Machine Learning (CSCI 6364)
Assignment 3

Dursun Dashdamirov

November 20, 2024

1 Introduction

In this assignment, we are training two neural networks to classify the MNIST digits. Concretely, each digit is a 28x28 grayscale image (so there is only one color channel so the shape of an input is just 28x28). There are 60000 training examples provided, and 10000 validation/test images.

2 A fully connected network

Since the structure for the first model is given to us with instruction, the remaining hyperparameters for the training process are the choice of the optimizer, its learning rate, and activation functions. I have used ReLu as an activation function and Stochastic Gradient Descent (SGD) as an optimizer. After receiving reasonable accuracy (90%+), I decided to focus on the learning rate and how long we should train the model. To determine the optimal number of epochs I decided to train it up to 15 epochs (which is reasonable for this relatively simple problem) and test it on training data at each epoch. This produces the following figure:

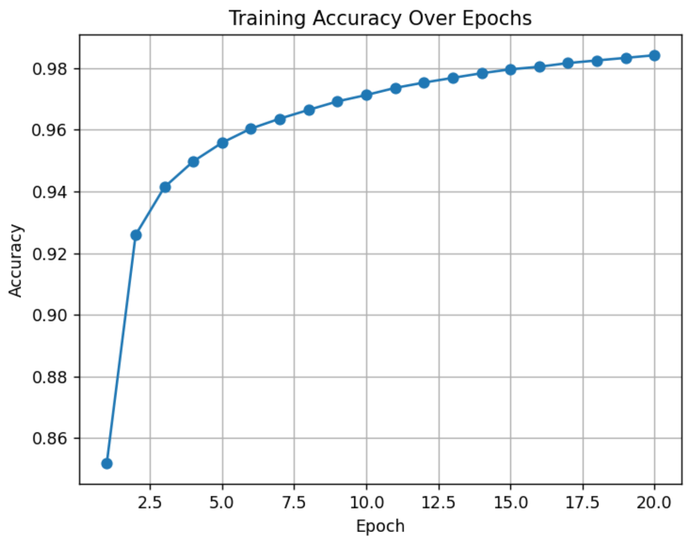


Figure 1: Model performance through over epochs. The learning rate of 1e-3 has been used.

The accuracy increases as we train more. However, this performance is on the training dataset, and after 10 epochs, we can clearly see the diminishing returns. Thus, the maximum of 15 epochs seems reasonable. This number will ensure that we are close to the optimum solution in a given learning rate (15 instead of 10 is selected to give more time to the small learning rates to converge).

Since we decided how long we should train, let's determine the optimal learning rate for the optimizer. After some trials (upper bound and lower bound being 1e-2 and 1e-4 respectively), we determine the reasonable learning rate value as 0.003.

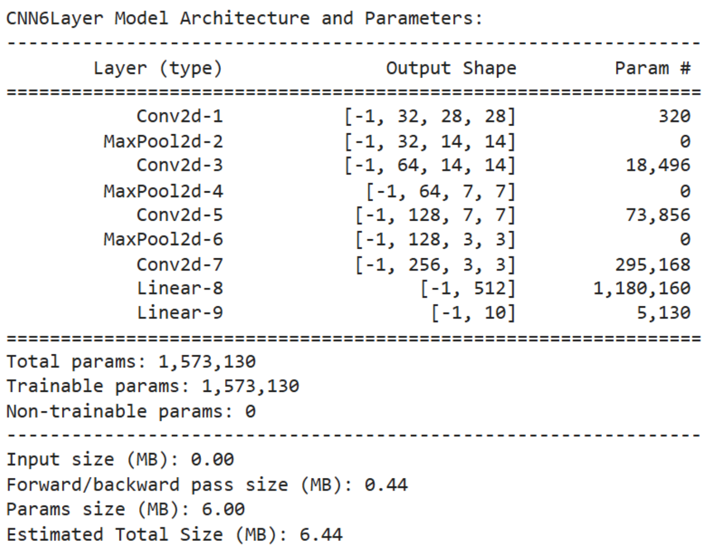
Learning Rate	**Accuracy (Test)**
1×10^{-2}	86.99%
1×10^{-3}	95.83%
3×10^{-3}	96.37%
5×10^{-3}	95.61%
1×10^{-4}	93.17%

Table 1: Accuracy results for different learning rates.

With these hyperparameters, the accuracy on training and test datasets are 98.27% and 96.37%, respectively.

3 A convolutional neural network

The main effort has been spent on defining the structure of convolutional neural network since we have approximate hyperparameter values for this problem. The model was built with four convolutional layers, starting with 32 filters and doubling up to 256, each followed by ReLU activation and max pooling to reduce spatial size. I initially experimented with a larger stride but saw overly reduced feature map dimensions, so I adjusted the stride to 1 and added padding to preserve spatial information. The fully connected layers include 512 neurons before the final layer. This architecture was chosen to capture both low-level and high-level features while keeping the model efficient. Figure 2 illustrates the summary table output from torch. summary which includes model architecture and the number of parameters.



```
CNN6Layer Model Architecture and Parameters:
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 28, 28]             320
         MaxPool2d-2           [-1, 32, 14, 14]               0
            Conv2d-3           [-1, 64, 14, 14]          18,496
         MaxPool2d-4             [-1, 64, 7, 7]               0
            Conv2d-5            [-1, 128, 7, 7]          73,856
         MaxPool2d-6            [-1, 128, 3, 3]               0
            Conv2d-7            [-1, 256, 3, 3]         295,168
            Linear-8                  [-1, 512]       1,180,160
            Linear-9                   [-1, 10]           5,130
================================================================
Total params: 1,573,130
Trainable params: 1,573,130
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.44
Params size (MB): 6.00
Estimated Total Size (MB): 6.44
```

Figure 2: Model architecture summary.

There was no need to tune the hyperparameters since the model achieved 99.07% performance on the test dataset. The training accuracy was 99.90%.

Lastly, I wanted to test if the more powerful model can perform with fewer training loops and the answer turned out to be yes:

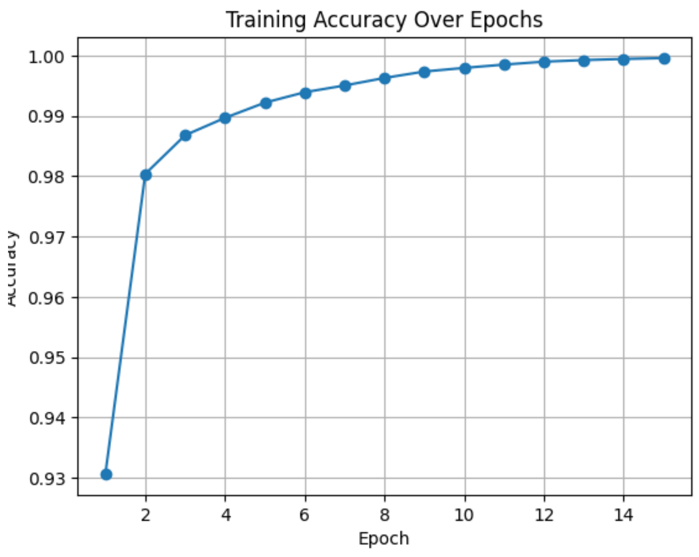


Figure 3: CNN architecture training summary. The model achieves 99% accuracy with only 4 epochs.