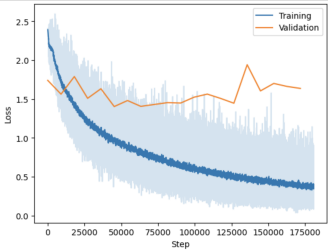
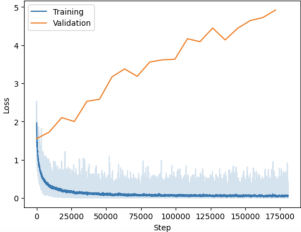
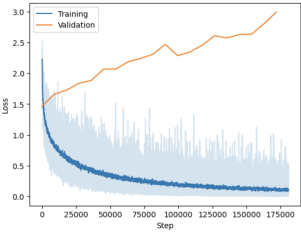
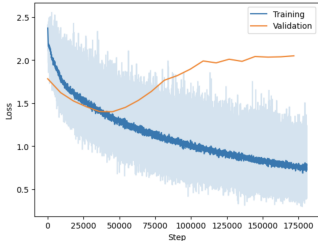
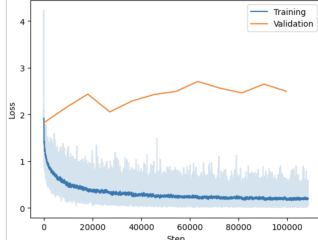
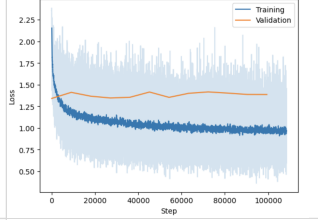
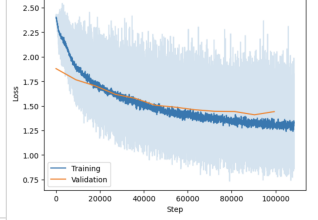
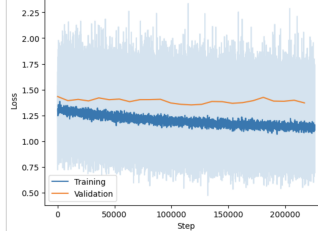


Model	Best Model Validation Loss	Best Model Validation Accuracy	Best Model Epoch #	Test Loss	Test Accuracy	Graph of Losses	Notes
default model	1.403620551	63.80%	5	1.440197341	63.67%		The validation accuracy here isn't particularly inspiring and is pretty much flat in the 60-ish% range for the entire 20 epochs of training
Adam instead of SGD optimizer, lr=1e-3	1.551941952	67.06%	0	1.533584043	67.80%		Seems to have exploding validation loss. The first epoch had the lowest validation loss but not the highest validation accuracy, which hovered around 70-73% for most epochs.
Adam optimizer with decreasing learning rate, from 1e-1 to 1e-4, using lr_scheduler							Stopped training because, after 8 epochs, the validation loss hovered almost exactly around 2.17 and the validation accuracy hovered almost exactly around 20%. Trying something different with the learning rate than starting at 0.1. Maybe I also used lr_scheduler incorrectly?
Adam optimizer, lr=1e-4	1.46805188	63.73%	0	1.462113539	64.58%		The validation loss still seems to be exploding, but the validation accuracy is increasing slightly over the course of training epochs.
dropout of 0.2 at init, batch size 16, Adam optimizer, lr=1e-3							Stopped training because the validation loss continued to explode after 12 epochs.
tanh activation, SGD optimizer, lr=1e-3, two dropouts, batch size 32	1.400650253	50.89%	5	1.391292155	50.90%		The loss looks lower in this model, but the training loss has a much wider range throughout the training epochs, and the validation and test accuracy are actually lower. Since tanh returns values both above and below 0, I wonder if that's a problem. I'll switch once more to sigmoid and see what happens.

Model	Best Model Validation Loss	Best Model Validation Accuracy	Best Model Epoch #	Test Loss	Test Accuracy	Graph of Losses	Notes
sigmoid activation, Adam optimizer with L2 regularization							Exploding validation loss still! The training loss is getting very low very quickly with the L2 regularization, but not helping with validation. Cutting off training.
back to relu, default model but with L2							The training loss is dropping way, way faster than the validation loss. It's overfitting (?) almost immediately. I'm torn between simplifying the number of layers in the model and trying a larger stride.
stride=4							Training loss dropping even faster in relation to validation loss, which is increasing after 7 epochs.
shallower network with 2 conv layers, dropout at init	1.821893176	59.07%	0	1.880837428	59.72%		The validation loss at least explodes less?
shallow 2-conv layer network, no dropout, fewer output channels per conv layer, max pooling (4,4)	1.340920331	56.95%	0	1.368010471	56.52%		The validation loss and the training loss are much closer now, though the validation loss is pretty noisy
same as previous, but SGD optimizer	1.409422438	51.59%	10	1.412617266	52.29%		The validation loss and the training loss look much more similar through the first part of training! And the test loss is similar. The validation loss is in a downward trend and the validation accuracy is in an upward trend. This seems very promising. I'm going to re-run this but for 25 epochs instead of 12.
same as previous, 25 epochs	1.352058532	54.57%	13	1.356469091	54.93%		

The previous two pages are a chronological spreadsheet of model training, with each row representing a separate trained model.

The Model column lists the changes made to the model from the previous attempt. The following five columns list the validation and test performance of each trained model. For those models trained to completion, I provide a plot of the training vs. validation loss across the epochs of training. The final Notes column contains my assessment of what I think might be happening.

### Summary

- 1) For the first five training runs, I play with the optimizer (SGD vs. Adam), learning rate (static vs. decreasing), batch size, and initial dropout. These runs mostly have exploding validation loss (that's what I'm calling it), though they also have the highest validation and test accuracy of any of the training runs. Sadly, in most of these cases, the best model is the initial model, so this is mostly due to luck. My highest overall test accuracy is 67.80%, with the model being identical with the default model except for an Adam optimizer rather than SGD.
- 2) For the following four training runs, I try tanh and sigmoid activation functions, add and remove more dropouts, and introduce L2 regularization as well as increasing stride size. I cut off three of these runs before they finished because the validation loss was increasing so dramatically with each epoch. The tanh activation function had the worst overall validation and test accuracy of any model I tried, at 50.89% and 50.90% respectively, which could be because tanh returns values between -1 and 1 instead of sigmoid and relu's 0-1 range.
- 3) For the final four training runs, I shrank the size of the model to two convolutional layers with more downsampling in the pooling layers. In these runs, the validation loss corresponded more directly with the training loss and test loss (and accuracy) throughout the training process, which I thought was a good sign. If I were to keep tweaking this model, I'd continue working with a model with fewer layers.

The final and longest training run had 25 epochs with a SGD optimizer, a static learning rate of 1e-3, L2 regularization, and a 54.93% accuracy, which isn't good and seemed to hold steady throughout the training process. The previous run had used exactly the same structure and seemed to have steadily decreasing losses. In the case of this assignment, I felt like I should pursue models where the training, validation, and test accuracy follow expected patterns and remain consistent with each other throughout the training process, since this might indicate a more stable training that would lead to more predictable results.