Rethinking the Value of Network Pruning

Seungmin Lee (profile2697@gmail.com; 2013-11420), Dept. of Computer Science and Engineering, Seoul National University

1. Introduction

Network pruning is a way of reducing the size of deep neural networks. Network pruning methods can be classified into two categories: unstructured pruning and structured pruning. Unstructured pruning prunes a network at the level of individual weights. Otherwise, structured pruning prunes a model at a higher level like channels.

The typical process of network pruning consists of three steps: First, train an over-parameterized neural network on a training dataset. Second, prune the pre-trained network by removing the weights or channels that have small values. Finally, fine-tuning the pruned network (also called target architecture) on the dataset. In this step, the pruned network is typically initialized using the remaining weights of the larger network.

This paper argues the last step is not significant in **structured pruning**. This claim means the performance of the target network using random initialization can recover the performance of the target network updated using fine-tuning. The paper demonstrated its thought using carefully designed experiments.

2. Experiments Setting

Network Architectures, Datasets and Pruning Methods The authors used ResNet, VGG, and DenseNet as the architectures for experiments. For datasets, they used CIFAR-10, CIFAR100, and ImageNet. For pruning methods, they chose four predefined structured methods (Li *et al*. [5], Luo *et al*. [7], He *et al*. [3], He *et al*. [2]), two automatic structured methods (Liu *et al*. [6], Huang *et al*. [4]) and one unstructured method (Han *et al*. [1]). For predefined structure methods, a human defines pruning rates of each layer. Otherwise, for automatic structured methods, the pruning algorithm automatically determines pruning rates of each layer. In other words, unlike predefined ones, pruning rates of each layer can be changed for each run.

Training Budget The authors used two types of training budgets: **Scratch-E** and **Scratch-B**. In Scratch-B, they used the same number of epochs for both models with random initialization and fine-tuning. However, the authors claimed the Scratch-B could be unfair to the pruned networks because the pruned models consume smaller computations for each epoch. Therefore, they also used Scratch-B in their experiments, which concerns FLOPs. For example, if a pruned model saved 2xFLOPs, the authors doubled the number of epochs.

3. Results

There are many numbers in this paper. In this section, I summarize the results that the results show. First, the Scratch-B almost always shows better results than the Scratch-E. When the Scratch-E shows better results, the margin between them was small. The authors practically always could achieve better or comparable performances for the structured pruning algorithms, often with large margins. Otherwise, for the unstructured pruning method, the authors also could make better performances (with small margins) but not as often as the experiments on structured ones. These results show initializing the weights with the over-parameterized models' weights is not critical in the structured pruning. Moreover, the results show that the found architecture could be more crucial than the typical pruning steps.

4. Personal Memo

The last step is not more than typical pre-training and fine-tuning. Therefore, if these results are accurate, what we should rethink is pre-training. If we could find the pre-training is not meaningful, it would be interesting. Anyway, even if the results are accurate, the last step in the process still has meaning since it is helpful to faster convergence.

References

[1] S. Han, et al. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 1

[2] Y. He, et al. Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI*, 2018. 1

[3] Y. He, et al. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 1

[4] Z. Huang et al. Learning efficient convolutional networks through network slimming. In *ECCV*, 2018. 1

[5] H. Li, et al. Pruning filters for efficient convnets. In *ICLR*, 2017. 1

[6] Z. Liu, et al. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 1

[7] J.-H. Luo, et al. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017. 1