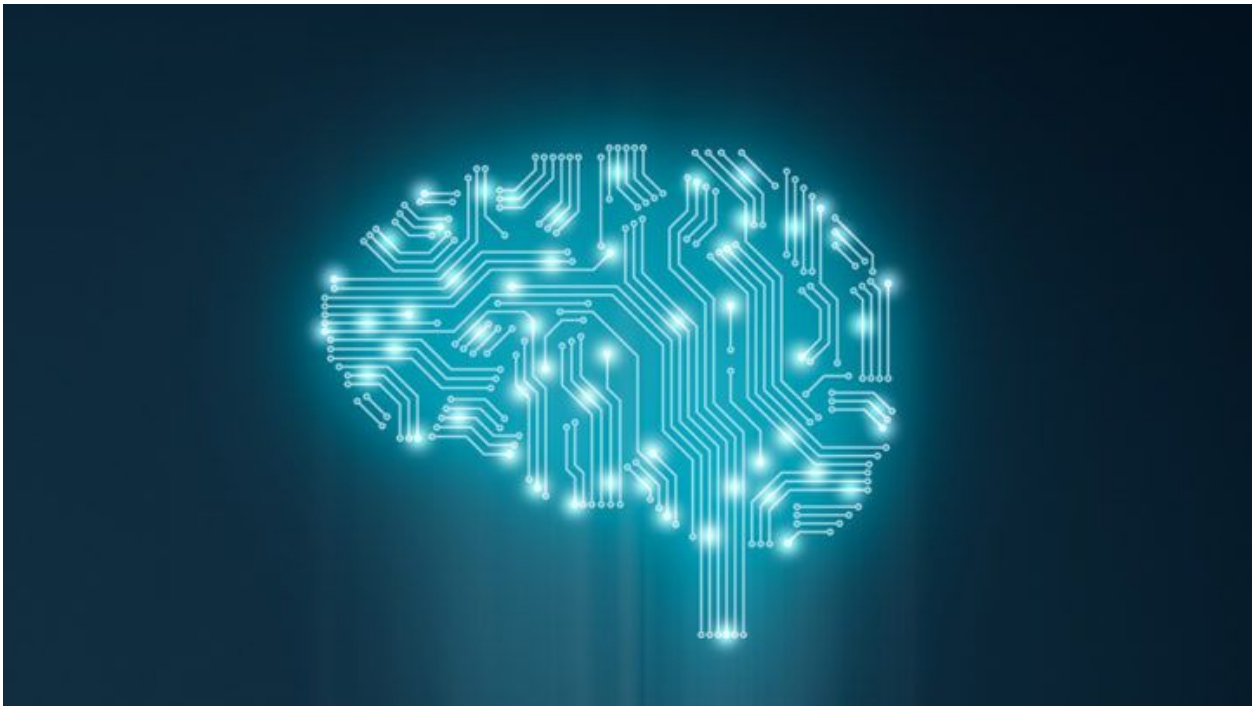


(“A Conversation With”)

Research Internship Report:
Machine Learning Optimization

Jeffrey Cao
sparklyjeff@gmail.com

What makes a cat recognizable as a cat and not as a dog? Or what about the difference between a ball and a hairbrush? Machine learning utilizes self-refining formulas to slowly uncover the hidden patterns of our world, but at the cost of exponentially increasing computational costs. To combat this, pruning is one of many procedures that simplify the computation for added efficiency. Through my experiments, I tested various methods of pruning to increase the accuracy and efficiency of a simple PyTorch image recognition model. Between pruning by standard deviation, pruning by percentile, soft thresholding, and trimming, it was evident soft thresholding produced the highest potential accuracy of 97.47%.

Table of Contents

Pruning by Standard Deviation

BACKGROUND:

Pruning is a method of machine learning that optimizes efficiency by setting less important values to 0 for a faster computation rate. In this experiment, I ran a pruning procedure that prunes away data based on the standard deviation (STD) of the data set and a given threshold.

$$s_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

$n =$ The number of data points

$\bar{x} =$ The mean of the x_i

$x_i =$ Each of the values of the data

Standard Deviation Formula("How to Calculate")

The Pruning.py model that I used for each experiment implements a single step pruning between an initial training and a final retraining after values are pruned. Both the initial and the retraining undergo 100 epochs of training. The final accuracies are calculated on a separate test data set.

PROCEDURE:

I ran the Pruning.py model that optimizes its sample training size by pruning away certain values after calculating the data set's STD while adjusting the sensitivity of the model to affect its respective compression rate. After running the model multiple times, I recorded the accuracies for both after pruning and the retraining.

DATA:

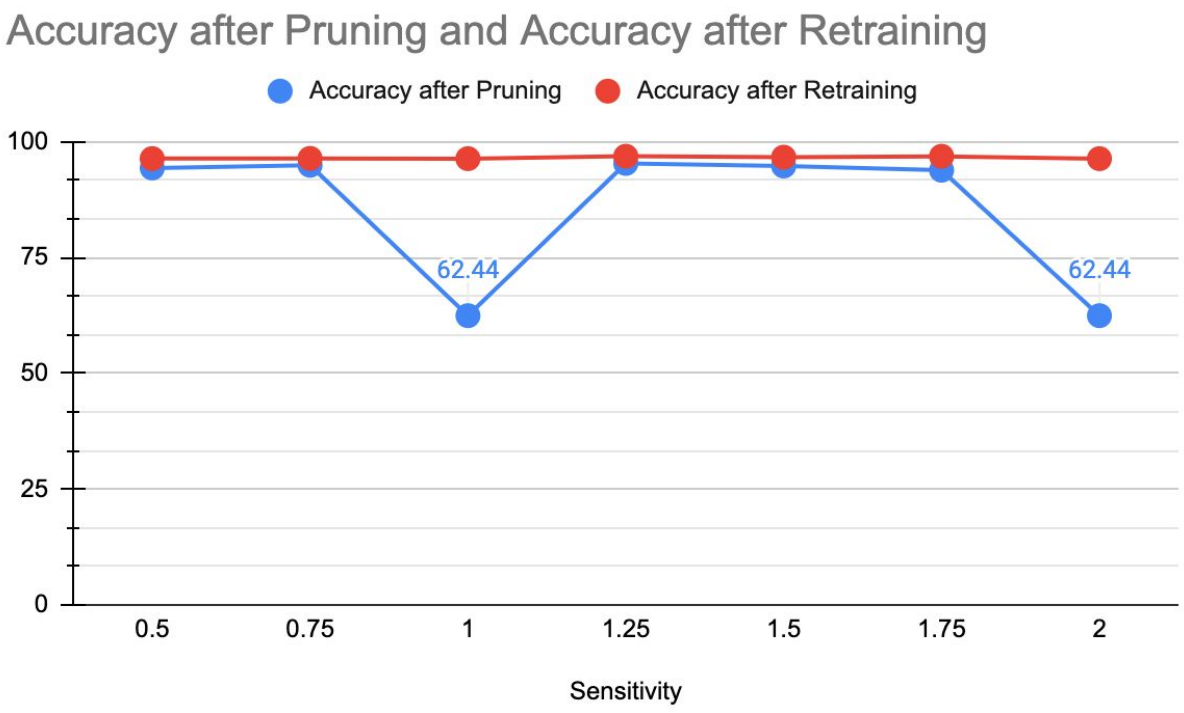


Figure 1. Effect of Sensitivity on Accuracy After Pruning and Accuracy after Retraining

Sensitivity	Accuracy after Pruning	Accuracy after Retraining
0.5	94.32	96.35
0.75	94.91	96.39
1	62.44	96.34
1.25	95.31	96.86
1.5	94.75	96.65
1.75	93.85	96.85
2	62.44	96.34

Figure 2. Data Table of Values from Pruning by STD

Aftering testing varying sensitivities from 0.5 to 2, the pruning by STD model showed slight variation in the Accuracy after Pruning, while there were little to no differences in the Accuracy after Retraining as shown in figure 1. The most notable difference is the similar sudden drop in Accuracy after Pruning in the 1 and 2 sensitivity levels. The outlier value from the sensitivity level of 1 is mostly likely due to human error since the sensitivity is automatically set to 2 if no sensitivity was provided.

WHY:

Since this is a small sample size, the variations are expected to be minimal as displayed in the chart. The performance peaked at around 96.86% at around 1.25 sensitivity, but most of the models converged to a similar accuracy after retraining. This is because even with a higher compression rate, the small sample size means that there is little difference in the information processed, thus the accuracy will remain similar. Overall, there was a distinct correlation between increasing the sensitivity and the Accuracy after Retraining with a negative correlation between sensitivity and Accuracy after Pruning. This is because increasing the sensitivity will raise the compression rate and prune away larger amounts of data so the Accuracy after Pruning will drop. However, after Retraining, the optimization the pruning provides increases both the program's efficiency and produces similar, if not better, results.

PROS:

- Higher Accuracies

CONS:

- Not Consistent

Pruning by Percentile

BACKGROUND:

While Pruning by STD produced data with a clear trendline, some data points were more random than others and did not follow a clear pattern. Since the STD can be random depending on the type of data in each epoch and training steps, this model might not be as consistent as a model that has clear pruning goals. In Pruning by Percentile, a given percentile dictates how much data is pruned away each time the prune() is called. For example, at a 70% sensitivity, the data values with an absolute value of below a 70th percentile in its data group will be pruned to 0. Since this will always prune away a set amount of data, even if the data is very similar or very different, the model will be more consistent in keeping the more important data points.

PROCEDURE:

I ran the Pruning.py image recognition model with a modified pruning method that prunes away a percentile of the data set in an attempt to maximize efficiency. Each sensitivity passed in represents the lower percentile of data pruned away. After each run, the corresponding Accuracy after Pruning and Accuracy after Retraining and compression rates are recorded.

DATA:

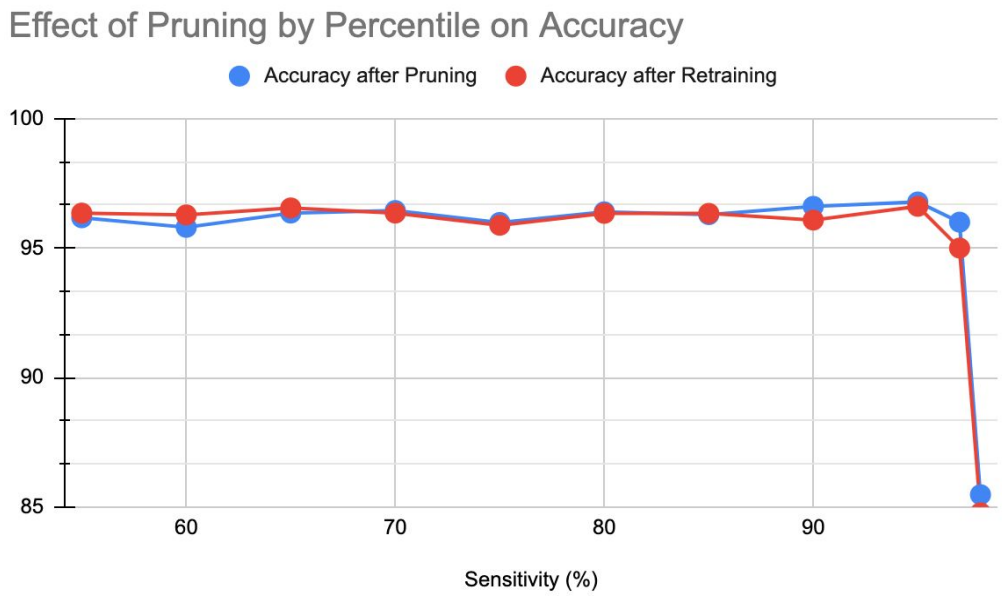


Figure 3. Line graph of Accuracy after Pruning and Retraining in comparison to Sensitivity

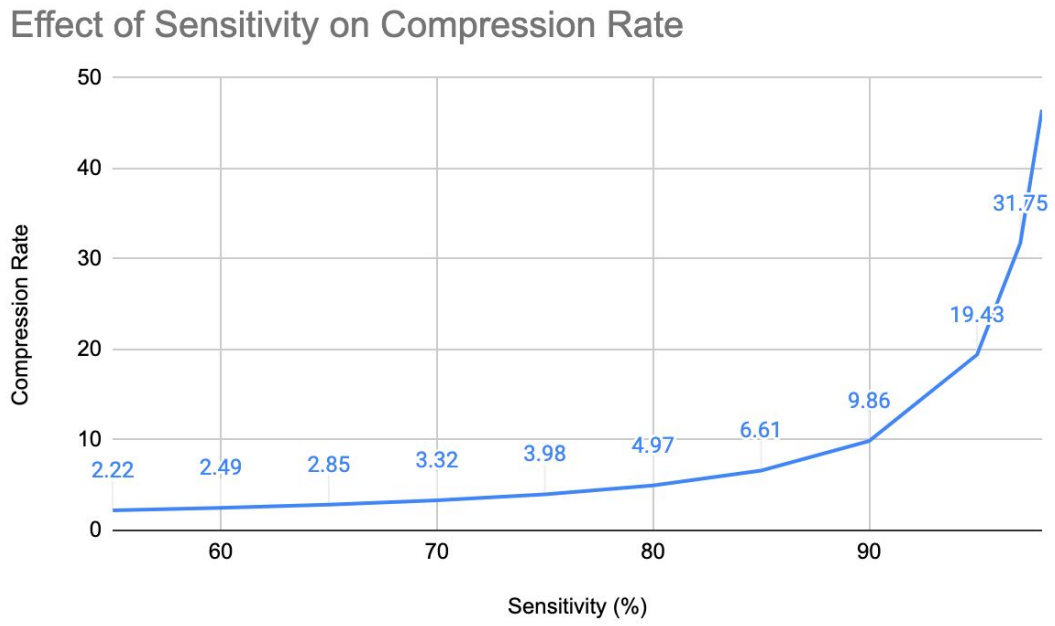


Figure 4. Line graph of Compression Rate in comparison to Sensitivity

Sensitivity	Accuracy after Pruning	Accuracy after Retraining	Compression Rate
98	85.48	84.78	46.5
97	96	95	31.75
95	96.78	96.61	19.43
90	96.61	96.08	9.86
85	96.29	96.34	6.61
80	96.4	96.34	4.97
75	95.99	95.88	3.98
70	96.45	96.35	3.32
65	96.35	96.55	2.85
60	95.8	96.28	2.49
55	96.17	96.35	2.22

Figure 5. Table of all data points recorded

Although I wasn't able to test all values of sensitivity from 100 to 0, the general trend is very apparent. Figure 3 displays the steady average accuracy around 96% that decays exponentially after around 97%. As seen in Figure 4, increasing the sensitivity only increments the compression rate by a slight amount until around 90% when the compression rate jumps in an exponential rate.

WHY:

The exponential growth of the compression rate reflects how much data is actually lost when the sensitivity starts to reach around 90% when almost every piece of data is pruned away. When over 90% of data is pruned away, more potentially crucial data values can be lost and sharply drop the program's accuracy as displayed in Figure 3. No matter how much the program retrains, the 97% of data loss sharply hinders its ability to assess the data. In other sensitivities, the data loss is only moderate and retains enough crucial data to perform efficiently. As the percentile of data pruned drops, the Accuracy after Retraining begins to slowly creep above the Accuracy of Pruning. This displays the model training itself with more data as the sensitivity drops to reach better Accuracy percentages.

PROS:

- Consistency

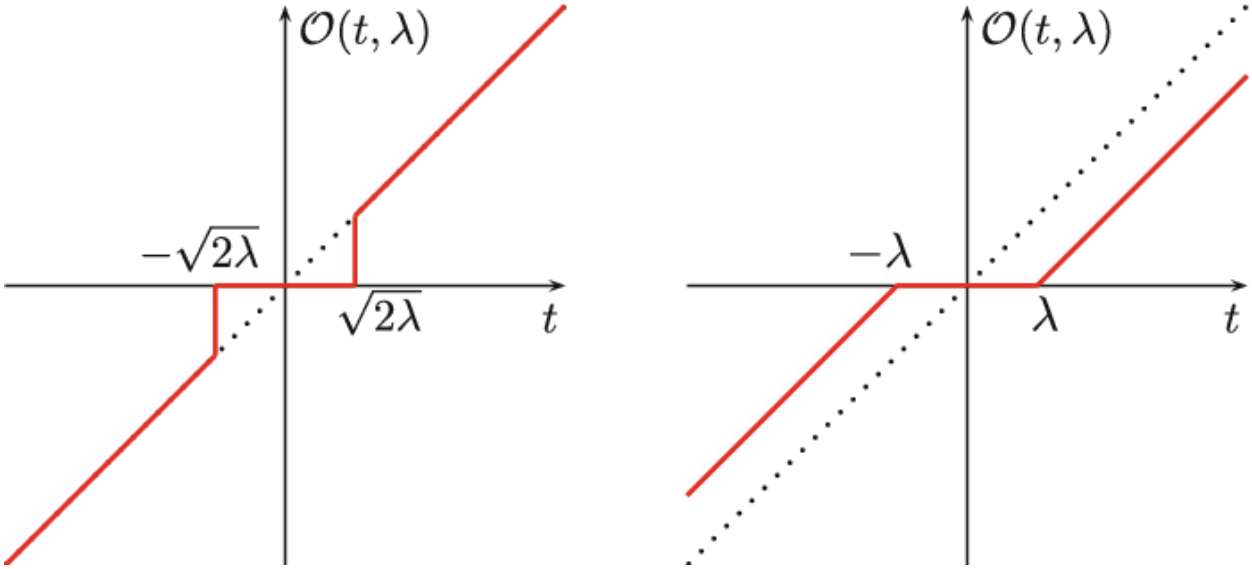
CONS:

- Less Accuracy

Soft Thresholding

BACKGROUND:

Between pruning by STD and pruning by percentile, both methods do a hard thresholding where every single value is either set to 0 or unchanged. While this may help the model identify the important data values quicker, this may also be detrimental to the model's accuracy. Since smaller values are completely reset to 0, the difference between larger values and smaller values are increased to larger values and 0. The model will place a much larger emphasis on the larger values now that they stand out even more.



Soft Thresholding Visualization (Hoang)

PROCEDURE:

I ran the Pruning.py model with a customized soft threshold pruning model with varying threshold values. After each run, I recorded the Accuracy after Pruning and the Accuracy after Retraining.

DATA:

Threshold	Accuracy after Pruning	Accuracy after Retraining
0.01	96.98	96.87
0.05	96.33	97.47
0.001	96.08	96.67
0.0001	95.3	95.49
0.00001	95.98	95.08

Figure. 6 Table Chart of Data

Threshold values between 0.05 and 0.0001 displayed a higher Accuracy after Retraining than the Accuracy after Pruning as shown in figure 6. The highest overall accuracy was in Accuracy after Retraining at the 0.05 threshold.

WHY:
I believe that when the threshold is set too large or too small, such as 0.01 or 0.00001, the model has either too much variation in the data or too little variation to reach an optimal performance. Something to note is that around 0.05, the final accuracy reached round 97.47%, which is one of the highest accuracies achieved out of any of my tested pruning models. Since soft thresholding lessens the impact of pruning away data to 0, it can formulate a better model.

PROS:

- Reduces dramatic differences

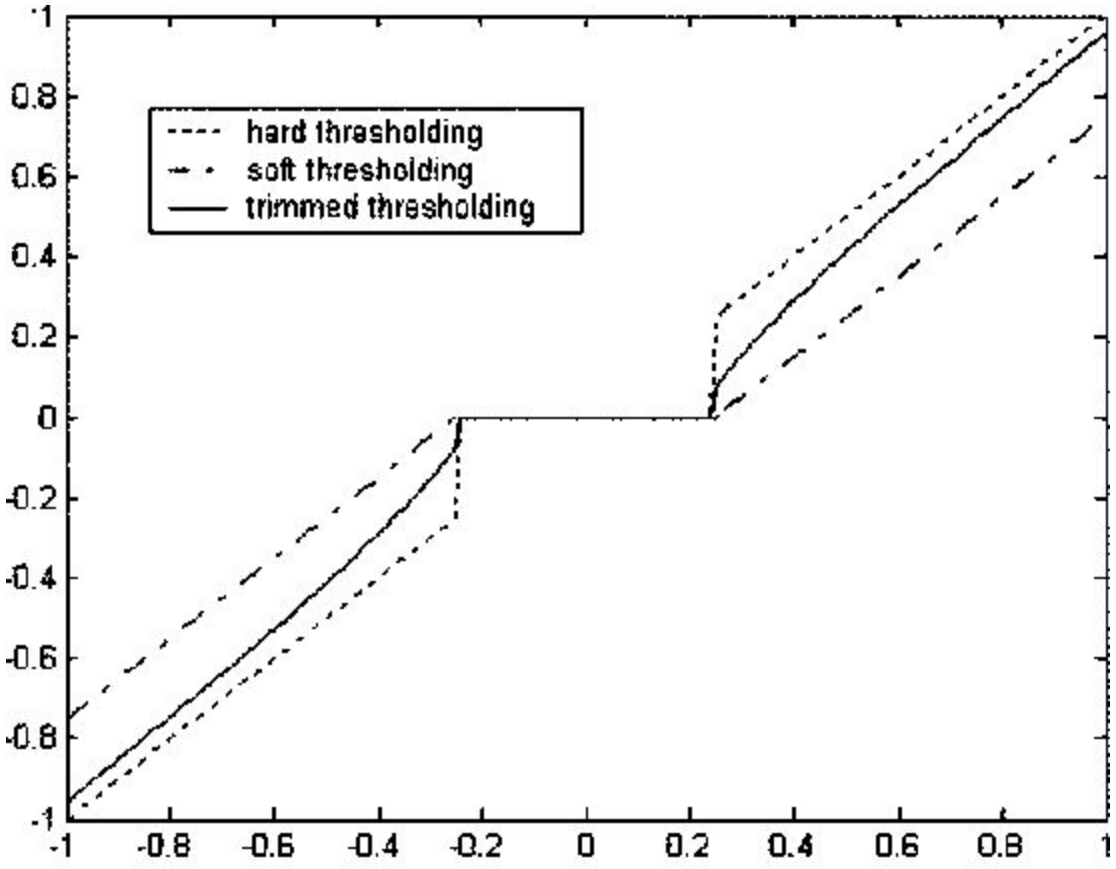
CONS:

- Minimizes impact of largest values

Trimming

BACKGROUND:

While soft thresholding helps the model minimize the impact of making a massive shift from 0 to large values, it also means that the impact of the largest values are also minimized. This could potentially be detrimental in a model that needs to recognize the largest values as much as possible since those data points are generally the most crucial for the model. Trimming offers a hybrid between soft thresholding and hard thresholding. At lower values, the data is pruned by a soft threshold while data above a certain percentage is kept the same to preserve the largest values.

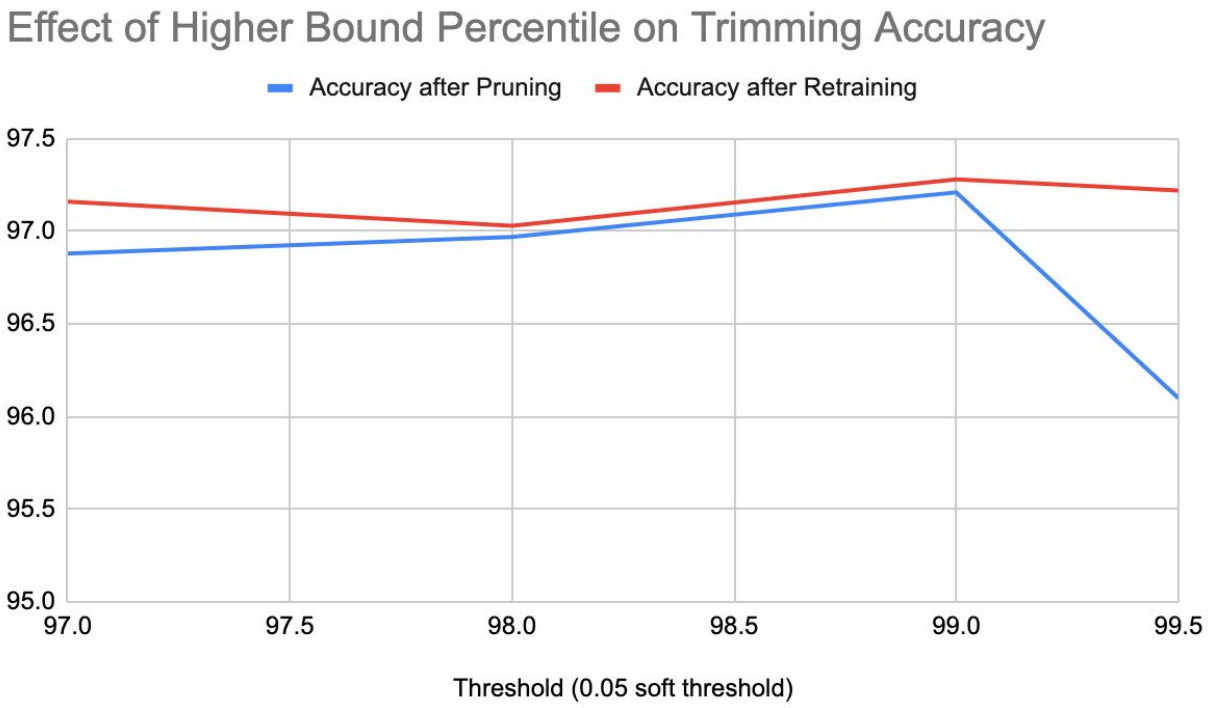


Trimmed Thresholding Visualization (Fang)

PROCEDURE:

I ran the Pruning.py image recognition model using a trimming method with a 0.05 threshold for soft pruning and varying upper bound thresholds.

DATA:



Threshold (0.05 soft threshold)	Accuracy after Pruning	Accuracy after Retraining	Compression Rate
99.5	96.1	97.22	122.81
99	97.21	97.28	85.04
98	96.97	97.03	46.25
97	96.88	97.16	31.73

All the retraining accuracies ended up around 97% with a 99% threshold achieving a 97.28% accuracy, which is the highest of any model so far. Increasing the threshold also increases the compression rate of the model.

WHY:

The benefits of a trimming method is that it maintains the higher impact of the largest data points, which can help the model isolate the more important data values. If the threshold is too high, too much crucial data is trimmed and will not have the same efficiency. On the other hand, if the upper bound is too low, it will also be more difficult for the model to figure out the important data points. The highest accuracy was found around a 99% threshold because the remaining 1% has a larger impact in this model. Although it did not produce a higher accuracy than the previous thresholding, all the values are a lot more consistent at a higher average accuracy.

PROS:

- Consistency

CONS:

- Less accuracy

Future Questions

- How would these methods perform on a much larger model?
    - GPU
    - CIFAR-10
    - Iterative Pruning

Since I was limited to a laptop, I was unable to run excessively large data sets without risking damage to my CPU. This meant that the model I trained for each trial was one of the simplest that only had 100 epochs of training and retraining and only labeled images into a few categories. A small model also meant that the differences in my data points were very subtle which can make it more difficult to draw precise conclusions.

- Is it possible that I missed a crucial data point since I was only able to test a limited amount of data points?

The highest accuracy that I was able to train during this summer was 97.47% from soft thresholding, but the trimmed thresholding should theoretically be able to achieve a higher accuracy since it covers more weakness of the previous model. This also has to do with the limitations of a small model that make differences between values very minute. It is very possible that there is a much higher potential accuracy on each model that I was unable to test.

- Are there any training and coding methods to optimize those aspects of the model too?

Pruning optimizes the model's data input by processing it for the model to understand easier and faster, but there are also a lot of aspects of the training that could also be optimized. At a certain point, pruning will have reached the maximum potential optimization possible on its own and will need to rely on compounded other methods on top of it.

Works Cited

"A Conversation With BlackLine's Machine Learning Experts." *BlackLine Magazine*, 21 Mar. 2019, www.blackline.com/blog/finance-automation/machine-learning-experts/.

Fang, Hai-Tao, and D. D Huang. "Wavelet De-Noising by Means of Trimmed Thresholding: Semantic Scholar." *Undefined*, 1 Jan. 1970, www.semanticscholar.org/paper/Wavelet-de-noising-by-means-of-trimmed-thresholding-Fang-Huang/a2e34becb9f42b6b6e274c97c3f70f7d4f93ffbd.

Hoang, Thai V, and Elisa H Smith. "Sparsity-Based Edge Noise Removal from Bilevel Graphical Document Images." *ResearchGate*, Aug. 2014, www.researchgate.net/publication/258161038_Sparsity-based_edge_noise_removal_from_bilevel_graphical_document_images.

"How to Calculate Standard Deviation: Standard Deviation, Math Work, Math Poster." *Pinterest*, www.pinterest.com/pin/302796774919831541/.