Project 2 Part 2

Jason Van Tuinen, Bharadwaj Satyanarayana, Anthony Sainez, Karthik Gnanakumar

1 Individual Contributions

Each team member contributed to doing an initial hyperparameter search for the reference network, and in running different compression schemes on the resulting network. For the initial hyperparameter search, we each used a different batch size and explored other parameters keeping that one constant. For the compression section, we split up the compression schemes and each ran our individual sets on our own, and reported back to the group our results. Jason Bharadwaj prepared & proof read the report.

2 References

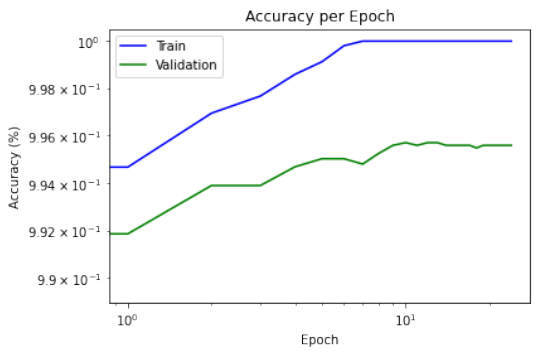
We received help from Yerlan Idelbayev for calculating the number of parameters in a model and choosing good LC algorithm starting parameters.

3 Reference Network

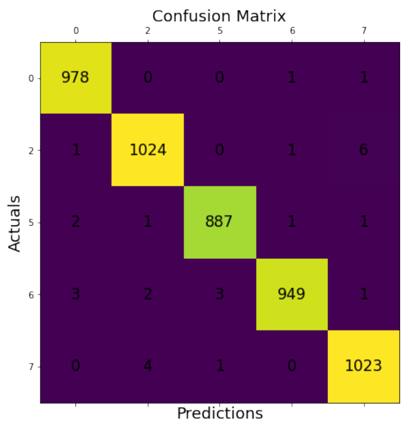
In the hyperparameter search, we noticed in part 1 that different batch sizes could have a significant impact on the resulting error, and that the best selection of other parameters tended to vary the most with batch size. For this reason, we started by selecting a couple of batch sizes for each of us to find the best models we could on, and went with the best model that any one of us came up with. The model that we trained with the lowest validation error had a batch size of 16, learning rate of .005, gamma of .95 with a step size of 1, and a momentum of 0.9 with nesterov activated. The resulting model had a test error of 0.59%. The training and validation accuracy per epoch graph is shown in Figure 1a, and the confusion matrix for the resulting model is shown in Figure 1b.

4 Compression

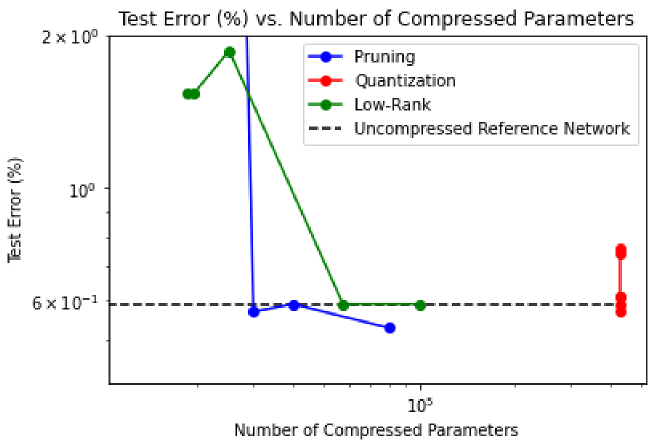
When we started out on compression, we were having a lot of trouble getting any parameter selections for the L-step and μ schedule to produce decent results. With help from the TA, we were able to figure out that we were miscalculating our number of parameters and applying an extreme compression on the model. Once we realized this, we were able to quickly pin down L-step parameters and a μ schedule that worked on all three compression types. For the L-step, we used an initial learning rate of 0.2, and a gamma of .98. Through our analysis, we also found 10 L-step epochs per iteration to be the best. For the μ schedule, we used an initial μ of $5 * 10^{-5}$ with an increase rate of 1.15x per step and 30 total steps. In Figures 2a and



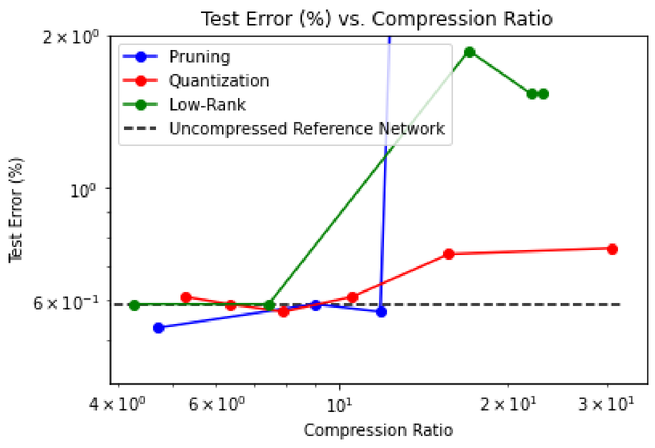
(a) Figure 1a: Accuracy per epoch with the following training parameters: batch size: 16, lr: 0.005, gamma: 0.95, step size: 1, momentum: 0.9, weight decay: 0, nesterov, True



(b) Figure 1b



(a) Figure 2a



(b) Figure 2b

2b, we show the resulting test errors by applying Pruning, Quantization, and Low-Rank individually using these parameters, for different compression ratios. In Figure 3, we show the test errors for quantization on codebook sizes from 2 to 64, applied to every layer. For each compression type, we applied different values of the associated compression parameters (codebook size for quantization, α for low-rank, and κ for pruning) to find the values necessary to give a decent range of compression. For codebook size, we ended up using the powers of 2 from 2 to 64. For α, we used values from 10^{-9} to $5 * 10^{-8}$. For κ, the actual values used depended on how many parameters the initial model/layer had, but they generally went from about $\frac{1}{35}$ to $\frac{1}{5}$ of the original size.

4.1 Mixing Compression Types

Our approach to testing out different mixes of compression types was to go through each combination of two compression types, and test a few models where the convolutional layers are compressed with one type, and the linear layers are compressed with the other type. Our motivation for taking this approach was the assumption that different compression types would have a different impact on convolutional and

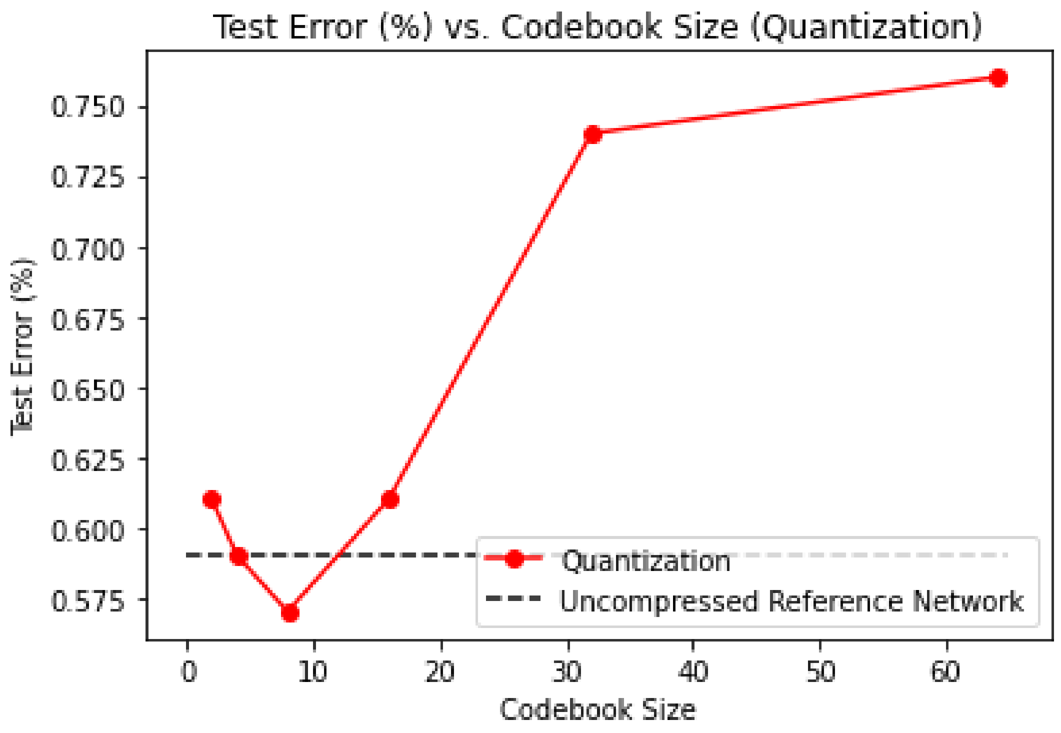


Figure 3

linear layers. At the same time, we would also be testing how different compression types behave at the beginning or end of the neural network, since we always have the first two layers being convolutional and the last two layers being linear. We would then take what we observed from each of the combinations to try to come up with the combination of all three compression types that was most likely to work well in our opinion.

4.1.1 Pruning on Convolutional layers and Quantization on Linear layers

With this compression scheme, we applied pruning to the convolutional layers, and quantization to the linear layers. Our initial findings were that the number of parameters of the first layer and the codebook size of both linear layers could be pretty low without much harm to the model. This compression scheme gave us one of our highest compression ratio to error ratios, with a test error of .51% and compression ratio of approx 26.4. This was given by compression with 104 parameters in the first convolutional layer, 2505 parameters in the second convolutional layer, and a codebook size of 2 for both linear layers. We tried increasing the compression a little bit more from that, and saw that it was impacting error enough as to not be worth pushing further. It looks like we got lucky with these particular compression parameters, as the test error for the model mentioned above is lower than those of models compressed with the same scheme but with more parameters in the second layer. A plot of this compression type's test error vs compression ratio is given in Figure 4. The results from training models with this compression scheme are in Table 1.

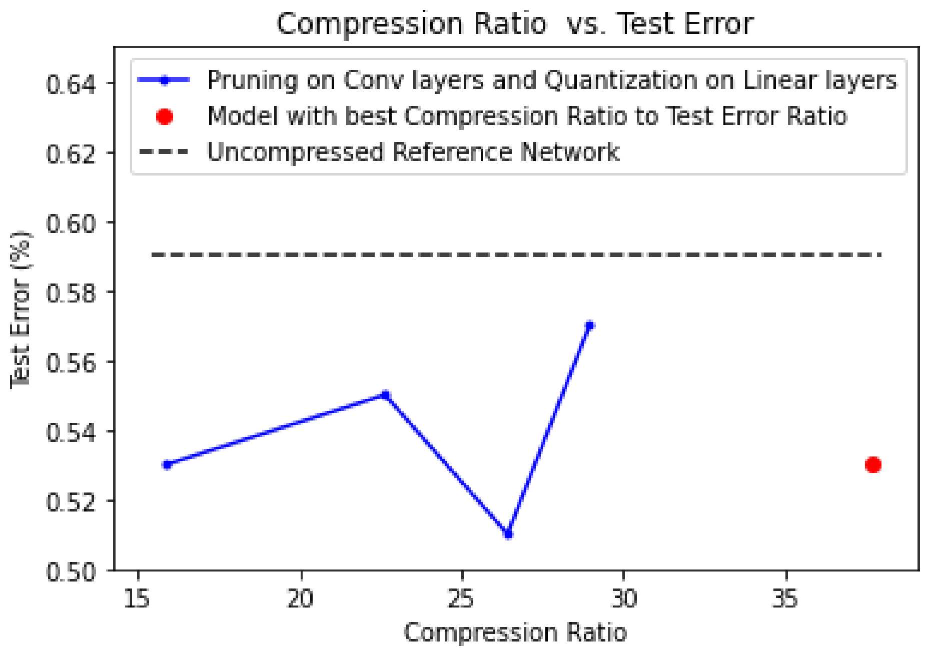


Figure 4: The model shown here with the lowest test error was compressed with 104 parameters in the first convolutional layer, 2505 parameters in the second convolutional layer, and with a codebook size of 2 for the linear layers

Table 1. Pruning on Convolutional layers and Quantization on Linear layers

Layer 1 # of Compressed Parameters (κ)	Layer 2 # of Compressed Parameters (κ)	Quantization Codebook Size (k)	Train Error (%)	Test Error (%)	# of Compressed Parameters	Compression Ratio
104	12525	2	0.01	0.53	415289	15.891025551894028
104	5010	2	0.00	0.55	407618	22.635991067346463
104	2505	2	0.03	0.51	405113	26.42542655375396
104	1250	2	0.06	0.57	403858	28.963396605759566

4.1.2 Quantization of Convolutional layers and Low-Rank on Linear Layers

For this compression scheme, we saw that we were able to get a pretty high compression while maintaining good accuracy. We wanted to see if reducing the amount of compression would allow the model to perform better with a still relatively high compression. When we lowered the compression by a factor of 3, we saw the error hardly decreased, and we had already seen a model that had higher compression with less error than the second run. Because of this, we decided to stop here, but the first run of this compression scheme still ended up giving our highest compression ratio with not terrible accuracy. The results for this scheme are shown in Table 2.

Table 2. Quantization on Convolutional layers and Low-Rank on Linear Layers

Quantization Codebook Size (k)	Low-Rank α	Train Error (%)	Test Error (%)	# of Compressed Parameters	Compression Ratio
2	$1 * 10^{-9}$	0.01	0.63	54029	14.333192590517836
2	$2.625 * 10^{-9}$	0.1	0.67	33229	47.09162580521104

4.1.3 Low-rank on Convolutional Layers and Quantization on Linear Layers

With this compression scheme, we decided to keep the codebook size as 2 for the quantization of linear layers for all runs, since we have noticed that models tend to work pretty well with that type of compression. Here, we varied the amount of low-rank compression on the convolutional layers by changing the alpha value. There were no stand-out compressed models from this compression scheme, but the results are shown in Table 3.

Table 3. Low-Rank on Convolutional Layers and Quantization on Linear Layers

Low-Rank α	Quantization Codebook Size (k)	Train Error (%)	Test Error (%)	# of Compressed Parameters	Compression Ratio
$1 * 10^{-9}$	2	0.00	0.55	434179	9.559551326197454
$2.5 * 10^{-9}$	2	0.00	0.65	422279	13.01388841442816
$5 * 10^{-9}$	2	0.00	0.59	410379	20.37716112851174
$1 * 10^{-8}$	2	0.00	0.67	407229	23.966670627791718
$2.5 * 10^{-8}$	2	0.01	0.74	406179	25.461728688445458
$5 * 10^{-8}$	2	78.45	78.98	404114	29.02223689445305

4.1.4 Quantization on Convolutional Layers and Pruning on Linear Layers

With this compression scheme, we again keep our quantization codebook at 2 for both layers, but apply quantization to the convolutional layers only. On the linear layers, we apply pruning with different numbers of parameters proportional to the number of starting parameters for each layer. We saw strange results here, with lower compression ratios giving higher error. The best compressed model that we got with this compression scheme had relatively high error of .78%, with a compression ratio of only approx. 17. We noticed that this scheme gave some of our worst results, and is the inverse of the scheme that gave some of our best results (quantization on linear layers and pruning on convolutional layers). This leads us to believe that, for this model, quantization is more effective on linear layers and pruning is more effective on convolutional layers. The results from this compression scheme are shown in Table 4.

Table 4. Quantization on Convolutional Layers and Pruning on Linear Layers

Quantization Codebook Size (k)	Layer 3 # of Compressed Parameters (κ)	Layer 4 # of Compressed Parameters (κ)	Train Error (%)	Test Error (%)	# of Compressed Parameters	Compression Ratio
2	100000	1000	0.11	0.84	126504	3.766573718079878
2	80000	500	0.14	0.82	106004	4.660869919845843
2	40000	250	0.10	0.84	65754	8.876197675713039
2	20000	125	0.06	0.78	45629	16.641104294478527

4.1.5 Low-Rank on Convolutional Layers and Pruning on Linear Layers

With this compression scheme, we again observed that pruning was not working very well on the linear layers. For very low compression ratios, we were already seeing error rates higher than we had seen before with much higher compression ratios. When we pushed the model to have a compression ratio close to 20, it started giving garbage output, so we decided to stop there. The results are shown in Table 5.

Table 5. Low-Rank on Convolutional Layers and Pruning on Linear Layers

Low-Rank α	Layer 3 # of Compressed Parameters (κ)	Layer 4 # of Compressed Parameters (κ)	Train Error (%)	Test Error (%)	# of Compressed Parameters	Compression Ratio
$1 * 10^{-9}$	80000	500	0.00	0.63	111125	3.5195712376729418
$2.625 * 10^{-9}$	40000	250	0.00	0.57	56875	6.693464472083884
$2.625 * 10^{-9}$	15000	250	80.15	80.41	17875	19.249415404136105

4.1.6 Pruning on Convolutional Layers and Low-Rank on Linear Layers

This compression scheme gave us our best compression ratio to error ratio. With 104 parameters in the first layer, 2505 parameters in the second layer, and an α of $2.5 * 10^{-9}$ for low-rank compression applied to the linear layers. The resulting error was .53%, and the compression ratio was 37.7. We have noticed a trend that some of our best models were compressed with pruning on the convolutional layers, so when we tried to combine all three compression types into one model, we decided to fix the compression on the first two layers to be pruning with 104 parameters for the first layer and 2505 parameters for the second layer. The results from these runs are shown in Table 6. A plot of this compression type's test error vs compression ratio is given in Figure 5.

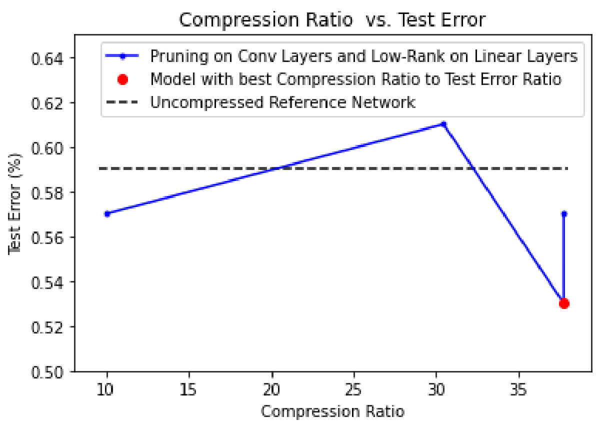


Figure 5: The model shown here with the lowest test error was compressed with 104 parameters in the first convolutional layer, 2505 parameters in the second convolutional layer, and with an α of $2.5 * 10^{-9}$ for the linear layers

Table 6. Pruning on Convolutional Layers and Low-Rank on Linear Layers

Layer 1 # of Compressed Parameters (κ)	Layer 2 # of Compressed Parameters (κ)	Low-Rank α	Train Error (%)	Test Error (%)	# of Compressed Parameters	Compression Ratio
104	5010	$1 * 10^{-9}$	0.00	0.57	41439	10.04371385404743
104	5010	$2.5 * 10^{-9}$	0.00	0.61	12839	30.45465244869227
104	2505	$2.5 * 10^{-9}$	0.00	0.53	10334	37.73248519798384
104	2505	$2.75 * 10^{-9}$	0.00	0.57	10334	37.72926725264447

4.1.7 Mixing All Compression Types into One Model

As stated above, we started by setting the compression type for the convolutional layers to pruning, with 104 parameters for the first layer and 2505 for the second layer. We also kept the quantization codebook size at 2 for all runs, since this maximizes the compression ratio and didn't seem to hurt accuracy much in previous experiments. That left us with only two free parameters: the α value for low-rank compression, and which of the linear layers to apply quantization or low-rank compression to. We tried both putting quantization first and putting low-rank first, and for each one we tried α values on the lower and higher range of what we had used for previous tests. For all tests, the resulting compression ratio to test error ratio wasn't holding up to the models that we had found by just mixing two compression types. The results for the tests that we ran are in Table 7.

Table 7. Mix of Pruning, Low-Rank, and Quantization Compressions

Layer 1 # of Compressed Parameters (κ)	Layer 2 # of Compressed Parameters (κ)	Layer 3 Compression	Layer 4 Compression	Train Error (%)	Test Error (%)	# of Compressed Parameters	Compression Ratio
104	2505	Quantization w/ k = 2	Low-Rank w/ $\alpha = 1*10^{-9}$	0.06	0.74	405136	22.971771039394586
104	2505	Quantization w/ k = 2	Low-Rank w/ $\alpha = 2.5 * 10^{-9}$	0.09	0.67	405136	22.973618130436893
104	2505	Low-Rank w/ $\alpha = 2*10^{-9}$	Quantization w/ k = 2	0.19	1.17	10311	48.08948577239336
104	2505	Low-Rank w/ $\alpha = 2.5 * 10^{-9}$	Quantization w/ k = 2	0.04	0.80	10311	48.08661930842
104	2505	Low-Rank w/ $\alpha = 3*10^{-9}$	Quantization w/ k = 2	0.09	0.80	10311	48.08594489596993

5 Comparison to Part 1 Results

In part 1, the test error for our reference network was 1.33%. In part 2, the test error was only .59%. Figures 6-8 below show the change in model accuracy for different compression ratios on each compression type. As you can see from the plots, error for all compression types on both parts tended to increase as compression ratio increased, sometimes very dramatically. It looks like pruning worked better as compression ratio increased on the part 1 model, and low-rank performed better on the part 2 model.

Overall, we can see clearly that convolutional layers at the beginning of the network really help with accuracy on image data. It would also seem based on our results that reducing the number of parameters in the initial convolutional layers can actually help model performance, since our best compressed test error was lower than the reference network's error. We managed to compress a model with a compression ratio of about 26 which had the lowest test error we saw at .51%. We also got a model with a compression ratio of about 47, which maintained .67% accuracy. This is about half of the error that we saw for the part 1 uncompressed network, with a considerably smaller model. For what we were calling our ”best” model, we got an error down to .53%, with a compression ratio of about 37.7.

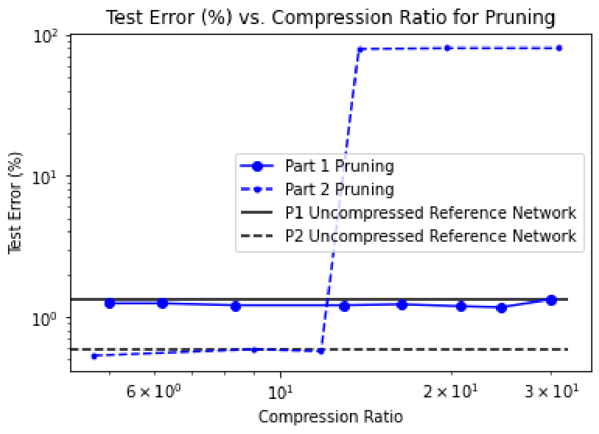


Figure 6

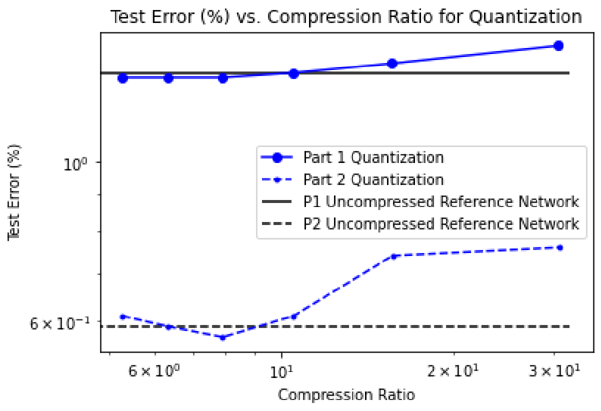


Figure 7

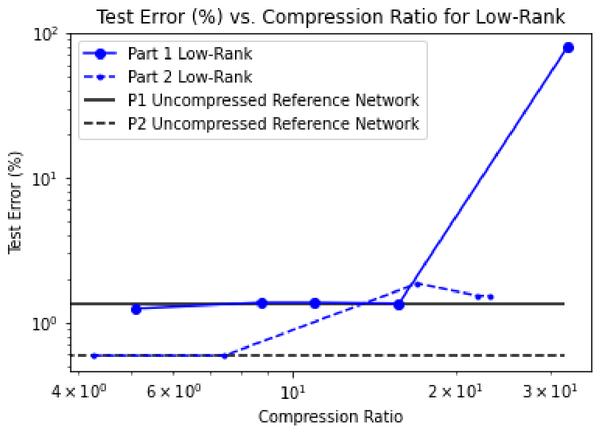


Figure 8