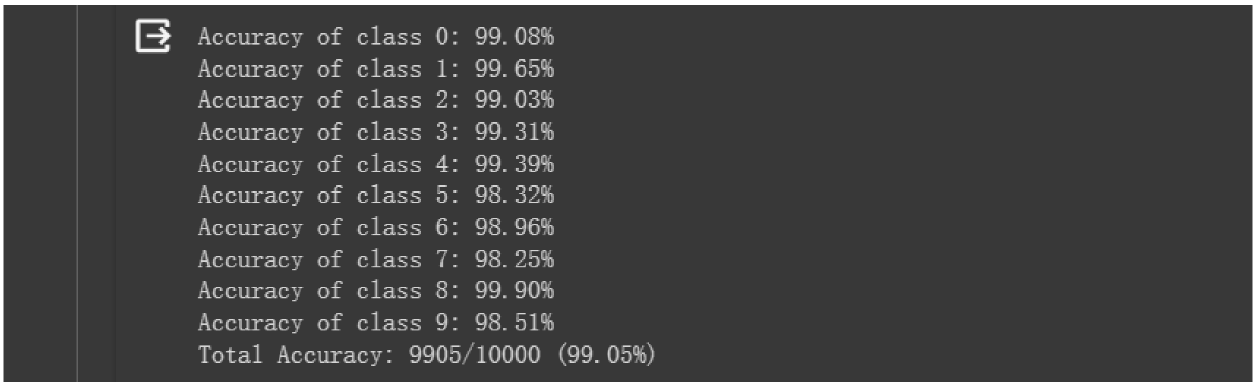
1.3 准确率

- 在训练集上的准确率：



```
Accuracy of class 0: 99.61%
Accuracy of class 1: 99.84%
Accuracy of class 2: 99.33%
Accuracy of class 3: 99.46%
Accuracy of class 4: 99.62%
Accuracy of class 5: 98.62%
Accuracy of class 6: 99.59%
Accuracy of class 7: 99.17%
Accuracy of class 8: 99.90%
Accuracy of class 9: 99.55%
Total Accuracy: 59687/60000 (99.48%)
```

- 在测试集上的准确率：



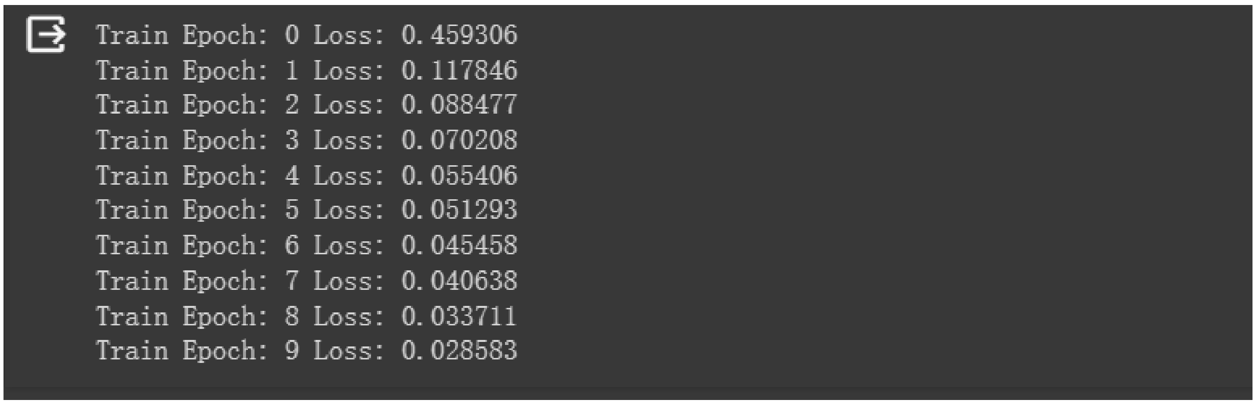
```
Accuracy of class 0: 99.08%
Accuracy of class 1: 99.65%
Accuracy of class 2: 99.03%
Accuracy of class 3: 99.31%
Accuracy of class 4: 99.39%
Accuracy of class 5: 98.32%
Accuracy of class 6: 98.96%
Accuracy of class 7: 98.25%
Accuracy of class 8: 99.90%
Accuracy of class 9: 98.51%
Total Accuracy: 9905/10000 (99.05%)
```

- 可以看出在ResNet18架构下，训练集和测试集的精度都达到了百分之九十九以上，其中训练效果最好的数字为'8'；训练效果最差的数字为'2'和'7'，猜测是因为'2'和'7'比较相似，存在混淆的现象

2 尝试一：使用不同的激活函数

2.1 sigmoid函数

- 我们主要关注使用不同激活函数之后，模型损失的收敛速度，如图所示：

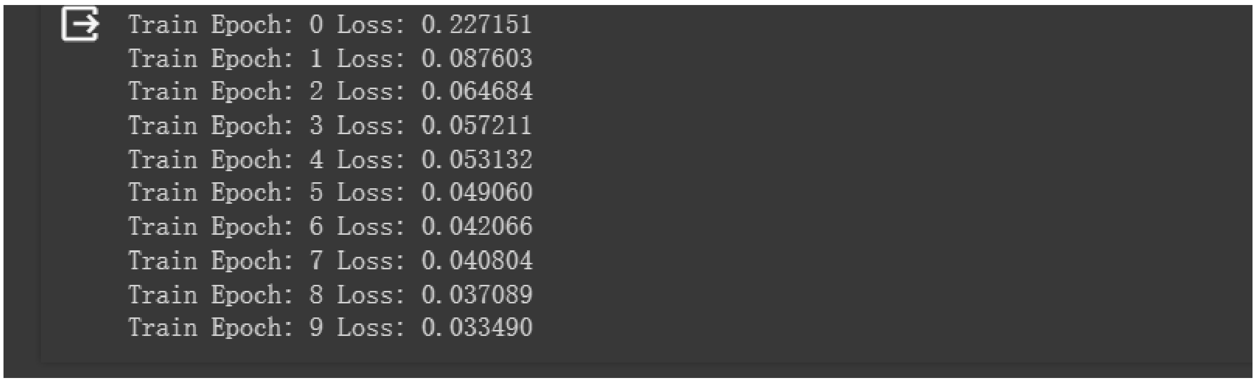


```
Train Epoch: 0 Loss: 0.459306
Train Epoch: 1 Loss: 0.117846
Train Epoch: 2 Loss: 0.088477
Train Epoch: 3 Loss: 0.070208
Train Epoch: 4 Loss: 0.055406
Train Epoch: 5 Loss: 0.051293
Train Epoch: 6 Loss: 0.045458
Train Epoch: 7 Loss: 0.040638
Train Epoch: 8 Loss: 0.033711
Train Epoch: 9 Loss: 0.028583
```

- 对比使用relu函数的结果，我们可以看出sigmoid函数的性能要差于relu函数

2.2 tanh函数

- 收敛速度如图所示：



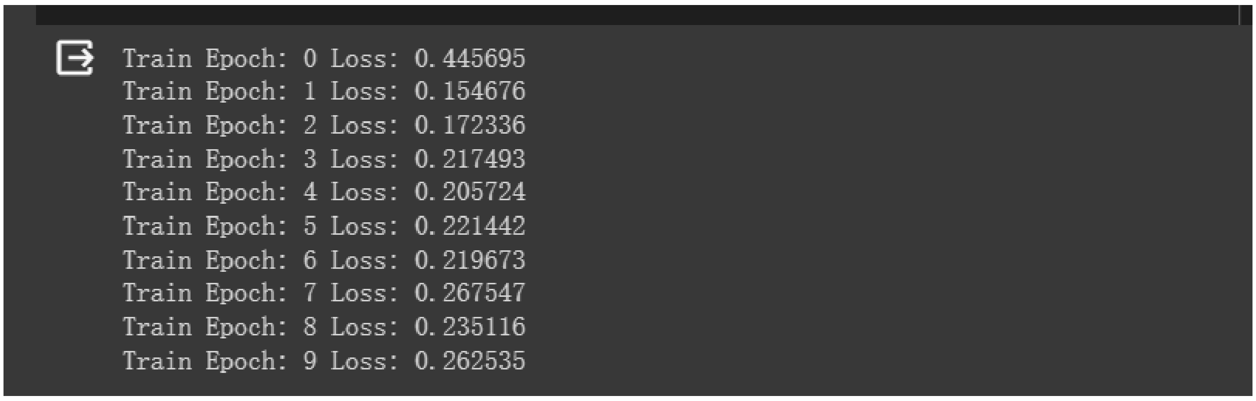
```
Train Epoch: 0 Loss: 0.227151
Train Epoch: 1 Loss: 0.087603
Train Epoch: 2 Loss: 0.064684
Train Epoch: 3 Loss: 0.057211
Train Epoch: 4 Loss: 0.053132
Train Epoch: 5 Loss: 0.049060
Train Epoch: 6 Loss: 0.042066
Train Epoch: 7 Loss: 0.040804
Train Epoch: 8 Loss: 0.037089
Train Epoch: 9 Loss: 0.033490
```

- 对比relu和sigmoid的结果，可以发现tanh函数的性能略逊于relu，优于sigmoid

3 尝试二：修改超参数

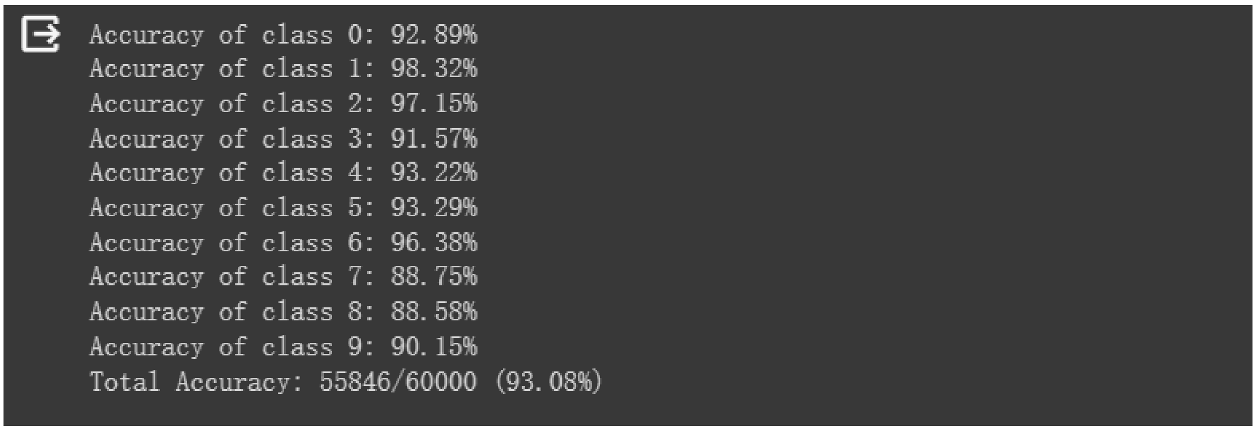
3.1 修改学习率

- 将学习率从0.01更改至0.1
- 训练过程中的损失如图所示：



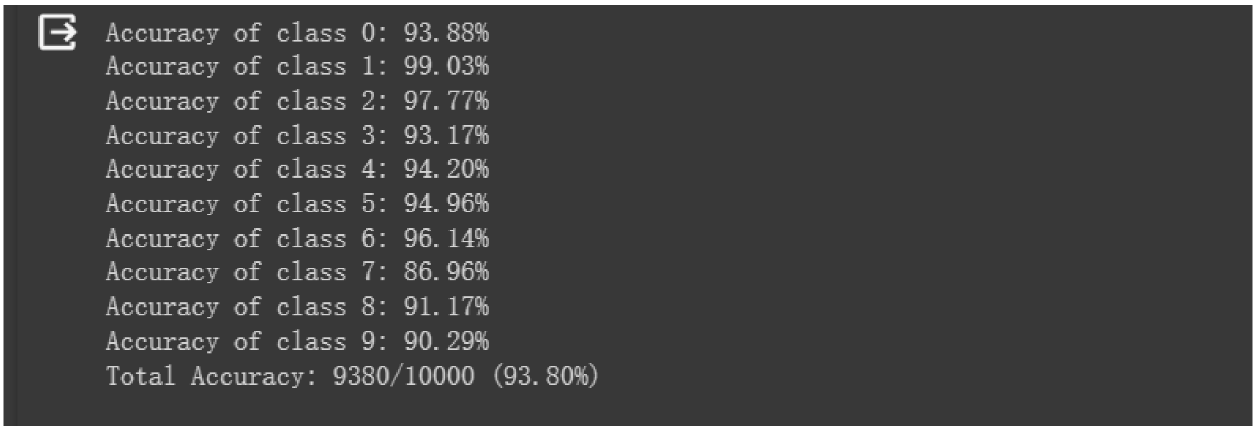
```
Train Epoch: 0 Loss: 0.445695
Train Epoch: 1 Loss: 0.154676
Train Epoch: 2 Loss: 0.172336
Train Epoch: 3 Loss: 0.217493
Train Epoch: 4 Loss: 0.205724
Train Epoch: 5 Loss: 0.221442
Train Epoch: 6 Loss: 0.219673
Train Epoch: 7 Loss: 0.267547
Train Epoch: 8 Loss: 0.235116
Train Epoch: 9 Loss: 0.262535
```

- 在训练集上的精确率：



```
Accuracy of class 0: 92.89%
Accuracy of class 1: 98.32%
Accuracy of class 2: 97.15%
Accuracy of class 3: 91.57%
Accuracy of class 4: 93.22%
Accuracy of class 5: 93.29%
Accuracy of class 6: 96.38%
Accuracy of class 7: 88.75%
Accuracy of class 8: 88.58%
Accuracy of class 9: 90.15%
Total Accuracy: 55846/60000 (93.08%)
```

- 在测试集上的精确率：

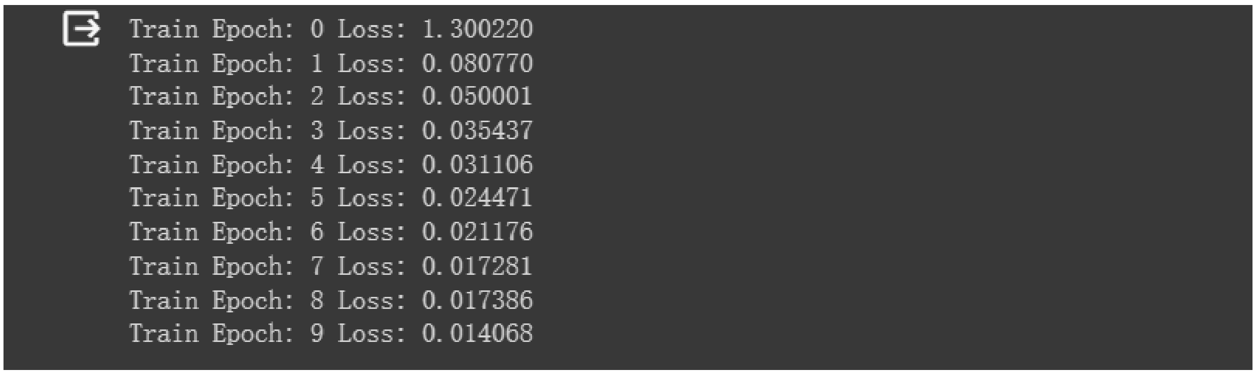


```
Accuracy of class 0: 93.88%
Accuracy of class 1: 99.03%
Accuracy of class 2: 97.77%
Accuracy of class 3: 93.17%
Accuracy of class 4: 94.20%
Accuracy of class 5: 94.96%
Accuracy of class 6: 96.14%
Accuracy of class 7: 86.96%
Accuracy of class 8: 91.17%
Accuracy of class 9: 90.29%
Total Accuracy: 9380/10000 (93.80%)
```

- 可以看出在学习率为0.1的情况下，模型的精度下降了不少，学习率设置为0.01是一个更优的选择

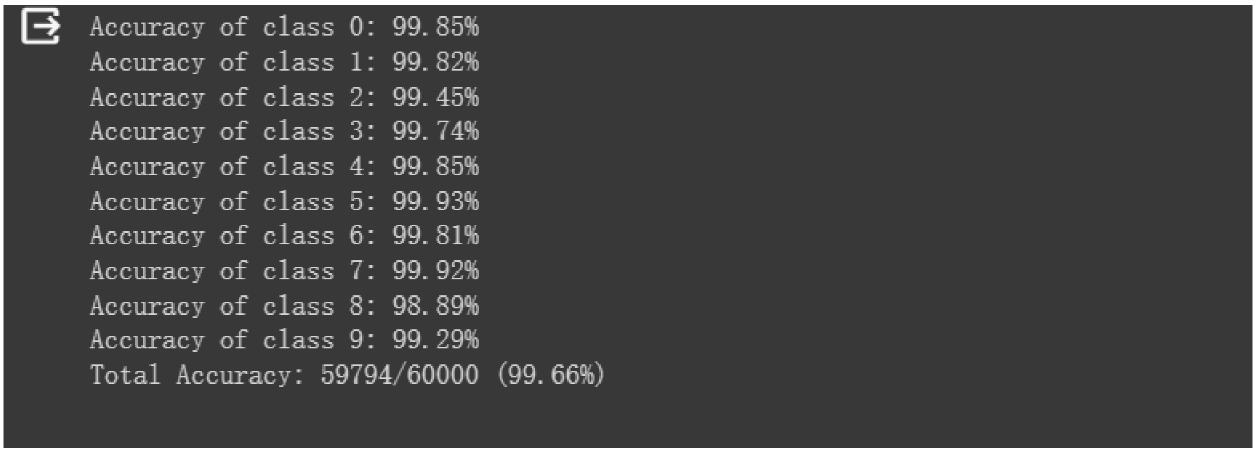
3.2 修改batch_size

- 将训练集的batch_size从64修改为1000
- 训练过程中的损失如图所示：



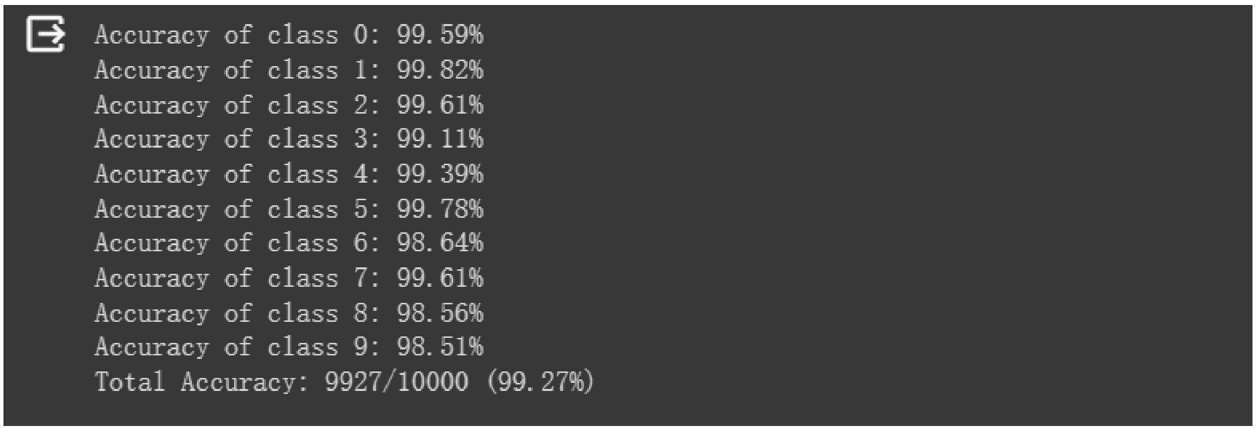
```
Train Epoch: 0 Loss: 1.300220
Train Epoch: 1 Loss: 0.080770
Train Epoch: 2 Loss: 0.050001
Train Epoch: 3 Loss: 0.035437
Train Epoch: 4 Loss: 0.031106
Train Epoch: 5 Loss: 0.024471
Train Epoch: 6 Loss: 0.021176
Train Epoch: 7 Loss: 0.017281
Train Epoch: 8 Loss: 0.017386
Train Epoch: 9 Loss: 0.014068
```

- 在训练集上的精确率：



```
Accuracy of class 0: 99.85%
Accuracy of class 1: 99.82%
Accuracy of class 2: 99.45%
Accuracy of class 3: 99.74%
Accuracy of class 4: 99.85%
Accuracy of class 5: 99.93%
Accuracy of class 6: 99.81%
Accuracy of class 7: 99.92%
Accuracy of class 8: 98.89%
Accuracy of class 9: 99.29%
Total Accuracy: 59794/60000 (99.66%)
```

- 在测试集上的准确率：



```
Accuracy of class 0: 99.59%
Accuracy of class 1: 99.82%
Accuracy of class 2: 99.61%
Accuracy of class 3: 99.11%
Accuracy of class 4: 99.39%
Accuracy of class 5: 99.78%
Accuracy of class 6: 98.64%
Accuracy of class 7: 99.61%
Accuracy of class 8: 98.56%
Accuracy of class 9: 98.51%
Total Accuracy: 9927/10000 (99.27%)
```

- 可以看出，无论是模型收敛速度还是精确率，batch_size修改后的模型效果都优于修改前的。batch_size设置为1000是一个更合理的超参数