

# 基于 Pytorch 的 CNN 实现

171860659 吴紫航

## 实现步骤

- 1.加载并标准化 Mnist 数据集(60000 训练集和 10000 测试集)
- 2.原训练集随机划分(55000 训练集和 5000 验证集)
- 3.定义卷积神经网络模型
- 4.定义损失函数
- 5.用训练集数据训练网络参数并用验证集验证
- 6.用测试集数据测试训练成果
- 7.人工调整超参数

## 测试性能

对应第 3 题第 1 问, 运行代码 `CNN_main.py` 会打印训练和测试的细节

(1) 训练过程打印: training loss、validation loss、validation accuracy

注: 若连续两次 validation loss 上升, 则停止训练, 并记录 validation loss 最小时的轮次 EPOCH, 测试时采用该轮次对应的网络结果

```
epoch: 1
training progress: 0 %
training progress: 50 %
training progress: 100 %
training loss: 0.3194630
validation loss: 0.0988722|
Accuracy of the network on the 5000 validation images: 96.9800000 %
epoch: 2
training progress: 0 %
training progress: 50 %
training progress: 100 %
training loss: 0.0825507
validation loss: 0.0920766
Accuracy of the network on the 5000 validation images: 97.3800000 %
```

Figure 1 - 训练过程

(2) 测试过程打印：test accuracy、accuracy for each class

```
Accuracy of the network on the 10000 test images: 99 %
Accuracy of  zero : 99 %
Accuracy of  one : 99 %
Accuracy of  two : 99 %
Accuracy of three : 100 %
Accuracy of  four : 99 %
Accuracy of  five : 99 %
Accuracy of  six : 99 %
Accuracy of seven : 97 %
Accuracy of eight : 97 %
Accuracy of  nine : 98 %
```

Figure 2 - 测试结果

网络结构

对应第 3 题的第 2 问， 本题的神经网络结构图如下

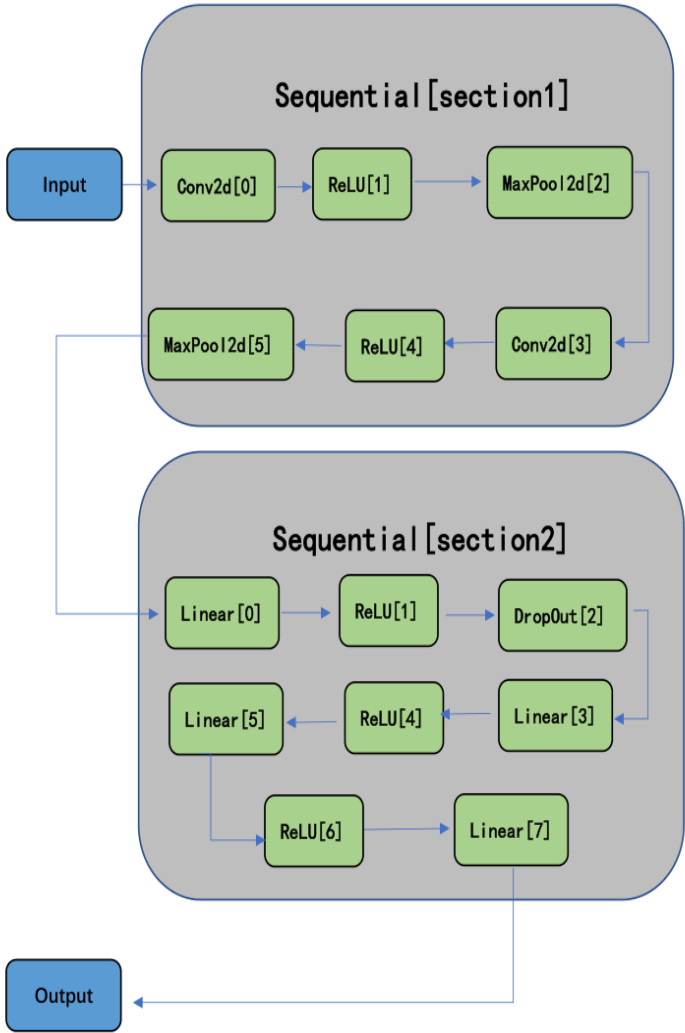


Figure 3 - 卷积神经网络结构

**单个输入图片的数据流变换过程为：**

**(1)** Section1-Conv2d[0]

输入图片( $28 \times 28 \times 1$ ); 卷积核( $3 \times 3 \times 64$ ); padding=1; stride=1

输出( $28 \times 28 \times 64$ )

**(2)** Section1-ReLU[1]

输入( $28 \times 28 \times 64$ )

输出( $28 \times 28 \times 64$ )

**(3)** Section1-MaxPool2d[2]

输入( $28 \times 28 \times 64$ )

输出( $14 \times 14 \times 64$ )

**(4)** Section1-Conv2d[3]

输入图片( $14 \times 14 \times 64$ ); 卷积核( $3 \times 3 \times 128$ ); padding=1; stride=1

输出( $14 \times 14 \times 128$ )

**(5)** Section1-ReLU[4]

输入( $14 \times 14 \times 128$ )

输出( $14 \times 14 \times 128$ )

**(6)** Section1-MaxPool2d[5]

输入( $14 \times 14 \times 128$ )

输出( $7 \times 7 \times 128$ )

**(7)** Section2-Linear[0]

输入( $7 \times 7 \times 128$ )

输出(1024)

**(8)** Section2-ReLU[1]

输入(1024)

输出(1024)

**(9)** Section2-Dropout[2]

输入(1024)

输出(1024)

**(10)** Section2-Linear[3]

输入(1024)

输出(120)

**(11)** Section2-ReLU[4]

输入(120)

输出(120)

**(12)** Section2-Linear[5]

输入(120)

输出(84)

**(13)** Section2-ReLU[6]

输入(84)

输出(84)

**(14)** Section2-Linear[7]

输入(84)

输出(10)

**即最后得到 10 个类的预测概率**

# 优化方法和超参数确定

对应于第 3 题第 3 问

采用了两种优化方法：Adam 和 SGD，每批次大小 Batch size=25

learning rate 和 epoch 的学习情况(关于 accuracy) 如下表：

(1) Adam:

Epoch/轮次	LR=5e-3/%	LR=1e-3/%	LR=5e-4/%	LR=1e-4/%
1	95.38	97.64	96.76	96.98
2	96.22	97.16	98.16	97.38
3	95.84	98.48	98.18	98.36
4	96.54	98.66	98.78	98.48
5	96.28	98.86	97.70	98.48
6	96.32	99.14	98.66	98.56
7	96.68	99.08	98.58	98.88
8	96.44	98.82	98.50	98.76
9	96.88	/	/	99.00
10	/	/	/	98.86
11	/	/	/	98.84

Table 1 - Adam 方法超参数学习情况

由表得，adam 法的学习速率 lr 设为 1e-3 最合适，过大会难以收敛，过小则收敛过慢；在此学习率下，迭代轮次到第 6 轮的时效果最好，验证集准确率为 99.14%。

(2) SGD:

Epoch/轮次	LR=1e-2/%	LR=5e-3/%	LR=1e-3/%	LR=5e-4/%
1	94.16	85.72	19.40	15.88
2	97.06	94.28	38.02	18.20
3	97.44	96.70	59.20	20.16
4	97.88	97.34	81.08	26.30
5	98.04	97.58	87.88	34.22
6	98.16	97.98	90.62	42.52
7	98.44	98.08	91.92	64.92
8	98.32	98.30	93.28	78.46
9	98.72	98.44	94.72	83.54
10	98.54	98.52	95.20	87.10
11	98.86	98.66	96.18	89.12

12	98.54	98.64	96.02	90.36
13	98.86	98.62	96.34	91.88
14	98.74	98.60	96.70	92.12
15	98.82	98.64	96.70	93.02
16	98.92	98.70	96.92	93.54
17	98.70	98.78	97.02	93.90
18	98.90	98.92	96.88	94.36
19	/	98.76	97.08	94.82

Table 2 - SGD 方法超参数学习情况

由表得，SGD 法的学习速率  $lr$  设为  $1e-2$  最合适，过大会难以收敛，过小则收敛过慢；在此学习率下，迭代轮次到第 16 轮的时效果最好，验证集准确率为 98.92%。

## Loss 可视化

对应于第 3 题第 4 问

(1) 当优化算法为 SGD（学习速率设为  $1e-2$ ），loss-epochs 曲线如下

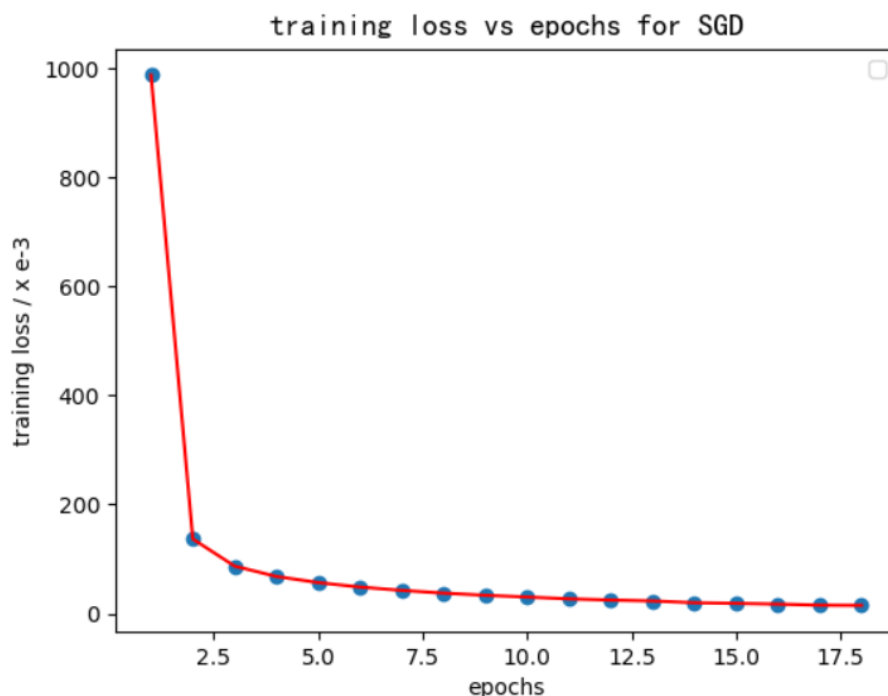


Figure 4 training loss curve for SGD

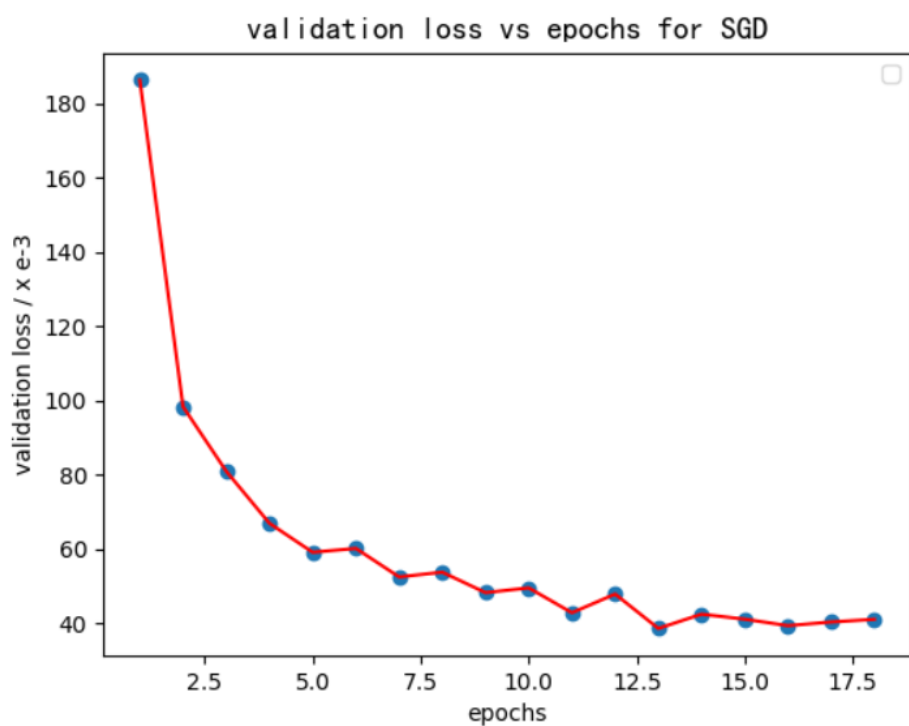


Figure 5 validation loss curve for SGD

(2) 当优化算法为 Adam（学习速率设为  $1e-3$ ），loss-epochs 曲线如下

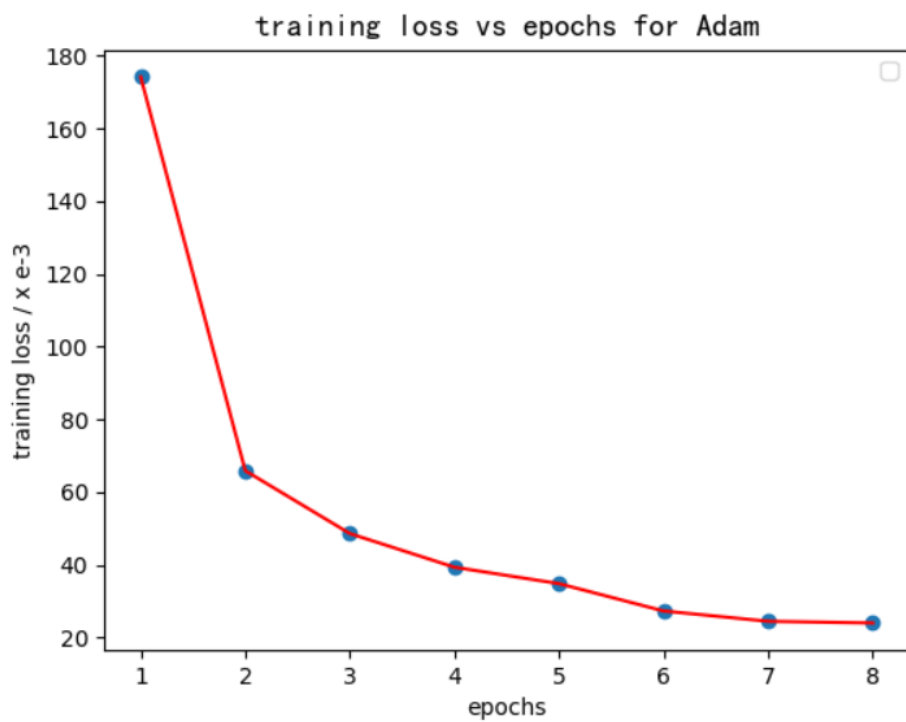


Figure 6 training loss curve for Adam

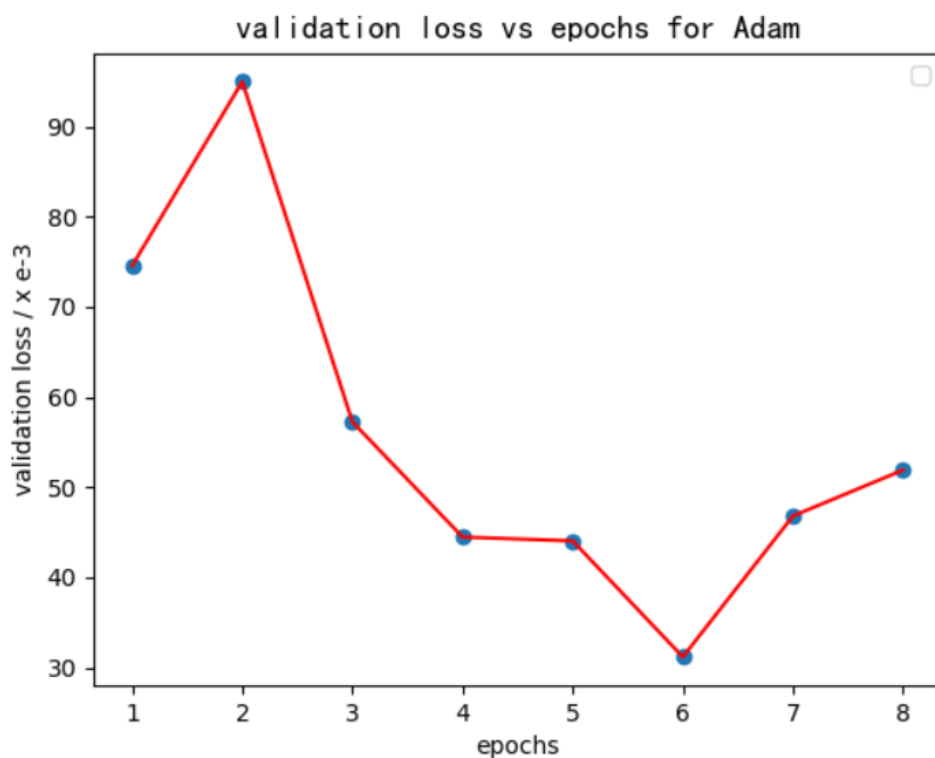


Figure 7 validation loss curve for Adam

## 参考文献

[1] pytorch 官方教程 理解了 pytorch 框架的基本使用接口和方法

[https://pytorch.org/tutorials/beginner/blitz/cifar10\\_tutorial.html](https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html)

[2] 卷积神经网络原理指南 理解了 CNN 的计算原理和基本步骤

<https://zhuanlan.zhihu.com/p/27908027>

3. pytorch 学习笔记 参考了验证集的划分方法

<https://blog.csdn.net/SHU15121856/article/details/88827238>

4. 如何理解卷积 加深了对图像处理步骤中卷积的理解

<https://www.zhihu.com/question/30888762>

5. pytorch 教程 参考了 torch.nn.Sequential 方法和 torch.nn.Dropout 方法的使用

<https://zhuanlan.zhihu.com/p/128137225>