Project Module 2 (ECE 246)

Connor Roberts, Johannes Lee, Michael Kleinman

March 2021

1 Centralized Neural Network

For reference, we first run `centralized.py` using the given hyperparameters (20 hidden units, batch size 10, learning rate 0.2, decay rate 0.99) and show the results:

Epoch	Train Loss	Train Acc	Test Loss	Test Acc	Learning rate
1	0.21	0.9358	0.23	0.9325	0.20000
2	0.15	0.9529	0.18	0.9456	0.20000
3	0.13	0.9634	0.16	0.9553	0.20000
4	0.13	0.9604	0.17	0.9504	0.20000
5	0.10	0.9694	0.15	0.9595	0.19800
6	0.13	0.9608	0.19	0.9468	0.19800
7	0.11	0.9655	0.17	0.9549	0.19602
8	0.10	0.9700	0.17	0.9576	0.19602
9	0.09	0.9739	0.16	0.9617	0.19602
10	0.08	0.9739	0.17	0.9576	0.19602

The first hyperparameter we adjusted was the number of hidden units. The below table shows the test loss relative to the number of hidden units used.

Number of Hidden Units	Test Loss	Train Loss
30	0.17	0.08
60	0.15	0.04
100	0.14	0.04
200	0.09	0.01

As we increase the number of hidden units the train and test loss decrease. While this does improve the accuracy of the model, as the model complexity increases so does the time required to train. Additionally, we see decreasing returns of the test loss when increasing the number of hidden units. Thus, the trade-off between accuracy and processing time should be considered.

The next hyperparameter we altered was the batch size:

Batch Size	Test Loss	Train Loss
10	0.17	0.08
20	0.13	0.07
40	0.16	0.10

When altering the batch size we see an initial decrease in test loss and train loss followed by an increase as we increase the batch size further.

Learning Rate	Test Loss	Train Loss
0.1	0.20	0.13
0.2	0.17	0.08
0.4	0.36	0.27

When looking at changing the learning rate we observe a similar scenario to the batch size. Our best performance comes from the original value of 0.2. If we increase or decrease the learning rate we observe negative effects on our train and test loss.

Decay Rate	Test Loss	Train Loss
0.85	0.17	0.09
0.95	0.21	0.14
0.99	0.17	0.08

The last hyperparameter we altered was the decay rate. Interestingly, we saw that decay rates of .99 and .85 were extremely similar in their performance, however, a rate of 0.95 performed worse than both even though it is in between .85 and .99.

Taking the best performing hyperparameters of our testing and combining them into a final model gives a train loss of 0.00 and a test loss of 0.07, which is the best performing thus far. The parameters used in this model were:

Number of Hidden Nodes	Batch Size	Learning Rate	Decay Rate
200	20	0.20	0.99

The limiting factor, however, is the processing time which is significantly longer due to our increased number of nodes. In conclusion, analysis should be done when deciding hyperparameters for a specific use case to balance accuracy and the time required to train the model.

2 Federated Neural Network

We next implemented the distributed neural network training (see attached code in `distributed-sol.py` . We wrote a custom script to sweep over the relevant hyperparameter values in `run-sweeps.sh`. Specifically we ensure that $B * N = 20$, where B refers the "batch size" and N refers to the number of

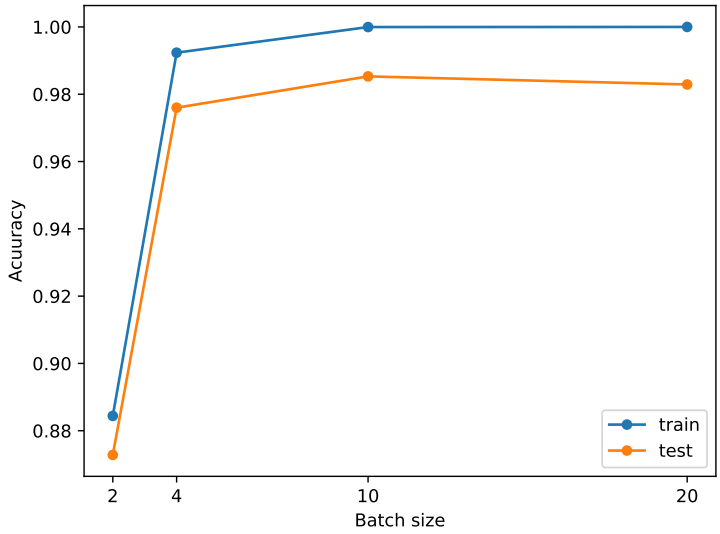

Figure 1: Train and test accuracy for different batch sizes (and corresponding number or workers). $B * N = 20$ for all the sweeps. We find that a batch size of 10, and 2 workers seems to do best, though similar to batch sizes of 4 and 20. When the batch size becomes small (2) and the number of workers becomes large(10) we observe a noticeable dip in the training and test accuracy.

workers. We sweep B belonging to the range 2, 4, 10, 20. Since the ratio is constant, this means that on the sweeps the corresponding value of N is 10, 5, 2, 1, respectively. In Fig 1 we plot the corresponding training and testing loss, averaging the training and test loss over the last 3 of the 20 epochs. We find that a batch size of 10, and 2 workers seems to do best, though similar to batch sizes of 4 and 20. When the batch size becomes small (2) and the number of workers becomes large (10) we observe a noticeable dip in the training and test accuracy. We conclude that a small batch size impairs training. Interesting, we achieve close results to single-worker performance while using 5 workers and a batch size of 4, suggesting that with appropriate batch size and number of workers, we can train a model in a federated manner.