Standardizing Evaluation of Neural Network Pruning

Jose Javier Gonzalez Ortiz
MIT CSAIL

Davis W. Blalock
MIT CSAIL

John V. Guttag
MIT CSAIL

Abstract

Neural network pruning consists of reducing the size of a network by removing parameters. In this work, we introduce ShrinkBench, an open-source library to facilitate standardized evaluation of neural network pruning methods. ShrinkBench simplifies using standardized datasets, pretrained models, and evaluation metrics for implementing pruning methods. In addition to describing the functionality of ShrinkBench, we demonstrate its utility by using it to implement and evaluate several pruning methods. We show that ShrinkBench's comprehensive evaluation can prevent common pitfalls when comparing pruning methods.

1 Introduction

Deep neural network inference often requires large amounts of computation and memory. *Pruning*, i.e. reducing the size of a network by removing parameters, is a popular approach for reducing these requirements. Typically, methods start with a large and accurate network, and then try to make it smaller with as little loss in accuracy as possible. Unfortunately, unlike in some other areas of machine learning, there is no widely agreed upon set of benchmarks or measurements that facilitate comparison of methods [1, 8]. Different papers use different datasets, prune different network architectures, and report performance in differing ways, making it almost impossible to know under what conditions one technique is better than another. In this work, we present ShrinkBench, an open-source library designed to make it easier for researchers to test pruning methods in a way that facilitates direct comparisons. In addition to describing ShrinkBench, we use it to examine some existing pruning heuristics. Our empirical findings demonstrate that a pruning method's performance can vary across datasets, networks, initial weights and reported metrics—confirming the need for standardized evaluation of pruning methods.

2 System Overview

ShrinkBench[1] is a Python library designed to ease evaluation of neural network pruning methods. ShrinkBench was designed with two goals in mind: 1) enable rapid prototyping of neural network pruning methods, and 2) facilitate the use of standardized metrics, datasets, network architectures, and finetuning setups. Using ShrinkBench, researchers can easily implement pruning strategies and evaluate their effectiveness across a wide range of scenarios. The library provides tools for running experiments using standardized dataset-model combinations and controlling for potentially confounding factors such as finetuning hyperparameters or initial weights. Results can then be aggregated to produce method comparisons like the ones presented in the results section. ShrinkBench works with off-the-shelf PyTorch[9] model architectures, simply requiring the user to provide parameter masks that indicate which parameters to keep, and optionally new values for the remaining weights. To facilitate the task of computing the set of parameter masks, ShrinkBench provides primitives for retrieving parameters, activations, and gradients. Since only the parameter masks are required, ShrinkBench supports arbitrary scoring functions, allocation of parameters across layers, and sparsity structures. From the parameter masks, ShrinkBench can train and fine-tune models ensuring that masked parameters do not contribute to the output of the network and are not updated during backpropagation. Given these masks, ShrinkBench will automatically apply the pruning, update the network according to a training or fine-tuning setup, and compute metrics across many models, datasets, random seeds, and compression ratios. Since the existing literature on pruning has mostly focused on computer vision tasks, ShrinkBench provides a set of vision-related dataset-model combinations with pretrained weights.

[1]Source code: https://github.com/shrinkbench

3 Baselines

We used ShrinkBench to implement several baseline pruning methods, both as examples of how to use our library and as baselines to which new methods can compare.

- Global Magnitude Pruning - prunes a fraction f of all the weights with the lowest absolute value.
- Layerwise Magnitude Pruning - for each layer, prunes a fraction f of the weights with the lowest absolute value.
- Global Gradient Magnitude Pruning - prunes a fraction f of all the weights with the lowest absolute value of weight times gradient, evaluated on a batch of inputs.
- Layerwise Gradient Magnitude Pruning - for each layer, prunes a fraction f of the weights with the lowest absolute value of weight times gradient, evaluated on a batch of inputs.
- Random - prunes a random fraction f of all the weights.

Magnitude-based pruning approaches are common baselines in the literature that have proven to be competitive with more complex methods [2–5]. Gradient-based methods are less common, but have recently gained popularity [6, 7, 10]. Random pruning is just a straw man approach. The proposed baselines are not faithful reproductions of any of the cited methods. We just employed similar pruning heuristics.

4 Results

We tested ShrinkBench by pruning over 800 networks using the described baselines and varying datasets, networks, compression ratios, initial weights, and random seeds. In doing so, we identified various pitfalls associated with experimental practices that are currently common in the literature but that can be avoided by using ShrinkBench's evaluation.

Metrics are not Interchangeable. It is a common practice to describe the amount of pruning by reporting either the reduction in the total number of parameters or in the number of theoretical FLOPs. If these metrics are sufficiently correlated, reporting only one is sufficient to characterize the effectiveness of a pruning method. Our experiments indicate that this not necessarily true. As Figure 1 shows, Global pruning methods are more accurate than Layerwise methods for a given model size, but Layerwise methods are more accurate for a given theoretical speedup. This discrepancy is related to the fact that pruning layers with larger inputs can result in higher computational savings.

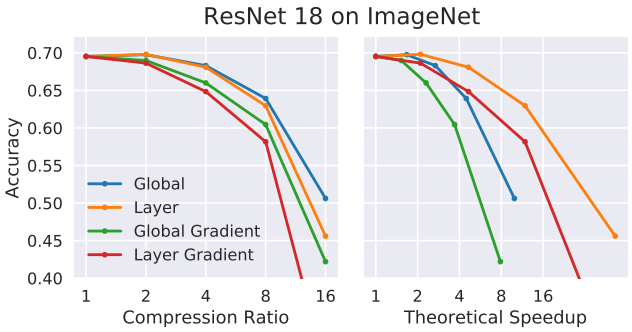

Figure 1. Accuracy for ResNet-18 on ImageNet for several compression ratios and their respective theoretical speedups.

Results Vary Across Models, Datasets, and Ratios Many methods report results on only a small number of datasets, models, and compression ratios. If the relative performance of different methods were constant across these factors, this would not be problematic. But they are not. Figure 2 shows accuracy for increasing compression ratios for CIFAR-VGG [11] and ResNet56 on CIFAR10. In general, Global methods are more accurate than Layerwise methods, magnitude-based methods are more accurate than gradient-based methods, and random performs worst of all. However, if one were only to look at CIFAR-VGG for compression ratios smaller than 10, one could conclude that Global Gradient outperforms all other methods. In contrast, for ResNet56, Global Gradient always underperforms compared to Global and Layerwise Magnitude Pruning. Moreover, we found that for some settings (such as Global Gradient, compression 16), different random seeds yielded significantly different accuracies (0.88 vs 0.61).

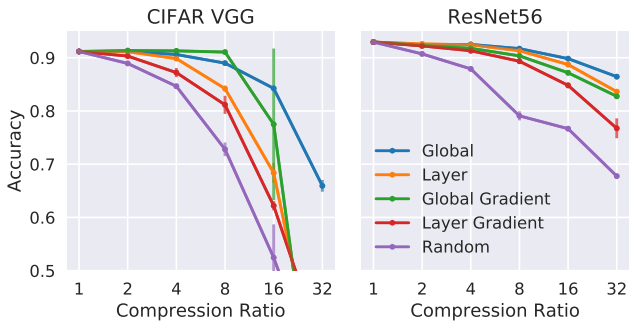

Figure 2. Top 1 Accuracy on CIFAR10 for several compression ratios. Global Gradient performs better than Global for CIFAR-VGG on low compression ratios, but worse otherwise. One standard deviation bars across three runs are included.

Using the Same Initial Model is Essential. Many papers report results on the same neural network architecture whilst using different initial models (sets of pretrained weights). Using ShrinkBench, we found that initial model can be a significant confounding factor when comparing methods. We trained two ResNet56 models from scratch using Adam until convergence with $\eta = 10^{-3}$ and $\eta = 10^{-4}$. For brevity, we will refer to these sets of pretrained weights as weights A and B respectively. Figure 3 shows pruning results (after fine-tuning) for each set set of weights for both Global and Layerwise Magnitude Pruning. With pretrained weights A the methods seem comparable in terms of accuracy. On the other hand, for pretrained weights B, Global is more accurate for higher compression ratios. For both, Global strictly outperforms Layerwise Magnitude Pruning, suggesting it is a better approach. However, had we only compared Layerwise pruning with pretrained weights A to Global with pretrained weights B, we would have reached a different conclusion.

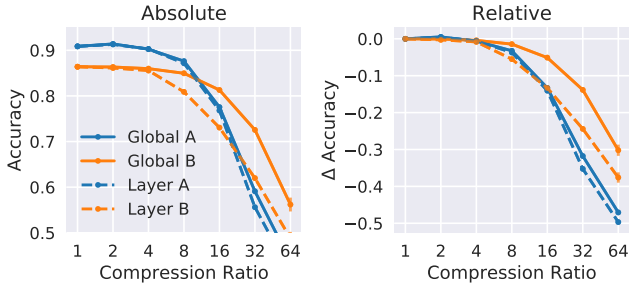

Figure 3. Global and Layerwise Magnitude Pruning on two different ResNet-56 models.

All these results evidence the need for standardized experiments when evaluating neural network pruning methods. With ShrinkBench we aim to provide a toolkit that facilitates the task of producing comprehensive and reproducible evaluations of proposed pruning approaches.

References

[1] Bin Dai, Chen Zhu, and David Wipf. 2018. Compressing neural networks using the variational information bottleneck. *arXiv preprint arXiv:1802.10399* (2018).

[2] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. 2019. The Lottery Ticket Hypothesis at Scale. *arXiv preprint arXiv:1903.01611* (2019).

[3] Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The State of Sparsity in Deep Neural Networks. arXiv:cs.LG/1902.09574

[4] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[5] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*. 1135–1143.

[6] Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip H. S. Torr. 2019. A Signal Propagation Perspective for Pruning Neural Networks at Initialization. *arXiv:1906.06307 [cs, stat]* (June 2019). http://arxiv.org/abs/1906.06307 arXiv: 1906.06307.

[7] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. SNIP: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340* (2018).

[8] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270* (2018).

[9] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[10] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. 2018. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9194–9203.

[11] Sergey Zagoruyko. 2015. 92.45% on CIFAR-10 in Torch. https://torch.ch/blog/2015/07/30/cifar.html. Accessed: 2019-07-22.