Deep Learning From Scratch Lab| Kyle Rodrigues

Introduction

In this lab we take our previous 2 iterations of our lab and take it the next step further with testing our neural net on actual datasets. We will be testing our neural net on the Fashion-MNIST and CIFAR datasets. I choose to implement the neural net shown in Lab 6, and while I explored other more complex neural nets with more layers, I found the time to actually train the neural net just took too long.

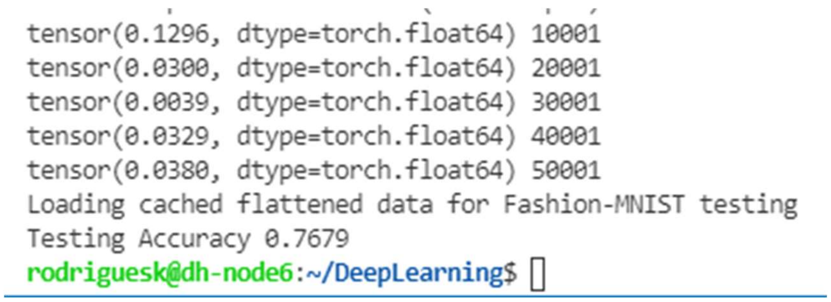
Summary Tables for Best Accuracy Achieved

	Fashion MNIST	CIFAR 10	CIFAR 100
Epoch Count	1	20	20
Step size	0.0001	0.0001	0.0001
Final Loss (Training)	0.0380	0.0172	0.3588
Accuracy (Testing)	76.79 %	43.08%	18.08%

Side note on loss:

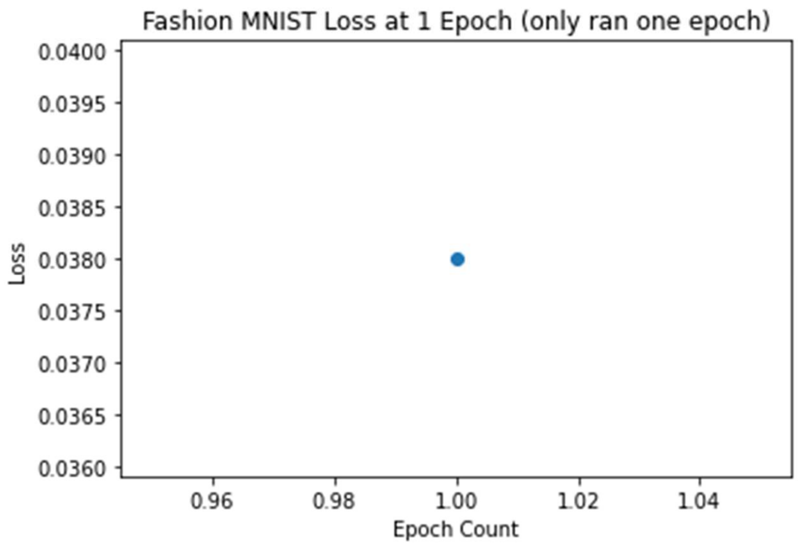
I choose to take the loss at intervals while training, which meant it would take the loss on one batch, or one image, which turned out to be a highly inaccurate way to determine loss. The biggest issue was how long it took to train the neural net, so I still used the loss intervals for my graphs and table, but in hindsight I should have tried to take the loss of the whole epoch. For the plots I wanted to standardize which loss I was using, so I only plotted the loss for the last image in the dataset (granted for the MNIST dataset, this is only one point).

Screenshots for Fashion MNIST (proof for summary table)

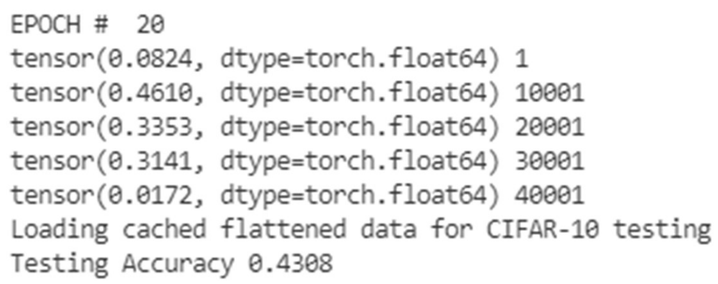


```
tensor(0.1296, dtype=torch.float64) 10001
tensor(0.0300, dtype=torch.float64) 20001
tensor(0.0039, dtype=torch.float64) 30001
tensor(0.0329, dtype=torch.float64) 40001
tensor(0.0380, dtype=torch.float64) 50001
Loading cached flattened data for Fashion-MNIST testing
Testing Accuracy 0.7679
rodriguesk@dh-node6:~/DeepLearning$
```

The loss is above the testing accuracy and is recorded every 10,000 batches. To standardize the graph, we only take loss from the last image, but since we only ran this on 1 epoch due to the high accuracy after only 1 epoch (deliverables stated we needed to get as close to 50% as possible), we will only have one point.

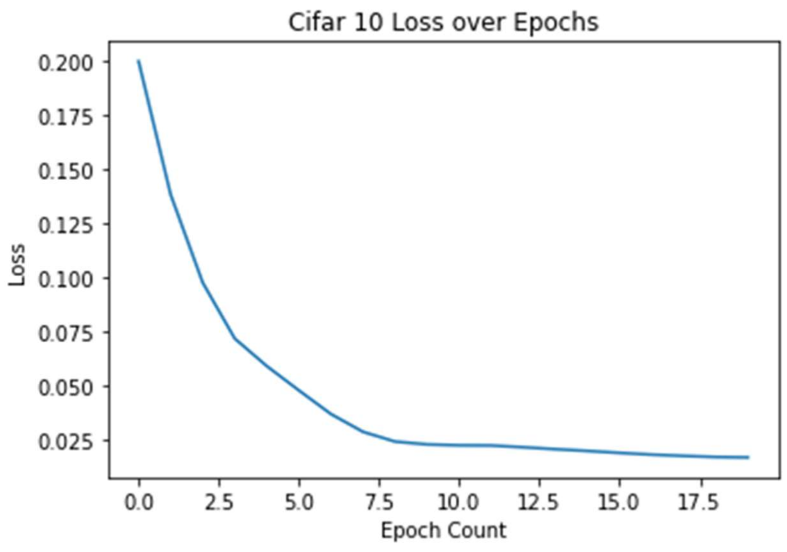


Screenshots for CIFAR 10 (proof for summary table)



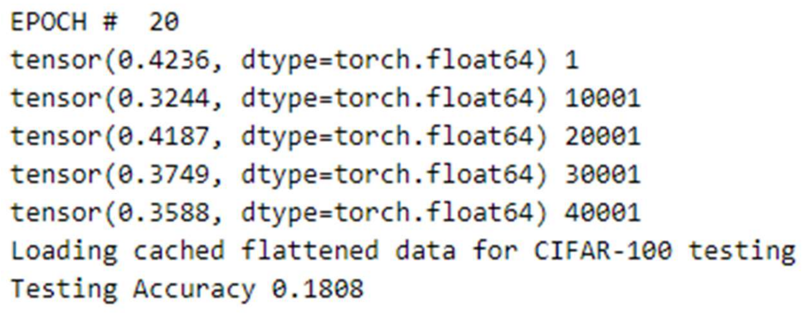
```
EPOCH #  20
tensor(0.0824, dtype=torch.float64) 1
tensor(0.4610, dtype=torch.float64) 10001
tensor(0.3353, dtype=torch.float64) 20001
tensor(0.3141, dtype=torch.float64) 30001
tensor(0.0172, dtype=torch.float64) 40001
Loading cached flattened data for CIFAR-10 testing
Testing Accuracy 0.4308
```

Once again, we plot the loss of the last image. This time we ran on multiple epochs since our accuracy was really low and we notice the loss decreasing over time. The model still probably could be ran more, but time was a big constraint for this lab.

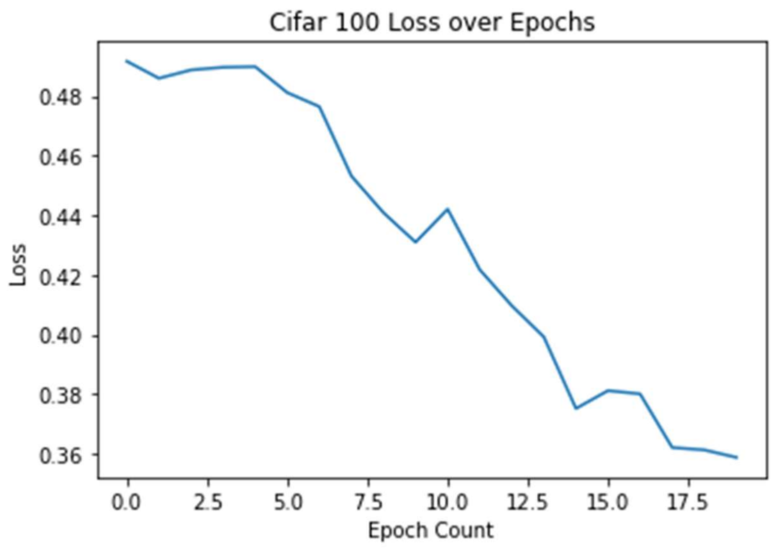


Screenshots for CIFAR 100 (proof for summary table)

This test took around 14 hours to run, and halfway through my pc shut down so I had to rerun it and save the loss/accuracy to a file.



```
EPOCH #  20
tensor(0.4236, dtype=torch.float64) 1
tensor(0.3244, dtype=torch.float64) 10001
tensor(0.4187, dtype=torch.float64) 20001
tensor(0.3749, dtype=torch.float64) 30001
tensor(0.3588, dtype=torch.float64) 40001
Loading cached flattened data for CIFAR-100 testing
Testing Accuracy 0.1808
```



It is clear that the model can do better, but unlike the CIFAR 10 dataset, the loss curve fluctuates a lot and seems to be learning at a much slower pace. The curve likely would flatten out and look better if we were to increase the epochs by at least 2 fold, but I did not have the time, nor did I want to use more resources than necessary for this.

Experience with this Lab Sequence

Overall while this lab sequence was much harder that other labs in the past, it felt enjoyable being able to create a neural net from scratch and be able to run it. Doing the backprop equations by hand made me feel like I better understood what goes on behind the scenes and allowed me to understand the math behind neural nets. Being able to create the neural net was really hard since I was not entirely sure if it was working as was expected or created as was expected, but in the end I was able to create an implementation that worked under my own understanding of neural nets and I am pretty proud of it.

Surprisingly, the debugging of the software with the unit tests was more fun than I expected it to be. Not only was I able to ensure the software was running at my own implementation, I better understood debugging in python which I haven't done much off in the past. The training curves and final accuracy kinda showed that my network wasn't that good, but I believe it is just that we are using our network on images, which seems less effective than other networks like a CNN. Additionally, the loss I was reporting was from a single image over time and may not have been the best indicator of loss. Despite this, I still think the network did pretty well with how simple it was in comparison to the pre-built networks from Scikit-learn.

For this lab, I tried many different iterations of my network parameters to try and get better accuracy, and while I believe I could have still got better accuracy, I felt that the time/benefit ratio to achieve better accuracy was not worth it since I already had played with the parameters a lot and when my code takes upwards of 12-14 hours to run (on the CIFAR 100 dataset), I just couldn't justify playing with my network more. Despite that, I felt that I better understood how difficult it is to tune the hyperparameters and can understand how affecting certain parameters can change the loss/accuracy based on how the network is setup behind the scenes. I also found that at the start of this lab, many of my backprop equations were incorrect and I hadn't created a Regularization layer, so I had to go back and make sure everything was correct multiple times, until I was satisfied with my equations. This means that my accuracy should be correct because I believe that I created my network correctly, and the math behind it seems fine to me and the reason why I did not hit 50 % accuracy for CIFAR was not due to my math being wrong, but just not having the time to make a more complicated network or to test more iterations of different hyperparameters.