Paige Rosynek
CS 3450 021
Dr. Sebastian Berisha
05.11.2023

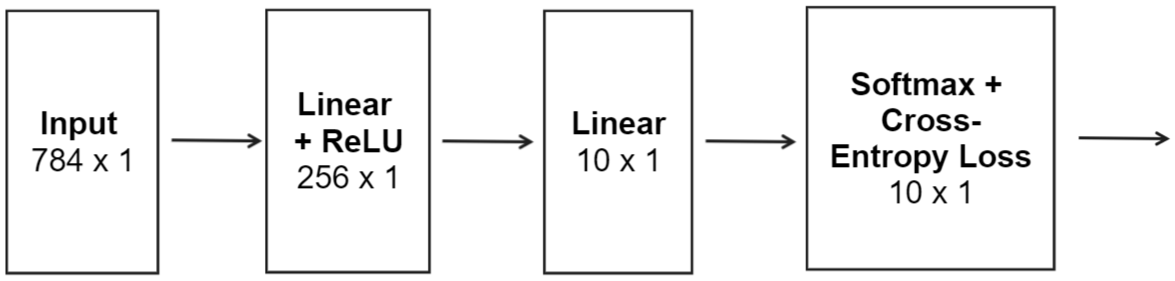
Lab 08 - Training & Testing with From-Scratch Library

Introduction

In this lab, we will be using our completed from-scratch neural network library to train a network on the Fashion-MNIST dataset. The goal of this lab is to implement a client training loop that trains and evaluates the accuracy of a neural network built using our library implementation on the given dataset. The Fashion-MNIST dataset contains 60k training samples and 10k testing samples with a total of 10 classes that each represent an item of clothing. Each sample in the dataset is a 28x28 grayscale image that is represented as a flattened, 784x1, tensor which will be the input to the network. In order to evaluate the progress of the network over the course of training, the accuracy and loss on the training and testing sets is recorded after every epoch. We then plot the testing and training curves for the network to better visualize the performance of the model and identify any potential overfitting. The target test accuracy for this dataset is 50%, however we will attempt to exceed this.

Network Architecture

The diagram below depicts the architecture of the final network, however note that the shapes of each of the layers, represents the shape of the output of that layer. The input layer consists of 784 neurons, the hidden layer consists of 256 neurons with ReLU activation, and the output layer has 10 neurons with a softmax activation.



Model Results - Fashion MNIST

The following hyperparameters were used to train this network:

Epochs = 10
Batch size = 1
Learning rate = 0.01
Regularization Coefficient = 0.001

The table below shows the performance of the model after every epoch of training. On the test set, I was able to achieve an accuracy of 85.63% and a cross-entropy loss of 14.636002540588379 and an accuracy of 86.063% and loss of 14.527243614196777 on the training set.

Training Time: 613.13 seconds

Epoch	Train Loss	Test Loss	Train Accuracy	Test Accuracy
1	5901.27392578125	3049.665283203125	75.7233%	79.9999%
2	1778.581787109375	924.3596801757812	82.5583%	82.8199%
3	542.7330322265625	286.86578369140625	84.52%	83.78%
4	172.3510284423828	95.95709228515625	85.1217%	84.25%
5	61.595375061035156	38.814510345458984	85.435%	84.86%
6	28.489294052124023	21.678647994995117	85.695%	85.43%
7	18.610748291015625	16.726118087768555	85.82%	85.08%
8	15.649323463439941	15.321357727050781	85.905%	84.6%
9	14.772022247314453	14.804988861083984	85.965%	85.27%
10	14.527243614196777	**14.636002540588379**	86.0633%	**85.63%**

Training Curves

Figure 1: Loss v. Epochs

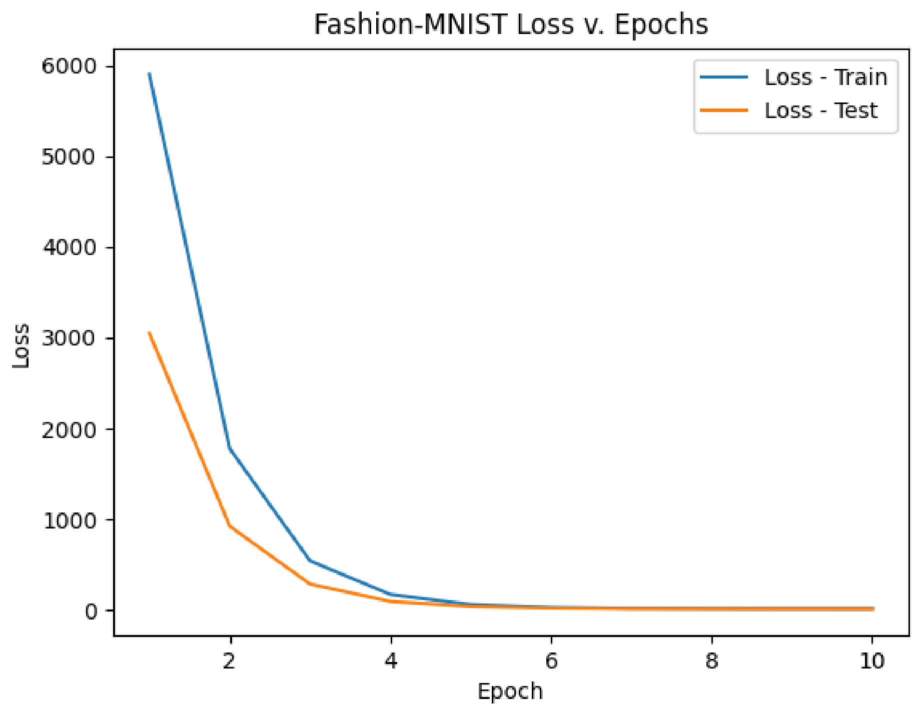
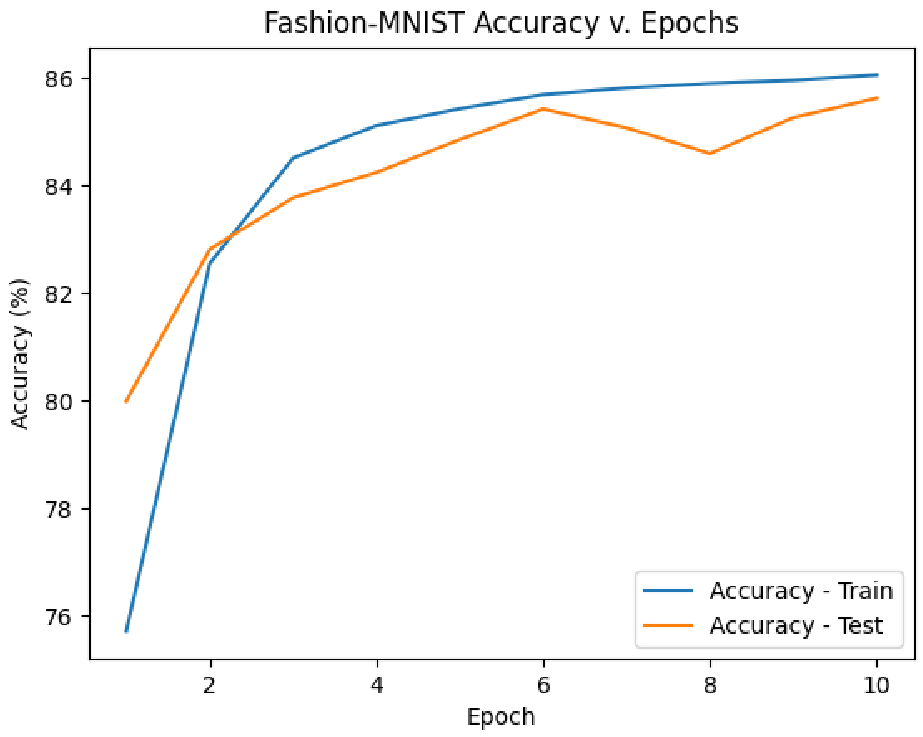


Figure 2: Accuracy v. Epochs



Reflection

From the training curves above, we can observe that the training and testing loss are very similar across later epochs. Additionally, in Figure 2 it can be seen that the training and testing accuracies start to converge around the 10th epoch which suggests that if the model was trained for more epochs the network could have achieved an even better accuracy. The observed trends in both the loss and accuracy curves indicate that the model is not drastically overfitting and is an overall decent model.

I have really enjoyed implementing the from-scratch neural network. I think that this sequence of labs was an extremely valuable exercise and it helped thoroughly solidify my understanding of neural networks in addition to the math behind how they work. When it came to deriving the equations for backpropagation, I felt that I was well prepared and that I had an adequate amount of reference material from class such that I was able to pretty confidently complete the derivations. I previously dreaded, but now appreciate how many times we did examples of backpropagation derivations in class as well as for labs. This helped me become more comfortable with the backpropagation equations and the calculus involved. Throughout the process of implementing the from-scratch library I found that writing unit tests and including assert statements in the classes made debugging the layer and network classes much less of a headache. The use of assert statements to check the shapes of layers was immensely helpful because it guided me to where the problem was occurring to serve as a starting point for debugging, instead of blindly putting print statements to find what is going wrong. I also felt that I had a good enough understanding of the math behind neural networks that I was able to identify and fix any shape issues easily. I never felt like I had no idea what my code was doing when it broke (which was refreshing) and is mostly due to the fact that I had enough knowledge of how the network was supposed to work. In conclusion, I really enjoyed this series of labs and I am glad to have this type of experience to put on my resume.