AI 535 HW2

Matthew Pacey

February 2024

1 Implementation

The code is in FeedForward.py. It implements SigmoidCrossEntropy, ReLU, and LinearLayer.

1.1 Hyperarameter Testing Results

This section contains plots from empirical results by tuning different hyperparameters. In each case, one parameter was varied while all other parameters remained constant. Each point on the graph represents the highest testing accuracy that a model could attain over 10 epochs (this value is when testing accuracy leveled off across configurations). Each configuration was run for 10 trials and the average accuracy (comparison of predicted class versus labeled class) is displayed.

1.2 Batch Size

In Figure 1, the batch size (number of training examples for each training step) is varied across: $16, 32, 64, 128, 256, 512, 1024$.

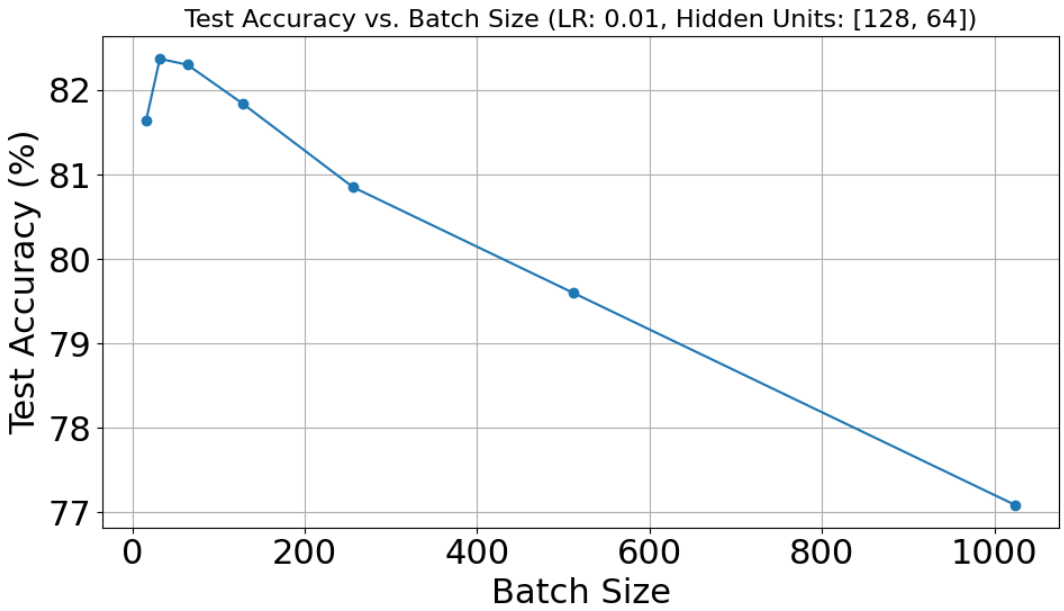

Figure 1: Comparison of Batch Sizes

1.3 Learning Rate

In Figure 2, the learning rate (the factor that determines how fast/slow the weights and biases are updated at each step) is varied across $0.0001, 0.001, 0.01, 0.1, 0.5$.

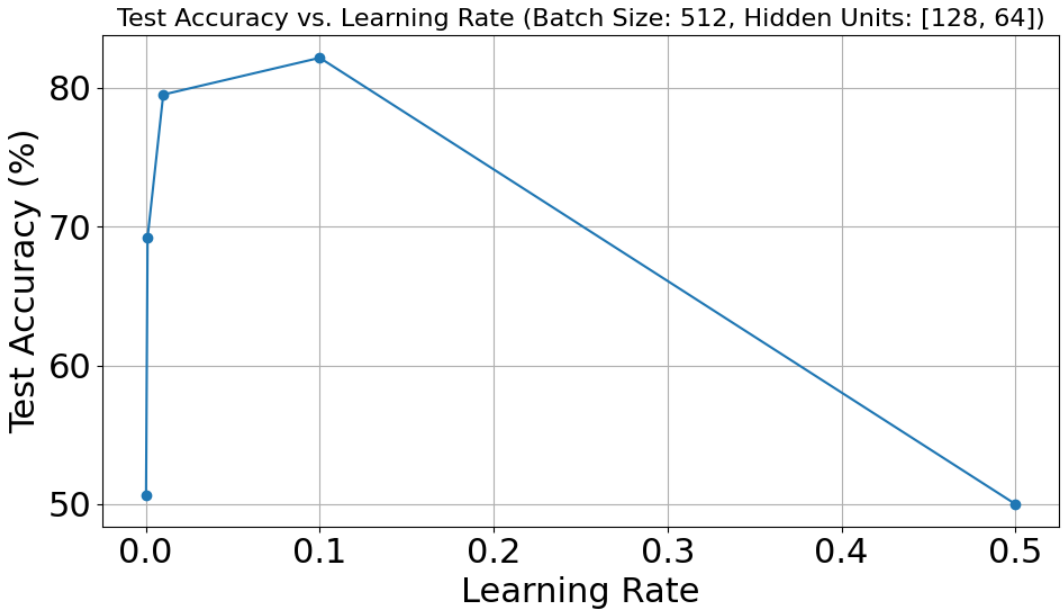

Figure 2: Comparison of Learning Rates

1.4 Hidden Units

In Figure 3, the configuration of the hidden units (layers between input and output) is varied. We ran the configuration below. Each row represents a separate configuration and each item in the configuration represents a layer's output size.

- 64
- 128
- 128, 64
- 512, 64
- 1024, 256
- 256, 128, 64
- 1024, 128, 64
- 512, 256, 128, 64,
- 1024, 512, 256, 128, 64

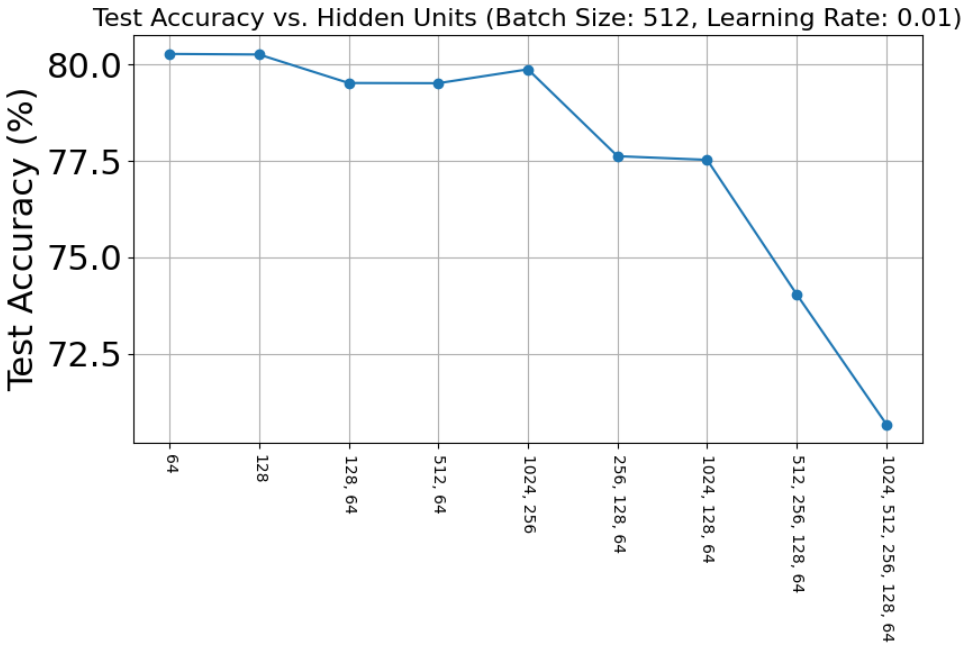

Figure 3: Comparison of Hidden Units

2 Discussion

As shown in the previous figures, the ideal hyperparameters for this problem setup are a learning rate of 0.1, batch size of 32, two hidden layers.

The smaller batch size of 32 demonstrates the effectiveness of the backpropagation update of the model parameters since there are more batches that each training epoch tests on compared to higher batch sizes (more 'steps' and therefore more updates). This could possibly be countered by training for more epochs, but the smaller batch size demonstrates its efficiency.

The ideal learning rate for this problem was 0.1, because this struck the best balance of updating the parameters just enough but not too much at each step.

The tests across the various hidden layers was the most interesting. It showed that only one or two layers was needed (both had similar performance). It was interesting to see performance drop off significantly (by about ten percent or more) when the model became more complex with three and four layers.

Each of the variables were tested in isolation, we could likely find a better combination of hyperparameters by doing a grid search across all combinations. Another way to increase accuracy would be to train more epochs and to have a cutoff to stop training once the validation loss bottoms out and starts increasing.