CS 525: Deep Neural Networks
Homework 6

1 Final Model Architecture

Our final model has an input layer, three hidden layers (each with 200 hidden units), and an output layer. Each hidden layer uses ReLU as its activation function, while the output layer uses Softmax.

We used a Xavier Initialization on all of the weight vectors. We initialized all of the bias vectors to the correct shape, and filled them with the constant value 0.01.

We used the Adam Optimizer, and passed an initial learning rate of 0.001, which Adam decayed using the default beta values.

We chose to fix minibatch size to 128.

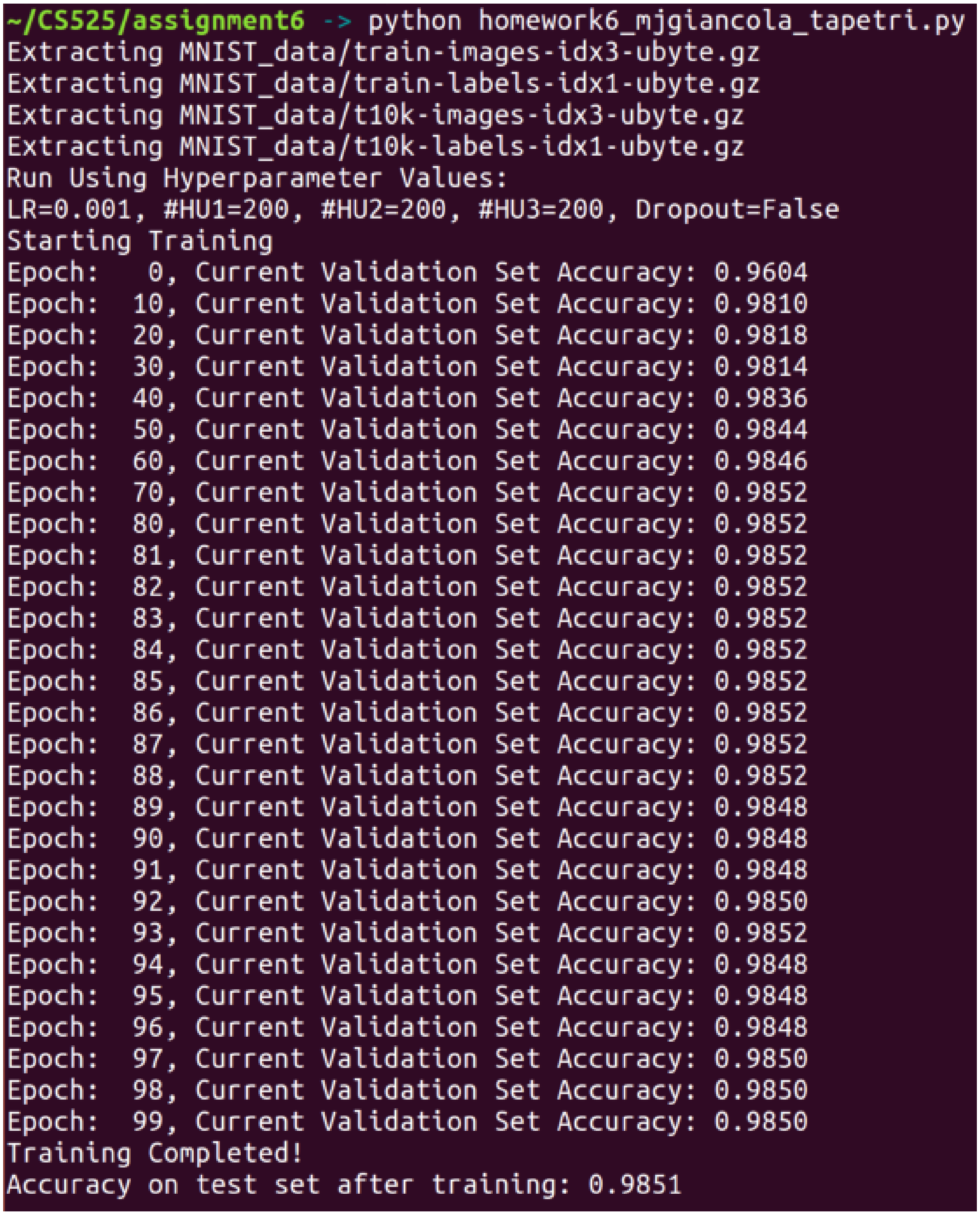
2 Hyperparameter Testing

Originally, much of our efforts went into trying to optimize a network consisting of two hidden layers and somewhere between 60-20 units per layer. We experimented with the vanilla SGD optimizer, as well as the Adam and Momentum Optimizers. Each were found to have their own sets of learning rates and momentum that worked well, and going past 100 epochs generally did not improve the models. However, with these networks, we were not able get past 95 % accuracy on the validation data.

We then decided to increase the total number of units in the network by adding another hidden layer (for a total of three) and increase the size of each layer to 200 units. We immediately saw a boost in accuracy and the network could be trained to attain over 98 % in very few epochs.

In the first case of the smaller network, dropout always led to worse results. The networks seemed to be regularized too heavily. In the case of the bigger, more complex network we saw similar results with and without dropout.

3 Screenshots



```
~/CS525/assignment6 -> python homework6_mjgiancola_tapetri.py
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
Run Using Hyperparameter Values:
LR=0.001, #HU1=200, #HU2=200, #HU3=200, Dropout=False
Starting Training
Epoch:   0, Current Validation Set Accuracy: 0.9604
Epoch:  10, Current Validation Set Accuracy: 0.9810
Epoch:  20, Current Validation Set Accuracy: 0.9818
Epoch:  30, Current Validation Set Accuracy: 0.9814
Epoch:  40, Current Validation Set Accuracy: 0.9836
Epoch:  50, Current Validation Set Accuracy: 0.9844
Epoch:  60, Current Validation Set Accuracy: 0.9846
Epoch:  70, Current Validation Set Accuracy: 0.9852
Epoch:  80, Current Validation Set Accuracy: 0.9852
Epoch:  81, Current Validation Set Accuracy: 0.9852
Epoch:  82, Current Validation Set Accuracy: 0.9852
Epoch:  83, Current Validation Set Accuracy: 0.9852
Epoch:  84, Current Validation Set Accuracy: 0.9852
Epoch:  85, Current Validation Set Accuracy: 0.9852
Epoch:  86, Current Validation Set Accuracy: 0.9852
Epoch:  87, Current Validation Set Accuracy: 0.9852
Epoch:  88, Current Validation Set Accuracy: 0.9852
Epoch:  89, Current Validation Set Accuracy: 0.9848
Epoch:  90, Current Validation Set Accuracy: 0.9848
Epoch:  91, Current Validation Set Accuracy: 0.9848
Epoch:  92, Current Validation Set Accuracy: 0.9850
Epoch:  93, Current Validation Set Accuracy: 0.9852
Epoch:  94, Current Validation Set Accuracy: 0.9848
Epoch:  95, Current Validation Set Accuracy: 0.9848
Epoch:  96, Current Validation Set Accuracy: 0.9848
Epoch:  97, Current Validation Set Accuracy: 0.9850
Epoch:  98, Current Validation Set Accuracy: 0.9850
Epoch:  99, Current Validation Set Accuracy: 0.9850
Training Completed!
Accuracy on test set after training: 0.9851
```

Figure 1: Training on Optimized Hyperparameters + Final Test Set Accuracy

Note: We noticed that the accuracy on the validation set seemed to asymptote after 70 epochs, so perhaps 100 was unnecessary. However, as seen, the model did not start to overfit with the additional epochs.