Model Compression and Network Pruning

Ethan Gaskin
Computational Biology Department
Carnegie Mellon University

Thomas Zhang
Computational Biology Department
Carnegie Mellon University

Jen Yi Wong
Computational Biology Department
Carnegie Mellon University

1 Methodology

We performed the following Neural Network (NN) Pruning methods:

- Magnitude Based Filter
- Network Slimming
- Multi-armed Bandit Pruning (MAB)

1.1 Magnitude Based Filter

Magnitude based filtering removes weights based on magnitude. This is done by first training a fully connected model and then removing all connections that fall below some threshold. This means a fully-connected layer is converted into a sparse layer [1]. The idea behind this is that training the fully connected model allows us to identify the connections that are important. Then the connections that have the smallest magnitude are removed because those contribute the least to the output of the next layer. To further increase sparsity, the model is retrained, and the retrained connections are once again removed based on whether they exceed some threshold.

1.2 Network Slimming

Network Slimming was pioneered by Liu et al. in 2017 [2]. Network Slimming is a pruning method that introduces an auxiliary training variable, called a scaling factor and denoted γ, for each channel in all the trainable layers of a NN; the output of each channel is multiplied by its own scaling factor. This is temporarily incorporated into model training by summing over a function of the scaling factors, multiplying this sum by a regularization term, and adding this to the loss for that layer. The function used by the original paper was the L1-norm of each gamma. After retraining a model with these additional scaling factors, Network Slimming prunes channels whose corresponding scaling factor falls below a global threshold. This global threshold is determined across all gamma values (across all layers).

1.3 MAB

MAB pruning uses the multi-armed bandit framework to decide which weights to set to 0. The specific algorithm used was the Upper Confidence Bound (UCB) algorithm, which performs optimistic exploration by selecting arms that maximizing the expected rewards [3]. For this use case, each channel in the NN is treated as an arm. Due to constraints on the computational resources, we focused on the output channels for the Dense Layer (Layer 13), and only considered each of these as an arm. Then we reduced the number of arms here. Pruning of each layer was tested, but it was found that because the Dense Layer accounts for the large majority of the parameters, pruning this layer was most effective. Hence, pruning of the NN focused mostly on zeroing weights in this layer.

2 Comparison

Table 1: Table showing the accuracy, sparsity and evaluation score for the best model for each method on the validation dataset.

Method	Accuracy	Sparsity	Evaluation Score
Magnitude Based Filter	0.852	0.722	0.787
Network Slimming	0.820	0.501	0.661
MAB	0.797	0.640	0.718

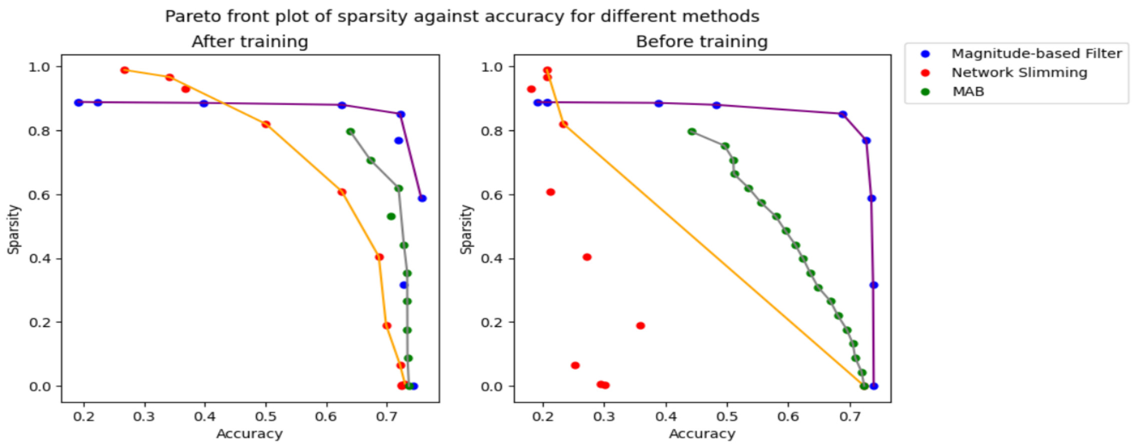

Figure 1: Evaluation metrics for each classifier tested on the 3 different mouse and human datasets.

From Fig. 1, it can be seen that the Magnitude Based Filter method performed the best in terms of sparsity to accuracy trade-off. After training, the accuracy for all models improved as the Pareto Frontier plots for each of the methods shifted right indicating that we were able to attain a higher amount of sparsity while retaining relatively high accuracy. The network slimming and MAB curves moving further to the right than the magnitude based pruning. This implies that the network slimming and MAB methods benefit more from training compared to the magnitude based pruning.

The optimal models for each method is shown in Table 1. These models were chosen based on the provided evaluation metric: $(\text{accuracy} + \frac{\text{num zero weights}}{\text{total parameters}})/2$. The models that have the highest evaluation metric correspond to the points closest to the top right of the plots.

The pruning method that led to the best performance score was magnitude based pruning. This was somewhat surprising since magnitude based pruning is a relatively simple pruning method. Magnitude based pruning only depends on an arbitrary threshold, hence there ability for this method to accommodate complex relationships between the layers would be limited.

We found that implementing the magnitude based pruning was the easiest to implement. To implement the magnitude based pruning method, we created a mask for each of the layers we wanted to prune. Each mask indicated the weights that exceeded the threshold. Then we used the mask to zero all the edge weights in the target layers that did not exceed some threshold. The masks were used to repeatedly remove edges during the retraining process until the final model was produced.

The MAB was the second-most difficult method to implement. Using the UCB algorithm, we rank the rewards from each arm and set the lowest ranked arms to 0. For this algorithm, since it is an optimistic exploration algorithm and assumes each arm has some underlying distribution, we select weights that maximize our reward and update the mean for each weight on each iteration. Each arm is pulled at least once to form the base case. Because each weight must be investigated at least once with a forward pass through the model, the runtime was significantly longer than the other methods.

Lastly, the network slimming method was the most difficult method to implement. As the original paper suggested, we were able to train scaling factors using Batch Normalization (BN) layers (which exist already in standard NN libraries). We temporarily added a BN layer after each layer with trainable parameters, then re-trained the model with these temporary layers. After re-training the model with these additional scaling factors, we then calculate a set global percentile threshold across all scaling factors and choose to prune channels whose scaling factor fell below the selected threshold, and removed the temporary BN layers to obtain our final Network Slimming pruned model. Note, since the first Dense layer contains so many parameters we based thresholding on this layer alone. This method was the most difficult to implement because it required us to map weights in one layer to weights in another layer, and to adjust the global selection of threshold since so many of the trainable parameters fell in the first Dense layer.

3 Reflections

The results did not match what we expected from pruning the model's weights. We had originally expected that pruning the majority of the weights in the NN would significantly adversely affect its performance due to the lack of contribution from the weights. However, the model performed well especially after additional training with the zeroed weights. This showed that the NN had a large difference in the importance of each weight, and ultimately the original model was likely wasting a large amount of resources by using the full set of parameters specified by the architecture and obtained from the original training. Based on our results here, the efficiency of the model could be improved via pruning since less weights would be stored (saving memory resources), which speeds up computation since less parameters means less calculations performed on input (saving time).

References

[1] Song Han et al. *Learning both Weights and Connections for Efficient Neural Networks*. 2015. arXiv: 1506.02626 [cs.ne]. URL: https://arxiv.org/abs/1506.02626.

[2] Zhuang Liu et al. *Learning Efficient Convolutional Networks through Network Slimming*. 2017. arXiv: 1708.06519 [cs.CV]. URL: https://doi.org/10.48550/arXiv.1708.06519.

[3] Salem Ameen and Sunil Vadera. "Pruning Neural Networks Using Multi-Armed Bandits". In: *The Computer Journal* 63.7 (Sept. 2019), pp. 1099–1108. ISSN: 0010-4620. DOI: 10.1093/comjnl/bxz078. eprint: https://academic.oup.com/comjnl/article-pdf/63/7/1099/33505995/bxz078.pdf. URL: https://doi.org/10.1093/comjnl/bxz078.