WHAT IS THE STATE OF NEURAL NETWORK PRUNING?

Davis Blalock [*1] **Jose Javier Gonzalez Ortiz** [*1] **Jonathan Frankle** [1] **John Guttag** [1]

ABSTRACT

Neural network pruning—the task of reducing the size of a network by removing parameters—has been the subject of a great deal of work in recent years. We provide a meta-analysis of the literature, including an overview of approaches to pruning and consistent findings in the literature. After aggregating results across 81 papers and pruning hundreds of models in controlled conditions, our clearest finding is that the community suffers from a lack of standardized benchmarks and metrics. This deficiency is substantial enough that it is hard to compare pruning techniques to one another or determine how much progress the field has made over the past three decades. To address this situation, we identify issues with current practices, suggest concrete remedies, and introduce ShrinkBench, an open-source framework to facilitate standardized evaluations of pruning methods. We use ShrinkBench to compare various pruning techniques and show that its comprehensive evaluation can prevent common pitfalls when comparing pruning methods.

1 INTRODUCTION

Much of the progress in machine learning in the past decade has been a result of deep neural networks. Many of these networks, particularly those that perform the best (Huang et al., 2018), require enormous amounts of computation and memory. These requirements not only increase infrastructure costs, but also make deployment of networks to resource-constrained environments such as mobile phones or smart devices challenging (Han et al., 2015; Sze et al., 2017; Yang et al., 2017).

One popular approach for reducing these resource requirements at test time is neural network *pruning*, which entails systematically removing parameters from an existing network. Typically, the initial network is large and accurate, and the goal is to produce a smaller network with similar accuracy. Pruning has been used since the late 1980s (Janowsky, 1989; Mozer & Smolensky, 1989a;b; Karnin, 1990), but has seen an explosion of interest in the past decade thanks to the rise of deep neural networks.

For this study, we surveyed 81 recent papers on pruning in the hopes of extracting practical lessons for the broader community. For example: which technique achieves the best accuracy/efficiency tradeoff? Are there strategies that work best on specific architectures or datasets? Which high-level design choices are most effective?

There are indeed several consistent results: pruning parameters based on their magnitudes substantially compresses networks without reducing accuracy, and many pruning methods outperform random pruning. However, our central finding is that the state of the literature is such that our motivating questions are impossible to answer. Few papers compare to one another, and methodologies are so inconsistent between papers that we could not make these comparisons ourselves. For example, a quarter of papers compare to no other pruning method, half of papers compare to at most one other method, and dozens of methods have never been compared to by any subsequent work. In addition, no dataset/network pair appears in even a third of papers, evaluation metrics differ widely, and hyperparameters and other counfounders vary or are left unspecified.

Most of these issues stem from the absence of standard datasets, networks, metrics, and experimental practices. To help enable more comparable pruning research, we identify specific impediments and pitfalls, recommend best practices, and introduce ShrinkBench, a library for standardized evaluation of pruning. ShrinkBench makes it easy to adhere to the best practices we identify, largely by providing a standardized collection of pruning primitives, models, datasets, and training routines.

Our contributions are as follows:

1. A meta-analysis of the neural network pruning literature based on comprehensively aggregating reported results from 81 papers.
2. A catalog of problems in the literature and best practices for avoiding them. These insights derive from analyzing existing work and pruning hundreds of models.
3. ShrinkBench, an open-source library for evaluating neural network pruning methods available at `https://github.com/jjgo/shrinkbench`.

[*]Equal contribution [1]MIT CSAIL, Cambridge, MA, USA. Correspondence to: Davis Blalock <dblalock@mit.edu>.

Proceedings of the 3^{rd} MLSys Conference, Austin, TX, USA, 2020.

2 Overview of Pruning

Before proceeding, we first offer some background on neural network pruning and a high-level overview of how existing pruning methods typically work.

2.1 Definitions

We define a neural network *architecture* as a function family $f(x;\cdot)$. The architecture consists of the configuration of the network's parameters and the sets of operations it uses to produce outputs from inputs, including the arrangement of parameters into convolutions, activation functions, pooling, batch normalization, etc. Example architectures include AlexNet and ResNet-56. We define a neural network *model* as a particular parameterization of an architecture, i.e., $f(x;W)$ for specific parameters W. Neural network *pruning* entails taking as input a model $f(x;W)$ and producing a new model $f(x;M\odot W')$. Here W' is set of parameters that may be different from W, $M\in\{0,1\}^{|W'|}$ is a binary mask that fixes certain parameters to 0, and $\odot$ is the elementwise product operator. In practice, rather than using an explicit mask, pruned parameters of W are fixed to zero or removed entirely.

2.2 High-Level Algorithm

There are many methods of producing a pruned model $f(x;M\odot W')$ from an initially untrained model $f(x;W_0)$, where W_0 is sampled from an initialization distribution $\mathcal{D}$. Nearly all neural network pruning strategies in our survey derive from Algorithm 1 (Han et al., 2015). In this algorithm, the network is first trained to convergence. Afterwards, each parameter or structural element in the network is issued a score, and the network is pruned based on these scores. Pruning reduces the accuracy of the network, so it is trained further (known as *fine-tuning*) to recover. The process of pruning and fine-tuning is often iterated several times, gradually reducing the network's size.

Many papers propose slight variations of this algorithm. For example, some papers prune periodically during training (Gale et al., 2019) or even at initialization (Lee et al., 2019b). Others modify the network to explicitly include additional parameters that encourage sparsity and serve as a basis for scoring the network after training (Molchanov et al., 2017).

2.3 Differences Betweeen Pruning Methods

Within the framework of Algorithm 1, pruning methods vary primarily in their choices regarding sparsity structure, scoring, scheduling, and fine-tuning.

Structure. Some methods prune individual parameters (*unstructured pruning*). Doing so produces a sparse neural network, which—although smaller in terms of parameter-count—may not be arranged in a fashion conducive to speedups using modern libraries and hardware. Other methods consider parameters in groups (*structured pruning*), removing entire neurons, filters, or channels to exploit hardware and software optimized for dense computation (Li et al., 2016; He et al., 2017).

Algorithm 1 Pruning and Fine-Tuning

Input: N, the number of iterations of pruning, and X, the dataset on which to train and fine-tune

$W \leftarrow initialize()$
$W \leftarrow trainToConvergence(f(X;W))$
$M \leftarrow 1^{|W|}$
for i in 1 to N **do**
  $M \leftarrow prune(M, score(W))$
  $W \leftarrow fineTune(f(X;M\odot W))$
end for
return M, W

Scoring. It is common to score parameters based on their absolute values, trained importance coefficients, or contributions to network activations or gradients. Some pruning methods compare scores locally, pruning a fraction of the parameters with the lowest scores within each structural subcomponent of the network (e.g., layers) (Han et al., 2015). Others consider scores globally, comparing scores to one another irrespective of the part of the network in which the parameter resides (Lee et al., 2019b; Frankle & Carbin, 2019).

Scheduling. Pruning methods differ in the amount of the network to prune at each step. Some methods prune all desired weights at once in a single step (Liu et al., 2019). Others prune a fixed fraction of the network iteratively over several steps (Han et al., 2015) or vary the rate of pruning according to a more complex function (Gale et al., 2019).

Fine-tuning. For methods that involve fine-tuning, it is most common to continue to train the network using the trained weights from before pruning. Alternative proposals include rewinding the network to an earlier state (Frankle et al., 2019) and reinitializing the network entirely (Liu et al., 2019).

2.4 Evaluating Pruning

Pruning can accomplish many different goals, including reducing the storage footprint of the neural network, the computational cost of inference, the energy requirements of inference, etc. Each of these goals favors different design choices and requires different evaluation metrics. For example, when reducing the storage footprint of the network, all parameters can be treated equally, meaning one should evaluate the overall compression ratio achieved by pruning. However, when reducing the computational cost of

inference, different parameters may have different impacts. For instance, in convolutional layers, filters applied to spatially larger inputs are associated with more computation than those applied to smaller inputs.

Regardless of the goal, pruning imposes a tradeoff between model efficiency and quality, with pruning increasing the former while (typically) decreasing the latter. This means that a pruning method is best characterized not by a single model it has pruned, but by a family of models corresponding to different points on the efficiency-quality curve. To quantify efficiency, most papers report at least one of two metrics. The first is the number of multiply-adds (often referred to as FLOPs) required to perform inference with the pruned network. The second is the fraction of parameters pruned. To measure quality, nearly all papers report changes in Top-1 or Top-5 image classification accuracy.

As others have noted (Lebedev et al., 2014; Figurnov et al., 2016; Louizos et al., 2017; Yang et al., 2017; Han et al., 2015; Kim et al., 2015; Wen et al., 2016; Luo et al., 2017; He et al., 2018b), these metrics are far from perfect. Parameter and FLOP counts are a loose proxy for real-world latency, throughout, memory usage, and power consumption. Similarly, image classification is only one of the countless tasks to which neural networks have been applied. However, because the overwhelming majority of papers in our corpus focus on these metrics, our meta-analysis necessarily does as well.

3 Lessons from the Literature

After aggregating results from a corpus of 81 papers, we identified a number of consistent findings. In this section, we provide an overview of our corpus and then discuss these findings.

3.1 Papers Used in Our Analysis

Our corpus consists of 79 pruning papers published since 2010 and two classic papers (LeCun et al., 1990; Hassibi et al., 1993) that have been compared to by a number of recent methods. We selected these papers by identifying popular papers in the literature and what cites them, systematically searching through conference proceedings, and tracing the directed graph of comparisons between pruning papers. This last procedure results in the property that, barring oversights on our part, there is no pruning paper in our corpus that compares to any pruning paper outside of our corpus. Additional details about our corpus and its construction can be found in Appendix A.

3.2 How Effective is Pruning?

One of the clearest findings about pruning is that it works. More precisely, there are various methods that can significantly compress models with little or no loss of accuracy. In fact, for small amounts of compression, pruning can sometimes *increase* accuracy (Han et al., 2015; Suzuki et al., 2018). This basic finding has been replicated in a large fraction of the papers in our corpus.

Along the same lines, it has been repeatedly shown that, at least for large amounts of pruning, many pruning methods outperform random pruning (Yu et al., 2018; Gale et al., 2019; Frankle et al., 2019; Mariet & Sra, 2015; Suau et al., 2018; He et al., 2017). Interestingly, this does not always hold for small amounts of pruning (Morcos et al., 2019). Similarly, pruning all layers uniformly tends to perform worse than intelligently allocating parameters to different layers (Gale et al., 2019; Han et al., 2015; Li et al., 2016; Molchanov et al., 2016; Luo et al., 2017) or pruning globally (Lee et al., 2019b; Frankle & Carbin, 2019). Lastly, when holding the number of fine-tuning iterations constant, many methods produce pruned models that outperform retraining from scratch with the same sparsity pattern (Zhang et al., 2015; Yu et al., 2018; Louizos et al., 2017; He et al., 2017; Luo et al., 2017; Frankle & Carbin, 2019) (at least with a large enough amount of pruning (Suau et al., 2018)). Retraining from scratch in this context means training a fresh, randomly-initialized model with all weights clamped to zero throughout training, except those that are nonzero in the pruned model.

Another consistent finding is that sparse models tend to outperform dense ones for a fixed number of parameters. Lee et al. (2019a) show that increasing the nominal size of ResNet-20 on CIFAR-10 while sparsifying to hold the number of parameters constant decreases the error rate. Kalchbrenner et al. (2018) obtain a similar result for audio synthesis, as do Gray et al. (2017) for a variety of additional tasks across various domains. Perhaps most compelling of all are the many results, including in Figure 1, showing that pruned models can obtain higher accuracies than the original models from which they are derived. This demonstrates that sparse models can not only outperform dense counterparts with the same number of parameters, but sometimes dense models with even more parameters.

3.3 Pruning vs Architecture Changes

One current unknown about pruning is how effective it tends to be relative to simply using a more efficient architecture. These options are not mutually exclusive, but it may be useful in guiding one's research or development efforts to know which choice is likely to have the larger impact. Along similar lines, it is unclear how pruned models from different architectures compare to one another—i.e., to what extent does pruning offer similar benefits across architectures? To address these questions, we plotted the reported accuracies and compression/speedup levels of pruned models on ImageNet alongside the same metrics

for different architectures with no pruning (Figure 1).[1] We plot results within a family of models as a single curve.[2]

Figure 1 suggests several conclusions. First, it reinforces the conclusion that pruning can improve the time or space vs accuracy tradeoff of a given architecture, sometimes even increasing the accuracy. Second, it suggests that pruning generally does not help as much as switching to a better architecture. Finally, it suggests that pruning is more effective for architectures that are less efficient to begin with.

4 Missing Controlled Comparisons

While there do appear to be a few general and consistent findings in the pruning literature (see the previous section), by far the clearest takeaway is that pruning papers rarely make direct and controlled comparisons to existing methods. This lack of comparisons stems largely from a lack of experimental standardization and the resulting fragmentation in reported results. This fragmentation makes it difficult for even the most committed authors to compare to more than a few existing methods.

4.1 Omission of Comparison

Many papers claim to advance the state of the art, but don't compare to other methods—including many published ones—that make the same claim.

Ignoring Pre-2010s Methods There was already a rich body of work on neural network pruning by the mid 1990s (see, e.g., Reed's survey (Reed, 1993)), which has been almost completely ignored except for Lecun's Optimal Brain Damage (LeCun et al., 1990) and Hassibi's Optimal Brain Surgeon (Hassibi et al., 1993). Indeed, multiple authors have rediscovered existing methods or aspects thereof, with Han et al. (2015) reintroducing the magnitude-based pruning of Janowsky (1989), Lee et al. (2019b) reintroducing the saliency heuristic of Mozer & Smolensky (1989a), and He et al. (2018a) reintroducing the practice of "reviving" previously pruned weights described in Tresp et al. (1997).

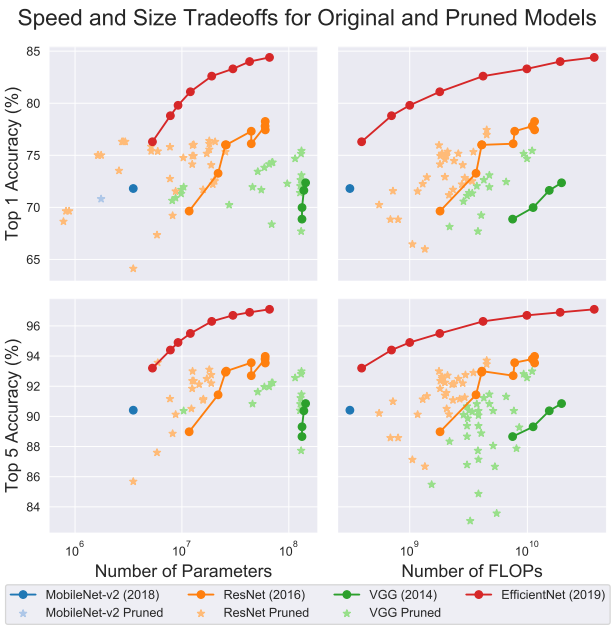

Figure 1: Size and speed vs accuracy tradeoffs for different pruning methods and families of architectures. Pruned models sometimes outperform the original architecture, but rarely outperform a better architecture.

Ignoring Recent Methods Even when considering only post-2010 approaches, there are still virtually no methods that have been shown to outperform all existing "state-of-the-art" methods. This follows from the fact, depicted in the top plot of Figure 2, that there are dozens of modern papers—including many affirmed through peer review—that have never been compared to by any later study.

A related problem is that papers tend to compare to few existing methods. In the lower plot of Figure 2, we see that more than a fourth of our corpus does not compare to any previously proposed pruning method, and another fourth compares to only one. Nearly all papers compare to three or fewer. This might be adequate if there were a clear progression of methods with one or two "best" methods at any given time, but this is not the case.

4.2 Dataset and Architecture Fragmentation

Among 81 papers, we found results using 49 datasets, 132 architectures, and 195 (dataset, architecture) combinations. As shown in Table 1, even the most common combination of dataset and architecture—VGG-16 on ImageNet[3] (Deng et al., 2009)—is used in only 22 out of 81 papers. Moreover, three of the top six most common combinations involve MNIST (LeCun et al., 1998a). As Gale et al. (2019) and others have argued, using larger datasets and models is essential when assessing how well a method works for real-

[1] Since many pruning papers report only change in accuracy or amount of pruning, without giving baseline numbers, we normalize all pruning results to have accuracies and model sizes/FLOPs as if they had begun with the same model. Concretely, this means multiplying the reported fraction of pruned size/FLOPs by a standardized initial value. This value is set to the median initial size or number of FLOPs reported for that architecture across all papers. This normalization scheme is not perfect, but does help control for different methods beginning with different baseline accuracies.

[2] The EfficientNet family is given explicitly in the original paper (Tan & Le, 2019), the ResNet family consists of ResNet-18, ResNet-34, ResNet-50, etc., and the VGG family consists of VGG-{11, 13, 16, 19}. There are no pruned EfficientNets since EfficientNet was published too recently. Results for non-pruned models are taken from (Tan & Le, 2019) and (Bianco et al., 2018).

[3] We adopt the common practice of referring to the ILSVRC2012 training and validation sets as "ImageNet."

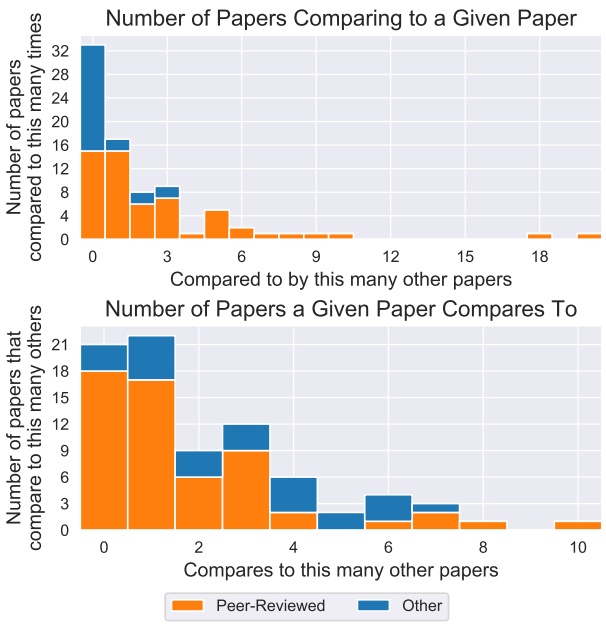

Figure 2: Reported comparisons between papers.

world networks. MNIST results may be particularly unlikely to generalize, since this dataset differs significantly from other popular datasets for image classification. In particular, its images are grayscale, composed mostly of zeros, and possible to classify with over 99% accuracy using simple models (LeCun et al., 1998b).

4.3 Metrics Fragmentation

As depicted in Figure 3, papers report a wide variety of metrics and operating points, making it difficult to compare results. Each column in this figure is one (dataset, architecture) combination taken from the four most common combinations[4], excluding results on MNIST. Each row is one pair of metrics. Each curve is the efficiency vs accuracy tradeoff obtained by one method.[5] Methods are color-coded by year.

It is hard to identify any consistent trends in these plots, aside from the existence of a tradeoff between efficiency and accuracy. A given method is only present in a small subset of plots. Methods from later years do not consistently outperform methods from earlier years. Methods within a plot are often incomparable because they report results at different points on the x-axis. Even when methods are nearby on the x-axis, it is not clear whether one meaningfully outperforms another since neither reports a standard deviation or other measure of central tendency. Finally, most papers in our corpus do not report any results with any of these common configurations.

[4]We combined the results for AlexNet and CaffeNet, which is a slightly modified version of AlexNet (caf, 2016), since many authors refer to the latter as "AlexNet," and it is often unclear which model was used.

[5]Since what counts as one method can be unclear, we consider all results from one paper to be one method except when two or more named methods within the paper report using at least one identical x-coordinate (i.e., when the paper's results can't be plotted as one curve).

(Dataset, Architecture) Pair		Number of Papers using Pair
ImageNet	VGG-16	22
ImageNet	ResNet-50	15
MNIST	LeNet-5-Caffe	14
CIFAR-10	ResNet-56	14
MNIST	LeNet-300-100	12
MNIST	LeNet-5	11
ImageNet	CaffeNet	10
CIFAR-10	CIFAR-VGG (Torch)	8
ImageNet	AlexNet	8
ImageNet	ResNet-18	6
ImageNet	ResNet-34	6
CIFAR-10	ResNet-110	5
CIFAR-10	PreResNet-164	4
CIFAR-10	ResNet-32	4

Table 1: All combinations of dataset and architecture used in at least 4 out of 81 papers.

4.4 Incomplete Characterization of Results

If all papers reported a wide range of points in their tradeoff curves across a large set of models and datasets, there might be some number of direct comparisons possible between any given pair of methods. As we see in the upper half of Figure 4, however, most papers use at most three (dataset, architecture) pairs; and as we see in the lower half, they use at most three—and often just one—point to characterize each curve. Combined with the fragmentation in experimental choices, this means that different methods' results are rarely directly comparable. Note that the lower half restricts results to the four most common (dataset, architecture) pairs.

4.5 Confounding Variables

Even when comparisons include the same datasets, models, metrics, and operating points, other confounding variables still make meaningful comparisons difficult. Some variables of particular interest include:

- Accuracy and efficiency of the initial model
- Data augmentation and preprocessing
- Random variations in initialization, training, and fine-tuning. This includes choice of optimizer, hyperparameters, and learning rate schedule.
- Pruning and fine-tuning schedule
- Deep learning library. Different libraries are known to

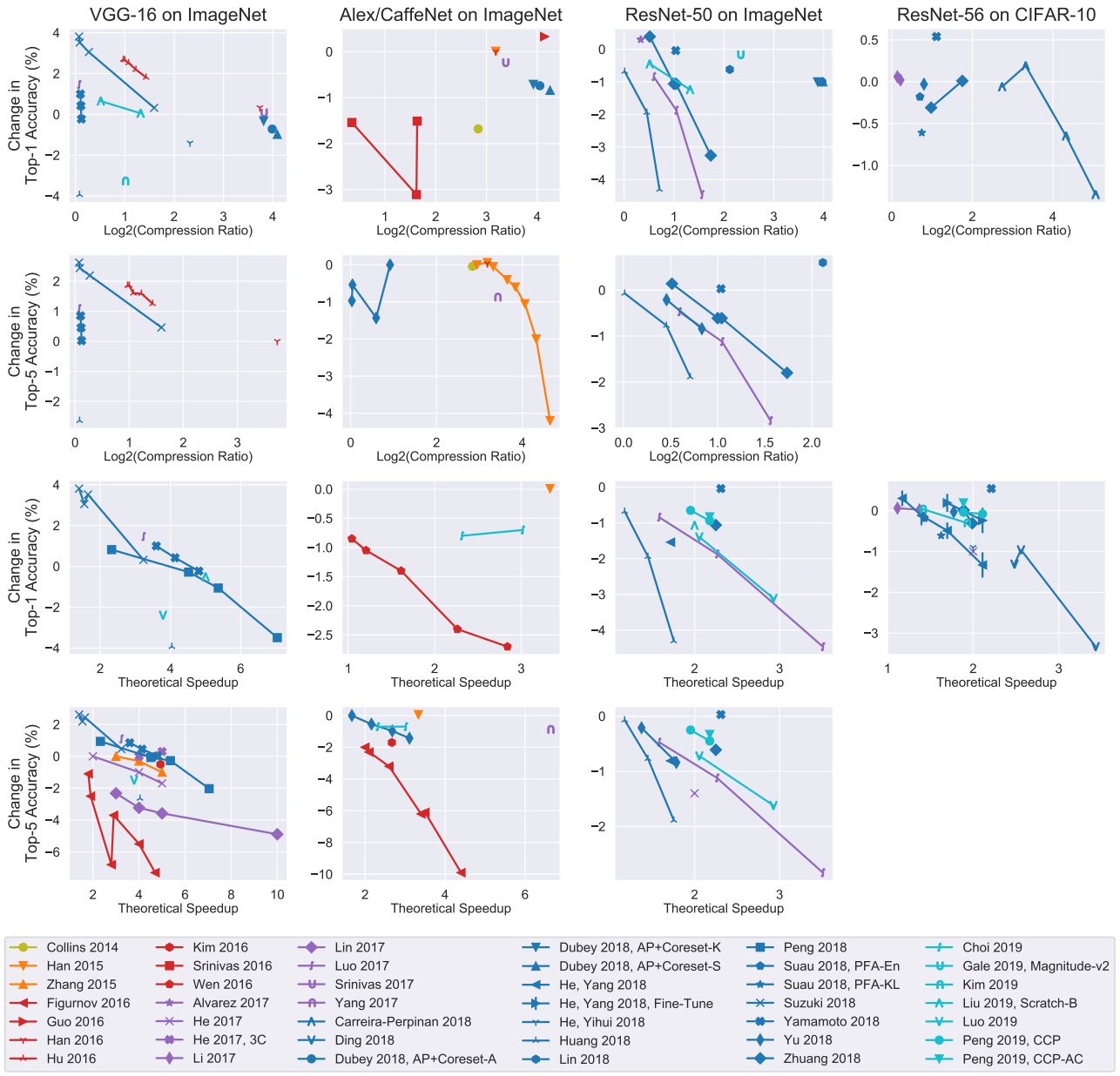

Figure 3: Fragmentation of results. Shown are all self-reported results on the most common (dataset, architecture) combinations. Each column is one combination, each row shares an accuracy metric (y-axis), and pairs of rows share a compression metric (x-axis). Up and to the right is always better. Standard deviations are shown for He 2018 on CIFAR-10, which is the only result that provides any measure of central tendency. As suggested by the legend, only 37 out of the 81 papers in our corpus report any results using any of these configurations.

yield different accuracies for the same architecture and dataset (Northcutt, 2019; Nola, 2016) and may have subtly different behaviors (Vryniotis, 2018).

- Subtle differences in code and environment that may not be easily attributable to any of the above variations (Crall, 2018; Jogeshwar, 2017; unr, 2017).

In general, it is not clear that any paper can succeed in accounting for all of these confounders unless that paper has both used the same code as the methods to which it compares and reports enough measurements to average out random variations. This is exceptionally rare, with Gale et al. (2019) and Liu et al. (2019) being arguably the only examples. Moreover, neither of these papers introduce novel pruning methods *per se* but are instead inquiries into the efficacy of existing methods.

Many papers attempt to account for subsets of these confounding variables. A near universal practice in this re-

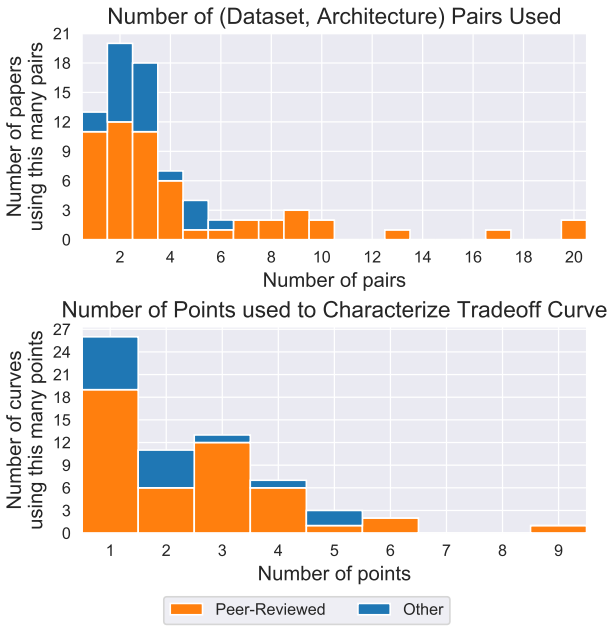

Figure 4: Number of results reported by each paper, excluding MNIST. *Top)* Most papers report on three or fewer (dataset, architecture) pairs. *Bottom)* For each pair used, most papers characterize their tradeoff between amount of pruning and accuracy using a single point in the efficiency vs accuracy curve. In both plots, the pattern holds even for peer-reviewed papers.

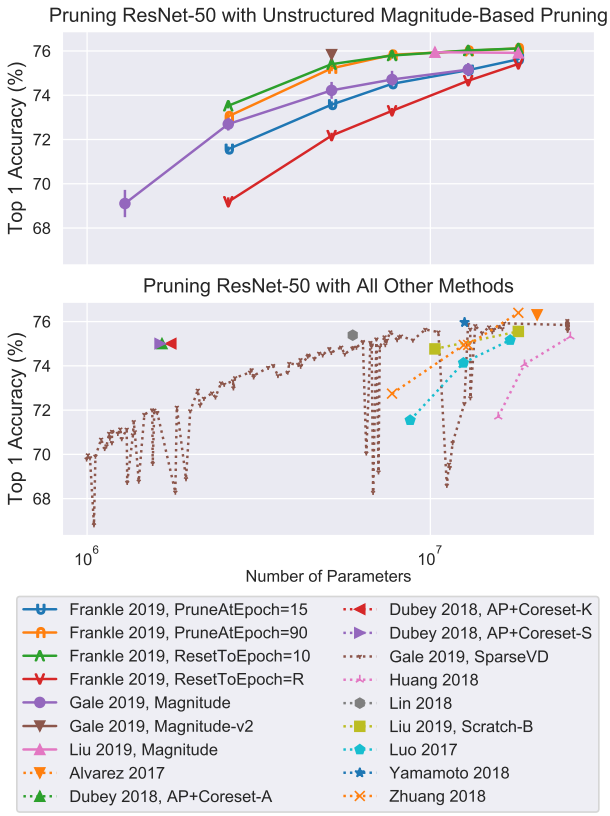

Figure 5: Pruning ResNet-50 on ImageNet. Methods in the upper plot all prune weights with the smallest magnitudes, but differ in implementation, pruning schedule, and fine-tuning. The variation caused by these variables is similar to the variation across different pruning methods, whose results are shown in the lower plot. All results are taken from the original papers.

gard is reporting change in accuracy relative to the original model, in addition to or instead of raw accuracy. This helps to control for the accuracy of the initial model. However, as we demonstrate in Section 7, this is not sufficient to remove initial model as a confounder. Certain initial models can be pruned more or less efficiently, in terms of the accuracy vs compression tradeoff. This holds true even with identical pruning methods and all other variables held constant.

There are at least two more empirical reasons to believe that confounding variables can have a significant impact. First, as one can observe in Figure 3, methods often introduce changes in accuracy of much less than 1% at reported operating points. This means that, even if confounders have only a tiny impact on accuracy, they can still have a large impact on which method appears better.

Second, as shown in Figure 5, existing results demonstrate that different training and fine-tuning settings can yield nearly as much variability as different methods. Specifically, consider 1) the variability introduced by different fine-tuning methods for unstructured magnitude-based pruning (Figure 6 top) and 2) the variability introduced by entirely different pruning methods (Figure 6 bottom). The variability between fine-tuning methods is nearly as large as the variability between pruning methods.

5 Further Barriers to Comparison

In the previous section, we discussed the fragmentation of datasets, models, metrics, operating points, and experimental details, and how this fragmentation makes evaluating the efficacy of individual pruning methods difficult. In this section, we argue that there are additional barriers to comparing methods that stem from common practices in how methods and results are presented.

5.1 Architecture Ambiguity

It is often difficult, or even impossible, to identify the exact architecture that authors used. Perhaps the most prevalent example of this is when authors report using some sort of ResNet (He et al., 2016a;b). Because there are two different variations of ResNets, introduced in these two papers, saying that one used a "ResNet-50" is insufficient to identify a particular architecture. Some authors do appear to deliberately point out the type of ResNet they use (e.g., (Liu et al., 2017; Dong et al., 2017)). However, given that few papers

even hint at the possibility of confusion, it seems unlikely that all authors are even aware of the ambiguity, let alone that they have cited the corresponding paper in all cases.

Perhaps the greatest confusion is over VGG networks (Simonyan & Zisserman, 2014). Many papers describe experimenting on "VGG-16," "VGG," or "VGGNet," suggesting a standard and well-known architecture. In many cases, what is actually used is a custom variation of some VGG model, with removed fully-connected layers (Changpinyo et al., 2017; Luo et al., 2017), smaller fully-connected layers (Lee et al., 2019b), or added dropout or batchnorm (Liu et al., 2017; Lee et al., 2019b; Peng et al., 2018; Molchanov et al., 2017; Ding et al., 2018; Suau et al., 2018).

In some cases, papers simply fail to make clear what model they used (even for non-VGG architectures). For example, one paper just states that their segmentation model "is composed from an inception-like network branch and a DenseNet network branch." Another paper attributes their VGGNet to (Parkhi et al., 2015), which mentions three VGG networks. Liu et al. (2019) and Frankle & Carbin (2019) have circular references to one another that can no longer be resolved because of simultaneous revisions. One paper mentions using a "VGG-S" from the Caffe Model Zoo, but as of this writing, no model with this name exists there. Perhaps the most confusing case is the Lenet-5-Caffe reported in one 2017 paper. The authors are to be commended for explicitly stating not only that they use Lenet-5-Caffe, but their exact architecture. However, they describe an architecture with an 800-unit fully-connected layer, while examination of both the Caffe `.prototxt` files (Jia et al., 2015a;b) and associated blog post (Jia et al., 2016) indicates that no such layer exists in Lenet-5-Caffe.

5.2 Metrics Ambiguity

It can also be difficult to know what the reported metrics mean. For example, many papers include a metric along the lines of "Pruned%". In some cases, this means fraction of the parameters or FLOPs remaining (Suau et al., 2018). In other cases, it means the fraction of parameters or FLOPs removed (Han et al., 2015; Lebedev & Lempitsky, 2016; Yao et al., 2018). There is also widespread misuse of the term "compression ratio," which the compression literature has long used to mean $\frac{\text{original size}}{\text{compressed size}}$ (Siedelmann et al., 2015; Zukowski et al., 2006; Zhao et al., 2015; Lindstrom, 2014; Ratanaworabhan et al., 2006; Blalock et al., 2018), but many pruning authors define (usually without making the formula explicit) as $1 - \frac{\text{compressed size}}{\text{original size}}$.

Reported "speedup" values present similar challenges. These values are sometimes wall time, sometimes original number of FLOPs divided by pruned number of FLOPs, sometimes a more complex formula relating these two quantities (Dong et al., 2017; He et al., 2018a), and sometimes never made clear. Even when reporting FLOPs, which is nominally a consistent metric, different authors measure it differently (e.g., (Molchanov et al., 2016) vs (Wang & Cheng, 2016)), though most often papers entirely omit their formula for computing FLOPs. We found up to a factor of four variation in the reported FLOPs of different papers for the same architecture and dataset, with (Yang et al., 2017) reporting 371 MFLOPs for AlexNet on ImageNet, (Choi et al., 2019) reporting 724 MFLOPs, and (Han et al., 2015) reporting 1500 MFLOPs.

6 Summary and Recommendations

In the previous sections, we have argued that existing work tends to

- make it difficult to identify the exact experimental setup and metrics,
- use too few (dataset, architecture) combinations,
- report too few points in the tradeoff curve for any given combination, and no measures of central tendency,
- omit comparison to many methods that might be state-of-the-art, and
- fail to control for confounding variables.

These problems often make it difficult or impossible to assess the relative efficacy of different pruning methods. To enable direct comparison between methods in the future, we suggest the following practices:

- Identify the *exact* sets of architectures, datasets, and metrics used, ideally in a structured way that is not scattered throughout the results section.
- Use at least three (dataset, architecture) pairs, including modern, large-scale ones. MNIST and toy models do not count. AlexNet, CaffeNet, and Lenet-5 are no longer modern architectures.
- For any given pruned model, report both compression ratio and theoretical speedup. Compression ratio is defined as the original size divided by the new size. Theoretical speedup is defined as the original number of multiply-adds divided by the new number. Note that there is no reason to report only one of these metrics.
- For ImageNet and other many-class datasets, report both Top-1 and Top-5 accuracy. There is again no reason to report only one of these.
- Whatever metrics one reports for a given pruned model, also report these metrics for an appropriate control (usually the original model before pruning).
- Plot the tradeoff curve for a given dataset and architecture, alongside the curves for competing methods.
- When plotting tradeoff curves, use at least 5 operating points spanning a range of compression ratios. The set of ratios $\{2, 4, 8, 16, 32\}$ is a good choice.

- Report and plot means and sample standard deviations, instead of one-off measurements, whenever feasible.
- Ensure that all methods being compared use identical libraries, data loading, and other code to the greatest extent possible.

We also recommend that reviewers demand a much greater level of rigor when evaluating papers that claim to offer a better method of pruning neural networks.

7 SHRINKBENCH

7.1 Overview of ShrinkBench

To make it as easy as possible for researchers to put our suggestions into practice, we have created an open-source library for pruning called ShrinkBench. ShrinkBench provides standardized and extensible functionality for training, pruning, fine-tuning, computing metrics, and plotting, all using a standardized set of pretrained models and datasets.

ShrinkBench is based on PyTorch (Paszke et al., 2017) and is designed to allow easy evaluation of methods with arbitrary scoring functions, allocation of pruning across layers, and sparsity structures. In particular, given a callback defining how to compute masks for a model's parameter tensors at a given iteration, ShrinkBench will automatically apply the pruning, update the network according to a standard training or fine-tuning setup, and compute metrics across many models, datasets, random seeds, and levels of pruning. We defer discussion of ShrinkBench's implementation and API to the project's documentation.

7.2 Baselines

We used ShrinkBench to implement several existing pruning heuristics, both as examples of how to use our library and as baselines that new methods can compare to:

- **Global Magnitude Pruning** - prunes the weights with the lowest absolute value anywhere in the network.
- **Layerwise Magnitude Pruning** - for each layer, prunes the weights with the lowest absolute value.
- **Global Gradient Magnitude Pruning** - prunes 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 the weights the lowest absolute value of (weight $\times$ gradient), evaluated on a batch of inputs.
- **Random Pruning** - prunes each weight independently with probability equal to the fraction of the network to be pruned.

Magnitude-based approaches are common baselines in the literature and have been shown to be competitive with more complex methods (Han et al., 2015; 2016; Gale et al., 2019; Frankle et al., 2019). Gradient-based methods are less common, but are simple to implement and have recently gained popularity (Lee et al., 2019b;a; Yu et al., 2018). Random pruning is a common straw man that can serve as a useful debugging tool. Note that these baselines are not reproductions of any of these methods, but merely inspired by their pruning heuristics.

7.3 Avoiding Pruning Pitfalls with Shrinkbench

Using the described baselines, we pruned over 800 networks with 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 are avoided by using ShrinkBench.

We highlight several noteworthy results below. For additional experimental results and details, see Appendix D. One standard deviation bars across three runs are shown for all CIFAR-10 results.

Metrics are not Interchangeable. As discussed previously, it is common practice to report either reduction in the number of parameters or in the number of FLOPs. If these metrics are extremely correlated, reporting only one is sufficient to characterize the efficacy of a pruning method. We found after computing these metrics for the same model under many different settings that reporting one metric is not sufficient. While these metrics are correlated, the correlation is different for each pruning method. Thus, the relative performance of different methods can vary significantly under different metrics (Figure 6).

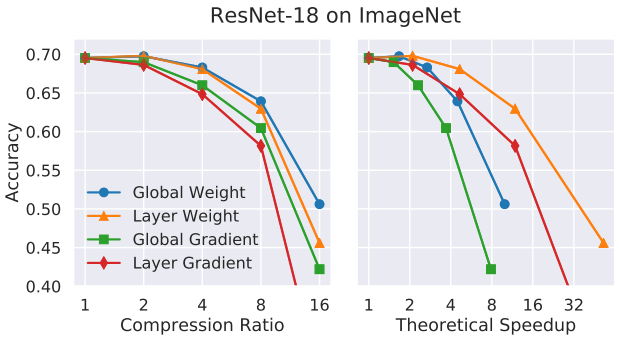

Figure 6: Top 1 Accuracy for ResNet-18 on ImageNet for several compression ratios and their corresponding theoretical speedups. Global methods give higher accuracy than Layerwise ones for a fixed model size, but the reverse is true for a fixed theoretical speedup.

Results Vary Across Models, Datasets, and Pruning Amounts Many methods report results on only a small number of datasets, models, amounts of pruning, and random seeds. If the relative performance of different methods tends to be constant across all of these variables, this may

not be problematic. However, our results suggest that this performance is not constant.

Figure 7 shows the accuracy for various compression ratios for CIFAR-VGG (Zagoruyko, 2015) and ResNet-56 on CIFAR-10. In general, Global methods are more accurate than Layerwise methods and Magnitude-based methods are more accurate than Gradient-based methods, with random performing worst of all. However, if one were to look only at CIFAR-VGG for compression ratios smaller than 10, one could conclude that Global Gradient outperforms all other methods. Similarly, while Global Gradient consistently outperforms Layerwise Magnitude on CIFAR-VGG, the opposite holds on ResNet-56 (i.e., the orange and green lines switch places).

Moreover, we found that for some settings close to the drop-off point (such as Global Gradient, compression 16), different random seeds yielded significantly different results (0.88 vs 0.61 accuracy) due to the randomness in minibatch selection. This is illustrated by the large vertical error bar in the left subplot.

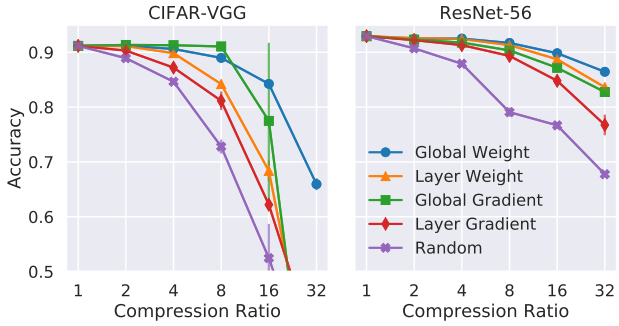

Figure 7: Top 1 Accuracy on CIFAR-10 for several compression ratios. Global Gradient performs better than Global Magnitude for CIFAR-VGG on low compression ratios, but worse otherwise. Global Gradient is consistently better than Layerwise Magnitude on CIFAR-VGG, but consistently worse on ResNet-56.

Using the Same Initial Model is Essential. As mentioned in Section 4.5, many methods are evaluated using different initial models with the same architecture. To assess whether beginning with a different model can skew the results, we created two different models and evaluated Global vs Layerwise Magnitude pruning on each with all other variables held constant.

To obtain the models, we trained two ResNet-56 networks using Adam until convergence with $\eta = 10^{-3}$ and $\eta = 10^{-4}$. We'll refer to these pretrained weights as Weights A and Weights B, respectively. As shown on the left side of Figure 8, the different methods appear better on different models. With Weights A, the methods yield similar absolute accuracies. With Weights B, however, the Global method is more accurate at higher compression ratios.

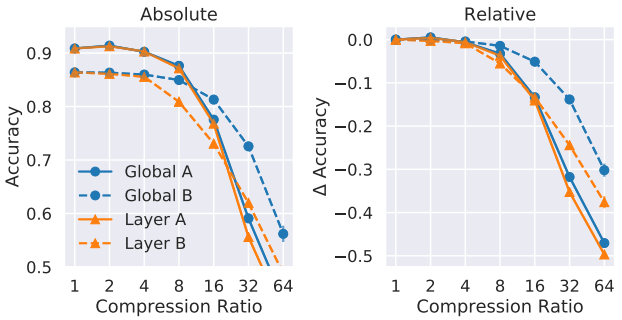

Figure 8: Global and Layerwise Magnitude Pruning on two different ResNet-56 models. Even with all other variables held constant, different initial models yield different tradeoff curves. This may cause one method to erroneously appear better than another. Controlling for initial accuracy does not fix this.

We also found that the common practice of examining changes in accuracy is insufficient to correct for initial model as a confounder. Even when reporting changes, one pruning method can artificially appear better than another by virtue of beginning with a different model. We see this on the right side of Figure 8, where Layerwise Magnitude with Weights B appears to outperform Global Magnitude with Weights A, even though the former never outperforms the latter when initial model is held constant.

8 Conclusion

Considering the enormous interest in neural network pruning over the past decade, it seems natural to ask simple questions about the relative efficacy of different pruning techniques. Although a few basic findings are shared across the literature, missing baselines and inconsistent experimental settings make it impossible to assess the state of the art or confidently compare the dozens of techniques proposed in recent years. After carefully studying the literature and enumerating numerous areas of incomparability and confusion, we suggest concrete remedies in the form of a list of best practices and an open-source library—ShrinkBench—to help future research endeavors to produce the kinds of results that will harmonize the literature and make our motivating questions easier to answer. Furthermore, ShrinkBench results on various pruning techniques evidence the need for standardized experiments when evaluating neural network pruning methods.

Acknowledgements

We thank Luigi Celona for providing the data used in (Bianco et al., 2018) and Vivienne Sze for helpful discussion. This research was supported by the Qualcomm Innovation Fellowship, the "la Caixa" Foundation Fellowship, Quanta Computer, and Wistron Corporation.

REFERENCES

What's the advantage of the reference caffenet in comparison with the alexnet? https://github.com/BVLC/caffe/issues/4202, 5 2016. Accessed: 2019-07-22.

Keras exported model shows very low accuracy in tensorflow serving. https://github.com/keras-team/keras/issues/7848, 9 2017. Accessed: 2019-07-22.

Bianco, S., Cadene, R., Celona, L., and Napoletano, P. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.

Blalock, D., Madden, S., and Guttag, J. Sprintz: Time series compression for the internet of things. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):93, 2018.

Changpinyo, S., Sandler, M., and Zhmoginov, A. The power of sparsity in convolutional neural networks. *arXiv preprint arXiv:1702.06257*, 2017.

Choi, Y., El-Khamy, M., and Lee, J. Jointly sparse convolutional neural networks in dual spatial-winograd domains. *arXiv preprint arXiv:1902.08192*, 2019.

Crall, J. Accuracy of resnet50 is much higher than reported! https://github.com/kuangliu/pytorch-cifar/issues/45, 2018. Accessed: 2019-07-22.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Ding, X., Ding, G., Han, J., and Tang, S. Auto-balanced filter pruning for efficient convolutional neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Dong, X., Huang, J., Yang, Y., and Yan, S. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5840–5848, 2017.

Dubey, A., Chatterjee, M., and Ahuja, N. Coreset-based neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 454–470, 2018.

Figurnov, M., Ibraimova, A., Vetrov, D. P., and Kohli, P. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems*, pp. 947–955, 2016.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. The lottery ticket hypothesis at scale. *arXiv preprint arXiv:1903.01611*, 2019.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks, 2019.

Gray, S., Radford, A., and Kingma, D. P. Gpu kernels for block-sparse weights. *arXiv preprint arXiv:1711.09224*, 2017.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1510.00149.

Hassibi, B., Stork, D. G., and Wolff, G. J. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018a.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European*

Conference on Computer Vision (ECCV), pp. 784–800, 2018b.

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V., and Chen, Z. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*, 2018.

Huang, Z. and Wang, N. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 304–320, 2018.

Janowsky, S. A. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600–6603, June 1989. ISSN 0556-2791. doi: 10.1103/PhysRevA.39.6600. URL https://link.aps.org/doi/10.1103/PhysRevA.39.6600.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. lenet. https://github.com/BVLC/caffe/blob/master/examples/mnist/lenet.prototxt, 2 2015a.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. lenet-train-test. https://github.com/BVLC/caffe/blob/master/examples/mnist/lenet_train_test.prototxt, 2 2015b.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Training lenet on mnist with caffe. https://caffe.berkeleyvision.org/gathered/examples/mnist.html, 5 2016. Accessed: 2019-07-22.

Jogeshwar, A. Validating resnet50. https://github.com/keras-team/keras/issues/8672, 12 2017. Accessed: 2019-07-22.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A. v. d., Dieleman, S., and Kavukcuoglu, K. Efficient neural audio synthesis. *arXiv preprint arXiv:1802.08435*, 2018.

Karnin, E. D. A simple procedure for pruning back-propagation trained neural networks. *IEEE transactions on neural networks*, 1(2):239–242, 1990.

Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L., and Shin, D. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

Lebedev, V. and Lempitsky, V. Fast convnets using group-wise brain damage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2554–2564, 2016.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998a.

LeCun, Y., Cortes, C., and Burges, C. The mnist database of handwritten digits, 1998b. Accessed: 2019-09-6.

Lee, N., Ajanthan, T., Gould, S., and Torr, P. H. S. A Signal Propagation Perspective for Pruning Neural Networks at Initialization. *arXiv:1906.06307 [cs, stat]*, June 2019a. URL http://arxiv.org/abs/1906.06307. arXiv: 1906.06307.

Lee, N., Ajanthan, T., and Torr, P. H. S. Snip: single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL https://openreview.net/forum?id=B1VZqjAcYX.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Lindstrom, P. Fixed-rate compressed floating-point arrays. *IEEE transactions on visualization and computer graphics*, 20(12):2674–2683, 2014.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJlnB3C5Ym.

Louizos, C., Ullrich, K., and Welling, M. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pp. 3288–3298, 2017.

Luo, J.-H., Wu, J., and Lin, W. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

Mariet, Z. and Sra, S. Diversity networks: Neural network compression using determinantal point processes. *arXiv preprint arXiv:1511.05077*, 2015.

Molchanov, D., Ashukha, A., and Vetrov, D. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2498–2507. JMLR. org, 2017.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Morcos, A. S., Yu, H., Paganini, M., and Tian, Y. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *arXiv:1906.02773 [cs, stat]*, June 2019. URL http://arxiv.org/abs/1906.02773. arXiv: 1906.02773.

Mozer, M. C. and Smolensky, P. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pp. 107–115, 1989a.

Mozer, M. C. and Smolensky, P. Using Relevance to Reduce Network Size Automatically. *Connection Science*, 1(1):3–16, January 1989b. ISSN 0954-0091, 1360-0494. doi: 10.1080/09540098908915626. URL https://www.tandfonline.com/doi/full/10.1080/09540098908915626.

Nola, D. Keras doesn't reproduce caffe example code accuracy. https://github.com/keras-team/keras/issues/4444, 11 2016. Accessed: 2019-07-22.

Northcutt, C. Towards reproducibility: Benchmarking keras and pytorch. https://l7.curtisnorthcutt.com/towards-reproducibility-benchmarking-keras-pytorch, 2 2019. Accessed: 2019-07-22.

Parkhi, O. M., Vedaldi, A., Zisserman, A., et al. Deep face recognition. In *bmvc*, volume 1, pp. 6, 2015.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Peng, B., Tan, W., Li, Z., Zhang, S., Xie, D., and Pu, S. Extreme network compression via filter group approximation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 300–316, 2018.

Ratanaworabhan, P., Ke, J., and Burtscher, M. Fast lossless compression of scientific floating-point data. In *Data Compression Conference (DCC'06)*, pp. 133–142. IEEE, 2006.

Reed, R. Pruning algorithms-a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, September 1993. ISSN 10459227. doi: 10.1109/72.248452. URL http://ieeexplore.ieee.org/document/248452/.

Siedelmann, H., Wender, A., and Fuchs, M. High speed lossless image compression. In *German Conference on Pattern Recognition*, pp. 343–355. Springer, 2015.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Suau, X., Zappella, L., and Apostoloff, N. Network compression using correlation analysis of layer responses. 2018.

Suzuki, T., Abe, H., Murata, T., Horiuchi, S., Ito, K., Wachi, T., Hirai, S., Yukishima, M., and Nishimura, T. Spectral-pruning: Compressing deep neural network via spectral analysis. *arXiv preprint arXiv:1808.08558*, 2018.

Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. Efficient processing of deep neural networks: A tutorial and survey. *arXiv preprint arXiv:1703.09039*, 2017.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Tresp, V., Neuneier, R., and Zimmermann, H.-G. Early brain damage. In *Advances in neural information processing systems*, pp. 669–675, 1997.

Vryniotis, V. Change bn layer to use moving mean/var if frozen. https://github.com/keras-team/keras/pull/9965, 4 2018. Accessed: 2019-07-22.

Wang, P. and Cheng, J. Accelerating convolutional neural networks for mobile applications. In *Proceedings of the 24th ACM international conference on Multimedia*, pp. 541–545. ACM, 2016.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pp. 2074–2082, 2016.

Yamamoto, K. and Maeno, K. Pcas: Pruning channels with attention statistics. *arXiv preprint arXiv:1806.05382*, 2018.

Yang, T.-J., Chen, Y.-H., and Sze, V. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5687–5695, 2017.

Yao, Z., Cao, S., and Xiao, W. Balanced sparsity for efficient dnn inference on gpu. *arXiv preprint arXiv:1811.00206*, 2018.

Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., Gao, M., Lin, C.-Y., and Davis, L. S. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, 2018.

Zagoruyko, S. 92.45% on cifar-10 in torch. https://torch.ch/blog/2015/07/30/cifar.html, 7 2015. Accessed: 2019-07-22.

Zhang, X., Zou, J., He, K., and Sun, J. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2015.

Zhao, W. X., Zhang, X., Lemire, D., Shan, D., Nie, J.-Y., Yan, H., and Wen, J.-R. A general simd-based approach to accelerating compression algorithms. *ACM Transactions on Information Systems (TOIS)*, 33(3):15, 2015.

Zukowski, M., Heman, S., Nes, N., and Boncz, P. Super-scalar ram-cpu cache compression. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pp. 59–59. IEEE, 2006.

A Corpus and Data Cleaning

We selected the 81 papers used in our analysis in the following way. First, we conducted an ad hoc literature search, finding widely cited papers introducing pruning methods and identifying other pruning papers that cited them using Google Scholar. We then went through the conference proceedings from the past year's NeurIPS, ICML, CVPR, ECCV, and ICLR and added all relevant papers (though it is possible we had false dismissals if the title and abstract did not seem relevant to pruning). Finally, during the course of cataloging which papers compared to which others, we added to our corpus any pruning paper that at least one existing paper in our corpus purported to compare to. We included both published papers and unpublished ones of reasonable quality (typically on arXiv). Since we make strong claims about the lack of comparisons, we included in our corpus five papers whose methods technically do not meet our definition of pruning but are similar in spirit and compared to by various pruning papers. In short, we included essentially every paper introducing a method of pruning neural networks that we could find, taking care to capture the full directed graph of papers and comparisons between them.

Because different papers report slightly different metrics, particularly with respect to model size, we converted reported results to a standard set of metrics whenever possible. For example, we converted reported Top-1 error rates to Top-1 accuracies, and fractions of parameters pruned to compression ratios. Note that it is not possible to convert between size metrics and speedup metrics, since the amount of computation associated with a given parameter can depend on the layer in which it resides (since convolutional filters are reused at many spatial positions). For simplicity and uniformity, we only consider self-reported results except where stated otherwise.

We also did not attempt to capture all reported metrics, but instead focused only on model size reduction and theoretical speedup, since 1) these are by far the most commonly reported and, 2) there is already a dearth of directly comparable numbers even for these common metrics. This is not entirely fair to methods designed to optimize other metrics, such as power consumption (Louizos et al., 2017; Yang et al., 2017; Han et al., 2015; Kim et al., 2015), memory bandwidth usage (Peng et al., 2018; Kim et al., 2015), or fine-tuning time (Dubey et al., 2018; Yamamoto & Maeno, 2018; Huang & Wang, 2018; He et al., 2018a), and we consider this a limitation of our analysis.

Lastly, as a result of relying on reading of hundreds of pages of dense technical content, we are confident that we have made some number of isolated errors. We therefore welcome correction by email and refer the reader to the arXiv version of this paper for the most up-to-date revision.

B Checklist for Evaluating a Pruning Method

For any pruning technique proposed, check if:

- It is contextualized with respect to magnitude pruning, recently-published pruning techniques, and pruning techniques proposed prior to the 2010s.
- The pruning algorithm, constituent subroutines (e.g., score, pruning, and fine-tuning functions), and hyperparameters are presented in enough detail for a reader to reimplement and match the results in the paper.
- All claims about the technique are appropriately restricted to only the experiments presented (e.g., CIFAR-10, ResNets, image classification tasks, etc.).
- There is a link to downloadable source code.

For all experiments, check if you include:

- A detailed description of the architecture with hyperparameters in enough detail to for a reader to reimplement it and train it to the same performance reported in the paper.
- If the architecture is not novel: a citation for the architecture/hyperparameters and a description of any differences in architecture, hyperparameters, or performance in this paper.
- A detailed description of the dataset hyperparameters (e.g., batch size and augmentation regime) in enough detail for a reader to reimplement it.
- A description of the library and hardware used.

For all results, check if:

- Data is presented across a range of compression ratios, including extreme compression ratios at which the accuracy of the pruned network declines substantially.
- Data specifies the raw accuracy of the network at each point.
- Data includes multiple runs with separate initializations and random seeds.
- Data includes clearly defined error bars and a measure of central tendency (e.g., mean) and variation (e.g., standard deviation).
- Data includes FLOP-counts if the paper makes arguments about efficiency and performance due to pruning.

For all pruning results presented, check if there is a comparison to:

- A random pruning baseline.
  - A global random pruning baseline.
  - A random pruning baseline with the same layerwise pruning proportions as the proposed technique.
- A magnitude pruning baseline.
  - A global or uniform layerwise proportion magnitude pruning baseline.
  - A magnitude pruning baseline with the same layerwise pruning proportions as the proposed technique.
- Other relevant state-of-the-art techniques, including:
  - A description of how the comparisons were produced (data taken from paper, reimplementation, or reuse of code from the paper) and any differences or uncertainties between this setting and the setting used in the main experiments.

C EXPERIMENTAL SETUP

For reproducibility purposes, ShrinkBench fixes random seeds for all the dependencies (`PyTorch`, `NumPy`, `Python`).

C.1 Pruning Methods

For the reported experiments, we did not prune the classifier layer preceding the softmax. ShrinkBench supports pruning said layer as an option to all proposed pruning strategies. For both Global and Layerwise Gradient Magnitude Pruning a single minibatch is used to compute the gradients for the pruning. Three independent runs using different random seeds were performed for every CIFAR10 experiment. We found some variance across methods that relied on randomness, such as random pruning or gradient based methods that use a sampled minibatch to compute the gradients with respect to the weights.

C.2 Finetuning Setup

Pruning was performed from the pretrained weights and fixed from there forwards. Early stopping is implemented during finetuning. Thus if the validation accuracy repeatedly decreases after some point we stop the finetuning process to prevent overfitting.

All reported CIFAR10 experiments used the following finetuning setup:

- Batch size: 64
- Epochs: 30
- Optimizer: Adam
- Initial Learning Rate: 3×10^{-4}
- Learning rate schedule: Fixed

All reported ImageNet experiments used the following finetuning setup

- Batch size: 256
- Epochs: 20
- Optimizer: SGD with Nesterov Momentum (0.9)
- Initial Learning Rate: 1×10^{-3}
- Learning rate schedule: Fixed

D ADDITIONAL RESULTS

Here we include the entire set of results obtained with ShrinkBench. For CIFAR10, results are included for CIFAR-VGG, ResNet-20, ResNet-56 and ResNet-110. Standard deviations across three different random runs are plotted as error bars. For ImageNet, results are reported for ResNet-18.

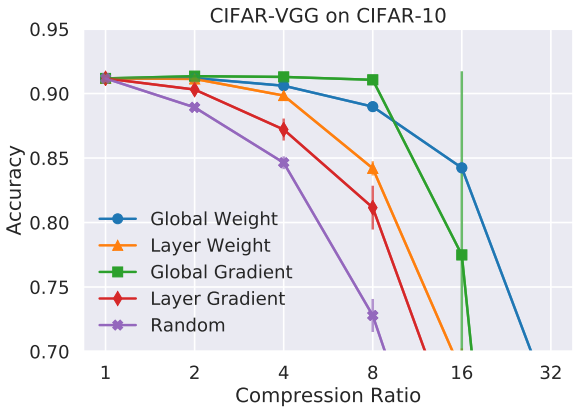

Figure 9: Accuracy for several levels of compression for CIFAR-VGG on CIFAR-10

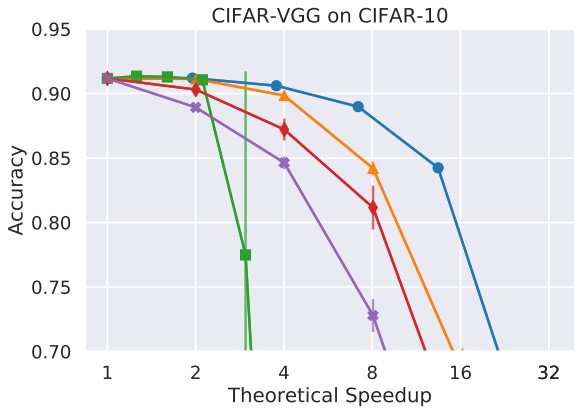

Figure 10: Accuracy vs theoretical speedup for CIFAR-VGG on CIFAR-10

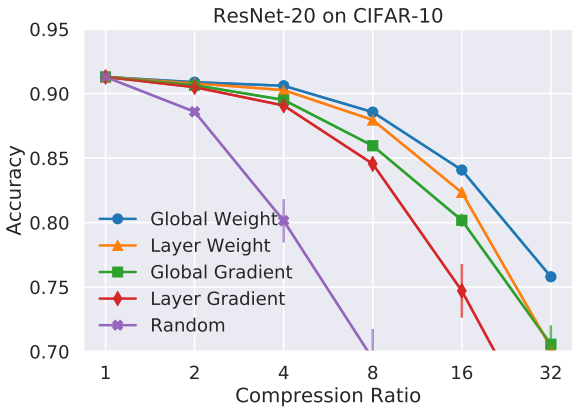

Figure 11: Accuracy for several levels of compression for ResNet-20 on CIFAR-10

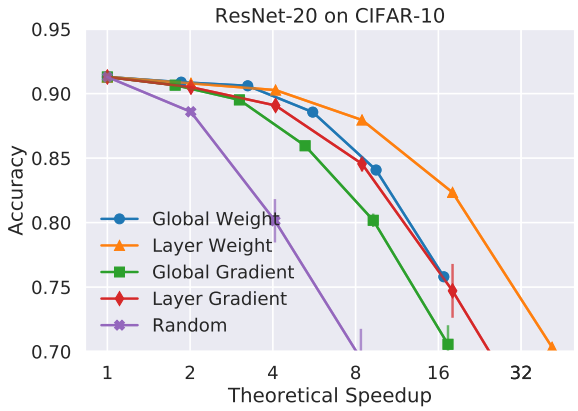

Figure 12: Accuracy vs theoretical speedup for ResNet-20 on CIFAR-10

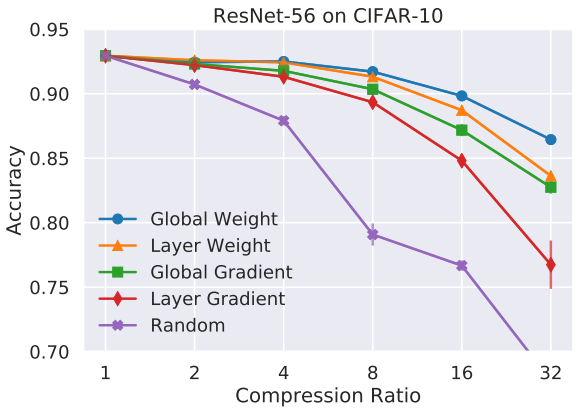

Figure 13: Accuracy for several levels of compression for ResNet-56 on CIFAR-10

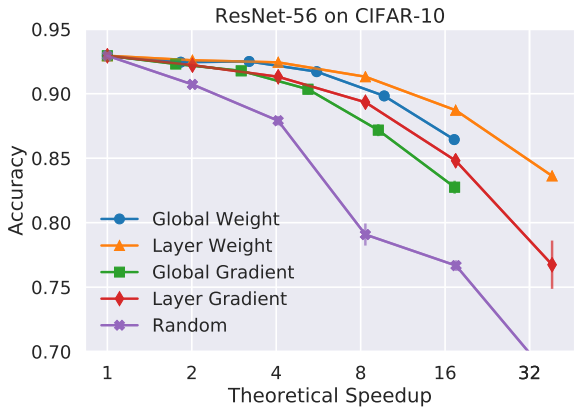

Figure 14: Accuracy vs theoretical speedup for ResNet-56 on CIFAR-10

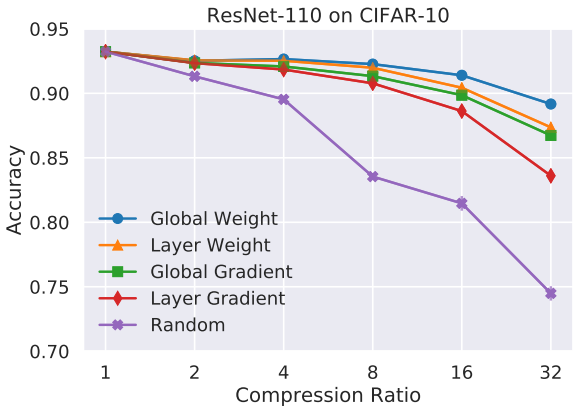

Figure 15: Accuracy for several levels of compression for ResNet-110 on CIFAR-10

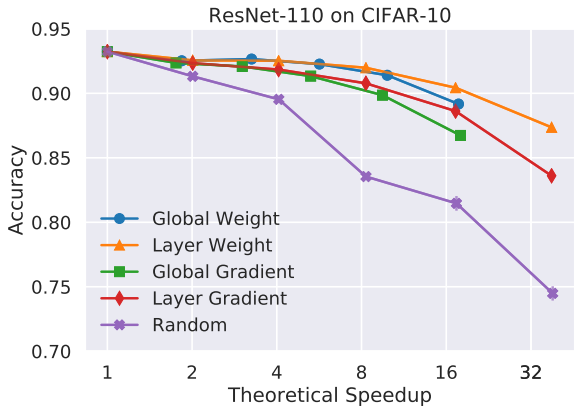

Figure 16: Accuracy vs theoretical speedup for ResNet-110 on CIFAR-10

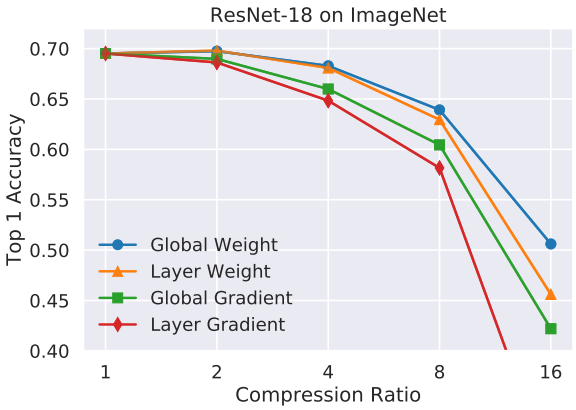

Figure 17: Accuracy for several levels of compression for ResNet-18 on ImageNet

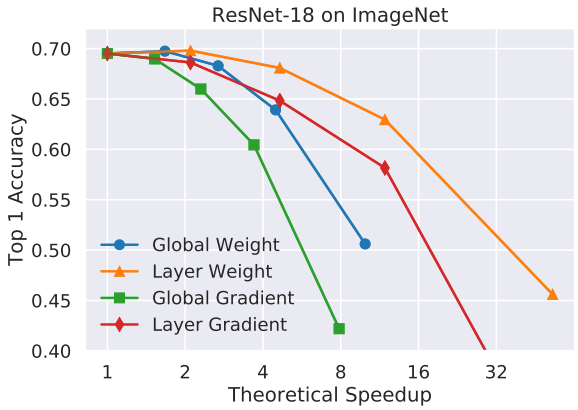

Figure 18: Accuracy vs theoretical speedup for ResNet-18 on ImageNet