10605 Mini-Project Final Report

Jacob Rast
Biomedical Engineering Department
Carnegie Mellon University
`jrast@andrew.cmu.edu`

Tony Lee
Tepper School of Business
Carnegie Mellon University
`dongheel@tepper.cmu.edu`

1 Methodology

1.1 Magnitude-base pruning

Magnitude-based pruning is the simplest method that our team investigated and served as our baseline. It was inspired by Han et al [1]. Conceptually, it is straightforward: all weights less than a given magnitude are "pruned" or set to 0. As a rational, if a weight is very small it is likely that this contributes little or nothing to the final model output. One drawback of this approach is that fine tuning is difficult: TensorFlow does not allow for individual weights to be frozen, and any attempt at training a model after magnitude-based pruning will inevitably update the 0 valued weights.

1.2 Greedy layer-wise pruning

This method is a simplification of the ThiNet architecture [3]. We adopted the core idea of ThiNet, namely a structured (rather than unstructured as in magnitude-based pruning) pruning approach. Just as in ThiNet, an entire filter of a CNN is pruned. For ease of implementation and optimization, we defined a greedy objective function for pruning filters. Rather than evaluating the effect of pruning on the layer output or on the next layer output, we optimized for removing filters which would cause the smallest decrease in validation accuracy. Structured pruning, such as our greedy layer-wise pruning strategy, have the benefit of allowing for fine-tuning. The result of pruning a filter is a new model architecture, similar to the old but with one fewer filter layer in a CNN. This architecture can be explicitly defined using a large framework such as TensorFlow and subsequently trained. Using this paradigm, a high degree of sparsity can be achieved. Weights are iteratively pruned which is then followed by training.

1.3 Sparse Structure Selection

Implementation of 'Sparse Structure Selection' [2] for CNN pruning starts with initializing scaling factors. It then defines a sparse regularization term and uses a custom training loop to train a pruned model with a sparse regularization term. The pruning process involves iterating through Conv2D layers, computing L2-norms of grouped filters, generating binary masks, and pruning layer weights based on a threshold.

2 Comparison

Our best overall model score was achieved by a hybrid method. We recognized that magnitude-based pruning is a general good "polishing" strategy that will prune unimportant weights. As such, greedy layer-wise pruning was performed followed by magnitude-based pruning. This can be justified as greedy layer-wise pruning removing redundant filters and magnitude-base pruning removing unimportant weights from the filters which remain. The three methods considered had wildly differing degrees of complexity. The model which used fine-tuning required writing code to transfer weights from a small model to a mathematically equivalent sparse large model, which was a tricky task. For the sparse Structure Selection model, the model introduced additional hyperparameters that can impact both accuracy and sparsity. However, these hyperparameters also increased the complexity of the model and made implementation more challenging. Magnitude-based pruning, on the other hand, was trivial to implement. The full comparison of all methods can be found below in figure 1. The "Pareto Frontier" can be seen, which is the location where no model with equal accuracy has better sparsity and vice-versa. Note that all methods have hyperparameters which can be tuned, and the pruning performance across several hyperparameter values is shown for all.

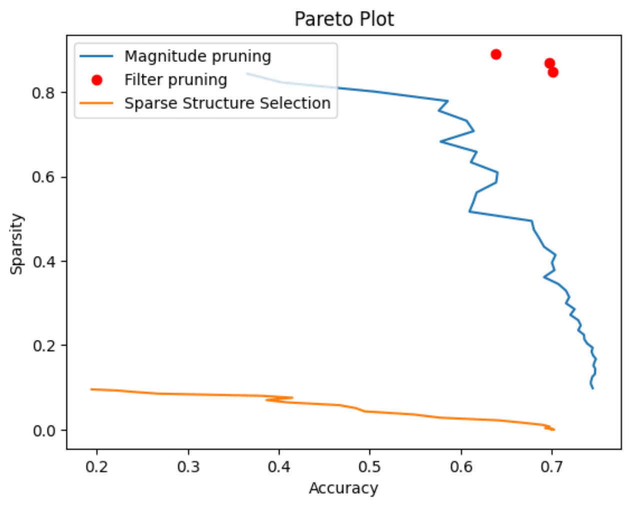

Figure 1: Pareto plot of sparsity vs. accuracy for all methods implemented

3 Reflection

The shocking effectiveness of magnitude-based tuning was a big takeaway for us in this project. Using a method that can be implemented in no more than 30 minutes the model 30% sparsity can be achieved with little-to-no loss of accuracy, and 60% sparsity with only a 5% decrease in accuracy. Ultimately, fine-tuning was a critical method to achieve our results. This highlights a weakness of magnitude-based tuning: while high model sparsity can be trivially achieved, this does not necessarily translate to lower model memory usage at inference time due to poor support for unstructured sparsity in traditional ML frameworks. Given some knowledge of neuroscience, the effectiveness of pruning is not surprising. While we traditionally think of learning as the acquisition of new knowledge it has been shown that removal of connection between neurons is critical for learning [4]. Given the neurological inspiration for these statistical models, it is, therefore, reasonable to assume that model performance can be improved by pruning unimportant features as well.

References

References follow the acknowledgments in the camera-ready paper. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. Note that the Reference section does not count towards the page limit.

[1] Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). *Learning both Weights and Connections for Efficient Neural Networks.*

[2] Huang, Z., & Wang, N. (2017). *Data-Driven Sparse Structure Selection for Deep Neural Networks.*

[3] Luo, J.-H., Wu, J., & Lin, W. (2017). *ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression.*

[4] Sakai, J. (2020). *Core Concept: How synaptic pruning shapes neural wiring during development and, possibly, in disease.* Proceedings of the National Academy of Sciences of the United States of America, 117(28), 16096–16099. doi:10.1073/pnas.2010281117