Deep Learning

MNIST Digits Classification with CNN

Author: Yantian Luo

Institute: Electronic Engineering

ID: 2018310742

Update: April 8, 2019

Contents

Chapter 1 Introduction

MNIST digits dataset is a widely used database for image classification in machine learning field. It contains 60,000 training samples and 10,000 testing samples. Each sample is a 784×1 column vector, which is transformed from an original 28×28 pixels grayscale image.

In this homework, we will continue working on MNIST digits classification problem by utilizing convolutional neural network (CNN). The main challenge is to implement the forward and backpropagation functions of convolutional layer and pooling layer from scratch! And at last, we compare the the difference of results you obtained when working with MLP.

Chapter 2 Algorithm Design

In this homework, we have basis about the Linear layer, Relu layer, Sigmoid layer, EuclideanLoss and SoftmaxCrossEntropyLoss from homework1, thus we don't need to complete their forward and backward functions another time. Therefore, the most important challenge for us is to implement the forward and backpropagation functions of convolutional layer and pooling layer.

Chapter 3 Experients and Results

In this chapter, we use CNN to work on MNIST digits classification problem and get the accuracy and loss, and we compare the difference of results obtained when working with MLP in homework1. And at last, we visualize the first convolution layer's output after ReLU for 0-9 digit images.

3.1 Experiments1 (two conv layers + one linear layer)

The network structure in this section is designed as Figure 3.1 and training arguments is as follow:

```
config = {
  'learning_rate': 0.01,
  'weight_decay': 0.0001,
  'momentum': 0.9,
  'batch_size': 100,
  'max_epoch': 100,
  'disp_freq': 50,
}
```

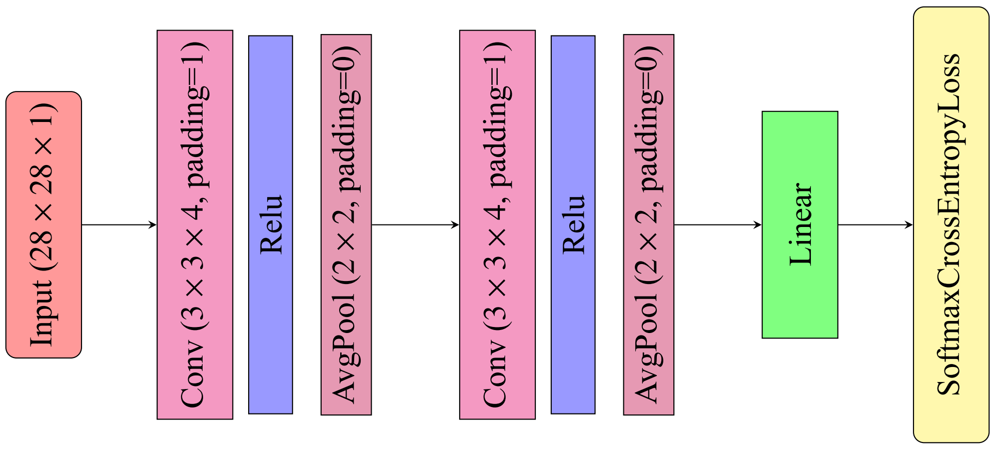
Input (28 × 28 × 1)
Conv (3 × 3 × 4, padding=1)
Relu
AvgPool (2 × 2, padding=0)
Conv (3 × 3 × 4, padding=1)
Relu
AvgPool (2 × 2, padding=0)
Linear
SoftmaxCrossEntropyLoss

Figure 3.1: Experiments1 Network Structure

In this experiments, the accuracy on test set is 0.983 and the loss on test set is 0.052. We draw the train accuracy curve and train loss curve as shown in Figure 3.2 and we draw the test accuracy curve and test loss curve with respect to epoch as shown in Figure 3.3

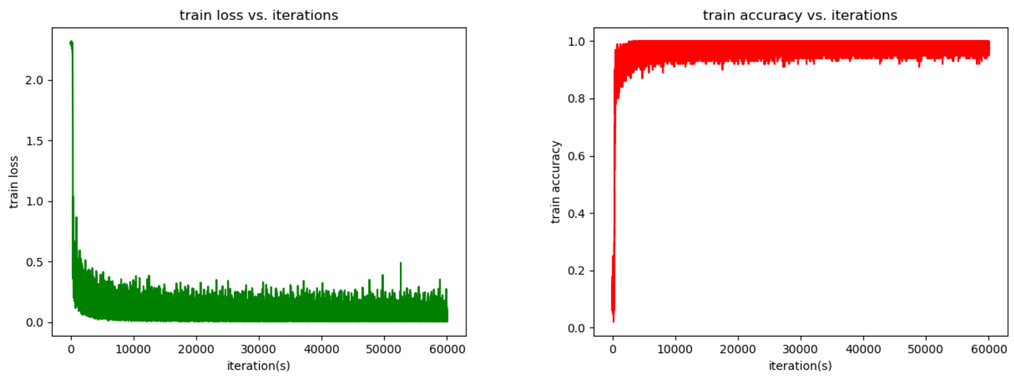

Figure 3.2: train loss curve and train accuracy curve in Experiments1

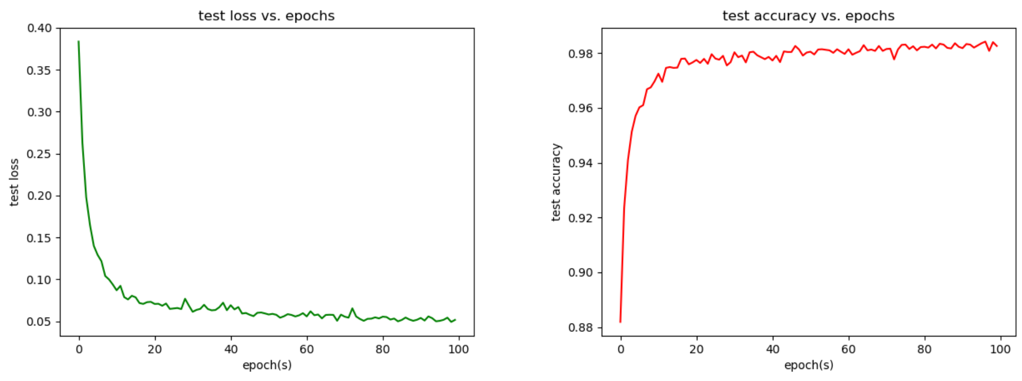

Figure 3.3: test loss curve and test accuracy curve in Experiments1

3.2 Two hidden layers experiments

The network structure in this section is designed as Table**??** and training arguments is as follow:

```
config = {
    'learning_rate': 0.1,
    'weight_decay': 0.0001,
    'momentum': 0.9,
    'batch_size': 100,
    'max_epoch': 100,
    'disp_freq': 50,
}
```

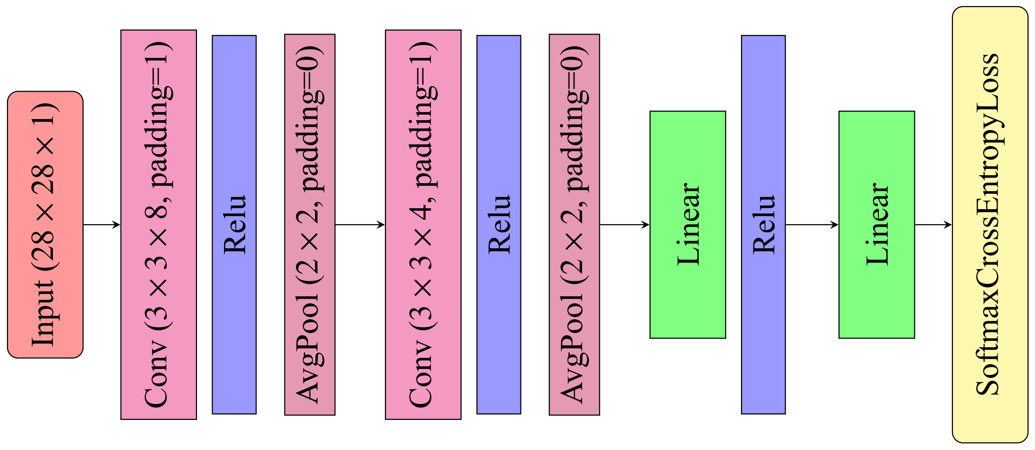
Input $(28 \times 28 \times 1)$
Conv $(3 \times 3 \times 8$, padding=1)
Relu
AvgPool $(2 \times 2$, padding=0)
Conv $(3 \times 3 \times 4$, padding=1)
Relu
AvgPool $(2 \times 2$, padding=0)
Linear
Relu
Linear
SoftmaxCrossEntropyLoss

Figure 3.4: Experiments2 Network Structure

3.3 Results

We summarize all test accuracys and test losses of every experiments as shown in Table 3.1(In this table, RE represents the experiments using Relu activation function and EuclideanLoss, SE represents the experiments using Sigmoid activation function and EuclideanLoss, RS represents the experiments using Relu activation function and SoftmaxCrossEntropyLoss, SS represents the experiments using Sigmoid activation function and SoftmaxCrossEntropyLoss). From the results, we can find the best chance is the experiments that use Relu activation function and EuclideanLoss with two hidden layers, the test accuracy is 0.983.

Table 3.1: Results Summary

Cases	RE	SE	RS	SS
Test Acc with one hidden layer	0.975	0.951	0.982	0.975
Test Loss with one hidden layer	0.039	0.062	0.062	0.083
Test Acc with two hidden layers	0.983	0.947	0.981	0.976
Test Loss with two hidden layers	0.021	0.053	0.065	0.075