The Generalization-Stability Tradeoff in Neural Network Pruning

Brian Bartoldson[1], **Adrian Barbu**[2], **Ari Morcos**[3], **Gordon Erlebacher**[1]

{[1]Department of Scientific Computing, [2]Department of Statistics} at Florida State University, [3]Facebook AI Research

Introduction

Pruning deep neural network (DNN) parameters to reduce memory/computation requirements is an area of much interest, but a variety of pruning approaches also *increase* generalization (accuracy on unobserved data). Knowing how pruning improves generalization could lead to better pruning algorithms, and a better understanding of the factors affecting generalization. Traditionally, pruning was thought to prevent overfitting to the data used to train the model by reducing the number of parameters. However, such an explanation cannot account for the results in Figure 1, which shows different pruning algorithms affecting overfitting behavior differently despite pruning the same number of parameters.

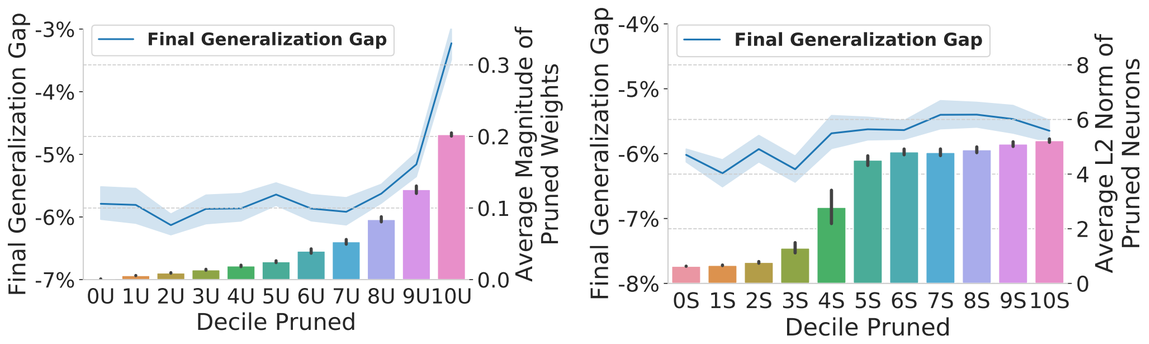
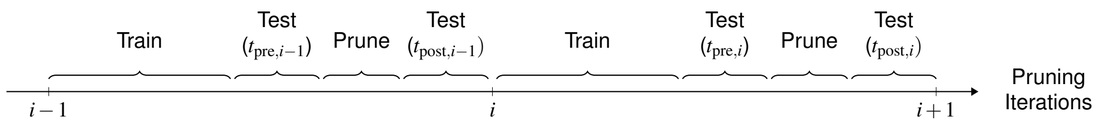


Figure 1: When pruning 10% of Conv4's largest dense layer, the generalization gap depends on the magnitude of the weights that were pruned during training, and the use of structured pruning (S vs. U).

Furthermore, recent empirical studies and generalization bounds suggest that more parameters may actually be better for generalization (Neyshabur et al., 2014, 2018). These results raise a puzzling question: if larger parameter counts help generalization, how does pruning increase performance?

Approach

We address this question by studying an aspect of pruning that is unrelated to parameter counts. Specifically, we consider pruning algorithm instability, which we define on pruning iteration i as $\text{instability}_i = t_{\text{pre},i} - t_{\text{post},i}$, where t is the test-dataset accuracy.



In the context of three distinct models (ResNet18, VGG11, and Conv4), we examine how generalization changes in response to changes in pruning instability, which we modify by varying pruning algorithm facets that are unrelated to total parameters pruned.

- **Pruning Algorithm Facets:** *Target*: small- ("S"), random- ("R"), or large- ("L") magnitude parameters. *Iterative pruning rate*: fraction of parameters removed on each pruning iteration.

In all experiments, the DNN's cross-entropy loss on CIFAR-10 training data is optimized with Adam. In all plots, data points are averages of 10-20 runs, from which 95% confidence error bars are bootstrapped.

Pruning Batch-Normalized Filters

Consider the 2D feature map R produced by convolution with batch normalization (BN) and a ReLU activation function:

$$R_{ij} = \max\{0, M_{ij}\}, \quad \text{where} \quad M = \gamma \text{BN}(W * x) + \beta$$

Batch normalization obscures the relationship between filter (W) and output (R) magnitude, making it difficult to predict the instability associated with pruning traditional pruning targets (e.g., pruning the filter with the smallest magnitude). To address this we define the "E[BN]" magnitude associated with W as $\mathbb{E}[R_{ij}]$, which we approximate under the assumption that $M_{ij} \sim \mathcal{N}(\beta, \gamma)$ (justified by the central limit theorem when the products in $W * x$ are i.i.d. and sufficiently numerous; see paper for empirical support). Our assumption ensures that the R_{ij} are either 0 or samples from a truncated normal distribution with left truncation point $l = 0$, right truncation point $r = \infty$, and mean μ where

$$\mu = \gamma \frac{\phi(\lambda) - \phi(\rho)}{Z} + \beta, \text{ and where } \lambda = \frac{l - \beta}{\gamma}, \ \rho = \frac{r - \beta}{\gamma}, \ Z = \Phi(\rho) - \Phi(\lambda),$$

and $\phi(x)$ and $\Phi(x)$ are the standard normal distribution's PDF and CDF (respectively) evaluated at x. Thus, we can simply approximate $\mathbb{E}[R_{ij}]$:

$$\mathbb{E}[R_{ij}] \approx \Phi(\lambda)0 + (1 - \Phi(\lambda))\mu.$$

Pruning Instability during Learning Improves DNN Generalization

Pruning improves generalization most when it is unstable. For example, targeting more important parameters leads to particularly higher instability and generalization

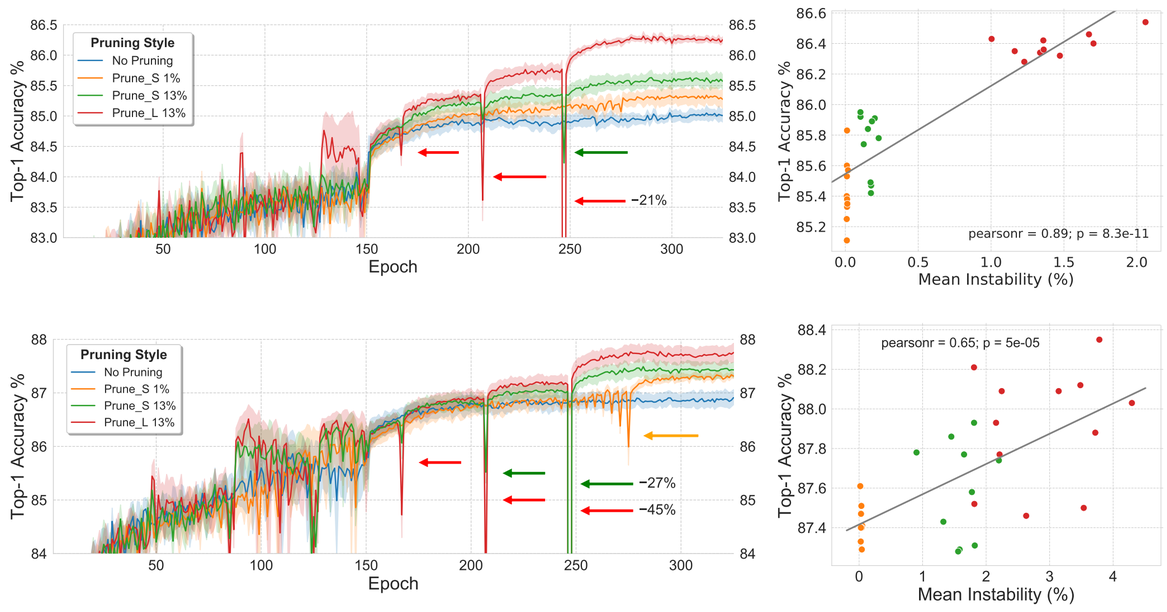


Figure 2: Pruning instability improves generalization of (Top) VGG11 and (Bottom) ResNet18 when training on CIFAR-10 (10 runs per configuration). (Left) Test accuracy during training of several models illustrates how adaptation to more unstable pruning leads to better generalization. (Right) Means reduce along the epoch dimension (creating one point per run-configuration combination).

Given Pruning Target, Iterative Rate Affects Instability/Generalization

Iterative pruning rate also affects instability and generalization, meaning that pruning algorithms with the same target do not necessarily affect generalization similarly. This could help explain how typical pruning studies, which tend to have the same target (small-magnitude parameters), report various generalization benefits from pruning.

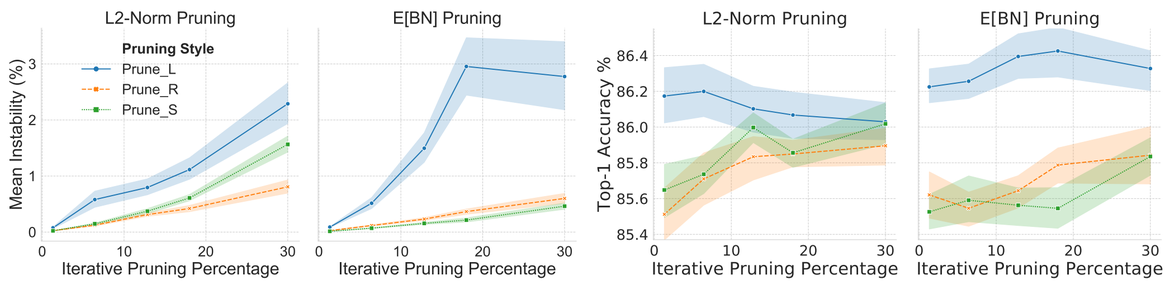


Figure 3: In VGG11, increasing the iterative pruning rate (and decreasing the number of pruning events in order to hold total pruning percentage constant) leads to more instability, and can allow methods that target less important parameters to generalize better. Additionally, E[BN] magnitude better approximates parameter importance than ℓ_2-norm magnitude. An unpruned baseline model had 85.21% average accuracy.

Pruning Acts Like Noise Injection

We have shown that pruning regularizes through an instability-associated mechanism. Since pruning can be thought of as noise injection, we hypothesize that the instability level determines the amount of noise, and therefore noise-based regularization, applied to the network. Figure 4 shows that replacing pruning with temporary noise injection creates generalization dynamics similar to pruning's. In sum, then, our study supports the idea that pruning improves generalization via noise injection, wherein pruning instability corresponds to the level of noise, which is modifiable via pruning facets such as the use of the prune$_L$ pruning target based on the E[BN] magnitude that we derived.

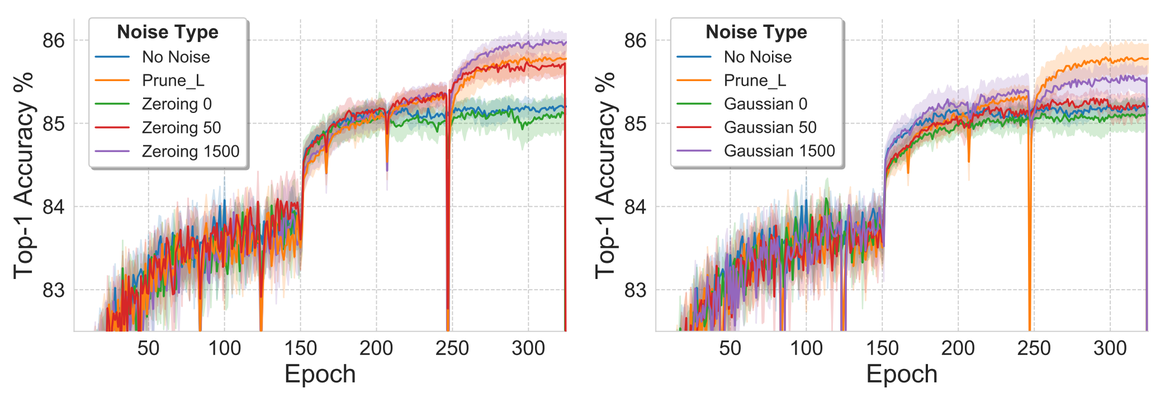


Figure 4: Generalization improvements from pruning bear resemblance to those obtained by using *temporary* (Left) multiplicative zeroing noise, and (Right) additive Gaussian noise, as long as the noise is applied for enough batches/steps.

Link to Paper