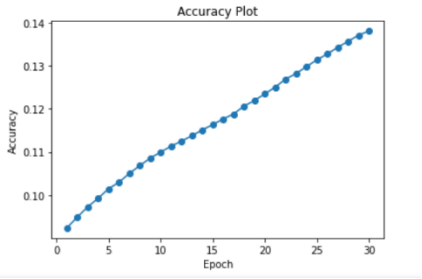
Tyler Faulkner
CS3450
Lab 4
4/4/2022

Attempt	Epochs	Learning Rate	Regularization Rate	Batch Size	Structure	GPU Memory Estimate	True GPU Memory	Time to Train Estimate	True Time to Train	Final Training Accuracy	Final Test Accuracy
1	30	0.0001	0.0	64	Fully Connected [784, 28, 14, 10, 10]	352 KiB	1350 MiB	100 seconds	81.59s	0.1381	0.1418
2	90	0.0001	0.0	64	Fully Connected [784, 28, 14, 10, 10	1350 MiB	1350 MiB	240 seconds	241.134s	0.3535	0.353
3	90	0.0001	0.0001	64	Fully Connected [784, 28, 14, 10, 10	1350 MiB	1350 MiB	241 seconds	243.03s	0.221	0.220
4	180	0.0001	0.0001	64	Fully Connected [784, 28, 14, 10, 10	1350 MiB	1350 MiB	483 seconds	492.38s	0.6103	0.6075
5	240	0.0001	0.0001	64	Fully Connected [784, 28, 14, 10, 10	1350 MiB	1352 MiB	640 seconds	645.77	0.5301	0.5263
6	240	0.0001	0.0001	32	Fully Connected [784, 28, 14, 10, 10	1350 MiB	1352 MiB	1280 seconds	1275.18	0.7634	0.7594
7	300	0.0001	0.0001	32	Fully Connected [784, 28, 14, 10, 10	1350 MiB	1352 MiB	2000 seconds	1623.56	0.7685	0.7589
8	150	0.001	0.0001	256	Fully Connected [784, 28, 14, 10, 10	1350 MiB	1352 MiB	150 seconds	99.2 seconds	0.71845	0.7159
9	150	0.001	0.0001	256	Fully Connected [784, 14, 7, 10]	800 MiB	1356 MiB	100 seconds	94.5 s	0.74668	0.7317
10	450	0.01	0.0001	1024	Fully Connected [784, 14, 7, 10]	1356 MiB	1356 MiB	240 second	73.08s	0.8336	0.8201

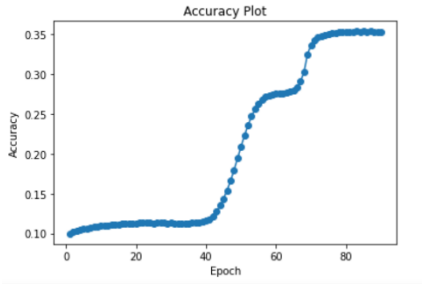
Attempt 1



The motivation for the structure of the network was a funnel design, I added a second 10 neuron hidden layer to maybe solidify the outputs. The chosen hyper-parameters were mostly just intuition based off what felt right and what I have seen in other models.

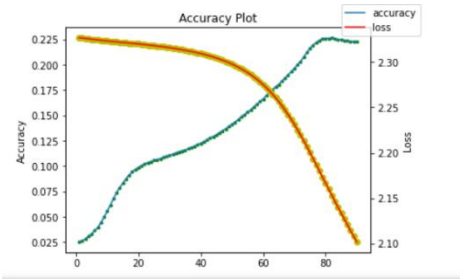
Tyler Faulkner
CS3450
Lab 4
4/4/2022

Attempt 2:



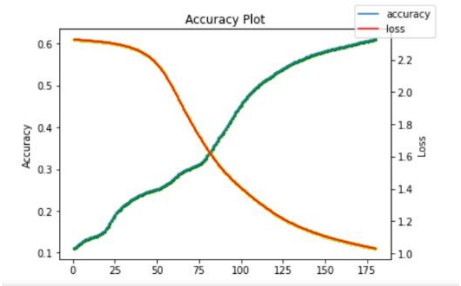
The reason I change the epochs in the second attempt was since the first attempt was obviously increasing in accuracy, but never plateaued. I expect to see an increase in accuracy simply by running it longer.

Attempt 3:



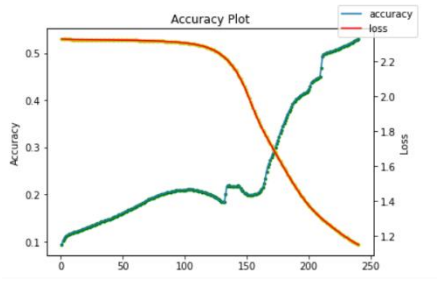
After viewing the interesting accuracy plot from the previous run (which I believe are caused by saddle points), I added a regularization factor that matched the learning rate to hopefully avoid those saddle points. The regularization coefficient should factor in the magnitude of the weights to avoid very large weights. I also decided to display the loss line now.

Attempt 4:



Attempt 3's plot showed a much smoother accuracy curve with the regularization coefficient; however, the accuracy was about 12% less. I decided to double the epochs which should hopefully achieve a better accuracy.

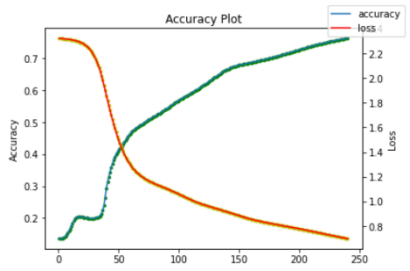
Attempt 5:



The last attempt reached over 50% accuracy for the check off for the lab, and the Training accuracy seems to by plateauing, so I am going to increase the epochs by 60 this run to verify that the training is converging.

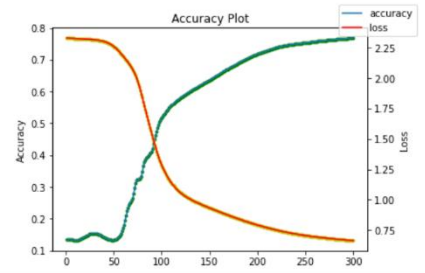
Tyler Faulkner
CS3450
Lab 4
4/4/2022

Attempt 6:



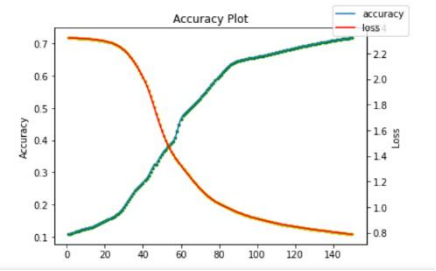
The last attempt had a large jump at the end and narrowly got above 50% accuracy at the end. I believe the network hit some of the saddle points as mentioned early and got lucky in the last run and avoided them. For this run, I am going to halve the batch size in an attempt to get around those saddle points.

Attempt 7:



The decrease in batch size improved on creating a smoother increase in accuracy, but I believe if I run the model on more epochs, I can squeeze at least a few more percentage points before the graph begins to plateau. I will increase the epochs to 300 for this experiment, and I will likely cap it at this amount.

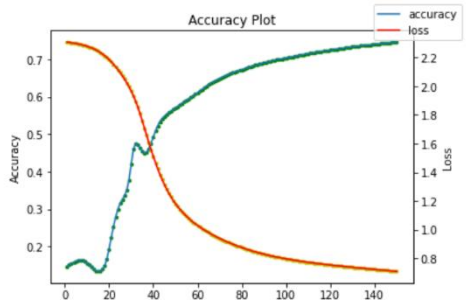
Attempt 8:



As I expected, I was only able to get a few more percentage points of accuracy. I was reading this article which said that increasing the batch size and the learning rate together should provide similar results to a lower learning rate and smaller batch size So, I am going to increase the batch size to 256 and increase the learning rate to 0.001. I am also going to halve the epochs to see the effect more quickly it has.

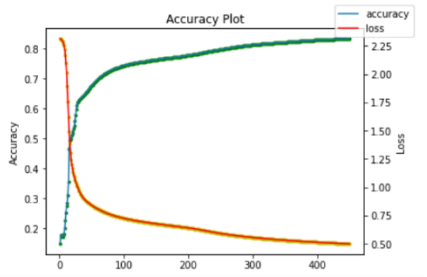
Tyler Faulkner
CS3450
Lab 4
4/4/2022

Attempt 9:



To my surprise the last attempt proved very successful. I achieved a very similar accuracy to my very low batch rate in a fraction of the time. I now wonder if I can achieve similar results in a smaller Network. I am going to change my network to 2 hidden layers with 14 neurons and 7 neurons respectively. This should also help reduce saddle points since there are less parameters making the loss function less complex.

Attempt 10:



The last attempt got even better results than the large network, so I will keep using this network. For my final attempt I am going to increase the learning rate and batch size again and triple the epochs. I want to reach the limits of my model.

Reflection:

I was surprised at how well my final model did. I started to plateau around epoch 100 but achieve and over 80% accuracy. It was also interesting to see how increasing the batch size and learning rate together improved training time immensely and outputted similar accuracy values. I think my original system was suffering from the complexity of the system. With so many parameters there was bound to be many saddle points when converging on the loss that it would get stuck on. The smaller system at the end had the benefit of a much simpler design. One last thing I want to note is the GPU memory. My initial prediction was based on just the design of my system, but I believe I forgot to take into account the other variables in my notebook (like the training and testing data).