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 We included both published papers and unpublished ones of reasonable quality (typically on arXiv).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., 2015b; 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., 2018), and we consider this a limitation of our analysis.

A further limitation is that, 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 clsasification 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 proprtion 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 used random seeds were performed for every CIFAR10 experiments. We found some variance across methods that relied in 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. Reported values are always for the validation set. 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 VGG-CIFAR, ResNet20, ResNet56 and ResNet110. Standard deviation across three different random runs is reported. For ImageNet results are reported for ResNet18.

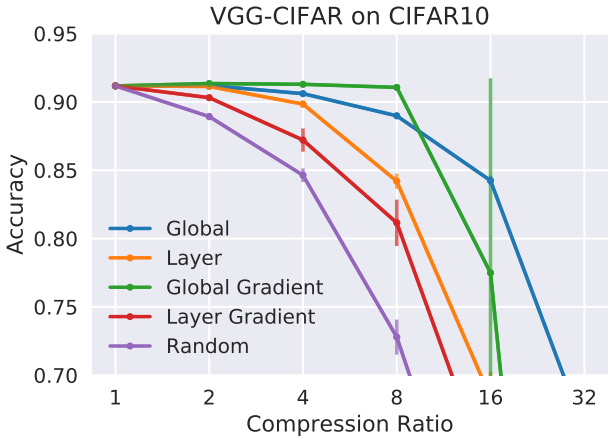

Figure 9: Accuracy for several levels of compression for VGG-CIFAR on CIFAR10

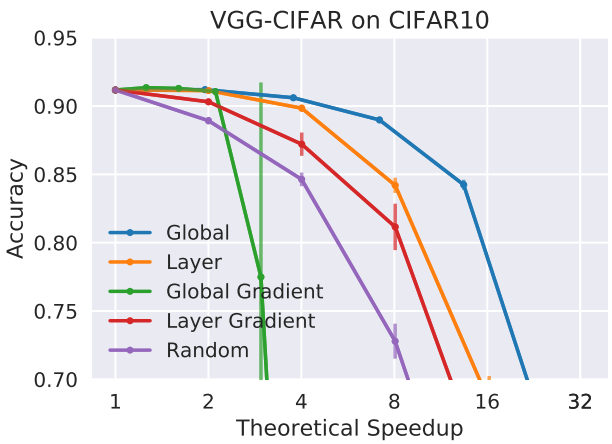

Figure 10: Accuracy vs theoretical speedup for VGG-CIFAR on CIFAR10

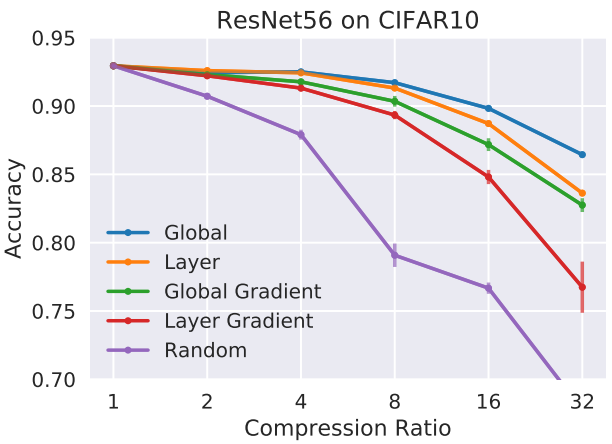

Figure 13: Accuracy for several levels of compression for ResNet56 on CIFAR10

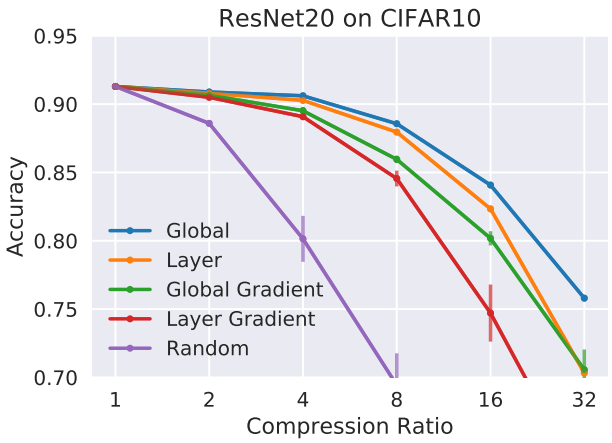

Figure 11: Accuracy for several levels of compression for ResNet20 on CIFAR10

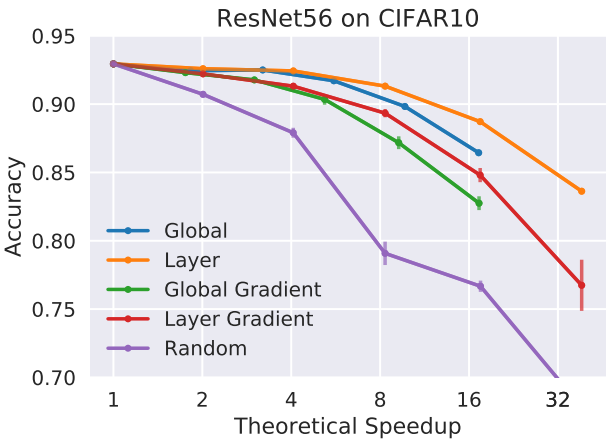

Figure 14: Accuracy vs theoretical speedup for ResNet56 on CIFAR10

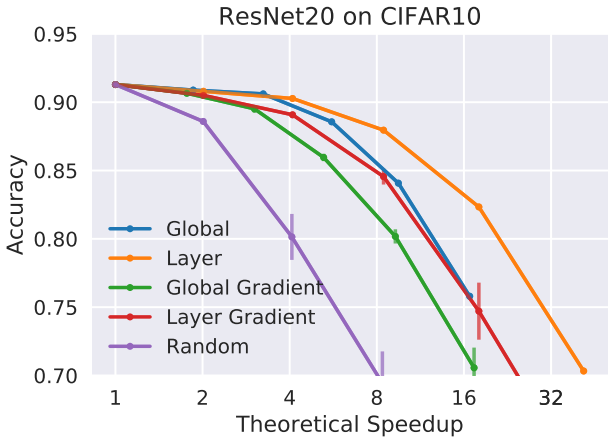

Figure 12: Accuracy vs theoretical speedup for ResNet20 on CIFAR10

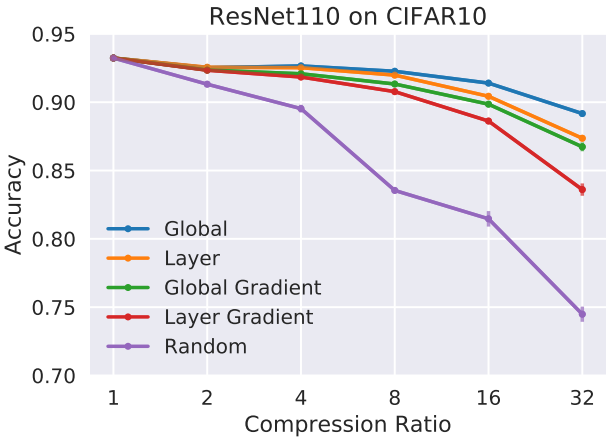

Figure 15: Accuracy for several levels of compression for ResNet110 on CIFAR10

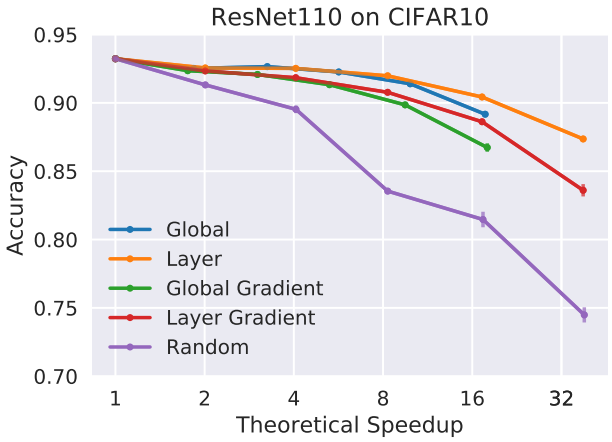

Figure 16: Accuracy vs theoretical speedup for ResNet110 on CIFAR10

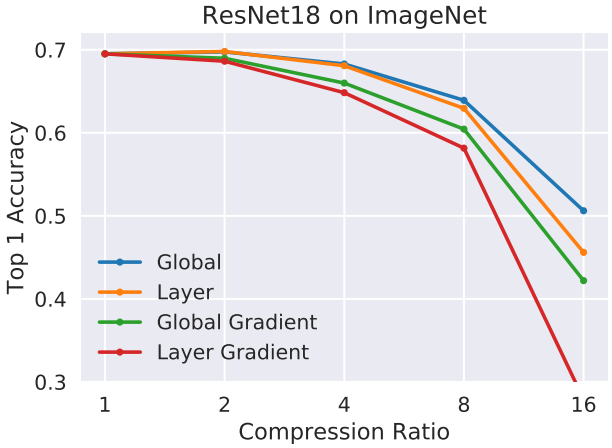

Figure 17: Accuracy for several levels of compression for ResNet110 on CIFAR10

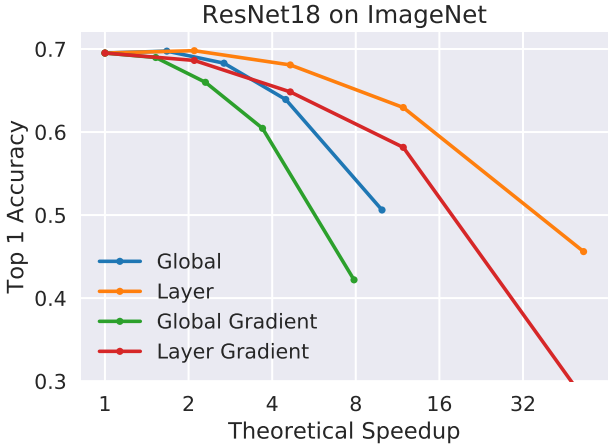

Figure 18: Accuracy vs theoretical speedup for ResNet110 on CIFAR10