The Effects of Pruning on CIFAR-10 Model

Ruben Parra Montenegro and Brandon Mercado Franco
Oakland University Graduate and Undergraduate Student
318 Meadow Brook Rd, Rochester, MI 48309
parramontenegro@oakland.edu or bmercadofranco@oakland.edu

Abstract

This study investigates the effects of the implementation of the pruning compression methods. The evaluation metrics used for the compression model are the number of parameters, FLOPS, training accuracy and test accuracy. Once the model is fully completed, further analysis is conducted to investigate the actual effects pruning had on the model. Iterative and oneshot pruning will be implemented with structured and unstructured pruning with varying amounts of pruning and 3 learning rates.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

Contents

1 Pruning Implementation

1.1 Dataset

The dataset chosen in this project was CIFAR-10. Which is a dataset that is composed of 60,000 images in its entirety, 50,000 training images, and 10,000 testing images. The images are RGD images so the shape of the data will be [3,32, 32] for one image. Depending on our batch size we will take the data in like this: [Batch size, 3, 32, 32]

1.2 Model

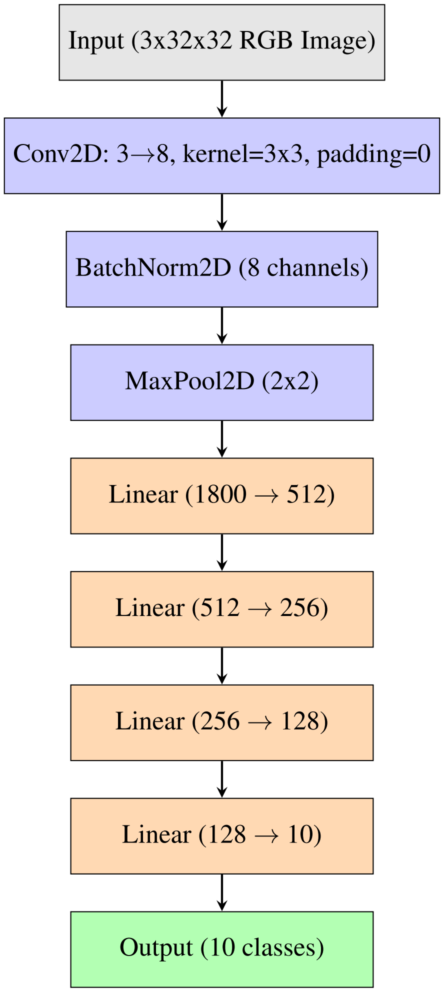

Figure 1: Current model before compression methods

The model above is what was developed in Project 1 of this class. Our batch size that was chosen for the model was 512, and the chosen learning rate was .8. 8 convolution channels were used as that's what ran decently well on our manual implementation. Implemented with libraries we could use more kernels, but we are limited to the model we made in project 2.

1.3 Pruning methods

In this study, the effects of iterative and one shot pruning were studied while varying the values of the learning rates and amount being pruned.

The learning rates that were looked at were .8, which was the learning rate used in the original model presented. As well as .01 and .0001 learning rates.

The basic pruning methods that PyTorch has were implemented for unstructured pruning: prune.l1_unstructured

For structured pruning this function was used: prune.ln_structured

Table 1: Percentage of Parameters Pruned During Training

Pruning Percentage
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
0.95
0.99

Above are the amounts that were pruned in one-shot pruning and over the iterative pruning process.

The models in the one-shot pruning process were each trained for 10 epochs over each amount of pruning. The model that was trained for Project 1 was used as the baseline for each of the 12 pruning amounts.

For the iterative pruning process, the base model used was the same model from project one. As the iterative training was done after each model was trained it was saved and loaded so it could be pruned more until the desired amount of 99 % of the parameters were pruned. For each pruning step the model was trained for 10 epochs.

2 Abalation Study Results(One Shot Pruning)

For the one shot pruning models that were trained, essentially 12 models were trained at various pruning amounts to see how they would behave.

2.1 Structured pruning

2.1.1 Learning rate: .8

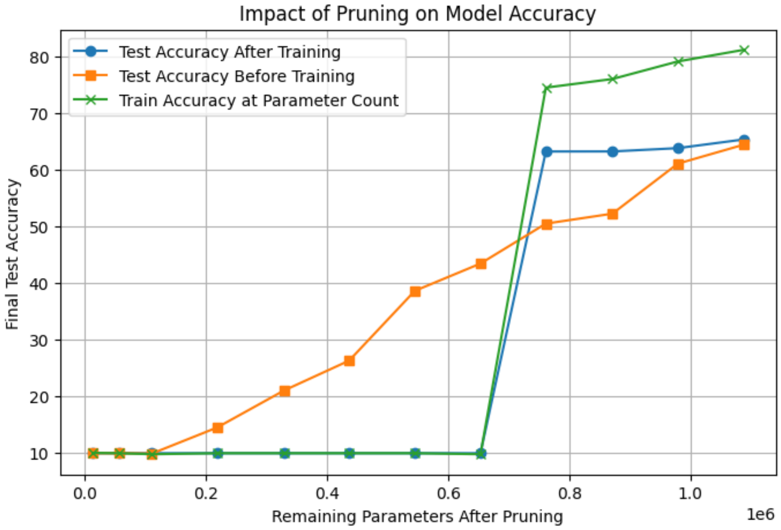

Figure 2: Test Accuracy of Models Pruned(Structured) with Learning Rate of .8

Here we can see that during the structured pruning the model does not react well to very aggressive pruning at a high learning rate such as .8 as it takes a huge dip down to an accuracy of 10 % after removing more than 30 % of the parameters in the model.

Now this can be attributed to most likely to the high learning rate, the small number of kernels(The model has 8 kernels).

The reason for believing the learning rate is messing with accuracy is due to the results of the training at lower learning rates such as .0001. There was not such a steep drop off of accuracy in the model and it was able to gain some accuracy from where it started. The drop of accuracy also happened a less aggressive manner. Kernels being wiped out entirely could also completely mess up the model and disallow it from functioning properly. The model seems to get stuck after 30 % of the parameters are removed.

2.1.2 Learning rate: .01

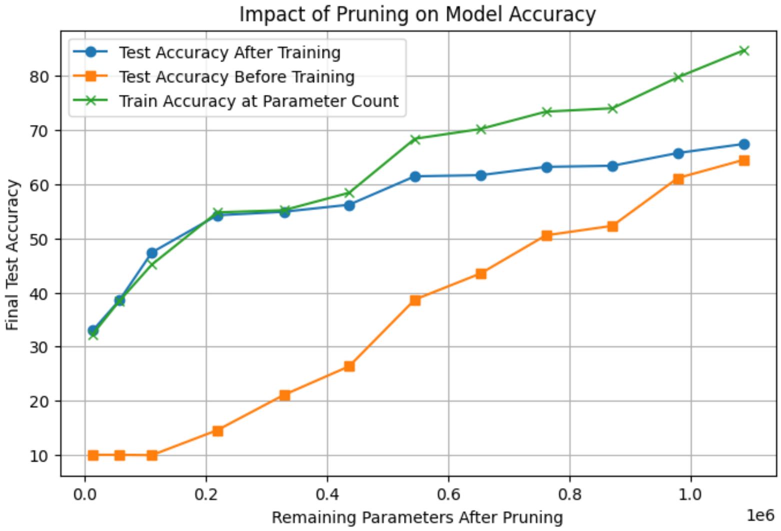

Figure 3: Test Accuracy of Models Pruned(Structured) with Learning Rate of .01

We can see from this graph that the learning rate of .01 seems to be the best out of the 3 chosen, as you see a good recovery from an initial test accuracy of 10 % at a pruning amount of 99 %. The models seem to not be overfitting either, as the training accuracy and the test accuracy after training seem to be pretty similar as more parameters are pruned.

2.1.3 Learning rate: .0001

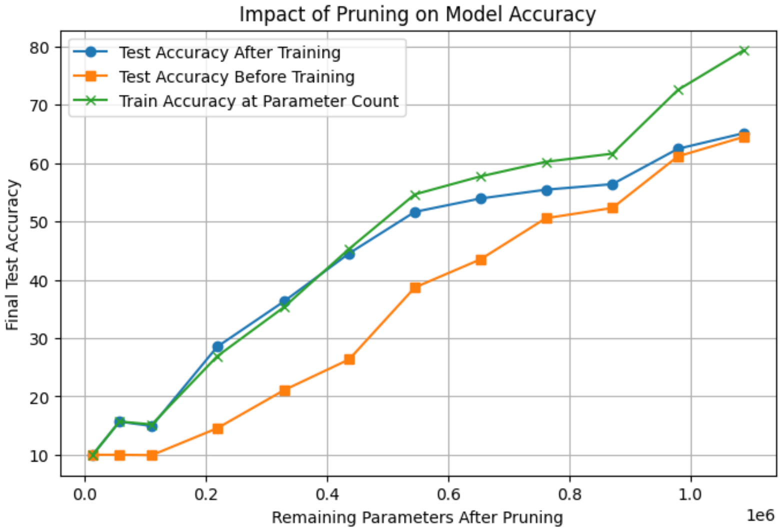

Figure 4: Test Accuracy of Models Pruned(Structured) with Learning Rate of .0001

While this way of training is not the best it is not the worst either, as you can see the models are not overfitting or underfitting, but the accuracy is not there when compared the the models that were trained with a learning rate of .01

2.2 Unstructured pruning

2.2.1 Learning rate: .8

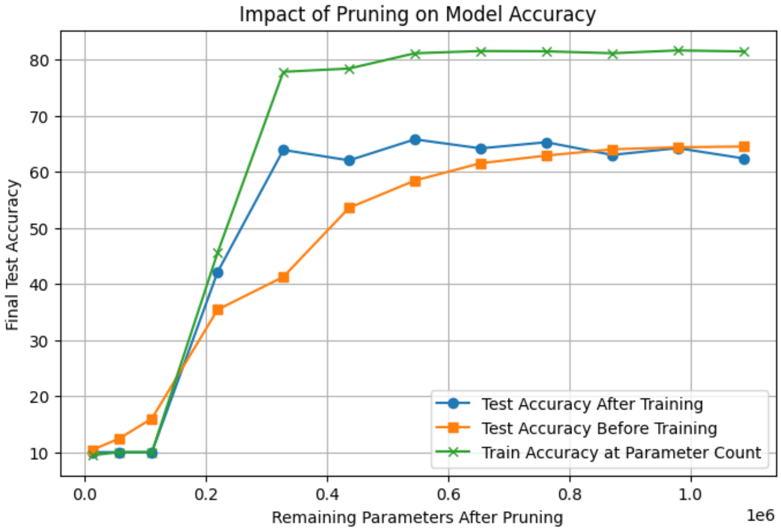

Figure 5: Test Accuracy of Models Pruned(Unstructured) with Learning Rate of .8

When training the pruned model with the unstructured pruning method, we see a drop off in accuracy in a much later stage in the pruning. This can most likely be attributed to this method of pruning being a far less aggressive way to prune the model especially since the model has so little kernels for the convolutions.

2.2.2 Learning rate: .01

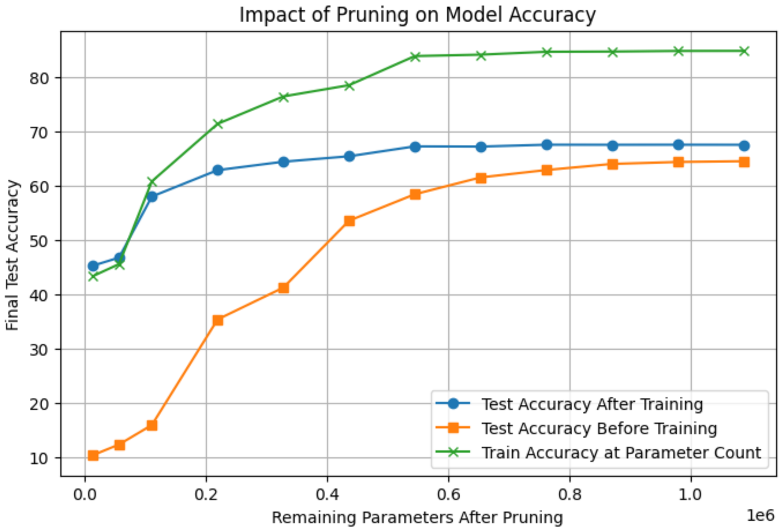

Figure 6: Test Accuracy of Models Pruned(Unstructured) with Learning Rate of .01

We can see again that models trained with a learning rate of .01 performed better when being pruned then at the other learning rates chosen. The accuracy drop when compared the amount of pruning is "minimal" when compared to the other rates chosen. We can also see that the training and testing accuracies converge as the model starts to behave similarly to its training.

2.2.3 Learning rate: .0001

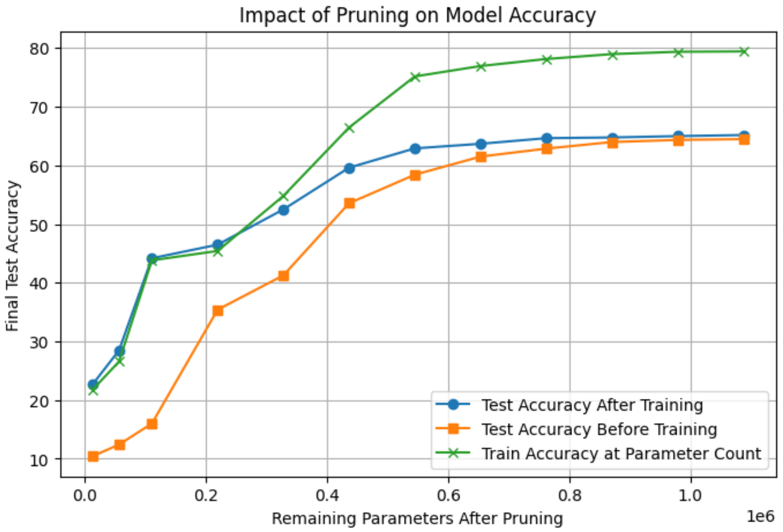

Figure 7: Test Accuracy of Models Pruned(Unstructured) with Learning Rate of .0001

All of the models in this section seem to have a fall in accuracy around the same area, around the middle of the chart, when about half of the parameters are missing. But it looks like there is a middle ground in the learning rates, as if we go too high, we see a sharp decline in accuracy, and if we go too low, the decline isn't at sharp but it happens over a longer period. On the other hand the middle ground learning rate(.01) has a very acceptable decline given the model that is being pruned.

3 Abalation Study Results(Iterative Pruning)

Iterative pruning was also done as exploring the differences between the 2 sounded interesting to see if one was objectively better than the other at a glance. They have their trade-offs. One such being that we train the model and iteratively prune it as we go instead of just picking an amount to prune and then just train it like in one shot pruning. Iterative pruning does have its advantages as the results are either better or similar to one shot pruning which is interesting to see.

Iterative pruning proved to be a better approach for at least the model we developed in terms of accuracy and accuracy drop off, but this might be due to the very nature of iterative pruning.

3.1 structured pruning

3.1.1 Learning rate: .8

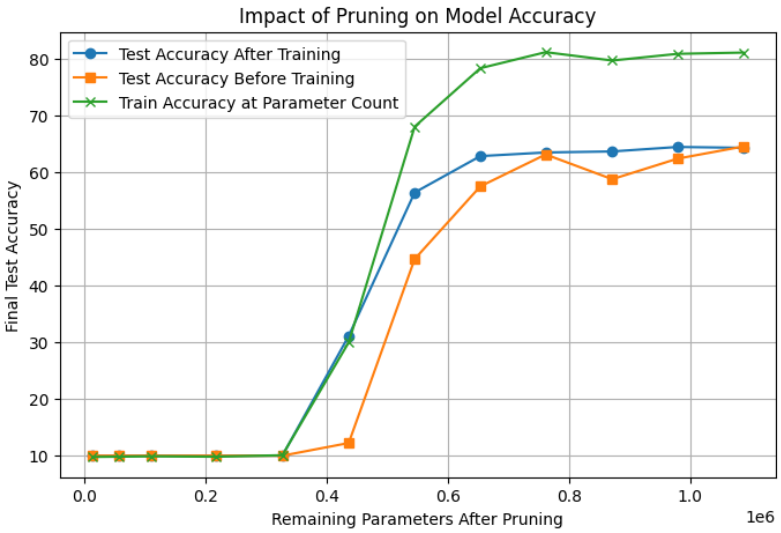

Figure 8: Test Accuracy of Model Iteratively Pruned(Structured) with Learning Rate of .8

When structured pruning was done with the one-shot method, the accuracy drop off was sooner and much more drastic for higher learning rate fine-tuning. But the model is overfitting and converges down to 10 % accuracy after about half of the parameters are removed.

3.1.2 Learning rate: .01

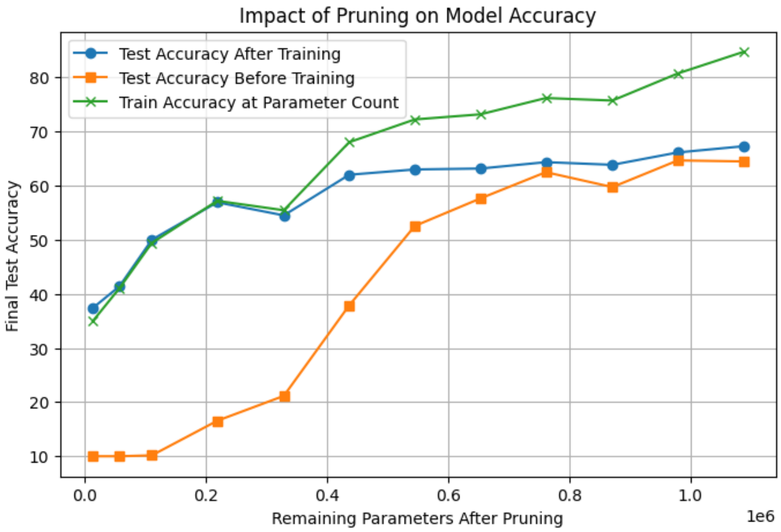

Figure 9: Test Accuracy of Model Iteratively Pruned(Structured) with Learning Rate of .01

We continue to see the trend of a middle ground learning rate which was .01, which allows the model to retain some accuracy even after loosing 99 % of all parameters, which includes all the kernels. The model also goes from slightly overfitting to having similar training and test accuracies towards the end. At an accuracy of about 40 % at 99 % of parameters removed, it is doing pretty well.

3.1.3 Learning rate: .0001

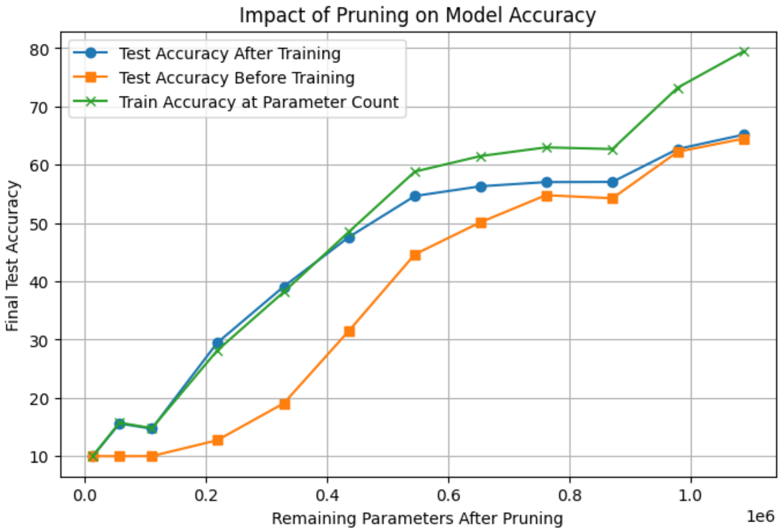

Figure 10: Test Accuracy of Model Iteratively Pruned(Structured) with Learning Rate of .0001

We continue to observe the trends of going from higher learning rates to lower learning rates and see that the lower the rate gets the closer the accuracy declines at a linear rate with the removal of parameters which is relatively interesting suggesting the training at the learning rate is insignificant but not completely tanking the accuracy like something with a high learning rate.

3.2 Unstructured pruning

3.2.1 Learning rate: .8

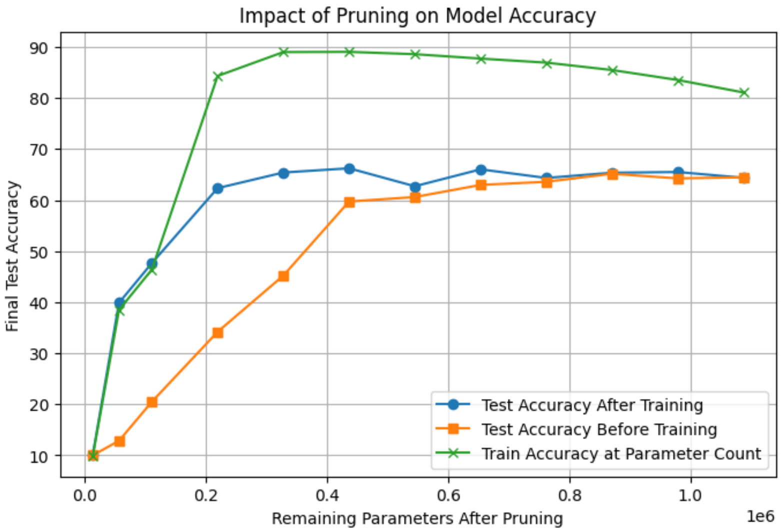

Figure 11: Test Accuracy of Model Iteratively Pruned(Unstructured) with Learning Rate of .8

When having a high learning rate in conjunction with iterative pruning and unstructured pruning being the method. We start seeing much better results than any of the other examples before. The sharp drop-off we are used to seeing early isn't happening until around 70 % of the parameters have been pruned and at 95 % of the parameters being removed, we observe a training and testing accuracy of 40 %, which is relatively impressive.

3.2.2 Learning rate: .01

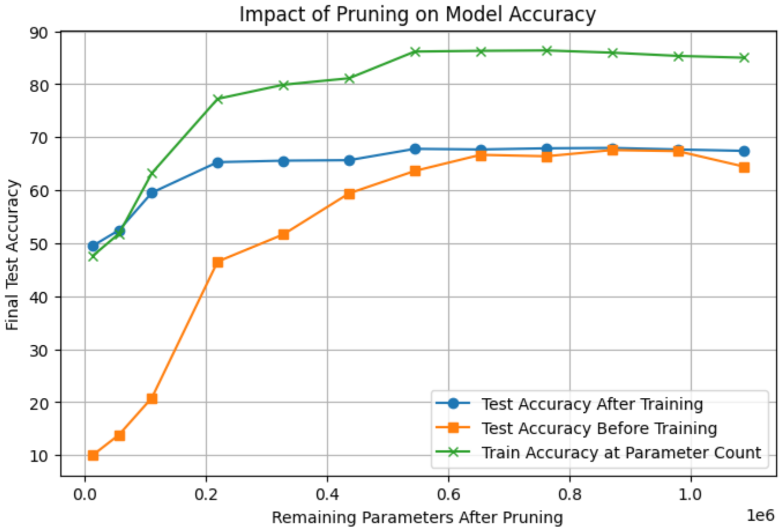

Figure 12: Test Accuracy of Model Iteratively Pruned(Unstructured) with Learning Rate of .01

The most impressive example out of all of them is this one. We observe an accuracy drop of 15 % after 99 % of parameters have been removed, while it was able to recover from a 10 % test accuracy before it was fine tuned. The model was overfitting until it had about 80 % of all parameters removed. Now this being said, I'm not entirely sure if what is being stated here is true. This is based on what I personally understand so far.

3.2.3 Learning rate: .0001

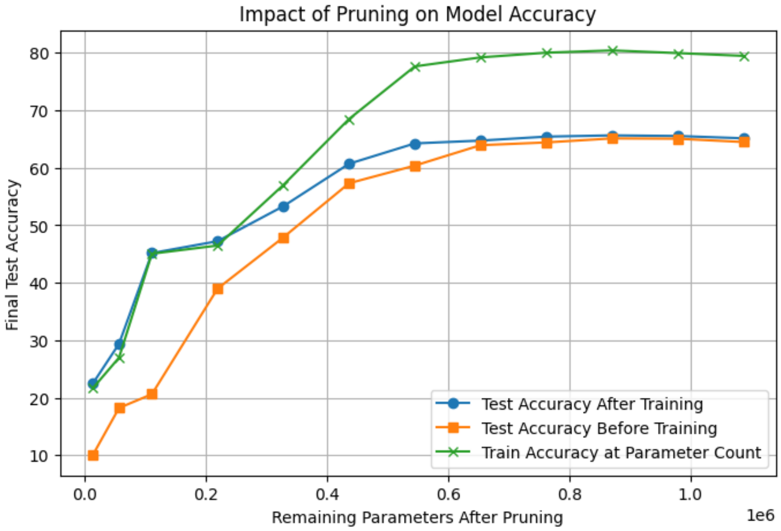

Figure 13: Test Accuracy of Model Iteratively Pruned(Unstructured) with Learning Rate of .0001

The same trend persists as we can note that there is a sweet spot learning rate as .8 crashes the accuracy and .0001 crashes it less. This learning rate shows us a slow decline in accuracy as previous models were unable to keep up with the loss of parameters it seems, as shown by the accuracies.

4 references

[1]Pruning tutorial¶. Pruning Tutorial - PyTorch Tutorials 2.6.0+cu124 documentation. (n.d.). https://pytorch.org/tutorials/intermediate/pruning_tutorial.html

[2]Torch.nn¶. torch.nn - PyTorch 2.6 documentation. (n.d.). https://pytorch.org/docs/stable/nn.html

[3]Zhao, K. (n.d.-a). Lecture18-Deep Learning Model Compression.pdf.